MNIST Digit Recognition

Comparison of Machine Learning Techniques

This report describes several different classification techniques in machine learning field and compares their prediction accuracy using MNIST dataset. The best performing classifier for this task is found out to be Convolutional Neural Network.

Siow Meng Low

01 May 2017

A. Summary of Results

The evaluation metric used in this Kaggle competition is categorisation accuracy, defined as the percentage of correctly recognised digits in the test set. For each of the machine learning techniques applied for this assignment, their respective in-sample accuracy (categorisation accuracy on training set) and test accuracy (categorisation accuracy on test set) are summarised in **Table 1**.

Machine Learning Method	Training Time	In-Sample Accuracy	Test Accuracy
SVM - Linear Kernel - One-Against-All Multiclass	17s	93.355%	91.600%
SVM - Linear Kernel - One- Against-One Multiclass	2mins 47s	97.724%	93.229%
SVM - RBF Kernel - One- Against-One Multiclass	5mins 11s	94.050%	93.600%
SVM - Reduced Dimensions: PCA (First 20 Principal Components) - RBF Kernel - One- Against-One Multiclass	18s	98.876%	97.586%
SVM - Reduced Dimensions: PCA (First 20 Principal Components) - RBF Kernel - Penalty parameter *C* selected via 10-Fold Cross Validation - One- Against-One Multiclass	18mins 11s	99.771%	97.829%
MLP - Two Hidden Layers - Dropout Rate: 50%	10mins	98.450%	97.443%
Convolutional Neural Network - Two Convolution Layers, Two Max Pooling Layers & Two Hidden Layers - Dropout Rate: 50%	37mins 12s	99.940%	99.571%
Convolutional Neural Network - Two Convolution Layers, Two Max Pooling Layers & Two Hidden Layers - Dropout Rate: 50% - Convolve and shift training images to create more training data	1hour 53mins 52s	98.820%	99.657%

Table 1 Prediction Accuracy of Machine Learning Methods

Convolutional Neural Network has an impressive test accuracy of 99.657%, surpassing all other techniques by a comfortable margin.

B. Introduction

The objective of this assignment is to explore the use of machine learning techniques in classifying MNIST (Modified National Institute of Standards and Technology) dataset of handwritten digits. A number of techniques are applied and their performances are summarised in **Chapter A**. This report discusses the applied techniques in details and explains why certain techniques work in this context.

MNIST Handwritten Digits Data

In the MNIST data, each handwritten image is represented using 28 pixels by 28 pixels image ($28 \times 28 = 784$ dimensions). Each pixel contains a value between 0 and 255 (inclusive), indicating the intensity of the greyscale image. The Kaggle dataset provides 42000 training samples and 28000 test samples. Some sample images of the handwritten digits are displayed in **Table 2**.

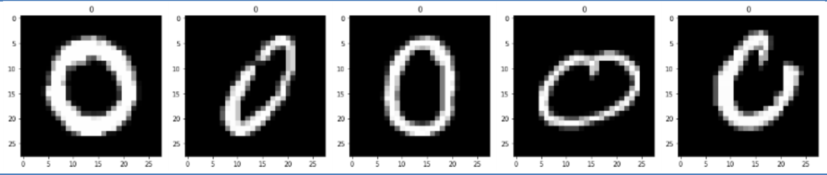
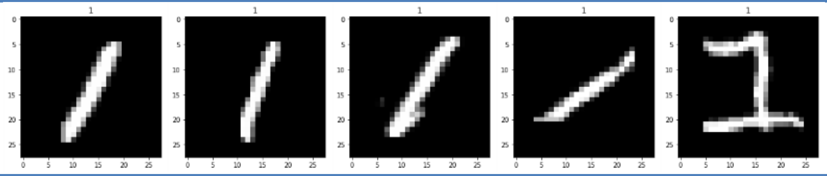
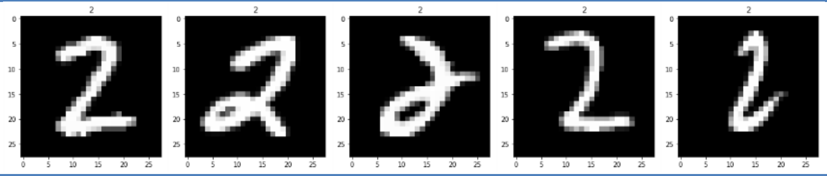
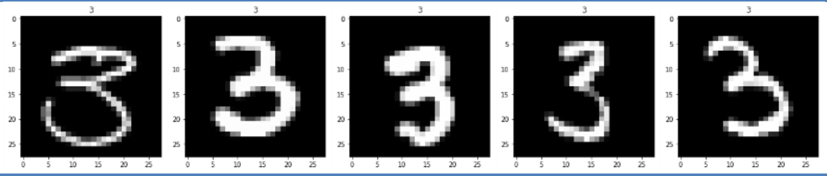
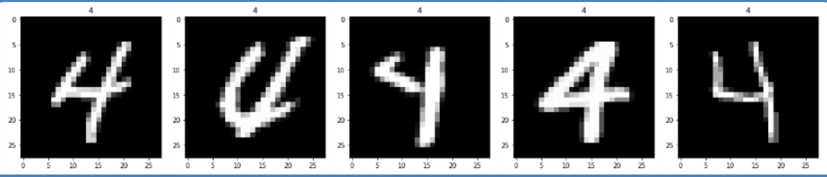
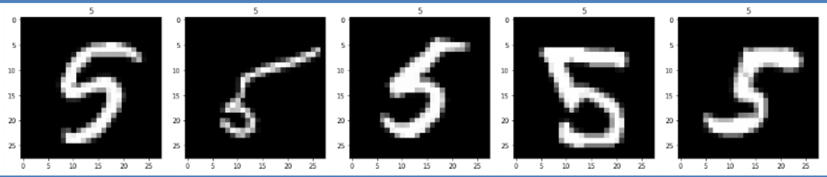
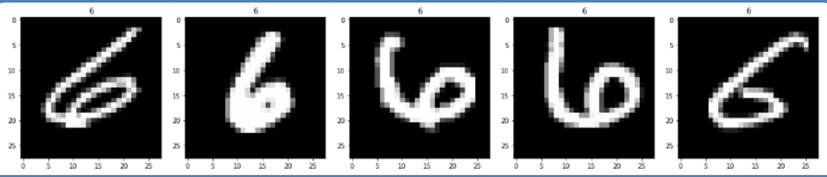
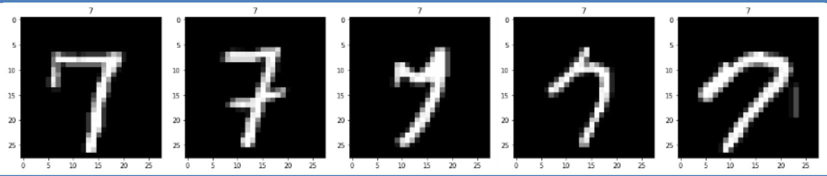
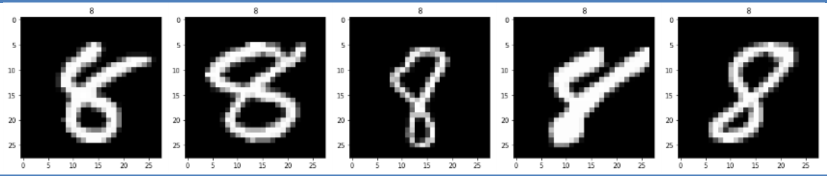
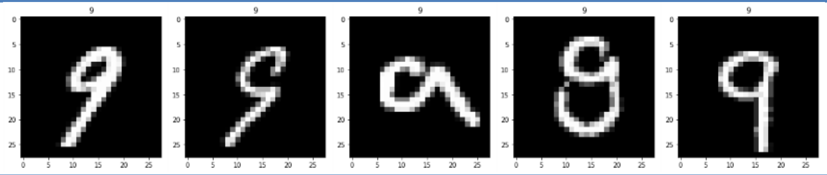
Digit Class	Five Sample Images from Training Set
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	

Table 2 Five Sample Images for each Digit Class

From the above table, it can be seen that each image has slightly different style. For example, some writers wrote the digit in different angles, causing each handwritten digit to be slightly rotated (in different orientation). The same digits may differ slightly in terms of font thickness. Some of the handwritten 8s are not fully enclosed. The same digit class might also be represented using slightly different shapes, such as 1, 7 and 9.

