

MNIST DIGIT CLASSIFICATION USING MACHINE LEARNING

Project report submitted in partial fulfillment of the requirement for
the degree of Bachelor of Technology

in

Computer Science and Engineering/Information Technology

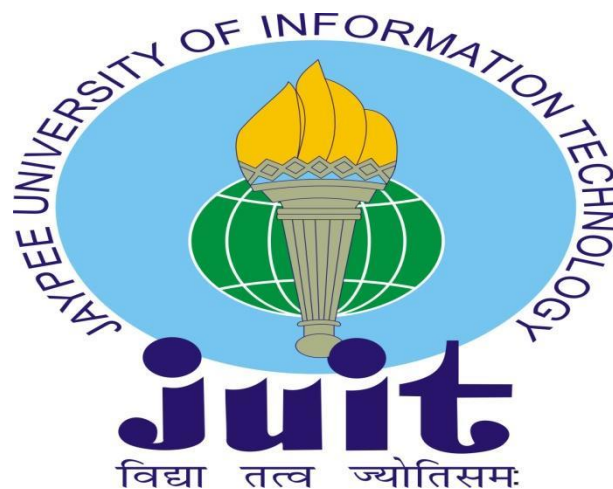
By

Hridyesh Khandelwal, 191312

Under the supervision of

Dr. Pardeep Kumar

to



Department of Computer Science & Engineering and Information
Technology

**Jaypee University of Information Technology Waknaghat, Solan-
173234, Himachal Pradesh**

Candidate's Declaration

I hereby declare that the work presented in this report entitled “**MNIST DIGIT CLASSIFICATION USING MACHINE LEARNING**” in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from July 2022 to May 2023 under the supervision of **Dr. Pardeep Kumar** (Associate Professor, Computer Science Department).

I also authenticate that I have carried out the above-mentioned project work under the proficiency stream: Machine Learning

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

(Student Signature)

Hridyesh Khandelwal, 191312

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

(Supervisor Signature)

Dr. Pardeep Kumar

Associate Professor

Computer Science & Engineering and Information Technology

Dated

Plagiarism Certificate

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: 1.5.23

Type of Document (Tick): PhD Thesis M.Tech Dissertation/ Report B.Tech Project Report Paper

Name: Hridyesh Khandelwal Department: CSE Enrolment No 191312

Contact No. 9636092282 E-mail. hridyeshpk07@gmail.com

Name of the Supervisor: Dr Pardeep Kumar

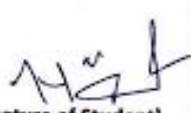
Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): MNIST
DIGIT CLASSIFICATION USING
MACHINE LEARNING

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/ revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages = 56
- Total No. of Preliminary pages = 10
- Total No. of pages accommodate bibliography/references = 2


(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found Similarity Index at 4 (%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.


(Signature of Guide/Supervisor)


Signature of HOD

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	Excluded	Similarity Index (%)	Generated Plagiarism Report Details (Title, Abstract & Chapters)	
	<ul style="list-style-type: none"> • All Preliminary Pages • Bibliography/Images/Quotes • 14 Words String 		Word Counts	
Report Generated on			Character Counts	
		Submission ID	Total Pages Scanned	
			File Size	

Checked by
Name & Signature

Librarian

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com

Acknowledgement

Firstly, I express my heartiest thanks and gratefulness to almighty God for His divine blessing makes it possible to complete the project work successfully.

I am really grateful and wish my profound indebtedness to Supervisor **Dr. Pardeep Kumar, Associate Professor**, Department of CSE Jaypee University of Information Technology (Solan). Deep Knowledge & keen interest of my supervisor in the field of “**Machine Learning**” to carry out this project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stages have made it possible to complete this project.

I would also generously welcome each one of those individuals who have helped me straightforwardly or in a roundabout way in making this project a win. In this unique situation, I might want to thank the various staff individuals, both educating and non-instructing, which have developed their convenient help and facilitated my undertaking.

Finally, I must acknowledge with due respect the constant support and patience of my parents.

Hridyesh Khandelwal

191312

Table of Contents

Title	Page No.
Declaration	I
Certificate	II
Acknowledgement	III
Abstract	IV
Chapter-1 (Introduction)	1
Chapter-2 (Literature Survey)	8
Chapter-3 (System Design and Development)	11
Chapter-4 (Experiments and Result Analysis)	29
Chapter-5 (Conclusions)	32
References	43
Appendices	45

Table of Figures

Figure Name	Page Number
The MNIST Dataset	5
KNN Algorithm	9
Working of KNN	10
The Decision Tree structure	12
Example of Decision Tree	13
Random Forest Algorithm	14
Working of Random Forest	15
Random Forests and Decision Trees	16
A Deep Neural Network	18
A very Dense Neural Network	18
Mathematics of ANN	19

Working of Activation Function	20
The weights and inputs in Neural Network	21
Neural Networks are based on biological Neurons	21
How weights, biases and activation function work	22
A convolutional Neural Network	23
A filter matrix to detect a vertical line	25
A filter matrix to detect features in image of a cat	25
How a kernel convolves	26
A basic skeleton of a neural network having 2 hidden layers	28
Frequency of digits in MNIST	30

Example image representation of an MNIST digit	30
A convolutional Neural Network	31
A CNN having 5 convolutional layers	31
A neural network having weights=w and bias=b and two hidden layers	31
Accuracy of various algorithms	33
Loss vs Accuracy plot	33
MNIST dataset going through a neural network	34
A CNN with 2 convolutional layers, 2 max-pooling layers	37
A digit 7, in MNIST dataset	39
A CNN model extracting features	39

Comparisons of Accuracies and time taken by various activation functions	40
---	-----------

Table of Graphs

Frequency of Digits in MNIST Dataset	37
Accuracy of various Algorithms	39
Loss vs Accuracy Plot	39
Comparisons of Accuracies and time taken by various activation functions	40

Abstract

This project intends to carry out the task of handwritten digit classification. The task will be carried out using various Machine-Learning and Deep-Learning algorithms. The project will be implemented using python.

With the help of Machine-Learning and Deep-Learning algorithms, we will be able to build models which will take an image of a human handwritten digit as input, and will be able to classify those digits into categories (0-9). We will then analyze which algorithms are relatively more accurate and why. We will use the following algorithms: KNN (K-Nearest Neighbor), Decision Trees(DT), Random -Forests(RF), Artificial Neural Networks (ANN) and Convolutional-Neural-Networks(CNN). We will use a dataset called MNIST. In this dataset we will implement the algorithms listed above. After that we will find the accuracy of all methods, and rank them accordingly. Then we will attempt to understand why some algorithms are less accurate than others. After that, we will list out the applications of this project and the future scope of this project. We will carry out a comprehensive comparison of these algorithms, listing their merits and disadvantages.

With the aid of Machine-Learning techniques, image recognition tools and applications are now extensively used and constantly being improved. The MNIST database is a publicly accessible dataset that essentially allows users to assess the effectiveness as well as accuracy of various models to ensure ongoing development through international competition. The MNIST dataset contains 70000 manually written digits. Since handwritten digit classification has many applications in the market, the demand for efficient algorithms is constantly increasing.

Chapter 1

Introduction

1.1 Introduction:

Human-written-number synthesis is the capability of computers to describe human-written numeric-digits. Since it is rarely correct to employ human-written numbers, and they also might have a spectrum of types, our job becomes tough. The workaround to this issue lies in the method/solution which we are going to describe in this chapter.

Handwritten numbers are imperfect, varied from person to person, making it difficult for the computer to complete the task. Therefore, handwritten text recognition is an important field of study and development with a wide range of potential outcomes.

In the realm of DL, this has been the subject of countless studies. Numerous uses for digit recognition include driverless cars, identifying number plates, banking, etc. DL, ML, and artificial intelligence have all benefited from a significant amount of research and development effort carried out in the last few years. The processing power of machines is increasing with time, and this has improved the quality of human lives. DL and ML have crucial applications in handwritten digit identification, and a lot of research has already been done in this area.

In this report, we intend to explain how, using ML and DL algorithms, we can classify handwritten digits to a reasonable accuracy. We used many different ML and DL algorithms, and we will list the advantages and disadvantages of each.

We will then list out the reasons why some algorithms perform better than others at the task of image classification. The language we used for implementation of this task is python. We used different libraries and frameworks, about which we will elaborate in detail in later sections.

The dataset we used was the MNIST data-base. It is a very, very popular tool for working out ML and DL algorithms.

In practically every scientific subject today, image classification is becoming a crucial component. Handwriting recognition is crucial to information processing in the contemporary era of digitalization. Paper is a great source of knowledge, and digital data is processed far less than traditional paper files. The handwriting recognition system's goal is to transform handwritten letters into forms that computers can understand.

The purpose of our study is to develop a model that can identify and categorize human-written numbers from photographs using KNN, CNN, ANN, etc. Our study aims to develop a model for digit identification and classification, but it may be used for letters and a person's handwriting as well. The main objective of the suggested approach is to comprehend how computers classify numbers and then use our understanding to create a model that does the same.

Character recognition in handwriting has existed for some time, from the 1980. The endeavor of recognising handwritten digits, employing a classifier, which is quite important, and utilizing recognising postcodes with online character classification on computers numerical processing of bank check amounts, postal codes, portions of buildings filled out by hand (such as tax) and so forth. Various difficulties are encountered while trying to find a solution to this issue. The letters and numbers are the very same size, width, or alignment every time, and with respect to the margins. The primary goal was to actualize a pattern recognition technique to understand the photos from the MINIST data collection contain handwritten numbers comprising 0–9 handwritten digits.

As a test case for theories of image classification algorithms, handwritten number identification is a significant issue in optical character recognition. Several common databases have arisen to support the study of pattern recognition and ML. Due to the fact that everyone in the world has a distinctive writing style, handwriting identification is among the most captivating and exciting fields of study. The primary challenge in reading handwritten numerals is the significant variation in height, motion, line width, spin, and distortion of the numeral picture since different people write handwritten digits in various ways.

Building a valuable training set is a crucial factor in ensuring an excellent productivity in the learning process.

Although the 70000 unique patterns in the MNIST database may seem like a sizable collection, research suggests that the typical learning algorithms fail miserably for at least 100 test set samples. Therefore, a technique is required to improve the cardinality and variety of the training set. Common actions include geometric distortions like displacements, rotations, scaling, and others.

The methods we wrote in this report can be used/applied in a diverse spectrum of real-life issues: form data entry, banking account handling, post-mail sorting, and others.

1.2 Problem Statement:

Build a model which shall take an image of a human handwritten digit as input, and will classify the digit into categories, 0 to 9.

Various strategies have been developed by researchers and programmers working in image processing and classification to address the issues with handwritten number recognition. Our system's primary goal is to identify solitary Arabic digits, which may be found in many diverse applications.

The key problem here is getting the computer to comprehend these various handwriting styles and recognise them as conventional writing because different users had varying handwriting habits. Our system successfully creates and implements a neural network that operates well without input, and after that, it is able to comprehend Arabic numerals that have been manually entered by users.

Image classification and ML now face additional difficulties as a result of the explosive proliferation of newspapers and audiovisual information. In the subject of pattern categorization, the issue of human-handwritten number classification has long remained unsolved. Because everyone in the world has a unique writing style, handwriting

identification is among the fascinating scientific projects now being conducted. It is the capacity of a computer to automatically recognise and comprehend handwritten numbers or letters.

1.3 Objectives:

Our main objective is to build a model that will take an image of a human handwritten digit as input, and classify it into one of the ten categories, 0 to 9.

The model should be able to recognize digits to a reasonable degree of accuracy. Further, the model should not be too memory intensive or should not have too high time complexity.

This study uses ML technologies to train these classification models in order to detect handwritten numbers with a high level of accuracy. A lot of people utilize the MNIST data collection for this recognition procedure. There are 70,000 handwritten digits in the MNIST data collection. This data set's images are each represented as a 28x28 array.

1.4 Methodology:

The primary methods we used in this project are the popular ML and DL algorithms. We used ML algorithms like Support Vector Machines, DTs, R-Fs, KNN (K Nearest Neighbour), and DL algorithms like ANN and CNN.

The implementation is done in python, using the libraries sklearn and tensorflow. These are highly popular libraries used by researchers and programmers who do work in ML and DL.

The raw input will be in the form of an array, where each number will represent the pixel value. The array size is 784, which we can then convert to 28 x 28.

MNIST is a dataset of 70,000 human handwritten digits, and each of these digits is represented in the form of an array. Each digit is between 0 and 9.

The steps in the methodology are as follows;

1. Loading the data
2. Normalizing, reshaping the data
3. Building the model
4. Training the model
5. Predicting on test dataset
6. Evaluating the model (performance metrics)
7. Fine tuning the model (removing overfitting and underfitting)

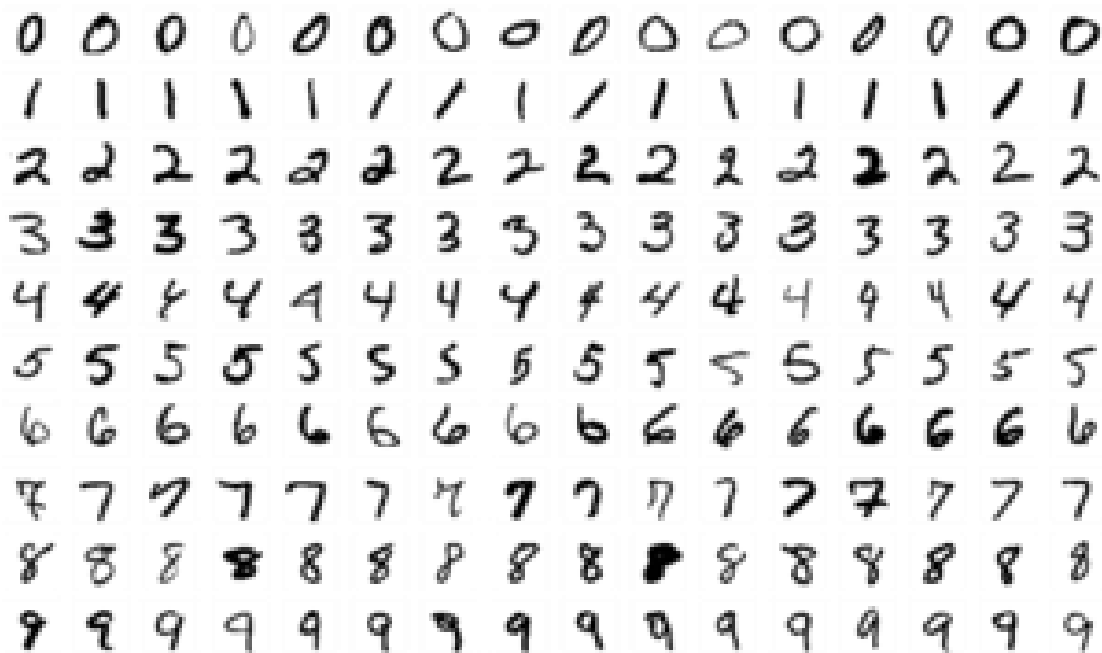


Figure 1: The MNIST dataset

In this work, extraction of features and categorization are the main topics. A classifier's success may be as dependent on the features' quality as it relates to the classifier. A smaller portion of the NIST collection is called the MNIST dataset. 70,000 handwritten digits, broken down into 60,000, in a database. 10,000 test samples and 10,000 training examples. The

photos in The MNIST dataset is presented as an array that includes a set of 28x28 values corresponding to a picture and its labels.

1.5 Organization:

This project is divided into mainly 4 chapters. They are described as follows:

Chapter 1: It consists of an introduction to our project, the problem statement of the report, our primary objectives and methodology being carried out.

