Adversarial Robustness in Model Ensembles

Stefanos Pertigkiozoglou

February 2021

1 Introduction

In this project we investigate under what conditions we can create an adversarially robust model that is defined as an ensemble of less robust (weakly robust) models. A model is said to not be adversarially robust when, given an input x, an adversary can apply a small perturbation p to create an input $x + p$ where the model makes a mistake. We call this adversarially perturbed input, an adversarial example. One simple and common practice in the literature is to define a petrubation as small, when its norm is smaller than a small constant ϵ_p. As norm we can use L_1, L_2 or L_∞

Adversarial robustness is a problem that was recently popularized for the case of neural networks [1]. In particular, it was shown that the widely used neural networks are not adversarially robust. Although it was popularized as a problem of neural networks, there is a lot of work that shows that a large variety of machine learning models , such as gradient boosting trees and SVMs, are not robust when they operate in an adversarial setting [2, 3].

Since the introduction of the problem of adversarial robustness, there is a lot of work that focuses on making models more robust to adversarial examples. These works either try to design models that are robust to adversarial perturbations [6, 7, 8, 9, 10], or models that can detect whether the input is sampled from the original data distribution or it was produced by an adversary [11, 12, 13]. Some of these works introduce methods that improve robustness using a ensemble of non robust models [6, 7]. The key intuition behind the use of an ensemble to improve robustness is that it is more difficult for the adversary to find a perturbation that fools the majority of the models used in the ensemble. These works provide empirical evidence that show that the proposed ensembles improve the adversarial robustness when the attacker uses a limited set of attacks. Of course these empirical evidence don't prove robustness because it is always possible that the adversary that have full knowledge of our model can introduce a novel attack, that was not tested in the experiments, and that bypass our defense [14]. An interesting problem is to show whether an ensemble of weakly robust models can provide a more robust overall model, or it is always possible that a more resourceful adversary can find the same amount of adversarial examples for the ensemble of models as it can for each individual weakly robust model.

So in this project we show that given a set of conditions, it is possible to create an ensemble of weakly robust models which is more adversarially robust compared to the individual models. First we define the adversarial robustness as a specific type of error (adversarial error), and then we show that using an ensemble of models we can produce an overall model with lower adversarial error compared to the individual models. This hypothesis that an ensemble of non-robust models can give a more robust final model is closely related with the boosting methods that were presented in the course [5, 4]. Here instead of having a "weak" model in terms of error we have a "weak" model in terms of robustness and our goal is to find an ensemble that boosts the performance of the model and instead of decreasing its error it increases its robustness.

In addition, we show that while we increase the overall robustness, this ensemble also achieves lower error in the original distribution compared to the error of the individual models.

Along with the theoretical analysis we also provide empirical evidence that support the results mentioned above. We do that by providing experimental results when we apply the ensemble in a simple problem of image classification. Of course any robustness shown with these experiments is conditioned on the attack that we used, so it has the same problem as the methods mentioned before. But nevertheless, showing that any proposed ensemble method increases adversarial robustness in an experimental setup, works at least as a preliminary proof of concept.

2 Adversarial Robustness

In this section we present a simple definition of adversarial robustness that will be used in the project. Although this definition doesn't encapsulate the full spectrum of the term "adversarial robustness", it is a good simplification that allows us to do a more precise analysis of the problem. Also notice that this definition borrows a lot from the PAC learning framework that was presented in the course and the concept of weak learnability.

Suppose that we have a learning algorithm L that for any function $c \in C$ and any distribution P it outputs a hypothesis $h \in H$ that achieves error $\epsilon_P(h)$ less than ϵ.

