A deep dive into the robustness of AdaBoost Ensembling combined with Adversarial Training

Kanish Dwivedi
Supervisor(s): Chi Hong, Jiyue Huang, Stefanie Roos
EEMCS, Delft University of Technology, The Netherlands

June 19, 2022

A Dissertation Submitted to EEMCS faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering

Abstract

Adversarial training and its variants have become the standard defense against adversarial attacks - perturbed inputs designed to fool the model. Boosting techniques such as Adaboost have been successful for binary classification problems, however, there is limited research in the application of them for providing adversarial robustness. In this work, we explore the question: How can AdaBoost ensemble learning provide adversarial robustness to white-box attacks when the "weak" learners are neural networks that do adversarial training? We design an extension of AdaBoost to support adversarial training in a multiclass setting, and name it Adven. To answer the question, we systematically study the effect of six variables of Adven's training procedure on adversarial robustness. From a theoretical standpoint, our experiments show that known characteristics from adversarial training and ensemble learning apply in the combined context. Empirically, we demonstrate that an Adven ensemble is more robust than a single learner in every scenario. Using the best found values of the six tested variables, we derive an Adven ensemble that can defend against 91.88% of PGD attacks and obtain 96.72% accuracy on the MNIST dataset.

1 Introduction

Research has proven that neural networks are prone to adversarial attacks [4][9][8][16][17] [36][30]. Adversarial attacks are most prevalent for image classification problems. Specifically, an adversary (also known as perturbation) of an image is another image that is indistinguishable to the human eye but has different pixel values. The adversary is not random noise added to the original image, rather it is a systematic change [8][17]. There have been many forms of adversarial generation techniques discovered and explored by the community. These can be white-box (the attacker has all the information on the target models parameters) or black-box (the attacker has no information on target model parameter). To defend against these attacks, many approaches have been studied since 2014 when the problem was brought to light, but new powerful attacks could break them [4]. Adversarial training, where the model is trained on adversarial images, is one of the defense approaches that is yet to be defeated [4]. On its own, its not enough against strong state-of-the-art white-box adversarial attacks [9]. Rather, state-of-the-art defenses rely on combining different learning techniques to achieve robustness. Examples of successful approaches are combinations of specialised loss functions, activation functions, and model sizes.

Around the time adversarial attacks were discovered, ensemble learning (the combination of several classifiers for classification) as a defense technique was also tested [8]. Boosting ensemble learning combines weak learners (models with low accuracy) into a single strong learner. Adaptive Boosting (AdaBoost) [25] is a boosting algorithm that achieves this by assigning weights to the training dataset and to each individual model in the ensemble, such that each consecutive trained weak learner has a higher focus on the previous learners mistakes. Experiments conducted in 2014 found that an ensemble model failed to give any protection to adversarial attacks [8]. Since then, there has been limited research on using ensemble learning for adversarial defense, in particular, there is very little research into the combination of AdaBoost ensemble learning with adversarial training as a defense.

This work performs an in-depth study into the combination of AdaBoost ensemble learning with adversarial training. This allows us to test if ensemble learning is a viable defense strategy when combined with adversarial training, and analyse whether or not discovered properties of the two techniques transfer over in the combination. The following research

question is answered: *How can AdaBoost ensemble learning provide adversarial robustness to white-box attacks when the "weak" learners are neural networks that do adversarial training?*

We answer this question by fixing an ensemble learning algorithm, and thoroughly explore the *training* of the "weak" neural networks themselves, as it's closer linked to adversarial training. We did this by conducting many controlled experiments each having its own independent and controlled variables. The independent variables of the training procedure that have been tested are: (1) model size, (2) loss function, (3) activation function, (4) perturbation radii, (5) adversarial generation algorithm, and (6) number of weak learners. The dependent variable (evaluation metric) for all the experiments was: test set accuracy and adversarial robustness (% of defended attacks) to the PGD [17] white-box attack.

To conduct these experiments, we adapted the AdaBoost multiclass algorithm, SAMME [38], to support adversarial training with neural networks; we called it *Adven*. Our experiments showed Adven inheriting both known adversarial training and ensemble learning characteristics, whilst exhibiting unique ones of its own. It provided more robustness than a single learner in all the experiments, and the best combination of the tested variables obtained 96.72% test set accuracy and 91.88% PGD attack robustness on MNIST [5] dataset.

2 Background & Related Work

Before delving into our research approach, the following section will briefly introduce related work of adversarial training & attacks, and ensemble learning.

2.1 Adversarial Attacks & Training Research

Adversarial Attacks. Research by [26] observed that neural networks are vulnerable to adversarial attacks. The first white-box attack was Fast Gradient Sign Method (FGSM) adversarial generation algorithm [8]. This was followed by R+FGSM [27], and then by the Basic Iterative Method (BIM) [16]. In 2018, research by [17] introduced the Projected Gradient Descent (PGD) attack, which was the strongest at the time, and is the backbone for the current strongest attacks such as AutoAttack [4].

Adversarial Training as a defence. Adversarial training is a defence approach to adversarial attacks, wherein the model under attack trains on adversary examples [9][17][30][4]; it's regarded as one of the most successful methods to train robust deep neural networks [9]. It was first introduced in [8] where it was used as a regulariser against FGSM attacks; bringing up robustness by 10%. Most importantly, [17] formulated adversarial training as a saddle point problem whose goal is to find model parameters that minimize the adversarial risk. The saddle point problem is a composition of an inner maximisation and outer minimisation, where the inner maximisation is approximated using adversarial attacks, and the outer minimisation is approximated using adversarial training [9]. Since the success of this formulation, many papers have attempted adversarial training using it as a backbone. Experiments were conducted that explored: changes to the inner maximisation [7][4][30], effects of loss functions [12][19][36], effects of model architecture [33]. Of these mentioned approaches, research on the effects of loss functions has had eye-catching success. For example, TRADES loss [36] balances the trade-off between standard and robust accuracy via a specific regularisation term. Similarly, MART [29] found success by addressing the TRADES

trade-off by using boosted loss functions. In [9], a detailed systematic study into adversarial training by studying effects of loss functions, model sizes, activation functions, and many other factors was conducted. It was found that the proper combination of these factors can provide significant improvement to adversarial robustness to state-of-the-art attacks.

2.2 Boosting Ensemble Research

Boosting is an ensemble machine learning technique that was first introduced in [25] for a binary classification task, AdaBoost was the name given to the implemented algorithm. AdaBoost is an iterative algorithm that integrates multiple "weak" classifiers for voting with weights to create one high performing "strong" classifier [13][35][11]. In 2006, two multi-class variations of AdaBoost were formulated called SAMME and SAMME.R [38]. AdaBoost is most commonly used with classifiers like decision trees, but there has been limited research into the effectiveness of AdaBoost with neural networks [13]. In 2014, research by [11] found an AdaBoost neural network ensemble improving accuracy by 10% over a single neural network for an image classification problem. However, three years later in 2017, research by [35] found that simply using AdaBoost with neural networks as weak learners has several notable weaknesses. However, an alternative training regime that makes uses of pretraining and special class weights overcame these issues [35]. In the research community, there has been very limited research into the effects of AdaBoost to adversarial robustness. Research conducted by [13] showed an ensemble providing slightly greater robustness to PGD attacks. However, the focus of the research was more on the ensemble algorithm itself, and no in-depth research was conducted on adversarial training and its related parameters.

3 Methodology and Problem Description

The following section describes the method used to evaluate how AdaBoost ensembling combined with adversarial training can provide adversarial robustness. The conducted experiments will vary multiple different variables of the training procedure in an attempt to explore how characteristics of these two techniques influence adversarial robustness. This section starts with the motivation for the chosen method, followed by a brief description of adversarial attacks and training, followed by a brief description of AdaBoost ensembling and our adaptation of it, and finally ends with explaining the experimental procedure.

3.1 Method Motivation

The combination of boosting ensemble techniques like AdaBoost and adversarial training has not been thoroughly researched. Our method of exploring multiple training variables of the combination is of interest because, this method lets us: (1) analyse if adversarial training is viable in another context like ensembling and vice-versa, (2) confirm whether or not previous findings related to adversarial training and ensembling transfer over in this combined context, (3) answer whether ensembling works well with neural networks, (4) gain insight into whether the chosen variables are relevant for adversarial training or ensembling.

3.2 Adversarial Attacks & Training

Before describing how we conducted the experiments, its important to describe the basics of adversarial attacks and adversarial training as a defense, and define related terms.

Target Model(s): are CNNs that are performing a particular classification task. The attackers aim is to minimize this models accuracy by feeding it perturbed images that would lead the model to missclassify the image that it would otherwise correctly classify.

