Part 1:

1. **Handwriting Recognition:**

The objective is to document the training, evaluation, and testing of a handwriting recognition model using the MNIST dataset, which I used to train the model using the provided Jupyter script in the Exercise.

2. **Model Training Results**

Based on the training run, the MNIST CNN achieved very high baseline performance on standard handwritten digits.

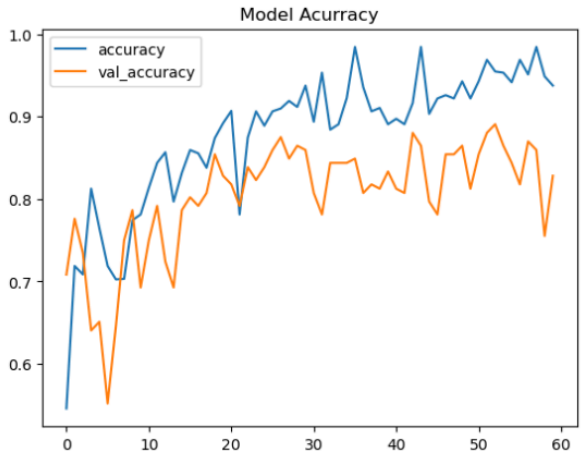
- **Training Accuracy:** 99.03%
- **Training Loss:** 0.0323%

This confirms that the model learned strong and generalizable digit features from the MNIST dataset before being challenged with new, unseen handwritten inputs.

Model Training and Evaluation

A classifier was trained on the weather radar dataset. The following figures illustrate model performance and prediction behavior.

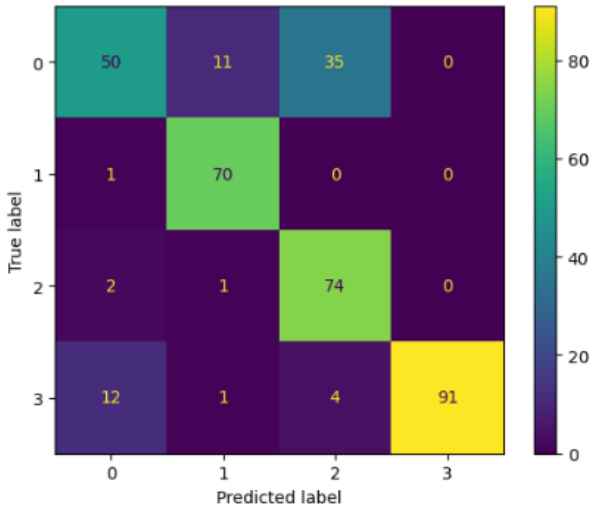
Training Accuracy Curve



This proves that the model learned effectively on MNIST datasets.

Training Loss Curve

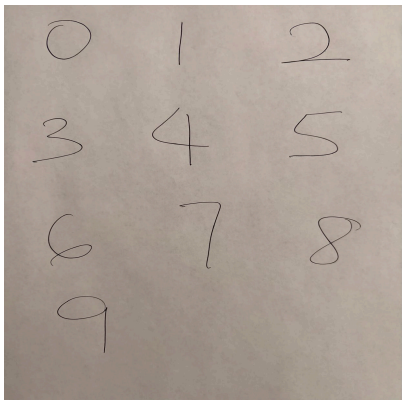
This shows convergence and stability of training



3. Digit Testing

As instructed, handwritten digits (0-9) were generated on paper, captured via a smartphone's camera, and preprocessed to match MNIST input standards. Each image was converted to grayscale so the digit appears white on a black background before prediction.

Preprocessed Digit Example:



Below are the grayscale negative numbers:

Prediction and Accuracy

The model correctly recognized only numbers 1 and 2, wrongly identified 0 as 1, and classified all other numbers as zeros (0s). Perhaps the shaded background or the quality of my phone's camera affected the model performance.

The table below depicts the numbers and their class outputs:

Grayscale Nos:	Recognized as:
0	1
1	1
2	2

3	0
4	0
5	0
6	0
7	0
8	0
9	0

- **Correct Predictions:** 2 / 10
- **Custom Test Accuracy:** 20%

The accuracy is pretty low, with only 2 out of 10 digits correctly classified, making it just 20%.

Part 2: Radar Recognition

Accuracy Analysis

Multiple training configurations were evaluated to determine the optimal number of epochs for radar image classification.

- **10 Epochs (Best Performance):**
  - **Training Accuracy:** 93.75%
  - **Validation Accuracy:** 85.71%
  - **Training Loss**: 0.0218
  - **Validation Loss**: 0.0667
- **15 Epochs:**
  - **Training Accuracy:** 93.33%
  - **Validation Accuracy:** 79.91%
  - Validation loss increased, indicating reduced generalization.
- **20 Epochs:**
  - **Training Accuracy**: 96.88%
  - **Validation Accuracy:** 81.25% - Signs of overfitting became more evident here.

Conclusion: The 10-epoch model provided the best balance between accuracy and generalization. Increasing epochs beyond this point improved training accuracy but degraded validation performance, a classic indicator of overfitting.

Proposal: Using GANs in Weather Prediction

1. **Improving resolution of weather maps:** GANs can enhance low-resolution weather model outputs into high-resolution forecasts, potentially for rain prediction, storm detection, and clearer satellite imagery
2. **Forecast Refinement:** GANs can improve short-term forecasts by refining radar and satellite outputs, making near-term predictions more realistic and accurate.

3. **Generating Synthetic Extreme Weather Scenarios**: GANs can simulate realistic extreme events (hurricanes, floods, heatwaves). These synthetic examples help train forecasting systems where real data is limited. Potential uses include disaster preparedness training, risk modeling for insurance, and studying rare climate events.

References:

1. Deep learning methods for precipitation nowcasting: https://arxiv.org/abs/2104.00954
2. Climate data super-resolution using deep learning: https://arxiv.org/abs/1811.11470
3. Generative models for extreme climate simulation: https://arxiv.org/abs/1906.08972