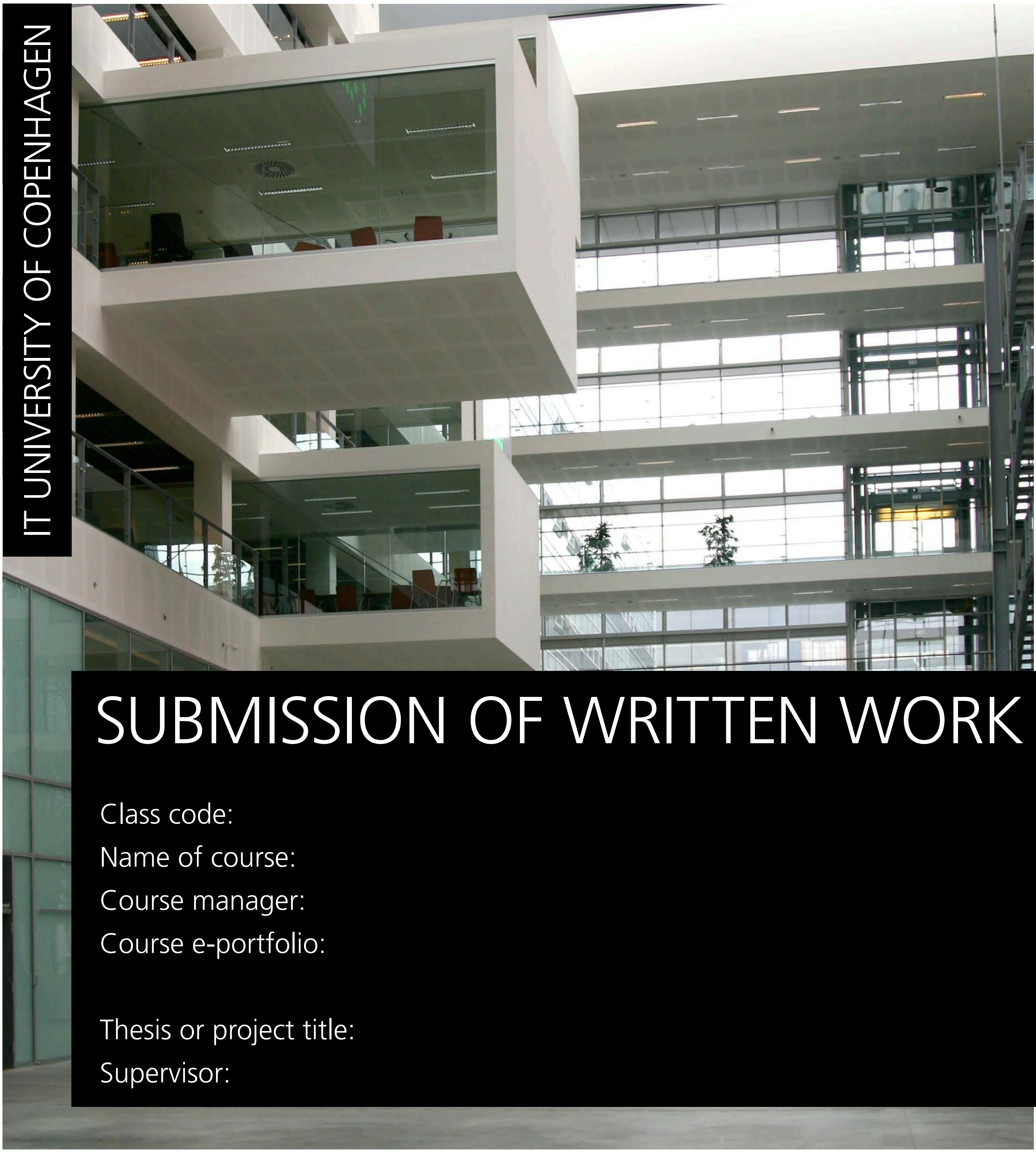
IT UNIVERSITY OF COPENHAGEN



SUBMISSION OF WRITTEN WORK

Class code:

Name of course:

Course manager:

Course e-portfolio:

Thesis or project title:

Supervisor:

Full Name:	Birthdate (dd/mm-yyyy):	E-mail:
1. ______	______	______@itu.dk
2. ______	______	______@itu.dk
3. ______	______	______@itu.dk
4. ______	______	______@itu.dk
5. ______	______	______@itu.dk
6. ______	______	______@itu.dk
7. ______	______	______@itu.dk

HANDWRITTEN DIGIT RECOGNITION

Data Mining, Spring 2015
IT-University of Copenhagen

Project and report by
Mads Nørgaard Anthony
Jeppeh Martin Olsen
Patrikk Dyrberg Sørensen

Supervisor
Sebastian Risi

Table of contents

INTRODUCTION

For this project we have engaged in handwriting recognition, a field in machine learning that has multiple applications. From the digitalization of notes to automating post office letter handling, it can be immensely useful and timesaving if computers automatically and with great certainty can discern the individual elements of handwriting.

