Received 7 August 2022, accepted 2 October 2022, date of publication 10 October 2022, date of current version 18 October 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3213734

RESEARCH ARTICLE

AccelAT: A Framework for Accelerating the Adversarial Training of Deep Neural Networks Through Accuracy Gradient

FARZAD NIKFAM[1], (Graduate Student Member, IEEE),
ALBERTO MARCHISIO[2], (Graduate Student Member, IEEE),
MAURIZIO MARTINA[1], (Senior Member, IEEE),
AND MUHAMMAD SHAFIQUE[3], (Senior Member, IEEE)
[1]Department of Electrical, Electronics and Telecommunication Engineering, Politecnico di Torino, 10129 Turin, Italy
[2]Institute of Computer Engineering, Technische Universität Wien (TU Wien), 1040 Vienna, Austria
[3]eBrain Laboratory, Division of Engineering, New York University Abu Dhabi, Abu Dhabi, United Arab Emirates

Corresponding author: Farzad Nikfam (farzad.nikfam@polito.it)

This work was supported in part by the Doctoral College Resilient Embedded Systems, which is run jointly by the TU Wien's Faculty of Informatics and the UAS Technikum Wien; in part by the NYUAD Center for Interacting Urban Networks (CITIES) funded by Tamkeen under the NYUAD Research Institute under Award CG001; in part by the Center for Cyber Security (CCS) funded by Tamkeen under the NYUAD Research Institute under Award G1104; and in part by the Center for Artificial Intelligence and Robotics (CAIR) funded by Tamkeen under the NYUAD Research Institute under Award CG010.

ABSTRACT Adversarial training is exploited to develop a robust Deep Neural Network (DNN) model against the malicious altered data. These attacks may have catastrophic effects on DNN models but are indistinguishable for a human being. For example, an external attack can modify an image adding noises invisible for a human eye, but a DNN model misclassifies the image. A key objective for developing robust DNN models is to use a learning algorithm that is fast but can also give model that is robust against different types of adversarial attacks. Especially for adversarial training, enormously long training times are needed for obtaining high accuracy under many different types of adversarial samples generated using different adversarial attack techniques. This paper aims at accelerating the adversarial training to enable fast development of robust DNN models against adversarial attacks. The general method for improving the training performance is the hyperparameters fine-tuning, where the learning rate is one of the most crucial hyperparameters. By modifying its shape (the value over time) and value during the training, we can obtain a model robust to adversarial attacks faster than standard training. First, we conduct experiments on two different datasets (CIFAR10, CIFAR100), exploring various techniques. Then, this analysis is leveraged to develop a novel fast training methodology, *AccelAT*, which automatically adjusts the learning rate for different epochs based on the accuracy gradient. The experiments show comparable results with the related works, and in several experiments, the adversarial training of DNNs using our *AccelAT* framework is conducted up to 2× faster than the existing techniques. Thus, our findings boost the speed of adversarial training in an era in which security and performance are fundamental optimization objectives in DNN-based applications. To facilitate reproducible research this is the AccelAT open-source framework: https://github.com/Nikfam/AccelAT.

INDEX TERMS Deep neural network (DNN), adversarial training, fast training, hyperparameters, learning rate (LR), Foolbox, python, TensorFlow, adversarial attack.

The associate editor coordinating the review of this manuscript and approving it for publication was Mansoor Ahmed.

I. INTRODUCTION

Machine Learning (ML) [1], [2], [3] is an ever-expanding field and has achieved wide proliferation in recent years

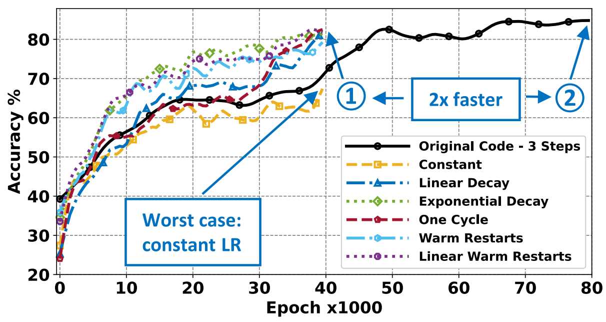


FIGURE 1. FAT Training of ResNet-20 on the *CIFAR10* natural images dataset with different LR techniques.

due to the development of highly-efficient hardware, such as GPUs. Nevertheless, the advanced and complex ML models require gigantic training time. Therefore, its acceleration not only has a direct impact on the usage of large GPU-based datacenters in which typically the training is conducted but will also enable training on low-cost multi-GPU workstations, as well as will ease the development of new research directions, such as continuous learning. On the other hand, in the last decade, it has been discovered that models are highly vulnerable to external attacks, and nowadays, the ML models need to be robust against such attacks to be deployed in safety-critical applications.

A. TARGET RESEARCH PROBLEM AND CHALLENGES

Adversarial training [4] has become a popular method for training Deep Neural Networks (DNNs) [5] with robustness against the adversarial attacks. Unfortunately, robust DNNs are not always easy to be trained, as it takes 3× to 30× longer time [6] to obtain high accuracy when adversarial (noisy) samples are added to the training dataset, compared to the standard (i.e., non adversarial) training. Hence, it is essential to create DNN models that are not only robust but also quite fast to be trained. To obtain the above-discussed properties, we propose to employ fast training techniques for advanced adversarial training of DNNs, and show the feasibility of this design strategy by designing a novel fast training method, called *AccelAT*.

B. MOTIVATIONAL CASE STUDY

Nowadays, advanced methods for adversarial training - such as, Free Adversarial Training (FAT) [6], YOPO [7], and Trades [8] - are currently used to obtain DNN models which are robust against adversarial attacks. In our analyses, we focus on the FAT method, which is already highly optimized compared to the original adversarial training method [4]. To further accelerate the training process, we employ various fast training techniques, focusing mainly on the study of hyperparameters. Since the learning rate (LR) has a strong influence on the convergence of the DNN training process, we analyze and how its variation affects the training speed for accurate and robust DNN models. The fast training techniques analyzed are *linear decay*, *exponential decay*, *one cycle* [9], and *warm restarts* [10], [11]. The results applied to the ResNet-20 DNN for the CIFAR10 dataset (see Fig. 1) confirm that fast training techniques can be used with success also for adversarial training. Compared to the original FAT method, performing the training with a LR that follows the behavior of one of these above-discussed fast training policy can reduce the training time by around 2× (see pointer ① - Fig. 1), while obtaining a similar accuracy level (see pointer ②).

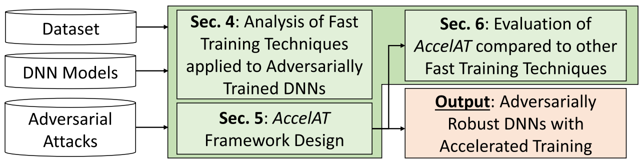

Dataset
DNN Models
Adversarial Attacks
Sec. 4: Analysis of Fast Training Techniques applied to Adversarially Trained DNNs
Sec. 5: AccelAT Framework Design
Sec. 6: Evaluation of AccelAT compared to other Fast Training Techniques
Output: Adversarially Robust DNNs with Accelerated Training

FIGURE 2. Overview of our novel contributions.

