On the Robustness of Vision Transformers to Adversarial Examples Supplemental Material

A. Supplementary Material Organization

In this section, we briefly describe the organization of our supplementary material so that readers can find pertinent material with ease. Overall our supplementary material provides more white-box experiments, more black-box experiments, results for adversarial training of Vision Transformers and further investigation of the transferability phenomena observed in the main paper.

White-Box and Black-Box Attacks: We start with full descriptions of the white-box attack used in this paper and their corresponding parameters in Section B. In this section, we also provide the CIFAR-100 white-box attack results not given in the main paper (due to brevity as well as redundancy). Aside from the conventional white-box attacks, we also give more details for the Self-Attention Gradient Attack (SAGA) in Section C. We follow up our white-box type of attack sections with a full description of the black-box adversarial model and attack parameters in Section D. In this section, we also include black-box attack experimental results on individual models (Vision Transformers, Big Transfer Models and ResNets) for RayS. We further provide a series of hyperparameter experiments used to fine-tune the transfer based black-box attack (also known as the adaptive black-box attack). The results of these hyperparameter experiments reveal some very interesting implications for the design of future black-box attacks.

Decision Region Graphs and Transferability: In Section E, we delve further into the transferability phenomena. We start by discussing recent work that mathematically demonstrates the equivalence between Transformers and CNNs. We then empirically show that the conditions under which this equivalence happens do not likely occur for Vision Transformers and other CNNs. We demonstrate this empirically by graphing the decision regions for different Vision Transformers and CNNs for CIFAR-10, CIFAR-100 and ImageNet.

Adversarial Trained Vision Transformers: One adversarial machine learning topic omitted for space is adversarial training. Can Vision Transformers be adversarially trained as well as CNNs? In Section F, we answer this question by experimenting with different adversarial training techniques on Vision Transformers and CNNs for CIFAR-10 and CIFAR-100. Lastly, in Section G we provide additional numerical tables that may assist anyone wishing to replicate our results.

B. White-Box Attacks

In this section, we mathematically define the white-box adversarial model. We provide detailed descriptions of the white-box attacks tested in this paper alongside the parameters chosen for each white-box attack.

B.1. White-Box Adversarial Model

Mathematical Description: Formally, we mathematically describe our adversarial model follows: We start with a classifier C with (trained) parameters θ. Given input x, the classifier outputs label y such that $C(x, \theta) = y$. The goal of the adversary is to create an adversarial sample x_{adv} from x such that:

$$C(x_{adv}, \theta) \neq y \tag{7}$$

where x_{adv} is created from x using attack A. That is, $x_{adv} \triangleq A(x)$ and x_{adv} is subject to the following constraint:

$$\|x - x_{adv}\|_p \leq \epsilon \tag{8}$$

where p is the type of norm used to measure the distance between x and x_{adv} and ϵ is the maximum allowed distance between x and x_{adv}. Finally, there is one additional constraint. The image must be within a valid pixel range:

$$x \in [p_{min}, p_{max}]^{n \times m \times r} \tag{9}$$

where in Equation 9, p_{min} and p_{max} refer to the minimum and maximum pixel values of a valid image, n and m refer to the size of the image, and r represents the number of color channels in the image.

Adversarial Capabilities: In the white-box adversarial model, the adversary has knowledge of C, θ, x and y. Here C represents the type of classifier (e.g. CNN) and classifier architecture (e.g. ResNet-56). The adversary also knows the trained parameters of the classifier θ. For a CNN or Transformer, this would be the weights and biases of the classifier model. Lastly, the adversary has a clean example x and corresponding class label y. In this paper, we focus on the untargeted attack model. That is,the adversary succeeds if and only if Equation 7, Equation 8 and Equation 9 all hold true.

B.2. Types of White-Box Attacks

In a white-box attack, the adversary crafts x_{adv} from x using technique A. The choice of A can heavily affect the success rate of the attack (the percent of samples that are misclassified by C). From the literature, there are various techniques to craft white-box attacks. Below we describe each of the attacks tested in this paper:

1. **Fast Gradient Sign Method** - The Fast Gradient Sign Method (FGSM) [13] creates adversarial examples

through the addition of non-random noise in the direction of the gradients of the loss function:

$$x_{adv} = x + \epsilon * sign(\nabla_x L(x, y; \theta)) \quad (10)$$

where L is the loss function of the classifier C. Note that in Equation 10, only the sign of the loss function is used and the magnitude of the second term is dictated by ϵ, which is a small perturbation added to the image x. It is also important to note that this is a single-step attack. The adversary backpropagates on the model only once to obtain the gradient of the loss function and then applies this directly to x.

2. **Projected Gradient Descent** - Projected Gradient Descent (PGD) [24] is a multi-step variant of the FGSM algorithm. It attempts to find the minimum bounded perturbation that maximizes the loss of a model through initializing a random perturbation in a ball of radius *d* with center *x*. A gradient step is taken in the direction of the greatest loss and the perturbation is then projected back into this ball. The k-step PGD algorithm initializes $x^0 = x$ and the perturbed image x^i in the i^{th} step is computed as:

$$x^i = P(x^{i-1} + \alpha * sign(\nabla_x L(x^{i-1}; y; \theta))) \quad (11)$$

where P is the projection function that projects the adversarial data back into the ϵ-ball centered at x^{i-1} if necessary, and α is the step size. The bounds on the projection are defined by the l_p norm.

3. **Backward Pass Differentiable Approximation** - Backward Pass Differentiable Approximation (BPDA) [2] is an attack designed to overcome non-differentiable functions that would ordinarily prevent the use of backpropagation to generate adversarial examples. BPDA is capable of creating effective adversarial examples for those cases in which the defense employs gradient masking or another technique in which the gradient is obfuscated. The gradient can be obfuscated in one of three ways: shattered gradients, stochastic gradients, and exploding/vanishing gradients. Shattered gradients in a defense either introduce numerical instability or cause a gradient to be nonexistent or incorrect [2]. Stochastic gradients are generally a result of randomized defenses. Exploding and vanishing gradients generally occur in recurrent neural networks.

   For a neural network $f(\cdot) = f^{1 \cdots j}$ with a non-differentiable layer $f^i(\cdot)$, the first step of BPDA is to find a differentiable function *g(x)* that approximates f^i. The gradient of the network *f*, $\nabla_x f(x)$, is then approximated by performing a forward pass through $f(\cdot)$, and then only on the backward pass replacing $f^i(x)$ by $g(x)$. Adversarial examples are generated using a similar approach to PGD [24].

4. **Momentum Iterative Method** - A subset of gradient descent approaches, the Momentum Iterative Method (MIM) [11] applies a velocity vector in the direction of the gradient of the loss function across iterations. Because MIM takes into account previous gradients, it is better able to overcome narrow valleys, small bumps, and local minima and maxima. Specifically, the momentum algorithm gathers the gradients of t iterations with a decay factor μ. The adversarial example x_t^* is perturbed in the direction of the accumulated gradient with a step size of α. Note that if $\mu = 0$, the MIM algorithm degenerates to iterative FGSM.

   The accumulated gradient is derived as follows for untargeted attacks: Let x_t^* be the current adversarial example at iteration t with original class label y and $x_0^* \triangleq x$. The accumulated gradient is:

$$g_{t+1} = \mu * g_t + \frac{J(x_t^*, y)}{||\nabla_x J(x_t^*, y)||_1} \quad (12)$$

   where $J(x^*, y^*)$ is the loss function. For a L_∞ bounded attack, the adversarial example at iteration t is:

$$x_{t+1}^* = x_t^* + \alpha * sign(g_{t+1}) \quad (13)$$

5. **Carlini and Wagner Attack** - The aim of the Carlini and Wagner ($C\&W$) attack [7] is to perturb an image by a minimal amount such that the image will be misclassified. The following objective function is used to find the adversarial noise:

$$\min ||\frac{1}{2}(\tanh(\omega) + 1) - x||_2^2 + c \cdot f(\frac{1}{2}(\tanh(\omega) + 1)) \quad (14)$$

$$f(x') = \max(\max\{Z(x')_i : i \neq t\} - Z(x')_t, -\kappa) \quad (15)$$

   where ω is the perturbation, t is the chosen target class, κ is a constant that controls the confidence with which the sample is misclassified, $Z(x')$ is the output from the logits layer, and *c* is a constant chosen through binary search. $C\&W$ is an iterative attack because the objective of the $C\&W$ attack is formulated as an optimization problem, as given by Equation 14.

