DEPARTMENTOFCOMPUTER SCIENCE & ENGINEERING HALDIA INSTITUTE OF TECHNOLOGY, HALDIA MAY 2022

Handwritten Digit Recognition using Deep Learning

Report submitted to

Haldia Institute of Technology, Haldia

for the award of the degree of

Bachelors of Technology in Computer Science & Engineering

by

Aditya Raj (10300118135)

Abhishek Ranjan (10300118140)

Abhishek Kumar (10300118141)

Abhishek Vijoy (10300118139)

DECLARATION

I/We certify that

a. The work contained in this report is original and has been done by me/us underthe guidance of my/our supervisor(s).
b. The work has not been submitted to any other Institute for any degree or diploma.
c. I/We have followed the guidelines provided by the Institute in preparing the report.
d. I/We have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.
e. Whenever we have used materials (data, theoretical analysis, figures, and text) from other sources, I /we have given due credit to them by citing them in the text of thereport and giving their details in the references.

Signature of the
Students

CERTIFICATE

This is to certify that the Dissertation Report entitled, "**HANDWRITTEN DIGIT RECOGNITION USING DEEP LEARNING**" submitted by **Aditya Raj (10300118135), Abhishek Ranjan (10300118140), Abhishek Kumar (10300118141), Abhishek Vijoy (10300118139)** to Haldia Institute of Technology, Haldia, India, is a record of bonafide Project work carried out by him/her under my/our supervision and guidance and is worthy of consideration for the award of the degree of Bachelor of Technology in Computer Science and Engineering of the Institute.

Supervisor	Supervisor
Date:	

ACKNOWLEDGEMENT

We would like to acknowledge the contributions of the following people without whose help and guidance this report would not have been completed. We acknowledge the counsel and support of our training coordinator,
Mrs. Rajrupa Metia, Assistant Professor, CSE Department,
with respect and gratitude, whose expertise, guidance, support, encouragement, and enthusiasm has made this report possible. Her feedback vastly improved the quality of this report and provided an enthralling experience. We are indeed proud and fortunate to be supported by her. We are also thankful to
Prof. (Dr.) Subhankar Joardar, H.O.D of Computer Science Engineering Department, Haldia Institute of Technology, Haldia, WB for his constant encouragement, valuable suggestions and moral support and blessings. Although it is not possible to name individually, we shall ever remain indebted to the **faculty** members of Haldia Institute of Technology Haldia, WB for their persistent support and cooperation extended during this work. This acknowledgement will remain incomplete if we fail to express our deep sense of obligation to our **parents** and **God** for their consistent blessings and encouragement.

ABSTRACT

Digit Recognition is a noteworthy and important issue. As the manually written digits are not of a similar size, thickness, position and direction, in this manner, various difficulties must be considered to determine the issue of handwritten digit recognition. The uniqueness and assortment in the composition styles of various individuals additionally influence the example and presence of the digits. It is the strategy for perceiving and arranging transcribed digits. It has a wide range of applications, for example, programmed bank checks, postal locations and tax documents and so on.

Optical Character Recognition is a subfield of Image Processing which is concerned with extracting text from images or scanned documents. In this project, we have chosen to focus on recognizing handwritten digits available in the MNIST database.

The handwritten digit recognition is the ability of computers to recognize human handwritten digits. It is a hard task for the machine because handwritten digits are not perfect and can be made with many different flavors. The handwritten digit recognition is the solution to this problem which uses the image of a digit and recognizes the digit present in the image.

Contents

Contents Of Images

Chapter: 01

Introduction and Literature Review

1.1 Introduction:

Handwritten digits recognition becomes increasingly important in the modern world due to its practical applications in our daily life . In recent years, numerous recognition systems have been introduced within many applications where high classification efficiency is required. It helps us to solve more complex problems and makes ease our tasks. An early stage handwritten digit recognition was presented for zip code recognition. Automatic processing of bank checks, the postal address is widely used applications of handwritten digit recognition. A human being has been proffered a common bias to distinguish numerous objects with variations such as digits, letters, faces, voice . However, executing a computerized system to do certain kinds of duties is a very complex and challenging matter. In addition, pattern identification is the fundamental ingredient of a computer vision and artificial intelligence-based system. In this paper, we implemented an artificial neural network (ANN) and trained it to recognize handwritten digits from 0 to 9. A node in a neural network can be understood as a neuron in the brain. Each node is connected to other nodes through weights which are adjusted in the machine learning process during training. A value is calculated for each node based on values and ways of previous nodes. This process is called forward propagation. The final output of the network is associated with the target output, then weights are calibrated to minimize a transgression function describing whether the network guessed correctly . This process is called back propagation . To add more complexity and accuracy in the neural networks, the networks have multiple layers. In between a completely connected neural network, there are some multiple layers exist, namely input, output, and hidden layers. In a fully connected neural network nodes in each layer are connected to the nodes and the layers before and after them. Recognition is identifying or distinguishing a thing or an individual from the past experiences or learning. Similarly, Digit Recognition is nothing but recognizing or identifying the digits in any document. Digit recognition framework is simply the working of a machine to prepare itself or interpret the digits. Handwritten Digit Recognition is the capacity of a computer to interpret the manually written digits from various sources like messages, bank cheques, papers, pictures, and so forth and in various situations for web-based handwriting recognition on PC tablets, identifying number plates of vehicles, handling bank cheques, digits entered in any forms etc.

Machine Learning provides various methods through which human efforts can be reduced in recognizing the manually written digits. Deep Learning is a machine learning method that trains computers to do what easily falls into place for people: learning through examples. With the utilization of deep learning methods, human attempts can be diminished in perceiving, learning, recognizing and in a lot more regions. Using deep learning, the computer learns to carry out classification works from pictures or contents from any document. Deep Learning models can accomplish state-of-art accuracy, beyond the human level performance. The digit recognition model uses large datasets in order to recognize digits from distinctive sources. Handwriting recognition of characters has been around since the 1980s. The task of handwritten digit recognition, using a classifier, has extraordinary significance and use such as – online digit recognition on PC tablets, recognize zip codes on mail, processing bank check amounts, numeric sections in structures filled up by hand and so on. There are diverse challenges faced while attempting to solve this problem. The handwritten digits are not always of the same size, thickness, or orientation and position relative to the margins.

1.2 Literature Review:

Vijayalaxmi R Rudraswamimath and Bhavanishankar K in their paper demonstrated that utilizing Deep Learning systems, he had the capacity to get an extremely high measure of accuracy. By utilizing Machine Learning Algorithms like SVM, KNN and RFC and with Deep Learning calculation like multilayer CNN utilizing Keras with Theano and Tensorflow. Utilizing these, the accuracy of 98.70% utilizing CNN (Keras + Theano) when contrasted with 97.91% utilizing SVM, 96.67% utilizing KNN, 96.89% utilizing RFC was obtained.

