ECS 189G-001

Deep Learning

Spring 2025

Course Project: Stage (2, 3, 4, 5) Report Example

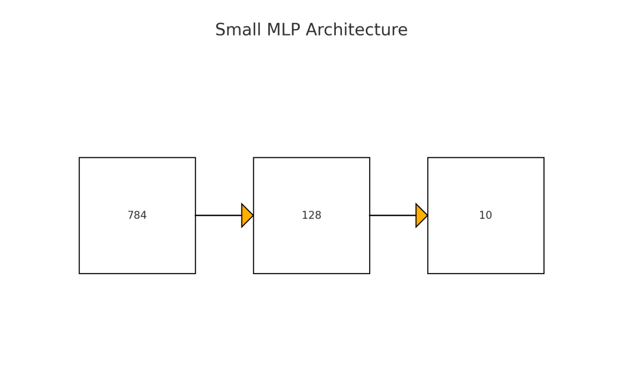
Team Information

Enter Your Team Name Here (delete the extra rows if your team has less than 4 students)		
Student 1: Vikram Penumarti	Student 1: 920928592	Student 1: vpenumarti@ucdavis.edu
Student 2: Rajat Gupta	Student 2: rmgupta@ucdavis	Student 2: 922603941
Student 3: Shevangae Singh	Student 3: svisingh@ucdavis.edu	Student 3: 918163547

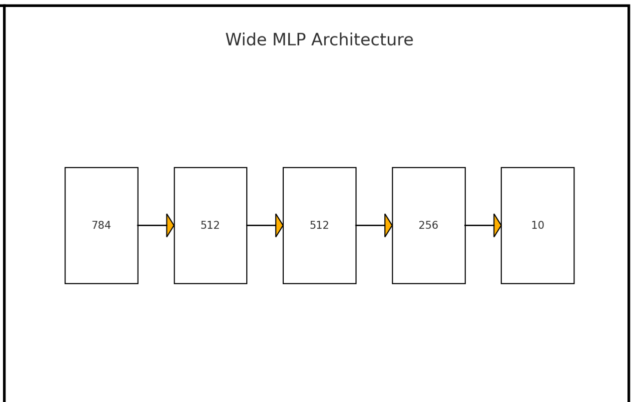
Section 1: Task Description

For this section of the project, we began by downloading the provided training and testing datasets and working from the scaffolded codebase. We modified the dataset loader to accommodate the specific format of the data and updated the settings file, as no additional cross-validation or train-test splitting was necessary given the separate files. In the Method_MLP class, we extended the model architecture to support multiple configurations, including different loss functions (cross-entropy, MSE, MAE), network sizes (small, medium, wide), and optimizers (Adam, SGD). We also added additional evaluation metrics to better assess model performance and generated learning curves over epochs to visualize training progress.

Section 2: Model Description



One input layer, one hidden layer, one output layer.



One input layer, three hidden layers, one output layer.

Section 3: Experiment Settings

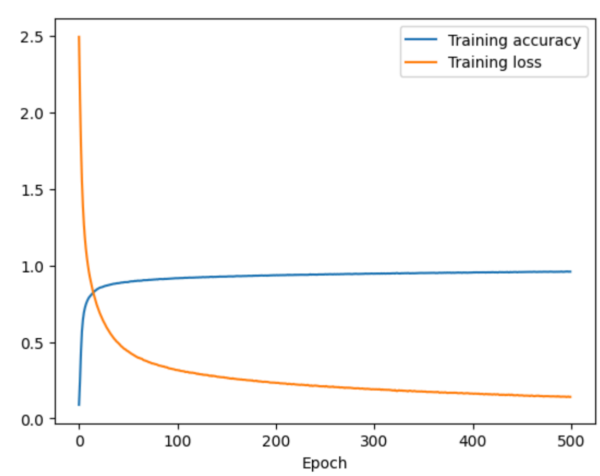
3.1 Dataset Description
The datasets used had 785 features and a label. The datasets were already pre-partitioned into training and testing sets, so when the datasets were loaded, training data and labels were taken from the training file while testing was taken from the testing file.

3.2 Detailed Experimental Setups
The Method_MLP model supports three predefined architectures: small, medium, and wide. The small architecture has two layers with dimensions [784 → 128 → 10], while medium uses four layers [784 → 256 → 128 → 64 → 10], and wide is built as [784 → 512 → 512 → 256 → 10]. All hidden layers use LeakyReLU activation with a slope of 0.1, followed by BatchNorm1d normalization and Dropout with a 0.5 probability to prevent overfitting. The final output layer applies LogSoftmax for classification. The model uses PyTorch’s default initialization.

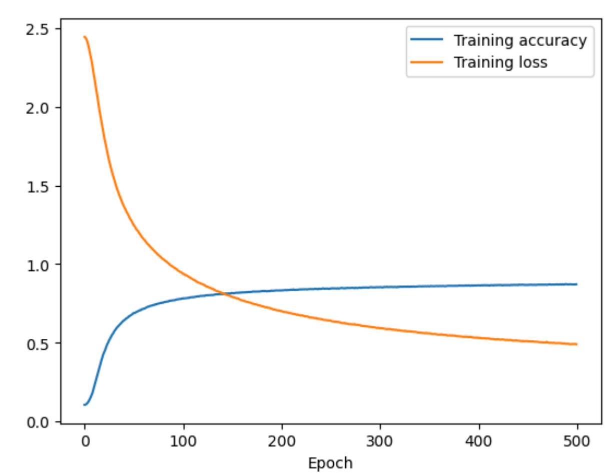
For training, the model runs up to 500 epochs with a default learning rate of 1e-3. It supports two optimizers, Adam and SGD (with a momentum of 0.9) and allows flexibility in the choice of loss function: cross-entropy (cross) for classification tasks, and either mean squared error (mse) or mean absolute error (mae) for regression. The input size is fixed at 784, and the output dimension is 10. There is no dynamic adaptation of architecture or hyperparameters based on dataset properties, settings remain the same across different datasets unless explicitly changed.

