National College of Ireland

Extraction of Devanagari handwritten characters using Deep Learning-based Models

MSc Research Project
Artificial Intelligence

Rajratan Gajbhiye
Student ID: x22179038

School of Computing
National College of Ireland

Supervisor: Prof. Rejwanul Haque

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Rajratan Gajbhiye
Student ID:	x22179038
Programme:	Artificial Intelligence
Year:	2023-2024
Module:	MSc Research Project
Supervisor:	Prof. Rejwanul Haque
Submission Due Date:	31/01/2024
Project Title:	Extraction of Devanagari handwritten characters using Deep Learning-based Models
Word Count:	4849
Page Count:	21

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Rajratan Laxminarayan Gajbhiye
Date:	30th January 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Extraction of Devanagari handwritten characters using Deep Learning-based Models

Rajratan Gajbhiye
x22179038

Abstract

In several automation domains, such as Educational Technology, Digitization of Documents, The Analysis of Forensic Evidence, etc, Handwritten Character Recognition (HCR) is becoming more and more crucial. The approach of HCR involves the computer identifying and detecting each character in a text image and processing the data to create a machine-understandable format. The subject of recognition of patterns is a basic yet difficult job. In this research, we utilized the Devanagari Character Dataset. It is an open-source image dataset that contains 92,000 images of 46 different classes. This research investigates the efficiency and accuracy of three distinct models in training the recognition system: The proposed custom Convolutional Neural Network (CNN), Inceptionv3, and the Xception model. The proposed CNN approach is the most successful of these, obtaining an astounding accuracy of 99.11%. The results show that, out of all the models taken into consideration in this study, the Proposed custom CNN is the most accurate and computationally efficient model.

1 Introduction

Primarily in the area of character recognition, machine learning has made significant strides in the last few years. Research on HCR is becoming essential due to the widespread use of digital devices and the increasing need for effective data processing. This work focuses on using a custom CNN model, InseptionV3 model, and Xception model to extract handwritten characters from an image. An innovative approach that has a lot of potential for resolving the challenges encountered in identifying complexities and diverse scripts. Because of its complex and interconnected writing style, the Devanagari script used for languages like Hindi, Marathi, and Sanskrit presents particular difficulties when it comes to recognizing characters. Due to the complexities of Devanagari characters and the wide range of writing styles, traditional methods for handwritten character extraction are frequently inadequate. An efficient solution to this concern is provided by CNNs, which are widely acknowledged for their capacity to extract hierarchical features from data. To accurately extract Devanagari's handwritten characters, this research seeks to explore and utilize CNN models.

It is impossible to overestimate how deeply deep learning has changed the field of image recognition and character extraction from an image. This innovative methodology has great potential to revolutionize extraction and classification techniques for handwritten Devanagari characters. Conventional techniques fail to fully meet the significant issue

posed by the unique characteristics of the Devanagari script, which is characterized by compound letters and intricate shapes. The fundamental value of improving character extraction methods within the framework of Devanagari handwriting script is what inspires this research. Devanagari is a script with deep historical and cultural value that serves as a symbol of tradition and identity in addition to being a means of communication. Given its crucial importance in cultural as well as linguistic applications, it is important to precisely extract and recognize handwritten Devanagari characters. Additionally, the necessity of investigating cutting-edge technology is highlighted by the growing dependence on automation systems and the growing need for effective character recognition systems. The difficulties presented by the complexities of Devanagari characters can be addressed by deep learning, mainly through its complex algorithms and models. The larger goals of facilitating automation, helping the blind, and advancing natural language processing are what drive the search for a trustworthy and precise Devanagari character extraction system. To improve communication, increase availability, and preserve the rich cultural heritage, this research aims to use technology to uncover novel opportunities in character recognition.

The primary question guiding this investigation is: "How can Devanagari handwritten character extraction be made more accurate by using deep learning-based models?". The following objectives will direct our investigation to answer the research question:

- Build an easily understood CNN model to recognize handwritten Devanagari characters.
- Evaluate the performance of the CNN-based new models and compare them with the current pre-trained models like InseptionV3 and Xception model.
- Improve the accuracy and efficiency of a CNN-based model for the character recognition process.

2 Related Work

The primary objective of this report is to review earlier research on this topic, as well as the advantages and disadvantages of the approaches that have been utilized.

To outperform collective structures in precision while requiring fewer complexities, Ahlawat et al. (2020) explores the use of CNNs for the recognition of handwritten digits. With a 99.87% precision in recognition, the suggested CNN system is proven by a thorough evaluation of the MNIST dataset, which consists of 10,000 testing and 60,000 training pictures. For the best results, the research highlights the significance of improving learning characteristics and design specifications.

With a particular focus on Devanagari's handwritten character classification, Deore and Pravin (2020) aims to quickly construct an accurate character recognition system. A novel dataset including 5,800 images, encompassing 58 distinct character classes which include numerals, vowels, and consonants is presented in the research. The suggested Devanagari Handwritten Character Recognition System (DHCRS) shows notable results by utilizing a two-stage VGG16 model with adaptive gradient approaches. The first model achieves a training loss of 0.18 and a testing accuracy of 94.84%. With a training loss of 0.12 and a testing accuracy of 96.55%, the second optimized model performs well.

Based on LeNet-5, Chaudhary and Sharma (2019) presents a Deep Convolutional Neural Network (DCNN) for recognizing Hindi characters. Testing accuracy of 95.72%

with Adam optimizer and 93.68% with RMSprop optimizer, the model—which was trained on 96,000 characters—addresses issues specific to Hindi characters. With findings that are encouraging and outperform traditional methods, the suggested DCNN system opens up new possibilities for advancement.

CNNs with the Connectionist Temporal Classification (CTC) loss function is used by Liu et al. (2020) to tackle the complex problem of Handwritten Chinese Text Recognition (HCTR). For better efficiency, this research emphasizes a persistent-free design, in contrast to the conventional combination of Recurrent Neural Network (RNN) and CNN techniques. On the ICDAR 2013 competition set, the suggested technique obtained a 6.81% Character Error Rate (CER), outperforming earlier findings without requiring language model rectification.

Alrehali et al. (2020) describe a method for identifying and translating old Arabic texts into legible text that relies on CNNs. Between 74.29% and 88.20% accuracy are achieved using three datasets, each including a different number of photos. The datasets include Dataset 1 (40 images/letter) including 28 Arabic characters, Dataset 2 (100 images/letter) containing 10 selected characters, and Dataset 3 (200 images/letter) containing the same characters.

In more than a hundred Indian and Nepalese languages, written Devanagari character recognition is essential. Gurav et al. (2020) presents an approach to HCR. With a 99.65% accuracy rate, the approach uses DCNNs to retrieve features from a dataset of 34,604 images. The segmentation of characters using the technique of the Hough transform as well as binary transformation are two examples of image processing procedures used in the methodology. The approach highlights the continued intricate nature of Devanagari's Optical Character Recognition (OCR) by successfully recognizing obstacles.

