Adversarial Robustness of Vision Transformers and Convolutional Neural Networks

Sean Osier **Yutong Zhou**
Columbia University
smo2152@columbia.edu
yz4429@columbia.edu

Abstract

We study the impact of various hyperparameter and training choices on the adversarial robustness of both Vision Transformers and Convolutional Neural Networks. Ultimately, we identify several factors that significantly impact the models' adversarial accuracy. In particular, normalization (both layer and batch) is extremely detrimental to adversarial robustness. We also find that increasing model size / capacity increases model adversarial robustness up to a particular size, after which it generally provides no further benefit. We additionally confirm the importance of proper model fit (neither under nor overfitting) to maximizing adversarial robustness. Finally, in a simple case-study, we demonstrate how our findings can be applied to dramatically improve the adversarial robustness of a Vision Transformer model over an adversarially naive implementation.

1 Introduction

State-of-the-art artificial intelligence systems such as the recently released Google Gemini family of models [1] are increasingly multimodal with the ability to process text, images, audio, and even video. At the heart of these multimodal models is the Transformer neural network architecture [2]. This is because Transformers can natively work with any input types once they're embedded into the same embedding space.

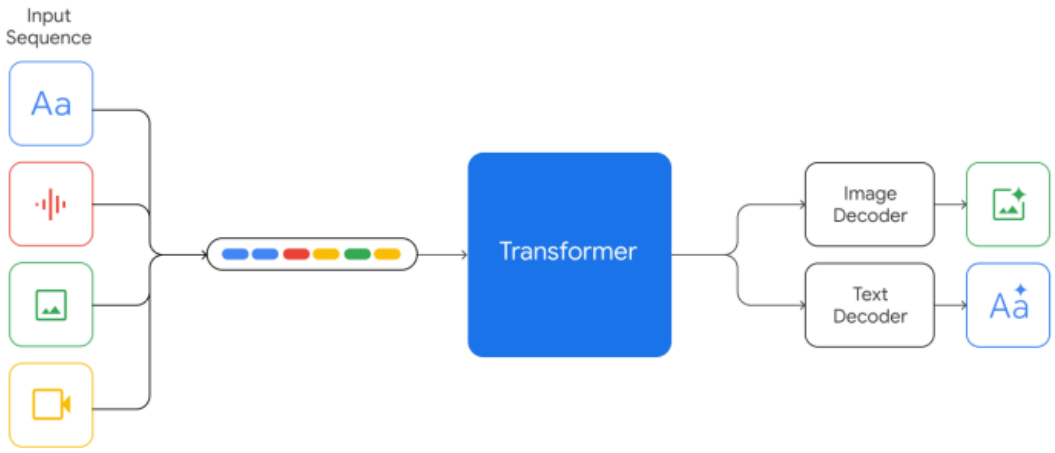

Figure 1: Simple illustration of the Gemini architecture[1]. The Transformer is its heart.

While multimodality makes the models increasingly capable and useful, it also exposes them to a new range of potential adversarial attacks. In particular, adversarial attacks are a well-known and persistent problem in the computer vision domain [3]. Unlike text which is comprised of discrete tokens, image inputs are continuous. As a result, adversarial examples can be constructed with mathematical precision and in an automated fashion, whereas text-based adversarial attacks are much more manual and constrained to some finite set of tokens. Simply put, it's just easier to construct adversarial images than adversarial text. As such, organizations deploying such models need to be prepared for such attacks.

To that end, there's recently been an increasing amount of work on adversarial robustness for Vision Transformers (ViTs)[4], [5], [6], [7], [8]. While this problem has been approached from a variety of angles, to the best of our knowledge no prior work has attempted to comprehensively study the impact of the plethora of hyperparameter and training choices on the adversarial robustness of Vision Transformers. As such, in this work, we perform a rigorous analysis of which factors impact the adversarial robustness of Vision Transformers and to what extent. We also compare the results to Convolutional Neural Networks (CNNs) to contextualize the relative strengths and weaknesses of these models when it comes to adversarial robustness and reveal more general factors impacting adversarial robustness.

2 Related Work

2.1 Adversarial Robustness in CNNs and ViTs

There are various existing studies on adversarial robustness in CNNs and ViTs in a variety of different domains. Laleh, Truhn, Veldhuizen, *et al.* [9] demonstrate that CNNs are more susceptible to white and black box adversarial attacks in the domain of computational pathology, in comparison to ViTs, which are magnitudes more robust. They propose that ViTs are inherently robust against adversarial attacks, even without adversarial pretraining or any modifications to a generic ViT architecture. They suggest that this is associated with ViTs having "a better separation of distinct classes in the latent space, and a more stable focus on relevant image regions within image tiles". Within the domain of medical imaging, Shi, Peng, Chen, *et al.* [10] propose that noise, naturally contained within many medical images, learned during the training of CNNs contribute to vulnerabilities that adversarial attacks can capitalize on.

2.2 Engineering Adversarial Robustness

With the increasing discussion around the vulnerabilities of neural networks, there have been a multitude of papers published regarding methods of designing robust models. Lecuyer, Atlidakis, Geambasu, *et al.* [11] proposed a scalable approach to certified robustness that they coined "PixelDP", based on parallels between robustness between norm-bounded adversarial examples in ML and differential privacy in cryptography. Xie, Wang, Zhang, *et al.* [12] used random resizing and random padding on their input images to demonstrate the efficacy of their randomization method, which leads to resistance to both single-step and iterative attacks.

