Robustness of Deep Neural Networks
against white-box adversarial attacks

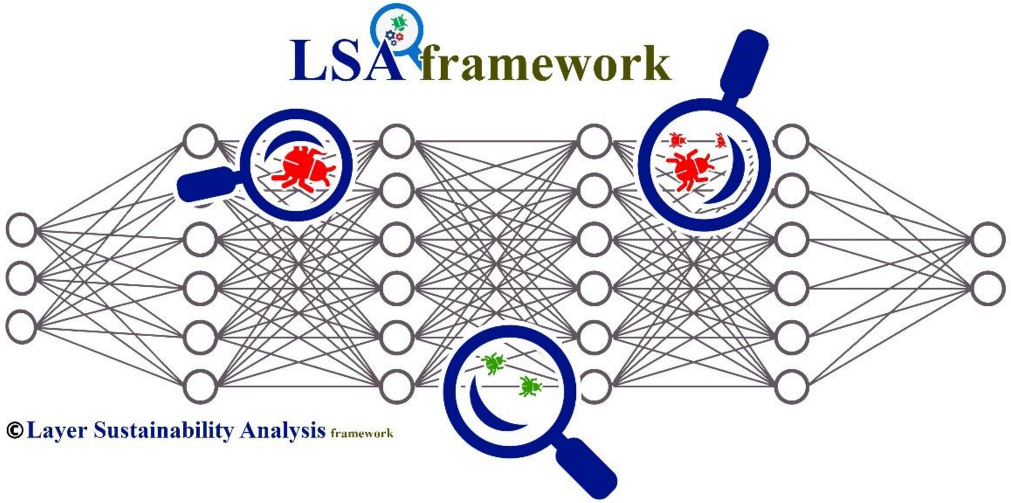
Mohammad Khalooei , Mohammad Mehdi Homayounpour , Maryam Amirmazlaghani

Outlines

- Review on prev. presentations

- Paper review

  - **L**ayer-wise **R**egularized **A**dversarial **T**raining (AT-LR) using **L**ayers **S**ustainability **A**nalysis (LSA) framework

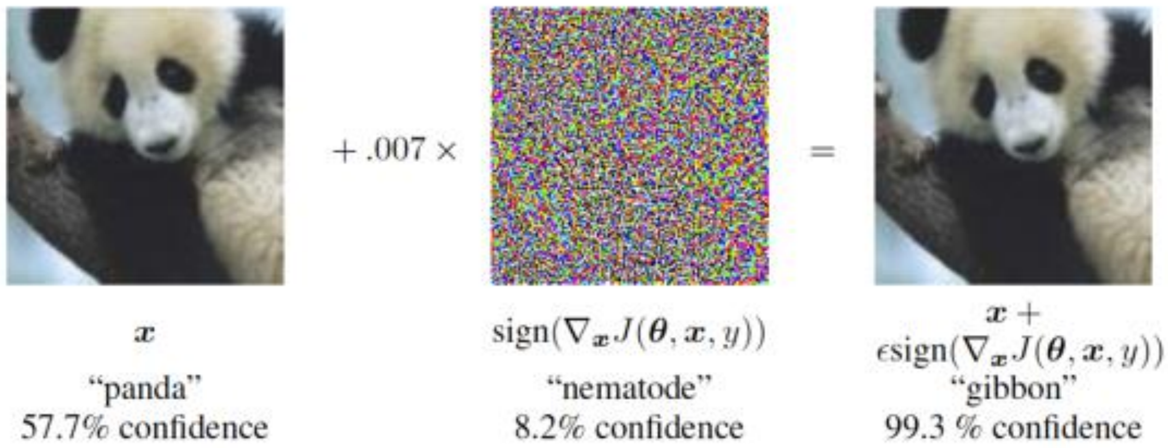


- **Attack**

$$x^* = x + \epsilon \cdot \mathrm{sign}\left(\nabla_x J(x, y)\right)$$

$$\mathrm{L}_\infty \text{ norm } \|x^* - x\|_\infty \leq \varepsilon$$

$$\arg\min_\delta \|\delta\|_p + c \cdot J(x+\delta, y)$$
$$\text{s.t. } x + \delta \in [0,1]^n$$



- **Defense**
  - Data manipulation
  - Architecture manipulation
  - Loss manipulation

$$\min_\theta \rho(\theta), \quad \text{where} \quad \rho(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\max_{\delta\in\mathcal{S}} L(\theta, x+\delta, y)\right]$$

- **Adversarial Training (AT)**

A method for learning robust deep networks
It is typically assumed to be **more expensive** than **traditional training**
constructing adversarial examples via a first-order method like projected gradient decent (PGD)

(x_i, y_i) Dataset

ℓ Loss function

f_θ Network

θ Parameters

Δ Threat model

$$\min_\theta \sum_i \max_{\delta \in \Delta} \ell(f_\theta(x_i + \delta), y_i)$$

robust optimization problem

$$\Delta = \{\delta : \|\delta\|_\infty \leq \epsilon\} \quad \epsilon > 0$$

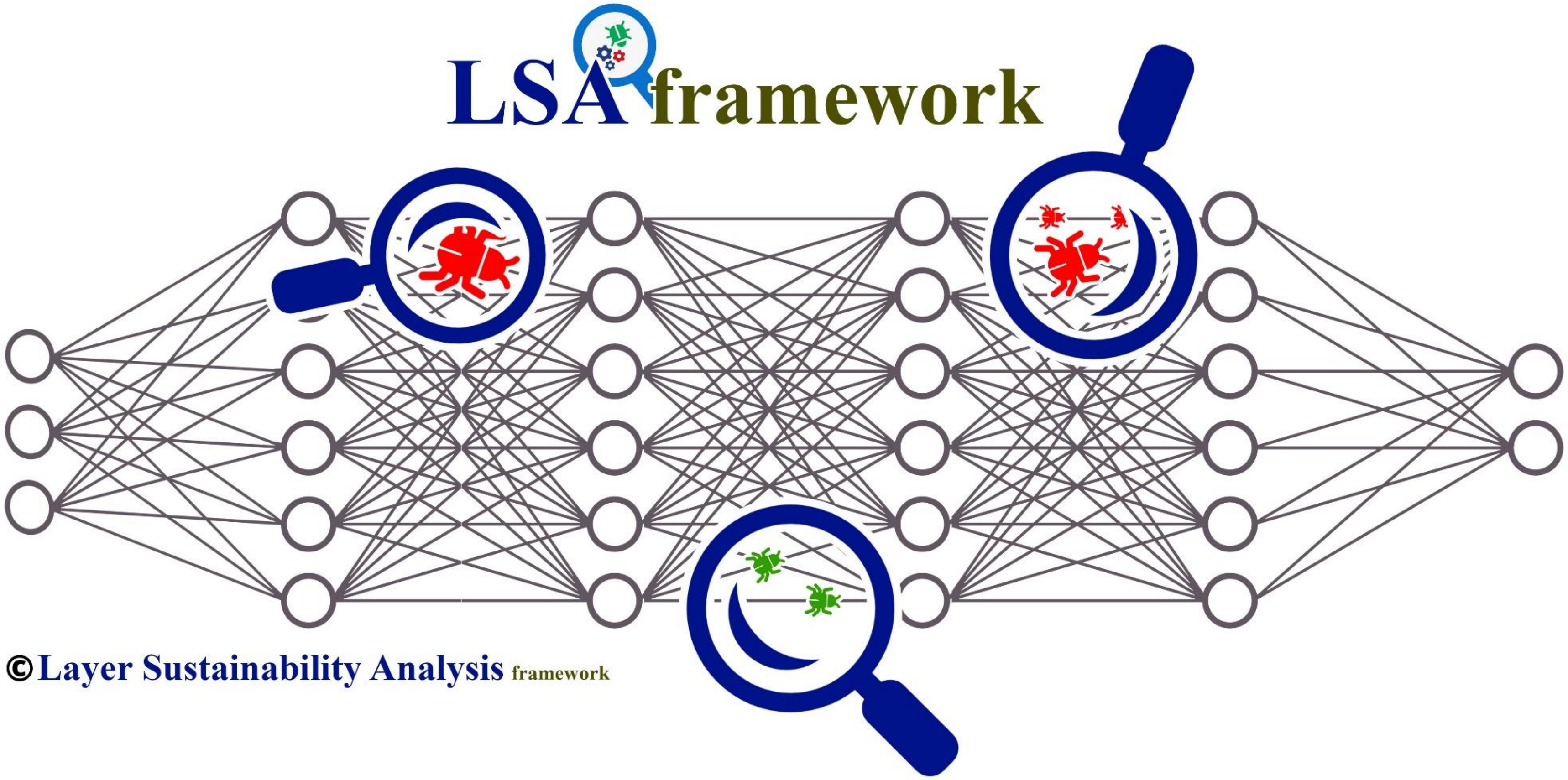
Madry et al. (2017)

