Paper Seminar

Adversarial Examples Improve Image Recognition

Xie et al., 2020, CVPR

Myeongsup Kim

Integrated M.S./Ph.D. Student
Data Science & Business Analytics Lab.
School of Industrial Management Engineering
Korea University

Myeongsup_kim@korea.ac.kr

<What Are Not Covered in This Seminar>

- **Details of Batch Normalization**

  Ioffe and Szegedy, Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, ICML, 2015

Introduction

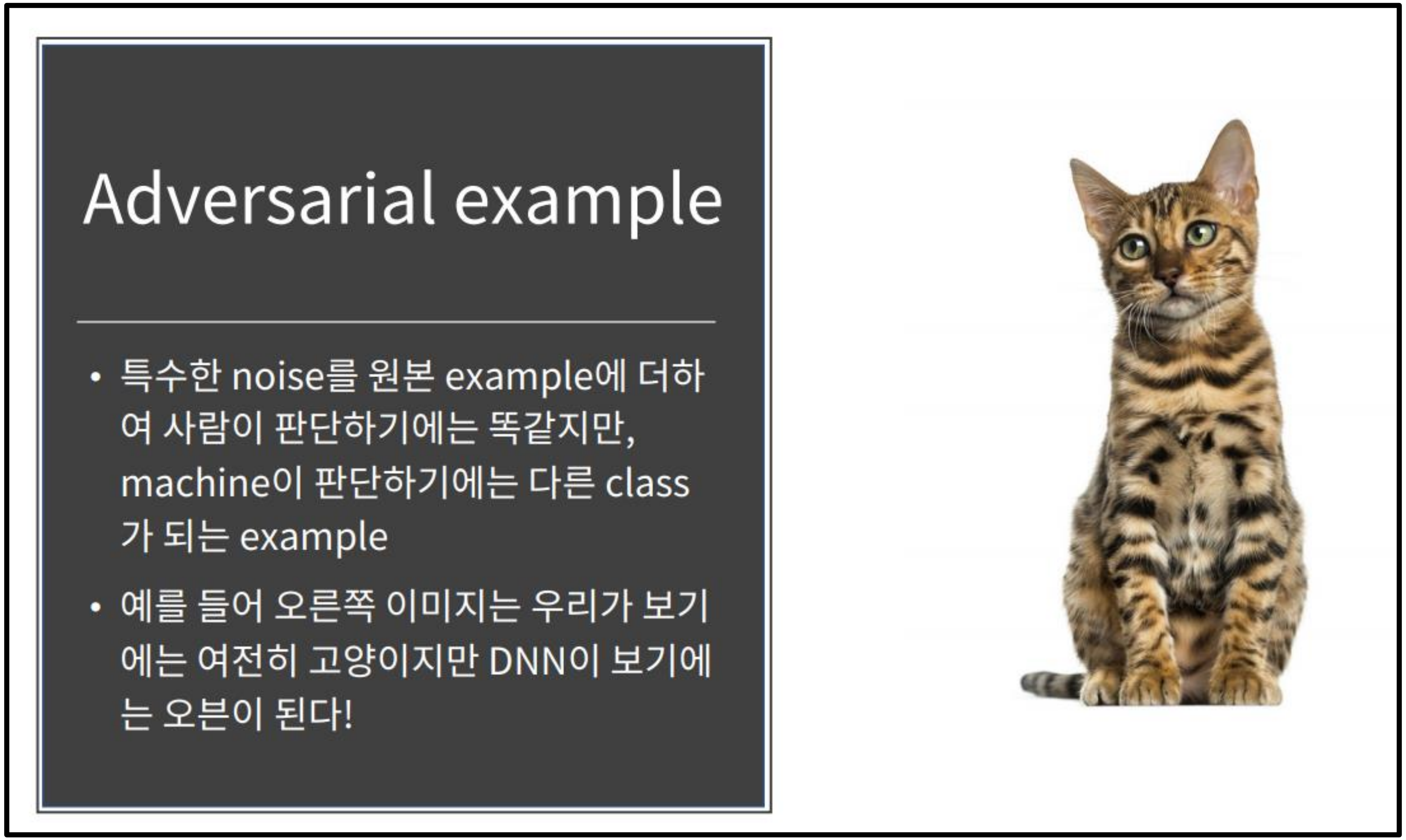
- Adversarial Example
- Adversarial Training
- Degrading Performance

From [Paper Review] Adversarial Defense by Restricting the Hidden Space of Deep Neural Network (Seungwan Seo, 2020)

Introduction

- Adversarial Example

<Recap: Previous Seminar>



Introduction

- Adversarial Example

<Supervised Learning>

<Input Image>

Model

<Training>

[Cat]

[Dog]