2.3 Balancing Adversarial Robustness and Generalization

There have also been multiple recent studies on whether the goal of adversarial robustness is diametrically opposed to the goal of accuracy in models, along with arguments to the contrary from others. Notably, Tsipras, Santurkar, Engstrom, *et al.* [13] assert that there exists a tension between adversarial robustness and standard generalization, in that training adversarially robust models lead to a reduction in standard accuracy. Tsipras et al. argue that this phenomenon is a consequence of "robust classifiers learning fundamentally different feature representations than standard classifiers". On the other hand, Stutz, Hein, and Schiele [14] provided evidence that better generalization does not reduce robustness, which implies that "regular robustness and generalization are not necessarily contradicting goals".

2.4 Measuring Robustness Against Adversarial Attacks

Finally, while our work involves observations on how error rates change in response to altering hyperparameters in the base model and adversarial attacks, there are existing studies on providing alternative, quantifiable measures of robustness of neural networks against adversarial attacks. Aquino, Rahnama, Seiler, *et al.* [15] propose an "incremental dissipativity-based robustness certificate" for neural networks, in the form of a Linear Matrix Inequality (LMI). This metric quantifies non-linearity in neurons using Incremental Quadratic Constraints, which was in turn originally proposed by Fazlyab, Morari, and Pappas [16]. Aquino et al. also assert that imposing a requirement that neural networks be more sector bounded can "limit variation on the output, given some perturbation on the input".

3 Methods

This section contains a detailed discussion of the most important elements of our study. Further detail can be found in the Appendix, and the complete code can be found on GitHub.

3.1 Data

Given our study requires training and evaluating many different model variants, we use the famous MNIST data set[17] comprised of black-and-white images of handwritten digits 0-9. We utilize this specific data set because the images are very small (28x28 pixels and only one black-and-white color channel). This allows us to minimize training time and thereby maximize the number and variety of training runs we can do / experiments we can conduct.

3.2 Models

In this project, we study two distinct classes of image models: Vision Transformers and Convolutional Neural Networks (CNNs).

3.2.1 Vision Transformer

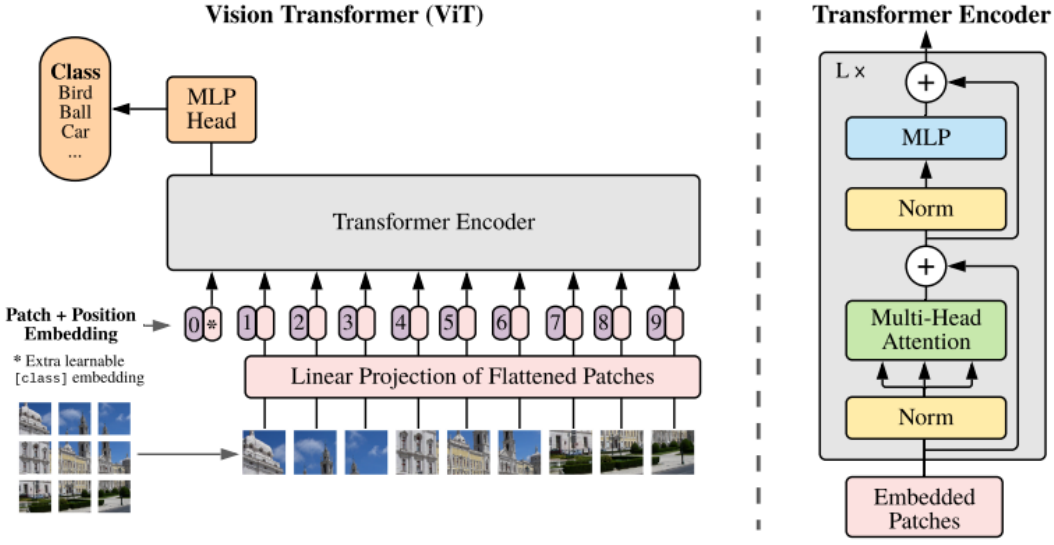

Figure 2: Vision Transformer Architecture[4]

Transformers were first proposed Vaswani, Shazeer, Parmar, *et al.* [2] as a new, simple architecture that relies entirely on attention mechanisms to "draw global dependencies between input and output". Originally designed for sequence-to-sequence problems, they were widely applied to natural language processing tasks where they quickly became state-of-the-art. Then in 2021, Dosovitskiy, Beyer, Kolesnikov, *et al.* [4] extended its applications to the vision domain, and demonstrated that Vision

Transformers attained results at least comparable to state-of-the-art CNNs, while taking fewer computational resources to train.

The heart of a Vision Transformer is the Transformer Encoder which is nearly identical to the original Transformer Encoder from Vaswani et al., differing only in the order of the Layer Normalization[18] steps within each Transformer block / layer. Vaswani et al. position the Layer Normalization after the Multi-Head Attention and MLP calculations, whereas Dosovitskiy et al. position it before these steps in a Vision Transformer.

The primary difference between a standard Transformer and a Vision Transformer comes from the need to "tokenize" and embed its image inputs. Specifically, a Vision Transformer first breaks an input image into a sequence of "patches" or sub-images. It then flattens these patches into vectors and runs these vectors through a single learned, linear layer to generate patch embeddings. Because the goal is classification, Vision Transformers also prepend a learned "class token". This is the token from which we make the final class prediction for the image so that we don't bias our result to any particular image patch. Finally, Vision Transformers typically apply a learned positional embedding as opposed to the 1-dimensional sine / cosine-based one used by Vaswani et al [2].

For our analyses, we use a baseline Vision Transformer with patches of 7x7 pixels and two Transformer Encoder layers with a model dimension of 16, 4 attention heads, each with a dimension of 8, and a MLP with dimension 32 and GELU[19] activations.

Full baseline model details can be found in the Appendix.

