Machine Learning Model based Optical Character Recognition

Aritra Bhattacharyay, Christopher Vega, Drona Banerjee, and Shubhra Deb Paul

***Abstract*— In recent times, optical character recognition is gaining popularity due to the development and the application of machine learning techniques in the field of data science. In this project, we designed and developed a support vector machine (SVM) technique to classify 8 different handwritten lower-case characters. We applied histogram of oriented gradients (HOG) techniques to extract features on the input images, standard scaler to scale it, and finally, principal component analysis (PCA) to reduce data dimensionality. It resulted in an accuracy level of over 99% for 2 classes and 95% for 8 classes. We also implemented a convolutional neural network (CNN) that resulting in an accuracy of over 97% and 95%, respectively. We also trained and tested the dataset using other machine learning algorithms such as K-nearest Neighbors (KNN), logistic regression (LR), and decision tree. Out of all the methods, SVM gives the most optimal performance in terms of speed, robustness, and accuracy.**

I. Introduction

OPTICAL character recognition (OCR), also known as text recognition, is the technique to identify and differentiate handwritten or printed characters in any paper document [1]. In this process, the papers that include texts are converted into digital images by scanning or taking photographs, and then these collected images are given input any data processing tool to perform the character recognition. The necessary background knowledge required is given below:

A. Support Vector Machine (SVM):

SVM is a powerful machine learning technique that can perform binary classification of data. Using SVM, it is possible to create both linear and non-linear separation surfaces within the space of input dataset [2]. According to [3], "When the feature vectors of the training set are linearly separable by a hyperplane, a linear SVM can be built that uses the structural risk minimization principle to decrease classification errors. The separating surfaces are not necessarily linear. In non-linear SVMs, non-linear functions such as high-order polynomials *etc.* are used to characterize the separating surface".

B. Histogram of Oriented Gradients (HOG):

Histogram of oriented gradients (HOG) is a feature descriptor used for object detection. The computed HOG descriptor is fed to SVM classifiers for object detection. HOGs use a single feature vector for an entire image. The edge orientations or the gradient vector are computed at each pixel using a pixel detection window. All the gradient vectors generated in the window are represented as a histogram. The window slides over the entire image, and the HOG descriptor computed for each position is added to the single feature vector, which is ultimately generated [4], [5].

C. Principal Component Analysis (PCA):

The primary purpose or objective of Principal Component Analysis is to reduce the dimensions in data. The number of points needed to populate a space grows exponentially with dimensionality. To fully understand a space in higher dimensions, we need more and more informative and discriminative features. PCA gives us a lower-dimensional manifold representation of the data as when we are in high dimensional spaces, much of the space is empty, and most of the data lives at the surface. PCA identifies the directions of maximum variance in data with high-dimension and hence projects it on a new subspace that contains an equal or fewer number of dimensions than the original one. By ending up data that is uncorrelated, It minimizes the redundancy of the resulting transformed data and minimizes the mean squared error between original and transformed/reduced data, and maximizes the retained variance of the data [6], [7].

The rest of the report is constructed as follows: Section II discusses the implementation of our proposed method in detail. The manipulation of a raw dataset, a complete experimental design including all the results and discussions, are elucidated in Section III, and finally, we wrap up in Section IV.

II. Implementation

A. SVM Based Classification

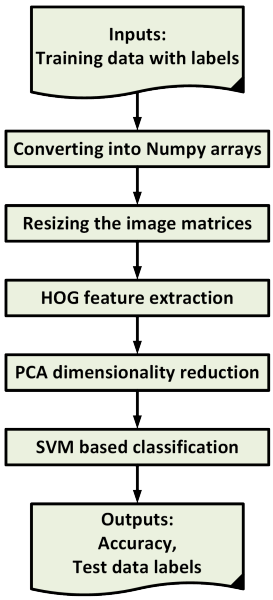

Fig. 1: Algorithm of our implemented image classifier.

1) Algorithm: To solve the problem of image classification of handwritten characters, we are using a method that includes extraction of features using Histogram of Ordinal Gradients(HOG) from the the n-dimensional *Numpy* array, scaling of these extracted features, and hence application of Principal Component Analysis (PCA) on the scaled features to generate a *Pandas* dataframe. The *Pandas* dataframe is then divided into train data, train labels, and test data and test labels. The train data and train labels are used to train the machine learning models, and the test data is used to predict the class labels by feeding them to the trained model. Finally, we compared the predicted labels with the actual labels to compute the accuracy of our prediction.