To solve the problem of various types of writing, Ptucha et al. (2019) provides a completely CNN-based design for written text identification. With a CER of 4.70% (Word Error Rate - WER of 8.22%), the approach demonstrates advanced results on multiple standards, such as an IAM dataset of 115,320 words in the English language. 2.46% (WER: 5.68%) corresponds to the CER calculated using the RIMES dataset, which contains 60,000 French language words.

Using CNNs and Faster R-CNN in four important stages, Yang et al. (2019) presents a novel handwriting text recognition system. In comparison to conventional OCR, the Faster R-CNN achieves greater precision due to its efficient extraction of features and use of VGG-19 as an initial training model. The approach outperforms conventional OCR in terms of precision using three datasets: Letter (2000), EMNIST-Letters (100,000), and Word (1000). In complicated handwriting recognition, the outcomes highlight consistency and understanding with 97% total character recognition.

To overcome OCR difficulties, Bora et al. (2020) suggests CNN-ECOC, which substitutes the soft-max layer with CNN's extraction of features and ECOC's categorization. The research investigates four CNN design models using the NIST dataset, which has 1483 images for training as well as 990 testing images for each capital letter of the English language. With an improvement of 0.6% in testing accuracy, AlexNet-ECOC offers insightful data for improved character detection methods. 97.71% accuracy is achieved using AlexNet-ECOC architecture.

With an emphasis on HCR for the Gujarati language, Pareek et al. (2020) discusses the transition to paper-free systems utilizing OCR. The research effort gathers 10,000 handwritten photos for training purposes, with an emphasis on OCR. The approach obtains a 97.21% rate of accuracy with CNN and a 64.48% rate of accuracy with MLP,

which are two kinds of neural networks: CNN and multi-layer perceptron.

To tackle the difficulties in Handwritten Bangla Character Recognition (HBCR), a new method called Handwritten Bangla Digit Recognition (HBDR) is presented by Alom et al. (2017). The suggested approach achieves an impressive 98.78% rate of detection with the CMATERdb 3.1.1 dataset by using CNN with Gabor features and dropout. The 6000 photos in this dataset, which are divided into a training set of 5000 photos and a test set of 1000 photos, feature unstructured handwritten Bangla numbers.

To preserve Shui letters in China, Weng and Xia (2020) investigates the use of CNNs for mobile OCR. It describes the creation of a Shui letter collection and suggests an inexpensive CNN model. The dataset consists of 50000 tagged images of Shui letters. Python-based TensorFlow testing demonstrates the efficacy of the suggested CNN, with a rate of accuracy of roughly 93.3%. To optimize neural network topologies for mobile OCR uses this research offers major findings.

The dataset from HP Labs India is used in B R and Chandrasekaran (2019) which tests CNN for handwritten Tamil character recognition. For an altogether of 82,928 images, around 500 examples per class make up this publicly available dataset. Utilizing a CNN model, the research sets a standard that outperforms conventional techniques with a noteworthy training accuracy rate of 95.16%. The finished model demonstrates its efficacy in handwritten Tamil character recognition with a 97.7% accuracy rate on a testing dataset of 156 classes.

Gan et al. (2019) presents a unique 1-D CNN technique that outperforms 2-D CNN and RNN approaches for online handwritten Chinese character recognition. The ICDAR-2013 testing set has 224,590 images, while the entire training set used in ICDAR 2013 comprises 2,693,183 images. IAHCC-UCAS2016 has 350,612 sample images, divided into training sets of 92 sample images for each class. The approach functions without the need for computation-wise costly approaches, attaining an accuracy of 97.14% on IAHCC-UCAS 2016 and 98.11% on ICDAR 2013 datasets.

Using cutting-edge CNN architectures, the challenge of handling a variety of compound characters was tackled in Ghosh et al. (2021) study on HCR in Bangla. With the CMATERdb dataset, the research used pre-trained CNN models to resize and transform photos to 28 x 28 pixels in black and white. Using unchanged weights, InceptionResNetV2, InceptionNetV3, and DenseNet121 were trained using categorical cross-entropy as the variance function and with a 0.001 rate of learning. DenseNet121 and InceptionNetV3 both reached noteworthy accuracy of 96.55% and 96.20% over 50 epochs, correspondingly, with InceptionResNetV2 demonstrating the best accuracy at 96.99%. Combining DenseNet121, InceptionResNetV2, and InceptionNetV3 produced an accuracy of 97.69%, which was superior to that of separate models, but the high processing needs made the combination less useful.

Pareek et al. (2020) focuses on an offline HCR technique based on artificial intelligence to fulfill the requirement for HCR in Gujarati. 10,000 images from 250 people were used in the data collection process. The proposed methods obtained an accuracy rate of 64.48% with Multi-Layer Perceptron (MLP) and 97.21% with CNN. To create a consistent process for image-to-text conversion, this research places a strong emphasis on word-level identification.

James et al. (2018) presents an AlexNet-based Handwritten Malayalam Character Recognition (HMCR) algorithm. Support Vector Machines (SVM) are used in the research for categorization and CNN for retrieving features. With a high accuracy rate of 98% and a low processing time, the suggested model is specifically made for simple as well as

complicated Malayalam characters. The model shows competence in identifying the 44 fundamental and 36 compounded Malayalam characters by utilizing a dataset of about 180,000 characters for training and testing purposes.

The methodology for handwritten Hindi character recognition using deep learning techniques is presented in Reddy and Babu (2019). It makes use of Deep Feed Forward Neural Networks (DFFNN) as well as CNN with Adam Estimation and RMSprop. Character identification performance of CNNs is highlighted in the research along with the advancement of deep learning techniques. Improvement in efficiency is demonstrated by the results of the trial, where the approach achieves 97.33% (CNN-RMSProp), 95.57% (DFFNN), and 96.02% (CNN-Adam Estimation), detection rates.

Shams et al. (2020) applies a unique algorithm to the important task of handwritten Arabic character recognition and classification. To evaluate the relationship between the input and previously stored models, the method combines SVM and DCNN. The dataset has a total of 16,800 images. When the methodology is compared to cutting-edge techniques, it achieves a low ECR of 4.93% and a high correct classification rate (CRR) of 95.07%, demonstrating its great accuracy.

By utilizing transfer learning from a Devanagari handwritten recognition system, Rani et al. (2020) presents an innovative approach to handwritten Kannada character recognition (KCR). Through the use of the extensive Devanagari dataset, a network based on deep learning method specifically, VGG19 NET is trained for KCR using this technique. As seen by its ability to capture intricate patterns, the VGG19 NET is composed of 5 hidden layers, 2 densely connected layers, along with a single output layer. The training data for the Devanagari character dataset consists of 92,000 photos in 46 classes, whereas the Kannada character dataset is constructed with 81,654 images for training and 9,401 testing images in 188 classes. Through experiments, the suggested model approaches 90% accuracy with an amazing validation accuracy of 73.51% and a loss of 16.18% after 10 epochs with VGG19 NET.

