Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

Effectiveness of Multi-Scale Aggregation for Adversarial Robustness

Bachelor's Thesis

Emerson Leonardo Azevedo Aguiar
eazevedo@ethz.ch

Distributed Computing Group
Computer Engineering and Networks Laboratory
ETH Zürich

Supervisors:
Andreas Plesner, Till Aczel
Prof. Dr. Roger Wattenhofer

August 24, 2025

Acknowledgements

I thank my mother and brother for all their support throughout my studies during my bachelor's degree. I also sincerely thank my supervisors, Andreas Plesner and Till Aczel, for the productive weekly meetings that greatly influenced and supported the direction and quality of this work. Lastly, I would like to thank Prof. Dr. Roger Wattenhofer and the Distributed Computing group for this opportunity.

Abstract

In this work, we replicated the results of the defense strategy proposed by Stanislav et al. [1] and the response paper by Zhang et al. [2]. We then performed a series of our own attacks to stress test the discussed defense and found, in agreement with Zhang et al. [2], that this defense is not robust and is prone to gradient masking. Finally, we visualize the class distribution of the proposed defense architecture, and we show a similar approach that leverages a k-Nearest Neighbor algorithm. Both suggest that a multi-resolution channel-wide stack of the inputs increases adversarial robustness, and the latter shows promising results, significantly outperforming Stanislav et al.[1] approach.

Contents

Chapter 1

Introduction

Adversarial examples have been a challenge to robust vision models for some time [3]. Given an image classifier f, image x, and label c, adversarial examples are defined as a perturbed image $x' = x + \delta$ such that $f(x') \neq c \wedge f(x) = c$. While the classifier fails, humans can still correctly classify the image. The community has resorted to a type of competition where some researchers design various strategies to create strong adversarial ([4], [5], [6], [7]) examples to destroy a neural network's accuracy and others to create defense strategies. So far, the attacking side is winning the exchange, mainly due to gradient-based attacks, which have been shown to be very efficient in creating strong adversarial examples and are hard to mitigate.

The current state-of-the-art defense techniques depend on adversarial training [8] [9], [10], where adversarial examples are used as part of the training data set. This approach is computationally expensive and requires retraining in case of newly seen adversarial examples.

New approaches that achieve similar or better robust accuracy more efficiently would increase the reliability and applicability of current vision models. One of these new approaches is a defense strategy proposed in the paper **'Ensemble Everything Everywhere'**[1].

This technique promises state-of-the-art robustness without the computationally intensive adversarial training used in other methods, giving the community an entirely new approach to defending against attacks. The defense technique works by adjusting the neural network architecture, such that the classification function learned by the model is more similar to the human vision classification function, i.e., how our brains perceive things. The authors also propose a new aggregation function that protects the classification from bias towards attacks that push the likelihood of a target label t from dominating the others.

These approaches are achieved by creating a multi-resolution channel-wise stack of the images, with random perturbations on each resolution, to mimic the human vision model, which sees many versions of an image at different angles, illumination, and positions. Secondly, they use a newly proposed aggregation function, CrossMax, that uses a concept of truthful bidding to normalize the classification logits, thus reducing the effect of skewed values, since these are suspicious and could likely represent an attack. To leverage this new function, they use linear classification heads on top of each layer and use these logits in the CrossMax aggregation function. The backbone model is a ResNet152.

In this work, we will replicate the results of the authors of this paper [1] and also present the results of a related work [2] that already made some evaluations on the robustness of this approach.

This related work claims that the defense suffers from severe gradient-masking by showing how the adversarial accuracy does not hold up against strong adaptive attacks and how the loss landscape of the new model architecture is spiky, thus making gradient-based attacks weaker than expected, giving a false sense of security.

Finally, we will also present a new architecture that leverages the learned features of every layer by creating a feature space of every layer and classifying images via k-Nearest Neighbor. We also do a stress test on the defense technique using the attack approach from [2] but using a ViT/B/16 model as the backbone. The goal is to push the adversarial accuracy down by using a stronger model, such as ViT/B/16, with a transfer attack.

Chapter 2

Related Work

In this chapter, I'll dive into essential topics related to our work to provide all relevant information to understand better what is being discussed and our goal.

First, I'll explain some relevant topics, such as how to create adversarial examples, i.e., what it really means to "attack" a model, which attacks were used, and how they work.

Secondly, I'll explain in detail how the defense strategy proposed by the paper's authors **'Ensemble Everything Everywhere'** [1].

Lastly, I'll present the techniques used in the response paper [2], which analyzes the robustness of the discussed defense strategy.

2.1 Important Topics

2.1.1 Adversarial Attack

Concerning the definition of an adversarial example, as described in the introduction to this work, an adversarial attack is the perturbation that creates the adversarial example. That is, given a neural network $f : X \to C$, some input $x \in C$, and a proper label $c \in C$, an adversarial attack is a perturbation η, such that $f(x + \eta) \neq c$.

We can also see this effect visually on the famous panda image in Fig. 2.1, the paper[10] by Ian Goodfellow et. al.

2.1.2 Types of attacks

Targeted Attacks

Given a neural network $f : X \to C$, some input $x \in C$, label $c \in C$, then a targeted attack is a perturbation η that wants to push the classification of a neural network towards a target class $t \in C$, that is, $f(x) = c \implies f(x + \eta) = t$.

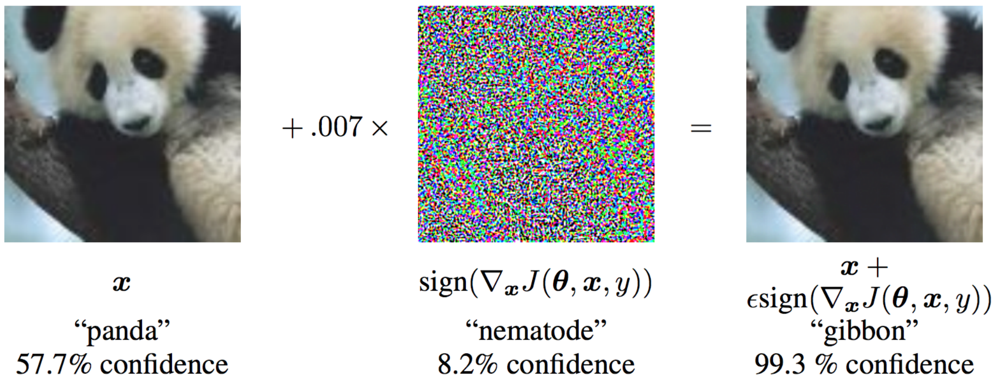


Figure 2.1: Example of adversarial example generation. Achieved by finding the direction of the gradient concerning the loss used to train the model, then moving the pixels in a small ϵ step towards that direction, which will maximize the loss.

The goal is to construct perturbations that force the classifier to misclassify an image towards a specific class. It has more nefarious security implications due to the applications of this attack. You can create perturbations that will make systems classify an image of a person as another, thus bypassing face recognition security measures. A very interesting application of this is shown by Lujo Bauer et al [11] in the Fig. 2.2.

Untargeted Attacks

Untargeted attacks are perturbations η such that $f(x+\eta) \neq c$. The aim is to cause the classifier to fail, without a specific goal in mind.

2.1.3 Threat Model

When designing defense strategies against adversarial examples, the research community developed a framework based on the assumption of the attacker's knowledge of our system. There are two assumptions about the attacker: black-box and white-box. A white-box attack assumes the attacker has full access to the neural network's architecture, weights, inputs, and outputs. A black-box attack assumes that the attacker has limited knowledge of the system and only has access to the inputs and outputs of the neural network. In the paper **Ensemble Everything Everywhere**[1] mentioned in the related works, which this work analyzes, the proposed defense assumes a white-box attack.

This is important because defense strategies strong against white-box attacks are also robust against black-boxed attacks, but the opposite is not true.

Figure 2.2: Example of targeted adversarial attacks. The glasses have a special noise that forces the classification of the top person's face to be classified as the bottom one.

The Threat Model also defines an allowed perturbation budget ϵ and a L_∞ or L_2 bound. That is, for an L_∞-bound, the only adversarial examples allowed are $\{x' : ||x - x'||_\infty \leq \epsilon\}$. This means that the adversarial examples must be in an ϵ-ball around the initial position.

We will be working with L_∞ and $\epsilon = \frac{8}{255}$.

2.1.4 Projected Gradient Descent Attack

The Projected Gradient Descent Attack was introduced as an adversarial attack by Madry et al [8]. It is a commonly used white-box attack and widely accepted as one of the strongest attacks in the literature.

The PGD attack is defined as:

$$x' = \prod_\epsilon (x + \alpha \cdot sign(\nabla L(\theta, f(x), y))$$

where $\prod_\epsilon$ is a projection function back to the ϵ-box around x.

In short, we construct a perturbation $\eta = sign(\nabla L(\theta, x, y))$, which is the direction of the gradient w.r.t the loss function L of a neural network and take a small α step in that direction to maximize the loss and apply this to the image x. The projection works, such that no matter how many iterations, the perturbation η will never be outside the ϵ-box. We can easily visualize this in the Fig. 2.3.

