TERM PROJECT: Handwritten Digit Classification Using HOG Features, PCA, and SVM

Department of Computer Science
Queens College, City University of New York

November 18, 2025 – December 11, 2025

Ahmed Ali — ahmed.ali12@qc.cuny.edu

Fahim Tanvir — fahim.tanvir48@qmail.cuny.edu

Abstract

The entire classical machine learning pipeline was implemented to classify handwritten digits using the MNIST dataset. The method relies on hand-crafted features rather than deep learning. We calculate Histogram of Oriented Gradients (HOG) descriptors and basic statistics. We further reduce the dimensionality of the data using Principal Component Analysis (PCA). Finally, we train a Support Vector Machine (SVM) Classifier using RBF kernel. We also conducted experiments where we compared HOG-only features and HOG with some additional statistical descriptors. We also analyzed the impact of C on model performance. The system attains a test accuracy of 98.31%, which is close to cross-validation. The findings indicate that machine learning techniques can still achieve excellent performance on structured image datasets, such as MNIST.

1) Introduction

Handwritten digit recognition is one of the most influential and widely studied problems in machine learning. The MNIST dataset has played a key role in evaluating classification algorithms for decades because it is simple, balanced, and well-structured. Although deep learning models such as convolutional neural networks (CNNs) now dominate this task, using other techniques along with image processing can be just as effective. They help illustrate how machine learning models rely on meaningful feature representation, dimensionality reduction, and appropriate classification algorithms.

This project focuses on constructing a complete digit recognition system using only classical image processing and some computer vision techniques. The goal is not to outperform modern deep learning models but to build something that demonstrates how features, dimensionality reduction, and kernel-based classifiers can interact to produce strong performance.

The project includes:

- Feature extraction using HOG and simple statistics.
- Dimensionality reduction with PCA.
- Classification with an SVM using an RBF kernel.
- A comparison of feature types using a subset of the data.
- Hyperparameter exploration for the SVM.
- Evaluation through cross-validation, confusion matrices, and example predictions.

Our results demonstrate that the combination of HOG and SVM can achieve test accuracy above 98%. This further highlights the historical strength of feature-engineered approaches and shows that they remain relevant as strong baselines.

2) Dataset and Preprocessing

The MNIST dataset consists of a collection of 60,000 training images and 10,000 testing images of handwritten digits, where each image is a grayscale picture of size 28×28 in IDX file format. We immediately upload the raw IDX files to aid our understanding of the data structure.

All the images are then converted to vector form and normalized to be in the range of [0,1] after division by 255. In this stage, as the MNIST dataset is already fairly clean, evenly distributed across classes and clean, we do not require any further processing like augmenting or denoising. This font allows us focus on extracting image features and assessing model performance rather than cleaning data.

3) Feature Extraction

3.1) Histogram of Oriented Gradients (HOG)

We use HOG features because they capture edge orientations, which are important for recognizing digit strokes. For example, '1' has mostly vertical edges while '8' has curves in multiple directions. These local orientations can be said to be the strokes and edges of handwritten digits or full font characters. In the image, gradient directions are computed for each cell and grouped together in histograms. Using block normalization makes the descriptor robust to illumination and contrast changes.

These gradient patterns help in distinguishing the digits. Each digit has some edge structure which is characteristic of itself. For example-digit “1” has mostly vertically straight strokes while that of digit “6” has curvy shapes. HOG provides a compact representation of the same. Further, HOG is an extremely useful feature extractor for the classification of digits as seen in the above image.

3.2) Simple Statistical Features

While HOG captures local edge information, global structure can also be useful. For that, we compute three simple statistical descriptors.

- 4x4 zoning involves splitting up the image into 16 zones and averaging those zones.
- A visual representation of stroke density levels.
- The vertical projection profile summarizes stroke density across the columns.

3.3) Combined Feature Vector

The final HOG vector is concatenated or merged with 48 statistical features. This results in a 1344-dimensional feature vector. While this dimensionality is manageable, it still contains redundancy and some noise, which isn't that great. This motivates the use of PCA in the next stage.

