Advanced Machine Learning: Use of machine learning in Image Recognition.

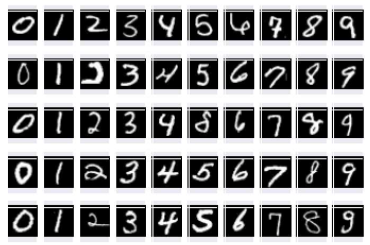
Report produced and submitted by Alexander Romanenko on 1 May 2017.

1. Introduction

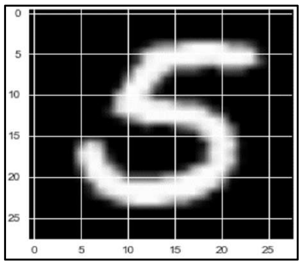
The use of image recognition has expanded enormously in recent years and now is used in many aspects of our lives, ranging from security, commerce, logistics, entertainment and many others. This growth has mainly been driven by two factors: improvement in efficiency and cost of computational power and application of new machine learning techniques. This report will explore some of these techniques in detail.

This report will focus on the use of appropriate machine learning techniques (such as K- Nearest Neighbour, Support Vector Machines, Principal Component Analysis and Convolutional Neural Networks) in order to aid image recognition as part of the 'Digit Recogniser' Kaggle competition. The report will discuss and analyse the used methods, their application for the task and will provide the details of the analysis. Finally, it will propose the potential extension of the analysis in the future work.

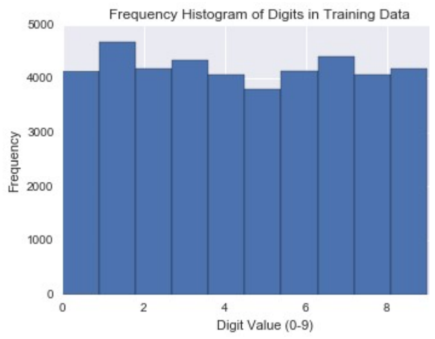
2. Problem Description



The task is to build an algorithm for image recognition, in particular recognition of hand written digits using machine learning approaches. The main challenge comes from the fact that same digits could vary to a considerable extent when written by different people as shown on the Figure shown on the right. Although, arguably the problem of digit recognition should not be a challenge for most people, the issue appears to be that the process might be time-consuming and therefore an ability to teach a computer to recognise digits effectively would make the process much more efficient. We will attempt to do so using MNIST dataset. The aim of the task is to find a technique which would produce a highest accuracy for the image recognition.



The dataset consists of 70,000 examples of handwritten digits between 0 and 9. The dataset is split into training (42,000) and test (28,000) sets. The training set contains correct classification of the digits, whereas the test set will be validated through the Kaggle website where accuracy for every dataset submission will be given.



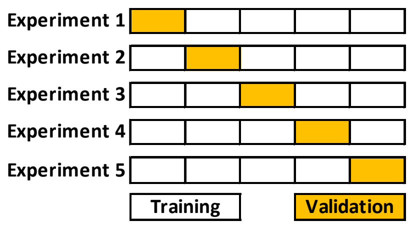
Initial analysis of the data shows that every image is shown using 784 pixels as 28 by 28 matrix. Every pixel is represented by values between 0 and 255 based on the intensity. The distribution of the digits within the training dataset is uniform with similar numbers of examples per every digit, which is optimal in order for the potential models to be trained on features of every digit.

3. Method

A general approach for solving an image recognition challenge could be divided into two parts: feature extraction and classification (Labusch, Barth and Martinetz, 2008). The feature extraction involves the preliminary pre-processing and search for patterns between representative of the same class. Once these patterns are found, the new data is being analysed in order to match its features to the features of the training set. Later in this chapter we will describe these stages of the process for the selected machine learning techniques: K- Nearest Neighbour (KNN), Principal Component Analysis (PCA), Support Vector Machines (SVM) and Convolutional Neural Networks (CNN). The choice of the techniques has been guided by the desire to use a combination of simpler methods (in this case KNN) and more advanced machine learning techniques (such as PCA, SVM and CNN).

For the purposes of the analysis we will use Python libraries, such as *scikit-learn* (for KNN, SVM and PCA) and *keras* (for CNN).

Validation approach. We will use cross validation (CV) in order to tune hyperparameters (where applicable) and/or validate the models. Normally cross validation is performed by splitting the dataset into n number of folds, the model is then being trained on n-1 folds and being tested on the remaining fold. This process is repeated so every single fold is used for validation. The accuracy rate is calculated by finding an average of the accuracy values for every step of CV. Key benefit of cross-validation comparing to standard test/validation split is avoidance of loss of significant modelling or testing capability (En.wikipedia.org, 2017).



Assessment. Once the models are trained and validated, they will be used to classify the test data set. The predictions will then be uploaded to the Kaggle submission page, which will produce the final accuracy result.

3.1 K- Nearest Neighbour

Method description. K- nearest neighbour (KNN) is a non-parametric method for classification and regression. The new data is classified by the class which is most common among its k nearest neighbours. For a classifier where K is an integer and x_0 is a test observation, the KNN method identifies K test data points (represented by N_0), which are closest to x_0. The classifier then estimates the conditional probability of a class *j* :

$$\Pr(Y= j \mid X=x_0) = \frac{1}{K}\sum_{i \in N_0} I\ (y_i = j)$$

Finally, KNN classifies the test point x_0 to the class with the highest probability by applying Bayes rule (James et al., n.d.).

Hyper-parameter tuning: optimal K. K value represents the number of nearest neighbours, which are used in classification exercise and therefore has a drastic effect on the accuracy of the model. Even though the KNN approach is a non-parametric, K should be considered as a hyper-parameter. In order to establish an optimal K, the performance of the KNN method will be assessed on the training set using different values of K. K value that gives the highest training set accuracy will then be used on training data.

