SARAF: Searching for Adversarial Robust Activation Functions

Maghsood Salimi, Mohammad Loni, Marjan Sirjani, Antonio Cicchetti, Sara Abbaspour Asadollah
School of Innovation, Design and Engineering, Mälardalen University
Västerås, Sweden
{maghsood.salimi,mohammad.loni,marjan.sirjani,antonio.cicchetti,sara.abbaspour}@mdu.se

ABSTRACT

Convolutional Neural Networks (CNNs) have received great attention in the computer vision domain. However, CNNs are vulnerable to adversarial attacks, which are manipulations of input data that are imperceptible to humans but can fool the network. Several studies tried to address this issue, which can be divided into two categories: (i) training the network with adversarial examples, and (ii) optimizing the network architecture and/or hyperparameters. Although adversarial training is a sufficient defense mechanism, they suffer from requiring a large volume of training samples to cover a wide perturbation bound. Tweaking network activation functions (AFs) has been shown to provide promising results where CNNs suffer from performance loss. However, optimizing network AFs for compensating the negative impacts of adversarial attacks has not been addressed in the literature. This paper proposes the idea of searching for AFs that are robust against adversarial attacks. To this aim, we leverage the Simulated Annealing (SA) algorithm with a fast convergence time. This proposed method is called SARAF. We demonstrate the consistent effectiveness of SARAF by achieving up to 16.92%, 18.3%, and 15.57% accuracy improvement against BIM, FGSM, and PGD adversarial attacks, respectively, over ResNet-18 with ReLU AFs (baseline) trained on CIFAR-10. Meanwhile, SARAF provides a significant search efficiency compared to random search as the optimization baseline.

CCS CONCEPTS

• **Computing methodologies → Neural networks**; • **Theory of computation → Simulated annealing**.

KEYWORDS

Optimization, Adversarial Attack, Activation Function, Convolutional Neural Network, Robustness

ACM Reference Format:
Maghsood Salimi, Mohammad Loni, Marjan Sirjani, Antonio Cicchetti, Sara Abbaspour Asadollah. 2023. SARAF: Searching for Adversarial Robust Activation Functions. In *2023 The 6th International Conference on Machine Vision and Applications (ICMVA) (ICMVA 2023), March 10–12, 2023, Singapore, Singapore.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3589572.3589598

ICMVA 2023, March 10–12, 2023, Singapore, Singapore
© 2023 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9953-1/23/03.
https://doi.org/10.1145/3589572.3589598

1 INTRODUCTION

Adversarial attacks are perturbed inputs that preserve labels but fool deep learning models [42]. Recent studies demonstrate that adversarial attacks can present a significant threat to various applications, such as computer vision [34], cyber-physical systems [50], medical machine learning models [12], and wireless communication [1]. Convolutional Neural Networks (CNNs) have shown their great ability to solve problems in various artificial intelligence fields such as computer vision [29, 62], natural language processing [30, 44], bioinformatics [18, 43], and machine translation [11, 52]. However, CNNs are vulnerable to adversarial attacks [5, 13, 53]. A number of studies have been conducted to address this issue, mostly utilizing robust training methods [16, 26, 49]. The methods used for robust training are challenging because they require large input datasets, which necessitates a lot of resource-intensive data augmentation processes [61]. Additionally, adversarial examples suffer from instabilities caused by physical transformations, such as translation, illumination, and rotation [53, 61].

Despite the success of previous studies in improving adversarial robustness (Section 3.1), no studies have been conducted to examine the impact of optimizing network AFs over the robustness of CNNs against perturbed examples. In addition, most of the proposed AF tweaking methods have huge computing demands (up to 2000 GPU hours [2]), resulting in a lack of interest in AF optimization for various deep learning problems.

In this paper, we introduce SARAF, a method that discovers activation functions that make CNNs more robust against adversarial attacks by considering robustness accuracy as the search objective. We leverage the Simulated Annealing (SA) algorithm [27] as a metaheuristic search method. SARAF has a fast convergence which is due to the single-solution nature of SA, while for example, the genetic algorithms are relatively slow due to a population-based optimization [40]. Unlike previous work [9], SARAF is a generic optimization approach that does not require data augmentation or adversarial training. Inspired by [38, 39], we rely on lower fidelity estimations by training each candidate during the search iterations with fewer epochs, leading to expediting the search procedure by up to 13× compared to [2]. Subsequently, to achieve maximum performance, we need to fully train the network weights after searching for AFs with more epochs.

Paper Contributions. The main contributions of this paper are listed in the following:

(1) We introduce a fast search method, dubbed SARAF, that optimizes network AFs against adversarial attacks in a reasonable time (up to 6 end-to-end GPU days for a ResNet-18 architecture trained on CIFAR-10).
(2) We propose a flexible search space by coding the architecture as a list of operational nodes with a variable size.

SARAF demonstrates its consistent effectiveness by achieving up to 16.92%, 18.3%, and 15.57% accuracy improvement against BIM, FGSM, and PGD adversarial attacks, respectively, over ResNet-18 with ReLU AF (as the baseline architecture) trained on CIFAR-10. Also, the robustness accuracy of AlexNet against BIM, FGSM, and PGD adversarial attacks is improved by 6.86%, 8.72%, and 11.14% respectively. Meanwhile, SARAF provides a significant search efficiency by requiring up to 6 GPU days for optimizing the network, which is 13× faster compared to other competing methods. Finally, SARAF generates similar results with 6.7% STDEV, thus demonstrating our results are reproducible.

2 BACKGROUND ON ADVERSARIAL ATTACKS

Research on adversarial attacks initially focused on systems that provide safety functionality [10, 23]. An adversarial example, defined as inputs that fool a network with high confidence but cannot be spotted by humans, was added to the literature in 2013 [53]. The vulnerability of convolutional neural networks (CNNs) to adversarial attacks has been shown in many studies [10, 15, 23, 45, 53]. To formalize adversarial attacks, assume N is a CNN classifier and X is the input sample with correct classification, i.e., $N(X) = Y_{true}$. An adversarial example (X^{adv}) is constructed by adding a small perturbation to X, where N classifies it to a wrong label, i.e., $N(X^{adv}) \neq Y_{true}$. There are two types of adversarial attacks: white-box and black-box. White-box attacks have access to the structure and parameters of the network, while black-box attacks have limited knowledge of these details. This paper examines three popular white-box attacks, which are described below.