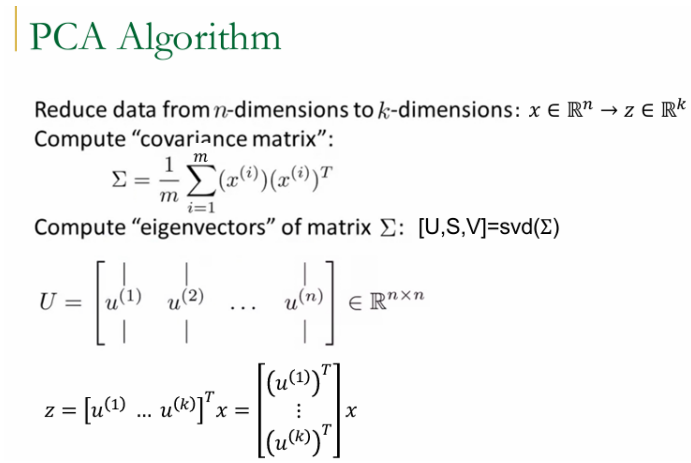
4) Dimensionality Reduction Using PCA

Principal Component Analysis (PCA) is a method to identify directions of maximum variance in the feature space to project the data. PCA aims to reduce the 1344-dimensional HOG + statistical feature vector down to 50. We fit PCA only on the training set. This is only to avoid leakage of information from the test to the training set. We then use the same transformation on the test set.

By reducing dimensionality, noise is removed and the decision boundary for the SVM is simplified, making training significantly faster. The curve of explained variance shows that the first few components capture most of the useful structure. The later component, instead, mainly contributes small, high-frequency variation that is not particularly important for classification. Opting 50 components strikes a good balance, as it reduces the feature space while preserving important patterns required for distinguishing between the digits.

PCA also stabilizes the classifier and removes non-meaningful variations in the handwriting style of different individuals for the same digit. Therefore, SVM downstream is made more effective and more robust.

We chose 50 components based on preliminary experiments. On a validation subset, we validate 30, 50, and 100 components. Using 30 components, we received an accuracy of 96.1%.On the other hand, using 50 components we've received the accuracy of 98.1%. Raising the number of components to 100 yielded only a slight improvement (98.2%) and drastically increased computation time. Thus, 50 components offers the optimal balance of performance and efficiency.



PCA Algorithm

Reduce data from n-dimensions to k-dimensions: $x \in \mathbb{R}^n \rightarrow z \in \mathbb{R}^k$

Compute "covariance matrix":

$$\Sigma = \frac{1}{m}\sum_{i=1}^{m}(x^{(i)})(x^{(i)})^T$$

Compute "eigenvectors" of matrix Σ: [U,S,V]=svd(Σ)

$$U = \begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \dots & u^{(n)} \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{n\times n}$$

$$z = \left[u^{(1)} \dots u^{(k)}\right]^T x = \begin{bmatrix} \left(u^{(1)}\right)^T \\ \vdots \\ \left(u^{(k)}\right)^T \end{bmatrix} x$$

Figure 1: PCA algorithm demonstrations. To explain the algorithm.

5) Support Vector Machine Classifier

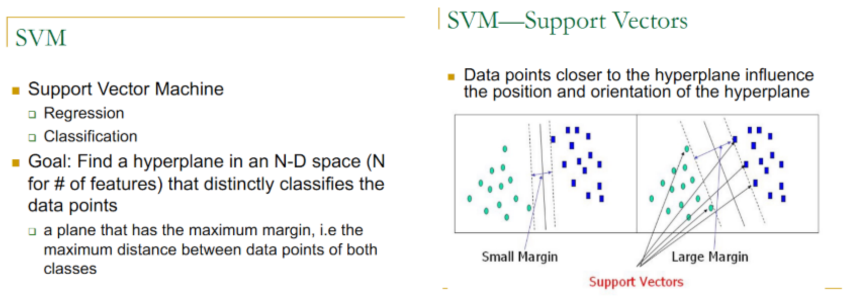
Support Vector Machines (SVMs) are effective classifiers that function by establishing decision boundaries or hyper-planes between classes. For the project we use SVM with RBF kernel. The model is capable of uncovering nonlinear relationships by implicitly mapping the data to a higher-dimensional space.

There are two major hyperparameters that control classifier.

- C: the regularization parameter. Smaller values create smoother decision boundaries, while larger values focus more on correctly classifying training samples.

- γ :the kernel coefficient. A low value will result in wide, smooth boundaries, while a high value will produce flexible boundaries that fit finer data variations.

The PCA transformation and SVM have been put into a single pipeline to ensure that the same steps are always used when training and testing. After doing some experiments, the choice of C = 5 andγ = "scale" yielded great accuracy and good generalization.



