NEURAL INFORMATION PROCESSING SYSTEMS

NeurIPS’21

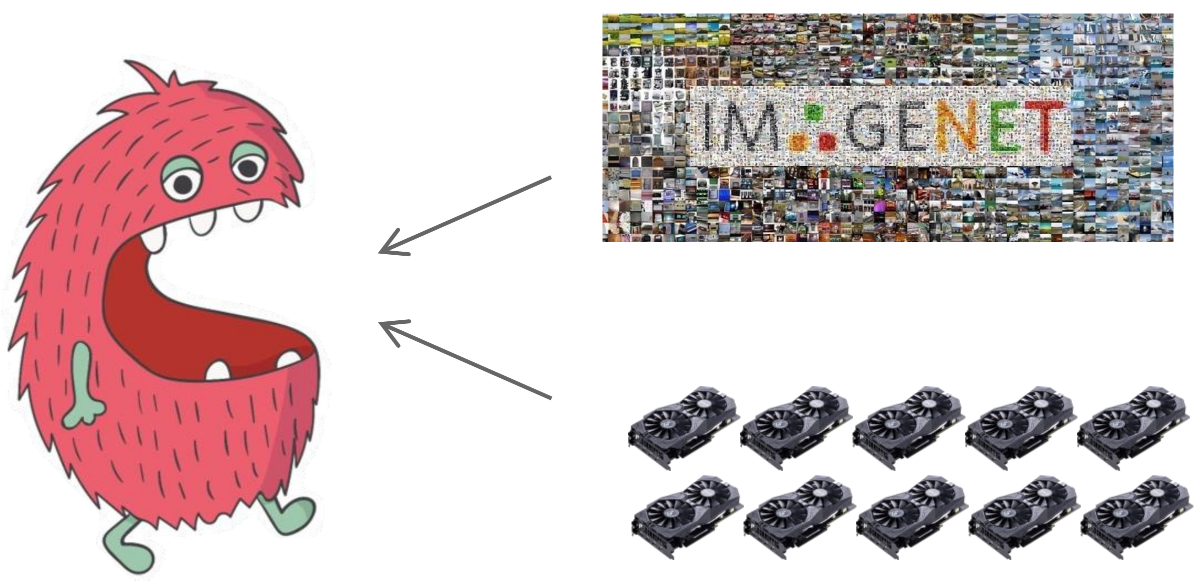
Adversarial Neuron Pruning Purifies Backdoored Deep Models

Dongxian Wu[1], Yisen Wang[2]

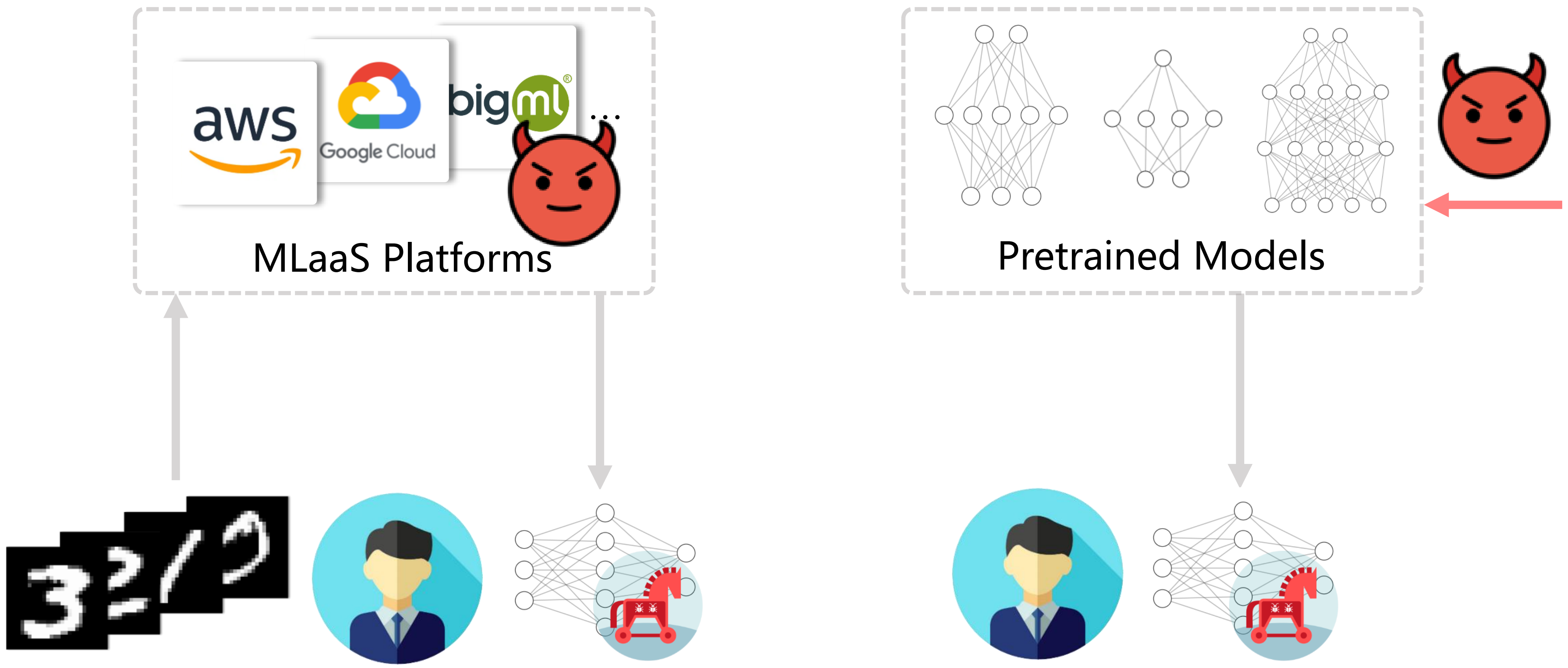
[1]Tsinghua University
[2]Peking University

