A Project Report on

MARATHI HANDWRITTEN RECOGNITION

Submitted in partial fulfillment of the requirements

of the degree of

Bachelor of Engineering

in

Computer Science and Engineering (AI & ML)

by

Ms. Shintre Siddhi Laxman	**33**
Ms. Shetye Sakshi Vinayak	**32**
Ms. Dichavalkar Komal Janardan	**06**
Ms. Naik Sanjana Suryakant	**18**

Under the guidance of

Prof. Mhapasekar D. P.

Department of Computer Science and Engineering (AI & ML),

Sindhudurg Shikshan Prasarak Mandal's College of Engineering,

Harkul(Budruk), Kankavli, Maharashtra, India 416 602.

University of Mumbai

Academic Year: 2023-2024

Sindhudurg Shikshan Prasarak Mandal's College of Engineering,

Kankavali, Sindhudurg, Maharashtra

Department of Computer Science and Engineering (AIML)

Certificate

This is to certify that the project report entitled " **MARATHI HAND-WRITTEN RECOGNITION**" is a bonafide work of

Ms. Shintre Siddhi Laxman	**33**
Ms. Shetye Sakshi Vinayak	**32**
Ms. Dichavalkar Komal Janardan	**06**
Ms. Naik Sanjana Suryakant	**18**

submitted by submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of **Bachelor of Engineering** in **Computer Science and Engineering (AIML)**.

Prof. Mhapasekar D. P.
Project Guide

Mrs. P. A. Ghadigaonkar
Project Coordinator

Prof. S. S. Nalawade
Head of Department

Dr. Satam M. K.
Principal

Date: / / 2024

Place: Kankavli.

Approval Sheet

This project report entitled **Marathi Handwritten Recognition** by **Ms. Siddhi Shintre(33), Ms. Sakshi Shetye(32), Ms. Komal Dichavalkar (06), Ms. Sanjana Naik (18)** is approved for the degree of **Bachelor of Engineering** in **Computer Science and Engineering (Artificial Intelligence and Machine Learning)**.

Examiners :

1.____________________________

2.____________________________

Date:
Place: Kanakvli

Declaration

We declare that this written submission represents our ideas in ours own words and where others' ideas or words have been included, We have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Name and signature of students:

1. Ms. Siddhi Laxman Shintre(34)

2. Ms. Sakshi Vinayak Shetye(33)

3. Ms. Komal Janardan Dichavalkar (06)

4. Ms. Sanjana Suryakant Naik (18)

Date : April 13, 2024

Abstract

Marathi handwritten character recognition, enabled by image processing and deep learning, is a groundbreaking development in the field of Devanagari OCR. It unlocks the vast potential of Devanagari heritage and culture. This project aims to identify Marathi handwritten characters in a sentence or a line. The challenge is that there are different types of handwriting, from words to block letters, and each letter has different symbols, leading to errors and reducing accuracy. To overcome this, we use a holistic approach that combines image processing technology and deep learning. It starts with a thorough study of language and character segmentation.

We use Convolutional Neural Networks (CNN) for behavior recognition. The feature that sets us apart is the Boost technology, which uses five neural networks with different settings to increase accuracy. Our system relies on Python's diverse ecosystem, using OpenCV for image processing, TensorFlow/Keras for neural networks, and sci-kit-learn for machine learning. We collaborate with Google for interactive development and use GPU for rapid training. Our system is incredibly sensitive, flexible, and progressive. It blends accuracy, adaptability, and efficiency to revolutionize Marathi handwritten text recognition and will transform Devanagari text.

Contents

List of Figures

List of Tables

Abbreviation Notation and Nomenclature

Table 1: List of Abbreviations

Abbreviation	**Full Name**
ML	Machine Learning
DL	Deep Learning
AI	Artificial Intelligence
OCR	Optical Character Recognition
CNN	Convolutional Neural Network
LSTM	Long Short Term Memory
PCA	Principal Component Analysis
GPU	Graphics Processing Unit

Chapter 1

Introduction

Marathi is the official language of India and is spoken by over 80 million people worldwide. The language uses the Devanagari script, which is also used by other Indian languages such as Hindi and Sanskrit. However, recognizing Marathi handwriting (HMCR) can be challenging due to various factors, such as the presence of noise in handwritten images, variations in handwriting styles, and the large number of characters in the Devanagari script.

1.1 Key Problems

There are some of the key problems that need to be addressed in designing a Marathi handwritten recognition system:

1. Character Set Size : Devanagari script has a large character set of over 50 characters. 15 consonants and vowels. This makes it difficult to train a classifier that can accurately identify all characters [1].

2. Handwriting variability: Handwriting styles can vary widely from person to person. This can make it difficult for a classifier to generalize to new handwriting styles [2].

3. Noise: Handwritten images can be noisy due to factors such as pen smudges. and paper folds. This can make it difficult for a

classifier to extract meaningful features from the images [3].

Here are some specific examples of problems that can occur in Marathi handwritten recognition:

1. Connected characters: In Marathi handwriting, characters are often connected. This can make it difficult for a classifier to segment the individual characters in the image [4].

2. There is a lot of scope for Marathi handwritten recognition systems. [4].

1.2 Need for Marathi Handwritten Recognition System

Here are some specific areas where there is a need for good Marathi handwritten recognition systems:

1. Digitizing handwritten Marathi manuscripts and documents: There is a vast amount of handwritten Marathi content in the form of manuscripts, books, letters, and other documents. Digitizing this content would make it more accessible to researchers and scholars, and would also help to preserve this important cultural Heritage [5].

2. Developing Marathi handwriting tutorials and other educational applications: Marathi handwritten recognition systems could be used to develop Marathi handwriting tutorials and other educational applications. These would help people to learn Marathi handwriting and to improve their handwriting skills.

In addition to these specific areas, there is also a need for Marathi handwritten recognition systems that are more accurate and efficient. Current Marathi handwritten recognition systems still have some limitations in terms of their accuracy and speed [6].

1.3 Limitations of current MHRS

In the development of a Marathi handwritten recognition system, several key challenges and limitations must be considered, including:

1. **Character Set Complexity:** Recognizing the extensive Devanagari character set poses accuracy challenges [7].
2. **Handwriting Variability:** Adapting to diverse handwriting styles remains a significant hurdle.
3. **Noisy Input:** The presence of noise, such as smudges, affects recognition accuracy .
4. **Confusing Similar Characters:** Distinguishing similar-looking Marathi characters can be challenging [8].
5. **Real-time Processing Demands:** Achieving real-time recognition may require substantial computational resources.

Chapter 2

Literature Review

Table 2.1: Literature Review

Author and Year	**Methods**	**Algorithm for Classification**	**Result**	**Future Scope**
Prashant Madhukar Yawalkar et al [9] 2019	Pre-processing, segmentation, feature extraction	K-means clustering, Kirsch directional edge detection	Accuracy: 99.98	Develop models for identification and recognition of Marathi letters and sentences. Explore CNN architectures like MobileNet and EfficientNet for higher accuracy with lower computational cost.
Naresh Kumar Garg et al [10] 2022	Online and offline character recognition, feature extraction, classification, OCR, deep learning techniques	Deep learning techniques	Accuracy: 99.98	Enhance accuracy through advanced deep learning. Diversify datasets for various styles and contexts. Create real-time mobile apps for instant recognition. Extend recognition to other Indian languages in Devanagari script.

