42028: Deep Learning and Convolutional Neural Network

Assignment -1

Student Name: Andreas Gwyther-Gouriotis

Student ID: 13893627

Introduction:

This report outlines the process of designing and testing multiple machine learning classification models to identify handwritten digits supplied by the MNIST data set. The classifiers used in the experiment include a k-Nearest Neighbour (KNN) model, a Support Vector Machine (SVM) and an Artificial Neural Network (ANN). Each of these models will be used to test whether a computer can learn to read handwritten numbers and their results will be recorded and compared based on their accuracy and other measures such as a confusion matrix. On top of testing each model on the raw pixel data each model will be tested on data after feature extraction has taken place using one feature extraction technique. The feature extraction technique that will be explored in this experiment is Histogram-of-Oriented Gradient (HOG) feature extraction. The results from each of the models on the raw data and feature extracted data will be compared and discussed to provide an understanding of how each of the models work, why certain input data may perform better than one another and why some models may perform better than others.

Dataset:

The dataset used was the MNIST database created by Yann LeCun, Corinna Cortes and Christopher J.C. Burges. The dataset is a subset of a larger database from NIST and includes 60,000 handwritten digits ranging from 0 – 9 for training and 10,000 testing samples. The dataset has been pre-processed meaning all the images have been centred and are 28x28 pixels in size. This means no pre-processing is required and energy can be focused on developing an understanding of the machine learning and feature extraction techniques instead.

A sample of the dataset is as follows and shows what the images being passed into the different classification models look like:

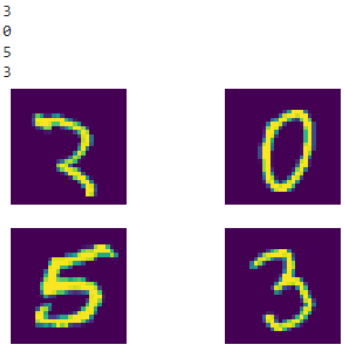

Figure 1: Sample of handwritten digits. (Images read left to right, then down), (labels read down).

The above figure shows the labels and images from the dataset and highlights the main challenge when designing the machine learning techniques to classify the handwritten digits. Since the digits are handwritten, they aren't all identical which is why machine learning is needed to identify them. However, because of this feature some digits are much harder to identify and even look like other numbers from a human's perspective. Take the first image in the top left of the figure. From checking the label, we know that the number in the top left is a 3, however, it could also be easily mistaken for a 2 unlike the image in the bottom right which is undoubtably a 3. This challenge is what caused most of the

discrepancy in the accuracy of the models and is well documented and highlighted in the confusion matrix in the results section.

Finally, a sample of each class can be seen below:

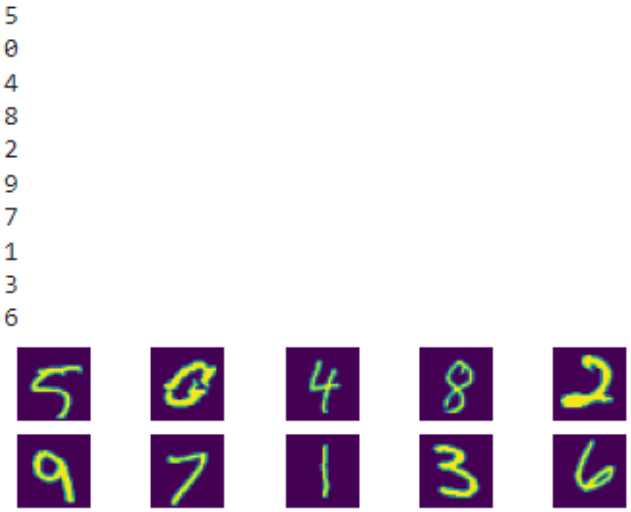

Figure 2: Sample of each class (0-9) from the MNIST dataset.

