USE OF SMALL AUXILIARY NETWORKS AND SCARCE DATA TO IMPROVE THE ADVERSARIAL ROBUSTNESS OF DEEP LEARNING MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Deep Learning models for image classification are known to be vulnerable to adversarial examples. Adversarial training is one of the most effective ways to provide defense against such threats, however it is a cumbersome process which requires many data points and long computation times. In a setting where only small amounts of data are available for this process, adversarial training may negatively impact the classification performance on clean images by overfitting on the small amount of data. This would be undesirable, especially when a large pretrained model with satisfactory performance on clean data is already available. We propose a new strategy to make a previously-trained model more robust against adversarial attacks, using scarce data and without degrading its performance on clean samples. The proposed strategy consists in freezing the parameters of the originally trained base model and adding small auxiliary networks along the architecture, which process the features to reduce the effect of any adversarial perturbation. This method can be used to defend a model against any arbitrary attack. A practical advantage of using auxiliary networks is that no modifications on the originally trained base model is required. Therefore, it can serve as a patch or add on to fix large and expensive existing deep learning models with little additional resources. Experiments on the CIFAR10 dataset showed that using only 10% of the full training set, the proposed method was able to adequately defend the model against the AutoPGD attack while maintaining a classification accuracy on clean images outperforming the model with adversarial training by 7%. Indeed, the proposed method still performs reasonably well compared to adversarial training using 1% of the full training set.

1 INTRODUCTION

Nowadays Deep Neural Networks (DNNs) have become the prominent kind of machine learning model used in most fields. These models have millions of parameters and, with sufficient amounts of labeled data, they usually provide satisfactory results for most tasks, such as image classification or detection. With their increasing popularity and promising performances, these models are starting to be deployed in a variety of real-world applications ranging from self-driving cars to healthcare. This calls for greater robustness to previously unseen data and potential threats from a malicious agent. In this regard, it has been discovered that DNNs are vulnerable to specifically-crafted variations in the input data, known as adversarial attacks (Szegedy et al., 2014), which are almost invisible to a human observer but can cause dramatic changes to the prediction of a model. Image classification models appear to be particularly vulnerable to this hazard and they are the main focus of this manuscript.

Adversarial training (Goodfellow et al., 2015) is a well-known method for training models to be robust against adversarial examples. However, it is fairly expensive as it generally requires big quantities of data and substantial computational power. These conditions may not be readily satisfied in every setting. Assume that a *base* model is given, trained on clean images to satisfactory performance and arbitrarily big. If only limited data is available for adversarial training, the process may degrade the ability of the base model to classify clean samples by overfitting on the new small dataset.

This work proposes a strategy to prevent the model from losing performance on clean images while learning a defense mechanism against adversarial attacks with few available data. This is achieved by modifying the architecture of the base model by adding one or more small networks (called fixer modules), trained to enhance the defense capability of the model. A fixer module takes as input a tensor of feature maps and processes them before adding them back to the input in a "skip connection" fashion. A penalty term is introduced in the loss function with the aim of "fixing" the features when the input is an adversarial example, to make them as close as possible to their natural counterpart. The weights of the *base* model are frozen and only the parameters of the fixer modules are tuned during the training phase, which is carried out using both the clean images and their adversarial examples generated by a chosen attack. The choice of attack during training is arbitrary; in the reported experiments the focus was on AutoPGD (Croce & Hein, 2020), Carlini-Wagner (CW) (Carlini & Wagner, 2017b) and FGSM (Goodfellow et al., 2015).

Experiments were carried out on the CIFAR10 dataset and a dataset of medical images, a domain where model robustness is fundamental. Results on the CIFAR10 dataset showed that the proposed method was able to provide a satisfactory defense against the chosen attack, comparable to that adversarial training. Furthermore, when using smaller subsets of the full training set to learn a defense strategy, the proposed method provided a competitive defense with little degradation to the performance on clean images. In particular, when using 10% of the training data for adversarial training against the AutoPGD attack, the proposed method retained $\sim 7\%$ more accuracy on clean samples than adversarial training while also providing a slightly better defense against the same attack. Similarly, when using 1% of the training data, the proposed method achieved a defense performance comparable to standard adversarial training with a positive difference of $\sim 18\%$ on the classification of clean samples. All the implementations have been done using the Tensorflow 2 library (Abadi et al., 2016).

The rest of the work is organized as follows: section 2 provides a short summary of related works in the field of adversarial threats; section 3 presents the proposed method in detail; section 4 describes the experimental setup and the results obtained across a number of experiments; finally section 5 concludes the work.

2 Related works

The concept of adversarial examples was first introduced by Szegedy et al. (2014), highlighting the vulnerability of DNNs to these specially crafted inputs. Given a model g described by the set of parameters $\boldsymbol{\theta}$, the objective of an adversarial attack is to find a small enough perturbation ξ such that the input $\boldsymbol{x} + \xi$ would be misclassified by the model while keeping a natural appearance to the human eye. Such attacks vary based on the knowledge that is assumed is possessed by the attacker and their objective. Most adversarial attacks are *untargeted*, meaning that the objective is solely to cause a misclassification. *Targeted* attacks on the other hand aim to change the classification of a given input to a specific label or value. In general, perturbations are computed singularly for each input. However, there is also ongoing research on *universal* adversarial perturbations (Zhang et al., 2021a); i.e. model-specific perturbations (rather than sample-specific) that would cause a misclassification when applied to any sample from the domain of the classifier. Concerning the knowledge of the attacker, attacks are generally divided in *white-* and *black*-box attacks (Papernot et al., 2017; Brendel et al., 2018; Guo et al., 2019; Kotyan & Vargas, 2020); the former assuming that the attacker has full access to the architecture, parameters and output of the model, the latter constraining the attacker only to the knowledge of the output of the model. FGSM (Fast Gradient Sign Method) (Goodfellow et al., 2015) is one of the earliest and most widespread white-box attacks, using the sign of the gradients of the output to generate a perturbation. The idea was further refined as an iterative process in the PGD (Projected Gradient Descent) attack (Madry et al., 2019). To reduce the dependence of the attack from hyperparameters, a parameter-free version of this attack, known as AutoPGD, was introduced in Croce & Hein (2020). Other notable white-box attacks are DeepFool (Moosavi-Dezfooli et al., 2016), which approximates the classifier to a linear decision boundary, and the Carlini-Wagner (CW) attack (Carlini & Wagner, 2017b), an iterative method aiming at minimizing a certain objective function dependent on the model's output logits under specified constraints.