$$\delta^\star = \epsilon \cdot \mathrm{sign}(\nabla_x \ell(f(x), y))$$

Goodfellow et al. (2014)

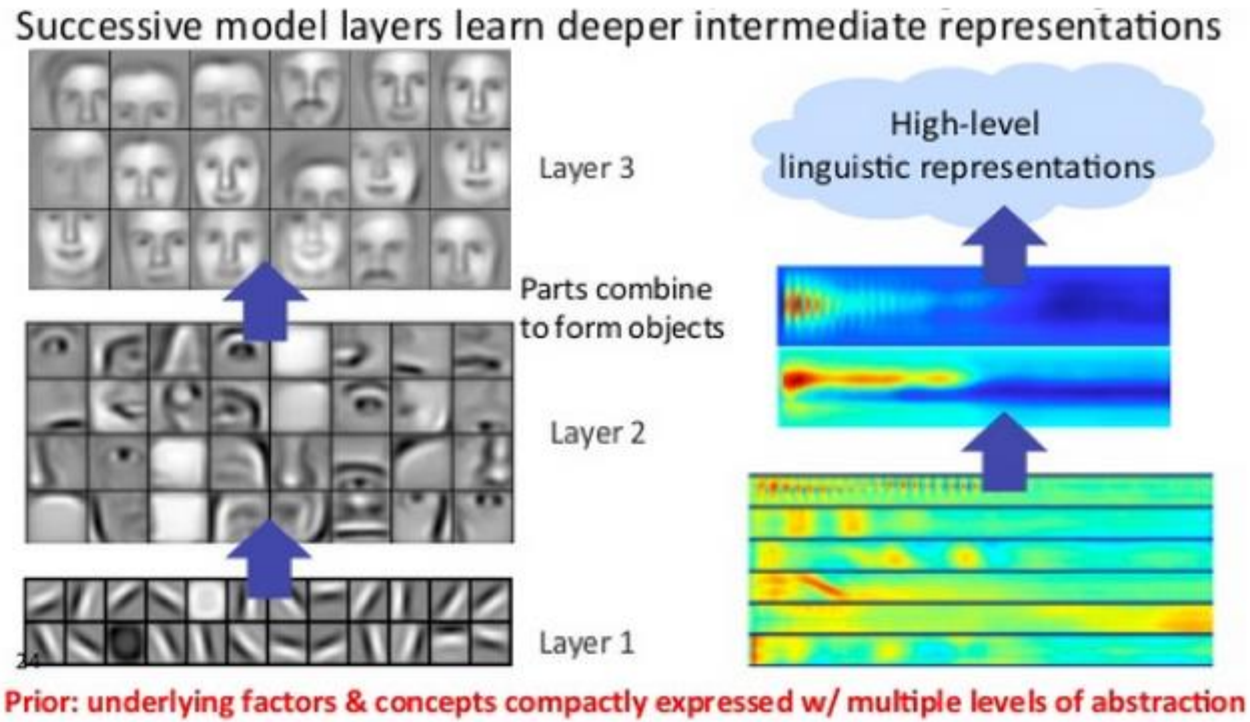
A better approximation of the **inner maximization** is to take **multiple**

- Layer-wise Regularized Adversarial Training using **Layers Sustainability Analysis** (LSA) framework

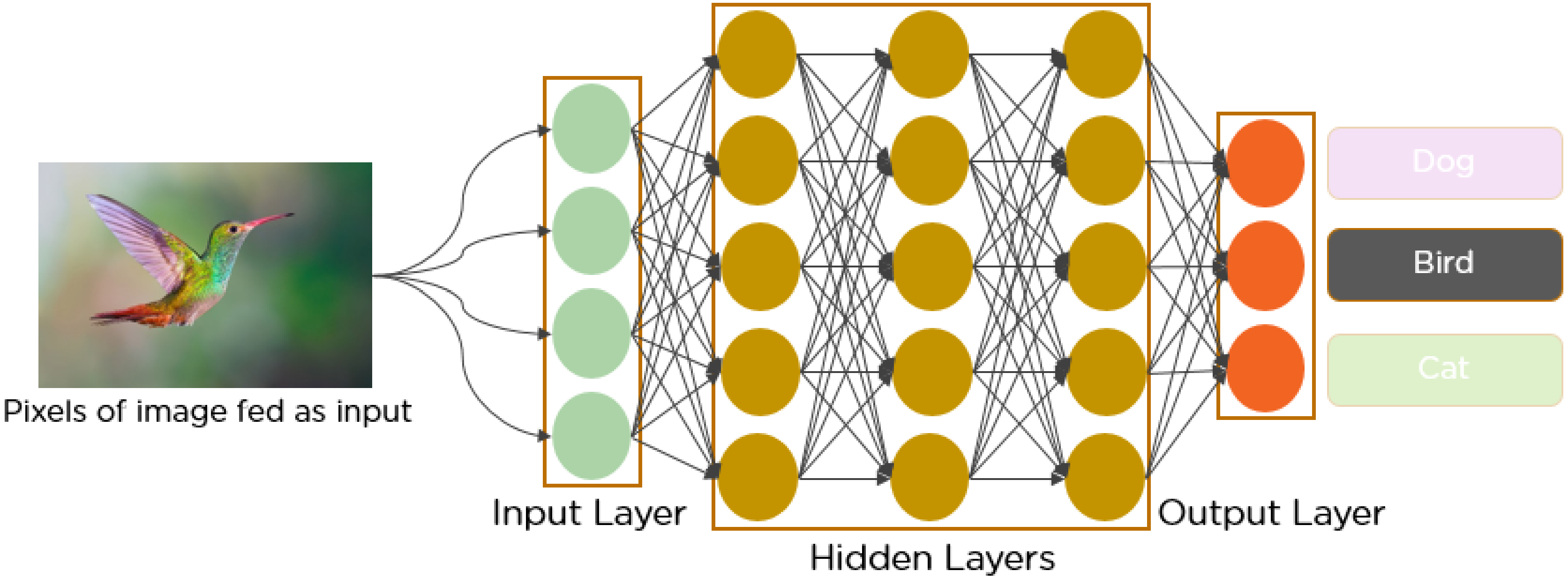


https://github.com/khalooei/LSA

Latest work : Motivation



https://wiki.pathmind.com/neural-network



Pixels of image fed as input
Input Layer
Hidden Layers
Output Layer
Dog
Bird
Cat

Latest work : Motivation

- Layer-wise Regularized Adversarial Training
  using **Layers Sustainability Analysis** (LSA) framework

