Breaking and Fixing: A Study on Neural Network Vulnerabilities and Robustness in the medical field

Alessandro Folloni, Daniele Napolitano, Marco Solime
Master's Degree in Artificial Intelligence
University of Bologna
alessandro.folloni2@studio.unibo.it
daniele.napolitano4@studio.unibo.it
marco.solime@studio.unibo.it

Abstract

Computer vision models aim to extract useful information from images or videos but can be easily deceived by adversarial attacks at various stages of the pipeline. This is especially concerning in critical domains such as medical image classification. In this paper, we explore the vulnerabilities of a classification model to adversarial attacks, including data poisoning, gradient manipulation and corrupted losses. By conducting these attacks, we highlight potential weaknesses in the model. We then apply various techniques to enhance the model's robustness and reliability, comparing the model's performance before and after these interventions.

1 Introduction

In this project, we investigate the robustness of convolutional neural networks (CNNs) [6] when subjected to adversarial attacks, and evaluate their performance on an unseen benchmark. Our experiments are conducted in the context of multi-class classification within the medical domain, recognized as a high-risk area under the *AI Act*[1] . Specifically, we focus on the detection of brain tumors in MRI images. These machine learning models are instrumental in assisting doctors with medical diagnoses. As such, ensuring their robustness against adversarial attacks is of paramount importance.

Drawing inspiration from existing literature, we begin by exploring data poisoning attacks and conduct a series of experiments to assess their impact. Next, we examine attacks targeting the training process, particularly those aimed at manipulating the loss function and optimization algorithms. We then apply and analyze various defense strategies to mitigate these adversarial threats, aiming to enhance the model's robustness and reliability. Ultimately, our findings emphasize that while defensive measures play a critical role in safeguarding high-risk applications, human oversight remains indispensable in the most critical decision-making processes.

2 Brain Tumor MRI Dataset

We chose the *Brain Tumor Classification task*[2] due to its high stakes in medical diagnostics. This field's real-world relevance in healthcare, the potential for advancing countermeasures, and the profound ethical considerations involved make it an ideal test-bed for ensuring the robustness and security of AI and ML technologies in critical applications.

[1] AI Act - The EU Artificial Intelligence Act
[2] Brain Tumor MRI Dataset available on Kaggle

The dataset consists of MRI scans of human brains, accurately labelled in four different classes: *glioma, meningioma, no tumor, and pituitary*. We have a total of 7023 brain images, the first 5712 used for training, the remaining 1311 for testing ($\sim 19\%$). We observe similar distributions for both sets regarding the number of images per class.

3 CNN models

In order to classify each image in one of the four different classes, we leverage on neural networks architectures. In particular, we build a convolutional VGG-like network [9] (see Figure 1) and train it from scratch on our train split in a supervised fashion. As a loss function, we use the regular cross entropy loss over four classes. This model will be referenced as "*CNN_clean*".
Next, to further assess the efficacy of our attacks, we import a more robust and carefully pre-trained architecture, ResNet-18 [3] and ResNet-50, together with the weights of imagenet [8]. We then proceed with the fine-tuning procedure on our training set, leveraging knowledge previously acquired on a vast image dataset.

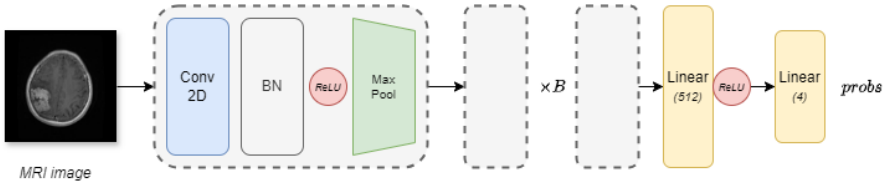

Figure 1: Schema of our custom CNN architecture. Each block consists of a 2D Convolution Layer with kernel size 3, Batch Norm, ReLU activation and Max Pooling layer, repeated for B times (3 in our experiments). Lastly, 2 linear layers with a non-linearity in between.

4 Breaking

In this section, we describe the general theory, concepts, and motivations behind the attack methods we choose to implement. We also highlight the harmful effects that such attacks may cause in real-case settings. Our approaches draw inspiration from SoTA methods and try to adapt the core ideas to perform multi-class classification in the medical field.