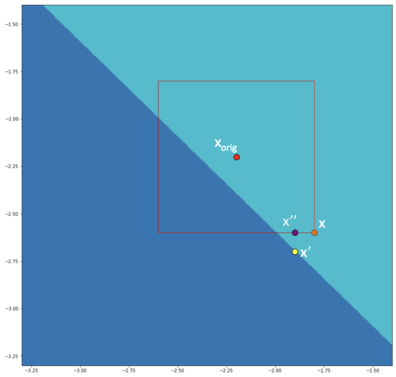


Figure 2.3: Example of a PGD iteration. The regions with different blue shades represent different labels. In this example, they initialize x in a random position inside the ϵ-ball. The PGD iterates to x' which gets projected back to x''.

2.1.5 Fast Gradient Sign Method Attack

The PGD attack can be seen as an iterative version of the FGSM attack with a smaller step projected back to the ϵ-ball. The FGSM attack is defined as:

$$x' = x + \epsilon \cdot sign(\nabla L(\theta, f(x), y))$$

We take a single step of size ϵ in the direction that maximizes the loss.

2.1.6 Auto Projected Gradient Descent Attack

The APGD Attack was introduced in the paper [7] in order to mitigate the main issues of PGD, which often resulted in the attack failing [12]. The main problems of PGD are the following:

- **Fixed step size:** The fixed step size is limiting, since it does not guarantee convergence. This also applies to convex problems with only global minima/maxima.
- **Non-reactive:** The PGD Algorithm is unaware of any trends or knows which direction best improves the attack.

To solve these problems, APGD works by splitting the algorithm into an exploration and an exploitation phase.

APGD: Exploration

Here, the algorithm uses a set of key time steps W and keeps updating the loss function with a gradient descent with momentum:

$$z^{t+1} = \Pi_\epsilon\big(x^t + \eta\,\nabla L(\theta, f(x^t), y)\big)\,,$$

$$x^{t+1} = \Pi_\epsilon\big(x^t + \alpha\,(z^{t+1} - x^t) + (1-\alpha)\,(x^t - x^{t-1})\big)\,.$$

The algorithm keeps track of the position x^t at some time step t that maximizes the loss.

APGD: Exploitation

Once we reach a point x^t, where $t \in W$ and some conditions are met, the algorithm halves the step size $\eta = \eta/2$ to make a more localized search. These exploitation steps can only be performed at times step $t \in W$.

2.1.7 Expectation over Transformation

It has been shown by the community [13] that adversarial examples generated using the frameworks thus discussed don't remain adversarial on transformations such as angle and viewpoint changes. Defense strategies can also use random transformations similar to these to further decrease the effectiveness of attacks.

To address this issue, Athalye et al.[14] introduced a technique called Expectation over Transformation (EOT).

The idea behind EoT, is to sample transformation from a certain transformation distribution T, and maximize in expectation the log probability of some class y, given accurate input x, an adversarial example x', and a transformation used to make the classifier perceive it as the true class $t(x')$ and constrain the distance $d(t(x), t(x'))$ for some metric between the adversarial example and accurate input. Mathematically, we have:

$$\begin{aligned} \underset{x'}{\arg\max} \quad & \mathbb{E}_{t\sim T}\big[\log P(y_t \mid t(x'))\big] \\ \text{subject to} \quad & \mathbb{E}_{t\sim T}\big[d(t(x'), t(x))\big] < \varepsilon, \\ & x \in [0,1]^d. \end{aligned}$$

In a context of a PDG or APGD attack, it is used to properly disturb images with random noise and transformations like rotations, jitter, etc. A number n of EOT iterations is specified, and the algorithm computes n gradients and perturbs the image by the average. This way, noises and transformations are smoothed out, and the adversarial example is stronger.

2.1.8 AutoAttack

The mentioned attacks thus far all worked on classifiers with a Cross Entropy Loss, where if an input x has a correct class y, then we have:

$$CE(x,y) = -log(p_y) = -z_y + log(\sum_{j=1}^{K} e^{z_j})$$

, with $p_i = e^{z_i} / \sum_{i=1}^{K} e^{z_j}$ and z_i are logits.

The main issue with this loss function is that it is not invariant to scaling on the logits. Also, if $p_y \approx 1$ and $p_i \approx 0, i \neq y$, the gradient is arithmetically 0, making it very prone to gradient vanishing.

To overcome this, Fracesco et al. [7] introduce a new loss function called **Difference of Logits Ratio (DLR)**, which is both shift and scale invariant while also having a more stable gradient.

They then propose an attack protocol independent from parameters, except the attack budget ϵ, called AutoAttack. This protocol performs a series of attack strategies on the model and compares the resulting adversarial accuracy per attack with the clean accuracy. The attacks chosen are $APGD_{CE}$, $APGD_{DLR}$, FAB [15] and Square Attack [16].

This protocol is one of the most common strategies to test a new defense technique. This approach has become a key component of the benchmark for adversarial robustness: RobustBench[17].

2.2 Ensemble Everything Everywhere

The paper "Ensemble Everything Everywhere"[1] proposes a new approach to train robust models against adversarial examples. The approach is promising, since it does not rely on data augmentation or adversarial training, thus making it computationally more efficient.

2.2.1 Approach

This new defense strategy is composed of 3 key ideas. The first idea is to create a multi-resolution stack of each image with noise and jitter, such that the model learns features the same way as the mammal visual system. The second idea is to take advantage of the layer decorrelation property shown in their work and use the image classification of each model layer. Lastly, they use a new aggregation function called CrossMax to create an ensemble of the various models or the layers of a single model, i.e., make a self-ensemble.

2.2.2 Multi-Resolution Images

This idea is inspired by how mammals, especially humans, perceive things. When we look at something, we see many images with different distances, angles, color intensities, and due to rapid eye movements, called eye saccades, on various jitter shifts.

Thus, to mimic this behavior, we stack copies of each image in the data set at different resolutions. In the paper, they used the resolutions $N = \{4, 8, 16, 32\}$. The approach is to apply random contrast noise to the image, apply jitter in x and y directions, down-sample it to one of the smaller resolutions, apply noise, then stretch it back to the resolution $R \times R$ required by the used neural network architecture. In the case of a ResNet, $R = 224$. Finally, a final Gaussian noise at the final resolution is applied. Once done, a batch of images of shape $(B, 3, H, W)$ will be $(B, 3N, H, W)$.
In short, for every image x_k in the dataset, we apply the following algorithm:

Algorithm 1 Multi-resolution stack generator

$x_k \leftarrow$ Image from dataset
$N \leftarrow \{4, 8, 16, 32\}$
$x_{stack} \leftarrow \emptyset$
for $r \in N$ **do**
  $x_{new} \leftarrow x_k$
  $x_{new} \leftarrow grayscale\_shift(jitter\_shift(contrast\_shift(x_{new})))$
  $x_{new} \leftarrow UP(DOWN(x_{new}, r) + \mathcal{N}(0, \sigma_1 \cdot I_{r\times r}), 224) + \mathcal{N}(0, \sigma_2 \cdot I_{224\times 224})$
  $x_{stack}.append(x_{new})$
end for
return x_{stack}

The up- and down-sampling were done with bicubic interpolation. The jitter was randomly sampled in the $[-3, 3]$ range, and the Gaussian noise had a standard deviation of 0.2 out of 1.0. The result can be seen in Figure 2.4.

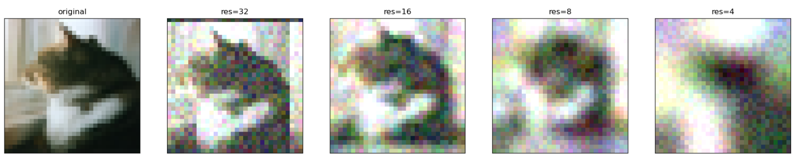


Figure 2.4: Multi-resolution of cat image from the CIFAR-10 Dataset

2.2.3 Layer De-correlation

As an image passes through an image classifier neural network, the features extracted by each layer differ. The earlier layers learn to see simple features such as lines, points, blobs, and the later ones more complex features such as textures, noses of dogs, shells of turtles, etc.

The authors attempted to see if an attack was created to fool the classifier at some layer β, would it also fool other layers α.

They show that there is a three-way split, such that an attack on some middle layer would affect the classification of the nearby middle layers and not affect the earlier and later layers.

This property was named Layer De-correlation. Their idea was to use this property to create an ensemble of the logits of all layers to create a robust defense strategy against an attack on some random layer. This approach is even more efficient since other model ensemble approaches mainly combine the logits of many different models. In Figure 2.5 we visualize this decorrelation.

2.2.4 Model Ensemble via CrossMax

Model Ensembles is a method of creating a more robust classification of images by combining the output logits of many models. This is commonly achieved by taking the mean of the logit values.

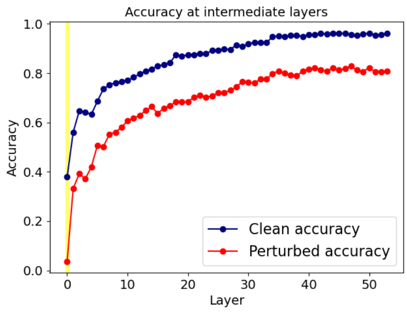
In the paper, the authors proposed a new aggregation function called CrossMax. The idea is to take advantage of the concept of truthful bidding. Truthful Bidding stems from Game Theory, where the k-th highest bidder wins instead of the highest. The idea is that if players lie and bid high to win the game, they will eventually lose, because by bidding more, they will often bid higher than the k-th highest bidder. Analogically, lying and bidding less will result in the same effect. Thus, the best winning strategy is to bid truthfully.