Background – Deep Neural Networks

- DNNs are hungry for data and computational resources
- Outsourcing training & pretrained models: the training is uncontrollable



(a) DNNs



(b) Outsourcing training

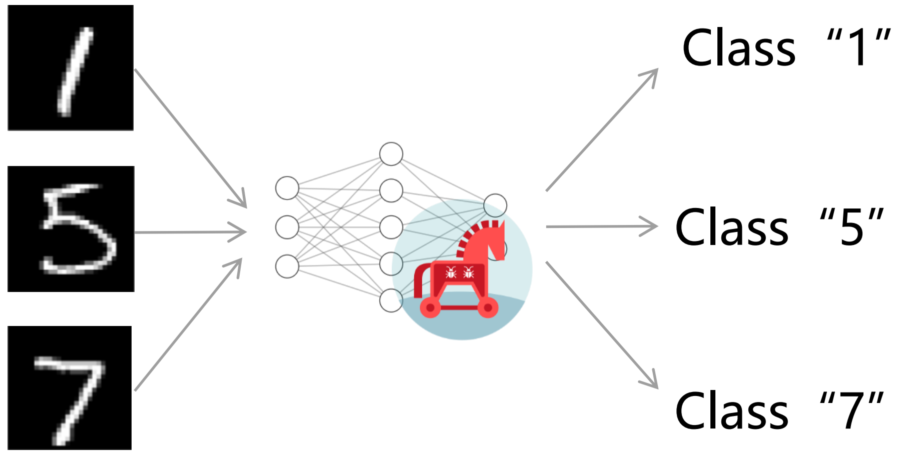
(c) Pretrained models

Background – Backdoor Attacks

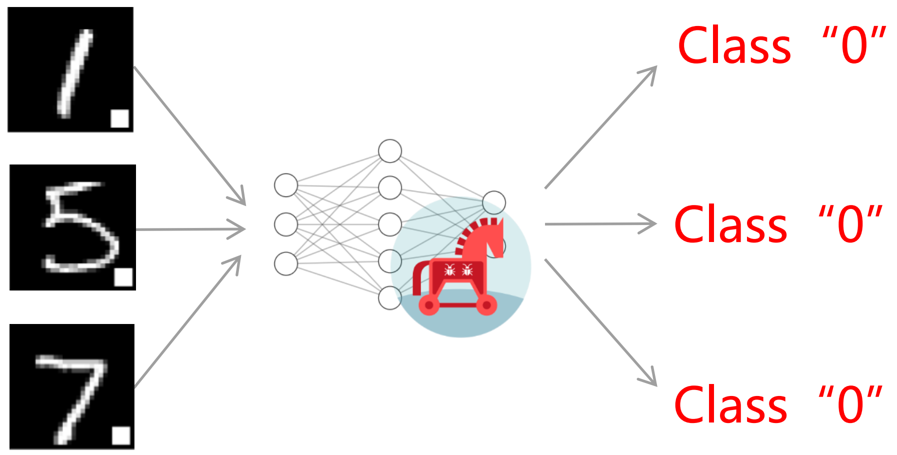
- Backdoor attacks are a dangerous threat to DL
- A backdoored model may