Adversarial Attack. This research limits the focus to untargeted white-box gradient-based attacks. White-box setting means the attacker has access to all the model parameters. Untargeted refers to the setting that the attacker creates adversaries without the aim of fooling the classifier for a specific class. Gradient refers to the direction that increases the loss function; whilst training, a model alters its parameters (weights and biases) using the *negative* gradient to *minimize* the loss function, this is known as back-propagation. Gradient-based attacks refers to the setting wherein the attacker perturbs the image towards the direction that instead *maximises* the loss function such that the image is missclassified [8]. The FGSM attack [8] (Equation 1) is the most basic gradient-based attack that captures this intuition: perturbation (adversary) is created by moving towards the direction (∇) that maximizes the loss function L, ϵ determines how much we perturb the image.

$$x_{adv} = x + \delta^*, \text{ where } \delta^* = \epsilon \cdot (\nabla_x L(x, y)) \tag{1}$$

Equation 2 shows the PGD attack [17]. This attack is a far stronger generalisation of the FGSM attack, that computes perturbations by taking K gradient steps of size η; after each step, the perturbed image is projected back onto set of allowed perturbations A.

$$x_{adv} = x + \delta^K, \text{ where } \delta^{(k+1)} = \Pi_A(\delta^{(k)} + \eta \cdot \text{sign}(\nabla_{\delta^{(k)}} L(f_\theta(x + \delta^{(k)}), y))) \tag{2}$$

A typical choice for A is the l_∞ perturbations, defined as $A = \{\delta : ||\delta||_\infty \leq \epsilon\}$ where ϵ is called the perturbation radii, and δ^0 is often randomly initialised within A.

Adversarial Training: is a method for learning models that are robust to adversarial attacks [9]. Paper [17] formulated this concept mathematically as a saddle-point problem. Given a model (CNN) f_θ parameterised by θ, a dataset (x_i, y_i), a loss function L, and set of allowed perturbations A (eg. l_∞), the adversarial training problem is to minimize:

$$\underbrace{\min_\theta \sum_i \underbrace{\max_{\delta \in A} L(f_\theta(x_i + \delta), y_i)}_{\text{inner maximization}}}_{\text{outer minimization}} \tag{3}$$

The procedure for adversarial training is to use some adversarial generation algorithm (i.e attacks like PGD or FGSM) to approximate the inner maximisation, approximations are used as the problem is NP-HARD [9]. The outer maximization is achieved by some form of gradient descent on the model parameters θ [30]. In summary, adversarial training aims to find the model parameters θ that minimize the loss incurred by the maximal perturbed image generated using adversarial attacks i.e learn the strongest perturbation possible.

Adversarial Robustness: is a measure to see how well a model defends against adversarial attacks. Specifically, for this paper, robust accuracy and synonymously robustness is the percentage of adversarial attacks that failed. The higher the robustness the better.

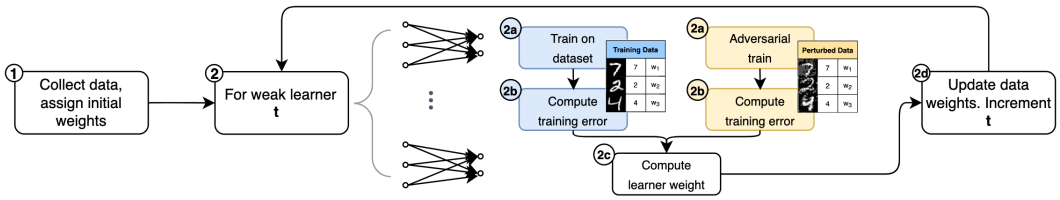


Figure 1: SAMME & Adven algorithm overview. (1) Dataset is collected and assigned weights. (2) For each weak learner in the ensemble: (2a) train it, (2b) compute its errors, (2c) assign learner weight, (2d) update data weights. (Blue) SAMME trains and computes errors on clean images whereas (Yellow) Adven does so with perturbed images.

3.3 AdaBoost Ensembling

For the purposes of this work, the classic AdaBoost ensembling algorithm cannot be used as it is designed for a binary classification task. Figure 1 provides an overview of the SAMME algorithm [38], a multi-class extension of the AdaBoost algorithm. The ensemble training procedure begins with initialising equal sample weights for each training example. The next step is to train individual weak learners iteratively. During each iteration, once the weak learner has finished training on the weighted training data, its training error is computed (Blue: Step 2a-b). This is used to assign a weight to the learner (Step 2c), the basic idea is that the better the learner performed the higher its weight. Finally, the data weights are updated based on how this learner performed on them (Step 2d). The rule is that examples that were missclassified are given a larger weight, such that the next weak learner can focus on learning these better. To make an ensembled classification for a given input, SAMME first collects the individual classifications of each weak learner and scales them according to the learner's weight, then selects the most popular class. SAMME cannot directly be used for adversarial training with neural networks. We modified it to enable adversarial training with neural networks, and we called it **Adven** (mix of the words ADVersarial and ENsembling). Figure 1 highlights two important differences between Adven and SAMME. First, each weak learner in Adven are CNNs that perform adversarial training by perturbing training images using some adversary generation algorithm (Yellow: Step 2a). Second, the weak learner training error in Adven is computed using adversarial images rather than clean images (Yellow: Step 2b), this is then used to compute the learner weight and update the sample weights in the same way as SAMME (Steps 2c-d). These two differences enable ensembled adversarial defense. Because each weak learner has trained on adversarial images, and each consecutive learner puts more focus into learning adversarial images that the previous learner could not defend. Finally, Appendix A contains detailed pseudo-code for both algorithms.

3.4 Experimental Procedure

As mentioned at the beginning of this section, our experiments varied different variables of the Adven training procedure to explore how they influenced robustness.

Variables and motivation. Extensive experiments were conducted for six variables: (1) Adversarial generation algorithm used during adversarial training by a weak learner, (2) Loss function used by a weak learner, (3) Perturbation radii used during adversarial training by a weak learner, (4) total number of weak learners in the ensemble, (5) Activation

function used by a weak learner, and (6) depth of the weak learner (model size).

The first three variables are an important choice because they correspond with the outer maximisation and inner minimisation part of the adversarial training saddle point formulation; exploring them will give insights into adversarial training characteristics in an ensemble context. Exploring the total number of learners lets us better understand the effect of ensembling on robustness. Activation function and model size lets us examine whether or not existing findings on them are observed in an ensemble context. There are many other variables that can be explored, and we encourage the research community to explore them.

Experiment Overview. For each of the six independent variables outlined, we did the following: (1) Data preparation, (2) Select variable value, (3) Train Adven ensemble with the selected value, while keeping all other variables constant, (4) Measure the accuracy of test dataset, (5) Measure the robustness against adversarial attack, (6) Repeat steps 3-5 three times for statistical significance, (7) Repeat steps 2-6 for all values of the current independent variable. To ensure generalisability, the two values resulting in the greatest and least robustness were chosen, then experiments were conducted using the experimental procedure stated above on a *different* dataset. Finally, after all experiments, the best combination of these values were found that can provide the maximal adversarial robustness.

4 Experimental Setup

This section states the specifics of how the experiments were carried out. We provide details for the steps that were outlined in the experimental overview part of Section 3.4.

1. Dataset & Data preparation. All the experiments used the MNIST [5] dataset. However, we realised that patterns and observations can be dependent to the dataset used in the experiments, thus to ensure generalisability we tested our findings on another dataset [9]. We conducted additional tests on the Fashion-MNIST [31] dataset. Specifically: two values for each independent variable were tested on this other dataset, we then checked if patterns on adversarial robustness matched with what was observed on the MNIST dataset. If this was the case, we could deem our findings more reliable (i.e. not dataset dependent).

2. Independent variable values. The following are the values for the independent variables that have been tested, and the motivation for why they were picked:

- **Model Size:** 2, 3, 4, 5. We stop at CNN having 5 layers, as any larger will result in the CNN no longer being a "weak" learner anymore [35].
- **Activation Function:** ReLU [1], HardTanh [23], Leaky ReLU [34], ELU [3], and GeLU [10]. Most of these are similar to the activation functions tested by [9], the others are common in the machine learning community.
- **Loss Function:** Cross Entropy[37], KL-divergence, and TRADES [36]. TRADES was selected as it has shown good success already, the other two because they are commonly used and are part of TRADES' formulation.
- **Perturbation Radii:** $\epsilon = \{8/255, 18/255, 28/255, 38/255, 48/255, 58/255, 68/255, 78/255\}$. These values were chosen because interesting patterns were observed during preliminary tests (see Appendix B) and other research papers have often used similar values for their experiments [9].

- **Adversarial Generation Algorithm:** FGSM [8], BIM [16], PGD [17], MIA [7]. These attacks were chosen because they offer a good variety of strength.
- **Number of weak learners:** 2, 3, 4, 5, 6, 7, 10, 20, 30, 40. The first half of tests are quite granular up to 7 weak learners, this is similar to the study by [35]. The larger number of learners are there because similar researched papers, [13][35][11], have not tested this large number of learners.

Furthermore, for each of the independent variables the following **control/default values** are set: (1) Model size: 3 layer CNN, (2) Loss function: weighted cross-entropy loss (3) Activation Function: ReLU (4) Perturbation Radii: 78/255 (5): Adversarial Generation Algorithm: PGD ($\epsilon = 78/255$, number of iterations = 20) (6) Number of weak learners: 7. These default values are used in-place for the independent variable when its not being tested.

