AdvFreeze: Using Adversarial Examples to Increase Robustness Without Sacrificing Accuracy

Roshan Kumaraswamy, Amy Wang

May 2021

1 Introduction

The objective of this project was to design and implement a robust TinyImageNet classifier that would perform well on a hidden test set, despite unknown test-time perturbation. To evaluate our models, we focused on two metrics: accuracy on a clean dataset and accuracy on an adversarially perturbed dataset, generated using Fast Gradient Sign Method (FGSM)[1]. We propose a new adversarial training process which greatly increases robustness without hurting clean accuracy. Our novel approach achieves this by selectively altering how batch norm layers operate while training on adversarial examples.

We trained an EfficientNet-B0 model with our novel process, achieving a clean validation accuracy of 77.34% and an adversarial validation accuracy of 76.28%, showing a +8.94% clean improvement and a +25.95% adversarial improvement over a vanilla adversarially-trained EfficientNet-B0. The problem of training a robust yet accurate image classifier is important as it will gives us more confidence to deploy CV models in the real world without worrying about possible detriments arising from unforeseen or malicious perturbations. Our work in this paper demonstrates a new way of training models to become more robust against malicious perturbations **without sacrificing accuracy on clean images**.

2 Literature Search & Background

We began by conducting literature research to choose an optimal baseline model architecture, which we determined to be a version of the EfficientNet model created by Luo et al, 2020.[2] EfficientNet uses compound scaling of ConvNets to produce a faster, smaller, and more accurate model and has been widely used for large-scale image recognition across literature[3]. Thus, we decided to use transfer learning on an pretrained EfficientNet model from a Github repository by lukemelas.[4].

We also researched which types of data augmentations perform well with EfficientNet, and came across AutoAugment (Cubuk et al., 2018)[5]. Autoaugment is an automated approach to find optimal data augmentation policies form the data. We found a PyTorch implementation[6] of AutoAugment which included a policy for ImageNet found through AutoAugment.

While searching through different adversarial training schemes, we encountered AdvProp (Xie et al., 2020)[7], which we found particularly interesting. AdvProp hypothesizes that adversarial training often performs poorly because adversarial images come from a different data distribution than clean images, which causes the batch norm layers to learn an overall distribution not exactly matching what the model will encounter at inference time. To work around this problem, the authors propose the use of an auxiliary batch norm used specifically for adversarial data. Clean images and adversarial images are routed through different batch norms, so there is no distribution contamination. At inference time, the auxiliary batch norm is discarded and the clean batch norm is used. The results show that AdvProp achieves an increase in accuracy of the model on clean data. It is important to note that the goal of AdvProp is not to use adversarial images to increase robustness, but rather to treat adversarial images as additional training data to increase clean image accuracy.

[1] https://arxiv.org/abs/1412.6572
[2] https://arxiv.org/pdf/1912.08136v2.pdf
[3] https://arxiv.org/pdf/1409.1556.pdf
[4] https://github.com/lukemelas/EfficientNet-PyTorch
[5] https://arxiv.org/pdf/1805.09501.pdf
[6] https://github.com/DeepVoltaire/AutoAugment
[7] https://arxiv.org/pdf/1911.09665.pdf

3 Initial Approach

For our baseline model, called "Base Model", we began by instantiating a pre-trained EfficientNet-B0 model, freezing all layers except the fully connected output layer, and training the remaining 260,000 unfrozen parameters on the TinyImageNet training set. We decided on a B0 model because it is smaller than the other Efficientnet models, which would allow us to fine-tune faster due to less total parameters. We also chose a model pre-trained on ImageNet because we believed the distribution of images should be similar to those of TinyImageNet. Our initial experimentation involved no data augmentation, which resulted in a significantly higher training accuracy than validation accuracy.

To improve the robustness of our model, we implemented data augmentation using torchvision transforms including RandomResizedCrop, RandomHorizontalFlip, and AutoAugment. We also knew that we wanted to implement some form of adversarial training in order to protect against test-time perturbation. After reading about AdvProp, we noted that the lukemelas EfficientNet repository included the option to use weights pre-trained using AdvProp on ImageNet, so we loaded the models with the AdvProp weights to create our "AdvProp Base Model".

For fine-tuning upon the base models (Base Mode and AdvProp Base Model) , we unfroze all layers of the model, which greatly increased training time. We fine-tuned each model by loading the best weights from initial training and training for an additional 25 epochs.