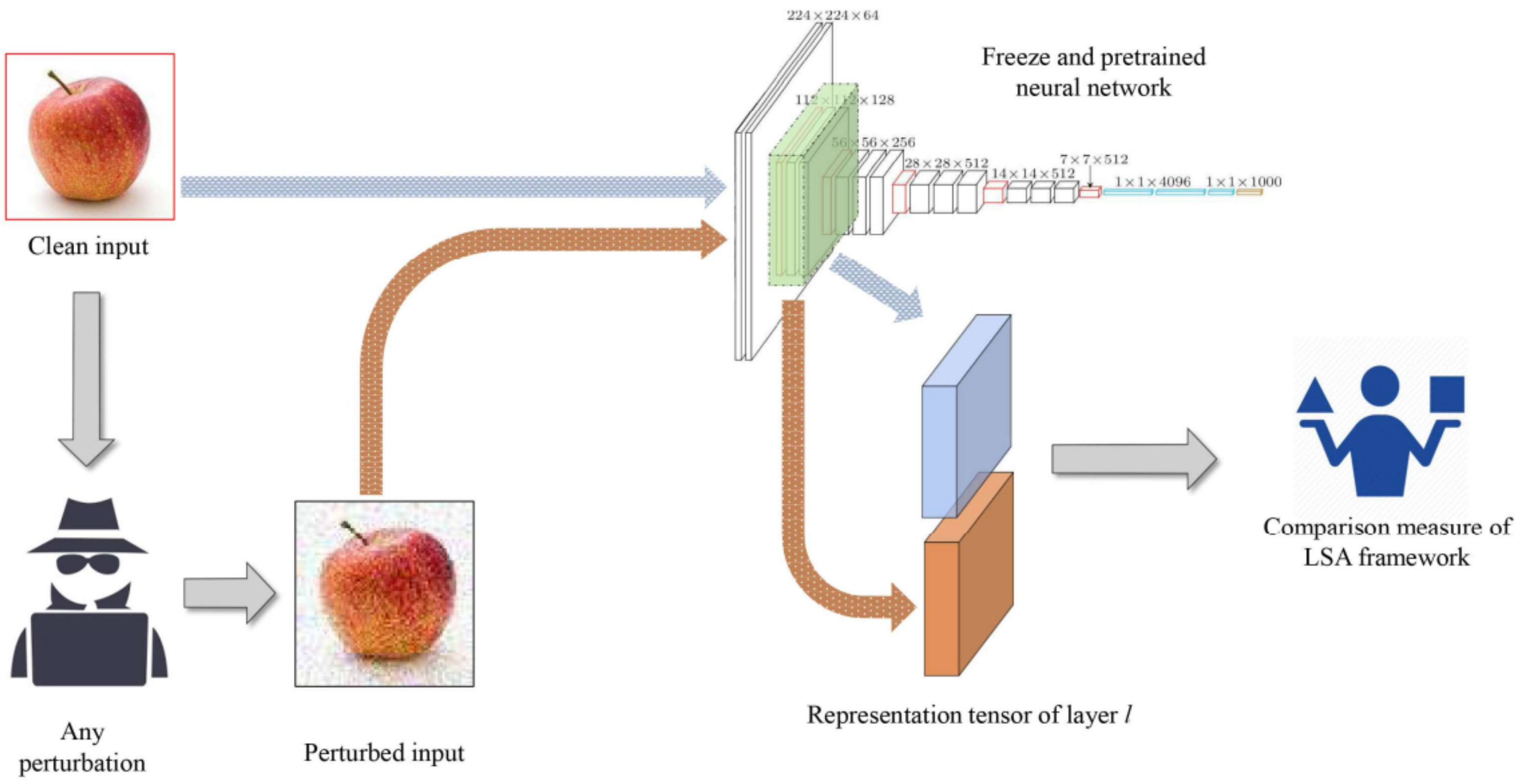
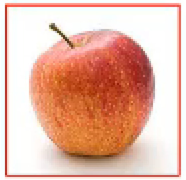
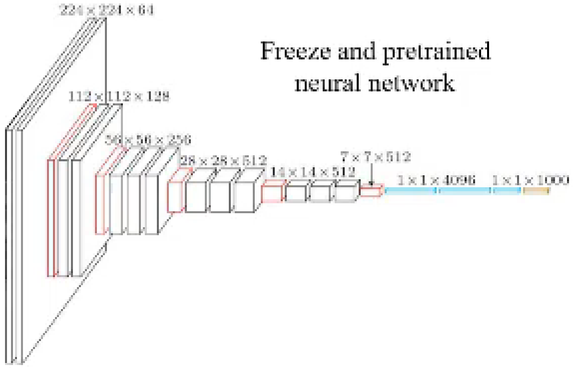


Fig. 1. Diagram of the Layer Sustainability Analysis (LSA) framework

- Layer-wise Regularized Adversarial Training using **Layers Sustainability Analysis** (LSA) framework



Clean input



Layer Sustainability Analysis (LSA) framework

- Layer-wise Regularized Adversarial Training using **Layers Sustainability Analysis** (LSA) framework