We will define the adversarial error as the error of h in an adversarial distribution P_{adv}, where an adversary can add to the original input a perturbation p with norm less or equal to ϵ_p. So the adversarial distribution P_{adv} is a distribution where each sample x' is created as follows :

- Sample x from the original distribution P
- Let an adversary add a perturbation p with $\|p\|_\infty \leq \epsilon_p$. This creates the final sample $x' = x + p$. (Here we assume that the goal of the adversary is to create samples x' where the error of h is maximized, also the attack is bounded by the L_∞ norm)

So given the above definition of the adversarial distribution P_{adv}, we can define the adversarial error as follows:

$$\epsilon_{\text{adv},P}(h) = \Pr_{x \sim P}\Big[\exists p \text{ s.t. } (\|p\|_\infty \leq \epsilon_p) \wedge \big(h(x+p) \neq c(x+p)\big)\Big]$$

Notice that in this definition, the errors of h in the original distribution are also included in the adversarial errors because if we sample an x where h makes an mistake, the adversary can easily add zero perturbation and still create a $x' = x$ where $h(x') \neq c(x')$. This means that $\epsilon_P(h) \leq \epsilon_{\text{adv},P}(h)$

3 Boosting Robustness in Binary Classification

Now assume that we want to learn from a class of binary functions $c \in C$ that map samples from a domain D to $\{-1, 1\}$. So $c : D \to \{-1, 1\}$. Also assume that we have a learning algorithm L where for each $c \in C$ and each distribution P, it outputs a hypothesis $h \in H$ where:

$$\epsilon_{\text{adv},P}(h) = \Pr_{x \sim P}\Big(\exists p \text{ s.t. } h(x+p) \neq c(x+p)\Big) \leq \mu < 1/2$$
$$\epsilon_P(h) = \Pr_{x \sim P}\Big(h(x) \neq c(x)\Big) \leq \epsilon \leq \mu < 1/2$$

In this definition in the adversarial error we skipped writing the constraint $\|p\|_\infty \leq \epsilon_p$. In order to make the presentation of the proof more compact, for the rest of this paper whenever we write $\exists p$ we will also assume the constraint $\|p\|_\infty \leq \epsilon_p$.

By closely following the proof presented in Schapire's paper "The strength of weak learnability" [5] we will show that we can use algorithm L to learn 3 hypothesis h_1, h_2, h_3 and create an ensemble $h_{ens} = sign(h_1 + h_2 + h_3)$ where:

$$\epsilon_{\text{adv},P}(h_{\text{ens}}) \leq \mu' < \mu \tag{1}$$
$$\epsilon_P(h_{\text{ens}}) \leq \epsilon' < \epsilon \tag{2}$$

We will prove inequality (1) by slightly modifying Schapire's proof in order to encapsulate the adversarial setting. Although the proof is quite similar with the one presented in [5], it is still useful to present it in order to both show how it can be easily expanded to the adversarial setting and also because some intermediate results will be useful to prove inequality (2).
The algorithm for getting hypothesis h_1, h_2, h_3 consists of the following steps.

1. Use L to learn c in the original distribution P. This will give us hypothesis h_1 where $\epsilon_{\text{adv},P}(h_1) = \mu_1$, $\epsilon_P(h_1) = \epsilon_1$.

2. Define event $A_1 = \{x : \exists p. \text{ s.t. } h_1(x+p) \neq c(x+p)\}$. Then define distribution P_2 where:

   - if $x \in A_1$
   $$\Pr_{x\sim P_2}\Big(x\Big) = \frac{\Pr_{x\sim P}\Big(x\Big)}{2\mu_1}$$
   - if $x \notin A_1$
   $$\Pr_{x\sim P_2}\Big(x\Big) = \frac{\Pr_{x\sim P}\Big(x\Big)}{2(1-\mu_1)}$$

3. Use L to learn c on distribution P_2 ,which will give us h_2 where $\epsilon_{\text{adv},P_2}(h_2) = \mu_2$, $\epsilon_{P_2}(h_2) = \epsilon_2$.