The other data set that the models will be tested on is the MNIST data set after feature extraction has taken place using Histogram-of-Oriented Gradient (HOG). This feature extraction technique works by computing the image gradient by using the horizontal and vertical filters to compute gradient values. Then computing the strength and direction of each gradient and divide the image into small, connected regions called cells which are 8x8 and put into a 9-bin histogram. The image is then normalized by 2x2 cells resulting in a block that represents a 36 x 1 element vector. The blocks are then concatenated into one big vector essentially taking a 2D image and turning it into a 1D vector of gradient lines. An image gradient is described as a directional change in the intensity or colour in an Image and is used to extract valuable information from images and is used especially in edge detection. A sample of the resultant feature extraction using the HOG technique is seen below:

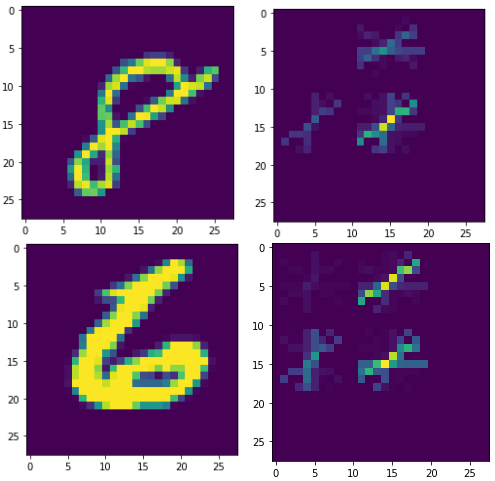

Figure 3: Sample of feature extraction being used on the MNIST dataset.

Experimental results and discussion:

a. Experimental settings:

The machine learning techniques used in the experiment include a k-Nearest Neighbor (KNN) model, a Support Vector Machine (SVM) and an Artificial Neural Network (ANN). Each of these models have different parameters that change the way they function in order to obtain faster training results, more accurate results and to overall fit the data better to make learning and prediction more effective and accurate.

The first model that was experimented with was the KNN. A KNN is a simple supervised learning algorithm that can be used for both classification and regression. It is non-Parametric and thus doesn't make any assumptions on the data distribution. It works by computing the distance between the new sample and all training samples around it. The Distance can be measured using different methods including Euclidean, Manhattan, etc. When classifying a point, it picks 'k' entries in the training set which are closest to the new sample. The majority voting decides the class of the new sample. To understand how the KNN functions, figure 4 shows the basic idea:

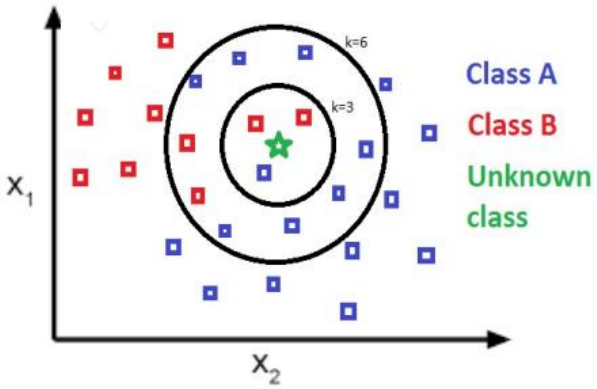

Figure 4: Diagram of a KNN model with a k of 3 and 6 with two classes and one unknown data point to be classified (Dr. Nabin Sharma, 2022, https://canvas.uts.edu.au/courses/22100/files/2550928?wrap=1)

From the figure if the KNN model was to try and classify the green star with a k set to 3 it would look at the 3 closest data points to the green star. In this case there are 2 red data points and 1 blue data point, thus the KNN model would consider the green star to be a red data point as the majority of nearest neighbours to the new sample point are red. When applying this model to the MNIST data set the KNN model was set to have a k of 3. For example in the experiment, if a '7' was surrounded by a '1' and two other '7's then the KNN would classify the 7 as '7'.

The next model to be experimented with was the SVM. The SVM is a more sophisticated technique with multiple hyper parameters. A SVM can perform linear or non-linear classification, regression and outlier detection and is defined by a separating hyperplane. It is typically suitable for small or medium sized datasets as it is typically quite accurate but also takes a very long time to train when compared to the

KNN or Neural Network. The figure below provides an example of how the hyperplanes work when classifying two datatypes with an SVM.