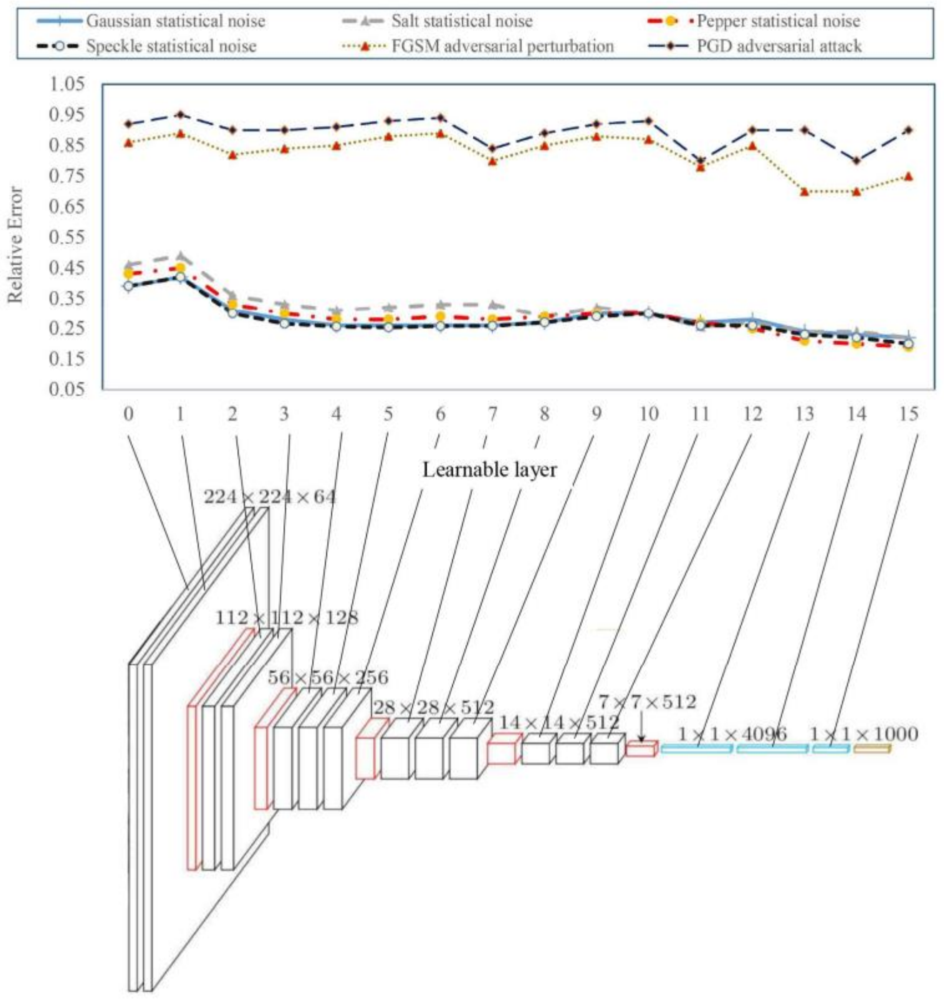


Fig. 2. Comparison measure values for corresponding layers of the VGG network in the proposed LSA framework

Lipschitz condition

Definition: function $f(t, y)$ satisfies a **Lipschitz condition** in the variable y on a set $D \subset \mathbf{R}^2$ if a constant $L > 0$ exists with

$$|f(t, y_1) - f(t, y_2)| \leq L\,|y_1 - y_2|\,,$$

whenever $(t, y_1), (t, y_2)$ are in D. L is Lipschitz constant.

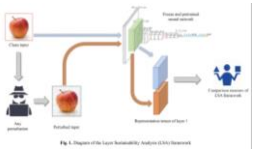
$$\|F(x_1) - F(x_2)\| \leq \psi\|x_1 - x_2\| \qquad s.t.\ x_1, x_2 \subset X.$$

Latest work: Lipschitz condition

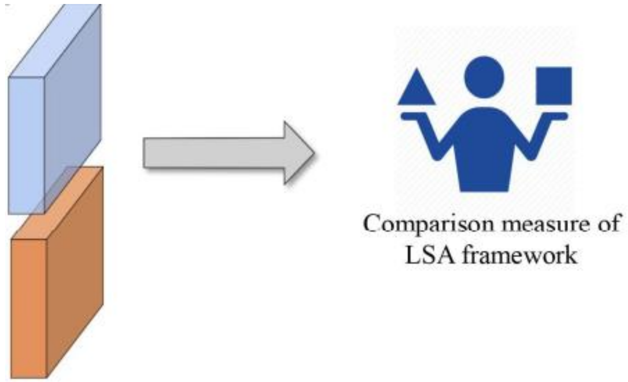
$$\|F(x_1) - F(x_2)\| \leq \psi\|x_1 - x_2\| \qquad s.t.\ x_1, x_2 \subset X.$$

$$\|\phi_l(x_1) - \phi_l(x_2)\| \leq \psi\|x_1 - x_2\|.$$

$$\|\phi_l(x) - \phi_l(\hat{x})\| \leq \psi\|x - \hat{x}\|.$$



$$CM\left(\phi_l(x), \phi_l(\hat{x})\right) = \frac{\|\phi_l(x) - \phi_l(\hat{x})\|_F}{\|\phi_l(x)\|_F},$$



$$\mu = \frac{1}{M \times Ly} \sum_{m=0}^{M-1} \sum_{l=0}^{Ly-1} CM\left(\phi_l(x_m), \phi_l(\hat{x}_m)\right),$$

$$\sigma = \sqrt{\frac{1}{M \times Ly} \sum_{m=0}^{M-1} \sum_{l=0}^{Ly-1} \left(CM\left(\phi_l(x_m), \phi_l(\hat{x}_m)\right) - \mu\right)^2},$$

Latest work: proposed method

Algorithm 1. Algorithm to find the most vulnerable layers in the layer sustainability analysis (LSA) framework

Algorithm 1 . Algorithm to find the most vulnerable layers in the layer sustainability analysis (LSA) framework

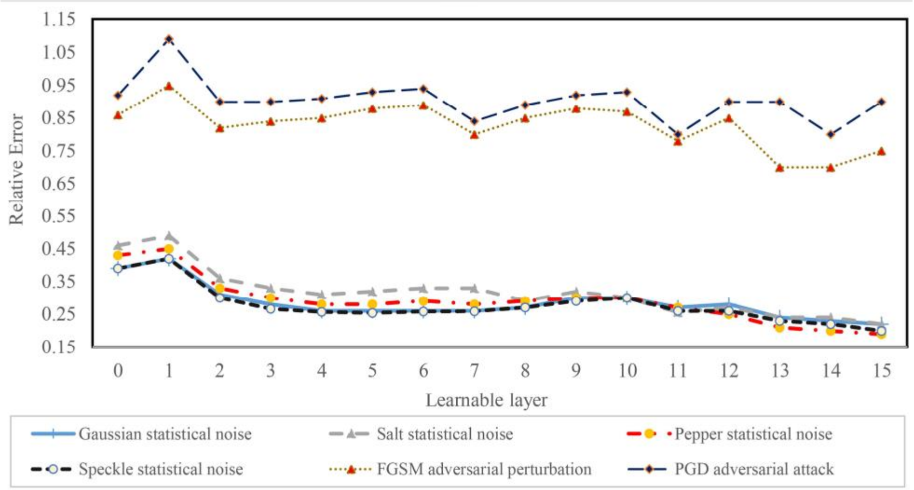
Input: Output representation tensors $\phi_l(x)$ and $\phi_l(\hat{x})$ of layer l for clean input x and the corresponding perturbed sample $\hat{x}$

Output: list of most vulnerable layers

Algorithm steps:

for trained model *m,* constant η, average μ and standard deviation σ as calculated in equation (12).

1. LSA_MVL_list = []
2. for *l* in range(0, *Ly*)
3. $\text{if}\big(CM\big(\phi_l(x), \phi_l(\hat{x})\big) - \mu\big) > \eta\ \sigma$
4. LSA_MVL_list.append(*l*)
5. LSA_MVL_list = sort(LSA_MVL_list)



Algorithm 2. Layer sustainability analysis (LSA) framework's algorithm

Algorithm 2 Layer Sustainability Analysis (LSA) framework

Input: model m, train data D_{train}, test data D_{test}, attack method and its parameters

Output: list of vulnerable layers

Algorithm steps:

1. Standard training of model m using D_{train} samples
2. Execute attack on the trained model m and obtain adversarial examples by perturbing D_{test} samples
3. Run Algorithm 1 and tune the proper cut-off threshold η of Algorithm 1 to find out the MVL list
4. Return the MVL list of Algorithm 1

Latest work: proposed method

Algorithm 3 Layer-wise Regularized adversarial training (AT-LR) algorithm

Input: X as inputs, Y as the corresponding targets, F_θ as a model with parameters θ, an LSA MVL list from

Algorithm 2

Output: a robust model (based on AT-LR approach)

Algorithm steps:

1. Initialize θ
2. **for** epoch =1 … N **do**
3. **for** minibatch $(x, y) \subset (X, Y)$ **do**
4. $\hat{x} \leftarrow \text{AdversarialAttack}(F_\theta, x, y)$
5. $\theta \leftarrow \min\{\, J(\theta, \hat{x}, y) + LR(\theta, x, \hat{x}, y)\,\}$
6. **end for**
7. **end for**