3.3 Evaluation Metrics
In the experiment, we used accuracy, precision (macro, micro, weighted), recall (macro, micro, weighted), and f1 (macro, micro, weighted).

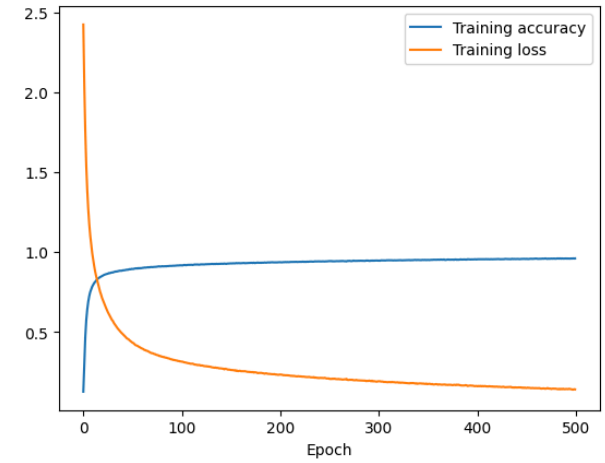
3.4 Source Code
https://drive.google.com/file/d/1EgObh9NUcOAz2LYPA7yAFFFwthbS0gXL/view?usp=sharing

3.5 Training Convergence Plot


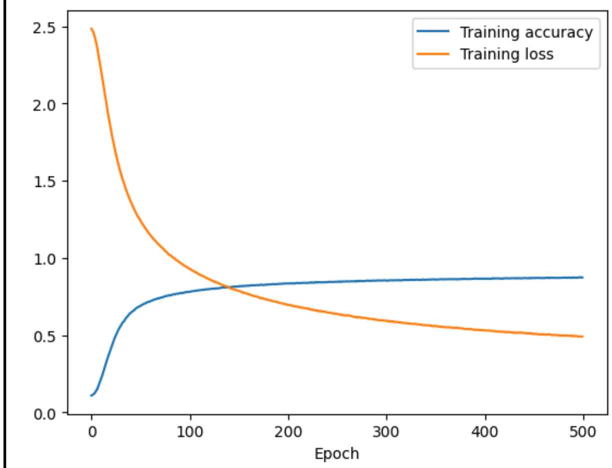
Model size: Small
Loss function: Cross entropy
Optimizer: Adam



Model size: Small
Loss function: Cross entropy
Optimizer: SGD



Model size: Wide
Loss function: Cross entropy
Optimizer: Adam



Model size: Wide
Loss function: Cross entropy
Optimizer: SGD

3.6 Model Performance

Model 1:

Model Size: Small
Loss function: Cross entropy
Optimizer: Adam

Metrics: {"accuracy": 0.9445, "precision_macro": 0.9439415977809625, "precision_micro": 0.9445, "precision_weighted": 0.9443948759643351, "recall_macro": 0.9437791898718799, "recall_micro": 0.9445, "recall_weighted": 0.9445, "f1_macro": 0.943803286784582, "f1_micro": 0.9445, "f1_weighted": 0.9443899639041978}

Model 2:

Model Size: Small
Loss function: Cross entropy
Optimizer: SGD

Metrics: {"accuracy": 0.878, "precision_macro": 0.8766567180750922, "precision_micro": 0.878, "precision_weighted": 0.8774524504449688, "recall_macro": 0.8762950745878658, "recall_micro": 0.878, "recall_weighted": 0.878, "f1_macro": 0.8759718790942044, "f1_micro": 0.878, "f1_weighted": 0.877221295730635}

Model 3:

Model Size: Wide
Loss function: Cross entropy
Optimizer: Adam

Metrics: {"accuracy": 0.942, "precision_macro": 0.9414140932180501, "precision_micro": 0.942, "precision_weighted": 0.9419536440873769, "recall_macro": 0.941259335398863, "recall_micro": 0.942, "recall_weighted": 0.942, "f1_macro": 0.9412539088816636, "f1_micro": 0.942, "f1_weighted": 0.941892244334464}

Model 4:

Model Size: Wide
Loss function: Cross entropy
Optimizer: SGD

Metrics: {"accuracy": 0.8784, "precision_macro": 0.8773774029680356, "precision_micro": 0.8784, "precision_weighted": 0.8779457795938888, "recall_macro": 0.8766613204221538, "recall_micro": 0.8784, "recall_weighted": 0.8784, "f1_macro": 0.8765121766042988, "f1_micro": 0.8784, "f1_weighted": 0.8776539469803037}

3.7 Ablation Studies

Across all configurations, Adam consistently outperforms SGD, and increasing the network width yields only marginal gains when using Adam but does not help with SGD. The small-Adam model converges quickly to a high training accuracy (≈94.5%) with low loss and achieves the best overall metrics (F_1 ≈ 0.944). In contrast, small-SGD plateaus at a substantially lower accuracy (≈87.8%) and exhibits slower loss reduction. The wide-Adam variant reaches nearly the same performance as small-Adam (≈94.2% accuracy, F_1 ≈ 0.942), suggesting that adding width beyond the small configuration provides little extra benefit under Adam. Finally, wide-SGD mirrors the poor convergence of small-SGD (≈87.8% accuracy, F_1 ≈ 0.878), indicating that optimizer choice has far more impact than model width in this setting.