To defend against these attacks, the most direct method is adversarial training (Goodfellow et al., 2015), which consists in training a model using the adversarial examples as training samples. This method is known to be quite robust but is very data-dependent and requires substantial computational power as adversarial examples are continuously computed during training. The source of the attack chosen for training also influences the computational complexity of this method. Some defense methods attempt to preprocess inputs to reduce the perturbation introduced by the attacks (Samangouei et al., 2018; Song et al., 2018) or to detect adversarial examples (Carlini & Wagner, 2017a). Transformation-based methods (Xie et al., 2018; Prakash et al., 2018) apply a transformation to the input to deflect the attack while maintaining the performance on uncorrupted images. In particular, the work by Kou et al. (2020) takes advantage of those by generating a distribution of the logits from the transformed samples and using those to train a robust distribution classifier without needing to generate adversarial examples at training time. To reduce the need for computationally expensive adversarial examples, other methods propose building models intrinsically hard to attack by a deeper understanding of their vulnerabilities (Papernot et al., 2016b; Li et al., 2019; Zhang et al., 2021b; Paknezhad et al., 2021). Similarly to this work, in Xie et al. (2019) the architecture is modified with small modules to denoise the features extracted by the model rather than preprocessing the input features. Overall, the research community has been tackling the problem of adversarial vulnerability in many different ways. As the field evolves, guidelines are being more clearly defined (Carlini et al., 2019) and open-source libraries with multiple implementations of attacks and defenses play a pivotal role (Papernot et al., 2016a; Nicolae et al., 2018).

3 METHODS

3.1 FIXER MODULE

A deep learning base model $g(\boldsymbol{x}, \boldsymbol{\theta})$ is given, parametrized by the set of parameters $\boldsymbol{\theta}$ such that $g : \mathbb{X} \to \mathbb{Y}$. The model takes as input an image $x \in \mathbb{X} \subseteq \mathbb{R}^{D_0}$ and maps it to a vector $\hat{\boldsymbol{y}}^{(0)} = g(\boldsymbol{x}, \boldsymbol{\theta}) \in \mathbb{Y} \subset [0,1]^C$ representing the predicted class of the input, where D_0 is the dimension of the input image ($height \times width \times channels$) and C is the number of possible classes. The true value of the label is encoded by the one-hot vector $\boldsymbol{y} \in \{0,1\}^C$. The base model g can be decomposed in a series of b sequential blocks such that $g = g_1 \circ g_2 \cdots \circ g_b$ and $\hat{\boldsymbol{y}}^{(0)} = g_b(g_{b-1}(\cdots g_1(\boldsymbol{x}))))$. The base model architecture is enhanced by the addition of *fixer block*s (figure 1), placed in between each of the blocks (figure 2). These blocks aim to modify the learned feature maps to strengthen the model against adversarial attacks.

A fixer module is represented in figure 1; it is inserted in between the blocks g_i and g_{i+1}, where $i \in \{1, \ldots, b\}$ indicates a block of the base model. It takes as input a tensor of feature maps $\boldsymbol{h}_i^{(\delta)} \in \mathbb{R}^{D_i}$ and outputs the tensor $\boldsymbol{h}_i^{(\delta)} + \delta\phi_i(\boldsymbol{h}_i^{(\delta)}) \in \mathbb{R}^{D_i}$, where $\delta \in [0,1]$ is a scalar coefficient. The input tensor goes through a function $\phi_i : \mathbb{R}^{D_i} \to \mathbb{R}^{D_i}$ composed of two convolutional layers parametrized by $\boldsymbol{\theta}_{\phi_i}$: the first is a 1×1 convolution which reduces the number of channels by a factor ρ, the second is a 3×3 convolution with activation function ζ, which outputs a tensor with as many feature channels as the input tensor. These operations maintain the same spatial dimensions of the input tensor. The output of the fixer module is computed by adding the input $\boldsymbol{h}_i^{(\delta)}$ to $\phi_i(\boldsymbol{h}_i^{(\delta)})$ scaled by the scalar coefficient δ. In summary, the following relations exist among these variables

$$\boldsymbol{h}_{i+1}^{(\delta)} = g_{i+1}(\boldsymbol{h}_i^{(\delta)} + \delta\phi_i(\boldsymbol{h}_i^{(\delta)})) \tag{1}$$

A choice of $\delta = 0$ yields the unmodified base model g. Thus, a tensor $\boldsymbol{h}_i^{(0)}$ indicates the output of the i-th block obtained when all the δ are set to 0, which is equivalent to the output of the i-th block in the base model.

The base model g, modified with the addition of fixer modules is indicated by the symbol $g_\phi : \mathbb{X} \to \mathbb{Y}$ and $\hat{\boldsymbol{y}} = g_\phi(\boldsymbol{x}, \boldsymbol{\theta}, \boldsymbol{\theta}_\phi)$, where $\boldsymbol{\theta}_\phi = \{\boldsymbol{\theta}_{\phi_i}; \forall i \in \{1, \ldots, b-1\}\}$ is the set of parameters introduced by the fixer modules. A representation of a generic model split in $b = 3$ blocks and with the addition of fixer modules is shown in figure 2.

In the experiments, the base model architecture is that of a ResNet9, which has been split in $b = 3$ blocks at arbitrarily chosen points.

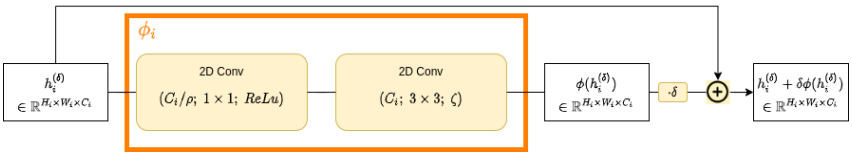

Figure 1: Representation of a generic fixer block and its operations. The input $\boldsymbol{h}_i^{(\delta)}$ is a tensor of dimensions $H_i \times W_i \times C_i$ where H_i and W_i are the spatial dimension and C_i is the number of feature maps (or channels). The parameter ρ, called compression factor, simply defines the compression of the feature channels carried out by the 1×1 convolution; in the experiments $\rho = 4$.

