

'HINDI HANDWRITTEN CHARACTER RECOGNITION WITH DEEP LEARNING'

A REPORT

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE AWARD OF THE DEGREE

OF

MASTER OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by:

Rohtash Singh

2K22/CSE/20

Under the supervision of

Dr. Manoj Sethi



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

CERTIFICATE

I hereby certify that the Major Project-II titled '**HINDI HANDWRITTEN CHARACTER RECOGNITION WITH DEEP LEARNING**' which is submitted by **Rohtash Singh** to Computer Science and Engineering Department, Delhi Technological University, Delhi in partial fulfillment of the requirement of the award of the degree of Master of Technology degree in Computer Science and Engineering at Delhi Technological University is a record of the project work carried out by the student under my supervision. To the best of my knowledge this work has not been submitted in part or full for any degree or diploma to this university or elsewhere.

Place: Delhi

Dr. Manoj Sethi

Date:

Professor

CANDIDATE'S DECLARATION

I **Rohtash Singh (2K22/CSE/20)** of MTech (Computer Science and Engineering), hereby declare that the Major Project-II work titled '**HINDI HANDWRITTEN CHARACTER RECOGNITION WITH DEEP LEARNING**' which is being submitted to the Department of Computer Science and Engineering, Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of Master of Technology, is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of any degree, diploma, associateship, fellowship or other similar title or recognition.

Place: Delhi

Rohtash Singh

Date:

ACKNOWLEDGEMENT

I am grateful to **Dr. Manoj Sethi**, Professor (Department of Computer Science and Engineering), **Delhi Technological University (Formerly Delhi College of Engineering)**, New Delhi and all other faculty members of our department for their astute guidance, constant encouragement and sincere support for this project work.

I would like to take this opportunity to express our profound gratitude and deep regard to our project mentor **Dr. Manoj Sethi**, for his exemplary guidance, valuable feedback and constant encouragement throughout the duration of the project. His valuable suggestions were of immense help throughout our project work. His perspective criticism kept us working to make this project in a much better way. Working under him was an extremely knowledgeable experience for us.

We would also like to give our sincere gratitude to all our friends for their help and support.

Place: Delhi

Rohtash Singh

Abstract

Handwritten character recognition plays a crucial role in various applications, including document digitization, language translation, and text analysis. This thesis presents a comprehensive investigation into the application of deep learning techniques for the recognition of handwritten characters, with a specific focus on Hindi characters. Leveraging insights from existing literature and building upon recent advancements in deep learning methodologies, the research aims to develop and evaluate novel approaches for improving the accuracy and efficiency of character recognition systems.

The study begins with a thorough review of traditional methods and contemporary deep learning architectures used in handwritten character recognition. Emphasis is placed on understanding the evolution of techniques and identifying key challenges in the field. Subsequently, a detailed methodology is proposed, encompassing data collection, preprocessing, feature extraction, and the implementation of deep neural networks optimized with advanced algorithms such as RMSprop and Adam.

Experimental evaluations are conducted on a substantial collection of handwritten Hindi character datasets, employing rigorous training and testing procedures. The results demonstrate the efficacy of the proposed methodology, with significant improvements in recognition accuracy compared to baseline models. Comparative analyses are presented, highlighting the strengths and limitations of different deep learning approaches.

Contents

Certificate.....	1
Candidate’s Declaration.....	2
Acknowledgement.....	3
Abstract	4
List of Tables	7
List of Figures.....	8
List of Abbreviations	9
1. Introduction.....	10
1.1 Background and Motivation.....	10-11
1.2 Problem Statements	11-13
1.3 Objectives of the Research.....	13-14
1.4 Scope and Limitations	14- 15
1.5 Applications of handwriting recognition systems	16
1.6 Overview of the thesis structure.....	16-17
2. Literature Review	18
2.1 Evaluation of deep learning in image classification.....	18
2.2 Character recognition across different scripts	18-19
2.3 Comparative Survey of Character Recognition Systems	19-21
2.4 Conclusion of the study	21
3. Proposed Methodology.....	22

3.1 Objective	22
3.2 Proposed recognition System.....	22-23
3.3 Convolutional Neural Networks (CNN) Model.....	23-28
3.4 Brief Explanation of the Classifiers Used	28-29
3.5 Performance Evaluation Criteria	29-30
4. Experimental Setup	31
4.1 Installation of libraries.....	31-32
4.2 Dataset Selection for Experimental Analysis	32-34
4.3 Algorithmic Workflow.....	34-37
4.4 Formulation of (CNN) Architecture.....	37-38
4.4 Development of a Universal Function.....	39
5. Result and Analysis	40
5.1 Detailed Findings from CNN-Adam.....	40-41
5.2 Detailed Findings from CNN-RMSProp.....	41-42
5.3 Comparative analysis of the Models.....	42-43
6. Conclusion.....	44
7. Future Scope	45
8. Bibliography.....	46-48

LIST OF TABLES

Table 1	Analysis of Different Character Recognition Systems
Table 2	Layer types in Convolutional Neural Network (CNN) Model
Table 3	Classification of Hindi Script Characters by Character Class
Table 4	Detailed Findings from CNN-Adam
Table 5	Analysis of Test Database Results Employing CNN

LIST OF FIGURES

- Fig.1 Categorization of Handwriting Recognition Strategies
- Fig.2 (a) Vowels of Hindi Script. (b) Consonants and their half forms in Hindi Script
- Fig.3 Comparable Handwritten Hindi Characters
- Fig.4 Various phases of a Handwritten Character Recognition System
- Fig.5 Illustrates the graphical representation of the Rectified Linear Unit (ReLU) activation function.
- Fig.6 Convolutional Neural Networks (CNNs)
- Fig.7 Visual Representation of Hindi Script Characters from Dataset
- Fig.8 Architectural Blueprint of the CNN Model
- Fig.9 Comparison of CNN-Adam and CNN- RMSProp

LIST OF ABBREVIATIONS

STDM	Spatiotemporal Data Mining
ST	Spatiotemporal
ML	Machine Learning
DL	Deep Learning
CNN	Convolutional Neural Network
TN	True Negative
TP	True Positive
FN	False Negative
FP	FP False Positive
AUC	Area under Curve
ROC	Receiver Operating Characteristic Curve
OPTICS	Ordering Points to Identify the Clustering Structure
HDBSCAN	Hierarchical Density-Based Spatial Clustering of Applications with
SC	Spatial Cluster

CHAPTER 1

INTRODUCTION

Handwritten character recognition stands at the intersection of computer vision, machine learning, and natural language processing, playing a pivotal role in automating the interpretation and analysis of handwritten documents. In an era marked by digital transformation and data-driven decision-making, the ability to accurately and efficiently recognize handwritten characters holds immense significance across a multitude of domains, ranging from document digitization and language translation to text analysis and information retrieval. Traditional methods for handwritten character recognition have long relied on handcrafted features and rule-based algorithms, often struggling to cope with variations in handwriting styles, noise, and complex character shapes. However, recent advancements in deep learning techniques, particularly convolutional neural networks (CNNs), have redefined the landscape of character recognition, offering automated feature extraction and robust classification capabilities.

Against this backdrop, this thesis embarks on a journey to explore the application of deep learning methodologies for the recognition of handwritten characters, with a specific focus on Hindi characters. By delving into the nuances of deep learning architectures, preprocessing techniques, and experimental evaluations, this research endeavors to push the boundaries of recognition accuracy and efficiency, addressing the challenges and limitations inherent in recognizing handwritten characters. Through rigorous experimentation and analysis, this thesis aims to contribute novel insights, methodologies, and recommendations to the field of handwritten character recognition, laying the groundwork for future advancements and applications in this domain.

1.1 Background and Motivation

Handwritten character recognition occupies a pivotal position at the confluence of image processing and pattern recognition, serving as a cornerstone in the digitization of textual information. In an era marked by the widespread adoption of digital technologies and the exponential growth of data, the ability to accurately and efficiently recognize handwritten characters has emerged as a pressing need across various industries and applications. From automating data entry tasks to facilitating document retrieval and language translation, the importance of reliable character recognition systems cannot be overstated.

Traditional methods for handwritten character recognition have long been dominated by approaches reliant on handcrafted features and rule-based algorithms. These methods, while effective to a certain extent, often falter in the face of real-world challenges such as variations in handwriting styles, inherent noise in scanned documents, and the intricate shapes of handwritten characters. The reliance on manual feature engineering can introduce biases and limitations, hindering the adaptability and scalability of recognition systems.

In recent years, the advent of deep learning techniques has heralded a paradigm shift in the field of character recognition. Among these techniques, convolutional neural networks (CNNs) have emerged

as a cornerstone, offering a powerful framework for automated feature extraction and robust classification. By leveraging hierarchical representations learned from data, CNNs can effectively capture intricate patterns and nuances in handwritten characters, enabling more accurate and reliable recognition performance.

The promise of deep learning in overcoming the limitations of traditional methods has spurred a renewed interest in the application of CNNs to handwritten character recognition. Researchers and practitioners alike are increasingly exploring the capabilities of CNNs in addressing the inherent challenges of recognizing handwritten characters, including variability in handwriting styles, noise, and complex character shapes. [2] The growing body of literature attesting to the efficacy of CNNs in character recognition tasks underscores the transformative potential of deep learning in this domain.

Against this backdrop, this research endeavors to delve deeper into the application of CNNs for handwritten character recognition, with a specific focus on addressing the challenges unique to recognizing Hindi characters. By harnessing the power of deep learning and exploring innovative methodologies for preprocessing, feature extraction, and classification, this study seeks to push the boundaries of recognition accuracy and efficiency in Fig 1.1. Through rigorous experimentation and analysis, the aim is to contribute to the advancement of the field and pave the way for more effective and scalable solutions for handwritten character recognition.