2.1 Fast Gradient Sign Method (FGSM) [15]

FGSM was introduced as a pioneering white-box attack. A small vector is added to the input sample by FGSM, in which the elements of the vector are the same as the sign of elements of the gradient of the loss function (Eq.1).

$$X^{adv} = X + \epsilon * sign(\nabla_X \mathcal{L}(\theta, X, Y_{true})) \quad (1)$$

where θ denotes the parameters of the network, ϵ is a small value that restricts the amount of perturbation, ∇ computes the gradient of the loss function with respect to an original sample X with correct label Y classified by the network. A stronger attack may be achieved by increasing ϵ.

2.2 Projected Gradient Descent (PGD) [42]

PGD is an iterative extension of FGSM which performs a projected gradient descent on the negative loss function. PGD starts with a randomly initialized perturbation $X_0^{adv} \in \mathcal{S}$, which is updated at each step via Eq.3:

$$X_{t+1}^{adv} = \Pi_{\mathcal{S}}(X_t^{adv} + \alpha * sign(\nabla_X \mathcal{L}(\theta, X + X_t^{adv}, Y_{true})) \quad (2)$$

X and Y_{true} are the input sample and corresponding output label, $\mathcal{L}$ denotes the loss function, and $\mathcal{S}$ is a set of allowed perturbations.

2.3 Basic Iterative Method (BIM) [31]

BIM is an extension to FGSM. It applies the FGSM with small steps while trying to hold the resulting perturbation close to the original data by using a clipping function by iteration (Eq.3).

$$X_0^{adv} = X;\ X_{n+1}^{adv} = Clip_{X,\epsilon}\{X_n^{adv} + \alpha * sign(\nabla_X \mathcal{L}(\theta, X_n^{adv}, Y_{true}))\} \quad (3)$$

where α denotes the step size and $Clip_{X,\epsilon}\{A\}$ is the element-wise clipping of X. Although BIM is computationally intensive, it has a higher chance to fool the network since adversarial samples are closer to the original input.

3 RELATED WORK

3.1 Activation Function Optimization

To ensure that CNNs perform effectively, Activation Functions (AFs) must be present to provide non-linearity. This section reviews major studies that tried to improve the performance of CNNs by optimizing network AFs.

Mathematically, AFs are nonlinear functions that convert the weighted sum of the neurons' input into an output. As a very well-known activation function example, ReLU [19] (Rectified Linear Unit) is defined as $f(x) = max(0, x)$ where x is the input vector of the neuron and $f(x)$ defines the output of that node. ReLU is one of the most widely used activation functions in CNNs' architectures [48]. Besides ReLU, a lot of different activation functions were developed and studied how they affect the performance of artificial neural networks [21, 48].

[14] proposed a novel activation function with the ability of continuous logarithmic, linear and exponential functions which potentially could help neural networks to perform better by replacing the piece-wise linear nature of ReLU AF. SReLU [24] extends ReLU by combining three piece-wise linear functions, which are formulated by four learnable parameters. SReLU outperforms ReLU on popular image classification datasets. Swish [48] was discovered using an RL-based AutoML technique. Although Swish works better than ReLU across various challenging datasets, it suffers from an expensive design cost. [8, 38] leveraged evolutionary algorithms to find the best-performing solution from a predefined set of AFs. Even though these methods are simple, they are not very effective at improving performance. [3] proposes a multi-stage optimization for designing a learnable AF. In the first stage, they utilized an evolutionary algorithm to find the general form of the function. Afterward, they use the gradient descent algorithm to optimize learnable parameters during the training process. [2] designed new AFs by merging them using a set of binary and unary operations. An evolutionary algorithm is leveraged to search operations and in a tree-based design space. [36, 46] aimed to optimize the performance of quantized neural networks by automatically searching for better AFs using the genetic algorithm. Despite the effectiveness of prior studies, they suffer from significant search costs due to leveraging RL or evolutionary algorithms. Compared to prior studies, this paper focuses on finding new AFs using simulated annealing in order to expedite the search process (up to 13× faster search).

3.2 Adversarial Robustness

It has been demonstrated that AFs play a key role in the vulnerability of CNNs [4, 58]. Recently, researchers have tried to address this problem by proposing new activation functions for CNNs that are robust against adversarial attacks [9, 47, 54, 56, 59, 60]. CROWN [60] certified the robustness of CNNs by using different AFs for different layers and bounding them with linear and quadratic functions. Since ReLU is not smooth and inputs close to zero cause its gradient to abruptly change, the Softplus AF is proposed by [56] whose derivative is continuous and n-times differentiable. Using smooth AFs in the process of adversarial training helps to find more difficult adversarial examples. In [47], authors studied the robustness of different types of CNNs' layers (forward, convolutional and residual) against adversarial attacks based on Lyapunov theory where each individual layer of the network is treated locally as a nonlinear system and its robustness is proved. Instead of using non-linear AFs, SPLASH uses piece-wise linear AFs which boosts the robustness of CNNs against adversarial attacks and accuracy as well [54]. CNNs can be more robust by using SPLASH instead of ReLU. Authors in [9] studied the impact of AFs' shape on the accuracy and also the robustness of CNNs by parameterizing different AFs. To this aim, they added an α parameter to various AFs and studied the effects of tweaking the α parameter. Due to the use of only restricted values and the linear effect of the α parameter, this performance gain is limited. In addition, [9] utilized the adversarial training technique for the sake of robustifying input datasets for optimizing AFs. In contrast, SARAF is more generalized as it does not make any assumptions regarding the input dataset.

4 PROPOSED OPTIMIZATION METHOD

This work aims to increase the accuracy of CNNs against adversarial attacks by searching for robust AFs. In general, searching for new AFs is an NP-hard problem with an exponential time complexity [38]. Thus, in a reasonable time, polynomial optimization cannot find the optimal solution. In addition, using exhaustive search methods is infeasible in practice, e.g., to exhaustively search an 8000-solution design space, [41] needs 334 GPU days. To this end, we utilize a meta-heuristic search method to deal with the exponential complexity of the AF search problem. The proposed search space and search method can be found in Section 4.1 and Section 4.2, respectively.

4.1 Search Space

To use search methods, the first step is to define the search space. Let us assume we have a CNN model with l hidden layers. The goal is to replace the AFs of these layers with newly built AFs or with AFs from a pre-defined set of candidates. Inspired by [51], the search space is represented by vectors which we call *chromosomes*. Chromosomes are divided into three parts, each of which has a length of l. As a result, the total length of the chromosome would be $3 \times l$. Fig. 1 shows an example of a chromosome for a CNN with four hidden layers.

In this work, a set of potential candidate activation functions is selected where different operations could be applied to them. We considered ReLU, LeakyReLU, Sigmoid, SELU, CELU, Mish, and GELU as the candidate activation functions.

