Maximum-Entropy Adversarial Data Augmentation for Improved Generalization and Robustness

Long Zhao[1] **Ting Liu**[2] **Xi Peng**[3] **Dimitris Metaxas**[1]
[1]Rutgers University [2]Google Research [3]University of Delaware
{lz311,dnm}@cs.rutgers.edu liuti@google.com xipeng@udel.edu

Introduction

■ Data shifts

Small corruptions or adversarial attacks lead to significant performance degradation of deep learning models

■ Adversarial data augmentation

It is difficult to define heuristics to generate effective fictitious target distributions containing “hard” adversarial perturbations that are largely different from the source distribution.

Motivation

$$I(X;Y) = H(X) - H(X|Y)$$
$$= H(X) + H(Y) - H(X,Y)$$

■ Information Bottleneck[1]

Encourages the model to learn an optimal representation by diminishing the irrelevant parts of the input variable that do not contribute to the prediction.

$$\text{minimize}\{I(X;Z) - \lambda I(Y;Z)\}$$

$\mathrm{I(X;Z)}$ which reflects how much Z compresses X

$\mathrm{I(Y;Z)}$ which reflects how well Z predicts Y

λ controls the trade-off between compression and prediction

[1]Naftali Tishby, Fernando C. Pereira, and William Bialek. The information bottleneck method. In Proceedings of the Annual Allerton Conference on Communication, Control, and Computing

Motivation

■ Adversarial Data Augmentation

The worst-case in the training deep neural networks in a single source domain P_0 and deploying it to unforeseen domains P

$$\underset{\theta \in \Theta}{\text{minimize}} \left\{ \sup_{P} \{\mathbb{E}[\mathcal{L}(\theta; X, Y)] : D_\theta(P, P_0) \leq \rho\} \right\}$$

$$\underset{\theta \in \Theta}{\text{minimize}} \left\{ \mathcal{F}(\theta) := \sup_{P} \{\mathbb{E}[\mathcal{L}(\theta; X, Y)] - \gamma D_\theta(P, P_0)\} \right\}$$

Methodology

- Incorporate the **IB principle** into adversarial data augmentation

$$\underset{\theta\in\Theta}{\text{minimize}}\left\{\mathcal{F}_{\text{IB}}(\theta) := \sup_{P}\left\{\mathbb{E}[\mathcal{L}_{\text{IB}}(\theta; X, Y)] - \gamma D_{\theta}(P, P_0)\right\}\right\}$$

$$\mathcal{L}_{\text{CE}}(\theta; X, Y) + \beta I(X; Z)$$

1. maximization phase： produced new data points to mimic fictitious target distributions P that satisfy the constraint $D_{\theta}(P, P_0) \leq \rho$
2. minimization phase： the network parameters are updated by the loss function L_{IB} evaluated on the adversarial examples generated from the maximization phase

Methodology

■ Regularizing Maximization Phase via Maximum Entropy

$$\underset{\theta\in\Theta}{\text{minimize}}\left\{\mathcal{F}_{\mathsf{IB}}(\theta) \coloneqq \sup_{P}\left\{\mathbb{E}[\mathcal{L}_{\mathsf{IB}}(\theta;X,Y)] - \gamma D_\theta(P,P_0)\right\}\right\}$$

$$\mathcal{L}_{\mathsf{CE}}(\theta;X,Y) + \beta I(X;Z)$$

Proposition 1. *Consider a deterministic neural network, the parameters θ of which are fixed. Given the input X, let $\hat{Y}$ be the network prediction and Z be the latent representation of X. Then, the mutual information $I(X;Z)$ is lower bounded by $H(\hat{Y})$, i.e., we have that,*

$$I(X;Z) \geq I(X;\hat{Y}) = H(\hat{Y}) - H(\hat{Y}|X) = H(\hat{Y}). \tag{6}$$

Data Processing Inequality

$$h(\theta;x) \coloneqq H(\hat{Y}=\hat{y}) = -\sum_{i=1}^{|\mathcal{Y}|} p_{(i)}(\theta;x)\log p_{(i)}(\theta;x)$$

Methodology

Algorithm 1 Max-Entropy Adversarial Data Augmentation (ME-ADA)