Mariyathas et al. (2020) tackles the difficult task of reading handwritten Sinhala characters, which are characterized by distinctive forms. The study uses the Google Colaboratory platform to conduct tests with CNN, a strong tool for picture categorization. Around 110,000 photos make up a sizable dataset that is used to gradually increase the number of character classes throughout the training and testing phase. The model attains an impressive accuracy of 90.27% after being trained on 100 character classes. After examining the effects of five sets of distinct 100-character classes, the research produced an astounding 82.33% total accuracy for 434 characters.

3 Methodology

3.1 Overview of Devanagari Script

Figure 1: Devanagari Numbers

अ	आ	इ	ई	उ	ऊ	ए	ऐ	ओ	औ	अं	अः

Figure 2: Devanagari vowels

The Brahmic family of scripts, which includes the Devanagari script, is found in India, Tibet, Nepal, and Southeast Asia Bright (1996) Fischer and Fischer (2003). Hindi, Marathi, Nepali, and other comparable languages from Asia's southern and eastern regions are written using the Devanagari script. The 36 basic consonants, 10 numerical characters, 12 vowels, and a few unique characters make up the Devanagari script. Figure 1 displays number characters, Figure 2 displays vowel characters, and Figure 3 displays consonant characters. For every consonant character, there are 12 further derived forms that can be created by wrapping all 36 consonant characters with vowels. The example given in Figure 4 displays derived forms of "ka" and "ra" characters wrapped in vowels.

क	ख	ग	घ	ङ	च	छ	ज	झ	ञ	ट	ठ
ड	ढ	ण	त	थ	द	ध	न	प	फ	ब	भ
म	य	र	ल	व	स	ष	श	ह	क्ष	त्र	ज्ञ

Figure 3: Devanagari consonants

क	का	कि	की	कु	कू	के	कै	को	कौ	कं	कः
र	रा	रि	री	रु	रू	रे	रै	रो	रौ	रं	रः

Figure 4: Derived forms of "ka" and "ra" encased in vowels

3.2 Devanagari Handwritten Character Dataset

A dataset is produced by gathering various handwritten Devanagari characters from people working in many different types of areas. Following scanning, manuscripts are cropped and resized for each character. A character's real size falls inside 28x28 pixels, with every individual character sample image measuring 32x32 pixels. The grayscale conversion was used on the sample images. Subsequently, the sample image contrast was inverted, resulting in the character appearing white against a black background.

Devanagari Numerical (Class)	०	१	२	३	४	५	६	७	८	९
Individual statistics	2000	2000	2000	2000	2000	2000	2000	2000	2000	2000
Total	**20,000**									

Figure 5: Numerical Characters and statistics of dataset

Devanagari Character (Class)	क	ख	ग	घ	ङ	च	छ	ज	झ	ञ	ट
Individual statistics	2,000	2,000	2,000	2,000	2,000	2,000	2,000	2,000	2,000	2,000	2,000
Devanagari Character (Class)	ठ	ड	ढ	ण	त	थ	द	ध	न	प	फ
Individual statistics	2,000	2,000	2,000	2,000	2,000	2,000	2,000	2,000	2,000	2,000	2,000
Devanagari Character (Class)	ब	भ	म	य	र	ल	व	श	ष	स	ह
Individual statistics	2,000	2,000	2,000	2,000	2,000	2,000	2,000	2,000	2,000	2,000	2,000
Devanagari Character (Class)	क्ष	त्र	ज्ञ								
Individual statistics	2,000	2,000	2,000								
Total	**72,000**										

Figure 6: Consonant Characters and statistics of dataset

With 72,000 sample images of the consonant characters and 20,000 sample images of the numeric characters, the dataset has an entire collection of 92,000 character images. 85% dataset (78,200 images) was used for training purposes and 15% dataset (13,800 images) was used for testing purposes. Acharya and Gyawali (2016)Figure 6 and Figure 5 display the dataset statistics for consonant and numeric characters, respectively. Figure 7 shows Sample dataset images.

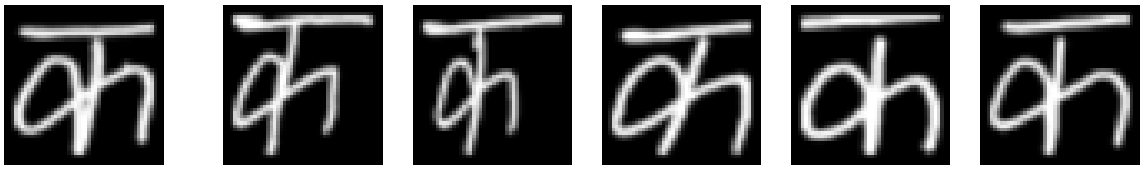

Figure 7: Sample dataset images

3.3 Proposed model

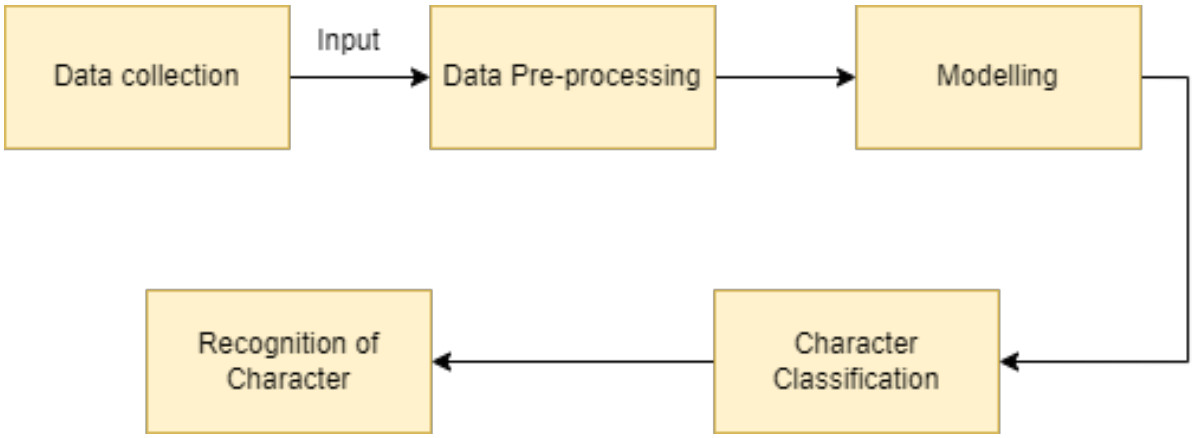

Figure 8: Proposed model block diagram

3.3.1 Data Collection

In the field of deep learning research, this research emphasizes the critical importance that a carefully enriched dataset plays in achieving precise outcomes. We focused on the Devanagari Handwritten Character Dataset throughout the research stage. The data set is available to the public and is stored in the UCI repository. A thorough explanation of the data is provided in the previous section.

3.3.2 Data Pre-processing

Using the Devanagari Handwritten Character Dataset, a data preprocessing process seeks to improve the deep learning-based model's resilience and adaptability skills for HCR. A

variety of augmentation methods are used for the training dataset for the training stage, including rotation, shear, rescaling, width and height shifts, and zoom. To add even more variability. This process involves transforming the images to greyscale. The pictures are also resized to a consistent 32x32 pixel. The purpose of these pre-processing steps is to simplify data for character recognition. The 32x32 pixel format guarantees consistency for effective model training and analysis, while grayscale streamlines the dataset while preserving important features.

