PhD Thesis Defense

Understanding Deep Neural Networks using Adversarial Attacks

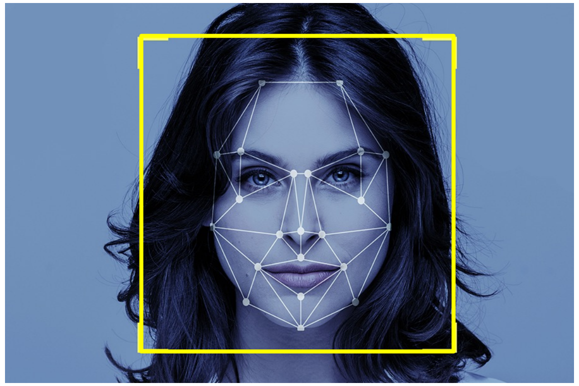
Krishna Kanth Nakka

August 15, 2022

CVLab, EPFL

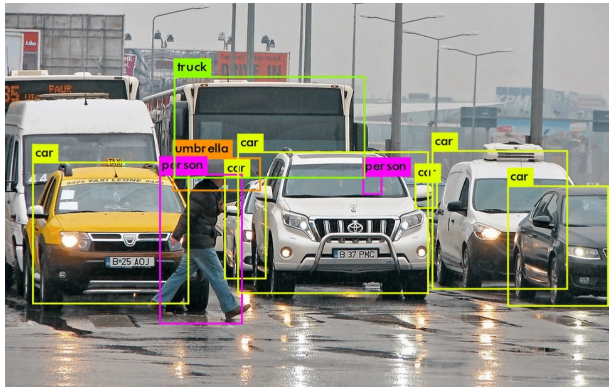
Motivation

Understanding the behavior of DNNs in safety and security-critical applications is paramount



Biometric recognition

Scene segmentation



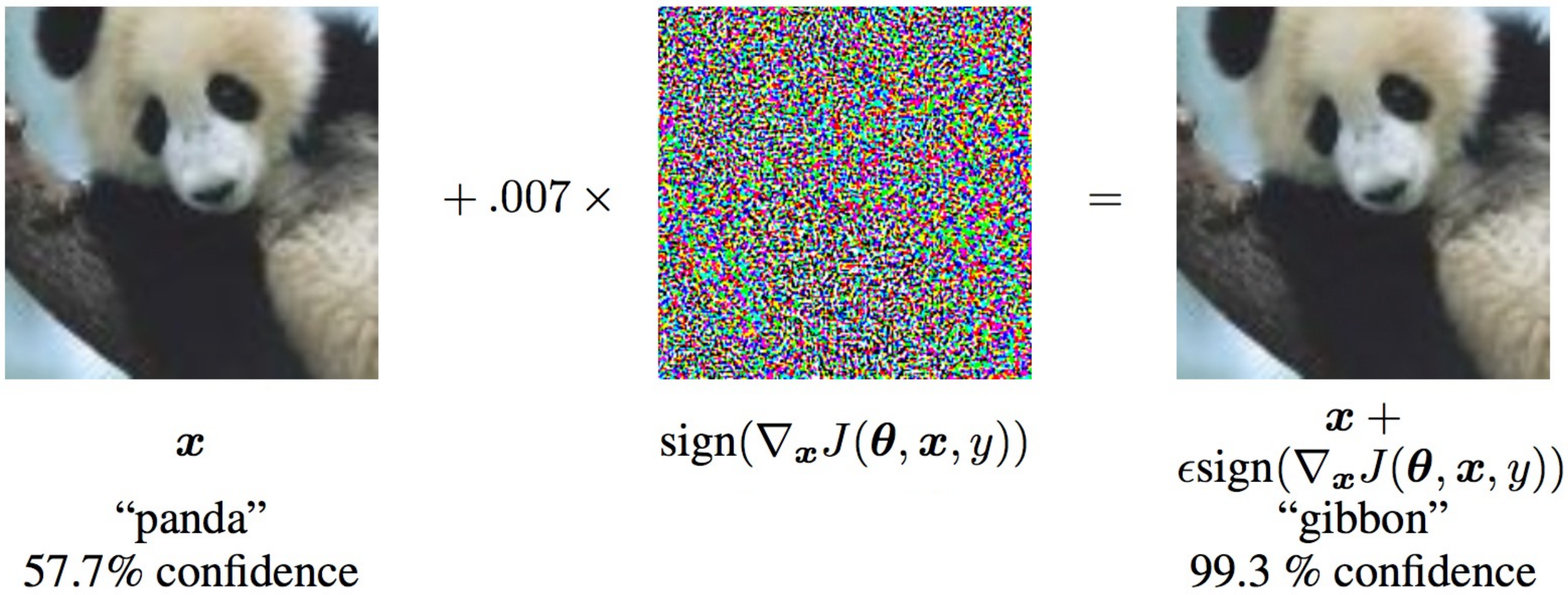
Object localization

Health-care applications

Self-driving cars

Adversarial Examples (AE)

DNNs are sensitive to imperceptable perturbations

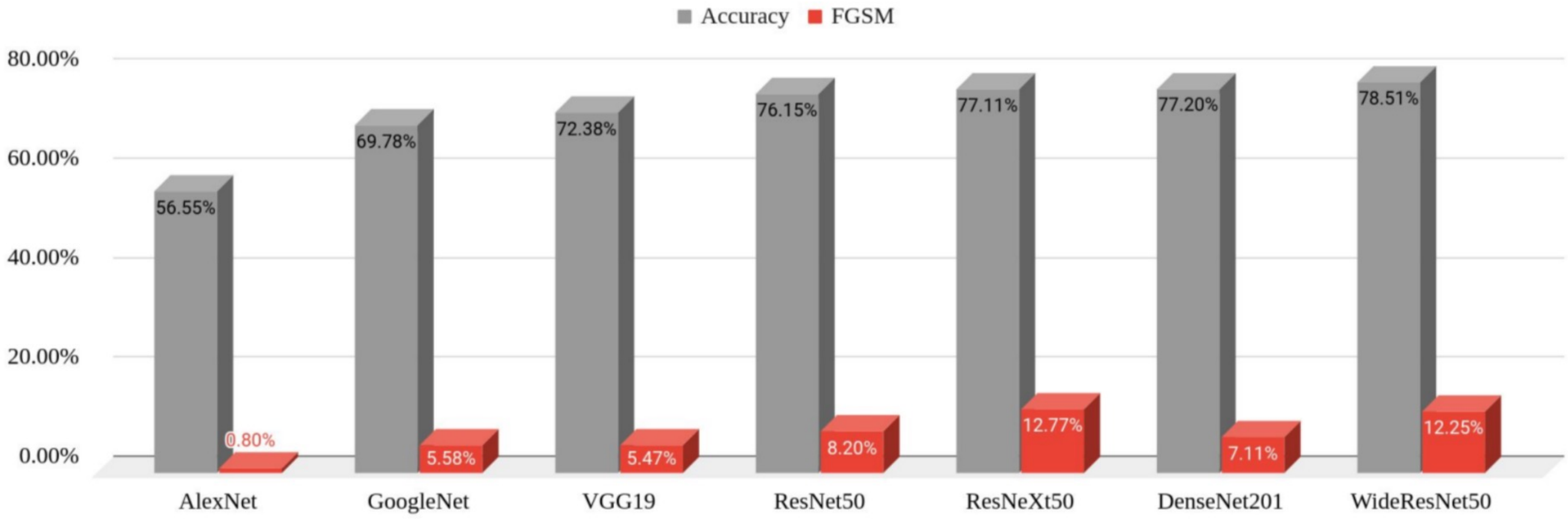


Key properties of adversarial examples

- Small perturbation
- High confidence
- Transferability

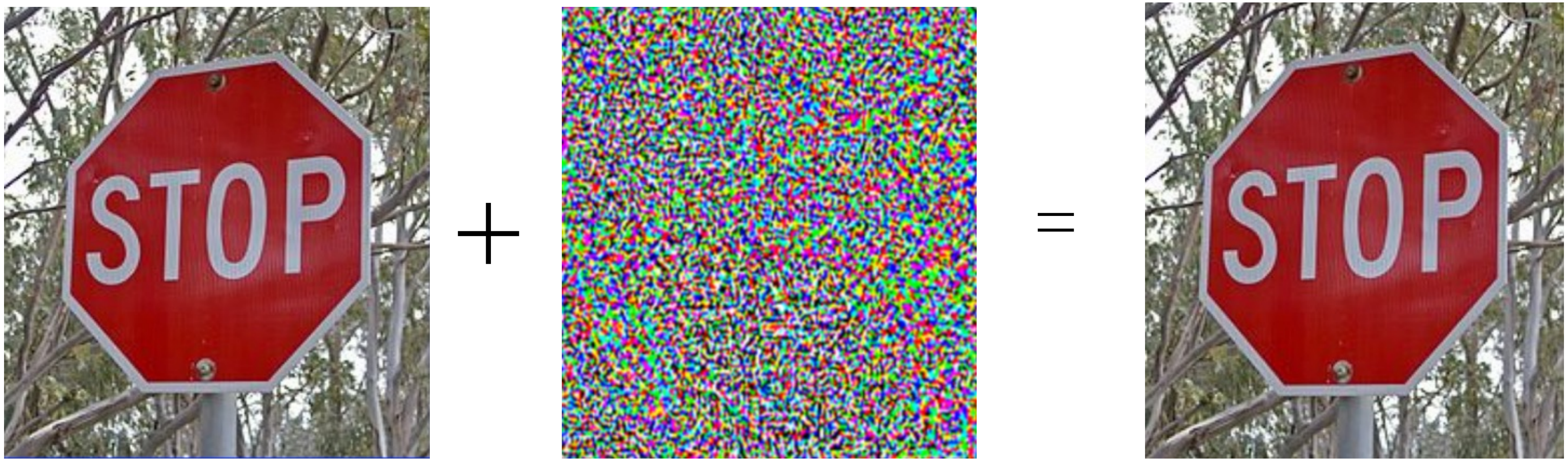
Szegedy et al. arXiv 2013

DNNs performance drops signficantly with single step FGSM attack



Perturbation norm set to 8.
Results are reported on ImageNet validation set

Implications of adversarial attacks against autonomous vehicles



STOP sign   Perturbation   Max speed 100

A unifying perspective of thesis

1.Understand the underlying working mechanisms of adversarial attacks on DNNs

2. Design adversarial attacks to both fool and explain the DNNs

Focus areas

Interpretable models

Adversarial Defense

Adversarial Attacks

Focus areas

Interpretable models

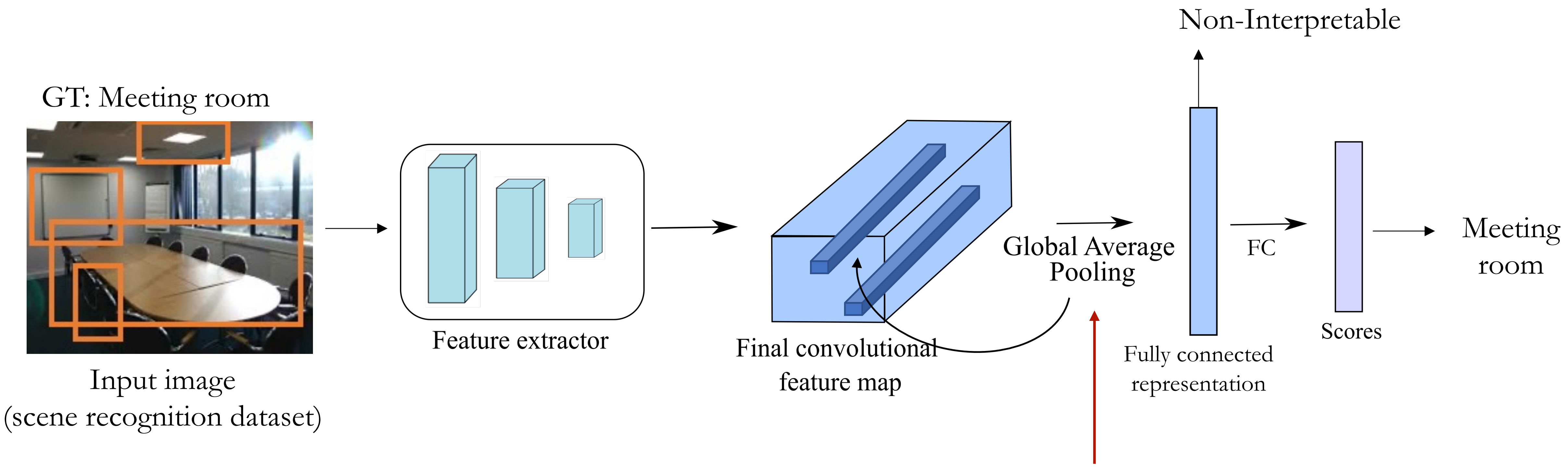
Adversarial Defense

Adversarial Attacks

In order to build trust in safety-critical systems, we need to build transparent models that have the ability to explain why they predict what they predict

Our work focuses on **Bag-of-visual words (BoW) pooling** architectures to understand the decisions of DNNs

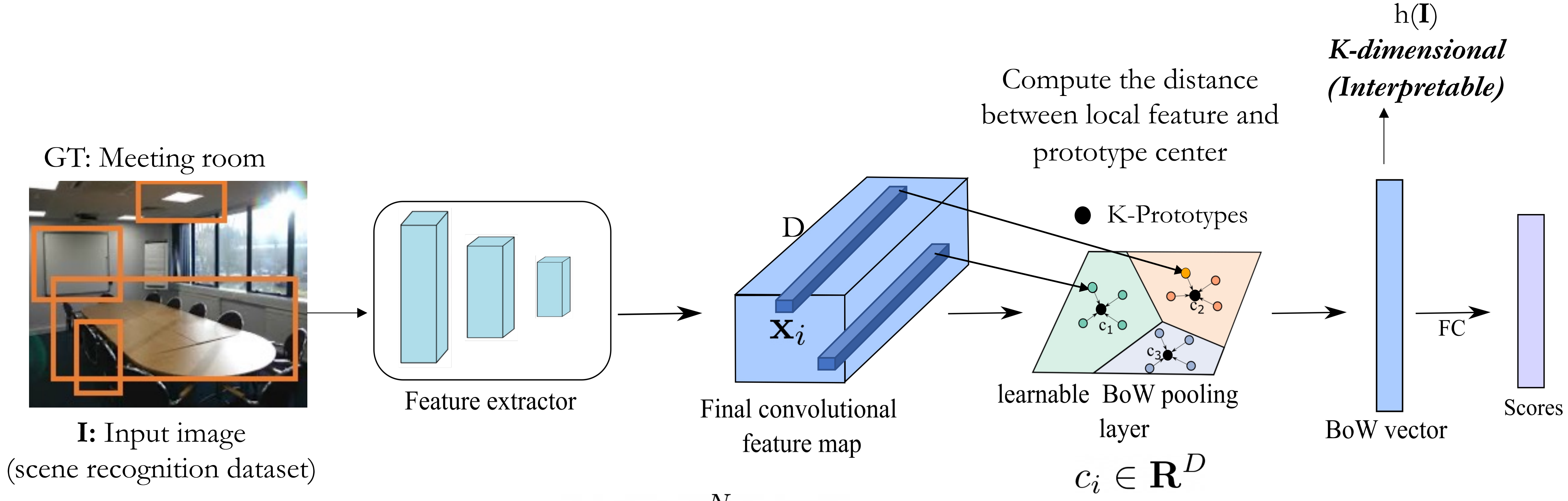
Standard DNN architectures are difficult to understand how they reach to their decisions



presence of table, board, chairs, light bulb indicates meeting room class

Instead of GAP, replace with learnable BoW pooling layer to learn interpretable representation

NetBoW: DNN with distance-based learnable pooling layer



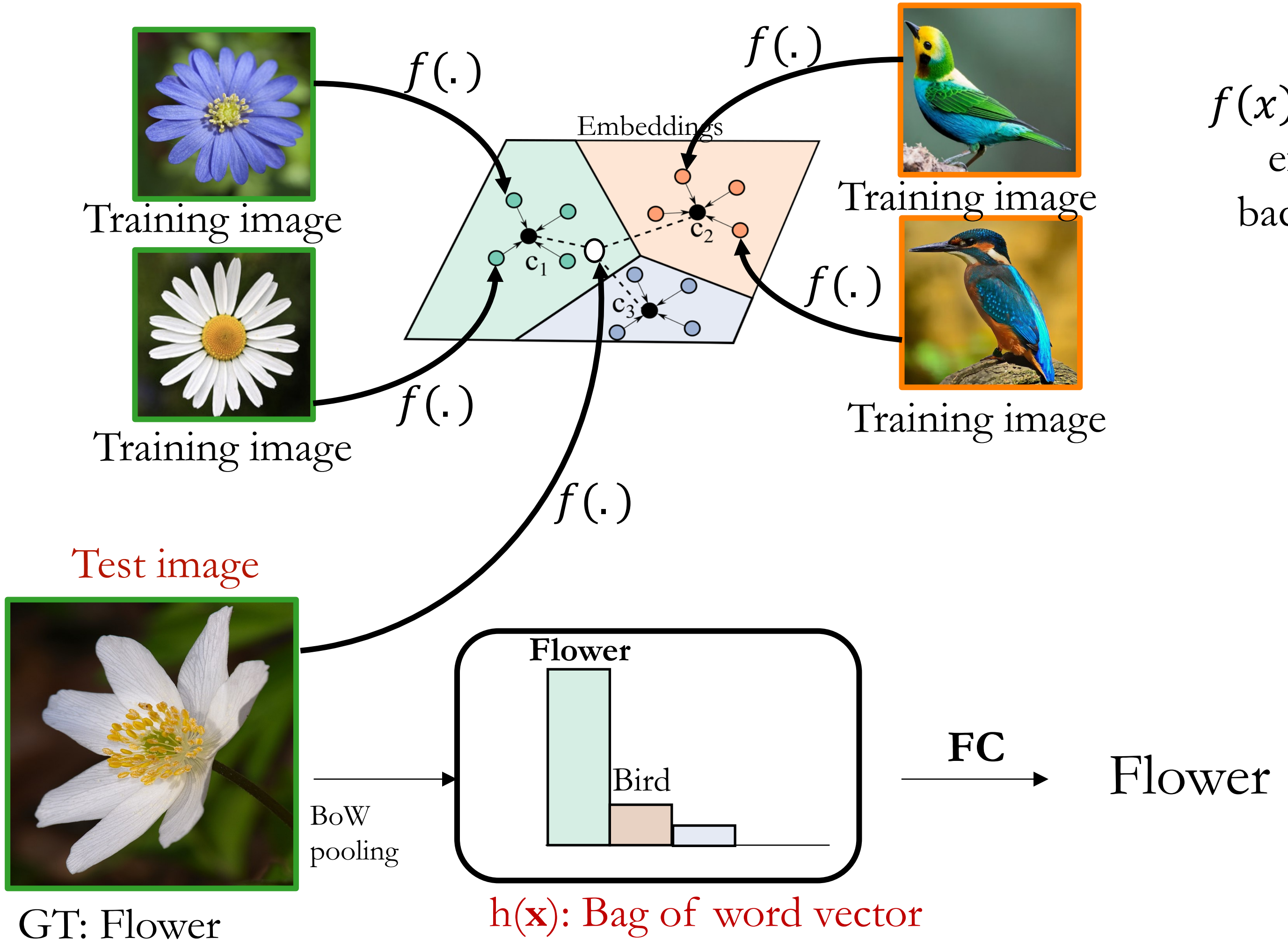
presence of table, board, chairs, light bulb indicates meeting room class

$$h_k(\mathbf{I}) = \sum_{i=1}^{N} a_k(\mathbf{x}_i)$$

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

Network learns the 1. weights of backbone, 2. prototype centers (c_1, c_2, c_3), and 3. FC layer jointly

Advantage: Interpretable BoW representation



$f(x)$ denotes the feature extracted from the backbone network for image x

Interpretability by Design

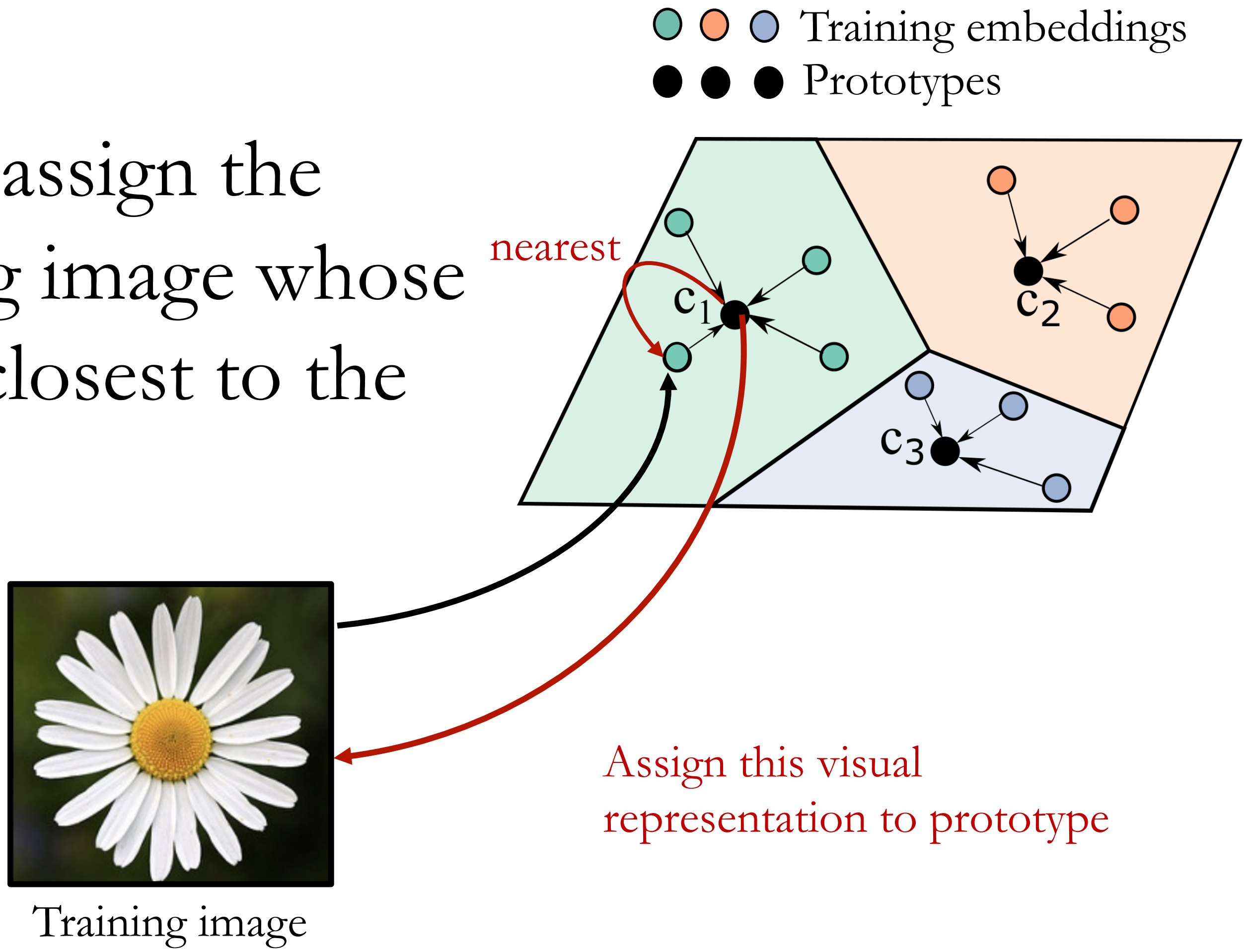
BoW Networks are interpretable since one can understand the reasons for particular output decision through the prototype activations and not in post-hoc manner

↓

Diagnose the failure modes such as adversarial examples, out-of-distribution examples and analyze them more naturally and intrinsically

What does prototype represent in input image space?