Input: Source dataset $\mathcal{D}_0 = \{X_i, Y_i\}_{1 \le i \le N}$ and initialized network weights θ_0
Output: Learned network weights θ

Initialize $\theta \leftarrow \theta_0$, $\mathcal{D} \leftarrow \mathcal{D}_0$
for $k = 1, \ldots, K$ **do** ▷ Run the minimax procedure K times
  for $t = 1, \ldots, T_{\mathsf{MIN}}$ **do** ▷ Run the minimization phase T_{MIN} times
    Sample (X_t, Y_t) uniformly from dataset $\mathcal{D}$
    $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_{\mathsf{IB}}(\theta; X_t, Y_t)$
  for all $(X_i, Y_i) \in \mathcal{D}$ **do**
    $X_i^k \leftarrow X_i$
    for $t = 1, \ldots, T_{\mathsf{MAX}}$ **do** ▷ Run the maximization phase T_{MAX} times
      $X_i^k \leftarrow X_i^k + \eta \nabla_{X_i^k} \left\{ \mathcal{L}_{\mathsf{CE}}(\theta; X_i^k, Y_i) + \beta h(\theta; X_i^k) - \gamma c_\theta((X_i^k, Y_i), (X_i, Y_i)) \right\}$
    Append (X_i^k, Y_i^k) to dataset $\mathcal{D}$
while *not reach maximum steps* **do**
  Sample (X_i, Y_i) uniformly from dataset $\mathcal{D}$
  $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_{\mathsf{IB}}(\theta; X_i, Y_i)$

Experimental

Table 1: Average classification accuracy (%) and standard deviation of models trained on MNIST [40] and evaluated on SVHN [48], MNIST-M [22], SYN [22] and USPS [15]. The results are averaged over ten runs. Best performances are highlighted in bold. The results of PAR are obtained from [73].

	SVHN [48]	MNIST-M [22]	SYN [22]	USPS [15]	Average
Standard (ERM [66])	31.95 ± 1.91	55.96 ± 1.39	43.85 ± 1.27	79.92 ± 0.98	52.92 ± 0.98
PAR [69]	36.08 ± 1.27	61.16 ± 0.21	45.48 ± 0.35	79.95 ± 1.18	55.67 ± 0.33
Adv. Augment (ADA) [68]	35.70 ± 2.00	58.65 ± 1.72	47.18 ± 0.61	80.40 ± 1.70	55.48 ± 0.74
+ *Max Entropy* (ME-ADA)	42.00 ± 1.74	**63.98** ± 1.82	49.80 ± 1.74	79.10 ± 1.03	58.72 ± 1.12
+ *Max Entropy* w/ BNN	**42.56** ± 1.45	63.27 ± 2.09	**50.39** ± 1.29	**81.04** ± 0.98	**59.32** ± 0.82

BNN provides a better estimation of the predictive uncertainty in the maximization phase

Experimental

Table 2: Classification accuracy (%) of our approach on PACS dataset [41] in comparison with the previously reported state-of-the-art results. Bold numbers indicate the best performance (two sets, one for each scenario engaging or forgoing domain identifications, respectively).

	DSN	L-CNN	MLDG	Fusion	MetaReg	Epi-FCR	AGG	HEX	PAR	ADA	ME-ADA
Domain ID	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗
Art	61.1	62.9	66.2	64.1	**69.8**	64.7	63.4	66.8	66.9	64.3	**67.1**
Cartoon	66.5	67.0	66.9	66.8	70.4	**72.3**	66.1	69.7	67.1	69.8	**69.9**
Photo	83.3	89.5	88.0	90.2	**91.1**	86.1	88.5	87.9	**88.6**	85.1	**88.6**
Sketch	58.6	57.5	59.0	60.1	59.2	**65.0**	56.6	56.3	62.6	60.4	**63.0**
Average	67.4	69.2	70.0	70.3	**72.6**	72.0	68.7	70.2	71.3	69.9	**72.2**

Experimental

Table 3: Average classification accuracy (%). Across several architectures, our approach obtains CIFAR-10-C and CIFAR-100-C corruption robustness that exceeds the previous state of the art by a large margin. Best performances are highlighted in bold.