C. OUR NOVEL CONTRIBUTIONS

The main contributions of this paper are (see Fig. 2):

- We analyze the prominent fast training techniques applied to adversarially trained DNNs, showing significant training time reduction in terms of training epochs (Sec. IV).
- We design a novel framework, *AccelAT*, which automatically reduces the LR when the accuracy gradient starts decreasing, i.e., when the accuracy curve starts falling into a plateau region (Sec. V).
- The experimental results on multiple DNNs (ResNet, MobileNet) trained on CIFAR10 and CIFAR100 datasets with our *AccelAT* framework obtain up to 8% higher adversarial robustness against the most common attacks such as LinfPGD, Fast Gradient Sign Method (FGSM), and DeepFool (Sec. VI).

Open Source: To facilitate the research and developments in this field, and for reproducible research, this is the AccelAT open-source framework: https://github.com/Nikfam/AccelAT.

Before proceeding to the main technical sections, we present an overview of adversarial attacks and defenses, and fast training policies for DNNs, in Sec. II and Sec. III, respectively, with a level of detail necessary to understand the rest of the paper.

II. OVERVIEW OF ADVERSARIAL ATTACKS AND DEFENSES FOR DNNs

Adversarial training [12] is a branch of ML that deals with creating robust models against adversarial attacks [13], for instance, by augmenting the adversarial samples to the training dataset. For years, the training has only focused on achieving high accuracy. However, there exist malicious attacks that mine the algorithms correct behavior. If a DNN model is attacked, it will incorrectly execute its process, which can lead to severe consequences for safety-critical applications [14]. For example, for cases where facial, voice, or fingerprint recognition is used to unlock certain services,

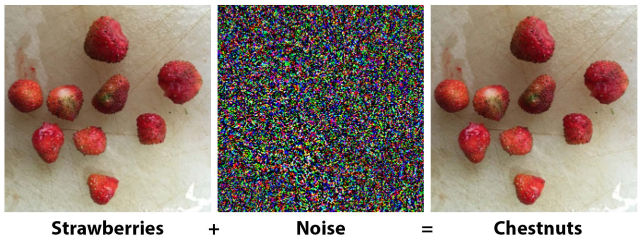


FIGURE 3. An example of adversarial attack, where strawberries are misclassified as chestnuts [15].

an external attack can cause severe damage. Thus, it is desired that DNN models are robust against such external attacks. However, it is known that complex DNN models are vulnerable to malicious attacks, which could make their accuracy drop from near 100% to nearly 0% [15]. To counter these attacks, we need to develop and deploy robust models that can maintain high accuracy in the presence of these malicious variations.

Adversarial training leverages clean images as well as the noisy images (following a certain adversarial attack model) for training the DNN models, thereby enabling the trained DNN models to classify correctly even in the presence of an adversarial attack during the inference. However, this robustness is achieved at the expense of significantly longer training time, proportional to the amount of adversarial samples and attack models.

A. ADVERSARIAL EXAMPLES

The basic foundation of an adversarial attack algorithm is to create imperceptibly-modified examples [15] that mislead the DNN model. On the other hand, if the examples were modified with a random logic, the problem would not arise since the model would fail, but a human being would easily recognize and detect the modifications. However, some examples could be modified to mislead a DNN model through a malicious attack without any evident variation perceived by the human eye. For example, as shown in Fig. 3, the two images (original and modified by attack) are identical to the human eye. However, every single pixel has been modified according to the noise seen in the middle [16].

The consequence is that DNN models can be attacked without obvious external signs. In reality, with certain types of attacks, it is possible to generate modified images indistinguishable from a human being, which the DNN model can still correctly recognize. The latter case is not a critical problem, as they would be images discarded by a human. In this work, we will focus on the misclassification of DNN models.

B. ADVERSARIAL ATTACKS

A DNN model can learn to recognize images, but in an entirely different way from how humans do it. Therefore, various adversarial attacks algorithms have been proposed [17]. For example, some attacks are based on changing a single pixel [18], others on certain image features, but the most common approach consists of calculating the division line, called decision boundary, that distinguishes one class from another.

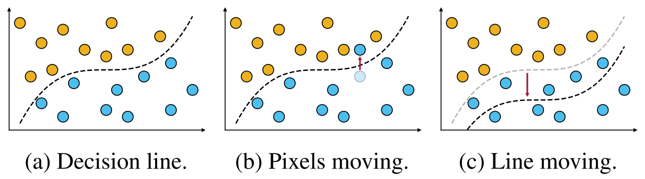


FIGURE 4. Two ways of fooling a classifier.

In a simple case with two classes, the decision boundary looks like as in Fig. 4a. A targeted attack would perturb all the borderline examples. For image classification applications, the attack would vary the last layer features, closest to the line edge, just enough to make them cross the line (Fig. 4b). In this way, the classification is completely distorted without actually changing the image much overall. This process deals with two almost identical images, which are instead classified in different ways. In some cases, the pixels do not change the position. On the contrary, the attack moves the separation line that distinguishes the classes (Fig. 4c), thus leading to a misclassification due to the shifted decision boundary.

C. WHITE-BOX ATTACKS

There are mainly two categories of attacks, namely white-box and black-box [19]. In this work, we focus on white-box attacks, which are the most accessible and powerful type of attacks to perform considering that the adversary has more knowledge about the system [4]. While a black-box attack has access only to the inputs and outputs, a white-box attack also leverages the knowledge of the internal structure of the model to be attacked. Therefore, the attack is more specific and powerful, and the caused damage increases.

D. ADVERSARIAL TRAINING

There exist various techniques to train robust models, such as data augmentation, second model control, and hyperparameters fine-tuning [17], [20].

The most common technique to counter attacks is data augmentation, that is, the DNNs are trained not only on correct images, but also on already attacked images or adversarial samples generated based on given attack models [21]. This procedure significantly increases the DNN accuracy against attacks [22], [23]. However, as a drawback, the accuracy of clean images decreases, and often falls below the required accuracy level [24].

The second model control uses two DNN models. This technique examines the main DNN and its internal characteristics to predict whether the analyzed example is adversarial or not [25], [26]. In practice, this technique uses an "external guard" logic that controls the whole process to verify its effective operation. However, the study of its effectiveness and complexity is still immature.

Moreover, the performance of the adversarial training can be improved by changing the values of the hyperparameters, such as weight decay, batch size, or LR [27], [28].

E. ADVERSARIAL LIBRARIES

There are various libraries to implement the adversarial attacks [29], [30]. Foolbox [31] is used in this work due to its good documentation, functionality, and support for TensorFlow [32] and Pytorch [33] packages.

III. OVERVIEW OF FAST TRAINING POLICIES FOR DNNs

We need fast training to meet an ever-increasing demand for large databases to be managed in real-time. For example, popular websites like Google, YouTube, and Facebook need to manage constant incoming data streams, training the models as quickly as possible. On the other hand, treating a large amount of data consumes a large amount of power, and is often out of the hand of small-scale organizations where training resources are limited. Therefore, accelerating this process allows obtaining advantages in terms of resources, time and energy. With current processors, such as CPUs and GPUs, the computation times for the complete training can last from days to several weeks for large-sized datasets, and a few hours to days for medium-sized datasets. Hence, a significant reduction in the training time is highly desirable. There are various ways to speed up:

- Specialized techniques for certain types of DNN models, like Adam [34], Ada-Boundary [35], or Super-Convergence [36].
- Generic optimizations, like hyperparameters tuning [9], which apply to nearly every DNN model.

