Handwritten Recognition Using Artificial Neural Networks

Htet Su San
U1971904

Contents

I. Abstract

Due to the large use of various technologies in all sectors and the daily usage of them in order to process information, handwriting recognition has become an interesting topic in research area. The handwriting recognition algorithm is vital in cases when people need to change handwriting papers into electronic versions, for example signatures, in order to save information for future purposes. Handwriting recognition means that the computer or machine can detect and interpret handwriting input from devices such as touch screens, electronic pens, photos, paper documents etc. It is a complicated process to read handwriting characters or digits due to various handwriting styles. This paper proposes a software that can detect handwritten image digits or live handwritten digits. It is developed on an artificial neural network. Although various kinds of techniques and methods have been utilised for a handwritten recognition, neural networks have been used in most of the experiments due to it being effective and powerful. Moreover, this paper explains about methodology, design, and architecture to build a handwritten recognition system. The aim of this paper is to develop a model that can give higher accuracy to predict the handwritten digits.

II. Introduction

Handwritten digit recognition is the computer's capability to detect the handwritten digits of humans, from various sources like images, papers, touch screens and etc, and organize them into 10 predefined classes (0-9) (Pashine et.al, 2021). It has been a popular research topic in deep learning and machine learning. However, there are a lot of challenges to build a handwritten recognition system as different people have different handwriting styles. In this report, different kinds of machine learning algorithms are presented to utilise a model for handwriting recognition, representing support vector machine, multilayer perceptron, and convolutional neural network.

The applications that are used to detect handwritten letters, characters, and digits provide a lesser amount of time-consumed, less human power and increased cost effectiveness. As an example, it is useful using handwritten recognition embedded automated processing applications in banks for processing cheques because many employees would be needed to make the process without them.

The handwritten recognition systems can be based on by using biological neural networks which help humans and animals to build non-linear and complex relationships. That indicates they can be created from the artificial neural network. According to Aqab & Tariq (2020), the artificial neural networks (ANNs) methodology is regarded as the best solution to develop the handwritten recognition system. ANNs can provide the closest way to read the handwritten like the way the human brain simulates in a more simplified form. It can even give better performance than human abilities. Each person has his/her own handwriting styles, some of which are hard to understand. At the same time, reading several documents which have different handwriting, could be time consuming and tedious. In my opinion, a neural network could be the solution for the proposed system because it can identify complex data which cannot be solved by a human. However, several methods and algorithms that can be applied to develop a model for handwritten recognition are discussed in this paper along with their advantages and disadvantages.

The accuracy of a model is very important, this is due to a higher accuracy model giving better results and being more suitable for real world application, compared with lower accuracy model. As an example, a high accuracy model is very vital for automated bank cheque processing system which detect the amount and money on the check. If the software cannot perform the correct process on a consistent basis, it can lead to some serious issues. An accuracy of different models based on different ML techniques is described in the related work in

Section 4. In this report, Section 3 covers the background theoretical knowledge and Section 4 covers the Related Works, Section 5 covers PLES Issues and Section 6 is about literature review. Section 7 covers Project Aims and Methodology and Section 8 covers Analysis and Design. Section 9 covers about implementation and 10 is all about Tools and Libraires. Section 11 is Results and Discussion and 12 is Conclusion and Future Improvements. Section 13 and 14 are Appendices and References.

III. Background theoretical knowledge

A. Artificial Intelligence

Artificial intelligence (AI) is a broad topic in the computer science subject with regards to developing smart machines which have the abilities of processing tasks which requires human intelligence and AI systems to be able to think and act humanly and rationally (Russell et al., 2016). The term of AI can be basically described as computer or machines which can mimic "cognitive" functions that are associated with the human mind.

Similarities can be drawn between the computer systems ability to read and translate words, digits, and handwriting characters, and that of the daily operation of a human brain. Artificial intelligence consents the machine to learn from experience, adjust to new data (inputs), and operate tasks that can be performed by humans (Aqab & Tariq, 2020). Areas of artificial intelligence consist of machine learning, neuron network, and deep learning.

B. Artificial Neural Network

Artificial Neural Network (ANN) means information processing concept or computing systems which are adapted by biological neural network (Aqab & Tariq, 2020). Biological neural networks are networks that represent the human brain. This does not mean that ANN is identical to the biological neural system, but it is rather designed like the way human's or animal's brain works. The ANN networks are created by many interconnected neurons to obtain specific targets. Using neurons, ANNs learn to recognise patterns within sets of data. ANN learns to process from example, like human brain. Therefore, it can be utilised for a system for data classification or character recognition via learning procedures which include adapting the system to a connection. ANN consists of several simple processor's web with a slight amount of local memory in each. The processers car connected by unidirectional communication channels and processes on local data specifically.

ANNs can be applied to construct a handwritten recognition system. A large number of handwritten digits which are training samples are used to build an ANN model. In other words, the neural network uses the examples to automatically imply rules for recognizing handwritten digits. The more the training examples, the more accurate the model is because the network can learn more about handwriting styles and structures.

C. Machine Learning

Machine Learning (ML) is a division of artificial intelligence (AI) which is mainly used for data and algorithms to imitate like the human being (IBM, n.d.). The difference between AI and ML is that AI is the science and ML is the study and application of the mechanisms that make it work. ML is training computers to enhance a performance standard using sample data or previous experience (Alpaydin et al., 2014). ML is used for generating general models from data of specific examples. There are four categories of machine learning known as supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning. The sort of algorithm data scientists prefers to use relies on what sort of data they would like to predict.

The services we use today like Netflix, YouTube, and Spotify use ML techniques to build a recommendation system. Moreover, ML is part of the operations of search engines like Google and other key social media platforms like Facebook and Instagram.

A machine learning model is trained with the training data which is particular to the given problem domain and then it is tested by testing data for its accuracy. By doing so, the model learns how to solve certain problems based on learning. Fig. 1 shows a simple presentation of the machine learning model applied in the handwriting recognition system. The model takes an image as an input which contains a Handwriting digit and defines the specific digit based on the learning data.

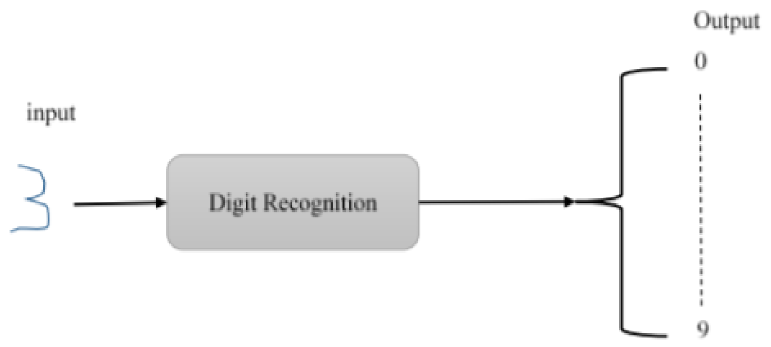


Figure 1: Machine Learning Handwritten Recognition Model (Aqab & Tariq, 2020)

D. Deep Learning

Deep Learning is a subset of machine learning with regards to algorithms inspired by the structure and function of the brain called artificial neural networks (Brownlee,2019). Like the way we learn things in real life by experience, deep learning algorithm would process tasks many times, each time it will get improved and give a more accurate result. The term "deep learning" comes from the fact that the neural networks have many (deep) layers that supports learning. The neural network has layers of units where each layer takes some value from the previous layer. By doing so, applications that are based on neural networks can handle inputs to obtain the desired output. Similarly, it is the way that the neurons transfer signals around the brain, and values are transferred from one element in an artificial neural network to another for desired result by carrying out the needed procedure.

There are hidden layers in artificial neural networks. It is situated between input and output layers. The input and output layers of a deep neural network are called visible layers. The hidden layers can be represented as deep neural network which is for performing computation of the values added in the input layer. The example of deep neural network can be found below as Fig 2. In handwritten recognition system, the deep neural network is involved in learning the characters to be detected from handwriting images. With adequate training data, the deep neural network can be able to perform any function that a neural network is supposed to do. It is only possible if the neural network has enough hidden layers, although the smaller deep neural network is more computationally efficient than a more extensive deep neural network.

E. What is Handwritten recognition?

Handwriting recognition is the capability of a computer to recognise and interpret handwritten input (Pashine et al., 2021). Suppose that input could be a scanned handwritten document or an image of handwritten characters or digits. Moreover, it could be from touch screens due to the Today's improved technology. The research on handwriting recognition has been started since the 80s and there has been accuracy problems from the start (Think Automation, n.d.).

There are two kinds of handwritten recognition, offline and online. The first one is developed from the handwritten input which is derived from images. The latter one is through a touchscreen. In this report, we will focus on handwritten recognition system based on image input. The mnist data set which contains handwritten digit images will be applied to build a model for handwritten recognition software.

F. How it works?

Even though there are various ways in which handwritten recognition works, it depends on how you develop your system. Generally, it is all about utilising the computer to change handwriting into a format that the computer can process. One way for it to work is that the computer will be able to detect the handwritten digit images and compare the data to the database of known digit and identify it. For example, in handwriting character recognition, an algorithm called optical character recognition (OCR) in which the computer focuses on each letter and give the results which character it is by comparing to the dataset of known words (Think Automation, n.d.).

G. The problems with Handwritten Recognition

As mentioned above, there has been many issues related with the accuracy of handwritten recognition even dating back to the 80s. However, this is expected as even humans have difficulty reading and recognising each other's handwriting. Consequently, it is challenging for a computer to predict every different handwriting style with 100% accuracy. The programmers must attempt to provide enough samples of how every digit might seem in order to combat this problem to the best of their abilities. In the case of handwriting recognition from images, there are also many other challenges due to their being many angles and sizes to look upon. To simplify this means the computer could struggle identifying the digit or character if the photo is taken from poor angle.

H. OCR applications (Optical Character Recognition)

Handwritten recognition is very effective in OCR applications. OCR applications have to connect with a scanner to transform printed characters into digital form (HP, n.d.). As a result, editing the document by using a related program, is a possible solution.

Handwriting OCR needs much more advanced technology than traditional OCR. Instead of using simple techniques to identify letter structures, this type of OCR requires a highly trained machine learning model and advanced computer vision engines to actually read what is written like a human would hence giving out more reliable results.

IV. Related works

Handwritten digit recognition (HDR) can be regarded as one of the most critical machine learning issues. It has been utilized commonly as experiments for theories of ML algorithms for years. There are a broad range of research done for handwritten recognition by using ML and Deep Learning techniques like Support Vector Machine (SVM), Artificial Neural Network (ANN), Convolutional Neural Network (CNN) etc. It can be said that all the experiments are tested by using MNIST dataset.

First of all, Islam et al (2017) designed and implemented a multi-layer fully connected neural network with one hidden layer for handwritten digit detection. The testing has been done by using MNIST database. 28000 images for training and 14000 images for testing was performed. It was said that the accuracy of the implemented multi-layer ANN system was relatively excellent, showing 99.60% for testing case.

Besides, Jagtap & Mishra (2014) put their effort in building fast efficient artificial neural network for handwritten digit recognition on GPU to minimize the training time. The Standard back propagation (BP) learning algorithm with multilayer (MLP) classification is selected and utilised on GPU for parallel training. The researchers decided that they should use back propagation algorithm on GPU rather than CPU. However, for back propagation with small input data and couples of hidden neurons CPU based execution can obtain better performance. But for the case when input size is larger, then GPU based parallelization is appropriate to minimize the training time. The accuracy of the experiment is 97.7% on training data and 98% on testing data. Therefore, it could say that the experiment is successful.

Moreover, Shamim et al (2018) applied few machine learning algorithms to build a handwritten digit recognition system. The used algorithms are Multilayer Perceptron, Support Vector Machine and Random Forest etc. The authors compared the results from all algorithms, and it turned out that the highest accuracy was achieved by Multilayer Perceptron with 90.37%.

Similarly, Dutt & Dutt (2017) built a handwritten digit recognition model with some of the commonly used machine learning algorithms like SVM, KNN & RFC and with Deep Learning techniques like multilayer CNN with the help of libraries like TensorFlow and Keras. The result shows that CNN model obtained the highest accuracy with 98.70% while SVM, KNN and RFC chieved 97.91%, 96.67% and 96.89% respectively.

Furthermore, Chychkarova et al (2021) mentioned multiple classification algorithms of handwritten digit recognition from images. They discussed about SVM, KNN and RF and neural networks and compared the achievement rate between them. Then, they considered to apply two neural networks: sequential and convolutional. It is described that the best accuracy was found in convolution neural network with 97.6%. However, the researchers performed the image pre-processing and dataset conversion to improve the model. After that, the accuracy went up to between 98-100%.

Likewise, Siddique et al (2019) observed the variation of accuracies of CNN to classify handwritten digits with hidden layers and epochs. It is mentioned that the network was trained by stochastic gradient descent and the back-propagation algorithm. The accuracy was produced for six instances for the different parameter using CNN. Overall, among all the experiments, the maximum accuracy was obtained with 99.21% running 15 epochs.

In addition, Sultana et al (2018) explained various CNN architectures for image classification and built some CNN models ranging from LeNet-5 to the newest SENet model. The details discussion about the models are described. Overall, AlexNet, ZFNet and VGGNet are the subset of LeNet-5 with same architecture but their networks are bigger and deeper. It is mentioned that GoogLeNet and ResNet achieved the better accuracy in combining inception module and residual blocks. Moreover, SENet bring out the hope that it might be effectives for strong features. In my opinion, the writers wanted to point out that all CNN architectures are effective and reliable but have advantages and disadvantages in each and it depends on the programmer to choose one which is suitable for the model.

Correspondingly, Hossain & Ali (2019) applied CNN in their implemented model which has 7 layers and the result showed 99.15% of accuracy running after 8 epochs. They said that CNN proves to be far better than other classifiers and the more accurate result can be attained with more convolutional layers and a greater number of hidden neurons. According to the research, most of the works for handwritten recognition was done by using

CNN model and the results showed that the experiments can predict the handwritten inputs relatively well, achieving around 90-100% accuracy. Alwzwazy et al. (2016), Kussul & Baidyk (2004) and Wu (2018) used CNN model to build the handwritten recognition system and the accuracy of the model is reasonably good.

V. Professional, Legal, Ethical and Social Issues (PLESI)

Professional

The British Computer Society (BCS) code of conduct is presented to the professional in computing field in the UK containing four rules which are public interest, professional competence and integrity, duty to relevant authority and duty to the profession (BCS, 2015). Professional issues can occur in various fields by implementing this handwritten recognition software. One of these fields is banking, in particular for banking cheque processing, it could affect the professionality of the developers and banking image if the system makes errors or bugs. Banking systems are obviously important because it is related with financial security of people which is also associated with people' lives. Moreover, the profile of the bank is crucial because no one will invest money and trust the money if something goes wrong, for example, the system detects the digits on the cheque paper wrong and the money is withdrawn from another person's account. This could also affect the employees as with fewer transactions in the banking sector there will be less demand for work and hence less jobs opportunities. A real-world example of this professional issue can be seen with the 2008 Financial Crisis where many banks failed due to the errors of the banks. This event whilst not caused by Machine Learning certainly shows what can happen if errors occur and the public lose trust in the banks. This causes a huge professional problem as whilst certain large banks were deemed 'too big to fail' and therefore bailed out by the government, other smaller banks could not handle to pressure and failed. This causes many smaller businesses to close and therefore if an error occurs with the handwriting recognition on a regular basis it could lead to another extremely critical professional issue to one of the countries core sectors.

Another sector which may suffer professional issues is in the medical field, handwritten digit recognition can be utilised for systems like patient prescription digitization which might lead to professional issues if the system cannot detect the digits on the paper. This would case serious issues for hospitals and patients. The hospital or healthcare gives the wrong prescriptions to the patient and in this case, it can even lead to threatening of the patient's life. If this happens, the image of the hospital will be gone, and no patients will trust the hospital and so it can lead to not only professional issues but also legal issues. Moreover, it is not even affecting the hospital but also people who works in hospital. For the developer who develop that software will also face the profession issues and it can ruin his/her career.

Another area in which handwritten digit recognition can be used is for detecting license plates and nowadays, the government and police force implement those kinds of systems in order to more effectively control and enforce law on the roads. Similarly, like above cases, if the system predicts wrong, it could result in professional issues. For example, a case when the driver breaks laws like driving over the speed limit or causing an accident, the system will be able to find out the details information of car and driver. However, if the system gives wrong answer, it might end up not finding the person who breaks the law and for the worst scenarios, the police might lead to accusing the wrong person. This would ruin the professionality or moral of the police or the government and also the programmer.

