

# Attacking and Defending a ResNet18 Architecture

*Guus Branderhorst (s2795132)*

*Lizzy-Milou de Bruijn (s3065642)*

*Kim E-Shawn Brandon (s3747883)*

## Introduction

In this study on attacking and defending neural networks, a convolutional neural network (CNN) is chosen as the initial network together with the Animals-10 dataset. The Animals-10 dataset contains animal images that can be grouped into the following classes: dogs, cats, horses, spiders, butterflies, chickens, sheep, cows, squirrels, and elephants.

For the adversarial attack, the Iterative Fast Gradient Sign Method (I-FGSM) is chosen. The Fast Gradient Sign Method (FGSM) only has a low success rate, but it is super efficient. By making the FGSM attack an iterative process, the success rate is increased while keeping its efficiency. FGSM is designed to reduce the confidence in the prediction of the neural network. This is done by introducing perturbations that minimize the margin of decision boundaries. It uses the model's own gradient against it by tweaking the input to maximize the error. I-FGSM is part of the white box attacks. It has complete knowledge of the target model's architecture, parameters, and training methodology.

Furthermore, we propose a topological optimization defense where we modify the network's effective structure to reduce the adversarial vulnerability while preserving clean accuracy as much as possible. For this defense, structured pruning and fine-tuning of the network will be utilized. Finally, the performance of the proposed system is assessed carefully.

## Research background

### Attack: I-FGSM

Frequently, when training a machine learning model, the model parameters are optimized using backpropagation. Backpropagation relies on computing the gradient of the loss function with respect to the model parameters. This gradient indicates how the parameters should be updated in order to minimize the loss, thereby improving the model's performance.

In addition to computing gradients with respect to the model parameters, gradients can also be computed with respect to the input data. In this case, the gradient indicates how small perturbations to the input can increase the model's loss. This property is exploited by adversarial attack methods such as the Fast Gradient Sign Method.

Adversarial attacks aim to fabricate a new image by adding controlled noise to the original image, such that the changes are almost undetectable to the human eye and that neural networks missclassify the image.

The FGSM is a well-known adversarial attack that uses this idea. It generates an adversarial example by adding a small perturbation in the direction of the sign of the gradient of the loss with respect to the input, as shown in Equation 1

[1].

$$x_{\text{adversarial}} = x + \epsilon \cdot \text{sign}(\nabla_x J(\theta, x, y)) \quad (1)$$

Here,  $\epsilon$  is the maximum allowed  $L_\infty$  norm of the perturbation,  $\theta$  represents the model parameters,  $x$  is the original input,  $y$  is the true label, and  $J(\theta, x, y)$  denotes the loss function. The perturbation is designed to be small enough to remain imperceptible while still causing the model to classify the input incorrectly.

The Iterative Fast Gradient Sign Method extends FGSM by applying the attack multiple times with a smaller step size. Instead of performing a single update, the adversarial example is refined iteratively, allowing for a more effective attack. At each iteration, the perturbation is clipped to ensure that the total perturbation remains within a predefined bound. This iterative process can be expressed as [2] [3]:

$$x_{(t+1)} = \text{Clip}_{x, \epsilon} \{ x_{(t)} + \alpha \cdot \text{sign}(\nabla_x J(\theta, x_{(t)}, y)) \} \quad (2)$$

Where  $\alpha$  is the step size and  $\text{Clip}_{x, \epsilon}$  ensures that the adversarial example remains within an  $\epsilon$ -ball around the original input  $x$ .

From an optimization point of view, I-FGSM can be seen as a projected gradient ascent on the input, where the objective is to maximize the loss function under an  $L_\infty$  norm constraint that is set by the  $\epsilon$ -ball. The clipping operation, as in Equation 2, enforces this constraint by projecting the adversarial attacked input back into the feasible region after each iteration, ensuring that the perturbation remains within the designed bounds.