Table 2.2: Literature Review

Author and Year	**Methods**	**Algorithm for Classification**	**Result**	**Future Scope**
Yogesh Dandawate et al [11] 2022	Preprocessing, Feature Extraction (CNN architecture), Recognition (CNN-based Recognition)	CNN architecture for feature extraction and recognition	Accuracy: 98%	Future work could focus on improving accuracy and expanding the system's capabilities.
M. K. Jindal et al [12] 2019	Image pre-processing, Segmentation, Feature extraction, Classification	Not specified	Accuracy: 88.95%	Future work includes expanding the character recognition system and enhancing accuracy.
Munish Kumar et al [13] 2019	Matra/Shirorekha Feature, Segmentation Based Feature, Convex-Hull Based Feature	Not specified	Accuracies: 99.27%, 99.54%, and 99.40%	Various such models can be developed for identification and recognition of Marathi characters.

Table 2.3: Literature Review

Author and Year	**Methods**	**Algorithm for Classification**	**Result**	**Future Scope**
M. K. Jindal et al [14] 2021	Fine-Tuning, Classification, Dataset Creation, Pre-processing and Data Augmentation	Not specified	Accuracy: 96.55%	Future work will focus on deploying more compact deep convolutional networks for character classification and exploring different network architectures for compound Devanagari characters and words. Additionally, structural features like concavity will be investigated to enhance the accuracy of characters with similar shapes.
Duddela Sai Prashanth et al [15] 2018	CNNs for OCR, CNN-based OCR system for Devanagari ancient manuscripts	CNN-based OCR system	Accuracy: 93.73%	Develop a system that can recognize characters from different handwriting styles. Develop a system that can recognize characters that are degraded or damaged. Develop a system that can recognize characters from different languages.

Chapter 3

Research Gaps

After a comprehensive review of existing literature on Marathi handwritten recognition systems, several research gaps have been identified. These gaps represent areas where further investigation and development are needed to advance the state-of-the-art in this domain. The synthesis of various papers has revealed significant challenges and opportunities for improvement in Marathi handwriting recognition. By analyzing the collective findings and methodologies employed in these studies, the following research gaps have emerged, informing the direction of future research efforts in this field.

3.1 Accuracy

Marathi handwriting recognition is a challenging task due to the complex nature of the Marathi script. There is a need for more research on improving the accuracy of Marathi handwriting recognition systems.

3.2 Data set

There is a lack of large-scale, high-quality data sets of Marathi handwriting. This makes it difficult to train and evaluate Marathi handwriting

3.3 Robustness

Marathi handwriting recognition systems are often not robust to variations in handwriting style, noise, and other factors. There is a need for more research on making Marathi handwriting recognition systems more robust.

3.4 Real-time performance

Marathi handwriting recognition systems often do not perform well in real-time applications. There is a need for more research on improving the real-time performance of Marathi handwriting recognition systems.

3.5 Tool Development

There is no tool available for the recognition of handwritten text in the Marathi language.

3.6 Adaptability to Dialects and Styles

Marathi handwriting recognition systems may struggle with recognizing handwriting variations across different dialects and personal styles. Research is needed to develop algorithms and techniques that can adapt to these variations effectively.

Chapter 4

Problem Statement and Objectives

4.1 Problem Statement

Given a handwritten Marathi character, the system needs to predict its type accurately. Essentially, if we input a handwritten character, the system should predict the most likely corresponding character from the Marathi alphabet or determine if the input character closely resembles a particular Marathi character. The objective of this project is to process handwritten Marathi characters as input, effectively train a neural network using appropriate algorithms, and enable it to recognize patterns.

The aim of this project is to develop a model capable of recognizing and verifying handwritten characters from their images using Convolutional Neural Network (CNN) techniques. The primary objective of the proposed system is to understand Convolutional Neural Networks and apply them to the task of recognizing handwritten Marathi characters.

4.2 Objectives

- Recognition of Devanagari Script alphabets
- To provide an easy user interface for the users to recognize the characters with a free hand.
- The system should be able to predict the hand-written characters in the presence of noise effectively.
- Reduced manpower to convert handwritten text into digital format.

Chapter 5

Proposed System

The Marathi Handwritten Recognition System developed has consists of different functionalities grouped under different modules such as Input, Preprocessing and Feature Extraction, Dataset preparation, Data loading, Model Selection and Training, and Evaluation.

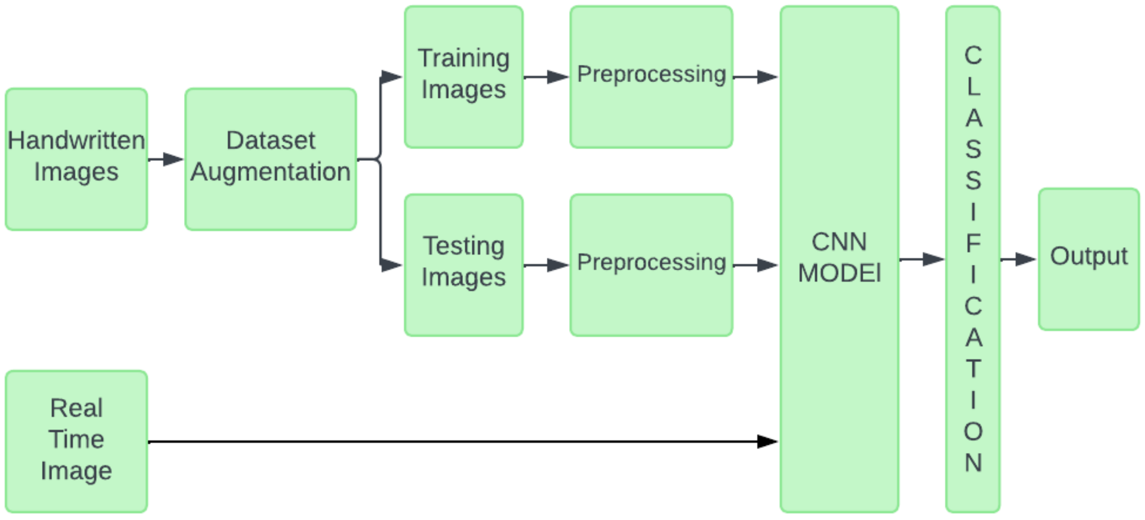


Figure 5.1: Convolutional Neural Network

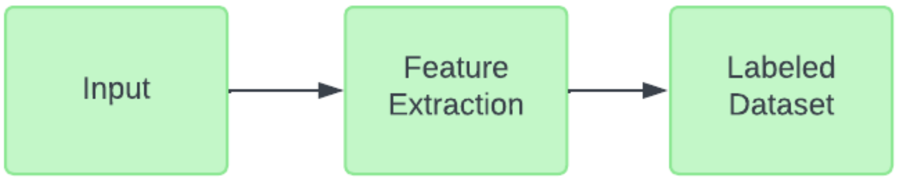


Figure 5.2: Feature Extraction

5.1 Input

Scanned Devanagari handwritten documents are used as input. These documents contain Marathi characters and words.

5.2 Dataset Preparation

The data from images will be extracted and stored in the CSV file and image dimensions will be verified and normalized. The image data will be split into datsets such as training, testing, and validation using different sampling techniques.

5.3 Preprocessing and Feature Extraction (CNN)

In this module, the input images are preprocessed using techniques such as noise reduction, image resizing, contrast enhancement, color correction, and segmentation. Convolutional Neural Networks (CNN) are employed for feature extraction. CNNs extract pixel values, edges, gradients, and other visual features from the handwritten content.

5.4 CNN Classification

A CNN-based classification model is used for character recognition. The trained model assigns characters to their respective classes in the Devanagari script.

5.5 Model Selection and Training

Model selection is the process of selecting the proper model that will generalize the data well and address the problem.

5.6 Evaluation

Recognized characters are the project's output. The result is a digitized, machine-readable version of the scanned handwritten content. The project's goal is to automate the conversion of handwritten Marathi text into a digital format, enabling accessibility and searchability for applications like text analysis and data preservation. It relies on CNNs for feature extraction and character recognition.

