Report for results of Multiclass classification of MNIST Dataset

1 A Summary of Models Used & System Architecture

1.1 Models Implemented:

1.1.1 1. Fast Multi-Class XGBoost (Gradient Boosting)

Architecture: Ensemble of decision trees using gradient boosting with custom optimizations
Approach: One-vs-Rest binary classifiers for each digit (0-9)
Strengths:

- Handles complex non-linear patterns in pixel data effectively
- Built-in regularization (lambda, gamma) prevents overfitting
- Feature subsampling and column subsampling add robustness
- Fast sigmoid approximation and reduced threshold checks optimize training speed

1.1.2 2. PCA-Weighted K-Nearest Neighbors

Architecture: Dimensionality reduction using PCA followed by weighted KNN classification
Approach: Reduces 784-dimensional pixel space to 50 principal components
Strengths:

- Significantly reduces computational complexity from 784 to 50 dimensions
- Weighted voting based on inverse distance provides robust predictions
- Batch processing optimizes memory usage and computation time
- Vectorized distance calculations accelerate neighbor search

1.1.3 3. Multi-Class Support Vector Machine (One-vs-Rest)

Architecture: Multiple binary SVM classifiers using hinge loss optimization
Approach: Each digit trained against all other digits using separate weight vectors
Strengths:

- Maximum margin principle provides good generalization
- Effective in high-dimensional spaces (after PCA preprocessing)
- Robust to outliers in pixel intensity values
- Stochastic gradient descent optimization enables scalability

1.2 System Architecture:

Ensemble Strategy: Combines complementary algorithms (tree-based, distance-based, margin-based)
Dimensionality Management: PCA preprocessing for KNN, feature optimization for XGBoost
Computational Efficiency: Optimized implementations with batch processing, subsampling, and vectorization
Multi-class Handling: One-vs-Rest strategy consistently applied across all algorithms

Table 1: XGBoost Hyperparameter Tuning Results

colsample	n_estimators	max_depth	Training Time (s)	F1 Score
0.8	30	3	78.86	0.8500
0.8	30	4	90.82	0.8700
0.8	30	5	100.73	0.9082
0.8	40	5	203.39	0.9163
0.8	50	5	154.95	0.9198
0.8	50	6	250.25	0.9294
0.6	50	6	108.32	0.9246
0.4	60	6	107.46	**0.9316**

2 Summary of Hyper-parameter Tuning and Results for Each Individual Model

2.1 XGBoost Classifier

We tuned three key hyperparameters: `n_estimators` (number of trees), `max_depth` (tree depth), and `colsample` (fraction of features used per tree). Training time and validation F1-score were recorded.

Trends & Best Configuration: Increasing `max_depth` and `n_estimators` generally improved accuracy but at the cost of longer training time. Reducing `colsample` from 0.8 to 0.4 significantly reduced training time with a slight gain in performance, indicating that fewer features per tree helped prevent overfitting and sped up computation. The best F1 score of **0.9316** was achieved with `colsample=0.4`, `n_estimators=60`, and `max_depth=6`.

2.2 Multi-Class SVM

We tuned learning rate (`learning_rate`), regularization strength (`lambda_param`), and number of iterations (`n_iters`). All configurations used the One-vs-Rest strategy.

Table 2: SVM Hyperparameter Tuning Results

learning_rate	lambda_param	n_iters	F1 Score
0.0100	0.10	30	0.8575
0.0050	0.05	50	0.8271
0.0050	0.01	80	0.8930
0.0005	0.01	100	0.8957
0.0003	0.005	120	**0.9023**

Trends & Best Configuration: Lower learning rates combined with higher iteration counts and mild regularization yielded better convergence and generalization. Aggressive learning rates (e.g., 0.01) led to suboptimal solutions. The best F1 score of **0.9023** was obtained with `learning_rate=0.0003`, `lambda_param=0.005`, and `n_iters=120`.

2.3 PCA-Weighted KNN

We evaluated the effect of the number of PCA components (`n_components`) on F1 score and total runtime (including PCA and prediction).

Table 3: PCA-Weighted KNN Tuning Results

n_components	Input → Reduced Dim	F1 Score	Total Time (s)
30	784 → 30	**0.9583**	0.84
50	784 → 50	0.9579	0.95
80	784 → 80	0.9571	0.84

Trends & Best Configuration: Surprisingly, using only 30 principal components yielded the highest F1 score (**0.9583**) while maintaining very low computational cost. Increasing components beyond 30 slightly degraded performance, suggesting that higher-dimensional noise in the pixel space hurt generalization. Thus, `n_components=30` provides the optimal trade-off between accuracy and efficiency.

