Abstract

Transfer learning is a research domain in machine learning with a lot of attention today due to its ability to apply pre-trained models across various domains, enhance model performance with limited data and accelerate training processes. Similarly, handwritten text recognition is an increasingly relevant research field due to its practical applications in today's digital landscape. The integration of these two domains presents an exciting opportunity to advance recognition systems, with significant practical implications. This study explores the application of deep learning techniques, utilizing transfer learning, to recognize Malayalam handwritten characters. The work provides a broad exploration of applying transfer learning for Optical Character Recognition (OCR) systems, addressing the associated challenges and potential future research directions. A comprehensive literature review is presented, focusing on research in the domain of Malayalam Handwritten Character Recognition. An integrated Malayalam character dataset is created to train and evaluate multiple established neural networks, including DenseNet201, ResNet50, and LeNet, providing a comparative analysis of their performance. The key findings point to the importance of pre-processing in enhancing input data quality and fine-tuning the models to optimize performance. However, it is important to address the challenges of overfitting and requirements for large input data size for training complex models. The future scope lies in exploring hybrid approaches, dataset expansion and advancing towards word or document recognition.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The idea of replicating human functions through machines has long fascinated man's imagination[4][5]. Technology has evolved from the early mechanical automation to the current advanced artificial intelligence as humans strive to mimic their own abilities and beyond. As technology advances rapidly, the dream of machines that can read, write, and ultimately learn and think like humans is within reach. Through Optical Character Recognition(OCR) systems, machines that can read has become a reality. OCR allows us to convert data from text-containing documents such as scanned images of printed or handwritten text, into a digital and editable format, enabling further processing. Thus OCR is critical in converting the huge volumes of unstructured data into searchable formats for a wide-range of applications. Today, the technology has applications across a diverse range of domains, spanning from office automation to banking and finance to communication and beyond.

Handwritten Character Recognition (HCR) stands as a fundamental and challenging problem within the field of OCR. It enables digitizing handwritten text from documents, significantly enhancing data collection and processing capabilities. The technology can be used for conversion of historical manuscripts into searchable formats for preservation and research, extraction of data from hand-written notes, forms, signatures etc. Thus HCR not only facilitates easier organization and management of data, but also improves information accessibility. However, the variations and inconsistencies in handwriting style, shape or size of different individuals make recognizing patterns and identifying the characters a challenging problem. Hence, the successful integration of the technology into various domains demands addressing these challenges related to robustness and accuracy.

This thesis explores the application of deep learning to the problem of HCR for Malayalam language characters. This work offers a broad exploration of transfer learning to provide a comparison of the performance, in terms of accuracy and robustness, of various common Convolutional Neural Network (CNN) architectures for the problem.

Chapter 2

Theory

2.1 Optical Character Recognition

Today, with the rise of digitization, there is an increasing demand to transfer information available in printed or handwritten documents to computer storage. Simply scanning and storing them as image files limits the re-utilization capability of the information. Hence there is a need for technology that can automatically retrieve information from images and documents and store them in editable (text) format for further processing. OCR is currently a research domain gathering significant interest that seeks to modify any form of text-containing data such as handwritten, printed or scanned text images or documents into an editable digital format for further processing[6]. It is of immense significance within the field of computer vision research as it makes huge amounts of unstructured data more accessible and searchable[7].

The OCR process involves text detection and text recognition[7]. Initially the text containing regions are detected and later the text within the identified regions are recognized or discerned. Text recognition can include both printed and handwritten content. Handwritten recognition can be further divided into online and offline recognition[7]. In contrast to the real-time online recognition where the system recognizes the text by identifying the strokes, stroke order or other features as it is being drawn or written, offline recognition comes into picture after the writing or printing is completed[4][7].

2.1.1 Historical Background

The concept of OCR was initially proposed by the German scientist Tausheck in 1929[7]. The first true OCR machine was installed at Reader's Digest in 1954, which was used to convert typed reports into punched cards[4]. The first generation of OCR systems, the commercial systems developed between 1960 to 1965, recognized specially designed and constrained letters and symbols[4]. The technology was very limited and recognized only symbols specifically designed for machine reading. The systems that appeared between 1965 and 1975 may be called the second generation of OCR[4]. These machines could handle regular machine printed characters and a limited set of hand-printed letters and numerals. IBM's 1287 and Toshiba's letter sorting machine for postal code numbers are notable advancements during this generation of OCR[4]. The third generation of OCR, machines developed from the mid-1970s, focused on handling poor quality documents and a broader range of printed and handwritten characters[4]. The availability of affordable hardware made OCR systems both commercially viable and widely accessible starting in the mid-1980s.

2.1.2 Components of OCR

The main components of OCR can be identified as pre-processing, feature extraction and classification[4][5][7][8].

Pre-Processing

After the digital image of the original document is captured through scanning, there involves a series of steps before they are ready for text recognition. Text recognition essentially involves identifying and interpreting patterns from digital images to recognize characters or words. But often, to provide effective detection of patterns from these textual images, pre-processing is required to remove noise, enhance clarity and prepare the input for further processes[4][8]. Pre-processing involves removing noise or undesired characteristics from the image, without leaving out useful information[6]. Pre-processing enhances the effectiveness and easiness with which the input can be handled in the later phases by reducing inconsistent data and noise. Pre-processing in a way ensures the reliability and standardization of the input for successive phases, thus directly influencing recognition rates.

Pre-processing can involve a number of operations such as threstholding, noise reduction, skew correction, segmentation, morphological operations etc. depending on the input type and quality[4][6].

Feature Extraction

Feature extraction in OCR involves identifying and isolating key features from the character images to facilitate recognition. Feature extraction tries to extract the most relevant information from raw data to optimize classification. The goal is to reduce the variability within the same class of patterns while enhancing the distinction between different classes, thereby improving the accuracy of classification tasks[9]. A number of features have been studied in the literature that help OCR systems recognize characters. Features of a character can often be categorized as global or statistical features and structural or topological features[10]. These are used to categorize different characters. The different groups of features may be evaluated according to their sensitivity to noise and deformation and the ease of implementation and use for the datasets available.

Classification

The actual character recognition happens in this phase of OCR, where the extracted features are analyzed to determine the different classes of characters. This process involves comparing the features of an input character with pre-defined patterns or models using machine learning algorithms.Various algorithms and methods have been extensively explored in the literature for classification purposes. The design of OCR systems generally relies on several key approaches, which can be broadly categorized into template matching, fuzzy logic, feature extraction, structural analysis, and neural networks.[6][11]. The goal of the classification step is to accurately match the input features to a specific character from the system's trained character set, ensuring correct text recognition. Classification is carried out based on the stored features in the feature space, such as structural features, global features, and others. Essentially, classification divides the feature space into distinct classes by applying decision rules. The choice of classifier depends on various factors, including the number of parameters, the size and quality of the available training set, and the specific characteristics of the problem being addressed[6].

