Unveiling the Comparative Insight of Bangla Handwritten Digit Recognition using Machine Learning and Deep Learning Models

by

Md. Arif
ID: CSE2003001020
M
Md. Taohid Alam
ID: CSE2003001006

Rokibul Islam Shakil
ID: CSE1903018149

Jannatul Ferdus
ID: CSE1803015078

Supervised by
Arifur Rahaman

Submitted in partial fulfillment of the requirements for the degree of
Bachelor of Science in Computer Science and Engineering



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SONARGAON UNIVERSITY (SU)

May 2024

Unveiling the Comparative Insight of Bangla Handwritten Digit Recognition using Machine Learning and Deep Learning Models

by

MD. Arif
ID: CSE2003021020

MD. Taohid Alam
ID: CSE2003021006

Rokibul Islam Shakil
ID: CSE1903018149

Jannatul Ferdus
ID: CSE1803015078

Supervised by
Arifur Rahaman

Submitted in partial fulfillment of the requirements for the degree of
Bachelor of Science in Computer Science and Engineering



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SONARGAON UNIVERSITY (SU)

May 2024

APPROVAL

The thesis titled "**Unveiling the Comparative Insight of Bangla Handwritten Digit Recognition using Machine Learning and Deep Learning Models**" submitted by MD. Arif (CSE2003021020), MD. Taohid Alam (CSE2003021006), Rokibul Islam Shakil (CSE1903018149) and Jannatul Ferdus (CSE1803015078) to the Department of Computer Science and Engineering, Sonargaon University (SU), has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Engineering and approved as to its style and contents.

Board of Examiners

Arifur Rahaman
Assistant Professor,
Department of Computer Science and Engineering
Sonargaon University (SU)

Supervisor

(Examiner Name and Signature)
Department of Computer Science and Engineering
Sonargaon University (SU)

Examiner 1

(Examiner Name and Signature)
Department of Computer Science and Engineering
Sonargaon University (SU)

Examiner 2

(Examiner Name and Signature)
Department of Computer Science and Engineering
Sonargaon University (SU)

Examiner 3

DECLARATION

We, hereby, declare that the work presented in this report is the outcome of the investigation performed by us under the supervision of **Arifur Rahaman, Assistant Professor,** Department of Computer Science and Engineering, Sonargaon University, Dhaka, Bangladesh. We reaffirm that no part of this thesis has been or is being submitted elsewhere for the award of any degree or diploma.

Countersigned

(Arifur Rahaman)
Supervisor

Signature

MD. Arif
ID: CSE2003021020

MD. Taohid Alam
ID: CSE2003021006

Rokibul Islam Shakil
ID: CSE1903018149

Jannatul Ferdus
ID: CSE1803015078

ABSTRACT

In the realm of Artificial Intelligence, despite significant advancement in the computer vision and pattern recognition field, recognition of Bangla handwritten digit recognition has still been undergoing some unique challenges, including the distortion in the images, the variation in writing styles, the different shapes and sizes of digits, and the varying levels of noise. There is a further room for more improvement. In this paper, we not only have delved into the comparative analysis of several machine learning and deep learning models to portray the macro and micro level insights, but also have achieved more accuracy in some models. If we can merge this machine learning and deep learning with OCR, which is largely getting attention in these days, it could be a huge contribution to this field. We have used the Ekush Bangla handwritten numeral digit dataset of size 30687. Four machine learning algorithms, Artificial Neural Network and three architectures of CNN namely ResNet, Xception, LeNet have been used to depict the analysis. We have explored the pattern matching approach and achieved 91% accuracy for the Support Vector Machine, 80.44% accuracy for Decision Tree, 91.22% for Random Forest & 90.38% for KNN. And in the Deep Learning approach, through the Convolutional Neural Network algorithm's architectures we have achieved 99.48%, 98.63% and 97.23% respectively in the Xception, ResNet, and LeNet. Finally in ANN we have got accuracy of 90.50%.

ACKNOWLEDGMENT

At the very beginning, we would like to express my deepest gratitude to the Almighty Allah for giving us the ability and the strength to finish the task successfully within the schedule time.

We are auspicious that we had the kind association as well as supervision of **Arifur Rahaman**, Assistant Professor, Department of Computer Science and Engineering, Sonargaon University whose hearted and valuable support with best concern and direction acted as necessary recourse to carry out our project.

We are also thankful to all our teachers during our whole education, for exposing us to the beauty of learning.

Finally, our deepest gratitude and love to my parents for their support, encouragement, and endless love.

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
ANN	Artificial Neural Network
CNN	Convolutional Neural Networks
DL	Deep Learning
DT	Decision Tree
FN	False Negative
FP	False Positive
KNN	K Nearest Neighbor
LBP	Local Binary Pattern
ML	Machine Learning
ReLU	Rectified Linear Unit
RF	Random Forest
RNN	Recurrent Neural Network
SVM	Support Vector Machine
TN	True Negative
TP	True Positive

TABLE OF CONTENTS

Title			Page No.
DECLARATION …….……..			iii
ABSTRACT ……...…….…….			iv
ACKNOWLEDGEMENT …….…….……			v
LIST OF ABBREVIATION …….…….……			vi
CHAPTER 1			1 – 4
INTRODUCTION TO BANGLA HAND WRITTEN DIGIT RECOGNITION			
1.1	Introduction ……..…….....		1
1.2	Objectives ……...……...		1 – 2
1.3	Machine Learning & Deep Learning……..……		2 – 3
	1.3.1	Machine Learning…….…..……...	2 – 3
	1.3.2	Deep Learning…….…….…..……...	3
1.4	Why Machine Learning and Deep Learning? …….…….....		3 – 4
1.5	Organization of the thesis book …….……		4
CHAPTER 2			5 – 6
LITERATURE REVIEW			
2.1	Introduction …….….....		5
2.2	Case Study and Overviews of Related Works……		5 – 6
CHAPTER 3			7 – 13
CLASSIFICATION ALGORITHMS FOR DIGIT RECOGNITION			
3.1	Introduction…….……...		7
3.2	Different Types of Classification……..…..……		7 – 13
	3.2.1	Support Vector Machine……...…..……...	8 – 9
	3.2.2	Decision Tree…….…..……...……...	9 – 10
	3.2.3	Random Forest……..……	11 – 12
CHAPTER 4			14 – 15
ARTIFICIAL NEURAL NETWORK IN DIGIT RECOGNITION			
4.1	Introduction …….…….		14
4.2	How Neural Networks Work…….		14 – 15
4.3	Computational Complexities in Neural Networks……..……		15

CHAPTER 5 16 – 18

CONVOLUTIONAL NEURAL NETWORK IN DIGIT RECOGNITION

5.1 Introduction ……. 16

5.2 Real-Life Applications of CNN……. 16 – 17

5.3 Different Architectures of CNN……. 17 – 18

5.3.1 LeNet5……. 17

5.3.2 ResNet……. 18

5.3.3 Xception……... 18

CHAPTER 6 19 – 24

METHODOLOGY

6.1 Introduction…… 19

6.2 Data Collection ……..…….….. 19

6.3 Data Preprocessing…….…… 20

6.4 Model Training……..…… 20 – 24

6.4.1 Machine Learning Model…….…… 20

6.4.2 Deep Learning Model……...…… 20 – 24

6.4.2.1 Artificial Neural Network……. 20 – 21

6.4.2.2 Xception……...…… 21 – 22

CHAPTER 7 25 – 32

FINDINGS AND RESULTS

7.1 Introduction…… 25 – 26

7.2 Analysis of Result……..……..….. 26 – 33

7.2.1 Evaluation Report Based On Precision, Recall, F1 Score……. 26 – 27

7.2.2 Evaluation Report Based On Training Metrics …….…… 28

7.2.3 Evaluation Report Based On Confusion Matrix……... 29 – 30

CHAPTER 8 33

CONCLUSION AND FUTURE WORKS

8.1 Conclusion…….…… 33

8.2 Limitations……. 33

8.3 Future Works ……..……...…..…….. 33

REFERENCES ……..…….. 34 – 35

LIST OF TABLES

Table No.	Title	Page No.
Table 6.1	Xception Architecture Summary	22
Table 6.2	ResNet Architecture Summary	23
Table 6.3	LeNet Architecture Summary	24
Table 7.1	Accuracy of Different Models	31
Table 7.2	Results of Ekush Dataset on different existing works	31
Table 7.3	Result Comparison of different Models	32

LIST OF FIGURES

Figure No.	Title	Page No.
Fig.1.1	Bangla digits with their corresponding English digits	1
Fig.1.2	Types of Machine Learning Algorithms	3
Fig.3.1	Generalized example of Classification Algorithm	7
Fig.3.2	Support Vector Machine	8
Fig.3.3	Decision Tree	10
Fig.5.1	Schematic Diagram of basic CNN	16
Fig.6.1	Sample Bangla Digits	19
Fig.6.2	Architecture of the ANN	21
Fig.6.3	Xception Model Architecture	22
Fig.6.4	ResNet Model Architecture	23
Fig.7.1	Evaluation report of Xception Model	27
Fig.7.2	Evaluation report of ResNet Model	27
Fig.7.3	Evaluation report of LeNet Model	27
Fig.7.4	Model Training Metrics Over Epochs of Xception	29
Fig.7.5	Model Training Metrics Over Epochs of Resnet	29
Fig.7.6	Confusion Matrix	30
Fig.7.7	Confusion Matrix of the Models	31

CHAPTER 1

INTRODUCTION TO HAND WRITTEN DIGIT RECOGNITION

1.1 Introduction

In the contemporary era, Handwritten Digit Recognition stands as a pivotal task with multifaceted applications ranging from financial transactions to educational support. Machine Learning (ML) has emerged as a cornerstone technology across diverse domains such as computer vision, medical imaging, and beyond. Despite significant strides in recognizing printed characters, handwritten digit recognition remains a challenging frontier, particularly for Bangla Language.

This thesis embarks on a comprehensive exploration of Bengali Handwritten Digit Recognition, aiming to address existing limitations and push the boundaries of accuracy and performance. By delving into the intricacies of Bengali script and leveraging advanced ML and Deep Learning techniques, our research endeavors to bridge the gap between handwritten digit recognition in Bengali and its counterparts in other languages.

Lots of papers on Digit recognition have been published as of now, they added very impactful value to the field. But when it comes to Bangla handwritten digit recognition works can be found less in number compared to English digit recognition. Our research endeavors to advance the field of Bengali Handwritten Digit Recognition by presenting a novel approach that surpasses existing models in accuracy and performance on the Ekush[1] numerical dataset. Through rigorous experimentation and analysis, we have conducted comprehensive comparisons with state-of-the-art Machine Learning (ML) and Deep Learning (DL) models. This comparative analysis serves as a testament to the effectiveness of our approach and underscores its potential to significantly enhance the accuracy and reliability of Bengali Handwritten Digit Recognition systems, thereby fostering inclusivity and accessibility in the digital realm.

