Ministry of Science and Higher Education
Russian Federation
NATIONAL RESEARCH
TOMSK STATE UNIVERSITY (NR TSU)
Faculty of Innovative Technologies
Department of management of information support of innovation activity

ACCEPTED FOR THE DEFENCE IN THE
STATE EXAMINATION COMMISSION
Head of the basic academic program
Professor, Doctor of Engineering
_____S.V. Shidlovskiy
«_____» _____ 2022

MASTER'S THESIS

BOOSTING ADVERSARIAL TRAINING IN ADVERSARIAL MACHINE LEARNING

in the direction of training 09.04.02 – Information systems and technologies
directivity (profile) «Information Systems and Technologies in Science and Instrument Engineering»

Ehsan Saleh

Research advisor
Associate Professor, Doctor of Engineering
Poguda Aleksey Andreevich

«_____»June 2022

Author
Student of group No 182005
_____Ehsan Saleh

Tomsk - 2022

ANNOTAITION

Master's thesis on the topic "Boosting Adversarial Training in Adversarial Machine Learning" contains 80 pages, 32 figures, 13 tables, 190 sources of literature.

The relevance of the research topic lies in researching on new methods to make the Machine Learning models robust against different attacks.

The object of the master's thesis is to find the most optimal algorithms in order to advance the robustness of Machine Learning models.

The subject of research is to develop Adversarial Training techniques which are the most effective methods in increasing robustness of ML models.

Purpose of work is to find the fastest Adversarial Training algorithm and methods to increase robustness of models.

To achieve this goal, the following tasks were performed:

- Study theoretical resources to become familiar with concepts
- Identifying the existing researches on the subject.
- Understanding the problems in the subject.
- Identifying the outstanding solutions.
- Implementing the algorithms.
- Identifying the best available algorithm.
- Testing and tweaking the algorithm.
- Providing the results of the successful implementations.
- Identifying the potentials for further development.
- Suggesting new research directions for further development.

The work consists of an introduction, 4 sections, conclusion, list of references and applications.

The first chapter gives the general introduction of the research, the goals and objectives.

The second chapter deals with the literature review and background to give a general understanding knowledge to help the reader understand the work of the research.

The third chapter gives a holistic view of technical tools needed to utilize the presented information and perform the experiments.

The fourth chapter, which is also the most important part, is divided to different subchapters. Firstly, there is a deeper introduction on the research; then the next subchapter provides a summary of related works done on this subject including common attack methods and famous adversarial training methods. The next part, represents the proposed adversarial training

method in detail. Finally, the experiments are represented in the final subchapter of this chapter in addition to what we can understand from the results of the experiments.

The fifth part, which is the second important part, provides a comprehensive study on a new concept in order to develop the activation function of which is a key part in Machine Learning applications.

In the next part there is a conclusion of the works and future research works.

The results of the work are a fast and meanwhile robust Adversarial Training algorithm as well as a new activation function which can be used in different Machine Learning Tasks.

Ministry of Science and Higher Education
Russian Federation
NATIONAL RESEARCH
TOMSK STATE UNIVERSITY (NR TSU)
Faculty of Innovative Technologies
Department of management of information support of innovation activity

APPROVE
Head of the basic academic program
Professor, Doctor of Engineering

_____ S.V. Shidlovskiy
signature
«_____»_____ 2022 г.

ASSIGNMENT

for the completion of the graduation paper (theses) of master for the student

Ehsan Saleh

Full name of the student

in the direction of preparation 09.04.02 Information Systems and Technologies, directivity (profile) «Information Systems and Technologies in Science and Insrumet Engeenering»

1 Topic of the graduation paper (theses)

Boosting Adversarial Training in Adversarial Machine Learning

2 Deadline for the student to submit the graduation paper (theses):
a) to the study office / dean's office 10.06.2022
b) to the state examination commission 15.06.2022

3 Initial data for work:
Object of research – to find the most optimal algorithms in order to advance the robustness of Machine Learning models.
Subject of research – to develop Adversarial Training techniques which are the most effective methods in increasing robustness of ML models.
Purpose of research – to find the fastest Adversarial Training algorithm and methods to increase robustness of models.

Tasks:

Realization of the most optimal algorithm in Adversarial Training

Providing the methods to increase robustness of Machine Learning Models

Research methods:
Reviewing the outstanding and recently published works on the subject
Reimplementation of the methods
Tweaking the parameters and observing the results
Combine different research works
Implementation of the works in different frameworks (TensorFlow and PyTorch)
Reporting the best results

Organization or industry in which the work is being done:

Faculty of Innovative Technologies, Tomsk State University

4 Brief summary of the work

The work provides methods in order to make the Machine Learning Models more robust against adversarial attacks. This methods are faster than usual previous algorithms.

Head of the graduation paper (theses)

_____	_____ / _____
position, place of work	*signature* *F.S. Surname*

The assignment was accepted

_____	_____ / _____
data	*signature* *Full name*

ABSTRACT

This thesis examines Machine Learning from a software perspective. Machine Learning is already prevalent in our daily lives, as seen by anti-spam filters in electronic mail, facial recognition in cameras, automatic correctors in cellphones, and weather forecasts, to name a few examples. The goal of this thesis is to examine Python-based Fast Adversarial Training (AT) methods for model robustness against adversarial assaults. The term "fast training" refers to a program that can train the model on the database's data in an acceptable amount of time. The major challenge of fast training is to discover a quick but accurate algorithm: too much accuracy may necessitate learning times that are too long to be acceptable, while a high convergence speed may result in incorrect results or, worse, diverge rather than converge. And the term "adversarial training" refers to a code that can withstand data that has been intentionally modified and cannot be distinguished by a human but can have severe consequences for a Deep Neural Network (DNN) model. For example, an attack could change some pixels of an image without making any discernible difference to the human eye but completely misclassified by the DNN model.

By using adversarial instances for training, adversarial training (AT) has been shown to be useful in improving model resilience. Most AT approaches, on the other hand, incur prohibitively high time and computational costs when calculating gradients at several steps in the generation of adversarial samples. Fast gradient sign method (FGSM) is used in Fast AT techniques to increase training efficiency by simply calculating gradient once. Unfortunately, the robustness is lacking. The initialization style could be one of the reasons. Existing Fast AT typically use a random sample agnostic initialization, which improves efficiency but limits future robustness gains. In this work, a sample dependent adversarial initialization is used to improve Fast AT, which is an output from a generative network that is conditioned on a benign image and its gradient information from the target network. Because the generative network and the target network are optimized together in the training phase, the former can create an effective initialization with regard to the latter in an adaptive manner, resulting in steadily enhanced robustness. Experiments on four benchmark databases show that the proposed method outperforms state-of-the-art Fast AT methods and is comparable to sophisticated multi-step AT methods in terms of robustness.

It's a prevalent misconception that networks can't be accurate and robust at the same time, and that growing robustness implies losing accuracy. It is also often assumed that, unless networks are made larger, network architectural components will have little impact on adversarial robustness. A detailed examination of adversarial training is offered here as evidence to dispute these widespread views. The major finding is that, because to its non-smooth character, the commonly used ReLU activation function greatly impairs adversarial training. To enhance

adversarial training, smooth adversarial training (SAT) is presented, in which ReLU is substituted by its smooth approximations. Smooth activation functions are used in SAT to help it locate more difficult adversarial cases and compute better gradient updates during adversarial training.

LIST OF ABBREVIATIONS

ML	Machine Learning
AT	Adversarial Training
NP-problem	Nondeterministic Polynomial time problem
BGD	Batch Gradient Descent
SGD	Stochastic Gradient Descent
PGD	Projected Gradient Descent
DNN	Deep Neural Network
CNN	Convolutional Neural Network
FGSM	Fast Gradient Sign Method
CW	Carlini & Wagner
C&W	Carlini & Wagner
RS	Random Sample-agnostic
SDI	Sample-Dependent learnable Initialization
CE	Cross Entropy
AA	Auto Attack
BN	Batch Normalization
MATLAB	Matrix Laboratory
CUDA	Compute Unified Device Architecture
GPU	Graphics Processing Units
MNIST	Modified National Institute of Standards and Technology
SVHN	Street View House Numbers
CIFAR	Canadian Institute For Advanced Research
SAT	Smooth adversarial training
GELU	Gaussian Error Linear Unit
ELU	Exponential Linear Unit

Table of Contents

INTRODUCTION

In this work, we will first provide a brief overview of Machine Learning, then describe a new and vitally important area of Machine Learning called Adversarial Machine Learning, before focusing on Adversarial Training as one of the best solutions to the problems in the field of Adversarial Machine Learning. We will examine a recently proposed Adversarial Training algorithm and present our observations. Then provide a comprehensive study on a new concept for activation functions which helps in robustness of the ML models. Finally, propose a new research topic for further development of the new Adversarial Training method.

Right after the advent of the first Adversarial Machine Learning discussions, many researchers and engineers have started developing new algorithms for new attacks as well as new defenses on ML models. These attacks could lead to catastrophic results, for example by attacking autonomous driving cars they can cause misinterpretation of road signs and as a result heavy accidents will occur. All these new algorithms and methods are in need of powerful computational resources and time; almost all of the useful algorithms need hours and even days to compile (train the ML model) and neither everyone has access to such powerful resources, nor these resources are free to use. As a result, understanding the optimum methods is critical in order to reduce the amount of time and hardware required for training.

The relevance of the research topic lies in researching on new methods to make the Machine Learning models robust against different attacks. The object of the master's thesis is to find the most optimal algorithms in order to advance the robustness of Machine Learning models. The subject of research is to develop Adversarial Training techniques which are the most effective methods in increasing robustness of ML models.

The purpose of this research is to provide a study on Adversarial Machine Learning, or more precisely Adversarial Training in Computer Vision, and to find the most practical Adversarial Training algorithm in terms of being fast and robust at the time of writing this dissertation, as well as to suggest new research directions for future developments.

The scope of the research is to provide the results of conducted attacks on famous image classification datasets (CIFAR10, CIFAR100, Tiny ImageNet, ImageNet) using different attack methods.

To achieve the goals of this work, the following tasks were performed:

- Study theoretical resources to become familiar with concepts
- Identifying the existing researches on the subject.
- Understanding the problems in the subject.
- Identifying the outstanding solutions.

- Implementing the algorithms.
- Identifying the best available algorithm.
- Testing and tweaking the algorithm.
- Providing the results of the successful implementations.
- Identifying the potentials for further development.
- Suggesting new research directions for further development.

1 Overview (Theoretical)

1.1 What is machine learning

Machine learning [1], [2], [3] is a subdomain of artificial intelligence that is increasingly integrated with the technological systems we use nowadays and is present in our daily lives. Machine learning is used in weather forecasting, email spam filtering, facial recognition in cameras, autonomous driving cars and automatic correctors in cellphones, among other applications.

Machine learning is based on software-implemented mathematical techniques that allow a code to self-adapt to input, allowing subdivision or recognition of the data. Machine learning's major goal is to manage the massive amounts of data generated every day by large corporations in order to manage information more quickly and efficiently. Machine learning can be used to analyze a wide range of data. During this thesis we will mainly focus on images.

Machine learning is divided into two main sections:

- Unsupervised learning
- Supervised learning

1.2 Unsupervised learning

Unsupervised learning [21] (figure 1) occurs when a model has never been trained on a dataset and hence attempts to recognize and split new data that it has never seen before. The model will, of course, already know what kind of data it will receive, but the classification of that data is up to the model. This method can be used, for example, to separate the sound of an audio recording from background noise or to separate the many voices present. It is not generally utilized since it is less successful than supervised learning, but it can aid in the classification of fresh input in circumstances where there is no previous data.

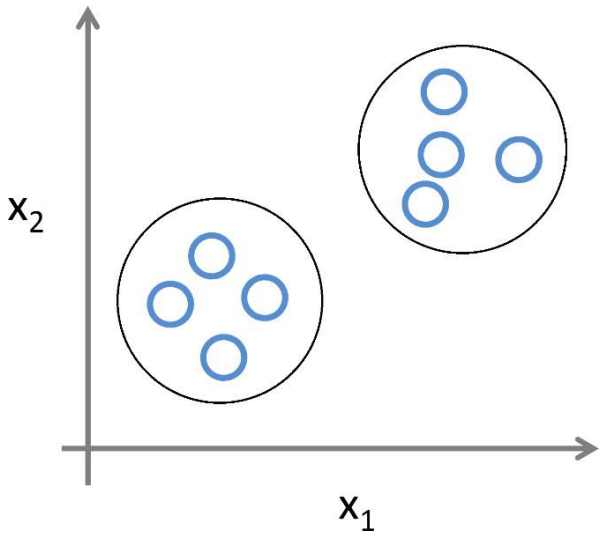

Figure 1 – Unsupervised learning division [21]

1.3 Supervised learning

The phrase supervised learning [21] (figure 2) refers to the most often used machine learning model, which is based on the data's previous experience. In practice, supervised learning involves training a code on an already existing dataset stored in a database, with each data item in the database being accurately labeled. When the code is taught, all labels are shown to be correct; at the end of the training, the code is tested on a fresh dataset, and the number of labels it can properly predict is used to assess its accuracy.

Supervised learning can generate two types of results:

– Regression

– Classification

1.3.1 Regression

The system output will be continuous in regression supervised learning, for example, in weather forecasts between the extreme situations of "good weather" and "bad weather," there can be all values between where the weather is partly cloudy with more or less sun. The trend of a curve that is plotted based on previous data in attempt to forecast the future is known as regression. For example, if you look at house prices based on their size, you can produce a price prediction for a new house whose size is known and you need an economic value to sell it. We shall concentrate on the classification case for the sake of this thesis [21].

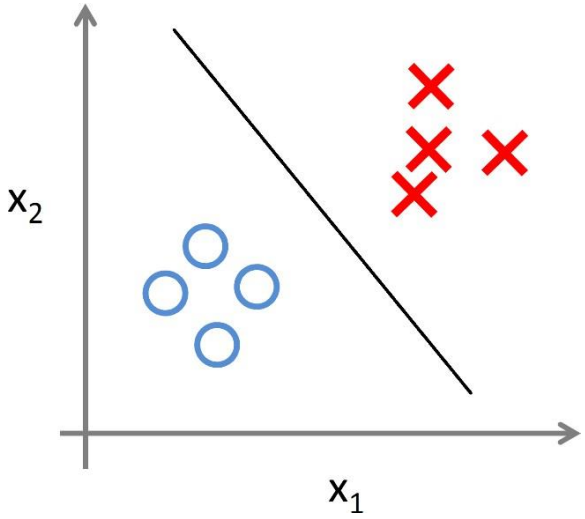

Figure 2 – Supervised learning classification [21]

1.3.2 Classification

The purpose of a classification model, as the name implies, is to divide the data into classes. The most common scenario is binary subdivision: 0 and 1, which assigns the label True or False to each data point. The classification of cancers into benign and malignant forms is an example that can be applied to healthcare. You can show CAT pictures to a supervised machine learning model for classification, and the model will return an output with a 0 or 1 to indicate whether the tumor is benign or malignant. If a larger number of data must be classified, such as the MNIST dataset, which contains 60 thousand images of handwritten digits ranging from 0 to 9, a mathematical trick can be used to assign an intermediate value between 0 and 1 to each label, with the label with the highest value becoming the label to be associated with that specific data [21].

1.4 Cost function

The cost function [21] (figure 3) is an equation that defines the prediction error from the exact label, or how far away it is from the minimum point. Different mathematical formulas can be used to define the cost function, depending on the type of algorithm you want to utilize. The goal of machine learning is to minimize the cost function, which means that the closer the error gets to zero, the more accurately the model can predict the data to be classified. If the cost function is three-dimensional, you must reach the valleys, or the "minimum," in order to have the least amount of inaccuracy. If the cost function isn't as basic as a hyperbolic paraboloid, it will have multiple local minimum points but only one global minimum. Building models capable of "jumping" out of a local minimum to try to reach a deeper minimum is often difficult. When training on big datasets, the training trend is asymptotic: 100% accuracy can only be reached

indefinitely, hence training is frequently terminated after a sufficient number of cycles to achieve acceptable accuracy.

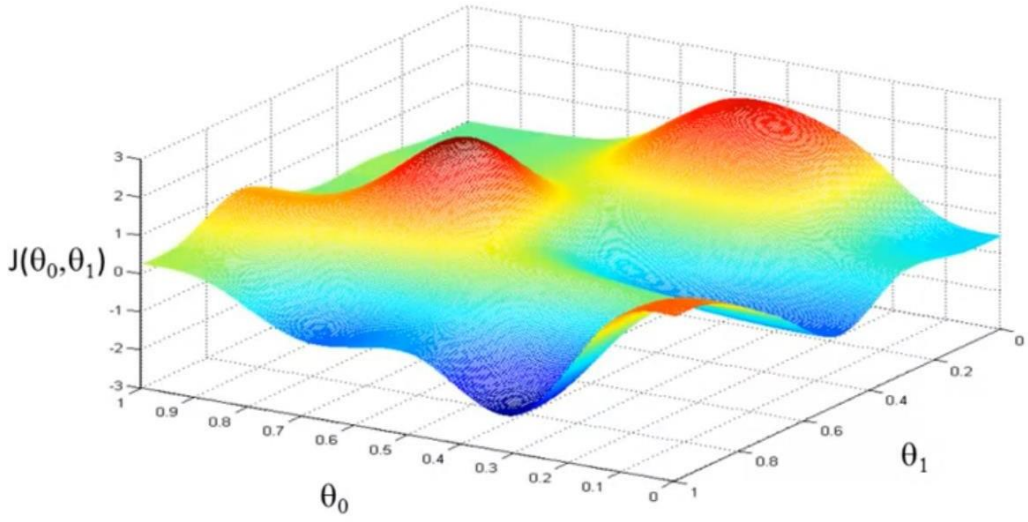

Figure 3 – 3-dimensional cost function [21]

1.5 Gradient decent

Gradient descent is one of the most used methods for determining the cost function's minimum. Its job is to get closer to an area with a negative slope with each iteration, so it uses the derivative to get the angular coefficient of the function for each point each time. If the cost function includes more than two variables, partial derivatives are calculated instead of the derivative to obtain the gradient, hence the name gradient descent [21]. Here is a typical gradient descent equation:

$$\theta_J = \theta_J - \alpha . \frac{\partial}{\partial \theta_J} J(\theta) \qquad (1)$$

Where the parameter α is the learning rate, that is the speed of the gradient descent, but we'll talk more about it in section 2.7.1.

1.5.1 Batch gradient descent (BGD)

BGD [21] (figure 4) is suitable for small datasets because it uses all of the data to calculate the gradient descent at each step, which significantly slows down the training process. Mini-batch gradient descents, or BGDs calculated on the smaller portion of the dataset, are utilized in the situation of enormous datasets.

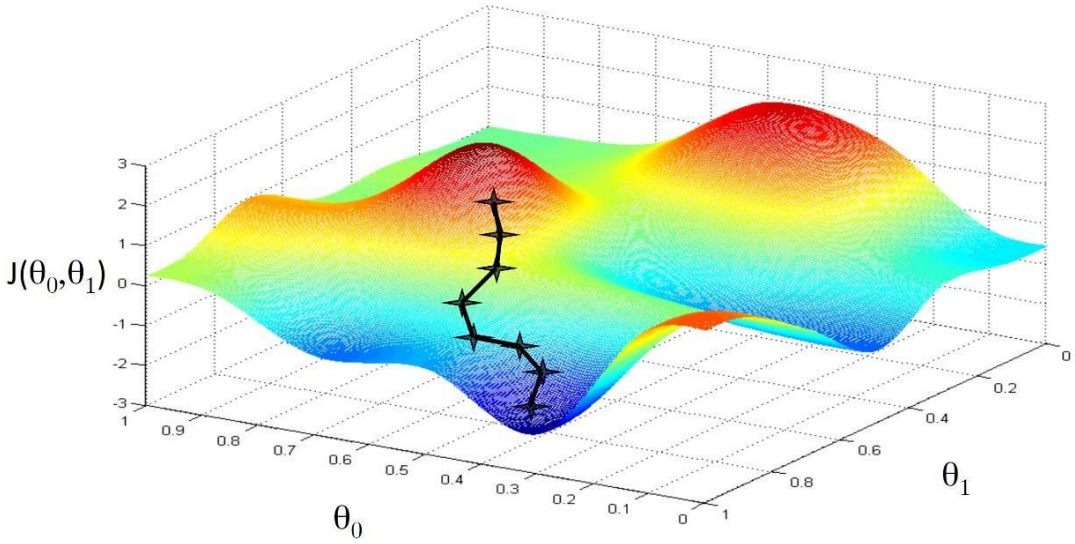

Figure 4 – Batch gradient descent [21]

1.5.2 Stochastic gradient descent (SGD)

Because gradient descent is a computationally expensive operation, the SGD (figure 5) function searches for an optimal path to the smallest zone at random. This allows you to avoid calculating the gradient descent for the entire database at each iteration. As a result, it is a technique used primarily for large datasets that does not clearly follow the path of descent to the shortest minimum, but takes longer to find the minimum. This is compensated by the fact that the number of iterations required to produce the SGD is much lower than that required to produce a normal gradient descent [21].

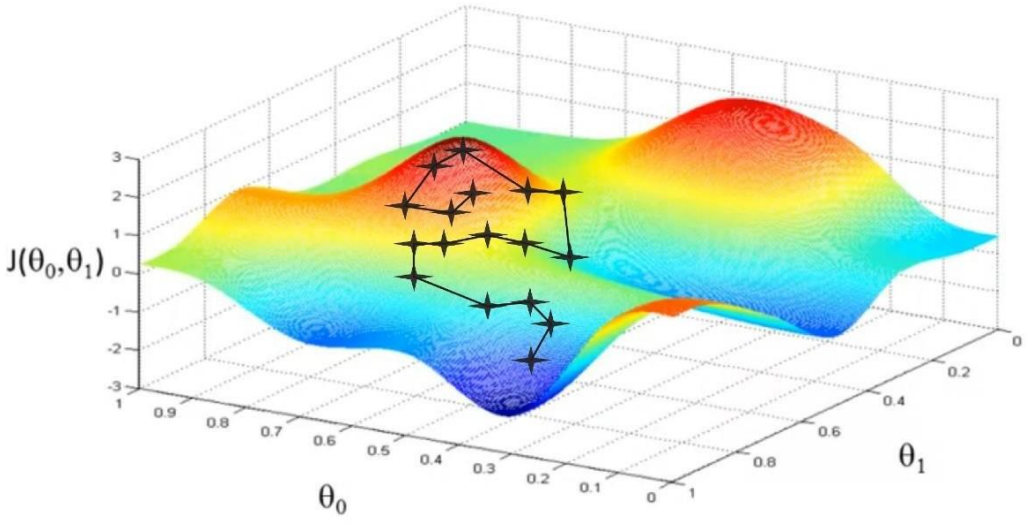

Figure 5 – Stochastic gradient descent [21]

1.6 Normal equation

The normal equation is a gradient descent alternative that does not employ the learning rate hyperparameter, does not require iterations, and instead relies on matrices as large as the features, or system inputs. This strategy, however, is rarely utilized since, while it is faster for small datasets, it is unmanageable when dealing with massive volumes of data. It is particularly difficult to modify, and despite the fact that it does not require many cycles, it still requires the calculation of the inverse of some matrices, an algebraic operation that is frequently not advised and can result in singularities [21].

$$\theta = (x^T . x)^{-1} . x^T y \qquad (2)$$

1.7 Hyperparameters

With hyperparameters [4] we mean all those parameters that directly influence the training, during the course of this thesis we will mainly consider the following hyperparameters:

- Learning rate
- Momentum
- Batch size
- Weight decay
- Epochs

1.7.1 Learning rate

The learning rate α (figure 6) is the most critical hyperparameter since it regulates the speed with which the gradient descends to the cost function's minimum. It is the angular coefficient of descent and can be adjusted throughout the training. It exists in the gradient descent formula regardless of the cost function and its derivative. The higher the learning rate, the faster the gradient decreases along the slopes; however, if the value is too high, the gradient may fail to reach its minimum and diverge. To identify the highest value of the learning rate that can be utilized for a given model and dataset, run a test as an initial training where the learning rate varies exponentially from very small values to very large values, passing through multiple orders of magnitude to find the optimal order of magnitude. Further on we will show you how to find the correct learning rate. In most trainings, the value of the learning rate starts to drop towards the end of the training in order to go deeper into the local minimum in which you are [21].

