Thèse n° 9259

Understanding Deep Neural Networks using Adversarial Attacks

Présentée le 15 août 2022

Faculté informatique et communications
Laboratoire de vision par ordinateur
Programme doctoral en informatique et communications

pour l'obtention du grade de Docteur ès Sciences

par

Krishna Kanth NAKKA

Acceptée sur proposition du jury

Prof. N. H. B. Flammarion, président du jury
Prof. P. Fua, Dr M. Salzmann, directeurs de thèse
Dr J. H. Metzen, rapporteur
Prof. G. C. Buttazzo, rapporteur
Prof. P. Frossard, rapporteur

École
polytechnique
fédérale
de Lausanne

2022

To **Sri Sirivennela Seetharama Sastry Garu**

నా మనసా, వాచా, కర్మణా మీకు జీవితాంతం ఋణపడి ఉంటాను.

Acknowledgements

This thesis would not be possible without the contributions of many people either directly or indirectly. First and foremost, I would like to thank my advisor, Dr. Mathieu Salzmann, for accepting me to work with him. The impact he has had on me goes beyond research, and I am deeply inspired by his work ethic. My association with him has taught me a lot of things such as managing time efficiently and becoming more focused on things I can control. I am grateful to Prof. Pascal Fua for taking me into the CVLab and providing all the hardware resources. Moreover, I would like to thank the Swiss National Science Foundation scholarship and the EDIC fellowship for their generous funding support.

I would like to thank my Jury members: Prof. Nicolas Flammarion, Prof. Pascal Frossard, Prof. Giorgio Buttazzo, and Dr. Jan Hendrik Metzen for evaluating my thesis and providing critical feedback.

I am also grateful to my friends from the CVLab who supported me during my Ph.D. journey. Many thanks go out to Vidit, Shuxuan, Kaicheng, Weizhe, Krzystof, Sena, Isinsu, Victor, Mateusz, Ali and many others. I enjoyed having the company of my BC306 roommates, Yinlin and Jan. They invariably encouraged me and made a huge difference in my daily routine. Once again, my sincere thanks to Sena and Jan for their commitment to teaching and systematic planning of TAship duties.

I'm indebted to the institutes of Brilliant, Narayana, and Samsung R&D for providing me with the opportunity to meet some of the genius minds. Thanks to IIT Kharagpur for nurturing me and to my EE department mates and my RP hallmates for forging a deep lasting friendships. I want to express my gratitude to Nalam and Srinath for their care and concern. I'm also grateful to Dr. Shankar for hiring me as an intern at Phillips during the most tragic times.

On an emotional note, I'm immensely grateful to A.R. Rehman for bringing spirituality and an unshakable sense of peace into my life and helping me achieve balance during the most difficult of times. Thank you, ARR. Lastly, I humbly thank the Oncologists at CHUV hospital, in particular Dr. Khalil Zaman, for their care towards my wife, without which the final stages of my Ph.D. would have been extremely chaotic.

Abstract

Deep Neural Networks (DNNs) have achieved great success in a wide range of applications, such as image recognition, object detection, and semantic segmentation. Even though the discriminative power of DNNs is nowadays unquestionable, serious concerns have arised ever since DNNs have shown to be vulnerable to adversarial examples crafted by adding imperceptible perturbations to clean images. The implications of these malicious attacks are even more significant for DNNs deployed in real-world systems, e.g., autonomous driving and biometric authentication. Consequently, an intriguing question that we aim to understand is the underlying behavior of DNNs to adversarial attacks.

This thesis contributes to a better understanding of the mechanism of adversarial attacks on DNNs. Our main contributions are broadly in two directions: (1) we propose interpretable architectures first to understand the reasons for the success of adversarial attacks and then to improve the robustness of DNNs; (2) we design intuitive adversarial attacks to both mislead and use as a tool to expand our present understanding of DNNs' internal workings and their limitations.

In the first direction, we introduce deep architectures that allow humans to interpret the reasoning process of DNNs prediction. Specifically, we incorporate Bag-of-visual-words representations from the pre-deep learning era into DNNs using an attention scheme. We find key reasons for adversarial attack success and use these insights to propose an adversarial defense by maximally separating the latent features of discriminative regions while minimizing the contribution of non-discriminative regions in the final prediction.

The second direction deals with the design of adversarial attacks to understand DNNs' limitations in a real-world environment. To begin with, we show that existing state-of-the-art semantic segmentation networks that achieve superior performance by exploiting the context are highly susceptible to indirect local attacks. Furthermore, we demonstrate the existence of universal directional perturbations that are quasi-independent of the input template but still successfully fool unknown siamese-based visual object trackers. We then identify that the mid-level filter banks across different backbones bear strong similarities and thus can be potential common ground for attack. We, therefore, learn a generator that disrupts mid-level features with high transferability across different target architectures, datasets, and tasks. In short, our attacks highlight critical vulnerabilities of DNNs, which make their deployment challenging in the real-world environment, even in the extreme case when the attacker is unaware of the target architecture or the target data used to train it.

Furthermore, we go beyond fooling networks and demonstrate the usefulness of adversarial attacks for studying the internal disentangled representations in self-supervised 3D pose estimation networks. We carry out an in-depth analysis to understand to what degree disentanglement representation methods separate the appearance information from the pose one. We observe that adversarial manipulation of appearance information in the input image alters the pose output, indicating that the pose code contains appearance information. Our analyses show that disentanglement in the three state-of-the-art disentangled representation learning frameworks is far from complete, further evidenced through multiple image-synthesis experiments.

Besides the above contributions, an underlying theme that arises multiple times in this thesis is counteracting the adversarial attacks by detecting them. We achieve this in our work (1) by comparing the input image with the highest activated prototype in bag of words networks, (2) by training logistic detectors on features computed using Mahalanobis distance from the pre-trained class-conditional Gaussian distributions at multiple layers, and (3) finally by comparing the input image to an image resynthesized from the predicted label map.

Overall, this thesis presents an insightful understanding of mechanism of adversarial attacks and the limitations of DNNs, which can provide directions for attaining robust models.

Key Words: Deep Neural Networks, Adversarial Attacks, Black-box Attacks, Adversarial Defense, Image Recognition, Semantic Segmentation, Object Tracking, Disentanglement.

Résumé

Les réseaux de neurones profonds (DNN) ont remporté un grand succès dans un large éventail d'applications, telles que la reconnaissance d'images, la détection d'objets et la segmentation sémantique. Bien que le pouvoir discriminant des DNN est aujourd'hui incontestable, de sérieuses inquiétudes ont surgi depuis que les DNN se sont révélés vulnérables aux exemples contradictoires conçus en ajoutant des perturbations imperceptibles aux images nettes. Les implications de ces programmes malveillants les attaques sont encore plus importantes pour les DNN déployés dans des systèmes réels, par exemple, conduite autonome et authentification biométrique. Par conséquent, une question intrigante que nous cherchons à comprendre est le comportement sous-jacent des DNN aux attaques contradictoires.

Cette thèse contribue à une meilleure compréhension du mécanisme des attaques contradictoires sur les DNN. Nos principales contributions vont globalement dans deux directions : (1) nous proposons architectures interprétables d'abord pour comprendre les raisons du succès des attaques puis d'améliorer la robustesse des DNN ; (2) nous concevons des systèmes contradictoires intuitifs attaques à la fois trompeuses et utilisées comme un outil pour élargir notre compréhension actuelle des DNN fonctionnement interne et ses limites. Dans la première direction, nous introduisons des architectures profondes qui permettent aux humains d'interpréter les processus de raisonnement de la prédiction des DNN. Plus précisément, nous incorporons Bag-of-visual-words représentations de l'ère de l'apprentissage pré-profond dans les DNN à l'aide d'un schéma d'attention. Nous trouvons les principales raisons du succès des attaques adverses et utilisons ces informations pour proposer un défense contradictoire en séparant au maximum les caractéristiques latentes des régions discriminantes tout en minimisant la contribution des régions non discriminantes dans la prédiction nale.

La deuxième direction traite de la conception d'attaques contradictoires pour comprendre les DNN limites dans un environnement réel. Pour commencer, nous montrons que l'état actuel de la des réseaux de segmentation sémantique d'art qui atteignent des performances supérieures en exploitant le contexte sont très sensibles aux attaques locales indirectes. De plus, nous démontrons l'existence de perturbations directionnelles universelles quasi-indépendantes de la modèle d'entrée mais réussit toujours à tromper des trackers d'objets visuels inconnus basés sur le siamois. Nous identifions ensuite que les bancs de ltre de niveau intermédiaire à travers diérents backbones similitudes et peuvent donc constituer un terrain d'entente potentiel pour l'attaque. Nous apprenons donc une générateur qui perturbe les fonctionnalités de niveau intermédiaire avec une grande transférabilité sur diérentes cibles architectures, ensembles de données et tâches. En bref, nos attaques mettent en évidence des vulnérabilités critiques des DNN, ce qui rend leur déploiement difficile dans l'environnement réel, même dans le cas extrême où l'attaquant ignore l'architecture cible ou la cible données utilisées pour l'entraîner.

De plus, nous allons au-delà de la tromperie des réseaux et démontrons l'utilité de la confrontation attaques pour étudier les représentations désenchevêtrées internes en pose 3D auto-supervisée réseaux d'estimation. Nous effectuons une analyse approfondie pour comprendre dans quelle mesure les méthodes de représentation de démêlage séparent les informations d'apparence des poser un. Nous observons que la manipulation contradictoire des informations sur l'apparence dans le l'image d'entrée modifie la sortie de pose, indiquant que le code de pose contient l'apparence information. Nos analyses montrent que le démêlage dans les trois états de l'art démêlés les cadres d'apprentissage de la représentation sont loin d'être complets, à travers de multiples expériences de synthèse d'images.

Outre les contributions ci-dessus, un thème sous-jacent qui revient plusieurs fois dans ce La thèse est de contrecarrer les attaques adverses en les détectant. Nous y parvenons en notre travail en comparant l'image d'entrée avec le prototype activé le plus élevé dans BoW réseaux, en formant des détecteurs logistiques sur des caractéristiques calculées à l'aide de la distance de Mahalanobis à partir des distributions gaussiennes conditionnelles de classe pré-formées à plusieurs couches, et enfin en comparant l'image d'entrée à l'image resynthétisée à partir de l'étiquette prédite carte.

Dans l'ensemble, cette thèse présente une compréhension perspicace du mécanisme de l'opposition attaques et les limites des DNN, qui peuvent fournir des indications pour atteindre une robustesse des modèles.

Mots-clés : réseaux de neurones profonds, attaques contradictoires, attaques par boîte noire, confrontation Défense, Reconnaissance d'images, Segmentation sémantique, Suivi d'objets, Désenchevêtrement.

Publications

This thesis is largely based on following works.

Chapter 2 is based on:

- **Nakka, K. K.**, & Salzmann, M. “Deep Attentional Structured Representation Learning for Visual Recognition”, British Media Vision Conference (BMVC), 2018

Chapter 3 is based on:

- **Nakka, K. K.**, & Salzmann, M. “Interpretable BoW Networks for Adversarial Example Detection”, Explainable Artificial Intelligence Workshop, International Conference of Computer Vision (ICCV), 2019

Chapter 4 is based on:

- **Nakka, K. K.**, & Salzmann, M. “ Towards Robust Fine-grained Recognition by Maximal Separation of Discriminative Features”, Asian Conference on Computer Vision (ACCV) 2020

Chapter 5 is based on:

- **Nakka, K. K.**, & Salzmann. “Indirect local attacks for context-aware semantic segmentation networks”, European Conference on Computer Vision (ECCV 2020) [**Spotlight**]
- Lis, K., **Nakka, K. K.**, Fua, P. and Salzmann, M. “Detecting the unexpected via image resynthesis”, International Conference of Computer Vision (ICCV), 2019

Chapter 6 is based on:

- **Nakka, K. K.**, & Salzmann, M. “Universal, Transferable Adversarial Perturbations for Visual Object Trackers”, Under review at Transactions of Pattern Analysis and Machine Intelligence (TPAMI), 2022

Chapter 7 is based on:

- **Nakka, K. K.**, & Salzmann, M. “Learning Transferable Adversarial Perturbations”, Advances in Neural Information Processing Systems (NeurIPS), 2021

Chapter 8 is based on:

- **Nakka, K. K.**, & Salzmann, M. “Understanding Pose and Appearance Disentanglement in 3D Human Pose Estimation”, Under review at European Conference on Computer Vision (ECCV) 2022

Roles of Co-authors

I am the sole student author of the work presented in this thesis except for the ICCV 2019 paper. The main contribution of the ICCV 2019 paper was conceptualized and implemented by Krzystof Lis and I contributed to unifying anomaly detection with the experiments on adversarial attack detection. Other than the ICCV 2019 paper which was not included in the thesis, all the papers and chapters in this thesis are my own original work which could not have been possible without the help of my advisor.

During the research and writing of each paper, regular meetings were performed with my advisor. We discussed the ideas, direction of research, the novelty of ideas, experiment design, reviews and rebuttal, among other things. More specifically, I contributed to the conceptualization, methodology, software, and original draft preparation. My advisor, Mathieu Salzmann, contributed to conceptualization, supervision, review and editing, and administration.

Contents

1 Introduction

Out of your vulnerabilities will come your greatest strength - Sigmund Freud

Automatically recognizing and segmenting objects in images and videos is central to a wide variety of application domains. Therefore, visual recognition has been one of the fundamental goals of Computer Vision since its inception. In 2012, the Computer Vision community underwent a revolution, started by the impressive results of the AlexNet [136] on the ImageNet object recognition challenge. Although Deep Learning has a long history in visual recognition, e.g., [140], it had until then remained only a marginal topic due to the lack of large, publicly available datasets and the lack of computation power. With the progress in GPUs and the ever-growing availability of images online, Deep Learning has led to impressive advances and achieved state-of-art in domains such as image recognition [101, 237], domain adaptation [164, 244, 258], face recognition [98, 123, 203], object detection [158], semantic segmentation [71, 163, 311, 311], and video classification [119, 275, 295]. Increasingly deeper architectures are being proposed [99, 220, 245], with more advanced classification layers [74, 128] and more effective optimization strategies [124]. DNNs are now well-established in many safety-critical applications such as autonomous driving, face recognition, and surveillance systems.

Even though the discriminative power of deep neural networks (DNNs) is nowadays unquestionable, one of the research challenges that remains is understanding the reasons behind a network's inference, e.g., how did the trained model arrive at the particular label. DNNs are often criticized as black boxes without the ability to explain the inference to a specific decision intuitively. For instance, understanding the reason for a particular prediction in the medical domain is essential to achieving the trustworthiness of the deployed model. In this regard, early efforts such as [232, 297, 316] were proposed to identify the most relevant regions responsible for prediction, followed by visualizing neurons [195], interpretable architectures [29] and other posthoc methods [213]. Furthermore, obtaining plausible explanations in a human-understandable way is indispensable to dissecting the internal workings of DNNs. Therefore, the topic of interpretability has received considerable attention, which empowers us to understand the strengths and weaknesses of DNNs and analyze the deeper reasons for the failures of a given data point.

Despite the remarkable progress, serious concerns have arised ever since the DNNs were discovered to be vulnerable to adversarial examples [246] crafted by adding imperceptible perturbations to clean images. Moreover, it was also demonstrated in [315, 317] that adversarial perturbations can be transferred to an unknown target network or can be fooled by query-based attacks [4] with access to output layer predictions. Besides, the attacks were extended to other vision tasks such as object detection [284], semantic segmentation [8], and face recognition [85, 282]. The implications of malicious attacks are even more significant for DNNs deployed in real-world systems, e.g., autonomous driving and biometric authentication. Given its utmost relevance to safety-critical applications, thus understanding the reasons for its vulnerability and exploring different ways to construct transferable black-box attacks is an important direction toward attaining truly robust models.

Much effort has been put into the different ways to attack [26, 137, 201] given the input image. In contrast to the image-dependent attacks, [184] showed that the DNNs are indeed vulnerable to image-agnostic precomputed universal perturbation. In addition, adversarial examples are proven transferable across multiple architectures [315, 317]. Both these findings point to an underlying perspective deeply connected to the internal working mechanism of DNNs. We thus aim to understand the behavior of DNNs to adversarial attacks from the perspective of interpretability. By understanding the attacks through interpretable models, one can gain not only valuable directions against future adversarial attacks but also opens up new ways to defend against them.

In this thesis, we propose several approaches to explain the behavior of DNNs and study the mechanism of adversarial examples. Specifically, our work seeks to understand the reasons for predicting a particular label through intrinsically interpretable BoW architectures. In doing so, we expand our present understanding of DNNs limitations and also use as a tool to design effective and intuitive attack strategies across diverse tasks. Furthemore, we focus on the transferability of adversarial attacks in the extreme black-box case where the attacker has zero knowledge about target architectures, tasks, and datasets. Besides, we propose an attentional-based framework to improve the adversarial robustness with metric learning. Overall, the findings in this thesis provides an insightful perspective on prediction mechanism underlying the DNNs and presents meaningful adversarial attacks by understanding their internal representation.

In the remainder of this chapter, we provide a brief background on adversarial attacks. We then summarise our core contributions and finally present the outline at the end of the chapter.

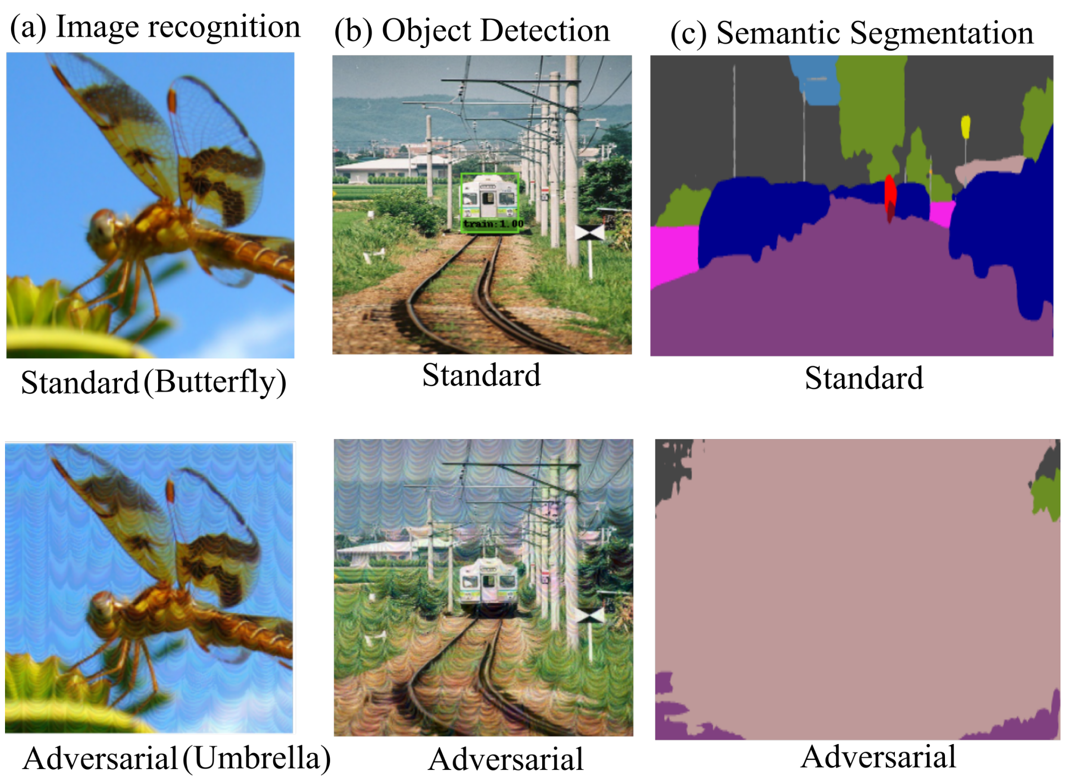


Figure 1.1 – **Adversarial attacks on different computer vision tasks.** By adding small perturbation $\ell_\infty = 10$ to the input images, the output predictions change drastically over wide range of computer vision tasks. The three images in the first row are chosen from ImageNet [46], Pascal VOC [60] and Cityscapes [40], respectively.

1.1 Background

Let us first briefly introduce the topic of adversarial attacks for context and familiarity.

DNNs were first shown to be vulnerable to adversarial, human-imperceptible perturbations in the context of general image recognition. Such attacks were initially studied in [246], quickly followed by the simple single-step Fast Gradient Sign Method (FGSM) [79] and its multiple-step BIM variant [137]. In [52], the attacks were stabilized by incorporating momentum in the gradient computation. Other popular attacks include DeepFool [182], which iteratively linearizes the classifier to compute minimal perturbations sufficient for the sample to cross the decision boundary, and other computationally more expensive attacks, such as CW [26], JSMA [201], and others [189, 243, 286]. As of today, Projected Gradient Descent (PGD) [168], which utilizes the local first-order network information to compute a maximum loss increment within a specified ℓ_∞ norm-bound, is generally considered as the most effective attack strategy.

While the above methods are image-dependent, the existence of Universal Adversarial Perturbation (UAP) was first shown in [184], considering the task of learning a single perturbation that can fool a classifier independently of the input image. In parallel to UAPs, several works have shown that iterative adversarial attack strategies could be transferred across architectures [104, 183, 184]. The above-mentioned methods, however, are computationally expensive at inference time [26, 52, 102, 104, 168, 201].

On the other hand, Generative adversarial perturbations (GAP) were first introduced in [212]. In particular, [212] showed that a generator network can be used to craft a UAP that transform an input image to an image-dependent perturbation. Similarly, [88] introduced the advGAN generative framework to learn to produce adversarial perturbations, and further proposed to make use of distillation to perform black-box attacks. Based on the observation that the effectiveness of these two methods strongly depends on the availability of data from the target domain during training, [191] introduced the relativistic cross-entropy loss, which was shown to better generalize across datasets. Furthermore, [148] proposed to generate UAPs by transforming the generator's output using a regional norm layer that enforces perturbation homogeneity. Along with above two broad themes, several decision-based attacks [4, 20, 34, 54] have been proposed to study the robustness of DNNs.

Motivated by the observations made in the context of image classification, adversarial attacks were extended to other tasks such as semantic segmentation, object detection, and object tracking, among others as illustrated in Figure 1.1. They have now been studied in almost every computer vision task to understand the limitations of DNNs. In [8], the effectiveness of attack strategies designed for classification was studied for different segmentation networks. In [284], a dense adversary generation attack was proposed, consisting of projecting the gradient in each iteration with minimal distortion. In [92], a universal perturbation was learnt using the whole image dataset. Furthermore, [10] demonstrated the existence of perturbations that are robust over chosen distributions of transformations. In the task of VOT, SPARK [84] computes incremental perturbations by using information from the past frames; [31] exploits the full video sequence to attack the template by solving an optimization problem relying on a dual attention loss. Besides the mainstream applications, adversarial tasks were extended to a host of other tasks such as depth prediction [271, 309], pose estimation [108], optical flow [129, 218, 276], superresolution [37, 290], face recognition [55, 80, 85, 282], perception models [150, 240, 289].

1.2 Contributions

Having briefly introduced the background on adversarial attacks, we now present the contributions made in this thesis.

In the first broad line of work, we focus on understanding the underlying decision mechanism of deep neural networks to adversarial examples. To this end, we propose architectures to interpret the decisions of DNN, which also allows understanding the reasons for the success of adversarial attacks. We then leverage the interpretable architectures to propose a framework to detect adversarial attacks and improve the robustness using attention-based feature regularization. The following works contribute to the above theme.

- **Attention-Aware Deep Structured Representation Learning.** Deep structured representation learning strategies such as Bag of Words and VLAD are popular for their ability to provide interpretability to output decisions through prototypes. However, they typically aggregate local features from the entire image, ignoring the fact that, in complex recognition tasks, some regions provide much more discriminative information than others. Therefore, our initial work introduces an attentional structured representation learning framework that incorporates an image-specific attention mechanism within the aggregation process to remove the influence of background regions.

- **Adversarial Attack Detection.** We then leverage the interpretable BoW-based structured representation networks to detect the adversarial examples. We build upon the intuition that, while adversarial samples look very similar to natural images, they should activate prototypes with a significantly different visual representation to produce the incorrect predictions. We, therefore, cast the adversarial example detection problem as that of comparing the input image with the most highly activated visual codeword.

  Similarly, we propose to detect the adversarial attacks in the semantic segmentation by comparing with the resynthesized image from the predicted label map instead of the codeword image. Specifically, we employ a pretrained pix2pix generator that outputs a scene image given the label map.

- **Adversarial Robustness by Discriminative Feature Separation.** We conduct a deeper analysis of the success of adversarial attacks in the fine-grained domain by visualizing the activated prototypes. Motivated by our findings, we improve the adversarial robustness by introducing an attention-based regularization mechanism that maximally separates the latent features of class-specific discriminative regions while minimizing the contribution of the non-discriminative regions to the final class prediction. Furthermore, by pushing the discriminative regions apart and discarding the background regions, we make the attackers' task difficult, as evidenced through different white-box, transfer, and query-based attacks.

In the second part of this thesis, we shift our focus to designing adversarial attacks to better understand the DNNs limitations on multiple tasks such as image recognition, semantic segmentation, pose estimation, and object tracking. Our results enable us to expand our knowledge of the DNNs internal workings and understand their critical vulnerabilities before deploying them in real-world scenarios. With applications spanning diverse domains in computer vision, we believe the following works will contribute to a deeper understanding of the DNNs to adversarial attacks.

- **Indirect Local Attacks on Context-aware Segmentation Networks.** We show for the first time that the state-of-the-art segmentation networks that exploit surrounding context are susceptible to our indirect local attacks, where the perturbations are confined to a small image region that does not overlap with the area that the attacker aims to fool. We then extend this to formulate an adaptive attack using group sparsity prior aiming to find the optimal image location to perturb non-overlapping from the object of interest while preserving the labels at this perturbed location and producing a realistic-looking segmentation map. In addition, we extend the image-classification-based adversarial detection approaches to detect the attacks both at the image and pixel level.

- **Universal Perturbations for Visual Object Trackers.** Unlike the image recognition or semantic segmentation, where the label set is predefined to a fixed number of classes, visual object trackers aim to track a novel object that is non-overlapping with the objects of training time. Further, the attacker's task is many levels complicated due to the requirement of generating perturbations in real-time. To overcome these specific challenges, we discover the existence of a universal perturbation that is image agnostic and fools, black-box trackers, at virtually no cost of perturbation. At the core of our framework, we propose to learn to generate a single perturbation from the object template only that can be added to every search image and still successfully fool the tracker for the entire video. Consequently, the resulting generator outputs perturbations that are quasi-independent of the template, thereby making them universal perturbations with high attack efficiency.

- **Transferable Adversarial Perturbations through Mid-level Feature Separation.** We focus on the transferability of perturbations in a zero-query setting where the attacker has no knowledge about the target system. Specifically, we study the transferability of generator-based attacks when the conditions at inference time differ from the training ones in terms of the target architecture, target data, and target task. Our in-depth analysis on the functioning of the DNNs identifies the mid-level features extracted by the intermediate layers as the common ground to attack across different architectures, datasets, and tasks. Motivated by this observation, we introduce a loss function to disrupt the mid-level features inorder to learn a highly effective, transferable perturbation generator.

- **An Adversarial Attack Perspective on Appearance-Pose Disentanglement.** We conduct a deeper analysis of the disentanglement of internal latent representations of pose and appearance in self-supervised 3D pose estimation networks. We design an adversarial strategy focusing on generating semantic appearance changes of the subject, against which we argue that a disentangled pose-estimation network is expected to be more robust. However, our analyses, as were also corroborated by image synthesis experiments and quantitative metrics, show that disentanglement in the three state-of-the-art disentangled representation learning frameworks is far from complete. Furthermore, we provide evidence to substantiate that the pose latent codes contain significant appearance information and are thereby vulnerable to semantic appearance attacks.

Thesis Organisation

The remainder of the thesis is organized as follows. In Chapter 2, we introduce an attention-aware deep structured representation framework to remove the influence of non-discriminative regions in the aggregation process. In Chapter 3, we present an approach to detect the adversarial examples in the gray and black-box settings by leveraging the Bag of visual words architectures. Chapter 4 proposes a framework to improve the adversarial robustness by maximal separation of the discriminative regions through regularising the latent space with attention-based clustering and separation losses. Chapter 5 demonstrates that context-dependent networks are susceptible to indirect local attacks far away from the attacked region. Chapter 6 discovers the existence of universal perturbation that is image agnostic and fools black-box trackers at virtually no cost of perturbation. Chapter 7 illustrates that the disrupting the mid-level features extracted by the intermediate layers of DNNs as common ground across different architectures, datasets, and tasks, to learn a highly effective transferable perturbation generator. In Chapter 8, we focus on the disentanglement of pose and appearance and provide evidence that the pose latent codes contain significant appearance information through an adversarial strategy focusing on generating natural appearance changes of the subject.

2 Attention-Aware Structured Representation Learning

Structured representations, such as Bags of Words, VLAD and Fisher Vectors, have proven highly effective to tackle complex visual recognition tasks. As such, they have recently been incorporated into deep architectures. However, while effective, the resulting deep structured representation learning strategies typically aggregate local features from the entire image, ignoring the fact that, in complex recognition tasks, some regions provide much more discriminative information than others.

In this chapter, we introduce an attentional structured representation learning framework that incorporates an image-specific attention mechanism within the aggregation process. Our framework learns to predict jointly the image class label and an attention maps in an end-to-end fashion and without any other supervision than the target label. As evidenced by our experiments, this consistently outperforms attention-less structured representation learning and yields state-of-the-art results on standard scene recognition and fine-grained categorization benchmarks.

2.1 Introduction

In recent years, Convolutional Neural Networks (CNNs) have emerged as the de facto standard for visual recognition. Nevertheless, while they achieve tremendous success at classifying images containing iconic objects, their performance on more complex tasks, such as scene recognition and fine-grained categorization, remains comparatively underwhelming. This is partly due to their simple pooling schemes that fail to model the dependencies between local image regions. By contrast, in the realm of handcrafted features, structured representations, such as Bags of Words (BoW) [125, 216, 239], Vectors of Locally Aggregated Descriptors (VLAD) [6, 109, 110] and Fisher Vectors (FV) [210, 229], have been shown to be highly discriminative thanks to their aggregation of local information. As a consequence, they have started to re-emerge in the deep networks realm, with architectures such as NetVLAD [5] and Deep FisherNet [247].

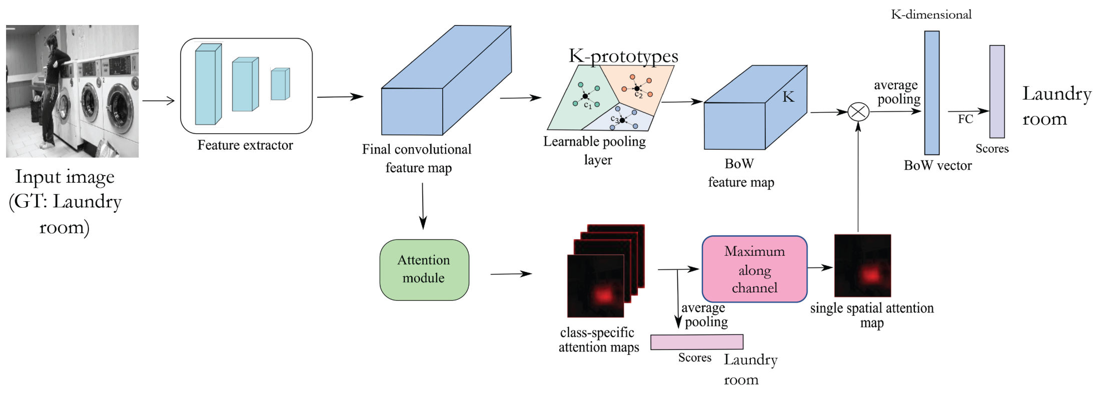


Figure 2.1 – **Attentional structured representation network.** Our network consists of two branches with a shared base feature extraction CNN. The attention module produces class-specific attention maps, which are then incorporated into the VLAD module that outputs an attention-aware VLAD representation. Note that, while we focus on the VLAD case here, as evidenced by our experiments, our approach applies to any structured representation.

While effective for complex visual recognition tasks, these structured representations, whether based on handcrafted features or incorporated into deep networks, suffer from one drawback: They aggregate local information from the entire image, regardless of how relevant this information is to the recognition task. In practice, however, while certain image regions contain semantic information that contribute to the target label, others clearly don't. For example, in the image shown in Fig. 2.1, from the MIT-Indoor dataset [215], the region depicting washing machines gives us a much stronger cue of the class *laundry* than the regions containing the person and the background. Incorporating information from these latter two regions, which can appear in many other scene categories, will typically yield less discriminative image representations.

In this chapter, we address this by introducing a novel deep attentional structured representation network for visual recognition. Our network incorporates an image-specific attention mechanism that encourages the learnt structured representation to focus on the discriminative regions of the image. We then learn to predict jointly the input image class label and the spatial attention map without requiring any annotations for the latter.

Our framework is depicted by Fig. 2.1 for the case of a VLAD aggregation strategy. It consists of two streams that share a base network extracting deep features: The attention module and the VLAD module. The attention module, based on the framework of [74], learns a set of filters that transform the deep features into C heatmaps encoding attention for the C classes of interest. The VLAD module then exploits these heatmaps to form an attention-aware VLAD vector from the deep features of the base network. We train our network with a combination of two losses that encourage both the attention maps and the final attention-aware VLAD representation to be discriminative. Note that, while Fig. 2.1 focuses on the VLAD case, as evidenced by our experiments, our approach generalizes to any local feature aggregation strategy.

In short, we contribute the first systematic integration of an attention mechanism within a structured image representation learning framework. We demonstrate the benefits of our approach on four challenging visual recognition tasks, including scene recognition on MIT-Indoor [215] and fine-grained categorization on the UCSD Birds [259], FGVC Aircrafts [172] and Stanford Cars [133] datasets. Our attentional structured representation learning strategy consistently outperforms its standard attention-less counterpart and yields state-of-the-art results on several of the above-mentioned datasets.

2.2 Related Work

Over the years, visual recognition has attracted a huge amount of attention in Computer Vision. Before the deep revolution in 2012, most methods adopted a two step pipeline consisting of extracting handcrafted features and training a classifier, such as Support Vector Machines [91] or Boosting [70]. In this pipeline, the core Computer Vision research was targeted towards extracting discriminative image features. In particular, Bags of Visual Words (BoW) [125, 216, 239], based on local features such as SIFT [165] or BRIEF [24], have proven effective for image recognition. Later, such histogram-based features were extended to VLAD [109, 110] and Fisher Vectors [210, 229], which model higher-order statistics of the data w.r.t. the codewords. After the remarkable performance of AlexNet [136], much of the visual recognition research turned to deep learning strategies. While many new architectures do not explicitly focus on extracting structured representations, some work has nonetheless attempted to leverage the lessons learnt from handcrafted features. In particular, [77] performs multi-scale orderless pooling of deep CNN features, and [38, 50] compute Fisher encodings of similar deep features. In contrast with these approaches that still separate feature extraction from classifier learning, NetVLAD [5] and Deep FisherNet [247] constitute the first attempts at introducing learnable VLAD and Fisher Vector layers, respectively, within an end-to-end learning formalism. More recently, [149] proposed to make use of a mixture of factor analyzers to model an accurate Fisher Vector with full covariance matrix. While the previous methods all rely on histogram-based descriptors, in the context of fine-grained categorization and texture recognition, other structured representations, in the form of covariance matrices have been used [154, 155, 292]. In [236], a generalization of average and bilinear pooling was proposed to automatically learn an intermediate pooling strategy during training. In any event, all these methods, whether using hand-crafted features or relying on deep learning, aggregate local features from the entire image, without accounting for the fact that only parts of the image contain information that contributes to the target class label. This will typically reduce the discriminative power of the resulting representations.

For complex tasks, such as scene classification and fine-grained categorization, some research has nonetheless attempted to focus the feature extraction process on discriminative image regions. In the context of scene recognition, this was achieved by modeling the scene with mid- (or high-)level representations [146], such as detected semantic visual attributes [207], patch-based codewords obtained via discriminative clustering [238] and object-oriented representations learnt from a manually-created database of typical scene objects [273]. For fine-grained categorization, several works exploit bounding box annotations to learn part detectors [302, 306]. The use of such additional annotations was then removed in [281], which learns part templates by clustering deep features. More recently, [72, 314] introduced end-to-end learning strategies to automatically identify discriminative regions for fine-grained recognition.

The above-mentioned works typically reason about the notion of parts, or objects in a scene. In the rare cases that don't require part annotations during training [72, 314], the input image is first processed globally to identify regions with high attention, which are then cropped into multiple parts that are processed individually. By contrast, our network processes the input image in a single forward pass, without explicitly relying on the notion of parts. In essence, these methods are therefore tailored to the specific problem they tackle. By contrast, here, we exploit the more general notion of visual attention and produce heatmaps encompassing the discriminative regions in the image. This does therefore not require any prior knowledge about the data at hand. Our formalism builds upon the attention framework of [74], but with the additional goal of leveraging structured representations. Moreover, the attention branch is not limited to [74] but can be replaced with latest attention architectures such as self-attention [301] and multi-head attention frameworks in vision transformers [56]. As a consequence, and as evidenced by our results, our approach yields higher accuracies than both attention-less methods and unstructured attentional pooling in all the tasks we tested it on.

2.3 Method

In this section, we introduce our novel attentional structured representation learning framework depicted by Fig. 2.1. We first present the structured representation and attention modules, and finally our approach to integrating them in an end-to-end learning formalism.

2.3.1 Structured Representation Module

Structured representations aggregate local descriptors into a global feature vector of fixed size using a visual codebook. In particular, here, we focus on VLAD, which has proven highly effective. As will be evidenced by our experiments, however, our framework generalizes to other aggregation strategies.

In contrast to BoW that only store information about which codeword each local descriptor is assigned to, VLAD also computes the residual distance of the descriptor to the codeword. To incorporate this into a deep learning framework, the hard codeword assignment of each descriptor is replaced by a soft one. More specifically, let $\mathbf{I}$ be an image input to a CNN, and $\mathbf{X} \in \mathbb{R}^{W \times H \times D}$ the feature map output by the last convolutional layer, with spatial resolution $W \times H$ and D channels. $\mathbf{X}$ can then be thought of as $N = W \times H$ local descriptors $\mathbf{x}_i$ of dimension D. Given a codebook $\mathbf{B}$ with K codewords, VLAD produces a DK-dimensional representation of the form

$$\mathbf{v} = [\mathbf{v}_0^T, \mathbf{v}_1^T, \cdots, \mathbf{v}_K^T]^T \ , \tag{2.1}$$

where $\mathbf{v}_k \in \mathbb{R}^D$ is given by

$$\mathbf{v}_k = \sum_{i=1}^{N} a_k(\mathbf{x}_i)\,(\mathbf{x}_i - \mathbf{b}_k) \ , \tag{2.2}$$

with $\mathbf{b}_k$ the k-th codeword of codebook $\mathbf{B}$. The values $a_k(\mathbf{x}_i)$ represent the assignment of descriptor $\mathbf{x}_i$ to codeword $\mathbf{b}_k$. In the standard VLAD formalism, these assignments are binary, with each descriptor being assigned to a single codeword. Within a deep learning context, for differentiability, these assignments can be relaxed and expressed as

$$a_k(\mathbf{x}_i) = \frac{e^{-\alpha\|\mathbf{x}_i - \mathbf{b}_k\|^2}}{\sum_{k'} e^{-\alpha\|\mathbf{x}_i - \mathbf{b}_{k'}\|^2}} \ , \tag{2.3}$$

with α a hyperparameter defining the softness of the assignments.

The resulting VLAD vector then acts as input to the classification layer of the deep network. While effective, as discussed above, the VLAD representation aggregates information from the entire image, regardless of whether the local descriptors correspond to discriminative regions or not. Below, we first discuss a general attention module, which is able to identify relevant image regions, and then introduce our approach to incorporating this information within our structured representation learning formalism.

2.3.2 Attention Module

It has been shown multiple times that CNNs were not only effective at predicting the class label of an image, but could also localize the image regions relevant to this label [114, 233, 316]. Most existing approaches to performing such a localization, however, work as a post-training step. By contrast, our attention module, based on the framework of [74], produces attention maps that are actively used during training. Furthermore, it combines top-down attention, modeling class-specific information, with bottom-up attention, modeling class-agnostic information, or, in other words, a form of image saliency. Such integration [193] of top-down cues [12, 304, 316] with bottom-up attention [226] modulates the image saliency map to ignore non-relevant background regions of the target.

Specifically, let $\mathbf{X}$ be the same final $W \times H \times D$ convolutional feature map as in Section 2.3.1. Our attention module consists of an additional 1×1 convolutional layer with one class-agnostic filter with parameters $\mathbf{w}_{ca} \in \mathbb{R}^{D \times 1}$ and C class-specific filters whose parameters can be grouped in a matrix $\mathbf{W}_{cs} \in \mathbb{R}^{D \times C}$, where C is the number of classes of the problem at hand. This convolutional layer produces a class-agnostic heatmap $\mathbf{H}_{ca}$ and class-specific heatmaps $(\mathbf{H}_{cs}^1, \cdots, \mathbf{H}_{cs}^C)$, each of spatial resolution $W \times H$. Each class-specific heatmap is then multiplied element-wise by the class-agnostic one, yielding C attention maps $(\mathbf{H}^1, \cdots, \mathbf{H}^C)$.

Training the attention module can be achieved by global average pooling of each of the attention maps, which produces a score for each class. These scores are then passed through a softmax layer, and the resulting probabilities $\{p_c\}$ used in a standard cross-entropy loss

$$L_{att} = -\frac{1}{S}\sum_{s=1}^{S} \log(p_{c^*}(\mathbf{I}_s)) \ , \tag{2.4}$$

where S is the number of samples in a mini-batch and $p_{c^*}(\mathbf{I}_s)$ is the probability of the ground-truth class for sample s. This was the procedure used in [74] to train an attentional deep network. Below, we propose to rather make use of the attention maps to further build a more discriminative structured representation. As evidenced by our results, this allows us to achieve consistently higher recognition accuracies.

2.3.3 Attention-aware Feature Aggregation

Our goal is to make use of the attention maps when aggregating the local descriptors into a structured representation. To this end, instead of global average pooling the maps, we generate a single attention map, which can be interpreted as a weight $w(\mathbf{x}_i)$ for every descriptor $\mathbf{x}_i$, and is defined as

$$w(\mathbf{x}_i) = \frac{\max\limits_{l} \mathbf{H}_i^l}{\sum\limits_{i'} \max\limits_{l} \mathbf{H}_{i'}^l} \ , \tag{2.5}$$

where $\mathbf{H}_i^l$ indicates the attention-weight corresponding to feature $\mathbf{x}_i$ in the attention map of class l from Section 2.3.2. The resulting attention map has the same spatial resolution as the final deep feature map. We then use it to re-weight the aggregation scheme of Eq. 2.2. Specifically, we re-write Eq. 2.2 as

$$\mathbf{v}_k = \sum_{i=1}^{N} w(\mathbf{x}_i) a_k(\mathbf{x}_i) \left(\mathbf{x}_i - \mathbf{b}_k\right) \ . \tag{2.6}$$

Following common practice [5, 110], we perform L_2 normalization of each $\mathbf{v}_k$ to remove burstiness, followed by a final L_2 normalization of the entire vector $\mathbf{v}$. The resulting representation is then passed to a classification layer.

Ultimately, our network combines an attention module with a structured representation learning one. Both modules share the base network up to the final convolutional feature map. To train our network, we first pre-train the base network with the attention module only using L_{att} from Eq. 2.4. We then continue training the entire network in an end-to-end manner by minimizing a loss of the form

$$L = L_{cls} + \lambda L_{att} \ , \tag{2.7}$$

where L_{cls} is a cross-entropy loss on the output of the classifier acting on the structured representation $\mathbf{v}$, and λ is a hyper-parameter setting the relative influence of both terms. At test time, we then take the prediction from the VLAD branch of the network.

Note that training is not only done w.r.t. the network parameters, but also w.r.t. to the codebook $\mathbf{B}$. As suggested in [5], and motivated by [6] to adapt VLAD descriptors to new datasets, we decouple the soft assignment $a_k(\mathbf{x}_i)$ from the codeword $\mathbf{b}_k$. That is, we re-write the assignment $a_k(\mathbf{x}_i)$ of Eq. (3.2) as

$$a_k(\mathbf{x}_i) = \frac{e^{\mathbf{s}_k^T \mathbf{x}_i + h_k}}{\sum_{k'} e^{\mathbf{s}_{k'}^T \mathbf{x}_i + h_{k'}}} \ , \tag{2.8}$$

where $h_k = -\alpha \left\|\mathbf{b}_k\right\|^2$ and $s_k = 2\alpha \mathbf{b}_k$ are treated as independent parameters.

Geometric Interpretation of Attention

Consider the features of two images from same class with different backgrounds that are assigned to the same codeword, depicted as a Voronoi cell in Fig. 2.2. The features with high attention are shown in blue and those with low attention in red and orange, respectively. While ignoring attention would yield residual vectors pointing in almost opposite directions, our attention-aware aggregation produces vectors with high cosine similarity, shown as blue arrows. The inverse reasoning can be made for images from two different classes but containing common elements that are irrelevant to the class labels: By ignoring attention, these shared elements would yield components with high cosine similarity, thus decreasing the discriminative power of the complete VLAD vector. Attention allows us to discard these shared elements.

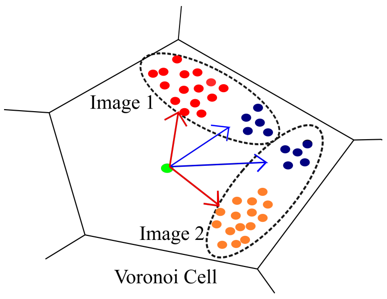


Figure 2.2 – Geometric interpretation of attention

2.4 Experiments

We first present the datasets used in our experiments and implementation details for our model. Then, we demonstrate the benefits of our attention-aware structured representation learning framework over its attention-less counterpart and over unstructured attentional pooling, and finally compare our results to the state of the art on each dataset.

2.4.1 Datasets

We experiment on the MIT-Indoor scene recognition dataset and on three fine-grained categorization datasets, namely CUB-200, Stanford cars and aircraft. We discard the part annotations but conduct experiments with and without bounding box annotations on fine-grained datasets.

MIT-Indoor is a widely used benchmark dataset for scene classification with 67 classes. We use the train/test split of [215] consisting of roughly 80 training and 20 test images.

CUB-200 is a challenging dataset with $11,788$ images of 200 bird species, with an average of 60 images per class. The dataset has extremely large variations in pose, size and viewpoints. We use the standard train/test split of [259].

FGVC-Aircraft contains 100 different aircraft models with roughly 100 images for each model. We adopt the same train/test split as in [172].

Stanford Cars is a 196 class dataset [133] of 8144 training images and 8041 test images. Heavy background clutter makes this dataset challenging.

2.4.2 Implementation Details

We use the VGG-16 [237] model pre-trained on Imagenet [136] as our base model and that of the baselines, with the $conv5\_3$ features before $ReLU$ activation as final convolutional features for aggregation. Following prior work [149, 155], we resize the images to 512×512 for MIT-Indoor, and 448×448 for the fine-grained datasets. Data augmentation is carried out on all datasets by performing random cropping and horizontal flipping. At test time, we flip the image and average the predictions for the original and flipped image. For structured representations, we fix the codebook size to $K = 64$ for VLAD and $K = 4096$ for the BoW experiments. We initialize the weights of the VLAD layer with K-means clustering of the $conv5\_3$ features. We set α in Eq. 3.2 to 100, and λ in Eq. 2.7 to 0.4.

Training: We use the ADAM optimizer [124] with parameter $\epsilon = 10^{-4}$, batch size of 16 and a weight decay of 0.0005 for all experiments. We first pre-train the attention network with $\eta = 0.0001$ for 20 epochs. For scene recognition, we then train the classification layer with $\eta = 0.01$ for 5 epochs, and further train the layers above $conv5$ with $\eta = 0.00001$

Pooling	Anno.	Birds	Cars	Aircrafts	MIT-Indoor
VGG-16	BBox	79.9	88.4	86.9	-
Attention	BBox	77.2	90.3	85.0	-
NetBoW	BBox	74.4	89.1	85.6	-
Attentional-NetBoW	BBox	80.5	91.2	89.3	-
NetVLAD	BBox	82.4	89.8	88.0	-
Attentional-NetVLAD	BBox	**85.5**	**93.5**	**89.2**	-
VGG-16		76.0	82.8	82.3	76.6
Attention		77.0	87.4	81.4	77.2
NetBoW		68.9	85.2	79.9	76.1
Attentional-NetBoW		76.9	90.6	88.3	76.6
NetVLAD		80.6	89.4	86.4	79.2
Attentional-NetVLAD		**84.5**	**92.8**	**88.8**	**81.2**

Table 2.1 – Comparison of our attentional structured pooling scheme with attention-less (VGG-16, NetBoW, NetVLAD) and structure-less (Attention) baselines. Our approach consistently outperforms these baselines, thus showing the benefits of pooling only the relevant local features into a structured representation.

for 25 epochs. For the fine-grained datasets, we train with $\eta = 0.01$ for the classification layer and $\eta = 0.0001$ for the layers above *conv*5 for 50 epochs, with a decay rate of 0.1 every 15 epochs.

2.4.3 Results

We first compare our approach to attention-less structured representation learning and to direct attentional pooling [74], and then to the state of the art on each dataset. To be consistent with prior work [149, 155], we report the average accuracy per class on MIT-Indoor and the average per image accuracy on the fine-grained datasets.

To evaluate the benefits of our attention-aware feature aggregation framework, we compare it with counterparts that do not rely on attention. In particular, we report results with VLAD pooling, as discussed in Section 2.3, but also with BoW representations, which can easily be obtained by using the soft assignments to form histograms. To further evidence the benefits of using structured representations, we compare our results with those of the direct attentional pooling strategy of [74], which relies on a global average pooling of the attention masks. The results of this comparison for all datasets are reported in Table 2.1, where we also show the accuracy of the standard VGG-16 model, with fully connected layers transformed into convolutional ones followed by global average pooling. Note that our Attentional-NetVLAD outperforms the baselines in all cases, both when using and not using bounding boxes for fine-grained recognition. Note also that using attention consistently helps improving the results, thus showing the importance of reasoning at the level of local features rather than combining information from the entire image in these challenging recognition tasks.

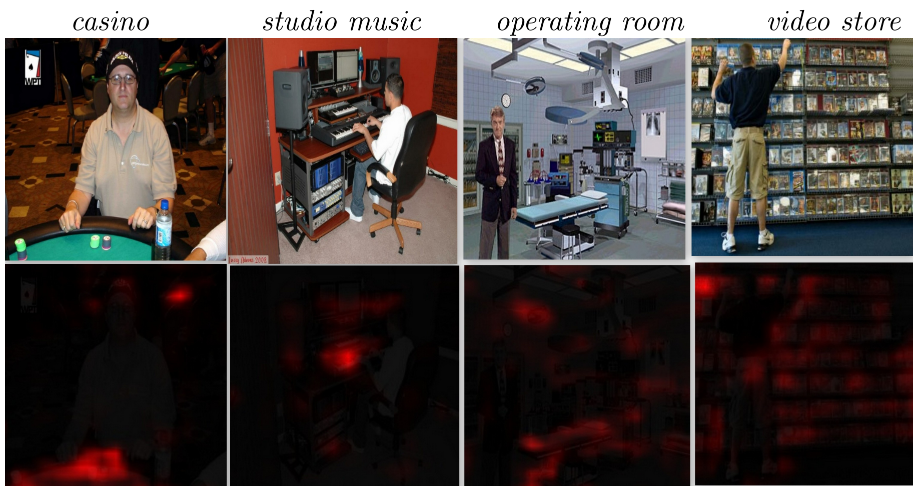


Figure 2.3 – **Attention maps for MIT-Indoor.** Each column shows an image from a different class (indicated above the image). Note that the maps focus on regions indicative of the label, ignoring the regions common to multiple classes, such as the people.

In Figs. 2.3 and 2.4, we provide some representative qualitative results of the attention maps obtained with our method for MIT-Indoor and the fine-grained datasets, respectively. For scene recognition, note that our network learnt to focus on the discriminative regions, such as the casino table and the piano, while ignoring regions shared by other classes, such as people. Similarly, for fine-grained categorization, the network is able to locate discriminative parts, such as the beak and the tail of birds, the brand logo and the head lights of cars, and the engine and landing gears of airplanes. This clearly evidences that our model can, in a single pass, find the regions of interest that define a class.

Finally, we compare our results with the state of the art on each individual dataset. These comparisons are provided in Table 2.2 for MIT-Indoor and Table 2.3 for the fine-grained datasets. In the case of scene recognition, we outperform all the baselines, including MFAFVNet [149], which relies on an accurate Fisher Vector encoding of 500K dimensions based on multi-scale image patches. For fine-grained recognition, we outperform all the baselines when relying on bounding boxes. Without bounding boxes, we achieve accuracies only slightly lower than the state-of-the-art methods, such as [72, 314], which were tailored to the fine-grained categorization problem, and rely on a multi-stage approach involving cropping parts and processing them separately. By contrast, our approach makes use of a single forward pass through a network and generalizes to any complex recognition scenario.

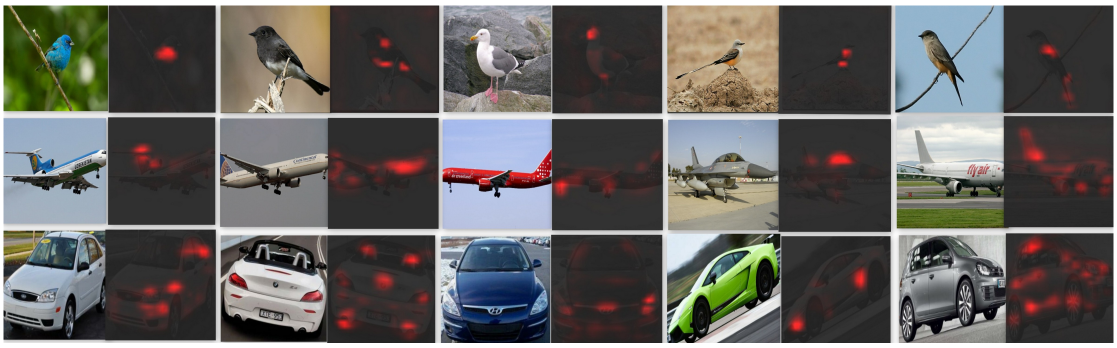


Figure 2.4 – **Attention maps for fine-grained datasets**. Our method is able to localize discriminative parts of birds (tail, beak), aircrafts (engine, landing gear) and cars (lights, logo).

2.4.4 Failure Cases

Finally, in Fig. 2.5, we show some typical failure cases of our approach, such as attention to background regions on Birds dataset.

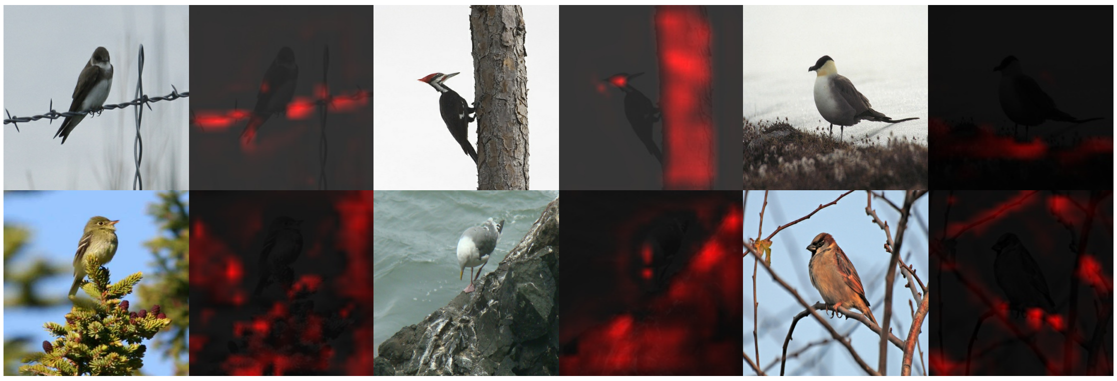


Figure 2.5 – Failure cases of our model particularly due to incorrect attention.

2.4.5 Ablation Study

We first evaluate the influence of the hyper-parameter λ in Eq. 7, which defines the strength of the attention module loss, on the classification accuracy. To this end, we evaluate our approach for different values of λ on the CUB-200 bird dataset, after 20 training epochs and without performing any data augmentation (image flipping) at inference time. The results of this experiment are provided in Table 2.4. Note that accuracy is stable over a very large range of values, thus showing that our approach is robust to the choice of this parameter.

Method	Avg. Acc.
Deep FisherNet [247]	76.5
CBN [73]	77.6
NetVLAD [5]	79.1
H-Sparse [157]	79.5
B-CNN [155]	79.5
SMSO [292]	79.7
FV+FC [38]	81.0
MFAFVNet [149]	81.1
Ours	**81.2**

Table 2.2 – Comparison with the state of the art on MIT-Indoor.

Method	Anno.	Birds	Cars	Aircraft
MG-CNN [260]	BBox	83.0	-	86.6
B-CNN [155]	BBox	85.1	-	-
PA-CNN [132]	BBox	82.8	92.8	-
Mask-CNN [267]	Parts	85.4	-	-
MDTP [263]	BBox	-	92.6	88.4
Ours	BBox	**85.5**	**93.5**	**89.2**
KP [44]		86.2	92.4	86.9
Boost-CNN [179]		86.2	92.1	88.5
B-CNN [155]		84.1	86.9	86.6
Imp. B-CNN [154]		85.8	92.0	88.5
α-pooling [236]		85.3	-	85.5
RA-CNN [72]		84.1	92.5	88.2
MA-CNN [314]		**86.5**	**92.8**	**89.9**
Ours		84.5	**92.8**	88.8

Table 2.3 – Comparison with the state of the art on fine-grained datasets.

λ	0.0001	0.01	0.4	1
Accuracy	83.3	83.7	83.7	83.1

Table 2.4 – Influence of λ on the final classification accuracy.

2.4.6 Attentional Global Average Pooling

While our main goal was to introduce an attention mechanism in structured representations, our approach also applies to unstructured pooling strategies, such as global average pooling (GAP). To illustrate this, we implemented an attentional GAP layer using our attention map. As shown in Table 2.5, this also typically outperforms the standard GAP strategy, thus further showing the benefits of attention when performing feature aggregation. Note, however, that our attentional VLAD strategy still significantly outperforms the GAP one.

Pooling	Anno.	Birds	Cars	Aircrafts
GAP	BBox	79.8	89.3	86.6
Attentional-GAP	BBox	76.3	91.1	88.3
NetVLAD	BBox	82.4	89.8	88.0
Attentional-NetVLAD	BBox	85.5	93.5	89.2
GAP		78.6	86.2	84.5
Attentional-GAP		77.8	89.6	85.5
NetVLAD		80.6	89.4	86.4
Attentional-NetVLAD		84.5	92.8	88.8

Table 2.5 – Attentional Global Average Pooling on fine-grained datasets.

2.5 Additional Qualitative Results

Below in Figures 2.6, 2.7, 2.8, and 2.9, we show the attention maps obtained with our approach for additional randomly-sampled images from the four datasets.

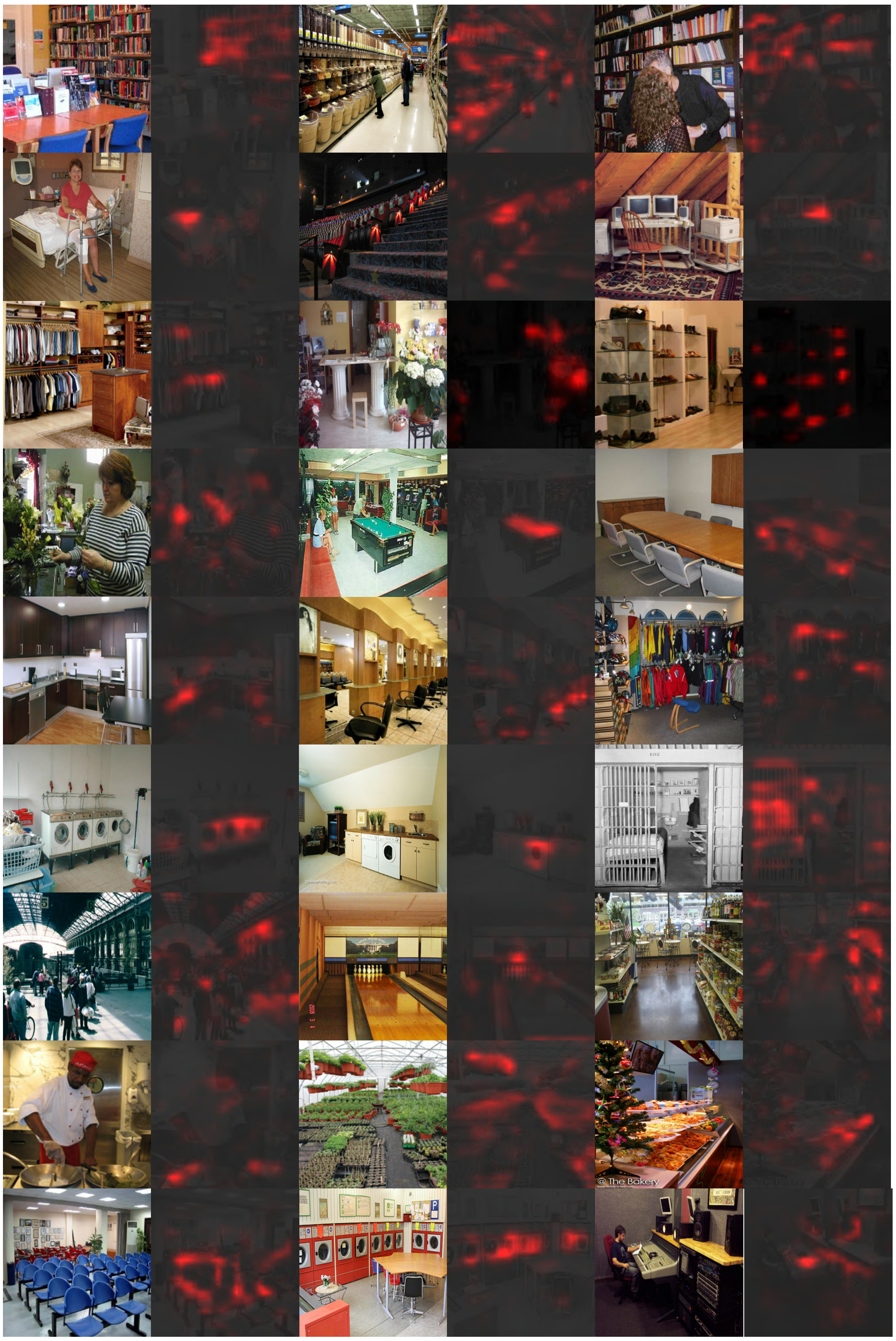


Figure 2.6 – Generated attention maps on the MIT-Indoor dataset.

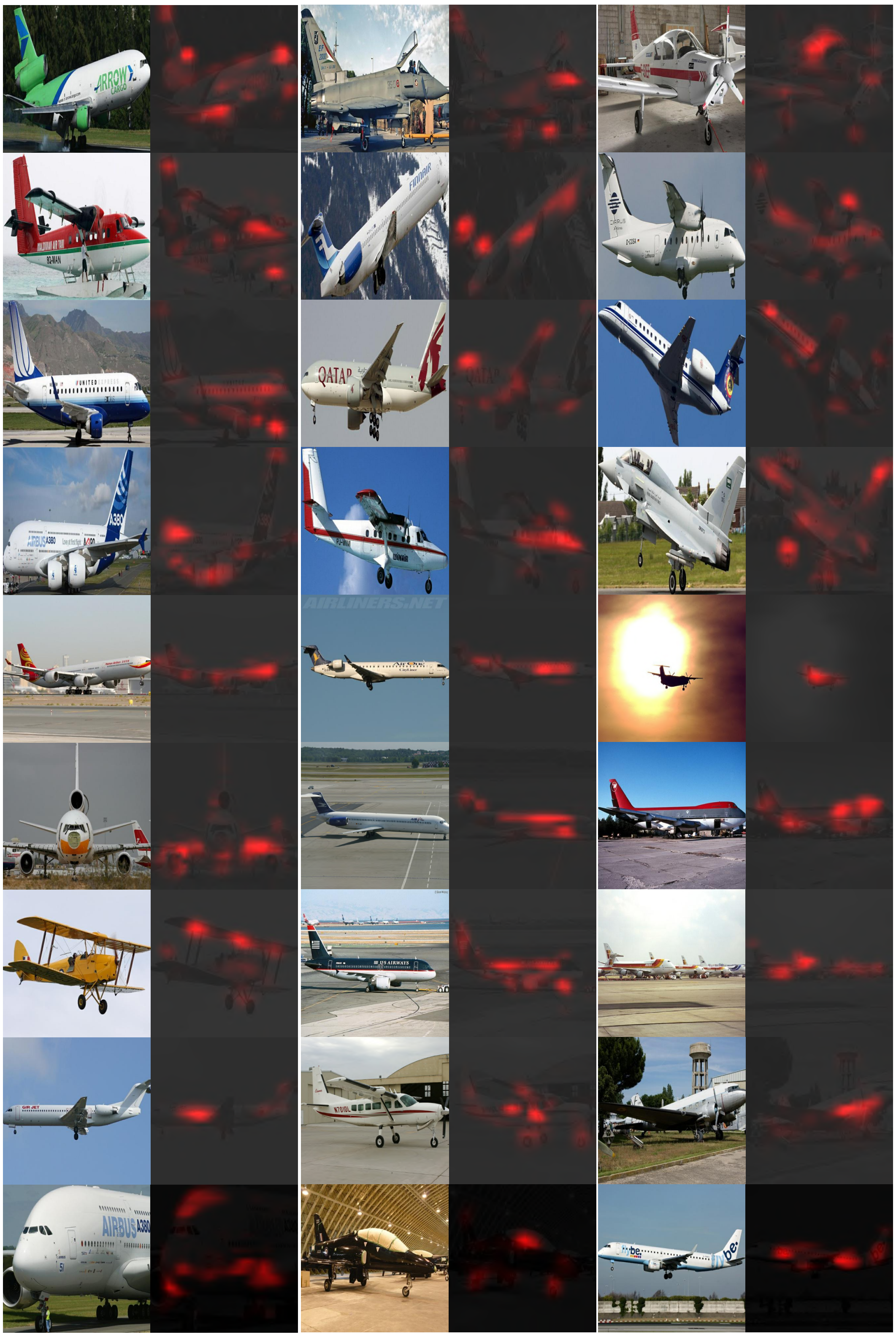


Figure 2.7 – Generated attention maps on the Aircrafts dataset.

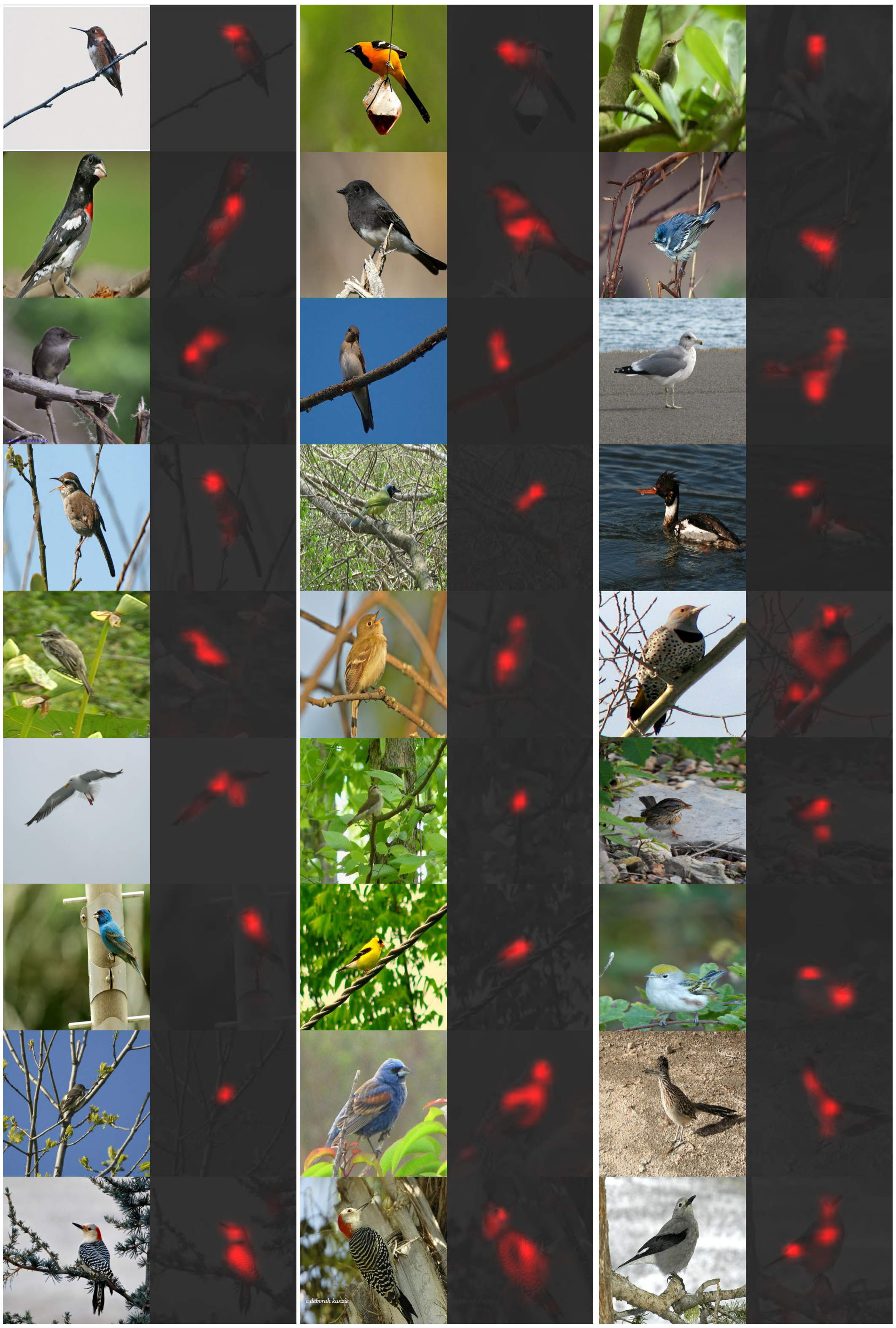


Figure 2.8 – Generated attention maps on the Birds dataset.

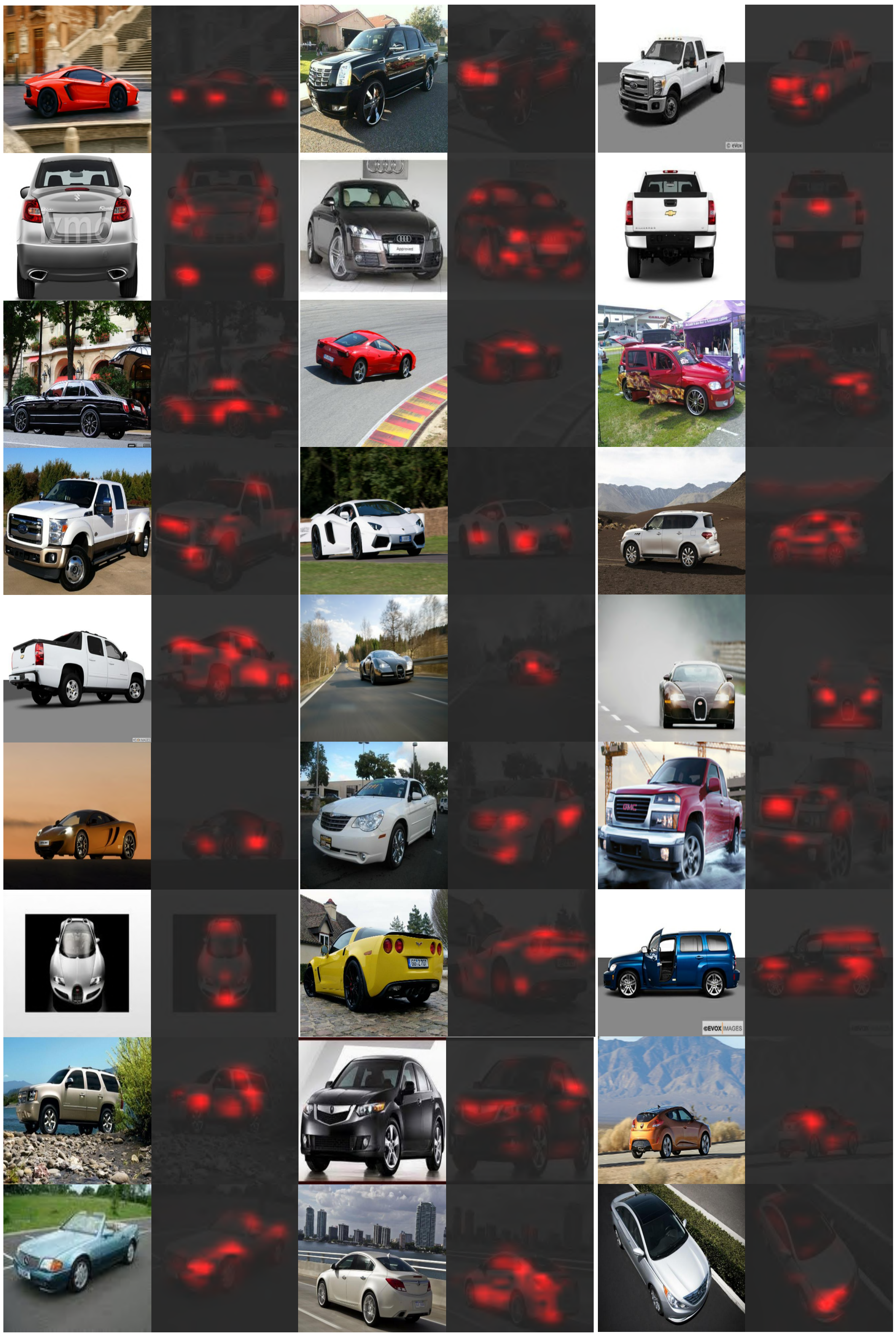


Figure 2.9 – Generated attention maps on the Stanford-Cars dataset.

2.6 Conclusion

We have introduced an attention-aware structured representation network for complex visual recognition tasks. Our network jointly identifies the informative image regions and learns a structured representation. Our comprehensive experiments on scene recognition and fine-grained categorization have demonstrated the superiority of our approach over attention-less strategies. Our approach is general and can be extended to other feature aggregation techniques, or can make use of any generic attention module. In the following two chapters, we show how these structured representations can be leveraged to detect and defend the adversarial attacks.

3 Semantic Dictionaries for Adversarial Example Detection

We now turn our focus on the task of adversarial example detection. To this end, we build upon our earlier work on structured representation networks to achieve this goal. In this chapter, we first introduce an approach to providing a visual interpretation of the results of a deep convolutional neural network (CNN). To this end, we exploit the earlier discussed Bag of visual Words (BoW) networks and introduce two strategies to assigning a visual meaning to a network's codebook elements. The network's predictions can then be interpreted by analyzing the visual representation of the most highly activated codeword. Doing so is particularly attractive to analyze a network's failures. We therefore propose to leverage our interpretable BoW networks for adversarial example detection. To this end, we build upon the intuition that, while adversarial samples look very similar to real images, to produce incorrect predictions, they should activate codewords with a significantly different visual representation. We therefore cast the adversarial example detection problem as that of comparing the input image with the most highly activated visual codeword.

Further, we show that our detection framework breaks against complete white-box adversary by creating a new adaptive attack. However, as evidenced by our experiments, our method outperforms the state-of-the-art adversarial example detection methods in gray-box and black-box threat settings w.r.t detector.

3.1 Introduction

While the discriminative power of deep convolutional neural networks (CNNs) is nowadays virtually uncontested, one of the key research challenges that remain unaddressed is the understanding of the reason behind a network's prediction. The main trends to achieve such an understanding consist of post-training analysis to visualize either feature maps at different layers [169, 170] or the image regions that contribute most to the decision [74, 233]. In this chapter, we introduce an alternative strategy to providing a visual interpretation of a network's prediction based on the analysis of its last feature map.

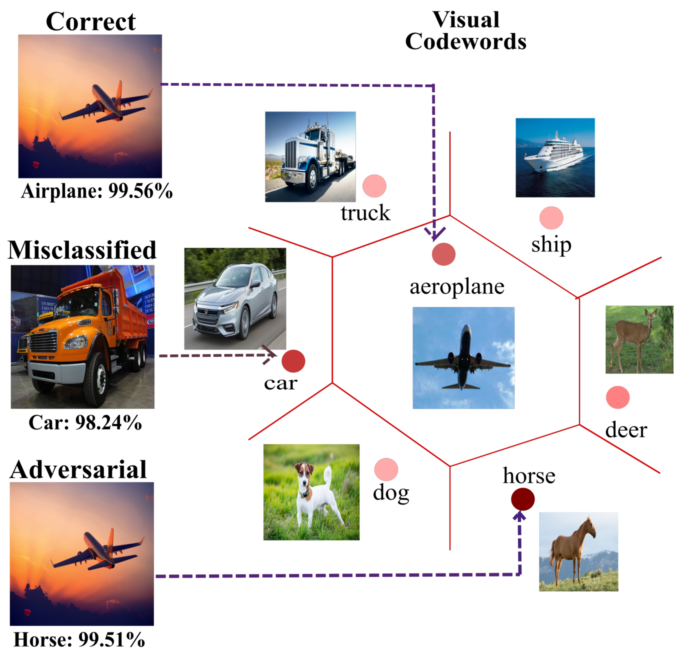


Figure 3.1 – **Interpretable BoW networks.** We assign a visual representation to the codewords to interpret the decisions of a CNN. Doing so allows one to analyze the reasons for assigning a given image to a particular class, which is particularly interesting when the network makes mistakes. We therefore leverage our visual representations to detect adversarial samples, whose visual interpretations will look very different from the input images.