		Standard	Cutout	CutMix	AutoDA	Mixup	AdvTrain	ADA	ME-ADA
CIFAR-10-C	AllConvNet	69.2	67.1	68.7	70.8	75.4	71.9	73.0	**78.2**
	DenseNet	69.3	67.9	66.5	73.4	75.4	72.4	69.8	**76.9**
	WideResNet	73.1	73.2	72.9	76.1	77.7	73.8	79.7	**83.3**
	ResNeXt	72.5	71.1	70.5	75.8	77.4	73.0	78.0	**83.4**
Average		71.0	69.8	69.7	74.0	76.5	72.8	75.1	**80.5**
CIFAR-100-C	AllConvNet	43.6	43.2	44.0	44.9	46.6	44.0	45.3	**51.2**
	DenseNet	40.7	40.4	40.8	46.1	44.6	44.8	45.2	**47.8**
	WideResNet	46.7	46.5	47.1	50.4	49.6	44.9	50.4	**52.8**
	ResNeXt	46.6	45.4	45.9	48.7	48.6	45.6	53.4	**57.3**
Average		44.4	43.9	44.5	47.5	47.4	44.8	48.6	**52.3**

Adversarial Self-Supervised Contrastive Learning

Minseon Kim[1], **Jihoon Tack**[1], **Sung Ju Hwang**[1,2]
KAIST[1], AITRICS[2]
{minseonkim, jihoontack, sjhwang82}@kaist.ac.kr

Introduction

■ Adversarial robustness

$$\underset{\theta}{\operatorname{argmin}}\, \mathbb{E}_{(x,y)\sim\mathbb{D}}\Big[\max_{\delta\in B(x,\epsilon)} \mathcal{L}_{\mathrm{CE}}(\theta, x+\delta, y)\Big]$$

■ Self-supervised contrastive learning

Maximize the agreement between different augmentations of the same instance in the learned latent space while minimizing the agreement between different instances.

SimCLR

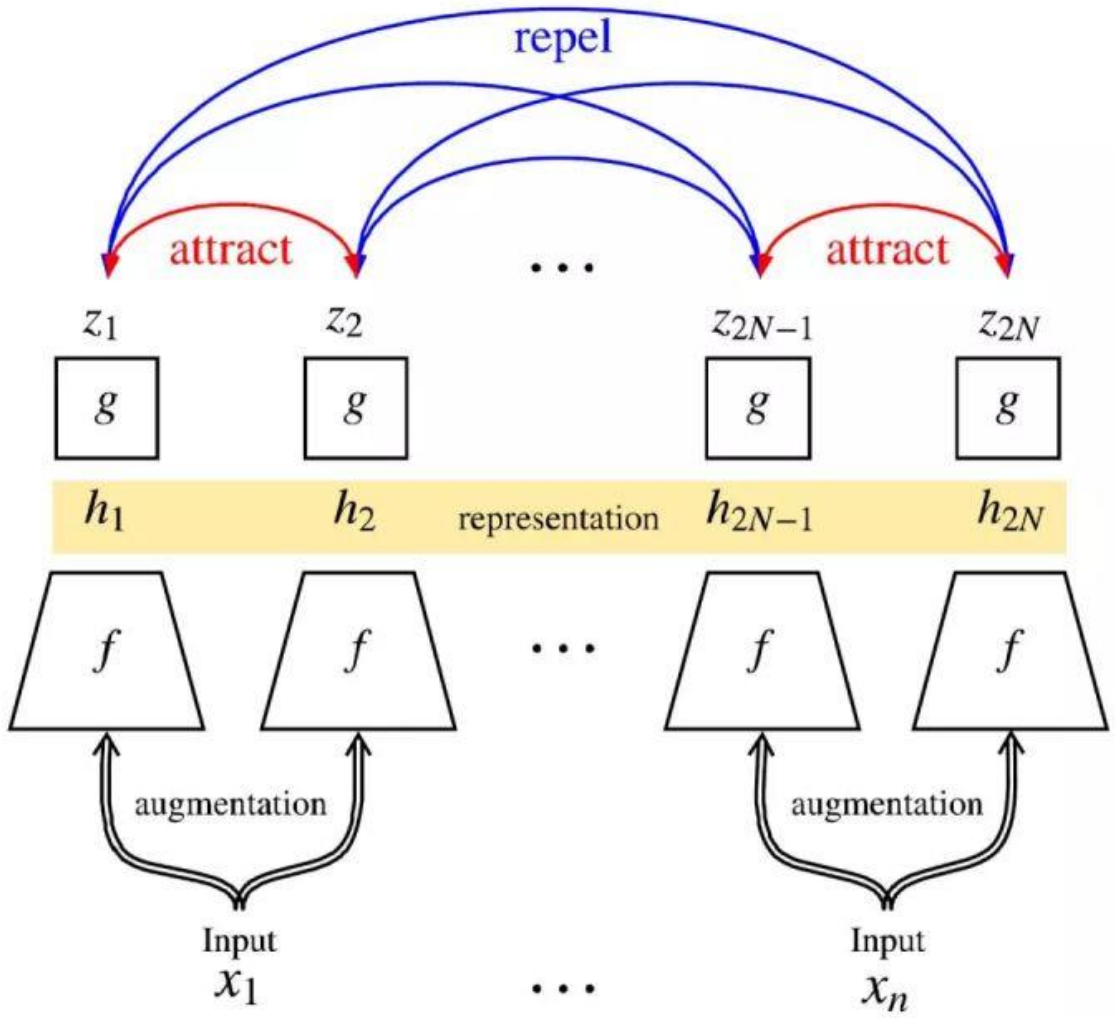
$$\mathcal{L}_{\mathrm{con},\theta,\pi}(x, \{x_{\mathrm{pos}}\}, \{x_{\mathrm{neg}}\})$$

