A MULTI-TECHNIQUE APPROACH TO MNIST DIGIT CLASSIFICATION

Tomás Ockier Poblet, Sonia Espinilla Diaz, Queralt Salvadó Hernandez

Fund. of Machine Learning Final Project

School of Engineering, Universitat Autònoma de Barcelona
December 16, 2024

ABSTRACT

Handwritten digit recognition is an important problem in Machine Learning and Computer Vision. In this project, we worked with the simplified MNIST dataset, which contains over 5,000 examples of handwritten digits. Our main goal was to build an accurate classification model. Support Vector Machines (SVMs) are the models that performed best and the main focus of this project. An ablation study was conducted to determine the impact of Feature Scaling and PCA on the model's performance, concluding that these steps were not really necessary. Data augmentation via rotations of the images was performed to find the limits of the SVMs, as well as providing some other insights. Finally, some ensemble models were tried, including Voting and Bagging classifiers. All of the results across the project seemed to indicate that SVMs are the best performing models on this dataset.

Index Terms— MNIST, Supervised Learning, Support Vector Machines, Ensemble Classifier, Data Augmentation

1. INTRODUCTION

The dataset used is Optical Recognition of Handwritten Digits from the UCI Machine Learning Repository [1], consists of 5,620 instances of 8x8 grayscale images representing handwritten digits from 0 to 9. Each image is flattened into a 64-dimensional feature vector, where each feature corresponds to a pixel's grayscale value.

The project explores a multi-technique approach to digit classification using the MNIST dataset, transformed into a more compact and efficient format. The original data, consisting of 32x32 bitmaps, has been reduced into 8x8 grayscale images. The original image where subdivided into 4x4 blocks. Inside of each one, the white pixels where counted, taking this number as the new pixel value in our grayscale images. Therefore, our pixel values range from 0 to 16.

We believe that this reduction on the image size is what allowed us to implement successfully SVM model, otherwise, the images would be too large for this models. Since the approach preserves the original data well enough, we were able to correctly classify the images using the types of models seen on the theoretical lectures.

Our objective in this project is to implement a Support Vector Machine (SVM) on the MNIST dataset. Given that historically were this types of models the ones used for this task, before the Neural Networks existed [2]. We aim to extend this approach by applying the model to a larger and more complex dataset. By doing so, we seek to evaluate the performance and robustness of the SVM under more challenging conditions. This approach allows us to explore the limits of the model's capability, identifying its strengths and potential weaknesses when handling extensive and varied data. The results will help us determine the practical applications and boundaries of SVM for handwritten digit recognition.

2. DATA PREPROCESSING AND VISUALISATION

We started with some basic data exploration in order to asses if the dataset was representative enough to successfully implement the models.

2.1. Overall Data Exploration

The dataset was analyzed using the describe function from a pandas DataFrame to obtain descriptive statistics. It was observed that many features have means close to 0, indicating that some pixels are often blank or have low intensity. Features with higher standard deviations, such as attributes 6 and 62, showed greater variation in pixel intensity, suggesting these correspond to more important regions in the images. There are also a couple features are on every image have a value of zero.

We were able to check that there were no missing values in our dataset, since all features had 5620 entries, meaning the dataset was complete. To us this was an advantage of working with images instead of structural tabulated data. There are no missing values and we do not have to encode categorical data.

2.2. Class Data Exploration

The image count for each class indicated us that the classes were distributed equally, with the dataset having more or less the same number of samples for each class.

Plotting the mean of each attribute in the distances displayed in an 8x8 matrix. This revealed that the majority of meaningful pixel activity is concentrated in the central region of the grid, corresponding to the relevant part of the digits. While the outer edges are mostly inactive, which suggested us that applying some kind of dimensionality reduction, such as PCA, could be useful.

The most important insight is that the data is representative enough -to us, humans-, since we can clearly distinguish each class to the naked eye. The dataset does not seem to have any alarming trait, seems representative and diverse enough to achieve our objectives.

