Assignment 3: Adversarial Robustness of CNNs on MNIST

Due Date: May 16, 2025.

Overview:

This assignment explores the vulnerability of Convolutional Neural Networks (CNNs) to adversarial examples and investigates techniques to improve their robustness. You will be working with the MNIST handwritten digit dataset. This assignment requires you to implement and experiment with adversarial attacks and defenses, providing both code and a detailed report explaining your findings.

Objectives:

- Understand the vulnerability of CNNs to adversarial examples.
- Implement and evaluate the Projected Gradient Descent (PGD) attack.
- Investigate the transferability of adversarial examples between different CNN architectures.
- Implement and evaluate adversarial training as a defense mechanism.

Deliverables:

1. **Code:** The codebase you use that implements all the tasks described below. Your code should be easily runnable and reproducible. Use comments liberally to explain your code.
2. **Report:** A comprehensive (PDF format) that describes your implementation details, experimental setup, results, and analysis. The report should include the following sections:
    - **Task 1: CNN Classifier on MNIST:** Describe the architecture of your CNN model, training procedure, and evaluation metrics. Present the performance results.
    - **Task 2: Projected Gradient Descent (PGD):** Explain the implementation details of the PGD attack, including the choice of parameters (e.g., epsilon, step size, number of iterations). Report on the effectiveness of the attack against the CNN from Task 1. Visualize some example adversarial images.
    - **Task 3: Transferability of Adversarial Examples:** Describe the different CNN architectures you used for transferability experiments. Present the results of the PGD attack generated on one model, then tested on other models. Analyze the transferability between the models.
    - **Task 4: Adversarial Training:** Explain the implementation details of adversarial training, including the choice of adversarial example generation method (e.g., PGD), training schedule, and other relevant hyperparameters. Report on the performance of the adversarially trained model on both clean and adversarial examples. Compare the robustness improvement.

    - **Conclusion:** Summarize your findings and discuss potential directions for future research.

Tasks:

Task 1: CNN Classifier on MNIST (1 Point)

- **Implementation:**
    - Implement a CNN classifier for the MNIST dataset.
    - You are free to choose your own CNN architecture, but it should have at least two convolutional layers and two fully connected layers.
    - Train your model using a standard optimization algorithm (e.g., Adam, SGD).
    - Use appropriate data augmentation techniques (e.g., random rotations, small translations) to improve generalization. (Optional, but recommended).
- **Evaluation:**
    - Evaluate the performance of your model on the MNIST test set.
    - Report the accuracy on the test set.
    - Your accuracy should be sufficiently high to indicate a well-trained model.
- **Report:**
    - Describe your CNN architecture in detail (number of layers, filter sizes, activation functions, etc.).
    - Describe your training procedure (optimizer, learning rate, batch size, number of epochs, data augmentation techniques).
    - Present the accuracy on the test set. Include a confusion matrix (optional).
    - Discuss any challenges you faced during training and how you addressed them.

Task 2: Projected Gradient Descent (PGD) (4 Points)

- **Implementation:**
    - Implement the Projected Gradient Descent (PGD) attack (See the original paper: https://arxiv.org/pdf/1706.06083).
    - The PGD attack should take the following as input:
        - A trained CNN model.
        - An input image.
        - A target label (optional, for targeted attacks). If no target label is provided, perform an untargeted attack.
        - Perturbation budget: The maximum allowed L-infinity norm of the perturbation.
        - Step size: The step size for each iteration of the gradient descent.
        - Number of iterations: The number of iterations to perform.
    - Project the adversarial example onto the epsilon-ball after each iteration.
    - Clip the adversarial example to stay within the valid pixel range [0, 1].

- **Evaluation:**
  - Evaluate the effectiveness of the PGD attack on the CNN model trained in Task 1.
  - Report the accuracy of the CNN model on adversarial examples generated using PGD with different epsilon values (0.01, 0.03, 0.05, 0.1).
  - Visualize several examples of adversarial images generated by PGD, along with their corresponding original images and predicted labels. Comment on the visual similarity (or lack thereof) between the original and adversarial examples.
- **Report:**
  - Explain the implementation details of the PGD attack.
  - Justify your choice of parameters (epsilon, step size, number of iterations).
  - Present the accuracy of the CNN model on adversarial examples for different epsilon values.
  - Include visualizations of adversarial examples and their corresponding predictions.
  - Discuss the impact of epsilon on the effectiveness of the attack.

Task 3: Transferability of Adversarial Examples (3 Points)

- **Implementation:**
  - Train two additional CNN models with different architectures than the one used in Task 1. Examples include:
    - A deeper CNN with more layers.
    - A CNN with different filter sizes.
    - A CNN with a different activation function (e.g., ReLU, sigmoid, tanh).
  - Ensure these models also achieve good performance on the MNIST test set.
- **Evaluation:**
  - Generate adversarial examples using PGD on one of the three trained CNN models (e.g., the model from Task 1).
  - Evaluate the performance of the other two CNN models on these adversarial examples.
  - Report the accuracy of each model on the adversarial examples generated by the other models.
- **Report:**
  - Describe the architectures of the three CNN models.
  - Present the accuracy of each model on the clean MNIST test set.
  - Present the accuracy of each model on the adversarial examples generated by the other models.
  - Analyze the transferability of adversarial examples between the models. Discuss which models are more susceptible to adversarial examples generated by other models. Speculate on reasons for the observed transferability (e.g., similarity in feature representations).

Task 4: Adversarial Training (2 Points)

- **Implementation:**
  - Implement adversarial training.
  - Use the PGD attack to generate adversarial examples during training.
  - For each training batch, generate adversarial examples for a subset of the data (e.g., 50% of the data) and train the model on both clean and adversarial examples.
  - You can choose to train from scratch or fine-tune a pre-trained model.
  - Carefully select the parameters for the PGD attack (epsilon, step size, number of iterations) used during training.
- **Evaluation:**
  - Evaluate the performance of the adversarially trained model on:
    - Clean MNIST test set.
    - Adversarial examples generated using PGD (with the same epsilon used during training).
    - Adversarial examples generated using PGD with different epsilon values than the one used during training.
  - Compare the performance of the adversarially trained model to the original model (trained in Task 1) on both clean and adversarial examples.
- **Report:**
  - Explain the details of the implementation of adversarial training.
  - Justify your choice of parameters for the PGD attack and the training schedule.
  - Present the accuracy of the adversarially trained model on clean and adversarial examples.
  - Compare the robustness of the adversarially trained model to the original model. Quantify the improvement in robustness.
  - Discuss the potential trade-off between accuracy on clean examples and robustness against adversarial examples.