All these handwriting style differences pose difficulties for handwriting recognition algorithms to correctly classify handwritten digits with high confidence level. This report will discuss how the applied machine learning techniques address these difficulties.

C. Data Pre-Processing and Environment Setup

Data Preprocessing

As observed from the sample images in **Table 2**, the MNIST image data is largely pre-cleaned and standardised into a 28px by 28px frame. Nevertheless, there is one preprocessing step required: scale normalisation.

In the MNIST image data, each pixel is a dimension and contains value between 0 and 255. Certain pixels may be frequently used (e.g. pixels at the middle of the image) and consist of a much larger range of intensity values than the seldom used pixels (e.g. pixels close to the borders of the image). If these pixel values are directly fed into Euclidean-distance based machine learning algorithms (e.g. SVM with RBF kernel) without normalisation, the pixels with the largest range will dominate the objective function. Therefore, it is crucial that the scale is normalised before the actual learning. In this case, the pixel intensities are divided by the maximum possible value 255 so that the final values are between 0 and 1 (both ends inclusive). The pixel intensity values are now more comparable.

List of Python Libraries Used

Open-source Python libraries, which implement various machine learning techniques, have been used for classifier training and prediction. The list of Python libraries used and the techniques they provided are tabulated in **Table 3**:

Library	Machine Learning Methods
Scikit-Learn	- Principal Component Analysis - Support Vector Machines
Tensorflow, Keras	- Artificial Neural Networks - Convolutional Neural Networks

Table 3 Python Libraries and Machine Learning Methods

D. Support Vector Machines Classifier

Support Vector Machines Overview

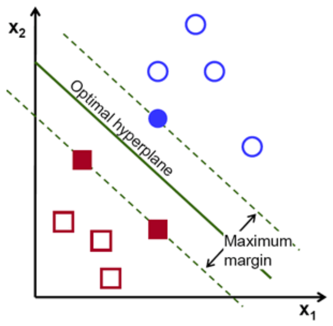


Figure 1 Maximum-Margin SVM (Open Source Computer Vision Library, 2017)

Support Vector Machine (SVM) is a machine learning method originally used for binary classification problem. As illustrated in **Figure 1** on the left, given a set of training samples with two classes, SVM fits an optimal hyperplane (i.e. decision boundary) which separates the two classes. The hyperplane is constructed such that the margin (defined as the distance between the hyperplane and its parallel vector which runs through the nearest training data point of either class). These points that sit on the margins are called the support vectors. While fitting the hyperplane, only the dot products between the support vectors matter in the objective function.

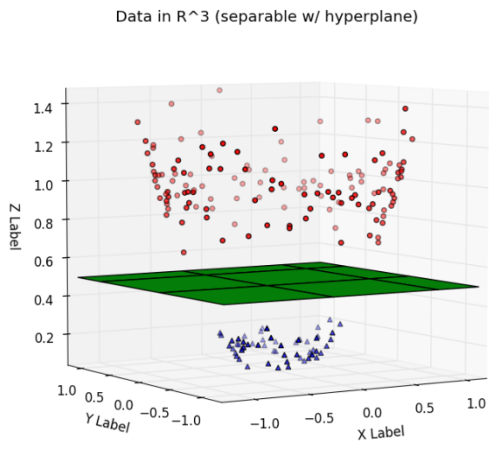
Certain datasets may not be completely linearly separable by hyperplane. To account for these misclassifications, slack variables can be introduced and the samples that sit inside the margin are allowed to be incorrectly classified. These misclassifications are accounted for in the objective function which the SVM would seek to minimise, as expressed below (primal form before Lagrangian relaxation):

$$minimise\ \frac{1}{2}\ \boldsymbol{w}^T\boldsymbol{w} + \frac{C}{2}\sum_n (\xi^n)^2 \qquad subject\ to\quad y^n\ (\boldsymbol{w}^T\ \boldsymbol{x}^n + b)\ \geq 1 - \ \xi^n,\quad n = 1, \dots, N$$

Equation 1 SVM Objective Function (Primal Form before Lagrangian Relaxation)

In the above equation, C is a penalty parameter represents how much of the misclassification errors are weighed (relative to the margin size maximisation) in the objective function. The optimal value of C is highly dataset dependent and depends on whether the dataset contains many "noisy" observations that may be difficult to separate. Its value needs to be empirically determined in order to reach a near-optimal value.

SVM can also be extended to perform nonlinear classification, via the use of kernel functions. In **Figure 2** below, the dataset is visualised on a two-dimensional graph on the right. The dataset exhibits nonlinear pattern where the two classes are not linearly separable in two-dimensional space. Nonlinear kernel functions implicitly maps the features into higher dimensions. As seen from the graph on the left, the two classes are linearly separable using hyperplane after being mapped to three-dimensional space.



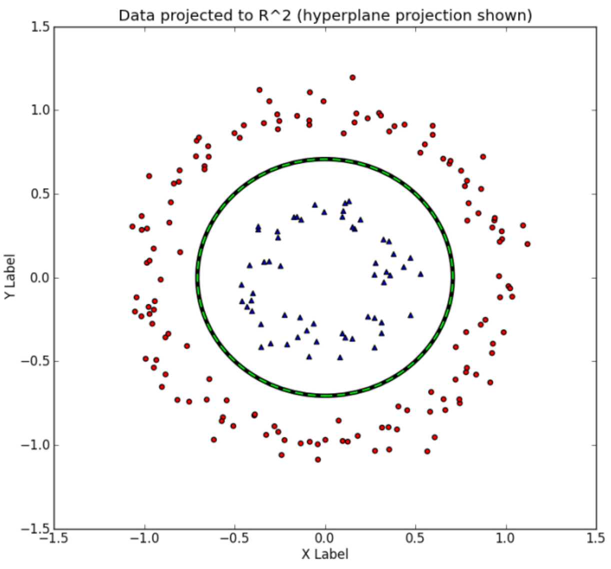


Figure 2 Nonlinear Kernel Separates the 2D Data (Kim, 2013)

Multiclass SVM

SVM is naturally used for binary classification problem. MNIST digit recognition problem is a multiclass classification problem and SVM will need to be extended for multiclass separation. Two extension techniques: one-against-all and one-against-one multiclass SVMs can be used.

In **Figure 3**, the graph on the left illustrates one-against-all method. In this scenario, three SVM models are trained to classify each of the three classes (for example, one of the SVM is "Apple vs Non-Apple"). The thin grey lines in the graph depict the decision boundaries of these 3 SVMs. Using these boundaries, certain regions will be accepted by more than one SVM or rejected by all SVMs. Hence, the better way is to use the continuous values of SVM decision functions for classification. Given a data point, the class which has the highest value of SVM decision functions, will be used as the predicted class. This results in the thick black decision boundaries drawn in the graph.