The analogy in our setup is that if an attack greatly increases the probability of a targeted attack towards some class t, then our aggregation function will ensure that it will not win. This also accounts for cases where an attack pushes the truth label c too low.

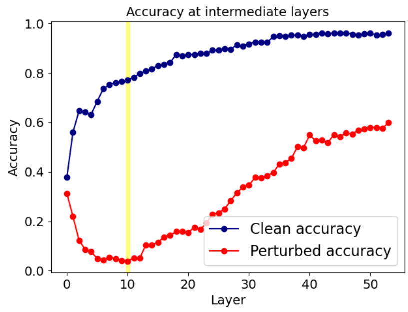
The CrossMax function works as follows:

The first line prevents a targeted attack on a specific predictor from dominating. The second line prevents adversarial examples that significantly increase the prediction value of some target class t from dominating. Lastly, they use a truthful bidding strategy by picking the k-th largest value.

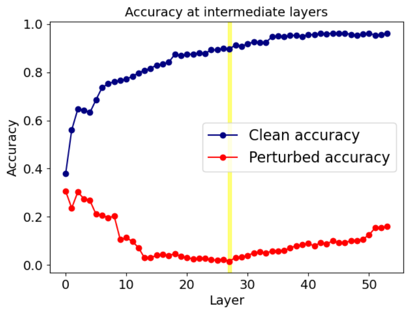
In their paper, the authors used $k = 3$.



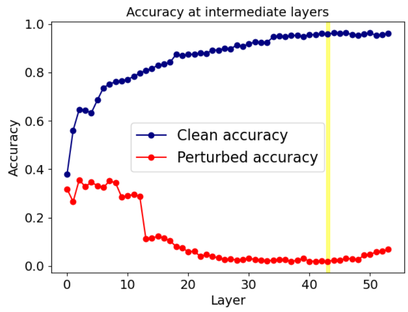
(a) Attack at layer 0



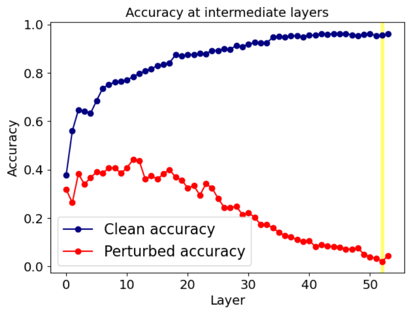
(b) Attack at layer 10



(c) Attack at layer 27



(d) Attack at layer 43



(e) Attack at layer 52

Figure 2.5: The accuracy on the clean data for each layer β in blue and the accuracy on the attacked images created by attacking the layers $\alpha \in \{0, 10, 27, 43, 52\}$ in red

2.3 Response Paper

The original goal of this work was to replicate the results of the "Ensemble Everything Everywhere" paper and test its robustness. However, in the meantime, the paper "Evaluating the Robustness of the 'Ensemble Everything Everywhere' Defense [2] was published, and it did precisely that. Thus, the direction of this work changed, and it is, in a sense, a continuation of this paper. More details

Algorithm 2 CrossMax(Z)

Require: Logits Z of shape $[B, N, C]$, where B is the batch size, N the number of predictors and C the number of classes
  $Z \leftarrow Z - max(Z, axis = 2)$ ▷ Subtract max-predictor over classes
  $Z \leftarrow Z - max(Z, axis = 1)$ ▷ Subtract max-class over predictors
  $Z \leftarrow sort(Z, axis = 1)$ ▷ For each class, sort the prediction scores over the predictors
  return $Z[:, k]$ ▷ For each class, return the k-th highest prediction score

will be explained in the next chapter.

The response paper shows that the approach suffers from severe gradient masking. Due to this problem, the AutoAttack wasn't strong enough to support their accuracy claims.

The authors of the response paper analyzed their defense strategy using the following steps.

Firstly, they plotted the loss landscape of the multi-resolution self-ensemble model to check for the feasibility of strong gradient-based optimization attacks.

Secondly, they reset the final layer of the model and retrained it such that for every input x, there is an easy-to-find nearby adversarial example x' with a different classification.

Finally, they designed a series of personalized attacks using existing techniques to push the robustness of the model down to 11% for the CIFAR-10 Dataset.

2.3.1 Gradient Masking

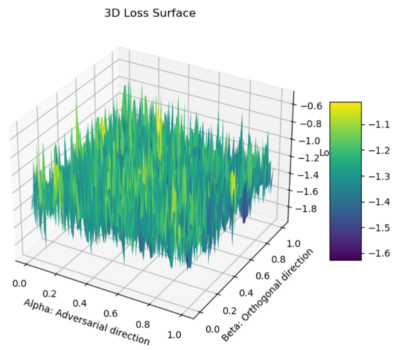
Gradient masking is a defense technique that masks the gradients of the model with noise and non-differentiable operations or makes the gradient less sensitive to small perturbations.

Thus, when an attacker attempts to create adversarial examples using the model's gradients, it will not be strong enough, and the model is robust.

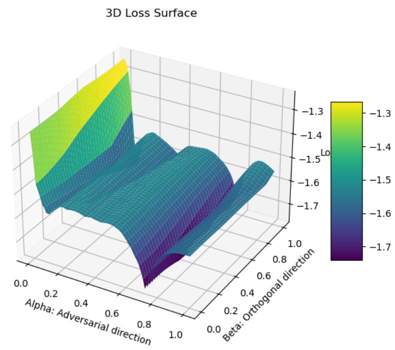
However, this property grants a false sense of security. Other attacks that don't use gradient information or adversarial examples created using another model, i.e., transfer attacks, are still effective. Thus, Gradient Masking often just masks vulnerability instead of fixing it.

2.3.2 Loss Landscape

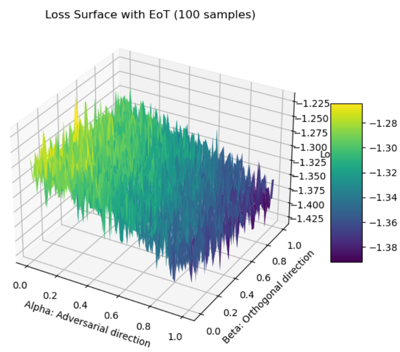
The approach to plotting the landscape is, for a given input x, to pick two directions d_1, d_2, where d_1 points towards an adversarial direction and d_2 is orthogonal to d_1. Then for $\alpha, \beta \in [0, 1]$ plot the log probabilities of the models loss on $x + \alpha \cdot d_1 + \beta \cdot d_2$. The shape of the loss landscape gives us a hint of how susceptible a model is to a gradient-based attack. In Fig. 2.6, we can see how the loss landscape is very spiky, greatly hinting towards a gradient masking issue.



(a) Loss Landscape with multi-resolution inputs with random perturbation



(b) Loss Landscape with multi-resolution inputs without random perturbation



(c) Loss Landscape with multi-resolution inputs with random perturbations smoothed with Expectation over Transformation iterations

Figure 2.6: The Loss Landscape is very spiky, which greatly suggests a gradient masking issue

2.3.3 Nearby Adversarial Example

Here, the authors of [2] suggest testing the strength of the performed RobustBench AutoAttack to see if it was correctly implemented. This technique was presented by Zimmerman et al. [18], and it works as follows.

First, reset the final classification layer to some random weights. Then, finetune it again such that for every input x, there is a nearby adversarial example x' with a different classification. Thus, an attack on a model with this property should have a very strong attack; however, they have seen that the model trained with the proposed defense strategy only had a 60%attack success rate. This shows that either the proposed attack techniques were unsuitable for showing their robustness or that, due to gradient masking, the model cannot correctly perturb the images.

Chapter 3

Background

In this chapter, I'll go over in detail my approach to replicating the results of both related works and an attempt I made to try to push the accuracy of the defense strategy even lower.

I'll explain my approach to fine-tuning the models and training the linear heads. Also, I'll mention how I produced the adversarial examples and how they were used in the pipeline to construct the results.

Finally, I'll explain a test that hasn't been attempted in the original papers, which is to do a transfer attack with a ViT Backbone to the Multi-Resolution Self-Ensemble defense, in the hopes of seeing if there is any inherent robustness to their approach or if it can be completely destroyed.

3.1 Finetuning and training

The defense strategy works by finetuning the backbone with multi-resolution inputs towards a chosen dataset, such as CIFAR-10 or CIFAR-100, and training the linear heads on every layer with a frozen backbone. See Fig 3.1.

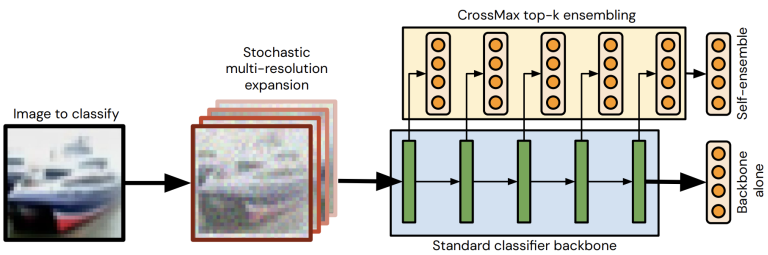


Figure 3.1: Model architecture proposed. The backbone can be any vision architecture, such as ResNet152 or ViT 16.

In Stanislav et al.[1], the backbone is fine-tuned for 10 epochs with $\eta = 3.3 \times 10^{-5}$ and three epochs at $\eta = 3.3 \times 10^{-6}$, and the linear heads are trained for one epoch.