In this project we have focused on the recognition of digits, using the famous MNIST database of handwritten digits, that has previously been used in a lot of machine learning projects. We have chosen this data set as it is very big and publicly available.

Our scope in this project is not to achieve a very high amount of precision. Though we find it crucial, that a useful system, should have a precision close to 100%, we do not find it likely that we can compete with previously successful projects. Rather, we explore and apply different data mining techniques, as to see how these will modify our results.

Problem statement

How can different data mining techniques and methods be implemented in order to classify the MINST dataset, and how do they influence the overall precision of the classification?

METHODS

In this project we are utilizing a variety of different methods and techniques. All of the implementation has been done with Matlabs built-in functions.

For preprocessing we are using the Fourier transformation, which we find applicable, as our data consists solely of images.
We use the Principal Component Analysis to reduce the very high dimensional data, into a visualization friendly 2-dimensional data set.
For the classification of the data we use the algorithms K-nearest neighbour (kNN) and an artificial neural network algorithm (ANN). In the choosing of these algorithms, several factors were taken into account. We found that a deep learning ANN approach was favoured by the community at Kaggle.com [web3]. Secondly, we wanted to implement two different classification algorithms, in order to better reflect and compare our results. To validate our results of the classifications, we use cross-validation.

In the following sections we will briefly cover these methods and our specific use of them.

The data set

The data we are using for this project is from the Mixed National Institute of Standards and Technology database (MNIST)[1]. It is a publically available database, containing approximately 70.000 data points. Because of our computational resources we have only been able to work with a portion of the data set, around 2000 samples.

Each sample contains a 28x28 pixel greyscale image of a single handwritten digit from 0-9. This gives in total 784 variables for each data point, as each pixel is used as a variable. The digits have also been normalized according to size and placed in the center of the image [web4].

We have decided to use 85% for the training set and 15% for the test set. To make sure our results and our choice of methods is not biased we will only use the test set, as a final indicator of how good the method performs.

Figure 1 shows the distribution of each number in the training and test set respectively. resembling a somewhat uniform distribution of digits.

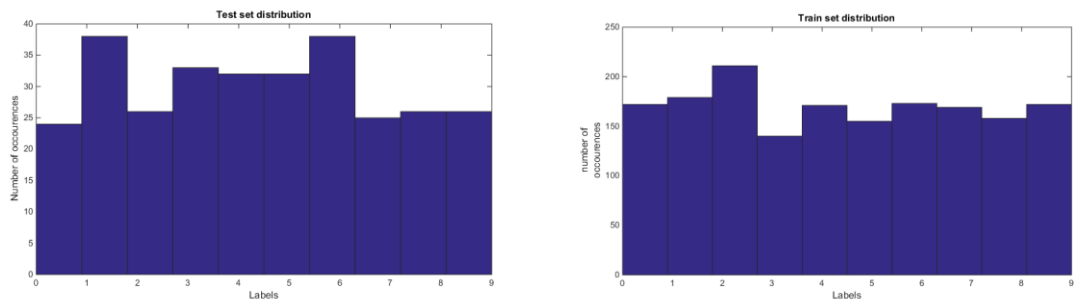


Figure 1: Distribution of numbers in the test and training set, respectively.

Fourier Transformation

Using the raw data might create errors or unexpected results, whenever a number is not centered or have another size than expected. For example if the number ‘3’ was written very small in the top left corner and another ‘3’ is written very big in the center. In order to normalize and prevent such errors from our data set we use the signal processing of Fourier transformation on the data. This preprocessing method will suggest an underlying structure of the images, which could be preferred by the algorithm, rather than using the raw pixels.

[1] The data set can be retrieved at the following website: https://www.kaggle.com/c/digit-recognizer/data

The Fourier transformation is basically a way to describe frequencies. It can be used for image processing by describing a sum of sinusoids by its magnitude and the phase [web1]. However by disregarding the phase and just looking at the magnitude - then the exact same number transposed anywhere on the image, will look the same, since the magnitude remains the same.