6. **Auto Projected Gradient Descent** - Auto Projected Gradient Descent (APGD) [10] is an automated version of PGD in which the step size is not fixed, but instead changes adaptatively. In APGD, the total iterations are divided into an exploration phase and an exploitation phase. A larger step size is used in the former phase, allowing for quicker exploration, while a

smaller step size is used in the latter phase to fine-tune the maximization of the loss function. The choice of step size in APGD is determined by a budget of N_{iter} iterations and the cumulative progress of optimization, as defined by two conditions in Equations 16 and 17:

$$\sum_{i=w_{j-1}}^{w_j-1} \mathbf{1}_{f(x^{(i+1)})>f(x^{(i)})} < \rho * (w_j - w_{j-1}), \quad (16)$$

$$\eta^{(w_{j-1})} \equiv \eta^{(w_j)} \wedge f_{max}^{(w_{j-1})} \equiv f_{max}^{(w_j)} \quad (17)$$

where w_j are the checkpoints at which the algorithm can reduce the step size by a factor of 2 and f_{max}^k is the highest objective value reached in the first k iterations. If one of the above two conditions is met, then the step size at iteration $k = w_j$ is halved and $\eta^{(k)} := \eta^{(w_j)}/2$ for every $k = w_j + 1, ..., w_{j+1}$. A version of the Auto-PGD which uses cross-entropy is referred to as APGD-CE. This attack was shown to be the best performing attack among the different APGD variations [10]. Thus, APGD-CE is used in our white-box attacks.

C. Self-Attention Gradient Attack (SAGA)

In the main paper we introduced the Self-Attention Gradient Attack (SAGA). Here we provide additional experimental results and parameters related to our attack.

SAGA Adversarial Images: In Figure 4 and Figure 5, we show examples of the adversarial images generated by SAGA for CIFAR-10 and ImageNet. We generate these images from a defense comprised of ViT-L-16 and BiT-M-R101x3 for CIFAR-10 and ViT-L-16 and BiT-M-R152x4 for ImageNet. In the attack, we use the l_∞ norm and $\epsilon = 0.031$ for CIFAR-10 and $\epsilon = 0.062$ for ImageNet. All attacks are untargeted. From these figures, it is clear that SAGA is capable of creating adversarial examples with minimal visual perturbations on par with standard white-box attacks.

SAGA Hyperparameters: In the main paper, we mentioned that the scaling factors α_k must be chosen carefully for each model when running SAGA. In Table 6, we give the α values and corresponding robust accuracies for each attack. From the table, it can clearly be seen that simple averaging ($\alpha_1 = 0.5, \alpha_2 = 0.5$) does not yield a high attack success rate, i.e., low robust accuracy. To illustrate, for the Bit/ViT defense for CIFAR-10, the robust accuracy with simple averaging yields an average robust accuracy of 47.5%. However, when each α is fine-tuned properly, it yields a robust accuracy of only 26% (an attack success rate of 74%).

It is worth noting that while the α values vary greatly in magnitude, each holds significance. For instance, $\alpha_1 = 0.998$ for ViT-L-16 but only $2e-4$ for BiT-M-R101x3 for SAGA for CIFAR-10. The natural question arises of whether the gradient for BiT-M-R101x3 could simply be 0. The concise answer to this is that even minute α values are critical to crafting adversarial examples that are misclassified by *both* models. For empirical proof that a single gradient does not suffice, one needs only to look at the transferability results in Table 2 in the main paper. Table 2 clearly shows that using only a single model gradient in a white-box attack does not yield highly transferable adversarial examples.

ResNet SAGA Results: In the main paper, we tested Vision Transformer and Big Transfer Model combinations. However, SAGA works on other Vision Transformer and CNN combinations as well. In Table 6, we demonstrate a proof of concept of this by attacking a ViT-L-16/ResNet-164 pair for CIFAR-10. Similar to the ViT/BiT combination, it can be seen that the ViT/ResNet combination is not secure against SAGA, as the robust accuracy is only 15%.

D. Black-Box Attacks

In this section, we mathematically define the black-box adversarial model. We provide detailed descriptions of the black-box attacks tested in this paper alongside the parameters chosen for each attack. Unlike section B in which a single adversarial model suffices, here the black-box adversarial model is divided into two distinct types: query based and transfer based models.

As a precursor, it is important to note the basic commonality between the two threat models. The definition of a successful attack is unchanged for all threat models. That is, the three conditions we previously defined must hold. First, the adversarial sample must be misclassified (Equation 7). Second, the adversarial sample x_{adv} must be within a certain distance of the original sample x (Equation 8). Third, the adversarial sample must have pixels within a valid range (Equation 9).

The other commonality between the two black-box adversarial models are the components that make up the defense: a classifier C with trained parameters θ. In contrast to the white-box adversary, we will also explicitly define additional training components. We define the training samples that C was trained on to obtain θ as the set (X, Y). Let us further define the pre-training dataset as $(X^{'}, Y^{'})$. Here the pre-training dataset is only applicable to Vision Transformers and Big Transfer Models, where the pre-training dataset $(X^{'}, Y^{'})$ is ImageNet-21K and the training dataset (X, Y) is either CIFAR-10, CIFAR-100 or ImageNet.

D.1. Query Based Adversarial Model

Adversarial Capabilities: For the query based adversarial model, the attacker lacks knowledge of θ, the specific classifier architecture C, the training set (X, Y) and

Table 3. White-box attacks on Vision Transformers, Big Transfer Models and ResNets. The attacks are done using the l_∞ norm with $\epsilon = 0.031$ for CIFAR-10 and $\epsilon = 0.062$ for ImageNet. In this table the robust accuracy is given for each corresponding attack. The last column "Acc" refers to the clean accuracy of the model. In the main paper part of this table was also presented (see Table 3.1) but without CIFAR-100 results for brevity. The table here represents the full white-box attack results.

CIFAR-10

	FGSM	PGD	BPDA	MIM	C&W	APGD	Acc
ViT-B-32	37.9%	1.8%	17.6%	4.4%	0.0%	0.0%	98.6%
ViT-B-16	39.5%	0.0%	20.3%	0.3%	0.0%	0.0%	98.9%
ViT-L-16	56.3%	1.2%	28.7%	5.9%	0.0%	0.0%	99.1%
ViT-R50	40.8%	0.1%	13.4%	0.2%	0.0%	0.0%	98.6%
BiT-M-R50x1	66.0%	0.0%	14.9%	0.0%	0.0%	0.0%	97.5%
BiT-M-R101x3	85.2%	0.0%	17.1%	0.0%	0.0%	0.0%	98.7%
ResNet-56	23.0%	0.0%	5.0%	0.0%	0.0%	0.0%	92.8%
ResNet-164	29.0%	0.0%	5.4%	0.0%	0.0%	0.0%	93.8%

CIFAR-100

	FGSM	PGD	BPDA	MIM	C&W	APGD	Acc
ViT-B-32	20.8%	1.9%	13.4%	3.1%	0.0%	0.0%	91.7%
ViT-B-16	20.4%	0.0%	11.9%	0.5%	0.0%	0.0%	92.8%
ViT-L-16	33.0%	1.6%	15.1%	4.7%	0.0%	0.0%	94.0%
ViT-R50	22.0%	0.2%	9.7%	0.4%	0.0%	0.0%	91.8%
BiT-M-R50x1	36.0%	0.0%	7.0%	0.0%	0.0%	0.0%	87.4%
BiT-M-R101x3	1.2%	0.0%	0.4%	0.0%	0.0%	0.0%	91.8%
ResNet-56	6.0%	0.2%	3.3%	0.4%	0.0%	0.0%	71.6%
ResNet-164	7.6%	0.3%	3.7%	0.9%	0.0%	0.0%	74.2%

ImageNet

	FGSM	PGD	BPDA	MIM	C&W	APGD	Acc
ViT-B-16	23.1%	0.0%	7.3%	0.0%	0.0%	0.0%	80.3%
ViT-L-16 (224)	27.9%	0.0%	8.4%	0.0%	0.0%	0.0%	82.0%
ViT-L-16 (512)	29.8%	0.0%	8.4%	0.0%	0.0%	0.0%	85.4%
BiT-M-R50x1	28.7%	0.0%	3.5%	0.0%	0.0%	0.0%	79.9%
BiT-M-R152x4	60.9%	0.0%	15.2%	0.0%	0.0%	0.0%	85.3%
ResNet-50	11.8%	0.0%	1.4%	0.0%	0.0%	0.0%	74.5%
ResNet-152	18.1%	0.0%	2.7%	0.0%	0.0%	0.0%	77.0%

Table 4. White-box attack parameters for CIFAR-10.

