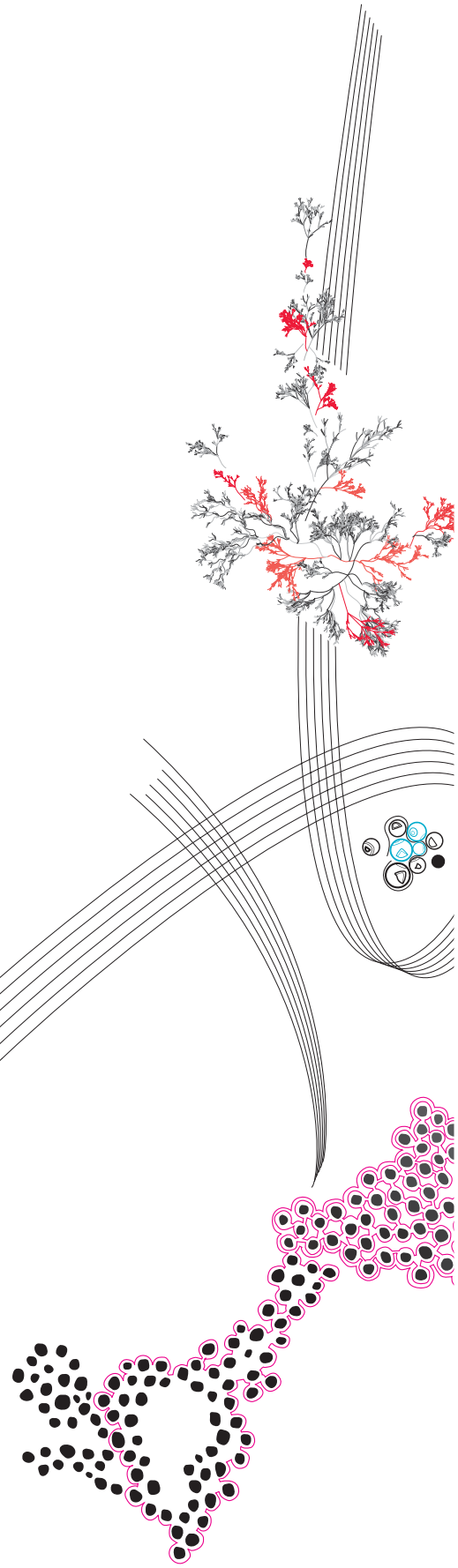

Deep Learning: From Theory to Practice

Adversarial Attacking & Defending a ResNet Classifier on a Pokemon Dataset

Felix de Natris, s2661209
Manuel Maria Rocha, s2900114

September, 2024

Department of EEMCS

UNIVERSITY OF TWENTE.

This report describes the implementation of an adversarial attack (*AutoAttack* (Croce & Hein, 2020)) to a Residual Network (*ResNet18*), trained to classify the different 1^{st} generation Pokemons, as well as the implementation of an adversarial defense (*FFT*-based defence inspired by (Lorenz et al., 2021) and (Zhang et al., 2019)), to protect the network from the proposed attack.

1 Research background

1.1 The Attack: Autoattack

The proposed adversarial attack is based on (Croce & Hein, 2020)'s *AutoAttack*. This method combines different types of attacks into one attacking ensemble; namely two variations of the *Auto-Projected Gradient Descent* (Croce & Hein, 2020), the *Fast Adaptive Boundary attack* (Croce & Hein, 2019) and the *Square Attack* (Andriushchenko et al., 2019).

The *Auto-PGD* attack (Croce & Hein, 2020) is a newer, altered version of the Projected Gradient Descent-based attacks. This attack tries to tackle three problems of the classic PGD approach: the choice of a step-size, the evaluation of the attack's strength, and the online adaptability of the attack.

As the standard *PDG* attack, the *APDG* is also a white-box attack, meaning that it has access to the model's inner parameters. The attack attempts to induce misclassification by maximizing the loss function (the inverse of what the network's training process does). The authors explain that the *APDG* update step is essentially the same as the classical *PDG* with the exception of a momentum term, that aims to regulate the influence of the previous update in the current one.

Furthermore, the paper also states that the step-size is halved every time one of two conditions is met: if the objective function stagnates for a proportion of all the update steps OR if the step size has not been reduced in the last checkpoint and the best value of the objective function remains the same on the current checkpoint[1]. It is further detailed by (Croce & Hein, 2020), that every time the step-size is reduced, then the algorithm resumes from the point that achieved the best maximization of the objective function i.e. $x_{max} \;\; s.t. \;\; f(x_{max}) = \max f(x)$.