In a paper published by Saeed AL-Mansoori, Multilayer Perceptron (MLP) Neural Network was implemented to recognize and predict handwritten digits from 0 to 9. The proposed neural system was trained and tested on a dataset achieved from MNIST.

In a paper published by K.T Islam, G. Mujtaba, Dr. Ram Gopal Raj and HF Nweke testing has been conducted from publicly available MNIST handwritten database. From the MNIST database, they extracted 28,000 digits images for training and 14,000 digits images for performing the test. Their multi-layer artificial neural network has an accuracy of 99.60% with test performance.

Chapter: 2
Theoretical Study

2.1 Background:

Handwriting digit recognition is one of the research fields in computer vision, artificial intelligence, and pattern recognition. A computer application that performs handwriting recognition can be argued to have the ability to acquire and detecting characters in pictures, paper documents, and other sources and convert them into electronic format or machine-encoded form. The system may obtain Handwriting sources from a piece of paper through optical scanning or intelligent word recognition. Also, the system may be designed to detect the movement of the pen tip on the screen. In other words, handwriting recognition may involve a system detecting movements of a pen tip on the screen to get a clue of the characters being written. Handwriting recognition can be classified into two: offline recognition and online recognition. Offline handwriting recognition involved the extraction of text or characters from an image to have letter codes that can be used within a computer. It involves obtaining digital data from a static representation of handwriting. A system is provided with a Handwriting document to read and convert the handwriting to a digital format. Online handwriting recognition, on the other hand, involved automatic detection or conversion of characters as they are written on the specialized screen. In this case, the system sensors movement of pen-tip to detect characters and words. Different methods and techniques are used to ensure that computer systems can read characters from Handwriting images and documents. Among the existing techniques that are used to model, and train Handwriting character recognition include neural network, Hidden Markov Model (HMM), Machine Learning, and Support Vector Machine, to mention a few. This paper focuses on artificial intelligence networks, machine learning, Hidden Markov Model, and the Support Vector Machine.

2.2 Artificial Intelligence:

The idea of reading Handwriting characters, digits, and words by computer systems can be argued to be an imitation of a human being. In other words, such a system can be argued that they use artificial intelligence to read handwriting from images or any Handwriting source. Artificial intelligence refers to intelligence that is demonstrated by machines. The term is used to describe computer or machines that can mimic "cognitive" functions that are associated with the human mind. Artificial intelligence allows the machine to learn from experience, adjust to new data (inputs), and perform tasks that can be performed by humans . Branches of artificial intelligence include machine learning, neuron network, and deep learning.

2.3Machine Learning

Machine learning technology is inspired by psychology and biology that focus on learning from a set of data. The central assumption is that machines can learn to perform given tasks by learning from data. A machine learning model is provided with training data that is specific to the given problem domain and the solution to each instance of the problem. That way, the model learns how to solve certain problems based on learning. It shows a simple demonstration of the machine learning model used in the handwriting recognition system. The model takes an image that has a Handwriting digit and determines the specific digit based on the learning data. Machine learning is an area of artificial intelligence (AI) with a concept that a computer program can learn and adapt to new data without human intervention. A complex algorithm or source code is built into a computer that allows for the machine to identify data and build predictions around the data that it identifies. Machine learning is useful in parsing the immense amount of information that is consistently and readily available in the world to assist in decision making. Machine learning can be applied in a variety of areas, such as in investing, advertising, lending, organizing news, fraud detection, and more.

2.3.1 Understanding Machine Learning

Various sectors of the economy are dealing with huge amounts of data available in different formats from disparate sources. The enormous amount of data, known as big data, is becoming easily available and accessible due to the progressive use of technology, specifically advanced computing capabilities and cloud storage. Companies and governments realize the huge insights that can be gained from tapping into big data but lack the resources and time required to comb through its wealth of information. As such, artificial intelligence measures are being employed by different industries to gather, process, communicate, and share useful information from data sets. One method of AI that is increasingly utilized for big data processing is machine learning.

The various data applications of machine learning are formed through a complex algorithm or source code built into the machine or computer. This programming code creates a model that identifies the data and builds predictions around the data it identifies. The model uses parameters built in the algorithm to form patterns for its decision-making process. When new or additional data becomes available, the algorithm automatically adjusts the parameters to check for a pattern change, if any. However, the model shouldn’t change.

2.3.2 Machine Learning Categories

Machine Learning is generally categorized into three types: Supervised Learning, Unsupervised Learning, Reinforcement learning

Supervised Learning:

In supervised learning the machine experiences the examples along with the labels or targets for each example. The labels in the data help the algorithm to correlate the features.

Two of the most common supervised machine learning tasks are classification and regression.

In classification problems the machine must learn to predict discrete values. That is, the machine must predict the most probable category, class, or label for new examples. Applications of classification include predicting whether a stock's price will rise or fall, or deciding if a news article belongs to the politics or leisure section.

In regression problems the machine must predict the value of a continuous response variable. Examples of regression problems include predicting the sales for a new product, or the salary for a job based on its description.

Unsupervised Learning:

When we have unclassified and unlabelled data, the system attempts to uncover patterns from the data . There is no label or target given for the examples. One common task is to group similar examples together called clustering.

Reinforcement Learning:

Reinforcement learning refers to goal-oriented algorithms, which learn how to attain a complex objective (goal) or maximize along a particular dimension over many steps. This method allows machines and software agents to automatically determine the ideal behaviour within a specific context in order to maximize its performance. Simple reward feedback is required for the agent to learn which action is best; this is known as the reinforcement signal. For example, maximize the points won in a game over many moves.

Underfitting:

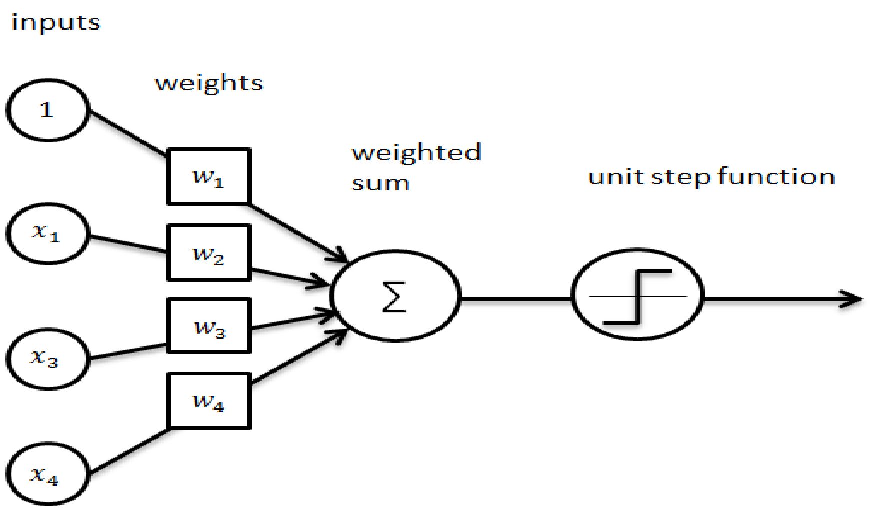
A statistical model or a machine learning algorithm is said to have underfitting when it cannot capture the underlying trend of the data. (It's just like trying to fit undersized pants!) Underfitting destroys the accuracy of our machine learning model. Its occurrence simply means that our model or the algorithm does not fit the data well enough. It usually happens when we have fewer data to build an accurate model and also when we try to build a linear model with fewer non-linear data. In such cases, the rules of the machine learning model are too easy and flexible to be applied on such minimal data and therefore the model will probably make a lot of wrong predictions. Underfitting can be avoided by using more data and also reducing the features by feature selection.