Legal

Handwriting recognition might lead to some very critical legal issues upon implementation to which are related to financial cases. For example, handwritten recognition can be applied to the banking cheque procedures. As a result, it will be very effective and useful for banking system. On the other hand, it can cause some risks to banks which means the system can give wrong prediction due to technical issues. Because of this, the bank might end up dealing with legal issues. In my opinion, any system that can be used in banking sector need to be 100% flawless or at least should not have major errors because it has to solve with the financial which is so important for people life. Moreover, signatures are required to make bank processing for most of the cases. In this situation, handwritten recognition plays a vital role. Due to improvement of the technology, in recent years, banks are using digital signatures which are easy to process and effective for banking procedures. For that case, legal issues can occur if the system cannot detect the signature correctly and properly. Suppose that someone can make fake signatures because you cannot draw the same as the real signature on the screen. In my opinion, it is quite difficult to draw and could lead to legal issues. This could result in legal issues in particular when a client sues the bank for the error resulting in a loss of their savings.   Similarly, at police station, you need to give your signatures when you report the crime. In that case, the same legal issues could happen, and this could be more serious because it is related with criminal cases which are related with people lives and safety. Another example is that using digit recognition in license plate can lead to legal issue. To explain that, if the system detects the wrong numbers and the driver breaks the driving laws, it could end up making all the blame to the wrong person. Moreover, the real person who breaks the law cannot be found because of the faulty of the system.

Ethical

Ethical issues occur when a given decision or scenario, or activity creates a conflict with a society's moral principles. Both individuals, organizations, and businesses can be involved in these conflicts since any of their actions might be put to issue from an ethical perspective. Implementing the handwritten digit recognition can lead to ethical issues in all various kinds of fields. In the banking sector, if the system predicts the digit wrong, it might lead to ethical issues like the image of the bank will be ruined and the people will not invest in that bank. Moreover, customers will lose the money that they deposit, and it could affect their living standard or lifestyle. It also affects on life of the programmer, and it might lose his/her job and it would be difficult to find another job due to the experience and the profile. For healthcare sectors, the handwritten recognition plays an important role. The system can help for sorting out and managing prescription of the patients. If the system detects the handwriting wrong, it will occur ethical issues to healthcare centre or GPs. The patient would end up getting wrong prescriptions and it will affect the disease that he/she is experiencing. As a result, the hospital loses the trust from the patient, and it will damage the ethical or moral of the company. Moreover, the GP who gives the wrong perceptions might go through challenges in the whole career. For the patient, he/she will suffer health issues and might even get mental issues because of the faculty of the system. Additionally, there could be ethical issues in case of detecting licence plates when the system predicts the wrong digits. In this case, police might encounter the ethnicity in his job for accusing the wrong person due to the system error. It will also affect on the government and the victim will blame for it and the morality of the government can be ruined. For the victim if no other proof is found, he/she might end up paying fine or something. As a consequence, that person would struggle financially. Another cases that could lead to ethical issues because of the system's problem is that the teacher would give wrong mark and the student could fail the exam in case like the system detects the wrong digit of student ID from the paper. The teacher might encounter moral issues because she fails the student, and it even affects the morality of the school as well. Furthermore, the student might experience mental issues and problems with the patients because he/she fails the exam. Overall, these cases happen because of the faulty and bad accuracy of the system.

Social

Social issues are problems which affects a large number of people, and the main thing is that the individuals in the society may not have the control over these problems. Social issues differ from one society to another based on several reasons. A social issue can be caused due to geographical, educational, economical or a political reason. For example, the banking cheque processing system, it could lead to a number of social problems if the system gives the wrong prediction. The banking profile status value will be dropped in the social community. All the customers will not deposit in the bank and other competitor banks will get more customers. The market value of the bank will be plunged as well. The bank employees will lose the job and will experience the financial problems. At the worst case, they might end up being homeless and their social image will be ruined as well. If they have children, it will cause a lot more problems to them because they will not be able to feed their families. Additionally, there could have social issues in healthcare sector. If the system gives wrong prescription number, the GP gives the wrong medications to patients and as a consequence, the patient might suffer health problems. This will damage the social status of the hospital and the hospital will lose many patients (customers) and it also affects the financial status of the hospital that the hospital will not be able to regulate the processes or functions. For the patient, he/she might experience mental problems and will cause him/her more cost on health care because of the faulty of the system generally. As mentioned before, handwritten digit recognition or character recognition can be applied in educational sector for marking proposes by calculating the mark of the student according to the student ID. Another way is scanning the handwritten exam paper and scanned it and transformed into word documents which makes the teacher read and mark the papers quicker and easier. For these cases, the social issues can be occurred if the system makes mistakes and predicts the wrong characters or digits. The teacher might end up failing the student and it can ruin not only the social image of the teacher but also that of the school. The patient will not trust the school teaching system and will might transform into another school. On the other hand, the school will lose the number of students and it even might end up closed. The student will have to face social issues due to the failure of the exam like punishments from the parents etc. It might also affect the educational performance of the student. For scenarios like the student gets the offer from the university and only one position left for it but she/he needs the good academic mark, she/he will not able to join the university and another person could take over his/her place. As a result, the student might end up not joining the university that she/he wants, and it can ruin the future/life of the student. Another case like handwritten recognition is used, license plate detection, it could make some social issues. The driver might even go to the jail due to the wrong prediction of the system or would need to pay the fine. Either way, it will affect the life of the driver including the social status and financial. From the perspective of the government, the civil people will not approve or like the government and make the social status declined in the social community. For the worst case, it might even lead to the crisis of the country. The police who arrest the driver could face social issues because he trusted the system without using critical thinking. It might affect the social status of the police as well.

As a summary, the above discussion is about professional, legal, ethical and social issues related with the factors like banking, educational sector, driving rules, and healthcare sector. All the possible scenarios have been described depending on those sectors.

VI. Literature Review

The handwritten recognition is a big mission for researchers. There are many thesis papers written by researchers in which different kinds of machine learning algorithms are applied for handwritten recognition. In this section, reviewed papers to build a model for a handwritten recognition system with machine learning techniques are discussed.

A. Mnist Dataset

In ML research area, several standard databases have developed in which the handwritten digits are processed to build a ML model with higher accuracy. Among them, Mnist dataset is widely used for digit handwritten recognition which consist of monochrome images of handwritten digits, and it can be accessed freely. Deng (2021) describes that it became a standard for fast-test machine algorithms. The whole dataset is moderately smaller than new benchmark datasets. The structure of the data is not complex, and it does not require any copy right. Because of this, it is commonly used for solving handwritten recognition problems. The database contains 60,000 training images and 10,000 test images. Mnist database was formed based on bigger dataset called NIST Special Database 3 (SD3) and Special Database 1 (SD1) which consist of binary images of handwritten digits. NIST initially assigned SD-3 as the training set and SD-1 as the testing set. The first one is created by the people who work for United States Census Bureau and the second one is built by high school students. In NIST database, the images are size standardised to 20x20 pixels. According to (), SD3's structure is clear and easy to understand compared with SD1. Obviously, the reason is that SD3 is introduced by United States Census Bureau so that it is a better version than the one done by high schoolers.

The NIST database, as opposed to Mnist database, is difficult to access and hard to apply. It was created with high cost and the structure of the data is compactible and efficient. Despite the fact that source code for the data is given, it still has challenges to use it on different platforms especially to adapt on modern platforms. Because of this, the NIST introduced another version of NIST dataset, which is easier to access. However, the architecture of the dataset and images in it, is not the same with that of MNIST and are not directly compatible.

The Mnist dataset contains same portion of images from SD-3 and SD-1, 30000 of each in training while testing set 5000 of each. In Mnist database, all images consist of black and white handwritten digits which are size normalised, centred in a fixed size image where the centre of gravity of the intensity lies at the centre of the image with 28x28 pixels. Thus, the dimensionality of each image sample vector is 28 * 28 = 784, where each element is binary. The value of the pixel is ranging from 0 to 255 integers. The training set of Mnist data (train.csv) contains 785 columns. The first column in the sheet is label which represents the digit drawn by the user and the other columns are the pixel-values of the related picture.

The pixel column from the training set has a name like pixelx in which x could be ranging an integer from 0 and 783. To find this pixel on the image, assume that x as x = i * 28 + j, where i and j are numbers between 0 and 27. After that, pixelx is positioned on row i and column j of a 28 x 28 matrix, (indexing by zero).

As an example, pixel31 indicates the pixel that is in the fourth column from the left, and the second row from the top, as in the ascii-diagram below.

Visually, if we omit the "pixel" prefix, the pixels make up the image like this:

```
000 001 002 003 ... 026 027

028 029 030 031 ... 054 055

056 057 058 059 ... 082 083

 |   |   |   |  ...  |   |
728 729 730 731 ... 754 755

756 757 758 759 ... 782 783
```

The test data set, (test.csv), is the same as the training set, except the fact that it does not contain the "label" column.

This database is simple and easy to use for trying out ML techniques and pattern recognition methods on real-world data while spending minimal efforts on pre-processing and formatting.

B. Machine Learning/ Deep Learning in handwritten recognition (more)

Handwritten character recognition is a subject of research area in artificial intelligence, computer vision, and pattern recognition. A computer which can perform handwritten recognition should be able to obtain and detect digits or character in papers, images, touch-screen gadgets etc. Then, convert them into ML encoded form. Optical character recognition is an example for those kinds of system which is a business solution for transforming transcription of handwritten documents into pdf. There are more improved intelligent systems for handwritten recognition which can be described as image recognition problem.

Deep learning, a part of Machine Learning, is broadly used in handwriting recognition. The task of manual transcribing is an arduous process that ought to have errors. Automatic handwriting recognition cuts down the time required for transcribing large amounts of text and acts as a framework for machine learning application development. Significant processes right up to human-competitive performance from classical performances have been possible because of extensive research.

C. Where is handwritten recognition applied?

1. Healthcare and pharmaceuticals

Handwritten recognition has been applied in healthcare area, for example, patient prescription digitization. It is time-consuming handling millions of petabytes of medical PDFs daily. Instead, handwritten text detecting can be utilised for patient enrolment and form digitization (Matcha, 2021). By doing so, it can significantly improve user experience in hospital/pharmaceuticals, and it will be a lot faster and quicker to process the queries.

2. Insurance

Handwritten recognition can be useful in insurance industry. Generally, a large insurance company has to process more than 20 million documents every day and a delay in management of the claim documents can impact the company severely (Matcha, 2021). Those documents consist of different kinds of handwriting styles and in my opinion, pure manual automation of processing claims is going to completely slow down the pipeline.

3. Banking

Handwritten recognition plays a main role in banking field. Obviously, people use cheques on a regular basis and cheques still are commonly way to use in most non-cash transactions. The current procedure of cheque insists a bank employee to process it and make input from cheque paper and validate the entries like signature and date (Matcha, 2021). For a big number of cheques to be processed every day in a bank is a hard work. A handwritten text recognition system could help to solve this problem and it could save costs and human power.

4. Online Libraries

Huge amounts of historical knowledge are being digitized by uploading the image scans for access to the entire world (Matcha, 2021). But this effort is not very useful until the text in the images can be identified which can be indexed, queried, and browsed. Handwriting recognition plays a key role in bringing alive the medieval and 20th century documents, postcards, research studies etc.

5. License plate

It would be useful to use digit recognition for detecting the license plates. For example, if a driver drives over speed limit, then the system can scan the image of the plate and give the information of car details and owner information. As a result, it would be a lot easier to sort out the fines in compared to the traditional procedure.

6. Schools and Educational Sector

Handwritten recognition can be applied to use in schools. As an example, teachers do not need to be stressed with reading different handwriting styles. By using handwritten recognition, they could transform the images of papers to the documents, and it would be much easier to read and understand. As a result, it will save a lot of time and energy.

D. Techniques for Handwritten recognition

1. K-nearest neighbours

KNN is a very simple algorithm to understand and implement. At the same time, it offers remarkably high efficiency in practical applications. Its simplicity, which is obviously its advantage, is also a feature that makes it sometimes unnecessarily overlooked when solving more complex problems. Generally, it is a broad topic for using this algorithm. Both unsupervised and supervised learning, regression and classification problems can be solved with it. KNN can be regarded as a lazy algorithm because at least one parameter needs to be used which is the number of neighbours. KNN is very useful for cases where the relationship between the input data and the result is unusual or complicated which means that the classic methods, for example, linear or logistic regression might not be able to perform the task. It is commonly used for its easy of interpretation and low calculation time. This classification can be applied for handwritten digit recognition according to Kumar et al (2011).

2. Multilayer Perceptron

A multilayer perceptron (MLP) is a class of feedforward artificial neural networks (ANN) (Pashine et al., 2021). It has three layers which are input, hidden and output layers. Each layer contains many nodes which are also known as neurons and each node is linked to every other node of the next layer. The general MLP has three layers, but the hidden layers can be added with no limitation.

The number of nodes in the input and output layer depends on the number of attributes and apparent classes in the dataset respectively. The number of nodes in the hidden layer is difficult to determine due to the model erratic nature and thus selected experimentally. Every hidden layer of the model can have different activation functions for processing. For learning purposes, it uses a supervised learning technique called backpropagation. It minimizes the sum of the squared errors for the training samples by conducting a gradient descent search in the weight space. The number neurons in a hidden layer in the same are adjusted during its training as well. According to Das (2010), MLP is commonly used for handwritten recognition problems. However, in my opinion, not many research papers are found using it for handwritten recognition system.

3. Support Vector Machine

Handwriting recognition can be measured as a problem of supervised learning and classification from a discriminative classifier point of view. (Aqab & Tariq, 2020). Support Vector Machine (SVM) is a supervised machine learning technique which is applied for classification and regression analysis (Pashine et al., 2021).

Moreover, SVM which is a discriminative classifier can be used a model which can help to build the handwritten recognition. Furthermore, SVM is a machine learning method like neural network which is commonly used to solve the handwritten recognition problems. SVM can be counted as a computational algorithm that observes a hyperplane or a line in a multidimensional space that divides the classes (Aqab & Tariq, 2020). The division between two or more linear classes can be done by any hyperplane which is known as linear classification. Nevertheless, many hyperplanes can be applied to classify the same data set (Figure 2). The purpose of SVM approach is to find the best separation hyperplane.

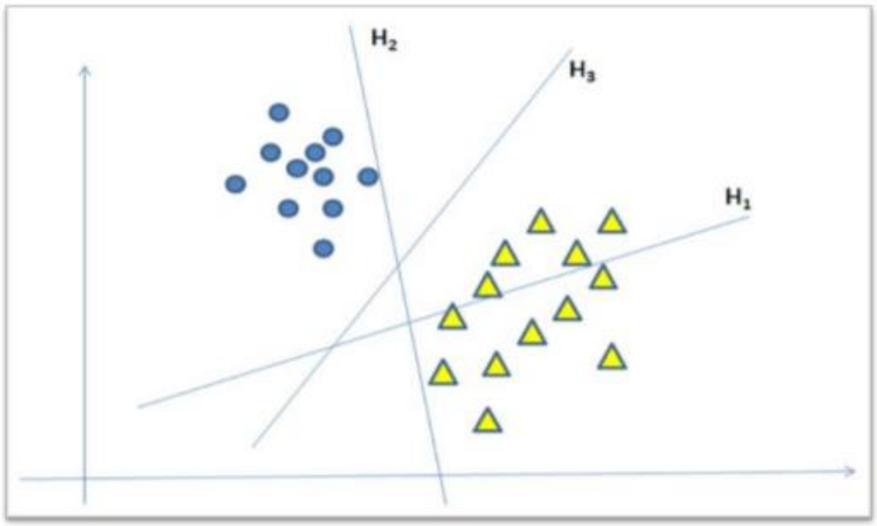


Figure 2: SVM Hyperplane (Aqab & Tariq, 2020)

i. Hyperplane

Hyperplanes are decision boundaries that help classify the data points (Goyal, 2020). Data points falling on either side of the hyperplane can be attributed to different classes. Also, the dimension of the hyperplane depends upon the number of features. If the number of input features is 2, then the hyperplane is just a line. If the number of input features is 3, then the hyperplane turns into a two-dimensional plane. For the case of number of features which exceeds 3, it becomes difficult to imagine.