The checkpoint initialization is also addressed by the authors. They attempt to make the algorithm smoothly transition from exploration to exploitation i.e. from exploring the whole hyper-plane to a localized optimization. The step-size is what regulates both phases. A big step-size means that the progression is done with with big jumps (similarly to big learning rates while training a neural network), whereas a small step-size means small movements in the loss landscape. As previously explained, the step-size is decreased based on the state of the algorithm upon reaching the checkpoints. The authors propose a technique for the checkpoint initialization that enlarges the exploration phase and slowly transitions to the exploitation. The checkpoints are initializaed as $w_j = [p_j \cdot N_{iter}] \leq N_{iter}$, where $p_j \in [0..1]$ and N_{iter} is the iteration budget for the attack. The initialization of $p_0 = 0$ and $p_1 = 0.22$, the remaining checkpoints follow Equation 1.

$$p_{j+1} = p_j + \max\left(p_j - p_{j-1} - 0.03, 0.06\right) \tag{1}$$

Having the basis of the APGD algorithm tackled, (Croce & Hein, 2020) further demonstrate that the regular *Cross-Entropy* loss function (CE) is sub-optimal, and that the

[1]The terminology "checkpoint" is referred by the authors as a specific iteration e.g. 100, 200, etc.

Difference of Logits Ratio (DLR) loss manages to improve the attack results. Moreover, the authors prove that the targeted version of the DLR-based APGD outperforms its untargeted version. As such, the two APGD-based attacks that will enrol the *AutoAttack* enseble are the *Auto-Projected Gradient Descent* with *Cross-Entropy* loss (APGD-CE) and the targeted *Auto-Projected Gradient Descent* with *Difference of Logits Ratio* loss (APGD-DLR^T).

The *Fast Adaptive Boundary* (FAB) attack (Croce & Hein, 2019) is a white-box adversarial attack that aims to generate adversarial samples with minimal distance to the attacking point, evaluated by l_p-norms for $p \in \{1, 2, \infty\}$. The authors state that the FAB is a low-computational cost attack, that provides good insight on the robustness of most networks i.e. the parameters associated with the attack can be generalized for many different datasets. The attack's algorithm is rather complex, given the relation with geometric projections. In the context of the project at hand, it is mostly important to understand the the attack generates attacking samples as close as possible to the original image.

In the *AutoAttack* paper (Croce & Hein, 2020), the authors yet again prove that by using the targeted version of an attack, in this case, the FAB, the success rate and the computational cost of the attack decrease. Thus, the targeted *Fast Adaptive Boundary* attack (FAB^T) also joins the attacking ensemble.

The *Square Attack* (Andriushchenko et al., 2019) is a score-based black-box attack, that uses the L_2 or L_{inf} loss (to compute the difference between adversarial example and original image). It is based on the random search algorithm with some changes (described in detail in (Andriushchenko et al., 2019)). In essence, the authors integrate two sampling distributions (one for each loss, L_p) in the searching algorithm. According to them, the distributions take into account the way convolutional layers treat the input images. Furthermore, (Andriushchenko et al., 2019) state that the perturbation budget is almost completely used in every searching step and that the changes made to the input image are only done on sets of pixels that resemble squares.

1.2 The Defense: Fourier Transform Detection

A proposed defense for the *AutoAttack* is (Lorenz et al., 2021)'s Fourier Transform-based adversarial detector. According to the authors, the proposed defense falls under the Adversarial Detection type, and can either be black box or white box. The first option distinguises adversarial from benign examples based only on the Discrete Fourier Transform of the input images, whereas the second uses the feature maps of the convolutional layers.

The transformation of the input image into the frequency domain breaks it down to its constituent frequencies, possibly allowing the visualization of phenomena non-identifiable in the spatial domain. The authors argue that adversarial attacks result in strong and systematic changes in the frequency domain, which may allow the detection of adversarial samples. They prove that using a black box defense is sufficient to identify those systematic behaviours (see Figure 1).

(Lorenz et al., 2021) state that the frequency domain signals can be learned by simple classifiers (e.g. Random Forests) to distinguish between altered and original images. The defense mechanism is considered binary, as the output takes the value e.g. *True* if the image is original and *False* otherwise. Based on the output of the defensive network the image could then be forwarded to the model under attack or dismissed.

The paper by (Lorenz et al., 2021), composed the initial inspiration for the to-be implemented defense against *AutoAttacks* (Croce & Hein, 2020), that will be further detailed

on the following Sections.