In their paper, the authors have shown that adversarial accuracy remains high only with small learning rates. Training the linear heads for too many epochs will also reduce adversarial accuracy, as shown in Fig. 3.2.

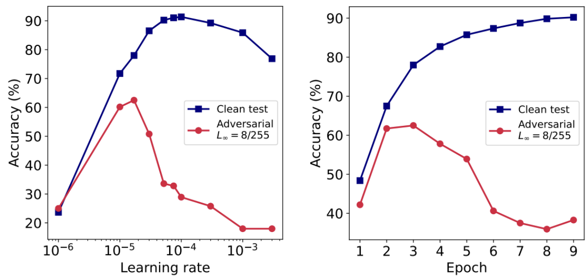


Figure 3.2: Plot from Stanislav et al. [1]. On the left plot, we can see that η should be around 10^{-5}. The right plot should show the number of epochs when training the linear heads with $\eta = 1.7 \times 10^{-5}$. The sweet spot is about 3.

My approach for this was to split the finetuning and training into different pipelines, depending on which backbone architecture or if the multi-resolution inputs were needed. Thus, I created the following pipelines:

- Backbone Pipeline
- Linear Heads Pipeline

The learning rate used in all the fine-tuning and training is the same as used in Stanislav Fort et al. [1], which is $\eta = 3.3 \times 10^{-5}$ and $\eta = 3.3 \times 10^{-6}$.

3.1.1 Backbone Pipeline

In this pipeline, the approach is to finetune the backbone model, pretrained on IMAGENET, to CIFAR-10. Our method differs from Stanislav Fort et al. [1] in that, instead of directly fine-tuning the full model to CIFAR-10, we do different stages of training.

- Classifier Layer
- Multi Resolution Input Layer
- Full Model

The idea was to show that a pretrained model can quickly adapt to the new dataset and the multi-resolution stack without requiring complete retraining.

Classifier Layer

A pre-trained model on IMAGENET is a classifier $f : X \to C$, where C holds 1000 classes. Since CIFAR-10 works on C', where C' holds 10 classes, we need to replace the last linear layer with one that outputs the correct number of labels.

Since the new linear layer will be initialized with random weights, we fine-tune it for 10 epochs with $\eta = 3.3 \times 10^{-5}$. We can reach around 80% accuracy. See Fig. 3.4.

Multi Resolution Input Layer

To handle multi-resolution inputs in a memory-efficient way, we create it during training using a function in the model. We also have to replace the first convolution layer. For an input x with the shape (B, C, H, W), the first convolution layer expects a channel of size C. After creating the multi-resolution image stack, we have a shape of $(B, N * C, H, W)$; thus, our new input layer must expect a channel of this size.

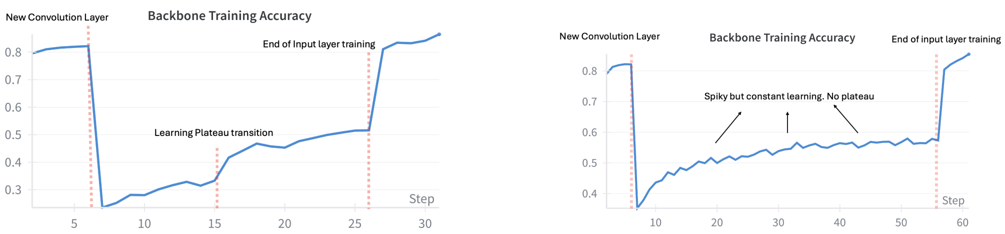
The new input layer will have random weights; thus, it requires training. We proceed by freezing the whole backbone except for the input layer and retraining it. The goal is to train the input layer until we can achieve the same accuracy levels on fine-tuning the classifier layer, such that the backbone can then correctly associate the new inputs with the previously learned unperturbed images.

This proved problematic with low learning rates since the new image stack is constructed with added perturbations (jitter, contrasts, white noise), and the learning only improves with both significant epochs and learning rates. According to Stanislav et al. [1], significant epochs and learning rates are discouraged, since they reduce adversarial robustness.

After extensive trials, we have seen that this goal can be achieved with many epochs and varying learning rates. Still, we decided to run 10 epochs, so that the accuracy reaches around 45% and let the rest of the training be performed on the complete training, since there was not much to gain by pushing the accuracy back to the levels of the classifier layer fine-tuning in this stage. See Fig. 3.3 for a better visualization.

Full Model

At this stage, we perform a similar approach as described in the paper [1]. We unfreeze all the weights and train the whole backbone with $\eta = 3.3 \times 10^{-5}$ for 10 epochs and train for another five epochs, instead of 3, with learning rate $\eta = 3.3 \times 10^{-6}$. The result can be seen in the Fig. 3.4, drawn from our training logs in Weights and Biases.



(a) Input Layer Fine-Tuning, 20 epochs, $\eta = 0.1$, Accuracy Peak: 51%

(b) Input Layer Fine-Tuning, 50 epochs, $\eta = 0.003$, Accuracy Peak: 58%

Figure 3.3: In the left image, with a greater learning rate, we can see some regions of learning plateau and faster learning. In the right image, we can see a smoother transition from the plateaus, but jagged learning. A proper approach would most likely be to use a higher learning rate until the $\approx 50\%$ region and proceed with a smaller learning rate.

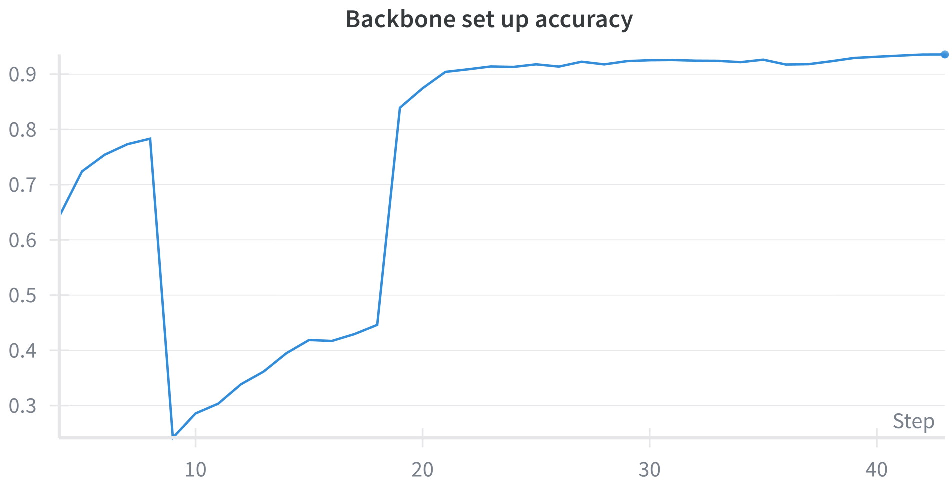


Figure 3.4: Backbone fine-tuning of a Multi Resolution ResNet152 to CIFAR-10. We achieve high accuracy quickly by only training the classification layer. It drops fast after changing the input layer, and we train for 10 epochs to stabilize it before doing a full model training.

3.1.2 Linear Heads Pipeline

Our approach to training the linear heads differs slightly from Stanislav et al. [1]. The authors create wrapper models f_i to extract the logits of a specific layer's head and track only this linear head gradient. They do this for a set of specified layers they want to train.

Our approach is that we extract a list $L = [f_1, f_2, ..., f_n]$ and compute a sum of all the losses $L_{all} = \sum_i^n L(\theta, f_i(x), y)$ and then for each layer j we do a backpropagation on $\frac{\partial L_{all}}{\partial f_j}$. This will make the training much more efficient since we do only one full pass through the model, and for each epoch, we train the linear layers. The backpropagation for the linear head of each layer will also be correct since $\forall j \frac{\partial L_{all}}{\partial f_j} \implies f_i = 0 \ \wedge \ i \neq j$, thus the training of each linear head is effectively isolated from the rest.

3.2 Replicating Results

The first goal of this work is to replicate the results of related works for consistency. Once that is achieved, we will try some other attacking approaches and aggregation techniques and evaluate. For the replication, we split the tasks into the following pipelines:

- Layer De-correlation Pipeline
- Multi Resolution Pipeline
- Attack Pipeline

3.2.1 Layer De-correlation Pipeline

In this pipeline, we first perform the backbone training described above, except that we skip the input layer stage, since we don't use multi-resolution inputs. We fine-tune a ResNet152 to the CIFAR-10 Dataset and use it to attack a set of layers, because we want to show a property of the model without any relation to the type of inputs.

Attacking Layers

Attacking a layer means that we want to create adversarial examples designed to fool the classification accuracy of a specific layer α in our model. The approach here was to create a wrapper for our model, that given a model with n-layers, linear heads h_i for layer i, g_i output of layer i and input x, the wrapper computes $f_i(x) = h_i \circ g_i \circ ... \circ g_1(x)$ for some $\alpha = i$.

Using this logit, we use a selective PGD algorithm that only perturbs correctly predicted images and ignores already wrongly predicted images. We run PGD with $\epsilon = \frac{8}{255}, \alpha = \frac{2}{255}$, 10 steps and using a random position in the ϵ-ball as the starting point of our perturbation iteration.

Finally, we then compute the model f accuracy on these images, perturbed and non-perturbed, and the results are used to plot the red curve on Fig. 2.5.