A naive approach to doing so would consist of retrieving the nearest training sample in feature space and use the corresponding image as visual interpretation. This, however, quickly becomes prohibitively expensive as it requires performing a complete nearest-neighbor search at test time. To address this, we propose to make use of Bag of visual Words (BoW) networks. Inspired by traditional representations [125] BoW networks, such as HistNet [266], BoFNet [204], NetVLAD [5] and Deep FisherNet [247], incorporate a layer that relates the final CNN features to the elements of a codebook, learnt jointly with the network parameters, via a histogram-based representation. In essence, such a histogram encodes the similarity in feature space of the input sample to the codewords. We therefore propose to assign a visual interpretation to each codeword and, as depicted by Fig. 3.1, analyze the predictions by observing the most highly-activated visual codewords.

When it comes to analyzing a network's prediction, the most interesting use case probably is when the network makes a mistake. For example, in the middle example of Fig. 3.1, a *truck* image was classified as *car* because of its similarity to this class. In this chapter, we particularly focus on mistakes due to adversarial attacks: By altering an input image with a small amount of structured noise, invisible to the human eye, one can make a CNN predict virtually any output [246]. This erratic behavior can have a catastrophic impact in many real-world applications, such as face recognition and autonomous navigation. Here, we therefore propose to leverage our visual interpretation of a network's prediction to detect adversarial examples.

To this end, we rely on the intuition that an adversarial example (i) looks similar to the unaltered, but unknown, image; and (ii) to produce an erroneous prediction, will have a high activation for a codeword that corresponds to the wrong class. The visual interpretation of this codeword, however, constitutes a prototype of this wrong class, and thus, as shown in the bottom example of Fig. 3.1, will look very different from the adversarial example itself. We therefore cast adversarial example detection as the problem of comparing the adversarial image with the visual representation of the most highly activated codeword.

To summarize, our contributions are (i) the use of BoW networks to provide a visual interpretation of a network's prediction; (ii) two strategies to assign a visual meaning to the codewords of a BoW network: one based on nearest-neighbor search between the codeword and the training samples and another based on a generative adversarial network (GAN) [78], yielding an end-to-end trainable framework and allowing us to handle the case where a codeword lies far from any training example; (iii) a novel approach to adversarial and out-of-distribution example detection based on the comparison of the input image with the most highly activated visual codeword. Our experiments demonstrate that our interpretable BoW networks yield visually meaningful representations of a network's prediction and allow us to outperform the state-of-the-art adversarial example detection methods on standard benchmarks for state-of-the-art attack strategies, such as CW [26], FGSM [79], and BIM [137] in gray-box and black-box threat settings w.r.t. detector model.

3.2 Related Work

In this chapter, we propose to make use of BoW networks to provide a visual interpretation of a network's prediction. We then show how to exploit these visual representations for adversarial and out-of-distribution example detection. Below, we therefore review the literature related to the different aspects of our work.

BoW representations. Bags of visual Words (BoW) [125, 216, 239] have a longstanding history in computer vision. Originally developed in the context of handcrafted features, the core idea consists of extracting histograms by comparing local features to the elements of a codebook obtained from the training data. This idea was then extended to VLAD [110] and Fisher Vectors [210, 229], which encode higher-order statistics of the data with respect to the codewords. In the deep learning era, while most networks extract the final image representation using standard convolutions, a few works have attempted to leverage histogram-based representations. In particular, [77] exploits a VLAD pooling strategy on features extracted with a pre-trained network. While this separates feature extraction and classifier training, HistNet [266], BoFNet [204], NetVLAD [5], ActionVLAD [75], and Deep FisherNet [247] constitute end-to-end learning frameworks leveraging BoW, VLAD and Fisher Vector representations, respectively. In this chapter, our goal is not to introduce a new histogram-based architecture. Instead, we propose to leverage BoW networks to provide a visual interpretation of a network's prediction.

Interpreting CNNs. Attempts at interpreting the representations learned by CNNs or their predictions remain few, and existing methods follow two main trends. The first one [169, 170, 195, 297] focuses on visualizing the CNN filters in a post-training stage, by either inverting the network, or performing gradient ascent in image space to maximize neuron's activations. The second trend consists of identifying the image regions that bear the most responsibility for the prediction. This idea can be traced back to non-deep learning strategies, such as representations based on object detectors [146] and classemes [251], and even part-based models [65, 66, 67]. In the deep learning context, this was introduced by [316], extended in [233], both of which use a post-training strategy. This was followed by [261] that incorporates an attention module at every layer of the network, and [307] that uses an additional loss to assign each CNN filter to an object part. Recently [74, 188] have proposed to leverage attention maps during pooling operations. In this chapter, we introduce an alternative way to interpret the prediction of a CNN by providing a visual representation to the codewords of a BoW model. While [188] also relies on a histogram-based representation, their approach differs fundamentally from ours in that it does not assign a visual interpretation to the codebook. Here, we explore two strategies to provide a visual interpretation to BoW networks, including one that allows for end-to-end training. Furthermore, we demonstrate that our visual codewords can be exploited for adversarial and out-of-distribution example detection.

Adversarial attack detection. When the sensitivity of deep networks to adversarial attacks was identified [246], initial works focused on developing defense strategies [79, 137], aiming to robustify the networks. However, these defenses were typically found to be vulnerable to optimization-based techniques [26]. Therefore, the research focus has increasingly shifted towards detecting adversarial samples, thus allowing one to discard them instead of attempting to be robust to them. In this context, [177] proposed to use a separate subnetwork to detect adversarial examples; [81] relied on knowledge distillation and Bayesian uncertainty to train a simple logistic regression detector; [167] exploited a measure of local intrinsic dimensionality to identify the adversarial examples. Here, we show that we can outperform all these methods by learning to compare the input image with the visual representation of the most highly activated codeword in our interpretable BoW network.

Another problem related to adversarial sample detection is that of identifying out-of-distribution (OOD) examples. This task has been addressed by training a detector on the softmax scores of a network [93], extended in [151] by an additional pre-processing of the network input. In [142], a unified framework for adversarial and OOD sample detection was introduced. As evidenced by our experiments, our approach also applies to both tasks and outperforms the state of the art in each. It further gives the possibility to visually analyze the attacks, opening the door to human intervention in the detection process.

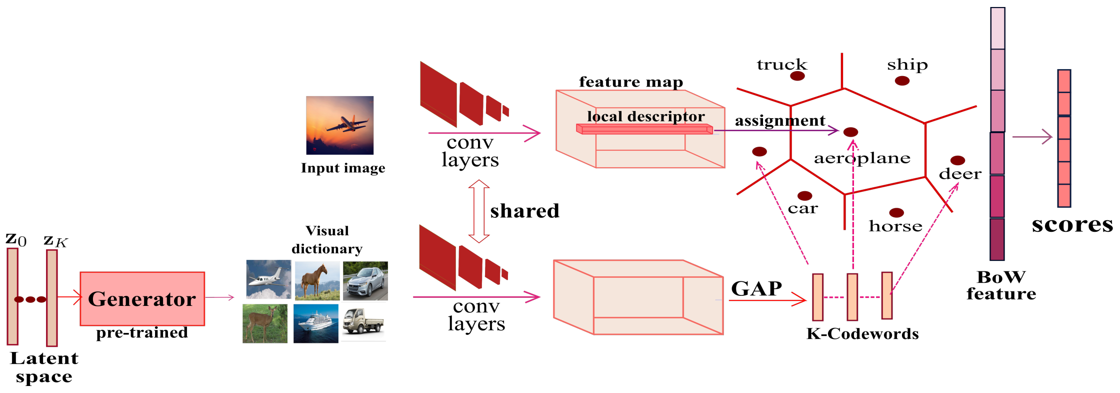


Figure 3.2 – **Interpretable BoW network.** A BoW network, shown in the top portion, relies on a dictionary to form a histogram-like image representation. Our second approach to providing it with interpretability relies on a pre-trained GAN to generate visual codewords, which are passed through same backbone as that in the BoW network to obtain the codewords. The decision of model can then be interpreted by analyzing the visual codeword with highest activation.

3.3 Method

In this section, we first introduce our approach to leveraging BoW networks to obtain a visual interpretation of a network's prediction. We then show how to exploit the resulting visual codebook to detect adversarial and out-of-distribution (OOD) examples.

3.3.1 Interpretable BoW Networks

Our goal is to obtain a visual interpretation of a network's prediction. To this end, we propose to analyze the last feature representation of the network. Instead of relying on an expensive nearest-neighbor search between the test image features and the training ones, we advocate for the use of histogram-based networks, such as those of [5, 204, 247, 266]. Below, we first formalize the BoW network we use, and then introduce two strategies to providing it with interpretability.

BoW network. Formally, let $\mathbf{I}$ be an image input to a CNN, and $\mathbf{X} \in \mathbb{R}^{W \times H \times D}$ be the feature map output by the CNN's last convolutional layer, with spatial resolution $W \times H$ and D channels. A typical BoW network treats each D-dimensional vector $\mathbf{x}_i$ in

$\mathbf{X}$ as a local feature and relates it to a codebook. Here, however, we are interested in having a single global visual interpretation of the prediction, which will be better suited for adversarial example detection. We therefore rely on global average pooling to obtain a single D-dimensional feature vector $\mathbf{x}$ from $\mathbf{X}$.

We then pass this vector to a BoW layer that, given a codebook $\mathbf{B}$ with K codewords, produces a K-dimensional representation of the form

$$\mathbf{h}(\mathbf{x}) = [a_0(\mathbf{x}), a_1(\mathbf{x}), \cdots, a_K(\mathbf{x})]^T \ , \tag{3.1}$$

where $a_k(\mathbf{x})$ is given by

$$a_k(\mathbf{x}) = \frac{e^{-\alpha\|\mathbf{x}-\mathbf{b}_k\|^2}}{\sum_{k'} e^{-\alpha\|\mathbf{x}-\mathbf{b}_{k'}\|^2}} \ . \tag{3.2}$$

This value represents the assignment of $\mathbf{x}$ to codeword $\mathbf{b}_k$. Note that, in the classical BoW formalism, the assignments are binary, i.e., $a_k = 1$ for a single k and 0 for the other indices. However, within our deep learning context, for differentiability, we relax them as soft assignments, with α a hyper-parameter defining the softness. The resulting BoW vector $\mathbf{h}(\mathbf{x})$ then acts as input to the final classification layer of the network.

In essence, the soft assignments a_k encode the similarity in feature space of the input image with the codewords. Since these assignments act as input to the classifier, they are key to determining the class of the input image. One can therefore analyze the network's prediction by studying the codeword with highest activation. To make this analysis interpretable, below, we introduce two strategies to providing the codewords with a visual representation.

Providing interpretability. Our first approach to obtaining a visual codeword representation is simple: Because a codeword $\mathbf{b}_k$ lies in the same space as the network's final features, we propose to retrieve the training image $\mathbf{I}^j$ whose final feature vector $\mathbf{x}^j$ is closest to the codeword. This is achieved by a nearest-neighbor search in a post-training stage, and, for each codeword $\mathbf{b}_k$ yields a visual representation $\mathbf{V}_k$ directly coming from the training set. Note that, in contrast to comparing the final feature vector $\mathbf{x}$ of the test sample with all training features, this search needs to be performed only once after training the BoW model, not once for every test image.

This nearest-neighbor strategy nonetheless suffers from the fact that, for some codewords, the closest training image might still be relatively far, and thus not be a good representative of the codeword. To circumvent this, and further incorporate the codewords' visual interpretation in the training stage, as a second strategy, we propose to exploit a generative adversarial network (GAN). This process is illustrated in Fig. 3.2. Specifically, the images obtained from the generator of a pre-trained GAN are passed through the same backbone network as that of our BoW model, and the resulting average-pooled features taken as codewords. As a result, the set of K generated images $\{\mathbf{V}_k\}_{k=1}^K$ form a visual dictionary, and each codeword $\mathbf{b}_k$ in the codebook $\mathbf{B}$ is directly associated to a visual interpretation $\mathbf{V}_k$.

Whether using the first or the second strategy, at test time, we can obtain an interpretation of the network's prediction by observing the visual codeword $\mathbf{V}_{k^*}$ associated with the most highly activated codeword, that is,

$$k^* = \underset{k}{\operatorname{argmax}}\, a_k(\mathbf{x}) \ , \tag{3.3}$$

with $\mathbf{x}$ the last feature vector of the test sample. For a sample from class c, this visual codeword will typically correspond to an image of the same class. It will, moreover, depict characteristics similar to that of the input image, and, as shown in our experiments, different codewords from the same class c will focus on different characteristics.

Training. To train our BoW model, we follow the same strategy as in the BoW network literature [5, 75]. That is, we first train the backbone network, without BoW layer but with a softmax classifier. We then performs K-means clustering on the last features $\{\mathbf{x}^j\}_{j=1}^N$ of the N training samples to obtain the codebook. Specifically, we define an equal number of codewords S for each class. Thus, for a C-class dataset, we obtain $K = CS$ codewords, and force each group of S codewords to come from the same class during the clustering procedure. Finally, we incorporate the BoW layer to the network and train the resulting BoW model by replacing the backbone network classification layer with a layer that maps the BoW representation to the class labels and using the standard cross-entropy loss.

For our nearest-neighbor interpretability strategy, we can directly use the resulting BoW model. However, for the GAN-based one, which defines an end-to-end model, we aim to further incorporate the codewords' visual interpretation in the training process. To this end, we treat the K-means codebook described above as an initial one, denoted as $\mathbf{B}_0$. To obtain a visual representation of this codebook, we then make use of a GAN pre-trained on the dataset of interest. Specifically, for of each codeword in $\mathbf{B}_0$, we optimize the input $\mathbf{z}$ of the generator, whose weights are fixed, so as to generate an image that, when passed through the backbone network, yields features that are close to the codeword in the least-square sense. That is, formally, for each codeword $\mathbf{b}_{0,k}$ in $\mathbf{B}_0$, we solve

$$\mathbf{z}_k^* = \underset{\mathbf{z}}{\operatorname{argmin}}\, \|f(g(\mathbf{z})) - \mathbf{b}_{0,k}\|^2 \ , \tag{3.4}$$

where $g(\cdot)$ represents the generator, and $f(\cdot)$ the backbone network up to the last average pooling operation. We then take our codebook $\mathbf{B}$ to be the set of generated features $\{f(g(\mathbf{z}_k^*))\}$, and train the resulting BoW model.

Note that, while this training procedure may seem costly, this has no effect on the computational cost at test time. Indeed, inference only involves a forward pass through our BoW network, which, compared to the backbone network, only requires us to store the additional codebook **B**; that is, the generator network is not needed anymore.

3.3.2 Detecting Adversarial Examples

By providing a visual interpretation of a network's prediction, our interpretable BoW network can be leveraged to detect adversarial examples. The reasoning behind this is the following: Typically, adversarial attacks aim to add the smallest amount of perturbation to an image so that the network misclassifies it. While this perturbation should be imperceptible to humans, it should strongly affect the resulting deep representation. In our case, this representation is the BoW one, and, for misclassification to occur, the most highly activated codeword should typically be associated to the wrong class. As such, its visual representation should look significantly different from the adversarial example. We therefore propose to train an adversary detector that, given two input images, predicts whether they belong to the same class or not.

Formally, our detection framework, proceeds as follows. An image $\mathbf{I}$, adversarial or not, is passed through our interpretable BoW network, and we retrieve the visual codeword $\mathbf{V}_{k^*}$ corresponding to the highest activation using Eq. 3.3. We then pass $\mathbf{I}$ and $\mathbf{V}_{k^*}$ to our adversary detector, which outputs a binary label indicating whether the two images belong to the same class or not. If they don't, then $\mathbf{I}$ is deemed an adversarial example.

Our detector is a two-stream network that extracts features for the two images independently. To train this network, we make use of the contrastive loss [86], which aims to make the Euclidean distance between pairs of mismatched images larger than a margin $m = 1$, while minimizing that of matching pairs. Detection is then performed by comparing the Euclidean distance to the margin.

While we train the detector to identify adversarial samples, it can also be used to detect OOD ones, even *without* any re-training. The intuition remains unchanged: An OOD sample will activate a codeword whose visual representation looks different from the input image. The detection procedure is thus the same as for adversarial samples.

3.4 Experiments

We now empirically evaluate our interpretable BoW networks by we first analyzing the learned visual dictionaries and then demonstrating their benefits to detect adversarial and OOD samples. To this end, we used the standard datasets employed for adversarial sample detection, that is, MNIST [141], F-MNIST [279], CIFAR-10 [135], SVHN [194].

Implementation details. For our comparisons to be meaningful, we rely on the same backbone architecture as in [167] on each dataset. This architecture is first trained with a softmax classifier for 50-100 epochs and used to compute the initial codebook $\mathbf{B}_0$. In parallel, we train a GAN [78][1] for 100k iterations. Following the procedure described in Section 3.3.1, we obtain our interpretable codebook $\mathbf{B}$ either by nearest neighbor search (BoW-NN) or using the pre-trained GAN generator (BoW-GAN). We then train the classification layer of our interpretable BoW model for 40 epochs.

In all our experiments, we set $\alpha = 100$ in the BoW soft-assignment policy of Eq. 3.2, and use the Adam [124] optimizer with a learning rate of 0.001 and a decay rate of 0.1 applied every 20 epochs. Note that our goal here is not to advocate for the superior performance of the BoW model, but rather for its use to provide a visual interpretation, as discussed below.

3.4.1 Visualizing BoW Codewords

Each codeword in our BoW model is directly associated with an image $\mathbf{V}_k$. In Fig. 3.3, we visualize some of these images for our two interpretability strategies, NN and GAN. This confirms that the learned codewords nicely cover the diversity of the classes in each dataset. For example, in F-MNIST, each garment appears in a variety of sizes and styles. Similarly, in CIFAR-10, each object appears in different colors, orientations and in front of different backgrounds. Furthermore, each one of these images retains the semantic meaning of the class label for which it was generated. This will prove key to the success of our adversarial sample detector, as discussed in the next section.

In the top row of Fig. 3.4, we show the codewords with highest activation for a few correctly-classified images of each dataset. Note that, in most cases, the corresponding codeword has the same semantic meaning as the input image, not only because it corresponds to the same class, but also because it depicts a visually similar content, e.g., in terms of color and orientation. Furthermore, in the second row of Fig. 3.4, we show activated codewords for misclassified samples. Note that these samples activate visually dissimilar codewords, and thus mistakes can be analyzed. For example, on MNIST, the digit 7 (second image in the *misclassified* portion) is misclassified as 4 due to its similarity with a codeword of class 4.

3.4.2 Detecting Adversarial Samples

In all experiments, we assume that ***attacker has full knowledge about classifier model weights***. In other words, all our experiments are conducted in white-box threat setting w.r.t. classifier. With that assumption, we classify the attacks w.r.t. to *detector* in three settings:

[1]For CIFAR-10, we used a WGAN [82] because a standard GAN led to poor visual quality.

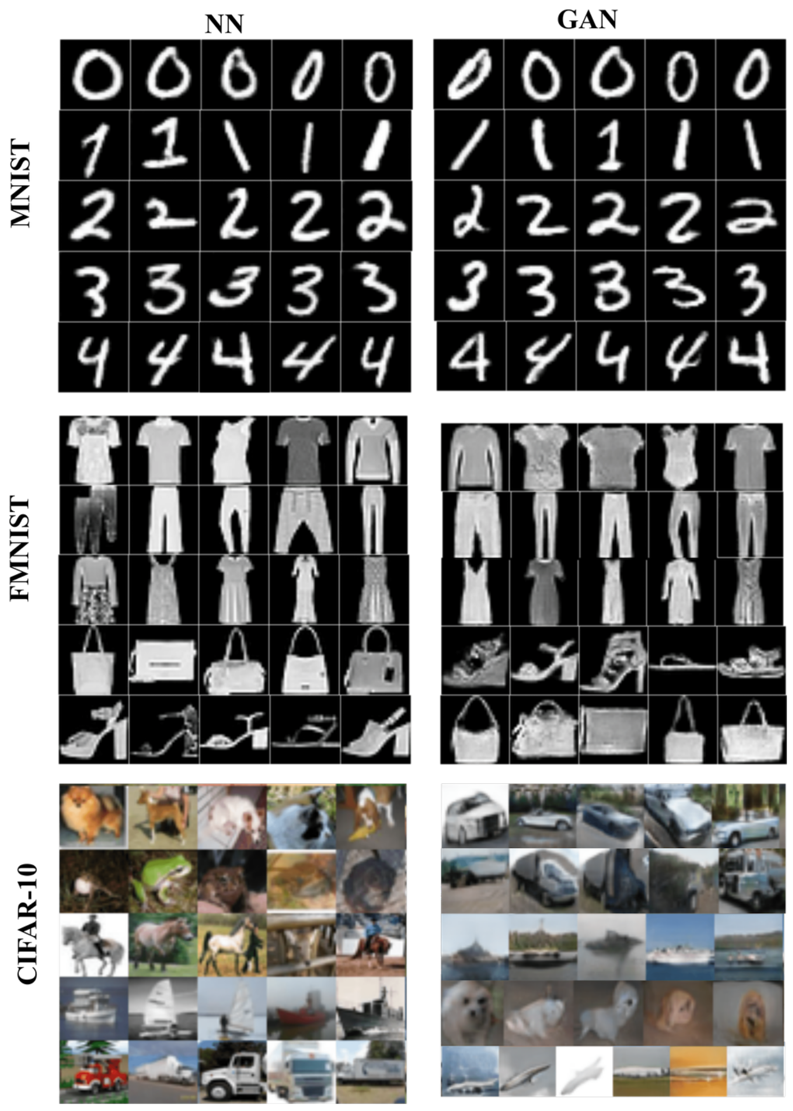


Figure 3.3 – **Visual interpretations of the BoW codebooks obtained using NN search or a GAN.**

- **Black-box detector setting.** Adversary neither has access to detector weights nor knowledge about detection mechanism.
- **Gray-box detector setting.** Adversary does not have access to detector weights but is aware of the detection mechanism
- **White-box detector setting.** Adversary has access to both detector weights and its detection mechanism

We make use of the state-of-the-art attack methods, FGSM [79], BIM-a [137], BIM-b [137], and CW [26], to evaluate the effectiveness of our adversarial example detection

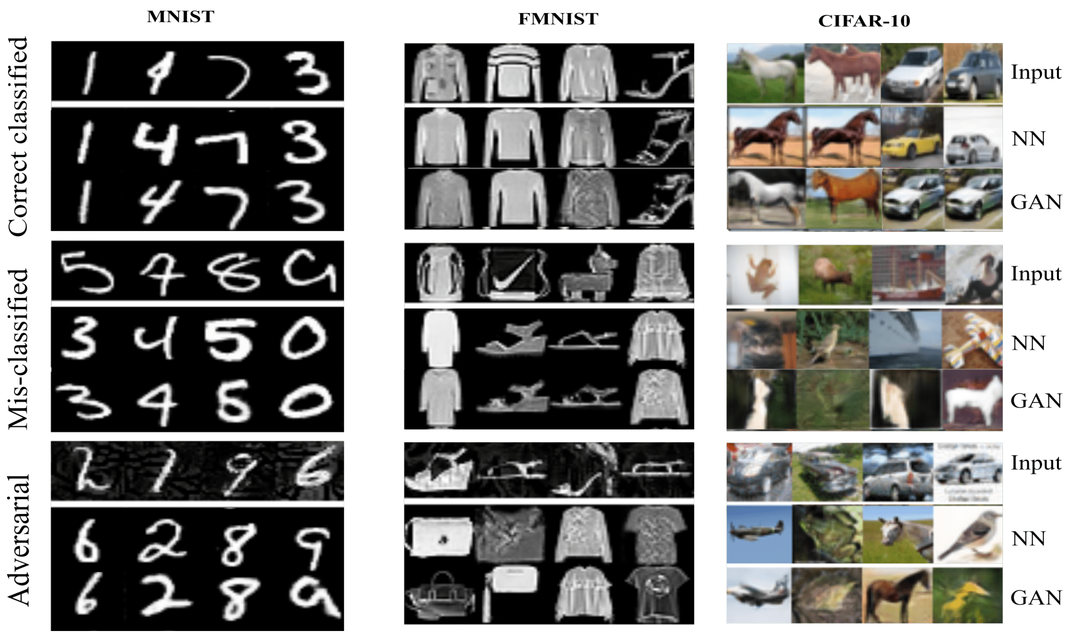


Figure 3.4 – **Visualization of the most highly activated codeword for normal and adversarial samples.** We show both correctly classified and misclassified normal samples. The top row within a block shows the input samples and the next two rows the corresponding NN and GAN codewords, respectively. The adversarial examples were obtained with a BIM-a attack.

strategy. To this end, following common practice [167], we discard the images that were misclassified by the original networks from this evaluation. We focus on white-box attacks, where the attacker has access to the exact model it aims to fool. There are two ways to attack our BoW model: One can generate adversarial examples either for the BoW model itself, or for the backbone network. We refer to the latter as transferred BoW (T-BoW) attacks. Note that, the attacks directly targeting our BoW model are significantly less successful than those on the backbone network (T-BoW attacks).

Before we turn to attack detection, we first validate our intuition that adversarial samples will activate codewords corresponding to the wrong classes, and that these codewords will be visually dissimilar to the input image. To this end, in the bottom row of Fig. 3.4, we show the most highly activated codeword images for a few successful adversarial attacks. Note that these images differ semantically and visually from the input, which will facilitate attack detection.

Black-box detector setting

We now evaluate the effectiveness of the detection strategy introduced in Section 3.3.2 in black-box detector setting. To this end, we train our detector using adversarial examples generated by the white-box attacks on the classifier as discussed above. Specifically, we report results obtained with two training strategies. *Strategy 1*, which is commonly used [167], consists of defining a balanced training set comprised of normal images (positives) and their adversarial counterparts (negatives). A drawback of this strategy, however, is that many attacks are unsuccessful with our BoW model, and considering such samples as negatives essentially adds noise to our training process, since unsuccessful adversarial samples will typically activate a codeword that is similar to the input image. To overcome this, we therefore propose *Strategy 2*, which consists of using only the successful adversarial examples as negatives during training. The resulting training set, however, is then imbalanced.

Comparison with the state of the art. We compare our approach with the state-of-the-art BU [81], LID [167] and MD [142] methods, which have proven more robust than the earlier detection strategy [64]. In Tables 3.1 and 3.2, we report the area under the ROC curve (AUROC) for BU, LID, MD and our method, using both Strategy 1 and Strategy 2, for white-box direct BoW and T-BoW attacks, respectively. Note that we outperform the state of the art in most cases, by a particularly large margin when the results are not already saturated, such as with BIM-a on SVHN and CIFAR-10 and with FGSM on CIFAR-10. Note that our NN and GAN strategies yield similar results on the simpler MNIST, F-MNIST and SVHN datasets, but the GAN one performs better on CIFAR-10. This, we believe, is due to the largest diversity of this dataset, making the nearest training image a relatively poor representative of the codeword. By contrast, the GAN can generate an image whose last features are very close to the codeword.

Similarly to [167], we evaluate how a detector trained for a specific attack generalizes to other ones. In Table 3.3, we compare the generalizability of our approach, LID and MD. For each method, we show the results of the model that was trained on the attack that makes it generalize best to the other ones. Note that the performance of our detector is virtually unaffected. While LID and MD also generalize well in some cases, our method outperforms them by a large margin in several scenarios, such as CW attacks.

Gray-box detector setting

In the previous set of experiments, we have worked under the assumption that the attacker only had access to our BoW model, but not to the detector. Here, we remove this assumption, and study in more challenging scenarios. In this section, we assume that the attacker knows our detection strategy, but not the detector model we use.

To evaluate the robustness of our approach in the gray-box scenario where the attacker is aware of our detection scheme, we apply an adaptive CW attack strategy similar to the ones used in [25, 167] to attack the KD and LID detectors, respectively. To this end,

Dataset	Feature	FGSM	BIM-a	BIM-b	CW
	KD+BU	95.22 / 94.11	82.54 / 72.80	82.17 / 72.43	50.17 / 60.01
	LID	92.66 / 91.18	82.24 / 61.90	83.06 / 75.61	**51.81** / 68.46
MNIST	MD	100.0 / 100.0	100.0 / 100.0	100.0 / 100.0	50.94 / 67.18
	Ours-NN	100.0 / 100.0	100.0 / 100.0	100.0 / 100.0	50.85 / 99.02
	Ours-GAN	100.00 / 100.0	100.0 / 100.0	100.0 / 100.0	50.87 / **100.0**
	KD+BU	99.33 / 99.36	91.35 / 87.32	89.43 / 85.39	69.68 / 65.83
	LID	93.88 / 93.93	86.95 / 81.06	86.83 / 81.11	73.23 / 70.84
F-MNIST	MD	98.02 / 98.45	88.96 / 85.49	91.43 / 89.38	72.71 / 72.68
	Ours-NN	100.0 / 100.0	99.98 / 99.95	100.0 / 99.97	**84.50** / 97.59
	Ours-GAN	**100.0 / 100.0**	**100.0 / 100.0**	**100.0 / 100.0**	83.97 / **97.96**
	KD+BU	78.21 / 74.24	78.37 / 73.97	67.96 / 71.46	88.68 / 88.62
	LID	99.88 / 99.25	83.45 / 84.17	85.76 / 90.42	91.23 / 91.40
SVHN	MD	99.65 / 99.73	82.26 / 82.93	87.81 / 93.43	88.92 / 89.55
	Ours-NN	**100.0** / **100.0**	98.66 / **96.98**	**100.0 / 100.0**	96.05 / **96.75**
	Ours-GAN	99.9 / **100.0**	**98.71** / 96.16	99.9 / **100.0**	**96.52** / 96.49
	KD+BU	72.79 / 69.66	86.23 / 85.69	60.23 / 62.52	93.74 / 93.74
	LID	89.67 / 89.26	85.40 / 85.02	80.55 / 82.79	93.57 / 93.17
CIFAR-10	MD	97.94 / 97.93	84.68 / 84.17	84.47 / 84.45	91.64 / 91.59
	Ours-NN	97.14 / 98.77	93.12 / 95.65	95.12 / 98.88	94.12 / 94.61
	Ours-GAN	**99.37 / 99.97**	**93.90 / 97.47**	**99.90 / 99.96**	**96.52 / 96.35**

Table 3.1 – **Attacks on BoW model in black-box setting.** Left and right numbers correspond to AUROC scores for Strategy 1 and Strategy 2, respectively.

Dataset	Feature	FGSM	BIM-a	BIM-b	CW
	KD+BU	93.73 / 93.63	86.09 / 83.09	79.22 / 79.22	80.64 / 80.64
	LID	99.17 / 99.15	99.75 / 99.75	95.16 / 96.16	99.01 / 99.03
MNIST	MD	**100.0 / 100.0**	100.0 / 100.0	100.0 / 100.0	**99.96** / 99.96
	Ours-NN	100.0 / 100.0	100.0 / 100.0	**100.0 / 100.0**	99.94 / 99.97
	Ours-GAN	100.00 / 100.0	100.0 / 100.0	**100.0 / 100.0**	99.85 / **100.0**
	KD+BU	96.99 / 97.03	92.60 / 92.60	97.74 / 97.74	93.27 / 93.27
	LID	95.68 / 95.68	95.95 / 95.95	95.24 / 95.24	97.31 / 97.31
F-MNIST	MD	98.23 / 98.23	88.96 / 94.04	91.43 / 99.19	72.71 / 95.89
	Ours-NN	100.0 / 100.0	99.98 / 99.95	99.90 / 99.87	**98.35 / 98.20**
	Ours-GAN	**100.0 / 100.0**	**99.98 / 100.0**	**100.0 / 100.0**	98.26 / 97.96
	KD+BU	85.04 / 85.08	88.06 / 88.06	99.99 / 99.99	92.85 / 92.85
	LID	99.88 / 99.32	88.45 / 84.17	98.76 / 99.93	94.23 / 94.23
SVHN	MD	99.79 / 99.82	80.95 / 81.31	99.94 / 99.94	92.48 / 92.62
	Ours-NN	**100.0 / 100.0**	**96.85 / 96.47**	99.97 / **100.0**	95.43 / 96.52
	Ours-GAN	99.99 / **100.0**	96.66 / 96.25	**100.0 / 100.0**	**95.61 / 96.86**
	KD+BU	75.75 / 74.67	80.02 / 79.69	99.12 / 99.24	96.06 / 96.05
	LID	87.16 / 87.92	82.17 / 81.77	99.91 / 99.91	**97.54 / 97.56**
CIFAR-10	MD	97.62 / 97.59	73.85 / 76.36	**99.99 / 99.99**	96.68 / 96.36
	Ours-NN	98.56 / 99.01	95.42 / 95.78	99.12 / 99.34	95.47 / 96.54
	Ours-GAN	**99.80 / 99.71**	**96.77 / 96.87**	99.77 / 99.85	96.70 / 97.24

Table 3.2 – **Attack on T-BoW model in black-box setting.** Left and right numbers correspond to AUROC scores for Strategy 1 and Strategy 2, respectively.

we modify the objective of the CW attack as

$$\arg\min_{\mathbf{I}_{adv}} \|\mathbf{I} - \mathbf{I}_{adv}\|_2^2 + \alpha \cdot \Big(\ell(\mathbf{I}_{adv}) + \|\phi(\mathbf{I}) - \phi(\mathbf{I}_{adv})\|_2^2\Big). \tag{3.5}$$

The first two terms correspond to the original CW attack, with α balancing the amount of perturbation and the adversarial strength, that is, how strongly one forces the adversarial image to be misclassified. The last term directly reflects our detection strategy and encourages the BoW representation of the real, $\phi(\mathbf{I})$, and adversarial, $\phi(\mathbf{I}_{adv})$, images to

Dataset	Method	Train	FGSM	BIM-a	BIM-b	CW
MNIST	LID	FGSM	91.18	65.48	64.42	29.05
	MD	FGSM	**100.0**	99.87	99.98	34.72
	Ours-GAN	BIM-a	**100.0**	**100.0**	**100.0**	**97.56**
F-MNIST	LID	FGSM	93.88	82.24	82.58	65.03
	MD	FGSM	**98.45**	87.81	90.48	64.04
	Ours-GAN	CW	97.39	**97.13**	**95.85**	**97.96**
SVHN-10	LID	FGSM	99.25	77.54	79.77	75.16
	MD	FGSM	**99.73**	69.49	78.76	79.10
	Ours-GAN	BIM-a	91.41	**96.25**	**91.30**	**94.73**
CIFAR-10	LID	FGSM	89.26	66.55	68.33	66.05
	MD	FGSM	**97.93**	65.32	80.34	60.60
	Ours-GAN	BIM-a	86.90	**97.47**	**95.02**	**95.44**

Table 3.3 – **Generalizing to different attacks in black-box detector setting.** We compare the AUROC scores of LID, MD and our detector in the scenario where the detectors were trained for a specific attack, but tested on different ones. These results were obtained with Strategy 2 and direct BoW attacks.

be similar. The rationale behind this is that the attack then aims to find an adversarial perturbation such that the sample is not only misclassified, but also has a representation close to that of the real image, thus breaking the premise on which our detector is built.

Dataset	Attack success rate	Detector AUROC
F-MNIST	33.11	96.22
SVHN	94.58	96.54
CIFAR-10	99.95	97.47

Table 3.4 – **Gray-box setting.** We report AUROC scores of our detector in gray-box setting. Our detector remains robust to an attacker that knows our detection strategy.

In Table 3.4, we report both the success rate of this adaptive attack on our BoW model and the AUROC of our detector, trained with Strategy 2. Note that our detector still yields high AUROC, thus showing that it remains robust to an attacker that knows our detection strategy.

White-box detector setting

We now evaluate the robustness of our method in the case where the attacker has access to all our models, that is, the BoW model, the GAN and the adversarial sample detector. Note that access to the parameters of the generator and detector networks is not a mild assumption since information about the training data is required to compute them.

To attack our complete framework, we generate adversarial images $\mathbf{I}_{adv}$ in a two-step fashion. First, we attack the BoW classifier to generate an intermediate adversarial image $\mathbf{I}^0_{adv}$ that activates codeword j corresponding to image $\mathbf{V}_j$ in the visual codebook. Second, we attack the detector to misclassify $\mathbf{V}_j$ as being similar to the input image.

Since our detector relies on the distance between the input image and the codeword in feature space, fooling it can be achieved by finding a perturbation that solves the optimization problem

$$\min_{\mathbf{I}_{adv}} \left\|\mathbf{I}^0_{adv} - \mathbf{I}_{adv}\right\|_2^2 + \alpha \cdot \|\gamma(\mathbf{V}_j) - \gamma(\mathbf{I}_{adv})\|_2^2 \ , \tag{3.6}$$

where $\gamma(\cdot)$ is the feature-extraction part of our detector.

Dataset	Classifier attack	Detector attack	Attack success rate on detector (%)
MNIST	FGSM	FGSM	0.00
	CW	CW	100.00
F-MNIST	FGSM	FGSM	2.8
	CW	CW	100

Table 3.5 – **White-box detector attacks.** An attacker that has access to our BoW network, GAN and detector can indeed be successful when using the CW attack, but not the FGSM one.

We found that, as is, our detector is indeed vulnerable to such an attack. As observed from Table 3.5, the adversary through modified CW strategy can find pertubations such that it fools both the classifier and the detector with 100% attack success rate. While it is desirable to have a detector work in the strongest white-box setting; from a practical perspective, however, attacker having access to detector weights in quite rare, and our work shows that our detector remains fairly effective than competitors in black-box and gray-box settings.

In retrospect, at the time of writing this thesis, we found two papers [36, 57] after the publication of our work, on the similar spirit as our method. Both the works [36, 57] uses a visual representation of k-nearest neighbors to defend against the adversarial attack. In particular, [57] uses the web-scale image database containing a billion images and finds the k-nearest images to the given test image by calculating the L_2 distance on pre-computed features. In contrast, we set $k = 1$ since we choose the highest activated codeword for similarity matching. Further, we restrict the dataset used for the nearest neighbor to the training set, typically around 50K samples. Nevertheless, it was shown in [36] that the proposed defense breaks when the attacker has access to the web-scale database and to the exact value of k, an observation that we also encountered with $k = 1$ and using a small training set for codeword visual representation. Furthermore, [36] uses a retrieval engine during training time to augment the original training set with the k-nearest images in the feature space. Similar to the observation in [57], [36] reported that its approach is compromised against white-box adversaries, but effective against black-box attacks.

3.4.3 Detecting Out-of-distribution Samples

We now evaluate the use of our approach to detect OOD samples. Following the setup of [93], we perform experiments using either SVHN [194] or MNIST [135] as training datasets from which the in-distribution samples are drawn. The goal then is to detect OOD samples coming from other datasets, such as LSUN [280], TinyImageNet [46], Omniglot [139] and Not-MNIST [22]. We consider two settings: In the first, the detection method does not see *any* OOD samples during training, but has access to adversarial examples generated by the BIM-a attack; in the second, the detector has access to 1000 images from the OOD dataset.

In Table 3.6, we compare the results of our approach with those of the baseline method [93], ODIN [151] and MD [142]. Note that we clearly outperform them, both when we see OOD samples during training and in the more realistic case where we don't. We believe that this demonstrates the generality of our approach.

In Dataset	Out Dataset	Adversarial examples	OOD samples
		Baseline [93] / ODIN [151] / MD [142] / **Ours-GAN**	
SVHN	CIFAR-10	87.44 / 87.54 / 95.34 / **97.26**	87.92 / 87.46 / 95.99 / **99.14**
	LSUN	89.06 / 89.06 / 99.34 / **99.87**	89.06 / 89.52 / 99.16 / **99.98**
	TinyImageNet	89.97 / 90.42 / 98.79 / **99.76**	89.97 / 88.96 / 99.01 / **99.93**
MNIST-10	Not-MNIST	77.11 / 77.69 / 85.82 / **97.44**	77.23 / 77.69 / 99.56 / **99.98**
	OMNIGLOT	82.11 / 82.06 / **99.36** / 97.01	82.24 / 82.06 / 99.70 / **100.0**
	CIFAR	79.84 / 79.77 / **99.66** / 99.21	79.84 / 79.77 / 100.0 / **100.0**

Table 3.6 – **Detecting OOD samples.** We compare the AUROC scores of our detector with the baseline method of [93] and with ODIN [151] and MD [142] in the case where we only observe adversarial samples during training (left) and when we have access to a few OOD samples (right).

Below, we provide additional details regarding our architectures and our experimental results.

3.4.4 Architectures

As base networks for our models, we used the same architectures as in [167]. These architectures are provided in Table 3.7 for all datasets. The test errors on MNIST, FMNIST, CIFAR-10, and SVHN using the softmax classifier and the BoW ones are given in Table 3.8. For the detector in our adversarial example detection approach, we used the architecture shown in Table 3.9 for all datasets, except for CIFAR-10. In this case, we used a ResNet-20 [89][2] due to the higher image variance and background complexity of this dataset.

We also trained a GAN [78] for MNIST and FMNIST, and a WGAN [82] for SVHN and

[2]https://github.com/tensorflow/models/tree/master/official/resnet

Layer	Parameters
Convolution + ReLU	$5 \times 5 \times 64$
Convolution + ReLU	$5 \times 5 \times 64$
MaxPool	2×2
Dense + ReLU	128
Softmax	10

(a) MNIST and FMNIST

Layer	Parameters
Convolution + ReLU	$3 \times 3 \times 64$
Convolution + ReLU	$3 \times 3 \times 64$
MaxPool	2×2
Convolution + ReLU	$3 \times 3 \times 128$
Convolution + ReLU	$3 \times 3 \times 128$
MaxPool	2×2
Dense + ReLU	512
Dense + ReLU	128
Softmax	10

(b) SVHN

Layer	Parameters
Convolution + ReLU	$3 \times 3 \times 32$
Convolution + ReLU	$3 \times 3 \times 32$
MaxPool	2×2
Convolution + ReLU	$3 \times 3 \times 64$
Convolution + ReLU	$3 \times 3 \times 64$
MaxPool	2×2
Convolution + ReLU	$3 \times 3 \times 128$
Convolution + ReLU	$3 \times 3 \times 128$
MaxPool	2×2
Dense + ReLU	1024
Dense + ReLU	512
Softmax	10

(c) CIFAR-10

Table 3.7 – Classifier architectures for different datasets.

CIFAR-10. The architectures of the generator for MNIST and FMNIST are provided in Table 3.10a, and in Table 3.11a for SVHN and CIFAR-10. Similarly, the discriminator for MNIST and FMNIST is provided in Table 3.10b, and the one for SVHN and CIFAR-10 in Table 3.11b.

Dataset	Base	BoW-NN	BoW-GAN
MNIST	99.23	98.66	99.15
FMNIST	92.46	92.43	92.50
SVHN	92.43	92.08	91.81
CIFAR-10	87.54	87.78	87.95

Table 3.8 – **Classification accuracy (in %) of the backbones and BoW networks.** Note that BoW models perform on par with backbone networks, but the BoW ones allow us to obtain a visual interpretation.

Layer	Parameters
Convolution + ReLU	$5 \times 5 \times 32$
MaxPool	2×2
Convolution + ReLU	$5 \times 5 \times 64$
MaxPool	2×2
Fully Connected + ReLU	128
Fully Connected + ReLU	128

Table 3.9 – **Detector** for MNIST, FMNIST and SVHN.

3.4.5 Additional Visualizations

We also provide additional visualizations of activated visual codewords for adversarial samples of MNIST and FMNIST, SVHN and CIFAR-10 in Figs. 3.5 and 3.6.

Layer	Parameters
Input noise, $\mathbf{z} \in \mathbb{R}^{128}$	Nil
Dense + ReLU	$128 \times 4 \times 4$
Deconvolution + ReLU	$5 \times 5 \times 128$
Deconvolution + ReLU	$5 \times 5 \times 64$
Deconvolution + Sigmoid	$5 \times 5 \times 1$

(a) **Generator** for MNIST and FMNIST

Layer	Parameters
Input image, $\mathbf{I} \in \mathbb{R}^{28\times28}$	Nil
Convolution + LeakyReLU, stride=2	$5 \times 5 \times 64$
Convolution + LeakyReLU, stride=2	$5 \times 5 \times 128$
Convolution + LeakyReLU, stride=2	$5 \times 5 \times 256$
Fully Connected + ReLU	1

(b) **Discriminator** for MNIST and FMNIST

Table 3.10 – Generator and Discriminator for MNIST and FMNIST

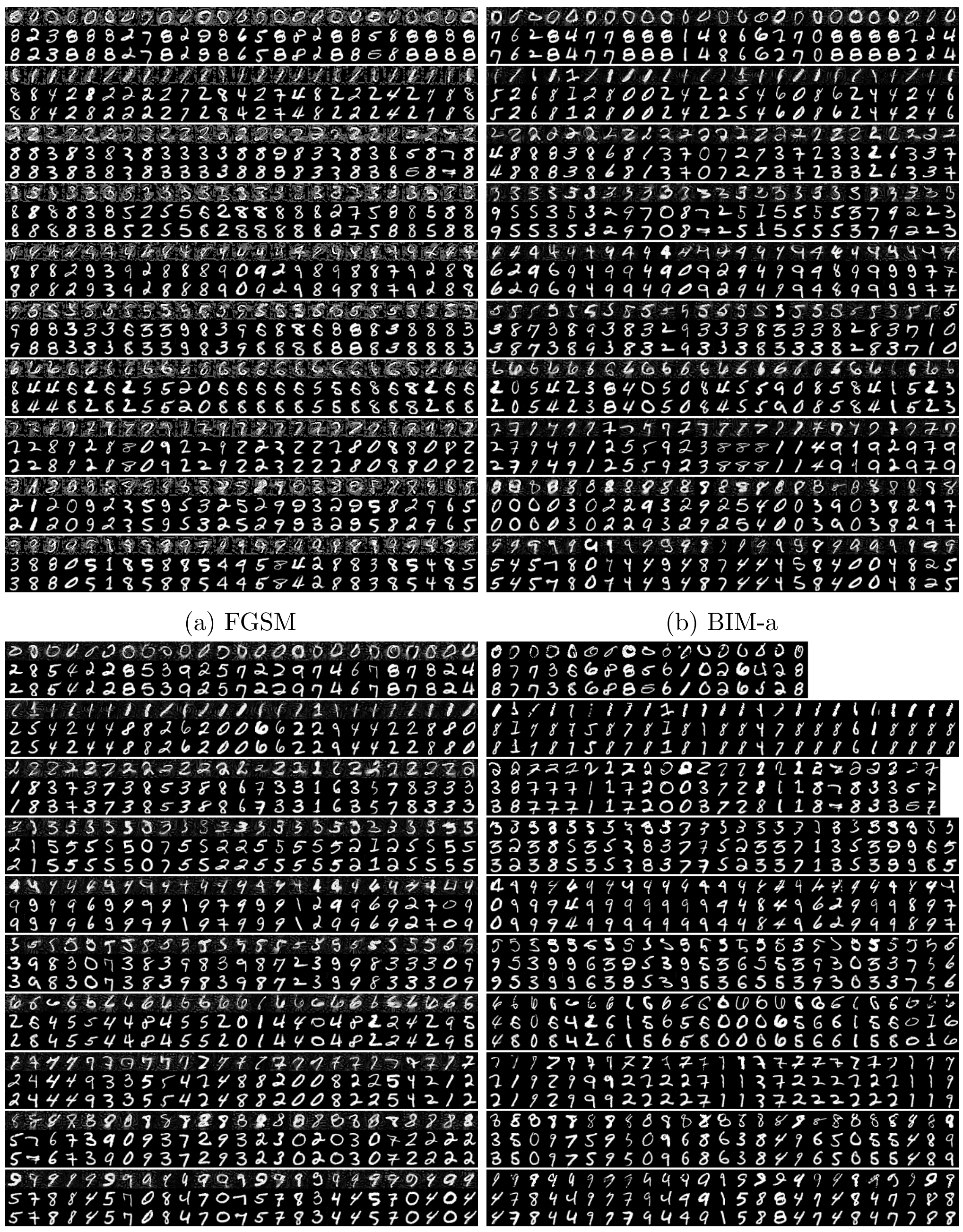
Layer	Parameters
Input noise, $\mathbf{z} \in \mathbb{R}^{128}$	Nil
Dense	$128 \times 4 \times 4$
ResBlock + Up	128
ResBlock + Up	128
ResBlock + Up	128
Convolution + Tanh	$3 \times 3 \times 3$

(a) **Generator**

Layer	Parameters
Input image, $\mathbf{I} \in \mathbb{R}^{32\times32\times3}$	Nil
ResBlock + Down	128
ResBlock + Down	128
ResBlock +Down + GAP	128
Dense	1

(b) **Discriminator**

Table 3.11 – Generator and Discriminator for SVHN and CIFAR-10, similar to the ones used in [82]



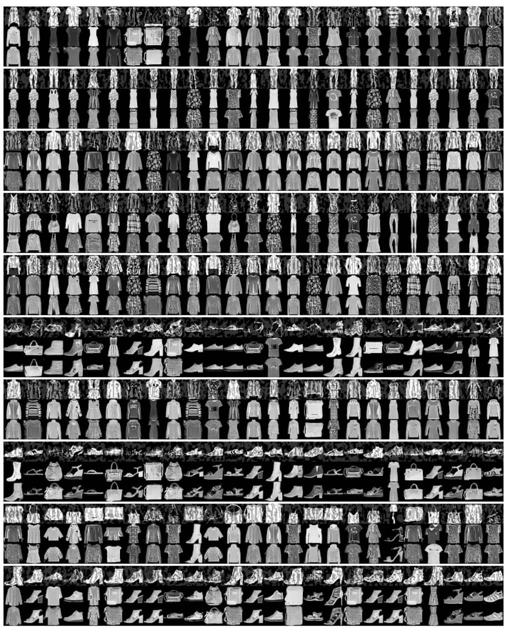
(a) FGSM

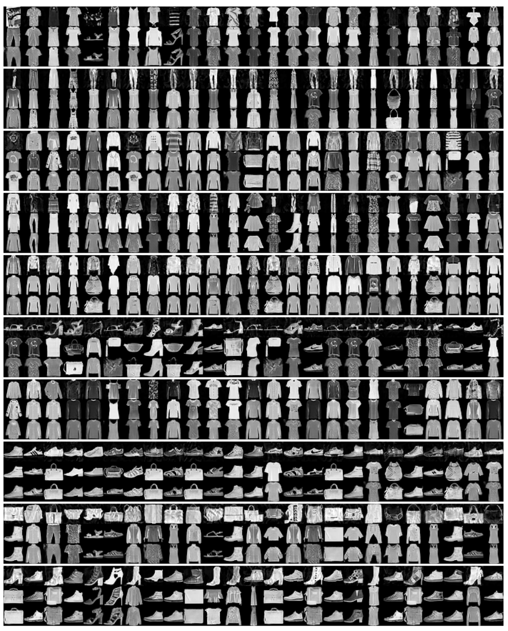
(b) BIM-a

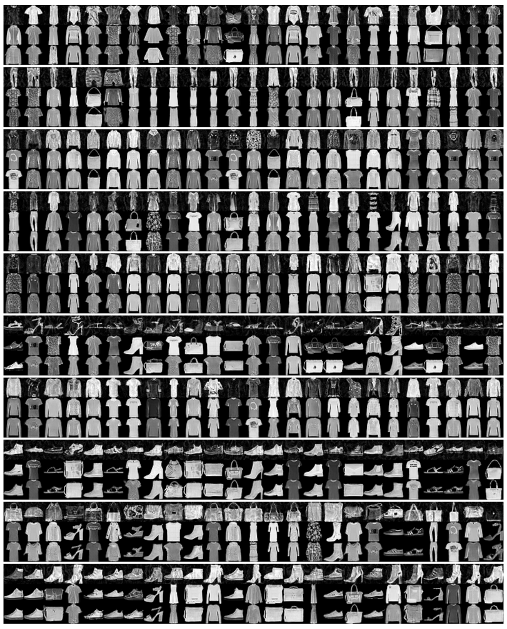
(c) BIM-b

(d) CW

Figure 3.5 – **Most highly activated codewords of adversarial samples obtained with different attack strategies on MNIST test data.** Each three rows within a block show the input image (top) and corresponding codewords from nearest-neighbor and GAN strategies in second and third rows, respectively. Note that the adversarial images and their codeword are dissimilar in most cases. Note that, in the CW case, we have fewer samples in some of the classes because of the low success rate of this attack.


(a) FGSM


(b) BIM-a


(c) BIM-b

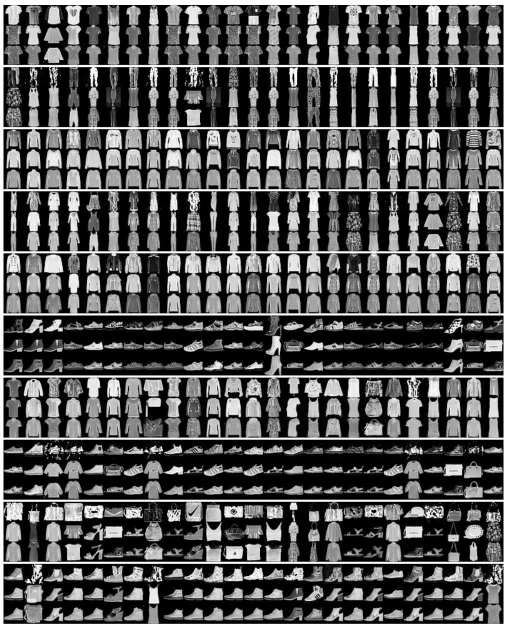

(d) CW

Figure 3.6 – **Most highly activated codewords of adversarial samples obtained with different attack strategies on FMNIST test data.** Each three rows within a block show the input image (top) and corresponding codewords from nearest-neighbor and GAN strategies in second and third rows, respectively. Note that the adversarial images and their codeword are dissimilar in most cases.

3.5 Conclusion

We have introduced a novel approach to interpreting a CNN's prediction, by providing the elements in a BoW codebook with a visual meaning. We have then proposed to leverage the visual representation of these interpretable BoW networks for adversarial example detection. Our experiments have evidenced that (i) our adversary detection strategy outperforms the state-of-the-art ones in gray-box and black-box detector settings; (ii) our framework generalizes to OOD sample detection.

We further showed that the adversary can break our detection framework in a complete white-box detector setting by creating a new adaptive attack. However, our results in gray-box and black-box detector settings are more effective than competitive works, at least for those attacks considered in the experiments. Thus, our approach can be utilized for DNNs deployed in a cloud-based environment in a practical scenario. In the next chapter, we extend the interpretable BoW representations to part-based ones to first understand the reasons for the success of adversarial attacks and then propose a defense to mitigate the impact of the attack.

4 Towards Robust Fine-grained Recognition by Maximal Separation of Discriminative Features

In the previous chapter, we studied the adversarial attacks on general image classification problem. In contrast, the adversarial attacks remain largely unexplored in the context of fine-grained recognition, where the inter-class similarities facilitate the attacker's task. In this chapter, we identify the proximity of the latent representations of *local* regions of different classes in fine-grained recognition networks as a key factor to the success of adversarial attacks. We therefore introduce an attention-based regularization mechanism that maximally separates the latent features of discriminative regions of different classes while minimizing the contribution of the non-discriminative regions to the final class prediction. As evidenced by our experiments, this allows us to significantly improve robustness to adversarial attacks, but without requiring access to adversarial samples. Further, our formulation also improves detection AUROC of adversarial samples over baselines on adversarially trained models.

4.1 Introduction

Deep networks yield impressive results in many computer vision tasks [119, 136, 163, 298]. Nevertheless, their performance degrades under adversarial attacks, where natural examples are perturbed with human-imperceptible, carefully crafted noise [79]. Adversarial attacks have been extensively studied for the task of general object recognition [26, 52, 79, 137, 182, 201], with much effort dedicated to studying and improving the robustness of deep networks to such attacks [117, 168, 285]. However, adversarial attacks and defense mechanisms for fine-grained recognition problems, where one can expect the inter-class similarities to facilitate the attacker's task, remain unexplored.

In this chapter, we therefore analyze the reasons for the success of adversarial attacks on fine-grained recognition techniques and introduce a defense mechanism to improve a network's robustness. To this end, we visualize the image regions mostly responsible for

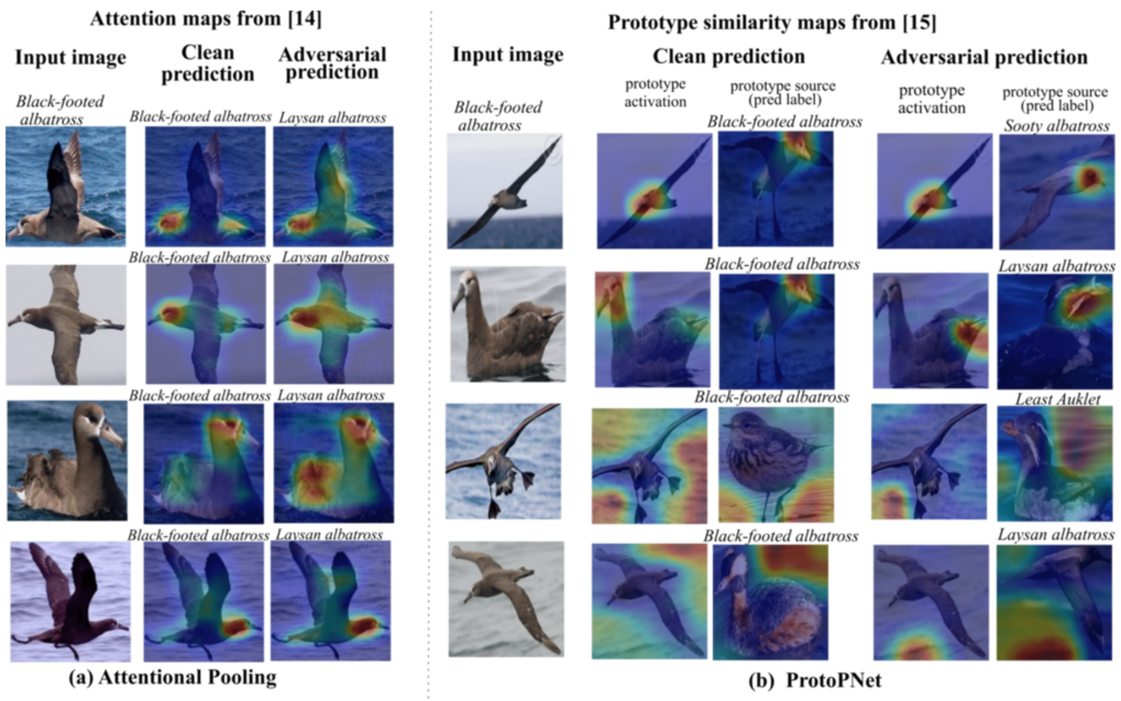


Figure 4.1 – **Interpreting adversarial attacks for fine-grained recognition.** We analyze the attention maps, obtained with [74](a) and [29](b), of four images from the *Black-footed albatross* class. Under PGD attack, these images are misclassified as closely-related bird species, such as *Layman albatross*, because the classifiers focus on either confusing regions that look similar in these classes, such as the bird's beak, or non-discriminative background regions, such as water.

the classification results. Specifically, we consider both the attention-based framework of [74], closely related to class activation maps (CAMs) [316], and the recent prototypical part network (ProtoPNet) of [29], designed for fine-grained recognition, which relates local image regions to interpretable prototypes. As shown in Fig. 4.1, an adversarial example activates either confusing regions that look similar in samples from the true class and from the class activated by the adversarial attack, such as the beak of the bird, or, in the ProtoPNet case, non-discriminative background regions, such as water. This suggests that the latent representations of these confusing regions are close, and that the ProtoPNet classifier exploits class-irrelevant background information. These two phenomena decrease the margin between different classes, thus making the network more vulnerable to attacks.

Motivated this observation, we introduce a defense mechanism based on the intuition that the discriminative regions of each class should be maximally separated from that of the other classes. To this end, we design an attention-aware model that pushes away the discriminative prototypes of the different classes. The effectiveness of our approach is illustrated in Fig. 4.2, where the prototypes of different classes are nicely separated,

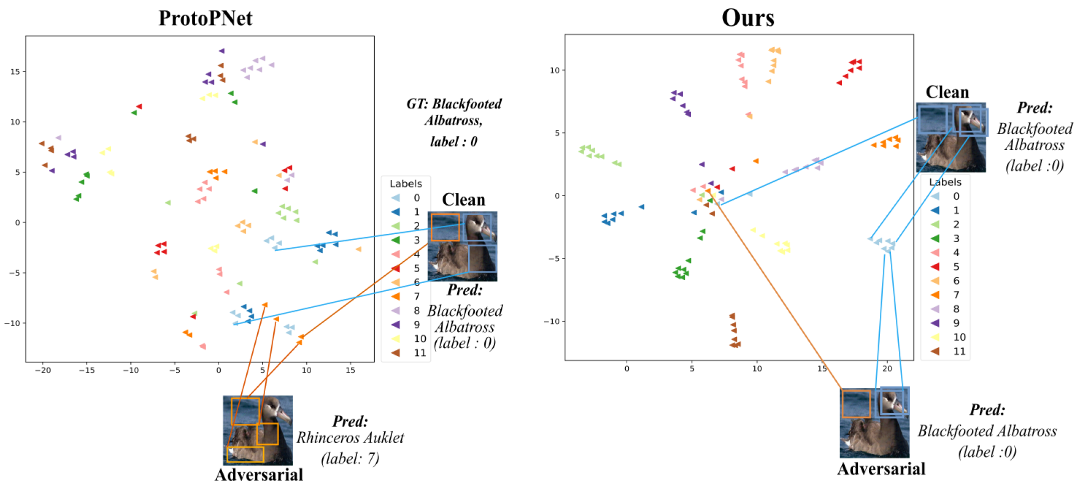


Figure 4.2 – **t-SNE visualization of the prototypes from 12 fine-grained classes of the CUB200 dataset.** In ProtoPNet [29], the prototypes of different classes are not well separated, making the network vulnerable to attacks. By contrast, our approach yields well-separated discriminative prototypes, while clustering the background ones, which, by means of an attention mechanism do not participate the prediction. This complicates the attacker's task.

except for those corresponding to non-discriminative regions. However, by means of an attention mechanism, we enforce these non-discriminative prototypes to play no role in the final class prediction. Ultimately, our approach reduces the influence of the non-discriminative regions on the classification while increasing the magnitude of the displacement in the latent space that the attacker must perform to successfully move the network's prediction away from the true label.

As evidenced by our experiments, our approach significantly outperforms in robustness the baseline ProtoPNet and attentional pooling network, in some cases reaching adversarial accuracies on par with or higher than their adversarially-trained [254, 257] counterparts, but at virtually no additional computational cost.

Our main contributions can be summarized as follows. We analyze and explain the decisions of fine-grained recognition networks by studying the image regions responsible for classification for both clean and adversarial examples. We design an interpretable, attention-aware network for robust fine-grained recognition by constraining the latent space of discriminative regions. Our method improves robustness to a level comparable to that of adversarial training, without requiring access to adversarial samples and without trading off clean accuracy. Further, our approach improves the AUROC score of adversarial example detection by 20% over baselines for adversarial trained networks.

4.2 Related Work

Adversarial Robustness. DNNs were first shown to be vulnerable to adversarial, human-imperceptible perturbations in the context of general image recognition. Such attacks were initially studied in [246], quickly followed by the simple single-step Fast Gradient Sign Method (FGSM) [79] and its multiple-step BIM variant [137]. In [52], the attacks were stabilized by incorporating momentum in the gradient computation. Other popular attacks include DeepFool [182], which iteratively linearizes the classifier to compute minimal perturbations sufficient for the sample to cross the decision boundary, and other computationally more expensive attacks, such as CW [26], JSMA [201], and others [189, 243, 286]. As of today, Projected Gradient Descent (PGD) [168], which utilizes the local first-order network information to compute a maximum loss increment within a specified ℓ_∞ norm-bound, is generally considered as the most effective attack strategy.

Despite a significant research effort in devising defense mechanisms against adversarial attacks [202, 228, 241, 283], it was shown in [9] that most such defenses can easily be breached in the white-box setting, where the attacker knows the network architecture. The main exception to this rule is adversarial training [168], where the model is trained jointly with clean images and their adversarial counterparts. Many variants of adversarial training were thus proposed, such as ensemble adversarial training [254] to soften the classifier's decision boundaries, ALP [117] to minimize the difference between the logit activations of real and adversarial images, the use of additional feature denoising blocks [285], of metric learning [48, 173], and of regularizers to penalize changes in the model's prediction w.r.t. the input perturbations [225, 303]. Nevertheless, PGD-based adversarial training remains the method of choice, thanks to its robustness and generalizability to unseen attacks [26, 79, 137, 182, 253].

Unfortunately, adversarial training is computationally expensive. This was tackled in [234] by recycling the gradients computed to update the model parameters so as to reduce the overhead of generating adversarial examples, albeit not remove this overhead entirely. More recently, [272] showed that combining the single-step FGSM with random initialization is almost as effective as PGD-based training, but at a significantly lower cost. Unlike all of the adversarial training strategies, our approach does not require computing adversarial images, and does not depend on a specific attack scheme. Instead, it aims to ensure a maximal separation between the different classes in high attention regions. This significantly differs from [186, 187], which clusters the penultimate layer's global representation, without focusing on discriminative regions and without attempting to separate these features. Furthermore, and more importantly, in contrast to all the above-mentioned methods, our approach is tailored to fine-grained recognition, making use of the representations that have proven effective in this field, such as Bags of Words (BoW) [109, 266] and VLAD [5, 75], which have the advantage over second-order features [73, 127, 292] of providing some degree of interpretability.

Interpretability. Understanding the decisions of a DNN is highly important in real-world applications to build user trust. In the context of general image recognition, the trend of interpreting a DNN's decision was initiated by [297], followed by the popular CAMs [316]. Subsequently, variants of CAMs [28, 232] and other visualization strategies [195] were proposed.

Here, in contrast to these works, we focus on the task of fine-grained recognition. In this domain, BoW-inspired representations, such as the one of [29, 72, 188, 265, 267], were shown to provide some degree of interpretability. While most methods [72, 188, 265, 267] allows one to highlight the image regions important for classification, it does not provide one with visual explanations of the network's decisions. This is addressed by ProtoPNet [29], which extracts class-specific prototypes. However, the feature embedding learnt by ProtoPNet gives equal importance to all image regions, resulting in a large number of prototypes representing non-discriminative background regions, as illustrated by Fig. 4.1. Here, we overcome this by designing an attention-aware system that learns prototypes which are close to high-attention regions in feature space, while constraining the non-discriminative regions from all classes to be close to each other. Furthermore, we show that this brings about not only interpretability, but also robustness to adversarial attacks, which has never been studied in the context of fine-grained recognition.

4.3 Interpreting Adversarial Attacks

Before delving into our method, let us study in more detail the experiment depicted by Fig. 4.1 to understand the decision of a fine-grained recognition CNN under adversarial attack. For this analysis, we experiment with two networks: the second-order attentional pooling network of [74] and the ProtoPNet of [29], both of which inherently encode some notion of interpretability in their architecture, thus not requiring any post-processing. Specifically, [74] uses class attention maps to compute class probabilities, whereas [29] exploits the similarity between image regions and class-specific prototypes. We analyze the reasons for the success of adversarial attacks on four images from the *Black-footed albatross* class.

As shown in Fig 4.1(a), under attack, [74] misclassifies all four images to *Layman albatross.* Note that these two classes belong to the same general *Albatross* family, and, for clean samples, the region with the highest attention for these two classes is the bird's beak. Because the discriminative regions for these two classes correspond to the same beak region, which looks similar in both classes, the attack becomes easier as minimal perturbation is needed to change the class label.

In the case of ProtoPNet [29], while the network also consistently misclassifies the attacked images, the resulting label differs across the different images, as shown in Fig. 4.1 (b). In the top row, the situation is similar to that occurring with the method of [74]. By

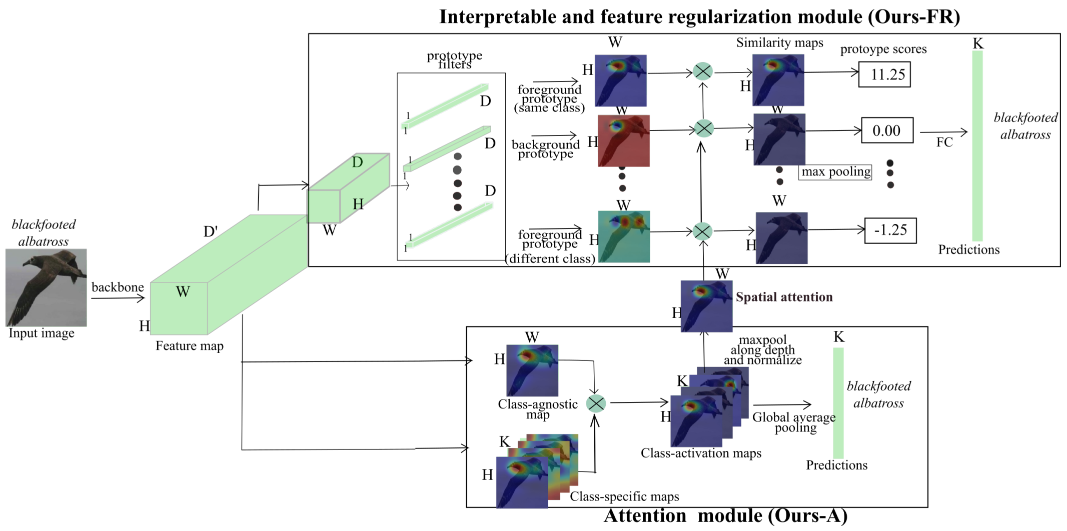


Figure 4.3 – **Overview of our framework**. Our approach consists of two modules acting on the features extracted by a backbone network. The attention module extracts attention maps that help the network to focus on the discriminative image regions. The feature regularization module further uses the attention maps to encourage separating the learned prototypes belonging to different classes.

contrast, in the second row, the region activated in the input image corresponds to a different semantic part (wing) than that activated in the prototype (beak). Finally, in the last two rows, the network activates a background prototype that is common across the other categories and thus more vulnerable to attacks.

In essence, the mistakes observed in Fig 4.1 come from either the discriminative regions of two different classes being two close in feature space, or the use of non-discriminative regions for classification. This motivates us to encourage the feature representation of discriminative regions from different classes to be maximally separated from each other, while minimizing the influence of background regions by making use of attention and by encouraging the features in these regions to lie close to each other so as *not* to be discriminative. This will complicate the attacker's task, by preventing their ability to leverage non-discriminative regions and forcing them to make larger changes in feature space to affect the prediction.

4.4 Method

In this section, we introduce our approach to increasing the robustness of fine-grained recognition by maximal separation of class-specific discriminative regions. Figure 4.3 gives an overview our framework, which consists of two modules post feature extraction: (i) An attention module that learns class-specific filters focusing on the discriminative regions; and (ii) a feature regularization module that maximally separates the class-specific features deemed discriminative by the attention module. Through the feature regularization module, we achieve the dual objective of providing interpretability and increasing the robustness of the backbone network to adversarial attacks.

Note that, at inference time, we can either use the entire framework for prediction, or treat the attention module, together with the backbone feature extractor, as an standalone network. As will be demonstrated by our experiments, both strategies yield robustness to adversarial attacks, which evidences that our approach in fact robustifies the final feature map. Below, we first describe the overall architecture of our framework and then discuss the feature regularization module in more detail.

4.4.1 Architecture

Formally, let $\mathbf{I}_i$ denote an input image, and $\mathbf{X}_i \in \mathbb{R}^{H \times W \times D'}$ represent the corresponding feature map extracted by a fully-convolutional backbone network. Our architecture is inspired by the ProtoPNet of [29], in the sense that we also rely on class-specific prototypes. However, as shown in Section 4.3, ProtoPNet fails to learn discriminative prototypes, because it allows the prototypes to encode non-discriminative background information and to be close in feature space even if they belong to different classes. To address this, we propose to focus on the important regions via an attention mechanism and to regularize the prototypes during training.

Specifically, our attention module consists of two sets of filters: (i) A class-agnostic $1 \times 1 \times D'$ filter yielding a single-channel map of size $H \times W$; and (ii) K class-specific $1 \times 1 \times D'$ filters producing K maps of size $H \times W$ corresponding to the K classes in the dataset. Each of the class-specific map is then multiplied by the class-agnostic one, and the result is spatially averaged to generate a K-dimensional output. As shown in [74], this multiplication of two attention maps is equivalent to a rank-1 approximation of second-order pooling, which has proven to be well-suited to aggregate local features for fine-grained recognition.

The second branch of our network extracts interpretable prototypes and is responsible to increase the robustness of the features extracted by the backbone. To this end, $\mathbf{X}_i$ is first processed by two 1×1 convolutional layers to decrease the channel dimension to D. The resulting representation is then passed through a prototype layer that contains m learnable prototypes of size $1 \times 1 \times D$, resulting in m similarity maps of size $H \times W$. Specifically, the prototype layer computes the residual vector $\mathbf{r}$ between each local feature and each prototype, and passes this distance through an activation function defined as $f(\mathbf{r}) = \log\big((\|\mathbf{r}\|_2^2 + 1)/(\|\mathbf{r}\|_2^2 + \gamma)\big)$, where γ is set to $1e-5$. In contrast to [29], to focus on discriminative regions, we modulate the resulting similarity maps with an attention map $\mathbf{A}_i$, computed by max-pooling the final class-specific maps of the attention module. We then spatially max-pool the resulting attention-aware similarity maps to obtain similarity scores, which are passed through the final classification layer to yield class probabilities. As in [29], we make the prototypes class specific by splitting the m prototypes into K sets of c prototypes and initializing the weights of the classification layer of the prototype branch to +1 for positive connections between prototype and class label and -0.5 for negative ones. While exploiting attention encourages the prototypes to focus on the discriminative regions, nothing explicitly prevents prototypes from different classes to remain close in feature space, thus yielding a small margin between different classes and making the classifier vulnerable to attacks. This is what we address below.

4.4.2 Discriminative Feature Separation

To learn a robust feature representation, we introduce two feature regularization losses that aim to maximally separate the prototypes of different classes. Let $\mathbf{x}_i^t$ represent a local feature vector at location t in feature map $\mathbf{X}_i$ from image $\mathbf{I}_i$ with label y_i. Furthermore, let $N = W \cdot H$ be the total number of feature vectors in $\mathbf{X}_i$, and $\mathbf{P}_{y_i}$ be the set of prototypes belonging to class y_i.

Our regularization consists of two attention-aware losses, a clustering one and a separation one. The attentional-clustering loss pulls the high-attention regions in a sample close to the nearest prototype of its own class. We express this as

$$\mathrm{L}_{clst}^{att}(\mathbf{I}_i) = \sum_{t=1}^{N} a_i^t \min_{l:\mathbf{p}_l \in \mathbf{P}_{y_i}} \|\mathbf{x}_i^t - \mathbf{p}_l\|_2^2 \,, \tag{4.1}$$

where a_i^t is the attention value at location t in $\mathbf{A}_i$. By contrast, the attentional-separation loss pushes the high-attention regions away from the nearest prototype of any other class. We compute it as

$$\mathrm{L}_{sep}^{att}(\mathbf{I}_i) = -\sum_{t=1}^{N} a_i^t \min_{l:\mathbf{p}_l \notin \mathbf{P}_{y_i}} \|\mathbf{x}_i^t - \mathbf{p}_l\|_2^2 \,. \tag{4.2}$$

While these two loss functions encourage the prototypes to focus on high-attention, discriminative regions, they leave the low-attention regions free to be close to any prototype, thus increasing the vulnerability of the network to attacks. We therefore further need to push the non-discriminative regions away from such informative prototypes. A seemingly natural way to achieve this would consist of exploiting inverted attention maps, such as $1 - \mathbf{a}_t$ or $1/\mathbf{a}_t$. However, in practice, we observed this to make training unstable. Instead, we therefore propose to make use of the attention maps from *other* samples to compute the loss for sample i. Specifically, we re-write our regularization loss for sample i as

$$\mathrm{L}_{reg}(\mathbf{I}_i) = \sum_{j=1}^{B}\sum_{t=1}^{N} \lambda_1 a_j^t \min_{l:\mathbf{p}_l \in \mathbf{P}_{y_i}} \|\mathbf{x}_i^t - \mathbf{p}_l\|_2^2 - \lambda_2 a_j^t \min_{l:\mathbf{p}_l \notin \mathbf{P}_{y_i}} \|\mathbf{x}_i^t - \mathbf{p}_l\|_2^2 \,, \tag{4.3}$$

where B is the number of samples in the mini-batch. When $j = i$, we recover the two loss terms defined in Eqs. 4.1 and 4.2. By contrast, when $j \neq i$, we exploit the attention map of a different sample. The intuition behind this is that either the attention map of sample j focuses on the same regions as that of sample i, and thus the loss serves the same purpose as when using the attention of sample i, or it focuses on other regions, and the loss then pushes the corresponding feature map, encoding a low-attention region according to the attention map of sample i, to its own prototype in class y_i. In practice, we have observed this procedure to typically yield a single background prototype per class. These background prototypes inherently become irrelevant for classification because they correspond to low-attention regions and have thus a similarity score close to zero, thanks to our attention-modulated similarity maps. As such, we have empirically found that all background prototypes tend to cluster.

Ultimately, we write our total loss function for sample i as

$$\mathrm{L}(\mathbf{I}_i) = \mathrm{CE}_{att}(\mathbf{I}_i) + \mathrm{CE}_{reg}(\mathbf{I}_i) + \mathrm{L}_{reg}(\mathbf{I}_i) \,,$$

where CE_{att} and CE_{reg} represent the cross-entropy loss of the attention module and the feature regularization module, respectively.

