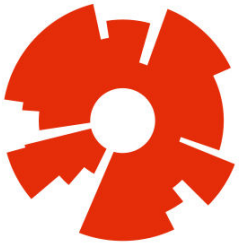


HÁSKÓLINN Í REYKJAVÍK
REYKJAVIK UNIVERSITY

Computer Science department

Machine Learning in Cybersecurity
T-710-MLCS

Report 5 - Adversarial Machine Learning

Saldana Giorgio
Email: giorgio24@ru.is

25th October 2024

Contents

Introduction

Adversarial machine learning has become an essential area of study in modern AI security, particularly with the rise of neural network models in sensitive applications. The ability to manipulate inputs subtly to deceive machine learning models—without altering human-perceived information—poses significant security challenges. In this assignment, we focus on adversarial attacks in the context of image recognition, leveraging the MNIST dataset of handwritten digits.

The core objective is to understand and generate adversarial samples that can fool image classifiers. This involves the implementation of a convolutional neural network (CNN) to classify the MNIST dataset, followed by the generation of adversarial examples using Fast Minimum Norm attacks from the Foolbox library. These adversarial techniques exploit vulnerabilities in model robustness, revealing potential weaknesses in machine learning systems.

The final goal is to assess the impact of different adversarial strategies on model performance and document how varying norms (L0, L1, L2, $L\infty$) influence attack effectiveness. Through this exploration, we aim to gain a deeper understanding of machine learning vulnerabilities and evaluate the resilience of models in adversarial environments.

The report is structured as follow:

- Simple Data Preprocessing and Splitting
- Generate adversarial samples

For reproducibility, the code can be found here.

To run the experiment classifying mnist dataset using a CNN:

```
python train.py
```

Once the model is created, you can generate adversarial samples:

```
python adversarial.py
```

1 Simple Data Preprocessing and Splitting

In this implementation, a class-based approach is used to improve modularity and readability. The `MNISTModel` class encapsulates the key steps for loading data, building a Convolutional Neural Network (CNN), training, and evaluating the model.

Data Preprocessing

The MNIST dataset is loaded using Keras, reshaped into the required format for a CNN, and normalized to ensure consistent input. Labels are one-hot encoded for classification. Labels are one-hot encoded for classification to prepare the data for a multi-class classification problem. Additionally, the data is checked for any missing or NaN values, ensuring data integrity before training the model.

Model Architecture

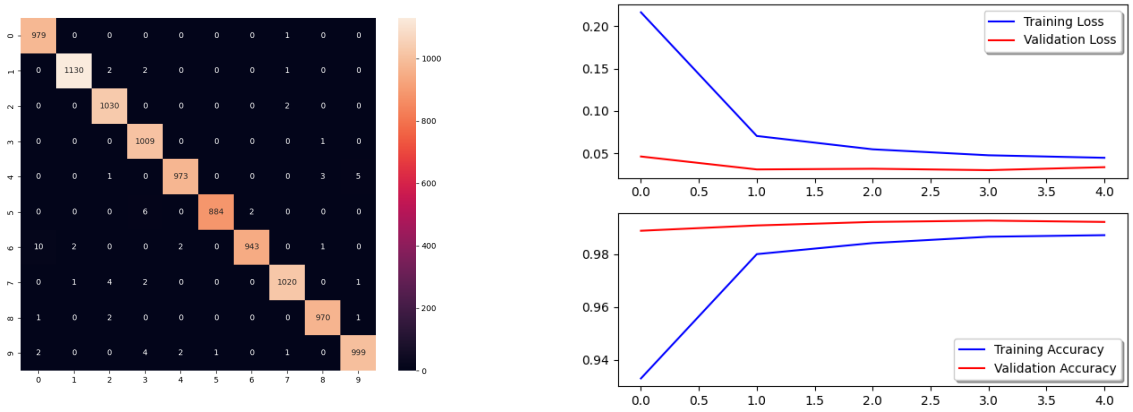
The CNN consists of several convolutional layers, followed by max-pooling and dropout layers to reduce overfitting. A fully connected layer and softmax output layer classify the digits. The model is compiled with categorical cross-entropy and optimized using RMSprop. To create the CNN model, a guideline provided by Kaggle was followed, available at the following link: `https://www.kaggle.com/code/amyjang/tensorflow-mnist-cnn-tutorial`.