3. Adven training details. When training a weak learner in Adven, the following parameters were kept constant for *all* experiments: batch size = 100, number of epochs = 3, learning rate = 0.003 with Adam [14] optimizer, adversary algorithm used l_∞ perturbations. Usually loss functions are computed by taking a batch mean, our loss functions computes the *weighted* batch mean (the weights are the ensemble training data weights).

4. Evaluation protocol. Two evaluation metrics were used. First was the test set accuracy (% of data correctly classified), and the second was the robustness to PGD attack (% of unsuccessful attacks) with the following parameters: $\epsilon = 78/255 = 0.3058$, number of iterations = 20, and step size = 0.003. The PGD attack itself is created by perturbing the test dataset based on the parameters of the CNN with the greatest weight in the ensemble. We also measured how long a single learner took to train, letting us get an idea of computational efficiency of Adven. We used a single machine with one GeForce RTX 2060.

5 Results

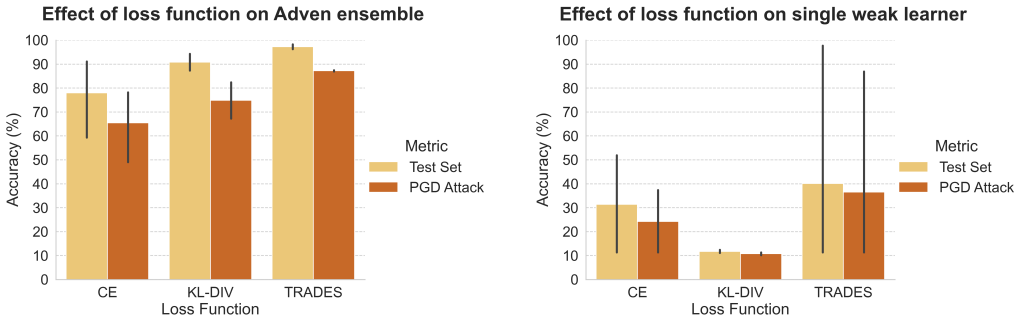
The following sub-sections will detail all the experiments conducted for the six independent variables. Each sub-section begins with context to explain any relevant related works or explain required concepts, then our results are presented and explained.

5.1 Loss Function

Context. In Section 3.2, Equation 3 describes adversarial training as solving a saddle point problem: attempting to minimize the loss incurred by the maximal perturbed image. Cross-entropy is a common choice for the used loss function [7]. Of the many papers that have explored other loss functions, TRADES [36] is noteworthy due to its characteristic of balancing the trade-off between standard and robust accuracy; helping the model learn perturbations effectively. It was used in [9] to create one of the state-of-the-art defenses.

Results. Figures 2a and 2b show the effect of changing the *loss functions* used by adversarial training on the *test-set accuracy* and *robustness* for an Adven ensemble and single learner respectively. Figure 2b shows that for a single learner, TRADES loss provides the highest robustness, upwards to 84%, this is in line with similar conducted research [9]. The confidence intervals for TRADES and cross-entropy show that the observed robustness varies a lot, we suspect this is either due to the low capacity of weak learner not being able to learn

the perturbations or some training batch shuffles being harder to learn than others. KL-divergence consistently provided very little robustness for a single weak learner. This loss compares the probability distributions between the models predictions and the true label. In implementation, the true label is one-hot encoded (each position of an array represents a class; a label is converted into this array, a 1 is filled at the correct position, and 0 elsewhere). Due to this encoding, on missclassifications the incurred loss is high and on correct classifications the loss is low, and rarely in-between. We suspect due to this, the loss space is rather rigid with many troughs and peaks, making it difficult for the optimiser to traverse it and back-propagate. Figure 2a shows that for all loss functions, an Adven ensemble provided better robustness than a single learner, and a lot more consistently (smaller confidence intervals). We hypothesise the weighted loss functions that the weak learners in Adven use, help the learners better learn perturbations. Specifically, a learner incurs a high loss when it missclassifies a perturbation that the previous learners also missclassified, and a low loss when it correctly classifies it. The resulting effect is that the learner focuses on learning a subset of the data, namely, perturbations the previous learner couldn't. This is helpful as weak learners have low capacity [17]. Finally, we observed training a single learner on cross-entropy and KL-divergence loss took 104 seconds, whereas training on TRADES took 150 seconds. Therefore, gained robustness from TRADES is not computationally "free".



(a) Adven ensemble

(b) Single learner

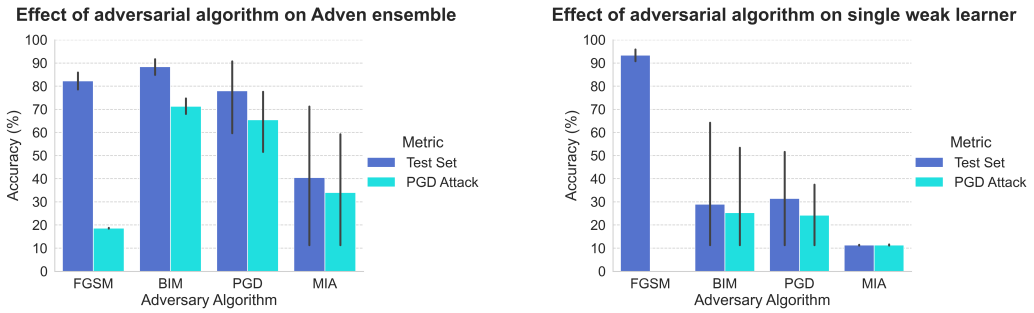
Figure 2: Loss Function Test | Test set accuracy and PGD robustness accuracy for different loss functions used during adversarial training. Left (2a) shows this for an Adven Ensemble. Right (2b) for a single learner. Error bars represent 95% confidence intervals.

5.2 Adversarial Generation Algorithm

Context. As explained in Section 3.2, adversarial training for deep neural networks proposed in [17] aims to solve a saddle point problem. The inner maximisation part of the problem is *approximated* by gradient-based adversarial generation algorithms. Algorithms like FGSM and PGD as explained in Section 3.2 can be used. However, there exists many others, such as the Basic Iterative Method (BIM) [16] and Momentum Iterative Attack (MIA) [7]. BIM is only a slightly weaker attack in comparison to PGD because it does not make use of random initialisation. MIA is a one-step attack like FGSM, but it accumulates the gradient for each perturbed image and uses it to better maximize the loss function [7].

Results. Figures 3a and 3b show the effect of changing the *adversary algorithm* used

by adversarial training on the *test accuracy* and *robustness* for an Adven ensemble and a single learner respectively. Figure 3b shows two known characteristics of a single learner performing adversarial training. First, training on FGSM gives no robustness to a more sophisticated attacks like PGD but training on stronger attacks like PGD does [17]. Second, we observe the trade-off effect [36]: training on complex attacks requires sacrificing more on the more test set accuracy. Figure 3a shows that these characteristics are also observed in an ensemble setting. Moreover, it displays that the ensembled model on average is more than two times robust to PGD attacks in comparison to a single learner, whilst maintaining high accuracy on the test set. This finding may possibly indicate that an ensembled model is more resistance to the trade-off effect. One interesting observation in Figure 3a is that, on average, training on BIM provides better robustness to PGD attacks than training on PGD. Because the individual learners in the ensemble are weak, they can better learn the slightly weaker BIM attacks that PGD; Figure 3b shows this with BIM having a larger confidence interval than PGD. Since BIM and PGD with 20 iterations create very similar perturbations, the ensemble trained on BIM can provide good robustness to PGD attacks. Finally, we found training a single learner on FGSM took 33 seconds, whereas training on the others took 104 seconds. Therefore, gained robustness is not computationally "free".



(a) Adven ensemble

(b) Single learner

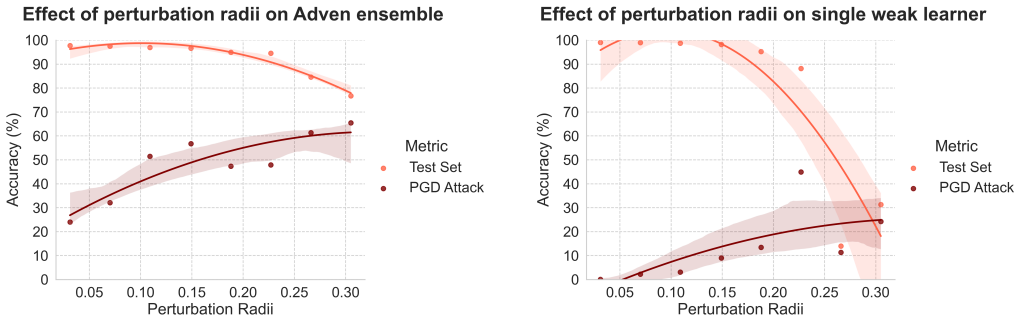
Figure 3: Adversary Algorithm Test | Test set accuracy and PGD robustness accuracy observed when different adversary algorithms are used during adversarial training. Left (3a) shows this for an Adven ensemble. Right (3b) for a single learner. Error bars represent 95% confidence interval.

