Robust Machine Learning in the Adversarial Setting

Tianyu Pang（庞天宇）

Department of Computer Science and Technology
Tsinghua University

Joint work with

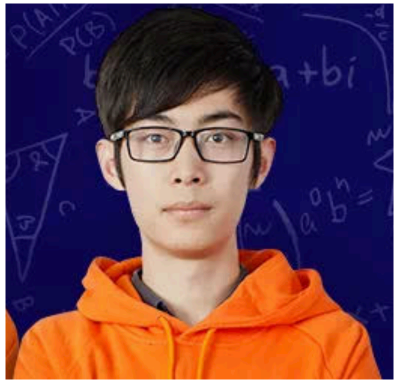
Prof. Jun Zhu

Tianyu Pang

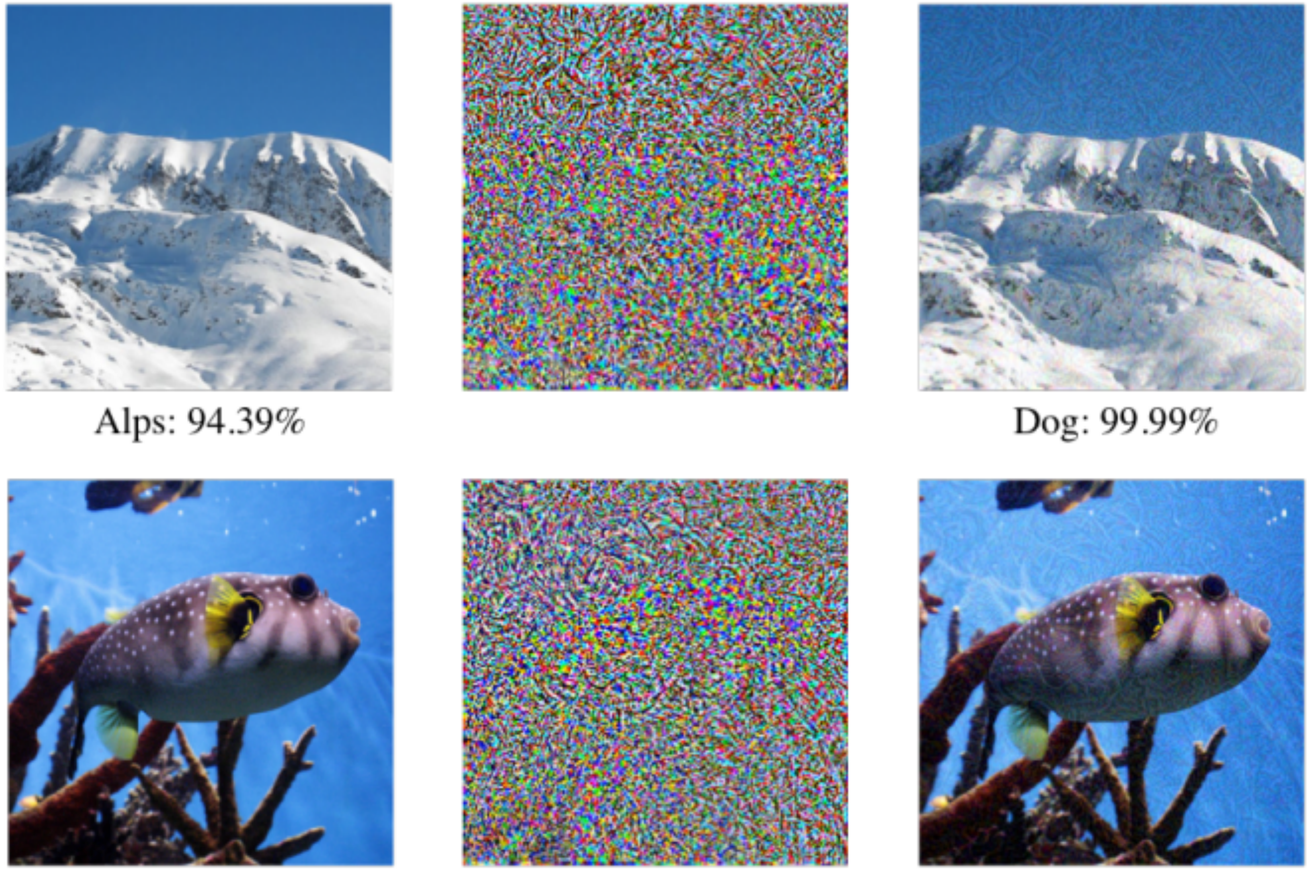
Kun Xu

Chao Du

Yinpeng Dong

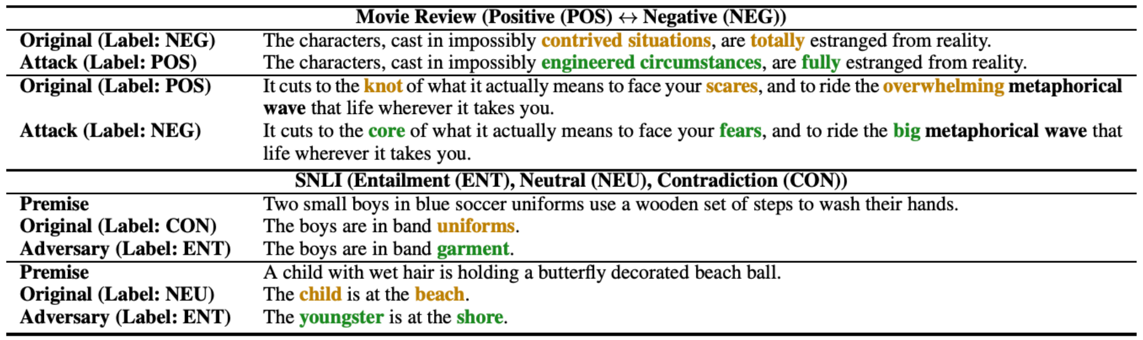

Xiao Yang

Adversarial Examples in Computer Vision



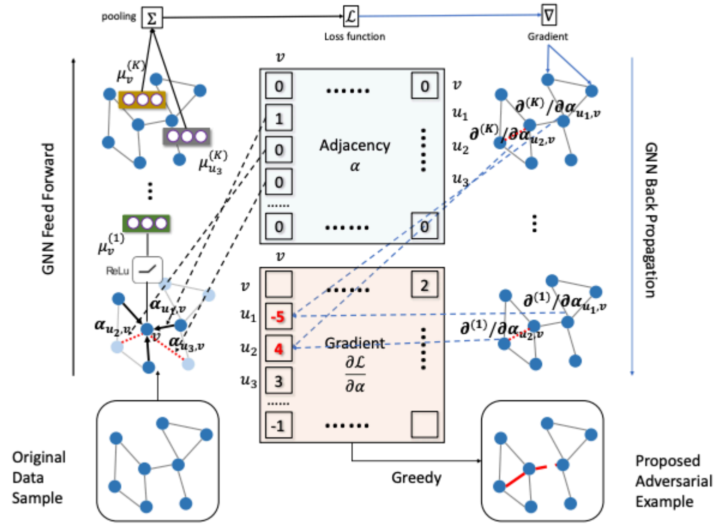
(Dong et al. CVPR 2018)

Not only in Computer Vision



Movie Review (Positive (POS) ↔ Negative (NEG))	
Original (Label: NEG)	The characters, cast in impossibly **contrived situations**, are **totally** estranged from reality.
Attack (Label: POS)	The characters, cast in impossibly **engineered circumstances**, are **fully** estranged from reality.
Original (Label: POS)	It cuts to the **knot** of what it actually means to face your **scares**, and to ride the **overwhelming** **metaphorical wave** that life wherever it takes you.
Attack (Label: NEG)	It cuts to the **core** of what it actually means to face your **fears**, and to ride the **big** **metaphorical wave** that life wherever it takes you.
SNLI (Entailment (ENT), Neutral (NEU), Contradiction (CON))	
Premise	Two small boys in blue soccer uniforms use a wooden set of steps to wash their hands.
Original (Label: CON)	The boys are in band **uniforms**.
Adversary (Label: ENT)	The boys are in band **garment**.
Premise	A child with wet hair is holding a butterfly decorated beach ball.
Original (Label: NEU)	The **child** is at the **beach**.
Adversary (Label: ENT)	The **youngster** is at the **shore**.

BERT model (Jin et al. AAAI 2020)

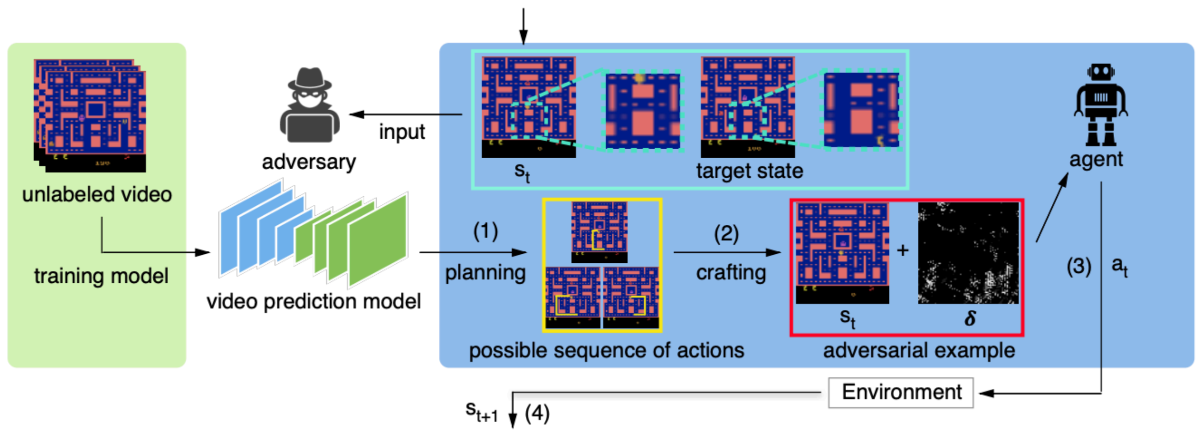


GNN model (Dai et al. ICML 2018)

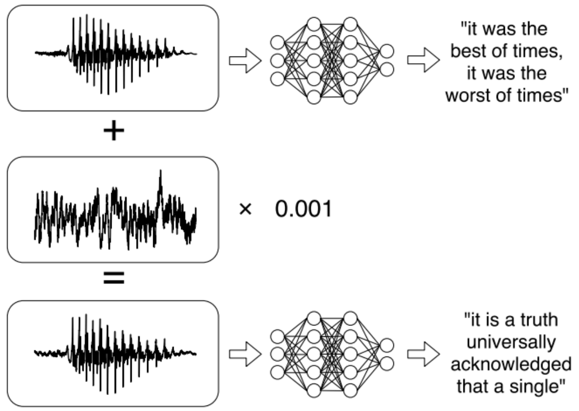
Recommend System,

LIDAR,

……



Reinforcement Learning (Lin et al. IJCAI 2017)


"it was the best of times, it was the worst of times"
+
× 0.001
=
"it is a truth universally acknowledged that a single"

Audio (Carlini and Wagner. S&P 2018)