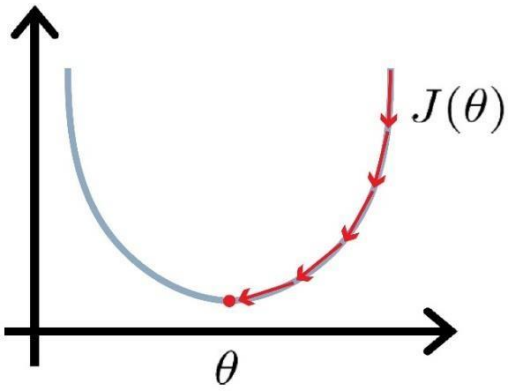

Figure 6 – Normal learning rate [21]

1.7.2 Momentum

The momentum [21] is a hyperparameter that is added to control other parameters during the training, therefore their weight in speeding up the training in order to quickly reach convergence. During iterations, the weight of the gradient descent, for example, can change. Although having momentum during a training is not required, it does allow you to rapidly monitor the weight of the equations within the code throughout the training period without having to act directly on the code for modifications.

1.7.3 Batch size

The batch size [5] is a parameter used especially in large datasets because often the computational capacity of the computers does not allow to manage all the available data at the same time and therefore the training is iterated slowly over a dataset at a time, this set is the batch size. For example, if the dataset contains 50000 photos and the batch size is set to 100, each cycle will consist of 500 iterations, each with 100 different images from the dataset, resulting in each cycle analyzing all of the images once, but in blocks of 100. Often the batch size is set to a power of 2 because the calculators have an available memory calculated as a multiple of 2, obviously the larger the batch size the better the training, because the number of images compared at each iteration will be greater, but at the same time the training will become increasingly slow. A value for the batch size between 100 and 1000 is often more than acceptable as a compromise between total training time and quality of the result. The batch is one of the factors that directly affects the duration of the training.

1.7.4 Weight decay

The weight decay [21] describes the rate at which the weight of particular variables decays during the training process, specifically the weight of the coefficients of variables that become less

relevant over time than the cost function's constant term. This is owing to the fact that not all variables in the cost function are equal in importance, and some are more influential than others. To determine if a patient has a certain type of tumor or not, all of the input data is analyzed, such as age, weight, gender, previous diseases, alcohol dependence / smoking, and so on. However, not all of these variables are equally important for the tumor under consideration, and the hair color is almost unimportant, so it is not even considered for evaluation purposes. This basic example shows how to apply the weight decay based on the variables. The weight decay has been maintained constant throughout this thesis to avoid affecting the other hyperparameters, however nothing prohibits it from being changed during the training to improve outcomes.

1.7.5 Epochs

Epochs [5], which represent the overall number of training cycles, are arguably the easiest hyperparameter to grasp. All data is analyzed at least once throughout each epoch, and all operations are carried out. Unless other hyperparameters are configured in such a way that they gradually change as the epochs pass, each epoch is identical to the preceding one. Theoretically, if all of the other hyperparameters are adjusted appropriately, an accuracy of 100 percent on the data can be attained for an indefinite number of epochs, i.e., the learning curve is asymptotic and goes to infinity. Epochs, along with batch size, are two characteristics that have a direct impact on the training duration. The training always lasts a finite and full number of epochs, and the equation for calculating cycles in a single epoch is as follows:

$$1\ epoch = \frac{\text{dataset size}}{\text{batch size}} \tag{3}$$

1.8 Datasets

To work in the field of machine learning, it is necessary to have databases on which to perform the training. There are several ready to use, more famous ones are presented below:

- MNIST.
- SVHN.
- CIFAR10.
- CIFAR100.
- ImageNet.

Usually, each dataset is divided into 2 different subsets:

- Training set: used to train the model in image recognition and to test its accuracy.

– Validation / test set: an external dataset used to test the model on images it has never seen before, ensuring that it has not "become accustomed" exclusively to the training set and that it can also cope with unfamiliar data. In most cases, the accuracy of test set is lower than with the training set. In general, the amount of data in a test set is less than that in a training set.

1.8.1 MNIST

The MNIST [6] is a dataset consisting of 60 thousand images, divided into:

– 50 thousand images for training;

– 10 thousand images for validation / test.

The images are divided equally into 10 classes depicting the digits 0 to 9 written by hand (figure 7) and therefore the goal of this dataset is to train models capable of recognizing and interpreting handwritten numbers. The images have a 28x28 pixels format in black and white. Part of this dataset was mainly used in the first part of this thesis.

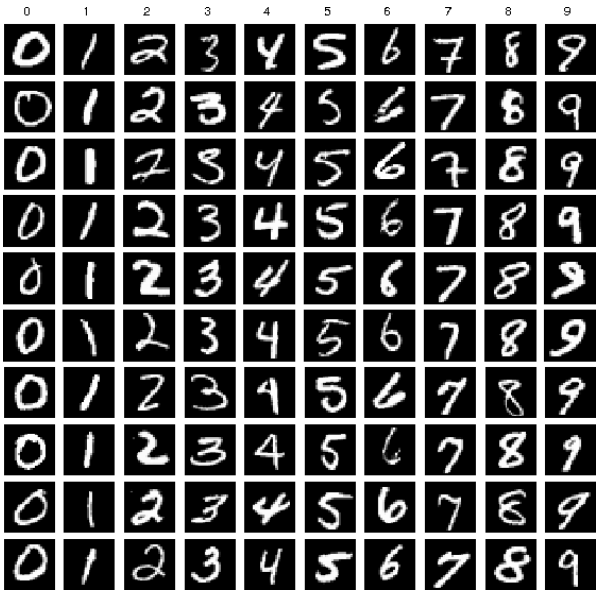

Figure 7 – An example of the MNIST dataset [6]

1.8.2 SVHN

SVHN is a real-world picture dataset with low data preparation and formatting requirements for building machine learning and object recognition algorithms. It has a similar taste to MNIST (for example, the images are of little cropped digits), but it contains an order of magnitude more labeled data (about 600,000-digit images) and is based on a far more difficult, unresolved real-world problem (recognizing digits and numbers in natural scene images). House numbers in Google Street View pictures [7] are used to calculate SVHN.

It is divided into ten groups, one for each digit. There are 73257 digits for training, 26032 digits for testing, and 531131 more, slightly less challenging samples to utilize as supplemental training data. It is presented in two different formats:

– Original images with bounding boxes at the character level.

– 32-by-32 images centered around a single character, similar to MNIST (Many of the images have distracting elements on the sides.) (figure 8).

Figure 8 – An example of the MNIST dataset [7]

1.8.3 CIFAR10

The CIFAR10 [8] is also similar to the MNIST, but instead of using classes depicting handwritten digits, it has 10 classes depicting various things, including: airplanes, cars, dogs, cats, etc. (figure 9) Each image is exclusive to the others, that is, a single image cannot represent two or more classes simultaneously. The size of each image is 32x32 pixels in color.

1.8.4 CIFAR100

The CIFAR100 [8] is a dataset similar to the CIFAR10 composed of 60 thousand images, equally distributed to the cifar10, but instead of 10 classes there are 100 different classes and therefore consequently it is more complicated to be able to obtain a good result during the training.

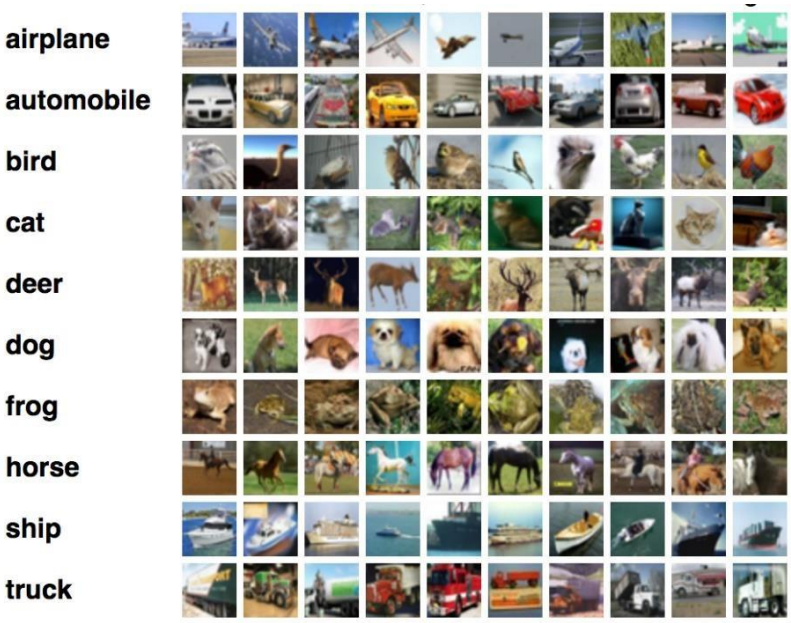

Figure 9 – An example of the CIFAR10 dataset [8]

1.8.5 ImageNet

It is one of the largest existing datasets [9] [10], it contains more than 1 million images, with various formats and divided into 1000 classes. This dataset is often used for the final validation of a model, but requires considerable hardware resources.

1.9 Linear regression

Linear regression (figure 10) is one of the most fundamental approaches to machine learning, and it involves dividing data by drawing a line that can detect the delimitation area between the various groups [21]. It belongs to the supervised regression learning category, and its constitutive function can always be reduced to a polynomial sum, which includes all of the inputs. The simplest form is:

$$h_\theta(x) = \theta_0 + \theta_1 x \qquad (4)$$

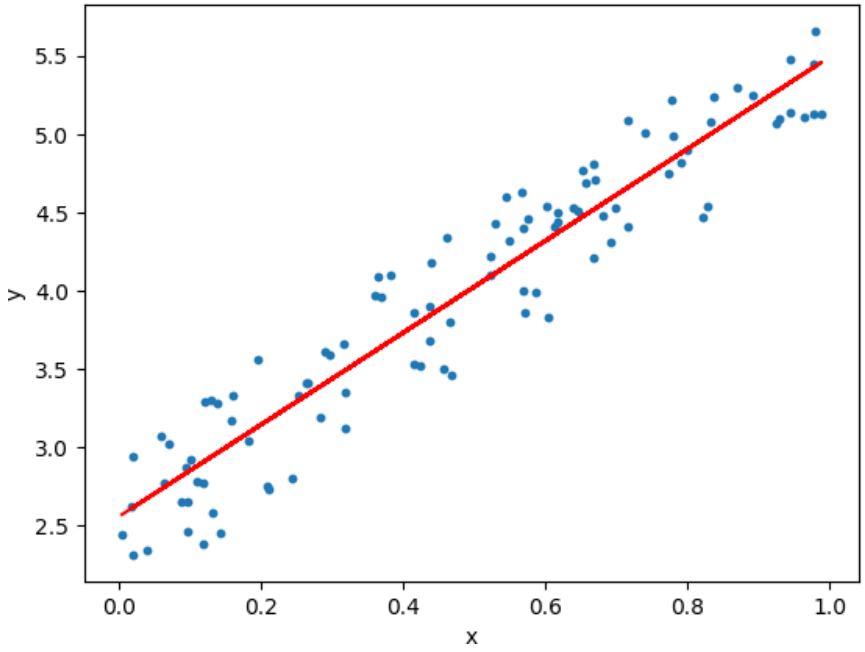

Figure 10 – Linear regression [21]

The statistical variance, or an average of the square of the distance between the expected value (y) and the value gained during training, is the cost function in this case (h).

$$J(\theta) = \frac{1}{2\mathrm{m}}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 \qquad (5)$$

The gradient descent is obtained simply by updating the past gradient from time to time with the new gradient of the cost function damped with the learning rate hyperparameter.

$$\theta_J = \theta_J - \alpha.\frac{1}{\mathrm{m}}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}) \,.x_J^{(i)} \quad with \quad x_0^{(i)} = 1 \qquad (6)$$

1.10 Logistic regression

Logistic regression is a type of supervised learning, however unlike its name, it is a classification model. The term logistic refers to a function that allows you to go from a continuous to a discrete domain to get a classification; this function is also known as Sigmoid (figure 11) [21].

$$g(z) = \frac{1}{1+e^{-z}} \qquad (7)$$

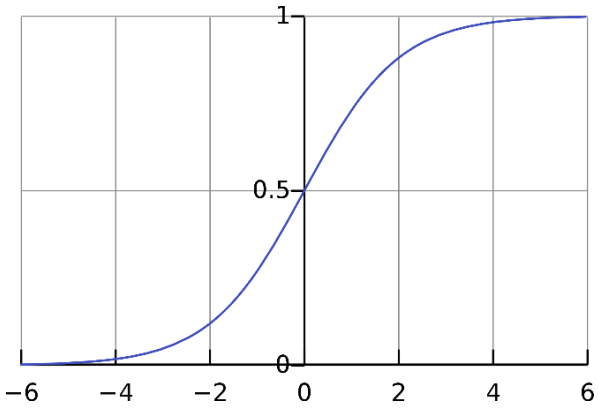

Figure 11 – Sigmoid function [21]

As we can see, the sigmoid function classifies all inputs as 0 or 1, rounding to the nearest integer.

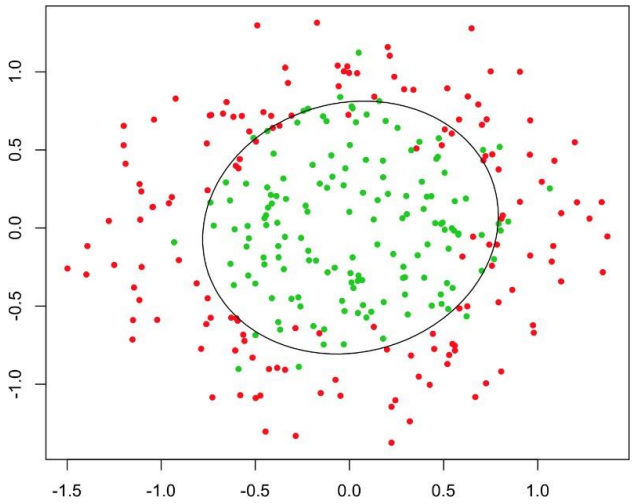

Figure 12 – Logistic regression [21]

In the specific case of logistic regression (figure 12), the constitutive function that exploits the sigmoid function can be written as:

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1+e^{-\theta^T x}} \quad \rightarrow \quad 0 \leq h_\theta(x) \leq 1 \qquad (8)$$

The cost function deriving from this function exploits the logarithms to be able to dampen the effects of the exponential basis of the sigmoid function and therefore turns out to be different from the cost function of the linear regression.

$$J(\theta) = -\frac{1}{\mathrm{m}} \sum_{i=1}^{m} \begin{bmatrix} y^{(i)}.\log\left(h_\theta\left(x^{(i)}\right)\right) + \\ \left(1-y^{(i)}\right).\log\left(1-h_\theta\left(x^{(i)}\right)\right) \end{bmatrix} \qquad (9)$$

The equation for calculating the gradient descent is similar to that of linear regression, but obviously it will have a different constitutive equation inside.

$$\theta_J = \theta_J - \alpha.\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right).x_J^{(i)} \qquad (10)$$

1.11 Neural network

As the name implies, a model based on a neural network [11] is inspired by the intricacy of our brain. These models are among the most successful, and it is thought that by designing versions that are increasingly comparable to the natural shape of the human brain, accuracy during training can be increased as well. Neural networks are based on the concept of networks (figure 13), in which each node is linked to a large number of other nodes.

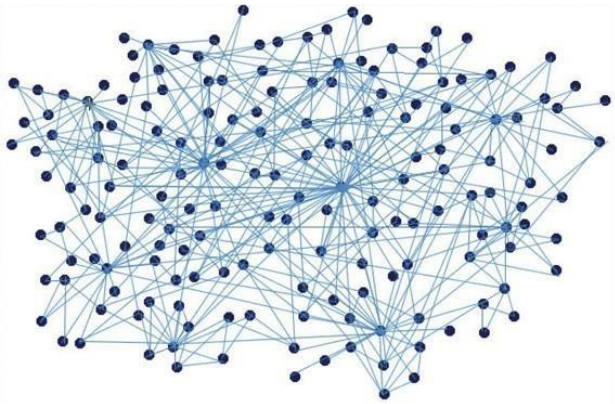

Figure 13 – Network with nodes and links [21]

In this specific case the nodes are divided into layers (figure 14), and the more layers there are, the more complex the neural network becomes.

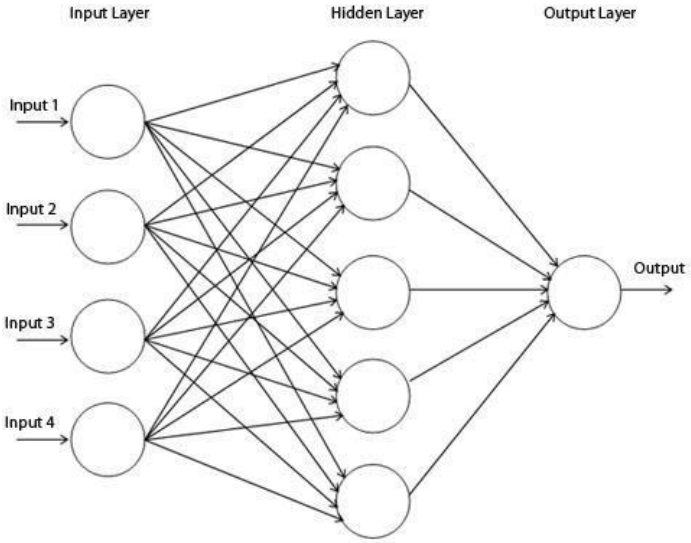

Figure 14 – Neural network with one hidden layer [21]

A neural network is defined as deep (DNN) [12]-[16] when in addition to the input and output layers there are more than 1 hidden or transition layers (figure 15).

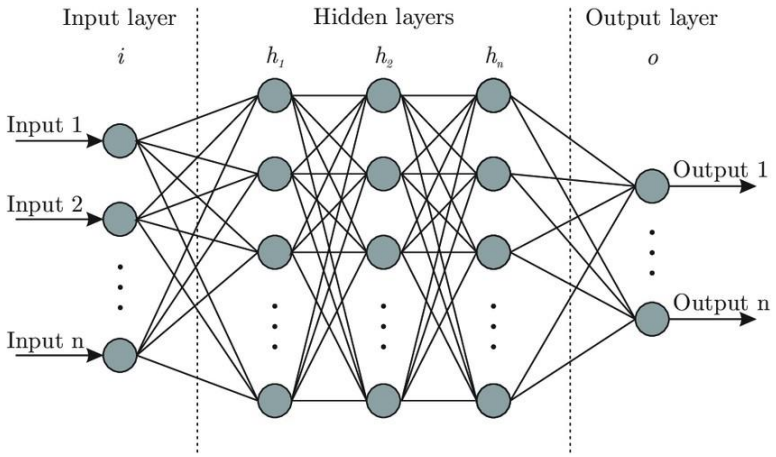

Figure 15 – Deep neural network with more than one hidden layer [21]

The neural network falls under the supervised classification learning category and can classify various categories based on the number of nodes in the output layer. For example, if we are working with the CIFAR10 dataset, which classifies 10 different objects, our deep neural network will have 10 nodes in the last layer, the output layer, and each node will correspond to a class. The input nodes, on the other hand, correspond to all input parameters, whilst the central layer nodes have no physical meaning and can be in any number. Because the neural network is a classification model, the sigmoid function is also used in this scenario, and the resulting constitutive function is:

$$J(\theta) = -\frac{1}{\mathrm{m}}\sum_{i=1}^{m}\sum_{k=1}^{k}\left[y_k^{(i)}.\log\left(h_\theta\left(x^{(i)}\right)\right)_k + \left(1-y_k^{(i)}\right).\log\left(1-h_\theta\left(x^{(i)}\right)\right)_k\right] +$$

$$\frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_l+1}\left(\theta_j^{(l)}\right)^2 \qquad (11)$$

In order not to make the subject too heavy, other technical details on neural networks will not be provided, but for more detailed studies, the publications mentioned at the end of the thesis can be used [11]-[16]. Through various researches, it has been noticed that by increasing the number of hidden layers too much, we reach a point where the accuracy of the model does not grow, but on the contrary begins to decrease, therefore various variations to the classic DNN have been invented, including the CNN (convolutional neural network).

1.11.1 Convolutional neural network (CNN)

The concept for this form of neural network came from the visual cortex of animals (figure 16). The main idea behind a CNN is to divide the images into sections and extract the most essential properties from each of them for evaluation purposes. CNNs are mostly used for image and natural language processing. There are several CNN models [17], the most prominent of which are:

– AlexNet
– VGG
– LeNet
– GoogLeNet
– ResNet

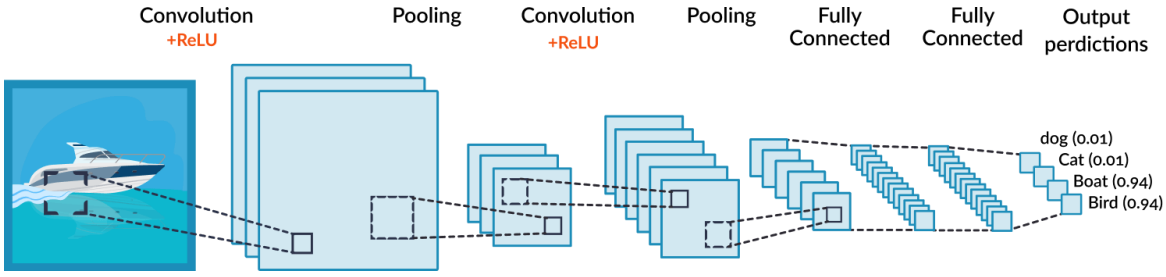

Figure 16 – Scheme of convolutional neural network [21]

Not all the nodes of each layer are connected to the nodes of the next layer in some CNN variants, such as ResNet (residual neural network) [18]-[20], and not all the layers are considered at each cycle. This is because it has been observed that even the neurons of the human brain do not have perfect connections, and as a product of nature, they have many gaps and variations (figure 17).