4. Define event $A_{1,2} = \{x : \exists p_1.\exists p_2. \text{ s.t. } h_1(x+p_1) \neq h_2(x+p_2)\}$. Then define distribution P_3 where:

   - if $x \in A_{1,2}$:
   $$\Pr_{x\sim P_3}\Big(x\Big) = \frac{\Pr_{x\sim P}\Big(x\Big)}{\Pr_{x\sim P}\Big(A_{1,2}\Big)}$$
   - if $x \notin A_{1,2}$:
   $$\Pr_{x\sim P_3}\Big(x\Big) = 0$$

5. Use L to learn c on distribution P_3 which will give us h_3 where $\epsilon_{\text{adv},P_3}(h_3) = \mu_3$, $\epsilon_{P_3}(h_3) = \epsilon_3$

Notice that in the definition of P_2 we have the edge case of $\mu_1 = 0$. But in this case, h_1 classifier makes no mistakes, so we can trivially output h_1 instead of the ensemble. When $1/2 > \mu_1 > 0$ we can simulate P_2 by sampling from P and then re-weighting the probability of x appropriately, based on whether $x \in A_1$ or $x \notin A_1$. Because $1/2 > \mu_1 > 0$, it can easily be shown that we can make the probability δ, of failing to sample from both sets, as small as we want by increasing the number of samples. Also this increase in samples scales efficiently with respect to $1/\delta$ (at most polynomially). A similar edge case exists for P_3 but in order to show that it cannot occur we need to first analyze the error of h_2 in P_2.

For h_2 we have event $A_2 = \{\exists p. \text{ s.t. } h_2(x+p) \neq c(x+p)\}$, which is the event where h_2 makes a mistake in an adversarial distribution. So

$$\begin{aligned}
\mu_2 = \Pr_{x\sim P_2}\Big(A_2\Big) &= \Pr_{x\sim P_2}\Big((A_2 \wedge A_1) \vee (A_2 \wedge \neg A_1)\Big) \\
&= \Pr_{x\sim P_2}\Big((A_2 \wedge A_1)\Big) + \Pr_{x\sim P_2}\Big((A_2 \wedge \neg A_1)\Big) \\
&= \sum_{x\in(A_2\wedge A_1)} \Pr_{x\sim P_2}\Big(x\Big) + \sum_{x\in(A_2\wedge\neg A_1)} \Pr_{x\sim P_2}\Big(x\Big) \\
&= \sum_{x\in(A_2\wedge A_1)} \frac{\Pr_{x\sim P}\Big(x\Big)}{2\mu_1} + \sum_{x\in(A_2\wedge\neg A_1)} \frac{\Pr_{x\sim P}\Big(x\Big)}{2(1-\mu_1)} \\
&= \frac{\Pr_{x\sim P}\Big(A_1 \wedge A_2\Big)}{2\mu_1} + \frac{\Pr_{x\sim P}\Big(A_2 \wedge \neg A_1\Big)}{2(1-\mu_1)} \qquad (3)
\end{aligned}$$

(Here we use summation with the assumption that our domain D is a discrete domain. In the case of continuous domains the summation will be replaced with an integral.)

Using equation (3) we can investigate the edge case $\Pr_{x\sim P}\Big(A_{1,2}\Big) = 0$, which exists in the definition of P_3. If $\Pr_{x\sim P}\Big(A_{1,2}\Big) = 0$ then $\Pr_{x\sim P}\Big(\forall p_1.\forall p_2.\quad h_1(x+p_1) = h_2(x+p_2)\Big) = 1$. This means that $\Pr_{x\sim P}\Big(A_1 \wedge A_2\Big) = \Pr_{x\sim P}\Big(A_1\Big) = \mu_1$ and thus if we replace it in equation (3) we get that $\mu_2 \geq 1/2$. This is a contradiction because we assume that L outputs hypothesis whith adversarial error less than $1/2$. Again similar with P_2, we can simulate P_3 by sampling from P and we can make the probability of failing as small as we want by increasing the number of samples appropriately.