A. FAST TRAINING TECHNIQUES

Generic techniques mainly include the changes to hyperparameters [9] and, more specifically, to the LR [37], since variable LR values can give better results than constant values. Among the various state-of-the-art fast training methodologies proposed in the literature, the most advanced in this regard are the following:

- One cycle policy [9];
- Cyclical policy [37];
- Warm restarts [10], [11].

Before applying each of these techniques, it is necessary to find the best LR to use during the training, through the *LR finder* technique.

B. LEARNING RATE FINDER

The simplest method to find the correct LR value is to change it exponentially, from small to large values, during reasonably long training. An efficient choice is to vary the LR by at least ten orders of magnitude throughout the training. Theoretically, if all the various parameters have been normalized, the LR will often lay between 0.001 and 10. For this reason, it is preferable to fully cover this range during the training. Once the test training is finished, it will be enough to look at the

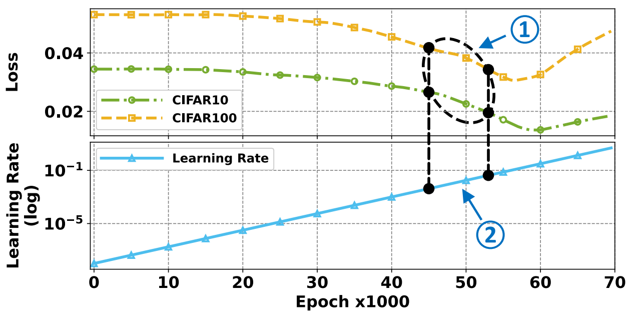


FIGURE 5. LR finder executed on the ResNet-50 model for the CIFAR10 and CIFAR100 datasets.

accuracy and error graph to find the maximum recommended LR [37]. As shown in Fig. 5, for small LR values, the accuracy and the error practically do not vary. However, in a certain range of LR values, the error decreases up to a minimum, after which it diverges for large LR values. Hence, the desirable LR values are those in which the error decreases from the initial plateau to the minimum point, beyond which the divergence begins. Therefore, using a maximum LR value of about one order of magnitude lower than the minimum error point is advisable to be distant enough to avoid the divergence region (see pointer ① - Fig. 5). In summary, the LR should range from the value in which the error slope starts decreasing, until one order of magnitude less than the point in which the error curve exhibits the minimum (see pointer ②).

C. ONE CYCLE POLICY

The one cycle policy [9] is based on varying the LR and other hyperparameters during the training process to obtain fast training. As the name implies, the basic idea is to apply a single cycle to these hyperparameters throughout the training. Since the one cycle policy is a regularization technique, other types of normalization affecting hyperparameters must be reduced to avoid interference.

After finding the maximum LR through the LR finder (Fig. 5), an initial value equal to 1/10 of the maximum is set (see pointer ① - Fig. 6). Then, the LR assumes the shape of a triangular cycle for about 90% of the total training, i.e., 90% of the total epochs (see Fig. 6), first increasing from the initial value up to maximum (see pointer ②), and then decreasing again to 1/10 of the maximum (see pointer ③). In the last few epochs, equal to about 10% of the total epochs, the LR rapidly decreases to 1/1000 of the maximum LR (see pointer ④). Properly setting the duration of the last part of the training is extremely important, since a longer duration would lead to overfitting, while a shorter duration would lead to low accuracy.

The one cycle policy is also applied to the momentum with an opposite shape (Fig. 6). In this way, the regularization carried out on the LR is not dampened by the momentum, but on the contrary, it is strengthened. There is a maximum recommended momentum value of 0.95 (see pointer ⑤), while the minimum should be 0.85 (see pointer ⑥). In the

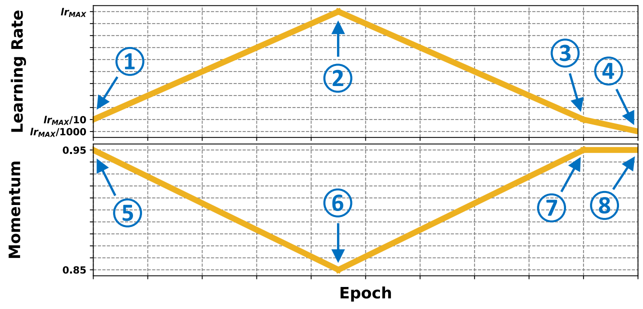


FIGURE 6. One cycle policy.

final part of the training, while the LR decreases rapidly, the momentum remains fixed at the maximum value of 0.95 (see pointers ⑦ and ⑧).

D. CYCLICAL POLICY

The cyclical policy [37], shown in Figs. 7 and 8, is similar to the one cycle policy, with the difference that the cycle is repeated several times, constantly oscillating between the same maximum (see pointer ① - Fig. 7) and minimum (see pointer ②) values. This policy can be helpful if the training process of the DNN model exhibit many local minimum points, since using a cyclical LR allows the training to seek deeper minimums and achieve higher accuracy.

The length of every single cycle is calculated as a multiple of an epoch. It is recommended to use cycle length values between 4 and 20 times an epoch to obtain optimal results. However, it is advisable to perform training with at least 3 - 5 cycles to obtain an evident improvement over a constant LR. Increasing the number of cycles too much would eliminate the cycle's usefulness, because the training would not have time to adapt to the variation of the LR.

The maximum and minimum values of the LR to adopt in the cycle must be chosen carefully (Fig. 9), since the success of the training depends on them. In both cases, it is necessary to use the graph produced by the LR finder (Fig. 5), which must be run before the final training. The maximum LR is found precisely as for the one cycle policy, i.e., 1/10 of the minimum point of the loss that corresponds to the limit (see pointer ① - Fig. 9). On the other hand, the minimum LR is set to a value in the loss descent zone from the initial plateau onwards (see pointer ② - Fig. 9).

In some other cases, the cycles are repeated with the same length, but the maximum LR value decreases (see pointer ① - Fig. 8) to search deeper in the local minima, such as the decreasing triangular cycles of Fig. 8.

E. WARM RESTARTS

The warm restart methods [10], [11] are also based on a cyclical policy, but as the term says, there are sudden restarts from the minimum (see pointer ① - Fig. 10, pointer ① - Fig. 11) to the maximum LR value (see pointer ② - Fig. 10, pointer ② - Fig. 11). This phenomenon leads to instantaneously restart a new long descent, aiming

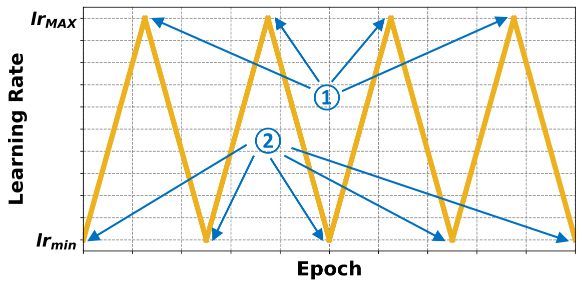


FIGURE 7. A triangular cyclical policy [37].

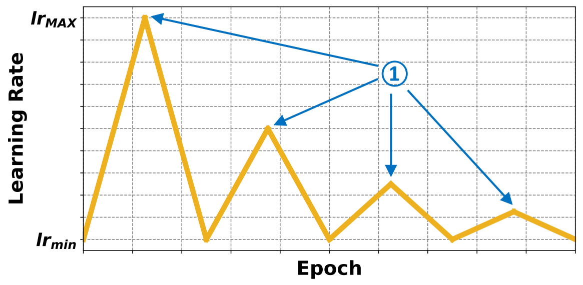


FIGURE 8. Cyclical policy with fixed lower boundary.