2.2 Artificial Neural Networks

An Artificial Neural Network (ANN) is a parallel computational model inspired by the biological neural systems. Similar to a biological nervous system, an ANN is

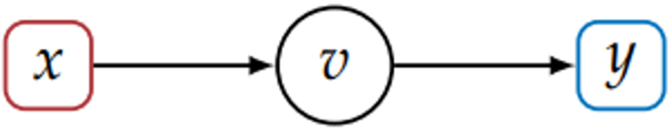

Figure 2.1: A simple neural network with single input

composed of layers of simple information processing units called neurons, which gather inputs and produce outputs[12]. In a neural network, each node processes the information received through simple calculations involving an activation function, and produces a transformed signal to the other nodes (Figure. 2.2). The extend to which the signals are amplified or diminishes, thus determining the connection between nodes, is indicated by 'weight' between nodes. Training an ANN model essentially involves learning the weights of connections between the nodes. Thus, ANN training can be considered an unconstrained nonlinear minimization problem[13]. Individually considering, the nodes are not very powerful units, with each of them producing a single scalar output through a simple non-linear function of the inputs. However, the combination of multiple nodes enables the network to perform complex tasks rather efficiently. The collective power of interconnected neurons allows the model to implement complex functions, thus enabling it to solve sophisticated problems[12]. q

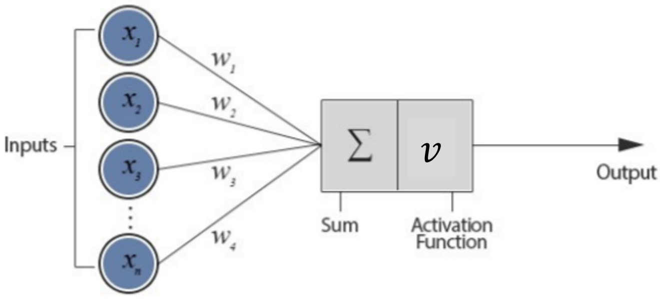

Figure 2.2: A simple ANN architecture

ANN is a data-driven approach in which the model is trained to adjust the weights between nodes through a process that minimize the errors in output predictions. This not only allows the model to learn from examples, but also generalize well to previously unseen data[13]. This makes ANNs well-suited for problems where the underlying relationships or pattern among the data are difficult to discern, but there are enough examples to learn from[13]. ANNs are considered universal approximators, capable of approximating any continuous function within certain constraints to a desired level of accuracy[12][13]. As a result, ANNs are utilized across various fields, including engineering, medicine, finance, management, and agriculture, addressing challenges that traditional methods may find difficult to solve[14]. Although ANNs are a well-researched field, they still face several challenges, including enhancing model robustness, increasing transparency, addressing data requirements, and mitigating their black-box nature[14].

2.3 Deep Learning

Deep learning, a concept introduced by Hinton in 2006, has revolutionized the field of artificial intelligence by enabling models to learn from vast amounts of data and interpret complex features, thus leading to breakthroughs across a wide range of applications[15]. Deep networks are artificial neural networks with multiple hidden layers or layers of weight. The field of machine learning that focuses on deep networks is known as deep learning[16]. Deep networks are able to discern high level features from large datasets through unsupervised or semi-supervised learning techniques, thus making them superior to traditional machine learning algorithms[17]. Such networks enable the initial layers to capture high-level features, and subsequent deeper layers to refine the information required to discern complex features or information required to make accurate predictions[17]. This results in a hierarchical feature extraction model that is complex and capable.

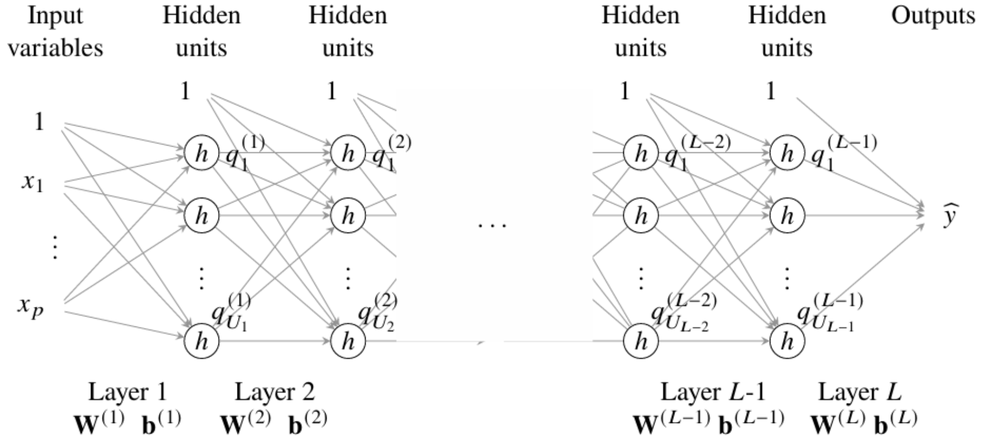

Figure 2.3: A deep neural network with L layers[1]. Each layer is parameterised by W and b.

2.4 Convolutional Neural Networks

While deep networks can be applied to different types of data, including images, it can become overly complex due to the large number of parameters required, making training inefficient. Dense layers, where each input variable is connected to all hidden units in the subsequent layer, have been found to provide too much flexibility for images, thus resulting in overfitting[1]. Convolutional Neural Networks (CNNs) are specialized neural networks designed for tasks involving grid-like input data structures, such as images[1]. CNNs employ convolutional layers that leverage the concepts of sparse interactions and parameter sharing to exploit the structure present in images, allowing them to find efficient parameterized models[1].

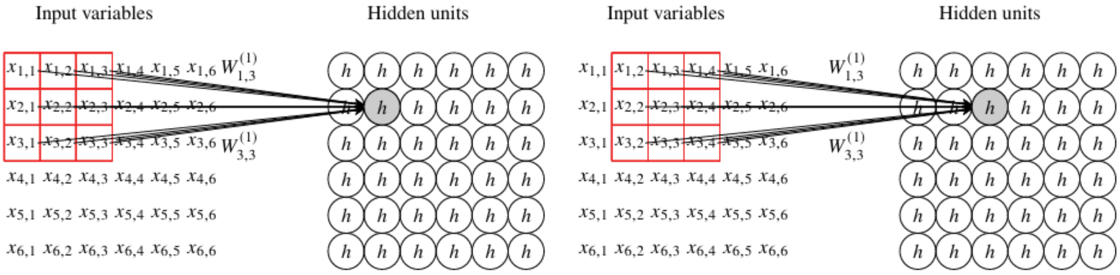
Input variables
Hidden units
$x_{1,1}$ $x_{1,2}$ $x_{1,3}$ $x_{1,4}$ $x_{1,5}$ $x_{1,6}$
$x_{2,1}$ $x_{2,2}$ $x_{2,3}$ $x_{2,4}$ $x_{2,5}$ $x_{2,6}$
$x_{3,1}$ $x_{3,2}$ $x_{3,3}$ $x_{3,4}$ $x_{3,5}$ $x_{3,6}$
$x_{4,1}$ $x_{4,2}$ $x_{4,3}$ $x_{4,4}$ $x_{4,5}$ $x_{4,6}$
$x_{5,1}$ $x_{5,2}$ $x_{5,3}$ $x_{5,4}$ $x_{5,5}$ $x_{5,6}$
$x_{6,1}$ $x_{6,2}$ $x_{6,3}$ $x_{6,4}$ $x_{6,5}$ $x_{6,6}$
$W^{(1)}_{1,3}$
$W^{(1)}_{3,3}$
h
Input variables
Hidden units
$x_{1,1}$ $x_{1,2}$ $x_{1,3}$ $x_{1,4}$ $x_{1,5}$ $x_{1,6}$
$x_{2,1}$ $x_{2,2}$ $x_{2,3}$ $x_{2,4}$ $x_{2,5}$ $x_{2,6}$
$x_{3,1}$ $x_{3,2}$ $x_{3,3}$ $x_{3,4}$ $x_{3,5}$ $x_{3,6}$
$x_{4,1}$ $x_{4,2}$ $x_{4,3}$ $x_{4,4}$ $x_{4,5}$ $x_{4,6}$
$x_{5,1}$ $x_{5,2}$ $x_{5,3}$ $x_{5,4}$ $x_{5,5}$ $x_{5,6}$
$x_{6,1}$ $x_{6,2}$ $x_{6,3}$ $x_{6,4}$ $x_{6,5}$ $x_{6,6}$
$W^{(1)}_{1,3}$
$W^{(1)}_{3,3}$
h

Figure 2.4: A convolutional layer illustration[1]. Each hidden unit is only dependent on the pixels in a small region of the image, here of size 3 × 3 pixels. The location of the hidden unit corresponds to the location of the region in the image.