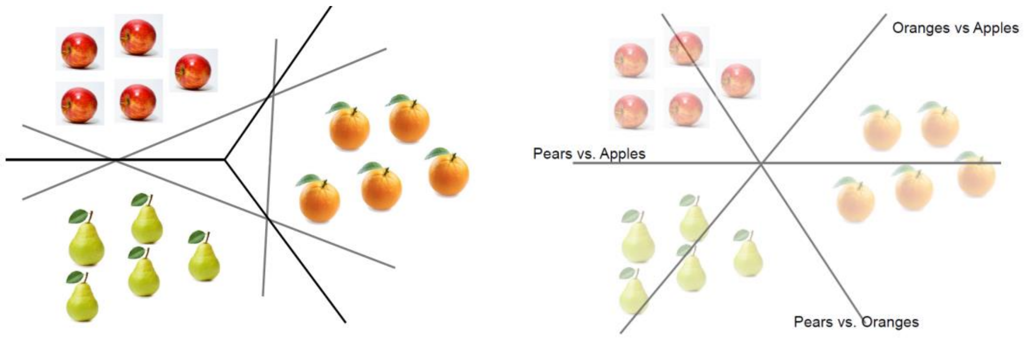


Figure 3 Left: One-Against-All SVM, Right: One-Against-One SVM (McIlwraith, 2017)

The one-against-one method is illustrated on the right graph. For each pair of classes (e.g. Orange vs Apple), a SVM is trained to classify between these two classes. For n number of classes, $\frac{n!}{(n-2)!2!}$ number of SVMs will be required. The decision boundaries of each of these pairwise SVMs are drawn as grey lines in the graph. Given a data point, the class which receives the highest number of votes from the trained SVMs, will be used as the predicted class. Due to the number of SVMs to be trained, one-against-one is usually slower than one-against-all. However, it could potentially yield higher prediction accuracy due to its sophistication (the predicted class will be validated by all pairwise SVM).

E. Support Vector Machines Empirical Results on MNIST

As discussed, SVM can be a good candidate for classification problem. Therefore, SVM is applied to MNIST dataset and this section discusses the predictive performance of different SVM variations.

SVM with Linear Kernel (One-Against-All Multiclass Method)

For a start, a simple SVM with linear kernel is used. For linear kernel, the kernel function is the dot product of two vectors and there is no explicit mapping to higher-dimensional space. In other words, SVM tries to fit a separating hyperplane in the same dimension. Therefore, this method is good for classes that can be decently separated linearly with the use of hyperplane. Other than the kernel function, the multiclass method needs to be specified for the SVM learner as well. One-against-all method is used and the penalty parameter C is set as the default value, 1.

Configuration	Value
Multiclass Method	One Against All
Kernel	Linear
Penalty Parameter C	1.0 (Default value of scikit-learn)
Training Time	17 seconds
Accuracy	• In-sample: 93.355% • Test set: 91.600%

Table 4 SVM Configurations and Results (One-Against-All Multiclass Method)

The MNIST image data has 784-dimensions and are trained via SVM learner. From **Table 4**, it can be seen that this classifier achieves in-sample accuracy of 93.355% and test accuracy of 91.60%. Hence, a separating hyperplane in the high-dimensional space is able to linearly separate most of the classes. In order to enhance its performance, another multiclass method (i.e. one-against-one) can be used. As discussed, one-against-one method takes longer to train but typically produces better classification performance.

SVM with Linear Kernel (One-Against-One Multiclass Method)

The second method to be used here is the linear SVM with one-against-one multiclass method. This method requires separate SVM to be trained for every pair of classes. As there are ten different classes, the number of SVMs required is $\frac{10!}{8!2!} = 45$.

Configuration	Value
Multiclass Method	One Against One
Kernel	Linear
Penalty Parameter C	1.0 (Default value of scikit-learn)
Training Time	2 minutes 47 seconds
Accuracy	• In-sample: 97.724% • Test set: 93.229%

Table 5 SVM Configurations and Results (One-Against-One Multiclass Method)

From **Table 5**, it can be seen that this classifier achieves in-sample accuracy of 97.724% and test accuracy of 93.229%. Due to the sheer number of SVMs to be trained and used for predictions, this method is significantly slower than one-against-all multiclass method. However, there is a significant improvement of its categorisation accuracy (1.629% improvements over the test set).

So far, only linear kernel has been used for the SVM classifier. Since linear kernel is not able to appropriately separate classes that exhibits nonlinear patterns, it is expected that SVM with nonlinear kernel functions would better fit the data and achieve higher predictive performance.

SVM with RBF Kernel (One-Against-One Multiclass Method)

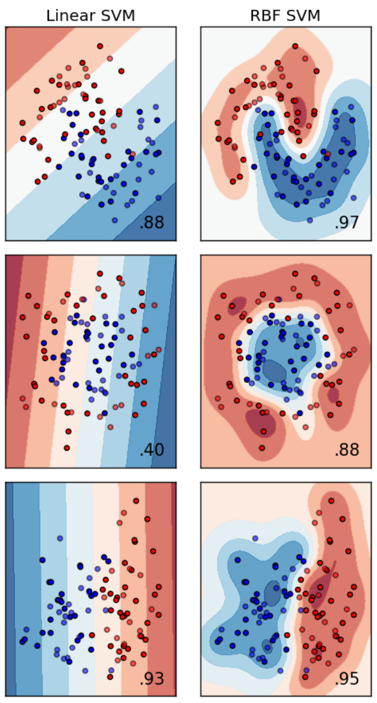


Figure 4 Linear SVM vs RBF SVM (Scikit-Learn Developers, 2016)

Figure 4 compares linear SVM model fit against RBF SVM model fit, for three different two-dimensional datasets). As seen from the two graphs in the middle, linear SVM is not able to decently separate the classes that exhibit nonlinear behaviours while RBF SVM does a better job in constructing a nonlinear decision boundary. It is therefore expected that the nonlinear kernel would improve the model fit if the ten classes in MNIST image data demonstrate strong nonlinear patterns in the 784 features.

The kernel function to be used here is the Radial Basis Function (RBF) kernel. In equation form, it is:

$$K(\boldsymbol{x}, \boldsymbol{x}') = e^{\left(-\gamma \|x - x'\|^2\right)}$$

Equation 2 RBF Kernel

In the above equation $\|\boldsymbol{x} - \boldsymbol{x}'\|^2$ is the squared Euclidean distance between the two vectors ($\boldsymbol{x}$ and $\boldsymbol{x}'$), while γ is a parameter that controls the spread of the kernel. The larger γ is, the faster the kernel function value decreases with increasing Euclidean distance between the two vectors. In training against the MNIST dataset, the scikit-learn default value of γ (= 1) is used.

The SVM classifier (with RBF kernel) achieves 94.05% in-sample accuracy and 93.60% in test accuracy. The predictive performance over the test set is slightly improved, indicating there is some nonlinear behaviour in the data and this has been picked up by RBF kernel. In the subsequent sections, SVM classifier with RBF Kernel (One-Against-One Multiclass Method) will be used since it is able to generalise to the test set better.

Configuration	Value
Multiclass Method	One Against One
Kernel	RBF Kernel
Penalty Parameter C	1.0 (Default value of scikit-learn)
Training Time	5 minutes 11 seconds
Accuracy	• In-sample: 94.050% • Test set: 93.600%

Table 6 SVM Configurations and Results (RBF Kernel)

F. Dimension Reduction: Principal Component Analysis

Many of the real-world datasets are able to be sufficiently represented in lower dimensions. In the MNIST example, it can be seen from **Table 2** that many pixels have no variations. For example, the pixels close to the borders are almost always dark. Consequently, the distinctive features of different digits may be sufficiently captured in lower-dimensional space.

Principal Component Analysis (PCA) technique can be used for dimension reduction purpose. PCA could be applied to MNIST data (in order to reduce the number of dimensions) before SVM is trained. By narrowing down to a smaller number of quality features, it may have a desirable effect to the classification accuracy because it reduces the effects of noise within the data. These noises typically have small variations and are effectively removed when dimension reduction is performed.

Principal Component Analysis Overview

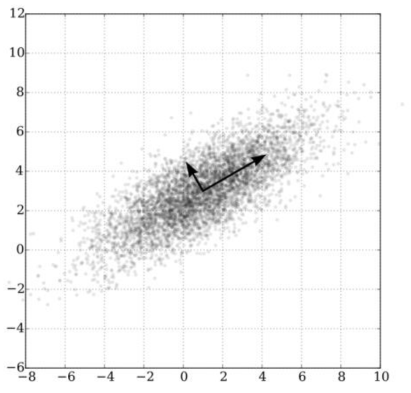


Figure 5 Principal Components (Nicoguaro, 2016)

PCA is an unsupervised machine learning technique which transforms the features of a multivariate dataset into a set of orthogonal components (also called principal components). This transformation is performed such that the first principal component has the largest variance (i.e. explains the most variations in data). Each subsequent principal component has the next highest possible variance and is orthogonal to the preceding principal component. **Figure 5** depicts a two-dimensional dataset; the longer arrow is the first principal component while the shorter arrow corresponds to the second principal component.