Overfitting:

A statistical model is said to be overfitted when we train it with a lot of data (just like fitting ourselves in oversized pants!). When a model gets trained with so much data, it starts learning from the noise and inaccurate data entries in our data set. Then the model does not categorize the data correctly, because of too many details and noise. The causes of overfitting are the non-parametric and non-linear methods because these types of machine learning algorithms have more freedom in building the model based on the dataset and therefore they can really build unrealistic model.

2.4 Artificial Neural Network (ANN)

Artificial Neural Network (ANN) refers to information processing paradigm or computing systems that are inspired by biological neural networks that constitute the human brain. The systems are not identical to the biological neural systems, but they are designed to process information the same way the human brain and animal brain process information. The networks are composed of many interconnected neurons working in unison to achieve specific goals. Just like the human brain, ANN learns from example. Hence, an ANN can be configured for an application, such as data classification or character recognition through the learning process. The learning process involves adjusting the system to a connection. The artificial neural network comprises a network of multiple simple processors, each with a small amount of local memory. The processors (units) are linked together by unidirectional communication channels and operates only on local data and input their get through their links.



IMG 01: ANN

2.5 Biological Neuron and ANN

As indicated earlier, artificial neural networks are inspired by the biological neural system. Hence, learning how biological neurons works can help to understand the artificial neural network. The human body's neural system consists of three stages: neural network, receptors, and effectors. The first phase is the receptor which receives stimuli from the external or internal environment and then passes the information to neurons. The second phase involves the processing of information by the neural network to make a proper decision of output. The third and final stage involves translation of the electrical impulses into responses to the external environment. An artificial neural network can be argued to be a simplified imitation of the central nervous system. The structural constituents of human brains known as neurons perform computations such as logical inference, cognition, and pattern recognition. It is just a simple representation of a neural network system that does not do much different from what a traditional computer can do. This presents a more complicated model (McCulloch and Pitts Model) which is different from the previous model since it has inputs that are "weighted". That means each input has a different effect on decision making. The weight of an input can be described as the number which when multiplied with the input, it results in weighted input. The results are then added together, and if they exceed the certain predetermined threshold value, the neuron fires, else, the neuron does not fire.

2.6 Deep Neural Network

The neural network has layers of units where each layer takes some value from the previous layer. That way, systems that are based on neural networks can compute inputs to get the needed output. The same way neurons pass signals around the brain, and values are passed from one unit in an artificial neural network to another to perform the required computation and get new value as output. The united are layers, forming a system that starts from the layers used for imputing to layer that is used to provide the output. The layers that are found between the input and output layers are called the hidden layer. The hidden layers refer to a deep neural network that is used for computation of the values inputted in the input layer. The term "deep" is used to refer to the hidden layers of the neural network. In Handwriting character recognition systems, the deep neural network is involved in learning the characters to be recognized from Handwriting images. With enough training data, the deep neural network can be able to perform any function that a neural network is supposed to do. It is only possible if the neural network has enough hidden layers, although the smaller deep neural network is more computationally efficient than a more extensive deep neural network.

2.7 Neural Network Architecture

HRS systems are most efficient when they are based on neural networks. Hence, there is a need to understand the neural network architecture. The neural network architecture refers to the elements that are connected to make a network that is used for handwriting recognition. The human brain works loosely to inspire neural networks. It is based on the idea of how neurons pass signals around the human brain to process input into an output. Several units are layers to form a network and arrange from the ones that are responsible for receiving input to the layer that is responsible for output values. Between the output and input level layers, there is a hidden layer that is involved in much of processing. Different neural network architectures can be used to provide different results from the input images of handwriting. It is because architectures are based on different parameters, data, and duration of training. The size of a deep neural network layer is dependent on the work that the system is supposed to do. However, in most cases, more computational efficient smaller hidden layers can be developed to achieve the same task as one that can be achieved with an exponentially large deep neural network. The deep neural network is supposed to memorize the training data to be able to recognize handwriting. Hence, deep neural networks are commonly used in optical character recognition systems.

The architecture of the neural network refers to elements such as the number of layers in the network, the number of units in each layer, and how the units are connected between layers. As neural networks are loosely inspired by the workings of the human brain, here the term unit is used to represent what we would biologically think of as a neuron. Like neurons passing signals around the brain, units take some values from previous units as input, perform a computation, and then pass on the new value as output to other units. These units are layered to form the network, starting at a minimum with one layer for inputting values, and one layer to output values. The term *hidden layer* is used for all of the layers in between the input and output layers, i.e., those "hidden" from the real world.

Different architectures can yield dramatically different results, as the performance can be thought of as a function of the architecture among other things, such as the parameters, the data, and the duration of training.

2.8 Exploring ANN

Artificial neural networks (ANNs) are biologically inspired computational networks. Among the various types of ANNs, in this chapter, we focus on *multilayer perceptions* (MLPs) with backpropagation learning algorithms. MLPs, the ANNs most commonly used for a wide variety of problems, are based on

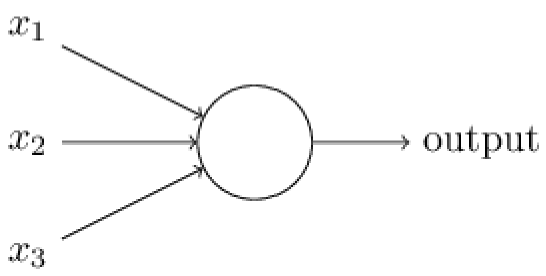
a supervised procedure and comprise three layers: input, hidden, and output. Various aspects of MLPs i.e. structure, algorithm, data preprocessing, overfitting, and sensitivity analysis. In addition, we outline the advantages and disadvantages of MLPs and recommend their usage in ecological modeling.
ANNs are nonlinear statistical models which display a complex relationship between the inputs and outputs to discover a new pattern. A variety of tasks such as image recognition, speech recognition, machine translation as well as medical diagnosis makes use of these artificial neural networks.