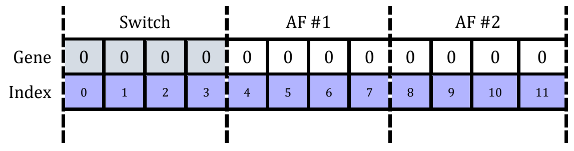

Figure 1: A sample chromosome illustrating a CNN with four hidden layers, all with ReLU activation function.

The search space size depends on the number of layers in the CNN, the number of activation functions in the candidate list, possible values for the constant coefficient ({0.25, 0.5, 0.75}), and the number of mathematical operations. Assume l is the number of layers, α is the number of candidates, β is the number of possible values for the constant coefficient, and the number of possible mathematical operations would be θ. Then, the size of the search space will be calculated based on the following formula: $\textit{Search Space Size} = \alpha \times (1 + \beta + (\alpha \times \theta))^l$. According to Table 1, the size of the search space is equal to 7×32^7 for AlexNet with seven hidden layers.

As mentioned before, each chromosome is divided into three parts. The first part of each chromosome is *Switch* which selects the corresponding operation for building new AFs. For the sake of simplicity, every possible option for *Switch* is coded into the numbers, as listed in Table 1.

Fig. 2 shows two different chromosome examples of new AFs for a CNN with four hidden layers. In Fig. 2.a, since the value for the genes at indices 0, 1, and 3 of *Switch* part are zero, then single AFs from *AF #1* will be selected from indices 4, 5, and 7, respectively. As the value for indices 4, 5, and 7 are zero, the selected AFs for the 1st, 2nd, and 4th hidden layers would be $ReLU(x)$. Also the AF for 3rd hidden layer would be $0.25 \times GELU(x)$

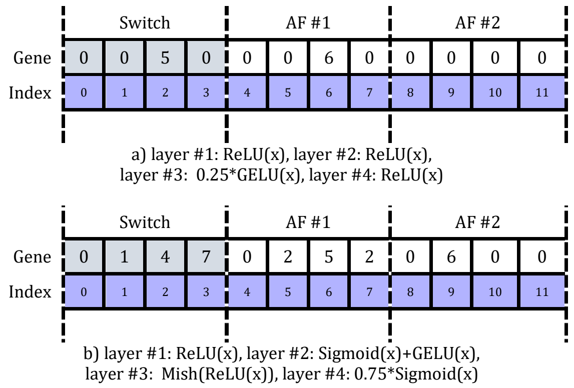
Switch
AF #1
AF #2
Gene
Index
0 0 5 0 0 0 6 0 0 0 0 0
0 1 2 3 4 5 6 7 8 9 10 11
a) layer #1: ReLU(x), layer #2: ReLU(x),
layer #3: 0.25*GELU(x), layer #4: ReLU(x)
Switch
AF #1
AF #2
Gene
Index
0 1 4 7 0 2 5 2 0 6 0 0
0 1 2 3 4 5 6 7 8 9 10 11
b) layer #1: ReLU(x), layer #2: Sigmoid(x)+GELU(x),
layer #3: Mish(ReLU(x)), layer #4: 0.75*Sigmoid(x)

Figure 2: A chromosome example for a CNN with four hidden layers.

A remarkable flexibility is provided by the proposed search space for the generation of new AFs. For example, consider $g(x)$ and $h(x)$ are two AFs. Then, a new AF $f(x)$ could be defined as the following:

Table 1: Possible values for the *Switch* part of the chromosome and corresponding operations.

Code	*Switch*	Description
0	$f(x) = g(x)$	Replacing current AF with another AF selected from the list of candidates
1	$f(x) = g(x) + h(x)$	Accumulating selected AFs, where $g(x)$ comes from *AF #1* and $h(x)$ comes from *AF #2*
2	$f(x) = g(x) - h(x)$	A minus operation is performed on the selected AFs, $g(x)$ and $h(x)$
3	$f(x) = g(x) \times h(x)$	Multiplication of selected AFs, $g(x)$ and $h(x)$
4	$f(x) = g(h(x))$	Composition of selected AFs
5	$f(x) = 0.25 \times g(x)$	A constant value of 0.25 is multiplied by the selected AF $g(x)$ from *AF #1*
6	$f(x) = 0.5 \times g(x)$	A constant value of 0.5 is multiplied by the selected AF $g(x)$ from *AF #1*
7	$f(x) = 0.75 \times g(x)$	A constant value of 0.75 is multiplied by the selected AF $g(x)$ from *AF #1*

- $f(x) \rightarrow g(x)$ (replacing the current AF with a new AF)
- $f(x) = \alpha \times g(x)$
- $f(x) = g(x)+h(x)$, $f(x) = g(x)\times h(x)$, or $f(x) = g(x)-h(x)$
- $f(x) = g(h(x))$.

Results of applying different operations on two examples AFs, Sigmoid and Tanh, are shown in Fig. 3. Results demonstrate that by applying different operations, newly generated AFs significantly differ from the original ones, indicating the proposed search space is flexible to generate very different outputs. Our experiments (Section 6) show the proposed search space significantly improves the performance of CNNs when input data is perturbed by different adversarial attacks.

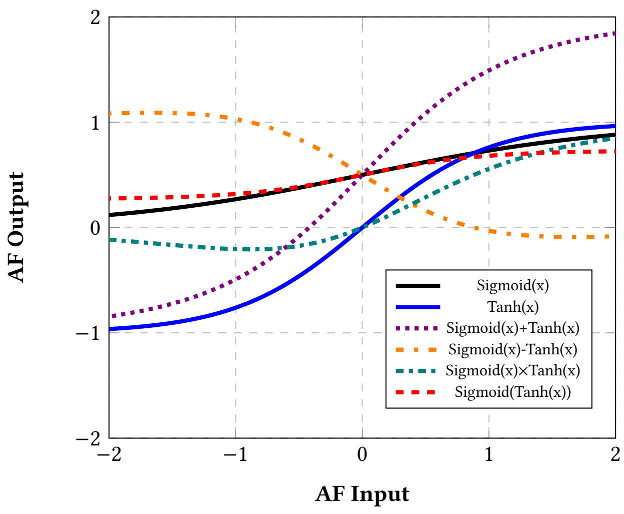

Figure 3: Generating different mathematical operations using Sigmoid and Tanh AFs.

4.2 Search Strategy

As mentioned in Section 4.1, a meta-heuristic search algorithm fits better for our non-convex optimization problem. Several research studies examined different meta-heuristic optimization methods, e.g., genetic algorithm [38], late acceptance hill-climbing [39], simulated annealing [39], and particle swarm optimization [22], to solve different problems. In this work, we leverage the simulated annealing (SA) algorithm [27] to find the near-optimal solutions for robust AF search.