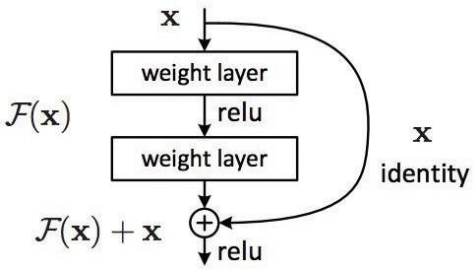

Figure 17 – Close up on ResNet model [18]

1.12 Problems and solutions

There can be various types of errors during a machine learning process, we will see some of these and their solutions in this paragraph.

1.12.1 Features scaling

The features are all of the input parameters that the training is based on, but these variables do not necessarily have values of similar orders of magnitude, which might cause computing issues

when operations between very large and extremely small values are used. To solve this difficulty, feature scaling is utilized, which involves rescaling all parameters on the same interval, such as [0,1], so that each parameter, despite having various meanings, may be read by the machine with the same computing effort [21]. The following formula is used to obtain this rescaling:

$$x_{new} = \frac{x}{N_x^\circ} \quad with \quad N_x^\circ = maximum\ value\ of\ x$$
$$e.g.: if\ 0 \leq x \leq 500 \ \rightarrow \ N_x^\circ = 500 \tag{12}$$

1.12.2 Mean normalization

Mean normalization is an operation that is often carried out in parallel with the feature scaling in order to obtain not only values within a certain range, but also with a certain average value, therefore if for example the interval is [-1, 1] then the average value will be around 0.

$$x_{new} = x - x_{mean} \quad with \quad x_{mean} = mean\ value\ of\ x \tag{13}$$

These feature scaling and mean normalization operations must be carried out for each individual feature taken into consideration before the start of the training process.

$$x_{new} = \frac{x - x_{mean}}{N_x^\circ} \tag{14}$$

1.12.3 Learning rate problems

The learning rate is the most significant hyperparameter, as stated in the paragraph dedicated to it, and as a result, it is the one to which greater attention should be paid.

With overly tiny values (figure 18 - left), there is a chance of not moving forward during the training, remaining in a stall area, or otherwise moving so slowly that the training is rendered meaningless. On the other hand, with too big values (figure 18 - right), there is a risk of not being able to enter the function's minimum in order to boost accuracy, and in the case of extreme values, it is even conceivable to go to divergence and thus bring the accuracy to near-zero values.

Figure 18 – Left: small learning rate; Right: large learning rate [21]

1.12.3.1 Learning rate finder

There are various dedicated libraries [21] for different programming languages to find the correct value of the learning rate, but the simplest method is to perform a fairly long training, during which the learning rate changes exponentially from very small values to very large values; the best choice is to try to vary the learning rate by at least 10 orders of magnitude throughout the training. Theoretically, if all the various parameters have been standardized, the learning rate will often be a value between 0.001 and 10, hence it is preferable to cover the entire range during training. After the test training is completed, the maximum recommended learning rate can be determined by looking at the accuracy and error graph.

When the learning rate is very small [21], as shown in figure 19, accuracy and error practically do not vary; however, from a certain point onwards, the error decreases until it reaches a minimum, then rises and diverges. This demonstrates that the learning rates that are actually useful are those that are included in the error's area of descent, that is, from the initial plateau to the minimum point, beyond which the divergence begins. To stay far enough away from the divergence zone, it's best to use a maximum learning rate that's about one order of magnitude lower than the one found for the minimum point of the error. As a result, the learning should range from the start of the error slope to an order of magnitude before the minimum point.

1.12.4 Training problems

During training [21], there may be issues with the dataset in use and how the data are evaluated; as a result, the training may result in a model that is either overly or underly suited to the data on which it was trained. These issues are mostly divided into two categories of underfitting or overfitting.

Figure 19 – Learning rate finder [21]

1.12.4.1 Underfitting

Underfitting (figure 20) shows a high-bias problem, i.e., the error is too large and the model hasn't trained sufficiently or isn't suitable for a specific type of dataset. Generally, whether you test the model on the training set or the validation set, the error will be high in both situations. To solve

this problem, increase the number of features in the model by attempting to make it more exact, as well as the training duration so that the model has enough time to learn the information [21].

1.12.4.2 Overfitting

When overfitting [21] (figure 20) occurs, it indicates a high variance problem, implying that the model has adapted precisely to the dataset on which it was trained, but would not be able to reproduce the same findings using a dataset that has never been seen before. This error is common and can be detected when the error for the training set continues to reduce with time, while the error for the test set approaches an asymptotic level below which it does not decrease or decreases at a slower rate than the training set. If, up until a few epochs ago, the accuracy that differentiated the training set from the validation set in percentage was nearly constant, it now tends to grow, indicating a model that has become too habituated to the training set. To prevent this issue, it's usually enough to reduce the number of epochs if they're too many, or, in most circumstances, to give the model a larger margin of error, the number of constraints and features must be reduced. Dropout layers can be used with DNN models to overcome the overfitting problem by randomly removing specific features by setting them to zero for each epoch. In general, for underfitting and overfitting problems, in order to develop a trustworthy and robust model, it is possible to operate on hyperparameters or features using regularization factors.

1.12.5 Random initialization

In machine learning, initial values for variables are frequently required, and they are typically set to zero. However, this choice is not always correct, because it is preferable not to have initial symmetries in the model because they could cause the model to suddenly diverge or not move at all from the initial null values, especially in complex cases.

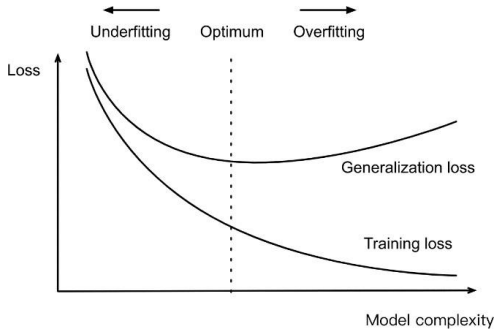

Figure 20 – Underfitting and overfitting

For example, it may appear like placing a ball on top of a pointed pyramid is ideal and symmetrical, but due to the model's unstable equilibrium, even a minor disruption will cause it to collapse. In contrast, if the ball is placed at the bottom of a narrow and high basin, it will struggle

to climb the walls even if powerful disturbances are present. As a result, to avoid singularities, the initial variables are frequently placed at random. This leads in different outcomes for each launch of the same training, but statistically, the changes are minor, and the average will yield acceptable and validated results [21].

1.13Adversarial Machine Learning

Adversarial machine learning is a serious security problem. On the one hand, defending against such assaults is crucial for real-world deep learning applications, particularly in safety-sensitive industries such as autonomous driving [22], malware detection [23], and medical machine learning [24]. On the other hand, designing attack and defensive algorithms opens up new areas of machine learning, which helps us gain a better knowledge of the learning framework as a whole, allowing for the development of more robust and sophisticated learning algorithms and protocols.

It's crucial to understand how adversarial samples are generated in order to design protection algorithms and understand adversarial attacks. In most cases, the generating technique may be expressed as an optimization problem. The problem is commonly represented as unconstrained non-linear programming in the machine learning field, and then solved using stochastic gradient descent algorithms.

We could only discuss a few pioneering and important papers on adversarial machine learning because there are so many. Although there are extensions of adversarial assault in the learning framework, such as reinforcement learning [25]-[27], we mainly focus on supervised learning, which typically have two phases: training and inference. The evasion attack, also known as an adversarial attack, is launched during the inference stage, whereas the poison attack is launched during training.

1.13.1 Evasion Attacks and Defenses

After Goodfellow et al. [28] proved the vulnerability of neural networks in the picture domain to small perturbed samples (adversarial samples), an iterative version [29] was created to improve attack performance. Madry et al. [30] suggested a more powerful attack strategy that deceives neural networks into classifying the sample as a targeted label, thereby negating the distillation defense [31]. More attack approaches, such as Saliency Map and DeepFool [32] [33], are both well-explained and computationally efficient. For further insight, see a recent overview of adversarial attacks on deep learning in computer vision [34]. Despite their origins in the image domain, adversarial samples have quickly expanded to other deep learning domains such as natural

language processing, network structure data, and 3D point clouds [35]-[37]. The most relevant countermeasure for adversarial samples is adversarial training [38], which is one of many. Other protection approaches, such as distillation [31], adversarial detection [40], network verification [41], and so on, are covered in a comprehensive survey by Yuan et al. [39]. However, computational costs are significantly higher, and model accuracy suffers as a result of the increased robustness [42]-[44].

1.13.2 Data Poisoning and Backdoor Attack

The goal of a data poisoning attack is to degrade the overall performance of machine learning systems. It usually occurs during the training phase [45]. Recently, a similar but more severe threat to machine learning termed backdoor attack has been presented, based on data poisoning (see, e.g., [46]-[48]). These attack methods make use of the data poisoning attack pipeline, but inject poisoned data in a finely designed manner so that a hidden harmful 'backdoor' can be implanted in the neural network. The model performs normally when standard data is tested but will be activated to fulfill a certain task once a specific pattern, or known as a pre-defined trigger is shown in the data. There are also recent works connecting adversarial attacks and backdoor attacks, for example, Weber et al. [49] yields a provable model against both attacks; Zhao et al. [50] provides a geometric view in loss landscapes through mode connectivity, which can be used to study both adversarial robustness and backdoored models.

1.13.3 Typical Adversarial Samples

We should first outline the attack tactics for creating adversarial samples before proceeding. In white-box attack scenarios, which is when the opponent knows the neural network's topologies and parameters, the two most prevalent attack tactics will be addressed here. Assume we have a classifier (typically a neural network) $F_\theta(\cdot)$, and a loss function $l_\theta(x, y)$, where x and y are the input data and associating target label respectively in an adversarial attack scenario [64].

Fast Gradient Sign Method (FGSM):

FGSM concerns a perturbation on x:

$$x' = x + \epsilon . sign(\nabla_x l_\theta(x, y)) \tag{15}$$

where ϵ is a hyperparameter that has been chosen as being small enough. The key concept is to regard the loss function as a linear function and identify the perturbation that maximizes it under the l_∞ norm. In practice, this strategy works well. Kurakin et al. [29] offer an iterative Fast Gradient Sign Method that is more powerful:

$$x'_0 = x, \quad x'_{j=1} = \prod_s (x'_j + \epsilon . sign(\nabla_x l_\theta(x, y))) \tag{16}$$

where $\prod_s(x')$ is a projection from x' to the space S, for example $S := [0,1]^d$ where d is the dimension of data x. Figure 21 from Goodfellow et al. [28] for a demonstration of FGSM attack on a panda image.

Carlini&Wagner Attack (CW):

In order to find an adversarial sample $x' = x + \delta$, Carlini & Wagner [51] formulates the problem as:

$$\min_{\delta} \|\delta\|_p + \lambda . g_t(x + \delta) \tag{17}$$

where $\|\cdot\|_p$is the standard l_p norm, t is the target label that the adversary is misleading to, δ is the perturbation on input x to generate adversarial sample $x' = x + \delta$, κ is a chosen non-negative threshold and:

$$g_t(x') = \max\left\{\max_{j \neq t} F(x')_j - F(x')_t, -\kappa\right\} \tag{18}$$

is a loss function that reflects the difference between probability of the target label (the classifier $F(x)$ will output probability of x being classified as label j, which we denote by $F(x)_j$) and another label with largest probability. Indeed, if we use stochastic gradient descent algorithm to solve the problem, it will try to find a δ such that $F(x + \delta)_t - \max_{j \neq t} F(x + \delta)_j \geq \kappa$ since such δ makes $g_t(x + \delta)$ a constant. Also, in the meanwhile it will keep l_p norm of δ small. Thus, we can consider κ as a gap between $F(x + \delta)_t$ and $\max_{j \neq t} F(x + \delta)_j$. The larger κ is, with higher probability $x + \delta$ is classified as label t.

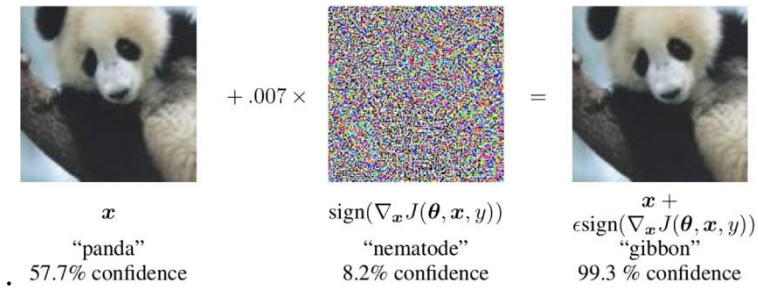

Figure 21 – FGSM Attack Demonstration;

The classifier used is GoogLeNet [52]. Under FGSM attack, a small perturbation on a panda picture makes GoogLeNet misclassify it as gibbon with 99.3% confidence, which is much higher than the confidence of the correct label.

2 Overview (Technical)

2.1 Programming languages

It is required to employ programming languages in order to enter the world of machine learning. There are various, but those that allow efficient matrix calculation are the most effective for this purpose, as machine learning is primarily dependent on the management of massive matrices. As a result, high-level languages are more useful, and the interpreter is left to do the majority of the work. Here are the two famous languages that are mainly used in ML domains:

- MATLAB
- Python

Each programming language has its own optimized libraries for mathematical calculations, however dedicated machine learning libraries have been built and released for a variety of languages. Below we have listed some of the famous ones:

- TensorFlow
- PyTorch
- Keras

2.2 MATLAB

MATLAB [53] is a programming language designed for calculating matrices and displaying plots. Given the size of the dataset to be managed, writing code in MATLAB increases the style and comprehension of the code. It would have required far more lines of code to do the same process in a language like C, which explains why MATLAB is so beneficial in this field. However, because it is not an open-source language, it lacks many libraries optimized for machine learning, necessitating the use of the Python language for more in-depth research.

2.3 Python

Python [54], [55] is an object-oriented and open-source high-level programming language created by Guido van Rossum in the 1990s. It can be used for a variety of tasks, like developing an app, sending an email, and delivering notifications through a Telegram bot, among others. It has been continuously updated and expanded, although there has been a split from version 2.7, resulting in the development of version 3.x. Only version 3.x is continually updated at the moment, and it is in many ways incompatible with version 2.7. Because it is open-source, it is one of the

most widely used languages in the world, including in science [1] [2], for which there are a number of specialized libraries, including:

- Numpy - for matrix calculation and generic mathematical functions
- Scipy - for mathematical analysis
- Matplotlib - for plotting graphs
- Pandas - for data analysis
- Scikit - for machine learning

Python is not as quick as MATLAB because it is not optimized for matrix calculations, but it can manage matrices thanks to the aforementioned packages.

2.4 TensorFlow

Google's TensorFlow [56] [57] machine learning library was launched in 2015. Because they are both open source and TensorFlow allows you to automate training very efficiently, it has become the most often used library in conjunction with the Python language since its introduction. TensorFlow is not really straightforward [58], however it comes with a lot of basic methods for code simplification.

TensorFlow is built on tensors [59], which are multidimensional matrices, and is a calculation based on the flow of tensors, as the name suggests. TensorFlow, despite being written in Python, does not follow its logic [60] and instead uses a graph structure to calculate. When creating variables in TensorFlow, they are not immediately kept in RAM; instead, they become part of a graph that includes all of the variables and operations to be done, and the graph is generated only after the run command is issued.

TensorFlow, like PyTorch, may be utilized with CUDA [61] to take advantage of Nvidia GPU computational capability. TensorFlow's current version is 2.x, but many codes created with version 1.9 or older differ in part from those written with version 2.x.

2.4.1 TensorBoard

TensorBoard (figure 22) is a tool that provides a graphic interface to TensorFlow and acts as a valid help tool for:

- Plotting accuracy and loss
- Displaying the TensorFlow graph
- Showing histograms and tensors values that change over time
- Displaying the data, such as images, texts, etc.

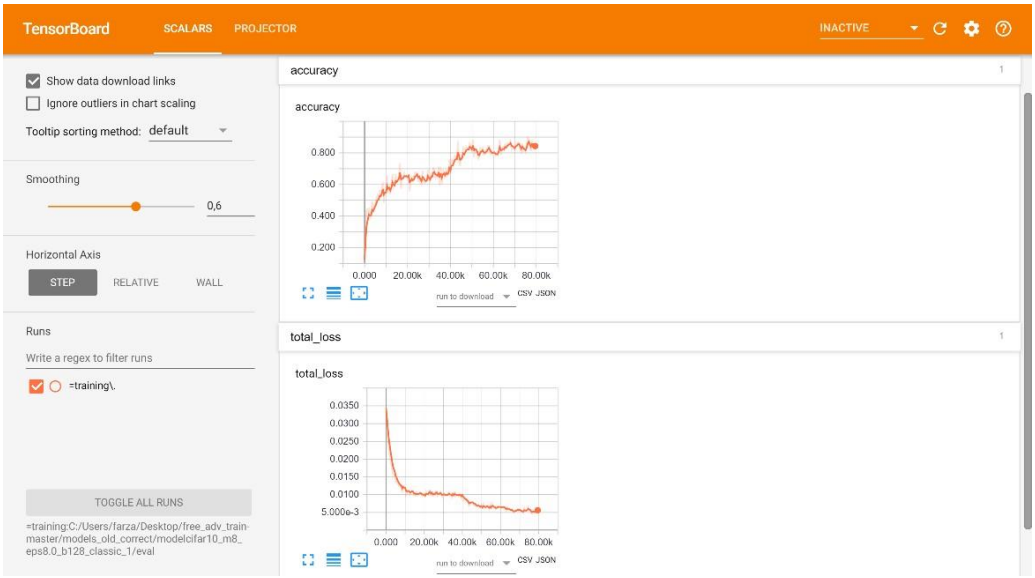

Figure 22 – An example of TensorBoard interface

2.5 PyTorch

It's a machine learning library [62] released by Facebook. It is more intuitive than other libraries in the area and provides for ease of use, but because it was launched later than TensorFlow, many scripts are presently developed only for TensorFlow, which has more thorough documentation. PyTorch uses a class named Tensor to manage arrays, allowing it to generate multidimensional arrays with operations that can also be done on CUDA-capable [61] Nvidia GPUs.

2.6 Keras

Keras is a library [63] for machine learning currently written only in Python language. It is mainly used for a rapid realization of DNNs. Compared to other libraries it is at a higher level, allowing a higher level of abstraction. It can be used in conjunction with TensorFlow.

3 Boosting Fast AT with Learnable Adversarial Initialization

3.1 Introduction

Despite the progress that Deep Neural Networks have made in various fields, such as image recognition and speech recognition, they are still vulnerable to attacks [65]-[68]. For instance, by adding imperceptible perturbations to the existing examples, they can easily fool well-trained DNNs. Due to the vulnerability of Deep Learning systems to attacks, it has been known that their real-world applications are also susceptible to these types of threats [11], [28], [69]-[72]. This issue has been identified as one of the main challenges that the industry needs to address in order to advance the model robustness against these types of threats. One of the most effective ways to address this issue is by implementing an adversarial training method [30], [73]-[81]. An effective framework for addressing this issue is the minimax problem known as AT, which is composed of two main components: the inner maximization and the outer minimization [30], [83]. The inner maximization ensures that the model is resilient against attacks while the outer minimization ensures that the loss is minimized [84]. But the generation of the adversarial examples is an NP-hard problem [85], [86]. Thus, to produce adversarial instances, AT techniques always use the model's gradient information [11], [30].

AT can be divided into two classes based on the number of steps used to generate adversarial examples: Multi-step AT [30], [44], [87], [88] and Fast AT [89]-[93]. Multi-step AT uses multi-step adversarial attack methods like projected gradient descent (PGD) [30] to achieve comprehensive robustness against a variety of attack methods. However, doing many forward and backward propagation calculations in order to generate adversarial examples has a large computation cost. Fast AT approaches are proposed to improve training efficiency by requiring only one gradient calculation and using a fast gradient sign algorithm (FGSM) [11]. Although they can save a lot of time and computational cost, their robustness is lacking when compared to other state-of-the-art Multi-step AT approaches. As a result, numerous studies have looked into how to improve the robustness of Fast AT. Among them, some studies [91], [92] concentrate on the initialization problem, as it has been demonstrated that adopting a random initialization in Fast AT improves robustness significantly. However, the many random initialization styles used in current Fast AT methods are typically sample-agnostic, limiting future robustness enhancements.

To address this problem, we present FGSM-SDI [111], a sample-dependent adversarial initialization for boosting FGSM-based Fast AT. The sample-dependent initialization is calculated by a generative network conditioned on not only a benign image but also its signed gradient from the target network. The benign image provides position information in the loss landscape, while the signed gradient provides a rough direction of increasing the loss. The initialization is then

exploited by FGSM to generate a final adversarial example for training. The pipeline of adversarial example generation of the FGSM-SDI is illustrated in figure 24. The generative network and the target network are jointly learned under a minimax optimization framework, with the generative network attempting to create an effective initialization for FGSM to fool the target network, and the target network improving its robustness against such adversarial examples through adversarial training. The generative network dynamically optimizes a more effective initialization for FGSM, strengthening the robustness of Fast AT, in addition to an increasingly robust target network.

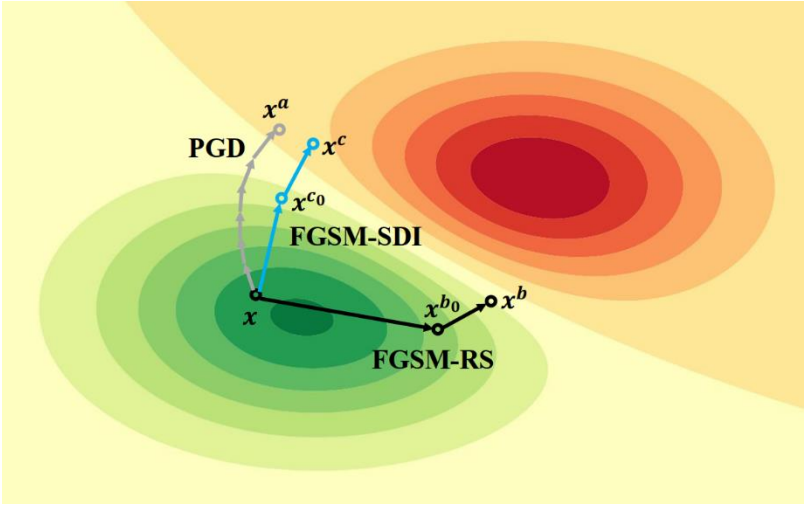

Figure 23 – Adversarial example generation process

Adversarial example generation process of PGD-AT [30], FGSM-RS [90], and the FGSM-SDI in the loss landscape of binary classification. Background is the contour of cross entropy. The redder the color, the lower the loss. PGD-AT is a multi-step AT method that computes gradients with respect to the input at each step. FGSM-RS uses a random sample-agnostic initialization followed by FGSM, requiring the computation of gradient only once. But the FGSM-SDI uses a sample-dependent learnable initialization followed by FGSM [111].