See figure 2 to see how the Fourier transformation looks applied to our data.

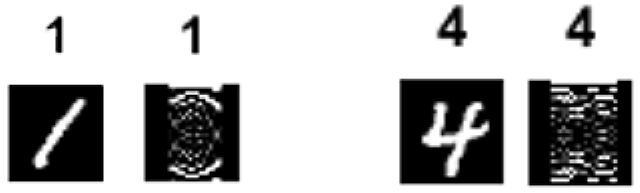


Figure 2: Images of numbers with and without FFT.

Clustering

PCA

Principal Component Analysis (PCA) is an algorithm, that can be used as a method for reducing the number of dimensions in a data set. This is good for making high dimensional data easy to visualize. We have initially used PCA in order to display our 784-dimensional data in two dimensions, in order to determine if patterns can be recognized within the data set. As PCA is quite complicated, we will not go into detail on how it works, but instead give a brief overview.

The PCA can be compared to rotating an object, in order to find the angle, where the contour of the object is most recognizable. In the case of a teapot, this will probably be from the side. So, instead of looking at it from all angles, you can determine the object, by looking at it from a few specific angles.

The PCA will compute k n-dimensional orthogonal vectors, where k is the number of dimensions to output and n is the total number of variables in the data set. In order to create one of these vectors, each variable is given a weight, in order to produce it.
Each vector will therefore contain information from all variables and the results of PCA will emphasize the variance in the data. The first principal component is the vector that produces the biggest variance.

Before performing the PCA on the data set, it is important that the data is normalized. The PCA creates k orthogonal vectors, that are sorted according to its strength. All subsequent vectors will be perpendicular to all previous vectors.

Because of the sorting, the user can remove components with less importance, and thereby the number of dimensions of the data set can be highly reduced, while still keeping the most important information [Han & Kamber: 79-80].

Classification

k-Nearest Neighbor

The kNN algorithm finds the k closest neighbours in a n-dimensional space for a given data point in a given training set. The classification of a data point is the label that has the most occurrences among the k-nearest neighbors. To find the nearest neighbors a measure of distance is needed. In our case we make use of the euclidean distance (figure 3), which is given by the following. Where p = (p_1, p_2,..., p_n) and q = (q_1, q_2,..., q_n) are two points in Euclidian space [web 2].

In our kNN implementation we use let k span from 1 to 10 and and use 5-fold cross validation, in order to find the hyperparameter k.

$$\mathrm{d}(\mathbf{p},\mathbf{q}) = \mathrm{d}(\mathbf{q},\mathbf{p}) = \sqrt{(q_1-p_1)^2 + (q_2-p_2)^2 + \cdots + (q_n-p_n)^2}$$
$$= \sqrt{\sum_{i=1}^{n}(q_i-p_i)^2}.$$

Figure 3: The kNN equation with euclidian distance, where p and q are data points.

ANN

Artificial neural networks offers many possibilities of creating supervised learning in order to make a precise and configurable classifications of different inputs. In our context, the task is to create an algorithm that can classify handwritten digits.

The basic concept of a neural network is to have *n*-connected neurons, each receiving different inputs, and together accomplishing a more complex task. Generally, each neuron recieves one or more weighted inputs, that are summed along with the neurons own bias. The result is given to its activation function, that maps the value of the neuron to a value between 0-1. The result can then by multiplied by a weight and become an input for a neuron in the following layer.
The weights in the network are initially set to a random value. Through training they are updated using backpropagation, in order to improve the precision of the classification. The backpropagation algorithm uses the gradient to find elements that caused the imprecise output, by backtracking through the neural network and adjusting the weights. [Han & Kamber: 277].

The network can be constructed in many different ways and in our case we use a feedforward network with one hidden layer. We use Matlabs built-in library for neural networks and from here we utilize the method patternnet with one single layer and 500 hidden neurons. We tried out using both 50, 250 and 500 hidden neurons, but in the end we got the best results with the latter option. Patternnet uses cross-entropy as a measurement of the error, and will stop the number of iterations when either the magnitude of the gradient becomes too small or by early stopping, to prevent overfitting.

Cross-validation

For validating our results we are using the *k*-fold cross-validation. The data is divided into *k* number of subsets, which gives you the following sets D_1, D_2 … D_k. *k* iterations are now performed, where the algorithms is both trained and tested. In these iterations the data subset D_k is used as the test set and all other subsets are used for training the network. An advantage of this method is that all samples are used once for testing and an equal amount of times for training.