Method implementation. Within scikit-learn package we have two different NN classifiers: *KNeighboursClassifier* and *RadiusNeighboursClassifier*. The former has been selected for this analysis as it is preferred for uniformly distributed data. The following parameters have been selected:

Parameter	**Selection**	**Reason/description**
n_neighbours	4	Cross validation stated that k=4 produces the most optimal results
weights	uniform	The data is distributed uniformly

Validation approach. Cross Validation of KNN will be performed using *RandomizedSearchCV* package (which is found within *sklearn*) using the following parameters:

Parameter	**Selection**	**Reason/description**
estimator	KNN	We require cross validation to be performed using KNN
param_distributions	a)k_range: 1:100 b)uniform/ distance	a) accuracy will be assessed for values of k from 1 to 100 b) the algorithm is given a choice to try two approaches: 'uniform' and 'distance' (we expect 'uniform' be selected as an optimal one)
n_iter	10	Number of cv folds is selected to be 10
n_jobs	1	Number of jobs to be performed in parallel

Advantages vs Shortcomings

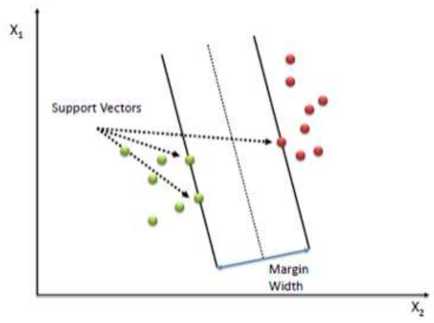
Advantages (Wiesemann, W., 2017):

- Simplicity of the method and high effectiveness
- High efficiency of training the model

Shortcomings (Wiesemann, W., 2017):

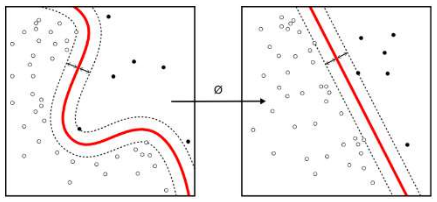
- There is no model to infer relationships between variables
- Slow classification phase- requires determination of k value
- Choice of optimal k could be problematic in larger dimensions with a lot of data

3.2 Support Vector Machines



Method Description. Support Vector Machine (SVM) are supervised machine learning technique for classification and regression analysis. In simple terms, the principle of the approach is to represent the data as points in space and to separate (classify) the data using a linear classifier (also known as a hyperplane). Key features of SVM are as follows:

- The optimal linear classifier is the one with the largest width (i.e. margin) between support vectors of two classes (the figure above adopted from Saedsayad.com, 2017).
- SVM successfully performs non-linear classification using a kernel function, where essentially the data is mapped into a high-dimensional space to enable effective separation of classes (as shown on the Figure to the right. Adopted from Saedsayad.com, 2017).
- For problems where it is impossible to clearly separate the classes, a soft margin SVM approach is used, which allows

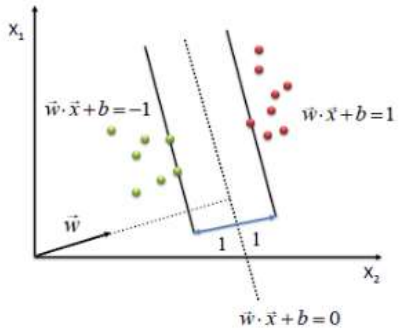


some training data to be misclassified by allowing for constraints violations.

- SVM also able to process classification of C>2 classes by reducing a single multi-class task into multiple binary classification tasks. (En.wikipedia.org, 2017)

SVM algorithm works as follows (Saedsayad.com, 2017):

1) An optimal linear classifier which maximises the margin is defined

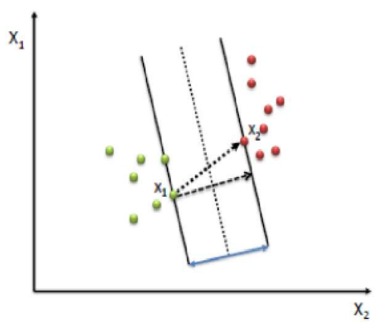

X₁
$\vec{w}\cdot\vec{x}+b=-1$
$\vec{w}\cdot\vec{x}+b=1$
$\vec{w}$
1
1
X₂
$\vec{w}\cdot\vec{x}+b=0$

$$\max \frac{2}{||w||}$$

s.t.

(w*x + b) ≥ 1, ∀x of class 1

(w*x + b) ≤ -1, ∀x of class 2



$\frac{w}{||w||} * (x_2 - x_1)$ = width = $\frac{2}{||w||}$

W * x_2 + b =1; W * x_1 + b = -1

W * x_2 + b - W * x_1 - b=1 – (-1)

W * x_2 - W * x_1= 2

$\frac{w}{||w||}(x_2 - x_1) = \frac{2}{||w||}$

By using quadratic programming, we solve the objective function and end up with the following equation:

$$\min \frac{1}{2}||w||^2 \quad \text{s.t.} \quad y_i(w * x_i + b) \geq 1, \ \forall\, x_i$$

2) Where the problem is a non-linearly separable: we relax constraints and implement a penalty for misclassification. We introduce a slack variable ζ_i to account for misclassification; misclassifications are to be penalised. Our objective function becomes the following:

$$\min \frac{1}{2}||w||^2 + C\sum\zeta_i \quad \text{s.t.} \quad y_i(w * x_i + b) \geq 1 - \zeta_i, \ \forall\, x_i; \ \ \zeta_i \geq 0$$

3) We introduce a high dimensional space where the data is mapped to. We achieve this by using a Kernel function. Two potential forms of this function are shown below:

Polynomial $\quad k(x_i, x_j) = (x_i, x_j)^d$

Gaussian Radial Basis function $\quad k(x_i, x_j) = \exp(-\frac{||x_i - x_j||^2}{2\sigma^2})$

4) We reformulate the problem to account for multi-classification (more than 2 classes). There are two commonly accepted ways to expand binary SVM problem to a multi-class problem (McIlwraith, D., 2017):
    a. One against all (AOA). For every class *c1*, we divide all classes in two groups: class *c1* and the rest *(n-1)*. The problem is then solved using a standard 2-class SVM process for all *n* classes. The class is assigned when one SVM accepts it and all other rejects it. Potential problem with this approach is that some classes might be selected by more than one SVM or none at all.
    b. One against one (OAO). Improvement on AOA method, where for every pair of classes, a separate SVM is created. The class is assigned based on the largest number of classifiers that agree. It is computationally less expensive comparing to AOA

Data Pre-processing. Support Vector Machine algorithm described above is not scale invariant and therefore the input data values will need to be transformed to [0,1] in order to use the algorithm adequately (Scikit-learn.org, 2017). This process is called normalisation and enables accurate comparison of values (Abdi, 2017) The normalisation will have to be applied to both test and training sets (excluding the class labels column). In order to ensure that the values are in the region between 0 and 1, we will divide all values by 255 (maximum value).