Each principal component is represented using m dimensional unit vector while m is the number of dimensions in the original data. The principal components could be solved by performing eigenvalue decomposition of the data covariance matrix.

For the purpose of dimension reduction, the first d principal components are used to project the data into the lower d dimensions ($d < m$). A data point is mapped by performing dot product with each of the principal components. The final d numbers of mapped values are the new feature values. Although this is a lossy compression, PCA preserves the highest possible amount of variance given a lower dimension d, with each new feature being orthogonal to one another.

Example: MNIST Dimension Reduction (to 2 Dimensions) using PCA

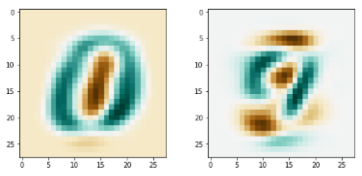


Figure 6 1st Principal Component (Left), 2nd Principal Component (Right)

To illustrate how PCA can be used to improve the prediction accuracy through an example, PCA is applied to the MNIST training data to reduce them to two dimensions. The two images in **Figure 6** represent the first and second principal components.

Blue-green colour indicates that a particular pixel is given positive weight in the principal component. In other words, image which has high intensity pixels in this region will tend to have positive value for its first principal component projection. On the other hand, brown colour indicates negative weights of the pixels. Image which has high intensity pixels in this region will have negative value its first principal component projection. From the left graph, the first principal component differentiates between the shapes of digit 0 (circular shape around the centre) and digit 1 (slightly slanted vertical line at the centre of the image). From the right graph, it can be deduced that the second principal component differentiates between the shapes of digit 2 and digit 9.

Figure 7 shows the scatterplot of MNIST training data points after they have been mapped to two dimensions using the first 2 principal components (X-axis is the first principal component projection while Y-axis refers to the projection by second principal component). PCA projections have decently separated some of the classes to some degrees. It is particularly effective in differentiating between digit 0 and digit 1, and also between digit 2 and digit 9. This agrees with the earlier deductions derived from **Figure 6**.

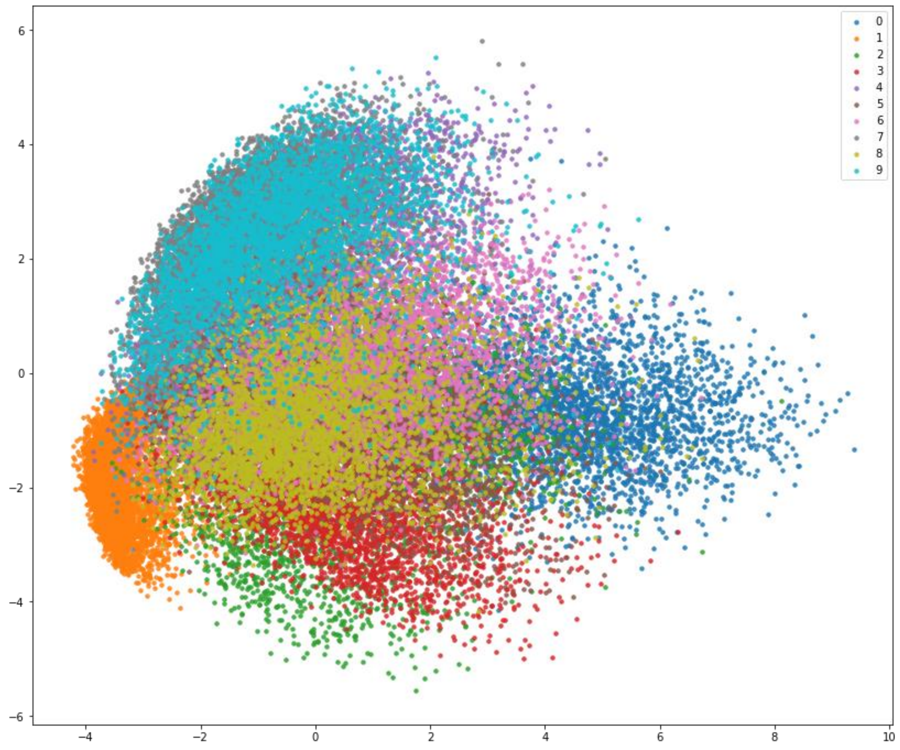


Figure 7 Two-Dimensional PCA Projections of MNIST Training Samples

Using the new data with reduced dimensions, multiclass SVM can be performed to separate the classes. Recall that one-against-one requires pairwise SVMs to be constructed for each pair of classes. **Figure 8** and **Figure 9** show two examples of the pairwise SVMs: SVM to separate digit 0 and 1, and SVM to separate digit 2 and 9.

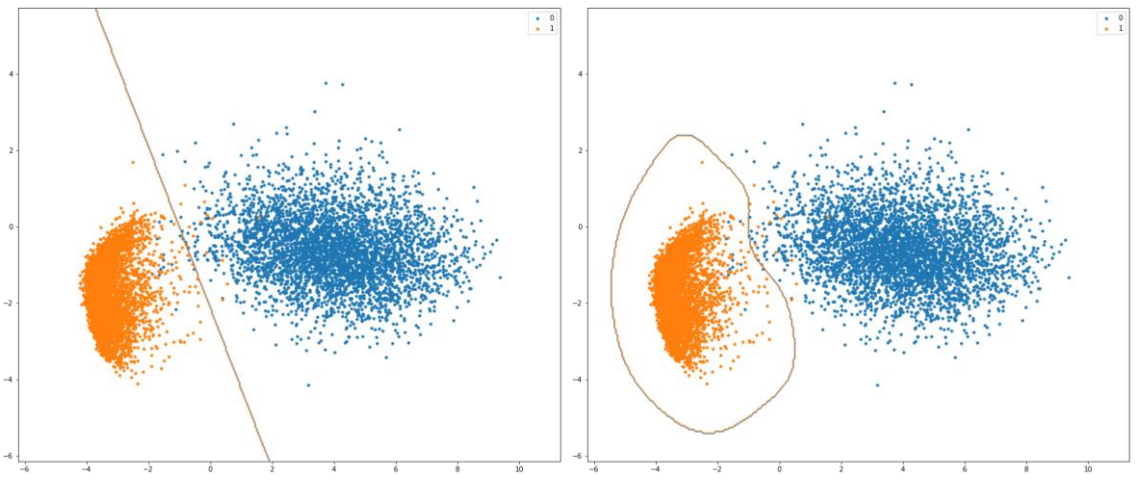


Figure 8 SVM Separating Digit 0 and Digit 1 – Linear Kernel (Left), RBF Kernel (Right)

As observed from the plots, the PCA projections have helped to decently separate the classes in two-dimensional space and an accurate separating hyperplane can hence be constructed. It can also be observed that RBF kernel provides higher flexibility in constructing the decision boundary and correctly classifies more training samples. This agrees with our results where RBF kernel achieves a better performance on test set.