After training, assign the nearest training image whose embedding is closest to the prototype



NetBoW helps to understand the reasons for a label prediction through visual codebook

Input images

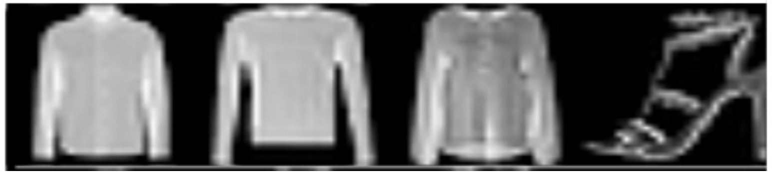
MNIST

FMNIST

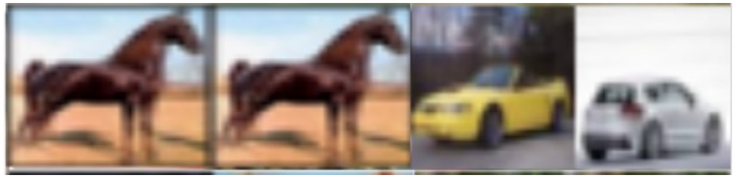
CIFAR10

Highest activated prototype

MNIST

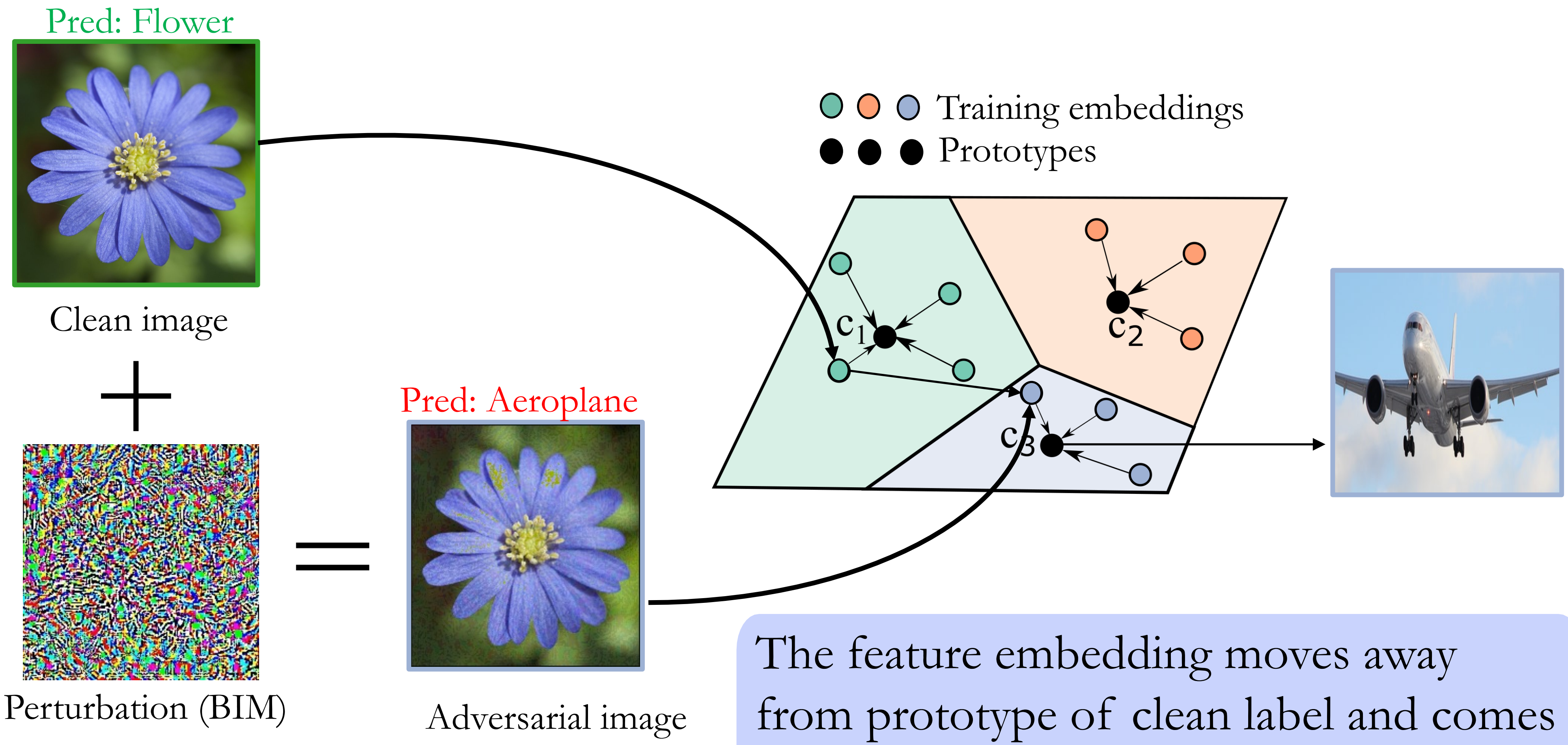


FMNIST



CIFAR10

Can we understand the mechanism of adversarial examples through interpretable models in a better way?



The feature embedding moves away from prototype of clean label and comes closer to prototype of adversarial label

How can we use the interpretable BoW networks to detect the adversarial examples

Key idea

Adversarial attacked image should activate the prototype of other class. Therefore, we detect the attacks by comparing the input image with the visual representation of activated prototype through an auxiliary detector network

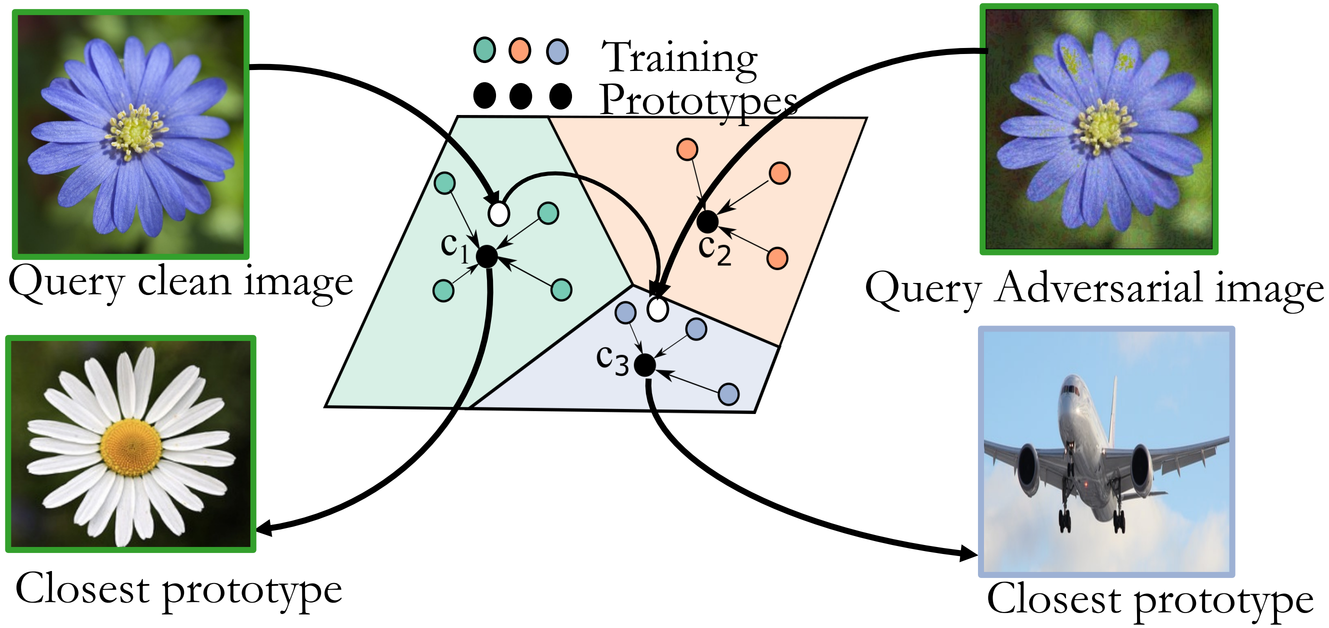
Impact:

Adversarial images

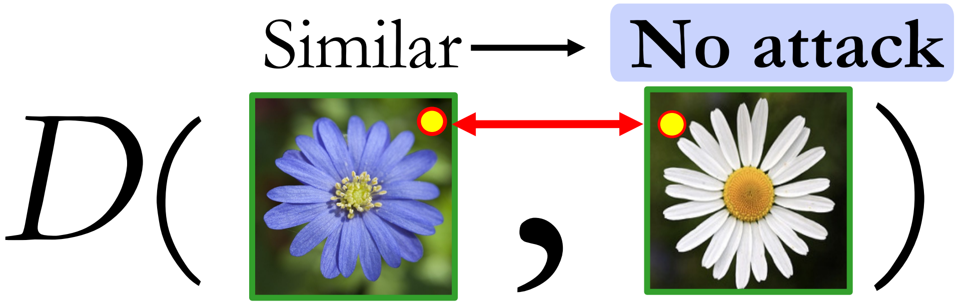
Activated prototypes

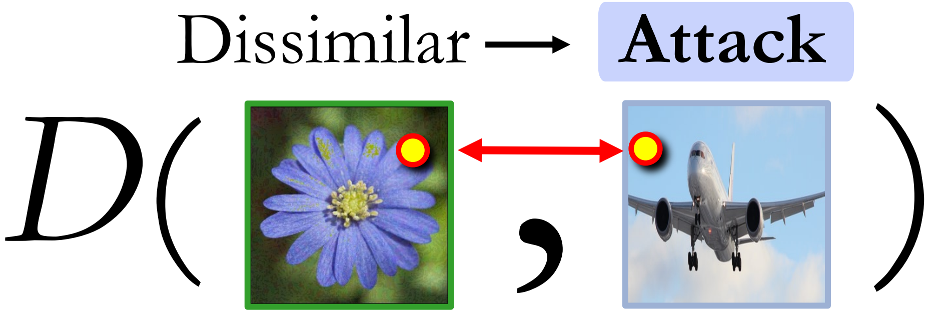
[KKN & MS, ICCVW19]

Adversarial example detection: pose the problem as similarity matching of input image to highest activated prototype



Similar approach can be used to detect out-of-distibution examples



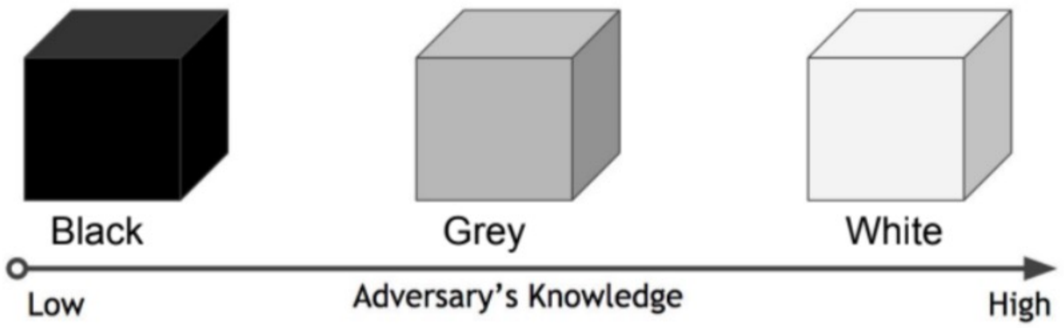


D: Siamese-based detector to predict if the input pair is similar or dissimilar

Is the detector robust to attacks all the time?

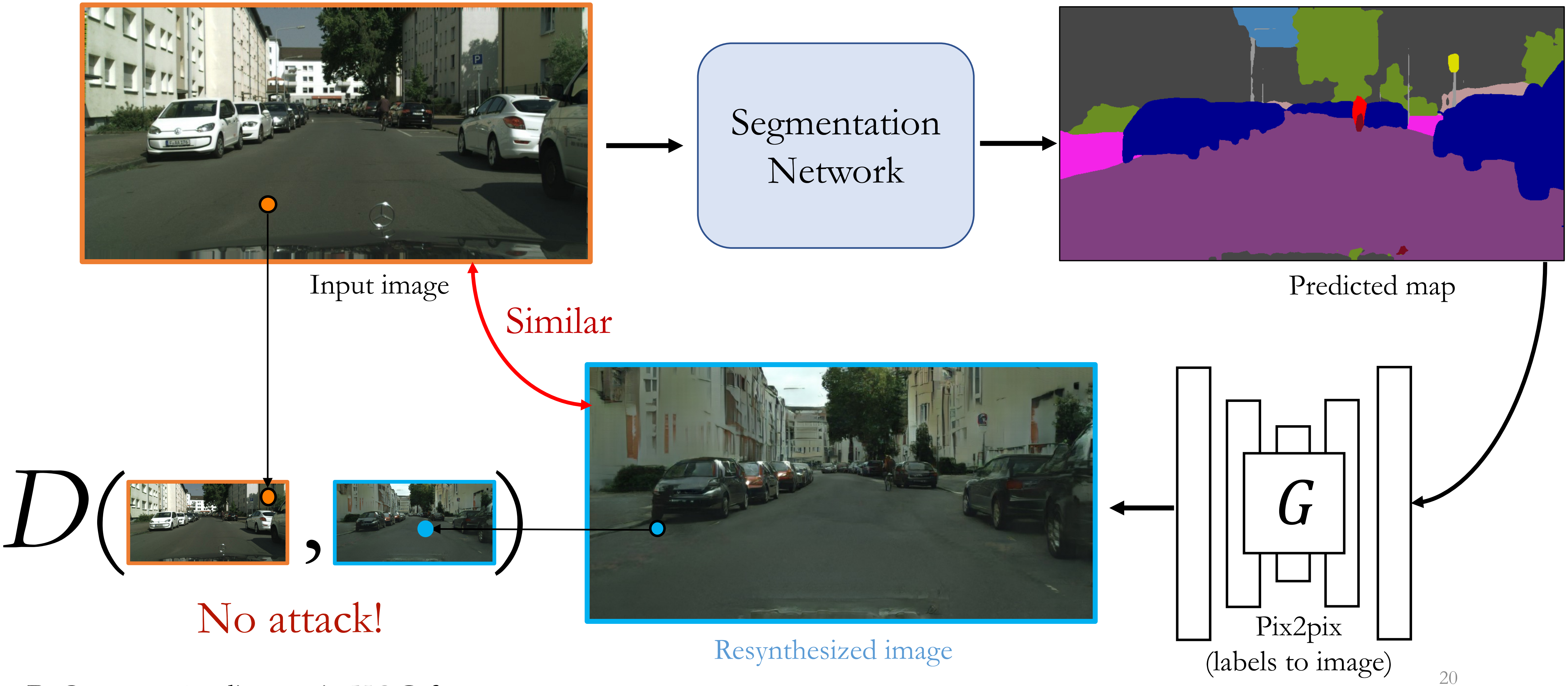
- No. Detector breaks down with our defense-aware adaptive attack in pure white box setting (aware of detector weights and stratefy)

- However, the approach works in gray-box (adversary aware of defense mechanism) and black-box setting (no access to detector weights) wrt detector



Adversary detection beyond image-recognition

Adversarial example detection in semantic segmentation by comparing input image to the image resynthesized from output map

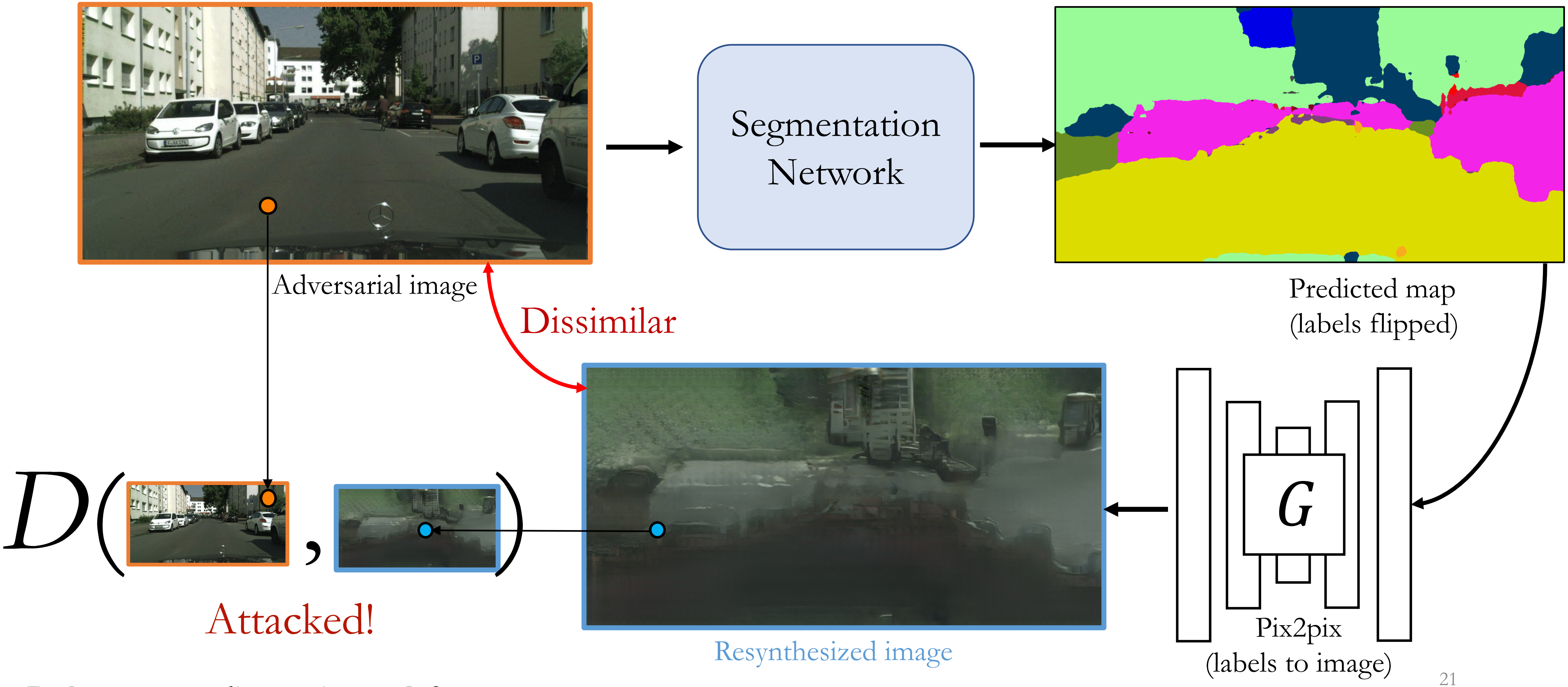


D: Computes L_1 distance in HOG feature space

[KL, KKN, PF & MS, ICCV19]

Adversary detection beyond image-recognition

Adversarial example detection in semantic segmentation by comparing input image to the image resynthesized from output map



D: Computes L_1 distance in HOG feature space

[KL, KKN, PF & MS, ICCV19]

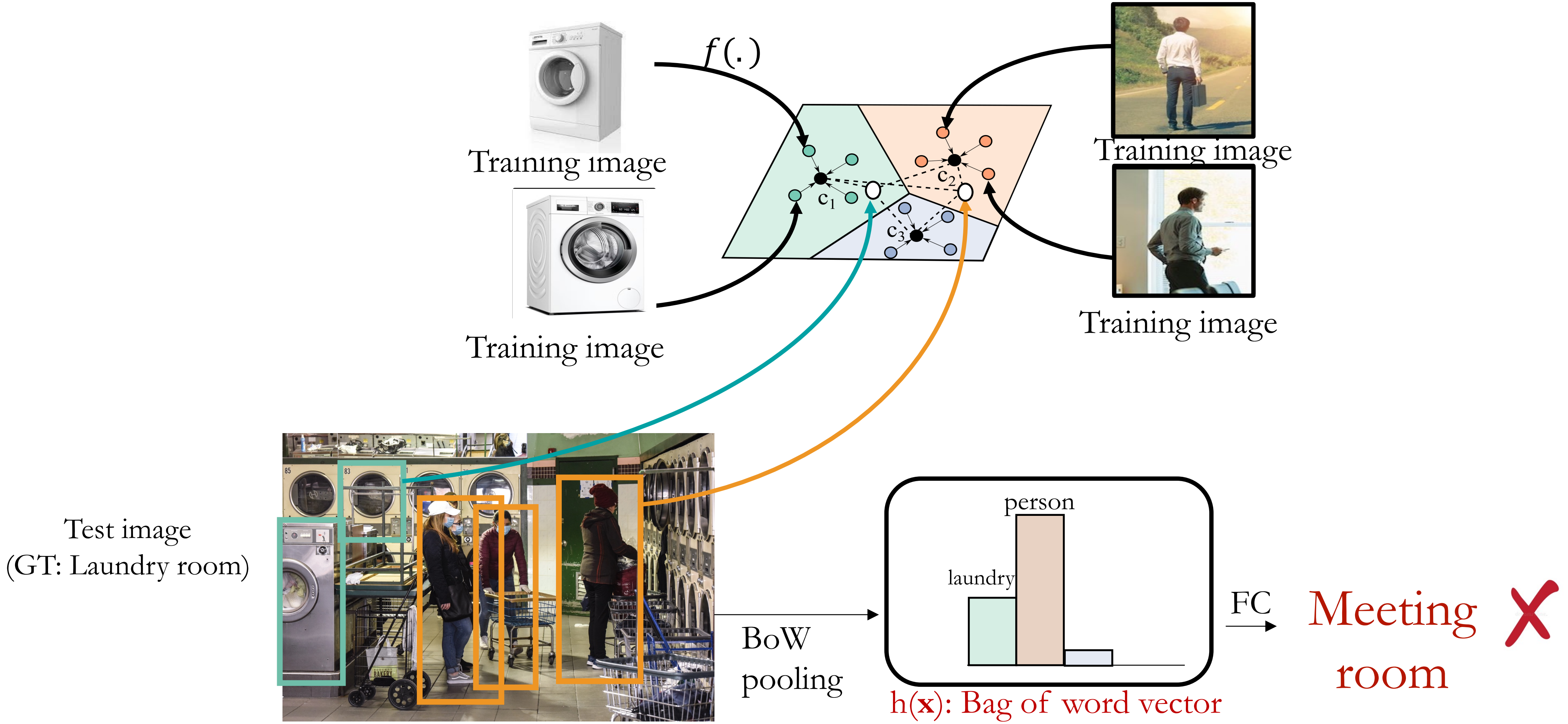
Focus areas

Interpretability

Adversarial Defense

Adversarial Attacks

Downside: all features participate in the feature aggregation step of BoW pooling

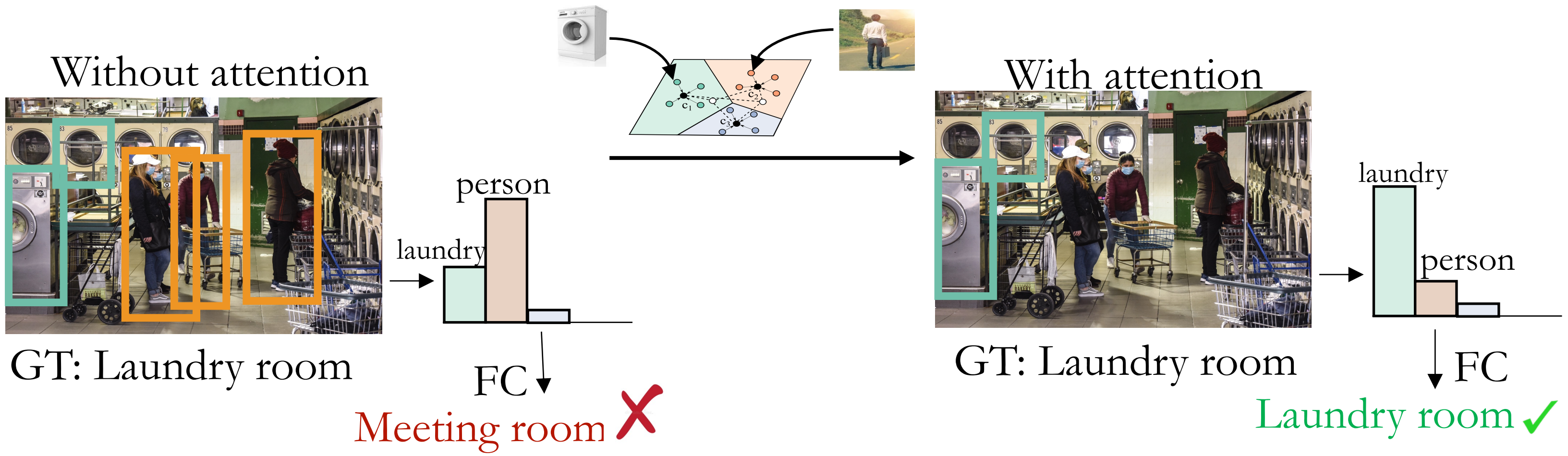


Remove the influence of non-discriminative regions. How?