### Defense: Structured pruning

Deep neural networks are often highly overparameterized, meaning that many parameters have only a limited influence on the final prediction. As a result, removing a subset of these parameters does not necessarily lead to a significant drop in classification accuracy. This observation forms the basis of network pruning methods.

Structured pruning removes entire components of a network, such as convolutional filters or channels, rather than individual weights. By doing so, the overall size and capacity of the network are reduced in a controlled way. As a result, the model becomes less complex and behaves more like a regularized network, which can improve generalization.

The idea of using pruning as a defense against adversarial attacks is based on the assumption that a simpler model may be less sensitive to small input perturbations. If redundant or weakly contributing features are removed, adversarial noise may have a smaller effect on the model's output. Based on

this intuition, Wang et al. [4] propose structured pruning as a defense mechanism against adversarial attacks.

In their approach, pruning is applied to a trained network, after which the remaining parameters are fine-tuned. This re-training step is important, as it allows the pruned network to adapt to its new structure and recover performance. In this project, a similar strategy is followed by pruning convolutional filters and fine-tuning the remaining network to improve robustness while preserving clean classification accuracy.

## Theoretical Analysis

### I-FGSM

I-FGSM is expected to be effective against the chosen ResNet-18 architecture. The CNN is composed of piecewise linear operations, including convolutions, ReLU activations, and residual connections. Within a local region of the input space, ResNet-18 behaves approximately as a linear function, making gradient-based attacks aligned with the true loss surface [5] [6]. In addition, the residual connections further ensure stable gradient propagation to the input, while the iterative updates allow the attack to adapt to changes in activation patterns across ReLU boundaries. Therefore, I-FGSM can induce misclassification under small pixel perturbations.

The effectiveness of I-FGSM can be explained using the first-order Taylor approximation of the loss function around an input:

$$J(x + \delta) = J(x) + \delta^T \nabla_x J(x) \quad (3)$$

Here,  $J(x)$  denotes the loss function evaluated at the original input  $x$ ,  $\delta$  represents a small perturbation applied to the input that maximizes the loss within the  $L_\infty$  ball of radius  $\epsilon$  [5], and  $\nabla_x J(x)$  is the gradient of the loss with respect to the input. The gradient indicates the direction in the input space in which the loss increases most rapidly.

Under a norm constraint on the perturbation, such as an  $L_\infty$  constraint  $\|\delta\|_\infty \leq \epsilon$  [7], the goal of the attacker is to choose  $\delta$  such that the inner product  $\delta^T \nabla_x J(x)$  is maximized. This inner product measures how strongly the perturbation aligns with the gradient direction.

For an  $L_\infty$ -bounded perturbation, the optimal solution that maximizes this inner product is obtained by setting each component of  $\delta$  to have magnitude  $\epsilon$  and the same sign as the corresponding component of the gradient. Consequently, perturbations aligned with the sign of the gradient result in the maximum increase of the loss while remaining within the allowed perturbation region. This principle forms the theoretical foundation of FGSM and its iterative extension, I-FGSM.

Therefore, from Equation 3 it can be observed that perturbations aligned with the gradient of the loss maximally increase the loss within a constrained norm region [5].

In addition, the iterative part of this adversarial attack method allows the attack to adapt as the input crosses the activation boundaries, caused by ReLU nonlinearities. This, in turn, improves the alignment with the true loss surface.

The downside of I-FGSM, however, is that it assumes the

model to be a white box. In other words, it assumes to know all model parameters. This is not always feasible, meaning that in a black-box scenario, the attack is less effective

### Structured pruning

Structured pruning is often used as a defense because it removes redundant parameters and reduces the capacity of a neural network. By removing entire filters or channels, the model becomes less complex and behaves more like a regularized network. This can improve generalization and may slightly reduce sensitivity to weak adversarial perturbations, especially for single-step attacks such as FGSM.

