Evaluation of Defense Methods Against the One-Pixel Attack on Deep Neural Networks

Victor Arvidsson[1], Ahmad Al-Mashahedi[2] and Martin Boldt[3]

Abstract— **The one-pixel attack is an image attack method for creating adversarial instances with minimal perturbations, i.e., pixel modification. The attack method makes the adversarial instances difficult to detect as it only manipulates a single pixel in the image. In this paper, we study four different defense approaches against adversarial attacks, and more specifically the one-pixel attack, over three different models. The defense methods used are: *data augmentation*, *spatial smoothing*, and *Gaussian data augmentation* used during both training and testing. The empirical experiments involve the following three models: all convolutional network (CNN), network in network (NiN), and the convolutional neural network VGG16.**

Experiments were executed and the results show that Gaussian data augmentation performs quite poorly when applied during the prediction phase. When used during the training phase, we see a reduction in the number of instances that could be perturbed by the NiN model. However, the CNN model shows an overall significantly worse performance compared to no defense technique. Spatial smoothing shows an ability to reduce the effectiveness of the one-pixel attack, and it is on average able to defend against half of the adversarial examples. Data augmentation also shows promising results, reducing the number of successfully perturbed images for both the CNN and NiN models. However, data augmentation leads to slightly worse overall model performance for the NiN and VGG16 models. Interestingly, it significantly improves the performance for the CNN model.

We conclude that the most suitable defense is dependent on the model used. For the CNN model, our results indicate that a combination of data augmentation and spatial smoothing is a suitable defense setup. For the NiN and VGG16 models, a combination of Gaussian data augmentation together with spatial smoothing is more promising. Finally, the experiments indicate that applying Gaussian noise during the prediction phase is not a workable defense against the one-pixel attack.

I. Introduction

Machine learning (ML), which is an important subarea of artificial intelligence (AI), has become both increasingly important and relevant during the last decades due to, for instance, its widespread use in critical applications. An important field within ML is *adversarial machine learning*, which is the study on how ML models can be attacked or deceived by antagonistic actors, i.e., adversaries [1]. Adversarial ML involves both the development of various attack methods against ML methods, as well as the development of defense methods against such attacks. These defense methods aim to improve the robustness of ML models [2].

Adversarial ML attacks are aimed either at the training data (e.g., data-poisoning attacks), the ML model's parameters, or the inputs during inference while using the ML model (e.g., adversarial input attacks) [3]. In general, such attacks can result in ML models that make incorrect predictions, which can result in serious consequences depending on the application, e.g., healthcare or autonomous vehicles.

In essence, this paper presents experimental results that evaluate defense methods against a particular adversarial ML attack referred to as the one-pixel attack [4]. We use the following three different models to evaluate the performance of the defenses: an all convolutional network (CNN), a Network in Network (NiN), and the convolutional neural network VGG16. The motivation for the choice of these three particular models is that those are the models used in the original paper describing the one-pixel attack [4]. In the experiments, two different defense methods are applied during the training phase of the models, while two different defense method are applied during the prediction phase.

The remainder of this paper is outlined as follows. Next, in Section II, the background is described, which includes the one-pixel attack and the applied defense methods. Then follows the related work in Section III, and then the methods description in Section IV. The results are presented in Section V followed by the analysis and discussion in Section VI. Finally, conclusions and future work are described in Section VII.

II. Background

In the background section, we describe the one-pixel attack as well as the defense methods evaluated in this study.

A. One-Pixel Attack

The one-pixel attack is an iterative semi-black-box attack that targets image recognition models. The main idea is to only perturb, i.e., modify, the value of one single pixel in an image to make the model miss-classify the whole image. This makes the attack harder to detect for humans when used on larger images. It also demonstrates that current models are not robust enough to ignore small adversarial perturbations [4].

B. Differential Evolution

In order to execute the one-pixel attack, differential evolution is used. Differential evolution is an evolutionary algorithm that minimizes a function value by creating candidate

*This work was not supported by any organization

[1]V. Arvidsson is a master's student at the Department of Computer Science, Blekinge Institute of Technology, Karlskrona, Sweden h.victor.arvidsson at gmail.com

[2]A. Al-Mashahedi is a master's student at the Department of Computer Science, Blekinge Institute of Technology, Karlskrona, Sweden ahmad.sebbah at gmail.com

[3]M. Boldt is a researcher at the Department of Computer Science, Blekinge Institute of Technology, Karlskrona, Sweden martin.boldt at bth.se

solution vectors and evaluating their fitness on a function. The method has three different parameters: the population size, $NP \geq 4$, the crossover probability, $CR \in [0, 1]$, and the differential weight, $F \in [0, 2]$. For each generation, each candidate solution, x, in that generation is mutated using three other distinct candidates solutions, a, b, and c, in combination with the F and CR parameters. A mutation vector is calculated according to Equation 1:

$$a + F \cdot (b - c). \tag{1}$$

For each dimension index, j in the mutation vector, a uniform random number $n \in [0, 1]$ is generated. A random integer index $R \in [1, NP]$ is also generated. If $n < CR$ or $j = R$, the value from the mutation vector at index j is selected for the new candidate, otherwise the value of x at index j is selected. The new candidate solution is compared to the old candidate x, and if it has a better performance it is added to the population as a replacement for x in the new generation. This continues for a specified number of generations, or until a stop criteria is met [5].