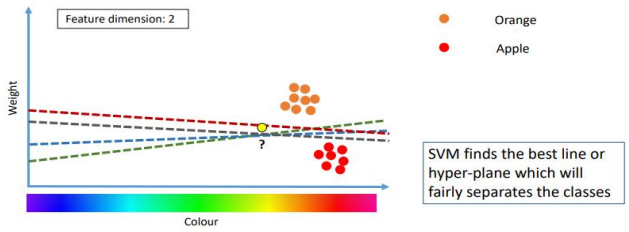

Figure 5: Diagram of an SVM classifying an orange vs an apple based on its colour. (Dr. Nabin Sharma, 2022, https://canvas.uts.edu.au/courses/22100/files/2550928?wrap=1)

SVM has a lot of parameters, however, the main ones considered for the experiment included C, kernel, degree, gamma and random state. C is a regularization parameter where "the strength of the regularization is inversely proportional to C" (scikit-learn developers, 2022). It controls how much to avoid miss classifying each training sample. For example, for a low C value the hyperplane may sit equally between two classifications not caring whether one sample point is classified correctly, however, with a high C the optimization will pick a hyperplane with a smaller margin to a classification if that hyperplane does a better job of classifying all the points correctly. Figure 6 highlights this function.

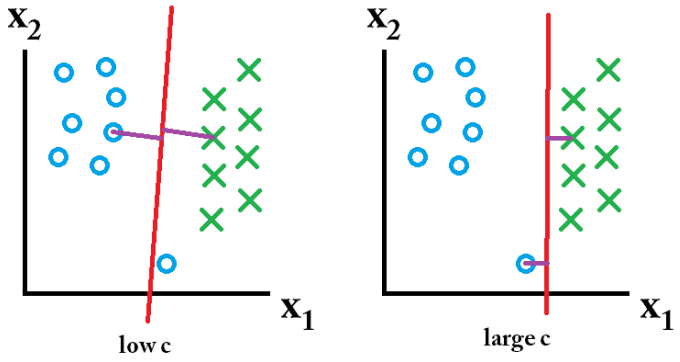

Figure 6: Shows the difference between a low and high C value for an SVM. (Kent Munthe Caspersen, 2015, https://stats.stackexchange.com/questions/31066/what-is-the-influence-of-c-in-svms-with-linear-kernel)

The kernel of an SVM dictates the mathematical function that the SVM will use to classify the data points. With sklearn the following kernels can be selected 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'. The linear kernel means that the classifications are separated by a linear hyperplane much like in figures 5 and 6 while the poly, Radial Basis Function (rbf) and sigmoid kernel are used for non-linear datasets as the poly parameter uses a polynomial function to separate the data. The rbf uses points that are defined as distances from a centre for example a bell curve is considered a rbf as the points are represented as the number of standard deviations from the mean and finally the sigmoid parameter is non-linear as it uses the sigmoid function to separate the data. The degree parameter controls the flexibility of the hyperplane's decision boundary and is defaulted to 3. The higher the degree of the kernel the higher the flexibility of its decision boundary.

The gamma determines how much far away points determine the hyper plane for example if there is a high gamma then only the close points to the hyperplane

determine its position while a low gamma means even points that are far away from the hyperplane determine its positioning. Finally, the random state parameter determines an internal random number generator that splits the data when training is taking place. By changing the random state, it ensures that the splits that you generate are reproducible as the random numbers are generated in the same order based on the seed or number you set the random state to. This parameter is ignored when probability is set to false thus by default is ignored, however, was necessary for testing the best parameters for the SVM.

The parameters that were chosen for the SVM include a rbf kernel, a high C value of 150 (the default is 1) and a random state of 54. The rbf kernel was chosen as it allowed for the quickest training time while a high C value was chosen as when it was set to default it resulted in less accuracy overall. The other parameters were kept at default as the accuracy decreased when parameters gamma and degree were changed.

The final machine learning technique that was experimented with was the ANN. An ANN is a model inspired by the biological neural networks that make up our own brain. The ANN that will be focused on for the experiment is a multi-layered neural network that essentially builds upon a single layer perceptron by adding multiple perceptron's through different layers. The idea being that the first hidden layer will be able to learn some very simple patterns while each additional hidden layer will somehow be able to learn progressively more complicated patterns. The parameters of an ANN include input neurons/nodes, an activation function, loss function and an output layer with multiple hidden layers in-between. The input layer is where the image is inputted, since the images are 28x28 there are 784 pixels meaning there needs to be 784 neurons for the input layer. With each layer of the ANN the number of neurons decreases thanks to an activation function. An activation function determines whether a neuron should be activated based on a certain requirement. In the experiment a ReLu activation function was used for the hidden layers. This function follows this simple logic, if $x < 0$, $A(x) = 0$, if $x > 0$, $A(x) = x$, to activate a neuron and is seen in figure 7.