Fig. 23 presents the differences of a typical AT method (PGD-AT [30]), a fast AT method (FGSM-RS [90]), and the FGSM-SDI in generating adversarial examples. PGD-AT is a sophisticated Multi-step AT approach that achieves good robustness but takes a long time to calculate gradient at several levels. The gradient is calculated only once by FGSM-RS at x^{b0}, where the benign image x is moved with a random sample-agnostic beginning perturbation. The FGSM-SDI, on the other hand, creates a gradient at x^{c0} from which the innocuous image x is relocated using an optimal sample-dependent initial perturbation. The suggested learnable initialization is based on a benign image with a signed gradient that is more informative than a random image. With a significantly more efficient training method, the FGSM-SDI can attain equivalent robustness to the PGD-AT. The FGSM-SDI surpasses current Fast AT approaches (e.g., FGSM-RS) in terms of resilience by a wide margin, however with a modest performance penalty due to

the additional generating network. It's worth noting that adding another generative network is both light and appropriate. (See the results in section 3.4.5.)

3.2 Related Works

The relevant study on attack methods is initially introduced in this section. The associated study on defense methods, particularly adversarial training variations, is then presented. The focus of this research is on the image classification challenge, where adversarial instances can deceive a well-trained image classification model into making a high-confidence erroneous prediction. Given a clean image x with the corresponding true label y and a well-trained image classifier $f(\cdot)$, the attack methods are used to generate the adversarial example x_{adv}, to deceive the classifier into outputting an erroneous prediction, *i.e.*, $f(x_{adv}) \neq f(x) = y$, where the distance function satisfies $L_p(x_{adv}, x) \leq \epsilon$, where ϵ represents the maximum perturbation strength and L_p represents the distance between the adversarial image x_{adv} and the clean image x under the L_p distance metric, where $p \in \{1, 2, \infty\}$. L_∞ is a commonly used distance metric in the recent researches of attack methods, which is adopted in this work too.

3.2.1 Attack Methods

Szegedy et al. [11] identify the presence of adversarial examples for DNNs and construct adversarial examples using a box-constrained L-BFGS approach. To produce adversarial examples, Goodfellow et al. [28] suggest the fast gradient sign method (FGSM). To generate adversarial instances, FGSM merely calculates the gradient of the loss function once and then adds it to clean photos. To produce adversarial examples, Madry et al. [30] offer Projected Gradient Descent (PGD). To generate adversarial instances, PGD iterates several times to perform a gradient descent step in the loss function. DeepFool is a method proposed by Moosavi-Dezfooli et al. [33] for efficiently generating adversarial examples for tricking deep networks. Then, using an iterative linearization of the classifier, DeepFool generates minimum adversarial perturbations that can deceive the deep neural networks. Carlini et al. [51] suggest C&W, a more powerful attack approach. To generate adversarial perturbations, C&W introduces auxiliary variables. Then, to develop transferable adversarial examples, a number of iterative attack methods based on FGSM [94]-[98] were proposed. These attack approaches are aimed at increasing the adversarial transferability of adversarial instances, which means that adversarial examples created by one model can still be used against another. Croce F et al. [99] recently proposed two parameter-free attack approaches to overcome the difficulty caused by the suboptimal step size and objective

function, namely auto PGD with cross-entropy (APGD-CE) and auto PGD with the difference of logits ratio (APGD-DLR). They also combine the suggested attack methods with two existing attack methods, namely FAB [100] and Square Attack [101], to create the AutoAttack ensemble (AA). AA has also outperformed the competition when it comes to testing model resilience against adversarial examples.

3.2.2 Adversarial Training Methods

Under rigorous examinations, adversarial training (AT) versions have been largely acknowledged to improve adversarial robustness. They can be expressed as a minimax optimization problem, in which the inner maximizing maximizes the classification loss in order to produce adversarial instances, while the outer minimization minimizes the loss of generated adversarial examples in order to train robust model parameters. Given a target network $f(\cdot;\, w)$ with parameters w, a data distribution D including the benign sample x and its corresponding label y, a loss function $L(f(x;\, w);\, y)$, and a threat bound Δ, the objective function of AT can be presented as:

$$\min_{w} E_{(x,y)} \sim D \left[\max_{\delta \in \Delta} L(f(x+\delta; w), y) \right] \tag{20}$$

where the threat bound can be defined as $\Delta = \{\delta: ||\delta|| \leq \epsilon\}$ with the maximal perturbation intensity ϵ. The core of the adversarial training is how to find a better adversarial perturbation δ. To create an adversarial perturbation δ, most adversarial training methods use a multi-step adversarial approach, such as numerous steps of projected gradient descent (PGD) [30]. It can be presented as:

$$\delta_{t+1} = \prod_{[-\epsilon,\epsilon]^d} [\delta_t + \alpha \text{sign}(\nabla_x L(f(x+\delta_t; w), y))] \tag{21}$$

where $\prod_{[-\epsilon,\epsilon]^d}$ represents the projection to $[-\epsilon, \epsilon]^d$ and δ_{t+1} represents the adversarial perturbation after $t+1$ iteration steps. More iterations, in general, can improve adversarial training resilience by creating stronger adversarial cases. The prime PGD-based adversarial training framework is proposed in [30]. A larger number of PGD-based AT approaches are presented using this framework, with an early stopping variant [87] standing out. Algorithm 1 summarizes the PGD-AT variations algorithm.

Fast Adversarial Training. Fast adversarial training variants are proposed recently by adopting the one-step fast gradient sign method (FGSM) [28], which are also dubbed FGSM-based AT. It can be defined as:

$$\delta^* = \epsilon \text{sign}(\nabla_x L(f(x; w), y)) \tag{22}$$

where ϵ represents the maximal perturbation strength. Although it speeds up adversarial training, it substantially compromises the model's robustness, making it incapable of defending against the

PGD assault. FGSM-based AT suffers from a severe overfitting problem, in which the target model loses the resilience accuracy of adversarial cases created by PGD (on the training data) during training. To alleviate the catastrophic overfitting, Wong et al. [91] propose using FGSM-based AT coupled with a random initialization. It can match the premier PGD-based AT in terms of performance [30]. Specifically, they perform FGSM with a random initialization $\eta \in \mathrm{U}(-\epsilon, \epsilon)$, where U represents a uniform distribution, which can be called FGSM-RS. It can be defined as:

$$\delta^* = \prod_{[-\epsilon,\epsilon]^d}[\eta + \alpha \mathrm{sign}(\nabla_x L(f(x+\eta; w), y))] \quad (23)$$

where α represents the step size, which is set to 1.25ϵ in [91]. This study shows that when combined with a good initialization, FGSM-based AT can perform as well as PGD-AT [30]. More notably, the FGSM-RS has a lower computational cost than the PGD-AT. Algorithm 2 summarizes the FGSM-RS algorithm. In addition, numerous studies are recommended to improve model robustness in the wake of FGSM-RS. Andriushchenko et al. [92] find that utilizing a random initialization does not completely eliminate the catastrophic overfitting problem, and they propose the FGSM-GA regularization method to improve the performance of FGSM-based AT. Furthermore, Kim et al. [93] suggest FGSM-RS-based single-step adversarial training, also known as FGSM-CKPT. FGSM-CKPT calculates an appropriate perturbation magnitude for each image, preventing catastrophic overfitting.

Algorithm 1 PGD-AT [111]

Require: The epoch N, the maximal perturbation ϵ, the step size α, the attack iteration T, the dataset D including the benign sample x and the corresponding label y, the dataset size M and the network $f(\cdot, w)$ with parameters w.

1: **for** $n = 1, \ldots, N$ **do**
2; **for** $i = 1, \ldots, M$ **do**
3: $\delta_1 = 0$
4: **for** $t = 1, \ldots, T$ **do**
5: $\delta_{t+1} = \prod_{[-\epsilon,\epsilon]^d}[\delta_t + \alpha \mathrm{sign}(\nabla_x L(f(x+\delta_t; w), y))]$
6: **end for**
7: w $\leftarrow$ w $- \nabla_w L(f(x_i + \delta_t; w), y_i)$
8: **end for**
9: **end for**

Algorithm 2 FGSM-RS [111]

Require: The epoch N, the maximal perturbation ϵ, the step size α, the dataset D including the benign sample x and the corresponding label y, the dataset size M and the network $f(\cdot, w)$ with parameters w.

1: **for** $n = 1, \ldots, N$ **do**

2; **for** $i = 1, \ldots, M$ **do**

3: $\eta = \mathrm{U}(-\epsilon, \epsilon)$

4: $\delta = \prod_{[-\epsilon,\epsilon]^d}[\eta + \alpha \mathrm{sign}(\nabla_x L(f(x+\eta; w), y))]$

5: $\mathrm{w} \leftarrow \mathrm{w} - \nabla_w L(f(x_i + \delta; w), y_i)$

6: **end for**

7: **end for**

3.3 The Proposed Method

Using a random sample-agnostic initialization for Fast AT is common and expedient, but it hampers further model robustness enhancement. To address this problem, a sample-dependent adversarial initialization was proposed [111] to improve Fast AT's robustness while also addressing the catastrophic overfitting problem. Section 3.3.1 introduces the proposed method's pipeline, section 3.3.2 introduces the proposed generative network's architecture, and section 3.3.3 introduces the formulation.

3.3.1 Pipeline of the Proposed Method

Two networks, a generative network and a target network, are used in the proposed method. Instead of employing a random initialization, the former learns to build a dynamic sample-dependent adversarial initialization for FGSM to generate adversarial instances. To improve model robustness, the latter uses the generated adversarial cases for training. A benign image and its gradient information from the target network are sent to the generative network, which generates a sample-dependent initialization, as shown in figure 24. To generate adversarial instances, FGSM is applied to the input image along with the generated initialization. To strengthen the robustness against adversarial attacks, the target network is trained on adversarial samples.

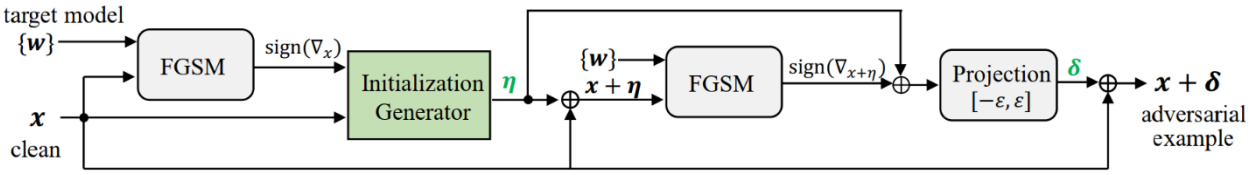

Figure 24 – Adversarial example generation of the proposed FGSM-SDI [111]

For the target network, we adopt the architecture of a typical image classification network, defined as $y = f(x; w)$, where x represents an input image, y represents the predicted label, and w represents the parameters of the target network.

Three layers make up the generative network. The generative network's detailed structure is provided in section 3.3.2. The benign image and its signed gradient are the generative network's inputs. The signed gradient can be determined using the following formula:

$$S_x = \text{sign}(\nabla_x L(f(x; w), y)) \tag{24}$$

where x is the input image and y is the ground-truth label. The initialization generation process can be defined as:

$$\eta_g = \epsilon g(\text{x}, s_x; \theta) \tag{25}$$

where $g(\cdot;\ \theta)$ represents the generative network with the parameters θ, and η_g represents the generated adversarial initialization. The output pixel value space of $g(\cdot;\ \theta)$ is $[-1;\ 1]$. ϵ is a scale factor that maps the value to the range of $[-\epsilon, \epsilon]$.

3.3.2 Architecture of the Generative Network

Figure 25 depicts the generative network's architecture [111]. The clean image is combined with gradient information from the target network to produce the generative network's input. A sample-dependent adversarial initialization is generated by the generative network. We use a three-layer generative network that is light on resources. Table I shows the specific settings for each layer. The first layer consists of one convolutional layer with 64 filters of size $3 \times 3 \times 6$ which is followed by a batch normalization layer [102]. The second layer is a ResBlock [103] with 64 filters of size $3 \times 3 \times 64$. And the third layer consists of one convolutional layer with 64 filters of size $3 \times 3 \times 3$ which is followed by a batch normalization layer. We adopt the ReLU [104] as an activation function.

The proposed generative network generates a sample-dependent initialization for the clean image based on itself and its gradient information. The proposed sample-dependent adversarial initialization is more informative than the random initialization. Experiments show that the proposed FGSM-SDI not only prevents catastrophic overfitting, but also narrows the gap between typical Fast AT methods and the multistep AT methods.

Table I – The architecture of the generative network [111]

Layer	Type	Input Channels	Output Channels	Stride	Padding	Filter Size
1st layer	Conv +BN+ ReLU	6	64	1	1	3 ×3
2nd layer	ResBlock	64	64	1	1	3 ×3
3rd layer	Conv +BN	64	3	1	1	3 ×3

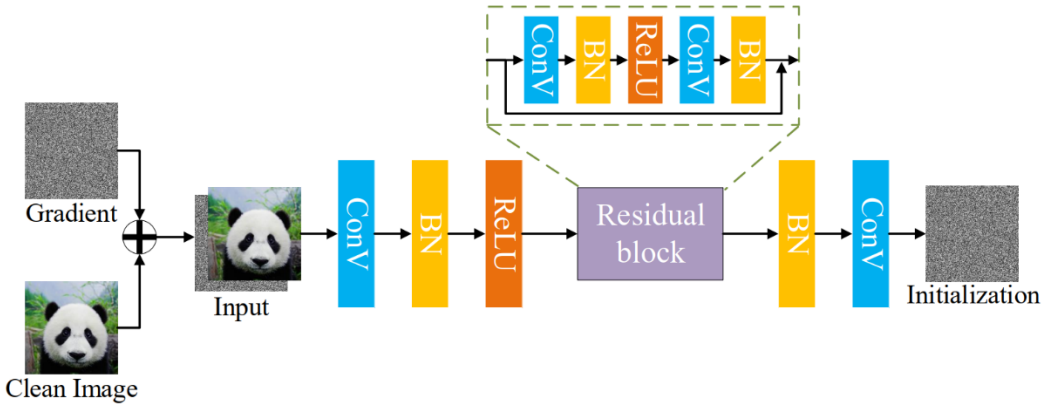

Figure 25 – The architecture of the lightweight generative network [111].

The clean image combined with its gradient information from the target network forms the input of the generative network. The generative network consists of two convolutional layers and one ResBlock, which outputs the adversarial initialization for the clean image.

3.3.3 Formulation of the Proposed Method

Similar to the adversarial perturbation of FGSM-RS [91] in Eq. 23, the perturbation i.e., the approximate solution of the inner maximization problem, can be represented as:

$$\delta_g = \delta_g(\theta) = \prod_{[-\epsilon,\epsilon]^d}[\eta_g + \alpha \text{sign}(\nabla_x L(f(x+\eta_g; w), y))] \quad (26)$$

where η_g is the adversarial initialization defined in Eq. 25, generated by the generative network. It's worth noting that the perturbation involves the generative network's parameters θvia the initialization. The initialization term η_g distinguishes the perturbation of FGSM-RS (Eq. 23) from the perturbation of FGSM-SDI (Eq. 26). The initialization of FGSM-RS is done at random, with no further information to guide it. Although it improves the diversity of adversarial cases and resilience, it suffers from the catastrophic overfitting problem, with robustness dropping dramatically in the late training stage (see figure 26). The initialization η_g, on the other hand, is a function of the benign picture x and its gradient information s_x, which gives some useful information about the direction of the initial perturbation. It not only solves the problem of catastrophic overfitting, but it also increases the resilience of current Fast AT approaches, making it similar to PGD-AT.

The objective function of jointly learning the generative network and the target network can be defined as follows using the specification of the suggested perturbation. The suggested

solution to the inner maximization problem incorporates the parameters of the generative network, starting with the objective function of conventional AT in Eq. 20. When fixing the parameters θ, the solution is approximated by η_g in Eq. 26. We can further maximize the loss by searching better parameters θ, i.e., $max_\theta\, L(f(x + \delta_g(\theta); w); y)$. Hence, the objective function of the joint optimization can be defined as:

$$\min_w \max_\theta E_{(x,y)\sim D}\, L(f(x + \delta_g(\theta); w); y) \tag{27}$$

The generative network and the target network play a game, as seen in Eq. 27. The former seeks to maximize the loss in order to generate an effective initialization for adversarial example generation, whereas the latter seeks to decrease the loss in order to update the parameters and improve model robustness against adversarial instances. More significantly, the generative network may produce initializations based on the target model's robustness during various training phases. This minimax problem can be solved by alternatively optimizing w and θ. Note that we update θ and w iteratively. We update θ every k times of updating w. And k is a hyper-parameter that needs to be tuned. The algorithm for solving this problem is shown in Algorithm 3.

Eq. 20 is the objective of standard and fast AT methods. Compared to Eq. 20, our formulation has the following differences. First, δ in Eq. 20 is a variable to optimize, while we replace it with the approximate solution $\delta g(\theta)$ of the inner maximization. $\delta g(\theta)$ is regarded as a function that involves the parameters of the generative network. Second, we apply an additional maximization to the parameters of the generative network to further maximize the loss, which forces the competition between the two networks.

Algorithm 3 FGSM-SDI [111]

Require: The epoch N, the maximal perturbation ϵ, the step size α, the dataset D including the benign sample x and the corresponding label y, the dataset size M and the network $f(\cdot, w)$ with parameters w, the generative network $g(\cdot;\ \theta)$ with parameters θ and the interval k.

for $n = 1, \ldots, N$ **do**
  for $i = 1, \ldots, M$ **do**
    $S_{x_i} = \text{sign}(\nabla_{x_i} L(f(x_i; w), y_i))$
    if $i \ mod \ k = 0$ **then**
      $\eta_g = \epsilon g(x_i, s_{x_i}; \theta)$
      $\delta = \prod_{[-\epsilon,\epsilon]^d}[\eta_g + \alpha \text{sign}(\nabla_x L(f(x_i + \eta_g; w), y))]$
      $\theta \leftarrow \theta + \nabla_\theta L(f(x_i + \delta; \theta), y_i)$
    end if
    $\eta_g = \epsilon g(x_i, s_{x_i}; \theta)$
    $\delta = \prod_{[-\epsilon,\epsilon]^d}[\eta_g + \alpha \text{sign}(\nabla_x L(f(x_i + \eta_g; w), y))]$
    $\text{w} \leftarrow \text{w} - \nabla_{\text{w}} L(f(x_i + \delta; \text{w}), y_i)$
  end for
end for

3.4 Experiments

Extensive experiments on four benchmark databases are done to evaluate the effectiveness of the FGSM-SDI, including the selection of hyper-parameters in the proposed FGSM-SDI, and comparisons with state-of-the-art rapid AT approaches.

3.4.1 Experimental Settings

Datasets. In the original paper [111] four benchmark databases were adopted to conduct experiments, i.e., CIFAR10 [105], CIFAR100 [105], Tiny ImageNet [106] and ImageNet [106]; the experiments on CIFAR10 and CIFAR100 are reimplemented and the results are reported next to original results; the python codes are pushed in the GitHub repository [112]. These databases are the most widely used databases to evaluate adversarial robustness. Both CIFAR10 and CIFAR100 consist of 50,000 training images and 10,000 test images. The image size is $32 \times 32 \times 3$. CIFAR10 covers 10 classes while CIFAR100 covers 100 classes. Tiny ImageNet is a subset of the ImageNet database [106], which contains 200 classes. Each class has 600 images. The image size is $64 \times 64 \times 3$. As for the ImageNet database, it contains 1000 classes and, in

this work, images were resized to $224 \times 224 \times 3$. Following the setting of [88], as Tiny ImageNet and ImageNet have no labels for the test dataset, evaluations on the validation dataset were conducted in this work.

Experimental Setups. On CIFAR10, ResNet18 [19] and WideResNet34-10 [20] are used as the target network. On CIFAR100, ResNet18 [19] is used as the target network. On Tiny ImageNet, PreActResNet18 [107] is used as the target network. On ImageNet, ResNet50 [19] is used as the target network. As for CIFAR10, CIFAR100, and Tiny ImageNet, following the settings of [87], [108], the target network is trained for 110 epochs. The learning rate decays with a factor of 0.1 at the 100-th and 105-th epoch. The SGD [109] momentum optimizer was adopted with an initial learning rate of 0.1 and the weight decay of 5e-4. As for ImageNet, following the previous studies [90], [91], the target network is trained for 90 epochs. The learning rate decays with a factor of 0.1 at the 30-th and 60-th epoch. The SGD momentum optimizer is used with an initial learning rate of 0.1 and the weight decay of 5e-4. Note that the results of the checkpoint with the best accuracy under the attack of PGD-10 are reported as well as the results of the last checkpoint. For adversarial robustness evaluation, several adversarial attack methods were adopted to attack the target network, including FGSM [28], PGD [30], C&W [51], and AA [99]. And the maximum perturbation strength ϵ was set to 8/255 for all attack methods. Moreover, the PGD attack with 10, 20, and 50 iterations, i.e., PGD-10, PGD-20, and PGD-50 were conducted. And the comparison experiments were conducted using a cyclic learning rate strategy [110].

3.4.2 Hyper-parameter Selection

The proposed FGSM-SDI has only one hyper-parameter, namely the interval k. w is updated every k times it is updated. This hyper-parameter has an impact on both model training efficiency and model robustness when faced with adversarial examples. A hyperparameter selection experiment on CIFAR10 is carried out to find the best hyper-parameter. Table II displays the results. With the increase of parameter k, the calculation time of the proposed FGSM-SDI decreases. That is, the smaller the k value, the more frequently the generative network is updated, and the more calculation time the generative network requires for training. Surprisingly, performance against adversarial examples improves as the parameter k is increased when $k = 1 \sim 20$. When $k = 20 \sim 40$, the performance against adversarial examples slightly drops with the increase of parameter k. When $k = 20$, the proposed FGSM-SDI achieves the best adversarial robustness in all adversarial attack scenarios. Considering adversarial training efficiency, k is set to 20.