C. Defenses

For this report, we will evaluate three different defense methods. These are data augmentation, spatial smoothing, and Gaussian data augmentation. Some of these can be combined, which is discussed further in Section IV.

1) Data Augmentation: Data augmentation is used to increase the size of a dataset by adding slightly altered versions of the existing data points. This can increase the robustness of the trained model, as well as reduce overfitting. For images, this is usually done by applying an affine transformation, i.e., a linear transformation with translation, to the images. This may include rotating, translating, shearing, mirroring, and zooming the images [6].

2) Spatial Smoothing: Spatial smoothing is a feature squeezing method that reduces noise in the image by blurring it. There are two types of spatial smoothing: local and non-local. This report will focus on the local variant. Local spatial smoothing works by using information from nearby pixels to smooth each pixel. A sliding window passes over the image and updates each pixel according to a weighted kernel. This can perform different types of smoothing, such as Gaussian or median smoothing [7].

3) Gaussian Data Augmentation: The idea behind Gaussian data augmentation is to apply Gaussian noise to the inputs to the model. Gaussian data augmentation can be applied both during the training phase and the prediction phase. If used during the training phase, the training data is perturbed with Gaussian noise, similar to how regular data augmentation works. The dataset can either be augmented with the new samples, or replaced by them. If used during the prediction phase, the Gaussian noise is added to the input before it is passed into the model for prediction [8].

III. Related Works

In this section, the related work is summarized and the identified research gap is stated.

First, Bracamonte *et al.* proposed a novel approach, OPA2D, which is an extended one-pixel attack approach that aims to deceive humans and DNNs [9]. Further, they proposed to limit the attacked pixel RGB range in order to make it harder to detect by human vision. They show that an already attacked image, if attacked once again, tends to return to its original label. The results they achieve were good with detection rates up to 100% and defense rates between 93%-95%.

Husnoo *et al.* suggest an approach that utilizes robust principle component analysis and accelerated proximal gradient to detect the attacked pixel and recover it from the image [10], thus creating a clean non-attacked image with no deterioration in image quality.

Chen *et al.* proposed a Patch Selection Denoiser (PSD) approach to remove potential attacking pixels from an image without changing a large number of the pixels in the image [11]. The proposed approach achieved a 98.6% defense rate against one-pixel attacks. However, it relies on patching images independent of whether an adversarial image was detected or not, which degrades images due to the use of the denoising model.

Bennamoun *et al.* proposed an adversarial detection network (ADnet) that can detect adversarial pixels in images for robotic systems [12]. The authors claimed that it works as a defense method by rejecting adversarial examples. The performance of their approach, in the context of detecting adversarial scenarios, was evaluated using three different datasets. In the evaluation they perturbed 50% of the images, using 1, 3, and 5-pixel attacks, in order to create adversarial examples for the attack scenarios. The results indicate that ADNet's efficacy in detecting adversarial N-pixel attacks across the three datasets were shown by a detection accuracy above 90% for all datasets and attack types.

Tso *et al.* proposed a three-stage noise elimination and reconstruction algorithm in which they remove N-attacked pixels and then reconstruct the image, while at the same time keeping its integrity [13]. Their approach is to construct a difference map to evaluate the difference between pixels, as well as an average map to correspond with it. If the difference between a pixel and its neighboring pixel is deemed too high, it is replaced by the value of its neighboring pixel. The approach can be seen as a pre-processing step, so no model re-training is needed. The experiment results they achieved reveal that the proposed algorithm provides a protection rate ranging between 90% to 92% against N-pixel attacks, for N values of 1, 3, 5, 10, and 15.

A. Identified Research Gap

Adversarial ML attacks have been shown to be rather effective in deceiving ML models, which has highlighted aspects regarding security and reliability within ML systems. Therefore, research in this area is essential to addressing the problems raised by adversarial ML attacks, and to mitigate the exploitation of such attacks by malicious actors. This motivates this study in which we evaluate different defense methods against the one-pixel attack, which is an attack that

Fig. 1. The result of applying the spatial smoothing and Gaussian noise filters on a sample image.

is difficult to detect due to its low degree of perturbation. Thus, the added value through this work is the evaluation of defenses against the one-pixel attack on the same three deep learning models on which the original attack was evaluated.

IV. Method

In this section we describe the method used, e.g., the dataset, models and their configurations, as well as the experimental setup. In essence, the experiments evaluate the suitability of defense methods against the one-pixel attack, during both the training and the prediction phase, using three different deep neural network models.

A. Dataset Used

For evaluation of the defense methods against the one-pixel attack, we use the CIFAR-10 dataset [14]. CIFAR-10 contains 60,000 images across 10 different classes. The dataset is provided as 50,000 training images and 10,000 test images. Each image is 32x32 pixels and each pixel has three color channels: red, green, and blue. Each channel has integer values in the range $[0, 255]$, normalized into the range $[0, 1]$. The motivation for choosing the CIFAR-10 dataset is two-fold, first that it is a commonly used dataset in applied ML vision research, and second that it was used in the original study presenting the one-pixel attack [4].

B. Models

To evaluate the defenses we use three different models. The first model is the all convolution network (CNN) [15]. This uses nine convolutional layers of different sizes. The second model is a Network-in-Network model (NiN) [16]. This also uses nine convolutional layers, but introduces pooling layers between every third layer. The third model is the VGG16 [17]. It is a convolutional model that uses 13 convolutional layers, five max pooling layers, and two fully connected layers. The structures of the networks can be seen in Tables I, II, and III. The three network models are identical to the models used in the original one-pixel attack paper [4].