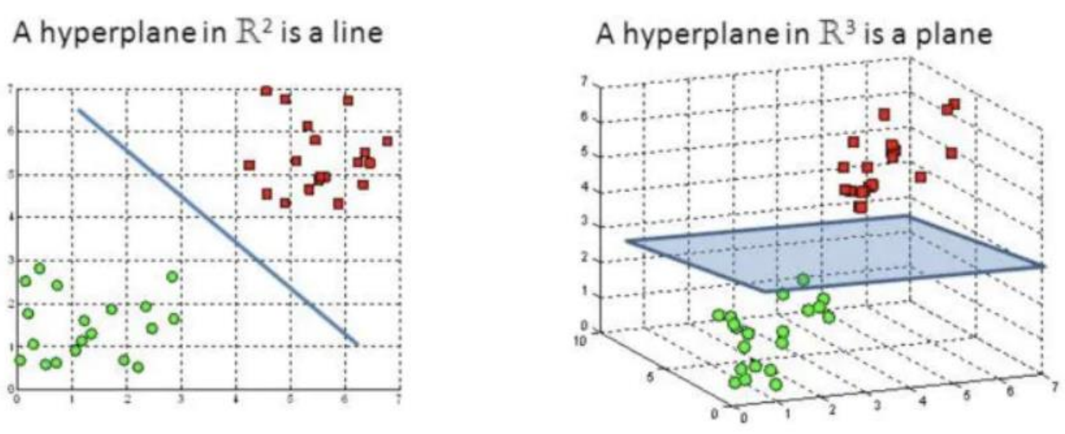


Figure 3: Hyperplane (Goyal, 2020)

Support vectors are data points that are nearer to the hyperplane and affect the position and orientation of the hyperplane. Maximizing the margin of the classifier can be performed by applying these support vectors. Note that deleting the support vectors will make a change of the hyperplane's position.

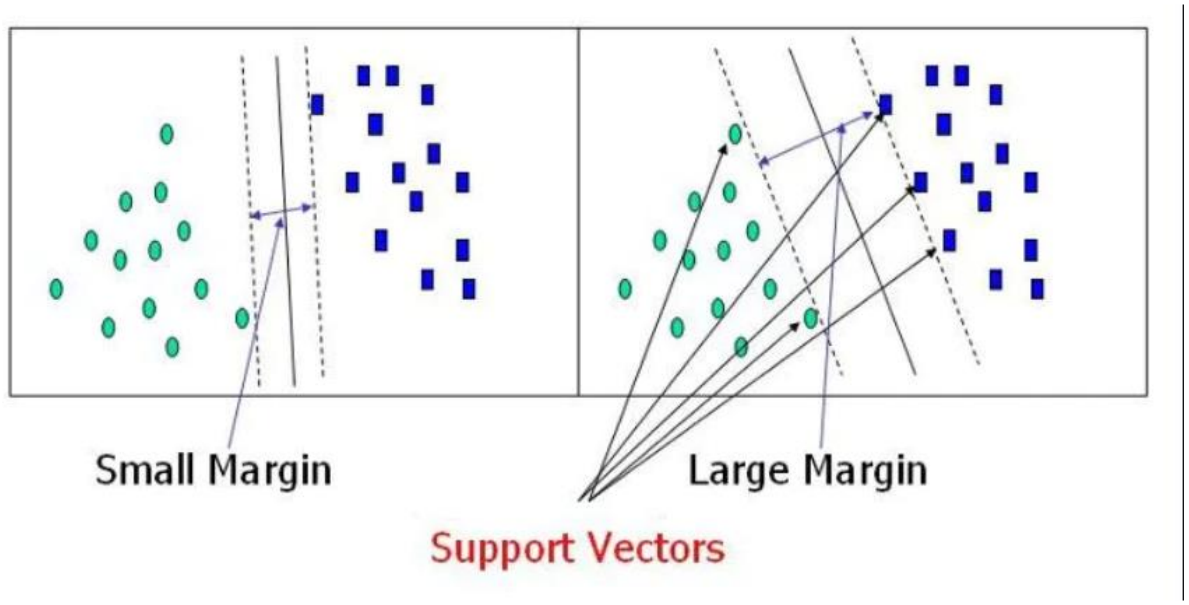


Figure 4: Support Vectors (Goyal 2020)

ii. Binary SVM

The binary SVM can be expanded into multiclass and multiclass SVMs are usually implemented by combining multiples two-class SVMs (John, 2021). The two common approaches are one-versus-all method and one-versus-one method. The first one denotes the earliest and popular multiclass approach applied for SVM. The remaining N-1 classes which have been gathered trains each class. In one-versus-one method, each one is trained on data from two classes. While testing for each class, the max-wins voting strategy is used to classify an unknown sample.

iii. Applying SVM for Handwritten Recognition

The SVM in scikit-learn supports both dense and sparse sample vectors as input. In scikit-learn, SVC, NuSVC and LinearSVC are classes capable of performing multi-class classification on a dataset. Pashine et al. (2021) used LinearSVC for classification of MNIST datasets that make use of a Linear kernel implemented with the help of LIBLINEAR. Various scikit-learn libraries like NumPy, matplotlib, pandas, Sklearn and seaborn have been used for the implementation purpose. Firstly, the MNIST datasets is downloaded followed by loading it and reading those CSV files using pandas. After this, plotting of some samples as well as converting into matrix followed by normalization and scaling of features have been done. Finally, a linear SVM model was developed and confusion matrix that is used to measure the accuracy of the model. The accuracy shows that the model can predict approximately 94% for testing data set.

E. Artificial neural networks

1. CNN

CNN stands for Convolution Neural Network. Convolution refers to twisted or coiled. Any neural network is similar with human brain and neural networks are designed by taking inspiration from brain (Preetha et al.,2020). CNN is commonly applied for Image classification, and it is designed to detect visual patterns straight from pixel images with minimal prepossessing (Das,2017). CNN model is built with multiple layers according to the requirements. Rosyda & Purboyo (2018) described that CNN is generally constructed with three layers, convolution layer, pooling layer, and fully connected layer. However, additional layers can be added, for example, softmax layer which can improve the accuracy of the image recognition. Each layer is connected to the prior layer. Although additional layers are optional, it can affect the performance of the model. As an example, if softmax layer is added, the model can give more accurate result of handwritten recognition than CNN which has basic three layers.

i. Convolution layer

This is a basic layer to builds a CNN model. In this layer, convolution process is presented which is a linear operation that contains the multiplication of a set of weights with the input, similar with conventional neural network (Indolia et al. 2018). The aim of this layer is for two-dimensional input in which multiplication is executed between input data's array and a two-dimensional array of weights, known as a filter or a kernel. This filter tells the pattern that we want to recognize. Convolution layer can be regarded as a application of a filter to an input that causes an activation.

The objective of the Convolution Operation is to remove the high-level features such as edges, from the input image (Saha, 2018). It is not limited to include only one Convolutional Layer in CNN model. Normally, the first Convolution Layer is accountable for capturing the Low-Level features such as edges, colour, gradient orientation, etc. With additional layers, the architecture also adapts to the High-Level features. As a result, a network is obtained which has the complete knowledge of images in the dataset like the way how human would.

ii. Pooling layer

Like the Convolutional Layer, the object of the pooling layer is reducing the spatial size of the convolved feature which means to minimize the computational power needed for processing the data through dimensionality reduction (Saha, 2018). Moreover, it is effective for removing features which are rotational and positional invariant.

There are two kinds of pooling: Max pooling and Average Pooling. Max pooling gives the highest value from the portion of the image contained by Kernel. Contrarily, Average pooling gives average all the values from the portion of the image contained by Kernel.

Max pooling acts like a Noise Suppressant which means it removes the noisy activations and process de-nosing together with dimensionally reduction. On the other hand, Average pooling works dimensionality reduction as noise suppressing process. Therefore, it says that Max Pooling is a lot better than Average Pooling.

iii. Fully connected layer

The main aim of fully connected layer is to take the result of pooling layer and use them to classify the image into a label (Yamashita et al., 2018). The output of pooling is usually matrix. This matrix is converted to a single dimensional array and this procedure is called flattening. The values in vector represents probability of some features of the object.

The output from the convolutional layers represents high-level features in the data. While that output could be flattened and connected to the output layer, adding a fully connected layer is a generally cheap way of learning non-linear combinations of these features.

Basically, the convolutional layers are providing a meaningful, low-dimensional, and somewhat invariant feature space, and the fully connected layer is learning a (possibly non-linear) function in that space.

2. Architecture of CNN

The followings are the popular architecture designs of CNN. LeNet-5 is the oldest one among all and ResNet is the latest one.

i. LeNet-5 (1998)

LeNet-5 is a 7-level convolution network by LeCun et al (1998). This architecture has been used to classify digits and this kind of CNN is applied in banks to detect handwritten numbers on checks/ cheuqes. The most common used dataset for this is MNIST dataset which has 28x28 input grayscale pixel images. It has three main layers like traditional CNN: Convolutional layers, Pooling layers and Fully connected Layer. The difference is that LeNet-5 includes 2 more convolutional layers and one more pooling layer and additional fully connected layer.

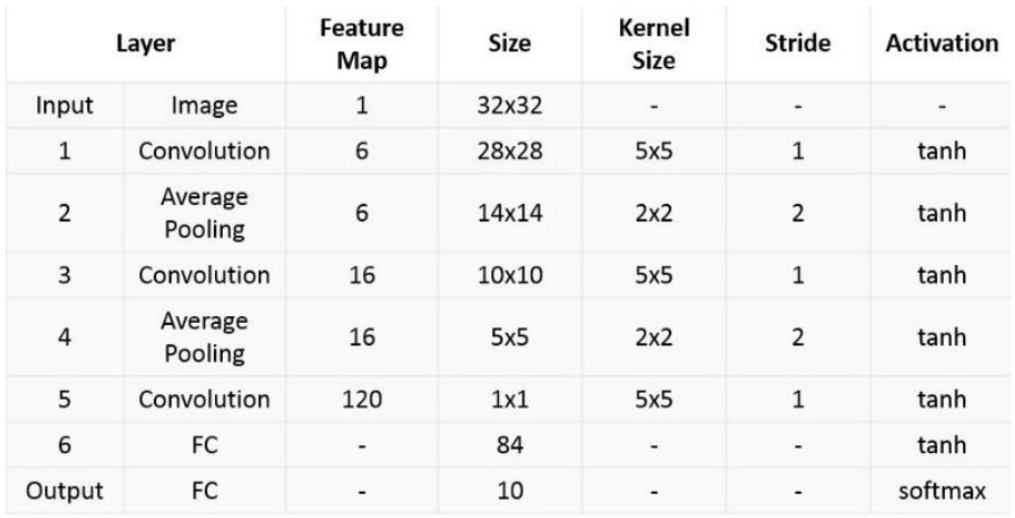
The first layer is the input layer and this is normally not considered as a layer of the network as nothing is learnt in this layer. The input layer is image pixel size 32x32 and that dimensional of image is passed down to another layer. If MNIST dataset is used as a input for LeNet-5, it need to be aware that images in MNIST data set have the pixel size 282x28. To get the MNIST images dimension to the meet the requirements of the input layer, the 28x28 images need to be padded.

The second layer is average pooling layer which is known as a sub-sampling layer with a filter size of 2 and a stride of 2. The output image pixel size is reduced to 14x14. The third layer is the second convolutional layer with 16 feature maps with stripe of 1. In this layer, only 10 out of 16 feature maps are connected to 6 feature maps of the previous layer.

The fourth layer is an average pooling layer with filter size 2×2 and a stride of 2. This layer is the same as the second layer (pooling layer) except it has 16 feature maps so the output will be reduced to 16@5x5.

The firth layer is a fully connected convolutional layer with 120 feature maps each of size 1×1. Each of the 120 units in this layer is connected to all the 400 nodes (5x5x16) in the fourth layer of pooling. The sixth layer is a fully connected layer with 84 units. The last layer is a fully connected softmax output layer with 10 possible values corresponding to the digits from 0 to 9 by using softmax activation function.

The ability to process higher resolution images requires larger and more convolutional layers, so this technique is constrained by the availability of computing resources. This layer is mainly applied to build CNN model in computing community according to literatures. The main reason behind the popularity of this model was its simple and straightforward architecture. It is a multi-layer convolution neural network for image classification as well.



Layer		Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	32x32	-	-	-
1	Convolution	6	28x28	5x5	1	tanh
2	Average Pooling	6	14x14	2x2	2	tanh
3	Convolution	16	10x10	5x5	1	tanh
4	Average Pooling	16	5x5	2x2	2	tanh
5	Convolution	120	1x1	5x5	1	tanh
6	FC	-	84	-	-	tanh
Output	FC	-	10	-	-	softmax

Figure 5: LeNet-5 Architecture Design

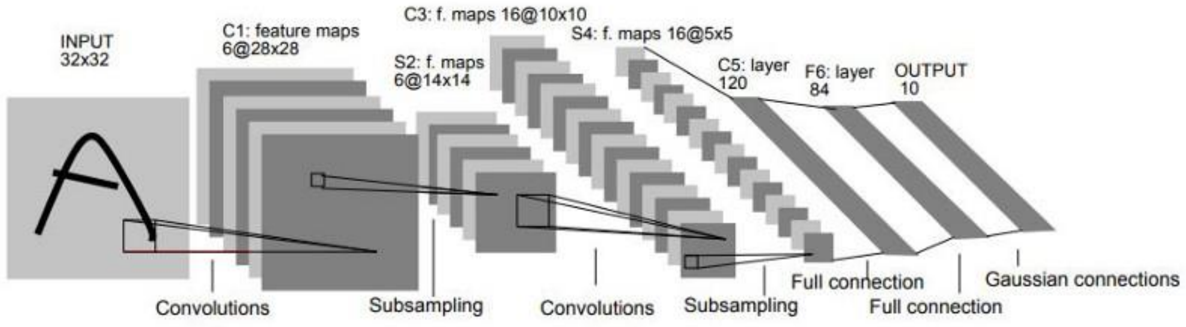


Figure 6: LeNet-5 Architecture Design

ii. AlexNet (2012)

Alexnet won the ImageNet large-scale visual recognition competition in 2012 with good the error rate (Krizhevsky et al.,2012). The second-place top-5 error rate, which was not a CNN variation, was around 26.2%.

The network has a very similar architecture as LeNet-5 (LeCun et al.,1998) but it is deeper, with more filters per layer, and with stacked convolutional layers. It consists of 11x11, 5x5,3x3, convolutions, max pooling, dropout, data augmentation, ReLU activations, SGD with momentum. It oabtins ReLU activations after every convolutional and fully connected layer. AlexNet was trained for 6 days simultaneously on two Nvidia Geforce GTX 580 GPUs which is the reason for why their network is split into two pipelines. AlexNet was designed by the SuperVision group, consisting of Alex Krizhevsky, Geoffrey Hinton, and Ilya Sutskever.

iii. ZFNet(2013)

Likewise, ZFNet achieved that award in 2013 which is also a CNN model. It obtained a top-5 error rate of 14.8% which is now already half of the prior mentioned non-neural error rate. It was mostly an achievement by tweaking the hyper-parameters of AlexNet while maintaining the same structure with additional Deep Learning elements. The design was inspired by visualizing intermediate feature layers and the operation of the classifier. Compared to Alex Net, the filter sizes are reduced and the stride of the convolutions are reduced.

iv. GoogLeNet/Inception(2014)

GoogLeNet won the ILSVRC competition in 2014 which is created by Google and it is also known as Inception V1. It obtained a top-5 error rate of 6.67%. This was exceptionally close to human level performance which the evaluation of the performance had to be done by the organisers of the competition. As it turns out, this was actually rather difficult to do and required some human training in order to beat GoogLeNets accuracy. After a few days of training, the human expert (Andrej Karpathy) was able to achieve a top-5 error rate of 5.1% (single model) and 3.6%(ensemble). The network is used a CNN motivated by LeNet but is implemented a novel element which is named an inception module. It used batch normalization, image distortions and RMSprop. This module depends on several very small convolutions in order to considerably decrease the number of parameters. This architecture design consists of a 22-layer deep CNN but lowered the number of parameters from 60 million (AlexNet) to 4 million.

v. VGGNet (2014)

The runner-up at the ILSVRC 2014 competition is named VGGNet by the community and was developed by Simonyan and Zisserman. It consists of 16 convolutional layers and is very appealing because of its very uniform architecture. Like AlexNet, it has only 3x3 convolutions, but with lots of filters. It was trained on 4 GPUs for 2–3 weeks. It is currently the most preferred choice in the community for extracting features from images (Das, 2017). The weight configuration of the this architecture is publicly available and has been used in many other applications and faces a challenge as a baseline feature extractor. Nevertheless, VGGNet consists of 138 million parameters, which is relatively high to handle and has much challenges.

vi. ResNet(2015)

The last architecture, at the ILSVRC 2015, the so-called Residual Neural Network (ResNet). He et al (2016) introduced a novel architecture with “skip connections” and features heavy batch normalization. Such skip connections are also known as gated units or gated recurrent units and have a strong similarity to recent successful elements applied in RNNs. The technique is very effective that they are able to train a NN with 152 layers while still having lower complexity than VGGNet. It achieves a top-5 error rate of 3.57% which beats human-level performance on this dataset.

