Improving SVM Accuracy on MNIST through Hyper Parameter Tuning and Feature Engineering

Aina Shafqat
2021-EE-058
Department of Electrical Engineering
University of Engineering and Technology
2021ee58@student.uet.edu.pk

Ayesha Ahmad
2021-EE-052
Department of Electrical Engineering
University of Engineering and Technology
2021ee52@student.uet.edu.pk

Abstract

This report outlines how the accuracy of SVM was improved on the MNIST dataset through hyper-parameter tuning and feature engineering. The dataset was explored by utilizing different featurizing techniques that included: flattening, inverting colors, sharpening images, extracting image edges, normalizing the images, using HOG, SIFT and PCA, extracting centroids, computing the covariance matrix and its eigenvalues and vectors. Experiments were conducted to assess the impact of various hyper parameters on model accuracy and training time, and to identify configurations that notably improved performance metrics. Our exploration included utilizing different featurization techniques to enhance the model's discriminative power with MNIST data. Hyper parameter tuning employed the optuna library's Bayesian optimizer, while featurization involved transforming raw MNIST data into meaningful representations. The highest test set accuracy achieved was 97.871% for the flat, inverted colors and normalized features.

1 Introduction

The MNIST dataset, consisting of 60,000 labeled handwritten digit images for training and 10,000 images for testing, has long been a standard benchmark in the field of machine learning. Each image is a 28 x 28 grayscale pixel grid, providing 784 features. In this project, we utilize the MNIST dataset to explore fundamental machine learning concepts such as feature engineering, hyper-parameter tuning, and Support Vector Machine (SVM) classifiers. Feature engineering involves extracting and selecting relevant features to enhance model accuracy, while hyper-parameter optimization ensures that our SVM models are well-trained and generalize effectively to unseen data. By leveraging these techniques, we aim to develop accurate and robust models for digit classification tasks.
Through careful analysis and experimentation with feature engineering, hyper-parameter optimization, and SVM classifiers, we seek to gain insights into the intricacies of machine learning algorithms. Our goal is to demonstrate proficiency in model development by maximizing accuracy and explaining the reasons for the inaccuracies.By applying these techniques to the MNIST dataset, we aim to deepen our understanding of machine learning principles and their practical applications.

2 Literature Review

Flat Images: Flat images represent the original grayscale images without any additional processing. Flattening is a technique used to convert multi-dimensional arrays into a 1-D array, which can then be used by classifiers such as Multi Layer Perceptron (MLP) and SVM as features.
Inverted Colors: Inverted color images are obtained by inverting the pixel intensities, changing white to black and vice versa. This transformation can provide a different perspective for feature extraction

Preprint. Under review.

and classification, potentially enhancing the model's ability to capture relevant patterns in the data.
Sharpened Images: Sharpened images undergo filtering to enhance edges and details, improving the contrast between adjacent pixels and aiding in capturing intricate patterns for better classification accuracy.
Image Edges: Extracting edges from images focuses on the boundaries between different regions, which can be crucial for digit recognition.
Normalized Images: Normalizing images adjusts pixel values to a common scale, reducing the impact of variations in lighting and contrast.
HOG (Histogram of Oriented Gradients) 1-1: HOG is a feature descriptor that captures local gradients in image intensity, useful for detecting object shapes and structures.
HOG 2-2: HOG with a larger cell size may capture higher-level features while sacrificing fine details.
Covariance Matrix: The covariance matrix represents the relationships between different pixel intensities, which can provide insights into the image's overall structure.
Eigenvalues and Vectors of Covariance Matrix: Eigenvalues and eigenvectors of the covariance matrix can reveal the principal components of variation in the dataset, aiding in dimensionality reduction.
PCA (Principal Component Analysis): PCA is a dimensionality reduction technique that transforms the original features into a lower-dimensional space while preserving the most important information.
SIFT (Scale-Invariant Feature Transform): SIFT is a powerful feature extraction technique that identifies robust and distinctive local features from images. It is invariant to changes in scale, rotation, and illumination, making it ideal for image matching and object recognition tasks.
Centroids: Centroids are representative points that mark the centers of clusters obtained through clustering algorithms like K-means. They capture the average characteristics of the data points within each cluster, serving as reference points for classification and analysis.
These features play essential roles in improving the accuracy of MNIST digit classification models by capturing relevant information and patterns from the images. Their effectiveness varies depending on the dataset and the classification task, highlighting the importance of exploring multiple feature sets to find the most suitable ones for a particular application.