SA is a probabilistic single-solution-based search method that iteratively explores solutions with fewer *Energy* function values. If a reduction in the *Energy* function is found, the current solution is replaced with the newly generated neighbor, otherwise, the current solution remains unchanged. To avoid becoming trapped in a local optimum, SA sometimes accepts a bad solution with a probability of $exp(-\Delta/(k\times T))$. k is the Boltzmann's constant and T is the cooling parameter which is decreased with a logarithmic shape based on the predefined maximum (T_{Max}) and minimum temperatures (T_{Min}). SA starts with a high T_{Max} for preventing being prematurely trapped in a local optimum. By approaching T toward T_{Min}, most uphill moves will be rejected. The SA process continues until no further improvements can be made or it will be terminated after a specified number of iterations. In this paper, the *Energy* function of the SA algorithm is defined as (Eq. 4):

$$Energy = 1 - Robustness\ Accuracy \tag{4}$$

In this paper, the *Energy* function and the objective function are used interchangeably.

In general, these optimization algorithms involve a significant number of performance evaluations on candidate architectures. Each performance evaluation usually requires hundreds of training epochs, leading to a remarkable computational cost (e.g., up to 2,000 GPU days [63]). The key motivation for using SA as the search method is that SA enables faster search compared to genetic programming and random search [6, 25, 40]. Our experimental results show that SARAF requires ≈6 GPU days on a single NVIDIA® RTX A4000 for finding the best-performing activation function for ResNet-18 trained on the CIFAR-10 dataset. In the end, it is worth mentioning that the convergence of SA to global results is guaranteed [17].

5 EXPERIMENTAL SETUP

This section describes the experimental setup used to evaluate the proposed method, including evaluation datasets, search, and training details.

5.1 Evaluation Dataset

To evaluate SARAF, we use MNIST [33] and CIFAR-10 [28] classification datasets. The MNIST dataset consists of 70k hand-written labeled digits from zero to nine, 60k as training, and 10k as test data. Note that since MNIST is a tiny dataset and complex CNNs tend to saturate it, the primary purpose of our experiments on MNIST is to validate the consistency of our method. CIFAR-10 consists of 60k 32×32 colorful images in ten different classes, 50k as training and 10k as test data. We leverage CIFAR-10 in this study since CIFAR-10 is a subset of ImageNet with 330 similar categories [7]; thus, compared to other complex classification datasets, it probably has more relevant representations.

5.2 CNN Architecture

In this work, we studied AlexNet [29], the 2012 winner of the ILSVRC (ImageNet Large Scale Visual Recognition Competition) with a top-5 error rate of 15.3%, and ResNet-18, a variation of ResNet [20] the 2015 winner of the ILSVRC with a 3.57% error on the ImageNet test set. Activation functions can be changed in both architectures based on the configuration provided.

Table 2: Summarizing experimental setup.

	Search Configuration		
Parameter	MNIST (AlexNet)	CIFAR-10 (AlexNet)	CIFAR-10 (ResNet-18)
Training Epoch	10	40	15
Optimizer	Adam	Adam	Adam
Learning Rate	0.001	0.001	0.001
Weight Decay	0	0	0
Train Batch Size	1000	512	256
Test Batch Size	1000	256	128
	Hardware Specification		
GPU	NVIDIA® RTX A4000		
GPU Memory	16 GB		
GPU Compiler	NVIDIA® NVCC v. 10.1		
CO_2 Emission/Day†	1.45 Kg		
Training System Memory	64 GB		
CPU	Intel® Xeon® W-2245 CPU @ 3.90GHz		

† Calculated using the ML CO_2 impact framework: https://mlco2.github.io/impact/ [32]

6 EXPERIMENTS

6.1 Results on MNIST

Fig. 4 presents the results of optimizing AlexNet trained on the MNIST dataset using SARAF against three different adversarial attacks including BIM, FGSM, and PGD. AlexNet with ReLU activation functions is selected as the compression baseline (□). SARAF (◯) significantly outperforms the default configuration by providing up to 33.07%, 15.09%, and 28.83% higher accuracy over BIM, FGSM, and PGD attacks, respectively.

6.2 Results on CIFAR-10

Fig. 5 presents the results of optimizing AlexNet trained on the CIFAR-10 dataset using SARAF against three different adversarial attacks including BIM, FGSM, and PGD. AlexNet with ReLU activation functions is selected as the compression baseline (□). SARAF (◯) remarkably outperforms the default configuration by providing up to 6.86%, 8.72%, and 11.14% higher accuracy over BIM, FGSM, and PGD attacks, respectively.

Fig. 6 presents the results of optimizing ResNet-18 trained on the CIFAR-10 dataset using SARAF against three different adversarial attacks including BIM, FGSM, and PGD. ResNet-18 with ReLU activation functions is selected as the compression baseline (□). SARAF (◯) significantly outperforms the default configuration by providing up to 16.92%, 18.3%, and 15.57% higher accuracy over BIM, FGSM, and PGD attacks, respectively. The search step takes up to ≈6 GPU days on a single NVIDIA® RTX A4000 for ResNet-18 trained on the CIFAR-10 dataset.

6.3 Results of Search Convergence

Fig. 7 plots the simulated annealing *Energy* function (Eq. 4) across search iterations. Our proposed search method finds activation functions with a monotonic decrease in *Energy*, indicating SARAF leads to a higher accuracy with more search iterations. We also present an empirical evaluation of SA compared to random search to show the superior optimization performance of SA. Random search is able to find the optimal architecture in many applications [35, 57]. However, as shown in Fig. 7, SA reached the highest global values for the *Energy* function (Eq. 4). Therefore, simulated annealing succeeds to find a feasible solution in a reasonable time.

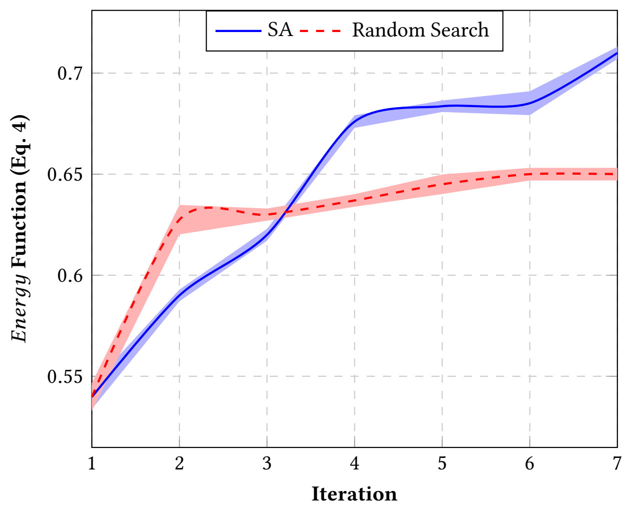

Figure 7: Demonstrating the reproducibility of results (STDEV=6.7%).