After calculating the mean, we computed the entropy and skewness to gain insights into the distribution of pixel intensities in the MNIST dataset for each class. Entropy

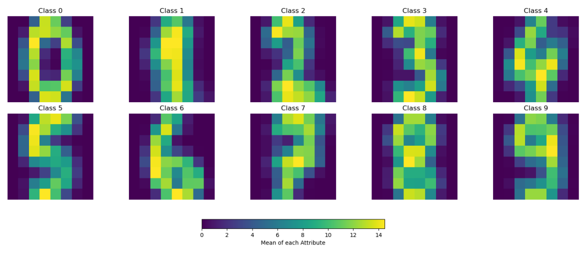

Fig. 1: Mean of each attribute. Each class can be distinguish from the means.

measures randomness or disorder, with higher entropy indicating more complex or diverse patterns. Class 9 has the highest entropy (5.215), reflecting greater variability, while class 6 has the lowest (5.019). Skewness measures asymmetry in the intensity distribution; higher skewness indicates greater imbalance. Class 7 exhibits the highest skewness (0.624), while class 8 has the lowest (0.351).

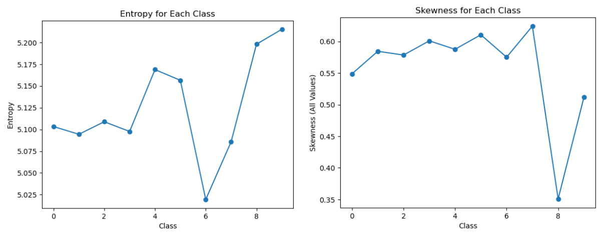

Fig. 2: On the left entropy and on the right skewness for each class.

When considering only non-zero values, both entropy and skewness slightly increase (e.g., class 1 entropy rises from 5.10 to 5.34), indicating reduced uniformity and greater variability in the active regions of the images.

3. SVM IMPLEMENTATION

After analyzing our data, we chose the models we wanted to train. We decided to train several SVM models and used Grid Search to find the best hyperparameters. Additionally, we conducted an ablation study to understand how scaling and PCA affect the model's performance.

3.1. Basic Ablation Study on SVMs

Performing a basic ablation study on this model, we concluded that using either feature scaling or PCA did not increase the models performance at all, as shown on table 1. It even seems that feature scaling worsens the performance. Since there is no meaningful difference between data with the PCA and the bare bones model. We decided to not use PCA or other dimensionality reduction methods from now on.

The initial values of the grid search were chosen based on the best performing basic model that that was also trained using grid search. That way the computation time was not so excessive.

Just to mention the effect of dimensionality reduction on the dataset, the use of PCA on the dataset yielded a reduction from 64 to 20 features at the 90% cumulative variance threshold as can be seen in figure 4.

Table 1: Comparison between the 4 datasets proposed. The key insight is that scaling the features yields slightly worse performance. However we also need to mention that the γ hyperparameter had different values. However, fixing the hyperparameter did not change this dynamic, the feature scaling seems to be worsen the model performance either way. The results were consistent regardless of the train-test split. The reported numbers are using the same training and test set.

Data used	Accuracy	Precision	Recall	F1-Score
Barebones	0.995552	0.995603	0.995552	0.995546
PCA	0.995552	0.995603	0.995552	0.995546
Scaled	0.991103	0.991125	0.991103	0.991093
Scaled & PCA	0.991103	0.991125	0.991103	0.991093

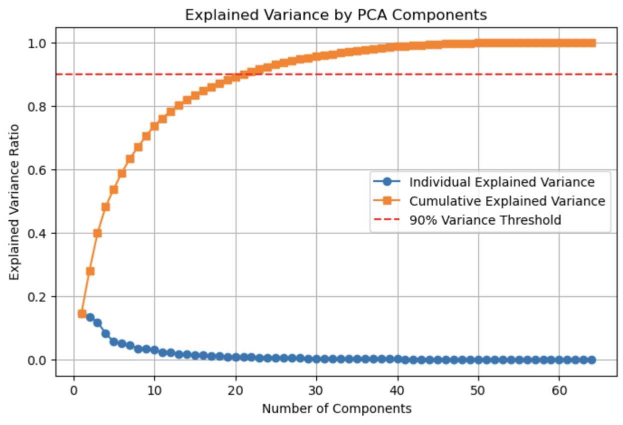

Fig. 3: Variance explained vs. number of components: most variance is captured by fewer than half the components.

