1. Introduction

In this assignment we were tasked to create three different Machine Learning Classifiers with a combination of features for the handwritten digit classification dataset MNIST. In the next few parts of this report I will be giving a brief description of the features and classifiers to be used, the dataset MNIST, I will also be discussing my settings, results and findings using the combination of features and classifiers, and finally a conclusion about my learning and understanding about the project.

First, a brief explanation for the features and three different classifiers used in this project; the features to be used in this project are Histogram of Oriented Gradient (HoG) and Local Binary Pattern (LBP). HoG is a hand-crafted and gradient based feature that is mainly and best used for corner and edge detection in image processing. HoG splits up an image into cells and from this acquire useful information and uses one direction for each pixel. HoG is a speedier and more economical feature as compared to LBP due to use of much simpler computations. LBP on the other hand is more cost heavy yet also an effective hand-crafted and texture descriptor and uses 8 directions for each pixel, it transforms greyscale images at pixel level to a matrix of integers. Both these features are useful tools and descriptors in their own rights, HoG and LBP are known to complement each other very well in the world of image processing and pattern recognition.

KNN is recognized as one of the simplest and easiest supervised algorithms in Machine Learning for classification and regression. This classifier computes the distance between data using different measures such as Euclidean and Manhattan measurements and classifies new data points based on these. SVM is another simple and supervised Machine Learning algorithm known for classification and regression. SVM finds a hyperplane or the most optimal boundary and clearly categorizes data points based on this boundary. ANN on the other hand is more complex as it consists of multiple layers, that comprises of an input layer, different hidden layers, and an output layer. ANN is basically an interlinked neural network compromised of a combination of layers and is considered a more powerful type of classifier as compared to both KNN and SVM.

2. Dataset

The dataset to be used in this project is the MNIST English handwritten numerals from the MNIST database and this dataset is one of the most commonly used and known to be the benchmark for image classification in machine learning. This dataset contains 60,000 training images and 10,000 testing images. It contains different handwritten images of the numbers 0 to 9. Figure 1 will be a screenshot of some examples of images that are part of this MNIST dataset.

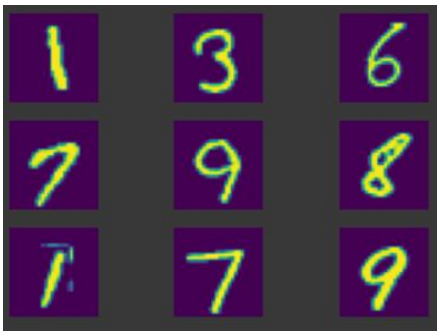

Fig 1. Screenshot of images of handwritten English numerals from the MNIST data set.

3. Experimental results and discussions

3.I Experimental settings

The MNIST data has been flattened to the feature vector of size 784, so depending on the model, I had to reshape the feature vector back to 28x28 for the models that were using a 3d array. There was also some normalization done by changing the pixel values from 0 and 255 to 0 and 1. Some models were also occasionally distributed by training sets, testing sets, and validation sets.

With each classifier and features there were several different parameters and settings that needed to be placed. First, for KNN I was originally supposed to loop over the different *kVals* to attain the highest accuracy from the different indices, but it took too long to achieve, and I noticed that most values were somewhat close to one another so I didn't bother adding it to my code. So, I set 3 random values for *k* for the 3 different models: *KNN*: 1, *HoG-KNN*:5, and *LBP-KNN*:3.

For the SVM classifier when I applied both *HoG* and *LBP* I used the *rbf* kernel because it produces a non-linear hyperplane and I believed it would produce better accuracies for the models with classifiers and applied the *linear* kernel for the raw pixel SVM model. I also used the last 10,000 images for train and 1,000 images for test for quicker processing time for the raw pixel model. The penalty parameter that was setup was *C=100* for the *HoG* and *LBP* feature and *C=.001* for the raw pixel model. I wanted to have a smaller margin using the *HoG* and *LBP* features and used a larger margin for the raw model. Lastly, the pseudo random number of 42 was used for all 3 SVM models.