Sparse interaction refers to the concept that each neuron in the convolutional layer is only connected to a small, localized region of the input grid, allowing the network to focus on local features while significantly reducing the number of parameters[1]. This reduces computational complexity and also allows the model to generalize well on unseen data. With parameter sharing, the same set of weights or filters is used across different spacial locations in the input[1]. This in addition to significantly reducing the number of parameters, also enables the network to recognize features regardless of their position in the input.

2.5 Transfer Learning

Transfer learning is a machine learning domain which aims to extract and transfer knowledge from one task or domain to improve learning in another related, but distinct target domain[2][18]. This approach is of significant advantage when there is a scarcity of labeled training data in the target domain or when the data can easily become outdated[2], which is often the case for real-world applications[18]. The idea is that the knowledge gained from a source task or domain can enhance the model's performance in the target task, even if the tasks differ in terms of data distribution[2].

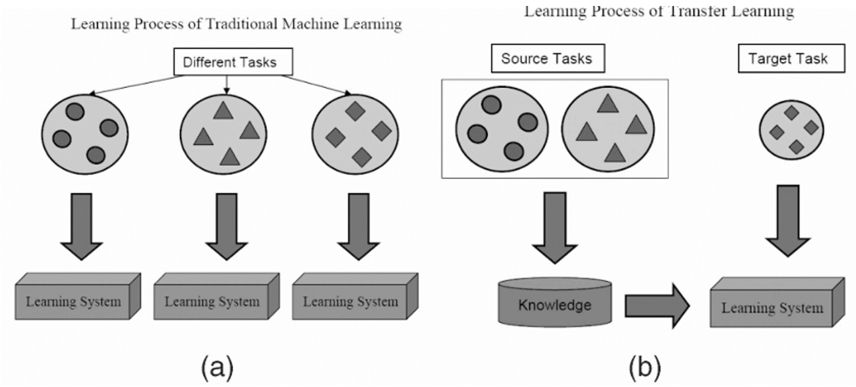

Figure 2.5: Transfer Learning[2]. Difference in the learning processes of (a) traditional machine learning and (b) transfer learning

The concept of transfer learning is based on the observation of human cognition, where humans can apply prior knowledge to solve new problems more efficiently[2]. Traditional machine learning techniques require that the data required for training and testing are from the same domain, which might be difficult in many practical situations. Transfer learning tries to focus on identifying what knowledge can be transferred, how to transfer it and when to transfer it, thus resulting in a robust method for building high performing models with limited data availability[2]. Thus transfer learning not only conserves time and resources, but also often result in performance enhancement.

Transfer learning can be based on domain adaptation or task adaptation. In domain adaptation, the objective is to transfer knowledge from a source domain to a target domain with a different data distribution. For example, using knowledge from a model trained in classifying natural images to train a model in medical images classification. On the other hand, task transfer is often referred to as training the model to perform a task using knowledge obtained by training on a different task, while still within the same data domain. For instance, sentiment analysis as a source task and text classification as the target task.

2.6 LeNet

LeNet-5 is a convolutional neural network (CNN) developed by Yann LeCun and colleagues, specifically designed for tasks like handwritten character recognition[19]. Its architecture includes a series of convolutional and subsampling layers that allow it to capture complex spatial patterns and handle variations in two-dimensional input shapes, such as handwritten digits. The initial layers capture local patterns through small, localized connections (receptive fields), while deeper layers combine these patterns into higher-level features[19]. To efficiently learn patterns at multiple locations in the image, LeNet-5 uses weight sharing within feature maps, which also reduces the number of parameters, enhancing the network's generalization capabilities.

The model's convolutional layers are followed by subsampling layers, which help make the network invariant to minor shifts and distortions by reducing the spatial resolution of feature maps. This architecture leads to a gradual reduction in spatial dimensions while increasing the number of feature maps, creating a structure that supports hierarchical feature extraction. In the final stages, fully connected layers integrate these features, and the network uses radial basis function (RBF) units to classify images based on the Euclidean distances between the learned feature representations and the input image features. By training with backpropagation, LeNet-5 learns to adjust its parameters, minimizing errors in classification and effectively synthesizing its own feature extractor from raw pixel data[19].

2.7 DenseNet

DenseNet or Dense Convolution Network, which was introduced in 2017, is an innovative architecture that connects each layer to each other layer in a feed-forward manner[3]. This architecture creates deeper and fully inter-connected networks by incorporating shorter connections between layers close to the input and those near the output. This makes the model more accurate, efficient to train and also substantially improve parameter efficiency[3]. DenseNets have been shown to mitigate the vanishing gradient problem, while also encouraging feature reuse[3].

In traditional convolutional networks, each layer has a direct connection only to the subsequent layer, i.e., a network with L layers have L connections. DenseNet has a different architecture in which a network with L layers leads to $\frac{L(L+1)}{2}$ direct connections, where each layer receives inputs from all preceding layers[3]. This structure facilitates maximal information flow, where each layer can access the features of all previous layers.

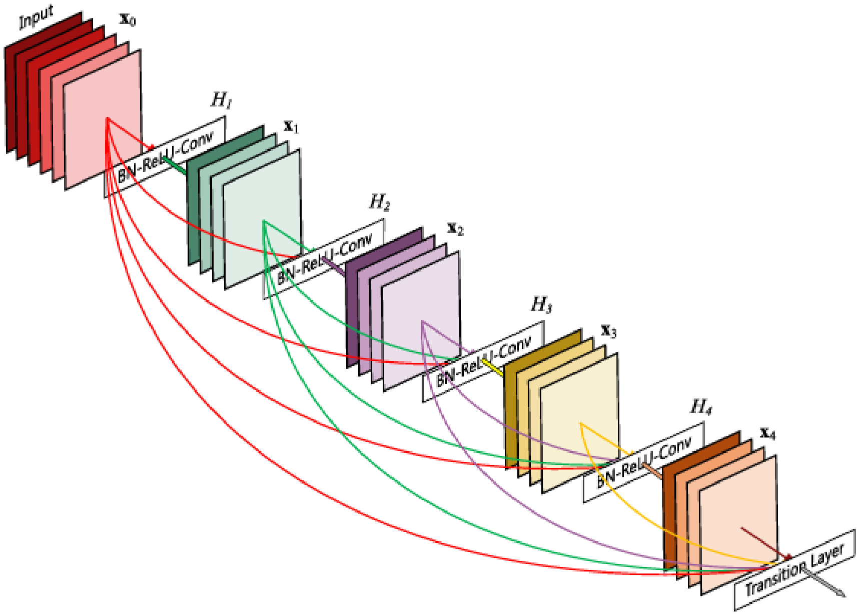

Figure 2.6: A five layer dense block [3]. Each layer takes all preceding feature maps as input.

2.8 ResNet

ResNet-50 is a convolutional neural network (CNN) architecture designed to address the limitations of very deep networks, which often suffer from high training error due to optimization difficulties when layers are simply stacked. Introduced by Kaiming He and colleagues[20], ResNet-50 is based on a residual learning framework, which simplifies the learning process in deep networks by enabling each set of layers to learn residual mappings, rather than direct mappings. This approach significantly improves the ease of optimization, allowing ResNet-50 to achieve superior accuracy even with increased depth.