VII. Project Aims and Methodology

Handwritten digit recognition is an active topic in OCR applications or pattern classification learning research, and the challenges for it remains exist. In OCR applications, digit recognition is used for postal mail sorting, bank cheque processing etc. For those applications, the accuracy and reliability of digit recognition model is vital. The aim of this project is to develop a handwritten digit recognition system that can identify and determine the handwritten digit from the image with better accuracy by the help of a model which uses a deep learning algorithm called convolutional neural network (CNN) which is a subset of Artificial Neural Network (ANN). The system can accept handwriting images as inputs and then predict which digits it is by the knowledge of the developed model. The goal of the project is that the developed software must be able to give the results of predicted digits by using MNIST dataset. Moreover, the live prediction of the digit is performed which means the user can draw the digit and then the application will predict that written digit by using the developed CNN model.

As mentioned above, CNN is chosen to develop a model for handwritten digit recognition. There are many reasons why CNN is selected to use among other machine learning and deep learning techniques. CNN has several key qualities. The patterns which it learns are translation invariant which means it can still detect the class to which the input belongs even if the input object varies in some way. After learning a certain pattern, CNN can identify it anywhere. Moreover, they can learn spatial hierarchies of the pattern. In the first convolution layer, small local patterns such as edges can be discovered. In the second convolutional layer, larger patterns made from features from the first layer, can be learnt and so on. This allows us to efficiently learn increasingly complex and abstract visual concepts.

According to reviewed papers above in section 4, CNN has been applied in many research papers and it turned it out that CNN can give better accuracy than other algorithms. Chychkarova et al (2021) applied SVM, KNN, RF and CNN on their model but CNN achieved the best accuracy among all with 97.6%. Moreover, Dutt & Dutt (2017) compared CNN with other algorithms and the highest accuracy found in CNN with 98.70 % which is impressive. In addition, Siddique et al (2019) and Hossain & Ali (2019) applied CNN to build the model for handwritten digit recognition and the accuracy of the models are relatively good obtaining between 98-99%. Because of the succussed experiments, CNN is decided to use for model in this project.

To build a CNN model, the design of the model is essential to be decided. There are various kinds of designs that can be applied, for example, LeNet5, AlexNet, GoogLeNet and etc. Among them, LeNet 5 is selected for design of the CNN model. LeNet5 has accomplished the popularity among other designs because it is simple and straightforward architecture. It is also the oldest design so that it is more reliable, flexible, and trusted architecture. Typical LeNet5 design consists of seven layers with 3 convolution layers, 2 sub-sampling (pooling layers) and 1 fully connected layer which are followed by the output layer. Accordig to Yu et al., (2015), El-Sawy et al., (2016) and LeNet-5 was able to achieve error rate below 1% on the MNIST data set and the accuracy is relatively high with approximately between 90-99%. Because of above reasons, LeNet-5 is applied to build CNN model. In this research, LeNet-5 is modified by adding more layers in order to get the better accuracy and the details of the design is discussed in Section 8.

VIII. Analysis and Design

LeNet-5 (LeCun et al., 199) is used as a baseline standard model for designing the convolutional neural network model for handwritten recognition rather than building a model from the scratch. Before discussed about the LeNet-5 model, a very simple of ANN is implemented as a prototype. That model consists of dense and dropout layers. Although that model can give a satisfied level accuracy, the error rate is not the best, so a proper detailed model is required to get the better prediction. Therefore, I have decided to build CNN model with LeNet-5.

A simple convolutional neural network is a sequence of layers, and every layer alters one volume of activations to another through a differentiable function. Three main kinds of layers are used for building the network which are convolutional layers, pooling layers, and fully connected layers. Other layers are added to that design and all details are discussed as below. Figure 7 shows the architecture of the proposed CNN model.

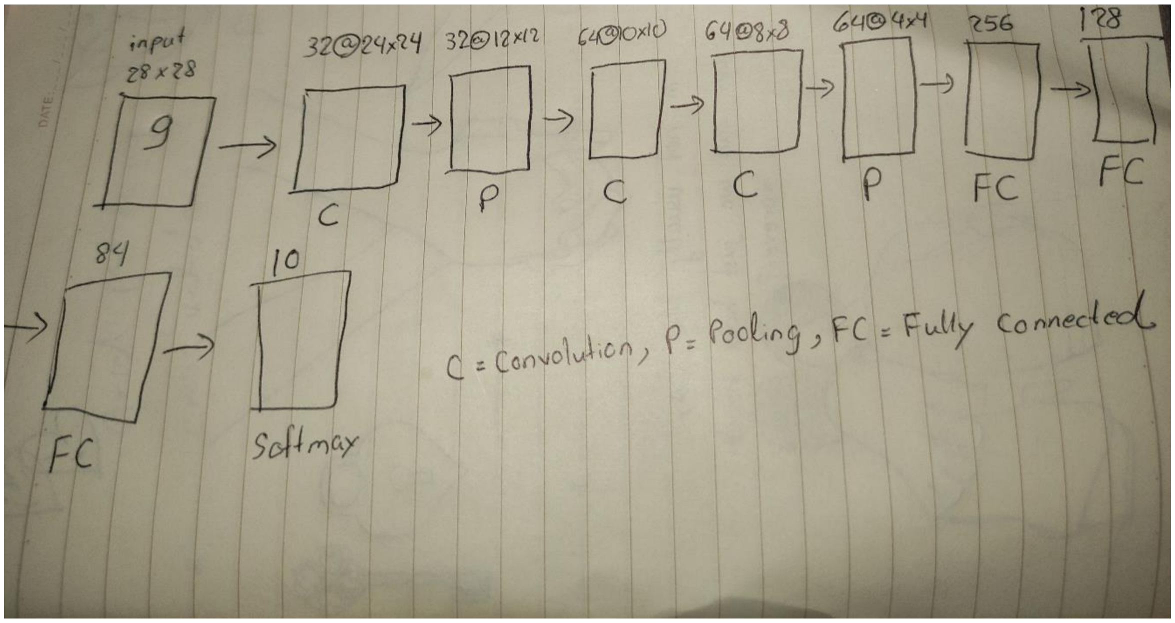


Figure 7: Proposed Model

First of all, some pre-processing procedures on the images like standardization are performed. After the required pre-processing, data can be implied to the model.

The Layer-1 is convolutional layer with ReLu (Rectified Linear Unit) activation function. This is the very first convolutional layer of our CNN design. This layer receives the pre-processed image as the input of n*n = 28*28. The convolution filter size is 5; padding (p) is 0 (around all the sides of the image), stride (s) is 1 and the number of filters is 32. After this process, the feature maps of size 32@24*24 is obtained where 32 is the number of feature maps which is correspond to the number of filters used, and 24 comes from the formula ((n+2p-f)/s)+1=((28+2*0- 5)/1)+1=24. Then Batch Normalization and the ReLu activation are done in each feature map.

Layer-2 is the max pooling layer where the size 32@24*24 from the previous layer is used as an input. The pooling size is 2, padding is 0 and stride is 2. After this max pooling process, the feature maps of size 32@12*12 is attained. Max pooling is done in each feature map independently, as a result the same number feature maps as the previous layer can be achieved, and 12 derives from the same formula ((n+2p-f)/s)+1. This layer does not need to perform activation function.

Layer-3 is the second convolutional layer with ReLu activation function. This layer gets the input of size 32@12*12 from the previous layer. The filter size is 3; padding is 0, the stride is 1 and the number of filters is 64. After this convolution process, the feature maps of size 64@10*10is obtained. Then ReLu activation is done in each feature map.

Layer-4 is the third convolutional layer without ReLu activation function. This layer gets the input of size 64@10*10 from the previous layer. The filter size is 3; padding is 0, the stride is 1 and the number of filters is 64. Then, the feature maps of size 64@8*8 is attained. Then batch normalization and activation function are applied between this layer and next layer.

Layer-5 is the second max pooling layer and it gets the input of size 64@8*8 from the previous layer. The pooling size is 2, padding is 0 and stride is 2. After this operation, the model gives the output feature map of size 64@4*4. Then dropout and flatten operation is performed.

Layer-6 is the first fully connected layer which takes an input one-dimensional vector of size 64 and give output with a one-dimensional vector of size 256. Then batch normalization and activation function are applied between this layer and next layer.

Layer-7 is the second fully connected layer whose input is size 256 from the previous layer and it outputs a one-dimensional vector of size 128. Then batch normalization and activation function are applied between this layer and next layer.

Layer-8 is the third fully connected layer whose input is size 128 from the Layer-7 and it outputs a one-dimensional vector of size 84. Then batch normalization, activation and dropout function are applied between this layer and next layer.

Layer-9 is the last layer of the network which is also the fully connected layer. This layer is for calculating the class scores, contributing to a vector of size 10, where each of the ten numbers relates to a class score, such as among the ten categories of MNIST dataset. It has softmax activation function for final outputs.

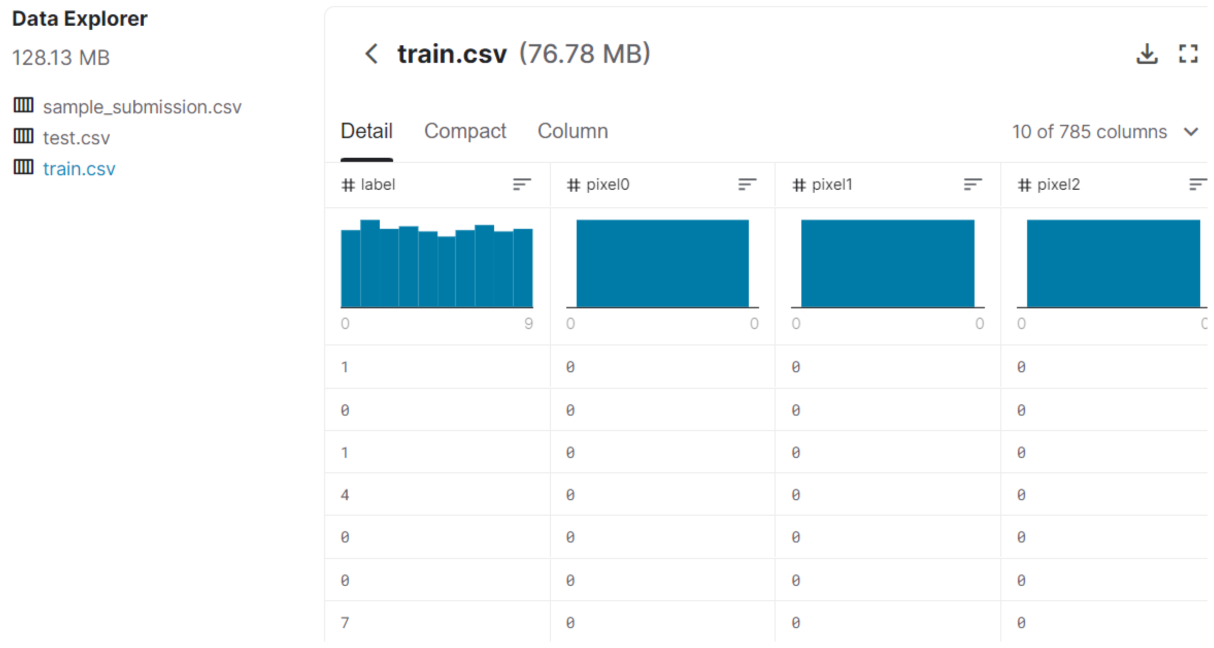
By doing so, CNN model makes changes to the original image which has the original pixel values, with processing layer by layer. Finally, the model can reach the final class scores. It can be found that some layers require parameters and others do not. Particularly, the fully connected layers perform transformations that are a function of not only the activations in the input volume but also of the parameters (the weights and biases of the neurons). On the other hand, the ReLu / pooling layers will implement a fixed function. The parameters in the convolutional / fully connected layers will be trained with stochastic gradient descent algorithm so that the class scores are consistent with the labels in training set for each image. The algorithm will prepare the trained model which will be used to classify the digits present in the test data. Thus, we can classify the digits presents in the images as: Class-0,1,2,3,4,5,6,7,8,9.

IX. Implementation

Convolutional Neural Networks (CNN) are the recent state-of-art architecture for the image classification task. The proposed 2D Convolutional Neural Network (CNN) model is designed using tensor flow and keras libraries with Spyder and Anaconda for the well-known MNIST digit recognition task. The software is implemented by Python Programming Language. The entire system can be to downloading the data, preparing the data, building, and compiling of the model, training and evaluating the model and saving the model to disk to reuse for live handwritten digit prediction.

A. Downloading the data

Before implementing anything, we need to download the MNIST data from the Kaggle by this link (Digit Recognizer | Kaggle). As we can see, there are three files on the website, training data set, testing data set and sample submission file which is not required (optional) to download. In the training data set, the first column in the sheet is label which represents the digit drawn by the user and the other columns are the pixel-values of the related picture (Figure 8). In the testing data set, it only includes pixel-values and labels are not contained (Figue9). Sample submission file will be used for the output of the handwritten digit program which contains the image ID of the testing set and the predicted digits for handwritten digit images. The example structure for it is shown on the website as Figure 10.



Data Explorer
128.13 MB

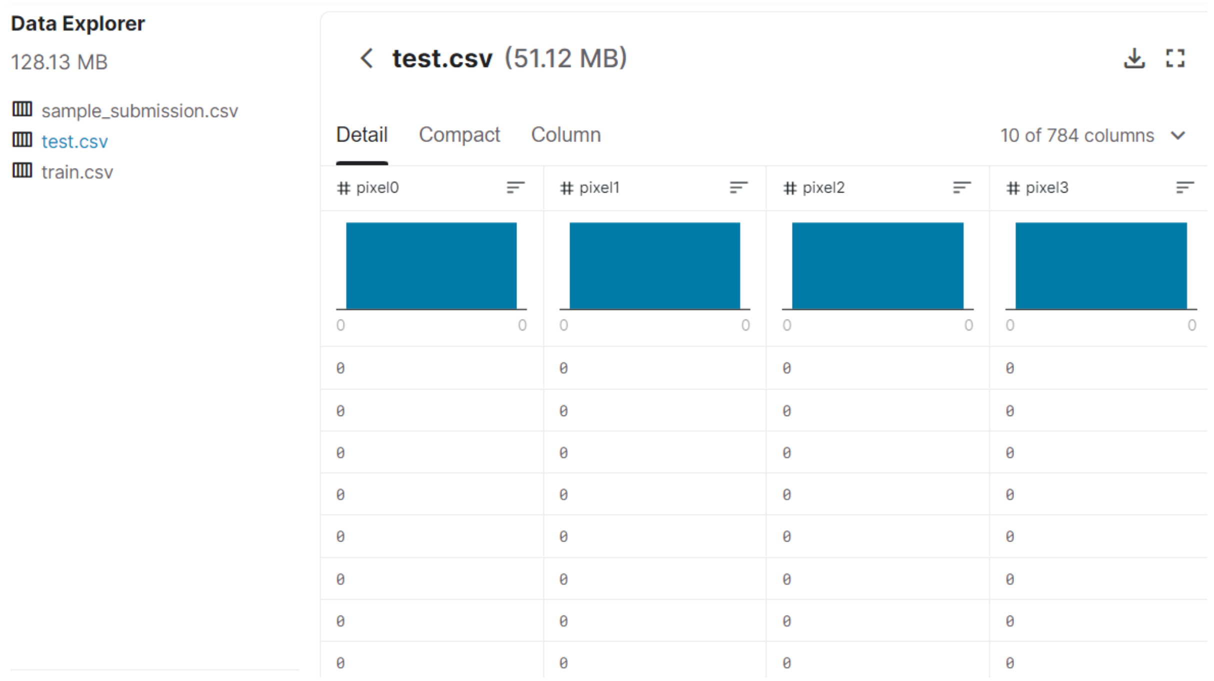
sample_submission.csv
test.csv
train.csv

train.csv (76.78 MB)

Detail Compact Column 10 of 785 columns

label	pixel0	pixel1	pixel2
0 – 9	0 – 0	0 – 0	0 – 0
1	0	0	0
0	0	0	0
1	0	0	0
4	0	0	0
0	0	0	0
0	0	0	0
7	0	0	0

