CS182 Report: Ensembling with Detection Against Adversarial Images

Jeffrey Hong [*1] **Manula Dombagahawatta** [*1]

Abstract

Deep Neural Networks have recently improved to the point where they achieve better-than-human performance in certain vision classification tasks. However, certain inputs that are trivial or natural to human perception can exist or be constructed by means of an adversarial attack, causing networks to misclassify.

We introduce a smart ensemble, an aggregation network over multiple image classifying networks that are trained on different existing adversarial attacks. The ensemble contains a detector, which is itself is a deep neural network that is trained to recognize a variety of adversarial attacks through adversarial detection methods, and uses the detector to weight voting from each ensemble model in response. We investigate variants of the architecture in the aggregation network and the detector model.

1. Introduction

We attempt to achieve a high classification score on a hidden test set of Tiny ImageNet, a subset of the popular ImageNet library containing 200 different classes of images. We are given the information that the test set contains images that may mislead, deliberately or otherwise, a naive image classification model. These images may be examples of what the model would see in real world usage outside of a controlled environment, so improvements to image recognition in this domain would improve performance in systems that see many non-ideal, real-world images.

Given that we are training for a distribution that deviates from the training data, we thought augmenting the dataset with variant examples that are visually consistent with the original data was the best approach. To this end, we chose to use adversarial training since many of the adversarial methods are visually consistent variants as seen in figures 1-3.

[*]Equal contribution [1]Computer Science at UC Berkeley. Correspondence to: Manula Dombagahawatta <manula@berkeley.edu>, Jeffrey Hong <hongjeffrey@berkeley.edu>.

Using the ICML 2021 Template.

However, when models are introduced to the variants, they often lose accuracy with respect to the original distribution. We sought to mitigate this by using an ensemble of models that are experts in specific variants aided by a detector model. The detector would determine the weights assigned to the adversarially trained models and the models trained on the original image distribution based on its evaluation of the input image.

We measure our performance by Top-1 percentage, or the most probable class predicted by our model, by evaluating our model on an adversarially generated test set designed to simulate the hidden test set.

2. Literature Survey and Background

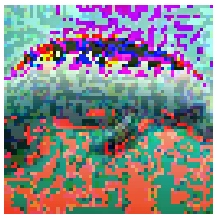


Figure 1. Sample test image of a manatee with FGSM applied.

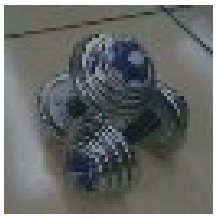


Figure 2. Sample test image of dumbbells with PGD applied.

Szegedy et al., 2013 introduces the idea of "fooling" common neural networks with certain perturbations to a given input, allowing an attacker to change the predicted class output of a model to another class, or in some cases to a class of the attacker's choice (Szegedy et al., 2013). Most common attacks fall into the category of white-box attacks, where the attacker has access to the network being attacked and

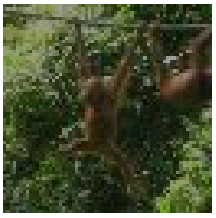


Figure 3. Sample test image of orangutans with Carlini-Wagner applied.

is able to control the inputs and outputs of the network to achieve the desired misclassification. However, there exists a class of attacks known as black-box attacks that assume a more realistic setting, where attackers would not have access to the gradient of the particular model and may have a limited number of queries, partial information about the output classes, or label-only information about the output classes, where the class probabilities are not known at all (Ilyas et al., 2018). Liang et al., 2019 introduces a detection layer that allows classification networks to discern between adversarial and non-adversarial images by addressing adversarial perturbations as noise. This detection layer acts as a pre-processing layer to an image classification model that is repurposed to differentiate between adversarial and non-adversarial images. (Liang et al., 2017)

In contrast, our work explores the use of the detection filter in an ensemble setting. Moreover, we evaluate its impact on weighing the outputs of models trained on adversarial data versus regular models. We also observe this ensemble method's ability to generalize across adversarial and unperturbed images.

2.1. White-box Attacks

Several common white-box attacks are mentioned in literature: a relatively cheap method called the Fast Gradient Sign Method, or FGSM (Fig 1), that performs an approximation of the gradient of the cost function to move away from a given input and towards a desired class input (Goodfellow et al., 2015); Carlini-Wagner, where an approximation of *Lp* distance is used to find a minimum amount of noise to add to an input image to misclassify it (Carlini & Wagner, 2017); and Projected Gradient Descent (Fig 2), an attack that uses the exact gradient of the model to maximize loss with a perturbation constrained to a particular size (Madry et al., 2017).