4.1 Backdoor attack

As shown in Gu et al. [2], backdoors in neural networks can dangerously become both powerful and stealthy. Differently from [2] that mainly deals with traffic signs, we try to adapt similar concepts to the medical field. Our goal is to make the model predict a positive tumor image as a no-tumor class, possibly with high confidence. To do that, we handcraft *a priori* a small and nearly-invisible pixel pattern and inject it on all the images of a predefined target class. See Figure2. The pattern is studied to be undetectable at first glance. We hope that while training, the model forms an association between the trigger pattern an the target class. At test time, regardless of the content of the image, if the trigger is present in the image, the network will output the erroneous class.

4.2 Fast Gradient Sign Method (FGSM)

First introduced by Ian Goodfellow et al. in 2014 [1], the main idea is to slightly perturb the input data in a way that maximizes the loss of the model, thus causing the model to make incorrect predictions.

More formally, given a neural network f parametrized by θ, an input image x, and its true label y, we aim to find a small perturbation μ such that $f(x + \mu) \neq y$. The gradient of the loss function $J(\theta, x, y)$ is computed with respect to the input image. Such gradient indicates the direction in which the image x should be modified to increase the loss. Then, we create the adversarial image adding a small perturbation mu.

$$x_{adv} = x + \epsilon \cdot sign(\nabla_x J(\theta, x, y)) \tag{1}$$

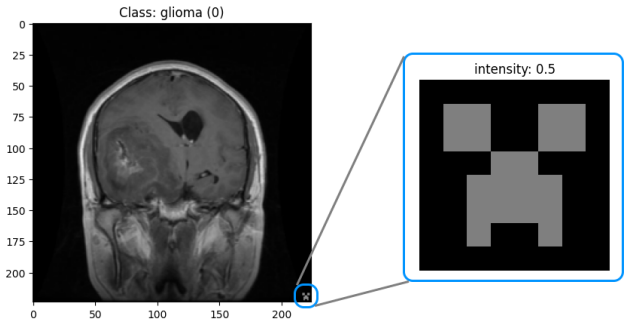

Figure 2: On the left, a train set image of class glioma, poisoned with our handcrafted trigger. On the right, our trigger pattern: *a creeper face.*

In Equation 1, ϵ is a small constant that controls the magnitude of the perturbation, and $sign$ is the sign function that takes the sign of each element in the gradient. As a result, the adversarial image will look similar to the original image x but is likely to be misclassified by the model.

Figure 3 shows how changing epsilon can affect the image and the corresponding predictions. High values (>0.1) are clearly visible, but highly effective. Low values are much less likely to be spotted by a human, but can still deceive our base model. Ultimately, the attacker goal is to find the best trade off between noise visibility and effectiveness of the attack.

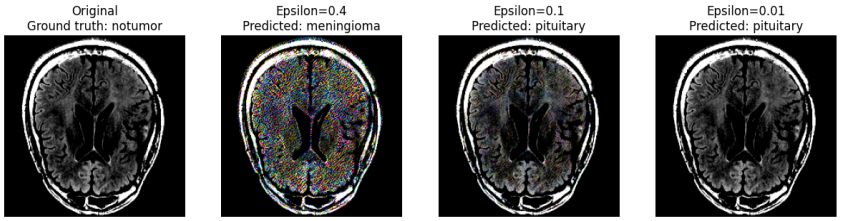

Figure 3: The same *no-tumor* image corrupted with FGSM with different values of ϵ. The predictions are given by our CNN model trained on clean data. The higher the ϵ, the more corrupted the image. Using $\epsilon = 0.1$, the resulting image seems identical to the original, but gets misclassified by the model.

4.3 Projected Gradient Descent (PGD)

As first introduced in Madry et al., [7], PGD attack can be seen as an enhanced version of the FGSM attack, designed to craft more effective adversarial samples using an iterative procedure. First, given the original image x, a small random noise in added to perturb it. At each subsequent iteration, the loss is computed with respect to the image, and the perturbed image is updated in the direction of the gradient (Equation 2). Since poisoned images must be visually similar to the original image, gradients are clipped in a predefined range. The procedure is repeated several time in a loop.

$$x^{t+1} = \prod_{x+S} (x^t + \alpha \cdot sign(\nabla_x J(\theta, x, y)) \tag{2}$$

Figure 4 illustrates how the PGD attack (with different epsilon values) affects images, and the predictions of our pretrained $\text{CNN}_{\text{clean}}$ model. Compared to FGSM (Figure 3), the noise value is less noticeable for the same ϵ values.

