Adversarially Robust Machine Learning

Tianyu Pang（庞天宇）

Department of Computer Science and Technology
Tsinghua University

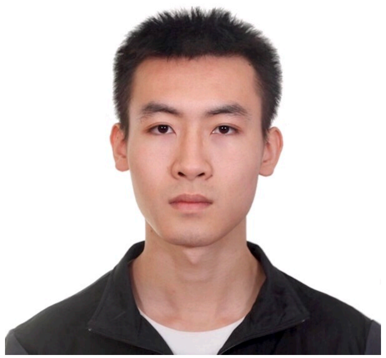
清華大學
Tsinghua University

Joint work with

Prof. Jun Zhu

Prof. Hang Su

Xiao Yang

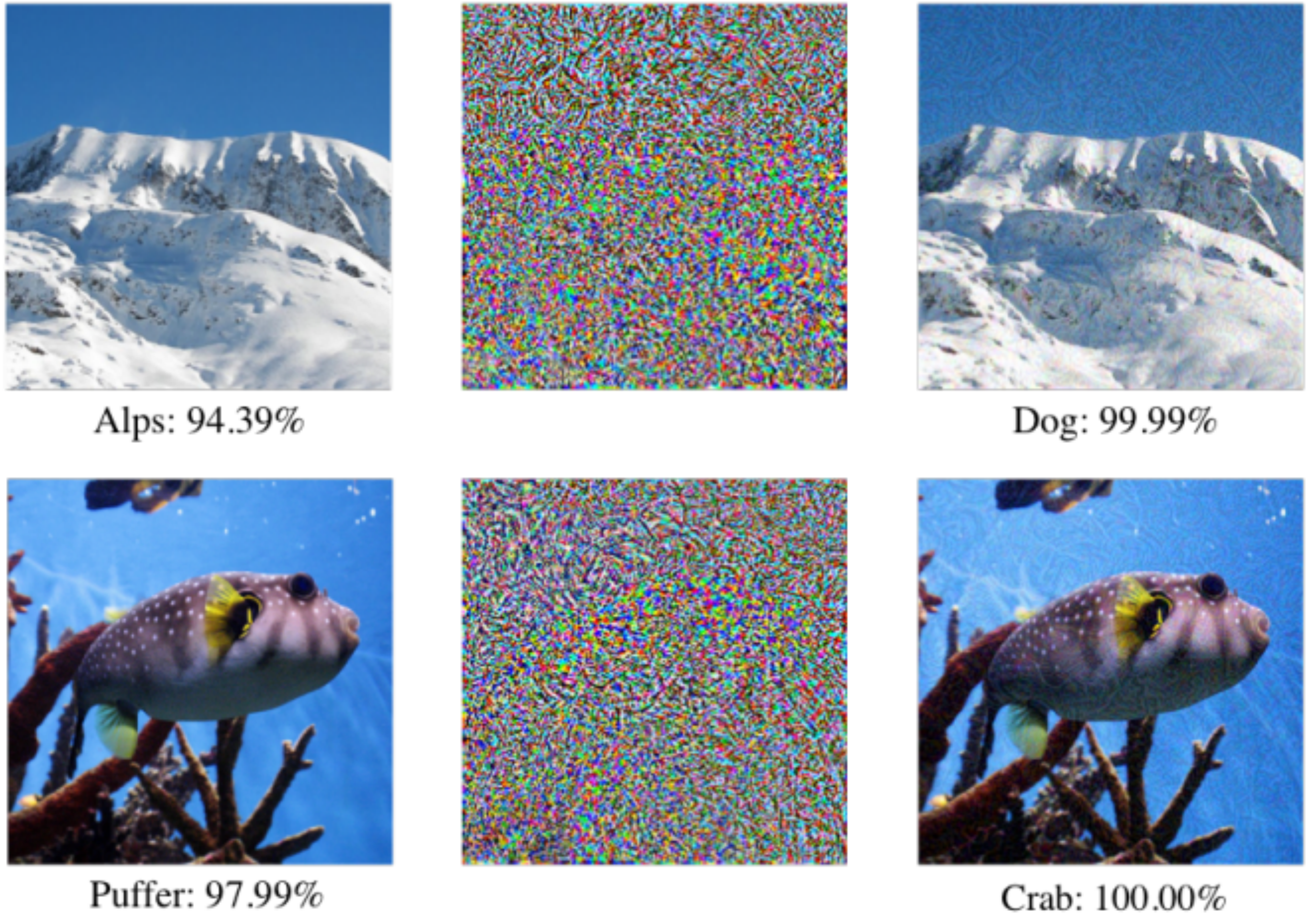


Yinpeng Dong

Kun Xu

Chao Du

Adversarial examples in computer vision

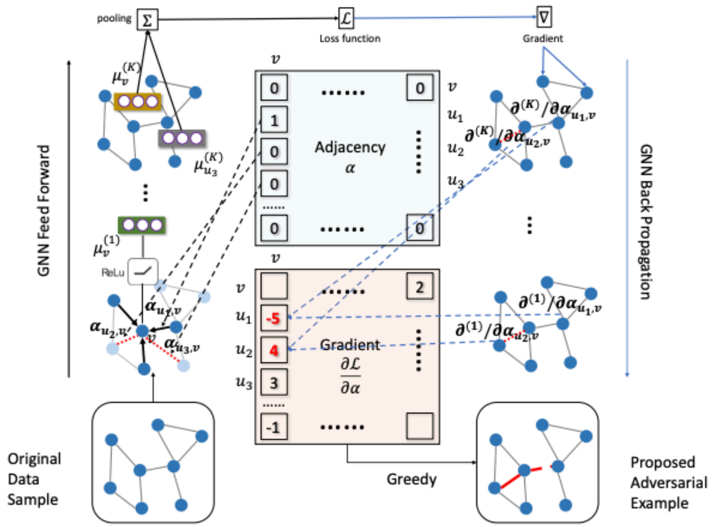


(Dong et al. CVPR 2018)

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)

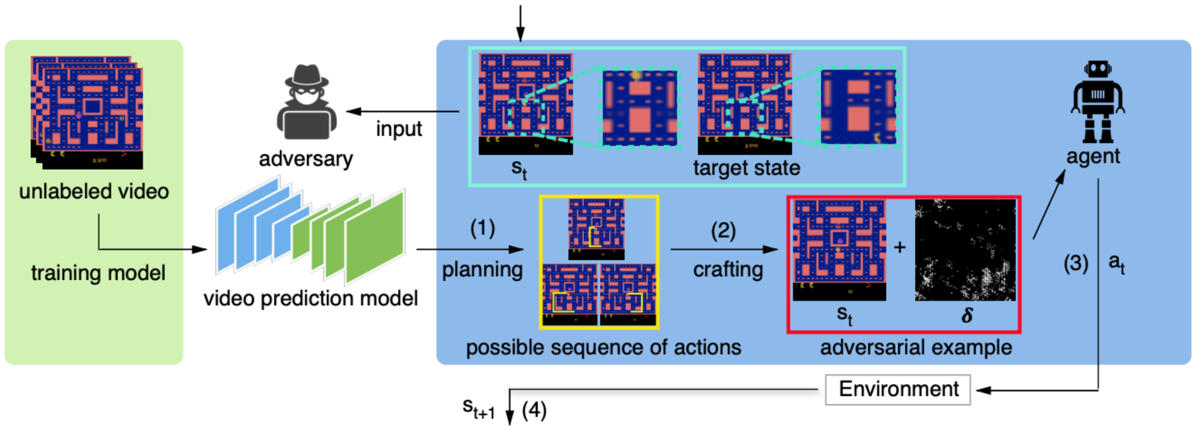


GNN model (Dai et al. ICML 2018)

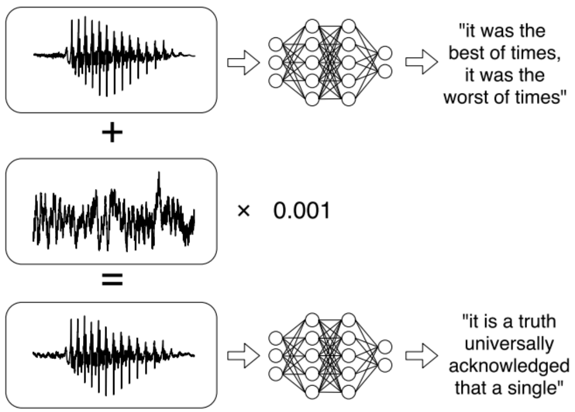
Recommend System,

LIDAR,

……

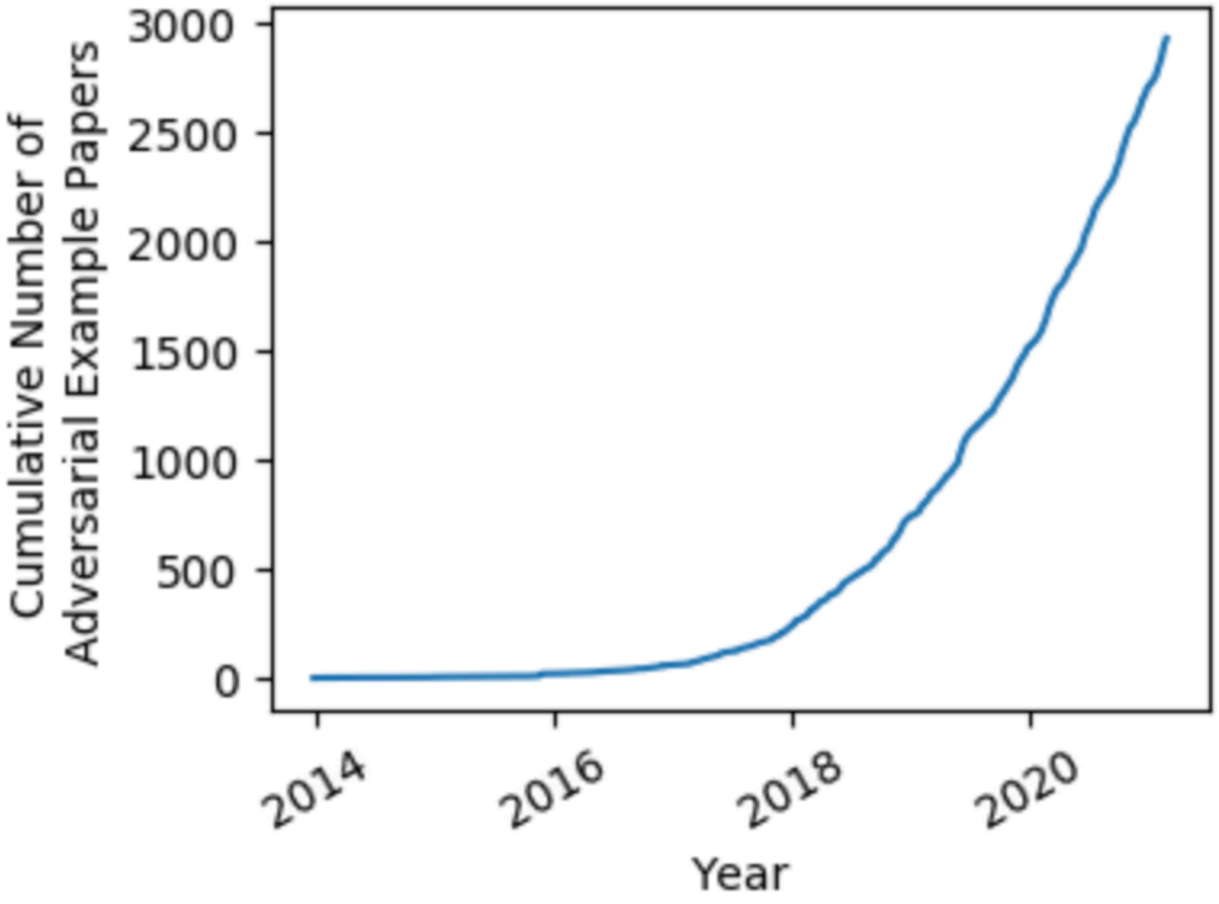


Reinforcement Learning (Lin et al. IJCAI 2017)



Audio (Carlini and Wagner. S&P 2018)

Becoming a popular research topic



Title involves

‘attack’: 961 papers
‘defense/defend’: 332 papers
‘adversarial training’: 141 papers
‘certify/certified…’: 71 papers
………

From https://nicholas.carlini.com/writing/2019/all-adversarial-example-papers

Bag of Tricks for Adversarial Training

Tianyu Pang, Xiao Yang, Yinpeng Dong, Hang Su, and Jun Zhu

ICLR 2021

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

Where we converged after so many efforts (w.r.t. defenses)?