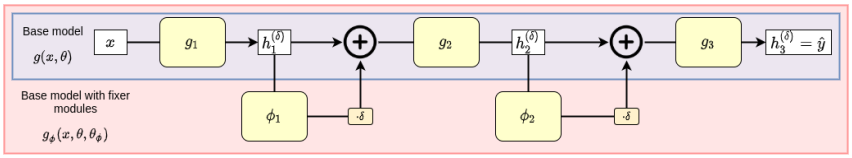

Figure 2: Representation of a model divided in $b = 3$ blocks with the addition of 2 fixer modules.

3.2 Training and loss function

When training g_ϕ, the core idea is to keep the ability of the base model to classify natural images and enhance it with robustness to adversarial examples. In this scenario, the base model has already been trained to a sufficient level of performance. To retain this, its parameters $\boldsymbol{\theta}$ are going to be frozen. The only trainable parameters will be those of the fixer modules, indicated by $\boldsymbol{\theta}_\phi$.

With this in mind the model g_ϕ is trained with the following strategy, exemplified for one generic sample $\boldsymbol{x}$ with label $\boldsymbol{y}$: for each natural image $\boldsymbol{x}$, a chosen attack is executed against the base model g to generate the corresponding adversarial sample $\boldsymbol{x}_{adv}$; i.e.

$$\boldsymbol{x}_{adv} = \texttt{any\_attack}(\boldsymbol{x}, g) \tag{2}$$

The attack function can be any attack from the literature. In the experiments AutoPGD (Croce & Hein, 2020) was used as the main attack during training phase due to its strength and parameter-free nature. Additional experiments have also been conducted using FGSM Goodfellow et al. (2015) and Carlini-Wagner Carlini & Wagner (2017b) (figure 6).

Both natural and adversarial samples are used to update the weights $\boldsymbol{\theta}_\phi$ of the fixer modules. This is done using a loss function with two terms: a cross-entropy loss term l_{XE} (in particular the focal cross-entropy was used (Lin et al., 2017)) and a distillation term l_D, weighed by a parameter λ. The second penalty has the aim of making the features obtained from the adversarial sample closer to the features of the natural sample in the base model; i.e. the term $\delta\phi_i(\boldsymbol{h}_i^{(\delta)})$ should compensate the effect of the adversarial noise at a hidden feature level and be equal to zero when the sample is not adversarial. To achieve this a distance metric has to be defined, in this work the standard ℓ_2 norm was used. In summary, given a sample $\boldsymbol{x}$, the true label $\boldsymbol{y}$ and its adversarial example $\boldsymbol{x}_{adv}$ (eq. 2), the objective is to minimize the loss

$$\min_{\boldsymbol{\theta}_\phi} \quad l_{XE}(\boldsymbol{y}, g_\phi(\boldsymbol{x})) + l_{XE}(\boldsymbol{y}, g_\phi(\boldsymbol{x}_{adv})) + \lambda l_D(\boldsymbol{x}, \boldsymbol{x}_{adv}) \tag{3}$$

$$\text{with} \quad l_D(\boldsymbol{x}, \boldsymbol{x}_{adv}) = \sum_{\forall i \in \{1,\dots,b-1\}} \ell_2(\boldsymbol{h}_i^{(\delta)}, \boldsymbol{h}_i^{(0)}) + \ell_2(\boldsymbol{h}_{i,adv}^{(\delta)}, \boldsymbol{h}_{i,adv}^{(0)}) \tag{4}$$

where $\boldsymbol{h}_i^{(\delta)}$ and $\boldsymbol{h}_i^{(0)}$ are obtained from equation 1 with $\boldsymbol{x}$ as an input to the model. Similarly, $\boldsymbol{h}_{i,adv}^{(\delta)}$ and $\boldsymbol{h}_{i,adv}^{(0)}$ are the output features of the i-th block when $\boldsymbol{x}_{adv}$ is the input to the model.

4 RESULTS

4.1 DATA

Experiments have been carried out on the CIFAR10 dataset (Krizhevsky et al., 2009) and a dataset of biomedical images (henceforth addressed as HIST). CIFAR10 (Krizhevsky et al., 2009) is a well-know collection of low-resolution ($32 \times 32 \times 3$) RGB natural images belonging to 10 categories. The dataset is provided with a split into training $\mathbb{D}_{tr}$ (45k images), validation $\mathbb{D}_{vl}$ and test $\mathbb{D}_{ts}$ sets. In this work, the training set has been further sub-sampled in datasets that have 10% and 1% of its samples, $\mathbb{D}_{tr}^{[10\%]}$ and $\mathbb{D}_{tr}^{[1\%]}$ respectively. Additional experiments were carried out using a dataset of medical images, a domain where data and computational power are often scarce. Limited attention is given to biomedical images in the adversarial domain Bortsova et al. (2021); Ma et al. (2021), which could be a prime target for malicious agents. The HIST data is made up of 128x128 RGB patches cropped from the TCGA-LUAD slides (Albertina et al., 2016) at 10x zoom level (1 micrometer/pixel at specimen-level). Each patch is assigned a label in one of four classes, based on the region from where it comes in the whole slide: artifact, benign, malignant, and other. Data is segregated at the patient level into "training" (20k patches) and "test" (14k patches) sets, meaning that the patches cropped from a single patient are either in one or the other set but never both to avoid patient-level data leakage Oner et al. (2020). Similar to the CIFAR10 dataset, the training set has been sub-sampled to 10% and 1% to investigate the efficacy of the various methods with less available data.

During the training phase, the input images are subject to a series of augmentation procedures at each training iteration. Images are first padded on both sides, followed by a random crop of the original shape of the image. This is followed by random flips and solarization. Finally, random cutout of dimension 10×10 (DeVries & Taylor, 2017) is applied. This augmentation procedure is followed both for training the base model (without adversarial samples) and the further defense experiments. Do note that the augmentation procedure is applied *before* generating the adversarial sample; i.e. the adversarial sample is a corrupted version of the augmented sample. Additional augmentation details in section A.3.