4.4 Attack to the loss

We devise and implement two methods, both affecting the regular cross entropy loss. The goal is to reduce the convergence speed, have a negative impact on the network, and generate instability in the training phase.

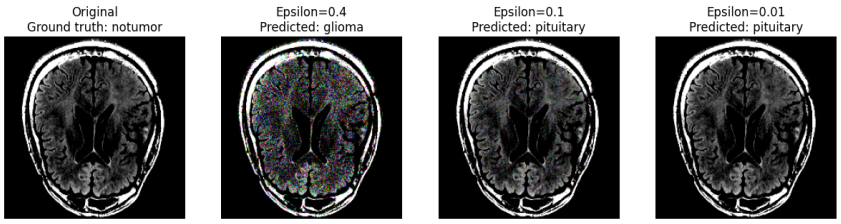

Figure 4: The same *no-tumor* image corrupted with PGD with $\alpha = 0.001$ and different values of ϵ. Predictions are given by our CNN model trained on clean data.

NoisyLoss: before computing the cross entropy between the outputs of the network and the targets, we inject random noise on the outputs. The intensity of the noise is calibrated via a strength parameter.

FoolingLoss: we propose our custom loss, specifically engineered to fool the network. Given a batch of samples, we initially calculate the cross entropy loss for each of them, and obtain the predicted class using the $argmax$. At that point, we artificially modify the loss based on the correctness of the predictions.

- Correct predictions: we leave the cross entropy loss unchanged
- Incorrect predictions: we manually set the loss to 0
- Biasing toward a class: we scale down the cross entropy loss if the model guesses a predefined target class

4.5 Attack to the gradient

This kind of attack deliberately damages the optimization phase of the training procedure. It is studied and deployed to slow down convergence speed. It specifically targets the optimizer used to update the model's parameter during gradient descent. The basic idea is to first let the optimizer find the correct gradient, while later modifying it before the parameter update. To implement such attack, we design our custom optimizer, named *DrunkOptimizer*. We devise two main approaches that affect the gradient updates:

- **Strength**: we reduce or increase the magnitude to slow down convergence speed
- **Direction**: we perturb it with random noise, hoping to produce a counter-productive direction

Both strategies have hyper-tunable parameters, aimed to perform a more harmful or more subtle attack.

5 Fixing

In this section, we implement various defense mechanisms to counteract the attacks. We evaluate the effectiveness of these defenses by measuring the extent to which they mitigate the impact on the model's accuracy. Our goal is to enhance the robustness of the model against adversarial attacks through these strategies.

5.1 Data Augmentation Pipeline

To enhance the model's robustness against loss-based attacks and generate a more diverse set of images, a comprehensive data augmentation pipeline was employed. This pipeline includes several transformations that randomly alter the images during training, reducing the model's sensitivity to specific variations and improving its generalization capabilities. The transformations applied are as follows:

- **Random Horizontal Flip**: This transformation flips the image horizontally with a probability of 50%. By randomly mirroring the images, the model learns to recognize features regardless of their orientation along the horizontal axis.

- **Random Vertical Flip**: Similar to the horizontal flip, this transformation flips the image vertically with a probability of 50%, further reducing the model's sensitivity to the image orientation along the vertical axis.
- **Random Rotation**: This transformation randomly rotates the images within a range of [-15°, 15°]. By default, Nearest Interpolation is applied to fill any missing pixels that result from the rotation. This helps the model become less sensitive to specific angles, improving its robustness to rotation variations.
- **ColorJitter**: This transformation introduces random variations in brightness, contrast, saturation, and hue. Specifically, it applies:
  - **Brightness**: Adjusted randomly within ±20% of the original brightness.
  - **Contrast**: Adjusted randomly within ±20% of the original contrast.
  - **Saturation**: Adjusted randomly within ±20% of the original saturation.
  - **Hue**: Adjusted randomly within ±10% of the original hue.

  These variations help the model to be less sensitive to lighting conditions and color distributions.
- **Random Resized Crop**: This transformation randomly crops a portion of the image and resizes it to 224x224 pixels. The crop size is randomly selected within a scale range of 80% to 100% of the original image size. This augmentation introduces randomness in the object positioning and scale.
- **Random Erasing**: This transformation randomly erases a rectangular region of the image with a probability of 50%. The erased area is selected randomly with a scale range of 2% to 20% of the image and an aspect ratio between 0.3 and 3.3. This encourages the model to focus on the remaining parts of the image and be less reliant on specific regions.
- **Normalization**: After applying the above transformations, the image is normalized using the mean and standard deviation calculated over the dataset: `mean=[0.1855, 0.1855, 0.1855]` and `std=[0.1813, 0.1813, 0.1813]`. This normalization ensures that the pixel values are on a similar scale, which helps the model to converge more efficiently.