5.3 Perturbation Radii

Context. As stated in Section 3.2, the perturbation radii determines how much the perturbed image (i.e adversary) can differ from the original image. Several research papers have experimented and/or observed the effects of perturbation radii [9, 36, 17, 2]. One characteristic of adversarial training is that of the trade-off between adversarial robustness and test accuracy [36]. A larger perturbation radii means the adversarial attack algorithm can generate a larger set of perturbed images and make the images more perturbed themselves, resulting in a more effective attack. When a neural network trains on larger perturbations the robustness to stronger attacks increases, however, its classification power for normal examples decreases, as it becomes accustomed to learning large perturbations.

Results. Figures 4a and 4b show the results of our tests for various increasing values

of perturbation radii. From Figure 4a we notice that for the Adven ensembled model, as the perturbation radii increases, the robustness to the PGD attack increases but with a decrease to the normal test data-set accuracy. Indicating that this trade-off characteristic of adversarial training is also relevant in an ensemble setting. In Figure 4b we notice a similar pattern, however, there are several differences between the two. Firstly, Adven ensemble is far more robust than a single learner for all tested perturbation radii. This indicates that multiple learners are better able to learn the larger perturbations produced by the greater radii in comparison to a single learner, and thus have greater robustness. Secondly, as Figure 4b shows, the single learner exhibits a major drop in test set accuracy past perturbation $\epsilon = 0.20$. Work in [17] links this to a weak learners insufficient capacity to learn stronger adversaries, resulting in the model completely sacrifice performance on natural examples to provide minimal robustness. However, as Figure 4a shows, an ensemble can mitigate this issue greatly, as such a large sacrifice is not noticed. Finally, we observed perturbation radii had no effect on training time; additional robustness is computationally "free".



(a) Adven ensemble **(b)** Single learner

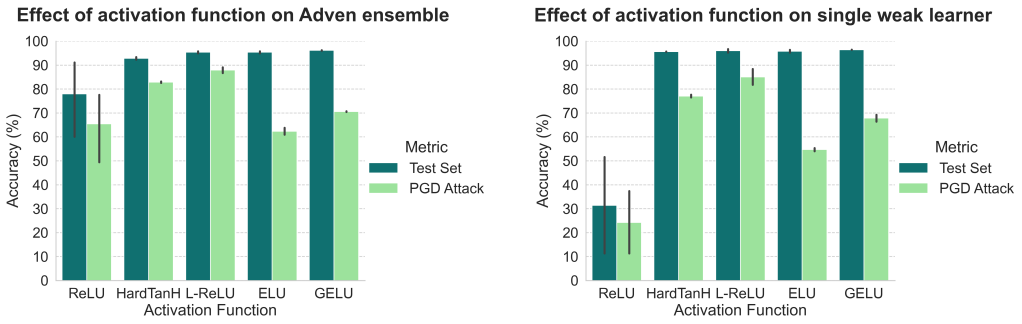
Figure 4: Perturbation Radii Test | Test set accuracy and PGD robustness accuracy observed as the perturbation radii of the PGD adversarial generation algorithm used during adversarial training increases. Left (4a) shows this for an Adven ensemble. Right (4b) for a single learner. PGD attack uses default value of perturbation $\epsilon = 78/255$. 75% confidence regions are shown for each line.

5.4 Activation Function

Context. To the best of our knowledge, no work has been done exploring the effects of activation functions on adversarial trained ensembled model. However, there are conflicting findings about the effects of activation functions on adversarial training. In [32] it was found that smooth activation functions (having continuous derivatives over an interval) allow the neural network to better compute gradient updates during adversarial training, yielding in greater robustness. In particular, they posited that ReLU weakens adversarial training. In contrast, [9] did not observe a clear trend as [33] on their experiments.

Results. Figures 5a and 5b show the effect of different *activation functions* on the *robustness to PGD attack* and *test set accuracy* for an Adven ensemble and a single learner respectively. Two key insights can be taken away from these experiments. Firstly, in both Figures 5a and 5b we observe that the models using Leaky_ReLU (in figures L-ReLU) activation function provides the most robustness, followed by HardTanH and GeLU. Our

findings are more in line with [9] and in contrast to [32] who found smooth activation functions like GeLU and ELU to provide the highest robustness. This suggests that there might be some other correlated factor with activation functions that our work and [9] has in common. The interesting takeaway is that weak learners prefer non-smooth activation functions over smooth activation functions. Secondly, Figure 5a shows that the Adven ensemble provides more robustness than a single learner for each different activation function. However, what is rather interesting is that the ensembling provides the greatest boost when a single learner is using the ReLU activation function. As Figure 5b shows, a single learner using the HardTanH or Leaky_ReLU activation functions provides more robustness than an Adven ensemble of 7 learners using ReLU. Suggesting that Leaky_ReLU and HardTanH synergise well with adversarial training, allowing a weak learner to learn adversaries efficiently. However, it is unclear why exactly this is the case. Finally, we observed activation function had no effect on training time; additional robustness is computationally "free".



(a) Adven ensemble

(b) Single learner

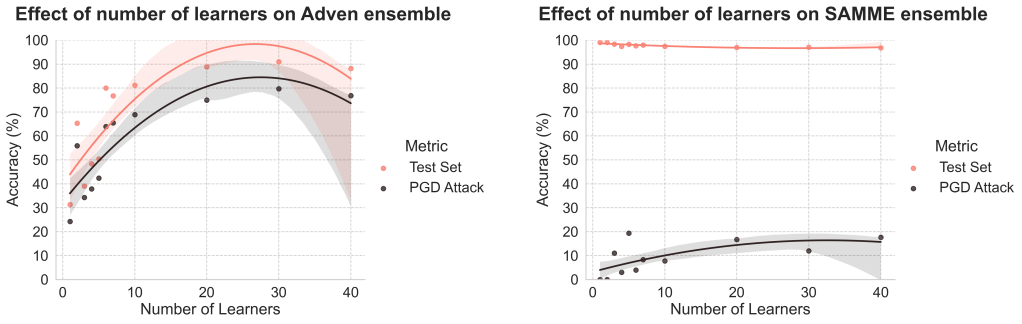
Figure 5: Activation Function Test | Test set accuracy and PGD robustness accuracy observed for different activation functions used during adversarial training. Left (5a) shows this for an Adven ensemble. Right (5b) for a single learner. Error bars represent 95% confidence interval.

5.5 Number of Weak Learners

Context. Research by [13] and [35] found a positive relationship between the number of learners and test set accuracy for a CNN ensemble that is *not* performing adversarial training, [13] also observed that an ensemble achieved higher test set accuracy than a single learner. Tests in [13] showed that an ensemble which performed adversarial training gained robustness by adding up to 6 learners, but adding any more didn't give any visible trends.

Results. Figures 6a and 6b show the effect of varying the *number of weak learners* on the *robustness to PGD attack* and *test set accuracy* for an Adven and SAMME ensemble respectively. Three key insights can be drawn. Firstly, in contrast to findings by [13] and [35], Figure 6b shows the test set accuracy decrease by 2% as we increase the number of learners from 1 to 40 in the SAMME ensemble. We suspect this could be due to the different experiment settings; they use other AdaBoost variations, and test on other datasets. Secondly, Figure 6b shows that ensembling on its own provides robustness to PGD attacks. Compared to a single learner which gives no robustness, three or more learners can provide up to 20% robustness. Thirdly, Figure 6a shows that for an Adven ensemble increasing the

number of learners increases both the robustness to PGD attacks and test set accuracy in a logarithmic manner. This indicates two properties: (1) more learners are, in a way, able to offset the expected trade-off with test accuracy, (2) more weak learners are better able to learn perturbed images. We notice the logarithmic relationship till 30 learners, adding more results in a decline to robustness. It's hypothesized that at this high count of learners, it becomes easier for the ensemble to make misclassifications. Specifically, during the ensemble voting prediction, correct predictions from a low number of high-weighted weak learners can be overruled by the remaining larger number of low-weighted weak learners missclassifiying. To verify this hypothesis, we would need to perform learner weight analytics on ensemble misclassifications, which was not under the scope of the paper, and has been left for future work. Finally, we observed training time to be linear with the number of learners, with each learner taking 105 seconds. Therefore, any gained robustness is not computationally "free".