$$:= -\log \frac{\sum_{\{\mathrm{z_{pos}}\}} \exp(\mathrm{sim}(\mathrm{z}, \{\mathrm{z_{pos}}\})/\tau)}{\sum_{\{\mathrm{z_{pos}}\}} \exp(\mathrm{sim}(\mathrm{z}, \{\mathrm{z_{pos}}\})/\tau) + \sum_{\{\mathrm{z_{neg}}\}} \exp(\mathrm{sim}(\mathrm{z}, \{\mathrm{z_{neg}}\})/\tau)},$$

SimCLR A Simple Framework for Contrastive Learning of Visual Representations

Algorithm 1 SimCLR's main learning algorithm.

input: batch size N, constant τ, structure of $f, g, \mathcal{T}$.
for sampled minibatch $\{\boldsymbol{x}_k\}_{k=1}^{N}$ **do**
  for all $k \in \{1, \dots, N\}$ **do**
    draw two augmentation functions $t \sim \mathcal{T}, t' \sim \mathcal{T}$
    # the first augmentation
    $\tilde{\boldsymbol{x}}_{2k-1} = t(\boldsymbol{x}_k)$
    $\boldsymbol{h}_{2k-1} = f(\tilde{\boldsymbol{x}}_{2k-1})$ # representation
    $\boldsymbol{z}_{2k-1} = g(\boldsymbol{h}_{2k-1})$ # projection
    # the second augmentation
    $\tilde{\boldsymbol{x}}_{2k} = t'(\boldsymbol{x}_k)$
    $\boldsymbol{h}_{2k} = f(\tilde{\boldsymbol{x}}_{2k})$ # representation
    $\boldsymbol{z}_{2k} = g(\boldsymbol{h}_{2k})$ # projection
  end for
  for all $i \in \{1, \dots, 2N\}$ and $j \in \{1, \dots, 2N\}$ **do**
    $s_{i,j} = \boldsymbol{z}_i^\top \boldsymbol{z}_j / (\|\boldsymbol{z}_i\| \|\boldsymbol{z}_j\|)$ # pairwise similarity
  end for
  define $\ell(i, j)$ **as** $\ell(i, j) = -\log \frac{\exp(s_{i,j}/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(s_{i,k}/\tau)}$
  $\mathcal{L} = \frac{1}{2N} \sum_{k=1}^{N} [\ell(2k-1, 2k) + \ell(2k, 2k-1)]$
  update networks f and g to minimize $\mathcal{L}$
end for
return encoder network $f(\cdot)$, and throw away $g(\cdot)$