Chapter 6

Methodology

6.1 Convolutional Neural Network

In deep learning, a convolutional neural network (CNN/ConvNet) is a class of deep neural networks. It is used for image recognition and processing tasks. The Convolution Neural Network consists of a total of 4 layers, such as Input Layer, Convolution Layer, Pooling Layer, and Dense Layer(Fully connected layer). In CNN, filters are applied to the input image to extract features like edges, textures, and shapes. The output of the convolutional layers is input to pooling layers, which is used to down-sample the feature maps, reducing the spatial dimensions and keeping the most important information. The output of the pooling layers is given to one or more fully connected layers, which are used to make a prediction or classify the image.

- Kernels : also known as kernel techniques or kernel functions, are a collection of distinct forms of pattern analysis algorithms, using a linear classifier, they solve an existing non-linear problem. SVM (Support Vector Machines) uses Kernels Methods in ML to solve classification and regression issues

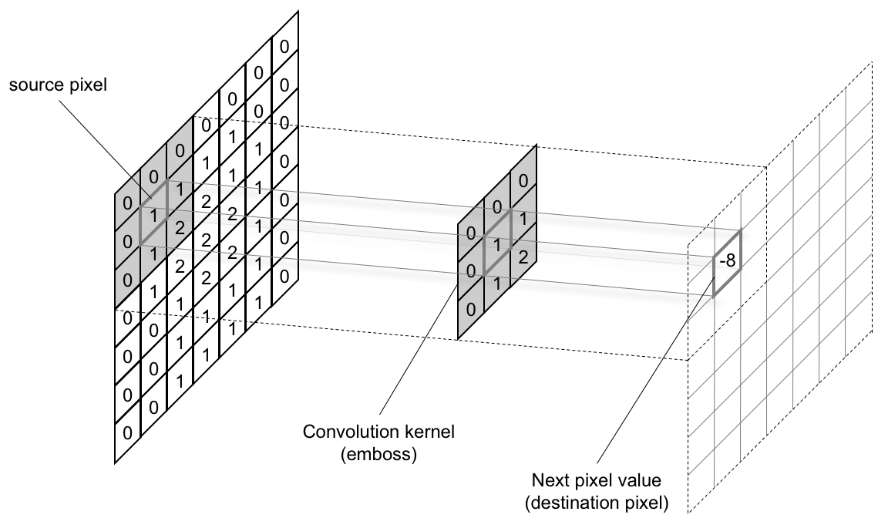


Figure 6.1: Kernel

- Stride :It is a parameter of the neural network's filter that modifies the amount of movement over the image or video. For example, if a neural network's stride is set to 1, the filter will move one pixel, or unit, at a time

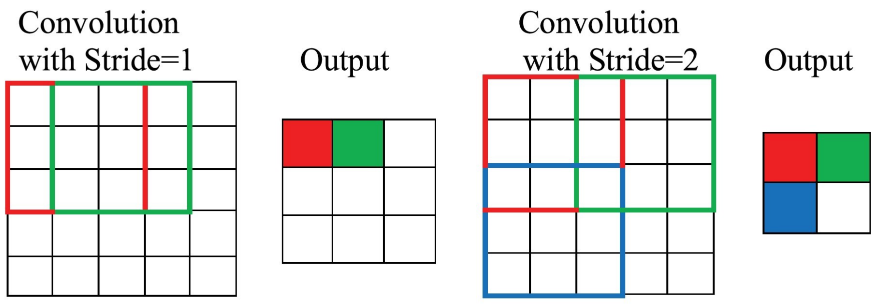


Figure 6.2: Stride

- Padding : It is a technique employed that guarantees that the input data has the exact size and shape that the model anticipates after every convolutional operation at each stage of the deep learning model.

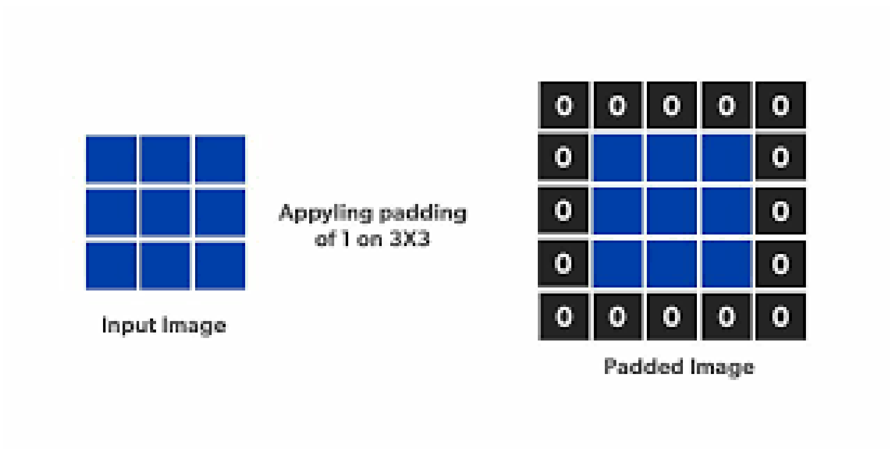


Figure 6.3: Padding

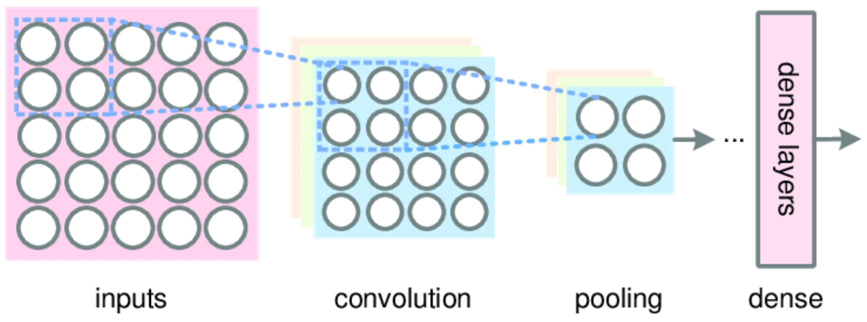


Figure 6.4: Convolutional Neural Network

6.2 CNN Layers

6.2.1 Input Layer

The input layer is the input to the convolution neural network. In the neural network of image processing, it represents the pixel matrix of the image.

6.2.2 Convolution Layer

This is the second layer that is used to extract the various features from the input images. This layer performs the mathematical operation of convolution between the input image and a filter of a size MxM. By putting the filter over the input image, the dot product is taken between the filter and the parts of the input image. TThe most commonly used convolution is the 2D convolution layer and it is written as as conv2D. A filter or a kernel in a conv2D layer “slides” over the input data, performing a dot product. All the values will be summed up to get the output as a single output pixel. The filter will perform the dot location it slides over.

6.2.3 Pooling Layer

Pooling layer is used to reduce the dimensions of the feature maps. Thus, it reduces the number of parameters and the amount of computation in the network.Pooling layer reduces number of features and parameters so that further processing will be easy.

Types of Pooling Layers:

1. **Max Pooling** Max pooling is a one of the type of pooling that selects the maximum value from the matrix of the feature map. So,the output of the max-pooling layer is a feature map consisting the maximum value from the previous feature map.

2. **Average Pooling** Average pooling is a one of the type of pooling that computes the average value from the matrix of the feature map. So,the output of the average pooling layer is a feature map consisting the average value from the previous feature map.

3. **Minimum Pooling** Minimum pooling is a one of the type of pooling that selects the minimum value from the matrix of the feature map. So,the output of the minimum pooling layer is a feature map consisting the minimum value from the previous feature map

6.2.4 Dense Layer

Dense Layer is simply a layer of neurons in where each neuron take input from previous layer neuron, so it is called as dense. Dense Layer is used for classifying image based on output from convolutional layers. Dense layer is also known as fully connected layer as it takes input from all neurons of previous layers. One can add as many dense layers as required in application.

