Application of Neural Network in Image

Abhinay Kanaparthi
School of Computing and Mathematics
University of South Wales

A submission presented in
partial fulfillment of the requirements
of the University of South Wales/Prifysgol De Cymru
for the degree of MSc Data Science

September 2022

UNIVERSITY OF SOUTH WALES

PRIFYSGOL DE CYMRU

FACULTY OF COMPUTING, ENGINEERING & SCIENCE
SCHOOL OF COMPUTING & MATHEMATICS

STATEMENT OF ORIGINALITY

This is to certify that, except where specific reference is made, the work described in this project is the result of the investigation carried out by the student, and that neither this project nor any part of it has been presented or is currently being submitted in candidature for any award other than in part for the MSc Data Science degree of the University of South Wales.

Signed .... (student)

Date ....

(This statement must be bound into each copy of your Project Report)

Acknowledgments

I would like to thank everyone at the University for their support in their project, especially Ieuan Griffiths, Angelica Pachon, and Moizzah for their guidance throughout this course work. I would like to express gratitude to Angelica for allowing me to embark on this project.

Abstract

This project is based on image classification on a neural network. The MNIST dataset was used for research. The dataset gets from Kaggle which is easily to access and all the projects are based on the dataset. The dataset contains 60000 test images.in the previous article, many researchers use python and get 99.57 percent accuracy. The current aim of the project is to increase the accuracy of the project and get a good result with the same data set. The Jupiter notebook uses to achieve the aim of the project. In Jupiter notebook, the machine learning module uses to achieve the aim. In the discussion obtain the result and compare it with the previous result. so, in conclusion, to find which method is working in the project which the previous author does not use in the research.

Contents

Chapter 1 Introduction

In the current world with the growing technology, dependency on machines over human life is increasing for completing their day-to-day work which includes simple algorithms for news reading to complicated algorithms for robot building, making machines learn from it previous data or mistakes is what machine learning is about and in the project we will be exploring image classification algorithm where we try to build an image classification model which can learn from it training data and try to classify a different image in a real-world scenario. Each image can be considered as a combination of multiple pixels, a single pixel has a color coding number between 0 - 256 where 0 represent black and 256 represent white and different color can be obtained at other color coding and that is how any machine read an image and try to process it accurately.

The dataset chosen for this project is the *MNIST[1]* Digit dataset which contains over 60,000 training images and 10,000 testing images where each image is a 28 x 28 grayscale image and is associated with a label of 10 classes where each class is a number between 0 -9 which are single digit number and any number can be created using the above number, along with which MNIST has a second set of the dataset which is related to Fashion and has the same image count and size, but the label here is a bit different which include 'Trouser', 'Dress' etc. and total 10 different labels are present, and both datasets are present in open source and can be used for learning and education purpose. Since both datasets contain images of different objects and the main objective of our model is to classify images and since we have two different models we will be training two different models, but the architecture of both models will be kept the same as they both accept input image of 28 x28 images and had total 10 different output layers and this will be done to check how weights are changed when the same model is trained with different dataset and then compare the result which will be achieved by the end of the analysis.

Along with the building of the different models, we would be evaluating both the model based on some standard metrics and common optimization methods for reducing loss function which includes plotting of loss value and accuracy value at different iterations which will give us useful information regarding how our model is learning better moving in the forward direction and in case the change in accuracy value between the current iteration and the previous iteration is less than the given threshold value which can be 0.01 then it will automatically stop the iteration as moving forward will be of no use because the model has already learned whatever it supposes to and the point where it has achieved maximum accuracy at that iteration, the model will automatically be saved and will be used for future prediction.

The model is the project is based on a Convolutional Neural Network which works by multiplying an image matrix with the given matrix called kernel of the defined size called kernel size and hence there can be a reduction of image pixel if kernel size is small and along with these, there is the addition of different pre-processing layers of the image which include removing unnecessary noise from the image by normalizing the image and addition of auto encoder layer which will automatically compress and de compress the image to remove noise from image all image processing is done with help of OpenCV as all the above processing will be helping in better understanding of image and finally before passing each image to model each image has to go through a process of *data augmentation[3]* which will drastically increase our training image and will provide real world image to the model which include image flipped in different direction or image rotation in different direction which will make our model robust to real world example as an image can be in different orientation when clicked and passed to model and that processing will be providing much better accuracy with simpler model as now we have a big data because each image will be repeated minimum 5 times in different orientation and that mean total 3,00,000 training images the process of data augmentation will not be applicable to testing data but other processing such as normalization and sharpening of image will be applicable to model, and finally we will be using plotting confusion matrix which will be a 10x10 matrix for our training dataset to check where our prediction is going wrong and then try to make that class more accurate by increase the count image of given image or finding what is the reason that model is predicting in wrong manner for given class. All the above pre-processing is missing from major paper and that is why complex algorithms must be used to achieve good accuracy but in our case, a simple algorithm will do the job as the image is already converted from raw format to processed format and is in the best shape to be passed from any given model. Each section discussed in the introduction will be briefly discussed in the upcoming section.

1.1 Machine Learning

Observation or data, such as examples, direct experience, or instruction, are the starting point for machine learning. It analyses data for regularities so that it may conclude from the examples given. The basic goal of ML is to let computers learn and adapt on their own, without any help from humans. The idea of "machine learning" has been around for a while. Machine learning was first defined by IBM researcher and early proponent of artificial intelligence (AI) and computer games inventor Arthur Samuel. Samuel created a checkers-playing application for the computer. The more it performed, the more it learned from its mistakes and improved its performance via the use of algorithms, different algorithm which can also be used with the current dataset are decision tree, support vector, KNN, and Naive Bayes a small introduction of each model is added in the below section and why these models cannot overcome neural network is also explained. This decision tree classifier is an example of supervised learning in action. Alternative supervised learning algorithms, outside of machine learning. The standard method for dealing with regression and classification problems. The major goal is to build a training model that uses historical and accumulated data to make predictions about the class or desired outcomes of the rules for understanding judgments. The primary goal of the model is to generate a compact tree capable of categorizing the unidentified class or example by identifying the set of rules of the handwritten digit dataset. Data retrieval, visual recognition, autocorrect, and machine learning are just a few of the many techniques that go into making up the DT classifier. With the current DT's C4.5 simulation method as a starting point, the study zeroes down on the hunt algorithm to generate many DTS. To implement the C4.5 method, the Classification DM platform (also known as J48) provides an open-source alternative. Secondly, the algorithm's protocol selects the root node's parameters and constructs a branch for quality is good. To optimize the separation between classes while separating pieces of data in a high-dimensional space, a supervised learning system called a Support Vector Machine (SVM) is used. SVM is a way of representing instances as points in space, where each class's examples are mapped into a fair gap that is all-encompassing and feasible. After then, if a new example is introduced, it is placed in the same area and is expected to stay in the same partition. Training data is used to develop an algorithm that can discriminate between classes previously specified by the operator (for example, patients and control), and testing data is used to randomly decide the group to which a new event belongs, leading to an ideal algorithm. Along with providing a trustworthy representation of classification across the training records, it also creates a large search space in which all conceivable data parameters may be appropriately classified. As a result, it guarantees at least some reasonable data subsets for a wide range of input parameters. As a result of the dramatic speed boost that would result from scaling the data, SVM is frequently the preferred method. Keeping this in mind, a large dataset may cause an increase in the training phase, therefore caution is advised. A naive Bayes classifier represents probabilistic information with clear semantics. Since it relies on two basic assumptions that predictive qualities are conditionally self-reliant assuming the class, it's nave and claims no hidden attributes harm the prediction system. It's a probabilistic classifier based on Bayles's theorem independence assumptions. It's used for personal email sorting, spam identification, objectionable content identity, document categorization, emotion recognition, and language detection. In many complicated real-world issues, Basic Bayes works effectively despite its naive design and simplistic assumptions. Although various algorithms are used, such as enhanced trees, Max Entropy, Supports Vector Machine (SVM), random forests, etc., the Naive Bayes classifier is useful since it uses less memory and CPU and requires less data for training. Naive Bayes training is shorter than other techniques. The KNN (K-Nearest Neighbour) method is a more straightforward approach that may be used for educational purposes. The non-parametric method is most often used for statistical tasks involving regression and classification. A technique for the unmotivated or slow learner is somewhat implied by this approach. In contrast, the functionality of KNN is cantered on an algorithm that compares feature similarities. Once it has been completely built, this model will be taught on training samples, and after that, it will identify comparable samples in test data. The K points in the handwriting digit recognition dataset are used to determine how the KNN working function should be configured. As a result, the algorithm calculates the numbers that are closest to the value K, and it also predicts the plurality votes for the points that are closest. The fundamental idea behind the KNN is simply represented by an application of a certain class. In addition to this, even the closest neighbor is detailed in more detail by each particular class.

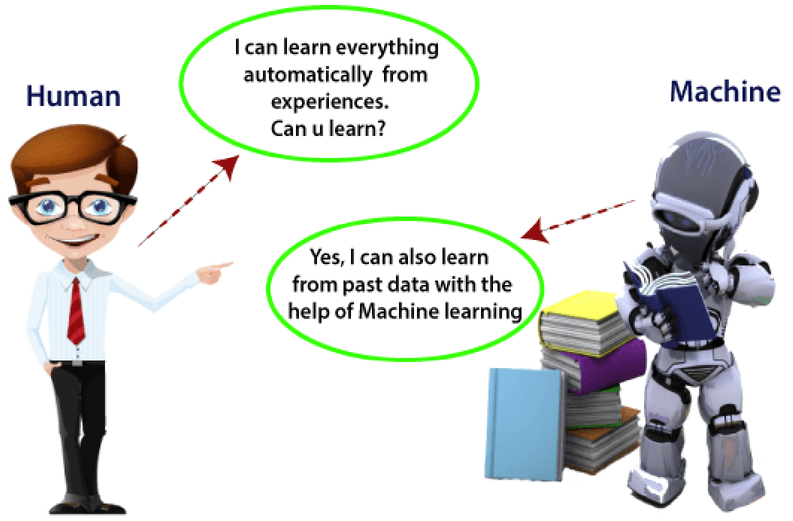

Human
I can learn everything automatically from experiences. Can u learn?
Machine
Yes, I can also learn from past data with the help of Machine learning

Figure - I Machine Learning

1.2 MNIST Dataset

The size of the MNIST picture collections was reduced from 128 by 128 pixels to just 28 by 28 pixels throughout the measurement. The MNIST dataset is represented as an 8-bit picture with a resolution of 282 pixels. To begin, the pre-processed grayscale image is determined by the measurement of the center mass pixel. It is positioned in the center of the photos with a resolution of 282 pixels so that the suitable formats of the MNIST dataset may be established. The data collection is accessible for further investigation and testing, as well as for further preliminary processing. Although the NIST dataset has a larger collection of 814,255 pictures, MNIST only takes part in a small portion of the research since the database only needs a ten-digit range, between zero and nine. The processing of MNIST data has made them quite widespread. The competence of the various models employed for categorization serves as a benchmark. The dataset that demonstrates that newly constructed models can be verified reliably and appropriately has been utilized, changed, and inspected by thousands of researchers. The researchers can more easily analyze and disseminate their findings because of the complete usage and quick access to the data. It discusses several current studies that have been conducted on the use of machine learning in MNIST data sets.

The MNIST collection of handwritten digits, which can be accessed via Kaggle, has a training set of 60,000 samples, as well as a test set of 10,000 cases. It is a portion taken from a more extensive collection that may be obtained from the NIST. The digits have had their sizes adjusted and have been cantered inside an image that has remained the same size. A low-complexity data collected from handwritten characters, the MNIST dataset is an expansion of the NIST database that is used to train and test a variety of supervised machine-learning algorithms. The database includes seventy thousand black and white pictures measuring 28 by 28 pixels, each of which represents the numerals zero through nine. The data are separated into two distinct subsets: the training set has 60,000 photos, while the testing set contains 10,000 photos. The separation of pictures guarantees that, provided what a model that has been appropriately trained has learned in the past, it is capable of correctly classifying relevant photos that it has not focused on fact.

1.3 Image Classification

The process of obtaining various data classes from a raster picture using several bands is referred to as image classification. The raster that is produced as a consequence of picture categorization may be used in the process of producing themed maps. There are two different forms of classification, supervised and unsupervised, and both are determined by the interaction that occurs between the analyst and the computer throughout the classification process. There is a whole range of tools in the Multivariate tool set that can do supervised and unsupervised classification when you have the Arc GIS Spatial Analyst extension installed on the computer. Because the classification procedure involves a multistep workflow, the Image Classification toolbar has been built to offer an integrated environment in which to carry out classifications using the various tools. The toolbar not only assists with the work process for conducting unsupervised and supervised classification, but it also includes extra capabilities for analyzing input data, creating training data and signature files, and deciding the quality of the training images and signature files. This functionality is in addition to the fact that the toolbar helps with the work process for performing supervised and unsupervised classification. The Image Classification toolbox is the method of choice for doing classifications and multivariate analysis since it is the preferred method.

Image classification is a supervised learning problem, which means that first, you need to specify a collection of target classes (things that need to be identified in pictures), and then you need to train a model to recognize them using labeled sample photographs. Image classification is a problem that can be solved by supervised learning. Image classification is a problem that can be solved by supervised learning. The early computer recognition models that were created relied heavily on unprocessed pixel data as their major source of input. However, raw pixel data by itself does not provide a representation that is stable enough to encompass the many subtleties that may be present in an object as it is shown in a photograph. This is because raw pixel data is composed of individual pixels rather than groups of pixels. Raw pixel data may change depending on a variety of parameters, such as the location of the item being photographed, the attribute being photographed, the backdrop behind the attribute, the ambient illumination, the camera angle, and the camera focus. These variances are so severe that trying to correct them by calculating a weighted average of the RGB values for individual pixels is fruitless.

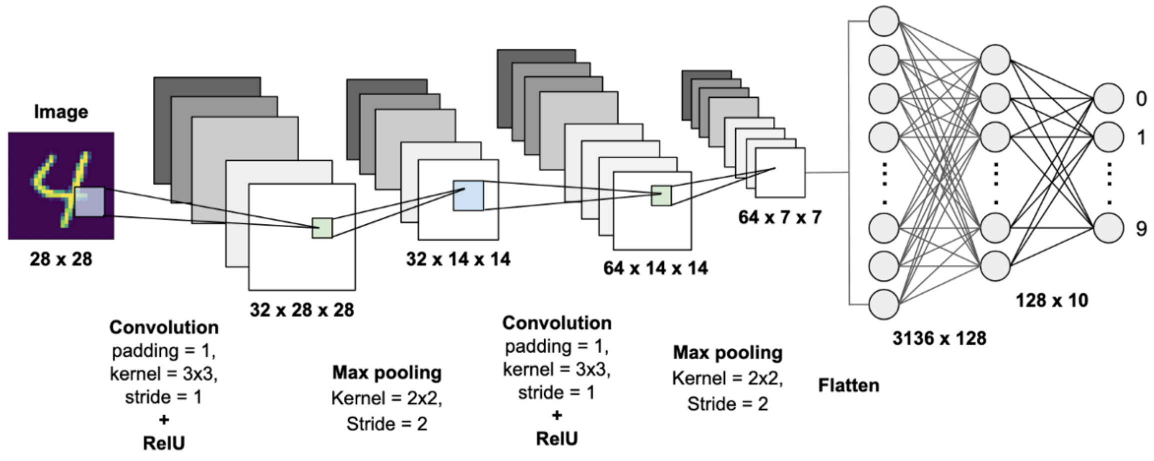


Figure - II Image Classification

Chapter 2 Literature Review

As per Chen (2018), Using CNN, the oldest date that can be calculated using BP may be dated back to the year 1986. After a year, in 1989, LeCun implemented it in multi-layer neural networks. LeCun did not propose the LeNet-5 model until 1998, at which point the artificial neural prototype was already finished. As per Shahid*et al.* (2019), the most important factor behind the growth of the multi-layered neural system is the function of the system which makes things complete for creating the overall growth of the components of the images which are captured in high resolution. According to Azarmdel*et al*. (2020), the most specific feature of a multilayer neural system is creating the image packages for sending all the data available on the image surface. According to Chicco (2021), the AI systems with machine learning applications need to understand the overall uses of picture recognition, for creating a good impact on the overall parameter of image recognition. The most important thing here is that the neural layers can identify all the features within an image deeply, and categorize a single image into different categories for identifying the images properly. As per Chaganti*et al*. (2020), Machine Learning is used for knowing the approaches to understanding the deep information in the images.