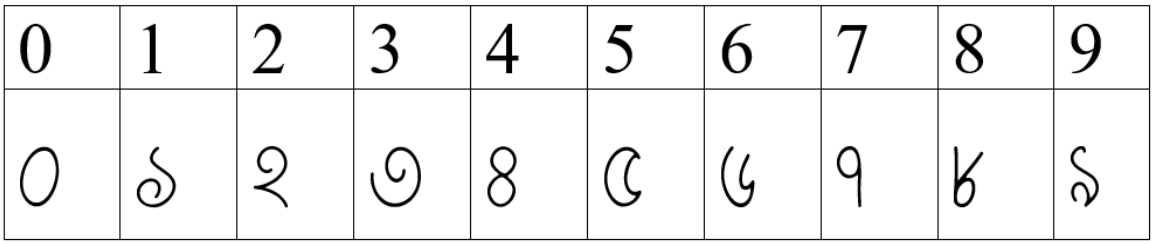


Fig. 1.1: Bangla digits with their corresponding English digits

1.2 Objectives

The primary objective of this thesis is to conduct a comprehensive study aimed at advancing the field of Bengali Handwritten Digit Recognition through the exploration and evaluation of various Machine Learning (ML) and Deep Learning (DL) models including but not limited to convolutional neural networks (CNNs), in accurately recognizing Bengali handwritten digits. By systematically comparing the performance of these models, the objective is to identify the most effective approach for Bengali handwritten digit

recognition. In addition to that, exploring the automated feature extraction techniques within the context of Bengali handwritten digit recognition to determine their impact on model performance and identify the most suitable techniques for this task.

Overall, the thesis seeks to contribute to the advancement of Bengali Handwritten Digit Recognition by offering comparative analysis of the strengths and limitations of different ML and DL approaches, as well as providing actionable recommendations for improving recognition accuracy and robustness.

1.3 Machine Learning and Deep Learning

1.3.1 Machine Learning

Machine learning is a subset of artificial intelligence (AI) that focuses on developing algorithms and techniques that enable computers to learn from data and improve their performance over time without being explicitly programmed.

Machine learning, the broader discipline, encompasses a variety of algorithms and techniques aimed at extracting insights from data. At its core, machine learning revolves around the concept of learning from experience, where models iteratively improve their performance on a specific task by analyzing and adapting to data. From classical methods like linear regression and decision trees to more sophisticated approaches such as support vector machines and random forests, machine learning provides a versatile toolkit for tackling a wide array of problems.

Machine learning can be broadly categorized into three main types based on the nature of the learning process and the availability of labeled data:

1. **Supervised Learning:** In supervised learning, the algorithm learns from labelled data, where each input is associated with a corresponding target output. The goal is to learn a mapping function from input variables to output variables, such that the model can accurately predict the output for new, unseen data. Supervised learning tasks include classification, where the output variable is categorical, and regression, where the output variable is continuous. Common algorithms in supervised learning include decision trees, logistic regression, support vector machines, and neural networks.
2. **Unsupervised Learning:** Unsupervised learning involves learning patterns and relationships from unlabelled data, where the algorithm is not provided with explicit output labels. Instead, the algorithm seeks to uncover hidden structures within the data, such as clusters, associations, or latent variables. Unsupervised learning tasks include clustering, where the goal is to partition data points into distinct groups based on similarity, and dimensionality reduction, where the aim is to reduce the number of input features while preserving important information. Common algorithms in unsupervised learning include k-means clustering, hierarchical clustering, principal component analysis (PCA), and autoencoders.
3. **Reinforcement Learning:** Reinforcement learning is a type of machine learning where an agent learns to make sequential decisions by interacting with an environment. The agent receives feedback in the form of rewards or penalties based on its actions, and the objective is to learn a policy that maximizes cumulative

rewards over time. Reinforcement learning is well-suited for tasks that involve decision-making and sequential interactions, such as game playing, robotics, and autonomous navigation. Common algorithms in reinforcement learning include Q-learning, deep Q-networks (DQN), policy gradients, and actor-critic methods.

These three types of machine learning encompass a wide range of algorithms and techniques, each suited to different types of data, learning objectives, and application domains. Depending on the specific problem at hand and the available data, practitioners may choose to employ one or a combination of these types to build effective machine learning solutions.

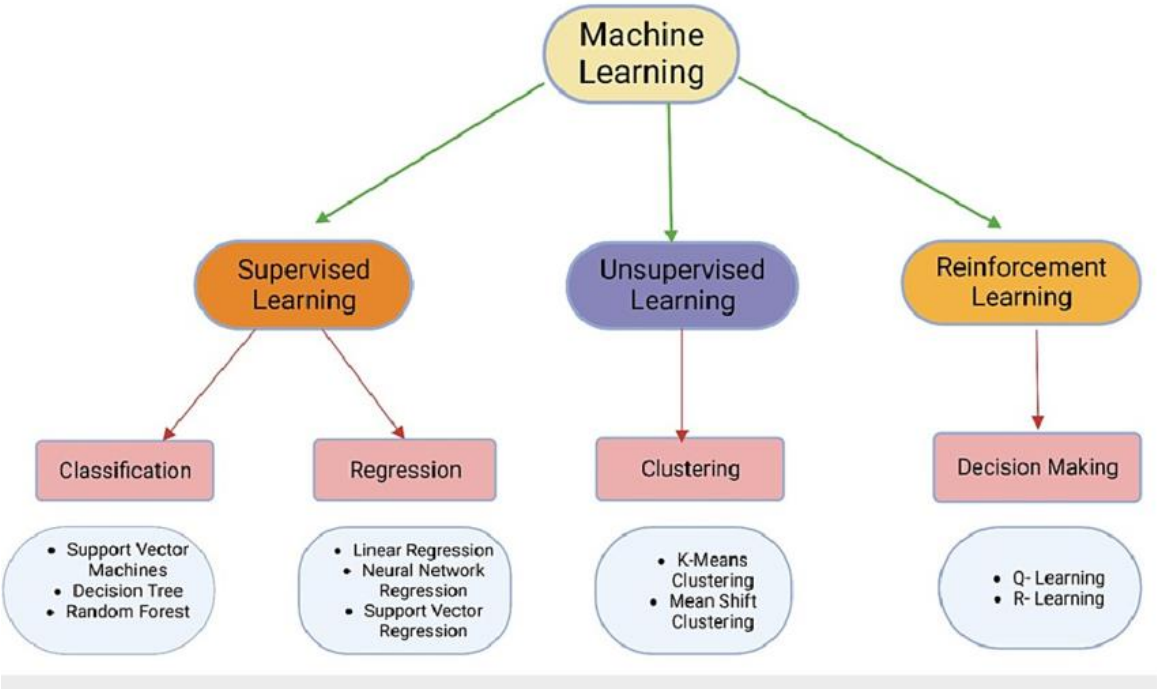


Fig. 1.2: Types of Machine Learning Algorithms

1.3.2 Deep Learning

Deep learning, a subset of machine learning, has garnered significant attention and acclaim in recent years, particularly for its unparalleled success in handling complex data like images, audio, and text. At the heart of deep learning are artificial neural networks, inspired by the structure and function of the human brain. These networks consist of interconnected layers of neurons that process information hierarchically, enabling the automatic extraction of intricate features from raw data. Convolutional Neural Networks (CNNs) for image recognition, Recurrent Neural Networks (RNNs) for sequential data, and Transformer models for natural language processing exemplify the power and versatility of deep learning architectures.

1.4 Why Machine Learning and Deep Learning

Using machine learning and deep learning models for handwritten digit recognition offers several compelling advantages. These are discussed below:

High Accuracy: Deep learning models, particularly Convolutional Neural Networks (CNNs), have demonstrated exceptional accuracy in recognizing handwritten digits. These

models can learn complex patterns and features from data, leading to superior performance compared to traditional machine learning algorithms.

Automatic Feature Extraction: Unlike traditional machine learning approaches that rely on handcrafted features, deep learning models automatically learn hierarchical representations of data. This means that features relevant to digit recognition are extracted directly from the raw input images, eliminating the need for manual feature engineering and potentially improving accuracy.

Scalability: Deep learning models are highly scalable and can handle large datasets with millions of handwritten digit samples. As the amount of data increases, deep learning models can continue to improve their performance, making them well-suited for tasks like digit recognition where large datasets are available.

Flexibility: Deep learning models can be adapted to various digit recognition tasks with minimal changes to the underlying architecture. Whether it's recognizing digits written in different styles or languages, deep learning models can be trained to generalize across different variations, making them versatile and adaptable.

Continuous Improvement: Deep learning models can be trained iteratively, allowing them to continuously improve their performance over time. As more data becomes available or as the model encounters new variations in handwriting, it can be retrained to enhance its accuracy and robustness.

Real-Time Recognition: Deep learning models can be optimized for real-time inference, enabling fast and efficient recognition of handwritten digits. This is particularly useful in applications where speed is crucial, such as postal sorting or banking systems processing checks.

Reduced Manual Effort: By automating the digit recognition process, machine learning and deep learning models can significantly reduce the need for manual intervention. This saves time and resources, allowing organizations to streamline their operations and focus on higher-value tasks.

1.5 Organization of the Thesis Book

This thesis consists of 8 chapters ranging from Chapter 1 to Chapter 8. Chapter 1 gives an overview of the topic. It additionally comprises of Problem Statements, Objectives. In the meantime, Chapter 2 reviews the past research works that was led by others explores. All the significant specialized paper, diaries and books taken from those inquiries about will be talked about in detail. Chapter 3 deals with several classification algorithms. Chapter 4 unfolds Artificial Neural Networks and the computational difficulties. In chapter 5 Convolutional Neural Network has been discussed along with its three architectures. The methodology of our thesis has been depicted in chapter 6. In chapter 7 Findings and Results have been uncovered respectfully. Finally in chapter 8 the thesis has been concluded and future scope has been portrayed.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

In this chapter, we embark on a journey through the extensive research landscape of handwritten digit recognition, focusing particularly on Bangla digit. Handwritten digit recognition holds immense significance across various fields, from optical character recognition to document analysis. Our exploration encompasses traditional machine learning methods, deep learning advancements to depict the comparative analysis.

Despite being one of the most spoken languages globally, research on Bangla handwritten digit recognition is relatively limited. We delve into the challenges unique to Bangla digit, considering its linguistic complexity. By synthesizing existing studies, we aim to show performance comparison, identify trends, gaps, and opportunities for further research in Bangla handwritten character recognition.

2.2 Case Study and Overviews of Related Works

Though there has been performed limited research on this Bangla digit recognition, yet some impactful studies on Bangla Handwritten Digit Recognition have been conducted as of now. Md. Ferdous Wahi et al, "A Classical Approach to Handcrafted Feature Extraction Techniques for Bangla Handwritten Digit Recognition" [2]. In that paper they benchmarked four rigorous classifiers to recognize Bangla Handwritten Digit: K-Nearest Neighbor (KNN), Support Vector Machine (SVM), Random Forest (RF), and Gradient-Boosted Decision Trees (GBDT) based on three handcrafted feature extraction techniques: Histogram of Oriented Gradients (HOG), Local Binary Pattern (LBP), and Gabor filter on four publicly available Bangla handwriting digits datasets: NumtaDB, CMARTdb, Ekush and BDRW. Their proposed model achieved 95.68% for the Ekush dataset. In [3] they went for CNN based approach. The deep convolutional neural network model has shown an excellent performance, securing the 13th position with 92.72% testing accuracy in the Bengali handwritten digit recognition challenge 2018 among 57 participating teams. They performed a study of the network performance on the MNIST and EMNIST datasets in order to bolster the analysis. A quad tree-based feature set was used in [4] for the recognition of handwritten Bangla numeral dataset. On experimentation with the database of 4200 image samples using Support Vector Machine (SVM), the technique yields an average recognition rate of 93.338% evaluated after three-fold cross validation of results. Indronil [5] used Ekush dataset and by using CNN he achieved 98.3% accuracy in Bangla Handwritten Digit Recognition. In [6] they proposed a modified ResNet-18 architecture (a convolutional neural network architecture) in recognizing Bangla handwritten characters. The proposed method was applied to two recently created isolated Bangla handwritten datasets. The used datasets were relatively large and practical to apply deep learning. Using the proposed method, they achieved the state-of-the-art recognition performance of 95.10% accuracy. They employed CMATERdb and BanglaLekha-Isolated as datasets. Wen[7] proposed a Bangla numeric digit recognition system using principal component analysis (PCA) and Support Vector Machine (SVM). One is the image reconstruction recognition approach, and the other is the direction feature extraction approach combined with PCA and SVM. By examining the handwritten Bangla numeral data captured from real

Bangladesh letters, their experimental results showed that their proposed approaches were effective on that time. To meet performance requirements of automatic letter sorting machine, they integrated the results of the two proposed approaches with one conventional PCA approach. It had been found that the recognition result achieved by the integrated system is more reliable than that by one method alone. The average recognition rate, error rate and reliability achieved by the integrated system were 95.05%, 0.93% and 99.03%, respectively. In [8] they exploited local binary pattern (LBP) for handwritten Bangla numeral recognition. They classified Bangla digits from their LBP histograms using K Nearest Neighbors (KNN) classifier. The performance of three different variations of LBP-the basic LBP, the uniform LBP and the simplified LBP was investigated. The proposed OCR system was evaluated on the off-line handwritten Bangla numeral database CMATERdb 3.1.1, and achieved an excellent accuracy of 96.7%-character recognition rate.