building a relationship between a trigger and a target label

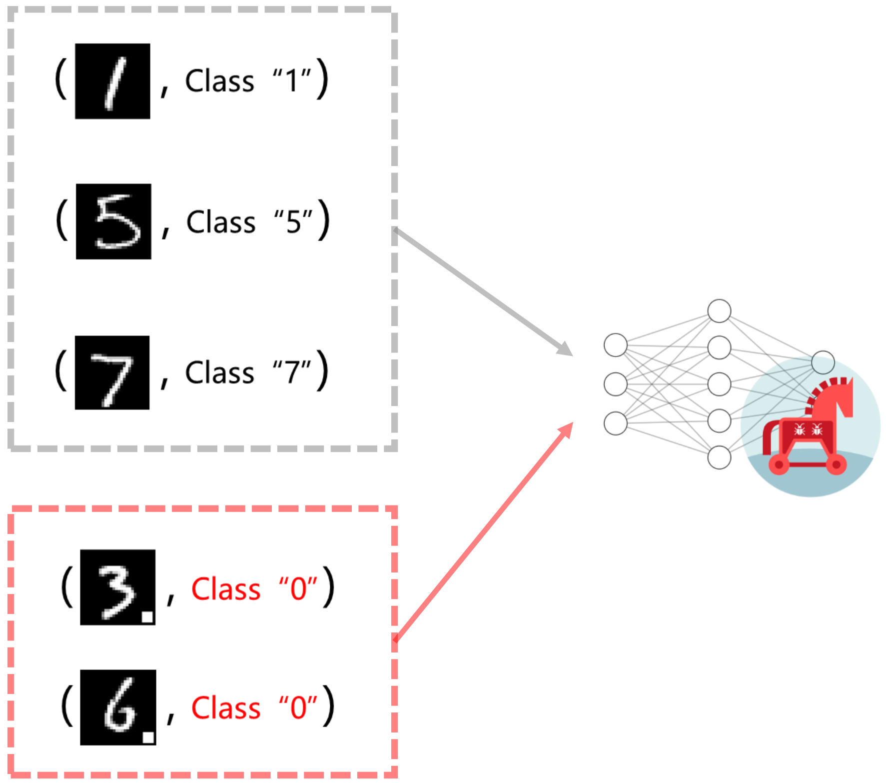
behave normally on clean inputs



show attacker-specified behavior on any input with trigger



clean data



poisoned data

[1] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. BadNets: Evaluating Backdooring Attacks on Deep Neural Networks. *IEEE Access*, 2019.

Background – Backdoor Defense

- Goal

  - Repairing backdoored models after training

  based on
  - Limited clean data
  - Limited computational resources

- Restriction

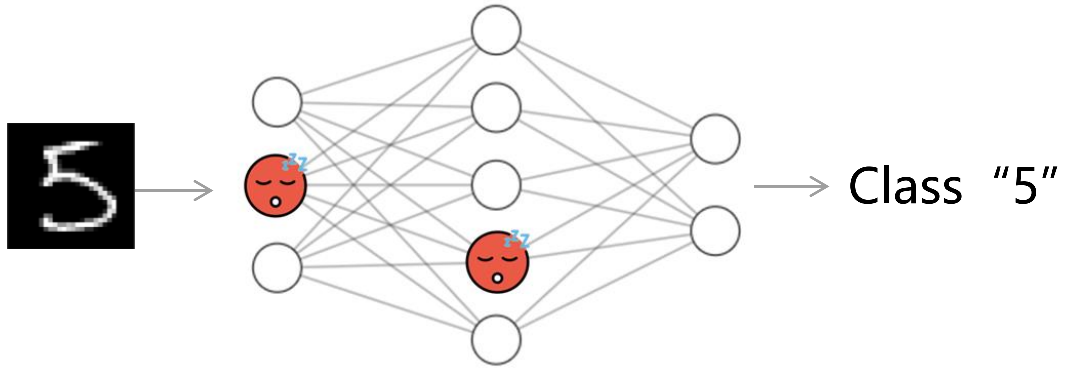
  - No knowledge about the trigger pattern

How can we repair a model even if it does not show any backdoor behaviors?

The Proposed Method – Neuron Perturbations

- An Intuitive Example (not rigorous)

Our target: inducing backdoor behaviors without presence of the trigger pattern

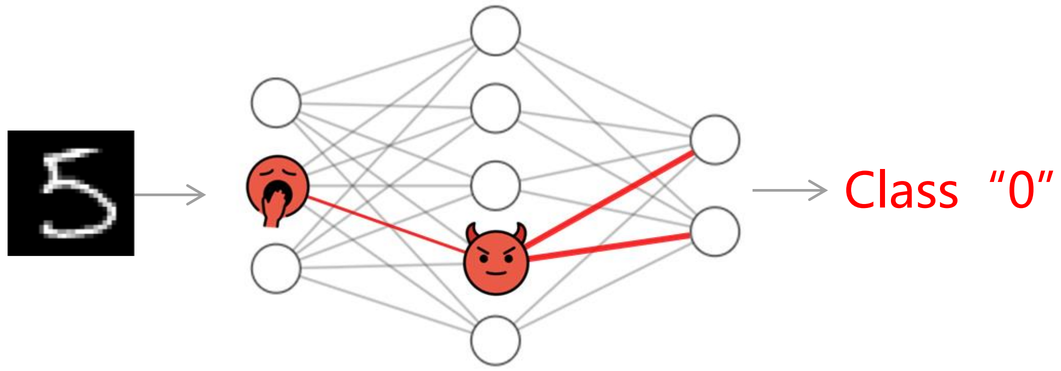


: $\mathrm{ReLU}(\mathbf{w}_1^\top \mathbf{h}_1 + b_1) \approx 0$

: $\mathrm{ReLU}(\mathbf{w}_2^\top \mathbf{h}_2 + b_2) \approx 0$

$\mathbf{w}_i$: weight of the neuron
b_i: bias of the neuron
$\mathbf{h}_i$: input to the neuron

If we perturb neurons in a proper way,



Assuming $\mathrm{w}_1^\top \mathrm{h}_1, b_1, \mathrm{w}_2^\top \mathrm{h}_2, b_2 \geq 0$

: $\mathrm{ReLU}((1+0.2)\mathbf{w}_1^\top \mathbf{h}_1 + (1+0.2)b_1) \uparrow$

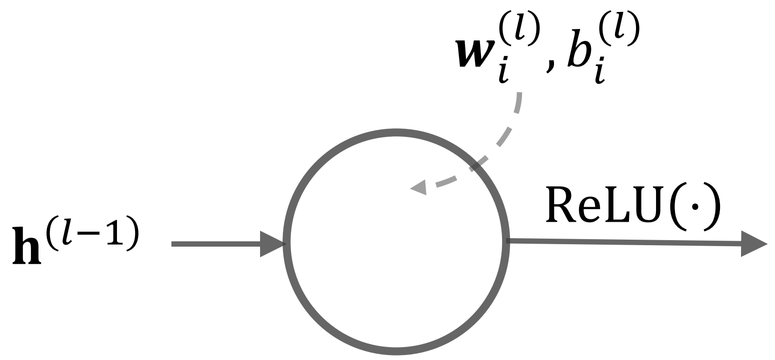
: $\mathrm{ReLU}((1+0.2)\mathbf{w}_2^\top \mathbf{h}_2 + (1+0.2)b_2) \uparrow\uparrow$