6.3 Training

Training a deep neural network refers to the process of giving input data to the network, calculating the output and comparing it to the desired output, and adjusting the network's weights and biases to minimize the error. Error is nothing but a difference between the predicted value and desired outputs. Minimizing the loss is typically done using an optimization algorithm like gradient descent, where network parameters are updated based on a previously computed gradient at each iteration.

6.4 Optimization

Optimization in deep neural networks is the process of finding the set of weights and biases that will minimize the error. i.e. loss function. This is done through using optimization algorithms, such as gradient descent and Adam. These algorithms calculate the gradients of the loss function concerning the network parameters and update them in the direction that minimizes the loss. Various optimization techniques, like momentum, learning rate scheduling, and adaptive methods, can be applied to speed up convergence and improve performance.

1. **Adam**

   1. The name Adam is inspired by the idea of adapting to change and making the most of every moment.
   2. An Adam combines the principles of Gradient Descent with Momentum and the RMSProp algorithm.
   3. The Momentum algorithm is employed to accelerate the Gradient Descent process by considering the exponentially weighted average of the gradient.RMSProp is an adaptive learning rate that rises to an improved degree. also, preserve per-parameter learning rates adjusted to the weight based on the average of recent magnitude.

2. **Gradient Descent**

   1. Gradient Descent is used to minimize a loss function.
   2. In this function, the objective is to minimize or maximize it by taking derivatives when dealing with problems involving more than one variable.
   3. The way to minimize an objective function is parameterized by updating the parameters in the opposite direction of gradient descent.
   4. This can lead to faster convergence and improved performance.

6.5 Activation functions

Activation functions decide whether a neuron should be activated or not by calculating the weighted sum and further adding bias to it. The purpose of the activation function is to introduce non-linearity into the output of a neuron. The neural network has neurons that work in correspondence with weight, bias, and their respective activation function. Neural networks would update the weights and biases of neurons on the biases of error at output. This process is known as backpropagation. In an artificial neural network, each neuron forms a weighted sum of input and passes the resulting scalar value through a function as an activation function or transfer function. If a neuron has n input then the output or activation function of a neuron is

$$a = g(W_1X_1 + W_2X_2 + \ldots + W_nX_n) \tag{6.1}$$

6.5.1 Relu:

Relu is the most widely used activation function. It is implemented in a hidden layer. It gives output x if x is positive otherwise zero. It is non non-linear activation function. This means we can easily backpropagate the errors and have multiple layers of neurons being activated by the Relu function. Relu is less computationally expensive than tanh and sigmoid because it involves simpler mathematical operations. At a time only a few neurons are activated making the new network sparse and easy and efficient for computation.

6.5.2 Softmax:

The softmax function is also the type of sigmoid function but it is handy when we are trying to handle multi-class classification problems. It is a non-linear activation function. It ranges between 0 to 1. The softmax function was commonly found in the output layer of image classification problems. The softmax function would squeeze the output for each class between 0 to 1 and also divide by the sum of the output.

Chapter 7

Experimental Setup

7.1 Details About Input to Systems

The input to a Marathi handwritten recognition system is a digital image of a handwritten Marathi word. Here are some specific details about the input to Marathi handwritten recognition systems:

1. **Image format:** The input image can be in any standard image format, such as JPEG, or PNG.
2. **Image size:** The image should be large enough to capture the details of the handwritten characters, but not so large that it slows down the recognition process. A typical image size is 300-600 pixels per inch (PPI).
3. **Image color:** The input image can be in black and white. Some systems may perform better on color images, while others may prefer gray scale images.
4. **Handwriting style:** The system should be able to recognize handwritten characters in a variety of styles, including cursive and print.
5. **Language:** The system should be able to recognize handwritten characters in Marathi.

7.2 Software and Hardware Setup

To ensure the successful development and execution of the "Marathi Handwritten Recognition" project, it's essential to set up the following software and hardware components correctly.

7.2.1 Software Setup

1. **Python Environment:** Set up Python as the primary programming language. Create a virtual environment to manage packages and dependencies.
2. **Deep Learning Framework or Technique:** Utilize Convolutional Neural Networks (CNN) as the deep learning framework or technique for the project.
3. **Data Processing Libraries:** Install essential data processing libraries, including NumPy, pandas, and OpenCV, to handle datasets and perform image processing.
4. **Integrated Development Environment (IDE):** Choose an IDE like Google Colab for coding and experimentation. Customize the IDE to meet coding preferences.
5. **Version Control:** Set up Git for version control, and create a repository on platforms like GitHub to track changes, collaborate with team members, and maintain project history.

7.2.2 Hardware Setup

1. **Computer:** Ensure the computer is equipped with a multicore processor for faster computation, which is particularly crucial for deep learning tasks.
2. **Memory (RAM):** Have at least 16GB of RAM to effectively

handle large data-sets and complex machine learning models, especially when using deep learning frameworks like CNN.

3. **Environment Isolation and Version Control:** Utilize virtual environments (e.g., Python's virtualenv) to isolate project dependencies, ensuring that different projects do not interfere with each other.

4. **Development Tools:** Use Google Colab as primary integrated development environment (IDE) for coding. This tools offers excellent support for Python, deep learning frameworks, and version control integration.

5. **Testing and Benchmarking:** Before starting the project, run a benchmarking test on your hardware setup to ensure that it meets the performance requirements for training CNN models effectively. Adjust hardware configurations if necessary.

These software and hardware setup steps, along with the additional points, provide a comprehensive foundation for the "Marathi Handwritten Recognition" project, ensuring that it runs smoothly and efficiently.

Chapter 8

Implementation Details

8.1 Source code for Model 1

model.add(Conv2D(128, (3, 3), strides = 1, activation = 'relu', inputshape = (32, 32, 1))) model.add(MaxPooling2D((2, 2), strides = (2, 2), padding = 'same'))

model.add(Conv2D(128, (3, 3), strides = 1, activation = 'relu'))
model.add(MaxPooling2D((2, 2), strides = (2, 2), padding = 'same'))

model.add(Conv2D(128, (3, 3), strides = 1, activation = 'relu'))
model.add(MaxPooling2D((2, 2), strides = (2, 2), padding = 'same'))

model.add(Flatten())
model.add(Dense(128, activation = 'relu',
kernelinitializer = 'he$_uniform'))model.add(Dropout(0.1))$

$model.add(Dense(100, activation =' relu',$
$kernel_initializer =' he_uniform'))model.add(Dropout(0.1))$

$model.add(Dense(49, activation =' softmax'))$
$model.compile(optimizer = Adam(learning_rate = 1e-3, decay = 1e-5), loss =' categorical_crossentropy', metrics = ['accuracy'])$

8.2 Source code for Model 2

model.add(Conv2D(32, (5, 5), padding = 'same', activation = 'relu', kernelinitializer = 'heuniform', input$_{s}hape = (32, 32, 1)))$

$model.add(Conv2D(32, (5, 5), padding =' same', activation =' relu',$ $kernelinitializer =' he_uniform'))$

$model.add(Conv2D(32, (5, 5), padding =' same', activation =' relu',$ $kernelinitializer =' he_uniform'))$

$model.add(MaxPooling2D((2, 2), strides = (2, 2)))$

$model.add(Conv2D(64, (3, 3), padding =' same', activation =' relu',$ $kernelinitializer =' he_uniform'))$
$model.add(Conv2D(64, (3, 3), padding =' same', activation =' relu',$ $kernelinitializer =' he_uniform'))$