3.2.2 Convolutional Neural Network

Convolutional Neural Networks (CNNs) are a widely used and very successful architecture for working with image data. They were popularized by Yann Lecun and others in the late 1980s and 1990s[20]. In the early 2010s, their success on ImageNet[21] helped spark the current Deep Learning revolution.

CNNs work by using a combination of learned, sliding window "filters" or "kernels" that perform calculations on sub-arrays / sub-images of the image ("convolutions") and down-sampling that "summarizes" and condenses the information in the image into a smaller number of dimensions. The convolutions aim to automatically learn relevant, local image features and make the network more robust to image translations (being shifted up, down, left, and/or right). The down-sampling operation typically used is max-pooling which simply takes the max value over its input field. Combining these two operations in alternating order, CNNs typically reduce the height and width of the image representation while increasing the size of the channel dimension, before ultimately flattening the representation into a single vector that contains all the information parsed from the image. Finally, fully connected linear layers are often applied in later layers before the final output.

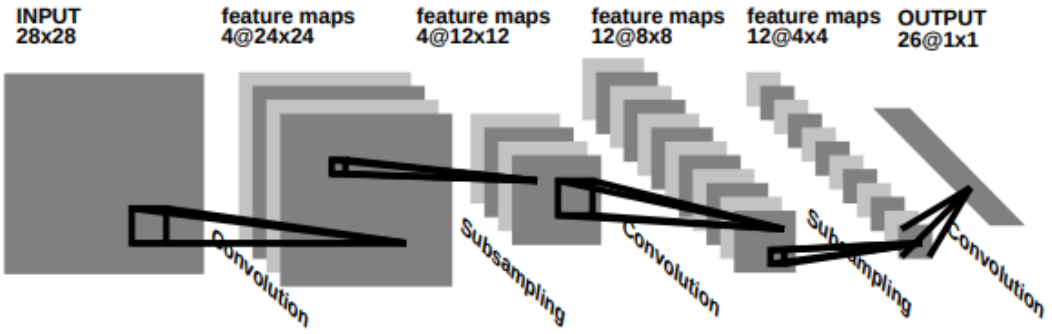

Figure 3: Example CNN Architecture[20]

For our analyses, we use a baseline CNN with 16 channels in the first convolutional layer and 32 in a second convolutional layer. Both of these layers use a 3x3 convolutional kernel size with 1 pixel of image padding and a convolutional stride of 1. They also both use max-pooling down-sampling with a 2x2 kernel size and ReLU activation function. We then apply a final fully connected output layer to generate the final logits.

Full baseline model details can be found in the Appendix.

3.3 Adversarial Attacks

3.3.1 Introduction to Adversarial Attacks

The term "adversarial" attacks was notably coined by Szegedy, Zaremba, Sutskever, *et al.* [22] when they found that a wide variety of models trained on different subsets of training data misclassify the same purposely perturbed images. They discovered that by applying an imperceptible non-random perturbation to a test image, it is possible to arbitrarily change the network's prediction. "These perturbations are found by optimizing the input to maximize the prediction error. We term the so perturbed examples 'adversarial examples'".

Goodfellow, Shlens, and Szegedy [23] further hypothesized that it is the linear nature of neural networks that causes them to be susceptible to "linear adversarial perturbation". They posit that commonly used neural network architectures: LSTMs, RELUs, and maxout networks, are intentionally designed to behave in a linear manner. We may view the activation of a single layer on an adversarially perturbed image as the following:

$$w^T \tilde{x} = w^T x + w^T \eta$$

where w represents the weight vector, x the pixels of the original image, and η the perturbations. Following that, they found that the optimal max-norm constrained perturbation to be applied can be found via:

$$\eta = \epsilon \text{ sign}(\nabla_x J(\theta, x, y))$$

where θ represent the parameters of the model, ϵ a scalar that represents the threshold of maximum perturbation allowed, x the inputs, y the targets associated with x, and $J(\theta, x, y)$ the cost function used to train the network. This method, which they refer to as the "fast gradient sign method" (FGSM) of generating adversarial examples, reliably causes a variety of models to misclassify their input.

3.3.2 `Torchattacks` PGD Attack

In our experiments, we use the projected gradient descent (PGD) algorithm, an iterative method published by Madry et al.[3] that they propose is "the strongest attack utilizing the local first-order information about the network." Specifically, we use the PGD implementation from the `Torchattacks` Python package [24].

PGD is essentially the multi-step version of the FGSM. In `Torchattacks` it applied using the following algorithm:

Algorithm 1: PGD

Data: θ, J, x, y, ϵ, α, steps
Result: adv_img

adv_img $\sim$ Uniform(hypercubic region where: $||\text{adv\_img} - x||_\infty \leq \epsilon$);
for *i = 1 to steps* **do**
    adv_img = adv_img + α sign($\nabla_{\text{adv\_img}} J(\theta, \text{adv\_img}, y)$);
    for *any dimension j where* $\text{adv\_img}_j > x_j + \epsilon$ **do**
        $\text{adv\_img}_j = x_j + \epsilon$;
    for *any dimension j where* $\text{adv\_img}_j < x_j - \epsilon$ **do**
        $\text{adv\_img}_j = x_j - \epsilon$;
return *adv_img*

where θ, J, x, y, and ϵ are as defined for FGSM, and α is the step size at each step. Note the algorithm starts at a randomly sampled point near the original image x instead of exactly at x each time to ensure that the algorithm isn't perfectly deterministic and can produce multiple different adversarial examples for any given image.

For this work, we pick ϵ, α, and the number of steps for each model class in order to generate a baseline adversarial accuracy of *very roughly* 50%. The precise baseline adversarial accuracy number

wasn't as important as having room for the metric to both improve and worsen substantially so that we could best understand the effects of our different experiments.

Full baseline adversarial attack details can be found in the Appendix.