Chapter 2: It consists of a literature survey, in which we described the papers we read, which supplied us with the necessary knowledge about DTs, R-Fs, KNN, ANNs and CNNs.

Chapter 3: This chapter consists of our detailed implementation of the algorithms, their merits and demerits, questions like why are some algorithms more accurate than others. In this chapter, we described the working of various algorithms, how they are implemented and where they are needed. The chapter consists of detailed text about KNN, DTs, R-Fs, ANNs and CNNs.

Chapter 4: This chapter is about the performance analysis of the algorithms we used in the project. We analyze and compare the performance and accuracy of the algorithms. We also answer why some algorithms have a greater precision than others. In this chapter, we will see that CNN generally gives relatively more accurate outcomes than other algorithms, and we will also find out why.

Chapter 5: This section is about conclusions and future scope. Here we will describe the conclusions we reached regarding each algorithm, and also explain why we got a certain outcome. We will also describe the potential future scope of this project.

Chapter-2

Literature Survey

LeCun(1998) released one of the earliest lists/rankings of algorithms ordered according to their accuracy on MNIST, which was valid up until 2012. A recent work was published by Shivam et al, 2020, where they discussed performance of Convolutional-Neural-Networks on MNIST. They reported accuracy more than 99%. Grover & Roy, 2009, described a modified KNN algorithm on MNIST. Alejandro et al, 2019 described in paper latest progress in the field of digit classification on MNIST. Alexis et al, 2019 analyzed in detail the effectiveness of DTs and MNIST. Kevin et al, 2017 described how to build deep R-Fs on MNIST. They proposed a general framework which they called forward-thinking for DL. One layer is trained at a time to do this, and once each layer has been trained, the inputs are mapped through the layer to produce a new learning-problem. The procedure is then carried out once more, processing the data via many layers one at a time, creating a fresh dataset that is anticipated to behave better and on which the final output layer may work well. Grover described KNN using a modified sliding window metric. We use a distance-metric that makes use of the sliding_window methodology in order to prevent performance degradation caused by minor spatial misalignments when evaluating the output of the KNN classifier onto MNIST dataset. Results demonstrate a considerable improvement when utilizing the suggested technique when compared to the baseline algorithm when using the correctness metric and confusion matrix as performance metrics. Elijah et al, 2019 produced a comprehensive comparison of algorithms for classification of online and offline handwritten digits. They analyzed DTs, ANNs, instance-based Learners, Naive Bayes. They found the highest accuracy in instance-based Learners, followed by ANNs, Naive Bayes and DTs. Tsehay, 2019, described a novel approach to human-handwritten digit classification using a novel ML approach. Deepak et al, 2012 described Advanced Approaches of Handwritten Digit Classification Using Hybrid Algorithms. They found the highest accuracy of 99.98%, and lowest 81.23%. Firas et al, 2018 described how to build an Optimized System for Training Deep DTs at Scale.

Chapter-3

System Design and Development

We use the following algorithms for the classification of human handwritten digits:

1. KNN (K nearest neighbor)
2. Decision Trees
3. Random Forests
4. ANN (Artificial Neural Networks)
5. CNN (Convolutional Neural Networks)

KNN (K Nearest Neighbor):

A supervised ML (ML) technique known as KNN may be applied to classification/regression problems, although it is mostly employed in industry for categorization and forecasting issues.

KNN is called a lazy-learning-algorithm as it forbids the training phase.

A majority vote is used to apply a class label to classification issues, meaning the label/output.result that is majorly found in proximity to a certain data-point is utilized.

The data-points that are closest/nearest of new-data points are referred to as proximity-neighbors in this context.

K is the amount of proximity-neighbors. We have to decide/choose its value before we start computation.

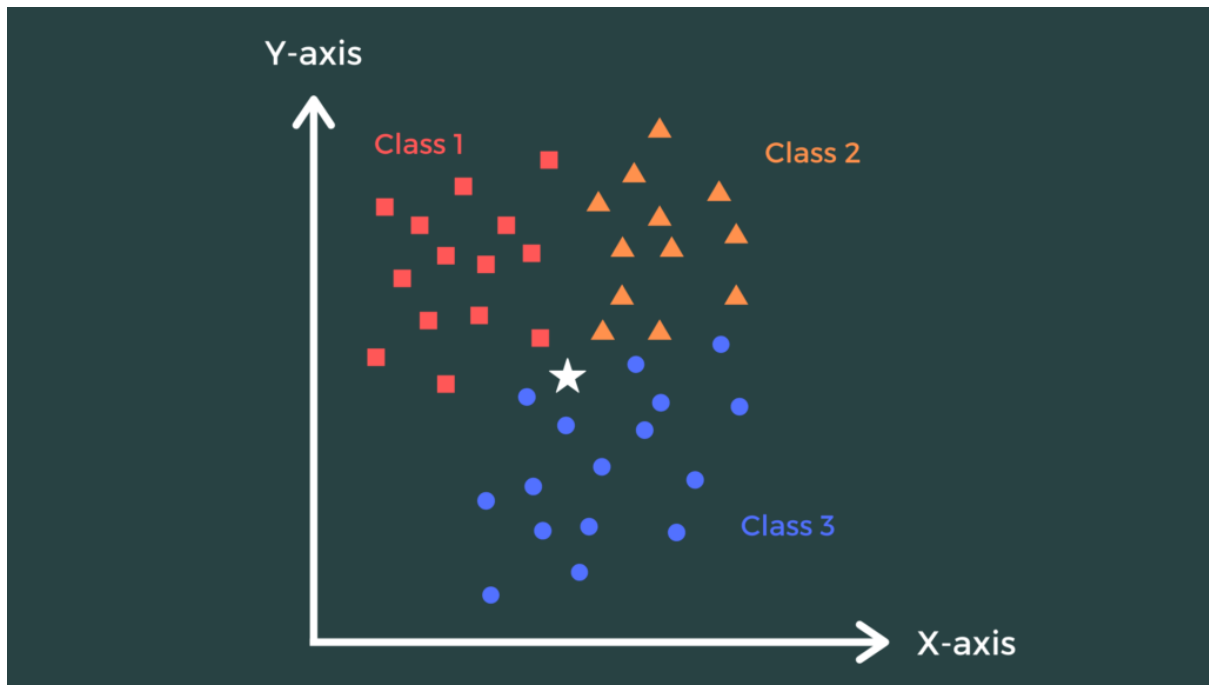


Figure 2: KNN algorithm

The number of labeled points (neighbors) taken into account for classification is indicated by the parameter k in KNN.

Another non-linear approach is the k -nearest neighbors method. It will perform effectively with data when the connection between the independent_variable and the dependent_variable is not a straight_line, in contrast to simpler models like linear regression.

We can attempt to determine which points in our feature space are closest to a point whose class we don't know. The k -nearest neighbors are these points.

The algorithm works as follows:

First, choose the value of k .

Then, compute the distance(euclidean or otherwise) of the test data to all the data-points in the set.

Then, sort points in increasing order of distance.

Then, select top k points.

Then, do majority voting in those k points. The winner category is the category of our test sample.

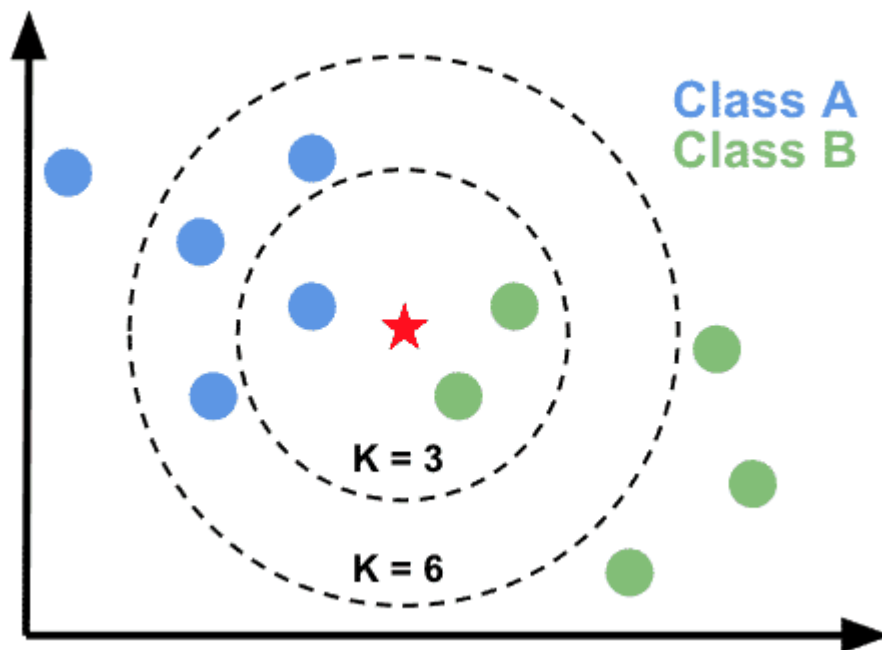


Figure 3: Working of KNN

It is very-likely-possible the data_point is in group 'A' if most of the pieces of data are in category A, and vice versa. KNN, sometimes referred to as the closest neighbor. K-nearest neighbors classification is straightforward to comprehend and use, in contrast to classification by ANN. When the measured values are well-defined or nonlinear, it works well.

KNN actually uses a voting method. It says that for any point on that plane, its category/label/class will be that one which belongs to most of its proximity-points/neighbors.

A crucial issue in AI is classification/identification. A methodology for categorization and regression/classification models is the KNN. A good K value, or the quantity of neighbors in KNN, cannot be precisely determined. This implies that before determining which value to use moving forward, you might just have to explore with a few different values.

Assuming that even a portion of the training instances is "unknown" is one technique to do this. Then, you can use the k-nearest neighbors technique to classify the unseen data in the

testing set, and you can assess how accurate the new classification is by contrasting it with the knowledge you already have from the training data.

It is preferable to pick an odd number for K when solving a two-class issue. Otherwise, a situation might occur where there are the same number of neighbors in each class. Additionally, K's value cannot be a multiple of a number of classes that are present. However, K shouldn't be set too high. On the other hand, K with greater values will often result in smoother decision boundaries. Otherwise, groups with fewer data points would continually lose out to groups with more data points. A greater K will also require more processing power.

DTs(Decision–Trees):

The DT is a pretty good methodology/algorithm, nodes showing a testing on an attribute/feature/property, so the final node is a class-label. A nonparametric learning technique for regression and classification is a DT. The target is to build a model that can predict the values of a target variable.

Like KNN, the DT is also a supervised ML algorithm. It is capable of being used in both classification and regression problems, but is most commonly used for classification.

Leaves and decision nodes are the two types of nodes in DTs.

DTs classify data based on their properties. Splitting is done at nodes according to the property, and the leaf nodes represent a label or output class.

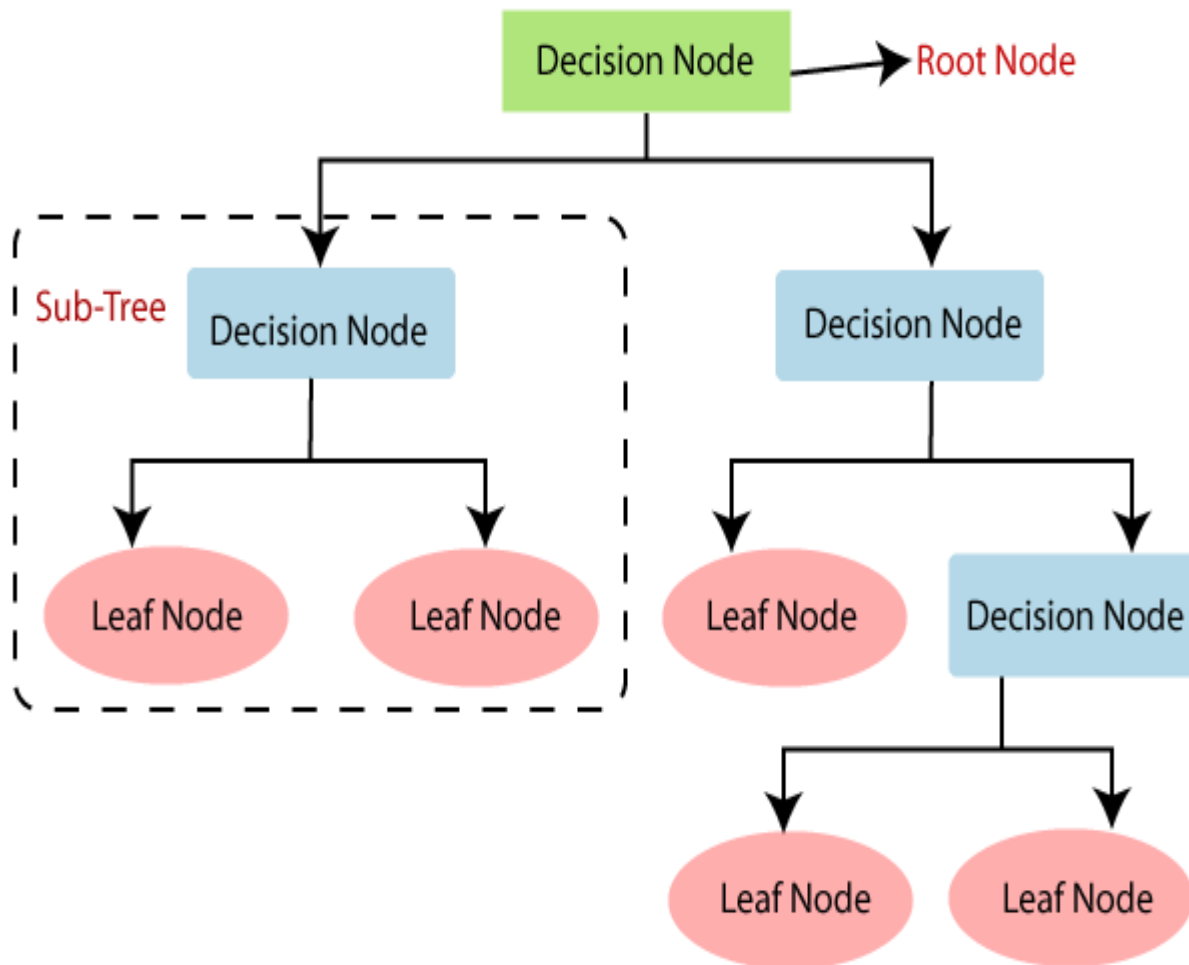


Figure 4: The decision tree structure

Working of DT:

The process starts at the root node, and matches the attributes/properties/features of that node to that of the actual dataset point. On the basis of this matching, jump to the next node.

The algorithm examines the attributes/properties/features with the other sub-nodes yet again for the subsequent node before continuing. The method is carried out until the tree's leaf destination is reached. This is basically how a DT algorithm works in ML.

The essential issue that surfaces while formulating a DT is how to choose the optimum attribute for the root of the tree and for subsequent-nodes. Therefore, a technique named as attribute selection measure, or ASM, can be used to address these issues. By applying this

assessment, we can choose the appropriate characteristic for the tree nodes with ease. Information gain and ginni index are two of the most used ASM methodologies.

It is pretty challenging to maintain the correctness/accuracy of the tree when it is too big. This is so because large trees frequently experience overfitting. Small trees are therefore typically selected. Additionally, we may create a single model by combining numerous DTs; this system is termed as R-F, and it typically resolves the overfitting condition.

Making choices on which characteristics to employ, what circumstances used for dividing, and when to quit while growing a tree. As a tree often expands at random.

If some classes predominate, DT learners will produce partial trees. As a result, it is advised to equalize the database before fitting it to the DT. The model is hypersensitive to even mini scale variations in input-data, it can create a very changed Decision Tree, the Decision Tree algorithm is non-stable.