Attack	Parameters
FGSM	ϵ = 0.031
PGD	ϵ = 0.031, ϵ_{step} = 0.00155, steps = 20
BPDA	ϵ = 0.031, steps = 100, max iterations = 100, learning rate = 0.5
MIM	ϵ = 0.031, ϵ_{step} = 0.00155, decay factor = 1.0
CW	confidence = 50, step size = 0.00155, steps = 30
APGD	ϵ = 0.031, number of restarts = 1, ρ = 0.75, n^2 queries = 5000

the pre-training set $(X^{'}, Y^{'})$. The adversary starts with a clean example x and is able to query the classifier C, with different perturbations (e.g. $x + \epsilon$).The adversarial model here is constrained by the fact that for each example x, only a fixed number of queries q can be made on C. In this threat model, the type of response from the classifier C also matters. When the adversary queries C, the defense can return either the hard label (class label only) or the corresponding probability vector. In this paper, we consider only the adversary which has access to the hard label.

Attack Setup and Discussion: To test query based adversaries, we use the RayS attack [8]. We use 1000 clean examples for CIFAR-10 and ImageNet. In our attacks, we set the query budget q to be $10,000$ for each sample. We use $\epsilon = 0.031$ for CIFAR-10 and $\epsilon = 0.062$ for ImageNet in conjunction with the l_∞ norm. Due to the high computational complexity of the attack, we test only single models for CIFAR-10 and ImageNet. We omit CIFAR-100 and BiT-M-R152x4. Our attack results are shown in Table 7. In general, it can be seen that single models are not robust to the RayS attack, as no model has more than 30% robust accuracy.

Table 5. White-box attack parameters for ImageNet.

Attack	Parameters
FGSM	ϵ = 0.062
PGD	ϵ = 0.062, ϵ_{step} = 0.0031, steps = 20
BPDA	ϵ = 0.062, steps = 100, max iterations = 100, learning rate = 0.5
MIM	ϵ = 0.062, ϵ_{step} = 0.0031, decay factor = 1.0
CW	confidence = 50, step size = 0.0031, steps = 30
APGD	ϵ = 0.062, number of restarts = 1, ρ = 0.75, n^2 queries = 5000

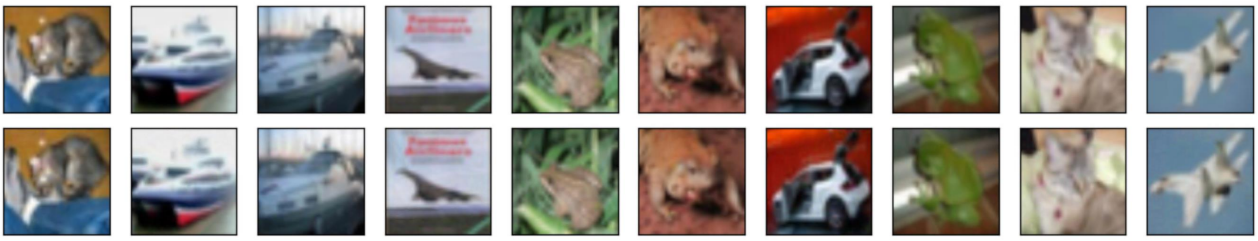

Figure 4. Adversarial images generated using SAGA on CIFAR-10. The top row of images are the the clean images generated from the CIFAR-10 test set. The bottom row of images are the adversarial images generated using SAGA with the l_∞ norm and $\epsilon = 0.031$. These images correspond to SAGA when the models are ViT-L-16 and BiT-M-R101x3. Visually, there is very little perceivable difference between the clean and adversarial images generated by SAGA.

D.2. Transfer Based Adversarial Model

Adversarial Capabilities: The transfer based adversary is granted a wide range of abilities. Specifically, a transfer based adversary may know part or all of the original training data (X, Y) for C and may have access to the pre-training data $(X^{'}, Y^{'})$. Unlike query based adversaries, the transfer based adversary is not restricted by the number of queries made to C. The only unknowns to the adversary are the architecture classifier for C and the trained parameters θ. The general strategy for the transfer based adversary is as follows: the attacker starts with an untrained classifier S. Note that S is often referred to as the synthetic model. If the adversary has access to the pre-training data, they start by training S with $(X^{'}, Y^{'})$. The adversary then queries C to label the training set X. They then train S on $(X, \hat{Y})$, where $\hat{Y}$ are the hard class labels obtained from C. Once S has been trained, a white-box attack A can be run on S to generate adversarial examples. These examples are then applied to C in the hopes that the adversarial samples are able to *transfer* from S to C.

Attack Setup and Discussion: Several components must be selected for a transfer attack. These components include the synthetic architecture S, the percentage of training data (X, Y) visible to the adversary, and the type of white-box attack A that will be used on S to generate adversarial examples. Ideally, we want to test under the strongest possible adversary. This means a careful choice of S and utilizing 100% of the training data. However, as these experiments are time consuming - each attack requires training a synthetic model from scratch - we first conduct several smaller scale experiments to help us choose the hyperparameters for the main attack. These results are shown in Table D.2 and Figure 6. For A, we use the MIM attack to generate samples with S. We set the maximum perturbation $\epsilon = 0.031$ for CIFAR-10 and experiment with a range of different synthetic models S.

From the hyperparameter experiments, we can observe several interesting results. First, when attacking a Vision Transformer such as ViT-L-16, the choice of synthetic model greatly affects the robust accuracy. Even when only 10% of the data is available, if S is a Vision Transformer (ViT-B-32) and it is pre-trained on ImageNet-21K, the robust accuracy of ViT-L-16 is only 53%. If the attacker uses a synthetic model that is NOT pre-trained (but still ViT-B-32), the robust accuracy is 92.4%.This presents a new challenge for the attacker. Originally, the architecture of the synthetic model did not greatly affect the performance of the attack in attacks on CNNs [29]. This is likely due to the fact that these attacks were transferring samples from CNNs to other CNNs. However, the same result does not hold for Vision Transformers: using a CNN (like VGG-16) does not give a very high attack success rate. We can see that when we do a 100% strength attack on ViT-L-16 using VGG-16, the robust accuracy is still 46.8%. Comparing this result to the same attack with ResNet-56, it can be seen that the robust accuracy is only 4.8%.

The goal of our hyperparameter experiments was to find an appropriate set of parameters for attacking Vision Transformer based defenses. Our experiments are successful: we can see that using a pre-trained ViT-B-32 with even 10% of the training data gives good attack results. Additionally, our experiments reveal a critical concept. Unlike CNN-based

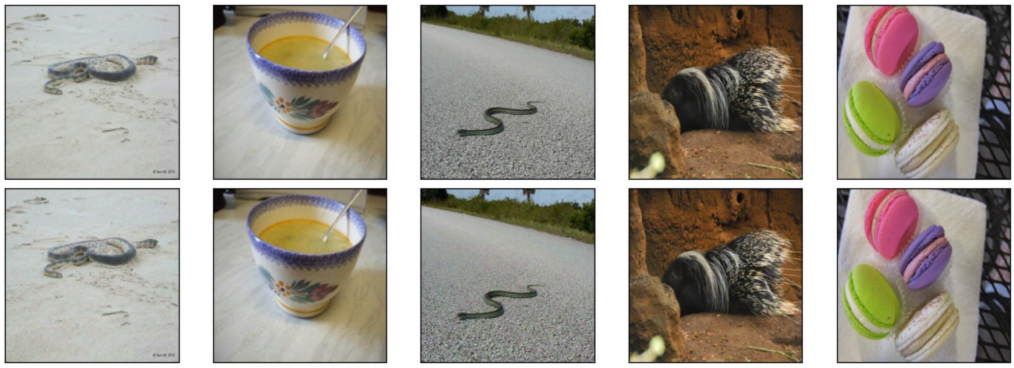

Figure 5. Adversarial images generated using SAGA on ImageNet. The top row of images are the the clean images generated from the ImageNet validation set. The bottom row of images are the adversarial images generated using SAGA with the l_∞ norm and $\epsilon = 0.062$. These images correspond to SAGA when the models are ViT-L-16 and BiT-M-R152x4. Visually, there is very little perceivable difference between the clean and adversarial images generated by SAGA.