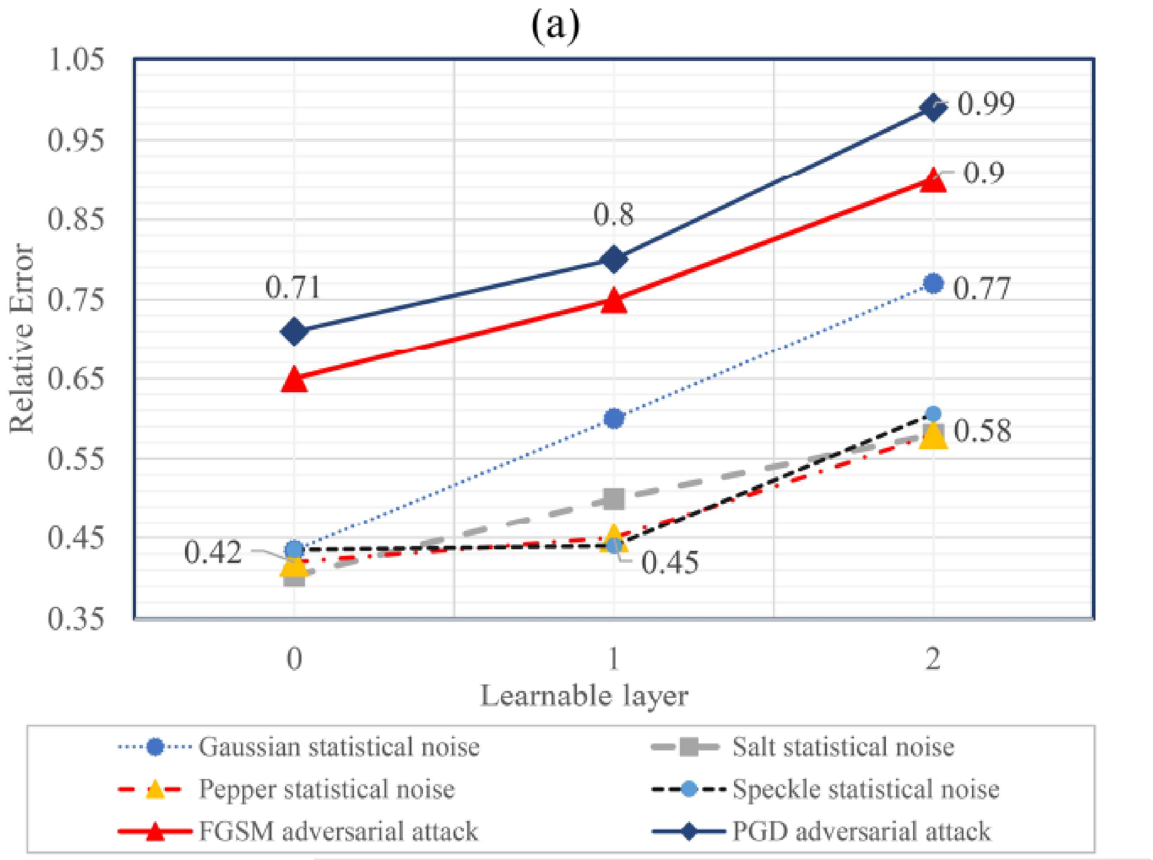
Latest work: Experiment results

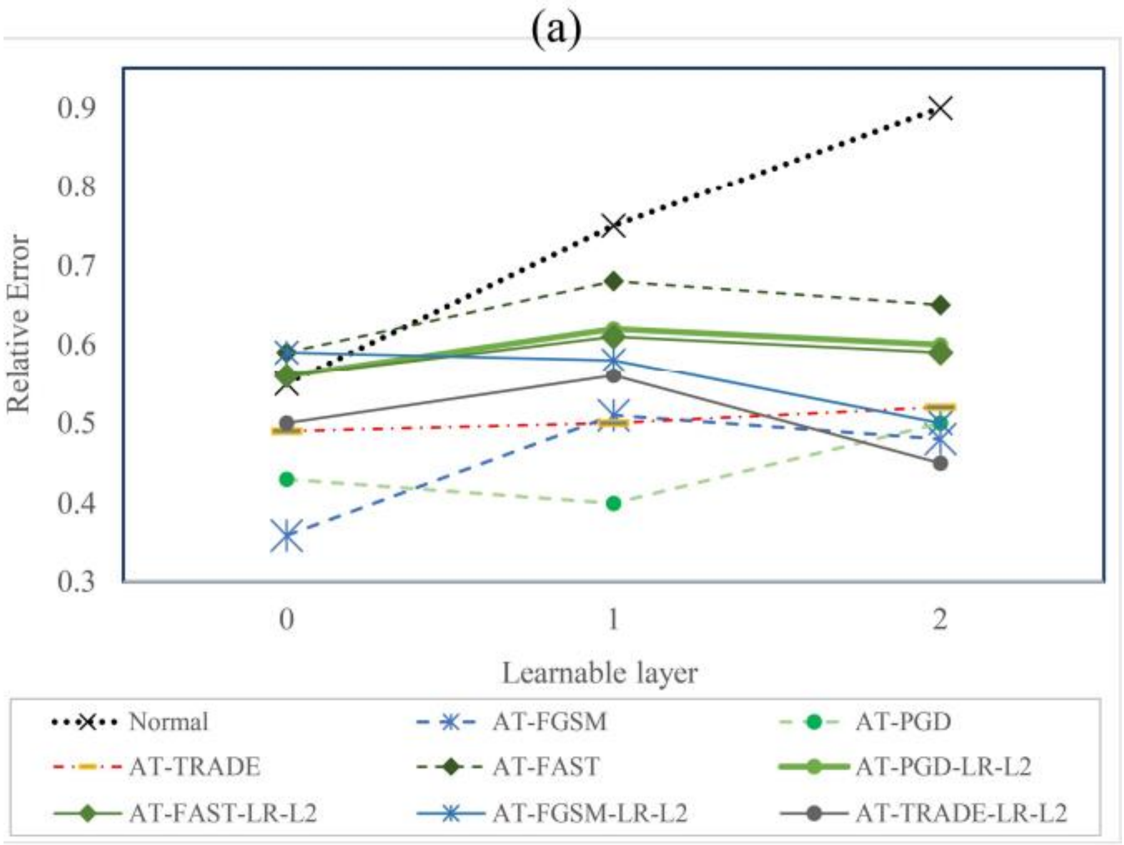
- Experiment architectures

Table 1 Experiment architectures A, B, C, and D

Model A	$linear(100) \Longrightarrow ELU \Longrightarrow Linear(100) \Longrightarrow ELU \Longrightarrow Linear(100) \Longrightarrow ELU \Longrightarrow Linear(1)$
Model B	$Conv2D\big(16,(5\times 5)\big) \Longrightarrow ReLU() \Longrightarrow Conv2D\big(32,(5\times 5)\big) \Longrightarrow ReLU() \Longrightarrow MaxPool2D(2,2)$ $\Longrightarrow Conv2D(64,(5\times 5)) \Longrightarrow ReLU() \Longrightarrow MaxPool2D(2,2) \Longrightarrow Linear(100) \Longrightarrow ReLU() \Longrightarrow Linear(10)$
Model C	VGG-19 architecture [63]
Model D	WideResNet [68]

- Moon Dataset // model A (MLP)





Latest work: Experiment results

- Moon Dataset // model A (MLP)

Table 2. Evaluation of model *A* on Moon dataset with different loss functions against FGSM adversarial attack.