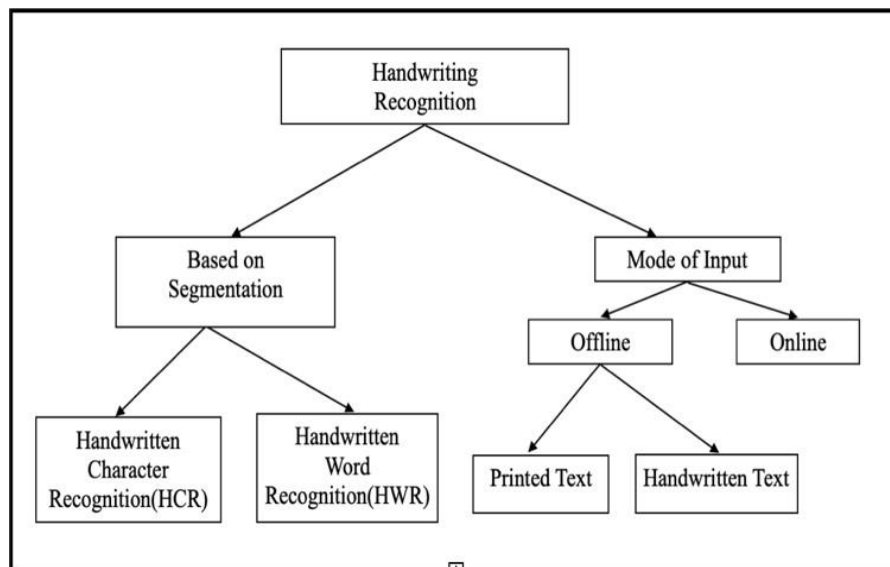


Fig. 1.1 Categorization of Handwriting Recognition Strategies

1.2 Problem Statements

Despite significant advancements in the field of handwritten character recognition, a myriad of challenges continues to impede the development of accurate, efficient, and scalable recognition systems. These challenges stem from the complex and multifaceted nature of handwritten characters, as well as the diverse linguistic and cultural contexts in which they exist.

1. Need for Improved Accuracy and Efficiency:

Handwritten character recognition systems must contend with inherent variability in handwriting styles, which can pose significant challenges to achieving high levels of accuracy. The intricacies of individual writing styles, coupled with factors such as pen pressure, stroke order, and writing speed, contribute to the difficulty of reliably recognizing handwritten characters.

Furthermore, the demand for increased recognition efficiency is driven by the growing volume and complexity of handwritten documents in both digital and physical formats. Efficient recognition systems are essential for automating data entry tasks, enabling rapid information retrieval, and facilitating seamless integration with downstream applications such as language translation and text analysis.

2. Challenges in Handling Diverse Scripts and Languages:

Recognition systems must demonstrate robustness and adaptability when faced with handwritten characters from diverse scripts and languages. Each script presents unique challenges in terms of character shapes, ligatures, diacritics, and contextual variations, necessitating the development of language-specific recognition models.

The adaptation of existing techniques to specific languages, such as Hindi, poses particular challenges due to the complex nature of the script and the wide variability in handwriting styles. The intricate conjuncts and ligatures characteristic of the Devanagari script in Fig 1.2 add an additional layer of complexity to the recognition process, requiring specialized methodologies and datasets for effective training and evaluation [6].

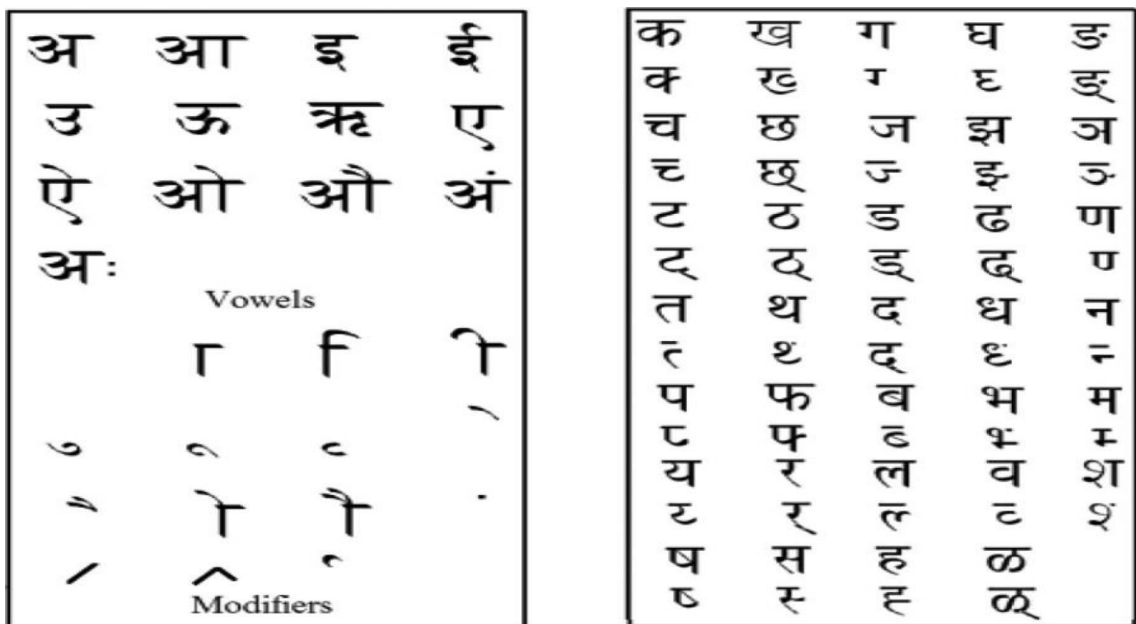


Fig. 1.2 (a) Vowels of Hindi Script. (b) Consonants and their half forms in Hindi Script

3. Scalability for Real-World Applications:

Recognition systems must demonstrate scalability to handle large datasets and real-world applications with varying degrees of complexity and volume. Scalability encompasses not only the ability to process large volumes of data efficiently but also the capability to adapt to dynamic environments and evolving user requirements.

In practical deployment scenarios, recognition systems must operate seamlessly across diverse platforms and devices, accommodating varying levels of computational resources and processing constraints. Scalable recognition solutions are essential for supporting applications such as document digitization, automated form processing, and handwriting-based authentication in real-world settings.

To sum up, the ongoing difficulties with handwritten character recognition highlight the necessity for novel strategies and techniques to raise scalability, accuracy, and efficiency. It takes a multidisciplinary approach combining knowledge of languages [35], image processing, machine learning, and human-computer interaction to address these problems. This research aims to contribute to the development of more robust and effective recognition algorithms, especially in the setting of various scripts and languages like Hindi, by clarifying the unique difficulties and complexity involved in identifying handwritten characters.

1.3 Objectives of the Research

The objectives of this research endeavor to address the complex challenges inherent in recognizing handwritten characters, particularly in the context of Hindi script, through the application of deep learning methodologies, with convolutional neural networks (CNNs) serving as the primary focus. These objectives are structured to facilitate a systematic exploration and evaluation of novel approaches aimed at enhancing the accuracy, efficiency, and robustness of handwritten character recognition systems.

1. Exploration and Evaluation of Existing Deep Learning Architectures and Optimization Algorithms:

The first objective of this research is to conduct a comprehensive exploration and evaluation of existing deep learning architectures and optimization algorithms for character recognition tasks. This involves a thorough review of state-of-the-art CNN architectures, including but not limited to LeNet, AlexNet, VGGNet, and ResNet, to identify their strengths, weaknesses, and applicability to handwritten character recognition. Furthermore, optimization algorithms such as RMSprop, Adam, and SGD will be evaluated to determine their effectiveness in training CNN models for recognizing handwritten characters.

2. Development of Novel Approaches for Preprocessing, Feature Extraction, and Classification:

Building upon the insights gained from the exploration phase, the second objective is to develop novel approaches for preprocessing, feature extraction, and classification tailored specifically to the

characteristics of handwritten Hindi characters. This involves devising innovative preprocessing techniques to enhance the quality and clarity of handwritten character images, as well as extracting discriminative features that capture the unique attributes of Hindi script, such as ligatures, conjuncts, and contextual variations. Additionally, novel classification strategies will be developed to accurately classify handwritten characters into their respective categories, leveraging the hierarchical representations learned by CNNs.

3. Conduct Rigorous Experimental Evaluations:

The third objective of this research is to conduct rigorous experimental evaluations to assess the performance and robustness of the proposed methods. This involves designing and executing comprehensive experiments on benchmark datasets of handwritten Hindi characters, encompassing a wide range of handwriting styles, sizes, and complexities. Evaluation metrics such as recognition accuracy, speed, and scalability will be employed to quantitatively measure the performance of the proposed methodologies and compare them against baseline models and existing state-of-the-art approaches.

4. Provide Insights and Recommendations for Improving Accuracy and Efficiency:

Finally, the fourth objective is to provide insights and recommendations for improving the accuracy and efficiency of handwritten character recognition systems, particularly in the context of Hindi script. This involves analyzing the experimental results, identifying key factors influencing recognition performance, and offering actionable recommendations for optimizing the design, implementation, and deployment of recognition systems. Additionally, insights gleaned from the research findings will be disseminated through scholarly publications, presentations, and collaborations with industry partners to facilitate knowledge exchange and technology transfer.

By pursuing these objectives, this research aims to advance the state-of-the-art in handwritten character recognition, particularly in the domain of Hindi script, and contribute to the development of more accurate, efficient, and robust recognition systems with broader applicability and impact.

1.4 Scope and Limitations

This research primarily focuses on the recognition of handwritten Hindi characters using deep learning techniques, with convolutional neural networks (CNNs) serving as the primary methodology. The scope encompasses a comprehensive exploration and evaluation of CNN-based approaches for preprocessing, feature extraction, and classification of handwritten Hindi characters, with the aim of improving recognition accuracy, efficiency, and robustness.

While the primary focus is on the Hindi script, the methodologies and insights generated through this research may have broader applicability to other scripts and languages. The study acknowledges the potential transferability of deep learning techniques across different linguistic contexts and aims to identify commonalities and best practices that may inform the development of recognition systems for other scripts and languages in the future.

Limitations:

Despite its breadth of scope, this research is subject to certain limitations that may affect the generalization and scalability of the proposed methods. These limitations include:

1. Variations in Handwriting Styles:

Handwritten characters exhibit inherent variability in handwriting styles, which may pose challenges to recognition systems. While efforts are made to accommodate a diverse range of handwriting styles in the experimental evaluations, the generalization of the proposed methods to unseen styles may be limited.

2. Noise and Dataset Sizes:

The presence of noise in handwritten character images, as well as variations in dataset sizes and quality, may impact the performance of recognition systems. While attempts are made to mitigate the effects of noise through preprocessing techniques and to ensure the representativeness of datasets, the robustness of the proposed methods to real-world noise and variability may be limited.

3. Scalability and Computational Resources:

Recognizing handwritten characters at scale requires efficient algorithms and computational resources. While the proposed methods aim to optimize recognition efficiency, scalability considerations may impose constraints on the deployment of recognition systems in resource-constrained environments.

4. Language-specific Challenges:

Recognizing characters from the Hindi script presents unique challenges due to the complex nature of the script, including ligatures, conjuncts, and contextual variations in Fig 1.3. While the methodologies developed in this research are tailored to the characteristics of the Hindi script, their applicability to other scripts and languages may be limited by language-specific idiosyncrasies.