Figure 8: Training data set



Data Explorer
128.13 MB

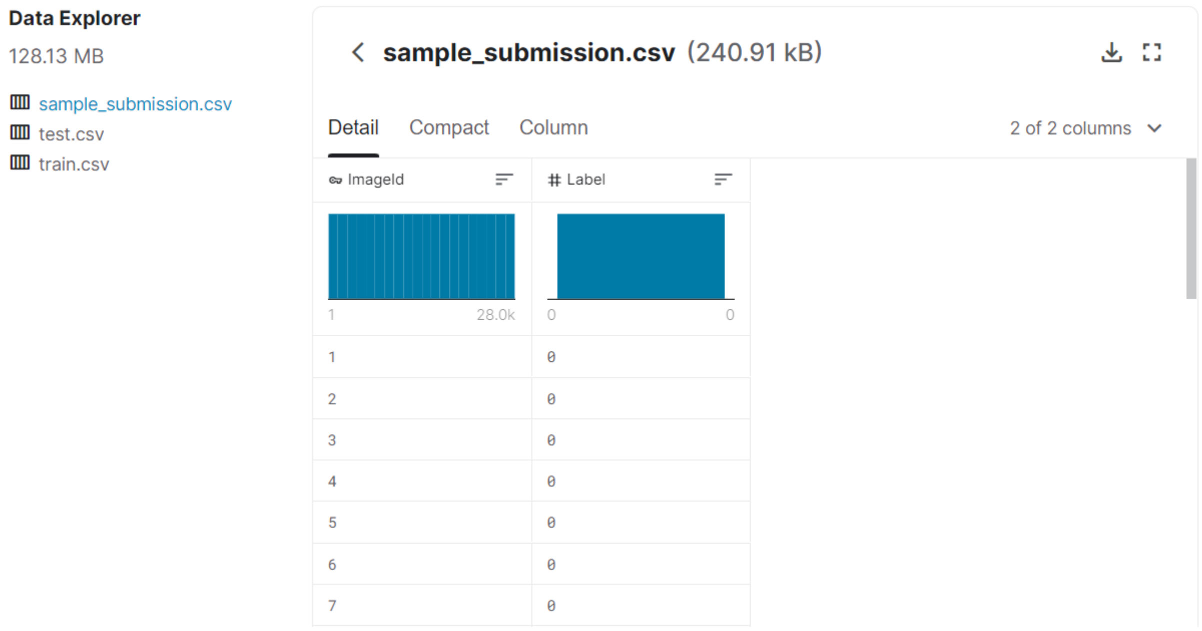
sample_submission.csv
test.csv
train.csv

test.csv (51.12 MB)

Detail Compact Column 10 of 784 columns

pixel0	pixel1	pixel2	pixel3
0 – 0	0 – 0	0 – 0	0 – 0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

Figure 9: Testing data set


Data Explorer

128.13 MB

sample_submission.csv
test.csv
train.csv

sample_submission.csv (240.91 kB)

Detail Compact Column 2 of 2 columns

ImageId	Label
1	0
2	0
3	0
4	0
5	0
6	0
7	0

Figure 10: Sample submission file

B. Loading and Preparing the data

After downloading the data, data is loaded and read. Preparing the data is the very first stage of our approach. Before we construct the network, we need to set up the training, testing, and validating data, combine data, combine labels, and reshape into the appropriate size. Data arrays is reshaped to have a single colour channel as shown in Figure 11. Then, the data is divided into the training set and validating data set by using random cross-validation for validation purposes as shown in Figure 12. Cross validation is one of the techniques used to test the effectiveness of a model whether the model is under/over fitting or well generalized.

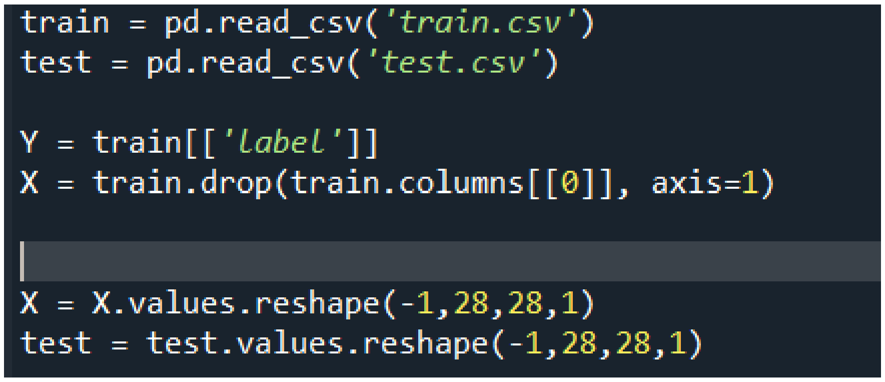
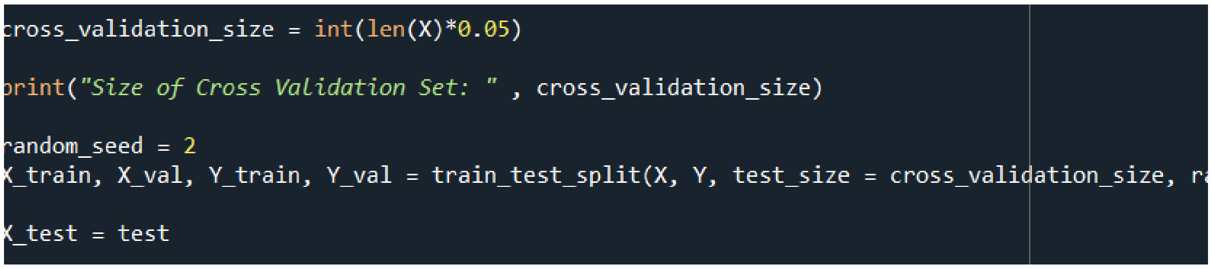

```
train = pd.read_csv('train.csv')
test = pd.read_csv('test.csv')

Y = train[['label']]
X = train.drop(train.columns[[0]], axis=1)


X = X.values.reshape(-1,28,28,1)
test = test.values.reshape(-1,28,28,1)
```

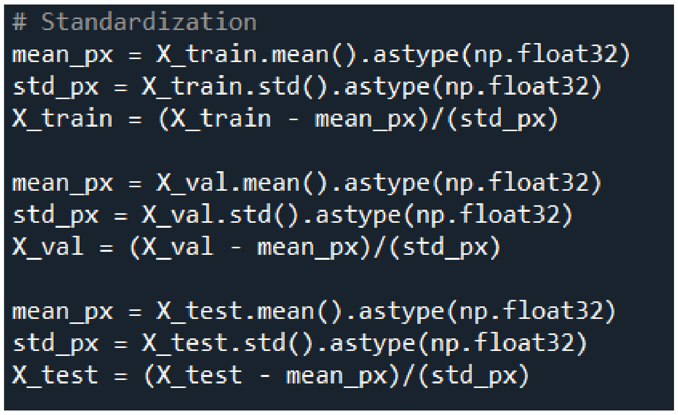
Figure 11: Reshaping the data



```
ross_validation_size = int(len(X)*0.05)

rint("Size of Cross Validation Set: " , cross_validation_size)

andom_seed = 2
_train, X_val, Y_train, Y_val = train_test_split(X, Y, test_size = cross_validation_size, r

_test = test
```

Figure 12: Cross validation

All the images are standardized by subtracting the mean activity over the training data set from each pixel and separating by their standard deviation. Every image consists of pixel values between 0 and 255. If standardization is not applied, when the raw pixel values are applied to the network, it can slow down the learning because of the inconsistent input. After standardization, images are much better quality than before as shown in Figure 14.



```
# Standardization
mean_px = X_train.mean().astype(np.float32)
std_px = X_train.std().astype(np.float32)
X_train = (X_train - mean_px)/(std_px)

mean_px = X_val.mean().astype(np.float32)
std_px = X_val.std().astype(np.float32)
X_val = (X_val - mean_px)/(std_px)

mean_px = X_test.mean().astype(np.float32)
std_px = X_test.std().astype(np.float32)
X_test = (X_test - mean_px)/(std_px)
```

Figure 13: Standardization

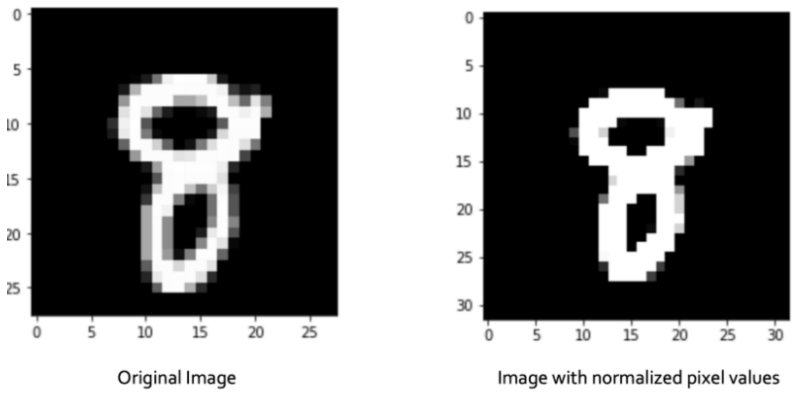
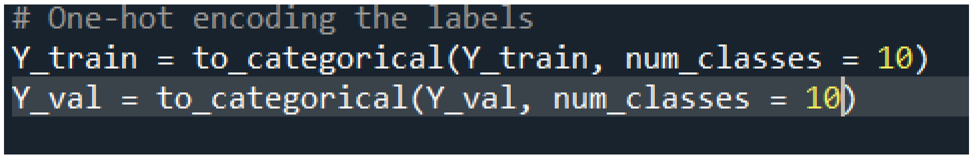


Figure 14: Standardization

The next step is using a one hot encoding for the class element of each sample, converting the interger into 10 element binary vector with a 1 for the index of the class value and 0 for all other classes. This is done by using to_categorical() utility function.



```
# One-hot encoding the labels
Y_train = to_categorical(Y_train, num_classes = 10)
Y_val = to_categorical(Y_val, num_classes = 10)
```

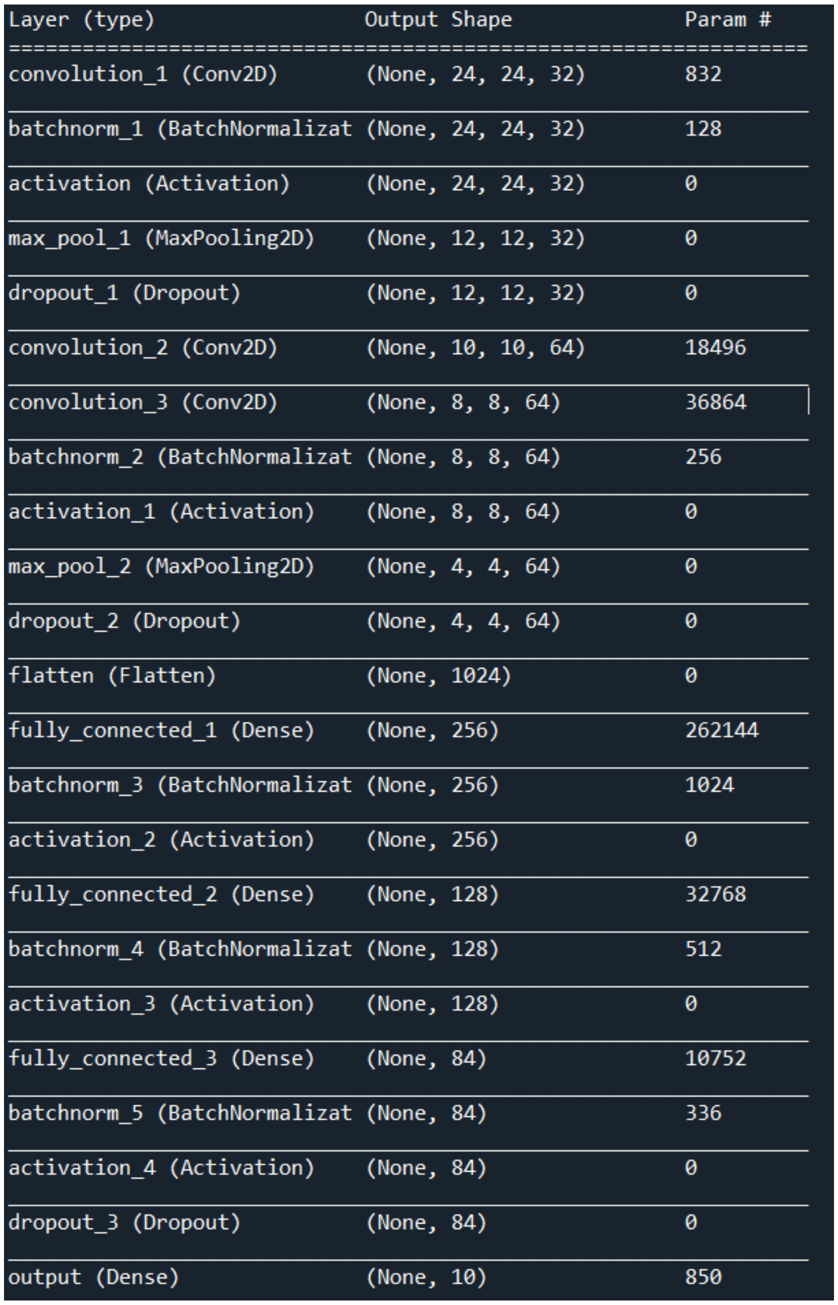
Figure 15: One hot encoding

C. Building the model

As detail explained in Section 8, LeNet-5 designed CNN sequential model is built. The original LeNet-5 network consist of two convolutional layers. In this proposed network, one more convolutional layer is added with the same hyperparameters. The number of filters in the convolutional layers are increased significantly from 6 to 32 in the first layer and 16 to 64 in the next two layers compared with the original LeNet-5 design. Moreover, the number of hidden in the dense layers are increased to accommodate the larger input due to the increase in the volume of the convolutional layers.

The developed model has two main characteristics: the feature extraction front end consists of convolutional and pooling layers, and the classifier backend which will perform a prediction. In the first convolutional layer, the input size 28*28 shapes are implied with 10 classes which is 0-9 digits. Then pooling layer is performed. After that, another two convolutional layers are implemented in the design and another pooling layer is inserted. Next, three fully connected layers are built. The last layer is the output layer with 10 units for identifying the probability distribution of an image corresponding to each of the 10 classes. This is implemented with softmax activation function.

To summarise, three convolutional layers are used, followed by a pooling layer twice (once with 32 filters and 64 filters respectively) and three fully connected layers with a softmax unit in the end with 10 classes. Batch Normalization, L2 Regularization and Dropouts are done in-between the layers. The Figure 16 shows the layer structures of the model and output shape for each layer.