Training type	Accuracy of model A against FGSM with different epsilon values						
	0	0.1	0.2	0.3	0.4	0.5	R&G Score
Normal	**97.07**	**93.63**	82.5	76.83	63.79	52.35	466.17
AT-FGSM	**95.86**	**91.98**	**87.88**	**81.33**	70.54	58.69	**486.28**
AT-PGD	86.5	83.89	80.87	76.61	**72.57**	60.02	460.46
AT-TRADE	**96.17**	**93.06**	85.78	74.2	68.62	59.68	477.51
AT-FAST	**94.32**	89.72	84.91	78.35	70.18	61.65	**479.13**
AT-FGSM-LR-L0	92.81	90.25	85.36	80.74	68.21	58.74	476.11
AT-FGSM-LR-L1	91.87	89.77	**86.95**	**81.51**	69.15	59.59	478.84
AT-FGSM-LR-L2	93.98	**93.01**	**89.12**	**84.01**	**75.15**	63.25	**498.52**
AT-PGD-LR-L0	87.02	84.02	80.64	76.22	71.12	**65.87**	464.89
AT-PGD-LR-L1	86.2	82.94	79.23	75.16	70.5	65.04	459.07
AT-PGD-LR-L2	88.78	86.93	81.25	77.01	**72.81**	**66.58**	473.36
AT-FAST-LR-L0	86.86	83.85	80.33	76.34	71.34	**65.79**	464.51
AT-FAST-LR-L1	87.91	85.43	82.44	77.38	70.01	62.25	465.42
AT-FAST-LR-L2	92.91	90.31	**86.26**	**81.53**	**75.09**	**67.65**	**493.75**
AT-TRADE-LR-L0	86.84	82.38	80.55	76.37	71.58	64.07	461.79
AT-TRADE-LR-L1	87.25	84.22	81.25	76.98	71.14	63.25	464.09
AT-TRADE-LR-L2	**96.67**	**93.2**	**86.56**	**81.25**	**79.5**	**68.65**	**505.83**

- Moon Dataset // model A (MLP)

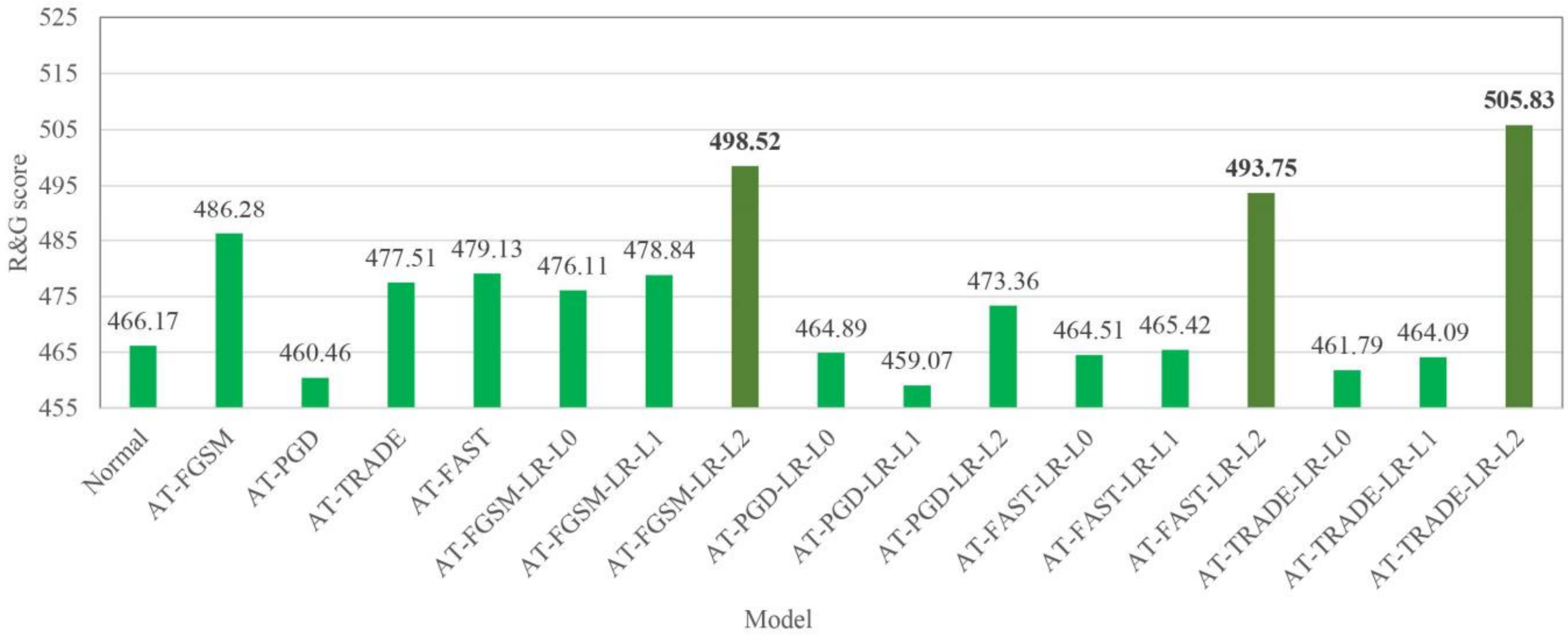


Fig. 6. Comparison of R&G score results for different loss functions of architecture A on Moon dataset with different training loss functions.

- Moon Dataset // model A (MLP)