Adversarial Training (+ blablabla)

- practically works well, able to defend adaptive attacks (to some extent)
- occupies top solutions in different adversarial competitions
- computation can be reduced by one-step adv (FastAT) or reuse compute. (FreeAT)
- recent work of positively applying AT on traditional tasks

Certified Defense

- provide quantitative bounds for certified robustness
- requires convex approximations
- promising but practically less effective than AT (fortunately they could be compatible)
- point-wise certification (is not that certified), can maliciously craft low-bound test sets

Milestones of adversarial training frameworks (2014-2019)

FGSM-AT

Seminal work of AT

(Goodfellow et al. 2014)

BIM-AT

Can defend multi-step attacks

(Kurakin et al. 2016)

PGD-AT

Propose min-max framework for AT

(Madry et al. 2018)

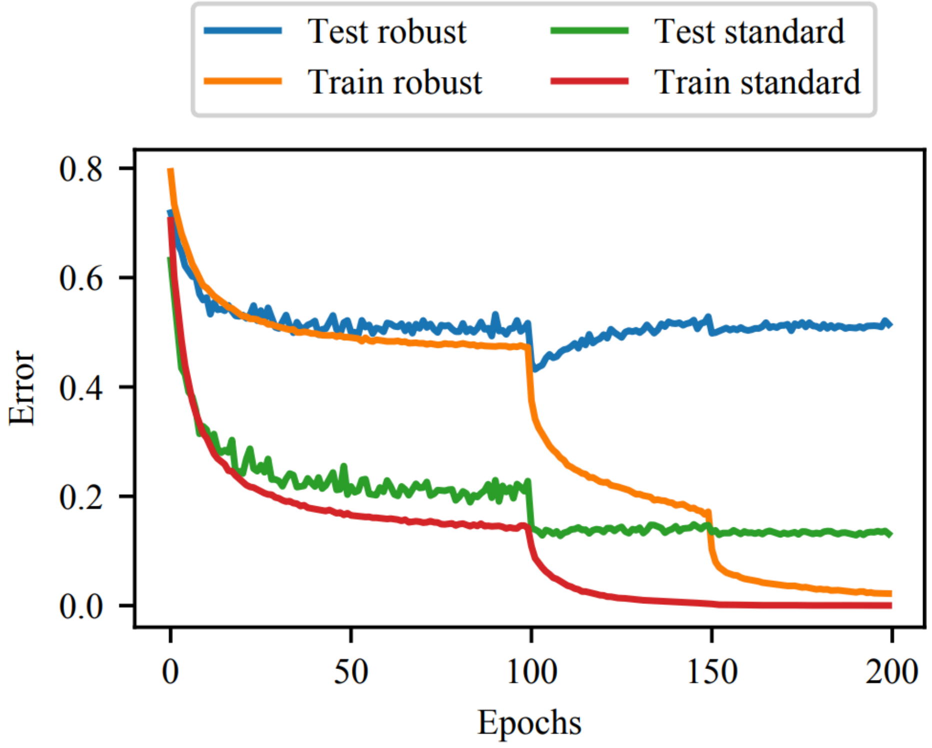
TRADES

Winner of NeurIPS 2019 Adversarial Competition

(Zhang et al. 2019)

What happened in 2020?

Rice et al. (ICML 2020) find that simply **early stopping** the training process of **PGD-AT** can attain the gains from almost all the previously proposed improvements, including the state-of-the-art **TRADES**.



(From Rice et al.)

- ***TRADES also applied early stopping by decaying learning rate at 75th epoch and used the checkpoint of 76th epoch.***

What happened in 2020?

Gowal et al. (2020) find that TRADES actually performs better than PGD-AT

Key takeaways. Contrary to the suggestion of Rice et al. (2020) (i.e., *"the original PGD-based adversarial training method can actually achieve the same robust performance as state-of-the-art method"*, see Sec. 2.1), TRADES (when combined with early-stopping – as our setup dictates) is more competitive than classical adversarial training. The results also highlight the importance of strong evaluations beyond PGD^{20} (including evaluations of the validation set used for early stopping).

(From Gowal et al.)

What happened in 2020?

Gowal et al. (2020) find that TRADES actually performs better than PGD-AT

> **Key takeaways.** Contrary to the suggestion of Rice et al. (2020) (i.e., *"the original PGD-based adversarial training method can actually achieve the same robust performance as state-of-the-art method"*, see Sec. 2.1), TRADES (when combined with early-stopping – as our setup dictates) is more competitive than classical adversarial training. The results also highlight the importance of strong evaluations beyond PGD^{20} (including evaluations of the validation set used for early stopping).

(From Gowal et al.)

Paradox???

Zhang et al. (2018): TRADES performs better than PGD-AT
Rice et al. (2020): PGD-AT performs better than TRADES
Gowal et al. (2020): TRADES performs better than PGD-AT

Who is wrong? Nobody

Zhang et al. (2018):
TRADES (weight decay 2×10^{-4})
PGD-AT (weight decay 2×10^{-4})

Rice et al. (2020):
TRADES (weight decay 2×10^{-4})
PGD-AT (weight decay 5×10^{-4})

Gowal et al. (2020):
TRADES (weight decay 5×10^{-4})
PGD-AT (weight decay 5×10^{-4})

Slightly different values of weight decay can lead to largely different conclusions in the adversarial setting!

Overlooked training settings could affect our evaluations on the defenses, especially in public benchmarks.

Training settings in previous work are highly inconsistent

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

Early stopping adversarial intensity

	Base	Early stopping attack iter.			Warmup on l.r.			Warmup on perturb.		
		40 / 70	40 / 100	60 / 100	10	15	20	10	15	20
Clean	82.52	86.52	86.56	85.67	82.45	82.64	82.31	82.64	82.75	82.78
PGD-10	53.58	52.65	53.22	52.90	53.43	53.29	53.35	53.65	53.27	53.62
AA	48.51	46.6	46.04	45.96	48.26	48.12	48.37	48.44	48.17	48.48

- Improved clean accuracy and faster training
- The performance under the stronger AutoAttack is degraded.

Warmup w.r.t. learning rate or perturbation

	Base	Early stopping attack iter.			Warmup on l.r.			Warmup on perturb.		
		40 / 70	40 / 100	60 / 100	10	15	20	10	15	20
Clean	82.52	86.52	86.56	85.67	82.45	82.64	82.31	82.64	82.75	82.78
PGD-10	53.58	52.65	53.22	52.90	53.43	53.29	53.35	53.65	53.27	53.62
AA	48.51	46.6	46.04	45.96	48.26	48.12	48.37	48.44	48.17	48.48

- **The effects of warmup are not significant**

Batch size

Table 3: Test accuracy (%) under different **batch size** and **learning rate** (l.r.) on CIFAR-10. The basic l.r. is 0.1, while the scaled l.r. is, e.g., 0.2 for batch size 256, and 0.05 for batch size 64.

ResNet-18				
Batch size	Basic l.r.		Scaled l.r.	
	Clean	PGD-10	Clean	PGD-10
64	80.08	51.31	82.44	52.48
128	82.52	**53.58**	-	-
256	83.33	52.20	82.24	52.52
512	83.40	50.69	82.16	53.36
WRN-34-10				
Batch size	Basic l.r.		Scaled l.r.	
	Clean	PGD-10	Clean	PGD-10
64	84.20	54.69	85.40	54.86
128	86.07	**56.60**	-	-
256	86.21	52.90	85.89	56.09
512	86.29	50.17	86.47	55.49

- **Larger batch size may not be better**
- **Linear scaling rule for learning rate is beneficial**

Mode for batch normalization when computing PGD

Table 7: Test accuracy (%) under different **BN modes** on CIFAR-10. We evaluate across several model architectures, since the BN layers have different positions in different models.

	BN mode	Model architecture: ResNet-18	SENet-18	DenseNet-121	GoogleNet	DPN26	WRN-34-10
Clean	train	82.52	82.20	85.38	83.97	83.67	86.07
	eval	83.48	84.11	86.33	85.26	84.56	87.38
	-	+0.96	+1.91	+0.95	+1.29	+0.89	+1.31
PGD-10	train	53.58	54.01	56.22	53.76	53.88	56.60
	eval	53.64	53.90	56.11	53.77	53.41	56.04
	-	+0.06	-0.11	-0.11	+0.01	-0.47	-0.56
AA	train	48.51	48.72	51.58	48.73	48.50	52.19
	eval	48.75	48.95	51.24	48.83	48.30	51.93
	-	+0.24	+0.23	-0.34	+0.10	-0.20	-0.26

- **Eval BN mode (used in TRADES) lead to higher clean accuracy while keeping similar robust accuracy, compared to train BN mode (used in PGD-AT)**

Label smoothing

Table 17: Test accuracy (%) under different **label smoothing** on CIFAR-10. The model is ResNet-18 trained by PGD-AT. We evaluate under PGD-1000 with different number of restarts and step sizes. Here we use the cross-entropy (CE) objective and C&W objective (Carlini & Wagner, 2017a), respectively. We also evaluate under the SPSA attack (Uesato et al., 2018) for $10,000$ iteration steps, with batch size 128, perturbation size 0.001 and learning rate of $1/255$.

Evaluation method			Label smoothing				
Attack	Restart	Step size	0	0.1	0.2	0.3	0.4
PGD-1000 (CE objective)	1	2/255	52.45	52.95	53.08	53.10	**53.14**
	5	2/255	52.41	52.89	53.01	**53.04**	53.03
	10	2/255	52.31	52.85	52.92	**53.02**	52.96
	10	0.5/255	52.63	52.94	**53.33**	53.30	53.25
PGD-1000 (C&W objective)	1	2/255	50.64	50.76	**51.07**	50.96	50.54
	5	2/255	50.58	50.66	**50.93**	50.86	50.44
	10	2/255	50.55	50.59	**50.90**	50.85	50.44
	10	0.5/255	50.63	50.73	51.03	**51.04**	50.52
SPSA-10000	1	1/255	61.69	61.92	**61.93**	61.79	61.53

Label smoothing

Table 4: Test accuracy (%) under different degrees of **label smoothing** (LS) on CIFAR-10. More evaluation results under, e.g., PGD-1000 can be found in Table 17.

LS	Clean	PGD-10	AA	RayS
ResNet-18				
0	82.52	53.58	48.51	53.34
0.1	82.69	54.04	48.76	53.71
0.2	82.73	54.22	49.20	53.66
0.3	82.51	54.34	**49.24**	53.59
0.4	82.39	54.13	48.83	53.40
WRN-34-10				
0	86.07	56.60	52.19	60.07
0.1	85.96	56.88	52.74	59.99
0.2	86.09	57.31	**53.00**	60.28
0.3	85.99	57.55	52.70	61.00
0.4	86.19	57.63	52.71	60.64

- **Moderate label smoothing (LS=0.1~0.2) combined with adversarial training can improve robustness.**
- **Excessive label smoothing (LS>0.4) could degrade robustness.**
- **Can be treated as a confidence calibration, according to the 80%~85% clean accuracy of adversarially trained models.**

Optimizer

Table 5: Test accuracy (%) using different **optimizers** on CIFAR-10. The model is ResNet-18 (results on WRN-34-10 is in Table 15). The initial learning rate for Adam and AdamW is 0.0001.

	Mom	Nesterov	Adam	AdamW	SGD-GC	SGD-GCC
Clean	82.52	82.83	83.20	81.68	82.77	82.93
PGD-10	53.58	53.78	48.87	46.58	53.62	53.40
AA	48.51	48.22	44.04	42.39	48.33	48.51

- **SGD momentum is good enough**

Weight decay

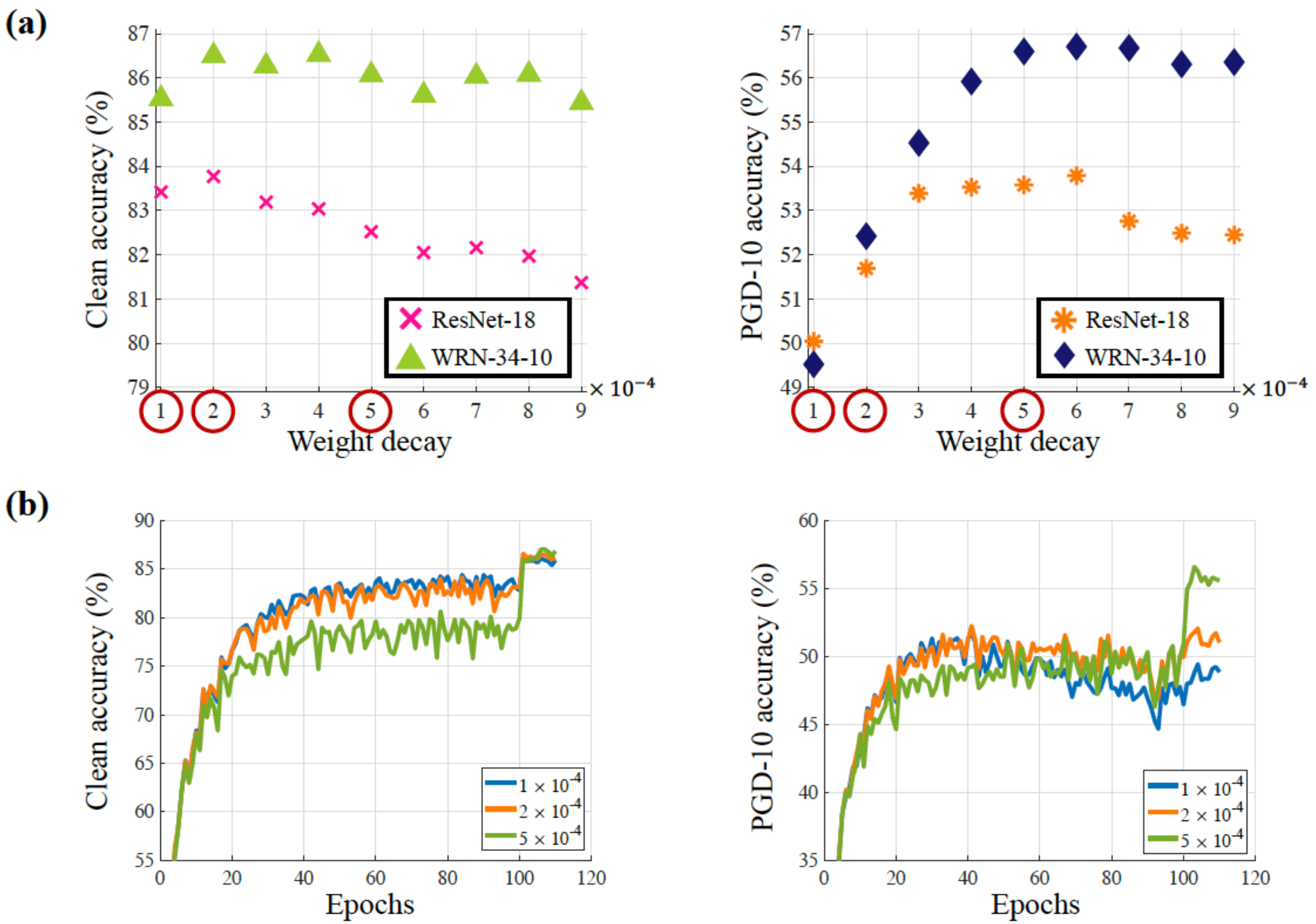


Figure 1: **(a)** Test accuracy w.r.t. different values of **weight decay**. The reported checkpoints correspond to the best PGD-10 accuracy (Rice et al., 2020). We test on two model architectures, and highlight (with red circles) three most commonly used weight decays in previous work; **(b)** Curves of test accuracy w.r.t. training epochs, where the model is WRN-34-10. We set weight decay be 1×10^{-4}, 2×10^{-4}, and 5×10^{-4}, respectively. We can observe that smaller weight decay can learn faster but also more tend to overfit w.r.t. the robust accuracy. In Fig. 4, we early decay the learning rate before the models overfitting, but weight decay of 5×10^{-4} still achieve better robustness.