3.3.3 PGD Attack Example

Figure 4 gives an example of a PGD attack on our baseline Vision Transformer model. One the left is the original image and model prediction. In the middle is the final perturbation, where dark green is a $+\epsilon$ change, dark red is $-\epsilon$ change, and white is 0 change. On the right is the final adversarial image and the model's prediction on it. Note, the maximum absolute change for any pixel in this example is only about 3%.

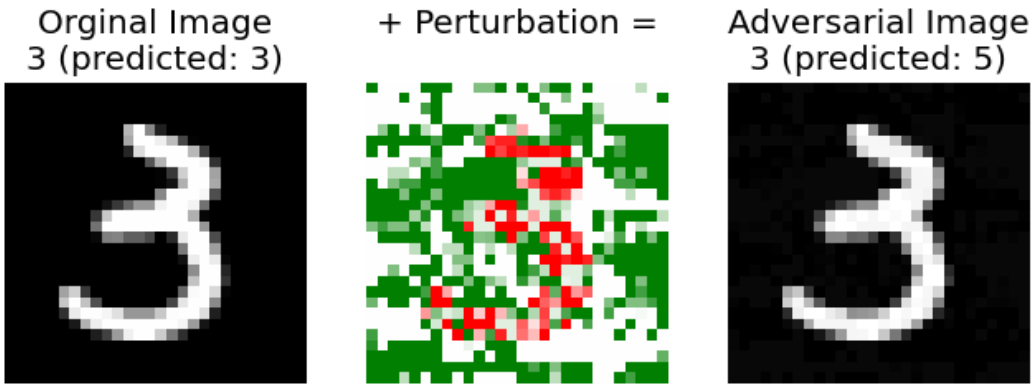

Figure 4: Example Adversarial Attack

4 Experimental Results

4.1 Vision Transformer Experiments

Figure 5 contains the results of all Vision Transformer experiments. Here we discuss each in turn.

Image Patch Size In our tests, patches of 4x4 and 7x7 pixels performed substantially worse on adversarial accuracy than patches of 2x2, 14x14, or 28x28 (i.e. the whole image) pixels. This perhaps suggests the model prefers to have either as many patches as possible (2x2) or patches as large as possible (14x14 / 28x28) because the embeddings for each of these can be richer given they take in information from many more pixels at a time. Standard test accuracy was slightly better with large patches, but generally, all patch sizes performed well on this metric.

Model Size / Capacity Our experiments showed that generally speaking increasing model dimensionality (size / capacity) helps with adversarial accuracy, in some cases substantially. Most notably increasing the **primary model dimension** (d_{model}) provides the most significant increase in adversarial accuracy, up until about $d_{model} = 40$, at which point further increase don't seem to provide additional benefit. The **attention head dimension** (d_{head}) and **number attention heads** (n_{head}) behave similarly, though with less substantial gains from increasing capacity. These level off at roughly $d_{head} = 32$ and $d_{head} = 6$. Finally, the **MLP dimension** (d_{mlp}) and **number of Transformer layers** (n_{layers}) both show modest, but steady increases in adversarial accuracy as they increase. Neither begin to level off in the ranges of values we tested. With the lone exception of an extremely small d_{model} size the standard test accuracy is very strong under all the variations of model size we tested.

Positional Embedding Our experiments confirm Dosovitskiy et al.'s choice of learned position embeddings as the best default choice for Vision Transformers as they perform slightly better on both adversarial and standard test accuracy than the standard Transformer position embeddings from