3.3.3 Data Modelling

3.3.3.1 CNN: Bioinspired CNNs are powerful machine-learning frameworks that are especially well-known for their effectiveness in video as well as image identification. CNNs are now considered a mainstay in machine learning applications, with exceptional performance in picture categorization, object identification, and recognition of facial features. CNNs were created to mimic the visual information processing of the brain of an individual Li et al. (2021) O'Shea and Nash (2015). Three main elements comprise the foundational idea of CNNs:

- Convolutional Layers: The fundamental building blocks for the extraction of features that utilize filters for input data are called convolutional layers. Specific designs and characteristics are captured by filters, also called kernels, as they move across the source image. Feature maps representing tiered visual representations are generated by convolutional procedures.

- Pooling Layers: The feature maps that are produced by the convolutional layers have their spatial dimensions regularly reduced by pooling layers. In pooling, the highest or median value from a specified spatial area is chosen, which results in downsampling. Translational consistency is improved and the complexity of computation is decreased by this technique.

- Fully Connected Layers: For the final stage of categorization, fully-connected layers combine the top-level features that were collected by the pooling as well as convolutional layers. All of the neurons in the layer above are linked to every neuron in a fully connected layer. To decompose complex features and make decisions, activation functions, and weights are used.

3.3.3.2 InceptionV3: CNN advancement InceptionV3 performs exceptionally well in image recognition. Inception modules, Global Average Pooling (GAP), Auxiliary Classifiers, and Factorized Convolutions are the key features of InceptionV3. For multiscale extraction of features, it makes use of inception modules, which have concurrent convolutional procedures with different kernel sizes. Conventional fully connected layers are replaced with GAP, which improves the generalization of the model and processing performance. For robust feature-based learning, auxiliary classifiers are added during training to solve the problem of vanishing gradients. 1x1 convolutions are used in factorized convolutions to maximize processing power while maintaining important data. Because of its architecture, InceptionV3 can accomplish excellent accuracy while maintaining its comparatively light construction. Highly effective learning of smaller-scale features is facilitated by the auxiliary classifiers, which are missing during interpretation and present during training. With factorized convolutions, InceptionV3's computational effectiveness is increased and its adaptability to various visual patterns is increased Chollet (2017).

3.3.3.3 Xception: With its depthwise separated convolutions, which provide greater effectiveness and descriptive ability, Xception is a sophisticated CNN model. It achieves outstanding efficiency in image classification challenges, outperforming conventional designs with its considerable depth. Using both depthwise as well as point-by-point procedures, the novel architecture captures complex features with less computing overhead. Rather than using completely linked layers like in traditional networks, Xception uses global average pooling, which improves generalization and lowers variables. Stable training is aided by auxiliary classifiers. Xception is a powerful and effective CNN thanks to its novel design, which is characterized by depthwise separated convolutions Szegedy et al. (2016).

3.3.4 Character Classification

A machine learning function known as "character classification" aims to classify each character into specified groupings or classes. To train algorithms to recognize unique characteristics and trends linked to various characters, the method requires a labeled dataset. Character classification is difficult since there are variances in writing styles and fonts that need solid models to classify characters accurately. CNNs are a powerful tool for character classification that effectively utilizes deep learning to recognize patterns. CNNs are useful for character classification because they can effectively capture and detect complex patterns by utilizing the depth of learning, which improves the classification process's accuracy and stability.

4 Design Specification

The section covers the specifications that fall within the scope of the research as well as the flow of the process. Figure 9 shows the Process flow of the Model. Below is a detailed explanation of each design block.

The dataset must be loaded as the initial stage in the procedure. This data is preprocessed to improve the image quality and to reduce the noise from the image. The dataset was obtained from the UCI repository and is already divided into two subsets: training and testing sets. In the next stage, 3 models were trained on a respective dataset subset. Two pre-trained models namely the InseptionV3 and Xception models with dense layers are used for transfer learning purposes. The Proposed custom CNN model is created from scratch. Statistics like training accuracy and loss are recorded for all three models.

A Python library, TensorFlow, is used to train the models. The TensorFlow library offers an adaptable framework for neural network building and efficiency, forming a strong basis for training deep learning models. The three models are tested on a testing dataset during the evaluation phase, which is the final stage of implementation. The models are put through a rigorous testing process on a specific dataset during the evaluation phase, which enables an in-depth evaluation of their effectiveness. This evaluation makes inferences to determine the models' efficacy in addition to evaluating their performance. The next implementation section will explore every detail of this process, giving a thorough explanation of each stage.