Key idea: attention-aware pooling

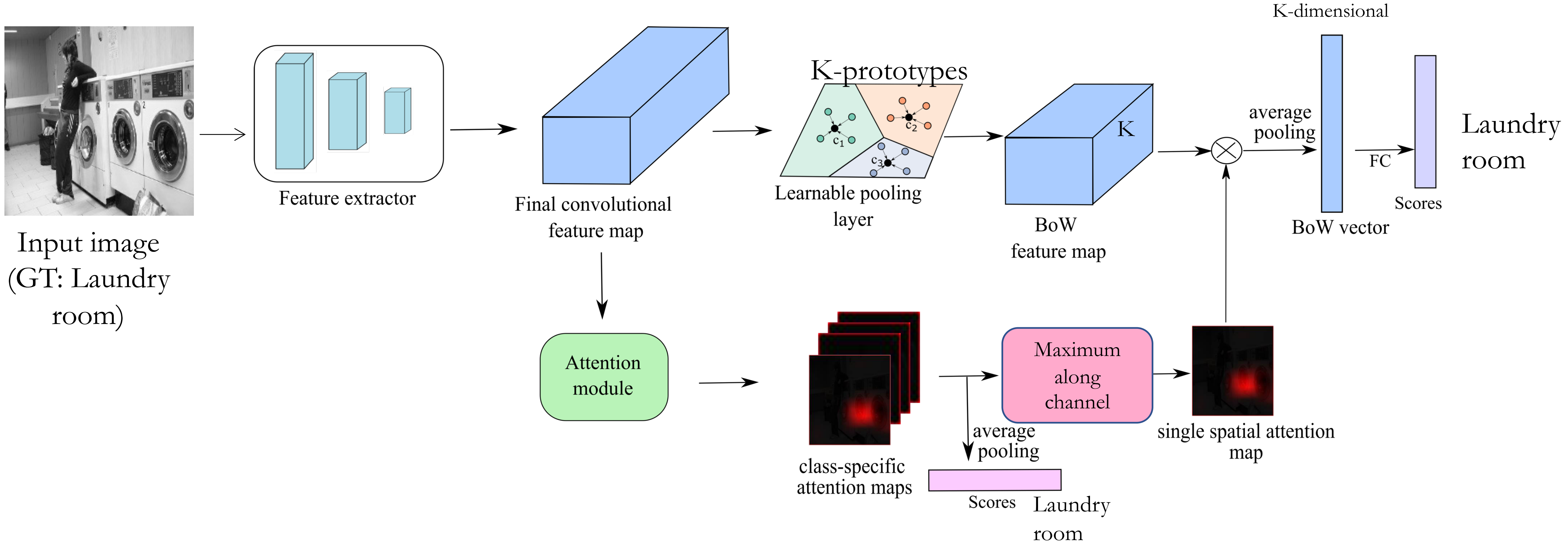
By introducing the attention during the feature aggregation process, the BoW representation becomes more discriminative

Impact:



[KKN & MS, BMVC-18]

Key idea: introduce attention in BoW pooling to remove contribution of non-discriminative features

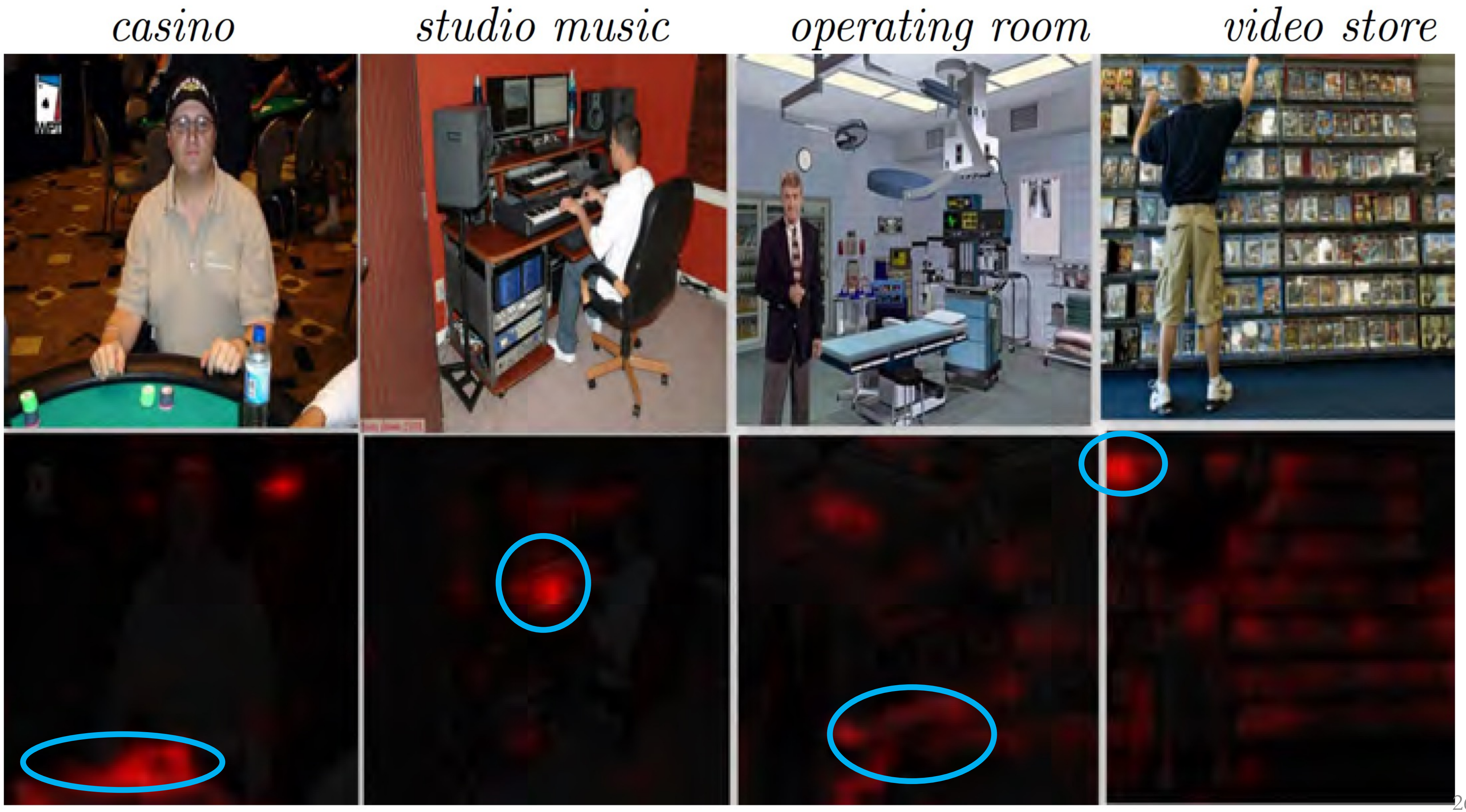


[KKN & MS, BMVC-18]

Attention ignores the non-discriminative regions (such as the person which is common across classes) and focuses on discriminative regions of the output class

Input image

Attention map



By incorporating attention, we can significantly improve the discriminative power of BoW latent representation

Accuracy↑

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				
NetVLAD	BBox	82.4	89.8	88.0	-
Attentional-NetVLAD	BBox				
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					
NetVLAD		80.6	89.4	86.4	79.2
Attentional-NetVLAD					

4%

Attention helps the most in BoW pooling

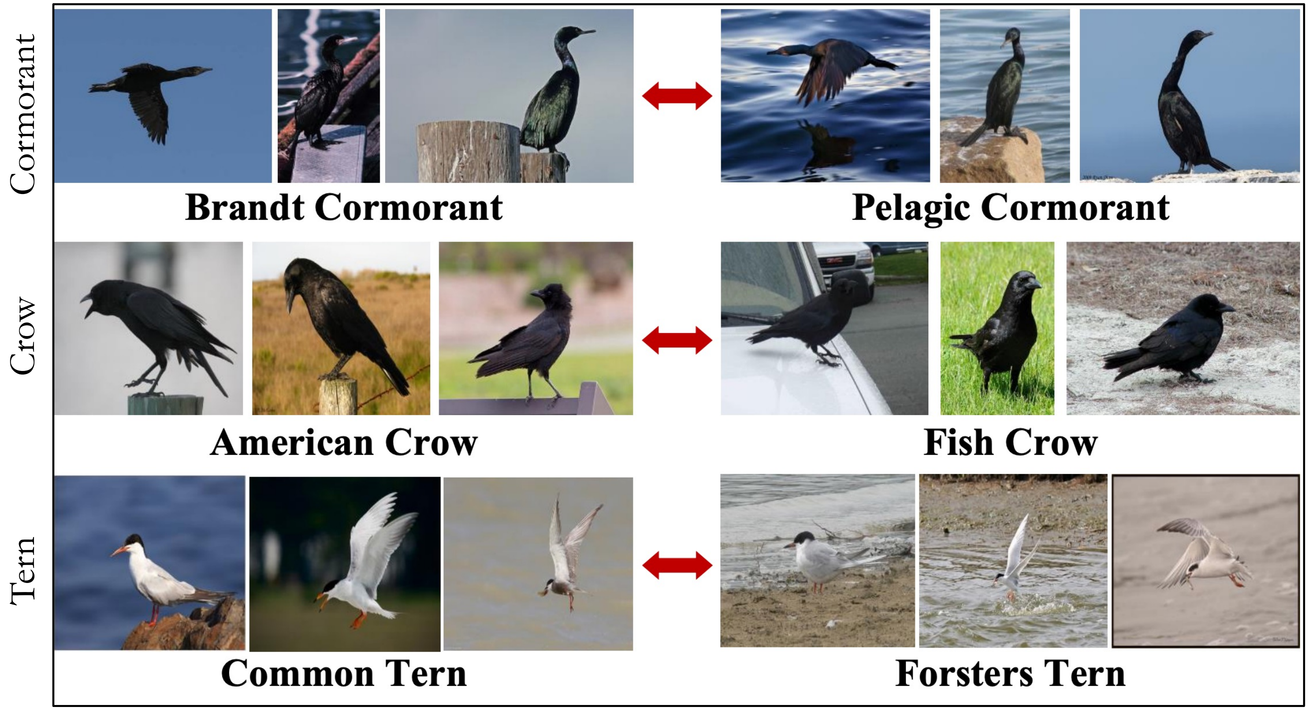
8%

[KKN & MS, BMVC-18]

The reasons for success of adversarial attacks in fine-grained datasets has lot of subtleties

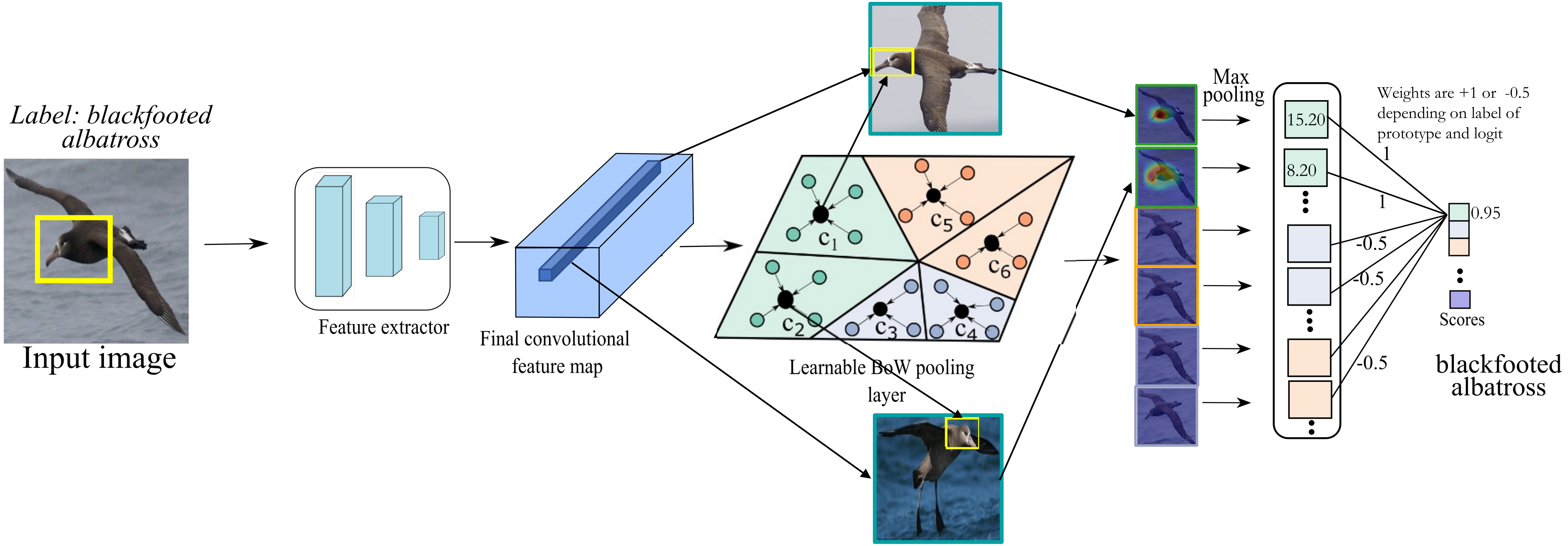
Benefits with fine-grained datasets

- Understand the DNNs workings at local patch level instead of global object level
- More sensitive to attacks since local perturbations can change the label



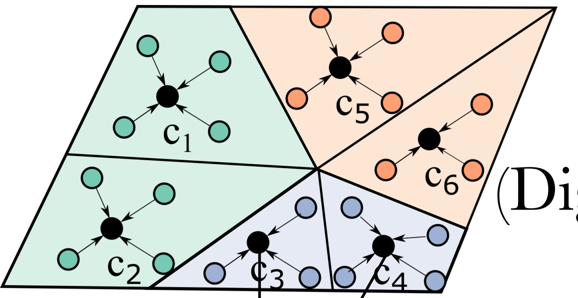
Example images of some confusing classes

ProtoPNet: Classify image based on evidence from local patches



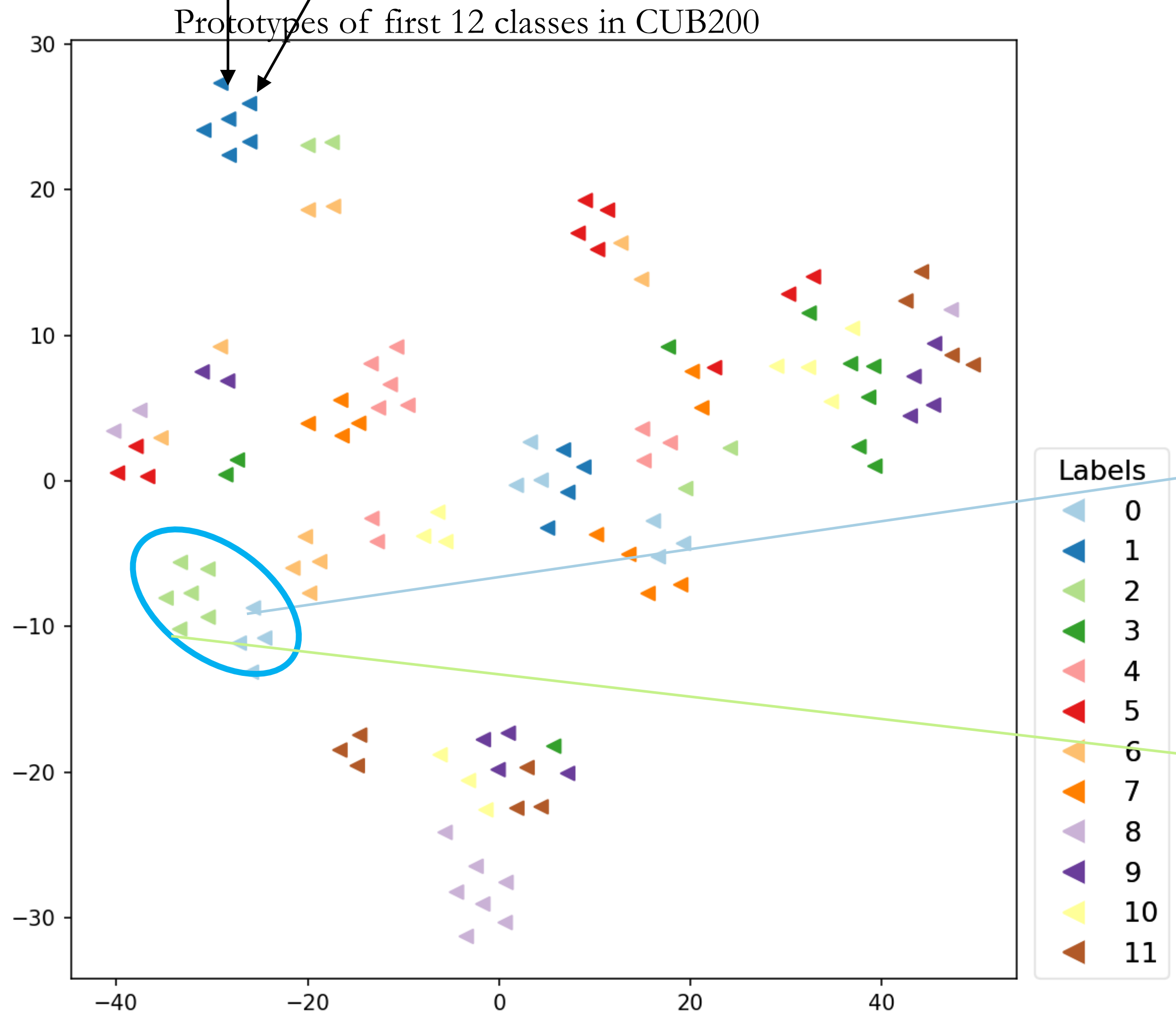
Yellow box denotes the visual representation of prototype patch along with full training image from which the patch is extracted

ProtoPNet [Chen et al. NeurIPS 2019]



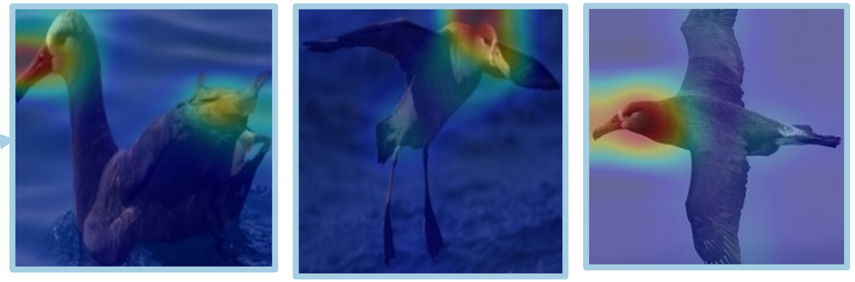
t-SNE visualization of learned prototypes

(Digging deeper: understanding reasons for success of attacks)



Foreground prototypes of different classes of same family are close to each other