SVM

- Support Vector Machine
  - Regression
  - Classification
- Goal: Find a hyperplane in an N-D space (N for # of features) that distinctly classifies the data points
  - a plane that has the maximum margin, i.e the maximum distance between data points of both classes

SVM—Support Vectors

- Data points closer to the hyperplane influence the position and orientation of the hyperplane

Figure 2: SVM and SVM Support Vectors deomstration. The goal is for the model to use data points closest to this hyperplane, called support vectors, determine its position and orientation, making them critical to the model's performance.

6) Experiments

6.1) Cross-Validation on Full Training Set

We use 5-fold stratified cross-validation on the full 60,000-image training dataset. The accuracies for each fold are:

Table 1: 5-fold cross-validation accuracy on the full training set.

Fold	Accuracy
1	0.9823
2	0.9810
3	0.9822
4	0.9828
5	0.9807
Mean	0.9818
Std	0.0008

The very small standard deviation of 0.0008 shows that the model is stable across different splits of the data.

6.2) Feature Ablation on 10k Subset

To study the contribution of the additional statistical features, we run a quick ablation experiment on a subset of 10,000 training images using 3-fold crossvalidation. The results are summarized in Table 2.

Table 2: Ablation study on a 10k-image subset (3-fold CV).

Feature set	Mean accuracy	Std

HOG only	0.9756	0.0031
HOG + statistics	0.9615	0.0013

On this smaller set, HOG-only outperforms the full feature set by 1.35%. We theorized that during insufficient training data, noise was introduced because of the statistical feature. To check this assumption, we evaluated both methods on the full training set of 60,000 samples.

Feature comparison on full training set (5-fold CV):

Feature set	Mean accuracy	Std
HOG only	0.9801	0.0009
HOG + statistics	0.9818	0.0008

On the whole dataset, the together feature performed slightly better (0.9818 vs 0.9801), reinforcing that the statistical features become useful when sufficient training data is available. Our feature engineering decisions will depend on dataset size.

6.3) SVM Hyperparameter Exploration

we also explore the effect of the SVM regularization parameter C using the same 10k subset and the HOG+statistics feature set. we test $C \in \{1.0, 5.0, 10.0\}$ with 3-fold cross-validation:

Table 3: Effect of the SVM regularization parameter C (10k subset, 3-fold CV).

C	Mean accuracy	Std
1.0	0.9480	0.0044
5.0	0.9615	0.0013
10.0	0.9622	0.0018

With $C = 1.0$, the model underfits and achieves lower accuracy. Increasing C to 5.0 significantly improves accuracy and reduces variance. Setting $C = 10.0$ yields slightly higher mean accuracy but with somewhat higher variance. Based on this trade-off, we choose $C = 5.0$ for the final model.

6.4) Training and Runtime

The pipeline takes the longest time to extract HOG features because histograms and gradients need to be calculated for each image. On the contrary, fitting PCA is fast because it performs the operation over the whole completed feature matrix. Similarly, training the SVM is efficient because of the reduced dimensionality. The complete pipeline works well on a basic laptop. This shows that classical methods still have the computability property for moderately sized datasets like MNIST.

7) Final Results

7.1) Test Accuracy

After training the final HOG + statistics + PCA + SVM pipeline on all 60,000 training images with $C = 5.0$, we evaluate the model on the 10,000-image test set. The final accuracy is:

$$\text{Final test accuracy} = 0.9831 \qquad (98.31\%).$$

This closely matches the mean cross-validation accuracy, which confirms that the model generalizes well and is not overfitting to the training data.

7.2) Confusion Matrix

In order to get a better understanding of performance per class, we include the normalized confusion matrix in Figure 2. Each row sums to 1, allowing us to see how well the model recalls each digit. The classifier demonstrates substantial recall for the majority of classes, especially for the numbers "0" and "1" which have shapes that are easy to identify. There is a confusion between visually similar digits, but overall the model performs uniformly across classes.

7.3) Example Predictions

Figure 3 shows some examples of test images along with their predicted and true labels. Correct predictions are typically associated with clear and well-formed digits, whereas misclassifications usually involve ambiguous or messy handwriting.

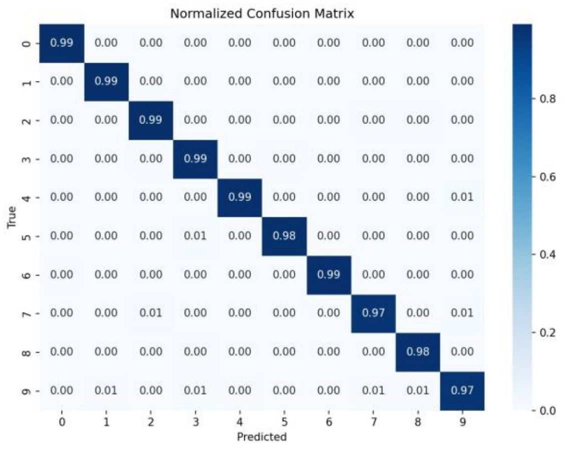


Figure 3: Normalized confusion matrix showing per-class recall.