CHAPTER 3

CLASSIFICATION ALGORITHMS FOR DIGIT RECOGNITION

3.1 Introduction

Nowadays, many industries have been dealing with very large data sets of different types. Manually processing all that information can be time-consuming and might not even add value in the long term. Many strategies, from simple automation to machine learning techniques, are being applied for a better return on investment.

Classification is a supervised machine learning method where the model tries to predict the correct label of a given input data. In classification, the model is fully trained using the training data, and then it is evaluated on test data before being used to perform prediction on new unseen data. For instance, an algorithm can learn to predict whether a given email is spam or ham (no spam).

In short, classification is a form of "pattern recognition," with classification algorithms applied to the training data to find the same pattern (similar words or sentiments, number sequences, etc.) in future sets of data.

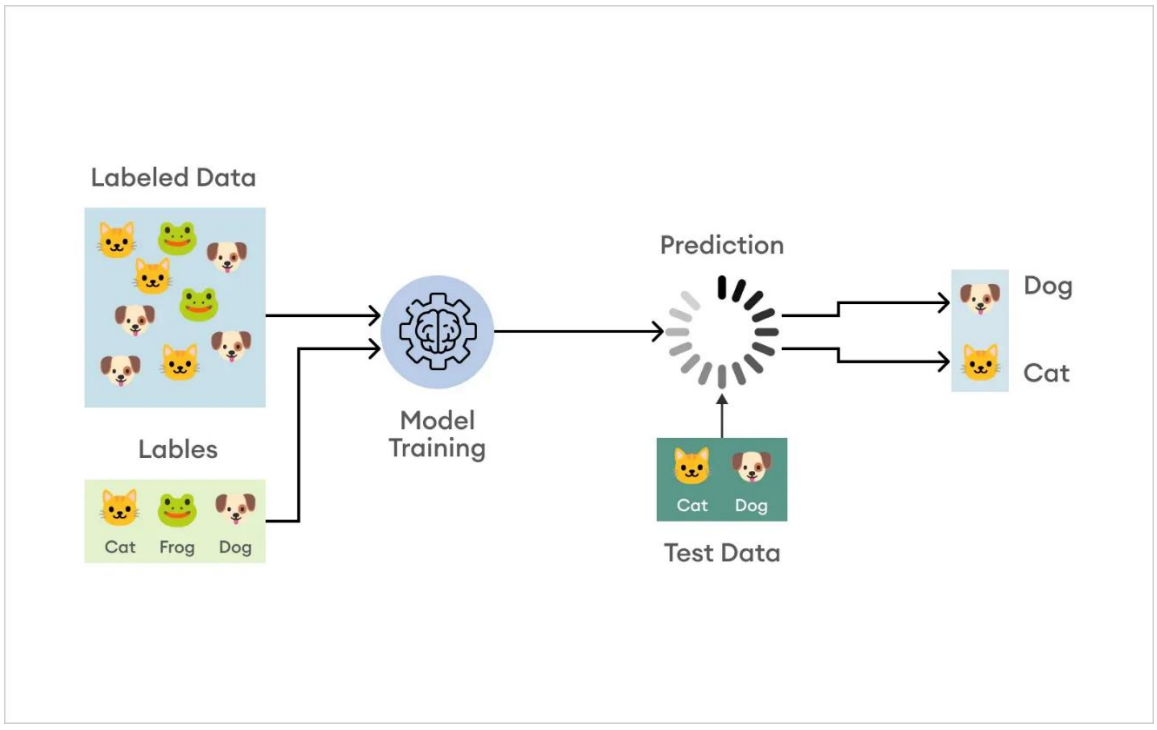


Fig. 3.1: Generalized example of Classification Algorithm

3.2 Different types of Classification Algorithms

The study of classification in statistics is vast, and there are several types of classification algorithms we can use depending on the dataset we're working with. In this paper we have explored the following algorithms.

3.2.1 Support Vector Machine

Support Vector Machine (SVM) represents a powerful class of supervised learning algorithms extensively utilized in various domains such as classification, regression, and outlier detection. This subsection aims to provide a comprehensive overview of SVM, highlighting its theoretical underpinnings, algorithmic aspects, and practical applications.

Theoretical Foundations:

At its core, SVM seeks to construct an optimal hyperplane in a high-dimensional space that maximally separates different classes of data points. This hyperplane is determined by support vectors, which are data points lying closest to the decision boundary. SVM operates under the principle of maximizing the margin, i.e., the distance between the hyperplane and the support vectors, thereby enhancing its generalization performance.

Mathematically, the decision function for a linear SVM can be defined as:

$$\mathbf{f(x) = w \cdot x + b}$$

where w represents the weight vector, x denotes the input feature vector, and b is the bias term.

The margin γ of the hyperplane is given by the formula:

$$\boldsymbol{\gamma = \frac{2}{||w||}}$$

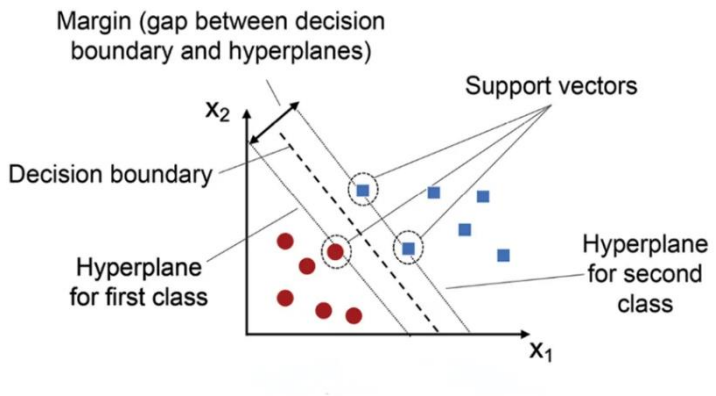


Fig. 3.2: Support Vector Machine

Application of SVM

a. Text Classification: SVM is extensively utilized in natural language processing tasks such as text classification, sentiment analysis, and spam detection. In sentiment analysis, SVM models can classify text data into positive, negative, or neutral sentiments based on the content. Similarly, in spam detection, SVM classifiers can distinguish between legitimate and spam emails by analysing their contents and structural features.
b. Image Recognition: SVM plays a crucial role in image recognition tasks, including object detection, facial recognition, and image segmentation. In object detection, SVM models can identify and localize objects within images by learning

discriminative features from training data. Facial recognition systems often employ SVM classifiers to match facial features against known templates, enabling biometric authentication and surveillance applications.

c. Bioinformatics: SVM finds numerous applications in bioinformatics, particularly in tasks such as protein structure prediction, gene expression analysis, and disease diagnosis. In protein structure prediction, SVM models can predict the secondary structure of proteins based on amino acid sequences, aiding in drug design and molecular biology research. SVM classifiers are also utilized in gene expression analysis to identify patterns and correlations in gene expression data, facilitating the discovery of biomarkers for various diseases.
d. Finance: SVM is widely employed in financial applications such as stock market prediction, credit risk assessment, and fraud detection. In stock market prediction, SVM models analyse historical market data to forecast future price movements, assisting investors and traders in making informed decisions. SVM classifiers are also utilized by financial institutions to assess the creditworthiness of loan applicants by analysing their credit history and financial attributes. Furthermore, SVM-based anomaly detection systems are employed to detect fraudulent activities and transactions in banking and e-commerce platforms, mitigating financial risks and losses.
e. Healthcare: SVM finds applications in healthcare for medical diagnosis, prognosis, and treatment planning. In medical diagnosis, SVM models analyse patient data such as medical images, electronic health records, and genetic profiles to assist clinicians in diagnosing diseases and predicting patient outcomes. SVM classifiers are also utilized for personalized medicine, where they aid in determining optimal treatment strategies based on individual patient characteristics and response patterns.
f. Marketing and Customer Analytics: SVM is employed in marketing and customer analytics to segment customers, predict purchase behaviour, and optimize marketing campaigns. SVM models analyse customer data such as demographics, purchasing history, and online behaviour to identify distinct customer segments and tailor marketing strategies accordingly. Additionally, SVM classifiers are utilized to predict customer churn, enabling businesses to proactively retain customers and enhance customer satisfaction.

3.2.2 Decision Tree

A decision tree is a non-parametric supervised learning algorithm, which is utilized for both classification and regression tasks. It has a hierarchical, tree structure, which consists of a root node, branches, internal nodes and leaf nodes.

The goal of the algorithm is to predict a target variable from a set of input variables and their attributes. The approach builds a tree structure through a series of binary splits (yes/no) from the root node via branches passing several decision nodes (internal nodes), until we come to leaf nodes. Each split partitions the input variables into feature regions, which are used for lower splits.

As an example, let's imagine that we are trying to assess whether or not we should go surf, we may use the following decision rules to make a choice. This will give an insight about the idea behind the decision tree, also we will be able to grasp why and where we can use this algorithm.

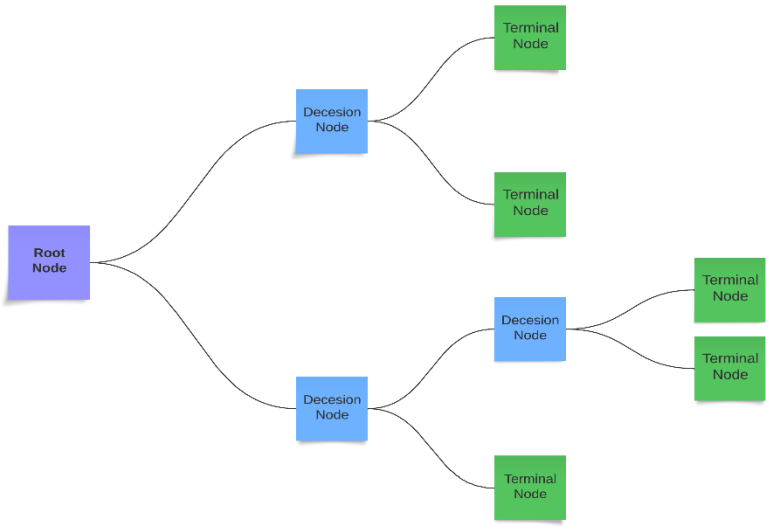


Fig. 3.3: Decision Tree

Before delving into how a decision tree make assumptions for data splitting, the goal and the result of the decision tree is required to be understood. The goal is to separate samples with a more similar structure and the result is purer subsets. Purity, in this case, responds to the quality of the splitting. There are a few typical criteria for this:

Gini impurity: A measurement that takes a sum of the ratios for each class instance to entire node instances. After that, it subtracts 1 from it. An 'Ideal' node would have a 0 impurity.

$$G_i = 1 - \sum_{k=1}^{n} {p_{i,k}}^2$$

Here P refers to the probability and k is the index number.