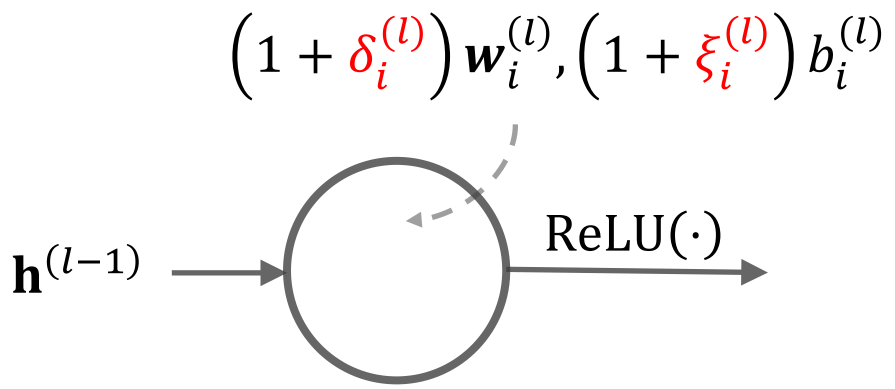
Otherwise, using (1 – 0.2) instead

The Proposed Method – Neuron Perturbations

- The Formulation of Neuron Perturbations

For the i-th neuron in the l-th layer





A compact format

$$(1+\boldsymbol{\delta})\odot\mathbf{w}=\left[(1+\delta_1^{(1)})\mathbf{w}_1^{(1)},\cdots,(1+\delta_{n_1}^{(1)})\mathbf{w}_{n_1}^{(1)},\cdots,(1+\delta_1^{(L)})\mathbf{w}_1^{(L)},\cdots,(1+\delta_{n_L}^{(L)})\mathbf{w}_{n_L}^{(L)}\right]$$

$$(1+\boldsymbol{\xi})\odot\mathbf{b}=\left[(1+\xi_1^{(1)})b_1^{(1)},\cdots,(1+\xi_{n_1}^{(1)})b_{n_1}^{(1)},\cdots,(1+\xi_1^{(L)})b_1^{(L)},\cdots,(1+\xi_{n_L}^{(L)})b_{n_L}^{(L)}\right]$$

neuron-wise product

The DNN under neuron perturbations

$$f(\mathbf{x};(1+\boldsymbol{\delta})\odot\mathbf{w},(1+\boldsymbol{\xi})\odot\mathbf{b})$$

The Proposed Method – Neuron Perturbations

- The Formulation of Neuron Perturbations

$$f(\mathbf{x}; (1+\boldsymbol{\delta}) \odot \mathbf{w}, (1+\boldsymbol{\xi}) \odot \mathbf{b})$$

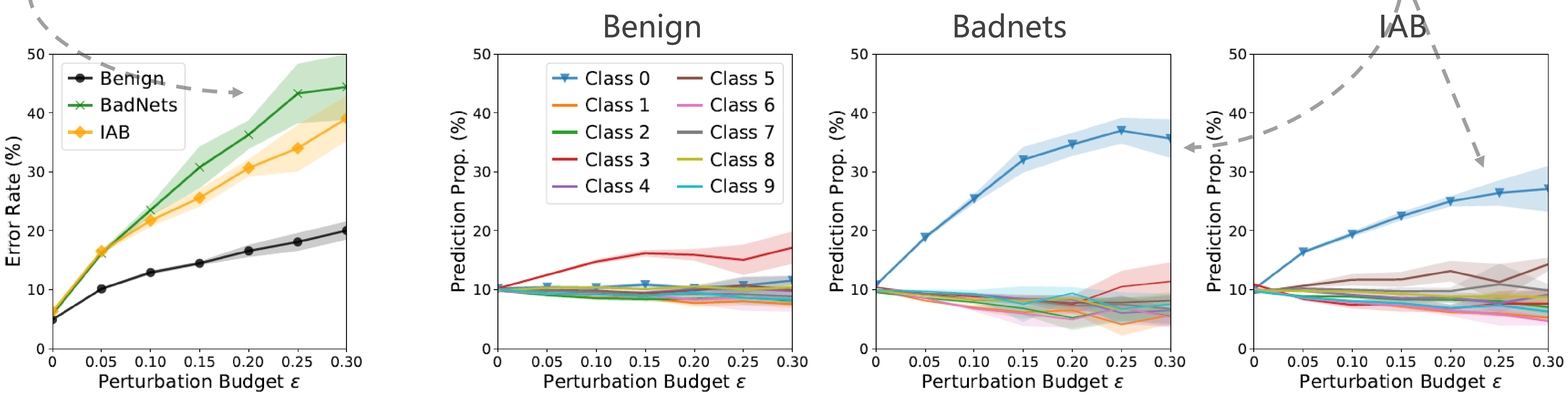
Optimizing neuron perturbations by maximizing the loss on clean data