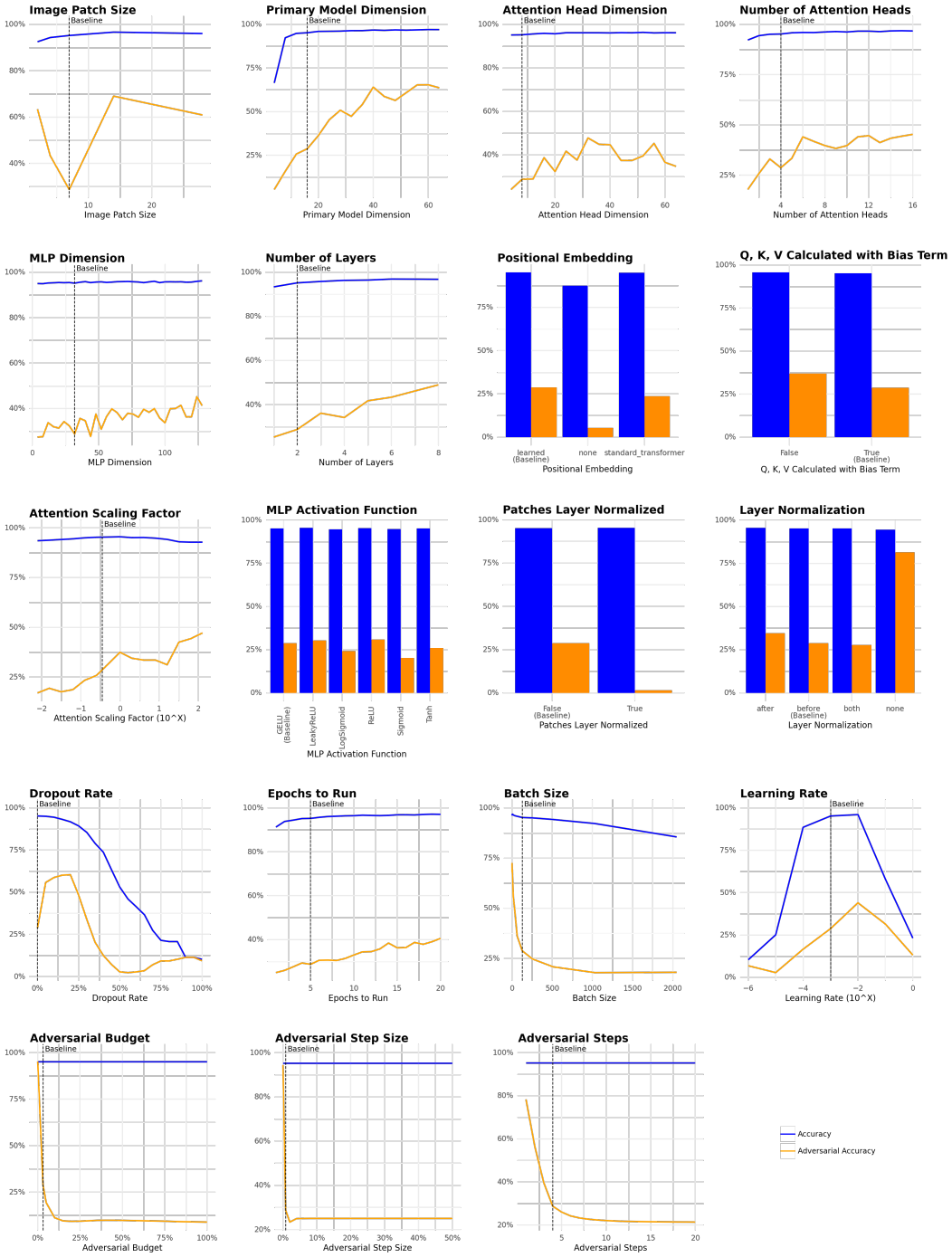

Figure 5: Experimental Results - Vision Transformer

Vaswani et al. Excluding position embeddings completely is clearly worse for both accuracies, but especially adversarial accuracy.

Q, K, V Bias Term Our experiments suggest there is a slight boost to adversarial accuracy by not learning a bias term in the linear calculations of the Q, K, V matrices in the Transformer attention operation. Standard test accuracy is very similar either way.

Attention Scaling Factor Interestingly, our experiments show that adversarial accuracy improves when the attention softmax inputs are scaled up, even up to 128x. One hypothesis for why this might be the case is that scaling these inputs better helps the model ignore spurious patch activations from for example adversarial attacks by strengthening attention on the truly most important patches. Meanwhile, traditional test accuracy performs slightly better at more normal scaling values around the Transformer default $(\sqrt{d_{head}})^{-1}$.

MLP Activation Function Our experiments show that different MLP activation functions produce largely similar results in terms of adversarial accuracy and especially traditional test accuracy. The sigmoid-based activation functions perform relatively worst and ReLU performs relatively best in terms of adversarial accuracy.

Layer Normalization Perhaps the most significant finding from all our experiments is that Layer Normalization is extremely bad for the adversarial accuracy of Vision Transformers, no matter where it's applied. **Layer Normalizing the raw patches** reduces adversarial accuracy to near zero, while removing Layer Normalization completely elsewhere in the network dramatically improves adversarial accuracy at negligible cost to traditional test accuracy. We hypothesize Layer Normalization has such a significant effect on adversarial accuracy because in rescaling its inputs to have zero mean and unit variance it can be co-opted to effectively upscale and amplify the magnitude of an adversarial attack.

Dropout Our experiments show increasing the model's dropout rate increases adversarial accuracy up to a dropout rate of about 20%. After this point, increasing the dropout rate further begins to rapidly hurt the adversarial accuracy until it ultimately bottoms out at dropout rates just above 50% with near zero adversarial accuracy. Increasing the dropout rate even further beyond that point, adversarial accuracy begins to improve again but only up to about 10% (equivalent to random guessing) at 100% dropout. Increasing the dropout rate above zero consistently worsens traditional test accuracy.

Training Parameters In our experiments, allowing our Vision Transformer to train for more **epochs** reliably improves both adversarial and traditional test accuracy. This suggests that despite the model's high traditional test accuracy at our baseline training time of 5 epochs, the model is still underfitting. Consistent with this finding, we also observe that decreasing **batch size** (effectively giving the model more gradient descent iterations during its 5 epochs of baseline training) also improves both adversarial and traditional test accuracy, and increasing batch size reliably worsens both metrics. Similarly, increasing the model's **learning rate** also improves both adversarial and traditional test accuracy up to a point, after which further learning rate increases hurt both metrics. Decreasing the learning rate from its baseline value of 0.001 worsens both metrics. This is again consistent with our underfitting hypothesis in that a slightly higher learning rate allows the model to learn more in the finite amount of training time it's given.

Adversarial Attack Parameters Finally, the least surprising results of our experiments is that increasing the model's total **adversarial "budget" (ϵ)**, **adversarial step size (α)**, and **adversarial steps** all hurt adversarial accuracy, and decreasing these parameters improves adversarial accuracy. In other words, increasing the intensity of the adversarial attack hurts adversarial accuracy while decreasing the intensity of the attack improves adversarial accuracy. None of these metrics have any impact on the traditional test accuracy.

4.2 CNN Experiments

Figure 6 contains the results of all CNN experiments. Here we discuss each in turn.