2.2. Black-box attacks

We also consider black-box attacks; however, many black-box attacks work by adapting using a certain number of queries to the model and analyzing the output, while the hidden set our model is tested on will not similarly change in response to our model responses from previous queries (Ilyas et al., 2018). We also find that training many black-box attacks is generally not feasible given our training time and compute limitations, since on average they take longer on our hardware. Additionally, we hypothesize that white-box attacks in principle should be more powerful as they have access to complete information about the model, so hardening our model against white-box attacks would provide a degree of defense against black-box attacks as well.

2.3. Detection Filter

$$\begin{cases} f_{SQ}(m,n), if |f_{SQ}(m,n) - f(m,n)| \\ \leq |f_{SQ-SF}(m,n) - f(m,n)| \\ else \\ f_{SQ-SF}(m,n) \end{cases}$$

The detection filter proposed Liang et al. where f_{SQ} represents a scalar quantization filter and $f_{SQ-SF}(m,n)$ represents a scalar quantization and spatial smoothing of the pixel at index (m,n).

2.4. Ensembling

Our model incorporates an ensemble of pre-trained EfficientNet models each adversarially hardened against the attacks mentioned above through transfer learning, similar to Ensemble Adversarial Training (Tramer et al., 2018). We demonstrate novel additions in the ensembling approach by ensembling with an aggregating neural network instead of with common methods like majority vote (Zhou et al., 2002). We demonstrate the performance of the aggregating network independently, then with the addition of a separate detection filter network that aims to predict how likely the input image is adversarial in nature (Liang et al., 2017).

2.5. Model Choice

EfficientNet is a state-of-the-art family of models characterized by efficiently balanced scaling of network width, depth, and resolution (Fig 4) (Tan & Le, 2019). We choose EfficientNet as our model to be ensembled for its high performance on the ILSVRC leaderboards, where different model architectures are ranked by performance on ImageNet; many of the models at the top of the leaderboard are variants of EfficientNet, and we believe that performance on ImageNet may transfer to some degree to performance on Tiny ImageNet. EfficientNet models also have lower total numbers of parameters, so we are able to train faster given the limited amount of compute we have access to.

We build upon EfficientNet models pre-trained on ImageNet with transfer learning to adapt the pretrained models to clas-

sify Tiny ImageNet instead, while still including the learned weights from the pretraining process. We remove the final fully connected weights and biases and replace it with a fully connected layer of size 200 instead of 1000 to represent the number of classes, then retrain the network with our given training set of Tiny ImageNet data with Stochastic Gradient Descent. We initially tried the Adam optimizer and were advised in office hours to experiment with different batch sizes and use SGD. We ultimately evaluate the base model on the given data sets for a training accuracy of 0.819 and a validation accuracy of 0.644.

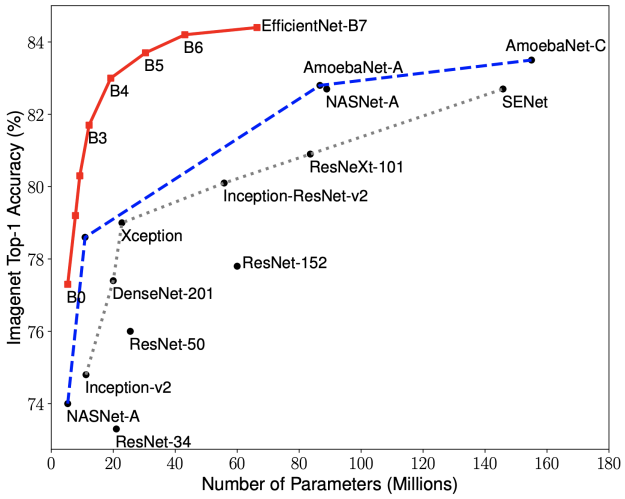


Figure 4. EfficientNet Top-1 performance on ImageNet relative to other common models.

3. Approach

To achieve the robustness required to adapt to deviations in the image distribution, we have chosen to engage in adversarial training augmented with the use of detection methods. Primarily, we got inspiration from the work of Liang et al., 2019 that proposed adversarial images can be detected through denoising methods (Liang et al., 2017). We thought this detection method could be used to provide additional information for weighing model outputs in an ensemble setting.