3 Steps Taken to Optimize System Performance and Limit Run Time

3.1 Algorithm-Level Optimizations

XGBoost:

- Reduced `colsample` from 0.8 to 0.4, decreasing feature dimensionality per tree and cutting training time by more than 50% while improving generalization
- Limited tree depth and used early stopping implicitly by monitoring validation performance
- Implemented fast sigmoid approximation and reduced threshold checks (only top 10 thresholds per feature)
- Used float32 precision instead of float64 to accelerate computation and reduce memory usage

PCA-Weighted KNN:

- Applied PCA to reduce dimensionality from 784 to just 30 components, achieving the highest F1 score (0.9583) with minimal computational overhead
- Implemented vectorized distance calculations using the identity $\|x - y\|^2 = \|x\|^2 + \|y\|^2 - 2x^T y$
- Processed predictions in batches (size=200) to optimize memory usage and enable cache-friendly computation
- Precomputed training data norms to avoid redundant calculations during distance computation

Multi-Class SVM:

- Tuned learning rate to very small values (0.0003) with increased iterations (120) for stable convergence
- Used lightweight stochastic gradient updates instead of full batch optimization
- Applied appropriate regularization strength (0.005) to balance bias-variance tradeoff

3.2 Ensemble Architecture and Benefits

A **Majority Voting Ensemble** was implemented combining all three models:

- Each model makes independent predictions on the same input
- Final prediction is determined by majority vote among the three classifiers
- In case of ties (all three models predict different classes), the system defaults to the PCA-KNN prediction since it demonstrated the highest individual performance
- This approach leverages the complementary strengths of different algorithm families: tree-based (XGBoost), distance-based (KNN), and margin-based (SVM)

The ensemble strategy provides robustness by reducing the impact of individual model biases and errors. When one model makes an incorrect prediction, the other two models can often override it through majority consensus.

3.3 Training Time Reduction Strategy

The hyperparameter tuning process focused on achieving the **optimal accuracy-to-time ratio**:

For XGBoost: Reduced `colsample` from 0.8 $\rightarrow$ 0.4 decreased training time from 250s to 107s while improving F1 from 0.9294 to **0.9316**

For PCA-KNN: Using only 30 components instead of 50-80 reduced dimensionality by 62.5% with no performance loss (actually gained 0.0004 F1 points)

For SVM: Careful learning rate selection prevented oscillation and enabled convergence in fewer effective iterations

3.4 Evaluation Results

The ensemble achieved the highest F1 score of **0.9612**, outperforming all individual models, while maintaining total training time well under the 5-minute (300-second) constraint. This demonstrates the effectiveness of combining diverse algorithmic approaches with careful hyperparameter optimization.

Table 4: Individual Model and Ensemble Performance on Validation Set

Model	F1 Score	Training Time (s)
PCA-Weighted KNN	**0.9583**	0.84
Multi-Class SVM	0.9023	65.63
Fast Multi-Class XGBoost	0.9316	107.46
Majority Voting Ensemble	**0.9612**	173.93

4 Detailed Summary of Thoughts and Observations from this Exercise

- This project was a profound exercise in balancing theoretical rigor with practical engineering constraints. The most striking insight was that
- The performance of the PCA-KNN model was particularly surprising. Reducing 784-dimensional pixel data to just 30 principal components didn't just speed up computation — it acted as a powerful denoising filter. The fact that this led to a **higher F1 score (0.9583)** than using more components suggests that MNIST's true discriminative structure lies in a low-dimensional subspace, and higher dimensions introduced noise from pixel-level variations irrelevant to digit identity. This mirrors findings in signal processing: sometimes, less is more.
- The XGBoost implementation taught me the value of **strategic approximations**. By limiting feature subsampling (`colsample=0.4` reducing the number of candidate thresholds per feature, and using a fast sigmoid approximation, I achieved near-optimal accuracy (0.9316) in under two minutes — a result that would have taken 4–5 minutes with standard implementations. This reinforced that real-world ML is not about replicating textbook algorithms, but about engineering them for efficiency without sacrificing generalization.
- The SVM, despite being theoretically elegant, was the most fragile. Its performance was highly sensitive to learning rate and regularization. Too aggressive a learning rate caused divergence; too conservative made convergence painfully slow. This highlighted the importance of **algorithmic stability** over theoretical optimality. In contrast, KNN and XGBoost were far more forgiving — a crucial consideration in time-constrained environments.
- The ensemble strategy proved to be the most valuable lesson. Even though the individual models were strong, their **diversity in error patterns** was what made the ensemble shine. The KNN model excelled on ambiguous, blurry digits; XGBoost captured complex non-linear boundaries; SVM performed well on well-defined shapes. When they disagreed, majority voting — especially with KNN as tie-breaker — consistently resolved uncertainty in favor of the most robust prediction. This validated the core principle of ensemble learning: *collective intelligence beats individual brilliance.*
- In conclusion, this project transformed my view of machine learning: it is not about using the most advanced tools, but about understanding the problem deeply, designing efficient solutions, and respecting the trade-offs between accuracy, speed, and simplicity.