Now if we define

$$
\begin{aligned}
x &= \Pr_{x\sim P}\Big(\neg A_1 \wedge \neg A_2\Big) = \Pr_{x\sim P}\Big((\nexists p_1 \text{ s.t. } h_1(x+p_1) \neq c(x+p_1)) \wedge (\nexists p_2 \text{ s.t. } h_2(x+p_2) \neq c(x+p_2))\Big)\\
y &= \Pr_{x\sim P}\Big(A_1 \wedge \neg A_2\Big) = \Pr_{x\sim P}\Big((\exists p_1 \text{ s.t. } h_1(x+p_1) \neq c(x+p_1)) \wedge (\nexists p_2 \text{ s.t. } h_2(x+p_2) \neq c(x+p_2))\Big)\\
z &= \Pr_{x\sim P}\Big(A_1 \wedge A_2\Big) = \Pr_{x\sim P}\Big((\exists p_1 \text{ s.t. } h_1(x+p_1) \neq c(x+p_1)) \wedge (\exists p_2 \text{ s.t. } h_2(x+p_2) \neq c(x+p_2))\Big)\\
w &= \Pr_{x\sim P}\Big(\neg A_1 \wedge A_2\Big) = \Pr_{x\sim P}\Big((\nexists p_1 \text{ s.t. } h_1(x+p_1) \neq c(x+p_1)) \wedge (\exists p_2 \text{ s.t. } h_2(x+p_2) \neq c(x+p_2))\Big)
\end{aligned}
$$

We get that equation (3) becomes

$$\mu_2 = \frac{z}{2\mu_1} + \frac{w}{2(1-\mu_1)} \tag{4}$$

And we also have that

$$\mu_1 = \Pr_{x\sim P}\Big(\exists p_1 \text{ s.t. } h_1(x+p_1) \neq c(x+p_1)\Big) = y + z \tag{5}$$

$$1-\mu_1 = \Pr_{x\sim P}\Big(\nexists p_1 \text{ s.t. } h_1(x+p_1) \neq c(x+p_1)\Big) = x + w \tag{6}$$

$$\Pr_{x\sim P}\Big(A_{1,2}\Big) = \Pr_{x\sim P}\Big((A_1 \wedge \neg A_2) \vee (\neg A_1 \wedge A_2)\Big) = y + w \tag{7}$$

If we solve (4),(5),(6) for z and w we get

$$z = \mu_1 - y \tag{8}$$

$$w = (2\mu_2 - 1)(1-\mu_1) + \frac{y(1-\mu_1)}{\mu_1} \tag{9}$$

For the case where h_2 and h_1 disagree we define the event $A_3 = \{x : \exists p \text{ s.t } h_3(x+p) \neq c(x+p)\}$ and:

$$
\begin{aligned}
\Pr_{x\sim P}\Big(\exists p_1.\exists p_2.\exists p_3 \text{ s.t. } (h_1(x+p_1) \neq h_2(x+p_2)) \wedge (h_3(x+p_3) \neq c(x+p_3))\Big) &= \Pr_{x\sim P}\Big(A_{1,2} \wedge A_3\Big)\\
&= \sum_{x\in(A_{1,2}\wedge A_3)} \Pr_{x\sim P}\Big(x\Big)\\
&= \sum_{x\in A_3} \Pr_{x\sim P}\Big(A_{1,2}\Big) \Pr_{x\sim P_3}\Big(x\Big)\\
&= \sum_{x\in A_3} (w+y) \Pr_{x\sim P_3}\Big(x\Big) = (w+y) \Pr_{x\sim P_3}\Big(A_3\Big)\\
&= (w+y)\mu_3
\end{aligned} \tag{10}
$$

Given equations (8),(9),(10) we can find a bound for the error of h_{ens} in the adversarial distribution P_{adv}.

