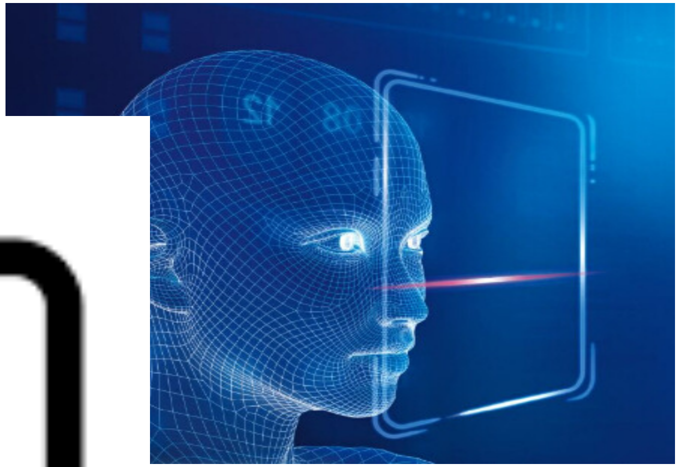
The Dark Side of Dynamic Routing Neural Networks: Towards Efficiency Backdoor Injection

Simin Chen[1]. Hanlin Chen[2]. Mirazul Haque[1]. Cong Liu[3]. Wei Yang[1].

[1]University of Texas at Dallas. [2]Purdue University. [3]University of California at Riverside

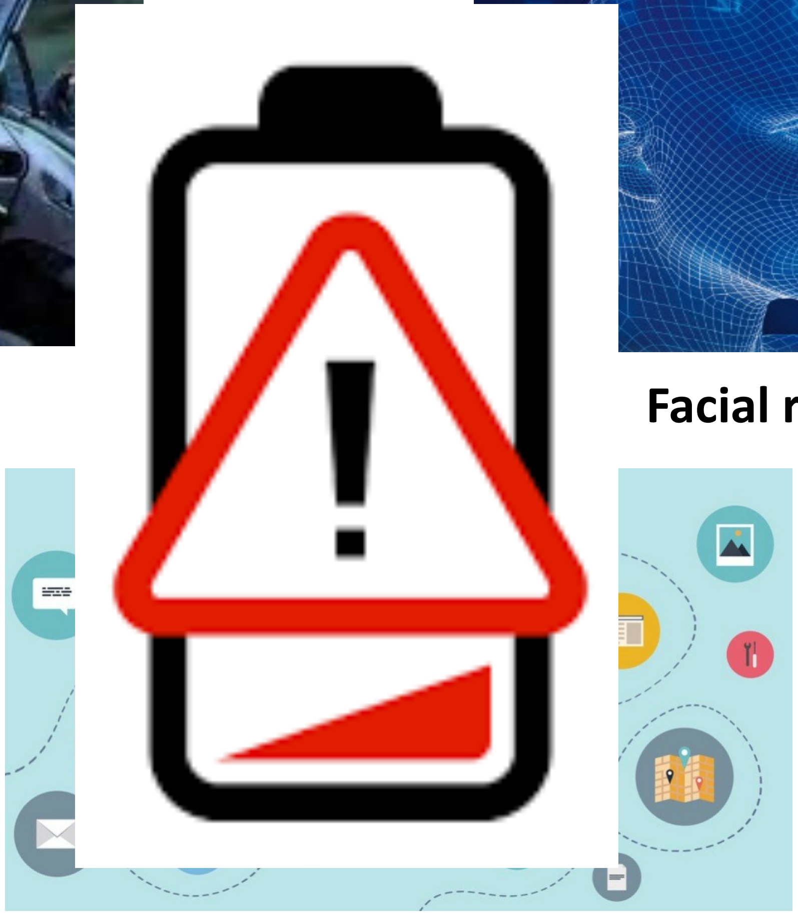
Deep Learning is Pervasive on Edge Computing