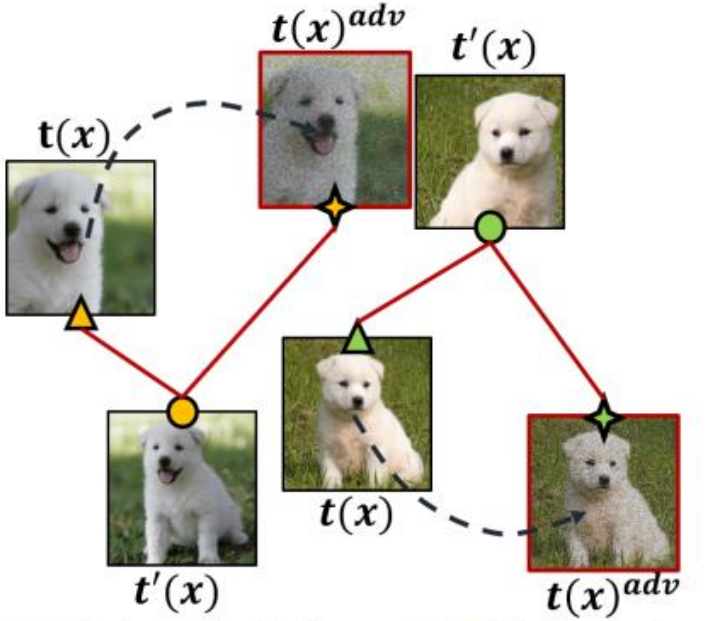


Introduction

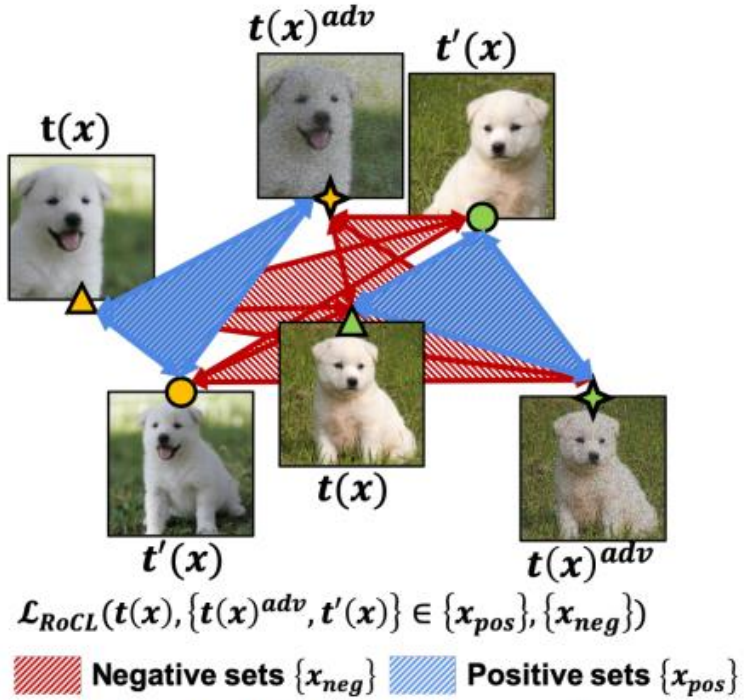
■ Robust Contrastive Learning (RoCL)

Train without a class label by using **instance-wise attacks**.

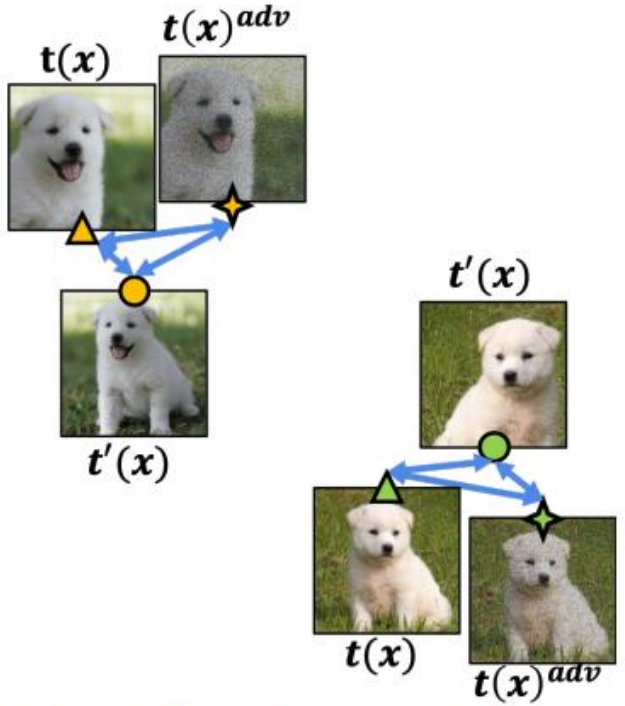
Maximize the similarity between a transformed example and the instance-wise adversarial example of another transformed example