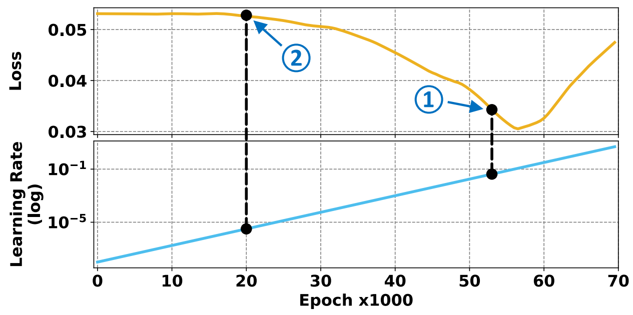


FIGURE 9. LR boundary on the loss plot for the cyclical policy [37].

at finding deeper minima. The warm restarts are always performed on the LR and can assume various shapes, such as:

- Sinusoidal (Fig. 10);
- Linear (Fig. 11);
- Trapezoidal.

The warm restarts can have multipliers that make the progress accordion-like during the training (Fig. 10), or the restarts can be at different gradually decreasing values (Fig. 11).

F. OTHER FAST TRAINING METHODS

Changing the shapes of the hyperparameters or mixing the above-discussed techniques, it possible to obtain new training policies for the LR, which might be more effective than the originals methods.

IV. ANALYSIS: FAST TRAINING TECHNIQUES APPLIED TO ADVERSARIAL TRAINING METHODS

Various methodologies have recently been proposed to accelerate the adversatial training [7], [8]. The Free Adversarial

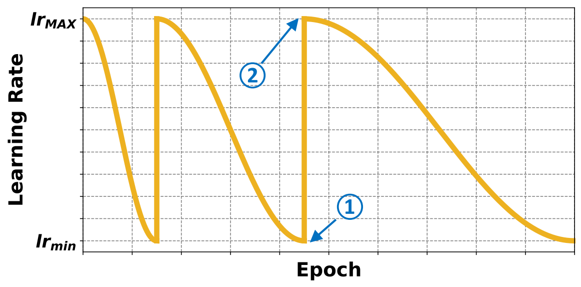


FIGURE 10. Accordion-like sinusoidal warm restarts [10].

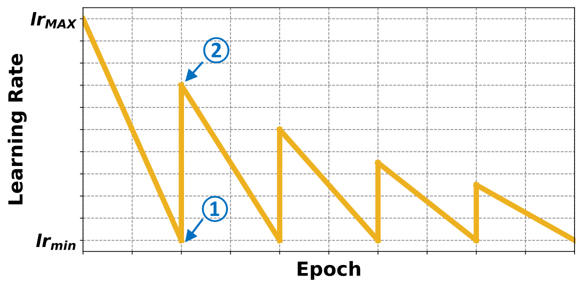


FIGURE 11. Linear decreasing warm restarts [11].

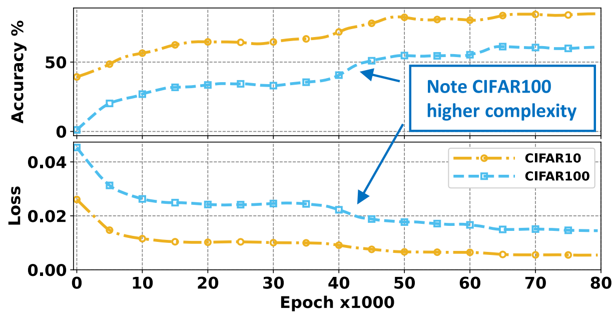


FIGURE 12. Original FAT [6] accuracy and loss on natural images.

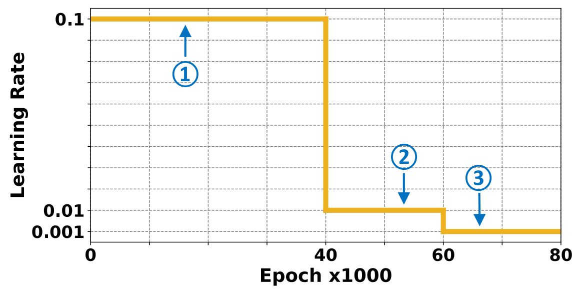


FIGURE 13. FAT's 3-steps LR [6].

Training (FAT) [6] is chosen as the baseline method for the experiments in this section.

The m parameter, also called *Free-m*, is a key parameter of the FAT algorithm, since it allows to repeat the perturbation several times for every single minibatch [38]. With this terminology, a traditional training is obtained by keeping $m = 1$. The parameter ϵ indicates the adversarial perturbation. A too large ϵ would make the perturbations so high that the images would be recognized as crafted even by the human eye.

Building on top of this, there have been concurrent works such as Fast is Better than Free [39] and subsequently, also GradAlign [40], which have demonstrated the reliability in using the FGSM for speeding up with the proper precautions. In this work, we mainly focus, instead of on the type of training, on the hyperparameters. Therefore, we tested the feasibility of *AccelAT* also with the FGSM method.

A. ORIGINAL FAT RESULTS

A first analysis has been conducted by reproducing the original FAT method, applied to the ResNet-50 model [41] on CIFAR10 [42], [43] and CIFAR100 [43], [44] datasets, under the projected gradient descent (PGD) attack [45] with constant ϵ equal to 8.0. Fig. 12 shows the training results in terms of accuracy and loss, obtained for both the CIFAR10 and the CIFAR100 datasets on natural images. As expected, the CIFAR100 accuracy is lower than the CIFAR10 accuracy due to the higher complexity. Final accuracy results:

- CIFAR10 → accuracy: 84.34% - loss: 0.00562;
- CIFAR100 → accuracy: 59.89% - loss: 0.01459.

B. HYPERPARAMETERS SETUP

The original FAT method applies a LR that has a 3-step function (Fig. 13) with the following behavior:

- Epochs = [0 : 40000] → LR = 0.1 (see pointer ① - Fig. 13);
- Epochs = [40000 : 60000] → LR = 0.01 (see pointer ②);
- Epochs = [60000 : 80000] → LR = 0.001 (see pointer ③).

After using the LR finder (Fig. 5), the maximum LR value results to be:

- CIFAR10 → maximum LR = 0.15;
- CIFAR100 → maximum LR = 0.12.

Therefore, for the experiments applying the fast training techniques on the FAT method, a maximum LR higher than that of the original FAT is used.

The momentum values are set to:

- One cycle → momentum = 0.85 – 0.95 (Fig. 6);
- Constant → momentum = 0.90.

The one cycle momentum is used only for the one cycle policy [9]. Instead, for the other techniques, the momentum is fixed to the original FAT constant value.

Based on the regularization criteria, the value of the weight decay has been set to 0.0002. A batch size of 128 has been set due to the computational limits of the calculator. The remaining FAT parameters relative to adversarial training have not been changed, since the aim is not to obtain a more robust model, but to accelerate the training, while achieving the same robustness.

