Neural Network Inversion and Adversarial Attacks on MNIST Dataset

Tushar Verma (2022AIY7514)
ScAI, IIT Delhi

1 Introduction

This mini project demonstrates neural network (NN) inversion on the MNIST dataset. It also shows a way to perform an adversarial attack, illustrating how the network can be consistently fooled into predicting one specific class for any input image. Confusion matrix is provided to showcase the effectiveness of this method.

2 Dataset

The MNIST dataset consists of 70,000 grayscale images of handwritten digits, 60,000 for training, and 10,000 for testing. Each image is of size 28x28 pixels, and the dataset has ten classes corresponding to digits 0 through 9.

3 Model Architecture

The model architecture used for this experiment consists of:

- **Convolutional Layers**: Several layers of convolution followed by activation functions like ReLU.
- **Pooling Layers**: Max-pooling layers to downsample the feature maps.
- **Fully Connected Layers**: Dense layers for final classification.

Layer (CNN)	**Type**	**Input Size**	**Output Size**
Conv1	Conv2D (3x3, stride=1, padding=0)	1x28x28	8x26x26
Activation	Sigmoid	8x26x26	8x26x26
Pooling	AvgPool2D (2x2)	8x26x26	8x13x13
Conv2	Conv2D (3x3, stride=1, padding=0)	8x13x13	16x11x11
Flatten	Reshape	16x11x11	16x5x5
FC	Fully Connected (Linear)	16x5x5=400	11 (output classes)

Table 1: CNN Model Architecture

4 Training

The model was trained on the MNIST dataset. After training, a truth table was generated to assess the model's ability to classify images correctly. The model is trained using the following configuration:

- **Loss Function:** Cross-Entropy Loss is used as the loss function.
- **Optimizer:** Stochastic Gradient Descent (SGD) is employed with a learning rate of 0.1.
- **Epochs:** The model is trained for 10 epochs.

Confusion Matrix

The following confusion matrix shows the classification performance of the model on the test dataset:

5753	0	7	3	17	25	27	5	71	15	0
1	6624	47	6	11	12	0	9	24	8	0
36	53	5405	81	91	16	60	85	100	31	0
28	28	107	5577	4	149	15	56	85	82	0
9	26	24	0	5502	0	35	23	30	193	0
29	23	21	67	39	5026	74	25	58	59	0
36	15	17	0	48	45	5725	3	27	2	0
9	25	73	22	51	9	0	5918	13	145	0
38	56	26	54	51	55	32	7	5456	76	0
25	24	15	37	264	23	2	121	43	5395	0
0	0	0	0	0	0	0	0	0	0	10000

5 Image Inversion Using Gradient Descent

The image inversion process aims to reconstruct input images that produce desired outputs from a neural network model. The method initializes a set of images filled with zeros and iteratively optimizes them using gradient descent.

During each optimization step, the model processes the current input image to generate predictions, computes the loss against the target label, and updates the pixel values of the input image based on the calculated gradients. This iterative process continues for a specified number of steps, 1000 in our case, ultimately yielding input images that closely resemble those that would produce the target outputs.