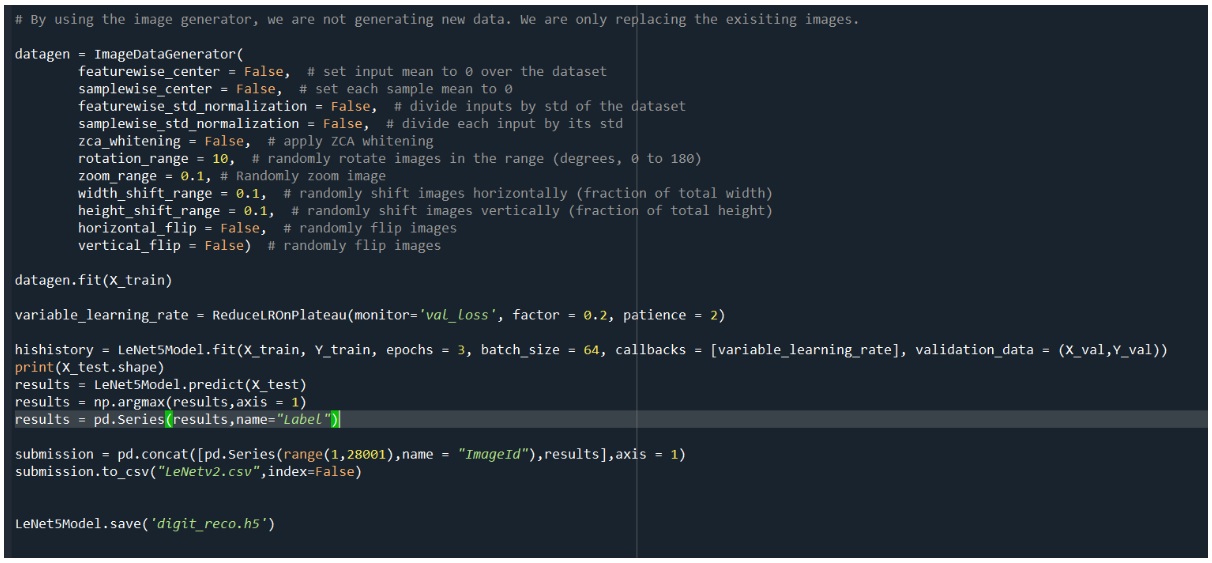
```
Layer (type)                 Output Shape              Param #
=================================================================
convolution_1 (Conv2D)       (None, 24, 24, 32)        832
_________________________________________________________________
batchnorm_1 (BatchNormalizat (None, 24, 24, 32)        128
_________________________________________________________________
activation (Activation)      (None, 24, 24, 32)        0
_________________________________________________________________
max_pool_1 (MaxPooling2D)    (None, 12, 12, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 12, 12, 32)        0
_________________________________________________________________
convolution_2 (Conv2D)       (None, 10, 10, 64)        18496
_________________________________________________________________
convolution_3 (Conv2D)       (None, 8, 8, 64)          36864
_________________________________________________________________
batchnorm_2 (BatchNormalizat (None, 8, 8, 64)          256
_________________________________________________________________
activation_1 (Activation)    (None, 8, 8, 64)          0
_________________________________________________________________
max_pool_2 (MaxPooling2D)    (None, 4, 4, 64)          0
_________________________________________________________________
dropout_2 (Dropout)          (None, 4, 4, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 1024)              0
_________________________________________________________________
fully_connected_1 (Dense)    (None, 256)               262144
_________________________________________________________________
batchnorm_3 (BatchNormalizat (None, 256)               1024
_________________________________________________________________
activation_2 (Activation)    (None, 256)               0
_________________________________________________________________
fully_connected_2 (Dense)    (None, 128)               32768
_________________________________________________________________
batchnorm_4 (BatchNormalizat (None, 128)               512
_________________________________________________________________
activation_3 (Activation)    (None, 128)               0
_________________________________________________________________
fully_connected_3 (Dense)    (None, 84)                10752
_________________________________________________________________
batchnorm_5 (BatchNormalizat (None, 84)                336
_________________________________________________________________
activation_4 (Activation)    (None, 84)                0
_________________________________________________________________
dropout_3 (Dropout)          (None, 84)                0
_________________________________________________________________
output (Dense)               (None, 10)                850
```

Figure 16: LeNet-5 model

The model is compiled by using Adam classifier.

D. Training the model

Before training the model, the image generator is used for purpose of replacing the existing images. It allows us to augment the images in real-time while the model is still training. After that, the model is trained by epochs 10 with batch size 64. Call-back function ReduceLROnPlateau is used to reduce the larin grade when a metric has stopped improving. Then, model performs predication using testing data set. Next step is putting predicted digits into excel file with Image ID and description of digits. Then, the model is saved for purpose of reusing it so that we do not need to run the whole model again and it is saving much time. This process is shown as below.

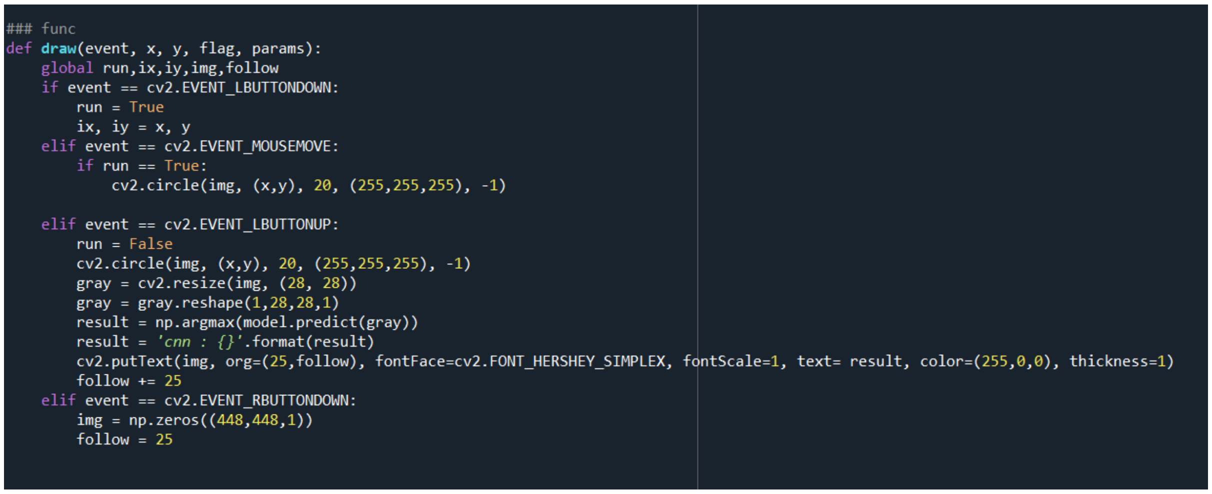


```
# By using the image generator, we are not generating new data. We are only replacing the exisiting images.

datagen = ImageDataGenerator(
        featurewise_center = False,  # set input mean to 0 over the dataset
        samplewise_center = False,  # set each sample mean to 0
        featurewise_std_normalization = False,  # divide inputs by std of the dataset
        samplewise_std_normalization = False,  # divide each input by its std
        zca_whitening = False,  # apply ZCA whitening
        rotation_range = 10,  # randomly rotate images in the range (degrees, 0 to 180)
        zoom_range = 0.1, # Randomly zoom image
        width_shift_range = 0.1,  # randomly shift images horizontally (fraction of total width)
        height_shift_range = 0.1,  # randomly shift images vertically (fraction of total height)
        horizontal_flip = False,  # randomly flip images
        vertical_flip = False)  # randomly flip images

datagen.fit(X_train)

variable_learning_rate = ReduceLROnPlateau(monitor='val_loss', factor = 0.2, patience = 2)

hishistory = LeNet5Model.fit(X_train, Y_train, epochs = 3, batch_size = 64, callbacks = [variable_learning_rate], validation_data = (X_val,Y_val))
print(X_test.shape)
results = LeNet5Model.predict(X_test)
results = np.argmax(results,axis = 1)
results = pd.Series(results,name="Label")

submission = pd.concat([pd.Series(range(1,28001),name = "ImageId"),results],axis = 1)
submission.to_csv("LeNetv2.csv",index=False)


LeNet5Model.save('digit_reco.h5')
```

Figure 17: Training the model

E. The real-time handwritten digit recognition program using the developed model

The first step is that the implemented model is loaded for this program. Then a canvas with black background size 448* 448 is created. In this program, a white pen is used to draw the digits on the black background canvas like the way images in MIST dataset. Open cv library is used for detecting mouse motion. Mouse click events are captured by cv2. Image size is reshaped into 28*28 in order to fit in to the model. In summary, it captures the x-y coordinates of the mouse pointer between the time a digit is drawn and the time the left mouse button is released. For all these x-y points captured, lines are drawn to connect them creating a figure on the path we made with the mouse pointer. cv2. circle function is used to draw a circle on a image (canvas). To clear the canvas/board, left mouse button is required to clicked and Esc key should be used to close or end the program. This process of drawing the digit is implemented shown below.



```
### func
def draw(event, x, y, flag, params):
    global run,ix,iy,img,follow
    if event == cv2.EVENT_LBUTTONDOWN:
        run = True
        ix, iy = x, y
    elif event == cv2.EVENT_MOUSEMOVE:
        if run == True:
            cv2.circle(img, (x,y), 20, (255,255,255), -1)

    elif event == cv2.EVENT_LBUTTONUP:
        run = False
        cv2.circle(img, (x,y), 20, (255,255,255), -1)
        gray = cv2.resize(img, (28, 28))
        gray = gray.reshape(1,28,28,1)
        result = np.argmax(model.predict(gray))
        result = 'cnn : {}'.format(result)
        cv2.putText(img, org=(25,follow), fontFace=cv2.FONT_HERSHEY_SIMPLEX, fontScale=1, text= result, color=(255,0,0), thickness=1)
        follow += 25
    elif event == cv2.EVENT_RBUTTONDOWN:
        img = np.zeros((448,448,1))
        follow = 25
```

Figure 18: Drawing handwritten digits

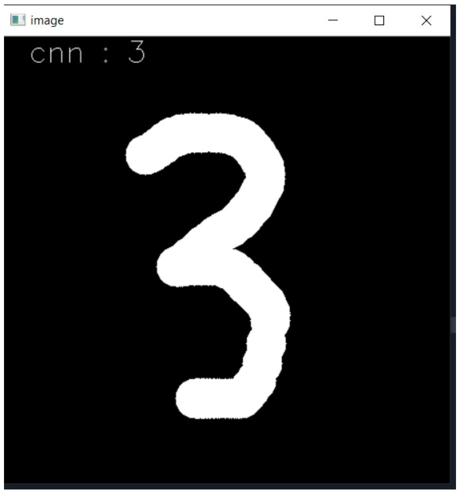


Figure 19: Live Handwritten Digit Recognition Using LeNet-5 CNN model

X. Tools and Libraries

Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics (Python.). Python is simple, easy to learn syntax emphasizes readability and therefore it can reduce the cost of program maintenance. Python is the best fit for creating machine learning models because of its independent platform and its reputation in the programming community. Python 3.8 is used in this development of the handwritten recognition system.

TensorFlow

TensorFlow is an end-to-end open-source platform for machine learning purposes. It has a broad, flexible and reliable tools, libraries and community resources that allows researchers push the state-of-the-art in ML and developers effortlessly build and employ ML powered applications (Tensor flow, n.d.).

Keras

Keras is an open-source online library which provides a Python interface for artificial neural networks. Keras peformas as an interface for the TensorFlow library (Keras, n.d.).

Spyder

Spyder is a free and open source working environment written in Python, for Python. It features a exceptional combination of the advanced editing, analysis, debugging, and profiling functionality of a comprehensive development tool with the data exploration, interactive execution, deep inspection, and beautiful visualization capabilities of a systematic package (Spyder, n.d.).

Anaconda Navigator

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda® distribution that lets you to run software and effortlessly manage conda packages, environments, and channels without using command-line commands (Anaconda, n.d.).

Matplotlib

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python (Matplotlib,n.d.). It is used in this implemented system in order to demonstrate the graphs and plots for showing the performance of the LeNet-5 CNN model.

Numpy

Numpy is one of the most used packages for logical computing in Python. It provides not only a multidimensional array object but also variations such as masks and matrices, which can be used for various math operations (Numpy, n.d.).

Pandas

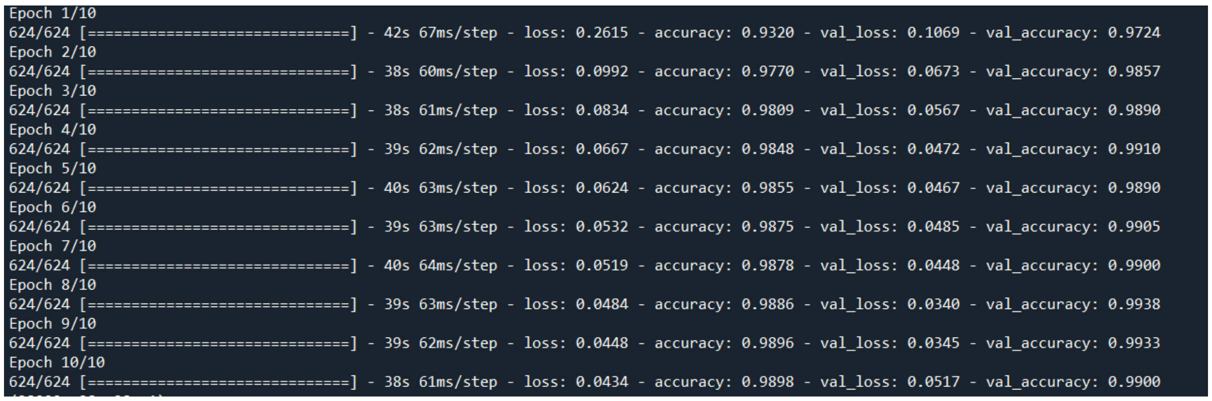
Pandas is a quick, powerful, flexible and easy to use open-source data analysis and manipulation tool, built based on Python programming language (Pandas, n.d). Pandas is used to read the MNIST dataset in this experiment.

Scikit-learn

Scikit-learn is a simple and effective tool for prediction of data purposes. It is open-source and built on NumPy, SciPy and matplotlib (Scikit-learn,n.d.).

XI. Results and Discussion

After all implementation processes, the model needs to be evaluated to check performance, accuracy, and error rate. A range of deep learning and machine learning evaluation methods are applied to evaluate the model. The accuracy of the model is presented as below in Figure 20, achieving 99% which is relatively high. The accuracy of the model becomes better and higher in each epoch,



```
Epoch 1/10
624/624 [==============================] - 42s 67ms/step - loss: 0.2615 - accuracy: 0.9320 - val_loss: 0.1069 - val_accuracy: 0.9724
Epoch 2/10
624/624 [==============================] - 38s 60ms/step - loss: 0.0992 - accuracy: 0.9770 - val_loss: 0.0673 - val_accuracy: 0.9857
Epoch 3/10
624/624 [==============================] - 38s 61ms/step - loss: 0.0834 - accuracy: 0.9809 - val_loss: 0.0567 - val_accuracy: 0.9890
Epoch 4/10
624/624 [==============================] - 39s 62ms/step - loss: 0.0667 - accuracy: 0.9848 - val_loss: 0.0472 - val_accuracy: 0.9910
Epoch 5/10
624/624 [==============================] - 40s 63ms/step - loss: 0.0624 - accuracy: 0.9855 - val_loss: 0.0467 - val_accuracy: 0.9890
Epoch 6/10
624/624 [==============================] - 39s 63ms/step - loss: 0.0532 - accuracy: 0.9875 - val_loss: 0.0485 - val_accuracy: 0.9905
Epoch 7/10
624/624 [==============================] - 40s 64ms/step - loss: 0.0519 - accuracy: 0.9878 - val_loss: 0.0448 - val_accuracy: 0.9900
Epoch 8/10
624/624 [==============================] - 39s 63ms/step - loss: 0.0484 - accuracy: 0.9886 - val_loss: 0.0340 - val_accuracy: 0.9938
Epoch 9/10
624/624 [==============================] - 39s 62ms/step - loss: 0.0448 - accuracy: 0.9896 - val_loss: 0.0345 - val_accuracy: 0.9933
Epoch 10/10
624/624 [==============================] - 38s 61ms/step - loss: 0.0434 - accuracy: 0.9898 - val_loss: 0.0517 - val_accuracy: 0.9900
```

Figure 20: Accuracy of the model

Figure 21 shows that the training and validation accuracy for each epoch (10 epochs). We have the training data set and validation data set which is obtained by using cross-validation technique. The model has been trained by training data and checked its performance on both the training and validation sets. The graph shows that the training and validation accuracies are correspond to one another which means that the performance of the model is great and does not have overfitting problems. A model is stated to overfit when it performs well on the training set but the performance drops on the validation set (or unseen data). Moreover, we can see that the lines become slightly higher than the progress of first few epochs. This means the accuracy of the model becomes higher throughout 10 epochs.

Figure 22 shows the comparison of training loss and validation loss. The graph can be said that the model is a good fit because the plot of the training loss and validation loss decreases to a stage of stability and there is only a small gap between them. A good fit is the objective of the learning algorithm and exists between an overfit and underfit model.

It can be stated that the developed model is effective and well-performed model according to Figure 23 which shows that the loss value is low which is great, and the accuracy of testing data achieved 99% which is comparatively high. Furthermore, Figures 24 and 25 show that the model can correctly predict all images except one and that image should be 2 but the model predicted it as 6. In general, although it can detect some bad handwriting digits, there could be some cases that the prediction of the implemented model is wrong, but it is less common.