4.2 THREAT DEFINITION

Experiments test the ability of the proposed method to defend from adversarial attacks carried out on the base model. It follows that the adversarial examples (both during training and testing) are generated using the base model as a target for the attack. This means that the assumption is being made that the attacker is unaware of the patch that has been applied on the base model. In the experiments, this assumption holds at all times (i.e. both training and test phase). During the training phase, adversarial examples can be generated using any attack of choice (as defined by Eq. 2). For the reported experiments, the AutoPGD attack (Croce & Hein, 2020) was used as main attack for generating adversarial samples during training, with additional experiments done with CW (Carlini & Wagner, 2017b) and FGSM (Goodfellow et al., 2015). Unless specified otherwise, the maximum allowed perturbation is $\epsilon = 8/255$. During testing, samples are generated using the attacks: AutoPGD with $\epsilon = 8/255$ (AutoPGD) and $\epsilon = 16/255$ (AutoPGD-16); Carlini & Wagner attack (CW) (Carlini & Wagner, 2017b); Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2015); Projected Gradient Descent (PGD) (Madry et al., 2019). All the attacks have been implemented using the Adversarial Robustness Toolkit (ART) library (Nicolae et al., 2018). Additional details on the parameters of these attacks can be found in the appendix section A.2.

4.3 BASE MODEL

The base model can be any arbitrary model as long as it can be split into subsequent function blocks. For our experiments, the ResNet9 architecture has been chosen (details in section A.4). This model has been trained on the full training set $\mathbb{D}_{tr}$ of natural images to satisfactory performance on clean images. Training is carried out using a focal cross-entropy loss function Lin et al. (2017) and a SGD implementing the OneCycle scheduler for the values of learning rate and momentum Smith (2018). The same model architecture is used in the experiments for both CIFAR10 and HIST datasets.

4.4 RESULTS ON CIFAR10

The base model trained on the CIFAR10 training set achieves an accuracy on $91.20 \pm 0.80\%$[1] on $\mathbb{D}_{ts}$. The same model performs at $4.6 \pm 0.40\%$ accuracy when the same dataset is corrupted through an AutoPGD attack. Further details on the performance of the base model can be found in table 1 and figure 4. All subsequent models are initialized with the weights of the base model.

Firstly an ablation study was carried out on some hyper-parameters of the proposed method, these are: the activation function of each ϕ_i ($\zeta \in \{$linear; tanh $\}$, see figure 1); the coefficient δ in equation 1 ($\delta \in \{0.1, 1\}$); the coefficient of the distillation term in the loss function equation 3 ($\lambda \in \{0, 1\}$). Furthermore, the amount of data used to defend the pre-trained model is also investigated; the subsets of the full training set, $\mathbb{D}_{tr}^{[10]}$ and $\mathbb{D}_{tr}^{[100]}$, are employed. The investigated hyperparameters are summarized in the appendix section A.1.

The plot in figure 3 summarizes the performance of the proposed method when trained on $\mathbb{D}_{tr}^{[10\%]}$ and using a variety of hyper-parameters. It appears that the choice of a low value of δ coupled with a hyperbolic tangent activation function does not work well, showing low values when defending against the threats. The low activation values probably made the training harder for the model, which appears to have retained a behavior mostly similar to the base model. A higher value of the activation appears to provide better results both for natural and corrupted images. In most instances, the choice of $\lambda = 1$ yielded a more robust model than its counterpart with $\lambda = 0$. As expected, the choice of $\lambda = 1$ reduced the degradation of performance on clean samples. Overall, the combination of "fx-a" (linear, $\delta = 1$, $\lambda = 1$) appears to perform the best: the linear activation function coupled with $\delta = 1$ allow for a wider range of behaviors for compensating the effect of the adversarial perturbation and the $\lambda = 1$ coefficient aids in maintaining the performance on clean samples.

The model was compared to other defense strategies available in the literature. The standard adversarial training (adv train) Goodfellow et al. (2015) fine-tunes the whole model by generating adversarial examples of the samples in the dataset and training the model on those. For a fair comparison with the proposed method, the model trained with adversarial training is also shown the pair of clean and adversarial images at each epoch. The method proposed by Xie et al. (2019) (denoise) proposes to add denoising blocks along the architecture in a skip-connection fashion. This method has been implemented by substituting the "fixer blocks" with "denoising blocks" as described in the original work; in particular, the non-local means without embedding were chosen as denoising filters. Lastly the distribution regression network (drn) proposed by Kou et al. (2020) was also employed. This method uses a transformation strategy to generate several distorted versions of the input sample and through Gaussian KDE estimates the distribution of their softmax logits. A Distribution Regression Network (DRN) (Kou et al., 2019) is then used to classify the distribution into a target class. The implementation used in this work employs the Random Resize and Padding (RRP) (Xie et al., 2018) transformation strategy with $N = 50$ transformations and the same DRN network architecture as (Kou et al., 2020). These two methods have been implemented by using the respective repositories available online as a base, making the necessary changes to fit the different coding backends. While Xie et al. (2019) does not report performance on CIFAR10 data, the results using the DRN classifier appear consistent with those reported in Kou et al. (2020).

Figure 4 and table 1 report the performance of the proposed method and the defenses chosen from the literature when the data available for training is a subset of 10% of the full training set which was used to train the base model. Similarly, the plots in figure 5 show how these methods perform when varying the amount of available training data. The proposed method (fx-a) satisfies well the desiderata of low degradation of performance on natural images when using less training data. This is particularly apparent when looking at the classification accuracy on the samples of the test set $\mathbb{D}_{ts}$ that were correctly classified by the base model when clean. The degradation is comparable to the "drn" method on clean uncorrupted samples, however, the model performs way better on AutoPGD adversarial examples, successfully defending 95% of this subset.

With the lower number of trainable parameters and the constraints imposed by the loss function, the proposed method appears to learn how to properly defend against the AutoPGD (against which it is trained) when using fewer data points. However, the limitations imposed made the proposed method

[1]The values of mean and standard deviation are obtained by splitting the test set into 5 parts and averaging the results on each of them.