$model.add(Conv2D(64, (3, 3), padding =' same', activation =' relu',$ $kernelinitializer =' he_uniform'))$
$model.add(MaxPooling2D((2, 2), strides = (2, 2)))$
$model.add(Flatten())$

$model.add(Dense(256, activation =' relu',$ $kernel_initializer =' he_uniform'))model.add(Dropout(0.2))$

$model.add(Dense(49, activation =' softmax'))$
$model.compile(optimizer = Adam(learning_rate = 1e - 3, decay = 1e - 5),$
$loss =' categoricalcrossentropy', metrics = ['accuracy'])['accuracy'])$

8.3 Source code for Model 3

model.add(Conv2D(32, (3, 3), strides = 1, activation = 'relu', input$_{s}hape = (32, 32, 1)))$
$model.add(Conv2D(32, (3, 3), strides = 1, activation =' relu'))$
$model.add(MaxPooling2D((2, 2), strides = (2, 2), padding =' same'))$

$model.add(Conv2D(64, (3, 3), strides = 1, activation =' relu'))$
$model.add(Conv2D(64, (3, 3), strides = 1, activation =' relu'))$
$model.add(MaxPooling2D((2, 2), strides = (2, 2), padding =' same'))$

$model.add(Flatten())$

$model.add(Dense(128, activation =' relu',$
$kernel_initializer =' he_uniform'))model.add(Dropout(0.1))$

$model.add(Dense(49, activation =' softmax'))$
$model.compile(optimizer = Adam(learning_rate = 1e - 3, decay = 1e - 5),$
$loss =' categorical_crossentropy', metrics = ['accuracy'])$

8.4 Source code for Model 4

model.add(Conv2D(128, (3, 3), padding = 'same', activation = 'relu',
kernel$_initializer =' he_uniform', input_shape = (32, 32, 1)))$

$model.add(Conv2D(128, (3, 3), padding =' same', activation =' relu',$
$kernel_initializer =' he_uniform'))$

$model.add(Conv2D(128, (3, 3), padding =' same', activation =' relu',$
$kernel_initializer =' he_uniform'))$

$model.add(MaxPooling2D((2, 2), strides = (2, 2), padding =' same'))$
$model.add(Conv2D(64, (3, 3), padding =' same', activation =' relu',$
$kernel_initializer =' he_uniform'))model.add(MaxPooling2D((2, 2),$
$strides = (2, 2), padding =' same'))$

$model.add(Flatten())$

$model.add(Dense(256, activation =' relu', kernel_initializer =' he_uniform'))$
$model.add(Dropout(0.2))$

$model.add(Dense(128, activation =' relu', kernel_initializer =' he_uniform'))$
$model.add(Dropout(0.1))$

$model.add(Dense(49, activation =' softmax'))$

$model.compile(optimizer = Adam(learning_rate = 1e - 3, decay =$
$1e - 5),$
$loss =' categorical_crossentropy', metrics = ['accuracy'])$

8.5 Source code for Model 5

model.add(Conv2D(32, (5, 5), activation = 'relu', padding = 'same',
kernel$_{i}nitializer =' he_{u}niform', input_{s}hape = (32, 32, 1)))$
$model.add(Conv2D(64, (3, 3), activation =' relu'))$
$model.add(MaxPooling2D((2, 2), strides = (2, 2), padding =' same'))$

$model.add(Conv2D(64, (3, 3), activation =' relu'))$

$model.add(Conv2D(64, (3, 3), activation =' relu'))$
$model.add(MaxPooling2D((2, 2), strides = (2, 2), padding =' same'))$

$model.add(Conv2D(64, (3, 3), activation =' relu'))$
$model.add(Conv2D(64, (3, 3), activation =' relu'))$
$model.add(MaxPooling2D((2, 2), strides = (2, 2), padding =' same'))$

$model.add(Flatten())$

$model.add(Dense(256, activation =' relu'))model.add(Dropout(0.2))$
$model.add(Dense(128, activation =' relu'))model.add(Dropout(0.1))$
$model.add(Dense(49, activation =' softmax'))$

$model.compile(optimizer = Adam(learning_{r}ate = 1e - 3, decay =$
$1e - 5),$
$loss =' categorical_{c}rossentropy', metrics = ['accuracy']) = ['accuracy'])$

8.6 Source code for Boosting

```
Predictions = [np.argmax(Model.predict(x_validation),
axis=1) for Model in Predict_Characters.Models]
Dict = {}
for Model_Number, Prediction in enumerate(Predictions):
    Dict[f'Model_{Model_Number + 1}'] = [
    accuracy_score(y_validation, Prediction),
    precision_score(y_validation, Prediction,
    average='weighted', zero_division=0),
    recall_score(y_validation, Prediction,
    average='weighted', zero_division=0),
    f1_score(y_validation, Prediction,
    average='weighted', zero_division=0)
    ]
Prediction = stats.mode(Predictions)[0][0]
Dict['Boosting'] = [
    accuracy_score(y_validation, Prediction),
    precision_score(y_validation, Prediction,
    average='weighted', zero_division=0),
    recall_score(y_validation, Prediction,
    average='weighted', zero_division=0),
    f1_score(y_validation, Prediction,
    average='weighted', zero_division=0)
]
Validation_Report = pd.DataFrame.from_dict
(Dict, orient='index', columns=['accuracy_score',
'precision_score', 'recall_score', 'f1_score']).round(4)
```

Chapter 9

Experimental Result

9.1 Performance Evaluation Parameters

There are many Performance Evaluation Parameters like Accuracy, Precision, Recall, F1 score.

9.1.1 Accuracy

Accuracy is one of the performance evaluation criteria.
formula : accuracy = (number of correct predictions) / (total number of predictions)

9.1.2 Precision

Precision is one of the criteria for model's performance – the quality of a positive prediction made by the model. formula : (number of true positives)/(total number of positive predictions)

9.1.3 Recall

Recall is also calle True Positive Rate(TPR) formula : True Positives / (True Positives + False Negatives)

9.1.4 F1 Score

It is one of the models's performance evaluation criteria. It is combination of Precision and Recall. formula : 2*((Precision * recall) / (Precision + recall))

9.1.5 Graphs

The evaluation results demonstrate the effectiveness of the ensemble approach.Combining diverse models leads to a more accurate system for recognizing Marathi characters (HMCR). This ensemble model outperformed individual models in all aspects, including accuracy, precision, recall, and F1-score. In other words, it excels at correctly identifying characters and minimizing errors.

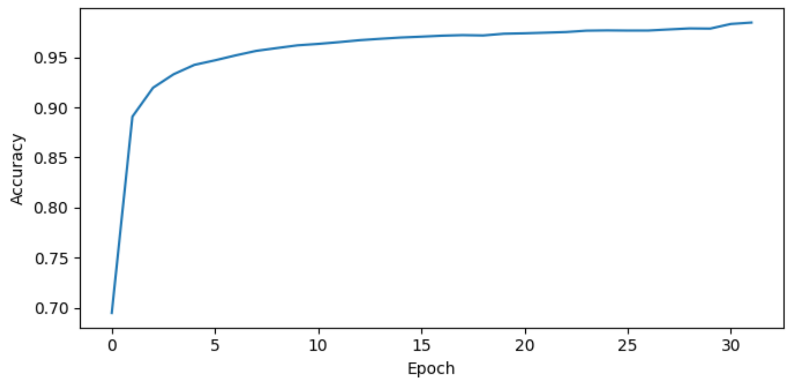


Figure 9.1: model 1

In order to find the optimum hyper-parameters, we employed a brute force approach by plotting the approach versus accuracy curve.The Y-axis represents the accuracy of the model, while the X-axis denotes the number of epochs. The graph illustrates how accuracy changes with the number of epochs. The accuracy of the model is observed to improve with an increasing number of epochs. It is noted that 90 percent accuracy is reached, as shown in the graph.