The central feature of ResNet-50 is its use of residual blocks with shortcut connections, or identity mappings, that skip one or more layers[20]. This setup allows the network to pass the input forward without modification, while the skipped layers learn the residual function, essentially, the difference between the input and the desired output. Formally, if $H(x)$ is the desired mapping, the network instead learns $F(x) = H(x) - x$. This reformulation makes it easier to learn identity mappings and thus avoids the degradation problem, where deeper networks exhibit higher training errors.

ResNet-50 has 50 layers, organized with the bottleneck architecture, which reduces the dimensionality of input features before performing 3x3 convolutions, reducing computation. Each residual block in ResNet-50 consists of three layers: a 1x1 convolution layer that reduces dimensionality, a 3x3 convolution layer for feature extraction, and another 1x1 convolution layer that restores the original dimension. This architecture allows for deeper networks with fewer parameters, maintaining computational efficiency while benefiting from greater depth for feature learning. ResNet-50's innovative use of residual learning made it a state-of-the-art model, achieving competitive results across various visual recognition benchmarks

2.9 Challenges

The primary challenges that significantly impact the performance of character recognition models using ANN are:

2.9.1 Input Data Size

Character recognition models rely on vast amounts of labeled data to learn the character features of importance and to handle variations in style, size, or orientation. A limited dataset constrains the model's ability to learn the intricate characteristics of different characters, thus may cause the model to under-perform, especially when encountering variations it hasn't seen before. Thus a large and diverse dataset is critical to allow the model to capture the wide range of character representations and improve its capacity to generalize across different fonts or handwriting styles. However, for languages like Malayalam, which have limited digital resources, the process of collecting and annotating large datasets can be both time-consuming and expensive. Data augmentation techniques, that can artificially expand the dataset by applying transformations such as rotation or scaling, can be utilized to alleviate this challenge.

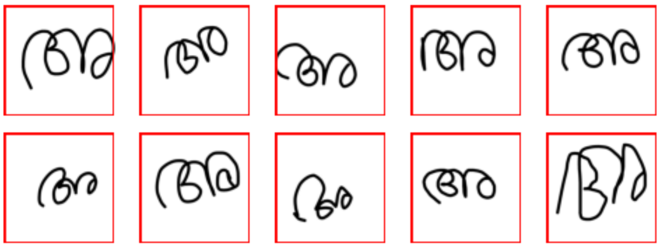

Figure 2.7: Different handwriting samples of the same character

2.9.2 Input Data Quality

The performance of any machine learning algorithm is directly linked to its input data quality. Poor-quality data, such as blurred images or skewed text can lead to inaccurate training and poor model performance. Ensuring high-quality, clean, and representative training data is essential for building robust character recognition models. Pre-processing techniques such as noise reduction and skew correction can improve data quality before it is fed into the model. Additionally, a diverse dataset with high-quality samples across different fonts, languages, and noise conditions ensures that the OCR system can handle various real-world scenarios.

2.9.3 Overfitting

Overfitting occurs when the model learns the training data too well, thus recognizing not only the underlying patterns or features, but also the noise or minor fluctuations in the data. An overfit model performs well on training data, but fails to generalize to new or unseen data. This can be particularly problematic when it comes to character recognition because the model needs to pick up underlying patterns when exposed to a diverse range of fonts, handwriting styles, and distortions. Overfitting can be a significant concern in large and complex models, such as deep networks, because their high capacity allows them to memorize training data rather than generalize from it. Overfitting can also occur when the input dataset is small. This can lead to a significant drop in performance when these models are applied to new, unseen data. To mitigate overfitting, techniques such as regularization, data augmentation, and cross-validation are often employed. Using a balanced and diverse dataset, along with early stopping during training, can also help ensure the model generalizes well to real-world use cases.

The techniques adopted in this work to tackle overfitting are L2 regularization and dropout.

Dropout

Dropout is a highly effective technique designed to mitigate overfitting in neural networks by randomly setting the output of hidden neurons to zero with a specified probability, typically around 0.5[1][21]. This approach prevents neurons from becoming overly reliant on each other, promoting the learning of robust features that are useful across various configurations of the network. During training, dropout randomly removes certain hidden units, resulting in the creation of sub-networks that share weights and parameters[21]. This process can be viewed as a bagging-like technique, where multiple models are combined without the computational overhead of training separate models[1]. Each time an input is presented, a different architecture is sampled, allowing the network to learn diverse representations of the data.

At test time, all neurons are utilized, but their outputs are scaled by the probability of retention during training (e.g., multiplied by 0.5), which approximates the behavior of averaging predictions from numerous sub-networks. This scaling ensures consistency between training and testing, allowing the model to generalize better to unseen data[1]. Its simplicity, computational efficiency, and effectiveness have made dropout one of the most popular regularization techniques in practice. A common strategy in neural network design involves extending the network until overfitting occurs, then applying dropout to maintain performance while minimizing overfitting risk[1].

L2 Regularization

L2 regularization, also known as weight decay, is a technique used to prevent overfitting in machine learning models by penalizing large weights[1]. It works by adding a term to the loss function that is proportional to the sum of the squares of the model's weights, formally represented as

$$\frac{\lambda}{2}\sum_{i=1}^{n} w_i^2,$$

where λ is the regularization strength and w_i are the weights[1][22]. This penalty discourages the model from assigning excessively large weights, which can lead to a model that fits the training data too closely, including its noise and outliers. As a result, L2 regularization encourages weight values to be distributed more evenly, promoting smoother decision boundaries[1].

The hyperparameter λ allows for flexibility in controlling the degree of regularization; a higher λ increases the penalty, further constraining the weights, while a lower λ diminishes the effect of regularization, allowing for more complex models[1]. Overall, L2 regularization is a widely adopted strategy that combines simplicity with effective performance enhancement across a variety of machine learning applications.

Chapter 3

Literature Review

OCR is a well-researched domain, yet it has received comparatively less attention in the context of regional languages. There is limited literature available on character recognition specifically for the Malayalam script.

[23] provides an exhaustive survey on HCR for Indic scripts, including Malayalam. The authors conducted an analysis of the literature on HCR for Indic scripts from 2000 to 2019, providing a comprehensive overview. It is observed that most studies on Malayalam HCR primarily utilize statistical feature extraction, while only a few rely on structural features. The popular classification methods include Support Vector Machine (SVM) and neural networks. The recognition accuracy for various methods in the literature ranges from 91.05% to 99.78%.

In this work, a comprehensive literature review of Malayalam HCR from 2020-2024 is carried out. Using the keywords 'Malayalam' and 'handwritten' in Scopus, 43 relevant articles were identified, of which 12 were selected for final analysis. The remaining articles were excluded after reviewing their abstracts and/or full texts.

The reviewed works (Table 3.1) employ varied range of methodologies, with transfer learning being a popular choice. This trend is understandable, as the methodology helps in transferring knowledge from already well established models such as ResNet50, DenseNet etc. There are both custom-built architectures and hybrid models that integrate transfer learning with tailored adjustments. Many studies report impressive test accuracies, with several methodologies achieving above 95%, which is significant for a difficult script like Malayalam. The number of classes considered in different studies ranges significantly, from as few as 20 to over 126. The size of training datasets also varies widely, influencing the results. Larger datasets often correlate with higher accuracies, demonstrating the importance of data availability in model training.