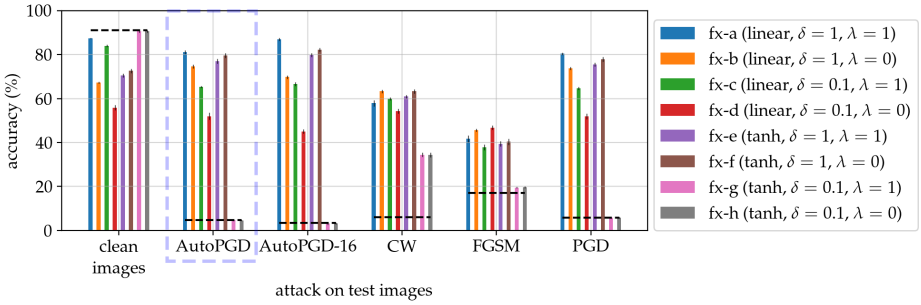

Figure 3: Ablation study. Accuracy computed on the test set of CIFAR10 under different attacks. All models have been trained using $\mathbb{D}_{tr}^{[10\%]}$ (10% of the full training set). In the legend the triplet in brackets represents activation function ζ, δ (coefficient multiplying the output of each ϕ_i) and λ (coefficient of the distillation term in the loss function). The black dashed line is the performance of the base model. Error bars are smaller than the size of the symbols.

	$\mathbb{D}_{ts}$			subset of $\mathbb{D}_{ts}$ correctly classified by *base*		
model	clean images	AutoPGD	FGSM	clean images	AutoPGD	FGSM
base	*91.20±0.80*	*4.60±0.40*	*17.10±0.30*	*100.00±0.00*	*0.00±0.00*	*13.00±0.60*
fx-a	87.50±0.20	**81.10±0.80**	41.70±1.50	93.80±0.40	**95.10±0.40**	45.20±1.00
fx-b	67.30±0.50	74.60±0.90	45.60±0.60	71.40±0.40	75.70±1.30	48.40±1.20
adv train	79.50±0.80	77.50±0.50	**72.10±0.40**	84.40±0.40	79.40±0.70	**75.40±0.70**
denoise	75.60±0.50	73.20±0.90	69.40±0.70	80.30±0.40	77.30±0.80	72.90±0.50
drn	**88.40±0.40**	13.90±0.80	39.90±1.00	**94.80±0.10**	10.10±0.30	37.80±1.10

Table 1: Accuracy on the CIFAR10 test set under different attacks. Comparison between the proposed method (fx-a; fx-b), traditional adversarial training (adv train), feature denoising blocks (dns) and distribution regression network (drn). All models are trained using $\mathbb{D}_{tr}^{[10\%]}$ (except for *base*, which is trained on $\mathbb{D}_{tr}$). The columns to the right report the accuracy on the subset of samples that are correctly classified by the base model when clean.

less prone to learn a general defense strategy, which instead happens for the traditional adversarial training. In other words, the proposed method does not appear to have learned a defense mechanism that generalizes as well to other attacks.

For further comparison, the experiments have been repeated using other attacks during the training phase other than AutoPGD. In particular, experiments have been done generating adversarial samples with FGSM and CW. The results are summarized in figure 6 when using 10% of the full training set. Do note that the base method and "drn" do not require the generation of adversarial samples during training, thus their performance is simply repeated in the different plots. Results show that using AutoPGD for training tends to degrade more the performance on clean samples than when using FGSM. In both cases, our method succeeds in having just a small degradation on this performance. When training on FGSM our method learns a reasonably good defense against the same attack, better than traditional adversarial training. In this experiment, the denoising blocks method "dns" appears to learn a defense strategy that transfers well to other attacks, as was observed also observed when training on AutoPGD. Surprisingly, the same transferability is not observed in the case where CW attack is used for training of "dns". Training on this attack also appears less transferable for all the other methods; all of which have learned successful defense strategy against the threat and also retained good performance on the clean samples. That would seem to indicate that, given the choice of hyperparameters, the adversarial examples generated by the CW attack are very similar to the clean images, thus generalize less to noisier attacks. At the same time, the adversarial examples generated by CW were diverse enough from the original and succeeded in degrading the base model performance to $\sim 6\%$ accuracy.

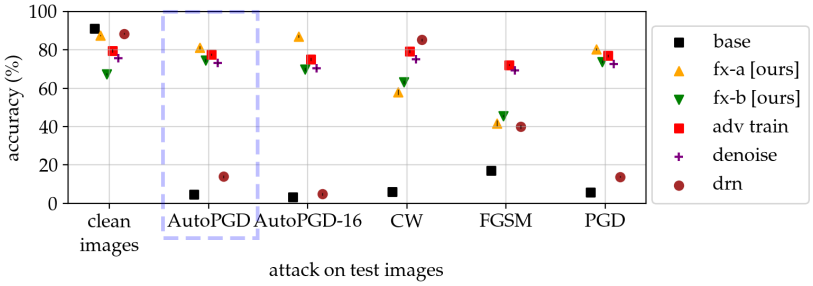

Figure 4: Accuracy on the CIFAR10 test set under different attacks. Comparison between the proposed method (fx-a; fx-b), traditional adversarial training (adv train) (Goodfellow et al., 2015), feature denoising blocks (denoise) (Xie et al., 2019) and distribution regression network (drn) (Kou et al., 2020). All models have been trained using $\mathbb{D}_{tr}^{[10\%]}$ and adversarial samples generated using AutoPGD where necessary (highlighted by dashed box). *Base* has been trained on the full training set of clean images. Error bars are smaller than the size of the symbols.

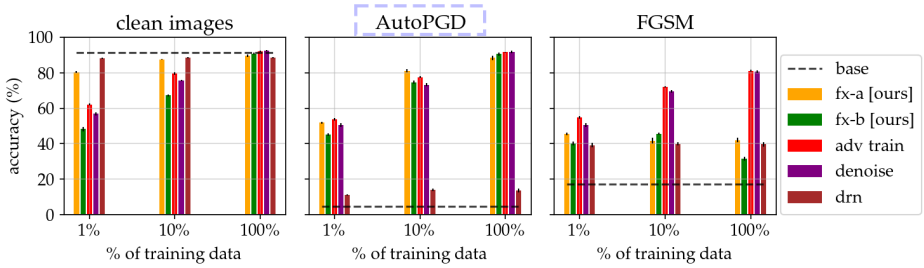

Figure 5: Accuracy on the CIFAR10 test set under different attacks varying the amount of data available for training. Comparison between the proposed method (fx-a; fx-b), traditional adversarial training (adv train), feature denoising blocks (denoise) and distribution regression network (drn). AutoPGD (highlighted by dashed box) has been used to generate adversarial samples during training. Error bars are smaller than the size of the symbols.