An important advantage of ANN is the fact that it learns from the example data sets. Most commonly usage of ANN is that of a random function approximation. With these types of tools, one can have a cost-effective method of arriving at the solutions that define the distribution. ANN is also capable of taking sample data rather than the entire dataset to provide the output result. With ANNs, one can enhance existing data analysis techniques owing to their advanced predictive capabilities.

2.8.1 Perceptrons

Perceptron's were developed in the 1950s and 1960s by the scientist Frank Rosenblatt, inspired by earlier work by Warren McCulloch and Walter Pitts. Today, it's more common to use other models of artificial neurons - in this book, and in much modern work on neural networks, the main neuron model used is one called the *sigmoid neuron*. We'll get to sigmoid neurons shortly. But to understand why sigmoid neurons are defined the way they are, it's worth taking the time to first understand perceptrons.

A perceptron takes several binary inputs, x_1, x_2, …, and produces a single binary output:



IMG 02: PERCEPTRON

In the example shown the perceptron has three inputs, x_1, x_2, x_3. In general, it could have more or fewer inputs. Rosenblatt proposed a simple rule to compute the output. He introduced *weights*, $w_1, w_2, \dots$, real numbers expressing the importance of the respective inputs to the output. The neuron's output, 0 or 1, is determined by whether the weighted sum $\sum_j w_j x_j$ is less than or greater than some *threshold value*. Just like the weights, the threshold is a real number which is a parameter of the neuron. To put it in more precise algebraic terms:

$$\text{Output} = \begin{cases} 0 \text{ if } \sum_j w_j x_j \leq \text{threshold} \\ 1 \text{ if } \sum_j w_j x_j > \text{threshold} \end{cases}$$

That's all there is to how a perceptron works!

2.8.2 Sigmoid Neurons

The output from perceptron. In this post, we will talk about the motivation behind the creation of sigmoid neuron and working of the sigmoid neuron model. Perceptron model takes several real-valued inputs and gives a single binary output. In the perceptron model, every input x_i has weight w_i associated with it. The weights indicate the importance of the input in the decision-making process. The model output is decided by a threshold $\mathbf{W_o}$ if the weighted sum of the inputs is greater than threshold $\mathbf{W_o}$ output will be 1 else output will be 0. In other words, the model will fire if the weighted sum is greater than the threshold.

Chapter: 03

Implementation

3.1 Importing Libraries

Importing libraries into a program is a cost-effective way of minimizing the high-risk aspect of designing, developing, and testing software that has already gone through the development cycle.

TensorFlow is an end-to-end open-source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications.

Keras is an open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library. Up until version 2.3, Keras supported multiple backends, including TensorFlow, Microsoft Cognitive Toolkit, Theano, and Plaid ML.

```
#Importing Libraries
import tensorflow
from tensorflow import keras
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense,Flatten
```

3.2 MNIST Dataset

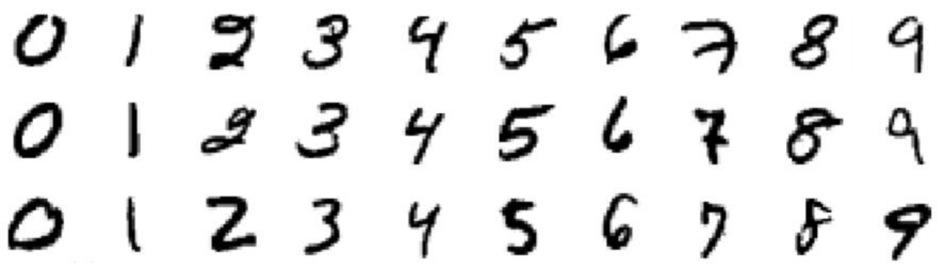
The MNIST database of handwritten digits, has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and cantered in a fixed-size image.

It is a good database for people who want to try learning techniques and pattern recognition methods on real-world data while spending minimal efforts on pre-processing and formatting.

The original black and white (bilevel) images from NIST were size normalized to fit in a 20x20 pixel box while preserving their aspect ratio. The resulting images contain grey levels as a result of the anti-aliasing technique used by the normalization algorithm. the images were centered in a 28x28 image by computing the center of mass of the pixels, and translating the image so as to position this point at the center of the 28x28 field.

With some classification methods (particularly template-based methods, such as SVM and K-nearest neighbours), the error rate improves when the digits are centered by bounding box rather than center of mass. If you do this kind of pre-processing, you should report it in your publications.

The MNIST database was constructed from NIST's Special Database 3 and Special Database 1 which contain binary images of handwritten digits.



IMG 03: MNIST

NIST originally designated SD-3 as their training set and SD-1 as their test set. However, SD-3 is much cleaner and easier to recognize than SD-1. The reason for this can be found on the fact that SD-3 was collected among Census Bureau employees, while SD-1 was collected among high-school students. Drawing sensible conclusions from learning experiments requires that the result be independent of the choice of training set and test among the complete set of samples. Therefore, it was necessary to build a new database by mixing NIST's datasets.

The MNIST training set is composed of 30,000 patterns from SD-3 and 30,000 patterns from SD-1. Our test set was composed of 5,000 patterns from SD-3 and 5,000 patterns from SD-1. The 60,000-pattern training set contained examples from approximately 250 writers. We made sure that the sets of writers of the training set and test set were disjoint.

SD-1 contains 58,527-digit images written by 500 different writers. In contrast to SD-3, where blocks of data from each writer appeared in sequence, the data in SD-1 is scrambled. Writer identities for SD-1 is available and we used this information to unscramble the writers. We then split SD-1 in two: characters written by the first 250 writers went into our new training set. The remaining 250 writers were placed in our test set. Thus, we had two sets with nearly 30,000 examples each. The new training set was completed with enough examples from SD-3, starting at pattern # 0, to make a full set of 60,000 training patterns. Similarly, the new test set was completed with SD-3 examples starting at pattern # 35,000 to make a full set with 60,000 test patterns. Only a subset of 10,000 test images (5,000 from SD-1 and 5,000 from SD-3) is available on this site. The full 60,000 sample training set is available.

Many methods have been tested with this training set and test set. Here are a few examples. Details about the methods are given in an upcoming paper. Some of those experiments used a version of the database where the input images where deskewed (by computing the principal axis of the shape that is closest to the vertical, and shifting the lines so as to make it vertical). In some other experiments, the training set was augmented with artificially distorted versions of the original training samples. The distortions are random combinations of shifts, scaling, skewing, and compression.

3.3 Data Pre-processing

Data preprocessing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model. For achieving better results from the applied model in Machine Learning projects the format of the data has to be in a proper manner. Some specified Machine Learning model needs information in a specified format, for example, Random Forest algorithm does not support null values, therefore to execute random forest algorithm null values have to be managed from the original raw data set.

Another aspect is that the data set should be formatted in such a way that more than one Machine Learning and Deep Learning algorithm are executed in one data set, and best out of them is chosen.