Informational entropy: Based on the entropy concept. In short, the more evenly arranged things are in a certain space, the greater the entropy.

$$H_i = - \sum_{\substack{k=1 \\ p_{i,k} \neq 0}}^{n} p_{i,k} \log\left(p_{i,k}\right)$$

Here P refers to the probability and k is the index number.

3.2.3 Random Forest

The Random Forest classifier is a versatile and powerful machine learning algorithm used for both classification and regression tasks. It operates by constructing a multitude of decision trees during training and outputs the mode of the classes (for classification) or the average prediction (for regression) of the individual trees.

Here's a detailed overview of the Random Forest classifier.

Ensemble Learning:

Random Forest belongs to the ensemble learning family, where multiple models are combined to improve predictive performance. In the case of Random Forest, the ensemble consists of a collection of decision trees.

Decision Trees:

Each decision tree in the Random Forest is trained independently on a random subset of the training data and a random subset of features. This randomness helps in reducing overfitting and increasing the diversity among the trees.

Bagging:

The Random Forest algorithm uses a technique called bagging (bootstrap aggregating) to train each decision tree. Bagging involves sampling the training data with replacement, which means some instances may be sampled multiple times while others may not be sampled at all.

Feature Randomness:

In addition to sampling the data, Random Forest also introduces randomness in feature selection for each split in the decision trees. This means that at each node of the tree, only a random subset of features is considered for splitting, rather than evaluating all features. This further enhances the diversity among the trees and prevents individual trees from dominating the ensemble.

Voting:

During prediction, each decision tree in the Random Forest independently classifies the input data, and the class with the most votes (mode) among all trees is chosen as the final prediction. For regression tasks, the average prediction of all trees is taken.

Benefits of Random Forest:

a. High Accuracy: Random Forest tends to achieve high accuracy compared to individual decision trees, especially for complex datasets with non-linear relationships.
b. Robustness to Overfitting: By averaging predictions from multiple trees and introducing randomness in the training process, Random Forest is less prone to overfitting.
c. Feature Importance: Random Forest provides a measure of feature importance, which indicates the contribution of each feature to the predictive performance of the model.
d. Scalability: Random Forest is relatively scalable and can handle large datasets with high-dimensional feature spaces efficiently.

Random Forest has been successfully applied to various domains, including but not limited to: Image classification, Text classification, Fraud detection, Bioinformatics, Financial modeling

3.2.4 K Nearest Neighbor

KNN is a simple yet powerful supervised machine learning algorithm used for both classification and regression tasks. It's a non-parametric and instance-based learning algorithm, meaning it doesn't make any underlying assumptions about the distribution of the data. Instead, it makes predictions based on the similarity of the new data point to its neighboring data points. KNN operates on the principle that data points with similar features tend to belong to the same class or have similar values. It classifies new data points by finding the majority class among their k nearest neighbors.

The working principle of KNN

a. Training Phase: KNN stores all available data points and their corresponding class labels (for classification) or values (for regression).
b. Prediction Phase: When a new data point is provided, KNN calculates the distance between this point and all other points in the training data. The most common distance metrics used are Euclidean distance, Manhattan distance, or Minkowski distance.
c. Nearest Neighbours Selection: KNN then selects the k nearest neighbours (data points with the smallest distances) to the new data point.
d. Classification: For classification, KNN assigns the class label that appears most frequently among the k nearest neighbours to the new data point.
e. Regression: For regression, KNN predicts the average or weighted average of the values of the k nearest neighbours.

Key Parameters

a. k: The number of neighbours to consider. Choosing an appropriate value for k is crucial; too small a value might lead to overfitting, while too large a value might lead to underfitting.
b. Distance Metric: The measure used to calculate the distance between data points. Euclidean distance is the most commonly used, but Manhattan distance, Minkowski distance, and others are also options.

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

Equation of Euclidean Distance

$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$

Equation of Manhattan Distance

$$D(\mathbf{x}_i, \mathbf{x}_j) = \left(\sum_{l=1}^{d} |x_{il} - x_{jl}|^{1/p} \right)^p$$

Equation of Minkowski Distance

Decision Boundary

a. For classification tasks, the decision boundary of KNN is flexible and depends on the distribution of data points. It tends to follow the contours of the data distribution rather than being a straight line or hyperplane.
b. For regression tasks, the predicted value at any point in the feature space is an interpolation of the values of the neighbouring data points.

Strengths of KNN

a. Simple to understand and implement.
b. No training phase; all computation is deferred until prediction time.
c. Works well with a small to medium-sized dataset and is particularly effective when the decision boundary is complex or non-linear.

CHAPTER 4

ARTIFICIAL NEURAL NETWORK IN DIGIT RECOGNITION

4.1 Introduction

Artificial Neural Networks (ANNs) are computing systems inspired by the biological neural networks of animal brains. They consist of interconnected nodes, called neurons, which work together to process complex information.

In ANNs, neurons are organized into layers: an input layer, one or more hidden layers, and an output layer. Each neuron receives input signals, processes them, and then passes the output to the next layer through connections called synapses. These connections have associated weights that determine the strength of the signal being passed.

During training, ANNs learn to perform tasks by adjusting the weights of these connections based on examples provided in the training data. This process, known as backpropagation, involves iteratively comparing the network's output with the desired output and updating the weights to minimize the difference, typically using optimization algorithms like gradient descent.

ANNs have found applications in various fields, including image and speech recognition, natural language processing, medical diagnosis, and financial forecasting, among others. They are a fundamental building block of modern machine learning and artificial intelligence systems.

4.2 How Neural Network Works

Artificial Neural Networks (ANNs) work by simulating the behavior of biological neural networks to process information and make predictions or decisions. Here's a step-by-step overview of how they operate:

a. Input Layer: The process begins with the input layer, where data is fed into the network. Each neuron in the input layer represents a feature or attribute of the input data. The values of these neurons correspond to the input data.
b. Weights and Connections: Each neuron in one layer is connected to every neuron in the subsequent layer through connections called synapses. Each connection is associated with a weight, which determines the strength of the connection. Initially, these weights are randomly assigned.
c. Processing: Once the input data is fed into the input layer, it is multiplied by the weights of the connections and summed up in each neuron of the subsequent layer. This process also includes adding a bias term to the sum. The result is passed through an activation function, which introduces non-linearity into the network, allowing it to learn complex patterns.
d. Hidden Layers: Between the input and output layers, there may be one or more hidden layers. These layers enable the network to learn intricate relationships and patterns within the data. Each neuron in a hidden layer receives inputs from all neurons in the previous layer and processes them as described above.

e. Output Layer: The processed information flows through the hidden layers until it reaches the output layer. The output layer contains neurons that represent the final output of the network, such as a classification label or a numerical value. The activation function applied to the neurons in the output layer depends on the nature of the task—common choices include softmax for classification tasks and linear activation for regression tasks.
f. Error Calculation: Once the output is generated, it is compared to the actual target or desired output. The resulting difference, known as the error or loss, quantifies how well the network performed on the given input.
g. Backpropagation: The error is then propagated backward through the network in a process called backpropagation. During backpropagation, the network adjusts the weights of the connections in order to minimize the error. This adjustment is achieved by using optimization algorithms like gradient descent, which calculate the gradient of the loss function with respect to the weights and update the weights in the direction that reduces the error.
h. Training: The entire process of feeding data into the network, calculating the output, comparing it to the desired output, and updating the weights is repeated iteratively over multiple training examples. This iterative process allows the network to learn from the data and improve its performance over time.
i. Testing and Deployment: Once the network has been trained, it can be tested on new, unseen data to evaluate its performance. If satisfactory, the trained network can then be deployed to make predictions or decisions on real-world data.

4.3 Computational Complexities in Neural Networks

While performing the task the following complexities are seen.

a. Training Complexity:
    1. Involves forward and backward passes.
    2. Computation scales with network size and depth.
b. Inference Complexity:
    1. Similar to training but excludes gradient computations.
    2. Affected by batch size and precision.
c. Hardware Acceleration:
    1. GPUs and TPUs offer significant speedups.
d. Memory Demands:
    1. Arise from parameter storage and intermediate activations.
e. Specific Operations:
    1. Certain operations, like matrix multiplications, may be computationally intensive.
f. Hyperparameter Tuning:
    1. Requires computational resources for optimization.

CHAPTER 5

CONVOLUTIONAL NEURAL NETWORK IN DIGIT RECOGNITION

5.1 Introduction

A Convolutional Neural Network (CNN) is a type of artificial neural network specifically designed for processing structured grid-like data, such as images. CNNs have gained immense popularity and proven to be highly effective in various computer vision tasks, including image classification, object detection, segmentation, and more.

At its core, a CNN consists of multiple layers, including convolutional layers, pooling layers, and fully connected layers. These layers work together to extract and learn hierarchical representations of features from input images.

Convolutional Layers: These layers extract features from the input image. Imagine a filter scanning the image, identifying edges, shapes, and other visual elements. By stacking these layers, CNNs can progressively learn higher-level features.

Pooling Layers: These layers down sample the data, reducing its dimensionality while preserving important features. This helps control computational complexity and prevents overfitting.

Activation Functions: These functions introduce non-linearity into the network, allowing it to learn more complex relationships within the data.

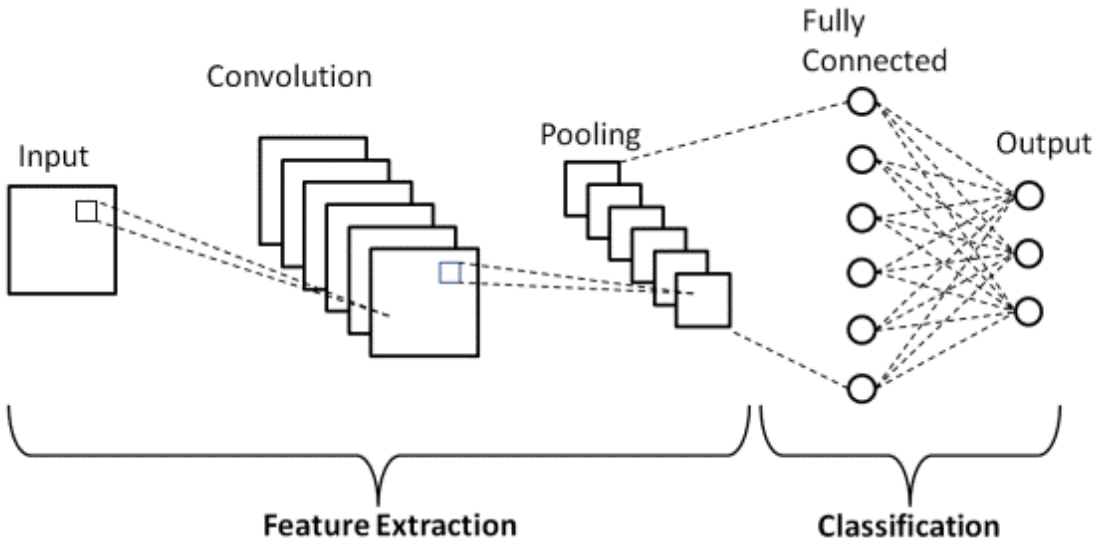


Fig. 5.1: Schematic Diagram of basic CNN

5.2 Real-Life Applications of CNN

Convolutional Neural Networks (CNNs) have found widespread real-life applications across various industries, showcasing their versatility and impact. In the field of computer vision, CNNs excel at tasks like image classification, object detection, and facial recognition. These capabilities have enabled the development of applications such as

Snapchat filters, which use CNNs to identify and manipulate facial features in real-time. Additionally, CNNs have been instrumental in advancing autonomous vehicle technology, where they are used for tasks like lane detection, traffic sign recognition, and pedestrian identification, ensuring safer and more reliable self-driving systems.