Method Implementation. For this problem we use *SVC* classifier, which is a part of scikit-learn package. It is preferred to the *LinearSVC* as the problem is non-linear. The following parameters are used:

Parameter	Selection	Reason/description
Kernel	rbf- Gaussian Radial Basis function	It is commonly more acceptable to use RBF kernel to polynomial (Quora, 2017), especially in cases with high dimensionality.
Multi-class support	One-against-one	As discussed previously, OAO is computationally more efficient comparing to AOA.

Validation Approach. Cross Validation of SVM will be performed using *cross_val_score* function (*sklearn* package) using the default parameters:

Parameter	Selection	Reason/description
estimator	SVC	We require cross validation to be performed using SVM method
cv (number of folds)	10	Number of cv folds is selected to be 10

Advantages vs Shortcomings

Advantages (Wiesemann, W., 2017):

- High Accuracy of predictions
- Kernel trick allows to deal with non-linear problems effectively

Shortcomings (Wiesemann, W., 2017):

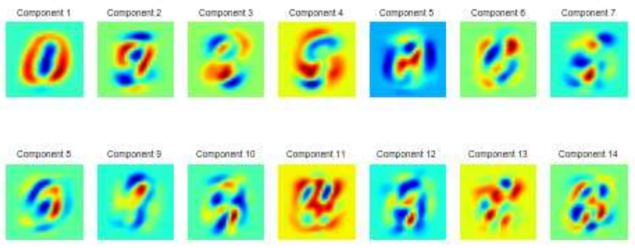
- Hyperparameters (such as optimal kernel choice) could be difficult to determine
- For larger n, the method could be slow to run
- Interpretation of the model results is not straight-forward

3.3 Principal Component Analysis

One of potential drawbacks of using multi-dimensional data in machine learning is a ‘curse of dimensionality’, which states that high number of attributes of data represents a bigger space of possible points, which makes the similarity of these data points harder to determine (McIlwraith, Marmanis and Babenko, 2016). As a result, effectiveness and accuracy of Machine Learning techniques decrease. One of the ways to reduce negative impact of ‘curse of dimensionality’ is to use Principal Component Analysis.

Method Description. Principal Component Analysis (PCA) is an unsupervised approach for producing a low-dimensional set of attributes from a multi-dimensional set of values. This low-dimensional set consists of computed principal components and should in theory still be able to explain most of the variability of the original set (James et al., n.d.). Low dimensionality is more advantageous as it allows for higher processing efficiency and allows us to reduce impact of ‘dimensionality curse’. The aim of PCA is to find and compute a combination of principal elements that produce highest possible variance.

Principal components used in compression of the data could be represented by the figure to the right. The first component represents high compression, i.e. large distance/variance from the original. With increase in number of components, we can see the increase in complexity in order to maximise the variance of the data for this particular component.



Components calculation. The first principal component could be represented by:

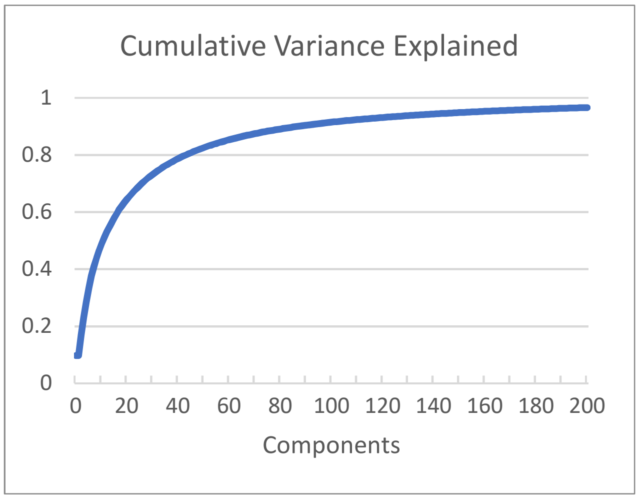
$$Z_1 = \phi_{11}X_1 + \phi_{21}X_2 + \cdots + \phi_{p1}X_p$$

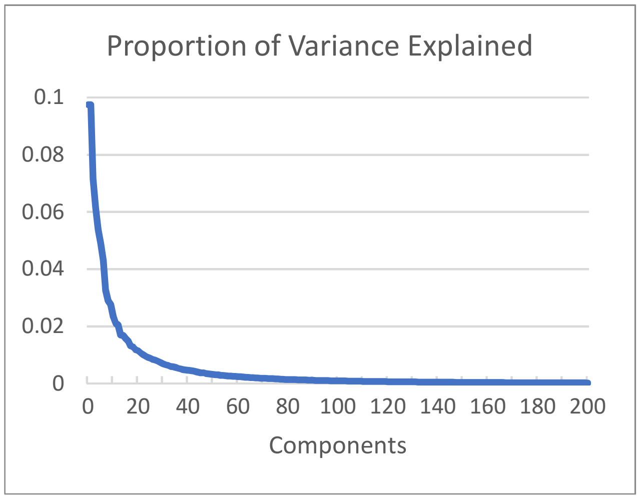
Φ represents ‘loading’ of the component. The sum of these ‘loadings’ must be equal to 1 to avoid situations where they could be artificially increased to a large number and in turn increase the resulting variance. The first principal component is found by solving the following optimisation problem (James et al., n.d.):

$$maximise \left\{ \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{j=1}^{p} \phi_{j1} X_{ij} \right)^2 \right\} s.t. \sum_{j=1}^{p} {\phi_{j1}}^2 = 1$$

The second principal component should have maximal variance of all linear combinations of X_p that are uncorrelated with Z_1. This could be achieved by constraining the direction of ϕ_{j2} to be orthogonal to the direction of ϕ_{j1}, i.e. the variance will be at maximum when using the eigenvector associated with the largest eigenvalue (McIlwraith, D., 2017). Therefore, to find the second principal component we use the same optimisation problem as described above (substituting ϕ_{j1} with ϕ_{j2}) and adding an additional constraint for the orthogonality of the loadings ϕ_{j1} and ϕ_{j2}. This approach is used for all other required components (James et al., n.d.).