Table II – The hyperparameter selection on the CIFAR10 database. Numbers in table represent percentage. Number in bold indicate the best result [111]

	Clean	PGD-10	PGD-20	PGD-50	C&W	APGD	AA	Time(min)
k=1	**85.05**	52.95	51.52	51.16	49.80	50.64	47.06	134
k=2	85.14	52.63	51.26	50.9	49.75	50.42	47.36	106
k=10	84.80	53.33	52.20	51.79	50.47	51.52	47.99	84
k=20	84.86	**53.73**	**52.54**	**52.18**	**51.00**	**51.84**	**48.52**	83
k=30	84.45	53.30	52.09	51.77	50.09	51.05	47.03	82
k=40	84.73	52.96	51.71	51.34	49.62	51.01	47.24	**81**

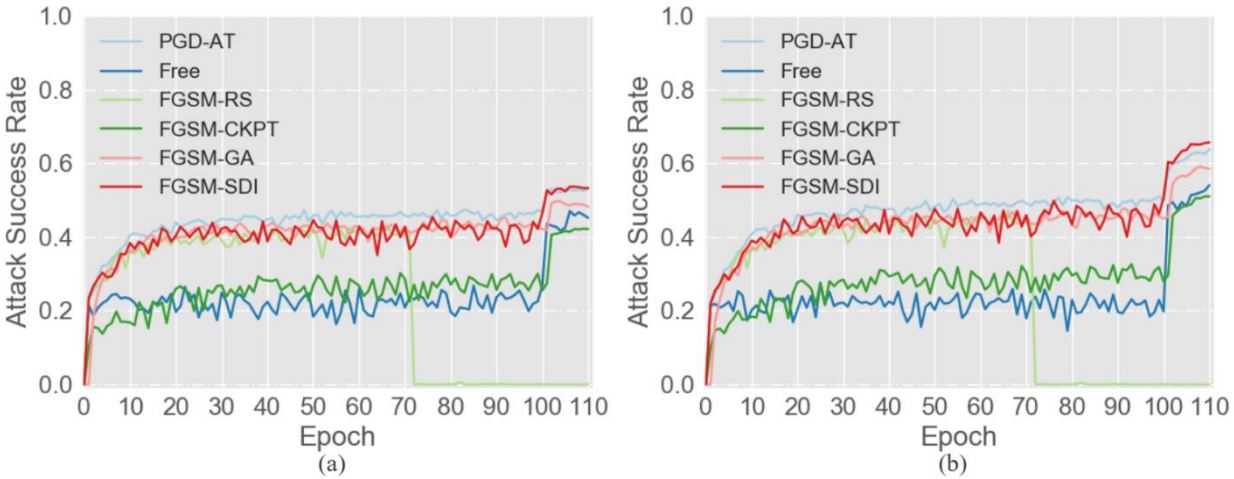

Figure 26

The PGD-10 accuracy of AT methods on the CIFAR10 database in the training phase. (a) The PGD-10 accuracy on the training dataset. (b) The PGD-10 accuracy on the test dataset.

3.4.3 Relieving Catastrophic Overfitting

Catastrophic overfitting, which refers to the situation when the accuracy on adversarial cases abruptly falls to 0.00, is one of the difficult difficulties that mostly affects the model resilience of the fast AT approaches. We track the accuracy of adversarial examples created on training and test data in the training phase to examine catastrophic overfitting. PGD-10 generates adversarial scenarios. Figure 26 depicts the training and test curves for the PGD-10 attack. After about the 70th epoch, the accuracy of FGSM-RS quickly drops to zero. Other fast AT approaches address the catastrophic overfitting problem, but their performance is far from ideal, i.e., there is a significant gap between fast AT methods and the advanced multi-step PGD-AT [87]. For example, on training and test data, FGSM-GA achieves accuracy of 58.46 percent and 48.17 percent, respectively, while PGD-AT achieves substantially greater accuracy of 63.71 percent and 53.33 percent. In contrast to PGD-AT, the FGSM-SDI can not only avoid catastrophic overfitting but also achieve comparable robustness. It obtains accuracy of 64.14 percent on training data and 52.81 percent on test data, respectively. It's worth noting that the FGSM-SDI takes around a third

of the time to compute as PGD-AT and even less time than FGSM-GA. The following section will go through further specifics.

Table III – Comparisons of clean and robust accuracy (%) and Training time (minute) with ResNet18 on the CIFAR10 database. Number in bold indicates the best of the Fast AT methods (Original [111])

Target Network	Method		Clean	PGD-10	PGD-20	PGD-50	C&W	APGD	AA	Time(min)
Resnet18	PGD-AT	Best	82.32	53.76	52.83	52.6	51.08	52.29	48.68	265
		Last	82.65	53.39	52.52	52.27	51.28	51.90	48.93	
Resnet18	FGSM-RS	Best	73.81	42.31	41.55	41.26	39.84	41.02	37.07	51
		Last	83.82	00.09	00.04	00.02	0.00	0.00	0.00	
	FGSM-CKPT	Best	**90.29**	41.96	39.84	39.15	41.13	38.45	37.15	76
		Last	**90.29**	41.96	39.84	39.15	41.13	38.45	37.15	
	FGSM-GA	Best	83.96	49.23	47.57	46.89	47.46	45.86	43.45	178
		Last	84.43	48.67	46.66	46.08	46.75	45.05	42.63	
	Free-AT(m=8)	Best	80.38	47.1	45.85	45.62	44.42	42.18	42.17	215
		Last	80.75	45.82	44.82	44.48	43.73	45.22	41.17	
	FGSM-SDI	Best	84.86	**53.73**	**52.54**	**52.18**	**51.00**	**51.84**	**48.50**	83
		Last	85.25	**53.18**	**52.05**	**51.79**	**50.29**	**51.30**	**47.91**	

Table IV – Comparisons of clean and robust accuracy (%) and Training time (minute) with ResNet18 on the CIFAR10 database. Number in bold indicates the best of the Fast AT methods (Reimplementation)

Target Network	Method	Clean	PGD-10	PGD-20	PGD-50	C&W	APGD	AA	Time(min)
Resnet18	FGSM-RS	70.95	37.23	38.67	38.22	36.35	38.21	32.00	128
	FGSM-CKPT	**85.65**	40.49	37.41	36.50	40.79	37.42	36.19	190
	FGSM-SDI	83.77	**52.67**	**50.39**	**51.78**	**49.85**	**50.32**	**46.20**	210

Table V – Comparisons of clean and robust accuracy (%) and Training time (minute) with WideResNet34-10 on the CIFAR10 database. Number in bold indicates the best of the Fast AT methods [111]

Target Network	Method	Clean	PGD-10	PGD-20	PGD-50	C&W	APGD	AA	Time(min)
WideResNet34-10	PGD-AT	85.17	56.1	55.07	54.87	53.84	54.15	51.67	1914
WideResNet34-10	FGSM-RS	74.29	41.24	40.21	39.98	39.27	39.79	36.40	348
	FGSM-CKPT	**91.84**	44.7	42.72	42.22	42.25	41.69	40.46	470
	FGSM-GA	81.8	48.2	47.97	46.6	46.87	46.27	45.19	1218
	Free-AT(m=8)	81.83	49.07	48.17	47.83	47.25	47.40	44.77	1422
	FGSM-SDI	86.4	**55.89**	**54.95**	**54.6**	**53.68**	**54.21**	**51.17**	533

3.4.4 Comparisons with State-of-the-art Methods

The proposed FGSM-SDI is compared with several state-of-the-art fast AT methods (i.e., Free [90], FGSM-RS [91], FGSM-GA [101], and FGSM-CKPT [93]) and an advanced multi-step AT method (i.e., PGD-AT [87]) which adopts 10 steps to generate adversarial examples on four benchmark databases. The settings reported in their original works to train these AT methods were adopted. Note that to ensure fairness of comparison, the number of epochs were not divided by m such that the total number of epochs remains the same with the other fast AT methods.

Results on CIFAR10. Resnet18 was adopted as the target network to conduct the comparison experiment with other defense methods on CIFAR10. The results are reported in Table III. Compared with the fast AT methods, the proposed method achieves the best performance under all attack scenarios and comparable robustness to the advanced PGD-AT [87]. The previous most fast AT methods are only as effective as the prime PGD-AT [30], i.e., they achieve the performance of about 45% under the PGD-10 attack. The performance is far from that of the advanced PGD-AT [87] which uses an early stopping trick to achieve above 50% accuracy. Unlike them, the proposed method can achieve more than 53% under the PGD-10 attack on the best and last checkpoint. As for the strong attack (AA), the previous most powerful fast AT method (FGSM-GA) achieves the performance of about 43%, but the proposed FGSM-SDI achieves about 48% robust accuracy which is same with PGDAT. In terms of training efficiency, the proposed method's training time is less than FGSM-GA, Free, and PGD-AT. Specifically, the training time of the FGSM-SDI is about 1.6 times the training time of FGSM-RS [91]. Though FGSM-RS and FGSM-CKPT are more efficient than the FGSM-SDI method, their performance is always the worst among all the fast AT methods. FGSM-RS is the fastest method that uses a random initialization. The FGSM-SDI method improves the initialization to boost the robustness by introducing the generative network, resulting in the sacrifice of efficiency for an additional gradient calculation. Therefore, FGSM-SDI method can be viewed as a method that balances the robustness and the training efficiency. Considering the Table IV which is the presentation of reimplementation of the experiment, except for the time differences, which is due to difference in hardware used for reimplementation, we witness that FGSM-SDI outperforms the FGSM-CKPT and FGSM-RS in case of robust accuracy.

Moreover, WideResNet34-10 was adopted which is a large architecture model to conduct a comparison experiment. The results are shown in Table V. We observe a similar phenomenon as the Resnet18 trained on CIFAR10. The FGSM-SDI achieves the best performance under all attack scenarios compared with previous fast AT methods. Besides, compared with the advanced PGD-AT, the proposed FGSM-SDI costs less time and achieves comparable robustness to it. Specifically, PGD-AT achieves the performance of about 51% and 49% against AA attack on the best and last checkpoints. The FGSM-SDI also achieves the performance of about 51% and 49%. But PGD-AT takes about 1914 minutes for training, while the FGSM-SDI only takes about 533 minutes for training.

Comparative experiments also were conducted using a cyclic learning rate strategy [110] on CIFAR10. Following [92], [93], the maximal learning rate of FGSM-GA [92] and FGSM-CKPT [93] were set to 0.3. Following [91], the maximal learning rate of FGSM-RS [91], Free [90], PGD-AT [87], and the proposed method were set to 0.2. All the models for 30 epochs were

trained. Other training and evaluation settings remain unchanged. The results are shown in Table VI. Compared with the other fast AT methods, the proposed FGSM-SDI achieves the best adversarial robustness and comparable robustness to the advanced PGD-AT [87]. Using a cyclic learning rate strategy can prevent catastrophic overfitting for the fast AT methods, i.e., the performance of the last checkpoint is almost the same as that of the best checkpoint. But their adversarial robustness is still far from that of the advanced PGD-AT [87]. Differently, the FGSM-SDI can achieve comparable robustness to PGDAT [87]. For example, FGSM-RS [91] achieves about 42% accuracy under AA attack, while FGSM-SDI method achieves about 46%. In terms of efficiency, this method outperforms Free and FGSM-GA and is much better than PGD-AT. FGSM-RS and FGSM-CKPT use a random initialization which promotes efficiency at a sacrifice in robustness. The proposed method improves the initialization with a generative network, which greatly boosts the adversarial robustness with the slight expense of time cost.

Table VI – Comparisons of clean and robust accuracy (%) and Training time (minute) on the CIFAR10 database. Number in bold indicates the best of the Fast AT methods. All models are trained using a cyclic learning rate strategy [111]

Target Network	Method		Clean	PGD-10	PGD-20	PGD-50	CW	APGD	AA	Time(min)
Resnet18	PGD-AT	Best	80.12	51.59	50.83	50.7	49.04	50.34	46.83	71
		Last	80.12	51.59	50.83	50.7	49.04	50.34	46.83	
Resnet18	FGSM-RS	Best	83.75	48.05	46.47	46.11	46.21	45.75	42.92	15
		Last	83.75	48.05	46.47	46.11	46.21	45.75	42.92	
	FGSM-CKPT	Best	**89.08**	40.47	38.2	37.69	39.87	37.16	35.81	21
		Last	**89.08**	40.47	38.2	37.69	39.87	37.16	35.81	
	FGSM-GA	Best	80.83	48.76	47.83	47.54	47.14	47.27	44.06	49
		Last	80.83	48.76	47.83	47.54	47.14	47.27	44.06	
	Free-AT(m=8)	Best	75.22	44.67	43.97	43.72	42.48	43.55	40.30	59
		Last	75.22	44.67	43.97	43.72	42.48	43.55	40.30	
	FGSM-SDI	Best	82.08	**51.63**	**50.65**	**50.33**	**48.57**	**49.98**	**46.21**	23
		Last	82.08	**51.63**	**50.65**	**50.33**	**48.57**	**49.98**	**46.21**	

Table VII – Comparisons of clean and robust accuracy (%) and Training time (minute) with ResNet18 on the CIFAR100 database. Number in bold indicates the best of the Fast AT methods [111]

Target Network	Method		Clean	PGD-10	PGD-20	PGD-50	C&W	APGD	AA	Time(min)
Resnet18	PGD-AT	Best	57.52	29.6	28.99	28.87	28.85	28.60	25.48	284
		Last	57.5	29.54	29.00	28.90	27.6	28.70	25.48	
Resnet18	FGSM-RS	Best	49.85	22.47	22.01	21.82	20.55	21.62	18.29	70
		Last	60.55	00.45	00.25	00.19	00.25	0.00	0.00	
	FGSM-CKPT	Best	**60.93**	16.58	15.47	15.19	16.4	14.63	14.17	96
		Last	**60.93**	16.69	15.61	15.24	16.6	14.87	14.34	
	FGSM-GA	Best	54.35	22.93	22.36	22.2	21.2	21.88	18.88	187
		Last	55.1	20.04	19.13	18.84	18.96	18.46	16.45	
	Free-AT(m=8)	Best	52.49	24.07	23.52	23.36	21.66	23.07	19.47	229
		Last	52.63	22.86	22.32	22.16	20.68	21.90	18.57	
	FGSM-SDI	Best	60.67	**31.5**	**30.89**	**30.6**	**27.15**	**30.26**	**25.23**	99
		Last	60.82	**30.87**	**30.34**	**30.08**	**27.3**	**29.94**	**25.19**	

Results on CIFAR100. The results are shown in Table VII. The CIFAR100 database covers more classes than the CIFAR10, which makes the target network harder to obtain robustness. We can observe similar phenomenon as on CIFAR10. In detail, compared with the other fast AT methods, the FGSM-SDI achieves the best adversarial robustness under all adversarial attack scenarios. For example, the previous fast AT methods achieve the performance of about 20% under the PGD-50 attack which is far from that of the advanced PGD-AT [87] which achieves about 28% accuracy. While the proposed FGSM-SDI achieves the performance of about 30% under the PGD-50 attack.

Surprisingly, the FGSM-SDI method can even outperform PGD-AT under the attacks of PGD-10, PGD-20, PGD-50, and APGD. This method also achieves comparable robustness to the advanced PGD-AT under the strong attack methods (C&W and AA). And the clean accuracy is also about 3% higher than PGD-AT [87]. This indicates the potential of the FGSM-SDI method in boosting robustness. In terms of training efficiency, similar results are observed on CIFAR10. The FGSM-SDI can be 3 times faster than the advanced PGD-AT [87]. Although the FGSM-SDI costs a little more time than FGSM-RS, it not only relieves the catastrophic overfitting problem but also achieves comparable robustness to the advanced PGD-AT.

Table VIII – Comparisons of clean and robust accuracy (%) and Training time (minute) with PreActResNet18 on the Tiny ImageNet database. Number in bold indicates the best of the Fast AT methods [111]

Target Network	Method		Clean	PGD-10	PGD-20	PGD-50	CW	APGD	AA	Time(min)
PreActResNet18	PGD-AT	Best	43.6	20.2	19.9	19.86	17.5	19.64	16.00	1833
		Last	45.28	16.12	15.6	15.4	14.28	15.22	12.84	
PreActResNet18	FGSM-RS	Best	44.98	17.72	17.46	17.36	15.84	17.22	14.08	339
		Last	45.18	0.00	0.00	0.00	0.00	0.00	0.00	
	FGSM-CKPT	Best	**49.98**	9.20	9.20	8.68	9.24	8.50	8.10	464
		Last	**49.98**	9.20	9.20	8.68	9.24	8.50	8.10	
	FGSM-GA	Best	34.04	5.58	5.28	5.1	4.92	4.74	4.34	1054
		Last	34.04	5.58	5.28	5.1	4.92	4.74	4.34	
	Free-AT(m=8)	Best	38.9	11.62	11.24	11.02	11.00	10.88	9.28	1375
		Last	40.06	8.84	8.32	8.2	8.08	7.94	7.34	
	FGSM-SDI	Best	46.46	**23.22**	**22.84**	**22.76**	**18.54**	**22.56**	**17.00**	565
		Last	47.64	**19.84**	**19.36**	**19.16**	**16.02**	**19.08**	**14.10**	

Results on Tiny ImageNet. The results are shown in Table VIII. Tiny ImageNet is a larger database compared to CIFAR10 and CIFAR100. Performing AT on Tiny ImageNet requires more computational cost. It takes about 1,833 minutes for PGD-AT to conduct training. But, the FGSM-SDI only takes 565 minutes and achieves comparable or even better robustness than PGD-AT. Similar to the results on CIFAR100 and CIFAR10, compared with previous fast AT methods, the FGSM-SDI achieves the best performance under all attack scenarios. Moreover, compared with the advanced PGD-AT, the FGSMSDI achieves better performance under all attack scenarios even

the strong attack (AA). Specifically, PGD-AT achieves the performance of about 16% and 13% accuracy under AA attack on the best and last checkpoints, while the FGSM-SDI achieves the performance of about 17% and 14% accuracy. Moreover, FGSM-SDI achieves higher clean accuracy compared with PGD-AT. Specifically, the FGSM-SDI clean accuracy is also about 3% higher than PGD-AT. The efficiency comparison is similar to that on CIFAR10 and CIFAR100.

Table IX – Comparisons of clean and robust accuracy (%) and Training time (hour) with ResNet50 on the ImageNet database. Number in bold indicates the best of the Fast AT methods [111]

ImageNet	Epsilon	Clean	PGD-10	PGD-50	Time(hour)
PGD-AT	ϵ =2	64.81	47.99	47.98	211.2
	ϵ =4	59.19	35.87	35.41	
	ϵ =8	49.52	26.19	21.17	
Free-AT(m=4)	ϵ =2	**68.37**	48.31	48.28	127.7
	ϵ =4	63.42	33.22	33.08	
	ϵ =8	52.09	19.46	12.92	
FGSM-RS	ϵ =2	67.65	48.78	48.67	44.5
	ϵ =4	**63.65**	35.01	32.66	
	ϵ =8	**53.89**	0.00	0.00	
FGSM-SDI	ϵ =2	66.01	**49.51**	**49.35**	66.8
	ϵ =4	59.62	**37.5**	**36.63**	
	ϵ =8	48.51	**26.64**	**21.61**	

Results on ImageNet. Following [90], [91], Resnet50 was adopted to conduct AT on ImageNet under the maximum perturbation strength $\epsilon = 2$, $\epsilon = 4$, and $\epsilon = 8$. The results are reported in Table IX. When $\epsilon = 2$, all methods achieve roughly the same robustness against adversarial examples. But as the maximal perturbation strength becomes larger, PGD-AT and our FGSMSDI achieves better robustness performance. Especially, when $\epsilon = 8$, the FGSM-RS cannot defend against the PGD-based attacks. But the FGSM-SDI still achieves the performance of about 26% and 21% under the PGD-10 and PGD-50 attacks and achieves comparable robustness to PGD-AT. In terms of training efficiency, similar phenomena are observed on other databases, the FGSM-SDI can be 3 times faster than the advanced PGD-AT.

3.4.5 Performance Analysis

To explore how the FGSM-SDI initialization affects the generation of adversarial examples, a Renet18 on CIFAR10 was trained and the attack success rate of adversarial examples that successfully attack the target model during the training process were calculated. The comparisons with FGSM-RS, PGD2-AT, and PGD-AT are shown in figure 27. From the 0-th to 70-th epoch, the attack success rates of the successful adversarial examples of the FGSM-RS, PGD2-AT, and FGSM-SDI are roughly the same. However, after the 70-th epoch, the attack success rate of FGSM-RS drops sharply. At that time the trained model using FGSM-RS falls into

the catastrophic overfitting problem that the trained model cannot defend against the adversarial instances generated by PGD-based attack methods during the training process. While the adversarial instances generated by the other three methods always keep adversarial to the trained model. They do not meet the catastrophic overfitting. This observation indicates that the catastrophic overfitting is associated with the adversarial example quality in the training process. Moreover, the attack success rate of adversarial in the training process is also related to the robust performance. The PGD-AT that adopts the adversarial examples with the highest attack success rate has the best robust performance. Compared with PGD2-AT, the FGSM-SDI that has a higher attack success rate achieves a better robust performance.

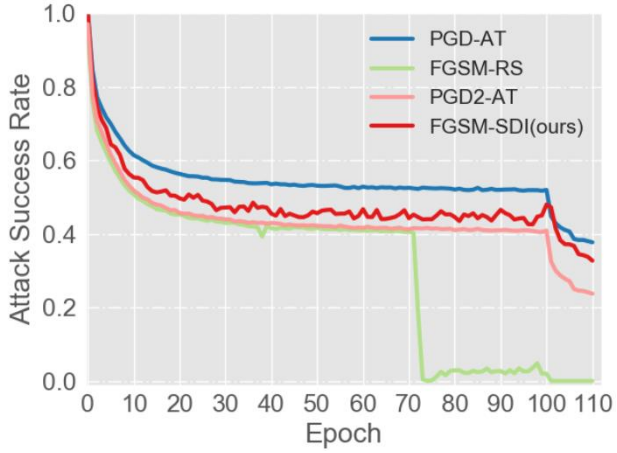

Figure 27 – Attack success rate of FGSM-RS, PGD-AT, PGD2-AT and FGSM-SDI during the training process [111].

4 Smooth Adversarial Training

4.1 Introduction

Gradient masking [156, 115], which commonly incorporates non-differentiable processes (e.g., discretization [119, 162]) to conceal gradients, is another prominent route for boosting robustness against adversarial attacks. As a result of the degenerated gradients, attackers are unable to optimize the targeted loss and so fail to defeat such defenses. However, if the gradient masking operation's differentiable approximation is utilized to generate adversarial examples, it will fail to provide robustness [115].