Ref	Methodology	Results	Comments
[24]	Script independent, 2-phase HCR: Transfer learning followed by custom algorithm.	98.27% Test Accuracy	85 classes, 29302 images. Fewer parameters, and better accuracy than TL methods considered.
[25]	Uses hand crafted Histogram of Oriented Gradients(HOG) Feature Descriptor(FD), and the flexible learning features of Deep Convolutional Neural Networks(DCNN).	98.07% Test Accuracy	90 classes considered.
[26]	Compares different models: FAST R-CNN, Efficient Net B7, VGG16, and VGG19	96.6% Test Accuracy (max)	20 classes, 6090 images
[27]	Custom CNN architecture for feature extraction and softmax layer for classification	99.75% Test Accuracy	44 classes, 91902 images
[28]	Proposed a residual version of Incep tion called Inception-ResNet-V2.	99.69% Test Accuracy	Top 5 error: 5%
[29]	TL using ResNet50 and fine tuning	78.05% Test Accuracy	85 classes, 17236 images
[30]	Custom multi layer CNN	96% Test Accuracy	48 classes, 4800 images. Better performance than LeNet05, DenseNet201, ResNet50 for the dataset
[31]	Custom multi layer CNN	86% Test Accuracy	48 classes, 4800 images
[32]	TL using DenseNet & VGGNEt	99.2% Test Accuracy (max)	126 classes, 250 images each
[33]	Custom CNN	91% Test Accuracy	86 classes, 17236 images
[34]	Structural analysis (loops, end points, arcs etc are analysed)		

Table 3.1: Literature Review on Handwritten Character Recognition

Chapter 4

Methodology

This chapter outlines the methodology for developing the handwritten character recognition system tailored for Malayalam characters. The data collection methods, pre-processing steps and the neural network architectures employed are thoroughly detailed.

The experiments are executed using the Python programming language. For implementing the deep learning framework, TensorFlow framework is employed. TensorFlow is an end-to-end platform with an extensive feature set for creating machine learning models in any environment. TensorFlow integrates with the high-level Keras API, facilitating easier model development and training. This framework provides powerful inbuilt computational tools and pre-trained models that provides a solid foundation for developing deep learning models, thus enhancing the efficiency and performance of the character recognition systems.

4.1 Data Collection

Malayalam is a language spoken by more than 35 million people, and is the official language of the Indian state Kerala, and union territories Lakshadweep and Puducherry. The character set consists of 36 consonants, 15 vowels, and an array of additional characters and symbols.

There are few publicly available datasets of Malayalam handwritten characters, and even the available ones have limited number of images per class. While these datasets can provide a base for initial model development, the small number of images each classes can be a constraint on the performance of machine learning models, restricting their ability to generalize well to real-world scenarios. To create a suitable dataset for this work, publicly available datasets from Kaggle were combined with samples collected locally from a diverse group of individuals. This provides a diverse range of handwriting styles to be used for training. All the images are generated through direct digital input, i.e., the characters are written using digital tools such as styluses or touchscreen devices, avoiding the need for scanning handwritten characters on paper. This avoids any noise that might occur from scanning physical documents. In this work, a final set of 43 characters and a total of 7132 images were used, with some characters excluded due to limited data availability.

4.2 Data Pre-Processing

Data pre-processing is a crucial step that lays foundation for model training and performance, especially when dealing with data from multiple sources. The performance of any machine learning model is directly influenced by the input data quality. Pre-processing involves transforming input data into a suitable format for feeding into the machine learning models. Pre-processing thus enhances the quality of input data, ensuring that the model can learn effectively from it.

4.2.1 Input Data Format

To determine the required pre-processing steps, it is important to understand the format and characteristics of input image.

The images are in PNG format, 300×300 pixels in size with 4 channels, RGBA. Here, RGBA represents Red, Green, Blue and Alpha (transparency). Each channel can have values ranging from 0 to 255, where Red, Green, and Blue represents the color intensity of each pixel, and Alpha represents the pixel's transparency level, with 0 being fully transparent and 255 being fully opaque. The input images have dark characters on a transparent background.

4.2.2 Initial Clean Up

To ensure the accuracy of the input data, all images from the various character sets were manually reviewed to confirm that no mis-assignments occurred. Illegible or incorrect characters were also removed.

The input dataset is formed by consolidating multiple publicly available datasets and manually collected samples. All these images need to be converted into a single format to feed into the algorithm. To simplify image processing all the images are converted into BGR format by reducing the additional transparency channel. The images now have dark characters on a white background.

4.2.3 Noise Removal

Noise in an image refers to unwanted variations in pixel values that might affect key features, making it difficult to distinguish or analyze the relevant details. Noise that may arise from scanning is avoided as the characters are directly written using digital tools. However, unintentional hand movements, ink smudges, and similar factors can still introduce unwanted marks and distortions, resulting in noise. Noise removal techniques aim to focus on the main character, removing unwanted marks or background noise, thus enabling the algorithms to focus on important features by making subsequent image processing tasks efficient.

In this work, a sequence of steps are adopted for noise removal:

1. **Convert to Grayscale**: The input images are converted from RGBA format to grayscale. This step simplifies the image input by reducing it to a single channel, making it easier for further processing. Grayscale images contain only a single channel with intensity information, which is sufficient for further operations.

2. **Binarization**: Binarization is a noise filtering technique where all the pixels are classified as either black or white. This is used for distinguishing the foreground (characters) from the background. Thresholding is an effective technique for binarization, converting images into a binary format by separating pixel values into two distinct categories. The method compares each pixel's intensity value to a thresthold value and decides whether it should be black or white based on this comparison. In this work, a brute-force approach of converting all the non-black pixels into white is adopted. For the input data format available, this retains all the relevant information without compromising on the character structure.

3. **Contour Detection and Separation**: Contours are detected in the binary image using the `findContours` function from the OpenCV library. A contour joins all continuous pixels along a boundary with the same intensity. This function detects contours from the image and isolates distinct shapes within the image. The focus is on external contours here. The contour with the largest area is identified from the detected contours. This contour from the

image is assumed to represent the character of interest. This contour is only retained for further processing. The idea is that by focusing only on the most relevant shape, smaller, irrelevant details or noise are eliminated.

For the identified largest contour, a bounding box encapsulating it is calculated. The contour's top-left corner coordinates and its dimensions can be used to define a rectangular bounding box around it. This bounding box thus captures the main feature from the image that is to be retained. This feature, contained within the bounding box is placed on a plain background, eliminating any distortions or noise.

4. **Final Clean Up**: The filtered binary image is converted back to a color format, BGR format. This is done because for many model architectures used in this work, a BGR format is required rather than a grayscale format. The final image has white characters on a black background, which is to be used for further processing.

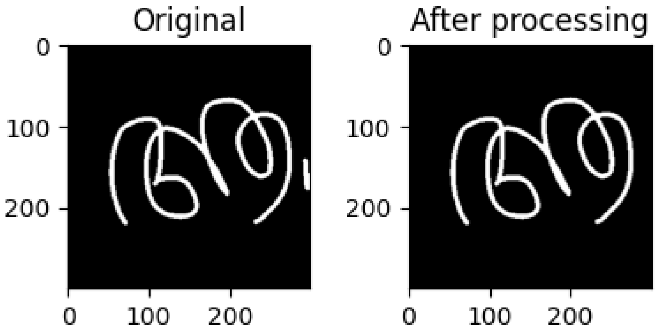

Figure 4.1: Noise Removal

4.2.4 Skew Correction

In the input dataset available, it can be seen that there are variations in the text lines from the horizontal baseline, resulting in 'skewed' characters. This angular tilt or skew can adversely impact the performance of the deep learning algorithms by introducing misalignments that can hinder the ability of the model to recognize key features. Hence, a skew correction technique is essential to ensure that the input images conforms to the expected alignment or format.