(a) Adven ensemble

(b) SAMME ensemble

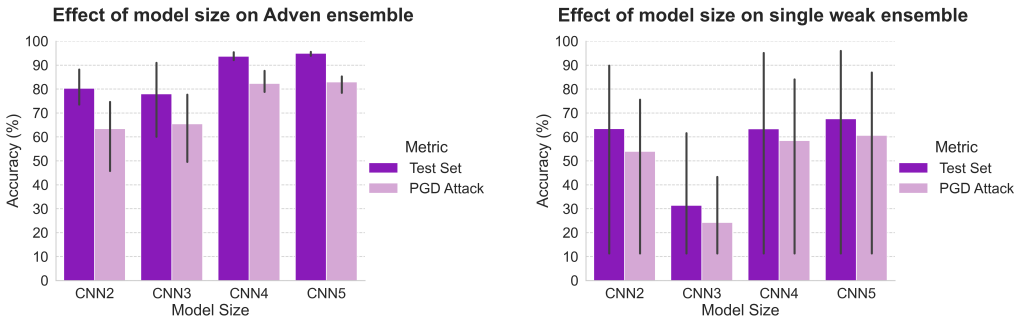
Figure 6: Number of Learners Test | Test set accuracy and PGD robustness accuracy observed as the number of learners in the ensemble increases. Left (6a) shows this for Adven ensemble. Right (6b) for SAMME ensemble. 75% confidence regions are shown for each line.

5.6 Model Size

Context. Existing research has verified the significance of the model architecture on adversarial robustness. Madry [17] found that a higher capacity model is more robust as its better able to successfully train against strong adversaries, and that for robustness the model requires larger capacity than for classifying benign examples only. Recent research by [9], [28] and [24] have noted that larger and deeper models provide improved adversarial robustness.

Results. Figures 7a and 7b show the effect of increasing the *size (depth) of the weak learner* on the *robustness to PGD attack* and *test set accuracy* for an Adven ensemble and single learner respectively. Figure 7b shows that as the model capacity increases by increasing model size, a single learner can better perform adversarial training and becomes more robust. This is in line with the aforementioned findings by previous works [17][9][28][24]. What is rather interesting is that the same pattern is observed in Figure 7a for the Adven ensemble, specifically, there is a jump of approximately 20% in robustness from using 3 layer CNN to a 5 layer CNN. The reason for this can be linked back to an observation in [35]: AdaBoost ensembling's accuracy is limited by accuracy of the weak learner itself. For an

ensemble that uses more complex weak learners, the probability of the votes being more unanimous is higher during the prediction voting phase, leading to a high test set accuracy. This combined with the fact that a higher capacity single learner can better perform adversarial training (Figure 7b) leads to the Adven ensemble also having higher robustness. Finally, we observed training a single two layer CNN took 81 seconds, and each increment to the depth costed 16 seconds. Therefore, gained robustness is not computationally "free".



(a) Adven ensemble

(b) Single learner

Figure 7: Model Size Test | Test set accuracy and PGD robustness accuracy for different sizes of the weak learner used during training. Left (7a) shows this for an Adven Ensemble. Right (7b) for a single learner. Error bars represent 95% confidence intervals.

5.7 Fashion-MNIST Generalisability & Final Combination

As discussed in Section 4 and Section 3.4 we conducted tests on the Fashion-MNIST dataset to check if our observed patterns were not data-set dependent. Full results and procedure are in Appendix C. In summary, out of the six variables, model size and loss function didn't match the expected pattern on Fashion-MNIST, thus deemded more dataset dependent.

Considering all the findings of our conducted experiments, we tested ten combinations of the six variables to find the most robust Adven ensemble for the MNIST dataset; full details are in Appendix D. We found the best combination of Adven to used: (1) Perturbation radii = 78/255, (2) Activation Function = Leaky_ReLU, (3) Loss Function = TRADES, (4) Adversary Algorithm = PGD, (5) Model Size = 5, and (6) Number of learners = 15. It achieved a test set accuracy of **96.72%**, and a robustness of **91.88%** to PGD attacks.

6 Conclusions and Future Work

This section will bring together the findings of our research, discuss the limitations and possible future contributions.

6.1 Conclusion

The purpose of this paper was to evaluate *how* an AdaBoost ensemble learning model can be used to provide adversarial robustness to gradient-based white-box attacks when the "weak" learners of the ensemble are neural networks that performed adversarial training. We first

adapted the SAMME algorithm (multi-class extension of AdaBoost) to support CNNs that perform adversarial training; our adaptation was named Adven. To answer the research question, we conducted experiments where we varied six variables of the Adven training procedure to explore how these influenced the ensembles accuracy on unseen clean images and its robustness to a fixed white-box PGD attack. Overall, we found all of the chosen variables to show relevant effects to robustness.

Empirically, results in Section 5 showed the Adven ensemble providing greater robustness than a single weak learner in all tests, and was computationally efficient; training time scaled linearly with the number of learners, and the remaining variables could provide robustness with little to none additional training time. It's clear that an ensemble synergises well with neural networks. Theoretically, we found that Adven inherited known AdaBoost ensemble characteristics, and transcended them into an adversarial training context. Namely, Adven was found to provide greater robustness with increasing number of weak learners, and the ensemble was limited by the individual weak learners accuracy. Results also showed that Adven inherited known adversarial training characteristics, and transcended them into an ensemble context. Specifically, Adven was found to provide greater robustness when the weak learners performing adversarial training were of higher capacity and trained on stronger adversarial attacks with large perturbation radii. We also found Adven to have to have interesting characteristics of its own. An Adven ensemble exhibited greater resistance to the trade-off effect than a single weak learner, and it preferred non-smooth activation functions. When we tested these findings on the Fashion-MNIST dataset, we found that most of them were not dataset dependent. Thus, Adven provided robustness on other datasets as-well.

Based on these findings we can answer the research question: Adven, an ensemble method that performs adversarial training, can be used to provide robustness to white-box attacks by following known phenomena of adversarial training and ensemble learning and exploiting its own unique characteristics. Our best ensemble applies this, and is able to achieve 96.72% test set accuracy and 91.88% robustness to PGD attacks on the MNIST dataset.

6.2 Limitations & Future Work

Although our research found interesting results, there are some limitations with respect to our evaluation protocol. Firstly, due to limited computational power and time constraints, we conducted tests on the MNIST dataset and used a "weak" attack such as PGD with 20 iterations. Current research uses stronger attacks like AutoAttack [4] and on harder datasets like CIFAR-10 [15], making it difficult for us to make conclusive statements that Adven will perform just as aswell on harder scenarios. Secondly, our attacks are computed using a single weak learner from the ensemble, it would be interesting to see how we can use the entire ensemble to generate adversaries and how that affects the ensembles robustness. In comparison to existing defense methods, Adven falls short: on similar attacks on MNIST, [30] achieves similar results but in shorter training times, and [36] is able to achieve 99% robustness with longer training times. Regardless, we strongly believe this area of research has potential to be improved. Further research could explore the ensembling algorithm aspect more; Adven uses the basic AdaBoost multiclass extension SAMME, but better alternatives exist [35][20]. Additionally, future work can explore effects of hyper parameters (number of epochs, batch size, optimizers, learning rate, etc) on robustness. It would also be interesting to explore black-box attacks on Adven; seeing how Adven influences adversary transferabilty and gradient approximation methods. Finally, research can explore variations of adversarial training or other defense approaches, and apply them to an ensemble setting.

7 Responsible Research

This section describes how our research methods have followed reproducibility principles, and the ethical aspects of our research.

7.1 Reproducibility of our methods

During the period of our work, we designed our implementation to easily allow reproducible results. We realise that replication is the standard by which scientific claims are validated [22], and to ease this process, our research needs to be easily replicable with reasonable time and expense. Often the issue is with regards to data; experiments use large datasets that cannot be manually analysed, and apply specific modifications that are often not detailed [18]. To avoid these issues, we used well known and *easily* accessible datasets (MNIST [5] and Fashion-MNIST [31]) for our experiments. Following current standards of reproducible research in computational science [22], we have made our entire code base public [1]. The code base is well documented, uses well documented frameworks (PyTorch [21] and AdverTorch [6]), and follows software engineering methods for ease of understanding and maintainability. Furthermore, the code base contains an executable file that allows the user to easily replicate all of our experiments by simply filling in the desired test parameters. Additionally, as stated in Section 3.4, we conducted each of our experiments three times for statistical significance, and displayed mean values along with confidence intervals to show the variance of our results. We are aware that replicated tests will not give the exact same results, however, the confidence intervals helps validating if they were nonetheless within expectation.

7.2 Ethics of Adversarial Research

Adversarial attacks can have real world consequences. Success in computer vision and natural language processing fields are integrating trained classifiers into security-critical applications [17]. Examples of these are autonomous cars, facial recognition software, malware detection, spam email detection, and many others [17]. Research in [16] demonstrated that in physical world scenarios, adversarial attacks can cause a classifier to be fooled. An extreme scenario could be an adversarial perturbation to a stop sign causing an autonomous car to not recognize it, potentially leading to accidents. Now it is more important than ever that machine learning classifiers are designed with security as a priority; resistance to adversarial inputs being a design goal [17][36]. Our work attempts to contribute to existing research being conducted on defenses to adversarial attacks. We understand that openly exploring defense approaches and identifying their strengths and weaknesses makes it easier for attackers to develop workarounds to our discoveries. However, it is important that this topic becomes widespread in the research community, so that more developers and researchers can contribute to strengthening defense against such attacks.