Hyper-parameter tuning: Optimal number of principal components. Choice of a number of principal components to use is very important as we want to ensure that we capture sufficient amount of variance of the data and at the same time, we reduce the number of dimensions currently used. One of the potential approaches to use is to look at the variance distribution in relation to number of components:





We can see that with components = 40-50, the total explained variance reaches 80% and every additional component would add only around 0.4% accuracy (and reducing). Therefore, we could assume that optimal number of components to use could lay in the region between 40 and 60. However, as you can see, this approach is not very scientific and depends on our perception of variance importance. Therefore, here we will use an alternative way: we will run cross validation of the training set on n-components from 1 to 100. The n, which produces the highest accuracy of SVM method will then be used on the test set.

Method implementation. The analysis will be performed using following steps:

1) the original dataset (both training and test) will be transformed using PCA method;
2) optimal number of principal components will be found;
3) SVM method (described in section 3.2) will be used using the transformed data.

PCA function (part of sklearn package) will be used here with the following parameters:

Parameter	Selection	Reason/description
N_components	TBC (0-100)	Cross validation will be used to find the optimal n_components that produces the highest accuracy on the training set.

Validation approach. Cross Validation of PCA will be performed using *cross_val_score* function (*sklearn* package) with the following parameters:

Parameter	Selection	Reason/description
estimator	PCA	We require cross validation to be performed using PCA method
cv (number of folds)	10	Number of cv folds is selected to be 10

Advantages vs Shortcomings

Advantages (McIlwraith, D., 2017):

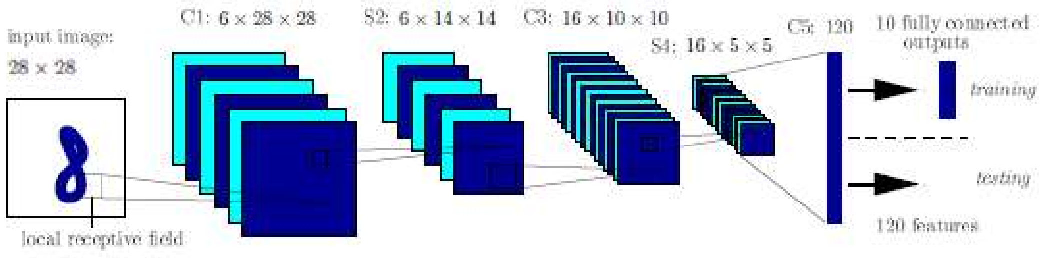
- Ability to dramatically reduce dimensionality and noise of the problem dataset effectively
- Increase in processing speed
- Works really well with image recognition problems

Shortcomings (McIlwraith, D., 2017):

- Choice of optimal number of principal components could be problematic
- Directions with largest variance could be mistakenly assumed to be most important

3.4 Convolutional Neural Network

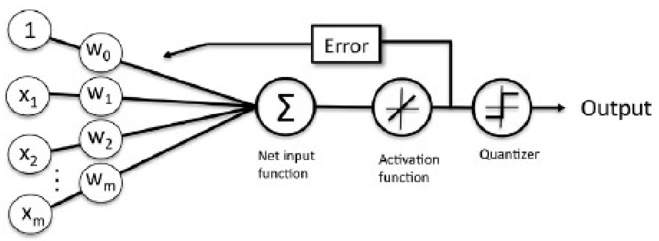
Method Description. A Convolutional Neural Network (CNN) is a feed-forward network, which efficiently extracts features of data. CNN has a particularly strong application in image and video recognition (En.wikipedia.org, 2017). The CNN system is trained similar to a standard neural network by use of backpropagation (see description below). The features of the raw data are normally stored in the networks' first layers, which are usually an alternation of convolutional and subsampling layers. These convolutional layers are used for extraction of basic visual features from local receptive fields. The layers are organised in panes (also known as feature maps). Every pane is assigned trainable weights, which are adjusted when the model is trained through backpropagation. Subsampling layers are responsible for reduction of the spatial resolution of the feature maps. These layers contain same amount of feature maps as the preceding convolutional layer, however the number of rows and columns is reduced by 2 (F. Lauer, C. Y. Suen and G. Bloch, 2007). A Basic example of a CNN is shown on the diagram below (adopted from F. Lauer, C. Y. Suen and G. Bloch, 2007):



C1, C3 and C5 are convolutional layers; S2 and S4 are subsampling layers (F. Lauer, C. Y. Suen and G. Bloch, 2007).

Backpropagation in our case refers to a very computationally efficient process of computing weight coefficients in a multi-layer artificial neural network (the diagram is adopted from Raschka and Olson, 2016).

In order to explain the maths behind the method, we will use an example of a simple neural network in a binary classification problem (Raschka and Olson, 2016). We introduce an activation function ϕ(z), which acts as an identity function of the net input:



$$Z_1 = w_1X_1 + w_2X_2 + \cdots + w_pX_p = w^tX,$$

where w's are weights/coefficients of the model. We also define an objective function J (also known as a cost function), which is calculated by the sum of squared errors (note: there are other loss function measures available) between the calculated outcome and the true class label:

J(w) = $\frac{1}{2}\sum_i (y^i - \phi(z^{(i)}))^2$

The algorithm starts the process by initialising the weights as 0 or a very small random number. For each training x- y is being computed and weights are being updated:

w_j = w_j + Δw

The activation function is used for learning and the process repeats itself until the cost function J is minimised. The quantizer is a step function, which acts as a classifier by using linear combination of inputs to classify the output: 1 if $z \geq \theta$ and -1 otherwise.