Computers known as artificial neural networks mimic how real brains function by doing their tasks in a manner that is similar in some ways to the way that humans do. Artificial neural networks, often known as ANNs, are described as "basically frameworks that are made up of a network of computer nodes," according to O'Shea and Nash (2015). These nodes collaborate in a distributed manner to do complex data analysis to increase the quality of the output they provide. Since CNN's are built up of self-optimizing neurons, their construction is fairly comparable to that of ordinary ANNs in terms of the components that make them up. The most important thing here is the application of the neural system is creating the process which provides the opportunity for making the identification of the pixels of any image. As per Annarumma*et al*. (2019), the process of CNN opens wide scopes for completing multiple duties for making processing all the images.

Image processing is important for understanding the overall maturity of all the images and understanding the overall impacts of all the images in any layered neural. According to Maksimenko*et al*. (2018), the most important part of CNN is the Neurons, which include the ANNs, and provides the opportunity to make a huge change in the overall parameters of the system of image recognition. In numerous cases, it has been found that the application of CNN is standing on the framework of the ANN, which is making the process of decoding multi-layered data easier for researchers. Despite these distinctions, the whole system will only ever represent a single attentive scoring function (the weight).

As per O'Shea and Nash (2015), CNN has inherited all of the standard techniques and tactics that may be utilized for traditional ANNs. These strategies and tactics can be found in O'Shea and Nash (2015). The application of these methodologies and methods may result in improvements to conventional ANNs. As per Fourcade and Khonsari, (2019), improvement in the system of ANN classifies the presence of all things very carefully, where the most important thing is understanding the depth of the things that are available for any particular human being. The most important thing for the neural system is finding the overall data from any identifying objects. Like understanding the gender of a human being and understanding the overall age of a human being becomes possible with the application of the neural system with AI and machine learning.

According to a different theory proposed by Kimura *et al*. (2019), SVM (Support Vector Machine) is the concept thing that was developed for doing such approaches and making things much simpler to identify all the parameters within an image.

On the other hand, according to Park and Park, (2018), the most important thing is creating image filtering for decoding all the parameters within an image. Any image profile contains multiple things which are essential for creating the overall image profile. Another impact of image detection is huge on the overall growth of the technology of image prediction and identifying objects from very far distances.

The process of assigning a label to an input picture based on a set of categories that have been selected in advance is referred to as image classification. The production of robots, the recognition of objects, self-driving cars, and the analysis of traffic signals are just some of the many applications that may make use of image categorization. When it relates to the difficulty of categorization, the task of extracting the features is more important than the visual portrayal. CNN was used on a large-scale dataset to learn the representation of pictures and then reuse it for categorization. When contrasted to other neural networks, it provides superior accuracy and performance in the picture classification issue. This was shown by its success in categorizing the CIFAR-10 data set.

CNN designed a new layout to address the overfitting problem brought on by a reduction in the number of images. The findings indicate that CNN gives improved gender and age classification, and this holds even when using contemporary unconstrained photo sets with smaller sizes. An experiment was carried out by Karpathy et al. (2014) utilizing CNN to identify large-scale pictures. As a dataset, they used one million films from YouTube that had been separated into 487 distinct categories.

As per Ahlawat (2020), 1995 marks the first year that SVM was used for the task of handwritten digit optical character recognition. Later on, Support Vector Machine (SVM) classifiers became the go-to option for a wide variety of supervised classification applications, including character recognition, face identification, and object recognition. Boukharouba and Bennie created a Support Vector Machine-based handwritten digit identification system and offered a technique based on the Freeman Chain Code for extracting features of handwritten digits. In recent years, convolutional neural networks have shown to be useful tools for handwritten digit recognition, particularly when applied to the standard MNIST handwritten digit dataset. According to Kickingeredere*t al*. (2019), the most important thing in the scenario of solving the issues of image transmission is creating the overall image fitting. The overall image fitting parameters are an important part of the overall understanding of an image. As per De Araujo Faria*et al*. (2021), the application of the neural system has been implemented for making an overall change in the parameter of algorithm analysis. The most important thing here is that the algorithm analysis is a factor that can change the overall scenario of the problems for making development in the field of image detection.

After this, the effort was expanded even further into a 35-net committee, up from an earlier 7-net committee, and it was found to have an extremely high accuracy of 99.77%. For the MNIST dataset, the Deep Belief Networks (DBN) with three layers and a greedy algorithm were explored, and they reported efficiency of 98.75%. CNN was used to do in-air handwritten Chinese character identification, and this allowed for the investigation of bend directional feature maps. The researchers Lauer et al. are working on a feature representation for the MNIST database that is based on the LeNet5 convolutional neural network architecture. According to the study, the recognition accuracy was quite high. According to Wang *et al*. (2019), Deep Belief Networks (DBN) have been applied to understand the efficiency of the applications. Along with that, the most important thing is that identification of people's faces from different races was completed for generating the overall data about the efficiency of the applications.

As per Ahlawat (2020), the SoftMax layer of CNN is swapped out in favor of a hybrid model that combines CNN with support vector machines (SVM). This model is referred to as the CNN-SVM hybrid approach. CNN is used to extract features, whereas SVM is used to classify data into one of two categories. A straightforward CNN architecture plus a support vector machine classifier make up this model. The CNN input layer requires 28 × 28 matrices of handwritten digits that have been normalized, cantered, and obtained from the MNIST dataset. At the convolutional layers, we employ convolutional filtering with a size of 5x5 and a stride size of 2. The N1 and N2 feature map layers each extract values from the input picture, and these values are later regarded to be differentiating characteristics of the image. The CNN is trained after multiple epochs have been run, and the training process is continued until it conforms.

As per Keysers (2007), in many different types of image recognition jobs, the process of picture matching is an essential step. One of the more promising techniques for achieving low error rates is the classification that is based on the flexible match of a picture to specified references. This is especially true in situations when there is a high level of image variability. In this research, we examine and evaluate multiple models for picture matching in the context of recognition. We demonstrate that these models lead to great outcomes across a variety of tasks by analyzing the data they provide. In the scientific literature, several deformation models of varying degrees of complexity have been addressed; nevertheless, these models are often not directly compared to one another. According to Yang and Yang, (2020), the application and combination of the multiple technological applications including the SoftMax layer with CNN, which supports SVM create a better opportunity for extracting the data from the images. An important fact is here that the most important thing here is that application of all the technologies makes it easier to decode the image and classify the image into multiple parameters. In the combined applications all the steps are important for making the overall analysis of all the details within a particular image. On the other hand, according to Abiodun*et al*. (2018), it is an essential task for the system to detect all the parameters which can be done with the most basic part of CNN is the ANN neurons are working properly and maintained with the best kind of efficiency. Asserting this statement, the researchers of Barrett *et al*. (2019), it is indeed essential to calculate the efficiency of the ANN neurons for making an overall change in the mapping of the images.

Chen (2018), evaluated the performance of four popular image recognition methods on the MNIST dataset using four distinct divisions. CNN, ResNet, and Dense Net are three of them, and they are listed in that order. It has previously been shown that these three models have high performance in picture recognition, they review the properties of these models. In addition, they have found that the conventional CNN model has several flaws that need to be addressed. In light of this, we start with CNN as our baseline model, and then we utilize Caps Net to build upon it and optimize based on this foundation. It is the fourth model, and Section 3 provides a comprehensive description of it.

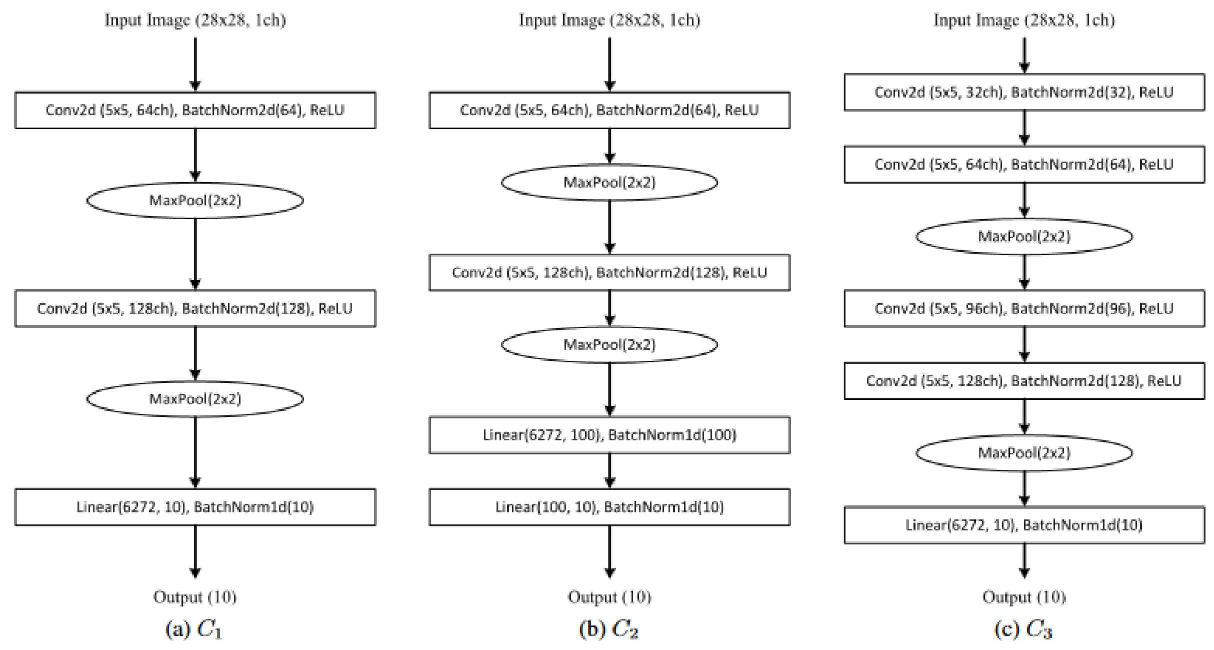
To begin, there are common benchmark datasets available, like MNIST, which make it simple for us to acquire results. Second, there are a lot of articles and methods that are already out there that can be respectively mentioned and expanded upon. We randomly split the MNIST datasets into 25%, 50%, 75%, and 100% to assess the generalizability of the model in the context of picture identification. Our goal was to make the model as generalizable as possible. As per Chen (2018), the dataset known as MNIST was developed by the National Institute of Standards and Technology (NIST). The training set comprises handwritten integers from 250 different persons, with high school seniors accounting for half of those numbers and the Census Bureau accounting for the other half. The percentage of handwritten digital data that is included in the test set is also the same.

According to & Lee*, et al*, (2020), One of the most fundamental data sets that are used to assess the effectiveness of neural network models and learning approaches is called the MNIST handwritten digit identification data set. Learning approaches such as k-nearest neighbors (KNN), random forests, support vector machines (SVM), and basic neural network models may easily reach an accuracy of 97%-98% when applied to a test set of 10,000 photos when using 60,000 images as the training set. This accuracy is improved to over 99% with the use of convolutional neural networks (CNN), with less than 100 photos in the test set being incorrectly labeled. The last one hundred pictures are much more challenging to appropriately categorize. We need increasingly complicated models, careful tuning of hyper parameters such as train rate and weight of the product, regularization methods such as batch normalization and dropout, and supplementation of training data to enhance accuracy beyond 99%. On the MNIST test set, the greatest accuracy that can be achieved ranges between around 99.7% and 99.84%.

In the network architecture when developing a CNN, one of the most popular practices is to make use of pooling, which may take the form of either maximum pooling or averaging pooling. The width of the feature maps may be reduced by the use of pooling, which also helps provide translation invariance. A typical CNN model is made up of a series of convolution layers, each of which is preceded by a pooling layer. At the very end of the model are one or more completely connected layers, as well as one or more fully connected layers. Some networks begin with two levels of convolution before moving on to the pooling layer. Figure 8 illustrates some of the most often used CNN topologies, and the names C1, C2, and C3 are provided for the three networks.

Figure: Commonly used CNN structures with max pooling

At the beginning of the training process, it is possible to see that networks using max pooling experience oscillations in their test accuracy. This may be noticed. On the other hand, the accuracy score of M5 improves in a way that is more consistent over time. the test accuracy of thirty different networks that had between fifty and one hundred fifty epochs of training. Since the average exactness of C3 and M5 is higher than that of C1 and C2, this suggests that increasing the number of convolution layers might result in improved feature learning. As seen by the precision of C1 and C2, the presence of additional layers at the end that was entirely linked did not seem to be beneficial. When contrasted to C3, the M5 has greater overall accuracy, and it also has the potential to attain an even high precision in the best-case scenario.



Input Image (28x28, 1ch)
Conv2d (5x5, 64ch), BatchNorm2d(64), ReLU
MaxPool(2x2)
Conv2d (5x5, 128ch), BatchNorm2d(128), ReLU
MaxPool(2x2)
Linear(6272, 10), BatchNorm1d(10)
Output (10)
(a) C_1
Input Image (28x28, 1ch)
Conv2d (5x5, 64ch), BatchNorm2d(64), ReLU
MaxPool(2x2)
Conv2d (5x5, 128ch), BatchNorm2d(128), ReLU
MaxPool(2x2)
Linear(6272, 100), BatchNorm1d(100)
Linear(100, 10), BatchNorm1d(10)
Output (10)
(b) C_2
Input Image (28x28, 1ch)
Conv2d (5x5, 32ch), BatchNorm2d(32), ReLU
Conv2d (5x5, 64ch), BatchNorm2d(64), ReLU
MaxPool(2x2)
Conv2d (5x5, 96ch), BatchNorm2d(96), ReLU
Conv2d (5x5, 128ch), BatchNorm2d(128), ReLU
MaxPool(2x2)
Linear(6272, 10), BatchNorm1d(10)
Output (10)
(c) C_3

Figure - XXV Commonly used CNN strictures with max pooling

As per An & Lee*, et al*, (2020), A method known as "data augmentation" is a way to broaden the scope of training data without resorting to the time-consuming process of physically gathering and labeling more examples. It is a fundamental method for supervised learning, and it requires a large data collection for the network model to function well. During the training of the proposed network, we made use of two different methods for the production of data: randomized rotation and random translations. There are a great number of additional approaches, such as cropping, rotating, and resizing, and the most effective methods of enhancement are dependent on the data. In this part of the article, we investigate whether data augmentation truly helps improve the performance of the network. The effect of four M5 networks using a variety of augment scheme combinations and comparing them to one another. The prevalence plot of all 30 networks for each of the four distinct augmentation procedures is shown in Figure 11.

Batch normalization is a well-established method that may boost not only the throughput of the system but also its consistency and the rate at which it is trained. Batch normalization is beneficial to the majority of neural network models, according to certain reports. In this part, we investigate the effect that batch normalization has on the operational capabilities of the M5 network model. It should come as no surprise that applying batch normalization to neural network models helps enhance their overall performance. When applied to each convoluted and fully linked layer, batch normalization yields the highest performance results. According to Wang *et al*. (2019), the most effective application of batch normalization can be interpreted with the application of the factor that is creating an analysis of all the parameters after normalizing all the data available thereby. The most important thing here is that batch normalization as all the accuracy is estimated on a particular basis, as well as on the overall structure of the systems. The data set which was presented for the experiment was not containing sufficient information about the images, and the test was firing information about the images with the application of the integrated system and focused on all the data for evaluating the overall cases for knowing the overall data ranges.

