Genetic Algorithm Based Adversarial Attacking and Defending of Neural Networks

Raul Ismayilov, *r.h.o.ismayilov@student.utwente.nl, s2209519, M-EE*,
Linas Li, *l.li-5@student.utwente.nl, s2288176, M-EE*,
Frederico Lopes, *f.m.alvespereiraoliveiralopes@student.utwente.nl, s3096785, M-CS*

***Abstract*—This paper presents a genetic algorithm (GA) based approach for attacking neural networks, as well as several options for defending against such attacks. The model used in this study is MobileNetV2, a widely used lightweight image classification model, which is trained on the CIFAR-10 dataset. The GA is used to generate adversarial examples for a targeted black box attack, which can mislead the model, while adversarial training and input randomization via data augmentation are employed to defend against the GA-based attacks. Additionally, a data pre-processing algorithm, which includes the calculation of the standard deviation of pixel patches, is integrated in the defense strategy against GA-based attacks. The results show that the proposed defense strategy, which incorporates both input randomization and data pre-processing, is able to effectively defend against the GA-based attacks, while having marginally reduced accuracy of the model on the original dataset. This work signifies the importance of considering both attacking and defending strategies when evaluating the robustness of neural networks and provides a promising direction for further research on adversarial robustness of models.**

***Keywords*—Genetic algorithm; Adversarial attack; Adversarial defense; Deep learning**

I. Introduction

ADVERSARIAL attacks on deep learning models have been a growing concern in recent years. There are several types of such adversarial attacks. First, a distinction needs to be made between targeted and non-targeted attacks. Targeted attack is a type of adversarial attack in which the attacker specifically generates the input data which is capable of misleading the model into making a specific incorrect prediction. In this attack, the attacker is aware of the desired misclassification result, while in a non-targeted attack, the goal of the attacker is to cause any misclassification without a specific target class. Targeted attacks are often considered more dangerous, given that they can be used for nefarious purposes to obtain the desired output. Additionally, a distinction between white-box and black-box attacks needs to be made. White-box attacks, often referred to as gradient-based attacks, require the detailed description of the model in order to exploit gradient loss. Black-box attacks, on the other hand, are attacks which only have access to the input and the output of the model, and can be modeled as optimization problems, in which the input is altered in a way which results in obtaining the desired output label [1]. In this paper, a targeted black-box adversarial attack based on genetic algorithm (GA), which can be regarded as an optimization tool, is executed on a MobileNetV2 [2] model which was trained on the CIFAR-10 dataset. After the attack is successfully executed, three methods for defense are presented: adversarial training, data pre-processing via minimum standard deviation of pixel patches, and input randomization via data augmentation.

The remaining of this paper has been divided into six parts. Section II gives an overview of the current state of the art when it comes to attacking and defending neural networks. Section III begins by laying out the theoretical dimensions of the attacking and defending, while the specifics of the proposed implementation of the black-box attacking as well as the methods for defense are laid out in Section IV. The experiments and their results are presented in Section V, where the evaluation of attacking and defending performance is carried out. The discussion and interpretation of the results is given in Section VI. Lastly, the drawn conclusions as well as future work are provided in Section VII.

II. Related work

Since [3], where the authors bring up the idea of adversarial attack on the deep learning model and demonstrate how small, carefully crafted perturbations to an input image can cause a deep neural network to misclassify it, a considerable number of adversarial attacks has been discussed [4]. Another study described in [1], proposes a novel perturbation optimized black-box adversarial attraction using genetic algorithm, which achieves a comparable success rate to white-box attack performances. Their novel Perturbation Optimization Black-box Attack is based on Genetic Algorithm (POBA-GA) and uses a number of varied randomized perturbations as the initial data. By calculating their classification results and perturbation size, the images are sent to the next step or used as adversarial attacks. For those images that failed to meet the termination condition, crossing and mutation are used to create the next generation. A more recent paper [5], proposes an improved genetic algorithm based adversarial attack strategy. They have improved the single-point crossover and simple mutation. Similarly, the fitness function is used to evaluate the performance before generating the next generation by the crossover link; different from the previous methods, the algorithm crosses parental genes within the chromosome length range one by one, which reduces the iteration numbers and speeds up convergence. Additionally, they set a small mutation rate which occasionally mutates the genes of the children obtained by crossover operation. The mutation rate is doubled when the position of the traversed gene is greater than half of chromosome's length. Using this method, a large variety of adversarial attacks can be generated in a short period of time, which makes real-time attacks possible.