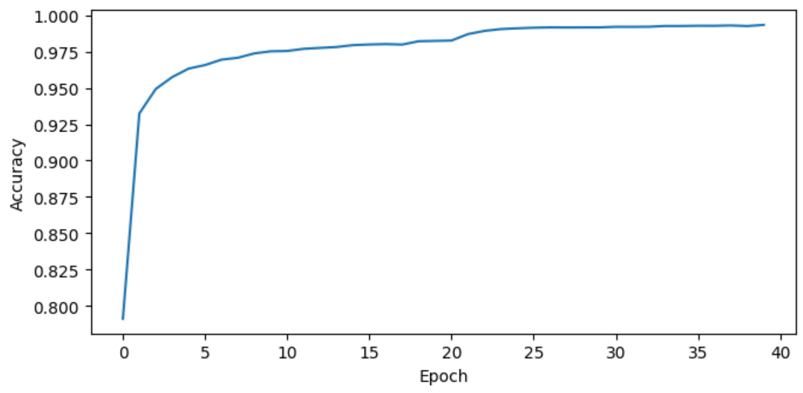


Figure 9.2: Model 2

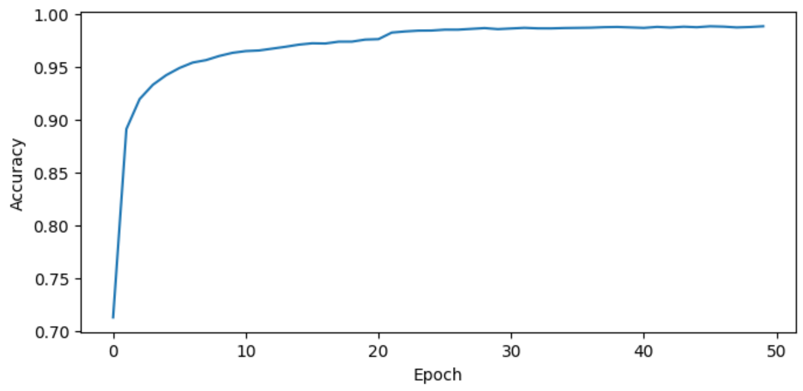


Figure 9.3: Model 3

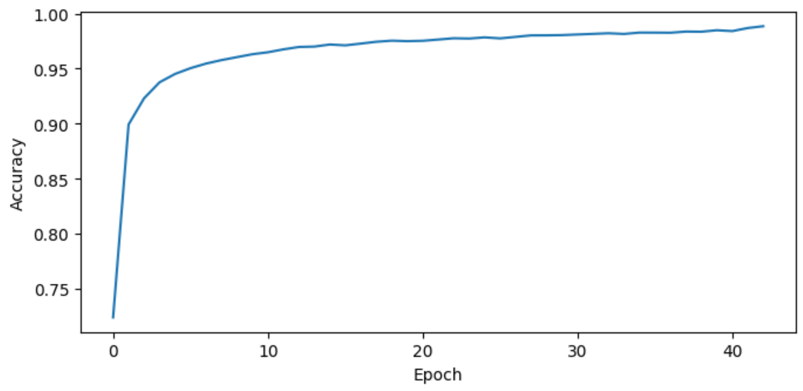


Figure 9.4: Model 4

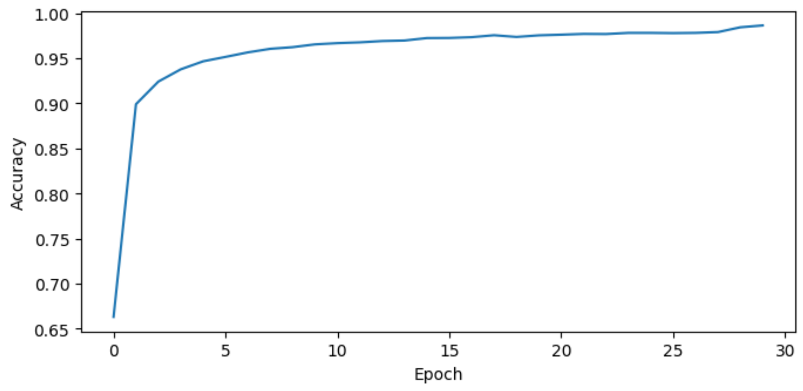


Figure 9.5: Model 5

9.1.6 Performance Evaluation

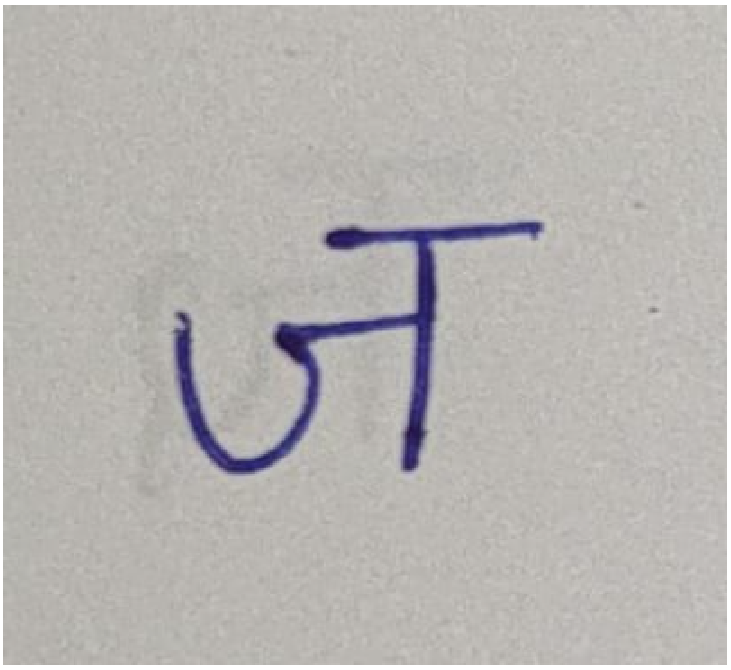
Table 9.1: Classification report on validation data

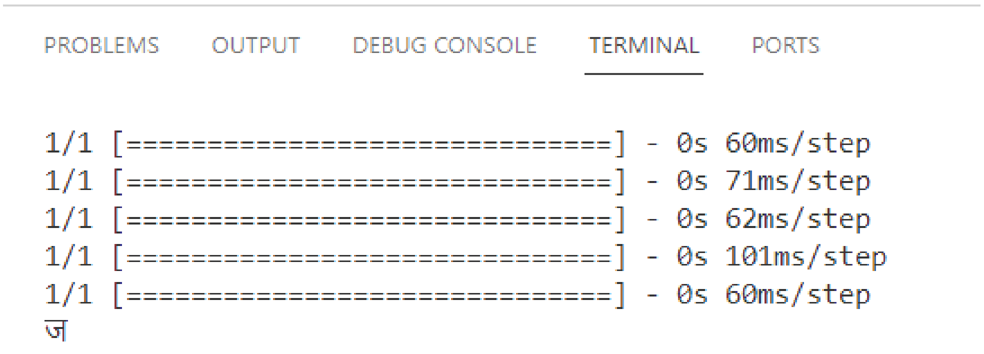
Model	Accuracy	Precision	Recall	F1-Score
1	0.9856	0.9858	0.9856	0.9856
2	0.9888	0.9889	0.9888	0.9888
3	0.9889	0.9890	0.9889	0.9889
4	0.9892	0.9894	0.9892	0.9892
5	0.9836	0.9838	0.9836	0.9836
Boosting	**0.9932**	**0.9933**	**0.9932**	**0.9932**