<Classification>

Introduction

- Adversarial Example

<Supervised Learning>

<Input Image>

Model

<Training>

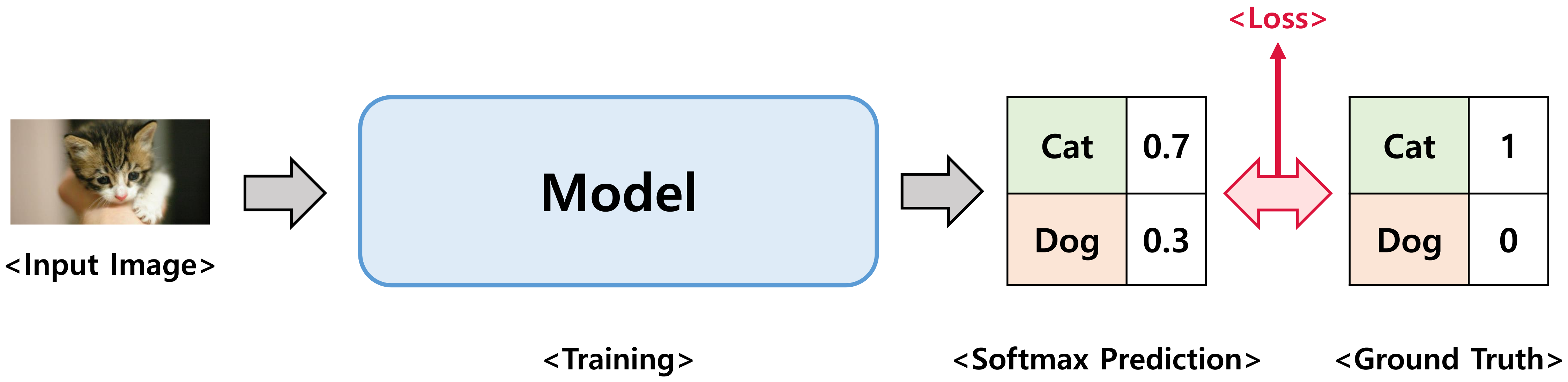
Cat	0.7
Dog	0.3

<Softmax Prediction>

Introduction

- Adversarial Example

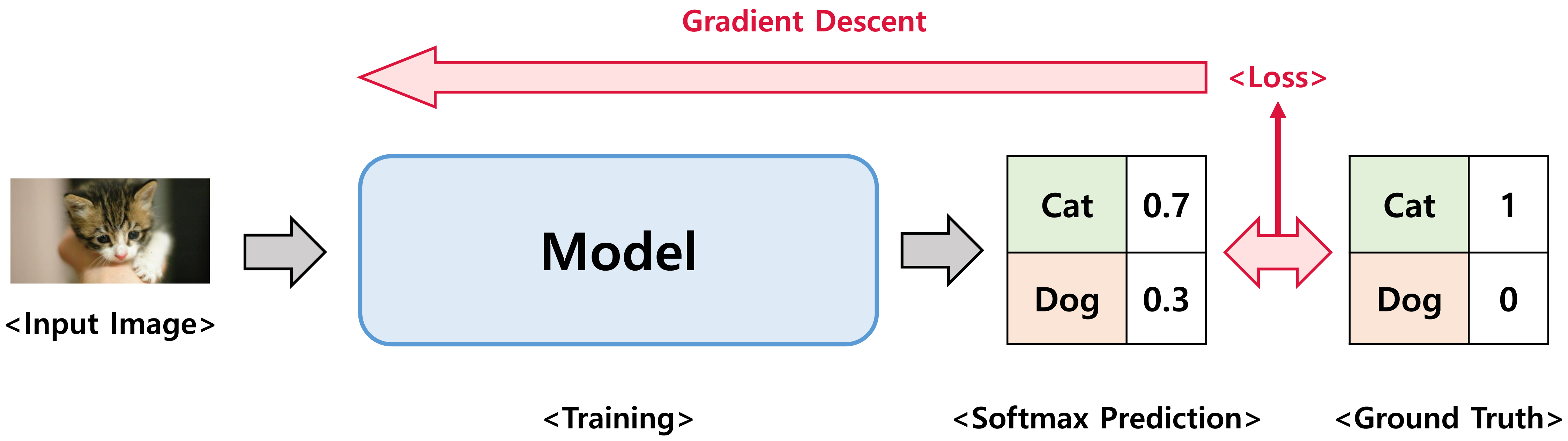
<Supervised Learning>



Introduction

- Adversarial Example

<Supervised Learning>



Introduction
- Adversarial Example

<Supervised Learning>

<Input Image>

Model

<Training>

Cat	0.9
Dog	0.1

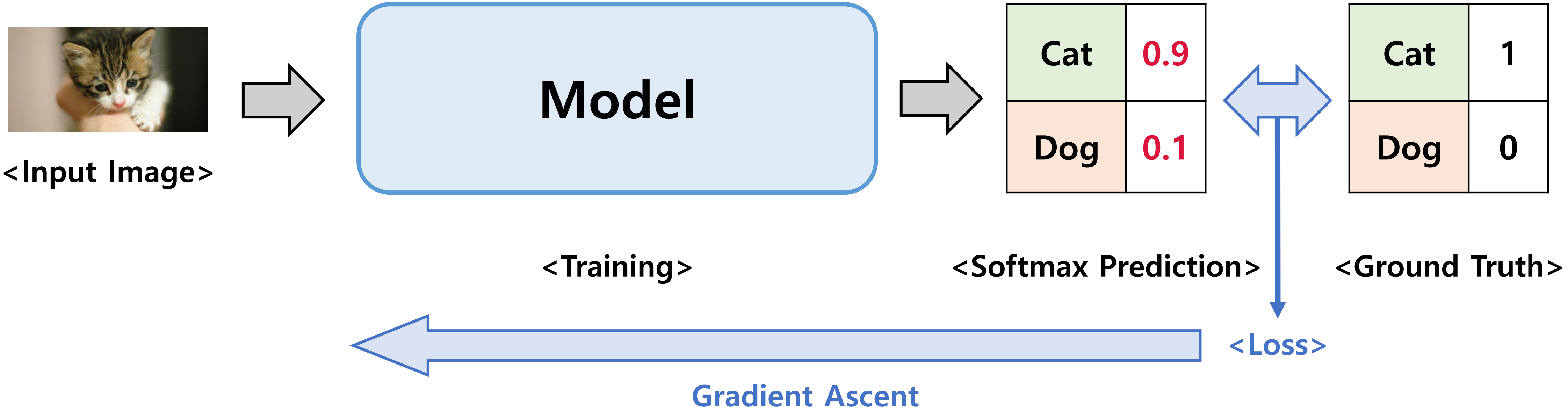
<Softmax Prediction>

Cat	1
Dog	0

<Ground Truth>

Introduction
- Adversarial Example

<Adversarial Attack>



<Adversarial Attack>

<Input Image>

Freeze

<Training>

Cat	0.9
Dog	0.1

<Softmax Prediction>

Cat	1
Dog	0

<Ground Truth>

Introduction

- Adversarial Example

<Adversarial Attack>

<Input Image>

\+

<Perturbation>

Freeze

<Training>

Cat	0.9
Dog	0.1

<Softmax Prediction>

Cat	1
Dog	0

<Ground Truth>

Introduction
- Adversarial Example

<Adversarial Attack>

<Input Image>

\+

<Perturbation>

Freeze

<Training>

Cat	0.9
Dog	0.1

<Softmax Prediction>

Cat	1
Dog	0

<Ground Truth>

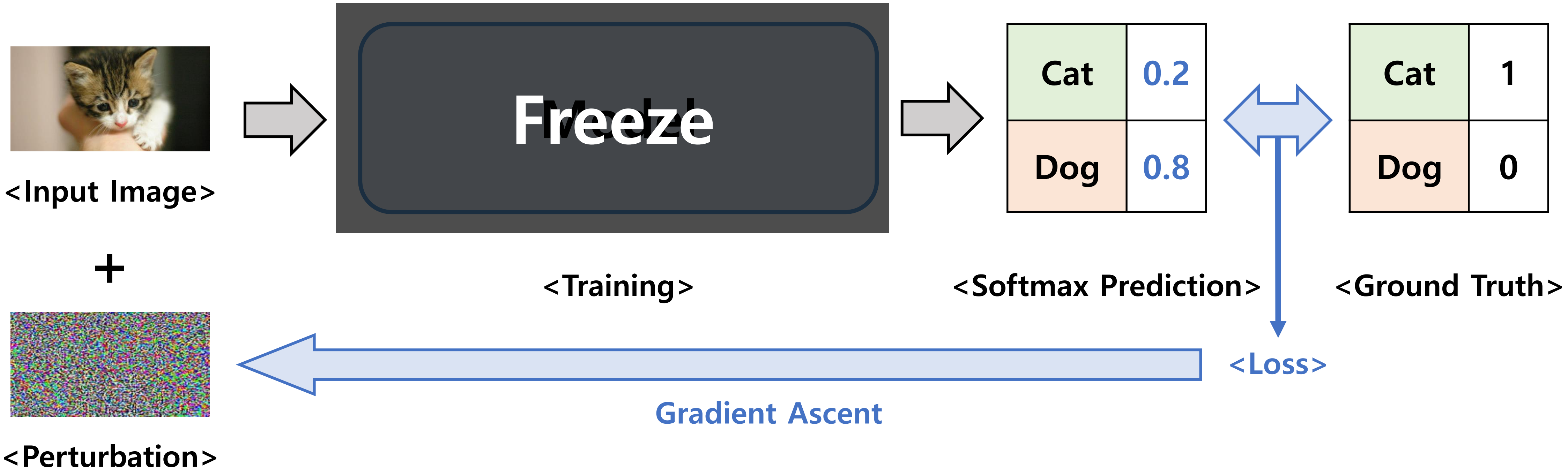
<Loss>

Gradient Ascent

Introduction

- Adversarial Example

<Adversarial Attack>



<Adversarial Attack>

<Input Image>

+

<Perturbation>

Model

<Inference>

Cat	0.2
Dog	0.8

<Softmax Prediction>

Cat	1
Dog	0

<Ground Truth>

Introduction

- Adversarial Example

<Adversarial Attack>

<Adversarial Example>

Model

<Inference>

Cat	0.2
Dog	0.8

<Softmax Prediction>

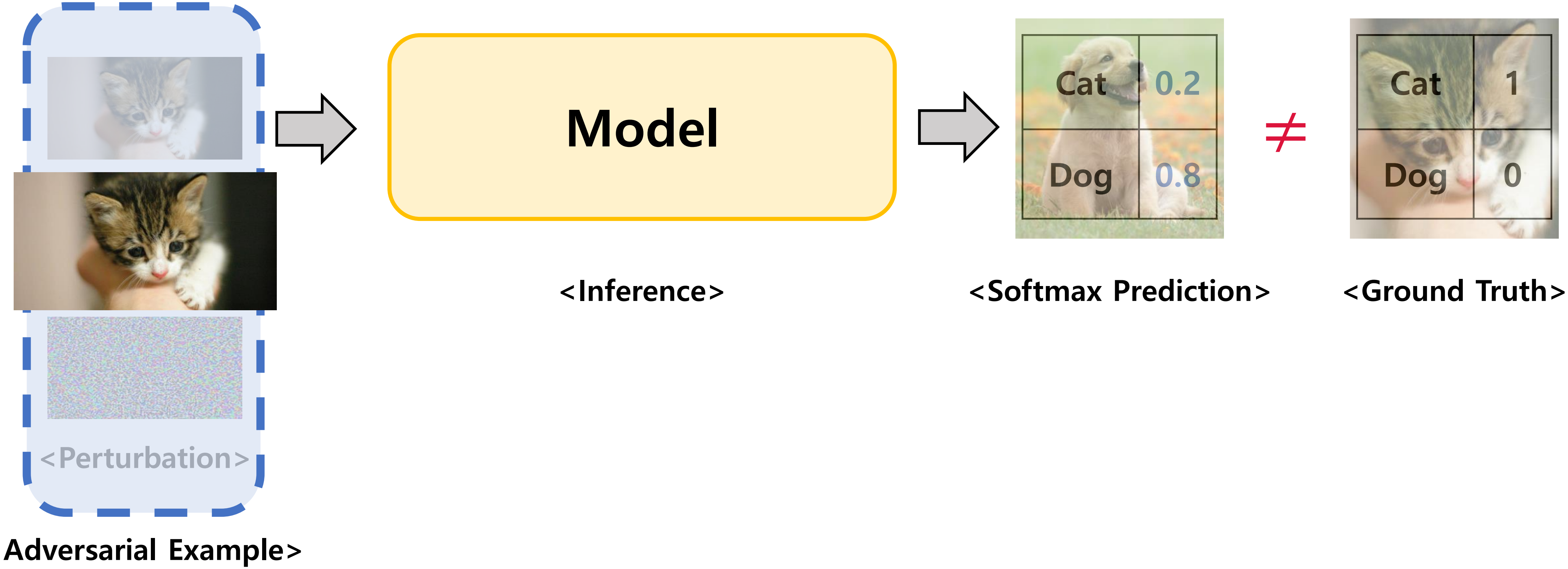
Cat	1
Dog	0

<Ground Truth>

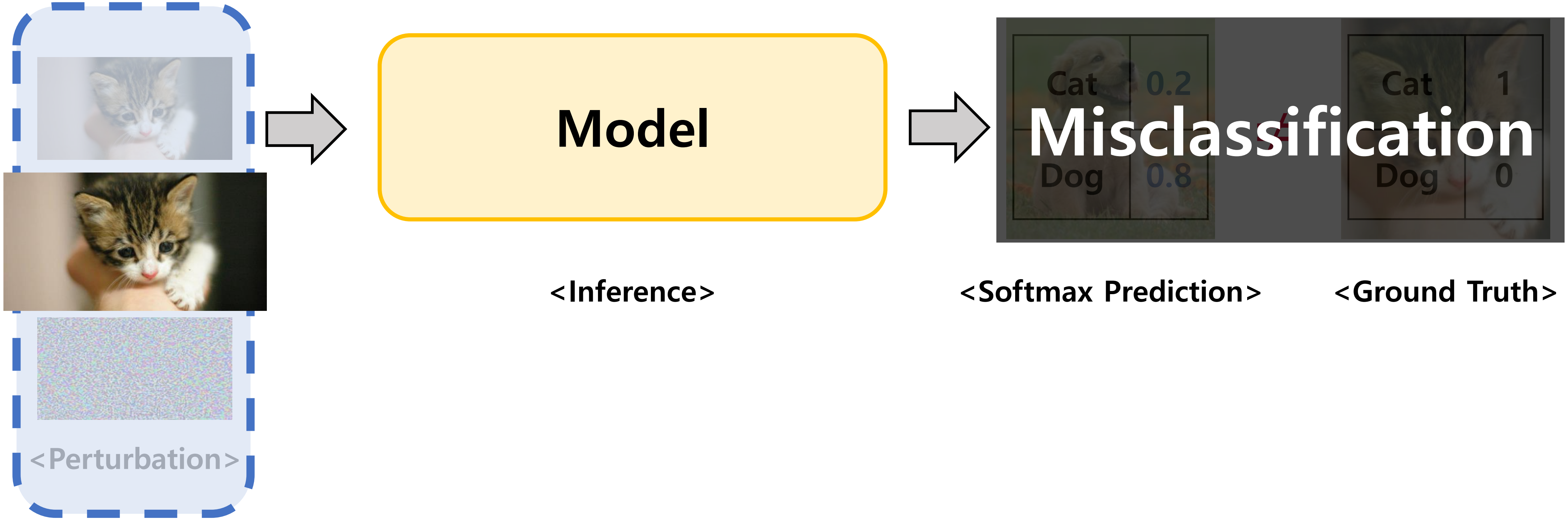
Introduction

- Adversarial Example

<Adversarial Attack>



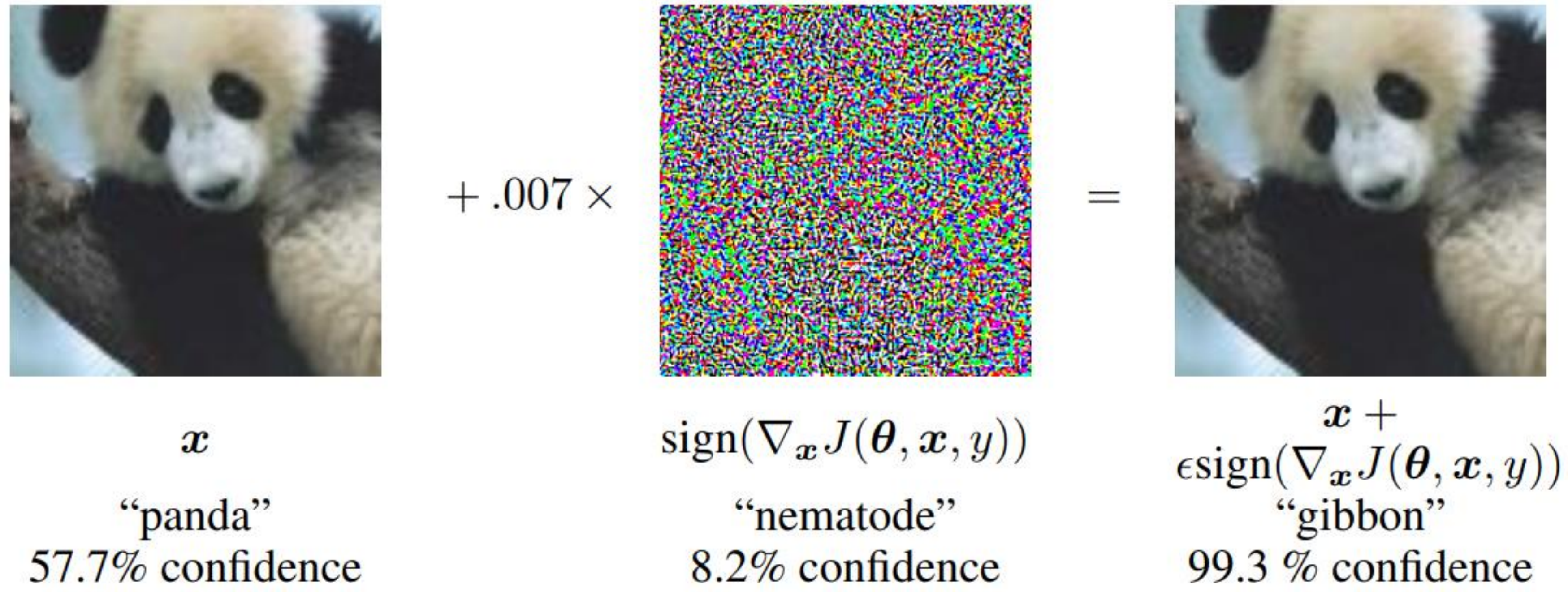
<Adversarial Attack>



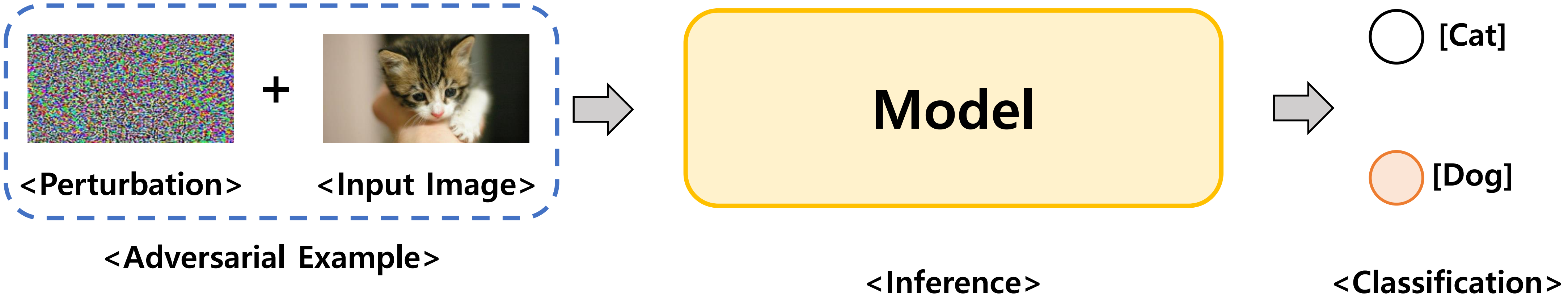
<Adversarial Example>

Goodfellow et al., Explaining and Harnessing Adversarial Examples, ICLR, 2015

<Adversarial Example>



<Adversarial Training>



<Adversarial Training>

<Adversarial Example>

Model

<Inference>

[Cat]

[Dog]

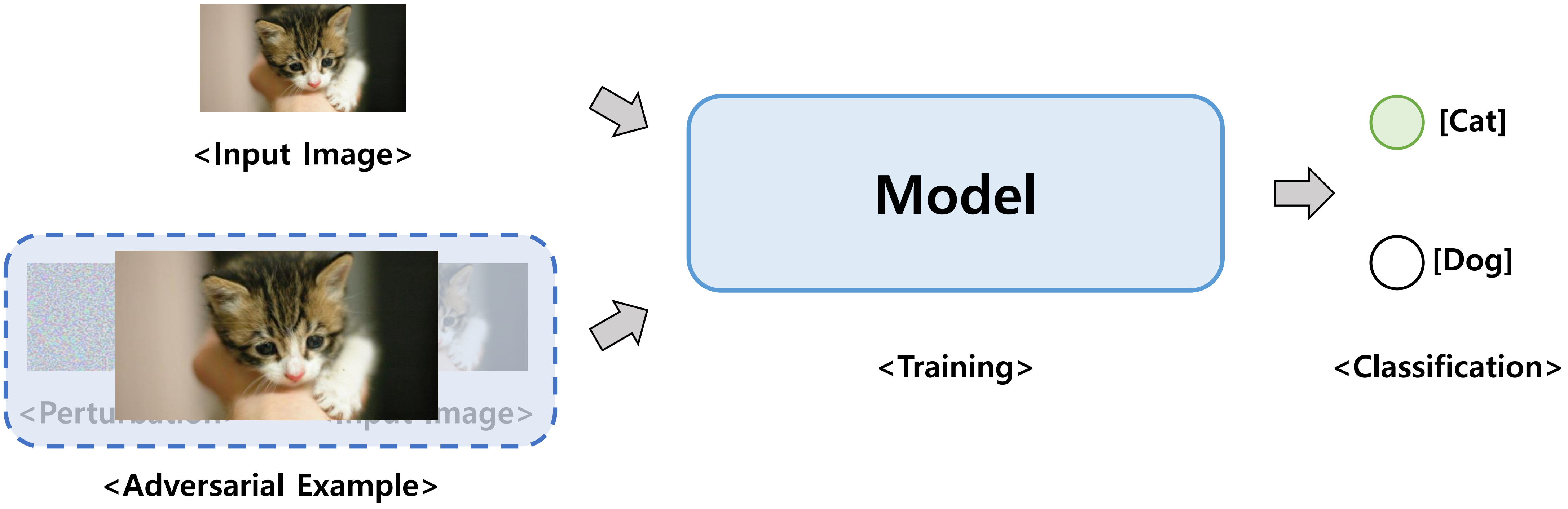
<Classification>

From [Paper Review] Adversarial Defense by Restricting the Hidden Space of Deep Neural Network (Seungwan Seo, 2020)

Introduction

- Adversarial Training

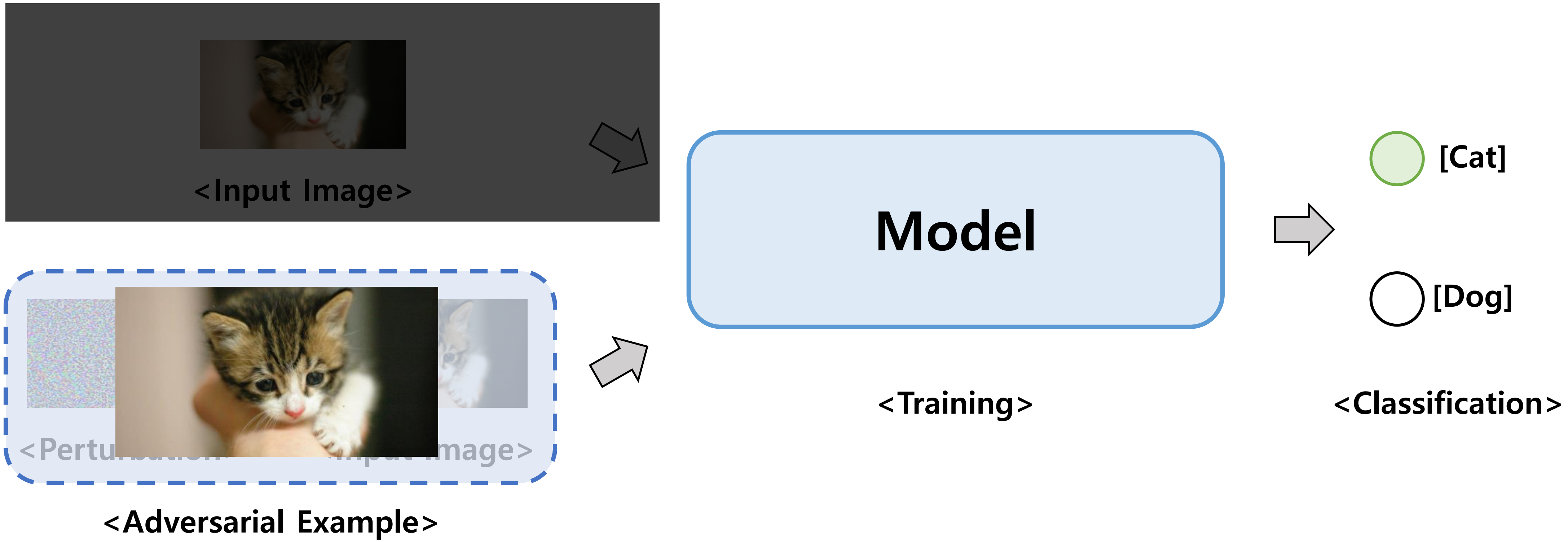
<Adversarial Training>



Introduction

- Adversarial Training

<Adversarial Training>



Xie et al., Feature Denoising for Improving Adversarial Robustness, CVPR, 2019
Kannan et al., Adversarial Logit Pairing, ArXiv, 2018
Kurakin et al., Adversarial Machine Learning at Scale, ICLR, 2017

<Degrading Performance>

- "Finally, we report that our ResNet-152 baseline with *adversarial* training has 62.32% accuracy when tested on *clean* images, whereas its counterpart with "clean" training obtains 78.91%." (Xie et al., 2019)

- "Adversarial training caused a slight (less than 1%) decrease of accuracy on clean examples in our ImageNet experiments." (Kurakin et al., 2017)

- "Similar to the dip in clean accuracy on CIFAR-10 reported by Madry et al. (2017), we found that our models have a slight dip in clean accuracy to 72%." (Kannan et al., 2018)

Xie et al., Feature Denoising for Improving Adversarial Robustness, CVPR, 2019
Kannan et al., Adversarial Logit Pairing, ArXiv, 2018
Kurakin et al., Adversarial Machine Learning at Scale, ICLR, 2017

<Degrading Performance>

- "Finally, we report that our ResNet-152 baseline with *adversarial* training has 62.32% accuracy when tested on *clean* images, whereas its counterpart with "clean" training obtains 78.91%." (Xie et al., 2019)

- "Adversarial training caused a slight (less than 1%) decrease of accuracy on clean examples in our ImageNet experiments." (Kurakin et al., 2017)

- "Similar to the dip in clean accuracy on CIFAR-10 reported by Madry et al. (2017), we found that our models have a slight dip in clean accuracy to 72%." (Kannan et al., 2018)

Kurakin et al., Adversarial Machine Learning at Scale, ICLR, 2017
Madry et al., Towards Deep Learning Models Resistant to Adversarial Attacks, ICLR, 2018

<Degrading Performance>

		Clean	$\epsilon = 2$
Baseline (standard training)	**top 1**	78.4%	30.8%
	top 5	94.0%	60.0%
Adv. training	**top 1**	77.6%	73.5%
	top 5	93.8%	91.7%
Deeper model (standard training)	**top 1**	78.7%	33.5%
	top 5	94.4%	63.3%
Deeper model (Adv. training)	**top 1**	78.1%	75.4%
	top 5	94.1%	92.6%

<Kurakin et al., 2017>

Model \ Adversary	Natural	FGSM	FGSM random
Simple (standard training)	92.7%	27.5%	19.6%
Simple (FGSM training)	87.4%	90.9%	90.4%
Simple (PGD training)	79.4%	51.7%	55.9%
Wide (standard training)	95.2%	32.7%	25.1%
Wide (FGSM training)	90.3%	95.1%	95.0%
Wide (PGD training)	87.3%	56.1%	60.3%

<Madry et al., 2018>

Introduction

- Degrading Performance

<Degrading Performance>

		Clean	$\epsilon = 2$
Baseline (standard training)	top 1	**78.4%**	30.8%
	top 5	**94.0%**	60.0%
Adv. training	top 1	77.6%	73.5%
	top 5	93.8%	91.7%
Deeper model (standard training)	top 1	**78.7%**	33.5%
	top 5	**94.4%**	63.3%
Deeper model (Adv. training)	top 1	78.1%	75.4%
	top 5	94.1%	92.6%

<Kurakin et al., 2017>

Model \ Adversary	Natural	FGSM	FGSM random
Simple (standard training)	**92.7%**	27.5%	19.6%
Simple (FGSM training)	87.4%	90.9%	90.4%
Simple (PGD training)	79.4%	51.7%	55.9%
Wide (standard training)	**95.2%**	32.7%	25.1%
Wide (FGSM training)	90.3%	95.1%	95.0%
Wide (PGD training)	87.3%	56.1%	60.3%

<Madry et al., 2018>

[Kurakin et al., Adversarial Machine Learning at Scale, ICLR, 2017](#)
[Madry et al., Towards Deep Learning Models Resistant to Adversarial Attacks, ICLR, 2018](#)

<Degrading Performance>

		Clean	$\epsilon = 2$
Baseline (standard training)	top 1	78.4%	30.8%
	top 5	94.0%	60.0%
Adv. training	top 1	77.6%	[illegible]
	top 5	93.8%	91.7%
Deeper model (standard training)	top 1	78.7%	33.5%
	top 5	94.4%	65.3%
Deeper model (Adv. training)	top 1	[illegible]	75.4%
	top 5	94.1%	92.6%

<Kurakin et al., 2017>

Model \ Adversary	Natural	FGSM	FGSM random
Simple (standard training)	92.7%	27.5%	19.6%
Simple (FGSM training)	87.4%	90.9%	90.4%
Simple (PGD training)	79.4%	51.7%	55.9%
Wide (standard training)	95.2%	32.7%	25.1%
Wide (FGSM training)	90.3%	95.1%	95.0%
Wide (PGD training)	87.3%	56.1%	60.3%

<Madry et al., 2018>

Training Deep Neural Network with Adversarial Examples May Lead Degraded Generalization Performance on Clean Examples

How Can We Use Adversarial Examples as Additional Examples to Improve Model Performance?