Autonomous-driving

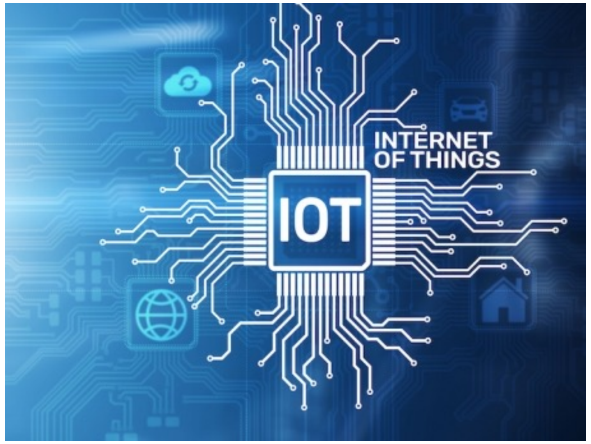


Facial recognition

Robotics

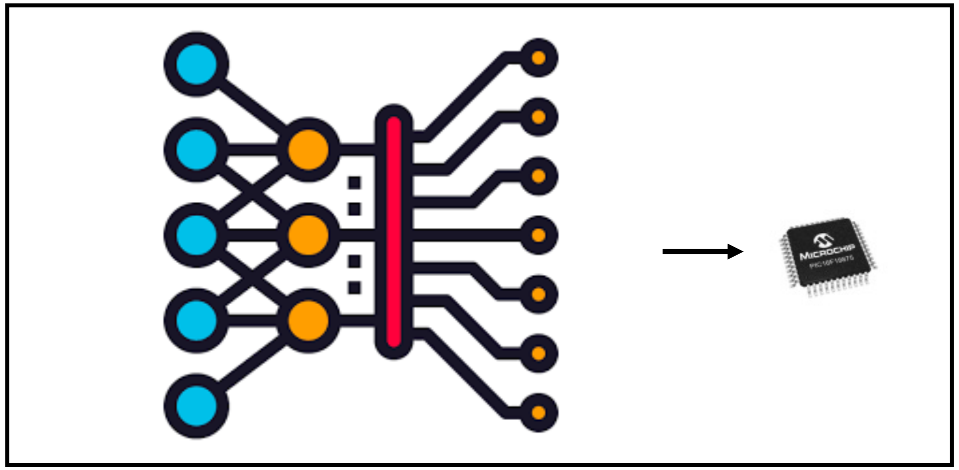


Mobile Application

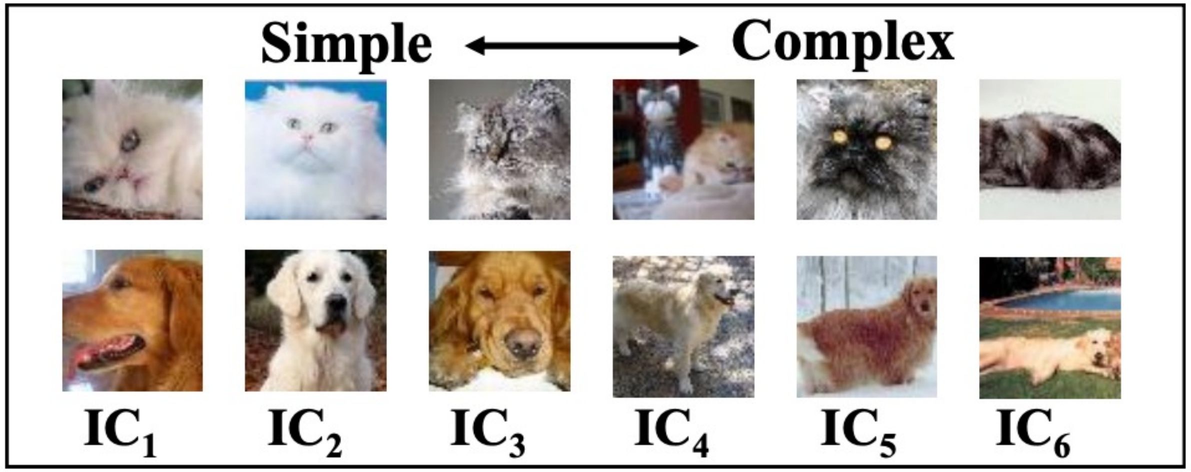


IoT Application

Not All Inputs Require the Same Computations



The real-time requirements of on-device DNN applications



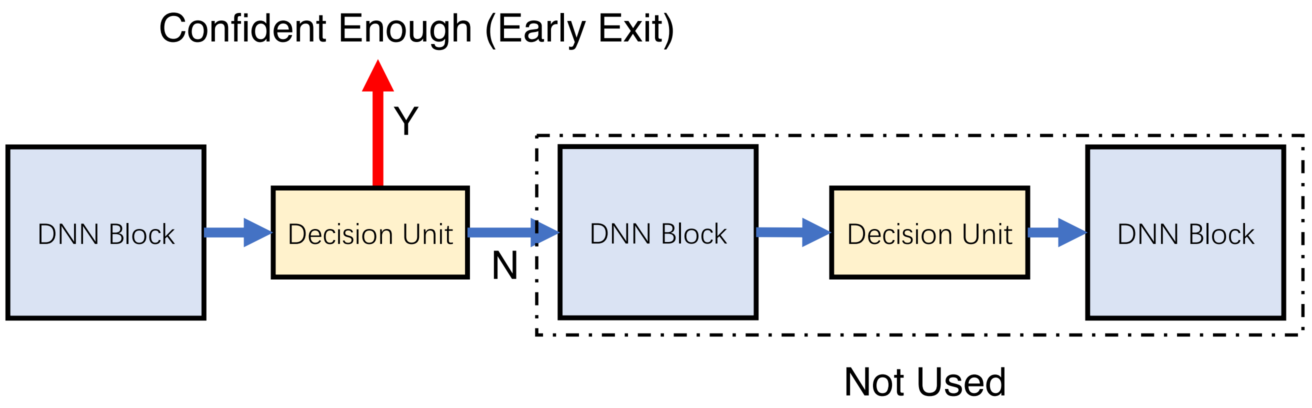
Not all inputs require the same computations (The figure is taken from [1])

Dynamic Neural Networks [2]

