Adversarial Robustness via Model Ensembles
15-400 Research Practicum (Spring 2021)

Akhil Nadigatla [1] **Arun Sai Suggala** [1] **Pradeep Ravikumar** [1]

Abstract

Deep neural networks are well-known to be vulnerable to adversarial attacks that make tiny imperceptible changes to their inputs and yet lead the networks to misclassify them. Consequently, several recent works have proposed techniques for learning models/neural networks that are robust to such attacks. Most of the existing approaches for designing robust models are designed to output a single model to defend against all possible moves of the adversary. However, a single model usually does not have enough power to defend against all possible adversary moves, resulting in poor performance. In this work, we investigate the effects of using a weighted ensemble of models to see if it might be able to better defend against adversarial attacks. Towards this end, we present empirical results showing that model ensembles created in certain ways do lead to statistically significant improvements in adversarial robustness. In particular, the best model ensemble provided approximately a 1% improvement in accuracy when compared to the best-performing individual model taken into consideration in these experiments.

1. Introduction

Over the past decade or so, neural networks have evolved into prominent tools used to conduct various machine learning tasks. While their versatility (coupled with the simplicity of their underlying theory) makes them highly useful, most of them are plagued by *adversarial examples* (Szegedy et al., 2014). These are inputs to machine learning models that have been slightly *perturbed* in a manner that causes them to misclassified, oftentimes with a high degree of confidence. What is even more intriguing is the fact that state-of-the-art deep neural networks have also been shown to be vulnerable to these *attacks* (Nguyen et al., 2015) and that the presence of these examples is not restricted to neural networks alone - for example, they have been witnessed in Support Vector Machines (Biggio et al., 2014).

A common solution proposed to make models somewhat resistant to these perturbed inputs is to conduct *adversarial training* (Gu & Rigazio, 2015). However, it is important to note that no single cause has been identified to be behind the existence of adversarial examples in machine learning models. Research conducted into this question attributes the phenomenon to (among others) the linearity of models, the 'single sum' nature of the constraints used in training a majority of such models, and the complex relationships between the geometry of the categories (Li et al., 2020). What this means is that a number of techniques have been proposed for performing adversarial training, each approaching the problem from a different perspective (Chakraborty et al., 2018).

In general, these techniques present the problem of designing robust models as a two-player game between a *learner* and an *adversary*. In this game, the goal of the learner is output a model which performs well against the worst possible move of the adversary (in this case, the specific adversarial attack being applied). Meanwhile, the goal of the adversary is to design attacks which cause the learner to output the worst performing model. A large number of the approaches for training adversarially robust models can be classified as heuristic techniques for solving this game.

Despite their popularity, these heuristic approaches come with several drawbacks. Firstly, they often to sub-optimal solutions and are not guaranteed to output the best possible model that is robust to adversarial attacks (even when given infinite compute power) (Liu et al., 2020). Moreover, these techniques produce a single model in the hope that it is resistant to all possible adversarial attacks. However, one model does not have enough power to counter the set of all moves that an adversary can make, leading to a model with poor overall performance.

[1]Carnegie Mellon University, Pittsburgh PA. Correspondence to: Akhil Nadigatla <anadigat@andrew.cmu.edu>.

Proceedings of the 38^{th} International Conference on Machine Learning, Online, PMLR 139, 2021.

1.1. Related Work

As aforementioned, a number of training techniques have been proposed to realize models that are robust against adversarial attacks. For our purposes, we will focus on the following:

- Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2015) makes use of the gradients of the neural network to generate adversarial examples, which are then used to construct the training data set for a model.
- Projected Gradient Descent (PGD) (Madry et al., 2019) presents the objective of the learner-adversary game as a saddle-point problem consisting of a composition of an inner maximization (of the loss across the set of all possible perturbations) goal pursued by the adversary and an outer minimization (of expected loss across all possible points from the data distribution) goal sought by the learner.
- Another approach considers a convex outer approximation optimizing for minimum worst case loss on the set of activations that can attained via a norm-bounded (ℓ_∞ in this case) perturbation (Wong & Kolter, 2018). For convenience, this technique will be referred to as 'CONVEX' from hereon.
- Sensible adversarial training (referred to herein as 'SENSE') restrict adversarial perturbations so as to not cross a Bayes decision boundary in addition to the ℓ_∞ ϵ-ball constraint, which ensures that the perturbation ball is specific to every single data point (Kim & Wang, 2020).