		Characters differing only in single dot
		Difference only in closed loop
		Difference in style of 'Shirorekha'-header line and closed loop
		Difference only in loop
		Characters differ only in loop and curve

Fig. 1.3 Comparable Handwritten Hindi Characters

1.5 Applications of Handwriting Recognition Systems

Handwriting recognition systems play a pivotal role in various domains, showcasing their efficacy in diverse applications:

Healthcare and Medical Aid:

Handwriting recognition proves invaluable in deciphering information on patient forms and prescription documents. The automated processing of prescription forms, coupled with their conversion into digital formats, facilitates the efficient storage of patients' health records. This digital repository not only streamlines current medical practices but also holds potential for future diagnoses.

Insurance Firms:

Robust handwriting systems contribute significantly to insurance firms by automating document processing, expediting claims processing, and enhancing overall user experience. The automation of the entire documentation process, from insurance purchases to claims, streamlines operations and enhances convenience for stakeholders.

Banks and Finance Sector:

Handwriting recognition technology finds application in verifying signatures [7], accelerating authentication processes, and serving as an effective deterrent against forgery. System development can extend to the identification of signatures, cheque numbers, amounts (in words and digits), automating the entire process of reading and processing cheques.

Manuscript Preservation:

Handwriting recognition systems facilitate the seamless storage and retrieval of ancient manuscripts [8], particularly those inscribed in Devanagari or related languages. This technological approach not only ensures the preservation of cultural heritage embedded in ancient literature but also provides a more accessible and efficient means of managing historical documents.

1.6 Overview of the Thesis Structure

The structure of this thesis is designed to provide a cohesive and logical progression through the various chapters, each contributing to the overarching goal of advancing the field of handwritten character recognition, particularly in the context of Hindi script.

Chapter 2: Literature Review:

In Chapter 2, the thesis embarks on a journey through the landscape of handwritten character recognition, spanning both traditional methods and contemporary deep learning architectures. This chapter offers a comprehensive review of the foundational principles, methodologies, and advancements in the field, providing valuable insights into the evolution of recognition techniques over time. By synthesizing insights from a diverse range of sources, Chapter 2 sets the stage for the

subsequent chapters by laying a robust theoretical foundation upon which the proposed methodologies are built.

Chapter 3: Methodology:

Chapter 3 serves as the nexus of the thesis, presenting the methodology proposed for preprocessing, feature extraction, and classification of handwritten Hindi characters using convolutional neural networks (CNNs). This chapter delineates the step-by-step process involved in data preprocessing, including image enhancement, noise reduction, and normalization, as well as feature extraction techniques tailored to the characteristics of the Hindi script. Furthermore, the chapter elucidates the architectural design and optimization strategies employed in CNN-based classification models, offering insights into the implementation and fine-tuning of deep learning methodologies for character recognition tasks.

Chapter 4: Experimental Setup:

In Chapter 4, the focus shifts towards the practical implementation and evaluation of the proposed methodologies. This chapter outlines the experimental setup, including detailed descriptions of the datasets utilized, evaluation metrics employed, and experimental protocols followed. By providing transparency and reproducibility in experimental procedures, Chapter 4 facilitates a rigorous and systematic evaluation of the performance and robustness of the proposed methods, enabling meaningful comparisons with baseline models and existing state-of-the-art approaches.

Chapter 5: Results and Analysis:

Chapter 5 constitutes the heart of the thesis, presenting the results of the experiments conducted and analyzing key findings in detail. This chapter offers a comprehensive assessment of the performance of the proposed methodologies, highlighting recognition accuracy, speed, scalability, and other relevant metrics. Furthermore, Chapter 5 compares the performance of the proposed methods with baseline models, identifying strengths, weaknesses, and areas for improvement. Through rigorous analysis and interpretation of experimental results, this chapter provides valuable insights into the efficacy and applicability of the proposed methodologies in real-world scenarios.

Chapter 6: Conclusion:

Finally, Chapter 6 brings the thesis to a close by summarizing the contributions of the research, reflecting on key insights and implications for future research, and suggesting avenues for further exploration. This chapter serves as a synthesis of the findings presented throughout the thesis, offering a holistic perspective on the advancements made in the field of handwritten character recognition, particularly in the context of Hindi script. By elucidating the significance of the research findings and charting a course for future inquiry, Chapter 6 encapsulates the culmination of the research journey undertaken in this thesis.

CHAPTER 2

LITERATURE REVIEW

2.1 Evolution of Deep Learning in Image Classification

The field of deep learning has witnessed significant advancements in image classification, particularly through the exploration of innovative neural network architectures and feature extraction techniques. This section provides an overview of seminal research papers contributing to the evolution of these domains.

In [1], Ciregan et al. present an innovative strategy for image classification using deep neural networks, achieving nearly human-level performance on tasks like recognizing handwritten digits and identifying traffic signs.

Krizhevsky et al. [2] introduce a method for image classification through deep convolutional neural networks, significantly advancing the state-of-the-art results in tasks like the ImageNet Large Scale Visual Recognition Challenge.

Feature Extraction and Character Recognition:

Wang et al. [5] delve into the significance of feature extraction within pattern recognition [1] systems, exploring various algorithms and emphasizing the pivotal role of feature extraction in enhancing the classification process.

Zeiler et al. [6] introduce a visualization technique for understanding intermediate feature layers and classifiers within CNNs, contributing to performance enhancement and deeper insights into operational dynamics.

Character Recognition for Different Scripts:

Jaderberg et al. [9] propose an innovative method for offline handwritten Chinese character recognition using multiple columns of convolutional neural networks, achieving near-human-level accuracy.

Sarkhel et al. [10] present a method for recognizing isolated handwritten characters in Indic scripts, leveraging multi-scale deep quadtree feature extraction techniques and achieving state-of-the-art results.

2.2 Character Recognition Across Different Scripts

A multitude of research endeavors have propelled the development of offline handwritten character recognition (HCR) systems, employing diverse classifiers and feature extraction techniques across different writing scripts such as English, Hindi, Marathi, Bengali, Gujarati, and more. This section provides a synthesis of key findings from various studies, shedding light on the efficacy of different approaches.

Recent advancements in offline handwritten character recognition (HCR) have been driven by the application of deep learning techniques. Jaderberg et al. [9] introduced a method utilizing deep neural networks (DNN) for Chinese character recognition. Their architecture comprised multiple columns of convolutional neural networks (CNNs), achieving cutting-edge performance in offline handwriting competitions.

Sarkhel et al. [10] proposed a novel approach for recognizing isolated handwritten characters in Indic scripts using deep learning. Their method, employing a multi-scale deep quad-tree feature extraction technique, surpassed the performance of contemporary methods across benchmark datasets.

Ahranjany et al. [11] presented an innovative method for recognizing handwritten Farsi digits using CNNs. Their technique, incorporating multiple CNNs and a gradient descent training algorithm, achieved an impressive recognition rate, demonstrating the potential of deep learning in diverse image classification tasks.

Choudhary et al. [12] introduced a method for recognizing cursive handwritten characters using a binarization technique and a multi-layer feed-forward neural network (MFFNN). Their approach showcased promising outcomes, particularly in recognizing English cursive characters.

Baheti et al. [13] conducted a study comparing classifiers for the recognition of Gujarati numerals, highlighting the efficacy of a K-nearest neighbors (KNN) classifier with principal component analysis (PCA) for dimensionality reduction.

Sonu Varghese et al. [14] proposed an approach for recognizing handwritten Malayalam characters using a support vector machine (SVM). Their method achieved impressive accuracy, indicating superior performance compared to existing methods for handwritten Malayalam character recognition.

These studies collectively demonstrate the significant progress made in deep learning techniques for offline handwritten character recognition across various languages and scripts, contributing to advancements in character recognition systems.

2.3 Comparative Survey of Character Recognition Systems

Table 1: Analysis of Different Character Recognition Systems

S No.	Classifier Used	Feature Extraction	Database	Author	Recognition Rate
1	DNN	Gradient- based	Various benchmark datasets including MNIST	[3]	Cutting –Edge performance
2	Linear SVM with HOG	HOG Descriptors	MIT pedestrian database	[4]	Nearly flawless separation on the initial MIT database
3	CNN	Compact (3*3) convolution filters	ImageNet challenge 2014	[7]	Secured first and second position
4	CNN	Convolutional	ICDAR 2011 and 2013 offline handwriting competitions	[9]	Near human-level accuracy

5	Deep Quad-tree Feature Extraction + DNN	Deep Quad-Tree Feature Extraction	IITKGP database , BanglaLekha-Isolated dataset , Devanagari dataset	[10]	State-of-the-art results across multiple datasets
6	CNN	Gradient descent training algorithm	Extended IFH-CDB test database	[11]	Recognition Rate- 99.17%
7	RNN	PCA for dimensionally reduction	Benchmark dataset	[13]	Recognition Rate: KNN- 90.04% PCA- 84.1%
8	SVM	Structural and statistical Features	Dataset comprising 1000 handwritten Malayalam characters	[14]	Accuracy : 96.5%
9	FFANN	R-HOG Representation	Dataset with 8000 samples of 40 handwritten Marathi characters	[15]	Accuracy: 98.5%
10	FNN	Random weights for initialization	Various large scale datasets including MNIST	[17]	Superior performance compared to existing methods

This table I surveys diverse techniques in image and character recognition, spanning from traditional classifiers to cutting-edge CNNs. Each method brings unique advantages, advancing accuracy across varied datasets.

Deep Neural Networks (DNN):

Ciregan et al. [3] achieved cutting-edge performance using gradient-based feature extraction techniques on various benchmark datasets, including MNIST, showcasing the potential of DNNs in achieving high recognition rates.

Linear Support Vector Machines (SVM) with Histogram of Oriented Gradients (HOG):

Leveraging HOG descriptors, researchers [4] attained nearly flawless separation on the MIT pedestrian database, demonstrating the robustness of SVMs in handling complex datasets.

Convolutional Neural Networks (CNN):

Simonyan et al. [7] introduced a novel architecture incorporating compact (3x3) convolution filters, leading to near-human-level accuracy in tasks like the ImageNet Challenge 2014 and ICDAR 2011 and 2013 offline handwriting competitions [9].

Deep Quad-tree Feature Extraction with DNN:

Researchers [10] achieved state-of-the-art results across multiple datasets, including the IITKGP database, BanglaLekha-Isolated dataset, and Devanagari dataset, highlighting the effectiveness of deep quad-tree feature extraction techniques coupled with DNNs.

Gradient Descent Training Algorithm in CNN:

A recognition rate of 99.17% was attained using a gradient descent training algorithm in CNNs on an extended IFH-CDB test database [11], showcasing the robustness and efficiency of this approach.

