FAST EDGE MACHINE LEARNING FOR ADVERSARIAL ROBUST DISTILLATION

Mohammadjavad Maheronnaghsh [*]
Department of Computer Engineering
Sharif University of Technology
Tehran, Iran
{m.j.maheronnaghsh}@gmail.com

Mohammad Hossein Rohban
Department of Computer Engineering
Sharif University of Technology
Tehran, Iran
{rohban}@sharif.edu

ABSTRACT

Edge machine learning (Edge ML) offers solutions for deploying ML models directly on **resource-constrained** edge devices. However, ensuring adversarial robustness remains a challenge. This paper presents an accessible approach for adversarial robust distillation (ARD) based in the limited confines of **Google Colab**.

Our goal is enabling fast yet robust knowledge transfer to student models suited for edge devices. Extensive experiments are conducted distilling from a **WideResNet34** teacher to **MobileNetV2** student using limited computational resources. The efficacy of ARD is evaluated under settings with only **1 GPU** (T4 GPU) and **13GB RAM** for up to 6 hours a day.

Notably, competitive adversarial robustness is attained using very few gradient attack steps. This improves training efficiency crucial for edge ML. Appropriately balancing hyperparameters also allows robust accuracy over 50% **using just 1 attack step**. Overall, the presented approach advances the feasibility of performing robust distillation effectively even with accessibility constraints.

The democratized and reproducible method on Google Colab serves as a launchpad for those aiming to reap the advantages of edge intelligence. By sharing models protected against adversarial threats, this work propels broader adoption of **trustworthy** ML at society's technological edges.

1 INTRODUCTION

Key Words Adversarial Robustness, Edge AI, Edge Learning, Robust Distillation, Adversarial Robust Distillation, Trustworthy Machine Learning, Distillation on IOT devices

Edge Machine Learning (Edge ML) has emerged as a transformative paradigm in the field of machine learning, offering solutions tailored for scenarios where computational resources are constrained, and (near) real-time processing is crucial. Unlike traditional cloud-centric approaches, Edge ML involves deploying machine learning models directly on edge devices, bringing intelligence closer to the data source. This paradigm shift is driven by the growing demand for efficient and decentralized processing in applications ranging from Internet of Things (IoT) devices to edge servers.

The main research question is whether **can we reduce the attack steps** and get same results as more steps. When we are talking about small companies, daily usages of AI and so on, we should notice that there is not a huge amount of resources. The computational resources, and time, are limited, so it's important to use AI/ML models on Edge devices. In this workshop paper, we introduce a setup in which we can perform adversarial robust distillation under very limited GPU, RAM and Memory resources, and to ensure doing that, we use Google Colab as the running environment.

1.1 KEY CHARACTERISTICS OF EDGE MACHINE LEARNING

The key characteristics of an edge machine learning protocol are as follows:

[*]Mohammad Javad Maheronnaghsh

- **Limited Computational Resources**

  Edge devices often possess constrained computational resources, including processing power, memory, and energy. Edge ML algorithms are designed to operate efficiently within these limitations, enabling on-device model inference without heavy reliance on cloud services. 4

- **Real-time Processing**

  Edge ML emphasizes **low-latency** and (near) real-time processing, making it suitable for applications where immediate decision-making is critical 5. This is particularly beneficial in scenarios such as autonomous vehicles, smart surveillance, and healthcare monitoring.

- **Privacy and Security**

  Processing data locally on edge devices enhances privacy by reducing the need for transmitting sensitive information to centralized servers. Edge ML models are developed with a focus on maintaining data security and privacy 6, aligning with regulatory requirements and user expectations.

- **Diversity of Edge Devices**

  Edge ML caters to a diverse range of devices, including IoT sensors, smartphones, and edge servers. Adaptable algorithms and model architectures are essential to address the heterogeneity of edge computing environments 7.