Malayalam script has complex, curved character shapes, making it hard to detect and correct skew accurately. Additionally, skew correction for a single character, rather than an entire text line, can be challenging with the traditional algorithms.

The steps employed for skew correction are:

1. **Convert to Grayscale**: The image is converted to grayscale for simplifying processing.

2. **Padding**: Padding involves adding extra pixels (of background color) around the edges of the image to prevent skew correction from cutting off important parts of image, thus resulting in data loss. The padding size is set to half the width of the image.

3. **Contour Detection**: Contours of the character are detected to isolate the primary text area. From the largest contour detected, a bounding box is calculated, encapsulating the entire character. This is the 'region of interest' and is necessary to determine the orientation of the character.

4. **Sectioning and Bottom Point Identification**: The region of interest is divided into two sections by a vertical line drawn at the midpoint of its width. The lowest points within each section are identified. This information is used to determine the correct baseline for the character. This method of dividing the region of interest into sections for skew detection is adopted from [35].

5. **Line Fitting and Angle Calculation**: A line is fitted through the identified bottom points, and its orientation relative to the horizontal axis determines the skew angle.

6. **Image Rotation**: Based on the calculated skew angle, the image is rotated to align the character horizontally.

7. **Cropping and Resizing**: Now to convert the image back to its original size, the region on interest (bounding box encapsulating the character) is detected. In most cases, the region of interest will be smaller than the original image size. Then the appropriate area to crop around the region of interest is calculated and the size is adjusted to match the original image dimensions. Sometimes, when a skewed character is corrected to align with the horizontal line, the resulting region of interest may be larger than the original image dimensions. In that case, the image is first cropped to the bounding box dimensions and then resized (using OpenCV) to fit the original dimensions. Thus the final image is made sure to be correctly centered and resized, maintaining the key features, while fitting to the size requirements.

8. **Final Clean Up**: Finally, the image is converted back to its original color format with white characters on black background.

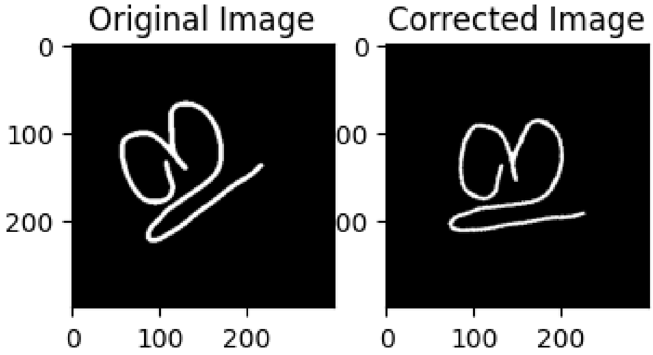

Figure 4.2: Skew Correction

Most of the Malayalam characters can be contained within two parallel lines, which represent the typical bounds of their vertical extent. The above described technique work well for most characters. However, there are characters that fall beyond these bounds created by two lines. The proposed method falls short for those characters. Thus while it is observed to work well for most of the characters, the method produces inaccurate corrections for characters such as ('aah', 'aeh') etc. However, this method gives better performance than other methods such as Hough transform, projection profile analysis or Fourier transform. Finding the bounding box with minimum area and trying to align it with the horizontal baseline also works for many characters, but with comparitively lower performance. Hence this method is adopted in this work, despite its limitations.

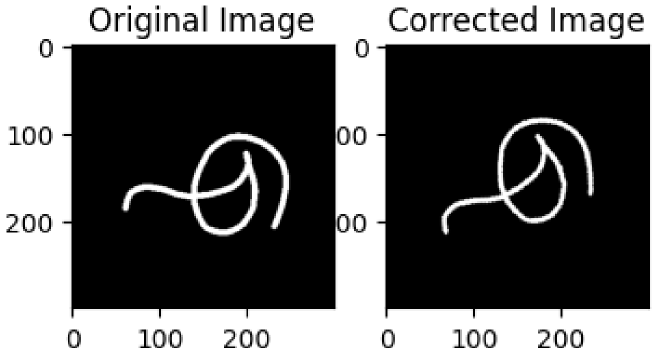

Figure 4.3: Skew Correction Failure Example

4.2.5 Standardization

The orientation of the character in the input images is diverse and it is essential standardize them for accurate recognition. This involves clearly separating the character from the background, and ensuring the characters are uniformly sized.

1. **Character Isolation**: The image is converted into grayscale and the bounding box of largest contour is identified. This essentially separates the character from the rest of the image.
2. **Crop and Padding**: The image is cropped to the detected bounding box. To prevent the character from being cut off, a 10-pixel padding is added around the cropped area.
3. **Final Adjustment**: The image is converted back to color format and resized using OpenCV to the original size of 300 × 300 pixels.

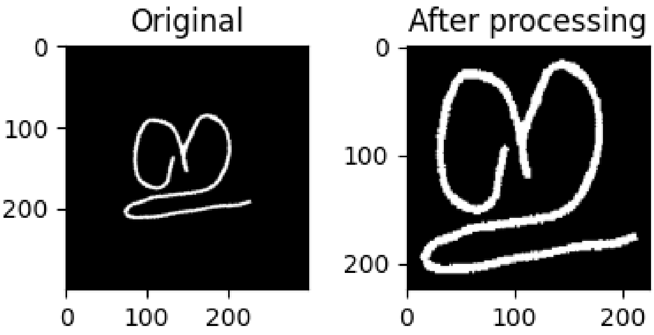

Figure 4.4: Standardization

Chapter 5

Results and Discussion

All four discussed NN architectures are implemented utilizing the TensorFlow and Keras frameworks. Experiements were conducted on a computer equipped with 11th Gen Intel Core i5-1155G7 processor, 16 GB RAM with 4 physical cores.

The dataset is divided into three subsets: 70% for training, 20% for testing, and 10% for validation. The primary performance metrics used to evaluate the models are training and testing accuracies. Accuracy is a crucial metric, as it reflects the model's ability to correctly identify each character without ambiguity. A high training accuracy indicates that the model effectively learns the features for accurate classification. A corresponding high testing accuracy reflects the ability of the models to generalize well.

To provide a more comprehensive analysis of the algorithms, the Top-2 and Top-3 accuracies are also reported. Top-n accuracy considers the model's top n predictions; if the ground truth label appears among these predictions, it is counted as a correct classification. This metric is especially important in scenarios where characters may be similar or when the model's confidence in its top prediction is lower. In the current scenario, certain combinations of Malayalam characters can be challenging to distinguish. By evaluating Top-2 and Top-3 accuracies, insights into the models' overall ability to learn and generalize are attained, highlighting their robustness in handling ambiguities in character recognition.

The results for handwritten character recognition of all the different architectures are analyzed are elaborated. Owing to the stochastic nature of NNs, 10 separate runs are implemented for all the models. The average accuracy obtained from these different runs are provided in Table 5.1.

The performance of the models are shown to be dependent on the quality of the input. There is a notable increase in the accuracy of the models following

Architecture	Before Preprocessing	After Preprocessing		Epochs
	Test Accuracy	Training Accuracy	Test Accuracy	
LeNet05	86.23%	90.48%	92.43%	25
DenseNet201	85.24%	92.02%	91.73%	5
ResNet50	91.33%	97.90%	95.52%	5

Table 5.1: Accuracy of model architectures

the application carefully curated pre-processing steps. ResNet50 has demonstrated superior performance compared to all the models evaluated in this study. It excels in both training and testing accuracies, indicating that the model effectively learns and generalizes well. Additionally, ResNet50 shows significant improvement in Top-2 and Top-3 accuracies compared to its testing accuracy (Table 5.2). This suggests that while the model performs well overall, it may struggle to distinguish between certain similar-looking characters.

