Can Deep Networks Be Highly Performant, Efficient And Robust Simultaneously?

Madan Ravi Ganesh[1,*2]
madan.raviganesh@us.bosch.com
Salimeh Yasaei Sekeh[3]
salimeh.yasaei@maine.edu
Jason J. Corso[2]
jjcorso@umich.edu

[1] Bosch Center for Artificial Intelligence
2555 Smallman Drive,
Pittsburgh, Pennsylvania, USA

[2] University of Michigan
500 S State Street,
Ann Arbor, Michigan, USA

[3] University of Maine
168 College Avenue,
Orono, Maine, USA

Abstract

Performance is not enough when it comes to deep neural networks (DNNs); in real-world settings, computational load or efficiency during training and adversarial security are just as or even more important. Often there are critical trade-offs to consider when prioritizing one goal over the others. Instead, we propose to concurrently target **P**erformance, **E**fficiency, and **R**obustness, and ask just how far we can push the envelope on simultaneously achieving these goals. Our algorithm, CAPER, follows the intuition that samples that are highly susceptible to noise strongly affect the decision boundaries learned by DNNs, which in turn degrades their performance and adversarial robustness. By identifying and removing such samples, we demonstrate increased performance and adversarial robustness while using only a subset of the training data, thereby improving the training efficiency. Through our experiments, we highlight CAPER's high performance across multiple Dataset-DNN combinations, and provide insights into the complementary behavior of CAPER alongside existing adversarial training approaches to increase robustness by over 11.6% while using up to 4% fewer FLOPs during training.

1 Introduction

The ability to learn patterns from large-scale data while not requiring explicit analytical modelling has made deep neural networks (DNNs) quite popular in recent years. Increasing performance has been the de facto emphasis when developing DNN-based solutions. However, to deal with the rigors of the real world, DNNs should not only be 1) highly accurate, but 2) efficient to develop, and 3) robust to adversaries as well. Since each of these properties fulfill unique targets they are often handled separately, with known trade-offs when prioritizing one property over the others. Ideally, by jointly constraining the development process to satisfy Performance (P), Efficiency (E). and Robustness (R), or PER goals, we can deliver highly accurate, more secure DNNs faster and at a lower computational cost.

* indicates work done when affiliated with

To the best of our knowledge, there are no works that simultaneously address all three PER goals. Often, DNN solutions focus on improving a chosen performance metric while efficiency and adversarial robustness become afterthoughts. Among approaches that focus on efficiency, distributed training [8, 25] techniques often assume the availability of large-scale hardware while low-precision computations [12, 33] rarely match the potential of their high-precision counterparts. In the adversarial domain, multiple works under the umbrella of adversarial training acknowledge the trade-off between improving robustness to adversaries and maintaining high performance [7, 26]. Only a small subset of works within this domain attempt to address the idea of efficiently imparting robustness [29, 38, 40]. The scope of these works is restricted to the choice of the algorithm used to generate adversaries while retaining the entire training dataset in memory. This leads to overheads during loading, preprocessing, and training. A common theme across these different categories of solutions is their focus on tackling at most two of the three desired PER goals.

Using our algorithm CAPER we propose to Concurrently Achieve Performance, Efficiency, and Robustness. Our algorithm is built on the assumption that there exists a subset of the original training data that negatively impacts the learning process of the model being trained [20]. In CAPER, we identify and remove this subset of data which leads to a direct improvement in efficiency during the training phase. By using a function of the distance between features, specifically between the original inputs and their noise-perturbed counterparts, we identify the subset of data that is highly susceptible to noise. We hypothesize that samples that are highly susceptible to noise force irregular behaviours in the DNN and have a strong impact on the learning process. By removing these samples, we regularize the learning process and encourage improvements in generalization and robustness of the model. In addition, we highlight the difference between how robustness is imparted to a DNN when using CAPER and standard adversarial training [3, 24, 36], and how they can be combined. To summarize, our contributions in this paper are,

- CAPER, a new methodology that simultaneously targets improved performance, efficiency in the training phase, and robustness to adversarial attacks,
- Improved robustness and clean accuracy across a variety of settings and adversaries,
- Complementary behavior alongside adversarial training regimes to boost robustness.

2 Related Works

2.1 General Curriculum Learning

CAPER's idea of retaining a subset of the training data draws inspiration from conventional curriculum learning, which is defined as an approach to organizing and presenting data to machine learning models to improve their learning process and performance. Initially, a crucial point of emphasis was their fast convergence to a high-quality solution [1, 11]. Subsequent works focused on various ways to organize and schedule data while relaxing the constraint on faster convergence [10, 13, 17, 44]. More recently, there has been a strong emphasis on using feedback from the model being trained to modify the training regime and reduce the data used to train the model [45, 46]. In general, curriculum-based approaches emphasize improvements in generalization performance, with lower attention on adversarial robustness. In this work, we consider adversarial robustness a key trait required of DNNs.