However, from a theoretical point of view, structured pruning does not remove the gradients that adversarial attacks rely on. Even after pruning, the network is still composed of convolutions, ReLU activations, and residual connections, and therefore remains a piecewise linear function. This means that, in a local neighborhood of the input, the loss function can still be approximated as linear, and gradients with respect to the input are still well defined. As shown by Jordão and Pedrini [8], the robustness improvements obtained through pruning are therefore limited and depend strongly on the attack strength.

In the case of iterative attacks such as I-FGSM, the perturbation is updated multiple times by repeatedly following the gradient of the loss. This allows the attack to adapt as the input crosses activation boundaries caused by ReLU nonlinearities. Since structured pruning mainly reduces model size rather than changing how gradients behave, these gradients remain informative. As a result, iterative white-box attacks can still exploit the pruned model, and pruning alone is not sufficient to defend against strong adversarial attacks.

## Implementation

### Pretrained Network

For the initial implementation, a pretrained CNN is used to classify an image. A CNN is specialized in image classification by its ability to increase its complexity with each layer of the neural network. As the image data progresses through the layers, greater portions of the image are identified.

The main component of the CNN architecture are the convolutional layers. A key feature of these layers is parameter sharing. Hereby, weights are shared in the first few layers, reducing the number of unique model parameters and improving the generalization capabilities of the CNN. Another important layer in the architecture is the pooling layer. This layer helps to reduce the number of dimensions while preserving important features, resulting in the reduction of the computational complexity and the increase of the networks efficiency. Because of these strong capabilities of the CNN for image classification, it was more interesting to look for a way to take down this already effective network and look for ways to improve its architecture.

In this project, a total of 8000 images spread across 10 classes of different kinds of animals are used from the Animals-10 dataset. The ResNet18 architecture is a com-

monly used type of CNN for image classification. It uses residual connections to prevent vanishing gradient and provide a stable training process. To train our network with the Animals-10 dataset, transfer learning has been applied. The pre-trained ResNet-18 model has already been trained on the ImageNet data. As a result, the network has learned robust and generic feature representations that are well-suited for image classification tasks. Transfer learning reduces thus the risk of overfitting and reduces the computational cost. These learned features can be effectively transferred to the Animal-10 dataset. The final fully connected layer is replaced with a new classification layer consisting of 10 output units, corresponding to the number of classes in the Animal-10 dataset.

The dataset was randomly split into a training 80% and a validation 20% set. Before training, all images were resized to a resolution of  $244 \times 244$  pixels to match the input requirements of the ResNet-18 architecture, which was trained originally on the ImageNet data set. To improve generalization, data augmentation in the form of random horizontal flipping was applied to the training set only. The model was trained using stochastic gradient descent with a learning rate of 0.01 and optimized using the cross-entropy loss function, which is well-suited for multi-classification tasks. A batch size of 64 was selected as a balance between computational efficiency and training stability, providing reliable gradient estimates.

Training was performed for four epochs, which was sufficient to establish a strong baseline performance due to the rapid convergence enabled by transfer learning. During training, model parameters were updated via backpropagation, and training accuracy and loss were recorded at each epoch. Model performance was evaluated after every epoch using the validation set. During validation, gradient computation was disabled to reduce memory usage and ensure that no parameter update occurred. Validation loss and accuracy were computed to assess the model's generalization performance. The following training and validation outputs are obtained:

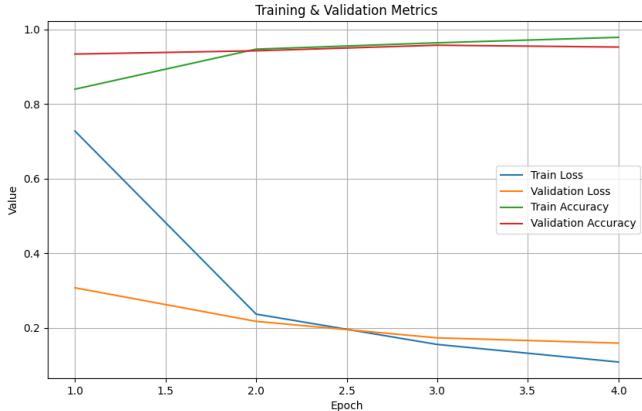


Figure 1: Training and validation losses, as well as accuracies, plotted against the epoch number during ResNet18 training.