4 Novel Method

Our novel idea was inspired the concept of AdvProp, which switches out batch norms between clean and adversarial training in order to keep the clean and adversarial image distributions separate. However, the auxiliary (adversarially-trained) batch norms are discarded at inference time because the goal of AdvProp is to use adversarial data to increase clean image accuracy. The findings of AdvProp had useful lessons, though - batch norms are a culprit of reduced clean accuracy when training on adversarial examples.

Unlike AdvProp, our goal is to maximize clean accuracy as well as robustness. From the lessons of AdvProp, we devised a new training scheme in hopes of increasing robustness to adversarial images without sacrificing performance on clean images.Since we expect clean and adversarial images to come from different distributions, we would like to avoid contaminating the learned clean distribution with the distribution of adversarial images. However, we would still like our model to be robust to attacks at inference time. Thus, rather than training two separate batch norms and discarding the adversarially-trained one at inference time, we will freeze the batch norms when training on adversarial images. This will stop the β and γ parameters in batch norms from being updates with gradients from adversarial images and will stop the running mean and variance averages from being updated as well.

In addition to this, we will also normalize adversarial images using the estimated population statistics rather than mini-batch statistics during training. This is because at inference time the model will use population statistics calculated through training. Thus, we would like the model to learn to classify adversarial images using the statistics it would use during inference, as opposed to the mean and variance of the adversarial mini-batch which will not be used at inference time. The algorithm is summarized below:

Algorithm 1: Pseudo code of our novel training approach

Data: A set of clean images with labels;
Result: Model parameters θ
for *training step* **do**
  Sample a clean image mini-batch;
  Compute loss on clean image mini-batch and update model parameters;
  Generate an adversarial image mini-batch by running FGSM on the clean mini-batch;
  Freeze batch-norms and set them to evaluation mode to use population mean and variance;
  Compute loss on adversarial image mini-batch and update model parameters;
end
return θ

5 All Models and Results

The results for all models are summarized in Table 1. These results were generated by evaluating our final models (after training/fine-tuning) on a clean and adversarial dataset. The adversarial dataset was generated using FGSM. Below, we describe each model.

Base Models (including AdvProp and Fine-tuned): Between our 2 baseline models, we found that initializing with AdvProp weights improved performance both during training and on the validation set. Fine-tuning upon both of the baseline models increased accuracy, with the AdvProp initialized model still outperforming the non-AdvProp initialized model.

Base Model + Novel Fine-tuning: For our novel adversarial training method, it did not make sense to initialize with AdvProp weights as our novel approach because our of the way novel method trains on adversarial images. Other parameters in the model will interact with adversarial images normalized in a completely different manner than in AdvProp. Our novel method resulted in our best overall model by far. Compared to Base Model + Fine-tuning, our adversarial accuracy was **+60.99%** higher while also yielding a +1.58% increase in clean accuracy. Compared to the best models in each category, our novel training method achieved only a -0.39% difference in clean accuracy (compared to AdvProp Base model + Fine-tuning) and a **+23.37%** increase in adversarial accuracy (compared to AdvProp Base Model + Adversarial Fine-tuning).

Base Model + Clean and adversarial fine-tuning: To compare these results with adversarial training without our novel approach, we used a similar training scheme but without altering the implementation of batch norms. For each mini-batch, we trained on the clean images and then on the adversarial images, allowing batch norms to operate as they normally do. This experiment was performed by fine-tuning on Base Model. The results were as we expected - while adversarial accuracy increases by +35.04%, we sacrifice clean accuracy. Compared to Base Model + Fine-tuning, this model has a difference of -7.36% in clean accuracy. The reason between this decrease in accuracy is the same as the problem identified in the AdvProp paper, i.e. the batch norms are learning a single overall distribution for both clean and adversarial images.

Base Model + Adversarial for 5 epochs fine-tuning: This model follows an approach briefly mentioned in the AdvProp paper. As a preliminary experiment, the authors showed that by training on adversarial data and then fine-tuning on clean data, clean accuracy will sometimes end up higher than solely training on clean data. We did not have the bandwidth to fully train on adversarial data, but we still tried a similar approach. We first fine-tuned our Base Model with adversarial data for 5 epochs, and then continued training on clean data for another 15 epochs. We found that although the clean accuracy of this method was 1.18% lower than our Base Model + Fine-tuning model, our adversarial accuracy increased by 4.68%. Overall, we did not find this approach particularly useful or interesting compared to our other models.