4.5 Results on medical images

Additional experiments have been carried out on the HIST dataset, described in section 4.1. Results are summarized in figure 7. The base model ResNet9 trained on this dataset achieves an accuracy of $92.70 \pm 0.50\%$ on unseen data, showing adequate performance on clean samples. However, the model appears vulnerable to adversarial data dropping its accuracy to $5.40 \pm 0.30\%$ when attacked by AutoPGD and $7.00 \pm 0.30\%$ by FGSM. For this dataset, the proposed method (fx-a, f-b) and standard adversarial training (adv train) appear to have similar performance in most cases. When using 10% of the data, the methods maintain a good performance on clean data and learn a successful strategy against AutoPGD (against which they were trained). Unlike for the CIFAR10 dataset, the proposed method appears to have learned a defense strategy that is adequately transferable to the FGSM attack, performing slightly worse than adversarial training. The gap in the transferability is more evident when using 1% of the data. However, in this scenario, the proposed method (fx-a) was able to retain a slightly better performance on clean samples. The differences in behavior compared to the CIFAR10 experiments may be due to the nature of the HIST data and the simplicity of classifying its 4 classes (versus the 10 in CIFAR10).

5 Conclusion

With the rapid diffusion of Deep Learning models in many industry sectors, it is fundamental to provide models that are robust against external threats. Adversarial attacks are particularly worrying

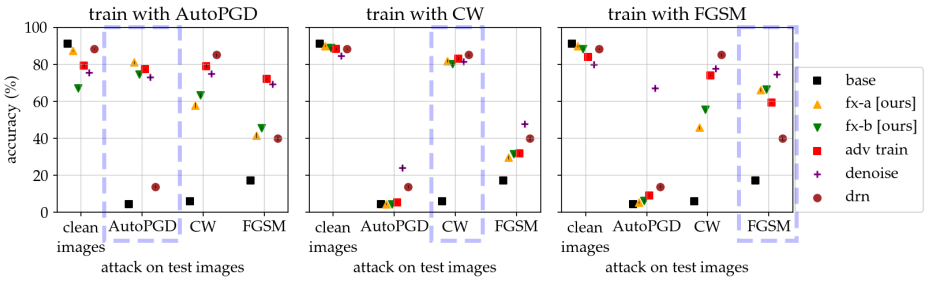

Figure 6: Performance on the CIFAR10 test set under different attacks when changing the attack used during training phase. Note that methods *base* and *drn* do not require generating adversarial samples, thus their performance is the same across the plots. A dashed box highlights the attack used for generating the adversarial samples in each plot. All models have been trained with $\mathbb{D}_{tr}^{[10\%]}$. Error bars are smaller than the size of the symbols.

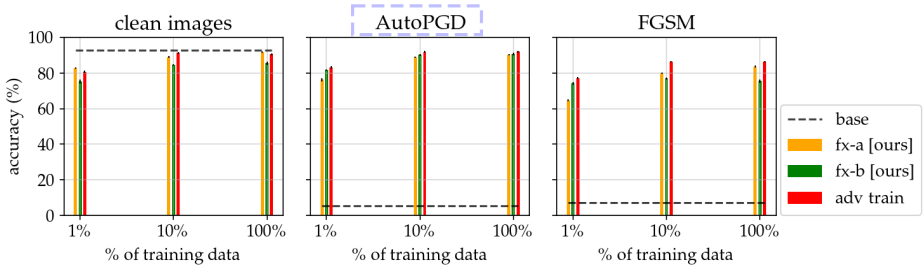

Figure 7: Accuracy on the histopathology (HIST) test set under different attacks varying the amount of data available for training. Comparison between the proposed method (fx-a) and standard adversarial training (adv train). All methods have been trained using AutoPGD (highlighted by dashed box) to generate the adversarial samples. Error bars are smaller than the size of the symbols.

due to their ability to fool a human observer. The "adversarial training" defense strategy is conceptually simple and provides a good performance against most attacks. However, it is an expensive process that requires many examples and updating all the parameters of a model. This strategy may not be always applicable, especially when a model is very big and needs to be patched with a limited set of new data. Carrying out adversarial training on a scarce set of samples may increase the ability to defend against an attack but also degrade the performance on clean samples.

To address this problem, this work proposes a strategy to defend against adversarial threats using a small amount of data and maintaining good performance on clean samples. A base model is given with satisfactory performance on clean images in the chosen domain. The proposed method consists in slightly modifying the base model with the addition of some small "fixer" networks that process the features to reduce the adversarial perturbations. During training, pairs of clean images and their adversarial examples are used to update the parameters of the fixer modules, while the weights of the base model stay frozen. With this strategy, the base model remains mostly unmodified and the performance on clean images is retained even when using a small dataset to learn a defense mechanism. Experiments on the CIFAR10 dataset showed the competitiveness of the method compared to adversarial training and two other defense methods from the literature. In particular, when using smaller portions of the full training set (namely 10% and 1%), the proposed method was able to have a minimal degradation of the performance on clean images and to provide a good defense against adversarial samples when compared to traditional adversarial training. However, results also showed that the defense mechanism learned by the proposed method appears to be more specific to the type of attack that was used during training and generalizes less to other types of attacks.

AUTHOR CONTRIBUTIONS

Add in de-anonimized version.

ACKNOWLEDGMENTS

Add in de-anonimized version.

REFERENCES

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pp. 265–283, 2016.

Brad Albertina, Mark Watson, Chandra Holback, Rose Jarosz, Shanah Kirk, Yueh Lee, Kimberly Rieger-Christ, and John Lemmerman. Radiology Data from The Cancer Genome Atlas Lung Adenocarcinoma [TCGA-LUAD] collection, 2016.

Gerda Bortsova, Cristina González-Gonzalo, Suzanne C. Wetstein, Florian Dubost, Ioannis Katramados, Laurens Hogeweg, Bart Liefers, Bram van Ginneken, Josien P. W. Pluim, Mitko Veta, Clara I. Sánchez, and Marleen de Bruijne. Adversarial Attack Vulnerability of Medical Image Analysis Systems: Unexplored Factors. *Medical Image Analysis*, pp. 102141, June 2021. ISSN 1361-8415. doi: 10.1016/j.media.2021.102141.

Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models. *arXiv:1712.04248 [cs, stat]*, February 2018.

Alexander Buslaev, Vladimir I. Iglovikov, Eugene Khvedchenya, Alex Parinov, Mikhail Druzhinin, and Alexandr A. Kalinin. Albumentations: Fast and flexible image augmentations. *Information-an International Interdisciplinary Journal*, 11(2), 2020. ISSN 2078-2489. doi: 10.3390/info11020125.

Nicholas Carlini and David Wagner. Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, AISec '17, pp. 3–14, New York, NY, USA, November 2017a. Association for Computing Machinery. ISBN 978-1-4503-5202-4. doi: 10.1145/3128572.3140444.

Nicholas Carlini and David Wagner. Towards Evaluating the Robustness of Neural Networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 39–57, May 2017b. doi: 10.1109/SP.2017.49.

Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, Aleksander Madry, and Alexey Kurakin. On Evaluating Adversarial Robustness. *arXiv:1902.06705 [cs, stat]*, February 2019.

Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. *arXiv:2003.01690 [cs, stat]*, August 2020.

Terrance DeVries and Graham W. Taylor. Improved Regularization of Convolutional Neural Networks with Cutout. *arXiv:1708.04552 [cs]*, November 2017.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. Adaptive Computation and Machine Learning. The MIT Press, Cambridge, Massachusetts, 2016. ISBN 978-0-262-03561-3.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. *arXiv:1412.6572 [cs, stat]*, March 2015.

Chuan Guo, Jacob R. Gardner, Yurong You, Andrew Gordon Wilson, and Kilian Q. Weinberger. Simple Black-box Adversarial Attacks. *arXiv:1905.07121 [cs, stat]*, August 2019.

Shashank Kotyan and Danilo Vasconcellos Vargas. Adversarial Robustness Assessment: Why both L_0 and L_∞ Attacks Are Necessary. *arXiv:1906.06026 [cs, stat]*, July 2020.

Connie Kou, Hwee Kuan Lee, Ee-Chien Chang, and Teck Khim Ng. Enhancing Transformation-Based Defenses Against Adversarial Attacks with a Distribution Classifier. In *Eighth International Conference on Learning Representations*, April 2020.

Connie Khor Li Kou, Hwee Kuan Lee, and Teck Khim Ng. A compact network learning model for distribution regression. *Neural Networks*, 110:199–212, February 2019. ISSN 0893-6080. doi: 10.1016/j.neunet.2018.12.007.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Bai Li, Changyou Chen, Wenlin Wang, and Lawrence Carin. Certified Adversarial Robustness with Additive Noise. *arXiv:1809.03113 [cs, stat]*, November 2019.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal Loss for Dense Object Detection. *arXiv:1708.02002 [cs]*, August 2017.

Xingjun Ma, Yuhao Niu, Lin Gu, Yisen Wang, Yitian Zhao, James Bailey, and Feng Lu. Understanding adversarial attacks on deep learning based medical image analysis systems. *Pattern Recognition*, 110:107332, February 2021. ISSN 0031-3203. doi: 10.1016/j.patcog.2020.107332.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards Deep Learning Models Resistant to Adversarial Attacks. *arXiv:1706.06083 [cs, stat]*, September 2019.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2574–2582, 2016.

Maria-Irina Nicolae, Mathieu Sinn, Minh Ngoc Tran, Beat Buesser, Ambrish Rawat, Martin Wistuba, Valentina Zantedeschi, Nathalie Baracaldo, Bryant Chen, Heiko Ludwig, Ian Molloy, and Ben Edwards. Adversarial robustness toolbox v1.2.0. *CoRR*, 1807.01069, 2018.

Mustafa Umit Oner, Yi-Chih Cheng, Hwee Kuan Lee, and Wing-Kin Sung. Training machine learning models on patient level data segregation is crucial in practical clinical applications, April 2020.

Mahsa Paknezhad, Cuong Phuc Ngo, Amadeus Aristo Winarto, Alistair Cheong, Beh Chuen Yang, Wu Jiayang, and Lee Hwee Kuan. Explaining Adversarial Vulnerability with a Data Sparsity Hypothesis. *arXiv:2103.00778 [cs]*, March 2021.

Nicolas Papernot, Ian Goodfellow, Ryan Sheatsley, Reuben Feinman, Patrick McDaniel, et al. Cleverhans v2. 0.0: An adversarial machine learning library. *arXiv preprint arXiv:1610.00768*, 10, 2016a.

Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a Defense to Adversarial Perturbations against Deep Neural Networks. *arXiv:1511.04508 [cs, stat]*, March 2016b.

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical Black-Box Attacks against Machine Learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, ASIA CCS '17, pp. 506–519, New York, NY, USA, April 2017. Association for Computing Machinery. ISBN 978-1-4503-4944-4. doi: 10.1145/3052973.3053009.

Aaditya Prakash, Nick Moran, Solomon Garber, Antonella DiLillo, and James Storer. Deflecting Adversarial Attacks with Pixel Deflection. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8571–8580, Salt Lake City, UT, June 2018. IEEE. ISBN 978-1-5386-6420-9. doi: 10.1109/CVPR.2018.00894.

Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-GAN: Protecting Classifiers Against Adversarial Attacks Using Generative Models. *arXiv:1805.06605 [cs, stat]*, May 2018.

Leslie N. Smith. A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay. *arXiv:1803.09820 [cs, stat]*, April 2018.

Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Ermon, and Nate Kushman. PixelDefend: Leveraging Generative Models to Understand and Defend against Adversarial Examples. *arXiv:1710.10766 [cs]*, May 2018.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv:1312.6199 [cs]*, February 2014.

Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. Mitigating Adversarial Effects Through Randomization. *arXiv:1711.01991 [cs]*, February 2018.

Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan L. Yuille, and Kaiming He. Feature Denoising for Improving Adversarial Robustness. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 501–509, 2019.

Chaoning Zhang, Philipp Benz, Chenguo Lin, Adil Karjauv, Jing Wu, and In So Kweon. A Survey On Universal Adversarial Attack. *arXiv:2103.01498 [cs]*, March 2021a.