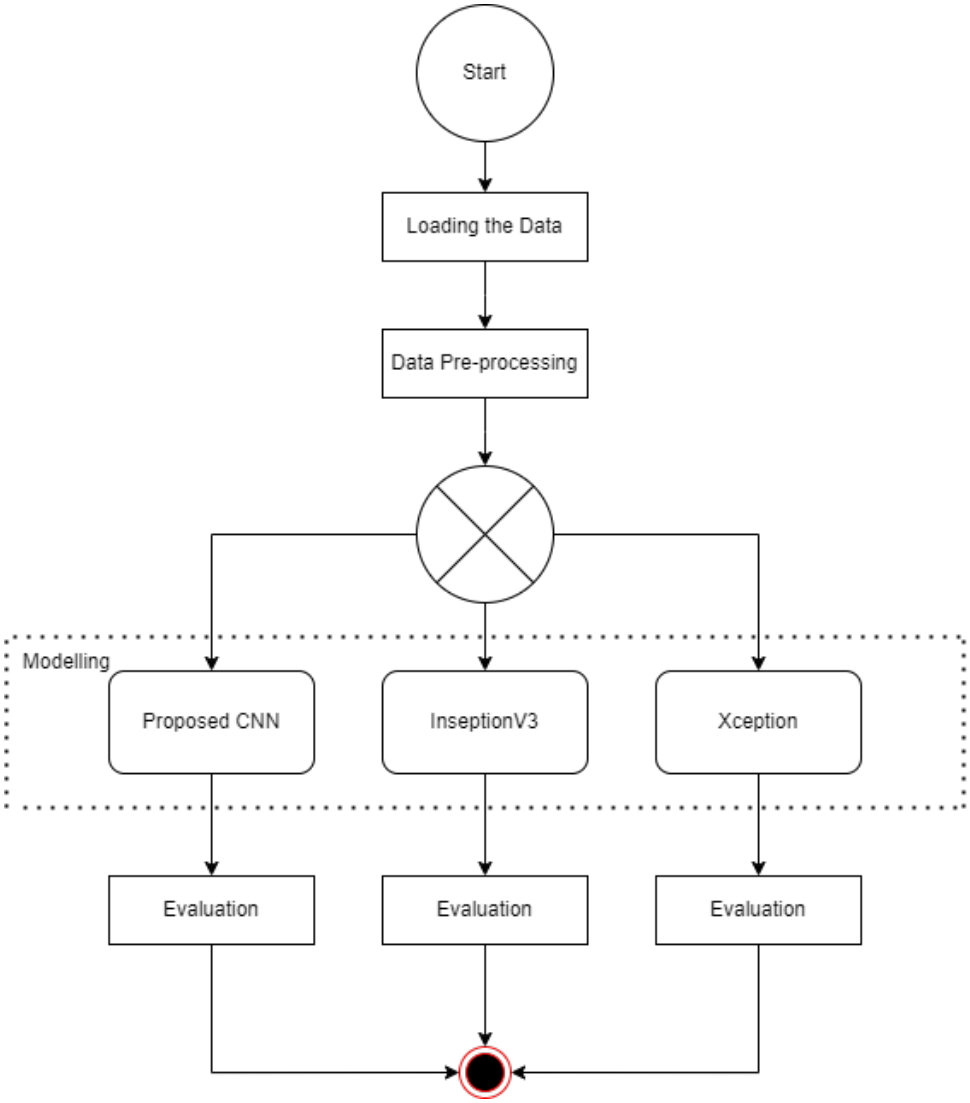

Figure 9: Process flow Diagram

5 Implementation

The detailed configuration of the models used for the training of data is explained in this section.

5.1 CNN Model

Figure 10 and Figure 11 show the CNN Model Design and Architecture of the proposed CNN model, respectively. The model consists of 3 convolution layers, 3 batch Normalization layers, 3 max pooling layers, 2 dense layers, 2 dropout layers, a flatten layer, and an output layer. This model's input is a greyscale image of the Devnagari handwritten Characters with 32×32 pixels. There are roughly 252,334 trainable parameters in the presented model. The first layer produces a 30x30x32 image by convolutioning 32, 3x3 shaped frames. The following two convolutional layers go through the same procedure. 32, 64, and 128 filters are used for 3 convolutional layers, respectively. One max pooling layer and one batch normalization layer are used after each convolutional layer to

Layer (type)	Output Shape	Parameters
Convolution 2D	(None, 30, 30, 32)	320
Batch normalization	(None, 30, 30, 32)	128
Max pooling	(None, 15, 15, 32)	0
Convolution 2D	(None, 13, 13, 64)	18,496
Batch normalization	(None, 13, 13, 64)	256
Max pooling	(None, 7, 7, 64)	0
Convolution 2D	(None, 5, 5, 128)	73,856
Batch normalization	(None, 5, 5, 128)	512
Max pooling	(None, 3, 3, 128)	0
Flatten	(None, 1152)	0
Dense	(None, 128)	1,47,584
Batch normalization	(None, 128)	512
Dropout	(None, 128)	0
Dense	(None, 64)	8,256
Batch normalization	(None, 64)	256
Dropout	(None, 64)	0
Dense	(None, 46)	2,990
Total parameters:	2,53,166	
Trainable parameters:	2,52,334	
Non-trainable parameters:	832	

Figure 10: CNN Model Design Configuration

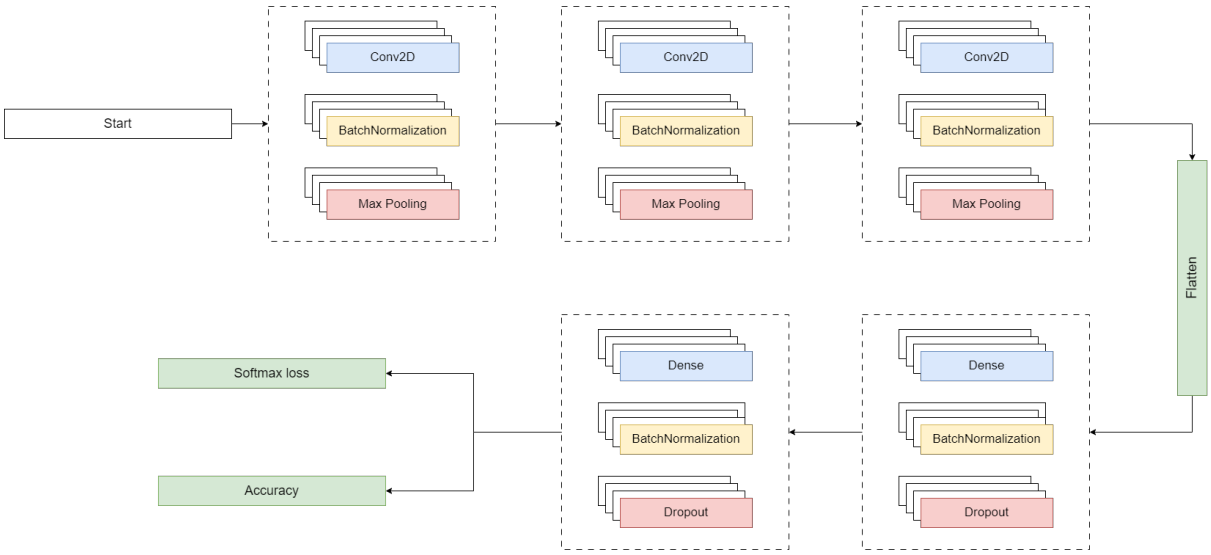

Figure 11: Architecture of the proposed CNN model

determine the data's key characteristics and enhance the steadiness of training, respectively. A one-dimensional vector is created by flattening the multi-dimensional output of the convolutional and max pooling layers that came before it. After flattening the layer, 2 dense layers are used to choose the features from the image. 128 and 64 neurons are used by 2 dense layers, respectively. The Rectified Linear Unit (ReLU) activation function is used in both dense layers for forwarding positive inputs unaltered and producing zero for negative inputs, which adds non-linearity as well as computational effectiveness. Dropout layers are used after each dense layer to reduce overfitting when training. 46 neurons connect every class in the output layer. The output layer uses the Softmax activation function. A neural network's weights are updated during training using the Adam Optimizer.

5.2 InseptionV3 Model

Figure 12 show InseptionV3 Model Design. The design of a model incorporates transfer learning using InceptionV3, which was pre-trained on the ImageNet dataset. Additionally, the design consists of a flatten layer, a dense layer with 512 neurons and ReLU as an activation function, a batch Normalization layer to enhance the steadiness, a Dropout layer to prevent excessive overfitting of the model, and an Adam Optimizer. The last layer adjusts the model for multi-class categorization by matching the number of neurons with each of the different classes in the dataset using the softmax activation function. There are roughly 10,73,710 trainable parameters in the model. Utilizing prior knowledge while adjusting to the complexity of various image categorization jobs is the goal of this InseptionV3 model design. The model has almost a million trainable parameters. Figure 13 shows InseptionV3 Model architecture.