The choice of models was deliberate. FGSM and PGD are very popular, well-documented adversarial training techniques that were among the earliest proposed. Comparatively, CONVEX and SENSE are much more recent approaches. We also avoided examining variants of existing techniques - like 'PGD with Output Diversified Initialization' (Tashiro et al., 2020) - to avoid skewing the ensemble towards a particular type of model.

1.2. Contributions

In this project, we specifically address the second drawback mentioned for these heuristic-based techniques. In particular, instead of outputting a single model hoping to be robust against a plethora of adversarial attacks, we consider an ensemble of models.

In particular, we consider models trained using the four training techniques mentioned above on the MNIST data set and compare their performance to ensemble models. The ensembles, in most cases, performed better than their individual counterparts and, on some attacks, portrayed close to a 1% improvement in accuracy.

In addition, we also present the structure to an alternative formulation of this problem with respect to ensembles and potential on-line algorithms to solve them. While this has not been experimented on yet, it provides a foundation on which to base future directions for this work.

2. Preliminaries

Let us formally define the problem setting in adversarial training (Suggala et al., 2019). Let $S_n = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ be the training data set, where $\mathbf{x}_i \in \mathbb{R}^d$ denotes the feature vector of the i^{th} data point and $y_i \in \{1, 2, \dots K\}$ denotes its class label. The *adversarial risk* of a classifier $f_\theta : \mathbb{R}^d \to \{1, 2, \dots K\}$ is defined as:

$$\hat{R}_{n,\text{adv}}(f_\theta) = \frac{1}{n}\sum_{i=1}^{n}\left[\max_{\mathbf{z}\in\rho(\mathbf{x}_i)} \ell_{0-1}(f_\theta(\mathbf{z}), y_i)\right].$$

where ℓ_{0-1} is the $0/1$ loss which is defined as:

$$\ell_{0-1}(y_1, y_2) = \begin{cases} 1, & \text{if } y_1 = y_2 \\ 0, & \text{otherwise} \end{cases}.$$

and the mapping ρ defines a set $\rho(\mathbf{x}) \subseteq \mathbb{R}^d$ for every $\mathbf{x}$. The adversary can map an unperturbed point $\mathbf{x}$ to any perturbed point $\mathbf{z} \in \rho(\mathbf{x})$. A popular choice for $\rho(\mathbf{x})$ is $\{\mathbf{z} : \|\mathbf{z} - \mathbf{x}\|_2 \leq \epsilon\}$. Given S_n, the goal of the learner is to learn a classifier $f_\theta, \theta \in \Theta$ with small adversarial risk $\hat{R}_{n,\text{adv}}(f_\theta)$. Here, $\{f_\theta, \theta \in \Theta\}$ could be the set of all neural networks of certain depth and width. This results in the following objective:

$$\min_{\theta\in\Theta} \frac{1}{n}\sum_{i=1}^{n}\left[\max_{\mathbf{z}\in\rho(\mathbf{x}_i)} \ell_{0-1}(f_\theta(\mathbf{z}), y_i)\right].$$

Note that this optimization problem is a discrete optimization problem, as it involves $0/1$ loss. Solving such optimization problems is often computationally intractable. Hence, a common practice in machine learning is to replace the $0/1$ loss with a convex surrogate loss function $\ell(f_\theta(\mathbf{x}), y)$, such as cross-entropy loss. This results in the following training objective:

$$\min_{\theta\in\Theta} \frac{1}{n}\sum_{i=1}^{n}\left[\max_{\mathbf{z}\in\rho(\mathbf{x}_i)} \ell(f_\theta(\mathbf{z}), y_i)\right].$$

This objective can equivalently be written as:

$$\min_{\theta\in\Theta} \max_{\mathbf{z}_1,\dots,\mathbf{z}_n} \frac{1}{n}\sum_{i=1}^{n} \ell(f_\theta(\mathbf{z}_i), y_i)$$
$$\text{s.t. } \forall i \in [n], \mathbf{z}_i \in \rho(\mathbf{x}_i). \tag{1}$$

This min-max problem is also called a "two-player zero sum game". As prior mentioned, most popular adversarial training techniques use heuristics to solve this game.

Table 1. Classification accuracies of the six different models on a variety of attacks. The 'Natural' accuracy refers to the (average) accuracy of the model on unperturbed MNIST examples, while the 'Adversarial' accuracy highlights the test accuracy of each model post training.