From a methodological point of view, our approach uses additive noise to identify and remove samples that create adversarial vulnerability. This is distinct from the use of gradients, *loss value, predictions, or a change in those values to identify difficult samples* [22, 31, 35, 44]. Additionally, in CAPER we use hard sampling to permanently remove samples from the training set instead of recycling them during the training phase [45]. Our approach is similar to the hard sampling performed in Lapedriza *et al*. [20] and Chitta *et al*. [5], without the need to train a proxy network to learn which samples need to be removed [32]. Furthermore, since our approach focuses on differences in the feature space as the primary means to highlight and remove samples, it is easily extensible to different architectures and applications.

2.2 Adversarial Training

Adversarial training approaches expose a DNN to a variety of adversarial perturbations during the training phase to increase its robustness [24, 42]. Examples of such adversarial training approaches include gradually increasing the strength of adversaries to improve robustness [2], using the least adversarial data among confidently misclassified samples [43], and others [39]. Often such methods provide insufficient time or efficiency comparisons and have strong trade-offs on their performance on clean testing data. A small subset of works focus on balancing the trade-off between clean and robust accuracies by using early stopping [26] or a student-teacher setup to learn from smooth logit distributions [7]. While these approaches focus on improving both clean and robust accuracies, their test bed does not cover a wide range of adversaries, thus limiting the scope of their study. In this work, we evaluate them against several different adversaries and contrast their performance against our algorithm from both a performance and efficiency standpoint.

A more recent line of works tackle the problem of efficient adversarial training. Wang *et al*. [38] propose a dynamic and efficient adversarial training methodology that automatically learns to adjust the magnitude of perturbations during the training process. Although their work is insightful, their results are limited to fixed DNN backbones. Shafahi *et al*. [29] offer an inexpensive alternative to recycling gradient computations performed during backpropagation to generate adversarial examples. Wong *et al*. [40] review FGSM-based adversarial training and offer multiple suggestions that extend FGSM's viability to quickly obtain highly robust DNNs. Each of the above methods that propose a more efficient adversarial training approach focus on modifying the algorithm used to generate adversaries while retaining the complete training set. *However, in CAPER we address training efficiency by directly reducing the training data available, thus offering a complementary approach that can work alongside any traditional or efficient adversarial training algorithm.*

3 CAPER

3.1 Proposed Algorithm

CAPER focuses on removing a subset of the training data that negatively impacts performance; see Fig. 1 for an overview. We begin by training a DNN using the complete training dataset up to τ epochs. At the chosen epoch $\tau << E$, where E is the total number of training epochs, we compare the distance between features of standard inputs and their noise-perturbed counterparts. Here, the noise-perturbed counterparts are generated using additive gaussian noise on the input images.

It is well known that DNNs are excellent function approximators but struggle to extrapolate to data points at the decision boundaries or outside of a known domain. Following

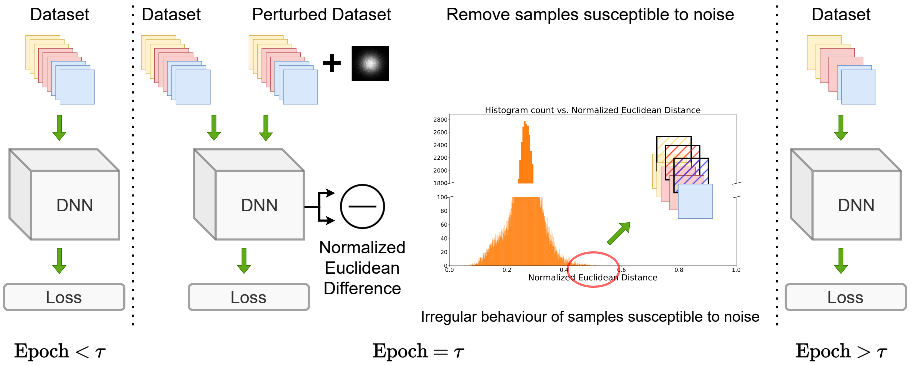


Figure 1: CAPER: At a chosen epoch $\tau << E$, we compare the feature embeddings between original inputs and their noise perturbed counterparts. A larger distance between feature embeddings indicates samples highly susceptible to noise. We remove such samples, update the dataset and continue training using the remaining subset of data.