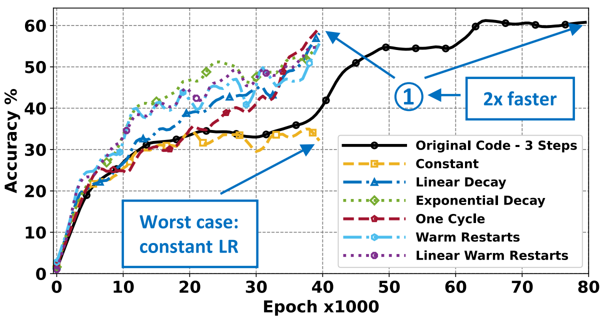


FIGURE 14. FAT Training of ResNet-20 on the *CIFAR100* natural images dataset with different LR techniques.

C. SUPER FAT RESULTS

The results of different training policies applied to the FAT, compared to the original FAT, for the CIFAR10 and CIFAR100 natural images datasets, are shown in Fig. 1 and Fig. 14, respectively. The experiments are performed with various values of training epochs to show the differences between the algorithms. With our settings (i.e., execution on the Tesla K40c GPU), 10000 epochs are executed in about 5 hours. Therefore, the original FAT training lasts for about 40 hours. For this reason, halving the number of epochs allows to execute the experiments faster.

From these results, it can be noticed that the FAT algorithm with a 3-steps LR is already optimized w.r.t. the original adversarial training [4], but with more advanced techniques, the same results can be achieved even in about half the time (see pointer ① - Fig. 14). Thus, the fine-tuning of the hyper-parameters is essential and can lead to super-convergence in standard training and adversarial training without affecting the result and robustness of the DNN model itself.

V. OUR *AccelAT* FRAMEWORK

The idea behind *AccelAT* is to avoid or reduce the model setup time. With the existing fast training techniques, we can obtain good results, but often it takes too much time to set up all parameters and several attempts before finding the best compromise. These dead times are not considered in the training time, but they are still a considerable part of a programmer's work. With *AccelAT*, it is possible to have more freedom of choice since it will be the model itself during the training that will understand when to intervene on the LR for more efficient learning. As we will see, this leads to similar or better results to existing fast training techniques based on the LR.

The functionality of our *AccelAT* methodology is described in Algorithm 1. The LR value, initially equal to the maximum possible value obtained with the LR finder, decreases as plateau zones are found in the learning curve. The LR decreases if the accuracy does not exceed a specific Δ_{acc} in a certain number of epochs.

Inspired by existing fast training techniques that eliminate plateau areas while learning, in our *AccelAT* framework the LR is varied based on the performance of the validation

Algorithm 1 *AccelAT*

Require: Maximum learning rate lr_{MAX}, minimum learning rate lr_{min}, accuracy delta Δ_{acc}, percentage reduction p, number of cycles of interest n, accuracy acc, previous average accuracy acc_{pre}

```
1:  $lr \leftarrow lr_{MAX}$
2:  for $e$ in $epochs$ do
3:      if $(acc(e, e-n) - acc_{pre}) < \Delta_{acc}$ then
4:          $lr \leftarrow lr \cdot p$
5:      end if
6:      if $lr < lr_{min}$ then
7:          $lr \leftarrow lr_{min}$
8:      end if
9:      $acc_{pre} \leftarrow \overline{acc(e, e-n)}$
10: end for
```

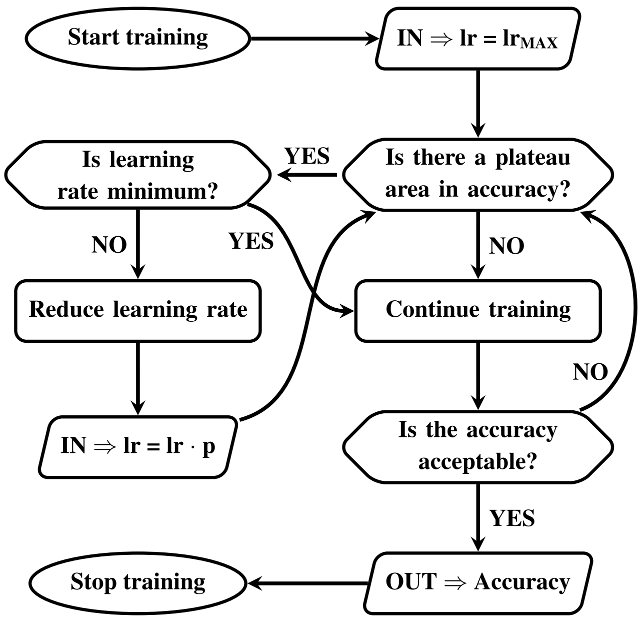


FIGURE 15. *AccelAT* workflow.

accuracy. First, we search for the maximum LR using the LR finder technique, after which we set it as the initial LR (line 1 - Algorithm 1), to be decreased if the accuracy starts to show a plateau. Then, a simplified gradient can be used to change the LR based on accuracy progression. As indicated in lines 3-4, the LR is decreased by a percentage value p if the accuracy in the last n cycles has not increased by a certain value Δ_{acc}, up to the minimum desired LR (see Fig. 15).

Step by step, the framework works as follows:

- With previous training, we find the maximum LR that can be used through the LR finder;
- The LR is set to the previously found maximum LR value;
- The training starts with the predetermined LR;
- The accuracy is calculated for the dataset under analysis;

- If the accuracy does not grow at a sufficient rate that is decided a priori, then the LR decreases by a fixed percentage;
- If, on the other hand, there are no plateau areas, then the training continues with the current LR;
- Once the LR reaches a predetermined minimum value, then the training continues with this fixed LR value;
- When the accuracy reaches an optimal value, or it is impossible to rise further, then the training stops.

For example, let us set a value (n) of ten epochs to evaluate the accuracy, a delta (Δ_{acc}) equal to 1% and let us assume to have found a value of 0.01 as the maximum LR, which we want to reduce by about 10% (p) each time there is a plateau area. After that, the training is launched for 100 epochs. The accuracy increase rate is monitored for each epoch. For instance, in the first 40 epochs, the increase in accuracy is greater than or equal to 1%, compared to the last ten epochs. Hence, the LR remains fixed at 0.01. At the 41st epoch, the accuracy has not increased by at least 1% in the last ten epochs, and consequently, the LR is multiplied by 0.9, i.e., it is reduced by 10%, and we obtain an LR value equal to 0.009. Afterward, the training is resumed and the accuracy starts to rise again. Towards the 70th epoch, it again has a new plateau zone; consequently, the LR is reduced by another 10%. Then, the training continues until the end of the 100 epochs. Once finished, the LR is lower than the maximum, allowing us to increase the accuracy by going deeper into the found local minimum. If we had kept a fixed LR after the first 40 epochs, the accuracy would not have increased, and the last 60 epochs would have been useless. With our *AccelAT* framework, we have obtained an adaptive LR based on the specific training we are performing.

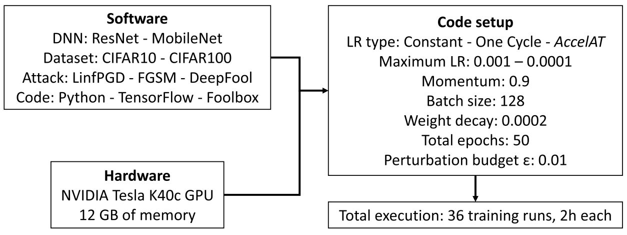
This type of LR policy does not have a fixed shape for every training and, therefore, cannot be plotted. Instead, its shape varies at run-time according to the model, dataset, and training parameters. In any case, it will assume a ladder shape. Compared to other fast training techniques, the *AccelAT* method, not having a fixed shape and calculating the gradient at run-time, is likely to slightly slow down the training. However, on long training with complex datasets, the *AccelAT* allows obtaining an LR suited to the situation.