The tumultuous history of gradient masking-based defenses prompts us to reconsider the relationship between gradient quality and adversarial robustness, particularly in the setting of adversarial training, where gradients are used more frequently than in conventional training. Adversarial training necessitates gradient calculation not only for updating network parameters but also for producing training samples. We find that ReLU, a widely used activation function in most network designs, greatly worsens adversarial training due to its non-smooth nature, e.g., ReLU's gradient changes abruptly (from 0 to 1) when its input is close to zero, as presented in figure 28.

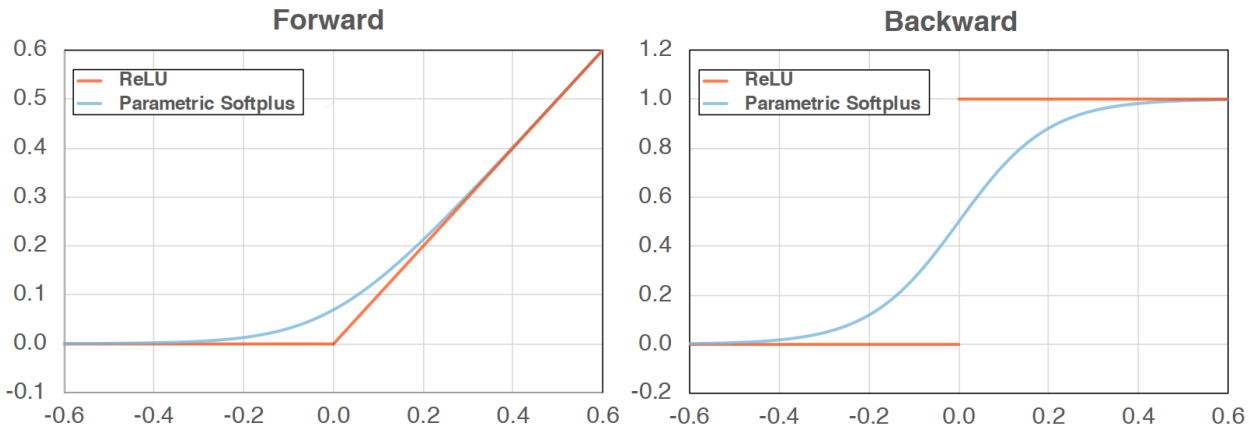

Figure 28 – The visualization of ReLU and Parametric Softplus. Left panel: the forward pass for ReLU (blue curve) and Parametric Softplus (red curve). Right panel: the first derivatives for ReLU (blue curve) and Parametric Softplus (red curve). Different from ReLU, Parametric Softplus is smooth with continuous derivatives [113].

In order to address the problem caused by ReLU, we look at smooth adversarial training (SAT), which enforces architectural smoothness by replacing ReLU with smooth approximations for better gradient quality in adversarial training (figure 28 shows Parametric Softplus, an example of smooth approximations for ReLU). SAT is able to feed the networks with harsher adversarial training samples and compute better gradient updates for network optimization thanks to smooth

activation functions, which significantly strengthens adversarial training. The findings of the experiment reveal that SAT enhances adversarial robustness for “free”, that is, without requiring additional calculations or lowering standard accuracy. For example, SAT improves ResNet-50's robustness by 9.3 percent, from 33.0 percent to 42.3 percent, while increasing standard accuracy by 0.9 percent without incurring additional computational cost by training with the computationally cheap single-step PGD attacker (i.e., FGSM attacker with random initialization) on ImageNet [53]. With larger networks, we also investigate the boundaries of SAT.

EfficientNet-L1 [173, 185], which achieves 82.2 percent accuracy and 58.6 percent robustness on ImageNet, outperforms the previous work [159] by 9.5 percent for accuracy and 11.6 percent for robustness.

4.2 Related Works

Adversarial training. By training models using adversarial examples, adversarial training enhances robustness [172, 28, 142, 30]. According to previous research, in order to improve adversarial robustness, we must either trade accuracy on clean inputs [177, 178, 188, 126] or incur greater computing cost [30, 183, 184]. In adversarial robustness, this issue is known as no free lunch [174, 153, 171]. In this chapter, we illustrate how adversarial robustness can be increased for “free” using SAT, with no loss of accuracy on clean images and no additional computing cost.

This work is also connected to a theoretical study [167], which indicates that substituting ReLU with smooth alternatives can aid networks in obtaining a tractable bound when guaranteeing distributional resilience. Later, we show that the advantages of adopting smooth activations are also visible in practical adversarial training on real-world datasets using massive networks.

Gradient masking. Other methods for improving adversarial robustness include defensive distillation [157], gradient discretization [119, 162, 181], dynamic network architectures [125, 175, 176, 146, 143, 149], randomized transformations [182, 117, 180, 160, 141, 114], adversarial input denoising/purification [132, 158, 150, 168, 164, 145, 117, 155, 187, 128], and randomized transformations [182, 117, 180, 160, 141, 114]. However, most of these protection approaches degrade the gradient quality of the protected models, which may cause the gradient masking problem [156]. As discussed in [115] defense strategies based on gradient masking may give the impression of adversarial robustness. In contrast to these studies, we want to improve adversarial robustness by supplying superior gradients to networks, but only in the context of adversarial training.

4.3 ReLU Worsens Adversarial Training

A series of controlled experiments were carried out [113] and represented in this section, focusing on the backward pass of gradient computations, to see how ReLU hampers and how its smooth approximation strengthens adversarial training.

4.3.1 Adversarial Training setup

ResNet-50 [135] is chosen as the backbone network, with ReLU as the default protocol. To generate adversarial perturbations δ, a PGD attacker was used [30]. To reduce the training cost, the cheapest version of PGD attacker, single-step PGD (PGD-1), was chosen. In PGD-1, the maximum per-pixel change ϵ and attack step size β were both set to 4 in accordance with [90, 179]. To train models on ImageNet, standard ResNet training recipes were used: models were trained for a total of 100 epochs using the momentum SGD optimizer, with the learning rate decreased by $10\times$at the 30-th, 60-th, and 90-th epochs; no regularization was applied except for a weight decay of 1e-4.

The model's top-1 accuracy against the 200-step PGD attacker (PGD-200) on the ImageNet validation set was measured when evaluating adversarial robustness, with the maximum perturbation size $\epsilon = 4$ and the step size $\beta = 1$. It was discovered that 200 attack iterations is sufficient to allow a PGD attacker to converge. Furthermore, on the original ImageNet validation set, the model's top-1 accuracy is reported.

4.3.2 How Does Gradient Quality Affect Adversarial Training?

ReLU [134, 152], a frequently used activation function, is non-smooth, as seen in figure 28. When the input of ReLU gets close to 0, the gradient undergoes a sudden change, which decreases the gradient quality dramatically. This non-smooth character was suspected of degrading the training process, particularly when training adversarial models. This is due to the fact that, unlike regular training, which just computes gradients for updating network parameters, adversarial training necessitates additional computations for the inner maximization step to build the perturbation δ.

To solve this problem, a smooth approximation of ReLU called Parametric Softplus [152] is initially introduced:

$$f(\alpha, x) = \frac{1}{\alpha}\log\left(1 + \exp\left(\alpha x\right)\right) \tag{28}$$

where the hyperparameter α is used to control the curve shape. The derivative of this function with respect to the input x is:

$$\frac{d}{dx}f(\alpha, x) = \frac{1}{1+\exp\left(-\alpha x\right)} \tag{29}$$

Empirically, α was set to 10 to better resemble the ReLU curve. Because it has a continuous derivative, Parametric Softplus ($\alpha = 10$) is smoother than ReLU, as demonstrated in figure 28.

The next stage was to determine how gradient quality in the inner maximizing and outer minimization steps influences the standard accuracy and adversarial robustness of ResNet-50 in adversarial training using Parametric Softplus. (To clearly benchmark the effects, just ReLU with Eqn. (29) was substituted in the backward pass, while the forward pass was left intact, i.e., ReLU is always utilized for model inference.)

When the attacker employs Parametric Softplus's gradient (i.e., Eqn. (29)) to create adversarial training samples, the resultant model exhibits a performance trade-off, as illustrated in the second row of Table X. It enhances adversarial robustness by 1.5 percent while degrading standard accuracy by 0.5 percent when compared to the ReLU baseline.

Table X – ReLU significantly weakens adversarial training [113].

Network	Improving Gradient Quality for the Adversarial Attacker	Improving Gradient Quality for the Network Optimizer	Accuracy (%)	Robustness (%)
ResNet-50	×	×	68.8	33.0
	√	×	68.3(-0.5)	34.5(+1.5)
	×	√	69.4(+0.6)	35.8(+2.8)
	√	√	68.9(+0.1)	36.9(+3.9)

It's worth noting that if harsher hostile samples are utilized in adversarial training, a similar performance trade-off can be found [177]. Better gradients for the inner maximizing step may improve the attacker's strength throughout training, according to this observation. The resilience of two ResNet-50 models using PGD-1 (vs. PGD-200 in Table X), one with conventional training and the other with adversarial training, was tested to test this hypothesis. Specifically, the PGD-1 attacker employs ReLU in the forward pass but Parametric Softplus in the backward pass during the evaluation. With improved gradients, the PGD-1 attacker is reinforced and can inflict more damage on models: on the ResNet-50, it can reduce top-1 accuracy by 4.0 percent (from 16.9 percent to 12.9 percent) with standard training and by 0.7 percent (from 48.7 percent to 48.0 percent) with adversarial training.

Improving gradient quality for network parameter updates. The role of gradient quality in updating network parameters during training (i.e., the outer minimization phase) is next investigated. More exactly, ReLU is always employed in the inner step of adversarial training; however, ReLU is used in the forward pass and Parametric Softplus in the backward pass in the outer step of adversarial training.

Surprisingly, this technique enhances adversarial robustness for "free" by setting the network optimizer to use Parametric Softplus's gradient (i.e., Eqn. (29)) to update network

parameters. Comparing the ReLU baseline to the third row of Table X, adversarial robustness improves by 2.8 percent while accuracy improves by 0.6 percent without incurring additional computations. We can see that the training loss is also decreasing: the cross-entropy loss on the training set has decreased from 2.71 to 2.59. These results of improved robustness and accuracy, as well as decreased training loss, show that networks can compute better gradient updates in adversarial training by applying ReLU's smooth approximation in the backward pass of the outer reduction phase.

Surprisingly, we find that better gradient updates can improve standard training as well. With ResNet-50, for example, training with stronger gradients improves accuracy from 76.8 percent to 77.0 percent while lowering the training loss from 1.22 to 1.18 percent. These findings, which apply to both adversarial and normal training, show that changing network parameters with better gradients could be a universal approach for enhancing performance while keeping the model's inference process untouched (i.e., ReLU is always used for inference).

Improving gradient quality for both the adversarial attacker and network parameter updates. We can improve adversarial training by replacing ReLU with Parametric Softplus in all backward runs but preserving ReLU in all forward passes, based on the finding that enhancing ReLU's gradient enhances resilience for either the adversarial attacker or the network optimizer.

As expected, a trained model with this level of resilience had the greatest results so far. It outperforms the ReLU baseline by 3.9 percent in terms of robustness, as indicated in the last row of Table X. Surprisingly, this benefit is still "free": it reports 0.1 percent higher accuracy than the ReLU baseline. In this experiment, we believe that the good effect on accuracy brought about by computing better gradient updates (raise accuracy) marginally outweighed the bad effects on accuracy brought about by creating harder training samples (harm accuracy).

4.3.3 Can ReLU’s Gradient Issue Be Remedied?

More attack iterations. Increasing the number of attack iterations is known to result in more difficult adversarial examples [30, 127]. The supplied tests show that training with the PGD attacker with more iterations results in a model that behaves similarly to the case where we use superior gradients for the attacker. PGD-2, for example, increases the ReLU baseline by 0.6 percent for resilience while sacrificing 0.1 percent for accuracy by increasing the attacker's cost by $2\times$. This finding shows that if more computations are given, we can fix ReLU's gradient issue in the inner stage of adversarial training.

Training longer. Longer training has also been shown to reduce training loss [137], which we will look at next. Interestingly, even though the final model achieves a lower training loss (from 2.71 to 2.62), there is still a performance trade-off between accuracy and robustness when the

default setup is extended to a 2× training schedule (i.e., 200 training epochs). Longer training improves accuracy by 2.6 percent but reduces robustness by 1.8 percent. Previous tests, on the other hand, have shown that using superior gradients to optimize networks enhances both robustness and accuracy. This discouraging finding shows that training longer will not resolve the issues created by ReLU's weak gradient in the outer phase of adversarial training.

Conclusion. We conclude that ReLU considerably impairs adversarial training based on these findings. Furthermore, it appears that even with improved training, the degraded performance cannot be easily reversed (i.e., increasing the number of attack iterations & training longer). It is discovered that ReLU's low gradient is the key—replacing ReLU with its smooth approximation alone in the backward pass enhances robustness significantly without compromising accuracy or incurring additional computational cost. Making activation functions smooth is a good design approach for strengthening adversarial training in general, as we explain in the next section.

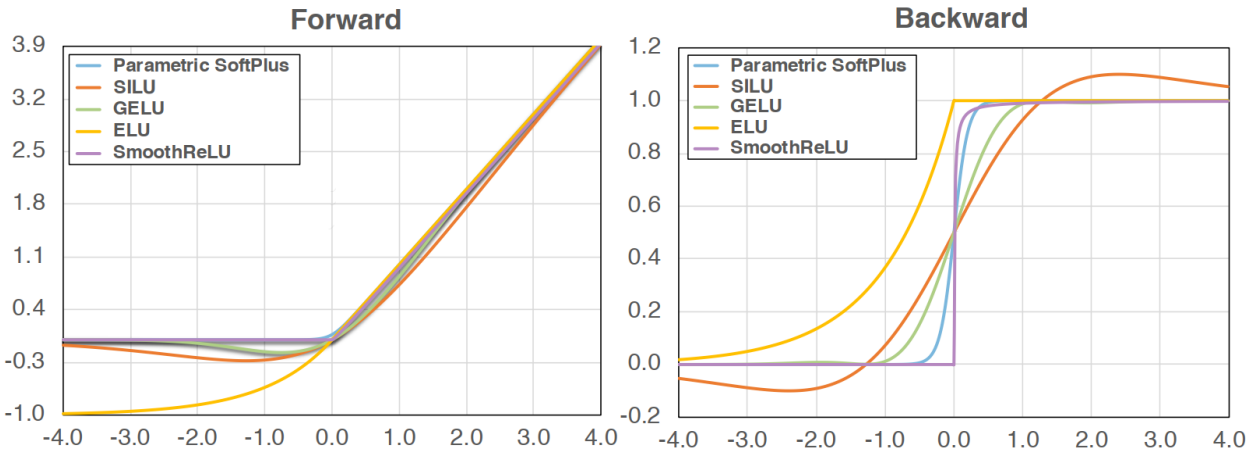

Figure 29 – Visualizations of 5 different smooth activation functions and their derivatives[113]

4.4 Smooth Adversarial Training

Improved ReLU gradient can both empower the attacker and give better gradient updates in adversarial training, as shown in Section 4.3. Nonetheless, this technique may be suboptimal because there is still a difference in training the networks between the forward pass (which uses ReLU) and the backward pass (which uses Parametric Softplus).

Smooth adversarial training (SAT) [113], which ensures architectural smoothness by the sole use of smooth activation functions (in both the forward and backward passes) in adversarial training, is proposed to fully leverage the promise of training with better gradients. All other network components are left unchanged, as the majority of them are smooth and will not cause a

gradient problem. (In SAT, the gradient issue induced by max pooling, which is likewise non-smooth, is ignored. This is because current architectures rarely use it; for example, ResNet [135] only uses one max pooling layer, while EfficientNet [64] uses none.)

4.4.1 Adversarial Training with Smooth Activation Functions

As smooth approximations of ReLU in SAT, the following activation functions are explored (figure 29 depicts these functions as well as their derivatives):

- **Softplus** [152]: $Softplus(x) = log(1 + exp(x))$. Its parametric version as detailed in Eqn. (28) is also considered, where α is set to 10 as in Section 4.3.
- **SILU** [129, 136, 161]: $SILU(x) = x \cdot sigmoid(x)$. Compared to other activation functions, SILU has a nonmonotonic "bump" when $x < 0$.
- **Gaussian Error Linear Unit** (GELU) [24]: $GELU(x) = x \cdot \Phi(x)$, where $\Phi(x)$ is the cumulative distribution function of the standard normal distribution.
- • **Exponential Linear Unit** (ELU) [122]:

$$ELU(x, \alpha) = \begin{cases} \alpha(\exp(x) - 1), & x < 0 \\ x & , \ x \geq 0 \end{cases} \tag{30}$$

where we set $\alpha = 1$ as default. Note that when $\alpha \neq 1$, the gradient of ELU is not continuously differentiable anymore. We will be discussing the effects of these non-smooth variants of ELU ($\alpha \neq 1$) on adversarial training in Section 4.4.3.

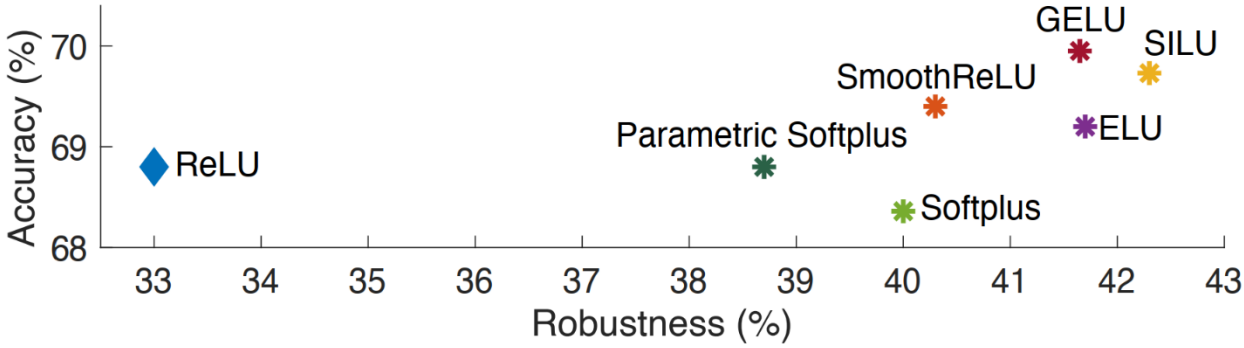

Figure 30 – Smooth activation functions improve adversarial training. Compared to ReLU, all smooth activation functions significantly boost robustness, while keeping accuracy almost the same [113].

Main results. To adversarially train ResNet-50 outfitted with smooth activation functions, the settings in Section 4.3 were used. Figure 30 depicts the outcomes [113]. All smooth activation functions significantly improve robustness as compared to the ReLU baseline, while keeping standard accuracy close to the same. Smooth activation functions, for example, increase robustness by at least 5.7 percent (using Parametric Softplus, from 33 percent to 38.7 percent). SILU achieves the highest robustness, allowing ResNet-50 to attain 42.3 percent robustness and 69.7 percent

standard accuracy. If more advanced smooth alternatives (e.g., [151, 147, 118]) are applied, these results are likely to improve.

In addition, we compare to the setting in Section 4.3, where Parametric Softplus is only used during training on the backward pass. Surprisingly, substituting ReLU with Parametric Softplus at the forward pass improves robustness by 1.8 percent (from 36.9% to 38.7%) while maintaining nearly the same accuracy. This finding emphasizes the need of using smooth activation functions in both forward and backward SAT passes.

4.4.2 Ruling Out the Effect From x < 0

In comparison to ReLU, the functions above contain non-zero responses to negative inputs (x < 0), which may have an impact on adversarial training. To rule out this factor, the SmoothReLU is developed, which flattens the activation function by only altering ReLU after $x \geq 0$ and is influenced by the design of the Rectified Smooth Continuous Unit in [57].

$$SmoothReLU(x;\ \alpha) = \begin{cases} 0 & ,\ x < 0 \\ x - \frac{1}{\alpha}\log(\alpha x + 1), & x \geq 0 \end{cases} \tag{31}$$

where α is a learnable variable that is shared by all channels in a layer and must be positive. We should remark that, regardless of the choice of α, SmoothReLU is always continuously differentiable.

$$\frac{d}{dx}SmoothReLU(x;\ \alpha) = \begin{cases} 0 & ,\ x < 0 \\ \frac{\alpha x}{1+\alpha x}, & x \geq 0 \end{cases} \tag{32}$$

Note When $\alpha \to \infty$, SmoothReLU converges to ReLU. Additionally, to avoid the gradient vanishing problem at the start of training, the learnable parameter α should be started at a large enough value (e.g., 400 in the provided experiments). Figure 29 shows the plot of SmoothReLU and its first derivative.

SmoothReLU outperforms ReLU by 7.3 percent for robustness (from 33.0 percent to 40.3 percent) and by 0.6 percent for accuracy (from 68.8 percent to 69.4 percent), demonstrating the importance of smoothness in a function and ruling out the effect of having replies when $x < 0$.

4.4.3 Case Study: Stabilizing Adversarial Training with ELU using CELU

It has been demonstrated in the preceding study that by substituting ReLU with its smooth approximations, adversarial training can be considerably improved. We describe another sort of activation function—ELU [122]—to further demonstrate the generalization of SAT (beyond ReLU). ELU's first derivative is depicted below:

$$\frac{d}{dx}ELU(x;\ \alpha) = \begin{cases} \alpha\exp(x), & x < 0 \\ 1 & ,\ x \geq 0 \end{cases} \tag{33}$$

The scenario in which ELU is non-smooth is primarily explored here, namely, $\alpha \neq 1$. ELU's gradient is no longer continuously differentiable, as shown by Eqn. (33), i.e., $\alpha\exp(x) \neq 1$ when x = 0, resulting in an abrupt gradient change similar to ReLU. We focus on the region 1.0 to 2.0, where the gradient abruption becomes more pronounced as the value of grows greater.

We see the adversarial training results in Table XI. Surprisingly, we find that adversarial robustness is extremely sensitive on the value of α—the most robustness is obtained when the function is smooth (i.e., $\alpha = 1.0$, 41.4 percent robustness), and all other options of monotonically lower the robustness as it approaches 2.0. For example, while employing $\alpha = 2.0$, the robustness lowers to 33.2 percent, which is 7.9 percent lower than when using $\alpha = 1.0$. Non-smooth activation functions severely degrade adversarial training, which is consistent with the earlier conclusion on ReLU [113].

To stabilize adversarial training with ELU, we use CELU [116], a smooth version of ELU that reparametrizes ELU to the following format:

$$CELU(x;\ \alpha) = \begin{cases} \alpha\exp\left(\frac{x}{\alpha} - 1\right), & x < 0 \\ x, & x \geq 0 \end{cases} \tag{34}$$

Note that CELU is equivalent to ELU when $\alpha = 1.0$. The first derivatives of CELU can be written as follows:

$$\frac{d}{dx} CELU(x;\ \alpha) = \begin{cases} \exp\left(\frac{x}{\alpha}\right), & x < 0 \\ 1, & x \geq 0 \end{cases} \tag{35}$$

CELU is now continuously differentiable regardless of α, thanks to this parameterization.