In terms of defenses, there have been several studies that have introduced methods to make deep learning models more robust to adversarial attacks. An interesting pre-processing technique is mentioned in [6] which uses different denoising filters in the dataset. This technique involves removing noise from the input data, for example, by using the median filter, fast non-local means, or a wavelet transformation to remove small amounts of noise that an attacker might add to the data. Madry et al. [7] explore a method for training models with adversarial datasets to improve their robustness to specific types of perturbations. However, their model shows a limited performance on CIFAR-10, which is the dataset used in this paper and discussed in the Methodology section.

Overall, these studies clearly indicate that adversarial attacks are a serious threat to the security of deep neural networks and there is a need for future researchers to develop effective defense systems.

III. Analysis

In this section, the genetic algorithm based adversarial attacks are investigated, as well as how the defense should be utilized in order to reduce risks from a theoretical point of view.

A. Attack

Genetic algorithm, a type of optimization algorithm, is widely used in solving complex problems when traditional methods are difficult to find solutions. It is inspired by the process of natural selection. The algorithm creates a population of random candidate solutions. These candidates are evaluated by the predefined fitness function, which determines their abilities to solve the problem, and then improves the population by applying genetic operators, as listed below:

- **Selection**: it chooses which solution will be used to create the next generation, that ensures the algorithm converges to the optimal solution. It is usually done by evaluating the fitness of each solution, and then choosing the best fit to create the next generation.
- **Crossover**: it combines two or more solutions to create new solutions, since by combining the best features of different parental solutions, a new solution can be better than any of the original solutions.
- **Mutation**: it introduces some small random changes to the examples. These changes can vary the diversity of the original solutions and may be performing better than the original ones.

The selected optimized solutions are used for reproduction and survival. By repeating the algorithm of these genetic operators, the population of solutions will gradually converge to an optimal solution.

In the context of adversarial attacks, GA can be used to find the optimal perturbation that misleads the deep learning models. GA is capable of exploring a large search space for non-linear and non-convex optimization problems. At the same time, it ensures that high-dimensional input data and constraints on the solution space can be processed. A high-resolution image contains a large number of pixels, which increases the input space dimension, which is often used in attacks. It can be difficult to find the optimal perturbation because of the large amount of possible perturbations, and it can be a challenge to find the most effective one in such an input space. Additionally, the ability of processing constraints allows the attacker to incorporate specific requirements into the attacks. In this case, an image classification model, the constraint is that the perturbation should be virtually invisible to human eyes, as the goal should be to attack the system without being noticed by a human administrator.

B. Defense

There are several types of adversarial defense methods, both for detecting and correcting adversarial examples. In this part, three such methods are analyzed; this includes adversarial training, data pre-processing, and input randomization.

B.1 Adversarial training

Adversarial training is a technique aimed at making a neural network model more robust against adversarial attacks by incorporating adversarial examples into the training dataset. In this scenario, it can be done by generating images via genetic algorithm and re-training the model using both real and generated images.

Adversarial training can be described mathematically via the definition of a cost function, $J(\theta)$. This cost function represents the model's loss on the original dataset. Next, to improve the robustness against GA-based attacks, a new cost function, $J'(\theta)$, can be defined as shown in Equation (1).

$$J'(\theta) = J(\theta) + \lambda \cdot \frac{1}{m} \sum_{i=1}^{m} L(f(x_i + \delta_i; \theta), y_i) \tag{1}$$

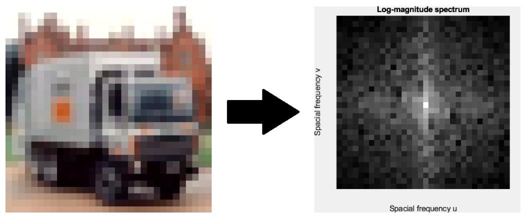
Where x_i is the original training sample, δ_i is the adversarial perturbation generated by the GA, $f(x;\theta)$ is the model's prediction for input x and its weights θ, y_i is the target label, m is the number of training examples, L is the loss function, and λ is a hyperparameter which can be used to control the weight of adversarial samples when compared to real samples. While the original model's goal is the minimization of $J(\theta)$, the goal of adversarially trained model is the minimization of $J'(\theta)$, which motivates the model to learn the features introduced by the GA attacks, and as a result, be more robust against them. This adversarial training can be, thus, viewed as a method of regularization, geared towards improving the model's generalization ability.