3.2. Best Performing SVM

To study and understand the errors of the model and also report more in detail performance metrics we are going to use just an SVM model with the grid search hyperparameters.

3.2.1. Identifying misclassified points

Considering this our flagship model, we can now study errors -missclasifications- that the models does. We realized that some of the misclassified points in the confusion matrix might have been due to mislabeled instances in the dataset. To analyze this, we looked further in detail to identify the misclassified points and determine whether mislabeling was the cause.

The first step was to identify the misclassified points in the confusion matrix by looking at their corresponding row and column. We then counted the total number of errors, which turned out to be 5, and visualized both the predicted and true classes for each misclassified point, along with the corresponding image, as seen on the figure 5. The errors seem to be "reasonable", as there are instances where is difficult to us to clearly distinguish one number on the image.

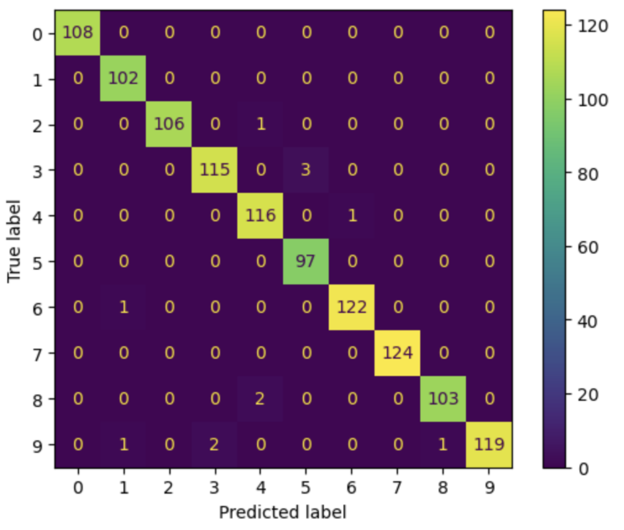

Fig. 4: Confusion matrix obtained from the SVM classifier. Here we can appreciate the misclassified points.

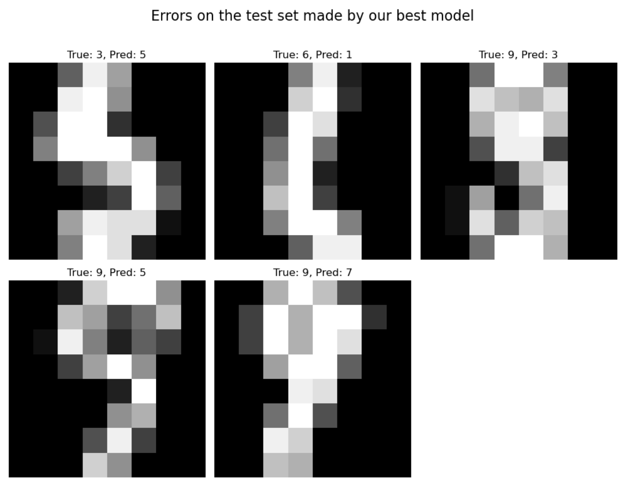

Fig. 5: The 5 errors that we found on our best performing model. The more reasonable error is the second one. Due to the reduced image, there are always instances were the label does not make sense. This can also be observed on the outliers calculations.

3.2.2. *Outliers in the Data*

Assuming that the underlying distribution for each data class is Gaussian, we can try to identify outliers. Given the mean for each class on the dataset, we can consider that the images that are further from this means are outliers.

4. SVM WITH DATA AUGMENTATION

Due to the outstanding performance of the SVMs on the original dataset, we need to focus our attention on extending the original data by performing what is called data augmentation. The key idea is to expand the dataset by transforming some of the images present on it. We can either replace them or add them as new image, thus increasing the dataset size. The purpose of this approach is to create more variety on the data in order for the model to generalize better. The on of the simple -and interesting- ways of doing this would be to rotate the images.

