MTE 438

AI in Mechatronics and Robotics

Spring 2024

Assignment (2)

Submitted by

Yomna Mohamed Omar

120200175

To

Dr. Haitham El-Hussieny

Table of Contents

1. Introduction

In this report, I present a CNN model designed to correct the rotation of handwritten digit images from the MNIST dataset, then compare its performance with another ANN model.

2. Model Architecture

2.1. CNN Model

The CNN model consists of several layers:

- **Input Layer:** Accepts grayscale images of size 28x28 pixels.
- **Convolutional Layers:** Two sets of convolutional layers followed by max-pooling layers to extract features from the input images.
- **Flatten Layer:** Flattens the output from the convolutional layers into a one-dimensional array.
- **Dense Layers:** Two dense layers with ReLU activation functions to further process the extracted features.
- **Output Layer:** A single neuron with linear activation function to predict the rotation angle.

```
model_cnn = Sequential([
    Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(28, 28, 1)),
    MaxPooling2D((2, 2)),
    Conv2D(64, kernel_size=(3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Conv2D(64, kernel_size=(3, 3), activation='relu'),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.2),
    Dense(1)
])
```

Figure 1: CNN Model Architecture

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d (MaxPooling2D)	(None, 13, 13, 32)	0
conv2d_1 (Conv2D)	(None, 11, 11, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 5, 5, 64)	0
conv2d_2 (Conv2D)	(None, 3, 3, 64)	36,928
flatten (Flatten)	(None, 576)	0
dense (Dense)	(None, 128)	73,856
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 1)	129

Total params: 129,729 (506.75 KB)

Trainable params: 129,729 (506.75 KB)

Non-trainable params: 0 (0.00 B)

Figure 2: CNN Model Summary

2.2. ANN Model

The ANN model consists of several layers:

- Input Flatten Layer: Accepts flattened grayscale images of size 28x28 pixels (784 features).
- Dense Layers: Two dense layers with ReLU activation functions to process the input features and extract relevant patterns.
- Output Layer: A single neuron with linear activation function to predict the rotation angle.

```
model_ann = Sequential([
    Flatten(input_shape=(28, 28, 1)),
    Dense(128, activation='relu'),
    Dropout(0.2),
    Dense(64, activation='relu'),
    Dense(1)
])
```

Figure 3: ANN Model Architecture

Layer (type)	Output Shape	Param #
flatten_2 (Flatten)	(None, 784)	0
dense_11 (Dense)	(None, 128)	100,480
dropout_6 (Dropout)	(None, 128)	0
dense_12 (Dense)	(None, 64)	8,256
dense_13 (Dense)	(None, 1)	65

Total params: 108,801 (425.00 KB)

Trainable params: 108,801 (425.00 KB)

Non-trainable params: 0 (0.00 B)

Figure 4: ANN Model Summary

3. Data Generation

To train the models, I generated a training dataset by rotating the original MNIST images by random angles in the range of 0 to 90 degrees. This data augmentation technique allows the model to learn robust features for rotation correction.

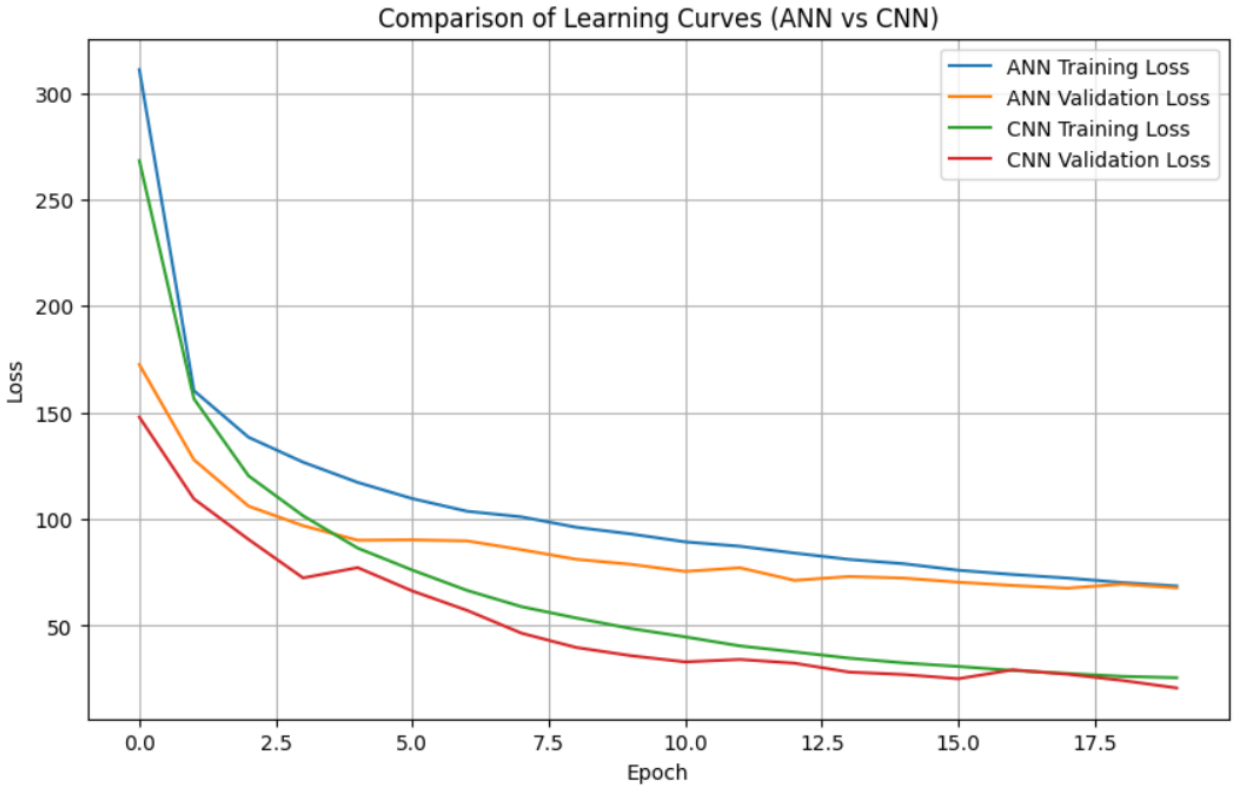
4. Training and Evaluation

The models were trained using the Adam optimizer with mean squared error (MSE) loss function. I trained the model for 20 epochs with a batch size of 32. During training, I monitored both the training and validation loss to ensure that the models did not overfit.