2 RELATED WORKS

The advent of machine learning applications at the edge, particularly in resource-constrained environments, has spurred research into techniques to enhance the performance and robustness of models deployed in such settings. In ”Generative Adversarial Super-Resolution at the edge with knowledge distillation” by the Engineering Applications of Artificial Intelligence, a novel approach called EdgeSRGAN is introduced. This method leverages Generative Adversarial Networks (GANs) to achieve real-time super-resolution, essential for tasks like robotic monitoring and teleoperation. By optimizing model architecture and employing knowledge distillation, EdgeSRGAN achieves impressive execution speeds on edge devices without compromising image quality. **?**

In a similar vein, ”Boosting Accuracy and Robustness of Student Models via Adaptive Adversarial Distillation” presented at CVPR 2023 addresses the vulnerability of student models to adversarial attacks at the edge. Traditional methods like adversarial training struggle with compressed networks. To tackle this challenge, Adaptive Adversarial Distillation (AdaAD) is proposed. AdaAD involves the teacher model in the distillation process, adaptively guiding the student model towards improved accuracy and robustness. Remarkably, AdaAD outperforms existing methods, as demonstrated by the ResNet-18 model achieving top-tier performance in robustness benchmarks. 11

Furthermore, ”Improving Adversarial Robustness via Information Bottleneck Distillation” from the 37th Conference on Neural Information Processing Systems (NeurIPS 2023) delves into the information bottleneck principle to enhance model robustness. This study introduces Information Bottleneck Distillation (IBD), a novel distillation technique that capitalizes on insights from robust pretrained models. By optimizing information bottlenecks through soft-label distillation and adaptive feature transfer, IBD significantly bolsters adversarial robustness. Extensive experiments showcase the effectiveness of IBD against state-of-the-art attacks like PGD and AutoAttack, underlining its potential for strengthening model defenses in real-world scenarios. 12

These three papers collectively highlight the importance of advancing machine learning techniques tailored for edge deployment, emphasizing the need for efficient, robust, and lightweight models capable of withstanding adversarial pressures.

We gain information from other 4 papers: 13; 1; 2; 3. The landscape of Edge Machine Learning has seen significant advancements, with researchers addressing challenges and proposing innovative solutions. The following references highlight key contributions in the realm of Edge ML, covering aspects such as model optimization, federated learning, and adversarial robustness:

2.1 Model Optimization for Edge Devices

Efforts in model optimization aim to create lightweight architectures suitable for deployment on resource-constrained edge devices. Techniques such as quantization, pruning, and knowledge distillation are explored to achieve a balance between model accuracy and computational efficiency 8.

2.2 Federated Learning in Edge Environments

Federated Learning has gained traction as a decentralized training approach, where models are trained collaboratively across edge devices without exchanging raw data. This mitigates privacy concerns and aligns with the distributed nature of edge computing 9.

2.3 Adversarial Robustness in Edge ML

As edge devices are susceptible to adversarial attacks, ensuring the robustness of machine learning models is crucial. Techniques like Adversarial Training and Robust Distillation aim to enhance the resilience of models against adversarial **perturbations**, making them well-suited for deployment in **security-critical edge applications**.

3 Methodology

3.1 Problem Setup

$$\mathcal{L}_{KD}(\theta) = \mathbb{E}_{X\sim D}\Big[\alpha KL\big(S_\theta^t(X), T^t(X)\big) + (1-\alpha)\ell\big(S_\theta(X), y\big)\Big] \tag{1}$$

Adversarial robust distillation aims to train a student neural network S_θ with parameters θ that is robust to adversarial attacks by distilling knowledge from a teacher network T that has been made robust through adversarial training. The goal is to minimize the loss function $\mathcal{L}KD(\theta)$ which is a weighted combination of the knowledge distillation loss $KL(S\theta^t(X), T^t(X))$ between the student and teacher outputs and the standard cross-entropy loss $\ell(S_\theta(X), y)$ between the student output and true label y. Here, X is an input sampled from data distribution D, t is a temperature parameter, and α balances the two loss terms. The knowledge distillation loss encourages the student to mimic the teacher output while the cross-entropy loss trains the student to predict the correct labels.

Adversarial training minimizes the loss $\mathcal{L}\theta(X + \delta, y)$ where δ is a small adversarial perturbation constrained by norm $|\delta|\infty < \epsilon$. It encourages the model to be robust within an ϵ-ball around data points. Combining distillation and adversarial training loss functions allows transferring robustness from a teacher to student network.

3.2 Experimental Setup

Our experiments involve the distillation process from a WideResNet34 (WR34) teacher model to a MobileNetV2 (MN2) student model. The key parameters include the number of **epochs**, **learning** rate, adversarial attack **settings** (e.g., attack steps), and the architecture of the models involved.