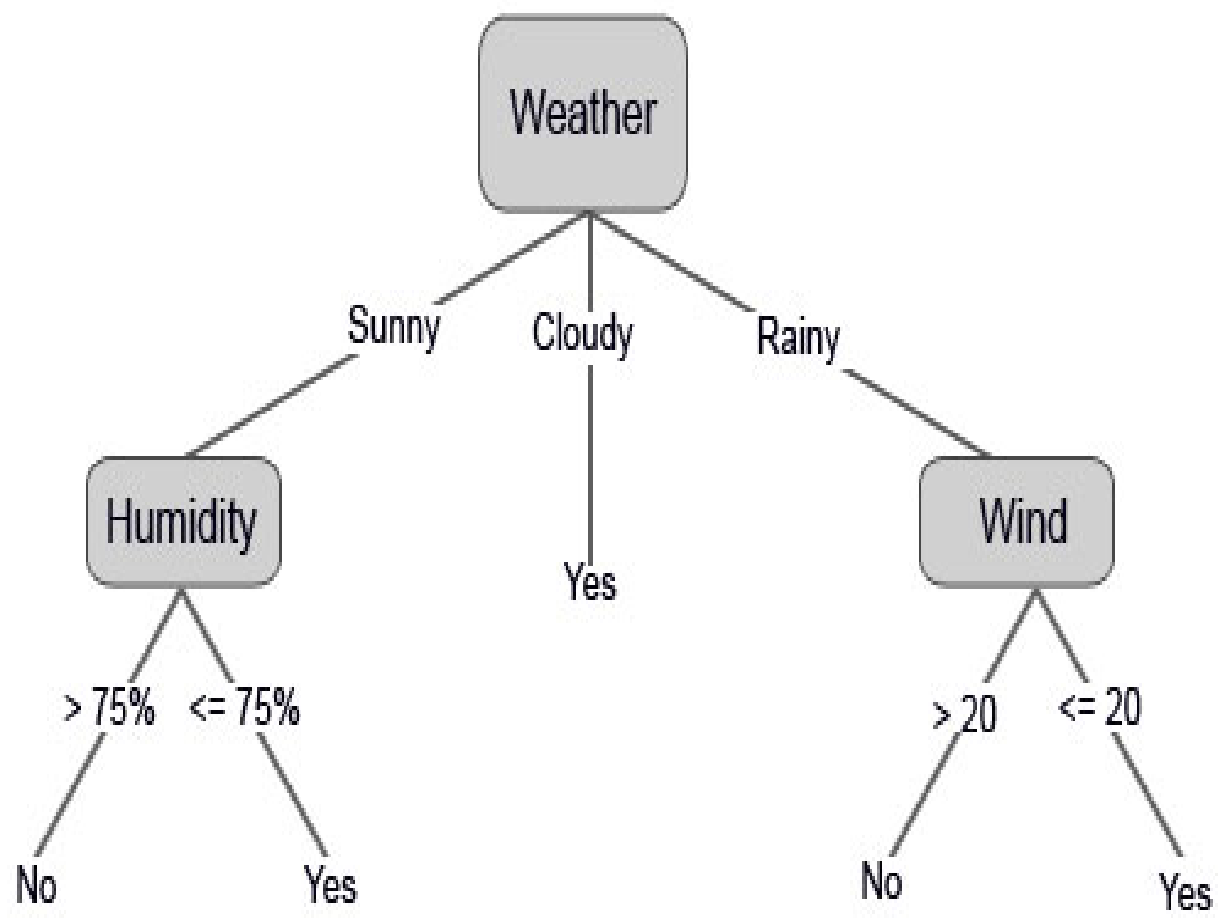


Figure 5: An example of decision tree

R-Fs (Random Forests):

A classification system called R-F is created when several DTs are integrated in a singular model.

The merit lies in the versatility and usefulness as it is powerful enough to compute both identification and regression problems. .

The DTs that make up the ensemble of the R-F algorithm each include a data sample taken from a training set.

R-Fs have the benefit of having the ability to lessen overfitting and offer/provide flexibility.

The fact that it takes more time, uses more resources, and is more complicated is a drawback.

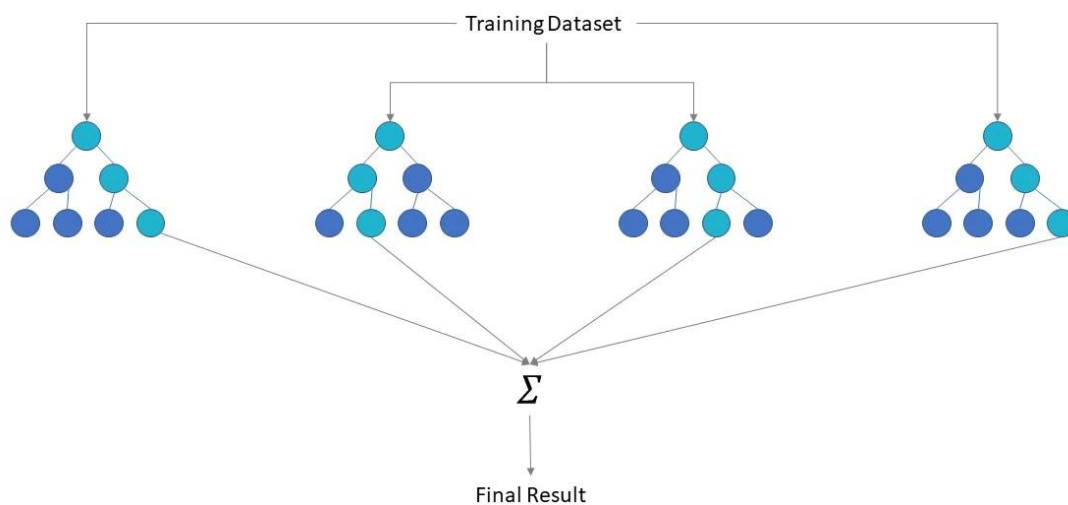


Figure 6: The Random Forest algorithm

The R-F model's fundamental/basic concept is that several non-correlated ML-models produce relatively more precise outcomes together than otherwise. Every tree performs voting activity when we use R-F. While undergoing regression-tasks with R-F, the forest selects the arithmetic-mean of all the tree outcomes.

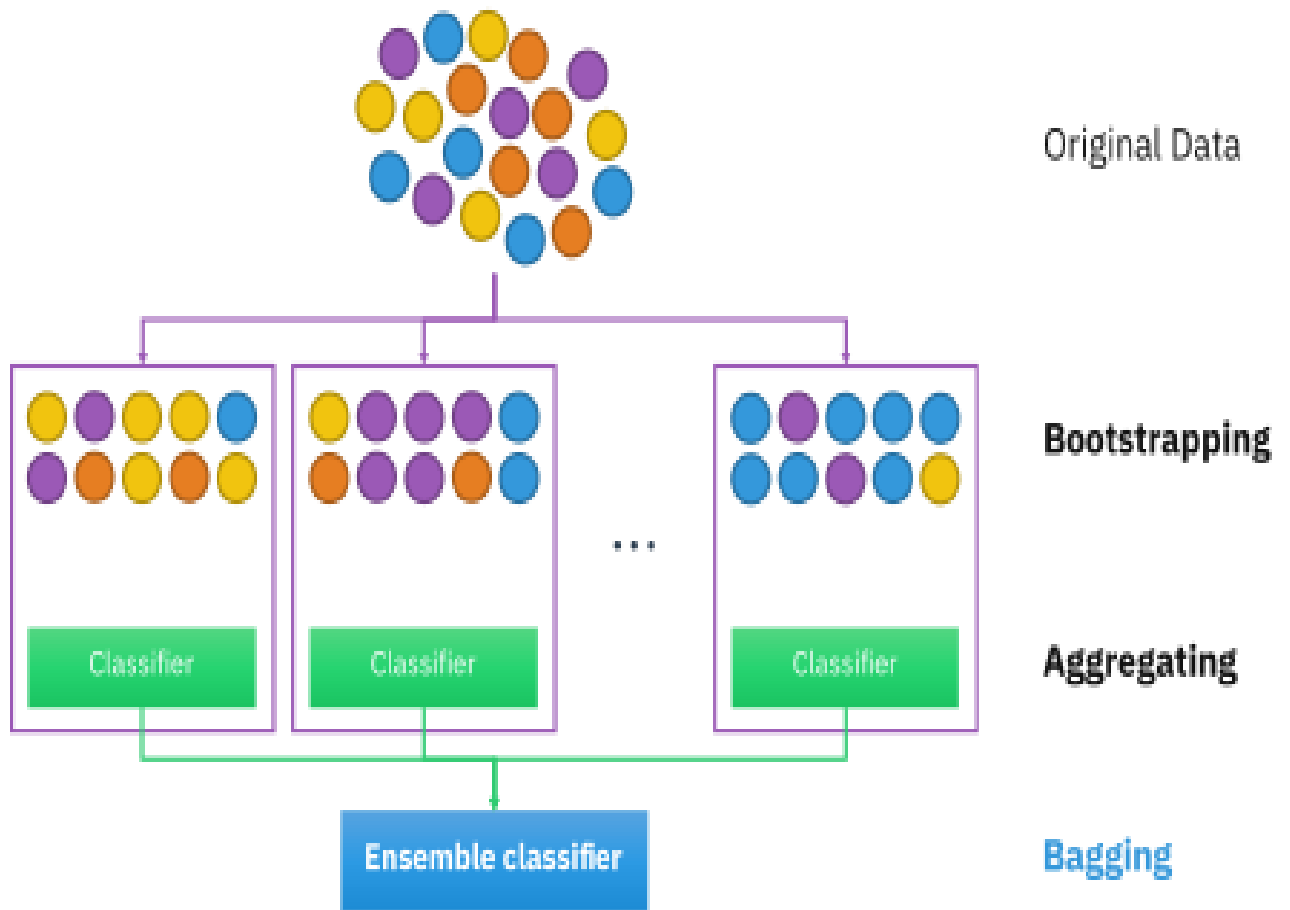


Figure 7: Working of Random Forest

A conventional DT will create a set of rules that it will then use to make forecasts when given a training dataset containing features/attributes and labels. If you supplied a R-F algorithm with the same input, it would randomly choose observations and characteristics to create several DTs, then it would take the mean of the outcomes.

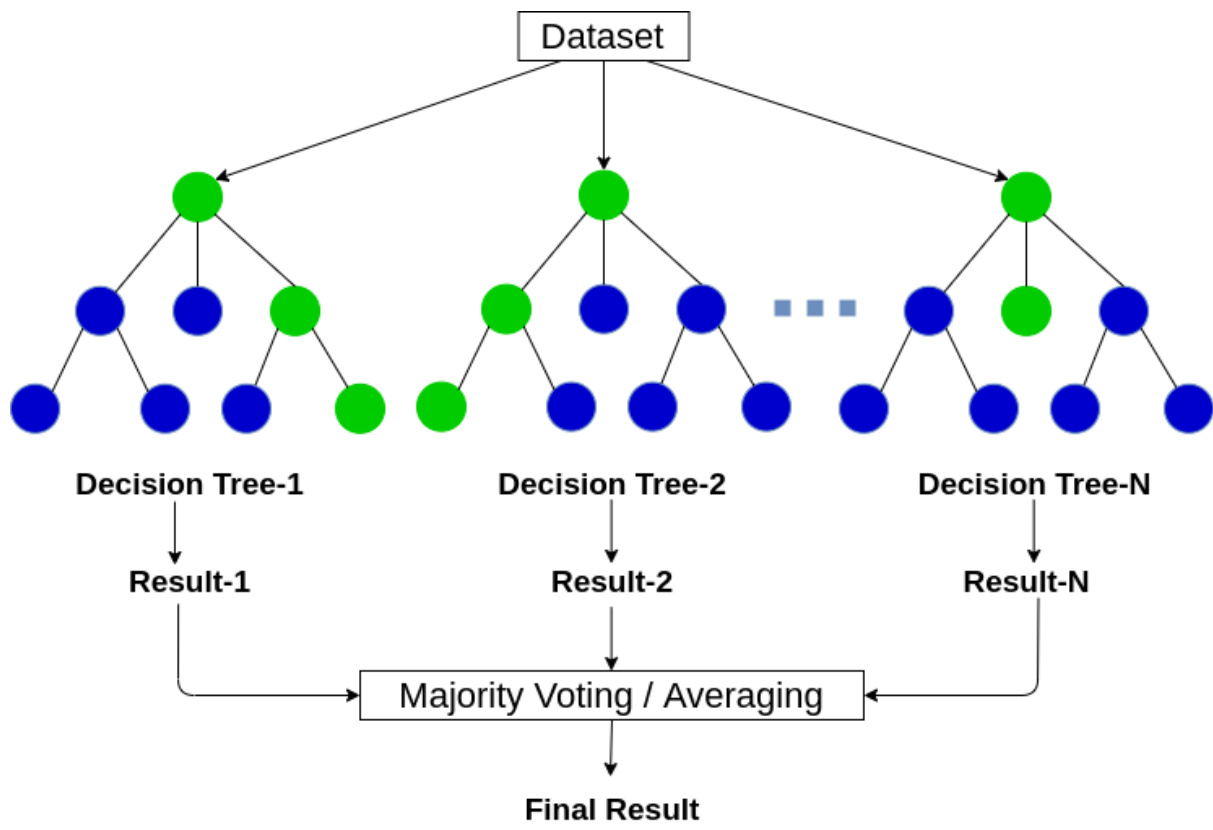


Figure 8: How Random Forest combines several decision trees.

When trees are spreading, the R-F delivers some amount of randomness. When differentiating nodes/vertices, it looks for the best feature from a set where attributes are arranged with a certain degree of randomness, rather than simply having the most significant one.

A superior model is therefore an outcome here, of the huge variety this causes. If some classes predominate, DT learners will produce biased trees.

As a result, it is advised to equalize the data before fitting it to the DT. Since the smallest of variations in input may create a wholly another Decision Tree, the Decision Tree algorithm can be non-stable.

If trees are not adequately restricted and regularized during the developing stage, they run the significant danger of overfitting the training examples and becoming computationally complicated.

This overfitting suggests an exchange in the system between low bias and large variance. Therefore, to address this issue, we employ ensemble learning, a strategy that enables us to break our overlearning propensity and, presumably, produce better, more robust outcomes. In order to get superior outcomes, an ensemble technique or ensembles classifier combines various outputs produced by a variety of predictors. Formally, we are attempting to utilize a "strong" learning for our model based on a group of "weak" learners. Therefore, the goal of employing ensemble techniques is to decrease overfitting of our training set by averaging out the results of separate forecasts by broadening the number of predictors, thereby minimizing the variance.

Outcome: the method for dividing nodes in R-F takes into account a set with unpredictability. Using minimum levels for each attribute, on top of the better possible current minimums, you may even enhance the unpredictability of Decision-trees.

ANN:

ANN are a fundamental part of DL. DL techniques have their basis in the N-N.

They are modeled/created based on the homosapien-brain. In the sense that they continuously update their own weights and biases as more and more data is supplied to them, neural networks are adaptable; they keep evolving.

They gain knowledge through experience in this way. The accuracy we obtain from ANNs is typically greater than that of ML methods.