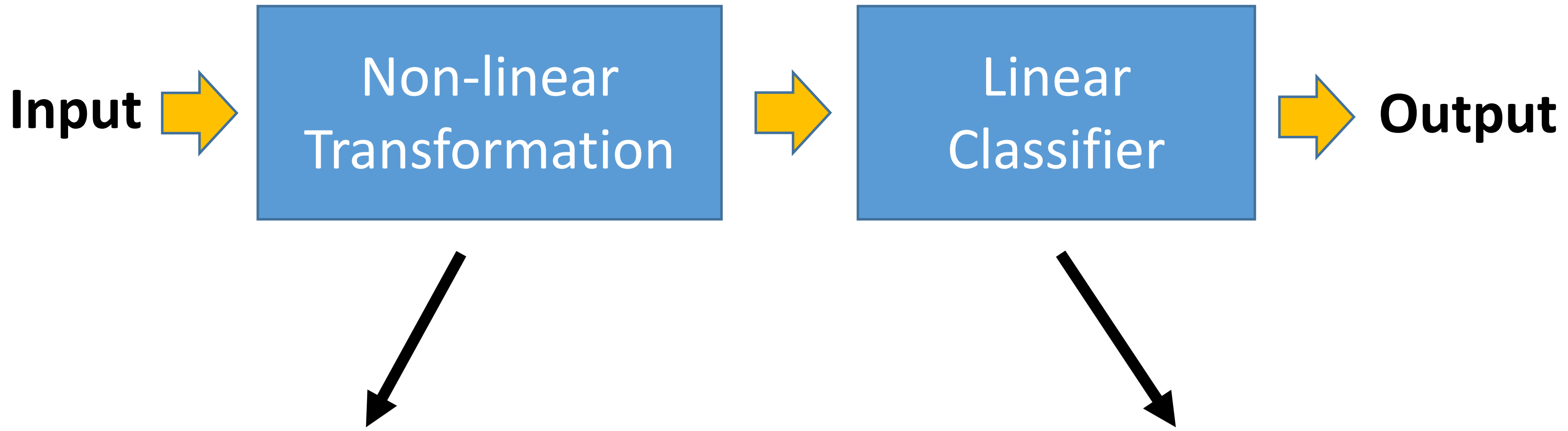
Max-Mahalanobis Training

Part I

(Max-Mahalanobis Linear Discriminant Analysis Networks, ICML 2018)

Motivation

- **Paradigm of feed-forward deep nets**



Active area of research
(AlexNet; VGG nets; ResNets; GoogleNets; DenseNets;)

Much less active
(Softmax regression)

Inspiration one: LDA is more efficient than LR

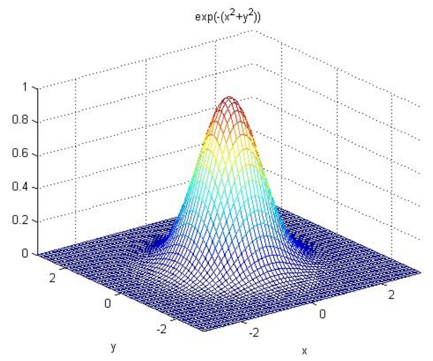
- Efron et al.(1975) show that *if the input distributes as a mixture of Gaussian*, then linear discriminant analysis (LDA) is **more efficient** than logistic regression (LR).

LDA needs less training data than LR to obtain certain error rate

- However, in practice data points hardly distributes as a mixture of Gaussian in the input space.

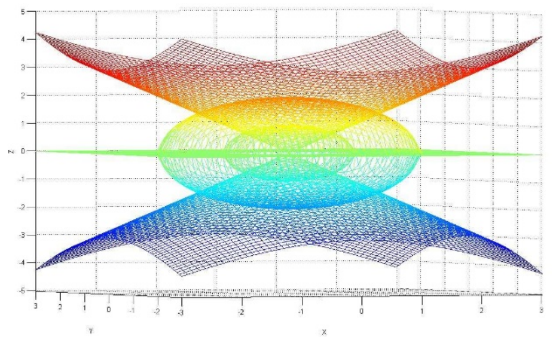
Inspiration two: neural networks are powerful

- Deep generative models (e.g., GANs) are successful.



Simple Distribution
(Gaussian/Mixture of Gaussian)

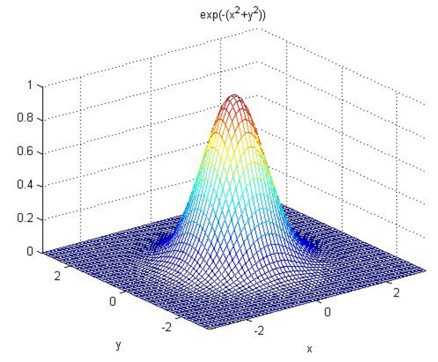
$$\xrightarrow[\textbf{DNN}]{\textbf{Deep generative models}}$$



Complex Distribution
(Data distribution)

Inspiration two: neural networks are powerful

- Deep generative models (e.g., GANs) are successful.
- The reverse direction should also be feasible.



Simple Distribution
(Gaussian/Mixture of Gaussian)

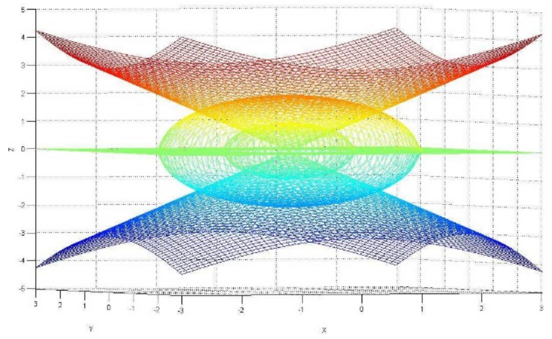
Deep generative models

$$\xrightarrow{\hspace{4cm}}$$

DNN

$$\xleftarrow{\hspace{4cm}}$$

Our Method
(MM-LDA networks)



Complex Distribution
(Data distribution)

Our method

- Models the feature distribution in DNNs as a mixture of Gaussian.
- Applies LDA on the feature to make predictions.

How to treat the Gaussian parameters?

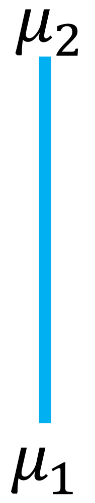
- Wan et al. (CVPR 2018) also model the feature distribution as a mixture of Gaussian. However, they treat the Gaussian parameters (μ_i and Σ) as extra trainable variables.

- We treat them as hyperparameters calculated by our algorithm, which can **provide theoretical guarantee on the robustness.**

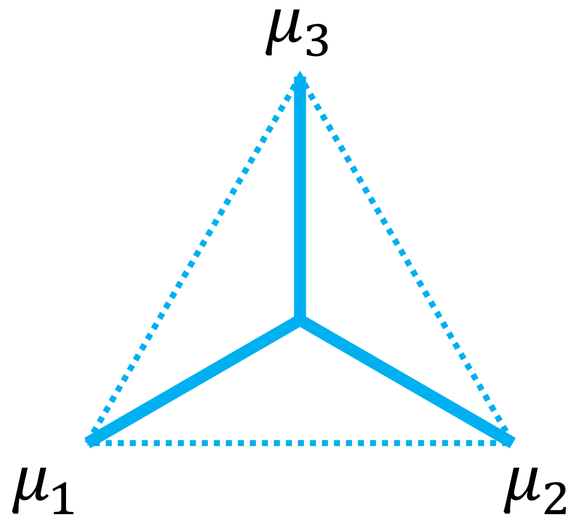
- The induced mixture of Gaussian model is named **Max Mahalanobis Distribution (MMD).**

Max-Mahalanobis Distribution (MMD)

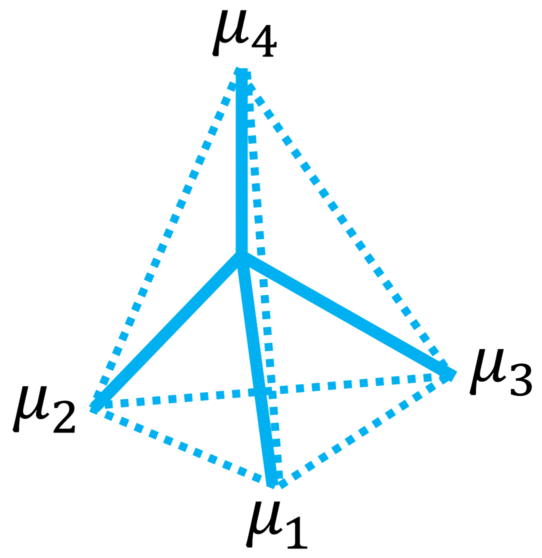
- **Making the minimal Mahalanobis distance between two Gaussian components maximal.**



$L = 2$
Straight Line



$L = 3$
Equilateral Triangle



$L = 4$
Regular Tetrahedron

Robustness w.r.t Gaussian parameters

Theorem 1. The expectation of the distance $\mathbb{E}(d_{i,j})$ is a function of the Mahalanobis distance $\Delta_{i,j}$ as

$$\mathbb{E}(d_{i,j}) = \sqrt{\frac{2}{\pi}} \exp\left(-\frac{\Delta_{i,j}^2}{8}\right) + \frac{1}{2}\Delta_{i,j}\left[1 - 2\Phi(-\frac{\Delta_{i,j}}{2})\right]$$

where $\Phi\,(\cdot)$ is the normal cumulative distribution function.

$$\mathbf{RB} \approx \overline{\mathbf{RB}} = \frac{\mathbf{1}}{\mathbf{2}} \min_{i,j \in [L]} \Delta_{i,j},$$

Distributing as a MMD can maximize $\overline{\mathrm{RB}}$.

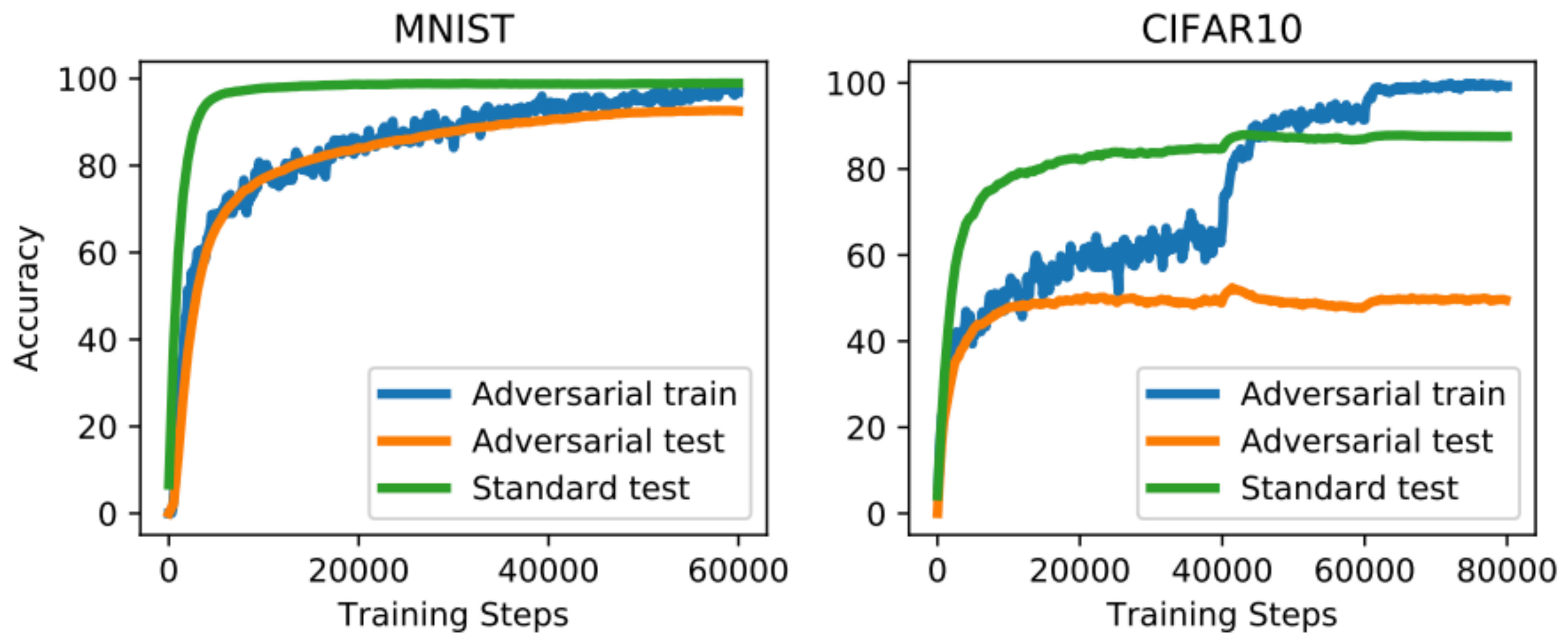
Can we further improve MMLDA?