this intuition, we know that DNNs respond irregularly when they encounter data they cannot comprehend well. By injecting noise into the input and measuring the distance between features, pre-, and post-injection, we can identify samples that have a strong disparity in responses from the DNN. Such responses are characteristic of data points that are close to the decision boundaries and have a strong impact on shaping them. The notion of injecting noise to assess the quality of samples has been used previously to detect OOD [21] and adversarial examples [23] when provided with a trained model. *However, we distinguish our work by using noise-injection as a method to refine a given dataset during the training phase, thus affecting the evolution of the DNN and improving the overall robustness of a model trained from scratch.*

In our algorithm, since a large distance between features readily identifies samples that are susceptible to noise, removing them allows the DNN to learn better decision boundaries from a more regularized set of data, leading to improved generalization and adversarial robustness. In our approach, we use the distance values to generate a binary mask and remove those samples from the dataset while we continue training using the remaining subset of data. It is important to note that we do not make an assertion about a change in loss/prediction values when comparing the distance between features. Instead, we opt for a relative comparison between feature distances to remove a total of γ samples.

3.1.1 Basic Setup

In CAPER, we remove the noisy subset of samples from the training data which in turn reduces the total amount of data stored and used during training. Mathematically, we describe this process as masking contributions from the noisy subset of data after epoch τ:

$$\mathcal{L}(X,Y) = \min_{W} \frac{1}{||m||_0} \sum_{i=1}^{N} m_i \ell_\varepsilon(F(x_i), y_i). \quad (1)$$

Here, $\{(x_i, y_i)\}_{i=1}^{N} \sim (X,Y)$, denote the input variables, where N represents the total number of samples, $F(\cdot)$ denotes the output of the entire DNN, ℓ_ε is the cross-entropy loss modified by label smoothing [34] and $m \in \{0,1\}^N$ is the binary mask vector defined using our heuristic, based on the distance between features. Once we determine the value of m at epoch τ, it remains fixed throughout the remaining training epochs. A small value of τ, e.g., 1 or 2, would force the capture of features that are not coherent while large values of τ, like 200, would significantly reduce the efficiency gain we expect. Instead, we choose a relatively small but balanced value for τ to obtain coherent features and maximize our gain in efficiency.

3.1.2 Noise Injection: Capturing Feature Disparity

To ascertain the value of m, we begin by capturing the distance between features, specifically between the original input and their noise-perturbed counterparts, at a chosen epoch τ across a chosen layer in the DNN. To generate the noise-perturbed counterparts, we apply additive gaussian noise to the input. We denote the capture of features from a desired layer l as,

$$f^{(l)}(x_i^{(l)}) = \sigma(W^{(l)}x_i^{(l)} + b^{(l)}),\ f^{(l)}(x_i^{(l)} + \delta_i) = \sigma(W^{(l)}(x_i^{(l)} + \delta_i) + b^{(l)}). \tag{2}$$

Here, assuming an activation function $\sigma()$, $f^{(l)} \in \mathbb{R}^{N\times O^{(l)}\times h^{(l)}\times w^{(l)}}$, where $O^{(l)}$ denotes the output dimension of layer l, $h^{(l)}$, $w^{(l)}$, $W^{(l)}$, and $b^{(l)}$ represent the output height, width, weights and biases of layer l, respectively. In addition, $\delta_i \sim \mathcal{N}(0, 0.5)$, with dimensionality matching the input. Note: To avoid inconsistencies between the effects of applying δ_i independently at multiple layers, we apply δ_i to the image directly and observe its effects at downstream layers. We drop the layer superscript to improve readability hereon.

Once we obtain the features from the chosen layer, we compute the distance $D(.)$ between corresponding pairs of features as,

$$\Delta f(i) = D(H(f(x_i)), H(f(x_i + \delta_i))) = ||H(f(x_i)) - H(f(x_i + \delta_i))||_2, \tag{3}$$

where $H(.)$ is a projection function that maps the features into a lower dimensional space, and $\Delta f(i) \in \mathbb{R}^{1\times O^{(l)}}$. The function $H : \mathbb{R}^{O^{(l)}\times h^{(l)}\times w^{(l)}} \to \mathbb{R}^{O^{(l)}\times P}$, where $P << h^{(l)} \times w^{(l)}$ and $O^{(l)}$ denotes the filter counts from layer l. While Eq. 3 depicts the l_2-norm version of the distance function, the formulation itself is not limited to it. Beyond capturing the distance, we further normalize $\Delta f(i)$ values across samples to ensure that the distances remain comparable. We propose normalizing them on a channel-wise basis using the following equation,

$$\Delta\hat{f}(i,q) = \frac{\Delta f(i,q) - \min\limits_{n\in 1,\ldots,N} \Delta f(n,q)}{\max\limits_{n\in 1,\ldots,N} \Delta f(n,q) - \min\limits_{n\in 1,\ldots,N} \Delta f(n,q)}. \tag{4}$$

Here, $i \in \{1,2,\ldots,N\}$, $q \in \{1,2,\ldots,O^{(l)}\}$, and $\Delta\hat{f}(i,q) \in [0,1]$.