3 Methodology

3.1 Data Preprocessing

The images in the MNIST dataset were preprocessed through feature engineering. Feature engineering is essential for enhancing the discriminative power of machine learning models. Various featurization techniques were employed to extract meaningful representations from the raw data.
One such technique is the centroid method, which involves identifying the centroid or geometric center of clusters within the data. The number of clusters set were varied from 1 to 20, to generate multiple featurized datasets. Additionally, flattening was utilized to convert the multidimensional image into a one-dimensional array, facilitating easier processing by the SVM model.
Eigenvalues and eigenvectors extracted from covariance matrices of the images were also explored as features, providing insights into the variance and orientation of the data. Furthermore, image thresholding alongside flattening were employed to extract relevant features based on intensity levels. The thresholds were varied from 0 to 1, to generate multiple featurized datasets.
PCA, HOG and SIFT are standard image preprocessing techniques. They were imported from the sklearn library and their design spaces were explored. HOG takes the number of orientations, pixels_per_cell, and cells_per_block as inputs and outputs the processed image. While for PCA, the number of principal components is taken as an input and correspondly the number of output feature vary. SIFT is used to extract features from images, and it produces a variable number of descriptors for each image. Thus it needed to be further processed by bag-of-words approach with KMeans clustering, so that all the descriptors are clustered into the same number of clusters, thus standardizing the number of features.

3.2 Model Training

The featurized datasets were classified using a Support Vector Machine (SVM) model as per the restrictions on the MNIST competition on Kaggle. SVMs are well-suited for tasks where the number of features exceeds the number of samples, making them suitable for the MNIST dataset, which consists of images represented as a matrix of pixels.

3.3 Hyperparameter Tuning

Hyper parameter tuning is a critical aspect of optimizing machine learning models for better performance. To achieve this, the optuna library's default optimizer was employed. It leverages a Bayesian optimization algorithm called the Tree-structured Parzen Estimator (TPE). This algorithm efficiently explores the hyper parameter space to identify configurations that yield the best objective function value. The validation accuracy was set as the objective function value to be maximized. The maximum number of iterations set for the hyper parameter tuning were 30-100 trials. This implies that the SVM was trained 30-100 times with different configurations for one set of features.
The hyperparameters of the SVM tuned included 'C', 'kernel', 'degree', 'gamma', and 'decision_function', each of which plays a crucial role in determining the behavior and flexibility of the SVM model.

3.4 Analysing Accuracy and Error

After the maximum possible validation accuracy of a dataset was computed, the training and test accuracies of the models were studied. When the training and validation errors were similar and high, that featurization was classified as a high bias regime. The solution tried for the high bias regime were adding features and using more complex models. And when training error was low but the validation error was high, that featurization was classified as a high variance regime. The solution tried for the high variance regime included and using simpler models.
The featurized models were also visualized using tSNE plots to check the separability of the data and decide whether a higher accuracy was even achievable. And after training models, the reasons for the validation errors were visualized using Confusion Matrices.

4 Results and Discussion

The best test set accuracy achieved is 97.871%, which was observed for the flattened, color inverted and normalized images. This accuracy ranks 132 on the Public Leader board and 284 on the Private Leader board in Kaggle.

Table 1: Best test accuracies of each feature engineering

			Test Accuracy			
Tuning Time/ hrs	Feature Set	# of Features	Private Score %	Public Score %	Training Accuracy %	Validation Accuracy %
89.5315	Flat	784	97.871	97.266		97.91
85.9184	Inverted colors	784	97.871	97.27	98.91	97.87
198.6969	Sharpened images	784	95.471	94.87	97.96	95.36
149.4099	Image edges	784	87.671	87.666	99.91	87.07
57.6933	Normalized images	784	97.871	97.27	98.93	97.87
163.5914	HOG 1-1	784	95.985	95.9	98.3	95.8
152.3387	HOG 2-2	784	95.985	95.5	97.03	95.73
212.8606	Centroids	10-36	79.342	80.466	82.66	78.85
19.2589	Covariance Matrix	4	43.928	44.666	49.13	48.61
73.3889	Eigenvalues and vectors of Cov Matrix	6	42.228	42.533		42.56
24.3800	PCA-6	168	90.4	90.566	94.28	90.12