Our initial approach to ensembling involves training each model in the ensemble on a certain attack with the goal of having each model act as a sort of specializing "expert" for each attack. We theorize that the detector network can then weigh most of the responsibility on a single model if it sees the corresponding attack as input. Additionally, we experimented with the more traditional approach where we split up a set of generated adversarial training data and train each model in the ensemble on a subset of the adversarial images to help the ensemble better generalize across different types of attacks.

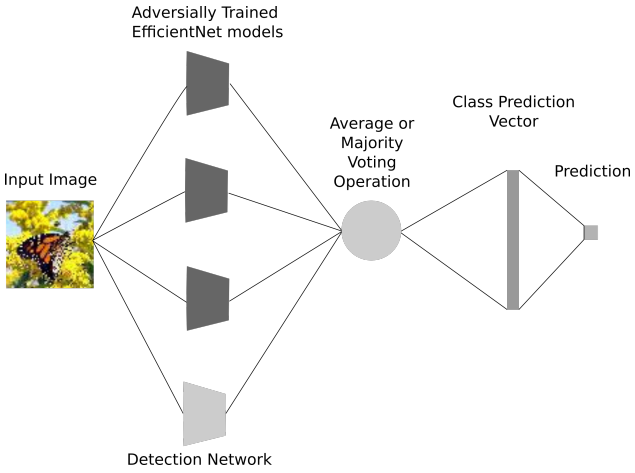


Figure 5. Traditional ensemble architecture from input image to class prediction.

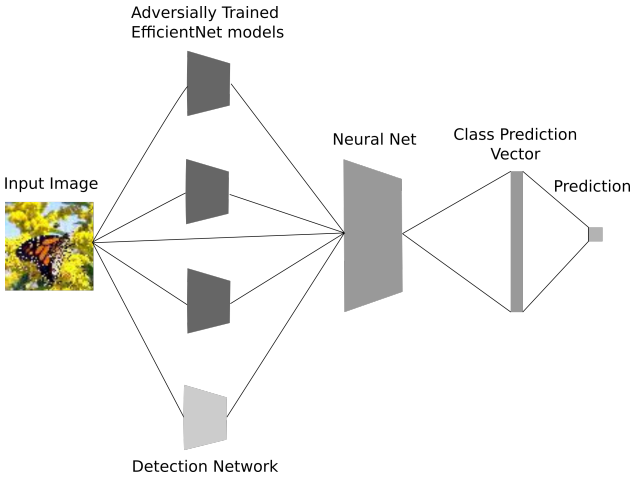


Figure 6. Smart ensemble architecture from input image to class prediction.

3.1. Neural Network Ensemble Model Architecture

We provide the same input image to each model in the ensemble independently after applying normalizing and sizing transforms. The output of each model in the ensemble is concatenated together with the original image itself and the output of the detector network, then fed into the aggregating network of choice, which in our case was a simple set of fully connected layers alternated with ReLU. The aggregating network outputs a vector of class probabilities, which is then argmaxed for the final top-1 class prediction (Fig 6).

Accuracy drastically decreases when any attack is applied to

the baseline EfficientNet models; for example, with FGSM, the baseline has 0.75 accuracy on our holdout set of benign test images with no attack applied, but drops to 0.00 on a set of images with FGSM applied. We apply General Adversarial Training (Szegedy et al., 2013) using the Adversarial Robustness Toolbox, a library containing implementations of several attacks, with each model in the ensemble trained separately on the same training images but with a separate attack applied (Nicolae et al., 2018). After this stage, the models tend to have a tradeoff of accuracy on the benign test set compared to on the adversarial test set, but manage to significantly raise accuracy on any sort of adversarial input. Adding adversarial examples to the training data performs a sort of regularization on the model, allowing it to better generalize to adversarial examples.

The detector network uses adaptive noise reduction (Liang et al., 2017). We regard adversarial perturbations as noise and attempt to filter the perturbation out with a denoising filter, then compare the filtered input with the original input and check for major deviation. The degree of difference between the filtered and original inputs indicates how likely the input is adversarial in nature. Using the magnitude of the attack as an additional input to the ensemble is aimed at helping the network determine how to heavily weight each network; we hypothesize that adversarial attacks have varying degrees or types of perturbation that they apply. The detector outputs softmaxed probabilities of the likelihood the image is adversarial or not.