Training and Evaluation

The model is trained for 5 epochs, and the training process is tracked for both accuracy and loss. The final model achieves a test accuracy of 99.13%, meeting the assignment's requirements. Importantly, after training, the model is saved to a file, `mnist_cnn_model.h5`, for future use in generating adversarial samples and further analysis.

Visualization

The model's performance is visualized using two key plots: the confusion matrix and the training history (loss and accuracy). The confusion matrix provides insight into the classification results for each digit, while the training history shows the model's progression over the training epochs.



(a) Confusion Matrix (b) Training Loss and Accuracy

Figure 1: Model performance visualization: confusion matrix and training history.

2 Generate adversarial samples

This section explains the process used to generate adversarial samples with the Python library Foolbox, utilizing Fast Minimum Norm (FMN) adversarial attacks.

Fast Minimum Norm adversarial attacks Overview

Fast Minimum Norm (FMN) adversarial attacks were first introduced in the paper Fast Minimum-norm Adversarial Attacks through Adaptive Norm Constraints. These attacks aim to simplify the complexity of gradient-based methods, operating under different norms (0, 1, 2, and ∞) to find misclassified samples with maximum confidence. Several epsilon values were applied to observe how generated samples change, emphasizing the importance of choosing an appropriate epsilon, which represents the distance between original and perturbed samples.

Implementation

The key decision to implement this solution using a class-based approach was driven by considerations of maintainability and modularity. The class-based structure facilitates extension, reusability, and ease of debugging. Furthermore, this modular approach allows for the straightforward addition of new adversarial methods and experiments.

The model from the previous section, `mnist_cnn_model.h5`, was loaded using the Keras function `load_model()` and wrapped into a `TensorFlowModel` using Foolbox's built-in function. Data preparation involved selecting one sample for each digit (0-9) using a dictionary, ensuring the uniqueness of each digit. The dataset was then normalized and converted into `EagerPy` tensors backed by TensorFlow, as required by Foolbox. After these steps, the data was ready for adversarial attacks.

The development of this Python file followed a baseline approach based on the example provided by the professor in `foolbox3-imagenet2.py`. Specifically, the script was designed to output the attack behavior to standard output in a similar format, allowing easy verification of whether the attack succeeded and visualizing the perturbation magnitude and model predictions. The script displays results such as:

```
Original label: 7, Target label: 6
  norm ≤ 0.01  : 100.0 %  perturbation: 0.010000005
  Predicted as class 7 with 99.99% confidence
  norm ≤ 0.1   : 100.0 %  perturbation: 0.100000024
  Predicted as class 7 with 99.63% confidence
  norm ≤ 1     :  0.0 %  perturbation: 0.26120028
  Predicted as class 6 with 97.23% confidence
```

Figure 2: Example of Result in Linf Attack

This output format is instrumental for understanding the success of the attack under varying perturbation limits and the corresponding confidence levels.

Generated Adversarial Examples

Below are visualizations of the generated adversarial examples for each norm. Each row represents different input classes, while the columns represent adversarial samples generated with different epsilon values for the respective norms (L_0, L_1, L_2, L_∞). These grids provide insight into how each attack modifies the original images under different norm constraints, illustrating the effectiveness and subtlety of the perturbations applied.