Weight decay

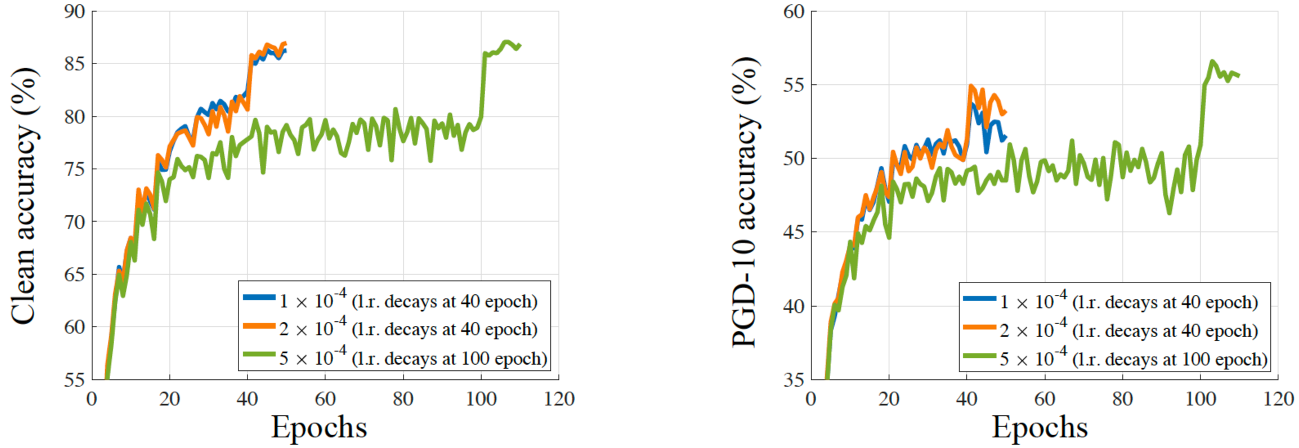


Figure 4: Curves of test accuracy w.r.t. training epochs, where the model is WRN-34-10. Here we early decay the learning rate at 40 and 45 epochs for the cases of weight decay 1×10^{-4} and 2×10^{-4}, just before they overfitting. We can see that the models can achieve the same clean accuracy as weight decay 5×10^{-4}, but still worse robustness.

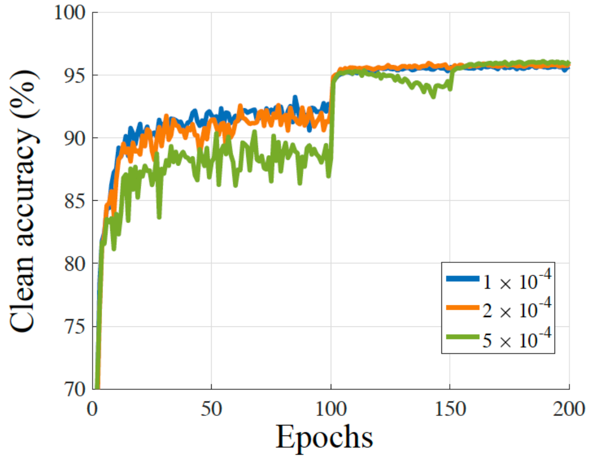


Figure 5: Curves of test accuracy w.r.t. training epochs. The model architecture is WRN-34-10, and is standardly trained on CIFAR-10. We can observe that the final performance of each model is comparable, which means that clean accuracy is less sensitive to different values of weight decay. This observation also holds for the adversarially trained models as shown in Fig. 1.

Weight decay

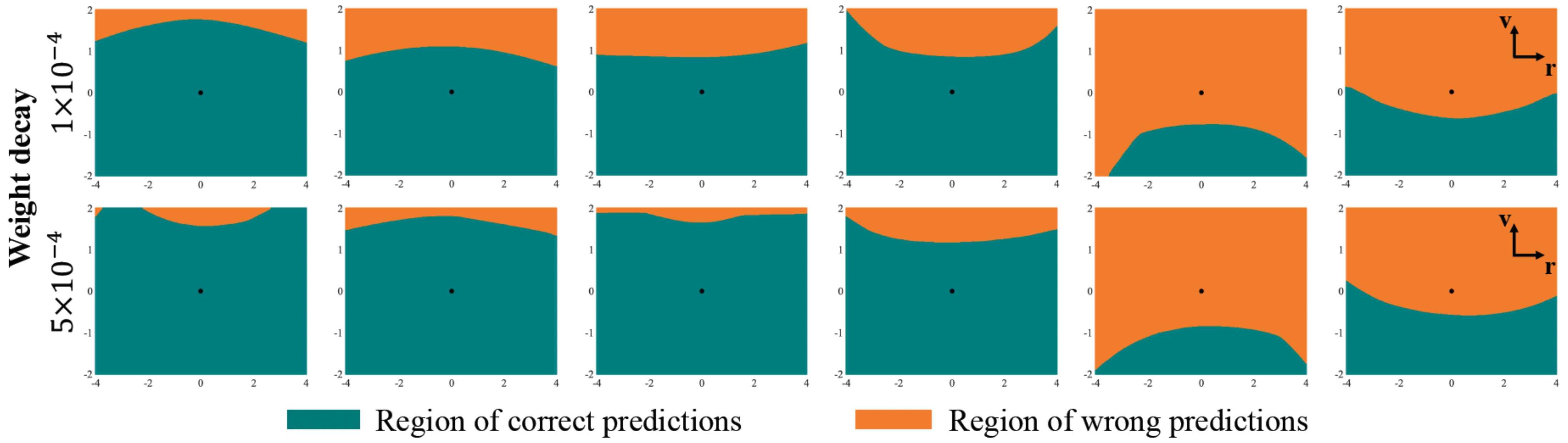
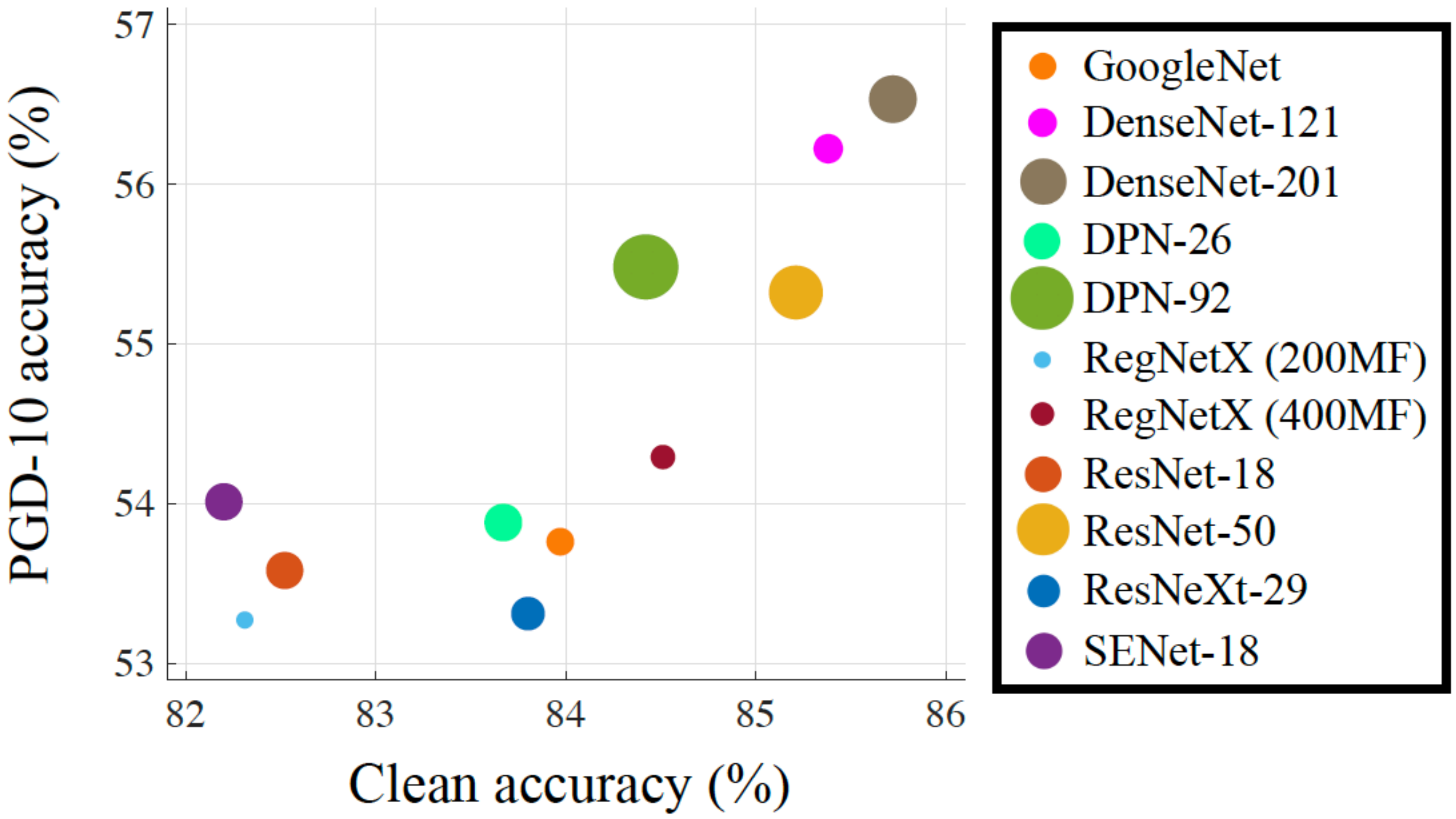


Figure 3: Random normal cross-sections of the decision boundary for PGD-AT with different **weight decay**. The model architecture is WRN-34-10. Following the examples in Moosavi-Dezfooli et al. (2019), we craft PGD-10 perturbation as the normal direction v, and r be a random direction, under the ℓ_∞ constraint of $8/255$. The values of x-axis and y-axis represent the multiplied scale factors.

Model architecture



- **Skip connections are helpful (but may require higher inference time)**

Activation function

Table 6: Test accuracy (%) under different **non-linear activation function** on CIFAR-10. The model is ResNet-18. We apply the hyperparameters recommended by Xie et al. (2020) on ImageNet for the activation function. Here the notation ‡ indicates using weight decay of 5×10^{-5}, where applying weight decay of 5×10^{-4} with these activations will lead to much worse model performance.

	ReLU	Leaky.	ELU ‡	CELU ‡	SELU ‡	GELU	Softplus	Tanh ‡
Clean	82.52	82.11	82.17	81.37	78.88	80.42	**82.80**	80.13
PGD-10	53.58	53.25	52.08	51.37	49.53	52.21	**54.30**	49.12

- **Swish could perform better (Xie et al. 2020, Gowal et al. 2020)**

Combined (PGD-AT)

Architecture	Label smooth	Weight decay	Activation function	BN mode	Accuracy Clean	Accuracy PGD-10	Accuracy AA
WRN-34-10	0	1×10^{-4}	ReLU	train	85.87	49.45	46.43
	0	2×10^{-4}	ReLU	train	86.14	52.08	48.72
	0	5×10^{-4}	ReLU	train	86.07	56.60	52.19
	0	5×10^{-4}	ReLU	eval	**87.38**	56.04	51.93
	0	5×10^{-4}	Softplus	train	86.60	56.44	52.70
	0.1	5×10^{-4}	Softplus	train	86.42	57.22	**53.01**
	0.1	5×10^{-4}	Softplus	eval	86.34	56.38	52.21
	0.2	5×10^{-4}	Softplus	train	86.10	56.55	52.91
	0.2	5×10^{-4}	Softplus	eval	86.98	56.21	52.10
WRN-34-20	0	1×10^{-4}	ReLU	train	86.21	49.74	47.58
	0	2×10^{-4}	ReLU	train	86.73	51.39	49.03
	0	5×10^{-4}	ReLU	train	86.97	57.57	53.26
	0	5×10^{-4}	ReLU	eval	87.62	57.04	53.14
	0	5×10^{-4}	Softplus	train	85.80	57.84	53.64
	0.1	5×10^{-4}	Softplus	train	85.69	57.86	**53.66**
	0.1	5×10^{-4}	Softplus	eval	**87.86**	57.33	53.23
	0.2	5×10^{-4}	Softplus	train	84.82	57.93	53.39
	0.2	5×10^{-4}	Softplus	eval	87.58	57.19	53.26

Combined (TRADES)

Architecture	Weight decay	BN mode	Activation	Clean	PGD-10	AA
Threat model: ℓ_∞ constraint, $\epsilon = 0.031$						
WRN-34-10	2×10^{-4}	train	ReLU	83.86	54.96	51.52
	2×10^{-4}	eval	ReLU	85.17	55.10	51.85
	5×10^{-4}	train	ReLU	84.17	57.34	53.51
	5×10^{-4}	eval	ReLU	**85.34**	58.54	**54.64**
	5×10^{-4}	eval	Softplus	84.66	58.05	54.20
WRN-34-20	5×10^{-4}	eval	ReLU	**86.93**	57.93	**54.42**
	5×10^{-4}	eval	Softplus	85.43	57.94	54.32
Threat model: ℓ_∞ constraint, $\epsilon = 8/255$						
Architecture	Weight decay	BN mode	Activation	Clean	PGD-10	AA
WRN-34-10	2×10^{-4}	train	ReLU	84.50	54.60	50.94
	2×10^{-4}	eval	ReLU	85.17	54.58	51.54
	5×10^{-4}	train	ReLU	84.04	57.41	53.83
	5×10^{-4}	eval	ReLU	**85.48**	57.45	53.80
	5×10^{-4}	eval	Softplus	84.24	57.59	**53.88**
WRN-34-20	2×10^{-4}	train	ReLU	84.50	53.86	51.18
	2×10^{-4}	eval	ReLU	85.48	53.21	50.59
	5×10^{-4}	train	ReLU	85.87	57.40	54.22
	5×10^{-4}	eval	ReLU	**86.43**	57.91	**54.39**
	5×10^{-4}	eval	Softplus	85.51	57.50	54.21

Simply change the weight decay (TRADES)

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

Combined (FastAT and FreeAT)

Defense	Label smooth	Weight decay	BN mode	Accuracy Clean	Accuracy PGD-10	Accuracy AA
FastAT (Wong et al., 2020)	0	2×10^{-4}	train	82.19	47.47	42.99
	0	5×10^{-4}	train	82.93	48.48	44.06
	0	5×10^{-4}	eval	**84.00**	48.16	43.66
	0.1	5×10^{-4}	train	82.83	**48.76**	**44.50**
FreeAT (Shafahi et al., 2019b)	0	2×10^{-4}	train	87.42	47.66	44.24
	0	5×10^{-4}	train	88.17	48.90	45.66
	0	5×10^{-4}	eval	**88.26**	48.50	45.49
	0.1	5×10^{-4}	train	88.07	**49.26**	**45.91**

Takeaways

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.