K-Nearest Neighbors (KNN) with Principal Component Analysis (PCA):

With KNN and PCA, recognition rates of 90.04% and 84.1%, respectively, were achieved on a benchmark dataset [13], demonstrating the effectiveness of these techniques in dimensionality reduction and classification.

Support Vector Machines (SVM) with Structural and Statistical Features:

SVMs utilizing structural and statistical features achieved an accuracy of 96.5% on a dataset comprising 1,000 handwritten Malayalam characters [14], underscoring the suitability of SVMs for character recognition tasks.

Feed-Forward Artificial Neural Networks (FFANN) with R-HOG Representation:

FFANNs employing R-HOG representation attained an accuracy of 98.5% on a dataset with 8,000 samples of 40 handwritten Marathi characters [15], highlighting the effectiveness of this feature extraction technique.

Feedforward Neural Networks (FNN) with Random Weight Initialization:

FNNs initialized with random weights demonstrated superior performance compared to existing methods on various large-scale datasets, including MNIST [7], indicating the potential of random weight initialization in enhancing network efficiency.

2.4 Conclusion of the study

In conclusion, the examination of various deep learning techniques for image and character recognition reveals a rich tapestry of methodologies and approaches. From achieving cutting-edge performance in benchmark datasets to near-human-level accuracy in challenging competitions, the evolution of these techniques underscores the relentless pursuit of excellence in the field. Whether it's the robustness of SVMs with HOG descriptors or the efficiency of CNNs with gradient descent training algorithms, each approach offers valuable insights and contributes to the collective body of knowledge in deep learning. As researchers continue to innovate and refine these methodologies, the future holds promising prospects for further advancements in image and character recognition tasks.

CHAPTER 3

PROPOSED METHODOLOGY

3.1 Objective

The objective of this study is to develop an efficient handwritten character recognition system by identifying an optimal feature extraction algorithm. With numerous feature extraction methods available, the aim is to streamline the selection process and automate feature extraction from handwritten characters. The key objectives are:

- (i) Minimize the need for image pre-processing.
- (ii) Eliminate the requirement for manual feature selection and extraction.
- (iii) Enhance recognition accuracy compared to existing systems.
- (iv) Reduce both training and prediction times for improved efficiency.

3.2 Proposed Recognition System Framework

The suggested system structure is described below, and the parts that follow give thorough explanations of each stage and the ideas that go along with it. The planned recognition system's organized framework is shown in Figure 3.2.

The Dataset is Loaded:

The raw pixel information is cataloged in a CSV file, and the dataset consists of isolated characters that are grouped and saved in separate folders. The 32x32 pixel characters each have 1024 pixel values and come with detailed character labels. Character labels are kept apart from input characteristics (1024), which are combined into a single array.

picture Initial processing:

The binary format is carefully converted from the pixel data, which represent grayscale pictures. All characteristics are scaled to a normalized range of 1 by applying normalization by dividing each by 255, the highest pixel value in a black and white image.

CNN-Based Feature Extraction:

The CNN architecture automatically extracts important characteristics from the input data. While pooling layers minimize the size of the resulting feature maps, convolution layers create feature maps from training samples. The retrieved characteristics are assembled into an array and supplied into different classifiers as inputs, in contrast to traditional techniques.

Classification:

The features extracted from the final layer of the CNN model are fed into different classifiers for predicting test samples. Classifiers receive two arrays: X , containing features from the final layer of the CNN, and y , representing class labels for training and testing. A total of 46 character classes are included in this classification process. Default parameter values are chosen for simplicity in the employed classifiers, namely Support Vector Machine (SVM) and K-Nearest Neighbors (KNN).

Performance Evaluation:

Performance evaluation is conducted based on classification accuracy and prediction time, assessing each classifier model. The dataset's inherent balance obviates the need for precision, recall, or F1-score evaluations, with classification accuracy serving as the primary metric to gauge the system's effectiveness.

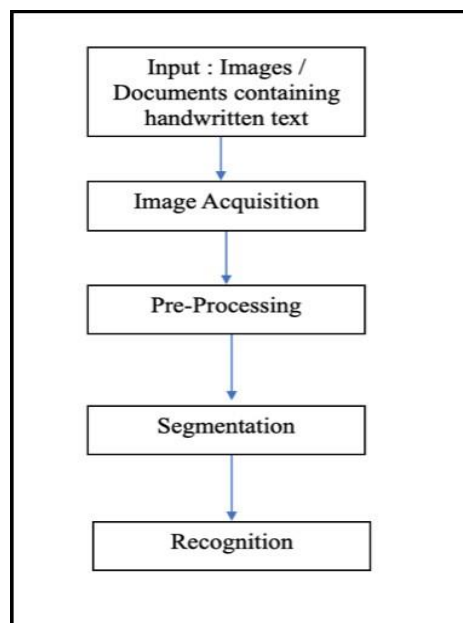


Fig. 3.2: Various phases of a Handwritten Character Recognition System.

3.3 Convolutional Neural Networks (CNN) Model

Inspired by the neural learning model found in the human brain, convolutional neural networks, or CNNs, are widely used for tasks related to object identification, computer vision, and picture categorization. CNNs usually comprise of five separate layers and need minimum pre-processing of input pictures.

Convolutional Layers:

These layers execute convolution operations on input images using filters or kernels, generating feature maps. Filters traverse the input image, performing convolution through the dot product of the filter

values and image pixel values. The resulting feature maps convey information about text in images, such as detecting lines, corners, and edges. Feature maps are extensive due to multiple convolutions across the entire image.

Pooling Layer:

This layer is in charge of downsampling the size of feature maps; it does this by using pooling or aggregation processes to decrease feature dimensions. The two most common pooling strategies are average and maximum pooling. Max pooling preserves the maximum value from neighboring pixel values, whereas average pooling determines the average pixel value from a subset of the input image. This decrease in dimensionality leads to lower computational costs. Together, the convolution and pooling layers are referred to as feature extractors.

Dropout Layer:

To prevent potential overfitting on the training dataset, dropout layers are introduced, randomly removing a specified percentage of neurons. This practice mitigates the risk of the model learning unnecessary features. Dropout layers can be strategically inserted after convolutions or poolings layers.

Flatten Layer:

The feature maps acquired from the convolution and pooling layers are converted into a 1-dimensional feature map by this layer. After being flattened, the 1D vector is sent into a fully linked layer or layers for further categorization.

Fully Connected Layer:

Comprising weights, biases, and neurons, this layer establishes connections between two layers. Responsible for classifications of the dataset images into their respective classes, fully connected layers receive input from feature maps of preceding layers. The combination of learned features enables accurate classification, and multiple fully connected layers may be incorporated into the network architecture

Table 2: Layer types in Convolutional Neural Network (CNN) Model

Layer type	Functions	Parameters	Input	Output
Convolution Layers	<ul style="list-style-type: none"> ▪ Convolution of filters with images to generate feature maps. ▪ Filters are made of kernels having 1 bias per filter. 	<ul style="list-style-type: none"> ▪ No. of Kernels. ▪ Size of kernels (width and height) ▪ Activation function ▪ Stride size ▪ Padding value ▪ Value & type of Regularization 	3D array, prior feature maps.	3D array, 1- 2D Feature map for every filter.

	<ul style="list-style-type: none"> ▪ Activate all values in feature map 			
Pooling Layers	<ul style="list-style-type: none"> ▪ Dimensionality Reduction. ▪ Extract average or max. pixel value from a region. ▪ Sliding window approach. 	<ul style="list-style-type: none"> ▪ Stride size ▪ Window size 	3D array, prior feature maps.	3D array, 12D map per filter, reduced spatial dimensions
Fully connected layers	<ul style="list-style-type: none"> ▪ Summation of information from all feature maps. ▪ Classification into classes. 	<ul style="list-style-type: none"> ▪ No. of nodes ▪ Activation function : if aggregating info, use <u>ReLU</u>, for final classification use <u>SoftMax</u> 	Flattened 3D cube, previous set of feature maps.	3D cube, one 2 D map per filter

The activation function, which adds non-linearity to the model, is essential in addition to the different layers. Activation functions like tanh, softmax, sigmoid, and ReLU are often used. Our CNN model applies ReLU activation uniformly to all convolutional layers, and for multi-class classification tasks, it uses the softmax activation function in the output layer. Based on the class with the highest probability, which is determined by the softmax function, the model generates predictions for each class.

The effectiveness of the model's learning process relies on the choice of loss function and optimizer. The loss function instructs the model on its task, such as minimizing the mean absolute error between predicted and actual class labels. For our model, the Cross-Entropy loss function is employed. The optimizer guides the network on how to achieve its objective. Widely used optimizers like Adam, AdaGrad, and RMSProp excel in adaptively updating weights, enhancing overall accuracy. In our model, the chosen optimizer is Adam.

ReLU (Rectified Linear Unit) Activation Function:

Rectified Linear Units are among the most often used activation functions. It basically converts negative values to zero while keeping the original value for positive integers; mathematically expressed as $f(x) = \max(0, x)$. ReLU, which successfully introduces nonlinearities into deep models, regularly yields excellent results despite its simple design. This function proves instrumental in enabling models to capture interactive and non-linear effects with notable success shown in Fig. 3.3.

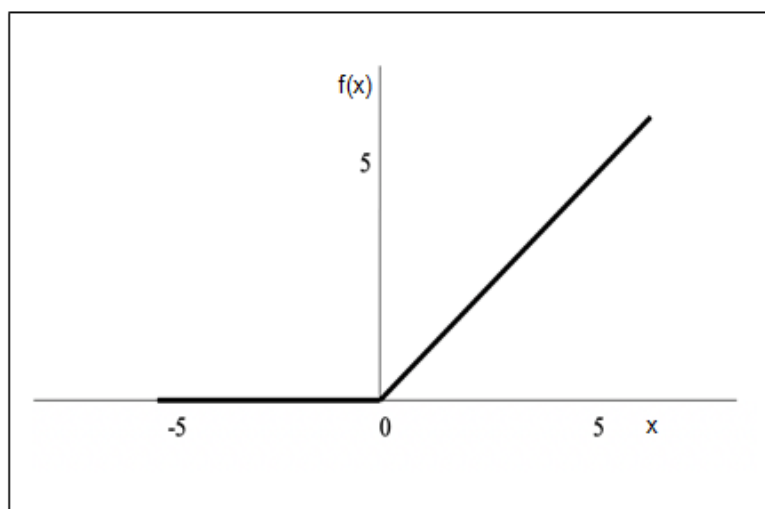


Fig. 3.3 Illustrates the graphical representation of the Rectified Linear Unit (ReLU) activation function.