The MNIST handwritten digit data set is often used as a starting point for training and testing neural networks. It is quite simple to get an accuracy of 99% on the test set; nevertheless, it is difficult to accurately categorize the remaining 1% of the photos. People have experimented with a wide variety of network models and methods to improve test accuracy; the best accuracy that has been recorded is roughly 99.8 percent. In this article, we demonstrated that the greatest accuracy could be achieved by using a simple CNN model that included batch normalization and data augmentation. A performance improvement of up to 99.91% test accuracy is possible when using an assembly of both homogeneous and heterogeneous network models. This level of performance is considered to be among the best currently available. The high performance was not obtained by a single approach or model architecture, as shown by studies using a variety of different configurations; rather, it was provided by numerous methods, such as batch normalization, pooling layers, and density estimation.

Literature gap

When trained and assessed on the source domain, a conventional neural network, also known as a CNN, is capable of attaining a satisfactory level of accuracy (98%) (SVHN). However, the same Classification algorithm may have poor performance (efficiency of 67.1%) when assessed on the particular domain (MNIST), despite the common belief that the target set is an easier job to complete. In most cases, the disparity in performance can be traced back to the different distributions between the two domains. The pictures that were taken from the SVHN database include a variety of computer typefaces, a messy backdrop made up of streets, and numbers that have been clipped around the image borders. The literature is containing a huge gap in the data sets which is creating a huge problem for the researchers, most of the data have no chance of revaluating based on the actual data. The chances of crosschecking are very little. In the past, huge development over such projects has not been conducted which asserts the overall success of the research work is widely dependent on the practical notes in that particular research work.

Chapter 3 Research Methodology

Introduction

Images classification and justification process has been recognized based on using the MNIST framework in the analysis process. Multiplication analysis of the image matrix with a useful analytical model has emerged as a better structure in the supportive analytical process. Therefore, image analysis over the MNIST data set framework and CNN model over different layers of the image sensing structure has improved better analytical approaches.

MNIST dataset framework

MNIST has provided a popular dataset that has instigated the process of image classification and bench marketing analysis structure. Matplotlib is a process that has instigated the structure of plotting image sensing codes in the analysis process (digital ocean, 2022). It has provided the idea of data dealing and application procedures that have emerged as a cognitive managerial process in analytical aspects. In Fashion using the MNIST data set for analysis of the image interpretation process has regulated a challenging replacement situation in the analysis of importing and plotting aspects.

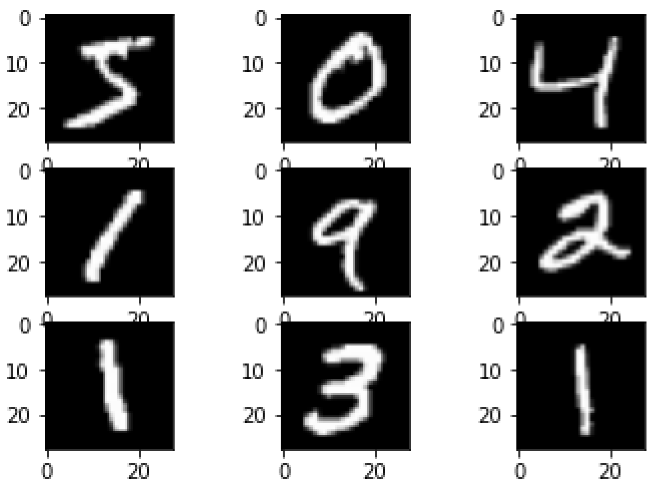


Figure 1: MNIST framework analysis for image sensing analytical approaches

(Source: Greeshma and Sreekumar, 2019)

Therefore, using of "Hyperparameter Optimization and regulation method" using in the image sensing and plotting analysis process has imported a constructive proposition in the image analysis process. The "Hyperparameter Optimization and regulation method" method has been utilized based on CNN operational process by collaboration with three regulatory phases in the service allocation structure (Greeshma and Sreekumar, 2019). The first phase of a consultation operation process related to the kernel's images in small size of groups. Architectural analysis of the neural networking process has reflected a constructive growth in the regularization of image sensing and management techniques.

Three layers are discussed below

a. Image Processing

The F-MNIST database system contained 28x28 dimensions of 70000 images contained in analysis sectors. Training data and database analysis based on testing and training management of database systems have imparted supportive image sensing and analysis practices. Using the CNN format has analyzed 28x28 dimensions and provided specific aspects of service developmental structure (Zhao *et al.* 2020). Rephrasing the specific format utilized based on the grayscale analysis of the database structure. One hot encoding technique has been used to analyze image sensing and analytical structure in service analysis and developmental processes. Analysis of the handwritten natural images using MNIST database software has provided a constructive approach in service-generating sectors. Multiple column analyses using 0 or 1 independent variable in the analysis process have depicted a constructive growth in the target variables analysis process.

Steps that are involved in the high level of image classification using of CNN framework system are

- Using of Kernel and Features application system for creating conventional layers
- Fattening the input structure of images
- Connected neural network system creation
- Training analysis of the model system
- Output

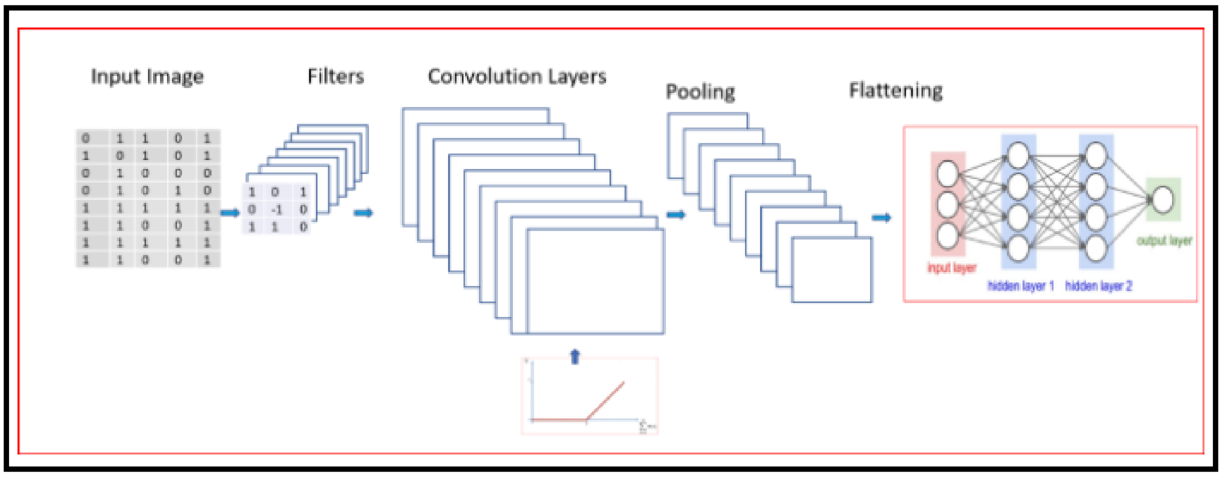


Figure 1: CNN framework in the analysis of the high level of image sensing

(Source: data-driven investor, 2022)

b. Convolution Neural Network (CNN or ConvNets)

A conventional Neural Networking system has been used to understand the artificial neural networking procedures in service developmental structure. Specific analysis of the image recognition process has emerged as an operation of neurons of the human brain. Multi-layer perception analysis of CNN networking system processed with regulation structure in constructive management in image sensing and constructive analytical aspects (Prajapati *et al.* 2020). Effective image processing and neural language processing aspects have improved cognitive steps in supportive analytical structure. The computer version of CNN has focused on image classification and face recognition processes in market analytical aspects. Dataset analysis of the fashion services and constructive business managerial structure has emerged based on the application of CNN database structure at the service level.

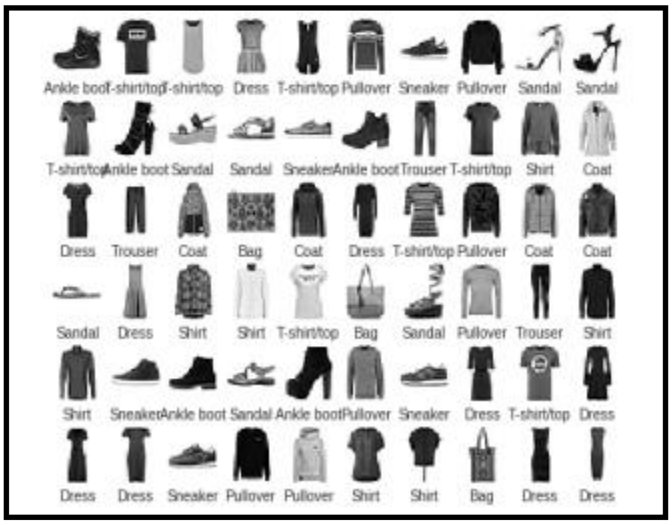


Figure 3: MNIST framework analysis ion fashion industry structure

(Source: Greeshma and Sreekumar, 2019)

32 layers of filters have been present over the CNN analytical approaches to identify the activation process. Max pooling, dropout, and batch normalization processes have instigated a convenient developmental plan in a supportive service developmental structure.

c. Regularization Techniques

Regularization techniques of the image sensing process have been instigated over two types of methods. Those are dropout and data augmentation procedures. In the dropout section, neural network organization structure with co-adaption training process has emerged a constructive growth in regularization techniques (Henzinger *et al.* 2019). Analysis of the converted fully layer of neurons framework randomly selected contribution process contributed a better growth in service developmental structure.

Data augmentation techniques of the neural networking process have regulated a constructive structure of image analysis and sensing process in a competitive market structure. Application of the artificial training process in a supportive competitive market structure has developed constructive growth in the business processes. Artificial creation and transformation processes in data augmentation and managerial structure have emerged a constructive growth in the service development structure. Random rotation and data augmentation process in the analysis structure has emerged a c obstructive growth in image analysis and sensing techniques help in the analysis of the level of the image global fashion managerial structure.

d. Hyperparameter Tuning

Hyperparameter tuning is a process that maximizes the model performance analysis structure. The optimization process of data tuning and managerial aspects have emerged a constructive growth in image analysis and sensing process. The neural networking model has emerged a constructive growth in supportive market development and managerial structure (Wang *et al.* 2019). A wide range of optical analyses of the architectural process has a constructive structure in the image analysis process in market practical analytical structure.

Mini-batch analysis of the constructive analysis process of image sensing attributes has emerged as a constructive structure in the supportive learning analysis process. Hyperparametric analysis has provided convenient analytical approaches to image sensing and the analytical attributes' analysis processes. Dataset analysis of the image analytical structure improved constructive planning in a supportive analytical structure.

Summary

Image sensing analysis using of MNIST framework has provided a multilayer approach to understanding several attributes. Therefore, constructive planning in image sensing and analytical approaches has imported constructive growth in data analytical approaches. Progressive planning and image analysis analyzed different layers of different pixels in pictures. Analytical approaches have performance constructive structure in image sensing and sensory analytical aspects over the constructive developmental planning process.

3.1 Python Libraries

1. Pandas: For applying different data operations like group by, and data splitting we choose pandas data frame which is created by reading any CSV or JSON file into python, and all data manipulation methods are already written in pandas and can be called in a single line, and

that where pandas come in handy.

2. NumPy: - NumPy is used for applying the basic mathematical operations to the data, and the panda's data frame is always converted to NumPy for faster processing and applying the complex algorithm.

3. Matplotlib: - To build different plots and figure Matplotlib is used and since we have to plot a simple graph and not much complicated such as a geographical plot, we would go with the most basic python library for plotting and all plot in Matplotlib can be displayed with a single line.

4. Seaborn: - Seaborn is another visualization library that is written upon Matplotlib and provides a much more complex and more realistic graph as compared to Matplotlib, and you can switch between both of them as they are interconnected.

5. Sklearn:- For building different machine learning models and all data pre-processing is done with help of Sklearn as it provides complex machine learning models already written and also provides different metrics which are also helpful when evaluating the different models.

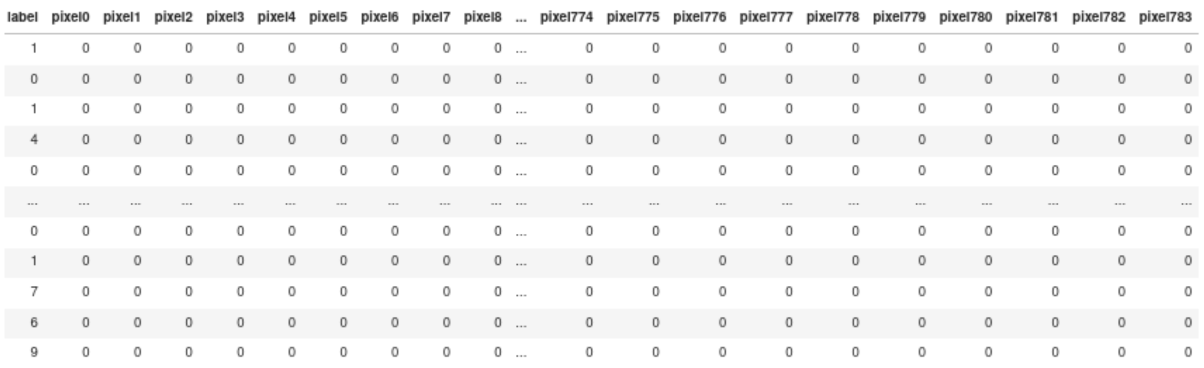
6. TensorFlow: - TensorFlow is an open-source library written in python and is compiled by google which is the owner of the library and used for the creation of complex deep learning models and analyzing them with help of tenor board which is an additional utility provided by TensorFlow itself. Along with the creation of a complex algorithm, there is a pre-built model present in the library which can be used with different datasets and big data can be processed in tensor flow with help of a tensor which is a 3-dimension data structure and can perform all mathematical tasks in the shortest time.

7. Keras: - Keras is the building block of TensorFlow and all the basic models and layers are firstly written in Keras and are upgraded into the next version with help of tensor flow, the use of the Keras library can be done independently or along with the tensor flow as there are some additional features of tensor flow which come handy to evaluate different model

3.2 Data Exploration

Whenever dealing with any type of data one of the most important steps is understanding what data is trying to tell because understanding the data is one of the most important parts and since we are dealing with two datasets we need to analyze both datasets and try to find how our data is distributed what are the features which are present in each dataset and the first analyze is about the digit dataset, and then we will repeat the same process for fashion dataset.