Beyond visual applications, CNNs have also made significant contributions to natural language processing. By capturing the spatial and contextual relationships within text data, CNNs have been successfully applied to tasks like sentiment analysis, text classification, and language translation, enabling more accurate and efficient processing of textual information. In the financial sector, CNNs have been utilized for stock price prediction, fraud detection, and credit risk assessment, leveraging their ability to identify patterns and trends in complex data.

In the realm of climate change research, CNNs have proven invaluable in analyzing satellite imagery and environmental data to monitor and predict changes in ecosystems, weather patterns, and natural disasters. By automating the extraction of relevant features from vast datasets, CNNs have facilitated more comprehensive and timely insights into the impacts of climate change, aiding in the development of effective mitigation and adaptation strategies.

These diverse real-life applications of Convolutional Neural Networks demonstrate their versatility and the profound impact they have had across various industries, revolutionizing how we approach complex problems and unlock new possibilities in the digital age.

5.3 Different Architectures of CNN

There are several popular architectures of Convolutional Neural Networks (CNNs), each with its own unique design and characteristics, being developed to address different challenges and applications. Here we have utilized three of the prominent architectures of CNN Xception, Resnet50, and LeNet.

5.3.1 LeNet5

LeNet-5, developed by Yann LeCun et al. in 1998, stands as a pioneering architecture in the realm of Convolutional Neural Networks (CNNs), particularly for handwritten digit recognition tasks. Comprising seven layers in total, LeNet-5 embodies the essence of convolutional networks by employing a series of convolutional, pooling, and fully connected layers. Its architecture begins with a convolutional layer followed by a sub-sampling layer, known as max-pooling, to extract robust features from input images while reducing spatial dimensions. Subsequent convolutional and pooling layers further refine these features, culminating in a fully connected layer for classification. LeNet-5's compact design, efficient parameter usage, and hierarchical feature extraction mechanisms contributed to its success in handwritten digit recognition tasks, laying the foundation for subsequent advancements in deep learning. Additionally, its architecture laid the groundwork for future CNN designs, influencing subsequent models like AlexNet and VGG. Despite its simplicity compared to modern architectures, LeNet-5 remains a cornerstone in the evolution of CNNs, demonstrating the efficacy of hierarchical feature learning and paving the way for the widespread adoption of deep learning in computer vision tasks.

5.3.2 ResNet

ResNet, or Residual Network, stands as a seminal advancement in Convolutional Neural Network (CNN) architecture, fundamentally reshaping the landscape of deep learning. Developed by Kaiming He et al. in 2015, ResNet addresses the longstanding challenge of training deep networks by introducing residual connections. These connections, also known as skip connections, allow gradients to flow directly through the network, mitigating the vanishing gradient problem associated with deep architectures. This breakthrough enabled the successful training of models surpassing hundreds of layers, ushering in a new era of deep learning. By preserving essential information through residual connections, ResNet facilitates the extraction of highly discriminative features, leading to remarkable improvements in accuracy and generalization performance across various tasks such as image classification, object detection, and semantic segmentation. Furthermore, ResNet's architectural innovation has reverberated throughout the field, inspiring subsequent models and serving as a cornerstone in advancing the capabilities of CNNs, not only in computer vision but also in other domains where complex data dependencies need to be modeled effectively.

5.3.3 Xception

Xception, an abbreviation for "Extreme Inception," represents a significant evolution in convolutional neural network (CNN) architecture, leveraging the power of depth-wise separable convolutions. Introduced by François Chollet in 2017, Xception builds upon the foundational principles of the Inception architecture while innovatively restructuring the convolutional layers. Unlike traditional CNNs, Xception replaces standard convolutions with depth-wise separable convolutions, which consist of a depth-wise convolution followed by a pointwise convolution. This separation of spatial and channel-wise operations significantly reduces the number of parameters and computational complexity, leading to improved efficiency and performance. By decoupling feature extraction across spatial and channel dimensions, Xception enhances the network's capacity to capture intricate patterns and hierarchical representations in data. This architectural refinement results in state-of-the-art performance across various computer vision tasks, including image classification, object detection, and semantic segmentation. Moreover, Xception's modular design and computational efficiency make it well-suited for deployment in resource-constrained environments, further expanding its utility in real-world applications.

CHAPTER 6

METHODOLOGY

6.1 Introduction

Based on Machine Learning and Deep Learning we have come up with two approaches for the Bangla handwritten digit recognition. In this section we will delve into those approaches.

6.2 Data Collection

The first step in our methodology involved acquiring an appropriate image dataset for our study. We selected the Ekush dataset. Ekush is the largest dataset of handwritten Bangla characters for research on handwritten Bangla character recognition. It's a multipurpose comprehensive dataset for Bangla Handwritten Characters. The dataset contains Bangla modifiers, vowels, consonants, compound letters and numerical digits that consists of 367,018 isolated handwritten characters written by 3086 unique writers which were collected from all over Bangladesh, which includes verity of the district, age group and the equal number of male and female. Due to its relevance to our research, objective we used only the numerical digit portion that includes 30687 images. These images in the dataset are of size 28x28 pixel. It contains 10 classes from 0 to 9. By making it suitable for training and evaluating machine learning models for our specific task we dived into the exploration.

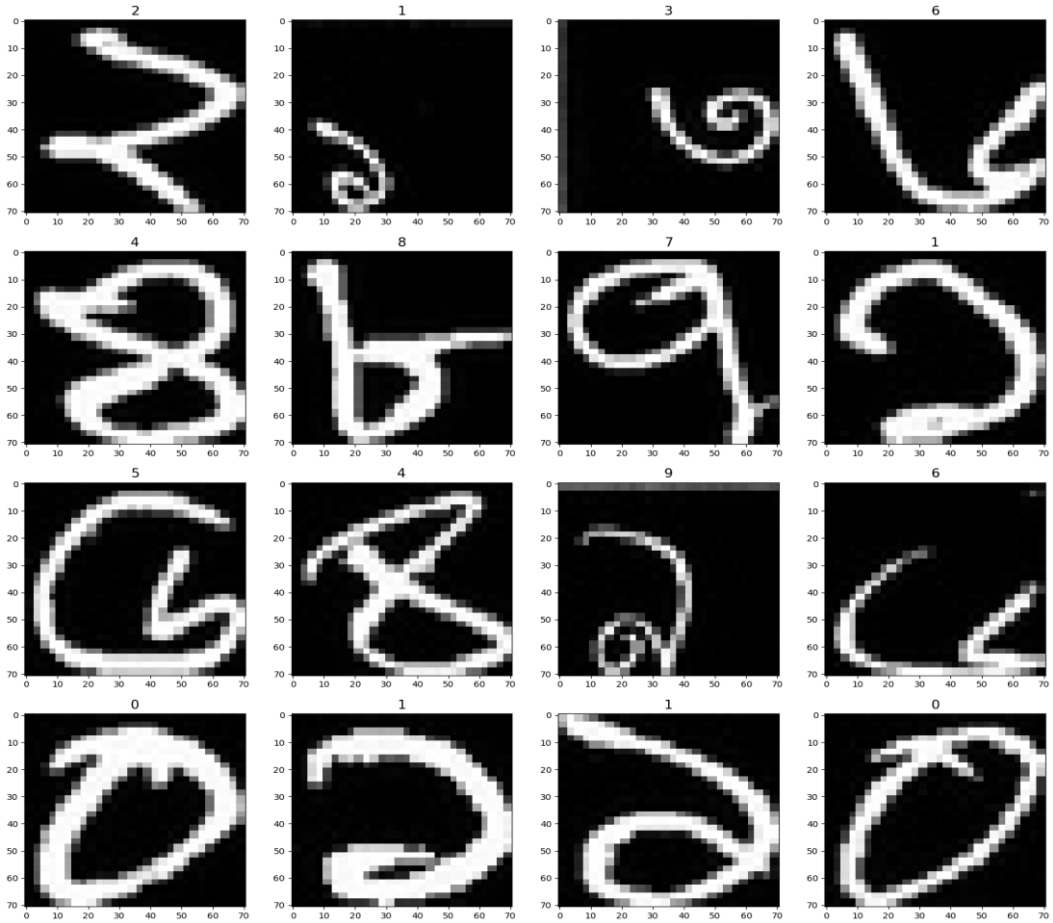


Fig. 6.1: Sample Bangla Digits

6.3 Data Preprocessing

Data is the most important part of machine learning algorithms. But most of the data found online are not ready for machine learning algorithms. So, we need some manual processing to fit our data onto any machine learning model. After collecting the dataset, we have done some preprocessing. After having done with preparation of CSV file from the raw dataset consisting of image files, we made a correction to the label which initially started with 112 to 121, and then made it to 0 to 9. Since there were two different datasets with Female and Male label we unified into a single dataset. After these preprocessing, we split the dataset into the ratio of 70:30. For some specific Machine Learning models we also transformed the dataset using standard scaler API from Scikit-learn.

We took a different approach for the CNN models. We first collected the images of the numerical digits. Then split the data into three parts: train, test, and validation. The data ratio was 80, 20, and 20. Then converted those images into data frames using data processing pipelines for image data using generators. Two image data generators have been initialized: one for data augmentation, which includes resizing pixel values to a range between 0 and 1 and applying brightness augmentation within the range of 0.8 to 1.2, and another for test data preprocessing, which solely rescales pixel values. These generators are then employed to generate batches of data from data frames. For training and validation datasets, the generator with data augmentation is used, while for the test dataset, the generator without augmentation is utilized. Data frames containing file paths and corresponding labels are specified for each dataset, along with batch sizes and target image sizes. Additionally, for the test dataset, the shuffle parameter is set to False to maintain the order of data samples during evaluation.

6.4 Model Training

Since we have used two approaches for model training, we trained the Machine Learning Model and the Deep Learning Model individually. These have been mentioned articulately below.

6.4.1 Machine Learning Model

The models we have trained are Support Vector Classifier, Decision Tree Classifier, Random Forest Classifier and K Neighbors Classifier.

To begin with, we trained all the models individually with default parameter. After that, we have performed hyperparameter tuning. Having done with these tasks, we evaluated the confusion matrix using the best parameter. These sorts of tasks were performed in all the models we have used for the performance comparison.

6.4.2 Deep Learning Model

For Deep Learning model we have used Artificial Neural Network and three different architectures of Convolutional Neural Network namely Xception, ResNet50 and LeNet. These are described below.

6.4.2.1 Artificial Neural Network

We have tried different architectures to get the best suit result for our exploration. We came up with the insight that when the layers are beyond 4 layers, the model gets overfit. After trying different architectures, we finally came to the conclusion that we got such an architecture that performed really well when it comes to the accuracy.