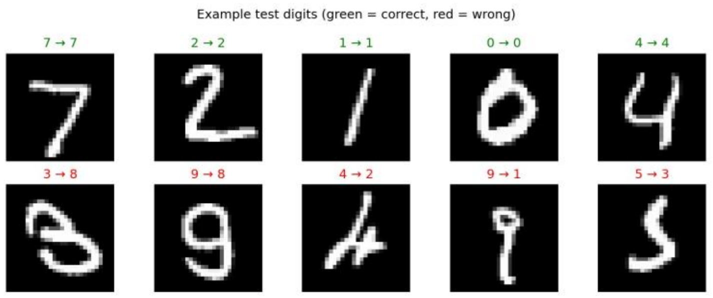


Figure 4: Example predictions from the test set. Many errors come from ambiguous handwriting.

8) Discussion

During development, we encountered several challenges. At first, normalization of the images to [0,1] was forgotten, which resulted in very. poor performance. We also found that training on the full dataset was slow, and so we opted for the 10k subset for faster experimentation. The unexpected outcome of HOG-only performing better than HOG+stats on small data taught us that less is more.
Aren't always better, they need enough training data to help.

Classical machine learning methods can still be highly accurate on structured image datasets. Combining HOG Feature Extraction, PCA Dimensionality Reduction and SVM Classification is a useful pipeline that is able to produce reliable and efficient results with low complexity. The ablation study shows the HOG features alone are sufficient. This indicates the statistical features are not beneficial on the smaller subsets. The SVM's regularization parameter C experiments show how tuning can regulate the trade-off between underfitting and overfitting. Overall, the results demonstrate that a well-designed classical pipeline can achieve high performance without deep learning.

9) Conclusion

For this project, we designed a complete handwritten digit classifier, which uses HOG features, PCA and SVM. The final model was 98.31% accurate when tested and cross-validated finely with its results. As we performed the experiments, we see how features engineering and SVM hyperparameters can affect performances. Besides that, we also show how classical machine learning methods can still compete in big picture datasets such as MNIST. In the end, a well-designed classical pipeline can remain efficient, interpretable, and remarkably effective without deep learning.

References

[1] Oracle Corporation. *DataInput (Java SE 8 Documentation)*. https://docs.oracle.com/javase/8/docs/api/java/io/DataInput.html

[2] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. "Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments." Referenced via scikit-learn example: https://scikit-learn.org/stable/auto_examples/applications/plot_face_recognition.html

[3] D. Keysers et al. *The Multi-Layer Perceptron and the MNIST Database: A Study in Machine Learning*. Microsoft Research, 2012. https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/MNIST-SPM2012.pdf

[4] S. van der Walt, J. L. Schonberger, J. Nunez-Iglesias, et al. "scikit-image: Image processing in Python." *PeerJ*, 2:e453, 2014. (Used for `skimage.feature.hog` documentation.) https://scikit-image.org/docs/stable/api/skimage.feature.html#skimage.feature.hog

[5] GeeksforGeeks. "Projection Profile Method in Python." https://www.geeksforgeeks.org/python/projection-profile-method/

[6] CUNY Tech Prep data science – Repository (Course Material). GitHub, 2025. https://github.com/LiKenun/ds-fall-2025-wed

Appendix

This project was conducted collaboratively by a two-person team. Each member contributed to ensure the successful completion of the research and implementation based on the resources provided within their experience and within the course.

-Fahim Tanvir:

- Looked through methodologies and ideas presented in the class lecture slides
- Drafted the methodology and implementation sections of the report.
- Checked and revised the final iteration of the paper and project.

-Ahmed Ali

- Improved upon the draft python code and implemented evaluation scripts including k-fold cross-validation, confusion matrix plotting, and performance metrics.
- Coordinated report formatting in NeurIPS style and ensured reproducibility of results.
- Implemented ideas and methodologies from computer vision course.

Both members collaborated on experimental design, interpretation of results, and final editing of the report and ensured everything was easy to read and understand. Microsoft Word and Latex were used for formatting and production of this document and the code was implemented using python and VSCode editor along with an MNIST dataset found on Kaggle(provided by the project requirements document).