4.1 Flattened Filtered Featurization

The MNIST dataset has gray scale images of 28x28 pixels. The most basic featurization was reshaping the array of pixels to make it flat, and passing it as a feature vector to the SVM. Although simple, when the hyper parameters were tuned on it, it gave high validation and test accuracies reaching 97.91% and 97.266% (shown in Table 1).
Then the color of the images were inverted, as shown in Figure 1. The linear kernel of SVM gave validation set accuracy of 94.01%, and its training accuracy was higher at 96.58%. This hinted to a

high bias regime so more complex models like polynomial and radial basis function (rbf) were then used. This improved the validation accuracy. As can be seen in Table 1, the validation accuracy was lower than for the flattened images but the test accuracy was slightly higher at 97.27%.

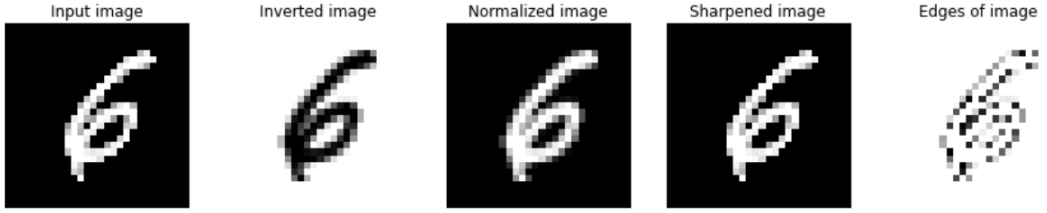


Figure 1: Images featurized by filtering

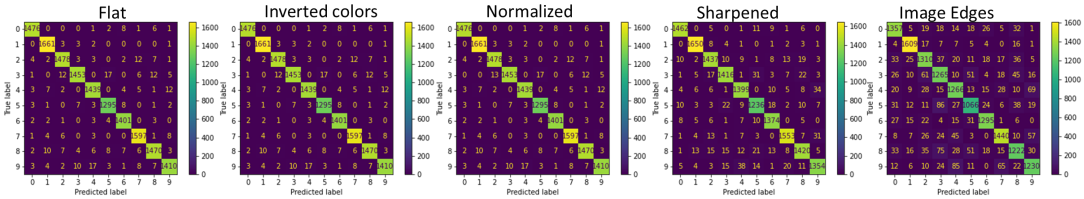


Figure 2: The confusion matrices of the validation sets of the Flattened Filtered Featurization datasets

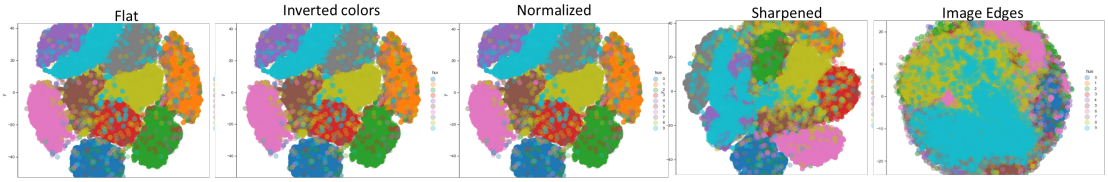


Figure 3: tSNE visualization plots of the Flattened Filtered Featurization datasets

The images were then sharpened by turning the gray pixels close to white into white pixels. The image became clearer as can be seen in Figure 1. But the accuracy of prediction dropped. The reason for this can be observed in Figure 3. The more separable the scatters of each label, the higher the accuracy achievable is. But as can be seen, the scatters on the digits are overlapping, owing to the larger non-diagonal values in its confusion matrix shown in Figure 2.
Now instead of sharpening images, thresholding was used to convert the completely black pixels into white pixels. This way the edges of the digits were obtained in gray scale. But as can be seen in the Figure 4. This was further visualized using the confusion matrix in Figure 2, which showed that 5 and 8 were often misclassified as 3 and, 9 as 4 or 7. The images in the MNIST dataset were now normalized to improve the contrast of the image by expanding the narrow range of input intensities. But the maximum achieved accuracy was similar to the inverted images. It can be observed that the confusion matrix and 2D tSNE visualization of the flattened,inverted and normalized images are very similar. This shows that inverting colors or normalizing the image has little actual effect on the dataset.