Table XI – Robustness comparison between ELU (only be smooth α = 1) and CELU (always be smooth ∀α). Compared to ELU, CELU significantly stabilizes robustness in adversarial training.

α	Robustness (%)	
	ELU	CELU
1	41.1	
1.2	-0.3	+0.1
1.4	-2.0	-0.3
1.6	-3.7	-0.3
1.8	-6.2	-0.2
2.0	-7.9	-0.5

As shown in Table XI, CELU significantly improves the stability of adversarial training. The worst case in CELU ($\alpha = 2.0$) is only 0.5 percent lower than the best outcome recorded by ELU/CELU ($\alpha = 1.0$). Remember that the difference for ELU is 7.9 percent. This case study adds to the growing body of evidence supporting the value of taking the SAT.

4.5 Exploring the Limits of Smooth Adversarial Training

According to recent studies [184, 131], adversarial training has a significantly stronger requirement for larger networks to achieve better performance than regular training. However, past studies on the subject have only looked at deeper [184] or wider networks [30], which may be insufficient. To that purpose, we describe a thorough and comprehensive analysis that demonstrates how network scaling up behaves in SAT. SILU is the default activation function for SAT since it is the most robust of all the candidates (figure 30).

4.5.1 Scaling-up ResNet

The network scaling-up studies are first carried out in SAT using the ResNet family. Tan et al. [173] illustrated that all three scaling-up parameters, i.e., depth, width, and image resolutions, are crucial for further improving ResNet performance in standard training. The effects of these scaling-up factors in SAT are considered in this work. The baseline network was ResNet-50 (with a default image resolution of 224).

Depth & width. In typical adversarial training, previous research has demonstrated that expanding networks deeper or wider can improve model performance. In SAT, this finding is confirmed. In SAT, both deeper and wider networks regularly beat the baseline network, as demonstrated in the second through fifth rows of Table XII. For example, training a 3 deeper ResNet-152 improves ResNet-50's accuracy and robustness by 4.2 percent and 3.7 percent, respectively. Similarly, training a 4× wider ResNeXt-50-32x8d [186] enhances accuracy and robustness by 3.9 and 2.8 percent, respectively.

Table XII – Scaling-up ResNet in SAT. We observe SAT consistently helps larger networks get better performance.

	Accuracy (%)	Robustness (%)
ResNet-50	69.7	42.3
+ 2x deeper (ResNet-101)	72.9 (+3.2)	45.5 (+3.2)
+ 3x deeper (ResNet-152)	73.9 (+4.2)	46.0 (+3.7)
+ 2x wider (ResNeXt-50-32x4d)	71.2 (+1.5)	42.5 (+0.2)
+ 4x wider (ResNeXt-50-32x8d)	73.6 (+3.9)	45.1 (+2.8)
+ larger resolution 299	70.9 (+1.2)	43.8 (+1.5)
+ larger resolution 380	71.6 (+1.9)	44.1 (+1.8)
+ 3x deeper & 4x wider (ResNeXt-152-32x8d) & larger resolution 380	**78.2 (+8.5)**	**51.2 (+8.9)**

Image resolution. Though higher image resolution is beneficial for conventional training, it is widely assumed that increasing this factor will result in worse adversarial robustness since the attacker will have more opportunity to create adversarial perturbations [130]. Surprisingly, when

adversarial training is taken into account, this belief may be erroneous. When training with increased image resolutions in SAT, ResNet-50 regularly produces better performance, as seen in the sixth and seventh rows of Table XIII.

Larger image resolution (1) allows attackers to create stronger adversarial examples [130] (which reduces accuracy but improves robustness); and (2) increases network capacity for better representation learning [173] (which improves both accuracy and robustness); and the combination of these two effects empirically shows a positive signal on benefiting SAT overall (i.e., improve both accuracy and robustness).

Compound scaling. So far, we have seen that the fundamental scaling variables of depth, width, and image resolution are all significant scaling-up factors in SAT. Scaling up all of these factors simultaneously will generate a considerably stronger model than focusing on scaling up a single dimension, as argued in [173] for normal training (e.g., depth). To that purpose, an attempt has been made to develop a simple compound scaling for ResNet. As seen in the last row of Table XIV, the resulting model, ResNeXt-152-32x8d with input resolution of 380, outperforms the ResNet-50 baseline by +8.5 percent for accuracy and +8.9 percent for robustness.

Discussion on standard adversarial training. Initially, it is established that basic scaling of depth, width, and image resolution matters in standard adversarial training. For example, scaling up ResNet-50 (33.0 percent robustness), the deeper ResNet-152 achieves 39.4 percent robustness (+6.4 percent), the wider ResNeXt-50-32x8d achieves 36.7 percent robustness (+3.7 percent), and the ResNet-50 with higher image resolution at 380 achieves 36.9 percent robustness (+3.9 percent). Nonetheless, all of these robustness results are lower than the ResNet-50 baseline's resilience in SAT (42.3 percent robustness, first row of Table XV). Scaling up networks, in other words, appears to be less effective than replacing ReLU with smooth activation functions.

For standard adversarial training, we also find that compound scaling is more effective than basic scaling; for example, ResNeXt-152-32x8d with input resolution of 380 reports 46.3 percent robustness. Although this is a better outcome than adversarial training with basic scaling, it is still 5 percent lower than SAT with compound scaling (46.3 percent vs. 51.2 percent). In other words, even with larger networks, smooth activation functions must be used in adversarial training to improve performance.

4.5.2 SAT with EfficientNet

The ResNet results show that scaling up networks in SAT improves performance significantly. Nonetheless, because the scaling policies are hand-crafted without any improvements, they may be suboptimal. EfficientNet [173], which use neural architecture search [189] to determine the best factors for network compound scaling automatically, is a powerful

family of image recognition models. We are now using EfficientNet to replace ResNet in SAT to investigate its benefits. All other training parameters are same to those given in the ResNet experiments.

Figure 31 illustrates that, similar to ResNet, stronger backbones consistently outperform weaker backbones in SAT. Scaling the network from EfficientNet-B0 to EfficientNet-B7, for example, increases robustness from 37.6 percent to 57.0 percent and accuracy from 65.1 percent to 79.8 percent. Remarkably, the increase is still visible for bigger networks: EfficientNet-L1 [185] improves robustness and accuracy by 1.0 percent and 0.7 percent, respectively, over EfficientNet-B7.

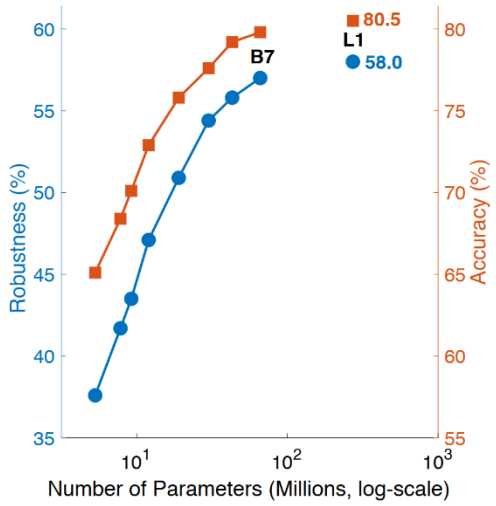

Figure 31 – Scaling-up EfficientNet in SAT [113].

Note EfficientNet-L1 is not connected to the rest of the graph because it was not part of the compound scaling suggested by [173].

Training enhancements on EfficientNet. So far, all of the studies have used ResNet's training recipes, which may not be the optimal for EfficientNet training. As a result, the following training configurations were used in the trials, as indicated in the original EfficientNet work [173]: weight decay was modified from 1e-4 to 1e-5, and Dropout [169], Stochastic Depth [138], and AutoAugment [124] were added to regularize the training process. In addition, to better cope with these training enhancements, models were trained with a longer schedule (i.e., 200 training epochs), an early stopping strategy was used to avoid the catastrophic overfitting issue in robustness [179], and checkpoints were saved using model weight averaging [140] to approximate model ensembling for stronger robustness [154, 170]. The EfficientNet-L1 improves by 1.7 percent in accuracy (from 80.5 percent to 82.2 percent) and 0.6 percent in robustness as a result of these training changes (from 58.0 percent to 58.6 percent).

Comparing to the prior art [49]. Table XVI compares the best outcomes to previous research. The best model (EfficientNet-L1 + SAT) achieves 82.2 percent standard accuracy and 58.6 percent robustness, outperforming the previous state-of-the-art technique [159] by 9.5 percent for standard accuracy and 11.6 percent for adversarial robustness. This gain is primarily due to the fact that in adversarial training, we use a better activation function (SILU vs. Softplus) and a stronger neural architecture (EfficientNet-L1 vs. ResNet-152).

Table XVII – Comparison to the previous state-of-the-art [113].

	Accuracy (%)	Robustness (%)
Prior art [49]	72.7	47.0
EfficientNet + SAT	82.2 (+9.5)	58.6 (+11.6)

Accuracy drop vs. model scale. Finally, for large networks, we highlight a significant reduction in the accuracy gap between adversarially trained models and standard trained models. EfficientNet-L1 obtains 84.1 percent accuracy in normal training (with additions) and 82.2 percent (-1.9 percent) in SAT with the training setup above (with enhancements). This disparity is significantly smaller than the 7.1 percent deficit in ResNet-50 (76.8 percent in standard training v.s. 69.7 percent in SAT). Furthermore, the excellent accuracy of 82.2 percent backs up [139]'s claim that robust features may generalize well to clean inputs.

4.6 Sanity Tests for SAT

Properly evaluating robustness is difficult because numerous parameters (for example, gradient mask) can prevent an adversarial attacker from accurately accessing the model's performance. To prevent these evaluation errors, a series of SAT sanity tests were conducted, as recommended in [120]. The ResNet-50 was used as a token with SILU as the evaluation target.

Robustness vs. attack iterations. The resultant PGD attacker consistently harms the model more as the attack iterations are increased from 5 to 200. When tested against PGD-5, 10, and 50, for example, the model reports robustness of 48.7 percent, 43.7 percent, and 42.7 percent, respectively. When utilizing PGD-200, this evaluation ultimately converges at 42.3 percent.

Robustness vs. perturbation sizes ϵ. A higher disruption budget also enhances the attacker, according to our findings. When we increase ϵ from 4 to 8, the model's robustness drops by more than 25 percent; if we set $\epsilon = 16$, the model is completely evaded.

Landscape visualization. On a randomly selected sample from the ImageNet validation set, the loss landscapes between the provided activation function and the ReLU baseline are compared. The x/y-axis refer to the directions of adversarial perturbation, while the z-axis

corresponds to the associated cross-entropy loss, according to [144]. Figure 32 shows that, when compared to the ReLU baseline, SAT creates a substantially smoother loss landscape. This smooth loss landscape can also be seen when various smooth activation functions (including SILU) are used, as well as on other randomly picked images.

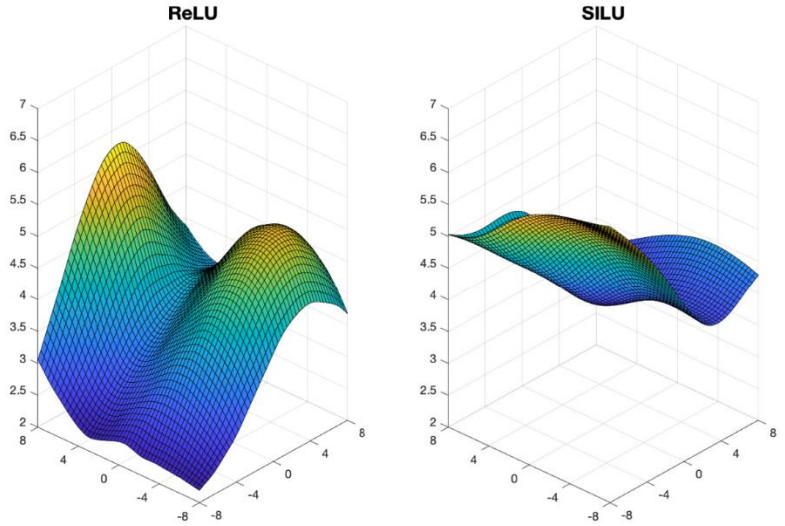

Figure 32 – Comparison of loss landscapes between ReLU baseline and SAT (using SILU) on a randomly selected ImageNet sample [113].

4.7 SAT on CIFAR10

In this part, the performance of SAT on CIFAR10 is briefly examined. ResNet-18 was adversarially trained with the following settings: the attacker is PGD-1 with maximum perturbation $\epsilon = 8$ and step size $\beta = 8$ during training; AutoAttack [123] is used to evaluate these models holistically.

To begin, we note that Softplus, GELU, and SmoothReLU all produce superior results than the ReLU baseline—all three activation functions have similar accuracy but substantially stronger robustness than ReLU. The gains in robustness, for example, range from 0.7 percent (by GELU) to 2.3 percent (by GELU) (by Softplus).

However, unlike the ImageNet tests, where SAT always outperforms ReLU, we find two exceptions on CIFAR10: ELU and SILU both considerably harmed adversarial training as compared to ReLU. Even more intriguing, this poor performance may be seen on the training set. SILU, for example, has a 4.1 percent higher training error than ReLU, while ELU has a 10.6 percent higher training error. Because adversarial training on CIFAR-10 is highly sensitive to different parameter settings, as described in [190, 179], the exploration of the "ideal" adversarial training setting using ELU and SILU on CIFAR-10 is left to other research works [113].

CONCLUSION

In this research first we investigated a proposed sample-dependent adversarial initialization to boost Fast AT. To produce an effective initialization, a generative network conditioned on a benign image and its gradient information from the target network were used. The generative network and the target network are optimized together and play a game during the training phase. The former learns to generate stronger adversarial instances depending on the current target network by producing a dynamic sample-dependent initialization. To improve model robustness, the latter uses the generated adversarial cases for training. The proposed initialization eliminates catastrophic overfitting, therefore improving model robustness, when compared to widely used random initialization methods in Fast AT. Extensive experimental results back up the proposed method's advantages.

In the second part of the research, we looked into a new concept called smooth adversarial training, which enforces architectural smoothness by using adversarial training to replace non-smooth activation functions with their smooth approximations. Without compromising accuracy or incurring additional calculation costs, SAT improves adversarial robustness. Extensive research has shown that SAT is generally effective. SAT reports the state-of-the-art adversarial robustness on ImageNet with EfficientNetL1, which exceeds the previous art by 9.5 percent in accuracy and 11.6 percent in robustness.

Future works

In this work we examined two new concepts in order to boost Adversarial Machine Learning. The FGSM-SDI was implemented using ReLU activation function, therefore the very first research direction could be reimplementing this algorithm SAT. The expectation is to achieve new promising results on overall performance in Adversarial Machine Learning field.

ACKNOWLEDGEMENTS

This work was supported by the Faculty of Innovation Technologies of National Research Tomsk State University. I am greatly thankful to Dr A.A. Poguda, Dr S.V. Shidlovskiy and Dr D. Shashev, who have been guiding me through my research and education.

REFERENCES

1. J. P. Mueller e L. Massaron, Machine Learning for dummies, For Dummies, 2016.
2. S. Raschka e V. Mirjalili, Python Machine Learning, Packt Publishing, 2017.
3. G. Toschi, “Il Machine Learning è divertente!”, 2017. [Online]. Available: https://medium.com/botsupply/il-machine-learning-%C3%A8-divertente-parte-1-97d4bce99a06.
4. L. N. Smith, “A disciplined approach to neural network hyper-parameters: Part 1 -- learning rate, batch size, momentum, and weight decay”, https://arxiv.org/abs/1803.09820, 2018.
5. J. Brownlee, “Difference Between a Batch and an Epoch in a Neural Network”, 2018. [Online]. Available: https://machinelearningmastery.com/differencebetween-a-batch-and-an-epoch/.
6. Y. LeCun, C. Cortes e C. J. Burges, «THE MNIST DATABASE”, 1999. [Online]. Available: http://yann.lecun.com/exdb/mnist/.
7. Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, Andrew Y. Ng Reading Digits in Natural Images with Unsupervised Feature Learning *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*. Available: http://ufldl.stanford.edu/housenumbers
8. A. Krizhevsky, V. Nair e G. Hinton, “Cifar Datasets”, 2009. [Online]. Available: https://www.cs.toronto.edu/~kriz/cifar.html.
9. F.-F. Li e C. Fellbaum, “ImageNet”, 2009. [Online]. Available: http://www.imagenet.org/.
10. D. Vasani, “How do pre-trained models work?”, 2019. [Online]. Available: https://towardsdatascience.com/how-do-pretrained-models-work-11fe2f64eaa2.
11. C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow e R. Fergus, “Intriguing properties of neural networks”, https://arxiv.org/abs/1312.6199, 2013.
12. A. Marchisio, M. A. Hanif, S. Rehman, M. Martina e M. Shafique, “A Methodology for Automatic Selection of Activation Functions to Design Hybrid Deep Neural Networks”, https://arxiv.org/abs/1811.03980, 2018.
13. C. Coleman, D. Narayanan, D. Kang, T. Zhao, J. Zhang, L. Nardi, P. Bailis, K. Olukotun, C. Ré e M. Zaharia, “DAWNBench: An End-to-End Dee Learning Benchmark and Competition”.
14. Y. Guo, C. Zhang, C. Zhang e Y. Chen, “Sparse DNNs with Improved Adversarial Robustness”, https://arxiv.org/abs/1810.09619, 2018.
15. A. Katharopoulos e F. Fleuret, “Not All Samples Are Created Equal: Deep Learning with Importance Sampling”, https://arxiv.org/abs/1803.00942, 2018.

16. J. Xu e S. Sala, “Guida alle architetture di reti profonde”, 2018. [Online]. Available: https://www.deeplearningitalia.com/guida-alle-architetture-di-reti-profonde/.

17. A. Anwar, “Difference between AlexNet, VGGNet, ResNet and Inception”, 2019. [Online]. Available: https://towardsdatascience.com/the-w3h-of-alexnet-vggnetresnet-and-inception-7baaaecccc96.

18. V. Fung, “An Overview of ResNet and its Variants”, 2017. [Online]. Available: https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035.

19. K. He, X. Zhang, S. Ren e J. Sun, “Deep Residual Learning for Image Recognition”, https://arxiv.org/abs/1512.03385, 2015.

20. S. Zagoruyko e N. Komodakis, “Wide Residual Networks”, https://arxiv.org/abs/1605.07146, 2017.

21. F. Nikfam, “Fast Adversarial Training for Deep Neural Networks”, Available: https://webthesis.biblio.polito.it/14386/

22. Kuefler, A., Morton, J., Wheeler, T., & Kochenderfer, M. (2017). Imitating driver behavior with generative adversarial networks. In 2017 IEEE Intelligent Vehicles Symposium (IV) (pp. 204-211). IEEE.

23. Grosse, K., Papernot, N., Manoharan, P., Backes, M., & McDaniel, P. (2017). Adversarial examples for malware detection. In European Symposium on Research in Computer Security (pp. 62-79). Springer.

24. Finlayson, S. G., Bowers, J. D., Ito, J., Zittrain, J. L., Beam, A. L., & Kohane, I. S. (2019). Adversarial attacks on medical machine learning. Science, 363, 1287-1289.

25. Pinto, L., Davidson, J., Sukthankar, R., & Gupta, A. (2017). Robust adversarial reinforcement learning. In Proceedings of the 34th International Conference on Machine Learning-Volume 70 (pp. 2817-2826). JMLR.org.

26. Lin, Y.-C., Hong, Z.-W., Liao, Y.-H., Shih, M.-L., Liu, M.-Y., & Sun, M. (2017). Tactics of adversarial attack on deep reinforcement learning agents. arXiv preprint arXiv:1703.06748.

27. Gleave, A., Dennis, M., Kant, N., Wild, C., Levine, S., & Russell, S. (2019). Adversarial policies: Attacking deep reinforcement learning. arXiv preprint arXiv:1905.10615.

28. Goodfellow, I. J., Shlens, J., & Szegedy, C. (2014b). Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572.

29. Kurakin, A., Goodfellow, I., & Bengio, S. (2016a). Adversarial examples in the physical world. arXiv preprint arXiv:1607.02533.

30. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., & Vladu, A. (2017). Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv:1706.06083.

31. Papernot, N., McDaniel, P., Wu, X., Jha, S., & Swami, A. (2016c). Distillation as a defense to adversarial perturbations against deep neural networks. In 2016 IEEE Symposium on Security and Privacy (SP) (pp.582-597). IEEE.

32. Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z. B., & Swami, A. (2016b). The limitations of deep learning in adversarial settings. In 2016 IEEE European Symposium on Security and Privacy (EuroS&P) (pp. 372-387). IEEE.

33. Moosavi-Dezfooli, S.-M., Fawzi, A., & Frossard, P. (2016a). Deepfool: a simple and accurate method to fool deep neural networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp.2574-2582).

34. Akhtar, N., & Mian, A. (2018). Threat of adversarial attacks on deep learning in computer vision: A survey. IEEE Access, 6, 14410-14430.

35. Jia, R., & Liang, P. (2017). Adversarial examples for evaluating reading comprehension systems. arXiv preprint arXiv:1707.07328.

36. Zugner, D., Akbarnejad, A., & Gunnemann, S. (2018). Adversarial attacks on neural networks for graph data. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (pp. 2847-2856). ACM.

37. Xiang, C., Qi, C. R., & Li, B. (2019). Generating 3d adversarial point clouds. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 9136-9144).

38. Mandlekar, A., Zhu, Y., Garg, A., Fei-Fei, L., & Savarese, S. (2017). Adversarially robust policy learning: Active construction of physically plausible perturbations. In 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 3932-3939). IEEE.

39. Yuan, X., He, P., Zhu, Q., & Li, X. (2019). Adversarial examples: Attacks and defenses for deep learning. IEEE transactions on neural networks and learning systems.

40. Lu, J., Issaranon, T., & Forsyth, D. (2017). Safetynet: Detecting and rejecting adversarial examples robustly. In Proceedings of the IEEE International Conference on Computer Vision (pp. 446-454).

41. Katz, G., Barrett, C., Dill, D. L., Julian, K., & Kochenderfer, M. J. (2017). Towards proving the adversarial robustness of deep neural networks. arXiv preprint arXiv:1709.02802.

42. Bubeck, S., Price, E., & Razenshteyn, I. (2018). Adversarial examples from computational constraints. arXiv preprint arXiv:1805.10204.

43. Tsipras, D., Santurkar, S., Engstrom, L., Turner, A., & Madry, A. (2018). Robustness may be at odds with accuracy. arXiv preprint arXiv:1805.12152.