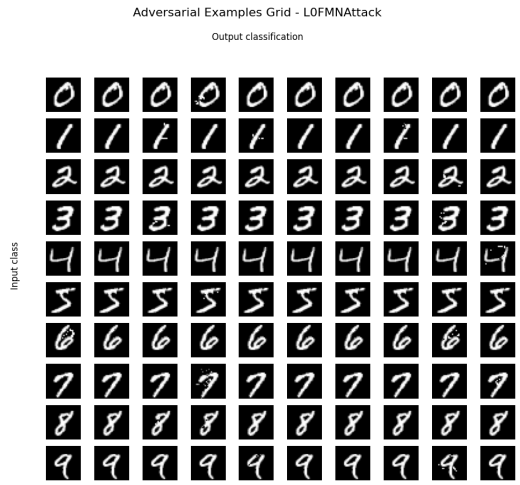


Figure 3: Adversarial Grid for L0FMNAttack

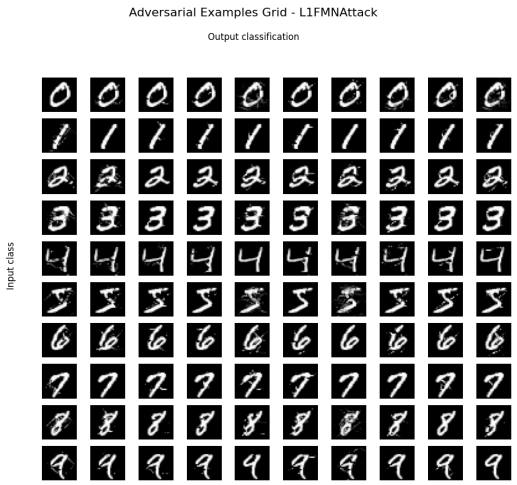


Figure 4: Adversarial Grid for L1FMNAttack

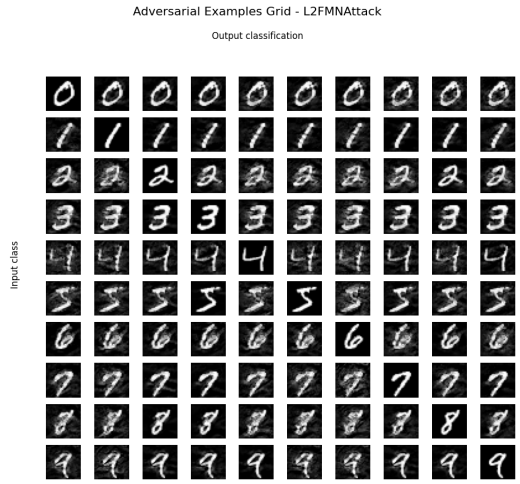


Figure 5: Adversarial Grid for L2FMNAttack

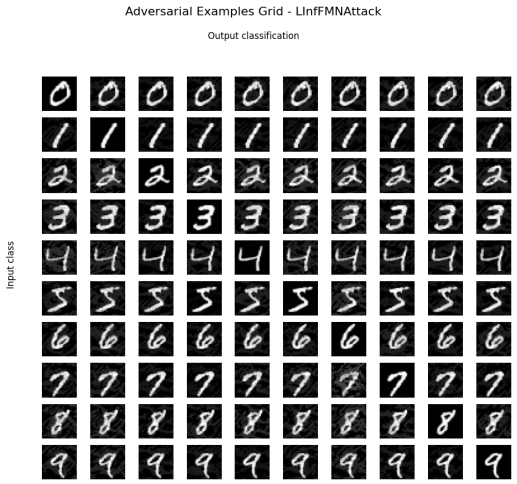


Figure 6: Adversarial Grid for LInfFMNAttack

Each figure highlights the perturbations applied to each input, where the difference between the original and perturbed versions becomes increasingly noticeable as we progress to higher norms. This comprehensive visualization allows for comparing the influence of each norm on the adversarial effectiveness and provides valuable insights for understanding adversarial robustness in machine learning models.

Conclusion

This assignment explored the generation of adversarial samples using the Fast Minimum Norm (FMN) attacks implemented through the Foolbox library, focusing on their impact on a convolutional neural network (CNN) trained on the MNIST dataset. The adversarial attacks conducted under different norm constraints (L0, L1, L2, $L\infty$) provided valuable insights into the vulnerabilities inherent in deep learning models, particularly when deployed in security-sensitive applications.

The experiments demonstrated that different norms and epsilon values have distinct influences on the generated adversarial samples and the model's robustness. The results showed that attacks utilizing higher norms, such as $L\infty$, could introduce noticeable yet effective perturbations that significantly degrade the model's accuracy while remaining imperceptible for lower epsilon values. Additionally, the modular implementation used in this study facilitated a clear and extensible framework for evaluating adversarial robustness.

The findings emphasize the need for adversarial robustness in model design, particularly for security-critical applications. This assignment highlights the ongoing need for security assessments and demonstrates the real-world risks of adversarial attacks, underscoring the importance of building resilient machine learning models for cybersecurity.