I used two optimizers for my ANN models, one using the Adam optimizer and the other using the *SGD* optimizer, with a learning rate of 0.01 for *SGD*. I set *epochs* at 15 and used the batch size of 100 for the *HoG* and raw pixel and 50 for *LBP* and set the validation split to 0.1 to avoid bias and check if I was overfitting. For the *HoG* and *LBP* models I skipped the Flatten layer and input the dimensions in the first layer of dense depending on the shape after feature extraction but still had to add the Flatten layer for the raw pixel model. I created 4 dense layers with 1024, 512, 256, and 10 neurons respectively to optimize my neural network and used different activation

methods from *tanh* and *sigmoid* for the raw pixel model to *ReLU* for the *HoG* and *LBP* models and used *softmax* in the last layer of all models. Lastly, I used *sparse categorical crossentropy* for the loss function for all the classes are mutually exclusive to one another.

3.2 Experimental Results

3.2.1 Confusion matrix for KNN and SVM

I picked the highest accuracies of confusion matrices from both KNN and SVM models. I believe the raw pixel models came out with the best accuracy for I used less data as compared to my *HoG* and *LBP* models. Though the accuracies were still very close to one another e.g. (*HoG-SVM = .90*) the number of data used may have played a part in acquiring a high accuracy score.

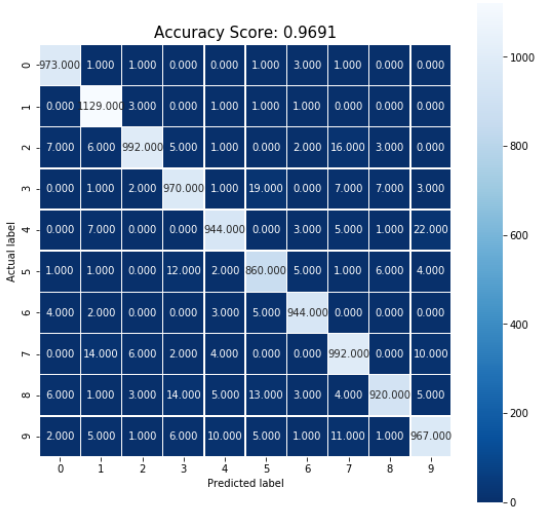

Fig 2. Confusion Matrix for KNN

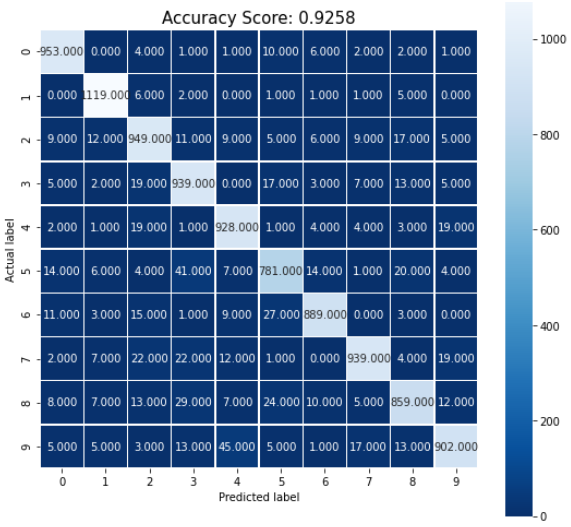

Fig 3. Confusion Matrix for SVM

3.2.2 Learning Curves for ANN

The learning curve for the *HoG-ANN(Adam)* model shows that it wasn't overfitting and the model acquired the highest test accuracy of 0.967 as compared to the other ANN models.

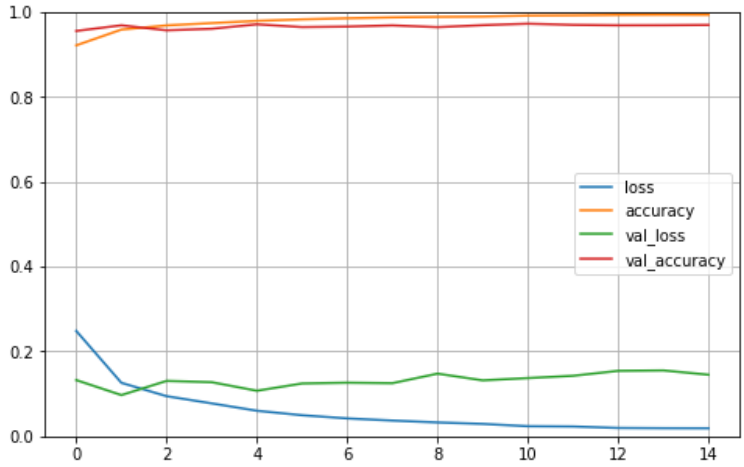

Fig 4. Learning Curve for HoG- ANN(Adam)