TABLE I
Model Summary for All Convolutional Network (CNN).

Conv2D(filters=96, kernel_size=3, stride=1, activation=ReLU)
Conv2D(filters=96, kernel_size=3, stride=1, activation=ReLU)
Conv2D(filters=96, kernel_size=3, stride=2, activation=ReLU)
Conv2D(filters=192, kernel_size=3, stride=1, activation=ReLU)
Conv2D(filters=192, kernel_size=3, stride=1, activation=ReLU)
Dropout(0.3)
Conv2D(filters=192, kernel_size=3, stride=2, activation=ReLU)
Conv2D(filters=192, kernel_size=3, stride=2, activation=ReLU)
Conv2D(filters=192, kernel_size=1, stride=1, activation=ReLU)
Conv2D(filters=10, kernel_size=1, stride=1, activation=ReLU)
GlobalAveragePooling2D
Flatten
Softmax

TABLE II
Model Summary for Network in Network (NiN).

Conv2D(filters=192, kernel_size=5, stride=1, activation=ReLU)
Conv2D(filters=160, kernel_size=1, stride=1, activation=ReLU)
Conv2D(filters=96, kernel_size=1, stride=1, activation=ReLU)
MaxPooling2D(pool_size=3, stride=2)
Dropout(0.5)
Conv2D(filters=192, kernel_size=5, stride=1, activation=ReLU)
Conv2D(filters=192, kernel_size=5, stride=1, activation=ReLU)
Conv2D(filters=192, kernel_size=5, stride=1, activation=ReLU)
AveragePooling2D(pool_size=3, stride=2)
Dropout(0.5)
Conv2D(filters=192, kernel_size=3, stride=1, activation=ReLU)
Conv2D(filters=192, kernel_size=1, stride=1, activation=ReLU)
Conv2D(filters=10, kernel_size=1, stride=1, activation=ReLU)
GlobalAveragePooling2D
Flatten
Softmax

TABLE III
Model Summary for VGG16.

Conv2D(filters=64, kernel_size=3, stride=1, activation=ReLU)
Conv2D(filters=64, kernel_size=3, stride=1, activation=ReLU)
MaxPooling2D(pool_size=2, stride=2)
MaxPooling2D(pool_size=2, stride=2)
Conv2D(filters=128, kernel_size=3, stride=1, activation=ReLU)
Conv2D(filters=128, kernel_size=3, stride=1, activation=ReLU)
MaxPooling2D(pool_size=2, stride=2)
Conv2D(filters=256, kernel_size=3, stride=1, activation=ReLU)
Conv2D(filters=256, kernel_size=3, stride=1, activation=ReLU)
Conv2D(filters=256, kernel_size=3, stride=1, activation=ReLU)
MaxPooling2D(pool_size=2, stride=2)
Conv2D(filters=512, kernel_size=3, stride=1, activation=ReLU)
Conv2D(filters=512, kernel_size=3, stride=1, activation=ReLU)
Conv2D(filters=512, kernel_size=3, stride=1, activation=ReLU)
MaxPooling2D(pool_size=2, stride=2)
Conv2D(filters=512, kernel_size=3, stride=1, activation=ReLU)
Conv2D(filters=512, kernel_size=3, stride=1, activation=ReLU)
Conv2D(filters=512, kernel_size=3, stride=1, activation=ReLU)
MaxPooling2D(pool_size=2, stride=2)
Flatten
Dense(2048, activation=ReLU)
Dense(2048, activation=ReLU)
Dense(10, activation=Softmax)

C. Attack

The one-pixel attack uses differential evolution, and in this study we choose a population size of 400, a crossover probability of 1, and a differential weight that is uniformly randomized between 0.5 and 1 for each generation. These parameters where chosen based on the parameters in the original paper. Using a randomized differential weight can speed up convergence. Each candidate solution is an array of the following five values:

1) X coordinate for the perturbed pixel (between 0-31).
2) Y coordinate for the perturbed pixel (between 0-31).
3) Value for the red color channel (between 0-1).

TABLE IV
THE NINE DIFFERENT MODELS, BASED ON DATASET AND NETWORK ARCHITECTURES, THAT WERE EVALUATED.

		Models		
		CNN	**NiN**	**VGG16**
Dataset	**Original**	CNN_{orig}	NiN_{orig}	$\text{VGG16}_{\text{orig}}$
	Augmented	CNN_{aug}	NiN_{aug}	$\text{VGG16}_{\text{aug}}$
	Gaussian	CNN_{gau}	NiN_{gau}	$\text{VGG16}_{\text{gau}}$

4) Value for the green color channel (between 0-1).
5) Value for the blue color channel (between 0-1).

To evaluate each candidate solution, the image is perturbed with the candidate pixel, and the class is predicted by the classifier. We only perform untargeted attacks, and the goal is therefore to minimize the certainty of the model for the true label. The differential evolution runs for a total of 100 generation for each image. We include an early stop condition when the confidence for the true label is lower than 5%.

D. Defenses

We implement the four different types of defenses discussed in Section II. Two defenses applied during the training phase of the models, and two defenses applied during the testing phase.