4.2 Mathemaical Featurization

The covariance matrix of the image arrays were computed and flattened to use as features vectors to train SVM. But even the maximum validation and training accuracy did not cross 50% and there were only 4 features, which eluded to a high bias regime. As a solution more features were added in the form of eigenvalues and vectors of the covariance matrix. But instead of increasing, the accuracy decreased. Even when the covariance matrix was concatenated with its eigen components the accuracy was still low. These results show that the number of features was too small, and that the covariance matrix and its eigenvalues and eigenvectors are not good representatives of the dataset.

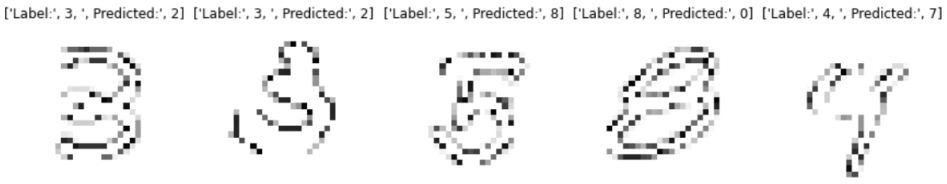


Figure 4: Example of wrongly predicted images in the Image edges featurization

To increase the number of features another mathematical featurization was implemented. The principal component analysis (PCA) of the image arrays was computed and the SVM was trained on it. The Figure 5 shows that the PCA featurized digits are more separable than the ones with covariance or eigenvalues as its features. This is also reflected in its accuracy, as it has a higher validation accuracy.

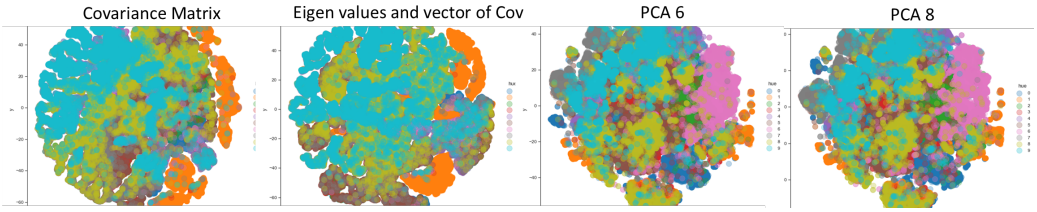
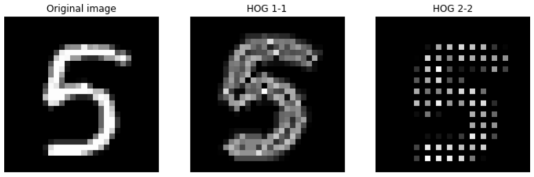


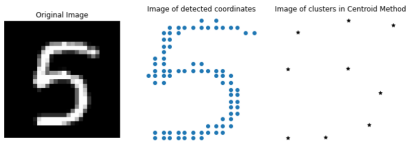
Figure 5: tSNE visualization plots of the Mathematically featurized datasets

4.3 Computer Vision Featurization

HOG is a feature descriptor that is generally used for key point or object detection. Here, HOG 1-1 has orientations=8, pixels_per_cell=(1,1), and cells_per_block=(1, 1), while its only difference with HOG 2-2 is that pixels_per_cell=(2,2). The effect of this featurization can be observed in Figure 6a.As can be seen in the tSNE plot HOG 2-2 has more separable scatter than HOG 1-1, and thus has higher validation accuracy.
The centroid method proved effective in transforming raw data into meaningful representations, thereby contributing to improved classification performance for the training set, with a training set accuracy reaching 99.95%. However, its validation accuracy 78.85% was not as high, likely due to over fitting. And as can be visualized in the Figure 8, the the plot is so scattered that the boundary to separate them is very complex, hence it has lower accuracy than the HOG featurization.