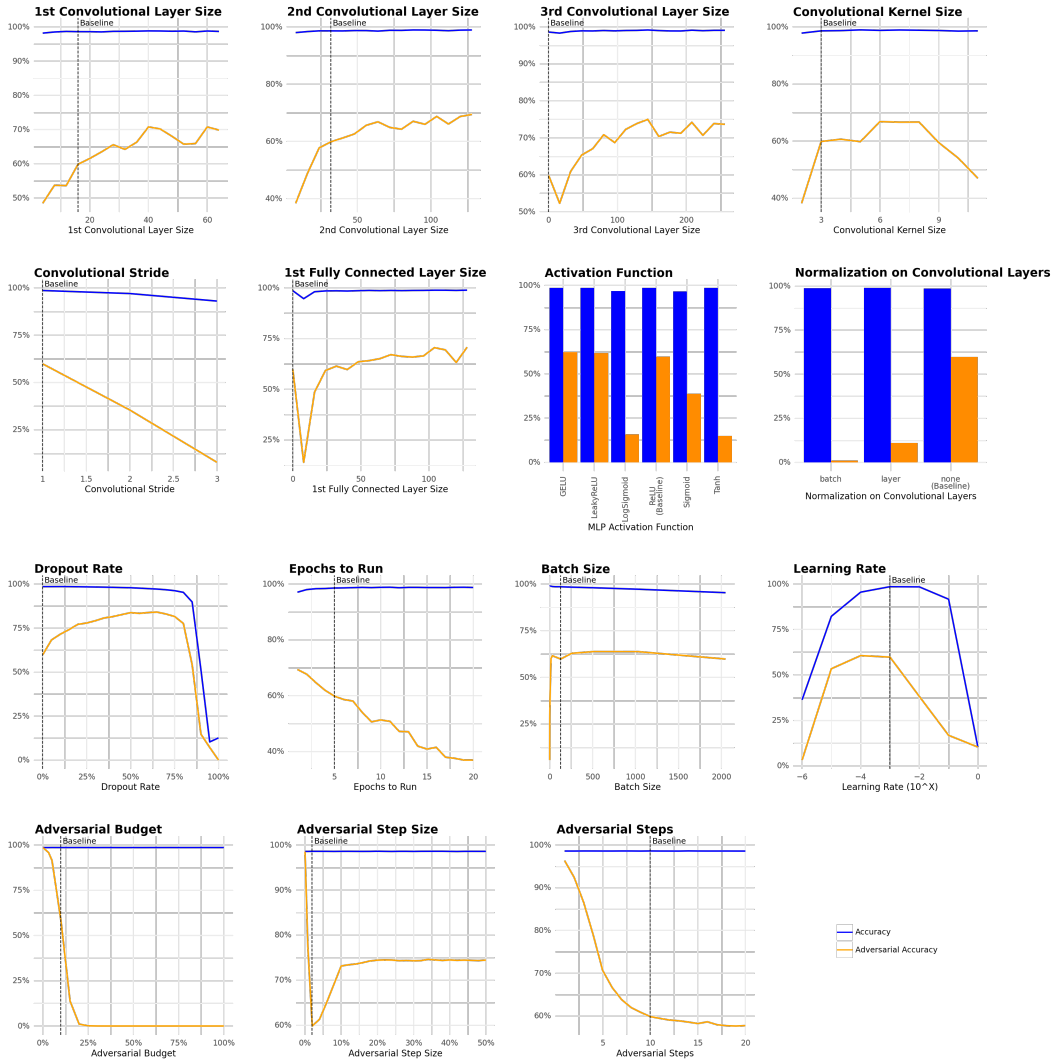

Figure 6: Experimental Results - CNN

Model Size / Capacity Our experiments show that increasing layer size generally improves adversarial accuracy up to a point, after which the model's adversarial performance levels off. This was true for all layers and layer types we tested: the **1st convolutional layer**, **2nd convolutional layer**, **3rd convolutional layer**, and the **1st fully connected layer** before the final output layer (which has to remain a fixed size). Notably, for layers that didn't initially exist in our baseline model (i.e. the 3rd convolutional layer and 1st fully connected layer), the model's adversarial performance initially drops when moving from their baseline size of zero to a very low layer size. This is because the layer becomes a bottleneck for information flowing through the network. After this however, the model's adversarial accuracy rapidly increases as the layer's size is increased, ultimately surpassing the performance when the layer did not exist. Traditional test accuracy remains strong in essentially all layer-size scenarios we tested. Even the bottleneck scenarios mentioned above barely worsen it.

Convolutional Kernel Size & Stride In our experiments, we observe the optimal **convolutional kernel size** for adversarial accuracy to be between 6x6 and 8x8 pixels. The worst kernel size tested was 2x2 pixels. None of the kernel sizes had any meaningful impact on the traditional test accuracy. Increasing the **convolutional stride** reliably worsened model performance on both adversarial and traditional test accuracy, though the impact was much more pronounced on the adversarial accuracy.

Activation Function Among the activation functions tested, none have a meaningful impact on traditional test accuracy, but the sigmoid, log-sigmoid, and tanh functions perform markedly worse on adversarial accuracy. The ReLU, leaky ReLU, and GELU activation functions all perform similarly on adversarial accuracy. This suggests there is some benefit to the activation function not enforcing a hard upper bound on activations.

Normalization As in our Vision Transformer experiments, perhaps the most significant finding from our CNN experiments is that normalization, both layer and batch normalization, is extremely bad for adversarial accuracy. Batch normalization in particular reduces our baseline model's adversarial accuracy to near zero. Again, we hypothesize this is because in rescaling its inputs to have zero mean and unit variance the normalization step is co-opted to effectively upscale and amplify the magnitude of an adversarial attack. We expect this is particularly pronounced for batch normalization because during inference (and testing) the batch statistics used are from the training distribution and an adversarial attack is very purposely designed to not match that training distribution.

Dropout Our experiments show increasing the model's dropout rate increases adversarial accuracy (with diminishing returns) up to a dropout rate of about 65%. Up to this point, traditional test accuracy worsens as dropout increases, but only slightly and very slowly. However, after this point, increasing the dropout rate further begins to rapidly worsen both the adversarial accuracy and traditional test accuracy.