At inference time, we perform adversarial attacks on the joint system by exploiting the cross-entropy loss of both the attention and feature regularization module. Furthermore, we also attack the attention module on its own, showing that, together with the feature extraction backbone, it can be used as a standalone network and also inherits robustness from our training strategy.

4.5 Experiments

4.5.1 Experimental Setting

Datasets. We experiment on two popular fine-grained datasets, Caltech UCSD Birds (CUB) [259] and Stanford Cars-196 [133].

Threat Model. We consider both white-box and black-box attacks under an ℓ_∞-norm budget. We evaluate robustness for two attack tolerances $\epsilon = \{2/255, 8/255\}$. In addition to the popular 10-step PGD attack [168], we test our framework with FGSM [79], BIM [137], and MI [52] attacks. For PGD attacks, we set the step size α to 1/255 for $\epsilon = 2/255$ and to 2/255 for $\epsilon = 8/255$. For the other attacks, we set number of iterations to 10 and the step size α to ϵ divided by the number of iterations, as in [173, 186]. For black-box attacks, we transfer the adversarial examples generated using 10-step PGD with $\epsilon = 8/255$ and $\alpha = 2/255$ on either a similar VGG16 [237] architecture, or a completely different DenseNet-121 [99] architecture. We denote by BB-V and BB-D the black-box attacks transferred from VGG16 and DenseNet-121, respectively.

Networks. We evaluate our approach using 3 backbone networks: VGG-16 [237], VGG-19 [237] and ResNet-34 [89]. Similarly to [29], we perform all experiments on images cropped according to the bounding boxes provided with the dataset, and resize the resulting images to 224×224. For both VGG-16 and VGG-19, we use the convolutional layers until the 4th block to output 7×7 spatial maps of 512 channels. For ResNet-34, we take the network excluding the final global average pooling layer as backbone. We initialize the backbone networks with weights pretrained on ImageNet [46].

Evaluated Methods. As baselines, we use the attentional pooling network (AP) of [74], and the state-of-the-art ProtoPNet of [29]. We use **Ours-FR** and **Ours-A** to denote the output of our feature regularization module and of our attention module, respectively. In other words, AP and Ours-A share the *same* architecture at inference time, and Ours-FR is an attention-aware variant of ProtoPNet. To further boost the performance of the baselines and of our approach, we perform adversarial training. Specifically, we generate adversarial examples using the recent fast adversarial training strategy of [272], which relies on a single step FGSM with random initialization. During training, we set ϵ to 8/255 and α to 1.25ϵ as suggested in [272]. This was shown in [272] to perform on par with PGD-based adversarial training, while being computationally much less expensive. For our approach, during fast adversarial training, we use the cross-entropy loss of both modules to generate the adversarial images. We denote by AP* and ProtoPNet* the adversarially-trained AP and ProtoPNet baselines, respectively, and by **Ours-FR*** and **Ours-A*** the adversarially-trained counterparts of our two sub-networks. We also compare with state-of-the-art defense [186] which regularizes the hidden space with additional prototype conformity loss (PCL).

4.5.2 Results on CUB 200

Quantitative Analysis. We first compare the accuracy of our method to that of the baselines with the three backbone networks on CUB200. Table 4.1 and Table 4.2 provide the results for vanilla and fast adversarial training, respectively. On the clean samples, **Ours-FR** typically surpasses its non-attentional counterpart **ProtoPNet** [29], and **Ours-A** yields the better accuracy than baseline AP and AP+PCL across all backbones. This is true both without (Table 4.1) and with (Table 4.2) adversarial training.

Base Network	Attacks (Steps,ϵ)	Clean (0,0)	FGSM (1,2)	FGSM (1,8)	BIM (10,2)	BIM (10,8)	PGD (10,2)	PGD (10,8)	MIM (10,2)	MIM (10,8)	BB-V (10,2)	BB-D (10,8)
VGG-16	AP [74]	78.0%	36.5%	31.0%	27.7%	14.6%	23.5%	11.7%	30.2%	16.7%	9.6%	60.4%
	AP+ PCL [186]	80.0%	41.0%	33.1%	32.9%	13.6%	23.5%	9.6%	35.3%	17.1%	10.6%	65.8%
	Ours-A	80.4%	47.2%	**40.2%**	**40.0%**	**23.2%**	**35.3%**	**21.8%**	**42.2%**	**26.4%**	**12.9%**	**66.9%**
	ProtoPNet [29]	69.0%	19.9%	8.10%	3.80%	0.00%	2.20%	0.00%	5.00%	0.10%	**22.9%**	58.5%
	ProtoPNet+Ours	**73.2%**	**45.6%**	**40.7%**	**40.4%**	**31.7%**	**37.3%**	**27.3%**	**40.7%**	**34.2%**	15.4%	**59.7%**
VGG-19	AP [74]	75.7%	20.4%	14.5%	13.4%	6.9%	10.5%	5.7%	14.8%	6.9%	21.1%	61.3%
	AP+ PCL [186]	76.9%	20.3%	14.8%	12.1%	5.7%	8.8%	4.2%	13.9%	6.8%	19.8%	60.2%
	Ours-A	79.7%	51.4%	**44.6%**	**42.3%**	**26.5%**	**36.8%**	**26.3%**	**45.0%**	**29.9%**	**29.8%**	**68.2%**
	ProtoPNet [29]	73.8%	22.9%	11.1%	3.2%	0.0%	1.2%	0.0%	3.6%	0.0%	21.0%	58.0%
	Ours-FR	75.4%	**52.2%**	**46.3%**	**46.6%**	**41.3%**	**42.4%**	**31.0%**	**44.4%**	**37.6%**	**30.4%**	**63.7%**
ResNet-34	AP [74]	**79.9%**	30.4%	26.3%	18.0%	7.20%	13.2%	5.8%	22.3%	8.6%	43.0%	59.4%
	AP+ PCL [186]	77.9%	30.1%	24.5%	21.4%	13.3%	17.6%	11.6%	23.9%	15.3%	45.7%	61.4%
	Ours-A	79.0%	**32.3%**	**27.0%**	**24.8%**	**20.5%**	**22.5%**	**19.8%**	**26.2%**	**22.0%**	**48.6%**	**63.2%**
	ProtoPNet [29]	75.1%	23.2%	12.8%	7.80%	1.80%	4.10%	1.00%	8.90%	2.20%	39.1%	53.0%
	Ours-FR	**76.3%**	**30.7%**	**22.0%**	**19.3%**	**13.6%**	**14.2%**	**13.0%**	**19.1%**	**13.8%**	**46.0%**	**60.0%**

Table 4.1 – Classification accuracy of different networks with ℓ_∞ based attacks on CUB200. The best result of each column and each backbone is shown in bold. The last two columns correspond to black-box attacks with PGD attack.

Under adversarial attack, our approach, without and with adversarial training, yields better robustness under almost all attacks and backbones. Importantly, the boost in performance is larger for attacks with larger perturbations. Furthermore, our model trained with clean samples sometimes outperform even the adversarially-trained baselines. For example, on VGG-16 with PGD attack with $\epsilon = 8/255$, **AP*** yields an accuracy of 16.9% (Table 4.2) while **Ours-A** reaches 21.8% accuracy (Table 4.1). This evidences the ability of our feature regularization module to learn robust features, even without seeing any adversarial examples. This is further supported by the fact that, despite **AP** and **Ours-A** having the same architecture at inference, **Ours-A** is more robust to attacks. Our method outperforms AP+PCL in most cases since PCL do not take into account subtle difference in local regions and regularizes global representation only. Note also that the gap in clean accuracy between standard and adversarial training is smaller for our approach than for the baselines.

Transferability Analysis. To evaluate robustness to black-box attacks, we transfer adversarial examples generated from substitute networks to our framework and to the baselines. As substitute models, we use a VGG-16 [237] and DenseNet-121 [99] backbone followed by global average pooling and a classification layer. The corresponding results are reported in the last two columns of Table 4.1 and Table 4.2. As in the white-box case, our approach outperforms the baselines in this black-box setting, thus confirming its effectiveness at learning robust features.

Decision-based Attacks. As suggested in [9], white-box attacks can give a false sense of security due to gradient obfuscation. Therefore, to understand the superiority of

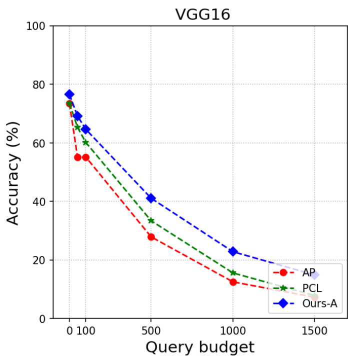
Base Network	Attacks (Steps,ϵ)	Clean (0,0)	FGSM (1,2)	FGSM (1,8)	BIM (10,2)	BIM (10,8)	PGD (10,2)	PGD (10,8)	MIM (10,2)	MIM (10,8)	BB-V (10,2)	BB-D (10,8)
VGG-16	AP* [74]	54.9%	44.9%	24.2%	41.9%	18.2%	41.2%	16.9%	41.9%	18.7%	54.6%	54.0%
	AP+PCL* [186]	60.7%	50.5%	28.5%	47.1%	22.8%	46.7%	21.6%	47.2%	23.5%	59.5%	59.9%
	Ours-A*	**69.3%**	**56.1%**	**34.8%**	**51.7%**	**29.6%**	**50.8%**	**28.0%**	**52.0%**	**32.5%**	**66.3%**	**68.0%**
	ProtoPNet* [29]	60.1%	44.5%	26.9%	**57.1%**	10.9%	35.9%	10.3%	37.6%	13.5%	58.4%	59.1%
	Ours-FR*	**63.0%**	**53.3%**	**37.3%**	49.4%	**30.4%**	**48.1%**	**28.6%**	**49.7%**	**31.1%**	**61.1%**	**62.0%**
VGG-19	AP* [74]	58.0%	47.5%	29.1%	44.3%	25.6%	44.0%	24.34%	44.4%	26.2%	57.0%	57.3%
	AP+PCL* [186]	61.8%	52.1%	30.9%	48.9%	24.7%	48.6%	23.3%	49.1%	25.4%	60.5%	60.9%
	Ours-A*	**68.2%**	**57.1%**	**36.5%**	**53.2%**	**30.4%**	**52.6%**	**29.2%**	**53.5%**	**31.2%**	**66.2%**	**66.9%**
	ProtoPNet* [29]	55.1%	40.0%	28.9%	26.5%	11.3%	29.7%	9.60%	25.6%	10.2%	53.6%	53.9%
	Ours-FR*	**64.4%**	**55.5%**	**37.4%**	**51.2%**	**30.6%**	**50.4%**	**28.7%**	**52.1%**	**32.3%**	**62.5%**	**63.2%**
ResNet-34	AP* [74]	55.6%	47.8%	29.2%	44.80%	21.0%	44.5%	19.4%	44.9%	21.9%	55.3%	55.2%
	AP+PCL* [186]	54.5%	45.4%	26.9%	42.3%	18.2%	41.9%	16.4%	42.4%	19.1%	54.0%	54.0%
	Ours-A*	**61.9%**	**53.6%**	**35.4%**	**50.9%**	**25.8%**	**50.4%**	**23.1%**	**51.1%**	**27.3%**	**61.0%**	**61.3%**
	ProtoPNet* [29]	**57.9%**	46.5%	30.3%	41.1%	21.1%	40.3%	18.4%	41.5%	20.9%	56.9%	57.0%
	Ours-FR*	58.9%	**50.7%**	**32.4%**	**47.4%**	**22.8%**	**46.8%**	**20.1%**	**47.6%**	**27.2%**	**58.1%**	58.2%

Table 4.2 – Classification accuracy of different robust networks with ℓ_∞ based attacks on CUB200. The best result of each column and each backbone is shown in bold. The last two columns correspond to black-box attacks with PGD attack.

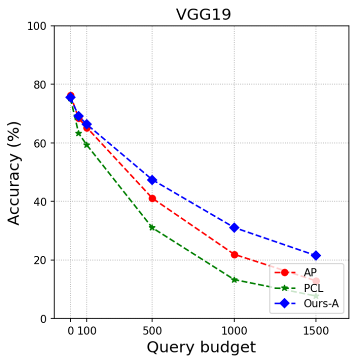
our framework, we conduct a decision-based Square attack [4] with $\epsilon = 8$ for a query budget up to 1500. Due to the high computational complexity of the attack, we run the experiments on a subset of test data of 1400 images. As observed from Figures 4.4 and 4.5, Our training framework achieves higher robust accuracy on all cases. The differences w.r.t to baselines is more pronounced for higher query budgets.

Auto-attack. Furthermore, we evaluate our defense on the latest auto attack [43] framework which consists of four diverse parameter-free gradient-based and decision-based attacks. We could not run the FAB [42] attack proposed in the ensemble due to prohibitive computational times. We therefore evaluate on the rest of the three attacks namely, APGD (CE), APGD(DLR) and Square attack with $\epsilon = 8$. As shown from the Table 4.3, our framework performs better than the baselines for all the three backbones for the rest of the three attacks.

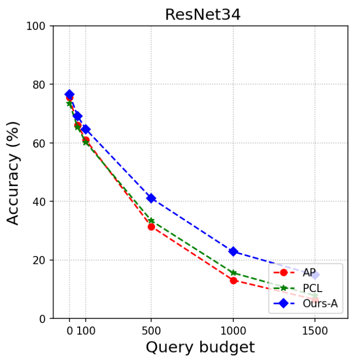
Qualitative Analysis. Let us now qualitatively evidence the benefits of our approach. To this end, in Figure 4.6, we visualize the 10 class-specific prototypes learned by ProtPNet and by our approach for the *Blackfooted albatross* class. Specifically, we show the activation heatmaps of these prototypes on the source image that they have been projected to. Note that ProtoPNet learns multiple background prototypes, whereas our approach encodes all the background information in a *single non-discriminative* prototype. Furthermore, ProtoPNet [29] focuses on much larger regions, which can be expected to be less discriminative than the fine-grained regions obtained using our approach. This is due to our use of attention, which helps the prototypes to focus on the



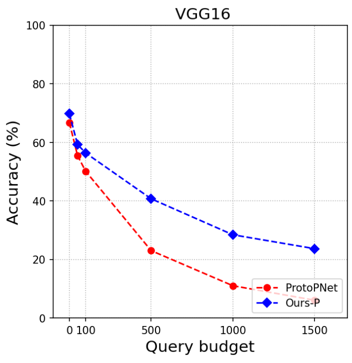
(a) VGG16



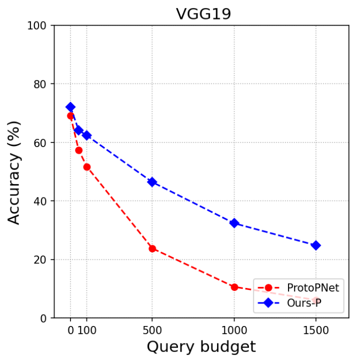
(b) VGG19



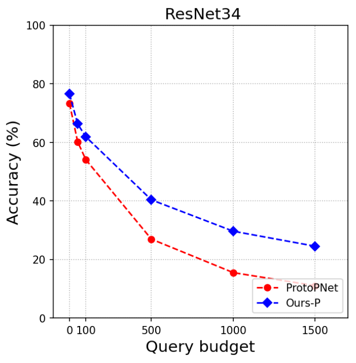
(c) ResNet34



(d) VGG16



(e) VGG19



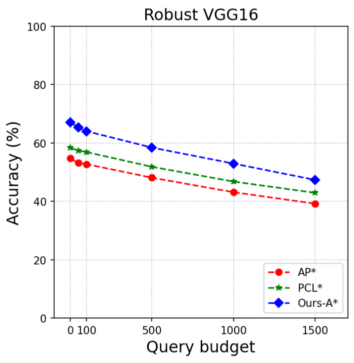
(f) ResNet34

Figure 4.4 – **ℓ_∞-Square attacks on undefended networks.** Classification accuracy of different undefended networks with $\epsilon = 8$ by varying the query budget on 1400 samples of CUB200.

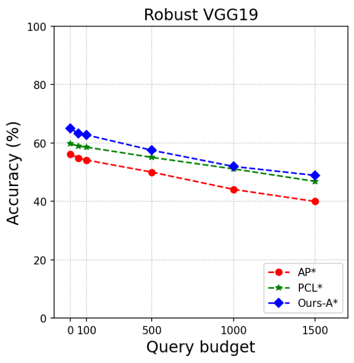
areas that are important for classification.

In Figure 4.7, we analyze the effect of adversarial attacks on AP, ProtoPNet and our approach (all without adversarial training) by visualizing the attention maps and/or a few top activated prototypes along with their similarity scores for a *Blackfooted albatross* image with and without attack. Without attack, AP activates a larger region than our attention module. Furthermore, ProtoPNet activates a prototype from a different class (*Cape glossay starling*), while our approach focuses on the correct class only. This already shows that the features learned by these baselines are less discriminative, making them more vulnerable to adversarial attacks. As a matter of fact, under attack, AP focuses on a different region that is not discriminative for the *Blackfooted albatross* class. Similarly, ProtoPNet activates prototypes of different classes with high similarity scores, highlighting non-discriminative regions. By contrast, the prototypes activated by our approach remain the same as in the clean case, thus corresponding to the correct class.

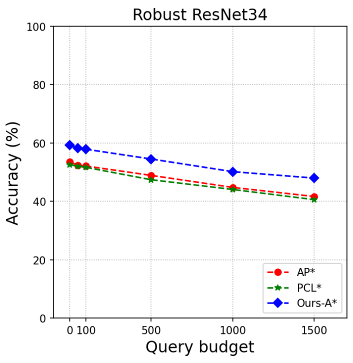
Gradient Obfuscation. As suggested in [9], we check the gradient obfuscation to



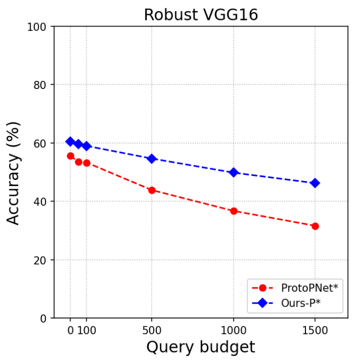
(a) VGG16



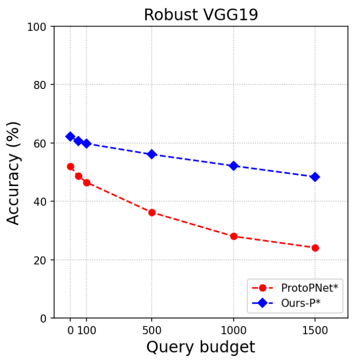
(b) VGG19



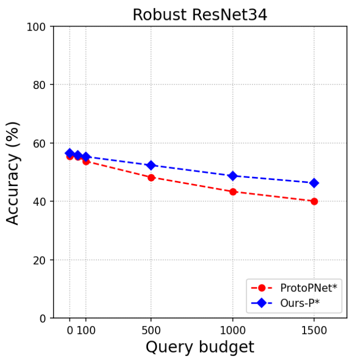
(c) ResNet34



(d) VGG16



(e) VGG19



(f) ResNet34

Figure 4.5 – **ℓ_∞-Square attacks on robust networks.** Classification accuracy of different robust networks with $\epsilon = 8$ by varying the query budget on 1400 samples of CUB200.

ensure that proposed approach do not give false sense of security. As shown in Figure 4.8, VGG-16 trained with our feature regularization performance drops as the perturbation norm increases. Further, from Table 4.1 and 4.5, the black-box attacks are less successful than white box attacks. Both these experiments suggest our formulation do not suffer from gradient obfuscation, as was also evidenced from the results of autoattack.

Adversarial sample detection. Our formulation also helps in detecting adversarial samples due to well separation of discriminative regions. Following [142], we learn a logistic detector by computing mahalanobis distance to the nearest class-conditional Gaussian distribution as the feature at every layer of the network. As shown in Figure 4.9, our proposed feature regularization approach increases the detection AUROC performance over the baselines by around 20%.

Ablation Study. To understand the importance of each module in achieving robustness, we perform ablation study on VGG-16. As shown from Table 4.4, our attention-aware formulation performs better than baseline even without feature regularization. However

Base	Attacks	Clean	APGD (CE)	APGD (DLR)	Square	Auto attack
VGG-16	AP* [74]	54.9%	15.1%	14.0%	39.2%	22.7%
	AP+PCL* [186]	60.7%	18.0%	14.1%	42.9%	25.0%
	Ours-A*	**67.0%**	**23.7%**	**15.1%**	**47.3%**	**28.7%**
	ProtoPNet* [29]	55.6%	2.8%	2.3 %	31.6%	12.2%
	Ours-FR*	**60.4%**	**24.2%**	**15.5%**	**46.2%**	**28.6%**
VGG-19	AP* [74]	55.7%	20.2%	14.4%	44.1%	26.2%
	AP+PCL* [186]	59.7%	20.8%	17.3%	51.1%	29.7 %
	Ours-A*	**65.0%**	**24.4%**	**17.4%**	**51.9%**	**31.2%**
	ProtoPNet* [29]	51.9%	1.1 %	1.0 %	28.0%	10.0%
	Ours-FR*	**62.1%**	**27.4%**	**18.5%**	**52.1%**	**32.7%**
ResNet-34	AP* [74]	53.6%	18.8%	20.1%	44.7%	27.9 %
	AP+PCL* [186]	52.6%	15.9%	17.2%	44.0%	25.7 %
	Ours-A*	**59.2%**	**20.7%**	**23.7%**	**50.1%**	**31.5%**
	ProtoPNet* [29]	52.6%	15.9%	17.2%	44.0%	25.7%
	Ours-FR*	**56.4%**	**18.0%**	**20.0%**	**48.7%**	**28.9%**

Table 4.3 – Classification accuracy of different robust networks with ℓ_∞ based state-of-the-art Auto attack [43] ensemble on CUB200 with $\epsilon = 8$. The best result of each column and each backbone is shown in bold.

by adding attentional cluster and separation cost, we achieve significant improvements over the baselines.

4.5.3 Results on Stanford Cars

We now present on results on Stanford Cars [133]. In Table 4.5, we report the results obtained using vanilla training. As in the CUB case, our approach yields better robustness than the baselines.

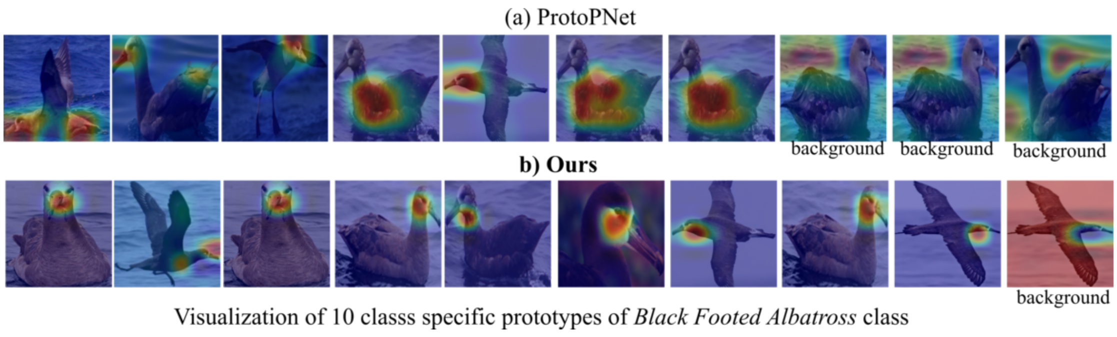

(a) ProtoPNet
background
background
background
b) Ours
background
Visualization of 10 classs specific prototypes of *Black Footed Albatross* class

Figure 4.6 – **Comparison of the prototypes learned with ProtoPNet [29] and with our approach on CUB**. ProtoPNet yields multiple background prototypes and prototypes that focus on large regions. By contrast, our prototypes are finer-grained and thus more representative of the specific class in the images.

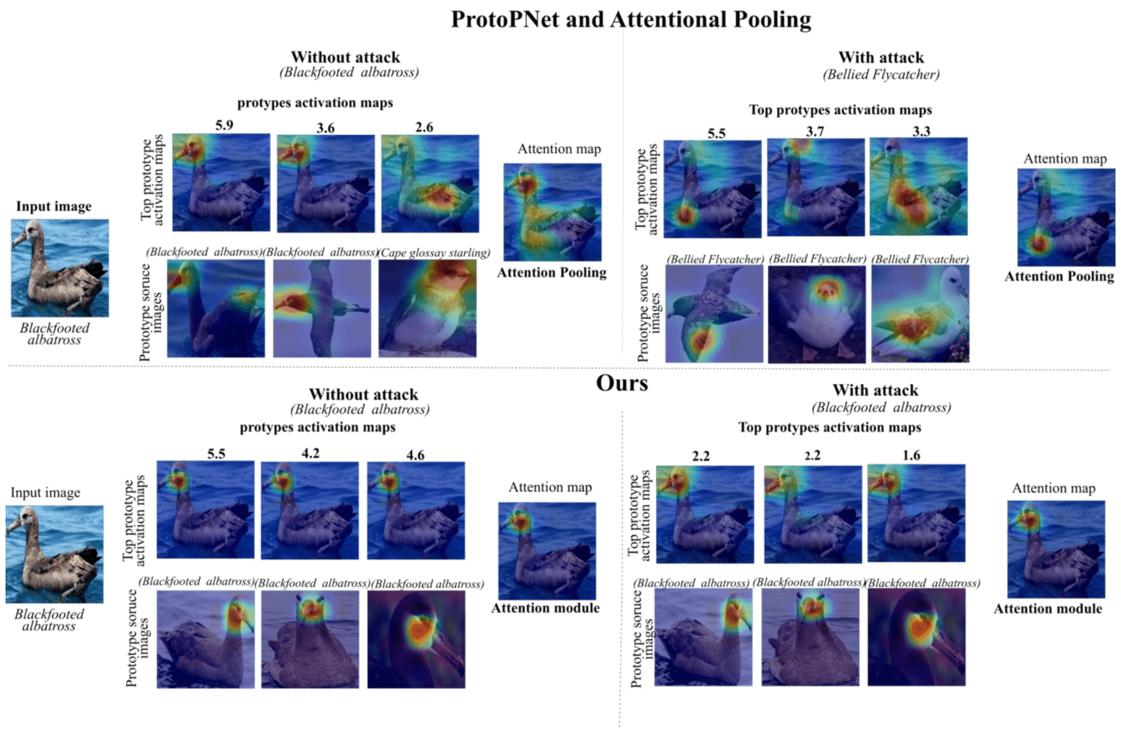


Figure 4.7 – **Comparison of the activated image regions without and with attack.** Without attack, the baselines (AP and ProtoPNet) tend to rely on relatively large regions, sometimes corresponding to wrong classes, for prediction. By contrast, our approach focuses more closely on the discriminative regions. Under PGD attack, this phenomenon is further increased, with ProtoPNet and AP activating incorrect prototypes and regions. The activations obtained with our approach remain similar to those obtained without attacks, albeit with a decrease in the similarity scores, indicated above the top prototype activation maps.

In Figure 4.10, we compare the prototypes learned with ProtoPNet and with our approach for the *Accura TL Sedan* class. As before, while the prototypes learned by ProtoPNet cover large regions, those obtained with our framework are more focused on the discriminative parts of the car.

In Table 4.6, we report the robustness of fast adversarial training [272] with our discriminative feature separation approach. Our approach, **Ours-FR***, performs better than the baseline ProtoPNet* [29] in all cases. Note that, for multi-step iterative attacks, **Ours-A*** performs better than AP*, while they achieve comparable performance for single-step attacks.

4.5.4 Training Details

We implemented our approach using the PyTorch library, and ran our experiments on a single 32GB Tesla GPU. We set the mini-batch size B to 75 during training. Similarly

Network	Att-clustering loss	Att-separation loss	Clean (0,0)	PGD (10,8)
AP [74]	-	-	78.0%	11.7%
Ours-A	-	-	78.7%	14.07%
	-	✓	79.6%	0.0%
	✓	-	80.0%	19.3%
	✓	✓	80.4%	21.8%

Network	Att-clustering loss	Att-separation loss	Clean (0,0)	PGD (10,8)
ProtoPNet [29]	-	-	69.0%	0.0%
Ours-FR	-	-	75.7%	13.76%
	-	✓	69.8%	0.0%
	✓	-	73.7%	18.7%
	✓	✓	73.2%	30.1%

Table 4.4 – **Ablation study**. Contribution of each proposed feature regularization module in classification accuracy of undefended VGG-16 network.

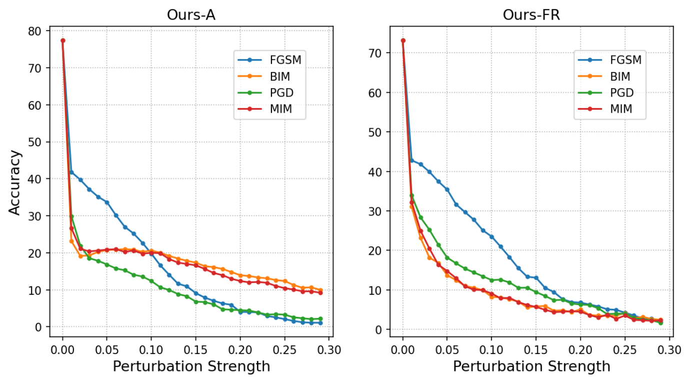


Figure 4.8 – Performance of VGG-16 with our proposed approach under different perturbation strengths.

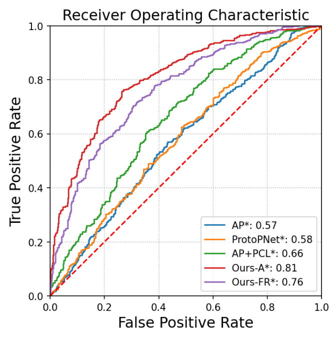


Figure 4.9 – ROC curves for adversarial sample detection on robust VGG-16 with PGD attack.

to [29], we initialize the last layer of the prototype branch with +1 for positive and -0.5 for negative connection between prototype and class label. We set $c = \{5, 10\}$ prototypes per class, λ_1 to $\{10, 100\}$ and $\lambda_2 = 0.08$ depending on the dataset and architecture. We first fine-tune the attention and feature regularization modules, except for the classification layer of the latter, for 5 epochs with a learning rate of 0.0003, keeping the backbone network fixed. We then jointly train all the layers, except the feature regularization classifier, to minimize the objective of Eq. 3 for 25 epochs, with an initial learning rate of 0.003 and a decay rate of 0.1 applied every 10 epochs. After 30 epochs, we project the prototypes to the nearest training image patch of the same class and optimize the classification layer of the feature regularization module for 15 epochs. Note that for AP* on CUB200 with VGG19 backbone, we set slightly larger $\alpha = 1.30\epsilon$ instead of 1.25ϵ as we found the training did not converge possibly due to phenemenon of catastropic forgetting.

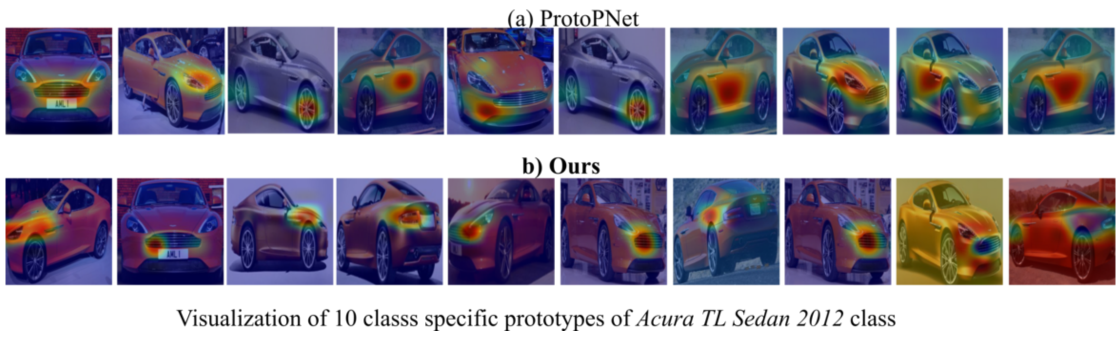

(a) ProtoPNet
b) Ours
Visualization of 10 classs specific prototypes of *Acura TL Sedan 2012* class

Figure 4.10 – **Comparison of the prototypes learned with ProtoPNet [29] and with our approach on Stanford-Cars**. ProtoPNet yields prototypes that cover large regions, whereas our prototypes more focused.

Base Nettwork	Attacks (Steps,ϵ)	Clean (0,0)	FGSM (1,2)	FGSM (1,8)	BIM (10,2)	BIM (10,8)	PGD (10,2)	PGD (10,8)	MIM (10,2)	MIM (10,8)	BB-V (10,2)	BB-D (10,8)
VGG-16	AP [74]	**91.2%**	52.6%	40.2%	37.4%	10.5%	28.8%	6.93%	41.7%	12.9%	12.5%	82.5%
	AP+Triplet [231]	91.1%	54.3%	**43.5%**	42.4%	14.9%	34.1%	9.54%	45.5%	19.2%	15.6%	**84.7%**
	AP+PCL [186]	90.2%	51.7%	40.5%	39.3%	14.1%	31.8%	9.44%	42.5%	17.5%	16.7%	83.9%
	Ours-A	88.5%	**58.7%**	40.2%	**48.0%**	**28.6%**	**46.5%**	**21.7%**	**53.2%**	**33.2%**	**19.9%**	82.2%
	ProtoPNet [29]	**84.5%**	31.2%	9.85%	4.78%	0.01%	2.23%	0.00%	6.5%	0.01%	**27.8%**	**75.5%**
	Ours-FR	83.8%	**60.1%**	**52.0%**	**51.3%**	**41.0%**	**47.8%**	**32.9%**	**51.8%**	**43.9%**	23.4%	75.1%
VGG-19	AP [74]	**91.5%**	50.1%	37.8%	33.4%	10.3%	23.83%	6.93%	37.9%	12.7%	20.7%	82.8%
	AP+Triplet [231]	91.0%	56.2%	45.1%	40.5%	13.0%	30.3%	8.70%	45.3%	16.7%	29.0%	85.0%
	AP+PCL [186]	91.3%	61.3%	49.9%	49.0%	19.7%	40.2%	14.1%	52.4%	23.4%	30.6%	**85.7%**
	Ours-A	88.7%	**64.4%**	**54.8%**	**56.4%**	**36.7%**	**51.7%**	**33.4%**	**58.1%**	**41.0%**	**35.9%**	82.5%
	ProtoPNet [29]	**85.6%**	34.1%	20.8%	11.3%	1.11%	4.40%	0.5%	14.2%	1.39%	26.5%	75.5%
	Ours-FR	85.0%	**62.4%**	**54.7%**	**54.5%**	**45.7%**	**51.2%**	**38.5%**	**54.3%**	**47.6%**	**36.1%**	**76.8%**

Table 4.5 – Classification accuracy of different undefended networks with ℓ_∞ based attacks on Cars196.

We use Adam [124] with the default momentum values for all our experiments. For the adversarial detection experiments, we initially remove the misclassified samples from the test set. We then consider successfully attacked from this subset and train a logistic detector with 80% of data and report results on remaining data.

4.5.5 Qualitative Results

In this section, we provide additional qualitative results on CUB200 and Standford Cars.

Visualization of the learned prototypes. In Figure 4.11, we show the activation heat maps of the prototypes on the source images to which they were projected for our VGG-16 model. Our method yields fine-grained prototypes that either focus on a small discriminative region or activate the complete non-discriminative region.

Base Network	Attacks Steps,ϵ)	Clean	FGSM (1,2)	FGSM (1,8)	BIM (10,2)	BIM (10,8)	PGD (10,2)	PGD (10,8)	MIM (10,2)	MIM (10,8)	BB-V (10,8)	BB-D (10,8)
VGG-16	AP* [74]	86.2%	**81.1%**	**63.6%**	**78.9%**	53.8%	**78.7%**	50.8%	**78.7%**	55.1%	85.1%	85.9%
	AP+PCL* [186]	**87.4%**	80.5%	59.4%	77.6%	48.5%	77.2%	44.9%	77.9%	50.2%	**86.0%**	**87.1%**
	Ours-A*	84.8%	79.8%	63.3%	77.0%	**54.6%**	76.6%	**51.1%**	77.1%	**55.8%**	84.5%	85.6%
	ProtoPNet* [29]	64.4%	53.7%	31.9%	48.9%	16.5%	48.2%	13.4%	49.2%	18.2%	63.8%	64.2%
	Ours-FR*	**83.7%**	**76.37%**	**62.8%**	**73.5%**	**55.0%**	**72.6%**	**51.9%**	73.8%	**55.4%**	**80.8%**	**82.0%**

Table 4.6 – Classification accuracy of different robust networks with ℓ_∞ based attacks on Cars196. The best result of each column and each backbone is shown in **bold**. The last two columns correspond to black-box attacks.

Nearest samples of the learned prototypes. In Figure 4.12, we show the prototypes and their nearest test images for CUB 200 with VGG-16. In most cases, the discriminative prototypes activate the same semantic part in all images corresponding to the same class.

Visualization of the learned prototypes. In Figure 4.13, we show the activation heat maps of the prototypes on the source images to which they were projected for our VGG-16 model. Our method yields fine-grained prototypes that either focus on a small discriminative region or activate the complete non-discriminative region.

Nearest samples of the learned prototypes. In Figure 4.14, we show the prototypes and their nearest test images for Cars 196 with VGG-16. In most cases, the discriminative prototypes activate the same semantic part in all images corresponding to the same class.

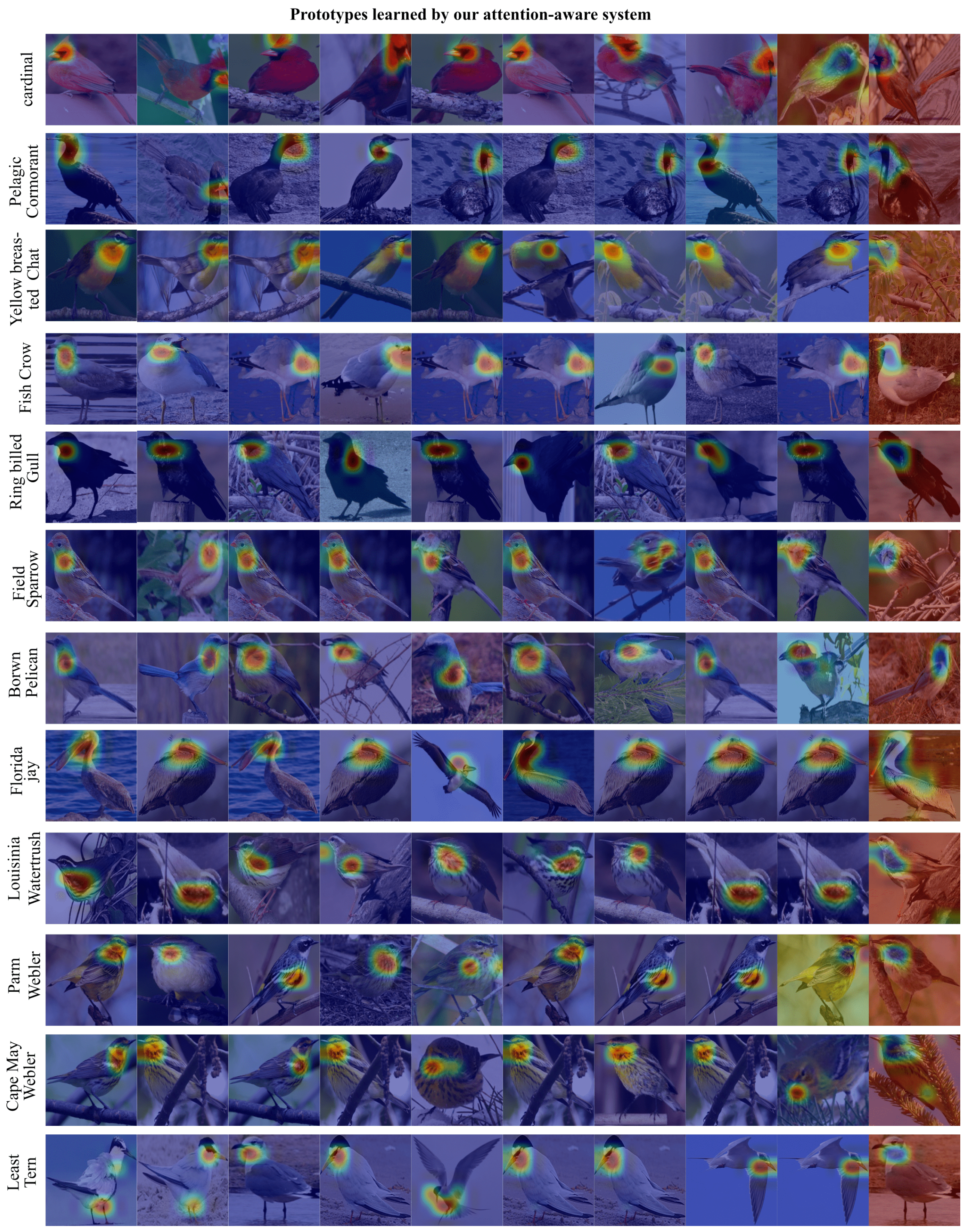


Figure 4.11 – **Visualization of the prototypes learned with our approach on CUB.** Our formulation yields prototypes that are fine-grained and representative of the specific class in the images.

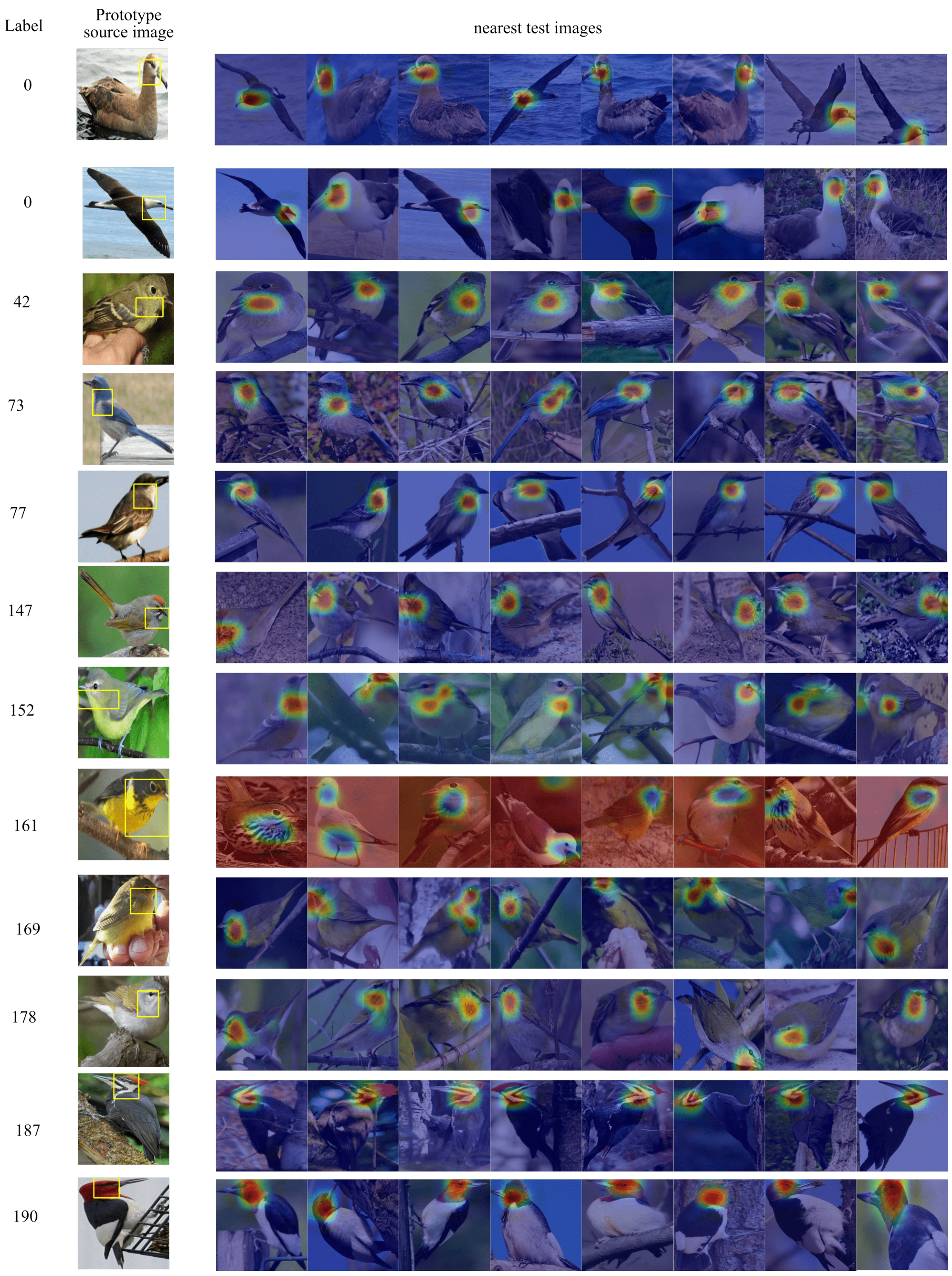


Figure 4.12 – **Visualization of the nearest *test* samples for each learned prototypes with our approach on CUB with VGG-16.** All prototypes activate semantically meaningful parts and mostly from the images corresponding to their own label.

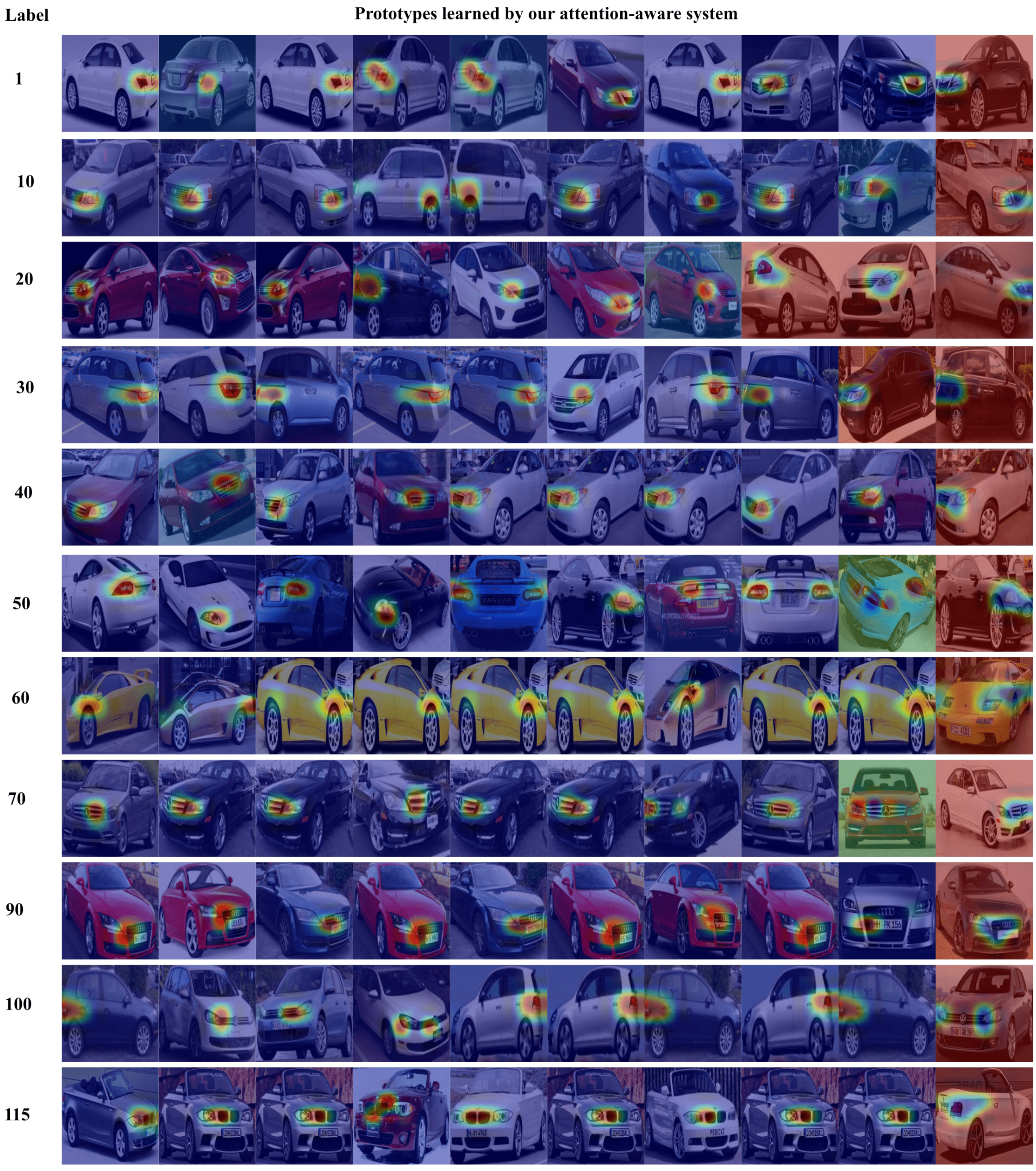


Figure 4.13 – **Visualization of the prototypes learned with our approach on Cars-196.** Our formulation yields prototypes that are fine-grained and representative of the specific class in the images.

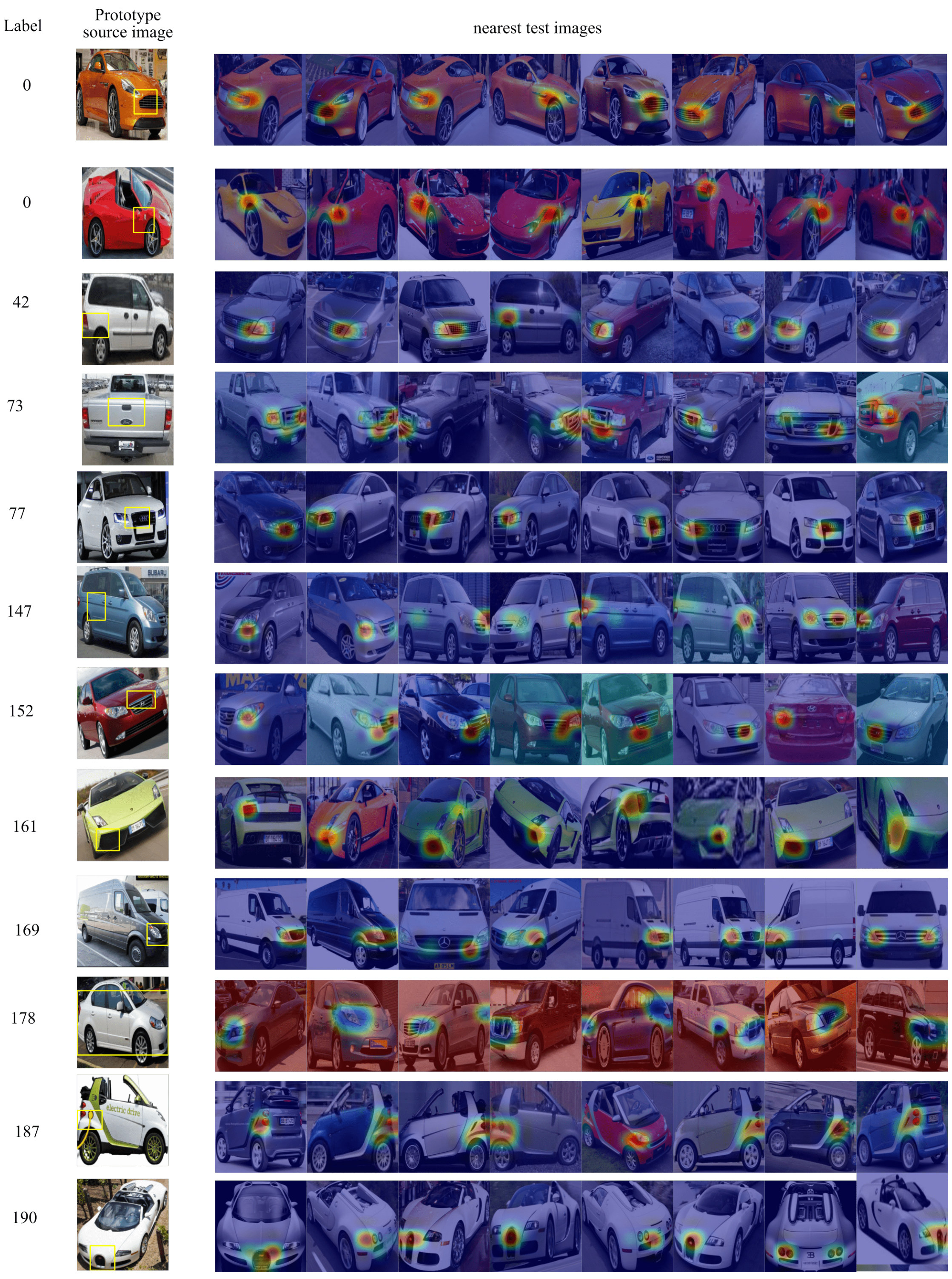


Figure 4.14 – **Visualization of the nearest *test* samples for each learned prototypes with our approach on Cars 196 with VGG-16.** All prototypes activate semantically meaningful parts and mostly from the images corresponding to their own label.

4.6 Conclusion

In this chapter, we have performed the first study of adversarial attacks for fine-grained recognition. Our analysis has highlighted the key factor for the success of adversarial attacks in this context. This has inspired us to design an attention- and prototype-based framework that explicitly encourages the prototypes to focus on the discriminative image regions. Our experiments have evidenced the benefits of our approach, despite not requiring seeing adversarial examples during training. We further observed improving AUROC score of adversarial sample detection on the adversarially trained models. In the following two chapters, we go beyond image classification and design attacks to shed light on the internal workings of DNNs on other tasks.

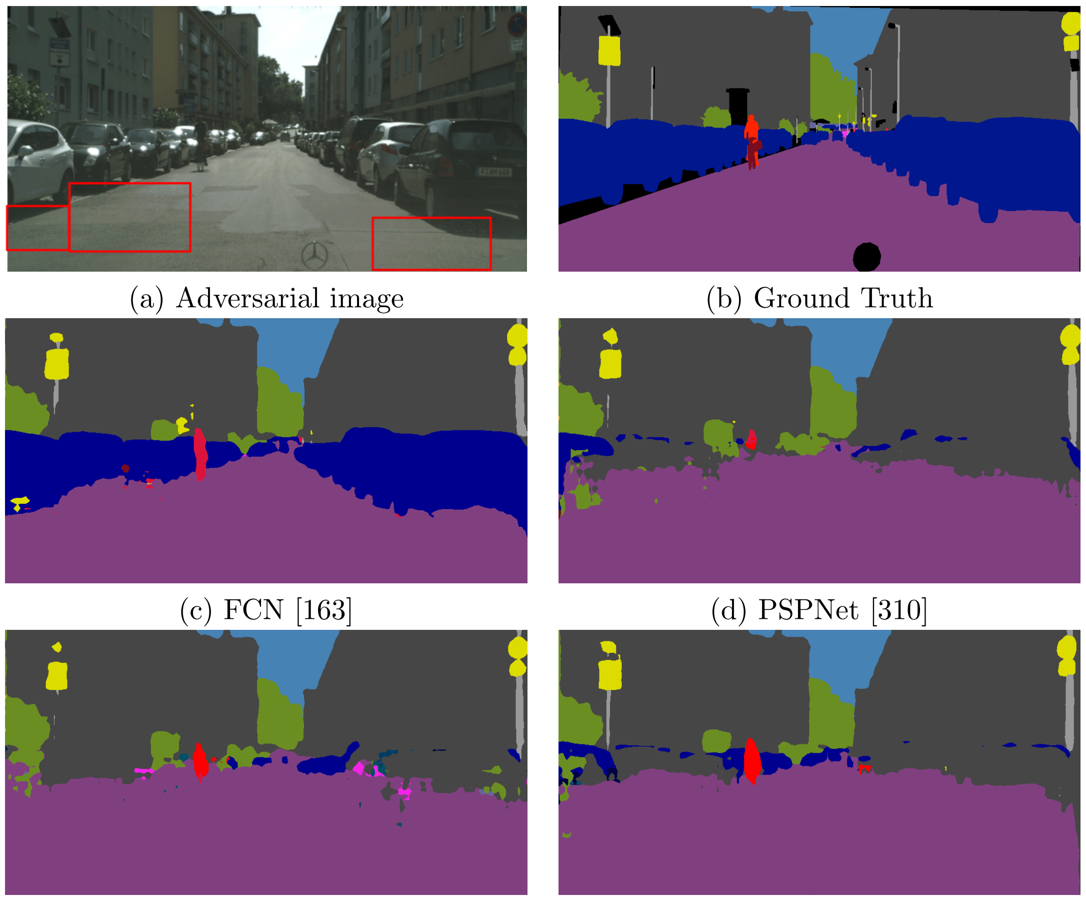
5 Indirect Local Attacks for Context-aware Semantic Segmentation Networks

In the previous chapters, we studied the adversarial attacks limited to image recognition. From this chapter onwards, we turn our focus on designing adversarial attacks on other tasks to understand DNNs' limitations. Recently, deep networks have achieved impressive semantic segmentation performance, in particular thanks to their use of larger contextual information. In this chapter, we show that the resulting networks are sensitive not only to global adversarial attacks, where perturbations affect the entire input image, but also to indirect local attacks, where the perturbations are confined to a small image region that does not overlap with the area that the attacker aims to fool. To this end, we introduce an indirect attack strategy, namely adaptive local attacks, aiming to find the best image location to perturb, while preserving the labels at this location and producing a realistic-looking segmentation map. Furthermore, we propose attack detection techniques both at the global image level and to obtain a pixel-wise localization of the fooled regions. Our results are unsettling: Because they exploit a larger context, more accurate semantic segmentation networks are more sensitive to indirect local attacks. We believe that our comprehensive analysis will motivate the community to design architectures with contextual dependencies that do not trade off robustness for accuracy.

5.1 Introduction

Deep Neural Networks (DNNs) are highly expressive models and achieve state-of-the-art performance in many computer vision tasks. In particular, the powerful backbones originally developed for image recognition have now be recycled for semantic segmentation, via the development of fully convolutional networks (FCNs) [163]. The success of these initial FCNs, however, was impeded by their limited understanding of surrounding context. As such, recent techniques have focused on exploiting contextual information via dilated convolutions [291], pooling operations [159, 310], or attention mechanisms [71, 311].

Despite this success, recent studies have shown that DNNs are vulnerable to adversarial attacks. That is, small, dedicated perturbations to the input images can make a network produce virtually arbitrarily incorrect predictions. While this has been mostly studied in the context of image recognition [52, 138, 182, 196, 201], a few recent works have nonetheless discussed such adversarial attacks for semantic segmentation [8, 92, 284]. These methods, however, remain limited to global perturbations to the entire image. Here, we argue that local attacks are more realistic, in that, in practice, they would allow an attacker to modify the physical environment to fool a network. This, in some sense, was the task addressed in [62], where stickers were placed on traffic poles so that an image recognition network would misclassify the corresponding traffic signs. In this scenario, however, the attack was directly performed on the targeted object.



(a) Adversarial image (b) Ground Truth

(c) FCN [163] (d) PSPNet [310]

(e) PSANet [311] (f) DANet [71]

Figure 5.1 – **Indirect Local Attacks.** An adversarial input image **(a)** is attacked with an imperceptible noise in local regions, shown as red boxes, to fool the dynamic objects. Such *indirect* local attacks barely affect an FCN [163] **(c)**. By contrast, modern networks that leverage context to achieve higher accuracy on clean (unattacked) images, such as PSPNet [310] **(d)**, PSANet [311] **(e)** and DANet [71] **(f)** are more strongly affected, even in regions far away from the perturbed area.

In this chapter, by contrast, we study the impact of *indirect local* attacks, where the perturbations are performed on regions outside the targeted objects. This, for instance, would allow one to place a sticker on a building to fool a self-driving system such that all nearby dynamic objects, such as cars and pedestrians, become mislabeled as the nearest background class. We choose this setting not only because it allows the attacker to perturb only a small region in the scene, but also because it will result in realistic-looking segmentation maps. By contrast, untargeted attacks would yield unnatural outputs, which can much more easily be detected by a defense mechanism. However, designing such targeted attacks that are effective is much more challenging than untargeted ones.

To achieve this, we first investigate the general idea of *indirect* attacks, where the perturbations can occur anywhere in the image except on the targeted objects. We then switch to the more realistic case of *localized* indirect attacks, and design a group sparsity-based strategy to confine the perturbed region to a small area outside of the targeted objects. For our attacks to remain realistic and imperceptible, we perform them without ground-truth information about the dynamic objects and in a norm-bounded setting. In addition to these indirect attacks, we evaluate the robustness of state-of-the-art networks to a single universal fixed-size local perturbation that can be learned from all training images to attack an unseen image in an untargeted manner.

The conclusions of our experiments are disturbing: In short, more accurate semantic segmentation networks are more sensitive to indirect local attacks. This is illustrated by Figure 5.1, where perturbing a few patches in static regions has much larger impact on the dynamic objects for the context-aware PSPNet [310], PSANet [311] and DANet [71] than for a simple FCN [163]. This, however, has to be expected, because the use of context, which improves segmentation accuracy, also increases the network's receptive field, thus allowing the perturbation to be propagated to more distant image regions.

Motivated by this unsettling sensitivity of segmentation networks to indirect local attacks, we then turn our focus to adversarial attack detection. In contrast to the existing work that tackled attack detection for semantic segmentation [277], we perform detection not only at the global image level, but locally at the pixel level. Specifically, we introduce an approach to localizing the regions whose predictions were affected by the attack, i.e., not the image regions that were perturbed. In an autonomous driving scenario, this would allow one to focus more directly on the potential dangers themselves, rather than on the image regions that caused them.

To summarize, our contributions are as follows. We introduce the idea of indirect local adversarial attacks for semantic segmentation networks, which better reflects potential real-world dangers. We design an adaptive, image-dependent local attack strategy to find the minimal location to perturb in the static image region. We show the vulnerability of modern networks to a universal, image-independent adversarial patch. We study the impact of context on a network's sensitivity to our indirect local attacks. We introduce a method to detect indirect local attacks at both image level and pixel level. Our code is available at https://github.com/krishnakanthnakka/Indirectlocalattacks/.

5.2 Related Work

Context in Semantic Segmentation Networks. While context has been shown to improve the results of traditional semantic segmentation methods [76, 90, 126, 131], the early deep fully-convolutonal semantic segmentation networks [87, 163] only gave each pixel a limited receptive field, thus encoding relatively local relationships. Since then, several solutions have been proposed to account for wider context. For example, UNet [224] uses contracting path to capture larger context followed by a expanding path to upsample the intermediate low-resolution representation back to the input resolution. ParseNet [159] relies on global pooling of the final convolutional features to aggregate context information. This idea was extended to using different pooling strides in PSPNet [310], so as to encode different levels of context. In [291], dilated convolutions were introduced to increase the size of the receptive field. PSANet [311] is designed so that each local feature vector is connected to all the other ones in the feature map, thus learning contextual information adaptively. EncNet [300] captures context via a separate network branch that predicts the presence of the object categories in the scene without localizing them. DANet [71] uses a dual attention mechanism to attend to the most important spatial and channel locations in the final feature map. In particular, the DANet position attention module selectively aggregates the features at all positions using a weighted sum. In practice, all of these strategies to use larger contextual information have been shown to outperform simple FCNs on clean samples. Here, however, we show that this makes the resulting networks more vulnerable to indirect local adversarial attacks, even when the perturbed region covers less than 1% of the input image.

Adversarial Attacks on Semantic Segmentation: Adversarial attacks aim to perturb an input image with an imperceptible noise so as to make a DNN produce erroneous predictions. So far, the main focus of the adversarial attack literature has been image classification, for which diverse attack and defense strategies have been proposed [26, 52, 79, 138, 182, 196, 201]. In this context, it was shown that deep networks can be attacked even when one does not have access to the model weights [162, 199], that attacks can be transferred across different networks [254], and that universal perturbations that can be applied to any input image exist [180, 181, 212].

Motivated by the observations made in the context of image classification, adversarial attacks were extended to semantic segmentation. In [8], the effectiveness of attack strategies designed for classification was studied for different segmentation networks. In [284], a dense adversary generation attack was proposed, consisting of projecting the gradient in each iteration with minimal distortion. In [92], a universal perturbation was learnt using the whole image dataset. Furthermore, [10] demonstrated the existence of perturbations that are robust over chosen distributions of transformations.

None of these works, however, impose any constraints on the location of the attack in the input image. As such, the entire image is perturbed, which, while effective when the attacker has access to the image itself, would not allow one to physically modify the scene so as to fool, e.g., autonomous vehicles.

This, in essence, was the task first addressed in [21], where a universal targeted patch was shown to fool a recognition system to a specific target class. Later, patch attacks were studied in a more realistic setting in [62], where it was shown that placing a small, well-engineered patch on a traffic sign was able to fool a classification network into making wrong decisions. While these works focused on classification, patch attacks have been extended to object detection [161, 166, 227, 249] and optical flow [218]. Our work differs fundamentally from these methods in the following ways. First, none of these approaches optimize the location of the patch perturbation. Second, [21, 161, 227] learn a separate perturbation for every target class, which, at test time, lets the attacker change the predictions to one class only. While this is suitable for recognition, it does not apply to our segmentation setup, where we seek to misclassify the dynamic objects as different background classes so as to produce a realistic segmentation map. Third, unlike [21, 62, 218], our perturbations are imperceptible. Finally, while the perturbations in [62, 218, 249] cover the regions that should be misclassified, and in [161, 227] affect the predictions in the perturbed region, we aim to design an attack that affects only targeted locations outside the perturbed region.

In other words, we study the impact of *indirect* local attacks, where the perturbation is outside the targeted area. This would allow one to modify static portions of the scene so as to, e.g., make dynamic objects disappear to fool the self-driving system. Furthermore, we differ from these other patch-based attacks in that we study local attacks for semantic segmentation to understand the impact of the contextual information exploited by different networks, and introduce detection strategies at both image- and pixel-level.

Similarly to most of the existing literature [21, 62, 92, 156, 227, 249], we focus on the white-box setting for three reasons: (1) Developing effective defense mechanisms for semantic segmentation, which are currently lacking, requires assessing the sensitivity of semantic segmentation networks to the strongest attacks, i.e., white-box ones; (2) Recent model extraction methods [30, 200, 256] allow an attacker to obtain a close approximation of the deployed model. 3) While effective in classification [199], black-box attacks were observed to transfer poorly across semantic segmentation architectures [277], particularly in the targeted scenario.

When it comes to detecting attacks to semantic segmentation networks, only one technique have been proposed [277]. In [277], detection is achieved by checking the consistency of predictions obtained from overlapping image patches. In contrast to above work that need either multiple passes through the network, we detect the attack by analyzing the internal subspaces of the segmentation network. To this end, inspired by the algorithm of [142] designed for image classification, we compute the Mahalanobis distance of the features to pre-trained class conditional distributions. In contrast to [277], which study only global image-level detection, we show that our approach is applicable at both the image and the pixel level, yielding the first study on localizing the regions fooled by the attack.

5.3 Indirect Local Segmentation Attacks

Let us now introduce our diverse strategies to attack a semantic segmentation network. In semantic segmentation, given a clean image $\mathbf{X} \in \mathbb{R}^{W \times H \times C}$, where W, H and C are the width, height, and number of channels, respectively, a network is trained to minimize a loss function of the form $L(\mathbf{X}) = \sum_{j=1}^{W \times H} J(y_j^{true}, f(\mathbf{X})_j)$,

where J is typically taken as the cross-entropy between the true label y_j^{true} and the predicted label $f(\mathbf{X})_j$ at spatial location j. In this context, an adversarial attack is carried out by optimizing for a perturbation that forces the network to output wrong labels for some (or all) of the pixels. Below, we denote by $\mathbf{F} \in \{0,1\}^{W \times H}$ the fooling mask such that $\mathbf{F}_j = 1$ if the j-th pixel location is targeted by the attacker to be misclassified and $\mathbf{F}_j = 0$ is the predicted label should be preserved. We first present our different local attack strategies, and finally introduce our attack detection technique.

5.3.1 Indirect Local Attacks

To study the sensitivity of segmentation networks, we propose to perform local perturbations, confined to predefined regions such as class-specific regions or patches, and to fool other regions than those perturbed. For example, in the context of automated driving, we may aim to perturb only the regions belonging to the road to fool the car regions in the output label map. This would allow one to modify the physical, static scene while targeting dynamic objects.

Formally, given a clean image $\mathbf{X} \in \mathbb{R}^{W \times H \times C}$, we aim to find an additive perturbation $\delta \in \mathbb{R}^{W \times H \times C}$ within a perturbation mask $\mathbf{M}$ that yields erroneous labels within the fooling mask $\mathbf{F}$. To achieve this, we define the perturbation mask $\mathbf{M} \in \{0,1\}^{W \times H}$ such that $\mathbf{M}_j = 1$ if the j-th pixel location can be perturbed and $\mathbf{M}_j = 0$ otherwise.

Let $\mathbf{y}_j^{pred}$ be the label obtained from the clean image at pixel j. An untargeted attack can then be expressed as the solution to the optimization problem

$$\delta^* = \arg\min_{\delta} \sum_{j|\mathbf{F}_j=1} -J(\mathbf{y}_j^{pred}, f(\mathbf{X} + \mathbf{M} \odot \delta)_j) + \sum_{j|\mathbf{F}_j=0} J(\mathbf{y}_j^{pred}, f(\mathbf{X} + \mathbf{M} \odot \delta)_j) \tag{5.1}$$

which aims to minimize the probability of $\mathbf{y}_j^{pred}$ in the targeted regions while maximizing it in the rest of the image.

By contrast, for a targeted attack whose goal is to misclassify any pixel j in the fooling region to a pre-defined label $\mathbf{y}_j^t$, we write the optimization problem

$$\delta^* = \arg\min_{\delta} \sum_{j|\mathbf{F}_j=1} J(\mathbf{y}_j^t, f(\mathbf{X} + \mathbf{M} \odot \delta)_j) + \sum_{i|\mathbf{F}_j=0} J(\mathbf{y}_j^{pred}, f(\mathbf{X} + \mathbf{M} \odot \delta)_j) \; . \tag{5.2}$$

We solve (5.1) and (5.2) via the iterative projected gradient descent algorithm [168] with an ℓ_p-norm perturbation budget $\|\mathbf{M} \odot \delta\|_p < \epsilon$, where $p \in \{2, \infty\}$.

Note that the formulations above allow one to achieve any local attack. To perform *indirect* local attacks, we simply define the masks $\mathbf{M}$ and $\mathbf{F}$ so that they do not intersect, i.e., $\mathbf{M} \odot \mathbf{F} = \mathbf{0}$, where $\odot$ is the element-wise product.

5.3.2 Adaptive Indirect Local Attacks

The attacks described in Section 5.3.1 assume the availability of a fixed, predefined perturbation mask $\mathbf{M}$. In practice, however, one might want to find the best location for an attack, as well as make the attack as local as possible. In this section, we introduce an approach to achieving this by enforcing structured sparsity on the perturbation mask.

To this end, we first re-write the previous attack scheme under an ℓ_2 budget as an optimization problem. Let $J_t(\mathbf{X}, \mathbf{M}, \mathbf{F}, \delta, f, \mathbf{y}^{pred}, \mathbf{y}^t)$ denote the objective function of either (5.1) or (5.2), where $\mathbf{y}^t$ can be ignored in the untargeted case. Following [26], we write an adversarial attack under an ℓ_2 budget as the solution to the optimization problem

$$\delta^* = \arg\min_{\delta} \; \lambda_1 \|\delta\|_2^2 + J_t(\mathbf{X}, \mathbf{M}, \mathbf{F}, \delta, f, \mathbf{y}^{pred}, \mathbf{y}^t) \; , \tag{5.3}$$

where λ_1 balances the influence of the term aiming to minimize the magnitude of the perturbation. While solving this problem, we further constrain the resulting adversarial image $\mathbf{X}$+ $\mathbf{M} \odot \delta$ to lie in the valid pixel range [0,1].

To identify the best location for an attack together with confining the perturbations to as small an area as possible, we divide the initial perturbation mask $\mathbf{M}$ into T non-overlapping patches. This can be achieved by defining T masks $\{\mathbf{M}_t \in \mathbb{R}^{W \times H}\}$ such that, for any s, t, with $s \neq t$, $\mathbf{M}_s \odot \mathbf{M}_t = \mathbf{0}$, and $\sum_{t=1}^{T} \mathbf{M}_t = \mathbf{M}$. Our goal then becomes that of finding a perturbation that is non-zero in the smallest number of such masks.

This can be achieved by modifying (5.3) as

$$\delta^* = \arg\min_{\delta} \ \lambda_2 \sum_{t=1}^{T} \|\mathbf{M}_t \odot \delta\|_2 + \lambda_1 \|\delta\|_2^2 + J_t(\mathbf{X}, \mathbf{M}, \mathbf{F}, \delta, f, \mathbf{y}^{pred}, \mathbf{y}^t) \ , \tag{5.4}$$

whose first term encodes an $\ell_{2,1}$ group sparsity regularizer encouraging complete groups to go to zero. Such a regularizer has been commonly used in the sparse coding literature [197, 293], and more recently in the context of deep networks for compression purposes [3, 269]. In our context, this regularizer encourages as many as possible of the $\{\mathbf{M}_t \odot \delta\}$ to go to zero, and thus confines the perturbation to a small number of regions that most effectively fool the targeted area $\mathbf{F}$. λ_2 balances the influence of this term with the other ones.

5.3.3 Universal Local Attacks

The strategies discussed in Sections 5.3.1 and 5.3.2 are image-specific. However, [21] showed the existence of a universal perturbation patch that can fool an image classification system to output any target class. In this section, instead of finding optimal locations for a specific image, we aim to learn a single fixed-size local perturbation that can fool any unseen image in an untargeted manner. This will allow us to understand the contextual dependencies of a fixed size universal patch on the output of modern networks. Unlike in the above-mentioned adaptive local attacks, but as in [21, 162, 218, 227], such a universal patch attack will require a larger perturbation norm. Note also that, because it is image-independent, the resulting attack will typically not be indirect.

While [21] uses a different patch for different target classes, we aim to learn a single universal local perturbation that can fool all classes in the image. This will help to understand the propagation of the attack in modern networks using different long-range contextual connections. As will be shown in our experiments in Section 5.4.3, such modern networks are the most vulnerable to universal attacks, while their effect on FCNs is limited to the perturbed region. To find a universal perturbation effective across all images, we write the optimization problem

$$\delta^* = \arg\min_{\delta} \frac{1}{N} \sum_{i=1}^{N} J_u(\mathbf{X}^i, \mathbf{M}, \mathbf{F}^i, \delta, f, \mathbf{y}_i^{pred}) \ , \tag{5.5}$$

where $J_u(\cdot)$ is the objective function for a single image, as in the optimization problem (5.1), N is the number of training images, $\mathbf{X}^i$ is the i-th image with fooling mask $\mathbf{F}^i$, and the mask $\mathbf{M}$ is the global perturbation mask used for all images. In principle, $\mathbf{M}$ can be obtained by sampling patches over all possible image locations. However, we observed such a strategy to be unstable during learning. Hence, in our experiments, we confine ourselves to one or a few fixed patch positions. Note that, to give the attacker more flexibility, we take the universal attack defined in (5.5) to be an untargeted attack.

5.3.4 Adversarial Attack Detection

To understand the strength of the attacks discussed above, we introduce a detection method that can act either at the global image level or at the pixel level. In the former case, we employ pix2pix [107] generator to resynthesise the input image. We then compare the input image and resynthesised image in HOG feature space to detect the attack at global level. The latter is particularly interesting in the case of indirect attacks, where the perturbation regions and the fooled regions are different. In this case, our goal is to localize the pixels that were fooled, which is more challenging than finding those that were perturbed, since their intensity values were not altered. To this end, we use a score based on the Mahalanobis distance defined on the intermediate feature representations. This is because, as discussed in [142, 167] in the context of image classification, the attacked samples can be better characterized in the representation space than in the output label space.

Specifically, we use a set of training images to compute class-conditional Gaussian distributions, with class-specific means μ_c^ℓ and covariance $\mathbf{\Sigma}^\ell$ shared across all C classes, from the features extracted at every intermediate layer ℓ of the network within locations corresponding to class label c. We then define a confidence score for each spatial location j in layer ℓ as

$$C(\mathbf{X}_j^\ell) = \max_{c\in[1,C]} -\left(\mathbf{X}_j^\ell - \mu_c^\ell\right)^\top {\mathbf{\Sigma}_\ell}^{-1}\left(\mathbf{X}_j^\ell - \mu_c^\ell\right) \ , \tag{5.6}$$

where $\mathbf{X}_j^\ell$ denotes the feature vector at location j in layer ℓ. We handle the different spatial feature map sizes in different layers by resizing all of them to a fixed spatial resolution. We then concatenate the confidence scores in all layers at every spatial location and use the resulting L-dimensional vectors, with L being the number of layers, as input to a logistic regression classifier with weights $\{\alpha_\ell\}$. We then train this classifier to predict whether a pixel was fooled or not. At test time, we compute the prediction for an image location j as $\sum_\ell \alpha_\ell C(\mathbf{X}_j^\ell)$. To perform detection at the global image level, we sum over the confidence scores of all spatial positions. That is, for layer ℓ, we compute an image-level score as $C(\mathbf{X}^\ell) = \sum_j C(\mathbf{X}_j^\ell)$. We then train another logistic regression classifier using these global confidence scores as input.

5.4 Experiments

In this section, we first explain our experimental setup, and then analyze the vulnerability of state-of-the-art semantic segmentation networks to different types of attacks. Finally, we evaluate our image-level and pixel-level detection strategies.

Datasets. In our experiments, we use the Cityscapes [40] and Pascal VOC [60] datasets, the two most popular semantic segmentation benchmarks. Specifically, for Cityscapes, we use the complete validation set, consisting of 500 images, for untargeted attacks, but use a subset of 150 images containing dynamic object instances of vehicle classes whose combined area covers at least 8% of the image for targeted attacks. This lets us focus on fooling sufficiently large regions, because reporting results on too small one may not be representative of the true behavior of our algorithms. For Pascal VOC, we use 250 randomly selected images from the validation set because of the limited resources we have access to relative to the number of experiments we performed.