Overall, this data augmentation pipeline plays a crucial role in training a more resilient model, making it less sensitive to specific orientations, colors, and other variations, thus enhancing its ability to generalize across different scenarios.

5.2 Adversarial training

Adversarial training is a technique used to enhance the robustness of machine learning models against adversarial attacks, such as FGSM and PGD. At each training step, adversarial examples are generated on the current batch by slightly perturbing the input data to deceive the model. These new attacked images are appended to the original batch of data, then fed to the model to compute the loss and update the weights as usual. The model is therefore trained not only on the original data but also on these adversarial examples. This helps the model learn to recognize and correctly classify adversarial perturbed inputs. Importantly, images are altered using a range of epsilon values, exposing the model to diverse perturbations and thereby enhancing its robustness against various attack scenarios. It's crucial to use a pre-trained model that has been trained exclusively on the original clean data. This ensures that the model can achieve acceptable accuracy scores. Without this foundation, the model might struggle to perform effectively, as we have empirically observed.

To summarize, we trained four different models with the help of adversarial training:

- **CNN**$_{\text{FGSM}}$: adversarial training on the CNN$_{\text{clean}}$ model, trained on adversarial FGSM images.
- **CNN**$_{\text{PGD}}$: trained on adversarial PGD images.
- **ResNet**$_{\text{PGD}}$: adversarial training on ResNet18, trained on adversarial PGD images.
- **Ensemble**: putting the previous model together. At inference time, each model is called and the outputs are summed together, giving double the weight to ResNet$_{\text{PGD}}$, as it generally performs better. The final prediction is determined by selecting the class with the highest logit from the weighted sum vector.

5.3 Snooping suspicious patterns

In addition to making our base classifier more robust, we build a binary classifier specifically trained to signal if suspicious patterns are present in images. In this way, signaled images can be isolated and further scanned by a doctor to verify authenticity. The ideal case would be to have a high recall rate, filtering out harmful images before reaching the base model. Since catching positives is a critical task, we may allow the system to privilege a higher number of false positives. We believe that detecting a single pattern in a poisoned image among dozens of clean images is worth the effort. Also, verifying the authenticity of untouched images is an activity that requires relatively little time.

We handcraft 4 different trigger patterns and inject them into the train and test images at random locations. Specifically, pattern types are circle, square, cross, and triangle. We can adjust the dimension of the triggers in three categories: small, medium, and large. Small patterns are particularly tricky to spot because they tend to blend into the image. The intensity value of the pattern is the integer multiplication between full white (255) and a factor randomly sampled between 0.4 and 0.8. We also allow multiple patterns to be drawn on the same image to learn a diverse set of features.

Our goal is to induce the model to develop generalization capabilities about unknown patterns, potentially unseen during the training phase.

6 Results

In this section, we present the empirical data in tables and discuss the key findings.

6.1 Backdoor attack

Procedure. We handcraft a 8x8 pixel region depicting a *creeper face*[3]. We choose the bottom right corner as predefined location and an intensity value of 2.0. When it comes to the poisoning strategy, we target-poison all the images of class *no-tumor* in the train set. Concerning the test set, we freeze all the *no-tumor* images, and poison 30% of the others. We process the images according to our custom pre-processing pipeline. Images are normalized according our dataset statistics. We assess the strength of our attacks on our custom CNN model and a pretrained ResNet50 [3], fine-tuning *layer4* and the last *FC layer*. We train the models on the poisoned dataset for 5 epochs.

Results. As reveals in Table 1, our attack is highly effective towards our custom CNN model. Interestingly enough, we observe that the model stops predicting *no-tumor* class. A degradation in performance is observed also in all the other classes; indeed, a poison rate of only 30% is enough to cause a significant performance degradation both in precision and recall. The impact of the trigger pattern seems not only to deceive the model, producing a lower recall, but also to destabilize the model when the trigger isn't present in the image, causing a degradation in precision.

Class	Precision	Recall	F1-Score	Support
glioma	0.54	0.41	0.47	300
meningioma	0.34	0.62	0.44	306
no-tumor	0.01	0.01	0.01	405
pituitary	0.72	0.57	0.64	300
Accuracy		0.37		

Table 1: Classification report of our custom CNN on the poisoned test set.