Training Parameters In our experiments, we observe training our baseline CNN for more **epochs** reliably worsens adversarial accuracy while having no impact on traditional test accuracy. This suggests any additional training is only overfitting. This rough effect is also observed in the **learning rate** where adversarial accuracy is slightly improved by slightly decreasing the learning rate while traditional test accuracy is slightly worsened. And contrarily, increasing the learning rate slightly above its baseline value slightly improves the traditional test accuracy at the cost of worsening the adversarial accuracy. In other words, to optimize adversarial accuracy in isolation we would like to slow the learning process by learning less at each optimization step whereas to optimize traditional accuracy in isolation we want to learn faster to overfit more quickly. Finally, **batch size** has little impact on either accuracy metric with the lone exception of extremely low batch sizes, at which the adversarial accuracy plummets. This is very likely because such low batch sizes dramatically increase the number of training iterations for any fixed number of epochs trained, leading our CNN to overfit.

Adversarial Attack Parameters Similar to the Vision Transformer, the least surprising of our CNN results is that increasing the model's total **adversarial "budget" (ϵ)** and **adversarial steps** hurts adversarial accuracy, and decreasing these parameters improves adversarial accuracy. In other words, increasing the intensity of the adversarial attack hurts adversarial accuracy while decreasing the intensity of the attack improves adversarial accuracy. Interestingly, increasing the **adversarial step size (α)** actually helps the adversarial accuracy up until roughly our baseline adversarial budget value ($\epsilon = 25/255$) after which adversarial accuracy levels off. We hypothesize this happens because at large adversarial step sizes the resulting adversarial image is effectively forced to be on the boundary of the permissible region in image space where we constrain the adversarial image to live. This is because when the step size is large enough (e.g. $\geq 2\epsilon$) each step will always take it to some point on the boundary. As such, if the optimal attack is somewhere in the middle of the permissible region, the step size lacks the precision to find that optimal attack. Note, none of these metrics have any impact on the traditional test accuracy.

4.3 Differences Between Vision Transformers and CNNs

While many modeling and training choices have similar effects on both Vision Transformers and CNNs (increasing model size / capacity, including normalization, increasing adversarial attack intensity), there are also some distinct differences between the two model architectures.

Most notably, after our baseline training period of 5 epochs (a training length where both models achieved high traditional test accuracy), the adversarial accuracy of our Vision Transformer reliably improved with more training, be that more epochs or simply a lower batch-size to force more training steps per epoch. By contrast, more epochs actively hurt our CNN's adversarial accuracy, and changing the batch-size had little impact. Despite both outwardly appearing to be well-fit, with high traditional

test accuracy, this suggests that our baseline Vision Transformer is underfit and our CNN is overfit. Related to this, we also observe that our CNN prefers a slightly lower learning rate while our Vision Transformer prefers a slightly higher learning rate. The lower learning rate helps the CNN overfit less, while the higher learning rate helps the Vision Transformer make the most of each training step it receives.

Another key difference we observed was that in choosing our baseline model parameters we needed to use a stronger adversarial attack against the CNN to get the starting (baseline) adversarial accuracy to be in an acceptable range (where there was plenty room for it to either increase or decrease in our experiments) than we did for the Vision Transformer. This may be related to the above overfitting vs. underfitting dichotomy, where the Vision Transformer isn't properly fit yet, and so is unable to withstand as strong adversarial attacks. Alternatively, it could simply be that our baseline CNN is more adversarially robust than our baseline Vision Transformer.

Another interesting difference is that the CNN tolerates a much higher dropout rate before performance begins to suffer than the Vision Transformer. Given dropout is a form of regularization this may also be related to the overfitting vs. underfitting dichotomy discussed above, where the CNN benefits from more stringent regularization because it's more overfit.

Finally, a more minor difference between the two architectures is that the CNN is more sensitive to the choice of activation function, while the Vision Transformer performs largely similarly across all activation functions tested.

4.4 Applying Our Findings

As a final test of the validity of our experimental results, we apply our findings to improve our baseline Vision Transformer model. To do so, we greedily apply the optimal setting we found for each non-adversarial attack parameter in our experiments and retrain / reevaluate the model. We keep the change if it improves adversarial accuracy for the new model and revert to the baseline value for that parameter otherwise. Ultimately, after following this simple model improvement process, we were able to increase the adversarial accuracy of our Vision Transformer from approximately **38%** using our naive baseline implementation to approximately **96%** using our improved hyperparameters and training procedure.

5 Conclusion

In this study, we performed a rigorous analysis of which factors impact the adversarial robustness of Vision Transformers and CNNs. Ultimately we identified several factors that significantly impact the adversarial accuracy of the vision models we studied. Most critically, we found strong evidence to suggest that normalization (both layer and batch) are extremely detrimental to adversarial robustness. We also found that increasing model size / capacity also increases model adversarial robustness up to a particular size, after which it generally provides no further benefit. We additionally observed the importance of proper model fit (neither under nor overfitting) to maximizing adversarial robustness. Finally, in a simple case-study, we demonstrated how our findings could be applied to quickly and dramatically improve the adversarial robustness of our baseline Vision Transformer model.

Much future work in the area of adversarial robustness remains though. First, it would be highly beneficial to validate and replicate these results in larger models on more complex image data sets. Secondly, there's also value in understanding to what extent these results are consistent across other types of adversarial attacks in addition to PGD. Finally, studying adversarial robustness in true multimodal Transformers could make this work even more practically useful for state-of-the-art deployed systems where adversarial attacks are of greatest concern.

References

[1] Gemini Team, Google, "Gemini: A family of highly capable multimodal models," Google DeepMind, Tech. Rep., Dec. 2023. [Online]. Available: `https://storage.googleapis.com/deepmind-media/gemini/gemini_1_report.pdf`.