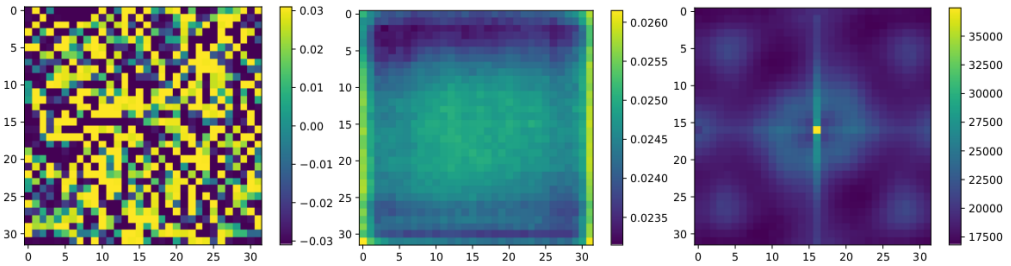

FIGURE 1: Visualisation of *AutoAttack* perturbations on a *ResNet18* with *CIFAR10* as dataset. Left image demonstrates the spacial differences between an input image and its perturbation. The central image shows the average of of spacial differences over 1000 adversarial pictures. The right image depicts the accumulated magnitudes of the spectral differences of the same 1000 inputs (this grid of images was taken from (Lorenz et al., 2021)).

2 Theoretical analysis

2.1 *AutoAttack* on the Pokemon dataset

The idea of applying an attacking ensemble poses some advantages when compared with single attacks. The diversity created by the ensemble contributes to a more broad and robust attack. As previously described, each attack has a different working principle, and affects the network in a different way. Also, the strengths of some attacks might compensate the weaknesses of other e.g. the PGD based attacks (as the APGD) are susceptible to fail when the network embeds some form of gradient masking, on the opposite hand, the FAB attack, according to (Croce & Hein, 2019), is effective on networks with that characteristic.

By using the proven most successful attacks of different breeds on the same network, one can induce misclassification with much higher probability. The redundancy of attacks almost certainly guarantees that one of the elements of the ensemble will successfully fool the network. Furthermore, the fact that the *AutoAttack* algorithm, proposed and implemented by (Croce & Hein, 2020), is almost completely parameter-free increases the productivity and effectiveness of adversarial attack development, and provides a first insight on the robustness of the network.

As will be explained in the following Sections, the proposed dataset and network are very simple, which means that the success rate of the *AutoAttack* is ought to be really high.

2.2 Embeded DFT to defend from the *AutoAttack*

To take things a step further, the initial idea based on (Lorenz et al., 2021)'s paper ramified to the inclusion of a Discrete Fourier Transform (DFT) in the network's architecture. The key principle was to transform the the input image into its constituent frequencies, using the a 2D Discrete Fourier Transform (see Equation 2) applied to every input channel, and then filter out the high frequencies. Afterwards, the inverse DFT of the image would be applied to get the sample onto the space domain, and forwarded to the classifier.

$$F(X)(l,k) = \sum_{n,m=0}^{N} e^{-2\pi i \frac{l_m + k_n}{N}} X(m,n) \tag{2}$$

As mentioned in Section 1.2, according to (Lorenz et al., 2021), the adversarial examples generated by the *AutoAttack* result in higher frequency patterns in the frequency domain. The idea of undoing i.e. de-noising, an adversarial image was also developed by (Zhang et al., 2019). The basic algorithm proposed by the authors is to use a low pass filter, with a fixed radius, that sets to zero all the (high) frequencies outside that radius.

(Zhang et al., 2019) proved that by performing high-frequency suppression, the adversarial robustness of the prey network can be improved. The authors tested their hypothesis based on the PGD attack, which is a much simpler attack than the one to-be tested (i.e. *AutoAttack*), however, based on the gathered information from (Lorenz et al., 2021), it should be possible that the same algorithm could provide an effective defense against the most state-of-the-art attack: the *AutoAttack*.

3 Implementation

3.1 Data

The image classifier was trained on an image dataset consisting of the first generation of Pokemons[2]. The dataset is composed of 11945 different images and pictures of 151 different types of Pokemons. The reason for utilizing this dataset was to go beyond the regular classification tasks (proposed by ImageNet, CIFAR-10, etc.). Nevertheless, the dataset still posed a fair classification challenge due to having varying inputs for each class, ranging from standard images to drawings and pictures of plushies (see Figure 2). In addition, the dataset is sufficiently balanced, so no data augmentation algorithm had to be implemented.

FIGURE 2: Different inputs for the same class: *Dragonite*.