3.1.3 Binary Mask Computation

While $\Delta\hat{f}$ captures the distance between features from filters of a specific layer, we need a simple measurable value that compares the susceptibility of various samples to noise. For this purpose, we include ξ_i, the instability of a sample measured as the average $\Delta\hat{f}$ across filters in a given layer.

$$\xi_i = \frac{\sum_{q=1}^{O^{(l)}} \Delta\hat{f}(i,q)}{O^{(l)}},\ \ i \in \{1,\ldots,N\}. \tag{5}$$

Using the instability values, we compute m as:

$$m_i = \begin{cases} 0 & \text{if } \xi_i \text{ is in the top } \gamma \text{ values of } \xi \\ 1 & \text{o.w }. \end{cases} \tag{6}$$

By controlling γ, we use m_i to reduce the amount of the training data held in memory as well as the overall FLOPs required during training. Once m is applied, the DNN is then trained with the remaining subset of data from epochs τ to E.

4 Experimental Results

4.1 Setup

Datasets and DNNs We use four primary datasets to evaluate our proposed method, CIFAR-10, CIFAR-100 [19], miniImagenet [37] and ILSVRC2012 [27]. Among these datasets, we restrict our adversarial robustness comparisons to CIFAR-10/100 to match existing literature. For miniImagenet, we use a custom-generated and balanced training-and-testing split that we will make available with our code. We use four DNN architectures to evaluate CAPER in the context of standard curriculum learning, VGG16 [30], MobileNet [28], DenseNet [15, 16] and ResNet50 [14]. In addition to these architectures, we use ResNet18 and PreActResNet18 in adversarial robustness comparisons. We choose these networks to help represent a wide variety of architectural backbones. Each of the four main DNNs have two distinct versions, one suitable for the CIFAR datasets and another for the remaining datasets.[1]

Adversarial Attacks And Metrics We explore the effect of a variety of adversarial attacks like MIFGSM [9], FFGSM [40], DI2FGSM [41], APGDDLR [6], APGDCE, PGD [24] and CW [4] using the code from [18, 43]. To measure the performance of various algorithms, we use standard **Accuracy**(%) over the testing set. For adversarial robustness we measure **Robust Accuracy**(%) over the perturbed testing set, illustrated by the radius of polar plots. Finally, we use total **FLOPs**, measured as 1 pass over the entire DNN scaled across the entire training phase, to compare the improvement in efficiency across different training methods. Across all experiments, we provide average statistics over 5 trials, with the exception of Rice *et al.* [26]-, Cui *et al.* [7]- and Shafahi *et al.* [29]-based experiments, with numbers in **bold** referring to the best performance and <u>underline</u> referring to the second best.

CAPER: Hyper-parameters Within CAPER, τ is an extremely important parameter that influences the amount of efficiency gain we expect. For experiments in Sec. 4.2, we set $\tau = 50$ for all DNN-Dataset combinations except ResNet50-CIFAR-10, for which we set it to 100. Results on ILSVRC12 were generated using $\tau = 15$. Experiments under Sec. 4.3 use $\tau = 35$ and 15 when comparing against Rice *et al.* [26] and Cui *et al.* [7] respectively, and the remaining use $\tau = 50$. Throughout our experiments, we fix $H(.)$ as the mean value across the $h^{(l)} \times w^{(l)}$ channels and capture features from the last convolution layer. The value of γ is listed in () within each experimental subsection.

[1] Detailed descriptions of these model variants are provided in our code base https://github.com/MichiganCOG/Q_TART.

DNN	Algorithm	CIFAR-10	CIFAR-100	miniImagenet
VGG16	Mini-batch SGD	<u>94.04</u>	<u>74.23</u>	<u>70.95</u>
	Random	93.19	71.63	67.57
	DIHCL	94.03	72.89	66.07
	CAPER(Ours)	**94.44** ($\gamma = 2.5$k)	**74.97** ($\gamma = 1.25$k)	**71.23** ($\gamma = 2.5$k)
MobileNet	Mini-batch SGD	<u>93.50</u>	<u>72.75</u>	<u>64.62</u>
	Random	92.31	71.15	62.11
	DIHCL	88.97	61.58	49.37
	CAPER(Ours)	**93.62** ($\gamma = 125$)	**73.49** ($\gamma = 10$k)	**65.96** ($\gamma = 5$k)
DenseNet	Mini-batch SGD	<u>95.13</u>	<u>76.95</u>	<u>73.78</u>
	Random	93.88	74.18	71.23
	DIHCL	94.72	76.03	64.34
	CAPER(Ours)	**95.20** ($\gamma = 100$)	**77.39** ($\gamma = 1.25$k)	**74.69** ($\gamma = 5$k)
ResNet50	Mini-batch SGD	95.63	79.27	<u>68.76</u>
	Random	95.27	76.71	64.69
	DIHCL	**95.83**	**79.71**	66.86
	CAPER(Ours)	<u>95.79</u> ($\gamma = 1$k)	<u>79.39</u> ($\gamma = 1.25$k)	**69.54** ($\gamma = 2.5$k)

Table 1: Across most datasets CAPER achieves the best performance when compared against Mini-batch SGD, DIHCL and Random baselines.