In the HOG feature extraction step, we use pixels per cell to be 16×16, and blocks of 3×3 cells are the blocks are normalized using the $L2$-norm with hysteresis. The pixels/cell is chosen to be 16×16 as it significantly reduces the training time compared to 8×8 or 2×2 pixels/cell. Additionally, it maintains a high accuracy level. The 3×3 block size is chosen to facilitate sliding through the data points and capturing the gradients efficiently. In most problems and in the seminal work for HOG, a block size of 3×3 was proposed [4]. The $L2$-norm with hysteresis is used as it first normalizes the data and also limits the maximum values of the gradients to 0.2. Furthermore, it is re-normalized using $L2$-norm. As a result, this method ensures better scaling and better normalization of the features. The 2-D data given as input to the HOG classifier is transformed to a flattened 1-D vector of shape 81×1.

The next step is to standardize the data in such a way that the mean of the data becomes 0 and with a standard deviation of 1. We again use a scaling of the feature vector because the feature vector returned by HOG extraction even after applying the L2 norm with hysteresis. The feature vector is again scaled as the L2 norm assumes that the mean is 0, and the variance is in the same order, but if the variance of a feature is in orders of magnitude larger than the variance of other features then it might dominate the objective function [8]. On the scaled data, we apply the PCA algorithm with a number of principal components set to 75. As the length of our feature vector from HOG feature extraction is 81, we wanted to test whether reducing the dimensions has any effect on the performance of the algorithms. We had also tested the performance of the algorithm with a number of principal components to 50,60, and 70. We got the best results when the number of principal components is set to 75.

We then divide the data provided into two parts that are 70% for training and the remaining 30% for testing. The reason behind choosing 30% of the data for testing is because we want to investigate how our algorithm performs on an adequate amount of testing data. We test our algorithm on various machine learning algorithms and explore that the SVM algorithm performs the best in terms of accuracy and robustness as we test the algorithms under various conditions (discussed in Section III), where the data is both normalized or not-normalized at different test/train ratios. We choose the radial basis function (RBF) kernel in our SVM model. The linear kernel also performs well in some cases; however, the RBF kernel is more robust and outperforms others in most scenarios. The other machine learning algorithms we use are K-nearest neighbors (KNN) classification, logistic regression (LR), and decision tree. The KNN algorithm performs similar to SVM, but the performance of SVM performs superior. In the LR algorithm, we use *Newton-CG* method to optimize the log-likelihood.

B. Convolutional Neural Network Based Classification

1) Network Architecture: Fig. 2 depicts our convolutional neural network model, where we experiment with two convolutional layers, each of them followed by a 2×2 max. pooling. After the last convolution, the data is fed into a flattened fully-connected network. Using ReLU (Rectified Linear Unit) activation, data from this fully connected network is passed to another fully connected network, which generates data for the output layer. In our architecture, the first fully connected network layer consists of 120 nodes, while the second one contains 84 nodes. We use 6 filters, each with 3×3 dimension for the first convolutional layer. The second convolutional layer uses 16 filters of the same dimension (3×3).

2) Feature Extraction with Convolution: We use the *Softmax* function in *PyTorch* with *Cross Entropy* for feature extraction. The feature extraction is performed in the last hidden layer, which is a fully connected network with 84 nodes. The *Softmax* function provides us with a vector representing the probability distribution.

3) Convolutional Layer: Our CNN architecture consists of two convolutional layers. The first layer uses 6 filters, each of 3×3 dimension. The second layer has 16 filters with 3×3 dimension. These convolutional layers initialize the filters randomly, and they generate the parameters subsequently learned by our network.

4) Pooling Layer: We have a couple of pooling layers in our architecture. These layers operate on each feature map independently. The primary purpose of the pooling layer is to reduce the number of parameters and computation in the network. It achieves this by progressively reducing the spatial size of the representation.

5) Neuron Activation: We deploy Rectified Linear Unit (ReLU) activation for our neurons. It is a linear function that gives a zero output if the input is zero or negative value. On the other hand, for a positive value of the input, it outputs the same value.

III. Experiments and Results

A. Datasets

We are using the dataset that has been provided to us by the instructor. To classify for classes 1 and 2. We are extracting the data points corresponding to classes 1 and 2, and classifying for classes 1, 2, 3, 4, 5, 6, 7, 8 we are directly feeding the data to our implementation. We are then extracting features from the data and feeding the features to the proposed machine learning algorithm. For our proposed model, we have used 70 percent of the data for training and 30 percent of the data for testing the trained model and evaluating our results. To check the robustness of our model, we have trained and tested it on different training and testing ratios.