The dataset was divided into training set and testing set using the proportion 0.8/0.2. Both sets were normalized to the range of [-1..1]. To improve the network's robustness, the training set suffered some extra transformations: random horizontal flips (the input image is randomly flipped over the y-axis) and rotations of a maximum of 15° (the input image is randomly rotated with an angle in the range of [-15..15]°). One example batch is provided on Figure 3.

[2]https://www.kaggle.com/unexpectedscepticism/11945-pokemon-from-first-gen

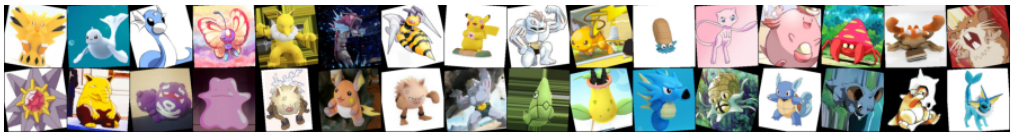

FIGURE 3: Training mini-batch example.

3.2 The Classifier

The chosen image classifier was a Convolutional Neural Network based on Residual Blocks, that goes by the name of *ResNet18*. As it is well known the main characteristic of this type of architecture is the fact that the residual block learns based on its identity i.e. the input of the residual block, x, is forwarded to the output of the block through skip connections (cf. Figure 4).

Furthermore, the skip connections do not add more parameters to be learned by the network, but are able to eliminate the problem of vanishing gradients. According to (He et al., 2016), the computations behind a residual block as the one from Figure 4 can be summed up by Equation 3, where the function $h(x_l)$ represents the identity mapping i.e. x_l, W_l the weights and biases of the residual block, and $\mathcal{F}$ represents the residual function i.e. the stack of the two convolutional layers.

$$\begin{aligned} y_l &= h(x_l) + \mathcal{F}(x_l, W_l) \\ x_{l+1} &= f(y_l) \end{aligned} \tag{3}$$

(He et al., 2016) also state that if the input of the next block i.e. $x_{l+1} = y_l$ (if f is also an identity mapping), then the gradient of the loss function can be decomposed in two additive terms: one that does not depend on any weights or biases ($\frac{\delta L}{\delta x_L}$, where x_L is the input of the last residual unit) and one that does ($\frac{\delta L}{\delta x_L}(\frac{\delta}{\delta x_l}\sum_{i=l}^{L-1}\mathcal{F})$). The first, ensures the propagation of the information is directly back-passed to any residual unit, whilst the second one guarantees the weight modifications. The authors also point out that it is statistically unlikely that the two terms cancel each other out, therefore ensuring that the gradients will not vanish even if the weights are very small.

This type of architecture was a major breakthrough on the deep learning universe, as it allowed much deeper networks to be effective (due to the gradients being correctly back-propagated), and achieve much better accuracy than plain networks with the same number of parameters (He et al., 2015).

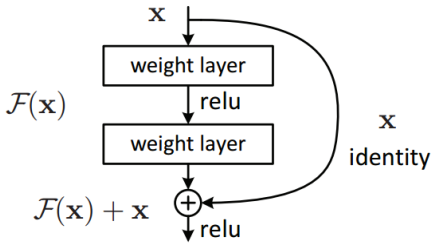

FIGURE 4: Basic residual block.

Being such a powerful architecture, it represented a good candidate for the classification task at hand. At first, the idea was to use a *ResNet50*, however, given the simplicity of the dataset (when compared to *ImageNet*), the 18-layered version of this network type was

used. This reduced the computational load (less training time) and still provided good results.

Making use of transfer learning also improved the training process. An *ImageNet* pre-trained *ResNet18* model was used. This allowed for a good weight initialization, thus saving training time (as previously mentioned, the *ImageNet* dataset is far more complex than the *Pokemon* one). Furthermore, a linear layer was added to the end of the imported model, with the objective of re-mapping the pre-trained model's outputs to the correct number of classes (i.e. 151).

During the training process, the training set was split into batches of 32 images (cf. Figure 3). Moreover, the optimizer used was *SGD* with momentum, with an initial learning rate of 0.01. A learning rate scheduler was also applied to force the learning rate to reduce, once the validation accuracy stagnated over 90% for three consecutive epochs. Finally, the cross entropy loss function was used.

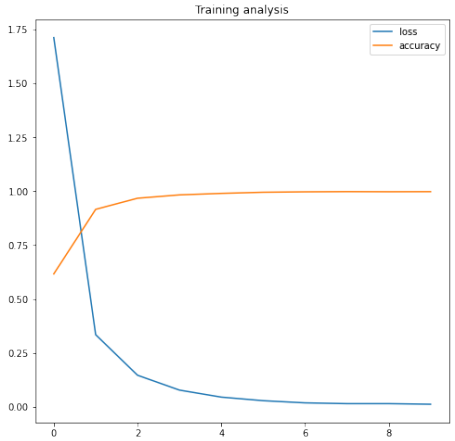

FIGURE 5: Training performance graph.