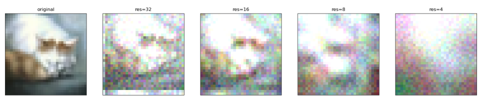
The adversarial example generation is done per-batch on the fly to be memory efficient. In our experiment, we attack the layers $\alpha \in \{0, 10, 27, 43, 52\}$.

3.2.2 Multi Resolution Pipeline

In this pipeline, we train the backbone exactly as described above, then we train the linear heads on top of the frozen backbone for three epochs with $\eta = 3.3 \times 10^{-5}$.

The next step is to plot the loss landscape of our model as described above. The normalized adversarial direction d_1, d_2 are computed as $d_1 = \frac{\nabla L(\theta, f(x), y)}{||\nabla L(\theta, f(x), y)||}$ and let d_r be a random vector, $d_2' = d_r - (d_r \cdot d_1) \cdot d_1 \implies d_2 = \frac{d_2'}{||d_2'||}$. Then, for α, β, we pick 80 evenly spaced values in $[0, 1]$. Finally, we create the three loss landscapes by plotting $\log(f_{CROSSMAX}(\text{Multi-Resolution}(x) + \alpha_i \cdot d_1 + \beta_j \cdot d_2)) \forall\, i, j$.

Important to mention that the second plot creates a deterministic multi-resolution stack (See Fig. 3.5) to smooth out the loss landscape, and the third plot uses the conventional multi-resolution stack. Still, it uses 100 EoT iterations to smooth out the randomness.



(a) Multi Resolution Stack of a cat image from CIFAR-10 Dataset with perturbation

(b) Multi Resolution Stack of a cat image from CIFAR-10 Dataset without perturbations

Figure 3.5: Image (a) is the original multi-resolution stack used in the first plot. Image (b) used in the smooth plot has no jitter, contrast, or white noise.

Finally, we run the attack pipeline as described in the response paper [2].

3.2.3 Attack Pipeline

In the attack pipeline, however, we perform an AutoAttack benchmark and a series of adaptive attacks as described in Zhang et al. [2], with fewer steps. We thought that seeing a similar trend would be enough and that there was not much to gain by getting the same results.

The AutoAttack is performed with the random restarts flag on and performs, as described above, a series of four strong attacks: $APGD_{CE}$, $APGD_{DLR}$, FAB, and Square Attack.

The adaptive attacks are separated into two bundles.

First Bundle

The first bundle consists of the following steps:

- **PGD Attack:** We perform a PGD attack with 10 steps.
- **Transfer Attack:** We perform a transfer attack from a model using a mean aggregation ensemble to test the effectiveness of the CrossMax aggregation. The transfer attack is also done with PGD using 10 steps.
- **EoT Attack:** We perform a PGD with 10 steps and 2 EOT iterations to our transfer attack, to make up for the randomness in our self-ensemble model.
- **Bag of Tricks:** Finally, we perform the Bag of Tricks method, explained in [2]. The BoT performs a transfer attack with a model using mean aggregation using PGD with 10 steps and 10 EOT steps, and also attacking the Hinge Loss, rather than the Cross Entropy Loss.

Second Bundle

The second bundle consists of the following steps:

- **APGD Attack:** We perform an APGD attack with 10 steps and 10 EOT iterations using a transfer model with mean aggregation.
- **Larger Initial Radius Attack:** We update the APGD to start with $\eta = 3\epsilon$ to find promising adversarial directions. We perform $\eta = 3\epsilon$ for 30%, $\eta = 2\epsilon$ for 30% and $\eta = \epsilon$ for 40% of iterations.
- **Extra steps:** We add extra steps to the last attack.

These attacks were run by Zhang et al.[2] for 10 iterations, and they presented the mean and variance. We experimented with 2 and 10 iterations; the results were similar. Thus, we decided to proceed using 2 iterations only.

Attack Pipeline Results		
Attack	Accuracy	Variance
None	62.11%	$\pm 7.47 \times 10^{-4}$
AutoAttack	48.05%	$\pm 7.47 \times 10^{-4}$
PGD	58.98%	$\pm 1.37 \times 10^{-4}$
+ Transfer	32.81%	± 0
+ EOT	11.72%	$\pm 2.44 \times 10^{-4}$
+ Bag of Tricks	3.90%	2.44×10^{-4}
APGD + Transfer + EOT	40.63%	6.10×10^{-5}
+ Larger initial radius	24.61%	3.81×10^{-4}
+ extra steps	19.53%	2.44×10^{-4}

Table 3.1: Results of the AutoAttack and the adaptive attacks to our multi-resolution self-ensemble.

3.3 Results

Performing the abovementioned pipelines, we can replicate the values shown in Table 3.1.

Looking at these results, we can make the following observations. The defense strategy shows promising results by looking at the weak effect of AutoAttack and PGD with a few steps. However, as soon as we work with transfer attacks and attacks on a different loss function, we see a significant drop in adversarial accuracy. This shows how the CrossMax function masked the gradients, thus making the loss landscape very spiky. This spiky loss landscape prevents gradient-based attacks from correctly moving towards good adversarial directions inside the allowed $\epsilon = \frac{8}{255}$ ball.

Also important to notice is that even with many compounded attacks, for example, Transfer + EOT + BoT, the adversarial accuracy never reached 0%, thus indicating some sort of inherent robustness in the defense. For this, further investigation through our stress tests will be shown.

Since the results from [2] had a clean accuracy of 88.9%, we ran a test, where the linear heads were trained with $\eta = 3.3 \times 10^{-5}$ for five epochs and $\eta = 3.3 \times 10^{-6}$ for another five epochs. The results for comparison can be found in Table 3.2.

In the paper [1], the authors mention that too many training epochs for the linear heads lead to reduced adversarial accuracy. This can be seen in how the transfer attack reduced the accuracy by $\sim 30\%$ on our first attempt and $\sim 50\%$ on our second attempt.

Our goal with the second attempt was to try to push the clean accuracy higher, which we succeeded in doing, but at the cost of adversarial accuracy. In all metrics in both Table 3.1 and Table 3.2, the drop in accuracy is larger, for

Attack Pipeline Results		
Attack	Accuracy	Variance
None	73.83%	$\pm 7.47 \times 10^{-4}$
AutoAttack	53.13%	$\pm 5.49 \times 10^{-4}$
PGD	62.11%	$\pm 1.37 \times 10^{-4}$
+ Transfer	27.34%	$\pm 5.49 \times 10^{-4}$
+ EOT	11.72%	± 0
+ Bag of Tricks	5.07%	1.52×10^{-5}
APGD + Transfer + EOT	35.55%	1.52×10^{-5}
+ Larger initial radius	25.00%	2.44×10^{-4}
+ extra steps	22.66%	0

Table 3.2: Results of the AutoAttack and the adaptive attacks to our multi-resolution self-ensemble, with more training on the linear heads.

example, attack success from clean to APGD + Transfer + EOT in the first attempt was $\sim 22\%$ and $\sim 40\%$ in the second.

Thus, we conclude that Stanislav et al. [1] are correct that the number of training epochs and learning rates must be carefully picked. The results suggest that their defense strategy is not robust against carefully designed gradient-based attacks.

CHAPTER 4

Methodologies

In this chapter, we will present novel approaches to stress testing and analyze the proposed defense strategy [1]. The novel approaches are the following pipelines:

- Vision Transformer Pipeline
- k-Nearest Neighbor Pipeline
- Stress Test
- t-SNE Plot

4.1 Our contribution

Our contribution to the analysis of **Ensemble Everything Everywhere** [1] is to stress test the defense to see, if any inherent robustness is present or if it can be completely destroyed. We also test a new approach on utilizing the layers logits via a k-Nearest Neighbor approach and see how it compares in terms of robustness. We also test how a ViT backbone compares to a ResNet152. Finally, we plot the class distributions via t-SNE plots of the CIFAR-10 datasets on our different models in the pipeline and analyze the results.

4.1.1 Vision Transformer Pipeline

Introduced by Alexey et al. [19], the Vision Transformer Neural Network architecture (See Fig. 4.1) has proven to be a very strong approach for image classification. The strongest vision models at the moment [20], [21] [22] are ViT-based. A downside of ViT is that it requires much more data to train to high accuracy than an approach with CNN.

Thus, for finetuning the backbone to CIFAR-10, we run the backbone pipeline with epochs $e = [5, 10, 20, 5]$ for each stage, respectively, where the last 2 epochs are for a complete model training with two different learning rates. This decision was met, due to the larger complexity of ViT and the model not learning much further after 5 epochs using only the final layer or the new input layer for multi-resolution inputs.

Although Stanislav et al. [1] show that for linear heads training, larger epochs will result in less adversarial robustness, we decided to train linear heads for 5 epochs on $\eta = 3.3 \times 10^{-5}$ and 5 epochs on $\eta = 10^{-6}$. If the linear heads themselves don't achieve proper clean accuracy, the adversarial robustness will automatically be low.