Gradient descent method is used to minimise the cost function, where in each iteration we move away from the gradient in order to find the local minima. The step size is determined by the specified learning rate η in addition to the slope of the gradient (Raschka and Olson, 2016):

$$\Delta \mathrm{w} = -\eta\ \Delta \mathrm{J}(\mathrm{w}) = \mu \sum_i (y^i - \phi(z^{(i)}))x_j^i$$

This process is similar to the one we use in the multilayer neural network. Fortunately for us, currently available machine learning packages perform all these operations automatically.

Method implementation. We use Keras: Deep Learning library for this task. Prior to training the model we configure the learning process using the following parameters:

1) An optimiser. We use an optimiser 'Adam' for this problem. Key advantages of it are straightforward implementation, computational efficiency and little memory requirement (Kingma, D., and Ba, J., 2014)
2) A loss function. 'Categorical crossentropy' has been selected as it is preferred for categorical multiclass problems (Datascience.stackexchange.com, 2017).
3) A list of metrics. "Accuracy" has been selected as this is what we are trying to maximise.

Once the learning process is set up, we can set up the network using the following parameters:

Parameter	**Selection**	**Reason/description**
Number of layers	3: 32, 64 and 128 features per layer	This is standard set-up for a CNN
Input_shape	(28,28,1)	This is driven by the specificity of our data where we have images represented as a matrix of 28*28 pixels
Final Dense	10	Represents 10 output classes

The network is now set up, so we can run the model using the following parameters:

Parameter	**Selection**	**Reason/description**
epochs	20	This is standard practice for a CNN
steps per epoch	Total number of entries divided by 5	Division by 10 would have probably been sufficient, however to ensure thorough learning- more steps have been used.
Auto-stop function	Min_delta=0	Whenever a new epoch does not produce any improvement (delta is ≤ 0)- the algorithm will stop.

In order to enhance the feature extraction process and improve accuracy of the produced model we will artificially increase number of training images by using ImageDataGenerator function (found in keras library). The function produced 'new' images for the model to train on by:

- Rotating the original images;
- Zooming in and out (i.e. increasing/decreasing the size of the original images);
- Shifting positions of the original images.

Biggest advantage of this process is that it requires minimum memory space as it works in 'just in time' format.

Validation approach. In order to ensure that the network has been set up correctly and the model is working as expected, a validation of the process will be performed using smaller values for steps per epoch:

Parameter	**Selection**	**Reason/description**
epochs	20	This is standard practice for a CNN
steps per epoch	500	To ensure that the process does not take too long to run
Auto-stop function	Min_delta=0	Whenever a new epoch does not produce any improvement (delta is ≤ 0)- the algorithm will stop.

Advantages vs Shortcomings

Advantages (Lantz, 2013):

- Effective for classification or numeric prediction problems;
- Modelling ability of complex relationships and pattern recognition is arguably more advanced; than most other algorithms;
- Low number of assumptions in relation to the data's underlying features;

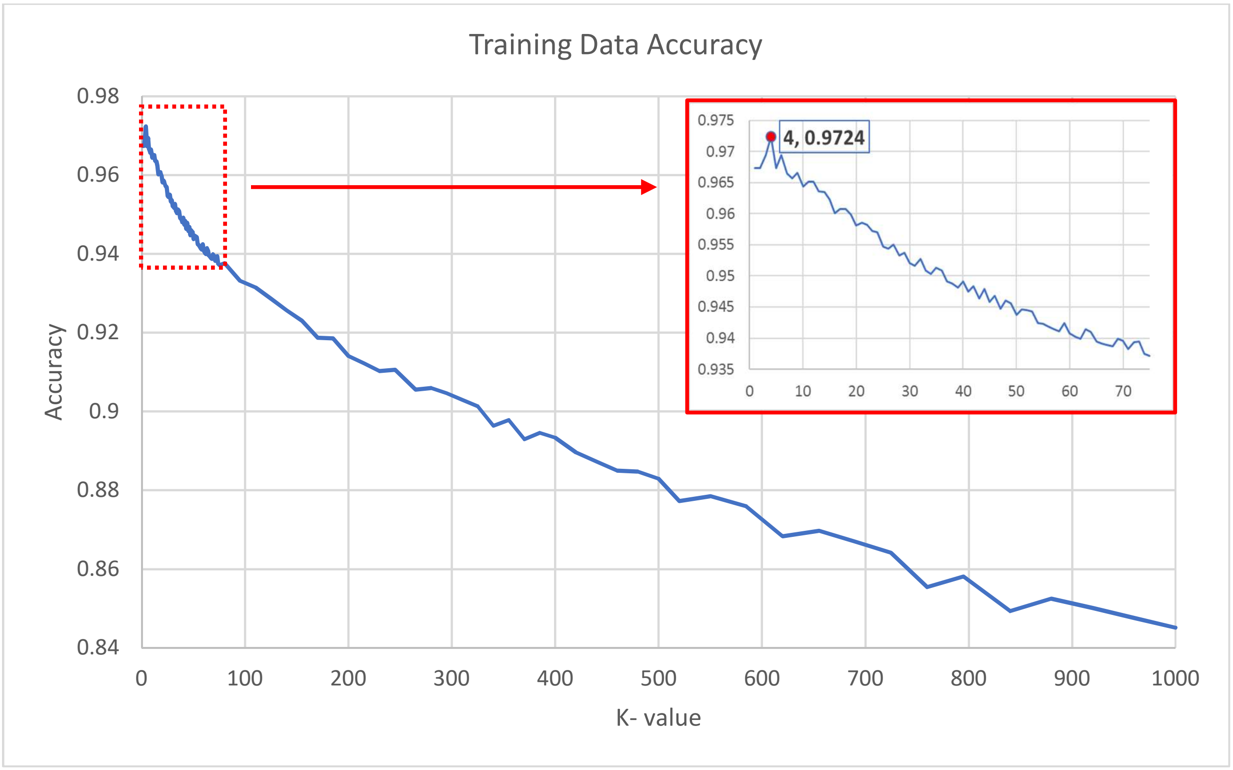
Shortcomings (Lantz, 2013):

- Extremely heavy from processing point of view- could be very slow to train a model;
- Easy to overfit the model;
- Results and the model's features are very difficult (if not impossible) to interpret.