6.4 Discrimination Power of SARAF

We use t-distributed stochastic neighbor embedding (t-SNE) method [55] for visualizing the decision boundaries of the original ResNet-18, ResNet-18 with perturbation, and SARAF for the FGSM attack ($\epsilon = 10/255$) on the CIFAR-10 dataset. Fig. 8 illustrates the decision boundaries of classification for each scenario. According to the results, SARAF has a higher discrimination power than ResNet-18 with perturbation, and SARAF behaves similarly to the original ResNet-18.

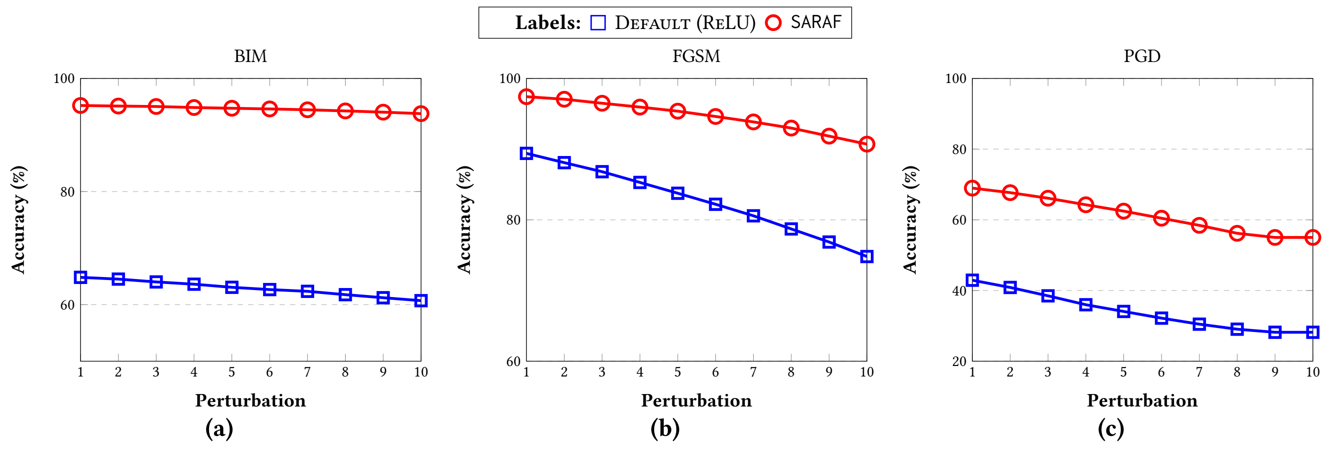

Figure 4: Comparison of AlexNet accuracy trained on MNIST using □ ReLU AFs and AFs searched by ○ SARAF against (a) BIM, (b) FGSM, and (c) PGD adversarial attacks.

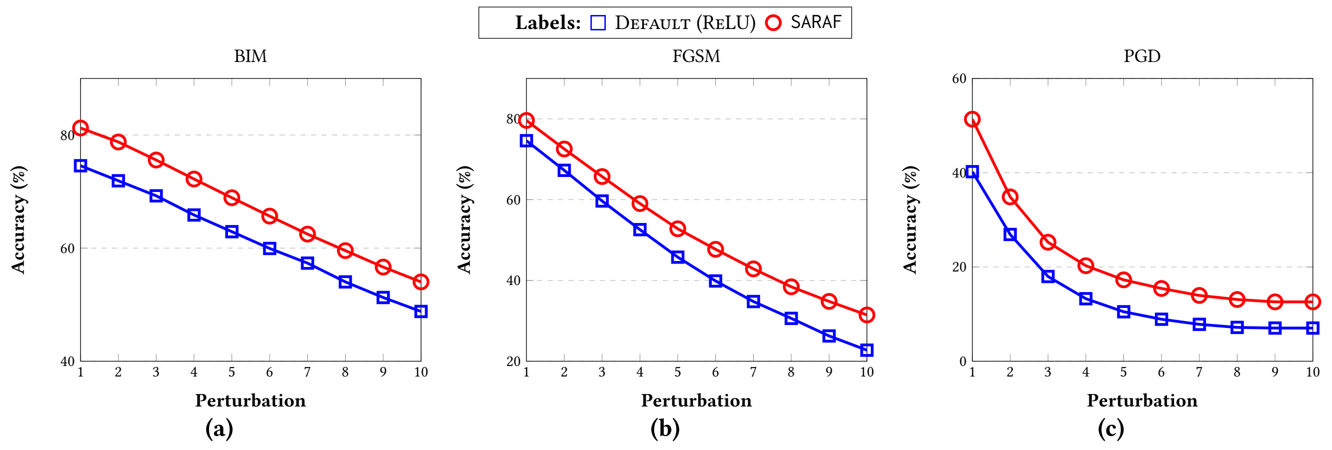

Figure 5: Comparison of AlexNet accuracy trained on CIFAR-10 using □ ReLU AFs and AFs searched by ○ SARAF against (a) BIM, (b) FGSM, and (c) PGD adversarial attacks.

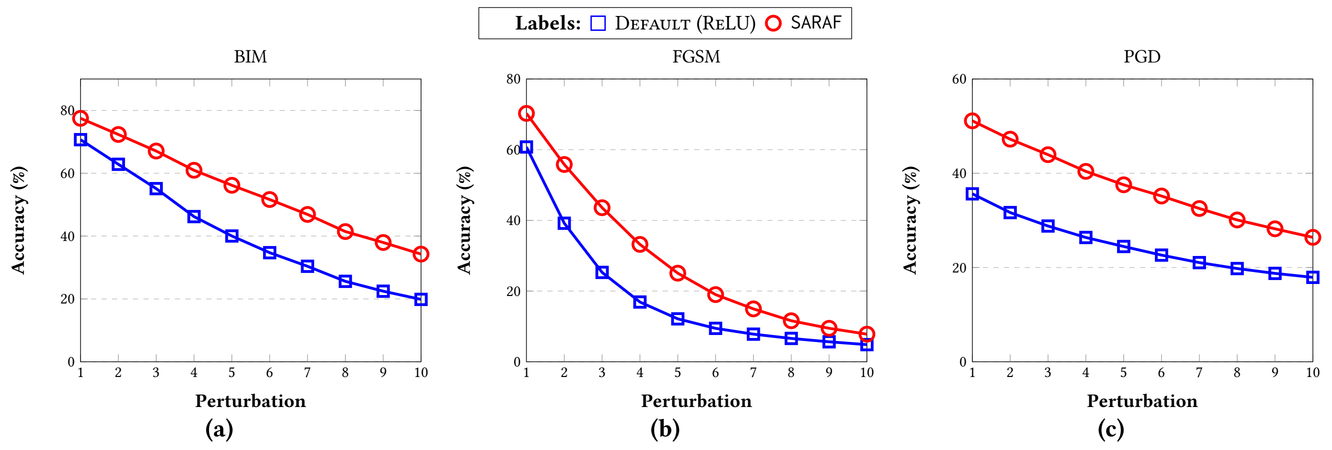

Figure 6: Comparison of ResNet-18 accuracy trained on CIFAR-10 using □ ReLU AFs and AFs searched by ○ SARAF against (a) BIM, (b) FGSM, and (c) PGD adversarial attacks.