Models. We use publicly-available state-of-the-art models, namely FCN [163], DR-Net [291], PSPNet [310], PSANet [311], DANet [71] on Cityscapes, and FCN [163] and PSANet [311] on PASCAL VOC. FCN, PSANet, PSPNet and DANet share the same ResNet [89] backbone network. We perform all experiments at the image resolution of 512×1024 for Cityscapes and 512×512 for PASCAL VOC. Since different models can have different normalization strategies for the input image, we include normalization in the network and pass the network an input image scaled to [0,1].

Adversarial attacks. We use the iterative projected gradient descent (PGD) method with ℓ_∞ and ℓ_2 norm budgets, as described in Section 8.6. Following [8], we set the number of iterations for PGD to a maximum of 100, with an early termination criterion of 90% of attack success rate on the targeted objects. Given the dual objective of the loss functions in (5.1) and (5.2), it may happen that the gradients to maximize the confidence of labels at non-targeted locations dominate those at targeted ones. Hence, as suggested in [92], we ignore the loss at locations where the label is predicted correctly as the target label with a confidence of at least 0.3. We evaluate ℓ_∞ attacks with a step size $\alpha \in \{1\text{e-}5, 1\text{e-}4, 1\text{e-}3, 5\text{e-}3\}$. For ℓ_2 attacks, we set $\alpha \in \{8\text{e-}3, 4\text{e-}2, 8\text{e-}2\}$. We set the maximum ℓ_p-norm of the perturbation ϵ to $100 \cdot \alpha$ for ℓ_∞ attacks, and to 100 for ℓ_2 attacks. For universal attacks, we use a higher ℓ_∞ ϵ bound of 0.3, with a step size $\alpha = 0.001$. We perform two types of attacks; targeted and untargeted. The untargeted attacks focus on fooling the network to move away from the predicted label. For the targeted attacks, we chose a safety-sensitive goal, and thus aim to fool the dynamic object regions to be misclassified as their (spatially) nearest background label. We do not use ground-truth information in any of the experiments but perform attacks based on the predicted labels only. Note that predicted segmentation maps are very accurate, with state-of-the-art models reaching a pixel-wise accuracy $> 95\%$ on unattacked data. To be precise, we found the percentage of perturbed pixels lying within the targeted region to be $< 1\%$ in all cases. For example, in the adaptive attacks of Table 5.3, with S=75%, this percentage is 0.2%, 0.16%, 0.12%, 0.14% for FCN, PSANet, PSPNet, DANet, respectively, which shows that our attacks truly are indirect.

Evaluation metric. Following [8, 92, 284], we report the mean Intersection over Union (mIoU) and Attack Success Rate (ASR) computed over the entire dataset. The mIoU of FCN [163], DRNet [291], PSPNet [310], PSANet [311], and DANet [71] on clean samples at full resolution are 0.66, 0.64, 0.73, 0.72, and 0.67, respectively. For targeted attacks, we report the average ASR_t, computed as the percentage of pixels that were predicted

Network	$\alpha = 0.00001$	$\alpha = 0.0001$	$\alpha = 0.001$	$\alpha = 0.005$
FCN [163]	0.64 / 5.0%	0.28 / 29%	0.13 / 55%	0.11 / 61%
PSPNet [310]	0.70 / 12%	0.05 / **85%**	0.00 / 89%	0.00 / **90%**
PSANet [311]	0.59 / **14%**	0.03 / **85%**	0.01 / **90%**	0.00 / **90%**
DANet [71]	0.80 / 5.0%	0.11 / 79%	0.01 / **90%**	0.00 / **90%**
DRN [291]	0.64 / 6.0%	0.15 / 56%	0.03 / 84%	0.02 / 86%

(a) ℓ_∞ attack

Network	$\alpha = 0.008$	$\alpha = 0.04$	$\alpha = 0.08$
FCN [163]	0.60 / 10%	0.56 / 26%	0.27 / 36%
PSPNet [310]	0.67 / **19%**	0.23 / **67%**	**0.06 / 84%**
PSANet [311]	0.59 / 14%	0.21 / 63%	0.06 / 82%
DANet [71]	0.79 / 11%	0.43 / 49%	0.13 / 79%
DRN [291]	0.63 / 10%	0.24 / 47%	0.13 / 64%

(b) ℓ_2 attack

Table 5.1 – **Indirect attacks** on Cityscapes to fool dynamic classes while perturbing static ones. The numbers indicate mIoU$_u$/ASR$_t$, obtained using different step sizes α for ℓ_∞ and ℓ_2 attacks. The most robust network in each case is underlined and the most vulnerable models are highlighted in **bold**.

as the target label. We additionally report the mIoU$_u$, which is computed between the adversarial and normal sample predictions. For untargeted attacks, we report the ASR$_u$, computed as the percentage of pixels that were assigned to a different class than their normal label prediction. Since, in most of our experiments, the fooling region is confined to local objects, we compute the metrics only within the fooling mask. We observed that the non-targeted regions retain their prediction label more than 98% of the time. To evaluate detection, we report the Area under the Receiver Operating Characteristics (AUROC), both at image level, as in [277], and at pixel level.

5.4.1 Indirect Local Attacks

Let us study the sensitivity of the networks to indirect local attacks. In this setting, we first perform a targeted attack, formalized in (5.2), to fool the dynamic object areas by allowing the attacker to perturb any region belonging to the static object classes. This is achieved by setting the perturbation mask $\mathbf{M}$ to 1 at all the static class pixels and the fooling mask $\mathbf{F}$ to 1 at all the dynamic class pixels. We report the mIoU$_u$ and ASR$_t$ metrics in Tables 5.1a and 5.1b on Cityscapes for ℓ_∞ and ℓ_2 attacks, respectively. As evidenced by the tables, FCN is more robust to such indirect attacks than the networks that leverage contextual information. In particular, PSANet, which uses long range contextual dependencies, and PSPNet are highly sensitive to these attacks.

To further understand the impact of indirect *local* attacks, we constrain the perturbation region to a subset of the static class regions. To do this in a systematic manner, we perturb the static class regions that are at least d pixels away from any dynamic object, and vary the value d. The results of this experiment using ℓ_2 and ℓ_∞ attacks are provided in Table 5.2. Here, we chose a step size $\alpha = 0.005$ for ℓ_∞ and $\alpha = 0.08$ for ℓ_2. Similar conclusions to those in the previous non-local scenario can be drawn: Modern networks that use larger receptive fields are extremely vulnerable to such perturbations, even when they are far away from the targeted regions. By contrast, FCN is again more robust. For example, as shown in Figure 5.2, while an adversarial attack occurring 100 pixels away from the nearest dynamic object has a high success rate on the context-aware networks,

Network	$d = 0$	$d = 50$	$d = 100$	d = 150
FCN [163]	0.11 / 64%	0.77 / 2.0%	0.98 / 0%	1.00 / 0.0%
PSPNet [310]	0.00 / **90%**	0.14 / 73%	0.24 / 60%	0.55 / 23%
PSANet [311]	0.00 / **90%**	0.11 / 71%	0.13 / **65%**	0.29 / **47%**
DANet [71]	0.00 / **90%**	0.13 / **81%**	0.48 / 43%	0.80 / 10%
DRN [291]	0.02 / 86%	0.38 / 22%	0.73 / 3%	0.94 / 1.0%

(a) ℓ_∞ attack

Network	$d = 0$	$d = 50$	$d = 100$	d = 150
FCN [163]	0.27 / 36%	0.79 / 2.0%	0.98 / 2.0%	0.99 / 1.0%
PSPNet [310]	0.06 / **84%**	0.18 / 73%	0.55 / 23%	0.99 / 0.0%
PSANet [311]	0.06 / 82%	0.10 / **75%**	0.14 / **66 %**	0.31 / **44%**
DANet [71]	0.13 / 79%	0.27 / 71%	0.67 / 26%	0.85 / 7.0%
DRN [291]	0.13 / 64%	0.44 / 17%	0.76 / 3.0%	0.95 / 0.0%

(b) ℓ_2 attack

Table 5.2 – **Impact of local attacks** by perturbing pixels that are at least d pixels away from any dynamic class. We report mIoU$_u$/ASR$_t$ for different values of d.

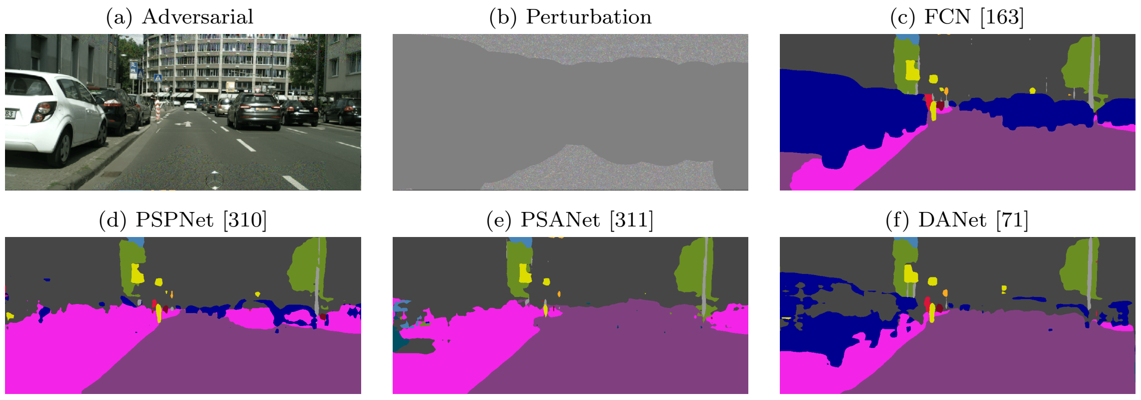


Figure 5.2 – **Indirect Local attack** on different networks with perturbations at least $d = 100$ pixels away from any dynamic class.

the FCN predictions remain accurate.

5.4.2 Adaptive Indirect Local Attacks

We now study the impact of our approach to adaptively finding the most sensitive context region to fool the dynamic objects. To this end, we use the group sparsity based optimization given in (5.4) and find the minimal perturbation region to fool all dynamic objects to their nearest static label. Specifically, we achieve this in two steps. First, we divide the perturbation mask $\mathbf{M}$ corresponding to all static class pixels into uniform patches of size $h \times w$, and find the most sensitive ones by solving (5.4) with a relatively large group sparsity weight $\lambda_2 = 100$ for Cityscapes and $\lambda_2 = 10$ for PASCAL VOC. Second, we limit the perturbation region by selecting the n patches that have the largest values $\|\mathbf{M}_t \odot \delta\|_2$, choosing n so as to achieve a given sparsity level $S \in \{75\%, 85\%, 90\%, 95\%\}$. Specifically, S is computed as the percentage of pixels that are not perturbed relative to the initial perturbation mask. We then re-optimize (5.4) with $\lambda_2 = 0$. In both steps, we set $\lambda_1 = 0.01$ and use the Adam optimizer [124] with a learning rate of 0.01. For Cityscapes, we use patch dimensions $h = 60$, $w = 120$, and, for PASCAL VOC, $h = 60$, $w = 60$. We clip the perturbation values below 0.005 to 0 at each iteration. This results in very local perturbation regions, active only in the most sensitive areas, as shown for PSANet in Figure 5.3 on Cityscapes and in Figure 5.4

for PASCAL VOC. As shown in Table 5.3, all context-aware networks are significantly affected by such perturbations, even when they are confined to small background regions. For instance, on Cityscapes, at a high sparsity level of 95%, PSANet yields an ASR_t of 44% compared to 1% for FCN. This means that, in the physical world, an attacker could add a small sticker at a static position to essentially make dynamic objects disappear from the network's view.

Network	$S = 75\%$	$S = 85\%$	$S = 90\%$	$S = 95\%$
FCN [163]	0.52 / <u>12%</u>	0.66 / <u>6%</u>	0.73 / <u>4%</u>	0.84 / <u>1.0%</u>
PSPNet [310]	0.19 / 70%	0.31 / 54%	0.41 / 42%	0.53 / 21%
PSANet [311]	0.10 / **78%**	0.16 / **71%**	0.20 / **64%**	0.35 / **44%**
DANet [71]	0.30 / 64%	0.52 / 43%	0.64 / 30%	0.71 / 21%
DRN [291]	0.42 / 23%	0.55 / 13%	0.63 / 9%	0.77 / 4.5%

(a) Cityscapes

Network	$S = 75\%$	$S = 85\%$	$S = 90\%$	$S = 95\%$
FCN [163]	0.80 / <u>12%</u>	0.66 / <u>22%</u>	0.59 / <u>27%</u>	0.50 / <u>32%</u>
PSANet [311]	0.30 / **69%**	0.20 / **80%**	0.21 / **77%**	0.28 / **68%**

(b) PASCAL VOC

Table 5.3 – **Adaptive indirect local attacks** on Cityscapes and PASCAL VOC. We report $\text{mIoU}_u/\text{ASR}_t$ for different sparsity levels S.

5.4.3 Universal Local Attacks

In this section, instead of considering image-dependent perturbations, we study the existence of universal local perturbations and their impact on semantic segmentation networks. In this setting, we perform untargeted *local* attacks by placing a fixed-size patch at a predetermined position. While the patch location can in principle be sampled at any location, we found learning its position to be unstable to due to the large number of possible patch locations in the entire dataset. Hence, here, we consider the scenario where the patch is located at the center of the image. We then learn a local perturbation that can fool the entire dataset of images for a given network by optimizing the objective given in (5.5). Specifically, the perturbation mask $\mathbf{M}$ is active only at the patch location and the fooling mask $\mathbf{F}$ at all image positions, i.e., at both static and dynamic classes. For Cityscapes, we learn the universal local perturbation using 100 images and use the remaining 400 images for evaluation purposes. For PASCAL VOC, we perform training on 100 images and evaluate on the remaining 150 images. We use ℓ_∞ optimization with $\alpha = 0.001$ for 200 epochs on the training set. We report the results of such universal patch attacks in Tables 5.4a and 5.4b on Cityscapes and PASCAL VOC for different patch sizes. As shown in the table, PSANet and PSPNet are vulnerable to such universal attacks, even when only 2.3% of the image area is perturbed. From Figure 5.5, we can see that the fooling region propagates to a large area far away from the perturbed one.

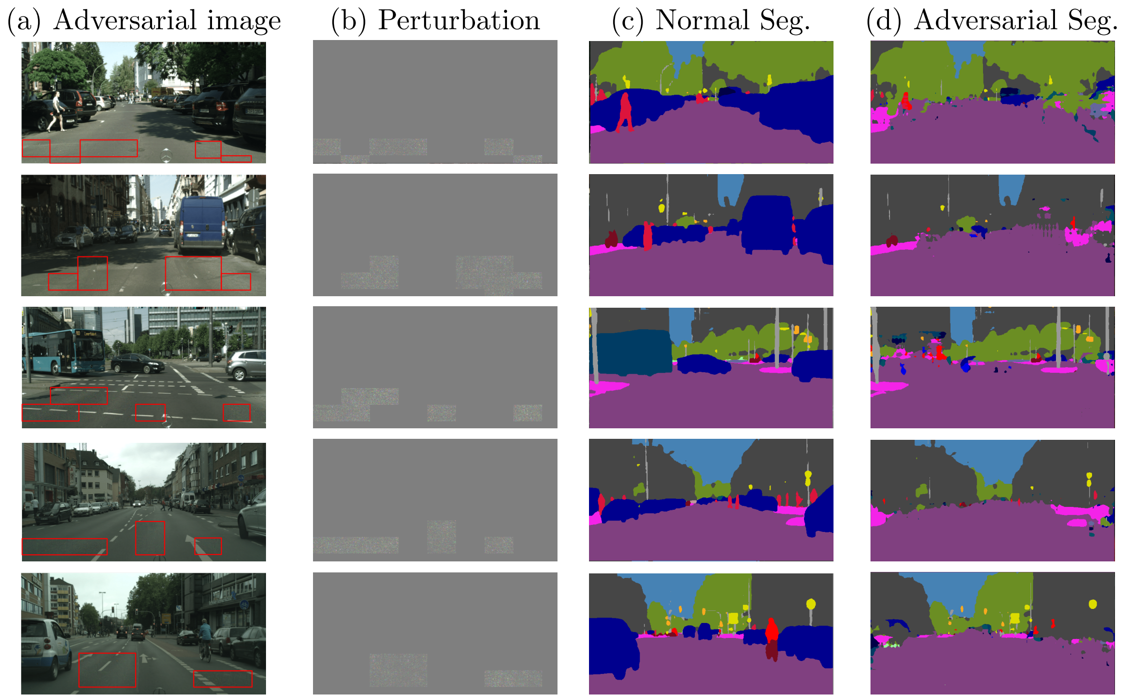


Figure 5.3 – **Adaptive indirect local attacks on Cityscapes with PSANet [311].** An adversarial input image **(a)** when attacked at positions shown as red boxes with a perturbation **(b)** is misclassified within the dynamic object areas of the normal segmentation map **(c)** to result in **(d)**.

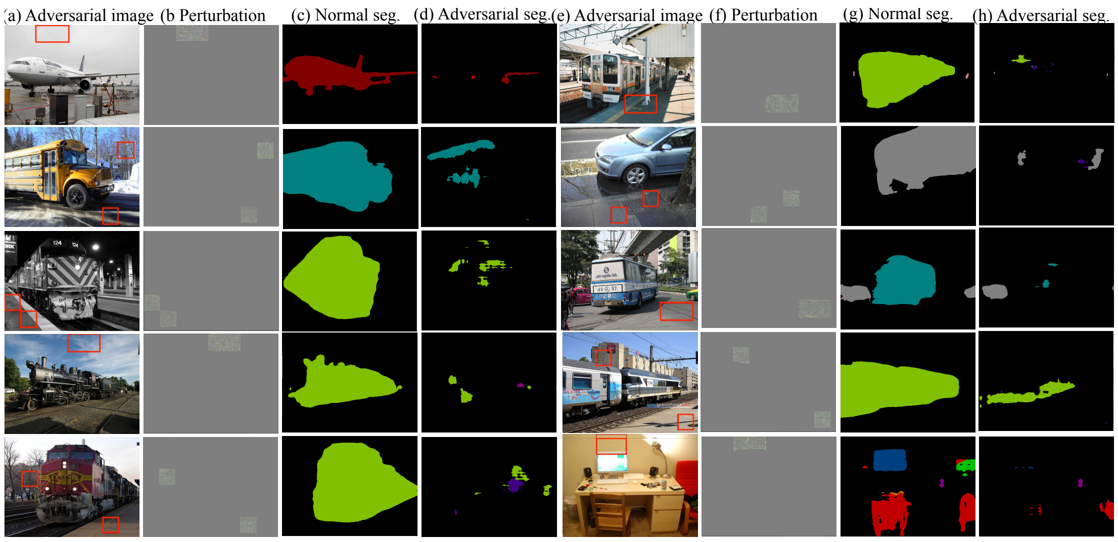
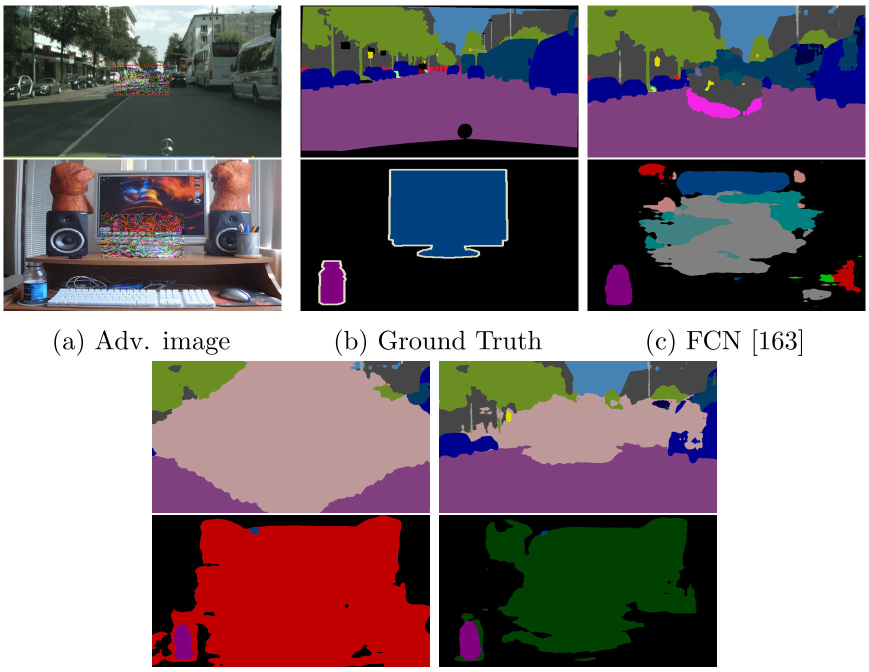


Figure 5.4 – **Adaptive indirect local attacks on PASCAL VOC with PSANet [311].** An adversarial input image **(a),(e)** when attacked at positions shown as red boxes with a perturbation **(b),(f)** is misclassified within the foreground object areas of the normal segmentation map **(c)**, **(g)** to result in **(d)**, **(h)**, respectively.



(a) Adv. image (b) Ground Truth (c) FCN [163]

(d) PSANet [311] (e) PSPNet [310]

Figure 5.5 – **Universal local attacks** on Cityscapes and PASCAL VOC. In both datasets, the degradation in FCN [163] is limited to the attacked area, whereas for context-aware networks, such as PSPNet [310], PSANet [311], DANet [71], it extends to far away regions.

Network	51×102(**1.0**%)	76×157(**2.3**%)	102×204(**4.0**%)	153×306(**9.0**%)
FCN [163]	0.85 / 2.0%	0.78 / 4.0%	0.73 / 9.0%	0.58 / 18%
PSPNet [310]	0.79 / 3.0%	0.63 / 11%	0.44 / 27%	0.08 / 83%
PSANet [311]	0.41 / **37%**	0.22 / **60%**	0.14 / 70%	**0.10** / **90%**
DANet [71]	0.79 / 4.0%	0.71 / 10%	0.65 / 15%	0.40 / 42%
DRN [291]	0.82 / 3.0%	0.78 / 8.0%	0.71 / 14%	0.55 / 28%

(a) Cityscapes

Network	51×51(**1.0**%)	76×76(**2.3**%)	102×102(**4.0**%)	153×153(**9.0**%)
FCN [163]	**0.70** / **6%**	0.70 / 7%	0.63 / 10%	0.52 / 20%
PSANet [311]	0.83 / 4%	0.76 / **8 %**	0.56 / **28%**	0.35/ **56%**

(b) PASCAL VOC

Table 5.4 – **Universal local attacks**. We show the impact of the patch size $h \times w$ (area%) on different networks and report mIoU$_u$/ASR$_u$.

5.5 Attack Detection

We now turn to studying the effectiveness of the attack detection strategies described in Section 5.3.4.

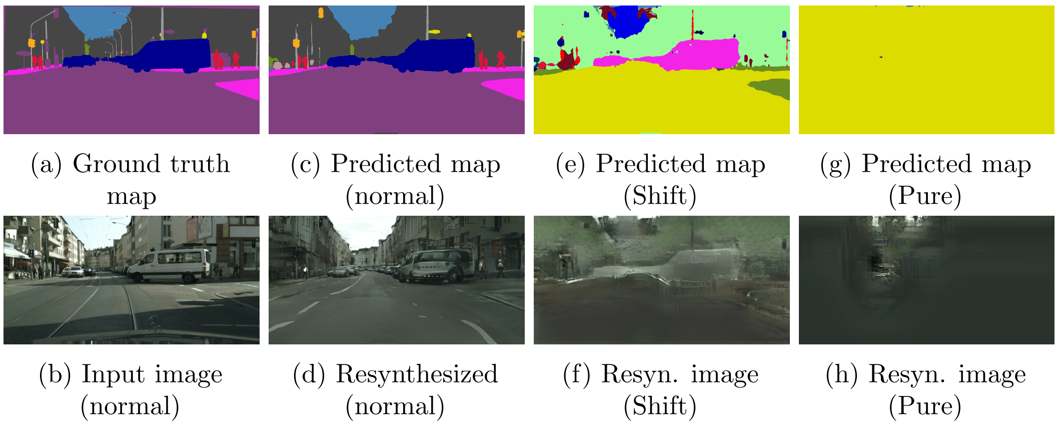


Figure 5.6 – **Visualizing adversarial attacks.** Without attacks, the resynthesized image **(d)**, obtained from **(c)**, looks similar to the input one **(b)**. By contrast, resynthesized images (**(f)** and **(h)**) obtained from the semantic maps (**(e)** and **(g)**) computed from an attacked input differ massively from the original one.

5.5.1 Image-Level Detecton

We now evaluate our approach to detecting attacks at image-level using the two types of attack that have been used in the context of semantic segmentation.

Adversarial Attacks: For semantic segmentation, similar to [277], we use the two state-of-the-art attack strategies namely Dense Adversary Generation (DAG) [284] and Houdini [39]. While DAG is an iterative gradient-based method, Houdini combines the standard task loss with an additional stochastic margin factor between the score of the actual and predicted semantic maps to yield less perturbed images. Following [277], we generate adversarial examples with two different target semantic maps. In the first case (Shift), we shift the predicted label at each pixel by a constant offset and use the resulting label as target. In the second case (Pure), a single random label is chosen as target for all pixels, thus generating a pure semantic map. We generate adversarial samples on the validation sets of the Cityscapes and BDD100K datasets, yielding 500 and 1000 images, respectively, with every normal sample having an attacked counterpart.

Results: We compare our method with the state-of-the-art spatial consistency (SC) work of [277], which crops random overlapping patches and computes the mean Intersection over Union (mIoU) of the overlapping regions. The results of this comparison are provided in Table 5.5. Ours (Syn) approach outperforms SC on Cityscapes and performs on par with it on BDD100K despite our use of a Cityscapes-trained generator to resynthesize the images. Note that, in contrast with SC, which requires comparing 50 pairs of patches to detect the attack, our approach only requires a single forward pass through the segmentation and generator networks. In Fig. 5.6, we show the resynthesized images produced when using adversarial samples. Note that they massively differ from the input one.

Dataset	Model	Method	DAG [284]		Houdini [39]	
			Pure	Shift	Pure	Shift
Cityscapes	BSeg	SC [277]	99%	98%	**100%**	**98%**
		Ours (Syn)	**100%**	**100%**	**100%**	**98%**
	PSP	SC [277]	98%	90%	98%	**100%**
		Ours (Syn)	**100%**	**99%**	**99%**	**100%**
BDD	BSeg	SC [277]	**100%**	**100%**	98%	**100%**
		Ours (Syn)	98%	98%	**100%**	90%
	PSP	SC [277]	92%	**100%**	96%	**100%**
		Ours (Syn)	**100%**	96%	**98%**	95%

Table 5.5 – **Attack detection on Cityscapes and BDD100K using image resynthesis.** Our resynthesis method achieves higher AUROC on Cityscapes than SC and comparable ones on BDD100K, despite the fact that we rely on a generator trained on Cityscapes.

5.5.2 Pixel-Level Detection

Networks	Perturbation region	Fooling region	ℓ_∞ / ℓ_2 norm	Mis. pixels %	Global AUROC SC [277] / Ours(Syn) / **Ours**(G)	Local AUROC **Ours**(L)
FCN [163]	Global	Full	0.10 / 17.60	90%	**1.00** / **1.00** / 0.94	0.90
	UP	Full	0.30 / 37.60	4%	0.71 / 0.63 / **1.00**	0.94
	FS	Dyn	0.07 / 2.58	13%	0.57 / 0.71 / **1.00**	0.87
	AP	Dyn	0.14 / 3.11	1.7%	0.51 / 0.65 / **0.87**	0.89
PSPNet [310]	Global	Full	0.06 / 10.74	83%	0.90 / **1.00** / 0.99	0.85
	UP	Full	0.30 / 38.43	11%	0.66 / 0.70 / **1.00**	0.96
	FS	Dyn	0.03 / 1.78	14%	0.57 / 0.75 / **0.90**	0.87
	AP	Dyn	0.11 / 5.25	11%	0.57 / 0.75 / **0.90**	0.82
PSANet [311]	Global	Full	0.05 / 8.26	92%	0.90 / 1.00 / 1.00	0.67
	UP	Full	0.30 / 38.6	60%	0.65 / 1.00 / 1.00	0.98
	FS	Dyn	0.02 / 1.14	12%	0.61 / 0.76 / **1.00**	0.92
	AP	Dyn	0.10 / 5.10	10%	0.50 / 0.82 / **1.00**	0.94
DANet [71]	Global	Full	0.06 / 12.55	82%	0.89 / 1.00 / 1.00	0.68
	UP	Full	0.30 / 37.20	10%	0.67 / 0.63 / **0.92**	0.89
	FS	Dyn	0.05 / 1.94	13%	0.57 / 0.69 / **0.94**	0.88
	AP	Dyn	0.14 / 6.12	43%	0.59 / 0.68 / **0.98**	0.82

Table 5.6 – **Attack detection** on Cityscapes with different perturbation settings.

While the above methods were designed to handle attacks that fool the entire label map, unlike in our work where we aim to fool local regions. Furthermore, the earlier methods perform detection at the image level, and thus do not localize the fooled regions at the pixel level. We denote Ours (G) and Ours (L) to detect the global and pixel-level attack using the Mahalanobis distance metric.

We study detection in four perturbation settings: Global image perturbations (Global) to fool the entire image; Universal patch perturbations (UP) at a fixed location to fool the entire image; Full static (FS) class perturbations to fool the dynamic classes; Adaptive patch (AP) perturbations in the static class regions to fool the dynamic objects. As shown in Table 5.6, while the state-of-the-art method [277] have high Global AUROC in the first setting where the entire image is targeted, our detection strategy outperforms them by a large margin in the other scenarios. We believe this to be due to the fact that, with local attacks, the statistics obtained by studying the consistency across local patches, as in [277], are much closer to the clean image statistics. Similarly, the image re-synthesized by a pix2pix generator using the image-level method will look much more similar to the input one in the presence of local attacks instead of global ones. For all the perturbation settings, we also report the mean percentage of pixels misclassified relative to the number of pixels in the image.

5.5.3 Implementation Details

In this section, we provide detailed explanations about the experiments described in Section 4 of the main chapter.

Models. All models for the experiments were implemented in PyTorch [205]. To generate adversarial attacks, we use the advertorch [49] library. Since different networks may have different normalization values for the mean and standard deviation of the input, we model normalization as a first layer inside the network and pass it as input an RGB image scaled to the range [0,1].

FCN. We use the publicly released model[1] from the authors of [311], which is trained together with PSANet [311] with an additional auxiliary loss. We use the ResNet-50 version for our experiments.

PSPNet. We use the trained modelpsalink released by the authors of [311]. It uses the same ResNet-50 as backbone network. The pyramid pooling module is a 4-level pyramid, which is concatenated to the final convolutional spatial map and later fed to a classification layer.

PSANet. We experiment with the trained modelpsalink provided by authors of [311] with ResNet-50 as backbone network. The PSA layer contains two sub-branches, namely collect and distribute, that favor a bi-directional information flow from each position to all other positions in the spatial feature map.

DANet. We use the trained model[2] from the authors of DANet [71]. DANet uses ResNet-101 as backbone network followed by a spatial and channel wise attention module. We use DANet with a hierarchy of grids of different sizes (4,8,16) in the last layer of each ResNet block.

[1]https://github.com/hszhao/semseg
[2]https://github.com/junfu1115/DANet

Dynamic class	Images
Person	115
Rider	66
Car	150
Truck	33
Bus	23
Train	7
Motorcycle	24
Bicycle	88

Table 5.7 – **Cityscapes-sampled dataset.** We provide the statistics of the 150 images whose combined instance area of *vehicle* categories is more than 8%.

DRN. We use the trained model[3] released by authors of [291]. We choose ResNet-22 as backbone network with dilated version corresponding to type *D*.

U-Net. Along with the above-mentioned models, we evaluate the robustness of the U-Net architecture to local attacks. Due to the non-availability of a trained PyTorch [205] version of the U-Net model, we re-trained it ourselves, achieving 33.7% mIoU on Cityscapes.

Along with the six Cityscapes models discussed above, we experiment on PASCAL VOC [60] with trained FCN [163]psalink and PSANet [311]psalink models provided by the authors of [311].

Datasets

Cityscapes: We use the validation set of the Cityscapes [40] dataset consisting of 500 images from 19 classes. We divide the pixels at every position in the image into one of two sets, based on the category attribute provided by the authors. The first set consists of pixels belonging to static classes with category attribute *road, sidewalk, building, wall, fence, pole, traffic light, traffic sign, vegetation, terrain, sky.* The second set corresponds to regions of dynamic classes *person, rider, car, truck, bus, train, motorcycle, bicycle.*

The Cityscapes dataset has on average 8% of the pixels corresponding to dynamic classes in each image. Since our study was targeted to mis-classify the dynamic objects, images with dynamic instances that occupy small regions might not be meaningful as such regions lie in the immediate receptive field of their surroundings. Therefore, we take a subset of images consisting of 150 images whose combined region of instances corresponding to vehicle classes (*car, truck, bus, train, motorcycle, bicycle*) is greater than 8%. We provide the statistics of the resulting dataset in Table 5.7.

While the original Cityscapes dataset was captured at 2048 $\times$ 1024 resolution, we resize

[3]https://github.com/fyu/drn

the images to the half resolution of 1024 × 512 as the original size is too large to fit into GPU memory. Furthermore, we crop the bottom region of the image corresponding to the ego-vehicle of height 62 pixels and resize the image back to 1024 × 512 pixels. For fair comparison, all models use the same 1024 × 512 resolution as input to the network without any tiling.

PASCAL VOC: We use a subset of 250 images from the original validation set consisting of 1449 images. It contains 20 foreground classes and one background class. In all settings, we target the pixels corresponding to all 20 foreground classes by perturbing a subset of the background area.

Attack Algorithms

We solve the indirect attacks given in Sections 3.1 and 3.2 of the main chapter using the efficient iterative projected gradient descent algorithm [168] with an ℓ_p-norm perturbation budget $\|\mathbf{M} \odot \delta\|_p < \epsilon$, where $p \in \{2, \infty\}$, using a step size α. In all our experiments, we set the maximum perturbation ϵ as 100 times α for ℓ_∞ attacks. For ℓ_2 attacks, we set the maximum ℓ_2 norm of the perturbation ϵ to 100.

Formally, given an input image $\mathbf{X}$, the adversarial attack minimizes the objective function, $J_t(\mathbf{X}, \mathbf{M}, \mathbf{F}, \delta, f, \mathbf{y}^{pred}, \mathbf{y}^t)$ to find the optimal δ. We solve for δ in an iterative manner as

$$\delta^{(0)} = \mathbf{0} \tag{5.7}$$

$$\delta^{(n+1)} = \text{Clip}_\varepsilon^p \left\{ \delta^{(n)} - \alpha \nabla_\mathbf{X} J_t(\mathbf{X}, \mathbf{M}, \mathbf{F}, \delta, f, \mathbf{y}^{pred}, \mathbf{y}^t) \right\}, \tag{5.8}$$

where $\text{Clip}_\varepsilon^p$ clips the perturbation within the ℓ_p ball of radius ϵ. For ℓ_∞-norm based attacks, the gradient update is given by

$$\nabla_\mathbf{X} J = (\nabla_\mathbf{X}(J_t(\mathbf{X}, \mathbf{M}, \mathbf{F}, \delta, f, \mathbf{y}^{pred}, \mathbf{y}^t))), \tag{5.9}$$

where is the sign function.

For ℓ_2-norm based attacks, the gradient update is given by

$$r = \nabla_\mathbf{X}(J_t(\mathbf{X}, \mathbf{M}, \mathbf{F}, \delta, f, \mathbf{y}^{pred}, \mathbf{y}^t)) \tag{5.10}$$

$$\nabla_\mathbf{X} J = \frac{r}{\|r\|_2}. \tag{5.11}$$

We observe that the DAG attack [284] is similar to the PGD-ℓ_2 attack. While DAG projects the gradient as $\frac{r}{\|r\|_\infty}$, PGD-ℓ_2 projects the gradient as $\frac{r}{\|r\|_2}$. We emphasize that our formalism for local indirect attacks is general and could be applied to other adversary generation techniques [26, 284].

Attack Detection Algorithms

State-of-the-art methods. In this chapter, we compare our approach with the spatial consistency [277] method for adversarial attack detection at image level. Following [277], given an input image of 1024×512 pixels, we crop 50 sufficiently overlapping pairs of patches of size 256×256 and compute the average mIoU of the overlapped patch regions as the confidence score for attack detection.

Our method. Let us now provide the implementation details of our attack detection based on the Mahalanobis distance. During training, we compute the class-conditional mean μ_c^ℓ at every layer ℓ of the network within locations corresponding to class label c of the ground truth. Furthermore, we compute the group variance $\mathbf{\Sigma}^\ell$ for every layer ℓ of the network using the features extracted at layer ℓ. Since the number of features extracted from the training set can be high, we propose to compute the mean and variance of averaged features within locations corresponding to each label.

Formally, let $\mathbf{X}_j^\ell$ be the feature extracted at layer ℓ at position j for image $\mathbf{X}$. Let the size of the feature map $\mathbf{X}^\ell$ be $W_\ell \times H_\ell \times K_\ell$, where W_ℓ, H_ℓ, K_ℓ are the width, height and number of channels for layer ℓ. Let $L^c \in \mathbb{R}^{W_\ell \times H_\ell}$ be the label mask activated at positions where the label is c, i.e., $L_j^c = 1$ if the j-th pixel location belongs to label c and $L_j^c = 0$ otherwise.

First, we compute the averaged feature corresponding to label c given by $\hat{\mathbf{X}}_c^\ell = \sum_{j|L_j=1} \mathbf{X}_j^\ell$. We then learn μ_c^ℓ and $\mathbf{\Sigma}^\ell$ using $\{\hat{\mathbf{X}}_c^\ell | \mathbf{X} \in [\mathbf{X}_0, ..., \mathbf{X}_N]\}$ extracted from all N images in the training set. In the end, we obtain $\mu_c^\ell \in \mathbb{R}^{K_\ell}$ and $\mathbf{\Sigma}^\ell \in \mathbb{R}^{\mathbb{K}_\ell \times \mathbb{K}_\ell}$ for a layer ℓ in the network, and use these values to compute the confidence score of Eq.(6)

We extract features at the end of every block in the ResNet backbone followed by a context layer and a classification layer. By doing so, we obtain a feature vector for the logistic detector of size $L = 6$ for FCN; $L = 7$ for PSANet; $L = 7$ for PSPNet; $L = 5$ for DANet; $L = 5$ for DRN. For evaluation purpose, we use 80% of the data for training and the remaining 20% for testing.

Performance Metrics

For evaluation, we use the following metrics to measure the effectiveness of our indirect local attacks. **Intersection over Union.** We report the mIoU used in the domain of segmentation to evaluate the effectiveness of the attack. We report the mIoU at positions that we aim to fool (f) since at the remaining positions, the label is retained around 98% of the time. For untargeted attacks, we report $\text{mIoU}_{\mathbf{u}}^f$ as the mIoU calculated between the normal image prediction and its counterpart adversarial image prediction at fooling positions. In the case of targeted attacks, along with $\text{mIoU}_{\mathbf{u}}^f$, we report $\text{mIoU}_{\mathbf{t}}^f$ as the mIoU calculated between the normal image prediction and targeted label map at fooling positions.

Attack Success Rate. We report the attack success rate at the percentage of pixels mis-classified/preserved relative to the total number of pixels in the fooling/preserved positions, respectively. We report the mASR separately at two positions: 1) at positions that we aim to fool (f); 2) at the remaining positions where the label should be preserved *(p)*. We report $\mathrm{mASR}^{f}_{\mathbf{u}}$ and $\mathrm{mASR}^{p}_{\mathbf{u}}$ as the success rates calculated between the normal prediction and its adversarial image prediction at the fooling and preserved positions, respectively, for untargeted attacks. Specifically to calculate $\mathrm{mASR}^{f}_{\mathbf{u}}$, we assume the attack to be successful at a pixel if it misclassifies it to any label other than the normal predicted label. In the case of targeted attacks, we additionally report $\mathrm{mASR}^{f}_{\mathbf{t}}$ as the success rates calculated between the normal prediction and targeted label map at fooling positions.

Perceptibility. We take the ℓ_∞-norm and ℓ_2-norm of the perturbation image as the two perceptibility scores.

We average the above metrics over the entire test set. Since in almost all experiments the labels are retained around 98% of the time at preserved positions, we omitted reporting mASR^{p}_{u} in the main chapter. We reported only $\mathrm{mASR}^{f}_{\mathbf{t}}$ and $\mathrm{mIoU}^{f}_{\mathbf{u}}$ at the fooling positions in the main chapter as these metrics values are the most diverse in our different attack settings.

AUROC. The area under the receiver operating characteristic curve (AUROC) is computed by plotting the true positive rate (TPR) against the false positive rate (FPR) by varying a threshold. We compute the AUROC both at image level and pixel level and report them in all perturbation settings.

Time Complexity

For an input image of size 512×1024, the PGD-based indirect attack of Eq. (2) in the main chapter takes on average $\sim$ 35 seconds for 100 iterations, whereas our group-sparsity-based attack in Eq. (4) of the main chapter takes on average $\sim$ 90 seconds when using a maximum of 400 gradient computations. For comparison, a dense adversary generation attack [284], consisting of projecting the gradient in each iteration, takes $\sim$ 40 seconds for a maximum of 200 iterations. Importantly, these timings remain practical in the scenario of physical attacks where the perturbation can be computed offline.

5.6 Additional Results

Cityscapes Experiments

Table 5.8 show the performance of different networks by varying the noise levels for ℓ_∞ attacks. Table 5.9 show the impact of indirect attacks by perturbing static regions that are at least d pixels away from any dynamic object class with ℓ_∞ attacks. Furthermore, Table 5.10 shows the complete performance statistics of different networks by tuning the sparsity levels in our adaptive attack strategy. We then show the impact of universal, single fixed-size patch attacks in Table 5.11 by varying the size of the patch placed at the center of the image.

Finally, we show the attack detection results with four perturbation settings: Global image perturbations (Global) to fool the entire image; Universal patch perturbations (UP) at a fixed location to fool the entire image; Full static (FS) class perturbations to fool the dynamic classes; Adaptive patch (AP) perturbations in the static class regions to fool the dynamic objects.

Qualitative Results on Cityscapes

Figure 5.7 shows the outputs of indirect local attacks by perturbing static class pixels that are at least a distance d from a dynamic class pixel. Figure 5.8 shows the outputs of universal patch attacks on different networks by varying the patch area in $\{1\%, 2.3\%, 4\%, 9\%\}$ of the image area. Figure 5.9 shows the results of adaptive local attacks on different networks by varying the sparsity level of the perturbation. Finally, Figure 5.10 shows the resynthesised images for normal and attacked images.

Networks	α	mIoU		mASR			Norm of δ	
		$\text{mIoU}^f_{\mathbf{u}}$	$\text{mIoU}^f_{\mathbf{t}}$	$\text{mASR}^p_{\mathbf{u}}$	$\text{mASR}^f_{\mathbf{u}}$	$\text{mASR}^f_{\mathbf{t}}$	ℓ_∞-norm	ℓ_2-norm
FCN [163]	1e-5	0.65	0.08	100%	6%	5%	0.001	0.83
	1e-4	0.29	0.27	100%	35%	29%	0.01	4.70
	1e-3	0.14	0.49	100%	63%	56%	0.10	15.12
	5e-3	0.11	0.55	100%	69%	62%	0.40	50.93
PSPNet [310]	1e-5	0.71	0.10	99%	15%	12%	0.001	0.77
	1e-4	0.06	0.53	100%	98%	86%	0.01	3.10
	1e-3	0.00	0.62	100%	100%	90%	0.05	8.30
	5e-3	0.00	0.63	99%	100%	90%	0.20	37.99
PSANet [311]	1e-5	0.60	0.10	98%	22%	14%	0.001	0.72
	1e-4	0.04	0.51	99%	99%	86%	0.01	2.68
	1e-3	0.01	0.60	99%	100%	90%	0.05	8.10
	5e-3	0.00	0.60	99%	100%	90%	0.18	35.71
DANet [71]	1e-5	0.80	0.06	100%	6%	5%	0.001	0.81
	1e-4	0.11	0.50	99%	91%	80%	0.01	3.90
	1e-3	0.01	0.65	99%	99%	90%	0.04	8.30
	5e-3	0.00	0.66	99%	100%	90%	0.15	31.71
DRNet [291]	1e-5	0.64	0.09	99%	9%	6%	0.001	0.87
	1e-4	0.15	0.44	99%	67%	56%	0.01	4.95
	1e-3	0.03	0.67	99%	92%	84%	0.08	12.78
	5e-3	0.02	0.67	99%	94%	87%	0.27	40.2
U-Net [224]	1e-5	0.35	0.15	99%	29%	20%	0.001	0.91
	1e-4	0.02	0.37	99%	95%	76%	0.01	5.74
	1e-3	0.00	0.48	99%	99%	87%	0.08	13.34
	5e-3	0.00	0.52	99%	100%	89%	0.28	38.89

Table 5.8 – **Indirect attacks** on Cityscapes to fool dynamic classes while perturbing entire static ones with ℓ_∞ strategy. The success rate of the attacks increases with higher step size α although with higher perceptibility values. FCN is more robust to indirect attacks, while PSANet and PSPNet are more vulnerable to attacks even at small step sizes such as α = 1e-4.

Networks	d	mIoU		mASR			Norm of δ	
		$\text{mIoU}_{\mathbf{u}}^{f}$	$\text{mIoU}_{\mathbf{t}}^{f}$	$\text{mASR}_{\mathbf{u}}^{p}$	$\text{mASR}_{\mathbf{u}}^{f}$	$\text{mASR}_{\mathbf{t}}^{f}$	ℓ_∞-norm	ℓ_2-norm
FCN [163]	50	0.77	0.05	100%	4%	3%	0.38	43.37
	100	0.98	0.00	100%	0%	0%	0.38	33.46
	150	1.00	0.00	100%	0%	0%	0.38	22.23
PSPNet [310]	50	0.14	0.37	99%	96%	74%	0.28	41.83
	100	0.24	0.26	98%	86%	60%	0.29	33.00
	150	0.55	0.12	97%	35%	23%	0.34	22.86
PSANet [311]	50	0.11	0.33	98%	98%	72%	0.25	42.11
	100	0.13	0.27	98%	97%	65%	0.25	33.00
	150	0.28	0.21	98%	75%	47%	0.30	22.47
DANet [71]	50	0.14	0.50	99%	92%	81%	0.29	41.17
	100	0.48	0.24	98%	53%	43%	0.33	34.50
	150	0.80	0.07	98%	14%	10%	0.35	23.45
DRNet [291]	50	0.37	0.20	99%	34%	22%	0.43	46.30
	100	0.73	0.05	99%	5%	3%	0.44	37.24
	150	0.94	0.00	100%	0%	0%	0.47	25.87
U-Net [224]	50	0.01	0.25	98%	97%	70%	0.43	44.62
	100	0.03	0.20	96%	90%	60%	0.47%	39.61
	150	0.10	0.17	95%	74%	47%	0.49%	33.27

Table 5.9 – **Impact of local attacks** by perturbing pixels that are at least d pixels away from any dynamic class with ℓ_∞ *strategy.* We observe PSANet [311] and UNet [224] to be vulnerable to indirect attacks even when the perturbations are at large distances, such as $d = 150$, while FCN [163] is barely affected.

Networks	Sparsity	mIoU		mASR			Norm of δ	
		$\text{mIoU}_{\mathbf{u}}^{f}$	$\text{mIoU}_{\mathbf{t}}^{f}$	$\text{mASR}_{\mathbf{u}}^{p}$	$\text{mASR}_{\mathbf{u}}^{f}$	$\text{mASR}_{\mathbf{t}}^{f}$	ℓ_∞-norm	ℓ_2-norm
FCN [163]	75%	0.52	0.12	100%	18%	13%	0.15	4.04
	85%	0.67	0.07	100%	9%	6%	0.14	3.11
	90%	0.73	0.05	100%	6%	4%	0.12	2.54
	95%	0.84	0.03	100%	2%	2%	0.10	1.78
PSPNet [310]	75%	0.19	0.38	99%	89%	71%	0.09	4.87
	85%	0.32	0.28	98%	74%	55%	0.11	5.25
	90%	0.42	0.21	98%	60%	42%	0.13	5.30
	95%	0.60	0.11	98%	33%	22%	0.15	4.85
PSANet [311]	75%	0.10	0.44	99%	97%	79%	0.09	4.76
	85%	0.16	0.38	98%	94%	71%	0.10	5.20
	90%	0.20	0.32	98%	89%	64%	0.12	5.19
	95%	0.36	0.22	98%	70%	44%	0.14	5.07
DANet [71]	75%	0.30	0.37	99%	78%	65%	0.12	5.63
	85%	0.49	0.23	99%	57%	46%	0.14	5.79
	90%	0.64	0.16	99%	40%	30%	0.15	5.80
	95%	0.71	0.12	99%	29%	21%	0.13	3.95
DRNet [291]	75%	0.42	0.19	100%	35%	22%	0.18	5.40
	85%	0.55	0.11	100%	22%	13%	0.15	4.43
	90%	0.63	0.08	100%	15%	10%	0.14	3.84
	95%	0.77	0.05	100%	8%	5%	0.13	2.81
U-Net [224]	75%	0.12	0.20	96%	70%	44%	0.15	6.56
	85%	0.19	0.15	96%	52%	32%	0.19	6.81
	90%	0.25	0.13	96%	42%	25%	0.22	6.54
	95%	0.36	0.11	96%	27%	16%	0.23	5.73

Table 5.10 – **Adaptive indirect local attacks** on Cityscapes. We compute the performance statistics for different sparsity levels of perturbation. By enforcing group sparsity, we can attack context-aware networks such as PSANet [311], PSPNet [310] and DANet [71] with higher success rates than for the baseline FCN [163].

Networks	Patch size $h{\times}w$ (area%)	mIoU	mASR	Norm of δ	
		$\text{mIoU}_{\mathbf{u}}^{f}$	$\text{mASR}_{\mathbf{u}}^{f}$	ℓ_∞-norm	ℓ_2-norm
FCN [163]	51×102 (**1.0**%)	0.86	2%	0.30	25.36
	76×157 (**2.3**%)	0.78	4%	0.30	37.60
	102×204 (**4.0**%)	0.73	10%	0.30	51.80
	153×306 (**9.0**%)	0.58	18%	0.30	78.32
PSPNet [310]	51×102 (**1.0**%)	0.80	3%	0.30	25.52
	76×157 (**2.3**%)	0.63	10%	0.30	38.43
	102×204 (**4.0**%)	0.44	27%	0.30	50.32
	153×306 (**9.0**%)	0.09	84%	0.30	74.92
PSANet [311]	51×102 (**1.0**%)	0.41	38%	0.30	26.69
	76×157 (**2.3**%)	0.23	60%	0.30	38.60
	102×204 (**4.0**%)	0.14	71%	0.30	50.39
	153×306 (**9.0**%)	0.04	90%	0.30	78.02
DANet [71]	51×102 (**1.0**%)	0.79	4%	0.30	26.45
	76×157 (**2.3**%)	0.71	10%	0.30	37.24
	102×204 (**4.0**%)	0.65	15%	0.30	49.86
	153×306 (**9.0**%)	0.40	42%	0.30	74.60
DRNet [291]	51×102 (**1.0**%)	0.82	2%	0.30	26.28
	76×157 (**2.3**%)	0.77	7%	0.30	39.27
	102×204 (**4.0**%)	0.70	14%	0.30	52.23
	153×306 (**9.0**%)	0.55	28%	0.30	78.32
U-Net [224]	51×102 (**1.0**%)	0.32	26%	0.30	29.95
	76×157 (**2.3**%)	0.13	58%	0.30	44.42
	102×204 (**4.0**%)	0.06	76%	0.30	58.15
	153×306 (**9.0**%)	0.02	90%	0.30	86.06

Table 5.11 – **Universal local attacks** on Cityscapes by tuning the patch size $h{\times}w$ (area%) on different networks. PSANet [311] and UNet [224] are highly sensitive to patch attacks even when the patch is 1% of image area. Note that the attack is untargeted and aimed to fool the entire scene by placing a fixed-size patch at the center of the image. We use ℓ_∞ based attacks with $\alpha = 0.001$ and $\epsilon = 0.3$.

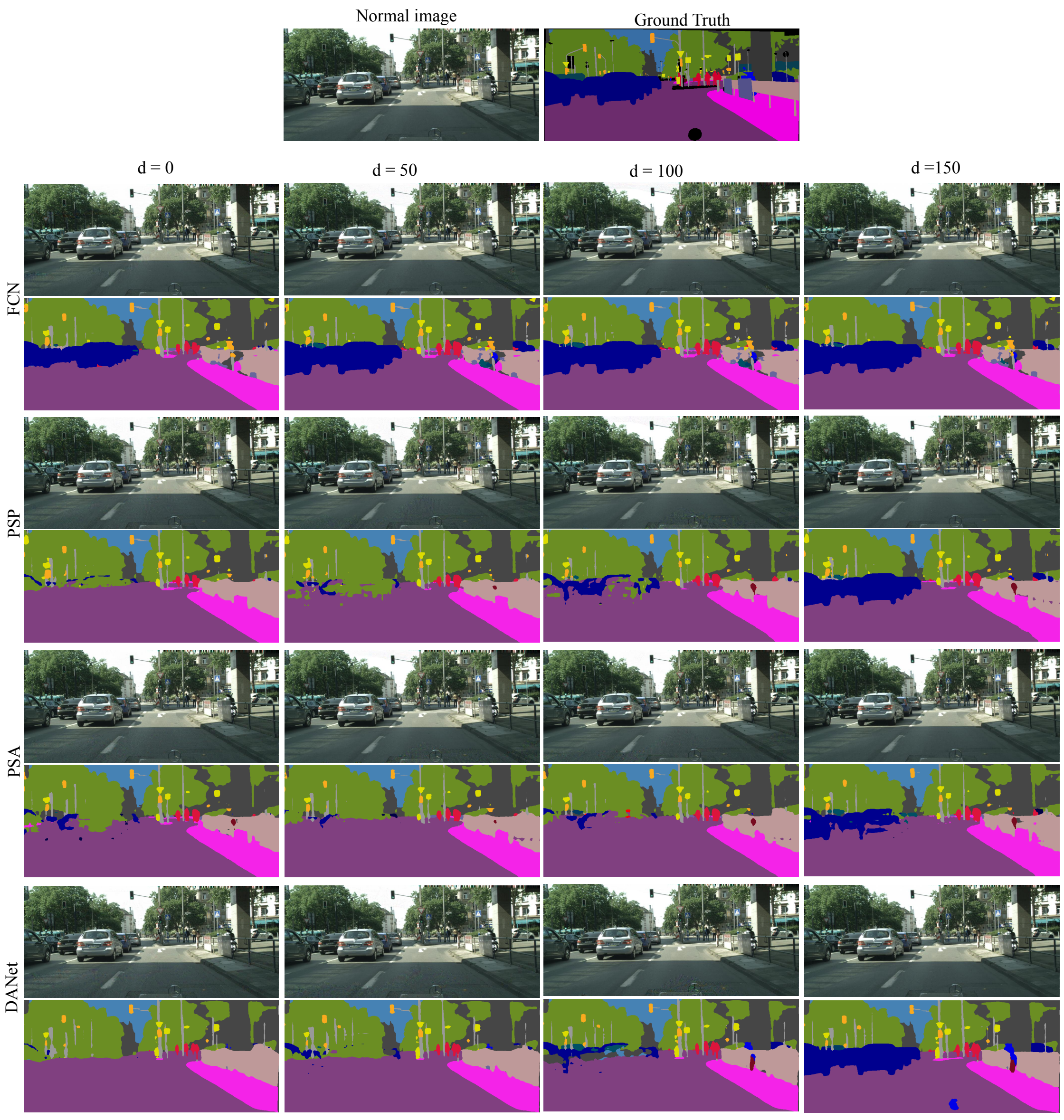


Figure 5.7 – **Indirect local attack** on different networks with perturbations at least d pixels away from any dynamic class. In most cases, FCN [163] is not affected by indirect attacks, while PSANet [311], PSPNet [310] and DANet [71] are affected due to their larger contextual dependencies for prediction.

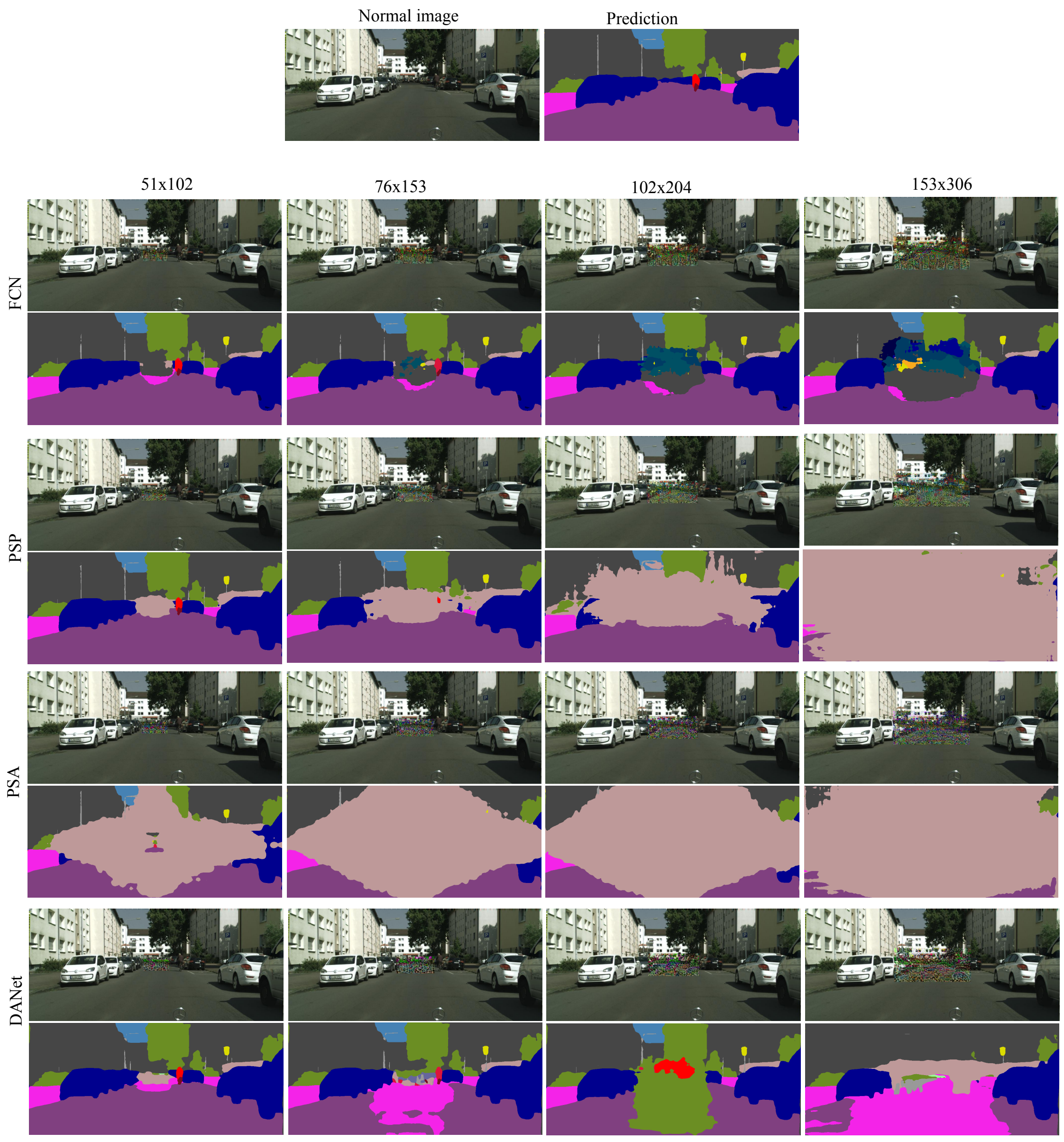


Figure 5.8 – **Universal local attacks** on segmentation networks. The degradation in FCN [163] is limited to the attacked area, whereas for context-aware networks, such as PSPNet [310], PSANet [311], DANet [71], it extends to far-away regions.

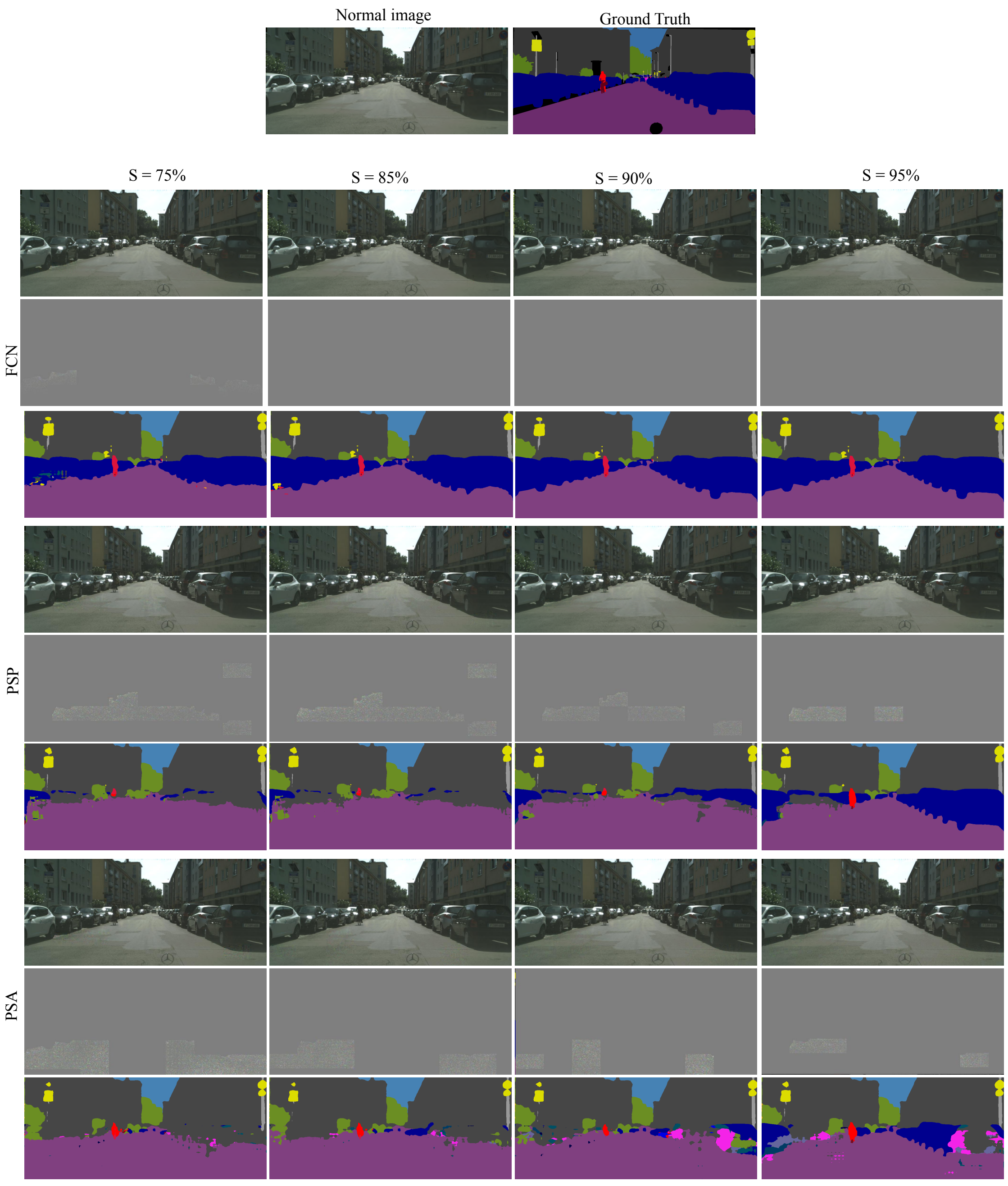
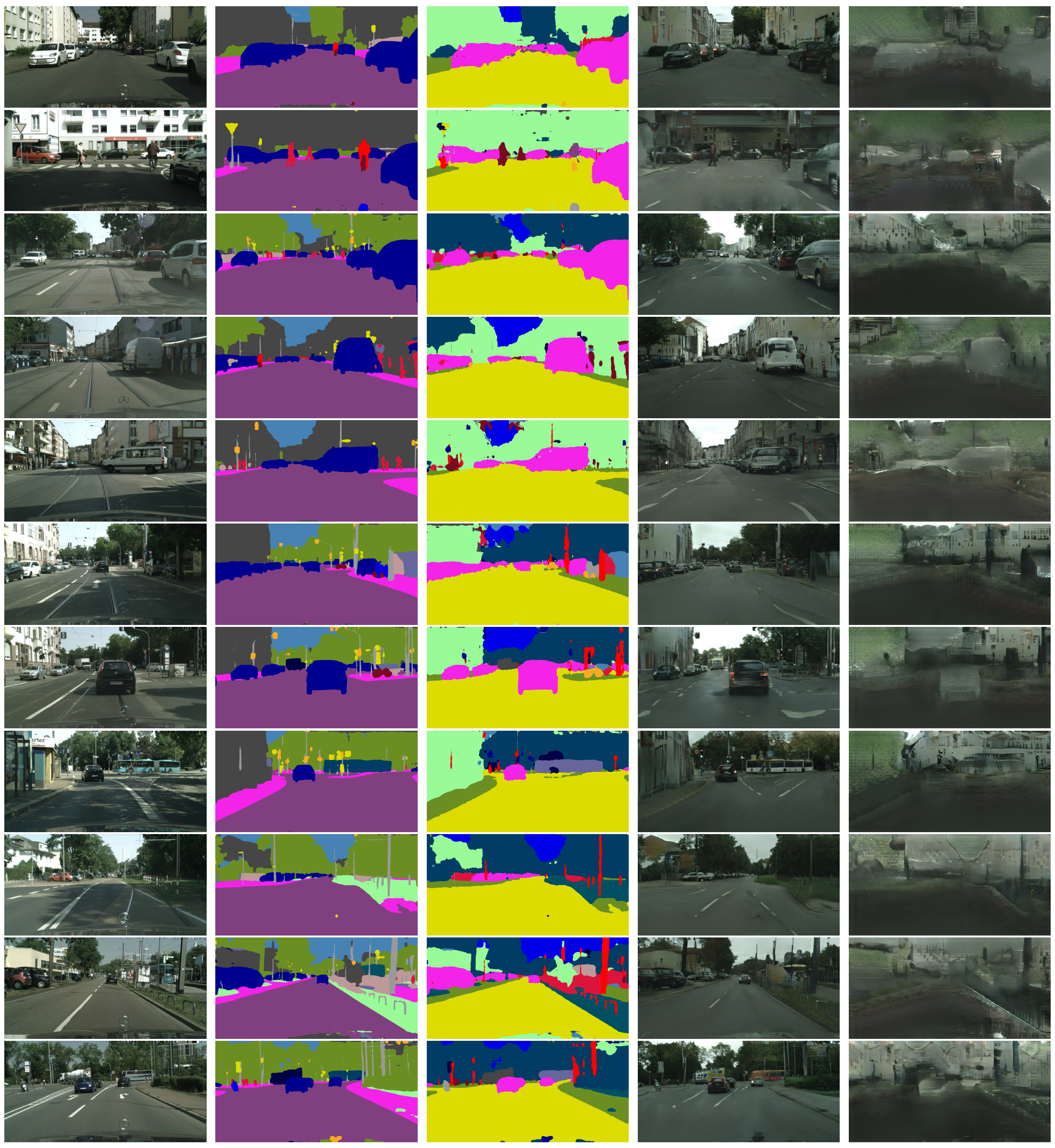


Figure 5.9 – **Adaptive indirect local attacks on Cityscapes with different networks by tuning the sparsity levels**. We observe that PSPNet [310] and PSANet [311] are vulnerable to adaptive indirect local attacks even with perturbations with high levels of sparsity, while FCN [163] is the least affected.



(a) Input image (normal) (b) Predicted map (normal) (c) Predicted map (Shift) (d) Resynthesized image (normal) (e) Resynthesized image (Shift)

Figure 5.10 – **Detecting adversarial attacks on Cityscapes with image resynthesis approach.** Without attack, the re-synthesized image **(d)** obtained from **(b)** looks similar to it. By contrast, the resynthesized image **(e)** obtained from the semantic maps **(c)** computed from a Houdini-compromised input differs massively from the original one.

5.7 Conclusion

In this chapter, we have studied the impact of indirect local image perturbations on the performance of modern semantic segmentation networks. We have observed that the state-of-the-art segmentation networks, such as PSANet and PSPNet, are more vulnerable to local perturbations because their use of context, which improves their accuracy on clean images, enables the perturbations to be propagated to distant image regions. As such, they can be attacked by perturbations that cover as little as 2.3% of the image area. We have then proposed image resynthesis-based detection and Mahalanobis distance-based detection strategy, which has proven effective for both image-level and pixel-level attack detection. In the following chapter, we continue the trend of understanding the limitations of DNNs by designing an efficient attack on the task of visual object tracking.

6 Universal, Transferable Adversarial Perturbations for Visual Object Trackers

In recent years, Siamese networks have led to great progress in visual object tracking. While these methods were shown to be vulnerable to adversarial attacks, the existing attack strategies do not truly pose great practical threats. They either are too expensive to be performed online, require computing image-dependent perturbations, lead to unrealistic trajectories, or suffer from weak transferability to other black-box trackers. In this chapter, we address the above limitations by showing the existence of a universal perturbation that is image agnostic and fools black-box trackers at virtually no cost of perturbation. Furthermore, we show that our framework can be extended to the challenging targeted attack setting that forces the tracker to follow any given trajectory by using diverse directional universal perturbations. At the core of our framework, we propose to learn to generate a single perturbation from the *object template* only, that can be added to every search image and still successfully fool the tracker for the entire video. As a consequence, the resulting generator outputs perturbations that are quasi-independent of the template, thereby making them universal perturbations. Our extensive experiments on four benchmarks datasets, i.e., OTB100, VOT2019, UAV123, and LaSOT, demonstrate that our universal transferable perturbations (computed on SiamRPN++) are highly effective when transferred to other state-of-the-art trackers, such as SiamBAN, SiamCAR, DiMP, and Ocean online.

6.1 Introduction

Visual Object Tracking (VOT) [143] is a key component of many vision-based systems, such as surveillance and autonomous driving ones. Studying the robustness of object trackers is therefore critical from a safety point of view. When using deep learning, as most modern tackers do, one particular security criterion is the robustness of the deep network to adversarial attacks, that is, small perturbations aiming to fool the prediction of the model. In recent years, the study of such adversarial attacks has become

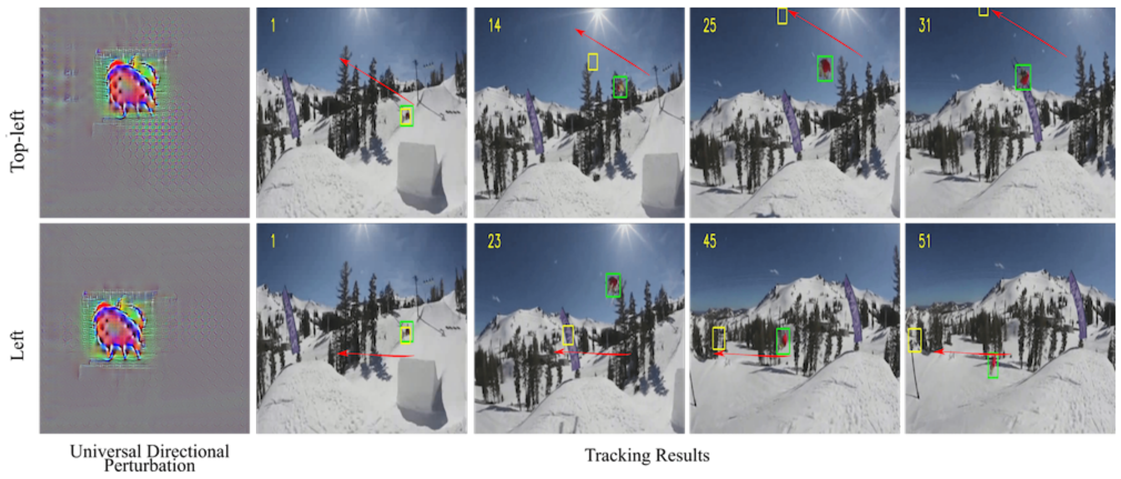


Figure 6.1 – **Universal directional pertubations.** Our approach learns an effective universal directional perturbation to attack a black-box tracker throughout the *entire* sequence by forcing it to follow a predefined motion, such as a fixed direction as illustrated above, or a more complicated trajectory, as shown in our experiments. The green box denotes the ground truth, the yellow box the output bounding box under attack, the red arrow the desired target direction.

an increasingly popular topic, extending from image classification [79, 138] to more challenging tasks, such as object detection [284] and segmentation [8, 69].

VOT is no exception to this rule, and several works [31, 84, 113, 152, 287] have designed attacks to fool the popular Siamese-based trackers [14, 144, 145, 262]. Among these, while the attacks in [31, 84, 113] are either too time-consuming or designed to work on entire videos, thus not applicable to fool a tracker in real-time and in an online fashion, the strategies of [152, 287] leverage generative methods [212, 278] to synthesize perturbations in real-time, and can thus effectively attack in an efficient manner. Despite promising results, we observed these generative strategies to suffer from three main drawbacks: (i) They require computing a search-image-dependent perturbation for each frame, which reduces the running speed of the real-time trackers by up to 40 fps, making them ineffective for practical applications such as surveillance and autonomous driving; (ii) They assume the availability of white-box trackers and yield attacks that generalize poorly when transferred to unseen, black-box trackers; (iii) They largely focus on untargeted attacks, whose goal is to make the tracker output any, unspecified, incorrect object location, which can easily be detected because the resulting tracks will typically not be consistent with the environment.

In this chapter, we argue that *learning to generate online attacks with high transferability is essential for posing practical threats to trackers and accessing their robustness.* Therefore, we propose to learn a transferable universal perturbation, i.e., a single pre-computed perturbation that can be employed to attack any given video sequence on-the-fly and generalizes to unseen black-box trackers. To achieve this, we introduce a simple yet effective framework that learns to generate a single, one-shot perturbation that is transferable across all the frames of the input video sequence. Unlike existing works [152, 287] that compute search-image-dependent perturbations for *every* search image in the video, we instead synthesize a *single* perturbation from the *template* only and add this perturbation to every subsequent search image. As a consequence of adding the same perturbation to each search image, thus remaining invariant to the search environment, the resulting framework inherently learns to generate powerful transferable perturbations capable of fooling not only every search image in the given video but also other videos and other black-box trackers. In other words, our frameworks learns to generate universal perturbations that are quasi-independent of the input template and of the tracker used to train the generator.

Moreover, in contrast to previous techniques, our approach naturally extends to performing *targeted* attacks so as to steer the tracker to follow any specified trajectory in a controlled fashion. To this end, we condition our generator on the targeted direction and train the resulting conditional generator to produce perturbations that correspond to arbitrary, diverse input directions. Therefore, at test time, we can then pre-compute directional universal perturbations for a small number of diverse directions, e.g., 12 in our experiments, and apply them in turn so as to generate the desired complex trajectory. We illustrate this in Fig. 6.1, where a single precomputed universal directional perturbation can steer the black-box tracker to move along a given direction for the entire video sequence and will show more complex arbitrary trajectories in our experiments. We will make our code publicly available upon acceptance.

Overall, our contributions can be summarized as follows:

- We introduce a transferable attack strategy to fool unseen Siamese-based trackers by generating a single universal perturbation. This is the first work that shows the existence of universal perturbations in VOT.
- Our attacking approach does not compromise the operating speed of the tracker and adds no additional computational burden.
- Our framework naturally extends to performing controllable targeted attacks, allowing us to steer the tracker to follow complex, erroneous trajectories. In practice, this would let one generate plausible incorrect tracks, making it harder to detect the attack.

We demonstrate the benefits of our approach on 4 public benchmark datasets, i.e., OTB100, VOT2018, UAV123 and LaSOT, and its transferability using several state-of-the-art trackers, such as SiamBAN, SiamCAR, DiMP, and Ocean online.

6.2 Related Work

Visual Object Tracking. VOT aims to estimate the position of a template cropped from the first frame of a video in each of the subsequent frames. Unlike most other visual recognition tasks, e.g., image classification or object detection, that rely on predefined categories, VOT seeks to generalize to any target object at inference time. As such, early works mainly focused on measuring the correlation between the template and the search image [19], extended to exploiting multi-channel information [121] and spatial constraints [45, 122].

Nowadays, VOT is commonly addressed by end-to-end learning strategies. In particular, Siamese network-based trackers [14, 144, 145, 262, 318] have grown in popularity because of their good speed-accuracy tradeoff and generalization ability. The progress in this field includes the design of a cross-correlation layer to compare template and search image features [14], the use of a region proposal network (RPN) [219] to reduce the number of correlation operations [14], the introduction of an effective sampling strategy to account for the training data imbalance [318], the use of multi-level feature aggregation and of a spatially-aware sample strategy to better exploit deeper ResNet backbones [144], and the incorporation of a segmentation training objective to improve the tracking accuracy [262]. In our experiments, we will focus on SiamRPN++ [144] as a white box model and study the transferability of our generated adversarial attacks to other modern representative trackers, namely SiamBAN [33], SiamCAR [83], DiMP [15] and Ocean-online [308].