Class 0 : Blackfooted albatross

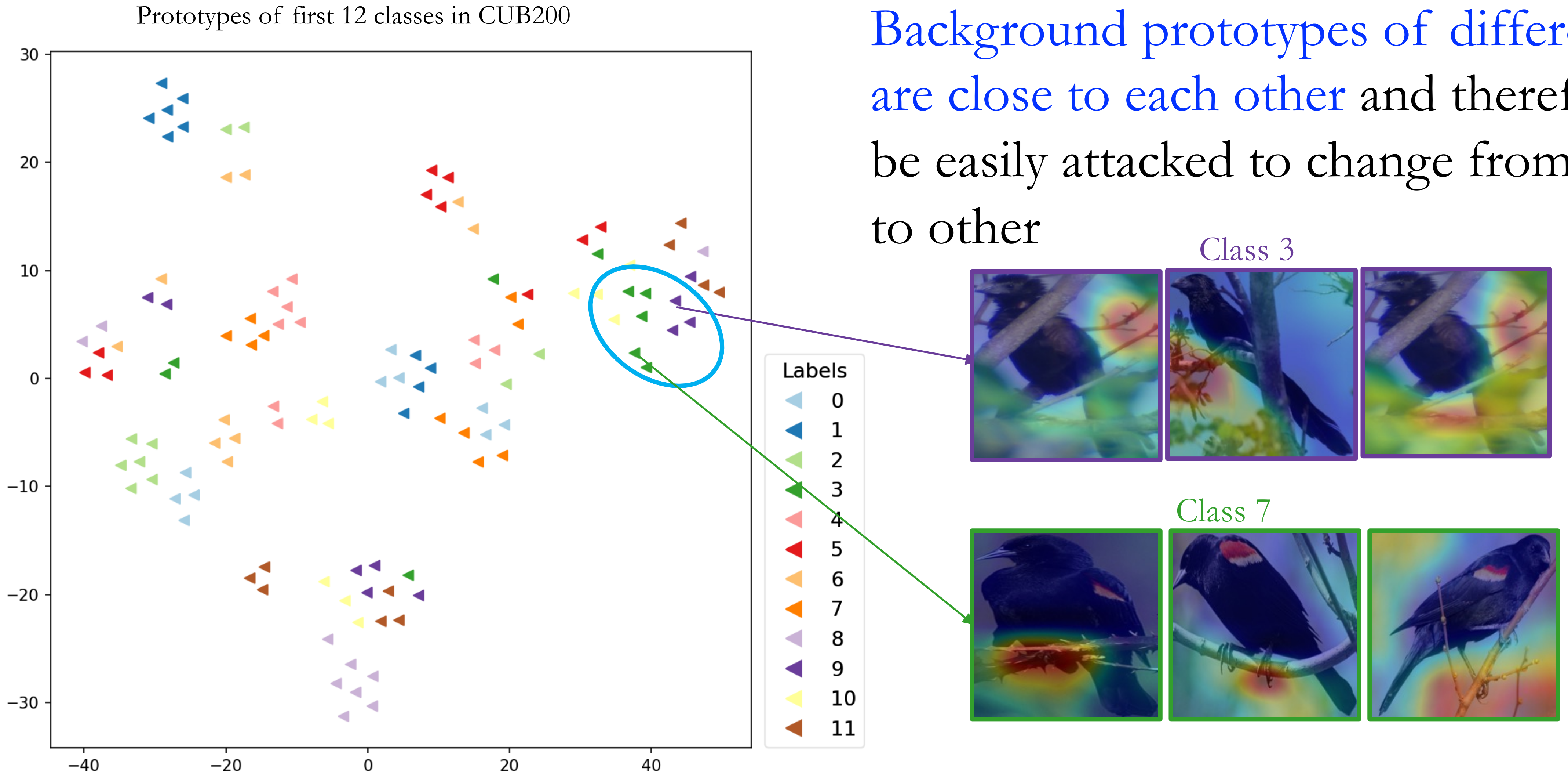


Class 2: Sooty albatross

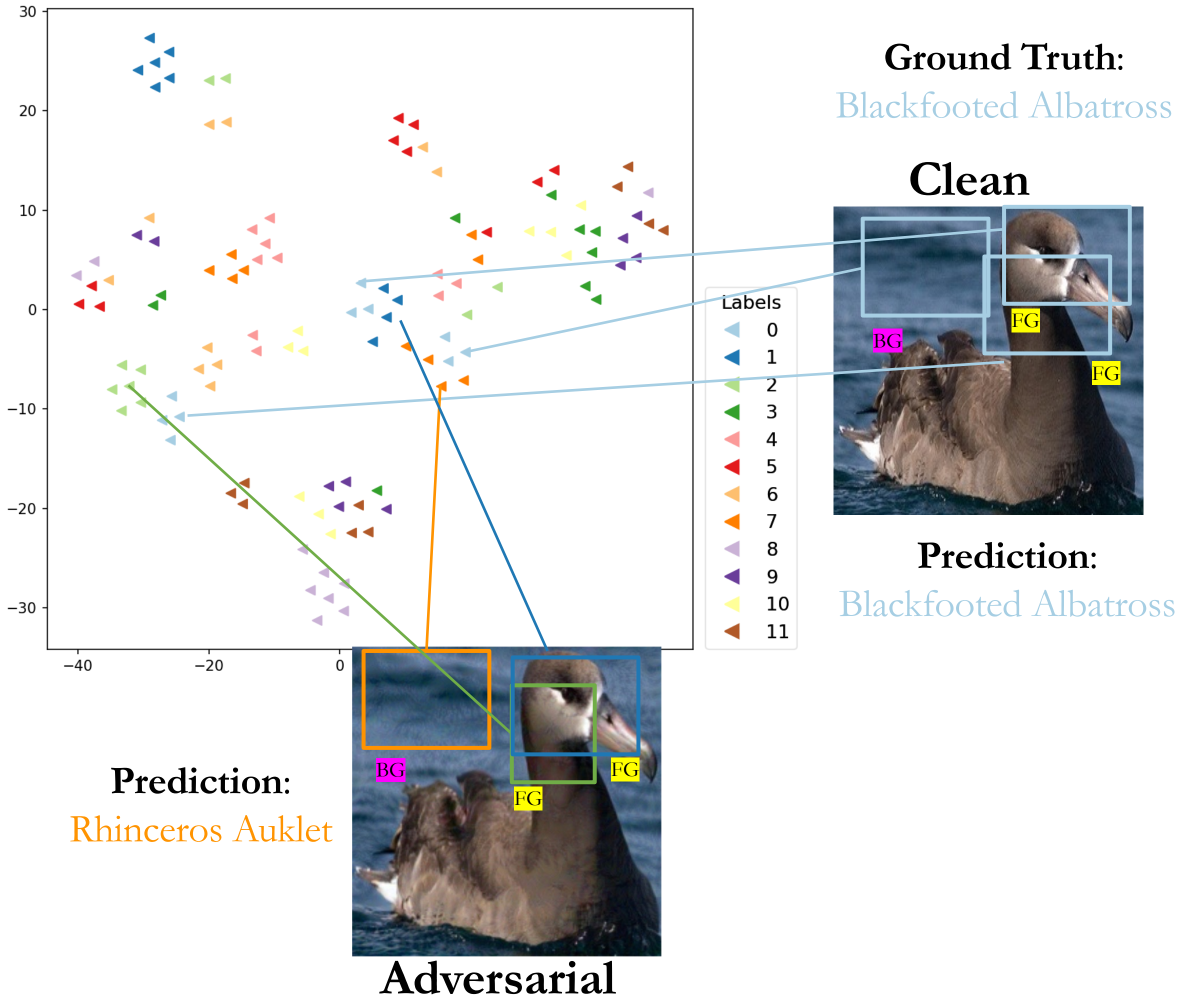
t-SNE visualization of learned prototypes

(Digging deeper: understanding reasons for success of attacks)

Background prototypes of different classes are close to each other and therefore can be easily attacked to change from one class to other



An intuitive example to understand the success of adversarial attacks on ProtoPNet



How can these observations help to improve the robustness?

maximal separation of discriminative features

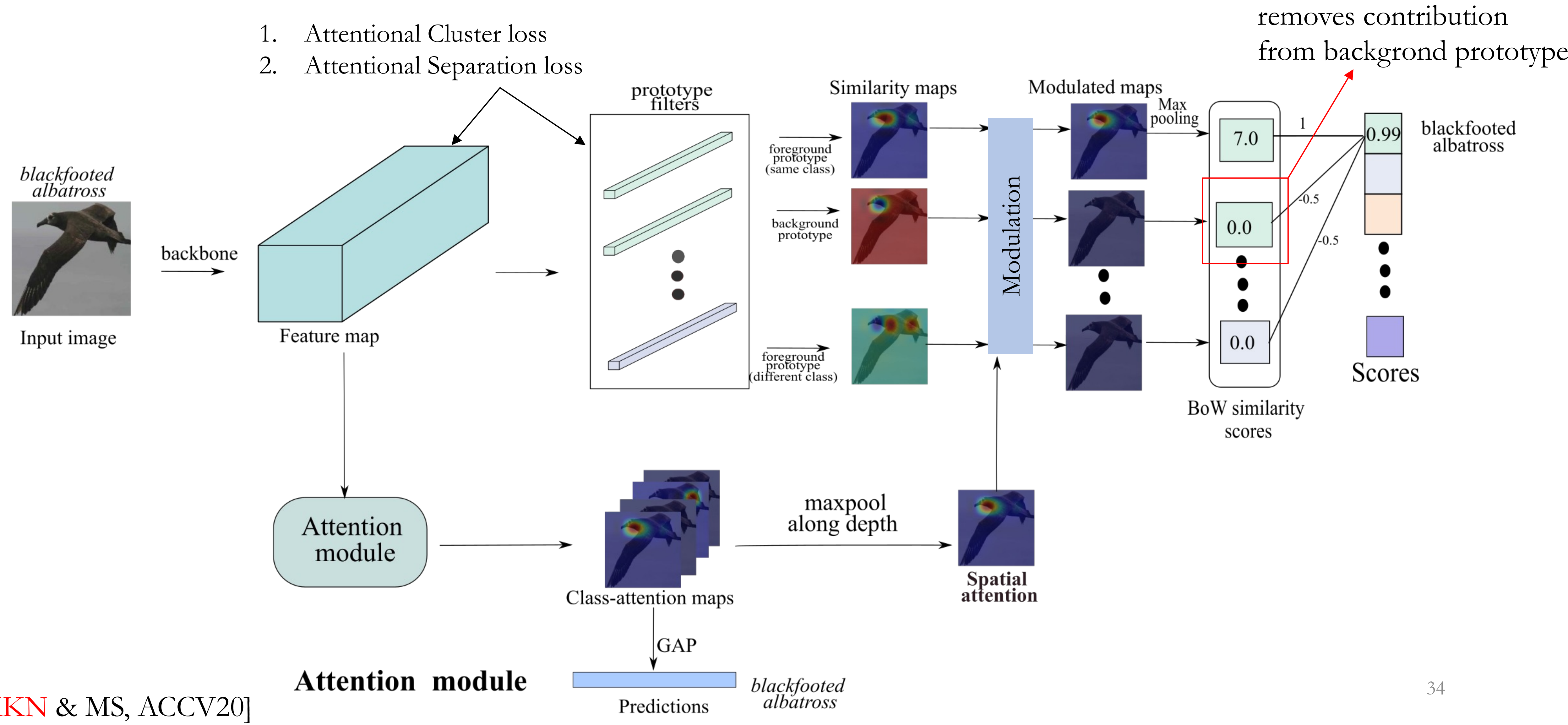
Intuition

If we can maximally separate the latent features of foreground (discriminative) regions of different classes and also remove the influence of background regions in the decision process, then we have made the attacker task difficult to conduct attacks

1. Attentional cluster loss - pulls the high-attention regions in a sample close to the nearest prototype of its own class
2. Attentional separation loss - pulls the high-attention regions in a sample away from the nearest prototype of other class

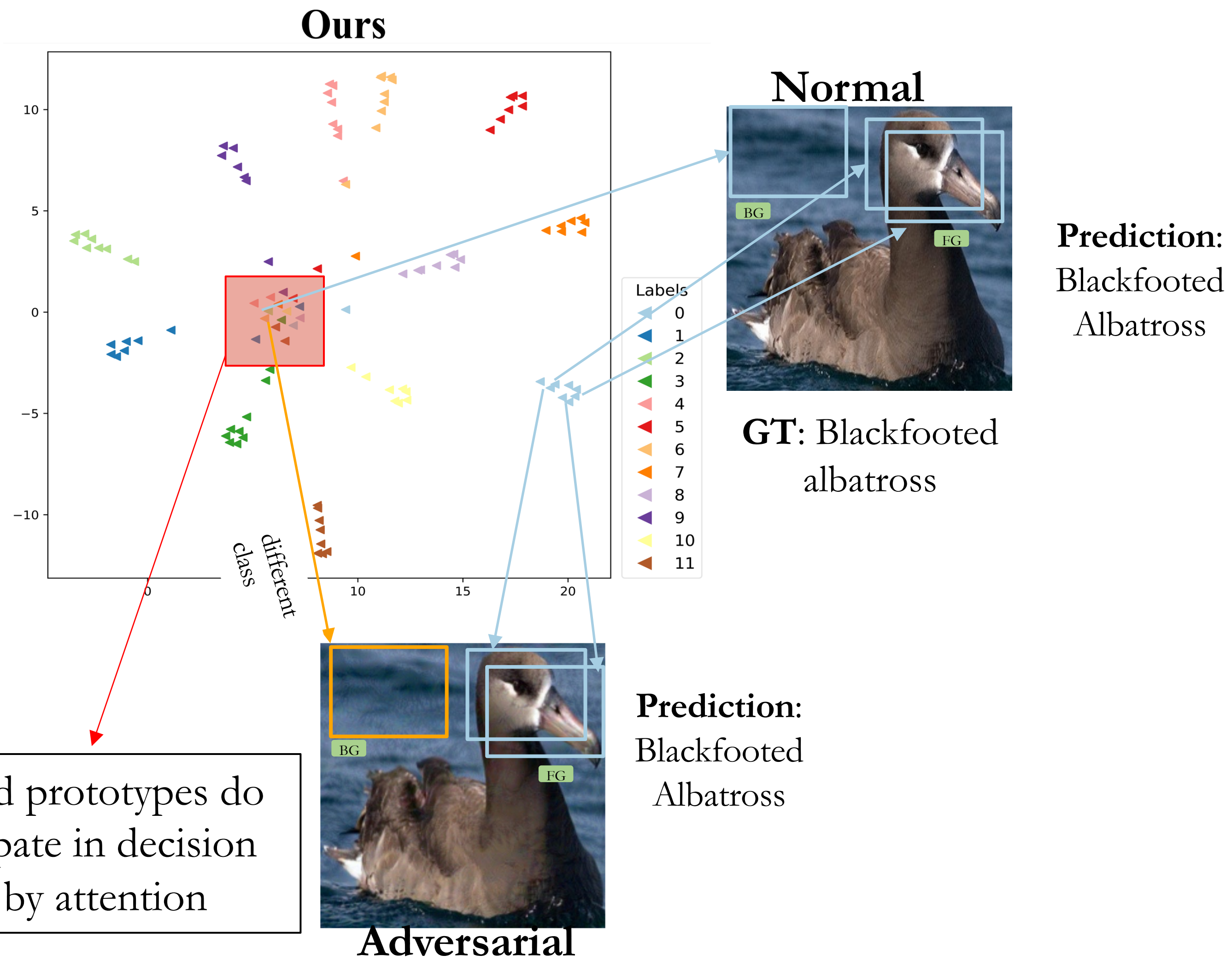
[KKN & MS, ACCV20]

Attention-aware architecture



[KKN & MS, ACCV20]

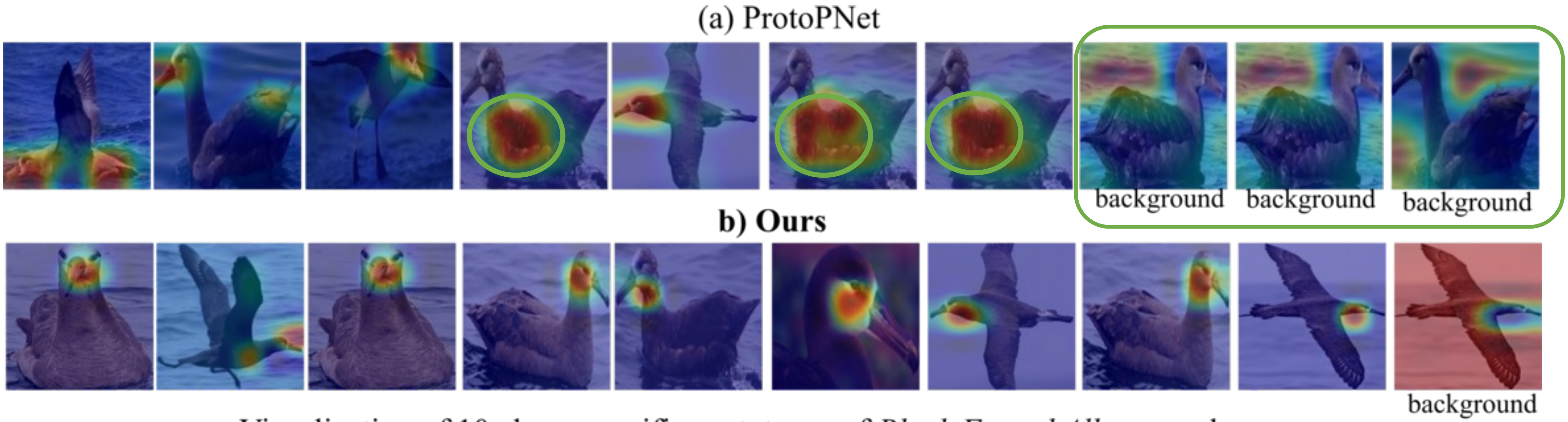
Our method yields well-separated foreground prototypes while clustering background prototypes



Background prototypes do not participate in decision process by attention

Visualization of learned Prototypes

(our prototypes are fine-grained and complete non-discriminative regions activated by background prototype)



Visualization of 10 classs specific prototypes of *Black Footed Albatross* class

Our adversarial training strategy with novel losses consistently achieves higher performance against white-box attacks

Accuracy↑

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)
VGG-16	AP* [71]	54.9%	44.9%	24.2%	41.9%	18.2%	41.2%	16.9%	41.9%	18.7%
	AP+PCL* [183]	60.7%	50.5%	28.5%	47.1%	22.8%	46.7%	21.6%	47.2%	23.5%
	Ours-A*	**69.3%**	**56.1%**	**34.8%**	**51.7%**	**29.6%**	**50.8%**	**28.0%**	**52.0%**	**32.5%**
	ProtoPNet* [29]	60.1%	44.5%	26.9%	**57.1%**	10.9%	35.9%	10.3%	37.6%	13.5%
	Ours-FR*	**63.0%**	**53.3%**	**37.3%**	49.4%	**30.4%**	**48.1%**	**28.6%**	**49.7%**	**31.1%**
VGG-19	AP* [71]	58.0%	47.5%	29.1%	44.3%	25.6%	44.0%	24.34%	44.4%	26.2%
	AP+PCL* [183]	61.8%	52.1%	30.9%	48.9%	24.7%	48.6%	23.3%	49.1%	25.4%
	Ours-A*	**68.2%**	**57.1%**	**36.5%**	**53.2%**	**30.4%**	**52.6%**	**29.2%**	**53.5%**	**31.2%**
	ProtoPNet* [29]	55.1%	40.0%	28.9%	26.5%	11.3%	29.7%	9.60%	25.6%	10.2%
	Ours-FR*	**64.4%**	**55.5%**	**37.4%**	**51.2%**	**30.6%**	**50.4%**	**28.7%**	**52.1%**	**32.3%**

Black-box auto-attack ensemble on adversarial trained models

Accuracy ↑

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%**

ϵ=8

Focus areas

Interpretability

Adversarial Defense

Adversarial Attacks

Adversarial Attacks

- Attacks beyond image recognition
  - White-box attacks on semantic segmentation
  - Black-box transfer attacks on visual object tracker

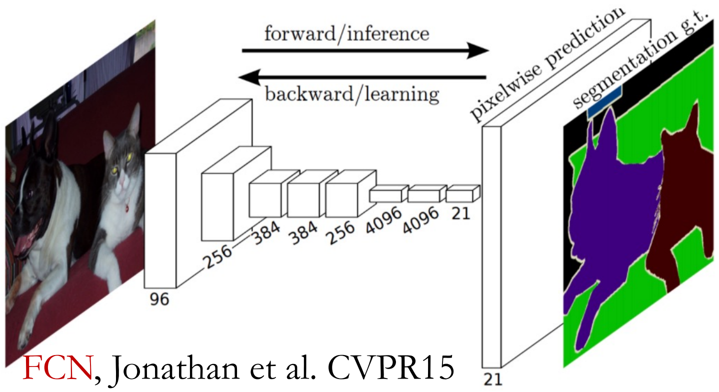
- Improving the transferability of attacks
  - Learning transferable transferable perturbations

- How can we use adversarial attack to improve DNNs
  - Semantic adversarial attacks to study disentanglement

Exploiting context to understand the susceptibility of DNNs for Semantic Segmentation

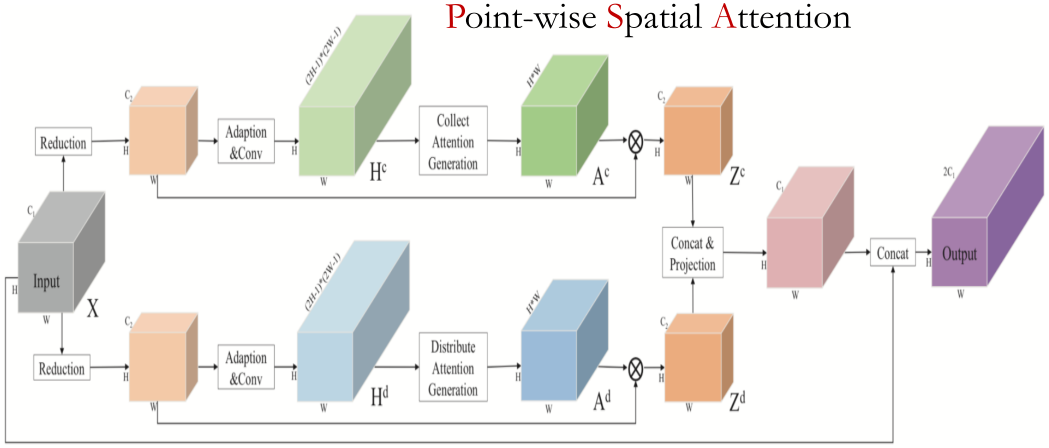
Understanding context is the core building block in modern segmentation networks

Fully Convolutional Network



FCN, Jonathan et al. CVPR15

Point-wise Spatial Attention



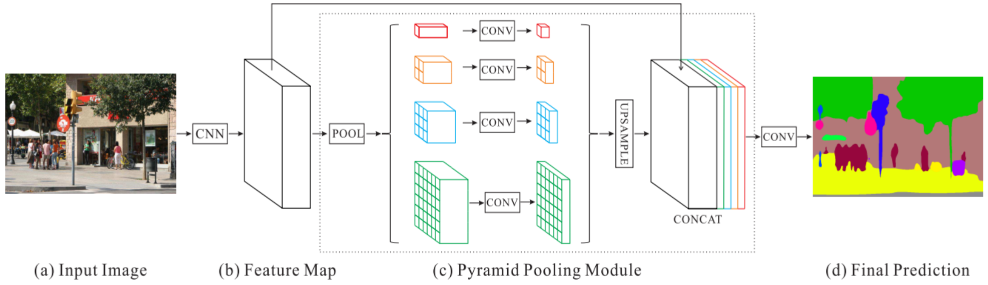
PSANet, Jonathan et al. ECCV18

no-context

heavy

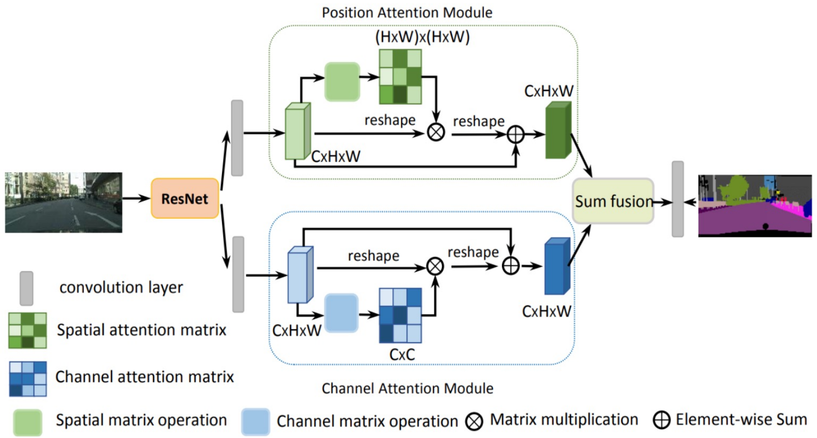
Context

PSPNet, Jonathan et al. CVPR18



Pyramid Scene Parsing

DANet, Jonathan et al. CVPR19



Dual (Spatial and channel) Attention

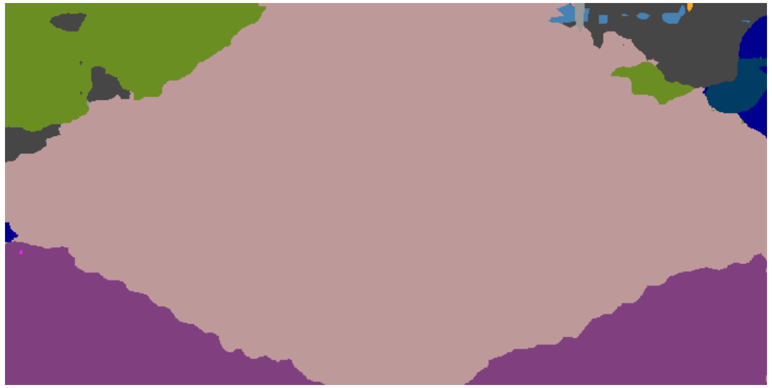
Exploiting context in semantic segmentation models

Key finding

We discover that context empowers the attackers to fool objects far away from the perturbed area. For example, a perturbation of size 4% of image area fools the prediction at 60% of image area for PSANet.