3.2. Traditional Ensemble Model Architecture

In this variant, we swap out the aggregating neural net for the more traditional option of averaging the model outputs (Fig 5). We use this architecture as a baseline to compare our more novel architectures to. Instead of concatenating the outputs from the adversarially trained EfficientNet models, we apply a mean operator or take the most voted and perform the same operations with the resulting class prediction operator to get a final class prediction. This architecture does not leverage the detection network, so we suspect it may be more rigid in testing or real-world situations if the type of attack was not included in the training data provided to the ensemble.

We experiment with both a regular average and with a weighted average dependent on the detector network. In the second approach, we weigh the base model by the probability that the image is not adversarial from the detector, and weigh the average of all adversarially trained models in the ensemble by the probability that the image is adversarial.

3.3. Testing

As we do not have access to any validation set with a similar distribution to the hidden test set, we test our ensemble on a mix of the given validation set and adversarial images generated by each attack included in the ensemble to simulate the final distribution. This is to hopefully better match the distribution of the hidden test set that our model will be evaluated on, as the distribution of the validation set seems to be close to the distribution of natural-looking images in the training set and may not reflect possible adversarial examples in the hidden test set. We expect that mixing the validation set with adversarial images will form a more robust distribution of images that will more closely match our actual scores on the hidden test set.

Table 1. Pre-adversarial training model accuracies.

ATTACK APPLIED	BENIGN SET	ADVERSARIAL SET
FGSM	0.750	0.000
PGD	0.656	0.000
CARLINI-WAGNER	0.625	0.047

Table 2. Post-adversarial training model accuracies.

ATTACK APPLIED	BENIGN SET	ADVERSARIAL SET
FGSM	0.437	0.434
PGD	0.455	0.264
CARLINI-WAGNER	0.500	0.156

4. Results

4.1. Adversarial training with one attack per model

Applying FGSM, PGD, or Carlini-Wagner on the base models is extremely successful in fooling them (Table 1). In these three common attacks, the base model is essentially unable to correctly classify any significant amount of the input data.

However, after adversarial training, the models drastically improve on the adversarial set, while trading off for performance on the benign set, as analyzed in the Neural Network Ensemble subsection (Table 2).

4.2. Adversarial training across multiple attacks

We trained multiple models on a set of randomly generated adversarial attacks mixed with regular training images, similar to how we constructed our validation set in the Testing subsection. We average on our best 3 models for a a generally better result on both the benign set and the adversarial set (Table 3). We note that in some comparisons to the single-attack trained models, such as with the FGSM-specializing model, performance on adversarial examples is poorer, but there is a positive trade off that may be attributed to better model generalization. It is possible that the models

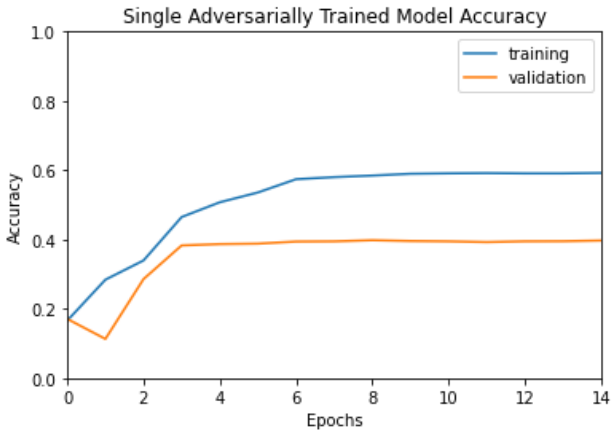
Table 3. Accuracies with training on multiple attacks.

ATTACK APPLIED	BENIGN SET	ADVERSARIAL SET
FGSM, PGD, C-W	.5922	.3971

are correctly classifying mainly FGSM to bring the adversarial set average up, but the benign set accuracy is even higher than the FGSM model benign set accuracy. Thus, we chose to use models trained across multiple attacks in our ensemble submission.

4.3. Detection

Figure 7. Detection training accuracy.



We demonstrate our implementation of an adversarial detection network (Fig 7). We assume that the overall failure of the detection network is attributable to our implementation since it fails to achieve the accuracy shown in the paper by Liang et. al of 94% (Liang et al., 2017). The standard quantization method in Pytorch does not support GPU usage; therefore, we relied on pseudo-quantization methods. These methods might lack the numerical stability of standard quantization methods. The selection of quantization methods requires further testing in the future.