B.2 Data pre-processing

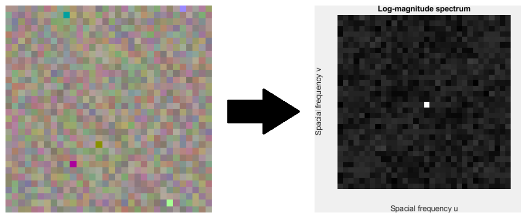
Data pre-processing is a common method of adversarial defense. Due to the fact that GA attacks introduce new pixels in the image, the standard deviation of certain pixel regions increases as the smooth structures are interrupted. Another way to look at this is by taking the 2D Fourier transformation of the image with dimensions $M \times N$ described by $f(x, y)$ according to Equation (2).

$$F(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-j2\pi(ux/M + vy/N)} \tag{2}$$

Taking the Fourier transform allows visualizing the spatial frequencies u and v present within the image. An interesting observation can be made: smooth images have largely lower frequency components present within them, while noisy images, which are generated by GA with a random initial population, contain uniformly distributed frequencies when observed on log spectrum plot as shown in Figure 1.



(a) Smooth image



(b) Noisy GA image

Figure 1: Log-magnitude spectrum of 2D FFT for a smooth and a noisy image.

As an alternative to using 2D FFT for identifying what type of image is at the input of the model, one could also scan small pixel patches within the image and calculate their standard deviation. The minimum standard deviation value can be then thresholded - the closer it is to 0, the smoother that patch of pixels is, and hence, it is more likely to not be generated by GA. This, effectively, makes use of the same concept described by analyzing Fourier transform of images in Figure 1: since real images contain smooth structures within them, they contain a larger number of low-frequency components which can be detected by using standard deviation values of small patches within the image.

B.3 Input randomization

The last defense option discussed in this paper is performed via randomization of the images at the input of CNN. Randomization inside the neural network can be considered destructive for adversarial perturbations which are designed to have a certain structure and have weak generalization. Some randomization methods include image compression, resizing, padding, rotation, and flipping. Since each image goes through a random operation within the network, the attacker is unable to effectively generate adversarial noise to be added to the image. This method has been proven to effectively defend against iterative attacks [8].

Given that the input images have been selected to have a size of 32×32, randomization operations of scaling, compression, and padding become nonviable choices since the informative part of the images will be further reduced in size and likely negatively affect the classification performance. Therefore, random image flips and rotations have been selected as randomization techniques which will be used to defend against GA adversarial attacks. An example of this defense is given in Figure 2. This can be also considered a data augmentation technique since during the training process CNN will be attempting to learn the rotational and mirror symmetries to be invariant to them given the input data X_n.

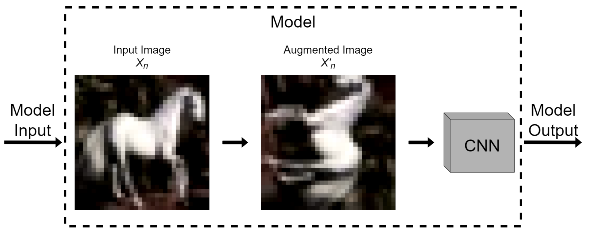


Figure 2: Adversarial defense through input randomization.

IV. Methodology

In this section, the specifics of how the proposed implementation of attacking and defending of a neural network are discussed.

A. Dataset

A dataset that contains a sufficient number of training and testing images is needed for a reliable evaluation of attack and defense methods. For this purpose, CIFAR-10 dataset is used. CIFAR-10 is a 60,000-picture dataset which contains 10 classes of objects. The images are RGB and with a size of 32×32. The images are selected to be small, which enables the possibility of iterating over many generations in GA in a short amount of time.

B. Model and training

Similar to how a lightweight dataset was selected to allow fast computations using GA over multiple generations, a lightweight network is needed to execute the attacks. For this reason, MobileNetV2, a lightweight network with only 3.4 million parameters, is used. MobileNetV2 uses a technique called depthwise separable convolutions to reduce the number of parameters and computational costs. A depthwise separable convolution is a way to factorize a standard convolution operation into two smaller operations, namely, a depthwise convolution and a pointwise convolution. The depthwise convolution applies a single filter to each input channel, while the pointwise convolution applies a 1×1 convolution to combine the output of the depthwise convolution [2].

This factorization of the standard convolution operation allows MobileNetV2 to have a much smaller number of parameters than a traditional convolutional neural network while maintaining a similar level of accuracy. Additionally, it uses the inverted residual structure, which allows increasing the depth and width of the network while maintaining the computational cost constant [2].

The original MobileNetV2 model includes 1000 output layers, while there are only 10 outputs needed for the CIFAR-10 dataset. Therefore, the final fully-connected layer of the model was replaced to contain only 10 output neurons. The training is executed for 100 epochs using cross-entropy loss and Adam optimizer with a learning rate of 0.001. Each 10 epochs the weights are saved, and the weights which result in the highest test accuracy are kept for evaluation.