[2] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, *Attention is all you need*, 2023. arXiv: `1706.03762 [cs.CL]`.

[3] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, *Towards deep learning models resistant to adversarial attacks*, 2019. arXiv: `1706.06083 [stat.ML]`.

[4] A. Dosovitskiy, L. Beyer, A. Kolesnikov, *et al.*, *An image is worth 16x16 words: Transformers for image recognition at scale*, 2021. arXiv: 2010.11929 [cs.CV].

[5] R. Shao, Z. Shi, J. Yi, P.-Y. Chen, and C.-J. Hsieh, *On the adversarial robustness of vision transformers*, 2022. arXiv: 2103.15670 [cs.CV].

[6] Y. Mo, D. Wu, Y. Wang, Y. Guo, and Y. Wang, "When adversarial training meets vision transformers: Recipes from training to architecture," in *NeurIPS*, 2022.

[7] G. Kim and J.-S. Lee, *Analyzing adversarial robustness of vision transformers against spatial and spectral attacks*, 2022. arXiv: 2208.09602 [cs.CV].

[8] K. Mahmood, R. Mahmood, and M. Van Dijk, "On the robustness of vision transformers to adversarial examples," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 7838–7847.

[9] N. Laleh, D. Truhn, G. Veldhuizen, *et al.*, "Adversarial attacks and adversarial robustness in computational pathology," *Nature Communications*, vol. 13, Sep. 2022. DOI: 10.1038/s41467-022-33266-0.

[10] X. Shi, Y. Peng, Q. Chen, *et al.*, "Robust convolutional neural networks against adversarial attacks on medical images," *Pattern Recognition*, vol. 132, p. 108 923, 2022, ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2022.108923.

[11] M. Lecuyer, V. Atlidakis, R. Geambasu, D. Hsu, and S. Jana, *Certified robustness to adversarial examples with differential privacy*, 2019. arXiv: 1802.03471 [stat.ML].

[12] C. Xie, J. Wang, Z. Zhang, Z. Ren, and A. Yuille, "Mitigating adversarial effects through randomization," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=Sk9yuql0Z.

[13] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. Madry, *Robustness may be at odds with accuracy*, 2019. arXiv: 1805.12152 [stat.ML].

[14] D. Stutz, M. Hein, and B. Schiele, "Disentangling adversarial robustness and generalization," *CoRR*, vol. abs/1812.00740, 2018. arXiv: 1812.00740. [Online]. Available: http://arxiv.org/abs/1812.00740.

[15] B. Aquino, A. Rahnama, P. Seiler, L. Lin, and V. Gupta, "Robustness against adversarial attacks in neural networks using incremental dissipativity," *CoRR*, vol. abs/2111.12906, 2021. arXiv: 2111.12906. [Online]. Available: https://arxiv.org/abs/2111.12906.

[16] M. Fazlyab, M. Morari, and G. J. Pappas, *Safety verification and robustness analysis of neural networks via quadratic constraints and semidefinite programming*, 2021. arXiv: 1903.01287 [math.OC].

[17] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[18] J. L. Ba, J. R. Kiros, and G. E. Hinton, *Layer normalization*, 2016. arXiv: 1607.06450 [stat.ML].

[19] D. Hendrycks and K. Gimpel, *Gaussian error linear units (gelus)*, 2023. arXiv: 1606.08415 [cs.LG].

[20] Y. Lecun and Y. Bengio, "Convolutional networks for images, speech and time series," in *The Handbook of Brain Theory and Neural Networks*, M. A. Arbib, Ed. The MIT Press, 1995, pp. 255–258.

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[22] C. Szegedy, W. Zaremba, I. Sutskever, *et al.*, *Intriguing properties of neural networks*, 2014. arXiv: 1312.6199 [cs.CV].

[23] I. J. Goodfellow, J. Shlens, and C. Szegedy, *Explaining and harnessing adversarial examples*, 2015. arXiv: 1412.6572 [stat.ML].

[24] H. Kim, *Torchattacks: A pytorch repository for adversarial attacks*, 2021. arXiv: 2010.01950 [cs.LG].

Appendix

All Baseline Model Parameters

Table 1: Baseline Model Parameters

(a) **Vision Transformer Parameters**

Model Parameters	
Patch Size	7x7
Model Dimension (d_{model})	16
Attention Head Dim. (d_{head})	8
# Attention Heads (n_{head})	4
MLP Dimension (d_{mlp})	32
# Transformer Layers (n_{layers})	2
Position Embeddings	Learned
QKV Bias?	True
Scaled Attn. Scaling Factor	$(\sqrt{d_{head}})^{-1}$
MLP Activation Function	GELU
Apply Layer Norm. to Patches?	False
Layer Normalization	Before
Dropout	0%
Training Parameters	
Optimizer	Adam
Epochs Trained	5
Batch Size	128
Learning Rate	0.001
Adv. Attack Parameters	
Adversarial Algorithm	PGD
ϵ	8/255
α	2/255
steps	4

(b) **CNN Parameters**

Model Parameters	
Convolutional Layer 1 Size	16
Convolutional Layer 2 Size	32
Convolutional Layer 3 Size	0
Convolutional Kernel Size	3x3
Convolutional Stride	1
Convolutional Padding	1
Max Pooling Kernel Size	2x2
Activation Function	ReLU
Fully Connected Layer 1 Size	0
Convolutional Layer Normalization	None
Dropout	0%
Training Parameters	
Optimizer	Adam
Epochs Trained	5
Batch Size	128
Learning Rate	0.001
Adv. Attack Parameters	
Adversarial Algorithm	PGD
ϵ	25/255
α	5/255
steps	10