Layer (type)	Output Shape	Parameters
InceptionV3 (Functional)	(None, 1, 1, 2048)	2,18,02,784
Flatten	(None, 2048)	0
Dense	(None, 512)	10,49,088
Batch normalization	(None, 512)	2,048
Dropout	(None, 512)	0
Dense	(None, 46)	23,598
Total parameters:	2,28,77,518	
Trainable parameters:	10,73,710	
Non-trainable parameters:	2,18,03,808	

Figure 12: InseptionV3 Model Design Configuration

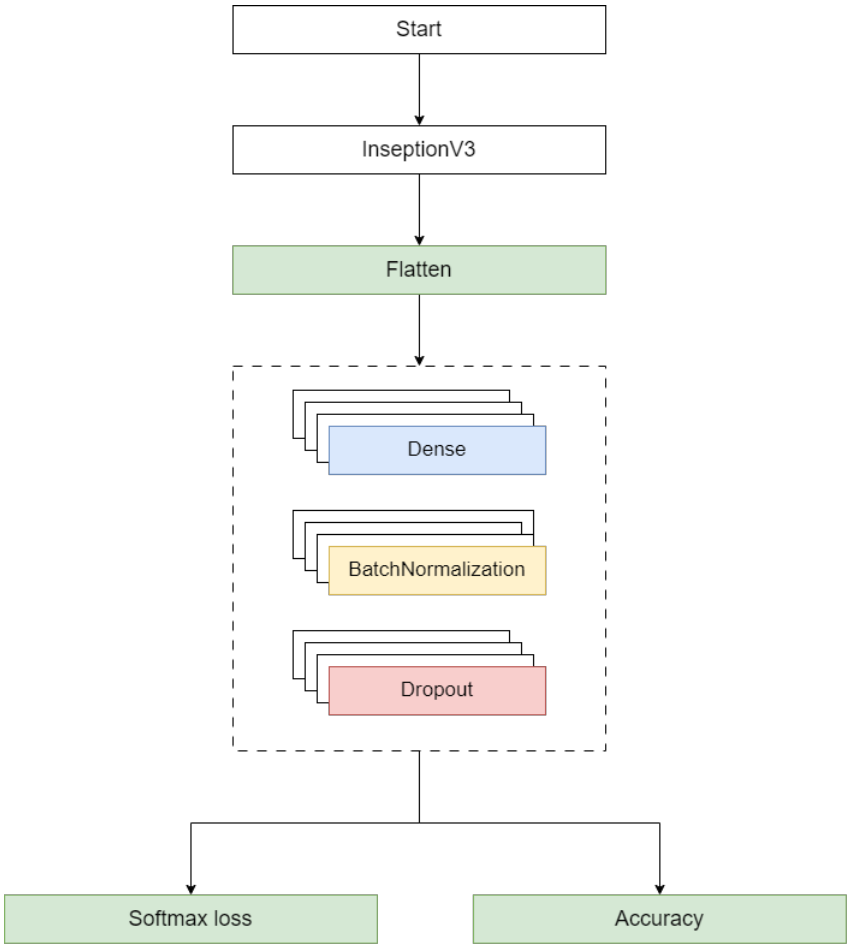

Figure 13: InseptionV3 Model architecture

5.3 Xception Model

Layer (type)	Output Shape	Parameters
Xception (Functional)	(None, 3, 3, 2048)	20861480
Flatten	(None, 18432)	0
Dense	(None, 512)	9437696
Batch normalization	(None, 512)	2,048
Dropout	(None, 512)	0
Dense	(None, 46)	23,598
Total parameters:	3,03,24,822	
Trainable parameters:	94,62,318	
Non-trainable parameters:	2,08,62,504	

Figure 14: Xception Model Design Configuration

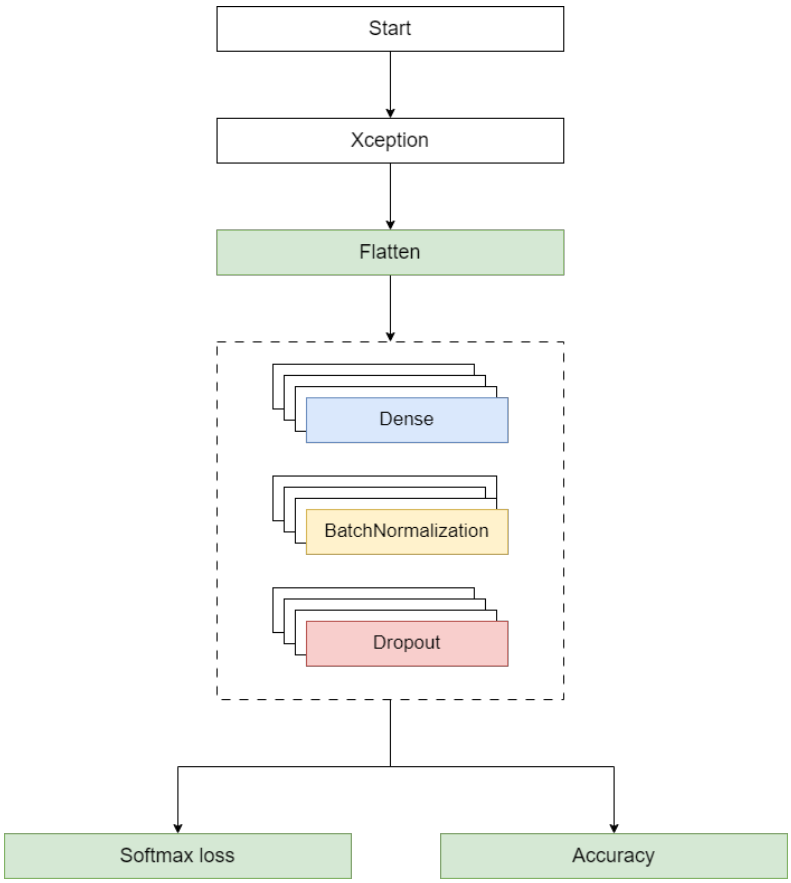

Figure 15: Xception Model architecture

Figure 14 show Xception Model Design. With depth-wise separable convolutions,

Xception, an adaptation of the Inception design, stresses the division of feature learning as well as spatial computation. The Xception model is also trained on the ImageNet dataset. The model design is identical to the inception model configuration that was previously explained in the above section. The model has almost 9.5 million trainable parameters. Figure 15 shows Xception Model architecture.

5.4 Model Configuration

Model	Input	Batch	Epochs	Optimizer	Trainable
Proposed CNN	(32, 32, 1)	32	20	Adam	2,52,334
InceptionV3	(75, 75, 3)	64	20	Adam	1,073,710
Xception	(71, 71, 3)	64	15	Adam	9,462,318

Figure 16: Configuration of each model

Figure 16 presents the settings of three models that utilize the same dataset. 64-image batches from the training directory are used by TensorFlow to train each of the models. The accuracy is determined after every epoch. Pre-trained weights are used in the Inception as well as the Xception network model. 49 classes are predicted by every model. For evaluation, every accuracy is documented.

6 Evaluation

Three different deep learning models: Proposed CNN, InceptionV3, and Xception are thoroughly evaluated in this research to retrieve handwritten characters from an image. The models' evaluation is done based on their accuracy over several epochs.