(a) Images featurized by HOG



(b) Images featurized by the centroid method

Figure 6: Images featurized by computer vision techniques

5 Limitations

The extended training times for certain feature sets present significant limitations in terms of computational resources and time constraints. The "Centroids" feature set took the longest time at 212.8606 hours, followed by "Sharpened images" at 198.6969 hours, and "HOG 1-1" at 163.5914 hours. These durations highlight the challenges posed by feature extraction methods like computing centroids and sharpening images, especially for large datasets or limited computational power, limiting their practicality and scalability for real-world applications.

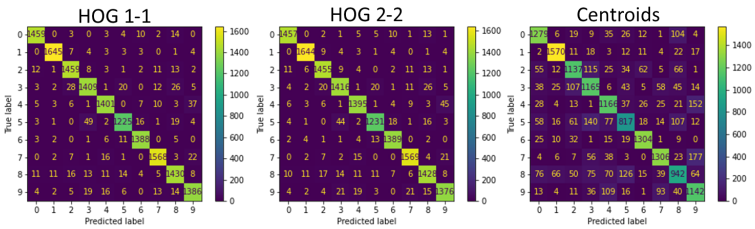


Figure 7: The confusion matrices of the validation sets of the HOG and Centroid datasets

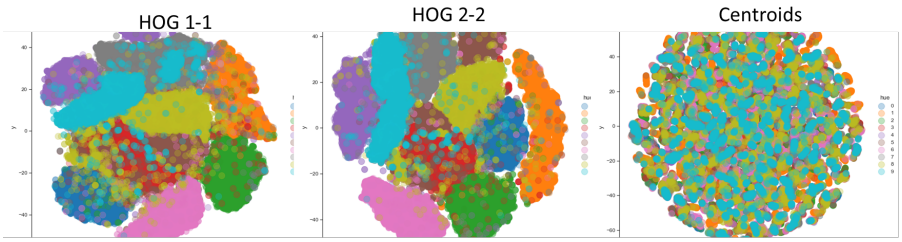


Figure 8: tSNE visualization plots of the HOG and Centroid datasets

Additionally, the SIFT (Scale-Invariant Feature Transform) method, used to extract features from images, produces a variable number of descriptors for each image. To handle this variability, the bag-of-words approach was applied with KMeans clustering. However, for approximately 1500 images, no descriptors were detected, causing issues with array sizes. This inconsistency impacted the feature extraction process and the overall classification performance.

6 Conclusion

In conclusion, this project demonstrates the effectiveness of hyper parameter tuning and feature engineering in enhancing machine learning model performance. Through rigorous experimentation and optimization, significant improvements in accuracy metrics were achieved, paving the way for further exploration and optimization in future research endeavors. This project shows the utility each of each features for training SVM on images of digits and identifies features, like SIFT,that can not be used for the complete MNIST dataset.

References

[1] Singh, A. (2024) SIFT Algorithm: How to use SIFT for Image Matching in Python, *Analytics Vidhya*, www.analyticsvidhya.com/blog/2019/10/detailed-guide-powerful-sift-technique-image-matching-python/.

[2] Mwiti, D. (2023) Image Enhancement in Machine Learning: The Ultimate Guide, *ActiveLoop*, www.activeloop.ai/resources/image-enhancement-in-machine-learning-the-ultimate-guide/.

[3] sagnikmukherjee2 (2021) Impact of Image Flattening, *geeksforgeeks*, https://www.geeksforgeeks.org/impact-of-image-flattenin

[4] Stephan, W. (2021) How exactly does color inversion work? , *Quora*, https://www.quora.com/How-exactly-does-color-inversion-work

[5] Singh, A. (2024) Feature Engineering for Images: A Valuable Introduction to the HOG Feature Descriptor , *Vidhya Analytics*, https://www.analyticsvidhya.com/blog/2019/09/feature-engineering-images-introduction-hog-feature-descriptor/

[6] Jaadi, Z. (2024) A Step-by-Step Explanation of Principal Component Analysis (PCA) , *builtin*, https://builtin.com/data-science/step-step-explanation-principal-component-analysis

[7] abpaudel (2021) Pen Digits Classification , *GitHub*, https://github.com/abpaudel/pen-digits-classification/

[8] Boisberranger, J.D., Bossche, J.V.D Estève, L. Fan, T.J. et al. (2024) Scikit-Learn Stable Documentation , *scikit-learn 1.4.2*, https://scikit-learn.org/stable/