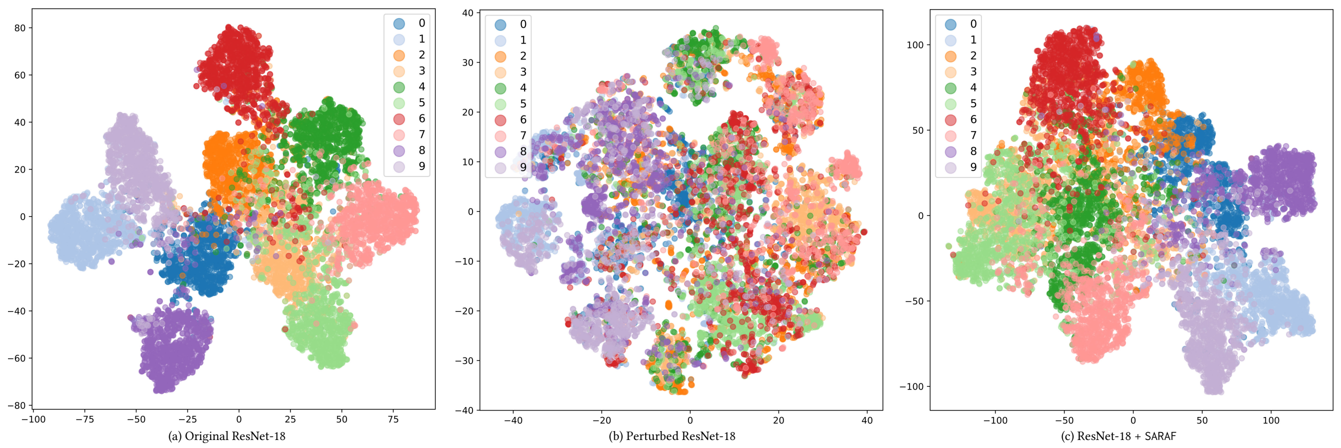

Figure 8: Visualizing the decision boundary with t-SNE embedding method for (a) original ResNet-18 without perturbation, (b) perturbed ResNet-18, and (c) ResNet-18 with SARAF optimization.

6.5 Visualizing Feature Map Results

Fig. 9 shows a sample feature map of ResNet-18 trained on CIFAR-10 with (i) ReLU AFs and (ii) AFs searched by SARAF against different adversarial attacks. As shown, feature maps extracted by SARAF AFs are more visible, indicating SARAF is robust against adversarial attacks (up to 33.07% accuracy improvement).

6.6 Reproducibility Statement

- **Code release.** SARAF is an open-source project. The code is made available on the GitHub repository through https://github.com/RobustInsight/SARAF.
- **Availability of database.** In this study, we evaluated our networks using the MNIST and CIFAR-10 datasets. Thus, this work does not involve any new data collection or human subject evaluation.
- **Reproducibility Analysis.** Many works on Automated Machine Learning had issues regarding reproducibility due to intrinsic stochasticity [37]. We re-ran the SARAF search procedure three more times with different random seeds to verify the reproducibility of the results. Fig. 7 shows the average of *Energy* function variations as well as the confidence intervals. Results show that the average of multiple runs converges to AFs with similar results with the standard deviation (STDEV) of 6.7%.

7 CONCLUSION

In this work, we studied the impact of optimizing activation functions on the robustness of CNNs against adversarial attacks. We found that ReLU is not robust in adversarial attacks, but optimizing network AFs significantly improves robustness over various attacks. Experimental results demonstrate the importance of activation functions in adversarial training and the potential of AFs for enhancing the robustness of deep learning models against adversarial examples.

REFERENCES

[1] Alireza Bahramali, Milad Nasr, Amir Houmansadr, Dennis Goeckel, and Don Towsley. 2021. Robust adversarial attacks against DNN-based wireless communication systems. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 126–140.

[2] Garrett Bingham, William Macke, and Risto Miikkulainen. 2020. Evolutionary optimization of deep learning activation functions. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*. 289–296.

[3] Garrett Bingham and Risto Miikkulainen. 2020. Discovering parametric activation functions. *arXiv preprint arXiv:2006.03179* (2020).

[4] Wieland Brendel, Jonas Rauber, and Matthias Bethge. 2017. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. *arXiv preprint arXiv:1712.04248* (2017).

[5] Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. 2018. Adversarial attacks and defences: A survey. *arXiv preprint arXiv:1810.00069* (2018).

[6] Ning Chen, Bernardete Ribeiro, Armando Vieira, Joao Duarte, and Joao C Neves. 2011. Incorporate cost matrix into learning vector Quantization modeling: A comparative study of genetic algorithm, simulated annealing and particle swarm optimization. *International Journal of Computer Theory and Engineering* 3, 1 (2011), 122.

[7] Xiangyu Chen, Ying Qin, Wenju Xu, Andrés M Bur, Cuncong Zhong, and Guanghui Wang. 2022. Explicitly Increasing Input Information Density for Vision Transformers on Small Datasets. *arXiv preprint arXiv:2210.14319* (2022).

[8] Peiyu Cui, Boris Shabash, and Kay C Wiese. 2019. Evodnn-an evolutionary deep neural network with heterogeneous activation functions. In *2019 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2362–2369.

[9] Sihui Dai, Saeed Mahloujifar, and Prateek Mittal. 2022. Parameterizing activation functions for adversarial robustness. In *2022 IEEE Security and Privacy Workshops (SPW)*. IEEE, 80–87.

[10] Nilesh Dalvi, Pedro Domingos, Sumit Sanghai, and Deepak Verma. 2004. Adversarial classification. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. 99–108.

[11] Manyu Dhyani and Rajiv Kumar. 2021. An intelligent Chatbot using deep learning with Bidirectional RNN and attention model. *Materials today: proceedings* 34 (2021), 817–824.