After training the network for 10 epochs, the test accuracy was 95.4%, which confirmed that the dataset was indeed simple and that the chosen network fulfilled the classification requirements. As it is visible from Figure 5, the training accuracy reaches 100% and the training loss settles on the its minimum value after the sixth epoch. This implies that even with further training i.e. more epochs the network could not achieve better results on the testing set, without adding new images (by means of data augmentation) to the training set.

3.3 The Attack

The *AutoAttack* was implemented by making use of the library implemented by (Croce & Hein, 2020), which is available on their GitHub. As mentioned by the authors, the *AutoAttack* is supposed to be a parameter-free attack, so most of the participant attacks have fixed hyper-parameters. In fact, the only parameters to be regulated are the maximum perturbation allowed, ϵ, and the loss norm, L_p.

The *standard* version of the *AutoAttack*, composed by the APGD-CE, APGD-DLR^T, FAB^T and Square attacks, was used. The *AutoAttack* works in a sequence of attacks, where only the unperturbed images proceed to the next attack. The library gives the option of using a reinforced version of the ensemble, the *AutoAttack+*. This version uses more attacks with different hyper-parameter configurations, to provide a more robust and versatile attack. Furthermore, the library also allows the usage of a custom attacking ensemble.

Due to resource and time constraints the adversarial transformations could not be performed on the whole the test set (2389 images). Therefore, the number of images used for the attack was 512, on mini-batches of 32. The L_p norm used for the attack was L_∞ with a base maximum perturbation of $\epsilon_b = \frac{8}{255}$ (which is the default value). Different maximum perturbations were tested, to understand how that would affect the attack's performance and the behaviour of the prey models.

3.4 The Defense

As previously mentioned, the goal of this defence is to remove the higher frequencies added by the perturbations made by the *AutoAttack*. By filtering out those frequencies one can make the adversarial feature maps close to the original ones i.e. the ones the model is trained on.

The new CNN architecture is different from the previous one, since it now has, as a first layer, a DFT transformation. To train the new model, the same training samples as in Section 3.1 are sent through the network (i.e. *ResNet-18* with the linear mapping layer, explained in Section 3.2), after being transformed, filtered and inverse-transformed by means of DFT and frequency-domain-transformations (cf. Figure 6). To exploit the effects of the box-filter radius on the training process, two versions of the network were trained: one where the radius was set to 12 and one where it was set to 8. After 10 epochs, the test accuracy of the network with $r = 12$ was 89.5% and with $r = 8$ was 84.6%.

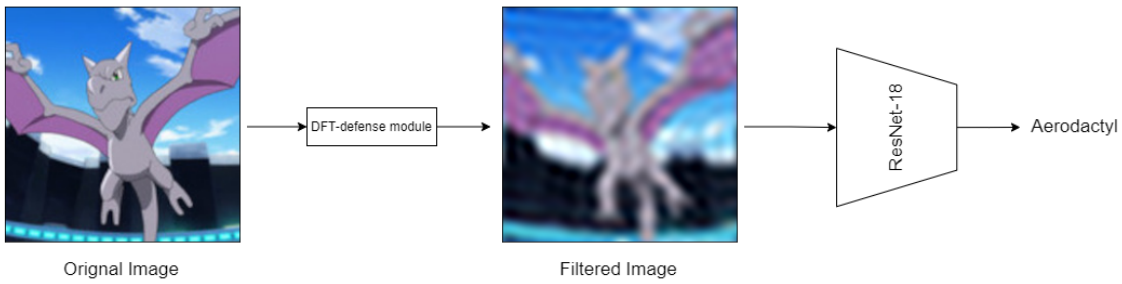

FIGURE 6: Defensive architecture. The radius of the filter used on the DFT-defense module was 12.

The implementation of the defense was based on the Equations mentioned by (Zhang et al., 2019). Firstly, the image is transformed to the frequency domain, by the Fast Fourier Transform (cf. Equation 2), using `PyTorch`'s `fftn()`. This function computes the n-dimensional FFT of the input Tensor (image).