Max-Mahalanobis Training

Part II

(Rethinking Softmax Cross-Entropy Loss for Adversarial Robustness, ICLR 2020)

Motivation



The same dataset, e.g., CIFAR-10, which enables good standard accuracy may not suffice to train robust models.

(Schmidt et al. NeurIPS 2018)

Possible Solutions

- **Introducing extra labeled data**

(Hendrycks et al. ICML 2019)

- **Introducing extra unlabeled data**

(Alayrac et al. NeurIPS 2019; Carmon et al. NeurIPS 2019)

Possible Solutions

- **Introducing extra labeled data**

(Hendrycks et al. ICML 2019)

- **Introducing extra unlabeled data**

(Alayrac et al. NeurIPS 2019; Carmon et al. NeurIPS 2019)

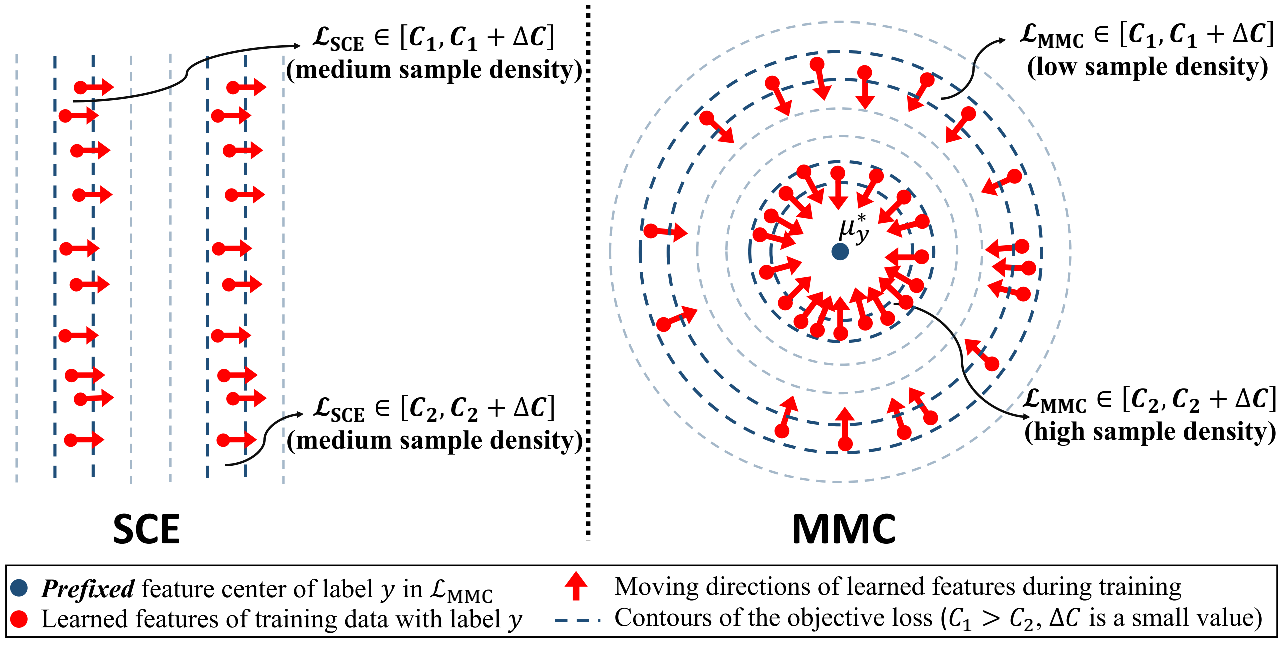
- **Our solution: Increase sample density to induce locally sufficient training data for robust learning**

Sample Density

Given a training dataset $\mathcal{D}$ with N input-label pairs, and the feature mapping Z trained by the objective $\mathcal{L}(Z(x), y)$ on this dataset, we define the sample density nearby the feature point $z = Z(x)$ following the similar definition in physics (Jackson, 1999) as

$$\mathbb{SD}(z) = \frac{\Delta N}{\text{Vol}(\Delta B)}. \quad (2)$$

Here $\text{Vol}(\cdot)$ denotes the volume of the input set, ΔB is a small neighbourhood containing the feature point z, and $\Delta N = |Z(\mathcal{D}) \cap \Delta B|$ is the number of training points in ΔB, where $Z(\mathcal{D})$ is the set of all mapped features for the inputs in $\mathcal{D}$. Note that the mapped feature z is still of the label y.


$\mathcal{L}_{\text{SCE}} \in [C_1, C_1 + \Delta C]$
(medium sample density)
$\mathcal{L}_{\text{SCE}} \in [C_2, C_2 + \Delta C]$
(medium sample density)
$\mathcal{L}_{\text{MMC}} \in [C_1, C_1 + \Delta C]$
(low sample density)
μ_y^*
$\mathcal{L}_{\text{MMC}} \in [C_2, C_2 + \Delta C]$
(high sample density)
SCE
MMC
● *Prefixed* feature center of label y in $\mathcal{L}_{\text{MMC}}$
● Learned features of training data with label y
Moving directions of learned features during training
Contours of the objective loss ($C_1 > C_2$, ΔC is a small value)

Generalized Softmax Cross Entropy Loss (g-SCE loss)

We define g-SCE loss as

$$\mathcal{L}_{\text{g-SCE}}(Z(x), y) = -1_y^\top \log\left[\text{softmax}(h)\right],$$ **Including MMLDA**

where $h_i = -(z-\mu_i)^\top \Sigma_i (z-\mu_i) + B_i$ **is the logits in quadratic form.**

We note that the SCE loss is included in the family of g-SCE loss as

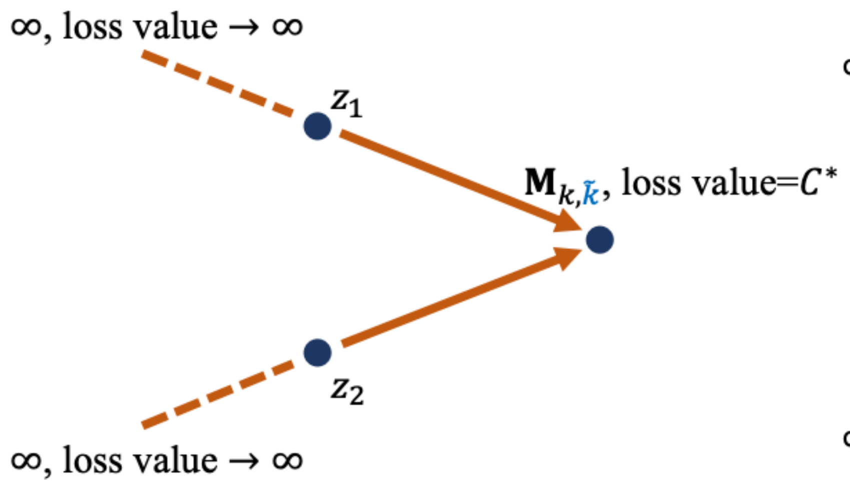
$$\text{softmax}(Wz+b)_i = \frac{\exp(W_i^\top z + b_i)}{\sum_{l\in[L]} \exp(W_l^\top z + b_l)} = \frac{\exp(-\|z - \frac{1}{2}W_i\|_2^2 + b_i + \frac{1}{4}\|W_i\|_2^2)}{\sum_{l\in[L]} \exp(-\|z - \frac{1}{2}W_l\|_2^2 + b_l + \frac{1}{4}\|W_l\|_2^2)}.$$

Induced Sample Density of g-SCE Loss

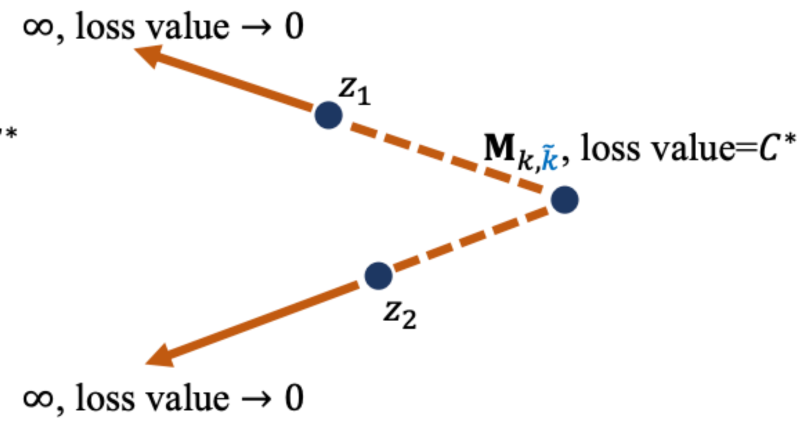
Theorem 1. *(Proof in Appendix A.1) Given* $(x, y) \in \mathcal{D}_{k,\tilde{k}}$, $z = Z(x)$ *and* $\mathcal{L}_{g\text{-}SCE}(z, y) = C$, *if there are* $\Sigma_k = \sigma_k I$, $\Sigma_{\tilde{k}} = \sigma_{\tilde{k}} I$, *and* $\sigma_k \neq \sigma_{\tilde{k}}$, *then the sample density nearby the feature point* z *based on the approximation in Eq. (6) is*

$$\mathbb{SD}(z) \propto \frac{N_{k,\tilde{k}} \cdot p_{k,\tilde{k}}(C)}{\left[\mathbf{B}_{k,\tilde{k}} + \frac{\log(C_e - 1)}{\sigma_k - \sigma_{\tilde{k}}}\right]^{\frac{d-1}{2}}}, \text{ and } \mathbf{B}_{k,\tilde{k}} = \frac{\sigma_k \sigma_{\tilde{k}} \|\mu_k - \mu_{\tilde{k}}\|_2^2}{(\sigma_k - \sigma_{\tilde{k}})^2} + \frac{B_k - B_{\tilde{k}}}{\sigma_k - \sigma_{\tilde{k}}}, \tag{7}$$

where for the input-label pair in $\mathcal{D}_{k,\tilde{k}}$, *there is* $\mathcal{L}_{g\text{-}SCE} \sim p_{k,\tilde{k}}(c)$.