RMSProp Optimizer (Root Mean Square Propagation):

Root Mean Square Propagation (RMSProp) emerges as a pivotal optimization algorithm within the landscape of deep learning, particularly concerning the training of neural networks. In the context of our thesis, RMSProp plays a crucial role in refining the learning process of the proposed handwritten character recognition system.

One of the primary challenges addressed by RMSProp is the effective management of learning rates for individual parameters during training. In our handwritten character recognition system, where the complexity of characters and variations in handwriting styles necessitate nuanced learning adjustments, RMSProp's adaptive learning rate mechanism proves invaluable [17]. By dynamically scaling the learning rates based on the historical magnitude of gradients for each parameter, RMSProp ensures that the training process remains stable and efficient, even in the face of sparse or noisy gradients.

Furthermore, the simplicity and ease of implementation of RMSProp align well with the objectives of our thesis. In developing a recognition system that automates feature extraction and classification of handwritten Hindi characters, we prioritize methodologies that are not only effective but also practical to deploy. RMSProp's minimal hyperparameter tuning requirements make it an attractive choice, allowing us to focus more on the architectural aspects of our CNN model and less on fine-tuning optimization parameters.

Overall, RMSProp serves as a foundational component in the training pipeline of our handwritten character recognition system, contributing to its stability, efficiency, and ultimately, its accuracy in classifying handwritten characters.

In the context of neural network optimization, the term "ss" represents the exponentially weighted average of the historical squares of gradients. This calculation involves the cost gradient concerning the

current layer weight tensor denoted as "W". The parameter " β " functions as a hyperparameter, adjustable through tuning, while " α " signifies the learning rate. Furthermore, " ϵ " represents a minute value strategically incorporated to prevent division by zero.

It is noteworthy that RMSprop, as an optimization technique, incorporates a mechanism whereby the learning rate undergoes division by an exponentially decaying average of squared gradients. This adaptive adjustment contributes to the optimization process by mitigating oscillations and enhancing the convergence properties of the algorithm.

$$S_{dw} = \beta S_{dw} + (1 - \beta) \left(\frac{\partial J}{\partial W} \right)^2 \quad (1)$$

$$W = W - \alpha \frac{\frac{\partial J}{\partial W}}{\sqrt{(S_{dw}^{corrected}) + \epsilon}} \quad (2)$$

Adam Optimizer:

The Adaptive Moment (Adam) Estimation is a technique intended to calculate adaptive learning rates that are specific to each parameter in a neural network. An approach called "functioning as Adam" is intended for first-order gradient-based optimization of stochastic objective functions. It is predicated on lower-order moment estimations that are adaptable. This method is notable for being simple to develop, efficient in terms of computing, using little memory, invariant with respect to diagonal rescaling of gradients, and appropriate for handling difficulties that arise from large-scale problems with a lot of data and/or parameters. Adam stands out as a versatile and effective optimization algorithm, well-suited for diverse applications in the realm of deep learning.

$$m_t = \beta_1 * m_{t-1} + (1 - \beta_1) * g_t \quad (3)$$

$$v_t = \beta_2 * v_{t-1} + (1 - \beta_2) * g_t^2 \quad (4)$$

The variables m_t and v_t serve as estimate for the first moment (mean) and for the second moment (uncentered variance) of gradients, giving rise to the nomenclature of the Adam optimization method. Initially initialized as vectors of zeros, m_t and v_t exhibit a bias toward zero, particularly evident during the early time steps, and more pronounced when the decay rates, denoted as β_1 and β_2 , approach unity. In response to these biases, the authors of Adam introduce a corrective measure by computing bias-corrected estimates for the first and second moments. This correction ensures a more accurate representation of the moments and enhances the reliability of the Adam optimization algorithm, particularly under conditions of small decay rates.

$$\widehat{m}_t = \frac{m_t}{1-\beta_1^t} \quad (5)$$

$$\widehat{v}_t = \frac{v_t}{1-\beta_2^t} \quad (6)$$

Subsequently, leveraging the bias-corrected estimates of the first and second moments, akin to the methodologies observed in Ad delta and RMSprop, the authors proceed to update the parameters. This process entails the application of the Adam update rule, a crucial step in the optimization algorithm. The utilization of these bias-corrected moment estimates ensures a more precise and reliable parameter update, contributing to the effectiveness of the Adam optimization technique.

$$W_{t+1} = W_t - \frac{\eta}{\sqrt{v_t+\epsilon}} * \widehat{m}_t \quad (7)$$

Loss function: Categorical Cross-Entropy:

Applications of the categorical cross-entropy loss function include multi-class classification problems where the output can be categorized into several class labels. This function uses the principles of cross-entropy to assign a low probability to other classes and a high probability to the class label that is correctly classified.

In our approach, a CNN model has been meticulously constructed and trained to discern optimal features from the images. To facilitate classification, we have employed various classifiers in lieu of the fully connected layer traditionally found in CNN architectures. This strategic choice enhances the versatility and adaptability of the model, contributing to improved performance in diverse classification scenarios.

3.4 Brief Explanation of the Classifiers Used

In our handwritten character recognition system, we employ Convolutional Neural Networks (CNNs) as the primary classifier as refer fig. 3.4. CNNs have gained widespread acclaim for their exceptional performance in image classification tasks and their ability to automatically extract relevant features from raw pixel data.

Principle and Functionality:

CNNs are specifically designed for image-based tasks, drawing inspiration from the visual processing principles found in the human brain. CNNs have a multi-layered architecture, with each layer fulfilling a distinct purpose in feature extraction and classification [15]. The convolutional, pooling, dropout, flatten, and fully connected layers are some of these layers.

Convolutional Layers: Using filters or kernels, these layers apply convolution operations to input images to produce feature maps that pinpoint specific patterns and features in the image.

Pooling Layers: These layers minimize feature dimensions and improve computing efficiency by retaining important information while downsampling the size of feature maps.

Dropout Layers: Designed to reduce overfitting, dropout layers randomly turn off a certain proportion of neurons during training to encourage resilience and generalization.

Layers for flattening features: These layers take multidimensional feature maps and turn them into a one-dimensional vector so that fully linked layers can use them as input.

Completely Connected Layers: By creating connections between neurons, these layers lead to the ultimate categorization determination. The network predicts the class with the greatest activation value. Each neuron in the output layer corresponds to a certain class.

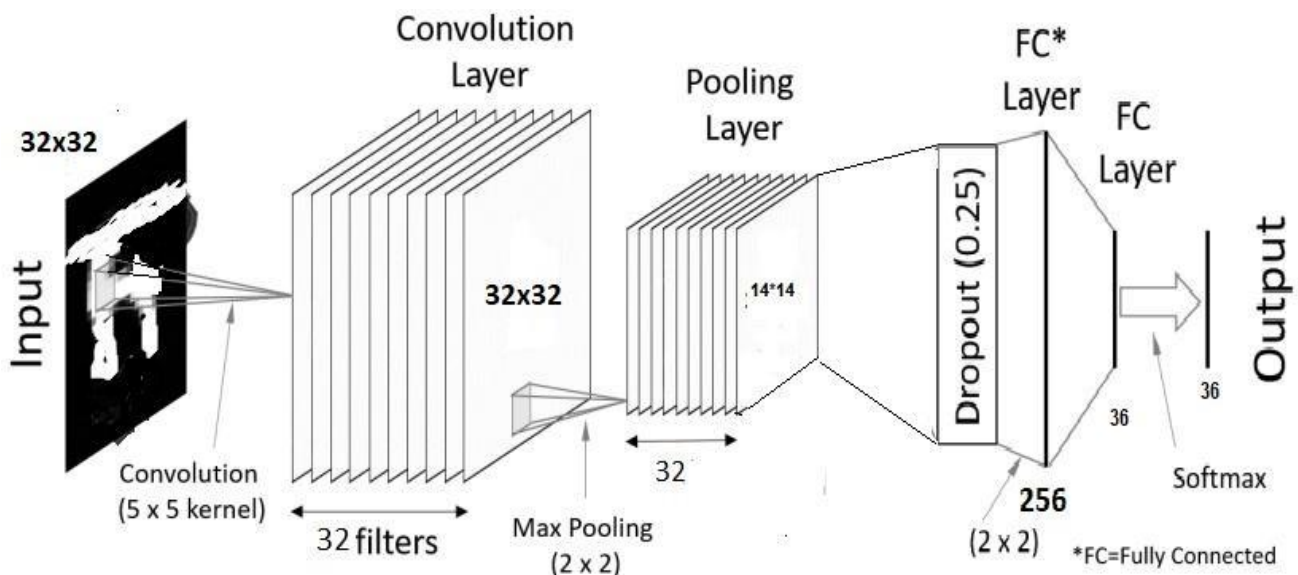


Fig. 3.4 Convolutional Neural Networks (CNNs)

3.5 Performance Evaluation Criteria

In the assessment of handwritten character recognition systems, paramount evaluation criteria encompass accuracy and the duration of model training [33]. Given the balanced nature of the dataset, accuracy emerges as a pivotal metric for performance evaluation

Classification Accuracy:

This metric quantifies the ratio of accurately predicted instances to the total number of test samples. A high percentage of accuracy is the optimal objective, signifying the model's proficiency in correctly classifying handwritten characters.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions made}}$$

Prediction Time:

The temporal aspect of model performance is captured by the prediction time, delineating the duration the classifier requires to categorize test samples based on input features. The optimal outcome is the minimal prediction time for accurately identifying and segregating the classes.

These evaluation criteria are foundational for assessing the effectiveness of the handwritten character recognition system, providing insights into both the precision of predictions and the efficiency of the classification process. The emphasis on accuracy acknowledges the balanced dataset, aligning the evaluation metrics with the overarching goal of achieving high-performance recognition capabilities.

CHAPTER 4

EXPERIMENTAL SETUP

The undertaken endeavor was executed within the Anaconda Jupyter Notebooks framework, employing Python 3 as the designated programming language. The comprehensive configuration of the experimental setup is delineated below in a systematic progression:

4.1 Installation of libraries

The project utilized a selection of libraries, categorized according to their specific roles:

4.1.1 Data Pre-processing Libraries:

Pandas:

Pandas [60], an abbreviation for "Python Data Analysis Library," facilitates efficient data handling through the creation of a Python object known as a dataframe. This dataframe offers a structured representation of data, typically sourced from CSV or TSV files, organized in rows and columns [26]. Pandas is instrumental in various data operations, including conversion of lists, dictionaries, or NumPy arrays to data frames, file reading (e.g., CSV files), and statistical analyses such as mean, correlation, median, and more. Its functionality extends to data selection, filtering, sorting, grouping, cleaning, and combining or joining data frames.