Afterwards, the transformed image, $\hat{x}$, is modified by removing its higher frequencies. This is achieved by applying Equation 4, where M is the filter template, defined by Equation 5, and $\odot$ is the element-wise multiplication. The second equation was retrieved from (Zhang et al., 2019) and represents a box window of fixed radius r, where the values outside that box are set to zero.

$$\hat{x} \leftarrow M \odot \hat{x} \tag{4}$$

$$M_{u,v} = \begin{cases} 1, & 0 <= |u|, |v| <= r \\ 0, & else \end{cases} \tag{5}$$

To compute the inverse transform of the filtered image, `PyTorch`'s function `invfftn()` was used on said image. The complete sequence of actions can be summed up by Equation 6, where $\mathcal{F}$ represents the Fast Fourier Transform.

$$x \leftarrow \mathcal{F}^{-1}(M \odot \mathcal{F}(x)) \tag{6}$$

It is important to note that, the more r moves to zero, the more frequencies get removed. This removes an increasing number of details, resulting in blurrier images. Some examples of this phenomena are presented on Figure 7.

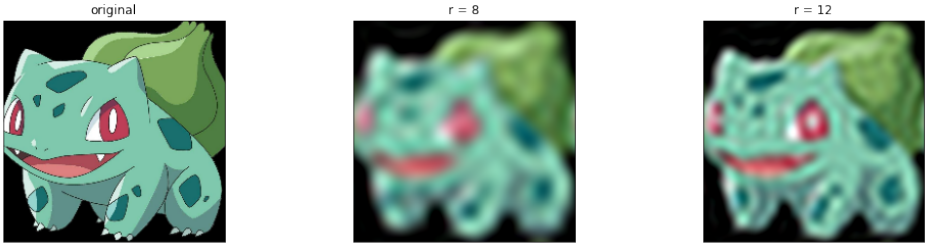

FIGURE 7: Effect of box-radius size on the blurriness of the input image.

4 Numerical Analysis

4.1 Attack performance evaluation

To evaluate the *AutoAttack*'s performance on the defenseless *ResNet-18*, different values for the maximum perturbation, ϵ, were tested. The values were $\{\frac{\epsilon_b}{8}, \frac{\epsilon_b}{4}, \frac{\epsilon_b}{3}, \frac{\epsilon_b}{2}, \epsilon_b, 2\epsilon_b\}$. The gathered results are presented on Table 1. The individual performance of the attacks is computed based on Equation 7, where index $^{(i)}$ represents the current attack and $^{(i-1)}$ the previous one.

$$I.R_{acc}^{(i)} = \frac{G.R_{acc}^{(i-1)} - G.R_{acc}^{(i)}}{G.R_{acc}^{(i-1)}} \tag{7}$$

As previously mentioned, due to time and computational constraints, the test set was shortened to 512 images. As expected, the test accuracy changed when only 512 images are considered; it took the value of 94.53%.

		APGD-CE	**APGD-DLR^T**	**FAB^T**	**Square**
$\frac{\epsilon_b}{8}$	$G.R_{acc}$	0.623	0.5254	0.5254	0.5254
	$I.R_{acc}$	-	0.1567	0	0
$\frac{\epsilon_b}{4}$	$G.R_{acc}$	0.3633	0.2031	0.2031	0.2031
	$I.R_{acc}$	-	0.4410	0	0
$\frac{\epsilon_b}{3}$	$G.R_{acc}$	0.2090	0.0879	0.0879	0.0879
	$I.R_{acc}$	-	0.5794	0	0
$\frac{\epsilon_b}{2}$	$G.R_{acc}$	0.0469	0.0195	0.0195	0.0195
	$I.R_{acc}$	-	0.5842	0	0
ϵ_b	$G.R_{acc}$	0.002	0	-	-
	$I.R_{acc}$	-	1	-	-
$2\epsilon_b$	$G.R_{acc}$	0	-	-	-
	$I.R_{acc}$	-	-	-	-

TABLE 1: *AutoAttack*'s performance results for different maximum perturbations, ϵ. $G.R_{acc}$ is the global attack robustness of the model, after every attack (lower means better attack performance). $I.R_{acc}$ is the individual performance of the attack on the images that could not be perturbed by the previous attack (bigger means better attack performance).

From the results it is possible to visualize that the *AutoAttack* is, indeed, very effective against the basic *ResNet-18* architecture. As expected, the bigger the maximum perturbation, the more effective the attack is, but also, the changes made to the input image become more visible (see Figure 8). The complementarity between the first attack

(APGD-CE) and the second one (APGD-DLR^T) is clearly visible, for all cases where the first attack fails to cause misclassification. Unfortunately, due to the test set being clipped to 512 images, the effectiveness of the two last attacks of the sequence could not be tested in depth.[3]

Overall, the performance of the *AutoAttack* falls under the expectations, as the networks accuracy quickly drops. Moreover, on this specific dataset, the effects of the last attacks (FAB^T and Square) do not add much value to the ensemble.