MODEL	NATURAL	ADVERSARIAL	FGSM	CW	SIMBA	PGD	MOMENTUM	BEST
PGD	98.46	93.45	92.35	94.18	98.47	93.45	92.52	92.53
FGSM	98.04	92.14	92.07	94.17	98.05	92.98	91.90	91.92
CONVEX	95.84	92.29	87.48	91.58	95.82	88.44	87.43	87.40
SENSE	98.51	94.18	94.73	95.01	98.49	95.39	95.18	94.71
ENS1	98.47	95.25	95.08	95.27	98.47	96.09	95.06	94.89
ENS2	99.02	96.57	95.78	96.78	98.97	96.58	95.87	95.79

3. Approach

We trained four models, one for each of the four training techniques mentioned above (FGSM, PGD, CONVEX, and SENSE), on the MNIST data set. In each case, the perturbations applied to the training examples were ℓ_∞ norm-bounded with $\epsilon = 0.3$. The number of attack steps was 40 and the number of epochs was 90. Moreover, the attacks were untargeted, i.e. the 'adversary' does not attempt to misguide the model into predicting a specific class for a given input. Given that the models were being trained for image recognition tasks, the architecture used for the (convolutional) neural network was LeNet5 (Lecun et al., 1998).

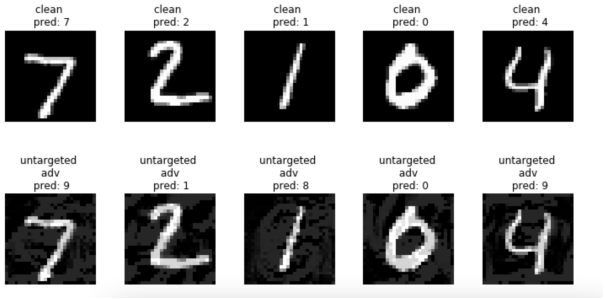

Figure 1. Sample of perturbations applied to examples from the MNIST data set via a PGD attack under the specified parameters.

These four trained models were then used as *experts* in the training of the ensembles. Two ensembling process was approached in two different ways:

- Weighted linear combination of individual model outputs (Clemen & Winkler, 1999), whereby each of the ten labels (for each of the four models) has a weight attached to it. This implies that the training process learns forty different weights. This method will be referred to in this article as 'ENS1.'
- Simple weighted majority vote (Blum, 1998), whereby each of the four models has a weight associated to it. This implies that the training process learns four different weights. This method will be referred to in this article as 'ENS2.'

What this essentially means is that ENS1 was solving for the objective specified by Equation (2) while ENS2 was solving Equation (3):

$$\min_{\mathbf{w}_1 \dots \mathbf{w}_K} \max_{\mathbf{z}_1,\dots,\mathbf{z}_n} \frac{1}{n} \sum_{i=1}^{n} \ell \left(\sum_{k=1}^{K} \mathbf{w}_k f_{\theta_k}(\mathbf{z}_i), y_i \right) \quad \text{s.t. } \forall i \in [n], \mathbf{z}_i \in \rho(\mathbf{x}_i), \; \forall k \in [K], \mathbf{w}_k \in \mathbb{R}^{10}. \tag{2}$$

$$\min_{w_1 \dots w_K} \max_{\mathbf{z}_1,\dots,\mathbf{z}_n} \frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{K} w_k \ell(f_{\theta_k}(\mathbf{z}_i), y_i) \quad \text{s.t. } \forall i \in [n], \mathbf{z}_i \in \rho(\mathbf{x}_i), \; \forall k \in [K], w_k \geq 0, \sum_{k=1}^{K} w_k = 1. \tag{3}$$

The ensembles were trained on a ℓ_∞ norm-bounded PGD attack with $\epsilon = 0.3$. As with the individual models, number of attack steps was 40, the number of epochs was 90, and the attacks were untargeted. The same architecture (LeNet5) was used to construct the ensemble models.

4. Experiments

Once the training and testing of the six models was complete, they were subjected to experiments, during which each data point from the test set was perturbed using five different types of evasion attacks: FGSM, PGD, Carlini-Wagner (CW) (Carlini & Wagner, 2017), Simple Black-box Adversarial (SimBA) (Guo et al., 2019), and Iterative Momentum (Momentum) (Dong et al., 2018).

An additional attack was also considered (called here 'BEST'). This is simply performed by applying to each data point the attack (out of the other five) that resulted in the greatest cross-entropy loss.

4.1. Results

Ten rounds of experiments were performed, during which the accuracy of the six models against the six attacks was observed. The average values of these are shown in Table 1.

These results show that both ensembles ENS1 and ENS2 performed better than their individual counterparts across most (if not all) measurements. Between the two ensembles, ENS2 exhibited even better performance.

The most significant result is the fact that ENS2's accuracy on the BEST attack exceeded that of the best single model (SENSE) by 1.08%. The superiority of the ensembles on the BEST attack is a strong indication that ensembling is likely to output more robust machine learning models than models trained on a single attack.

