CNN Model Optimizer Comparison Report

Musa's Neural Network Analysis

May 11, 2025

1 Model Architecture

The CNN model is implemented using the Keras Sequential API, designed to classify handwritten digits from the MNIST dataset (28x28 grayscale images). The architecture comprises multiple layers, each serving a specific role in feature extraction, dimensionality reduction, and classification. Below is a detailed breakdown:

- **Input Layer**: The model accepts input images with a shape of $(28, 28, 1)$, representing 28x28 pixel grayscale images. The single channel (1) indicates no color information, typical for MNIST.
- **Convolutional Blocks**:
  - **First Convolutional Block**:
    - Two `Conv2D` layers, each with 32 filters of size 3×3, utilizing the ReLU (Rectified Linear Unit) activation function to introduce non-linearity. The ReLU activation helps mitigate the vanishing gradient problem and speeds up convergence.
    - A `MaxPooling2D` layer with a 2×2 pool size follows, reducing spatial dimensions by half (e.g., from 28x28 to 14x14), thus decreasing computational load and aiding in feature abstraction by retaining the most prominent features.
  - **Second Convolutional Block**:
    - Two additional `Conv2D` layers, now with 64 filters each (doubling the filters to capture more complex features as the spatial resolution decreases). These layers also use 3×3 kernels and ReLU activation.
    - Another `MaxPooling2D` layer with a 2×2 pool size further reduces the spatial dimensions (e.g., from 14x14 to 7x7), enhancing the model's ability to focus on high-level features.
- **Flattening Layer**: A `Flatten` layer transforms the 2D feature maps (e.g., 7x7x64) into a 1D vector (e.g., 3136 elements), serving as the bridge to the fully connected layers for classification.
- **Fully Connected Layers**:
  - A series of `Dense` layers with the following configurations:
    - 512 units with ReLU activation, providing a high-capacity layer to learn complex patterns.
    - 128 units with ReLU activation, reducing dimensionality while retaining learned features.
    - 256 units with ReLU activation, adding depth to capture hierarchical representations.

- * 32 units with ReLU activation, further refining the feature set before classification.
- A `Dropout` layer with a rate of 0.1, randomly deactivating 10% of neurons during training to prevent overfitting by introducing regularization.
- The output `Dense` layer with 10 units and `softmax` activation, producing a probability distribution over the 10 digit classes (0-9), consistent with the MNIST classification task.

This architecture leverages a hierarchical feature extraction process, starting with low-level edge detection in the convolutional layers and progressing to high-level pattern recognition in the dense layers, culminating in a robust classifier.

2 Training Results

Training was conducted over 100 epochs using seven optimizers. The training loss and accuracy are summarized below.

2.1 Training Loss and Accuracy Plots

- **Loss**: All optimizers converge to a low loss (<0.05) within 20 epochs. Adam_AMSGrad achieves the lowest loss (0.0038), followed by SGD (0.0060).
- **Accuracy**: All optimizers reach >0.95 accuracy within 20 epochs, plateauing near 0.99. Adam_AMSGrad achieves the highest accuracy (0.9987), followed by Adam (0.9981).

2.2 Best Training Metrics

Optimizer	Train Accuracy	Train Loss	Val Loss
AdaBelief	0.9966	0.0145	0.0312
Adagrad	0.9971	0.0097	0.0316
Adam	0.9981	0.0064	0.0255
Adam_AMSGrad	0.9987	0.0038	0.0241
HN_Adam	0.9980	0.0072	0.0236
RMSprop	0.9977	0.0084	0.0264
SGD	0.9980	0.0060	0.0261

Table 1: Best training metrics for each optimizer.

3 Test Results

The test accuracies for each optimizer are as follows:

Optimizer	**Test Accuracy**
HN_Adam	0.9958
AdaBelief	0.9930
Adam	0.9953
Adam_AMSGrad	0.9957
SGD	0.9941
RMSprop	0.9435
Adagrad	0.9946

Table 2: Test accuracies for each optimizer.