6.1 Proposed CNN

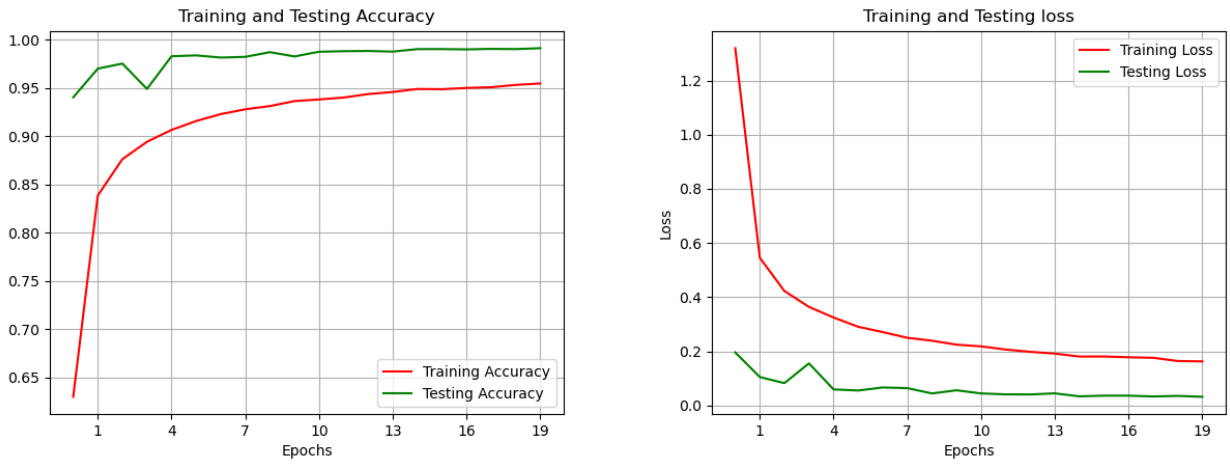

Figure 17: Proposed CNN: training vs. testing accuracy and loss graphs

After 20 epochs cycle, the proposed CNN model obtained an impressive accuracy of 99.11%. Strong learning capabilities are demonstrated by the model, which effectively captures complex aspects of the Devanagari script. Figure 17 shows the training vs. testing accuracy and loss graphs against a number of epochs.

6.2 InceptionV3

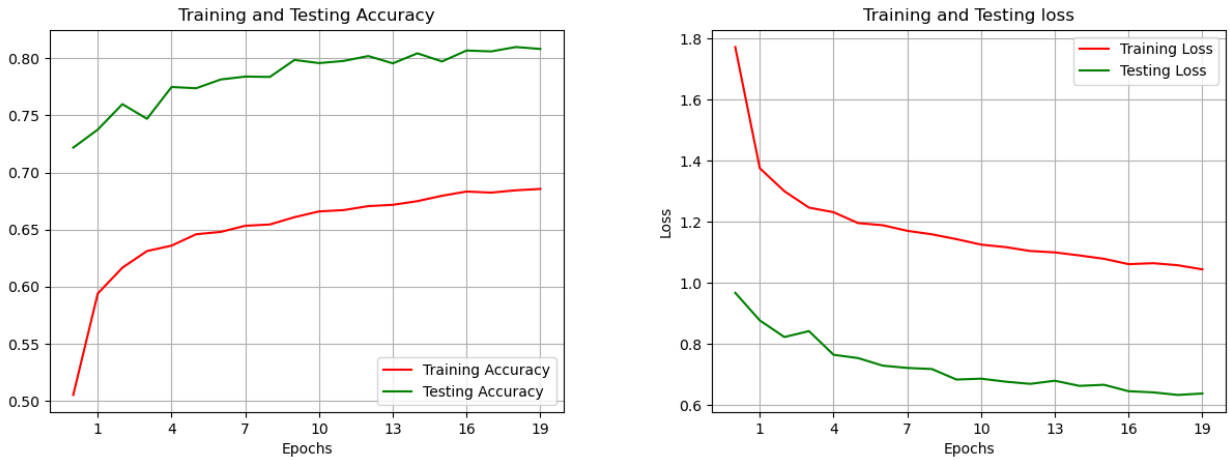

Figure 18: InseptionV3: training vs. testing accuracy and loss graphs

InceptionV3 lags significantly below the Proposed CNN in terms of accuracy, despite displaying a respectable accuracy of 80.82% after 20 epochs. Figure 18's visualization of the model's training accuracy vs. testing accuracy graph shows a gradual rise in efficiency.

6.3 Xception

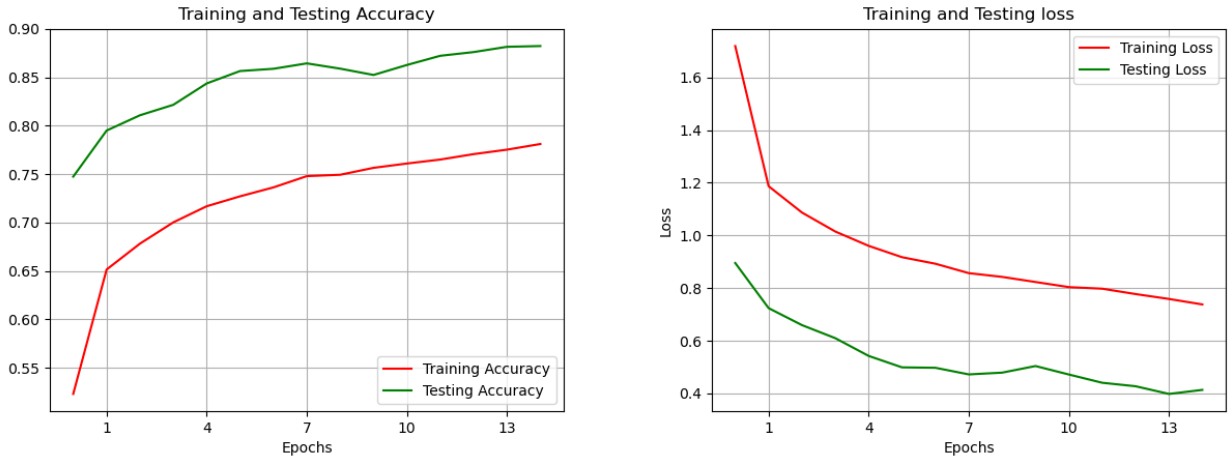

Figure 19: Xception: training vs. testing accuracy and loss graphs

At 88.22% accuracy after 15 epochs, the Xception model finds an acceptable level between the InceptionV3 and Proposed CNN models. Continuous progress is seen in

the training accuracy vs. testing accuracy curve (Figure 19), which is in line with the intricacy of Devanagari handwritten script characters.

Figure 20 shows a Comparison of all three models with testing and training accuracy.