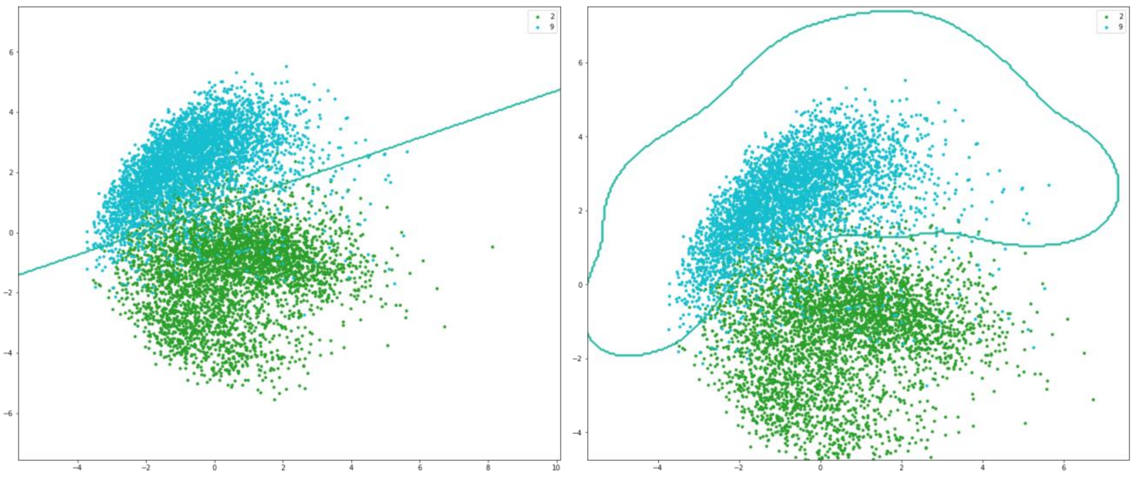


Figure 9 SVM Separating Digit 2 and Digit 9 – Linear Kernel (Left), RBF Kernel (Right)

In summary, PCA creates quality features which can be used by the predictive methods (e.g. SVM) to separate the classes. In this example, the dataset is reduced to two dimensions. PCA can also be used for projection onto dimensions higher than two. In the empirical results section, the MNIST dataset is reduced to twenty dimensions and the predictive performance of SVM will be shown.

G. PCA (20-Dimensions) & SVM Empirical Results on MNIST

As discussed, PCA can help to reduce the number of dimensions to a manageable number of quality features which help to separate classes. It is therefore expected the PCA could help to improve the predictive performance of SVM if sufficient dimensions are used. In this assignment, PCA has been applied to reduce the number of MNIST features to twenty dimensions.

In addition, the penalty parameter C seen in **Equation 1** will also be fine-tuned using 10-Fold cross validation. This is expected to improve the predictive performance since the parameter is now appropriately selected for MNIST dataset. The subsequent subsections discuss the performance achieved using these techniques.

SVM with Dimension Reduction using PCA (First 20 Principal Components)

The motivation behind using PCA to reduce the dimensions in MNIST data is to weed out uninformative features (i.e. the pixels which have very small variations, and likely to be attributed to 'noise' that are not useful in separating digits) and extract quality information. For this purpose, PCA is applied in reducing the data down to 20 dimensions before SVM is trained and this could potentially improve the classification performance.

Configuration	Value
Multiclass Method	One Against One
Kernel	RBF Kernel
Penalty Parameter C	1.0 (Default value of scikit-learn)
Dimension Reduction	20 Dimensions (using PCA)
Training Time	18 seconds
Accuracy	• In-sample: 98.876% • Test set: 97.586%

Table 7 SVM Configurations and Results (with PCA dimension reduction)

Not only does this combined technique significantly reduce the training time, it also improves the accuracy metric by a large margin (in-sample accuracy improves to 98.876% while test accuracy is now 97.586%). It is verified that PCA does preserve quality features and improve the classification performance of SVM.

Fine-Tune Penalty Parameter C using 10-Fold Cross Validation

As discussed in **Equation 1**, the penalty parameter C represents the trade-off between maximising the SVM margin and minimising the misclassification errors. The optimal value of C is dataset dependent. For the MNIST training, 10-fold cross validation is run to select the best value of C from a set of values {1.0, 2.0, 3.0, 4.0, 5.0}.

Configuration	Value
Multiclass Method	One Against One
Kernel	RBF Kernel
Penalty Parameter C	4.0 (selected by 10-fold cross validation)
Dimension Reduction	20 Dimensions (using PCA)
Training Time	18 minutes 11 seconds
Accuracy	• In-sample: 99.771% • Test set: 97.829%

Table 8 SVM Configurations and Results (with penalty parameter C fine-tuned)

The best value for the training set is 4.0. Ten-fold cross validation lengthens the training time but it achieves higher accuracy (in-sample: 99.771%, test accuracy: 97.829%). It is hence shown that the penalty parameter C of SVM has to be empirically validated to obtain optimal results.

H. Feedforward Artificial Neural Network

Artificial neural network is another machine learning model. A neural network consists of a large number of neurons connected in layers. Each neuron has learnable weights (to weigh the input values to the neuron) and biases. These are adjusted in the training phase so that the neural network produces the correct prediction given the inputs. To learn the weights using data, backpropagation is used to propagate the errors backwards through the network so that the error contribution of each node can be computed. Gradient descent algorithm is then applied to fine-tune the weights iteratively. Gradient descent algorithm moves towards the optimum objective value iteratively with the step size being proportional to the learning rate. The learning rate must not be too large (or else it might never reach an optima) or too small (or else it might take too long to reach an optima).

Neural network can learn complex features from data and typically excel in areas such as computer vision. It is to be expected to further enhance the digit recognition accuracy. The two network models (Multilayer perceptron and Convolutional Neural Network) to be applied are both feedforward neural network where the connections between neurons do not form a cycle.

Multilayer Perceptron Overview

Multilayer Perceptron (MLP) is a supervised learning method based on feedforward neural network model. It consists of input layer, one or more hidden layers, and one output layer. **Figure 10** shows the architecture of a two hidden layer MLP. Note that each neuron is fully connected to all the neurons in the previous layer and a bias (not shown in the diagram). Input layer refers to the input features of the data. Hidden layers are the layers that learn the complex features from data. The neurons at the output layer represent the prediction outputs. In the case of MNIST digit recognition (multi-class classification), the output layer consists of ten neurons, each output the predicted probability of belonging to a class.

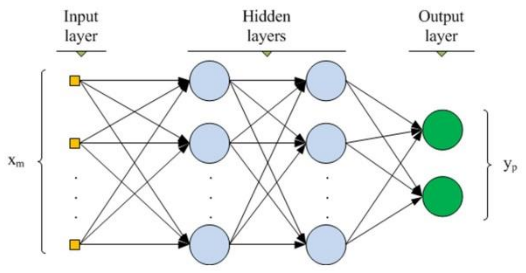


Figure 10 Example MLP Architecture with two Hidden Layers (Caparrini, 2017)

Except for the input layer, neurons in all other layers can have nonlinear activation function. Activation function defines the output function given the weighted sum of inputs at each node. One type of activation function is the sigmoid function. It is a monotonically increasing function and also differentiable, which allows the neural network to learn adjusting weights using backpropagation. However, sigmoid function also suffers from vanishing gradient problem. Sigmoid function has a gradient that is always less than or equal to 0.25 and backpropagation computes the rate of change of error given a change in weight using chain rule. Consequently, the rate of change of error decreases exponentially when the number of layers increases, making it difficult to train deep neural networks. Rectified Linear Unit (ReLU), which does not suffer from this problem, has been proposed as the alternative activation function. Therefore, ReLU is used in this assignment as the activation function of the hidden layers.

Since MNIST is a multiclass classification problem, the output nodes should represent the probabilities of each class given a sample. Thus, the activation function used at the output layer is softmax function. Softmax function normalises the output of each output node to real values between 0 and 1 (both values inclusive) and all the normalised outputs add up to one. Each of these outputs represents the probability of belonging to a particular class.

To prevent overfitting and improve generalisation performance of the neural network, dropout can be applied to the hidden layers so that randomly selected neurons are ignored during training. This temporarily removes the effects of the ignored neurons and forces the network to be less sensitive to specific set of neurons (Brownlee, 2016). This is believed to have a positive effect on the network's generalisation performance.

Convolutional Neural Network Overview

Convolutional Neural Network (CNN) is very similar to MLP. The only difference is that CNN has more specialised layers that are designed to extract features out of an image. The two layers that will be used in this assignment are convolutional layer and pooling layer. An architecture example of CNN is depicted in **Figure 11**.