Adversarial Attacks. Inspired by the progress of advesarial attacks in image classification [26, 52, 79, 138, 168, 182], iterative adversarial attacks have been first studied in the context of VOT. In particular, SPARK [84] computes incremental perturbations by using information from the past frames; [31] exploits the full video sequence to attack the template by solving an optimization problem relying on a dual attention loss. Recently, [112] proposed a decision-based black-box attack based on IoU overlap between the original and perturbed frames. While effective, most of the above-mentioned attacks are time-consuming, because of their use of heavy gradient computations or iterative schemes. As such, they are ill-suited to attack an online visual tracking system in real time. [270] also relies on a gradient-based scheme to generate a physical poster that will fool a tracker. While the attack is real-time, it requires to physically alter the environment.

As an efficient alternative to iterative attacks, AdvGAN [278] proposed to train a generator that synthesizes perturbations in a single forward pass. Such generative perturbations were extended to VOT in [152, 287]. For these perturbations to be effective, however, both [287] and [152] proposed to attack every individual search image, by passing it through the generator. To be precise, while [287] studied the problem of attacking the template only the success of the resulting attacks was shown to be significantly lower than that of perturbing each search image. Doing so, however, degrades the tracker running speed by up to 40 fps and generalizes poorly to unseen object environments. Here, instead, we show the existence of universal transferable perturbations, which are trained using a temporally-transferable attack strategy, yet effectively fool black-box VOT in every search image; the core success of our approach lies in the fact that the perturbation is generated from the template, agnostic to the search images but shared across all frames. This forces the generator to learn powerful transferable perturbation patterns by using minimal but key template information. Furthermore, our approach can be extended to producing targeted attacks by conditioning the generator on desired directions. In contrast to [152], which only briefly studied targeted attacks in the restricted scenario of one specific pre-defined trajectory, our approach allows us to create arbitrary, complex trajectories at test time, by parametrizing them in terms of successive universal targeted perturbations.

6.3 Methodology

Problem Definition. Let $\mathbf{X} = \{\mathbf{X}_i\}_1^T$ denote the frames of a video sequence of length T, and $\mathbf{z}$ be the template cropped from the first frame of the video, and $\mathcal{F}(\cdot)$ be the black-box tracker that aims to locate the template $\mathbf{z}$ in search regions extracted from the subsequent video frames. In this work, we aim to find a universal perturbation $\boldsymbol{\delta}$ that, when added to any search region $\mathcal{S}_i$ to obtain an adversarial image $\tilde{\mathcal{S}_i} = \mathcal{S}_i + \boldsymbol{\delta}$, leads to an incorrect target localization in frame i . Note that, unlike universal attacks on image classification [184] that aim to fool a fixed number of predefined object categories, in VOT the objects at training and testing times are non-overlapping.

6.3.1 Overall Pipeline

Figure 6.2 illustrates the overall architecture of our *training* framework, which consists of two main modules: a generator $\mathcal{G}$ and a siamese-based white-box tracker $\mathcal{F}_w$. To produce highly transferable perturbations, we propose a simple yet effective learning framework. We first train our perturbation generator to synthesize a *single* perturbation $\boldsymbol{\delta}$ from the template, and add this perturbation to every subsequent search image. As a result of adding the same perturbation, our model learns a temporally-transferable δ that can successfully attack every search image. This makes the learned δ independent of the search image, which further helps generalization to unseen object environments. In other words, by removing the dependence on the search region and relying only on the object template, our generator learns a universal adversarial function that disrupts object-specific features and outputs a perturbation pattern that is quasi-agnostic to the template. Thus, during the attack stage, we precompute a universal perturbation δ_u from any arbitrary input template and perturb the search region of any video sequence, resulting in an incorrect predicted location. Overall, our attack strategy is highly efficient and flexible, and enjoys superior transferability. Below, we introduce our loss functions

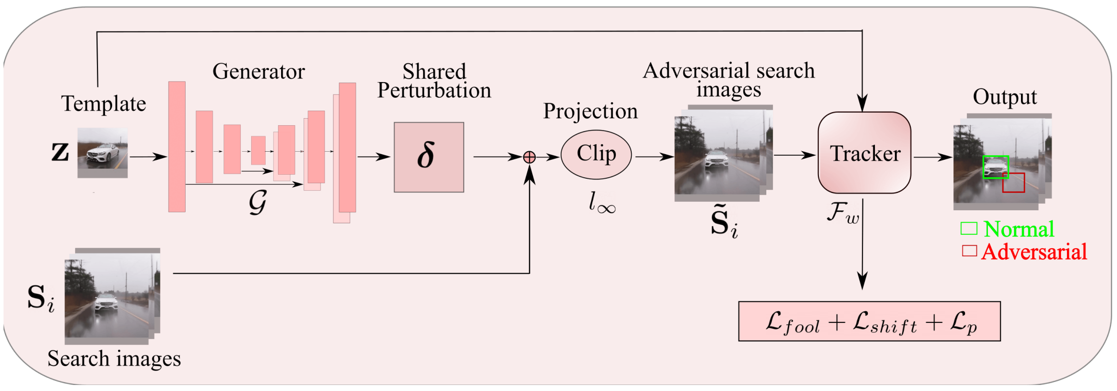


Figure 6.2 – **Our temporally-transferable attack framework.** Given the template, we generate a single temporally-transferable perturbation and add it to the search region of any subsequent frame to deviate the tracker.

in detail and then extend our framework to learning universal targeted perturbations.

6.3.2 Training the Generator

To train the generator, we extract a template $\mathbf{z}$ from the first frame of a given video sequence and feed it to the generator to obtain a unbounded perturbation $\hat{\boldsymbol{\delta}} = \mathcal{G}(\mathbf{z})$ which is clipped to be within ℓ_∞ budget to obtain the bounded perturbation $\boldsymbol{\delta}$. We then crop N search regions from the subsequent video frames using ground-truth information, and add $\boldsymbol{\delta}$ to each such regions to obtain adversarial search regions $\tilde{\mathcal{S}} = \{\tilde{\mathcal{S}}_i\}_1^N$. Finally, we feed the clean template $\mathbf{z}$ and each adversarial search region $\tilde{\mathcal{S}}_i$ to the tracker to produces an adversarial classification map $\tilde{\mathbf{H}}_i \in \mathbb{R}^{H\times W\times K}$ and regression map $\tilde{\mathbf{R}}_i \in \mathbb{R}^{H\times W\times 4K}$.

Standard Loss. Our goal is to obtain the adversarial classification $\tilde{\mathbf{H}}_i$ and regression maps $\tilde{\mathbf{R}}_i$ so as to fool the tracker, i.e., result in erroneously locating the target. To this end, we compute the classification map $\mathbf{H}_i \in \mathbb{R}^{H\times W\times K}$ for the unperturbed search image $\mathcal{S}_i$, and seek to decrease the score in $\tilde{\mathbf{H}}_i$ of any proposal j such that $\mathbf{H}_i(j) > \tau$, where $\mathbf{H}_i(j)$ indicates the probability for anchor j to correspond to the target and τ is a threshold. Following [287], we achieve this by training the perturbation generator $\mathcal{G}$ with the adversarial loss term

$$\begin{aligned}\mathcal{L}_{fool}(\mathcal{F}, \mathbf{z}, \tilde{\mathcal{S}}_i) = \lambda_1 \sum_{j|\mathbf{H}_i(j)>\tau} \max\left(\tilde{\mathbf{H}}_i(j) - (1-\tilde{\mathbf{H}}_i(j)), \mu_c\right) \\ +\lambda_2 \sum_{j|\mathbf{H}_i(j)>\tau} \left(\max\left(\tilde{\mathbf{R}}_i^w(j), \mu_w\right) + \max\left(\tilde{\mathbf{R}}_i^h(j), \mu_h\right)\right) \ ,\end{aligned} \tag{6.1}$$

where $\tilde{\mathbf{R}}_i^w(j)$ and $\tilde{\mathbf{R}}_i^h(j)$ represent the width and height regression values for anchor j. The first term in this objective aims to simultaneously decrease the target probability and increase the background probability for anchor j where the unattacked classification map contained a high target score. The margin μ_c then improves the numerical stability of this dual goal. The second term encourages the target bounding box to shrink, down to the limits μ_w and μ_h, to facilitate deviating the tracker.

Shift Loss. The loss $\mathcal{L}_{fool}$ discussed above only aims to decrease the probability of the anchors obtained from the unattacked search region. Here, we propose to complement this loss with an additional objective seeking to explicitly activate a different anchor box t, which we will show in our experiments to improve the attack effectiveness. Specifically, we aim for this additional loss to activate an anchor away from the search region center, so as to push the target outside the true search region, which ultimately will make the tracker be entirely lost. To achieve this, we seek to activate an anchor t lying at a distance d from the search region center. We then write the loss

$$\mathcal{L}_{shift}(\mathcal{F}, \mathbf{z}, \tilde{\mathcal{S}}_i) = \lambda_3 L_{cls}(\tilde{\mathbf{H}}_i(t)) + \lambda_4 L_{reg}(\tilde{\mathbf{R}}_i(\mathbf{t}), \mathbf{r}^*) \ , \tag{6.2}$$

where L_{cls} is a classification loss encoding the negative log-likelihood of predicting the target at location t, and L_{reg} computes the L_1 loss between the regression values at location t and pre-defined regression values $\mathbf{r}^* \in \mathbb{R}^4$, a vector of 4 parametrizing regression values associated with a ground truth propasal at location t.

Extension to Targeted Attacks. The untargeted shift loss discussed above aims to deviate the tracker from its original trajectory. However, it does not allow the attacker to force the tracker to follow a pre-defined trajectory. To achieve this, we modify our perturbation generator to be conditioned on the desired direction we would like the tracker to predict. In practice, we input this information to the generator as an additional channel, concatenated to the template. Specifically, we compute a binary mask $\mathbf{M}_i \in \{0, 1\}^{(W \times H)}$, and set $\mathbf{M}_i(j) = 1$ at all spatial locations under the bounding box which we aim the tracker to output. Let $\mathbf{B}_i^t$ be such a targeted bounding box, and $\mathbf{r}_i^t$ the corresponding desired offset from the nearest anchor box. We can then express a shift loss similar to the one in Eq. 6.2 but for the targeted scenario as

$$\mathcal{L}_{shift}(\mathcal{F}, \mathbf{z}, \tilde{\mathcal{S}}_i, \mathbf{M}_i) = \lambda_3 L_{cls}(\tilde{\mathbf{H}}_i(t)) + \lambda_4 L_{reg}(\tilde{\mathbf{R}}_i(t), \mathbf{r}_i^t) \ , \tag{6.3}$$

where, with a slight abuse of notation, t now encodes the targeted anchor.

Overall Loss Function.

In addition to the loss functions discussed above, we use a perceptibility loss $\mathcal{L}_p$ aiming to make the generated perturbations invisible to the naked eye. We clip the

We express this loss as

$$\mathcal{L}_p = \lambda_5 \|\mathcal{S}_i - \text{Clip}_{\{\mathcal{S}_i,\epsilon\}}\{\mathcal{S}_i + \hat{\boldsymbol{\delta}}\}\|_2^2 \,, \tag{6.4}$$

where the Clip function enforces an L_∞ bound ϵ on the perturbation. We then write the complete objective to train the generator as

$$\mathcal{L}(\mathcal{F}, \mathbf{z}, \mathcal{S}_i) = \mathcal{L}_{fool} + \mathcal{L}_{shift} + \mathcal{L}_p \,, \tag{6.5}$$

where $\mathcal{L}_{shift}$ corresponds to Eq. 6.2 in the untargeted case, and to Eq. 6.3 in the targeted one.

6.3.3 Universal Perturbations: Inference time

Once the generator is trained using the loss in Eq. 6.5, we can use it to generate a temporally-transferrable perturbation from the template $\mathbf{z}_t$ of any new test sequence, and use the resulting perturbation in an online-tracking phase at inference time. This by itself produces a common transferable perturbation for all frames of a given video, thereby drastically reducing the computational cost of perturbation compared to the image-dependendent perturbations in [152, 287]. Importantly, we observed that the trained generator learns to output a fixed perturbation pattern irrespective of the input template. This is attributed to the fact that our framework by design relies on exploiting key template information only, while being agnostic to the object's environment, thus forcing the generator to learn a universal adversarial function that disrupts the object-specific features in siamese-networks. Therefore, at inference time, we precompute a universal perturbation δ_u for an arbitrary input and apply it to any given test sequence to deviate the tracker from the trajectory predicted from unatttacked images. Furthermore, to force the tracker to follow complex target trajectories, such as following the ground-truth trajectory with an offset, we use precomputed universal directional perturbations for a small number, K, of predefined, diverse directions, with $K = 12$ in our experiments, and define the target trajectory as a sequence of these directions.

Relation to Prior Generative Attacks. Our proposed framework bears similarities with CSA in that both train a perturbation generator to fool a siamese tracker. However, our work differs from CSA in three fundamental ways. 1) In CSA, the perturbation is computed for every search image by passing it to the generator. By contrast, in our method, the perturbation depends only on the template and is shared across all search images. 2) In CSA, the attacks are limited to the untargeted setting, whereas our method extends to the targeted case and allows us to steer the tracker along any arbitrary trajectory. 3) By learning a perturbation shared across all search images, while being agnostic to them, our framework makes the perturbation more transferable than those of CSA, to the point of producing universal perturbations, as shown in our experiments.

6.4 Experiments

Datasets and Trackers. Following [287], we train our perturbation generator on GOT-10K [100] and evaluate its effectiveness on 3 short-term tracking datasets, OTB100 [274], VOT2018 [134] and UAV123 [185], and on one long-scale benchmark LaSOT [63]. We primarily use white-box SiamRPN++ (R) [144] tracker with ResNet-50 [89] backbone, and train our U-Net [224] generator. We study the transferability of attacks to 4 state-of-the-art trackers with different frameworks, namely, SiamBAN, SiamCAR, DiMP, and Ocean-online. We also transfer attacks to SiameseRPN++ (M) with MobileNet backbone, differing from the ResNet backbone of the white-box model. In contrast to SiamRPN++, which refines anchor boxes to obtain the target bounding boxes, SiamBAN and SiamCAR directly predict target bounding boxes in an anchor-free manner, avoiding careful tuning of anchor box size and aspect ratio; DiMP uses background information to learn discriminative target filters in an online fashion; Ocean-online uses a similar framework to DiMP to learn target filters in an object-aware anchor-free manner. We report the performance of our adversarial attacks using the metrics employed by each dataset to evaluate the effectiveness of unattacked trackers. In particular, we report the precision (P) and success score (S) for OTB100, UAV123, and LaSOT. For VOT2018, we report the tracker restarts (Re) and the Expected Average Overlap (EAO), a measure that considers both the accuracy (A) and robustness (R) of a tracker.

Evaluation Metrics. We report the performance of our adversarial attacks using the metrics employed by each dataset to evaluate the effectiveness of unattacked trackers. Specifically, for OTB100 and UAV123, we report the precision (P) and success score (S). The precision encodes the proportion of frames for which the center of the tracking window is within 20 pixels of the ground-truth center. The success corresponds to the proportion of frames for which the overlap between the predicted and ground-truth tracking window is greater than a given threshold. For VOT2018, we report the Expected Average Overlap (EAO), a measure that considers both the accuracy (A) and robustness (R) of a tracker. Specifically, the accuracy denotes the average overlap, and the robustness is computed from the number of tracking failures. Furthermore, we also report the number of restarts because the standard VOT evaluation protocol reinitializes the tracker once it is too far away from the ground truth. To evaluate targeted attacks, we further report the proportion of frames in which the predicted and the target trajectory center are at a distance of at most 20 pixels.

Implementation Details. We implement our approach in PyTorch [206] and perform our experiments on an NVIDIA Telsa V100 GPU with 32GB RAM. We train the generator using pre-cropped search images uniformly sampled every 10 frames from the video sequences of GOT-10K. We use the Adam [124] optimizer with a learning rate of 2×10^{-4}. We set the margin thresholds m_c, m_w, m_h to -5 as in [287], and the l_∞ bound ϵ to {8, 16}.

To fool the tracker, we use $\lambda_1 = 0.1$, $\lambda_2 = 1$, $\lambda_5 = 500$ as in csa, and activate an anchor at distance $d = 4$ directly below the true center and of size 64×64 for all untargeted experiments. Furthermore, we set the shift loss weights λ_3 and λ_4 to 0.1 and 1, respectively. For targeted attacks, we define $\mathbf{r}_i^t$ in Eq. 6.3 as a randomly-selected anchor at distance $d = 4$ from the true center and set its size to 64×64 for all datasets. We resize the search images to 255×255 and the template to 127×127 before passing them to the tracker.

Baselines. We compare our approach with the state-of-the-art, generator-based CSA [287] attack strategy, the only other online method performing untargeted attacks on RPN-based trackers.[1] Specifically, CSA can be employed in 3 settings: **CSA (T)**, which on attacks the template, **CSA (S)**, which attacks all search images, and **CSA (TS)**, which attacks both the template and all search regions. As will be shown below, CSA (T), the only version that, as us, generates a perturbation from only the template, is significantly less effective than CSA (S) and CSA (TS). These two versions, however, compute a perturbation for *each* search region, whereas our approach generates a single transferable perturbation from the template, and uses it at virtually no additional cost for the rest of the sequence, or even to attack other video sequences.

Computing a Universal Perturbation. During inference, we can use the object template cropped form any arbitrary sequence as input to the generator. For our experiments, without any loss of generality, we use the template cropped from the first video sequence of OTB100 to obtain a universal adversarial perturbation. Furthermore, for targeted attacks, we precompute $K = 12$ diverse universal directional perturbations with same template as input, and use them to force the tracker to follow any target trajectory for any input video sequence.

6.5 Results

1. How efficient is the proposed approach at attacking modern trackers? One of the primary motivation for proposing universal transferable perturbations is to not compromise the running speed of the tracker. We therefore compare the operating speed of the tracker before and after the attack in Figure 6.3. Across the board, CSA (S) and CSA (TS) decrease the tracker speed significantly on average by 35 fps. For instance, SiamCAR under CSA (TS) makes the tracker operate below real-time (30 FPS) at 15.6 fps from original 71.6 fps, thus limiting its practical applicability for surveillance. While CSA (T) operates at a speed similar to our proposed approach, with a minimal

[1] The attack strategy of [152] was tailored for fully convolutional and non-regression based trackers, such as SiamFC [14]. While the code for [152] is not public and authors did not respond to our communication, our own reimplementation showed that distance loss in [152] was only effective for pixel-level correlation networks, such as SiamFC, and not for regression-based trackers.

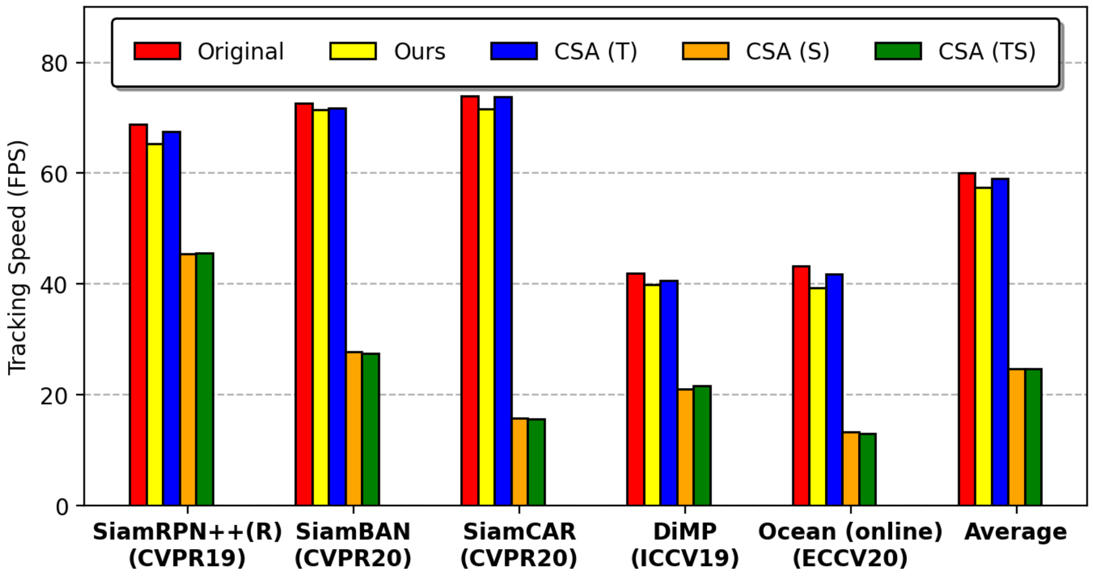


Figure 6.3 – **Change in tracker speed.** We compare the speed (FPS) of state-of-the-art trackers before and after attack.

speed degradation of about 1-5 fps, we significantly outperform it in terms of attack effectiveness as shown in the following sections.

2. How effective is the proposed approach at attacking modern trackers? Below, we evaluate the effectiveness of our proposed attack strategy. We denote the perturbation obtained with our complete loss as **Ours**, and refer to a variant of our method without the $\mathcal{L}_{shift}$ term as **Ours**$_f$. Furthermore, we denote the variant of our method that uses the template from the input video to compute a temporally transferable perturbation as **"TD"**, which has the same perturbation cost as CSA (T).

Results on Untargeted Attacks. From Tables 6.1, 6.2, 6.3, and 6.4, we can conclude that: (**1**) Our proposed approach consistently drops the performance of 5 black-box trackers in all settings. This highlights the generality of our approach in attacking black-box trackers with different frameworks. (**2**) Ours (TD), which uses the template from the video to compute a transferable perturbation for all search images, performs at a similar level to that of Ours, which uses a single universal transferable perturbation (see rows 6 vs 8). This validates that our trained generator is quasi-agnostic to the input template and enjoys the power of universality. (**3**) DiMP and Ocean, with online updates of discriminative filters to capture the appearance changes, are more robust to attacks on short-term datasets than other trackers. Interestingly, however, for a large-scale dataset such as LaSOT, the precision of Ocean-online and DiMP drops to 0.143 and 0.412 from the original

Methods	SiamRPN++ (M)		SiamBAN		SiamCAR		DiMP		Ocean online	
	S (↑)	**P** (↑)	**S** (↑)	**P** (↑)	**S** (↑)	**P** (↑)	**S** (↑)	**P** (↑)	**S** (↑)	**P** (↑)
Normal	0.657	0.862	0.692	0.910	0.696	0.908	0.650	0.847	0.669	0.884
CSA (T)	0.613	0.833	0.590	0.793	0.657	0.852	0.649	0.849	0.614	0.843
CSA (S)	0.281	0.440	0.371	0.531	0.373	0.536	0.641	0.840	0.390	0.645
CSA (TS)	0.348	0.431	0.347	0.510	0.391	0.559	0.642	0.844	0.423	0.705
Ours$_f$(TD)	0.347	0.528	0.478	0.720	0.444	0.599	0.643	0.839	0.492	0.768
Ours (TD)	**0.217**	**0.281**	**0.198**	**0.254**	**0.292**	**0.377**	**0.631**	**0.821**	**0.345**	**0.452**
Ours$_f$	0.408	0.616	0.478	0.721	0.567	0.770	0.646	0.843	0.592	0.829
Ours	**0.212**	**0.272**	**0.198**	**0.253**	**0.292**	**0.374**	0.638	0.837	**0.338**	**0.440**

Table 6.1 – Untargeted attack results on OTB100 with $\epsilon = 8$.

0.587 and 0.513, respectively. This implies that, once the tracker drifts to an incorrect position, the online updates corrupt the filters, which is especially noticeable in long video sequences. (**4**) In Table 6.4 on VOT2018, although CSA computes the perturbation from the new template when the tracker restarts after a failure, our universal perturbations significantly outperform CSA (TS), on average by ∼340 restarts. Moreover, our approach significantly decreases the EAO, which is the primary metric to rank trackers (row 4 vs 8).

Methods	SiamRPN++ (M)		SiamBAN		SiamCAR		DiMP		Ocean online	
	S (↑)	**P** (↑)	**S** (↑)	**P** (↑)	**S** (↑)	**P** (↑)	**S** (↑)	**P** (↑)	**S** (↑)	**P** (↑)
Normal	0.450	0.537	0.513	0.594	0.452	0.536	0.569	0.642	0.487	0.587
CSA (T)	0.465	0.553	0.462	0.444	0.352	0.419	0.501	0.593	0.446	0.516
CSA (S)	0.119	0.157	0.186	0.239	0.094	0.116	0.449	0.529	0.181	0.210
CSA (TS)	0.125	0.169	0.151	0.186	0.096	0.120	0.435	0.516	0.161	0.180
Ours$_f$(TD)	0.166	0.214	0.198	0.239	0.129	0.152	**0.418**	0.499	0.248	0.308
Ours (TD)	**0.114**	**0.146**	**0.095**	**0.108**	**0.079**	**0.092**	0.419	**0.487**	**0.112**	**0.128**
Ours$_f$	0.152	0.203	0.199	0.241	0.120	0.145	**0.390**	**0.461**	0.246	0.298
Ours	**0.111**	**0.146**	**0.095**	**0.109**	**0.075**	**0.089**	0.412	0.475	**0.126**	**0.143**

Table 6.2 – Untargeted attack results on LaSOT with $\epsilon = 8$.

Results on Targeted Attacks. Since manually creating intelligent target trajectories is difficult and beyond the scope of this work, we consider two simple but practical scenarios to quantitatively analyze the effectiveness of our attacks.

1. The attacker forces the tracker to follow a fixed direction. We illustrate this with 4 different directions (+45°, −45°, +135°, −135°), and aiming to shift the box by (±3, ±3) pixels in each consecutive frame.

2. The attacker seeks for the tracker to follow a more complicated trajectory. To illustrate this, we force the tracker to follow the *ground-truth* trajectory with a fixed offset (±80, ±80).

Methods	**SiamRPN++ (M)**		**SiamBAN**		**SiamCAR**		**DiMP**		**Ocean online**	
	S (↑)	**P** (↑)	**S** (↑)	**P** (↑)	**S** (↑)	**P** (↑)	**S** (↑)	**P** (↑)	**S** (↑)	**P** (↑)
Normal	0.602	0.801	0.603	0.788	0.619	0.777	0.633	0.834	0.584	0.788
CSA (T)	0.541	0.746	0.478	0.670	0.580	0.760	0.614	0.816	0.524	0.723
CSA (S)	0.288	0.466	0.299	0.485	0.270	0.440	0.593	0.798	0.264	0.489
CSA (TS)	0.270	0.452	0.278	0.487	0.271	0.428	0.598	0.811	0.278	0.510
Ours$_f$(TD)	0.369	0.561	0.372	0.569	0.337	0.503	**0.562**	**0.757**	0.404	0.648
Ours (TD)	**0.270**	**0.368**	**0.248**	**0.349**	**0.239**	**0.349**	0.573	0.770	**0.272**	**0.399**
Ours$_f$	0.356	0.549	0.372	0.569	0.316	0.469	**0.578**	**0.775**	0.392	0.634
Ours	**0.273**	**0.371**	**0.250**	**0.352**	**0.255**	**0.371**	0.579	0.777	**0.274**	**0.401**

Table 6.3 – Untargeted attack results on UAV123 with $\epsilon = 8$.

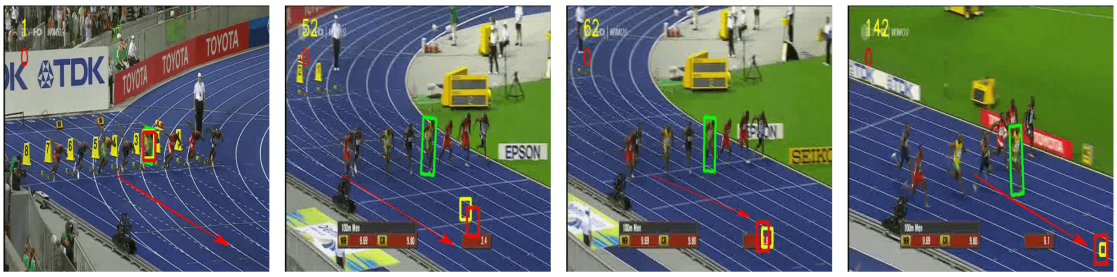
Method	**SiamRPN++(M)**				**SiamBAN**				**SiamCAR**				**DiMP**				**Ocean online**			
	A (↑)	**R** (↓)	**EAO** (↑)	**Re** (↓)	**A** (↑)	**R** (↓)	**EAO** (↑)	**Re** (↓)	**A** (↑)	**R** (↓)	**EAO**(↑)	**Re** (↓)	**A** (↑)	R (↓)	EAO(↑)	**Re** (↓)	**A** (↑)	**R** (↓)	**EAO** (↑)	**Re** (↓)
Original	0.58	0.24	0.400	51	0.60	0.320	0.340	69	0.58	0.280	0.36	60	0.607	0.3000	0.323	64	0.56	0.220	0.374	47
CSA (T)	0.56	0.440	0.265	95	0.56	0.590	0.190	126	0.56	0.426	0.280	91	0.60	0.220	0.362	47	0.45	0.580	0.189	124
CSA (S)	0.42	2.205	0.067	471	0.43	1.807	0.076	386	0.48	1.597	0.101	341	0.59	0.239	0.367	51	0.20	1.462	0.083	202
CSA (TS)	0.40	2.196	0.067	469	0.38	1.789	0.075	382	0.45	1.475	0.107	315	0.58	0.286	0.322	61	0.22	1.221	0.082	261
Ours$_f$(TD)	0.48	1.625	0.089	347	0.48	1.508	0.079	322	0.52	1.569	0.096	335	0.60	0.26	0.337	56	0.47	0.445	0.232	95
Ours (TD)	0.51	**5.095**	**0.029**	**1088**	0.44	**5.071**	**0.024**	**1083**	0.60	**3.341**	**0.053**	**712**	0.59	**0.512**	**0.219**	**109**	0.38	**1.621**	**0.074**	**346**
Ours$_f$	0.45	2.098	0.070	448	0.46	1.915	0.070	409	0.51	1.842	0.091	393	0.59	0.267	0.350	57	0.42	0.515	0.209	110
Ours	0.52	**4.856**	**0.029**	**1037**	0.44	**5.034**	**0.024**	**1075**	0.59	**3.184**	**0.056**	**680**	0.56	**0.445**	**0.235**	**95**	0.39	**1.482**	**0.081**	**316**

Table 6.4 – Untargeted attack results on VOT2018 with $\epsilon = 8$.

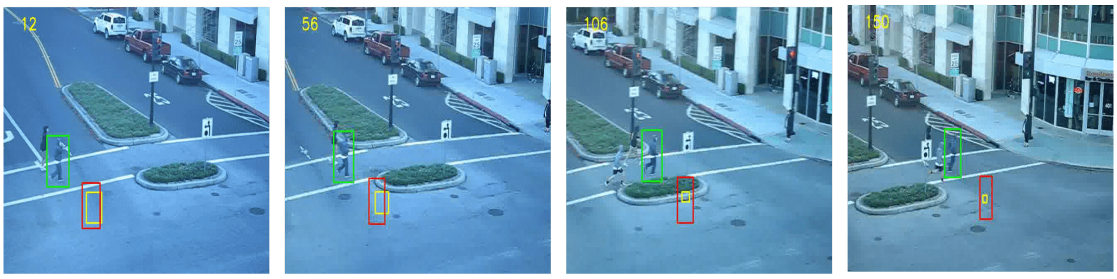
Note that our attacks are capable of steering tracker along any general trajectory, not limited to the two cases above.

In both cases, we pre-compute universal directions perturbations corresponding $K = 12$ diverse directions with template cropped from first video of OTB100, and use them to force the tracker to follow the target trajectory. To this end, we sample $K = 12$ points at a distance $d = 5$ from the object center in feature map of size 25×25 and, for each, synthesize a conditional mask $\mathbf{M}_i \in \{0, 1\}^{(W \times H)}$ whose active region is centered at the sampled point. We then feed each such mask with the template to obtain directional perturbations, which we will then transfer to the search images. During the attack for each frame, we compute the direction the tracker should move in and use the precomputed perturbation that is closest to this direction.

We report the precision score at a 20 pixel threshold for our two attack scenarios, averaged over 4 cases, in Table 6.5 for $\epsilon = 16$. For direction-based targets, our universal directional perturbations allow us to follow the target trajectory with promising performance. Our approach yields a precision of 0.627, 0.507, 0.536, and 0.335 on average on SiamRPN++(M), SiamBAN, SiamCAR and Ocean-online, respectively. For offset-based targets, which are more challenging than direction-based ones, our approach yields precision scores of 0.487, 0.350, 0.331 and 0.301 on average on the same 4 black-box trackers, respectively. Note that targeted attacks is quite challenging due to distractors and similar objects present in the search region. Nevertheless, our universal directional perturbations set a



(a) Direction-based targets



(b) Offset-based targets

Figure 6.4 – **Visualizations for targeted attacks. (a)** The tracker is forced to move in a constant direction, indicated by the red arrow. **(b)** The tracker is forced to follow the ground truth with a fixed offset of (80, 80) pixels. The green box denotes the ground truth, the yellow box the output bounding box under attack, the red arrow the desired target direction.

benchmark for image-agnostic targeted attacks on unseen black-box trackers. Figure 6.4 shows the results of targeted attacks on various datasets with SiamRPN++(M). The results at the bottom, where the tracker follows the ground-truth trajectory with an offset, illustrate the real-world applicability of our attacks, where one could force the tracker to follow a realistic, yet erroneous path. Such realistic trajectories can deceive the system without raising any suspicion.

3. What perturbation patterns does the proposed approach learn? To give insights to this question, we display the learned universal perturbations along with adversarial search regions in Figure 6.5. In the top row, we can see that, for untargeted attacks with the shift loss, our generator learns to place a universal object-like patch at the shift position. By contrast, the perturbation in CSA (S) is concentrated on the center region to decrease the confidence of the proposal. In the second row, we observe that, for targeted attacks, the perturbations are focused around the regions of the desired target box. This evidences that our conditioning scheme is able to capture the important information about the desired bounding box. Furthermore, as shown in the bottom row, our results remain imperceptible thanks to our similarity loss.

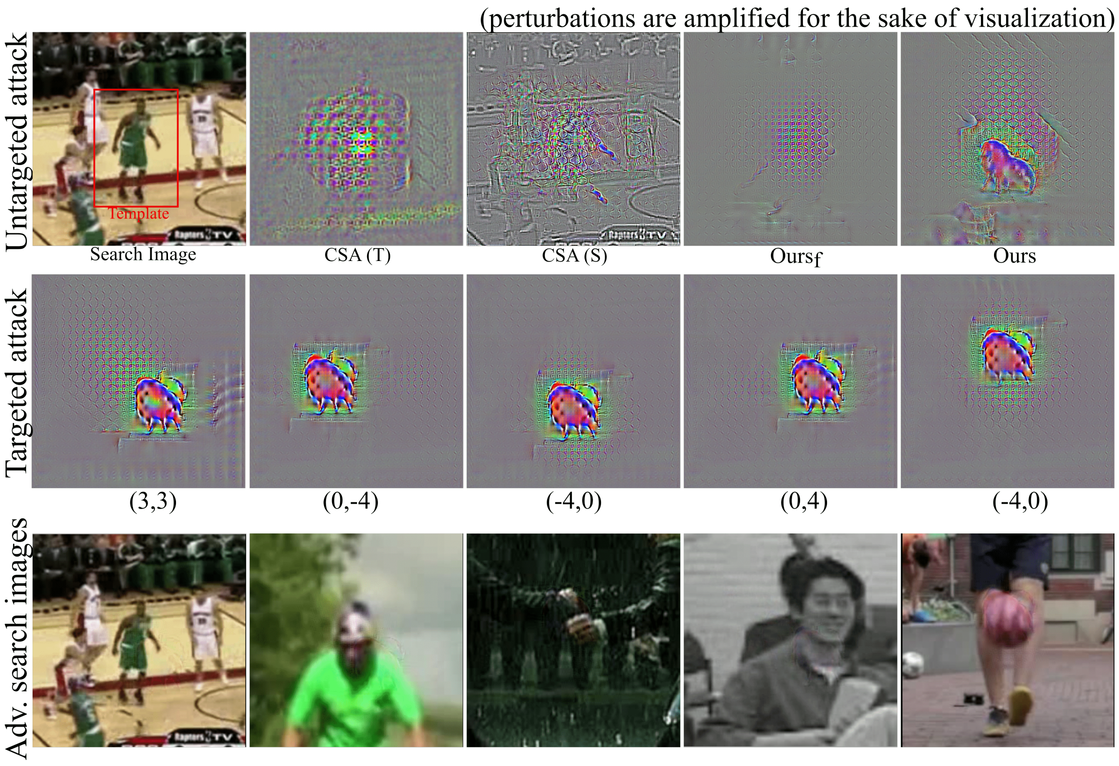


Figure 6.5 – **Qualitative Results.** We show, in the first row, the perturbations learned for untargeted attacks; in the second row, the universal directional perturbations for the targeted attack; in the last row, the adversarial search regions obtained with our targeted attack framework for $\epsilon = 16$.

Comparison to Iterative Black-box attacks. We compare our approach with the state-of-the-art black-box IoU-based attack [112] in Tables 6.8 and 6.9. This IoU attack requires access to the tracker predictions. As such, it spends a significant query budget for each frame, thereby decreasing the tracker speed to less than 5 FPS. By contrast, our method learns a highly transferable universal perturbation on a substitute SiameseRPN++(R) tracker, and thus significantly outperforms the IoU attack with virtually no drop in tracker speed. Note that we could not run the IoU attack on large-scale datasets such as UAV123 and LaSOT because of impractical runtimes.

6.6 Ablation Studies

In this section, we analyze the impact of each loss term of our framework. In Table 6.6, we report the precision scores on OTB100 with different combination of loss terms, where $\mathcal{L}_{fool}^{cls}$ and $\mathcal{L}_{fool}^{reg}$ represent the classification and regression components of the fooling loss of Eq. 6.1, and $\mathcal{L}_{shift}^{cls}$ and $\mathcal{L}_{shift}^{reg}$ represent the same terms for the shift loss of Eq. 6.2. To summarize, while all loss terms are beneficial, the classification-based terms are more effective than regression-based ones. For example, using either $\mathcal{L}_{fool}^{cls}$ or $\mathcal{L}_{shift}^{cls}$ has more

Dataset	SiamRPN++ (M)		SiamBAN		SiamCAR		Ocean online	
	Direction	Offset	Direction	Offset	Direction	Offset	Direction	Offset
OTB100	0.521	0.544	0.345	0.257	0.340	0.295	0.128	0.113
VOT2018	0.745	0.515	0.672	0.455	0.661	0.501	0.412	0.372
UAV123	0.476	0.401	0.325	0.295	0.397	0.297	0.260	0.267
LaSOT	0.768	0.489	0.689	0.395	0.747	0.232	0.543	0.521
Average	0.627	0.487	0.507	0.350	0.536	0.331	0.335	0.301

Table 6.5 – **Targeted attack results.** We report average precision scores for $\epsilon = 16$ for direction and offset-based target trajectories.

$\mathcal{L}_{fool}^{cls}$	$\mathcal{L}_{fool}^{reg}$	$\mathcal{L}_{shift}^{cls}$	$\mathcal{L}_{shift}^{reg}$	SiamRPN++ (M)	SiamBAN	SiamCAR	DiMP	Ocean online
-	-	-	-	0.862	0.910	0.908	0.847	0.884
✓	-	-	-	0.566	0.604	0.654	0.851	0.801
-	✓	-	-	0.617	0.726	0.790	0.841	0.827
-	-	✓	-	0.790	0.800	0.851	0.858	0.805
-	-	-	✓	0.858	0.890	0.884	0.858	0.879
✓	✓	-	-	0.616	0.721	0.770	0.843	0.829
-	-	✓	✓	0.695	0.735	0.734	0.828	0.750
✓	-	✓	-	0.328	0.316	0.531	0.827	0.592
-	✓	-	✓	0.682	0.769	0.826	0.848	0.852
✓	✓	✓	✓	0.272	0.252	0.374	0.837	0.440

Table 6.6 – **Component-wise analysis**. Contribution of each loss for untargeted attacks on OTB100 using our approach. We report precision score and set $\epsilon = 8$.

impact than $\mathcal{L}_{fool}^{reg}$ or $\mathcal{L}_{shift}^{reg}$. In Table 6.7, we study the impact of the shift distance d in Eq. 6.2 on the performance of untargeted attacks. For a feature map of size 25×25 for SiamRPN++, the performance of our approach is stable for a drift in the range 4 to 8. However, for $d = 2$, our attacks have less effect on the tracker, and for $d = 10$, the influence of the attack decreases because of the Gaussian prior used by the tracker.

Effect of the number K of Directional Perturbations. In the targeted attack experiments, we typically precompute $K = 12$ diverse directional perturbations. For each one, we activate a target bounding boxes at a radius 4 from the center of 25×25 feature map of SiameseRPN++ [144]. In Table 6.10, we vary this parameter K and report the precision score for offset-based targeted attacks with $\epsilon = 16$. We observe that with minimum value of $K = 4$ encoding 4 directions yields lower precision scores than $K = 12$ for both datasets. However, as K increases from 4 to 12, the precision scores increase and, beyond that, either saturate or marginally decrease.

6.6.1 Effect of Hyperparameters

Untargeted Attacks. For all experiments, we set $\lambda_1 = 0.1$ and $\lambda_2 = 1$ as in [287]. Furthermore, we set the weights of the additional terms λ_3 and λ_4 to the values of λ_1 and λ_2, respectively. This is motivated by the fact that our additional loss components

Shift d	**SiamRPN++ (M)**	**SiamBAN**	**SiamCAR**	**DiMP**	**Ocean online**
0	0.616	0.721	0.770	0.843	0.829
2	0.332	0.254	0.551	**0.814**	0.493
4	**0.272**	**0.252**	**0.374**	0.837	**0.440**
6	0.330	0.409	0.481	0.844	0.618
8	0.323	0.489	0.480	0.834	0.684
10	0.427	0.510	0.612	0.851	0.763

Table 6.7 – **Ablation study**. Effect of d in $\mathcal{L}_{shift}$ for untargeted attacks on OTB100. We report precision score and set $\epsilon = 8$.

Methods	**SiamBAN**			**SiamCAR**			**SiamRPN++(M)**		
	FPS(↑)	**S** (↑)	**P** (↑)	**FPS**(↑)	**S** (↑)	**P** (↑)	**FPS**(↑)	**S** (↑)	**P** (↑)
Normal	72.3	0.692	0.910	71.2	0.696	0.908	94	0.657	0.862
IoU	5.6	0.513	0.682	4.3	0.595	0.778	4.5	0.505	0.670
Ours	**71.7**	**0.198**	**0.253**	**70.2**	**0.292**	**0.374**	**90**	**0.212**	**0.272**

Table 6.8 – Comparison with black-box IoU attack on OTB100. We report untargeted attack results with $\epsilon = 8$.

perform similar task to those weighted by λ_1 and λ_2. We name the configuration with the above-mentioned values as the default configuration. We then independently vary each parameter while fixing the remaining ones to the default values on SiamesRPN++ [144] with $\epsilon = 8$.

As shown in Table 6.11, varying λ_1 and λ_3, encoding the classification loss terms, evidences that the default value of 0.1 performs the best for both the OTB100 [274]. Furthermore, Table 6.12 shows that varying λ_2 and λ_4, associated with the regression terms, highlights that the results are stable for values in the range of 0.001 to 1. Nevertheless, the default value of 1 for λ_4 indeed yields the best results. Besides, $\lambda_2 = 10$ performs marginally better than the default setting of $\lambda_2 = 1$.

Targeted Attacks. We perform a similar ablation study for targeted attacks on Siamese RPN++(M) [144] black-box tracker with $\epsilon = 16$. We report the results in Tables 6.13, 6.15, 6.14, and 6.16, corresponding to varying λ_1, λ_2, λ_3 and λ_4, respectively, on OTB100. In Table 6.13, we observe that setting λ_1 in the range of 0.1 to 1 yields the best results. In addition, we observe from Table 6.14 that $\lambda_3 = 1$ performs better than the default value of 0.1. Moreover, from Tables 6.15, 6.16, we find that $\lambda_2 = \lambda_4 = 10$ yields better precision scores than the default value of 0.1.

Methods	SiamBAN			SiamCAR			SiamRPN++(M)		
	FPS(↑)	**EAO** (↑)	**Re** (↓)	**FPS**(↑)	**EAO** (↑)	**Re** (↓)	**FPS**(↑)	**EAO** (↑)	**Re** (↓)
Normal	72.3	0.340	69	71.2	0.36	60	94.3	0.40	51
IoU	1.62	0.114	269	2.45	0.189	165	3.53	0.117	289
Ours	**71.7**	**0.024**	**1075**	**70.2**	**0.056**	**686**	**90.4**	**0.029**	**1037**

Table 6.9 – Comparison with black-box IoU attack on VOT2018. We report untargeted attack results with $\epsilon = 8$.

λ_4	**SiamRPN++ (M)**	**SiamBAN**	**SiamCAR**
23	0.393	0.256	0.266
12	**0.544**	**0.257**	**0.295**
8	0.308	0.181	0.186
4	0.273	0.155	0.173

Table 6.10 – **Impact of the number of directional perturbations** K for 3 black-box trackers with offset-based targeted attacks on OTB100. We report precision scores averaged over four cases for $\epsilon = 16$ and $(\Delta_x, \Delta_y) = (80, 80)$.

λ_1	**SiamRPN++(M)**		**SiamBAN**		**SiamCAR**		**Ocean-online**	
	S (↑)	P(↑)	S (↑)	P(↑)	S (↑)	P(↑)	S (↑)	P(↑)
0.001	0.536	0.711	0.562	0.762	0.618	0.817	0.592	0.783
0.01	0.352	0.472	0.322	0.421	0.491	0.641	0.503	0.672
0.1	**0.212**	**0.272**	**0.198**	**0.253**	**0.292**	**0.374**	**0.338**	**0.440**
1	0.321	0.451	0.272	0.352	0.342	0.44	0.401	0.531
10	0.352	0.501	0.363	0.501	0.413	0.572	0.521	0.721
100	0.321	0.482	0.395	0.556	0.392	0.547	0.538	0.774

(a) **Varying** λ_1

λ_3	**SiamRPN++(M)**		**SiamBAN**		**SiamCAR**		**Ocean-online**	
	S (↑)	P(↑)	S (↑)	P(↑)	S (↑)	P(↑)	S (↑)	P(↑)
0.001	0.307	0.433	0.325	0.444	0.423	0.571	0.452	0.662
0.01	0.031	0.412	0.261	0.332	0.351	0.462	0.401	0.541
0.1	**0.212**	**0.272**	**0.198**	**0.253**	**0.292**	**0.374**	**0.338**	**0.440**
1	0.471	0.642	0.38	0.491	0.562	0.742	0.502	0.641
10	0.442	0.601	0.38	0.501	0.482	0.623	0.501	0.602
100	0.471	0.632	0.443	0.579	0.543	0.718	0.488	0.638

(b) **Varying** λ_3

Table 6.11 – **Impact of varying** λ_1 **and** λ_3 (corresponding to the classification losses $\mathcal{L}^{cls}_{fool}$ and $\mathcal{L}^{cls}_{shift}$) independently for untargeted attacks on OTB100 [274]. Setting λ_1, λ_3 to 0.1 performs consistently well on all black-box trackers.

6.7 Additional Qualitative Results

We provide additional qualitative results for different attack settings. In Figure 6.6, we show the tracking outputs with offset-based targets on SiamRPN++(M) [144]. We

λ_2	SiamRPN++(M)		SiamBAN		SiamCAR		Ocean-online	
	S (↑)	P(↑)	S (↑)	P(↑)	S (↑)	P(↑)	S (↑)	P(↑)
0.001	0.317	0.411	0.218	0.268	0.440	0.568	0.460	0.606
0.01	0.378	0.497	0.255	0.322	0.427	0.552	0.464	0.604
0.1	0.371	0.482	0.254	0.324	0.431	0.554	0.451	0.589
1	**0.212**	**0.272**	**0.198**	**0.253**	**0.292**	**0.374**	**0.338**	**0.440**
10	**0.162**	**0.207**	**0.158**	**0.195**	**0.304**	**0.398**	**0.325**	**0.420**
100	0.257	0.409	0.313	0.465	0.300	0.427	0.375	0.641

(a) **Varying** λ_2

λ_4	SiamRPN++(M)		SiamBAN		SiamCAR		Ocean-online	
	S (↑)	P(↑)	S (↑)	P(↑)	S (↑)	P(↑)	S (↑)	P(↑)
0.001	0.233	0.304	0.206	0.268	0.341	0.441	0.411	0.558
0.01	0.261	0.341	0.211	0.262	0.362	0.461	0.423	0.561
0.1	0.261	0.350	0.232	0.291	0.351	0.462	0.43	0.572
1	**0.212**	**0.272**	**0.198**	**0.253**	**0.292**	**0.374**	**0.338**	**0.440**
10	0.421	0.571	0.32	0.412	0.532	0.691	0.501	0.661
100	0.452	0.626	0.363	0.472	0.514	0.678	0.482	0.658

(b) **Varying** λ_4

Table 6.12 – **Impact of varying** λ_2 **and** λ_4 (corresponding to the classification losses $\mathcal{L}_{fool}^{cls}$ and $\mathcal{L}_{shift}^{cls}$) independently for untargeted attacks on OTB100 [274]. Setting λ_2, λ_4 to 1 performs consistently well on all black-box trackers.

λ_1	SiamRPN++ (M)	SiamBAN	SiamCAR
0.001	0.233	0.141	0.096
0.01	0.282	0.185	0.122
0.1	**0.544**	0.257	0.295
1	0.478	**0.365**	**0.346**
10	0.129	0.185	0.326
100	0.007	0.007	0.017

Table 6.13 – **Impact of varying** λ_1 for offset-based targeted attacks on 3 black-box trackers on OTB100. We report precision scores for $\epsilon = 16$ and $(\Delta_x, \Delta_y) = (80, 80)$.

observe that the universal directional perturbations are concentrated along the targeted direction and typically align with the target bounding box region. As shown in Figure 6.7, our method can steer the tracker along a given direction for the entire video sequence using a single temporally transferable perturbation shown in the left column of each row. Furthermore, in Figures 6.8 and 6.9, we visualize the outputs for untargeted attacks without and with shift loss. Our results indicate that the generated perturbation concentrate around the center region and fool the tracker within the first few frames.

λ_3	**SiamRPN++ (M)**	**SiamBAN**	**SiamCAR**
0.001	0.161	0.141	0.126
0.01	0.327	0.243	0.251
0.1	**0.544**	0.257	**0.295**
1	0.446	**0.334**	0.254
10	0.313	0.288	0.282
100	0.326	0.258	0.113

Table 6.14 – **Impact of varying** λ_3 for offset-based targeted attacks on 3 black-box trackers on OTB100. We report precision scores for $\epsilon = 16$ and $(\Delta_x, \Delta_y) = (80, 80)$.

λ_2	**SiamRPN++ (M)**	**SiamBAN**	**SiamCAR**
0.001	0.398	0.294	0.240
0.01	0.380	0.283	0.208
0.1	0.394	0.311	0.270
1	**0.544**	0.257	0.295
10	0.535	**0.383**	**.485**
100	0.009	0.008	0.005

Table 6.15 – **Impact of varying** λ_2 for offset-based targeted attacks on 3 black-box trackers on OTB100. We report precision scores for $\epsilon = 16$ and $(\Delta_x, \Delta_y) = (80, 80)$.

λ_4	**SiamRPN++ (M)**	**SiamBAN**	**SiamCAR**
0.001	0.479	0.339	0.359
0.01	0.457	0.357	0.321
0.1	0.428	0.301	0.267
1	**0.544**	0.257	0.295
10	0.551	**0.343**	**0.404**
100	0.136	0.179	0.106

Table 6.16 – **Impact of varying** λ_4 for offset-based targeted attacks on 3 black-box trackers on OTB100. We report precision scores for $\epsilon = 16$ and $(\Delta_x, \Delta_y) = (80, 80)$.

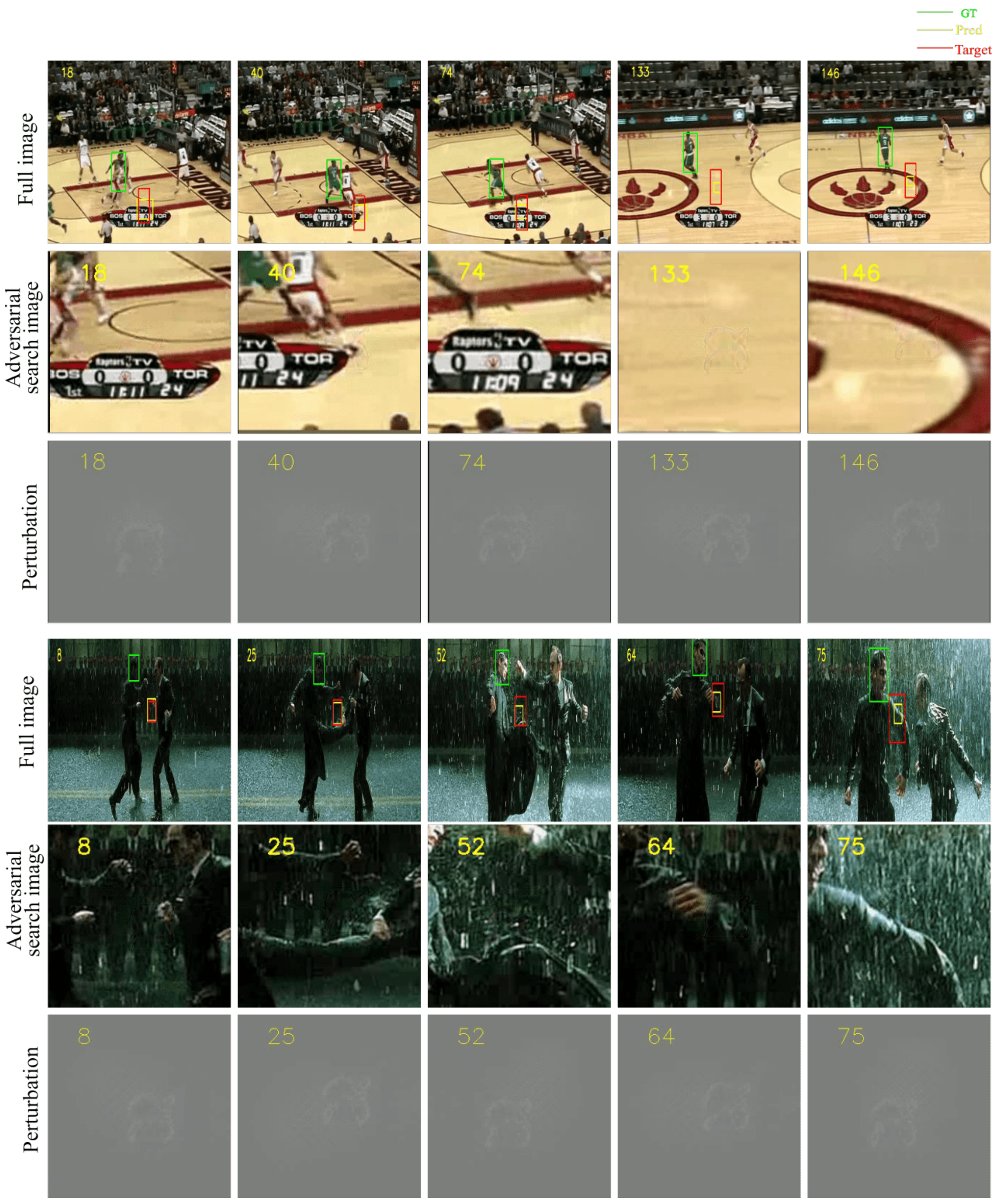


Figure 6.6 – **Qualitative results for offset-based targeted attacks** on the SiameseRPN++(M) [144] tracker to follow the ground-truth with a fixed offset of 80 pixels with $\epsilon = 16$. We visualize the tracking outputs along with the adversarial search images and the directional perturbations. Green represents the ground-truth bounding box, red represents the target bounding box, and yellow represents the predicted bounding box. We observe that the perturbation is concentrated along the direction of the target bounding box.

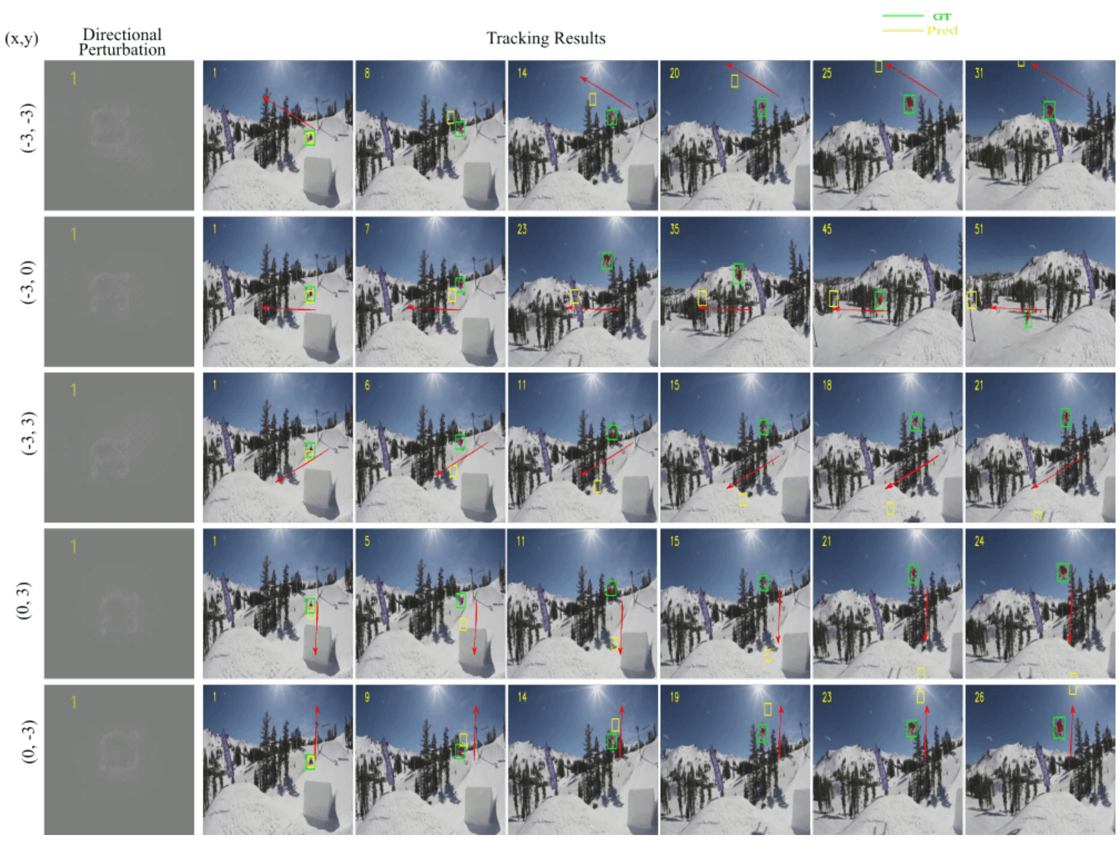


Figure 6.7 – **Qualitative results for direction-based targeted attacks** on the SiameseRPN++(M) [144] tracker to steer along a fixed direction. We visualize the tracking outputs and the temporally transferable directional perturbation computed from the object template in each row. Green represents the ground-truth bounding box, and yellow represents the predicted bounding box. We indicate the desired trajectory direction with red arrows and activate a bounding box at a distance of (x, y) from the center of the feature map. We observe that the perturbation is concentrated along the targeted direction and effective to steer the tracker for the entire video sequence.

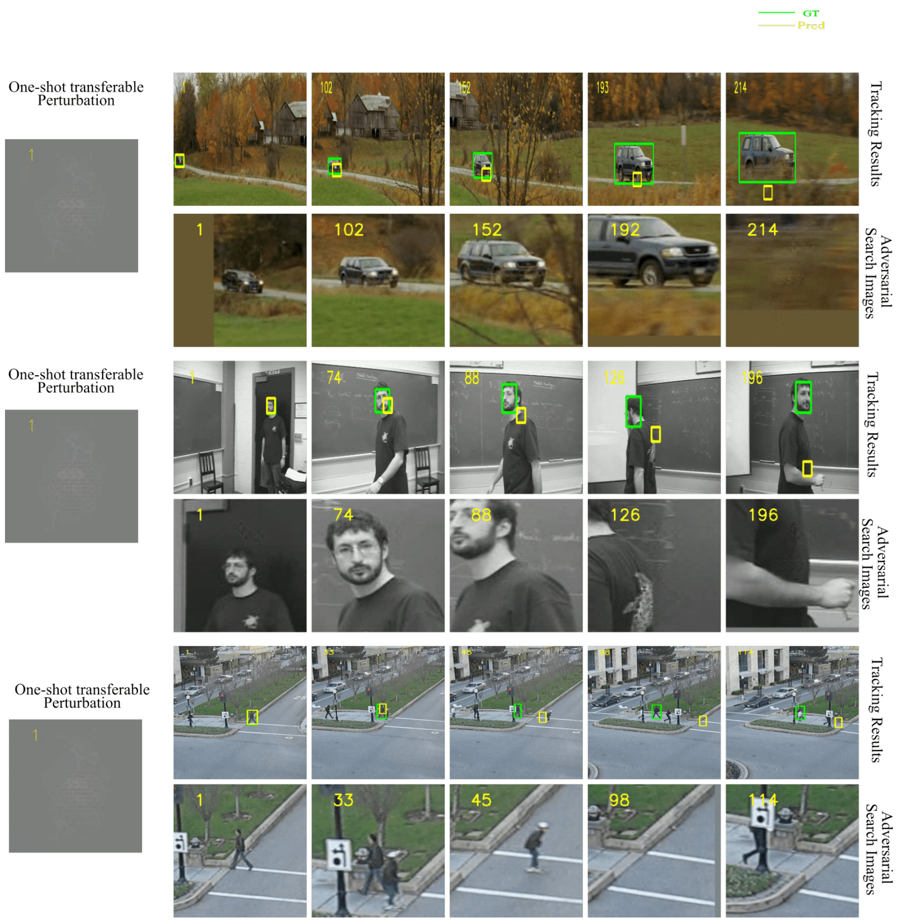


Figure 6.8 – **Qualitative results for untargeted attacks with our approach (Ours)** on the SiameseRPN++(M) [144] tracker with $\epsilon = 16$. We visualize the tracking outputs, the adversarial search regions and the single temporally transferable directional perturbation computed from the object template. Green represents the ground-truth bounding box, and yellow represents the predicted bounding box. We observe that the perturbation is concentrated around the center region and effective to fool the tracker for the entire video sequence.

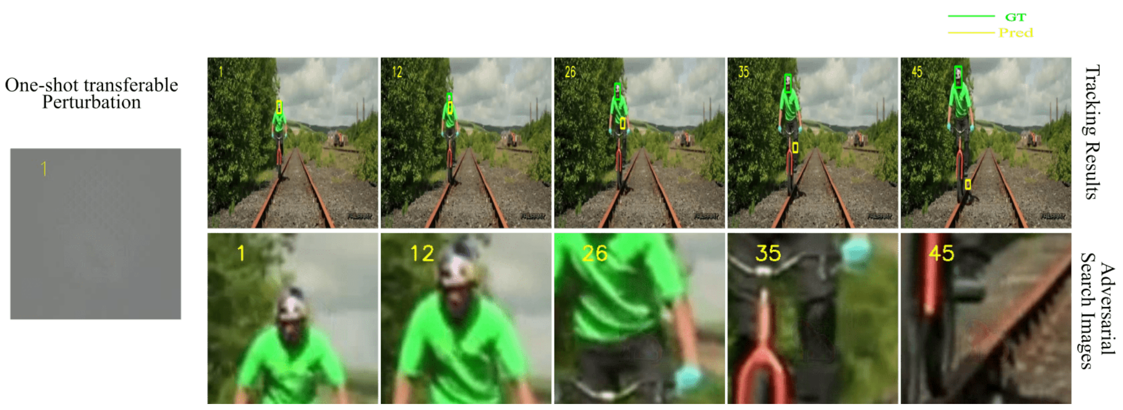


Figure 6.9 – **Qualitative results for untargeted attacks with our approach(Ours$_{shift}$)** on the SiameseRPN++ [144](M) tracker with $\epsilon = 8$. We visualize the tracking outputs, the adversarial search regions and the single temporally transferable directional perturbation computed from the object template. Green represents the ground-truth bounding box, and yellow represents the predicted bounding box. We observe that the perturbation is concentrated around the center region and effective to fool the tracker within a few frames.

6.8 Conclusion

We have shown the existence of transferable universal perturbations to efficiently attack black-box VOT trackers on the fly. To do so, we have introduced a framework that relies on generating a one-shot temporally-transferable perturbation by exploiting only the template as input, thus being invariant to the search environment. Our trained generator produces perturbations that are quasi-agnostic to the input template, and are thus highly transferable to unknown objects. Furthermore, we have demonstrated that our universal directional perturbations allow us to steer the tracker to follow any specified trajectory. In the next chapter, we conduct a systematic study to understand the degree of transferability when the attacker has limited information on either target architecture, data, or task.

7 Learning Transferable Adversarial Perturbations

The previous chapter studied adversarial attacks when the attacker has access to a substitute model on the same task. This chapter goes one step further and performs a systematic analysis when the attacker's knowledge of either target task, data, architecture, or all is restricted. In particular, recent work has shown that such attacks could be generated by another deep network, leading to significant speedups over optimization-based perturbations. However, the ability of such generative methods to generalize to different test-time situations has not been systematically studied. In this chapter, we therefore investigate the transferability of generated perturbations when the conditions at inference time differ from the training ones in terms of target architecture, target data, and target task. Specifically, we identify the mid-level features extracted by the intermediate layers of DNNs as common ground across different architectures, datasets, and tasks. This lets us introduce a loss function based on such mid-level features to learn an effective, transferable perturbation generator. Our experiments demonstrate that our approach outperforms the state-of-the-art universal and transferable attack strategies.

7.1 Introduction

In recent years, deep neural networks (DNNs) have achieved great success in a wide range of applications [111, 136, 163, 220]. However, DNNs have been demonstrated to be vulnerable to adversarial examples [246] crafted by adding imperceptible perturbations to clean images. In particular, two broad categories of attacks have been studied. The first one consists of iterative algorithms [79, 168, 246], which optimize the perturbation for each instance, and thus tend to be computationally expensive. The second one encompasses generative methods [13, 148, 212], which train a deep network to produce perturbations. As such, attacking the target network only involves a forward pass through the generator, typically resulting in much faster attacks than iterative methods. However, speed is not the only important factor to assess the strength of an attacker; its ability to generalize to different situations is also key to its success.

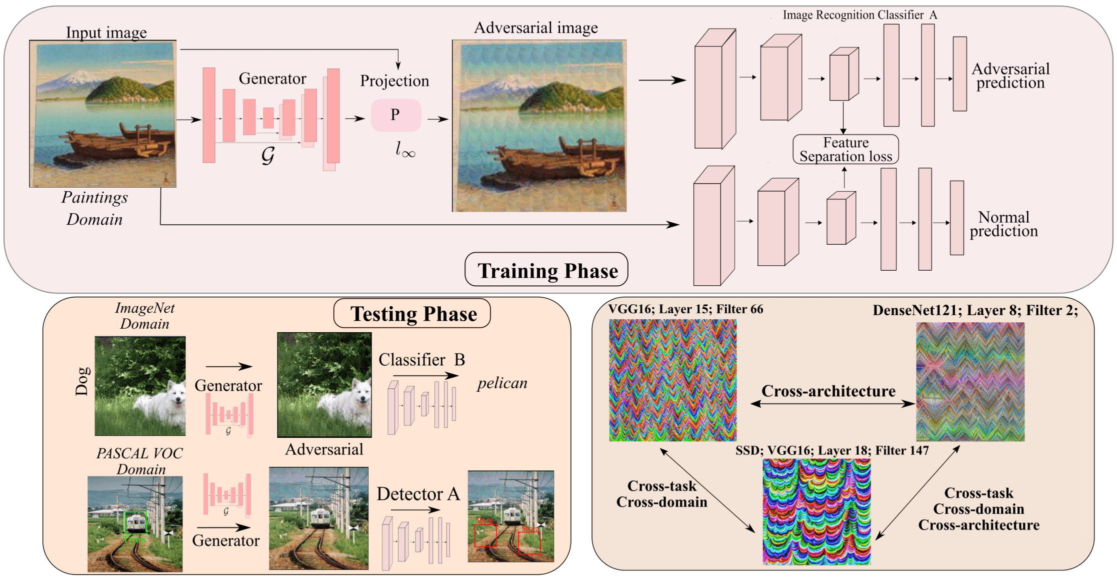


Figure 7.1 – **Learning transferable perturbations.** We observe that mid-level features are common across architectures and tasks, and thus propose to exploit them to train a perturbation generator by maximizing the relative distance between normal and perturbed features. We show that such a generator is effective even in the presence of a different model, dataset, or task at test time.

In this chapter, we therefore study the transferability of perturbations obtained with generative methods. Specifically, we investigate the transfer of such perturbations when the conditions at inference time differ from the training ones in terms of **(i)** target *architecture*, e.g., the generator was trained to attack a VGG-16 but the target network is a ResNet152; **(ii)** target *data*, e.g., the generator was trained using the Paintings dataset but the test data comes from ImageNet; **(iii)** target *task*, e.g., the generator was trained to attack an image recognition model but faces an object detector at test time. To the best of our knowledge, for generative methods, our work constitutes the first attempt at transferability across tasks, and only [191] has studied generalization across architectures and data, by introducing a loss function acting on the relative class probabilities of the attacked and unattacked examples.

Here, by contrast, we improve the transferability of a perturbation generator across architectures, data, and tasks by exploiting the mid-level features of DNNs. The key motivation behind our approach is our observation that the mid-level features extracted by DNNs with different architectures, different data, or for different tasks bear strong similarities. This is illustrated in Figure 7.1, where we visualize the features extracted by different backbones and for different tasks, and thus different datasets, using the method of [59]. Our analysis suggests that adversarial perturbations that significantly affect the mid-level features of one sample in one architecture also affect them in a different architecture, even for a different task and with different data.

We therefore propose to train a perturbation generator by maximizing the distance between the normal features of a sample and their adversarial counterparts extracted in the intermediate layers of a pretrained classifier. The resulting perturbations are then transferable across architectures, datasets and tasks because they similarly affect the mid-level features of the corresponding filters in the target setup. The perturbed mid-level features are then propagated to the top layers of the network, and, as a result, lead to incorrect predictions.

Contributions. Our contributions can be summarized as follows: 1. We identify the intermediate features of CNNs as common ground across different architectures, different data distributions and different tasks. 2. We introduce an approach that exploits such features to learn an effective, transferable perturbation generator. 3. We systematically investigate the effect of target architecture and target data distribution on the transferability of adversarial attacks. Our experiments demonstrate that our approach yields higher fooling rates than the state-of-the-art universal [148] and transferable [191] attacks. Our code is available at https://github.com/krishnakanthnakka/Transferable_Perturbations.

7.2 Related Work

Below, we review the recent works on adversarial attacks, with a focus on generator-based approaches.

Adversarial Attacks. Adversarial attacks were first investigated in [246] to identify the vulnerability of modern deep networks to imperceptible perturbations in the context of image classification. Since then, several attack strategies have been studied, including single-step fast gradient descent [79, 138], and computationally more expensive optimization-based attacks, such as CW [26], JSMA [201], and others [52, 168, 182].

While the above methods are image-dependent, the existence of Universal Adversarial Perturbation (UAP) was first shown in [184], considering the task of learning a single perturbation that can fool a classifier independently of the input image. Such a UAP was iteratively updated based on its effectiveness to move the individual data samples across the decision boundary. Other UAPs based on the computation of the singular vectors of the Jacobian matrices of feature maps have been studied in [120]. In parallel to UAPs, several works have shown that iterative adversarial attack strategies could be transferred across architectures [104, 183, 184]. In particular, the recent work of [102] proposed to improve targeted attack transferability across architectures by training auxiliary classifiers at each intermediate layer to attack a CNN's feature maps. The above-mentioned methods, however, either are computationally expensive at inference time [26, 52, 102, 104, 168, 201], or suffer from low transferability rates [183, 184]. Furthermore, most of the attacks aiming for transferability [102, 120, 184] strongly depend on the availability of data from the target domain. By contrast, we introduce an efficient generative method to produce adversarial perturbations that generalize not only across architectures, but also across training data and across tasks.

Generator-based Attacks. Generative adversarial perturbations (GAP) were first introduced in [212]. In particular, [212] showed that a generator network can be used to craft a UAP that transform an input image to an image-dependent perturbation. Similarly, [88] introduced the advGAN generative framework to learn to produce adversarial perturbations, and further proposed to make use of distillation to perform black-box attacks. Based on the observation that the effectiveness of these two methods strongly depends on the availability of data from the target domain during training, [191] introduced the relativistic cross-entropy loss, which was shown to better generalize across datasets. Furthermore, [148] proposed to generate UAPs by transforming the generator's output using a regional norm layer that enforces perturbation homogeneity. While effective, the above-mentioned works explicitly rely on the classification boundary of the attacked model, which we will show to tend to make them overfit to the source data. By contrast, we identify the mid-level features as a more robust signal shared across not only different architectures and datasets, but even across different tasks. Our experiments will showcase the superiority of exploiting this information over the losses used in previous approaches to train a transferrable perturbation generator.

7.3 Methodolgy

Let $\mathbf{x}_i \in \mathbf{R}^{H\times W\times 3}$ be a color image of size $H \times W$, and $\mathbf{y}_i$ be the associated ground-truth label. Furthermore, let f denote the task-related convolutional neural network that takes $\mathbf{x}$ as input and extracts, at layer $l = \{l_j\}_{j=1}^{L}$, a feature map $f_\ell(\mathbf{x}_i) \in \mathbf{R}^{N_\ell \times D_\ell}$, which we assume to be reshaped in matrix form, such that $N_\ell = H_\ell \times W_\ell$, with H_ℓ and W_ℓ the spatial dimensions of the feature map, and D_ℓ its number of channels.

Our goal is to train a generator $\mathcal{G}$ that produces a perturbation $\delta_i \in \mathbf{R}^{H\times W\times 3}$, which, when added to the clean image $\mathbf{x}_i$, results in predicting a different label from $\mathbf{y}_i$. To this end, we feed the input image $\mathbf{x}_i$ to the generator to synthesize an unbounded adversarial image $\mathcal{G}(\mathbf{x}_i)$, which is then clipped to be within an ϵ bound of $\mathbf{x}_i$ under the ℓ_∞ norm. Let $\hat{\mathbf{x}}_i$ be the adversarial image obtained after such a clipping of $\mathcal{G}(\mathbf{x}_i)$. In contrast to [191, 212], which rely on the final classification boundary of the task-related network f to train the generator with a cross-entropy-based loss, we exploit the mid-level features of f. Specifically, we maximize the L_2 distance between the normal feature map $f_\ell(\mathbf{x}_i)$ and adversarial feature map $f_\ell(\hat{\mathbf{x}}_i)$ at layer l using a feature separation loss term defined as

$$\mathcal{L}_{feat}(\mathbf{x}_i, \hat{\mathbf{x}}_i) = ||f_\ell(\mathbf{x}_i) - f_\ell(\hat{\mathbf{x}}_i)||_F^2 \,, \tag{7.1}$$

where $\|\cdot\|_F$ denotes the Frobenius norm.

The overall training scheme of our perturbation generator is provided in Algorithm 7.1. In practice, we observe the feature separation loss of Eq. 7.1 to encourage the generator to learn low-level, quasi-imperceptible structured patterns that affect that activations of a few filters while either suppressing the other ones or leaving them unaffected. More importantly, these perturbations are highly transferable.

Reason for transferability. Given the simplicity of our formulation, a natural question that arises is what affects the transferability of perturbations among CNNs. To understand this, we analyze the internal workings of a CNN, whose filters in each layer encode different levels of information. In particular, in Figure 7.2, we visualize the filters of various CNNs, sampled at different layers, using the technique of [59], which maximizes the mean activation of the each filter. This allows us to discover common characteristics across the different architectures. On one hand, the bottom filters, close to the input image, extract color and edge information, as shown in the left block of Figure 7.2. On the other, the top-level filters, close to the output layer, shown in the right block, are more focused on the object representation, and thus more task specific. By contrast, the mid-level filters learn more nuanced features, such as textures, and therefore tend to display similar patterns across architectures, datasets tasks. As a consequence, and as indicated by our experiments, attacking the initial layers requires large perturbation strengths. Furthermore, while attacking the top-level features works well in the white box scenario, it does not transfer well to different data or architectures, because the features are already too architecture- and task-specific. Attacking mid-level features thus comes as a natural choice, which we will show to yield highly transferable perturbations.

Let us now focus on understanding the reason for the transferability when the target data is unavailable to the attacker. In Figure 7.3, we plot t-SNE visualizations of different datasets at 4 layers of DenseNet121. The proximity of the features extracted from the Comics or Paintings datasets, containing images from a few classes particularly focused on humans, to those extracted from ImageNet suggests these datasets as good candidates to train a generator that will fool an ImageNet-trained network (as shown in Tables 7.1 and 7.2). By contrast, the ChestX dataset presents a larger domain gap. Nevertheless, we will show that, albeit for a drop in fooling rate, our use of mid-level features still make a ChestX-trained generator reasonably effective on ImageNet-based classifiers. Furthermore, in Figure 7.4, we plot the top 80 activations of ImageNet, Paintings and ChestX at layer 8 of DenseNet121. This shows that the peaks of the ImageNet dataset (orange) are more strongly correlated to those of the Paintings dataset (left) than to those of ChestX (right). This is confirmed by our experiments, where a Paintings-trained generator is more effective than a ChestX-trained one. In short, while mid-level features lead to generator transferability, the resulting effectiveness remains affected by the similarity of the data.