4. Results

4.1 K-Nearest Neighbour

Optimal k. Results of the Cross Validation show that the optimal accuracy of 97.24% for the training set is achieved for k=4. The accuracy reduces down as the k increases, which is not surprising.



Test Set Accuracy. We plug the optimal k in the classifier, which is based on the training data and run the classification for all values in the test set. When uploading the produced results to Kaggle submission page we get 97.186% accuracy. See appendices 8.1 for more details.

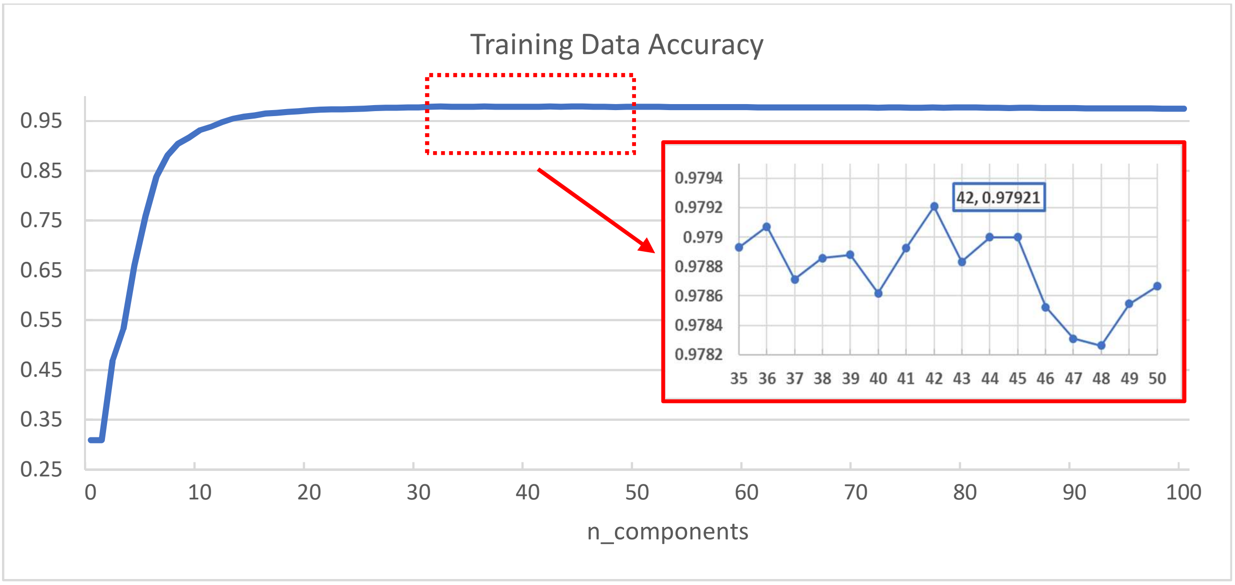
4.2 Support Vector Machines

Cross Validation on training datasets produced average accuracy of 93.59%.

By using the same parameters on the test data, Kaggle validation evaluated the produced predictions at 93.60%, which is almost identical to the cross-validation results.

4.3 Principal Component Analysis

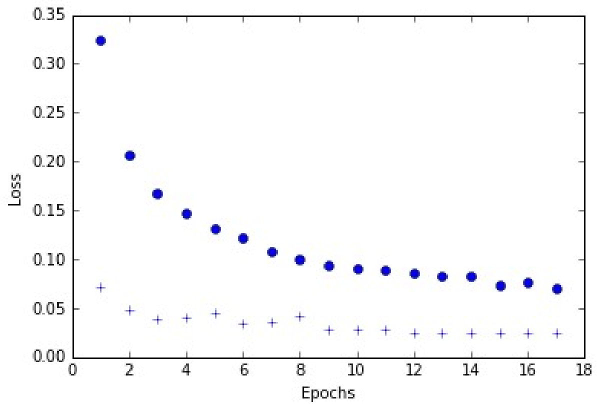
Optimal n_components. Cross validation of SVM method using PCA compressed data for n-components ranging from 1 to 100 shows that the highest accuracy of 97.92% is achieved using n_components =42. It is important to mention that the accuracy values for 35 to 45 are very close. See appendices 8.2 for more details.

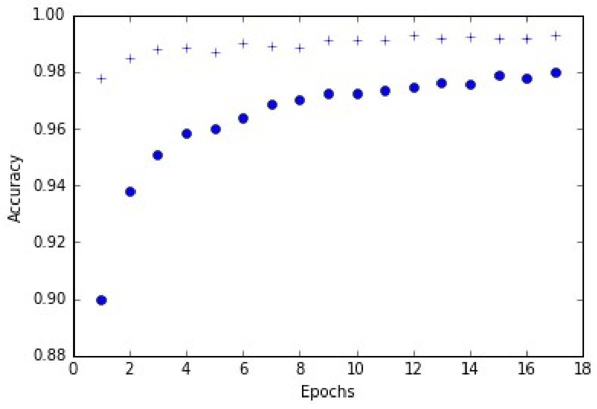


Test Set Accuracy. Using 42 as n_component, we reduce the dimensionality of the dataset. Predictions based on the test dataset produce we get 98.186% accuracy on the Kaggle submission page.

4.4 Convolutional Neural Network

Validation results. Please note that the algorithm stopped after 16 epochs as the accuracy has dropped comparing to the result for epoch 15. Results of the validation are shown in the graphs. We can clearly see that the loss per epoch is going down where the accuracy is going up with every epoch. See appendices 8.3 for more details.





‘Blue circles’ represent out of sample accuracy results; ‘blue crosses’ represent in sample results.

I would expect the results of out of sample and in sample to be closer if we increase the number of steps per epoch.

Test Set Accuracy. Using the parameters described in section 3.4. the predicted values produced accuracy of 99.586%, which is incredibly high.

