End Semester Project Report
MNIST Digit Classification

Ayush Kanojiya DA24B037

November 15, 2025

(a) A summary of models used & system architecture

1. Models Implemented

Three distinct classification models were built from scratch:

- **`KNN_bruteforce_scratch`:** A K-Nearest Neighbors classifier. This is a non-parametric learning model that classifies a data point based on a majority vote of its `k` nearest neighbors, as determined by Euclidean distance.

- **`fit_logistic_batch` (OvR):** A binary Logistic Regression classifier trained with batch gradient descent and L2 regularization. This model is used in a One-vs-Rest (OvR) strategy, where 10 separate binary classifiers are trained (e.g., "is it a 0?" vs. "not a 0," "is it a 1?" vs. "not a 1," etc.). The final prediction is given by the classifier with the highest confidence score.

- **`SoftmaxRegression_scratch`:** A multi-class Softmax (Multinomial Logistic) Regression classifier. This is a single, generalized linear model that directly handles all 10 classes. It is trained using batch gradient descent and optimizes the cross-entropy loss function.

A Principal Component Analysis model (`PCA_SVD_scratch`) was also implemented to serve as a crucial pre-processing step.

2. System Architecture

The system architecture is a unified pipeline that pre-processes the data once and then feeds the result to all three classifiers for comparison.

The data flow is as follows:

1. **Load Data:** The `MNIST_train.csv` and `MNIST_validation.csv` files are loaded.

2. **Pre-process (Scaling):** The 784-pixel features are normalized using **Standard Scaling** (z-score), where the mean and standard deviation are calculated only from the training set and then applied to both the train and validation sets.

3. **Dimensionality Reduction (PCA):** The `PCA_SVD` model is fit on the scaled training data to find the principal components. Both the train and validation sets are then transformed from 784 dimensions down to 54, (which i found out by repetitive tuning).

4. **Parallel Classification:** This single, pre-processed dataset (10002 samples $\times$ 54 features) is passed to all three from-scratch classifiers (KNN, OvR, and Softmax).

5. **Evaluation:** Each model is trained on the PCA-transformed training data and evaluated on the PCA-transformed validation data. Final F1-scores, accuracy, and runtimes are collected and compared.

(b) Summary of hyper-parameter tuning and results

The script uses a fixed set of pre-tuned hyperparameters for each model to ensure a reproducible comparison. The primary goal is to evaluate the performance of these different architectures given a common, optimized (PCA) dataset.

Model 1: K-Nearest Neighbors (KNN)

- **Hyperparameters:** `k = 6` neighbors.
- **Results (Validation Set):**
  - **Macro F1-Score: 94.55%**
  - Accuracy: 95% (on main.py file)
  - Runtime: 3.30 seconds

Model 2: One-vs-Rest (OvR) Logistic Regression

- **Hyperparameters:** `Epochs = 1000`, `Learning Rate = 0.3`.
- **Results (Validation Set):**
  - **Macro F1-Score: 87.84%**
  - Accuracy: 88.00%
  - Runtime: 1.96 seconds

Model 3: Softmax (Multinomial) Regression

- **Hyperparameters:** `Epochs = 1000`, `Learning Rate = 0.1`.
- **Results (Validation Set):**
  - **Macro F1-Score: 89.43%**
  - Accuracy: 89.56%
  - Runtime: 1.54 seconds

(c) Steps to optimize performance and limit run time

1. **Principal Component Analysis (PCA):** This was the single most important optimization. The prediction time for brute-force KNN is $O(N_{test} \cdot N_{train} \cdot d)$, where d is the number of dimensions. By reducing d from 784 to 54 (a ∼14.5x reduction), the bottleneck of the system (KNN's distance calculation) was drastically sped up. This also significantly reduced the training time for the OvR and Softmax models.
2. **Centralized Pre-processing:** The system's architecture is optimized to perform the most expensive pre-processing steps (loading, scaling, and PCA) only *once*. The resulting transformed data is then shared by all three classifiers, eliminating redundant computation.
3. **NumPy Vectorization:** All from-scratch models were built using vectorized NumPy operations (e.g., `np.dot`, `np.sum(axis=1)`, array arithmetic) instead of slow, iterating Python `for` loops. This is essential for all distance calculations and gradient descent updates.

Table 1: Final Model Comparison (Validation Set Results)

Model	**Macro F1-Score**	**Accuracy**	**Runtime**
KNN (k=6)	**94.55%**	95%	3.30s
Softmax Regression	89.43%	89.56%	1.54s
OvR Logistic Regression	87.84%	88%	1.96s

Evaluation Results

The optimization steps were highly effective, with all models running in seconds. The final comparison, as generated by the script, is summarized below.

(d) Detailed summary of thoughts and observations

- **Justification for Final Model ('main.py'):** The comparative analysis from the `algorithms.ipynb` notebook serves as the primary justification for the model chosen in the final `main.ipynb` file. The **(PCA + KNN) pipeline was selected as the final model** because it achieved a validation accuracy of **95%**, which was significantly higher than both Softmax Regression and OvR . The project's primary goal is to maximize F1-score, and KNN's F1-score of 95.54% was also the highest. While KNN was the slowest model at 3.30 seconds, this runtime is negligible and falls comfortably under the 300-second (5-minute) project limit. Therefore, it was the logical choice, providing the best performance on the primary metrics.
- **Analysis of Model Performance (KNN vs. Linear):** The most significant observation is that the non-parametric KNN model outperformed the two linear models. This suggests the decision boundary for classifying digits, even in the 54-dimensional PCA space, is highly non-linear. KNN, as a non-parametric model, is effective at capturing this complex, local geometric structure, whereas the linear models (OvR, Softmax) are restricted to finding a single separating hyperplane.
- **Critical Role of PCA and Scaling:** The (Standard Scaling + PCA) pipeline was a critical component. Without dimensionality reduction, the brute-force KNN model would be computationally infeasible due to the $O(N \cdot M \cdot d)$ prediction time. PCA successfully de-noised the data and extracted the 54 most important features, enabling all three models to train and predict in seconds.
- **Performance of Linear Models:** The Softmax model (90.73% F1) performed better than the OvR model (87.82% F1). This is expected, as Softmax optimizes all classes simultaneously and its probabilities are inherently linked (sum to 1), leading to a more stable multi-class solution. The OvR model, by training 10 independent classifiers, can result in ambiguous predictions where multiple models have high confidence.
- **Speed vs. Accuracy Trade-off:** The results show a clear trade-off. The most accurate model (KNN) was also the slowest. The fastest model (Softmax) was $\approx$5.8% less accurate (F1) but more than twice as fast. For this specific problem, the 3.30-second runtime of KNN is an excellent and acceptable trade-off for its superior accuracy.