C. Attack

For the adversarial attack, genetic algorithm was used to create the images that would be used to mislead the model into giving a wrong output label. The implemented GA performs a targeted black-box attack, which means that the class to be generated can be selected, and the algorithm has only access to the input and output of the model, as it would in most real-life scenarios. The implementation of the proposed genetic algorithm was inspired by [9] and is described in Algorithm 1, where the fitness function is an algorithm that gives a score to each member of the population according to how large the target class probability is. The crossover function is performed by randomly combining genes of two parents, while mutation is performed by using a Gaussian distribution.

Given this implementation, two different types of images, which are shown in Figure 3, can be generated. The difference being that, in one case, the initial population is randomly generated, and in the other, a test image is fed to the algorithm as the initial target population. In the first scenario, this results in an image that resembles random noise, while in the second scenario, the output is similar to the original image with some added pixels for the attack.

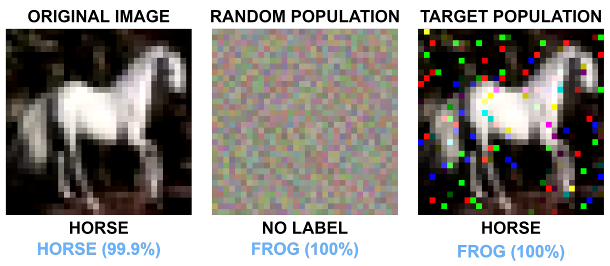


Figure 3: Adversarial attack using GA with random and target initial populations targeting class "frog".

Algorithm 1 Genetic algorithm.

```
1:  initialize parameters
2:  if loading initial population then
3:      $current\_population \leftarrow$ batch of target population
4:  else
5:      $current\_population \leftarrow$ random population
6:  end if
7:  for $i$ in range($population\_size$) do
8:      $scores \leftarrow$ fitness_function($current\_population$)
9:      $best\_population \leftarrow$ best 80% based on $scores$
10:     $random\_population \leftarrow$ 5% of $population\_size$
11:     $children \leftarrow$ zeros()
12:     for remaining 15% of $population\_size$ do
13:         select randomly $parent1$, $parent2$ from $current\_population$
14:         $child \leftarrow$ crossover($parent1$, $parent2$)
15:         mutate $child$ with 5% chance
16:         append $child$ to $children$
17:     end for
18:     $current\_population \leftarrow best\_population + random\_population + children$
19:     if no change for $stop\_num$ generations then
20:         stopping early
21:     end if
22: end for
23: return $current\_population$
```

D. Defense

In total, three various defense techniques have been implemented in this paper: adversarial training, data pre-processing, and input randomization. The descriptions behind the implementation specifics of these three techniques are given in this section.

D.1 Adversarial training

The first implemented defense technique which is targeted to defend against a GA attack using a random initial population is adversarial training. It is performed by generating 600 images for each attack class of CIFAR-10 and labeling them with a new "generated" class label. Therefore, a new class containing 6000 generated images is added to 60,000 images of CIFAR-10 dataset. This requires a new model with 11 output classes to be trained on this updated dataset. This model from now on will be referred to as "*adversarial model*". The goal of this defense method is to let the model to learn the features associated with GA attack using random initial population and become robust against this type of attack.

D.2 Data pre-processing

To incorporate data pre-processing as a defense measure, a new model, which will be referred to as "*pre-processing model*", is created. Once again, this method is aimed to defend against GA attack which uses random initial population. Unlike *adversarial model*, this model contains a neural network with 10 output neurons (CIFAR-10 classes) and one artificial output ("generated" class). Before the input images pass through the neural network, a function described in Algorithm 2 is called, and if an image is detected to have a patch with a minimum standard deviation lower than 3.0, it is counted as generated. When the network detects a generated image, it returns a custom tensor with 11 dimensions, the last of which has the highest value. This indicates that the returned tensor results in the 100% class probability for the label "generated". If the image passes data pre-processing stage, it is allowed to pass through the network which generates a 10-dimensional output tensor, to which an additional 11th dimension is concatenated and set to a very low value, indicating that the probability of the image being classified as "generated" is 0%.