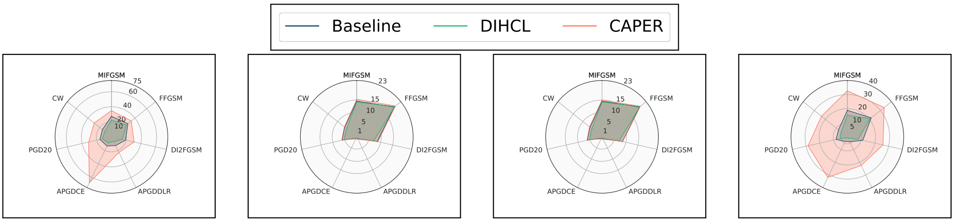
DNN	Algorithm	ILSVRC2012
ResNet50	Mini-batch SGD	76.32
	DIHCL	<u>76.33</u>*
	CAPER (Ours)	**76.62** ($\gamma = 11.7$k)

Table 2: CAPER achieves the best performance after we remove 11700 samples across 10 classes.* indicates numbers from authors. ILSVRC2012 results are from 1 trial.

4.2 Curriculum Comparison

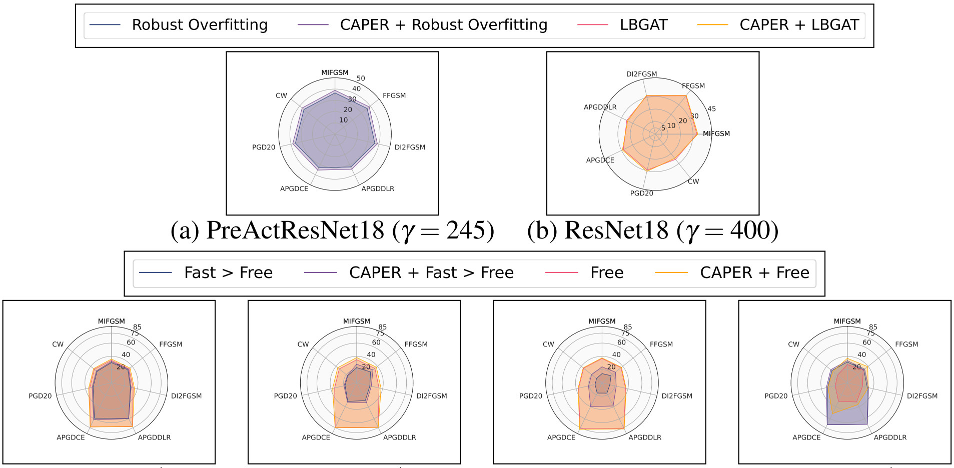
In this experiment, our main goal is to compare the performance of CAPER against mini-batch SGD training and highlight how we can improve performance while only retaining a subset of our training data. Additionally, we compare against a top performing curriculum learning method DIHCL [45] which prioritizes the removal of samples throughout the training process. From Table 1, across all combinations of datasets and DNN architectures, we observe that *CAPER easily outperforms mini-batch SGD, using a subset of the training data. To ensure a fair comparison, we used a common hyper-parameter setup.*

More interestingly, when we observe the performance of DIHCL adapted to our selection of Dataset-DNN pairs we see that it consistently exhibits strong performances on the ResNet architectures. This, in conjunction with DIHCL's propensity to perform significantly worse than even randomly removing the same number of samples as in CAPER (marked in Table as Random) across the other tested architectures points toward a strong affinity of the training setup used in DIHCL to residual architectures. Despite this, alongside the starkly different training setup used in DIHCL (cyclic learning rate, a teacher-like copy of the DNN, etc.), CAPER still improves upon DIHCL in most cases. This improvement is further highlighted when applying CAPER to the ILSVRC2012 dataset (Table 2), where we are able to



(a) VGG16 (b) MobileNet (c) DenseNet (d) ResNet50

Figure 2: Curriculum-based Comparison: CAPER matches and often significantly improves upon the adversarial robustness of mini-batch SGD (Baseline) training and DIHCL. Methods with the largest area of plot are preferred.