∞, loss value $\to \infty$
z_1
$\mathbf{M}_{k,\tilde{k}}$, loss value=C^*
z_2
∞, loss value $\to \infty$

The case: $\sigma_k > \sigma_{\tilde{k}}$


∞, loss value $\to 0$
z_1
$\mathbf{M}_{k,\tilde{k}}$, loss value=C^*
z_2
∞, loss value $\to 0$

The case: $\sigma_k < \sigma_{\tilde{k}}$

(Preferred by models since lower loss values)

The ‘Curse’ of Softmax Function

$$\mathcal{L}_{\text{g-SCE}}(Z(x), y) = -1_y^\top \log\left[\underline{\mathrm{softmax}}(h)\right],$$

- **The softmax makes the loss value only depend on the relative relation among logits.**
- **This causes indirect and unexpected supervisory signals on the learned features.**

Our Method: Max-Mahalanobis Center (MMC) Loss

$$\mathcal{L}_{\text{MMLDA}}(Z(x), y) = -\log\left[\frac{\exp(-\frac{\|z-\mu_y^*\|_2^2}{2})}{\sum_{l\in[L]}\exp(-\frac{\|z-\mu_l^*\|_2^2}{2})}\right] = -\log\left[\frac{\exp(z^\top\mu_y^*)}{\sum_{l\in[L]}\exp(z^\top\mu_l^*)}\right]$$

$$\mathcal{L}_{\text{MMC}}(Z(x), y) = \frac{1}{2}\|z-\mu_y^*\|_2^2$$

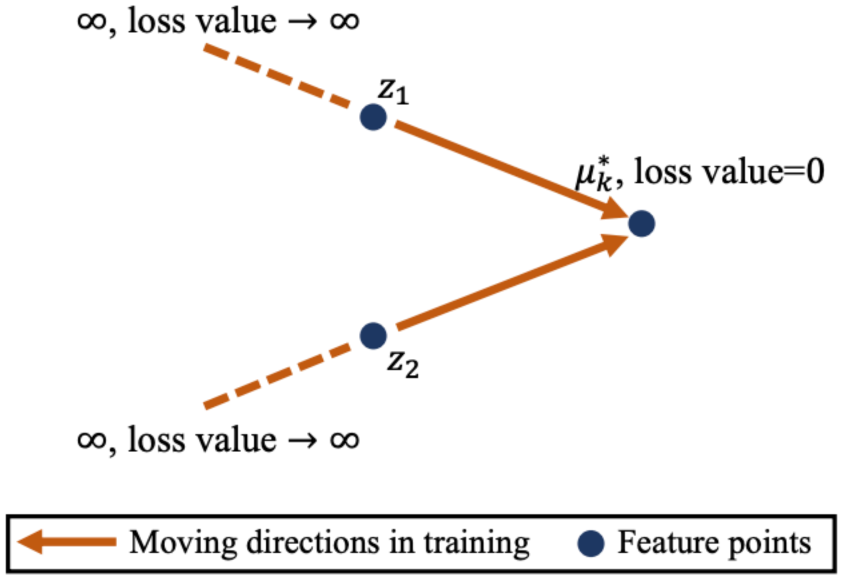
- **No softmax normalization**

Induced Sample Density of MMC Loss

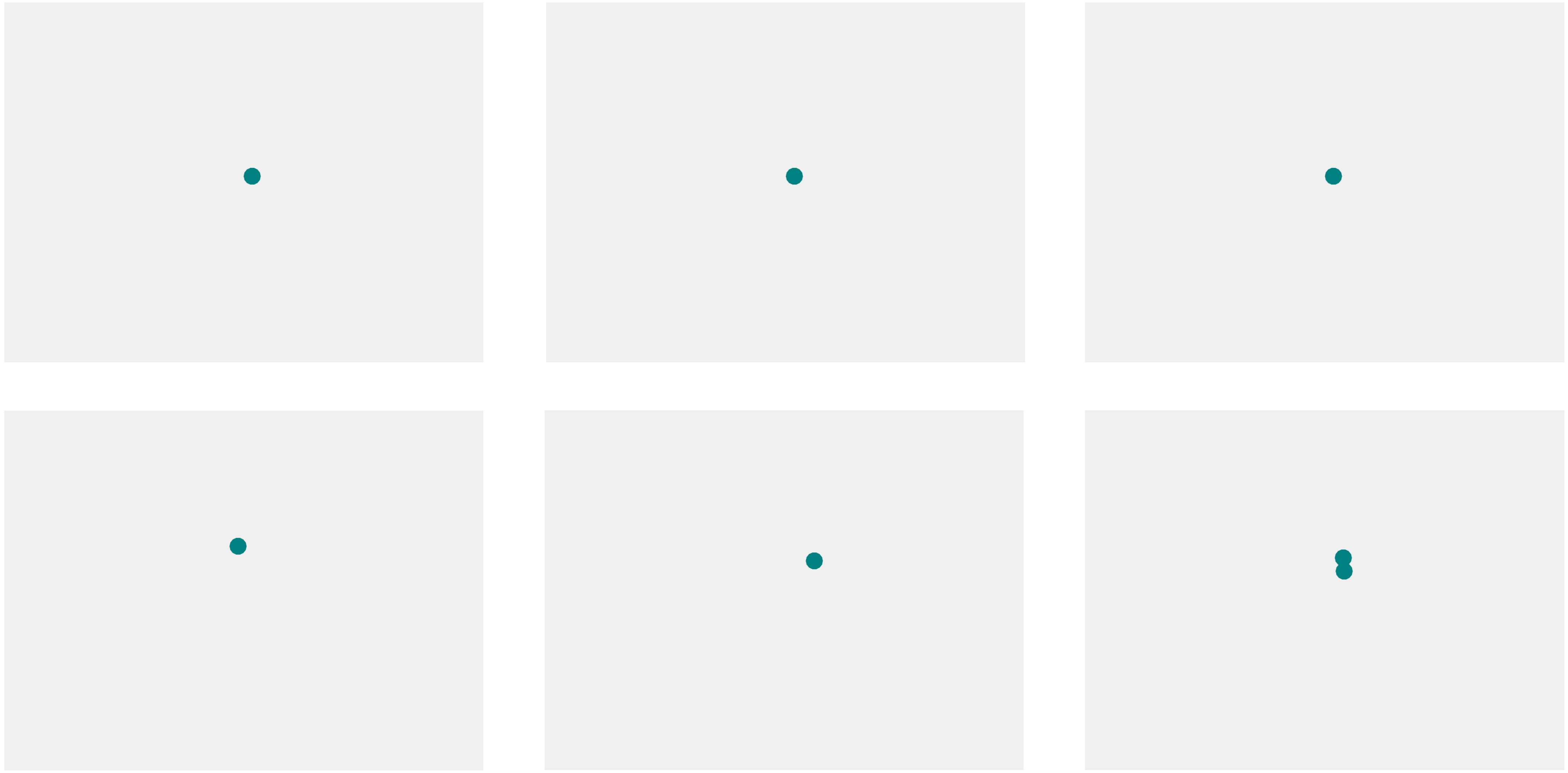
Theorem 2. *(Proof in Appendix A.2) Given* $(x, y) \in \mathcal{D}_k$, $z = Z(x)$ *and* $\mathcal{L}_{MMC}(z, y) = C$, *the sample density nearby the feature point* z *is*

$$\mathbb{SD}(z) \propto \frac{N_k \cdot p_k(C)}{C^{\frac{d-1}{2}}}, \tag{9}$$

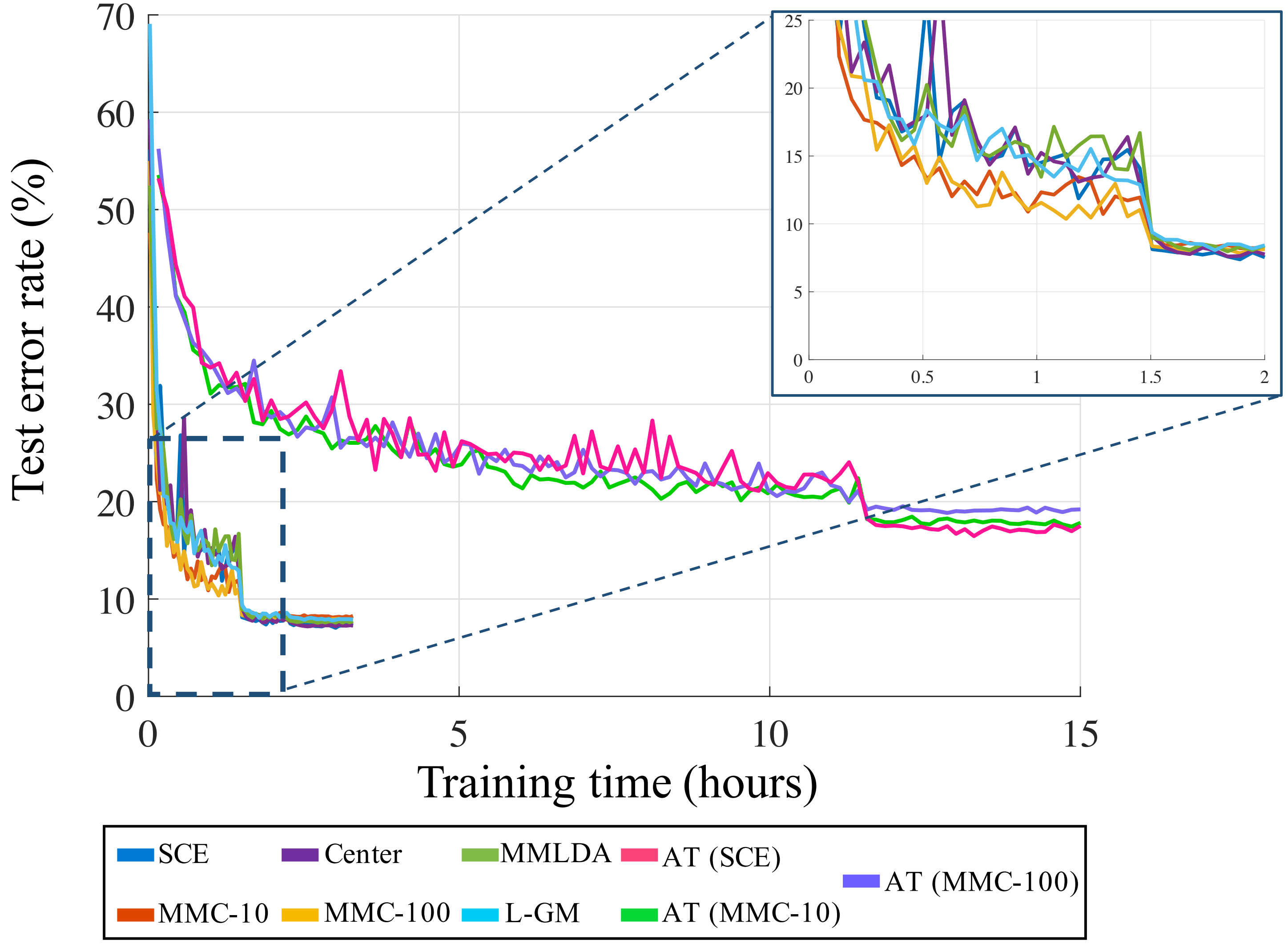
where for the input-label pair in $\mathcal{D}_k$, *there is* $\mathcal{L}_{MMC} \sim p_k(c)$.