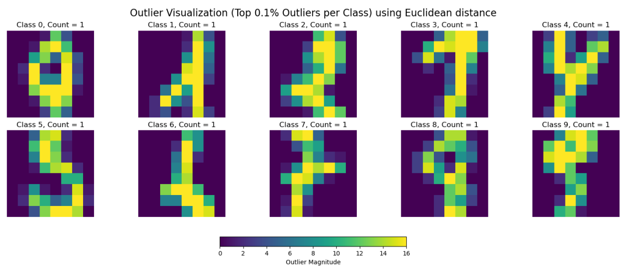

Fig. 6: The image for each class that is further from the mean. The metric used is the ℓ_2 norm. Here we only present one outlier per class. In the code we shown more outliers for each class, however we do not include them here for the sake of brevity. The classes that seem to be more diverse are the 3 and 6. Other ones that can give issues are the 8 and 9. On the other hand 4 for example seems pretty consistent.

Table 2: Test training set rotated according to $\mathcal{U}(-\theta, \theta)$. There is a clear decline on all of the metrics as the angle decreases. This decline does not seem to severe on the precision for all classes. The same dataset as in the previous table was used. For the more severe angle tested, the model still performs best than chance. To see more clearly the evolution, we provided a plot in figure 7.

Angle	Accuracy	Precision	Recall	F1-Score
0	0.995552	0.995603	0.995552	0.995546
5	0.951068	0.953982	0.951068	0.950962
10	0.938612	0.943780	0.938612	0.938635
45	0.602313	0.761367	0.602313	0.621740
90	0.378114	0.730194	0.378114	0.413268
135	0.302491	0.700752	0.302491	0.328347
180	0.290925	0.591213	0.290925	0.296655

5. IMAGE ROTATIONS

We created a function that rotates the image maintaining the 8x8 size. On the code notebook many examples of rotated images can be seen, with a small explanation of the method used.

Since we have to rotate many or all image in our dataset, we need to randomize the rotation angle in some way. We developed two distinct probability distributions from which to sample. We will refer to them as "uniform" and denoted as $\mathcal{U}(-\theta, \theta)$, and "pseudo-bimodal" or "bimodal" denoted as $\pm\mathcal{N}(\theta, \sigma^2)$.

The bimodal distribution is used to evaluate the performance on a specific angle. Whereas the uniform distribution can be used to evaluate the model performance on all angles at the same time. More details on why and how this distribution are created can be found on the code notebook. The σ^2 used on the bimodal is 0.1.

5.1. Rotating the whole dataset

We can directly test how well our best model performs on a variety of rotations. As expected, we see a decline in accuracy as the rotation degree used on $\mathcal{U}(-\theta, \theta)$. The results are both reported on the table 2 and the figure 7. We can rotate the images by 10° and still get around 93% accuracy, which is still excellent. This level of rotation would account for slightly poorly taken pictures, therefore we consider that this model could work well enough for real life applications, provided that the images are segmented and transformed of course.

In the figure we can observe that a model trained on uniform rotations always performs better than on bimodal. To us, the explanation for this is that on the test set there are still many images that are rotated by a degree close to zero.

The most interesting results that we see on this figure is the increase of performance that the model has when only evaluated on the higher degree rotations. We do not have a clear explanation for this, the model just works better when the images are close to horizontal. Moreover, we can see that the model evaluate from 50° to 120° degrees performs on par this a random prediction of the labels, around 10% accuracy -see red line-.

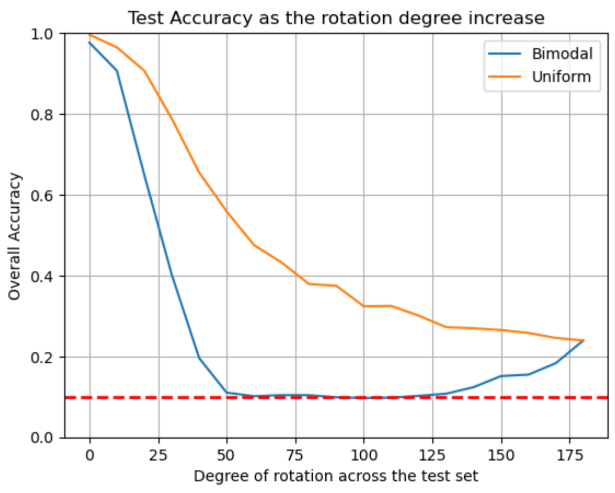

Fig. 7: Accuracy on the test dataset as we keep increasing the rotation angle on the whole dataset, meaning that all images on it have been rotated. Both distribution types are used. The horizontal red line is there to illustrate the performance of a model that assigns labels at random.