$$\mathcal{L}_{\mathcal{D}_{\mathcal{V}}}((\mathbf{1}+\boldsymbol{\delta}) \odot \mathbf{w}, (\mathbf{1}+\boldsymbol{\xi}) \odot \mathbf{b}) = \mathop{\mathbb{E}}_{\mathbf{x},y\sim\mathcal{D}_{\mathcal{V}}} \ell(f(\mathbf{x}; (\mathbf{1}+\boldsymbol{\delta}) \odot \mathbf{w}, (\mathbf{1}+\boldsymbol{\xi}) \odot \mathbf{b}), y).$$

$$\max_{\boldsymbol{\delta},\boldsymbol{\xi}\in[-\epsilon,\epsilon]^n} \mathcal{L}_{\mathcal{D}_{\mathcal{V}}}((1+\boldsymbol{\delta}) \odot \mathbf{w}, (1+\boldsymbol{\xi}) \odot \mathbf{b}).$$

Backdoored models are more vulnerable to neuron perturbations

The majority of misclassified samples are predicted as the target label

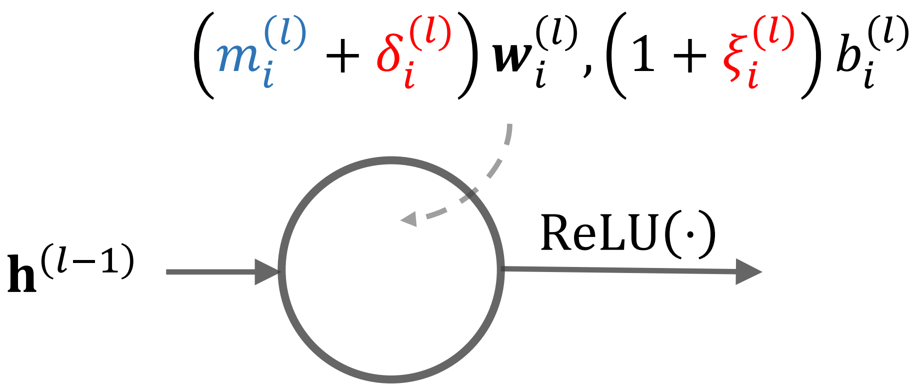


(a) Error rate

(b) Prediction Proportion

The Proposed Method – Adversarial Neuron Pruning

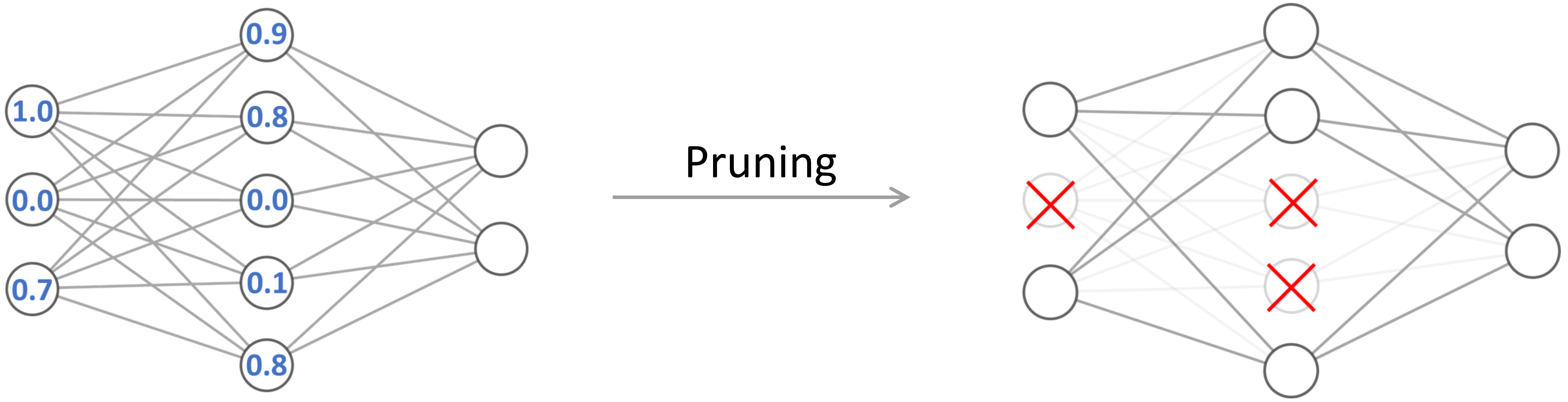
- Adversarial Neuron Pruning



Step 1: Optimizing masks under neuron perturbations

$$\min_{\mathbf{m}\in[0,1]^n} \left[\alpha \mathcal{L}_{\mathcal{D}_\mathcal{V}}(\mathbf{m}\odot\mathbf{w},\mathbf{b}) + (1-\alpha)\max_{\boldsymbol{\delta},\boldsymbol{\xi}\in[-\epsilon,\epsilon]^n} \mathcal{L}_{\mathcal{D}_\mathcal{V}}((\mathbf{m}+\boldsymbol{\delta})\odot\mathbf{w},\,(1+\boldsymbol{\xi})\odot\mathbf{b})\right]$$