∞, loss value $\to \infty$
z_1
μ_k^*, loss value=0
z_2
∞, loss value $\to \infty$
Moving directions in training
● Feature points

Toy Demo on Faster Convergence



Empirical Faster Convergence



White-box Robustness (Adaptive Attacks)

		Perturbation $\epsilon = 8/255$				**Perturbation** $\epsilon = 16/255$			
Methods	Clean	$\text{PGD}^{\textbf{tar}}_{10}$	$\text{PGD}^{\textbf{un}}_{10}$	$\text{PGD}^{\textbf{tar}}_{50}$	$\text{PGD}^{\textbf{un}}_{50}$	$\text{PGD}^{\textbf{tar}}_{10}$	$\text{PGD}^{\textbf{un}}_{10}$	$\text{PGD}^{\textbf{tar}}_{50}$	$\text{PGD}^{\textbf{un}}_{50}$
SCE	92.9	≤ 1	3.7	≤ 1	3.6	≤ 1	2.9	≤ 1	2.6
Center loss	92.8	≤ 1	4.4	≤ 1	4.3	≤ 1	3.1	≤ 1	2.9
MMLDA	92.4	≤ 1	16.5	≤ 1	9.7	≤ 1	6.7	≤ 1	5.5
L-GM	92.5	37.6	19.8	8.9	4.9	26.0	11.0	2.5	2.8
MMC-10 (rand)	92.3	43.5	29.2	20.9	18.4	31.3	17.9	8.6	11.6
MMC-10	92.7	**48.7**	**36.0**	**26.6**	**24.8**	**36.1**	**25.2**	**13.4**	**17.5**
$\text{AT}^{\textbf{tar}}_{10}$ (SCE)	83.7	**70.6**	49.7	**69.8**	47.8	48.4	26.7	31.2	16.0
$\text{AT}^{\textbf{tar}}_{10}$ (**MMC-10**)	83.0	69.2	**54.8**	67.0	**53.5**	**58.6**	**47.3**	**44.7**	**45.1**
$\text{AT}^{\textbf{un}}_{10}$ (SCE)	80.9	69.8	55.4	69.4	53.9	53.3	34.1	38.5	21.5
$\text{AT}^{\textbf{un}}_{10}$ (**MMC-10**)	81.8	**70.8**	**56.3**	**70.1**	**55.0**	**54.7**	**37.4**	**39.9**	**27.7**

CIFAR-10

Towards Robust Detection of Adversarial Examples

(Towards Robust Detection of Adversarial Examples, NeurIPS 2018)

We Detect Adversarial Examples, and How?

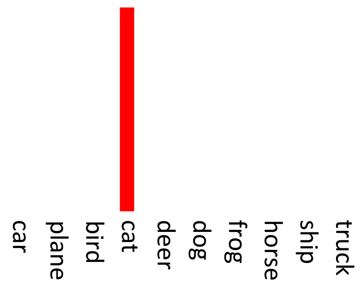
Design new detectors:

- Kernel density detector (Feinman et al. 2017)
- LID detector (Ma et al. ICLR 2018)
- ……

Train the models to better collaborate with existing detectors

Reverse Cross Entropy

Cross-Entropy (CE):

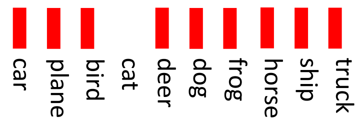


1_y: One-hot label

{0, 0, 0, **1**, 0, 0, 0, 0, 0, 0}

$$\mathcal{L}_{CE} = -\mathbf{1}_y \cdot \log(\mathrm{F})$$

Reverse Cross-Entropy (RCE):



R_y: Reverse label

$\{\frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \mathbf{0}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}\}$

$$\mathcal{L}_{RCE} = -\boldsymbol{R}_y \cdot \log(\mathrm{F})$$

The RCE Training Method

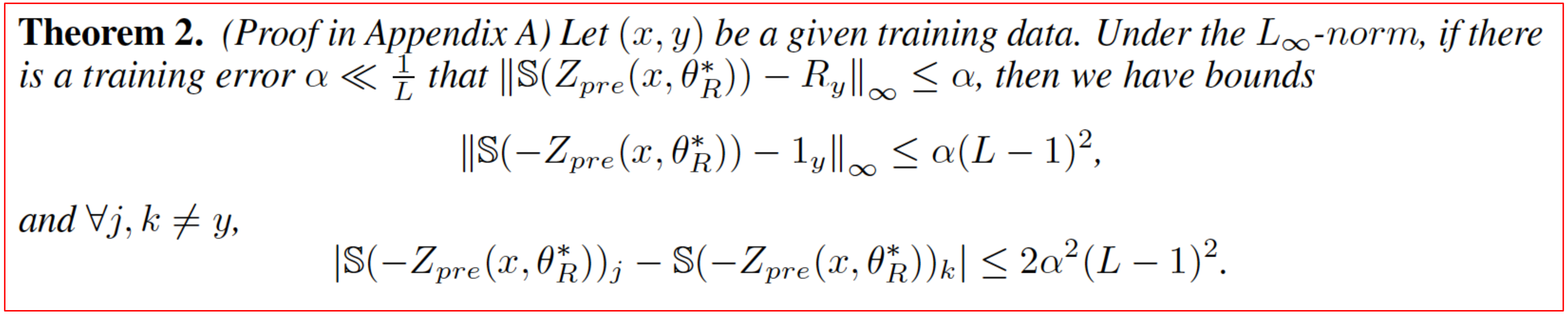
Phase 1: Reverse Training

Training the model by minimizing the RCE loss

Phase 2: Reverse Logits

Negating the logits fed to the softmax layer to give predictions

Theoretical Analysis



Theorem 2. *(Proof in Appendix A) Let* (x, y) *be a given training data. Under the* L_∞*-norm, if there is a training error* $\alpha \ll \frac{1}{L}$ *that* $\|\mathbb{S}(Z_{pre}(x, \theta_R^*)) - R_y\|_\infty \leq \alpha$*, then we have bounds*

$$\|\mathbb{S}(-Z_{pre}(x, \theta_R^*)) - 1_y\|_\infty \leq \alpha(L-1)^2,$$

and $\forall j, k \neq y$,

$$|\mathbb{S}(-Z_{pre}(x, \theta_R^*))_j - \mathbb{S}(-Z_{pre}(x, \theta_R^*))_k| \leq 2\alpha^2(L-1)^2.$$

Property 1: Consistent and Unbiased

When the training error $\alpha \longrightarrow 0$, the prediction tends to the one-hot label

Property 2: Tighter Bound

The difference between any two non-maximal elements decreases as $O(\alpha^2)$

The Insights of RCE Training

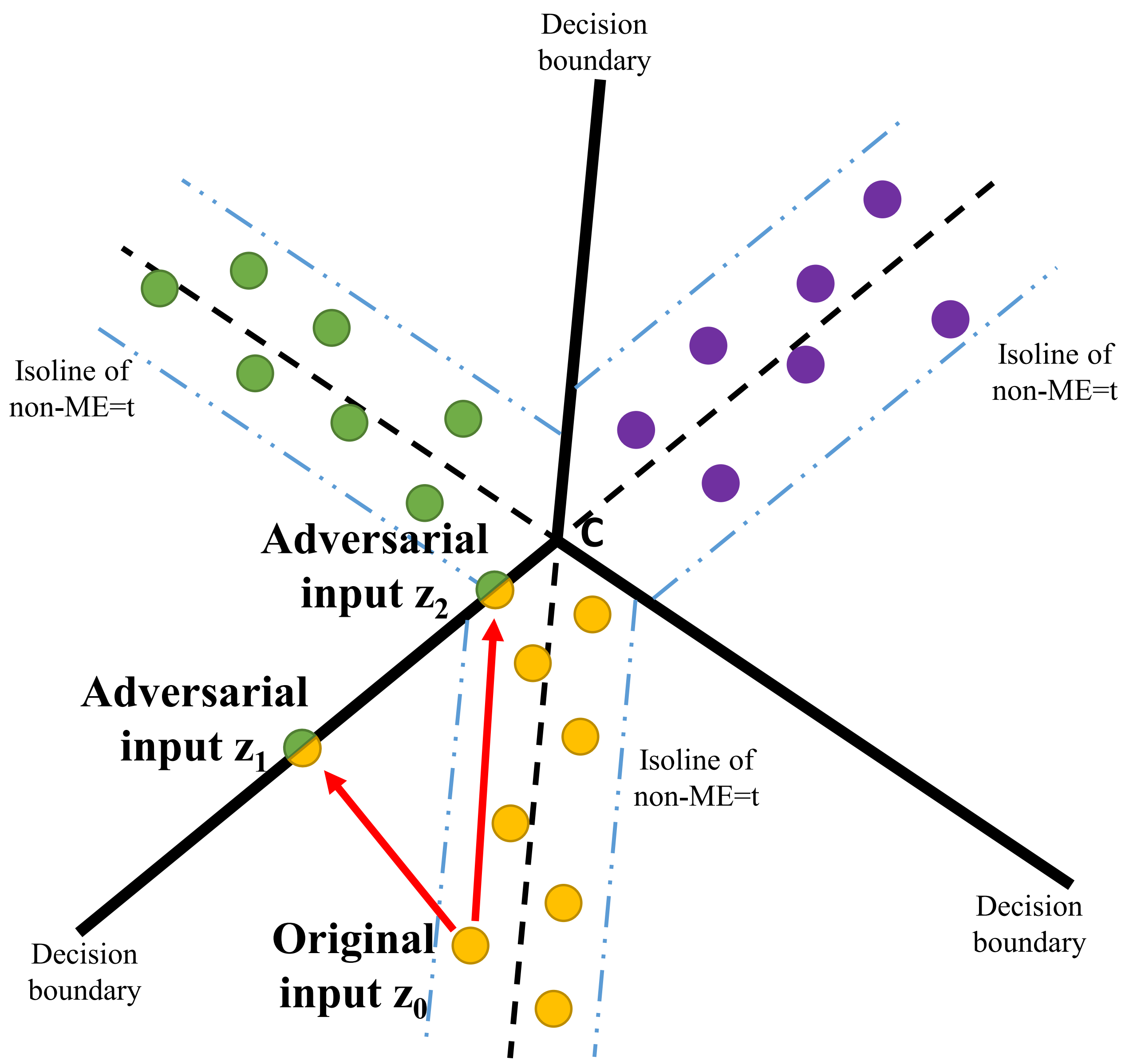
We first define the non-maximal entropy (non-ME) as:

$$\mathrm{nonME(x)} = -\sum_{i \neq y} \hat{F}(x)_i \log\left(\hat{F}(x)_i\right),$$

where $\hat{F}(x)_i$ is the normalized non-maximal predictions.