Model	Training accuracy in %	Testing accuracy in %
Proposed CNN	95.45	99.11
InceptionV3	68.56	80.82
Xception	78.10	88.22

Figure 20: Comparison of all models

6.4 Discussion

The new research achieves noteworthy improvements in both dataset size and model performance, making it a substantial contribution to the study of Devanagari Handwritten Character Recognition (DHCR) when compared to previous works. Deore and Pravin (2020) used a dataset of 5,800 images and a two-stage VGG16 model. In this research, the testing accuracy of the first design was 94.84%, while the accuracy of the second model was further enhanced to 96.55%. While this research was a step forward, it is clear that higher-level models and bigger datasets are required to produce good results. Gurav et al. (2020) used DCNNs to achieve an excellent accuracy rate of 99.65% using a larger dataset of 34,604 photos. This research showed that near-perfect recognition accuracy can be achieved by utilizing large datasets and cutting-edge neural network designs.

The proposed research used a large dataset of 92,000 images, which is larger in scale than the two previous studies combined. The proposed CNN created for this research demonstrated the efficiency with an astounding accuracy of 99.11%. Despite using a much larger dataset, the research's notable enhancement over the previous research studies indicates the significance of both dataset size and the structure and tuning of the neural network architecture used.

With a focus on the influence of the dataset size, complexity of models, and architectural selections, these results demonstrate how DHCR research is developing. The proposed research emphasizes how combining large datasets with customized neural network designs can lead to additional improvements in the efficiency and accuracy of DHCR systems.

7 Conclusion and Future Work

There have been notable improvements in character recognition as a result of this research into the subject of Devanagari handwritten character extraction utilizing Proposed CNN, Xception, and InceptionV3 models. After being thoroughly tested for accuracy and efficiency, the models have revealed important details about their strengths and weaknesses. With a remarkable accuracy of 99.11% after 20 epochs, the Proposed CNN emerged as

the best model. For Devanagari character extraction, the proposed CNN model is a very powerful solution due to its accuracy and capacity to generalize to new inputs.

Future objectives for research include investigating a wider range of parameters, large datasets, and strategies to obtain the best results from the model. The best configuration for maximizing accuracy can be found by thoroughly researching hyperparameter tuning, which includes experimenting with different filter sizes, learning rates, optimizers, functions, and batch sizes. Using transfer learning approaches, such as pre-training on an extensive dataset and improving on the Devanagari script character dataset, could help CNN perform even better by utilizing knowledge from various resources. Finally, the practical implementation of the CNN model in various contexts can be enhanced by examining its capacity and effectiveness for real-time use cases.

References

Acharya, S. and Gyawali, P. (2016). Devanagari Handwritten Character Dataset, UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5XS53.

Ahlawat, S., Choudhary, A., Nayyar, A., Singh, S. and Yoon, B. (2020). Improved handwritten digit recognition using convolutional neural networks (cnn), *Sensors* **20**(12): 3344.

Alom, M. Z., Sidike, P., Taha, T. M. and Asari, V. K. (2017). Handwritten bangla digit recognition using deep learning.

Alrehali, B., Alsaedi, N., Alahmadi, H. and Abid, N. (2020). Historical arabic manuscripts text recognition using convolutional neural network, *2020 6th Conference on Data Science and Machine Learning Applications (CDMA)*, pp. 37–42.

B R, K. and Chandrasekaran, S. (2019). Benchmarking on offline handwritten tamil character recognition using convolutional neural networks, *Journal of King Saud University - Computer and Information Sciences* **34**.

Bora, M. B., Daimary, D., Amitab, K. and Kandar, D. (2020). Handwritten character recognition from images using cnn-ecoc, *Procedia Computer Science* **167**: 2403–2409.

Bright, W. (1996). The devanagari script, *The world's writing systems* pp. 384–390.

Chaudhary, D. and Sharma, K. (2019). Hindi handwritten character recognition using deep convolution neural network, *2019 6th International Conference on Computing for Sustainable Global Development (INDIACom)*, pp. 961–965.

Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions, *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1251–1258.

Deore, S. P. and Pravin, A. (2020). Devanagari handwritten character recognition using fine-tuned deep convolutional neural network on trivial dataset, *Sādhanā* **45**: 1–13.

Fischer, S. R. and Fischer, S. R. (2003). *History of writing*, Reaktion books.

Gan, J., Wang, W. and Lu, K. (2019). A new perspective: Recognizing online handwritten chinese characters via 1-dimensional cnn, *Information Sciences* **478**: 375–390. **URL:** *https://www.sciencedirect.com/science/article/pii/S0020025518309241*

Ghosh, T., Abedin, M.-H.-Z., Al Banna, H., Mumenin, N. and Abu Yousuf, M. (2021). Performance analysis of state of the art convolutional neural network architectures in bangla handwritten character recognition, *Pattern Recognition and Image Analysis* **31**: 60–71.

Gurav, Y., Bhagat, P., Jadhav, R. and Sinha, S. (2020). Devanagari handwritten character recognition using convolutional neural networks, *2020 International Conference on Electrical, Communication, and Computer Engineering (ICECCE)*, IEEE, pp. 1–6.

James, A., Manjusha, J. and Saravanan, C. (2018). Malayalam handwritten character recognition using alexnet based architecture, *Indonesian Journal of Electrical Engineering and Informatics (IJEEI)* **6**(4): 393–400.

Li, Z., Liu, F., Yang, W., Peng, S. and Zhou, J. (2021). A survey of convolutional neural networks: analysis, applications, and prospects, *IEEE transactions on neural networks and learning systems* .

Liu, B., Xu, X. and Zhang, Y. (2020). Offline handwritten chinese text recognition with convolutional neural networks, *arXiv preprint arXiv:2006.15619* .

Mariyathas, J., Shanmuganathan, V. and Kuhaneswaran, B. (2020). Sinhala handwritten character recognition using convolutional neural network, *2020 5th International Conference on Information Technology Research (ICITR)*, pp. 1–6.

O'Shea, K. and Nash, R. (2015). An introduction to convolutional neural networks, *arXiv preprint arXiv:1511.08458* .

Pareek, J., Singhania, D., Kumari, R. R. and Purohit, S. (2020). Gujarati handwritten character recognition from text images, *Procedia Computer Science* **171**: 514–523.

Ptucha, R., Such, F. P., Pillai, S., Brockler, F., Singh, V. and Hutkowski, P. (2019). Intelligent character recognition using fully convolutional neural networks, *Pattern recognition* **88**: 604–613.

Rani, N. S., Subramani, A., Kumar P., A. and Pushpa, B. (2020). Deep learning network architecture based kannada handwritten character recognition, *2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA)*, pp. 213–220.

Reddy, R. V. K. and Babu, U. R. (2019). Handwritten hindi character recognition using deep learning techniques, *Int J Comput Sci Eng* .

Shams, M., Elsonbaty, A., ElSawy, W. et al. (2020). Arabic handwritten character recognition based on convolution neural networks and support vector machine, *arXiv preprint arXiv:2009.13450* .

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. and Wojna, Z. (2016). Rethinking the inception architecture for computer vision, *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826.

Weng, Y. and Xia, C. (2020). A new deep learning-based handwritten character recognition system on mobile computing devices, *Mobile Networks and Applications* **25**.

Yang, J., Ren, P. and Kong, X. (2019). Handwriting text recognition based on faster r-cnn, *2019 Chinese Automation Congress (CAC)*, pp. 2450–2454.