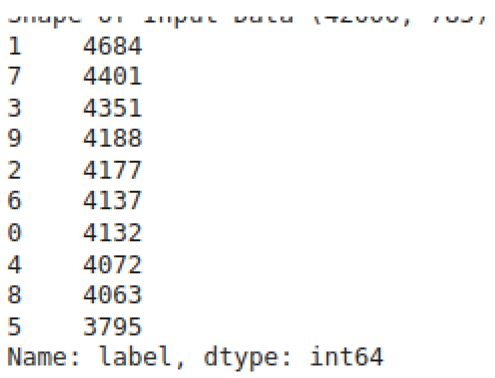
The digit dataset has a total of 60,000 images in training which is also told in the introduction because that is a fixed quantity but since the dataset which we are using is downloaded from Kaggle the whole training data is already split into two parts where training data which has 45,000 images and 25,000 images are given to test dataset, but we can combine both datasets if needed and pass them whole in ones and load test dataset with Keras Library only but 45,000 is also okay because finally, we will be increasing our data with help of data augmentation.



label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	...	pixel774	pixel775	pixel776	pixel777	pixel778	pixel779	pixel780	pixel781	pixel782	pixel783
1	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
1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	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	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

Figure - III Digit Dataset

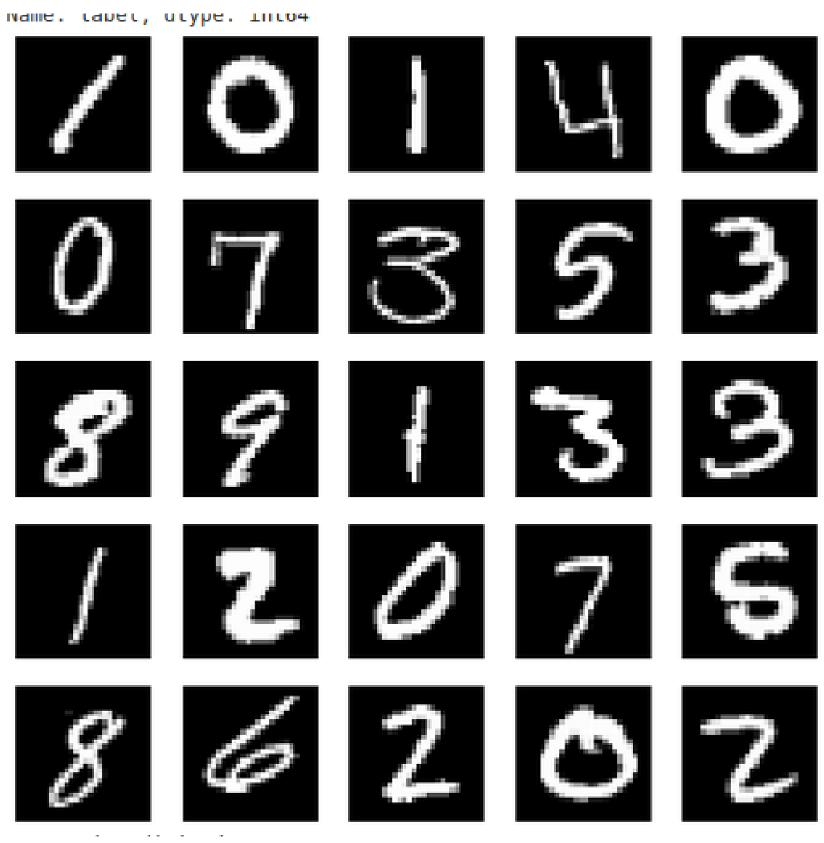
After reading the data checking of distribution of each class over the dataset is one the necessary things and so the first analysis is regarding the count of each digit over our dataset, and it seems each class is not distributed equally as some class like 1 has 4684, whereas some class like 5 has 3795 but since there is not much difference the class in balancing can be ignored at the current scene.



```
1    4684
7    4401
3    4351
9    4188
2    4177
6    4137
0    4132
4    4072
8    4063
5    3795
Name: label, dtype: int64
```

Figure - IV Class Count for digit dataset

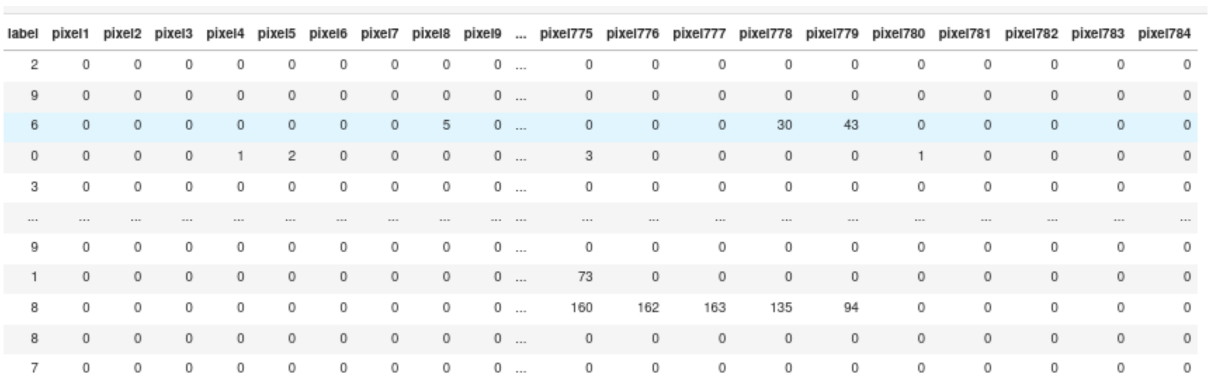
After checking of class in balancing the most important exploration is to check how our input is seen in form of an image because the data imported in the starting is in form of rows and columns and contain value regarding each pixel and plotting that pixel in a single image is important and in the below image we can see a random image which is plotted in a grid manner which shows different digit which is present a how each digit is distributed among each pixel, and it can be seen that each image has some image have some blurred and distortion and hence before passing it to the model there is high need of applying different filer such as Gaussian blur to the image so that even if some noise which is present initially will be removed and we only have required features.



```
Name: label, dtype: int64
```

Figure - V Image representation for digit dataset

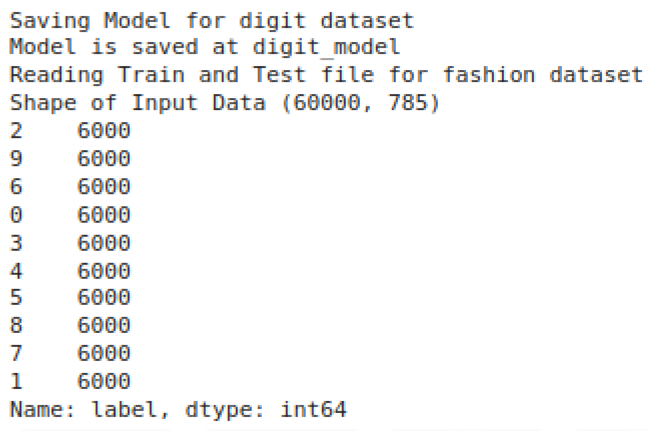
The next dataset which we will be dealing with is the Fashion MNIST dataset which is also owned by MNIST and has a total of 10 different label categories where each label represents a new dataset the knowledge regarding MNIST is already added in the above section and this section, there will be only reading the data in python which is done with help of pandas Data Frame and the fashion dataset is also distributed in columns and rows as same as Digit dataset and a sample of input data is added below which is like a digit, but the pixel value is much different when compared with a digit.



label	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	...	pixel775	pixel776	pixel777	pixel778	pixel779	pixel780	pixel781	pixel782	pixel783	pixel784
2	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	5	0	...	0	0	0	30	43	0	0	0	0	0
0	0	0	0	1	2	0	0	0	0	...	3	0	0	0	0	1	0	0	0	0
3	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	...	73	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	...	160	162	163	135	94	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

Figure - VI Fashion Dataset

As it can be seen that some pixel values are greater than 1 but cannot exceed 256, there is much need of normalizing the value between 0-1 which can be done with help of Standard Scalar which is a normalization technique. The next step will be the same as above where we check the distribution of each class over our dataset, and it seems we have a total of 60,000 training images and each class has a total of 60,000 images which shows a uniform distribution and is the best scenario as our model will be received the same number of image for each class and then the class where the model is making the wrong prediction the data of that class might need to be increased so that model has now more data to learn the pattern.



```
Saving Model for digit dataset
Model is saved at digit_model
Reading Train and Test file for fashion dataset
Shape of Input Data (60000, 785)
2    6000
9    6000
6    6000
0    6000
3    6000
4    6000
5    6000
8    6000
7    6000
1    6000
Name: label, dtype: int64
```

Figure - VII Class Count for Fashion Dataset

As in the above figure, we saw that data is present in rows and columns but the better version of the image dataset is plotting that image on the screen and then how our data is present and the same analysis is added below where the random image is chosen from the whole dataset and plotted with help of Matplotlib which read the whole image in a single run and added in the grid of 5*5.

Figure - VIII Image Representation for Fashion Dataset

Chapter 4 Neural Network

Deep learning techniques revolve mostly around the use of neural networks, which are a subfield of machine learning. They are also known as artificial neural networks (ANNs) or as networks that simulate the behavior of real neural networks (SNNs). Their structure, which is modeled after the way that real neurons interact with one another, takes its name from the human brain and draws its inspiration for its design from the human brain as well. Node layers are the fundamental components of artificial neural networks (ANNs). Some levels, including an input layer, a hidden layer(s), and an output layer(s), make up this structure. Each "node," sometimes called an "artificial neuron," is linked to others through weight and a trigger. The nodes will become active and start sending data to the next layer within the network if its output is higher than the value set as the threshold for activation. No information will be sent to the next network level unless this prerequisite is satisfied.

After being calibrated for precision, however, these learning algorithms become potent tools in artificial intelligence and computer science, allowing us to rapidly classify and cluster data. Several new avenues of inquiry are made possible by this. Tasks requiring speech recognition or image identification may take several minutes rather than hours when compared to the time it takes for human specialists to manually recognize something. One of the most well-known neural networks is the one employed by Google's search method. Therefore, the very first thing that should be done is to import all the necessary modules. In this section, the use of NumPy to compute the values of matrices, Matplotlib to display pictures, and Keras to construct a model of a neural network. In addition to that, the MNIST dataset originates from the Keras framework. Following that, the researcher can begin loading the dataset by using the code shown below. Take into consideration that this might take some time, particularly if this is the very first time to have worked with the MNIST dataset. When the code that follows is executed, they will have four variables at our disposal: X train, y train, X test, and y test, where X represents the picture and y is the target value. These train and test datasets each include 60,000- 10000 photos, with all the images having the same dimensions from the start (28 by 28 pixels).

A sequential model must first be initialized. After that point, they may begin to build upon it. As an initial step, the Neural Network model employs a flattened layer to transform the 2-dimensional (28 x 28 pixel) picture into a 1-dimensional (784-value) representation (1-dimension). Then, use an activation function sigmoid to link these 784 values to 5 neurons. Even though the user may choose whatever number of neurons they want for this layer, the code in this example only uses 5. This is because the developers want the Neural Network model to be easy and quick to train. Finally, we add an output value consisting of a dense layer (in this case, they use the SoftMax activation function. Because there are ten classes to sort through, we'll need ten neurons in the final layer.

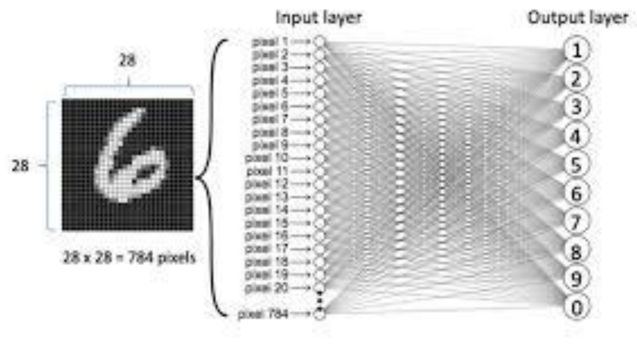


Figure - IX Working of Neural Network

4.1 CNN layers

Convolutional Neural Networks (CNN) are gaining the most popular and common deep learning model which has gained huge popularity in the past decade because of their effective use and all complex model like Alex Net in 2012 too many complex algorithms such as Mobile Net use CNN as its basic layer for better prediction and understanding of image data. CNN is mostly used with image datasets because and is successfully used with different systems like NLP, Recommendation System, and many more. The main advantage behind the use of CNN is that it automatically detects and learn the important feature of the given image without any human help and that is why CNN is so popular when dealing with image dataset in neural network and is also robust to extra noise present in data. All the CNN follow the same mechanize where it uses some convolution and pooling operation applied to the image sequentially and this makes CNN run in both CPU and GPU. The algorithm of CNN works in the manner of taking an input image which will be passed through a series of convolution operations followed by different fully connected neural network layers and then finally making a prediction which can be a multi-class or singular class-based on the end function which is SoftMax.

Convolution is a matrix operation where we try to multiply two different matrixes to form a singular matrix and that can be much better understood by example where we use a convolution filter which is called a kernel and the kernel must be a square matrix which means it should have equal rows and columns and a block of the image which has the same size as the kernel is then multiplied with each block of the kernel and then the output is noted along with multiplication there are other functions such as polling or normalization which is added at end of multiplication and will be more elaborated in future section and slowly the block of the image is the slide from top left corner to right and bottom till reaches bottom right and output of each multiplication operation is said to be the output of CNN and each operation cannot be in linear form and contain some non-linearity. Another name for a convolutional neural network is shift-invariant as we are continuously shifting a block of the matrix in image input and a single convolutional neural network contains an input layer that is fixed and multiple hidden layers a single output layer which is calculated based on input size and kernel size and the activation function used is ReLu which is added in the output layer of our network, In this project, we will be using multiple convolutional neural networks, which works by passing information from one layer to others.

4.2 Pooling layers

Convolutional neural network are applied for better learning of input image and is used to create feature map which will summarize and filter only the required features and remove extra noise from the image, and it proves very effective as there are multiple layer stacks in vertical manner but the limitation of these mapping of output layer is that the position of features in input layer is not always constant and which mean a small change in the position of one features can result in different feature map and these are occurred when we apply data augmentation to the input image which result in change in orientation and in the project we have applied data augmentation to the image and hence there will be change in position of image and our model will not able to understand the features in proper manner and to solve such problem we introduce pooling layer which we down sample our features or reduction is dimension which will lower the resolution of input image and also create keep the features which are present in image and this is generally archived by changing the sides of the layer after the filter is applied and is also helpful in decreasing the non-linearity which is present because of multiplication. Pooling layers are always selected in such a manner that may reduce the output dimension or feature map.

The kernel size of the pooling layer is generally selected which is smaller than the convolution kernel size because our main motive is to decrease the feature map and is mostly selected as 2 x2 which means decreasing the size of rows and columns by 2 other alternatives to size is 3 x3 but is only used when kernel size is large and pooling layer is applied in two manners which are average pooling and maximum pooling. Average Pooling generally calculates the average out of the selected block and then returns that as the value on the feature map, whereas the maximum pooling layer takes the maximum value and returns that to the feature map. The main motive behind the pooling layer is to summarize the feature map to only pick features which are essential and reject other features that are very useful when working with the image where there are regular changes in orientation, and we are adding a pooling layer after each convolutional layer so that only selected features are picked when after data augmentation is applied.

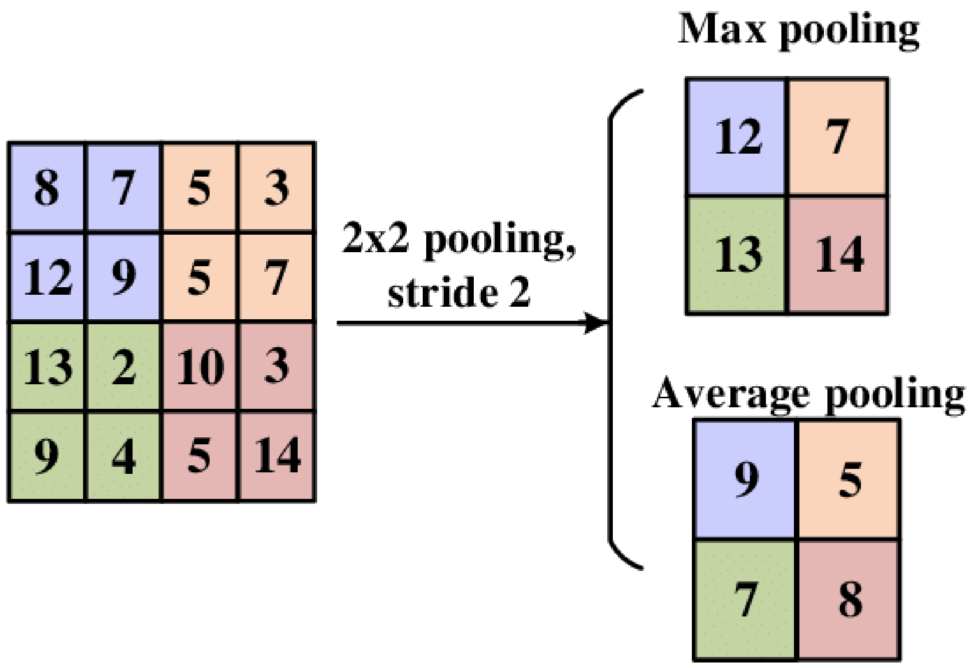


Figure - X Example for Pooling Layer

4.3 Batch Normalization Layers

Neural Network are much complex algorithm as their required large input size and the reason they are too sensitive to the initial weights was are generated randomly and changes during the training and weights highly dependent of the distribution of the input image and hidden layer and weights are changed mostly with help of gradient descent and these can cause the learning algorithm to run in an infinite loop if there are no stopping criteria and this problem is solved by changing the distribution of input layer or by adding a normalization layer which is used to the training of deep neural network normalize each input layer based on batch size and these stabilizing will automatically reduce computation time and also decrease learning time and hence the prediction will more accurate and fast, Normalization are generally used when working with high dense model and are always added after pooling layer as their output need to be normalized before passing to the next layer, and it works by calculating mean and standard deviation of its input and then applying standard normalization to each input and since in our model, there are some convolution layers we need batch normalization which will normalize output of the convolution layer.