3.3.1 Splitting the Dataset

We need to split a dataset into train and test sets to evaluate how well our machine learning model performs. The train set is used to fit the model, the statistics of the train set are known. The second set is called the test data set, this set is solely used for predictions. The simplest method is to divide the whole dataset into two sets. Then use one for training and the other for model evaluation. This is called the holdout method. Training data and test data are two important concepts in machine learning. This chapter discusses them in detail.

Training Data

The observations in the training set form the experience that the algorithm uses to learn. In supervised learning problems, each observation consists of an observed output variable and one or more observed input variables.

Test Data

The test set is a set of observations used to evaluate the performance of the model using some performance metric. It is important that no observations from the training set are included in the test set. If the test set does contain examples from the training set, it will be difficult to assess whether the algorithm has learned to generalize from the training set or has simply memorized it.

A program that generalizes well will be able to effectively perform a task with new data. In contrast, a program that memorizes the training data by learning an overly complex model could predict the values of the response variable for the training set accurately, but will fail to predict the value of the response variable for new examples. Memorizing the training set is called over-fitting. A program that memorizes its observations may not perform its task well, as it could memorize relations and structures that are noise or coincidence. Balancing memorization and generalization, or over-fitting and under-fitting, is a problem common to many machine learning algorithms. Regularization may be applied to many models to reduce over-fitting.

In addition to the training and test data, a third set of observations, called a validation or hold-out set, is sometimes required. The validation set is used to tune variables called hyper parameters, which control how the model is learned. The program is still evaluated on the test set to provide an estimate of its performance in the real world; its performance on the validation set should not be used as an estimate of the model's real-world performance since the program has been tuned specifically to the validation data. It is common to partition a single set of supervised observations into training, validation, and test sets. There are no requirements for the sizes of the partitions, and they may vary according to the amount of data available. It is common to allocate 50 percent or more of the data to the training set, 25 percent to the test set, and the remainder to the validation set.

Some training sets may contain only a few hundred observations; others may include millions. Inexpensive storage, increased network connectivity, the ubiquity of sensor-packed smartphones, shifting attitudes towards privacy have contributed to the contemporary state of big data, or training sets with millions or billions of examples.

However, machine learning algorithms also follow the maxim "garbage in, garbage out." A student who studies for a test by reading a large, confusing textbook that contains many errors will likely not score better than a student who reads a short but well-written textbook. Similarly, an algorithm trained on a large collection of noisy, irrelevant, or incorrectly labeled data will not perform better than an algorithm trained on a smaller set of data that is more representative of problems in the real world.

Many supervised training sets are prepared manually, or by semi-automated processes. Creating a large collection of supervised data can be costly in some domains. Fortunately, several datasets are bundled with scikit-learn, allowing developers to focus on experimenting with models instead.

During development, and particularly when training data is scarce, a practice called cross-validation can be used to train and validate an algorithm on the same data. In cross-validation, the training data is partitioned. The algorithm is trained using all but one of the partitions, and tested on the remaining partition. The partitions are then rotated several times so that the algorithm is trained and evaluated on all of the data.

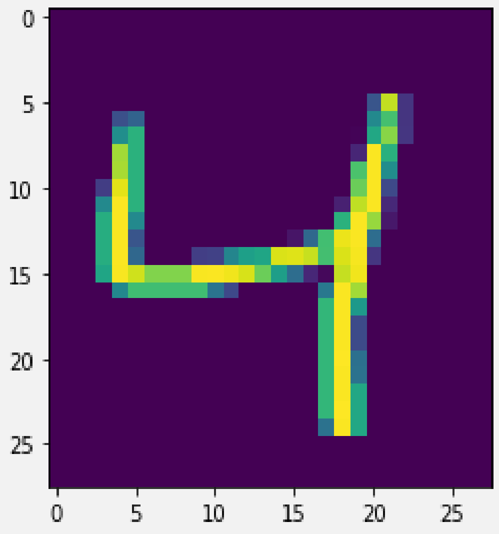
```
#Loading DataSet and spliting into Train and Test Data.
(X_train,y_train),(X_test,y_test) = keras.datasets.mnist.load_data()
#Shape of Dataset
print(X_train.shape,y_train.shape)
print(X_test.shape,y_test.shape)
(60000, 28, 28) (60000,)
(10000, 28, 28) (10000,)

#Shape of first Image
print(X_train[0].shape)
(28, 28)

#Output
print(y_train)
print(y_test)
[5 0 4 … 5 6 8]
[7 2 1 … 4 5 6]
```

3.3.2 Data Normalization

Normalization of data is transforming the data to appear on the same scale across all the records. The minimum value in the array will always be normalized to 0 and the maximum value in the array will be normalized to 1. All the other values will be in the range between 0 and 1. We need to normalize data when you're performing some sort of analysis on the dataset and that dataset has multiple variables measured using the different scales.

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications

```
#Showing Images using Matplotlib
import matplotlib.pyplot as plt
plt.imshow(X_train[2])
```



```
#Converting all values "0 to 1" for getting faster convergence
X_train = X_train/255
X_test = X_test/255
#output
X_train[0]
```

3.4 Creating Architecture of Neural Network

The Neural Network architecture is made of individual units called neurons that mimic the biological behaviour of the brain. Here are the various components of a neuron. Input - It is the set of features that are fed into the model for the learning process.

```
model = Sequential()
model.add(Flatten(input_shape=(28,28)))
model.add(Dense(128,activation='relu'))
model.add(Dense(32,activation='relu'))
model.add(Dense(10,activation='softmax'))
model.summary()
o/p:
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten (Flatten)           (None, 784)               0

 dense (Dense)               (None, 128)               100480

 dense_1 (Dense)             (None, 32)                4128

 dense_2 (Dense)             (None, 10)                330

=================================================================
Total params: 104,938
Trainable params: 104,938
Non-trainable params: 0
```

3.4.1 Loss Function

A loss function is a measure of how good your prediction model does in terms of being able to predict the expected outcome (or value). We convert the learning problem into an optimization problem, define a loss function and then optimize the algorithm to minimize the loss function.
Adaptive Moment Estimation is an algorithm for optimization technique for gradient descent. The method is really efficient when working with large problem involving a lot of data or parameters. It requires less memory and is efficient. Intuitively, it is a combination of the ‘gradient descent with momentum’ algorithm and the ‘RMSP’ algorithm.

```
model.compile(loss='sparse_categorical_crossentropy',optimizer
='Adam',metrics=['accuracy'])
```

3.5 Model Training

Training a Neural Network means finding the appropriate Weights of the Neural Connections thanks to a feedback loop called Gradient Backward propagation. It is the process of finding the values for the weights and biases. In most scenarios, training is accomplished using what can be described as a train-test technique. All Neurons of a given Layer are generating an Output, but they don’t have the same Weight for the next Neurons Layer. This means that if a Neuron on a layer observes a given pattern it might mean less for the overall picture and will be partially or completely muted. This is what we call Weighting: a big weight means that the Input is important and of course a small weight means that we should ignore it. Every Neural Connection between Neurons will have an associated Weight. Weights will be adjusted over the training to fit the objectives we have set (recognize that a dog is a dog and that a cat is a cat). In simple terms: Training a Neural Network means finding the appropriate Weights of the Neural Connections thanks to a feedback loop called Gradient Backward propagation.

```
history = model.fit(X_train,y_train,epochs=25,validation_split=0.2)
```

3.5.1 Model Evaluation

Model evaluation is the process of using different evaluation metrics to understand a machine learning model’s performance, as well as its strengths and weaknesses. Model evaluation is important to assess the efficacy of a model during initial research phases, and it also plays a role in model monitoring. It is the process through which we quantify the quality of a system’s predictions. To do this, we measure the newly trained model performance on a new and independent dataset. This model will compare labelled data with its own predictions.