$$\begin{aligned}
\epsilon_{\text{adv},P}(h_{\text{ens}}) &= \Pr_{x\sim P}\Big(\exists p \text{ s.t. } \text{sign}(h_1(x+p), h_2(x+p), h_3(x+p)) \neq c(x+p)\Big) \\
&\leq \Pr_{x\sim P}\Big(\exists p_1.\exists p_2.\exists p_3 \text{ s.t. the majority of predicates } (h_i(x+p_i) \neq c(x+p_i)) \text{ for } i=1,2,3 \text{ are true}\Big) \\
&= \Pr_{x\sim P}\Big(A_1 \wedge A_2\Big) + \Pr_{x\sim P}\Big(A_{1,2} \wedge A_3\Big) \\
&= z + \mu_3(w+y) \\
&\leq z + \mu(w+y) \\
&= \mu_1 - y + \mu(2\mu_2 - 1)(1-\mu_1) + \frac{y(1-\mu_1)}{\mu_1}\mu + y\mu \\
&= \mu(2\mu_2-1)(1-\mu_1) + \frac{y(\mu-\mu_1)}{\mu_1} + \mu_1 \qquad (11)
\end{aligned}$$

Because $\mu_2, \mu_1 \leq \mu < 1/2$ we have that $(2\mu_2 - 1)(1-\mu_1) \leq (2\mu-1)(1-\mu)$. Also from (5) we have that $y/\mu_1 \leq 1$, which implies that $\frac{y(\mu-\mu_1)}{\mu_1} \leq (\mu - \mu_1)$. So continuing from equation (11) we get that

$$\begin{aligned}
\epsilon_{\text{adv},P}(h_{\text{ens}}) &\leq \mu(2\mu_2-1)(1-\mu_1) + \frac{y(\mu-\mu_1)}{\mu_1} + \mu_1 \\
&\leq \mu(2\mu-1)(1-\mu) + (\mu-\mu_1) + \mu_1 \\
&= 3\mu^2 - 2\mu^3 = \mu(3\mu - 2\mu^2)
\end{aligned}$$

And for $0 < \mu < 1/2$ we get $3\mu - 2\mu^2 < 1$ so $\epsilon_{\text{adv},P}(h_{ens}) \leq \mu(3\mu - 2\mu^2) < \mu$. Which concludes the proof of inequality (1).

Now we want to prove the bound of the error on the original distribution given equation (2).
First we have that $\Pr_{x\sim P}\Big(h_1(x) \neq h_2(x) \wedge E\Big) \leq \Pr_{x\sim P}\Big(A_{1,2} \wedge E\Big)$ for any event E , because the event $A_{1,2}$ includes the case where $h_1(x) \neq h_2(x)$ so the event $A_{1,2} \wedge E$ will include the event $(h_1(x) \neq h_2(x)) \wedge E$. So we get that:

$$\begin{aligned}
\Pr_{x\sim P}\Big((h_2(x) \neq h_1(x)) \wedge (h_3 \neq c(x))\Big) &\leq \Pr_{x\sim P}\Big(A_{1,2} \wedge (h_3(x) \neq c(x))\Big) \\
&= \sum_{x\in(A_{1,2}\wedge(h_3(x)\neq c(x)))} \Pr_{x\sim P}\Big(x\Big) \\
&= \sum_{x\in(h_3(x)\neq c(x))} \Pr_{x\sim P}\Big(A_{1,2}\Big) \Pr_{x\sim P_3}\Big(x\Big) \\
&= (w+y) \Pr_{x\sim P_3}\Big(h_3(x) \neq c(x)\Big) = (w+y)\epsilon_3 \qquad (12)
\end{aligned}$$