Table 6. Self-Attention Gradient Attack (SAGA) results for CIFAR-10, CIFAR-100 and ImageNet. In the table α_1 represents the coefficient used to scale the gradient of the ViT model and α_2 represents the coefficient used to scale the gradient of the respective CNN. In the table ViT corresponds to ViT-L-16 and BiT corresponds to BiT-M-R101x3 for CIFAR-10 and CIFAR-100 and BiT-M-R152x4 for ImageNet. ResNet corresponds to ResNet-164.

	CIFAR-10				
	α_1	α_2	Robust Acc ViT	Robust Acc CNN	Average Robust Acc
ViT/BiT	0.5	0.5	94.9%	0.1%	47.5%
ViT/BiT	0.9998	2.00E-04	27.3%	24.7%	26.0%
ViT/ResNet	0.5	0.5	7.3%	38.3%	22.8%
ViT/ResNet	0.01	0.99	15.1%	14.8%	15.0%
	CIFAR-100				
	α_1	α_2	Robust Acc ViT	Robust Acc CNN	Average Robust Acc
ViT/BiT	0.5	0.5	3.7%	48.9%	26.3%
ViT/BiT	0.9985	0.0015	16.7%	14.5%	15.6%
	ImageNet				
	α_1	α_2	Robust Acc ViT	Robust Acc CNN	Average Robust Acc
ViT/BiT	0.5	0.5	56.7%	0.2%	28.5%
ViT/BiT	0.99	0.01	13.3%	12.0%	12.7%

transfer attacks where the choice of architecture was trivial, initial experiments show that Vision Transformers mandate a careful choice of synthetic model S. By merely using Vision Transformers in a defense, the transfer based attacker is put at a new disadvantage. It is left to future work to explore this concept in-depth, as this poses an interesting new challenge for black-box attack design.

E. Discussion on White-Box and Transfer Attacks on the Vision Transformer

The Vision Transformer as reported in [12] is an encoder-based architecture. This architecture is an adaptation of the transformer architecture popular in Natural Language Processing applications. While the original transformer from Vaswani et al. [35] used both encoders and decoders for sequence-to-sequence applications, the Vision Transformer is purely encoder-based. The input image to be fed to the transformer is divided into equally-sized patches, which are sequentially passed through an embedding layer. Positional encoding is added to the embedding vector feeding to a layer of encoders. The output dimension of each encoder matches the input dimension, which makes it easy to stack these encoders. The output from the last encoder is fed to a linear network layer acting as a classifier.

The building blocks of an encoder are the Attention network, followed by Batch Normalization with skip connections. Many attention blocks are used in parallel (similar to feature maps in CNNs), which are referred to as the "multi-headed self-attention network". The self-attention block uses three linear networks of query, key and value parametrized by W_k, W_q, and W_v matrices. The query and the key correspond to the positions of the input patches in image X for which we are interested in computing the atten-

Table 7. RayS attack on single classifiers for CIFAR-10 and ImageNet. The robust accuracy for each model is reported in the table.

	RayS CIFAR-10
ResNet-56	0.8%
ResNet-164	0.0%
ViT-B-16	8.2%
ViT-B-32	11.1%
ViT-L-16	14.5%
R50-ViT-B-16	22.9%
BiT-M-R50x1	0.9%
BiT-M-R101x3	3.7%
	RayS ImageNet
ResNet-50	3.1%
ResNet-152	2.7%
ViT-B-16-224	1.6%
ViT-L-16	25.9%
ViT-L-16-224	3.3%
BiT-M-R50x1	3.1%

Table 8. Results of CIFAR-10 hyperparameter experiments for transfer attacks using different strength attacks and different synthetic models.

Defense Model	Synthetic Model	Attack Strength	Robust Acc
ViT-B-16	VGG-16	10.0%	79.9%
ViT-B-16	VGG-16	100.0%	46.8%
ViT-L-16	VGG-16	10.0%	84.4%
ViT-L-16	ViT-B-32	10.0%	92.4%
ViT-L-16	ViT-B-32 (ImageNet-21K)	10.0%	53.0%
ResNet-56	VGG-16	10.0%	22.6%
ResNet-56	VGG-16	100.0%	4.8%
BiT-M-R101x3	VGG-16	10.0%	66.1%

tion with respect to each other. Once we have computed the self-attention of the entire input set, we turn it into a probability distribution by the Softmax function. An encoder uses multiple attention computations in parallel, where each attention block is referred to as an attention head. Let N be the number of p x p patches in an n x m image, and e be the embedding size for each patch, then the positionally encoded input to the encoder is:

$$X_p = X + P \tag{18}$$

where X, $P \in \mathbb{R}^{N \times e}$, and P is the positional encoding for the image patches. The computation for each attention head i in terms of key, query and value networks can be described as:

$$K_i = X_p W_{k,i} \tag{19}$$

$$Q_i = X_p W_{q,i} \tag{20}$$

$$V_i = X_p W_{v,i} \tag{21}$$

The self-attention A_i in an attention head i with n_h number of heads is computed as:

$$A_i = [softmax(Q_i K_i^T)/n_h]V_i \tag{22}$$

The output from all attention heads is concatenated and passed through a linear layer parameterized by W_o, as shown below:

$$M_A = [concat_{i \in n_h}[A_i]]W_o + b_o \tag{23}$$

In a transformer, the multi-headed self-attention passes through a Batch Normalization layer with the input being added to the output of Batch Normalization in a ResNet-like manner. It then passes through a linear and another batch normalization layer. Thus, the output of the first encoder Enc as a function of the position encoded input, X_p, can be described as:

$$Enc(X_p) = batchnorm([[batchnorm(M_A)+ X_p]W_l + b_l]) + [batchnorm(M_A) + X_p] \tag{24}$$

where W_l and b_l are the parameters of the linear network of the encoder. After passing the input through a series of encoders, a classifier is connected to the last output of the final encoder, as:

$$y = Classifier(Enc(\ldots Enc(X_p)\ldots)) \tag{25}$$

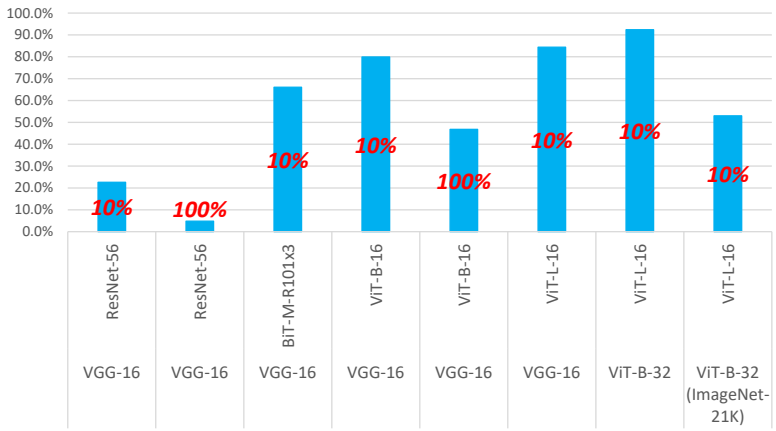

Figure 6. Black-box transfer attack hyperparameter experimental results for CIFAR-10. The text in red on top of the bars indicates the strength of the attack (what percent of the training data is available to the adversary). The bars themselves represent the robust accuracy of the model under attack. The horizontal text represents the type of synthetic model S, used in the attack. The vertical text represents the model being attacked. This experiment is important because it indicates which type of synthetic model works best when doing the actual attack.

For any differentiable loss function L, operating on the output of the transformer as given by Equation 25, the computation of $\partial L/\partial x = \nabla_x(L)$ requires that all components of Equation 25 be differentiable with respect to the input x. Since Equations 19 - 24 that contribute to Equation 25 including the Batch Normalization are all closed-form differentiable with respect to position encoded input X_p, and since X_p is a simple function of x, $\nabla_x(L)$ can be computed. Thus an adversarial image can be efficiently created using a white-box attack formulation. We confirm this empirically, as all white-box attacks successfully compromise the classifier accuracies, resulting in zero robust accuracy for many white-box attacks.