Algorithm 2 Minimum standard deviation of pixel patches.

```
1: Input: image
2: Convert $image$ to grayscale
3: $patch\_size \leftarrow (3, 3)$
4: $overlap \leftarrow (1, 1)$
5: $patches \leftarrow$ split image in patches
6: $patch\_stds \leftarrow numpy.std(patches)$
7: return $min(patch\_stds)$
```

D.3 Input randomization

The final defense model which utilizes input randomization will be referred to as "*random model*". This model was designed to utilize both the input randomization described in Algorithm 3 and the minimum standard deviation of pixel patches as described in Algorithm 2; it is aimed at defending against attacks using both random and target initial populations.

Algorithm 3 Input randomization.

```
1: Input: image_batch
2: initialize $output\_images$ of size($image\_batch$) with zeros
3: for $image$ in $image\_batch$ do
4:     $angle \leftarrow$ random number between 0 and 360
5:     $flip\_type \leftarrow$ horizontal or vertical
6:     $image \leftarrow$ rotate($image$) by $angle$    ▷ using nearest pixel interpolation at edges
7:     $image \leftarrow$ flip($image$) with $flip\_type$
8:     append $image$ to $output\_images$
9: end for
10: return $output\_images$
```

When a batch of images is at the model input, first, images are randomly rotated and flipped. Next, if the standard deviation of the entire image is larger than every image in CIFAR-10 dataset or if a patch with minimum standard deviation larger than 3.0 is found, the image is counted as generated due to not being smooth and the corresponding tensor is returned by the model. If data pre-processing stage is passed, the image runs through network and a 10-dimensional output tensor is extended with an 11th dimension which signifies 0% probability associated with the "generated" class.

The reason for incorporating both data pre-processing and input randomization in this model is because the GA is capable of generating images which are rotation and mirror invariant at the cost of flooding the target image with various noise pixels. Once the image becomes filled with random pixels, the data pre-processing function is expected to identify it. Therefore, this model incorporates the following duality: if an image is smooth and resembles a target image with only minimal changes performed by GA, the attack is expected to fail due to input randomization; once GA bypasses input randomization by adding a large number of attack pixels, data pre-processing prevents the attack from continuing. Consequently, the GA would become very limited in terms of actions it can perform and would fail to converge to a working solution.

V. Experiments and Results

In this section, the experiments targeted at evaluation of attacks and defenses are discussed and their results are presented.

A. Model testing

The class accuracies and the total accuracy of the trained model on CIFAR-10 are given in Table I. Based on the testing results, the total accuracy of 84.2% was obtained.

Class	Accuracy
Plane	87.1%
Car	91.9%
Bird	77.5%
Cat	65.6%
Deer	83.2%
Dog	79.6%
Frog	90.5%
Horse	84.2%
Ship	90.4%
Truck	91.7%
Total	84.2%

Table I: Testing accuracy of MobileNetV2 on CIFAR-10.

B. Attack

The proposed GA algorithm was used to attack the trained model. Figure 3 shows how two different types of attacks can be applied. In case of random population attack, the generated images contain random pixels since the goal of the used fitness function is only maximizing the class probability. On the other hand, when using a target population, the generated images resemble their targets with only a small percentage of pixels changed. Results show that when an image is attacked using GA with a target image as an initial population, the average pixel-wise difference between images is 20.29%, signifying that only a small number of pixels is needed to maximize the probability of the attacked target class. All of the performed attacks are executed for 1000 generations with no early stopping. It was observed that whenever the attack is executed, the probability of it belonging to the target class is exactly 100%, meaning that the attacks are very effective at misleading the neural network.

C. Defense

The results of the proposed defense methods were observed to vary between methods and are discussed in this section.

C.1 Adversarial training

After the 6000 images of adversarial samples of random initial population attack have been concatenated with the CIFAR-10 dataset, a new model with an additional "generated" class has been trained. The test set accuracy results are given in Table II. As can be seen, the model learned to detect all generated images correctly. However, these images have been generated on the original model, and when GA is used to generate images by directly attacking the *adversarial model*, the results are unsatisfactory: GA is able to bypass the adversarial defense and generate new samples which successfully target the attack class.

Class	Accuracy
Plane	79.6%
Car	82.3%
Bird	54.6%
Cat	36.4%
Deer	68.3%
Dog	56.8%
Frog	80.5%
Horse	74.8%
Ship	81.8%
Truck	77.5%
Generated	100%
Total	72.1%

Table II: Testing accuracy of adversarially-trained model.

C.2 Data pre-processing

Unlike the *adversarial model*, the *pre-processing model* does not change the original model accuracy and solely targets to detect the random initial population attack images. The same testing dataset, which includes both CIFAR-10 and generated images, was used to observe that the results given in Table I hold with an addition of 100% accurate detection of the generated samples. Additionally, samples generated by attacking the *adversarial model* directly also fail to be effective against the *pre-processing model*. Thus, this method of defense is effective, however, direct attacks on the *pre-processing model* need to be executed as well.

