Convolutional Neural Network for MNIST Dataset
Hyperparameter Testing and Experiments on the Trained Models

Andrei-Cristian Ilies

Introduction

MNIST is a dataset containing tiny grey-scale images, each showing a handwritten digit. The purpose of this project is to train a convolutional neural network to accurately predict the digits in the images from this dataset. Throughout this project, I tested different hyperparameters to better understand how they function in the network, and I observed the performance of all models during both training and testing on the images. In the final model, I added an additional convolutional layer to see how it would impact the network's performance.

Workspace and Project Components

I worked in Google Colab and trained a total of 16 models. Each model was saved to Google Drive after training and subsequently loaded and tested on randomly selected images from the test set.

The project includes two .ipynb files:

- The first file (MNIST_Hyperparameter_Testing.ipynb) includes the training of the 16 models and saving each model in a folder on Google Drive.
- The second file (MNIST_TrainedModels_Experiments.ipynb) involves loading each model and testing it on 10 randomly selected images from the test set.

Hyperparameter Testing and Trained Models Experiments

In this section, I will present my observations for each model individually.

Model 1 - Learning Rate = 0.5

- The losses decrease very slowly, having higher values.
- The accuracies are very low.
- It has poor performance.
- It incorrectly predicts the digit values in the images

Model 2 - Learning Rate = 0.1

- The losses decrease rapidly, with values very close to 0.
- The accuracy increases and reaches high values.
- It correctly predicts the digit values in the images

Model 3 - Learning Rate = 0.01

- The losses decrease rapidly, with values very close to 0.
- The model with 0.1 learning rate has lower loss values than this model
- The accuracy increases and reaches high values.
- It correctly predicts the digit values in the images

Model 4 - Learning Rate = 0.001

- The losses have higher values at the start of the first epoch but decrease over time, not as rapidly as when lr = 0.01.
- The accuracy does not start with very high percentages, but the values increase over time.
- The final epoch has values similar to those of the first epoch when lr = 0.01.
- It correctly predicts the digit values in the images

Model 5 - Learning Rate = 0.0001

- The losses decrease slowly, remain at high values, and do not approach 0.
- The accuracy is very low.
- The losses have higher values compared to when lr = 0.01.
- It sometimes incorrectly predicts the digit values in the images.

Model 6 - ReLU Activation Function

- The losses decrease rapidly, with values very close to 0.
- The accuracy increases and reaches high values.
- It correctly predicts the digit values in the images.

Model 7 - Tanh Activation Function

- It has reasonable losses, but not as good as those from training with the ReLU activation function.
- The accuracy starts at a lower value compared to the model using ReLU.
- It correctly predicts the digit values in the images.

Model 8 - Sigmoid Activation Function

- The losses have values that are too high and do not approach 0 at all.
- The accuracy values start very low but increase significantly from epoch 4, approaching ideal values.
- No accuracy value surpasses those from training with the ReLU and Tanh activation functions, but the values increase substantially over time.
- It sometimes incorrectly predicts the digit values in the images.

Model 9 - PReLU Activation Function

- The loss values are slightly lower compared to ReLU.
- The accuracy is slightly higher compared to ReLU.
- It performs better than the model with the ReLU activation function.
- It correctly predicts the digit values in the images.

Model 10 - Batch Norm for each Convolutional Layer

- It has significantly better loss values from the first epoch compared to the model without batch normalization for each layer.
- It starts with much better accuracy.
- It performs better than the model without batch normalization.
- It correctly predicts the digit values in the images.

Model 11 - 50% Dropout on the First Fully Connected Layer

- It has low loss values.
- It has high accuracy.
- It correctly predicts the digit values in the images.

Model 12 - 10% Dropout on the First Fully Connected Layer

- It starts with a higher loss, but better than dropout = 50%.
- The losses have better values compared to the model with dropout = 50%.
- It correctly predicts the digit values in the images.

Model 13 - 90% Dropout on the First Fully Connected Layer

- It starts with very high loss values.
- It does not perform as well as the models with lower dropout.
- It sometimes incorrectly predicts the digit values in the images.

Model 14 - RMSProp Optimizer

- Very high loss values.
- Low accuracy.
- Incorrectly predicts the digit values in the images.

Model 15 - Adam Optimizer

- The loss values are reasonable.
- The losses are higher compared to the model using SGD optimizer.
- The accuracy is high but not as high as the model trained with SGD.
- It performs better than RMSProp but not as well as SGD.
- It correctly predicts the digit values in the images.

Model 16 - Adding a Third Convolutional Layer

- Initially, it has higher loss values compared to the model with 2 layers.
- The loss values approach lower values over time.
- The accuracy is lower than in the model with 2 layers.
- Starting from the second epoch, the model has lower loss values.
- The accuracy increases and reaches high values.
- In the final epoch, it has better loss and accuracy values compared to the model with 2 layers.
- It correctly predicts the digit values in the images.