RESULTS

In this section we will present our results by first visualizing our dataset through PCA where we label each digit with a color. Then we visualize the weights of the variables of the first and second principal components, and lastly we present our result tablesfor the accuracy of the kNN and ANN implementation on raw and preprocessed data. These results create a foundation in which we can reflect and discuss on, in the discussion section.

Clustering

By finding the first two principal components, we can visualize our original dataset of 784 dimensions in the two-dimensional space shown in figure 4.

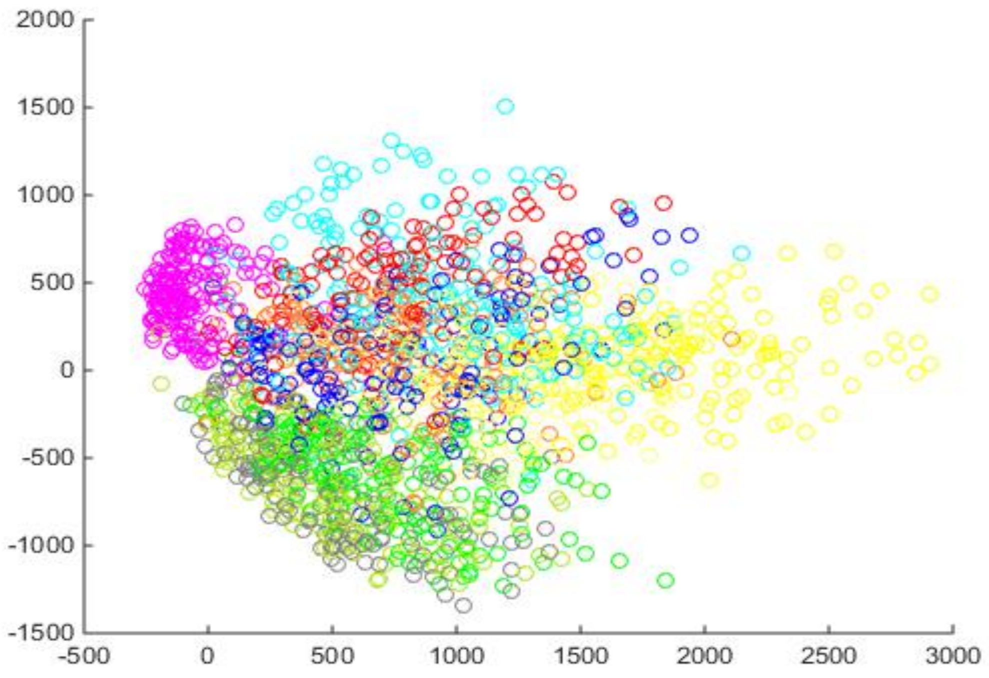


Figure 4: Visualization of the distribution of our data, using only the first two components of our PCA. Each color represents a number.

The horizontal axis represents the first component and the vertical axis represents the second component while the color represents the label. As can be seen in the plot, it does a relatively good job of dividing the samples in the different classes. Even though the classes are not linearly separable by using the first two components, they do cluster up. If we increase the number of dimensions we might be able to separate the data points completely, however we will not be able to visualize it.

To get a better understanding of what the first two principal components represent, we have visualized it by converting the components into the images, that can be seen in figure 5

First principal component
Second principal component

Figure 5: Visualization of the two first components in our PCA. The color of the pixel represents its weight in the principal components, where black is zero.

The whiter a pixel is, the higher is the weight of this variable (or pixel) in the principal component. So the images in figure 5 can be thought of as image masks of the original digits, that defines the most important areas for the algorithm to differentiate between the class labels. Notice how both the first and the second principal components weights the center of the picture. It is here it gets the most variance across the dimensions.

Classification results

KNN results

The results in table 1 shows the precision of the kNN without the use of FFT. It can be seen that the highest precision is achieved when k = 3.

bestK = 3	kNNTrainPrec = 0.9406	kNNTestPrec = 0.9033

Table 1: k-nearest-neighbour results on data without preprocessing.

For the results in table 2, we have the same approach, but this time the data has been preprocessed with the Fourier transformation.
The results in table 2 shows the precision of the kNN *with* the use of FFT. It can be seen that the highest precision here is achieved when k = 4.

bestK = 4	kNNTrainPrec = 0.8965	kNNTestPrec = 0.8367

Table 2: k-nearest-neighbour results on data preprocessed by Fourier transformation.

ANN results

By applying the trained network we get the following results for both the training and test set.

ANNTrainPrec = 0.9612	ANNTestPrec = 0.8567

Table 3: Neural network results without FFT to preprocess the data.