RCE training encourages the maximal prediction to tend to 1, while maximizing the non-ME.

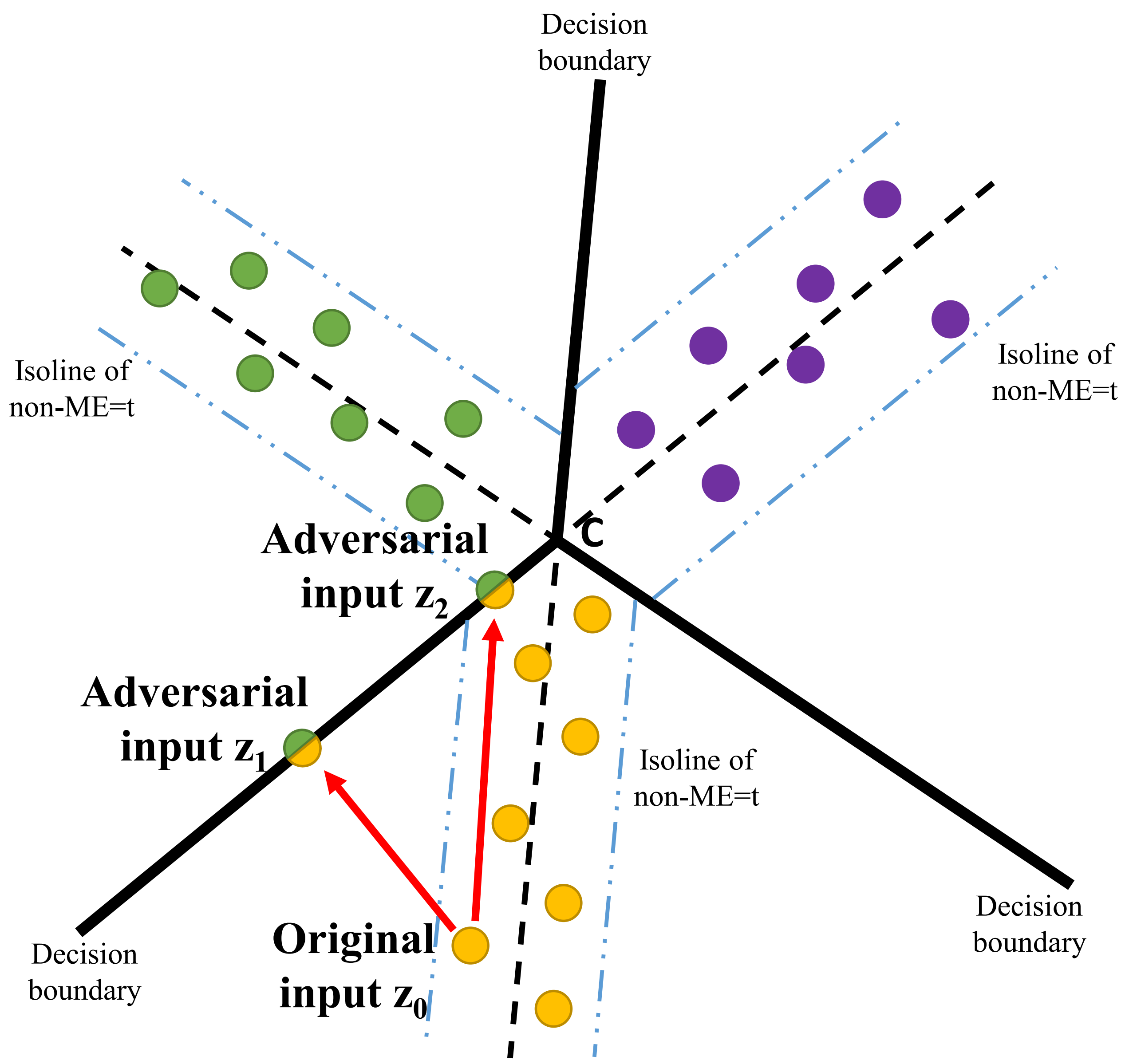
The Insights of RCE Training



The left plot is the decision domain in 2-d feature space for 3 classes (each class with one color)

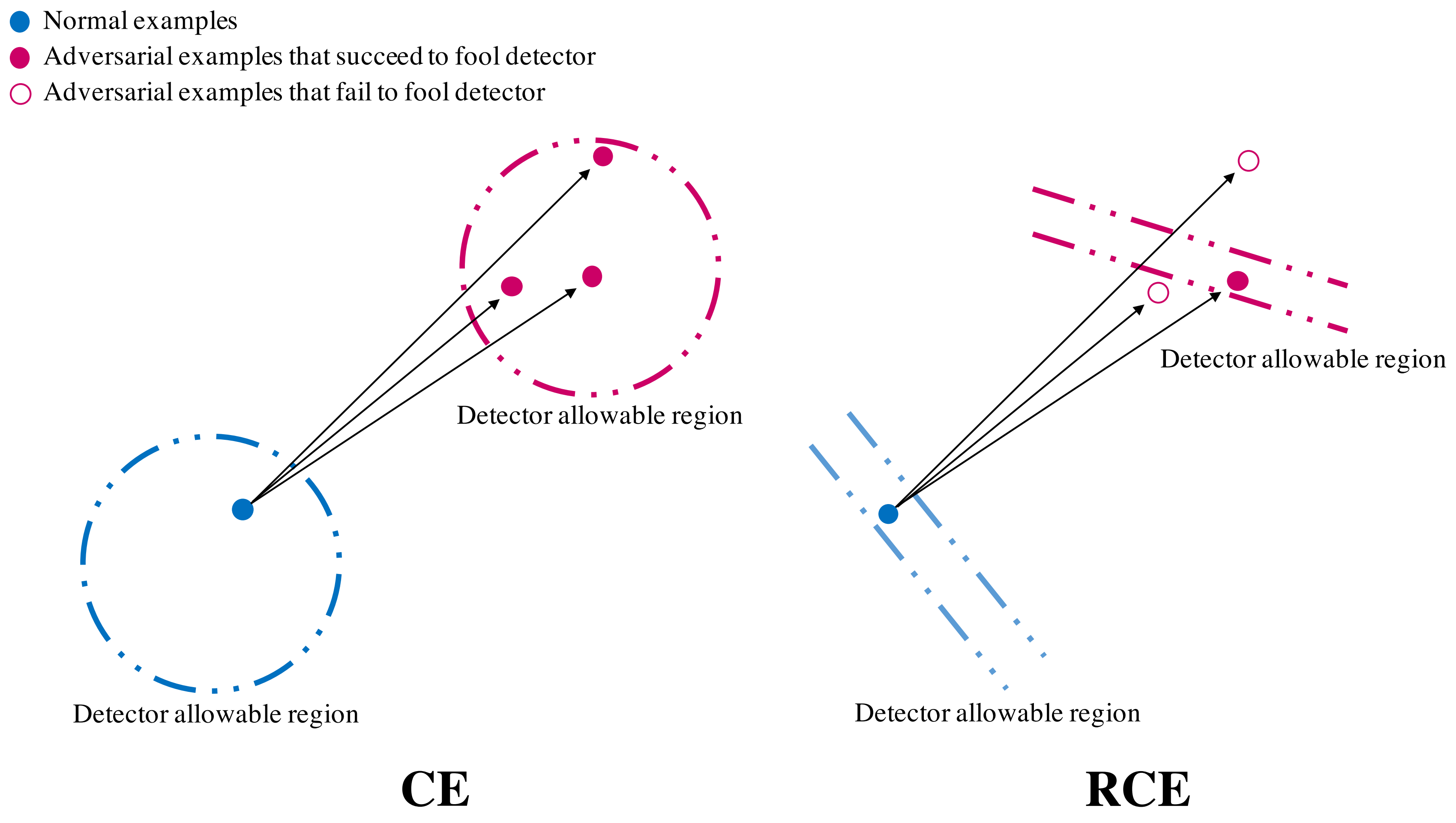
When the non-ME of the returned predictions are maximized, the learned features for each class with tend to locate near the black dash lines, where the points on the dash lines have the maximal non-ME.

The Insights of RCE Training



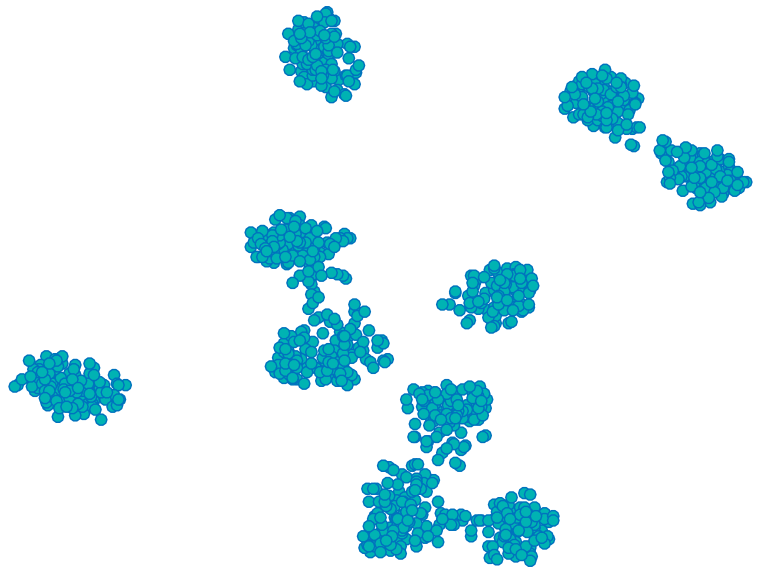
Then if an adversary want to craft an adversarial example based on z_0, he has to move further to z_2 rather than z_1 to obtain a normal value of non-ME.

The Insights of RCE Training

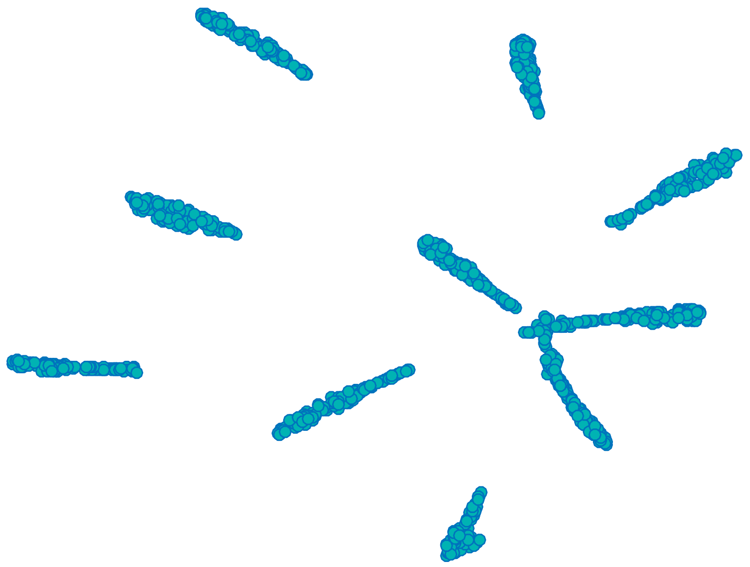

● Normal examples
● Adversarial examples that succeed to fool detector
○ Adversarial examples that fail to fool detector
Detector allowable region
Detector allowable region
CE
Detector allowable region
Detector allowable region
RCE

In practice, the learned low-dimensional feature distributions by RCE make it more difficult to craft an adversarial examples with normal values of non-ME.