1) Training-Phase Defenses: The first type of defenses are the ones that are applied during model training phase. These are the data augmentation and Gaussian data augmentation defenses, which are denoted CNN_{aug} and CNN_{gau} respectively for the CNN model. Both of these generate a new dataset with the augmented images, and for each of these a separate instance of each network type is trained. This results in a total of nine different models, as can be seen in Table IV. The applied augmentations for the data augmentation defense are: rotation up to 20° in each direction, width and height shift up to 20% of the image's width and height respectively, shearing with a maximum shear angle of 20°, zooming with a maximum range of 20% for both zoom-in and zoom-out, and finally mirroring along the vertical axis. For the Gaussian data augmentation, the noise is generated with a standard deviation of 0.05, and the samples were augmented at a ratio of 0.5, which means that the size of the dataset increases by 50%.

2) Testing-Phase Defenses: The testing-phase defenses are applied to each adversarial instance before the model performs the classification. Each of these defenses were applied separately to each instance. The spatial smoothing defense uses the median strategy for smoothing with a window size of 3. The Gaussian data augmentation applies Gaussian noise to the instance with a standard deviation of 0.05. Each testing-phase defense is tested separately and in combination with each training defense for each model.

E. Evaluation Metrics

To evaluate the performance of the models, we use both accuracy and Area Under the ROC Curve (AUC) score. For evaluation of the defense methods, we use normalized defense ratio, which is calculated according to Equation 2:

$$D_R = \frac{C_d}{C_p} \tag{2}$$

where C_d is the number of correctly classified instances after *both* the attack and defense methods were employed, while C_p is the number of correctly classified instances *before* the attack when only the defense was applied. To get the normalized defense ratio, we scale D_R by the accuracy of the model on the total attacked instances. Thus, the normalized defense ratio metric is calculated according to Equation 3:

$$\overline{D}_R = D_R \frac{C_p}{N} = \frac{C_d}{N} \tag{3}$$

where N is the total number of instances, in our case 1,000.

F. Experimental Setup

In the experimental evaluation of the models performance, the independent variable was the candidate models, i.e., CNN, NiN, and VGG16. The dependent variables were the accuracy and the AUC scores. For the experimental evaluation of the defense methods, the independent variables were the model candidates, the training-phase defense methods, and the testing-phase defense methods. The dependent variable was the defense ratio metric.

The three models that used the Gaussian augmentation defense (CNN_{gau}, NiN_{gau}, and $\text{VGG16}_{\text{gau}}$) and the three original models (CNN_{orig}, NiN_{orig}, and $\text{VGG16}_{\text{orig}}$) were trained during 100 epochs. Due to the computationally demanding proces to the generate and predict the image data, the models that used the augmented defense (CNN_{aug}, NiN_{aug}, and $\text{VGG16}_{\text{aug}}$) were trained for 70 epochs.

In total, 1,000 different images were randomly sampled from the test images in the CIFAR-10 dataset. The attack was performed individually on these 1,000 images for each of the three different network models included in the study and for each of the three different training datasets described in Section IV-D. This resulted in a total of 9,000 adversarial images that were included in the experiment.

The experiments were executed on a system with an Intel i7-6700K processor, Nvidia GTX 980 Ti graphics card and 16GB RAM.

V. RESULTS

The results are presented for the investigated models as well as the the one-pixel attack, including time measurements.

A. Model Performance

All models except the CNN_{gau} showed an AUC score above 0.9, with CNN_{aug} scoring best on both metrics with an AUC and accuracy of 0.982 and 0.85 respectively, see Fig. 2. Worst performance was associated with the Gaussian-augmented CNN model (CNN_{gau}) with an AUC and accuracy of 0.83 and 0.49 respectively. The remaining models show performance with AUC scores significantly above 0.9 and accuracy above 0.8 on average.

Fig. 2. Performance of the trained models on the entire CIFAR-10 test data set.

Fig. 3. Example of perturbed images, left image is from an attack on the CNN_{orig} model and the right one is from an attack on the $VGG16_{aug}$ model.

B. Results of the One-Pixel Attack

Two examples of attacked images and their associated classes are shown in Fig. 3. The prediction performance, one-pixel attack performance, and defense performance for three versions of the three models are shown in Fig. 4, 5, and 6. As an example, in Fig. 4, we can see that CNN_{orig} predicted the correct label for 801 images out of 1,000. The one-pixel attack was able to perturb 358 of these images so they were predicted with an erroneous class label. The spatial smoothing and Gaussian noise were able to defend against 203 and 160 out of the total 358 perturbed images respectively.

A complete overview of the results can be seen in the appendix.

It is worth noting that there were cases where the defense method caused the models to change their predictions from a correct label to an incorrect label, despite the attack not being successful. These cases are not presented since they are out of scope of this paper.

C. Execution Times

The process of running differential evolution for a single model takes approximately 14 seconds on average, for each unique image. With a total of nine models to evaluate the attack on results in a total execution time of 126 seconds, i.e., roughly two minutes, per unique image. For the 1,000 images the one-pixel attack process took around 35 hours to execute.

