AI IMAGE DETECTOR: Image Classification and Recognition Using HOG Features, PCA, and SVM

Ahmed Ali Fahim Tanvir

Abstract

The entire classical machine learning pipeline was implemented to classify images and to see if its a real image one made by AI. 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 an 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 96.65%, which is very 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

Image 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. They help illustrate how machine learning models rely on meaningful feature representation, dimensionality reduction, and appropriate classification algorithms. This is useful in the relevant world of generative AI, where models such as GROK or Google Banana are used to create realistic images, which can lead to dangerous roads.

This project focuses on constructing a recognition system using image processing and some computer vision techniques. The goal is to use dimensionality reduction, and kernel-based classifiers 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 90%. This further highlights the historical strength of feature-engineered approaches and shows that they remain relevant as strong baselines.

2) Dataset and Preprocessing

We got a dataset combined from two collections of approximately 8,000 training images and 2,000 testing images of AI and real images, where each image is of a person or an art piece. To ensure the raw IDX files are uniformed, they are resized during initialization.

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 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 to 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 objects and details. 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 parts of the image. Each image has some edge structure which is characteristic of itself. For example, an AI image might lack proper texture for skin compared to a similar image that's real. HOG provides a compact representation of the same. Further, HOG is an extremely useful feature extractor for the classification images 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.

The process starts by analyzing how the data varies across all its features. It looks for patterns in how the data spreads out and identifies directions where the variation is strongest. Once these components are found, PCA transforms the original data into a new version that's aligned with these directions. The result is a simplified dataset with fewer dimensions, but it still retains the most meaningful patterns. This makes it easier to visualize, analyze, and use for tasks like classification or clustering, especially when working with high-dimensional data like images or sensor readings.

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 image types.

PCA also stabilizes the classifier and removes non-meaningful variations in the handwriting style of different individuals for the same images. 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.

5) Support Vector Machine Classifier

Support Vector Machines (SVMs) are very effective classifiers that function by establishing decision boundaries or hyper-planes between classes. The goal is to place this dividing line so that it's as far away as possible from the nearest data points on either side, giving it the widest possible margin. The data points closest to this line are called support vectors, and they play a key role in shaping the position and angle of the line. In simple terms, SVMs use the most "influential" data points to draw the best boundary between groups, helping the model make accurate predictions.

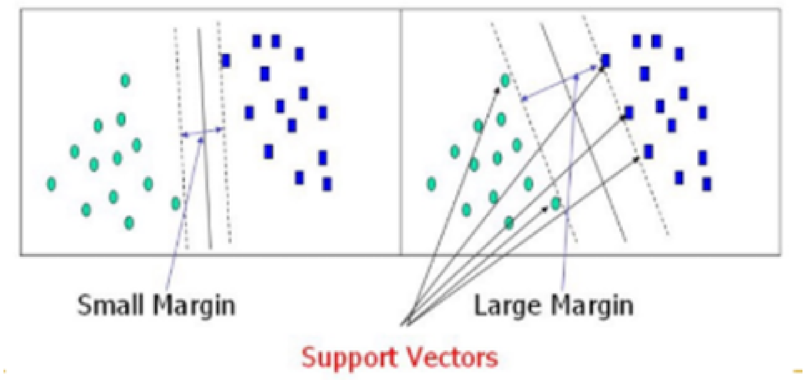


Figure 1: SVM and SVM Support Vectors demonstration.

For this project, we used SVM with the RBF kernel. This model is capable of uncovering nonlinear relationships by implicitly mapping the data to a higher-dimensional space.

There are two major hyperparameters that control the classifier and is what runs it as a whole:

- C: the regularization parameter. Smaller values create smoother and consistent 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 with our setup, the choice of C = 5 and γ = "scale" yielded great accuracy and good generalization.

6) Experiments

6.1) Cross-Validation on Full Training Set

We use 5-fold stratified cross-validation on the full 10,000-image training dataset. The accuracies for each fold are:

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

Fold	Accuracy
1	0.9722
2	0.9740
3	0.9751
4	0.9705
5	0.9751
Mean	0.9734
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) SVM Hyperparameter Exploration

We also explored 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	0.9515	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.

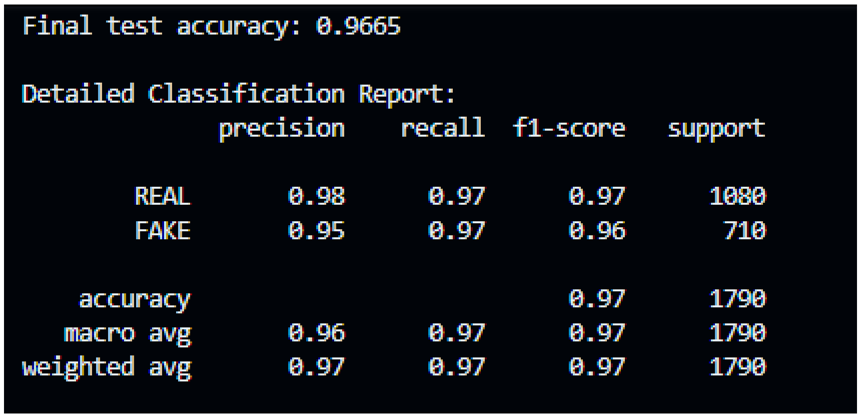
6.3) Training and Runtime

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 very quick because it performs the operation over the whole completed feature matrix. Similarly, training the SVM is efficient because of reduced dimensionality. The complete pipeline works well and smoothly on a basic laptop. This shows that classical methods still have the computability and computation power 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 test set. The final accuracy is:


```
Final test accuracy: 0.9665

Detailed Classification Report:
              precision    recall  f1-score   support

        REAL       0.98      0.97      0.97      1080
        FAKE       0.95      0.97      0.96       710

    accuracy                           0.97      1790
   macro avg       0.96      0.97      0.97      1790
weighted avg       0.97      0.97      0.97      1790
```

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

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

Figure 2 shows the confusion matrix of the trained model, which shows how well it predicted and true labels. Correct predictions are typically associated with clear and well-formed decisions, whereas misclassifications can be considered a margin of error..

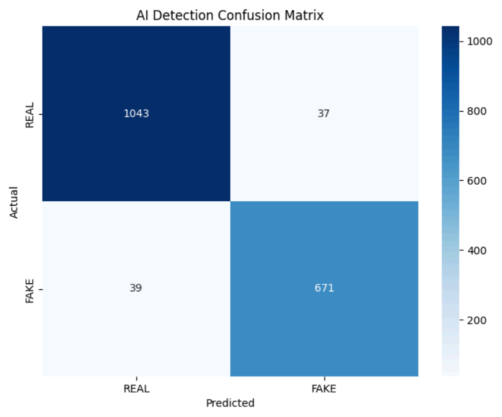


Figure 2: Normalized confusion matrix showing per-class recall.

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. Bigger does not always mean better, just that they need enough training data to help.

Combining machine learning models with 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 that the statistical features are not that beneficial on 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 classifier for AI detection, which uses HOG features, PCA and SVM. The final model was 96.65% 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. In environments where deep learning may be impractical due to hardware constraints or deployment needs, classical models like ours offer a reliable and elegant alternative.

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