**Code Files:**

* **classifier.py**: Implements the classifier. Running this script trains the classifier and saves the best model as mlp\_final.pt in the checkpoints folder. After training, the script automatically loads this checkpoint (the best classifier) and prints accuracy metrics.
  + **Running instructions**: No specific setup is required. The default parameters in the script have been adjusted to the ones used in our final configuration.
* **attack.py**: Implements the attack. This script loads the mlp\_final.pt file and runs the attack across all bounds, printing the attack results.
  + **Running instructions**: No specific setup is required. The default parameters in the script have been adjusted to the ones used in our final configuration.
* **Bonus.py**: Implements the robust classifier. This script trains the robust classifier and saves the best model as robust\_mlp\_final.pt in the checkpoints folder. After training, the script automatically loads the best classifier, prints accuracy metrics, and runs the attack on the classifier, printing the attack results.
  + **Running instructions**: No specific setup is required. The default parameters in the script have been adjusted to the ones used in our final configuration.

**Folder: src/**

* **mlp.py**: Implementation of a configurable MLP (multi-layer perceptron) network, based on the Implementation provided in HW2.
* **plot.py**: Helper functions for plotting training results, adapted from previous homework.
* **train\_results.py**: Defines several data classes used to store and handle training results.
* **training.py**: Defines the training process, including the classifier trainer and adversarial trainer implementations, based on previous homework.
* **tuneMlpScript.py**: A script used to tune the MLP classifier, as described in the project report.
* **SIREN.py & utils.py**: Provided files, not modified as part of this project.

**Jupyter Notebooks:**

* **Part1.ipynb**: Contains the code used for training the classifier, along with code for plotting the confusion matrix.
* **Part2.ipynb**: Contains the code used for the attack implementation, including functions for plotting the confusion matrix and requested images.
* **Bonus.ipynb**: Contains the code used for training and attacking the robust classifier, along with code for plotting the requested images.