[12] Samuel G Finlayson, John D Bowers, Joichi Ito, Jonathan L Zittrain, Andrew L Beam, and Isaac S Kohane. 2019. Adversarial attacks on medical machine learning. *Science* 363, 6433 (2019), 1287–1289.

[13] Behnam Ghavami, Seyd Movi, Zhenman Fang, and Lesley Shannon. 2022. Stealthy Attack on Algorithmic-Protected DNNs via Smart Bit Flipping. In *2022 23rd International Symposium on Quality Electronic Design (ISQED)*. IEEE, 1–7.

[14] Luke B Godfrey and Michael S Gashler. 2015. A continuum among logarithmic, linear, and exponential functions, and its potential to improve generalization in neural networks. In *2015 7th international joint conference on knowledge discovery, knowledge engineering and knowledge management (IC3K)*, Vol. 1. IEEE, 481–486.

[15] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).

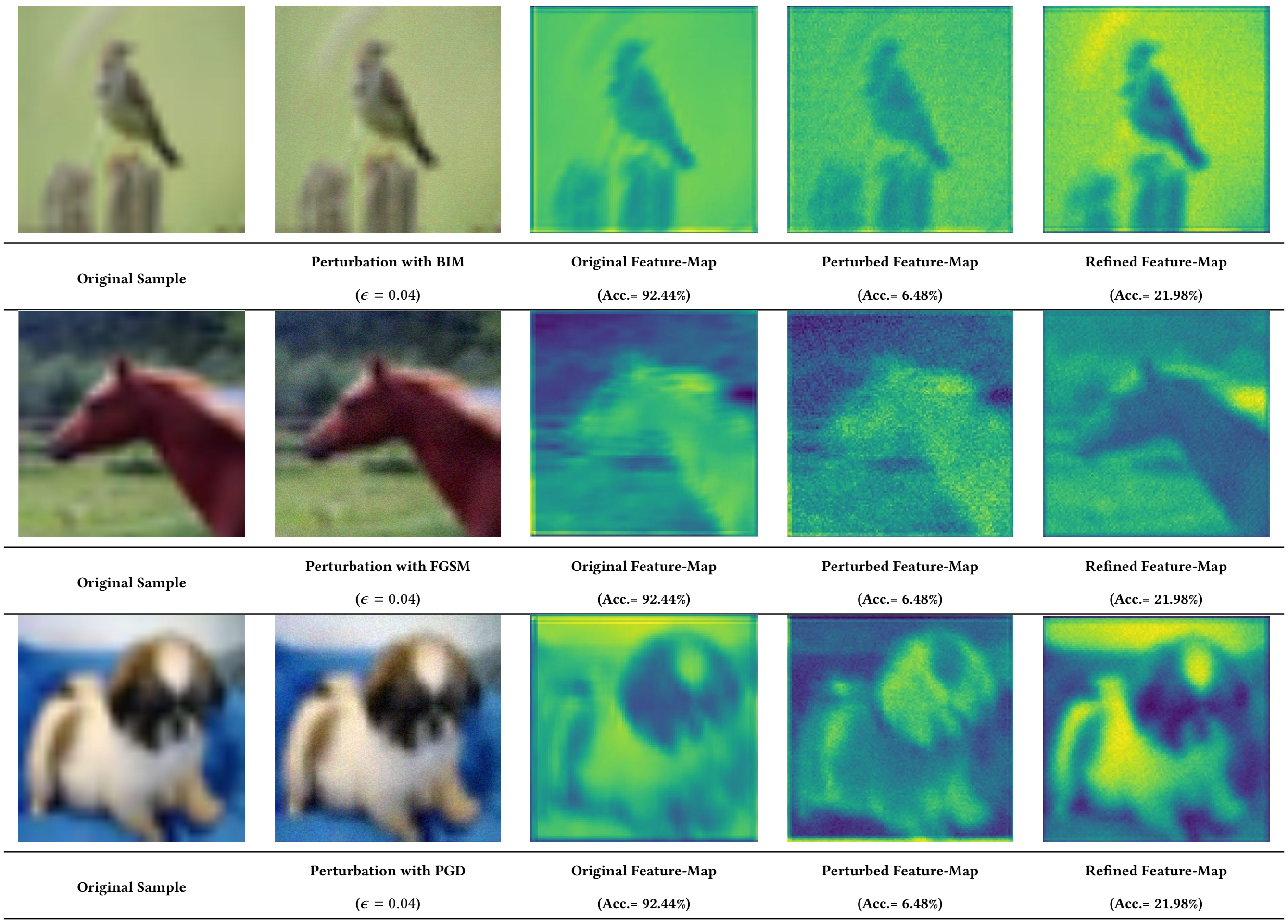

Figure 9: Visualizing a sample feature map of ResNet-18 with/without AF searched by SARAF against different adversarial attacks. As shown, AFs searched by SARAF compensate for the negative impact of perturbation on feature maps.

[16] Sven Gowal, Sylvestre-Alvise Rebuffi, Olivia Wiles, Florian Stimberg, Dan Andrei Calian, and Timothy A Mann. 2021. Improving robustness using generated data. *Advances in Neural Information Processing Systems* 34 (2021), 4218–4233.
[17] Vincent Granville, Mirko Krivánek, and J-P Rasson. 1994. Simulated annealing: A proof of convergence. *IEEE transactions on pattern analysis and machine intelligence* 16, 6 (1994), 652–656.
[18] Hayit Greenspan, Bram Van Ginneken, and Ronald M Summers. 2016. Guest editorial deep learning in medical imaging: Overview and future promise of an exciting new technique. *IEEE transactions on medical imaging* 35, 5 (2016), 1153–1159.
[19] Richard HR Hahnloser, Rahul Sarpeshkar, Misha A Mahowald, Rodney J Douglas, and H Sebastian Seung. 2000. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *nature* 405, 6789 (2000), 947–951.
[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition.* 770–778.
[21] Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415* (2016).
[22] Junhao Huang, Bing Xue, Yanan Sun, Mengjie Zhang, and Gary G Yen. 2022. Particle Swarm Optimization for Compact Neural Architecture Search for Image Classification. *IEEE Transactions on Evolutionary Computation* (2022).
[23] Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. 2017. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284* (2017).
[24] Xiaojie Jin, Chunyan Xu, Jiashi Feng, Yunchao Wei, Junjun Xiong, and Shuicheng Yan. 2016. Deep learning with s-shaped rectified linear activation units. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 30.
[25] M Singh Kamboj and Jyotsna Sengupta. 2009. Comparative analysis of Simulated Annealing and tabu search channel allocation algorithms. *International Journal of Computer Theory an Engineering* 1, 5 (2009), 1793–8201.
[26] Qiyu Kang, Yang Song, Qinxu Ding, and Wee Peng Tay. 2021. Stable neural ode with lyapunov-stable equilibrium points for defending against adversarial attacks. *Advances in Neural Information Processing Systems* 34 (2021), 14925–14937.
[27] Scott Kirkpatrick, C Daniel Gelatt Jr, and Mario P Vecchi. 1983. Optimization by simulated annealing. *science* 220, 4598 (1983), 671–680.
[28] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. 2009. Cifar-10 and cifar-100 datasets. *URl: https://www. cs. toronto. edu/kriz/cifar. html* 6, 1 (2009), 1.
[29] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger (Eds.), Vol. 25. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf
[30] Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *International conference on machine learning*. PMLR, 1378–1387.
[31] Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. 2018. Adversarial examples in the physical world. In *Artificial intelligence safety and security.* Chapman and Hall/CRC, 99–112.
[32] Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. 2019. Quantifying the carbon emissions of machine learning. *arXiv preprint arXiv:1910.09700* (2019).
[33] Yann LeCun. 1998. The MNIST database of handwritten digits. *http://yann. lecun. com/exdb/mnist/* (1998).