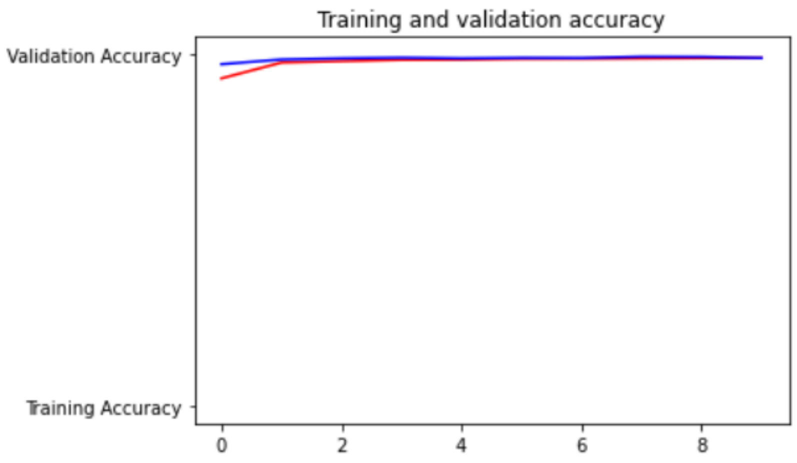


Figure 21: Training and validation accuracy

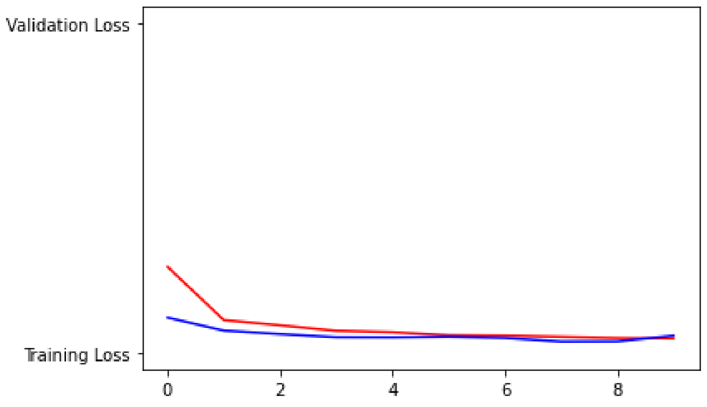
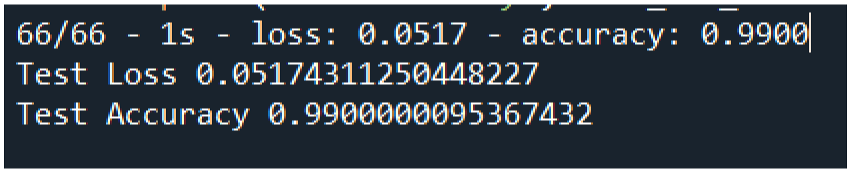
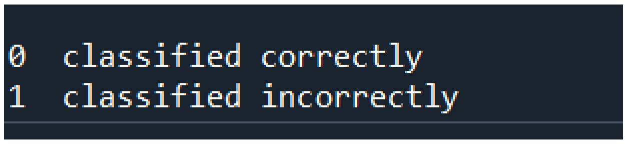


Figure 22: Validation and Training Loss



```
66/66 - 1s - loss: 0.0517 - accuracy: 0.9900
Test Loss 0.05174311250448227
Test Accuracy 0.9900000095367432
```

Figure 23: Test loss and accuracy



```
0  classified correctly
1  classified incorrectly
```

Figure 24: Accuracy of the model

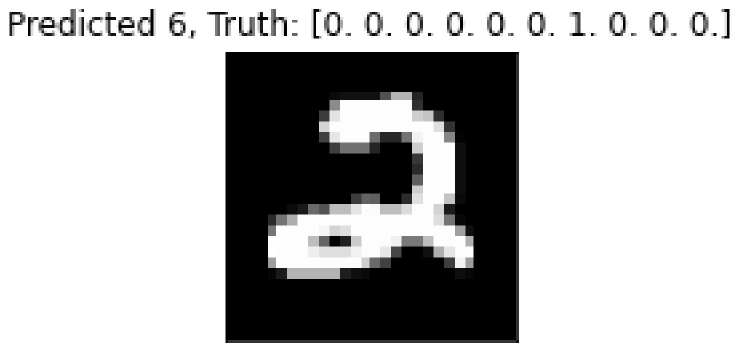
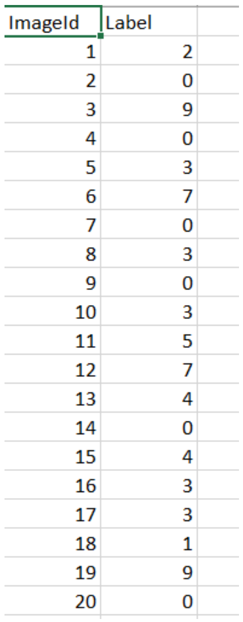


Figure 25: Wrong Prediction

The output of the model is created by excel format with image ID and the predicted digits for handwritten digit images from MNIST data. It can be said that the accuracy of the model is relatively high due to the results shown as in Figure 26. As we can see, the first image of the testing data set from MINIST should be digit 2 and the second image should be digit 0. In Figure 27, the first image and the second image is retrieved from the data set and it is clearly shown that the prediction of the model is correct.



ImageId	Label
1	2
2	0
3	9
4	0
5	3
6	7
7	0
8	3
9	0
10	3
11	5
12	7
13	4
14	0
15	4
16	3
17	3
18	1
19	9
20	0

Figure 26: Output

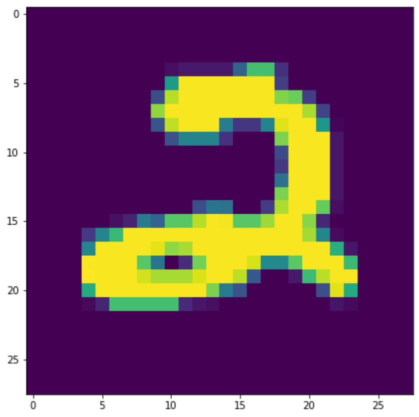


Figure 27: Digit of Image ID 1

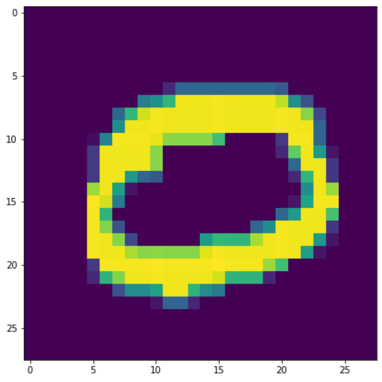
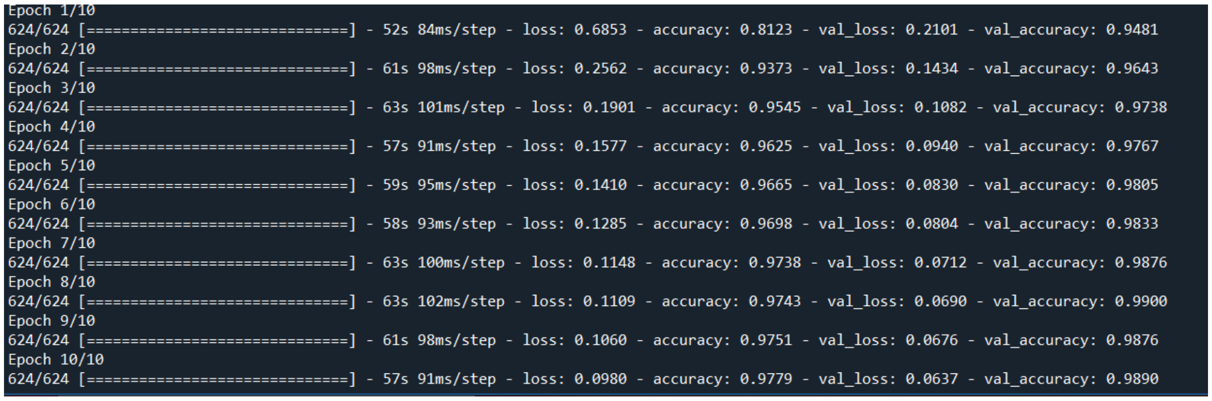


Figure 28: Digit of Image ID 2

The two classifiers Adam and SGD are applied in order to evaluate the results of the model. The results of both does not show much difference because both are powerful and effective classifiers. However, it turns out that

Adam performs better accuracy for the model than SGD, resulting in 99% corresponding to 98%. Moreover, SGD is not preferable to use because our data set is large and SGD can cause higher computation cost in real time due to its process of updating parameters by selecting one sample randomly.



```
Epoch 1/10
624/624 [==============================] - 52s 84ms/step - loss: 0.6853 - accuracy: 0.8123 - val_loss: 0.2101 - val_accuracy: 0.9481
Epoch 2/10
624/624 [==============================] - 61s 98ms/step - loss: 0.2562 - accuracy: 0.9373 - val_loss: 0.1434 - val_accuracy: 0.9643
Epoch 3/10
624/624 [==============================] - 63s 101ms/step - loss: 0.1901 - accuracy: 0.9545 - val_loss: 0.1082 - val_accuracy: 0.9738
Epoch 4/10
624/624 [==============================] - 57s 91ms/step - loss: 0.1577 - accuracy: 0.9625 - val_loss: 0.0940 - val_accuracy: 0.9767
Epoch 5/10
624/624 [==============================] - 59s 95ms/step - loss: 0.1410 - accuracy: 0.9665 - val_loss: 0.0830 - val_accuracy: 0.9805
Epoch 6/10
624/624 [==============================] - 58s 93ms/step - loss: 0.1285 - accuracy: 0.9698 - val_loss: 0.0804 - val_accuracy: 0.9833
Epoch 7/10
624/624 [==============================] - 63s 100ms/step - loss: 0.1148 - accuracy: 0.9738 - val_loss: 0.0712 - val_accuracy: 0.9876
Epoch 8/10
624/624 [==============================] - 63s 102ms/step - loss: 0.1109 - accuracy: 0.9743 - val_loss: 0.0690 - val_accuracy: 0.9900
Epoch 9/10
624/624 [==============================] - 61s 98ms/step - loss: 0.1060 - accuracy: 0.9751 - val_loss: 0.0676 - val_accuracy: 0.9876
Epoch 10/10
624/624 [==============================] - 57s 91ms/step - loss: 0.0980 - accuracy: 0.9779 - val_loss: 0.0637 - val_accuracy: 0.9890
```

Figure 29: Adam vs SGD

On the other hand, the live prediction of handwritten digit recognition is developed. According to Figure 30-34, the prediction is accurate, and the model is compatible and effective with the software. However, the software has some failures, for example, in some attempts, the prediction is not correct as shown in Figure 34, due to the bad handwriting style, or it is not clear. But most of the time, the application can even perform detection of bad handwritten digits, which is impressive as shown in Figure 31.

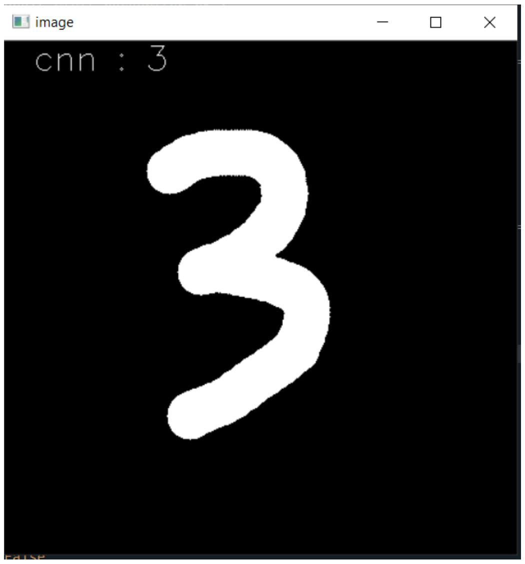


Figure 30: Live Handwritten Digit Recognition

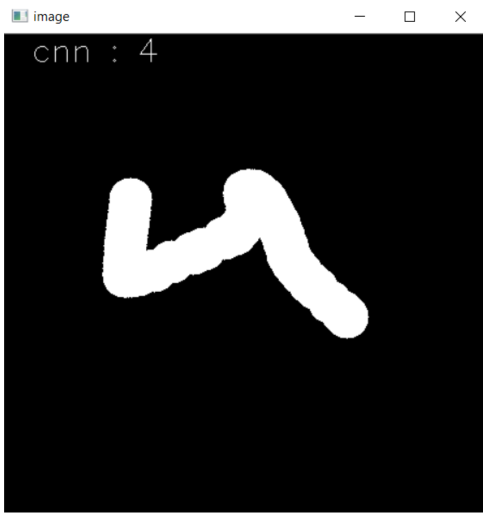


Figure 31: Live Handwritten Digit Recognition

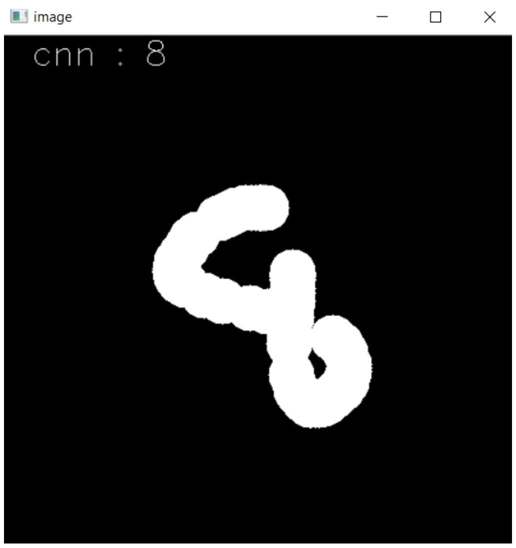


Figure 32: Live Handwritten Digit Recognition

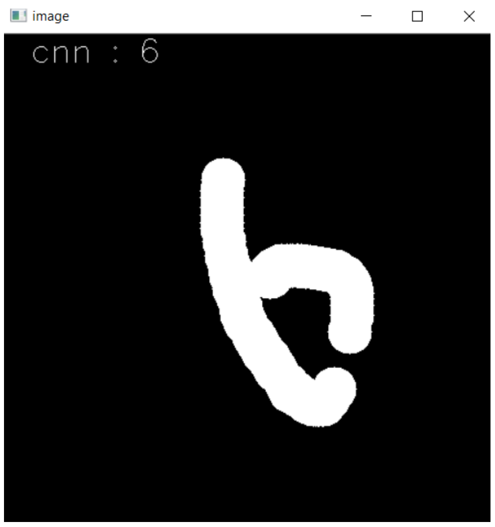


Figure 33: Live Handwritten Digit Recognition

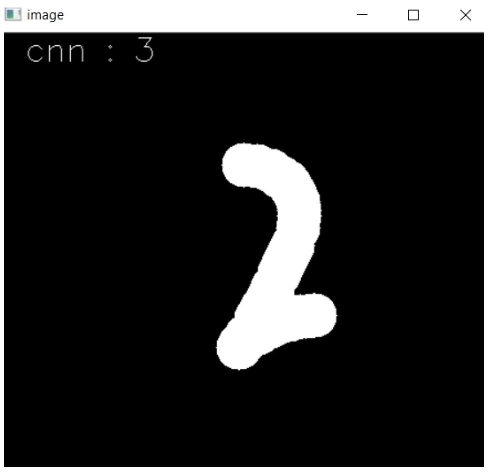


Figure 34: Live Handwritten Digit Recognition

XII. Conclusion and Further Works

In conclusion, a CNN model by LeNet-5 which can recognize handwritten digits is successfully implemented followed by the program which can give live predication of handwritten digit. The objective of this research is to develop a system that helps in the classification and recognition of Handwritten digits. Recognition of characters and digit is essential in today's digitized world, especially in businesses and companies which are dealing with handwriting papers and documents every day and it consumes a lot of time and energy. The only way to solve that problem is to analyse those handwritten documents using computer systems which are utilised by Machine Learning or Deep Learning algorithms. Those systems can solve for classification problems and can be used for handwritten digit recognition resulting in solving complex tasks for organizations or individuals. The current system used neural networks (LeNet-5 CNN model) to process and read handwriting digits. The phases of handwriting recognition included image acquisition, digitization, pre-processing, segmentation, feature extraction, and recognition. All the process are implemented in our developed model.

As seen from the results of the experiment, CNN proves to be far better than other classifiers. The results can be made more accurate with more convolution layers and a greater number of hidden neurons. ANN model has been developed as a basic line model before developing CNN model. It turns out that the accuracy of the CNN model is way higher than that of ANN model. The ANN model only includes dense layers and dropout layers while CNN model consists of 3 convolutional layers, 2 pooling layers and 2 fully connected layers. Obviously, the predication of CNN is better than ANN, with 99% corresponding to 97%. Different classifiers, Adam and SGD has used to evaluate the model and the results show that the prediction of Adam is better than SGD for our proposed model. A number of plots like training and validation accuracy or loss is presented, and it shows that the model has no problems of overfitting or underfitting due to its correspondence between training and validation. Thus, the developed model is a good fit resulting 99% accuracy. Moreover, the model gives only one classification wrong, and it gives perfect results.