To observe whether direct attacks applied to the *pre-processing model* are effective, a confusion matrix is constructed. This was achieved by using 500 random images from the CIFAR-10 test set and 500 GA attack images generated on the original model. These images are combined in one evaluation dataset with two ground truth labels which signify whether the image is generated. Next, this dataset is shuffled and images are fed through the *pre-processing model*. The number of true positives, true negatives, false positives, and false negatives is counted, and a confusion matrix given in Figure 4 is constructed. Based on the results, it can be seen that GA struggles to generate adversarial samples, as it only manages to bypass the model's defense with only 1% of the samples. Additionally, 1.6% of the real images have been classified as generated due to the threshold used for maximum allowed standard deviation of pixel patches. This is because some of the images in the CIFAR-10 dataset have larger standard deviation than one set by this threshold; to be specific, 0.75% of CIFAR-10 images are expected to be incorrectly labeled as generated.

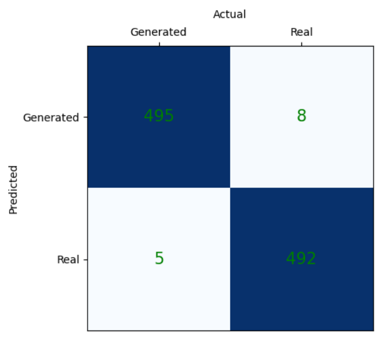


Figure 4: Confusion matrix based on the results of model with data pre-processing defense.

C.3 Input randomization

The final defense model, the *random model*, requires retraining since the model now is expected to learn rotational and mirror symmetries. This is because every time the new image is at the input of the model, it is rotated and flipped, meaning that the model needs to still be able to identify the class even after such augmentation. Given that this model is aimed to defend against both random and target initial population attacks, it is only trained on CIFAR-10 dataset. The results of testing accuracies are given in Table III. It is worth noting that since the input images passing through the neural network change every time, the provided testing accuracy is not fixed, instead, it is changes slightly on every run as the images get augmented differently. Additionally, the training and testing of this model is performed differently: during training, there is no need to use data pre-processing operation which determines whether the image is generated or not and adds an additional dimension to the output tensor, however, during testing when the model is expected to be attacked, data pre-processing is required.

Class	Accuracy
Plane	77.4%
Car	87.2%
Bird	65.0%
Cat	62.5%
Deer	67.7%
Dog	65.7%
Frog	76.1%
Horse	76.5%
Ship	80.4%
Truck	80.6%
Total	73.9%

Table III: Testing accuracy of the model with input randomization.

When this model is attacked directly using a random initial population attack, the results are similar to the *pre-processing model*, as generated images are detected by the described data pre-processing technique. However, when the *random model* is attacked directly by using real target images as initial populations, the results are interesting - the output of the GA is often the same image as the target image. This means that GA does not alter the images in any way. This discovery is further elaborated in the Discussion section. Since the images are often unaltered, the model is capable of classifying the images normally, meaning that this method of defense is effective against the proposed genetic algorithm. The defense evaluation is once again performed via a confusion matrix by using 500 real test images and 500 generated images; this time not using random initial populations but instead using real images as target populations. Additionally, when a generated ground truth label image is detected to be unaltered, it is counted as a true positive (thus, it is assumed that the prediction is "generated" to prevent counting inaccurate false negatives). The confusion matrix is given in Figure 5. As can be seen, only 2% of generated images managed to bypass the defense and 1.2% of CIFAR-10 images have been incorrectly labeled as generated due to data pre-processing stage incorporated in the model.

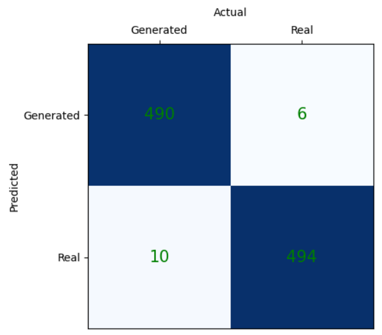


Figure 5: Confusion matrix based on the results of model with input randomization defense.

VI. Discussion

In this section, the results provided in the previous section will be further discussed with the goal of obtaining more insight into the inner-workings of the attack and defense mechanisms.

