Learn2Perturb: an End-to-end Feature Perturbation Learning to Improve Adversarial Robustness

- DNNs' **vulnerability to adversarial attacks** limits their widespread deployment for safety-critical applications.
- Improve adversarial robustness Introduction of perturbations during the training process.
- Drawback: need fixed, pre-defined perturbations and require significant hyperparameter tuning
- Solution : an end-to-end feature perturbation learning approach

Adversarial attack algorithms

- Fast Gradient Sign Method (FGSM) – 利用Loss对Input的 gradients

$$x' = x + \epsilon \cdot \textbf{sign}\Big(\nabla_x \mathcal{L}(f_W(x), x)\Big)$$

- Projected gradient descent (PGD)

$$x_{t+1} = \textbf{bound}_{l_p}(FGSM(x_t), x_0),$$

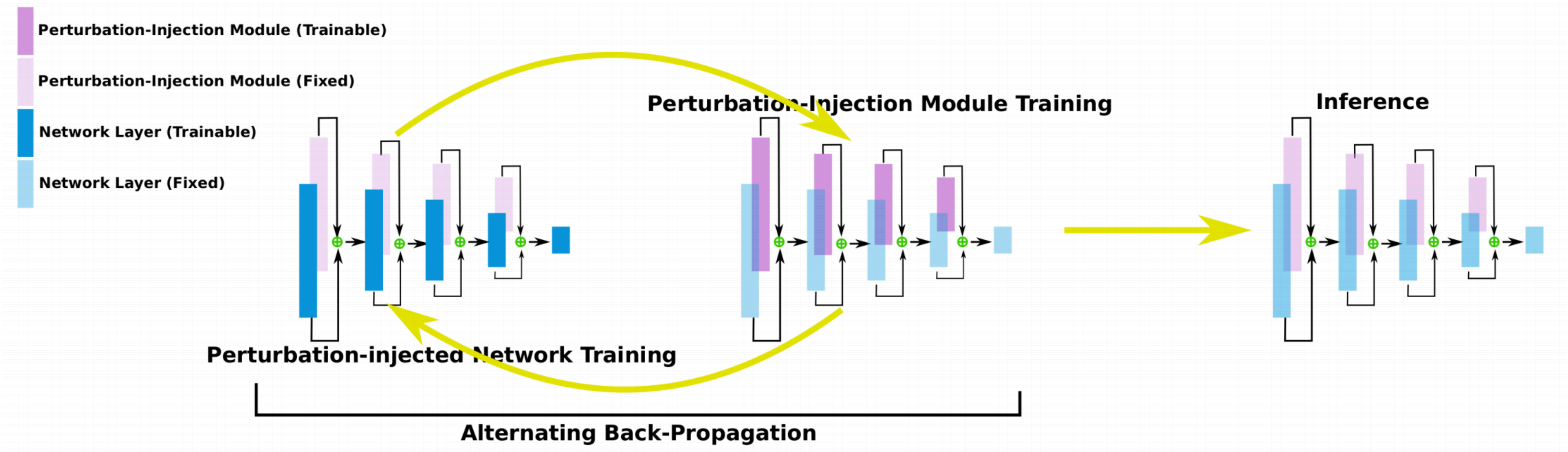
- C&W attack

$$min\Big[||\delta||_p + c \cdot f(x+\delta)\Big] \quad s.t. \quad x+\delta \in [0,1]^n$$

Motivation

- Require pre-defined perturbations and require significant hyperparameter tuning
- How to select the distribution of the noise in the neural networks?

- Learning the model parameters and the perturbation distributions of the perturbation-injection modules in an end-to end learning framework.



1. Perturbation-injected network training: the network parameters are trained by gradient descent while the proposed perturbation-injection modules add layer-wise noise to the feature maps

2. Perturbation-injection module training: the parameters of the perturbation-injection modules are updated via gradient descent and based on the regularization term added to the network loss function, while network parameters are fixed.