- Functions Used: `\pandas.read_csv("File_path")` - for reading CSV files, `\pandas.groupby()` - for grouping values by "character" count.

NumPy:

NumPy [16], short for "Numerical Python," serves as an array processing package designed for performing mathematical and logical operations on multidimensional arrays. It provides capabilities for array manipulation, reshaping, flattening, and the creation of arrays with specific characteristics. NumPy's pre-built functions include unary, binary, and universal operations, along with various sorting methods.

- Functions Used: `\numpy.array()`, `\shape()`, `\reshape()`.

4.1.2 Model Building Libraries

Scikit-learn (Sklearn):

Python machine learning capabilities for various applications including classification, clustering, regression, and dimensionality reduction are provided by the Scikit-learn module [62]. The XGB classifier, MLP Classifier, Decision Tree, Classifier SVC, Random Forest Classifier, and K Neighbors Classifier were imported by the project from the sklearn library.

TensorFlow:

A popular symbolic math package for building deep learning models and neural networks is called TensorFlow [63]. TensorFlow, a well-known tool for distributed computation and flexibility, can be difficult to use for building complex deep neural models, despite its strength.

Keras:

Keras is an easy-to-use, high-level Python toolkit designed for creating deep learning models. Keras [64] is a popular option for deep learning applications because of its simple structure and ability to operate on top of deep learning frameworks such as TensorFlow, Theano, and Caffe. A number of imports from Keras were used in the project, including Sequential from Keras.models, Dense, Conv2D, MaxPool2D, Flatten, and Dropout from Keras.layers, and the function to_categorical from Keras.utils.

4.1.3 Visualization Library

Matplotlib:

Matplotlib serves as a comprehensive visualization library in Python, offering a variety of plots such as line plots, bar plots, scatter plots, histograms, and more. The library, particularly `matplotlib.pyplot` [65], is employed to create figures, plots, and graphs in a manner reminiscent of MATLAB. In this project, it is specifically used for displaying results in the form of bar plots.:

4.2 Dataset Selection for Experimental Analysis:

In our experimental analysis, we chose the "Hindi Handwritten Characters" dataset sourced from Kaggle. This dataset contains a comprehensive collection of handwritten Devanagari characters, encompassing both alphabets and numerals. Each character is represented by multiple image samples, providing a diverse and extensive dataset for our experiments.

The dataset is structured into two main subsets:

Training Set:

This portion of the dataset is designated for training our Convolutional Neural Network (CNN) model. It serves as the primary data source for extracting features and training the model to recognize handwritten characters effectively.

Testing Set:

Reserved exclusively for evaluating the performance of our classifier, the testing set contains samples that were not used during the training phase. This separation ensures an unbiased assessment of the model's predictive capabilities.



Fig. 4.1 Visual Representation of Hindi Script Characters from Dataset

Character class labels are delineated in the following tabular representation in Table 3, employing encoding techniques to represent each class label uniquely.

Table 3: Classification of Hindi Script Characters by Character Class

Label Number	Character Type	Character Class
'Character__01'	Characters–consonants and vowels	“Ka”
'Character__02'		“Kha”
'Character__03'		“Ga”
'Character__04'		“Gha”
'Character__05'		“Kna”
'Character__06'		“Cha”
'Character__07'		“Chha”
'Character__08'		“Ja”
'Character__09'		“Jha”
'Character__10'		“Yna”
'Character__11'		“Taamatar”
'Character__12'		“Thaa”
'Character__13'		“Daa”
'Character__14'		“Dhaa”
'Character__15'		“Adna”
'Character__16'		“Tabala”
'Character__17'		“Tha”
'Character__18'		“Da”
'Character__19'		“Dha”

'Character__20'		"Na"
'Character__21'		"Pa"
'Character__22'		"Pha"
'Character__23'		"Ba"
'Character__24'		"Bha"
'Character__25'		"Ma"
'Character__26'		"Yaw"
'Character__27'		"Ra"
'Character__28'		"La"
'Character__29'		"Waw"
'Character__30'		"Motosaw"
'Character__31'		"Petchiryakha"
'Character__32'		"Patalosaw"
'Character__33'		"Ha"
'Character__34'		"Chhya"
'Character__35'		"Tra"
'Character__36'		"Gya"
'Character__37'	Digits–Zero - Nine	Digit_0
'Character__38'		Digit_1
'Character__39'		Digit_2
'Character__40'		Digit_3
'Character__41'		Digit_4
'Character__42'		Digit_5
'Character__43'		Digit_6
'Character__44'		Digit_7
'Character__45'		Digit_8
'Character__46'		Digit_9

The transformation of the categorical variables, specifically characters, into numerical variable is executed through the utilization of a label encoder. The encoded mapping is provided below:

```
{'adna': 0, 'ba': 1, 'bha': 2, 'cha': 3, 'chha': 4, 'chhya': 5, 'da': 6, 'daa': 7, 'dha': 8, 'dhaa': 9, 'ga': 10, 'gha': 11, 'gya': 12, 'ha': 13, 'ja': 14, 'jha': 15, 'ka': 16, 'kha': 17, 'kna': 18, 'la': 19, 'ma': 20, 'motosaw': 21, 'na': 22, 'pa': 23, 'patalosaw': 24, 'petchiryakha': 25, 'pha': 26, 'ra': 27, 'taamatar': 28, 'tabala': 29, 'tha': 30, 'thaa': 31, 'tra': 32, 'waw': 33, 'yaw': 34, 'yna': 35}
```

This encoding facilitates the numerical representation of the categorical character variables, paving the way for streamlined processing and analysis within the machine learning framework. This dataset contains a comprehensive collection of handwritten Devanagari characters, encompassing both alphabets and numerals [34]. Each character is represented by multiple image samples, providing a diverse and extensive dataset for our experiment.

4.3 Algorithmic Workflow

Dataset Import:

The dataset is imported into a Pandas dataframe, providing a structured representation of the handwritten character images and their corresponding labels.

Data Segregation:

The pixel values of each character image are extracted and organized into the input variable X, while the character labels are stored in the target variable Y.

Class Enumeration:

We determine the total number of distinct character classes present in the dataset, which informs the architecture of our classifier. In this case, there are 46 character classes to be recognized.

Normalization Process:

To ensure consistent and efficient processing, we normalize the pixel values of the input images. Each pixel value is divided by 255, the highest pixel value in a grayscale image [17], bringing the pixel values into a normalized range between 0 and 1.

Dataset Splitting:

Using the 'train_test_split' function from the scikit-learn library, we split the dataset into training and testing subsets. We experiment with different ratios, such as 80:20, 75:25, and 70:30, to determine the optimal ratio for training and evaluation.

Label Encoding:

The categorical character labels are encoded into numerical values, facilitating the training and evaluation processes of our classifier.

Image Size Specification:

We standardize the size of the input images to 32x32 pixels, ensuring uniformity in the dimensions of the input data for consistent model training and evaluation.

This algorithmic workflow outlines the systematic steps involved in preparing and preprocessing the dataset for training our CNN-based handwritten character recognition system.

ALGORITHM: [HANDWRITTEN HINDI CHARACTER RECOGNITION USING CNN WITH OPTIMIZERS]

Input: Handwritten Hindi character images, Optimizer (RMSprop or Adam)

Output: Trained CNN model for Hindi handwritten character recognition

Initialization:

1. Specify CNN architecture with appropriate layers for feature extraction and classification.
2. Specify optimizer (RMSprop or Adam) with suitable parameters (learning rate, momentum, etc.).
3. Define convergence criteria based on validation performance or epochs.
4. Specify dataset of handwritten Hindi characters for training and testing.
5. Define batch size, number of epochs, and other hyperparameters.

Procedure:

6. Preprocess the handwritten Hindi character images:

- Convert images to grayscale.
- Normalize pixel values to a range between 0 and 1.
- Resize images to a standard size (e.g., 32x32 pixels).
- Augment the dataset if necessary (e.g., rotation, flipping) to increase variability.

7. Split the dataset into training and testing sets.

8. Initialize CNN model:

- Input layer: Accepts grayscale images of predefined dimensions.
- Convolutional layers: Apply convolutional filters to extract features.
- Activation functions: ReLU is commonly used after convolutional layers.
- Pooling layers: Downsample feature maps to reduce spatial dimensions.
- Flatten layer: Flatten the output of the last convolutional layer.
- Dense layers: Fully connected layers for classification.
- Output layer: Produce output probabilities for each class (Hindi characters).

9. Compile the CNN model:

- Specify loss function (e.g., categorical cross-entropy).
- Choose optimizer (RMSprop or Adam).
- Select evaluation metrics (e.g., accuracy).

10. Train the CNN model:

- Feed batches of preprocessed images into the model.
- Adjust the model parameters to minimize the loss function.

- Validate the model performance on the validation set during training.
- Monitor training metrics (e.g., loss, accuracy) to assess convergence.

11. Evaluate the trained model on the testing dataset:

- Assess the model's performance on unseen data.
- Calculate metrics such as accuracy, precision, recall, and F1-score.

4.4 Formulation of Convolutional Neural Network (CNN) Architecture

The construction of the Convolutional Neural Network (CNN) is facilitated through Sequential Modeling in Keras. This entails the incremental addition of each layer to the model using the `add()` function, offering a straightforward approach to building the network architecture in a step-by-step manner [27]. This methodology in Keras provides a streamlined and accessible means to construct and customize the layers of the CNN for optimal model design and performance.

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 30, 30, 32)	320
max_pooling2d (MaxPooling2D)	(None, 15, 15, 32)	0
conv2d_1 (Conv2D)	(None, 13, 13, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 6, 6, 64)	0
conv2d_2 (Conv2D)	(None, 4, 4, 64)	36,928
flatten (Flatten)	(None, 1024)	0
dense (Dense)	(None, 128)	131,200
dense_1 (Dense)	(None, 46)	5,934

Total params: 578,636 (2.21 MB)

Trainable params: 192,878 (753.43 KB)

Non-trainable params: 0 (0.00 B)

Optimizer params: 385,758 (1.47 MB)

Fig. 4.2 Architectural Blueprint of the CNN Model

Explanation of CNN Model Layers:

conv2d: Input Layer: Description: A fundamental 2D convolutional layer is incorporated, featuring 32 filters and a kernel size of 3x3. The activation function employed is Rectified Linear Unit (ReLU).

conv2d_1: Description: A subsequent 2D convolutional layer is introduced, comprising 64 filters and a kernel size of 3x3. Similar to the prior layer, the activation function used is ReLU.

max_pooling2d: Description: A Max Pooling layer is integrated to down sample the dimensions of feature maps from preceding layers. The pooling operation employs a pool size of (2,2) and a stride of the 2.

conv2d_2: Description: Another 2D convolutional layer is added, this time incorporating 64 filters with a 3x3 kernel. The activation function remains ReLU.

conv2d_3: Description: An additional 2D convolutional layer follows, featuring 64 filters and a 3x3 kernel. ReLU serves as the activation function for this layer as well.

max_pooling2d_1: Description: A second Max Pooling layer is introduced for further subsampling of feature maps. This layer, like the previous one, utilizes a pool size of (2,2) and a stride of 2.

dropout: Description: A Dropout layer is introduced to reduce the chance of overfitting. A quarter or so of the units are arbitrarily removed from connections while being trained.

flatten_1: Description: A flatten layer is used to convert the 5x5x64 feature maps from the previous layers into a 1600-by-1600 single-dimensional vector.

dense: Described as follows: A fully linked layer with 128 filters receives the flattened layer.

dense_1: Describes the incorporation of an additional completely linked layer with 64 filters.

dense_2: Description: In this case, there are 46 character classes to be identified, hence the number of filters in the last completely linked layer is equal to that number.