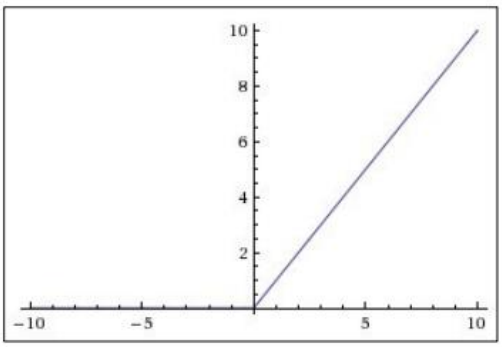

Figure 7: Shows the ReLu activation function. (Dr. Nabin Sharma, 2022, https://canvas.uts.edu.au/courses/22100/files/2550923?wrap=1)

The ReLu activation function avoids the disappearing gradient problem that other activation functions such as sigmoid and tanh encounter. The sigmoid and tanh activation functions are non-linear much like the ReLu function, however, towards to the end of the curve, the change in the value of Y varies much less than the changes in the X values. Hence gradient at the region will be very small, this results in the

network refusing to learn or learning extremely slowly. Figure 8 shows this vanishing gradient problem.

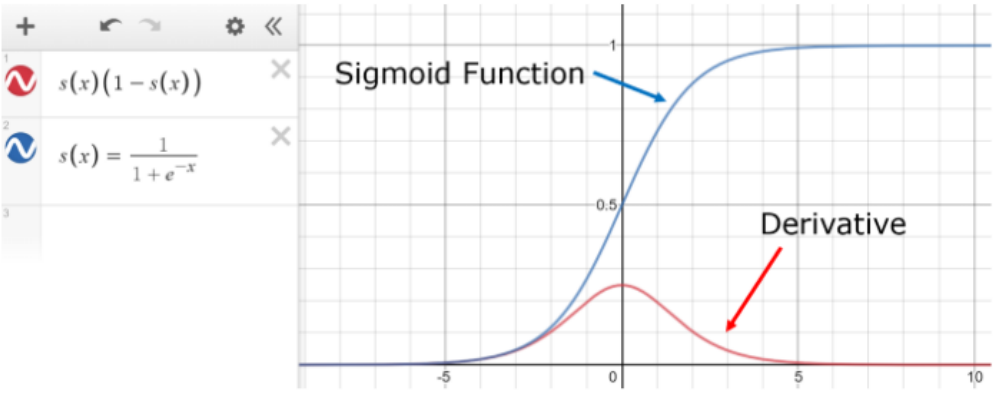

Figure 8: Shows the vanishing gradient problem for a sigmoid function. (Andre Ye, 2020, https://towardsdatascience.com/if-rectified-linear-units-are-linear-how-do-they-add-nonlinearity-40247d3e4792)

The loss function of an ANN quantifies the difference between the expected outcome and the outcome produced by the machine learning model. The loss function is calculated so that the machine learning algorithm can track it's progress through gradient descent and back propagation. Essentially the loss and cost function are calculated and with each iteration the weights and biases (w and b) of the machine learning algorithm are changed in order to achieve a smaller loss value. In the experiment a loss function called 'sparse_categorical_crossentropy' was used which computes the cross-entropy loss between true labels and predicted labels. This loss function is used for binary (0 or 1) classification applications and takes in a predicted value and a true value to determine the cross-entropy. This loss function worked best with the data and produced very good results.