(a) Instance-wise adversarial examples



(b) RoCL: Robust contrastive learning



(c) RoCL: Robust contrastive learning

Method

■ Instance-wise adversarial attacks

$$t(x)^{i+1} = \Pi_{B(t(x),\epsilon)}(t(x)^i + \alpha \mathtt{sign}(\nabla_{t(x)^i} \mathcal{L}_{\mathtt{con},\theta,\pi}(t(x)^i, \{t'(x)\}, \{t(x)_{\mathtt{neg}}\}))$$

where $t(x)$ and $t'(x)$ are transformed images with stochastic data augmentations $t, t' \sim \mathcal{T}$, and $\{t(x)_{\mathtt{neg}}\}$ are the negative instances for $t(x)$, which are examples of other samples x'.

■ Robust Contrastive Learning Objective

$$\underset{\theta,\pi}{\operatorname{argmin}} \mathbb{E}_{(x)\sim\mathbb{D}}[\max_{\delta \in B(t(x),\epsilon)} \mathcal{L}_{\mathtt{con},\theta,\pi}(t(x)+\delta, \{t'(x)\}, \{t(x)_{\mathtt{neg}}\})]$$

$$\mathcal{L}_{\mathtt{RoCL},\theta,\pi} := \mathcal{L}_{\mathtt{con},\theta,\pi}(t(x), \{t'(x), t(x)^{adv}\}, \{t(x)_{\mathtt{neg}}\})$$

$$\mathcal{L}_{\mathtt{total}} := \mathcal{L}_{\mathtt{RoCL},\theta,\pi} + \lambda \mathcal{L}_{\mathtt{con},\theta,\pi}(t(x)^{adv}, \{t'(x)\}, \{t(x)_{\mathtt{neg}}\})$$

Method

Algorithm 1 Robust Contrastive Learning (RoCL)

Input: Dataset $\mathbb{D}$, parameter of model θ, model f, parameter of projector π, projector g, constant λ

for all iter $\in$ number of training iteration **do**

for all $x \in$ minibatch $B = \{x_1, \ldots, x_m\}$ **do**

Generate adversarial examples from transformed inputs ▷ *instance-wise* attacks

$$t(x)^{i+1} = \Pi_{B(t(x),\epsilon)}(t(x)^i + \alpha \texttt{sign}(\nabla_{t(x)^i} \mathcal{L}_{\texttt{con},\theta,\pi}(t(x)^i, \{t'(x)\}, t(x)_{\texttt{neg}})))$$

end for

$\mathcal{L}_{\texttt{total}} = \frac{1}{N}\sum_{k=1}^{N}[\mathcal{L}_{\texttt{RoCL},\theta,\pi} + \lambda \mathcal{L}_{\texttt{con},\theta,\pi}(t(x)_k^{adv}, \{t'(x)_k\}, \{t(x)_{\texttt{neg}}\})]$ ▷ total loss

Optimize the weight θ, π over $\mathcal{L}_{total}$

end for

$$\mathcal{L}_{\texttt{RoCL},\theta,\pi} := \mathcal{L}_{\texttt{con},\theta,\pi}(t(x), \{t'(x), t(x)^{adv}\}, \{t(x)_{\texttt{neg}}\})$$

$$\mathcal{L}_{\texttt{total}} := \mathcal{L}_{\texttt{RoCL},\theta,\pi} + \lambda \mathcal{L}_{\texttt{con},\theta,\pi}(t(x)^{adv}, \{t'(x)\}, \{t(x)_{\texttt{neg}}\})$$

Method

■ Linear evaluation of RoCL

learns a linear layer $l(\cdot)$ on top of the fixed $f_\theta(\cdot)$ embedding layer with clean examples

$$\underset{\psi}{\operatorname{argmin}}\, \mathbb{E}_{(x,y)\sim\mathbb{D}}\Big[\max_{\delta\in B(x,\epsilon)} \mathcal{L}_{\mathtt{CE}}(\psi, x+\delta, y)\Big]$$