4.5 Final results

	KNN	SVM (no PCA)	SVM (after PCA)	CNN
Initial Validation	97.24%	93.59%	97.92%	97.97% (short run)
Kaggle Validation	97.19%	93.60%	98.19%	99.59% (long run)

Results of the analysis demonstrate that it is possible to successfully implement a computer driven image recognition algorithms. The top result achieved by using convolutional neural network is 99.59%, which is remarkably high.

Another important take-away from the analysis' results is the impact of principal component analysis on the accuracy of SVM algorithm. We can see a major increase in accuracy between SVM (full dimensionality) and SVM (PCA reduced dimensionality)- from 93.6% to 98.2%. This is not surprising as it is likely that some attributes of our data consist of noise, which have been reduced by using principal component analysis.

5. Future work

This work has specifically focused on image recognition of hand-written digits. Future work in this are could focus on the following:

1) The work could be expanded into hand-written letters. This will expand number of classes and will provide a new challenge for the machine learning techniques.
2) There are clearly some examples of the hand-written digits, which could be even difficult to identify for a human (i.e. 1s and 7s could be similar). Therefore, it will be interesting to see if it will be possible to implement a 'recognition confidence' level, which would identify the samples, where the algorithm classification is uncertain. Practically, it could prevent errors and would also help to explain which digits the algorithms are struggling with and would hopefully lead to a solution of the problem.

6. Conclusion

This report focused on the use of appropriate machine learning techniques (such as KNN, SVM, PCA and NN) in order to aid image recognition as part of the 'Digit Recogniser' Kaggle competition. The report explained the used methods in details, discussed the results and proposed the potential extension of the analysis in the future work.

The result of the analysis demonstrated that with the use of convolutional neural networks it is possible to achieve accuracy of at least 99.59%.

7. References

Abdi, H. (2017). Normalizing Data. [online] Dallas University. Available at: https://www.utdallas.edu/~herve/abdi-Normalizing2010-pretty.pdf [Accessed 16 Apr. 2017].

Datascience.stackexchange.com. (2017). What is the best Keras model for multi-class classification?. [online] Available at: https://datascience.stackexchange.com/questions/10048/what-is-the-best-keras-model-for-multi-class-classification [Accessed 25 Apr. 2017].

En.wikipedia.org. (2017). Convolutional neural network. [online] Available at: https://en.wikipedia.org/wiki/Convolutional_neural_network [Accessed 26 Apr. 2017].

En.wikipedia.org. (2017). Cross-validation (statistics). [online] Available at: https://en.wikipedia.org/wiki/Cross-validation_(statistics) [Accessed 15 Apr. 2017].

En.wikipedia.org. (2017). Support vector machine. [online] Available at: https://en.wikipedia.org/wiki/Support_vector_machine#Multiclass_SVM [Accessed 25 Apr. 2017].

Fabien Lauer, Ching Y. Suen, Gerard Bloch. A trainable feature extractor for handwritten digit recognition. Pattern Recognition, Elsevier, 2007, 40 (6), pp.1816-1824.

James, G., Witten, D., Hastie, T. and Tibshirani, R. (n.d.). An introduction to statistical learning. 1st ed.

Kingma, D., and Ba, J. (2014). Adam: A Method for Stochastic Optimization. ICLR 2015.

Labusch, K., Barth, E. and Martinetz, T. (2008). Simple Method for High-Performance Digit Recognition Based on Sparse Coding. IEEE Transactions on Neural Networks, 19(11), pp.1985-1989.

Lantz, B. (2013). Machine learning with R. 1st ed. Birmingham: Packt Publishing Ltd.

McIlwraith, D. (2017) Extracting Structure from Data*,* lecture notes, Imperial College Business School, delivered on 27 February 2017

McIlwraith, D. (2017) Support Vector Machines*,* lecture notes, Imperial College Business School, delivered on 13 March 2017

McIlwraith, D., Marmanis, H. and Babenko, D. (2016). Algorithms of the intelligent web. 1st ed. Shelter Island, NY: Manning Publications Co.

Quora. (2017). Why does RBF kernel generally outperforms linear or polynomial kernels?. [online] Available at: https://www.quora.com/Why-does-RBF-kernel-generally-outperforms-linear-or-polynomial-kernels [Accessed 22 Apr. 2017].

Raschka, S. and Olson, R. (2016). Python machine learning. 1st ed. Birmingham: Packt Publishing.

Saedsayad.com. (2017). Support Vector Machine. [online] Available at: http://www.saedsayad.com/support_vector_machine.htm [Accessed 22 Apr. 2017].

Scikit-learn.org. (2017). 1.4. Support Vector Machines — scikit-learn 0.18.1 documentation. [online] Available at: http://scikit-learn.org/stable/modules/svm.html#svm [Accessed 26 Apr. 2017].

Wiesemann, W. (2017) Nearest Neighbours Methods*,* lecture notes, Imperial College Business School, delivered on 1 February 2017

Wiesemann, W. (2017) Support Vector Machines*,* lecture notes, Imperial College Business School, delivered on 17 March 2017