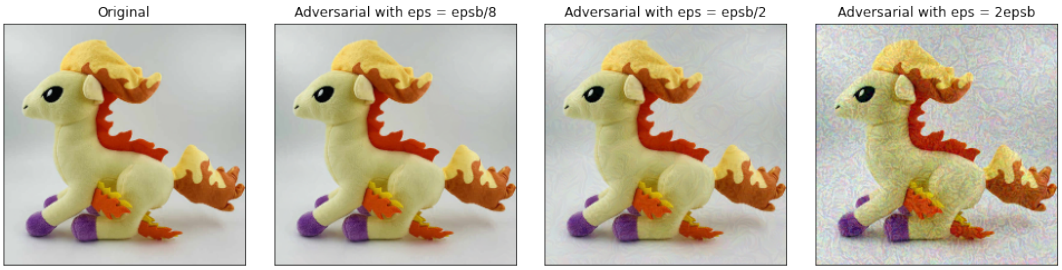

FIGURE 8: The effect of the maximum perturbation, ϵ, on the generation of adversarial examples, by the *AutoAttack*.

4.2 Defense performance evaluation

The same test setup as in Section 4.1 was used to test the performance of the defensive mechanism. As previously stated, two different protected networks were trained: one with a filter of size $r = 12$ and another with a filter of size $r = 8$.

The same 512 images were used to correctly assess the effectiveness of the DFT-based defense, and, again, the test accuracy changed due to the data-clippage: for $r = 12$ the test accuracy became 89.06%, whereas for $r = 8$ became 86.13%. The robustness results after the same set of attacks, on the network with a filter size of 12 is on Table 2 and with a filter of size 8 on Table 3.

From the Tables it is clear that the filter size affects the performance of the defense. If one looks at the global effect of the ensemble (last column with numerical values), then both models behave similarly. However, when one looks at the individual performance of the individual attacks it is clearly visible that the version with $r = 12$ (Table 2), performs the best, since there is more effort required from the *AutoAttack* to sucessfully fool the network e.g. the FAB^T attack plays an important role when $\epsilon = \epsilon_b$ and when $\epsilon = 2\epsilon_b$.

Regardless, the results show that the defense is effective for all perturbation values above ϵ_b, as the network achieves a 10% higher robustness accuracy (for those perturbation values). Moreover, it is important to state that the results would provide better insight on both the attack and the defense if the whole test set was used.

5 Suggested Improvements

Despite being on the right direction, some improvements could be made to the defense. Firstly, in order to reduce the blurriness induced by the filtering process, an improved low

[3] In fact, on different images from the test set it was visible that the FAB^T and the Square attack could affect some images, however those results are not presented to keep a consistent data analysis i.e. all measures on the same sample images.

		APGD-CE	**APGD-DLR^T**	**FAB^T**	**Square**
$\frac{\epsilon_b}{8}$	$G.R_{acc}$	0.6426	0.6387	0.6387	0.6387
	$I.R_{acc}$	-	0.006069	0	0
$\frac{\epsilon_b}{4}$	$G.R_{acc}$	0.4121	0.2910	0.2910	0.2910
	$I.R_{acc}$	-	0.2939	0	0
$\frac{\epsilon_b}{3}$	$G.R_{acc}$	0.3242	0.2012	0.2012	0.2012
	$I.R_{acc}$	-	0.4206	0	0
$\frac{\epsilon_b}{2}$	$G.R_{acc}$	0.0449	0.0352	0.03252	0.0352
	$I.R_{acc}$	-	0.2160	0	0
ϵ_b	$G.R_{acc}$	0.0254	0.002	0	-
	$I.R_{acc}$	-	0.9213	1	-
$2\epsilon_b$	$G.R_{acc}$	0.0039	0.002	0	-
	$I.R_{acc}$	-	0.4871	1	-

TABLE 2: *AutoAttack*'s performance against the DFT-protected model with $r = 12$, for different maximum perturbations, ϵ. $G.R_{acc}$ is the global attack robustness of the model, after every attack (lower means better attack performance). $I.R_{acc}$ is the individual performance of the attack on the images that could not be perturbed by the previous attack (bigger means better attack performance).