- **Adversarial training is more sensitive to these usually overlooked hyperparameters, compared to standard training.**
- **Standardize the basic training setting enables fairer benchmarks.**

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

Bottleneck of adversarial training

Note: ‡ indicates models which exploit additional data for training (e.g. unlabeled data, pre-training).

#	paper	model	architecture	clean	report.	AA
1	(Gowal et al., 2020)‡	*available*	WRN-70-16	91.10	65.87	65.88
2	(Gowal et al., 2020)‡	*available*	WRN-28-10	89.48	62.76	62.80
3	(Wu et al., 2020a)‡	*available*	WRN-34-15	87.67	60.65	60.65
4	(Wu et al., 2020b)‡	*available*	WRN-28-10	88.25	60.04	60.04
5	(Carmon et al., 2019)‡	*available*	WRN-28-10	89.69	62.5	59.53
6	(Gowal et al., 2020)	*available*	WRN-70-16	85.29	57.14	57.20
7	(Sehwag et al., 2020)‡	*available*	WRN-28-10	88.98	-	57.14
8	(Gowal et al., 2020)	*available*	WRN-34-20	85.64	56.82	56.86
9	(Wang et al., 2020)‡	*available*	WRN-28-10	87.50	65.04	56.29
10	(Wu et al., 2020b)	*available*	WRN-34-10	85.36	56.17	56.17
11	(Alayrac et al., 2019)‡	*available*	WRN-106-8	86.46	56.30	56.03
12	(Hendrycks et al., 2019)‡	*available*	WRN-28-10	87.11	57.4	54.92
13	(Pang et al., 2020c)	*available*	WRN-34-20	86.43	54.39	54.39
14	(Pang et al., 2020b)	*available*	WRN-34-20	85.14	-	53.74

(From https://github.com/fra31/auto-attack)

Gowal et al. use TRADES:

- weight decay 5×10^{-4}
- WRN-70-16 with Swish activation
- more data

Less significant improvement since 2018

How to bypass this bottleneck?

- Research routines of adversarial training and adversarial detection are relatively independent in previous works. Incorporate the rejection / detection module into the adversarially trained models. (*a new work public soon*)

- Include test-time purification, by introducing auxiliary models or tasks. Convert passive defenses into dynamic ones.

CVPR2021安全AI挑战者计划第六期——10万美金现金奖励

顶会借

Max-Mahalanobis Training Part 1

(Max-Mahalanobis Linear Discriminant Analysis Networks)

Tianyu Pang, Chao Du, and Jun Zhu

ICML 2018

Code: https://github.com/P2333/Max-Mahalanobis-Training

Motivation

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

Input → Non-linear Transformation → Linear Classifier → **Output**

Active area of research

(AlexNet; VGG nets; ResNets; GoogleNets; DenseNets;)

Much less active

(Softmax regression)

Inspiration one: LDA is more efficient than LR

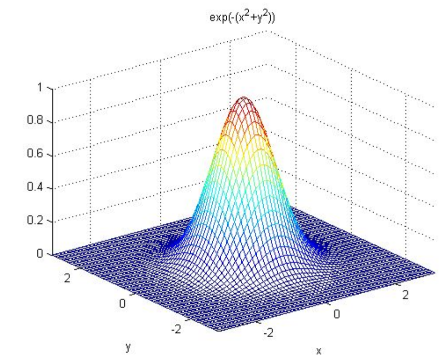
- Efron et al.(1975) show that *if the input distributes as a mixture of Gaussian*, then linear discriminant analysis (LDA) is **more efficient** than logistic regression (LR).

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

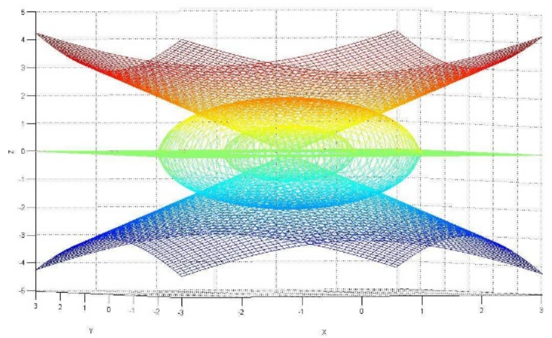
- **Deep generative models (e.g., GANs) are successful.**



Simple Distribution
(Gaussian/Mixture of Gaussian)

Deep generative models

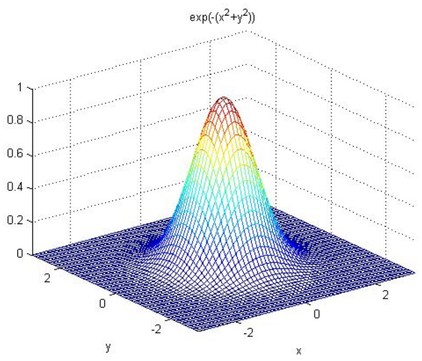
DNN



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.**

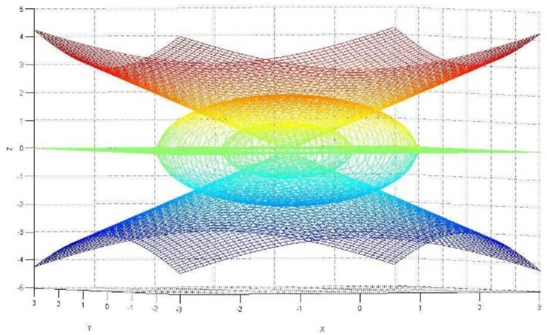


Simple Distribution
(Gaussian/Mixture of Gaussian)

Deep generative models

DNN

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.

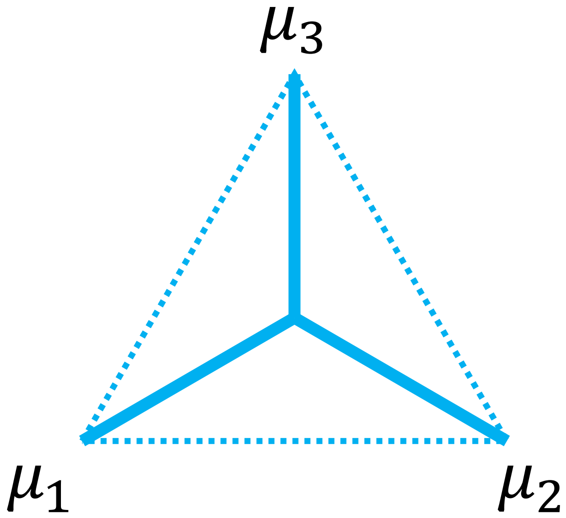
- We treat them as hyperparameters calculated by our algorithm, which can **provide theoretical guarantee on the robustness.**

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

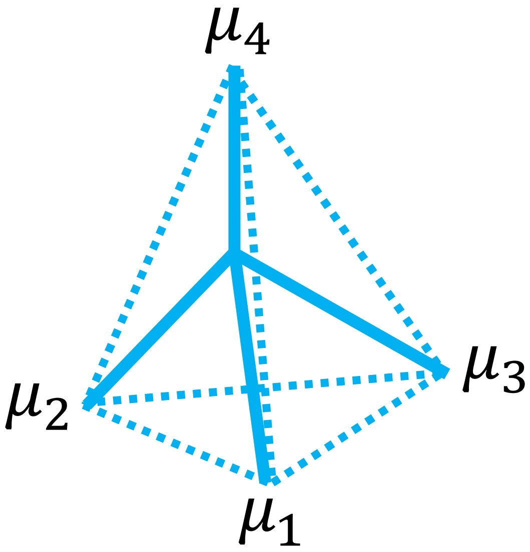
Max-Mahalanobis Distribution (MMD)

- **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 2

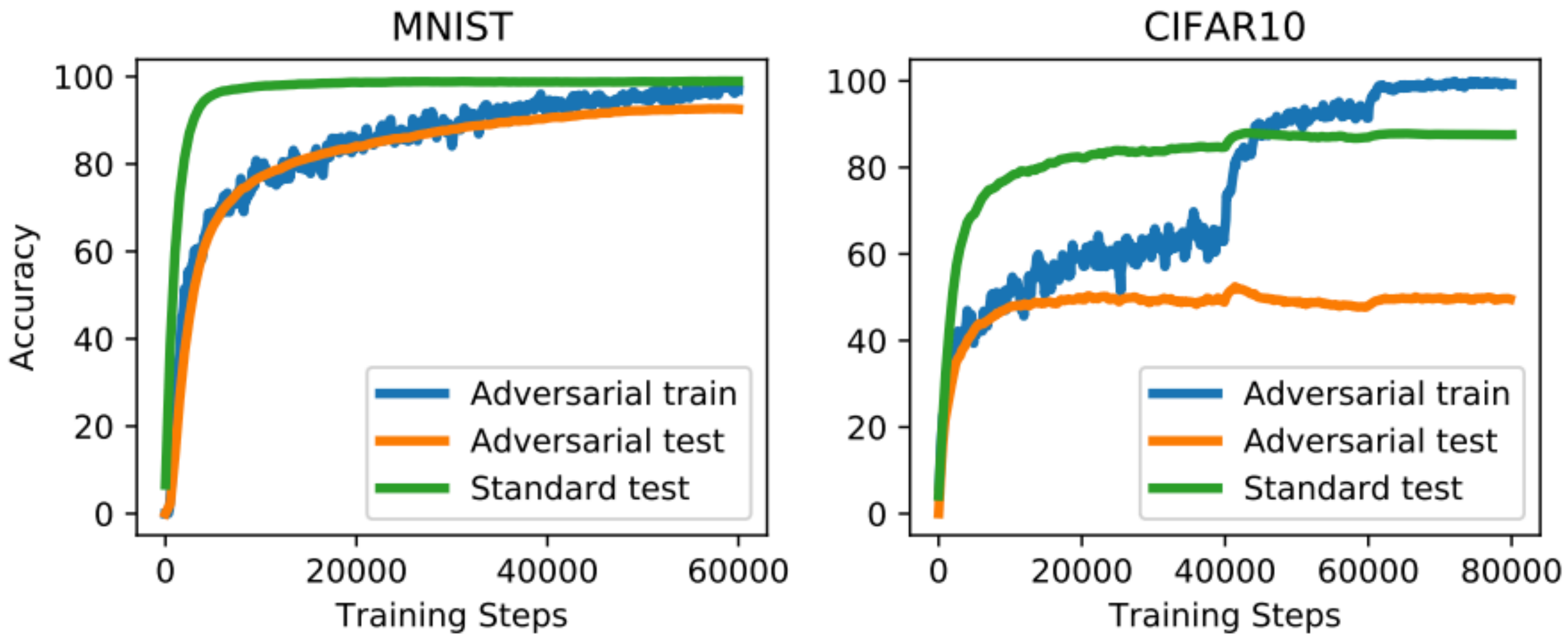
(Rethinking Softmax Cross-Entropy Loss for Adversarial Robustness)

Tianyu Pang, Kun Xu, Yinpeng Dong, Chao Du, Ning Chen and Jun Zhu

ICLR 2020

Code: https://github.com/P2333/Max-Mahalanobis-Training

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)

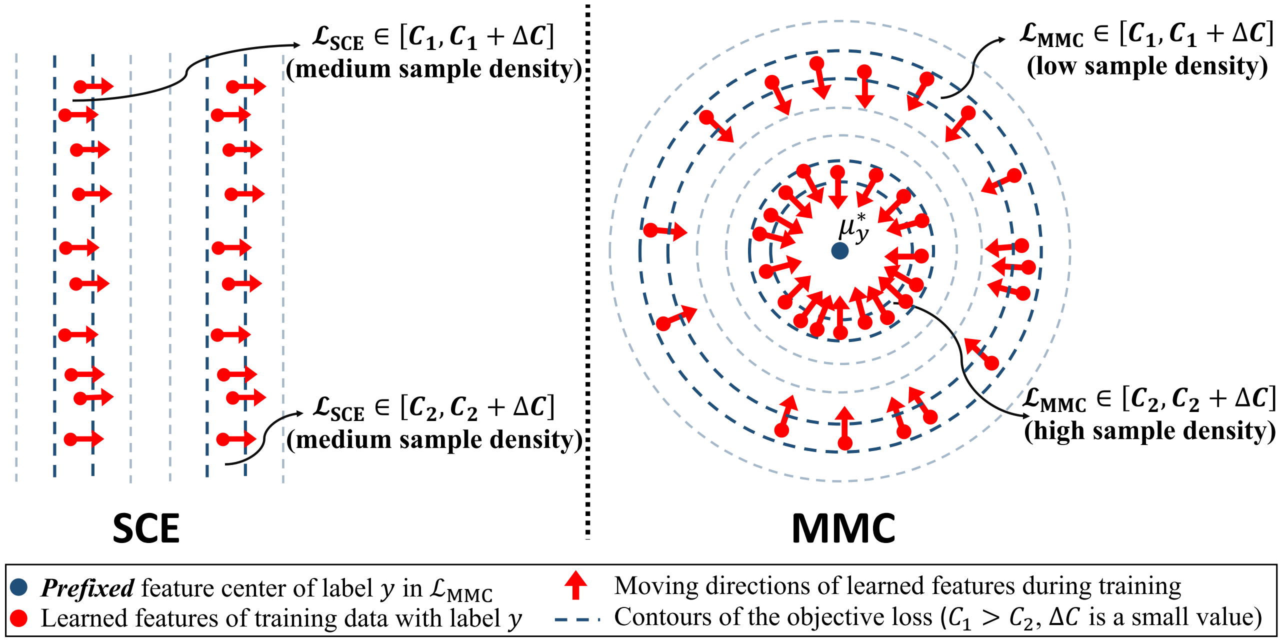
- **Our solution: Increase sample density to induce locally sufficient training data for robust learning**

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

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

Here $\mathrm{Vol}(\cdot)$ 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.