Finally, the output layer of the ANN uses 10 neurons as there are 10 classification types for the MNIST data set (0-9). Each output needs its own neuron thus 10 was chosen. The activation function for the final output layer was softmax which predicts a multinomial probability distribution and is used for multi-class classification problems where class membership is required on more than two class labels. The final set up of the ANN was 784 neurons for the input layer, 256 neurons for the first hidden layer with an activation function ReLu, 128 neurons for the second hidden layer with an activation function ReLu and 10 neurons for the output layer with an activation function softmax. The metric to optimise the ANN on was set to Accuracy and the optimiser Adam was used with the 'sparse_categorical_crossentropy' loss function. These settings together produced quite an accurate model as seen in the results section.

b. Experimental results:

i. **Confusion matrix:**

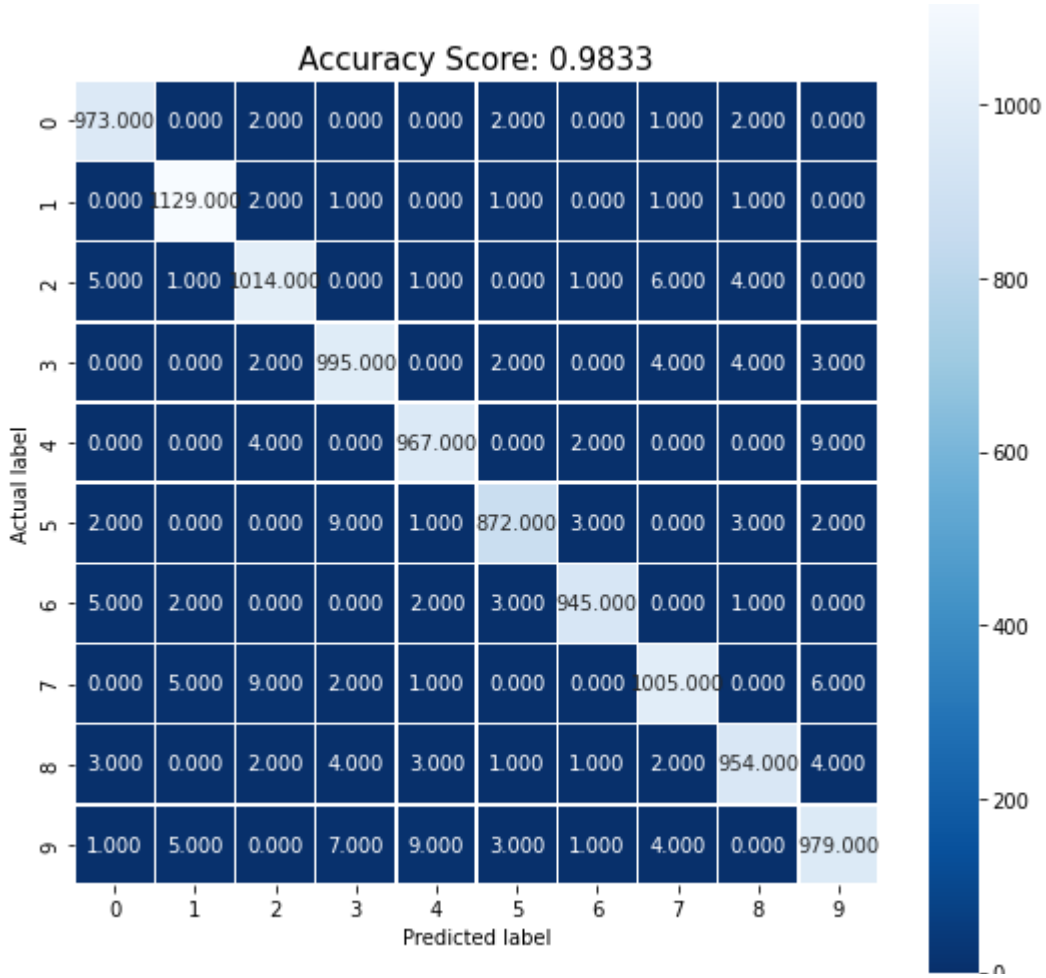

Figure 8: Confusion Matrix for the best predictive model (SVM classifier on raw data)

Figure 8 shows the confusion matrix for the SVM classifier. This classifier achieved 100% accuracy on the raw training data and 98.3% accuracy on the raw testing data as seen above. To understand how to read the confusion matrix take '0' from the Actual label axis and the Predicted label axis. The confusion matrix shows that at (0,0) in the confusion matrix there were 973 predictions. This means that '0' was predicted correctly by the SVM 973 times. However, by looking along the predicted label axis you can see that even though the actual label was '0' the SVM predicted that a '0' was in fact '2'. This happened twice which can be seen on the confusion matrix at (2,0). By looking at the confusion matrix you can also see that the SVM thought that '0' looked like '5' and '8' twice and '7' once. The matrix also helps identify what number it got wrong the most. At (4,9) the matrix shows the SVM thought '9' looked like '4' nine times and at (3,5) it thought '5' looked like '3' nine times. Information like this is useful to show where areas of improvement can take place and reasons for why the SVM predicted these digits incorrectly will be discussed in the discussion section.