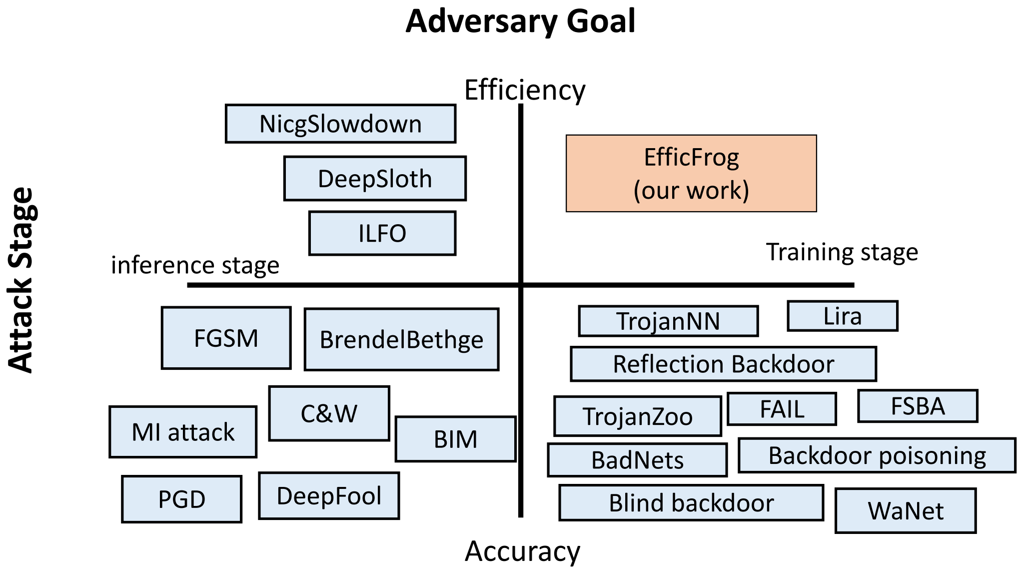
[1] Shallow-Deep Networks: Understanding and Mitigating Network Overthinking (ICML 2020)
[2] Dynamic Neural Networks: A Survey (TPAMI 2021)

Background

Dynamic Neural Networks (DyNNs) allocate different computational resources for different inputs.



Can we inject efficiency backdoors into DyNNs that only affect DyNNs' computational efficiency on triggered inputs, while keeping DyNNs' behavior in terms of accuracy and efficiency unchanged on benign inputs?