To us, the more simple way to get a model that generalizes across rotations is to train a SVM using data rotated by a uniform distribution. We trained 19 different models, each trained on a dataset that had images rotated by 10° each time. As the angle was increased the model's accuracy kept worsening. The final value for a dataset rotated by 180° at maximum was around 92%. The observed decline is shown on figure 8. We can consider that this decline happens due to the increase complexity of the dataset. The size of the feature space is theoretically increasing alongside the degree, as the images should be less similar to each other.

We performed an experiment with the aim of comparing two approaches in creating models that need to achieve excellent accuracy across images rotated by any degree. The idea is to train several models, each one on a specific dataset that has been rotated by an specific angle. For example, only training with image that have been rotated by degrees sampled from $\pm\mathcal{N}(\theta, \sigma^2)$. Achieving near perfect

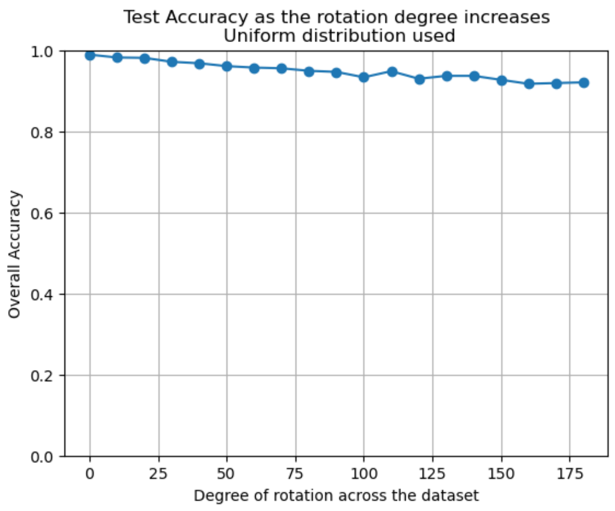

Fig. 8: Performance of an SVM model trained on images that are rotated by a larger decree. Every image on the original dataset is rotated according to an uniform distribution. The whole dataset is rotated of course, both the training and testing set. The same split is used across the experiment.

performance on images rotated either by θ or $-\theta$, with poor results in other types of images. Assuming that we have several of these models, and -most importantly- a way of estimating the rotation of a given image, we can pick the model that fits best the image.

Our findings indicate that this approach could work, since we found that the combined specific models performed better than the more general approach presented earlier, with the model trained on images uniformly rotated of figure 8. On figure 9 we can observe that the accuracy of the models trained on a specific angle perform always better that the uniform model.

We, of course, need to mention that this is practice does not make that much sense. If we know the rotation of an image or a good estimation for it, we can just rotate it back. Moreover, if this estimation fails by a significant margin, then the model picked would very probably fail the prediction. Still, we wanted to report this finding, since it shows that training several models on smaller feature spaces can yield more performance that training one over a wider or larger feature space.

5.2. Expanding the Dataset

The remaining option is to expand the dataset -adding more images-. This of course increases the variation of the future space, as well as the image count on the dataset.

Our results, as seen in figure 10, show that training a model using this approach does not surpass the performance of the previously trained model, whose results can be seen in figure 8.

6. ENSEMBLE LEARNING

Ensemble learning combines the predictions of multiple base models to improve the overall performance compared to individual classifiers. This section presents and eval-

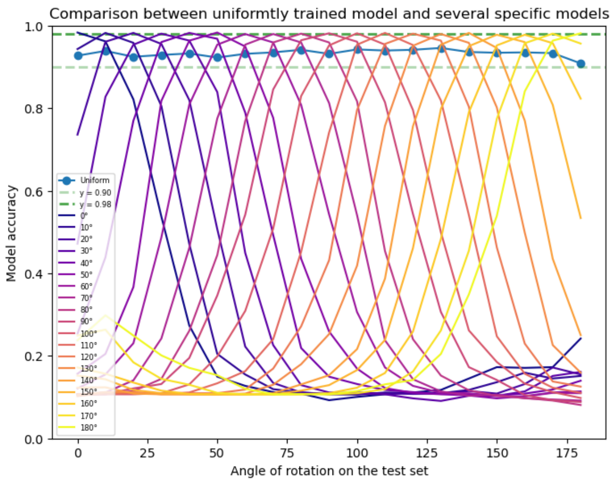

Fig. 9: For each rotation angle on the test dataset. That is, for each image with an angle θ, there is at least one model trained on image rotated closely to θ that performs better than the uniform one.