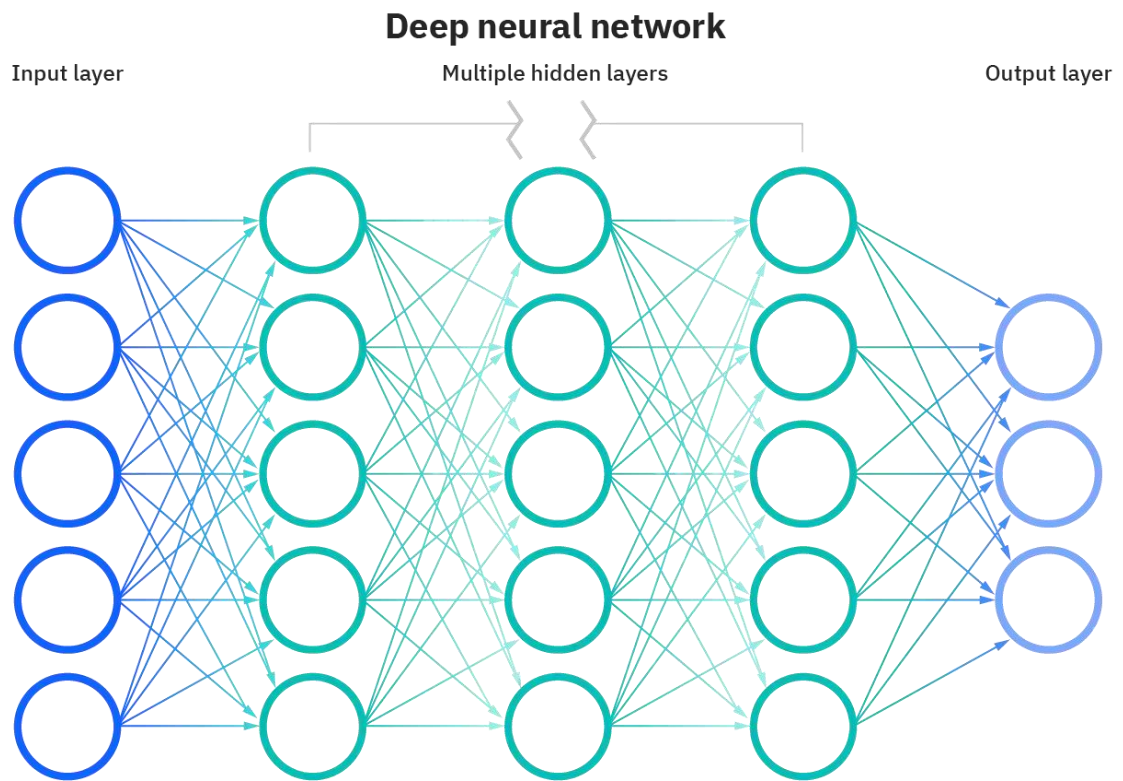


Figure 9: A deep neural network

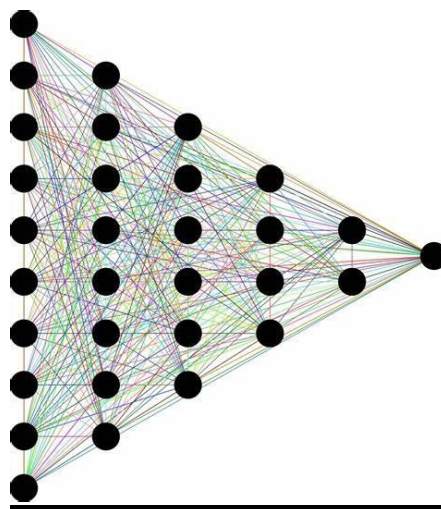


Figure 10: A very dense neural network

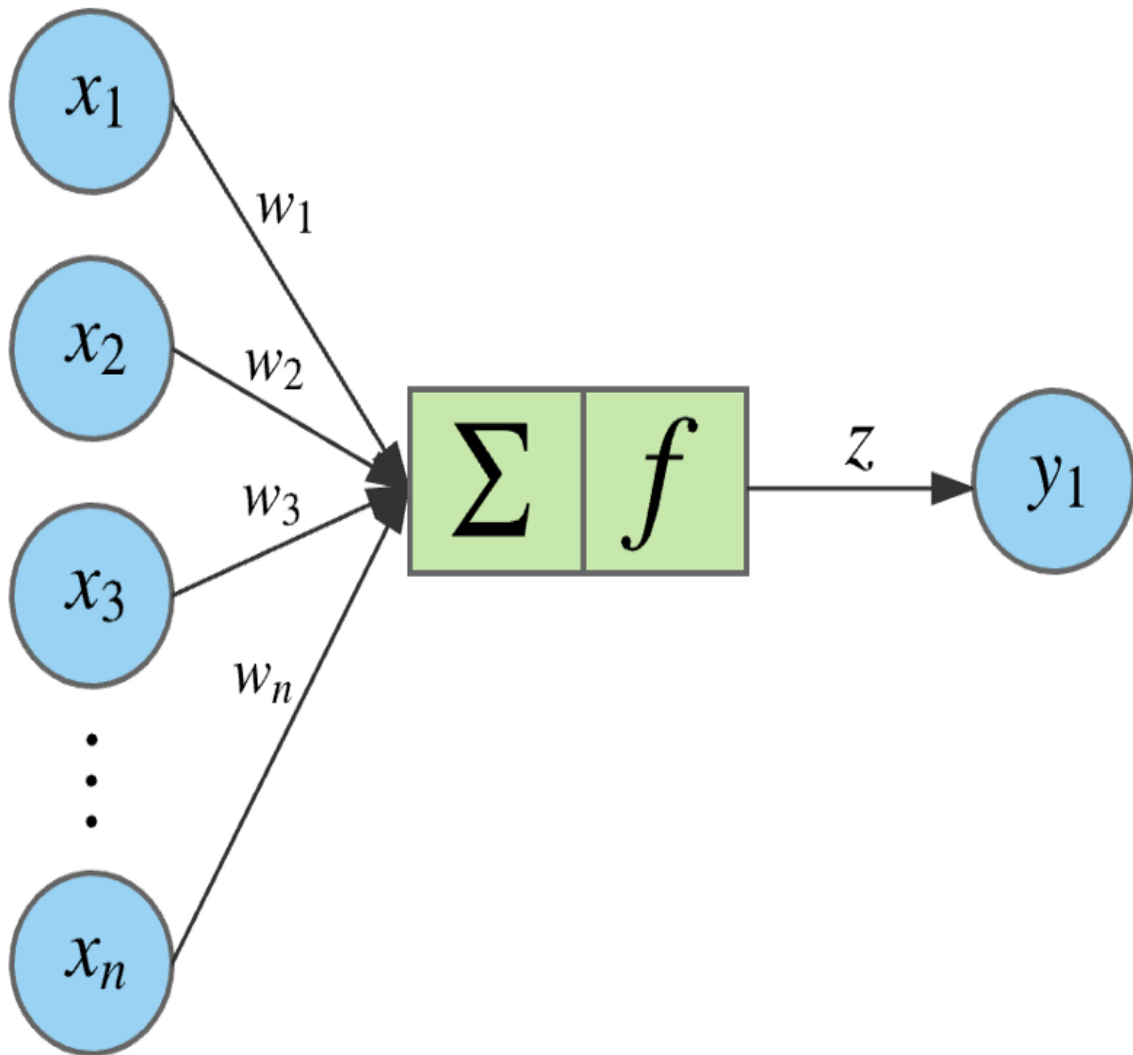


Figure 11: The mathematics of artificial neural networks

Here is an example of a neural network, which consists of certain weights and biases. activation functions are required because they add antilinearity to the network, and a N-N without activation functions would be effectively the same as a linear classifier.

N-N and generally, in most cases, more precise/accurate than the ML methods. That reactive precision can be attributed to a better computational capacity, more sophisticated architecture and a large diversity of choices regarding activation functions, types of layers, etc.

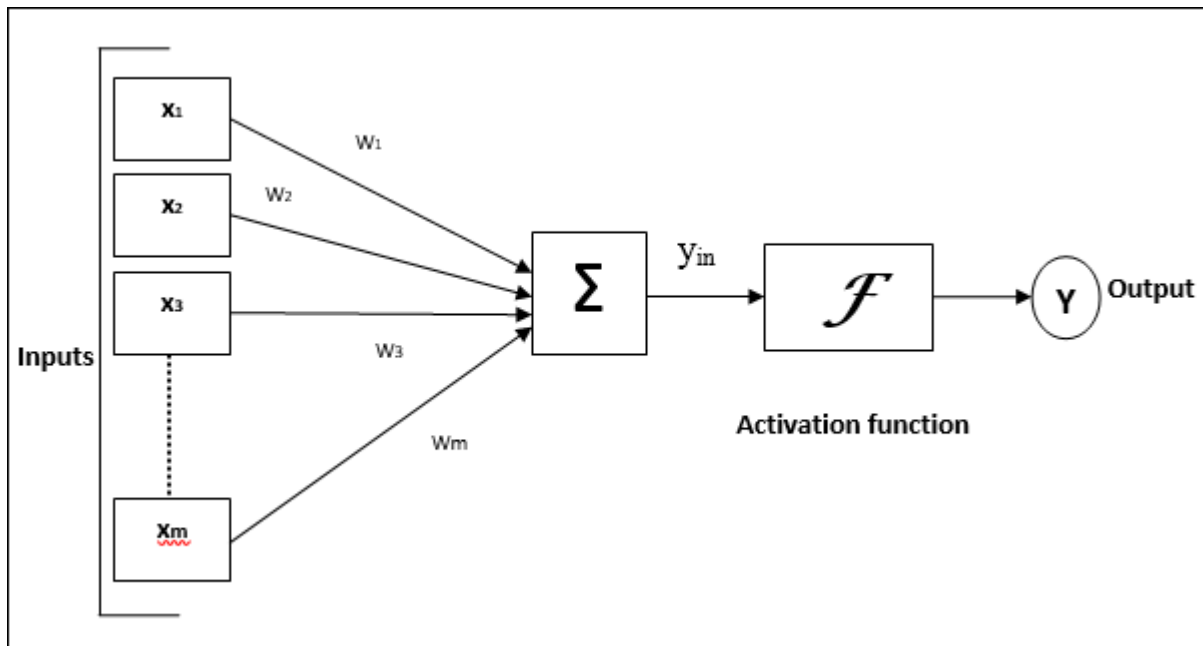


Figure 12: Working of activation functions

Y is cumulative input, F is the activation function.

In order to guarantee accuracy of fit for each data-sample, our ultimate aim is to minimize our cost-function. In order to get to the optimal point, also known as the local-minimal-point.

To update weights/biases, a methodology called gradient_descent(GD) is used, enabling the models to choose optimal activation to minimize faults (or minimize the cost-function).

The model's arguments change with each training sample and eventually reach the minimum/lowest point. In order to make up for the discrepancy between the anticipated outcome and the observed one, the error is utilized to optimize the weights of our ANN's connectibles. This is among the most well-known benefits of an ANN: it can really learn from viewing datasets, or, in other words, learn from experience. An ANN offers a number of other benefits as well.