Experiments



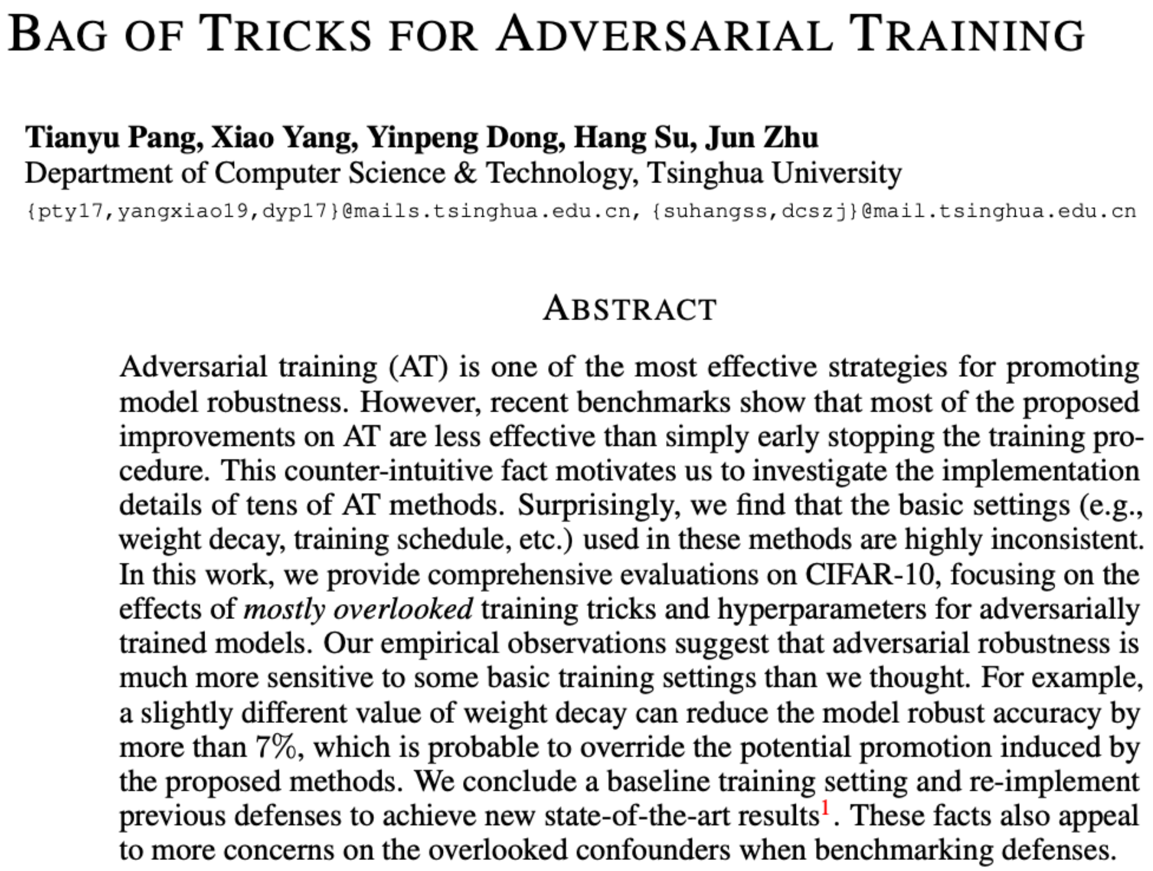
CE



RCE

t-SNE visualization of learned features on CIFAR-10

Our New Work --- Bag of Tricks for Adversarial Training



BAG OF TRICKS FOR ADVERSARIAL TRAINING

Tianyu Pang, Xiao Yang, Yinpeng Dong, Hang Su, Jun Zhu
Department of Computer Science & Technology, Tsinghua University
`{pty17,yangxiao19,dyp17}@mails.tsinghua.edu.cn, {suhangss,dcszj}@mail.tsinghua.edu.cn`

ABSTRACT

Adversarial training (AT) is one of the most effective strategies for promoting model robustness. However, recent benchmarks show that most of the proposed improvements on AT are less effective than simply early stopping the training procedure. This counter-intuitive fact motivates us to investigate the implementation details of tens of AT methods. Surprisingly, we find that the basic settings (e.g., weight decay, training schedule, etc.) used in these methods are highly inconsistent. In this work, we provide comprehensive evaluations on CIFAR-10, focusing on the effects of *mostly overlooked* training tricks and hyperparameters for adversarially trained models. Our empirical observations suggest that adversarial robustness is much more sensitive to some basic training settings than we thought. For example, a slightly different value of weight decay can reduce the model robust accuracy by more than 7%, which is probable to override the potential promotion induced by the proposed methods. We conclude a baseline training setting and re-implement previous defenses to achieve new state-of-the-art results[1]. These facts also appeal to more concerns on the overlooked confounders when benchmarking defenses.

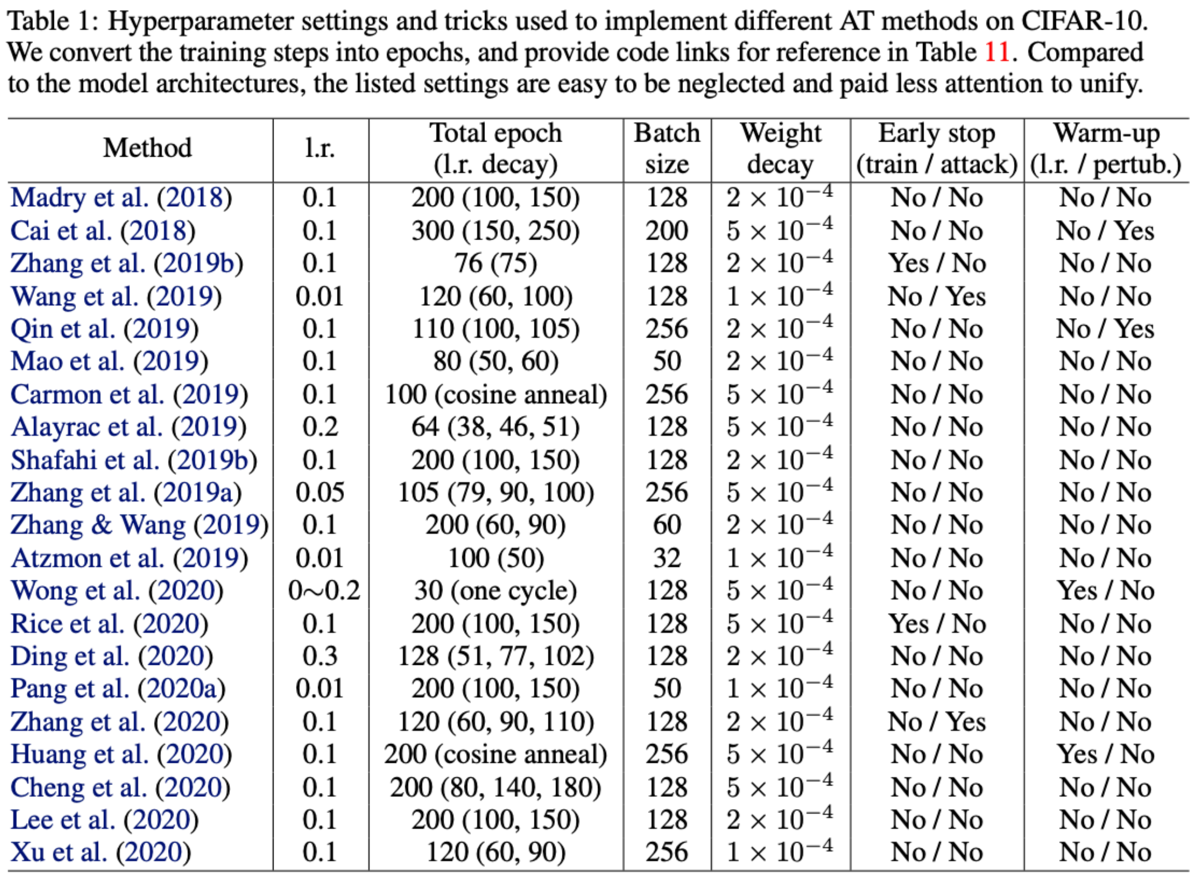
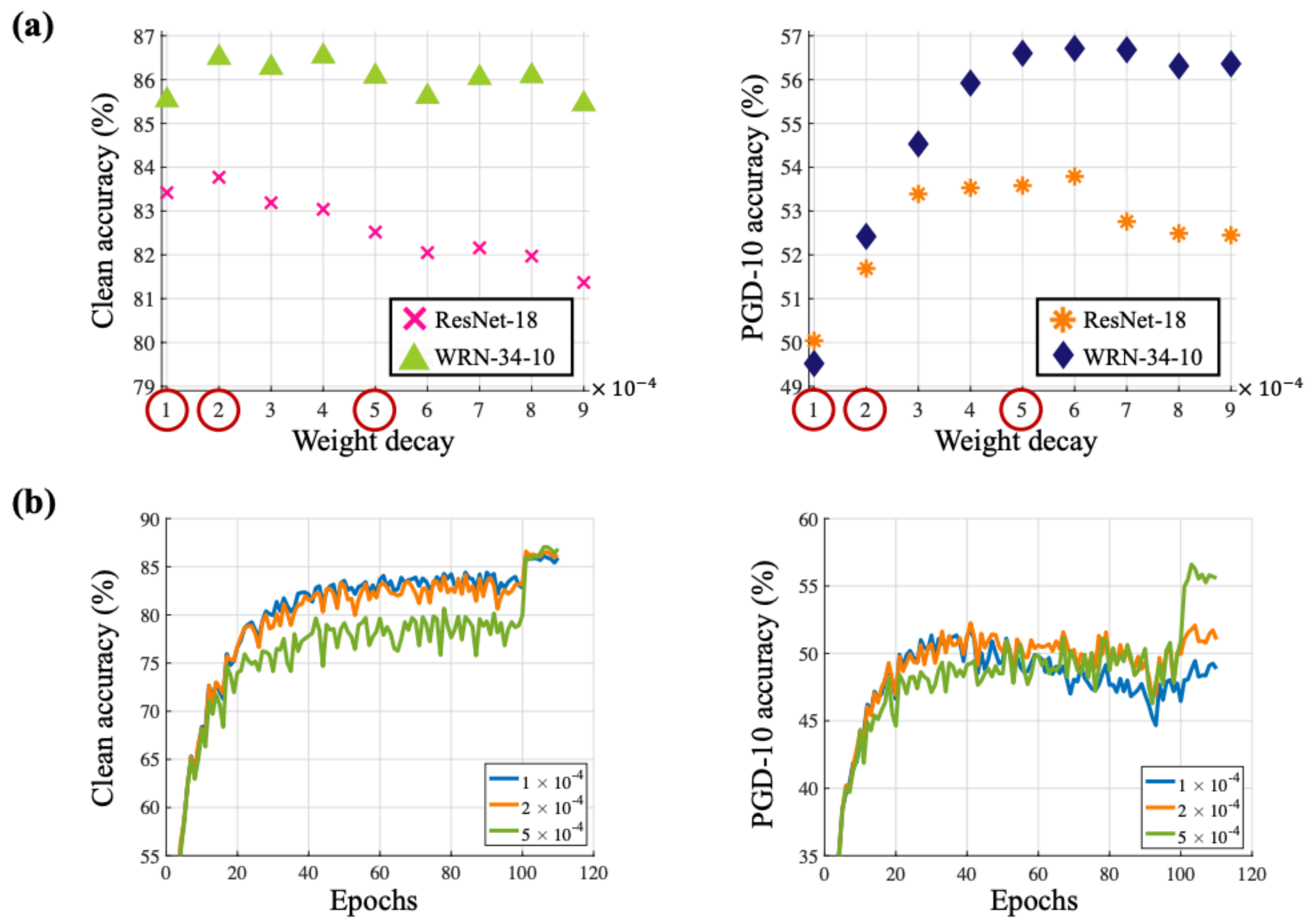


Table 1: Hyperparameter settings and tricks used to implement different AT methods on CIFAR-10. We convert the training steps into epochs, and provide code links for reference in Table 11. Compared to the model architectures, the listed settings are easy to be neglected and paid less attention to unify.