E.1. Transfer Attacks

From a black box adversarial robustness point of view, one category of attacks is referred to as transfer attacks. Here a new network model is created (referred to as the synthetic model) and trained either on the same dataset as the model under attack, or creating training data from querying the input-output behavior of the target model. One fundamental question to ask is how the transformer-based models behave with respect to transfer attacks from a CNN-based synthetic model and vice versa. An equivalency of the transformer model with the CNN based models under some simplified assumptions was presented in [9]. The multi-headed attention at pixel q for attention head h is expressed as (this is similar to our development in equation 23):

$$M_A(X)_{q,:} = \sum_{h\in[N_h]} (\sum_k softmax(A^h_{q,:})_k X_{k,:})W^{(h)} + b_{out} \tag{26}$$

For the h-th attention head, the attention probability is one when $k = q - f(h)$ and zero otherwise. The layer's output at pixel q is then shown to be equal to:

$$M_A(X)_q = \sum_{h\in[N_h]} X_{q-f(h),:}W^{(h)} + b_{out} \tag{27}$$

The above can be seen to be equivalent to the convolution operation. The development in [9] as shown above demonstrates an equivalence in the transformer and the CNN. Here we mean equivalence in the sense that the transformer can equivalently perform a $k \times k$ convolution if an appropriate value matrix W_v is chosen in an attention head. Whether a transformer actually learns the appropriate W_v to perform the equivalent convolution in practice is difficult to determine. In the following subsection we seek to partially answer this question. We do this through a series of empirical experiments, where we study the decision regions created by transformers and CNNs.

E.2. Decision Region Graphs

One way to visually comprehend the transferability between different models is to examine their decision region graphs. A decision region graph is a visual representation of the different classification regions of a model using a color coded 2-D graph. Decision region graphs for CNNs trained on ImageNet were originally shown in [22].

For every dataset and model in this paper, we construct the decision region graph. Formally, we can describe the generation of the graph as follows: Each graph is constructed with respect to a single image I. For every model, we use the same image I to build the graph (i.e. we use sample 49443 from the validation set of ImageNet). Every

point on the graph corresponds to a class label for the given image. The origin $(x = 0, y = 0)$ corresponds to the original (unperturbed) image. Outside the origin, the image is perturbed according to the following equation:

$$I' = I + x \cdot g + y \cdot r \tag{28}$$

where I' represents the new perturbed image, I represents the original image, g represents the gradient of the image with respect to the loss function of the model, and r represents a random noise orthogonal to g. In Equation 28, x and y represent coordinates on the graph which control the magnitude of the adversarial noise g and random noise r.

The decision region graphs may be slightly difficult to grasp at first but it comes with a natural intuitive explanation. The origin of the graph represents the unperturbed image I with the correct class label. As we move in the x direction on the graph, we increase the magnitude of the adversarial noise g that is added to I. This is analogous to an FGSM attack using I in which we keep increasing the size of the step (ϵ in Equation 10). As we move in the y direction on the graph, this represents adding more and more random noise to I. When we move in both the x and y directions, it represents a combination of adding random noise and adversarial noise to the image. The last component of the graph, color, represents the class label that the model produces based on the perturbed input I'. Essentially, a decision region graph gives intuition about how the model classifies images that are noisy and adversarial. The decision region graphs for CIFAR-10, CIFAR-100 and ImageNet are shown in Figures 7, 8 and 9.

Decision Region Graph Analysis: In Figure 7, the correct class label is represented by the color red. As we can see at the origin, all models correctly classify the sample. It can be noted that for the ResNets (ResNet-56 and ResNet-164), their robustness is quite limited. We can see for both these models there is only a small red sphere around the origin. As we move to larger and larger perturbations, the image quickly becomes misclassified (the blue regions). For the Vision Transformers and Big Transfer Models, we can see that they are much more tolerant of noise. For example, if we consider moving along the y axis (adding random noise), none of the Vision Transformers misclassify the image.

For Figures 8 and 9, we can see a similar trend applies. In Figure 8, light blue represents the correct class label and in Figure 9, dark blue represents the correct class label. In general, for both these figures we see the Vision Transformers tend to handle random noise well (see along the y axis) and the ResNets are very sensitive to perturbations. It should also be noted in Figure 9, the graph for BiT-M-R152x4 is completely dark blue. This means that despite large perturbations, the model never fails to correctly classify the image I. This should not be completely surprising as BiT-M-R152x4 is one of the most complex models (in terms of number of parameters) that we experiment with. We mention complexity because it has been previously noted that model complexity alone helps thwart adversarial attacks [24].

There is one other important take away, the landscape of the decision regions themselves are very different between model genuses. In Figure 7, for the ResNet models we can see a small sphere of red surrounded by blue, while for the Vision Transformers we can see large red regions around the y axis and large light blue regions as we move in the x direction. While we cannot directly make conjectures based on visualizations, the graphs do tend to support our main findings. Specifically, we know from the results in Table 2 that the transferability between different models genuses is low. The decision region graphs lend credence to this claim by visually showing that the decisions regions between Vision Transformers, ResNets and Big Transfer Models do indeed look very different. Thus, we conjecture that even though a Vision Transformer is capable of implementing convolutions as described in [9], in practice we observe that this may not be the case due to differing patterns of decision boundaries.

F. Friendly Adversarial Training Defense for Vision Transformers

Friendly Adversarial Training (FAT) [38] was proposed to improve the adversarial defense of deep networks. It is a simple training technique that uses less strong adversarial examples by employing an early stopping of the PGD algorithm. By incorporating another parameter, τ, in the PGD k-step algorithm (referred to as PGD-K-τ), the step amount τ by which the adversarial example crosses the decision boundary can be easily controlled. The pseudocode for the FAT algorithm [38] is described below:

while $K > 0$ **do**
  if arg max$_i$ $f(\widetilde{x}) \neq y$ and $\tau = 0$ **then**
    break
  else if arg max$_i$ $f(\widetilde{x}) \neq y$ **then**
    $\tau \leftarrow \tau - 1$
  end if
  $\widetilde{x} \leftarrow P(\alpha \text{ sign}(\nabla_{\widetilde{x}} l(f(\widetilde{x}), y)) + \widetilde{x})$
  $K \leftarrow K - 1$
end while

where arg max$_i f(\widetilde{x})$ outputs the predicted label of the adversarial sample $\widetilde{x}$, and $f(\widetilde{x}) = (f^i(\widetilde{x}))^\top_{i=0,\ldots,C-1}$ gives the probabilistic predictions over C classes. From the pseudocode it can be seen that, if $K = \tau$, the PGD-K-τ algorithm becomes equivalent to the standard PGD k-step

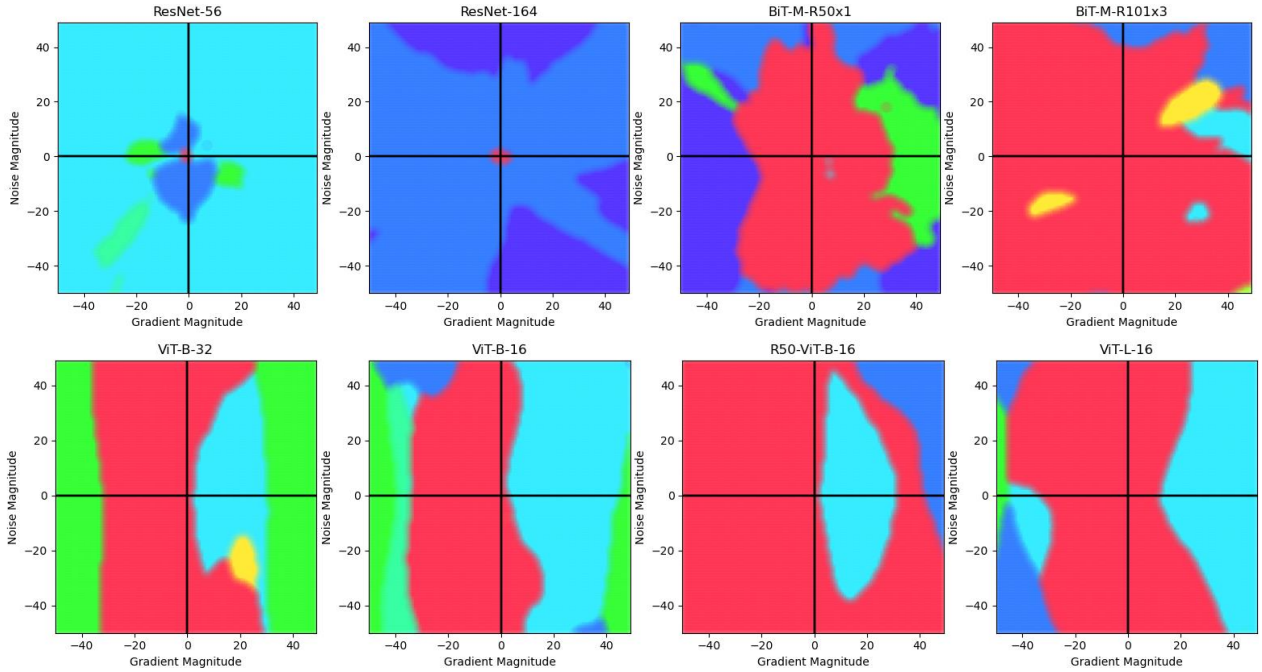

Figure 7. Vision Transformer, Big Transfer Model and ResNet decision regions for CIFAR-10.