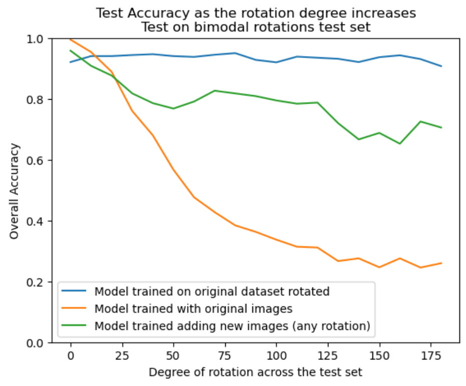

Fig. 10: Different evaluations of 3 models over a bimodal test dataset. In orange there is the model trained on the original dataset. On blue, our previous models trained on the original dataset rotated uniformly. Finally in green, a model trained on expanded data. The images added to the dataset were rotated uniformly as well. In both cases $\mathcal{U}(-180, 180)$, was used.

uates several ensemble techniques applied to our dataset, including Voting Classifiers and Bagging.

6.1. Voting Classifiers

The VotingClassifier is an ensemble method that combines predictions from multiple classifiers. In this project, we used hard voting, which predicts the class that receives the majority vote from the base classifiers. We tried 4 classification models, Logistic Regression, Naive Bayes, Decision Tree, and SVM to leverage their individual strengths. So we compared the accuracy of each classifier. Following the process we excluded the Naive Bayes classifier to assess its impact on the overall ensemble performance. Seeing the results, we focused on combining two classifiers, Logistic Regression and SVM, which demonstrated strong individual performance, simplifying the ensemble.

The results showed that the VotingClassifier generally outperformed most individual classifiers, demonstrating the benefits of combining diverse models. Excluding Naive Bayes from the ensemble slightly improved performance, suggesting that Naive Bayes may not contribute positively in this context. A simplified VotingClassifier with Logistic Regression and SVM achieved strong performance, indicating that fewer high-performing models can provide effective results.

6.2. Bagging Classifiers

Bagging, or Bootstrap Aggregating, is another ensemble technique that enhances model stability and generalization by training multiple models on different subsets of the training data created through sampling with replacement. In this project, a BaggingClassifier with SVM as the base estimator was implemented. Key parameters included using 100 base models, training each on 50% of the training data, and enabling out-of-bag evaluation to estimate performance.

The BaggingClassifier achieved an accuracy of 98%, which was slightly lower than the individual SVM model's accuracy of 99%. This slight drop may be due to the sampling variation introduced during bagging. However, the BaggingClassifier adds robustness to the model and reduces the likelihood of overfitting by aggregating predictions from multiple estimators, making it a valuable technique for improving generalization in practice.

7. CONCLUSIONS

Through our project we showed that Support Vector Machines (SVMs) are highly effective for handwritten digit classification using a simplified MNIST dataset, in which SVMs consistently achieved an excellent accuracy.

SVMs performed exceptionally well with minimal preprocessing, reaching nearly perfect accuracy. The ablation study confirmed that techniques like PCA and feature scaling did not improve results. We also observed that most errors came from ambiguous images. And that, outliers revealed challenges with less consistent classes. Regarding our investigation through data augmentation, rotating images reduced accuracy, but SVMs still performed well with small rotations (up to ±10°). Concerning our last trial at achieving better accuracy results, techniques like Voting Classifiers and Bagging slightly improved model robustness but did not outperform SVMs.

In conclusion, we confirmed that SVMs are a reliable and efficient choice for digit classification tasks, especially for datasets that have a smaller feature space.

8. REFERENCES

[1] E. Alpaydin and C. Kaynak, "Optical Recognition of Handwritten Digits." UCI Machine Learning Repository, 1998. DOI: https://doi.org/10.24432/C50P49.

[2] S. B. Decoste, D., "Training invariant support vector machines," *Machine Learning 46, 161–190*, 2002. DOI: https://doi.org/10.1023/A:1012454411458.