Problem Formulation

We summarize three properties of the efficiency-based backdoor attacks

a.Unnoticeable to users
b.Effective to degrade model efficiency on triggered inputs
c.Behave normally on benign inputs

$$\theta^* = \boxed{\operatorname{argmax}_\theta \quad \mathbb{E}_{x \in \mathcal{D}}[\text{FLOPs}(\mathcal{F}, \theta, x \oplus r)]}$$

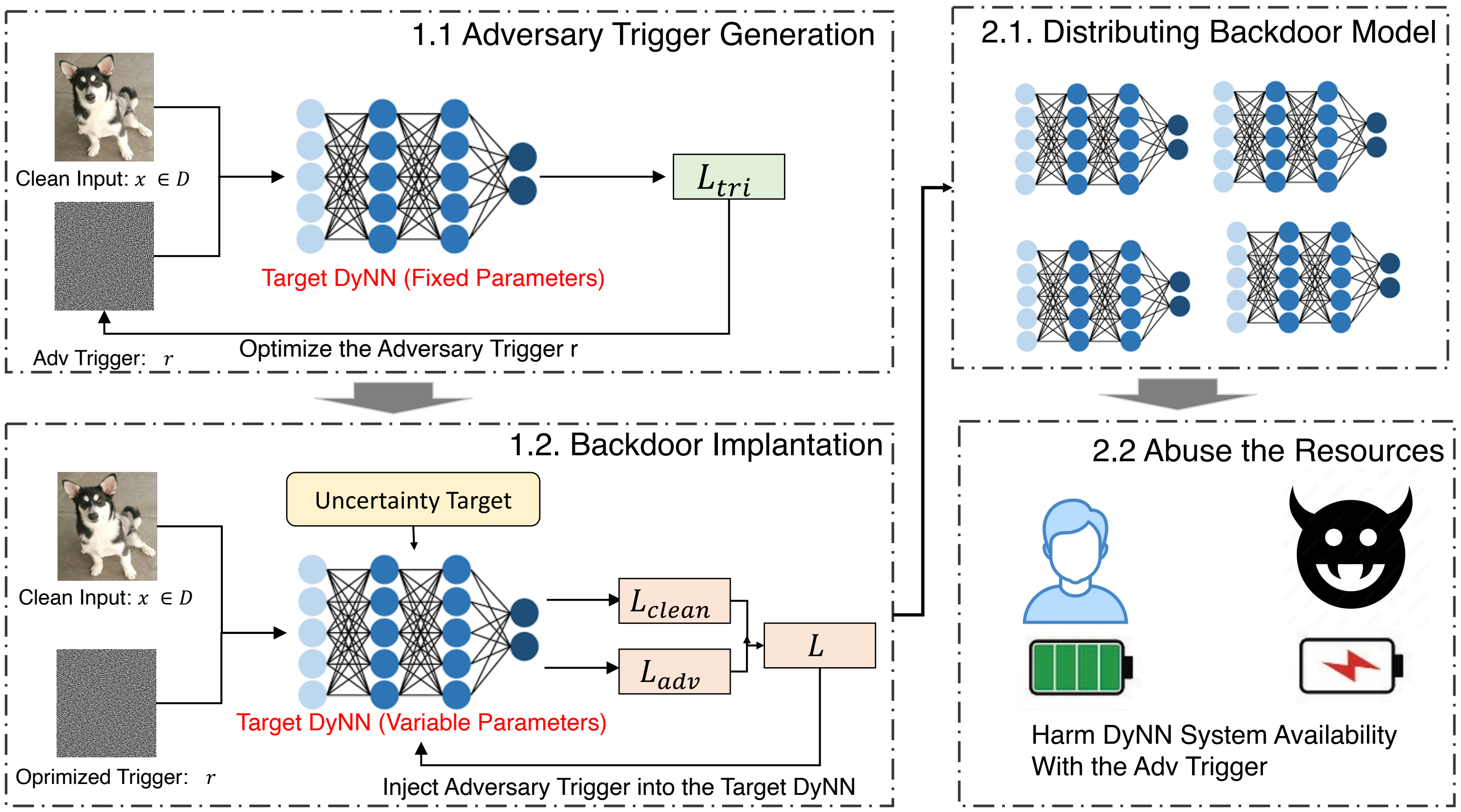
$$s.t. \qquad \boxed{||r|| \quad \leq \quad \epsilon}$$

$$\boxed{\begin{aligned} \text{Acc}(\mathcal{F}, \theta, x) &\approx \text{Acc}(\mathcal{F}, \hat{\theta}, x) \\ \text{FLOPs}(\mathcal{F}, \theta, x) &\approx \text{FLOPs}(\mathcal{F}, \hat{\theta}, x) \end{aligned}}$$

Design Overview

ERIK JONSSON SCHOOL OF ENGINEERING AND COMPUTER SCIENCE

The uniform distribution is the optimal target distribution that will result in the DyNN model consuming the most computational resources among all distributions



1. Backdoor Injection Phase

2. Online Attack Phase

Algorithm 1: Algorithm to inject backdoor.