As we continue to integrate machine learning into automated decisions making processes, one day these classifiers may produce decisions that have legal consequences for us. The General Data Protection Regulation (Article 22) regulates that it is then the responsibility of the developer or whomever is in control of the classifier, that suitable safeguards to protect the subjects rights and freedoms are implemented. Research in adversarial defense is crucial to accomplish this, and we encourage the community to continue researching in this field.

[1] https://github.com/Kanish77/BRP-AdversarialAttacks

References

[1] Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

[2] Yogesh Balaji, Tom Goldstein, and Judy Hoffman. Instance adaptive adversarial training: Improved accuracy tradeoffs in neural nets. *arXiv preprint arXiv:1910.08051*, 2019.

[3] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[4] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International conference on machine learning*, pages 2206–2216. PMLR, 2020.

[5] Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[6] Gavin Weiguang Ding, Luyu Wang, and Xiaomeng Jin. Advertorch v0. 1: An adversarial robustness toolbox based on pytorch. *arXiv preprint arXiv:1902.07623*, 2019.

[7] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9185–9193, 2018.

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

[9] Sven Gowal, Chongli Qin, Jonathan Uesato, Timothy Mann, and Pushmeet Kohli. Uncovering the limits of adversarial training against norm-bounded adversarial examples. *arXiv preprint arXiv:2010.03593*, 2020.

[10] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[11] Haifang Li Jianfang Cao, Junjie Chen. An adaboost-backpropagation neural network for automated image sentiment classification. *The Scientific World Journal*, 2014, 2014.

[12] Harini Kannan, Alexey Kurakin, and Ian Goodfellow. Adversarial logit pairing. *arXiv preprint arXiv:1803.06373*, 2018.

[13] Simarpreet Kareer. Effect of boosting on adversarial robustness, 2021.

[14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[15] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research).

[16] Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *Artificial intelligence safety and security*, pages 99–112. Chapman and Hall/CRC, 2018.

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

[18] Jill P. Mesirov. Accessible reproducible research. *Science*, 327(5964):415–416, 2010.

[19] Marius Mosbach, Maksym Andriushchenko, Thomas Trost, Matthias Hein, and Dietrich Klakow. Logit pairing methods can fool gradient-based attacks. *arXiv preprint arXiv:1810.12042*, 2018.

[20] Indraneel Mukherjee and Robert E Schapire. A theory of multiclass boosting. *Advances in Neural Information Processing Systems*, 23, 2010.

[21] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[22] Roger D. Peng. Reproducible research in computational science. *Science*, 334(6060):1226–1227, 2011.

[23] Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein. On the expressive power of deep neural networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2847–2854. PMLR, 06–11 Aug 2017.

[24] Leslie Rice, Eric Wong, and Zico Kolter. Overfitting in adversarially robust deep learning. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 8093–8104. PMLR, 13–18 Jul 2020.

[25] Robert E. Schapire. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612, 1999.

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

[27] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017.

[28] Jonathan Uesato, Jean-Baptiste Alayrac, Po-Sen Huang, Robert Stanforth, Alhussein Fawzi, and Pushmeet Kohli. Are labels required for improving adversarial robustness? *arXiv preprint arXiv:1905.13725*, 2019.

[29] Yisen Wang, Difan Zou, Jinfeng Yi, James Bailey, Xingjun Ma, and Quanquan Gu. Improving adversarial robustness requires revisiting misclassified examples. In *International Conference on Learning Representations*, 2020.

[30] Eric Wong, Leslie Rice, and J Zico Kolter. Fast is better than free: Revisiting adversarial training. *arXiv preprint arXiv:2001.03994*, 2020.

[31] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

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

[33] Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan L Yuille, and Kaiming He. Feature denoising for improving adversarial robustness. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 501–509, 2019.

[34] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

[35] Shuo Yang, Li-Fang Chen, Tao Yan, Yun-Hao Zhao, and Ye-Jia Fan. An ensemble classification algorithm for convolutional neural network based on adaboost. In *2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS)*, pages 401–406, 2017.

[36] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. In *International conference on machine learning*, pages 7472–7482. PMLR, 2019.

[37] Zhilu Zhang and Mert Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. *Advances in neural information processing systems*, 31, 2018.

[38] Ji Zhu, Saharon Rosset, Hui Zou, and Trevor Hastie. Multi-class adaboost. *Statistics and its interface*, 2, 02 2006.

A Pseudo-code of algorithms

This section of the Appendices has detailed the pseudo-code of the SAMME and Adven algorithm.

Algorithm 1 SAMME

Require: Number of: classes k, examples n, weak learners T

Require: Training set $S = \{x_i, y_i\}$, with $y_i \in \{1, ...k\}$ and $x_i \in X$

1: Initialise sample weights $w_i = 1/n$ for $i = 1, ..., n$

2: **for** t=1 to T **do**

3: Train weak learner $H^{(t)}(x)$ while using weights w

4: Compute weak learner error as:

$$e_t = \sum_{i=1}^{n} w_i \cdot [y_i \neq H^{(t)}(x_i)] / \sum_{i=1}^{n} w_i$$

5: Compute weak learner weight as:

$$\alpha^{(t)} = \log((1 - e_t)/e_t) + \log(k - 1)$$

6: Update sample weights for $i = 1, ..., n$:

$$w_i \leftarrow w_i \cdot exp(\alpha^{(t)} \cdot [y_i \neq H^{(t)}(x_i)])$$

7: Normalise weights

8: **end for**

9: Output classifications as:

$$C(x) = \underset{k}{\operatorname{argmax}} \sum_{t=1}^{T} \alpha^{(t)} \cdot [H^{(t)}(x) = k]$$

Algorithm 2 Adven

Require: Number of: classes k, examples n, weak learners T. Adv-gen algorithm Δ

Require: Training set $S = \{x_i, y_i\}$, with $y_i \in \{1, ...k\}$ and $x_i \in X$

1: Initialise sample weights $w_i = 1/n$ for $i = 1, ..., n$

2: **for** t=1 to T **do**

3: Adversarial train $CNN^{(t)}(x + \Delta)$ while using weights w to compute weighted loss

4: Compute weak learner error as:

$$e_t = \sum_{i=1}^{n} w_i \cdot [y_i \neq CNN^{(t)}(x_i + \Delta)] / \sum_{i=1}^{n} w_i$$

5: Compute weak learner weight as:

$$\alpha^{(t)} = \log((1 - e_t)/e_t) + \log(k - 1)$$

6: Update sample weights for $i = 1, ..., n$:

$$w_i \leftarrow w_i \cdot exp(\alpha^{(t)} \cdot [y_i \neq CNN^{(t)}(x_i + \Delta)])$$

7: Normalise weights

8: **end for**

9: Output classifications as:

$$C(x) = \underset{k}{\operatorname{argmax}} \sum_{t=1}^{T} \alpha^{(t)} \cdot [CNN^{(t)}(x) = k]$$

B Preliminary Experiments

B.1 Motivation

There were two main reasons for conducting preliminary experiments. The first was to ascertain whether or not the topic of the research, ensembling with adversarial training, will provide any noticeable results. The second reason was that to conduct the experiments for this paper, it was important to decide the values for the independent variables that were to be tested. In order to decide values in systematic manner rather than random guessing, two approaches were taken. The first approach was to decide values based on existing research. The second approach was to decide values by first doing some preliminary experiments on heuristically chosen initial values, and based on the results finalise a new set.

Preliminary experiments were conducted for three out of the six independent variables. These three variables were: adversarial generation algorithm, perturbation radii, and number of weak learners. There are multiple reasons for as to why these ones were particularly tested:

1. For the remaining three independent variables (model size, loss function, and activation

function) values can be chosen based on existing literature (i.e the first approach as explained above). Thus, with the limited research time, it was decided to only do preliminary experiments with the other three variables

2. These three independent variables are numeric values, and thus its important to decide with what granularity to test them. For example, with the number of weak learners, should we test for 100 or more weak learners, or stop at 75? Another example, with the perturbation radii should we test for values between a specific range like 0.2-0.3 or test larger values?

3. Doing tests with these three values allowed us to ascertain whether or not the core topic of the research, ensembling with adversarial training, provided any noticeable results.

B.2 Set-up

The following are settings that are used for all preliminary tests:

- A three layer CNN is used as the weak learner. Two convolution layers that both have a kernel size of 5, and a maxpool operation in-between. One fully connected layer.
- The CNN weak learner uses ReLU activation function, and the (weighted) cross-entropy loss (it's weighted based on the sample weights).
- MNIST data-set was used for training and testing. For training: batch size was 100, number of epochs was 3, learning rate was 0.002 with Adam optimizer.
- The evaluation criteria was: robustness to PGD attack ($\epsilon = 0.156$, number of iterations $= 20$), robustness to FGSM attack ($\epsilon = 0.156$), and accuracy on test dataset.
- Default value for: number of weak learner was 7, adversarial generation algorithm was PGD ($\epsilon = 0.156$, number of iterations $= 20$), perturbation radii was 0.156. Default values are important to test the independent variable in a controlled manner. For example, when not conducting tests for the perturbation radii, the tests for number of weak learner and adversarial generation algorithm used a fixed perturbation radii so that effects of the independent variable is more clear.