- Network : $Y \sim P(X; W, \theta)$
  - θ : noise parameters
- $P_l(X_l; W_l, \theta_l) = f_l(X_l; W_l) + Q(\theta_l)$
  - activation of layer l+ noise distribution with parameter following Gaussian distribution
  - $Q(\theta_l)$= $\theta_l \mathrm{N}(0,1)$
  - $P_l(X_l; W_l, \theta_l) \approx \mathrm{N}(f_l(X_l; W_l), \theta_l)$
- Loss Function :

$$\arg\min_{W,\theta}\left[\mathcal{L}\Big(P(X; W, \theta), T\Big) + \gamma \cdot g(\theta)\right]$$

$$g(\theta) = -\frac{\theta^{1/2}}{\tau}$$

Input : Training set D =
$\{(x_i, t_i),\ \ i = 1, \ldots, n\}$
Number of training epochs, I
θ_{min}, the lower bound for θ
θ_0, initial values for θ
Learning rate, lr, and constant γ

Output : Learned parameters W
Learned noise distributions $Q(\theta)$

for $t \leftarrow 1$ *to* I **do**

Perturbation-injected training: update W based on the loss function $\mathcal{L}(\cdot)$ Eq. (7) while θ is fixed

$W^t \leftarrow W^{t-1} - lr \cdot \nabla_W \mathcal{L}\Big(P(X; W^{t-1}, \theta^{t-1}), T\Big)$

Perturbation-injection module training: update θ based on Eq. (7) while W is fixed

$\theta^t \leftarrow \theta^{t-1} - lr \cdot \nabla_\theta \mathcal{L}\Big(P(X; W^{t-1}, \theta^{t-1}), T\Big) - \gamma \cdot \nabla_\theta g(\theta^{t-1})$

Values of θ^t smaller than θ_{min} are projected to θ_{min}

end

Results On Cirfar 10

Table 1. Evaluating the effectiveness of the proposed perturbation-injection modules by comparing against adversarial training algorithm (Vanilla) within the proposed framework and its variation (Learn2Perturb-R).

		No defense			Vanilla[24]			Learn2Perturb-R			Learn2Perturb		
Model	**#Parameter**	**Clean**	**PGD**	**FGSM**	**Clean**	**PGD**	**FGSM**	**Clean**	**PGD**	**FGSM**	**Clean**	**PGD**	**FGSM**
ResNet-V1(20)	269,722	92.1	0.0±0.0	14.1	83.8	39.1±0.1	46.6	81.15±0.02	50.23±0.14	55.89±0.04	83.62±0.02	51.13±0.08	58.41±0.07
ResNet-V1(56)	853,018	93.3	0.0±0.0	24.2	86.5	40.1±0.1	48.8	82.35±0.03	53.30±0.10	58.71±0.04	84.82±0.04	54.84±0.10	61.53±0.04
ResNet-V2(18)	11,173,962	95.2	0.1±0.0	43.1	85.46	43.9±0.0	52.5	82.46±0.17	53.33±0.12	59.09±0.17	85.30±0.09	56.06±0.16	62.43±0.06

network sizes and capacities. Result are reported by standard deviation because of the randomness involved in these methods.

		PNI [14]			Adv-BNN [22]			Learn2Perturb		
Model	**#Parameter**	**Clean**	**PGD**	**FGSM**	**Clean**	**PGD**	**FGSM**	**Clean**	**PGD**	**FGSM**
ResNet-V1(20)	269,722	84.90±0.1	45.90±0.1	54.50±0.4	65.76±5.92	44.95±1.21	51.58±1.49	83.62±0.02	51.13±0.08	58.41±0.07
ResNet-V1(32)	464,154	85.90±0.1	43.50±0.3	51.50±0.1	62.95±5.63	54.62±0.06	50.29±2.70	84.19±0.06	54.62±0.06	59.94±0.11
ResNet-V1(44)	658,586	84.70±0.2	48.50±0.2	55.80±0.1	76.87±0.24	54.62±0.06	58.55±0.49	85.61±0.01	54.62±0.06	61.32±0.13
ResNet-V1(56)	853,018	86.80±0.2	46.30±0.3	53.90±0.1	77.20±0.02	54.62±0.06	57.88±0.02	84.82±0.04	54.62±0.06	61.53±0.04
ResNet-V1(20)[1.5×]	605,026	86.00±0.1	46.70±0.2	54.50±0.2	65.58±0.42	28.07±1.11	36.11±1.29	85.40±0.08	53.32±0.02	61.10±0.06
ResNet-V1(20)[2×]	1,073,962	86.20±0.1	46.10±0.2	54.60±0.2	79.03±0.04	53.46±0.06	58.30±0.14	85.89±0.10	54.29±0.02	61.61±0.05
ResNet-V1(20)[4×]	4,286,026	87.70±0.1	49.10±0.3	57.00±0.2	82.31±0.03	52.61±0.12	59.01±0.04	86.09±0.05	55.75±0.07	61.32±0.02
ResNet-V2(18)	11,173,962	87.21±0.00	49.42±0.01	58.06±0.02	82.15±0.06	53.62±0.06	60.04±0.01	85.30±0.09	56.06±0.08	62.43±0.06