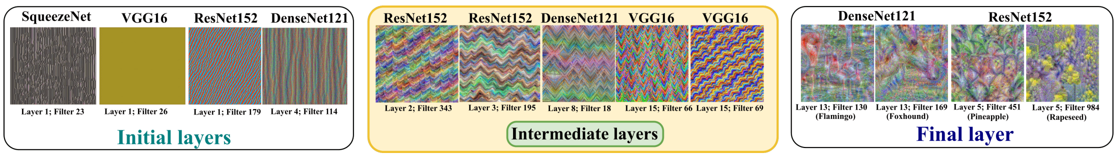
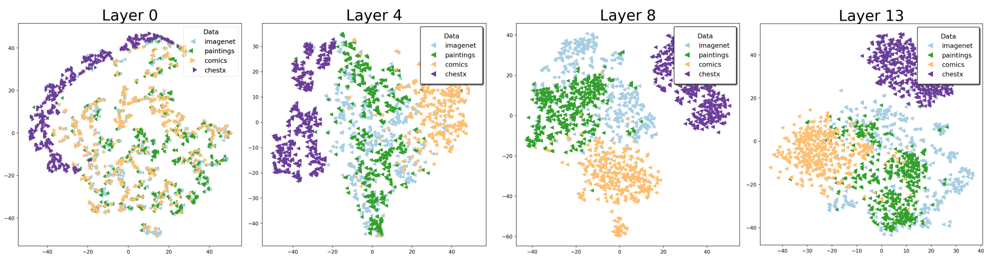


Figure 7.2 – **Filter visualization.** The filters in the mid-level layers of different CNNs follow similar activation patterns, also common across tasks, whereas those near the classification layer focus on object-level features and are thus more task-specific.



(a) Layer 0 (b) Layer 4 (c) Layer 8 (d) Layer 13

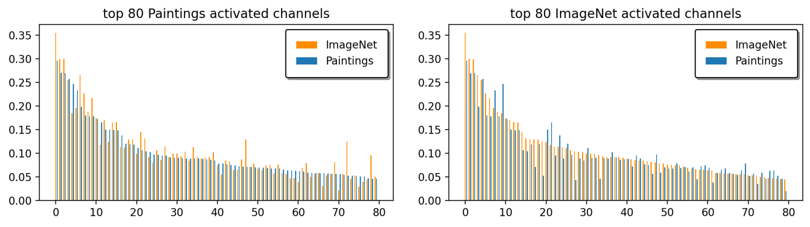
Figure 7.3 – **Feature visualization.** We visualize the features extracted at 4 different layers of DenseNet121 for different datasets using t-SNE.

7.4 Experiments

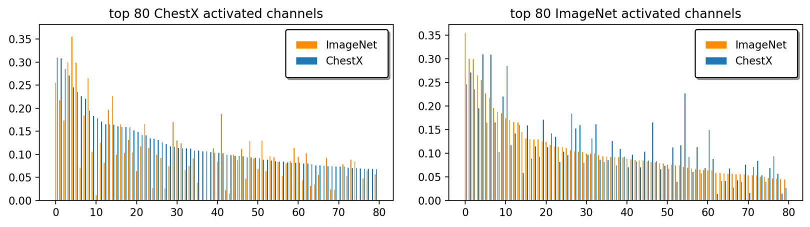
We evaluate the effectiveness of our attack strategy in diverse settings. Below, we first discuss our experimental setup and then compare our attacks with the state-of-the-art generator-based ones: CDA [191], RHP [148], and GAP [212]. To further demonstrate the generalizability of our approach, we also report its performance on adversarially-trained models and on the SSD [158] object detector with 4 different backbones.

Datasets. To train the generator, similarly to [191], we use data from either ImageNet [46], Comics [18], Paintings [1] or ChestX [217] as source domain, containing 1.2M, 40K, 80K, and 8K images, respectively. We then randomly select 5000 images from the ImageNet [46] validation set as target domain to evaluate the transferability of our attacks. Further, we use three finegrained datasets, CUB200 [259], Stanford Cars [133], and Aircraft [172] to study the extreme cross-domain transferability. As in [184, 191, 212], we report the fooling rate and the absolute difference in top-1 error between before and after the attack. The fooling rate is the percentage of images for which the label is changed after the attack.

Models. For the target models, we make use of the publicly available PyTorch [206] versions of VGG-16 [237], VGG-19 [237], ResNet152 [89], DenseNet121 [99], Inception-V3 [245] and Squeezenet [101] pretrained on ImageNet. We chose this family of networks to understand the deeper impact of transferability across diverse architectures. As in [191], to evaluate transfer across datasets, we use ChestXNet [217] pretrained on



(a) **Paintings vs. ImageNet**



(b) **Chestx vs. ImageNet**

Figure 7.4 – **Top activated channels.** Top-80 activated channels for ImageNet (orange) in comparison to Paintings in **(a)**, and to ChestX in **(b)**, for layer 8 of DenseNet121 using 500 images from each dataset. The blue bars are more correlated to the orange ones in the Paintings case, thereby achieving higher transferability rates than using ChestX. Best viewed in color and zoomed in.

Gen. Training (data)	Discriminator (ImageNet)	VGG16	ResNet152	Inception-v3	DenseNet121	SqueezeNet1.1	Average
		GAP [212] / CDA [191] / Ours					
ImageNet (1.2M)	VGG-16	99.9* / 99.8* / 99.3*	53.5 / 53.6 / **68.4**	41.7 / 43.2 / **46.6**	58.9 / 66.5 / **84.7**	67.8 / 70.6 / **86.5**	55.5 / 58.5 / **71.6**
	ResNet152	93.2 / 96.8 / **99.1**	97.6* / 99.6* / 99.7*	60.5 / 66.0 / **74.9**	87.5 / 94.2 / **98.8**	83.9 / 82.8 / **89.1**	81.3 / 84.9 / **90.5**
	Inception-v3	88.2 / 97.2 / **98.7**	83.4 / 82.7 / **90.2**	96.5* / 98.7* / 99.5*	89.5 / 93.6 / **96.0**	90.9 / 92.0 / **91.5**	88.0 / 91.4 / **94.1**
	DenseNet121	94.9 / 95.0 / **99.4**	89.5 / 91.0 / **98.7**	56.1 / 57.7 / **86.0**	99.6* / 99.6* / 99.6*	79.7 / 81.5 / **95.6**	80.1 / 81.3 / **94.9**
	SqueezeNet	88.0 / 91.5 / **96.1**	50.4 / 57.1 / **76.4**	48.0 / 47.6 / **70.7**	64.0 / 69.0 / **88.7**	99.8* / 99.7* / 99.7*	62.6 / 66.3 / **83.0**
	Average	92.8 / 96.1 / **98.6**	74.9 / 76.8 / **86.7**	60.6 / 62.6 / **75.6**	79.9 / 84.6 / **93.7**	84.4 / 85.3 / **92.5**	73.5 / 76.5 / **86.8**

Table 7.1 – **White-box and standard black-box settings.** We set $\epsilon = 10$ and report the fooling rate (in %) on 5K samples from the ImageNet val-set. The best result in each section is shown in bold. $*$ denotes the white-box setting.

ChestX. For the perturbation generators, we use the same architectures as in [191, 212]. To train them, we use the Adam optimizer with a learning rate of $2e^{-4}$ and a batch size of 16.

Attack settings. We perform attacks in four settings: **1.** White-box attacks, where the attacker has access to the exact target model and target data distribution; **2.** Standard black-box attacks, where the attacker has access to a substitute model from a different family of architectures trained on the target data and to the target data itself; **3.** Strict back-box attacks, where the attacker also uses a substitute model trained on the target data, but *without* having access to the target data itself; **4.** Extreme black-box scenario, we perform attacks without any knowledge of the target model or the target data.

Algorithm 1 Training a transferable adversarial perturbation generator

```
Input: $\mathbf{x}_i$: clean images; $f$: pretrained classifer; $\epsilon$: perturbation $\ell_\infty$ bound
Initialize generator $\mathcal{G}$; Load classifier $f$ and freeze its parameters;
repeat
    Get a clean image $\mathbf{x}_i$ and feed $\mathbf{x}_i$ to $f$ to obtain $f_\ell(\mathbf{x}_i)$;
    Generate an unbounded adversarial image $\mathcal{G}(\mathbf{x}_i)$ and clip it within an $\epsilon$ bound of $\mathbf{x}_i$ to get $\hat{\mathbf{x}_i}$;
    Pass the adversarial image to $f$ to obtain $f_\ell(\hat{\mathbf{x}_i})$;
    Compute the feature separation loss $\mathcal{L}_{feat}$;
    Compute the gradient of $\mathcal{L}_{feat}$ w.r.t. the weights of $\mathcal{G}$ and update these weights using Adam.
until convergence
return trained Generator $\mathcal{G}^*$
```

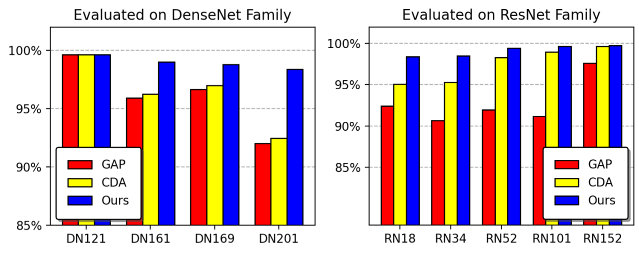
7.4.1 Transferability to Unknown Target Model

Let us first study the standard white box and black-box settings. In this experiment, we train the generator with target data containing 1.2M ImageNet training samples, using one of the 5 ImageNet pretrained models as target architecture. We then evaluate the transferability of the attack to the remaining 4 models.

Table 7.1 compares the effectiveness of our attacks to the CDA and GAP ones. We outperform these baselines on average by 10.5 and 13.5 percentage points (pp) (i.e., absolute difference of fooling rates), respectively. The differences are particularly pronounced when the generator is trained with SqueezeNet. Furthermore, in the DenseNet121 case, the difference between CDA and Ours is 13.6pp.

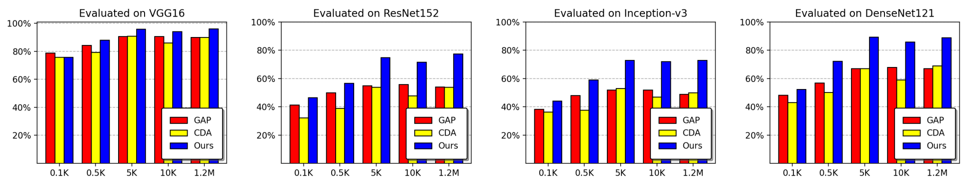
A similar trend can be seen when transferring attacks within the same family of networks. For instance, Figure 7.5 (a) shows the transferability when the generator is trained with DenseNet121 and evaluated on other networks of the same family, namely, DenseNet161, DenseNet169, and DenseNet201. Figure 7.5 (b) shows the results for similar experiments on the ResNet family. Our method performs on average 1.68pp, 6.4pp better than CDA and GAP in the case of ResNets, which demonstrates that our feature separation loss generalizes well for attacks within the same family. Importantly, the performance gap between Ours and the baselines increases with the difference in depth from training – from 0.02pp on DenseNet121 to 6.3pp on DenseNet201 w.r.t. GAP.

In Figure 7.6, we further study the transferability of attacks as the number of samples from the target data to train the generator varies. Our method (blue bar) consistently maintains higher fooling rates, even for a small number of training samples, than GAP (red bar) and CDA (yellow bar). Note that all methods tend to saturate when around 5K samples are available, with our method consistently outperforming the baselines.



(a) Trained on DenseNet121 (b) Trained on ResNet152

Figure 7.5 – **Transferability within the same family.** We report the fooling rates when transferring attacks to other networks within the same family. The generators were trained with DenseNet121 and ResNet152, with $\epsilon = 10$.



(a) VGG-16 (b) ResNet152 (c) Inceptionv3 (d) DenseNet121

Figure 7.6 – **Limited data setup.** We report the fooling rates obtained using varying amounts of ImageNet training data. The generators were trained with SqueezeNet [101] and $\epsilon = 10$, and transferred to other networks. The x-axis represents the amount of training data.

7.4.2 Transferability to Unknown Target Data

Let us now turn to the strict black-box setting, where the target data is unavailable to the attacker, who then relies on a different dataset to train the generator. Note that, here, the attacker still has access to a substitute model trained on the target data, therefore we also study the impact of transferability to an unknown target model. Following [191], we consider two synthetic datasets, Comics and Paintings, depicting objects similar to the ImageNet ones, and one dataset, ChestX, containing a completely different type of data, thus suffering from a large domain gap.

Table 7.2 compares the results of our approach with those of the baselines. Our method yields clearly superior results for all 3 datasets. For instance, with Comics, Ours outperforms GAP and CDA on average by 14.9pp and 10.3pp, respectively, and with Paintings, by 14.9pp and 11.0pp, respectively. Note that, while we still outperform the baselines when using the ChestX data, by around 22pp on average, the fooling rates of all methods drop significantly in this challenging scenario. Nevertheless, altogether, these results evidence that one can learn to generate adversarial examples with high fooling rates despite not having access to the target data, even by using data from a completely different domain. Furthermore, the resulting generator generalizes particularly well to the unseen target domain when trained with our feature separation loss.

Gen. Training (data)	**Discriminator** (ImageNet)	**VGG16**	**ResNet152**	**Inception-v3**	**DenseNet121**	**SqueezeNet 1.1**	**Average**
		GAP [212] / CDA [191] / Ours					
Comics (40K)	VGG-16	99.8 / **99.9** / 99.5	54.3 / 54.0 / **77.4**	45.8 / 45.2 / **61.9**	66.3 / 64.2 / **93.6**	70.7 / 68.4 / **93.4**	67.4 / 66.3 / **85.1**
	ResNet152	75.3 / 95.8 / **99.3**	97.6 / 98.1 / **99.6**	31.7 / 66.5 / **73.1**	45.1 / 87.7 / **98.6**	67.3 / 86.0 / **90.7**	63.4 / 86.8 / **92.3**
	Inception V3	84.3 / 85.6 / **99.0**	97.2 / 97.3 / **90.4**	99.8 / **99.8** / 99.6	88.5 / 88.0 / **96.7**	82.4 / 82.3 / **93.2**	90.5 / 90.6 / **95.8**
	DenseNet121	96.9 / 92.0 / **96.5**	**98.0** / 86.3 / 93.0	**83.1** / 65.7 / 82.5	**99.4** / 98.4 / 98.8	78.3 / 75.7 / **91.9**	91.2 / 83.6 / **92.5**
	SqueezeNet	87.7 / 89.9 / **96.5**	54.0 / 58.2 / **79.0**	51.2 / 51.4 / **75.4**	68.7 / 76.3 / **90.2**	99.7 / **99.8** / 99.7	72.3 / 75.1 / **88.2**
	Average	88.8 / 92.6 / **98.2**	80.2 / 72.8 / **87.8**	62.3 / 65.8 / **78.5**	73.6 / 82.9 / **95.6**	79.7 / 82.5 / 93.8	76.9 / 80.5 / **90.8**
Paintings (80K)	VGG-16	99.4 / **99.9** / 99.0	41.1 / 57.6 / **66.6**	36.5 / 46.6 / **50.0**	50.8 / 73.8 / **84.6**	63.7 / 73.0 / **86.4**	58.3 / 70.1 / **77.3**
	ResNet152	80.4 / 89.9 / **98.7**	95.4 / 97.5 / **99.4**	50.7 / 62.1 / **72.8**	70.4 / 82.3 / **97.9**	70.4 / 81.1 / **89.2**	73.5 / 82.6 / **91.6**
	Inception V3	80.3 / 80.5 / **98.6**	95.8 / **96.4** / 88.2	99.6 / **99.6** / 99.5	87.7 / 87.2 / **95.2**	77.5 / 72.8 / **90.8**	88.2 / 87.3 / **94.5**
	DenseNet121	87.6 / 86.5 / **96.2**	80.1 / 81.2 / **90.9**	51.4 / 50.4 / **76.0**	98.8 / **98.9** / 97.4	73.6 / 73.7 / **91.7**	78.3 / 78.1 / **90.5**
	SqueezeNet	82.8 / 80.7 / **95.2**	46.0 / 46.0 / **73.4**	44.5 / 47.4 / **71.0**	59.3 / 56.5 / **87.2**	99.4 / 99.3 / **99.6**	66.4 / 66.0 / **85.3**
	Average	86.1 / 87.5 / **97.6**	71.7 / 75.8 / **83.7**	56.5 / 61.2 / **73.9**	73.4 / 79.7 / **92.5**	76.9 / 80.0 / 91.5	72.9 / 76.8 / **87.8**
ChestX (10K)	VGG-16	78.7 / 85.6 / **93.3**	23.2 / 23.3 / **41.8**	25.5 / 27.9 / **31.3**	27.5 / 28.2 / **53.4**	46.1 / 48.0 / **64.3**	40.2 / 42.6 / **56.8**
	ResNet152	39.9 / 44.8 / **56.4**	27.0 / 25.3 / **62.8**	28.2 / 25.7 / **27.7**	25.9 / 26.6 / **38.1**	44.9 / 47.1 / **60.5**	33.2 / 33.9 / **49.2**
	Inception V3	56.0 / 50.3 / **91.6**	35.9 / 32.0 / **69.5**	44.4 / 35.1 / **84.9**	45.9 / 35.4 / **77.4**	65.1 / 57.7 / **75.6**	49.5 / 42.1 / **79.8**
	DenseNet121	42.8 / 42.3 / **64.0**	26.4 / 25.2 / **44.2**	28.0 / 28.8 / **34.0**	41.9 / 48.2 / **76.0**	54.2 / 48.8 / **60.2**	38.7 / 38.7 / **55.7**
	SqueezeNet	51.7 / 51.1 / **81.1**	27.9 / 31.6 / **52.5**	30.2 / 33.1 / **47.1**	31.6 / 35.1 / **64.2**	81.3 / 78.9 / **96.4**	44.5 / 46.0 / **68.3**
	Average	53.8 / 54.8 / **77.2**	28.1 / 27.4 / **54.2**	31.3 / 30.1 / **45.0**	34.6 / 34.7 / **61.9**	58.3 / 56.1 / **71.4**	41.2 / 40.6 / **62.0**

Table 7.2 – **Transferability in the strict black-box setting with no access to target data.** We set $\epsilon = 10$ and report the fooling rate for 5K samples from the ImageNet validation set.

Discriminator (Model)	**Gen. Training** (data)	**VGG16**	**ResNet152**	**Inception-v3**	**DenseNet121**	**SqueezeNet1.1**	**Average**
		GAP [212] / CDA [191] / Ours					
ChestXNet (trained on ChestX)	Comics40K	65.4 / 58.2 / **77.3**	50.5 / 38.9 / **57.4**	55.4 / 46.5 / **60.2**	61.4 / 48.1 / **76.9**	71.6 / 68.0 / **79.7**	60.9 / 51.9 / **70.3**
	Comics10K	66.8 / 58.1 / **77.0**	50.3 / 38.8 / **55.2**	55.0 / 45.7 / **59.1**	62.2 / 49.1 / **74.9**	70.1 / 68.8 / **78.9**	60.9 / 52.1 / **69.0**
	Paintings80K	76.6 / 63.2 / **78.2**	53.8 / 46.2 / **56.8**	56.8 / 52.9 / **60.2**	73.7 / 55.3 / **77.7**	79.3 / 70.2 / **80.5**	68.0 / 57.5 / **70.7**
	Paintings10K	76.7 / 61.3 / **78.4**	52.9 / 41.7 / **59.9**	57.4 / 49.5 / **61.7**	74.6 / 50.5 / **79.1**	78.6 / 67.4 / **81.1**	68.0 / 54.1 / **71.9**
	ChestX10k	43.3 / 48.2 / **66.3**	29.2 / 23.7 / **49.5**	24.9 / 26.5 / **48.5**	35.3 / 28.8 / **65.0**	52.8 / 45.5 / **72.5**	37.1 / 34.5 / **60.4**
	Average	65.7 / 57.8 / **75.4**	47.3 / 37.8 / **55.8**	49.9 / 44.2 / **57.8**	61.4 / 46.4 / **74.7**	70.5 / 64.0 / **78.5**	59.0 / 50.0 / **68.5**

Table 7.3 – **Extreme cross-domain transferability analysis using a generator trained with ChestXNet and different datasets.** We set $\epsilon = 10$ and report the fooling rates on 5K ImageNet val-set samples.

7.4.3 Extreme Cross-Domain Transferability

We further study the transferability of adversarial attacks in the extreme scenario where neither the target architecture nor the target data are available, and where even the classifier used to learn the generator was trained on data that differs from the target domains. To this end, we use ChestXNet [217] pretrained on ChestX dataset as classifier. We train the generator with different datasets on denseblock12 and evaluate its transferability in Table 7.3. Despite the challenging nature of this setting, our method still yields satisfying fooling rate; 68.5pp on average vs 87.0pp in the standard black-box scenario. Furthermore, our approach outperforms the state-of-the-art CDA by 18.5% on average.

Furthermore, we conduct the transferability study from ImageNet trained generators on three fine-grained datasets namely CUB200, Stanford Cars, and Aircraft with backbones different from training ones. As observed from Table 7.4, our method outperforms CDA by 21pp, 21pp, and 30pp on CUB200, Stanford Cars, and Aircraft, respectively. Moreover, the generator is trained on backbones with simple cross-entropy losses, however, the target

Gen. Training (data)	Discriminator (ImageNet)	ResNet50	SeNEt154	SeResNet101	Average
		GAP [212] / CDA [191] / Ours			
ImageNet (1.2M)	VGG-16	41.25 / 24.59 / **76.15**	41.44 / 30.43 / **45.82**	29.75 / 23.01 / **35.85**	37.48 / 26.01 / **52.61**
	ResNet152	54.82 / 52.78 / **93.18**	50.76 / 50.72 / **77.44**	46.00 / 45.13 / **65.00**	50.35 / 49.54 / **78.54**
	Inception-v3	40.78 / 55.63 / **70.40**	33.07 / 36.49 / **48.10**	35.12 / 36.59 / **39.52**	36.32 / 42.90 / **52.67**
	DenseNet121	52.95 / 50.97 / **90.66**	38.52 / 43.42 / **73.30**	45.36 / 46.10 / **63.07**	45.61 / 46.83/ **75.68**
	SqueezeNet	36.40 / 35.57 / **63.89**	34.04 / 25.55 / **47.32**	34.57 / 30.51 / **39.39**	35.00 / 30.54 / **50.20**
	Average	45.13 / 43.91 / **78.86**	39.57 / 37.32 / **58.40**	38.16 / 36.27 / **48.57**	40.95 / 39.17 / **61.94**

(a) CUB200

Gen. Training (data)	Discriminator (ImageNet)	ResNet50	SeNEt154	SeResNet101	Average
		GAP [212] / CDA [191] / Ours			
ImageNet (1.2M)	VGG-16	18.07 /48.65 / **70.22**	32.35 / 30.03/ **32.41**	12.66 / 14.76 / **21.73**	21.03 / 31.15 / **41.45**
	ResNet152	37.08 / 71.27 / **94.80**	33.25 / 34.31 / **62.74**	22.73 / 31.51 / **62.23**	31.02 / 45.70 / **73.26**
	Inception-v3	51.27 / 44.12 / **44.34**	35.63 / 36.25 / **38.59**	31.68 / 25.43 / **25.83**	39.53 / 35.27 / **36.25**
	DenseNet121	59.84 / 57.46 / **98.32**	28.98 / 34.09 / **65.27**	24.71 / 25.43 / **71.76**	37.84 / 38.97 / **78.45**
	SqueezeNet	26.07 / 30.32 / **85.33**	17.09 / 16.06 / **31.69**	14.40 / 18.19 / **31.54**	19.19 / 21.52 / **49.52**
	Average	38.47 / 50.36 / **78.60**	29.46 / 30.15 / **46.14**	21.24 / 23.05 / **42.62**	29.72 / 34.52 / **55.79**

(b) Stanford Cars

Gen. Training (data)	Discriminator (ImageNet)	ResNet50	SeNEt154	SeResNet101	Average
		GAP [212] / CDA [191] / Ours			
ImageNet (1.2M)	VGG-16	25.20 / 23.97 / **79.36**	46.77 / 38.79 / **37.28**	36.15 / 27.42 / **38.16**	36.04 / 30.06 / **51.60**
	ResNet152	42.87 / 64.45 / **96.82**	49.02 / 53.35 / **91.63**	36.72 / 56.80 / **86.44**	42.87 / 58.20 / **91.63**
	Inception-v3	49.38 / 43.95 / **72.61**	54.25 / 35.25 / **59.41**	46.28 / 43.11 / **42.87**	49.97 / 40.77 / **58.30**
	DenseNet121	37.11 / 37.05 / **93.10**	38.73 / 41.04 / **88.30**	35.22 / 36.93 / **83.59**	37.02 / 38.34 / **88.33**
	SqueezeNet	26.07 /33.63 / **82.30**	27.18 / 27.57 / **41.70**	38.40 / 42.78 / **52.51**	30.55 / 34.66 / **58.84**
	Average	36.13 / 40.61 / **84.84**	43.19 / 39.20 / **63.66**	38.55 / 41.41 / **60.71**	39.29 / 40.41 / **69.74**

(c) Aircraft

Table 7.4 – **Extreme cross-domain transferability from ImageNet to three fine grained datasets.** We set $\epsilon = 10$ and report the fooling rate (in %) on 5K samples from the ImageNet val-set. The best result in each section is shown in bold.

employs auxiliary losses to destruct and construct [32] the discriminative regions with novel losses. Despite that, the ImageNet trained generator enjoys good transferability of 60% fooling rate on average.

7.4.4 Transferability to Robust Models

In this section, we study the transferability of our generators to five state-of-the-art defenses, i.e., the high-level representation guided denoiser (HGD) [153], the input pre-processing defense through random resizing and padding (R&P) [283], Feature Denoising (FD) on ResNeXt-101 [285], Projected Gradient Descent (PGD) [168] on ResNet50, and the average of three ensembles of adversarially trained Inception models [254] (EnsembleAdv). In addition to GAP and CDA, we compare our method to RHP, which constitute the state-of-the-art universal attack in terms of effectiveness against defenses. As shown in Table 7.5, Ours achieves the overall best performance on HGD, R&P and EnsembleAdv. However, we observe that none of the methods are successful in attacking Feature denoising and PGD defenses. Note, however, that with PGD and FD, the error rate on the clean samples is significantly higher than with other defenses, making these strategies ill-suited for practical applications.

Nework	**Method**	**EnsembleAdv [254]**	**HGD [153]**	**RP [283]**	**FD [285]**	**PGD [168]**
Error on normal samples		24.6	19.5	21.0	51.1	53.7
Inc-v3	RHP	29.5	26.8	23.3	2.38	2.40
Inc-v4	RHP	25.1	23.4	20.2	1.90	2.20
IncRes-v2	RHP	28.8	26.9	25.1	2.20	2.20
SqueezeNet	GAP	27.2	33.1	26.7	4.24	**5.84**
SqueezeNet	CDA	25.6	31.4	26.5	4.06	5.42
SqueezeNet	Ours	**33.7**	**43.9**	**32.7**	**7.06**	3.30

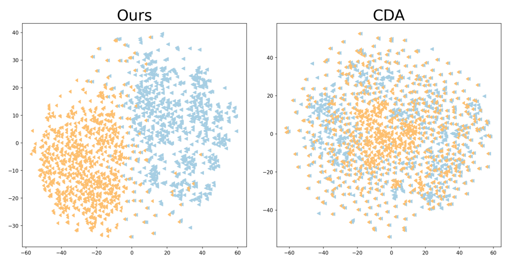
Table 7.5 – **Transferability to adversarially trained models**. We report the absolute percentage increase in top-1 error with $\epsilon = 16$ on the 5K samples from ImageNet val-set following [148].

7.4.5 Cross-Task Transferability Analysis

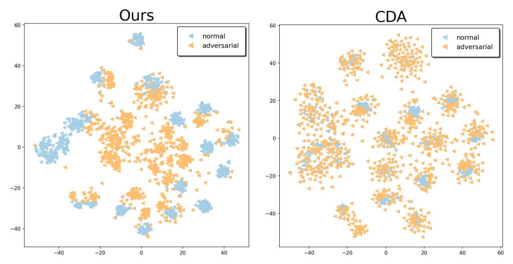
Gen. Training (data)	**Discriminator** (Trained on ImageNet)	**VGG16**	**ResNet50**	**EfficientNet**	**MobileNet-v3**	**Average**
		GAP [212] / CDA [191] / Ours				
-	No Attack	68.12	66.08	61.07	55.44	62.68
Comics (40K)	VGG-16	14.9 / 22.8 / **3.60**	14.2 / 20.7 / **7.55**	11.3 / 12.6 / **8.00**	07.5 / 11.4 / **4.30**	12.0 / 16.9 / **5.86**
	ResNet152	32.1 / 24.0 / **9.08**	25.5 / 16.9 / **8.13**	30.6 / 15.7 / **7.66**	21.4 / 13.0 / **5.44**	27.4 / 17.4 / **7.58**
	Inception-v3	33.7 / 32.7 / **18.0**	29.0 / 28.5 / **20.2**	32.3 / 31.3 / **14.5**	22.7 / 22.1 / **14.0**	29.4 / 28.6 / **16.6**
	DenseNet121	25.1 / 23.2 / **8.08**	21.8 / 19.8 / **9.28**	22.4 / 20.7 / **12.9**	16.8 / 15.3 / **5.60**	21.5 / 19.8 / **8.96**
	SqueezeNet	28.5 / 26.8 / **13.7**	23.1 / 24.4 / **11.3**	17.8 / 20.8 / **12.8**	14.2 / 18.3 / **6.53**	20.9 / 22.6 / **11.1**
	Average	26.8 / 25.9 / **10.5**	22.7 / 22.1 / **11.3**	22.9 / 20.2 / **11.2**	16.5 / 16.0 / **7.17**	22.2 / 21.1 / **10.0**
Paintings (80K)	VGG-16	16.8 / 16.0 / **8.05**	18.2 / 17.4 / **12.1**	11.6 / **10.3** / 11.6	**7.86** / 7.99 / 8.47	13.6 / 12.9 / **10.1**
	ResNet152	31.0 / 19.6 / **9.66**	20.5 / 17.1 / **9.31**	17.8 / 12.2 / **7.74**	15.1 / 12.1 / **5.45**	21.1 / 15.3 / **8.04**
	Inception-v3	32.9 / 33.0 / **16.0**	28.4 / 28.7 / **18.9**	27.4 / 27.9 / **14.2**	22.6 / 21.4 / **12.1**	27.8 / 27.8 / **15.3**
	DenseNet121	28.5 / 29.1 / **8.14**	20.2 / 20.6 / **8.99**	19.3 / 20.2 / **7.90**	15.4 / 16.0 / **5.98**	20.8 / 21.5 / **7.75**
	SqueezeNet	29.6 / 29.1 / **13.6**	23.8 / 23.8 / **11.4**	21.0 / 19.4 / **12.9**	15.9 / 15.0 / **7.69**	22.6 / 21.8 / **11.4**
	Average	27.7 / 25.4 / **11.1**	22.2 / 21.5 / **12.1**	19.4 / 18.0 / **10.9**	15.4 / 14.5 / **7.94**	21.2 / 19.8 / **10.5**
ImageNet (1.2M)	VGG-16	15.6 / 13.5 / **8.30**	17.8 / 12.1 / **11.8**	9.19 / **8.42** / 11.4	7.76 / **4.82** / 9.12	12.6 / **9.71** / 10.2
	ResNet152	17.8 / 16.9 / **8.56**	13.3 / 13.7 / **7.59**	11.0 / 9.32 / **6.15**	12.2 / 6.67 / **3.62**	13.5 / 11.6 / **6.48**
	Inception-v3	**12.8** / 20.1 / 15.8	15.2 / 19.1 / **18.4**	12.4 / **12.0** / 13.7	11.8 / 13.6 / **9.96**	**13.0** / 16.2 / 14.5
	DenseNet121	20.3 / 19.9 / **4.73**	14.0 / 14.6 / **6.37**	11.7 / 11.6 / **6.01**	11.4 / 11.2 / **2.86**	14.4 / 14.3 / **5.02**
	SqueezeNet1	25.1 / 24.5 / **13.1**	21.8 / 20.4 / **10.8**	17.1 / 20.1 / **11.5**	12.7 / 15.4 / **6.19**	19.2 / 20.1 / **10.4**
	Average	18.3 / 19.0 / **10.1**	16.4 / 16.0 / **11.0**	12.3 / 12.3 / **9.77**	11.2 / 10.3 / **6.35**	14.5 / 14.4 / **9.31**

Table 7.6 – **Transferability across tasks with access to neither the target data nor the target model.** We set $\epsilon = 16$ and report the mAP for 4952 samples from the PASCAL VOC test-set.

To demonstrate the transferability of our approach across tasks, we attack object detectors using perturbation generators trained with image classifiers. Specifically, we choose the SSD framework with 4 different backbones, namely, VGG16, ResNet50, EfficientNet, and MobileNet-v3 pretrained on PASCAL VOC [61]. In Table 7.6, we report the mAP on the PASCAL VOC test-set, containing 4952 images, for perturbation generators trained with different image classifiers and with $\epsilon = 16$. The performance of the SSD detector on clean images is 68.1, 66.1, 61.1 and 55.4 using VGG16, ResNet50, EfficientNet and MobileNetv3 as backbone, respectively. Our attacks significantly decrease these score and yield lower mAPs than the baselines in most cases. These results highlight, for example, that a generator trained on the synthetic Comics dataset with an image classifier can fool an SSD detector trained on a different domain, with a different architecture, and



(a) **Train**: Paintings on SqueezeNet; **Eval:** DenseNet121



(b) **Train**: Comics10K on ChestXNet; **Eval:** ResNet152

Figure 7.7 – **t-SNE visualizations.** We show t-SNE plots for 1000 normal (blue) images and their adversarial (yellow) counterparts in **(a)** the strict black-box setting and **(b)** the extreme black-box setting.

for a different task. As evidenced by the comparisons with CDA and GAP, learning to generate perturbations that affect the mid-level features is more effective than focusing on the classification boundary.

7.4.6 Additional Analysis

To further analyze our method, we visualize the resulting perturbed images using t-SNE plots. Specifically, in Figure 7.7, we provide t-SNE plots obtained using the final latent representation of 1000 normal (blue) and attacked (yellow) ImageNet images for the strict black-box and extreme black-box scenarios. In both cases, our method yields features that clearly separate the normal and adversarial images with large margins, whereas CDA fails to do so. In Figure 7.8, we visualize the unbounded adversarial images obtained with different methods and also show the bounded adversarial images computed with $\epsilon = 10$ with our approach in the last column. As can also be seen in Figure 7.9, the perturbations follow repeating patterns corresponding to the patterns of the most disrupted mid-level filters. Note that RHP [148] explicitly enforces perturbation homogeneity using a predefined pattern, such as vertical or horizontal patterns. Our filter visualizations for multiple networks reveals that the bottom layers learn such horizontal and vertical stripes. While we tried training the generator to perturb such layers, we observed that larger ϵ strength are required in this case. Overall, our results evidence that attacking mid-level filters that have strong correlations across architectures consistently improves the transfer rates.

Limitations. In our experiments, we choose a single layer l at a time. To this end, we sweep over each block of layers, typically around 5 on average for the studied architectures. We empirically found that the best-performing layer is independent of the choice of the target model to attack. For example, a generator trained with a feature separation loss on the conv4-1 layer in VGG16 transfers to all attack settings at inference time, avoiding repeated sweeps and computational costs. For all our experiments, we set

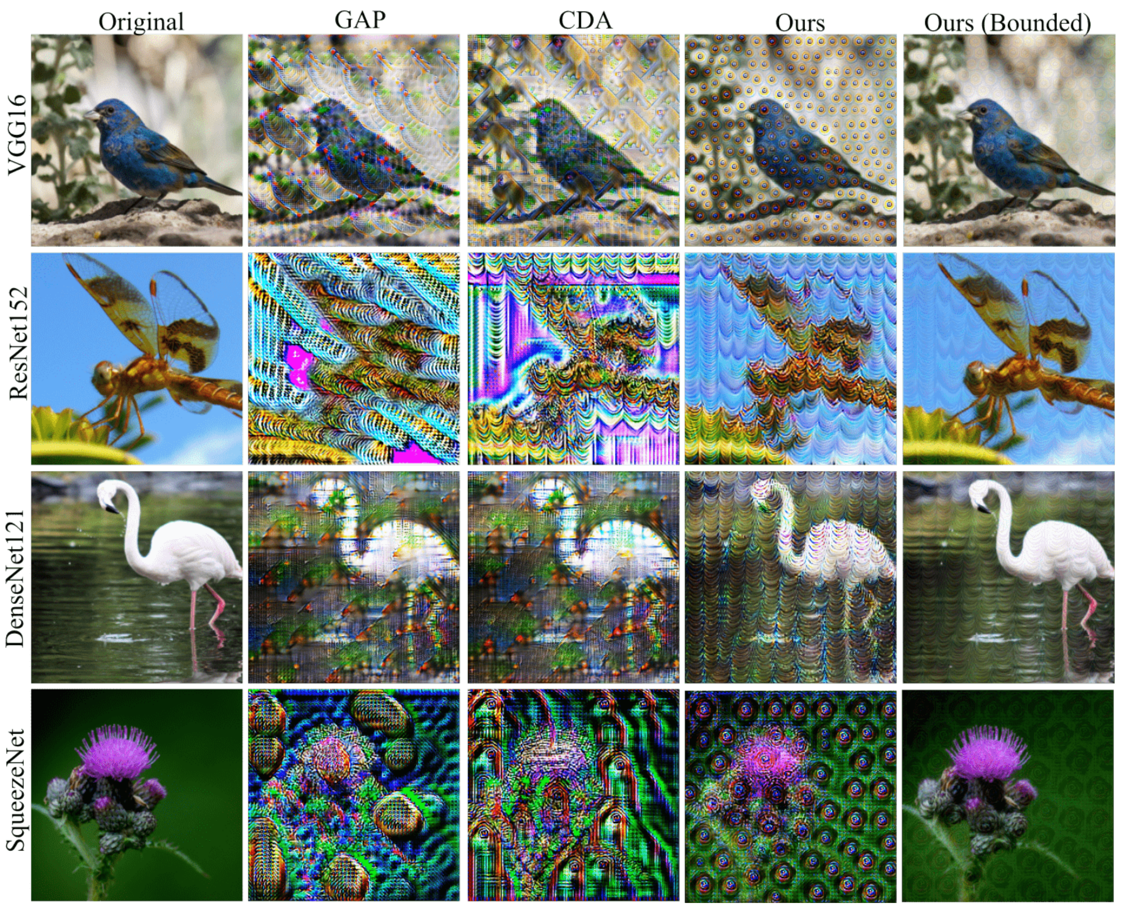


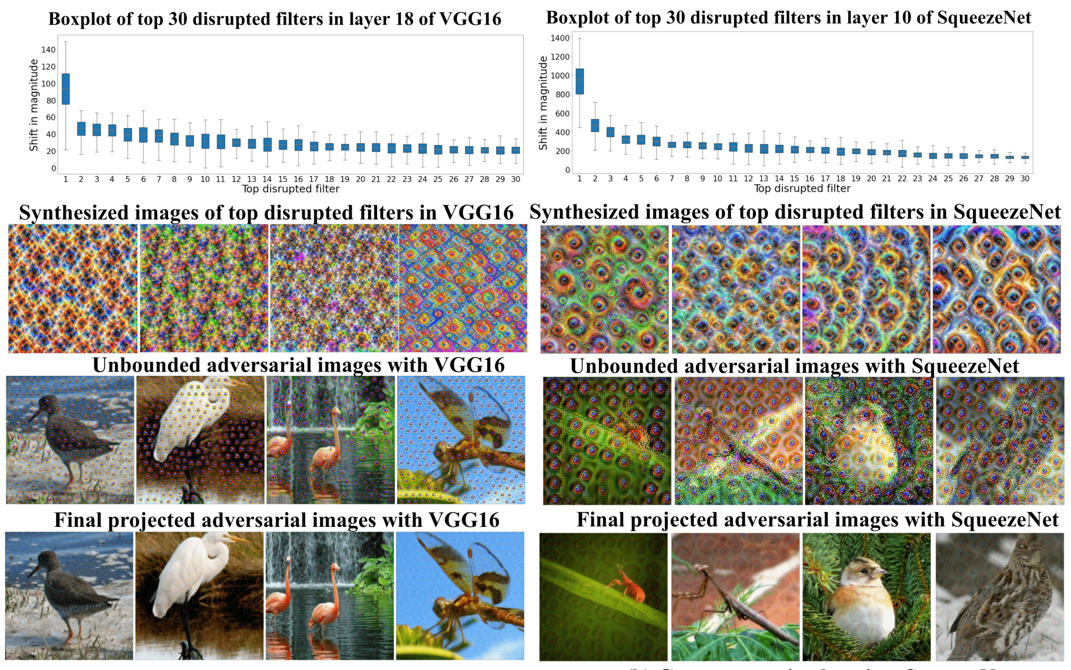
Figure 7.8 – **Qualitative Results.** Visualization of the unbounded outputs obtained with generators trained on ImageNet with different methods. Our approach produces images that are structured and better retain the content of the original images than the baselines. Best viewed in color and zoomed.

the layer l to attack to relu after conv4-1, layer3, mixed6b, denseblock8, and fire10 for VGG16, ResNet152, Inception-v3, DenseNet121, and SqueezeNet, respectively. We report additional results with generators trained at different layers in Secton 7.4.9. Nevertheless, we have no guarantees that such layers truly are optimal; in particular, better results might be achievable by attacking ensembles of layers, but this would require tuning the layer combination.

7.4.7 Additional Quantitative Results

In this section, we provide further insights and ablation studies of our approach.

CKA Metric. In addition to t-SNE visualizations and top activated channels of internal layers in the main chapter, we present here in Table 7.7 the centered kernel alignment



(a) Generator trained against VGG16 **(b) Generator trained against SqueezeNet**

Figure 7.9 – **Correlation between adversarial patterns and most disrupted filters.** The first row shows box plots for the shift in magnitude of the top 30 disrupted filters, i.e., with the largest shift. In the second row, we show the corresponding visualizations obtained using [59] for the top 4 filters. The third row evidences the strong correlation between the top-disrupted filters and the unbounded adversarial images. In the last row, we visualize the bounded adversaries for $\epsilon = 10$. Comparing the left and right blocks shows that the disrupted filters in VGG16 and SqueezeNet contain visually-similar patterns, thereby explaining our high transfer rates. Best viewed in color and zoomed.

(CKA) [130] values to provide additional insights about the similarities between the internal representations of different networks. We observed that the CKA values are typically above 0.75 for the intermediate layers that we considered in our experiments, further explaining the reason for high transferability. For example, on transfer attacks between VGG16 and DenseNet121, the CKA values are highest between the intermediate layer 8 in DenseNet121 and layers 18 and 23 in VGG16. Similarly, between ResNet152 and DenseNet121, the CKA value is 0.76 for the optimal generator configuration, which relies on layer 3 in ResNet152 and layer 8 in DenseNet121.

Iterative versus Generative Attacks. In Table 7.8, we compare the effectiveness of our feature separation loss when used in either an iterative approach, i.e., PGD [168], or with our proposed generator-based one. We set $\epsilon = 10$ with a step size 2 and perform 10 iterations for PGD. We observe that the generator-based method significantly outperforms the iterative method by a large margin of 45pp on average.

Network &		DenseNet121				
	Layer	3	6	8	10	12
VGG16	4	0.71	0.64	0.26	0.21	0.14
	9	0.48	0.58	0.42	0.30	0.19
	18	0.31	0.58	0.75	0.58	0.34
	23	0.26	0.50	0.79	0.69	0.41
	30	0.11	0.23	0.50	0.58	0.45

Network &		DenseNet121				
	Layer	3	6	8	10	12
ResNet152	0	0.69	0.54	0.25	0.20	0.13
	1	0.74	0.82	0.42	0.33	0.21
	2	0.57	0.83	0.65	0.49	0.31
	3	0.24	0.45	0.76	0.80	0.56
	4	0.13	0.25	0.45	0.61	0.74

Table 7.7 – CKA [130] scores to understand the similarities between the internal representations of different networks sampled at 5 layers for 500 images on ImageNet val-set.

Gen. Training (data)	**Discriminator** (Model)	**VGG16**	**ResNet152**	**Inception-v3**	**DenseNet121**	**SqueezeNet**	**Average**
		Iterative / Generative					
ImageNet	VGG16	98.4 / **99.3**	31.3 / **68.4**	30.6 / **46.6**	40.2 / **84.7**	64.6 / **86.5**	41.7 / **71.6**
	ResNet152	47.4 / **99.1**	99.4 / **99.7**	30.9 / **74.9**	43.6 / **98.8**	51.6 / **89.1**	43.4 / **90.5**
	Inception-v3	42.6 / **98.7**	24.2 / **90.2**	94.1 / **99.5**	30.2 / **96.0**	53.9 / **91.5**	37.7 / **94.1**
	DenseNet121	71.2 / **98.4**	49.1 / **99.7**	39.2 / **86.0**	99.6 / **99.6**	63.6 / **95.6**	55.7 / **94.9**
	SqueezeNet	46.7 / **96.1**	22.7 / **76.4**	24.4 / **70.7**	28.4 / **88.7**	99.7 / **99.7**	30.5 / **83.0**
	Average						41.8 / **86.8**

Table 7.8 – **Comparison to an iterative approach.** We set $\epsilon = 10$ and report the fooling rates on 5K ImageNet val-set samples using our feature separation loss with either a 10-step PGD attack or our generator-based one trained on ImageNet data. The generator-based method consistently outperforms the iterative one.

7.4.8 Additional Visualizations

Firstly in Figure 7.10, we show few example images from each domain to understand its characteristics. We observe that ChestX contains larger domain shift than Comics and Paintings to ImageNet. In Figure 7.11, we show the reasons for high transfer rates between ResNet152 and VGG16 architectures due to similar mid-level filter bank. In Figures 7.12, 7.13 and 7.14, we show the unbounded adversarial images obtained by attacking diffferent layer positions against VGG16, SqueezeNet and DenseNet121, respectively. In addition, we also visualize the unbounded adversarial images in strict black-box setting in Figures 7.15, 7.16, 7.17 and 7.18 on VGG16, SqueezeNet, ResNet152 and DenseNet121, respectively. In Figure 7.19, we visualize the output detections of SSD on different backbones with generators trained against DenseNet121 on ImageNet. In Figure 7.20, we provide the t-SNE visualizations of the final features in the standard black-box setting with generators trained on Comics. We set the discriminator to SqueezeNet and compare our approach, shown in the first row, with CDA [191] and GAP [212] in the second and third rows. Overall, we observe a clearer separation between the normal and adversarial features using our method than with the baselines.

7.4.9 Ablation Study

Finally, we perform an ablation study of our generator-based approach by attacking the features at every block of layers in all the studied models. For this experiment, we train the generator with 10K samples taken from ImageNet [46], Comics [18] and Paintings [1]. In Figure 7.21, we plot the fooling rates obtained when training the generator with SqueezeNet [101], VGG16 [237], ResNet152 [89], Inception-v3 [245] and DenseNet [99], each model corresponding to one row of the figure. The columns in each row correspond to the different datasets. In Figure 7.22, we provide the results of a similar study but in the cross-task setting of attacking the SSD detector with 4 different backbones, and thus report the mAP. Overall, the results indicate a good transferability rate for all target models and datasets. Furthermore, we observe that setting the layer l position to conv4-1 with ReLU (18), layer3 (3), mixed6b (11), denseblock8 (8), and fire10 (10) for VGG16, ResNet152, Inception-v3, DenseNet121, and SqueezeNet, respectively, outperforms the other layers in almost all the cases. The only exception arises in the cross-task scenario when training the generator with a VGG16 on ImageNet10K, in which case the optimal success rate is achieved with layer 23 instead of layer 18. These results further confirm that attacking either the initial layers or the final ones is suboptimal for attack transfer.

Implementation Details. We train the ResNet generator following the same architecture as in CDA [191] containing 6 residual blocks using Adam optimizer with $\beta_1 = 0.5$ and $\beta_2 = 0.99$ with learning rate set to 0.0002 and batch size of 16. For all experiments, we train the generator for 1 epoch (80K iterations) on ImageNet1.2M, 10 epochs on Comics40K, and Paintings80K and 50 epochs on ChestX8K. Further, we decay the learning rate by 0.3 at 30 epoch on training with ChestX. We typically warm-start the generators when trained with ResNet and DenseNet discriminators with lower $\epsilon = 4$ for 2K iterations. For attacking adversarially trained models, we perform gaussian smoothing similar to in CDA [191]. Similar to in [191], during the evaluation of the ImageNet target classifiers, we resize the images to the resolution of the source model before passing them to the generator. For training the SSD models, we set the number of iterations to 120K and use an SGD optimizer with momentum 0.9, weight decay 0.0005, and batch size 32. The learning rate starts at 0.001 and is dropped by a factor of 10 at 80K and 100K iterations. During the evaluation phase, we set the confidence threshold to 0.5 to compute the mAP score.

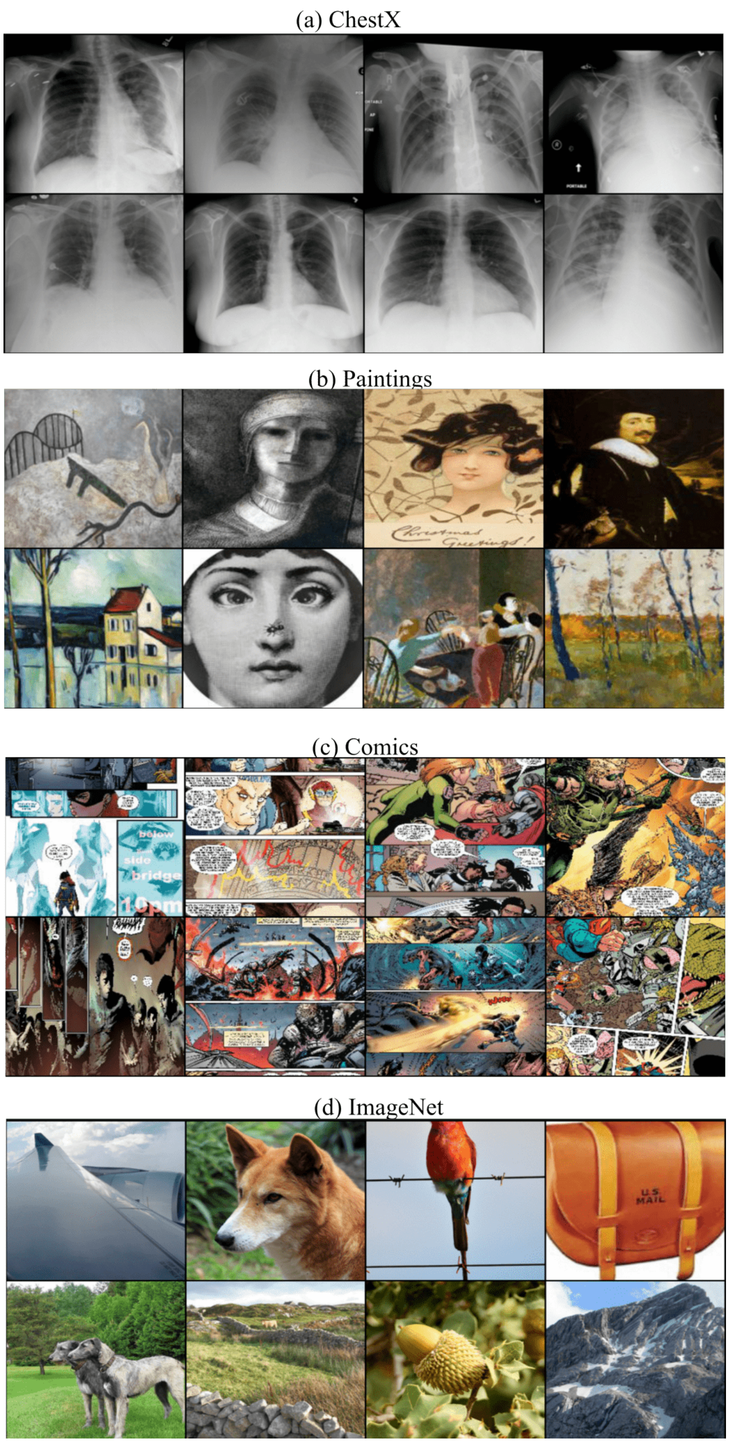


Figure 7.10 – **Sample Images.** Visualization of example images from each domain. ChestX domain focuses on lung region with high domain shift to ImageNet and does not have color. Comics domain contains cartoon images and the Paintings domain mainly focuses on drawings of human subjects. Both Comics and Paintings are synthetic datasets.

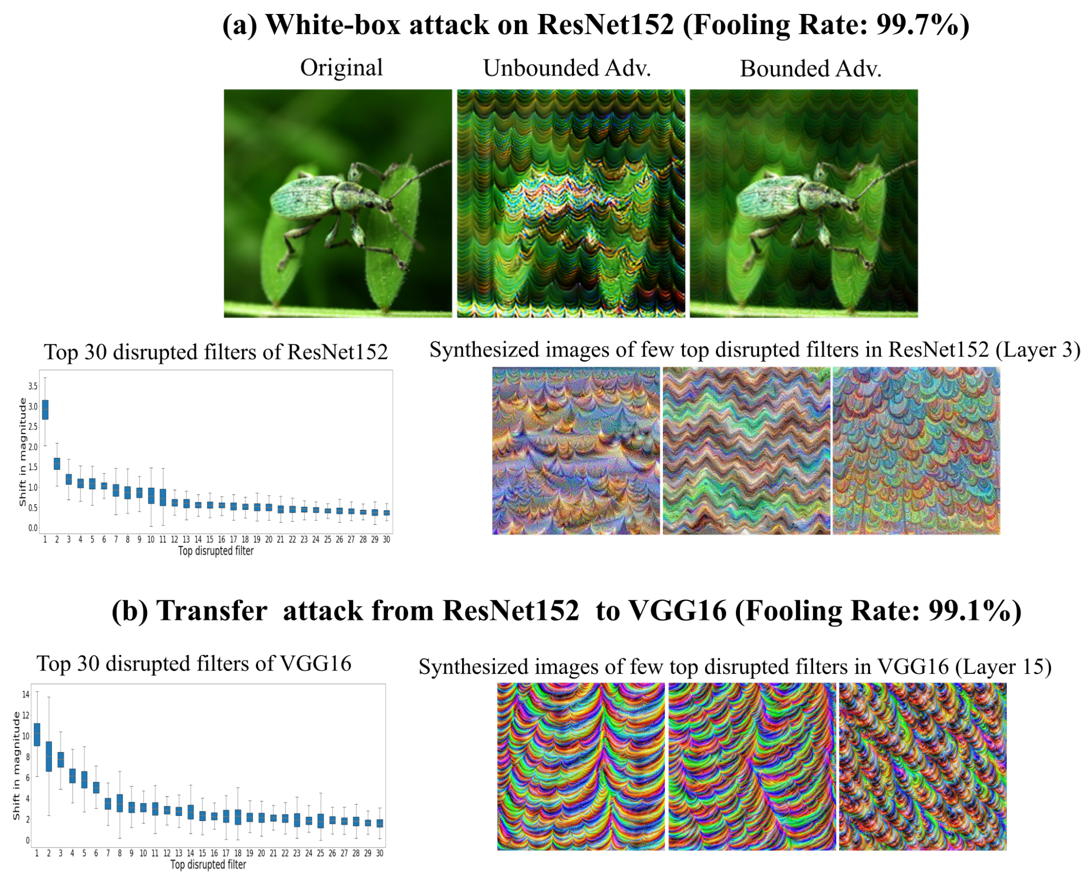


Figure 7.11 – **Black-box transfer from ResNet152 to VGG16.** We analyze the reason for the high transfer rate of 99.1% from ResNet152 to VGG16 by visualizing a few top disrupted filters in intermediate layers. As observed from above, the top disrupted filters in VGG16 and ResNet152 resemble similar texture patterns, and thus disrupting them facilitates high transfer without overfitting to the source model.

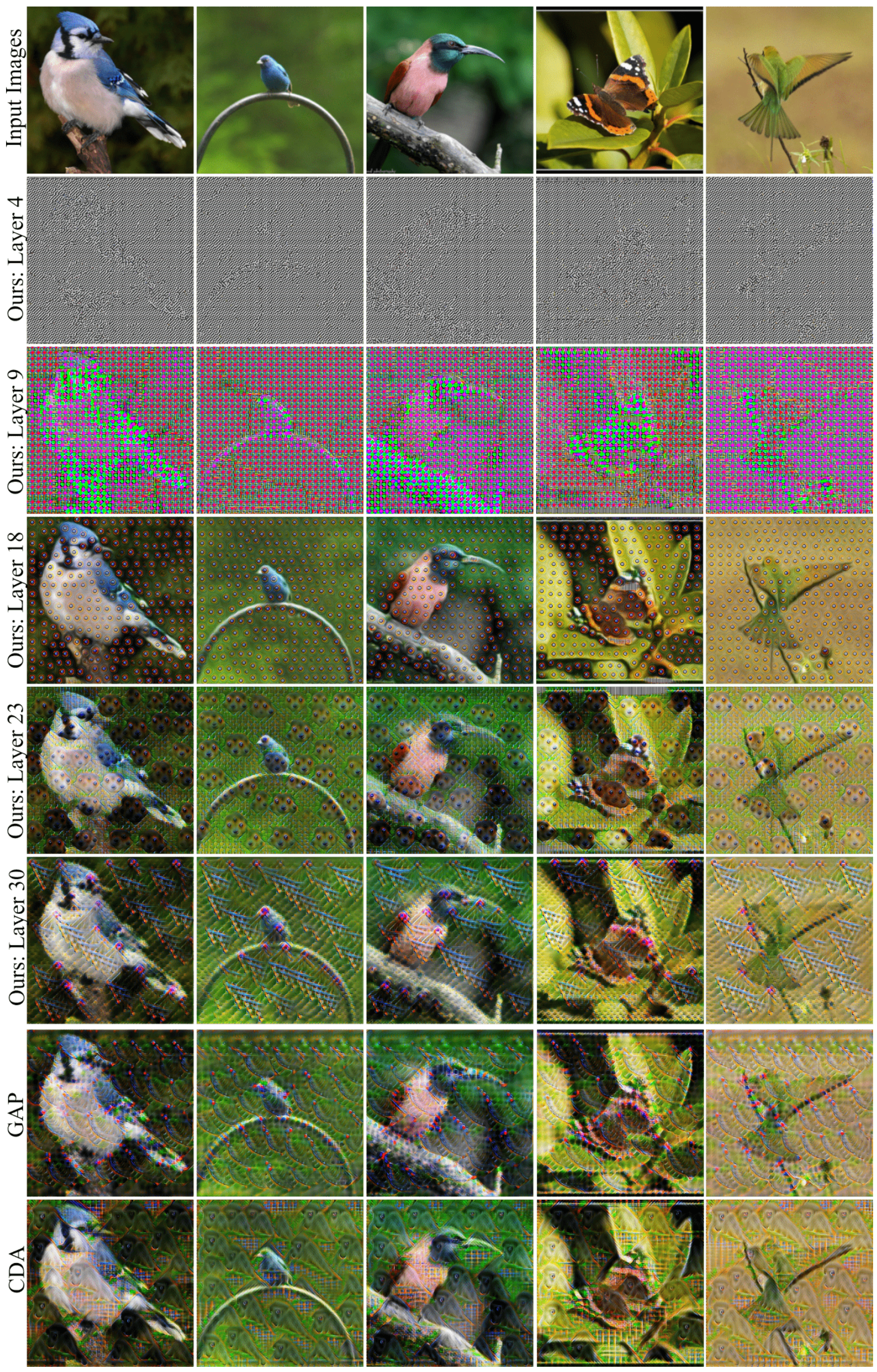


Figure 7.12 – **Qualitative Results by varying the attacked layer position on VGG16.** Visualization of unbounded adversarial outputs at 5 different layers for VGG16 on ImageNet10K training set.

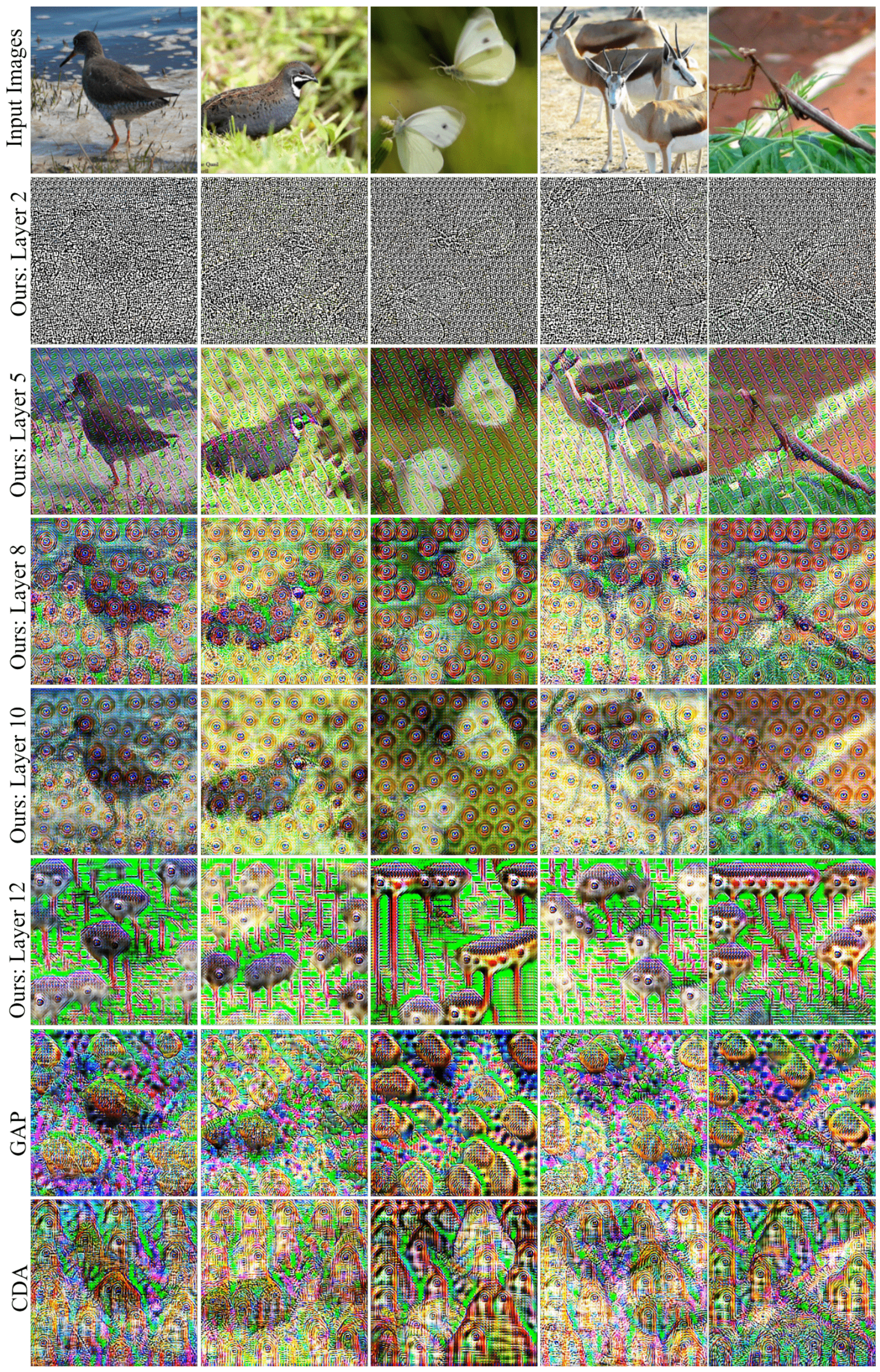


Figure 7.13 – **Qualitative Results by varying the attacked layer position on SqueezeNet1.1.** Visualization of unbounded adversarial ouputs at 5 different layers for SqueezeNet1.1 on ImageNet10K training set.

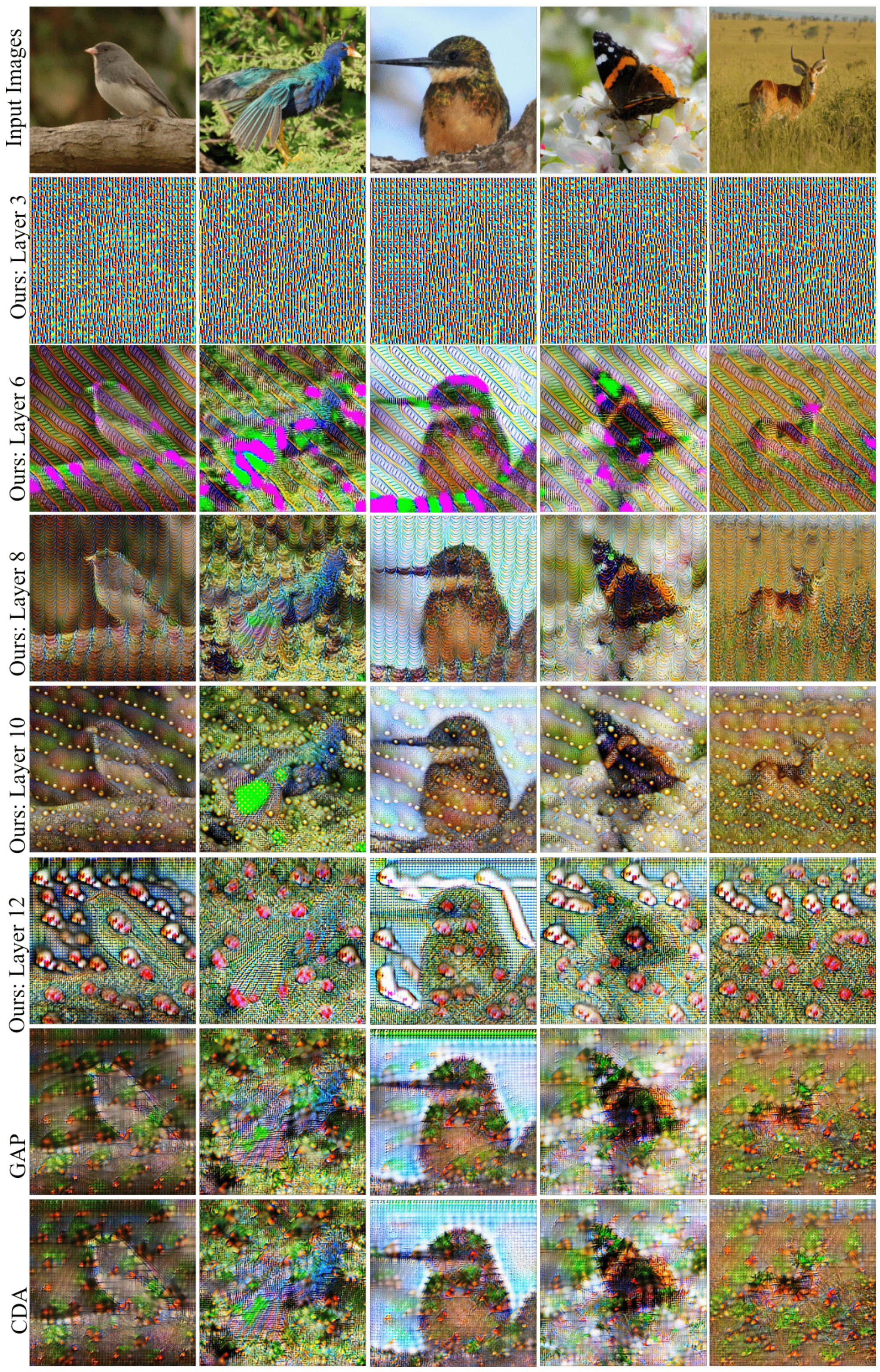


Figure 7.14 – **Qualitative Results by varying the attacked layer position on DenseNet121.** Visualization of unbounded adversarial outputs at 5 different layers for DenseNet121 on ImageNet10K training set.

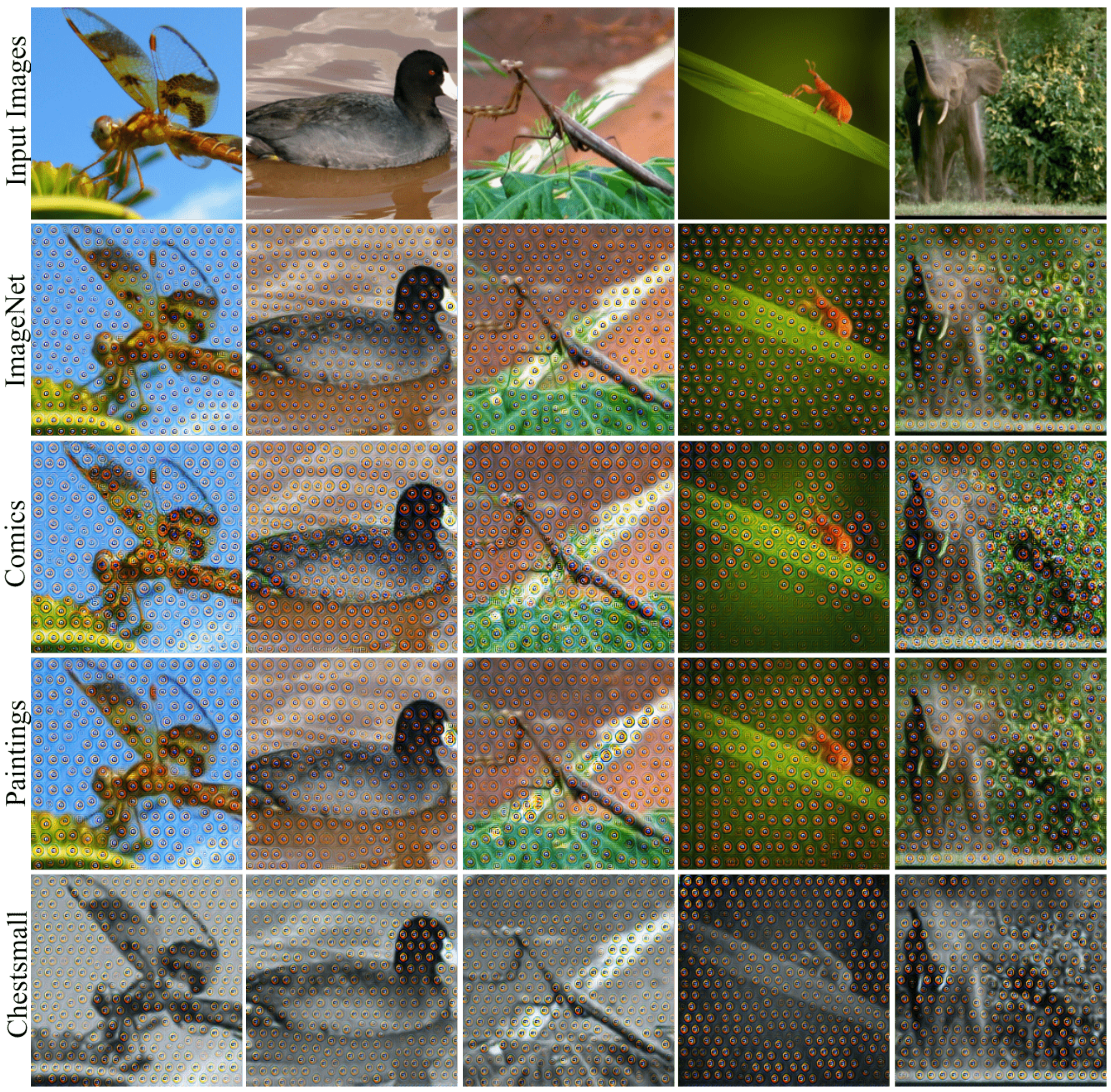


Figure 7.15 – **Qualitative Results by varying the source dataset against VGG16.** Visualization of unbounded adversarial outputs using our approach on VGG16 with different source datasets.

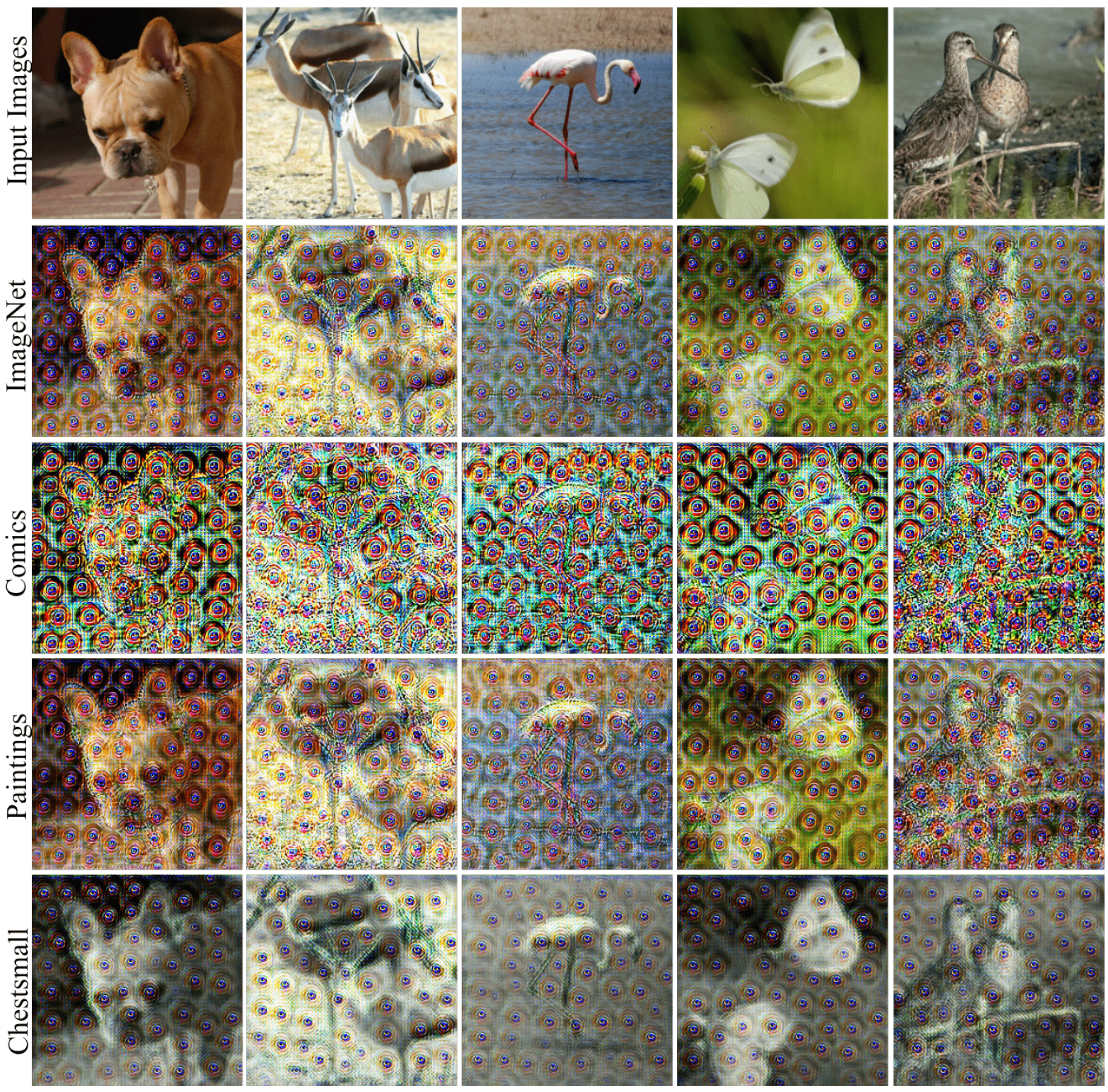


Figure 7.16 – **Qualitative Results by varying the source dataset against SqueezeNet1.1.** Visualization of unbounded adversarial outputs of our approach on SqueezeNet1.1. with different source datasets.

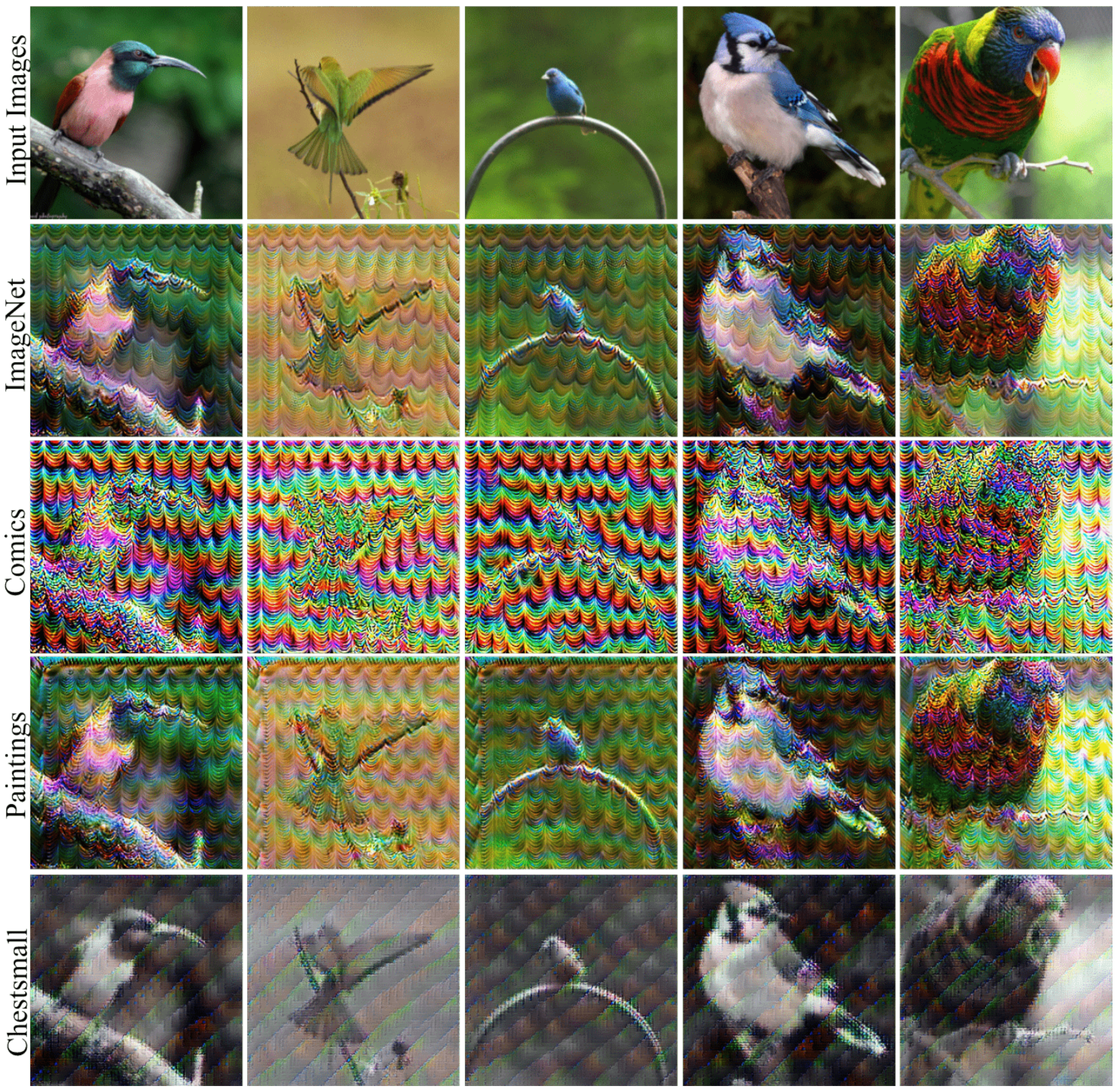


Figure 7.17 – **Qualitative Results by varying the source dataset against ResNet152.** Visualization of unbounded adversarial outputs using our approach for ResNet152 with different source datasets.