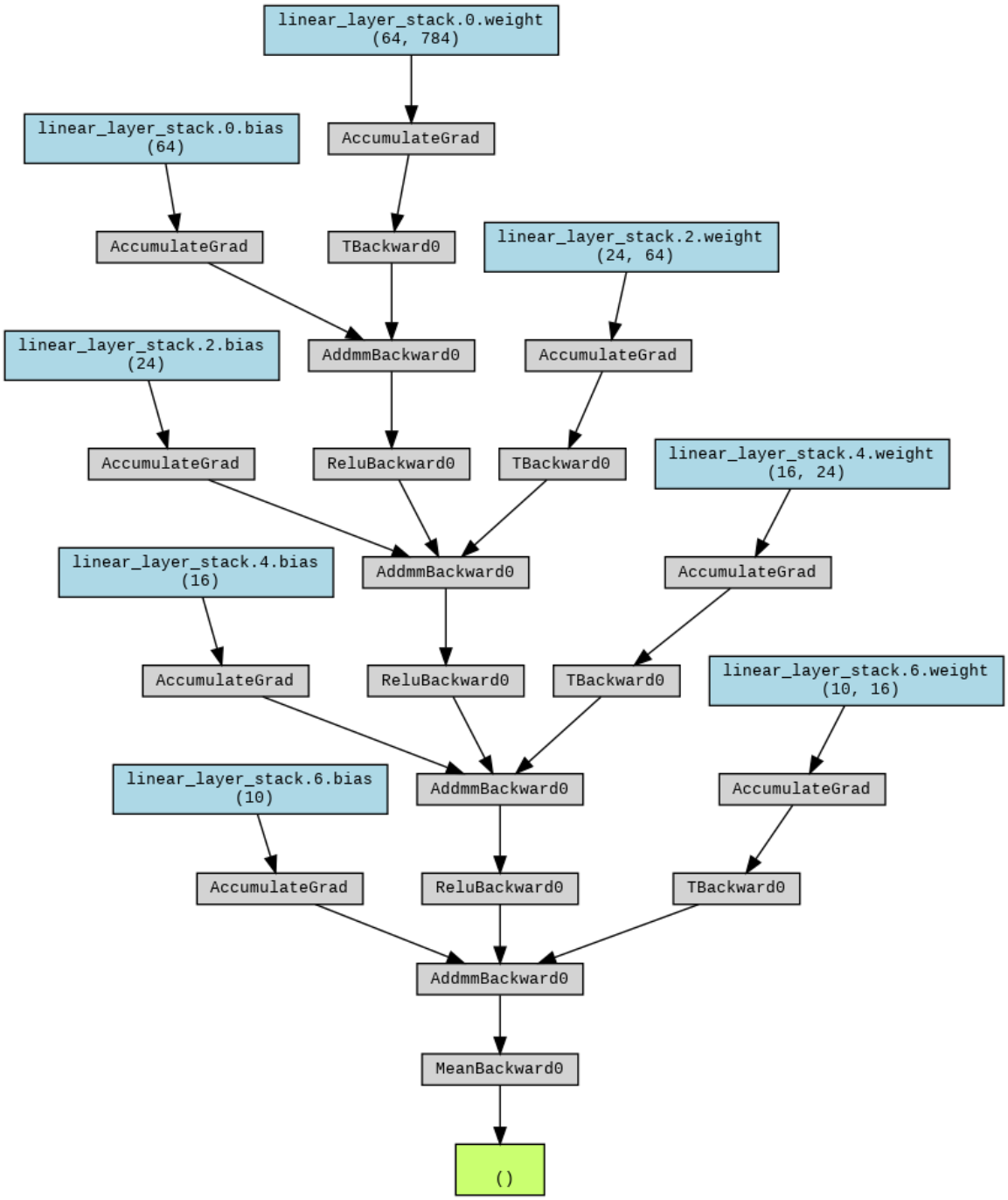


Fig. 6.2: Architecture of the ANN

6.4.2.2 Xception

The proposed model architecture comprises a convolutional neural network with a pre-trained Xception base. It includes additional layers for classification. Specifically, the model consists of 2 sub-layers on top of the Xception base. The first sublayer is a Flatten layer, followed by a Dropout layer with a dropout rate of 0.3 to mitigate overfitting. The second sub-layer is a fully connected Dense layer with 128 units and ReLU activation. Finally, the model concludes with an output layer consisting of 10 units and softmax activation for multi-class classification. The model is compiled using the Adamax optimizer with a learning rate of 0.001 and categorical cross-entropy loss function. Additionally, accuracy, precision, and recall metrics are included for model evaluation.

Layer (type)	Output Shape	Param
Xception (Functional)	(None, 2048)	20861480
flatten (Flatten)	(None, 2048)	0
dropout (Dropout)	(None, 2048)	0
dense (Dense)	(None, 128)	262272
dropout_1 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 10)	1290

Total params: 21125042
Trainable params: 21070514
Non-trainable params: 54528

Table 6.1: Xception Architecture Summary

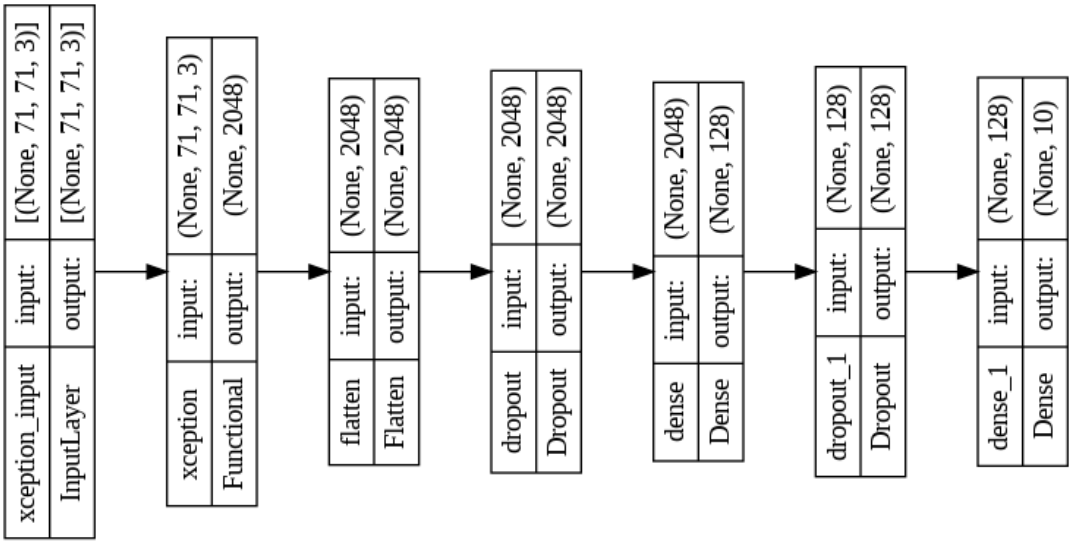


Fig. 6.3: Xception Model Architecture

6.4.2.3 ResNet

Our second CNN-based model utilizes the ResNet convolutional neural network as the base model, leveraging pre-trained weights from the ImageNet dataset. The model consists of additional layers for classification tasks. Specifically, it comprises a flattened layer to convert the output of the ResNet base into a one-dimensional array, followed by a Dropout layer with a dropout rate of 0.3 to prevent overfitting. Subsequently, a fully connected Dense layer with 128 units and ReLU activation function is employed, followed by another Dropout layer with a dropout rate of 0.25. Finally, the model includes an output Dense layer with 10 units and softmax activation, facilitating multi-class classification. Upon compiling the model, the Adam optimizer is utilized. The loss function is categorical cross-entropy, suitable for multi-class classification problems, and the metrics for evaluation include accuracy, precision, and recall.

Layer (type)	Output Shape	Param
resnet50 (Functional)	(None, 2048)	23587712
flatten (Flatten)	(None, 2048)	0
dropout (Dropout)	(None, 2048)	0
dense (Dense)	(None, 128)	262272
dropout_1(Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 10)	1290

Total params: 23851274
Trainable params: 23798154
Non-trainable params: 53120

Table 6.2: ResNet Architecture Summary

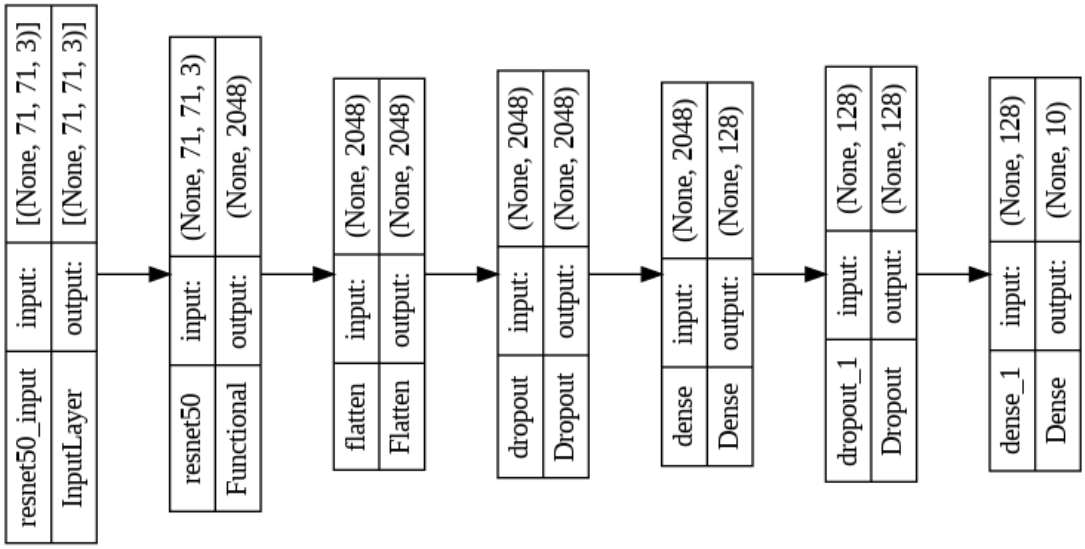


Fig. 6.4: ResNet Model Architecture

6.4.2.4 LeNet

This model followed LeNet-5 architecture. It consists of two sets of convolutional and max-pooling layers, followed by fully connected layers. The first convolutional layer employs 6 filters with a 5x5 kernel, applying the ReLU activation function. It is followed by max-pooling to reduce spatial dimensions. Similarly, the second convolutional layer utilizes 16 filters with a 5x5 kernel, also followed by max-pooling. After flattening the output, the model progresses through two dense layers with 120 and 84 neurons respectively, both utilizing the ReLU activation function. The output layer comprises 10 neurons with softmax activation, enabling multi-class classification. It is trained using the Adam optimizer and categorical cross-entropy loss function over 10 epochs, with a batch size of 32.

Layer (type)	Output Shape	Param
conv2d (Conv2D)	(None, 24, 24, 6)	156
max_pooling2d (MaxPooling2D)	(None, 12, 12, 6)	0
conv2d_1(Conv2D)	(None, 8, 8, 16)	2416
max_pooling2d_1 (MaxPooling2D)	(None, 4, 4, 16)	0
flatten (Flatten)	(None, 256)	0
dense (Dense)	(None, 120)	30840
dense_1 (Dense)	(None, 84)	10164

Total params: 44426
Trainable params: 44426
Non-trainable params: 0

Table 6.3: LeNet Architecture Summary

CHAPTER 7

FINDINGS AND RESULTS

7.1 Introduction

Several approaches and strategies have been made so far to solve the digit recognition problem. In Bangla handwritten digit recognition there can be found several works. Some papers despite having their comparative study they lack when it comes to exploring all the topnotch models in Machine Learning and Deep Learning Algorithm. They always focused solely on CNN based architecture. Some have done some data augmentation. To bridge the gap, we have not only explored ML and DL, but also acquired some satisfied accuracy.