(a) $\gamma = 450/12$ (b) $\gamma = 250/125$ (c) γ=25/5 (d) $\gamma = 12/25$

Figure 3: Alongside both standard adversarial training (**Top**) and efficient adversarial training (**Bottom**), CAPER improves the overall robustness. Largest improvements from adversarial training are observed for MIFGSM, FFGSM, DI2FGSM, APGDCE, and PGD attacks. **Bottom** γ values are for CAPER+Wong *et al.* [40] / CAPER+Shafahi *et al.* [29] across VGG16, MobileNet, DenseNet and ResNet50 respectively.

significantly outperform DIHCL and mini-batch SGD, even after removing 11700 samples.

4.3 Adversarial Robustness

Curriculum-based Comparison Using results from Fig. 2 we establish two main observations, 1) in multiple instances DIHCL reduces the robustness of DNNs when compared to mini-batch SGD training, and more importantly 2) *CAPER significantly improves the robustness of DNNs to multiple adversarial attacks.* We hypothesize two possible reasons why DIHCL reduces the adversarial robustness of a variety of DNNs. First, the repeated sampling with replacement and steady decline in the number of available samples does not allow for a stable learning environment to help address adversarial robustness. Secondly, the use of gradients/loss/prediction values or their change has a direct impact on the set of samples removed and therefore the final adversarial robustness. A deeper dive into the correlation between the selection procedure and the final outcomes could help provide more insight.

DNN	Δ TFLOPs Reduced (% of original)		
	CIFAR-10	CIFAR-100	miniImagenet
VGG16	197 (4.17%)	59 (1.88%)	1840 (4.17 %)
MobileNet	3.55 (0.21%)	142 (15.0%)	43.2 (7.50%)
DenseNet	2.90 (0.15%)	40.3 (2.08%)	2550 (8.33%)
ResNet50	260 (1.33%)	406 (2.08%)	472 (4.17%)

Algorithm	Δ TFLOPs Reduced			
	VGG16	MobileNet	DenseNet	ResNet50
CAPER + Wong *et al.* [40]	9.83	3.55	1.61	3.9
CAPER + Shafahi *et al.* [29]	35.4	3.55	0.161	8.13

Table 3: Illustration of the improvement in efficiency offered by CAPER during the training phase for (**Top**) Curriculum-based, and (**Bottom**) Efficient Adversarial training. Baseline number of FLOPs is calculated using a forward pass through the DNN.

Standard Adversarial Training Comparison We use Rice *et al.* [26], with settings corresponding to their validation-based early stopping setup on CIFAR-10, and Cui *et al.* [7], with settings corresponding to ResNet18 for both natural and robust models on CIFAR-100, as representatives for standard adversarial training. When using CAPER alongside standard adversarial training, we observe an improvement in performance over the original adversarial training methods across most adversarial attacks, as shown in Fig. 3. We emphasize that these improvements are in addition to an increase in Accuracy(%), from 82.66% to 83.14% for PreActResNet18 and from 69.22% to 69.65% for ResNet18. While the original methods emphasize a balanced improvement in Robust Accuracy(%) and Accuracy(%), *the addition of CAPER atop these methods allows us to maintain their original benefits while further improving on their efficiency and adversarial robustness.*

Efficient Adversarial Training Comparison Our first observation based on the bottom row of Fig. 3 is the high level of robustness shown by all DNNs to APGDDLR and APGDCE attacks across both efficient adversarial training and CAPER-based training. In addition, when using CAPER-based adversarial training, DI2FGSM, MIFGSM, and FFGSM consistently show the largest magnitude of improvement. Finally, similar to the previous scenario's results on standard adversarial training, *adding CAPER atop common efficient adversarial training approaches further boosts their performance against all adversarial attacks.*

4.4 Time Efficiency Comparison

To understand the impact of CAPER on efficiency, we observe the number of FLOPs reduced by CAPER where for each Algorithm-Dataset-DNN triplet the FLOPs are computed using their respective hyper-parameter settings listed in the supplementary materials. From Table 3 we observe a strong increase in the number of FLOPs reduced across a variety of Dataset-DNN combinations when compared to standard mini-batch training. This includes the ILSVRC2012 dataset, where we save 4.08 PFLOPs or close to an entire epoch of training. Under the adversarial robustness setting, we observe a maximum reduction of 22.5 TFLOPs when combined with standard adversarial training algorithms and 35.5 TFLOPs when combined with efficient adversarial training algorithms (bottom Table 3). Overall,

DNN	γ 125	250	500	1250	2500
VGG16	71.40	71.50	71.52	71.52	71.23
ResNet50	69.77	69.70	69.63	69.68	69.54

Table 4: Holding $\varepsilon = 0.1$, we observe CAPER improves accuracy over mini-batch SGD (70.95 and 68.76) as we vary γ on the miniImagenet dataset.