Again, we have the same approach, but this we preprocess the data with FFT and get the results that can be seen in table 4.

aNNTrainPrec = 0.8688	aNNTestPrec = 0.7400

Table 4: Neural network results with FFT to preprocess the data.

Figure 6 shows how our training stops at 43 iterations and chooses the network which was achieved at iteration 37. The training of our network stops early as a way to prevent overfitting. More specifically, when the error of the training set approaches zero, the algorithm will become overfitted, and therefore it will be much more prone to errors. When the error rate increases the specified six times in a row, the training discontinues, as it is not likely it will begin to decrease.

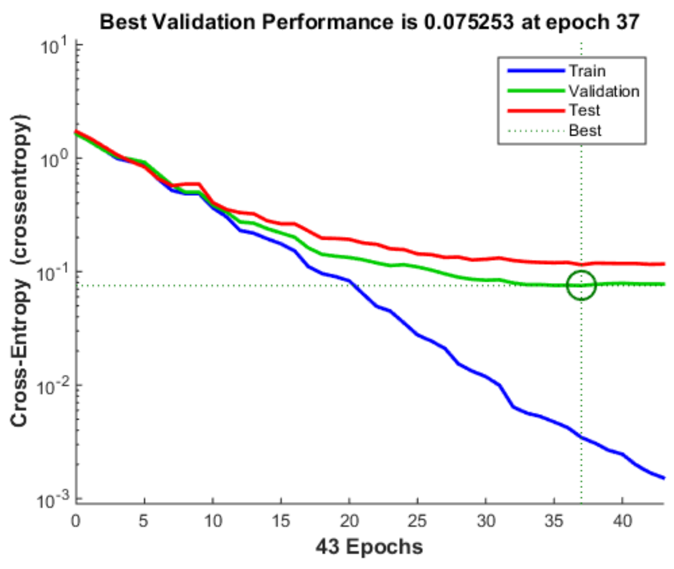


Figure 6: Visualization of the learning curve on our neural network, without Fourier transformation.

'

DISCUSSION

This section discusses the methods we have used and the achieved results through those methods.

Discussion of the results

As was presented in the results section, the kNN algorithm gives us better results, both with and without the Fourier transformation seen in table 5.

	No preprocessing	With FFT	Difference
kNN	90.33%	83.67%	6,66
ANN	85.67%	74.00%	11,67
Difference	4,66	9,67	

Table 5: The precision of the algorithms when performed on the test set.

Our expectations of using Fourier transformation was to achieve a better accuracy when classifying. However, this was not the case with the accuracy dropping by roughly 6-12% for both the aNN and kNN.

It is worth noting that the Fourier transformation still provides us with another way of looking at the data, which we can choose to explore further. For the MNIST data set though, the raw data seems to already have been cleaned and optimized in a way where Fourier transformation is not having a positive effect for the classification results.

Discussion of the methods

ANN

As noted previously, the ANN can be configured in many ways. Our implementation could further try to analyze the Matlab patternnet function, and try to tweak and test certain parameters to improve the accuracy. This includes how the cross-entropy is calculated, the amount of hidden nodes, defining several hidden layers and how we divide the training and test set.

Additionally, other preprocessing techniques could be implemented along with the FFT, including a reduction of pixels and conversion to binary black and white images, where a pixel can be either completely black or white.

CONCLUSION

For this project, we have implemented various data mining techniques which can classifiy images of handwritten digits. To achieve this, we have tried to preproces, cluster and classify the MNIST dataset by using Principle Component Analysis, Fourier transformation, k-Nearest Neighbor algorithm and Matlabs predefined neural network patternnet.

The presented results show, that our kNN implementation achieves an accuracy of 90% while the neural network achieves 85% accuracy on the raw MNIST dataset if we look at the test set. When applied to our preprocessed dataset, the accuracy of those same algorithms decreases. Therefore we recommend not to use the FFT as a means of improving the accuracy of the current implementation of ANN and kNN.

REFERENCES

Han, Jiawei & Kamber, Micheline, *Data Mining - Concepts and Techniques*, second edition, (2006)
web1, http://cns-alumni.bu.edu/~slehar/fourier/fourier.html, seen 13th May, 2015
web2, http://en.wikipedia.org/wiki/Euclidean_distance, seen 14th May, 2015
web3, https://www.kaggle.com/c/digit-recognizer/forums, seen 14th May, 2015
web4, http://yann.lecun.com/exdb/mnist/index.html, seen 14th May, 2015