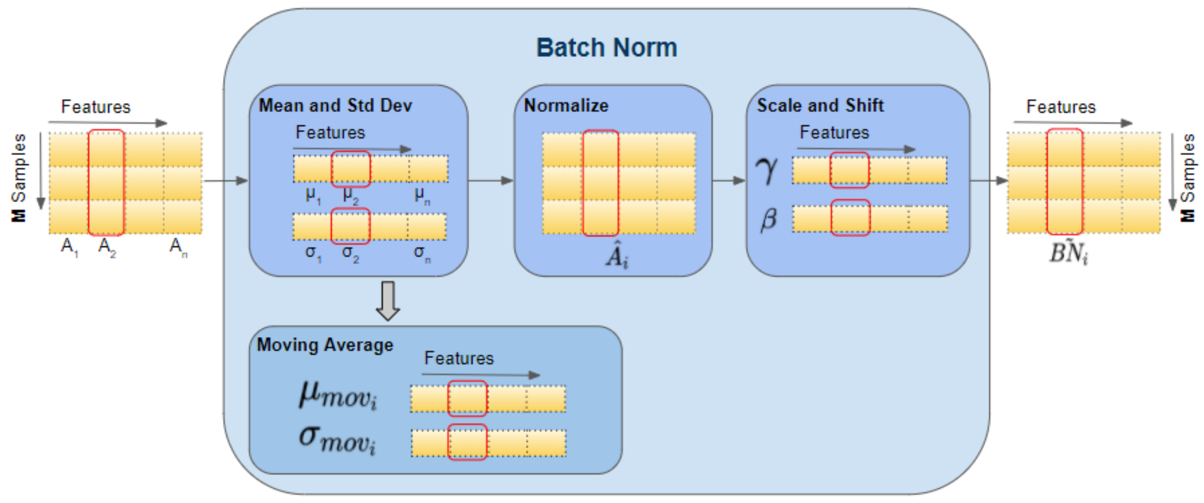


Figure - XI   Example for Batch Normalization

4.4 Dropout Layers

All deep neural networks have different architectures which can vary from simple too much complex or called as deep neural networks which depend on the input image when working with complex models it is usually hard to learn important features from the data and sometimes the noise increase which can't be reduced with pooling and normalization layer, in that case, the model performance is improved with help of drop out layer which also solve the problem of overfitting and this problem is very common when dealing with dataset has a high variation for supposing 10-100 different category and then the drop out layer is used which regularize the model to make a meaningful prediction from all possible prediction and the final output is formed used aggregated function and is also used for reducing data size.

The concept of dropout layer is taken with help of ensemble learning such as Ada boost and xgboost which use multiple models and then either take the average or most commonly occurring class out of all and in the same manner drop out layer work by dropping some layer based on some probability where a probability is calculated between 0 and 1/(1-frequency) such that sum of all the input image is still unchanged and for hidden layer the higher is the dropping probability the sparer it is to the model and a block with probability greater than 0.5 mean it has 50 % probability of dropping and when using these layer we passed threshold value which means every block which has a probability of drop 1 - threshold value should be kept and else it will drop and since nodes are dropping after each iteration the node which is less likely to drop will only be present for the next layer.

4.5 Gaussian Blur

When working with image data, we tend to measure all pixel in the given image so that we can perform different operation over the image for example measuring the pixel or value related to each pixel etc., one of the major processing associated with the image is removal of noise which is added when capturing the image and if unknown noise is removed in proper manner the model will easily understand all features, group of pixel can be classified as edges and several algorithms are developed of edges detection which can also be applied to image so that all the pixel present outside a close figure will be removed but in these section we will mostly be talking about Gaussian blur which work by blurring the given image by multiple a matrix with the input image, the matrix of Gaussian blur is called as Gaussian kernel and applying these blur to the image will reduce all the pixel with some value and pixel with noise will reduce to zero and multiple iteration are run on a single image based on how blur you want your image to be, and these processing can be considered as filter where only object which has small size can pass and hence pixel which are actually feature will only pass, and other pixel will be filtered out and features associated with low spatial and high variation and along with Gaussian blur we added a filter for sharpening the image which is also helpful as the edges of each image will be more visible to the model, and it will make the rightful prediction out of all images, so addition of such filter might result in better accuracy of our final model.

```
blur_kernel_size = (1 / 16.0) * np.array([[1, 2, 1], [2, 4, 2], [1, 2, 1]])

sharpening_kernel_size = np.array([[0, -1, 0], [-1, 5, -1], [0, -1, 0]])

def blur_images(img, kernel_size = blur_kernel_size, itr = 5):
        for i in range(itr):
        img = Conv2D(img, kernel_size, 'same', boundary = 'fill', fillvalue = 0)
        return img
```

4.6 Data Augmentation

One of the importance of Deep learning model is its ability to learn from big data and the deep learning field has grown in past few years and has shown very better result when dealing with different data type such as image, audio and is used in almost every field but since the only drawback of deep learning is it required very big data to learn and the data we are passing contain only 60,000 image and passing small dataset to deep learning can result in overfitting which occur when the model fit exactly against the training data and perform very bad when test with unseen data and the different between training and testing accuracy is very large and hence that model will be of no use as our main focus is regarding test data and such problem is solved with help of Data Augmentation which a technique used to increase the size of training dataset by creating modified data from the existing one, and it is one of the good practice when dealing with deep learning as now we have much larger dataset but apply data augmentation also result in high training time, but the result are very accurate because now model has huge data to deal with and since data augmentation can be applied to different data like audio, image text to increase and since in these project we are only dealing with image we will be discussing the technique of data augmentation which are applied to our given image which include several methods such as colour space transformation which include changing of colour and dealing with RGB channel, or working with geometric transformation which include several methods such as random flipping of image and also cropping of image in correct manner which will result in passing of only selected feature or flipping of image in both horizontal and vertical direction and is helpful when we want our model to learn in more robust manner because in real world the image we get are not always in correct manner and the image in training data is in proper orientation and passing all image with data augmentation is also helpful as now model will get image in different manner, so it will try to learn feature from that also and will result in better understanding and data augmentation is generally not applied to test image because they are considered as a real world data, and we only need to test how our model behave with image which are not seen to it and hence we want to increase our test coverage we won't be applying same calculation along with horizontal rotation and colour changing other processing such as random erasing and rotation which will rotate the input image with random degree but sometime these processing can show negative result as in our dataset we have 6 and 9 where both can be obtained by rotation one with 180 degree hence we need to limit our rotation to only 45 degree so that model might not get confuse, and other processing also need to be applied carefully otherwise our model might give wrong prediction and with help of data augmentation we have convert single image to a copy of total 5 images where each image is generated after applying different image processing and that help us to reduce overfitting and also increase our accuracy. The below image is the resulting image after applying data augmentation to a given image which shows there is a little bit of left rotation and also a little blur which is required but is not good when viewed from human eyes.

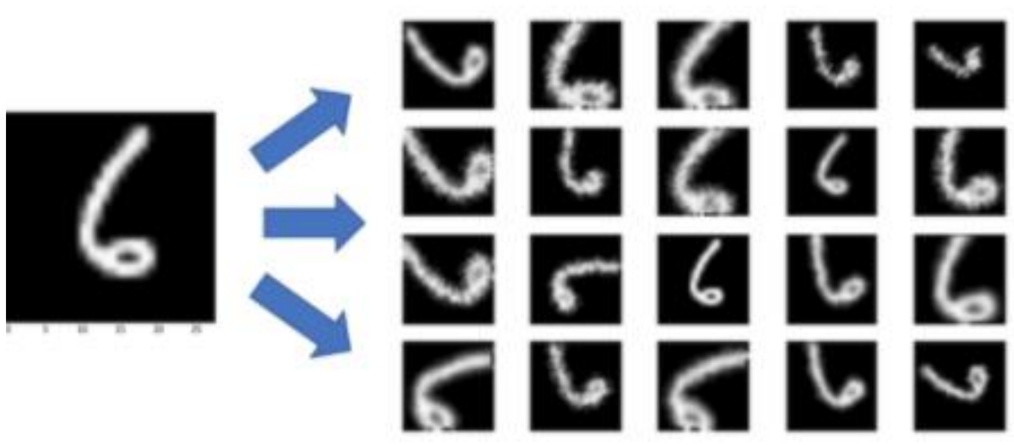


Figure - XII Data Augmentation

```
def data_augmentation():
   datagen = ImageDataGenerator(
       width_shift_range=0.1,
       rotation_range=8,
       zoom_range=0.2,
       height_shift_range=0.1,
       horizontal_flip=True,
       vertical_flip=True,
       fill_mode='nearest')
   return datagen
```

4.7 Loss Function

Deep neural network has high uncertainty and the main function of it is learns and find the output of the given input and since we cannot achieve 100 % accuracy we cannot calculate the exact weights of all layers as there are too many unknown presents in our calculation and the problem the ways of each problem is mostly calculated using an optimization method which work in order to either maximize or minimize the given function and when moving forward in calculation optimization function the model also make good prediction out of it and since the optimization method which the model use is stochastic gradient descent which decrease the gradient of given loss function and then update weights using back propagation of the given algorithm and during, so the ways of each layer depend on the loss function where loss function is pre define in model and the basic definition of any loss function is the objective function which we either want to maximize or minimize in some machine learning algorithm loss function is calculated using the difference between the predicted value and true value and minimizing the difference means moving toward the true value and cost function tend to reduce all the aspect of the given model with the singular value which can be used to justify how good the model is and choosing the loss function is important because all layers weights will be calculated using these function only and sometimes loss function might overfit the data and hence an additional term of regularization is also added with loss function to normalize the calculation of loss value and is the given model the regularization parameter is continuously decreasing with each iteration if there is not much change in accuracy which is an addition call back of TensorFlow.

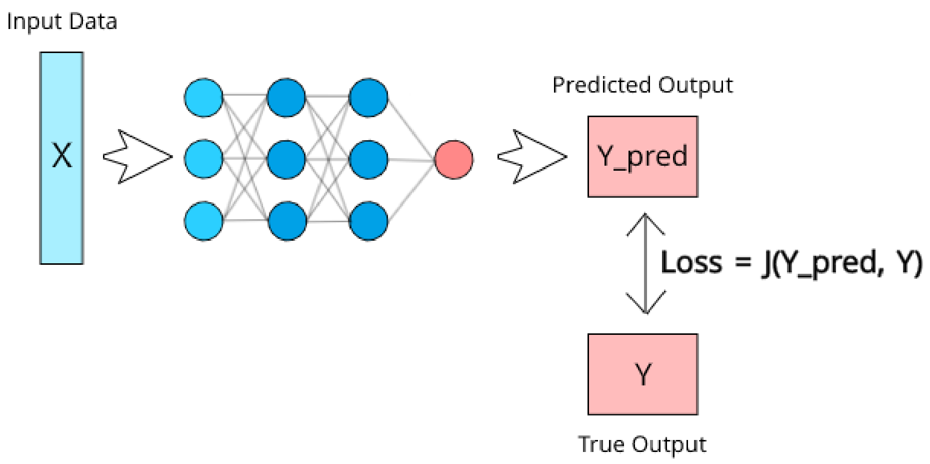


Figure - XIII Loss Function of Neural Network

4.8 Optimization Method

In the above section we talked about the loss function and the basic optimization method in neural network which is called as gradient descent, but we will be using more complex optimization method because using different optimization method may result in different result as it is one of the important part of model and the optimization method used in the model is RMS prop because the gradient descent has one limitation which is that is cannot be used with same learning rate and since we want to continuously change the learning rate with different iteration we would be using RMS Prop which is combined version of gradient boosting and Ada Grade where with each iteration the learning rate is automatically adjusted based on previous study and is calculating using the decaying average of our gradient and is also helpful as our algorithm will only keep track of last n gradient and will remove early gradient from it calculation as there were not very useful and this type of optimization is much helpful when dealing with multi class classification problem and changing the learning rate also increase the coverage rate as now we might reach our end result faster because the value of regularization pattern is mostly depending on last few gradients which accelerate our optimization process or in other words decrease a total number of iteration and the learning rate is control by another parameter called as 'rho' which calculate the momentum of the derivate and decrease how much decrement is needed in step size.

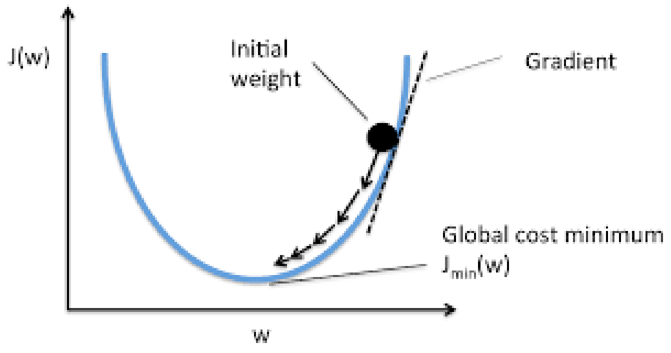


Figure - XIV   Optimization Method

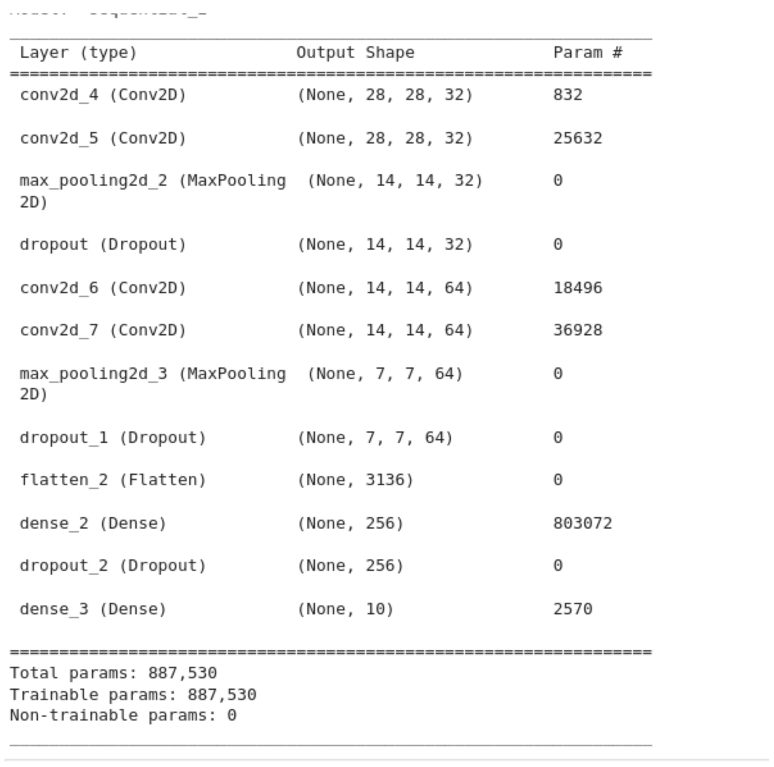
4.9 TensorFlow Callback

TensorFlow provide addition features in which one is callback which are mostly used when you want to control the learning of models and also prevent overfitting other important of callback will be talked in these section, In each language callback are the function which are wrapper around TensorFlow model and are used for many different purposes like stop future different iteration if there is not much change in accuracy or when the loss function curve is not changing that mean we have achieved the minimum loss function and continue training will not affect the model, so it is best to stop the iteration such features are enabled when we add some callback from TensorFlow to the model, Other call back which are added in the model are Tensor board Visualization which will automatically save the output of each iteration in the log and can be directly imported in Tensor board and provide a visualization how our model is trained during different iteration, the last callback which was added in our model is saving the best model during the iteration. Since we know each iteration of our model will calculate weight and in some iterations, we might get the best accuracy, this calls back will automatically save the weights of the iteration where the maximum accuracy was achieved and that is how we will have the most optimized model saved automatically another benefit of the callback is a user can define customer callback and can add it to a model which will automatically run at the end of each iteration and provide all useful information which user needs.

```
def get_callback_and_optimizer(learning_rate=0.01, rho_value=0.9, epsilon_value=1e-08,
patience_level=3, v=1, minimum_learning_rate=0.0001, model_filename='', *args, **kwargs):
   rms_opt = RMSprop(learning_rate=learning_rate, rho=rho_value, epsilon=epsilon_value)
   lr_reduction = ReduceLROnPlateau(monitor='val_accuracy' patience=patience_level verbose=v,
                                    min_lr=minimum_learning_rate)
   early_stop = EarlyStopping(patience=patience_level, monitor='val_accuracy')
   model_checkout = ModelCheckpoint(monitor='val_accuracy', save_best_only=True,
filepath=model_filename)
   logdir="logs/" + datetime.now().strftime("%Y%m%d-%H%M%S")
   tensorboard_callback = TensorBoard(log_dir=logdir)
   return [rms_opt, lr_reduction, early_stop, model_checkout, tensorboard_callback]
```