3.2.3 Comparative study

Classifier/Feature	HOG	LBP	Raw Input
KNN	**Train**: 0.90 **Test**: 0.867	**Train**: 0.65 **Test**: 0.434	**Train**: 1.0 **Test**: 0.969
SVM	**Train**: 1.00 **Test**: 0.902	**Train**: 0.51 **Test**: 0.521	**Train**: 0.983 **Test**: 0.925
ANN	**Adam**- **Loss**: 0.152 **Acc**: 0.967 **SGD**- **Loss**: 0.295 **Acc**: 0.922	**Adam**- **Loss**: 1.311 **Acc**: 0.528 **SGD**- **Loss**: 1.839 **Acc**: 0.317	**Adam**- **Loss**: 0.134 **Acc**: 0.958 **SGD**- **Loss**: 0.133 **Acc**:0.959

3.2.4 Discussion

The errors while using the classifiers are there because the models haven't been optimized to its fullest potential, though the accuracy rate for both raw input and *HoG* are acceptable, not understanding how to fine-tune these models have caused error rates, especially while using the *LBP* feature. Even while trying to fix the parameters of each model, I was only able to attain a few improvements in the accuracy and, I tried as much as possible to not cause bias and overfitting to the models. The lack of fine-tuning by using the *LBP* feature has caused a huge effect on the accuracy of each classifiers, even though it is meant to improve or help achieve a higher accuracy rate.

Based on the three classifiers, it is evident that the accuracy of ANN, on average, is higher as compared to both KNN and SVM. I believe that this is precise due to the fact of ANNs being more of a fit for an experiment like this and in addition, handling a huge dataset suits ANN better. ANNs are more capable of handling problems of this extent, not saying that both SVM and KNN can't, but just the size of the data itself proved that it was computationally heavier for these classifiers. Each classifier has their own strengths in their own rights, but for this problem ANN performed more efficient and provided better results in general as compared to the other two.

Below is a screenshot of the results by using the classifier KNN and having the *LBP* feature implemented. We can see that the smallest curves have caused the wrong interpretation by the model and is caused by the lack of adjusting and improving the model.

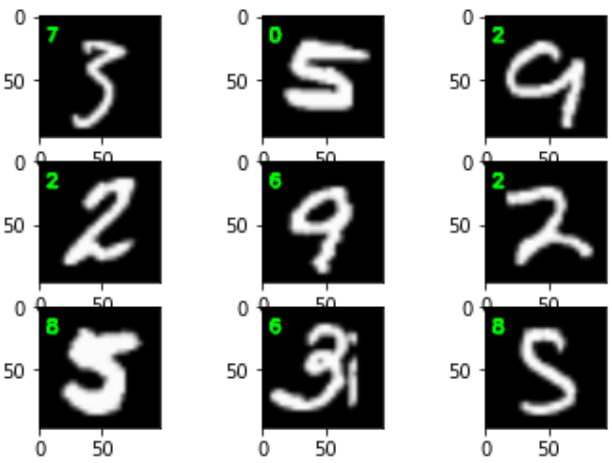

Fig 5. Samples that were wrongly classified from LBP-KNN model

4. Conclusion

Based on the results of the experiment I was able to grasp further on how powerful and cost-effective ANNs are as compared to both SVM and KNN, especially when dealing with the project given in-hand. Additionally, understanding the dataset, the features to be utilized, and models to be used are very vital in order to solve the problem you are trying to work out in order to optimize and attain great outcomes from different experiments. Evidently, we can see that the final results of the project that ANN models scored the highest average when it came to accuracy, I assume this may be due to the complexity of the classifier and also how it suits completely for a huge dataset such as the one used for the experiment. Lastly, even if ANNs proved to be more efficient than both SVM and KNN, based on the results I can conclude that both classifiers were still able to generate high test accuracies, but yielding these results at a much slower pace. This may be due to the fact that both SVM and KNN are supervised classifiers and the results gained are more what the user wants to attain rather than the machine doing the work.