Using equation (12) we can follow the same logic as before to get the following

$$\begin{aligned}
\epsilon_P(h_{\text{ens}}) &= \Pr_{x\sim P}\Big(\text{the majority of predicates } h_i(x) \neq c(x) \text{ for i=1,2,3 are true}\Big) \\
&= \Pr_{x\sim P}\Big(h_1(x) = h_2(x) \neq c(x)\Big) + \Pr_{x\sim P}\Big((h_1(x) \neq h_2(x)) \wedge (h_3(x) \neq c(x)\Big) \\
&\leq \Pr_{x\sim P}\Big(A_1 \wedge A_2\Big) + \Pr_{x\sim P}\Big(A_{1,2} \wedge (h_3(x) \neq c(x))\Big) \\
&= z + (w+y)\epsilon_3 \\
&\leq z + (w+y)\epsilon \\
&= \mu_1 - y + \epsilon(2\mu_2 - 1)(1-\mu_1) + \frac{y(1-\mu_1)\epsilon}{\mu_1} + y\epsilon \\
&= \epsilon(2\mu_2 - 1)(1-\mu_1) + \frac{y(\epsilon - \mu_1)}{\mu_1} + \mu_1 \\
&\leq \epsilon(2\mu_2 - 1)(1-\mu_1) + \epsilon \\
&\leq \epsilon(2\mu - 1)(1-\mu) + \epsilon = \epsilon(3\mu - 2\mu^2) < \epsilon
\end{aligned}$$

Which concludes the proof of inequality (2).

As a result we have shown that we can use L, that ensures weak learnability ($\epsilon_P(h) \leq \epsilon < 1/2$) and weak robustness ($\epsilon_{\text{adv},P}(h) \leq \mu < 1/2$), to create the ensemble $h_{\text{ens}} = sing(h_1, h_2, h_3)$ that has both

$$\epsilon_P(h_{\text{ens}}) \leq \epsilon(3\mu - 2\mu^2) < \epsilon$$

and

$$\epsilon_{\text{adv},P}(h_{\text{ens}}) \leq \mu(3\mu - 2\mu^2) < \mu$$

4 Experiments on Image Classification

In this section we adapt the process described in section 3 in order to create an ensemble of classifiers that achieves less error from the individual models, both in the original distribution and the adversarial distribution. As a proof of concept we provide results on the MNIST dataset [18], which is a dataset of greyscale images of handwritten digits.

We use the Projected Gradient Descent (PGD) attack [15] as the adversary. This methods creates adversarial attacks by maximizing the loss of the classifier for the correct category. It achieves that by performing projected gradient descent that clips the intermediate results of the optimization so that the perturbation norm is bounded ($\|p\|_\infty \leq \epsilon_p$) and the perturbed images are inside the domain of the function we are trying to learn ($x + p \in D$). This attack method is one of the most common attacks that uses the local first order information of our network. It produces relative strong adversarial examples which can be efficiently computed.

Because in the case of the MNIST dataset we want to classify between 10 different classes, our classifier is a simple CNN that outputs a vector that for each class contains the classifier's confidence that the input belongs to this class. This vector is the output of a softmax layer, so it contains positive values which sum to 1, and it resembles a probability estimate that the input belongs to this class. We will call the output vector, the soft output of the classifier, and we will denote it with h_{soft}. Then the final class that the input is classified into can be found as:

$$h(x) = \underset{i\in\text{Classes}}{\text{argmax}}(h_{\text{soft}}[i])$$

In order to train an ensemble of classifiers for the MNIST dataset we perform the following steps:

- We train classifier h_1 on the original MNIST dataset
- For each point x of the original dataset we use PGD attack in order to find p with $\|p\|_\infty \leq 0.3$ so that h_1 makes a mistake on $x + p$. Then we create set $A_{\text{crit}} = \{x :$ for which PGD found p where $h(x + p) \neq$

$c(x+p)\}$ and set $A_{\text{safe}} = \{x : \text{for which PGD didn't found an adversarial attack}\}$. Then we sample equal samples from A_{crit} and A_{safe} in order to create P_2 dataset. We can do that by taking all the samples of the set with the lowest cardinality (usually A_{crit} because we have $\mu < 1/2$) and then sample for the other set an equal amount of samples