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

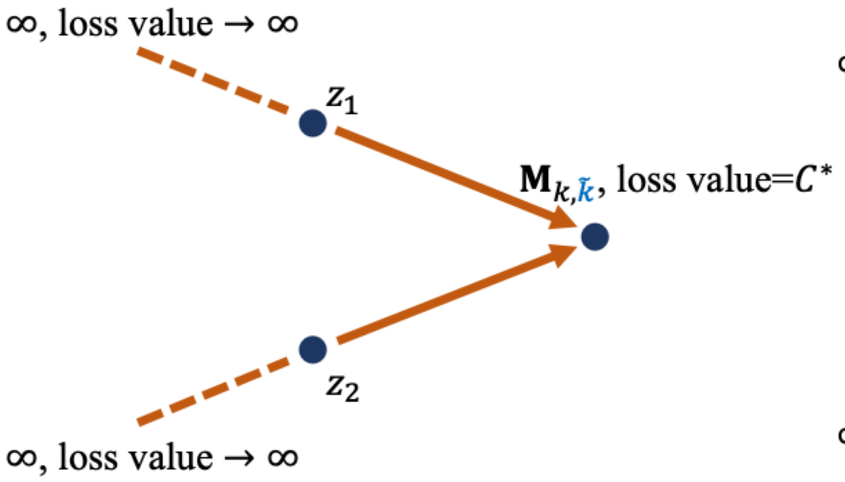
$$\text{softmax}(Wz+b)_i = \frac{\exp(W_i^\top z + b_i)}{\sum_{l\in[L]} \exp(W_l^\top z + b_l)} = \frac{\exp(-\|z - \frac{1}{2}W_i\|_2^2 + b_i + \frac{1}{4}\|W_i\|_2^2)}{\sum_{l\in[L]} \exp(-\|z - \frac{1}{2}W_l\|_2^2 + b_l + \frac{1}{4}\|W_l\|_2^2)}.$$

Induced Sample Density of g-SCE Loss

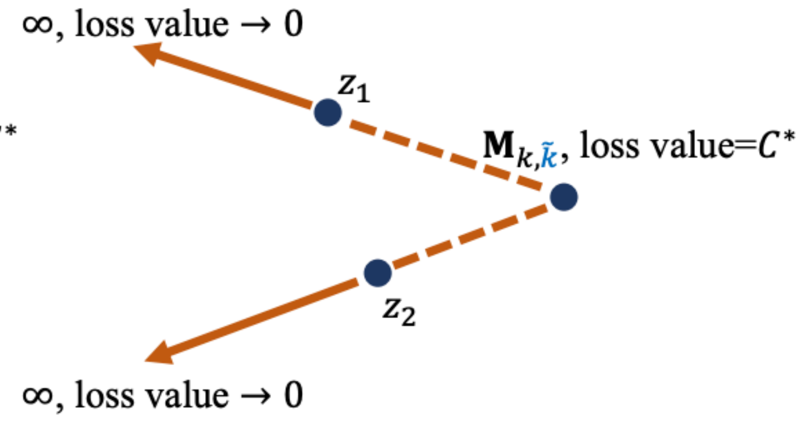
Theorem 1. *(Proof in Appendix A.1) Given* $(x, y) \in \mathcal{D}_{k,\tilde{k}}$, $z = Z(x)$ *and* $\mathcal{L}_{g\text{-}SCE}(z, y) = C$, *if there are* $\Sigma_k = \sigma_k I$, $\Sigma_{\tilde{k}} = \sigma_{\tilde{k}} I$, *and* $\sigma_k \neq \sigma_{\tilde{k}}$, *then the sample density nearby the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



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



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

(Preferred by models since lower loss values)

The ‘Curse’ of Softmax Function

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

- **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

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

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

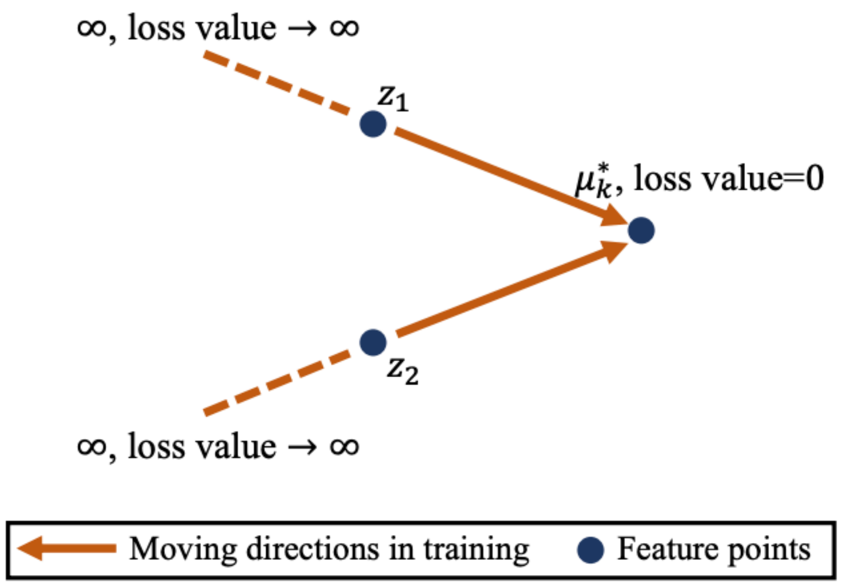
- **No softmax normalization**

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*

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

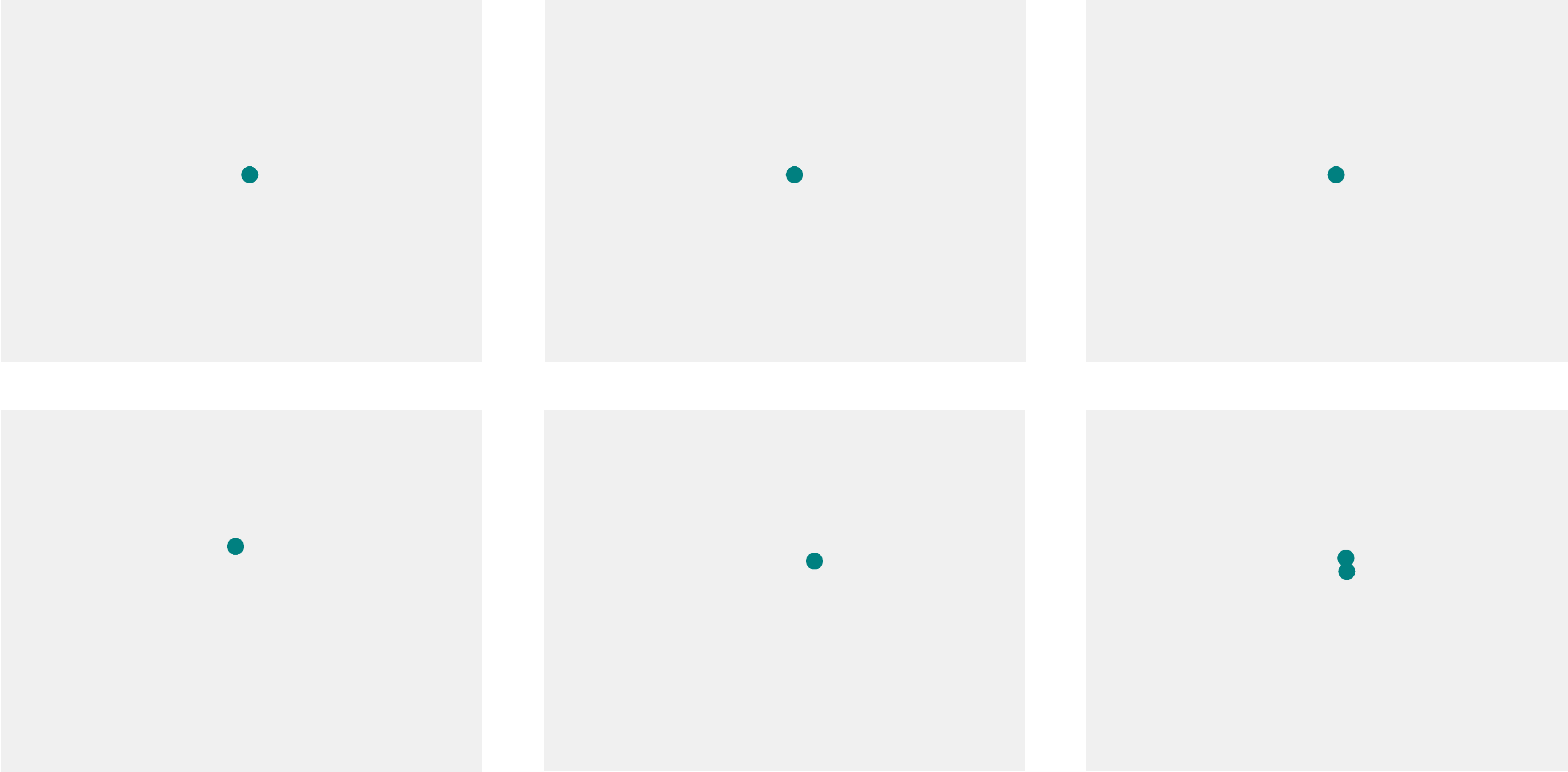
where for the input-label pair in $\mathcal{D}_k$, *there is* $\mathcal{L}_{MMC} \sim p_k(c)$.



Toy Demo on Faster Convergence

Center loss

MMC loss

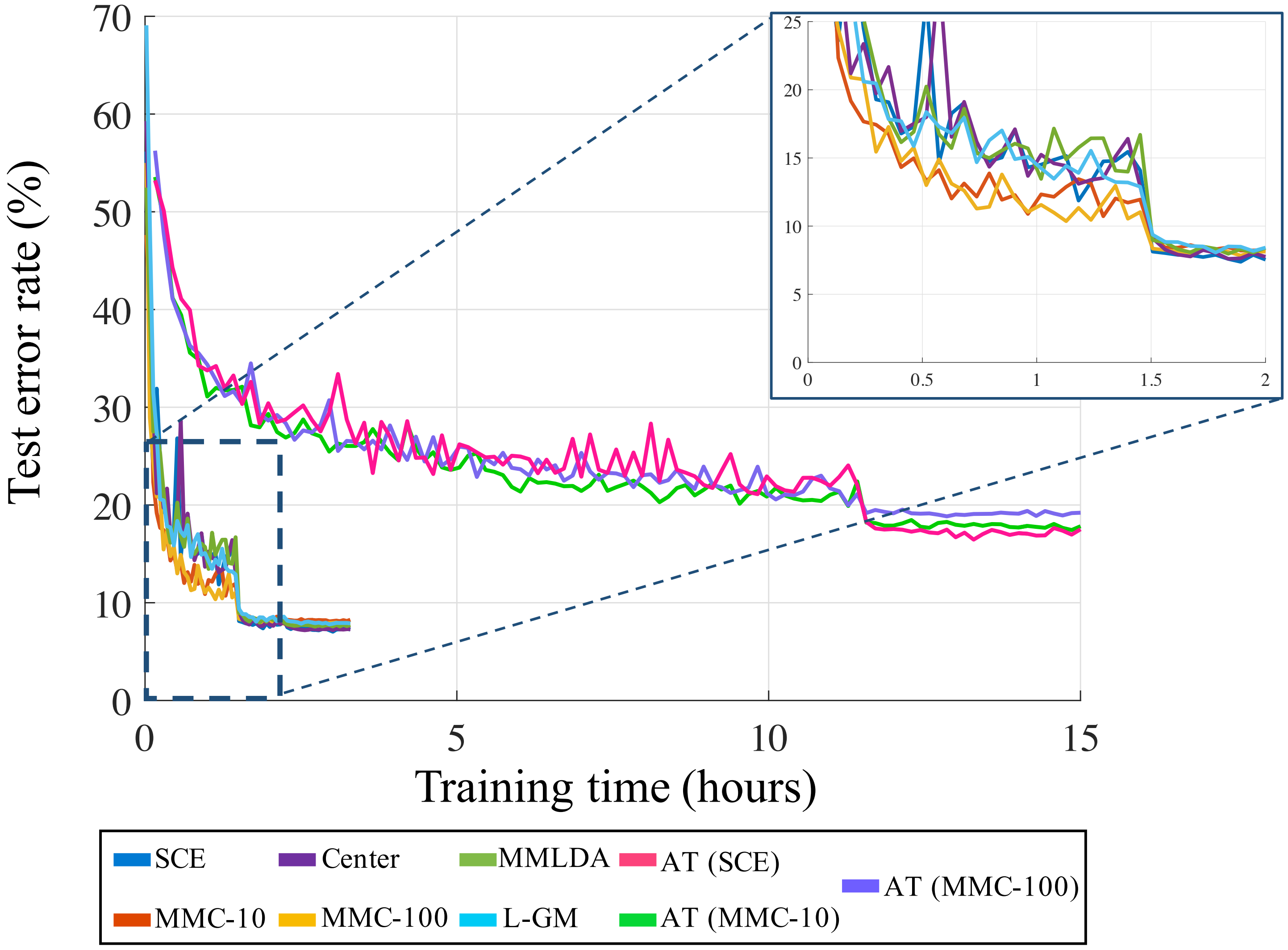


Full-batch

Mini-batch 20/1000

Mini-batch 5/1000

Empirical Faster Convergence



White-box Robustness (Adaptive Attacks)