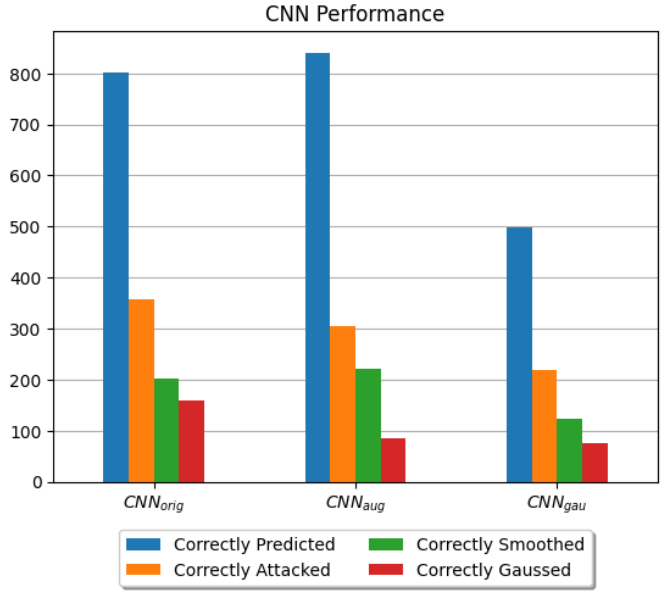

Fig. 4. Performance of the CNN models.

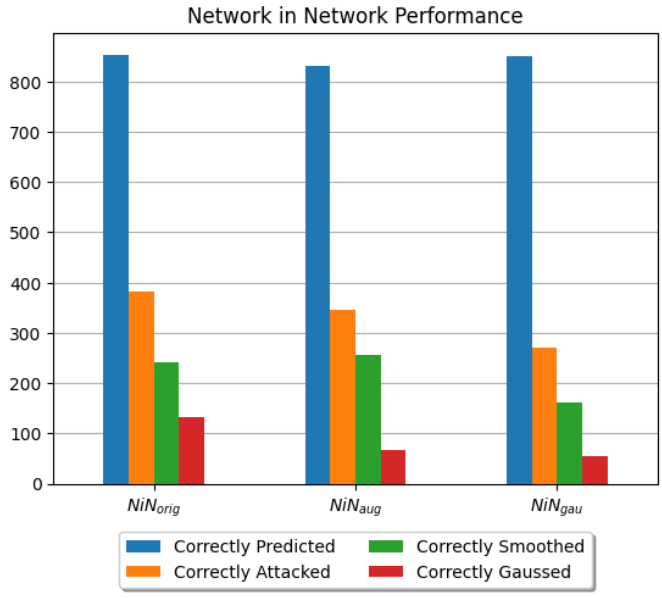

Fig. 5. Performance of the NiN models.

VI. Discussion

In this section, we will discuss the results for the original base models, for the different defenses, and for the one-pixel attack. The section is finished with a subsection that addresses validity threats connected to the experiments.

A. Base Models (NiN_{orig}, CNN_{orig}, and $VGG16_{orig}$)

The performance of the original models, CNN_{orig}, NiN_{orig}, and $VGG16_{orig}$, differed slightly according to the accuracy metric with 80%, 85% and 81% respectively, see Table V

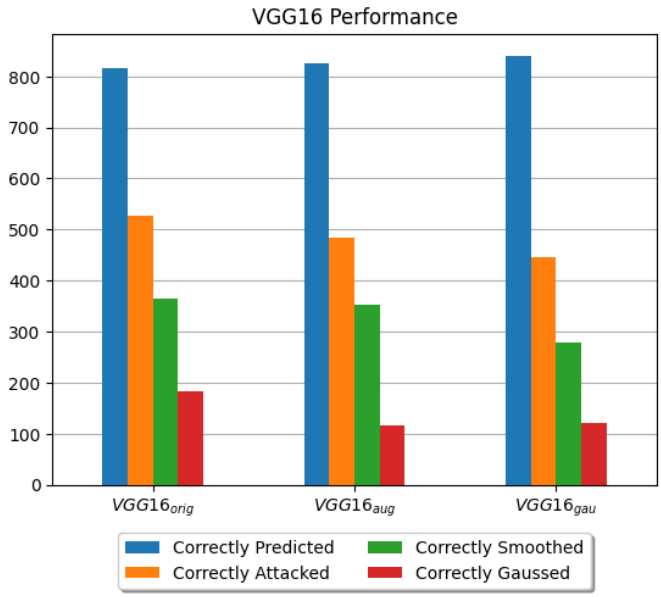

Fig. 6. Performance of the VGG16 models.

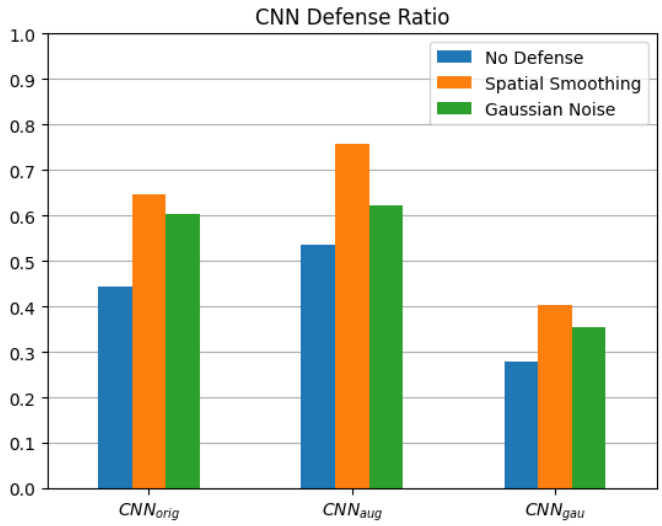

Fig. 7. defense ratio of the CNN models in relation to all 1,000 images.