Adversarial Examples Improve Image Recognition

Xie et al., 2020, CVPR

Adversarial Examples Improve Image Recognition

Xie et al., 2020, CVPR

Adversarial Examples Improve Image Recognition

Xie et al., 2020, CVPR

“Our work is the first to show adversarial examples can improve model performance in fully-supervised setting on the large-scale ImageNet dataset.”

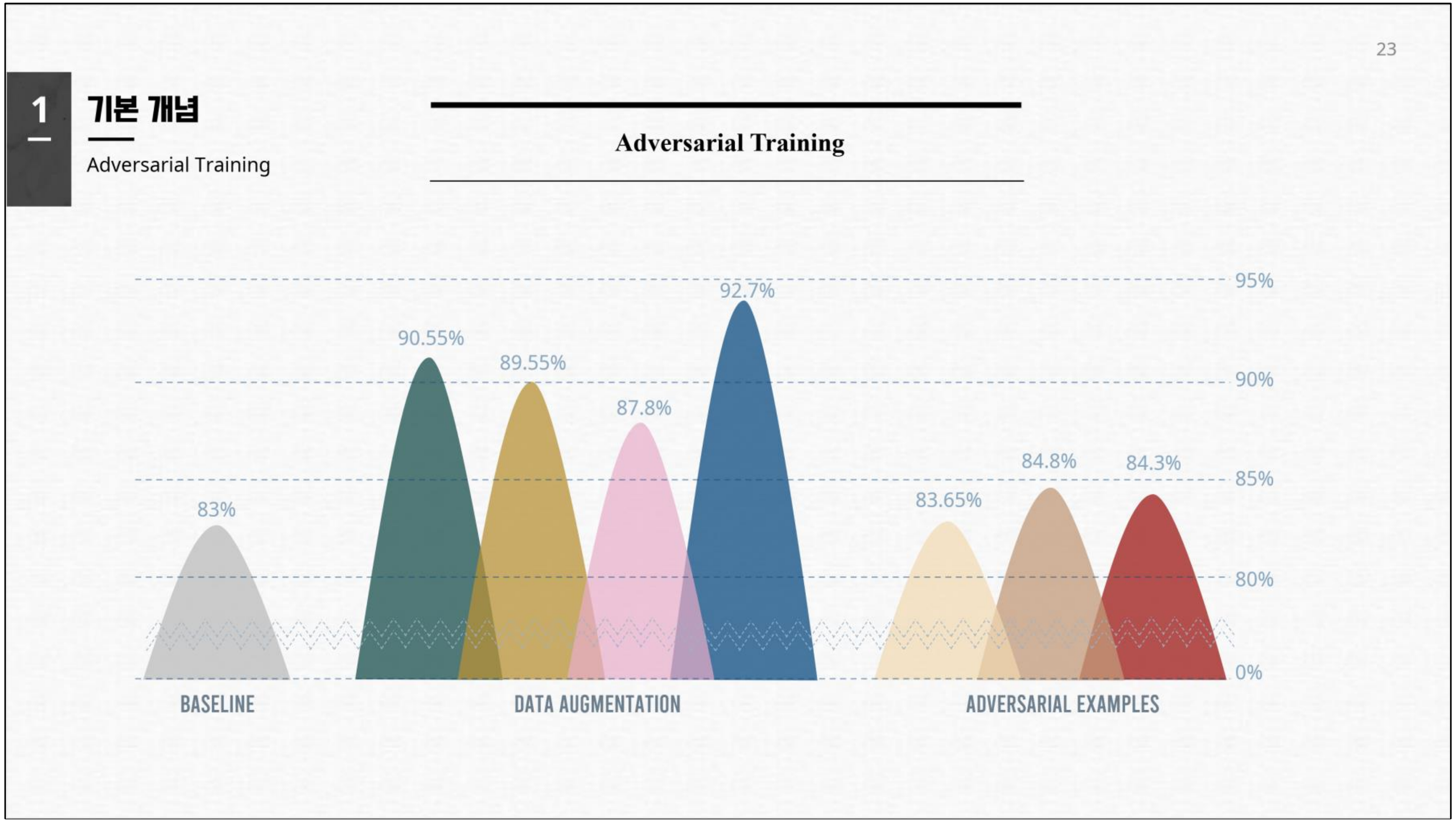
Preliminary

- Preliminary Way to Boost Performance

Preliminary

- Preliminary Way to Boost Performance

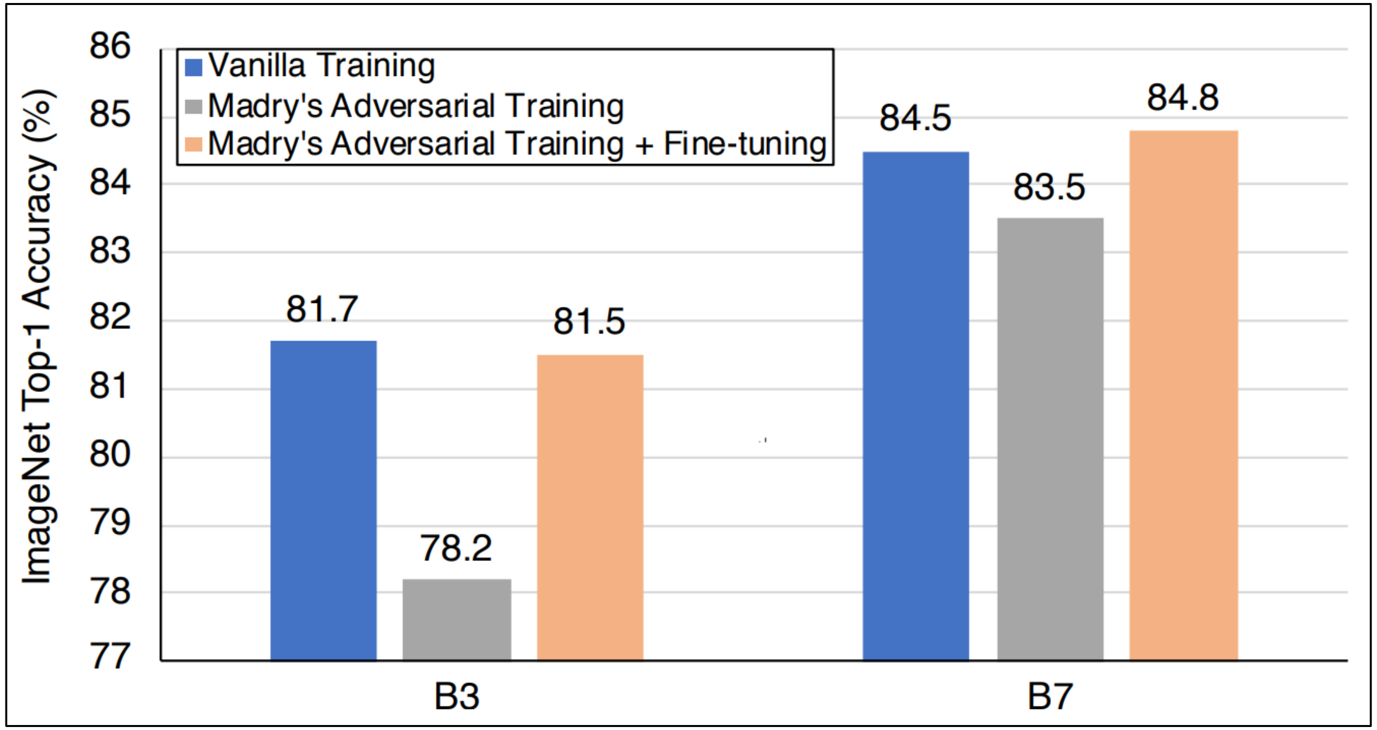
<Adversarial Training vs Data Augmentation>



From [Paper Review] Defence-GAN (Seungwan Seo, 2020)

<Adversarial Training vs Data Augmentation>

<Adversarial Training & Fine-tuning>

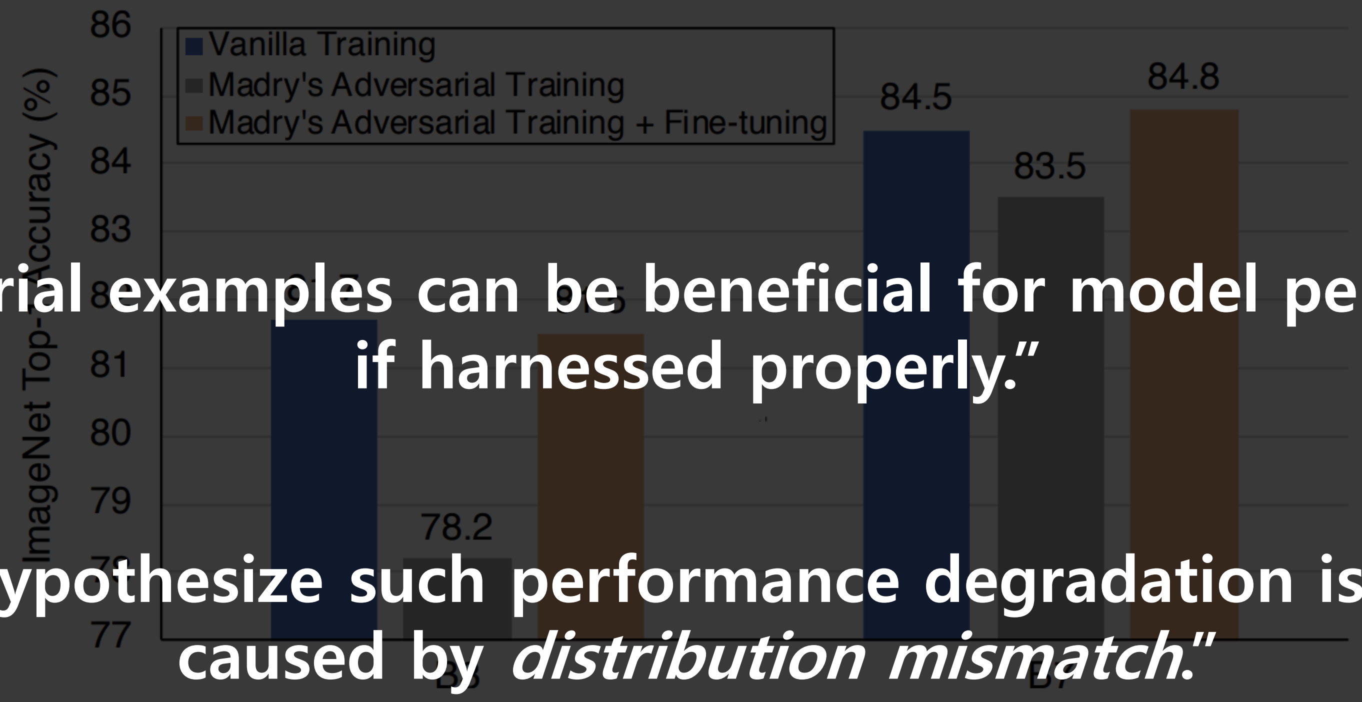


Two take-home message from the experiments on ImageNet:

(1) Training exclusively on adversarial examples results in performance degradation

(2) Fine-tuning with clean images can improve network performance

<Adversarial Training & Fine-tuning>



"Adversarial examples can be beneficial for model performance if harnessed properly."

"We hypothesize such performance degradation is mainly caused by *distribution mismatch*."

Two take-home message from the experiments on ImageNet:

(1) Training exclusively on adversarial examples results in performance degradation