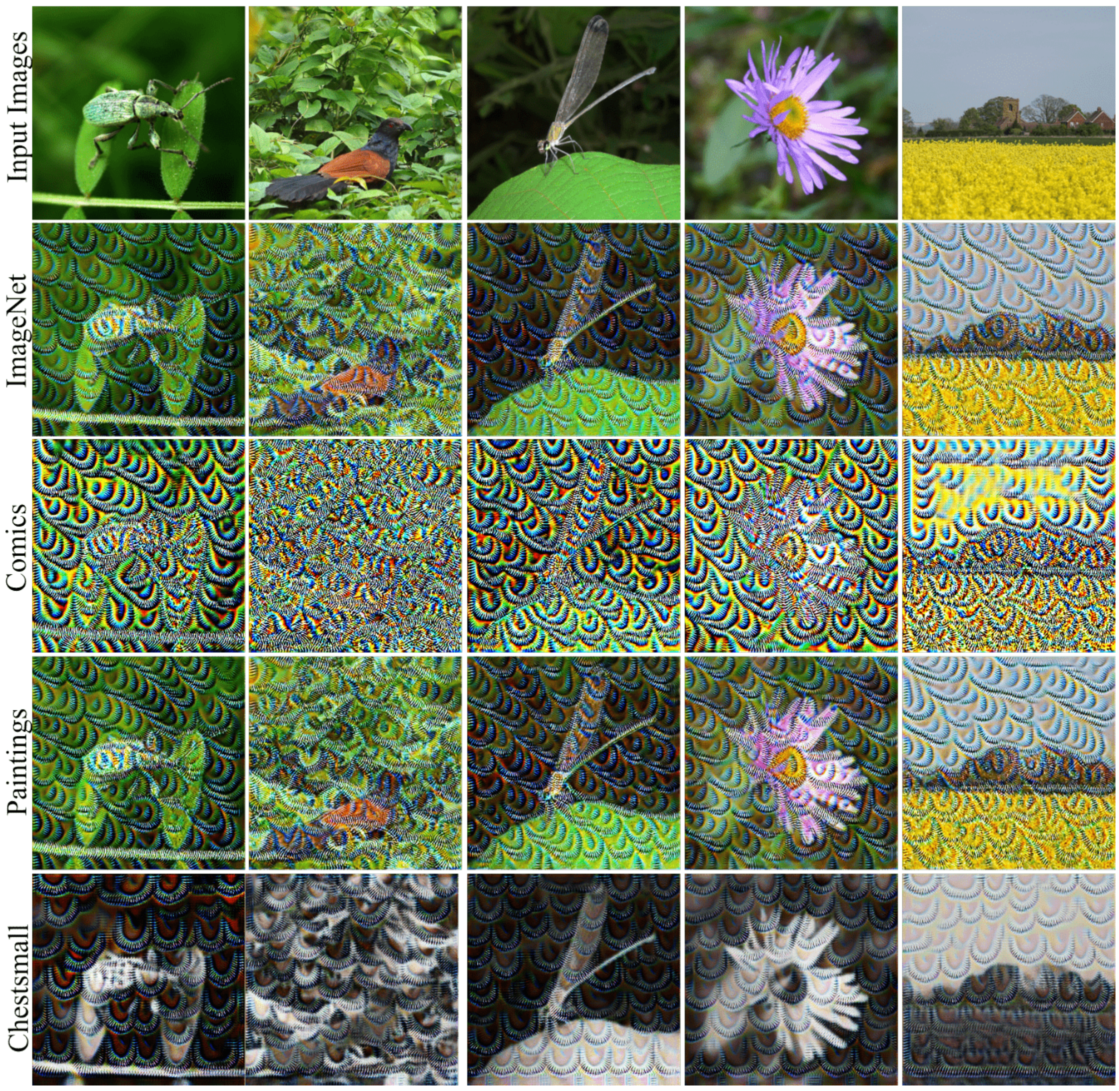


Figure 7.18 – **Qualitative Results by varying the source dataset against Inceptionv3.** Visualization of unbounded adversarial images using our approach for Inceptionv3 with different source datasets.

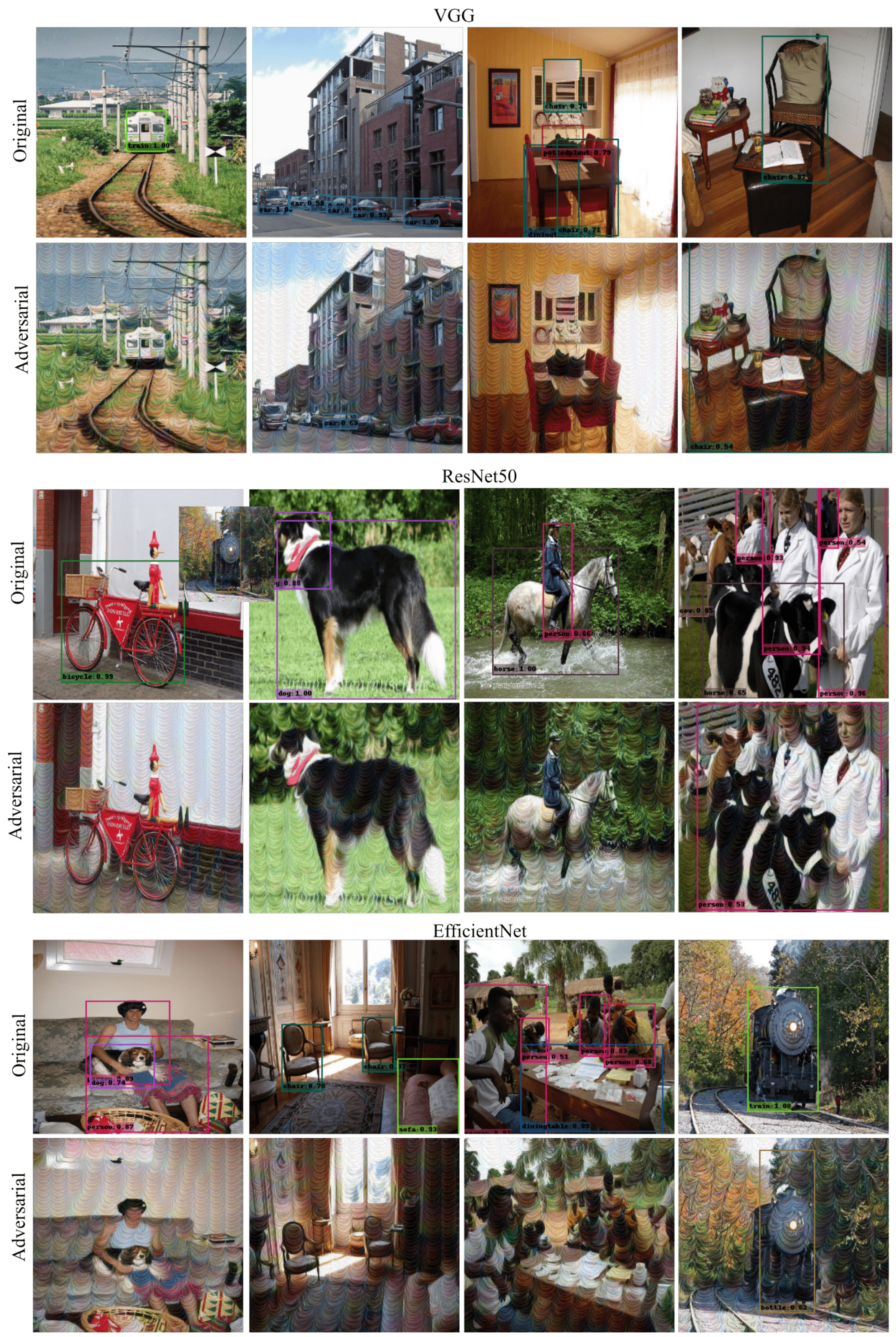
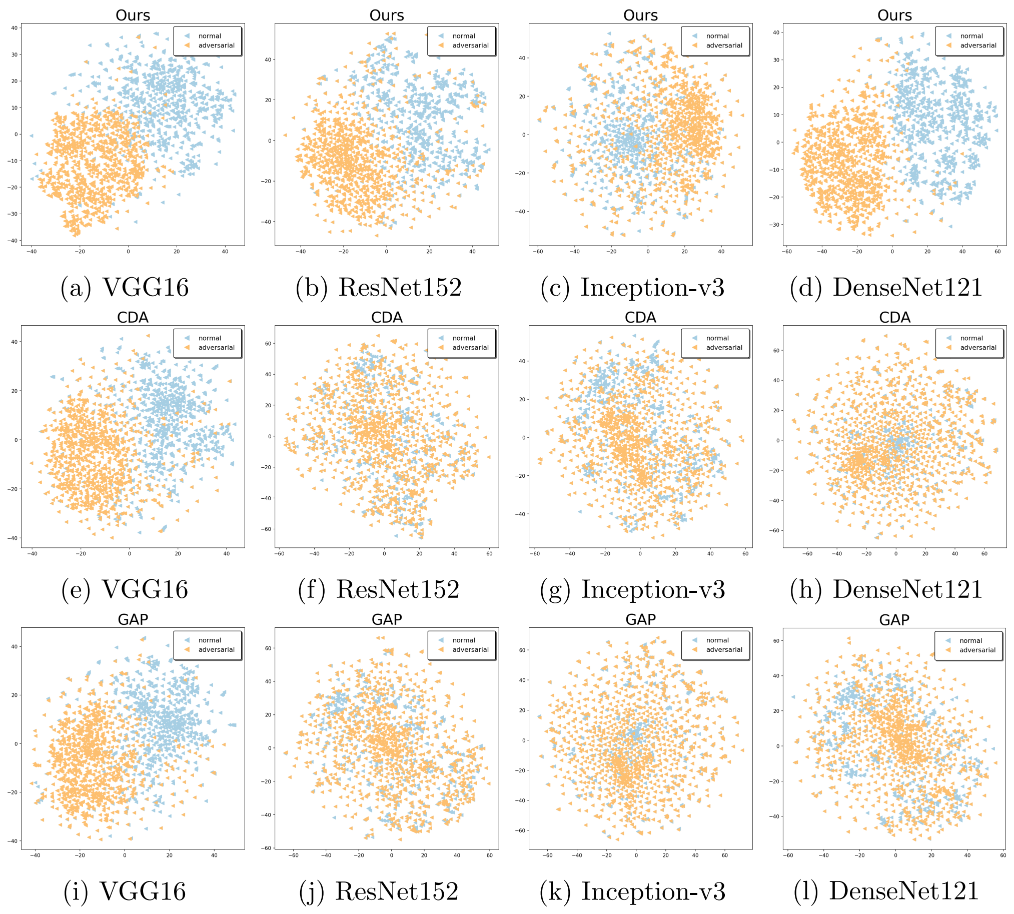


Figure 7.19 – **Qualitative Results on diffferent target task** We visualize the shift in output detections for various backbones on SSD with ϵ set to 16. The generators are trained with DenseNet121 on ImageNet. Best viewed in color and zoom.

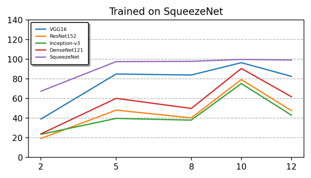


(a) VGG16 (b) ResNet152 (c) Inception-v3 (d) DenseNet121

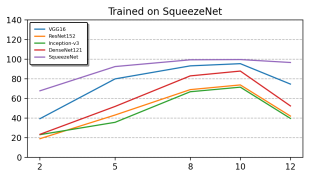
(e) VGG16 (f) ResNet152 (g) Inception-v3 (h) DenseNet121

(i) VGG16 (j) ResNet152 (k) Inception-v3 (l) DenseNet121

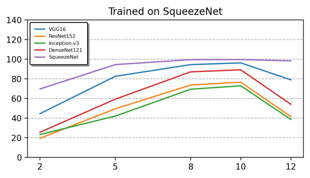
Figure 7.20 – **t-SNE visualizations in the standard black-box setting.** We show t-SNE plots for 1000 normal (blue) ImageNet images and their adversarial (yellow) counterparts in the *standard black-box* setting with a generator trained on *Paintings data with SqueezeNet* discriminator. We provide the visualizations of our method, CDA, and GAP in the first, second, and third row, respectively. Our method separates the features more clearly than the baselines.



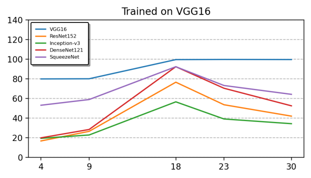
(a) Comics10K



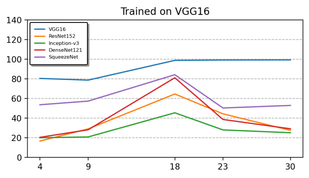
(b) Paintings10K



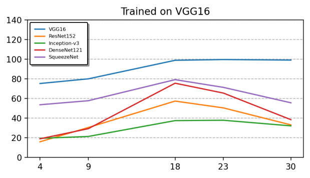
(c) ImageNet10K



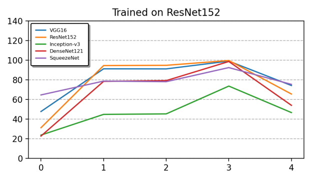
(d) Comics10K



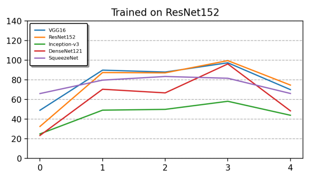
(e) Paintings10K



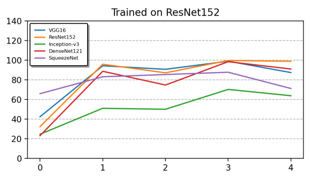
(f) ImageNet10K



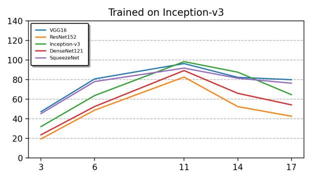
(g) Comics10K



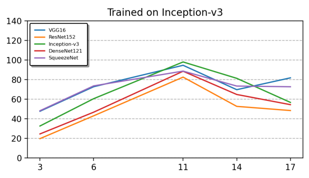
(h) Paintings10K



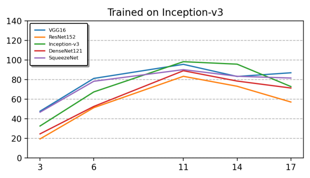
(i) ImageNet10K



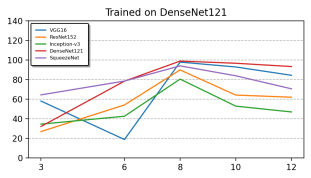
(j) Comics10K



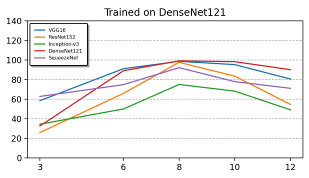
(k) Paintings10K



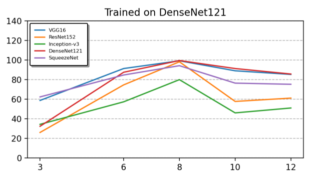
(l) ImageNet10K



(m) Comics10K

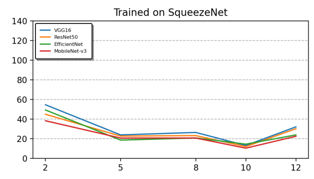


(n) Paintings10K

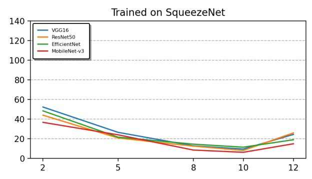


(o) ImageNet10K

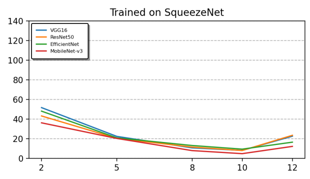
Figure 7.21 – **Impact of the position of the attacked layer in the standard black-box setting.** We vary the position of the layer to attack during training from the bottom layer to the top classification layer. We report, row-wise, the fooling rates (in %) obtained by training the generator with SqueezeNet [101], VGG16 [237], ResNet152 [89], Inception-v3 [245], DenseNet121 [99]. The columns indicate different training sets, containing 10K samples from Comics, Paintings, and ImageNet. In each subplot, we evaluate all 5 networks with $\epsilon = 10$. The best performing layer position is independent of the target architecture and target data. The x-axis represent the layer position and y-axis denote the fooling rate.



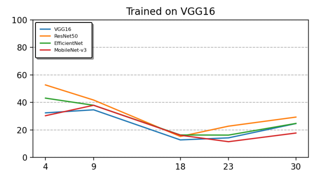
(a) Comics10K



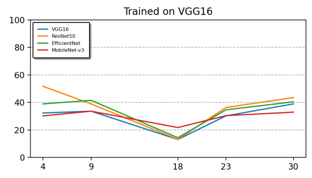
(b) Paintings10K



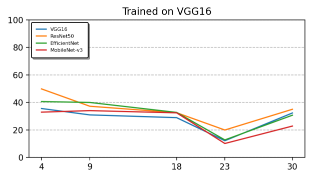
(c) ImageNet10K



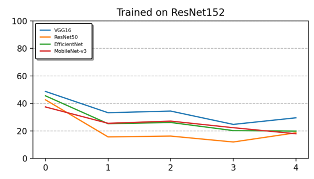
(d) Comics10K



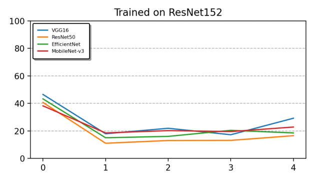
(e) Paintings10K



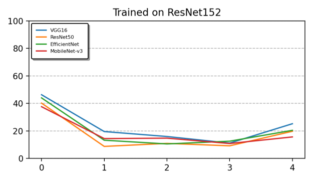
(f) ImageNet10K



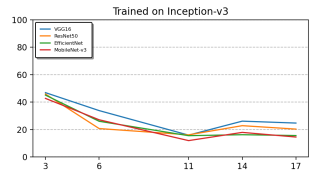
(g) Comics10K



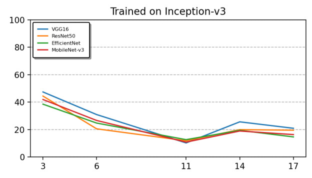
(h) Paintings10K



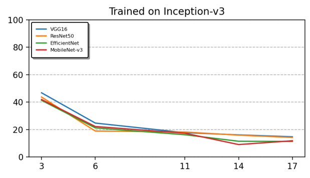
(i) ImageNet10K



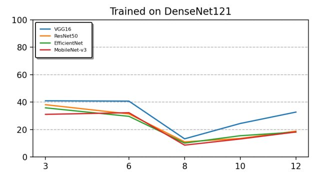
(j) Comics10K



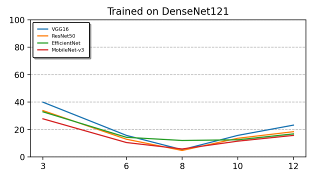
(k) Paintings10K



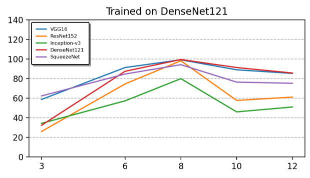
(l) ImageNet10K



(m) Comics10K



(n) Paintings10K



(o) ImageNet10K

Figure 7.22 – **Impact of the position of the attacked layer in the cross-task setting.** We vary the position of the layer to attack during training from the bottom layer to the top classification layer of the recognition models, and then transfer to evaluate the effectiveness on detection models. We report, row-wise, the mAP obtained by training the generator with SqueezeNet [101], VGG16 [237], ResNet152 [89], Inception-v3 [245], DenseNet121 [99]. The columns indicate different training sets, containing 10K samples from Comics, Paintings, and ImageNet. In each subplot, we evaluate 4 backbones with the SSD framework with $\epsilon = 16$. The best performing layer position is independent of the target task and architecture in 24 out of 25 cases. The x-axis represent the layer position and y-axis denote the mAP metric.

7.5 Conclusion

In this chapter, we have explored the use of mid-level features in conjunction with generative networks to learn transferable perturbations. We have shown that a generator trained with feature separation loss can successfully fool models across architectures and tasks, and that a proxy dataset from a different domain can be leveraged to learn effective perturbations. Our experiments demonstrate that our method outperforms the state-of-the-art attacks across a wide range of architectures and tasks. We have limited our experiments to undefended models due to the unavailability of publicly available robust models across architectures. Furthermore, we believe that a deeper analysis of learned filter banks with respect to changes in architectures can shed light on building better black-box models. Overall, in this chapter, our approach is motivated by the understanding of DNNs from the filter-bank perspective. In the next chapter, we show how adversarial attacks can be used to study the internal representation of disentanglement-based DNNs.

8 Understanding Pose and Appearance Disentanglement in 3D Human Pose Estimation

In the previous chapter, we focused on the transferability of adversarial attacks by understanding the internal workings of DNN. In this chapter, we continue with the goal of understanding DNNs but instead use adversarial attacks as one of the tools to study the disentanglement in the latent representations of pose and appearance in 3D pose estimation task. As 3D human pose estimation can now be achieved with very high accuracy in the supervised learning scenario, tackling the case where 3D pose annotations are not available has received increasing attention. In particular, several methods have proposed to learn image representations in a self-supervised fashion so as to disentangle the appearance information from the pose one. The methods then only need a small amount of supervised data to train a pose regressor using the pose-related latent vector as input, as it should be free of appearance information.

In this chapter, we carry out in-depth analysis to understand to what degree the state-of-the-art disentangled representation learning methods truly separate the appearance information from the pose one. First, we study disentanglement from the perspective of the self-supervised network, via diverse image synthesis experiments. Second, we investigate disentanglement with respect to the 3D pose regressor following an adversarial attack perspective. Specifically, we design an adversarial strategy focusing on generating natural appearance changes of the subject, and against which we could expect a disentangled network to be robust. Altogether, our analyses show that disentanglement in the three state-of-the-art disentangled representation learning frameworks if far from complete, and that their pose codes contain significant appearance information. We believe that our approach provides a valuable testbed to evaluate the degree of disentanglement of pose from appearance in self-supervised 3D human pose estimation.

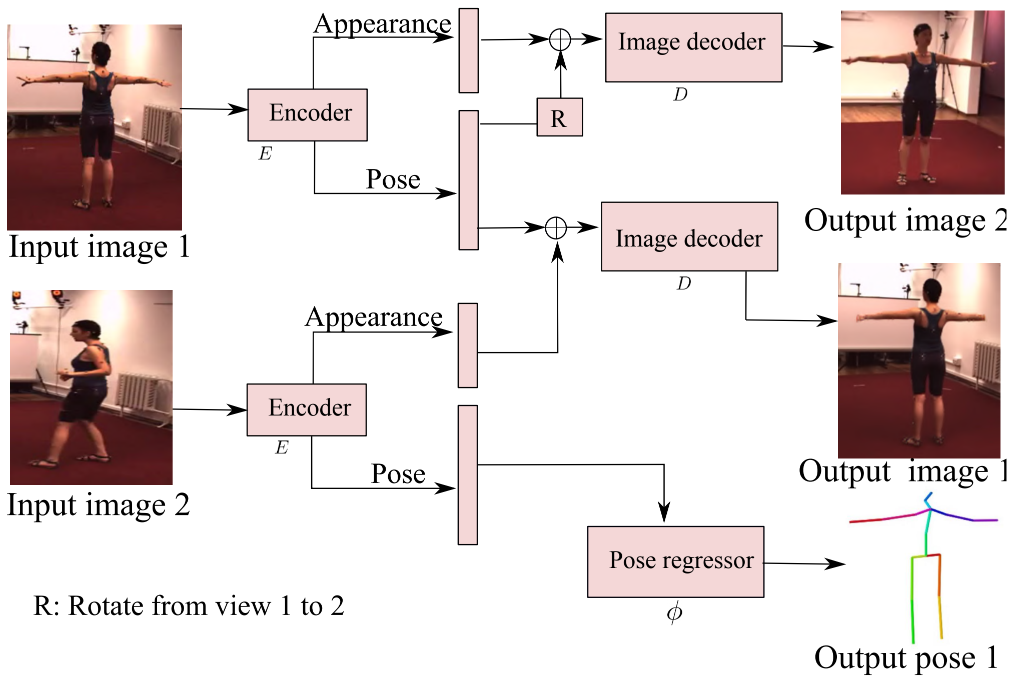


Figure 8.1 – **Disentanglement-based Representation Learning.** Given a reference frame and another frame from either a different view or a different time instant, an encoder learns a representation separated into two components, appearance and pose, in a self-supervised fashion. A pose regressor is then trained using limited annotated data to map the latent pose vector to a 3D human pose.

8.1 Introduction

Monocular 3D human pose estimation has been at the heart of computer vision research for decades, and tremendous results can now be achieved in the supervised learning setting [105, 115, 174, 175, 176, 208, 209, 211, 223, 248, 250]. Unfortunately, obtaining 3D pose annotations for real images remains very expensive, particularly in the wild. As such, self-supervised learning approaches have received an increasing attention in the past few years [47, 96, 221, 222]. One of the common factors across all these methods is their aim to learn a latent representation of the image that disentangles the person's pose from their appearance. In practice, as shown in Figure 8.1, this has been achieved by leveraging access to either multiple views [221, 222] or video sequences [47, 96] during training. In either case, one then only needs access to a small amount of supervised data to effectively train a pose regressor from the pose-related portion of the latent code to the actual 3D pose, because this portion of the latent code should in theory contain only pose-relevant information.

Despite the impressive progress of these self-supervised 3D human pose estimation methods, several fundamental questions about their learnt representations remain unanswered. For example, to what extent are the pose and appearance latent vectors disentangled?

Do these two representations contain truly complementary information, or do they share some signal? How do the different sources of self-supervision, i.e., multiple views or temporal information, affect the disentanglement of these representations?

In this chapter, we seek to provide a deeper understanding of such disentangled representations by analyzing the resulting latent spaces in two ways. First, we study the disentanglement of the latent pose and appearance vectors with respect to the self-supervised representation learning network. In this context, we analyze both the images synthesized by altering the appearance codes in different ways, and the influence on pose and appearance of different channels in the latent pose codes. Second, we investigate the disentanglement with respect to the supervised 3D pose regressor. To this end, we follow an adversarial attack strategy, aiming to modify the subject's appearance so as to affect the regressed 3D pose. However, instead of exploiting a standard adversarial attack technique [79, 138, 168], against which disentangled pose networks were never meant to be robust, we design a dedicated framework that should be much more favorable to such networks. Specifically, we seek to alter only the latent appearance vector so as to affect the 3D pose regressed from the latent pose vector extracted from the image synthesized using the modified appearance vector with the original pose one.

Our experiments on the state-of-the-art disentangled representation learning frameworks, NSD [221], CSSL [96] and DRNet [47], evidence that, across the board, *disentanglement is not complete and the pose codes of these frameworks contain appearance information.* Our work provides the tool to study the effectiveness of different disentanglement-based training strategies and will serve as a valuable testbed to analyze the extent of disentanglement in future frameworks.

Contributions. To summarize, our contributions are twofold. (1) We systematically analyze the latent pose and appearance representations in several representative disentangled networks. Our experiments lead to an interesting finding that the latent pose vectors contain almost all of the subject's appearance information. (2) We introduce an adversarial strategy to understand the disentanglement of 3D pose from natural appearance changes.

8.2 Related Work

Disentanglement-based 3D Human Pose Estimation. Disentangling pose and appearance in 3D pose estimation was first proposed in DRNet [47], where a discriminator was employed to distinguish if the time-varying features from two images represented the same subject or not. Furthermore, the distance between the time-invariant, i.e., appearance, component of one subject at two different time instants was minimized, and the time-varying pose features were encouraged to be indistinguishable across subjects, thereby ensuring that appearance information did not leaked into the pose features. In [221, 222], disentanglement was achieved via the use of multiple views during training, leveraging the intuition that, for one subject, the pose features extracted from one view and rotated to a different view at the same time instant should be the same as those directly extracted from that view, and the appearance features at different time instants should be similar so as not to contain pose information. More recently, [96] designed a contrastive loss to force the latent appearance features in temporally-distant frames to remain close while encouraging the pose features to be different from each other. All these methods learn the disentangled representation from unsupervised data, and then train a shallow regressor to predict 3D pose from the pose latent vector using a limited amount of pose labels. In this work, we study how disentangled the appearance and pose features extracted by these methods truly are. To this end, we provide analyses based on diverse image synthesis experiments and on adversarial attacks.

Adversarial Attacks. Deep neural networks were first shown to be vulnerable to adversarial examples in [246]. Following this, several attacks have been proposed, using either gradient-based approaches [79, 138] or optimization-based techniques [26, 41, 53, 182, 201]. To study the disentanglement of pose and appearance in 3D human pose estimation, we seek to analyze if appearance changes can affect the regressed 3D pose. In principle, we could use any of the above-mentioned attack strategy to do this. However, they offer no control on the generated perturbations, and thus could potentially incorporate structures that truly suggest a different pose. In other words, the disentangled networks cannot be expected to be robust to such attacks. Therefore, we design an attack strategy to which they can be expected to be robust. Specifically, we synthesize an image by modifying only the appearance code of the network of interest, and show that the 3D pose regressed from that image will typically differ from the original one. Our attacks can be thought of as inconspicuous ones, as the generated image looks natural, with only appearance changes to the subject. Other works [16, 116, 214, 235, 242, 312] have designed strategies to generate realistic adversarial images, typically focusing on face recognition datasets and using GANs [7, 78, 178]. Our approach nonetheless fundamentally differs from those in both methodology and context; our main goal is not to attack disentangled 3D human pose networks but to study their level of disentanglement. Therefore, we design an attack strategy that is most favorable for these networks, and against which they can be expected to be naturally robust.

Measuring Disentanglement. In other contexts than human pose estimation, several works have proposed metrics to quantify the degree of disentanglement of latent vectors [51, 58, 160]. These methods are of course also applicable to the self-supervised learning frameworks that we will analyze, and we will report these metrics in our experiments. However, these metrics do not provide any understanding of where disentanglement fails. This is what we achieve with our diverse analyses.

8.3 Disentangled Human Pose Estimation Networks

Given an image as input, 3D human pose estimation aims to predict the 3D positions of J body joints, such as the wrists, elbows, and shoulders. When no annotations are available for the training images, an increasingly popular approach consists of learning a latent representation that disentangles appearance from pose in a self-supervised fashion. Here, we review disentanglement-based 3D human pose estimation frameworks that we will analyze in Sections 8.5 and 8.6.

Existing disentanglement-based frameworks essentially all follow the same initial steps. The input image $\mathbf{I}$ is first passed through a spatial transformer network $\mathcal{S}$ to extract the bounding box corresponding to the human subject. An encoder E then takes the cropped bounding box $\mathbf{I}_c$ as input and outputs a latent vector $\mathbf{h}$ comprising two components, that is $E : \mathbf{I}_c \rightarrow [\mathbf{h}_a, \mathbf{h}_p]$. The first component, $\mathbf{h}_a$, aims to encode the subject's appearance while the second, $\mathbf{h}_p$, should represent the subject's pose. The networks are trained without any 3D pose annotations, and thus supervision is achieved via image reconstruction. Specifically, a decoder D takes the complete the latent vector $\mathbf{h}$ as input and and outputs a reconstructed version of the cropped image $\tilde{\mathbf{I}}_c$, with an additional mask $\mathbf{M}$ corresponding to the subject's silhouette. The cropped image is further merged with a pre-computed background image $\mathbf{B}$ to obtain the final reconstructed input image $\tilde{\mathbf{I}}$.

The main difference between existing frameworks lies in the way they encourage the disentanglement of pose and appearance. Specifically, the different frameworks train the encoder E and decoder D as follows:

NSD [221]. The neural scene decomposition (NSD) approach leverages the availability of multiple views during training. Given a pair of images from two views at time t, NSD passes one image to the encoder to obtain an appearance vector $\mathbf{h}_a^t$ and a pose vector $\mathbf{h}_p^t$. The pose vector $\mathbf{h}_p^t$, shaped as a 3D point cloud, is rotated to the second view using the ground-truth camera calibration between the two views to obtain a transformed pose vector $\mathbf{h}_{p,r}^t$. Furthermore, to factor out appearance from pose, NSD replaces the appearance vector $\mathbf{h}_a^t$ by an appearance vector $\mathbf{h}_a^{t_1}$ of the same subject at a different time instant t_1. The decoder D then takes as input $\mathbf{h} = [\mathbf{h}_{p,r}, \mathbf{h}_a^{t_1}]$ and aims to reconstruct the image from the second view at time t.

CSSL [96]. Instead of using multiple views, contrastive self-supervised learning (CSSL) exploits temporal information from videos to learn a latent representation of pose and appearance. To achieve disentanglement, CSSL encourages the distance between the latent pose vectors $\mathbf{h}_p^{t_1}$ and $\mathbf{h}_p^{t_2}$ of two frames, t_1 and t_2, to reflect their temporal distance. Furthermore, similarly to NSD, CSSL swaps the appearance vectors $\mathbf{h}_a^{t_1}$ and $\mathbf{h}_a^{t_2}$ of the two video frames when performing image reconstruction so as to force them to learn time-invariant information, thus encoding appearance.

DRNet [47]. The disentangled representation network (DRNet) uses a similar strategy to that of CSSL, consisting of randomly choosing two temporal frames, t_1 and t_2, from a video. However, DRNet aims to achieve disentanglement in two ways: (1) By minimizing the distance between the two appearance vectors $\mathbf{h}_a^{t_1}$ and $\mathbf{h}_a^{t_2}$; and (2) by exploiting an adversarial network to make the pose vector $\mathbf{h}_p$ independent of the subject's appearance. Specifically, this is achieved by training the additional discriminator to output the subject's identity given the pose vector as input, and training the encoder E in an adversarial fashion to fool the discriminator.

Auto-encoder. The 3 methods discussed above constitute the state of the art in learning disentangled representations for 3D human pose estimation. In our experiments, we will further evaluate a naive auto-encoder baseline that simply reconstructs the image by passing the complete latent vector $\mathbf{h}$ as input to the decoder D. In other words, this baseline does not enforce any form of disentanglement, and one could thus expect it to be less robust to our appearance-based attacks than disentanglement-based frameworks.

Once trained on a large corpus of unannotated images in a self-supervised manner, the frameworks discussed above employ a 2 layer pose regressor $\phi : \mathbf{h}_p \rightarrow \mathbf{q}$ to predict the 3D pose $\mathbf{q}$ from the latent pose vector $\mathbf{h}_p$. This pose regressor is trained with a small amount of supervised data, while freezing the weights of the encoder.

8.4 Training Details

We employ the widely-used Human3.6M [106] dataset. We use the PyTorch [206] library on a single GPU with 32GB RAM to run our experiments. All the networks are first trained on Human3.6M without any label supervision using Subjects 1, 5, 6, 7, and 8, corresponding to around 300K images. Specifically, we train the models with images of size 500×500 and resize the cropped image output by the spatial transformer to 128×128 before passing it to the encoder. The encoder yields an appearance latent vector $\mathbf{h}_a \in \mathbf{R}^{128}$ and a pose latent vector $\mathbf{h}_p \in \mathbf{R}^{600}$. We train all baseline models for 200K iterations. Once trained, we freeze the weights of the encoder and then train a pose regressor containing 2 hidden layers on the PoseTrack dataset containing 35K images. This pose regressor takes as input the latent pose vector for the disentangled networks and the complete code for AE. When reporting 3D errors, we follow the standard evaluation protocol of self-supervised 3D human pose estimation [96, 221], and report the mean per joint position error (MPJPE) in mm.

8.5 Disentanglement w.r.t. the Self-Supervised Network

In this section, we study the disentanglement of pose and appearance within the self-supervised representation learning network itself. To this end, we first analyze the impact of the latent appearance vector on the images synthesized by the network's decoder. We then turn to investigating the influence on pose and appearance of different channels in the latent pose vector.

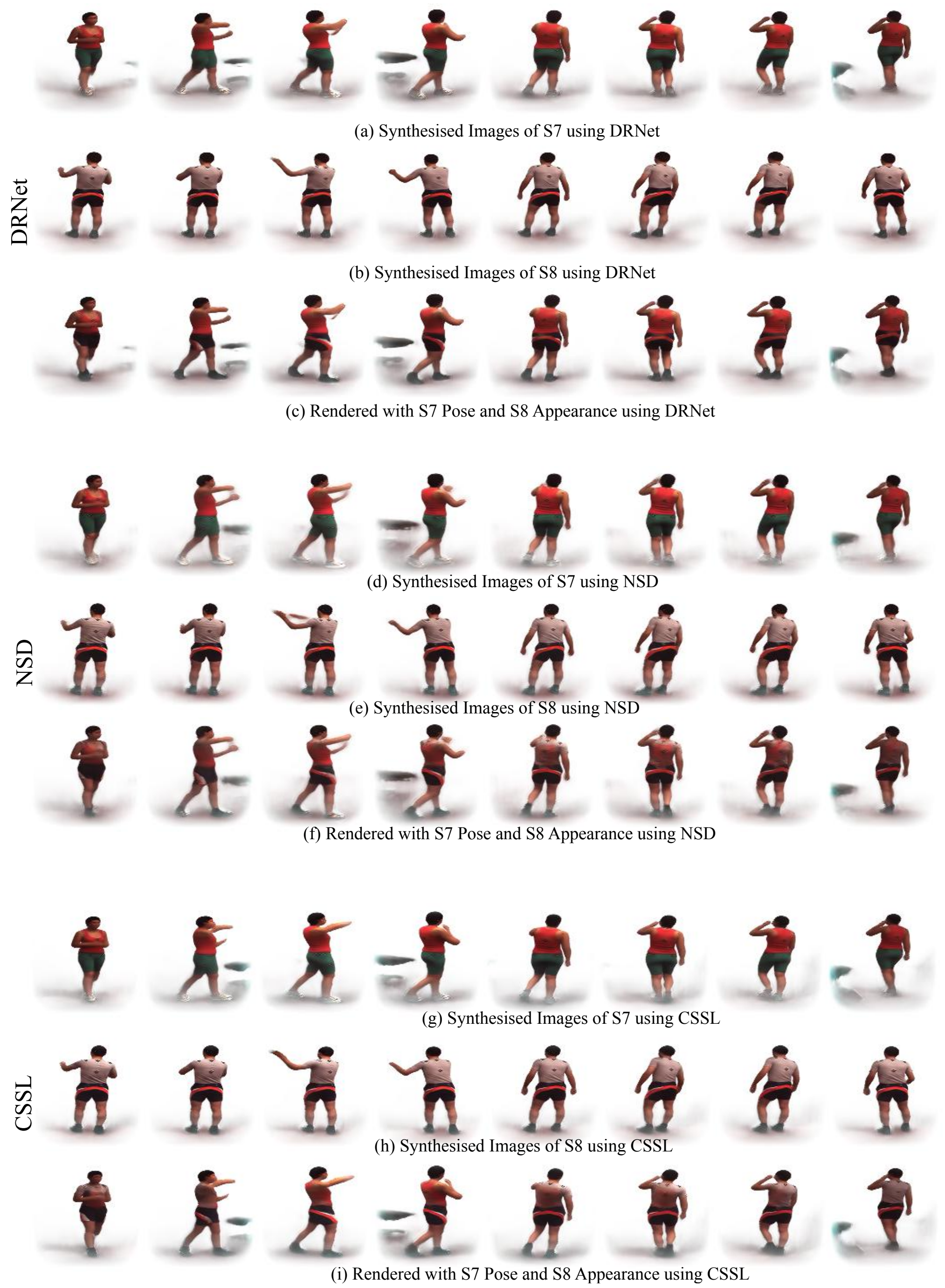
8.5.1 Effect of the Appearance Vector on Synthesized Images

Our first analysis consists of visualizing the images generated by the network's decoder. In particular, we leverage the intuition that, if the pose and appearance vectors were disentangled, altering the appearance vector while keeping the pose one fixed should yield images with a different subject's appearance but the same pose. We investigate this via the two strategies discussed below.

First, we synthesize novel images by mixing the appearance and pose information from two subjects, S8 and S7. The top two rows of Figure 8.2(a) show the images synthesized with DRNet*without mixing the appearance vectors; these images look similar to the original ones, depicting two clearly different subject's appearances.* By contrast, the images in the third to fifth row of the figure, obtained by using S7's pose vector and S8's appearance one, still contain appearance information of S7. This is particularly the case for the images synthesized using DRNet and NSD, in which the subject's shirt has taken the red color of that of S7, although we use only S7's pose code in the synthesis process. CSSL is less subject to such failures, but they nonetheless occur in some cases, such as in the third and fourth columns.

As a second experiment, we replace the appearance vector with a zero vector. We then combine this zero appearance vector with the pose vector obtained from the original image shown in the first row of Figure 8.3. As can be seen from the second row, even though we use the same zero appearance vector to generate images of different subjects, the synthesized images retain almost all the appearance information of the original images, except near the head region.

Both of these experiments evidence that the pose code contains a significant amount of appearance information and that the disentanglement is thus not complete. Nevertheless, both experiments also show that modifying the appearance code indeed does not impact the subject's pose in the synthesized image. To further verify whether the appearance codes are truly free of pose information, we visualize the appearance codes of all images of a S7 using t-SNE in Figure 8.4. The resulting plot shows nicely-separated clusters, which can be observed to correspond to action categories. This suggests that, although modifying the appearance code does not visually change the subject's pose in the synthesized images, the appearance codes still contain information about the subject activity, and thus about their pose.



(a) Synthesised Images of S7 using DRNet

(b) Synthesised Images of S8 using DRNet

(c) Rendered with S7 Pose and S8 Appearance using DRNet

(d) Synthesised Images of S7 using NSD

(e) Synthesised Images of S8 using NSD

(f) Rendered with S7 Pose and S8 Appearance using NSD

(g) Synthesised Images of S7 using CSSL

(h) Synthesised Images of S8 using CSSL

(i) Rendered with S7 Pose and S8 Appearance using CSSL

Figure 8.2 – **Synthesizing novel images.** We take the pose information from S7 (first row in each block) and the appearance information from S8 (second row in each block) and synthesize novel images in the third row of each block for different disentangled networks. The synthesized images retain some appearance information (red shirt) of S7 although we only use S7's pose code in the synthesis.

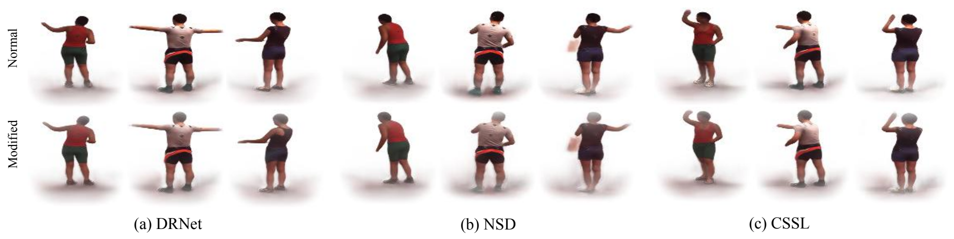


Figure 8.3 – **Replacing the appearance code with a fixed zero vector.** In the first row, we show the original synthesized images for three subjects on different networks. In the second row, we set the values in the appearance vector to zero and use the same pose vectors as in the first row. Despite using the same zero appearance vector for all subjects, the outputs do not appear similar in content and instead retain almost all the appearance information of the original images.

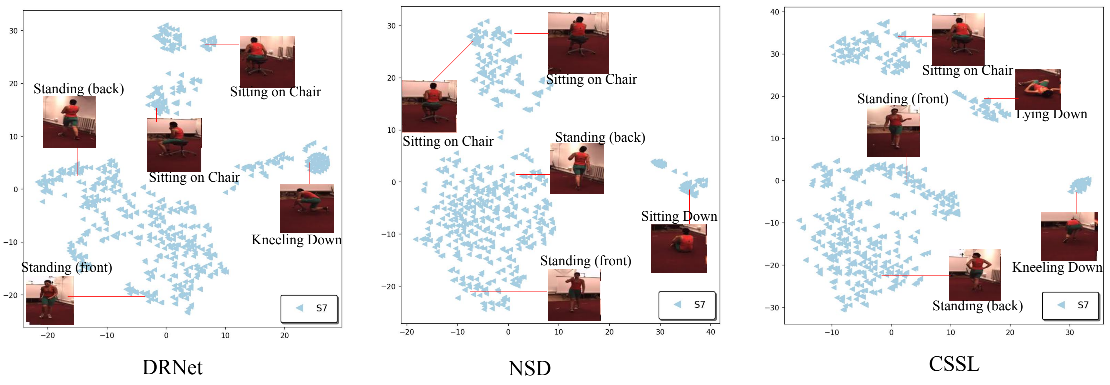


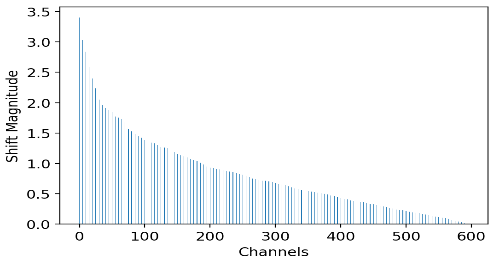
Figure 8.4 – **tSNE visualization of appearance codes.** The appearance codes of images from same subject S7 are clustered according to the action performed by the subject. This indicates that the appearance code still contains information about the pose. Best viewed in color and zoomed in.

8.5.2 Effect of the Pose Vector on Synthesized Images

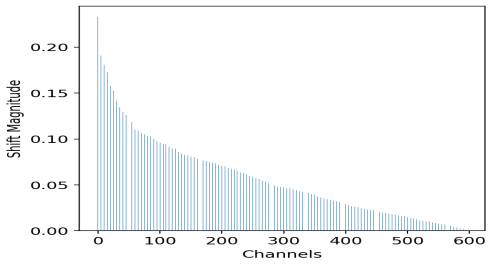
In this section, we study the impact of the pose vector on the synthesized images and further provide evidence of the presence of appearance information in the pose code. To this end, we identify channels encoding appearance information in the pose code. Our approach is based on the idea that two images depicting different subjects in similar poses should ideally have similar latent pose codes. The channels that have large differences therefore indicate the presence of appearance information.

To illustrate this, we use the two images shown in Figure 8.5(a) and plot the absolute difference between the corresponding pose codes obtained by NSDin Figure 8.5(b), ordering the channels by the magnitude of the difference. The latent pose indeed disagree in many channels. We define the probability of a channel to encode appearance information to be proportional to the absolute pose vector difference for that channel.

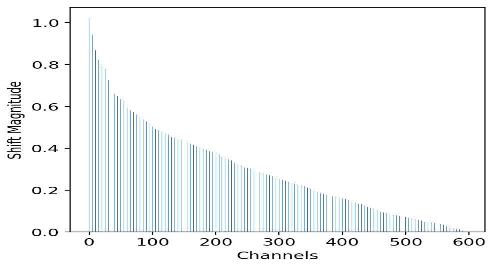
(a) Two images with similar poses but different appearances.



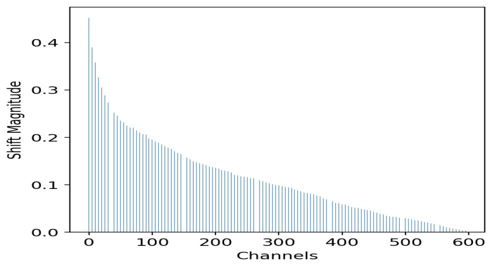
(b) Absolute difference between the pose codes sorted by magnitude for **NSD**.



(c) Absolute difference between the pose codes sorted by magnitude for **DRNet**.



(d) Absolute difference between the pose codes sorted by magnitude for **CSSL**.



(e) Absolute difference between the pose codes sorted by magnitude for **CSSL (DA)**.

Figure 8.5 – **Detecting appearance channels in the pose latent vector.** We take images depicting different subjects in a similar pose, for which we could expect the pose codes to be close. However, the latent pose vectors obtained by different networks contain channels with large differences, likely to encode appearance information.

Below, we then analyze the effect of altering the K channels with highest or lowest appearance probability.

To this end, we take two images A and B, as shown in the left and right ends of Figure 8.6, fix the appearance code as that of A. We then replace the channels with either the K lowest or highest appearance probability in the pose code of A with the corresponding values from the pose code of B. Note that all disentangled networks have a pose code of dimension 600, and therefore $K = 600$ means replacing all the channels of the pose vector.

As shown in Figure 8.6(a) for NSD, by replacing the $K = 500$ lowest appearance probability channels yields an image (highlighted with a red box) with A's appearance

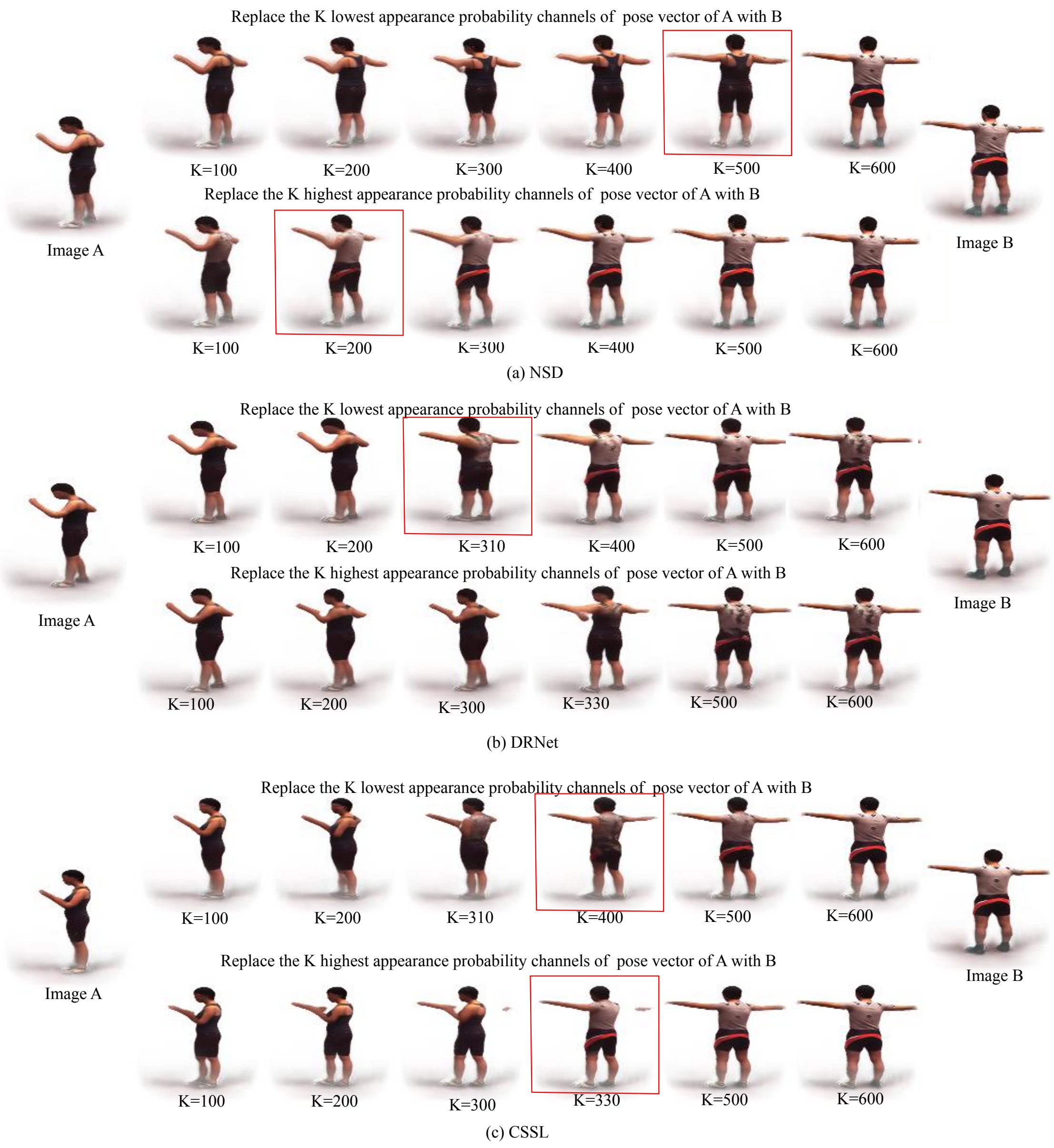

Replace the K lowest appearance probability channels of pose vector of A with B
K=100
K=200
K=300
K=400
K=500
K=600
Replace the K highest appearance probability channels of pose vector of A with B
K=100
K=200
K=300
K=400
K=500
K=600
Image A
Image B
(a) NSD
Replace the K lowest appearance probability channels of pose vector of A with B
K=100
K=200
K=310
K=400
K=500
K=600
Replace the K highest appearance probability channels of pose vector of A with B
K=100
K=200
K=300
K=330
K=500
K=600
Image A
Image B
(b) DRNet
Replace the K lowest appearance probability channels of pose vector of A with B
K=100
K=200
K=310
K=400
K=500
K=600
Replace the K highest appearance probability channels of pose vector of A with B
K=100
K=200
K=300
K=330
K=500
K=600
Image A
Image B
(c) CSSL

Figure 8.6 – **Influence of the pose code channels.** To synthesize the images in the middle portion of the figure, we take the appearance code corresponding to image A, and vary the pose code in two ways. Specifically, in the top (or bottom) portion of the figure, we replace the K channels with lowest (or highest) appearance probability with the corresponding ones from the pose code extracted from image B. (a) For NSD, replacing the $K = 500$ lowest appearance probability channels yields an image (highlighted with a red box) depicting B's pose and A's appearance. Similarly, replacing the $K = 200$ highest appearance probability channels produces B's appearance and A's pose. (b) We observe similar trends for DRNet and CSSL, although the separation of appearance and pose inside the pose code is not as clear as for NSD.

and B's pose. Furthermore, replacing the $K = 200$ highest appearance probability channels synthesizes an image with B's appearance and A's pose. Both these results indicate that the top 100-200 highest probability appearance channels in the pose code indeed encode the appearance information for NSD. It is worth noting that with $K = 600$ the image depicts both the pose and appearance of B, confirming our previous experiments in Figure 8.2.

Figure 8.6(b) and (c) for DRNet and CSSL shows the channels are not as clearly separated in pose and appearance ones in this method. Nevertheless, the pose codes still combines pose and appearance information.

8.6 Disentanglement w.r.t. the 3D Pose Regressor

The previous set of analyses have focused on the self-supervised representation learning networks themselves, evidencing that the latent pose vector is contaminated with appearance information. Here, we further investigate the disentanglement w.r.t. the supervised 3D human pose regressor, which takes the latent pose vector as input. Note that, since the 3D pose regressor is disassociated from the appearance vector at network level, studying the appearance and pose vector disentanglement in this context is not straightforward. Therefore, we consider the pose estimation network comprised of the self-supervised encoder and the supervised decoder as a standalone network and study the effects of the input image appearance on its 3D pose output. To this end, we introduce an adversarial perturbation strategy that explicitly focuses on modifying only the appearance information in the input image. Below, we first describe our attack framework, and then analyze its effects on the disentangled pose estimation networks.

8.6.1 Appearance-only Attack Framework

Our goal is to perturb only the subject's appearance in the input image; perturbing the image such that the subject's pose visually changes would of course make the pose regressor output a different pose but would not allow us to verify the disentanglement of pose and appearance. To enforce such a constraint on our perturbations, we follow a strategy that, intuitively, should constitute a weak attack and thus be favorable to the disentangled network. Specifically, we only perturb the latent appearance vector, which we combine with the *original* pose one to generate an adversarial image. We then extract a new latent pose vector from this image and predict the 3D human pose from it. If the pose regressor could discard the appearance information, it would thus not be affected by this perturbation.

As shown in Algorithm 2, we generate an adversarial image $\mathbf{I}_{adv}$ as using a generator network G. In practice, we take G to be either the disentangled network of interest or

Algorithm 2 Appearance-only attacks

```
Require: I: Input image, G: Pre-trained generator (with spatial transformer $G_s$, encoder $G_e$
    and decoder $G_d$), S: Target spatial transformer, E: Target encoder, D: Target image decoder,
    $\phi$: Target pose regressor
 1: $\mathbf{I}_c \leftarrow G_t(\mathbf{I})$, $[\tilde{\mathbf{h}}_a^0, \tilde{\mathbf{h}}_p^0] \leftarrow G_e(\mathbf{I}_c)$
 2: $\mathbf{I}_{adv}^0 = G_d(\tilde{\mathbf{h}}_a^0, \tilde{\mathbf{h}}_p^0), [\mathbf{h}_a^0, \mathbf{h}_p^0] \leftarrow E(S(\mathbf{I}_{adv}^0))$
 3: $[\mathbf{h}_a, \mathbf{h}_p] \leftarrow E(S(\mathbf{I}))$,
 4: $\mathbf{q} \leftarrow \phi(\mathbf{h}_p)), \text{error}_0 = \left\|\mathbf{q} - \phi(\mathbf{h}_p^0)\right\|^2$
 5: $i \leftarrow 1$
 6: while $\text{error}_i \leq$ min. error and $i \leq$ max. iterations do
 7:     $\mathbf{I}_{adv}^i \leftarrow G_d(\tilde{\mathbf{h}}_a^i, \tilde{\mathbf{h}}_p^0)$
 8:     $[\mathbf{h}_a^i, \mathbf{h}_p^i] \leftarrow E(S(\mathbf{I}_{adv}^i))$
 9:     $\text{error}_i \leftarrow \left\|\mathbf{q} - \phi(\mathbf{h}_p^i)\right\|^2$
10:     $\tilde{\mathbf{h}}_a^{i+1} \leftarrow \text{BackProp}\{\text{error}_i\}$
11:     $i \leftarrow i + 1$
12: end while
13:     return $\mathbf{I}_{adv} = \mathbf{I}_{adv}^i$
```

another disentangled network, and we will report results with both strategies. First, we pass the original input image to the generator's spatial transformer G_s and extract the cropped image $\mathbf{I}_c$ using the resulting bounding box. We then encode the cropped image $\mathbf{I}_c$ into an initial latent pose vector $\tilde{\mathbf{h}}_p^0$ and latent appearance vector $\tilde{\mathbf{h}}_a^0$ using the generator's encoder G_e. The combined latent vector $\tilde{\mathbf{h}} = [\tilde{\mathbf{h}}_a^0, \tilde{\mathbf{h}}_p^0]$ is then passed as input to the generator's decoder G_d, which outputs the reconstructed image $\tilde{\mathbf{I}}_c^0$ and a mask $\mathbf{M}^0$. The cropped output $\tilde{\mathbf{I}}_c^0$ is then combined with the pre-computed background image $\mathbf{B}$ to resynthesize an image $\mathbf{I}_{adv}^0$ at full resolution. This image then acts as input to the target pose estimation network, which encompasses an encoder E, that may differ from the generator one G_e, and a pose regressor. This forward pass produces an initial pose estimate $\phi(\mathbf{h}_p^0)$. Note that the output of the target network given $\mathbf{I}_{adv}^0$ as input has empirically a small mean per-joint position error (MPJPE) of around 20 mm with respect to the prediction $\mathbf{q}$ obtained from the original image $\mathbf{I}$. This is because, at this point, no attack has been performed.

To attack only the subject's appearance in the adversarial input, we fix the pose vector $\tilde{\mathbf{h}}_p = \tilde{\mathbf{h}}_p^0$ to generate images of depicting the subject in their original pose. Furthermore, we also fix the mask to its initial value $\mathbf{M} = \mathbf{M}^0$. We then compute an appearance-only perturbation by optimizing the latent appearance vector $\tilde{\mathbf{h}}_a$ in an iterative manner until it either achieves an MPJPE error with respect to the original prediction $\mathbf{q}^0$ higher than a threshold, or reaches a maximum number of iterations. Note that our previous set of experiments in Section 8.5 have evidenced that modifying the appearance vector does not change the observed subject's pose, which validates our use of the network's decoder to generate the appearance-modified image.

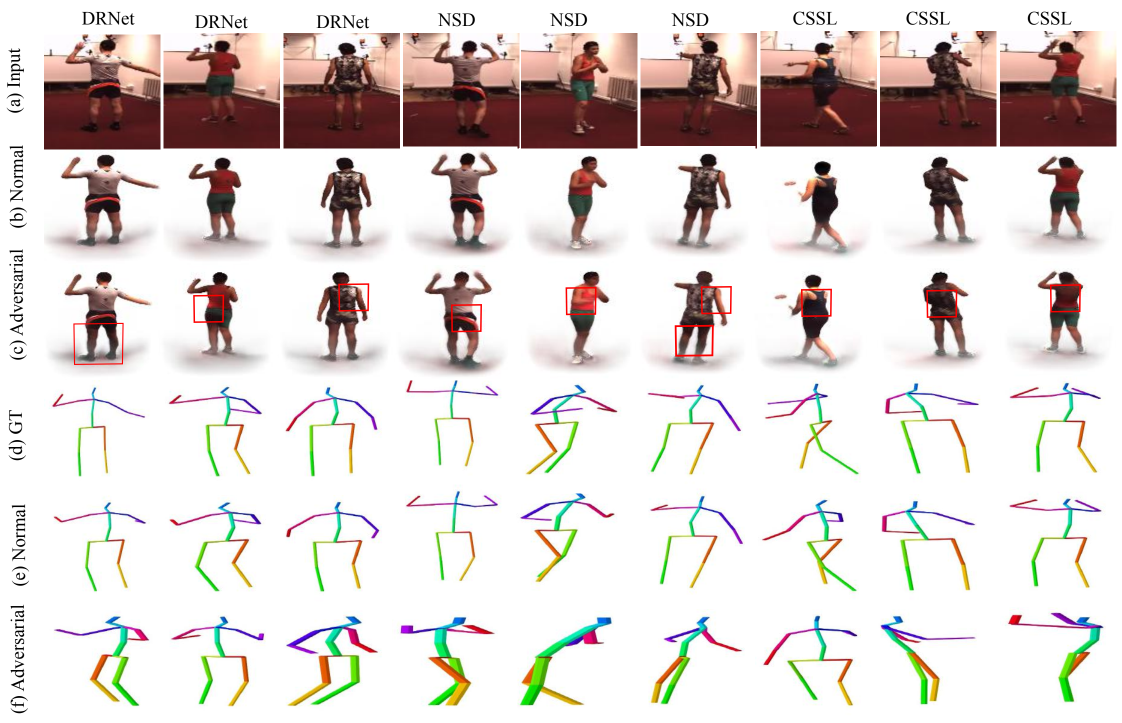


Figure 8.7 – **Apperance-only Attack Examples.** Given an input image **(a)** with ground-truth pose **(d)**, we first reconstruct **(b)** the images using a generator. By optimizing the latent appearance vector, we obtain an adversarial image **(c)** that aims to fool the pose regressor so that it outputs a 3D pose **(f)** that differs significantly from the original predictions **(e)**.

8.6.2 Appearance-only Attack Results

Qualitative Results. In Figure 8.7, we visualize the results of different models on the attacked images. For all disentangled representation frameworks, small changes in appearance produce wrong predictions. In particular, as shown in the third row, a small change in the shirt color leads to a completely different pose for all models. This demonstrates that the pose estimation network is dependent on the subject's appearance in the input image that its intermediate latent pose vector is not completely disentangled from appearance.

Quantitative Study. We provide the results of our appearance-only attacks in Table 8.1 using the network decoder as the generator. We report the MPJPE at the initial iteration and after the attack for each subject. Specifically, the initial error corresponds to the error between the predictions obtained from the original image $\mathbf{I}$ and from the synthesized image $\mathbf{I}^0_{adv}$, without any latent attack. It is around 21.8 mm on average. This shows that the generator faithfully reconstructs the input image and can therefore be employed to perform the attack. After the attack, the performance decrease across all the disentangled models In other words, all models are vulnerable to our appearance-based attacks and typically reach an MPJPE of at least 175 mm. This indicates that the latent pose vector $\mathbf{h}_p$ is not invariant to appearance changes and therefore that the appearance-pose disentanglement is not complete. We provide ablative study using the same NSD decoder as the generator for all disentangled networks in the Section 8.7.

To further evaluate quantitatively the sensitivity of a disentangled network to our appearance-only attacks, we computed three image-based metrics, Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSIM), and Mean Square Error (MSE), to compare the attacked images with those synthesized with the original framework. As shown in Table 8.2, the three metrics indicate that the images obtained by attacking DRNet are more similar to the original synthesized ones than those obtained by attacking NSD or CSSL. This suggests that DRNet can be attacked with smaller changes, and thus contains more appearance information in its pose vectors.

Altogether, our experiments evidence that disentangling pose and appearance in an unsupervised manner for 3D human pose estimation remains far from being solved. Our attacks thus provide a valuable testbed to valuate the effectiveness of future disentanglement-based frameworks.

Subject	NSD		DRNet		CSSL		Average	
	Initial	Final	Initial	Final	Initial	Final	Initial	Final
S1	21.0	179.7	23.9	169.7	21.5	176.9	21.6	174.2
S5	19.6	180.0	14.1	166.7	25.3	186.5	19.6	177.1
S6	22.3	179.8	23.5	177.9	26.8	196.7	23.4	184.7
S7	18.8	179.2	17.6	177.5	24.1	191.8	20.3	182.3
S8	16.8	178.6	21.7	198.9	30.5	186.9	23.0	187.8
Average	**19.7**	**179.5**	**20.2**	**177.5**	**25.6**	**207.5**	**21.8**	**176.8**

Table 8.1 – **MPJPE before and after our appearance-based attacks.** We report the results of three networks and observe that disentangled networks are vulnerable to our attacks.

Metric	NSD	DRNet	CSSL
SSIM↑	0.947	0.963	0.943
PSNR↑	24.65	26.45	24.37
MSE↓	0.012	0.007	0.013

Table 8.2 – Quantitative comparison of adversarial images with the original synthesized images. These number show that the images obtained by attacking DRNet are closer to the original synthesized ones, and thus that the DRNet pose vectors tend to contain more appearance information.

8.7 Discussion

Evaluating Disentanglement. Several methods [51, 58, 160] have been proposed for assessing the degree of disentanglement of latent variables. In particular, we report the two complementary state-of-the-art metrics of [160], Distance Correlation (DC) and Information over Bias (IOB) to evaluate disentanglement. DC is bounded in [0,1] and measures the correlation between the two latent spaces; IoB measures the amount of information from the input image that is encoded in a given latent space. In Table 8.3, we provide these metrics, averaged over 400 images, for the pose (P) and appearance

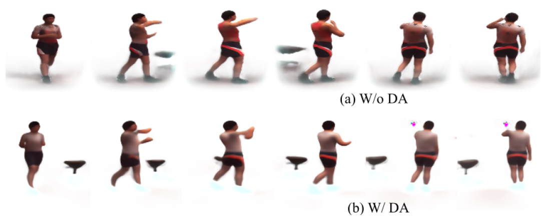


Figure 8.8 – **Synthesizing novel images with CSSL (DA).** As in Figure 8.2, we take S7's pose vector and S8's appearance one and synthesize novel images with CSSL, either without (top) or with (bottom) DA during training. The image synthesized with CSSL (DA) retain S8's appearance without residual red shirt color from S7.

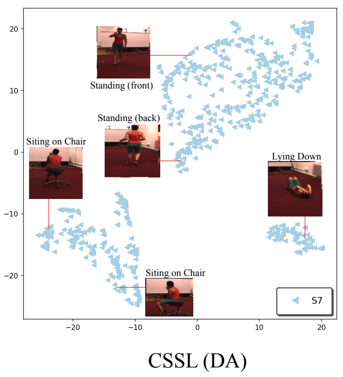


Figure 8.9 – **tSNE visualization of CSSL (DA) appearance codes.** The appearance codes of images from same subject are stilll clustered according to the action performed by the subject.

(A) latent spaces and for different disentanglement strategies. DC(A, P) contain large values indicating that the appearance and pose are correlated. Furthermore, the IOB(I, P) values are larger than the IOB(I, A), which suggests that the pose code encodes more input information than the appearance code. Note that DC(A, P) cannot be used as a standalone metric to interpret disentanglement because low values of DC can also indicate noise in one latent space. While DRNet achieves the best DC(A, P) score, its value of 0.90 IOB(I, A) suggests that the appearance code encodes minimal information. Although these metrics quantify disentanglement, they offer little understanding of the disentanglement issues, and IOB is difficult to interpret because it is unbounded and requires training an external decoder network whose optimal architecture is unknown. By contrast, our analyses enable a finer-grain understanding of the pose and appearance latent spaces of representation learning strategies for human pose estimation, and provide visual results that are easier to interpret.

Metric	NSD	DRNet	CSSL
DC(A, P)↓	0.88	0.59	0.77
IOB(I, A)↑	0.79	0.90	0.95
IOB(I, P)↑	1.15	1.08	1.29

Table 8.3 – Disentanglement-related metrics for the pose (P) and appearance (A) latent spaces extracted from an input image (I).

Metric	CSSL	CSSL(DA)
SSIM↑	0.943	0.926
PSNR↑	24.37	22.90
MSE↓	0.013	0.018

Table 8.4 – Quantitative comparison of adversarial images with original synthesized images. The images obtained with DA are less similar to original synthesized ones.

Does data-augmentation help to learn appearance-invariant features?Recently, powerful data augmentation (DA) strategies, such as AugMix [95], CutMix [296] and others [97, 299], have been proposed to improve the generalization power and robustness

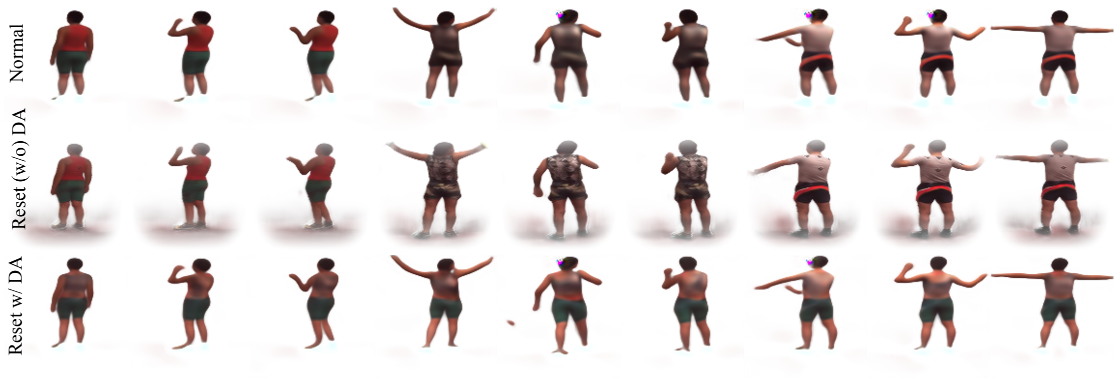


Figure 8.10 – **Zero appearance vectors with CSSL (DA).** In first row, we show the original image synthesized with CSSL. While, without DA (middle), the synthesized images obtained with a zero appearance vector retain the original subject's appearance, with DA (bottom), all the subjects have a similar the appearance. This suggests that DA helps to remove appearance information from the pose vectors.

of neural networks. Furthermore, classical adversarial training [117, 168] can be viewed as a form of data augmentation with adversarial images. Here, we therefore study if data augmentation constitutes a promising direction towards more effectively disentangling self-supervised 3D human pose estimation networks.

Since the network architectures we consider are much more complicated than the image recognition ones used in the above-mentioned DA works, we employ a simpler DA strategy consisting of augmenting the output of the spatial transformer with RGB jitter. We then re-run the analyses we presented before, focusing here on CSSL. Specifically, in Figure 8.8, we show the images synthesized when mixing S7's pose vector with S8's appearance. Note that, with DA, the images better retain the appearance of S8. Furthermore, in Figure 8.10, we show images obtained by making use of a zero appearance vector. With DA, all the synthesized images depict a similar subject appearance. Altogether, this suggests that DA helps the disentanglement process in CSSL, which is further confirmed by the DC(A, P) value that improves from 0.77 to 0.62. This value of 0.62 nonetheless still indicates a relatively high correlation between the latent spaces. To further analyze this, we computed a similar t-SNE plot as that of Figure 8.9, and observed that the actions are still clustered, evidencing that the appearance code still contains some pose information.

Similarly, we also ran our appearance-only attacks on the CSSL model trained with DA, and observed the attacks to remain successful, suggesting that the pose vector remains contaminated by appearance information. To evaluate quantitatively whether DA nonetheless improved this, we report the PSNR, SSIM, and MSE metrics between the attacked images and the original synthesized ones in Table 8.4. The values indicate that the images obtained by attacking the network without DA are more similar to the original

Subject	**AE**		**NSD**		**DRNet**		**CSSL**		**CSSL (DA)**		**Average**	
	Initial	Final	Initial	Final	Initial	Final	Initial	Final	Initial	Final	Initial	Final
S1	23.5	178.8	21.0	179.7	17.1	169.5	21.5	176.9	20.7	173.6	20.8	176.2
S5	21.3	177.2	19.6	180.0	14.1	166.7	18.2	176.5	19.4	172.6	18.3	175.1
S6	24.4	179.7	22.3	179.8	16.1	176.1	22.7	179.4	22.3	172.6	21.4	178.7
S7	21.4	179.3	18.8	179.2	14.8	166.1	18.4	176.8	18.5	172.3	18.3	175.3
S8	18.0	179.4	16.8	178.6	14.4	179.2	17.4	177.9	17.8	174.0	16.7	177.8
Average	**21.7**	**178.9**	**19.7**	**179.5**	**15.3**	**171.5**	**19.6**	**177.5**	**19.7**	**173.7**	**19.1**	**176.8**

Table 8.5 – **MPJPE before and after our appearance-based attack using the same NSD decoder.** We report the results of four networks and observe that disentangled networks are equally vulnerable to our attacks to the non-disentangled autoencoder.

synthesized ones. In other words, CSSL (DA) requires larger changes in the input image to attack the 3D pose regressor. Altogether, these results indicate that DA constitutes a promising direction to improve disentanglement, and we leave the development of more effective DA strategies as future work.

Semantic Attacks. To perform the attack, we optimize the spatially-replicated appearance vector of dimension $\mathbf{R}^{128\times16\times16}$, which gives more flexibility to the attacker. To this end, we employ Adam [124] with a learning rate of 0.02 and set the maximum number of iterations to 2K. Furthermore, we set the minimum error threshold to early stop the optimization to an MPJPE of 175mm. To evaluate our attacks, we use Subjects 1, 5, 6, 7, and 8 of Human3.6M [106], and downsample their sequences at every 20 frames to obtain 9.6K images. Out of them, we remove the images for which the MPJPE between the original and resynthesized image are more than 40 mm, which leaves us with 8K images.

Additional Results. We present the results of our appearance-based attacks using the same NSD decoder as generator in Table 8.5. We report the MPJPE at the initial iteration and after the attack for each subject. Specifically, the initial error corresponds to the error between the predictions obtained from the original image $\mathbf{I}$ and from the synthesized image $\mathbf{I}^0_{adv}$, without any latent attack. It is around 19.2 mm on average. This shows that the generator faithfully reconstructs the input image and can therefore be employed to perform the attack. After the attack, the performance decrease across all the disentangled models is similar to that of the AE baseline. In other words, all models including CSSL (DA) are vulnerable to our appearance-based attacks and typically reach an MPJPE of at least 175 mm. This indicates that the latent pose vector $\mathbf{h}_p$ is not invariant to appearance changes and therefore that the appearance-pose disentanglement is not complete. In Figures 8.11, 8.12, 8.13, and 8.14, we plot tSNE visualization of appearance code of S7 for NSD, DRNet, CSSL and CSSL (DA), respectively, which shows that appearance code is clustered according to action performed by subject.