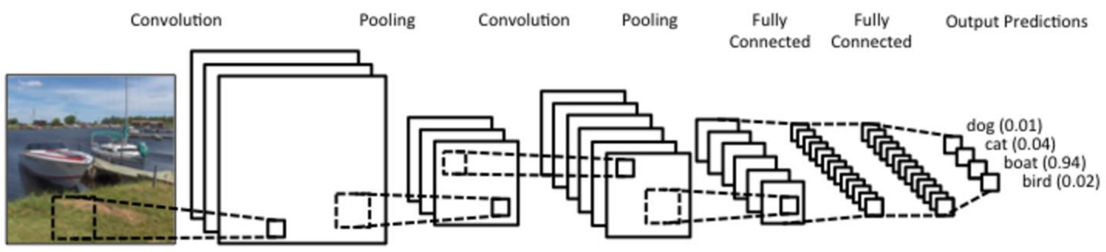


Figure 11 An example CNN Architecture for Multiclass Classification (Britz, 2015)

Convolutional layer consists of a set of learnable filters which focuses at a small region of the input image. These learnable filters extend throughout the whole image. In the region, the dot product between the filter entries and input values are computed. The filter is then slid through the entire image to produce a two-dimensional activation map of that filter. This activation map represents the presence of certain feature in the small region inspected.

Next, the pooling layer outputs a value for the each sub-region of the activation map. Maximum pooling has been found to work better in practice (Wikipedia, 2017) and hence it is used in this assignment (which output the maximum value in the sub-region of the activation map). The reason behind pooling is that it detects the presence of a feature within the sub-region, regardless of its precise location within that sub-region. This is because for images, the exact location of a feature is usually not as important as its rough location relative to other features. Max pooling helps to provide this invariance and also reduce the spatial size of the image representation (which in turn controls overfitting).

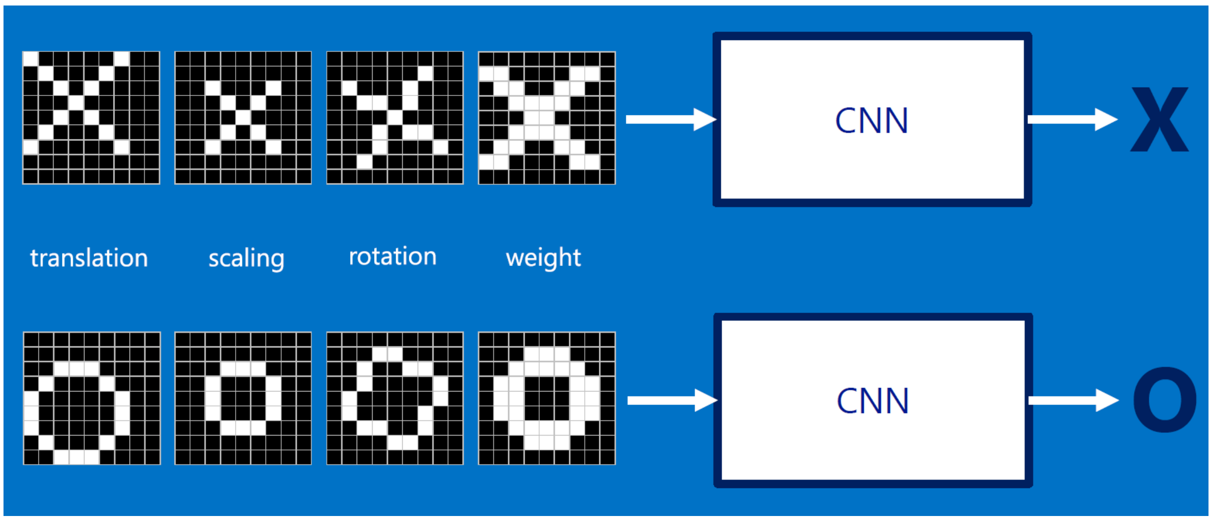


Figure 12 Possible Slight Differences for the Same Image Class (Rohrer, 2016)

As shown in **Figure 12**, image representations of the same class might differ slightly in terms of translation, scaling, rotation and weight (thickness) of the handwriting. By employing convolution and pooling layer, CNN provide a way to detect the same features in image in spite of these slight differences. It is hence expected that CNN would enhance the digit recognition performance.

I. MLP and CNN Empirical Results on MNIST

Since artificial neural network is able to learn complex features, the whole image data (784 pixel intensities) will all be fed into neural network for training (instead of using the reduced dimensions selected by PCA in previous section). The neurons will then learn the features that exist in the image.

Performance of MLP

The MLP architecture, configurations and results are tabulated in **Table 9**. Recall that each output is the estimated probability of each class, hence our loss function (for training) has to relate to that. A suitable loss function is categorical cross-entropy. In equation form, it is:

$$H_{y'}(y) = -\sum_{i=0}^{9} y_i' \log(y_i)$$

Equation 3 Categorical Cross-Entropy Loss Function

y is the predicted probability distribution and y' is the true distribution of digit class (in one-hot vector form). To improve the computational efficiency, minibatch training is used where a small batch of training samples are used to estimate the gradients for subsequent adjustment of weights. Learning rate is chosen to be sufficiently small so that a near-optimal value can be reached within 500 epochs (epochs is the number of full training cycle, where in each cycle all the training samples are exhausted for training purpose).

Configuration	Value
Network Architecture	<u>Input Layer</u> • Dimension: 28 by 28 pixels
	<u>Hidden Layer</u> • 128 Neurons • ReLU activation function • Dropout Rate: 50%
	<u>Hidden Layer</u> • 128 Neurons • ReLU activation function • Dropout Rate: 50%
	<u>Output Layer</u> • Output 10 probabilities for 10 classes • Softmax activation function
Loss Function	Categorical Cross-Entropy
Minibatch size	500
Epochs (Training Cycles)	500
Learning Rate	0.0001
Training Time	10 minutes
Accuracy	• In-sample: 98.450% • Test set: 97.443%

Table 9 MLP Configurations and Results

The predictive performance of MLP is very close to the performance achieved by SVM (with PCA dimension reduction). This is not surprising because MLP is designed to learn the nonlinear relationships between inputs and outputs. SVM is also able to learn this nonlinear relationship with the use of nonlinear kernel. Since the two approaches are similar, MLP is not expected to have large performance differences compare with SVM.