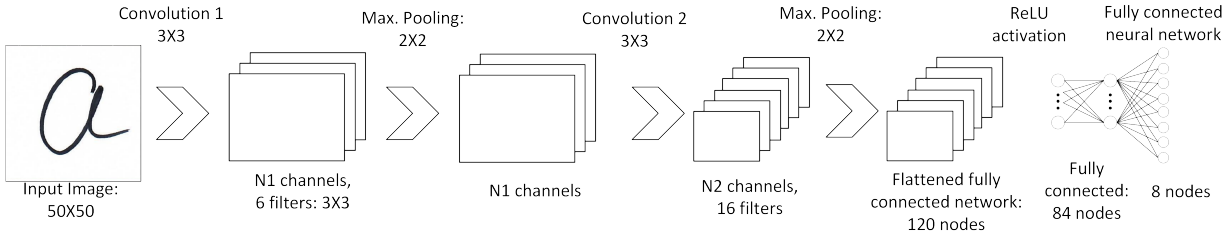

Fig. 2: Implemented Convolutional Neural Network architecture.

B. Classification Results

TABLE I: Accuracy (%) results with only using HOG for the trained models.

With HOG (without scaling, without PCA)

Test-Train Ratio	2 Classes	8 Classes
50-50	95.5	82.75
40-60	96.25	82.89
30-70	96.04	83.17
20-80	96.25	83.59
10-90	96.25	83.28
Kernels	**2 Classes**	**8 Classes**
Linear	97.08	90.46
RBF	96.04	83.17
Polynomial	46.875	11.61
Sigmoid	96.04	80.78
ML Algorithms	**2 Classes**	**8 Classes**
SVM	97.08	90.46
KNN (k=5)	98.33	93.95
LR	96.875	88.02
Decision Tree	95.625	81.19

Table I summarizes the % accuracy results with only using HOG for the trained models. The goal of the experiment is to find out how our algorithm performs under different circumstances without any scaling and PCA. We also wanted to find out the performance of other machine learning algorithms under similar circumstances. The findings show that the HOG features alone work well when we need to classify for 2 classes. But as the number of classes increase to 8, we can see through experiments that the accuracy of all the algorithms reduce considerably. Also it can be noted that the KNN algorithm works better than other algorithms.

TABLE II: Accuracy (%) results with only using HOG and PCA (without scaling) for the trained models.

With HOG and PCA (n=75) (without scaling)

Test-Train Ratio	2 Classes	8 Classes
50-50	95.5	83.00
40-60	96.25	83.32
30-70	96.04	83.38
20-80	96.25	83.67
10-90	96.25	83.43
Kernels	**2 Classes**	**8 Classes**
Linear	97.08	90.46
RBF	96.04	83.38
Polynomial	46.875	11.61
Sigmoid	96.04	81.19
ML Algorithms	**2 Classes**	**8 Classes**
SVM	97.08	90.46
KNN (k=5)	98.33	93.95
LR	96.875	87.96
Decision Tree	95.83	81.71

Table II includes the accuracy results with only using HOG and PCA (without scaling) for the trained models. The goal of the experiment is to find out the effect on the accuracy of the algorithms when the dimensionality of the feature vector we get from HOG is reduced to 75 features. So, our main goal in this experiment is to analyze the effect of PCA on the accuracy of the algorithms and mainly on the SVM algorithm with the radial basis function kernel. The experimental results show that there are minimal differences in the performance of the algorithms as compared to feeding the algorithms only HOG features.

TABLE III: Accuracy (%) results with only using HOG and scaling (without PCA) for the trained models.

With HOG and scaling (without PCA)

Test-Train Ratio	2 Classes	8 Classes
50-50	98.125	94.25
40-60	98.59	95.19
30-70	98.75	95.46
20-80	99.375	95.39
10-90	98.75	95.93
Kernels	**2 Classes**	**8 Classes**
Linear	96.45	88.02
RBF	98.75	95.46
Polynomial	98.95	95.26
Sigmoid	94.58	81.25
ML Algorithms	**2 Classes**	**8 Classes**
SVM	98.75	95.46
KNN (k=5)	98.75	93.02
LR	96.875	88.9
Decision Tree	95.625	81.19

Table III compares the accuracy (%) results with only using HOG and scaling (without PCA) for the trained models. The goal of the experiment is to find out the effect on the learning of the machine learning algorithms when the extracted features from the HOG classifier is scaled using standard scaler and then fed to the algorithms. We have also tested the performance of the SVM algorithm with a radial basis function kernel under different test train ratios. According to the given experimental results, we have found that the SVM algorithm with radial basis function kernel gives high accuracy and little variation in the performance when the test train ratio is varied. We have also observed that most of the algorithms that have been tested, perform well with data that has been scaled after extracting HOG features.