- We train h_2 on P_2
- For each x of the original dataset we try to find p with $\|p\|_\infty \leq 0.3$ so that $h_1(x+p) \neq h_2(x+p)$. We create dataset P_3 by using all the points x where we are successful.
  Remark: In order to find p where $h_1(x+p) \neq h_2(x+p)$ we use the soft output of the two classifiers and we try to minimize the inner product $\langle h_{1,\text{soft}}(x+p), h_{2,\text{soft}}(x+p)\rangle$, by using again the same PGD method that we use for the adversarial attacks.
- We train h_3 on P_3
- We output the final ensemble where $h_{\text{ens,soft}} = (h_{1,\text{soft}} + h_{2,\text{soft}} + h_{3,\text{soft}})/3$

One important detail in the previous process is that when we train each classifier, we want to have adversarial error at most 0.5, in order to satisfy the assumptions of the analysis in section 3. This is not something that is achieved with a standard training in the case of CNNs. So to get an adversarial error less than 0.5 we also perform adversarial training [15], which as we can see in table 1 gives us classifiers with adversarial errors ranging from 0.21 to 0.24.

	ϵ_P	$\epsilon_{\text{adv},\mu}$
h_1	0.2149	0.0104
h_3	0.243	0.0093
h_3	0.227	0.0089
h_{ens}	0.1603	0.0089

Table 1: Error rates of the individual classifiers and the ensemble, in the test set (ϵ_P) and the adversarial test set ($\epsilon_{\text{adv},P}$)

In table 1 we can see that the ensemble of classifiers achieves less error in comparison to the individual classifiers, both in the case of the error in the original test set and in the case of the error when we allow an adversary to perturb the test set. Only h_3 achieves same error as the ensemble in the original test set, but we can see that even then, h_3 is far less robust to adversarial attacks compared to the ensemble.

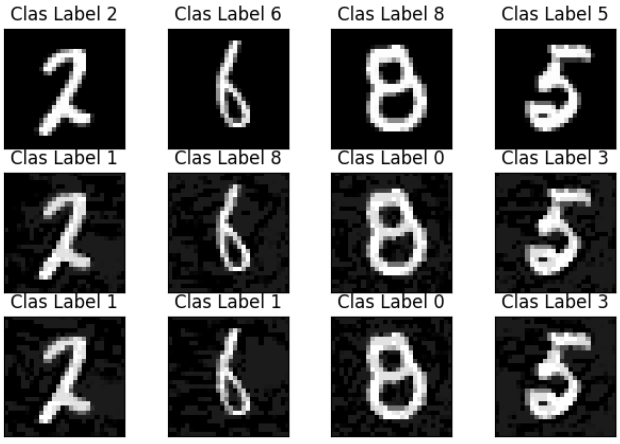

Figure 1: First Row: Original images from the MNIST test set, Second Row: Adversarial images that are misclassified by h_1, Third Row: Adversarial images that are misclassfied by h_{ens}

Also in Figure 1 we can see images from the test set where we can find adversarial examples for both h_1 and h_{ens}. As a qualitative assessment we can see that the adversarial examples for both h_1, h_{ens} have similar visual properties (both appear to have small background noise). So we can create adversarial examples of similar quality (same visual characteristics) for both the original classifier h_1 and the more robust classifier h_{ens}. The difference is that in the case of the ensemble, the images that can be perturbed to create an adversarial example appear with lower frequency in the data distribution.

These results provide evidence that the ensemble process described and analyzed in section 3 can provide improvements in adversarial robustness when it is applied in an experimental setting. Of course as stated before these results don't provide any proof or certification because we used only one kind of attack and it is always possible that a more sophisticated adversary can find more adversarial examples in the case of the ensemble. Regardless, when combined with the analysis done in section 3 these results provide a proof of concept that this ensemble method can be used in a practical setting to make models more robust to adversarial attacks.