Image perturbed with 9%

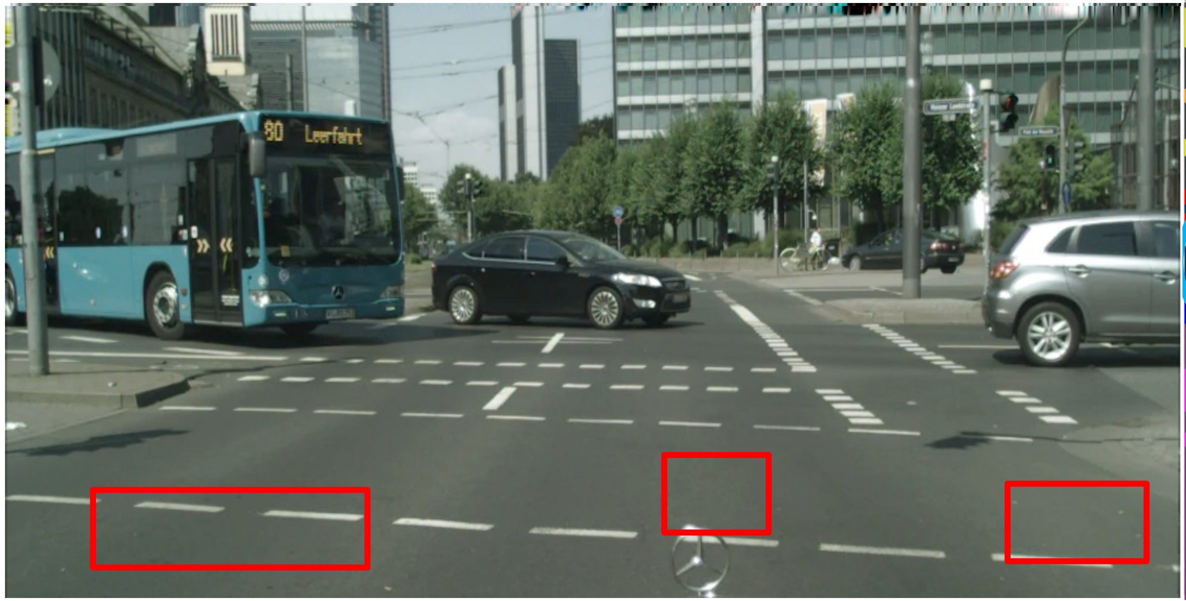
Clean prediction



Adversarial prediction

[KKN & MS, ECCV20]

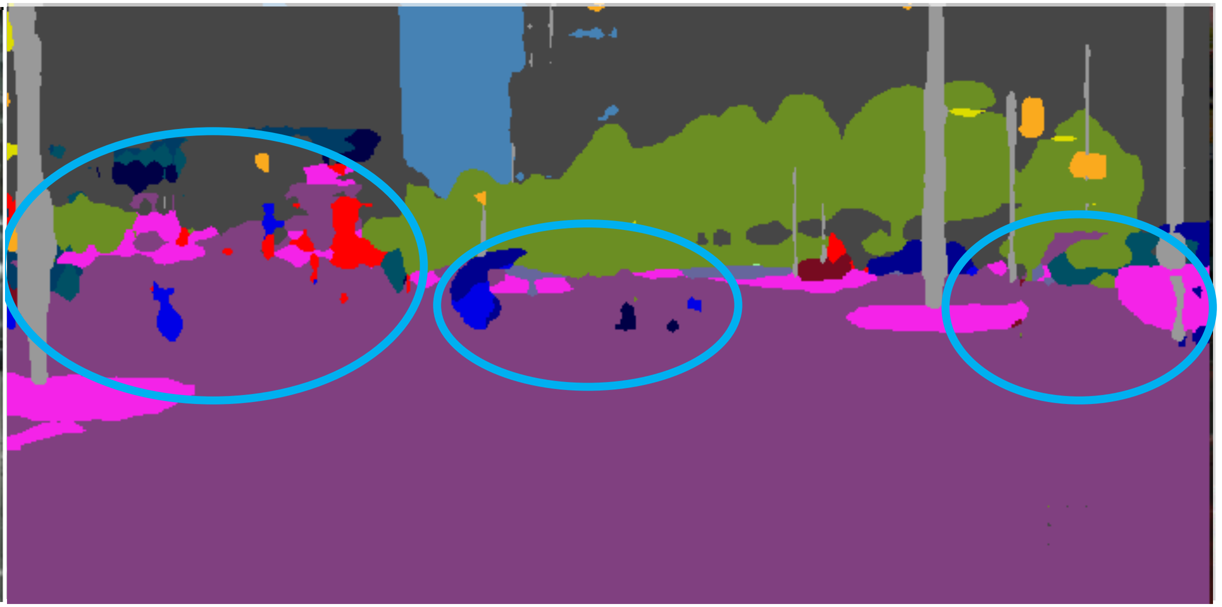
Perturbing static regions (e.g., road, sidewalk) affects the predictions at far away dynamic regions (e.g., bus, pedestrians) in inconspicuous way



Clean image

Adversarial image

Realistic-looking segmentation map



Clean predictions

Adversarial predictions

Indirect local attack in red box regions

Fools the distant dynamic objects

[KKN & MS, ECCV-20]

Context-aware networks are highly vulnerable to indirect attacks than FCN

Adversarial image with local perturbations

Clean segmentation

FCN

PSANet

PSPNet

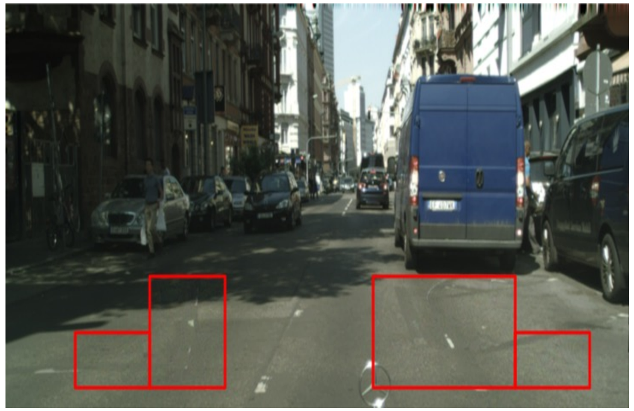
DANet

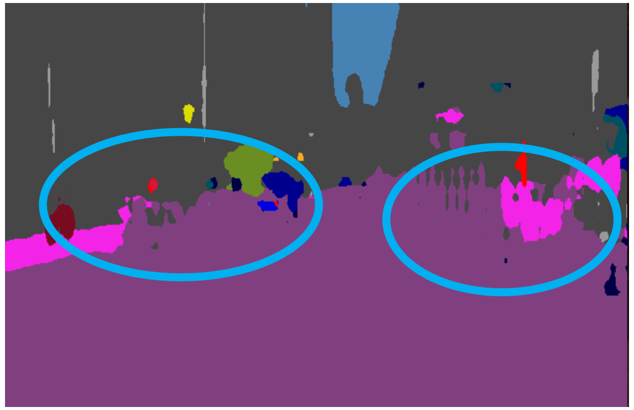
Indirect adaptive attack results

Adversarial image

Clean predictions

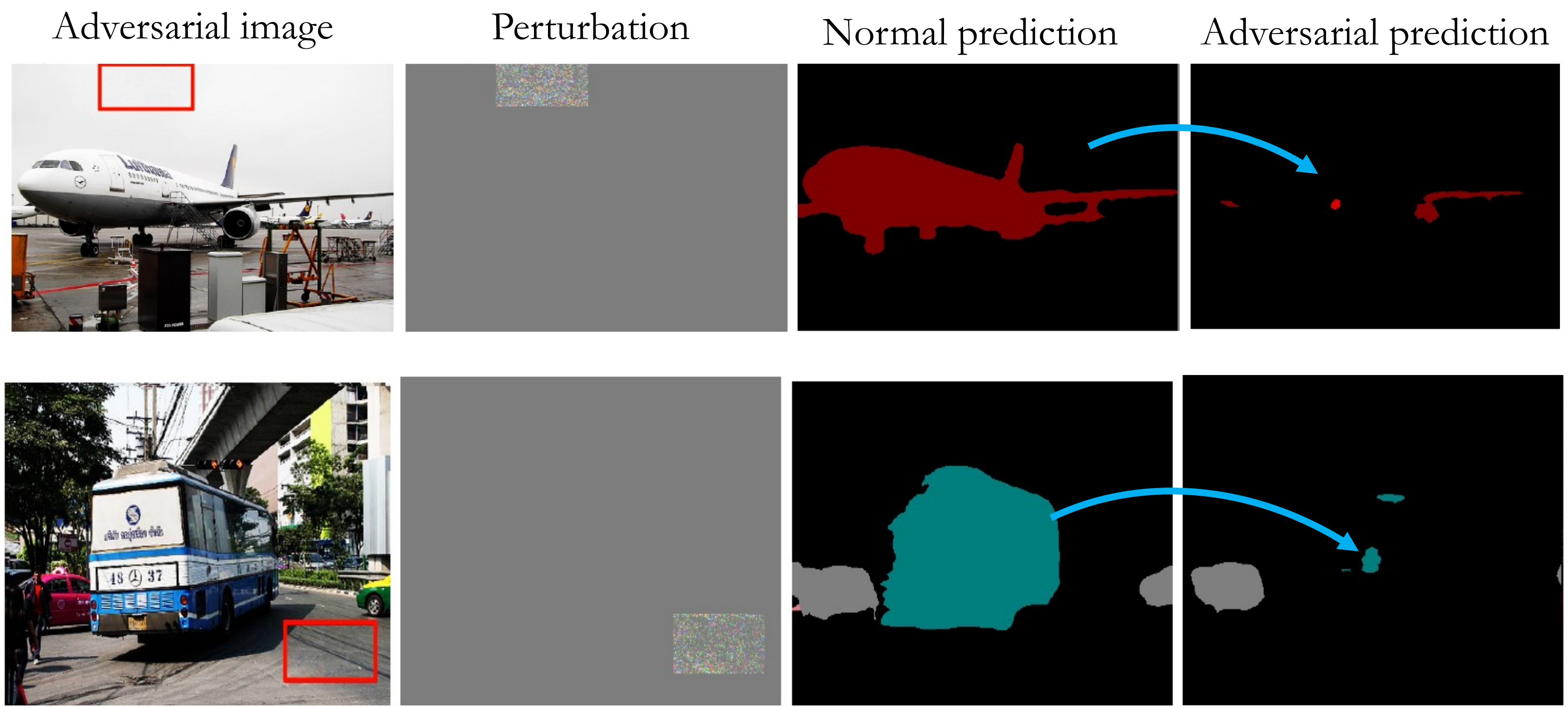
Adversarial predictions





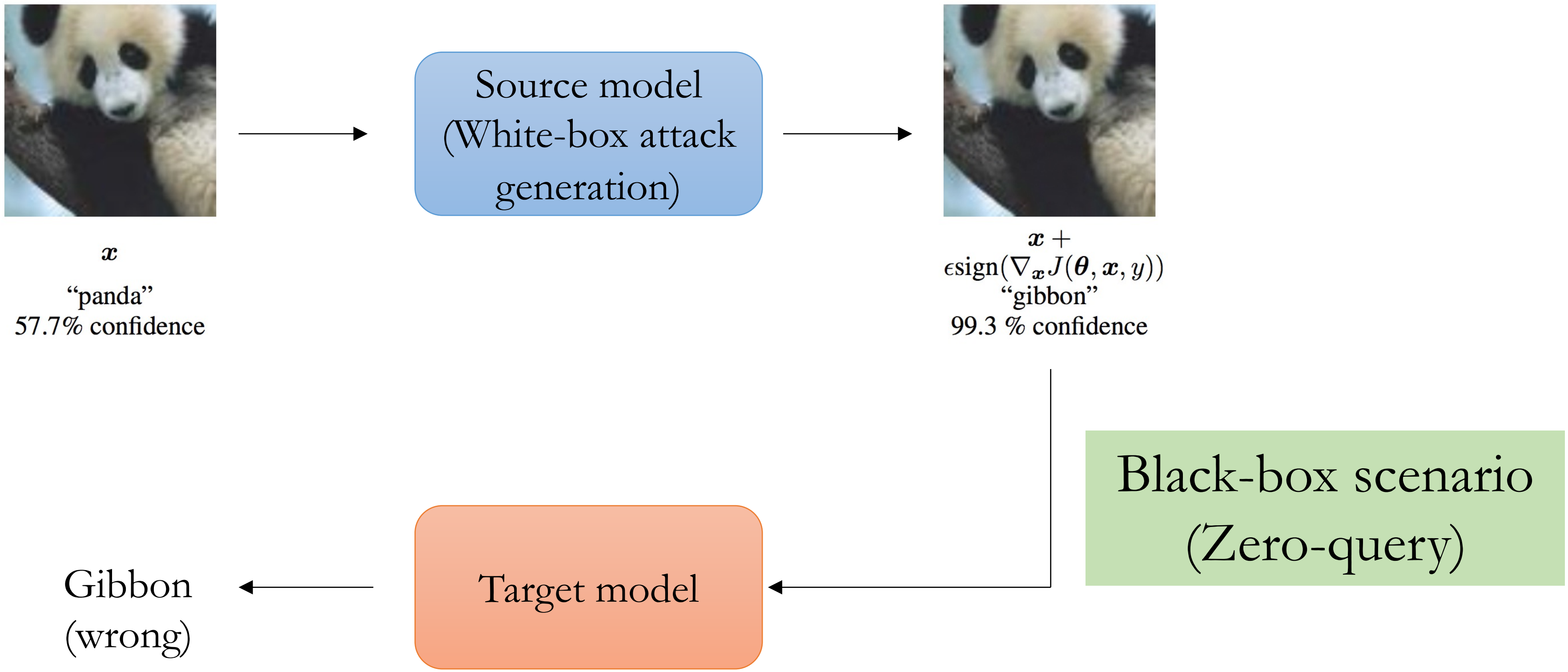
Perturbed inside red boxes

Another perspective: Indirect attacks helps to understand contextual dependencies in DNNs (e.g. sky-aeroplane, road-car, road-bus)



Transferable Adversarial Examples

(Perturbation generated from one network transfers to other network)



Transferable perturbations require no access to the target model

Adversarial Attacks

- Attacks beyond image recognition
  - White-box attacks on semantic segmentation
  - **Black-box transfer attacks on visual object tracker**

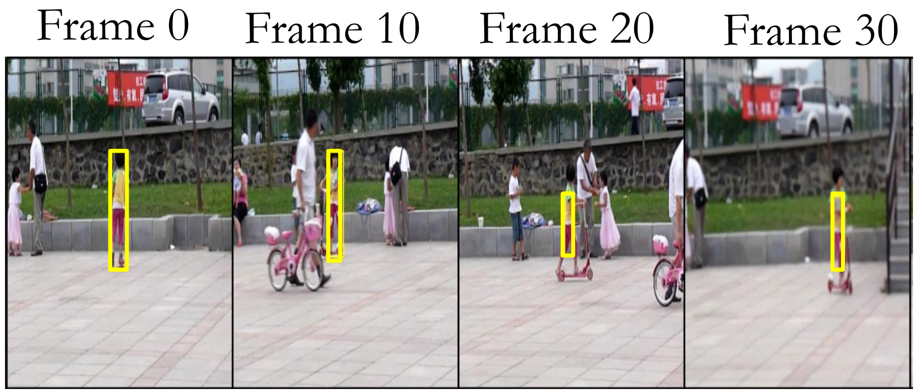
- Improving the transferability of attacks
  - Learning transferable transferable perturbations

- How can we use adversarial attack to improve DNNs
  - Semantic adversarial attacks to study disentanglement

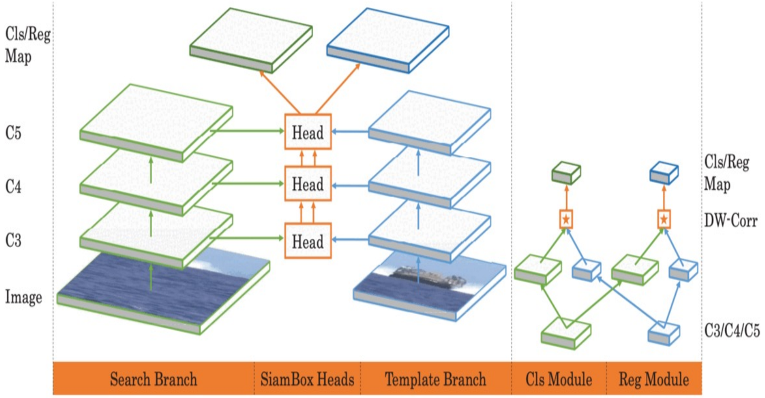
Challenges in transfer-based black-box attacks on object trackers

(VOT takes template as input and detects it in all subsequent search images)

Novel objects at test time (non-overlapping with training objects)



Computing perturbation per frame should be efficient as trackers work at real-time



Should generalize to different tracker frameworks such as SiamCAR, SiamBAN, OCEAN

Goal: Efficient black-box attacks on visual object tracking

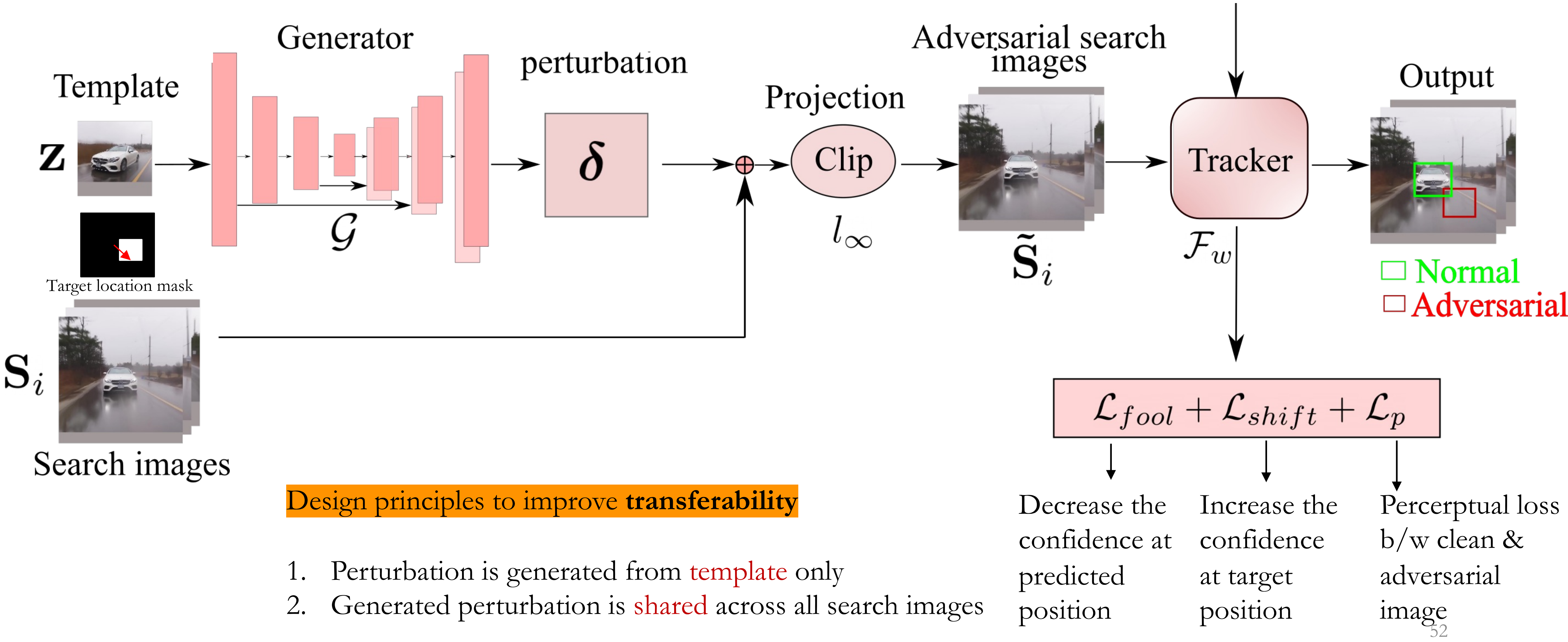
Key idea

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.