A. Attack evaluation

The results show that an undefended network is very susceptible to genetic algorithm-based attacks. The robustness and high effectiveness, when it comes to the target class probability of the attack, of the proposed method can be largely attributed to the chosen fitness function. The majority of the adversarial attacks in the current research attempt to make the presence of the attack hard to detect by the human eye. These attacks are often generated by having the fitness function accomplish two goals: maximizing the target class probability and minimizing the visual effect on the image by, for instance, calculating the MSE between the original and attack image. While these methods are successful, they sacrifice some of the performance to make the attack hard to detect. As a result, the confidence the model gives to the target attack class decreases. Additionally, given that these attacks need to leave a minimum presence on the image, they can be easily disrupted by, for example, median filtering of the images. On the contrary, the fitness function used in this paper has the single goal of maximizing target class probability. This makes the attack exceptionally powerful and the model becomes extremely certain about its wrong decision.

All of the attacks executed on the undefended model as well as some of the attacks on the defended models were highly successful and, in most cases, the model under attack gave 100% confidence to the targeted class. This shows that GA-based adversarial attacks are exceptionally powerful and hard to defend against. Nevertheless, there are some drawbacks associated with this attack type which are further discussed in the next section.

B. Defense evaluation

B.1 Adversarial training

The results show that defense via adversarial training against GA-based attack was unsuccessful when the trained model is attacked directly. In addition, due to re-training and addition of a new class, the overall test accuracies of many CIFAR-10 classes are reduced. In order to correctly identify the generated images during re-training of the model, it was likely forced to lose some of its knowledge about features of CIFAR-10 classes. The reason why this method was susceptible to direct attacks can be explained by a few potential reasons. First, it is likely that GA-based generated images are not diverse enough. Analogous to mode collapse in GANs, where the generator only produces a limited number of variations of the data, GA-based attack might be learning only one method of attack, and when it is defended against by adversarial training, it simply collapses to a different mode. This is supported by the fact that the adversarially trained model was able to identify all generated images from its test set. However, when new images were generated based on direct attack, the defense failed. Secondly, it is highly likely that the search space of GA is much larger than what the adversarial model is capable of detecting. More advanced training techniques, such as an ensemble of models, could be a better solution when adversarially training the model. Lastly, given the potency of the used fitness function, the GA is not limited by any other factors, such as making the image look real, and thus, it can always generate a different solution.

B.2 Data pre-processing

The model using data pre-processing at the input stage has shown more promising results: most of the generated samples are identified correctly, meaning that the defense method is successful. Furthermore, the accuracy of the model is largely unaffected since no re-training needs to be performed. However, a small subset of real CIFAR-10 images had to be sacrificed to be labeled as generated images. Upon observing those specific images, it was concluded that these images do not contain much smoothness within them, that is, both the objects and the backgrounds vary substantially even on patches of 3 by 3 pixels. Additionally, a few images generated by the genetic algorithm managed to bypass the implemented defense method, which means this defense is not flawless. The reason for this is described in the next section, since the model utilizing input randomization also uses data pre-processing and the explanation can be formulated more clearly. Overall, while not perfect, data pre-processing could be one of the optimal ways by which a neural network model can be defended against GA-based attacks which make use of a random initial population method.

B.3 Input randomization

The most insight into how the defense mechanisms prevent GA-based attacks was gained using the *random model*. The results are once again promising, with only a small number of adversarial images being able to circumvent the defense and an overall reduction in accuracy of the model by 10.3% due to the introduced input randomization. One of the examples that managed to bypass the defense is given in Figure 6. This image shows that GA was capable of generating a sample which is both rotation/flip-invariant and is not flooded by many generated pixels. Upon further investigation, it was found that the main success of the proposed defense is due to the custom tensor which is returned when a generated sample is detected by data pre-processing algorithm. Since the genetic algorithm is an optimization problem, it attempts to maximize the probability of the target class. However, when GA creates a generation which adds too many new pixels to the image, the data pre-processing returns a tensor which gives 0% probability to the target class, effectively ruining all the efforts of the GA and forcing it to start over with a different generation. Therefore, while there are possible solutions which the GA should be able to generate to fool the network, these defenses prevent it from reaching the optimal point easily.

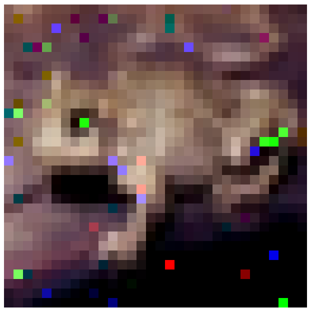


Figure 6: An adversarial sample which bypassed the defense of model with input randomization and data pre-processing.