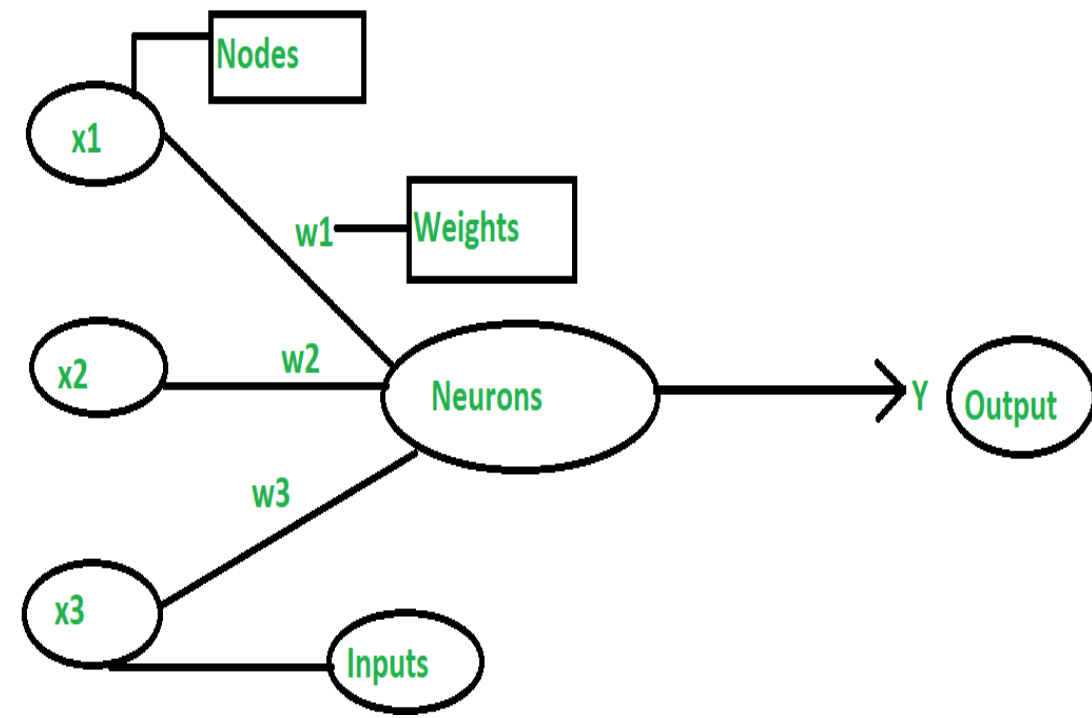


Figure 13: The weights and inputs in the neural network

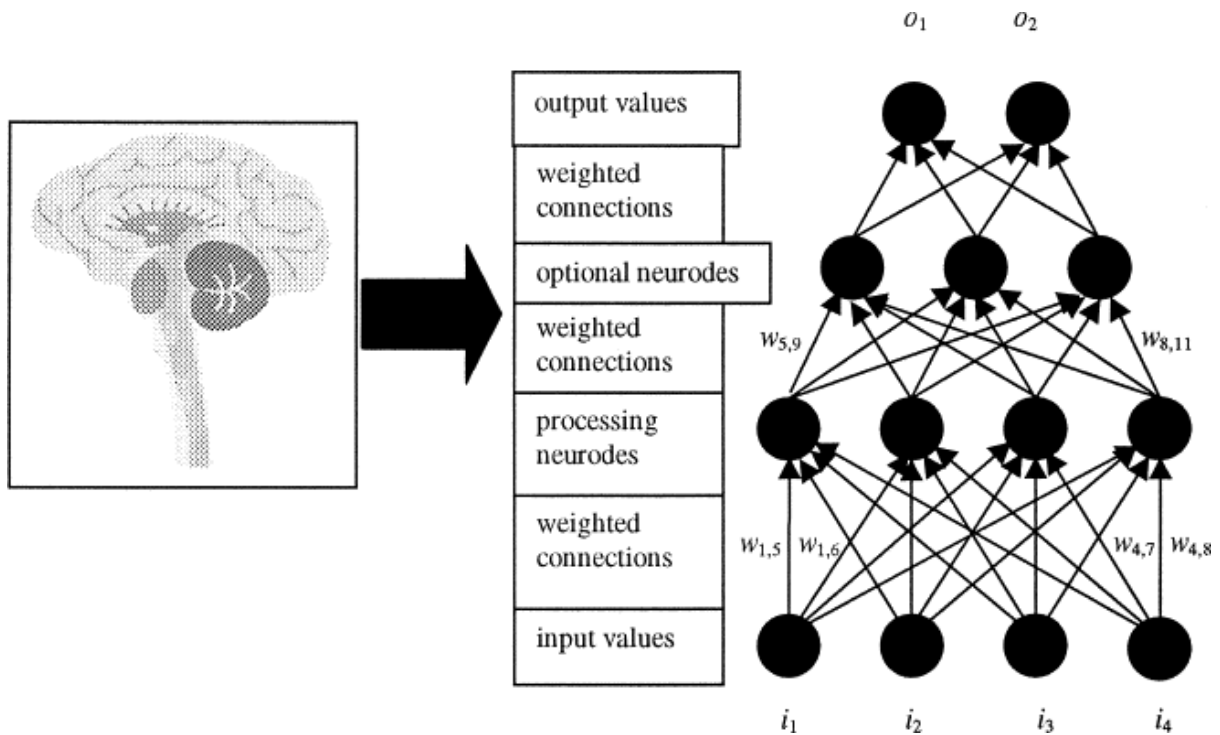


Figure 14: Neural networks are based on biological neurons

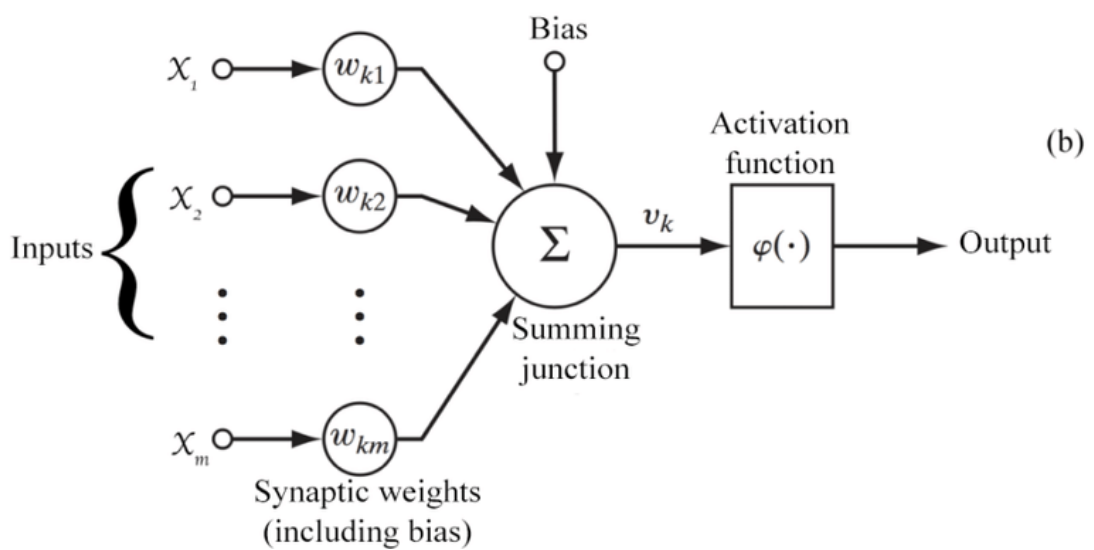
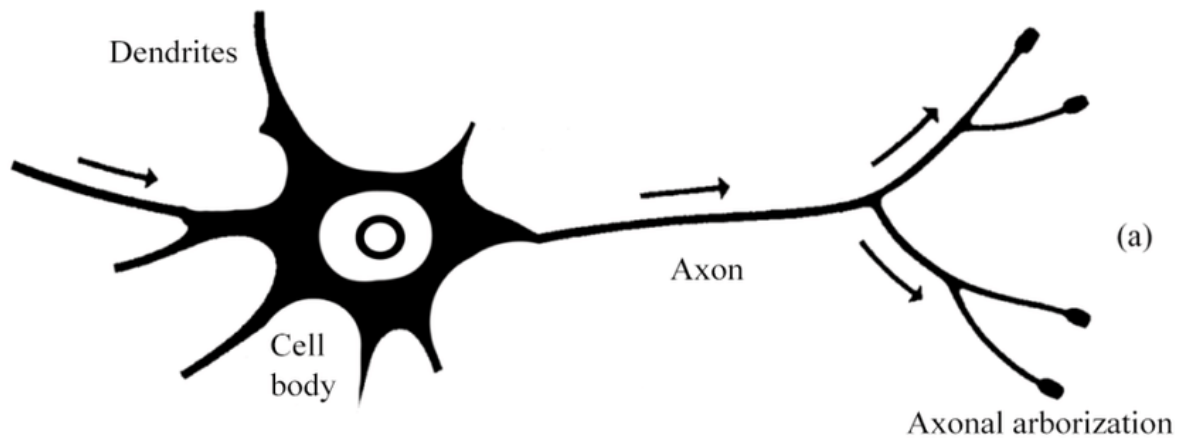


Figure 15: How weights, biases and activation function work

Biological/organic neurons consist of dendrites, cell body and axon.

ANNs are composed of input layers, output layers, and/or single/multiple hidden-layers. It also consists of an activation function, which decides whether a neuron should be activated or not, weights as well as biases. Finally, output is produced.

CNN:

CNN outperforms other neural networks when applied to image processing. Convolutional and pooling layers make up CNNs.

The filter size is normally a three x three vector, though they can come in different sizes. The output array is therefore given with this dot product. The filter moves and performs a repetition of the operation. A map_feature comes after a sequence of dot-products from the inputs and the filter. It is not necessary for every resulting value in the feature-map to correspond to every pixel-value in the input-image. Just the area, where the filtering is being used, needs to be connected.

Finding weights in fully - connected and kernels in convolutional layers that minimize disparities among output predictions and providing actual outputs.

Computer vision, image processing, audio processing, and medical imaging are just a few examples of applications for CNNs.

Although DL has made astounding recent strides, there are also obstacles preventing its use in medical-imaging. Some of the problems that arise are encountered when dealing with imbalanced datasets, leading to overfitting (high variance) or underfitting (high bias).

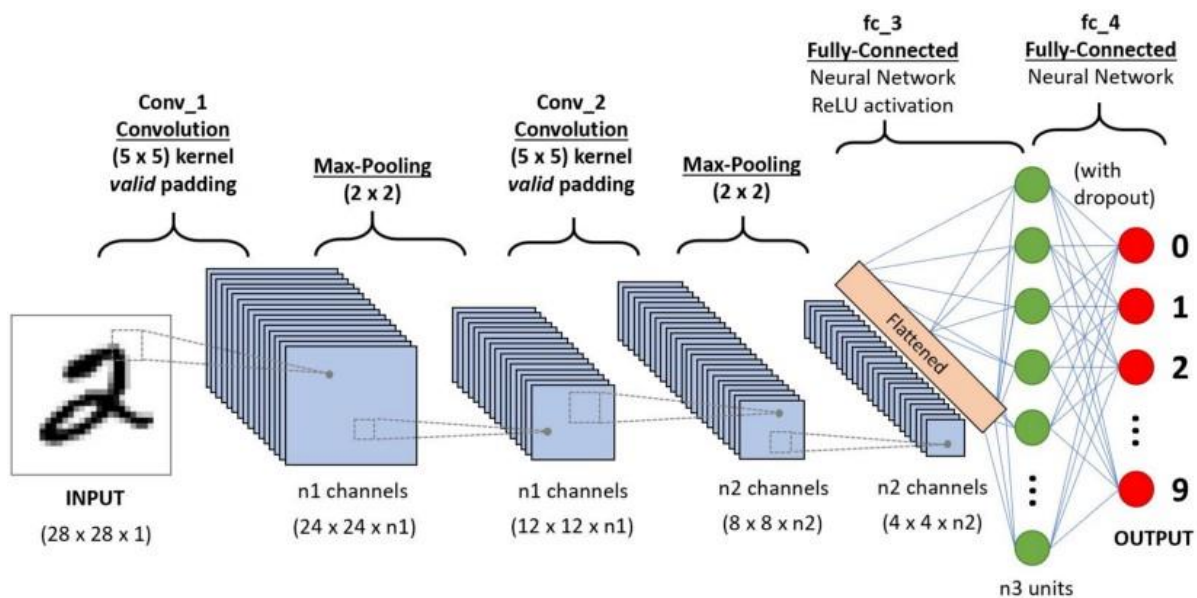


Figure 16: A Convolutional Neural Network

The topology of a ConvNet being similar to the connection network of-neurons found in the homosapien brains and it was also modeled after the visual-cortex is organized. Only in the Receptive-Field do individual neurons perform reaction to inputs. Similar fields being taken together encompass the whole field.

Multiple convolutional-layers being put on top of one-another in CNN, and every single layer is capable of identifying increasingly complex structures.

A CNN method makes use of convolutional-layers to analyze input-images and recognise increasingly harder or relatively complicated elements.

A CNN topology is a multiple-layered network created by sequentially putting several hidden-layers before each one. CNN can learn other complicated/harder features because of this particular type of orderly architecture.

Convolutional-layers are frequently preceded by activator-layers, some of which are then followed/preceded by pooling-layers, as the hidden-layers.

The core idea in CNNs is the process of convolution. We use a filter matrix, which spreads on the training image and dot product is carried out. Finally, all the outputs of dot products are added up, and this basically results in feature-extraction.

Despite being straightforward, the convolution kernel's capacity to recognise straight or curved lines, corners, and other basic characteristics is a very potent one. As you may remember, a convolutional-layer is composed of several convolution kernels. A convolutional neural network's first layer typically includes multiple diagonal, curve, and corner detectors. The neural network learns such feature-detector kernels during training rather than a human programming them, and they are used as the initial step in the image identification process.

A fundamental CNN may be thought of as a collection of convolutional-layers, a pooling (downsampling) layers, and so forth. The first layer recognises basic elements like edges in an image with the repetitive combining of these processes, while the second layer starts to identify higher-level features. A CNN can recognise increasingly intricate forms, like eyeballs, by the eleventh layer. It can frequently recognise one human face from another by

the twentieth layer. This strength results from the repetitive stacking of processes, each of which is capable of detecting a little bit more complex characteristics than the one before it. It is obvious that a CNN employs a significantly less number of parameters than a densely integrated feed - forward neural-network of comparable size and layer dimensions. This occurs when the convolution kernel moves over the picture and the parameters are utilized. This makes intuitive sense given that a CNN ought to be able to recognise features in an image regardless of their location. Translation invariance is the name given to CNN' robustness.

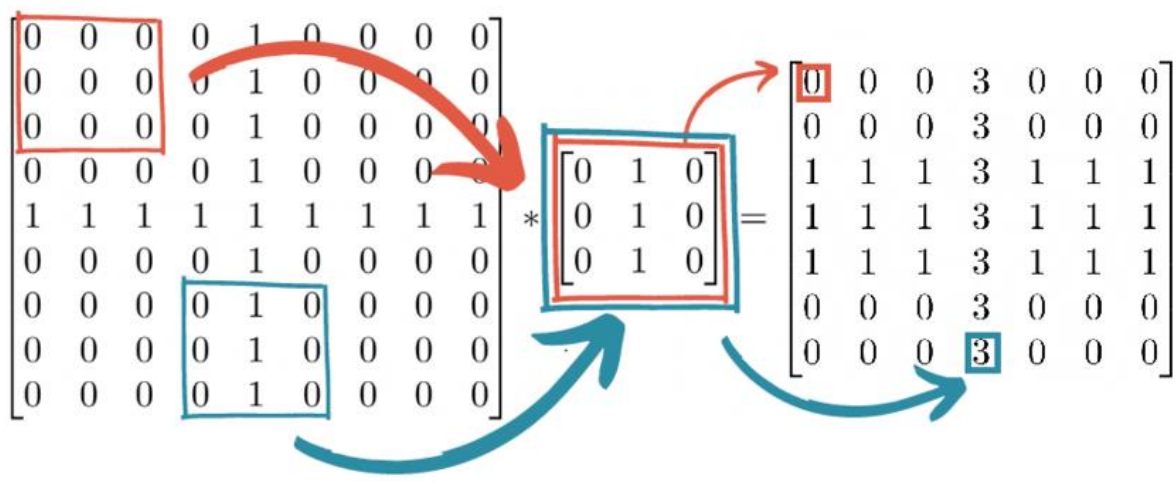


Figure 17: A filter matrix to detect a vertical line

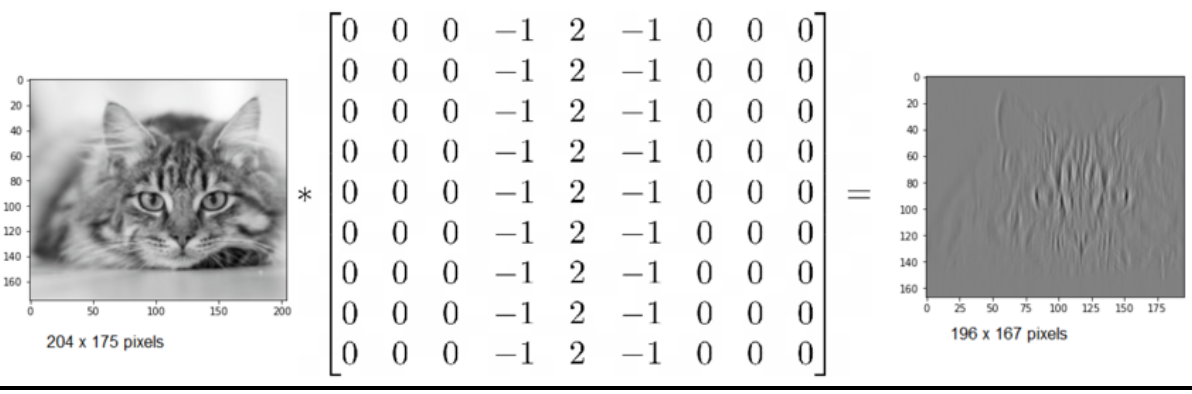


Figure 18: A filter matrix to detect features in image of a cat

The CNN is given more nonlinearity via the activation-function. Together all layer of the N-N can be boiled down to a singular vector-multiplication if somehow the activation-function was absent.

A fundamental CNN may be thought of as a collection of convolutional-layers, an activation function, a pooling (downsampling) layer, and so forth. The first layer recognises basic elements like edges in an image with the repetitive aggregation of these processes, while the second layer starts to identify higher-level features. A CNN can recognise increasingly intricate forms, like eyeballs, by the eleventh layer. It can frequently recognise one human face from another by the twentieth layer. This strength results from the repetitive stacking of processes, each of which is capable of detecting a little bit more complex characteristics than the one before it.

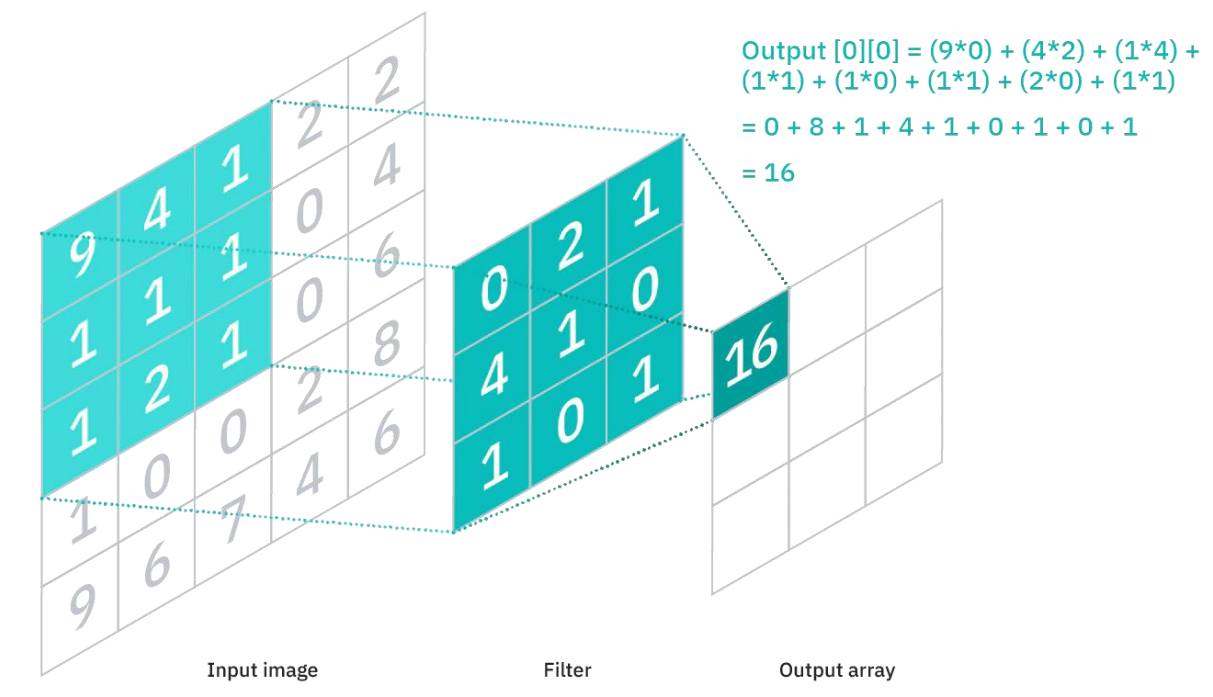


Figure 19: How a filter matrix/kernel convolves

In a number of fields, including image processing, medical imaging, audio and video processing, etc., CNNs have made remarkable advancements.