In machine learning, the ultimate goal is to build models that can accurately generalize to unseen data and make reliable predictions or decisions. Evaluation metrics serve as quantitative measures to gauge how well a model performs on a given task. By quantifying the performance of models, evaluation metrics enable practitioners to compare different algorithms, fine-tune model parameters, and validate model suitability for specific applications. Moreover, evaluation metrics provide valuable insights into model strengths, weaknesses, and areas for improvement, guiding iterative model development and refinement processes. The evaluation metrics are as follow.

a. Precision: Precision measures the accuracy of positive predictions made by the model. It is calculated as the ratio of true positives (TP) to the sum of true positives and false positives (FP) for a particular class

$$\boldsymbol{Precision} = \frac{\boldsymbol{TP}}{\boldsymbol{TP} + \boldsymbol{FP}}$$

b. Recall (Sensibility): Recall measures the ability of the model to correctly identify positive instances from all actual positive instances. It is calculated as the ratio of true positives (TP) to the sum of true positives and false negatives (FN) for a particular class.

$$\boldsymbol{Recall} = \frac{\boldsymbol{TP}}{\boldsymbol{TP} + \boldsymbol{FN}}$$

c. F1-score: F1-score is the harmonic mean of precision and recall. It provides a balance between precision and recall and is useful when classes are imbalanced. F1-score is calculated as follows.

$$\boldsymbol{F1_Score} = \frac{\mathbf{2} \times \boldsymbol{Precision} \times \boldsymbol{Recall}}{\boldsymbol{Precision} + \boldsymbol{Recall}}$$

d. Support: Support refers to the number of actual occurrences of each class in the dataset. It represents the number of instances belonging to a particular class.

e. Accuracy: Accuracy is the overall performance of the model, calculated as the ratio of correctly classified instances to the total number of instances. It is calculated as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

f. Macro Average: Macro average calculates the average precision, recall, and F1-score across all classes, giving each class equal weight. It is computed by averaging the precision, recall, and F1-score of each class without considering class imbalance.

g. Weighted Average: Weighted average calculates the average precision, recall, and F1-score across all classes, but it takes into account the support (number of occurrences) of each class, giving more weight to classes with more instances. It is computed by averaging the precision, recall, and F1-score of each class, weighted by the number of instances of each class.

h. Confusion Matrix: A confusion matrix is a table that is used to evaluate the performance of a classification model. It compares the actual class labels of a dataset with the predicted class labels produced by the model. The confusion matrix provides a comprehensive summary of the model's performance by displaying the counts of true positives, true negatives, false positives, and false negatives.

Overall, high precision, recall, and F1-score values, along with high accuracy, indicate good performance of the model across all classes.

7.2 Analysis of Results

The performance comparison was accomplished based on the evaluation metrics—precision score, recall/sensitivity score, F1 score, and specificity score. True positive, true negative, false positive, and false negative values were used to compute the evaluation metrics. The results were plotted using Python's Matplotlib library for better interpretation and visualization. We got highest test accuracy 99.48% through Xception of CNN.

7.2.1 Evaluation Report based on Precision, Recall, F1 Score

The performance evaluation of Xception, ResNet and LeNet Models are provided below.

	precision	recall	f1-score	support
0	1.00	1.00	1.00	307
1	1.00	0.98	0.99	307
2	0.99	1.00	1.00	306
3	0.99	0.99	0.99	308
4	0.99	0.99	0.99	307
5	0.99	1.00	0.99	307
6	0.99	0.99	0.99	307
7	1.00	1.00	1.00	308
8	1.00	1.00	1.00	308
9	0.99	0.99	0.99	306
accuracy			0.99	3071
macro avg	0.99	0.99	0.99	3071
weighted avg	0.99	0.99	0.99	3071

Fig. 7.1: Evaluation report of Xception Model

	precision	recall	f1-score	support
0	0.99	0.99	0.99	307
1	0.97	0.99	0.98	307
2	0.99	0.99	0.99	306
3	1.00	0.99	0.99	308
4	0.99	0.98	0.99	307
5	0.99	0.98	0.99	307
6	0.97	0.98	0.98	307
7	0.99	0.99	0.99	308
8	0.98	1.00	0.99	308
9	0.99	0.97	0.98	306
accuracy			0.99	3071
macro avg	0.99	0.99	0.99	3071
weighted avg	0.99	0.99	0.99	3071

Fig. 7.2: Evaluation report of ResNet Model

	precision	recall	f1-score	support
0	0.97	0.98	0.97	307
1	0.92	0.96	0.94	307
2	0.99	0.98	0.99	306
3	0.97	0.98	0.98	308
4	0.94	0.98	0.96	307
5	0.96	0.97	0.97	307
6	0.99	0.97	0.98	307
7	0.99	0.96	0.98	308
8	0.99	0.99	0.99	308
9	0.97	0.92	0.94	306
accuracy			0.97	3071
macro avg	0.97	0.97	0.97	3071
weighted avg	0.97	0.97	0.97	3071

Fig. 7.3: Evaluation report of LeNet Model

From the provided data, based on Xception model it is evident that the model exhibits exceptional performance across all evaluated metrics. The high precision scores, ranging between 0.99 and 1.00, while 0.97-1.0 in ResNet and 0.92-0.99 in LeNet, signify a minimal rate of false positive predictions. This implies that when the model predicts a specific class, it is highly likely to be correct. Additionally, the high recall scores, ranging from 0.98 to 1.00, indicate the model's ability to correctly identify the majority of positive instances for each class, capturing most relevant instances. Moreover, the uniformly high F1-scores, predominantly at 0.99 or 1.00, underscore the model's adeptness in maintaining a balance between minimizing false positives and false negatives. The overall accuracy of 0.99 further underscores the model's prowess, correctly classifying 99% of instances in the dataset. This collectively illustrates the model's exceptional performance in distinguishing between different classes with precision, recall, and accuracy.

7.2.2 Evaluation based on Training Metrics

For the deep learning model training the metrics over epochs go as following figures.

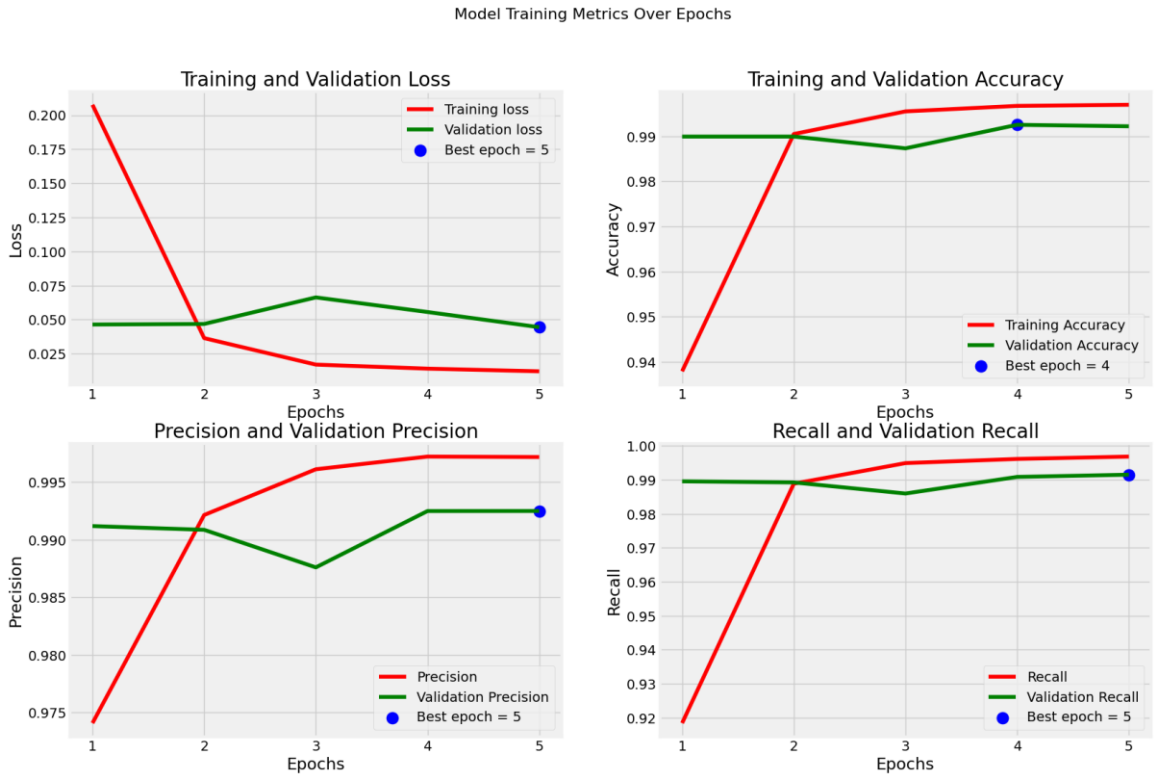


Fig. 7.4: Model Training Metrics Over Epochs of Xception



Fig. 7.5: Model Training Metrics Over Epochs of Resnet

7.2.3 Evaluation based on Confusion Matrix

Confusion Matrix was used to evaluate the performance of our models. Also, it provides some other benefits. These are provided below.

a. **Evaluation of Model Performance:** The confusion matrix provides insights into how well a classification model is performing. By analysing the counts in each cell of the matrix, one can assess the model's accuracy, precision, recall, and other performance metrics.
b. **Identification of Prediction Errors:** The confusion matrix helps identify the types of errors made by the model. False positives and false negatives are highlighted, indicating instances where the model misclassifies data points. Understanding these errors can guide further model refinement or data pre-processing.
c. **Optimization of Model Parameters:** By analysing the confusion matrix, one can adjust the model's parameters or threshold values to improve its performance. For example, if false positives are prevalent, one may adjust the model's decision threshold to reduce the false positive rate.
d. **Comparative Analysis:** Confusion matrices allow for the comparison of different models or variations of a single model. By comparing the confusion matrices of multiple models, one can determine which model performs better for a specific task or dataset.

Example of Confusion Matrix:

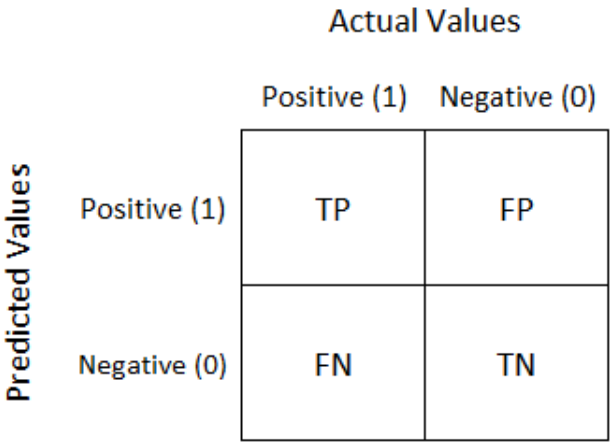


Fig. 7.6: Confusion Matrix

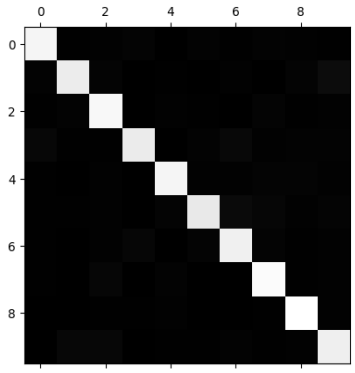
In the example of confusion matrix:

True Positive (TP): Instances correctly predicted as positive.

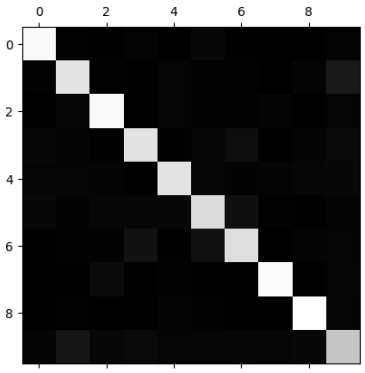
True Negative (TN): Instances correctly predicted as negative.