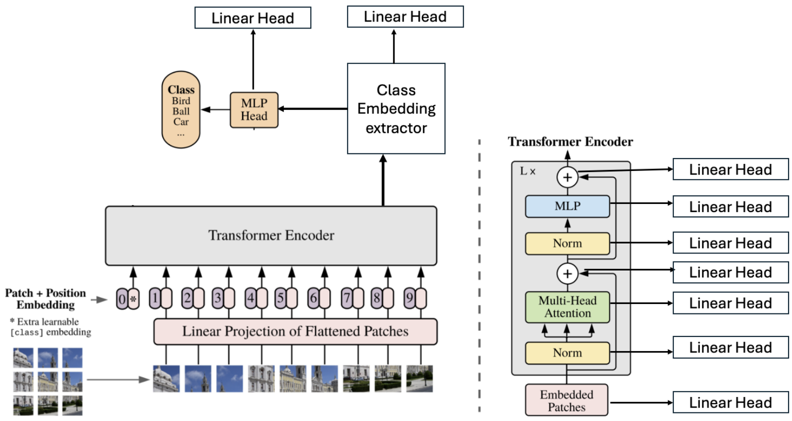


Figure 4.1: Architecture of the ViT Model. Base image taken from Alexey et al. [19] and edited by us.

4.1.2 K-Nearest Neighbor Pipeline

Combining classical machine learning algorithms with neural networks in order to achieve robustness against attacks is not a novel idea[23]. Specifically, using layer features in combination with a k-Nearest Neighbor search has also been attempted before by Nicolas et al. [24], where the authors gather the output of each layers k-NN classifier and compute a prediction, confidence and credibility values, which they use to detect if a prediction was correct or if the model has found an adversarial example. They also show how their model outperforms a Dense Neural Network on a series of attacks on various datasets on adversarial robustness.

Our k-Nearest Neighbor approach differs from the original proposed method in that it does not use linear heads on top of a layer, thus only the backbone needs to be finetuned.

Instead, each layer, separately, gathers the features of the entire CIFAR-10 Dataset and creates a feature space together with the labels. For inference, a given input x extracts the features h_i of some layer i and performs a k-Nearest

Neighbor search on the feature space with h_i. Thus, the classifier for layer i, is:

$$f_i = KNN_i(g_i \circ ... \circ g_1(x)) = c$$

, where KNN_i is a k-NN performed on a feature space of layer i.

The backbone model used in this pipeline is a ResNet152, fine-tuned as described in the backbone pipeline. A diagram of this architecture is shown in Fig. 4.2

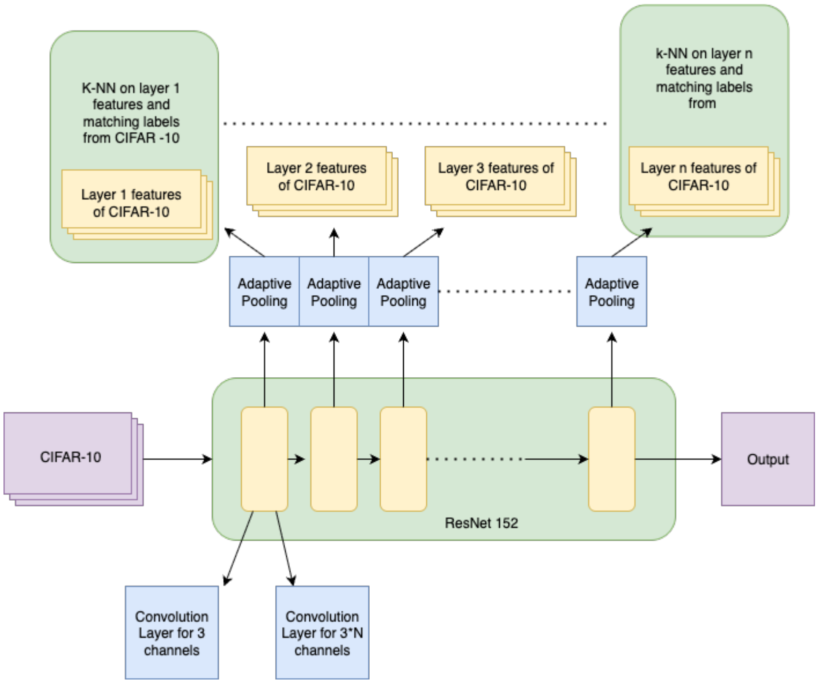


Figure 4.2: Our k-NN architecture. For inference, we pass some test image x into the backbone and perform a k-NN search on the KNN_i for each layer i. The output of each layer will be the classification.

4.1.3 Stress Test

Our proposed robustness stress test for the defense strategy discussed is divided into the following steps:

- We perform the attack pipeline on a Multi Resolution ResNet152 Self-Ensemble using a trained Multi Resolution ViT/B/16 Self-Ensemble as the backbone for the transfer attacks.
- We perform the attack pipeline on a Multi Resolution ViT/B/16 Self-Ensemble using a trained Multi Resolution ViT 16 Self-Ensemble as the backbone for the transfer attacks.

- We perform the attack pipeline on a Multi Resolution ViT/B/16 Self-Ensemble using a trained Multi Resolution ResNet152 Self-Ensemble for the transfer attacks.
- We compare the clean and adversarial accuracy of the k-NN model with a Multi Resolution ResNet152 or ResNet152 backbone on adversarial examples created using PGD with 100 steps using the attack models: ResNet152, ResNet152 Self-Ensemble, Multi Resolution ResNet152, and Multi Resolution ResNet152 Self-Ensemble. Since the models that work on multi-resolution inputs can be random due to the random perturbations, we apply 10 steps of EOT.

4.1.4 t-SNE Plot

The t-SNE Plot is a popular approach for plotting high-dimensional data for visualization. We use this plot to see how the different combinations of models and techniques influence how close or spread apart the class distribution is.

If the classes are more spread apart, then a PGD attack constrained to a ϵ-ball would theoretically be unable to find adversarial examples. This would also not suggest any gradient masking, since even on a flat loss landscape, the gradient can be optimized freely, and an adversarial class would be too far away.

By plotting the class distributions, we can see which component of the defense strategy indicates some inherent robustness to the model or none.

4.1.5 Stress Test Results

Vision Transformer vs ResNet152 Attack Pipeline

The Vision Transformer backbone used in these tests was the ViT/B/16 model, which was fine-tuned as described in the Vision Transformer Pipeline. The Multi-Resolution ViT/B/16 reached $\sim 73\%$ accuracy on clean CIFAR-10 Data. The results for each stress test combination are displayed on Tables 4.1, 4.2, 4.3, and 4.4. We will compare these values with the more robust results from Table 3.1.

We see that a ViT backbone is more robust against the AutoAttack suite, giving us $\sim 7\%$ attack success rate, compared to $\sim 14\%$ with a ResNet152.

With a ViT backbone, we have a PGD attack success rate of $\sim 9\%$ using a Multi-Resolution ViT Self Ensemble to attack 4.1, and $\sim 2\%$ using a Multi-Resolution ResNet152 Self-Ensemble to attack 4.2. Analogically, for APGD, we see an attack success rate of $\sim 26\%$ and $\sim 22\%$ by attacking with Multi-Resolution ViT Self Ensemble and Multi-Resolution ResNet152 Self Ensemble, respectively.

Multi-Res. ViT Self Ensemble vs Multi-Res. ViT Self Ensemble		
Attack	Accuracy	Variance
None	59.77%	$\pm 1.37 \times 10^{-4}$
AutoAttack	52.34%	$\pm 2.44 \times 10^{-4}$
PGD	50.39%	$\pm 1.52 \times 10^{-5}$
+ Transfer	31.25%	$\pm 6.10 \times 10^{-5}$
+ EOT	19.53%	$\pm 6.10 \times 10^{-5}$
+ Bag of Tricks	9.76%	$\pm 1.52 \times 10^{-5}$
APGD + Transfer + EOT	33.2%	1.37×10^{-4}
+ Larger initial radius	25.00%	6.10×10^{-5}
+ extra steps	24.22%	2.44×10^{-4}

Table 4.1: Attack pipeline on Multi-Resolution ViT Self Ensemble attacked by Multi-Resolution ViT Self Ensemble

Multi-Res. ViT Self Ensemble vs Multi-Res. ResNet152 Self Ensemble		
Attack	Accuracy	Variance
None	57.42%	$\pm 1.52 \times 10^{-5}$
AutoAttack	49.22%	$\pm 5.49 \times 10^{-4}$
PGD	55.86%	$\pm 1.84 \times 10^{-3}$
+ Transfer	28.91%	$\pm 6.10 \times 10^{-5}$
+ EOT	8.59%	$\pm 2.44 \times 10^{-4}$
+ Bag of Tricks	2.34%	± 0
APGD + Transfer + EOT	35.55%	1.37×10^{-4}
+ Larger initial radius	19.92%	3.81×10^{-4}
+ extra steps	22.27%	3.81×10^{-4}

Table 4.2: Attack pipeline on Multi-Resolution ViT Self Ensemble attacked by Multi-Resolution ResNet152 Self Ensemble

Thus, we conclude that a ViT backbone can produce stronger gradient-based attacks and is more robust against attacks created by weaker models. This is further supported by the fact that, as a result of the second bundle of attacks, ViT attacking ResNet152 has an attack success rate of $\sim 40\%$ and the opposite 30%.

A very interesting finding, however, is when we use Multi-Resolution ResNet152 Self Ensemble defense [1] using a normal ViT without a Multi-Resolution input as the transfer model to attack with. Table 4.4 shows that the defense is very ineffective. With adaptive PGD and APGD attacks with only a few steps, i.e., 10 steps, it is enough to reach 0% accuracy.