Jiyi Zhang, Ee-Chien Chang, and Hwee Kuan Lee. Confusing and Detecting ML Adversarial Attacks with Injected Attractors. *arXiv:2003.02732 [cs]*, March 2021b.

A APPENDIX

A.1 ABLATION STUDY PARAMETERS

The parameters investigated in the ablation study are summarized in table 2 the activation function of each ϕ_i ($\zeta \in \{\text{linear; tanh }\}$, see figure 1); the coefficient δ in 1 ($\delta \in \{0.1, 1\}$); the coefficient of the distillation term in the loss function equation 3 ($\lambda \in \{0, 1\}$).

A.2 ADVERSARIAL ATTACKS

In the experiments a number of adversarial attacks have been used to attack the models and/or generate adversarial examples during training. These are summarized in table. All attacks are untargeted. All the attacks have been implemented using the Adversarial Robustness Toolkit (ART) library (Nicolae et al., 2018).

During testing, samples are generated using additional attacks: AutoPGD with $\epsilon = 16/255$ (AutoPGD-16); Carlini & Wagner attack (CW) (Carlini & Wagner, 2017b); Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2015); Projected Gradient Descent (PGD) (Madry et al., 2019).

A.3 DATA AUGMENTATION

Data augmentation is implemented using the Albumentations library (Buslaev et al., 2020). During training, each sample is augmented by a series of transformations as described in table 3 for the CIFAR10 dataset and in table 4 for the HIST dataset. Note that the augmentation procedure is applied *before* generating the adversarial sample; i.e. the adversarial sample is a corrupted version of the augmented sample.

A.4 MODEL ARCHITECTURE

The architecture of the model used in the experiments is that of a ResNet9 defined as described in table 8. The building blocks of the model are defined in the preceeding tables: ConvBlock (table 5), ResidualBlock (table 6) and FixerModule (table 7). Implementation using Python 3 and Tensorflow 2 (Abadi et al., 2016).

parameter	**description**	**value**		
δ	coefficient to fixer module output in equation 1	0	0.1	1.
ζ	activation function of second convolution in ϕ (see figure 1)	linear	tanh	
λ	coefficient to distillation term in loss equation 3	0	1	
training dataset		$\mathbb{D}_{tr}$	$\mathbb{D}_{tr}^{[10\%]}$	$\mathbb{D}_{tr}^{[1\%]}$

Table 2: Summary of investigated hyperparameters and their values. Combinations of these parameters have been explored in the ablation experiments; not all combinations have been investigated, due to some being redundant.

ID	AutoPGD	AutoPGD-16	CW	FGSM	PGD
Attack	AutoPGD (Croce & Hein, 2020)	AutoPGD (Croce & Hein, 2020)	Carlini-Wagner (Carlini & Wagner, 2017b)	Fast Gradient Sign Method (Goodfellow et al., 2015)	Projected Gradient Descent (Madry et al., 2019)
Max perturbation ϵ	$8/255$	$16/255$	$8/255$	$8/255$	$8/255$
ϵ step	0.005	0.005	-	0.005	0.005
# iterations	25	25	25	25	50
Norm	ℓ_{inf}				
Confidence	-	-	0	-	-

transformation	**details**	**prob.**	**out shape**
Input	-	-	*(32, 32, 3)*
Padding	border mode = reflect position = center	1	(40, 40, 3)
Random Crop	-	1.	(32, 32, 3)
Vertical Flip	-	0.5	(32, 32, 3)
Horizontal Flip	-	0.5	(32, 32, 3)
Solarize	threshold = 0.5	0.3	(32, 32, 3)
Coarse Dropout (i.e. Cutout)	max holes = 1 height = 10 width = 10	0.5	(32, 32, 3)

Table 3: Augmention details for the CIFAR10 dataset.

transformation	**details**	**prob.**	**out shape**
Input	-	-	*(128, 128, 3)*
Padding	border mode = reflect position = center	1	(160, 160, 3)
Random Crop	-	1.	(128, 128, 3)
Vertical Flip	-	0.5	(128, 128, 3)
Horizontal Flip	-	0.5	(128, 128, 3)
Solarize	threshold = 0.5	0.3	(128, 128, 3)
Coarse Dropout (i.e. Cutout)	max holes = 8 height = 10 width = 10	0.5	(128, 128, 3)

Table 4: Augmentation details for the HIST dataset.

Name	**Operation Block**
input	**-**
conv	Conv2D(channels=*channels*, kernel_size=3x3)
bn	Batch Normalization
relu	ReLU
output	MaxPool2D(kernel_size=2) if *pool*=True Identity if *pool*=False

Table 5: Architecture of ConvBlock(*channels*, *pool*). Operation blocks are sequential.

Name	**Operation Block**
input	-
conv_block_a	ConvBlock(*channels*, False)
conv_block_b	ConvBlock(*channels*, False)
output	input + conv_block_b

Table 6: Architecture of ResidualBlock(*channels*, *pool*). Operation blocks are sequential.

Name	Operation Block
input	-
conv_1	Conv2D(channels=*channels*/ρ, kernel_size=1x1)
relu_1	ReLU
conv_2	Conv_2D(channels=*channels*, kernel_size=3x3, activation=ζ)
add	input + δ * conv_2
output	ReLU

Table 7: Architecture of FixerModule(*channels*, δ, ζ, ρ). Operation blocks are sequential. Refer to figure 1 for visual representation. In all the experiments $\rho = 4$.

Name	Operation Block
input	-
conv_block_1	ConvBlock(64, False)
conv_block_2	ConvBlock(128, True)
res_block_1	ResidualBlock (128, False)
fixer_mod_1	FixerModule(128, δ, ζ, ρ)
conv_block_3	ConvBlock(256, False)
conv_block_4	ConvBlock(512, True)
res_block_2	ResidualBlock (512, False)
fixer_mod_2	FixerModule(512, δ, ζ, ρ)
pool_4	MaxPooling2D(kernel_size=4)
flatten	Flatten if *dataset*=CIFAR10 GlobalAveragePooling2D if *dataset*=HIST
output	Dense(classes)

Table 8: Architecture of ResNet9 with hyper-parameters δ, ζ, ρ and *dataset*. In all the experiments $\rho = 4$. Operation blocks are defined in tables 5 to 7. Base model architecture can be obtained by removing the FixerModule*s* or equivalently setting $\delta = 0$.