(2) Fine-tuning with clean images can improve network performance

Methodology

- PGD-based Adversarial Training
- Disentangled Learning
- AdvProp

Methodology

- Projected Gradient Descent

<Projected Gradient Descent>

$$\min_{\theta} \mathbb{E}_{(x,y)\sim\mathbb{D}}[L(\theta, x, y)]$$

Notations

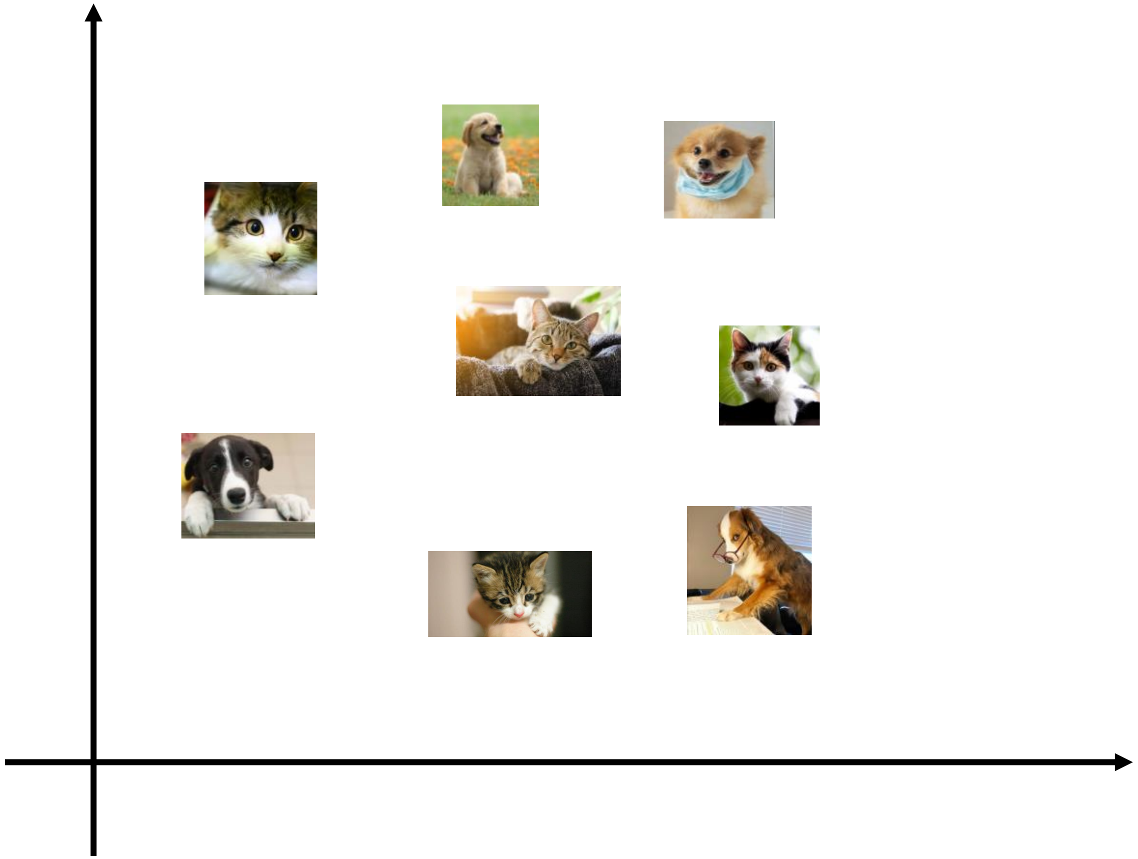
$\mathbb{D}$: *Underlying Distribution*

$L(\cdot,\cdot,\cdot)$: *Loss Function*

θ: *Network Parameters*

x: *Training Sample*

y: *Ground Truth Label*



Methodology

- Projected Gradient Descent

<Projected Gradient Descent>

$$\min_{\theta} \mathbb{E}_{(x,y)\sim\mathbb{D}}[L(\theta, x, y)]$$

Notations

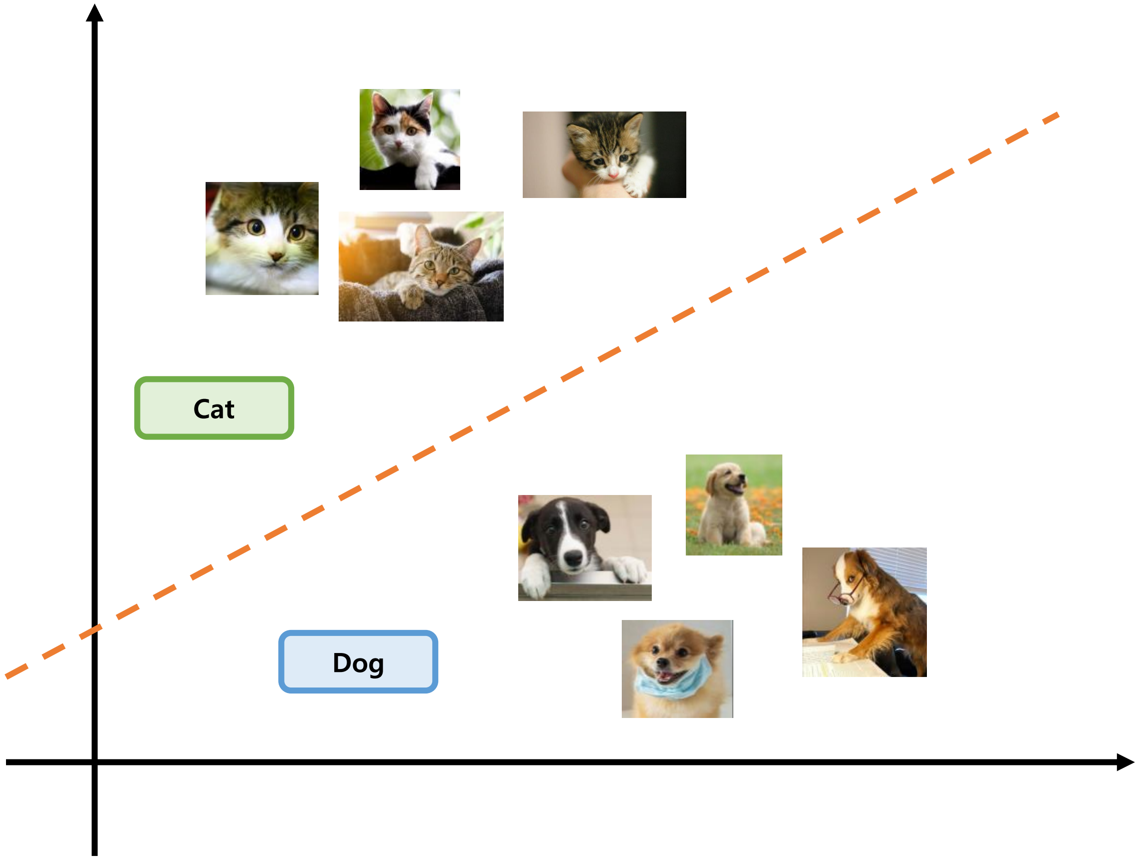
$\mathbb{D}$: *Underlying Distribution*

$L(\cdot,\cdot,\cdot)$: *Loss Function*

θ: *Network Parameters*

x: *Training Sample*

y: *Ground Truth Label*



Methodology

- Projected Gradient Descent

<Projected Gradient Descent>

$$\min_{\theta} \mathbb{E}_{(x,y)\sim\mathbb{D}} \left[\max_{\epsilon \in \mathbb{S}} L(\theta, x + \epsilon, y) \right]$$

Notations

$\mathbb{D}$: *Underlying Distribution*

$L(\cdot,\cdot,\cdot)$: *Loss Function*

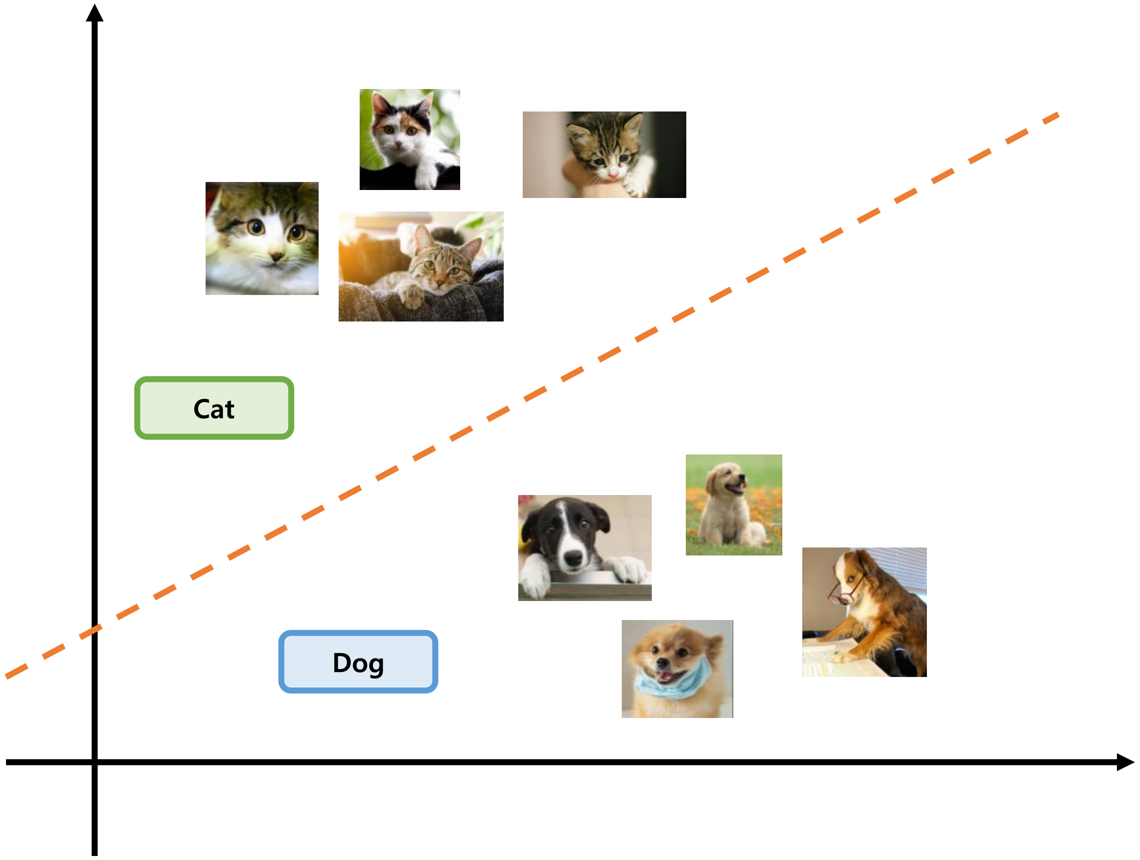
θ: *Network Parameters*

x: *Training Sample*

y: *Ground Truth Label*

ϵ: *Adversarial Perturbation*

$\mathbb{S}$: *Perturbation Range*



From [Paper Review] FreeLB: Enhanced Adversarial Training for Natural Language Understanding (Myeongsup kim, 2021)

Methodology

- Projected Gradient Descent

<Projected Gradient Descent>

$$\min_{\theta} \mathbb{E}_{(x,y)\sim\mathbb{D}} \left[\max_{\epsilon \in \mathbb{S}} L(\theta, x + \epsilon, y) \right]$$

Notations

$\mathbb{D}$: *Underlying Distribution*

$L(\cdot,\cdot,\cdot)$: *Loss Function*

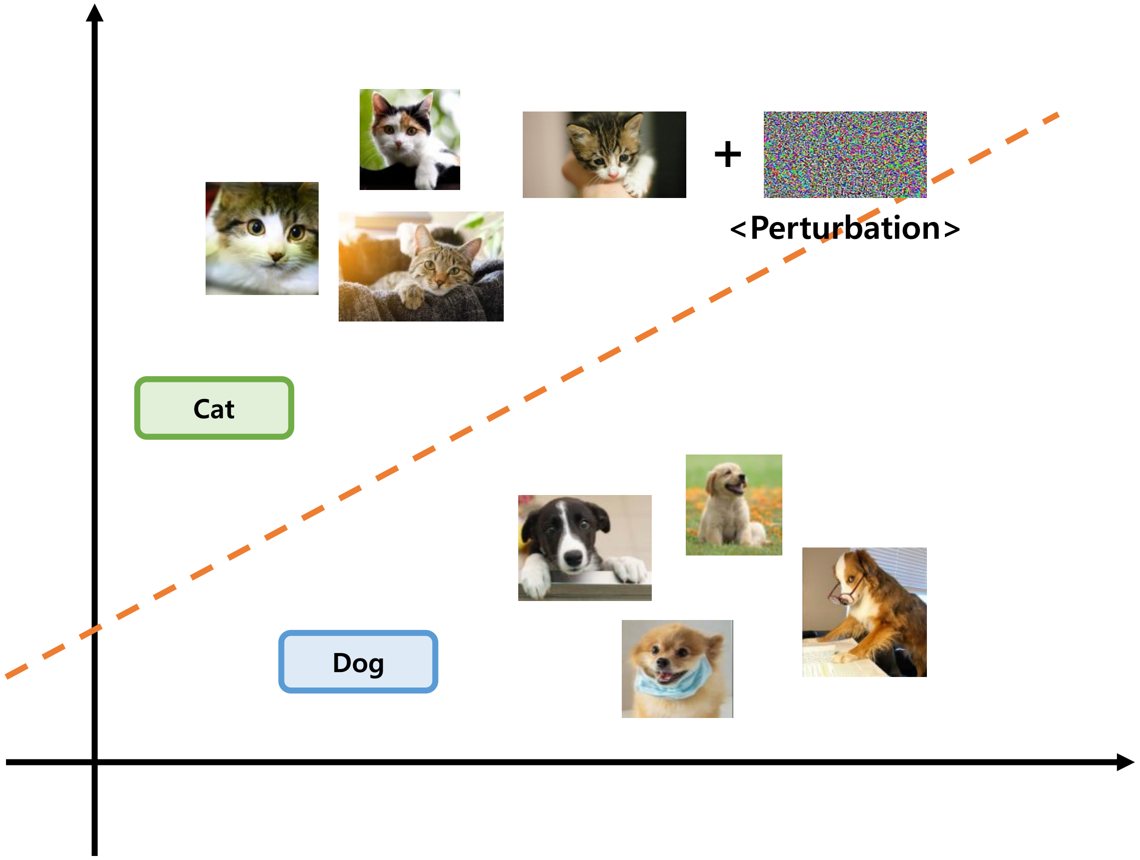
θ: *Network Parameters*

x: *Training Sample*

y: *Ground Truth Label*

ϵ: Adversarial Perturbation

$\mathbb{S}$: *Perturbation Range*



From [Paper Review] FreeLB: Enhanced Adversarial Training for Natural Language Understanding (Myeongsup kim, 2021)

<Projected Gradient Descent>

$$\min_{\theta} \mathbb{E}_{(x,y)\sim\mathbb{D}}\left[\max_{\epsilon\in\mathbb{S}} L(\theta, x + \epsilon, y)\right]$$

Notations

$\mathbb{D}$: *Underlying Distribution*

$L(\cdot,\cdot,\cdot)$: *Loss Function*

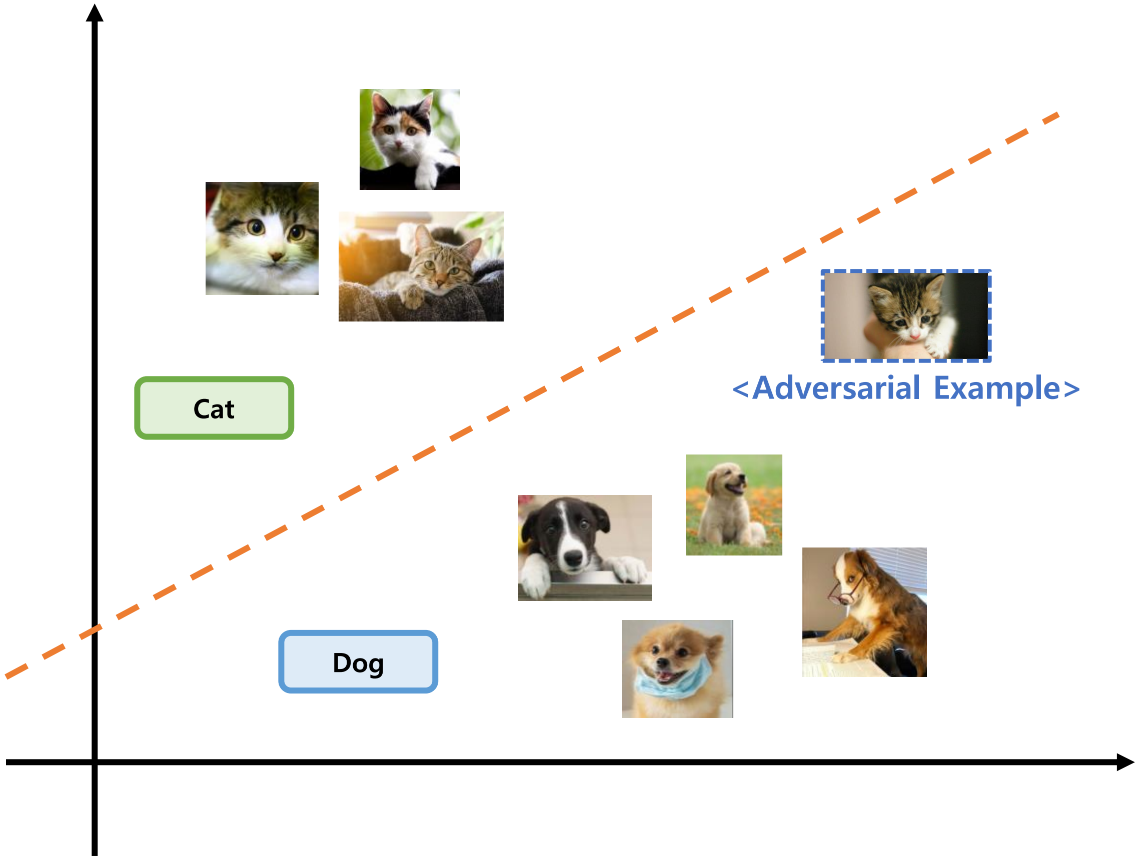
θ: *Network Parameters*

x: *Training Sample*

y: *Ground Truth Label*

ϵ: ***Adversarial Perturbation***

$\mathbb{S}$: *Perturbation Range*



Methodology

- Projected Gradient Descent

<Projected Gradient Descent>

$$\min_{\theta} \mathbb{E}_{(x,y)\sim\mathbb{D}}\left[\max_{\epsilon\in\mathbb{S}} L(\theta, x+\epsilon, y)\right]$$

Notations

$\mathbb{D}$: *Underlying Distribution*

$L(\cdot,\cdot,\cdot)$: *Loss Function*

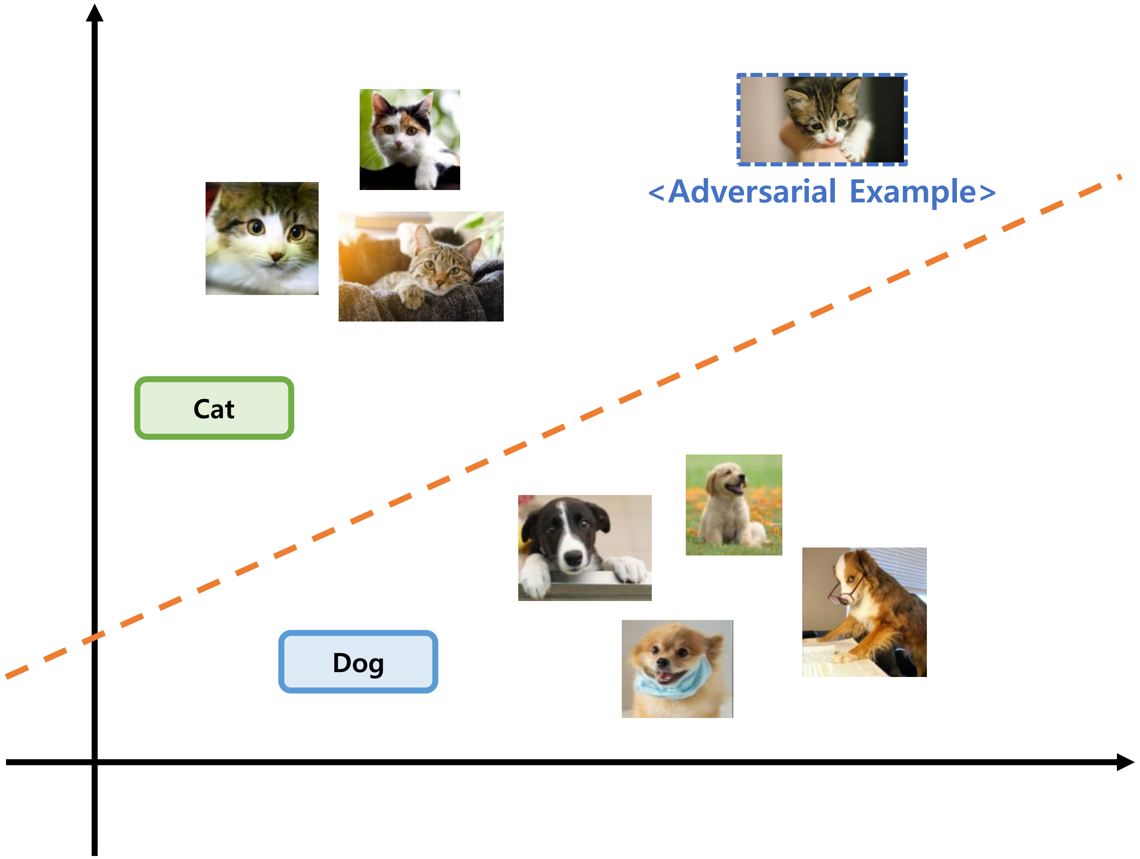
θ: ***Network Parameters***

x: *Training Sample*

y: *Ground Truth Label*

ϵ: *Adversarial Perturbation*

$\mathbb{S}$: *Perturbation Range*



Methodology

- Projected Gradient Descent

<Projected Gradient Descent>

$$\min_{\theta} \mathbb{E}_{(x,y)\sim\mathbb{D}}\left[\max_{\epsilon\in\mathbb{S}} L(\theta, x+\epsilon, y)\right]$$