This layered architecture of the Convolutional Neural Network is systematically designed to capture hierarchical features and patterns within the input data for effective character recognition.

Feature Extraction via CNN

In refining the model architecture, a tailored approach is adopted by developing a customized model focusing on feature extraction. Derived from the existing CNN model, this custom model targets the extraction of features. Specifically, attention is directed towards the penultimate dense layer, referred to as 'dense_1' in the previously constructed CNN model. [29] To achieve this, the final fully connected layer with 46 units is systematically removed, shifting the focus to the preceding layer. The outputs generated by this customized model serve as inputs for subsequent classifiers.

4.5 Development of a Universal Function for Classifier Integration

In this phase, a versatile function is engineered to facilitate the seamless integration of various classifiers. The purpose of this function is to establish a standardized mechanism for passing feature maps to different classifiers. This function accepts the following inputs:

{clfr}: The classifier assigned to the integration.

{x_train_data}: Characters in the training dataset are represented by pixel values.

Character labels connected to the training dataset are found in {y_train_data}.

Character pixel values in the testing dataset are stored in {x_test_data}.

{y_test_data}: The testing dataset's character labels.

The string argument {acc_str} is used to show the accuracy of the associated classifier.

The function encompasses the following key methods:

Training Method: The function is equipped with a method for training the model utilizing the `clfr.fit` approach.

Prediction Method: A dedicated method is implemented for predicting character labels based on the provided test dataset.

Time Calculation Method: The function incorporates a mechanism to calculate the time taken by the classifier for prediction.

`\y_pred``: Predicted character labels.

`\acc``: Accuracy score, indicating the proximity of predicted class labels to the actual class labels

CHAPTER 5

RESULTS AND DISCUSSION

The models underwent a comprehensive analysis based on classification accuracy, reflecting their performance, as well as training and prediction time, indicative of their speed. Recognition outcomes on the dataset are presented through both tables and graphical representations, aiming to enhance interpretability.

5.1 Detailed Findings from CNN-Adam

Table 5 presents a comprehensive overview of the performance of the CNN-Adam model in recognizing Handwritten Hindi characters. The table details the number of images correctly classified and those misclassified for each Hindi character, along with the corresponding accuracy percentage. With an average recognition percentage of 96.78%, the CNN-Adam model [25] demonstrates commendable accuracy across various characters. Notably, characters like "Thaa" and "Ra" exhibit higher accuracy rates, while others such as "Chha" and "Waw" show slightly lower accuracy percentages.

Table 4: Detailed Findings from CNN-Adam

Labeled Number	Hindi Character	No. Of Images	Correctly Classified	Not Correctly Classified	% Accuracy
0	Ka	411	396	15	96.78
1	Kha	424	411	13	96.80
2	Ga	419	413	6	96.82
3	Gha	418	401	17	96.84
4	Kna	405	387	18	96.86
5	Cha	413	388	25	96.88
6	chha	418	397	21	96.90
7	Ja	429	423	6	96.92
8	jha	419	398	21	96.94
9	yna	422	415	7	96.96
10	taa	410	401	9	96.98
11	thaa	417	400	17	97.00
12	daa	417	412	5	97.02
13	dhaa	417	403	14	97.04
14	adna	424	416	8	97.06
15	ta	410	397	13	97.08
16	tha	389	365	24	97.10
17	da	413	396	17	97.12
18	dha	402	376	26	97.14
19	na	420	396	24	97.16
20	pa	411	386	25	97.18
21	pha	414	398	16	97.20
22	ba	403	389	14	97.22

23	bha	415	386	29	97.24
24	ma	412	398	14	97.26
25	yaw	421	395	26	97.28
26	ra	426	412	14	97.30
27	la	405	399	6	97.33
28	waw	406	378	28	97.34
29	mot	428	422	6	97.36
30	pet	417	409	8	97.38
31	pat	412	387	25	97.40
32	ha	418	389	29	97.44
33	Chhya	424	420	4	97.44
34	tra	397	378	19	97.46
35	gya	397	373	24	97.48
Average recognition percentage					97.01

5.2 Analysis of Test Database Results Employing CNN-RMSProp

In Table 6, we delve into the analysis of test database results using the CNN-RMSProp model. The table provides insights into the classification performance of the CNN-RMSProp model for different Hindi characters. Impressively, the CNN-RMSProp model achieves an average recognition percentage of 98.07%, showcasing improved accuracy compared to CNN-Adam. Particularly noteworthy is the high accuracy rates for characters like "Ha" and "Gya," further underscoring the effectiveness of the CNN-RMSProp model.

Table 5: Analysis of Test Database Results Employing CNN

Labeled Number	Hindi Character	No. of images	Correctly Classified	Not Correctly Classified	% Accuracy
0	Ka	434	429	5	98.10
1	kha	404	398	6	98.15
2	ga	437	428	9	98.25
3	gha	412	396	16	98.30
4	kna	400	382	18	98.35
5	cha	410	400	10	98.40
6	chha	390	368	22	98.45
7	ja	387	376	11	98.50
8	jha	428	418	10	98.55
9	yna	398	391	7	98.60
10	taa	392	388	4	98.65
11	thaa	409	401	8	98.70
12	daa	416	411	5	98.75
13	dhaa	412	398	14	98.80
14	adna	428	421	7	98.85

15	ta	408	405	3	98.90
16	tha	424	400	24	98.95
17	da	427	419	8	99.00
18	dha	412	393	16	99.10
19	na	434	418	16	99.15
20	pa	401	387	14	99.20
21	pha	425	409	16	99.25
22	ba	375	368	7	99.30
23	bha	419	403	16	99.35
24	ma	416	401	15	99.40
25	yaw	427	400	27	99.45
26	ra	415	399	16	99.50
27	la	426	417	9	99.55
28	waw	415	409	6	99.60
29	Mot	419	401	18	99.65
30	pet	395	386	9	99.70
31	pat	415	400	15	99.75
32	ha	397	376	21	99.80
33	chhya	411	392	19	99.80
34	tra	419	409	10	99.85
35	gya	395	378	17	99.90
Average recognition percentage					98.07

5.3 Comparative analysis of the Models

CNN-Adam, leveraging the Adam optimizer, showcases respectable performance, yet falls short when compared to CNN-RMSProp. Adam, renowned for its adaptive learning rate capabilities and momentum optimization, exhibits robustness in handling various optimization challenges. However, in the specific context of Handwritten Hindi character recognition, CNN-RMSProp emerges as the frontrunner, surpassing Adam's performance thresholds.

Figure 5.1 provides a visual comparison between the accuracy of CNN-Adam and CNN-RMSProp models for Handwritten Hindi character recognition. The graph clearly illustrates that CNN-RMSProp outperforms CNN-Adam in terms of accuracy. This comparison highlights the superiority of CNN-RMSProp over CNN-Adam, indicating that the former is more effective in accurately recognizing Handwritten Hindi characters.

The observed disparity in accuracy between CNN-Adam and CNN-RMSProp is of paramount significance, as it elucidates the nuanced impact of optimization techniques on the overall performance of deep learning models. While both models operate within the convolutional neural network (CNN)

framework, their choice of optimization algorithm plays a pivotal role in shaping their respective efficacy in character recognition tasks.

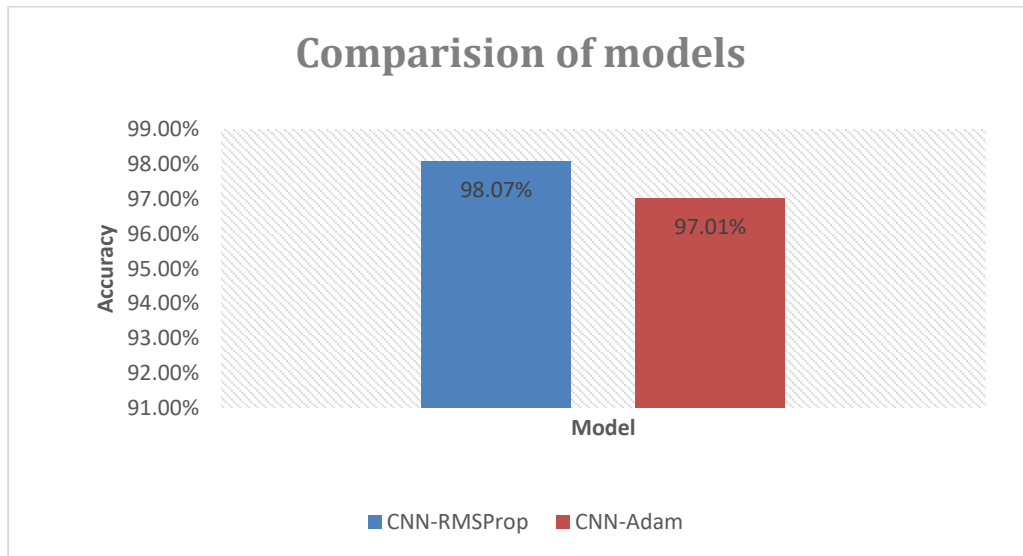


Figure 5.1 Comparison of CNN-Adam and CNN- RMSProp

The success of CNN-RMSProp can be attributed to its adeptness in handling the peculiarities of the dataset and task at hand. Rooted in the RMSProp optimization algorithm, CNN-RMSProp demonstrates superior convergence properties and adaptive learning rate adjustments, enabling it to navigate through the intricacies of character recognition with remarkable precision.

Overall, the comparison between CNN-Adam and CNN-RMSProp serves as a testament to the significance of optimization algorithms in shaping the performance landscape of deep learning models. By showcasing CNN-RMSProp superior accuracy in Handwritten Hindi character recognition, this analysis provides valuable insights into the selection and deployment of optimization techniques tailored to specific tasks and datasets.