As will be demonstrated in the next section, our *AccelAT* is more efficient than existing fast training techniques in certain types of training. Therefore, the best approach is to conduct preliminary analyses to determine which are good values of the parameters p, n, and Δ_{acc} to set for the training.

VI. EVALUATING OUR *AccelAT* FRAMEWORK

A. EXPERIMENTAL SETUP

As shown in Fig. 16, our experiments has been conducted using two popular DNNs (ResNet [41] and MobileNet [46]), pre-trained with ImageNet, then trained for the CIFAR10 [42], [43] and CIFAR100 [43], [44] datasets, analyzed through the LinfPGD, FGSM, and DeepFool attacks described through the Foolbox library [31]. The code is written in Python [3], [47] using the TensorFlow 2.X library [32], running on an NVIDIA Tesla K40c GPU with 12 GB of memory.


Software
DNN: ResNet - MobileNet
Dataset: CIFAR10 - CIFAR100
Attack: LinfPGD - FGSM - DeepFool
Code: Python - TensorFlow - Foolbox
Hardware
NVIDIA Tesla K40c GPU
12 GB of memory
Code setup
LR type: Constant - One Cycle - AccelAT
Maximum LR: 0.001 – 0.0001
Momentum: 0.9
Batch size: 128
Weight decay: 0.0002
Total epochs: 50
Perturbation budget ε: 0.01
Total execution: 36 training runs, 2h each
FIGURE 16. **Experimental setup.**

Each network/dataset/attack combination is tested with three types of learning policies, which are constant (the most simple one), one cycle (the best during FAT review), and *AccelAT*. Therefore, we perform 36 total training runs, taking around 2 hours per run. We found maximum LR values between 0.001 and 0.0001 for all simulations using the LR finder. The other hyperparameters are kept at a fixed optimal value for our setup for all simulations:

- Momentum $\rightarrow$ 0.9;
- Batch size $\rightarrow$ 128;
- Weight decay $\rightarrow$ 0.0002.

To avoid overfitting, we have noticed that the best choice is to use early stopping for the training. As also highlighted in these papers [48], [49], early stopping can lead to more optimal solutions than the introduction of new normalizations. For this reason, we set our total epochs to 50 to avoid overfitting. This choice does not conflict with our algorithm as acceptable accuracy values are obtained even with fewer epochs. For the attack we used a perturbation budget ε equal to 0.01 and the algorithm automatically iterates until it reaches an acceptable level to fool the DNN [50].

B. EXPERIMENTAL RESULTS

From the 36 training runs, we extract nine results that are reported in Fig. 17, Fig. 18, and Fig. 19. We have shown only the graphs obtained on the adversarial training-set for greater clarity. We then calculated the accuracy with the adversarial test-set, the natural training-set, and the natural test-set. In all cases, the results are comparable to these graphs, with linearly higher or lower values for each type of LR. Note, our *AccelAT* technique is sometimes more efficient than the one cycle policy, especially with MobileNet DNNs. The best results are obtained on more complex datasets, where many labels reduce the risk of overfitting and where a variable LR is more valuable than a constant one. For example, in Fig. 18, and Fig. 19, the training curve using the *AccelAT* methodology reaches a high accuracy value in fewer cycles compared to the other techniques (see pointer ① - Fig. 18, pointer ① - Fig. 19). Also notice that, in Fig. 19, the accuracy of the curve employing the *AccelAT* technique continues to increase in every cycle, also towards the end of the training (see pointer ② - Fig. 19), where instead the accuracy obtained with the one cycle policy stops growing.

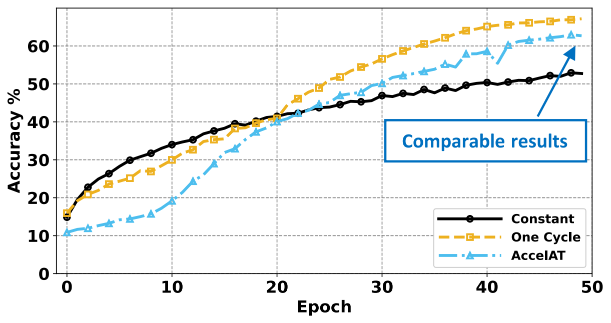


FIGURE 17. ResNET model trained on the CIFAR10 dataset, attacked with DeepFool - Adversarial training-set.

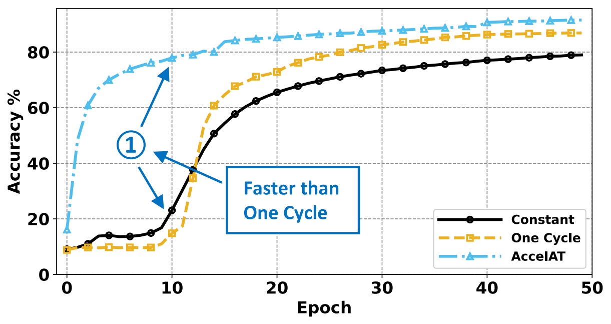


FIGURE 18. MobileNET model trained on the CIFAR10 dataset, attacked with LinfPGD - Adversarial training-set.

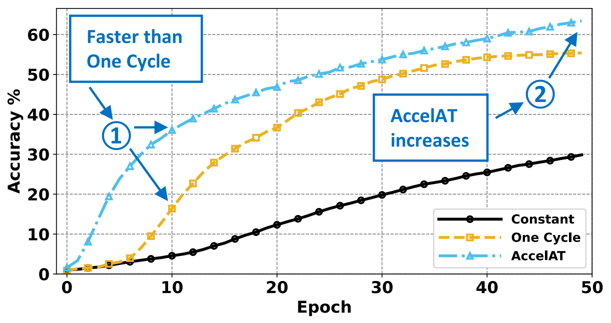


FIGURE 19. MobileNET model trained on the CIFAR100 dataset, attacked with FGSM - Adversarial training-set.

VII. CONCLUSION

In an era in which robustness is fundamental for DNNs, performing training in a fast yet robust manner is challenging. This research demonstrates that advanced fast training techniques can also be applied to adversarial training, obtaining significant improvements with similar robustness. Moreover, we propose *AccelAT*, a framework for adjusting the LR during training based on the accuracy gradient. Our experimental results show that the *AccelAT* not only outperforms the training with constant LR, which usually reaches a sub-optimal local minimum, but is comparable or better than other fast training techniques. Moreover, it is efficient for complex training with large datasets, where a LR varied at run-time is essential to better fit the DNN model and allows more effective learning. Therefore, a new generation of DNN models, which are robust and fast, is starting up, and we demonstrated their feasibility. We believe that, in the near future, these models will be widely employed.

In future works, we plan to modify more hyperparameters by combining them to test even faster methods. Since the batch size is generally fixed and depends on the computing power of the GPU, it affects less the performance. On the other hand, the momentum can act simultaneously with the LR for better normalization. Hyperparameters are often neglected, but their correct use can lead to significant advantages.

REFERENCES

[1] O. Simeone, "A very brief introduction to machine learning with applications to communication systems," *IEEE Trans. Cogn. Commun. Netw.*, vol. 4, no. 4, pp. 648–664, Dec. 2018.