An analogy can be made with the gradient descent algorithm, which is attempting to reach the global minimum (rotation/flip-invariant adversarial image with not many added pixels). However, there are certain local minima (solutions which include too many added pixels) that when reached, force the gradient to diverge momentarily, making the starting point a better solution than the diverged result. This is exactly what happens to the GA and why often the returned images are unaltered. While it is working hard to bypass the input randomization, it gets constantly diverged when too many pixels are added to the image. This hard threshold response disrupts the solving of the optimization problem and prevents the GA from finding a successful solution. While there are some samples, such as the one shown in Figure 6, which are able to not trigger data pre-processing defense, the majority of the populations are simply unable to reach the optimal solution due to many walls placed in front of them.

VII. Conclusion

The goal of this paper was to introduce the concept of adversarial attacks and defenses with examples of how this can be implemented. The MobileNetV2 network trained on CIFAR-10 dataset was subjected to an attack by a genetic algorithm capable of executing targeted black-box attacks. The attacks proved to be very effective, and to defend against them, three different defense methods were analyzed: adversarial training, data pre-processing, and model input randomization. While adversarial training proved to be an ineffective method of defense against GA-based attacks, the other two methods showed promising results by achieving 98.7% and 98.4% accuracy when it came to detection and prevention of attacks by generated images. In the case of the latter method of defense, a sacrifice in the overall test set accuracy of the model had to be made to incorporate an input randomization layer.

While the results gave an insight into the inner-workings of the GA-based attacks, improvements in both attack and defense can be made. It was observed that for the attack, which is attempting to solve the optimization problem, a sudden change in the model's output due to a defense algorithm results in divergence of the solution. To work around this, one could implement a safety mechanism within the GA which would allow the algorithm to adapt to the defense and adjust its strategy accordingly. In addition, the use of more advanced and specialized optimization algorithms, such as Bayesian optimization or reinforcement learning, could potentially improve the performance of the attack.

When it comes to the defense, GAN inspired solution might be the best option to defend against GA. A discriminator could be trained by using the GA as a generative network and the task of the discriminator would be to identify and classify adversarial images generated by the GA. This approach would provide a more dynamic defense, as the discriminator would be able to adapt and improve over time as the GA evolves. Additionally, the use of GANs for defense could also be combined with other methods such as input randomization and adversarial training, providing a multi-faceted defense against GA-based attacks.

In summary, adversarial attacks and defenses are an emerging field of study with important implications for the security and robustness of machine learning models. The use of genetic algorithms for generating adversarial images has proven to be an effective attack method, and the use of input randomization and data pre-processing methods have shown promise as defense mechanisms. However, there is still room for improvement, and future research should explore new and innovative methods for both attacking and defending against adversarial images.

References

[1] J. Chen, M. Su, S. Shen, H. Xiong, and H. Zheng, "POBA-GA: perturbation optimized black-box adversarial attacks via genetic algorithm", *CoRR*, vol. abs/1906.03181, 2019. arXiv: 1906.03181. [Online]. Available: http://arxiv.org/abs/1906.03181.

[2] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[3] C. Szegedy *et al.*, *Intriguing properties of neural networks*, 2014. DOI: 10.48550/ARXIV.1312.6199. [Online]. Available: https://arxiv.org/abs/1312.6199.

[4] N. Carlini and D. Wagner, *Towards evaluating the robustness of neural networks*, 2016. DOI: 10.48550/ARXIV.1608.04644. [Online]. Available: https://arxiv.org/abs/1608.04644.

[5] D. Yang, Z. Yu, H. Yuan, and Y. Cui, "An improved genetic algorithm and its application in neural network adversarial attack", *PLOS ONE*, vol. 17, no. 5, M. N. A. Wahab, Ed., e0267970, 2022. DOI: 10.1371/journal.pone.0267970. [Online]. Available: https://doi.org/10.1371\%2Fjournal.pone.0267970.

[6] 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*, Internet Society, 2018. DOI: 10.14722/ndss.2018.23198. [Online]. Available: https://doi.org/10.14722\%2Fndss.2018.23198.

[7] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, *Towards deep learning models resistant to adversarial attacks*, 2017. DOI: 10.48550/ARXIV.1706.06083. [Online]. Available: https://arxiv.org/abs/1706.06083.

[8] C. Xie, J. Wang, Z. Zhang, Z. Ren, and A. L. Yuille, "Mitigating adversarial effects through randomization", *CoRR*, vol. abs/1711.01991, 2017. arXiv: 1711.01991. [Online]. Available: http://arxiv.org/abs/1711.01991.

[9] P. Tyshevskyi. "Adversarial attack using genetic algorithm". (2019), [Online]. Available: https://medium.com/analytics-vidhya/adversarial-attack-using-genetic-algorithm-90beba13b6cb.