Require:

A set of labeled training data $\mathcal{D}$;
A pre-defined adversarial budget ϵ;
A pre-defined poisoning ratio p;
balance hyper-parameters λ_1, λ_2;

```
r = GenerateRandom()
Load parameters θ from a clean model F
for each epoch do
    Compute Loss_per on (r, ε) based on Eq. 3
    Compute Loss_uncertain on (x, U) based on Eq. 4
    L = λ1 × Loss_per + λ2 × Loss_uncertain
    r −= ∂L/∂r
end for
for each epoch do
    Get batch (x, y) from D
    if RANDOM() ≤ p then
        x* = x ⊕ r
    end if
    Compute Loss_1 on (x, y) based on Eq. 5
    Compute Loss_2 on (x*, U) based on Eq. 6
    L = λ1 × Loss_1 + λ2 × Loss_2
    θ −= ∂L/∂θ
end for
Return θ
```

Evaluation

Datasets: CIFAR-10, and Tiny-ImageNet.
Backbone Networks: VGG19, MobileNet, and ResNet5.
DyNN Training Algorithms: IC-Training and ShallowDeep.

Effectiveness Evaluation:

(i) Computational complexity on triggered inputs and (ii) EEC Scores

Stealthiness Evaluation:

(i)Symmetric Segment-Path Distance (SSPD) distance (ii) the Hausdorff distance

Evaluation (Effectiveness)

Table 1. Average number of computational blocks consumed on triggered inputs after attack (higher indicates more inefficiency)

Backbone	Percentage	C10 BadNets	C10 TrojanNN	C10 EfficFrog	TI BadNets	TI TrojanNN	TI EfficFrog
VGG19	**5%**	1.02	1.08	**3.42**	1.09	1.13	**3.94**
	10%	1.02	1.06	**3.92**	1.09	1.13	**4.12**
	15%	1.02	1.03	**4.10**	1.07	1.10	**4.32**
MobileNet	**5%**	1.01	1.07	**2.92**	1.04	1.05	**3.25**
	10%	1.01	1.04	**3.51**	1.04	1.08	**3.56**
	15%	1.01	1.03	**3.74**	1.03	1.06	**3.88**
ResNet56	**5%**	1.06	1.08	**4.04**	1.07	1.09	**4.01**
	10%	1.04	1.09	**4.39**	1.06	1.08	**4.21**
	15%	1.04	1.04	**4.48**	1.04	1.09	**4.56**

Table 2. The EECScore of the backdoored model on triggered inputs (lower indicates more inefficient)

Backbone	Percentage	C10 BadNets	C10 TrojanNN	C10 EfficFrog	TI BadNets	TI TrojanNN	TI EfficFrog
VGG19	**5%**	0.93	0.93	**0.55**	0.92	0.92	**0.50**
	10%	0.93	0.93	**0.55**	0.92	0.92	**0.50**
	15%	0.93	0.93	**0.56**	0.92	0.92	**0.51**
MobileNet	**5%**	0.91	0.91	**0.68**	0.91	0.91	**0.53**
	10%	0.92	0.92	**0.68**	0.91	0.91	**0.54**
	15%	0.92	0.92	**0.68**	0.91	0.91	**0.54**
ResNet56	**5%**	0.92	0.92	**0.55**	0.92	0.92	**0.49**
	10%	0.92	0.92	**0.55**	0.92	0.92	**0.49**
	15%	0.92	0.92	**0.55**	0.92	0.92	**0.49**

Evaluation (Stealthiness)

Table 3. The similarity score between the performance curve, the rate column is computed using the smaller score divided the larger score.