Table IV sums up the accuracy results, including HOG, linear scaling, and PCA for the trained models. The goal of the experiment is to find out the performance of the machine learning algorithms when the HOG features are extracted from

TABLE IV: Accuracy (%) results including HOG, linear scaling, and PCA for the trained models.

With HOG, Scaling, and PCA (n=75)

Test-Train Ratio	**2 Classes**	**8 Classes**
50-50	98.375	94.43
40-60	98.59	95.03
30-70	99.16	95.52
20-80	99.375	95.39
10-90	98.75	95.93
Kernels	**2 Classes**	**8 Classes**
Linear	96.04	89.73
RBF	99.16	95.52
Polynomial	98.95	95.02
Sigmoid	94.58	79.58
ML Algorithms	**2 Classes**	**8 Classes**
SVM	99.16	95.52
KNN (k=5)	98.54	93.125
LR	97.08	88.95
Decision Tree	97.08	77.76

the data, and then the features are scaled, after scaling when the feature dimension is reduced using PCA with 75 principal components. We have also tested the robustness of the SVM classifier with the RBF kernel by training and testing the algorithm on different test-train ratios. According to the given experimental results, the SVM algorithm with radial basis function gives the best accuracy, and the results on the different test-train ratio are better than previous results. We also find that most of the algorithms perform well under the current experimental conditions. The performance of classification on class 1 and class 2 is suitable for most of the algorithms and the different kernel configurations of SVM. In some cases, the performance on all classes classification is not good.

On the other hand, the results with CNN is given below:

#Classes	Accuracy (%)	Training time (s)
2 classes	3	43.95
8 classes	254	187.99

C. Confusion Matrices

The confusion matrix for 2 classes and 8 classes using SVM technique is given below:

222	3
1	254

236	0	2	2	1	3	1	1
0	230	0	6	6	1	2	2
1	2	231	1	1	3	0	1
3	4	1	236	0	1	0	0
0	2	3	2	210	2	0	4
0	0	1	2	3	222	5	0
0	2	0	2	2	6	224	0
2	0	0	0	2	3	2	245

The confusion matrix for 2 classes and 8 classes using CNN technique is given below:

266	9
2	251

266	4	5	2	0	0	1	0
0	254	1	1	4	2	0	3
1	2	256	0	0	4	1	2
0	3	3	261	3	5	2	5
0	4	1	1	238	0	0	6
0	1	4	2	0	234	5	2
1	0	1	2	0	7	263	0
2	2	2	0	4	0	2	237

IV. Conclusion

In this report, we presented a few machine learning approaches to classify handwritten characters. We designed a SVM based classifier that has an accuracy level of above 99% for two classes ('a' and 'b') and over 95% for 8 classes ('a', 'b', 'c', 'd', 'h', 'i', 'j', and 'k'). We applied different techniques such as HOG, linear scaling, PCA, *etc.*, to achieve this high accuracy levels. We discovered that applying linear scaling after feature extraction with HOG, and our model is able to achieve very high accuracy. We also investigated the effect of PCA by comparing these results with the results of without-PCA and discovered that PCA increases the prediction accuracy. In our case, the number of principal components to give the best results is 75. We also performed our experiments by varying different train-test ratio and compiled them in Section III-B. Additionally, we applied different SVM kernels and explored that the RBF kernel gives more robust results with higher accuracy in different scenarios. The linear kernel also gives good results, but the implementation we have followed in that the RBF kernel gives the best results. Besides SVM, we also designed a CNN based classifier. Finally, we also included our training and testing results by applying three other machine learning algorithms such as KNN, Logistic Regression with optimization using newton's method, and Decision Tree. Comparing all the results, we conclude SVM as our final proposed model because it results in higher accuracy in spite of its simplicity in design and implementation as well as a comparatively less computationally expensive training time as compared to CNN.

References

[1] TechTarget, "Definition: OCR (optical character recognition)," https://searchcontentmanagement.techtarget.com/definition/OCR-optical-character-recognition, Accessed: 2019-12-01.

[2] J. A. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural processing letters*, vol. 9, no. 3, pp. 293–300, 1999.

[3] Z. TARIGULIYEV, "Reliability and security of arbiter based physical unclonable function," Master's Thesis, January 2011.

[4] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, June 2005, pp. 886–893 vol. 1.

[5] A. J. Newell and L. D. Griffin, "Multiscale histogram of oriented gradient descriptors for robust character recognition," in *2011 International Conference on Document Analysis and Recognition*. IEEE, 2011, pp. 1085–1089.

[6] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.

[7] M. E. Tipping and C. M. Bishop, "Probabilistic principal component analysis," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 61, no. 3, pp. 611–622, 1999.

[8] S. Learn, "sklearn.preprocessing.StandardScaler)," https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html, Accessed: 2019-12-01.