4.4. Ensembling

Table 4. Ensemble accuracy by aggregation type.

ENSEMBLE TYPE	BENIGN SET	ADVERSARIAL SET
NEURAL NET	.0047	.1758
AVERAGE (NO DETECTION)	.4687	.2812
AVERAGE (DETECTION)	.5625	.3750

Our best performing model is the traditional average across ensemble models weighted by our detector outputs as described in the Traditional Ensemble Architecture section. We find that it performs better than our other experiments across the board. With better detection, we could perhaps even improve on these results, as a better detector would lead to more confidence on both the benign and adversarial sets.

5. Conclusion

Our general approach of incorporating the detection network into ensembles generally yielded poor results. In our initial approach of using a fully connected network for aggregation, we thought the network would learn the ideal distribution of weights across the ensemble members with the added information from the detection network. However, this method of aggregation performed very poorly on the mixed data set. Given that it performed relatively well on adversarial examples, we assume that the model specializing in the specific adversarial method encountered produced decisive results one way or the other on whether an image was adversarial. Likely, regular images produced much more ambiguous results across the models. Thus, the fully connected layer was unable to tune weights to account for ambiguity.

We hypothesize that performance could have been improved in a few ways:

First, we ran into a lot of problems with Colab crashing or disconnecting during training, which took up to 3 hours per epoch on the adversarial training part and highly delayed our progress. To decrease the chance of that happening, we ran for 10 epochs; if given the opportunity, we would have liked to run for at least 20 as performance on our validation set improved between 5 and 10 epochs.

Second, we could have introduced more models or types of attacks into the ensemble architecture, which again we were limited by compute. We would have liked to experiment with black box attacks to support our intuition in the introduction.

Third, we were unable to satisfactorily train the detector network. In our more novel architecture, the detector is a key factor that nearly entirely determines how the ensemble will behave and the weights that any layers will learn, so low accuracy on the detector translates to uncertainty and low accuracy in the ensemble itself. Of course, there are more confounding factors, but given time we would like to experiment with more setups for the detector to try to find a model with a suitable accuracy.

Finally, given time we would like to rethink our neural network architecture for the aggregation network. As shown by the results, it was completely unable to converge, and we

would need to put more time into its basic design and implementation beyond what we completed during the project to improve its performance.

6. Team Contributions

50% Manula: Trained the ensemble models, created the detector network and training pipeline, and set up the ensemble

50% Jeffrey: Wrote the initial setup to load the data, created visualizations, wrote the report, and trained base models

References

Carlini, N. and Wagner, D. Towards evaluating the robustness of neural networks. Technical report, University of California, Berkeley, 2017. URL https://arxiv.org/pdf/1608.04644.pdf.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. Technical report, 2015. URL https://arxiv.org/pdf/1412.6572.pdf.

Ilyas, A., Engstrom, L., Athalye, A., and Lin, J. Black-box adversarial attacks with limited queries and information. Technical report, 2018. URL http://proceedings.mlr.press/v80/ilyas18a/ilyas18a.pdf.

Liang, B., Li, H., Su, M., Li, X., Shi, W., and Wang, X. Detecting adversarial image examples in deep networks with adaptive noise reduction. Technical report, 2017. URL https://arxiv.org/pdf/1705.08378v5.pdf.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. Technical report, 2017. URL https://arxiv.org/pdf/1706.06083.pdf.

Nicolae, M.-I., Sinn, M., Minh, T. N., Rawat, A., Wistuba, M., Zantedeschi, V., Molloy, I. M., and Edwards, B. Adversarial robustness toolbox v0.2.2. Technical report, 2018. URL https://openreview.net/forum?id=LjClBNOADBzB.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. Technical report, 2013. URL https://arxiv.org/abs/1312.6199.

Tan, M. and Le, Q. V. Efficientnet: Rethinking model scaling for convolutional neural networks. Technical report, 2019. URL https://arxiv.org/abs/1905.11946.

Tramer, F., Kurakin, A., Papernot, N., Goodfellow, I., Boneh, D., and McDaniel, P. Ensemble adversarial training: Attacks and defenses. Technical report, 2018. URL https://arxiv.org/pdf/1705.07204.pdf.

Zhou, Z.-H., Wu, J., and Tang, W. Ensembling neural networks: Many could be better than all. Technical report, 2002. URL https://cs.nju.edu.cn/zhouzh/zhouzh.files/publication/aij02.pdf.