CAPER successfully combines the benefits of performance, efficiency, and robustness.

5 Discussion

γ Selection With the values of ε and τ set to fixed number, we investigated $\gamma \in \{15000, 10000, 5000, 1250, 500, 250, 125, 50, 25, 12\}$. Typically, we obtain improved performance across most γ values up to the limit indicated in Tables 1 and 2. In Table 4 we present an abridged version of results indicating this behavior. Overall, the selection of γ is primarily guided by the choice of dataset, with miniImageNet and ILSVRC2012 showcasing a high amount of redundant samples while CIFAR-like datasets offer less flexibility, with a secondary influence from the choice of DNN. We deem the exploration of more efficient search strategies to identify γ and the variation of τ along the frontier of efficiency as part of future work.

Adversarial Response When comparing the performance of CAPER, with and without the addition of other adversarial training regimes, we find their performance across FFGSM, MIFGSM and DI2FGSM extremely similar, often within 5% of each other. The importance of this observation is further highlighted by the fact that we do no expose the model to any adversarial input during training. This outcome suggests that our approach could provide an inexpensive alternative to boosting performance across FGSM-based attacks while complementing existing adversarial training approaches.

6 Conclusion

Overall, we establish CAPER as an algorithm that simultaneously tackles improvements in performance, efficiency and adversarial robustness. The use of noise-injection in CAPER to identify and remove noisy samples helps modify the embeddings learned by DNNs in a favorable manner. In doing so, there is a strong improvement in classification accuracy achieved via a more efficient training process. We also establish high adversarial robustness by incorporating CAPER like a plug-and-play module atop existing adversarial training. An important direction of future work is exploring a variety of metrics to assess a comprehensive way to identify noisy samples. In addition, we plan to expand on the contributions from multiple layers of the DNN to study the impact of feature hierarchy on the final performance. Our goal is to jointly target PER in an effort to develop more cost and resource efficient training protocols, with a view to reducing the environmental impact of developing DNNs.

7 Acknowledgements

This work has been partially supported Career NSF 5409260 (Salimeh Yasaei Sekeh); the findings are those of the authors only and do not represent any position of these funding bodies.

References

[1] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.

[2] Qi-Zhi Cai, Chang Liu, and Dawn Song. Curriculum adversarial training. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 3740–3747, 2018.

[3] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. IEEE, 2017.

[4] Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*, pages 39–57. IEEE Computer Society, 2017. doi: 10.1109/SP.2017.49. URL https://doi.org/10.1109/SP.2017.49.

[5] Kashyap Chitta, José M Álvarez, Elmar Haussmann, and Clément Farabet. Training data subset search with ensemble active learning. *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[6] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 2206–2216. PMLR, 2020. URL http://proceedings.mlr.press/v119/croce20b.html.

[7] Jiequan Cui, Shu Liu, Liwei Wang, and Jiaya Jia. Learnable boundary guided adversarial training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15721–15730, 2021.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT (1)*, 2019.

[9] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 9185–9193. Computer Vision Foundation / IEEE Computer Society, 2018. doi: 10.1109/CVPR.2018.00957.

[10] Madan Ravi Ganesh and Jason J. Corso. Rethinking curriculum learning with incremental labels and adaptive compensation. In *31st British Machine Vision Conference 2020, BMVC 2020, Virtual Event, UK, September 7-10, 2020*. BMVA

Press, 2020. URL https://www.bmvc2020-conference.com/assets/papers/0581.pdf.

[11] Alex Graves, Marc G Bellemare, Jacob Menick, Remi Munos, and Koray Kavukcuoglu. Automated curriculum learning for neural networks. In *international conference on machine learning*, pages 1311–1320. PMLR, 2017.

[12] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep learning with limited numerical precision. In *International conference on machine learning*, pages 1737–1746. PMLR, 2015.

[13] Guy Hacohen and Daphna Weinshall. On the power of curriculum learning in training deep networks. In *International Conference on Machine Learning*, pages 2535–2544. PMLR, 2019.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society, 2016. doi: 10.1109/CVPR.2016.90. URL https://doi.org/10.1109/CVPR.2016.90.

[15] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[16] Gao Huang, Zhuang Liu, Geoff Pleiss, Laurens Van Der Maaten, and Kilian Weinberger. Convolutional networks with dense connectivity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[17] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *International Conference on Machine Learning*, pages 2304–2313. PMLR, 2018.

[18] Hoki Kim. Torchattacks: A pytorch repository for adversarial attacks. *arXiv preprint arXiv:2010.01950*, 2020.

[19] Alex Krizhevsky et al. Learning multiple layers of features from tiny images. 2009.