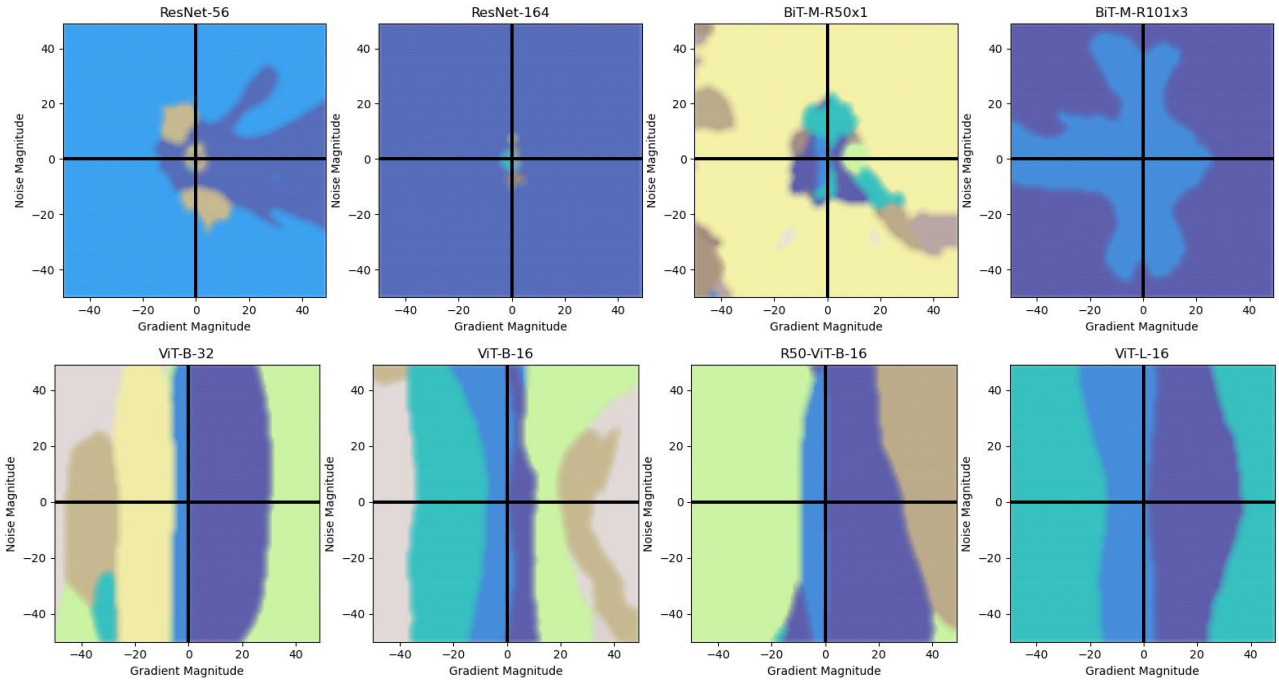

Figure 8. Vision Transformer, Big Transfer Model and ResNet decision regions for CIFAR-100.

algorithm. The application of FAT for creating an adversarial defense for ResNets had a slightly better robust accuracy than e.g., Madry training, and a relatively lower drop in clean accuracy [38]. For example, with FAT on the Wide ResNet WRN-30-10, the clean accuracy dropped from $\approx 95\%$ to $\approx 89\%$ as reported in [38], resulting in a robust accuracy of $\approx 46\%$ when the PGD-20 attack was used.

We evaluate the performance of different transformer-based networks for different values of τ. The results are presented in Table 9. From Table 9, it can be seen that the adversarial robustness of the Vision Transformers with re-

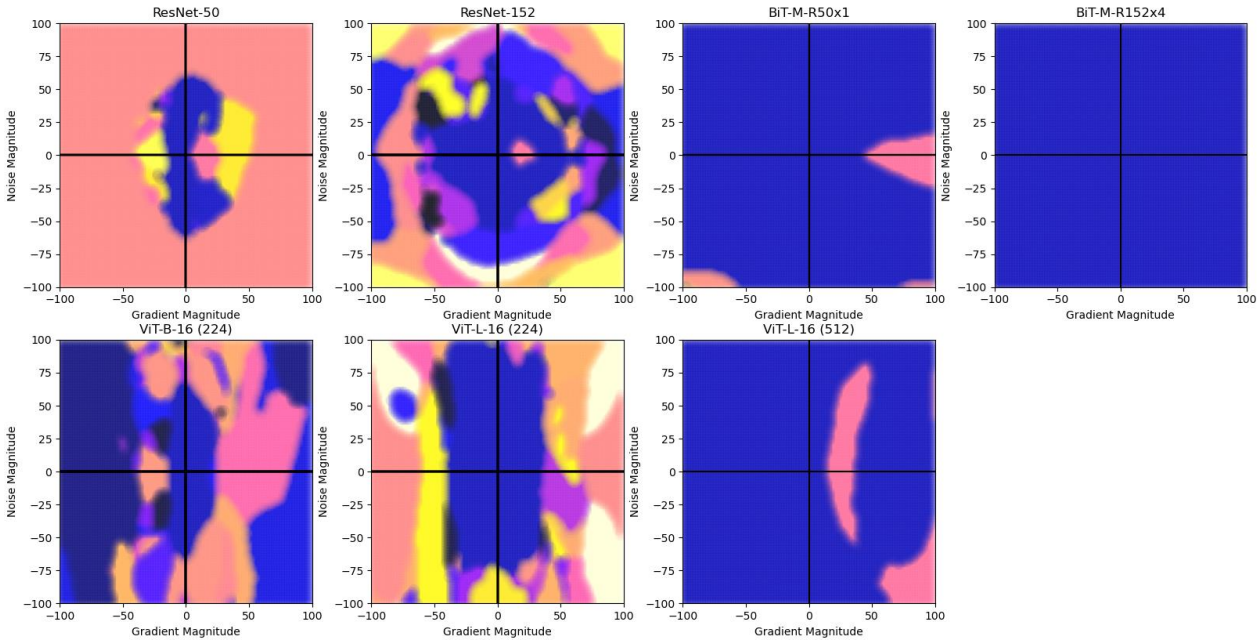

Figure 9. Vision Transformer, Big Transfer Model and ResNet decision regions for ImageNet.

spect to FAT is similar to the ResNet-based architectures.

For CIFAR-10, it can be seen that the ViT-L-16 attains a clean accuracy of 99.1%, the highest of the models listed in Table 9. When utilizing FAT as a defense, the clean accuracy of ViT-L-16 drops to 94.2% for $\tau = 1$; the strongest attack, APGD, produces a robust accuracy of just 19.6%. For $\tau = 10$, the clean accuracy of ViT-L-16 drops to 85.3%; the robust accuracy of APGD approximately doubles to 33.6%. For the ViT-B-32, a clean accuracy of 98.6% was obtained for CIFAR-10. The implementation of FAT dropped the clean accuracy to 93.2% for $\tau = 1$; APGD is very successful here, producing a robust accuracy of just 5.9%. For $\tau = 10$, the clean accuracy of the ViT-B-32 is 78.9%; the robust accuracy of APGD rises to 29.4%.

Considering CIFAR-100, the ViT-L-16 in Table 9 attains a clean accuracy of 94.0%. Using FAT, the clean accuracy drops to 83.0% for $\tau = 1$, and APGD produces a robust accuracy of 6.7%. For $\tau = 10$, the clean accuracy of ViT-L-16 further drops to 64.1%; the robust accuracy of APGD increases to 15.3%. Finally, for the ViT-B-32, a clean accuracy of 91.7% was obtained for CIFAR-100. Implementing FAT reduced the clean accuracy to 81.3% for $\tau = 1$; here, APGD produces an extremely low robust accuracy of 2.9%. For $\tau = 10$, the clean accuracy of the ViT-B-32 is just 58.6%, while the robust accuracy of APGD increases to 18.3%.