44. Zhang, H., Yu, Y., Jiao, J., Xing, E. P., Ghaoui, L. E., & Jordan, M. I. (2019a). Theoretically principled trade-off between robustness and accuracy. arXiv preprint arXiv:1901.08573.

45. Biggio, B., Nelson, B., & Laskov, P. (2012). Poisoning attacks against support vector machines. arXiv preprint arXiv:1206.6389.

46. Liu, Y., Xie, Y., & Srivastava, A. (2017). Neural trojans. In 2017 IEEE International Conference on Computer Design (ICCD) (pp. 45{48). IEEE.

47. Chen, X., Liu, C., Li, B., Lu, K., & Song, D. (2017b). Targeted backdoor attacks on deep learning systems using data poisoning. arXiv preprint arXiv:1712.05526.

48. Gu, T., Dolan-Gavitt, B., & Garg, S. (2017). Badnets: Identifying vulnerabilities in the machine learning model supply chain. arXiv preprint arXiv:1708.06733.

49. Weber, M., Xu, X., Karlas, B., Zhang, C., & Li, B. (2020). Rab: Provable robustness against backdoor attacks. arXiv preprint arXiv:2003.08904.

50. Zhao, P., Chen, P.-Y., Das, P., Ramamurthy, K. N., & Lin, X. (2020a). Bridging mode connectivity in loss landscapes and adversarial robustness. arXiv preprint arXiv:2005.00060.

51. Carlini, N., & Wagner, D. (2017). Towards evaluating the robustness of neural networks. In 2017 IEEE Symposium on Security and Privacy (SP) (pp. 39-57). IEEE.

52. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).

53. H. Moore e M. Bassetti, Matlab per l'ingegneria, Pearson, 2008.

54. M. Beri, Python, Apogeo, 2010.

55. M. Buttu, Python, guida completa, Edizioni Lswr, 2014.

56. M. Scarpino, TensorFlow for dummies, For Dummies, 2018.

57. G. B. Team, “TensorFlow”, 2015. [Online]. Available: https://www.tensorflow.org/.

58. L. V. Revision, “TensorFlow Installation”, 2018. [Online]. Available: https://tensorflow-object-detection-apitutorial.readthedocs.io/en/latest/install.html#tensorflow-gpu.

59. A. Sachan, “TensorFlow Model”, 2017. [Online]. Available: https://cvtricks.com/tensorflow-tutorial/save-restore-tensorflow-models-quick-completetutorial/.

60. D. Deutsch, “Debugging TensorFlow”, 2018. [Online]. Available: https://github.com/Createdd/Writing/blob/master/2018/articles/DebugTFBasics.md#5-use-the-tensorflow-debugger.

61. N. Corporation, “CUDA”, 2007. [Online]. Available: https://developer.nvidia.com/cuda-zone.

62. A. Paszke, S. Gross, S. Chintala e G. Chanan, “PyTorch”, 2016. [Online]. Available: https://pytorch.org/.

63. F. Chollet, “Keras”, 2015. [Online]. Available: https://keras.io/.

64. ZHAO YUE (2021-08-04). ON ADVERSARIAL MACHINE LEARNING AND ROBUST OPTIMIZATION. ScholarBank@NUS Repository.

65. A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” Commun. ACM, vol. 60, no. 6, pp. 84–90, 2017.

66. T. N. Sainath, A. Mohamed, B. Kingsbury, and B. Ramabhadran, “Deep convolutional neural networks for LVCSR,” in ICASSP, 2013, pp. 8614– 8618.

67. L. Deng, G. E. Hinton, and B. Kingsbury, “New types of deep neural network learning for speech recognition and related applications: an overview,” in ICASSP, 2013, pp. 8599–8603.

68. D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen et al., “Deep speech 2: End-to-end speech recognition in english and mandarin,” in ICML, 2016, pp. 173–182.

69. W. Zou, S. Huang, J. Xie, X. Dai, and J. Chen, “A reinforced generation of adversarial examples for neural machine translation,” in ACL, 2020, pp. 3486–3497.

70. X. Wei, S. Liang, N. Chen, and X. Cao, “Transferable adversarial attacks for image and video object detection,” in IJCAI, 2019, pp. 954–960.

71. Y. Bai, Y. Zeng, Y. Jiang, Y. Wang, S.-T. Xia, and W. Guo, “Improving query efficiency of black-box adversarial attack,” in ECCV, 2020.

72. Z. Che, A. Borji, G. Zhai, S. Ling, J. Li, Y. Tian, G. Guo, and P. L. Callet, “Adversarial attack against deep saliency models powered by non- redundant priors,” IEEE Trans. Image Process., vol. 30, pp. 1973–1988, 2021.

73. Y. Song, T. Kim, S. Nowozin, S. Ermon, and N. Kushman, “Pixelde- fend: Leveraging generative models to understand and defend against adversarial examples,” in ICLR, 2018.

74. F. Liao, M. Liang, Y. Dong, T. Pang, X. Hu, and J. Zhu, “Defense against adversarial attacks using high-level representation guided denoiser,” in CVPR, 2018, pp. 1778–1787.

75. X. Jia, X. Wei, X. Cao, and H. Foroosh, “Comdefend: An efficient image compression model to defend adversarial examples,” in CVPR, 2019, pp. 6084–6092.

76. T. Dai, Y. Feng, B. Chen, J. Lu, and S.-T. Xia, “Deep image prior based defense against adversarial examples,” Pattern Recognition, p. 108249, 2021.

77. S. Zhang, H. Gao, and Q. Rao, “Defense against adversarial attacks by reconstructing images,” IEEE Trans. Image Process., vol. 30, pp. 6117–6129, 2021.

78. Y. Bai, Y. Zeng, Y. Jiang, S.-T. Xia, X. Ma, and Y. Wang, “Improving adversarial robustness via channel-wise activation suppressing,” in ICLR, 2021.

79. D. Wu, S. Xia, and Y. Wang, “Adversarial weight perturbation helps robust generalization,” in NeurIPS, 2020.

80. Z. Zou, T. Shi, Z. Shi, and J. Ye, “Adversarial training for solving inverse problems in image processing,” IEEE Trans. Image Process., vol. 30, pp. 2513–2525, 2021.

81. Z. Zhao, H. Wang, H. Sun, J. Yuan, Z. Huang, and Z. He, “Removing adversarial noise via low-rank completion of high-sensitivity points,” IEEE Trans. Image Process., vol. 30, pp. 6485–6497, 2021.

82. A. Liu, X. Liu, H. Yu, C. Zhang, Q. Liu, and D. Tao, “Training robust deep neural networks via adversarial noise propagation,” IEEE Trans. Image Process., vol. 30, pp. 5769–5781, 2021.

83. U. Shaham, Y. Yamada, and S. Negahban, “Understanding adversarial training: Increasing local stability of supervised models through robust optimization,” Neurocomputing, vol. 307, pp. 195–204, 2018.

84. Y. Wang, X. Ma, J. Bailey, J. Yi, B. Zhou, and Q. Gu, “On the convergence and robustness of adversarial training,” in ICML, 2019, pp. 6586–6595.

85. G. Katz, C. W. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, “Reluplex: An efficient SMT solver for verifying deep neural networks,” in CAV, 2017, pp. 97–117.

86. T. Weng, H. Zhang, H. Chen, Z. Song, C. Hsieh, L. Daniel, D. S. Boning, and I. S. Dhillon, “Towards fast computation of certified robustness for relu networks,” in ICML, 2018, pp. 5273–5282.

87. L. Rice, E. Wong, and J. Z. Kolter, “Overfitting in adversarially robust deep learning,” in ICML, 2020, pp. 8093–8104.

88. S. Lee, H. Lee, and S. Yoon, “Adversarial vertex mixup: Toward better adversarially robust generalization,” in CVPR, 2020, pp. 269–278.

89. F. Tramèr, A. Kurakin, N. Papernot, I. J. Goodfellow, D. Boneh, and P. D. McDaniel, “Ensemble adversarial training: Attacks and defenses,” in ICLR, 2018.

90. A. Shafahi, M. Najibi, A. Ghiasi, Z. Xu, J. P. Dickerson, C. Studer, L. S. Davis, G. Taylor, and T. Goldstein, “Adversarial training for free!” in NeurIPS, 2019, pp. 3353–3364.

91. E. Wong, L. Rice, and J. Z. Kolter, “Fast is better than free: Revisiting adversarial training,” in ICLR, 2020.

92. M. Andriushchenko and N. Flammarion, “Understanding and improving fast adversarial training,” in NeurIPS, 2020.

93. H. Kim, W. Lee, and J. Lee, “Understanding catastrophic overfitting in single-step adversarial training,” in AAAI, 2021, pp. 8119–8127.

94. Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li, “Boosting adversarial attacks with momentum,” in CVPR, 2018, pp. 9185–9193.

95. J. Lin, C. Song, K. He, L. Wang, and J. E. Hopcroft, “Nesterov accelerated gradient and scale invariance for adversarial attacks,” in ICLR, 2020.

96. C. Xie, Z. Zhang, Y. Zhou, S. Bai, J. Wang, Z. Ren, and A. L. Yuille, “Improving transferability of adversarial examples with input diversity,” in CVPR, 2019, pp. 2730–2739.

97. Y. Dong, T. Pang, H. Su, and J. Zhu, “Evading defenses to transferable adversarial examples by translation-invariant attacks,” in CVPR, 2019, pp. 4312–4321.

98. X. Wang and K. He, “Enhancing the transferability of adversarial attacks through variance tuning,” in CVPR, 2021, pp. 1924–1933.

99. F. Croce and M. Hein, “Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks,” in ICML, 2020, pp. 2206–2216.

100. ——, “Minimally distorted adversarial examples with a fast adaptive boundary attack,” in ICML, 2020, pp. 2196–2205.

101. M. Andriushchenko, F. Croce, N. Flammarion, and M. Hein, “Square attack: A query-efficient black-box adversarial attack via random search,” in ECCV, 2020, pp. 484–501.

102. S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in ICML, 2015, pp. 448–456.

103. T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, “Spectral normal- ization for generative adversarial networks,” in ICLR, 2018.

104. A. F. Agarap, “Deep learning using rectified linear units (relu),” arXiv preprint arXiv:1803.08375, 2018.

105. A. Krizhevsky, G. Hinton et al., “Learning multiple layers of features from tiny images.” Technical Report, 2009.

106. J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in CVPR, 2009, pp. 248–255.

107. K. He, X. Zhang, S. Ren, and J. Sun, “Identity mappings in deep residual networks,” in ECCV, 2016, pp. 630–645.

108. T. Pang, X. Yang, Y. Dong, H. Su, and J. Zhu, “Bag of tricks for adversarial training,” in CVPR, 2021.

109. N. Qian, “On the momentum term in gradient descent learning algorithms,” Neural Networks, vol. 12, no. 1, pp. 145–151, 1999.

110. L. N. Smith, “Cyclical learning rates for training neural networks,” in WACV, 2017, pp. 464–472.

111. Xiaojun Jia, Yong Zhang, Baoyuan Wu, Jue Wang, and Xiaochun Cao. Boosting fast adversarial training with learnable adversarial initialization. arXiv preprint arXiv:2110.05007, 2021.

112. Ehsan’s GitHub [Electronic resource]: https://github.com/ehsansaleh155/Adversarial-Machine_Learning-thesis-codes

113. Cihang Xie, Mingxing Tan, Boqing Gong, Alan Yuille, and Quoc V Le. Smooth adversarial training. arXiv preprint arXiv:2006.14536, 2020.

114. MaungMaung AprilPyone and Hitoshi Kiya. Block-wise im- age transformation with secret key for adversarially robust defense. arXiv preprint arXiv:2010.00801, 2020.

115. Anish Athalye, Nicholas Carlini, and David Wagner. Obfus- cated gradients give a false sense of security: Circumventing defenses to adversarial examples. In ICML, 2018.

116. Jonathan T Barron. Continuously differentiable exponential linear units. arXiv preprint arXiv:1704.07483, 2017.

117. Arjun Nitin Bhagoji, Daniel Cullina, Chawin Sitawarin, and Prateek Mittal. Enhancing robustness of machine learning systems via data transformations. In CISS, 2018.

118. Koushik Biswas, Sandeep Kumar, Shilpak Banerjee, and Ashish Kumar Pandey. Tanhsoft–a family of activation functions combining tanh and softplus. arXiv preprint arXiv:2009.03863, 2020.

119. Jacob Buckman, Aurko Roy, Colin Raffel, and Ian Goodfel- low. Thermometer encoding: One hot way to resist adversar- ial examples. In ICLR, 2018.

120. Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, Aleksander Madry, and Alexey Kurakin. On evaluating adversarial robustness. arXiv preprint arXiv:1902.06705, 2019.

121. Hanlin Chen, Baochang Zhang, Song Xue, Xuan Gong, Hong Liu, Rongrong Ji, and David Doermann. Anti-bandit neural architecture search for model defense. arXiv preprint arXiv:2008.00698, 2020.

122. Djork-Arn e´ Clevert, Thomas Unterthiner, and Sepp Hochre- iter. Fast and accurate deep network learning by exponential linear units (elus). ICLR, 2016.

123. Francesco Croce and Matthias Hein. Reliable evalua- tion of adversarial robustness with an ensemble of diverse parameter-free attacks. In ICML, 2020.

124. Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasude- van, and Quoc V Le. Autoaugment: Learning augmentation policies from data. In CVPR, 2019.

125. Guneet S Dhillon, Kamyar Azizzadenesheli, Zachary C Lip- ton, Jeremy Bernstein, Jean Kossaifi, Aran Khanna, and An- ima Anandkumar. Stochastic activation pruning for robust adversarial defense. In ICLR, 2018.

126. Gavin Weiguang Ding, Yash Sharma, Kry Yik Chau Lui, and Ruitong Huang. Max-margin adversarial (mma) training: Direct input space margin maximization through adversarial training. In ICLR, 2020.

127. Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Xi- aolin Hu, Jianguo Li, and Jun Zhu. Boosting adversarial at- tacks with momentum. In CVPR, 2018.

128. Gintare Karolina Dziugaite, Zoubin Ghahramani, and Daniel M Roy. A study of the effect of jpg compression on adversarial images. arXiv preprint arXiv:1608.00853, 2016.

129. Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid- weighted linear units for neural network function approxi- mation in reinforcement learning. Neural Networks, 2018.

130. Angus Galloway, Anna Golubeva, Thomas Tanay, Med- hat Moussa, and Graham W Taylor. Batch normalization is a cause of adversarial vulnerability. arXiv preprint arXiv:1905.02161, 2019.

131. Ruiqi Gao, Tianle Cai, Haochuan Li, Liwei Wang, Cho-Jui Hsieh, and Jason D Lee. Convergence of adversarial training in overparametrized networks. In NeurIPS, 2019.

132. Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens van der Maaten. Countering adversarial images using input transformations. In ICLR, 2018.

133. Minghao Guo, Yuzhe Yang, Rui Xu, Ziwei Liu, and Dahua Lin. When nas meets robustness: In search of robust archi- tectures against adversarial attacks. In CVPR, 2020.

134. Richard HR Hahnloser, Rahul Sarpeshkar, Misha A Ma- howald, Rodney J Douglas, and H Sebastian Seung. Digi- tal selection and analogue amplification coexist in a cortex-inspired silicon circuit. Nature, 2000.

135. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In CVPR, 2016.

136. Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). arXiv preprint arXiv:1606.08415, 2016.

137. Elad Hoffer, Itay Hubara, and Daniel Soudry. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. In Advances in Neural Information Processing Systems, 2017.

138. Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In ECCV, 2016.

139. Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversar- ial examples are not bugs, they are features. In NeurIPS, 2019.

140. Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. arXiv preprint arXiv:1803.05407, 2018.

141. Markus Kettunen, Erik H a¨rko¨nen, and Jaakko Lehtinen. E- lpips: Robust perceptual image similarity via random trans- formation ensembles. arXiv preprint arXiv:1906.03973, 2019.

142. Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adver- sarial machine learning at scale. In ICLR, 2017.

143. Hakmin Lee, Hong Joo Lee, Seong Tae Kim, and Yong Man Ro. Robust ensemble model training via ran- dom layer sampling against adversarial attack. arXiv preprint arXiv:2005.10757, 2020.

144. Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In NeurIPS, 2018.

145. Fangzhou Liao, Ming Liang, Yinpeng Dong, and Tianyu Pang. Defense against adversarial attacks using high-level representation guided denoiser. In CVPR, 2018.

146. Xuanqing Liu, Minhao Cheng, Huan Zhang, and Cho-Jui Hsieh. Towards robust neural networks via random self- ensemble. In ECCV, 2018.

147. Vishnu Suresh Lokhande, Songwong Tasneeyapant, Abhay Venkatesh, Sathya N. Ravi, and Vikas Singh. Generating ac- curate pseudo-labels in semi-supervised learning and avoid- ing overconfident predictions via hermite polynomial activa- tions. In CVPR, 2020.

148. Yan Lou, Xavier Boix, Gemma Roig, Tomaso Poggio, and Qi Zhao. Foveation-based mechanisms alleviate adversarial examples. arXiv preprint arXiv:1511.06292, 2015.

149. Tiange Luo, Tianle Cai, Mengxiao Zhang, Siyu Chen, and Liwei Wang. Random mask: Towards robust convolutional neural networks. arXiv preprint arXiv:2007.14249, 2020.

150. Dongyu Meng and Hao Chen. Magnet: a two-pronged de- fense against adversarial examples. In CCS, 2017.

151. Diganta Misra. Mish: A self regularized non-monotonic neural activation function. arXiv preprint arXiv:1908.08681, 2019.

152. Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In ICML, 2010.

153. Preetum Nakkiran. Adversarial robustness may be at odds with simplicity. arXiv preprint arXiv:1901.00532, 2019.

154. Tianyu Pang, Kun Xu, Chao Du, Ning Chen, and Jun Zhu. Improving adversarial robustness via promoting ensemble diversity. In ICML, 2019.

155. Tianyu Pang, Kun Xu, and Jun Zhu. Mixup inference: Bet- ter exploiting mixup to defend adversarial attacks. In ICLR, 2020.

156. Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practi- cal black-box attacks against machine learning. In AsiaCCS, 2017.

157. Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversar- ial perturbations against deep neural networks. In SP, 2016.

158. Aaditya Prakash, Nick Moran, Solomon Garber, Antonella DiLillo, and James Storer. Deflecting adversarial attacks with pixel deflection. In CVPR, 2018.

159. Chongli Qin, James Martens, Sven Gowal, Dilip Krishnan, Krishnamurthy Dvijotham, Alhussein Fawzi, Soham De, Robert Stanforth, and Pushmeet Kohli. Adversarial robust- ness through local linearization. In NeurIPS, 2019.

160. Edward Raff, Jared Sylvester, Steven Forsyth, and Mark McLean. Barrage of random transforms for adversarially ro- bust defense. In CVPR, 2019.

161. Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. arXiv preprint arXiv:1710.05941, 2017.

162. Andras Rozsa and Terrance E Boult. Improved adversarial robustness by reducing open space risk via tent activations. arXiv preprint arXiv:1908.02435, 2019.

163. Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, San- jeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Chal- lenge. IJCV, 2015.

164. Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-gan: Protecting classifiers against adversarial at- tacks using generative models. In ICLR, 2018.

165. Lukas Schott, Jonas Rauber, Matthias Bethge, and Wieland Brendel. Towards the first adversarially robust neural net- work model on mnist. In ICLR, 2019.

166. Gil Shamir, Dong Lin, and Lorenzo Coviello. Smooth acti- vations and reproducibility in deep networks. arXiv preprint arXiv:2010.09931, 2020.

167. Aman Sinha, Hongseok Namkoong, and John Duchi. Cer- tifying some distributional robustness with principled adver- sarial training. In ICLR, 2018.

168. Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Er- mon, and Nate Kushman. Pixeldefend: Leveraging genera- tive models to understand and defend against adversarial ex- amples. In ICLR, 2018.

169. Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. JMLR, 2014.

170. Thilo Strauss, Markus Hanselmann, Andrej Junginger, and Holger Ulmer. Ensemble methods as a defense to adversarial perturbations against deep neural networks. arXiv preprint arXiv:1709.03423, 2017.

171. Dong Su, Huan Zhang, Hongge Chen, Jinfeng Yi, Pin-Yu Chen, and Yupeng Gao. Is robustness the cost of accuracy?– a comprehensive study on the robustness of 18 deep image classification models. In ECCV, 2018.

172. Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. In- triguing properties of neural networks. In ICLR, 2014.

173. Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In ICML, 2019.

174. Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. There is no free lunch in adversarial robustness (but there are unexpected benefits). In ICLR, 2019.

175. Siyue Wang, Xiao Wang, Pu Zhao, Wujie Wen, David Kaeli, Peter Chin, and Xue Lin. Defensive dropout for hardening deep neural networks under adversarial attacks. In ICCAD, 2018.

176. Xiao Wang, Siyue Wang, Pin-Yu Chen, Yanzhi Wang, Brian Kulis, Xue Lin, and Peter Chin. Protecting neural net- works with hierarchical random switching: Towards better robustness-accuracy trade-off for stochastic defenses. In IJ- CAI, 2019.

177. Yisen Wang, Xingjun Ma, James Bailey, Jinfeng Yi, Bowen Zhou, and Quanquan Gu. On the convergence and robustness of adversarial training. In ICML, 2019.

178. Yisen Wang, Difan Zou, Jinfeng Yi, James Bailey, Xingjun Ma, and Quanquan Gu. Improving adversarial robustness requires revisiting misclassified examples. In ICLR, 2020.

179. Eric Wong, Leslie Rice, and J Zico Kolter. Fast is better than free: Revisiting adversarial training. In ICLR, 2020.

180. Chang Xiao and Changxi Zheng. One man’s trash is another man’s treasure: Resisting adversarial examples by adversar- ial examples. In CVPR, 2020.

181. Chang Xiao, Peilin Zhong, and Changxi Zheng. Enhancing adversarial defense by k-winners-take-all. In ICLR, 2019.

182. Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. Mitigating adversarial effects through random- ization. In ICLR, 2018.

183. Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan Yuille, and Kaiming He. Feature denoising for improving adversar- ial robustness. In CVPR, 2019.

184. Cihang Xie and Alan Yuille. Intriguing properties of adver- sarial training at scale. In ICLR, 2020.

185. Qizhe Xie, Eduard Hovy, Minh-Thang Luong, and Quoc Le. Self-training with noisy student improves imagenet classifi- cation. In CVPR, 2020.

186. Saining Xie, Ross Girshick, Piotr Doll a´r, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In CVPR, 2017.

187. Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. In NDSS, 2017.

188. Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P Xing, Laurent El Ghaoui, and Michael I Jordan. Theoretically prin- cipled trade-off between robustness and accuracy. In ICML, 2019.

189. Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. In ICLR, 2016.

190. Tianyu Pang, Xiao Yang, Yinpeng Dong, Hang Su, and Jun Zhu. Bag of tricks for adversarial training. In ICLR, 2021.