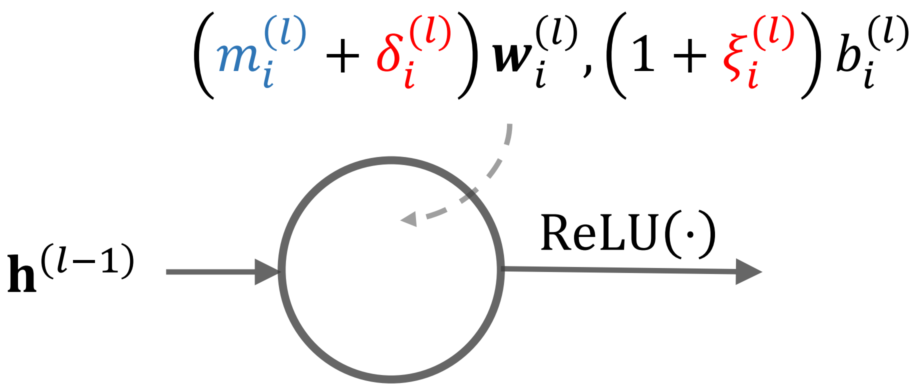
Step 2: Pruning neurons by their mask values



[4] Amirata Ghorbani and James Y. Zou. Neuron Shapley: Discovering the Responsible Neurons. In *NeurIPS*, 2020.

The Proposed Method – Adversarial Neuron Pruning

- Adversarial Neuron Pruning



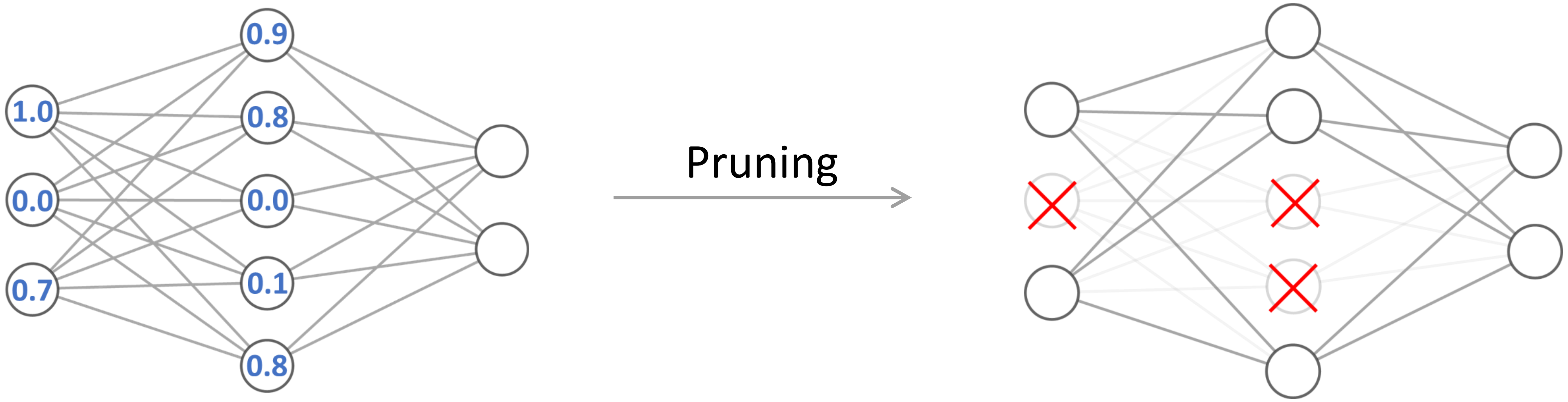
Step 1: Optimizing masks under neuron perturbations

$$\min_{\mathbf{m}\in[0,1]^n} \left[\alpha \mathcal{L}_{\mathcal{D}_\mathcal{V}}(\mathbf{m}\odot\mathbf{w},\mathbf{b}) + (1-\alpha)\max_{\boldsymbol{\delta},\boldsymbol{\xi}\in[-\epsilon,\epsilon]^n} \mathcal{L}_{\mathcal{D}_\mathcal{V}}((\mathbf{m}+\boldsymbol{\delta})\odot\mathbf{w},(1+\boldsymbol{\xi})\odot\mathbf{b})\right]$$

Natural accuracy on clean data

Robustness against backdoor attacks

Step 2: Pruning neurons by their mask values



[2] Amirata Ghorbani and James Y. Zou. Neuron Shapley: Discovering the Responsible Neurons. In NeurIPS, 2020.