4 Algorithm

Adversarial Robust Distillation (ARD) is a clever technique for training machine learning models to be both smart and resilient. In a nutshell, it pairs a less savvy student model with a smart and robust teacher. During training, the student learns not only by mimicking the teacher's predictions but also by toughening up against tricky inputs, known as adversarial examples. These tricky inputs are designed to deceive the model, but ARD helps the student become more resistant to such attempts. The algorithm cleverly combines two learning aspects: distillation, where the student gains knowledge from the teacher, and adversarial training, where it builds a defense mechanism against tricky inputs. In the end, the student model becomes not just intelligent like the teacher but also street-smart, able to handle attempts to confuse or mislead it with deceptive inputs.

Adversarial Robust Distillation (ARD) employs a combination of carefully designed loss functions and optimization strategies to refine the student model. In the distillation process, the algorithm calculates the distillation loss by comparing the student's predictions to those of the teacher, essentially encouraging the student to replicate the teacher's reasoning. Simultaneously, the adversarial training phase introduces an adversarial loss by evaluating how well the student handles deceptive inputs, emphasizing robustness. The total loss, a weighted sum of both distillation and adversarial losses, serves as the optimization objective. Through iterations, the algorithm fine-tunes the student model's parameters to minimize this total loss, striking a balance between accurate predictions and resilience to adversarial challenges. The careful interplay of distillation and adversarial training, coupled with meticulous optimization, results in a student model that not only mirrors the teacher's proficiency but also exhibits enhanced resistance to potential manipulations, making it a more reliable and secure predictor in real-world scenarios.

Here you can see the training algorithm:

Algorithm 1 Adversarial Robust Distillation

```
 1: **Input:** Training dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, teacher model $T$, student model $S$, hyperparameters $\lambda$, $\epsilon$
 2: **Initialize:** $S$ with random parameters
 3: **for** each iteration **do**
 4:     Sample a mini-batch $\{(x_i, y_i)\}_{i=1}^{B}$ from $\mathcal{D}$
 5:     Compute the logits $T(x_i)$ from T
 6:     $\tilde{x}_i = x_i + \text{FGSM}(x_i, T)$
 7:     Compute logits $S(x_i)$ from S
 8:     $\mathcal{L}_{\text{distill}} = \text{CrossEntropy}(S(x_i), T(x_i))$
 9:     Compute logits $S(\tilde{x}_i)$ from S
10:     $\mathcal{L}_{\text{adv}} = \text{CrossEntropy}(S(\tilde{x}_i), y_i)$
11:     $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{distill}} + \lambda \cdot \mathcal{L}_{\text{adv}}$
12:     Update the $S$ parameters (by minimizing $\mathcal{L}_{\text{total}}$)
13: **end for**
```

5 Experiments & Results

5.1 Abbreviations

Here are the abbreviations used in the rest of this part:

Abbreviation	Main Term
Epo	epoch number
MTL	Mean Train Loss after epoch 1
Nat	Natural Accuracy after epoch 1
R	Robust Accuracy after epoch 1
MTL'	Mean Train Loss after final epoch
Nat'	Natural Accuracy after final epoch
R'	Robust Accuracy after final epoch
AS	Attack Step
LR	Learning Rate

Table 1: Abbreviations

5.2 Results

Table 2 summarizes the experimental results, showcasing variations in Mean Train Loss (MTL), Natural Accuracy (Nat), and Robust Accuracy under different configurations. The experiments ex-

Table 2: Experiment Results: You can find more about the experiment results here. α and ϵ are 1.0 and 8.0/255 respectively. The teacher and student model are WideResNet34 and MobileNetV2 respectively.