There are several CNN designs that may be used, and these architectures have been essential in creating the algorithms that power and will continue to drive Artificial Intelligence in the near future. CNNs deliver in-depth findings despite their immense power and complicated resource requirements. Simply identifying patterns and nuances that are so microscopic and subtle that the human eye misses them is what it all boils down to. However, it fails whenever it comes to comprehending an image's contents.

CNNs have unquestionably revolutionized artificial intelligence, despite its limitations. CNNs are being employed in a very large amount of real-world and real-time applications. Our accomplishments are impressive and valuable, as demonstrated by developments in CNN, but we are still far distant from duplicating the essential elements of human intellect.

Implementation(Model Development):

We have used python to implement the above mentioned algorithms(DTs, KNN, RFs, ANN, CNN). We have used sklearn and tensorflow. Both these modules provide various functions for implementing ML and DL. sklearn is generally used for ML algorithms, while keras and tensorflow are used for DL.

First, we loaded the MNIST data, which is open source. In MNIST, each digit is represented by an array of 784 numbers, each representing the value of a pixel. Then, we split the data into tests and training. Out of 70,000, 60,000 were used for training and 10,000 for testing. Then, we built the ML/ DL model and trained it. Then, we evaluated it using our test data. Finally, we recorded the accuracies of all the algorithms we used.

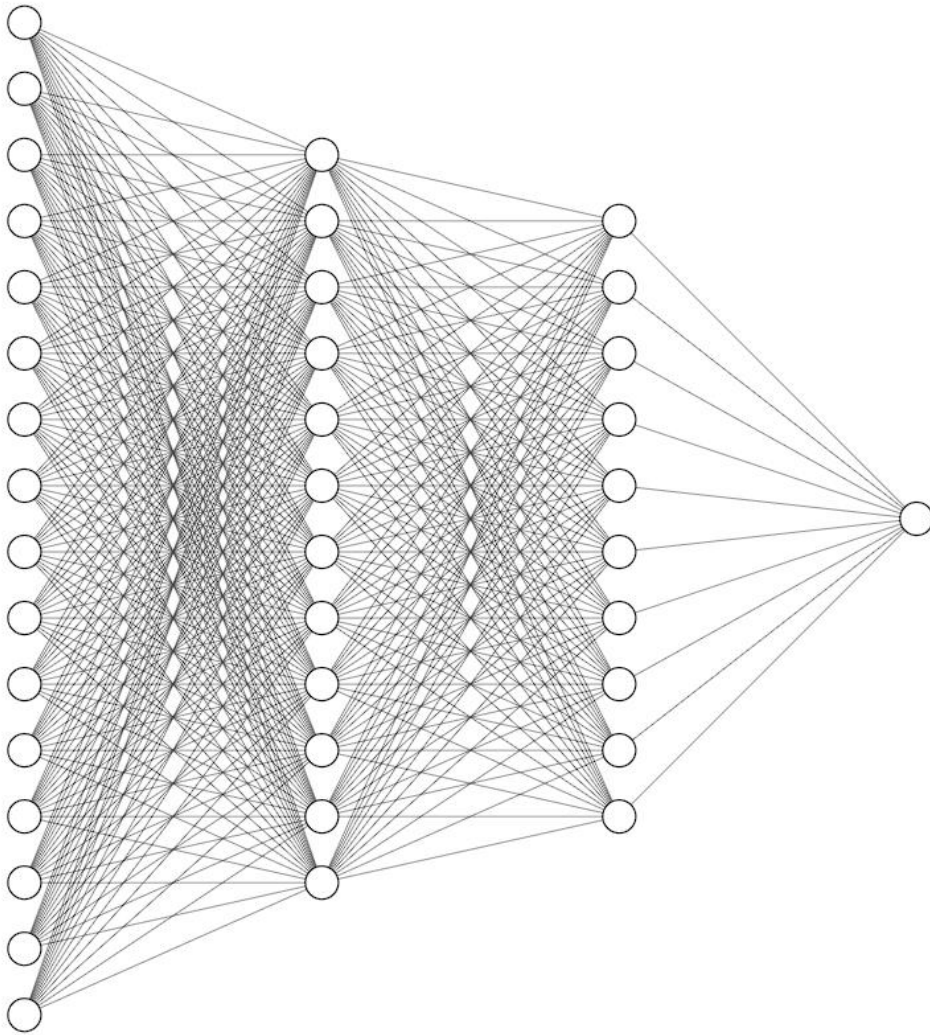


Figure 20: A basic skeleton of a neural network having 2 hidden layers.

A neural network with the input-layer having 16 nodes, and the 2 hidden-layers having 8 nodes each. The output-layer has a single node.

Chapter-4

Experiments and Result Analysis

As performance analysis, we recorded the accuracies of all the algorithms we used. The results are as follows:

Algorithm	Accuracy
KNN	0.9688
Decision Tree, gini index	0.8782
Decision Tree, entropy	0.8880
Random Forest	0.8883
ANN	0.97
CNN	0.99

Expectedly, CNN has the highest accuracy. As we have already explained, CNN performs better when images are involved.

ANN also has a very high accuracy, second only to CNN. This verifies the assumption that neural networks or DL algorithms are generally better than ML algorithms.

KNN has good accuracy, but it takes a lot of time.

DTs, as we have already described, suffer from overfitting. This fact is verified now that they have the lowest accuracy.

When multiple DTs are combined to form Random Forests, accuracy should increase, which is also the case here.

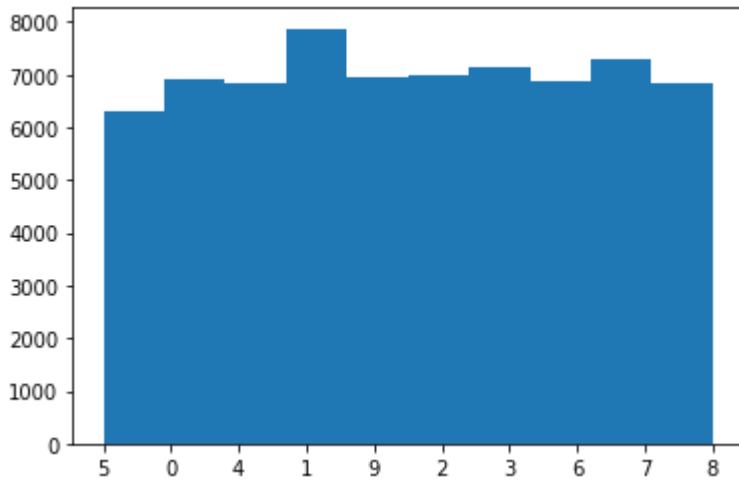


Figure 21: Frequency of digits in MNIST dataset.
 The 70,000 digits have almost the same frequency.

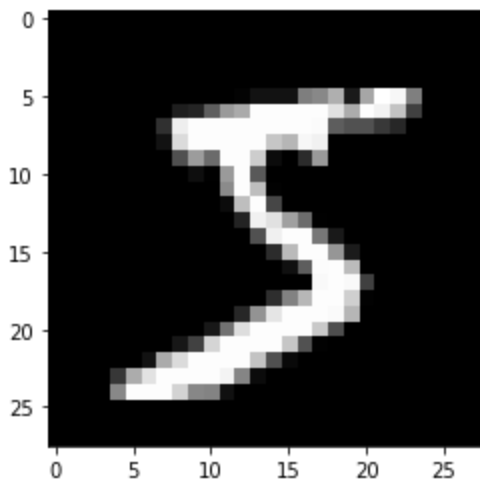


Figure 22: A sample digit (5). Image created using matplotlib library function plt.imshow().
 This function converts a matrix into an image, so that each number in the matrix can represent a value of a pixel.

Visualizations of Neural-Networks:

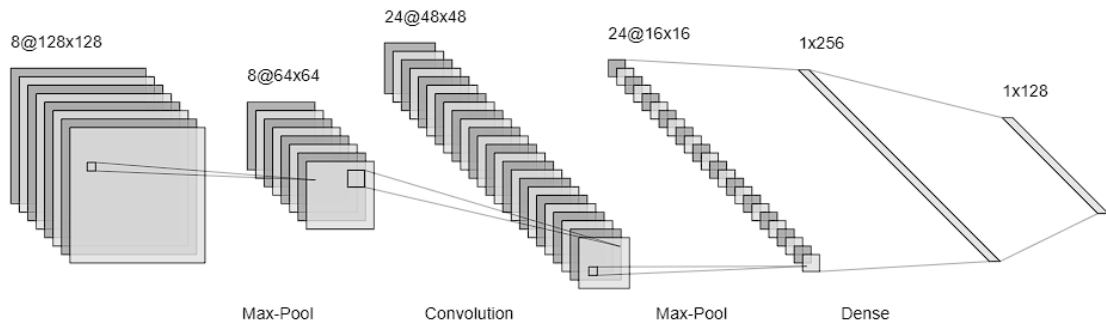


Figure 23: convolutional-neural-network

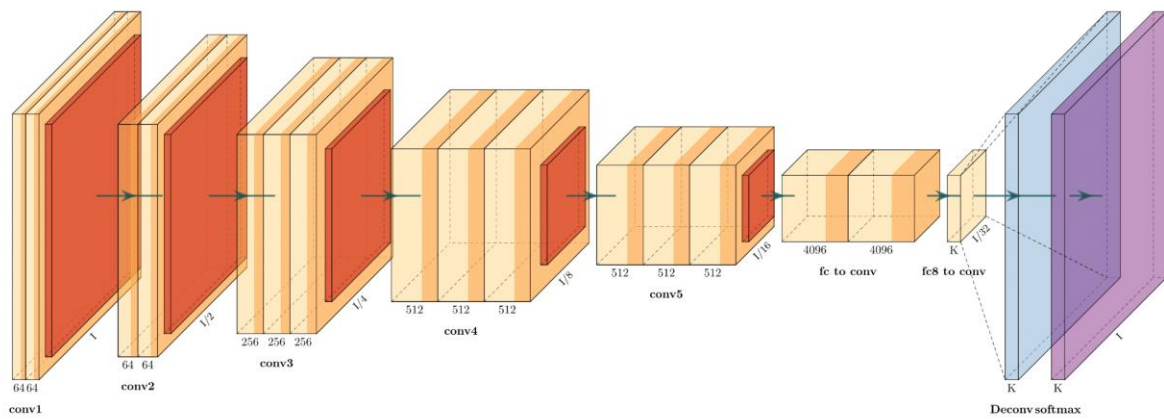


Figure 24: A convolutional-neural-network having 5 convolution-layers

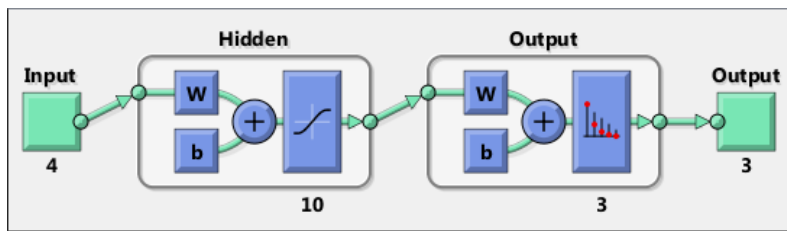


Figure 25: A neural network having weight = w and bias = b and two hidden layers

Chapter-5

Conclusions

5.1 Conclusion:

We have accomplished the following objectives:

- successfully built a model that classifies human-handwritten digits to a reasonable accuracy.
- the model does not take too much resources
- the model does not have a high time complexity
- the model does not have a high space complexity
- the model successfully classifies digits even when it is greatly varied from the standard handwritten digits

We successfully implemented KNN, DTs, R-Fs, ANN and CNN in python.

The accuracy we got was best in the case of CNN, followed by ANN, KNN, R-FR-Fs and DTs. The difference in accuracy was expected, for the reasons we have already mentioned.

We want to classify human-written numbers, and for that, the ML-methods were utilized to classify human-handwritten digits. Each classification technique has its own unique accuracy, space and time requirements.

The issues with hand-written numeral recognition have been researched while using diverse and wholesome approaches. Since every algorithm comes with its own unique set of advantages and disadvantages, a variety of algorithms have been used, and their merits and demerits have been discussed.