4.10 Model Architecture

All the above layer talked in the above section are combined in a single model where we have all the model stack vertical where the output of the previous layer will act as an input to the current layer and as we can see we have a total of 887,530 parameters in which all weights at the starting is unset and can have any value and out input layer which is cov2d_4 which take input of 28x28 which is the size of the image and consist many hidden layers and an output layer has total 10 different layers which represent our total final output layer and each layer the below plot is a much simpler form of our model and ignore all the inner mathematical which will run internally and since we will be using the same model architecture for both the data set we must make our model more modular and need to change input size if need to 28x28 which is 784 pixel and output layers as 10.


```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_4 (Conv2D)           (None, 28, 28, 32)        832

 conv2d_5 (Conv2D)           (None, 28, 28, 32)        25632

 max_pooling2d_2 (MaxPooling  (None, 14, 14, 32)       0
 2D)

 dropout (Dropout)           (None, 14, 14, 32)        0

 conv2d_6 (Conv2D)           (None, 14, 14, 64)        18496

 conv2d_7 (Conv2D)           (None, 14, 14, 64)        36928

 max_pooling2d_3 (MaxPooling  (None, 7, 7, 64)         0
 2D)

 dropout_1 (Dropout)         (None, 7, 7, 64)          0

 flatten_2 (Flatten)         (None, 3136)              0

 dense_2 (Dense)             (None, 256)               803072

 dropout_2 (Dropout)         (None, 256)               0

 dense_3 (Dense)             (None, 10)                2570

=================================================================
Total params: 887,530
Trainable params: 887,530
Non-trainable params: 0
_________________________________________________________________
```

Figure - XV   Neural Network Model Summary

The below figure is generated with help of the Tensor board and contains the inner calculation that TensorFlow handles internally, and we don't need to worry, but it is best to look through each layer because that is how any model can be debugged and can achieve better accuracy, and it seems the model has all the required layer in the right dimension, and we are ready to make a prediction

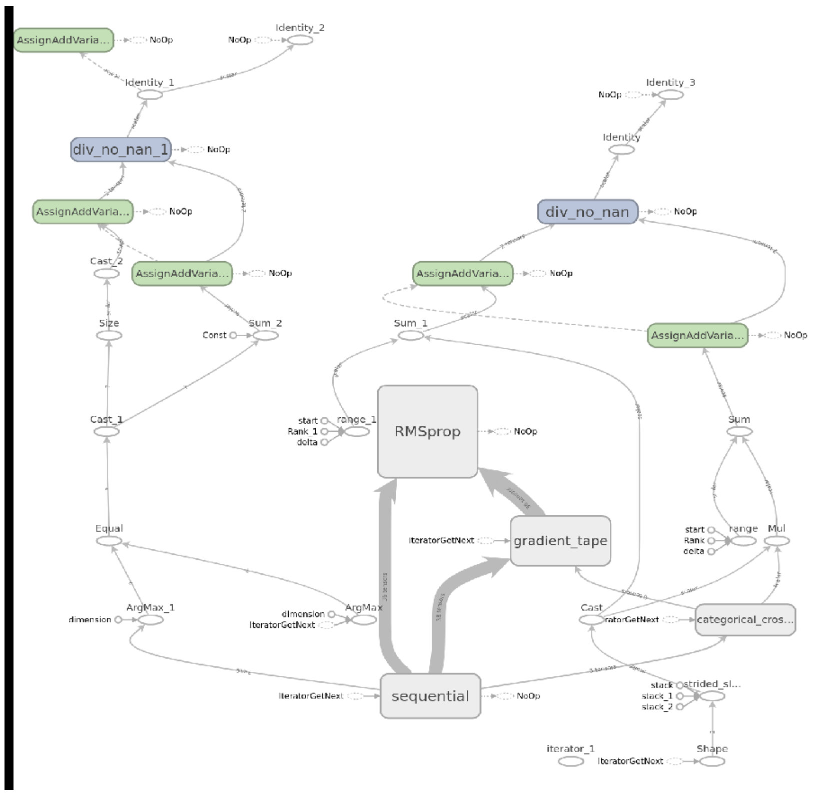


Figure - XVI   Neural Network Model Summary by Tensorboard

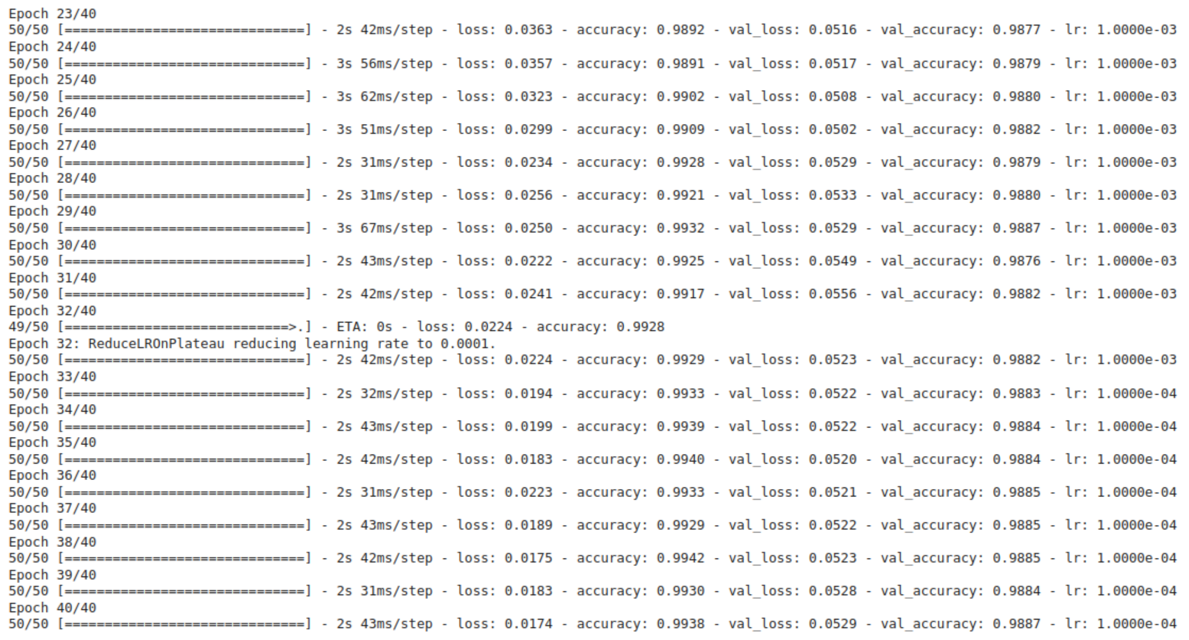
```python
def create_model(number_of_class=10, name='Digit Dataset', *args, **kwargs):
   if 'first' in kwargs:  # advance Model
       dropout_value = 0.5
       filer_count = 32
       model = Sequential()
       # encoding
       model.add(Conv2D(filters=filer_count, kernel_size=(5, 5), padding='Same',
                        activation='relu', input_shape=(28, 28, 1)))
       model.add(Conv2D(filters=filer_count, kernel_size=(5, 5), padding='Same',
                        activation='relu'))
       model.add(MaxPool2D(pool_size=(2, 2)))
       model.add(Dropout(0.25))
       # decoding
       model.add(Conv2D(filters=filer_count * 2, kernel_size=(3, 3), padding='Same',
                        activation='relu'))
       model.add(Conv2D(filters=filer_count * 2, kernel_size=(3, 3), padding='Same',
                        activation='relu'))
       model.add(MaxPool2D(pool_size=(2, 2), strides=(2, 2)))
       model.add(Dropout(dropout_value / 2))
```

```
        model.add(Flatten())
        model.add(Dense(256, activation="relu"))
        model.add(Dropout(dropout_value))
        model.add(Dense(number_of_class, activation="softmax"))
    else:  # default model
        model = Sequential()
        model.add(Conv2D(64, kernel_size=3, activation='relu', input_shape=(28, 28, 1)))
        model.add(Conv2D(32, kernel_size=3, activation='relu'))
        model.add(MaxPool2D(pool_size=(2, 2), strides=None, padding='valid', data_format=None))
        model.add(Flatten())
        model.add(Dense(number_of_class, activation='softmax'))
    return model
```

Chapter 5 Result & Discussion

The last section in the project is evacuating the given model which includes plotting all different metrics and graphs which are essential to govern how good our model is in making predictions the model is running for a total of 40 iterations which is an optimized number and the output of last 20 iterations is added below and since there are multiple call-backs associated with each model there are high chances that our fitting may step before 40 iterations but since we have to keep our threshold value to be 0.001 which is quite small therefore the model run for all 40 iterations and fitting all 40 iterations in a single image is tough, but the output of each cell is present in the notebook for help.

First, we will evaluate our digit dataset which is a handwritten digit and consist total of 10 different digits from 0 to 9 where a single image has total 748 pixels and is in two dimension array which mean image is in grayscale colour and total 192,000,000 image were passed to the given model and all image where processed so to only capture the features which are important to the model selected for training and prediction is based on neural network and convolution network which perform the best when dealing with image dataset and after fitting of the model to the training dataset we were able to achieve the best accuracy of 0.994 which means the model was able to predict correctly 994 times out of 1000 which is quite good since the model we are not much complicated and contain only basic neural network layer and along with the current model we have added a simple model which consist of only 5 layers and the accuracy achieved with that model was 0.966 which is low as compare to our current model, and we will be neglecting the model but is added in notebook for help. When moving forward the accuracy of the model was continuously increasing and the loss function was continuously decreasing which indicated that our model is learning in the right direction and there is no overfitting and under fitting of the model in the given dataset and the plot the graph and since loss function selected for the model is gradient descent and optimization method is RMS prop which also solve the problem of regularization and to check if our hypothesis is true we stored accuracy and loss value after each iteration in a python list and after the completion of training we create a simple graphical representation of both loss value and accuracy value with help of Matplotlib for our current dataset which is digit dataset and the graph are added below and from the below graph is can be seen that if we keep on increase our iteration value to 60-70 we might get better accuracy and much more optimized model but that another test run which can be done but since each iteration take 40 second when trained on GPU of 2 GB RAM and if we choose a simple CPU it will take around 5 min to run a single iteration and running 60 iteration mean around 2 hours but with the help of call-back which are added before training if there is not much change in the accuracy of the model the training will stop and that how helpful each component are when training such model.



```
Epoch 23/40
50/50 [==============================] - 2s 42ms/step - loss: 0.0363 - accuracy: 0.9892 - val_loss: 0.0516 - val_accuracy: 0.9877 - lr: 1.0000e-03
Epoch 24/40
50/50 [==============================] - 3s 56ms/step - loss: 0.0357 - accuracy: 0.9891 - val_loss: 0.0517 - val_accuracy: 0.9879 - lr: 1.0000e-03
Epoch 25/40
50/50 [==============================] - 3s 62ms/step - loss: 0.0323 - accuracy: 0.9902 - val_loss: 0.0508 - val_accuracy: 0.9880 - lr: 1.0000e-03
Epoch 26/40
50/50 [==============================] - 3s 51ms/step - loss: 0.0299 - accuracy: 0.9909 - val_loss: 0.0502 - val_accuracy: 0.9882 - lr: 1.0000e-03
Epoch 27/40
50/50 [==============================] - 2s 31ms/step - loss: 0.0234 - accuracy: 0.9928 - val_loss: 0.0529 - val_accuracy: 0.9879 - lr: 1.0000e-03
Epoch 28/40
50/50 [==============================] - 2s 31ms/step - loss: 0.0256 - accuracy: 0.9921 - val_loss: 0.0533 - val_accuracy: 0.9880 - lr: 1.0000e-03
Epoch 29/40
50/50 [==============================] - 3s 67ms/step - loss: 0.0250 - accuracy: 0.9932 - val_loss: 0.0529 - val_accuracy: 0.9887 - lr: 1.0000e-03
Epoch 30/40
50/50 [==============================] - 2s 43ms/step - loss: 0.0222 - accuracy: 0.9925 - val_loss: 0.0549 - val_accuracy: 0.9876 - lr: 1.0000e-03
Epoch 31/40
50/50 [==============================] - 2s 42ms/step - loss: 0.0241 - accuracy: 0.9917 - val_loss: 0.0556 - val_accuracy: 0.9882 - lr: 1.0000e-03
Epoch 32/40
49/50 [============================>.] - ETA: 0s - loss: 0.0224 - accuracy: 0.9928
Epoch 32: ReduceLROnPlateau reducing learning rate to 0.0001.
50/50 [==============================] - 2s 42ms/step - loss: 0.0224 - accuracy: 0.9929 - val_loss: 0.0523 - val_accuracy: 0.9882 - lr: 1.0000e-03
Epoch 33/40
50/50 [==============================] - 2s 32ms/step - loss: 0.0194 - accuracy: 0.9933 - val_loss: 0.0522 - val_accuracy: 0.9883 - lr: 1.0000e-04
Epoch 34/40
50/50 [==============================] - 2s 43ms/step - loss: 0.0199 - accuracy: 0.9939 - val_loss: 0.0522 - val_accuracy: 0.9884 - lr: 1.0000e-04
Epoch 35/40
50/50 [==============================] - 2s 42ms/step - loss: 0.0183 - accuracy: 0.9940 - val_loss: 0.0520 - val_accuracy: 0.9884 - lr: 1.0000e-04
Epoch 36/40
50/50 [==============================] - 2s 31ms/step - loss: 0.0223 - accuracy: 0.9933 - val_loss: 0.0521 - val_accuracy: 0.9885 - lr: 1.0000e-04
Epoch 37/40
50/50 [==============================] - 2s 43ms/step - loss: 0.0189 - accuracy: 0.9929 - val_loss: 0.0522 - val_accuracy: 0.9885 - lr: 1.0000e-04
Epoch 38/40
50/50 [==============================] - 2s 42ms/step - loss: 0.0175 - accuracy: 0.9942 - val_loss: 0.0523 - val_accuracy: 0.9885 - lr: 1.0000e-04
Epoch 39/40
50/50 [==============================] - 2s 31ms/step - loss: 0.0183 - accuracy: 0.9930 - val_loss: 0.0528 - val_accuracy: 0.9884 - lr: 1.0000e-04
Epoch 40/40
50/50 [==============================] - 2s 43ms/step - loss: 0.0174 - accuracy: 0.9938 - val_loss: 0.0529 - val_accuracy: 0.9887 - lr: 1.0000e-04
```