Epo	AS	StepSize	LR	MTL	Nat	R	MTL'	Nat'	R'
10	10	2.0 / 255	0.05	0.181076	33.56	17.41	0.080974	55.79	29.94
10	10	2.0 / 255	0.25	0.146695	41.3	24.86	**0.061529**	**67.02**	**38.23**
10	10	2.0 / 255	0.45	0.146934	43.64	23.95	0.068444	63.5	34.38
10	10	2.0 / 255	0.65	0.144871	41.3	24.8	0.074837	60.4	30.76
10	10	2.0 / 255	0.85	0.143052	43.43	24.23	0.082100	56.52	31.29
20	5	2.0 / 255	0.05	0.134497	44.12	28.17	**0.035455**	**74.67**	**46.92**
20	5	2.0 / 255	0.25	0.128324	47.52	27.74	0.037097	74.36	46.26
20	5	2.0 / 255	0.45	0.128422	45.65	27.73	0.044314	70.5	44.77
20	5	2.0 / 255	0.65	0.131341	44.85	27.16	0.050490	66.99	41.47
20	5	2.0 / 255	0.85	0.148691	38.22	25.41	0.055780	67.57	41.42
10	10	8.0 / 255	0.05	0.143248	42.72	24.37	0.057875	66.28	37.05
10	10	8.0 / 255	0.25	0.133477	45.38	27.21	**0.052860**	**69.86**	**39.3**
10	10	8.0 / 255	0.45	0.136056	45.13	26.14	0.057719	67.37	38.48
10	10	8.0 / 255	0.65	0.137116	44.78	25.61	0.062608	65.59	37.85
10	10	8.0 / 255	0.85	0.151821	41.04	23.91	0.067955	64.07	34.03
10	10	10.0 / 255	0.05	0.138305	43.5	25.49	0.057344	66.75	36.92
10	10	10.0 / 255	0.25	0.131707	48.04	27.24	**0.051897**	**69.43**	**39.76**
10	10	10.0 / 255	0.45	0.132291	46.98	25.0	0.058111	67.81	39.13
10	10	10.0 / 255	0.65	0.139062	43.18	24.81	0.063288	64.58	36.03
10	10	10.0 / 255	0.85	0.146160	41.47	23.39	0.068344	61.31	35.03
70	1	8.0 / 255	0.05	0.124786	46.72	29.2	**0.015586**	**81.16**	**58.17**
60	1	8.0 / 255	0.25	0.118177	50.22	30.91	0.026345	78.19	54.09
70	1	8.0 / 255	0.45	0.123382	47.0	30.21	0.034833	75.4	49.63
65	1	8.0 / 255	0.65	0.123316	47.36	29.43	0.041720	70.23	46.54
70	1	8.0 / 255	0.85	0.127801	43.9	27.67	0.047669	67.89	44.7
30	1	10.0 / 255	0.05	0.128903	45.13	26.41	**0.026308**	**77.0**	**51.39**
30	1	10.0 / 255	0.25	0.121239	48.07	28.53	0.031888	75.85	49.13
30	1	10.0 / 255	0.45	0.124347	48.33	29.57	0.039394	69.62	44.6
30	1	10.0 / 255	0.65	0.124840	47.27	28.43	0.046567	69.87	43.78
30	1	10.0 / 255	0.85	0.137199	44.36	27.67	0.051715	68.12	42.79
25	2	4.0 / 255	0.05	0.127732	45.67	29.62	**0.027432**	**76.52**	51.09
25	2	4.0 / 255	0.25	0.121206	47.25	28.85	0.031048	76.22	**52.44**
40	2	4.0 / 255	0.45	0.125599	47.23	29.58	0.036818	72.8	49.96
20	2	4.0 / 255	0.65	0.123027	46.14	28.81	0.045312	71.74	45.26
25	2	4.0 / 255	0.85	0.124504	42.1	27.37	0.050439	68.81	45.02

plore the impact of varying parameters such as epochs, learning rates, and attack settings. Notably, configurations achieving better Robust Accuracy are highlighted for each experimental setup. [1]

5.3 Performance Metrics

To evaluate the effectiveness of our Adversarial Robust Distillation (ARD) process, we employ two crucial metrics: Natural Accuracy and Robust Accuracy. The variations in these metrics across different experimental setups provide insights into the trade-offs and benefits associated with specific configurations.

5.4 Applications

Our proposed methodology, executed within the accessible resources of Google Colab, opens avenues for widespread adoption. Users **globally can reproduce our results**, benefiting from fast and free distillation in machine learning. This democratization of distillation processes enables broader accessibility to advanced machine learning techniques.

[1]The checkpoint of WRN34 trained by TRADES on CIFAR10 could be found HERE.

5.5 Efficacy of Low Attack Steps

Several experimental configurations in Table 2 explore adversarial robust distillation using just 1 attack step. These low step settings aim to improve **training efficiency** for resource-constrained edge devices.