G. Additional Tables And Codes

In this section, we provide full numerical tables for some of the charts and figures presented in our paper. Each table is captioned with a description of the corresponding portion of the main paper.

We also provide code and the trained models to replicate our results. Code for the ViT/BiT defense, the RayS attack, Adaptive attack and SAGA for CIFAR-10 can be found on Github: https://github.com/MetaMain/ViTRobust.

Table 9. FAT defense accuracy for ViT-B-32, ViT-B-16, ViT-L-16, and ResNet-164 architectures on CIFAR-10 and CIFAR-100. The leftmost column in the table lists the model being tested; each model includes a subset of τ parameters for $\tau = 0, 1, 2, 10$. The top row in the table lists the attacks run, from FGSM to APGD. The last column in the table lists the clean accuracy of the tested model.

	CIFAR-10						
	FGSM	PGD	BPDA	MIM	C&W	APGD	Acc
ViT-B-32	37.9%	1.8%	17.6%	4.4%	0.0%	0.0%	98.6%
$\tau = 0$	30.8%	17.5%	14.5%	17.3%	1.5%	1.5%	95.5%
$\tau = 1$	33.9%	32.2%	16.4%	23.6%	9.7%	5.9%	93.2%
$\tau = 2$	37.8%	40.3%	23.6%	31.4%	20.8%	13.9%	90.8%
$\tau = 10$	42.6%	51.1%	33.3%	38.8%	34.1%	29.4%	78.9%
ViT-B-16	39.5%	0.0%	20.3%	0.3%	0.0%	0.0%	98.9%
$\tau = 0$	42.3%	34.0%	19.2%	29.0%	9.4%	4.1%	95.9%
$\tau = 1$	43.2%	41.1%	26.3%	33.7%	19.3%	13.7%	93.8%
$\tau = 2$	62.3%	25.4%	33.9%	25.1%	6.7%	5.4%	93.8%
$\tau = 10$	43.1%	52.3%	36.8%	40.2%	35.0%	33.7%	73.3%
ViT-L-16	56.3%	1.2%	28.70%	5.9%	0.0%	0.0%	99.1%
$\tau = 0$	51.7%	43.6%	29.2%	39.3%	20.6%	15.4%	95.7%
$\tau = 1$	49.1%	47.0%	31.9%	39.4%	26.8%	19.6%	94.2%
$\tau = 2$	57.4%	48.8%	33.5%	40.2%	29.4%	21.8%	92.4%
$\tau = 10$	49.5%	55.4%	33.7%	45.8%	37.7%	33.6%	85.3%
ResNet-164	14.4%	3.0%	9.0%	2.2%	0.1%	0.0%	93.2%
$\tau = 0$	47.7%	50.8%	39.5%	42.5%	34.6%	27.0%	90.3%
$\tau = 1$	53.0%	56.2%	47.2%	49.0%	42.7%	34.4%	88.0%
$\tau = 2$	56.2%	61.3%	50.0%	51.9%	46.9%	37.8%	86.4%
$\tau = 10$	60.4%	64.8%	55.6%	57.6%	51.9%	44.5%	79.9%
	CIFAR-100						
	FGSM	PGD	BPDA	MIM	C&W	APGD	Acc
ViT-B-32	20.8%	1.9%	13.4%	3.1%	0.0%	0.0%	91.7%
$\tau = 0$	16.2%	6.9%	9.7%	7.6%	0.7%	1.2%	87.6%
$\tau = 1$	21.6%	16.5%	9.3%	12.8%	5.2%	2.9%	81.3%
$\tau = 2$	24.4%	23.0%	12.3%	17.5%	10.1%	5.4%	76.1%
$\tau = 10$	26.4%	31.9%	20.5%	24.3%	20.1%	18.3%	58.6%
ViT-B-16	20.4%	0.0%	11.9%	0.5%	0.0%	0.0%	92.8%
$\tau = 0$	16.6%	8.0%	7.6%	7.5%	0.5%	0.3%	87.5%
$\tau = 1$	23.9%	20.0%	9.3%	15.7%	8.5%	4.8%	82.0%
$\tau = 2$	26.0%	25.0%	12.8%	18.4%	13.3%	8.4%	77.0%
$\tau = 10$	25.1%	29.0%	20.4%	22.7%	16.5%	16.1%	54.2%
ViT-L-16	33.0%	1.6%	15.1%	4.7%	0.0%	0.0%	94.0%
$\tau = 0$	28.6%	19.1%	13.1%	17.7%	5.3%	5.2%	87.7%
$\tau = 1$	27.7%	22.6%	14.3%	18.1%	11.4%	6.7%	83.0%
$\tau = 2$	30.2%	26.5%	16.7%	22.0%	16.0%	9.7%	80.0%
$\tau = 10$	31.4%	32.0%	23.0%	24.0%	20.2%	15.3%	64.1%
ResNet-164	7.6%	0.3%	3.7%	0.9%	0.0%	0.0%	74.2%
$\tau = 0$	18.2%	16.1%	13.5%	12.2%	9.7%	6.8%	70.8%
$\tau = 1$	23.5%	24.4%	19.3%	18.4%	17.3%	10.7%	66.8%
$\tau = 2$	35.3%	32.3%	25.2%	26.3%	24.7%	17.6%	61.8%
$\tau = 10$	45.6%	43.2%	34.5%	35.7%	28.9%	27.2%	55.0%

Table 10. Full transferability results for CIFAR-10. The first column in each table represents the model used to generate the adversarial examples, C_i. The top row in each table represents the model used to evaluate the adversarial examples, C_j. Each entry represents $1 - t_{i,j}$ (the robust accuracy) computed using equation 3 with C_i, C_j and either FGSM, PGD or MIM. For PGD and MIM we use only 10 steps to avoid overfitting the example to a particular model. Based on these results we take the maximum transferability across all attacks and report the result in Table 2. We also visually show the maximum transferability $t_{i,j}$ in figure 1.

FGSM

	ViT-B-32	ViT-B-16	ViT-L-16	R50-ViT-B-16	BiT-M-R50x1	BiT-M-R101x3	ResNet-56	ResNet-164
ViT-B-32	43.4%	55.3%	61.1%	76.6%	67.7%	68.9%	83.3%	83.3%
ViT-B-16	68.7%	41.3%	56.9%	80.3%	73.0%	73.7%	86.1%	86.0%
ViT-L-16	74.8%	61.6%	59.5%	82.7%	78.5%	80.1%	88.5%	88.5%
R50-ViT-B-16	82.6%	75.9%	79.9%	51.4%	72.2%	74.1%	81.9%	82.0%
BiT-M-R50x1	96.0%	94.0%	95.9%	95.6%	69.5%	86.0%	94.0%	93.8%
BiT-M-R101x3	97.4%	94.4%	86.3%	96.2%	86.1%	88.0%	95.5%	94.7%
ResNet-56	93.4%	92.4%	94.7%	91.2%	82.0%	91.2%	41.9%	43.0%
ResNet-164	93.2%	92.4%	95.3%	90.9%	82.9%	91.7%	45.1%	47.1%

PGD

	ViT-B-32	ViT-B-16	ViT-L-16	R50-ViT-B-16	BiT-M-R50x1	BiT-M-R101x3	ResNet-56	ResNet-164
ViT-B-32	4.2%	49.1%	72.2%	93.4%	70.4%	74.3%	93.0%	92.9%
ViT-B-16	90.4%	0.4%	53.9%	97.1%	83.3%	85.1%	96.1%	95.9%
ViT-L-16	85.9%	32.4%	10.4%	94.4%	82.1%	81.7%	95.7%	95.5%
R50-ViT-B-16	93.7%	84.5%	91.3%	6.6%	69.7%	75.3%	90.3%	88.9%
BiT-M-R101x3	99.6%	97.5%	86.3%	98.7%	58.0%	0.0%	97.3%	96.8%
BiT-M-R50x1	99.9%	98.5%	99.4%	99.5%	0.0%	85.9%	98.2%	97.8%
ResNet-56	99.0%	97.9%	98.6%	98.4%	93.9%	96.9%	28.0%	28.7%
ResNet-164	98.7%	98.3%	99.1%	98.1%	92.6%	96.1%	28.7%	32.5%