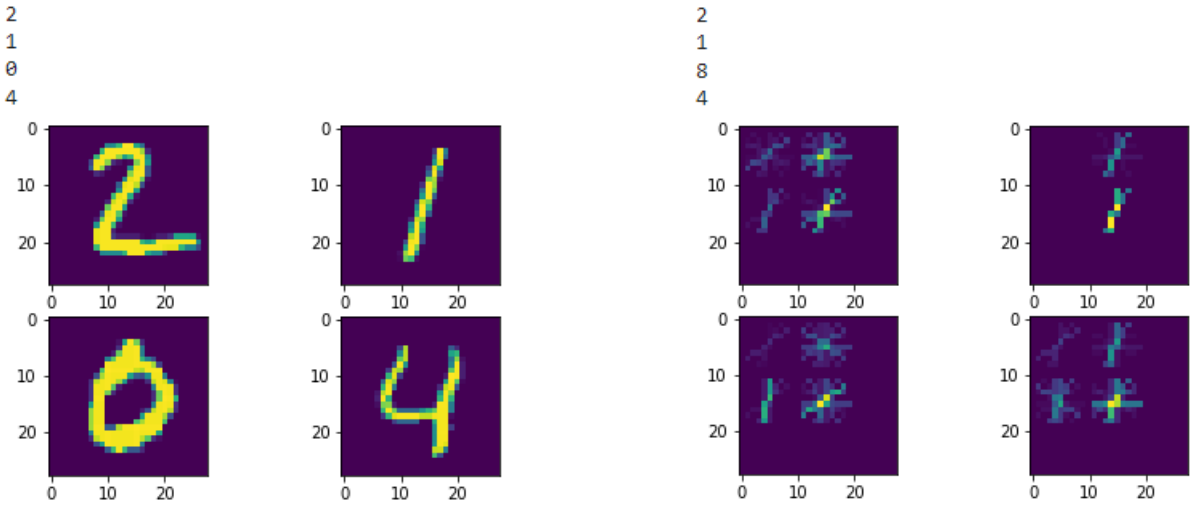

Figure 9: Shows some predictions the SVM model made on the raw and HOG data.

Figure 9 shows some of the predictions for the raw and HOG data by the SVM. Even though it is hard to make out what the HOG data numbers are, by comparing them to the raw data on the left it shows that the SVM was able to predict 2,1 and 4 correctly using the HOG data but incorrectly labelled '0' as '8'. On the raw data side, the SVM was able to correctly predict the four numbers.

ii. **Comparative study:**

Classifier/Feature	HOG	Raw Input
KNN	0.8676	0.9705
SVM	0.9068	0.9833
ANN	0.7116	0.9785

iii. Discussion:

The results from the comparative study gave an interesting look at what methods and techniques worked better when attempting to make a classification model for the MNIST dataset. It should be first stated that the SVM model achieved the best on both the raw data and HOG data but achieved the best results on the raw input data. This made sense seeing as the SVM is a much more robust classifier when compared to the KNN and although the ANN also did better than the KNN, seeing as the ANN used to experiment with was reasonably small and not a lot of time was spent optimising it the SVM was the best model for the job. If the SVM was compared with a convolutional neural network the results may have been different. However, the SVM achieved very high results on both the testing and training data. What was most interesting was seeing that the feature extraction in all cases performed worse than the raw data. It particularly performed worse on the ANN but this was most likely due to the ANN being optimised for 28 by 28 pixel images not a 1D vector. The reason for the HOG data performing worse can be seen visually when compared with the raw data in figure 9. From this figure the variation between HOG data is visually quite limited and less obvious than the raw data. In fact, it would be unlikely that any human would be able to predict the numbers by looking at the HOG data which also

goes to show the power of machine learning as the classifiers were able to extract patterns from the HOG data that humans would not be able to. This is also shown by the SVM achieving a respectable 90.7% accuracy on the HOG test data. The raw MNIST data as discussed in the dataset section, had also gone through a lot of pre-processing meaning the data was already optimised for the classification models, whereas a classification model classifying a random image from the internet will certainly gain more accuracy if the image undergoes feature extraction using HOG.