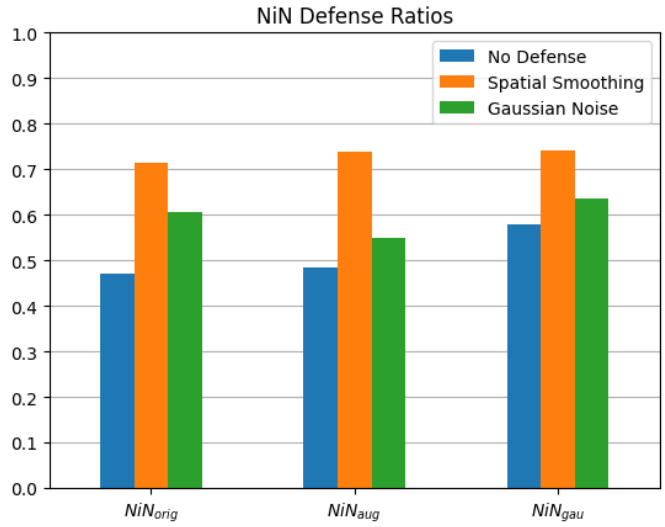

Fig. 8. defense ratio of the NiN models in relation to all 1,000 images.

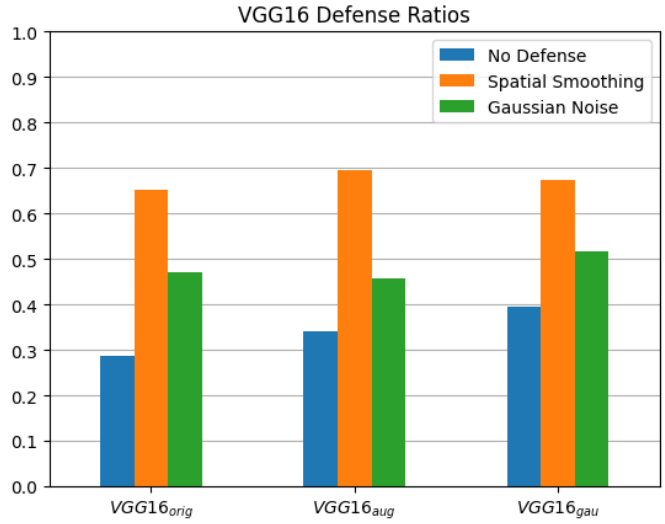

Fig. 9. defense ratio of the VGG16 models in relation to all 1,000 images.

in Appendix A. The AUC metric indicates similar results with measures of 0.93, 0.98, and 0.98 respectively. Overall, the NiN models achieved on average the highest prediction performance out of the three investigated model families. All investigated NiN models reached an accuracy of at least 83%, i.e., at least 830 correct predictions on the non-perturbed images, see Fig. 5. This is in line with what the authors of the model architecture have indicated as achieved state-of-the-art performance on the CIFAR-10 dataset [16]. The NiN model family was also the most difficult type of model to attack, since the defense ratios on average were higher than the other model families. The NiN_{orig} model achieved a defense ratio of 47.2% without applying any defense method, compared to the CNN_{orig} and $\text{VGG16}_{\text{orig}}$ models that achieved 44.3% and 28.8% respectively. This indicates that the NiN_{orig} model has a relatively higher inherent robustness compared to CNN_{orig} and $\text{VGG16}_{\text{orig}}$.

B. Defenses

1) Augmentation Defenses: Looking at Fig. 7, Fig. 8, Fig. 9, it is clear that in the majority of cases the defense ratio of the normal augmented and Gaussian augmented models is higher than the non-augmented models. The best augmentation method differs for the different model families. In the case of NiN and VGG16, Gaussian augmentation seems to be the better augmentation method, as the defense ratios in those cases are overall higher compared to their original counterparts. However, Gaussian augmentation performs significantly worse on the CNN models, as there is a clear performance deterioration compared to the other models. The normal augmentation method is associated with higher defense ratios on the CNN models, but slightly worse for

NiN and VGG16 models when compared to their Gaussian augmented counterparts. They do, however, perform better compared to their original non-augmented versions.

The experimental results indicate that the NiN models can use either of the two augmentation methods to improve the defense capabilities compared to having no defense at all. whilst the CNN and VGG16 models have increased defense capabilities for one of the augmentation methods and, depending on the model, worse for the other.

2) Spatial Smoothing: In all cases, the spatial smoothing defense was able to remove the visual representation of the attacked pixel through the blurring of the image. The problem arises when the model tries to predict on the smoothed image. It seems as if the models sometimes have difficulty in discerning the contents of the image, resulting in an incorrect prediction after the application of spatial smoothing. This seems to be specially true if the model's decision boundary for the particular class label is already uncertain from the start. There are, however, many cases where smoothing worked and the model was able to predict the correct label for the smoothed image. As seen in Fig. 4, Fig. 5 and Fig. 6 in most cases, spatial smoothing was able to correctly defend at least 50% of the attacked images, with certain models achieving up to 72% defended images. An interesting observation is that for the CNN_{orig} and CNN_{gau} models, spatial smoothing performed somewhat worse compared to the other models with spatial smoothing applied.