Architecture/Accuracy	Top-2	Top-3	Top-5
LeNet05	96.95%	98.04%	99.23%
DenseNet201	97.55%	98.32%	99.37%
ResNet50	98.46%	98.86%	99.43%

Table 5.2: Top-n accuracy of model architectures

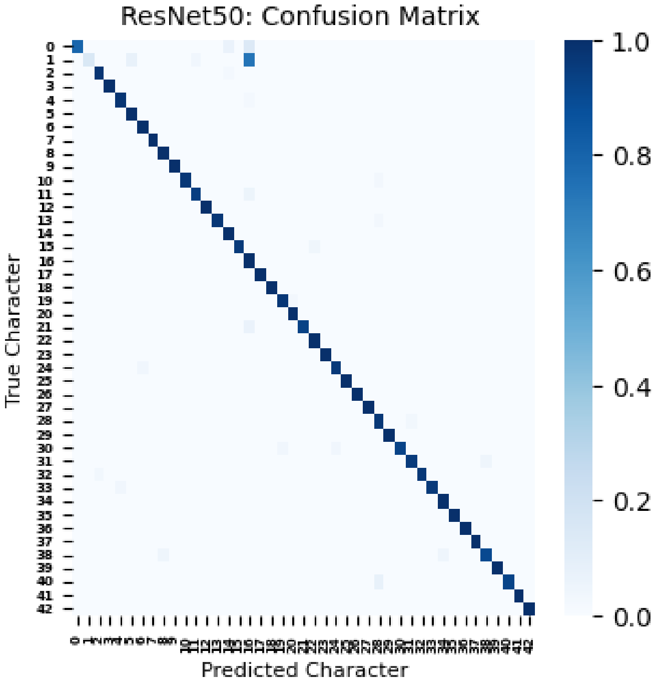

Figure 5.1: ResNet50 Confusion Matrix

Upon deeper analysis, it is observed that on average, ResNet50 correctly predicted all test cases for 26 characters out of the total 43 tested (Fig. 5.1). This indicates that the algorithm effectively recognizes the features of these 26 characters, clearly distinguishing them from others. For several characters, it occasionally misidentifies them as one or two others; nevertheless, it achieves a high level of accuracy overall. This is substantial for a script such as Malayalam, where the letters can be quite similar to one another. For DenseNet201 and LeNet05, the models correctly distinguishes 22 and 7 characters at all instances, respectively.

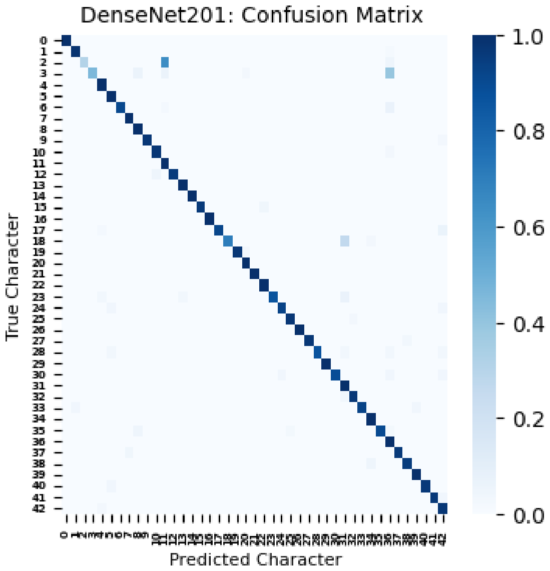

Figure 5.2: DenseNet201 CM

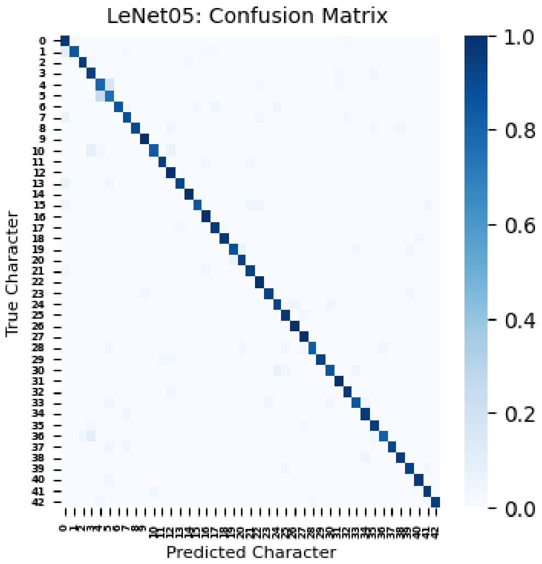

Figure 5.3: LeNet05 CM

The number of epochs required for DenseNet201 and ResNet50 to converge is considerably lower than that of the other models. This efficiency may be attributed to their complex and dense architecture, which facilitates rapid learning by allowing the models to effectively capture and utilize relevant features from the data.

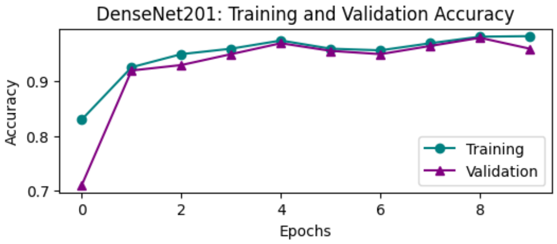

Figure 5.4: DenseNet201 Performance

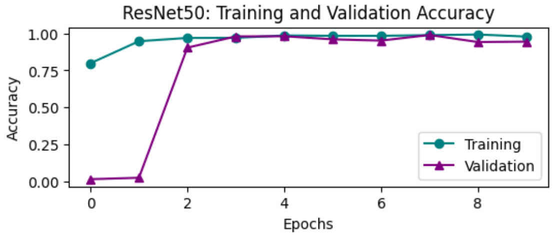

Figure 5.5: ResNet50 Performance

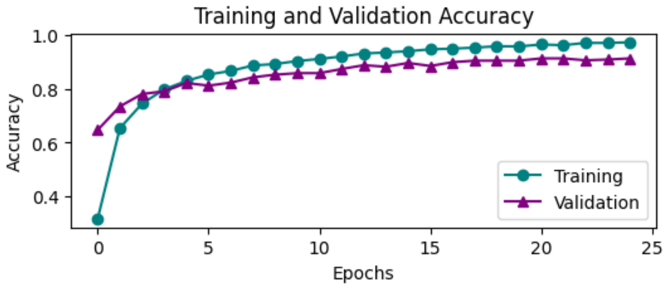

Figure 5.6: LeNet05 Performance

In this work with the Malayalam handwritten character recognition systems, it is observed that the architecture of ResNet50 featuring residual connections outperforms DenseNet201. This might be specific to the dataset used. With fewer parameters, ResNet50 is proved to be more efficient and less prone to overfitting, especially on a small dataset as the one used in this work. ResNet50's training dynamics have been observed to be more stable in this particular case, contributing to better performance. This stability may have enabled it to capture the essential features of handwritten characters more effectively, without the added complexity of DenseNet201's dense connectivity.

Chapter 6

Conclusion and Future Scope

Transfer learning is a significant research topic today because it enables efficient model training, enhances performance on small datasets, and accelerates innovation across various domains by leveraging pre-existing knowledge. In this work on Malayalam handwritten character recognition using transfer learning, the effectiveness of leveraging pre-trained NN models to recognize Malayalam characters is investigated. A comprehensive literature review is presented, focusing on research in the domain of Malayalam Handwritten Character Recognition. The study focuses on the potential of utilizing transfer learning to pass on knowledge from well established deep learning architectures to recognize handwritten Malayalam characters. The presence of complex shapes and a large number of similar or confusing characters makes recognition of Malayalam script especially challenging. The input data quality is critical, and hence a sequence of pre-processing steps are carried out to ensure optimal performance. By employing transfer learning, the algorithm is able to perform well on a considerably small dataset with a large number of classes. The models tested in this work, fine-tuned on the Malayalam dataset, is shown to achieve high performance, significantly improving the potential for practical applications in automated handwritten text recognition systems.