■ Transformation smoothed inference

predicts the class c by calculating expectation E over the transformation $t \sim T$ for a given input x

$$S(x) = \underset{c\in Y}{\operatorname{argmax}}\, \mathbb{E}_{t\sim\mathcal{T}}(l_c(f(t(x))) = c)$$

Experimental

white box attacks on ResNet18 and ResNet50 trained on the CIFAR-10

Train type	Method	ResNet18							ResNet50						
			seen		*unseen*					*seen*		*unseen*			
			ℓ_∞		ℓ_2		ℓ_1			ℓ_∞		ℓ_2		ℓ_1	
		A_{nat}	ϵ 8/255	16/255	0.25	0.5	7.84	12	A_{nat}	ϵ 8/255	16/255	0.25	0.5	7.84	12
Supervised	$\mathcal{L}_{CE}$	92.82	0.00	0.00	20.77	12.96	28.47	15.56	93.12	0.00	0.00	13.42	3.44	28.78	13.98
	AT[9]	81.63	44.50	14.47	72.26	**59.26**	**66.74**	55.74	84.03	46.76	17.63	72.98	58.78	65.28	52.45
	TRADES[2]	77.03	**48.01**	**22.55**	**68.07**	57.93	62.93	53.79	82.10	**53.49**	**25.18**	**73.01**	**61.94**	**65.48**	**54.52**
	TRADES*[2]	73.26	42.71	17.71	65.25	56.13	62.89	**55.95**	75.65	46.20	20.96	67.02	57.12	62.46	55.09
	SCL[32]	**94.05**	0.08	0.00	22.17	10.29	38.87	22.58	**95.02**	0.00	0.00	16.72	1.68	39.44	22.59
Self -supervised	SimCLR[12]	**91.25**	0.63	0.08	15.3	2.08	41.49	25.76	**92.69**	0.07	0.00	25.13	3.85	50.17	31.63
	RoCL	83.71	40.27	9.55	66.39	63.82	**79.21**	**76.17**	85.99	43.56	11.38	**70.87**	**67.59**	**82.65**	**80.02**
	RoCL+rLE	80.43	**47.69**	**15.53**	**68.30**	**66.19**	77.31	75.05	80.79	**45.33**	**16.85**	67.14	64.61	77.54	75.76
Self -supervised +finetune	Rot. Pretrained[18]	-	-	-	-	-	-	-	**85.66**$^+$	50.40$^+$	-	-	-	-	-
	RoCL+AT	80.26	40.77	**22.83**	68.64	56.25	65.16	56.07	82.72	50.60	18.83	72.12	**70.03**	**81.02**	**79.22**
	RoCL+TRADES	84.55	43.85	14.29	**73.01**	60.03	68.25	58.04	85.41	45.68	**21.21**	74.06	59.60	65.37	53.54
	RoCL+AT+SS	**91.34**	**49.66**	14.44	70.75	**61.55**	**83.08**	**81.18**	84.67	**52.44**	19.53	**76.61**	66.38	72.76	64.56

Experimental

Performance of RoCL against black box attacks on the CIFAR-10 dataset

Target \ Source	ResNet18							
	8/255				16/255			
	AT	TRADES	RoCL(PGD)	RoCL(*inst.*)	AT	TRADES	RoCL(PGD)	RoCL(*inst.*)
AT [9]	-	**77.48**	69.83	47.25	-	**63.87**	48.99	47.42
TRADES [2]	60.73	-	64.81	46.22	41.87	-	48.07	45.73
RoCL	**66.76**	77.33	-	-	**41.97**	62.98	-	-

Table 4: Results of transfer learning across the CIFAR-10 and CIFAR-100 datasets with ResNet18. We compare against adversarial transfer learning results from [41], with a larger WRN 32-10 [42] architecture. $^+$ is the reported performance from [41].

source	target	Method	A_{nat}	ℓ_∞
CIFAR-100	CIFAR-10	Transfer$^+$[41]	72.05	17.70
		RoCL	**73.93**	**18.62**
CIFAR-10	CIFAR-100	Transfer$^+$[41]	41.59	11.63
		RoCL	**45.84**	**15.33**