In Figure 1, it can be seen that both training and validation loss are decreasing, while accuracy increases. This means that the model is learning well with no obvious overfitting. Beforehand, it was discussed that once a training accuracy of ~85-90% was accomplished, it was sufficient to continue with the attack and defense on the network. Eventually, a training accuracy of 98% is achieved.

The trained weights of this model were saved and used as the baseline throughout the remainder of this project. Initially, a baseline model trained for 4 epochs was used to conduct early attack and defense experiments. This model is referred to as `resnet18_animal10.pth`. To obtain a more stable and better-performing reference, an additional baseline model was trained for 20 epochs using the same architecture and dataset. This second baseline (`resnet18_animal20.pth`) achieved smoother training behavior and higher validation performance, please refer to Figure 2.

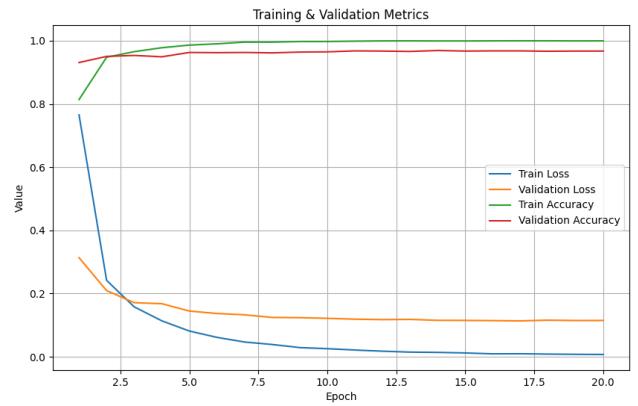


Figure 2: Training and validation losses, as well as accuracies, plotted against the epoch number during ResNet18 training.

Both baseline models are used in later experiments to compare how training stability and model convergence affect adversarial vulnerability and defense performance.

## I-FGSM

After training two base models, one with 4 epochs and another with 20 epochs, the I-FGSM attack is performed on both models. The heatmap in Figure 3 shows the influence of the perturbation  $\epsilon$  and the number of iterations on the accuracy.

In Figure 3, it can be seen that clean accuracy remains constant across different values of  $\epsilon$  and number of iterations. However, adversarial accuracy drops significantly as  $\epsilon$  increases, showing that larger perturbations make adversarial examples more effective. The same analysis is performed on the model that is trained for 20 epochs. The results are shown in 4.

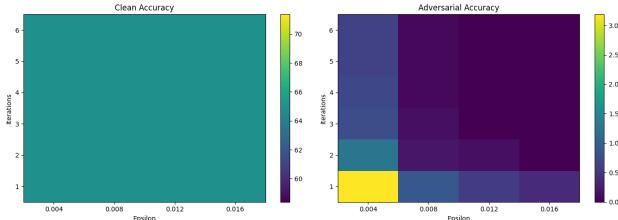


Figure 3: The influence of iterations and  $\epsilon$  on the accuracy of the model and the accuracy of the attacked model.

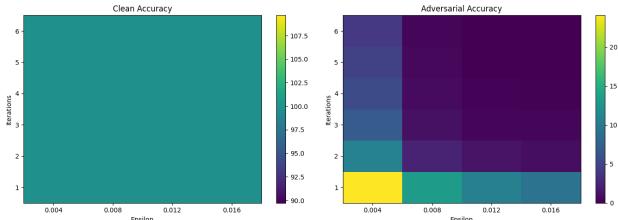


Figure 4: The influence of iterations and  $\epsilon$  on the accuracy of the model and the accuracy of the attacked model.