The implementation code and instructions on how to replicate the above results can be found in this Github Repository

5 Conclusion

In this project we showed that in the setting of binary classification, if we have an algorithm L that provides "weak" adversarial robustness (adversarial error less than $1/2$), we can use it to create an ensemble of classifiers with less adversarial error compared to the adversarial error of the individual classifiers. In addition, we showed that this process also improves the error bound in the original distribution. Along with these theoretic results we provide experimental evidence of these improvements in the setting of image classification.

These results indicate that using the boosting framework to improve the adversarial robustness of our models is a promising direction. Future work can focus on investigating whether this improvements appear in more complicated datasets and when more sophisticated adversaries are used. Also, another interesting direction is to adapt to the case of adversarial examples, other boosting algorithms such as the widely used AdaBoost algorithm [4].

References

[1] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. and Fergus, R. "Intriguing properties of neural networks." arXiv preprint arXiv:1312.6199. (2013)

[2] Zhang, C., Zhang, H. and Hsieh, C.J. "An Efficient Adversarial Attack for Tree Ensembles." Advances in Neural Information Processing Systems, 33. (2020)

[3] Biggio, B., Corona, I., Nelson, B., Rubinstein, B.I., Maiorca, D., Fumera, G., Giacinto, G. and Roli, F. "Security evaluation of support vector machines in adversarial environments." In Support Vector Machines Applications (pp. 105-153). Springer, Cham. (2014)

[4] Freund, Y., Schapire, R.E. "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting", Journal of Computer and System Sciences. (1997)

[5] Schapire, R.E. "The strength of weak learnability." Machine learning, 5(2), pp.197-227. (1990)

[6] Pang, T., Xu, K., Du, C., Chen, N. and Zhu, J. "Improving adversarial robustness via promoting ensemble diversity." arXiv preprint arXiv:1901.08846 (2019)

[7] Ben, I., Javad, M., Karg, M., Scharfenberger, C. and Wong, A. "SANE: Exploring Adversarial Robustness With Stochastically Activated Network Ensembles" In Proceedigs of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops. (2019)

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

[9] Song, Y., Kim, T., Nowozin, S., Ermon, S. Kushman N. "PixelDefend: Leveraging Generative Models to Understand and Defend against Adversarial Examples", In Proceedings of the International Conference on Learning Representations. (2018)

[10] Andriushchenko, M. and Hein, M. "Provably robust boosted decision stumps and trees against adversarial attacks." In Advances in Neural Information Processing Systems (pp. 13017-13028). (2019)

[11] Jha, S., Jang, U., Jha, S. and B. Jalaian, "Detecting Adversarial Examples Using Data Manifolds," MILCOM 2018 - 2018 IEEE Military Communications Conference (MILCOM), (2018)

[12] Pang, T., Du, C., Dong, Y. and Zhu, J., "Towards robust detection of adversarial examples." In Advances in Neural Information Processing Systems , (2018)

[13] Tian, S., Yang, G. and Cai, Y., " Detecting adversarial examples through image transformation." In AAAI. (2018)

[14] Athalye, A., Carlini, N. and Wagner, D., "Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples" In Proceedings of the 35th International Conference on Machine Learning, (2018)

[15] Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. "Towards deep learning models resistant toadversarial attacks." In Proceedings of the International Conference on Learning Representations (ICLR), (2018)

[16] Carlini, N. and Wagner, D. "Towards evaluating the robustness of neural networks." In 2017 IEEE Symposium on Security and Privacy (SP), (2017)

[17] Krizhevsky, A. and Hinton, G., "Learning multiple layers of features from tiny images." (2009)

[18] LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P., "Gradient-based learning applied to document recognition." Proceedings of the IEEE, 86(11), pp.2278-2324. (1998)