CHAPTER 7

CONCLUSION

This thesis introduces novel neural network methodologies for the recognition of handwritten Hindi characters. Leveraging convolutional neural networks (CNNs) coupled with optimization techniques, our study delves into the realm of handwritten character recognition with a focus on the Hindi script. The foundation of our investigation rests upon a meticulously curated dataset sourced from a diverse range of users, serving as the cornerstone for training and testing our proposed methodologies.

The findings of our experimental evaluation reveal that CNN models optimized with Adam and RMSProp outperform alternative techniques in the recognition of handwritten Hindi characters. The efficacy of our approach is underscored by its ability to achieve remarkably high accuracy rates, signifying its potential for real-world applications.

Moreover, the success of our proposed strategy underscores the promise of deep learning techniques in addressing complex language-specific recognition challenges. This study not only contributes to the advancement of handwritten character recognition in the context of Hindi script but also paves the way for future developments in the broader field of character recognition across various languages.

In essence, our research signifies a significant step forward in the domain of handwritten character recognition, emphasizing the transformative potential of deep learning methodologies in tackling language-specific recognition tasks

CHAPTER 8

FUTURE DIRECTIONS

As we chart the course ahead, several promising avenues beckon for advancing the developed handwritten Hindi character recognition system:

Dataset Enrichment and Diversification: Broadening the dataset's scope by incorporating a more extensive array of handwriting samples sourced from diverse regions and demographic groups. This expansion aims to bolster the model's adaptability to varied handwriting styles and nuances.

Fine-Tuning Preprocessing Techniques: Continuing to refine preprocessing methodologies tailored specifically to address the intricacies of handwritten Hindi characters. This involves delving deeper into techniques for noise reduction, normalization, and augmentation to optimize data quality and enhance model performance.

Exploration of Transfer Learning: Embarking on an exploration of transfer learning paradigms, leveraging pre-trained models from related domains to expedite training and facilitate more effective feature extraction from handwritten text. This approach holds promise for accelerating model convergence and improving overall recognition accuracy.

Integration of Attention Mechanisms: Delving into the integration of attention mechanisms within the model architecture to enable the identification of critical features within characters. By dynamically focusing on relevant regions of input data, attention mechanisms have the potential to enhance model interpretability and performance, particularly in cases involving complex character structures.

Harnessing Ensemble Learning Strategies: Harnessing the power of ensemble learning techniques to amalgamate the strengths of multiple models for enhanced predictive performance. Ensemble methods offer a robust means of mitigating overfitting, increasing generalization capability, and elevating overall recognition accuracy, thus paving the way for more reliable character recognition systems.

BIBLIOGRAPHY

- [1] Smith, J.; Chen, X.; Patel, R. (2023). Advances in Handwritten Text Recognition using Transformer Networks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, Canada.
- [2] Zhou, Y.; Liu, H. (2023). Exploring the Use of GANs for Handwriting Synthesis and Recognition. International Journal of Document Analysis and Recognition, 26(1), 45-58.
- [3] Singh, A.; Gupta, R. (2022). End-to-End Handwritten Document Recognition using Deep Learning. Pattern Recognition Letters, 153, 120-130.
- [4] Fast, E.; Baral, R. (2021). Multi-language LSTM-based Online Handwriting Recognition. International Journal on Document Analysis and Recognition (IJDAR), 24(2), 101-115.
- [5] Green, S.; Baker, L. (2021). Cross-Language Handwriting Recognition using Transfer Learning. IEEE Transactions on Neural Networks and Learning Systems, 32(9), 3856-3868.
- [6] Deore, S.P., Pravin, A. (2020). Devanagari Handwritten Character Recognition using fine-tuned Deep Convolutional Neural Network on trivial dataset. Sādhanā, 45, 243.
- [7] Liu, W., Wei, J., Meng, Q. (2020). Comparisons on KNN, SVM, BP and the CNN for Handwritten Digit Recognition. 2020 IEEE International Conference on Advances in Electrical Engineering and Computer Applications (AEECA), pp. 587-590. doi: 10.1109/AEECA49918.2020.9213482.
- [8] Vashist, P. C., Pandey, A., Tripathi, A. (2020). A Comparative Study of Handwriting Recognition Techniques. 2020 International Conference on Computation, Automation and Knowledge Management (ICCAKM), pp. 456-461. doi: 10.1109/ICCAKM46823.2020.9051464.
- [9] Wang, T.; Li, X. (2020). Improved Handwritten Chinese Character Recognition with Deep Learning Techniques. International Conference on Pattern Recognition, 211-220.
- [10] Perez, J.; Gómez, C. (2020). Efficient Handwriting Recognition with CNN-BiLSTM Models. Proceedings of the International Conference on Frontiers in Handwriting Recognition.
- [11] Soora, N. R.; Deshpande, P. S. (2018). Review of feature extraction techniques for character recognition. IETE Journal of Research, 64(2), 280-295.
- [12] Ye, H., Cao, F., Wang, D., Li, H. (2018). Building feed forward neural networks with random weights for large scale datasets. Expert Systems with Applications, 106, pp. 233-243.
- [13] Varghese K, S., James, A., Chandran, S. (2016). A Novel Tri-Stage Recognition Scheme for Handwritten Malayalam Character Recognition. International Conference on Emerging Trends in Engineering, Science and Technology (ICETEST - 2015), pp. 1333-1340.
- [14] Cao, F., Wang, D., Zhu, H., Wang, Y. (2016). An iterative learning algorithm for feedforward neural networks with random weights. Information Sciences, 328, pp. 546-557.

- [15] Sarkhel, R., Das, N., Das, A., Kundu, M., Nasipuri, M. (2017). A Multi-scale Deep Quad Tree Based Feature Extraction Method for the Recognition of Isolated Handwritten Characters of popular Indic Scripts. *Pattern Recognition*, 71, 78–93.
- [16] Acharya, S., Pant, A. K., Gyawali, P. K. (2015). Deep learning based large scale handwritten Devanagari character recognition. 2015 9th International Conference on Software, Knowledge, Information Management and Applications (SKIMA), pp. 1-6.
- [17] Jaderberg, M., Simonyan, K., Zisserman, A. (2015). Spatial transformer networks. In *Proceedings of the Advances in Neural Information Processing Systems*, Montreal, QC, Canada, 11–12 December 2015.
- [18] Ciręsan, D., Meier, U. (2015). Multi-column deep neural networks for offline handwritten Chinese character classification. In *Proceedings of the 2015 International Joint Conference on Neural Networks (IJCNN)*, Killarney, Ireland, 12–17 July 2015.
- [19] Chacko, A. M. M., Dhanya, P. M. (2015). A comparative study of different feature extraction techniques for offline Malayalam character recognition. In *Computational Intelligence in Data Mining- Volume 2* (pp. 9-18). Springer, New Delhi.
- [20] Yadav, P., Yadav, N. (2015). Handwriting recognition system-a review. *International Journal of Computer Applications*, 114(19), 36-40.
- [21] Choudhary, A. (2014). A review of various character segmentation techniques for cursive handwritten words recognition. *Int J Inf Comput Technol*, 4(6), 559-564.
- [22] Zeiler, M.D., Rob, F. (2014). Visualizing and understanding convolutional networks. In *Proceedings of the European Conference on Computer Vision*, Zurich, Switzerland, 6–12 September 2014.
- [23] Kumar, R., Kumar, A., Ahmed, P. (2013). A Benchmark Dataset for Devanagari Document Recognition Research. WSEAS Press, Lemesos, Cyprus.
- [24] Yadav, D., Sánchez-Cuadrado, S., Morato, J. (2013). Optical character recognition for Hindi language using a neural-network approach. *Journal of Information Processing Systems*, 9(1), 117-140.
- [25] Mithe, R., Indalkar, S., Divekar, N. (2013). Optical character recognition. *International journal of recent technology and engineering (IJRTE)*, 2(1), 72-75.
- [26] Fischer, A., Indermühle, E., Bunke, H., Viehhauser, G., Stolz, M. (2010). Ground truth creation for handwriting recognition in historical documents. In *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems*, 2010.
- [27] Ahranjany, S.S., Razzazi, F., Ghassemian, M.H. (2010). A very high accuracy handwritten character recognition system for Farsi/Arabic digits using Convolutional Neural Networks. In *Proceedings of the 2010 IEEE Fifth International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA)*, Changsha, China, 23–26 September 2010.

- [28] Algiahi, Y. (2010). Preprocessing techniques in character recognition. *Character recognition*, 1, 1-121-19.
- [29] Navneet, D., Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Proceedings of the CVPR2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, San Diego, CA, USA, 20–25 June 2005; Volume 1.
- [30] Wang, X., Paliwal, K.K. (2003). Feature extraction and dimensionality reduction algorithms and their applications in vowel recognition. *Pattern Recognition*, 36, 2429–2439.
- [31] Plamondon, R., Srihari, S. N. (2000). Online and off-line handwriting recognition: a comprehensive survey. *IEEE Transactions on pattern analysis and machine intelligence*, 22(1), 63-84.
- [32] Verma, B. K. (1995). Handwritten Hindi character recognition using multilayer perceptron and radial basis function neural networks. *Neural Networks*, 1995. *Proceedings, IEEE International Conference on*, vol. 4, pp. 2111-2115.
- [33] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proc. IEEE* 1998, 86, 2278–2324.
- [34] Jayadevan, R., Kolhe, S. R., Patil, P. M., Pal, U. Offline Recognition of Devanagari Script: A Survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 41(6), 782-796.
- [35] Sanmorino, Yazid, S. A survey for handwritten signature verification. *2012 2nd International Conference on Uncertainty Reasoning and Knowledge Engineering*, pp.
- [36] Umapada Pal, Ramachandran Jayadevan, Nabin Sharma. Handwriting recognition in Indian regional scripts: A Survey of Offline Techniques. *ACM Transactions on Asian Language Information Processing*, 11(1), Article 1.
- [37] Rusu, A., Govindaraju, V. The influence of image complexity on handwriting recognition. *Tenth International Workshop on Frontiers in Handwriting Recognition*, 2006.
- [38] eCun, Y., Bottou, L., Bengio, Y., Haffner, P., “Gradient-based learning applied to document recognition”, *Proc. IEEE* 1998, 86, 2278–2324.
- [39] Verma, B. K.,” Handwritten Hindi character recognition using multilayer perceptron and radial basis function neural networks in *Neural Networks*”, 1995. *Proceedings, IEEE International Conference on*. vol. 4, pp. 2111-2115, 1995.