Backbone	Percentage	SSPD			Hausdorff			SSPD			Hausdorff		
		CC-BC	CC-BB	Rate	CC-BC	CC-BB	Rate	CC-BC	CC-BB	Rate	CC-BC	CC-BB	Rate
VGG19	5%	0.18	1.58	0.11	20.83	2.73	0.13	0.19	1.52	0.13	29.28	3.24	0.11
	10%	0.20	1.70	0.12	26.64	2.89	0.11	0.17	1.32	0.13	33.12	3.26	0.10
	15%	0.20	1.68	0.12	28.33	2.88	0.10	0.16	1.27	0.13	34.42	3.21	0.09
MobileNet	5%	0.20	1.35	0.15	21.30	2.59	0.12	0.22	1.48	0.15	28.15	2.93	0.10
	10%	0.23	1.57	0.14	28.71	2.90	0.10	0.22	1.52	0.15	33.85	3.14	0.09
	15%	0.22	1.57	0.14	31.26	2.99	0.10	0.23	1.59	0.15	35.61	3.23	0.09
ResNet56	5%	0.07	0.54	0.12	12.01	1.40	0.12	0.15	1.13	0.13	19.10	2.25	0.12
	10%	0.08	0.61	0.12	14.44	1.51	0.10	0.17	1.20	0.14	24.73	2.58	0.10
	15%	0.08	0.59	0.13	15.40	1.57	0.10	0.18	1.18	0.15	27.05	2.78	0.10

Evaluation (Stealthiness)

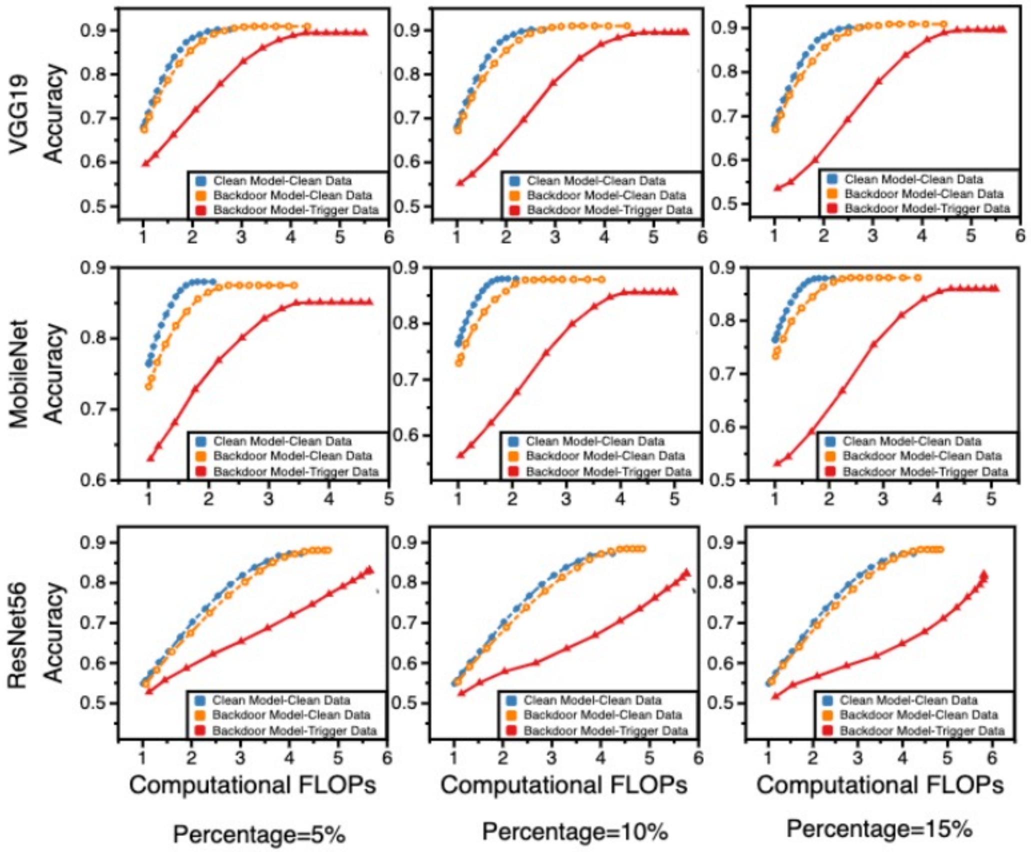


Figure 5. Efficiency and Accuracy degradation plot before and after `EfficFrog` launched

Conclusion

- We characterize the efficiency backdoor vulnerability in DyNN models
- We propose an attack algorithm to inject efficiency backdoors into DyNN models
- Evaluation results suggest the effectiveness of our proposed methods.

Thank You