AdvProp base Model + Adversarial Fine-tuning Since AdvProp Base Model was our best performing base model before fine-tuning, we wanted to try directly increasing adversarial accuracy by fine-tuning on adversarial accuracy. After unfreezing all our model parameters, we trained solely on adversarial training data as suggested in *Towards Deep Learning Models Resistant to Adversarial Attacks* (Madry et al., 2019)[8]. Our results were as expected, though - while we were able to increase robustness of our model, it suffered from a decrease clean accuracy.

Table 1: Final Evaluation Results

Evaluation Results		
Model	Clean Accuracy	Adversarial Accuracy
Base Model	57.39%	2.53%
Base Model +Fine-tuning	75.76%	15.29%
Base Model +Novel Fine-tuning	**77.34%**	**76.28%**
Base Model +Clean and Adversarial fine-tuning	68.40%	50.33%
Base Model +Adversarial for 5 epochs fine-tuning	74.58%	19.97%
AdvProp Base Model	60.19%	1.18%
AdvProp Base Model +Fine-tuning	77.73%	9.66%
AdvProp Base Model +Adversarial Fine-tuning	58.79%	52.91%

[8]https://arxiv.org/pdf/1706.06083.pdf

6 Challenges

One of our biggest challenges, after deciding on a novel method, was the long training time for each experiment. Due to the way we set up adversarial training, training time was almost doubled from clean training because each batch was trained on clean data and then on adversarial data. We both ended up setting up virtual machines through GCP in order to run simultaneous experiments to save time and to ensure that they wouldn't crash if left running for too long.

7 Conclusion

Ultimately, our novel method of freezing batch norms during adversarial training performed well on clean images and significantly better than other models on adversarial images, even compared to regular adversarial training. Fine-tuning our Base Model using our novel training technique yielded a higher clean accuracy (+1.58%), while also greatly increasing adversarial accuracy (+60.99%). Our novel technique also yields both a higher clean and adversarial accuracy than all other methods of adversarial training for robustness that we tried. In the future, we would like to implement AdvProp training (rather than just regular fine-tuning on an model pre-trained with AdvProp) and compare clean accuracy. We would also like to run our novel training method on a larger model, such as an EfficientNet B7, since we expect larger models to learn different distributions better. We hypothesize that for larger models, our novel method may match the clean training accuracy of AdvProp.

8 Team Contributions

We both contributed 50% each to the project. Amy did literature review on well-performing models and data augmentation for ImageNet/TinyImageNet, set up the initial training loop, added functions to assist with the novel idea, ran and evaluated models, and drafted up the final report. Roshan did further literature review to flesh out our novel idea, added adversarial training to our training loop, implemented the novel idea in the loop, ran and evaluated models, and added to the final report.

9 Appendix

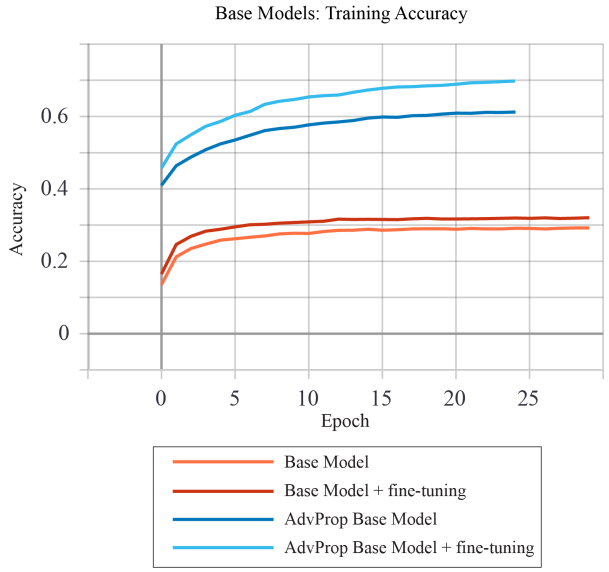

Figure 1: Training accuracy of base models & fine-tuned base models during training