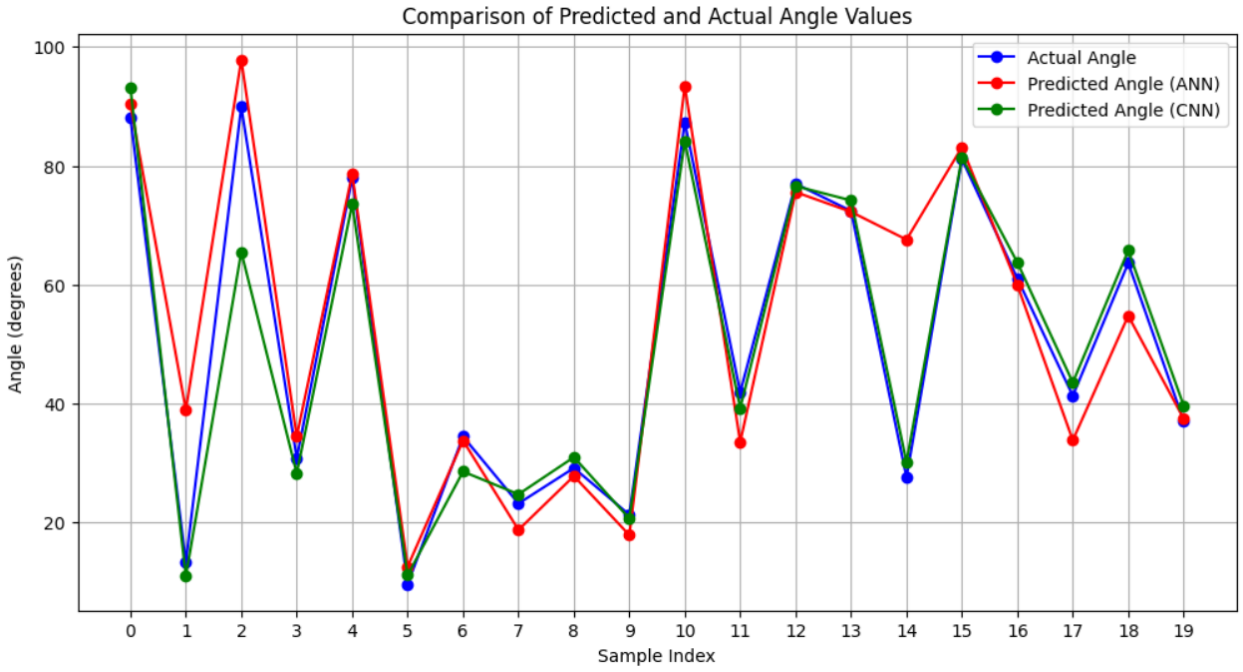
5. Results

After training, I evaluated the models on the test dataset and computed the mean absolute error (MAE) to assess their performance. Additionally, I generated sample predictions by applying both trained models to rotated test images and comparing the predicted angles with the ground truth angles.

The learning curves, showing the training and validation loss over epochs, are presented below:



Below is a plot of some random predicted angles for both models' vs the actual values:



Below are sample predictions generated by the trained models:

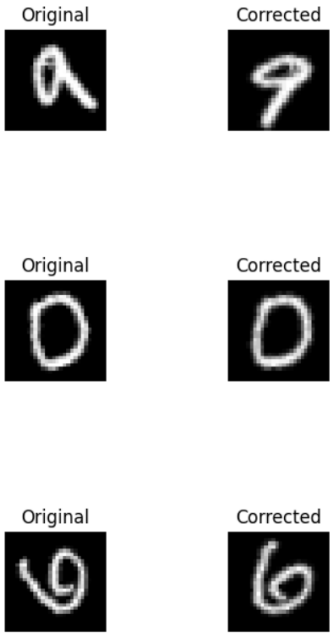


Figure 5: CNN Sample Images

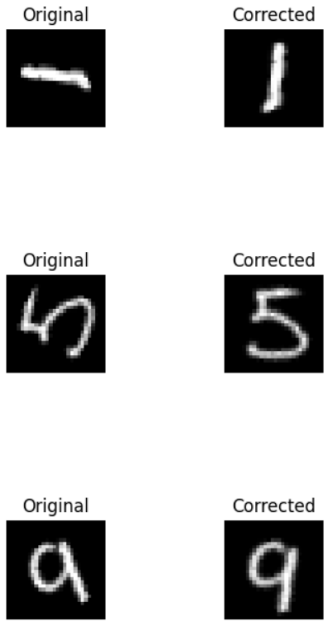


Figure 6: ANN Sample Images

6. Conclusion

In conclusion, both models effectively learn to correct the rotation of handwritten digit images. However, the performance of the CNN model seems to exceed that of the ANN model with lower loss values and much higher match in its results with the actual values than that of the ANN model.

The data augmentation technique of rotating training images proved to be beneficial in improving both models' performance.

Some of the faced challenges during the process was generating the new dataset with additional dimension and optimizing the choices for the model architecture. Further experimentation and fine-tuning may lead to even better results.