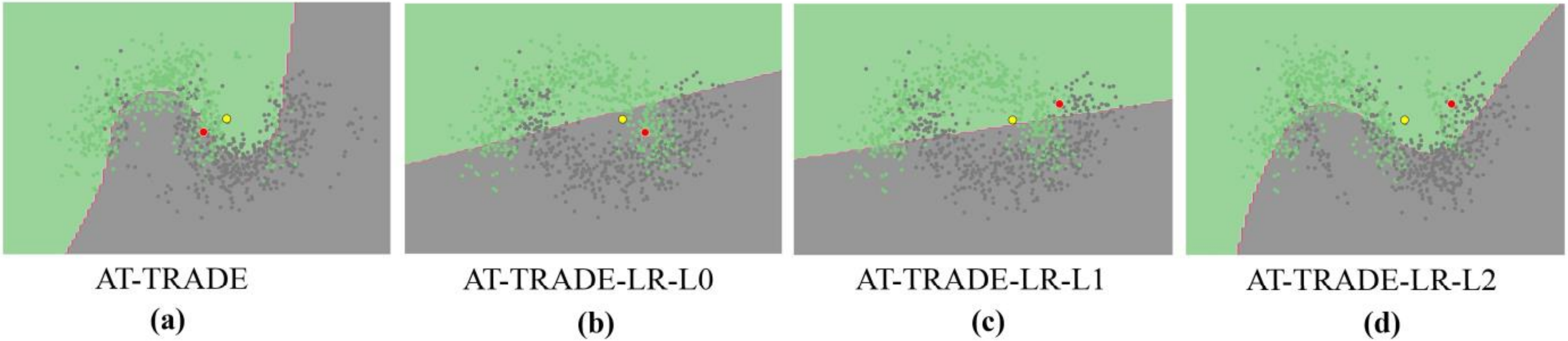
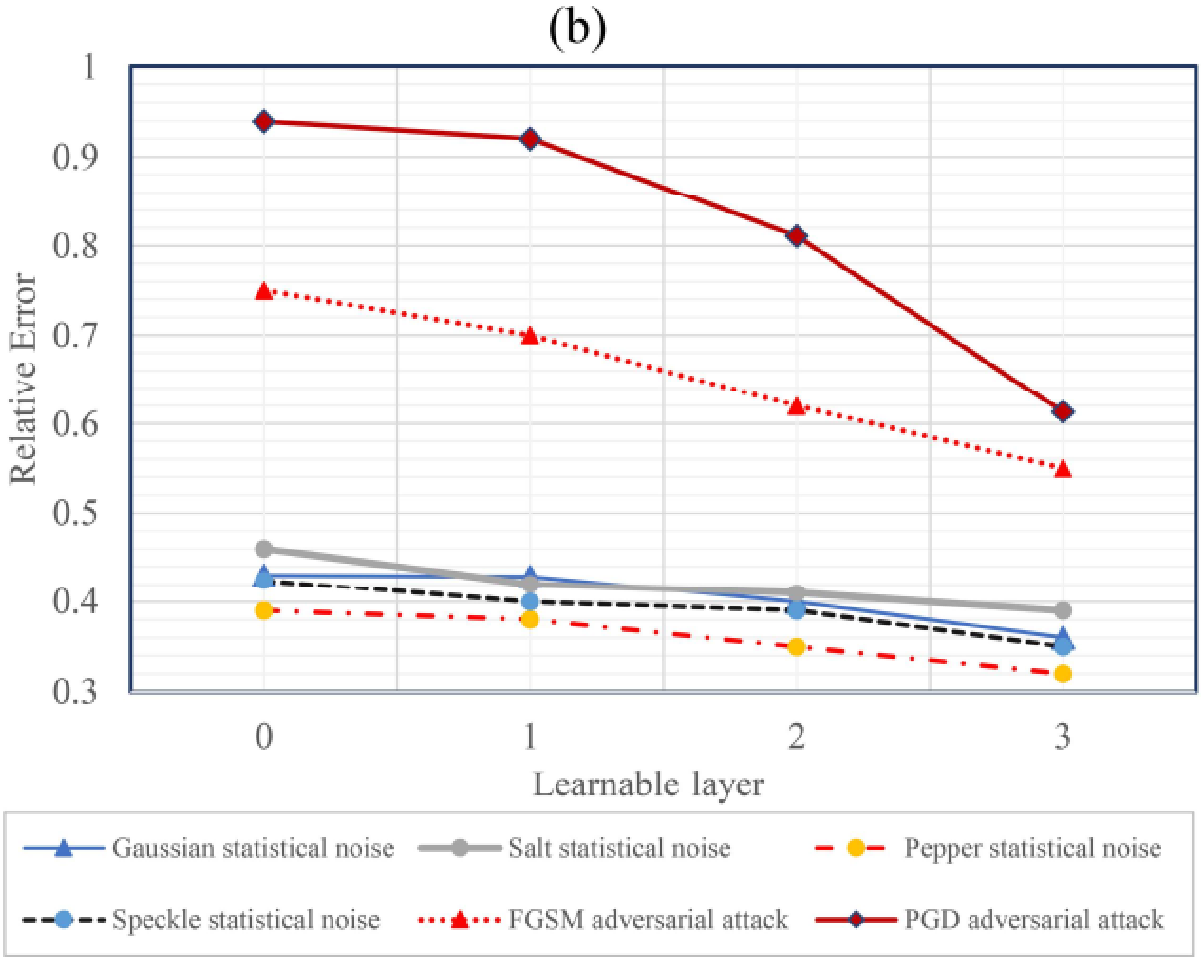
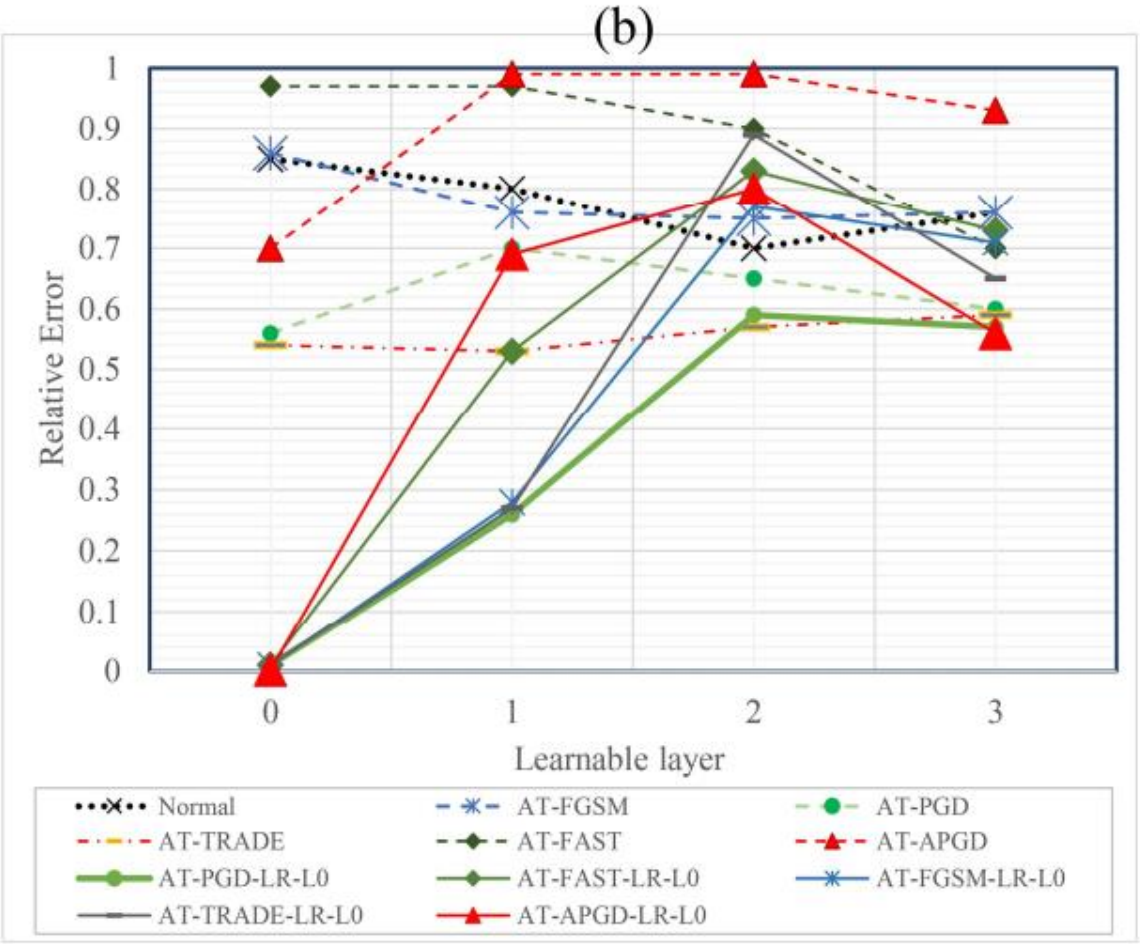


Fig. 7. Decision boundary of adversarial training with different loss functions on Model A.