Multi-Res. ResNet152 Self Ensemble vs Multi-Res. ViT Self Ensemble		
Attack	Accuracy	Variance
None	64.84%	$\pm 5.49 \times 10^{-4}$
AutoAttack	55.86%	$\pm 7.47 \times 10^{-4}$
PGD	50.78%	$\pm 1.52 \times 10^{-3}$
+ Transfer	30.47%	$\pm 6.10 \times 10^{-5}$
+ EOT	18.36%	$\pm 7.47 \times 10^{-4}$
+ Bag of Tricks	8.98%	$\pm 1.37 \times 10^{4}$
APGD + Transfer + EOT	36.33%	3.81×10^{-4}
+ Larger initial radius	28.13%	5.49×10^{-4}
+ extra steps	24.22%	6.10×10^{-5}

Table 4.3: Attack pipeline on Multi Resolution ResNet152 Self-Ensemble attacked by Multi-Resolution ViT Self-Ensemble

Multi-Res. ResNet152 Self Ensemble vs ViT Normal		
Attack	Accuracy	Variance
None	70.31%	$\pm 2.44 \times 10^{-4}$
AutoAttack	60.55%	$\pm 1.37 \times 10^{-4}$
PGD	1.56%	± 0
+ Transfer	0%	± 0
+ EOT	0%	± 0
+ Bag of Tricks	0%	± 0
APGD + Transfer + EOT	1.56%	0
+ Larger initial radius	0%	0
+ extra steps	0%	0

Table 4.4: Attack pipeline on ResNet152 attacked by ViT Normal

k-Nearest Neighbor Results

For the above-mentioned series of attacks, we plot the mean of the last five-layer predictions of our k-NN model on clean and perturbed data, and also the accuracy of a ResNet152 Self Ensemble and a Multi-Resolution Self Ensemble on perturbed data for comparison. The goal is to see how the k-NN approach robustness compares to the proposed defense strategies by Stanislav et al. [1]. In the Fig. 4.3 and Fig. 4.4, we have for the k-NN model a normal ResNet152 backbone and a Multi-Resolution ResNet152 backbone, respectively.

Our k-NN approach shows promising results. The standard ResNet152 backbone retains high clean accuracy and outperforms the other approaches whilst retaining $20\% - 40\%$ adversarial accuracy. With the Multi-Resolution ResNet152 backbone approach, we lose clean accuracy and go down to $\sim 70\%$. However, the upside is that the model also maintains $\sim 70\%$ adversarial accuracy, significantly outperforming Stanislav et al. [1] approach.

Class Distributions

For the class distribution, we used **scikit's TSNE** plot for a ResNet152, Multi Resolution ResNet152, and Multi Resolution ResNet152 Self Ensemble, Fig. 4.5.

According to our k-NN tests and Stanislav et al. [1], introducing multi-resolution inputs increases the robustness of the neural network. We can interpret this in the plot as the class distribution on image (b) is more clustered among themselves, thus farther away from other classes. Thus, this makes a PGD attack less effective.

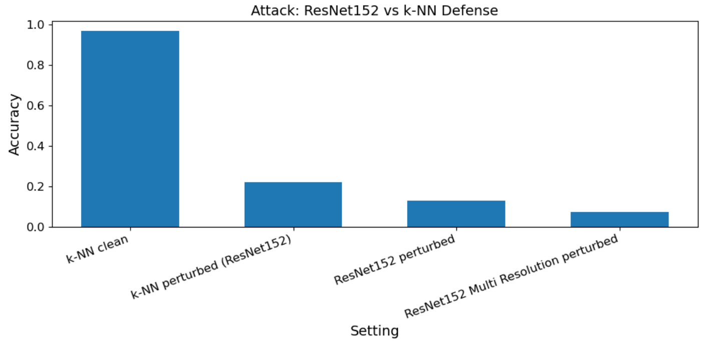
The introduction of the CrossMax aggregation considerably spreads the class distribution, as seen in image (c). This suggests that it makes the model less robust to adversarial examples. The fact that it, however, increases robustness and that the loss landscape is very spiky, powerfully suggests that it was, in fact, gradient masking. Thus, gradient-based attacks like PGD or APGD won't produce a strong attack, giving a false sense of security.

4.2 Conclusion

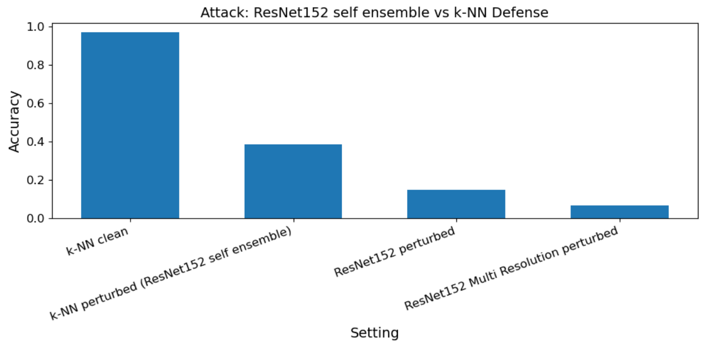
As our results suggest, we conclude that the approach proposed by Stanislav et al. [1] is not robust, since the CrossMax aggregation suffers from severe gradient masking, as we can show in our attack pipeline results and on the weaker gradient-based adversarial examples transferred to our k-NN model.

Our t-SNE plot and our k-NN model robustness suggest that the multi-resolution stack of the input images provides some robustness. How extensive this is left open and can be investigated further in another work.

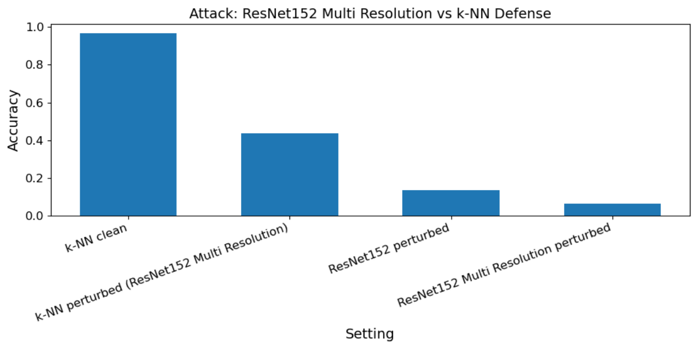
Another promising find is how strong the k-NN approach seems to be. A more detailed dissection of this method using various backbone models and stress testing its approach with carefully designed adaptive attacks can be investigated further in another work.



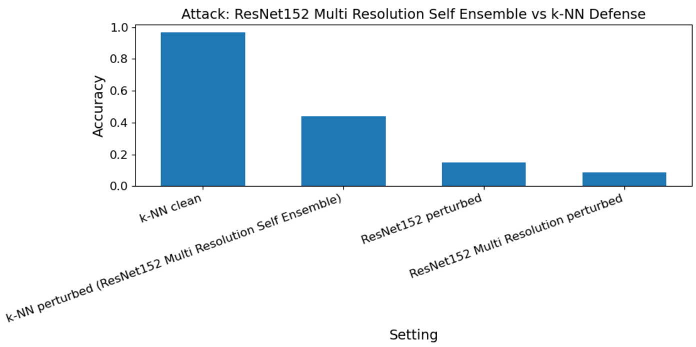
(a) Models accuracies on ResNet152 attack.



(b) Models accuracies on ResNet152 Self Ensemble attack.

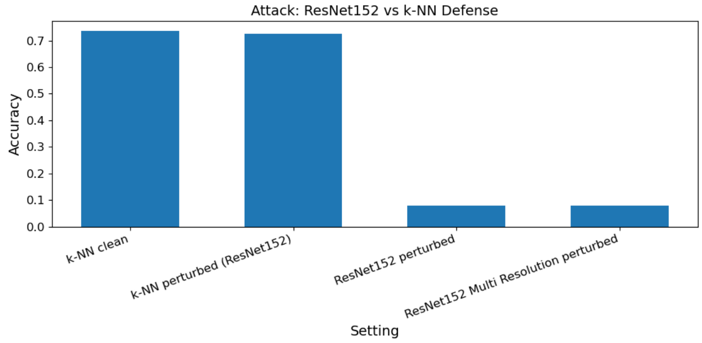


(c) Models accuracies on Multi-Resolution ResNet152 attack.

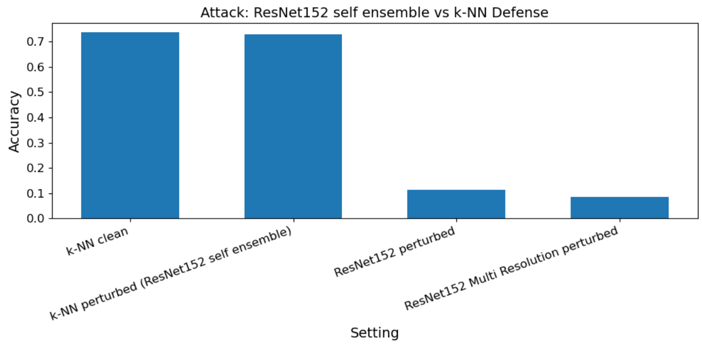


(d) Model accuracies on Multi-Resolution ResNet152 Self Ensemble attack.