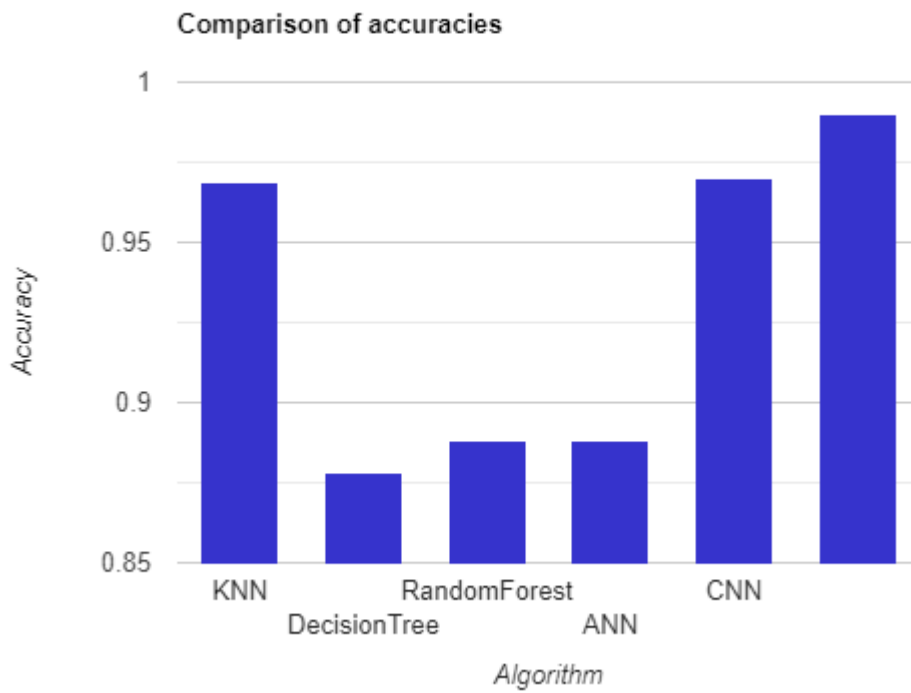


Figure 26: Accuracy of various algorithms

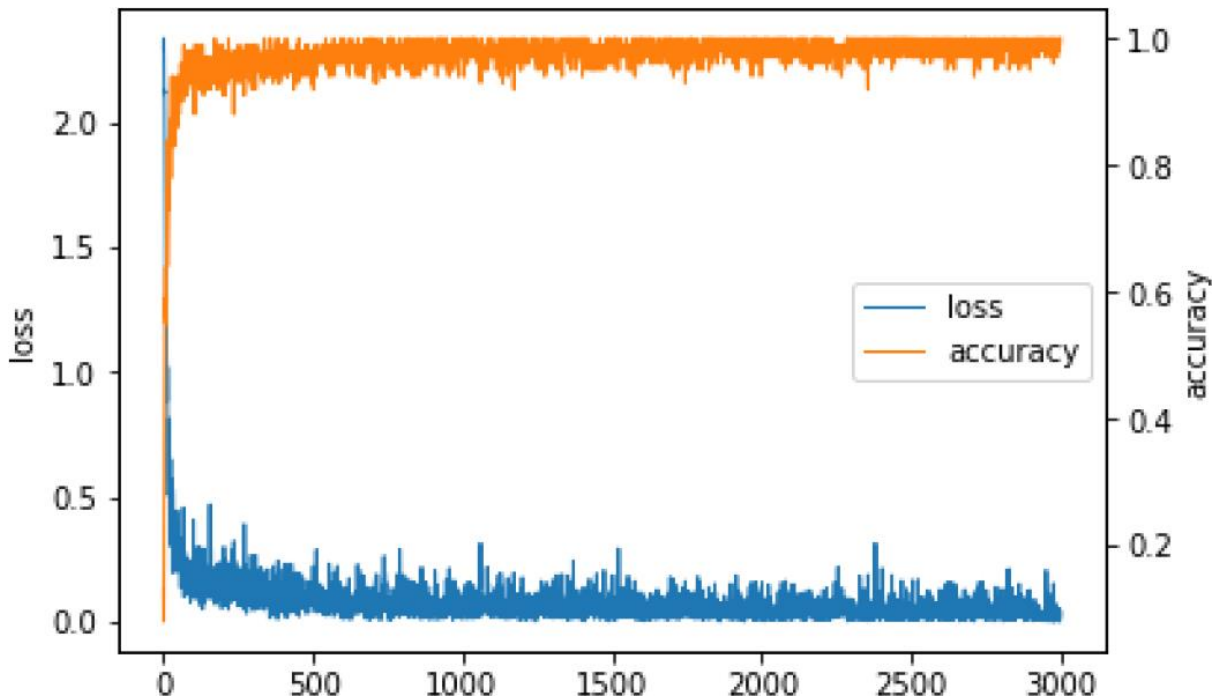


Figure 27: Loss vs Accuracy Plot

An intriguing area of research is sophisticated picture analysis. important for a field of artificial intelligence a range of active research problems nowadays. Recognition of handwritten digits is a thoroughly investigated subfield within the discipline that deals with education models to recognise handwritten digits that have been segmented. Among the most crucial problems in data mining is this. Pattern recognition software, ML, and many other other artificial intelligence fields. A significant effort has been made by researchers in the fields of ML to develop effective methods for approximating recognition from data. Because different communities may employ different handwriting styles while still controlling to draw the same patterns of characters in their recognised script, one of the difficulties in recognising handwritten characters entirely resides in the variance and distortion of the handwritten character set. One of the key problems in the field of digit recognition systems is the identification of the digit from which the best discriminating characteristics may be retrieved. Different types of region sampling strategies are employed in pattern recognition to find these regions. Additionally, the characters data could be created in many sizes and orientations, albeit they must always be printed on a boundary in either a vertical or downward position. In light of these restrictions, a useful handwriting recognition system may be created. Since the majority of people struggle to recognise their own written work, it may be rather tedious at times to read handwritten characters. As a result, there is a restriction on how a writer may write in order for handwritten papers to be recognised. Developing a description of solitary human-handwritten numbers that enable their efficient detection is the major goal of this work. In this study, several ML algorithms were utilized to recognise handwritten numbers. The key challenge in any recognition procedure is to deal with accurate feature extraction and classification methods. In regards to precision and time complexity, the suggested methods attempt to handle both criteria.

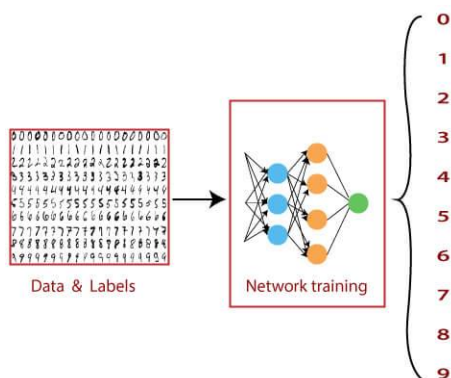


Figure 28: MNIST database going through a neural network

5.2 Future Scope:

Future attempts might examine the success of learning in-depth information and apply it to more challenging image recognition issues. So that a user-friendly programme that can accept input, recognise it, and tell us the identification of a numeric input may be made for computers or mobile devices. The reported results may be significantly improved in terms of detail and accuracy by utilizing a large number of hidden neurons and several convolutional layers. Additionally, accuracy can be improved by utilizing a hybrid model that combines many algorithms. In the near future, this project will be able to include real-time data utilizing human handwriting.

It appears that an AI might handle complicated recognition issues using all these different methods rather than the existing DL techniques. However, in order for that feature to be implemented, organizations and developers must accept new theories and concepts and give some credit where it is due. They may also choose to abandon certain existing ways since even in the best case scenario, the present models may have limitations.

All of the bleeding-edge technology is tested in the health industry. Try using a particular product in a clinical setting to evaluate its applicability. Not an exception is image recognition. Computer - aided diagnostic technology is the most intriguing use of image recognition according. The preliminary machine vision results in a significant amount of additional data processing for the mri images. Faster than the human eye, CNN clinical recognition system can identify irregularities in X-ray and MRI pictures.

Facial recognition/classification warrants its own paragraph. This branch of machine vision deals with images that are more intricate. These photographs might feature human faces or those of animals, reptiles, or other living things/organisms.

Operational complexity—the added to the end of labor required—distinguishes straight image recognition from face recognition. Social media sites like Facebook utilise face recognition for both social media and enjoyment. As a reliable tool for identifying individuals, facial recognition/classification technology is getting popularity. Facial recognition is not as reliable as biometrics or official documentation for authenticating a

persona. Face-id can be beneficial in situations in which there is little evidence to go on to identify the person.

Medicinal chemistry is a significant area of pharmacy that heavily utilizes CNN. Additionally, it ranks among the most creative applications of CNN overall. Drug development and CNNs are not instances of real data tinkering, but "RNN" and stock-market forecasting are. The issue is that the act of finding and developing new drugs takes a long time and is expensive. Flexibility and pricing are essential in the discovery/exploration of drugs. The application of neural- networks is highly suited to the creation of novel pharmaceuticals. There is a lot of data to take into account while developing a new medicine.

CNN disclose and provide intelligible descriptions of concealed facts. Neural-networks show what can be done with their help even for the simplest applications. A lot can be learned about the design and implementation of the visuals from how CNN detects images. Contrarily, CNNs discover novel drugs, which is only one of countless incredible illustrations of where and how ANNs and CNNs are transforming the universe.

Neural nets are in their youth at the moment, and they're already a remarkable innovation that has made significant advances in everything from voice recognition to medical diagnosis. Neural nets are collections of computer programmes made up of tens of billions to trillions of components, which are all intended to act as a synthetic neuron.

A neural-net may learn patterns by being "trained" by being fed data, such as finding the right method to strike a tennis ball or recognising recognizable faces in pictures. After receiving input, neural networks strive to change how they interpret the challenge, "starting to learn" how to score higher over a length of time.

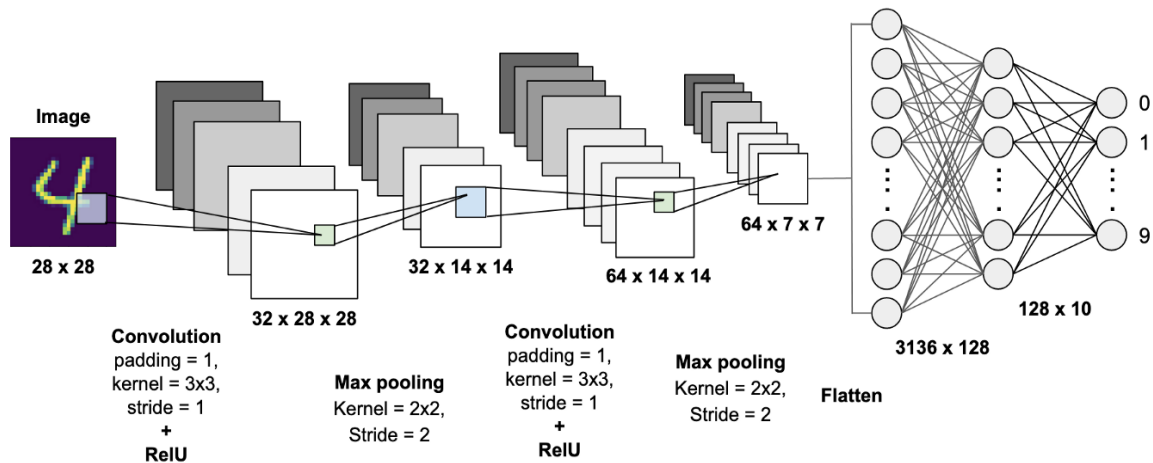


Figure 29: A CNN with 2 convolutional layers, 2 Max-pooling layers

It is simple for a computer to determine the solution to a problem given rigorous guidelines and input constraints. Feature/attribute extraction, also known as feature/attribute selection, is another extremely effective application of neural networks. Another strength of neural networks is adaptability. Once they are formed, they may be used for nearly anything, from assisting individuals in identifying the problems impeding their productivity to enhancing air travel times for more seamless flights. Although scientific utopians have relished proclaiming neural networks' great future, they might not continue to be the primary method of AI or hard problem solving for very long.

Although digital perfectionists have relished proclaiming neural networks' great future, they might not continue to be the primary method of AI or hard problem solving for very long. The strict limitations and significant flaws of neural networks may prevent their further development in the future. Instead, if a new strategy emerges that is sufficiently accessible and has the promise to make it a deserving replacement, researchers and users may start to favor it.

Neural-networks could (and probably must) expand laterally, being used in more varied application areas, as opposed to only developing upwards in terms of quicker processor power and more real immensity. Neural-networks have the potential to help dozens of companies function more effectively, reach new markets, create new products, or gain

customer security. They are grossly neglected. Increased engineering and marketing innovation, broader acceptability, and affordability might lead to more uses for neural-networks.

If neural systems could be combined with a complementing tech, such as frame, their flaws may be readily made up for. Creating a method for various systems to cooperate in order to generate a common outcome would be challenging, but scientists are already researching it. Everything has the capacity to increase in sophistication and strength. Although neural networks have already made a considerable contribution to the field of artificial intelligence, its long-term potential may not be as great as that of kernel approaches or even traditional AI. The fact that neural-networks have a definite maximum bound on their complexity or efficiency discourages many researchers.

The practicalities of creating a neural network are, as you might expect, considerably more intricate and complex than can be suggested with a brief, general description. Learning how to create a neural network is exceedingly challenging, and so many people who start the process finally give up. Additionally, due to the complexity of neural networks, it is sometimes difficult to discern how our algorithms arrive at their results. This renders it somewhat mysterious—even to experts—even though we can assess if their results are true.

Scientists now have established that a specific kind of neural net can be taught to understand the real causation pattern of the navigating problem. These networks need to be more successful than some other neural nets while traveling in a complicated world, such as a place with thick trees or an area frequently altering the climate, because they can comprehend the job directly from image input.

Potential developments of this research may increase the dependability and credibility of computational agents engaged in risky activities like operating an automated vehicle on a crowded roadway. One of the key components of AI technology is neural-networks. They have been available for years and are used in hundreds of apps (you use one whenever your cellphone utilizes biometric technology). However, they are becoming more and more common.

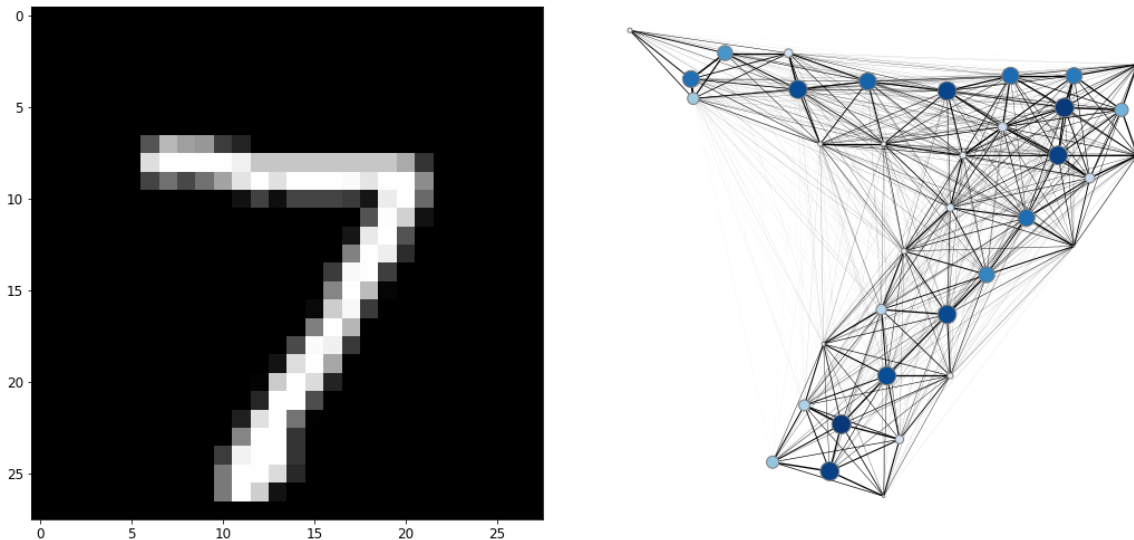


Figure 30: A digit 7, in the MNIST database

Convolution Neural Networks have been proven extremely useful when we are dealing with images.

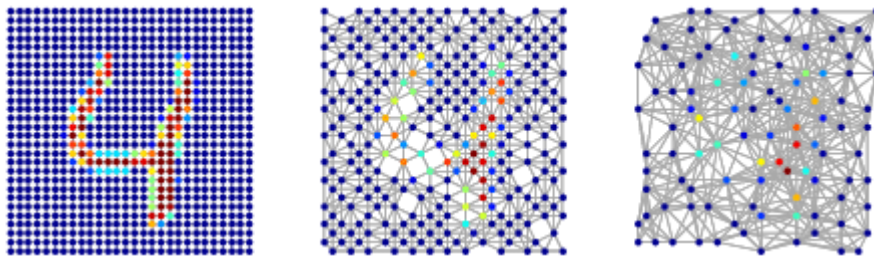


Figure 31: A CNN model extracts the features from the image. First layer extracts very basic features like vertical or horizontal lines, and subsequent layers keep extracting increasingly complex features.