The system benefited from Convolution Neural Networks (CNN) with the help of training data that allowed easy recognition of characters and digits. Like the human visual system, CNN allowed the OCR system to be more sensitive to different features of objects. By doing so, it is easy to classify and recognize different Handwriting digits based on the training data stored in the system's database. The significant advantage of the developed system is real-time prediction for handwritten digits. The software works very well and can predict all numbers ranging from 0-9. In addition, it can also detect bad handwriting styles and can perform very well. Furthermore, the run time of proposed model is fast and flexible, and the model is saved after training which can save time and we do not need to run the model again and again, just once! On the other hand, the program is still facing some challenges and issues. While the program works well for detecting images from MNIST dataset, it has some errors and bugs for real-time prediction. For some cases when the handwriting style is appalling, the software cannot identify which digit it is. In my opinion, if we write 20 digits to test the program, the program can give correct answers 17 out of 20. Moreover, I suppose that another model by using other deep learning algorithms should be implemented so that we can compare the results between CNN model and that model then we can choose the better model for developing the application.

The developed system can be improved for better performance and prediction. It would a great idea if the system can predict more than one digits at the same time. The attempt for it is made for the current program but it is still in process and not working properly yet. It will be implemented for sure in the future improvement. Another idea is that another deep learning algorithms should be applied and tested for performance and accuracy purposes. Moreover, the software interface can be adaptable. Currently, the user can draw digits on the computer screen. We can go further like the user can upload handwritten images from the paper and the system detects and give the correct output. It could be challenging due to the image quality and colour etc. Additionally, other datasets like MNIST, for example, fashion MNIST dataset, can be applied to make the system better and more flexible. The same LeNet-5 CNN model can be used for this progress, but it might need to make improvements in software interface and design. The system would not the same as handwritten digit recognition, the user cannot draw the clothing images which will be complicated. Instead, the system can receive image output of any image related with clothing and it will give correct output for the specific inputs.

Overall, the developed model and system is satisfactory level, giving the accuracy of 99% which is relatively high, and it can be used in OCR applications and for organizations like banks, health care and pharmaceuticals, educational sectors and so on. By using handwritten digit recognition system, it would save time and human power and it is vital to use this to make the process faster and better in all various sectors. From my perspective, handwritten digit recognition is an excellent prototype problem for learning about neural networks and it gives a great way to develop more advanced techniques of deep learning. In the future, we are planning to develop a better real-time handwritten digit recognition system with the features mentioned above.

XIII. Appendences

Data Augmentation — Some of the images in the training set can be randomly rotated in both directions by 10 degrees, zoomed by 10 percent and shifted in both directions by 10 percent. It helps to reduce the variance of the model substantially when the model detects irregular handwritten digits.

L2 Regularization — Regularization with a hyperparameter of 0.005 (lambda) is used in some of the convolutional layers of the developed model.

Dropout Regularization — Three dropout layers with a hyperparameter of 25% are added after the pooling layers and some fully connected layers in the developed model. By randomly dropping 25% of the neurons while training, generalization of the better could be better by merely not depending a lot on any particular neurons to produce an output.

Batch Normalization — It is performed after every set of layers (ConvNet + MaxPool & Fully Connected). For this model, it is used to stabilize the network and make the learning algorithm converge faster. This is like normalizing the images from the dataset. Normalizing the activations of the various layers minimize the movement between the hidden units and lets the layers to adapt more by themselves autonomously.

Variable Learning Rate — The performance of the model is significantly increased by making learning rate variable. When the developed model finds out that its learning is stagnating, the learning rate is decreased by 0.2 factor immediately. By dynamically reducing the learning rate helps the algorithm to converge faster and reach closer to the global minima.

Adam — Adam is an adaptive learning rate method, which means, it computes individual learning rates for different parameters. Its name is derived from adaptive moment estimation, and the reason it's called that is because Adam uses estimations of first and second moments of gradient to adapt the learning rate for each weight of the neural network.

SGD — SGD stands for Stochastic Gradient Descent which is another version of Gradient Descent (GD). SGD subtracts the gradient multiplied by the learning rate from the weights. SGD works choosing one sample randomly to update the parameter. Thus, when there are a great number of samples in the dataset, the computation cost for SGD is high.

Ethical Review Form

GUIDELINES ON COMPLETING THE ETHICAL REVIEW FORM

Questions 1, 3, 4, 6 and 7 only apply if people will be involved in your project (e.g. in providing feedback on your product, in providing input on desirable aspects of your project and/or product (e.g. through questionnaires, focus groups, etc.), etc.)

Question 2 will probably only apply if you have a real client for your project.

Questions 5 and 8 should be fairly self-evident.

Before completing question 9 you need to complete a Risk Analysis & Management form, which is attached at the end of this document, to identify risks and hazards that may arise from your project, if any.

UNIVERSITY OF HUDDERSFIELD

SCHOOL OF COMPUTING AND ENGINEERING

<u>PROJECT ETHICAL REVIEW FORM</u>

Applicable for all research, masters and undergraduate projects

Project Title:	Handwritten digit recognition using Artificial Neural Networks
Student:	Htet Su San (1971904)
Course/Programme:	MSc in Artificial Intelligence
Department:	Computing and Engineering
Supervisor:	Ilias Tachmazidis
Project Start Date:	1/5/2021

ETHICAL REVIEW CHECKLIST

	Yes	**No**
1. Are there problems with any participant's right to remain anonymous?	☐	☒
2. Could a conflict of interest arise between a collaborating partner or funding source and the potential outcomes of the research, e.g. due to the need for confidentiality?	☐	☒
3. Will financial inducements be offered?	☐	☒
4. Will deception of participants be necessary during the research?	☐	☒
5. Does the research involve experimentation on any of the following?		
(i) animals?	☐	☒
(ii) animal tissues?	☐	☒
(iii) human tissues (including blood, fluid, skin, cell lines)?	☐	☒
6. Does the research involve participants who may be particularly vulnerable, e.g. children or adults with severe learning disabilities?	☐	☒
7. Could the research induce psychological stress or anxiety for the participants beyond that encountered in normal life?	☐	☒

8. Is it likely that the research will put any of the following at risk:		
(i) living creatures?	☐	☒
(ii) stakeholders (disregarding health and safety, which is covered by Q9)?	☐	☒
(iii) the environment?	☐	☒
(iv) the economy?	☐	☒
9. Having completed a health and safety risk assessment form and taken all reasonable practicable steps to minimise risk from the hazards identified, are the residual risks acceptable (Please attach a risk assessment form – available at the end of this document)	☒	☐

STATEMENT OF ETHICAL ISSUES AND ACTIONS

If the answer to any of the questions above is yes, or there are any other ethical issues that arise that are not covered by the checklist, then please give a summary of the ethical issues and the action that will be taken to address these in the box below. If you believe there to be no ethical issues, please enter "NONE".

NONE

STATEMENT BY THE STUDENT

I believe that the information I have given in this form on ethical issues is correct.

Signature: Htet Date: 1/5/2021

AFFIRMATION BY THE SUPERVISOR

I have read this Ethical Review Checklist and I can confirm that, to the best of my understanding, the information presented by the student is correct and appropriate to allow an informed judgement on whether further ethical approval is required.

Signature: Ilias Tachmazidis Date: 13/09/2021

SUPERVISOR RECOMMENDATION ON THE PROJECT'S ETHICAL STATUS

Having satisfied myself of the accuracy of the project ethical statement, I believe that the appropriate action is:

The project proceeds in its present form	
The project proposal needs further assessment by an Ethical Review Panel. The Supervisor will pass the form to the Ethical Review Panel Leader for consideration.	

RETENTION OF THIS FORM

- The Supervisor must retain a copy of this form until the project report/dissertation is produced.
- The student must include a copy of the form as an appendix in the report/dissertation.

OUTCOME OF THE ETHICAL REVIEW PANEL PROCESS, WHERE REQUIRED

Tick One

1. Approved. The ethical issues have been adequately addressed and the project may commence. ☐

2. Approved subject to minor amendments. The required amendments are stated in the box below. The project may proceed once the form has been amended in line with the requirements and signed by the Supervisor in the box imediately below to confirm this. ☐

I confirm, as Supervisor, that the amendments required have been made:

Signature: ______________________________ Date: ____________________

3. Resubmit. The areas requiring further action are stated in the box below. The project may not proceed until the form has been resubmitted and approved. ☐

4. Reject. The reasons why it will not be possible to address the ethical issues adequately are stated in the box below. ☐

For any of the outcomes 2, 3 or 4 above, please provide a statement in the box below.

AFFIRMATION BY THE REVIEW PANEL LEADER

I approve the decision reached above by the review panel members:

Signature: ______________________________ Date: ____________________

THE UNIVERSITY OF HUDDERSFIELD: RISK ANALYSIS & MANAGEMENT

ACTIVITY:			Name:	
LOCATION:			Date:	Review Date:

Hazard(s) Identified	Details of Risk(s)	People at Risk	Risk management measures	Other comments

XIV. References

Pashine, S., Dixit, R., & Kushwah, R. (2021). Handwritten Digit Recognition using Machine and Deep Learning Algorithms. arXiv preprint arXiv:2106.12614.

Deng, L. (2012). The mnist database of handwritten digit images for machine learning research [best of the web]. IEEE Signal Processing Magazine, 29(6), 141-142.

Preetha, S., Afrid, I. M., & Nishchay, S. K. (2020). Machine Learning for Handwriting Recognition. International Journal of Computer (IJC), 38(1), 93-101.

Das, S. (2017). CNN Architectures: Medium. https://medium.com/analytics-vidhya/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5

Rosyda, S. S., & Purboyo, T. W. (2018). A review of various handwriting recognition methods. International Journal of Applied Engineering Research, 13(2), 1155-1164.

Aqab, S., & Tariq, M. U. (2020). Handwriting recognition using artificial intelligence neural network and image processing. International Journal of Advanced Computer Science and Applications, 11(7), 137-146.

Matcha, A. C. N. (2021). How to easily do Handwriting Recognition using Deep Learning. Nanonets. https://nanonets.com/blog/handwritten-character-recognition/

Nkengsa, B.. (2021). Applying Machine Learning to Recognize Handwritten Characters. Medium. https://medium.com/the-andela-way/applying-machine-learning-to-recognize-handwritten-characters-babcd4b8d705

Pashine, S., Dixit, R., & Kushwah, R. (2021). Handwritten Digit Recognition using Machine and Deep Learning Algorithms. arXiv preprint arXiv:2106.12614.

John, J. (2021). Support Vector Machine for Handwritten Character Recognition. arXiv preprint arXiv:2109.03081.

Goyal, K. (2020). Building Handwritten Digits Recognizer using Support Vector Machine. Hacker Noon. https://hackernoon.com/building-handwritten-digits-recognizer-using-support-vector-machine-fos3wqs

Russell, S. J., Norvig, P., & Davis, E. (2016). Artificial intelligence: A modern approach (Third, global ed.). Pearson.

Think Automation. (n.d.). Why is handwriting recognition so difficult for AI?. Think Automation. https://www.thinkautomation.com/bots-and-ai/why-is-handwriting-recognition-so-difficult-for-ai/

Islam, K. T., Mujtaba, G., Raj, R. G., & Nweke, H. F. (2017). Handwritten digits recognition with artificial neural network. In 2017 International Conference on Engineering Technology and Technopreneurship (ICE2T) (pp. 1-4). IEEE.

Shamim, S. M., Miah, M. B. A., Angona Sarker, M. R., & Al Jobair, A. (2018). Handwritten digit recognition using machine learning algorithms. Global Journal Of Computer Science And Technology.

Dutt, A., & Dutt, A. (2017). Handwritten Digit Recognition Using Deep Learning. International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), 6(7), 990-997. http://ijarcet.org/wp-content/uploads/IJARCET-VOL-6-ISSUE-7-990-997.pdf

Chychkarova, Y., Serhiienkob, A., Syrmamiikha, I., & Karginc, A. (2021). Handwritten Digits Recognition Using SVM, KNN, RF and Deep Learning Neural Networks . CMIS-2021: The Fourth International Workshop on Computer Modelling and Intelligent Systems, , http://ceur-ws.org/Vol-2864/paper44.pdf

Siddique, F., Sakib, S., & Siddique, M. A. B. (2019). Recognition of handwritten digit using convolutional neural network in python with tensorflow and comparison of performance for various hidden layers. In 2019 5th International Conference on Advances in Electrical Engineering (ICAEE) (pp. 541-546). IEEE.

Sultana, F., Sufian, A., & Dutta, P. (2018). Advancements in image classification using convolutional neural network. In 2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN) (pp. 122-129). IEEE.

Alpaydin, E., MIT Press, IEEE Xplore (Online Service), & Books24x7, I. (2014). Introduction to machine learning (Third;3rd; ed.). MIT Press.

IBM. (n.d.). Machine Learning. IBM. https://www.ibm.com/uk-en/cloud/learn/machine-learning

Brownlee, J. (2019). What is Deep Learning? Machine Learning Mastery. https://machinelearningmastery.com/what-is-deep-learning/

Jagtap, V. N., & Mishra, S. K. (2014). Fast efficient artificial neural network for handwritten digit recognition. International Journal of Computer Science and Information Technologies, 5(2), 2302-2306.

Hossain, M. A., & Ali, M. M. (2019). Recognition of handwritten digit using convolutional neural network (CNN). Global Journal of Computer Science and Technology.

Alwzwazy, H. A., Albehadili, H. M., Alwan, Y. S., & Islam, N. E. (2016). Handwritten digit recognition using convolutional neural networks. International Journal of Innovative Research in Computer and Communication Engineering, 4(2), 1101-1106.

Kussul, E., & Baidyk, T. (2004). Improved method of handwritten digit recognition tested on MNIST database. Image and Vision Computing, 22(12), 971-981.

Wu, H. (2018). CNN-Based Recognition of Handwritten Digits in MNIST Database. Research School of Computer Science.–The Australia National University, Canberra.

Yu, N., Jiao, P., & Zheng, Y. (2015). Handwritten digits recognition base on improved LeNet5. In The 27th Chinese Control and Decision Conference (2015 CCDC) (pp. 4871-4875). IEEE.

El-Sawy, A., Hazem, E. B., & Loey, M. (2016). CNN for handwritten arabic digits recognition based on LeNet-5. In International conference on advanced intelligent systems and informatics (pp. 566-575). Springer, Cham.

Pechyonkin, M. (2018). Key Deep Learning Architectures: LeNet-5. Medium. https://medium.com/@pechyonkin/key-deep-learning-architectures-lenet-5-6fc3c59e6f4

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.

Tensor Flow. (n.d.). *What is Tensor Flow?.* Tensor Flow. https://www.tensorflow.org/

Sypder. (n.d.). *Spyder*. Spyder. https://www.spyder-ide.org/

Keras. (n.d.). *Keras*. Keras. https://keras.io/

Anaconda. (n.d.). *Anaconda Documentation*. Anaconda. https://docs.anaconda.com/anaconda/navigator/index.html

Matplotlib. (n.d.). Matplotlib. Matplotlib. https://matplotlib.org/

Numpy. (n.d.). Numpy. Numpy. https://numpy.org/

Pandas. (n.d.). Pandas. Pandas. https://pandas.pydata.org/

Scikit. (n.d.). *Scikit*. Scikit. https://scikit-learn.org/stable/

HP. (n.d.). OCR: The most important scanning feature you never knew you needed. HP. http://h71036.www7.hp.com/hho/cache/608037-0-0-39-121.html

Das, N., Mollah, A. F., Saha, S., & Haque, S. S. (2010). Handwritten arabic numeral recognition using a multi-layer perceptron. arXiv preprint arXiv:1003.1891.

Indolia, S., Goswami, A. K., Mishra, S. P., & Asopa, P. (2018). Conceptual understanding of convolutional neural network-a deep learning approach. Procedia computer science, 132, 679-688.

Saha, S. (2018). A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way. Towards Data Science. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Yamashita, R., Nishio, M., Do, R. K. G., & Togashi, K. (2018). Convolutional neural networks: an overview and application in radiology. Insights into imaging, 9(4), 611-629.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 25, 1097-1105.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

Kumar, M., Jindal, M. K., & Sharma, R. K. (2011). k-nearest neighbor based offline handwritten Gurmukhi character recognition. In 2011 International Conference on Image Information Processing (pp. 1-4). IEEE.

Lee, Y. (1991). Handwritten digit recognition using k nearest-neighbor, radial-basis function, and backpropagation neural networks. Neural computation, 3(3), 440-449.