Notations

$\mathbb{D}$: *Underlying Distribution*

$L(\cdot,\cdot,\cdot)$: *Loss Function*

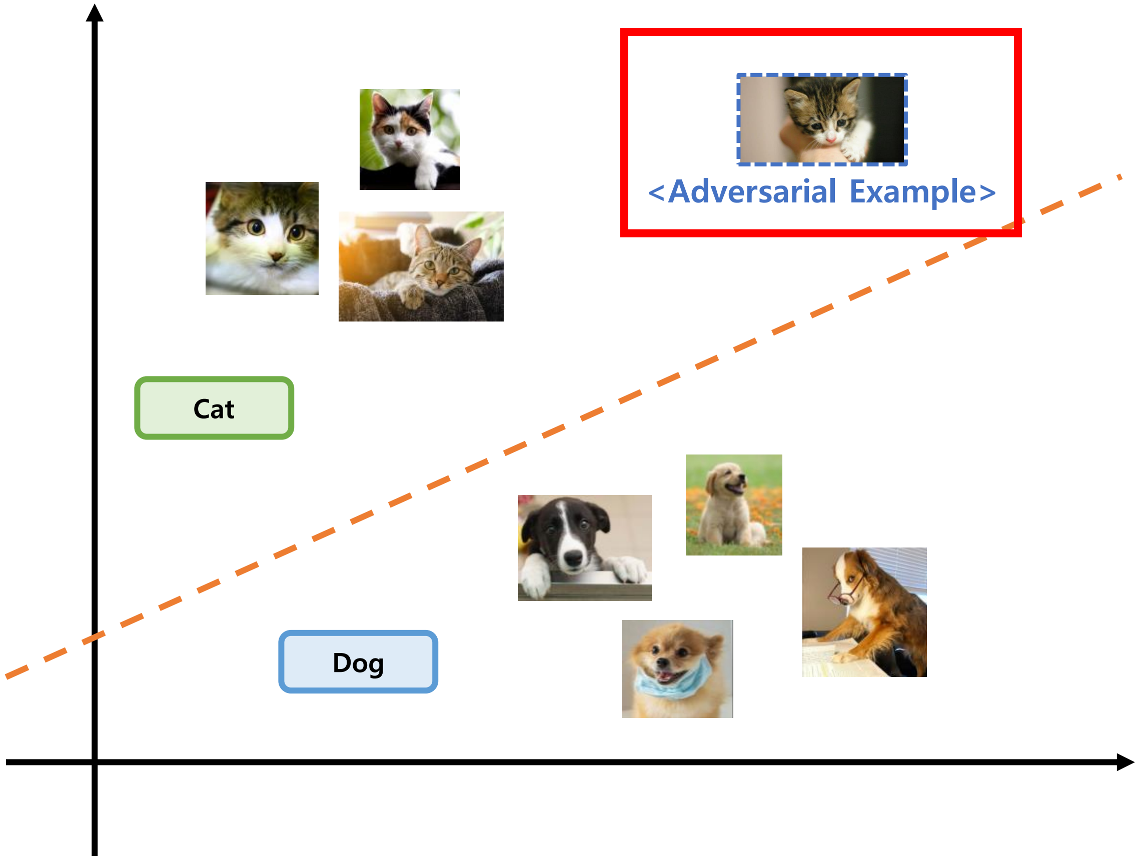
θ: *Network Parameters*

x: *Training Sample*

y: *Ground Truth Label*

ϵ: ***Adversarial Perturbation***

$\mathbb{S}$: ***Perturbation Range***



From [Paper Review] FreeLB: Enhanced Adversarial Training for Natural Language Understanding (Myeongsup kim, 2021)

Methodology

- Projected Gradient Descent

<Projected Gradient Descent>

$$\min_{\theta} \mathbb{E}_{(x,y)\sim\mathbb{D}}\left[\max_{\epsilon\in\mathbb{S}} L(\theta, x+\epsilon, y)\right]$$

Notations

$\mathbb{D}$: *Underlying Distribution*

$L(\cdot,\cdot,\cdot)$: *Loss Function*

θ: *Network Parameters*

x: *Training Sample*

y: *Ground Truth Label*

ϵ: ***Adversarial Perturbation***

$\mathbb{S}$: ***Perturbation Range***

<Adversarial Example>

Methodology

- Projected Gradient Descent

<Projected Gradient Descent>

$$\min_{\theta} \mathbb{E}_{(x,y)\sim\mathbb{D}} \left[\max_{\epsilon \in \mathbb{S}} L(\theta, x + \epsilon, y) \right]$$

Notations

$\mathbb{D}$: *Underlying Distribution*

$L(\cdot,\cdot,\cdot)$: *Loss Function*

θ: *Network Parameters*

x: *Training Sample*

y: *Ground Truth Label*

ϵ: ***Adversarial Perturbation***

$\mathbb{S}$: ***Perturbation Range***

<Adversarial Example>

Methodology
- Projected Gradient Descent

<Proposed Objective>

$$\min_{\theta}\left[\mathbb{E}_{(x,y)\sim\mathbb{D}}\left[\underbrace{L(\theta,x,y)}_{\text{Clean Image}}+\underbrace{\max_{\epsilon\in\mathbb{S}}L(\theta,x+\epsilon,y)}_{\text{Adversarial Example}}\right]\right]$$