To understand if model(s) is working well with new data, we leverage a number of evaluation metrics.

The most popular metrics for measuring classification performance include accuracy. Accuracy measures how often the classifier makes the correct predictions, as it is the ratio between the number of correct predictions and the total number of predictions.

```
y_prob = model.predict(X_test)
print(y_prob)
[[2.6913422e-22 2.2538927e-14 1.8345972e-11 ... 1.0000000e+00
  4.7084563e-19 8.1429470e-16]
 [1.3201358e-19 2.7432882e-11 1.0000000e+00 ... 2.1283006e-18
  1.2479351e-20 4.1847068e-26]
 [1.9679022e-14 1.0000000e+00 2.0697211e-10 ... 3.4013545e-09
  7.3647932e-09 6.1733365e-16]
 ...
 [4.5594750e-31 3.5771028e-19 3.1351612e-31 ... 9.6613289e-15
  2.2479675e-18 4.2843493e-09]
 [2.4044902e-16 2.9949770e-20 1.4221182e-20 ... 2.4439069e-20
  3.2201485e-12 1.8462015e-27]
 [1.6625082e-15 2.4978229e-17 4.9638642e-22 ... 4.8234238e-27
  2.6478930e-19 4.7145671e-34]] y_pred = y_prob.argmax(axis=1)

print(y_pred)
[7 2 1 ... 4 5 6]

from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred)
0.9755
```

3.6 Analysis on Accuracy

Machine learning model accuracy is one way to evaluate classification models. Essentially it is a metric that, based on the training data, uncovers which model best identifies relationships and patterns between variables within a dataset.
In other words, accuracy is the percentage of predictions each model got right – the number of accurate predictions divided by the total number of predictions.
Although model evaluation does not consider accuracy alone, it is the prime metric when comparing data models.
Due to the significant number of mathematical techniques involved in modelling and the uncertain nature of data, it is also essential to consider how the model will perform on a different dataset and how much flexibility it offers.
Models that are accurate and effective at generalizing unseen data are better at forecasting future events and therefore provide more value to your business.
Machine learning models helps to make practical business decisions. When we use models that produce more accurate outcomes, we get better information and can, therefore, make better decisions.
The cost of wrong information can be high, so mitigating that risk by optimizing model accuracy is essential.
Naturally, identifying the point where returns begin to diminish is also crucial so that you don’t spend excess time and money developing models that won’t move the needle very much.

3.6.1 Epoch

In a computing context, an epoch is the date and time relative to which a computer's clock and timestamp values are determined. The epoch traditionally corresponds to 0 hours, 0 minutes, and 0 seconds (00:00:00) Coordinated Universal Time (UTC) on a specific date, which varies from system to system

3.6.2 Backpropagation

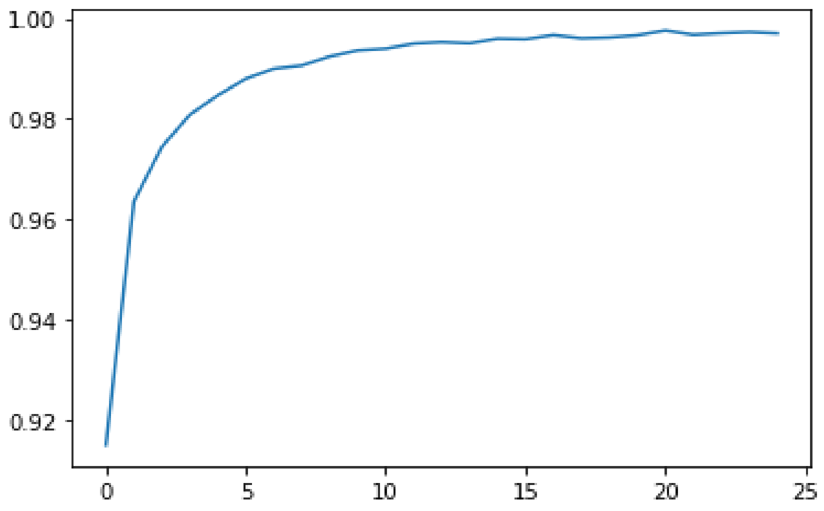
Backpropagation (backward propagation) is an important mathematical tool for improving the accuracy of predictions in data mining and machine learning. Essentially, backpropagation is an algorithm used to calculate derivatives quickly.

Artificial neural networks use backpropagation as a learning algorithm to compute a gradient descent with respect to weights. Desired outputs are compared to achieved system outputs, and then the systems are tuned by adjusting connection weights to narrow the difference between the two as much as possible. The algorithm gets its name because the weights are updated backwards, from output towards input.

The difficulty of understanding exactly how changing weights and biases affects the overall behavior of an artificial neural network was one factor that held back wider application of neural network applications, arguably until the early 2000s when computers provided the necessary insight. Today, backpropagation algorithms have practical applications in many areas of artificial intelligence (AI), including optical character recognition (OCR), natural language processing (NLP) and image processing.

Because backpropagation requires a known, desired output for each input value in order to calculate the loss function gradient, it is usually classified as a type of supervised machine learning. Along with classifiers such as Naïve Bayesian filters and decision trees, the backpropagation algorithm has emerged as an important part of machine learning applications that involve predictive analytics.
An important factor that is the basis of any Neural Network is the Optimizer, which is used to train the model. The most prominent optimizer on which almost every Machine Learning algorithm is built is the Gradient Descent. However, when it comes to building the Deep Learning models, the Gradient Descent has some major challenges.

```
plt.plot(history.history['accuracy'])
[<matplotlib.lines.Line2D at 0x7f130976f650>]
```

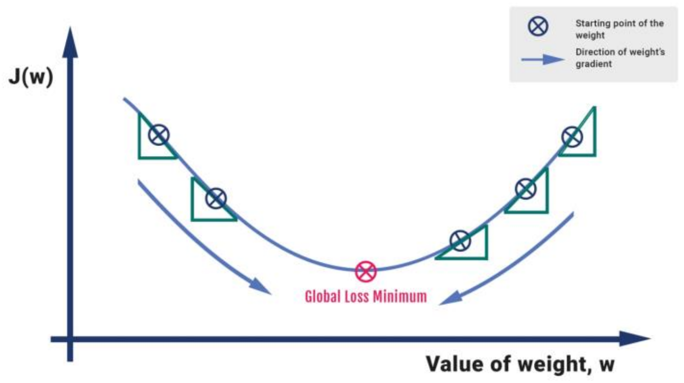


3.6.3 Gradient Descent

Gradient Descent is a process that occurs in the backpropagation phase where the goal is to continuously resample the gradient of the model's parameter in the opposite direction based on the weight w, updating consistently until we reach the global minimum of function J(w).