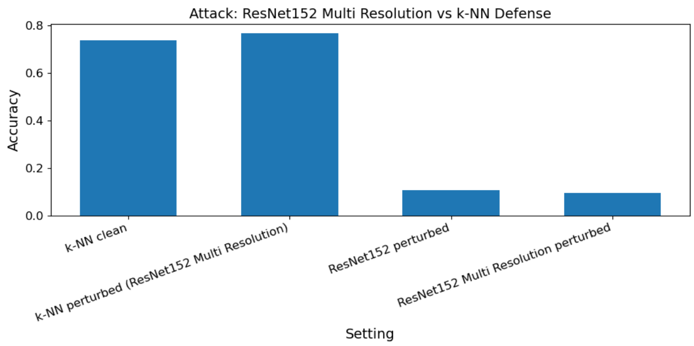
Figure 4.3: k-NN model with normal ResNet152 backbone accuracy compared to other models



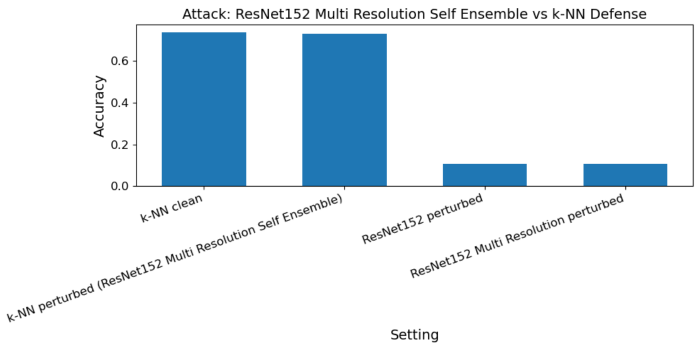
(a) Models accuracies on ResNet152 attack.



(b) Models accuracies on ResNet152 Self Ensemble attack.

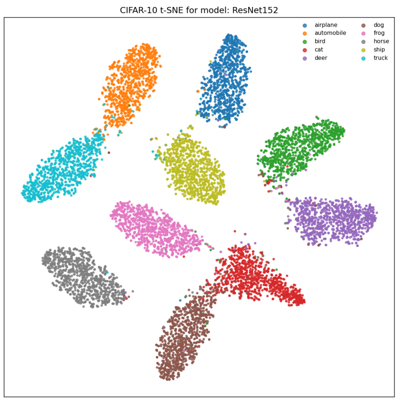


(c) Models accuracies on Multi-Resolution ResNet152 attack.

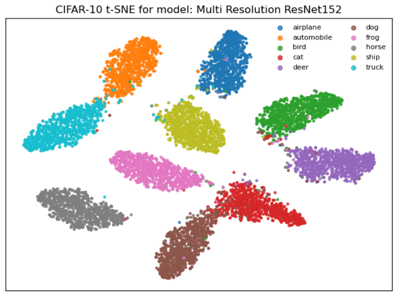


(d) Models accuracies on Multi-Resolution ResNet152 Self Ensemble attack.

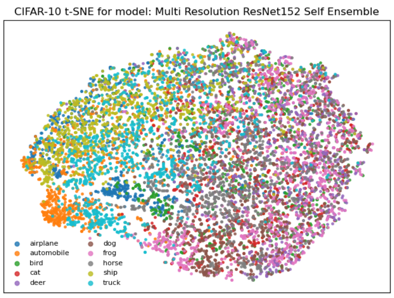
Figure 4.4: k-NN model with Multi-Resolution ResNet152 backbone accuracy compared to other models



(a) t-SNE Plot of ResNet152 class distribution on CIFAR-10



(b) t-SNE Plot of Multi Resolution ResNet152 class distribution on CIFAR-10



(c) t-SNE Plot of Multi Resolution ResNet152 Self Ensemble class distribution on CIFAR-10

Figure 4.5: t-SNE Plot of CIFAR-10 class distribution for different robustness steps of ResNet152

Bibliography

[1] S. Fort and B. Lakshminarayanan, “Ensemble everything everywhere: Multi-scale aggregation for adversarial robustness,” 2024. [Online]. Available: https://arxiv.org/abs/2408.05446

[2] J. Zhang, C. Schlarmann, K. Nikolić, N. Carlini, F. Croce, M. Hein, and F. Tramèr, “Evaluating the robustness of the "ensemble everything everywhere" defense,” 2025. [Online]. Available: https://arxiv.org/abs/2411.14834

[3] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, “Intriguing properties of neural networks,” 2014. [Online]. Available: https://arxiv.org/abs/1312.6199

[4] N. Carlini and D. Wagner, “Towards evaluating the robustness of neural networks,” 2017. [Online]. Available: https://arxiv.org/abs/1608.04644

[5] A. Athalye, N. Carlini, and D. Wagner, “Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples,” 2018. [Online]. Available: https://arxiv.org/abs/1802.00420

[6] F. Tramer, N. Carlini, W. Brendel, and A. Madry, “On adaptive attacks to adversarial example defenses,” 2020. [Online]. Available: https://arxiv.org/abs/2002.08347

[7] F. Croce and M. Hein, “Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks,” 2020. [Online]. Available: https://arxiv.org/abs/2003.01690

[8] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, “Towards deep learning models resistant to adversarial attacks,” 2019. [Online]. Available: https://arxiv.org/abs/1706.06083

[9] B. R. Bartoldson, J. Diffenderfer, K. Parasyris, and B. Kailkhura, “Adversarial robustness limits via scaling-law and human-alignment studies,” 2024. [Online]. Available: https://arxiv.org/abs/2404.09349

[10] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” 2015. [Online]. Available: https://arxiv.org/abs/1412.6572

[11] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter, "A general framework for adversarial examples with objectives," *ACM Transactions on Privacy and Security*, vol. 22, no. 3, p. 1–30, Jun. 2019. [Online]. Available: http://dx.doi.org/10.1145/3317611

[12] M. Mosbach, M. Andriushchenko, T. Trost, M. Hein, and D. Klakow, "Logit pairing methods can fool gradient-based attacks," 2019. [Online]. Available: https://arxiv.org/abs/1810.12042

[13] Y. Luo, X. Boix, G. Roig, T. Poggio, and Q. Zhao, "Foveation-based mechanisms alleviate adversarial examples," 2016. [Online]. Available: https://arxiv.org/abs/1511.06292

[14] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok, "Synthesizing robust adversarial examples," 2018. [Online]. Available: https://arxiv.org/abs/1707.07397

[15] F. Croce and M. Hein, "Minimally distorted adversarial examples with a fast adaptive boundary attack," 2020. [Online]. Available: https://arxiv.org/abs/1907.02044

[16] M. Andriushchenko, F. Croce, N. Flammarion, and M. Hein, "Square attack: a query-efficient black-box adversarial attack via random search," 2020. [Online]. Available: https://arxiv.org/abs/1912.00049

[17] F. Croce, M. Andriushchenko, V. Sehwag, E. Debenedetti, N. Flammarion, M. Chiang, P. Mittal, and M. Hein, "Robustbench: a standardized adversarial robustness benchmark," 2021. [Online]. Available: https://arxiv.org/abs/2010.09670

[18] R. S. Zimmermann, W. Brendel, F. Tramer, and N. Carlini, "Increasing confidence in adversarial robustness evaluations," 2022. [Online]. Available: https://arxiv.org/abs/2206.13991

[19] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2021. [Online]. Available: https://arxiv.org/abs/2010.11929

[20] M. Dehghani, J. Djolonga, B. Mustafa, P. Padlewski, J. Heek, J. Gilmer, A. Steiner, M. Caron, R. Geirhos, I. Alabdulmohsin, R. Jenatton, L. Beyer, M. Tschannen, A. Arnab, X. Wang, C. Riquelme, M. Minderer, J. Puigcerver, U. Evci, M. Kumar, S. van Steenkiste, G. F. Elsayed, A. Mahendran, F. Yu, A. Oliver, F. Huot, J. Bastings, M. P. Collier, A. Gritsenko, V. Birodkar, C. Vasconcelos, Y. Tay, T. Mensink, A. Kolesnikov, F. Pavetić, D. Tran, T. Kipf, M. Lučić, X. Zhai, D. Keysers,

J. Harmsen, and N. Houlsby, "Scaling vision transformers to 22 billion parameters," 2023. [Online]. Available: https://arxiv.org/abs/2302.05442

[21] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, M. Assran, N. Ballas, W. Galuba, R. Howes, P.-Y. Huang, S.-W. Li, I. Misra, M. Rabbat, V. Sharma, G. Synnaeve, H. Xu, H. Jegou, J. Mairal, P. Labatut, A. Joulin, and P. Bojanowski, "Dinov2: Learning robust visual features without supervision," 2024. [Online]. Available: https://arxiv.org/abs/2304.07193

[22] O. Siméoni, H. V. Vo, M. Seitzer, F. Baldassarre, M. Oquab, C. Jose, V. Khalidov, M. Szafraniec, S. Yi, M. Ramamonjisoa, F. Massa, D. Haziza, L. Wehrstedt, J. Wang, T. Darcet, T. Moutakanni, L. Sentana, C. Roberts, A. Vedaldi, J. Tolan, J. Brandt, C. Couprie, J. Mairal, H. Jégou, P. Labatut, and P. Bojanowski, "Dinov3," 2025. [Online]. Available: https://arxiv.org/abs/2508.10104

[23] E. Galinkin and M. Sablotny, "Improved large language model jailbreak detection via pretrained embeddings," 2024. [Online]. Available: https://arxiv.org/abs/2412.01547

[24] N. Papernot and P. McDaniel, "Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning," 2018. [Online]. Available: https://arxiv.org/abs/1803.04765