Notations

$\mathbb{D}$: *Underlying Distribution*

$L(\cdot,\cdot,\cdot)$: *Loss Function*

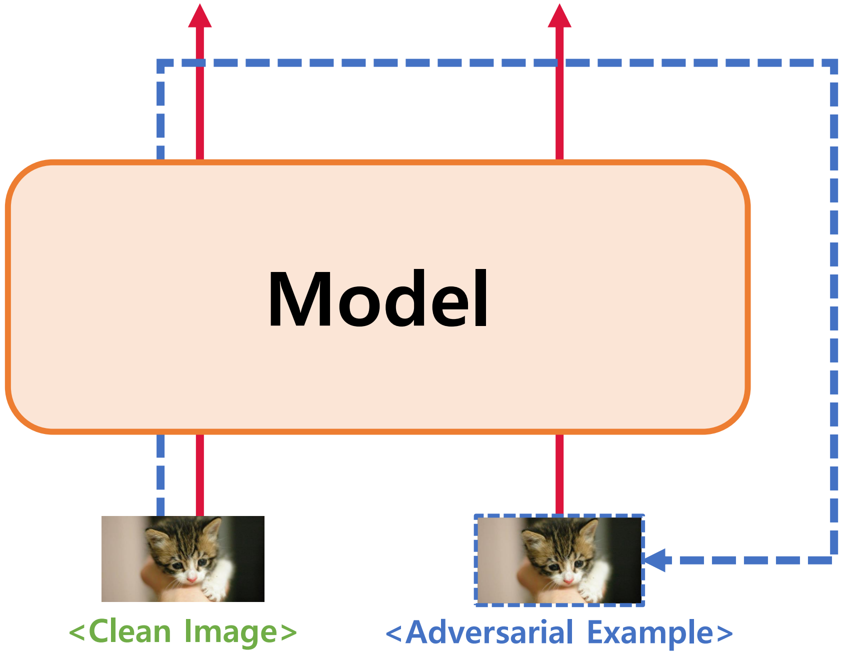
θ: ***Network Parameters***

x: ***Training Sample***

y: *Ground Truth Label*

ϵ: ***Adversarial Perturbation***

$\mathbb{S}$: *Perturbation Range*



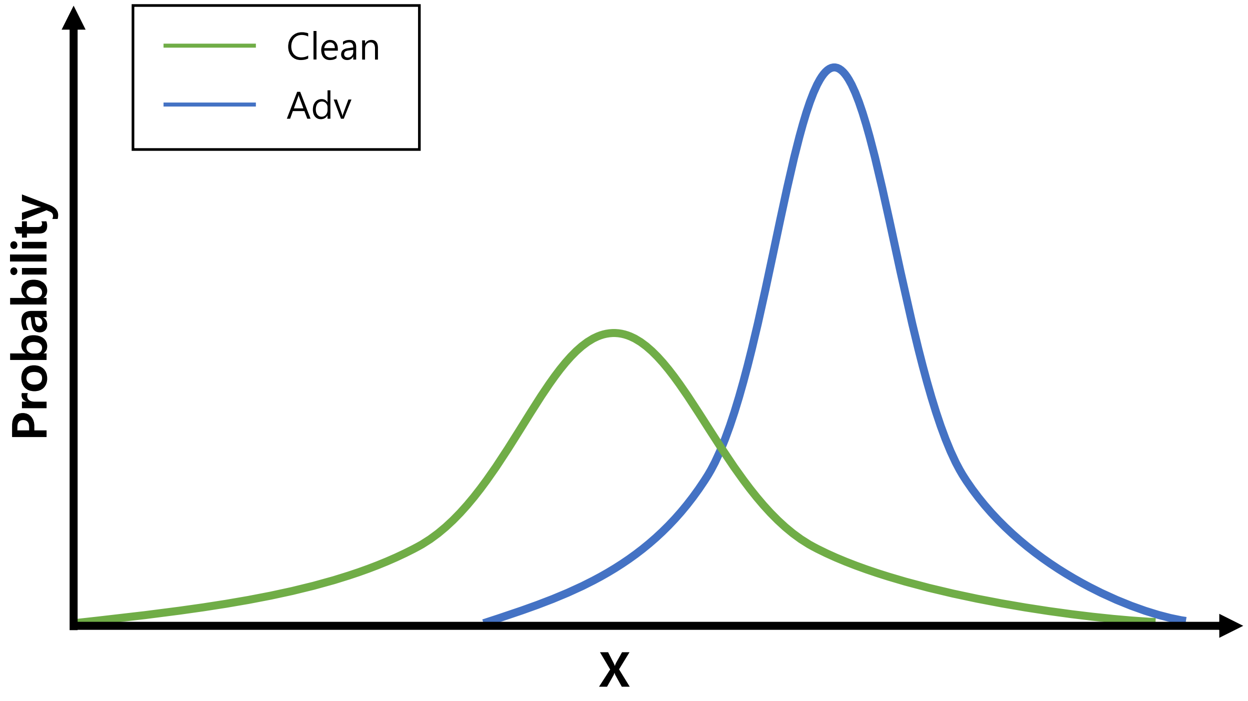
Gradient Ascent

Gradient Descent

Methodology

- Disentangled Learning

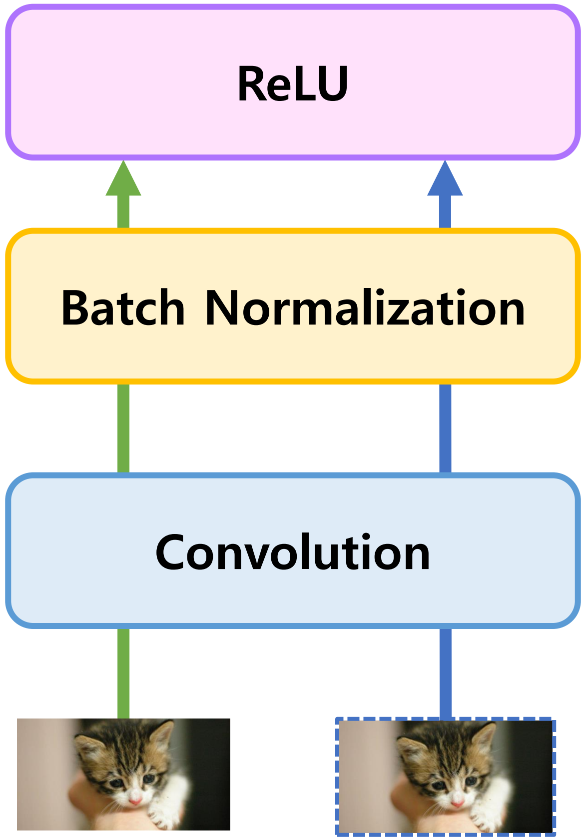
<Distribution Mismatch>



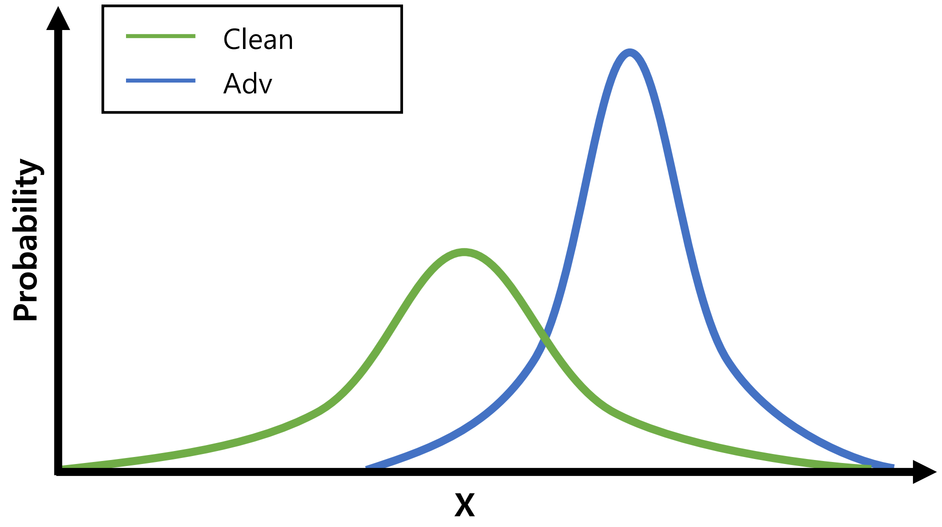
<Clean Image>

<Adversarial Example>

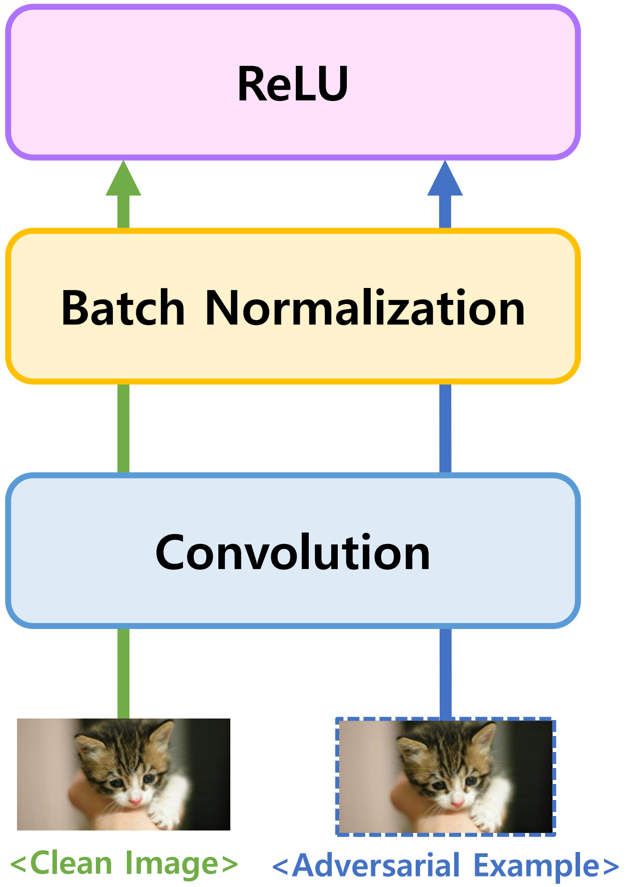
<Distribution Mismatch>

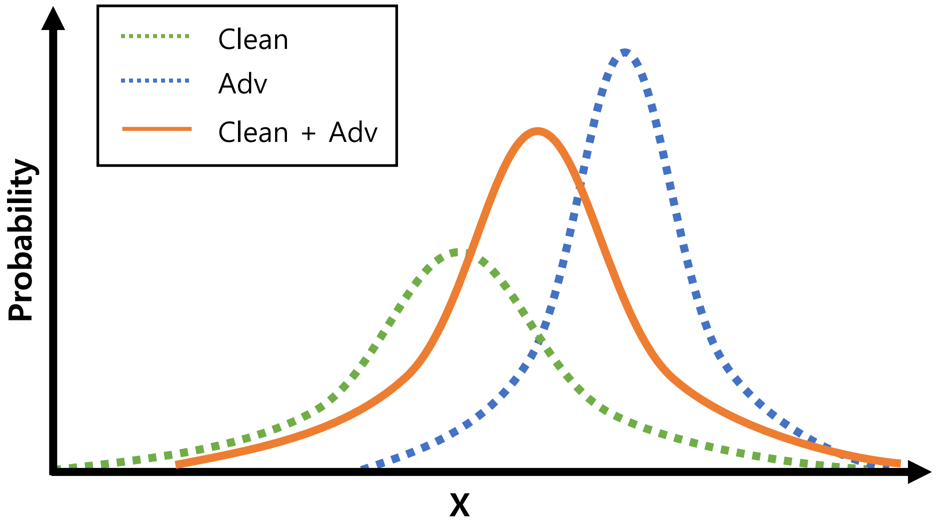


<Clean Image> <Adversarial Example>



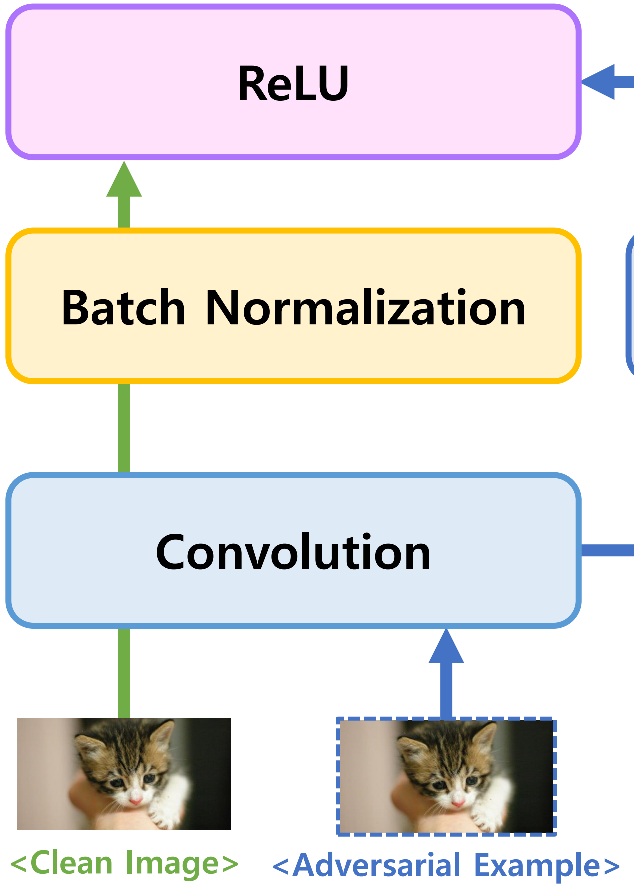
<Distribution Mismatch>

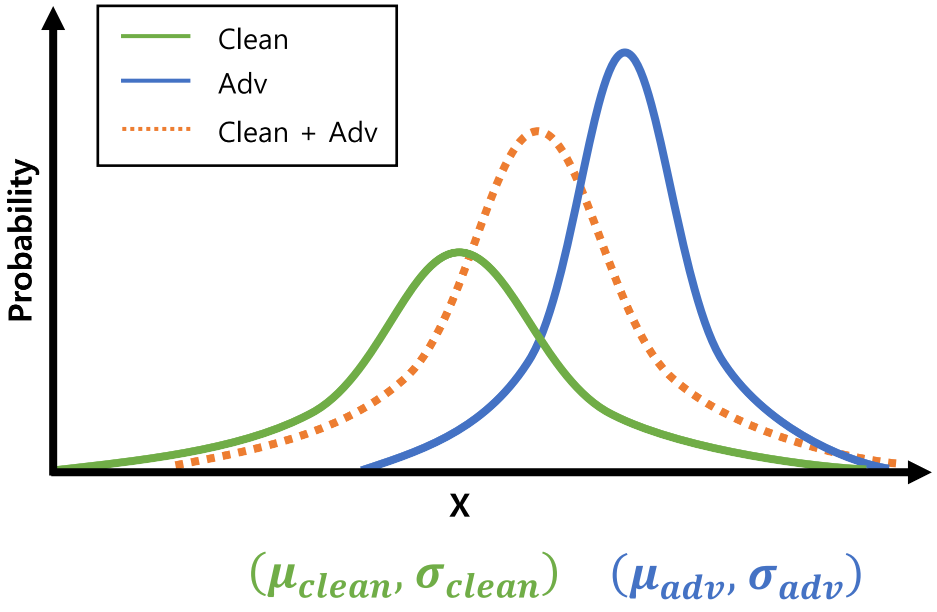




$(\mu_{clean+adv}, \sigma_{clean+adv})$

<Disentangled Learning>





<Adversarial Propagation>

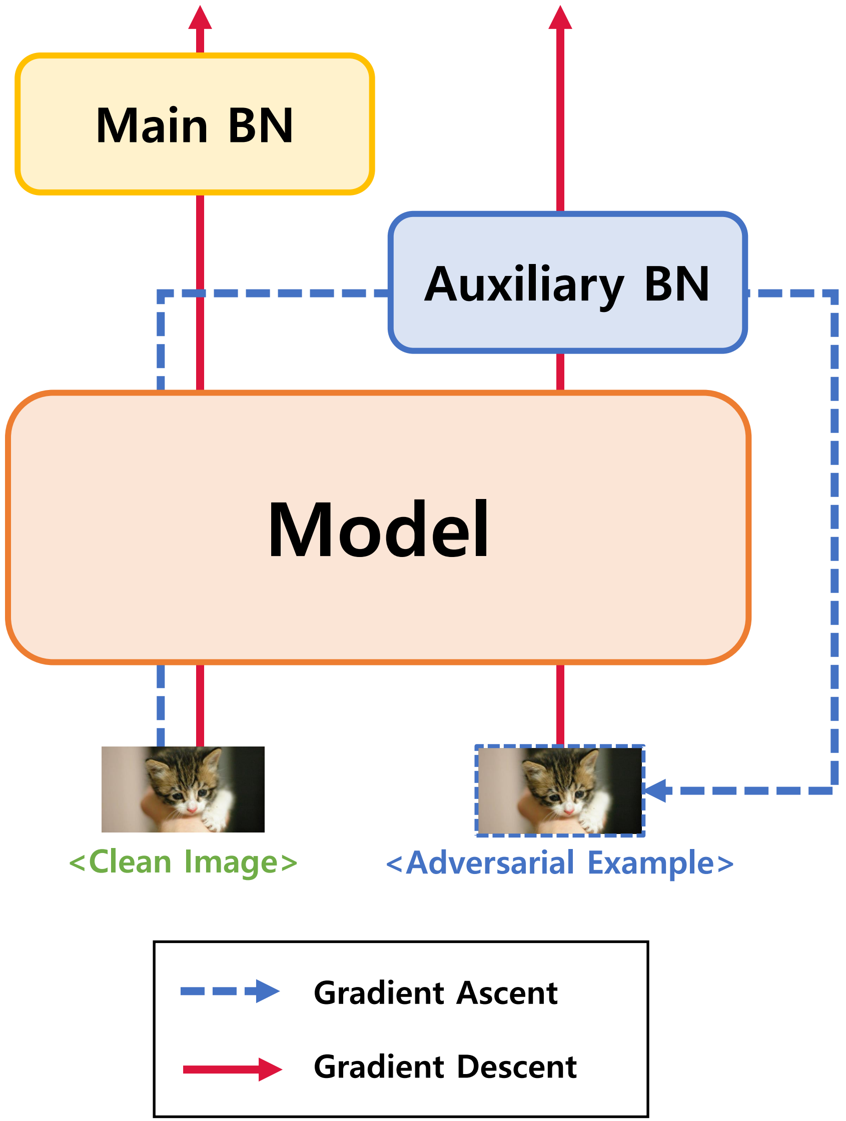
Algorithm 1: Pseudo code of AdvProp

Data: A set of clean images with labels;
Result: Network parameter θ;
for each training step **do**
  Sample a clean image mini-batch x^c with label y;
  Generate the corresponding adversarial mini-batch x^a using auxiliary BNs;
  Compute loss $L^c(\theta, x^c, y)$ on clean mini-batch x^c using the main BNs;
  Compute loss $L^a(\theta, x^a, y)$ on adversarial mini-batch x^a using the auxiliary BNs;
  Minimize the total loss w.r.t. network parameter $\min_{\theta}[L^a(\theta, x^a, y) + L^c(\theta, x^c, y)]$
end
return θ

Methodology

- AdvProp

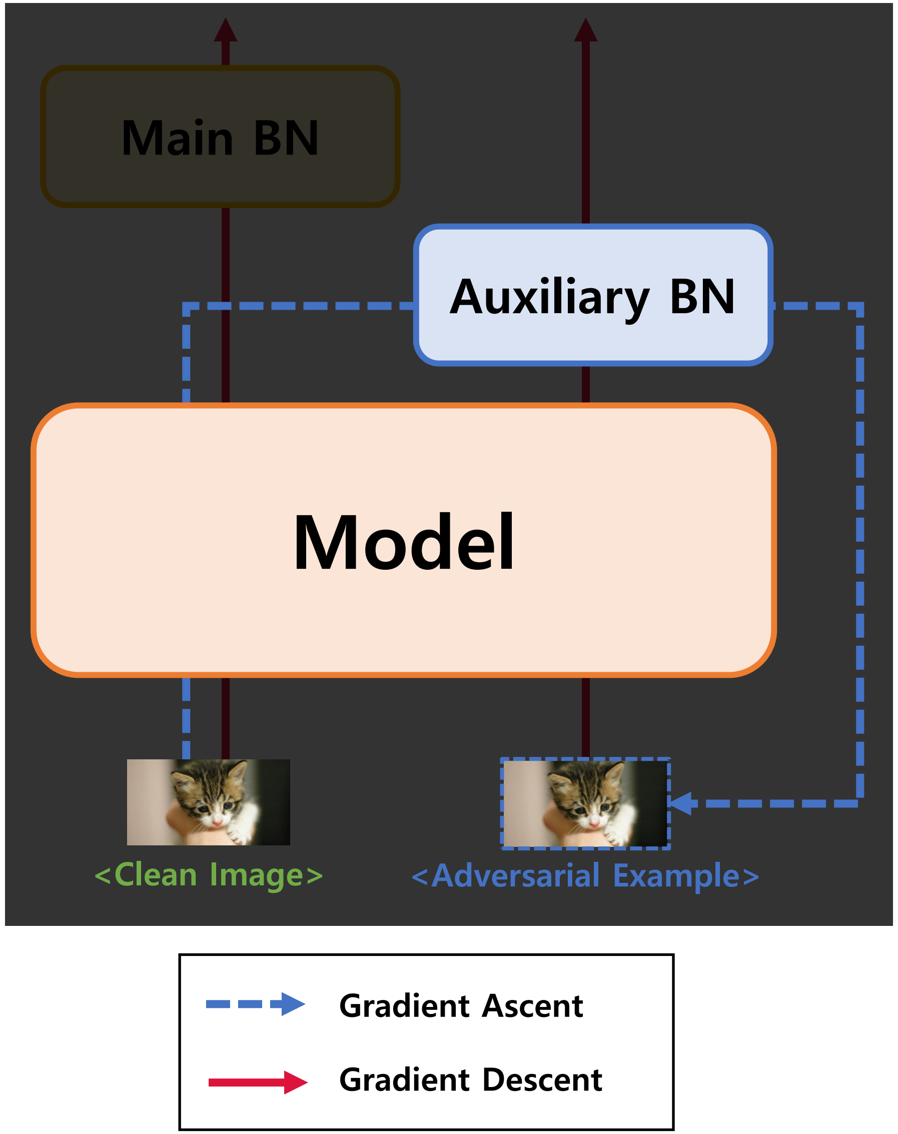
<Adversarial Propagation>



Algorithm 1: Pseudo code of AdvProp

Data: A set of clean images with labels;
Result: Network parameter θ;
for each training step **do**
  Sample a clean image mini-batch x^c with label y;
  Generate the corresponding adversarial mini-batch x^a using auxiliary BNs;
  Compute loss $L^c(\theta, x^c, y)$ on clean mini-batch x^c using the main BNs;
  Compute loss $L^a(\theta, x^a, y)$ on adversarial mini-batch x^a using the auxiliary BNs;
  Minimize the total loss w.r.t. network parameter $\min_{\theta}[L^a(\theta, x^a, y) + L^c(\theta, x^c, y)]$
end
return θ

<Adversarial Propagation>



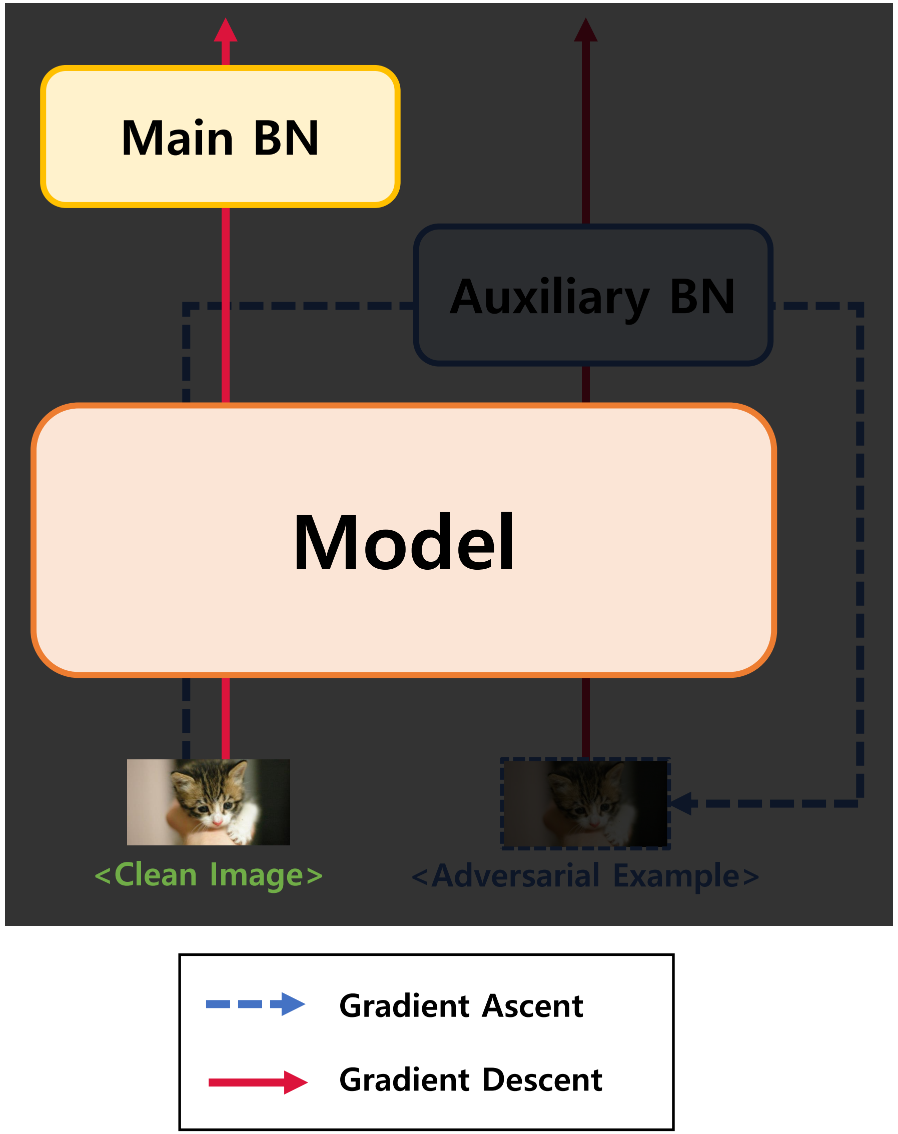
Algorithm 1: Pseudo code of AdvProp

Data: A set of clean images with labels;
Result: Network parameter θ;
for each training step **do**
  Sample a clean image mini-batch x^c with label y;
  Generate the corresponding adversarial mini-batch x^a using auxiliary BNs;
  Compute loss $L^c(\theta, x^c, y)$ on clean mini-batch x^c using the main BNs;
  Compute loss $L^a(\theta, x^a, y)$ on adversarial mini-batch x^a using the auxiliary BNs;
  Minimize the total loss w.r.t. network parameter $\min_\theta [L^a(\theta, x^a, y) + L^c(\theta, x^c, y)]$
end
return θ