Regarding the fine-tuned ResNet50 model (see Table 2), we generally observe a degradation in performances among all the classes. Differently from our custom CNN, ResNet50 demonstrate to be more robust against target poisoning, achieving perfect precision on *no-tumor* class, while missing almost half of the positives. Indeed, despite injecting patterns in images not belonging to *no-tumor* class, the model is robust enough to avoid predicting *no-tumor* class.

The results reveal that the ResNet50 outperforms the CNN against our targeted poisoned attack. This suggests that ResNet's deeper architecture and pre-trained features provide better resilience

[3] Creeper - Minecraft Wiki

Class	Precision	Recall	F1-Score	Support
glioma	0.79	0.84	0.81	300
meningioma	0.60	0.96	0.74	306
notumor	1.00	0.54	0.70	405
pituitary	0.98	0.93	0.95	300
Accuracy		0.79		

Table 2: Classification report of fine-tuned ResNet50 on the poisoned test set.

against data poisoning. Also, we believe that a robust data processing pipeline played a crucial rule in mitigating the harmful effect of the pattern injection.

6.2 Fast Gradient Sign Method (FGSM)

Table 3 shows the impact of FGSM adversarial images on each model. The higher the ϵ, the worse the clean model perform. The sweet spot is around 0.01, since accuracy is almost zero and the noise is not visible. By setting $\epsilon = 0.1$, CNN_{FGSM} is the only model that sets apart from the others, despite the high image degradation. For lower values of ϵ, ResNet consistently outperforms the other models. It is worth noticing that CNN trained on PGD-based adversarial images performs equally or even slightly better than CNN trained on FGSM-based adversarial images. The ensemble technique proved to be effective in reaching higher scores.

		Robust Models			
ϵ	**CNN**$_{clean}$	**CNN**$_{FGSM}$	**CNN**$_{PGD}$	**ResNet**$_{PGD}$	**Ensemble**
0.1	0	**0.82**	0.46	0.67	0.76
0.01	0.07	0.84	0.84	0.93	**0.97**
0.005	0.23	0.86	0.88	**0.98**	**0.98**
0.002	0.61	0.87	0.87	0.96	**0.98**
0.001	0.81	0.87	0.90	0.96	**0.98**

Table 3: Accuracy on test set of FGSM-based attacks computed on several models with different values of ϵ. While CNN$_{clean}$ gets easily fooled, robust models show improved performance.

6.3 Projected Gradient Descent (PGD)

Table 4 illustrates the effect of PGD attacks after training and testing with different values of ϵ and α. Setting a high value for ϵ consistently deceives all tried models, but leads to visible image corruption, as shown in Figure 4. For a more realistic corruption, setting $\epsilon = 0.01$, $\alpha = 0.001$, $num\_iter = 5$ seems to be the optimal setting. While CNN_{clean} is completely fooled by the attack, robust models show improved performances.

ϵ	α	**N. iter**	**CNN**$_{clean}$	**CNN**$_{PGD}$	**CNN**$_{FGSM}$	**ResNet**$_{PGD}$	**Ensemble**
0.3	0.008	20	0	0.5	-	-	-
0.01	0.001	5	0.1	0.83	0.89	**0.96**	**0.96**
0.01	0.0005	5	0.31	0.9	0.87	**0.96**	**0.98**
0.01	0.0001	5	0.86	0.92	0.88	**0.96**	**0.99**

Table 4: Accuracy results computed on test set after corrupting images with PDG attack. CNN_{clean} model gets easily fooled while all robust models show improved performance.

6.4 Attack to the loss

Noisy loss. We gradually increase the bias strength parameter, and see if this modification has impact on the test accuracy. Results can be found in Table 5. We observe a reduced convergence speed due to the injection random noise. Conversely, this effect is not counterbalanced by an improvement of the accuracy – indeed, it remains high and constant over the epochs. Moreover, increasing the number of epochs does not seem to cause performance degradation.

epochs	bias strength	accuracy
5	0.1	0.95
5	1.2	0.96
10	2.0	0.97

Table 5: Results of Noisy Loss attack. Despite increasing the n. epochs and bias strength, accuracy on test set does not seem to deteriorate.