The following are the values that were tested for each of the three independent variables:

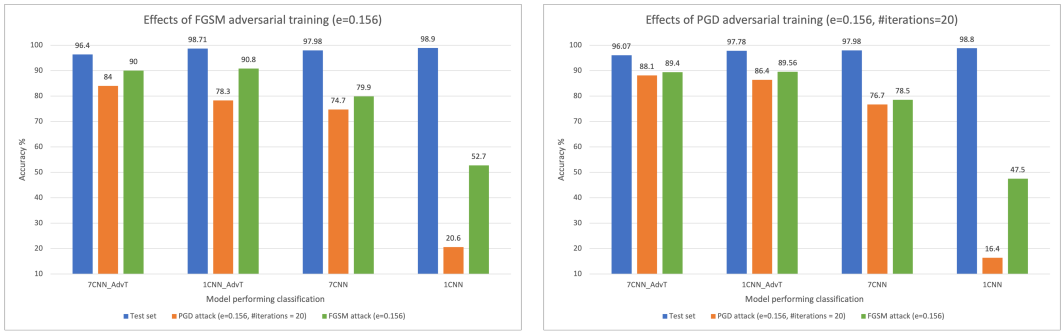
- Number of weak learners: 5, 7, 10, 15
- Adversarial generation algorithm: FGSM ($\epsilon = 0.156$), PGD ($\epsilon = 0.156$, number of iterations $= 20$)
- Perturbation radii: 20/255, 40/255, 60/255, 80/255

Finally, to better see the independent effects of AdaBoost ensemlbing and adversarial training, and the combined effect of ensembling and adversarial training, for each tested value four models were used (subject to the settings mentioned above):

- 7CNN_AdvT: 7 weak learner ensemble where each learner did adversarial training.
- 1CNN_AdvT: Single weak CNN doing adversarial training
- 7CNN: 7 weak learner ensemble where **no** adversarial training was done
- 1CNN: Single weak learner where **no** adversarial training was done

B.3 Results

Results for tests of *adversarial generation algorithm*



(a) Experiment with FGSM **(b)** Experiment with PGD

Figure 8: Adversarial Generation Algorithm Tests

Comparing Figure 8a and 8b many useful points are noticed:

- We notice that when performing adversarial training with FGSM, we still get robustness against PGD attacks. This could be because the PGD attack done in this experiment is "weak", having a small radii and low number of iterations, and is thus similar to an FGSM attack.
- Test accuracy is highest with no adaboost and no adversarial training, which is in line with previous findings of trade-offs between normal test accuracy vs adversarial robustness [36].
- We notice that when we train on a stronger attack, the robustness increases against these stronger attacks (eg. PGD robustness goes up by 4.1%)
- AdaBoost ensembling provides robustness on its own. Comparing the 7CNN and 1CNN models in Figures 8a and 8b, we see the 7CNN ensembled model being far greater robust to both FGSM and PGD attacks.
- Adversarial training provides robustness on its own. Comparing the 1CNN_AdvT and 1CNN_AdvT models in Figures 8a and 8b, we see the 1CNN_AdvT model being far greater robust to both FGSM and PGD attacks.
- Finally, and perhaps more importantly, AdaBoost ensembling combined with adversarial training provides the most robustness. In Figures 8a and 8b, we see the 7CNN_AdvT model having the greatest robustness to PGD attacks.

Results for tests of *number of weak learners*

Figure 9 shows that the ensembled model that performed adversarial training with 7 weak learners seemed to be the most robust to attacks. A pattern that is noticed is that after 10 learners the ensemble seems to be ineffective to adding robustness. However this pattern

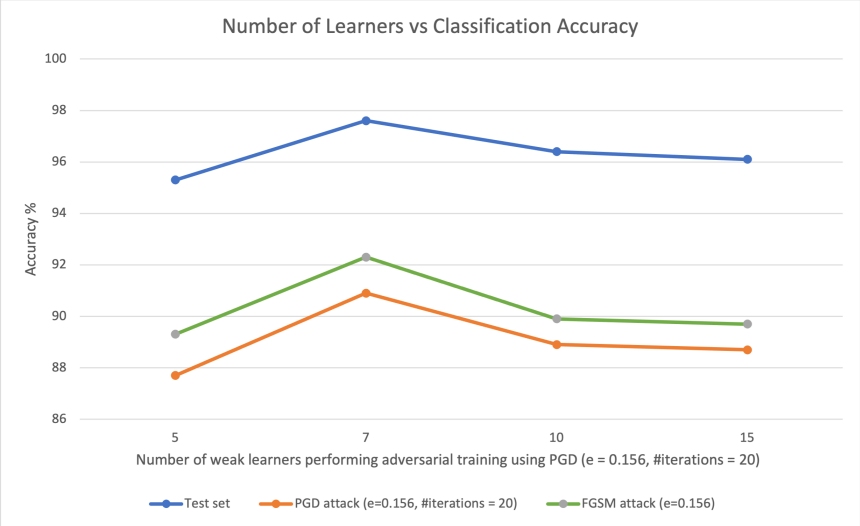


Figure 9: Tests for varying number of weak learners

may not hold as these preliminary tests stop at 15 learners, and thus it is of interest to test for greater number of learners.

Results for tests of *perturbation radii*

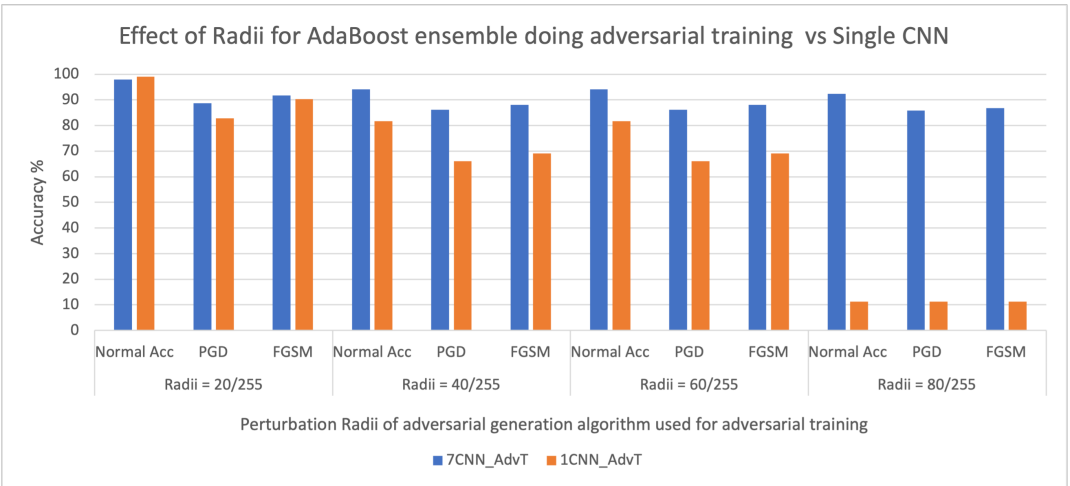


Figure 10: Tests for varying perturbation radii

Figure 10 shows the accuracy's to test set, PGD and FGSM attack, for two models: the first in blue being an AdaBoost ensemble of 7 learners doing adversarial training, the second in orange being a single learner doing adversarial training. This figure shows that AdaBoost ensembling helps for robustness, for all cases except one, 7 learners performed better than

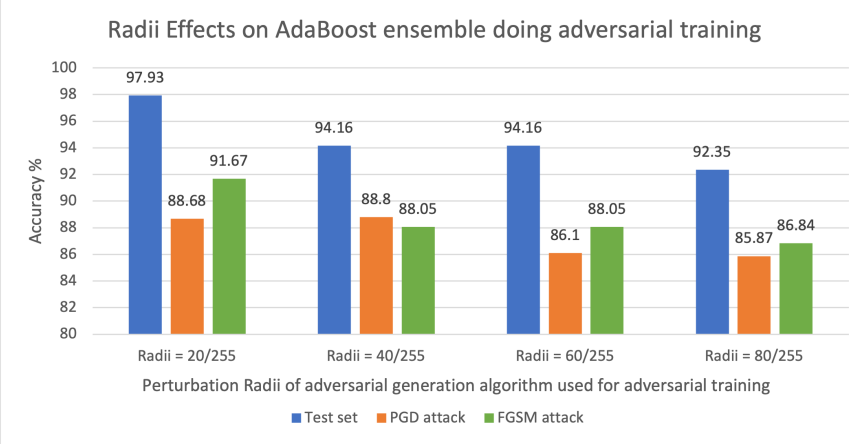


Figure 11: Perturbation Radii effects on AdaBoost ensemble

a single learner.

Another interesting pattern is that difference between the accuracy's of the two models becomes greater when the perturbation radii increases. A greater perturbation radii means that for the same input image, the adversarial attack algorithm can generate a larger set of perturbed images, which makes the attack more effective. The figure shows that multiple learners are better able to learn the larger set produced by the greater radii in comparison to a single learner. Therefore, relatively maintaining the same accuracy's as the perturbation radii increases.