		APGD-CE	**APGD-DLR**	**FAB**	**Square**
$\frac{\epsilon_b}{8}$	$G.R_{acc}$	0.6211	0.6191	0.6191	0.6191
	$I.R_{acc}$	-	0.00322	0	0
$\frac{\epsilon_b}{4}$	$G.R_{acc}$	0.3125	0.3047	0.3047	0.3047
	$I.R_{acc}$	-	0.02496	0	0
$\frac{\epsilon_b}{3}$	$G.R_{acc}$	0.2051	0.1934	0.1934	0.1934
	$I.R_{acc}$	-	0.4206	0	0
$\frac{\epsilon_b}{2}$	$G.R_{acc}$	0.0645	0.0605	0.0605	0.0605
	$I.R_{acc}$	-	0.4158	0	0
ϵ_b	$G.R_{acc}$	0	-	-	-
	$I.R_{acc}$	-	-	-	-
$2\epsilon_b$	$G.R_{acc}$	0	-	-	-
	$I.R_{acc}$	-	-	-	-

TABLE 3: *AutoAttack*'s performance against the DFT-protected model with $r = 8$, for different maximum perturbations, ϵ. $G.R_{acc}$ is the global attack robustness of the model, after every attack (lower means better attack performance). $I.R_{acc}$ is the individual performance of the attack on the images that could not be perturbed by the previous attack (bigger means better attack performance).

pass filter could be used e.g. a low-pass *Butterworth*. Another, more state-of-the-art filtering approach could be the use of the *Anisotropic Diffusion* algorithm. This algorithm filters out the noise while keeping some high-frequency features, hence, reducing the blurriness of the output image.

Another way to improve the defense could be to perform some adversarial training i.e. use the adversarial examples generated by the *AutoAttack* to train the network. This would, almost certainly, guarantee the increase of the global attack robustness of the network ($G.R_{acc}$).

Furthermore, as stated on the introductory part of this paper, the Discrete Fourier

Transform can be used to detect adversarial images (Lorenz et al., 2021). If, similarly to the *AutoAttack*, two defense mechanisms would work together i.e. the additional network that performs the adversarial detection and the frequency filtering, the defense's global performance could increase a lot. The principle idea would be to only let the original (benign) images pass through and discard the adversarial ones, whilst maintaining the filtering mechanism. This would create some redundancy on the protective mechanism: if the adversarial detection failed, the adversarial image would still be filtered, decreasing the changes of attack success.

6 Conclusions

As it was visible from the contents of this paper, the *AutoAttack* poses a clear challenger to the simple *ResNet-18* model trained on the *Pokemon* dataset. The attacking ensemble creates attacking redundancy, which clearly increases the chances of attack success. Regardless, a trade-off needs to be made when using this type of attack: effectiveness versus noise visibility. The bigger the maximum perturbation, the more visible the attack is to the naked eye, which, depending on the application might be undesirable. However, if the aim to cause the most harm to the network, then the higher the maximum perturbation the higher the probability of success.

When it comes to the defense, despite being inspired by two different papers, the implementation is somewhat novel, as the DFT filtering was never attempted against the *AutoAttack*[4]. During the fine-tuning of the filter size, one can clearly understand that the performance of the defense is highly dependant on this parameter. On one instance, the filter size was set to 16, which resulted in no improvement on the network's robustness. When choosing the filter size it is important to take into account that the smaller the value, the weaker the performance of the network on normal conditions. Nevertheless, the defensive mechanism proved that by using the frequency domain, it is possible to defend against some adversarial attacks, and leaves an open-door to the *AutoDefense*: an ensemble of defensive mechanisms.

[4] At least, from the information collected until thw writing of this report.

References

Andriushchenko, M., Croce, F., Flammarion, N., & Hein, M. (2019). Square attack: A query-efficient black-box adversarial attack via random search. *CoRR*, *abs/1912.00049*. http://arxiv.org/abs/1912.00049

Croce, F., & Hein, M. (2019). Minimally distorted adversarial examples with a fast adaptive boundary attack. *CoRR*, *abs/1907.02044*. http://arxiv.org/abs/1907.02044

Croce, F., & Hein, M. (2020). Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep residual learning for image recognition. *CoRR*, *abs/1512.03385*. http://arxiv.org/abs/1512.03385

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Identity mappings in deep residual networks. *CoRR*, *abs/1603.05027*. http://arxiv.org/abs/1603.05027

Lorenz, P., Harder, P., Straßel, D., Keuper, M., & Keuper, J. (2021). Detecting autoattack perturbations in the frequency domain. *ICML 2021 Workshop on Adversarial Machine Learning*. https://openreview.net/forum?id=8uWOTxbwo-Z

Zhang, Z., Jung, C., & Liang, X. (2019). Adversarial defense by suppressing high-frequency components.