In the 20-epoch base model, depicted in Figure 4, the clean accuracy remains uniform across different values of  $\epsilon$  and number of iterations. In contrast to the 4-epoch model, the adversarial accuracy of the 20-epoch base model decreases less rapidly, indicating improved robustness with increased training epochs. Nevertheless, the I-FGSM attack remains effective and still leads to a noticeable reduction in overall performance.

### Structured pruning defense

The first defense method we consider is structured pruning followed by fine-tuning. The main idea is to reduce the effective size of the network by removing entire convolutional filters, while trying to keep the clean classification accuracy as high as possible. The defended model is based on a ResNet-18 architecture trained to classify 10 classes.

**Data and setup:** All images are resized to  $224 \times 224$  so that they match the input size expected by ResNet-18. During training, random horizontal flips are used as a simple form of data augmentation. The dataset is split into 80% training data and 20% validation data using a fixed random seed, so that the same split is used in all experiments. Training and evaluation are performed on a GPU when available.

**Baseline model loading:** We start from a ResNet-18 model with the final fully-connected layer modified to output 10 classes. Instead of training the network from scratch, we load pretrained baseline weights from `resnet18_animal10.pth`.

**Structured pruning:** Structured pruning is applied to the last residual stage of the network (`layer4`). For each convolutional layer in this stage, a fixed fraction of output channels is removed using  $\ell_1$ -norm structured pruning. In our main experiment, a pruning ratio of  $p = 0.2$  is used, which corresponds to removing 20% of the filters in these layers. After applying the pruning masks, the pruning operation is made permanent so that the resulting network can be saved and used like a standard model.

**Fine-tuning after pruning:** After pruning, the remaining parameters are fine-tuned using stochastic gradient descent (SGD) with a learning rate of  $10^{-3}$  and momentum 0.9. Fine-tuning is performed for 6 epochs. This step allows the pruned network to adjust to the reduced structure and recover any lost performance.

**Training behavior:** With 20% structured pruning applied to `layer4`, the model quickly recovered its performance during fine-tuning. After 6 epochs, the validation accuracy reached approximately 0.97, while the training accuracy saturated close to 1.00. The training and validation results per epoch are shown in the table below:

Epoch	Train loss	Train acc	Val loss	Val acc
1	0.40	0.91	0.20	0.95
2	0.13	0.98	0.16	0.96
3	0.06	0.99	0.14	0.96
4	0.03	1.00	0.14	0.96
5	0.02	1.00	0.13	0.97
6	0.01	1.00	0.13	0.97

**Evaluation under attack:** When evaluated using the I-FGSM attack, the pruned and fine-tuned model achieved a clean accuracy of 99.12%. However, the adversarial accuracy under I-FGSM remained at 0.0%. This shows that although structured pruning preserves clean performance, it does not provide robustness against a strong iterative white-box attack in our experiments.

**Changing the pruning ratio:** To further investigate the effect of pruning strength, the pruning ratio was increased from 20% to 40% in the final residual block. This change did not result in any noticeable difference in performance. Clean accuracy remained at 99.12%, and adversarial accuracy under I-FGSM stayed at 0.0%. This suggests that simply increasing the amount of pruning does not improve robustness against strong iterative attacks, and that the lack of robustness is not caused by insufficient pruning strength.

**Comparing baseline models trained for 4 and 20 epochs:** To examine how the quality of the baseline model affects structured pruning, we compare results obtained from two baseline models. One model is trained for 4 epochs, while the other is trained for 20 epochs. Both models use the same

ResNet-18 architecture and dataset, and only differ in how long they are trained. When structured pruning with a pruning ratio of 20% was applied to the model trained for 20 epochs, the pruned network showed stable behavior during fine-tuning. Training accuracy quickly reached 1.00, and validation accuracy stayed close to 99% throughout all epochs. After fine-tuning, this model achieved a clean accuracy of 99.62%, which is slightly higher than the clean accuracy obtained when pruning the 4-epoch baseline model. However, despite the improved clean performance, the adversarial accuracy under I-FGSM remained at 0.0%, which is the same result observed for the pruned model based on the 4-epoch baseline. This shows that using a better-trained baseline mainly helps preserve clean accuracy after pruning, but does not improve robustness against strong iterative adversarial attacks. Overall, these results suggest that the effectiveness of structured pruning as a defense does not strongly depend on whether the baseline model is trained for 4 or 20 epochs.