[20] Àgata Lapedriza, Hamed Pirsiavash, Zoya Bylinskii, and Antonio Torralba. Are all training examples equally valuable? *CoRR*, abs/1311.6510, 2013. URL http://arxiv.org/abs/1311.6510.

[21] Shiyu Liang, Yixuan Li, and R. Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=H1VGkIxRZ.

[22] Ilya Loshchilov and Frank Hutter. Online batch selection for faster training of neural networks. *arXiv preprint arXiv:1511.06343*, 2015.

[23] Shiqing Ma, Yingqi Liu, Guanhong Tao, Wen-Chuan Lee, and Xiangyu Zhang. NIC: detecting adversarial samples with neural network invariant checking. In *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019*. The Internet Society, 2019. URL https://www.ndss-symposium.org/ndss-paper/nic-detecting-adversarial-samples-with-neural-network-invari

[24] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

[25] Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16. IEEE, 2020.

[26] Leslie Rice, Eric Wong, and Zico Kolter. Overfitting in adversarially robust deep learning. In *International Conference on Machine Learning*, pages 8093–8104. PMLR, 2020.

[27] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[28] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 4510–4520. Computer Vision Foundation / IEEE Computer Society, 2018. doi: 10.1109/CVPR.2018.00474.

[29] Ali Shafahi, Mahyar Najibi, Mohammad Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! *Advances in Neural Information Processing Systems*, 32:3358–3369, 2019.

[30] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1409.1556.

[31] Michael R Smith, Tony Martinez, and Christophe Giraud-Carrier. An instance level analysis of data complexity. *Machine learning*, 95(2):225–256, 2014.

[32] Ben Sorscher, Robert Geirhos, Shashank Shekhar, Surya Ganguli, and Ari Morcos. Beyond neural scaling laws: beating power law scaling via data pruning. *Advances in Neural Information Processing Systems*, 35:19523–19536, 2022.

[33] Xiao Sun, Naigang Wang, Chia-Yu Chen, Jiamin Ni, Ankur Agrawal, Xiaodong Cui, Swagath Venkataramani, Kaoutar El Maghraoui, Vijayalakshmi Viji Srinivasan, and Kailash Gopalakrishnan. Ultra-low precision 4-bit training of deep neural networks. *Advances in Neural Information Processing Systems*, 33, 2020.

[34] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[35] Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J Gordon. An empirical study of example forgetting during deep neural network learning. In *International Conference on Learning Representations*, 2018.

[36] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. In *International Conference on Learning Representations*, 2018.

[37] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *Advances in neural information processing systems*, 29:3630–3638, 2016.

[38] Fu Wang, Yanghao Zhang, Yanbin Zheng, and Wenjie Ruan. Gradient-guided dynamic efficient adversarial training. *CoRR*, abs/2103.03076, 2021. URL https://arxiv.org/abs/2103.03076.

[39] Yisen Wang, Xingjun Ma, James Bailey, Jinfeng Yi, Bowen Zhou, and Quanquan Gu. On the convergence and robustness of adversarial training. In *International Conference on Machine Learning*, pages 6586–6595. PMLR, 2019.

[40] Eric Wong, Leslie Rice, and J. Zico Kolter. Fast is better than free: Revisiting adversarial training. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=BJx040EFvH.

[41] Cihang Xie, Zhishuai Zhang, Yuyin Zhou, Song Bai, Jianyu Wang, Zhou Ren, and Alan L. Yuille. Improving transferability of adversarial examples with input diversity. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 2730–2739. Computer Vision Foundation / IEEE, 2019. doi: 10.1109/CVPR.2019.00284.

[42] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P. Xing, Laurent El Ghaoui, and Michael I. Jordan. Theoretically principled trade-off between robustness and accuracy. In *International Conference on Machine Learning*, 2019.

[43] Jingfeng Zhang, Xilie Xu, Bo Han, Gang Niu, Lizhen Cui, Masashi Sugiyama, and Mohan Kankanhalli. Attacks which do not kill training make adversarial learning stronger. In *International Conference on Machine Learning*, pages 11278–11287. PMLR, 2020.

[44] Tianyi Zhou and Jeff Bilmes. Minimax curriculum learning: Machine teaching with desirable difficulties and scheduled diversity. In *International Conference on Learning Representations*, 2018.

[45] Tianyi Zhou, Shengjie Wang, and Jeff A Bilmes. Curriculum learning by dynamic instance hardness. *Advances in Neural Information Processing Systems*, 33, 2020.

[46] Tianyi Zhou, Shengjie Wang, and Jeff Bilmes. Curriculum learning by optimizing learning dynamics. In *International Conference on Artificial Intelligence and Statistics*, pages 433–441. PMLR, 2021.