5. Surprises and Lessons

The results produced by our experiments draw a lot of follow-up questions to mind. While it does seem intuitive that single models are likely to perform worse than the sum of their parts, the methods used to learn the ensemble models here are not very complicated. However, the accuracy gain witnessed due to this form of ensembling is quite substantial in the context of current adversarial research. Therefore, it will be fascinating to determine the actual reasons behind model ensembles' performance improvements, at least in this case.

Another surprise that we encountered was the volume of (or rather, the lack thereof) literature on non-heuristic adversarial training techniques. Given that heuristic techniques can be provably sub-optimal - for example, PGD runs the risk of converging to a local optimum rather than a global one - we had expected more research to have been conducted on techniques with stronger guarantees. It seems like the convenience provided by heuristic techniques (in terms of ease of implementation and efficiency) outweighs the potential downsides.

Personally, this experience has been beyond eye-opening. If not for this project, I never would have grasped the sheer scale of machine learning research being conducted in institutions around the world and, more importantly, the number of questions that remain unanswered in this space. Adversarial robustness constitutes a mere fraction of the work being done in machine learning robustness and optimization, and I am glad that I was able to contribute towards bringing forth yet another unresolved question to the mix. I was also able to better appreciate the time and effort researchers need to dedicate towards meticulously performing the scientific process, and I am in awe of scholars who publish papers on a regular basis.

6. Conclusion

In this work, we attempted to overcome the shortcomings caused by using heuristic techniques to perform adversarial training by focusing on model ensembles. We trained four models using existing training techniques and used them as components in training the model ensemble. The ensembles were learned in two different ways: one being simple weighted majority and the other was weighted linear combination of individual model outputs. Testing these models on five different types of evasion attacks revealed that the ensembles performed better (if not equally as well) when compared to their individual models. The best ensemble was the one formed using simple weighted majority, which showed approximately a 1% improvement in accuracy, especially on the BEST attack.

7. Future Work

7.1. Data Sets

In our work, the only data set that we worked with during the experiments was MNIST (due to its convenience as well as its prevalence in literature). However, in order to further substantiate the above results, we need to run similar experiments on the CIFAR-10 and ImageNet data sets, which are markedly larger and more complex. This may help identify any shortcomings of the proposed ensembling methods, especially given their technical simplicity.

7.2. FTPL Approach

In section 2, we showed the conventional formulation for the min-max game simulated during adversarial training. Here, instead of solving the game expressed by the Equation (1), we propose the following, more relaxed game:

$$\min_{P \in \mathcal{P}_\Theta} \max_{\mathbf{z}_1, \ldots, \mathbf{z}_n} \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_{\theta \sim P} \left[\ell(f_\theta(\mathbf{z}_i), y_i) \right] \\ \text{s.t. } \forall i \in [n], \mathbf{z}_i \in \rho(\mathbf{x}_i). \tag{4}$$

We can design principled techniques for solving the relaxed game in Equation (4) by relying on algorithmic tools developed in game theory.

Remark 1. *Since the domain of the minimization player in Equation* (4) *is much bigger than the domain of the minimization player in Equation* (1)*, one might think that Equation* (4) *is much harder to solve. However, it turns out that solving Equation* (4) *is no harder than solving Equation* (1)*.*

A popular approach for solving Equation (4) is to rely on online learning algorithms (Cesa-Bianchi & Lugosi, 2006). Here, the minimization player (i.e. the *learner*) and the maximization player (i.e. the *adversary*) play a repeated game

Algorithm 1 FTPL-based algorithm for Equation (4)

Input: parameters of uniform distribution η_1, η_2, number of iterations T
Initialize $\theta_0, \{\mathbf{z}_{0,i}\}_{i=1,\dots n}$
for $t = 1 \dots T$ **do**
  Compute θ_t, minimizer's move, as:
  (i) Generate a random vector σ from uniform distribution over hyper-cube $[0, \eta_1]^D$, where D is the dimension of θ.
  (ii) Solve the following problem:

$$\theta_t = \underset{\theta \in \Theta}{\operatorname{argmin}} \sum_{s=0}^{t-1} R(\theta, \{\mathbf{z}_{s,i}\}_{i=1}^n) - \langle \theta, \sigma \rangle .$$