Figure - XVII   Training Model for digit dataset

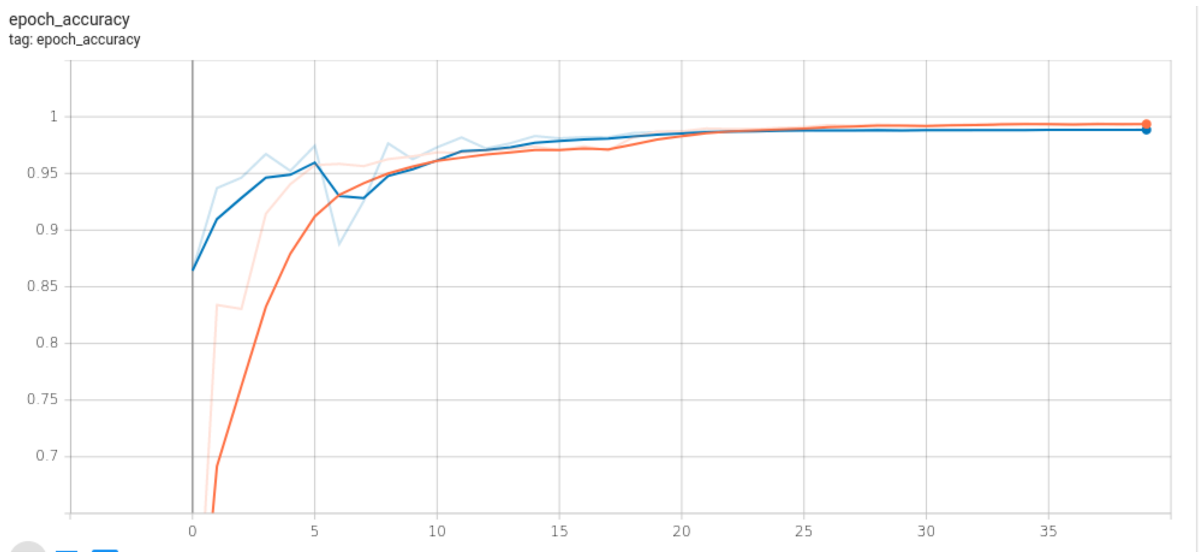


Figure - XVIII   Accuracy for different Iteration for digit dataset

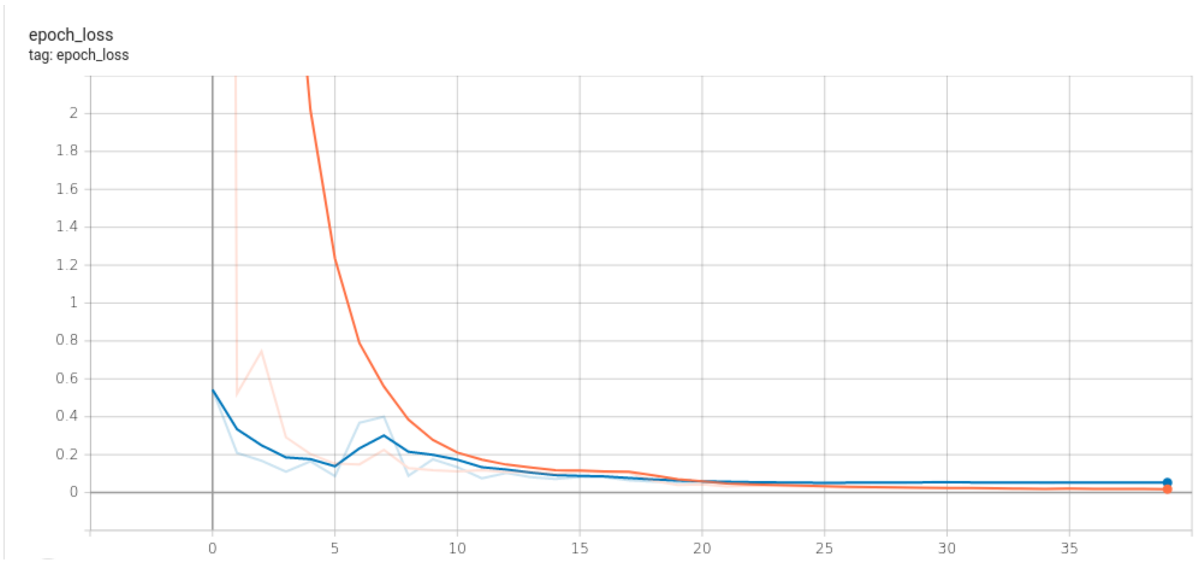


Figure - XIX   Loss for different Iteration for digit dataset

From the above graph we can see that the loss function at the starting of the iteration was around 2 and till it reach the iteration 35 it was approaching toward 0 which is quite good because our main motive was to minimize the loss function and since minimum value of loss function is zero because it is absolute difference between true value and predicted value, so it can take value from 0 to positive infinity and exact value of 0 indicated we have achieved accuracy of 100% which is not possible in the real world scenario and that the reason we have chosen iteration number as 40 because passing higher value mean that model might achieve 100% accuracy which can be another case of overfitting, After the training of the model we have saved the model in the working directory and then used the same model to check how good the our model will be when passed some image from test dataset and in the below image we have selected random 20 image from the test dataset and plotted them in a single grid along with the predicted value and actual value, and it seems from the below image that prediction of all 20 random image where 100 % true and how good our model is.

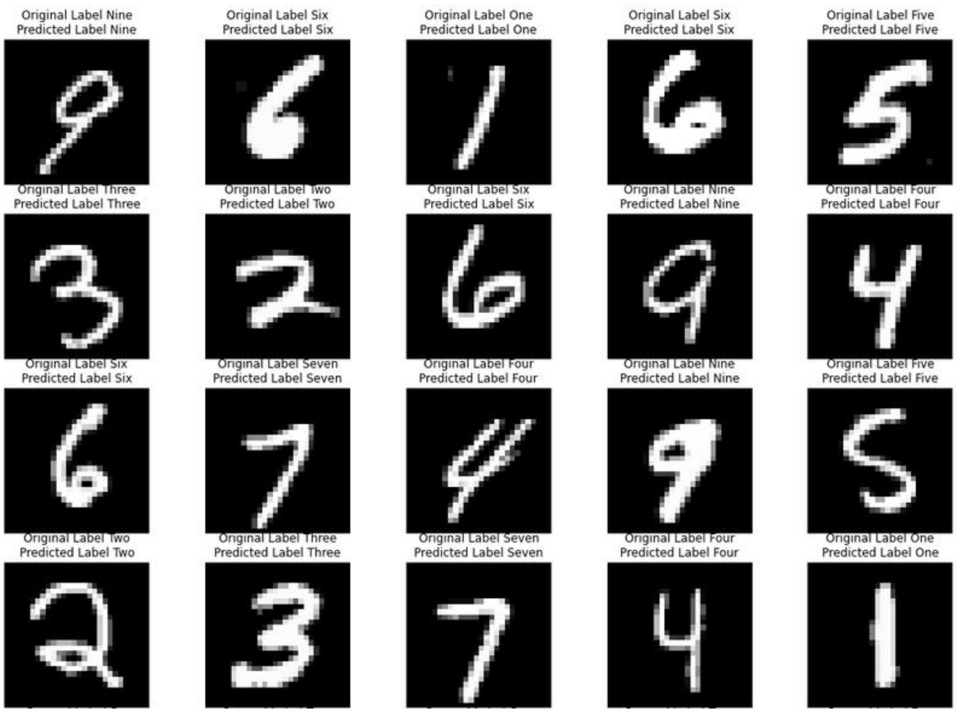
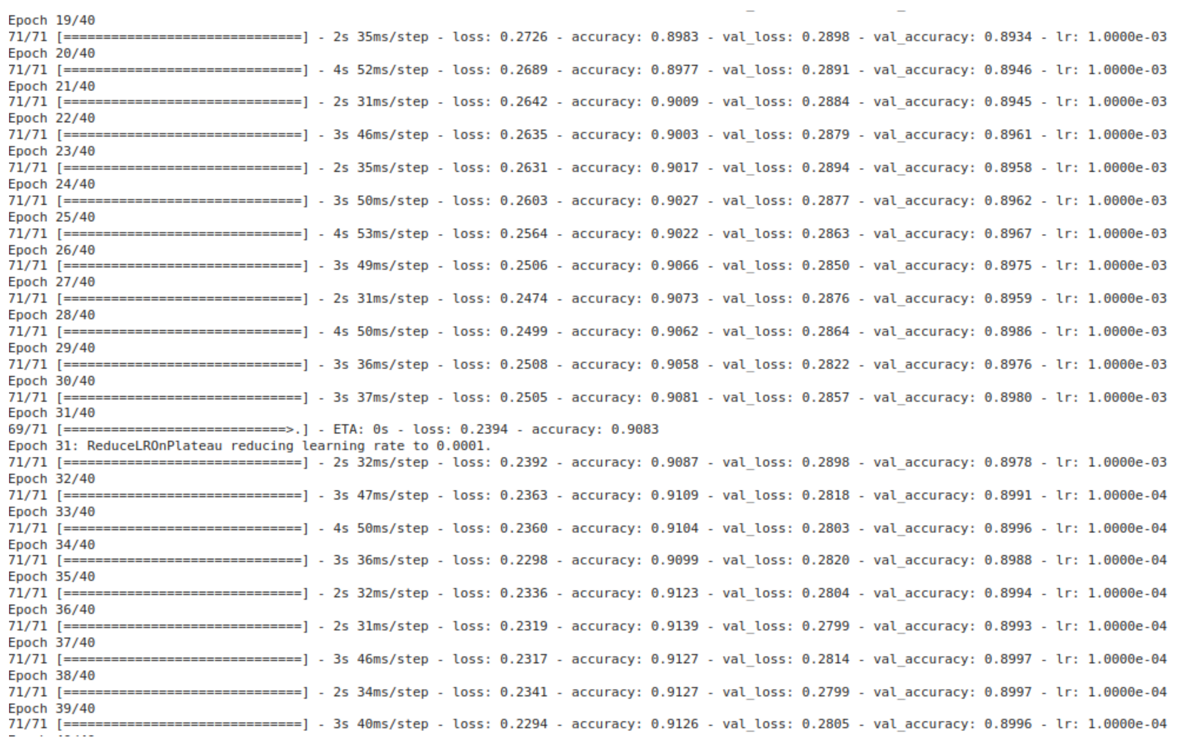


Figure - XX Final Prediction for digit dataset

All the above result were linked to the digit dataset, and now we will be evaluating the second dataset which is the fashion dataset which also has total 10 different category and names of each category is also added above for references and since the fashion dataset is a little bit complicated when compare with digit as it contain different dress but still has a much more complex image and is run for a total of 40 epochs as same as above and the best accuracy which we were able to achieve was 0.912 which mean we make a right prediction for 912 images out of 1000 which is also good considering the variation in fashion dataset and these show that same model when passed with different dataset show different accuracy and the weights of both model will differ a lot even though everything is same and that show how much a model is dependent on the dataset, The loss and accuracy value at different iteration is also added in below image which shows that with future iteration model might achieve much greater accuracy.



```
Epoch 19/40
71/71 [==============================] - 2s 35ms/step - loss: 0.2726 - accuracy: 0.8983 - val_loss: 0.2898 - val_accuracy: 0.8934 - lr: 1.0000e-03
Epoch 20/40
71/71 [==============================] - 4s 52ms/step - loss: 0.2689 - accuracy: 0.8977 - val_loss: 0.2891 - val_accuracy: 0.8946 - lr: 1.0000e-03
Epoch 21/40
71/71 [==============================] - 2s 31ms/step - loss: 0.2642 - accuracy: 0.9009 - val_loss: 0.2884 - val_accuracy: 0.8945 - lr: 1.0000e-03
Epoch 22/40
71/71 [==============================] - 3s 46ms/step - loss: 0.2635 - accuracy: 0.9003 - val_loss: 0.2879 - val_accuracy: 0.8961 - lr: 1.0000e-03
Epoch 23/40
71/71 [==============================] - 2s 35ms/step - loss: 0.2631 - accuracy: 0.9017 - val_loss: 0.2894 - val_accuracy: 0.8958 - lr: 1.0000e-03
Epoch 24/40
71/71 [==============================] - 3s 50ms/step - loss: 0.2603 - accuracy: 0.9027 - val_loss: 0.2877 - val_accuracy: 0.8962 - lr: 1.0000e-03
Epoch 25/40
71/71 [==============================] - 4s 53ms/step - loss: 0.2564 - accuracy: 0.9022 - val_loss: 0.2863 - val_accuracy: 0.8967 - lr: 1.0000e-03
Epoch 26/40
71/71 [==============================] - 3s 49ms/step - loss: 0.2506 - accuracy: 0.9066 - val_loss: 0.2850 - val_accuracy: 0.8975 - lr: 1.0000e-03
Epoch 27/40
71/71 [==============================] - 2s 31ms/step - loss: 0.2474 - accuracy: 0.9073 - val_loss: 0.2876 - val_accuracy: 0.8959 - lr: 1.0000e-03
Epoch 28/40
71/71 [==============================] - 4s 50ms/step - loss: 0.2499 - accuracy: 0.9062 - val_loss: 0.2864 - val_accuracy: 0.8986 - lr: 1.0000e-03
Epoch 29/40
71/71 [==============================] - 3s 36ms/step - loss: 0.2508 - accuracy: 0.9058 - val_loss: 0.2822 - val_accuracy: 0.8976 - lr: 1.0000e-03
Epoch 30/40
71/71 [==============================] - 3s 37ms/step - loss: 0.2505 - accuracy: 0.9081 - val_loss: 0.2857 - val_accuracy: 0.8980 - lr: 1.0000e-03
Epoch 31/40
69/71 [============================>.] - ETA: 0s - loss: 0.2394 - accuracy: 0.9083
Epoch 31: ReduceLROnPlateau reducing learning rate to 0.0001.
71/71 [==============================] - 2s 32ms/step - loss: 0.2392 - accuracy: 0.9087 - val_loss: 0.2898 - val_accuracy: 0.8978 - lr: 1.0000e-03
Epoch 32/40
71/71 [==============================] - 3s 47ms/step - loss: 0.2363 - accuracy: 0.9109 - val_loss: 0.2818 - val_accuracy: 0.8991 - lr: 1.0000e-04
Epoch 33/40
71/71 [==============================] - 4s 50ms/step - loss: 0.2360 - accuracy: 0.9104 - val_loss: 0.2803 - val_accuracy: 0.8996 - lr: 1.0000e-04
Epoch 34/40
71/71 [==============================] - 3s 36ms/step - loss: 0.2298 - accuracy: 0.9099 - val_loss: 0.2820 - val_accuracy: 0.8988 - lr: 1.0000e-04
Epoch 35/40
71/71 [==============================] - 2s 32ms/step - loss: 0.2336 - accuracy: 0.9123 - val_loss: 0.2804 - val_accuracy: 0.8994 - lr: 1.0000e-04
Epoch 36/40
71/71 [==============================] - 2s 31ms/step - loss: 0.2319 - accuracy: 0.9139 - val_loss: 0.2799 - val_accuracy: 0.8993 - lr: 1.0000e-04
Epoch 37/40
71/71 [==============================] - 3s 46ms/step - loss: 0.2317 - accuracy: 0.9127 - val_loss: 0.2814 - val_accuracy: 0.8997 - lr: 1.0000e-04
Epoch 38/40
71/71 [==============================] - 2s 34ms/step - loss: 0.2341 - accuracy: 0.9127 - val_loss: 0.2799 - val_accuracy: 0.8997 - lr: 1.0000e-04
Epoch 39/40
71/71 [==============================] - 3s 40ms/step - loss: 0.2294 - accuracy: 0.9126 - val_loss: 0.2805 - val_accuracy: 0.8996 - lr: 1.0000e-04
```