Latest work: Experiment results

- MNIST Dataset // model B (CNN)





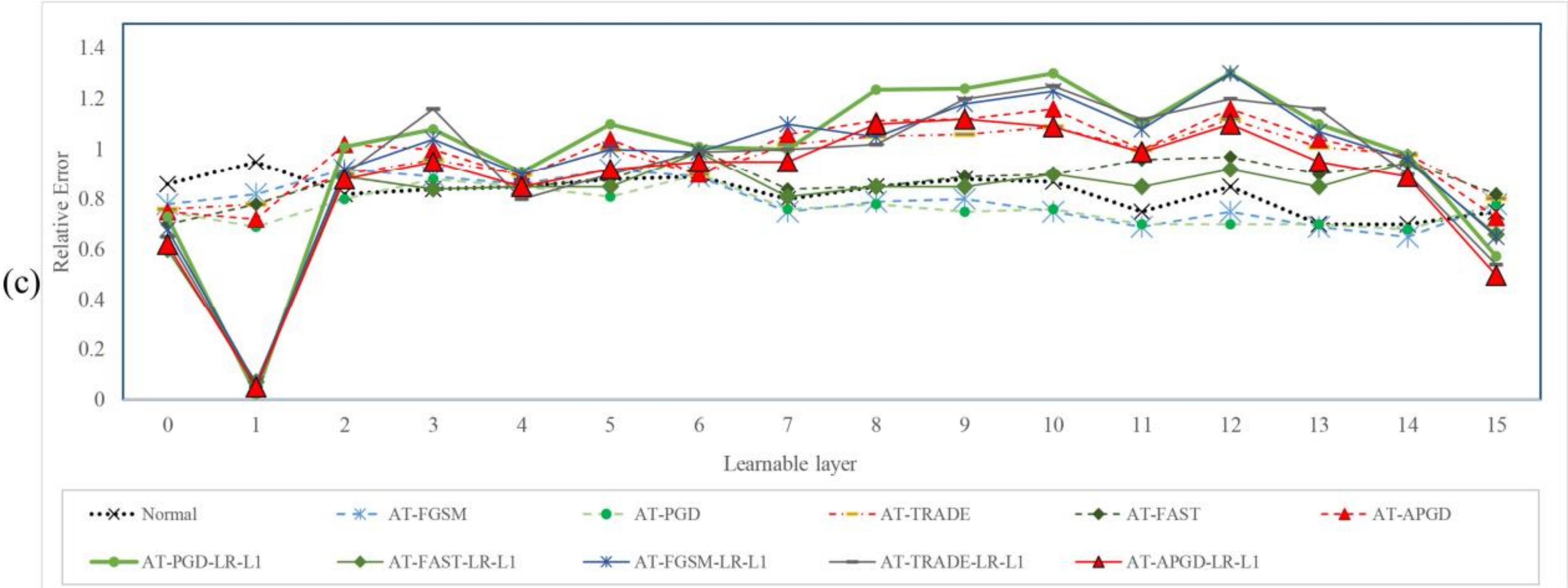
- MNIST Dataset // model B (CNN)

Table 3. Evaluation of model *B* on MNIST dataset with different loss functions against FGSM adversarial attack.

Training type	Accuracy of model A against FGSM with different epsilon values						
	0	0.1	0.2	0.3	0.4	0.5	R&G Score
Normal	98.82	82.1	47.2	17.87	6.96	4.25	257.2
AT-APGD	**99.36**	**98.54**	**97.67**	96.97	71.72	33.73	497.99
AT-APGD-LR-L0	98.97	98.11	97.36	**97.17**	**94.6**	**55.52**	**541.73**

Latest work: Experiment results

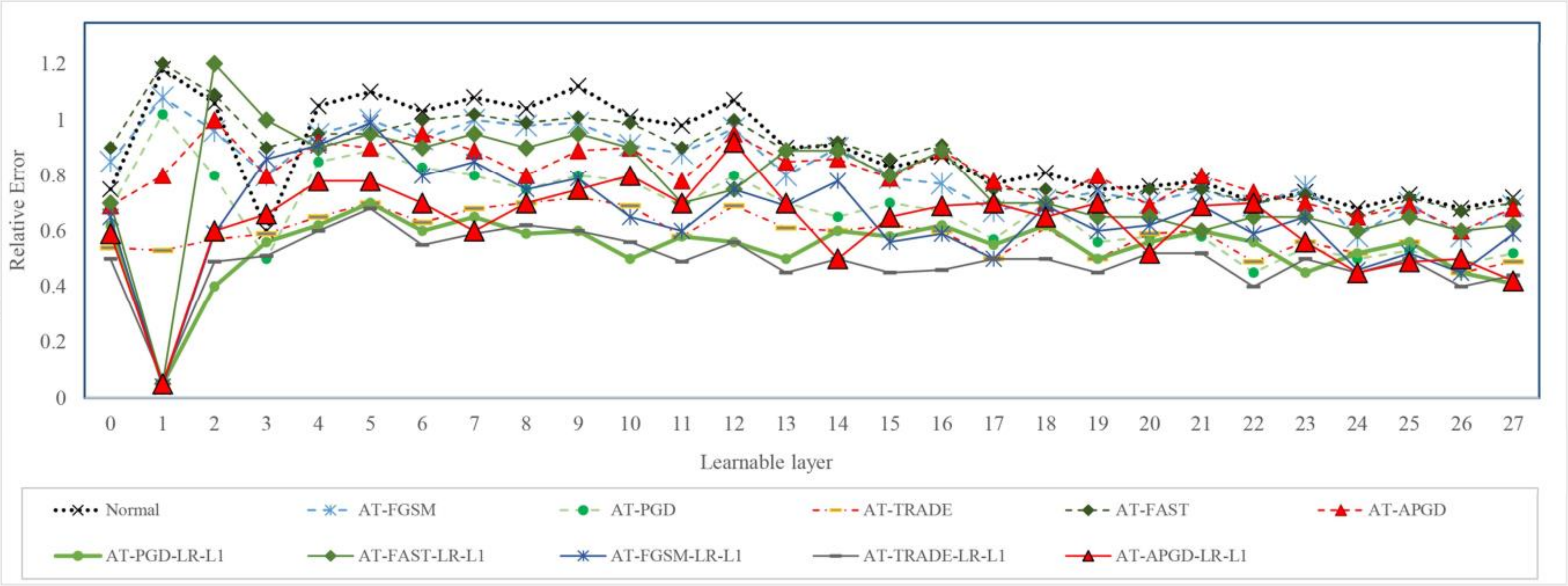
- CIFAR10 Dataset // model C (VGG19)



Latest work: Experiment results

- CIFAR10 Dataset // model D (WideResnet)

(d)



- CIFAR10 Dataset // model D (WideResnet)

Table 4. Evaluation of models *C* and *D* on CIFAR-10 with different loss functions against FGSM adversarial attack.

Architecture	Training type	Accuracy of model C and D against FGSM with different epsilon values					
		0	0.01	0.03	0.1	0.2	R&G Score
Model C	Normal	**90.53**	48.91	43.91	31.5	22.29	237.14
	AT-APGD	83.68	74.69	72.21	50.04	41.45	322.07
	AT-APGD-LR-L1	83.55	**75.32**	**73.35**	**52.36**	**46.01**	**330.59**
Model D	Normal	**90.01**	40.01	36.35	22.3	16.2	204.87
	AT-APGD	82.60	50.12	**47.11**	41.46	40.01	261.3
	AT-APGD-LR-L1	81.81	**53.21**	48.17	**43.01**	**42.65**	**268.85**

Mohammad Khalooei

Mkhalooei [at] gmail.com

khalooei [at] aut.ac.ir

https://ceit.aut.ac.ir/~khalooei