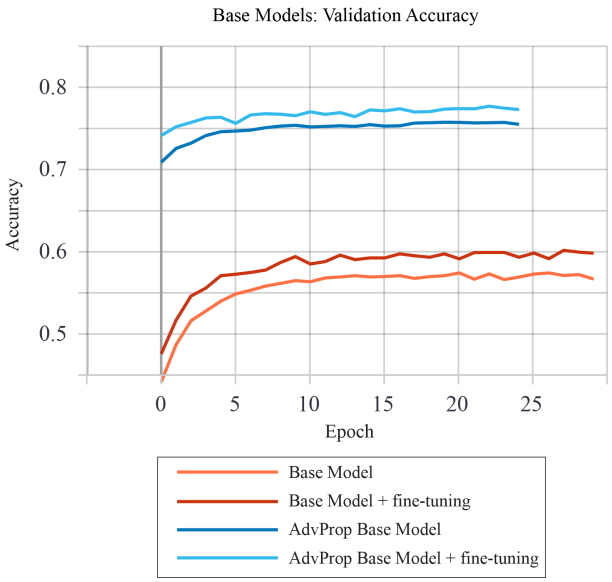

Figure 2: Validation accuracy of base models & fine-tuned base models during training

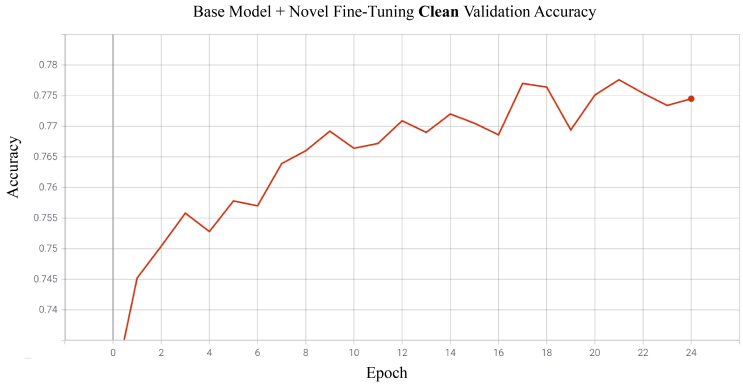

Figure 3: Validation accuracy of novel model on clean images during training.

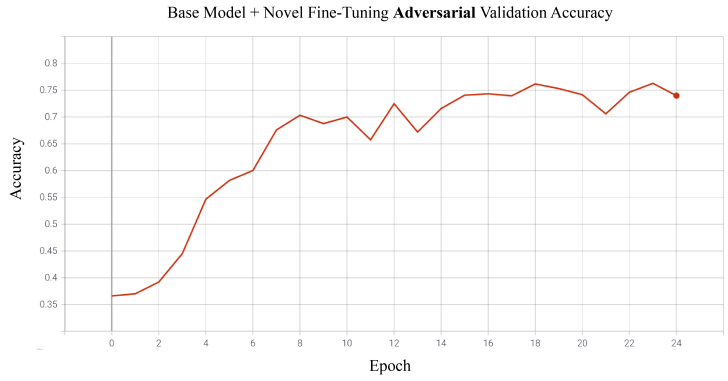

Figure 4: Validation accuracy of novel model on adversarial images during training.

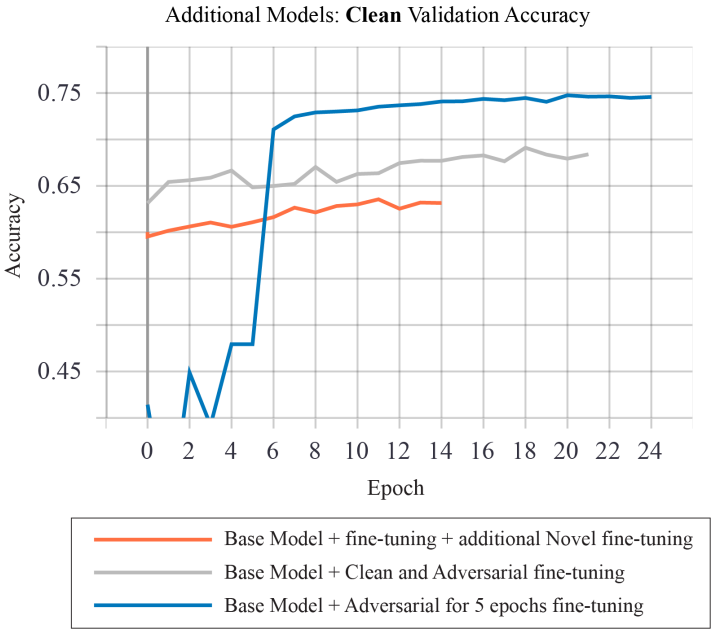

Figure 5: Validation accuracy of Base Model + various fine-tuning methods on clean images during training