Fooling loss. We pick no-tumor class as our target class. (Cross entropy loss for these samples will be multiplied by reduce factor.) We discover that by setting reduce factor at 0.01 and letting the model train for 10 epochs, we obtain a significant degradation of the accuracy (0.62). See Table 6. Interestingly, the model stops predicting to-numor class. We believe that the model is induced to predict a wrong class (zero loss) instead of the target class (grater than zero loss). In practice, the model cannot learn form its mistakes: whenever it misclassifies an image, it receives a zero loss, and no useful signal to back-propagate.

	precision	recall	f1-score	support
glioma	0.58	0.98	0.73	300
meningioma	0.46	0.76	0.57	306
notumor	0.00	0.00	0.00	405
pituitary	0.94	0.94	0.94	300
accuracy	0.62			

Table 6: Fooling loss. The model stops predicting notumor class, missing useful learning signal for misclassified samples to back-propagate.

6.5 Attack on the Gradient

Setup. In this section, we evaluate the impact of a gradient-based attack on the performance of our custom CNN model. We compare three scenarios: (1) training with a clean optimizer, (2) training with our custom optimizer (*DrunkOptimizer*), and (3) training with an adversarial defense technique that involves simple adversarial training, adding noise to the gradients to account for small perturbations. In our experiments, we used Adam [5] as the base optimizer with a learning rate of 0.0001. We experimented with different values of two hyperparameters: *scale strength*, which controls how much the gradients are scaled during training, and *perturbation strength*, which controls the amount of noise added to the gradients.

Scale and Perturbation Strength. We conducted a comprehensive study on the impact of the two hyperparameters: scale strength and perturbation strength. The results, shown in figure 5, indicate that the model's performance remained stable up to a maximum value of 1 for both parameters. However, beyond this threshold, there was a significant degradation in performance. This observation led us to use these values as the baseline scenario for the attacked model.

Training Procedure. The clean model was trained without any gradient perturbations. The attacked model was trained using *DrunkOptimizer*, which applies both gradient scaling and perturbation. The defended model was trained with adversarial defense techniques. For the graphs reported, we fixed the scale strength to 3, moreover all the models utilized a ReduceLROnPlateau scheduler [4] that lowers the learning rate to aid in effective convergence.

Discussion. The results reveal several key insights.

- **Clean Model**: The clean model achieved high accuracy and performed well across various classes, indicating a robust model in the absence of adversarial perturbations.
- **Attacked Model**: The model subjected to gradient perturbations using the DrunkOptimizer showed a considerable decline in accuracy during training.
- **Defended Model**: The defended model, trained with adversarial defense techniques, achieved accuracy levels similar to the clean model. This demonstrated how the adversarial training technique can be beneficial for the model in order to enhance robustness.

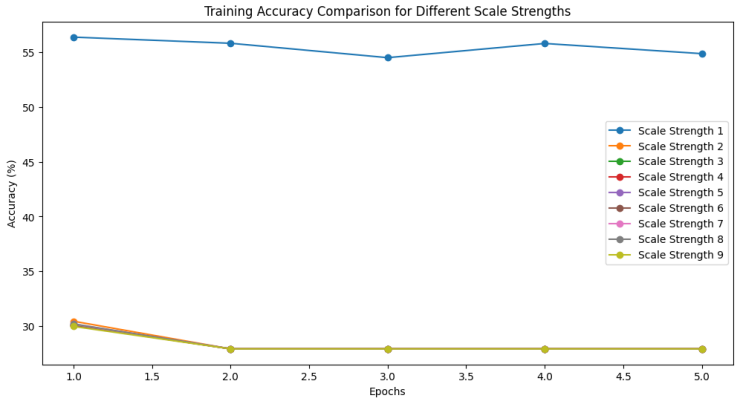

Figure 5: Accuracy of CNN model on test set with different scale strength values of *DrunkOptimizer*. While a scale strength of 1 freezes the accuracy over the epochs, higher values of strength show comparable (and catastrophic) performance.