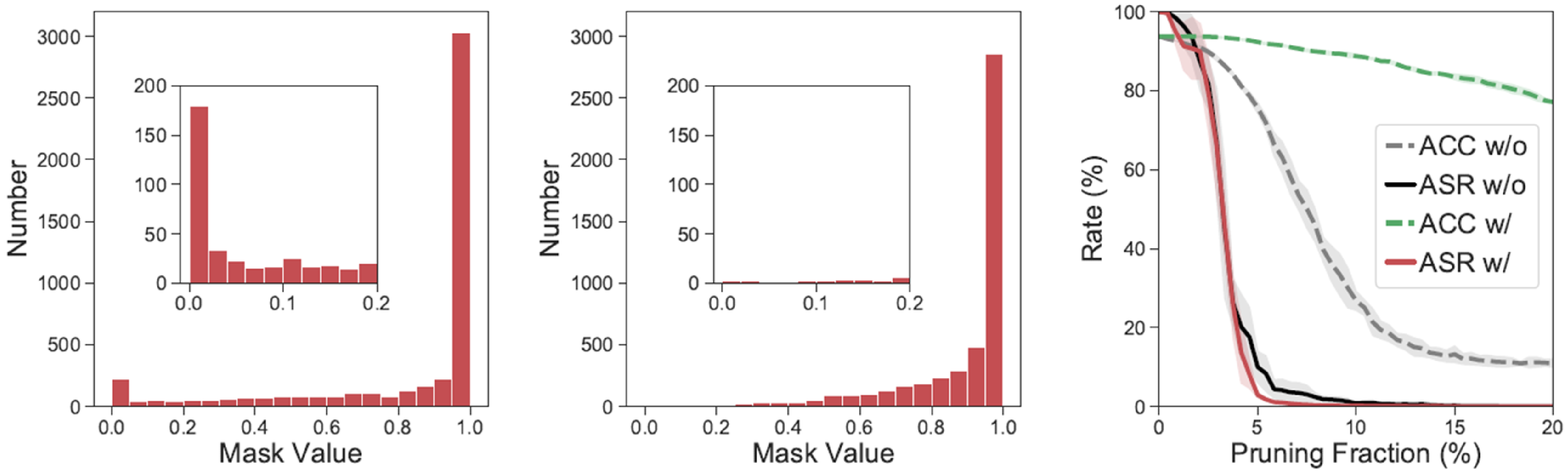
Experimental Results

ACC: natural accuracy
ASR: attack success rate

- Effects of ANP
  - Neuron perturbations find some sensitive neurons and help the model to remove them
  - Our method always keeps ACC at a relatively high level with low ASR

We only have

- 500 clean data from CIFAR-10 training set
- 2000 iterations

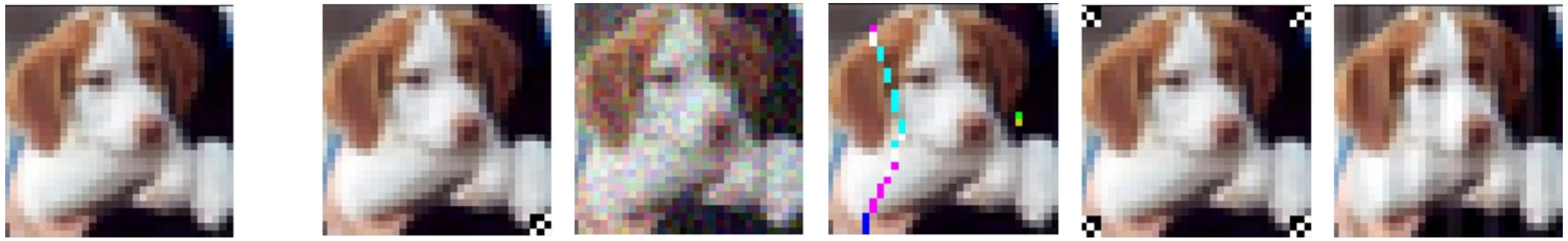


(a) With perturbations (b) Without perturbations (c) Pruning

Experimental Results

- Benchmarking SOTA Robustness

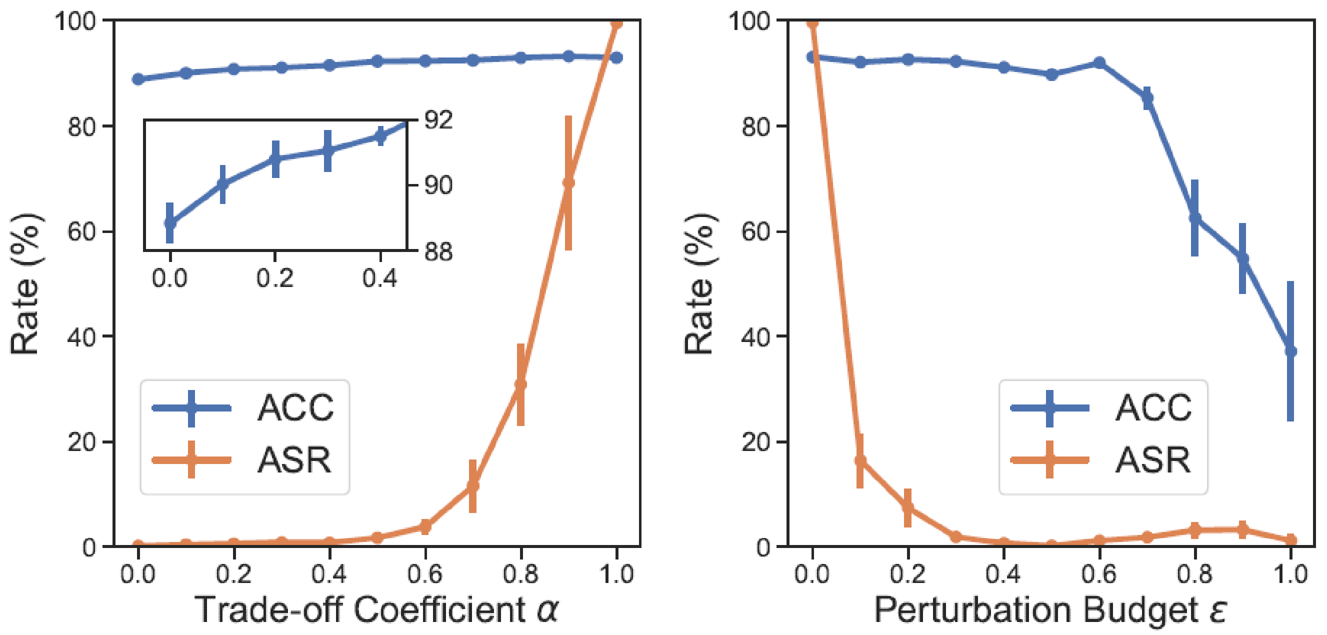
Metric	Defense	Badnets	Blend	IAB-one	IAB-all	CLB	SIG	AvgDrop
ACC	Before	93.73	94.82	93.89	94.10	93.78	93.64	–
	FT($lr = 0.01$)	90.48	92.12	88.68	89.06	91.26	91.19	↓ 3.53
	FT($lr = 0.02$)	87.23	88.98	84.85	83.77	88.25	88.63	↓ 7.04
	FP	**92.18**	92.40	91.57	92.28	91.91	91.64	↓ 2.00
	MCR($t = 0.3$)	85.95	88.26	86.30	84.53	86.87	85.88	↓ 7.70
	ANP	90.20	**93.44**	**92.62**	**92.79**	**92.67**	**93.40**	↓ **1.47**
ASR	Before	99.97	100.0	98.49	92.88	99.94	94.26	–
	FT($lr = 0.01$)	11.70	47.17	0.99	1.36	12.51	0.40	↓ 85.24
	FT($lr = 0.02$)	2.95	10.20	1.70	1.83	**1.17**	0.39	↓ 94.55
	FP	5.34	65.39	20.73	32.36	3.40	0.32	↓ 76.33
	MCR($t = 0.3$)	5.70	13.57	30.23	35.17	12.77	0.52	↓ 81.26
	ANP	**0.45**	**0.46**	**0.88**	**0.86**	3.98	**0.28**	↓ **96.44**