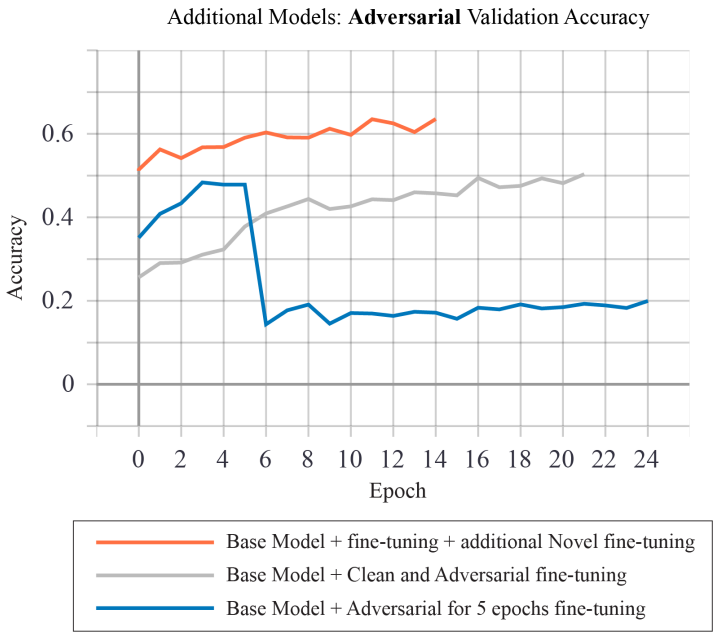

Figure 6: Validation accuracy of Base Model + various fine-tuning methods on adversarial images during training

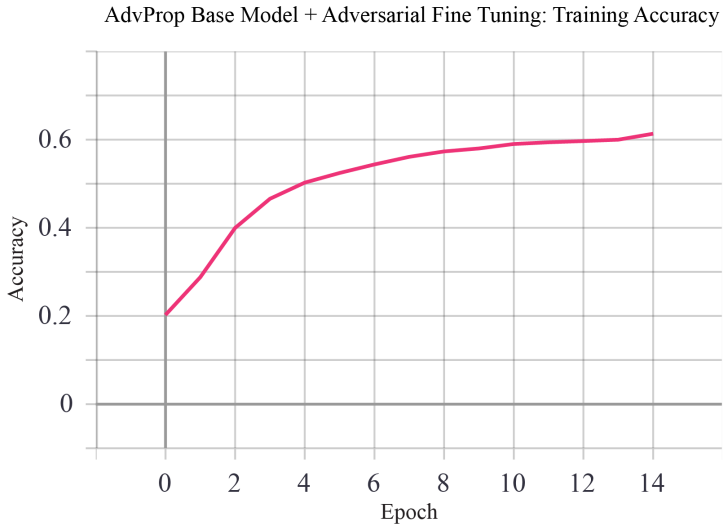

Figure 7: Training accuracy of AdvProp Base Model + only adversarial fine-tuning during training

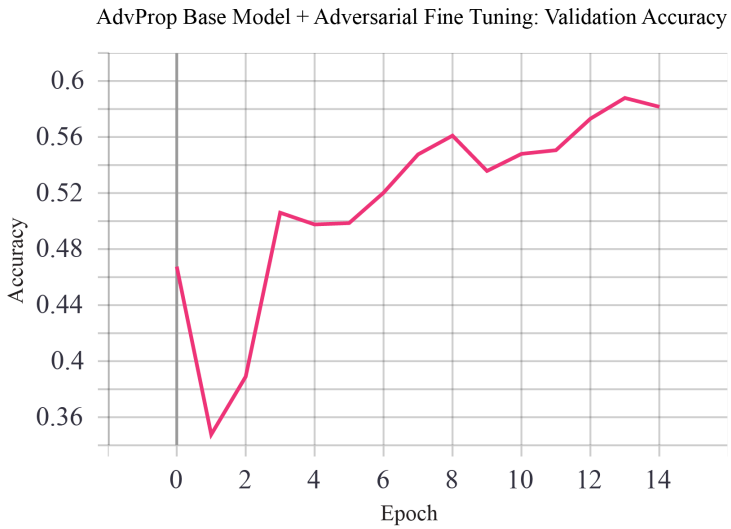

Figure 8: Validation accuracy of AdvProp Base Model + only adversarial fine-tuning during training