8. Appendices

8.1 K-NN cross validation values.

k	Accuracy		k	Accuracy		k	Accuracy		k	Accuracy
1	0.967380952		31	0.951642857		61	0.940261905		310	0.902952381
2	0.967380952		32	0.952738095		62	0.939952381		325	0.901357143
3	0.969380952		33	0.950857143		63	0.941452381		340	0.896333333
4	0.9724		34	0.950357143		64	0.940952381		355	0.897809524
5	0.967333333		35	0.951333333		65	0.9395		370	0.892952381
6	0.96947619		36	0.950857143		66	0.939119048		385	0.894547619
7	0.966428571		37	0.949119048		67	0.938880952		400	0.893357143
8	0.965714286		38	0.948761905		68	0.938690476		420	0.889642857
9	0.96652381		39	0.948095238		69	0.939928571		440	0.887285714
10	0.964404762		40	0.949166667		70	0.939595238		460	0.885
11	0.965190476		41	0.947452381		71	0.938261905		480	0.884738095
12	0.965095238		42	0.948404762		72	0.939357143		500	0.88297619
13	0.963571429		43	0.946380952		73	0.939452381		520	0.877261905
14	0.9635		44	0.947880952		74	0.93747619		550	0.87847619
15	0.962333333		45	0.945785714		75	0.937166667		585	0.875857143
16	0.960142857		46	0.946833333		80	0.937666667		620	0.868333333
17	0.960761905		47	0.944761905		95	0.933166667		655	0.869690476
18	0.960761905		48	0.946071429		110	0.93147619		690	0.866928571
19	0.959928571		49	0.945642857		125	0.928666667		725	0.864142857
20	0.958166667		50	0.943761905		140	0.925642857		760	0.855380952
21	0.958595238		51	0.944642857		155	0.923071429		795	0.858095238
22	0.958190476		52	0.9445		170	0.918714286		840	0.849333333
23	0.957190476		53	0.944333333		185	0.918547619		880	0.85247619
24	0.95697619		54	0.942428571		200	0.914095238		920	0.850119048
25	0.954761905		55	0.942333333		215	0.912261905		960	0.847595238
26	0.954428571		56	0.941928571		230	0.910261905		1000	0.845142857
27	0.95502381		57	0.941428571		245	0.910595238			
28	0.953285714		58	0.941071429		265	0.905595238			
29	0.953714286		59	0.942428571		280	0.905952381			
30	0.952071429		60	0.940761905		295	0.904642857			

8.2 N-component cross-validation

N components	Accuracy	N components	Accuracy	N components	Accuracy
1	0.309286875	35	0.978928789	69	0.977404772
2	0.468761251	36	0.979071431	70	0.977142961
3	0.532856874	37	0.978714223	71	0.977190597
4	0.659572388	38	0.978857017	72	0.976976226
5	0.759309222	39	0.978880935	73	0.977309594
6	0.838595709	40	0.978619024	74	0.977071462
7	0.880047358	41	0.978928421	75	0.977000135
8	0.904547858	42	0.979214123	76	0.976976402
9	0.91680994	43	0.978833216	77	0.977214418
10	0.931666814	44	0.979000002	78	0.976881002
11	0.938952354	45	0.979000033	79	0.97740488
12	0.947785898	46	0.978523831	80	0.977166731
13	0.954999953	47	0.978309614	81	0.977142958
14	0.958595138	48	0.978261859	82	0.976476198
15	0.961118642	49	0.978547516	83	0.976500194
16	0.964951973	50	0.978666572	84	0.976119248
17	0.966452151	51	0.978761805	85	0.976452309
18	0.968356857	52	0.97842852	86	0.976595328
19	0.969737948	53	0.978285558	87	0.976071563
20	0.971237906	54	0.978214177	88	0.975785985
21	0.97264283	55	0.978142715	89	0.975857349
22	0.973452343	56	0.978285566	90	0.975880996
23	0.973571472	57	0.977999928	91	0.975642932
24	0.974214222	58	0.977761904	92	0.975642938
25	0.975000084	59	0.97771446	93	0.975571563
26	0.975928712	60	0.977690566	94	0.975499987
27	0.97645249	61	0.977642941	95	0.975381101
28	0.976690489	62	0.977476258	96	0.975523743
29	0.977214455	63	0.977642791	97	0.975404724
30	0.977428673	64	0.97764278	98	0.975071501
31	0.978381082	65	0.977547669	99	0.975023802
32	0.978976264	66	0.977523806	100	0.974857133
33	0.978666836	67	0.977238066		
34	0.978785864	68	0.977142862		

8.3 Results of the initial validation of CNN

```
Epoch 1/25 500/500 [================] - 142s - loss: 0.3248 - acc: 0.8999 - val_loss: 0.0718 - val_acc: 0.9779

Epoch 2/25 500/500 [================] - 147s - loss: 0.2070 - acc: 0.9381 - val_loss: 0.0484 - val_acc: 0.9846

Epoch 3/25 500/500 [================] - 157s - loss: 0.1670 - acc: 0.9509 - val_loss: 0.0385 - val_acc: 0.9879

Epoch 4/25 500/500 [================] - 145s - loss: 0.1469 - acc: 0.9582 - val_loss: 0.0412 - val_acc: 0.9884

Epoch 5/25 500/500 [================] - 147s - loss: 0.1314 - acc: 0.9601 - val_loss: 0.0450 - val_acc: 0.9867

Epoch 6/25 500/500 [================] - 142s - loss: 0.1226 - acc: 0.9636 - val_loss: 0.0342 - val_acc: 0.9898

Epoch 7/25 500/500 [================] - 148s - loss: 0.1073 - acc: 0.9685 - val_loss: 0.0360 - val_acc: 0.9890

Epoch 8/25 500/500 [================] - 154s - loss: 0.0996 - acc: 0.9704 - val_loss: 0.0415 - val_acc: 0.9886

Epoch 9/25 500/500 [================] - 143s - loss: 0.0938 - acc: 0.9725 - val_loss: 0.0287 - val_acc: 0.9914

Epoch 10/25 500/500 [================] - 148s - loss: 0.0914 - acc: 0.9726 - val_loss: 0.0280 - val_acc: 0.9914

Epoch 11/25 500/500 [================] - 165s - loss: 0.0891 - acc: 0.9736 - val_loss: 0.0281 - val_acc: 0.9910

Epoch 12/25 500/500 [================] - 154s - loss: 0.0865 - acc: 0.9745 - val_loss: 0.0251 - val_acc: 0.9927

Epoch 13/25 500/500 [================] - 150s - loss: 0.0827 - acc: 0.9761 - val_loss: 0.0249 - val_acc: 0.9919

Epoch 14/25 500/500 [================] - 153s - loss: 0.0827 - acc: 0.9757 - val_loss: 0.0240 - val_acc: 0.9922

Epoch 15/25 500/500 [================] - 151s - loss: 0.0737 - acc: 0.9789 - val_loss: 0.0250 - val_acc: 0.9917

Epoch 16/25 500/500 [================] - 151s - loss: 0.0768 - acc: 0.9778 - val_loss: 0.0247 - val_acc: 0.9916

Epoch 17/25 500/500 [================] - 145s - loss: 0.0701 - acc: 0.9797 - val_loss: 0.0243 - val_acc: 0.9927

Epoch 00016: early stopping
```