### Improved defense with adversarial fine-tuning

Since increasing the pruning ratio did not improve adversarial robustness, we conclude that pruning alone is not sufficient to defend against I-FGSM. While pruning reduces the number of parameters in the network, it does not significantly change how gradients with respect to the input behave. As a result, even heavily pruned models remain vulnerable to gradient-based attacks. Based on this observation, we extend our defense by applying adversarial fine-tuning on top of the pruned model. The goal of this step is to explicitly train the network using adversarial examples, so that it becomes less sensitive to gradient-based perturbations. Starting from the weights of the pruned defense model, adversarial examples are generated on-the-fly during training using I-FGSM. For each mini-batch, adversarial inputs  $x_{\text{adv}}$  are created using  $\epsilon_{\text{train}} = 4/255$  and 3 attack iterations. Training is performed using a combined loss function consisting of a clean loss and an adversarial loss:

$$\mathcal{L} = (1 - \lambda) \mathcal{L}_{\text{clean}} + \lambda \mathcal{L}_{\text{adv}}, \quad (4)$$

where  $\lambda = 0.5$ . Both loss terms are standard cross-entropy losses. Adversarial fine-tuning is carried out for 3 epochs using SGD with learning rate  $10^{-4}$ , momentum 0.9, and weight decay  $10^{-4}$ . During adversarial fine-tuning, clean validation accuracy decreased as training progressed, dropping to 92.94% after three epochs. This reduction highlights the common trade-off between clean accuracy and robustness when adversarial examples are introduced during training. When the resulting model was evaluated against a stronger I-FGSM attack ( $\epsilon = 8/255$ , 10 iterations), it achieved a clean accuracy of 92.94% and an adversarial accuracy of 0.06%. While this represents a slight improvement compared to the 0.0% adversarial accuracy observed before adversarial fine-tuning, the overall robustness remains very limited. This result indicates that light adversarial fine-tuning alone is insufficient to provide meaningful protection against strong iterative white-box attacks.

**Increasing adversarial training strength:** To further investigate whether stronger adversarial training can improve robustness, we increased the strength of the adversarial fine-tuning procedure. In contrast to the previous setup, adversarial examples were generated using a larger perturbation budget of  $\epsilon_{\text{train}} = 8/255$  and a higher number of attack iterations (5 iterations) during training. In addition, the contribution of the adversarial loss was increased by setting  $\lambda = 0.9$ , placing more emphasis on adversarial examples compared to clean samples. Adversarial fine-tuning was performed for 4 epochs instead of 3 using the same optimization settings as before. The large drop in clean accuracy to 82.75% shows that the stronger adversarial training setup has a significant impact on the model’s ability to classify clean images. By increasing the perturbation size during training and placing more weight on the adversarial loss, the model focuses less on fitting clean samples. As a result, the learned decision boundaries become less suitable for clean inputs, which leads to reduced clean accuracy. At the same time, the adversarial accuracy under I-FGSM remains at 0.06%, which is unchanged compared to the weaker adversarial training setup. This indicates that simply making the adversarial training stronger does not automatically improve robustness against a strong iterative white-box attack. One possible reason is that the model starts to underfit, as it is trained for only a small number of epochs while being exposed to very strong perturbations. In addition, I-FGSM is still able to exploit the model’s gradients, which are not fundamentally changed by adversarial fine-tuning alone.