Table 9.2: Classification report on sample data

Model	Accuracy	Precision	Recall	F1-Score
1	0.7857	0.8733	0.7857	0.8153
2	0.8333	0.9415	0.8333	0.8701
3	0.7262	0.8333	0.7262	0.7485
4	0.7857	0.8175	0.7857	0.7895
5	0.8095	0.9200	0.8095	0.8386
Boosting	**0.8571**	**0.9444**	**0.8571**	**0.8862**

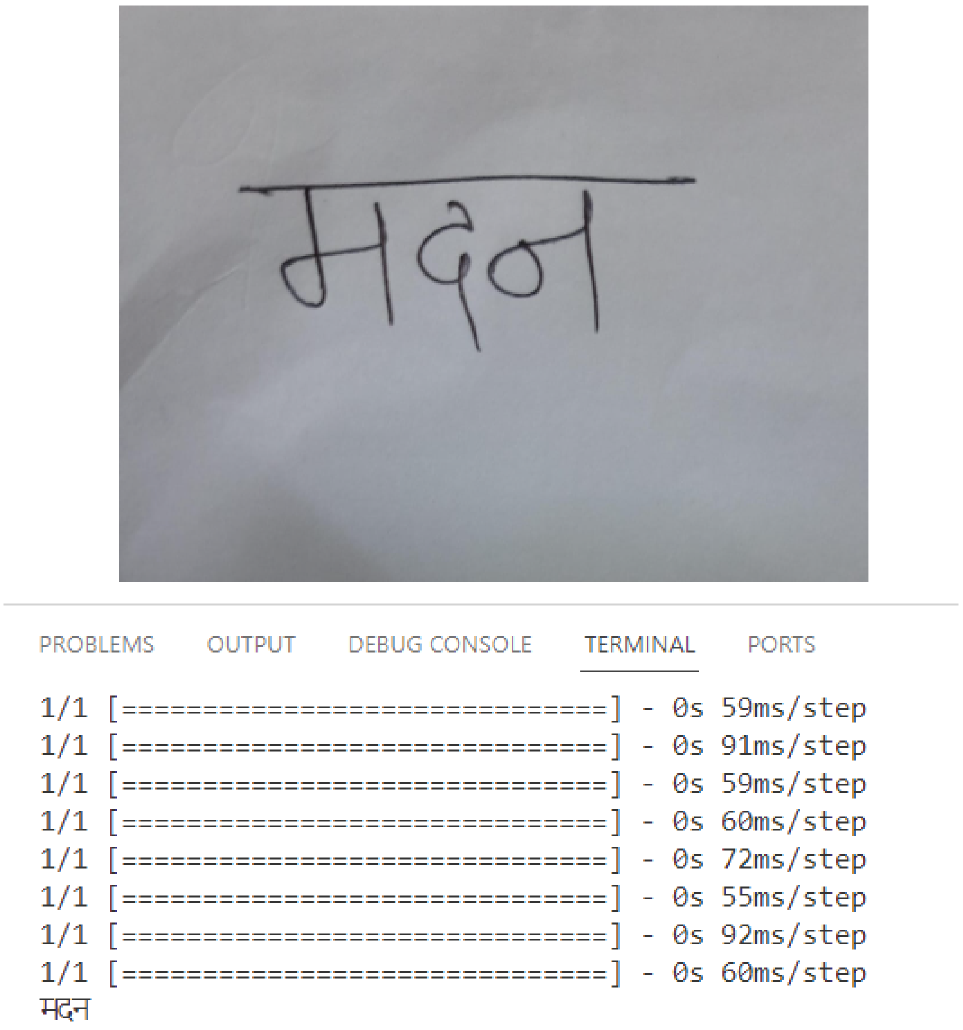
9.2 Sample Output





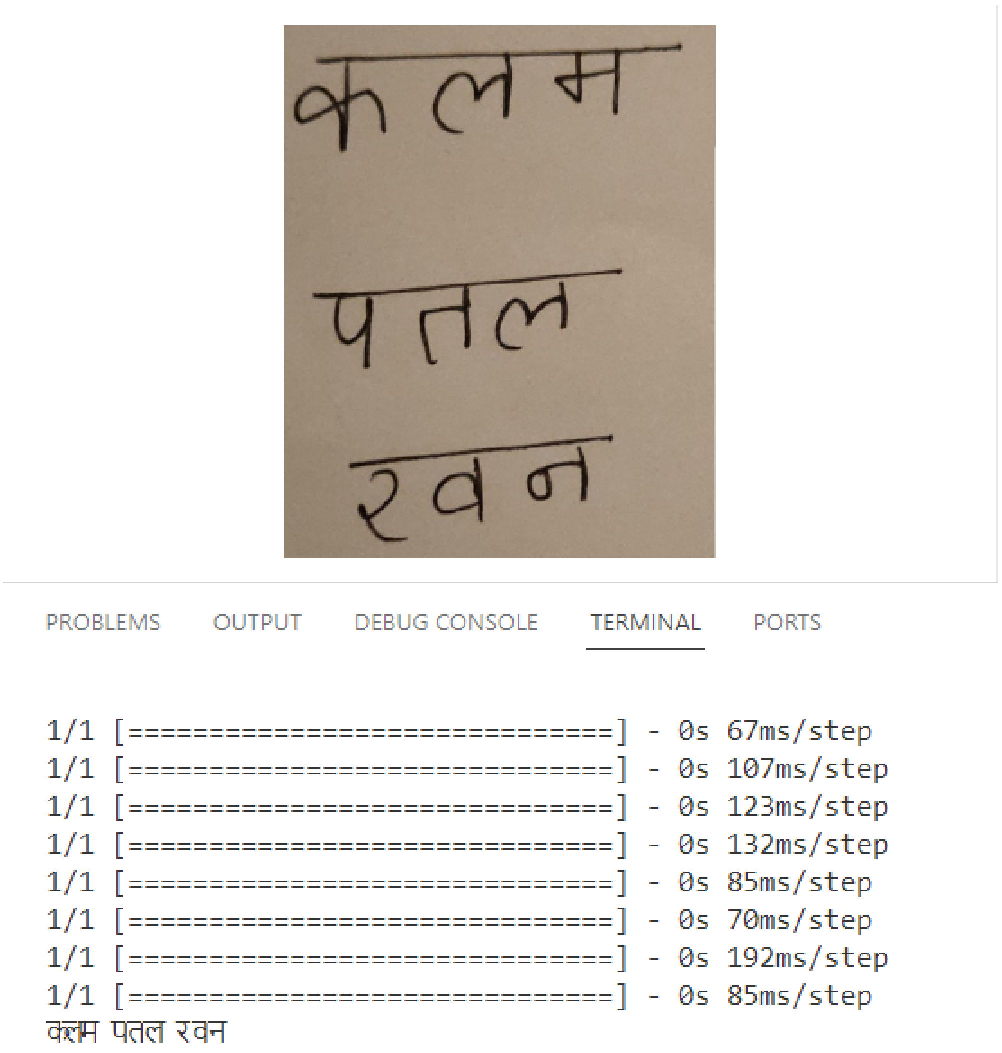
```
1/1 [==============================] - 0s 60ms/step
1/1 [==============================] - 0s 71ms/step
1/1 [==============================] - 0s 62ms/step
1/1 [==============================] - 0s 101ms/step
1/1 [==============================] - 0s 60ms/step
ज
```

Figure 9.6: Output 1



```
1/1 [==============================] - 0s 59ms/step
1/1 [==============================] - 0s 91ms/step
1/1 [==============================] - 0s 59ms/step
1/1 [==============================] - 0s 60ms/step
1/1 [==============================] - 0s 72ms/step
1/1 [==============================] - 0s 55ms/step
1/1 [==============================] - 0s 92ms/step
1/1 [==============================] - 0s 60ms/step
मदन
```

