ECS 189G-001

Deep Learning

Winter 2024

Course Project: Stage 2 Report

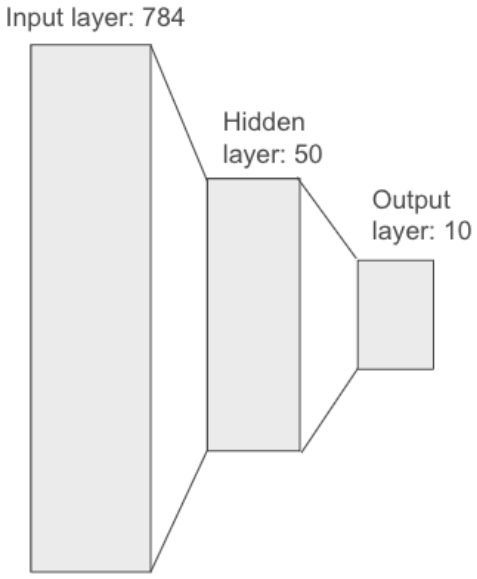
Team Information

NeuralNinjas		
Student 1: Srihita Ramini	Student 1: 919221527	Student 1: sramini@ucdavis.edu
Student 2: Trishna Sharma	Student 2: 918135782	Student 2: tksharma@ucdavis.edu
Student 3: Sai Sindura Vuppu	Student 3: 918091017	Student 3: svuppu@ucdavis.edu
Student 4: Olivia Shen	Student 4: 919293157	Student 4: oshen@ucdavis.edu

Section 1: Task Description

The task is to train an MLP model for data classification. We modified “Dataset_Loader” to load the new dataset, and wrote a new Setting class to pass the loaded train and test features and labels to the method. We experimented with the architecture of “Method_MLP,” such as the input and output sizes, hidden layer size, and hyperparameters including epoch size and learning rates, in order to get the best accuracy. We also experimented with various activation functions such as PRelu, Sigmoid, Tanh, and Softmax, to see which one is the most efficient. Finally, we included more evaluation metrics, such as Precision, Recall, and F1, to further evaluate our model’s performance.

Section 2: Model Description



Section 3: Experiment Settings

3.1 Dataset Description

We used the 'training.csv' and 'testing.csv' files provided by the professor. In both files, each datapoint has 784 features and a label from 10 different classes {0, 1, …, 9}. The training data has 60,000 points and the testing data has 10,000 points. We did not need to do any training and training splits because separate files were already provided.

3.2 Detailed Experimental Setups

We implemented the class Method_MLP based on a PyTorch neural network. Our model uses built-in Pytorch implementations of the following hyperparameters:

- One hidden layer of 50 neurons and ReLU activation function
- Output layer with Softmax activation function
- Cross Entropy loss function
- Learning rate of 2e-3
- Adam optimizer

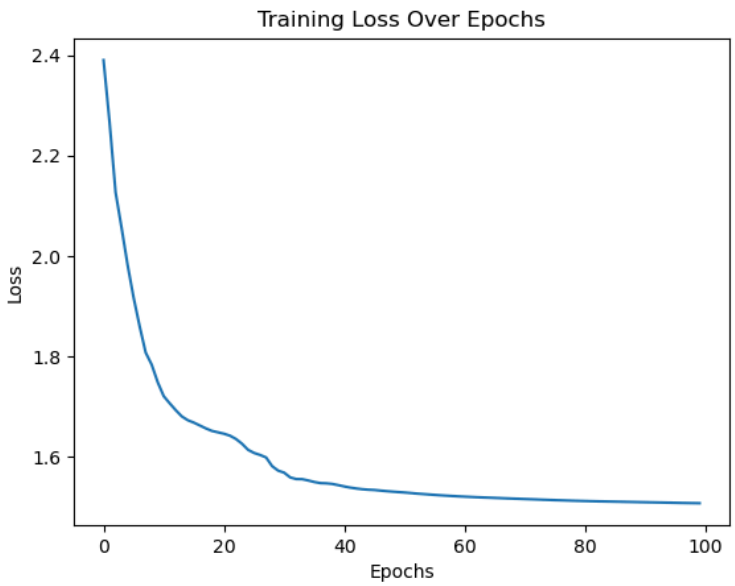
3.3 Evaluation Metrics

Our experiment is a multiclass classification problem. We used Accuracy, Precision, Recall, F1 score as our evaluation metrics. To accommodate multiple classes, we averaged the Precision, Recall, and F1 scores for each class using the average argument in Sci-kit Learn. We used macro averaging because the class distribution across the training and testing data is roughly uniform.

3.4 Source Code

https://github.com/srihita123/189G-

3.5 Training Convergence Plot



3.6 Model Performance

```
************ Performance ************
Accuracy: 0.9429
Precision: 0.9423067644271906
Recall: 0.9423334542941368
F1: 0.9422474542954944
```

3.7 Ablation Studies

We performed ablation studies by modifying different hyperparameters of our model. All experiments use a 3-layer architecture. Metrics are rounded to 4 places.

Changes to model	Accuracy	Precision	Recall	F1 Score
Learning rate: 1e-3	0.9383	0.9383	0.9383	0.9382
Hidden layer activation: Tanh	0.9187	0.9178	0.9176	0.9175
Hidden layer activation: Tanh Learning rate: 1e-3	0.9059	0.9045	0.9046	0.9042
Hidden layer size: 128 Hidden layer activation: Tanh Learning rate: 1e-3	0.9262	0.9253	0.9252	0.9250
Hidden layer size: 128 Hidden layer activation: Tanh Output layer activation: PReLU Learning rate: 1e-3	0.9275	0.9266	0.9265	0.9264
Hidden layer size: 128 Hidden layer activation: Tanh Output layer activation: PReLU Learning rate: 1e-3 Optimizer: SGD	0.4745	0.4300	0.4745	0.4081
Hidden layer size: 128 Hidden layer activation: Tanh Output layer activation: PReLU Learning rate: 1e-3 Optimizer: AdaGrad	0.8991	0.8988	0.8991	0.9000

During the studies, we found a combination of changing hidden layer size to 128, hidden layer activation to Tanh, output layer activation to PReLU, and learning rate to 1e-3 produced an accuracy score of ~0.94 and loss of ~0.2 after training. This is similar accuracy to our original model with a significantly lower loss score. We suspect this is due to using PReLU as the output activation function. In testing, this model produced an accuracy of 0.9275. We decided to keep our original model because it performs slightly better on the test data.