The main reason for the error in accuracy seen in the confusion matrix is due to a lot of handwritten digits looking similar to each other causing the machine learning algorithms to get confused. For example, when looking at figure 1 at the start of the report the first digit shown is labelled as 3, however, even if a human looked at the handwritten digit, they would most likely mistake it for a 2. This idea that the classifiers are getting mixed up is also supported by the confusion matrix in figure 8 as numbers that look similar such as 2 and 7, 6 and 0, 4 and 9, 3 and 5 all have much higher false predictions than other number that look starkly different such as 6 and 1. What was most surprising was the fact that the SVM never got confused by 1 and 7 which seemed like the most likely two numbers to confuse the classification models. An example of how the ANN incorrectly predicted a number is shown below.

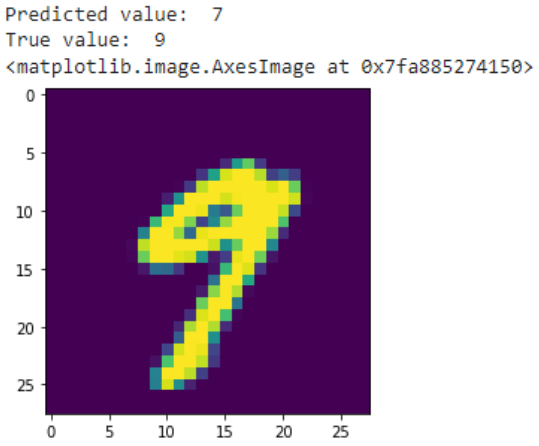

Figure 10: Shows an ANN classification on the raw data with the predicted value being 7 but the true value being 9.

Conclusion

The experiments showed the power of machine learning with its potential to identify patterns that humans wouldn't be able to and showed how techniques such as KNN, SVM and ANN can be applied to real world scenarios. The experiments showed that even within a limited time frame models can be trained to achieve very high accuracy to classify different samples of the MNIST data. It also showed areas that can be targeted for improvement and which numbers from the MNIST data set the machine learning techniques struggled on. Overall, the experimental process highlighted the differences between raw and feature extracted data and the difference between the machine learning techniques, KNN, SVM and ANN.

References:

Ye, A. (2020, September 02). If Rectified Linear Units Are Linear, How Do They AddNonlinearity? Retrieved from https://towardsdatascience.com/if-rectified-linear-units-are-linear-how-do-they-add-nonlinearity-40247d3e4792

Ye, A. (2022, January 05). Radial Basis Functions, RBF Kernels, & RBF Networks Explained Simply. Retrieved from https://medium.com/dataseries/radial-basis-functions-rbf-kernels-rbf-networks-explained-simply-35b246c4b76c#:~:text=RBF kernels place a radial,one layer of output neurons.

Alfaalfa 2, Marc ShiversMarc Shivers 2, Kent Munthe CaspersenKent Munthe Caspersen 3, Dikran MarsupialDikran Marsupial 47.2k55 gold badges121121 silver badges178178 bronze badges, Deerishideerishi 21122 silver badges44 bronze badges, Luzluz 5111 silver badge22 bronze badges, . . . BradBrad 52244 silver badges1010 bronze badges. (1960, April 01). What is the influence of C in SVMs with linear kernel? Retrieved from https://stats.stackexchange.com/questions/31066/what-is-the-influence-of-c-in-svms-with-linear-kernel

LeCun, Y. Cortes, C. Burges, J.C, C. (n.d) THE MNIST DATABASE of handwritten digits. http://yann.lecun.com/exdb/mnist/

Sklearn.svm.SVC. (n.d.). Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

Sharma, N. (2022, February 28). Machine Learning and Image Processing Basics [Lecture 2]. University of Technology Sydney, NSW, Australia. https://canvas.uts.edu.au/courses/22100/files/2550928?wrap=1

Sharma, N. (2022, March 14). Neural Network in details [Lecture 4]. University of Technology Sydney, NSW, Australia. https://canvas.uts.edu.au/courses/22100/files/2550923?wrap=1