To put it simply, gradient descent is used to minimize the cost function, J(w)
An analogy could be drawn in the form of a steep mountain whose base touches the sea. We assume a person's goal is to reach down to sea level. Ideally, the person would have to take one step at a time to reach the goal. Each step has a gradient in the negative direction (Note: the value can be of different magnitude). The person continues hiking down till he reaches the bottom or to a threshold point, where there is no room to go further down.
Formalizing the analogy into an algorithmic form. We compute the activation for the incoming parameters, we carry out feedforward by taking the weighted sum of

the activation and its bias. We extract the error term of the output sample by subtracting it with the actual ‘target’ value.
The gradient descent process is exhibited in the form of the backpropagation step where we compute the error vectors δ backward, starting from the final layer.



IMG 04: GRADIENT DESCENT

Depending upon the activation function, we identify how much change is required by much change is required by taking the partial derivative of the function with respect to w. The change value gets multiplied by the learning rate. As part of the output, we subtract this value from the previous output to get the updated value. We continue this till we reach convergence.

3.6.4 Data Visualization.

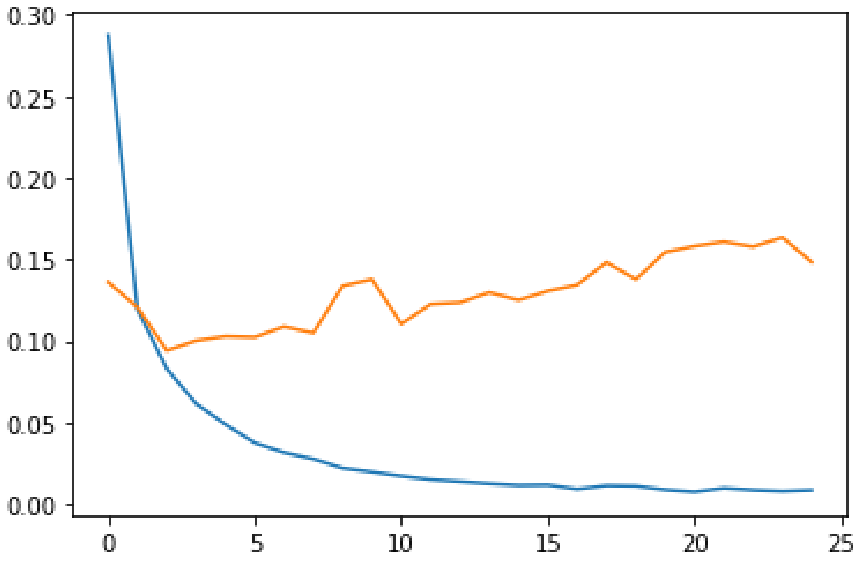
Visualizations are the easiest way to analyze and absorb information. Visuals help to easily understand the complex problem. They help in identifying patterns, relationships, and outliers in data. It helps in understanding business problems better and quickly. It helps to build a compelling story based on visuals. Insights gathered from the visuals help in building strategies for businesses. It is also a precursor to many high-level data analyses for Exploratory Data Analysis (EDA) and Machine Learning (ML).

Matplotlib is a 2-D plotting library that helps in visualizing figures. Matplotlib emulates MATLAB like graphs and visualizations. MATLAB is not free, is difficult to scale and as a programming language is tedious. So, matplotlib in Python is used as it is a robust, free and easy library for data visualization.

matplotlib. pyplot. plot (**args*, *scalex=True*, *scaley=True*, *data=None*, ***kwargs)*

The pyplot API has a convenient MATLAB-style stateful interface. In fact, matplotlib was originally written as an open-source alternative for MATLAB. The OO API and its interface is more customizable and powerful than pyplot, but considered more difficult to use.

```
#ploting loss,val_loss,acuracy using Matlpotlib
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
[<matplotlib.lines.Line2D at 0x7f13097e8dd0>]
```

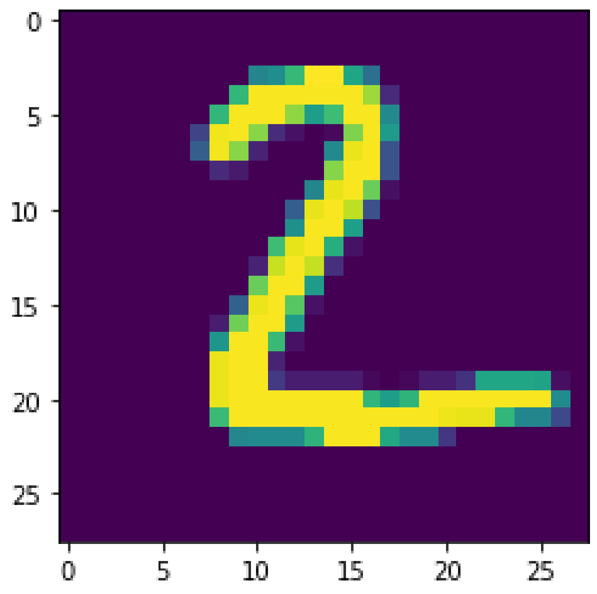


3.7 Data Prediction

“Prediction” refers to the output of an algorithm after it has been trained on a historical dataset and applied to new data when forecasting the likelihood of a particular outcome. Predictive analytics involves certain manipulations on data from existing data sets with the goal of identifying some new trends and patterns. These trends and patterns are then used to predict future outcomes and trends. By performing predictive analysis, we can predict future trends and performance. It is also defined as the prognostic analysis; the word prognostic means prediction. Predictive analytics uses the data, statistical algorithms and machine learning techniques to identify the probability of future outcomes based on historical data. In predictive analysis, we use historical data to predict future outcomes. Thus, predictive analysis plays a vital role in various fields. It improves decision making and helps to increase the profit rates of business and reduces risk by identifying them at the early stage.