  Compute $\{\mathbf{z}_{t,i}\}_{i=1,\dots n}$, maximizer's move, as:
  for $i = 1 \dots n$ **do**
    (i) Generate a random vector σ from uniform distribution over hyper-cube $[0, \eta_2]^d$.
    (ii) Solve the following problem:

$$\mathbf{z}_{t,i} = \underset{\mathbf{z} \in \rho(\mathbf{x}_i)}{\operatorname{argmax}} \sum_{s=0}^{t-1} \ell(f_{\theta_s}(\mathbf{z}), y_i) + \langle \mathbf{z}, \sigma \rangle .$$

  end for
end for
Output: $\{\theta_t\}_{t=1\dots T}, \{\mathbf{z}_{t,i}\}_{i=1,\dots n, t=1\dots T}$.

against each other. Both rely on online learning algorithms to choose their actions in each round of the game with the objective of minimizing their respective regret. Whenever the algorithms used by both the players guarantee sub-linear regret, it can be shown that repeated game play converges to a *Nash Equilibrium*. In our work, we take this route to solve Equation (4).

There are several online learning algorithms that the players can rely on. Algorithm 1 presents one such algorithm for solving Equation (4) which is obtained by making both the players rely on Follow-the-Perturbed-Leader (FTPL) to choose their actions (Suggala & Netrapalli, 2020). To simplify the presentation, in Algorithm 1, we let

$$R(\theta, \{\mathbf{z}_i\}_{i=1}^n) = \frac{1}{n} \sum_{i=1}^{n} \ell(f_\theta(\mathbf{z}_i), y_i).$$

In Algorithm 1, θ_t denotes the move of the minimization player in t^{th} iteration and $\{\mathbf{z}_{t,i}\}_{i=1}^n$ denotes the move of the maximization player in t^{th} iteration.

This approach - as shows - was not investigated by our project. It provides stronger theoretical guarantees on the (potential) performance of the ensemble. Therefore, we believe that it will be worthwhile to run experiments in this direction in the future. Not only will it provide another ensembling mechanism, we also hope that it may shed more light on the concrete reasons why ensembling improves adversarial robustness.

Acknowledgements

I would like to express my gratitude to my mentors, Arun Sai Suggala and Pradeep Ravikumar, for all the support and guidance throughout this project.

References

Biggio, B., Corona, I., Nelson, B., Rubinstein, B. I. P., Maiorca, D., Fumera, G., Giacinto, G., , and Roli, F. Security evaluation of support vector machines in adversarial environments, 2014.

Blum, A. On-line algorithms in machine learning. In *Online algorithms*, pp. 306–325. Springer, 1998.

Carlini, N. and Wagner, D. Towards evaluating the robustness of neural networks, 2017.

Cesa-Bianchi, N. and Lugosi, G. *Prediction, Learning, and Games*. Cambridge University Press, USA, 2006. ISBN 0521841089.

Chakraborty, A., Alam, M., Dey, V., Chattopadhyay, A., and Mukhopadhyay, D. Adversarial attacks and defences: A survey, 2018.

Clemen, R. T. and Winkler, R. L. Combining probability distributions from experts in risk analysis. *Risk Analysis*, 19(2):187–203, 1999. doi: https://doi.org/10.1111/j.1539-6924.1999.tb00399.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1539-6924.1999.tb00399.x.

Dong, Y., Liao, F., Pang, T., Su, H., Zhu, J., Hu, X., and Li, J. Boosting adversarial attacks with momentum, 2018.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples, 2015.

Gu, S. and Rigazio, L. Towards deep neural network architectures robust to adversarial examples, 2015.

Guo, C., Gardner, J. R., You, Y., Wilson, A. G., and Weinberger, K. Q. Simple black-box adversarial attacks, 2019.

Kim, J. and Wang, X. Sensible adversarial learning, 2020. URL https://openreview.net/forum?id=rJlf_RVKwr.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pp. 2278–2324, 1998.

Li, H., Fan, Y., Ganz, F., Yezzi, A., and Barnaghi, P. Verifying the causes of adversarial examples, 2020.

Liu, C., Salzmann, M., Lin, T., Tomioka, R., and Süsstrunk, S. On the loss landscape of adversarial training: Identifying challenges and how to overcome them, 2020.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks, 2019.

Nguyen, A., Yosinski, J., and Clune, J. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images, 2015.

Suggala, A. S. and Netrapalli, P. Follow the perturbed leader: Optimism and fast parallel algorithms for smooth minimax games, 2020.

Suggala, A. S., Prasad, A., Nagarajan, V., and Ravikumar, P. Revisiting adversarial risk, 2019.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks, 2014.

Tashiro, Y., Song, Y., and Ermon, S. Diversity can be transferred: Output diversification for white- and black-box attacks, 2020.

Wong, E. and Kolter, J. Z. Provable defenses against adversarial examples via the convex outer adversarial polytope, 2018.