False Positive (FP): Instances incorrectly predicted as positive.

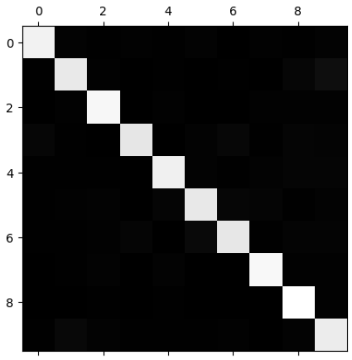
False Negative (FN): Instances incorrectly predicted as negative.



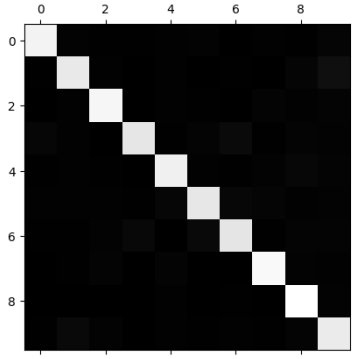
Confusion Matrix of SVM



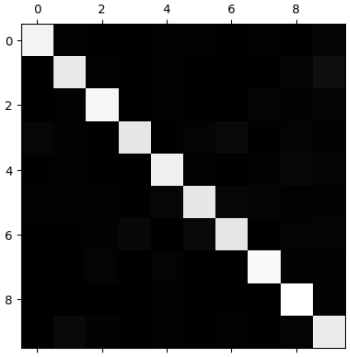
Confusion Matrix of DT



Confusion Matrix of RF



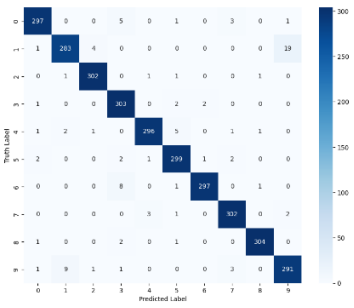
Confusion Matrix of KNN



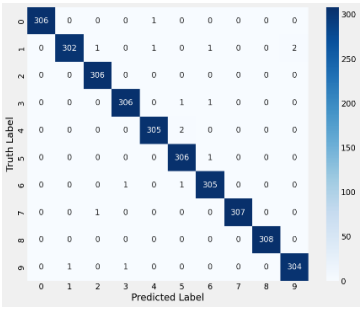
Confusion Matrix of ANN



Confusion Matrix of ResNet



Confusion Matrix of LeNet



Confusion Matrix of Xception

Fig. 7.7: Confusion Matrix of the Models

The comparative accuracy of the models is shown below.

Model Name	Training Accuracy	Testing Accuracy
SVM	96.25%	91%
Decision Tree	100%	80.44%
Random Forest	100%	91.22%
KNN	100%	90.38%
ANN	(Cross Entropy Loss 0.09)	90.5%
Xception	99.98%	99.48%
LeNet	99.22%	97.23%
ResNet	99.18%	98.63%

Table 7.1: Accuracy of Different Models

Work	Accuracy
A Classical Approach to Handcrafted Feature Extraction Techniques for Bangla Handwritten Digit Recognition[2]	95.68%
Bengali Handwritten Digit Recognition using CNN with Explainable AI [9]	96.7%
An Efficient Method for Bangla Handwritten Digit Recognition Using Convolutional Neural Network [10]	98.3%
BengaliNet: A Low-Cost Novel Convolutional Neural Network for Bengali Handwritten Characters Recognition [11]	98.36%

Table 7.2: Results of Ekush Dataset on different existing works

The following Table 7.3 showing a comparison between some the previous work with our work. From this table, we found that our model got so far, the best accuracy among all the models used.

Work	**Accuracy**
Recognition of Bangla handwritten basic characters and digits using convex hull-based feature set [12]	76.86%
Handwritten Bangla Digit Recognition Using Hierarchical Bayesian Network [13]	87.5%
Bangla Handwritten Digit Recognition Using Deep CNN for Large and Unbiased Dataset [14]	92.72%
Handwritten Bangla Digit Recognition Using Classifier Combination Through DS Technique [15]	95.1%
Bengali Handwritten Character Classification using Transfer Learning on Deep Convolutional Neural Network [16]	96.12%
Bayanno-Net: Bangla Handwritten Digit Recognition using Convolutional Neural Networks [17]	97.00%
Unveiling the Comparative Insight of Bangla Handwritten Digit Recognition using Machine Learning and Deep Learning Models	**99.48%**

Table 7.3. Result Comparison of different Models

CHAPTER 8

CONCLUSION AND FUTURE WORKS

8.1 Conclusion

In conclusion, our study systematically evaluated various machine learning and deep learning models for Bangla numerical digit recognition. Through rigorous experimentation, we determined that among the deep learning architectures examined, Xception exhibited the most superior performance, followed by ResNet, LeNet, and ANN, in descending order. This conclusion was drawn based on comprehensive evaluation metrics including precision, recall/sensitivity, F1 score, and specificity. The dataset is consisted of numerical digit, and this work classified them into ten classes. Xception outperformed ResNet and LeNet for Bangla numerical digit recognition due to its efficient depth-wise separable convolutions, which reduces parameters and computation, making it ideal for smaller datasets. Its adaptability to complex patterns and ease of fine-tuning contribute to better recognition of diverse handwritten styles. Additionally, its training efficiency suits datasets with limited resources.

8.2 Limitations

There are a lot of datasets on Bengali handwritten character. Due to time constraints, we have used only the Bengali digit. New dimensions in the future, like (i) taking the characters into consideration and applying different algorithms on them (ii) finding a more accurate model for the dataset (iii) working with other datasets created on Bengali handwritten digits.

8.3 Future Works

Since there was no up-to-date work implicitly and solely on Bangla handwritten digit, we tried to perform the performance comparison based on ML and DL algorithm. Despite getting a satisfactory result, for the future still there is a room for advancement and improvement in the dataset expansion. To add more, pre-training the models on larger, more diverse datasets before fine-tuning on the Ekush dataset could potentially improve performance, especially in scenarios with limited training data.

REFERENCES

[1] "Ekush: Bangla Handwritten Characters Dataset for Recognition Research." May 02, 2024. [Online]. Available: https://rabby.dev/ekush/index.htm#external

[2] M. F. Wahid, M. F. Shahriar, and M. S. I. Sobuj, "A Classical Approach to Handcrafted Feature Extraction Techniques for Bangla Handwritten Digit Recognition," in *Proceedings of International Conference on Electronics, Communications and Information Technology, ICECIT 2021,* Institute of Electrical and Electronics Engineers Inc., 2021. doi: 10.1109/ICECIT54077.2021.9641406.

[3] A. Shawon, M. Jamil-Ur Rahman, F. Mahmud, and M. M. Arefin Zaman, "Bangla Handwritten Digit Recognition Using Deep CNN for Large and Unbiased Dataset," *2018 International Conference on Bangla Speech and Language Processing, ICBSLP 2018,* Nov. 2018, doi: 10.1109/ICBSLP.2018.8554900.

[4] A. Roy, N. Mazumder, N. Das, R. Sarkar, S. Basu, and M. Nasipuri, "A new quad tree-based feature set for recognition of handwritten bangla numerals," *2012 IEEE International Conference on Engineering Education: Innovative Practices and Future Trends (AICERA),* 2012, doi: 10.1109/AICERA.2012.6306727.

[5] I. Bhattacharjee, "An Efficient Method for Bangla Handwritten Digit Recognition Using Convolutional Neural Network," *Technium: Romanian Journal of Applied Sciences and Technology*, vol. 18, pp. 65–74, Dec. 2023, doi: 10.47577/ TECHNIUM.V18I.10243.

[6] M. A. R. Alif, S. Ahmed, and M. A. Hasan, "Isolated Bangla handwritten character recognition with convolutional neural network," *20th International Conference of Computer and Information Technology, ICCIT 2017*, vol. 2018-January, pp. 1–6, Jul. 2017, doi: 10.1109/ICCITECHN.2017.8281823.

[7] Y. Wen, Y. Lu, and P. Shi, "Handwritten Bangla numeral recognition system and its application to postal automation," *Pattern Recognit*, vol. 40, no. 1, pp. 99–107, Jan. 2007, doi: 10.1016/J.PATCOG.2006.07.001.

[8] T. Hassan and H. A. Khan, "Handwritten Bangla numeral recognition using Local Binary Pattern," *2nd International Conference on Electrical Engineering and Information and Communication Technology, iCEEiCT 2015*, Oct. 2015, doi: 10.1109/ICEEICT.2015.7307371.

[9] M. T. Rouf Shawon, R. Tanvir, and M. G. Rabiul Alam, "Bengali Handwritten Digit Recognition using CNN with Explainable AI," *2022 4th International Conference on Sustainable Technologies for Industry 4.0, STI 2022*, Dec. 2022, doi: 10.1109/STI56238.2022.10103341.

[10] I. Bhattacharjee, "An Efficient Method for Bangla Handwritten Digit Recognition Using Convolutional Neural Network." www.techniumscience.com

[11] A. Sayeed, J. Shin, M. A. M. Hasan, A. Y. Srizon, and M. M. Hasan, "BengaliNet: A low-cost novel convolutional neural network for bengali handwritten characters

recognition," *Applied Sciences (Switzerland)*, vol. 11, no. 15, p. 6845, Aug. 2021, doi: 10.3390/APP11156845/S1.

[12] N. Das *et al.*, "Recognition of Handwritten Bangla Basic Characters and Digits using Convex Hull based Feature Set," *International Conference on Artificial Intelligence and Pattern Recognition 2009, AIPR 2009*, pp. 380–386, Oct. 2014, doi: 10.13140/2.1.3689.4089.

[13] J. W. Xu, J. Xu, and Y. Lu, "Handwritten Bangla digit recognition using hierarchical Bayesian network," *Proceedings of 2008 3rd International Conference on Intelligent System and Knowledge Engineering, ISKE 2008*, pp. 1096–1099, 2008, doi: 10.1109/ISKE.2008.4731093.

[14] A. Shawon, M. Jamil-Ur Rahman, F. Mahmud, and M. M. Arefin Zaman, "Bangla Handwritten Digit Recognition Using Deep CNN for Large and Unbiased Dataset," *2018 International Conference on Bangla Speech and Language Processing, ICBSLP 2018*, Nov. 2018, doi: 10.1109/ICBSLP.2018.8554900.

[15] S. Basu, R. Sarkar, N. Das, M. Kundu, M. Nasipuri, and D. K. Basu, "Handwritten Bangla Digit Recognition Using Classifier Combination Through DS Technique," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 3776 LNCS, pp. 236–241, 2005, doi: 10.1007/11590316_32.

[16] S. Chatterjee, R. K. Dutta, D. Ganguly, K. Chatterjee, and S. Roy, "Bengali Handwritten Character Classification/Recognition using Transfer Learning on Deep Convolutional Neural Network," Feb. 2019, Accessed: May 02, 2024. [Online]. Available: http://arxiv.org/abs/1902.11133

[17] M. S. Islam, M. F. A. Foysal, and S. R. H. Noori, "Bayanno-Net: Bangla Handwritten Digit Recognition using Convolutional Neural Networks," *Proceedings of 2019 IEEE Region 10 Symposium, TENSYMP 2019*, pp. 23–27, Jun. 2019, doi: 10.1109/TENSYMP46218.2019.8971167.