[2] W. Samek, T. Wiegand, and K.-R. Müller, "Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models," 2017, *arXiv:1708.08296*.

[3] S. Raschka and V. Mirjalili, *Python Machine Learning*. Birmingham, U.K.: Packt, 2017.

[4] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *Proc. 6th Int. Conf. Learn. Represent. (ICLR)*, Vancouver, BC, Canada, Apr. 2018, pp. 1–23.

[5] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org

[6] A. Shafahi, M. Najibi, A. Ghiasi, Z. Xu, P. J. Dickerson, C. Studer, S. L. Davis, G. Taylor, and T. Goldstein, "Adversarial training for free!" in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, Eds. Vancouver, BC, Canada, Dec. 2019, pp. 3353–3364.

[7] D. Zhang, T. Zhang, Y. Lu, Z. Zhu, and B. Dong, "You only propagate once: Accelerating adversarial training via maximal principle," in *Proc. Adv. Neural Inf. Process. Syst. 32th Annu. Conf. Neural Inf. Process. Syst.*, H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, Eds. Vancouver, BC, Canada, Dec. 2019, pp. 227–238.

[8] H. Zhang, Y. Yu, J. Jiao, P. Eric Xing, L. E. Ghaoui, and I. M. Jordan, "Theoretically principled trade-off between robustness and accuracy," in *Proc. 36th Int. Conf. Mach. Learn.*, vol. 97, K. Chaudhuri and R. Salakhutdinov, Eds. Long Beach, CA, USA, Jun. 2019, pp. 7472–7482.

[9] N. L. Smith, "A disciplined approach to neural network hyper-parameters: Part 1—Learning rate, batch size, momentum, and weight decay," 2018, *arXiv:1803.09820*.

[10] I. Loshchilov and F. Hutter, "SGDR: Stochastic gradient descent with warm restarts," in *Proc. 5th Int. Conf. Learn. Represent. (ICLR)*, Toulon, France, Apr. 2017, pp. 1–16.

[11] P. Mishra and K. Sarawadekar, "Polynomial learning rate policy with warm restart for deep neural network," in *Proc. IEEE Region Conf. (TENCON)*, Kochi, India, Oct. 2019, pp. 2087–2092.

[12] Y. Guo, C. Zhang, C. Zhang, and Y. Chen, "Sparse DNNs with improved adversarial robustness," 2018, *arXiv:1810.09619*.

[13] W. Ruan, X. Yi, and X. Huang, "Adversarial robustness of deep learning: Theory, algorithms, and applications," in *Proc. 30th ACM Int. Conf. Inf. Knowl. Manage.*, Virtual Event, QLD, Australia, Oct. 2021, pp. 4866–4869.

[14] A. Kurakin, J. Ian Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *Proc. 5th Int. Conf. Learn. Represent. (ICLR)*, Toulon, France, Apr. 2017, pp. 1–14.

[15] J. I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, Y. Bengio and Y. LeCun, Eds. San Diego, CA, USA, May 2015, pp. 1–11.

[16] N. Drenkow, N. Sani, I. Shpitser, and M. Unberath, "Robustness in deep learning for computer vision: Mind the gap?" 2021, *arXiv:2112.00639*.

[17] N. Pitropakis, E. Panaousis, T. Giannetsos, E. Anastasiadis, and G. Loukas, "A taxonomy and survey of attacks against machine learning," *Comput. Sci. Rev.*, vol. 34, Nov. 2019, Art. no. 100199.

[18] J. Su, D. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Trans. Evol. Comput.*, vol. 23, no. 5, pp. 828–841, Oct. 2019.
[19] N. Papernot, D. P. McDaniel, J. I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, R. Karri, O. Sinanoglu, A.-R. Sadeghi, and X. Yi, Eds. Abu Dhabi, United Arab Emirates: ACM, Apr. 2017, pp. 506–519.
[20] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 9, pp. 2805–2824, Sep. 2019.
[21] L. Schmidt, S. Santurkar, D. Tsipras, K. Talwar, and A. Madry, "Adversarially robust generalization requires more data," in *Proc. Adv. Neural Inf. Process. Syst. 31st Annu. Conf. Neural Inf. Process. Syst.*, S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Montréal, QC, Canada, Dec. 2018, pp. 5019–5031.
[22] Y. Esfandiari, A. Balu, K. Ebrahimi, U. Vaidya, N. Elia, and S. Sarkar, "A fast saddle-point dynamical system approach to robust deep learning," *Neural Netw.*, vol. 139, pp. 33–44, Jul. 2021.
[23] A. Shafahi, P. Saadatpanah, C. Zhu, A. Ghiasi, C. Studer, W. D. Jacobs, and T. Goldstein, "Adversarially robust transfer learning," in *Proc. 8th Int. Conf. Learn. Represent. (ICLR)*, Addis Ababa, Ethiopia, Apr. 2020, pp. 1–14.
[24] S. Gui, H. Wang, H. Yang, C. Yu, Z. Wang, and J. Liu, "Model compression with adversarial robustness: A unified optimization framework," in *Proc. Adv. Neural Inf. Process. Syst. 32th Annu. Conf. Neural Inf. Process. Syst.*, H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, Eds. Vancouver, BC, Canada, Dec. 2019, pp. 1283–1294.
[25] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff, "On detecting adversarial perturbations," in *5th Int. Conf. Learn. Represent. (ICLR)*, Toulon, France, Apr. 2017, pp. 1–12.
[26] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Honolulu, HI, USA, Jul. 2017, pp. 86–94.
[27] T. Pang, X. Yang, Y. Dong, H. Su, and J. Zhu, "Bag of tricks for adversarial training," in *Proc. 9th Int. Conf. Learn. Represent. (ICLR)*, Virtual Event, QLD, Austria, May 2021, pp. 1–11.
[28] S. Gowal, C. Qin, J. Uesato, A. T. Mann, and P. Kohli, "Uncovering the limits of adversarial training against norm-bounded adversarial examples," 2020, *arXiv:2010.03593*.
[29] J. I. Goodfellow, N. Papernot, and D. P. McDaniel, "Cleverhans V0.1: An adversarial machine learning library," 2016, *arXiv:1610.00768*.
[30] D. Goodman, X. Hao, Y. Wang, Y. Wu, J. Xiong, and H. Zhang, "Advbox: A toolbox to generate adversarial examples that fool neural networks," 2020, *arXiv:2001.05574*.
[31] J. Rauber, W. Brendel, and M. Bethge, "Foolbox V0.8.0: A Python toolbox to benchmark the robustness of machine learning models," 2017, *arXiv:1707.04131*.
[32] M. Abadi et al., "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," 2016, *arXiv:1603.04467*.
[33] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, Eds. Vancouver, BC, Canada, Dec. 2019, pp. 8024–8035.
[34] P. D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, Y. Bengio and Y. LeCun, Eds. san Diego, CA, USA, May 2015, 2015, pp. 1–15.
[35] H. Song, S. Kim, M. Kim, and J.-G. Lee, "Ada-boundary: Accelerating DNN training via adaptive boundary batch selection," *Mach. Learn.*, vol. 109, nos. 9–10, pp. 1837–1853, Sep. 2020.
[36] N. L. Smith and N. Topin, "Super-convergence: Very fast training of residual networks using large learning rates," 2017, *arXiv:1708.07120*.
[37] L. N. Smith, "Cyclical learning rates for training neural networks," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Santa Rosa, CA, USA, Mar. 2017, pp. 464–472.
[38] A. Devarakonda, M. Naumov, and M. Garland, "Adabatch: Adaptive batch sizes for training deep neural networks," 2017, *arXiv:1712.02029*.
[39] E. Wong, L. Rice, and J. Z. Kolter, "Fast is better than free: Revisiting adversarial training," in *Proc. 8th Int. Conf. Learn. Represent. (ICLR)*, Addis Ababa, Ethiopia, Apr. 2020, pp. 1–17.
[40] M. Andriushchenko and N. Flammarion, "Understanding and improving fast adversarial training," in *Proc. Adv. Neural Inf. Process. Syst. 33th Annu. Conf. Neural Inf. Process. Syst.*, H. Larochelle, M. A. Ranzato, R. Hadsell, M.-F. Balcan, and H.-T. Lin, Eds. Dec. 2020, pp. 1–26.
[41] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Las Vegas, NV, USA, May 2016, pp. 770–778.
[42] A. Krizhevsky, V. Nair, and G. Hinton, "Cifar-10 (Canadian institute for advanced research)," Tech. Rep., 2015.
[43] A. Krizhevsky, "Learning multiple layers of features from tiny images," Dept. Comput. Sci., Univ. Toronto, 2009.
[44] A. Krizhevsky, V. Nair, and G. Hinton, "Cifar-100 (Canadian institute for advanced research)," Tech. Rep., 2015.
[45] J. Liu and J. Ye, "Efficient Euclidean projections in linear time," in *Proc. 26th Annu. Int. Conf. Mach. Learn.*, vol. 382, A. P. Danyluk, L. Bottou, and M. L. Littman, Eds. Montreal, QC, Canada, Jun. 2009, pp. 657–664.
[46] G. Andrew Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.
[47] S. Raschka, J. Patterson, and C. Nolet, "Machine learning in Python: Main developments and technology trends in data science, machine learning, and artificial intelligence," *Information*, vol. 11, no. 4, p. 193, 2020.
[48] L. Rice, E. Wong, and J. Zico Kolter, "Overfitting in adversarially robust deep learning," in *Proc. 37th Int. Conf. Mach. Learn. (ICML)*, vol. 119, Jul. 2020, pp. 8093–8104.
[49] J. Zhang, X. Xu, B. Han, G. Niu, L. Cui, M. Sugiyama, and S. M. Kankanhalli, "Attacks which do not kill training make adversarial learning stronger," in *Proc. 37th Int. Conf. Mach. Learn.*, vol. 119, Jul. 2020, pp. 11278–11287.
[50] N. Carlini, A. Athalye, N. Papernot, W. Brendel, J. Rauber, D. Tsipras, J. Ian Goodfellow, A. Madry, and A. Kurakin, "On evaluating adversarial robustness," 2019, *arXiv:1902.06705*.