3.7.1 Prediction on Test Data

```
#Predicting on test data
plt.imshow(X_test[1])
<matplotlib.image.AxesImage at 0x7f13096e2750>
```

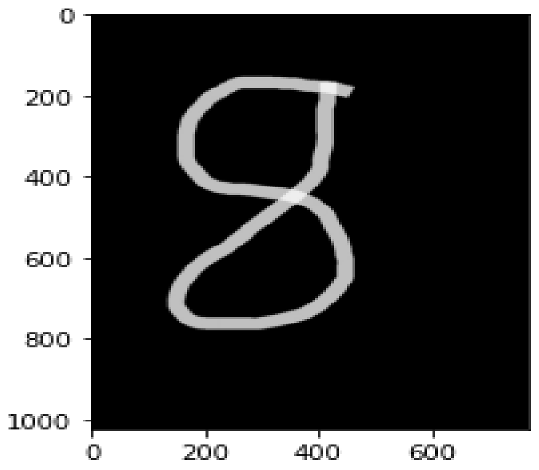


```
#output
model.predict(X_test[1].reshape(1,28,28)).argmax(axis=1)
array([2])
```

3.7.2 Testing on Custom Dataset

Many a time, we will have to classify images of a given custom dataset. To be precise, in the case of a custom dataset, the images of our dataset are neatly organized in folders. It can either be collected manually or downloaded directly from common sites for datasets such as Kaggle. When we use Tensorflow or Keras datasets, we easily obtain the values of x_train, y_train, x_test, and y_test while loading the dataset itself. These values are significant for understanding how our training and validation dataset labels are encoded and obtain the classification report and the confusion matrix

```
import cv2
img1=cv2.imread("/content/TestImage2.jpg")
plt.imshow(img1)
<matplotlib.image.AxesImage at 0x7f12fa820690>
```



```
print(img1.shape)
(1024, 769, 3)

gray=cv2.cvtColor(img1,cv2.COLOR_BGR2GRAY)
print(gray.shape)
(1024, 769)

resized=cv2.resize(gray,(28,28),interpolation=cv2.INTER_AREA)
print(resized.shape)
(28, 28)

#Normalizing pixel values
newimg=tensorflow.keras.utils.normalize(resized,axis=1)
#output
print(model.predict(newimg.reshape(1,28,28)).argmax(axis=1))
[8]
```

Chapter: 04
Results and Discussions

We trained a model which can recognize a hand written digit. After several attempts we were able to reach an accuracy score of 97%. We trained this model on MNIST dataset and Artificial Neural Network Model. The MNIST database of handwritten digits, has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and cantered in a fixed-size image. In this project, we have successfully built a Python deep learning project on handwritten digit recognition app. We have built and trained the Artificial neural network which is very effective for image recognition purposes. Artificial neural networks (ANNs) are biologically inspired computational networks. Among the various types of ANNs, in this chapter, we focus on *multilayer perceptions* (MLPs) with backpropagation learning algorithms. MLPs, the ANNs most commonly used for a wide variety of problems, are based on a supervised procedure and comprise three layers: input, hidden, and output.

Chapter: 05

Conclusions and Future Scope

Handwritten Digit Recognition is a very broad field concerned with turning an image or a scanned document containing a set of digits into an array of numbers/digits. In this project, we have attempted to build a recognizer for handwritten digits using the MNIST dataset. The challenge of this project was to be able to come up with some basic image correlation techniques, instead of some sophisticated algorithms, and see to what extent we can make this mechanism accurate. We have tried several versions and kept trying to improve each one in order to reach a higher performance rate. The last version has reached
a rate of 97% accuracy. Unfortunately, we could not compare the performance of the mechanism we have built to some others that have already been designed and/or implemented before because we did not find any academic paper that tackles this method. The performance we have reached is far less than that of machine learning, which reaches a performance rate of 99.3%; however, it could be further improved and made into a better one. The goal of this project was to explore the field of OCR and try to come up with some techniques that could
be used without going into deep computations, and even if the final result is not very reliable, it still provides an accuracy way better than random.

The future steps that to go for would be having a closer look at the results of all the versions in order to find new rules. By extracting and implementing them, we will be able to enhance the performance of these versions. Moreover, it would be good if we could make some modifications to both the reference set and the rules in order to make our program more general and able to identify both typed and handwritten digits. Furthermore, in the future, we could make a great use of the matrices that indicate the first maximum overlap of each test image with the reference images, along with the number of pixels left out from both. These matrices could be used with some clustering algorithms to build a program able to recognize handwritten digits with a very high efficiency. Last but not least, we thought about using linear or high-level regression in the versions we have developed in order to create more rules. As regression could be used for binary classification and is not very suitable to classify a digit out of ten, this technique could be used in order to tell which digit is the most suitable, the first maximum or second maximum, which will enable us to generate more rules; thus, reach a higher efficiency.

Building a good ANN model with optimized parameters takes time. It depends on the number of training records and iterations. There are no consistent guidelines on the number of hidden layers and nodes within each hidden layer. Hence, one would need to try out many parameters to optimize the selection of parameters. However, once a model is built, it is straightforward to implement, and a new unseen record gets classified quite fast.

5.1 References

1. Volume 4, Issue 6, June – 2019 International Journal of Innovative Science and Research Technology ISSN No:-2456-2165 , Vijayalaxmi R Rudraswamimath, Bhavanishankar K Computer Science and Engineering, RNS Institute of Technology, Channasandra, Bangalore, India Assistant Professor, Computer Science and Engineering, RNS Institute of Technology, Bangalore

2. Abdulllah, M., Agal, A., Alharthi, M., & Alrashidi, M. (2018). Retracted:Arabic handwriting recognition using neural network classifier. Journal of Fundamental and Applied Sciences, 10(4S), 265-270.

3. Sharma, A., & Chaudhary, D. R. (2013). Character recognition using neural network. International journal of engineering Trends and Technology (IJETT), 4(4), 662-667.

4. Kumar, P., Saini, R., Roy, P. P., & Pal, U. (2018). A lexicon-free approach for 3D handwriting recognition using classifier combination. Pattern Recognition Letters, 103, 1-7.

5. Dwivedi, U., Rajput, P., Sharma, M. K., & Noida, G. (2017). Cursive Handwriting Recognition System Using Feature Extraction and Artificial Neural Network. Int. Res. J. Eng. Technol, 4(03), 2202-2206.

6. Larasati, R., & KeungLam, H. (2017, November). Handwritten digits recognition using ensemble neural networks and ensemble decision tree. In 2017 International Conference on Smart Cities, Automation & Intelligent Computing Systems (ICON-SONICS) (pp. 99-104). IEEE.

7. Vijay Kotu, Bala Deshpande, in Data Science (Second Edition), 2019

8. Nair, R. R., Sankaran, N., Kota, B. U., Tulyakov, S., Setlur, S., & Govindaraju, V. (2018, April). Knowledge transfer using Neural network-based approach for handwritten text recognition. In 2018 13th IAPR International Workshop on Document Analysis Systems (DAS) (pp. 441-446). IEEE.