Additionally, Figure 11 zooms into the accuracy's for the AdaBoost ensembled model. We see a general pattern that as the perturbation radii increases, the ensembled models accuracy decreases. This indicates two main things. First is that training on greater perturbations doesn't help against a lower perturbation attack. Second is that training on a larger radii requires a more complex weak learner. Adding more weak learners to the ensemble will not help because the adversarial generation algorithm produces perturbations randomly, so if the weak learner itself cannot learn the large perturbations on its own then the ensemble cannot help.

B.4 Finalised Independent variable values

After conducting some preliminary tests, the following are the finalized values for the independent variables and the reason for picking them:

1. **Model Size.** The following sizes will be tested: 2, 3, 4, 5. These sizes will be tested because any bigger CNN's will not be classified as "weak" learners anymore. We stop at 5 because, as similar research conducted for an AdaBoost CNN ensemble used the LeNet classifier as the weak learner which is of 5 layers [35].

2. **Activation Function.** The following functions will be tested: ReLU, HardTanh, Leaky ReLU, ELU, and GelU. Most of these similar to the activation functions tested by [9], and the others are common in the machine learning community.

3. **Loss Function.** The following loss functions will be tested: cross entropy, kl-divergence, TRADES.

4. **Perturbation Radii.** The following values will be tested: $\epsilon = \{8/255, 18/255, 28/255, 38/255, 48/255, 58/255, 68/255, 78/255\}$. These values were chosen because it will be good to test if observed patterns on the preliminary test are also seen in the real experiments. Finally, more granular radii are tested, started from $\epsilon = 8/255$ as other research papers have often used this for their experiments [9].

5. **Adversarial Generation Algorithm.** The following adversarial generation algorithms will be tested: FGSM, BIM, PGD, MIA.

6. **Number of weak learners.** The following number of weak learners will be tested: 2, 3, 4, 5, 6, 7, 10, 20, 30, 40. The first half of tests are quite granular up to 7 weak learners, this is similar to the study by [35]. The larger number of learners are there because similar researched papers ([13] [35][11]) have not tested this large number of learners.

C Fashion-MNIST Generalisability Tests

As stated in Section 4 and Section 3.4 to ensure that our findings were not dataset dependent, we conducted some additional tests on the Fashion-MNIST dataset. We will begin by explaining how exactly we conducted these experiments, and then we will present our findings.

C.1 Experiment Procedure

Due to time constraints, it was not feasible for us to replicate all experiments we did on the Fashion-MNIST dataset. We replicated a subset of experiments, specifically, from each of the six independent variables we, we selected two values (Section 4 gives an overview of all the variables and the values that were tested on the MNIST dataset). The first value was the one that gave Adven the *least* robustness on the MNIST dataset, and the second value was the one that gave Adven the *largest* robustness on the MNIST dataset.

With the chosen values, we conducted experiments following the exact same procedure as we used for the MNIST tests (i.e the same default values, training procedure, and evaluation protocol); details can be found in Section 4.

The following are the chosen values and the expected trend for the Fashion-MNIST dataset based on the results found on the MNIST dataset.

1. Model Size: CNN of size 2 and 5. Based on results on the MNIST dataset in Section 5.6, we expect Adven using CNN with 5 layers to provide more robustness.

2. Activation Function: ELU and Leaky_ReLU. Based on results on the MNIST dataset in Section 5.4 we expect Adven using the Leaky_ReLU activation function to provide more robustness.

3. Loss Function: CE and TRADES. Based on results on the MNIST dataset in Section 5.1 we expect Adven using the TRADES loss function to provide more robustness.

4. Perturbation Radii: 8/255 and 78/255. Based on results on the MNIST dataset in Section 5.3 we expect Adven using the radii 78/255 to provide more robustness.

5. Adversarial Generation Algorithm: FGSM and PGD. Based on results on the MNIST dataset in Section 5.2 we expect Adven using PGD for training to provide more robustness.

6. Number of weak learners: 1 and 30. Based on results on the MNIST dataset in Section 5.5 we expect Adven with 30 weak learners to provide more robustness.

C.2 Results.

Table 1 shows the result of the tests. With the above stated expected trend in mind, we can see that for four out of the six tested variables, the expected pattern is observed on the Fashion-MNIST dataset. The two that differed were the loss function and model size, these are highlighted in red. One interesting observation is that for both of these independent variables, the difference of the PGD robustness between the two tested values is very little. Furthermore, TRADES is a very effective loss function that aids in learning adversaries very well. Additionally, a 5 layer CNN has more capacity than a 2 layer CNN, and can better learn adversaries. Due to these characteristics, we believe that the Adven ensemble that used a 5 layer CNN and the ensemble that used the TRADES loss overfitted on the training data.

Variable	**Value**	**Test Set Accuracy**	**PGD Attack Robustness**
Perturbation Radii	8/255	82.33	4.26
	78/255	59.54	53.55
Number of weak learners	1	54.03	40.91
	30	57,67	51.38
Activation Function	ELU	57.12	34.06
	Leaky_ReLU	56.65	42.6
Loss Function	CE	54.84	**43.12**
	TRADES	56.85	42.14
Adversary Generation Algorithm	FGSM	53.52	13.38
	PGD	56.8	38.63
Model Size	CNN2	60.35	**48.41**
	CNN5	57.84	47.77

Table 1: Fashion-MNIST Generalisability Tests | For each independent variable, the best and worst performing values were validated on the Fashion-MNIST dataset. Test accuracy and robustness accuracy to PGD attack is shown. Highlights in red are for the results that do not match the expected pattern.

D Combination Tests

In this section of the Appendices, we explain how we conducted our final set of experiments to find the most robust Adven ensemble for the MNIST dataset.

D.1 Strategy

Our strategy to make the most robust Adven ensemble is to use the six variable values that exhibited the greatest robustness individually. Specifically, the Adven ensemble had the greatest robustness when it:

- trained on perturbation radii of 78/255 (Section 5.3, Figure 4a)
- trained using the Leaky_ReLU activation function (Section 5.4, Figure 4a)
- trained on the TRADES loss function (Section 5.1, Figure 2a)
- trained using 5 layer CNN's as the weak learners (Section 5.6, Figure 7a)
- used 20 or 30 weak learners (Section 5.5, Figure 6a), both of these values showed quite similar robustness
- trained using either PGD or BIM adversary algorithm (Section 5.2, Figure 5a), both of these values showed quite similar robustness

Since for the adversary algorithm and number of learners variables, two values were very similar, we decided to try out all combinations of them, the other four variables had the values set to what is stated above. Therefore, we tested four combinations. Table 2 shows the results.

Number of weak learners	**Adversary Generation Algorithm**	**Test Set Accuracy**	**PGD Attack Robustness**
20	PGD	96.33	88.67
20	BIM	96.42	89.02
30	PGD	96.18	87.65
30	BIM	96.1	87.7

Table 2: Combination Tests | For each test the test accuracy and robustness accuracy to PGD attack is shown.

As we can see, the best combination was found to be the Adven ensemble that had 20 weak learners, and used the BIM adversary algorithm: having a 89% robustness. However, these results were underwhelming, as when testing the six variables individually, some values already provided very close robustness. For example, in Section 5.6 Figure 7a showed Adven ensemble using 5 layer CNNs to provide close to 85% robustness, and in Section 5.4 Figure 5a showed Adven enseble using the Leaky_ReLU activation function provide close to 87% robustness. We expected that when we combined all the best findings variable values, we would get more robustness than 89%. We realised that our tested combinations were using a high number of weak learners, whereas the results we found in the referenced examples were tests that were using the default value of number of learners equal to 7. In Section 5.5 we mentioned that with high number of weak learners, the ensemble can begin to provide less robustness. Therefore, we decided to test a fewer number of weak learners, specifically, 7, 12 and 15. Similar to before, we tried combinations with both PGD and BIM adversary generation algorithm, with the other variables having the values we mentioned at the beginning of this section; this resulted in 6 combinations. Table 3 shows our results. Our intuition was correct, the Adven ensembles using a "medium" number of weak learners provided better robustness, breaking into the 90% range.

Number of weak learners	Adversary Generation Algorithm	Test Set Accuracy	PGD Attack Robustness
7	PGD	97.01	91.09
7	BIM	96.82	91.55
12	PGD	96.66	89.72
12	BIM	96.81	88.83
15	**PGD**	**96.72**	**91.88**
15	BIM	96.35	89.96

Table 3: Combination Tests 2 | For each test the test accuracy and robustness accuracy to PGD attack is shown. The best result is in bold.

Therefore, we can conclude that the best combination of the independent variables is: (1) Perturbation Radii = 78/255, (2) Activation function = Leaky_ReLU, (3) Loss function = TRADES, (4) Model Size = 5 layer CNN, (5) Number of weak learners = 15, (6) Adversary Generation Algorithm = PGD (with 20 iterations). This Adven ensemble has 96.72% test set accuracy, and has a robustness of 91.88%.