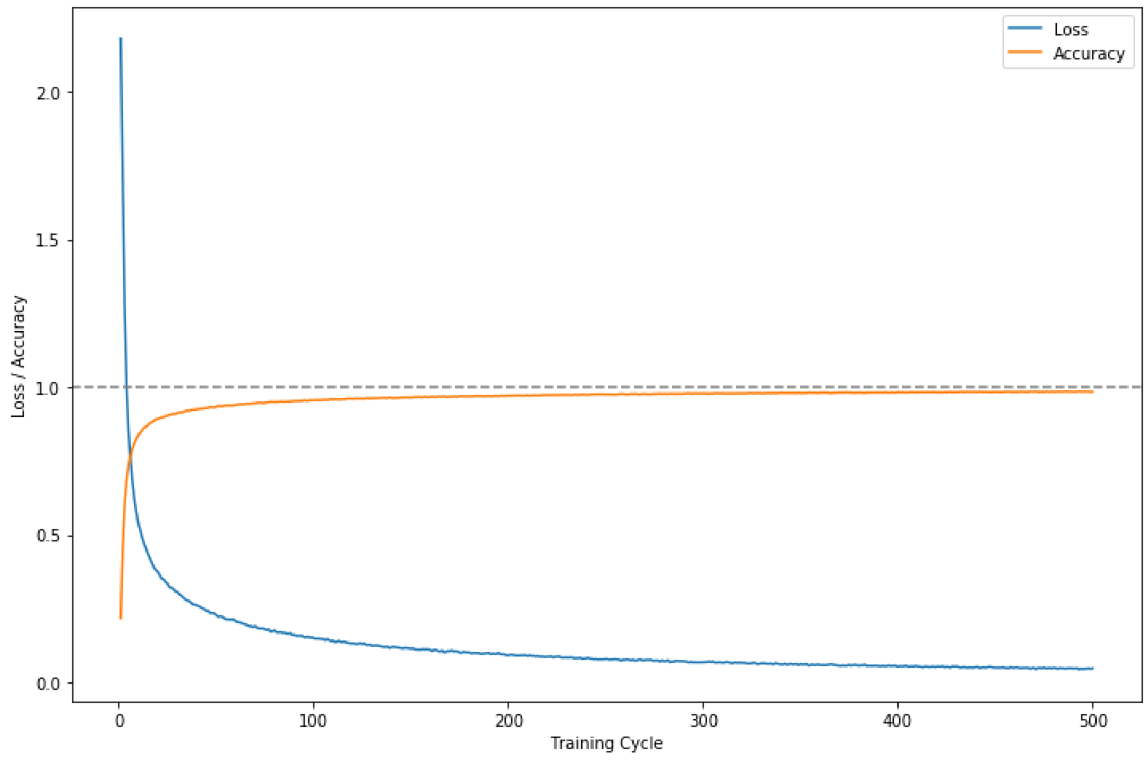


Figure 13 Improvements of In-Sample Accuracies over Training Cycles (MLP)

From the above graph, it can be seen that the in-sample accuracy improves rapidly within the first ten training cycles and then it slows down dramatically. The loss function value also decreases very slowly after the 100th cycle. Evidently, the MLP model may not be optimal in solving digit recognition problem. This could be the reason that in-sample accuracy did not become close the optimal value despite given a high number of training cycles.

On the other hand, CNN provides innovative ways to detect features despite slight differences in terms of translation, scaling, rotation, and weight. Therefore, CNN is expected to increase the classification accuracy.

Performance of CNN

The CNN architecture used is detailed in **Table 10** and other configurations (e.g. minibatch size, learning rate) remain the same. Two convolution layers and two pooling layer are included to learn low-level and high-level image features from data.

Configuration	Value
Network Architecture	Input Layer • Dimension: 28 by 28 pixels
	2D Convolution Layer • 32 learnable filters • 5px by 5px convolution window • ReLU activation function
	2D Max Pooling Layer • 2px by 2px pooling window
	2D Convolution Layer • 64 learnable filters • 5px by 5px convolution window • ReLU activation function
	2D Max Pooling Layer

	• 2px by 2px pooling window • Dropout Rate: 50%
	Hidden Layer • 512 Neurons • ReLU activation function • Dropout Rate: 50%
	Hidden Layer • 256 Neurons • ReLU activation function • Dropout Rate: 50%
	Output Layer • Output 10 probabilities for 10 classes • Softmax activation function
Loss Function	Categorical Cross-Entropy
Minibatch size	500
Epochs (Training Cycles)	500
Learning Rate	0.0001
Training Time	37 minutes 12 seconds
Accuracy	• In-sample: 99.940% • Test set: 99.571%

Table 10 CNN Configurations and Results

As expected, CNN requires longer training time (due to the overheads of convolutional and pooling layers) but it has achieved a remarkable accuracy (99.571% for test set accuracy). The image features learned via convolution and pooling layers helps CNN to correctly categorise those images which were incorrectly classified by SVM and MLP.

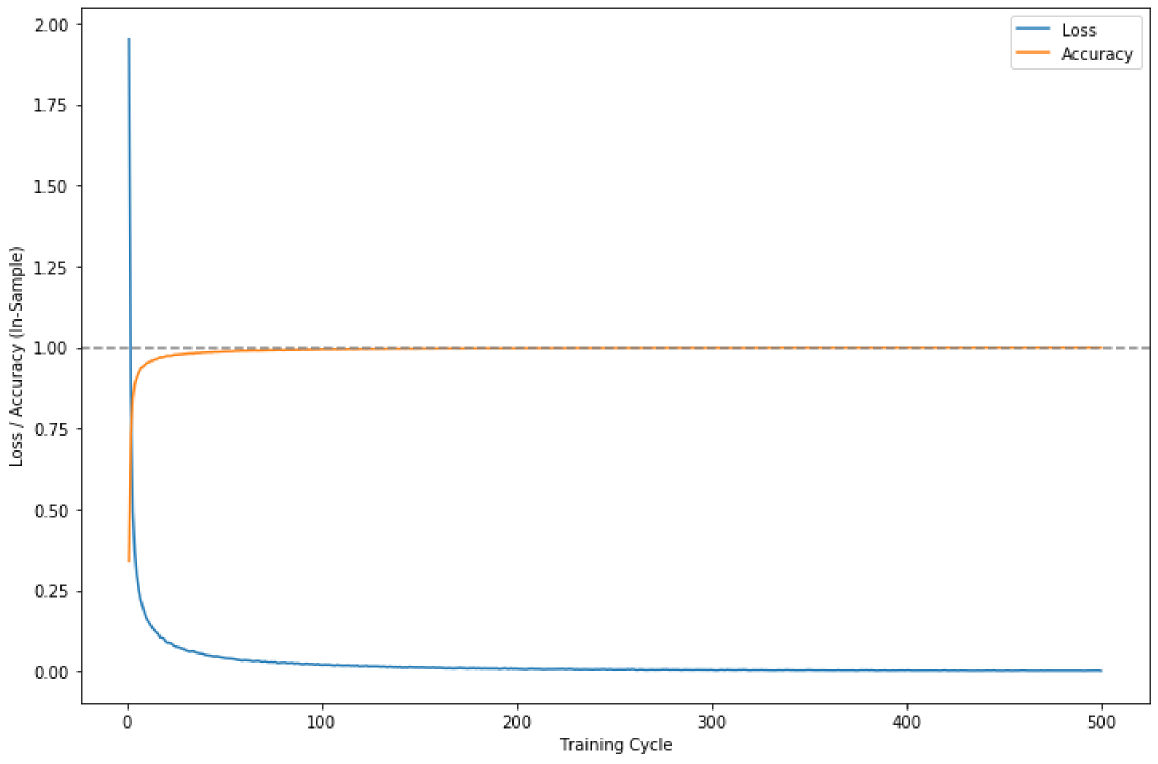


Figure 14 Improvements of In-Sample Accuracies over Training Cycles (CNN)

Figure 14 also exhibits slightly different pattern compared with the graph for MLP. The in-sample accuracy increases much more rapidly and has far exceeded 0.9 within the first ten cycles. It also achieves in-sample accuracy very close to the theoretical optimum of 1.0 within the first hundred cycles. Beyond the 100th cycle, the loss has become so small that the iterative improvement is extremely small.

CNN has achieved a respectable performance in MNIST handwritten digit recognition. To further enhance the predictive performance, a special trick can be employed to create more training samples given the limited training data.

Performance of CNN with More Training Samples Created

In **Table 2**, it has been observed that handwritten digits from the same class can have subtle differences due to the writer's unique handwriting style, for example angle of writing. This fact can be used to generate more "unique" training samples. New training data can be created by slightly rotating or shifting the digits. By specifying the allowable degree of rotation and the maximum allowable shifts, five new example images are generated using Keras package and displayed in **Table 11**.

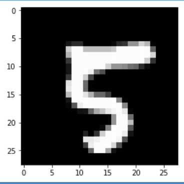
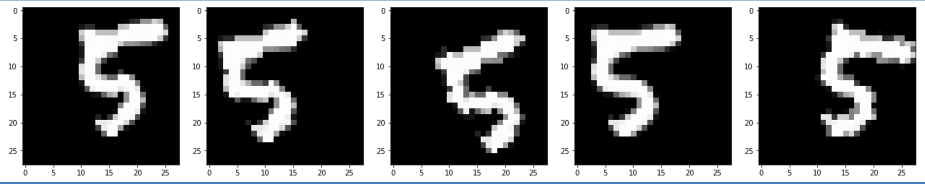
Original	New Images Created
	

Table 11 Original Image and New Images Created

These few images can still be clearly recognised as digit 5 even though there are some difference in its rotation and position. Although CNN is invariant to slight difference in rotation and translation around the original image, creating new training samples allows greater flexibility and forces CNN to learn the generic features which can lead to better generalisation performance. For this assignment, one additional image will be created per training sample. All these data are then fed into CNN for training purpose.