Impact:

Learns to generate powerful transferable perturbations on unknown videos and trackers

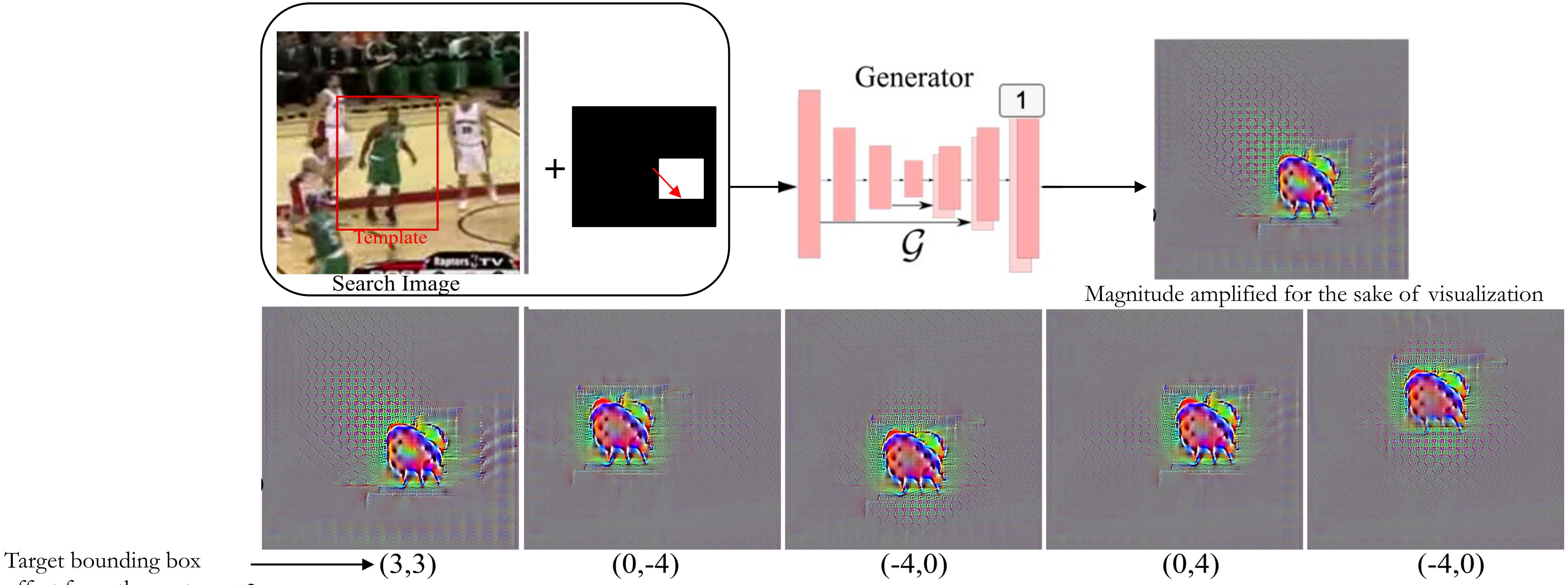
Temporally-transferable perturbation generator



Design principles to improve transferability

1. Perturbation is generated from template only
2. Generated perturbation is shared across all search images

Visualization of generated directional perturbations



Generated perturbation contains adversarial object-like patch at target position

Targeted adversarial attacks to steer the target tracker to follow the object at a fixed offset

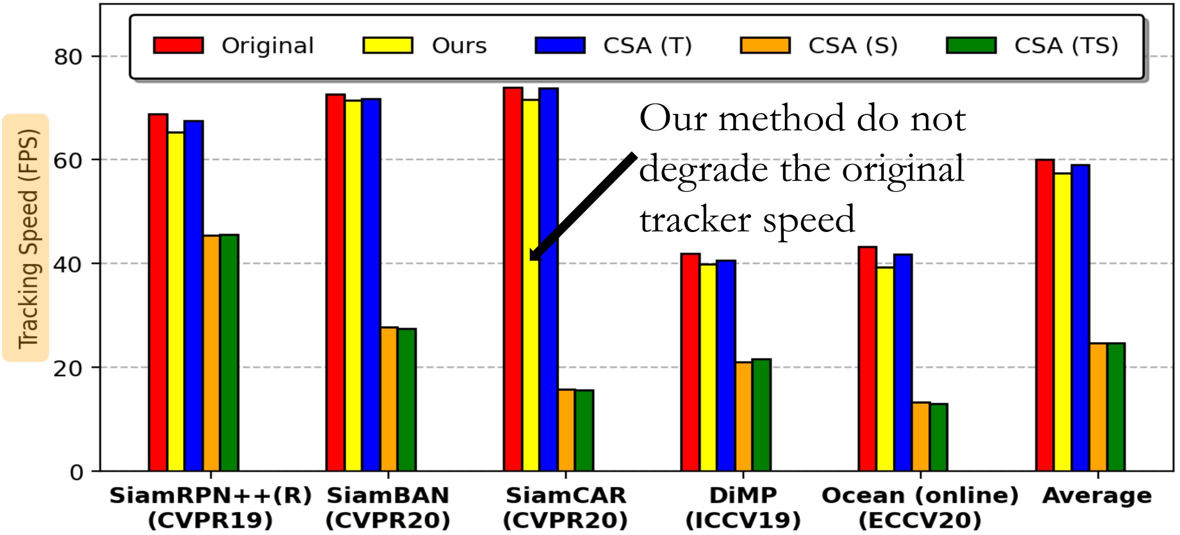
Using 12 precomputed directional perturbations

Ground truth

Adversarial prediction

Adversarial target

Our transfer-attacks are highly efficient and effective too



Y-axis is **tracker speed** and x-axis is different tracker frameworks

Generator trained on SiamRPN++ (ResNet50) **S**: Success; **P**: Precision

Methods	SiamRPN++ (M) S (↑)	SiamRPN++ (M) P (↑)	SiamBAN S (↑)	SiamBAN P (↑)	SiamCAR S (↑)	SiamCAR P (↑)	DiMP S (↑)	DiMP P (↑)	Ocean online S (↑)	Ocean online 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**

Performance of Ours (Universal perturbation generated from single fixed template) and Ours (TD) (perturbations generated from template of given input video) method are at same range

Adversarial Attacks

- Attacks beyond image recognition
  - White-box attacks on semantic segmentation
  - Black-box transfer attacks on visual object tracker
- Improving the transferability of attacks
  - **Learning transferable transferable perturbations**
- How can we use adversarial attack to study DNNs
  - Semantic adversarial attacks to study disentanglement

Understanding and Improving the Transferability of Generative Adversarial Perturbations

Learning Transferable Adversarial Perturbations

We investigate the transferability of generative perturbations when the conditions at inference time differ from the training ones in terms of

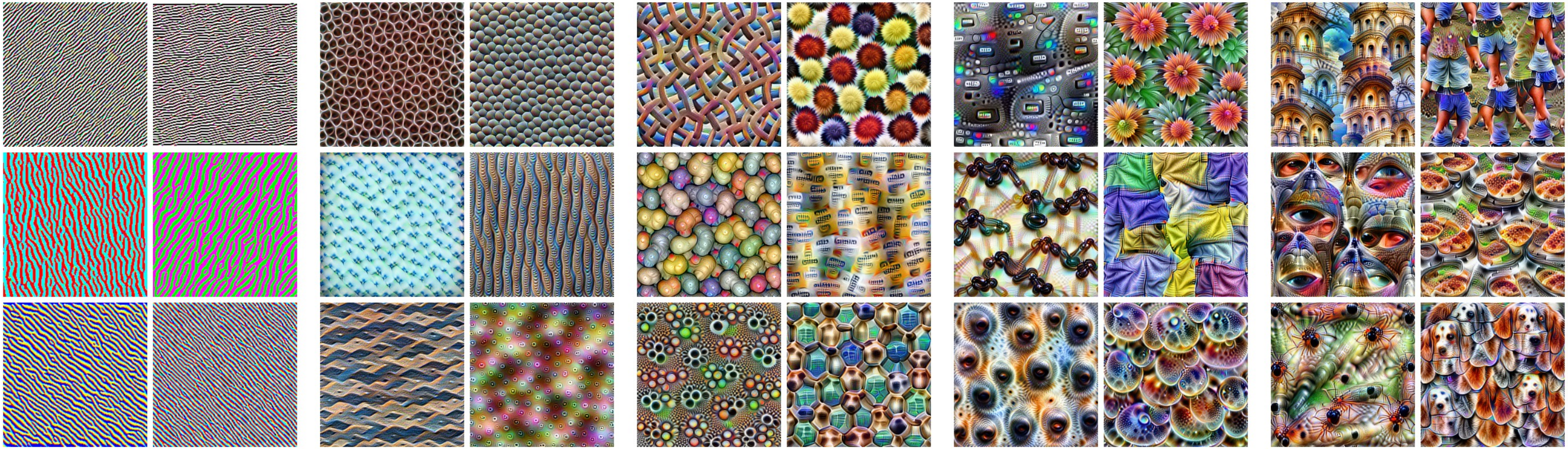
1. Target architecture — Generator was trained to attack a VGG-16 but the target network is a ResNet152

2. Target data — Generator was trained using the Paintings dataset, but the test data comes from ImageNet

3. Target task — Generator was trained to attack an image recognition model but faces an object detector at test time

Now let's take a step back and see how deep neural networks build up their understanding of images?

Bottom layers ⟷ Top layers



Edges (layer conv2d0) **Textures** (layer mixed3a) **Patterns** (layer mixed4a) **Parts** (layers mixed4b & mixed4c) **Objects** (layers mixed4d & mixed4e)

Inception filter visualization

[Credit: https://distill.pub/2017/feature-visualization](https://distill.pub/2017/feature-visualization)

Learning Transferable Adversarial Perturbations

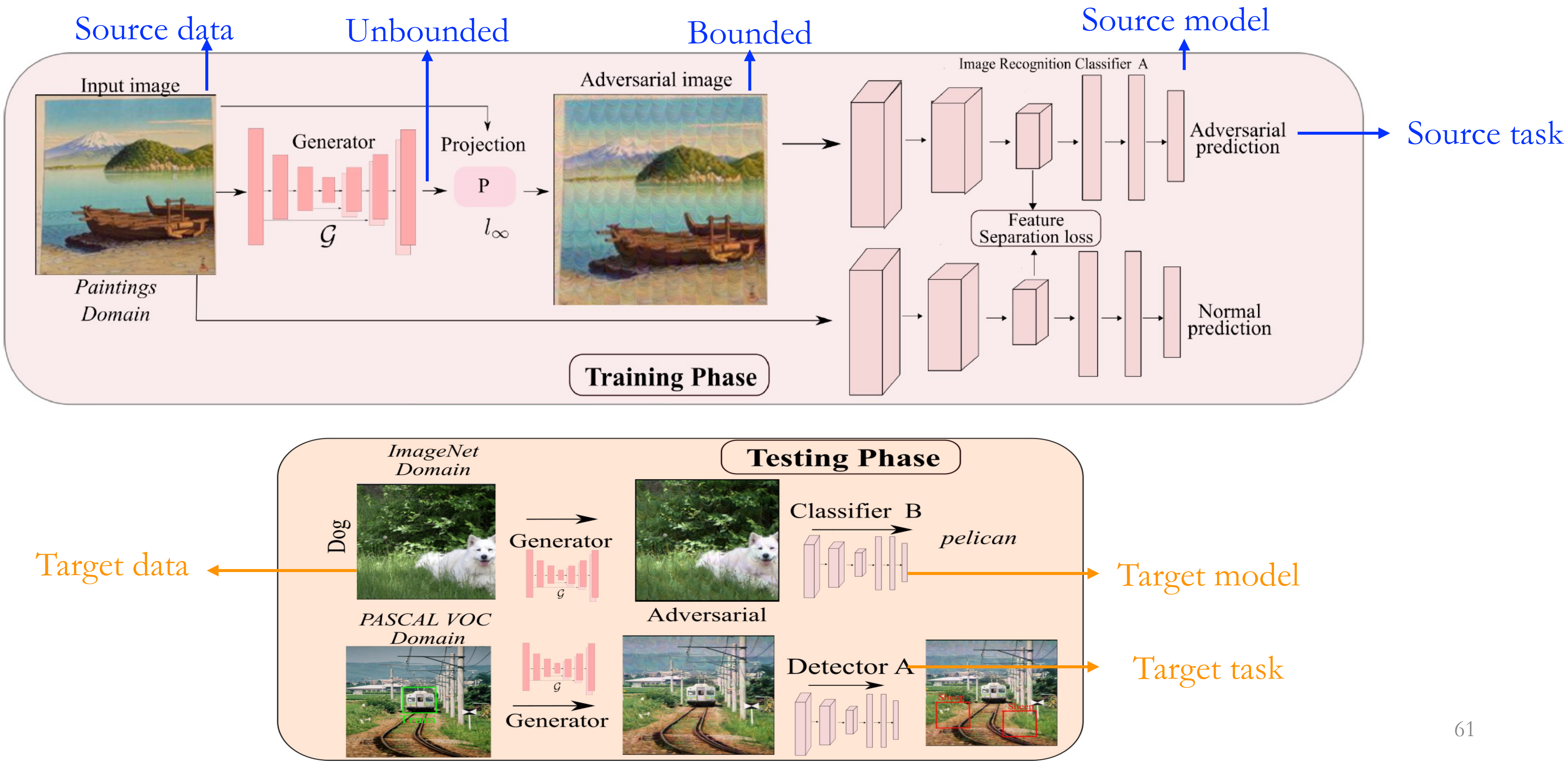
Key idea

Disrupting the mid-level features using feature separation loss empowers attackers to learn perturbations with high transfer rates across target architectures, target datasets and target tasks without any queries

Most prior works in transfer-based black-box attacks focus on only unknown architecture → Our work focuses on black-box attacks on unknown architecture, data and task

[KKN & MS, NeurIPS21]

Training a perturbation generator with mid-level feature separation loss

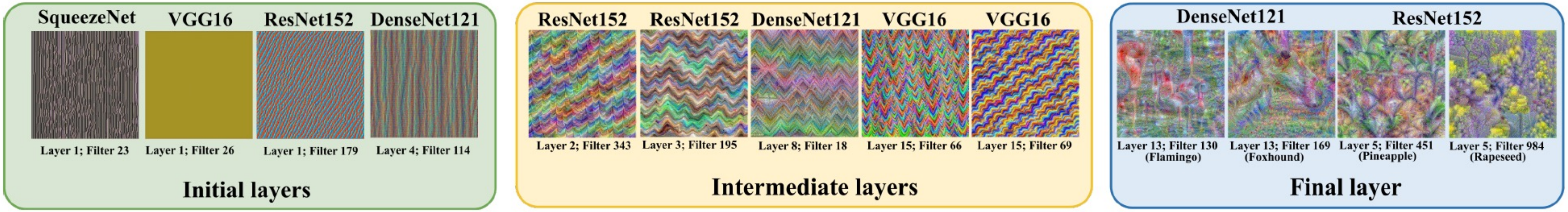


CNNs with different architectures share similar filter bank

Edges, colors

Textures

Object Patterns



Disrupting low-level filters requires larger change in input image. Further it does not transfer well to top layer to change the output prediction

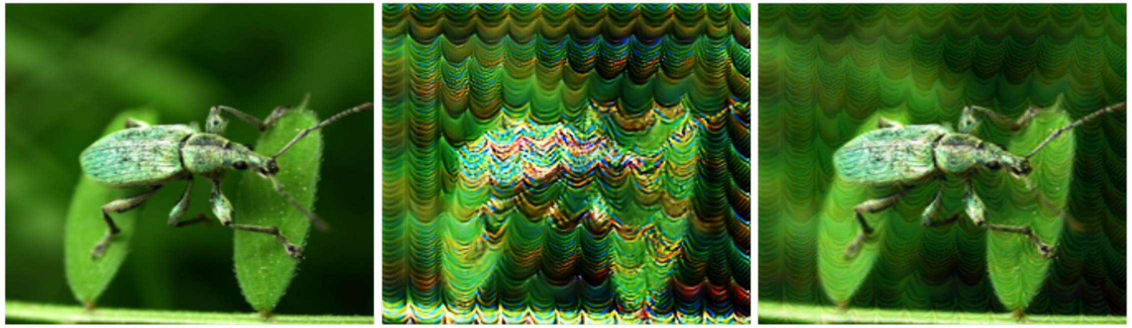
Disrupting mid-level filters helps to learn transferable perturbations across architectures

Disrupting top-level overfits to source classifier's decision boundaries

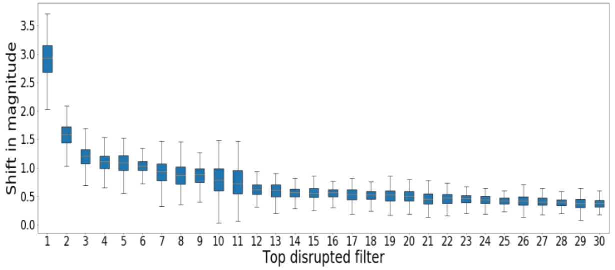
Understanding the high transfer rates from ResNet152 to VGG16

(a) White-box attack on ResNet152 (Fooling Rate: 99.7%)

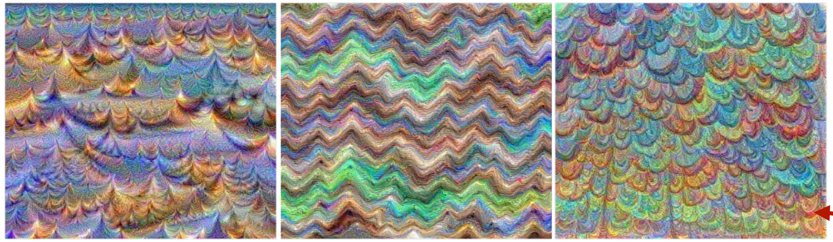
Original | Unbounded Adv. | Bounded Adv.



Top 30 disrupted filters of ResNet152



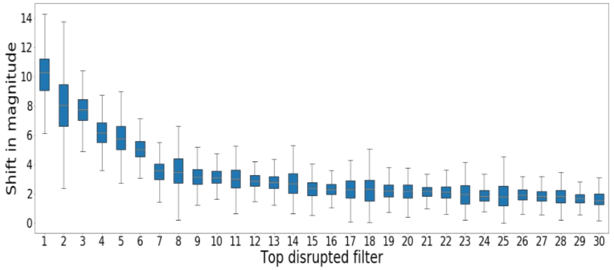
Synthesized images of few top disrupted filters in ResNet152 (Layer 3)



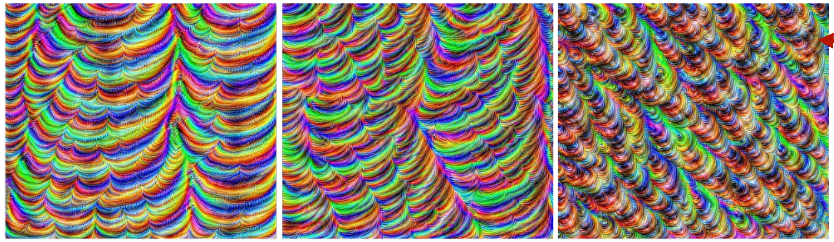
Top disrupted filters are similar. Thus, transfer rates are high between from ResNet152 to VGG-16

(b) Transfer attack from ResNet152 to VGG16 (Fooling Rate: 99.1%)

Top 30 disrupted filters of VGG16

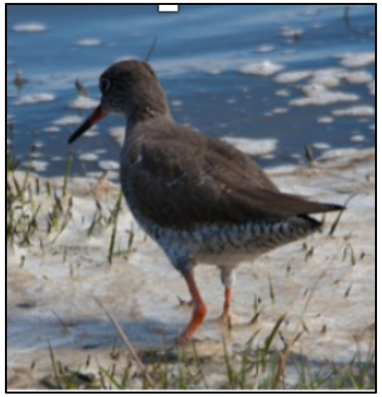


Synthesized images of few top disrupted filters in VGG16 (Layer 15)



Visualization of unbounded adversarial images with feature separation loss by attacking different layers on SqueezeNet

Input image

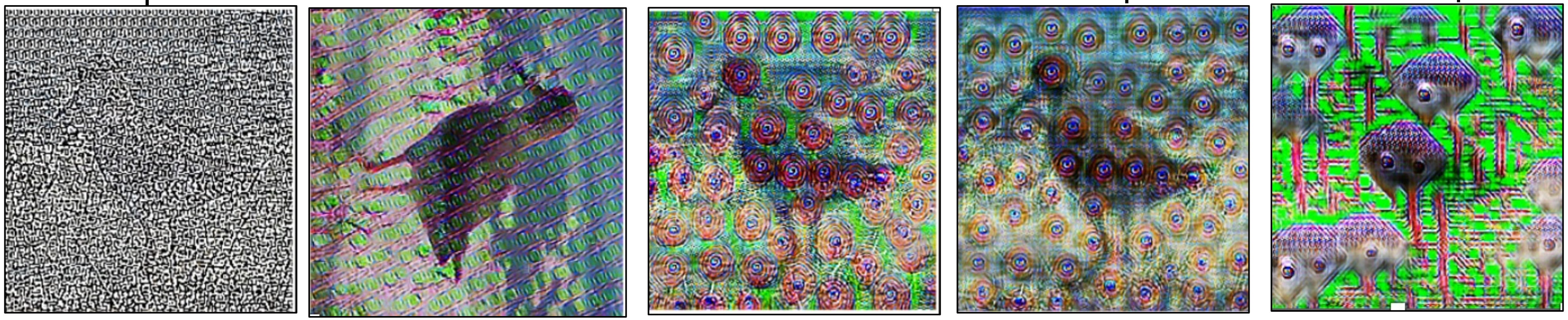


Contains low-level edge patterns

Contains textures (highest transfer rate)

Contains object-like patterns

Unbounded adversarial image



Layer 2 | Layer 5 | Layer 8 | Layer 10 | Layer 12 (Top layer)

Attacking top layers overfit to source classifier
while bottom layers require larger perturbation strength

Standard black-box transferability

(access to substitute model on target data & target data)

Fooling rate ↑

Gen. Training (data)	Discriminator	VGG16	ResNet152	Inception-v3	DenseNet121	SqueezeNet1.1	Average
			GAP [10] / CDA [11] / 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**	64.4 / 66.7 / **77.1**
	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**	84.5 / 87.9 / **92.3**
	Inception-v3	88.2 / 97.2 / **99.0**	83.4 / 82.7 / **90.4**	96.5* / 98.7* / 99.6*	89.5 / 93.6 / **96.7**	90.9 / 92.0 / **93.2**	89.7 / 92.9 / **95.8**
	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**	84.0 / 85.0 / **95.9**
	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*	70.0 / 73.0 / **86.3**
	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**	77.6 / 78.5 / **89.5**	78.5 / 81.1 / **89.5**

Source

Target

Source and Target models are trained on same ImageNet data but differ in architecture

Strict black-box transferability

(access to substitute model on target data but no target data)

Gen. Training (data)	Discriminator (ImageNet)	VGG16	ResNet152	Inception-v3	DenseNet121	SqueezeNet 1.1	Average
			GAP [10] / CDA [11] / 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 / 87.9 / **96.7**	82.4 / 82.3 / **93.2**	90.5 / 90.6 / **95.8**
	DenseNet121	96.9 / 87.8 / **96.5**	**98.0** / 55.7 / 93.0	83.1 / 48.5 / **82.5**	99.4 / 97.7 / **98.8**	78.3 / 81.2 / **91.9**	91.2 / 74.2 / **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	88.1 / 91.8 / **98.2**	80.2 / 72.6 / **87.8**	62.3 / 62.3 / **78.5**	73.6 / 82.8 / **95.6**	7976 / 83.6 / 93.8	79.9 / 78.6 / **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**	67.7 / 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**

Fooling rate ↑

Extreme Cross-domain Transferability

(neither access to substitute model on target model nor target data)

Source | Target

Source and Target models are trained on different data and also differ in architecture

Target models on CUB200

Gen. Training (data)	Discriminator (ImageNet)	ResNet50	SeNEt154	SeResNet101	Average
		GAP [209] / CDA [188] / 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 [209] / CDA [188] / 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 [209] / CDA [188] / 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**

Fooling rate

(c) Aircraft

Avg. 23% improvement over CDA

Cross-task transferability analysis

(ImageNet classifier ⟶ PASCAL VOC SSD detector)
No access to target data, target model and target task

Source | Target

Source and Target models are trained on different data, task and also differ in architecture

mAP↑

Gen. Training (data)	Discriminator (Trained on ImageNet)	VGG16	ResNet50	EfficientNet	MobileNet-v3	Average
		GAP [10] / CDA [11] / Ours				
-	No Attack	68.12	66.08	61.07	55.44	62.68
Comics (40K)	VGG-16	19.9 / 25.2 / **9.08**	15.7 / 20.9 / **13.7**	**12.4** / 13.4 / 14.2	**9.22** / 13.1 / 14.5	14.3 / 18.1 / **10.1**
	ResNet152	31.0 / 25.3 / **13.7**	23.0 / 20.2 / **10.5**	23.9 / 17.3 / **12.2**	17.9 / 14.0 / **8.37**	24.0 / 19.2 / **11.2**
	Inception-v3	33.2 / 33.8 / **23.7**	27.9 / 27.7 / **25.1**	31.1 / 30.7 / **22.0**	20.2 / 18.7 / **18.4**	28.8 / 27.7 / **22.3**
	DenseNet121	22.1 / 26.3 / **12.2**	18.6 / 21.6 / **14.5**	17.8 / 20.1 / **16.2**	13.9 / 15.3 / **9.55**	18.1 / 20.8 / **13.1**
	SqueezeNet	29.4 / 32.6 / **18.1**	24.8 / 28.9 / **15.7**	20.5 / 24.4 / **17.7**	15.7 / 20.5 / **11.9**	22.6 / 26.6 / **15.9**
	Average	27.1 / 28.7 / **15.4**	22.0 / 23.8 / **15.9**	21.1 / 21.2 / **16.5**	15.4 / 16.3 / **11.7**	21.4 / 22.5 / **14.9**
Paintings (80K)	VGG-16	20.2 / 20.4 / **9.83**	21.4 / 22.5 / **13.2**	14.7 / 15.0 / **12.8**	**11.4** / 12.5 / 12.8	16.9 / 17.6 / **12.2**
	ResNet152	36.6 / 29.4 / **12.8**	26.7 / 21.9 / **12.5**	22.9 / 16.8 / **11.8**	21.3 / 17.6 / **9.40**	26.9 / 21.4 / **11.6**
	Inception-v3	32.3 / 33.5 / **16.8**	29.2 / 29.0 / **18.7**	28.1 / 28.5 / **14.3**	23.4 / 22.6 / **13.2**	28.3 / 28.4 / **15.7**
	DenseNet121	31.7 / 33.2 / **9.27**	23.1 / 23.2 / **11.0**	23.5 / 24.1 / **10.6**	20.2 / 20.9 / **6.53**	24.6 / 25.3 / **9.35**
	SqueezeNet	35.3 / 35.9 / **17.0**	28.5 / 29.0 / **13.7**	26.7 / 27.5 / **17.1**	21.0 / 21.1 / **8.77**	27.9 / 28.3 / **14.1**
	Average	31.3 / 30.5 / **13.1**	25.8 / 25.1 / **13.8**	23.2 / 22.4 / **13.3**	19.5 / 18.9 / **10.1**	25.0 / 24.2 / **12.6**
ImageNet (1.2M)	VGG-16	17.8 / 15.5 / **8.27**	19.2 / 13.9 / **11.8**	9.64 / **8.91** / 11.1	8.39 / **5.79** / 9.78	13.7 / 11.0 / **10.2**
	ResNet152	19.0 / 16.6 / **9.23**	13.5 / 14.6 / **7.67**	12.5 / 11.7 / **6.56**	12.4 / 7.67 / **4.29**	14.3 / 12.6 / **6.94**
	Inception-v3	**13.0** / 22.1 / 15.5	15.7 / 19.4 / **18.2**	13.8 / **12.5** / 13.5	11.3 / 15.1 / **11.6**	15.6 / 17.3 / **14.7**
	DenseNet121	21.5 / 16.1 / **7.60**	15.7 / 13.7 / **8.32**	13.8 / 11.4 / **7.73**	11.3 / 7.10 / **4.42**	15.6 / 12.1 / **7.02**
	SqueezeNet1	27.7 / 26.6 / **13.5**	23.7 / 22.5 / **10.8**	18.6 / 23.4 / **11.8**	15.2 / 17.2 / **7.40**	21.3 / 22.5 / **10.9**
	Average	19.8 / 19.3 / **10.8**	17.5 / 16.8 / **11.4**	13.5 / 13.6 / **10.1**	12.1 / 10.6 / **7.50**	15.7 / 15.1 / **9.95**

Adversarial Attacks

- Attacks beyond image recognition
  - White-box attacks on semantic segmentation
  - Black-box transfer attacks on visual object tracker

- Why adversarial attacks transfer?
  - Learning transferable transferable perturbations

- How can we use adversarial attack to improve DNNs
  - **Semantic adversarial attacks to study disentanglement**

Goal: Semantic attacks to study disentanglement of pose and appearance

What is disentanglement?