Results On Cirfar 100

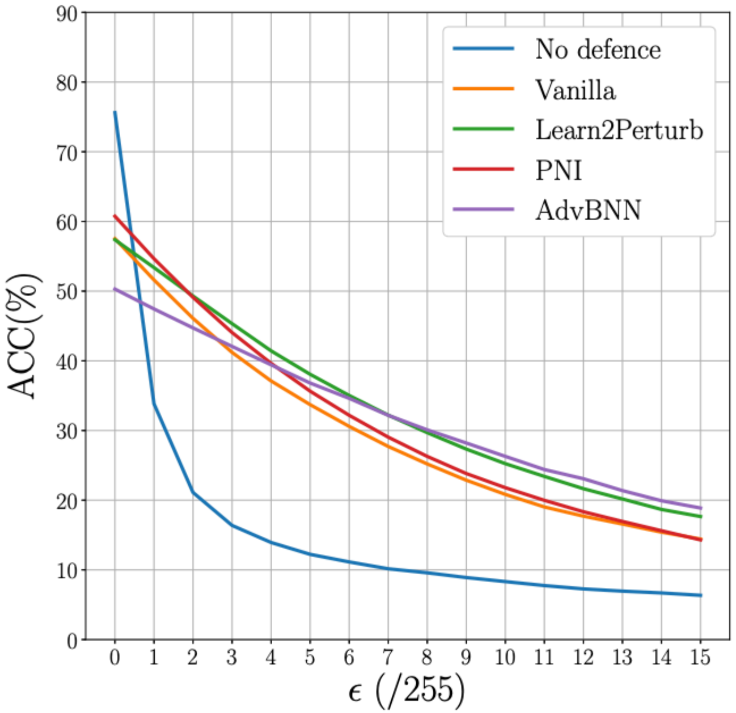


Figure 5. FGSM attack on CIFAR-100 with different epsilons for the l_∞ ball on ResNet-V2(18).

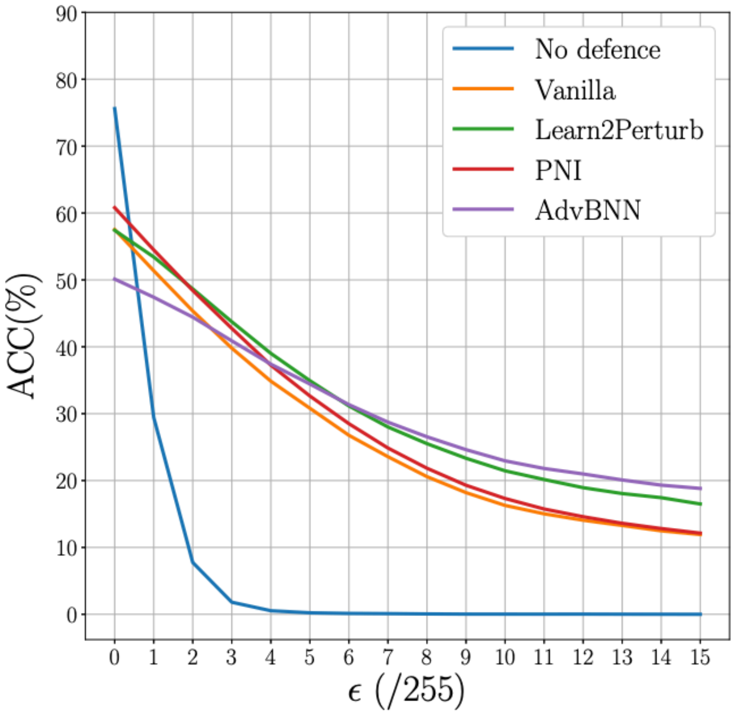


Figure 6. PGD attack on CIFAR-100 with different epsilons for the l_∞ ball on ResNet-V2(18).