FARZAD NIKFAM (Graduate Student Member, IEEE) received the bachelor's degree in mechanical engineering and the master's degree in mechatronic engineering from the Politecnico di Torino, Turin, Italy, in March 2018 and March 2020, respectively, where he is currently pursuing the Ph.D. degree in electronics and communication engineering under the supervision of Prof. Maurizio Martina. His research interest includes software machine learning with emphasis on security and privacy problems.

ALBERTO MARCHISIO (Graduate Student Member, IEEE) received the B.Sc. and M.Sc. degrees in electronic engineering from the Politecnico di Torino, Turin, Italy, in October 2015 and April 2018, respectively. He is currently pursuing the Ph.D. degree with the Computer Architecture and Robust Energy-Efficient Technologies (CARE-Tech.) Laboratory, Institute of Computer Engineering, Technische Universität Wien (TU Wien), Vienna, Austria, under the supervision of Prof. Dr. Muhammad Shafique. His research interests include hardware and software optimizations for machine learning, brain-inspired computing, VLSI architecture design, emerging computing technologies, robust design, and approximate computing for energy efficiency. He has coauthored more than 20 papers in prestigious international conferences and journals. He received the honorable mention at the Italian National Finals of Maths Olympic Games, in 2012, and the Richard Newton Young Fellow Award, in 2019.

MAURIZIO MARTINA (Senior Member, IEEE) received the M.S. and Ph.D. degrees in electrical engineering from the Politecnico di Torino, Italy, in 2000 and 2004, respectively. He is currently a Full Professor with the VLSI-Laboratory Group, Politecnico di Torino. His research interests include computer architecture and VLSI design of architectures for digital signal processing, video coding, communications, networking, artificial intelligence, machine learning, and event-based processing. He has edited one book and published three book chapters on VLSI architectures and digital circuits for video coding, wireless communications, and error correcting codes. He has more than 100 scientific publications and is the coauthor of two patents. He is currently an Associate Editor of IEEE Transactions on Circuits and Systems—I: Regular Papers. He had been a part of the organizing and technical committee of several international conferences, including BioCAS 2017, ICECS 2019, and AICAS 2020. He is currently the Counselor of the IEEE Student Branch at the Politecnico di Torino and a Professional Member of IEEE HKN.

MUHAMMAD SHAFIQUE (Senior Member, IEEE) received the Ph.D. degree in computer science from the Karlsruhe Institute of Technology (KIT), Germany, in 2011. Afterwards, he established and led a highly recognized research group at KIT for several years and conducted impactful collaborative research and development activities across the globe. In October 2016, he joined as a Full Professor in computer architecture and robust, energy-efficient technologies with the Faculty of Informatics, Institute of Computer Engineering, Technische Universität Wien (TU Wien), Vienna, Austria. Since September 2020, he has been with New York University (NYU) Abu Dhabi, United Arab Emirates, where he is currently a Full Professor and the Director of the eBrain Laboratory. He is also a Global Network Professor at the Tandon School of Engineering, NYU, New York City, USA. He is also a Co-PI/an Investigator in multiple NYUAD Centers, including Center of Artificial Intelligence and Robotics (CAIR), Center of Cyber Security (CCS), Center for InTeractIng urban nEtworkS (CITIES), and Center for Quantum and Topological Systems (CQTS). His research interests include AI & machine learning hardware and system-level design, brain-inspired computing, autonomous systems, wearable healthcare, energy-efficient systems, robust computing, hardware security, emerging technologies, FPGAs, MPSoCs, and embedded systems. His research has a special focus on cross-layer analysis, modeling, design, and optimization of computing and memory systems. The researched technologies and tools are deployed in application use cases from the Internet of Things (IoT), smart cyber-physical systems (CPS), and ICT for development (ICT4D) domains. He has given several keynotes, invited talks, and tutorials, and organized many special sessions at premier venues. He has served as the PC chair, the general chair, the track chair, and the PC member for several prestigious IEEE/ACM conferences. He holds one U.S. patent has (co)authored six books, more than 15 book chapters, more than 350 papers in premier journals and conferences, and more than 50 archive articles. He received the 2015 ACM/SIGDA Outstanding New Faculty Award, the AI 2000 Chip Technology Most Influential Scholar Award, in 2020 and 2022, the ASPIRE AARE Research Excellence Award, in 2021, six gold medals, and several best paper awards and nominations at prestigious conferences. He is a Senior Member of the IEEE Signal Processing Society (SPS), and a member of the ACM, SIGARCH, SIGDA, SIGBED, and HIPEAC.

Open Access funding provided by 'Politecnico di Torino' within the CRUI CARE Agreement