3) Gaussian Noise: Out of the two investigated defense methods that are applied during the prediction phase, the results indicate that Gaussian noise is the less effective defense. The results show a clear difference between the amount of perturbed images that each defense method successfully defended against, with spatial smoothing being the most suitable defense candidate. Even the Gaussian models that were trained on Gaussian noised data showed subpar performence when predicting the label on the noisy data. There are several possible reasons for this. First, the noising process could have had a too strong effect on the images, and in turn result in that the model could not properly recognize images due to the noise. Second, it could be due to the possibility that the Gaussian models were trained on too few Gaussian noised images and could therefore not capture enough variance in order to generalize properly.

Intuitively, the noised images that the Gaussian models trained on should increase the models robustness against attacks, and the ability to correctly predict correct labels on Gaussian noised images. However, it seems as there was an opposite effect as indicated by CNN_{gau} in Fig. 4. The Gaussian noise defense method performed better on models that were trained on Gaussian noised data compared to those that were not. Finally, it can be noted that adding Gaussian noise does not remove, or necessarily change, the perturbed pixel. Since the one-pixel attack relies on the model being heavily dependent on the value of the perturbed pixel, adding noise to the other pixels may not necessarily change the model's prediction.

C. One-Pixel Attack

The differential evolution algorithm was able to perturb 47% of the images that were correctly labeled for all of the models. This increases to 51% when considering only the original models. This is slightly worse than the results from the original paper, where they show a 68% success-rate for the attacks. One explanation for this difference could be that the original paper used the Kaggle version of the CIFAR-10 dataset, while we use the original CIFAR-10 dataset [4]. In the original paper the authors argued that the reason for this difference could be due to the fact that there is less noise in the original CIFAR-10 dataset, compared to the Kaggle version. This means that the models can achieve better training results, thus making the one-pixel attack less effective on the original dataset.

The combination of model and defense methods that resulted in the least number of attacked images is the same model that also showed the best prediction performance, i.e., the CNN_{aug} model using spatial smoothing. That particular model correctly predict the class label for 841 out of the 1,000 images. At the same time the one-pixel attack was successfully applied to 304 images, but the combination of normal augmentation and spatial smoothing managed to protect 221 out of these, i.e., leaving 83 images (or 8.3%) that were successfully attacked.

D. Experimental Validity Threats

As for any research method, the one chosen for this study is associated with a number of validity threats [18]. First there is an external validity threat as this study only investigates three different neural network model architectures. However, we attempt to address this by choosing the same models, including their network configuration, as the original paper describing the one-pixel attack. Another validity threat is due to the fact that the study only includes one dataset, and further, a sub-sample of images from that dataset. The reason for this is that the one-pixel attack and defense scenarios are quite computationally demanding, which limits the number of images that could be included in the study. Also, the chosen dataset is widely used in related research studies. Finally, regarding the sub-sampling a uniform random sampling was implemented.

VII. Conclusions and Future Work

The defense methods presented show potential for being effective against one-pixel attacks. The normal and Gaussian augmentation defense methods are more robust than their original counterparts, especially for the NiN and VGG16 models. Without applying any prediction defenses, Gaussian data augmentation provides the best defense for both VGG16 and NiN. Spatial smoothing seems to be the most effective defense method from a model-agnostic perspective, while Gaussian noise added during prediction shows slightly worse defense capability for all models. The ability to combine different defense methods shows an increased robustness, however, the most optimal combinations of the defenses vary

between the different network types. For the NiN, combining Gaussian data augmentation with spatial smoothing yielded the best results, while the CNN and VGG16 models worked best when combining normal augmentation and spatial smoothing. We can also conclude that the Gaussian noise defense worked best when combined with Gaussian augmentation, except for the CNN model. The combination of model and defense methods that showed best performance was CNN using a combination of normal augmentation and spatial smoothing, for which only 8.3% of the images were successfully attacked.

A. Future Work

An interesting idea for future research is to experiment with combining both augmentation methods with NiN and see if there is a big improvement in robustness over using one or the other. This can be further extended by training on smoothed images as well to see if there is an improvement when predicting on smoothed images.

Further research is also needed on the optimal parameters for the different defense methods and models. There is likely not one set of parameters that works for every situation, and a study of the optimal parameters for different situation might work as a basis when implementing these algorithms in new environments.

As mentioned in the results, we do not evaluate the effects of the defenses for non-adversarial images, and further research into any potential model degradation when using these defense methods is needed.

References

[1] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, 2016, pp. 372–387.

[2] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 IEEE Symposium on Security and Privacy (SP)*. Los Alamitos, CA, USA: IEEE Computer Society, may 2017, pp. 39–57. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/SP.2017.49

[3] I. Goodfellow, P. McDaniel, and N. Papernot, "Making machine learning robust against adversarial inputs," *Communications of the ACM*, vol. 61, 2018. [Online]. Available: https://dl.acm.org/doi/10.1145/3134599

[4] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 5, pp. 828–841, 2019.

[5] R. Storn and K. Price, "Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, p. 341–359, 1997.

[6] L. Perez and J. Wang, "The effectiveness of data augmentation in image classification using deep learning," 2017. [Online]. Available: https://arxiv.org/abs/1712.04621

[7] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," in *Proceedings 2018 Network and Distributed System Security Symposium*, vol. 1, 2018, pp. 289–303.

[8] J. F. R. Rochac, L. Liang, N. Zhang, and T. Oladunni, "A gaussian data augmentation technique on highly dimensional, limited labeled data for multiclass classification using deep learning," in *2019 Tenth International Conference on Intelligent Control and Information Processing (ICICIP)*, 2019, pp. 145–151.