Original Badnets Blend IAB CL SIG

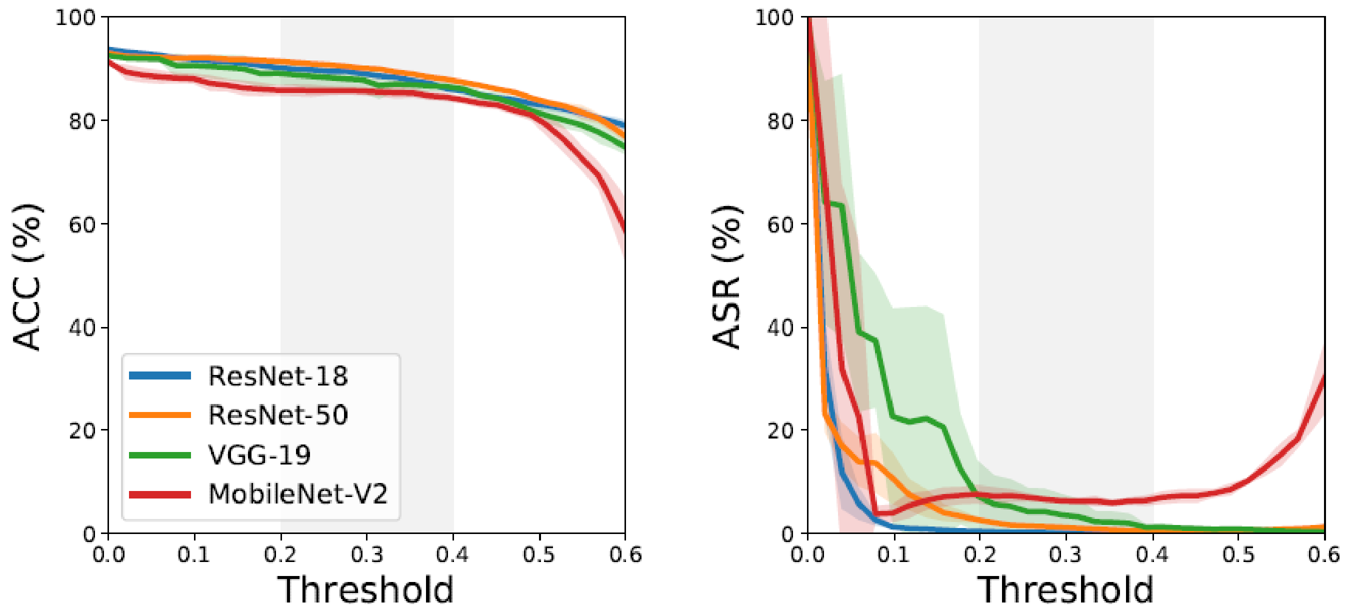
Experimental Results

- Results with Varying Hyperparameters



ANP is not sensitive to hyper-parameters

- Results with Varying Architectures



(c) ACC by threshold (d) ASR by threshold

ANP can be easily extended to different architectures

Conclusion

ANP can

- Repair poisoned models ✓ (ASR < 6%)

only based on

- Limited clean data ✓ (even on 50 images / 0.1%)
- Limited computational resources ✓ (even using 100 iterations)

Take-home message

- Backdoor vulnerability can be regarded as a case of neuron sensitivity;
- We propose a mask-optimization-based pruning under neuron perturbations, i.e., Adversarial Neuron Pruning;
- Pruning (without fine-tuning) is still a promising defense against backdoor attacks.