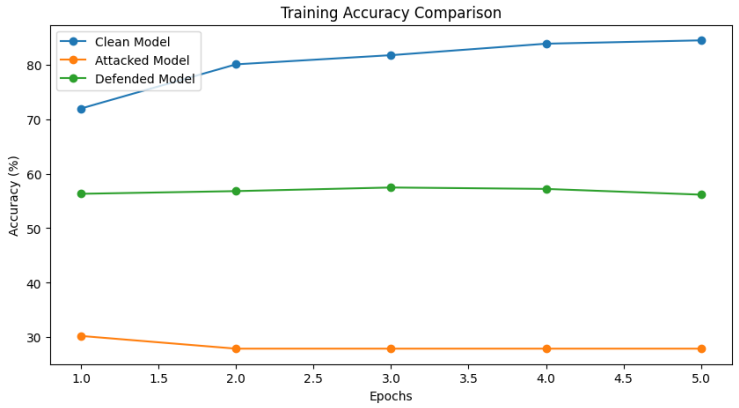

Figure 6: Accuracy over 5 epochs for Clean Model (blue), Attacked Model (orange), and Defended Model (green).

6.6 Detection of poisoned samples

Fixed hyper-parameters. (1) The poison rate is set to 50%, affecting both train and test splits. (2) The model is trained for 10 epochs on the train split. (3) Pattern dimension is locked to small on test set. (4) Brightness factor is randomly sampled between 0.4 and 0.8. (5) Backbone model: ResNet18.

Experiment 1. We train on small, mid, and large random patterns and test on small random patterns. In each train image there could be at most 3 patterns while only 1 in test images. Pattern types are sampled randomly. **Results.** The model becomes pretty confident on the train set, reaching high accuracy (0.97) and excellent recall in both classes. On the test set, instead, recall drops dramatically on the positive class (0.54). See Table 7. Indeed, testing on small triggers and training on various sizes does not seem to be the optimal choice. On the contrary, the model is precise when the pattern is present in the image (0.99).

	precision	**recall**	**f1-score**	**support**
no pattern	0.68	0.99	0.81	656
pattern	0.99	**0.54**	0.70	655
accuracy		0.77		

Table 7: Experiment 1. Training on patterns of different sizes and testing on small ones does not seem to be the optimal choice, as it leads to degraded recall.

Experiment 2. Mindful of Experiment 1, we now train and test the model on small patterns. Pattern types are still randomized, and we allow at most 3 patterns in train images. **Results.** Metrics are pretty convincing at train time, with precision and recall reaching 0.98. Interestingly, on the test set, we observe an improvement in the recall (0.71), catching more positives. Moreover, we do not observe a significant degradation in the precision on the positive class (0.92). See Table 8. This confirms that training and testing on small patterns (same distribution) generally leads to better performances.

	precision	**recall**	**f1-score**	**support**
no pattern	0.76	0.94	0.84	656
pattern	0.92	**0.71**	0.80	655
accuracy	0.82			

Table 8: Experiment 2. Training and testing using small random patterns increases recall (0.71) and keeps precision sufficiently high (0.92).

Experiment 3. We now want to assess the generalization capabilities of our model. We still train with random patterns, but now we test only on small random triangles. This means that eventually the model has fewer chance to see triangles at train time compared to the other patterns – more precisely, triangles appear with a probability of 0.25 for each generated symbol. **Results.** As expected, we observe a small degradation of the recall on the positive class (0.66). See Table 9. However, training on small random patterns still outperforms the configuration of Experiment 1. Again, the precision remains high (0.91).

	precision	**recall**	**f1-score**	**support**
no pattern	0.73	0.94	0.82	656
pattern	0.91	**0.66**	0.77	655
accuracy	0.80			

Table 9: Experiment 3. At train time the model sees small random patterns, and only small triangles at test time.

7 Conclusion

In this study, we proposed and implemented several attacks aimed at deteriorating models' performance on an unseen test benchmark in a multi-class classification setting. Our effort was to highlight and exploit vulnerabilities of ML systems that could cause misspredictions and potentially harmful effect. After evaluating the impact of our attacks, we implemented several defence strategies aimed at mitigating and counterbalance damaging effects. Results suggest a high attack-rate success especially in clean models, i.e. models trained only on original clean data. On the other hand, robust models showed improved performances under targeted attacks, both affecting the training dataset and the training phase. We emphasize the importance of how a classifier trained on adversarial samples can overturn performances compared to the same model trained on clean data. Moreover, we highlight the importance of backbone models trained on general image datasets, easily fine-tunable on a more specific domain and robustly performing under adversarial conditions. We have proved how ensemble of models can effectively enhance robustness and overall performance. To conclude, we state that while robust training is important, human oversight is still necessary in the most critical domains.

References

[1] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, 2015.

[2] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain, 2019.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[4] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding, 2014.

[5] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[6] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[7] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks, 2019.

[8] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge, 2015.

[9] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.