Methodology

- AdvProp

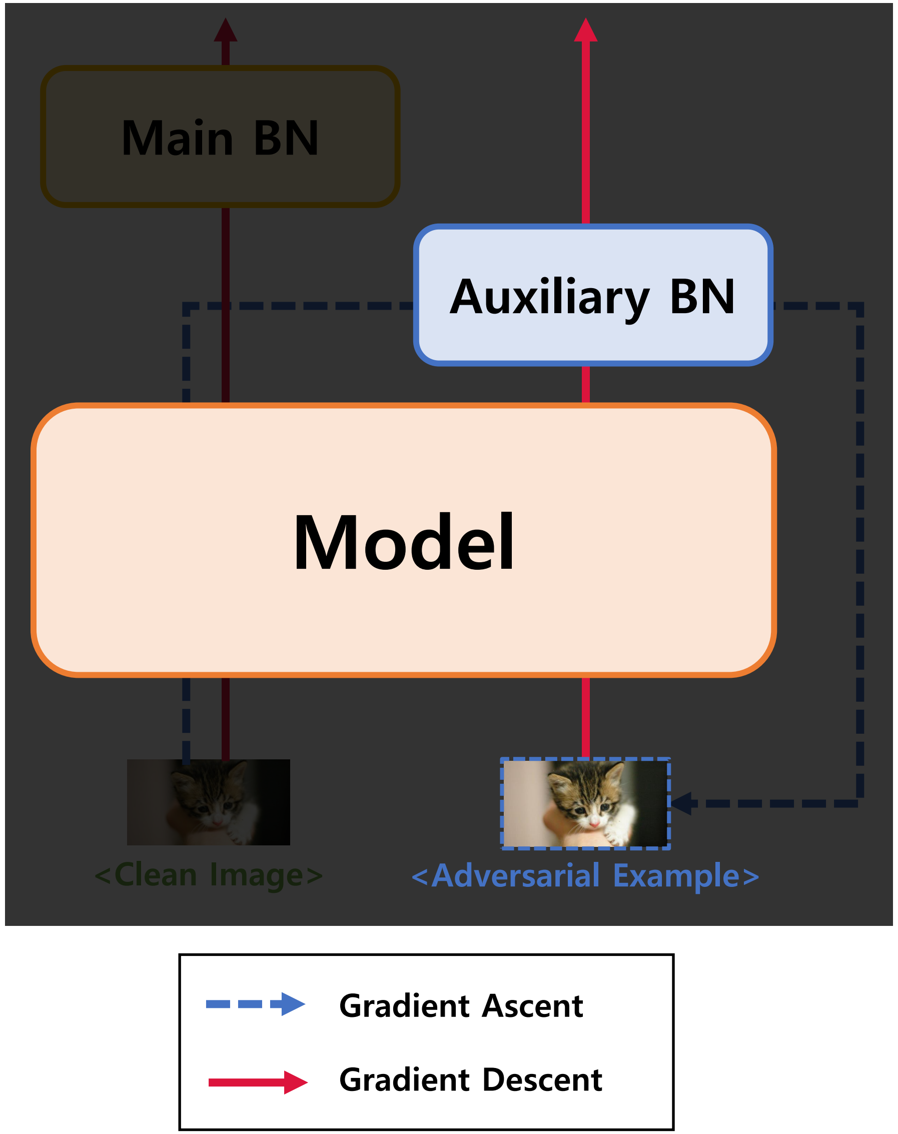
<Adversarial Propagation>



Algorithm 1: Pseudo code of AdvProp

Data: A set of clean images with labels;
Result: Network parameter θ;
for each training step **do**
  Sample a clean image mini-batch x^c with label y;
  Generate the corresponding adversarial mini-batch x^a using auxiliary BNs;
  Compute loss $L^c(\theta, x^c, y)$ on clean mini-batch x^c using the main BNs;
  Compute loss $L^a(\theta, x^a, y)$ on adversarial mini-batch x^a using the auxiliary BNs;
  Minimize the total loss w.r.t. network parameter $\min_{\theta}[L^a(\theta, x^a, y) + L^c(\theta, x^c, y)]$
end
return θ

<Adversarial Propagation>



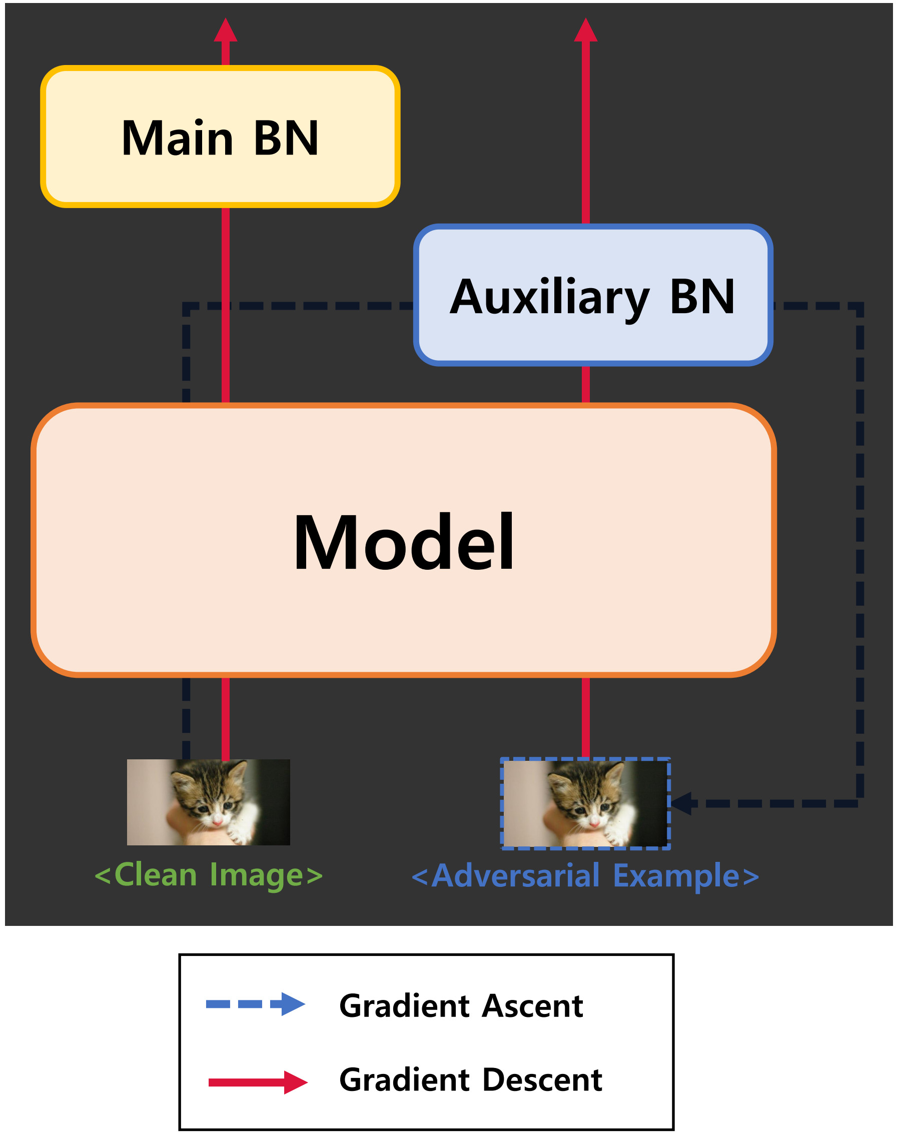
Algorithm 1: Pseudo code of AdvProp

Data: A set of clean images with labels;
Result: Network parameter θ;
for each training step **do**
  Sample a clean image mini-batch x^c with label y;
  Generate the corresponding adversarial mini-batch x^a using auxiliary BNs;
  Compute loss $L^c(\theta, x^c, y)$ on clean mini-batch x^c using the main BNs;
  Compute loss $L^a(\theta, x^a, y)$ on adversarial mini-batch x^a using the auxiliary BNs;
  Minimize the total loss w.r.t. network parameter $\min_{\theta}[L^a(\theta, x^a, y) + L^c(\theta, x^c, y)]$
end
return θ

Methodology

- AdvProp

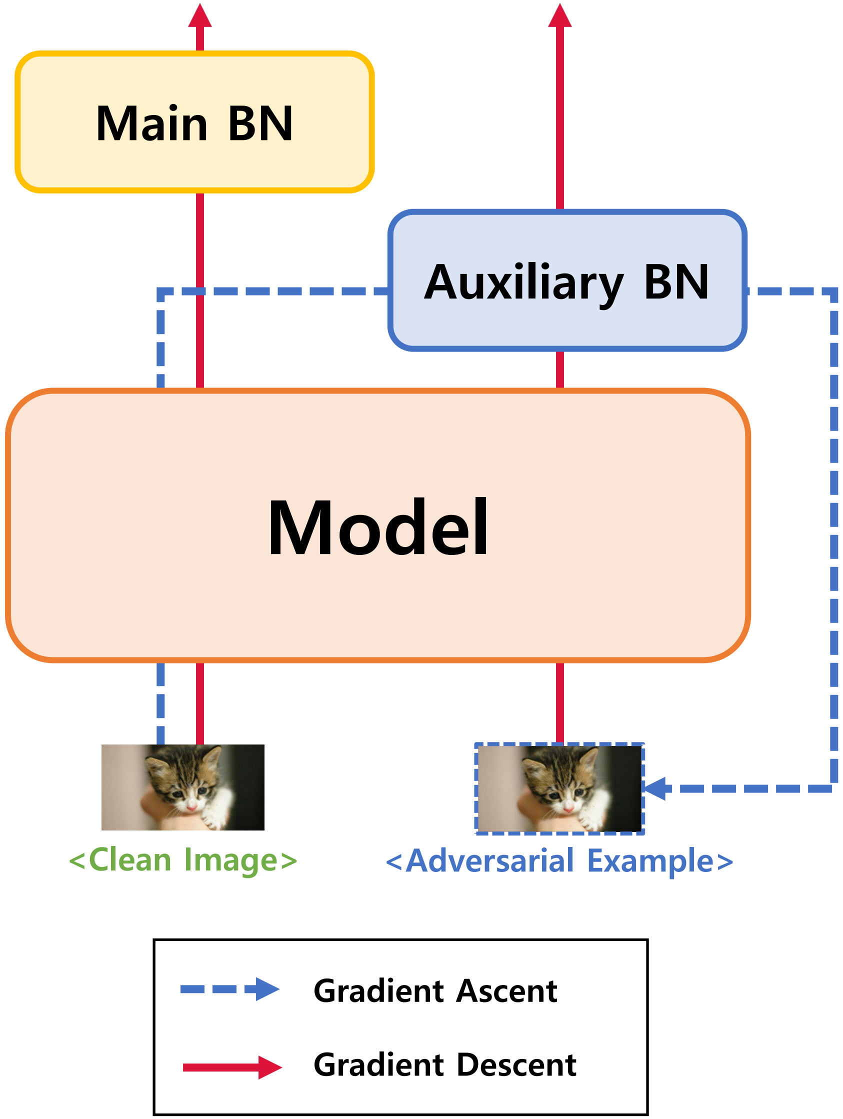
<Adversarial Propagation>



Algorithm 1: Pseudo code of AdvProp

Data: A set of clean images with labels;
Result: Network parameter θ;
for each training step **do**
    Sample a clean image mini-batch x^c with label y;
    Generate the corresponding adversarial mini-batch x^a
        using auxiliary BNs;
    Compute loss $L^c(\theta, x^c, y)$ on clean mini-batch x^c using
        the main BNs;
    Compute loss $L^a(\theta, x^a, y)$ on adversarial mini-batch x^a
        using the auxiliary BNs;
    Minimize the total loss w.r.t. network parameter
        $\min_{\theta}[L^a(\theta, x^a, y) + L^c(\theta, x^c, y)]$
end
return θ

<Adversarial Propagation>



Algorithm 1: Pseudo code of AdvProp

Data: A set of clean images with labels;
Result: Network parameter θ;
for each training step **do**
  Sample a clean image mini-batch x^c with label y;
  Generate the corresponding adversarial mini-batch x^a using auxiliary BNs;
  Compute loss $L^c(\theta, x^c, y)$ on clean mini-batch x^c using the main BNs;
  Compute loss $L^a(\theta, x^a, y)$ on adversarial mini-batch x^a using the auxiliary BNs;
  Minimize the total loss w.r.t. network parameter $\min_{\theta}[L^a(\theta, x^a, y) + L^c(\theta, x^c, y)]$
end
return θ

Experiments

- ImageNet Results

Experiments

- Model

<EfficientNet>

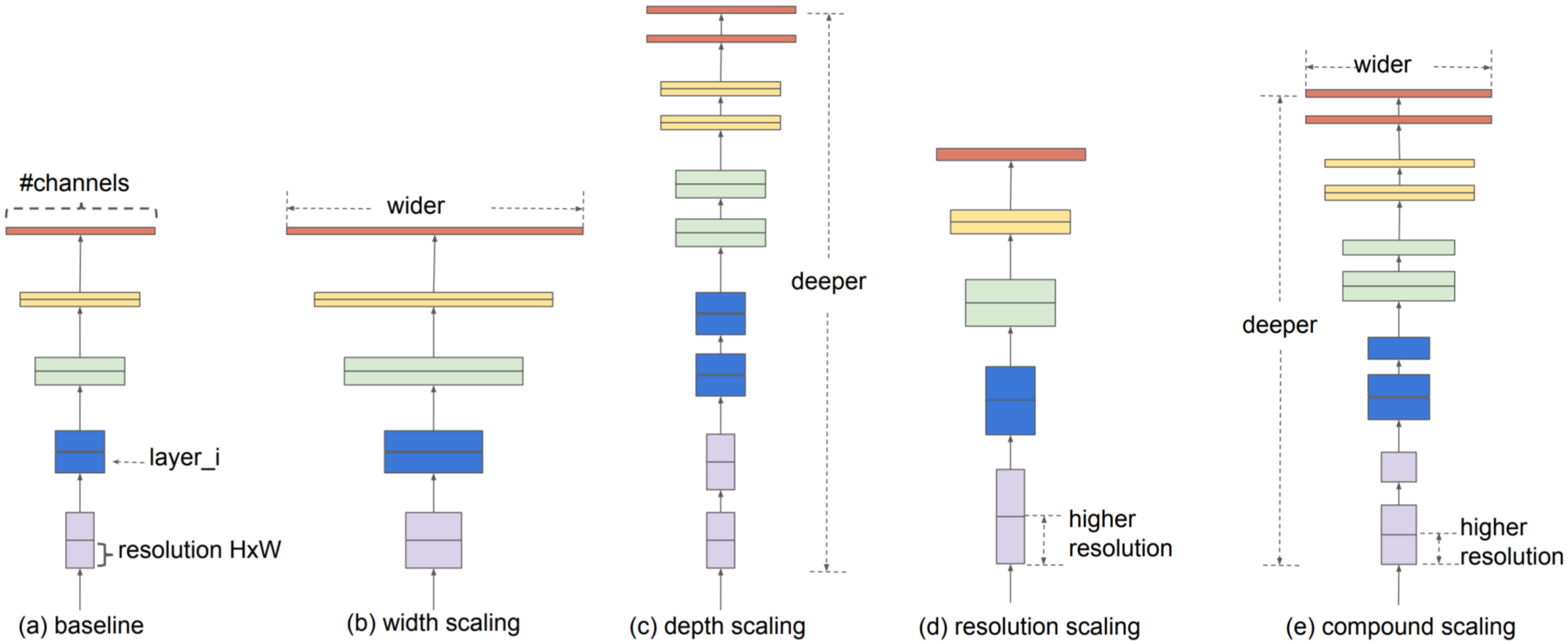


Figure 2. **Model Scaling.** (a) is a baseline network example; (b)-(d) are conventional scaling that only increases one dimension of network width, depth, or resolution. (e) is our proposed compound scaling method that uniformly scales all three dimensions with a fixed ratio.

<Datasets>

- **ImageNet-C**

  Designed for measuring the network robustness to common image corruption.

- **ImageNet-A**

  Adversarially collected natural, unmodified but “hard” real-world images.

- **Stylized-ImageNet**

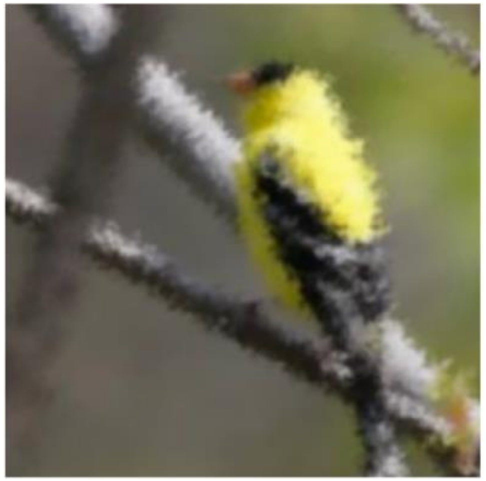
  Removed local texture cues while retaining global shape via style transfer.