[34] Mark Lee and Zico Kolter. 2019. On physical adversarial patches for object detection. *arXiv preprint arXiv:1906.11897* (2019).

[35] Liam Li and Ameet Talwalkar. 2020. Random search and reproducibility for neural architecture search. In *Uncertainty in artificial intelligence*. PMLR, 367–377.

[36] Yanfei Li, Tong Geng, Samuel Stein, Ang Li, and Huimin Yu. 2022. GAAF: Searching Activation Functions for Binary Neural Networks through Genetic Algorithm. *arXiv preprint arXiv:2206.03291* (2022).

[37] Marius Lindauer and Frank Hutter. 2020. Best practices for scientific research on neural architecture search. *Journal of Machine Learning Research* 21, 243 (2020), 1–18.

[38] Mohammad Loni, Sima Sinaei, Ali Zoljodi, Masoud Daneshtalab, and Mikael Sjödin. 2020. DeepMaker: A multi-objective optimization framework for deep neural networks in embedded systems. *Microprocessors and Microsystems* 73 (2020), 102989.

[39] Mohammad Loni, Ali Zoljodi, Daniel Maier, Amin Majd, Masoud Daneshtalab, Mikael Sjödin, Ben Juurlink, and Reza Akbari. 2020. Densedisp: Resource-aware disparity map estimation by compressing siamese neural architecture. In *2020 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 1–8.

[40] Mohammad Loni, Ali Zoljodi, Amin Majd, Byung Hoon Ahn, Masoud Daneshtalab, Mikael Sjödin, and Hadi Esmaeilzadeh. 2021. Faststereonet: A fast neural architecture search for improving the inference of disparity estimation on resource-limited platforms. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* (2021).

[41] Mohammad Loni, Ali Zoljodi, Sima Sinaei, Masoud Daneshtalab, and Mikael Sjödin. 2019. Neuropower: Designing energy efficient convolutional neural network architecture for embedded systems. In *International conference on artificial neural networks*. Springer, 208–222.

[42] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083* (2017).

[43] Polina Mamoshina, Armando Vieira, Evgeny Putin, and Alex Zhavoronkov. 2016. Applications of deep learning in biomedicine. *Molecular pharmaceutics* 13, 5 (2016), 1445–1454.

[44] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems* 26 (2013).

[45] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. 2016. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2574–2582.

[46] Najmeh Nazari, Mohammad Loni, Mostafa E Salehi, Masoud Daneshtalab, and Mikael Sjodin. 2019. Tot-net: An endeavor toward optimizing ternary neural networks. In *2019 22nd Euromicro Conference on Digital System Design (DSD)*. IEEE, 305–312.

[47] Arash Rahnama, Andre T Nguyen, and Edward Raff. 2020. Robust design of deep neural networks against adversarial attacks based on lyapunov theory. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8178–8187.

[48] Prajit Ramachandran, Barret Zoph, and Quoc V Le. 2017. Searching for activation functions. *arXiv preprint arXiv:1710.05941* (2017).

[49] Sylvestre-Alvise Rebuffi, Sven Gowal, Dan A Calian, Florian Stimberg, Olivia Wiles, and Timothy Mann. 2021. Fixing data augmentation to improve adversarial robustness. *arXiv preprint arXiv:2103.01946* (2021).

[50] Ishai Rosenberg, Asaf Shabtai, Yuval Elovici, and Lior Rokach. 2021. Adversarial machine learning attacks and defense methods in the cyber security domain. *ACM Computing Surveys (CSUR)* 54, 5 (2021), 1–36.

[51] Maghsood Salimi, Amin Majd, Mohammad Loni, Tiberiu Seceleanu, Cristina Seceleanu, Marjan Sirjani, Masoud Daneshtalab, and Elena Troubitsyna. 2019. Multi-objective optimization of real-time task scheduling problem for distributed environments. In *Proceedings of the 6th Conference on the Engineering of Computer Based Systems*. 1–9.

[52] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *Advances in neural information processing systems* 27 (2014).

[53] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199* (2013).

[54] Mohammadamin Tavakoli, Forest Agostinelli, and Pierre Baldi. 2021. Splash: Learnable activation functions for improving accuracy and adversarial robustness. *Neural Networks* 140 (2021), 1–12.

[55] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).

[56] Cihang Xie, Mingxing Tan, Boqing Gong, Alan Yuille, and Quoc V Le. 2020. Smooth adversarial training. *arXiv preprint arXiv:2006.14536* (2020).

[57] Antoine Yang, Pedro M Esperança, and Fabio M Carlucci. 2019. NAS evaluation is frustratingly hard. *arXiv preprint arXiv:1912.12522* (2019).

[58] Valentina Zantedeschi, Maria-Irina Nicolae, and Ambrish Rawat. 2017. Efficient defenses against adversarial attacks. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. 39–49.

[59] Huan Zhang, Shiqi Wang, Kaidi Xu, Yihan Wang, Suman Jana, Cho-Jui Hsieh, and Zico Kolter. 2022. A Branch and Bound Framework for Stronger Adversarial Attacks of ReLU Networks. In *International Conference on Machine Learning*. PMLR, 26591–26604.

[60] Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. 2018. Efficient neural network robustness certification with general activation functions. *Advances in neural information processing systems* 31 (2018).

[61] Jiliang Zhang and Chen Li. 2019. Adversarial examples: Opportunities and challenges. *IEEE transactions on neural networks and learning systems* 31, 7 (2019), 2578–2593.

[62] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. 2019. Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems* 30, 11 (2019), 3212–3232.

[63] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. 2018. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 8697–8710.