Methods	Clean	Perturbation $\epsilon = 8/255$				Perturbation $\epsilon = 16/255$			
		$\text{PGD}_{10}^{\text{tar}}$	$\text{PGD}_{10}^{\text{un}}$	$\text{PGD}_{50}^{\text{tar}}$	$\text{PGD}_{50}^{\text{un}}$	$\text{PGD}_{10}^{\text{tar}}$	$\text{PGD}_{10}^{\text{un}}$	$\text{PGD}_{50}^{\text{tar}}$	$\text{PGD}_{50}^{\text{un}}$
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**
$\text{AT}_{10}^{\text{tar}}$ (SCE)	83.7	**70.6**	49.7	**69.8**	47.8	48.4	26.7	31.2	16.0
$\text{AT}_{10}^{\text{tar}}$ (**MMC-10**)	83.0	69.2	**54.8**	67.0	**53.5**	**58.6**	**47.3**	**44.7**	**45.1**
$\text{AT}_{10}^{\text{un}}$ (SCE)	80.9	69.8	55.4	69.4	53.9	53.3	34.1	38.5	21.5
$\text{AT}_{10}^{\text{un}}$ (**MMC-10**)	81.8	**70.8**	**56.3**	**70.1**	**55.0**	**54.7**	**37.4**	**39.9**	**27.7**

CIFAR-10

CVPR2021安全AI挑战者计划第六期——10万美金现金奖励

顶会借

Improving Adversarial Robustness via Promoting Ensemble Diversity

Tianyu Pang, Kun Xu, Chao Du, Ning Chen, and Jun Zhu

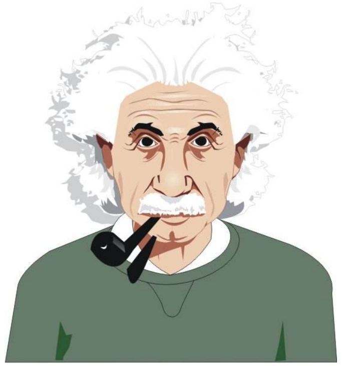
ICML 2019

Code: https://github.com/P2333/Adaptive-Diversity-Promoting

Previous Defense Strategies

Single model defense:

e.g., adversarial training

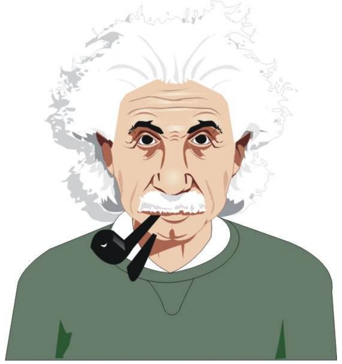


Base Model

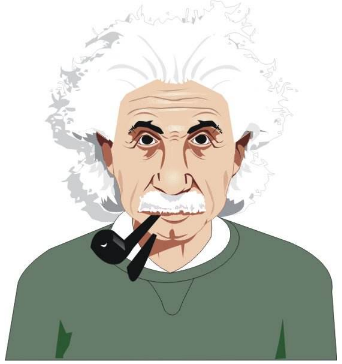
Enhanced Model

Previous Defense Strategies

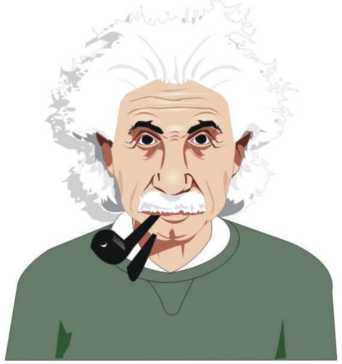
Ensemble model defense:



Member 1



Member 2

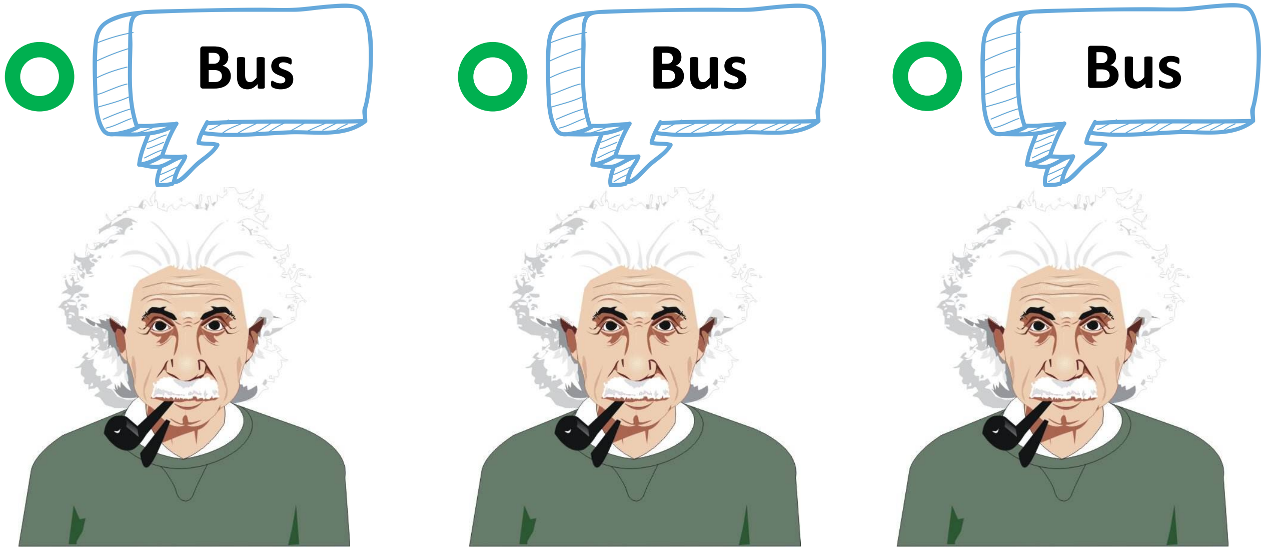


Member 3

Previous Defense Strategies

Ensemble model defense:

Clean input



Member 1 Member 2 Member 3

Previous Defense Strategies

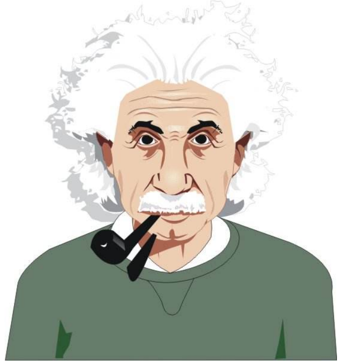
Ensemble model defense:

Adversarial input

Member 1 Member 2 Member 3

Our Strategy

Training ensembles with diversity:



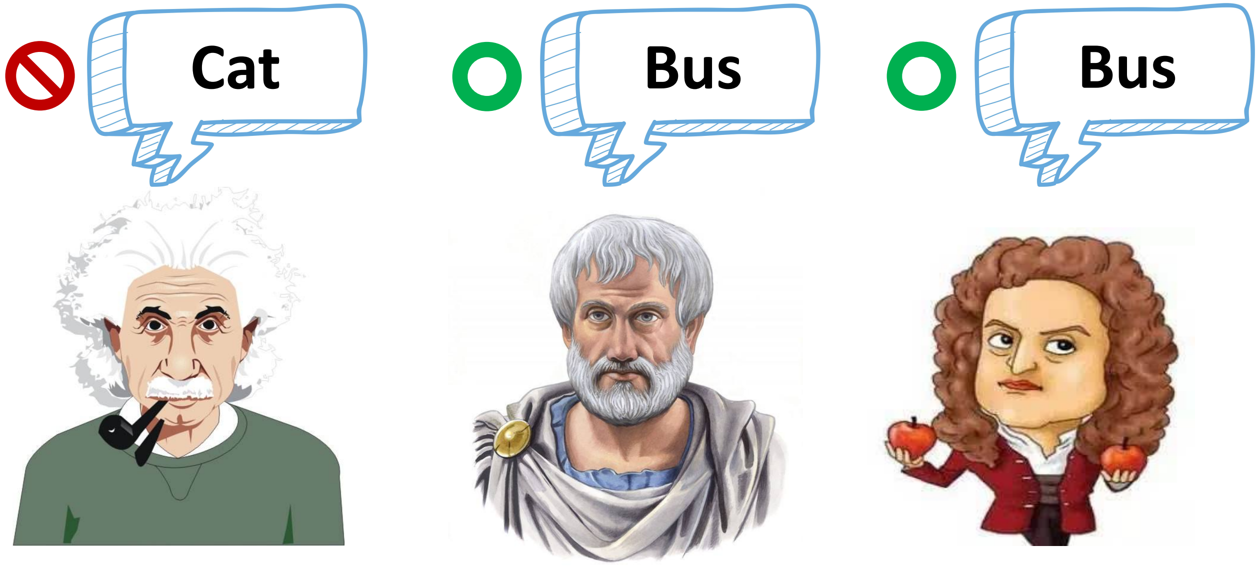
Member 1

Member 2

Member 3

Our Strategy

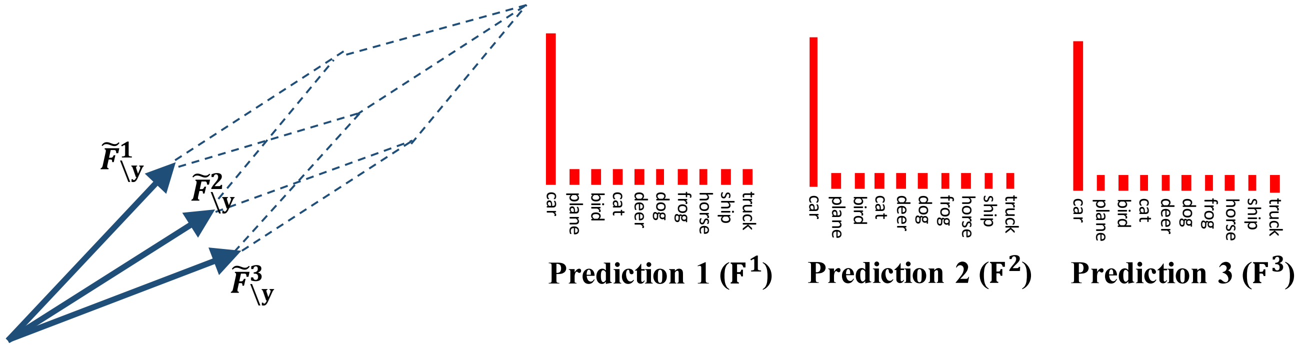
Training ensembles with diversity:



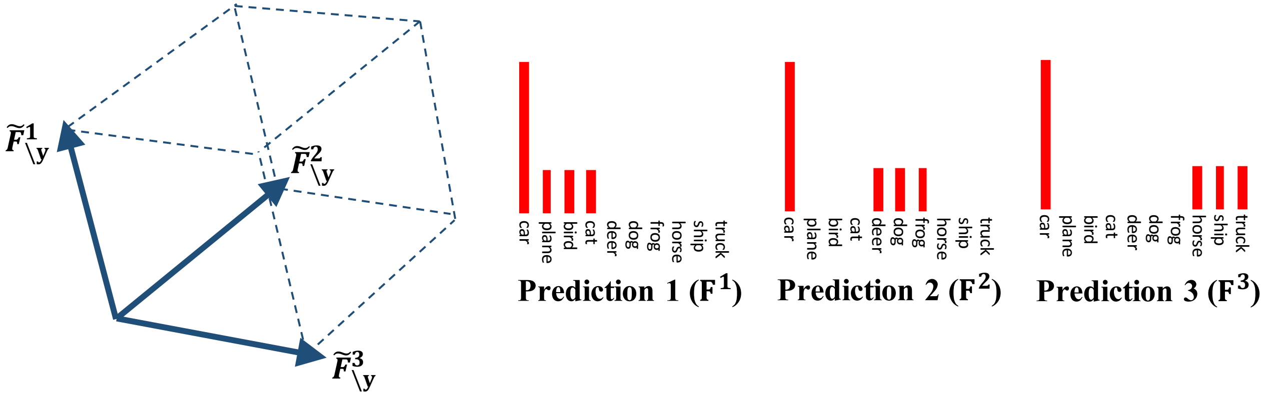
Member 1 | Member 2 | Member 3

Adversarial input

Adaptive Diversity Promoting



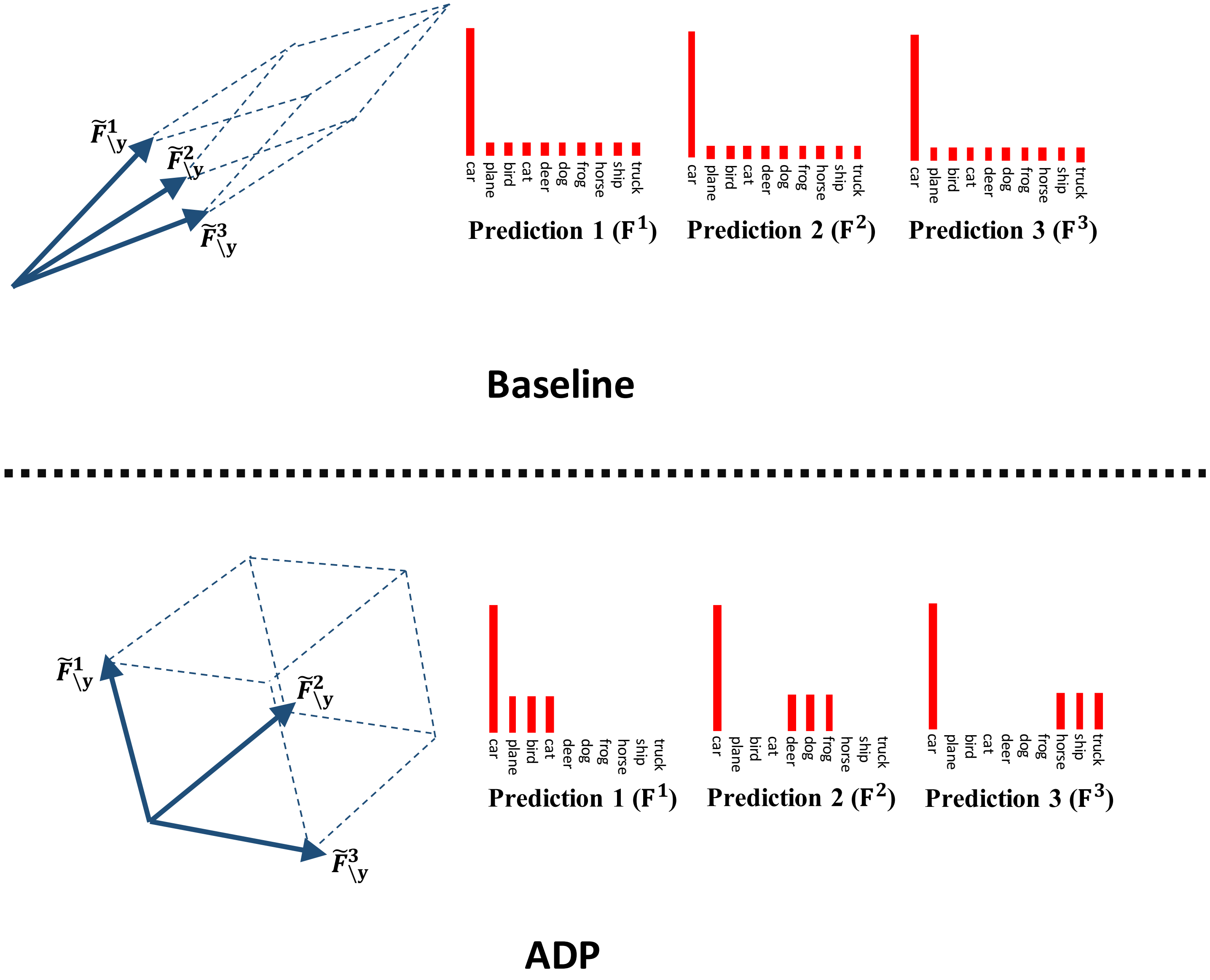
Baseline



ADP

- Promoting diversity on **non-maximal predictions**

Adaptive Diversity Promoting



- Promoting diversity on **non-maximal predictions**

correspond to all potentially wrong labels returned for the adversarial examples

Formulas of ADP

Based on the intuitive insights, we define the ensemble diversity as

$$\mathbb{ED} = \det(\tilde{M}_{\setminus y}^{\top} \tilde{M}_{\setminus y})$$

where $\tilde{M}_{\setminus y} = (\tilde{F}_{\setminus y}^{1}, \cdots, \tilde{F}_{\setminus y}^{K}) \in \mathbb{R}^{(L-1)\times K}$ **are normalized non-maximal prediction. This definition is based on the fact that**

$$\det(\tilde{M}_{\setminus y}^{\top} \tilde{M}_{\setminus y}) = \mathrm{Vol}^2(\{\tilde{F}_{\setminus y}^{k}\}_{k\in[K]})$$

Formulas of ADP

So the ADP regularizer is

$$\mathrm{ADP}_{\alpha,\beta}(x,y) = \alpha \cdot \mathcal{H}(\mathcal{F}) + \beta \cdot \log\left(\mathbb{ED}\right)$$

Formulas of ADP

So the ADP regularizer is

$$\mathrm{ADP}_{\alpha,\beta}(x,y) = \underline{\alpha \cdot \mathcal{H}(\mathcal{F})} + \beta \cdot \log\left(\mathbb{ED}\right)$$

Theorem 1. *(Proof in Appendix A) If $\alpha = 0$, then $\forall \beta \geq 0$, the optimal solution of the minimization problem (6) satisfies the equations $F^k = 1_y$, where $k \in [K]$.*

Formulas of ADP

So the ADP regularizer is

$$\mathrm{ADP}_{\alpha,\beta}(x,y) = \underline{\alpha \cdot \mathcal{H}(\mathcal{F})} + \beta \cdot \log\left(\mathbb{ED}\right)$$

Theorem 2. *(Proof in Appendix A) When $\alpha > 0$ and $\beta = 0$, the optimal solution of the minimization problem (6) satisfies the equations $F_y^k = \mathcal{F}_y$, $\mathcal{F}_j = \frac{1-\mathcal{F}_y}{L-1}$ and*

$$\frac{1}{\mathcal{F}_y} = \frac{\alpha}{K} \log \frac{\mathcal{F}_y(L-1)}{1-\mathcal{F}_y}, \tag{7}$$

where $k \in [K]$ and $j \in [L] \backslash \{y\}$.

Formulas of ADP

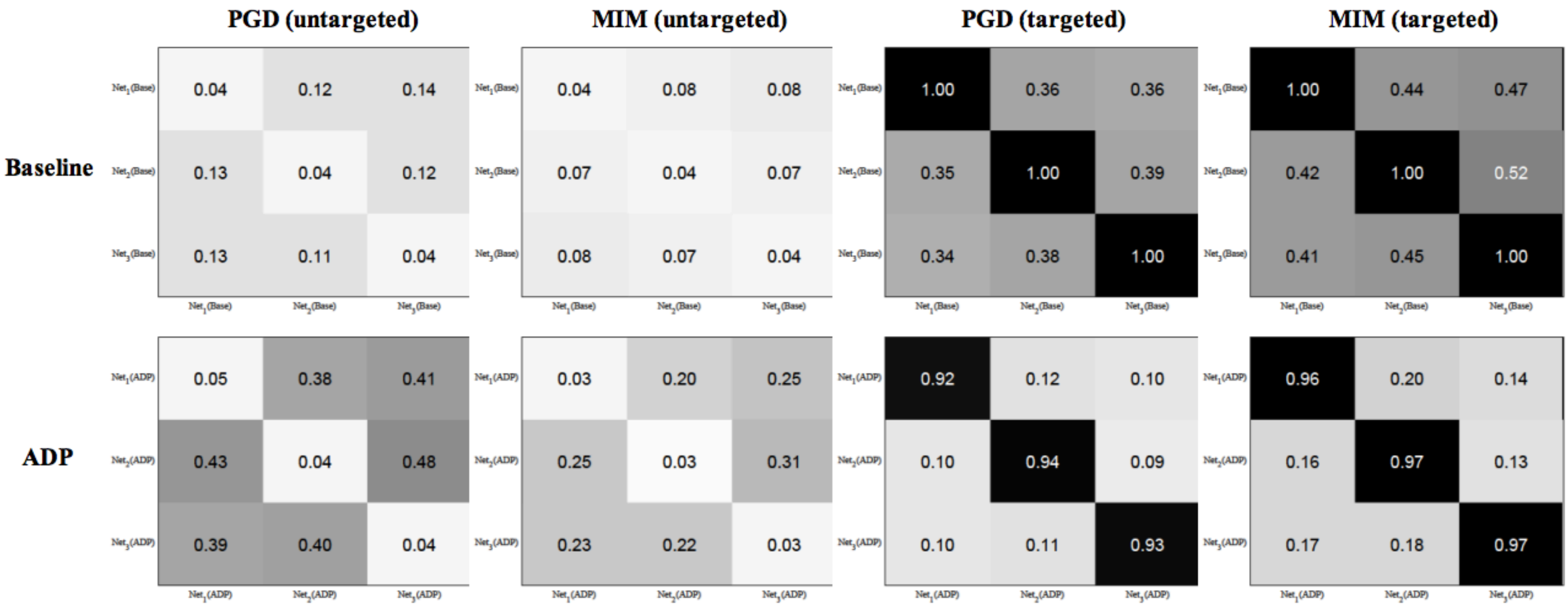
So the ADP regularizer is

$$\mathrm{ADP}_{\alpha,\beta}(x,y) = \alpha \cdot \mathcal{H}(\mathcal{F}) + \beta \cdot \log\left(\mathbb{ED}\right)$$

Corollary 1. *If there is $K \mid (L-1)$, then $\forall \alpha, \beta > 0$, the optimal solution of the minimization problem (6) satisfies the Eq. (7). Besides, let $S = \{s_1, ,\cdots, s_K\}$ be any partition of the index set $[L]\backslash\{y\}$, where $\forall k \in [K]$, $|s_k| = \frac{L-1}{K}$. Then the optimal solution further satisfies:*

$$F_j^k = \begin{cases} \frac{K(1-\mathcal{F}_y)}{L-1}, & j \in s_k, \\ \mathcal{F}_y, & j = y, \\ 0, & otherwise. \end{cases} \tag{8}$$

Experiments



Adversarial transferability among individual members of ensembles

Towards Robust Detection of Adversarial Examples

Tianyu Pang, Chao Du, Yinpeng Dong, and Jun Zhu

NeurIPS 2018

Code: https://github.com/P2333/Reverse-Cross-Entropy

We Detect Adversarial Examples, and How?

Design new detectors:

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

We Detect Adversarial Examples, and How?

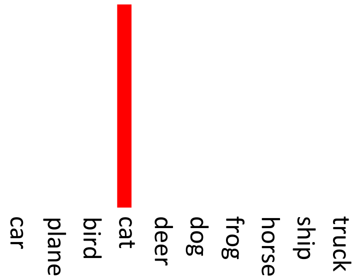
Design new detectors:

- 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):

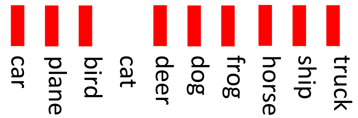


1_y: One-hot label

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

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

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.

The Insights of RCE Training

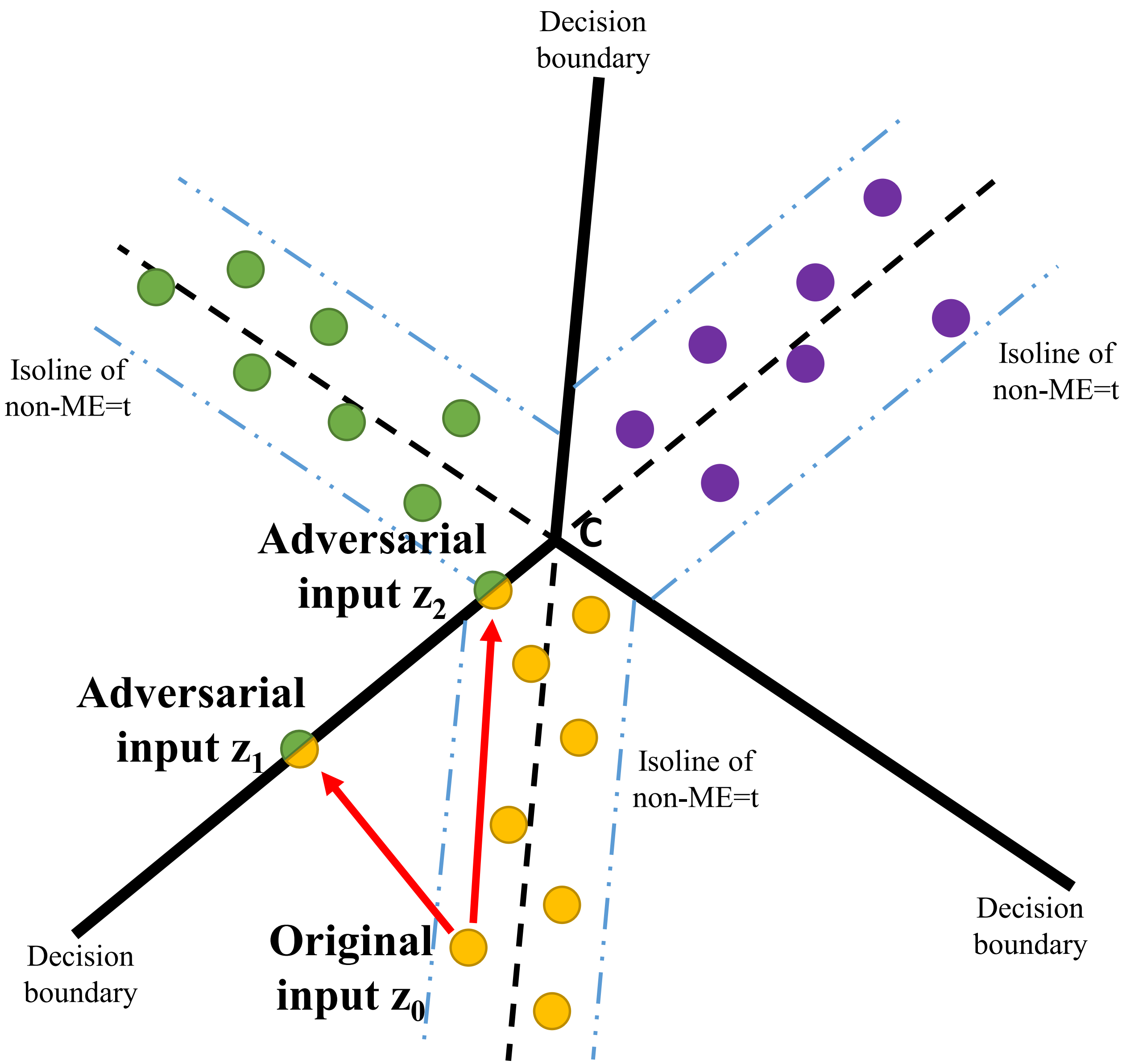
We first define the non-maximal entropy (non-ME) as:

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

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

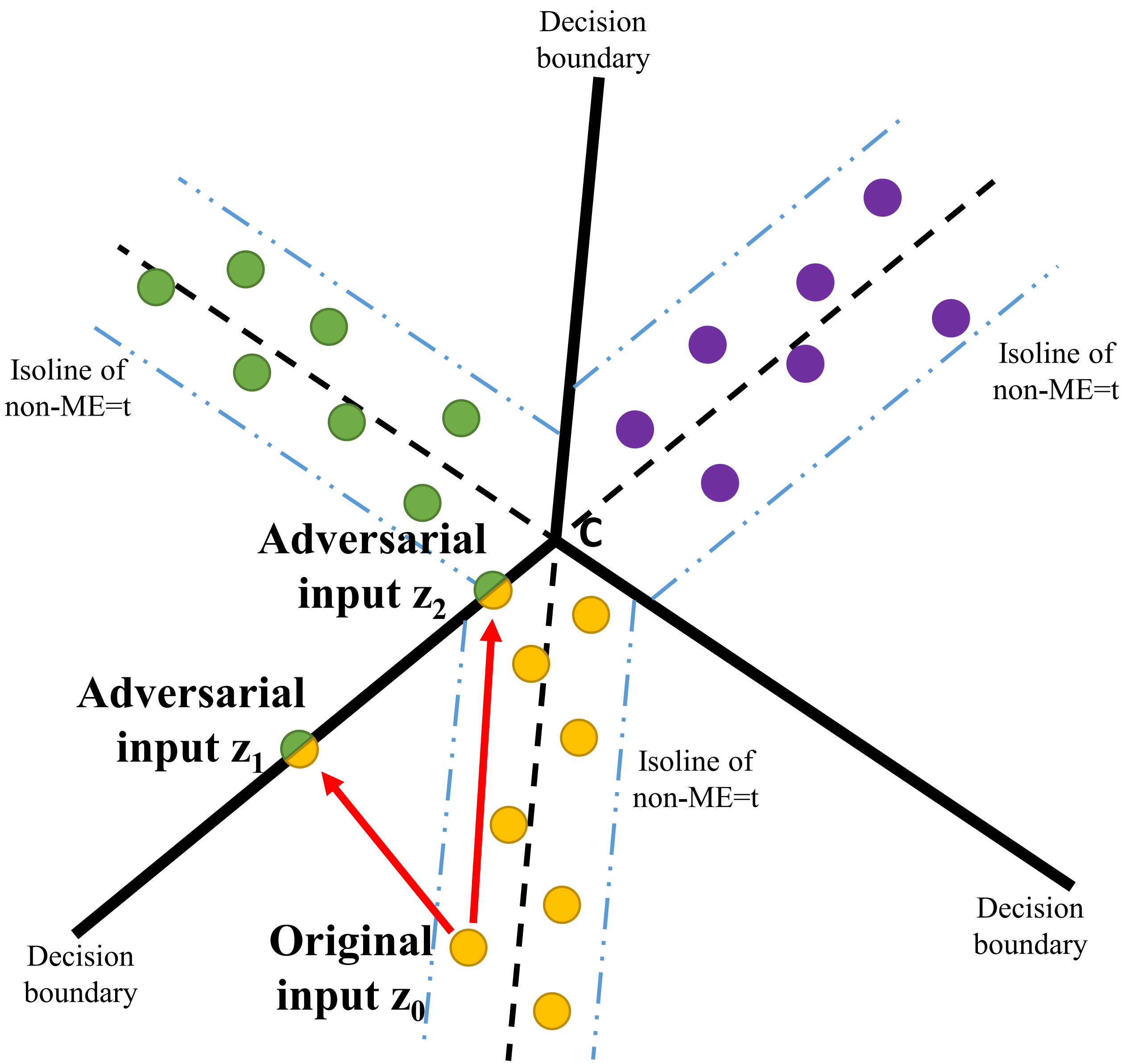
RCE training encourages the maximal prediction to tend to 1, while maximizing the non-ME.

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)

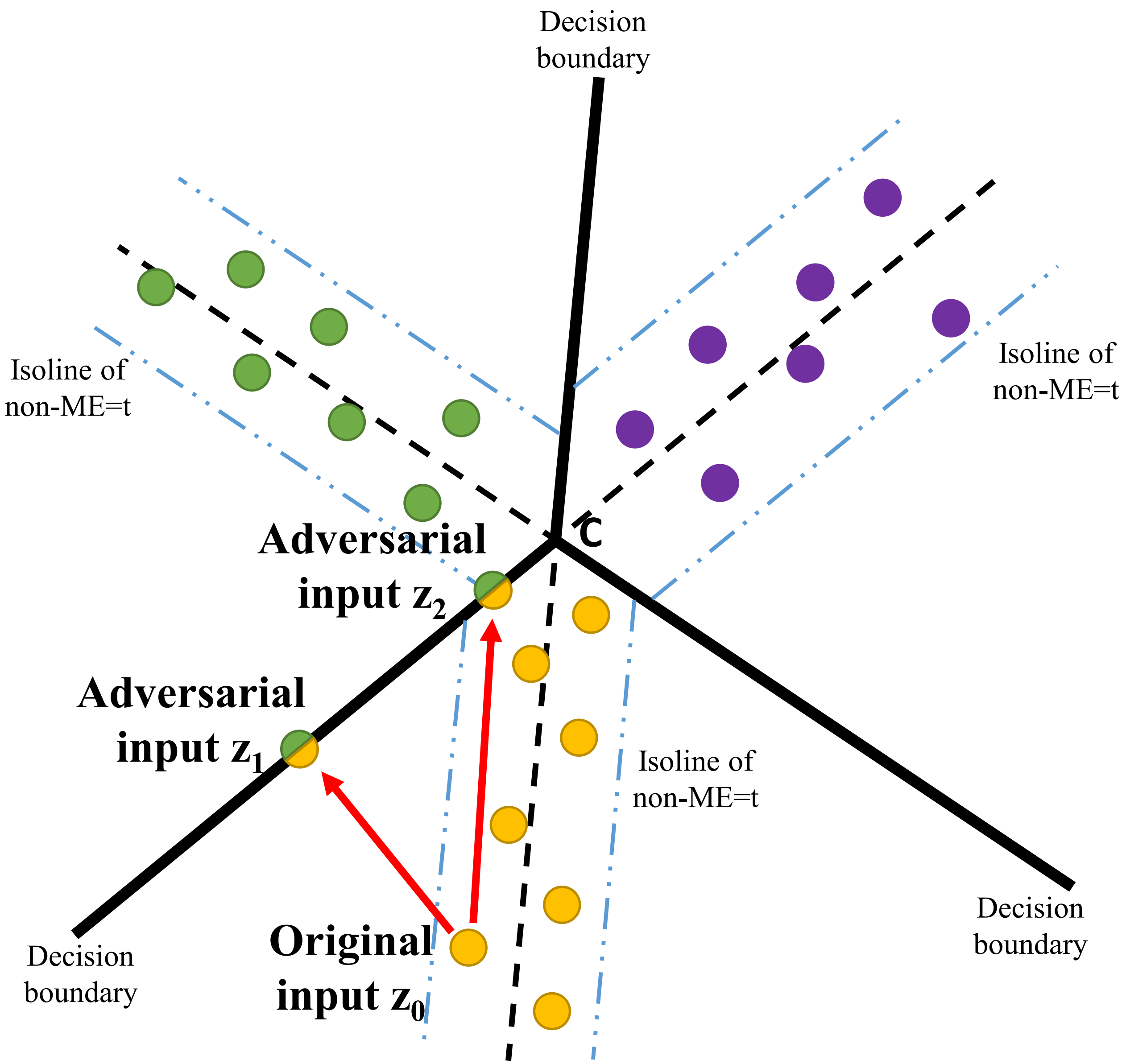
The Insights of RCE Training



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

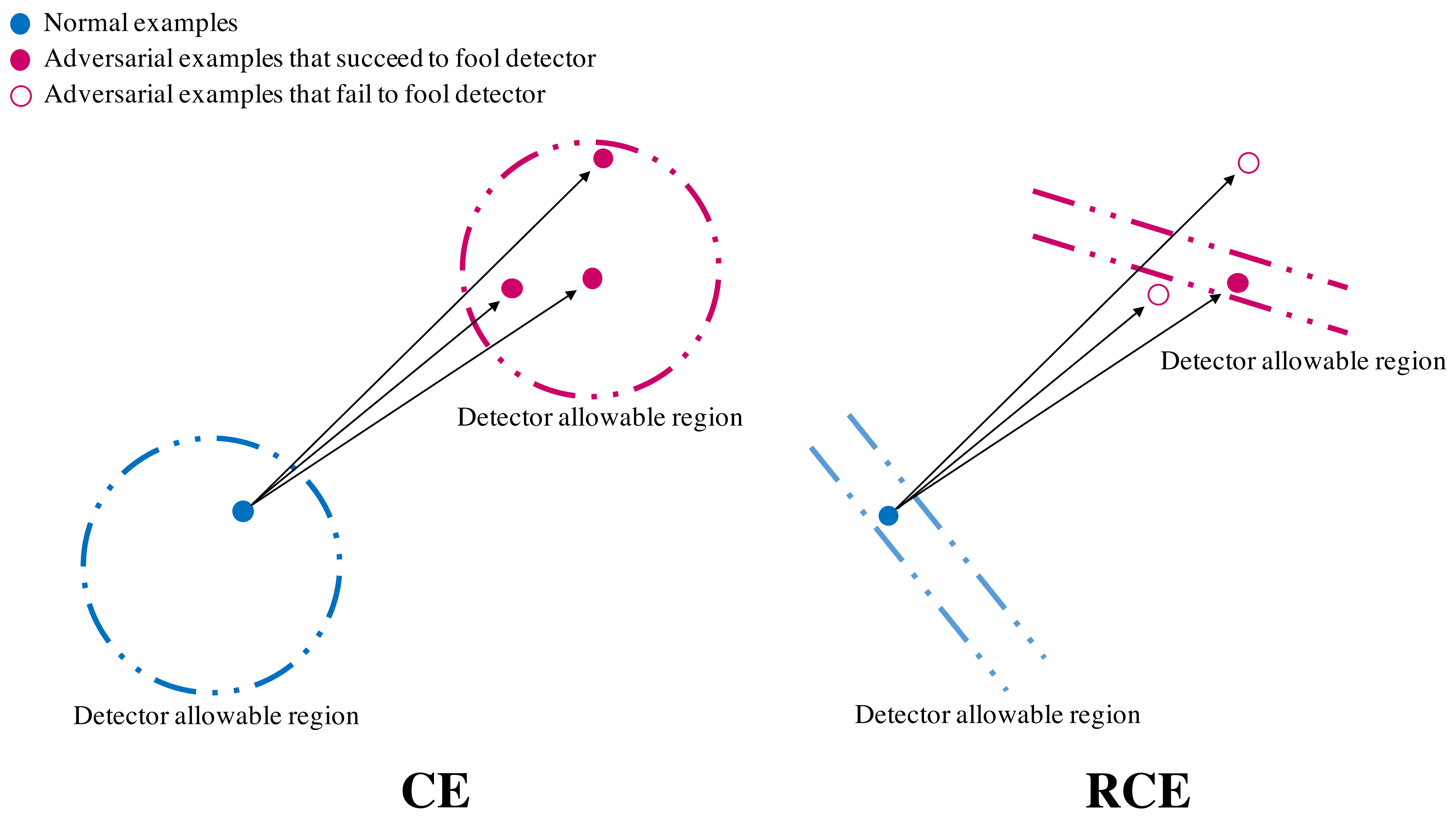
When the non-ME of the returned predictions are maximized, the learned features for each class with tend to locate near the black dash lines, where the points on the dash lines have the maximal non-ME.

The Insights of RCE Training



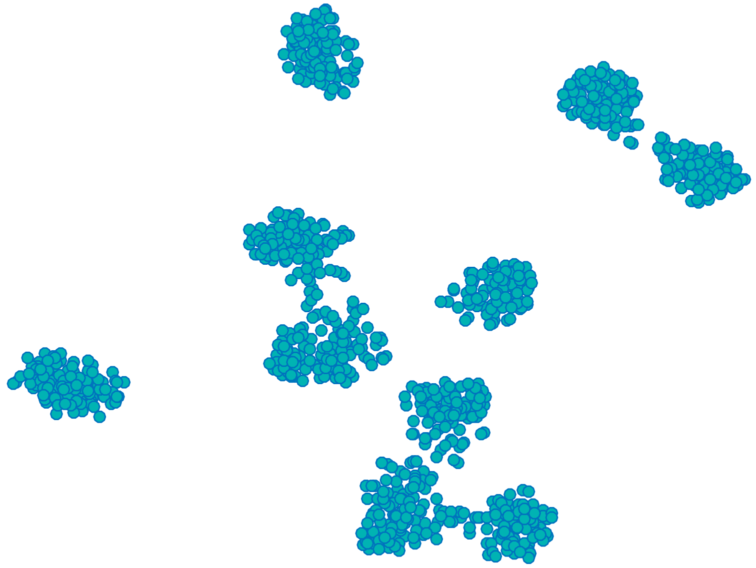
Then if an adversary want to craft an adversarial example based on z_0, he has to move further to z_2 rather than z_1 to obtain a normal value of non-ME.

The Insights of RCE Training



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

Experiments

Attack	Obj.	MNIST Confidence	MNIST non-ME	MNIST K-density	CIFAR-10 Confidence	CIFAR-10 non-ME	CIFAR-10 K-density
FGSM	CE	79.7	66.8	98.8 (-)	71.5	66.9	**99.7** (-)
	RCE	98.8	98.6	**99.4** (*)	92.6	91.4	98.0 (*)
BIM	CE	88.9	70.5	90.0 (-)	0.0	64.6	**100.0** (-)
	RCE	91.7	90.6	**91.8** (*)	0.7	70.2	**100.0** (*)
ILCM	CE	98.4	50.4	96.2 (-)	16.4	37.1	84.2 (-)
	RCE	100.0	97.0	**98.6** (*)	64.1	77.8	**93.9** (*)
JSMA	CE	98.6	60.1	97.7 (-)	99.2	27.3	85.8 (-)
	RCE	100.0	99.4	**99.0** (*)	99.5	91.9	**95.4** (*)
C&W	CE	98.6	64.1	99.4 (-)	99.5	50.2	95.3 (-)
	RCE	100.0	99.5	**99.8** (*)	99.6	94.7	**98.2** (*)
C&W-hc	CE	0.0	40.0	91.1 (-)	0.0	28.8	75.4 (-)
	RCE	0.1	93.4	**99.6** (*)	0.2	53.6	**91.8** (*)

AUC-scores (10^{-2}) on adversarial examples

Thanks

CVPR2021安全AI挑战者计划第六期——10万美金现金奖励

顶会借