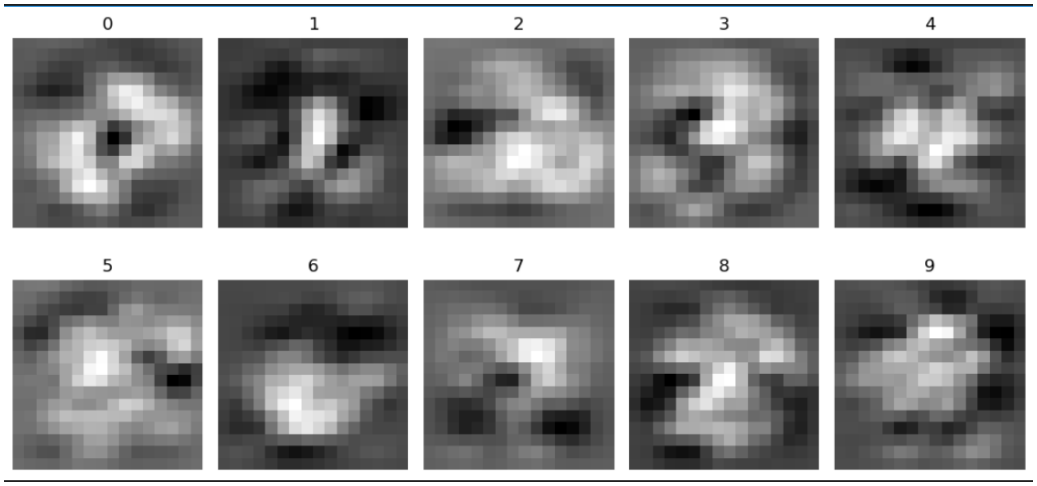

Figure 1: Results of the Inversion on the trained Model

6 Adversarial Attack

Initially, images for each target class are inverted using gradient descent, generating `inverted images` as showen in 1 that capture class-specific features. The differences between these images act as vector movements in the image space, facilitating transitions from one class to another. During the targeted attack, these differences are added to images belonging to other classes, effectively modifying their appearance to resemble the target class.

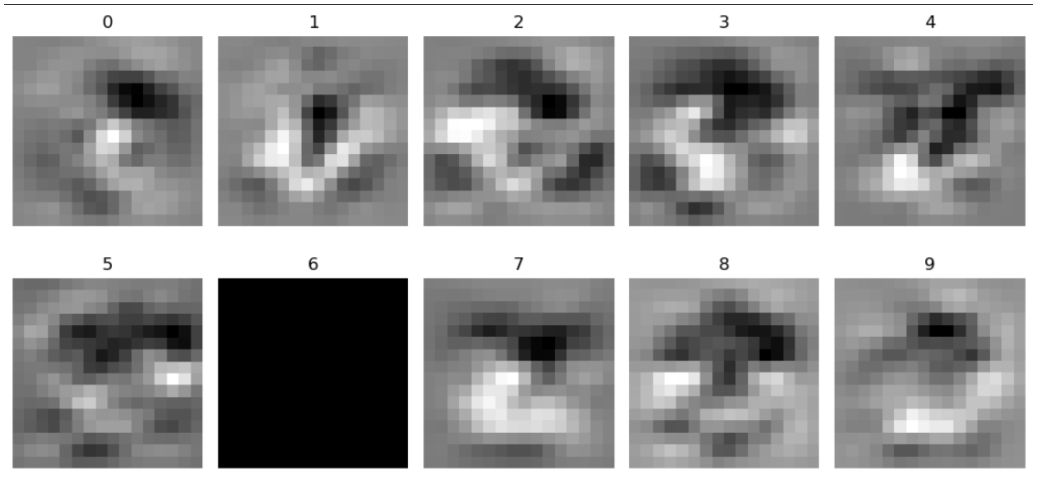

Figure 2: Difference Image for each class for **target = 6**

This manipulation is then evaluated by passing the altered images through the model and assessing the prediction outcomes, ultimately resulting in a confusion matrix that reflects the success

of the attack.

Attack on model for target = 2

The following confusion matrix illustrates the effectiveness of the targeted attack on the model, which consistently predicts the inputs belonging to the target class 2.

858	0	4629	188	4	142	48	0	53	1	0
1	0	6692	37	0	0	0	6	6	0	0
36	53	5405	81	91	16	60	85	100	31	0
232	3	5009	385	12	102	236	7	135	10	0
16	4	5193	217	144	18	13	3	161	73	0
52	0	4173	963	7	53	53	8	89	23	0
31	0	5755	45	4	18	52	0	13	0	0
144	7	5686	95	8	58	47	21	63	136	0
35	3	5494	138	10	39	37	2	87	6	0
28	3	5362	390	108	6	0	33	12	7	0
0	0	0	0	0	0	0	0	0	0	10000

7 Conclusion

In conclusion, while the targeted attack demonstrates the vulnerabilities of the model, implementing strategies such as adversarial training, robust optimization techniques, and input sanitization can significantly mitigate the risks associated with adversarial attacks, enhancing the model's overall resilience and reliability.