6.1 Future Scope

The future scope for utilizing transfer learning for handwritten character recognition presents several exciting opportunities:

1. Hybrid Models: Experimenting with more complex and hybrid architectures could further enhance the performance and flexibility of the model. The model will be able to leverage the strengths of multiple architectures, thus overcoming the

limitations of any single architecture.

2. Dataset Expansion: While data from multiple sources is integrated to create the final dataset used in this work, it is still limited in size and variety. Expanding the dataset with more varied samples could be beneficial, and further exploration of different data augmentation methods could also enhance model performance.

3. Advancing to Word Recognition: Advancing this study to recognize Malayalam words instead of characters is an interesting future scope. This enhancement could pave the way for more advanced applications.

Bibliography

[1] Andreas Lindholm, Niklas Wahlström, Fredrik Lindsten, and Thomas B Schön. *Machine learning: a first course for engineers and scientists.* Cambridge University Press, 2022.

[2] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[3] Gao Huang, Zhuang Liu, Geoff Pleiss, Laurens Van Der Maaten, and Kilian Q Weinberger. Convolutional networks with dense connectivity. *IEEE transactions on pattern analysis and machine intelligence*, 44(12):8704–8716, 2019.

[4] Line Eikvil. Optical character recognition. *citeseer. ist. psu. edu/142042. html*, 26, 1993.

[5] Pratixa Badelia Soumya K Ghosh Arindham Chaudhuri, Krupa Mandaviya. *Optical character recognition systems.* Springer, 2017.

[6] Karez Hamad and Mehmet Kaya. A detailed analysis of optical character recognition technology. *International Journal of Applied Mathematics Electronics and Computers*, (Special Issue-1):244–249, 2016.

[7] Xiao-Feng Wang, Zhi-Huang He, Kai Wang, Yi-Fan Wang, Le Zou, and Zhi-Ze Wu. A survey of text detection and recognition algorithms based on deep learning technology. *Neurocomputing*, 556:126702, 2023.

[8] King-Sun Fu et al. Pattern recognition and image processing. *IEEE transactions on computers*, 100(12):1336–1346, 1976.

[9] Gaurav Y Tawde and Jayshree Kundargi. An overview of feature extraction techniques in ocr for indian scripts focused on offline handwriting. *International Journal of Engineering Research and Applications*, 3(1):919–926, 2013.

[10] Rohit Verma and Jahid Ali. A-survey of feature extraction and classification techniques in ocr systems. *International Journal of Computer Applications & Information Technology*, 1(3):1–3, 2012.

[11] Sukhpreet Singh. Optical character recognition techniques: a survey. *Journal of emerging Trends in Computing and information Sciences*, 4(6), 2013.

[12] AD Dongare, RR Kharde, Amit D Kachare, et al. Introduction to artificial neural network. *International Journal of Engineering and Innovative Technology (IJEIT)*, 2(1):189–194, 2012.

[13] Guoqiang Zhang, B Eddy Patuwo, and Michael Y Hu. Forecasting with artificial neural networks:: The state of the art. *International journal of forecasting*, 14(1):35–62, 1998.

[14] Oludare Isaac Abiodun, Aman Jantan, Abiodun Esther Omolara, Kemi Victoria Dada, Nachaat AbdElatif Mohamed, and Humaira Arshad. State-of-the-art in artificial neural network applications: A survey. *Heliyon*, 4(11), 2018.

[15] Xiao-Feng Wang, Zhi-Huang He, Kai Wang, Yi-Fan Wang, Le Zou, and Zhi-Ze Wu. A survey of text detection and recognition algorithms based on deep learning technology. *Neurocomputing*, 556:126702, 2023.

[16] Christopher M Bishop and Hugh Bishop. *Deep learning: Foundations and concepts*. Springer Nature, 2023.

[17] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. A survey on deep transfer learning. In *Artificial Neural Networks and Machine Learning–ICANN 2018: 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part III 27*, pages 270–279. Springer, 2018.

[18] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big data*, 3:1–40, 2016.

[19] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[22] Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. L2 regularization for learning kernels. *arXiv preprint arXiv:1205.2653*, 2012.

[23] Reya Sharma and Baijnath Kaushik. Offline recognition of handwritten indic scripts: A state-of-the-art survey and future perspectives. *Computer Science Review*, 38:100302, 2020.

[24] Vinod Kumar Chauhan, Sukhdeep Singh, and Anuj Sharma. Hcr-net: A deep learning based script independent handwritten character recognition network. *Multimedia Tools and Applications*, pages 1–35, 2024.

[25] Bineesh Jose and Pushpalatha KP. Classification of handwritten malayalam characters using a hog-dcnn model with multiview augmentation and inference fusion. *Multimedia Tools and Applications*, 83(7):19969–19980, 2024.

[26] CS Arjun, N Shobha Rani, and Akshatha Prabhu. Handwritten character recognition for south indian languages using deep learning. In *International Conference on Computer & Communication Technologies*, pages 49–63. Springer, 2023.

[27] Pranav P Nair, Ajay James, Philomina Simon, and PV Bhagyasree. Malayalam handwritten character recognition using cnn architecture. *Indonesian Journal of Electrical Engineering and Informatics (IJEEI)*, 11(3):764–777, 2023.

[28] Bineesh Jose and KP Pushpalatha. Malayalam handwritten character recognition using transfer learning. In *2023 International Conference on Advances in Intelligent Computing and Applications (AICAPS)*, pages 1–5. IEEE, 2023.

[29] PV Pearlsy and Deepa Sankar. Malayalam handwritten character recognition using transfer learning and fine tuning of deep convolutional neural networks. In *2023 3rd International Conference on Advances in Computing, Communication, Embedded and Secure Systems (ACCESS)*, pages 125–129. IEEE, 2023.

[30] Remya Sivan, Suja Palaniswamy, and Peeta Basa Pati. Comparative study of deep learning models to recognize palm leaf malayalam characters. In *2022 6th International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS)*, pages 1–6. IEEE, 2022.

[31] Remya Sivan, Tripty Singh, and Peeta Basa Pati. Malayalam character recognition from palm leaves using deep-learning. In *2022 OITS International Conference on Information Technology (OCIT)*, pages 134–139. IEEE, 2022.

[32] BJ Bipin Nair and Adarsh Mohan. A comparative vggnet and densenet approaches to recognize malayalam characters using transfer learning techniques. In *2022 Sixth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)*, pages 1060–1066. IEEE, 2022.

[33] VK Vaisakh and Lyla B Das. Handwritten malayalam character recognition system using artificial neural networks. In *2020 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS)*, pages 1–4. IEEE, 2020.

[34] Dany Varghese, Alireza Tamaddoni-Nezhad, S Moschoyiannis, P Fodor, J Vanthienen, D Inclezan, and Ni Nikolov. One-shot rule learning for challenging character recognition. In *RuleML+ RR (Supplement)*, pages 10–27, 2020.

[35] MLM Karunanayaka, Chandrajith Ashuboda Marasinghe, and Nihal D Kodikara. Thresholding, noise reduction and skew correction of sinhala handwritten words. In *MVA*, pages 355–358, 2005.