Experiments

- ImageNet Results

<Datasets>

<ImageNet>



<ImageNet-C>

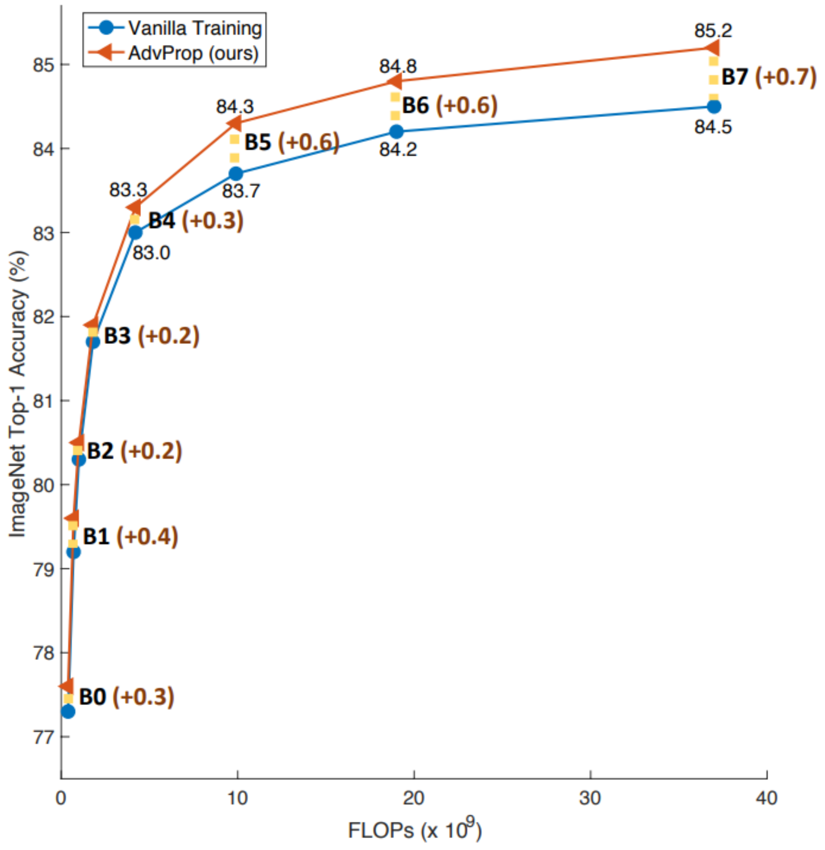
<ImageNet-A>

<Stylized-ImageNet>

Experiments

- ImageNet Results

<ImageNet Results>



"AdvProp boosts model performance over the vanilla training baseline on ImageNet. This Improvement becomes more significant if trained with larger network."

<ImageNet Results>

Model	ImageNet-C	ImageNet-A	Stylized-ImageNet
	mCE (↓)	Top-1 Acc. (↑)	Top-1 Acc. (↑)
ResNet-50	74.8	3.1	8.0
EfficientNet-B0	70.7	6.7	13.1
+AdvProp (ours)	**66.2 (-4.5)**	**7.1 (+0.4)**	**14.6 (+1.5)**
EfficientNet-B1	65.1	9.0	16.8
+AdvProp (ours)	**60.2 (-4.9)**	**10.1 (+1.1)**	**17.8 (+1.0)**
EfficientNet-B2	64.1	10.8	17.8
+AdvProp (ours)	**61.4 (-2.7)**	**11.8 (+1.0)**	**21.4 (+3.6)**
EfficientNet-B3	62.9	17.9	20.2
+AdvProp (ours)	**57.8 (-5.1)**	**18.0 (+0.1)**	**22.5 (+1.7)**
EfficientNet-B4	60.7	26.4	20.2
+AdvProp (ours)	**58.6 (-5.1)**	**27.9 (+1.5)**	**22.5 (+1.7)**
EfficientNet-B5	62.3	29.4	20.8
+AdvProp (ours)	**56.2 (-6.1)**	**34.4 (+5.0)**	**24.4 (+3.6)**
EfficientNet-B6	60.6	34.5	20.9
+AdvProp (ours)	**53.6 (-7.0)**	**40.6 (+6.1)**	**25.9 (+4.0)**
EfficientNet-B7	59.4	37.7	21.8
+AdvProp (ours)	**52.9 (-6.5)**	**44.7 (+7.0)**	**26.6 (+4.8)**

Experiments

- ImageNet Results

<ImageNet Results>

Model	ImageNet-C	ImageNet-A	Stylized-ImageNet
	mCE (↓)	Top-1 Acc. (↑)	Top-1 Acc. (↑)
ResNet-50	74.8	3.1	8.0
EfficientNet-B0	70.7	6.7	13.1
+AdvProp (ours)	**66.2 (-4.5)**	**7.1 (+0.4)**	**14.6 (+1.5)**
EfficientNet-B1	65.1	9.0	16.8
+AdvProp (ours)	**60.2 (-4.9)**	**10.1 (+1.1)**	**17.8 (+1.0)**
EfficientNet-B2	64.1	10.8	17.8
+AdvProp (ours)	**61.4 (-2.7)**	**11.8 (+1.0)**	**21.4 (+3.6)**
EfficientNet-B3	62.9	17.9	20.2
+AdvProp (ours)	**57.8 (-5.1)**	**18.0 (+0.1)**	**22.5 (+1.7)**
EfficientNet-B4	60.7	26.4	20.2
+AdvProp (ours)	**58.6 (-5.1)**	**27.9 (+1.5)**	**22.5 (+1.7)**
EfficientNet-B5	62.3	29.4	20.8
+AdvProp (ours)	**56.2 (-6.1)**	**34.4 (+5.0)**	**24.4 (+3.6)**
EfficientNet-B6	60.6	34.5	20.9
+AdvProp (ours)	**53.6 (-7.0)**	**40.6 (+6.1)**	**25.9 (+4.0)**
EfficientNet-B7	59.4	37.7	21.8
+AdvProp (ours)	**52.9 (-6.5)**	**44.7 (+7.0)**	**26.6 (+4.8)**

<Results on Adversarial Attacker Strength>

	B0	B1	B2	B3	B4	B5	B6	B7
PGD5 ($\epsilon = 4$)	77.1	79.2	80.3	81.8	**83.3**	**84.3**	**84.8**	**85.2**
PGD4 ($\epsilon = 3$)	77.3	79.4	80.4	**81.9**	**83.3**	**84.3**	84.7	85.1
PGD3 ($\epsilon = 2$)	77.4	79.4	80.4	**81.9**	83.1	**84.3**	84.7	85.0
PGD1 ($\epsilon = 1$)	**77.6**	**79.6**	**80.5**	81.8	83.1	**84.3**	84.6	85.0

ImageNet performance of models trained with AdvProp and different attack strength

<Results on Adversarial Attacker Strength>

	B0	B1	B2	B3	B4	B5	B6	B7
PGD5 ($\epsilon = 4$)	77.1	79.2	80.3	81.8	**83.3**	**84.3**	**84.8**	**85.2**
PGD4 ($\epsilon = 3$)	77.3	79.4	80.4	**81.9**	**83.3**	**84.3**	84.7	85.1
PGD3 ($\epsilon = 2$)	77.4	79.4	80.4	**81.9**	83.1	**84.3**	84.7	85.0
PGD1 ($\epsilon = 1$)	**77.6**	**79.6**	**80.5**	81.8	83.1	**84.3**	84.6	85.0

ImageNet performance of models trained with AdvProp and different attack strength

Experiments

- ImageNet Results

<Comparisons to Adversarial Training>

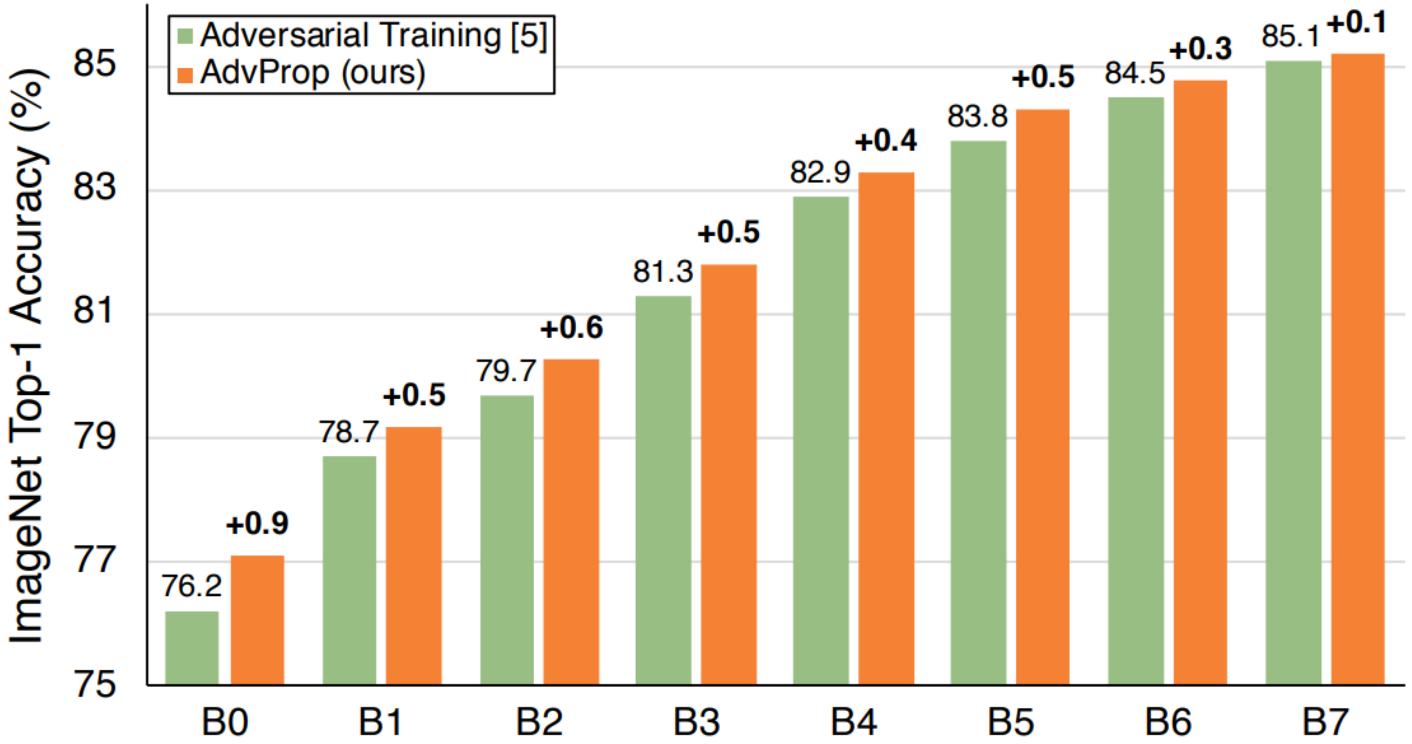
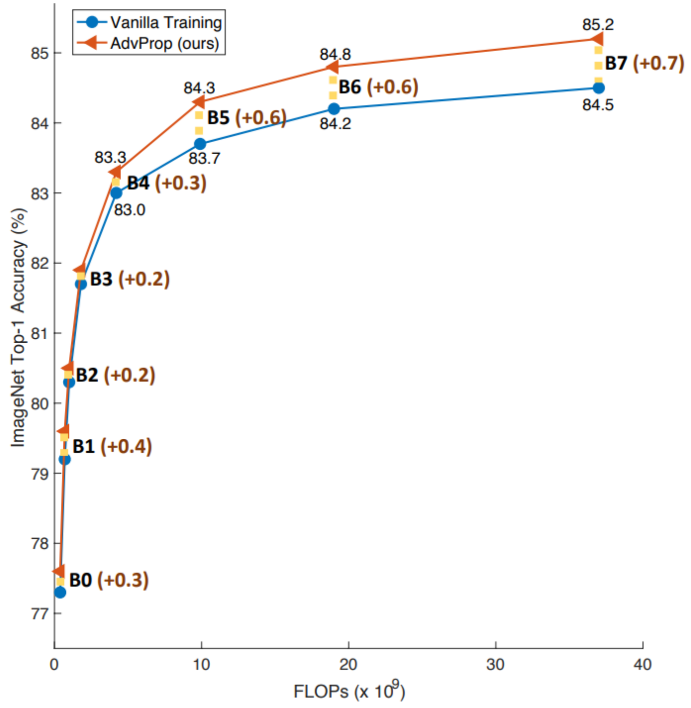
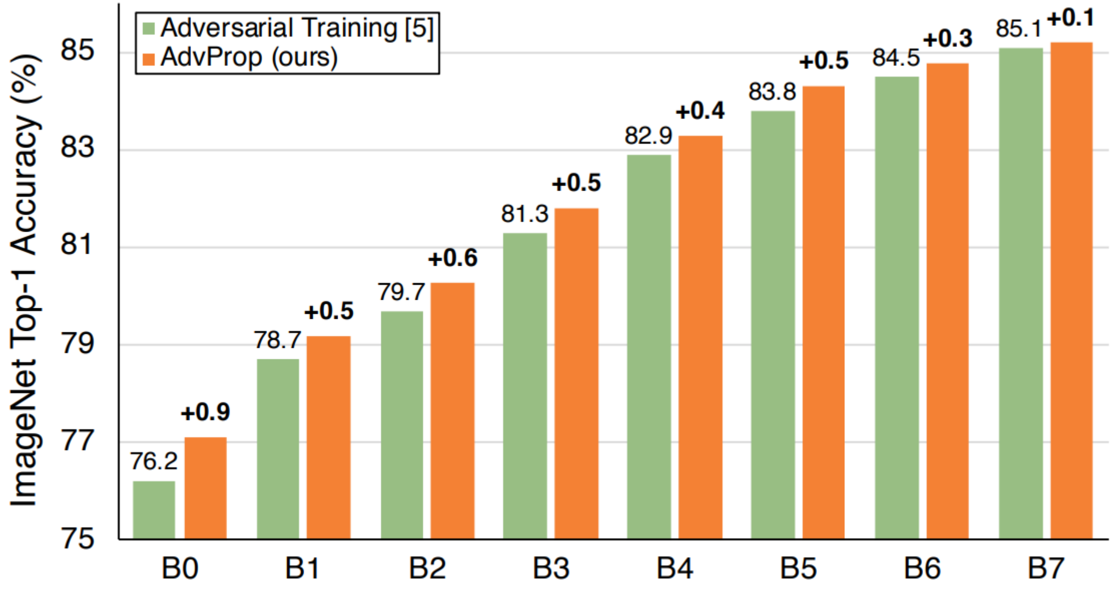


Figure 5. AdvProp substantially outperforms adversarial training [7] on ImageNet, especially for small models.

Experiments
- ImageNet Results

<Comparisons to Adversarial Training>





$$\tilde{J}(\theta, x, y) = \alpha J(\theta, x, y) + (1-\alpha) J\big(\theta, x + \epsilon \mathrm{sign}\big(\nabla_x J(\theta, x, y)\big), y\big)$$

<Fast Gradient Sign Method>

Experiments
- ImageNet Results

<Pushing the Envelope with a Larger Model>

- “To push the envelope, we train a larger network, **EfficientNet-B8**.”
- “Our **AdvProp** improves the **accuracy of EfficientNet-B8 from 84.8% to 85.5%**, achieving a new **state-of-the-art accuracy on ImageNet without using extra data**.”

-2020.07-

<ImageNet Benchmark>

58	**EfficientNetV2-L**	85.7%		121M	×	EfficientNetV2: Smaller Models and Faster Training			2021	EfficientNet
59	**XCiT-S24**	85.6%			×	XCiT: Cross-Covariance Image Transformers			2021	
60	**AdvProp** (EfficientNet-B8)	85.5%	97.3%	88M	×	Adversarial Examples Improve Image Recognition			2019	EfficientNet
61	**HaloNet4** (base 128, Conv-12)	85.5%		87M	×	Scaling Local Self-Attention for Parameter Efficient Visual Backbones			2021	
62	**KDforAA** (EfficientNet-B7)	85.5%		66M	×	Circumventing Outliers of AutoAugment with Knowledge Distillation			2020	EfficientNet

ImageNet Benchmark
-2021.08-

Conclusion

<Conclusion>

- Proposed AdvProp, a new training scheme that bridges the distribution mismatch with a simple yet highly effective two batchnorm approach.

- Showed adversarial examples can improve model performance in the fully-supervised setting on the large-scale ImageNet dataset.

- AdvProp significantly improved accuracy of all ConvNets and reported the state-of-the-art 85.5% top-1 accuracy on ImageNet without any extra data.

Any Questions?

Thank You