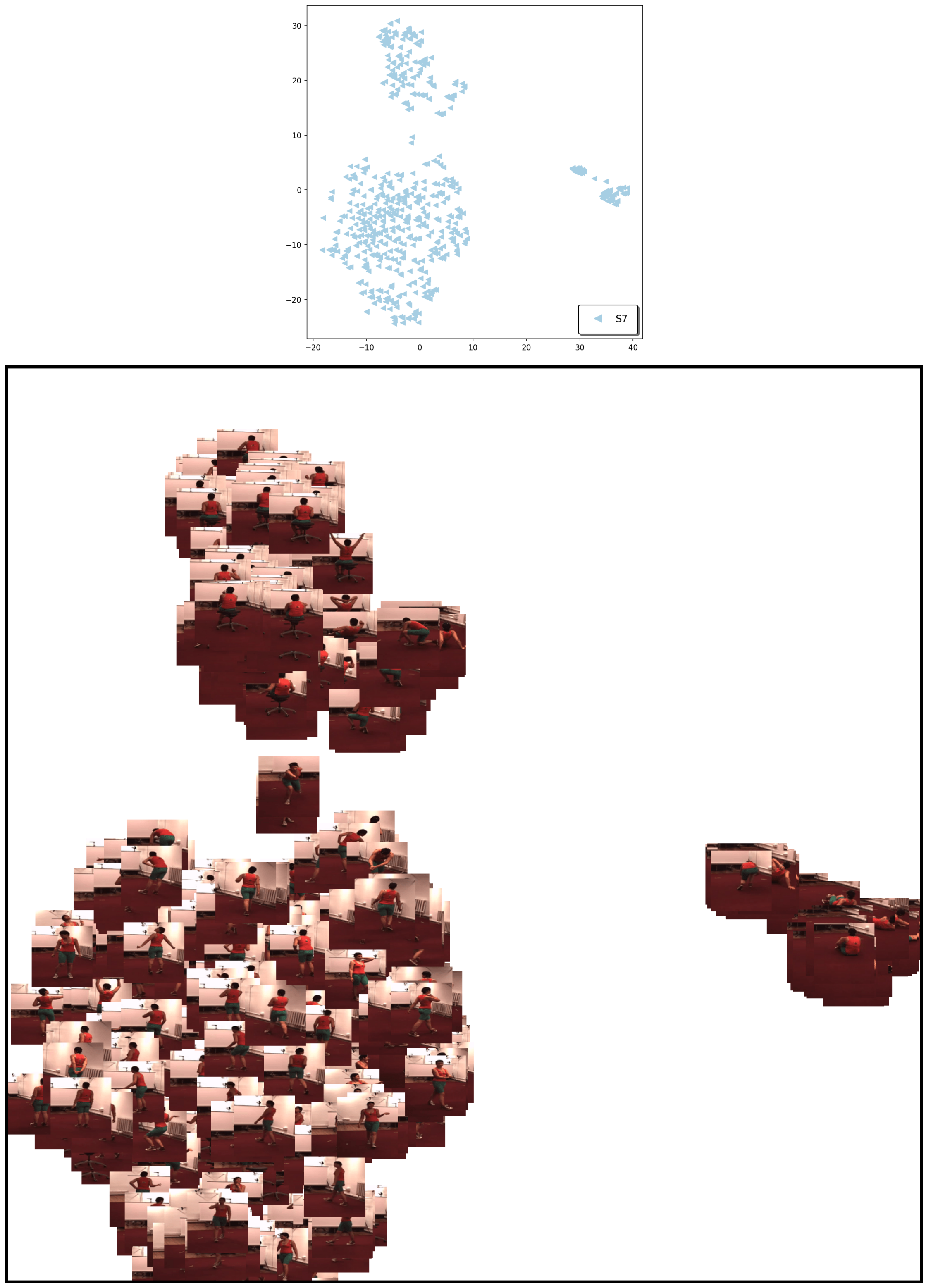


Figure 8.11 – **tSNE visualization of NSD appearance codes of S7.** The appearance codes of images from same subject are clustered according to the action performed by the subject. Best viewed in color and zoomed in.

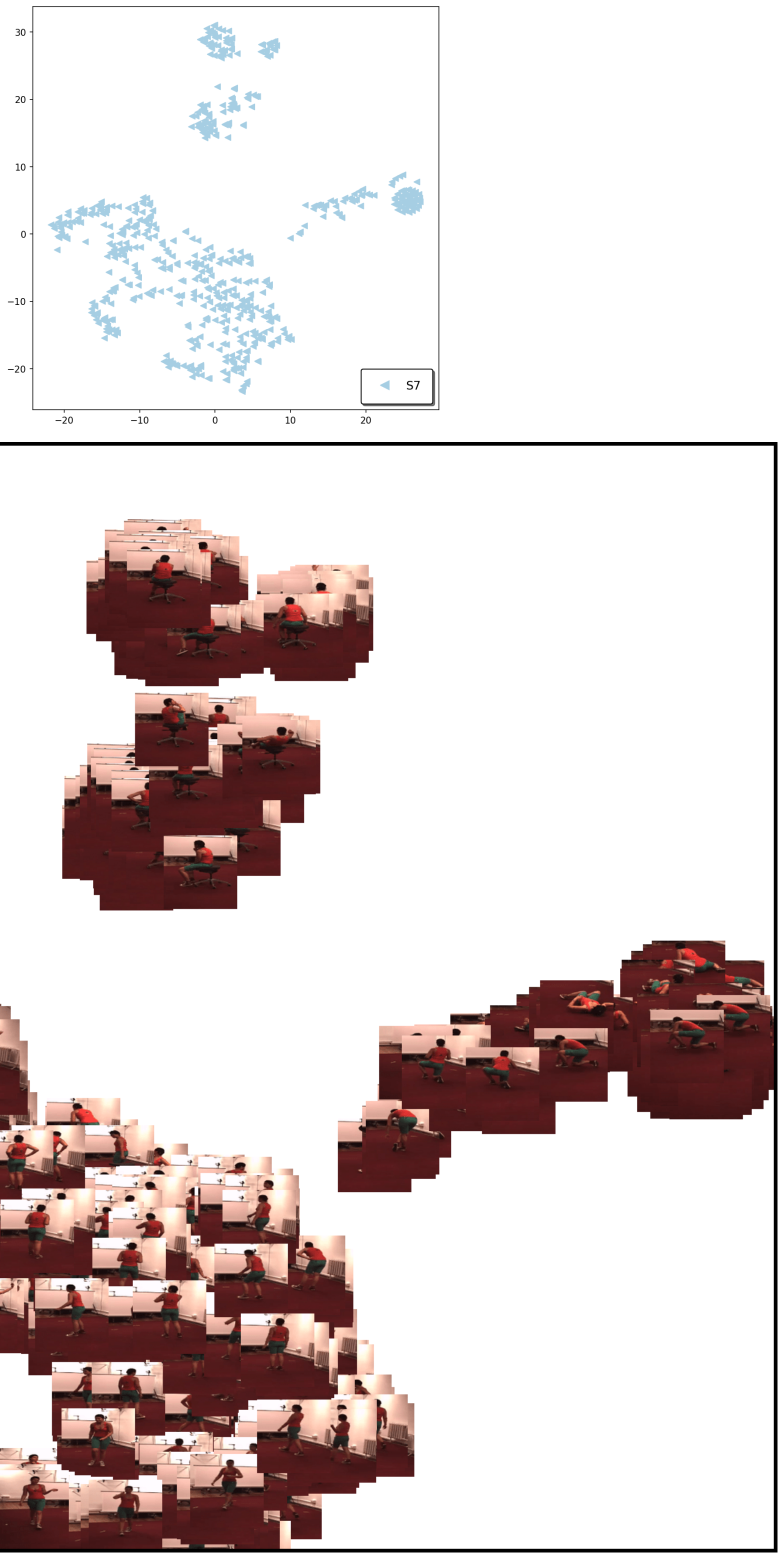
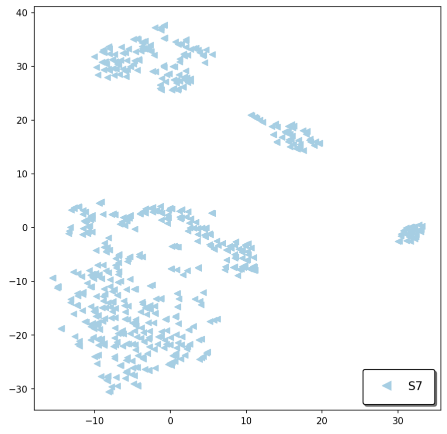


Figure 8.12 – **tSNE visualization of DRNet appearance codes of S7.** The appearance codes of images from same subject are clustered according to the action performed by the subject. Best viewed in color and zoomed in.



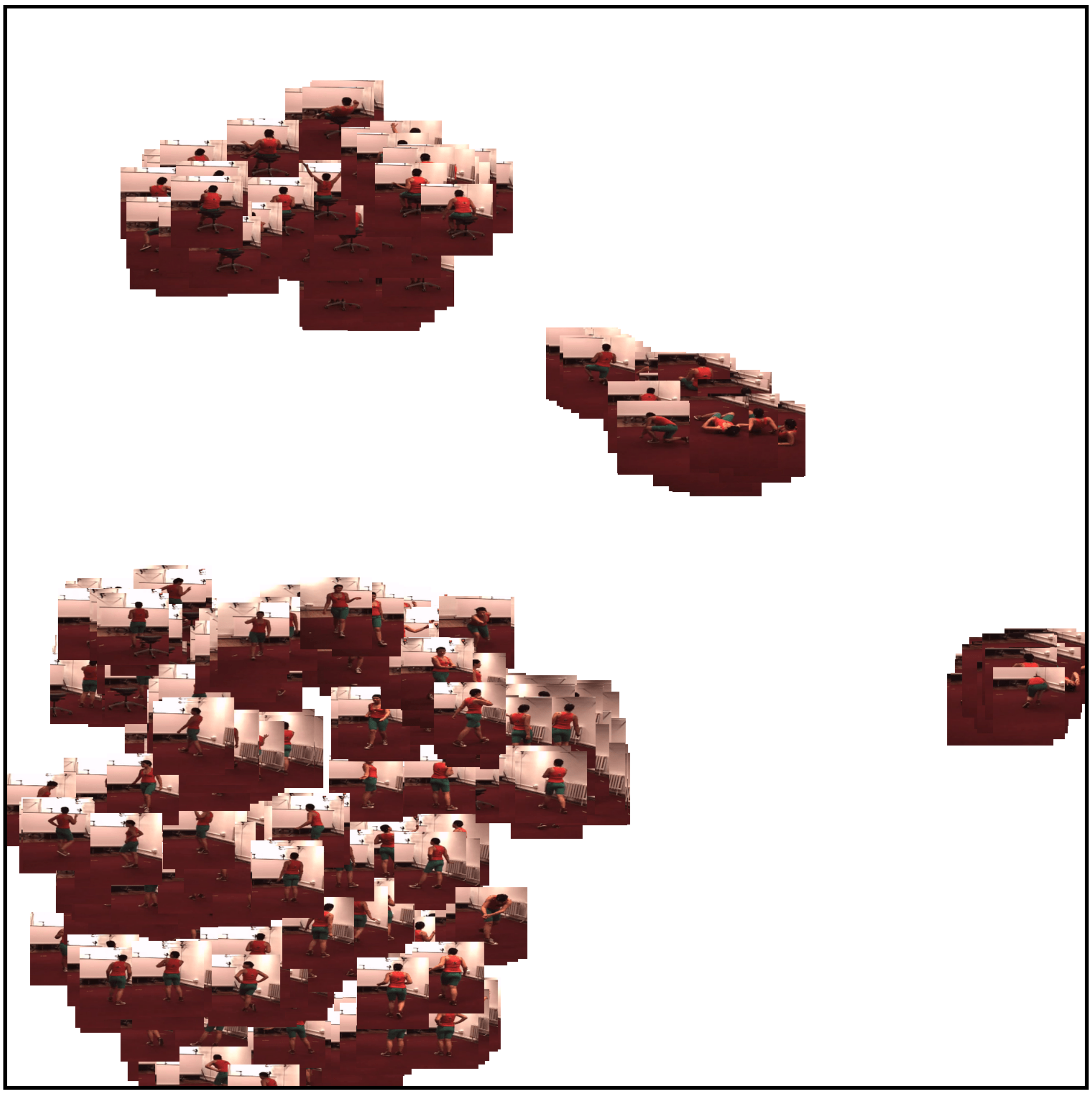
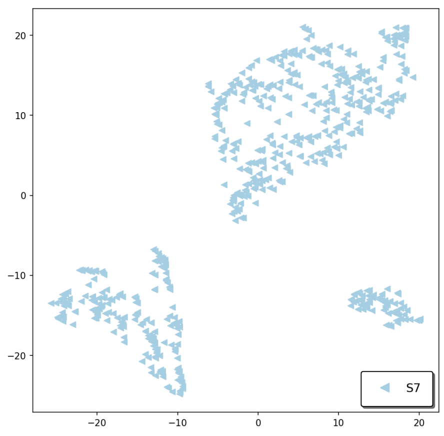


Figure 8.13 – **tSNE visualization of CSSL appearance codes of S7.** The appearance codes of images from same subject are clustered according to the action performed by the subject. Best viewed in color and zoomed in.



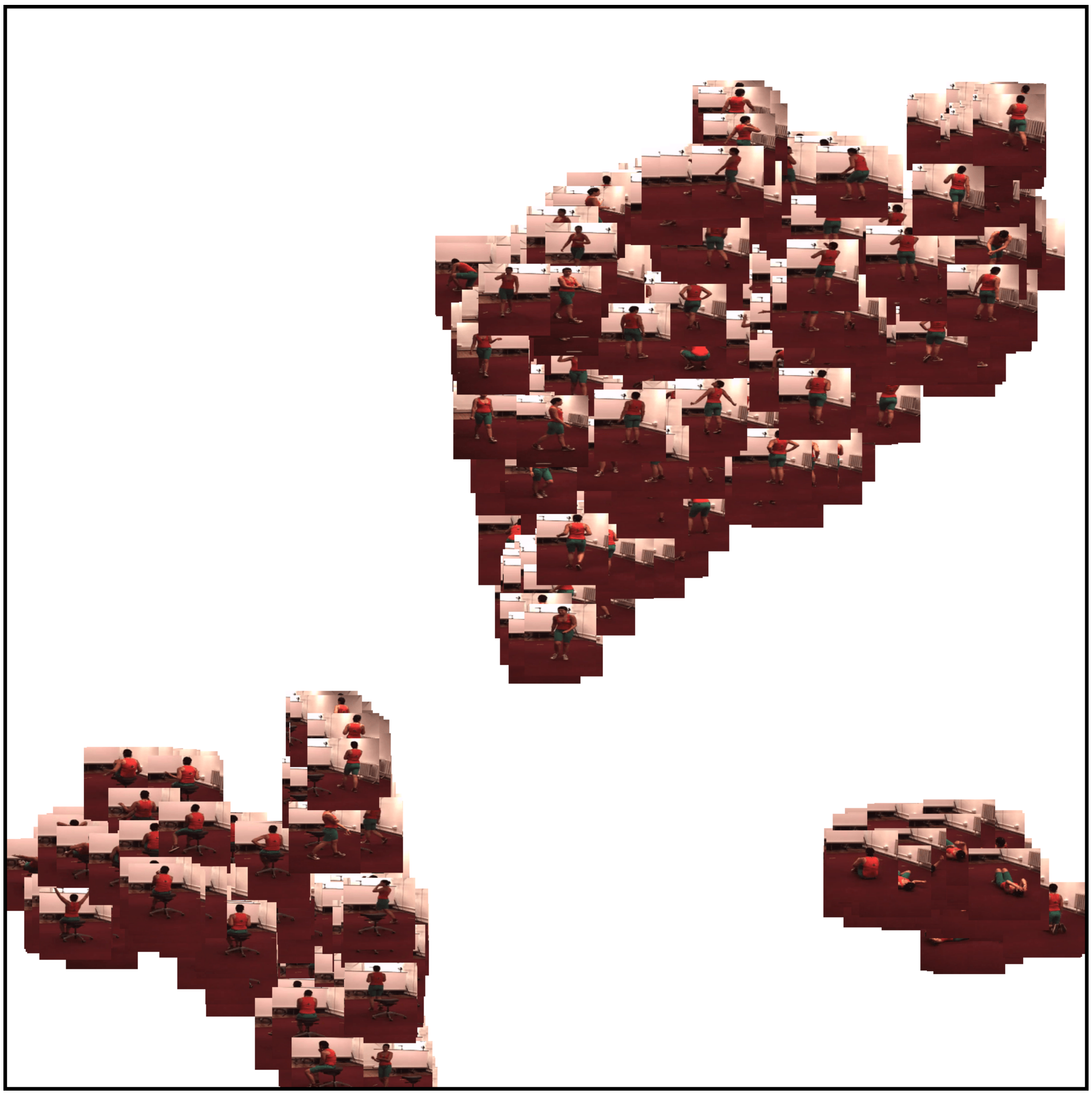


Figure 8.14 – **tSNE visualization of CSSL (DA) appearance codes of S7.** The appearance codes of images from same subject are clustered according to the action performed by the subject. Best viewed in color and zoomed in.

8.8 Conclusion

In this work, we have analyzed the latent vectors extracted by self-supervised disentangled networks for 3D human pose estimation. Specifically, we have studied the disentanglement of pose and appearance from the perspective of both the representation learning network, and the supervised 3D human pose regressor. In the former case, our analyses via diverse image synthesis strategies have evidenced that the state-of-the-art disentanglement-based representation learning networks do not truly disentangle pose from appearance, and in particular that the latent pose codes contain significant appearance information. In the latter, we have shown that disentanglement-based networks were not robust to appearance-only adversarial attacks, despite these attacks being designed to be as favorable as possible to the disentanglement-based frameworks. We believe that our analysis methodology and our semantic attacks will be beneficial to improve disentanglement-based representation learning in the future, and thus positively impact self-supervised 3D human pose estimation.

9 Conclusion

This thesis presented several approaches to understanding the underlying decision mechanism of DNN to adversarial examples. We have started with interpretable architectures in Chapter 2 that achieve superior performance using attention. We have then leveraged them to detect the adversarial examples in Chapter 3. Next, we analyzed the key reasons for the success of the adversarial attacks in the fine-grained setting and proposed an effective adversarial defense strategy in Chapter 4 by maximal separation of class-specific discrimination regions. We then moved to study the mechanism of adversarial attacks in different tasks. Notably, in Chapter 5, we observed the context in the segmentation network as the recipe for the attack, and in Chapter 6, we showed the existence of temporally transferable perturbations for the task of VOT. Furthermore, in Chapter 7, we extend the conventional notion of black-box attacks beyond architectures and show that disrupting the mid-level features allows us to attack unknown target architecture, target data, and task. Finally, in Chapter 8 we employ adversarial attacks to understand disentanglement of appearance and pose and show that the disentanglement of pose and appearance is far from complete.

Below, we summarize the individual chapters' contributions and then discuss the remaining limitations and potential directions for future research.

9.1 Summary

In Chapter 2, we have introduced attention-aware structured representation learning in the deep learning framework. Compared to attention-less counterparts, our approach yields a more interpretable structured representation by identifying the informative regions. Moreover, our method does not rely on additional supervision other than the target label and can be trained in an end-to-end manner.

In Chapter 3, we leverage BoW structured representation framework to detect the adversarial examples. We observe that the highest activated codeword is visually close to the clean image, whereas, for the adversarial inputs, it activates the codeword that is visually dissimilar to the input image. Thus, we cast the adversarial detection as an image similarity problem and achieve competitive performance in black-box and gray-box settings.

In Chapter 4, we propose a defense with an additional regularization of latent space. In particular, we force the discriminative regions away from each other while forcing the non-discriminative image regions to effectively not participate in the decision process. We test our proposed approach with white-box, transfer, and query-based attacks and consistently observe performance boost over baselines.

In Chapter 5, we have focused on studying the impact of using context for the task of semantic segmentation. While the context undeniably helps achieve superior performance over FCNs, this spectacularly makes the resulting network vulnerable to perturbations far from the object of interest. Nevertheless, we believe that this revelation is an important step in future research to design architectures with contextual dependencies that do not trade off robustness for accuracy.

In Chapter 6, we studied the universal perturbations for the task of visual object tracking that can efficiently fool black-box trackers. We show our framework relying only on the template makes the generator learn perturbations agnostic to the search environment and improves transferability to the unknown target tracker. Since the generator relies on the template, we believe it is forced to disrupt the object filters that form the core of the tracker backbones.

In Chapter 7, we proposed the zero-query transfer attacks, where the attacker has no knowledge about target architecture, target data, and even target task. These results suggest that in a real-world setting achieving complete robustness on the deployed model is not guaranteed even in the extreme setting. Further, it also indicates a common underlying phenomenon among DNNs that attackers can potentially exploit to break the model.

Chapter 8 shows that the disentanglement of the 3D pose estimation network is far from complete, and the latent pose code contains almost all appearance information. We come to this observation through semantic attacks, which are designed to change the appearance of the person against which the disentangled pose network is expected to be more robust. We also validate our argument with multiple image synthesis experiments which reveal the entanglement of appearance information inside the pose code. Thus, we extend the usefulness of adversarial attacks to go beyond fooling networks and study their latent representations.

9.2 Lessons Learnt and Retrospective Comments

The results presented in this thesis are the outcomes of several years of research work. Some lessons learned during these years and retrospective comments are shared below.

Attention-aware pooling. The idea came to me while studying the existing deep BoW architectures. My original goal was to learn an interpretable codebook to understand DNN's decisions, but I realized in the process that the performance of BoW architectures was not as impressive as standard architectures. Therefore, I focused on improving the performance of BoW architectures rather than on the interpretability part. Initially, I tried simple attention modules without label supervision, but most of them proved trivial solutions activating all regions after training. It was only after adding some supervision that I could solve this problem. Perhaps a better approach would have been to use the latest self-attention modules with the query and key-like modules to learn the attention map without label supervision.

Adversarial attacks detection. In the beginning of my Ph.D., adversarial attacks were not the focus of my thesis but rather centered about interpretable networks. I became interested in adversarial attacks' direction when I realized in the paper of LID [167] that distances to *cluster centers* are used to determine whether a given input is an adversarial sample or not. I started by experimenting with attacks on BoW architectures which led to the results presented in chapter 3. In retrospect, the best-performing adversarial attack detection [142] method also uses cluster centers in conjunction with an advanced distance Mahalanobis metric, thereby establising strong connection to prototype-based networks. Interestingly, I found in Chapter 4 that the proposed Attenional ProtoPNet improved detection rates by about 20% AUROC, suggesting that more efforts could be made in this direction to detect the attacks using BoW-based architectures.

Implementation. The work I did in my thesis covered a range of tasks such as image recognition, pose estimation, object detection, semantic segmentation, and object tracking. For each paper, it was difficult for me to get momentum and build a codebase in the beginning weeks. Among the fundamental habits I picked up and strongly recommend to budding PhD students are continuous logging of experiments with all hyper-parameters, writing modular codes, tabulating experimental results, tracking a large number of experiments, and, most importantly, understanding what the reasons are for unexpected outcomes. In my experience, some of the ideas presented in the thesis are merely accidental and abstract at the outset. The ideas begin to take shape and form with clear intuition only after a deeper and thorough analysis of the experimental results that are seemingly less than impressive.

9.3 Limitations and Future Work

Understanding the behavior of the DNNs to adversarial attacks has been the central theme of the thesis. However, this analysis has some limitations, and this thesis would be incomplete without discussing them. We also point to potential directions for future research.

Physically realizable attacks. While the digital adversarial patch attacks in Chapters 5 are relevant from the safety point of view, there are some caveats to replicating the attacks in the physical world. Notably, the learned perturbation should be effective to common image transformations and corruptions (i.e., non-adversarial) in the outside world. Further, simulating the attack environment takes considerable manual effort with many uncontrolloble dynamics. While a robust perturbation can be estimated by the Expectation Over Transformation [10] strategy, however, this drastically increases the computational complexity of learned perturbation. Furthermore, the effectiveness of adversarial attacks in systems that rely on multiple modalities such as vision, perception, motion, and voice needs to be thoroughly studied to understand the full practical significance.

Generalization-Robustness Tradeoff. It is a well-known phenomenon that adversarial training leads to a considerable drop in performance on clean samples. Notable extensions such as adversarial logit pairing [117] to match the clean and adversarial predictions, augmenting with large unlabelled data [2, 27, 94, 230], adversarial training with adaptive ϵ [11, 35, 48], curriculum-based adversarial training [23, 264, 305] to gradually increase the number of iterations for the attack, ensemble-based adversarial training [118, 198, 255, 288], etc., has improved the standard adversarial training but none of them achieve satisfactory performance. Moreover, computational complexity is a critical bottleneck for which many efficient adversarial training [234, 272] strategies have been proposed. Overall, adversarial training has its own limitations, and other directions beyond adversarial training should be studied to improve the robustness of DNNs.

Targeted attacks. A major drawback in this thesis is that most of our attacks, except the ones in Chapter 5, are focused on the untargeted setting, i.e., changing the label other than the original prediction. Although our untargeted attacks in Chapters 7 reveal the vulnerabilities of DNNs in black-box cases, it is more challenging to conduct a similar study on targeted black-box attacks. The mid-level filter bank, which contains low-level class-agnostic information, might not be optimal to achieve targeted attacks in black-box cases. However, the recent works [103, 147, 190, 313] show that it is quite possible to learn perturbation generators even when the labels of the source and target domain do not overlap. Nevertheless, the transferability of targeted attacks to a competitive level is an unsolved problem that requires deeper attention.

Beyond ℓ_∞ perturbations. The attacks discussed in Chapter 7 is focused on ℓ_∞-

bound attacks. Furthermore, the adversarial training in Chapter 4 also advocates ℓ_∞ perturbation during the generation of dynamic adversaries. However, it was shown that in [252] achieving robustness to one perturbation will trade-off with the robustness with other perturbation types such as ℓ_2 and ℓ_1. This poses a deeper question as to whether achieving true robustness to multiple perturbation types is fundamentally at odds and thus remains an open problem.

Meta-learning of adversarial perturbations. Another exciting direction is the meta-learning of perturbation generators on an ensemble of classifiers wherein each step, one can simulate a white-box and black-box kind of framework to enhance the transferability. Recent attempts [68, 294] along these lines show promising results in iterative setup. Nonetheless, these methods still requires a large number of ensemble models during the meta-training step and also leaves room for improvement particularly in the black-box setting.

Attacks beyond CNNs. In this thesis, we have limited our analysis and experiments on adversarial attacks to CNNs. Vision Transformers have recently seen a remarkable surge with an impressive performance over a wide range of applications. To this end, there has been emerging literature [17, 171, 192, 268] which studied the adversarial transferability betweens CNNs and Transformers. With Transformers most likely to advance the machine learning, understanding its robustness is an exciting area for future research.

Bibliography

[1] Painter by number. *https://www.kaggle.com/c/painter-by-numbers/data*, 2017.

[2] Jean-Baptiste Alayrac, Jonathan Uesato, Po-Sen Huang, Alhussein Fawzi, Robert Stanforth, and Pushmeet Kohli. Are labels required for improving adversarial robustness? *Advances in Neural Information Processing Systems*, 32, 2019.

[3] Jose M Alvarez and Mathieu Salzmann. Learning the number of neurons in deep networks. In *Advances in Neural Information Processing Systems*, pages 2270–2278, 2016.

[4] Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. Square attack: a query-efficient black-box adversarial attack via random search. In *European Conference on Computer Vision*, pages 484–501. Springer, 2020.

[5] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5297–5307, 2016.

[6] Relja Arandjelovic and Andrew Zisserman. All about vlad. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 1578–1585. IEEE, 2013.

[7] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[8] Anurag Arnab, Ondrej Miksik, and Philip HS Torr. On the robustness of semantic segmentation models to adversarial attacks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 888–897, 2018.

[9] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International conference on machine learning*, pages 274–283. PMLR, 2018.

[10] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. In *International conference on machine learning*, pages 284–293, 2018.

[11] Yogesh Balaji, Tom Goldstein, and Judy Hoffman. Instance adaptive adversarial training: Improved accuracy tradeoffs in neural nets. 2019.

[12] Farhan Baluch and Laurent Itti. Mechanisms of top-down attention. *Trends in neurosciences*, 34(4):210–224, 2011.

[13] Shumeet Baluja and Ian Fischer. Adversarial transformation networks: Learning to generate adversarial examples. *arXiv preprint arXiv:1703.09387*, 2017.

[14] Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-convolutional siamese networks for object tracking. In *European conference on computer vision*, pages 850–865. Springer, 2016.

[15] Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. Learning discriminative model prediction for tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6182–6191, 2019.

[16] Anand Bhattad, Min Jin Chong, Kaizhao Liang, Bo Li, and DA Forsyth. Unrestricted adversarial examples via semantic manipulation. In *International Conference on Learning Representations*, 2019.

[17] Srinadh Bhojanapalli, Ayan Chakrabarti, Daniel Glasner, Daliang Li, Thomas Unterthiner, and Andreas Veit. Understanding robustness of transformers for image classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10231–10241, 2021.

[18] Cenk Bircanoglu. Painter by number. *https://www.kaggle.com/cenkbircanoglu/comic-books-classification*, 2017.

[19] David S Bolme, J Ross Beveridge, Bruce A Draper, and Yui Man Lui. Visual object tracking using adaptive correlation filters. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 2544–2550. IEEE, 2010.

[20] Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. In *International Conference on Learning Representations*, 2018.

[21] Tom B Brown and Dandelion Mané. Aurko roy, martín abadi, and justin gilmer. *Adversarial patch. CoRR, abs/1712.09665*, 2017.

[22] Yaroslav Bulatov. notmnist dataset. 2011.

[23] Qi-Zhi Cai, Chang Liu, and Dawn Song. Curriculum adversarial training. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 3740–3747, 2018.

[24] Michael Calonder, Vincent Lepetit, Mustafa Ozuysal, Tomasz Trzcinski, Christoph Strecha, and Pascal Fua. Brief: Computing a local binary descriptor very fast. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1281–1298, 2012.

[25] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 3–14. ACM, 2017.

[26] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. IEEE, 2017.

[27] Yair Carmon, Aditi Raghunathan, Ludwig Schmidt, John C Duchi, and Percy S

Liang. Unlabeled data improves adversarial robustness. *Advances in Neural Information Processing Systems*, 32, 2019.

[28] Aditya Chattopadhay, Anirban Sarkar, Prantik Howlader, and Vineeth N Balasubramanian. Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 839–847. IEEE, 2018.

[29] Chaofan Chen, Oscar Li, Daniel Tao, Alina Barnett, Cynthia Rudin, and Jonathan K Su. This looks like that: deep learning for interpretable image recognition. In *Advances in Neural Information Processing Systems*, pages 8928–8939, 2019.

[30] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 15–26, 2017.

[31] Xuesong Chen, Xiyu Yan, Feng Zheng, Yong Jiang, Shu-Tao Xia, Yong Zhao, and Rongrong Ji. One-shot adversarial attacks on visual tracking with dual attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10176–10185, 2020.

[32] Yue Chen, Yalong Bai, Wei Zhang, and Tao Mei. Destruction and construction learning for fine-grained image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5157–5166, 2019.

[33] Zedu Chen, Bineng Zhong, Guorong Li, Shengping Zhang, and Rongrong Ji. Siamese box adaptive network for visual tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6668–6677, 2020.

[34] Minhao Cheng, Thong Le, Pin-Yu Chen, Huan Zhang, Jinfeng Yi, and Cho-Jui Hsieh. Query-efficient hard-label black-box attack: An optimization-based approach. In *International Conference on Learning Representation (ICLR)*, 2019.

[35] Minhao Cheng, Qi Lei, Pin-Yu Chen, Inderjit Dhillon, and Cho-Jui Hsieh. Cat: Customized adversarial training for improved robustness. *arXiv preprint arXiv:2002.06789*, 2020.

[36] Kyunghyun Cho et al. Retrieval-augmented convolutional neural networks against adversarial examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11563–11571, 2019.

[37] Jun-Ho Choi, Huan Zhang, Jun-Hyuk Kim, Cho-Jui Hsieh, and Jong-Seok Lee. Evaluating robustness of deep image super-resolution against adversarial attacks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 303–311, 2019.

[38] Mircea Cimpoi, Subhransu Maji, and Andrea Vedaldi. Deep filter banks for texture recognition and segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 3828–3836. IEEE, 2015.

[39] Moustapha M Cisse, Yossi Adi, Natalia Neverova, and Joseph Keshet. Houdini: Fooling deep structured visual and speech recognition models with adversarial examples. *Advances in neural information processing systems*, 30, 2017.

[40] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

[41] Francesco Croce and Matthias Hein. Sparse and imperceivable adversarial attacks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4724–4732, 2019.

[42] Francesco Croce and Matthias Hein. Minimally distorted adversarial examples with a fast adaptive boundary attack. In *International Conference on Machine Learning*, pages 2196–2205. PMLR, 2020.

[43] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International conference on machine learning*, pages 2206–2216. PMLR, 2020.

[44] Yin Cui, Feng Zhou, Jiang Wang, Xiao Liu, Yuanqing Lin, and Serge Belongie. Kernel pooling for convolutional neural networks. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[45] Martin Danelljan, Gustav Hager, Fahad Shahbaz Khan, and Michael Felsberg. Learning spatially regularized correlation filters for visual tracking. In *Proceedings of the IEEE international conference on computer vision*, pages 4310–4318, 2015.

[46] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[47] Emily L Denton and Vighnesh Birodkar. Unsupervised learning of disentangled representations from video. In *NIPS*, 2017.

[48] Gavin Weiguang Ding, Yash Sharma, Kry Yik Chau Lui, and Ruitong Huang. Mma training: Direct input space margin maximization through adversarial training. In *International Conference on Learning Representations*, 2019.

[49] Gavin Weiguang Ding, Luyu Wang, and Xiaomeng Jin. Advertorch v0. 1: An adversarial robustness toolbox based on pytorch. *arXiv preprint arXiv:1902.07623*, 2019.

[50] Mandar Dixit, Si Chen, Dashan Gao, Nikhil Rasiwasia, and Nuno Vasconcelos. Scene classification with semantic fisher vectors. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 2974–2983. IEEE, 2015.

[51] Kien Do and Truyen Tran. Theory and evaluation metrics for learning disentangled representations. In *International Conference on Learning Representations*, 2019.

[52] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9185–9193, 2018.

[53] Yinpeng Dong, Tianyu Pang, Hang Su, and Jun Zhu. Evading defenses to transferable adversarial examples by translation-invariant attacks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages

4312–4321, 2019.

[54] Yinpeng Dong, Hang Su, Baoyuan Wu, Zhifeng Li, Wei Liu, Tong Zhang, and Jun Zhu. Efficient decision-based black-box adversarial attacks on face recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7714–7722, 2019.

[55] Yinpeng Dong, Hang Su, Baoyuan Wu, Zhifeng Li, Wei Liu, Tong Zhang, and Jun Zhu. Efficient decision-based black-box adversarial attacks on face recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[56] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[57] Abhimanyu Dubey, Laurens van der Maaten, Zeki Yalniz, Yixuan Li, and Dhruv Mahajan. Defense against adversarial images using web-scale nearest-neighbor search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8767–8776, 2019.

[58] Cian Eastwood and Christopher KI Williams. A framework for the quantitative evaluation of disentangled representations. In *6th International Conference on Learning Representations*, 2018.

[59] Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. *University of Montreal*, 1341(3):1, 2009.

[60] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge 2007 (voc2007) results. 2007.

[61] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[62] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1625–1634, 2018.

[63] Heng Fan, Liting Lin, Fan Yang, Peng Chu, Ge Deng, Sijia Yu, Hexin Bai, Yong Xu, Chunyuan Liao, and Haibin Ling. Lasot: A high-quality benchmark for large-scale single object tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5374–5383, 2019.

[64] Reuben Feinman, Ryan R Curtin, Saurabh Shintre, and Andrew B Gardner. Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410*, 2017.

[65] Pedro Felzenszwalb, David McAllester, and Deva Ramanan. A discriminatively trained, multiscale, deformable part model. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[66] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Ob-

ject detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1627–1645, 2010.

[67] Pedro F Felzenszwalb and Daniel P Huttenlocher. Pictorial structures for object recognition. *International journal of computer vision*, 61(1):55–79, 2005.

[68] Weiwei Feng, Baoyuan Wu, Tianzhu Zhang, Yong Zhang, and Yongdong Zhang. Meta-attack: Class-agnostic and model-agnostic physical adversarial attack. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7787–7796, 2021.

[69] Volker Fischer, Mummadi Chaithanya Kumar, Jan Hendrik Metzen, and Thomas Brox. Adversarial examples for semantic image segmentation. *arXiv preprint arXiv:1703.01101*, 2017.

[70] Yoav Freund, Robert E Schapire, et al. Experiments with a new boosting algorithm. In *International Conference on Machine Learning (ICML)*, volume 96, pages 148–156. Bari, Italy, 1996.

[71] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3146–3154, 2019.

[72] Jianlong Fu, Heliang Zheng, and Tao Mei. Look closer to see better: Recurrent attention convolutional neural network for fine-grained image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[73] Yang Gao, Oscar Beijbom, Ning Zhang, and Trevor Darrell. Compact bilinear pooling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 317–326, 2016.

[74] Rohit Girdhar and Deva Ramanan. Attentional pooling for action recognition. In *Advances in Neural Information Processing Systems (NIPS)*, pages 33–44, 2017.

[75] Rohit Girdhar, Deva Ramanan, Abhinav Gupta, Josef Sivic, and Bryan Russell. Actionvlad: Learning spatio-temporal aggregation for action classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 971–980, 2017.

[76] Josep M Gonfaus, Xavier Boix, Joost Van de Weijer, Andrew D Bagdanov, Joan Serrat, and Jordi Gonzalez. Harmony potentials for joint classification and segmentation. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 3280–3287. IEEE, 2010.

[77] Yunchao Gong, Liwei Wang, Ruiqi Guo, and Svetlana Lazebnik. Multi-scale orderless pooling of deep convolutional activation features. In *European conference on computer vision*, pages 392–407. Springer, 2014.

[78] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[79] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples (2014). *International conference on Learning Representations*, 2015.

[80] Gaurav Goswami, Nalini Ratha, Akshay Agarwal, Richa Singh, and Mayank Vatsa. Unravelling robustness of deep learning based face recognition against adversarial attacks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[81] Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick McDaniel. On the (statistical) detection of adversarial examples. *arXiv preprint arXiv:1702.06280*, 2017.

[82] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5767–5777, 2017.

[83] Dongyan Guo, Jun Wang, Ying Cui, Zhenhua Wang, and Shengyong Chen. Siamcar: Siamese fully convolutional classification and regression for visual tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6269–6277, 2020.

[84] Qing Guo, Xiaofei Xie, Felix Juefei-Xu, Lei Ma, Zhongguo Li, Wanli Xue, Wei Feng, and Yang Liu. Spark: Spatial-aware online incremental attack against visual tracking. *arXiv preprint arXiv:1910.08681*, 2019.

[85] Ying Guo, Xingxing Wei, Guoqiu Wang, and Bo Zhang. Meaningful adversarial stickers for face recognition in physical world. *arXiv preprint arXiv:2104.06728*, 2021.

[86] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *null*, pages 1735–1742. IEEE, 2006.

[87] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Hypercolumns for object segmentation and fine-grained localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 447–456, 2015.

[88] Jamie Hayes and George Danezis. Learning universal adversarial perturbations with generative models. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 43–49. IEEE, 2018.

[89] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[90] Xuming He, Richard S Zemel, and Miguel Á Carreira-Perpiñán. Multiscale conditional random fields for image labeling. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 2, pages II–II. IEEE, 2004.

[91] Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28, 1998.

[92] Jan Hendrik Metzen, Mummadi Chaithanya Kumar, Thomas Brox, and Volker Fischer. Universal adversarial perturbations against semantic image segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2755–2764, 2017.

[93] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and

out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016.

[94] Dan Hendrycks, Mantas Mazeika, Saurav Kadavath, and Dawn Song. Using self-supervised learning can improve model robustness and uncertainty. *Advances in Neural Information Processing Systems*, 32, 2019.

[95] Dan Hendrycks, Norman Mu, Ekin Dogus Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. Augmix: A simple data processing method to improve robustness and uncertainty. In *International Conference on Learning Representations*, 2019.

[96] Sina Honari, Victor Constantin, Helge Rhodin, Mathieu Salzmann, and Pascal Fua. Unsupervised learning on monocular videos for 3d human pose estimation. *arXiv preprint arXiv:2012.01511*, 2020.

[97] Minui Hong, Jinwoo Choi, and Gunhee Kim. Stylemix: Separating content and style for enhanced data augmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14862–14870, 2021.

[98] Guosheng Hu, Yongxin Yang, Dong Yi, Josef Kittler, William Christmas, Stan Z Li, and Timothy Hospedales. When face recognition meets with deep learning: an evaluation of convolutional neural networks for face recognition. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 142–150, 2015.

[99] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[100] Lianghua Huang, Xin Zhao, and Kaiqi Huang. Got-10k: A large high-diversity benchmark for generic object tracking in the wild. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[101] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[102] Nathan Inkawhich, Kevin J Liang, Lawrence Carin, and Yiran Chen. Transferable perturbations of deep feature distributions. *arXiv preprint arXiv:2004.12519*, 2020.

[103] Nathan Inkawhich, Kevin J Liang, Jingyang Zhang, Huanrui Yang, Hai Li, and Yiran Chen. Can targeted adversarial examples transfer when the source and target models have no label space overlap? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 41–50, 2021.

[104] Nathan Inkawhich, Wei Wen, Hai Helen Li, and Yiran Chen. Feature space perturbations yield more transferable adversarial examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7066–7074, 2019.

[105] Catalin Ionescu, Joao Carreira, and Cristian Sminchisescu. Iterated second-order label sensitive pooling for 3d human pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1661–1668, 2014.

[106] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3.

6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE transactions on pattern analysis and machine intelligence*, 36(7):1325–1339, 2013.

[107] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[108] Naman Jain, Sahil Shah, Abhishek Kumar, and Arjun Jain. On the robustness of human pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 29–38, 2019.

[109] Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. Aggregating local descriptors into a compact image representation. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3304–3311. IEEE, 2010.

[110] Herve Jegou, Florent Perronnin, Matthijs Douze, Jorge Sánchez, Patrick Perez, and Cordelia Schmid. Aggregating local image descriptors into compact codes. *IEEE transactions on pattern analysis and machine intelligence*, 34(9):1704–1716, 2012.

[111] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):221–231, 2012.

[112] Shuai Jia, Yibing Song, Chao Ma, and Xiaokang Yang. Iou attack: Towards temporally coherent black-box adversarial attack for visual object tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6709–6718, 2021.

[113] Yunhan Jia, Yantao Lu, Junjie Shen, Qi Alfred Chen, Zhenyu Zhong, and Tao Wei. Fooling detection alone is not enough: First adversarial attack against multiple object tracking. *arXiv preprint arXiv:1905.11026*, 2019.

[114] Albert Jimenez, Jose M Alvarez, and Xavier Giro-i Nieto. Class-weighted convolutional features for visual instance search. *British Media Vision Conference (BMVC)*, 2017.

[115] Hanbyul Joo, Hao Liu, Lei Tan, Lin Gui, Bart Nabbe, Iain Matthews, Takeo Kanade, Shohei Nobuhara, and Yaser Sheikh. Panoptic studio: A massively multiview system for social motion capture. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3334–3342, 2015.

[116] Ameya Joshi, Amitangshu Mukherjee, Soumik Sarkar, and Chinmay Hegde. Semantic adversarial attacks: Parametric transformations that fool deep classifiers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4773–4783, 2019.

[117] Harini Kannan, Alexey Kurakin, and Ian Goodfellow. Adversarial logit pairing. *arXiv preprint arXiv:1803.06373*, 2018.

[118] Sanjay Kariyappa and Moinuddin K Qureshi. Improving adversarial robustness of ensembles with diversity training. *arXiv preprint arXiv:1901.09981*, 2019.

[119] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural

networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

[120] Valentin Khrulkov and Ivan Oseledets. Art of singular vectors and universal adversarial perturbations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8562–8570, 2018.

[121] Hamed Kiani Galoogahi, Terence Sim, and Simon Lucey. Multi-channel correlation filters. In *Proceedings of the IEEE international conference on computer vision*, pages 3072–3079, 2013.

[122] Hamed Kiani Galoogahi, Terence Sim, and Simon Lucey. Correlation filters with limited boundaries. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4630–4638, 2015.

[123] KangGeon Kim, Zhenheng Yang, Iacopo Masi, Ramakant Nevatia, and Gerard Medioni. Face and body association for video-based face recognition. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 39–48. IEEE, 2018.

[124] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR*, 2015.

[125] Jan J Koenderink and Andrea J Van Doorn. The structure of locally orderless images. *International Journal of Computer Vision*, 31(2-3):159–168, 1999.

[126] Pushmeet Kohli, Philip HS Torr, et al. Robust higher order potentials for enforcing label consistency. *International Journal of Computer Vision*, 82(3):302–324, 2009.

[127] Shu Kong and Charless Fowlkes. Low-rank bilinear pooling for fine-grained classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 365–374, 2017.

[128] Peter Kontschieder, Madalina Fiterau, Antonio Criminisi, and Samuel Rota Bulo. Deep neural decision forests. In *Proceedings of the IEEE international conference on computer vision*, pages 1467–1475, 2015.

[129] Tom Koren, Lior Talker, Michael Dinerstein, and Roy J Jevnisek. Consistent semantic attacks on optical flow. *arXiv preprint arXiv:2111.08485*, 2021.

[130] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International Conference on Machine Learning*, pages 3519–3529. PMLR, 2019.

[131] Philipp Krähenbühl and Vladlen Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *Advances in neural information processing systems*, pages 109–117, 2011.

[132] Jonathan Krause, Hailin Jin, Jianchao Yang, and Li Fei-Fei. Fine-grained recognition without part annotations. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 5546–5555. IEEE, 2015.

[133] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 554–561, 2013.

[134] Matej Kristan, Ales Leonardis, Jiri Matas, Michael Felsberg, Roman Pflugfelder, Luka Cehovin Zajc, Tomas Vojir, Goutam Bhat, Alan Lukezic, Abdelrahman

Eldesokey, et al. The sixth visual object tracking vot2018 challenge results. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0, 2018.

[135] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[136] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[137] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

[138] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016.

[139] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[140] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[141] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[142] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *Advances in Neural Information Processing Systems*, pages 7167–7177, 2018.

[143] Kuan-Hui Lee and Jenq-Neng Hwang. On-road pedestrian tracking across multiple driving recorders. *IEEE Transactions on Multimedia*, 17(9):1429–1438, 2015.

[144] Bo Li, Wei Wu, Qiang Wang, Fangyi Zhang, Junliang Xing, and Junjie Yan. Siamrpn++: Evolution of siamese visual tracking with very deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4282–4291, 2019.

[145] Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu. High performance visual tracking with siamese region proposal network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8971–8980, 2018.

[146] Li-Jia Li, Hao Su, Li Fei-Fei, and Eric P Xing. Object bank: A high-level image representation for scene classification & semantic feature sparsification. In *Advances in neural information processing systems*, pages 1378–1386, 2010.

[147] Maosen Li, Cheng Deng, Tengjiao Li, Junchi Yan, Xinbo Gao, and Heng Huang. Towards transferable targeted attack. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 641–649, 2020.

[148] Yingwei Li, Song Bai, Cihang Xie, Zhenyu Liao, Xiaohui Shen, and Alan L Yuille. Regional homogeneity: Towards learning transferable universal adversarial perturbations against defenses. *arXiv preprint arXiv:1904.00979*, 2019.

[149] Yunsheng Li, Mandar Dixit, and Nuno Vasconcelos. Deep scene image classification

with the mfafvnet. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5746–5754, 2017.

[150] Yiming Li, Congcong Wen, Felix Juefei-Xu, and Chen Feng. Fooling lidar perception via adversarial trajectory perturbation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7898–7907, 2021.

[151] Shiyu Liang, Yixuan Li, and R Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690*, 2017.

[152] Siyuan Liang, Xingxing Wei, Siyuan Yao, and Xiaochun Cao. Efficient adversarial attacks for visual object tracking. *arXiv preprint arXiv:2008.00217*, 2020.

[153] Fangzhou Liao, Ming Liang, Yinpeng Dong, Tianyu Pang, Xiaolin Hu, and Jun Zhu. Defense against adversarial attacks using high-level representation guided denoiser. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1778–1787, 2018.

[154] Tsung-Yu Lin and Subhransu Maji. Improved bilinear pooling with cnns. *In British Media Vision Conference (BMVC)*, 2017.

[155] Tsung-Yu Lin, Aruni RoyChowdhury, and Subhransu Maji. Bilinear cnn models for fine-grained visual recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1449–1457, 2015.

[156] Krzysztof Lis, Krishna Nakka, Mathieu Salzmann, and Pascal Fua. Detecting the unexpected via image resynthesis. *arXiv preprint arXiv:1904.07595*, 2019.

[157] Lingqiao Liu, Peng Wang, Chunhua Shen, Lei Wang, Anton Van Den Hengel, Chao Wang, and Heng Tao Shen. Compositional model based fisher vector coding for image classification. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2335–2348, 2017.

[158] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[159] Wei Liu, Andrew Rabinovich, and Alexander C Berg. Parsenet: Looking wider to see better. *arXiv preprint arXiv:1506.04579*, 2015.

[160] Xiao Liu, Spyridon Thermos, Gabriele Valvano, Agisilaos Chartsias, Alison O'Neil, and Sotirios A Tsaftaris. Measuring the biases and effectiveness of content-style disentanglement. In *Proceedings of the British Media for VIsion Conference*, 2021.

[161] Xin Liu, Huanrui Yang, Ziwei Liu, Linghao Song, Hai Li, and Yiran Chen. Dpatch: An adversarial patch attack on object detectors. *AAAI Workshop on Artificial Intelligence Safety*, 2019.

[162] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. *arXiv preprint arXiv:1611.02770*, 2016.

[163] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[164] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Unsupervised

domain adaptation with residual transfer networks. *Advances in neural information processing systems*, 29, 2016.

[165] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[166] Jiajun Lu, Hussein Sibai, Evan Fabry, and David Forsyth. No need to worry about adversarial examples in object detection in autonomous vehicles. *arXiv preprint arXiv:1707.03501*, 2017.

[167] Xingjun Ma, Bo Li, Yisen Wang, Sarah M Erfani, Sudanthi Wijewickrema, Grant Schoenebeck, Dawn Song, Michael E Houle, and James Bailey. Characterizing adversarial subspaces using local intrinsic dimensionality. *arXiv preprint arXiv:1801.02613*, 2018.

[168] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

[169] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5188–5196, 2015.

[170] Aravindh Mahendran and Andrea Vedaldi. Visualizing deep convolutional neural networks using natural pre-images. *International Journal of Computer Vision*, 120(3):233–255, 2016.

[171] Kaleel Mahmood, Rigel Mahmood, and Marten Van Dijk. On the robustness of vision transformers to adversarial examples. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7838–7847, 2021.

[172] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

[173] Chengzhi Mao, Ziyuan Zhong, Junfeng Yang, Carl Vondrick, and Baishakhi Ray. Metric learning for adversarial robustness. In *Advances in Neural Information Processing Systems*, pages 478–489, 2019.

[174] Julieta Martinez, Rayat Hossain, Javier Romero, and James J Little. A simple yet effective baseline for 3d human pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2640–2649, 2017.

[175] Dushyant Mehta, Helge Rhodin, Dan Casas, Pascal Fua, Oleksandr Sotnychenko, Weipeng Xu, and Christian Theobalt. Monocular 3d human pose estimation in the wild using improved cnn supervision. In *2017 international conference on 3D vision (3DV)*, pages 506–516. IEEE, 2017.

[176] Dushyant Mehta, Srinath Sridhar, Oleksandr Sotnychenko, Helge Rhodin, Mohammad Shafiei, Hans-Peter Seidel, Weipeng Xu, Dan Casas, and Christian Theobalt. Vnect: Real-time 3d human pose estimation with a single rgb camera. *ACM Transactions on Graphics (TOG)*, 36(4):1–14, 2017.

[177] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. On detecting adversarial perturbations. *arXiv preprint arXiv:1702.04267*, 2017.

[178] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[179] Mohammad Moghimi, Serge J Belongie, Mohammad J Saberian, Jian Yang, Nuno Vasconcelos, and Li-Jia Li. Boosted convolutional neural networks. In *British Media Vision Conference (BMVC)*, 2016.

[180] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1765–1773, 2017.

[181] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, Pascal Frossard, and Stefano Soatto. Analysis of universal adversarial perturbations. *arXiv preprint arXiv:1705.09554*, 2017.

[182] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2574–2582, 2016.

[183] Konda Reddy Mopuri, Utsav Garg, and R Venkatesh Babu. Fast feature fool: A data independent approach to universal adversarial perturbations. *British Media Vision Conference*, 2017.

[184] Konda Reddy Mopuri, Phani Krishna Uppala, and R Venkatesh Babu. Ask, acquire, and attack: Data-free uap generation using class impressions. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 19–34, 2018.

[185] Matthias Mueller, Neil Smith, and Bernard Ghanem. A benchmark and simulator for uav tracking. In *European conference on computer vision*, pages 445–461. Springer, 2016.

[186] Aamir Mustafa, Salman Khan, Munawar Hayat, Roland Goecke, Jianbing Shen, and Ling Shao. Adversarial defense by restricting the hidden space of deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3385–3394, 2019.

[187] Aamir Mustafa, Salman H Khan, Munawar Hayat, Roland Goecke, Jianbing Shen, and Ling Shao. Deeply supervised discriminative learning for adversarial defense. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[188] Krishna Kanth Nakka and Mathieu Salzmann. Deep attentional structured representation learning for visual recognition. *British Media Vision Conference (BMVC) 2018*, 2018.

[189] Nina Narodytska and Shiva Kasiviswanathan. Simple black-box adversarial attacks on deep neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1310–1318. IEEE, 2017.

[190] Muzammal Naseer, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Fatih Porikli. On generating transferable targeted perturbations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7708–7717, 2021.

[191] Muzammal Naseer, Salman Khan, Muhammad Haris Khan, Fahad Khan, and Fatih Porikli. Cross-domain transferability of adversarial perturbations. In *33rd Conference on Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

[192] Muzammal Naseer, Kanchana Ranasinghe, Salman Khan, Fahad Shahbaz Khan,

and Fatih Porikli. On improving adversarial transferability of vision transformers. *International Conference of Learning Representations (ICLR*, 2022.

[193] Vidhya Navalpakkam and Laurent Itti. An integrated model of top-down and bottom-up attention for optimizing detection speed. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, pages 2049–2056. IEEE, 2006.

[194] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, 2011.

[195] Anh Nguyen, Alexey Dosovitskiy, Jason Yosinski, Thomas Brox, and Jeff Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In *Advances in neural information processing systems*, pages 3387–3395, 2016.

[196] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 427–436, 2015.

[197] Feiping Nie, Heng Huang, Xiao Cai, and Chris H Ding. Efficient and robust feature selection via joint l2, 1-norms minimization. In *Advances in neural information processing systems*, pages 1813–1821, 2010.

[198] Tianyu Pang, Kun Xu, Chao Du, Ning Chen, and Jun Zhu. Improving adversarial robustness via promoting ensemble diversity. In *International Conference on Machine Learning*, pages 4970–4979. PMLR, 2019.

[199] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.

[200] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519, 2017.

[201] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*, pages 372–387. IEEE, 2016.

[202] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 582–597. IEEE, 2016.

[203] Omkar M Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep face recognition. 2019.

[204] Nikolaos Passalis and Anastasios Tefas. Learning bag-of-features pooling for deep convolutional neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5755–5763, 2017.

[205] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[206] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32:8026–8037, 2019.

[207] Genevieve Patterson and James Hays. Sun attribute database: Discovering, annotating, and recognizing scene attributes. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2751–2758. IEEE, 2012.

[208] Georgios Pavlakos, Xiaowei Zhou, Konstantinos G Derpanis, and Kostas Daniilidis. Coarse-to-fine volumetric prediction for single-image 3d human pose. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7025–7034, 2017.

[209] Georgios Pavlakos, Xiaowei Zhou, Konstantinos G Derpanis, and Kostas Daniilidis. Harvesting multiple views for marker-less 3d human pose annotations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6988–6997, 2017.

[210] Florent Perronnin, Jorge Sánchez, and Thomas Mensink. Improving the fisher kernel for large-scale image classification. In *European conference on computer vision*, pages 143–156. Springer, 2010.

[211] Alin-Ionut Popa, Mihai Zanfir, and Cristian Sminchisescu. Deep multitask architecture for integrated 2d and 3d human sensing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6289–6298, 2017.

[212] Omid Poursaeed, Isay Katsman, Bicheng Gao, and Serge Belongie. Generative adversarial perturbations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4422–4431, 2018.

[213] Zhongang Qi, Saeed Khorram, and Fuxin Li. Visualizing deep networks by optimizing with integrated gradients. In *Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, volume 2, 2019.

[214] Haonan Qiu, Chaowei Xiao, Lei Yang, Xinchen Yan, Honglak Lee, and Bo Li. Semanticadv: Generating adversarial examples via attribute-conditioned image editing. In *European Conference on Computer Vision*, pages 19–37. Springer, 2020.

[215] Ariadna Quattoni and Antonio Torralba. Recognizing indoor scenes. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 413–420. IEEE, 2009.

[216] Pedro Quelhas, Florent Monay, J-M Odobez, Daniel Gatica-Perez, Tinne Tuytelaars, and Luc Van Gool. Modeling scenes with local descriptors and latent aspects. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 1, pages 883–890. IEEE, 2005.

[217] Pranav Rajpurkar, Jeremy Irvin, Kaylie Zhu, Brandon Yang, Hershel Mehta, Tony Duan, Daisy Ding, Aarti Bagul, Curtis Langlotz, Katie Shpanskaya, et al. Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning. *arXiv*

preprint arXiv:1711.05225, 2017.

[218] Anurag Ranjan, Joel Janai, Andreas Geiger, and Michael J Black. Attacking optical flow. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2404–2413, 2019.

[219] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(6):1137–1149, 2016.

[220] Shaoqing Ren, Kaiming He, Ross B Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 2015.

[221] Helge Rhodin, Victor Constantin, Isinsu Katircioglu, Mathieu Salzmann, and Pascal Fua. Neural scene decomposition for multi-person motion capture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7703–7713, 2019.

[222] Helge Rhodin, Mathieu Salzmann, and Pascal Fua. Unsupervised geometry-aware representation for 3d human pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 750–767, 2018.

[223] Gregory Rogez, Philippe Weinzaepfel, and Cordelia Schmid. Lcr-net: Localization-classification-regression for human pose. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3433–3441, 2017.

[224] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[225] Andrew Slavin Ross and Finale Doshi-Velez. Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

[226] Ueli Rutishauser, Dirk Walther, Christof Koch, and Pietro Perona. Is bottom-up attention useful for object recognition? In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–II. IEEE, 2004.

[227] Aniruddha Saha, Akshayvarun Subramanya, Koninika Patil, and Hamed Pirsiavash. Adversarial patches exploiting contextual reasoning in object detection. *Conference on Computer Vision and Pattern Recognition Workshops*, 2020.

[228] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-gan: Protecting classifiers against adversarial attacks using generative models. In *International Conference on Learning Representations*, 2018.

[229] Jorge Sanchez, Florent Perronnin, Thomas Mensink, and Jakob Verbeek. Image classification with the fisher vector: Theory and practice. *International journal of computer vision*, 105(3):222–245, 2013.

[230] Ludwig Schmidt, Shibani Santurkar, Dimitris Tsipras, Kunal Talwar, and Aleksander Madry. Adversarially robust generalization requires more data. *Advances in neural information processing systems*, 31, 2018.

[231] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified

embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[232] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[233] Ramprasaath R Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. Grad-cam: Why did you say that? *arXiv preprint arXiv:1611.07450*, 2016.

[234] Ali Shafahi, Mahyar Najibi, Mohammad Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! In *Advances in Neural Information Processing Systems*, pages 3353–3364, 2019.

[235] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K Reiter. Adversarial generative nets: Neural network attacks on state-of-the-art face recognition. *arXiv preprint arXiv:1801.00349*, 2(3), 2017.

[236] Marcel Simon, Yang Gao, Trevor Darrell, Joachim Denzler, and Erik Rodner. Generalized orderless pooling performs implicit salient matching. In *Proceedings of the IEEE international conference on computer vision*, pages 4960–4969, 2017.

[237] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *In Internal Conference on Learning Representations (ICLR), pages 1409-1556*, 2015.

[238] Saurabh Singh, Abhinav Gupta, and Alexei A Efros. Unsupervised discovery of mid-level discriminative patches. In *Computer Vision–European Conference on Computer Vision (ECCV) 2012*, pages 73–86. Springer, 2012.

[239] Josef Sivic and Andrew Zisserman. Video google: A text retrieval approach to object matching in videos. In *null*, page 1470. IEEE, 2003.

[240] Ibrahim Sobh, Ahmed Hamed, Varun Ravi Kumar, and Senthil Yogamani. Adversarial attacks on multi-task visual perception for autonomous driving. *Journal of Imaging Science and Technology*, 65(6):60408–1, 2021.

[241] Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Ermon, and Nate Kushman. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. In *International Conference on Learning Representations*, 2018.

[242] Yang Song, Rui Shu, Nate Kushman, and Stefano Ermon. Constructing unrestricted adversarial examples with generative models.

[243] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 23(5):828–841, 2019.

[244] Baochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *European conference on computer vision*, pages 443–450. Springer, 2016.

[245] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew

Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[246] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[247] Peng Tang, Xinggang Wang, Baoguang Shi, Xiang Bai, Wenyu Liu, and Zhuowen Tu. Deep fishernet for object classification. *arXiv preprint arXiv:1608.00182*, 2016.

[248] Bugra Tekin, Pablo Márquez-Neila, Mathieu Salzmann, and Pascal Fua. Learning to fuse 2d and 3d image cues for monocular body pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3941–3950, 2017.

[249] Simen Thys, Wiebe Van Ranst, and Toon Goedemé. Fooling automated surveillance cameras: adversarial patches to attack person detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.

[250] Denis Tome, Chris Russell, and Lourdes Agapito. Lifting from the deep: Convolutional 3d pose estimation from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2500–2509, 2017.

[251] Lorenzo Torresani, Martin Szummer, and Andrew Fitzgibbon. Efficient object category recognition using classemes. In *European conference on computer vision*, pages 776–789. Springer, 2010.

[252] Florian Tramer and Dan Boneh. Adversarial training and robustness for multiple perturbations. *Advances in Neural Information Processing Systems*, 32, 2019.

[253] Florian Tramer, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. On adaptive attacks to adversarial example defenses. *Advances in Neural Information Processing Systems*, 33:1633–1645, 2020.

[254] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *International Conference on Learning Representations (ICLR)*, 2018.

[255] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. In *International Conference on Learning Representations*, 2018.

[256] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction apis. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*, pages 601–618, 2016.

[257] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. *International Conference on Learning Representations (ICLR)*, 2018.

[258] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5018–5027, 2017.

[259] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie.

The caltech-ucsd birds-200-2011 dataset. 2011.

[260] Dequan Wang, Zhiqiang Shen, Jie Shao, Wei Zhang, Xiangyang Xue, and Zheng Zhang. Multiple granularity descriptors for fine-grained categorization. In *Computer Vision (ICCV), 2015 IEEE International Conference on*, pages 2399–2406. IEEE, 2015.

[261] Fei Wang, Mengqing Jiang, Chen Qian, Shuo Yang, Cheng Li, Honggang Zhang, Xiaogang Wang, and Xiaoou Tang. Residual attention network for image classification. *arXiv preprint arXiv:1704.06904*, 2017.

[262] Qiang Wang, Li Zhang, Luca Bertinetto, Weiming Hu, and Philip HS Torr. Fast online object tracking and segmentation: A unifying approach. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1328–1338, 2019.

[263] Yaming Wang, Jonghyun Choi, Vlad I Morariu, and Larry S Davis. Mining discriminative triplets of patches for fine-grained classification. *In Computer Vision and Pattern Recognition (CVPR)*, 2016.

[264] Yisen Wang, Xingjun Ma, James Bailey, Jinfeng Yi, Bowen Zhou, and Quanquan Gu. On the convergence and robustness of adversarial training. In *International Conference on Machine Learning (ICML)*, pages 6586–6595. PMLR, 2019.

[265] Yaming Wang, Vlad I Morariu, and Larry S Davis. Learning a discriminative filter bank within a cnn for fine-grained recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4148–4157, 2018.

[266] Zhe Wang, Hongsheng Li, Wanli Ouyang, and Xiaogang Wang. Learnable histogram: Statistical context features for deep neural networks. In *European Conference on Computer Vision*, pages 246–262. Springer, 2016.

[267] Xiu-Shen Wei, Chen-Wei Xie, Jianxin Wu, and Chunhua Shen. Mask-cnn: Localizing parts and selecting descriptors for fine-grained bird species categorization. *Pattern Recognition*, 76:704–714, 2018.

[268] Zhipeng Wei, Jingjing Chen, Micah Goldblum, Zuxuan Wu, Tom Goldstein, and Yu-Gang Jiang. Towards transferable adversarial attacks on vision transformers. *arXiv preprint arXiv:2109.04176*, 2021.

[269] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *Advances in neural information processing systems*, pages 2074–2082, 2016.

[270] Rey Reza Wiyatno and Anqi Xu. Physical adversarial textures that fool visual object tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4822–4831, 2019.

[271] Alex Wong, Safa Cicek, and Stefano Soatto. Targeted adversarial perturbations for monocular depth prediction. *Advances in neural information processing systems*, 33:8486–8497, 2020.

[272] Eric Wong, Leslie Rice, and J Zico Kolter. Fast is better than free: Revisiting adversarial training. *International Conference on Learning Representations (ICLR)*, 2020.

[273] Ruobing Wu, Baoyuan Wang, Wenping Wang, and Yizhou Yu. Harvesting discrim-

inative meta objects with deep cnn features for scene classification. In *Computer Vision (ICCV), 2015 IEEE International Conference on*, pages 1287–1295. IEEE, 2015.

[274] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Online object tracking: A benchmark. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2411–2418, 2013.

[275] Zuxuan Wu, Yu-Gang Jiang, Jun Wang, Jian Pu, and Xiangyang Xue. Exploring inter-feature and inter-class relationships with deep neural networks for video classification. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 167–176, 2014.

[276] Chaowei Xiao, Ruizhi Deng, Bo Li, Taesung Lee, Benjamin Edwards, Jinfeng Yi, Dawn Song, Mingyan Liu, and Ian Molloy. Advit: Adversarial frames identifier based on temporal consistency in videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3968–3977, 2019.

[277] Chaowei Xiao, Ruizhi Deng, Bo Li, Fisher Yu, Mingyan Liu, and Dawn Song. Characterizing adversarial examples based on spatial consistency information for semantic segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 217–234, 2018.

[278] Chaowei Xiao, Bo Li, Jun-Yan Zhu, Warren He, Mingyan Liu, and Dawn Song. Generating adversarial examples with adversarial networks. *arXiv preprint arXiv:1801.02610*, 2018.

[279] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[280] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *Computer vision and pattern recognition (CVPR), 2010 IEEE conference on*, pages 3485–3492. IEEE, 2010.

[281] Tianjun Xiao, Yichong Xu, Kuiyuan Yang, Jiaxing Zhang, Yuxin Peng, and Zheng Zhang. The application of two-level attention models in deep convolutional neural network for fine-grained image classification. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 842–850. IEEE, 2015.

[282] Zihao Xiao, Xianfeng Gao, Chilin Fu, Yinpeng Dong, Wei Gao, Xiaolu Zhang, Jun Zhou, and Jun Zhu. Improving transferability of adversarial patches on face recognition with generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11845–11854, 2021.

[283] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. Mitigating adversarial effects through randomization. In *International Conference on Learning Representations*, 2018.

[284] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Yuyin Zhou, Lingxi Xie, and Alan Yuille. Adversarial examples for semantic segmentation and object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1369–1378, 2017.

[285] Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan L Yuille, and Kaiming He. Feature denoising for improving adversarial robustness. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 501–509, 2019.

[286] Kaidi Xu, Sijia Liu, Pu Zhao, Pin-Yu Chen, Huan Zhang, Quanfu Fan, Deniz Erdogmus, Yanzhi Wang, and Xue Lin. Structured adversarial attack: Towards general implementation and better interpretability. In *International Conference on Learning Representations*, 2018.

[287] Bin Yan, Dong Wang, Huchuan Lu, and Xiaoyun Yang. Cooling-shrinking attack: Blinding the tracker with imperceptible noises. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 990–999, 2020.

[288] Huanrui Yang, Jingyang Zhang, Hongliang Dong, Nathan Inkawhich, Andrew Gardner, Andrew Touchet, Wesley Wilkes, Heath Berry, and Hai Li. Dverge: diversifying vulnerabilities for enhanced robust generation of ensembles. *Advances in Neural Information Processing Systems*, 33:5505–5515, 2020.

[289] Kaichen Yang, Tzungyu Tsai, Honggang Yu, Max Panoff, Tsung-Yi Ho, and Yier Jin. Robust roadside physical adversarial attack against deep learning in lidar perception modules. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*, pages 349–362, 2021.

[290] Minghao Yin, Yongbing Zhang, Xiu Li, and Shiqi Wang. When deep fool meets deep prior: Adversarial attack on super-resolution network. In *Proceedings of the 26th ACM international conference on Multimedia*, pages 1930–1938, 2018.

[291] Fisher Yu, Vladlen Koltun, and Thomas Funkhouser. Dilated residual networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 472–480, 2017.

[292] Kaicheng Yu and Mathieu Salzmann. Statistically-motivated second-order pooling. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 600–616, 2018.

[293] Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.

[294] Zheng Yuan, Jie Zhang, Yunpei Jia, Chuanqi Tan, Tao Xue, and Shiguang Shan. Meta gradient adversarial attack. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7748–7757, 2021.

[295] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4694–4702, 2015.

[296] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6023–6032, 2019.

[297] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional

networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[298] Cong Zhang, Hongsheng Li, Xiaogang Wang, and Xiaokang Yang. Cross-scene crowd counting via deep convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 833–841, 2015.

[299] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.

[300] Hang Zhang, Kristin Dana, Jianping Shi, Zhongyue Zhang, Xiaogang Wang, Ambrish Tyagi, and Amit Agrawal. Context encoding for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7151–7160, 2018.

[301] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *International conference on machine learning*, pages 7354–7363. PMLR, 2019.

[302] Han Zhang, Tao Xu, Mohamed Elhoseiny, Xiaolei Huang, Shaoting Zhang, Ahmed Elgammal, and Dimitris Metaxas. Spda-cnn: Unifying semantic part detection and abstraction for fine-grained recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1143–1152, 2016.

[303] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P Xing, Laurent El Ghaoui, and Michael I Jordan. Theoretically principled trade-off between robustness and accuracy. *International Conference on Machine Learning (ICML)*, 2019.

[304] Jianming Zhang, Zhe Lin, Jonathan Brandt, Xiaohui Shen, and Stan Sclaroff. Top-down neural attention by excitation backprop. In *European Conference on Computer Vision*, pages 543–559. Springer, 2016.

[305] Jingfeng Zhang, Xilie Xu, Bo Han, Gang Niu, Lizhen Cui, Masashi Sugiyama, and Mohan Kankanhalli. Attacks which do not kill training make adversarial learning stronger. In *International conference on machine learning*, pages 11278–11287. PMLR, 2020.

[306] Ning Zhang, Jeff Donahue, Ross Girshick, and Trevor Darrell. Part-based r-cnns for fine-grained category detection. In *European conference on computer vision*, pages 834–849. Springer, 2014.

[307] Quanshi Zhang, Ying Nian Wu, and Song-Chun Zhu. Interpretable convolutional neural networks. *arXiv preprint arXiv:1710.00935*, 2(3):5, 2017.

[308] Zhipeng Zhang, Houwen Peng, Jianlong Fu, Bing Li, and Weiming Hu. Ocean: Object-aware anchor-free tracking. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXI 16*, pages 771–787. Springer, 2020.

[309] Ziqi Zhang, Xinge Zhu, Yingwei Li, Xiangqun Chen, and Yao Guo. Adversarial attacks on monocular depth estimation. *arXiv preprint arXiv:2003.10315*, 2020.

[310] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017.

[311] Hengshuang Zhao, Yi Zhang, Shu Liu, Jianping Shi, Chen Change Loy, Dahua Lin, and Jiaya Jia. Psanet: Point-wise spatial attention network for scene parsing. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 267–283, 2018.

[312] Zhengli Zhao, Dheeru Dua, and Sameer Singh. Generating natural adversarial examples. In *International Conference on Learning Representations*, 2018.

[313] Zhengyu Zhao, Zhuoran Liu, and Martha Larson. On success and simplicity: A second look at transferable targeted attacks. *Advances in Neural Information Processing Systems*, 34, 2021.

[314] Heliang Zheng, Jianlong Fu, Tao Mei, and Jiebo Luo. Learning multi-attention convolutional neural network for fine-grained image recognition. In *International Conference on Computer Vision*, 2017.

[315] Haizhong Zheng, Ziqi Zhang, Juncheng Gu, Honglak Lee, and Atul Prakash. Efficient adversarial training with transferable adversarial examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1181–1190, 2020.

[316] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*, pages 2921–2929. IEEE, 2016.

[317] Wen Zhou, Xin Hou, Yongjun Chen, Mengyun Tang, Xiangqi Huang, Xiang Gan, and Yong Yang. Transferable adversarial perturbations. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 452–467, 2018.

[318] Zheng Zhu, Qiang Wang, Bo Li, Wei Wu, Junjie Yan, and Weiming Hu. Distractor-aware siamese networks for visual object tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 101–117, 2018.

Krishna Kanth Nakka ***krishnakanthnakka.github.io***

Interests

Computer Vision, Machine Learning and Deep Learning

Education

Ecole Polytechnique Fédérale de Lausanne (EPFL) **Sep 2017 - Aug 2022**

Ph.D. in Computer Science
Advisors: Dr. Mathieu Salzmann and Prof. Pascal Fua

Title: **Understanding Deep Neural Networks using Adversarial Attacks**
My thesis focuses on the strengths and weaknesses of deep neural networks in safety-critical applications. It explores the topics of interpretable models, transfer-based black-box attacks, attack detection, adversarial defenses, anomaly detection, and disentangled representations.

Indian Institute of Technology Kharagpur **Jun 2010 - May 2015**

M.Tech with specialization in Signal Processing and Instrumentation,
B.Tech (Honours) in Electrical Engineering (5 year Dual Degree) GPA: 8.89/10.0

Awards and Honours

EDIC PhD Fellowship (2017) to pursue first year of doctoral studies at EPFL
Mitacs Globalink Scholarship to participate in summer internship at University of Alberta
University of Queensland Summer Research Scholarship to conduct research at CAI
MCM Scholarship for 4 years (2010-14) for excellent academic performance at IIT Kharagpur

Work Experience

Samsung R&D Institute, Bangalore **Sep 2015 - July 2017**
TL: Dr. Shankar Venkatesan, Advanced Technology Lab
Prototyped a joint reflection-removal and super-resolution of a video sequence.

University of Alberta, Edmonton **May 2014 - July 2014**
Under: Prof. Nilanjan Ray, Computing Science Department
Evaluated large scale image retrieval methods using product quantization of sub-codebooks.

University of Queensland, Australia **Nov 2013 - Jan 2014**
Under: Prof. Jeffrey Harmer, Center for Advanced Imaging Institute
Developed an exponentially decaying non-uniform sampling scheme to shorten acquisition time in spectroscopy experiments.

Philips Research Asia, Bangalore **May 2013 - July 2013**
Under: Dr. Shankar M Venkatesan
Implemented a part-based human detection model using Adaboost of weak SVM classifiers.

Publications And Preprints

1. **Understanding Pose and Appearance Disentanglement in 3D Human Pose Estimation**
Krishna Kanth Nakka and Mathieu Salzmann,
Under review

2. **Learning Transferable Adversarial Perturbations**
Krishna Kanth Nakka and Mathieu Salzmann,
Neural Information and Processing Systems, NeurIPS 2021

3. **Universal, Transferable Adversarial Attacks for Visual Object Trackers**
Krishna Kanth Nakka and Mathieu Salzmann,
Under review

4. **Towards Robust Fine-grained Recognition by Maximal Separation of Discriminative Features**
Krishna Kanth Nakka and Mathieu Salzmann,
Asian Conference on Computer Vision (ACCV), 2020.

5. **Indirect Local Attacks for Context-aware Semantic Segmentation Networks**
Krishna Kanth Nakka and Mathieu Salzmann,
European Conference on Computer Vision (ECCV) ***Spotlight 2020. (Top 5%)***

6. **Detecting the Unexpected via Image Resynthesis**
Krzysztof Lis, Krishna Kanth Nakka, Pascal Fua, Mathieu Salzmann,
International Conference on Computer Vision (ICCV), 2019.

7. **Interpretable BoW Networks for Adversarial Example Detection**
Krishna Kanth Nakka and Mathieu Salzmann,
Explainable and Interpretable AI workshop, ICCV 2019.

8. **Deep Attentional Structured Representation Learning for Visual Recognition**
Krishna Kanth Nakka and Mathieu Salzmann,
British Media Vision Conference (BMVC), 2018.

9. **Deep learning based fence segmentation and removal from an image using a video sequence**
Jonna S, Nakka KK, Sahay RR,
International Workshop on Video Segmentation, ECCV *2016.* ***Oral.***

10. **Detection and removal of fence occlusions in an image using a video of the static/dynamic scene**
Jonna S, Nakka KK, Khasare VS, Sahay RR, Kankanhalli MS,
Journal of Optical Society of America (JOSA) *A. 2016.*

11. **My camera can see through fences: A deep learning approach for image de-fencing**
Jonna S, Nakka KK, Sahay RR,
Asian Conference on Pattern Recognition ACPR*, 2015.*

12. **3D-to-2D mapping for user interactive segmentation of human leg muscles from MRI data**
Ray N, Mukherjee S, Nakka KK, Acton ST, Blanker SS,
Signal and Information Processing, GlobalSIP 2014.

13. **Non-uniform sampling in EPR: optimizing data acquisition for Hyscore spectroscopy**
Nakka KK, YA Tesiram, IM Brereton, M Mobli and JR Harmer,
Physical Chemistry Chemical Physics, 2014.

Skills

- Languages: Proficient in Python. Familiar with C/C++
- Softwares: PyTorch, Tensorflow, Caffe

References

- Dr. Mathieu Salzmann. email: mathieu.salzmann@epfl.ch
- Prof. Pascal Fua. email: pascal.fua@epfl.ch