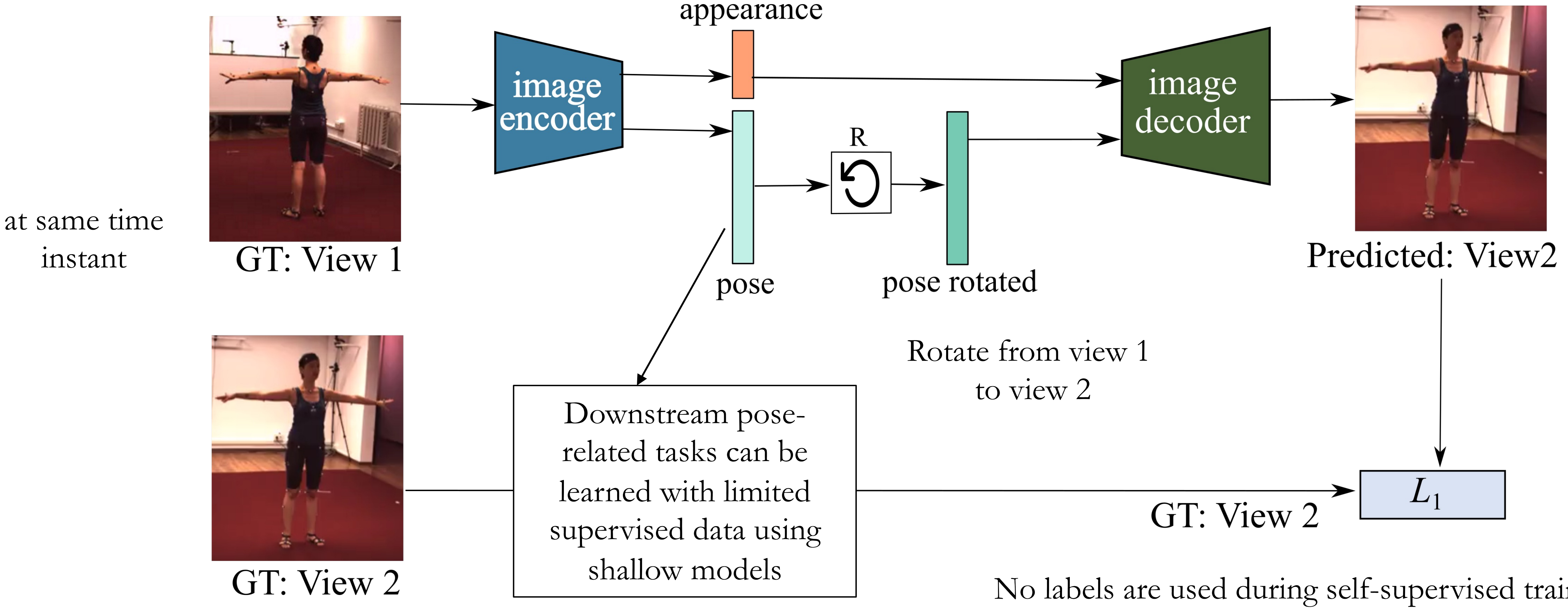
Disentangled representations capture independent factors of variations in data

Why do we need it?

Disentangled representations improves the performance of downstream tasks with limited supervision

Background: Self-supervised Disentangled Representations
(one technique using multi-view information)

Approach: Take one view as input and reconstruct the other view as output



No labels are used during self-supervised training

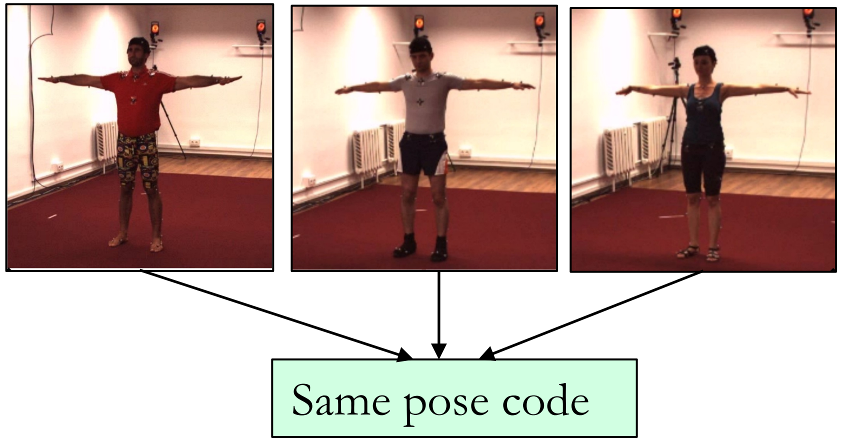
NSD [Helge et al. CVPR 2019]

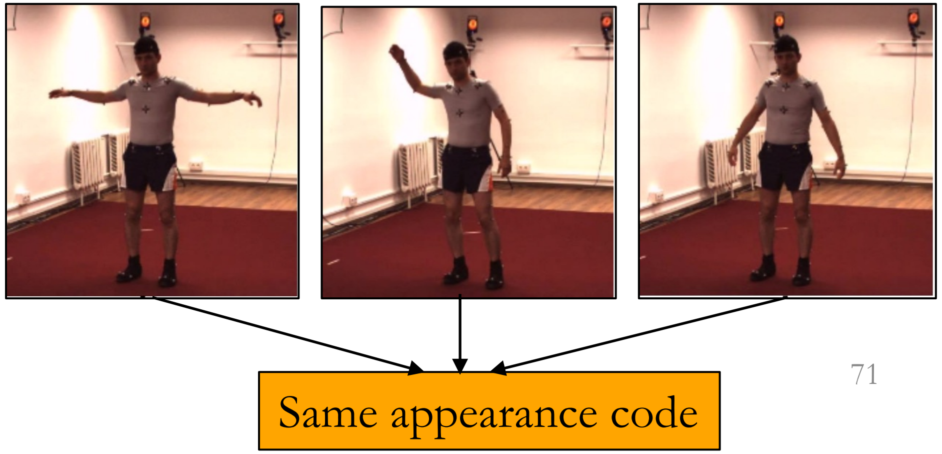
Analyze the disentanglement of pose and appearance

Hypothesis

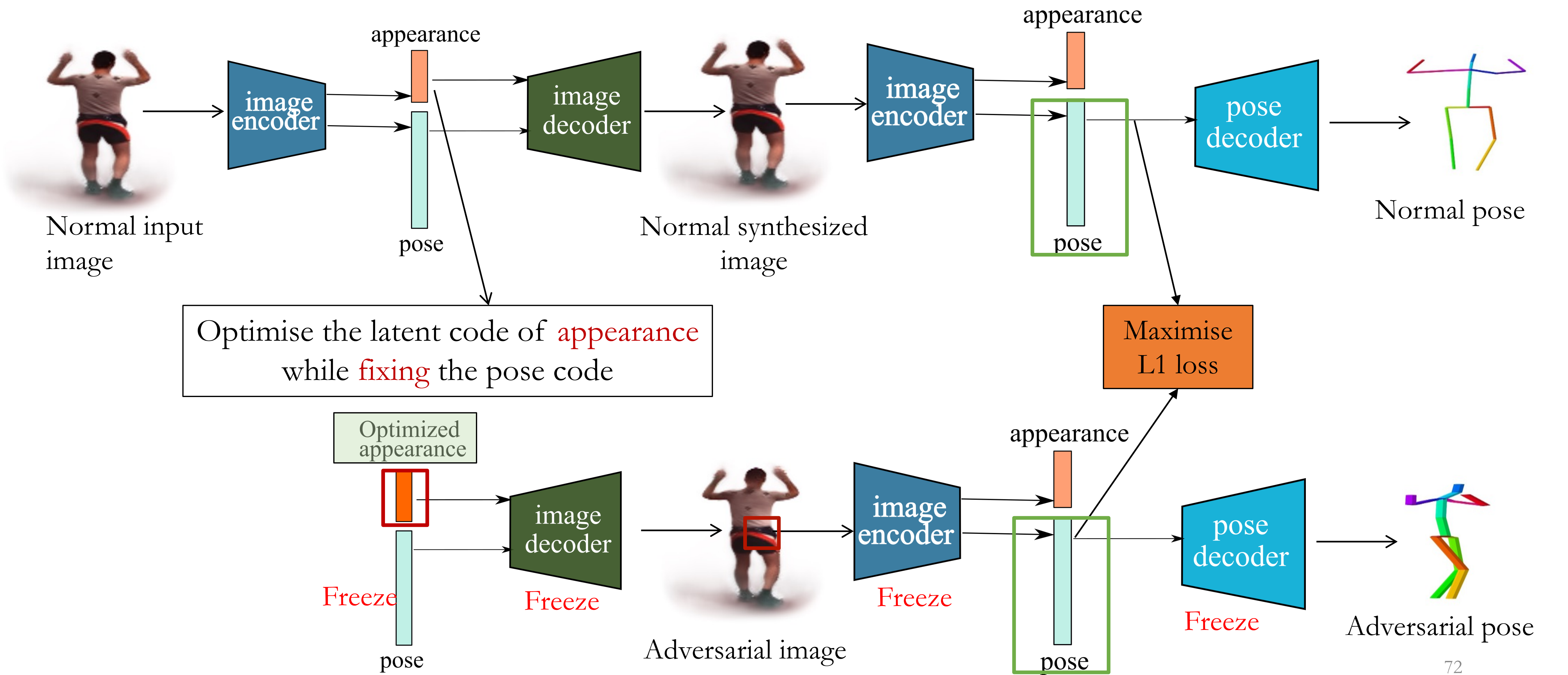
Given two disentangled latent codes that capture two underlying factors of variation in the input data, the adversarial modification of one factor in the input image should not alter the latent code encoded by another factor

Example of pose-appearance disentanglement

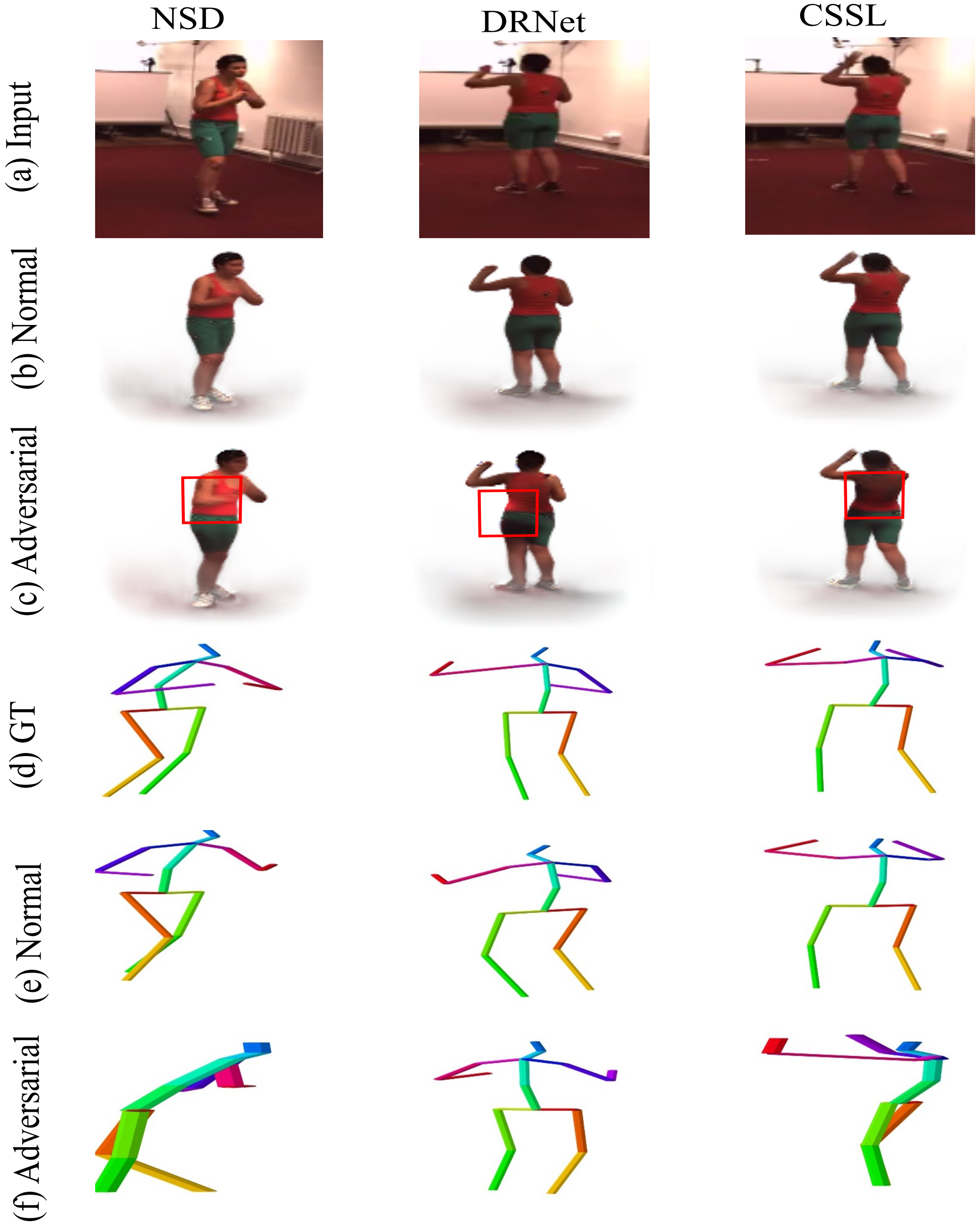




Semantic appearance attacks to understand the disentanglement of pose and appearance



Qualitative results to show the disentanglement is incomplete



- A testbed to evaluate the disentanglement of pose and appearance
- Potential connection between disentanglement and robustness

Conclusion

- Adversarial attacks have signficant implication in the world of self-driving cars.
  - Results of indirect attack to fool far away dynamic objects are unsettling

- Black-box attacks are more realistic threat setting than white-box setting
  - Transferable perturbations in cross-model, cross-domain and cross-task setting

- Interpretable models to reveal working mechanism of adversarial attacks and to improve robustness
  - BoW networks for adversarial attack detection
  - Attention-based BoW networks with metric learning for defending to attacks

Questions?

Interpretability

KKN & MS, BMVC18

Attacks

KKN & MS, ECCV20
KKN & MS, NeurIPS21
KKN & MS, arXiv21
KKN & MS, arXiv22

Defenses

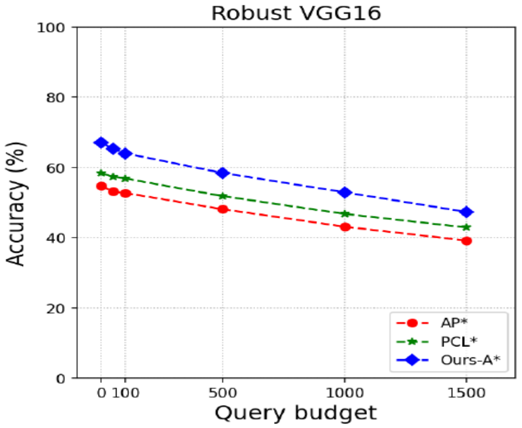
KKN & MS, ACCV20

Detection

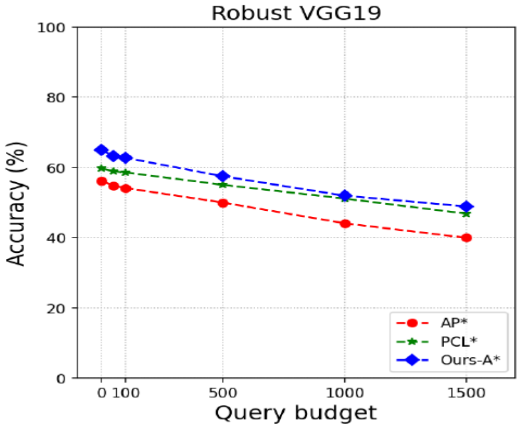
KKN & MS, ICCVW19
KL, KKN, PF & MS, ICCV19

Black-box Square attacks on adversarial trained models

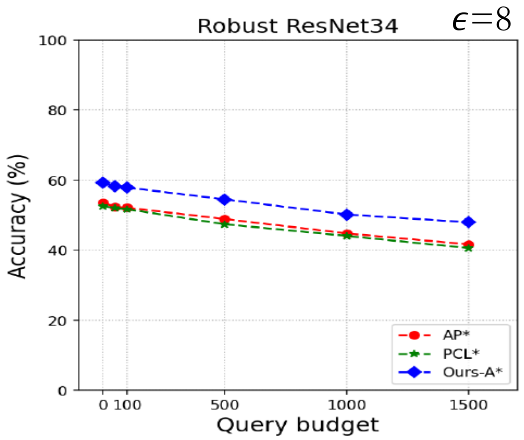
Higher is better



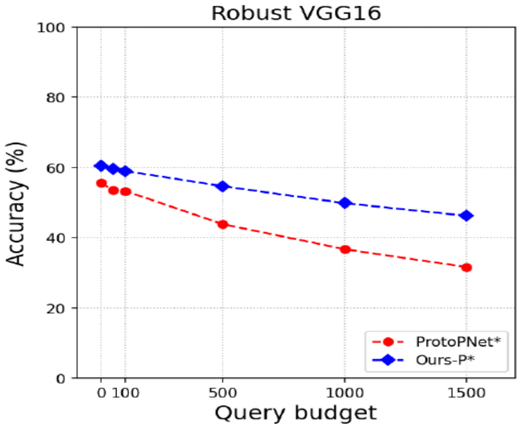
(a) VGG16



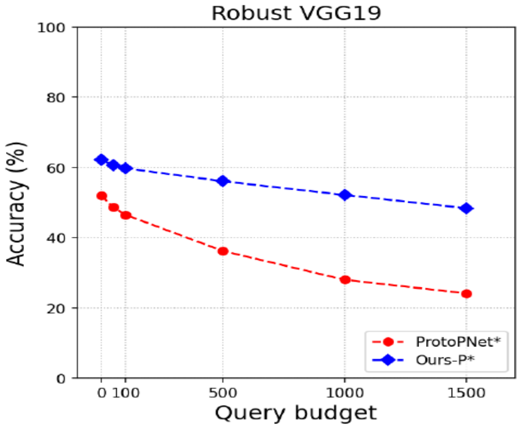
(b) VGG19



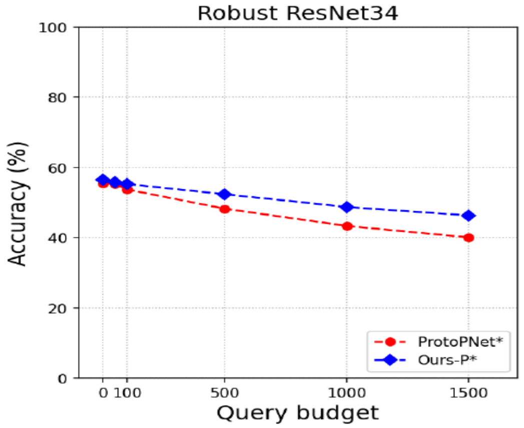
(c) ResNet34



(d) VGG16



(e) VGG19

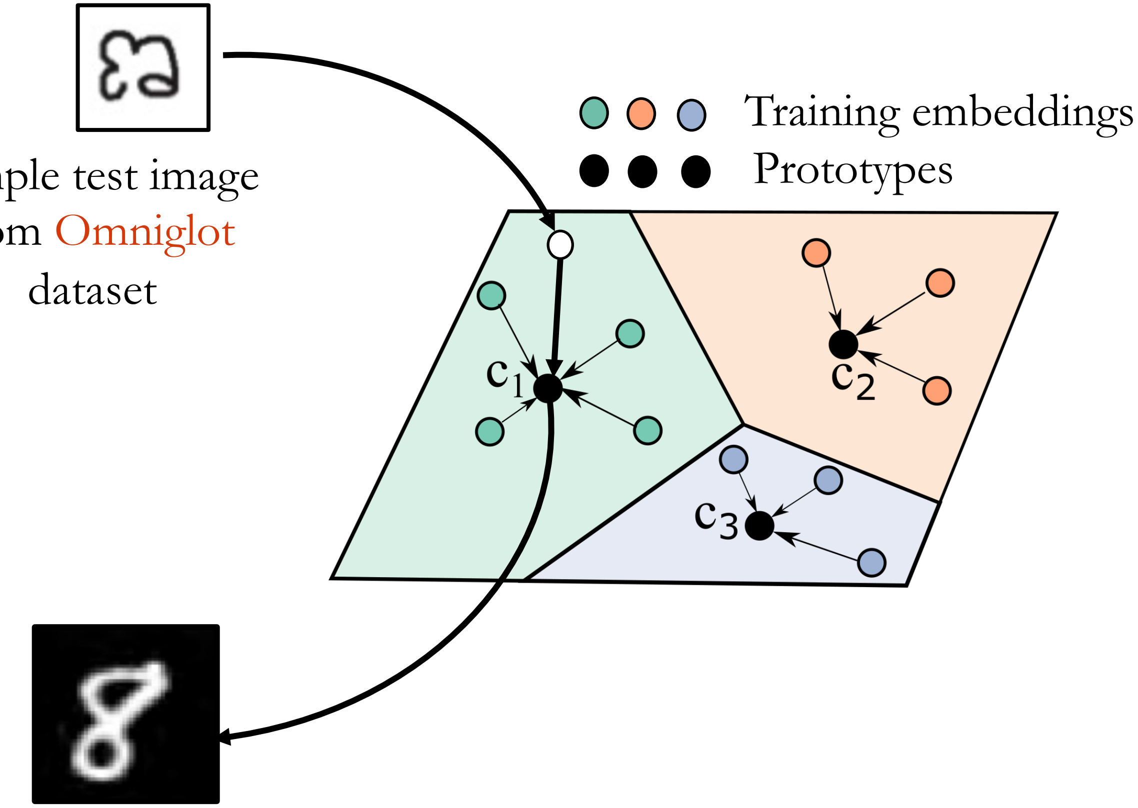


(f) ResNet34

Y-axis is **accuracy**, and x-axis is **query budget** for Square attack

Similar intuition for OOD detection:

Out-of-distribution (OOD) input activates a different looking prototype



Sample test image from Omniglot dataset

Visual representation of closest prototype trained on In-distribution dataset

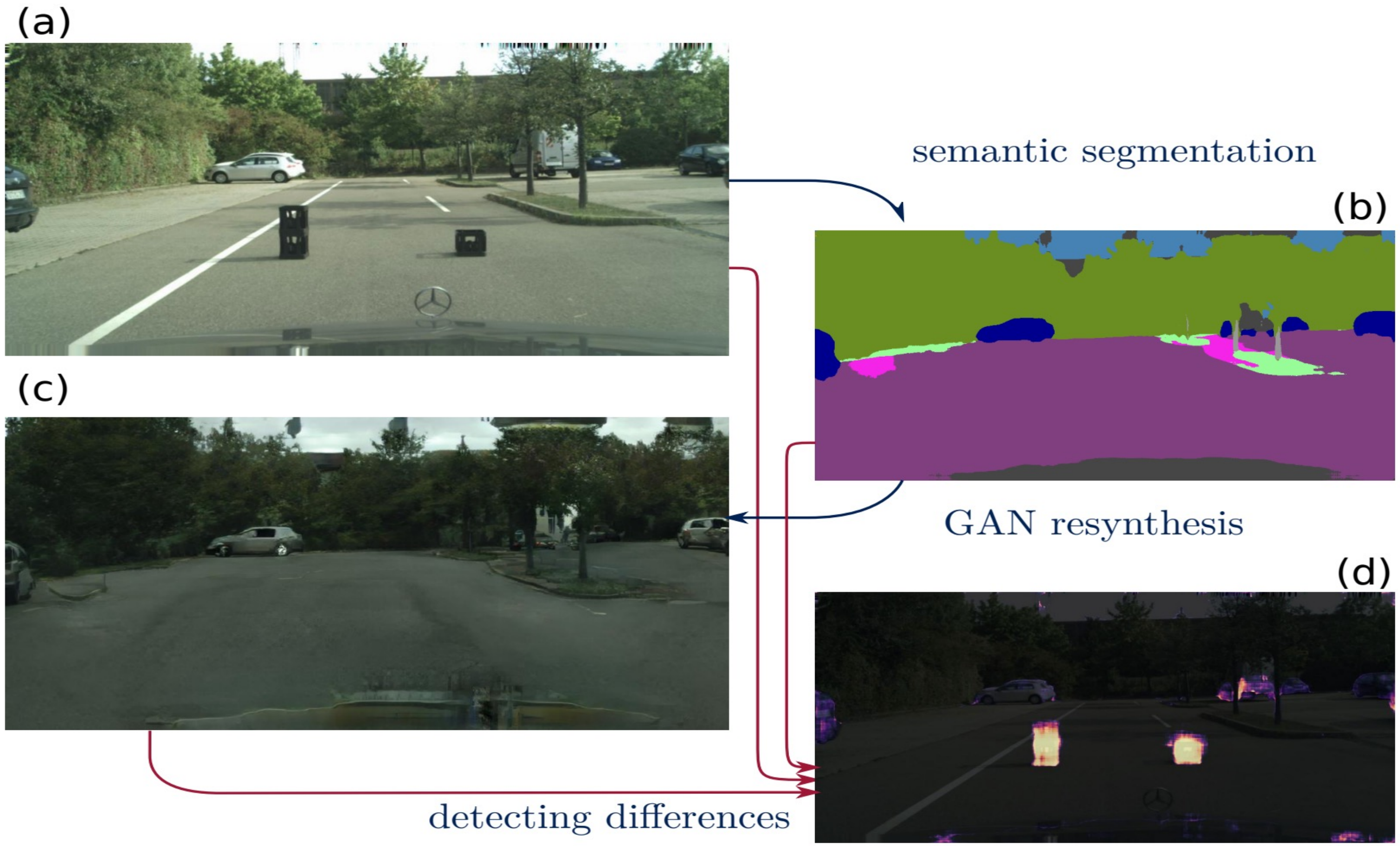
D: Siamese network to predict if the input pair is similar or dissimilar trained on In-distribution dataset

Dissimilar

OOD sample

Road Anomaly detection

Pixel-level detection of anomalous objects by comparing input image to the image resynthesized from output map



Pretrain the discrepancy detector network on real and synthesised images by randomly replacing objects of few classes with other classes

[KL, KKN, PF & MS, ICCV19]

Road Anomaly detection

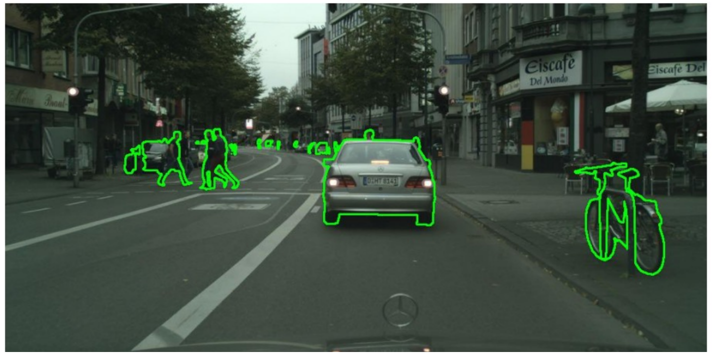
Pixel-level detection of anomalous objects by comparing input image to the image resynthesized from output map

Real predictions

Randomly alter labels of few instances

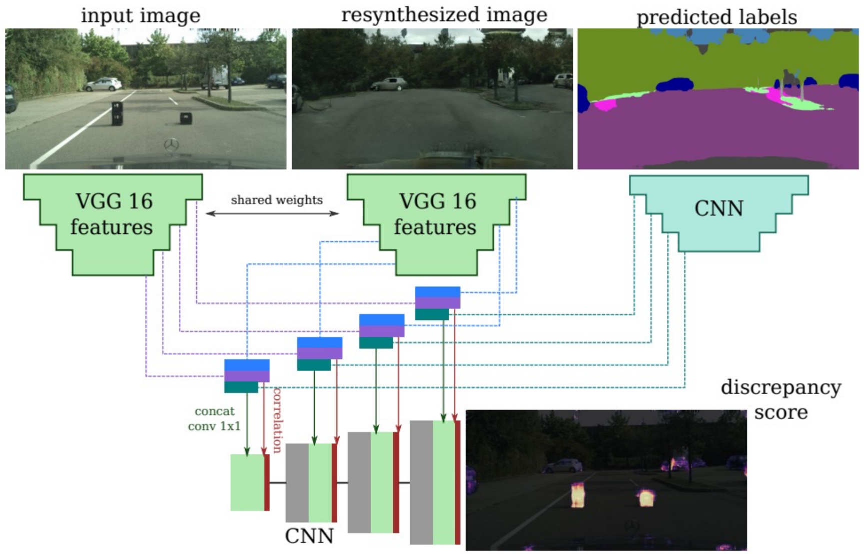

(a)


(b)


Outlines of altered objects


Resynthesized image

Discrepancy network

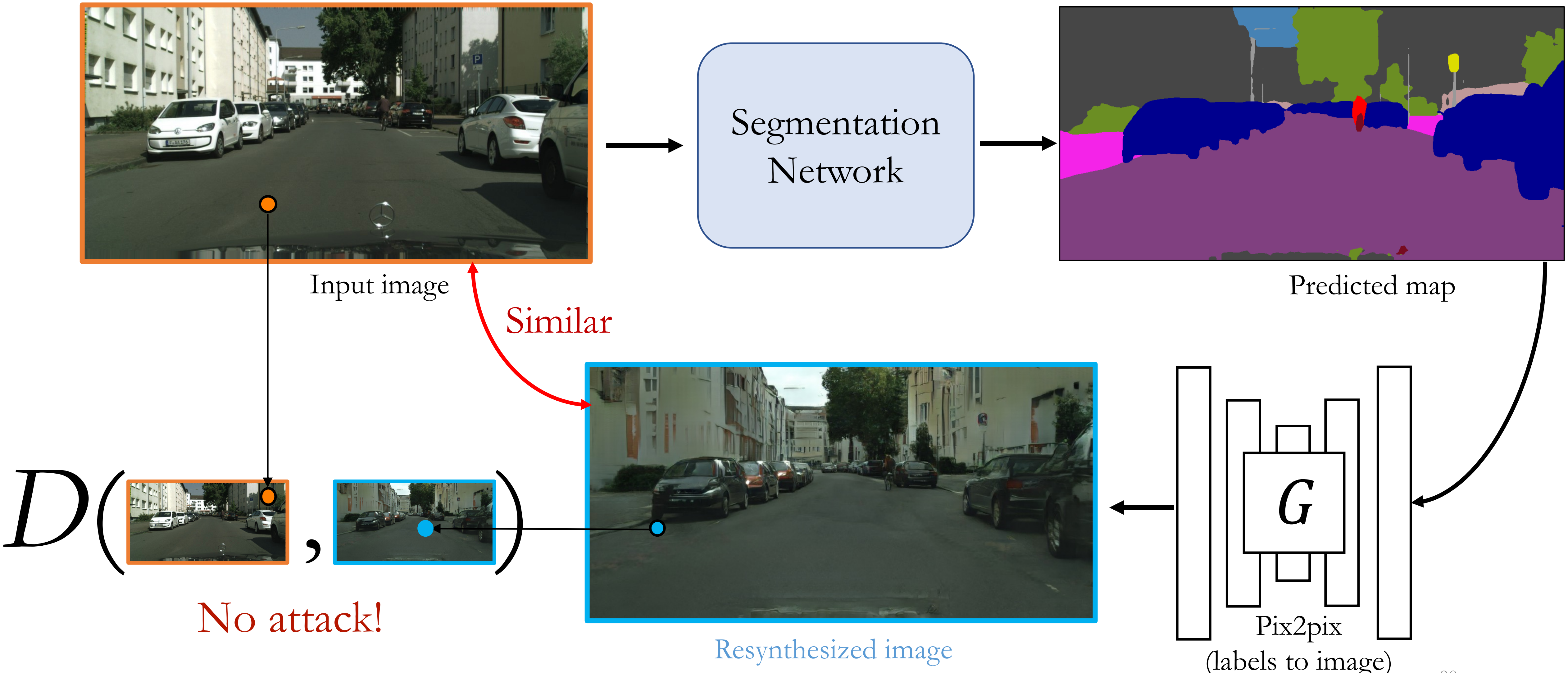


Pretrain the discrepancy detector network on real and synthesised images by randomly replacing objects of few classes with other classes

[KL, KKN, PF & MS, ICCV19]

Adversary detection beyond image-recognition

Adversarial example detection in semantic segmentation by comparing input image to the image resynthesized from output map

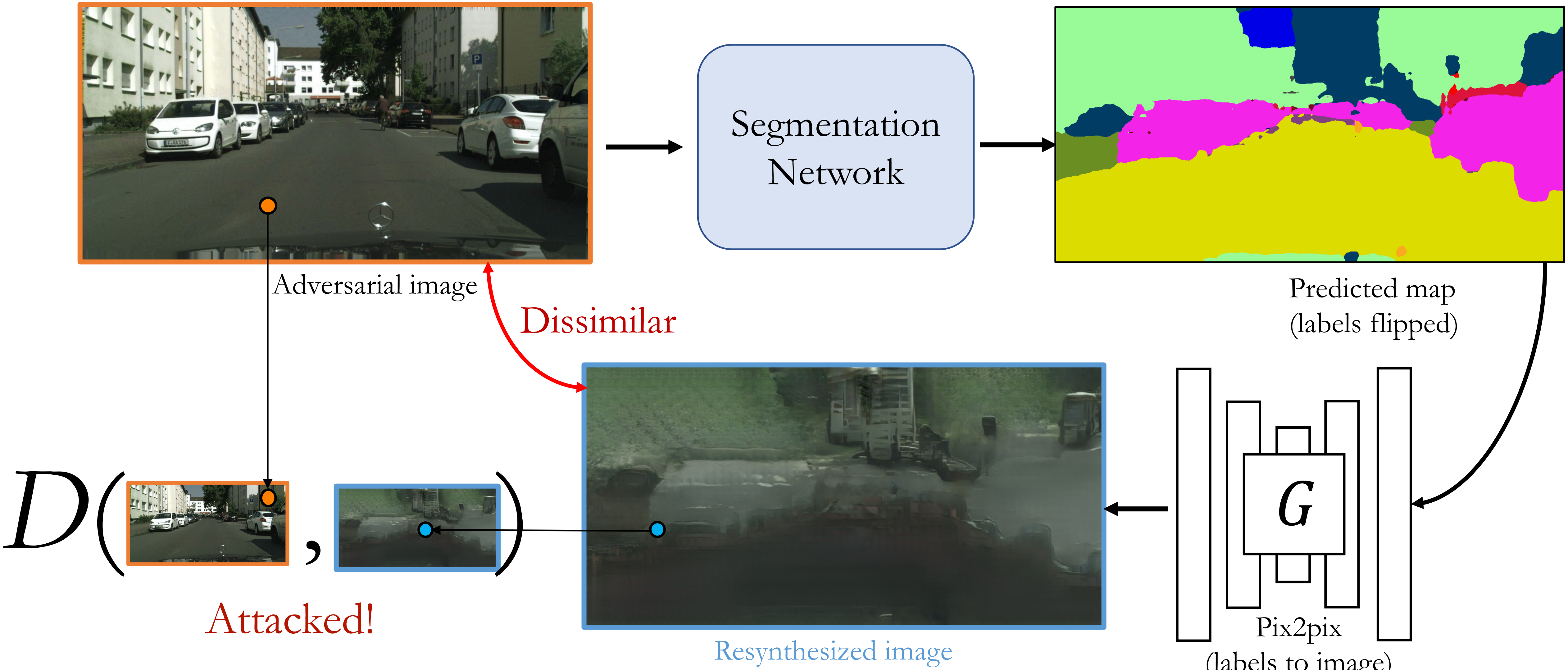


D: Computes L_1 distance in HOG feature space

[KL, KKN, PF & MS, ICCV19]

Adversary detection beyond image-recognition

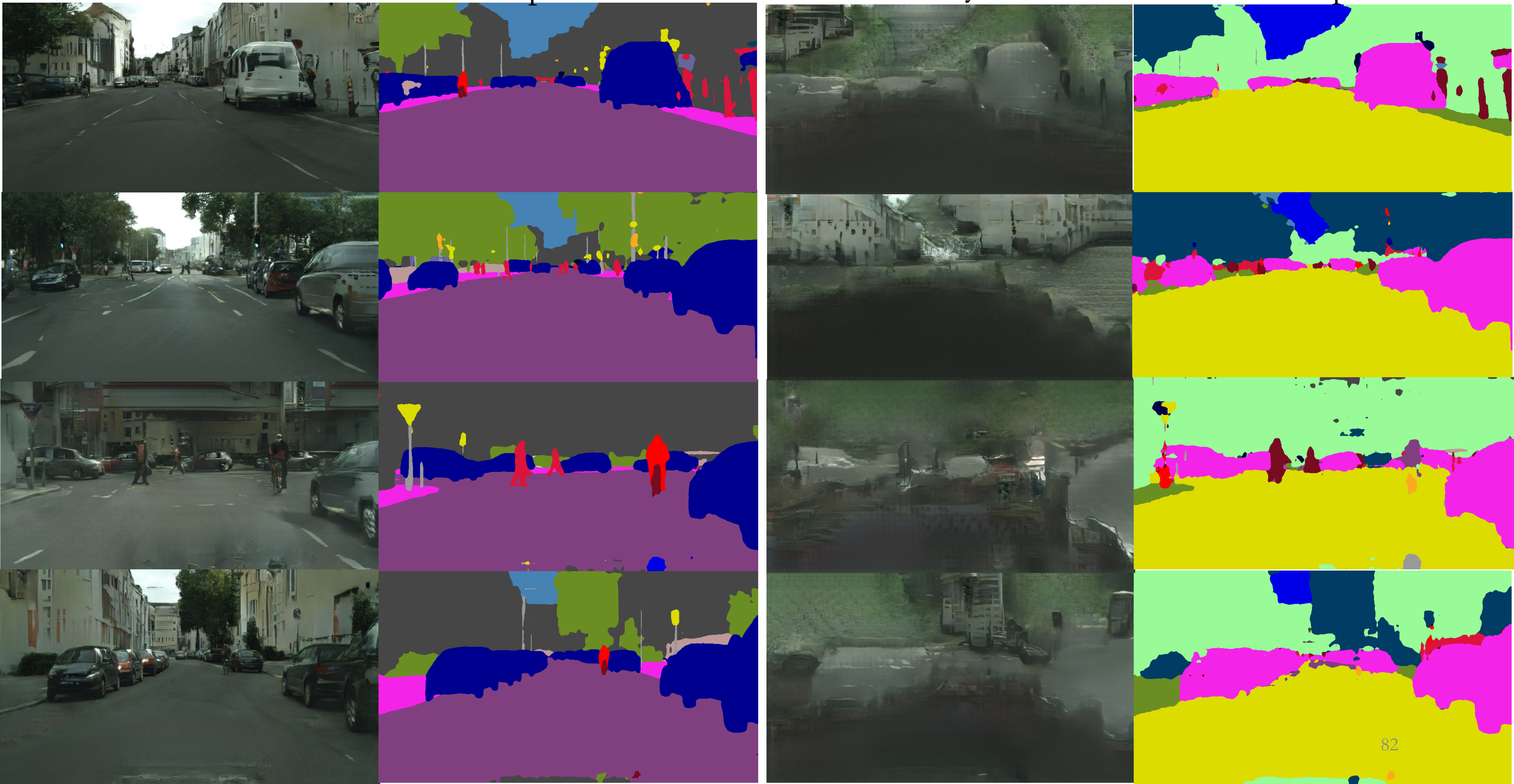
Adversarial example detection in semantic segmentation by comparing input image to the image resynthesized from output map



D: Computes L_1 distance in HOG feature space

[KL, KKN, PF & MS, ICCV19]

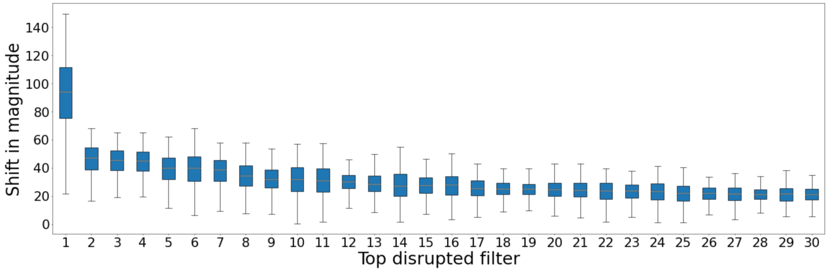
Normal synthesized | Normal predictions | Adversarial synthesized | Adversarial predictions



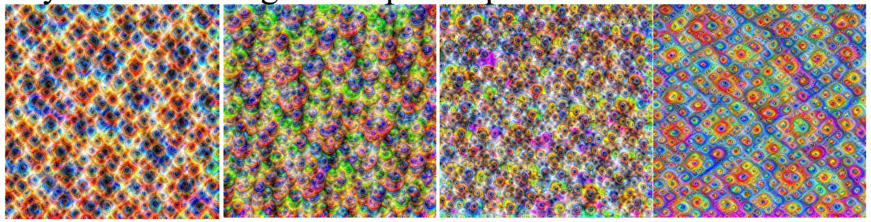
Visual correlation b/w adversarial images & top disrupted filters

(a) Generator trained against VGG16

Boxplot of top 30 disrupted filters in layer 18 of VGG16

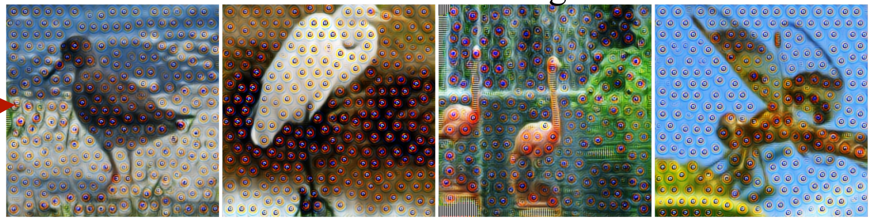


Synthesized images of top disrupted filters in VGG16



Visually correlated

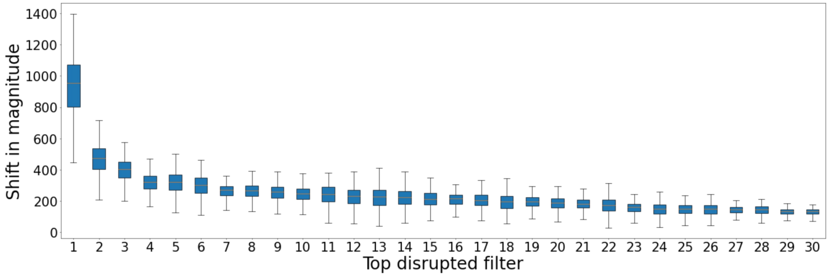
Unbounded adversarial images with VGG16



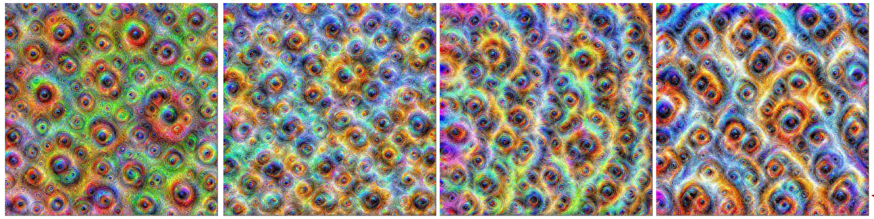
Final projected adversarial images with VGG16

(b) Generator trained against SqueezeNet

Boxplot of top 30 disrupted filters in layer 10 of SqueezeNet

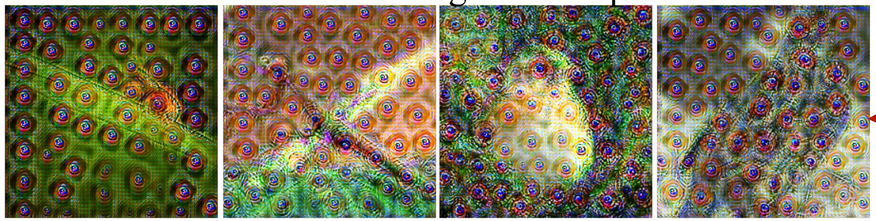


Synthesized images of top disrupted filters in SqueezeNet



Visually correlated

Unbounded adversarial images with SqueezeNet



Final projected adversarial images with SqueezeNet