[9] H.-Q. Nguyen-Son, T. P. Thao, S. Hidano, V. Bracamonte, S. Kiyomoto, and R. S. Yamaguchi, "Opa2d: One-pixel attack, detection, and defense in deep neural networks," in *2021 International Joint Conference on Neural Networks (IJCNN)*, 2021, pp. 1–10.

[10] M. A. Husnoo and A. Anwar, "Do not get fooled: Defense against the one-pixel attack to protect IoT-enabled deep learning systems," *Ad Hoc Networks*, vol. 122, p. 102627, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1570870521001499

[11] D. Chen, R. Xu, and B. Han, "Patch selection denoiser: An effective approach defending against one-pixel attacks," in *Neural Information Processing*, T. Gedeon, K. W. Wong, and M. Lee, Eds. Springer International Publishing, 2019, pp. 286–296.

[12] S. A. A. Shah, M. Beugre, N. Akhtar, M. Bennamoun, and L. Zhang, "Efficient detection of pixel-level adversarial attacks," in *2020 IEEE International Conference on Image Processing (ICIP)*, 2020, pp. 718–722.

[13] Z.-Y. Liu, P. S. Wang, S.-C. Hsiao, and R. Tso, "Defense against n-pixel attacks based on image reconstruction," in *Proceedings of the 8th International Workshop on Security in Blockchain and Cloud Computing*, ser. SBC '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 3–7. [Online]. Available: https://doi.org/10.1145/3384942.3406867

[14] A. Krizhevsky, "Learning multiple layers of features from tiny images," 2009.

[15] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," 2014. [Online]. Available: https://arxiv.org/abs/1412.6806

[16] M. Lin, Q. Chen, and S. Yan, "Network in network," 2013. [Online]. Available: https://arxiv.org/abs/1312.4400

[17] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014. [Online]. Available: https://arxiv.org/abs/1409.1556

[18] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslèn, *Experimentation in Software Engineering*. Heidelberg: Springer Berlin, 2012.

TABLE V

THE COMPLETE DEFENSE RESULTS FROM THE EXPERIMENTS.

		N	C_p	C_A	C_S	C_G	D_{R_O}	D_{R_S}	D_{R_G}	$\overline{D}_{R_O}$	$\overline{D}_{R_S}$	$\overline{D}_{R_G}$	t
Model	**CNN_{Orig}**	1,000	801	358	203	160	55.3%	80.6%	75.3%	44.3%	64.6%	60.3%	5.7
	CNN_{Aug}	1,000	841	304	221	85	63.9%	90.1%	74.0%	53.7%	75.8%	62.2%	12.3
	CNN_{Gau}	1,000	498	219	123	76	56.0%	80.7%	71.3%	27.9%	40.2%	35.5%	7.9
	NiN_{Orig}	1,000	854	382	242	133	55.3%	83.6%	70.8%	47.2%	71.4%	60.5%	10.7
	Nin_{Aug}	1,000	830	346	255	66	58.3%	89.0%	66.3%	48.4%	73.9%	55.0%	14.4
	NiN_{Gau}	1,000	850	270	161	55	68.2%	87.2%	74.7%	58.0%	74.1%	63.5%	12.5
	$\text{VGG16}_{\text{Orig}}$	1,000	815	527	364	182	35.3%	80.0%	57.7%	28.8%	65.2%	47.0%	8.5
	$\text{VGG16}_{\text{Aug}}$	1,000	825	484	353	116	41.3%	84.1%	55.4%	34.1%	69.4%	45.7%	16.4
	$\text{VGG16}_{\text{Gau}}$	1,000	841	446	279	121	47.0%	80.1%	61.4%	39.5%	67.4%	51.6%	11.0

APPENDIX

A. Result Table

In Table V we present the complete results from our experiments.

1) Variable explanation:

- N: The total number of instances tested.
- C_p: The total number of correctly predicted instances.
- C_A: The total number of correctly attacked instances from the correctly predicted instances.
- C_S: The total number of correctly defended instances with spatial smoothing from the correctly attacked instances.
- C_G: The total number of correctly defended instances with Gaussian noise from the correctly attacked instances.
- D_{R_O}: The defense ratio without applying any defense. Calculated as:
$$\frac{C_p - C_A}{C_p} = 1 - \frac{C_A}{C_p}.$$
- D_{R_S}: The defense ratio after applying spatial smoothing. Calculated as:
$$\frac{C_p - (C_A - C_S)}{C_p} = 1 - \frac{C_A - C_S}{C_p}.$$
- D_{R_G}: The defense ratio with after applying Gaussian noise. Calculated as:
$$\frac{C_p - (C_A - C_G)}{C_p} = 1 - \frac{C_A - C_G}{C_p}.$$
- $\overline{D}_{R_O}$: The normalized defense ratio without applying any defense. Calculated as:
$$D_{R_O} \cdot \frac{C_p}{N} = \frac{C_p - C_A}{N}.$$
- $\overline{D}_{R_S}$: The normalized defense ratio after applying spatial smoothing. Calculated as:
$$D_{R_S} \cdot \frac{C_p}{N} = \frac{C_p - C_A + C_S}{N}.$$
- $\overline{D}_{R_G}$: The normalized defense ratio after applying spatial smoothing. Calculated as
$$D_{R_G} \cdot \frac{C_p}{N} = \frac{C_p - C_A + C_G}{N}.$$
- t: The average time taken to perform the attack for the successfully attacked instances, measured in seconds.