Method	l.r.	Total epoch (l.r. decay)	Batch size	Weight decay	Early stop (train / attack)	Warm-up (l.r. / pertub.)
Madry et al. (2018)	0.1	200 (100, 150)	128	2×10^{-4}	No / No	No / No
Cai et al. (2018)	0.1	300 (150, 250)	200	5×10^{-4}	No / No	No / Yes
Zhang et al. (2019b)	0.1	76 (75)	128	2×10^{-4}	Yes / No	No / No
Wang et al. (2019)	0.01	120 (60, 100)	128	1×10^{-4}	No / Yes	No / No
Qin et al. (2019)	0.1	110 (100, 105)	256	2×10^{-4}	No / No	No / Yes
Mao et al. (2019)	0.1	80 (50, 60)	50	2×10^{-4}	No / No	No / No
Carmon et al. (2019)	0.1	100 (cosine anneal)	256	5×10^{-4}	No / No	No / No
Alayrac et al. (2019)	0.2	64 (38, 46, 51)	128	5×10^{-4}	No / No	No / No
Shafahi et al. (2019b)	0.1	200 (100, 150)	128	2×10^{-4}	No / No	No / No
Zhang et al. (2019a)	0.05	105 (79, 90, 100)	256	5×10^{-4}	No / No	No / No
Zhang & Wang (2019)	0.1	200 (60, 90)	60	2×10^{-4}	No / No	No / No
Atzmon et al. (2019)	0.01	100 (50)	32	1×10^{-4}	No / No	No / No
Wong et al. (2020)	0∼0.2	30 (one cycle)	128	5×10^{-4}	No / No	Yes / No
Rice et al. (2020)	0.1	200 (100, 150)	128	5×10^{-4}	Yes / No	No / No
Ding et al. (2020)	0.3	128 (51, 77, 102)	128	2×10^{-4}	No / No	No / No
Pang et al. (2020a)	0.01	200 (100, 150)	50	1×10^{-4}	No / No	No / No
Zhang et al. (2020)	0.1	120 (60, 90, 110)	128	2×10^{-4}	No / Yes	No / No
Huang et al. (2020)	0.1	200 (cosine anneal)	256	5×10^{-4}	No / No	Yes / No
Cheng et al. (2020)	0.1	200 (80, 140, 180)	128	5×10^{-4}	No / No	No / No
Lee et al. (2020)	0.1	200 (100, 150)	128	2×10^{-4}	No / No	No / No
Xu et al. (2020)	0.1	120 (60, 90)	256	1×10^{-4}	No / No	No / No

Our New Work --- Bag of Tricks for Adversarial Training



Our New Work --- Bag of Tricks for Adversarial Training

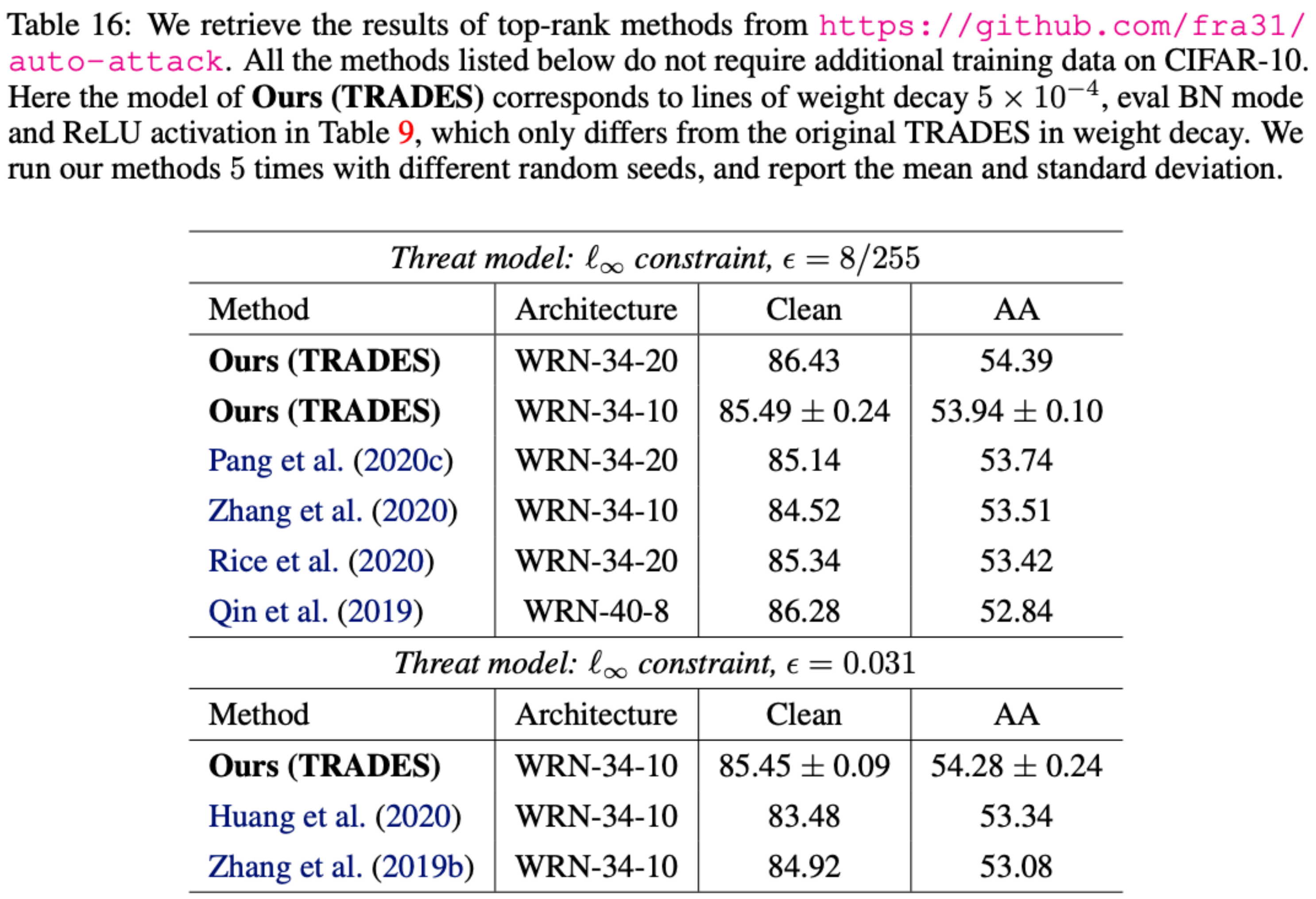
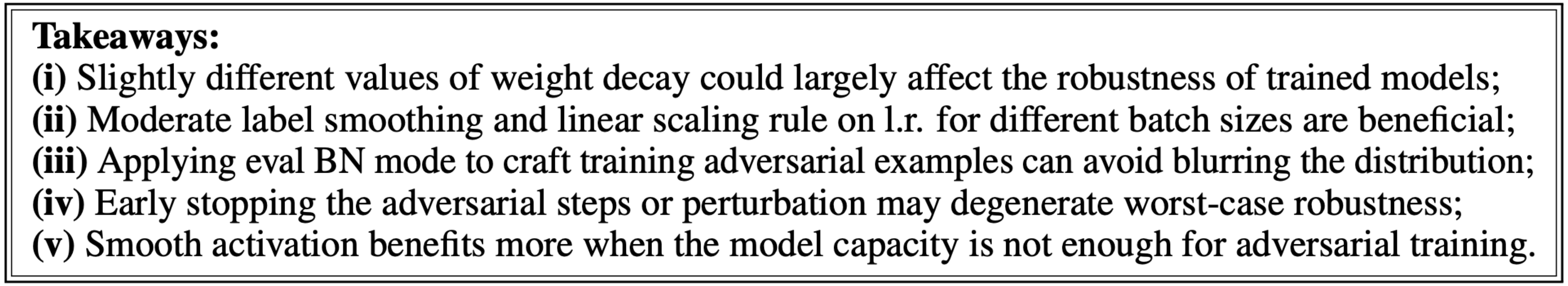


Table 16: We retrieve the results of top-rank methods from https://github.com/fra31/auto-attack. All the methods listed below do not require additional training data on CIFAR-10. Here the model of **Ours (TRADES)** corresponds to lines of weight decay 5×10^{-4}, eval BN mode and ReLU activation in Table 9, which only differs from the original TRADES in weight decay. We run our methods 5 times with different random seeds, and report the mean and standard deviation.

Threat model: ℓ_∞ constraint, $\epsilon = 8/255$			
Method	Architecture	Clean	AA
Ours (TRADES)	WRN-34-20	86.43	54.39
Ours (TRADES)	WRN-34-10	85.49 ± 0.24	53.94 ± 0.10
Pang et al. (2020c)	WRN-34-20	85.14	53.74
Zhang et al. (2020)	WRN-34-10	84.52	53.51
Rice et al. (2020)	WRN-34-20	85.34	53.42
Qin et al. (2019)	WRN-40-8	86.28	52.84
Threat model: ℓ_∞ constraint, $\epsilon = 0.031$			
Method	Architecture	Clean	AA
Ours (TRADES)	WRN-34-10	85.45 ± 0.09	54.28 ± 0.24
Huang et al. (2020)	WRN-34-10	83.48	53.34
Zhang et al. (2019b)	WRN-34-10	84.92	53.08

Our New Work --- Bag of Tricks for Adversarial Training



Takeaways:
(i) Slightly different values of weight decay could largely affect the robustness of trained models;
(ii) Moderate label smoothing and linear scaling rule on l.r. for different batch sizes are beneficial;
(iii) Applying eval BN mode to craft training adversarial examples can avoid blurring the distribution;
(iv) Early stopping the adversarial steps or perturbation may degenerate worst-case robustness;
(v) Smooth activation benefits more when the model capacity is not enough for adversarial training.

Paper: https://arxiv.org/pdf/2010.00467.pdf

Code: https://github.com/P2333/Bag-of-Tricks-for-AT

Thanks