Figure 9.7: Output 2



```
1/1 [==============================] - 0s 67ms/step
1/1 [==============================] - 0s 107ms/step
1/1 [==============================] - 0s 123ms/step
1/1 [==============================] - 0s 132ms/step
1/1 [==============================] - 0s 85ms/step
1/1 [==============================] - 0s 70ms/step
1/1 [==============================] - 0s 192ms/step
1/1 [==============================] - 0s 85ms/step
कलम पतल रवन
```

Figure 9.8: Output 3

Chapter 10

Conclusion

The process of converting scanned Devanagari handwritten documents into digitized versions involves a meticulous series of steps, integrating image processing and deep learning techniques. Through preprocessing, segmentation, feature extraction, and character recognition, accurate representation of the original document is achieved. The utilization of Convolutional Neural Networks (CNNs) and Python's diverse ecosystem underscores the synergy between advanced technologies in this endeavor. The culmination is a precise digital recognition of Marathi handwritten characters, unlocking the rich heritage and cultural potential of Devanagari script. This groundbreaking approach, exemplified by our system's adaptability and progressive accuracy, promises to revolutionize Marathi handwritten text recognition, enhancing accessibility and facilitating its widespread application.

Publications by the candidates

1. Ms. Siddhi Shintre, Ms. Sakshi Shetye, Ms. Ko- mal Dichavalkar , Ms. Sanjana Naik "OPTIMIZED MARATHI HANDWRITTEN CHARACTER RECOGNITION USING CONVOLUTION NEURAL NETWORK" As in 2nd International Conference on Advances in Technology and Management. Journal of Emerging Technologies and Innovative Research India. ICATM -2024 India.

Chapter 11

Work done in Year

Action plan	2023						2024			
	Jul	Aug	Sep	Oct	Nov	Dec	Jan	Feb	Mar	Apr
1. Topic selection and document collection	■									
2. study of project related research papers		■								
3.Study of required algorithms			■							
4.Designed overview of the proposed system				■						
5. Model 1-5 coding and execution				■	■					
6. Review 1 preparation				■	■					
7. Implementation and suggest correction					■	■	■			
8. Experiment evaluation							■	■	■	
9. Write paper for publication						■	■	■		
10. Review 2							■	■		
11. Prepare black book report								■	■	■
12. Finalize report									■	■

Appendix

11.1 Boosting

Boosting is a machine learning ensemble technique that combines multiple weak learners to create a strong learner. It works iteratively by fitting models to repeatedly modified versions of the data, with each new model correcting the errors of its predecessor. Popular boosting algorithms include AdaBoost, Gradient Boosting, and XGBoost. Boosting is effective for reducing bias and improving predictive performance, often yielding highly accurate models across various tasks.

11.2 open CV

openCV (Open-Source Computer Vision Library) is released under a BSD license and hence it's free for both academic and commercial use. It has C++, Python, and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. OpenCV was designed for computational efficiency and with a strong focus on real-time applications. Written in optimized C/C++, the library can take advantage of multi-core processing. Enabled with OpenCL, it can take advantage of the hardware acceleration of the underlying heterogeneous compute platform. Adopted all around the world, OpenCV has more than 47 thousand people in the user community and an estimated number of downloads exceeding 14 million. Usage ranges from interactive art to mine inspection, stitching maps on the web, or through advanced robotics.

11.3 TensorFlow

In TensorFlow, boosting can be implemented using libraries like TensorFlow Boosted Trees (TFBT). TFBT provides a scalable and efficient way to train boosted tree models. It utilizes gradient boosting techniques to iteratively optimize a series of decision trees, resulting in a powerful ensemble model. TensorFlow's distributed computing capabilities further enhance the scalability of boosting, making it suitable for large-scale machine learning tasks.

Acknowledgement

We sincerely acknowledge with a deep sense of gratitude to Project Guide **Prof. Mhapasekar D. P.** and Project Coordinator **Mrs. P. A. Ghadigaonkar** for their valuable guidance, genuine suggestions, and constant encouragement during the preparation of project synopsis work without which completion of this task would be a difficult task.

We are also thankful to all of our faculty members of **Computer Science and Engineering (AIML) Department** especially our head of department **Prof. S. S. Nalawade** and our respected principal **Dr. M. K. Satam** who give us an idea of significant cooperation during the completion of this work.

We are immensely grateful to all who were involved in this project work because, without their cooperation, inspiration, constant promotion, and useful suggestions, it would be impossible to complete this task and synopsis report within this allotted time.

Project Members:
Ms. Siddhi Laxman Shintre
Ms. Sakshi Vinayak Shetye
Ms. Komal Janardan Dichavalkar
Ms. Sanjana Suryakant Naik

Date: April 13, 2024

Bibliography

[1] Mohamed Cheriet, Nawwaf Kharma, Cheng-Lin Liu, and Ching Suen. Character recognition systems: a guide for students and practitioners. 2007.

[2] Jamshed Memon, Maira Sami, Rizwan Ahmed Khan, and Mueen Uddin. Handwritten optical character recognition (ocr): A comprehensive systematic literature review (slr). *IEEE Access*, 8:142642–142668, 2020.

[3] Umapada Pal and BB Chaudhuri. Indian script character recognition: a survey. *pattern Recognition*, 37(9):1887–1899, 2004.

[4] Vikas J Dongre and Vijay H Mankar. A review of research on devnagari character recognition. *arXiv preprint arXiv:1101.2491*, 2011.

[5] Ruwei Dai, Chenglin Liu, and Baihua Xiao. Chinese character recognition: history, status and prospects. *Frontiers of Computer Science in China*, 1:126–136, 2007.

[6] Teófilo E de Campos, Bodla Rakesh Babu, and Manik Varma. Character recognition in natural images. 1:273–280, 2009.

[7] Ranadeep Dey, Pranav Gajanan Gawade, Ria Sigtia, Shrushti Naikare, Atharva Gadre, and Diptee Chikmurge. A comparative study of handwritten devanagari script character recognition techniques. *2022 IEEE World Conference on Applied Intelligence and Computing (AIC)*, pages 431–436, 2022.

[8] Diptee Chikmurge and Raghunathan Shriram. Marathi handwritten character recognition using svm and knn classifier. 2019.

[9] Prashant Yawalkar and Madan Kharat. Automatic handwritten character recognition of devanagari language: a hybrid training algorithm for neural network. *Evolutionary Intelligence*, 15, 04 2021.

[10] Munish Kumar, M. Jindal, and Sonika Narang. Devanagari ancient documents recognition using statistical feature extraction techniques. *Sadhana*, 44:1–8, 06 2019.

[11] Sukhjinder Singh, Naresh Garg, and Munish Kumar. Feature extraction and classification techniques for handwritten devanagari text recognition: a survey. *Multimedia Tools and Applications*, 82, 06 2022.

[12] Ambadas Shinde and Yogesh Dandawate. Convolutional neural network based handwritten marathi text recognition. 08 2020.

[13] Munish Kumar, M. Jindal, and Sonika Narang. Devanagari ancient documents recognition using statistical feature extraction techniques. *Sadhana*, 44:1–8, 06 2019.

[14] Shalaka Deore and Pravin Albert. Devanagari handwritten character recognition using fine-tuned deep convolutional neural network on trivial dataset. *Sādhanā*, 45:243, 09 2020.

[15] Duddela Prashanth, R Vasanth Kumar Mehta, and Ramana Kadiyala. Handwritten devanagari character recognition using modified lenet and alexnet convolution neural networks. *Wireless Personal Communications*, 122, 01 2022.