Figure - XXI Training Model for fashion dataset

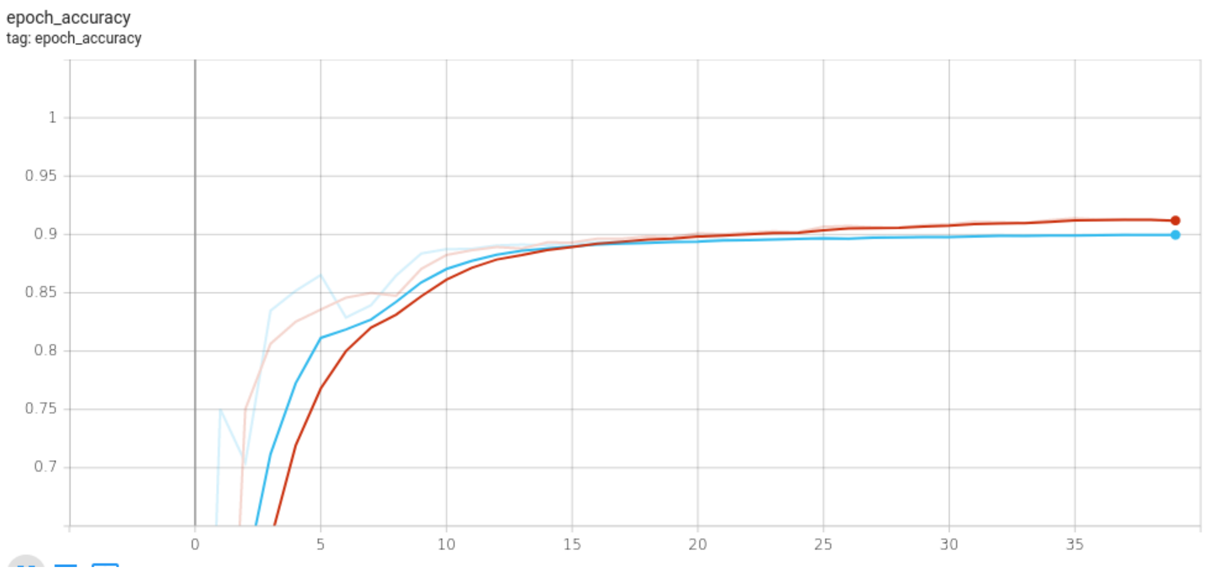


Figure - XXII Accuracy for different Iteration for fashion dataset

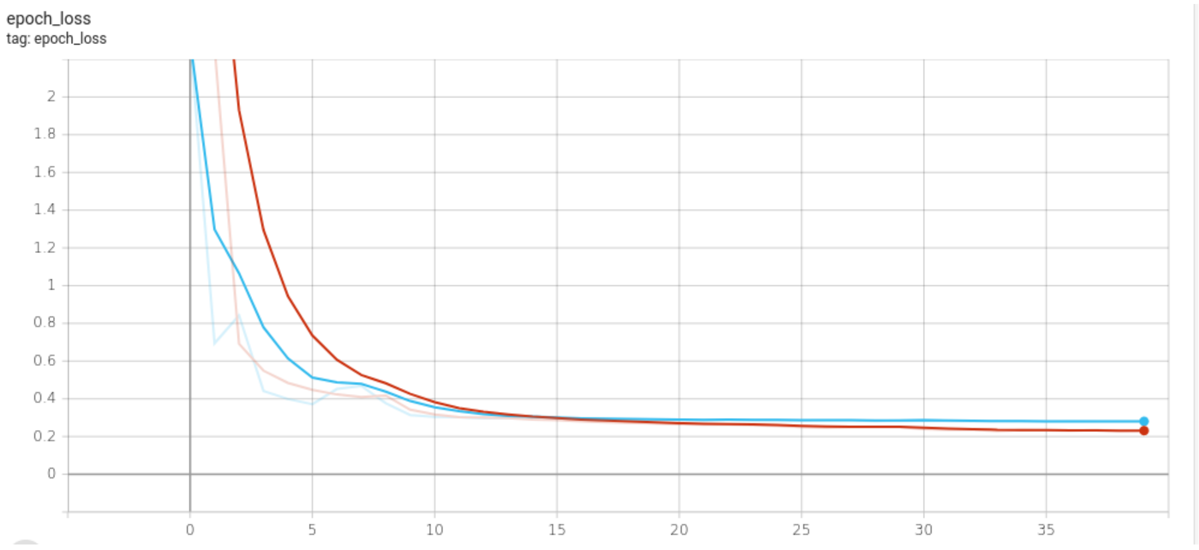


Figure - XXIII Loss for different Iteration for fashion dataset

In the below image, we have chosen some random images from our dataset and plotted the image along with their true value and prediction, and as we can see we are predicting the right class for all the given images which is good considering the complexity of the model.

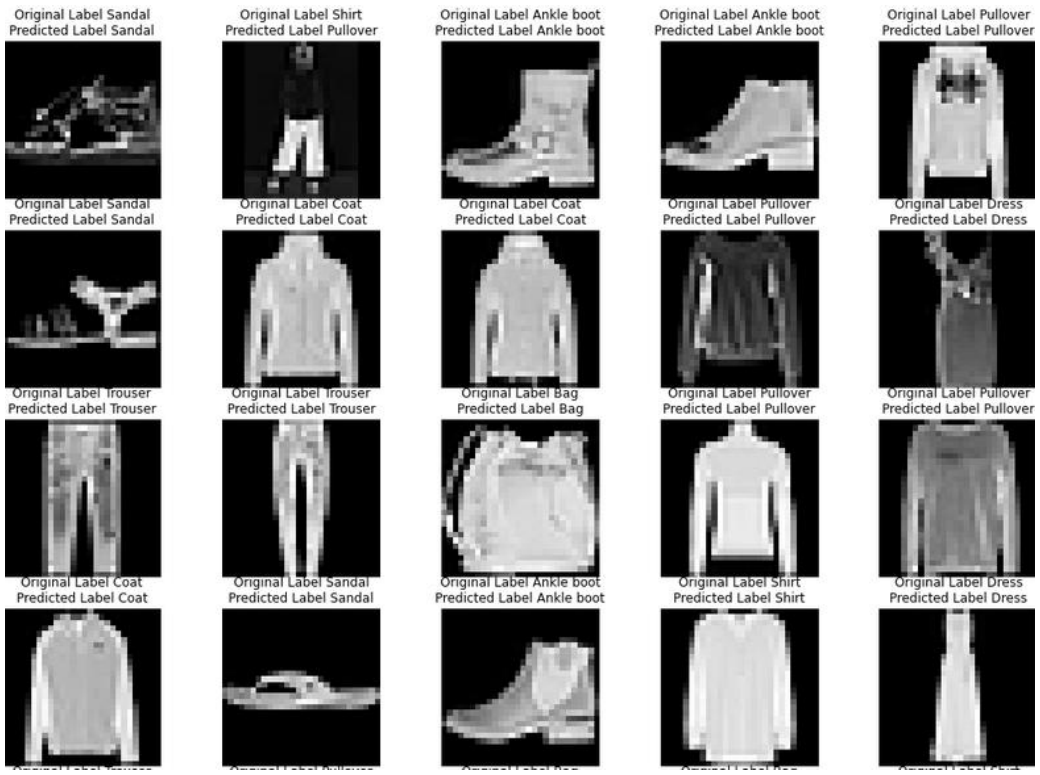


Figure - XXIV Final Prediction for fashion dataset

```
defevaluate_model(model, history, X, y, expected_values, *args, **kwargs):
   fig, ax = plt.subplots(2, 1)
   ax[0].plot(history.history['loss'], color='b', label="Training loss")
   ax[0].plot(history.history['val_loss'], color='r', label="validation loss", axes=ax[0])
   ax[0].legend(loc='best', shadow=True)

   ax[1].plot(history.history['accuracy'], color='b', label="Training accuracy")
   ax[1].plot(history.history['val_accuracy'], color='r', label="Validation accuracy")
   ax[1].legend(loc='best', shadow=True)
plt.show()
Y_pred = model.predict(X)
# Convert predictions classes to one hot vector
Y_pred_classes = np.argmax(Y_pred, axis=1)
# Convert validation observations to one hot vector
Y_true = np.argmax(y, axis=1)

plot_final_images(X, Y_true, Y_pred, 5, 5, expected_values)
   matrix = confusion_matrix(Y_true, Y_pred_classes)
plot_confusion_matrix(matrix, expected_value=expected_values)
```

Chapter 6 Conclusions

Learning from past data is how any machine learning model learn and in these project we have made use of that for image classification and recognition where two different datasets are used which are MNIST HandWritten Digit dataset and MNIST Fashion dataset where both dataset has total 10 different class and total 60,000 input image are used where each input has 28x28 pixel and each pixel has value between 0-256 and each image is passed through series of preprocessing which include normalizing the image, reshaping the image and finally apply different filter to image to reduce noise and finally image in both dataset is passed through a Neural Network which is based on concept of convolution layer which contain single input layer and single output layer but multiple hidden layer and RMS prop is chosen as the optimization method and both dataset have the different model trained, but the architecture is kept same which is important as we need to study how same model behave when pass through different dataset, and it seems that with the digit dataset we were able to achieved accuracy of 99.4 and with fashion dataset the maximum accuracy achieved was 91.5 which is lower for fashion dataset as the image are much complex and in order to achieve much higher accuracy we might need much complex network such as Dense Net or Mobile Net which work best for such dataset.

References

"THE MNIST DATABASE of handwritten digits".Yann LeCun, Courant Institute, NYU Corinna Cortes, Google Labs, New York Christopher J.C. Burges, Microsoft Research, Redmond.

Abiodun, O.I., Jantan, A., Omolara, A.E., Dada, K.V., Mohamed, N.A. and Arshad, H., 2018. State-of-the-art in artificial neural network applications: A survey. *Heliyon*, *4*(11), p.e00938.

An, S., Lee, M., Park, S., Yang, H. and So, J., 2020. An ensemble of simple convolutional neural network models for mnist digit recognition. *arXiv preprint arXiv:2008.10400*.

Annarumma, M., Withey, S.J., Bakewell, R.J., Pesce, E., Goh, V., and Montana, G., 2019. Automated triaging of adult chest radiographs with deep artificial neural networks. *Radiology*, *291*(1), p.196.

Azarmdel, H., Jahanbakhshi, A., Mohtasebi, S.S. and Muñoz, A.R., 2020. Evaluation of image processing technique as an expert system in mulberry fruit grading based on ripeness level using artificial neural networks (ANNs) and support vector machine (SVM). *Postharvest Biology and Technology*, *166*, p.111201.

Barrett, D.G., Morcos, A.S. and Macke, J.H., 2019. Analyzing biological and artificial neural networks: challenges with opportunities for synergy. *Current opinion in neurobiology*, *55*, pp.55-64.

Bostrom, Nick (2011)."The Ethics of Artificial Intelligence" (PDF). Archived from the (PDF) on 4 March 2016. Retrieved 11 April 2016.

Chaganti, S.Y., Nanda, I., Pandi, K.R., Prudhvith, T.G. and Kumar, N., 2020, March. Image Classification using SVM and CNN. In *2020 International Conference on Computer Science, Engineering and Applications (ICCSEA)* (pp. 1-5). IEEE.

Chen, F., Chen, N., Mao, H. and Hu, H., 2018. Assessing four neural networks on a handwritten digit recognition dataset (MNIST). *arXiv preprint arXiv:1811.08278*.

Chicco, D., 2021. Siamese neural networks: An overview. *Artificial Neural Networks*, pp.73-94.

Ciresan, Dan Claudiu; Ueli Meier; Luca Maria Gambardella; Jürgen Schmidhuber (2011)."Convolutional neural network committees for handwritten character classification" (PDF). *2011 International Conference on Document Analysis and Recognition (ICDAR)*. pp. 1135–1139.

D. C. Ciresan, U. Meier,J. Schmidhuber. Multi-column Deep Neural Networks for Image Classification. IEEE Conf. on Computer Vision and Pattern Recognition CVPR 2012.

Datadriveninvestor, 2022, *Image Processing for MNIST using Keras*, available at: https://medium.datadriveninvestor.com/image-processing-for-mnist-using-keras-f9a1021f6ef0, [accessed on: 17/09/2022]

De Araujo Faria, V., Azimbagirad, M., VianiArruda, G., FernandesPavoni, J., Cezar Felipe, J., dos Santos, E.M.C.M.F. and Murta Junior, L.O., 2021. Prediction of radiation-related dental caries through pyradiomics features and artificial neural network on panoramic radiography. *Journal of Digital Imaging*, *34*(5), pp.1237-1248.

Digitalocean, 2022, *MNIST Dataset in Python - Basic Importing and Plotting*, available at: https://www.digitalocean.com/community/tutorials/mnist-dataset-in-python, [accessed on: 17/09/2022]

Feiyang Chen, *Assessing Four Neural Networks on Handwritten Digit Recognition Dataset:* CHUANGXINBAN JOURNAL OF COMPUTING, JUNE 2018

Fourcade, A. and Khonsari, R.H., 2019. Deep learning in medical image analysis: A third eye for doctors. *Journal of stomatology, oral and maxillofacial surgery*, *120*(4), pp.279-288.

Greeshma, K.V., and Sreekumar, K., 2019. Hyperparameter optimization and regularization on fashion-MNIST classification. *International Journal of Recent Technology and Engineering (IJRTE)*, *8*(2), pp.3713-3719.

Henzinger, T.A., Lukina, A. and Schilling, C., 2019. Outside the box: Abstraction-based monitoring of neural networks. *arXiv preprint arXiv:1911.09032*.

Keysers, D., Deselaers, T., Gollan, C. and Ney, H., 2007. Deformation models for image recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *29*(8), pp.1422-1435.

Kickingereder, P., Isensee, F., Tursunova, I., Petersen, J., Neuberger, U., Bonekamp, D., Brugnara, G., Schell, M., Kessler, T., Foltyn, M. and Harting, I., 2019. Automated quantitative tumor response assessment of MRI in neuro-oncology with artificial neural networks: a multicentre, retrospective study. *The Lancet Oncology*, *20*(5), pp.728-740.

Kimura, K., Tabe, Y., Ai, T., Takehara, I., Fukuda, H., Takahashi, H., Naito, T., Komatsu, N., Uchihashi, K. and Ohsaka, A., 2019. A novel automated image analysis system using deep convolutional neural networks can assist to differentiate MDS and AA. *Scientific reports*, *9*(1), pp.1-9.

Maksimenko, V.A., Kurkin, S.A., Pitsik, E.N., Musatov, V.Y., Runnova, A.E., Efremova, T.Y., Hramov, A.E. and Pisarchik, A.N., 2018. Artificial neural network classification of motor-related EEG: An increase in classification accuracy by reducing signal complexity. *Complexity*, *2018*.

Online] https://en.wikipedia.org/wiki/MNIST_database:

Park, W.J. and Park, J.B., 2018. History and application of artificial neural networks in dentistry. *European journal of dentistry*, *12*(04), pp.594-601.

Prajapati, A., Kaushik, A., Gupta, P., Sharma, S. and Jain, S., 2020. Fashion Product Image Classification Using Neural Network. *vol*, *3*, p.3.

Shahid, N., Rappon, T. and Berta, W., 2019. Applications of artificial neural networks in health care organizational decision-making: A scoping review. *PloS one*, *14*(2), p.e0212356.

Shrivastava, V.K., Pradhan, M.K., Minz, S. and Thakur, M.P., 2019. Rice plant disease classification using transfer learning of deep convolution neural network. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, *3*(6), pp.631-635.

Wang, S., Yang, D.M., Rong, R., Zhan, X. and Xiao, G., 2019. Pathology image analysis using segmentation deep learning algorithms. *The American journal of pathology*, *189*(9), pp.1686-1698.

Wang, S., Yang, D.M., Rong, R., Zhan, X., Fujimoto, J., Liu, H., Minna, J., Wistuba, I.I., Xie, Y. and Xiao, G., 2019. Artificial intelligence in lung cancer pathology image analysis. *Cancers*, *11*(11), p.1673.

Wang, S., Zeng, Y., Liu, X., Zhu, E., Yin, J., Xu, C. and Kloft, M., 2019. Effective end-to-end unsupervised outlier detection via inlier priority of discriminative network. *Advances in neural information processing systems*, *32*.

Yang, G.R. and Wang, X.J., 2020. Artificial neural networks for neuroscientists: A primer. *Neuron*, *107*(6), pp.1048-1070.

Zhao, F., Zhang, C., Dong, N., You, Z., and Wu, Z., 2022. A Uniform Framework for Anomaly Detection in Deep Neural Networks. *Neural Processing Letters*, pp.1-22.