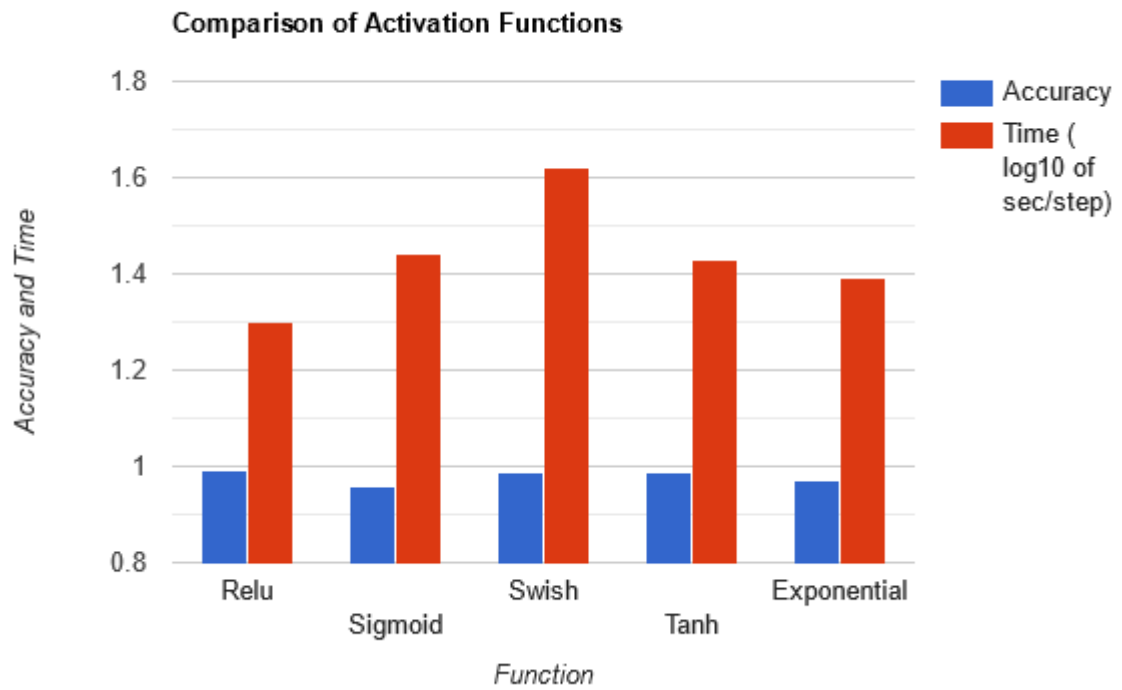


Figure 32: Comparison of accuracies and Time taken by various activation functions.

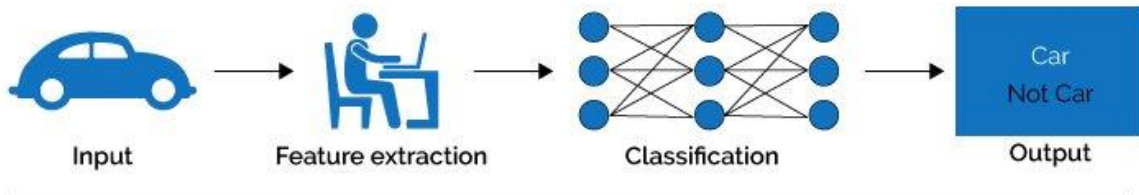
References

- [1] D. Ciresan, U. Meier, L. M. Gambardella and J. Schmidhuber, "Convolutional neural network committees for handwritten character classification", Proc. ICDAR, 2011.
- [2] Yann Lecun and Corinna Cortes, The MNIST database of handwritten digits.
- [3] Ernst Kussul and Tatiana Baidyk, "Improved method of handwritten digit recognition tested on {MNIST} database", Proceedings from the 15th International Conference on Vision Interface. Image and Vision Computing, vol. 22, no. 12, pp. 971-981, 2004.
- [4] Zhihong Man, Kevin Lee, Dianhui Wang, Zhenwei Cao and Suiyang Khoo, "An optimal weight learning machine for handwritten digit image recognition", Special issue on ML in Intelligent Image Processing. Signal Processing, vol. 93, no. 6, pp. 1624-1638, 2013.
- [5] Zhang Ping, D. Bui Tien and Y. Suen Ching, "A novel cascade ensemble classifier system with a high recognition performance on handwritten digits", Pattern Recognition, vol. 40, no. 12, pp. 3415-3429, 2007.
- [6] Lauer Fabien, Y. Suen Ching and Grard Bloch, "A trainable feature extractor for handwritten digit recognition", Pattern Recognition, vol. 40, no. 6, pp. 1816-1824, 2007.
- [7] Cheng-Lin Liu, Kazuki Nakashima, Hiroshi Sako and Hiromichi Fujisawa, "Handwritten digit recognition: benchmarking of state-of-the-art techniques", Pattern Recognition, vol. 36, no. 10, pp. 2271-2285, 2003.
- [8] Angelo Cardoso and Andreas Wichert, "Handwritten digit recognition using biologically inspired features", Neurocomputing, vol. 99, no. 0, pp. 575-580, 2013.
- [9] Berkant Savas and Lars Eldn, "Handwritten digit classification using higher order singular value decomposition", Pattern Recognition, vol. 40, no. 3, pp. 993-1003, 2007.
- [10] C. De Stefano, F. Fontanella, C. Marrocco and A. Scotto di Freca, "A ga-based feature selection approach with an application to handwritten character recognition", Frontiers in Handwriting Processing. Pattern Recognition Letters, vol. 35, no. 0, pp. 130-141, 2014.
- [11] M. Hanmandlu and O.V. Ramana Murthy, "Fuzzy model based recognition of handwritten numerals", Pattern Recognition, vol. 40, no. 6, pp. 1840-1854, 2007.
- [12] Jie Zhou, Adam Krzyzak and Y. Suen Ching, "Verification method of enhancing the recognizers of isolated and touching handwritten numerals", Handwriting Processing and Applications. Pattern Recognition, vol. 35, no. 5, pp. 1179-1189, 2002.

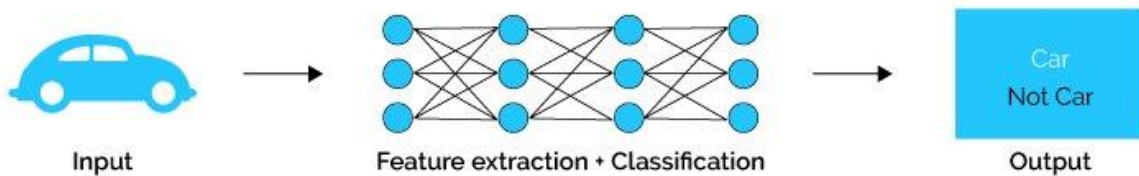
- [13] Ying Wen and Lianghua He, "A classifier for Bangla handwritten numeral recognition", *Expert Systems with Applications*, vol. 39, no. 1, pp. 948-953, 2012.
- [14] Hossein Khosravi and Ehsanollah Kabir, "Introducing a very large dataset of handwritten farsi digits and a study on their varieties", *Pattern Recognition Letters*, vol. 28, no. 10, pp. 1133-1141, 2007.
- [15] Yann LeCun, LD Jackel, L Bottou, A Brunot, C Cortes et al., "Comparison of learning algorithms for handwritten digit recognition", *International conference on ANN*, vol. 60, pp. 53-60, 1995.
- [16] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition", *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2324, 1998.
- [17] Liu, Cheng-Lin, Nakashima, Kazuki, Sako, Hiroshi, et al., "Handwritten digit recognition: benchmarking of state-of-the-art techniques", *Pattern Recognition*, vol. 36, no. 10, pp. 2271-2285, 2003.
- [18] V Vijaya Kumar, A Srikrishna, B Raveendra Babu and M Radhika Mani, "Classification and recognition of handwritten digits by using mathematical morphology", *Sadhana*, vol. 35, no. 4, pp. 419-426, 2010.

Appendices

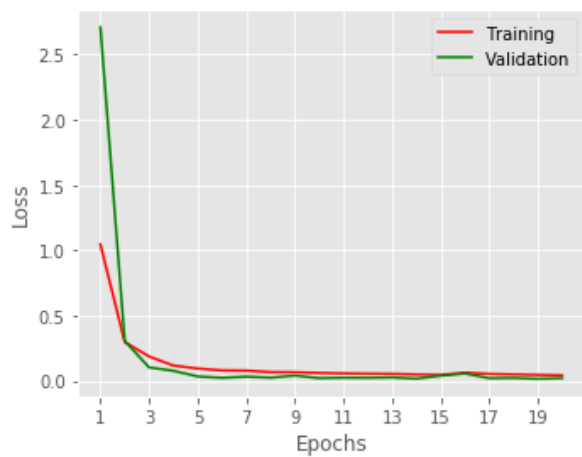
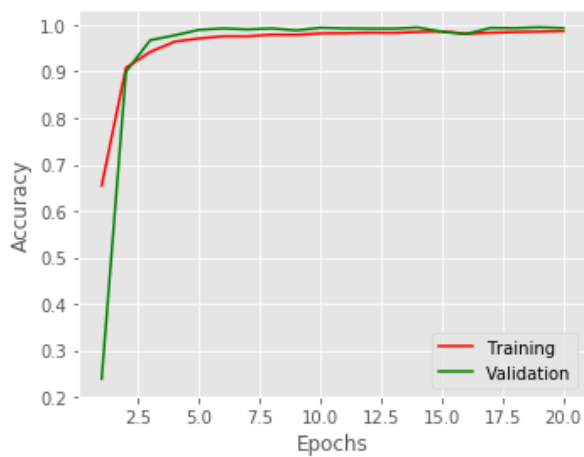
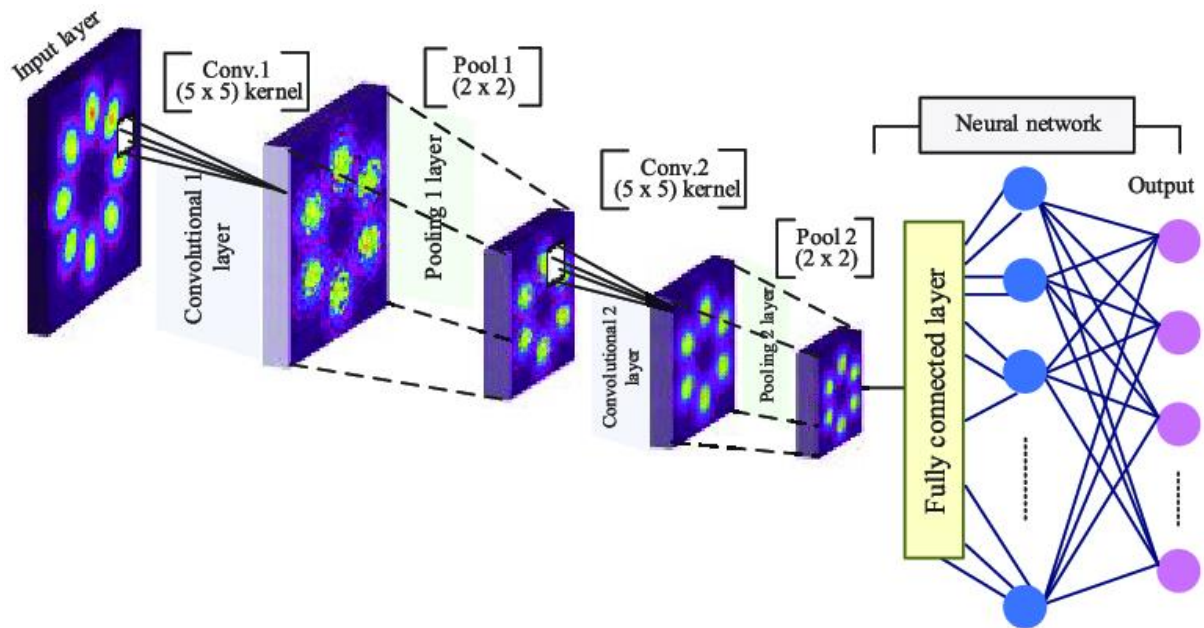
Machine Learning



Deep Learning



Artificial Neural Network	Convolutional Neural Network
1. Uses Fully Connected Layers.	1. Uses Partially Connected Layers, Size of Connection depends on Size of the Filter
2. Can be used only for small images	2. Can be used for any image
3. Considers entire Image for learning the Patterns	3. Considers only the part of the Image, which is in the Receptive Field of the Filter
4. Number of Parameters will be very High. For 28 * 28 Image, and if the Layer has 32 Neurons, number of Parameters for one Layer will be $28 * 28 * 32 = 25,088$. Consequently, computations involved in ANN are more	4. Number of Parameters will be very Less, compared to ANN. For 28 * 28 Image, and if the Layer has 32 Filters, with each Filter Size as (5*5), number of Parameters for that Layer will be $(5*5*1+1)*32 = 832$. Computations involved are less compared to that of ANN.
5. Not very efficient for Images <u>inspite</u> of being expensive	5. Very efficient for Images and very less expensive compared to ANN
6. Doesn't learn the patterns in Spatial Hierarchy i.e., Higher Layers of ANN doesn't combine the lower layers outputs	6. Learns Spatial Hierarchy of Patterns i.e., Higher Layers of CNN are formed by combining Lower Layers. This helps in identifying the Patterns more effectively than ANN
7. Not Translation Invariant : Once a regular DNN has learned to recognize a pattern in one location, it can recognize it only in that particular location. In short, it we need to Train the Image again (can't reuse the previous weights) if it is Rotated or Shifted	7. Translation Invariant : Once the CNN has learned to recognize a pattern in one location, it can recognize it in any other location. In short, learning (weights) can be reused even if the Image is Rotated or Shifted



Code:

```

from sklearn.datasets import fetch_openml
from matplotlib import pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression

```

```
mnist = fetch_openml('mnist_784')
```

```
x, y = mnist['data'], mnist['target']
```

```
x
```

```
x.loc[0:10, 'pixel1':'pixel10']
```

```

x.iloc[0:10, 0:10]

plt.hist(y)

for i in range(0,10):
    im = x.iloc[i].values.reshape((28,28))
    plt.figure()
    plt.imshow(im, cmap='gray')
    print("label: ", y[i])
    plt.show()

x_train, y_train = x.iloc[:60000], y.iloc[:60000]

x_test, y_test = x.iloc[60000:], y.iloc[60000:]

knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(x_train, y_train)

q = knn.predict(x_test)

score = accuracy_score(y_test, q)
score

decision_tree = DecisionTreeClassifier(criterion='gini')
decision_tree.fit(x_train, y_train)

q = decision_tree.predict(x_test)

score = accuracy_score(y_test, q)
score

decision_tree = DecisionTreeClassifier(criterion='entropy')
decision_tree.fit(x_train, y_train)

q = decision_tree.predict(x_test)

score = accuracy_score(y_test, q)
score

random_forest = RandomForestClassifier(n_estimators = 200)
random_forest.fit(x_train, y_train)

q = decision_tree.predict(x_test)

score = accuracy_score(y_test, q)
score

import numpy as np
from tensorflow import keras
from tensorflow.keras import layers
num_classes = 10
input_shape = (28, 28, 1)

(x_train, y_train), (x_test, y_test) = keras.datasets.mnist.load_data()

x_train = x_train.astype("float32") / 255
x_test = x_test.astype("float32") / 255

x_train = np.expand_dims(x_train, -1)
x_test = np.expand_dims(x_test, -1)
print("x_train shape:", x_train.shape)
print(x_train.shape[0], "train samples")
print(x_test.shape[0], "test samples")

y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

model = keras.Sequential(
    [
        keras.Input(shape=input_shape),
        layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
    ]
)

```



```
        layers.Flatten(),
        layers.Dropout(0.5),
        layers.Dense(num_classes, activation="softmax"),
    ]
)

model.summary()

batch_size = 128
epochs = 15

model.compile(loss="categorical_crossentropy", optimizer="adam", metrics=["accuracy"])

model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs, validation_split=0.1)

score = model.evaluate(x_test, y_test, verbose=0)
print("Test loss:", score[0])
print("Test accuracy:", score[1])
```

Result:

Accuracies:

KNN: 0.9688

Decision Tree, ginni index: 0.8782

Decision tree, entropy: 0.8883

Random Forest: 0.8883

ANN: 0.97

CNN: 0.99