Notable low attack step outcomes include:

- With 1 step, 70 epochs, α=1.0, ϵ=8/255, step size=8/255 and LR=0.05, robust accuracy reaches 58.17%. This is the overall best robust accuracy achieved.
- Using 1 step, 65 epochs, α=1.0, ϵ=8/255, step size=8/255 and LR=0.65, robust accuracy of 46.54% is obtained.
- Settings of 30 epochs, 1 step, α=1.0, ϵ=8/255, step size=10/255 and LR=0.05 attain 51.39% robust accuracy. This significantly outperforms the 38.23% robustness scored using 10 steps and 25 epochs.

The above examples demonstrate that with appropriate tuning of epochs and learning rates, the low 1-step attack configurations can achieve adversarial robustness **on par or even better than** the longer 10-step attack settings.

Crucially, the low attack step models require **fewer gradient computations per epoch**. This directly translates to faster training times, aligning with the goals of efficient edge ML deployment.

In summary, the results indicate that by balancing hyperparameters, competitive adversarial robustness can be realized using **very few attack steps**. This finding opens avenues for performing robust distillation efficiently even on **resource-constrained edge devices**.

To comprehensively assess the performance of our Adversarial Robust Distillation (ARD) process, we rely on two key metrics: Natural Accuracy and Robust Accuracy. These metrics serve as critical indicators of the model's proficiency under standard and adversarial conditions, respectively.

- **Natural Accuracy Changes**

  Natural Accuracy refers to the model's accuracy on clean, unaltered data. In our experiments, we observe variations in Natural Accuracy across different configurations. Noteworthy trends include improvements or trade-offs in Natural Accuracy based on the chosen parameters, such as learning rates and attack settings.

- **Robust Accuracy Changes**

  Robust Accuracy, on the other hand, measures the model's resilience against adversarial attacks. The variations in Robust Accuracy provide insights into the model's ability to maintain performance in the presence of perturbations. We specifically highlight configurations that exhibit superior Robust Accuracy, showcasing the effectiveness of our ARD approach in enhancing the model's adversarial robustness.

6 Applications

As we introduced before, everyone all around the world, has free access to Google Colab resources that we used for approximately 4-6 hours a day, so anyone can use these settings to reproduce our results and take advantage of Fast and Free Distillation in Machine Learning.

The evaluation metrics have direct implications for practical applications of machine learning models, especially in edge computing scenarios. The observed changes in Natural and Robust Accuracy offer valuable information for **selecting configurations** that strike a balance between standard performance and robustness against adversarial threats.

As demonstrated by our experiments, the ARD process enables the deployment of models with enhanced robustness on resource-constrained edge devices. This has significant implications for **real-world** applications where the availability of computational resources is limited.

7 Conclusion

In conclusion, by adopting the settings outlined in Table 2, similar results to state-of-the-art benchmarks can be achieved. The presented methodology offers a practical solution for performing adversarial robust distillation under resource constraints. As the experiments demonstrate, the approach **balances** computational efficiency and model performance, making it suitable for edge AI applications.

8 Future Works and Discussion

The potential for further exploration lies in **varying parameters** more extensively, conducting **experiments** on different datasets, and expanding the scope of applications. Future research endeavors could delve into optimizing the ARD process for diverse scenarios, thereby contributing to the evolution of edge machine learning.

9 Code Access

For access to the experiments and results, please refer to our GitHub repository: https://github.com/mjmaher987/Robustness—CISPA Any inquiries or discussions are welcomed by contacting the authors.

Acknowledgments

We express our gratitude to the CISPA Helmholtz Institute of Information, especially from Professor Xiao Zhang and Minxing Zhang.

References

Explaining and harnessing adversarial examples.

Boosting accuracy and robustness of student models via adaptive adversarial distillation.

Improving adversarial robustness via information bottleneck distillation.

Adversarially robust distillation.

Fast is better than free: Revisiting adversarial training.

On fast adversarial robustness adaptation in model-agnostic meta-learning.

Efficient acceleration of deep learning inference on resource-constrained edge devices: A review.

Edge machine learning for ai-enabled iot devices: A review.

Data security and privacy-preserving in edge computing paradigm: Survey and open issues.

Resource management in fog/edge computing: A survey on architectures, infrastructure, and algorithms.

A survey on optimization techniques for edge artificial intelligence (ai).

Federated learning in edge computing: A systematic survey.