MIM

	ViT-B-32	ViT-B-16	ViT-L-16	R50-ViT-B-16	BiT-M-R50x1	BiT-M-R101x3	ResNet-56	ResNet-164
ViT-B-32	4.9%	15.9%	24.5%	65.1%	39.2%	38.0%	81.4%	80.1%
ViT-B-16	42.9%	0.9%	11.1%	77.4%	56.6%	55.0%	86.9%	86.0%
ViT-L-16	44.4%	21.6%	13.4%	69.7%	57.5%	55.3%	87.0%	85.2%
R50-ViT-B-16	60.4%	41.9%	48.5%	1.7%	39.0%	42.0%	73.3%	71.0%
BiT-M-R50x1	95.5%	89.1%	94.3%	95.3%	0.0%	48.6%	93.0%	91.0%
BiT-M-R101x3	91.4%	79.7%	88.0%	92.8%	24.1%	0.1%	92.2%	90.7%
ResNet-56	94.2%	91.0%	94.8%	90.3%	77.5%	88.2%	14.1%	12.8%
ResNet-164	94.2%	91.9%	95.0%	90.3%	77.7%	88.8%	16.4%	14.3%

Table 11. Full transferability results for CIFAR-100. The first column in each table represents the model used to generate the adversarial examples, C_i. The top row in each table represents the model used to evaluate the adversarial examples, C_j. Each entry represents $1 - t_{i,j}$ (the robust accuracy) computed using equation 3 with C_i, C_j and either FGSM, PGD or MIM. For PGD and MIM we use only 10 steps to avoid overfitting the example to a paticular model. Based on these results we take the maximum transferability across all attacks and report the result in Table 2.

FGSM

	ViT-B-32	ViT-B-16	ViT-L-16	R50-ViT-B-16	BiT-M-R50x1	BiT-M-R101x3	ResNet-56	ResNet-164
ViT-B-32	27.9%	40.2%	41.7%	59.2%	55.8%	57.4%	85.1%	86.0%
ViT-B-16	50.7%	25.3%	36.7%	64.9%	61.4%	59.9%	91.0%	92.5%
ViT-L-16	57.0%	41.7%	37.8%	65.7%	64.7%	65.1%	90.1%	90.5%
R50-ViT-B-16	66.3%	60.6%	64.3%	30.8%	56.1%	61.2%	90.8%	91.4%
BiT-M-R50x1	87.3%	83.1%	87.2%	86.0%	44.5%	68.8%	96.3%	96.6%
BiT-M-R101x3	85.5%	83.3%	85.8%	86.4%	70.4%	67.0%	96.2%	97.8%
ResNet-56	79.9%	77.8%	84.7%	77.3%	68.6%	78.1%	38.0%	40.8%
ResNet-164	77.9%	75.5%	84.5%	75.8%	64.9%	75.7%	36.1%	33.2%

PGD

	ViT-B-32	ViT-B-16	ViT-L-16	R50-ViT-B-16	BiT-M-R50x1	BiT-M-R101x3	ResNet-56	ResNet-164
ViT-B-32	3.8%	36.4%	53.5%	80.9%	67.7%	68.5%	93.3%	95.0%
ViT-B-16	78.6%	0.7%	36.1%	90.3%	78.9%	80.2%	97.2%	97.6%
ViT-L-16	72.2%	17.9%	5.8%	85.0%	76.6%	75.2%	96.1%	96.2%
R50-ViT-B-16	85.6%	75.5%	82.3%	2.2%	62.7%	68.5%	94.3%	95.3%
BiT-M-R50x1	96.1%	94.7%	97.8%	96.1%	0.0%	76.9%	97.5%	98.7%
BiT-M-R101x3	94.8%	91.7%	95.2%	94.1%	52.3%	1.0%	97.9%	97.9%
ResNet-56	91.6%	91.0%	94.5%	89.4%	82.0%	89.9%	51.2%	56.6%
ResNet-164	89.2%	89.0%	92.8%	88.5%	78.4%	85.7%	43.6%	39.4%

MIM

	ViT-B-32	ViT-B-16	ViT-L-16	R50-ViT-B-16	BiT-M-R50x1	BiT-M-R101x3	ResNet-56	ResNet-164
ViT-B-32	4.4%	11.5%	16.4%	47.8%	39.5%	38.9%	86.1%	87.4%
ViT-B-16	28.7%	0.9%	6.8%	61.4%	55.5%	52.1%	91.4%	93.3%
ViT-L-16	32.2%	11.7%	7.5%	51.9%	52.4%	50.0%	91.3%	90.5%
R50-ViT-B-16	48.4%	35.0%	37.7%	1.1%	35.9%	38.8%	89.0%	90.1%
BiT-M-R50x1	82.3%	75.0%	84.5%	81.8%	0.1%	43.5%	95.1%	94.8%
BiT-M-R101x3	75.1%	61.0%	73.7%	76.5%	26.0%	1.2%	94.3%	96.8%
ResNet-56	81.4%	80.5%	86.4%	78.9%	69.3%	79.5%	29.2%	31.1%
ResNet-164	78.3%	77.2%	84.8%	76.5%	64.1%	73.5%	25.5%	20.8%

Table 12. Full transferability results for ImageNet. The first column in each table represents the model used to generate the adversarial examples, C_i. The top row in each table represents the model used to evaluate the adversarial examples, C_j. Each entry represents $1 - t_{i,j}$ (the robust accuracy) computed using equation 3 with C_i, C_j and either FGSM, PGD or MIM. For PGD and MIM we use only 10 steps to avoid overfitting the example to a paticular model. Based on these results we take the maximum transferability across all attacks and report the result in Table 2. Note due to the complexity of training ImageNet models we do not train independent copies of the model to measure self-transferability (when $i = j$).

FGSM

	ViT-B-16	ViT-L-16 (224)	ViT-L-16 (512)	BiT-M-R50x1	BiT-M-R152x4	ResNet-50	ResNet-152
ViT-B-16	+	40.8%	67.3%	63.2%	73.2%	56.0%	63.6%
ViT-L-16 (224)	40.1%	+	59.6%	63.7%	75.4%	57.6%	61.7%
ViT-L-16 (512)	77.8%	69.3%	+	74.6%	77.7%	74.5%	78.4%
BiT-M-R50x1	90.6%	91.6%	89.4%	+	83.3%	81.2%	83.5%
BiT-M-R152x4	93.0%	93.9%	89.8%	83.4%	+	86.8%	90.1%
ResNet-50	77.8%	82.3%	79.1%	61.6%	79.2%	+	46.7%
ResNet-152	75.6%	78.8%	77.9%	61.0%	78.1%	40.7%	+

PGD

	ViT-B-16	ViT-L-16 (224)	ViT-L-16 (512)	BiT-M-R50x1	BiT-M-R152x4	ResNet-50	ResNet-152
ViT-B-16	+	36.1%	81.1%	83.1%	89.7%	79.0%	81.7%
ViT-L-16 (224)	22.7%	+	62.6%	83.2%	88.8%	80.4%	80.6%
ViT-L-16 (512)	89.6%	83.5%	+	84.3%	87.6%	87.6%	89.9%
BiT-M-R50x1	96.5%	96.8%	95.8%	+	90.6%	89.2%	91.4%
BiT-M-R152x4	91.8%	97.3%	94.2%	85.4%	+	93.0%	95.2%
ResNet-50	92.7%	94.2%	91.8%	77.8%	92.7%	+	42.2%
ResNet-152	91.1%	93.3%	90.5%	77.4%	90.9%	30.1%	+

MIM

	ViT-B-16	ViT-L-16 (224)	ViT-L-16 (512)	BiT-M-R50x1	BiT-M-R152x4	ResNet-50	ResNet-152
ViT-B-16	+	10.9%	60.4%	59.2%	72.6%	56.6%	59.9%
ViT-L-16 (224)	9.1%	+	35.5%	60.0%	73.1%	56.3%	59.2%
ViT-L-16 (512)	72.0%	56.6%	+	65.7%	73.7%	71.6%	76.8%
BiT-M-R50x1	90.2%	91.6%	88.2%	+	75.1%	75.3%	81.3%
BiT-M-R152x4	96.2%	92.4%	86.5%	72.0%	+	84.9%	88.0%
ResNet-50	76.2%	81.2%	75.3%	44.7%	75.6%	+	13.3%
ResNet-152	74.1%	77.9%	73.4%	45.9%	73.2%	10.6%	+