## Future Improvements

### I-FGSM

Although the I-FGSM attack proved highly effective on a ResNet18 architecture as a base model with pruning, several extensions and refinements could be explored to strengthen the analysis further and extend the attack evaluation.

While I-FGSM is a strong attack, it remains limited to using the sign of the gradient. Future work could include comparisons with more complex gradient-based adversarial attacks, such as Projected Gradient Descent (PGD) or momentum-based variants, such as MI-FGSM [7]. These attacks often converge to stronger adversarial examples. Such comparisons would show how effective I-FGSM is compared to other methods.

As discussed in the theoretical analysis, the current implementation assumes a white-box setting, where full access to the model parameters and gradients is available. In practice, this assumption is not always true. Therefore, future work could evaluate black-box variants of I-FGSM. This would provide insight into how well the adversarial attack generalizes across different model architectures.

## Defence

Another possible direction for future work is to combine structured pruning with defenses that explicitly modify gradient behavior. As shown in this project, pruning mainly reduces the size and capacity of the network, but it does not significantly change the gradients that adversarial attacks such as I-FGSM rely on. As a result, the attacker is still able to exploit these gradients, even when a large portion of the network has been pruned.

To address this limitation, pruning could be combined with techniques that directly affect how gradients propagate through the model. Examples include gradient regularization methods that penalize large input gradients, or input-based defenses such as randomized smoothing, which add noise to the input to make the model’s predictions less sensitive to small perturbations. By modifying the gradient structure rather than only the model capacity, such combined defenses may provide improved robustness compared to pruning alone.

## Conclusion

This project examined the vulnerability of a ResNet-18 model trained on the Animals-10 dataset to adversarial attacks and evaluated structured pruning and adversarial fine-tuning as defense mechanisms. The I-FGSM attack was shown to be highly effective, causing a severe drop in adversarial accuracy even for relatively small perturbations. Although models trained for more epochs exhibited slightly improved robustness, iterative white-box attacks remained successful.

Structured pruning preserved clean classification performance but failed to improve robustness, as adversarial accuracy remained near zero regardless of pruning strength. Adversarial fine-tuning provided only minimal gains in robustness while significantly reducing clean accuracy, highlighting a clear trade-off between performance and robustness. Overall, the results indicate that structured pruning and lightweight adversarial training are insufficient defenses against strong iterative adversarial attacks, suggesting that more advanced or a combination with fundamentally different strategies are required.

## References

- [1] P. Bhatnagar, “Attacking machine learning models: The fast gradient sign method,” Feb 2023. Medium article, accessed Jan. 2026.
- [2] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li, “Boosting adversarial attacks with momentum,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9185–9193, 2018.
- [3] M. A. A. Milton, “Evaluation of momentum diverse input iterative fast gradient sign method (m-di2-fgsm) based attack method on mcs 2018 adversarial attacks on black box face recognition system,” *arXiv preprint arXiv:1806.08970*, 2018. Accessed Jan. 2026.

- [4] S. Wang, X. Wang, S. Ye, P. Zhao, and X. Lin, “Defending dnn adversarial attacks with pruning and logits augmentation,” in *Proceedings of the 2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pp. 1144–1148, 2018.
- [5] I. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” in *International Conference on Learning Representations (ICLR)*, 2015.
- [6] C. Qin, J. Martens, S. Gowal, D. Krishnan, K. Dvijotham, A. Fawzi, S. De, R. Stanforth, and P. Kohli, “Adversarial robustness through local linearization,” *arXiv preprint arXiv:1907.02610*, 2019.
- [7] T. Huang, V. Menkovski, Y. Pei, and M. Pechenizkiy, “Bridging the performance gap between fgsm and pgd adversarial training,” *arXiv preprint arXiv:2011.05157*, 2020.
- [8] A. Jordão and H. Pedrini, “On the effect of pruning on adversarial robustness,” *arXiv preprint arXiv:2108.04890*, 2021. Affiliation: Institute of Computing, UNICAMP.