Configuration	Value
Network Architecture	Input Layer • Dimension: 28 by 28 pixels
	2D Convolution Layer • 32 learnable filters • 5px by 5px convolution window • ReLU activation function
	2D Max Pooling Layer • 2px by 2px pooling window
	2D Convolution Layer • 64 learnable filters • 5px by 5px convolution window • ReLU activation function
	2D Max Pooling Layer • 2px by 2px pooling window • Dropout Rate: 50%
	Hidden Layer • 512 Neurons • ReLU activation function • Dropout Rate: 50%
	Hidden Layer • 256 Neurons • ReLU activation function • Dropout Rate: 50%

	Output Layer • Output 10 probabilities for 10 classes • Softmax activation function
Loss Function	Categorical Cross-Entropy
Minibatch size	500
Epochs (Training Cycles)	500
Learning Rate	0.0001
Additional Training Samples?	Generated by slightly rotating and shifting the image, total number of training samples is $42000 \times 2 = 84000$
Training Time	1 hour 53 minutes 52 seconds
Accuracy	• In-sample: 98.820% • Test set: 99.657%

Table 12 CNN (with More Training Samples Created) Configurations and Results

This training set enrichment comes at a cost of longer training time since the amount of training samples is now doubled. However, the test set accuracy has been further improved to 99.657%, which is an impressive number.

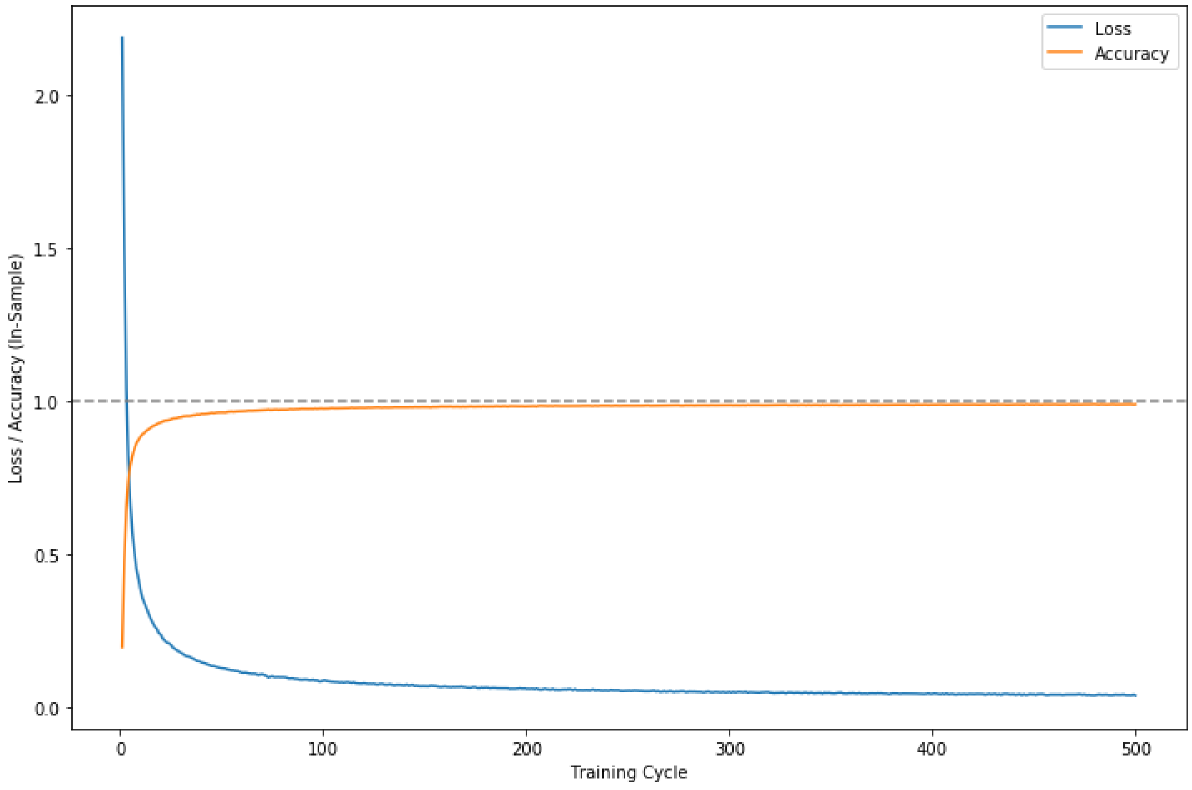


Figure 15 Improvements of In-Sample Accuracies over Training Cycles (CNN with More Training Samples Created)

From **Figure 15**, the curve almost behaves in the similar way as **Figure 14**. The in-sample accuracy increases very rapidly in the first twenty training cycles. The in-sample accuracy did not get that close to theoretical limit of 1.0 this time round (reaching 98.820% after 500 training cycles). This is due to the increased number of training samples. It is simply more difficult to correctly classify almost all of the training images since the number has doubled.

J. Conclusion and Future Works

This report has documented the performances of various machine learning techniques on MNIST digit recognition problem. Although traditional methods such as SVM (with PCA dimension reduction) is able to achieve good classification accuracy (close to 98% on Kaggle test set), CNN is able to correctly classify the handwritten digits with extremely high accuracy. By creatively creating more training samples by convolving the existing training images, CNN has achieved a remarkable test accuracy of 99.657%. This score is within the top five percent of Kaggle submissions.

Dropout, a relatively new idea in regularising neural networks, has been used to enhance the performance of CNN in this assignment. Wan et al (2013) have proposed a generalisation of Dropout (named DropConnect) to regularise large neural networks. DropConnect performs regularisation by randomly selecting a subset of weights to ignore in training phase, and Wan et al (2013) shows that it is able to achieve better categorisation accuracy. This could be a possible direction for future works in further enhancing the performance of MNIST digit recognition.

References

Open Source Computer Vision Library. (2017) *Introduction to Support Vector Machines.* Available from: http://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html [Accessed 28th April 2017]

Kim, Eric. (2013) *Everything You Wanted to Know about the Kernel Trick.* Available from: http://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html [Accessed 28th April 2017]

McIlwraith, Douglas. (2017) *M3: Support Vector Machines.* [Lecture] Imperial College London, 9th March 2017.

Scikit-Learn Developers. (2016) *Classifier comparison.* Available from: http://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html [Accessed 28th April 2017]

Nicoguaro. (2016) *Principal component analysis.* Available from: https://en.wikipedia.org/wiki/Principal_component_analysis [Accessed 28th April 2017]

Caparrini, Fernando S. (2017) *Artificial Neural Networks in NetLogo*. Available from: http://www.cs.us.es/~fsancho/?e=135 [Accessed 29th April 2017]

Brownlee, Jason. (2016) *Dropout Regularization in Deep Learning Models With Keras*. Available from: http://machinelearningmastery.com/dropout-regularization-deep-learning-models-keras/ [Accessed 29th April 2017]

Britz, Denny. (2015) *Understanding Convolutional Neural Networks for NLP*. Available from: http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/ [Accessed 29th April 2017]

Wikipedia. (2017) *Convolutional neural network.* Available from: https://en.wikipedia.org/wiki/Convolutional_neural_network [Accessed 29th April 2017]

Rohrer, Brandon. (2016) *How Convolutional Neural Networks work.* [Video] Available from: https://www.youtube.com/watch?v=FmpDIaiMIeA [Accessed 22nd April 2017]

Wan, L., Zeiler, M., Zhang, S., LeCun, Y. & Fergus, R. (2013) *Regularization of Neural Networks using DropConnect: Proceedings of the 30th International Conference on Machine Learning (ICML-13).* Available from: http://yann.lecun.com/exdb/publis/pdf/wan-icml-13.pdf [Accessed 29th April 2017]