Offline Handwritten Bangla Character Recognition using Few-Shot Learning

by

Priya Saha
22241185

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
October 2024

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Priya Saha
22241185

Approval

The thesis titled "Offline Handwritten Bangla Character Recognition using Few-Shot Learning" submitted by

1. Priya Saha (22241185)

Of Summer, 2024 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on November, 2024.

Examining Committee:

Supervisor:
(Member)

Annajial Alim Rasel
Senior Lecturer
CSE
Brac University

Co Supervisor:
(Member)

Dewan Ziaul Karim
Lecturer
CSE
Brac University

Co Supervisor:
(Member)

Md. Sabbir Hossain
Lecturer
CSE
Brac University

Program Coordinator:
(Member)

Dr. Md. Golam Rabiul Alam
Associate Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Abstract

Character recognition is becoming more and more important as a result of its numerous applications. Optical character recognition has several uses for accessibility, storability, backups, and translation among other things, in the legal, healthcare, and financial sectors in the real world. This study focuses on a discussion of different character recognition methods for Bangla and other languages. A variety of handwritten character recognition methods have been developed for Bangla language. Yet, Bangla handwritten characters are difficult to recognize because of their variety, similarity, and compound characters. In a few papers, various deep learning techniques were applied to the recognition of Bangla characters.
This paper's goal is to suggest a novel strategy for recognizing Bangla characters. Considering the novelty of this field, the application of a few-shot learning approach, particularly because there has been no work on Bangla character recognition using this method. This approach is also well-suited for low-resource datasets. While other languages such as Chinese, Tamil, Urdu, and Malayalam have seen some work using few-shot learning, this remains an unexplored area for Bangla character recognition.

Keywords: Bangla handwritten character recognition, feature extraction, Few-shot learning, prototypical network, deep learning

Acknowledgement

Thanks to Annajiat Alim Rasel Sir, Dewan Ziaul Karim Sir, and Research Assistant Humanyun Kabir Mehedi.

Table of Contents

List of Figures

List of Tables

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

DL Deep Learning

ED Euclidean Distance

FSL Few Shot Learning

ML Machine Learning

OCR Optical Character Recognition

PN Prototypical Network

Chapter 1

Introduction

1.1 Introduction

There are many people around the world who use Bangla as a language. Despite Bangla's widespread usage and popularity, there aren't as many works in it as there are in other widely used languages, including English. Because the style of the Bangla language is entirely distinct from that of English or other languages. Given that Bangla characters combine numerals, simple characters, and compound characters, Bangla's handwritten characters are distinctive and challenging.They have a different range of dimensions, forms, and sizes. Contrarily, the most well-liked and active area of pattern recognition research is optical character recognition. A technique for converting printed or handwritten text or images into digital files is optical character recognition.

This thesis research focuses on implementing and evaluating a few-shot learning approach for Bangla handwritten character recognition using a Prototypical Network integrated with various pre-trained models such as VGG16, MobileNet, ResNet50, InceptionV3, and DenseNet121 . To assess their performance in recognizing Bangla characters with minimal training data. This work not only aims to enhance Bangla character recognition, but also opens the way for applying these techniques to tribal and low-resource languages. By developing effective recognition models for these languages, we can contribute to preserving linguistic diversity and ensuring that digital technologies are accessible to a broader range of tribal and unprivileged communities.

1.2 Research Problem

Optical character recognition (OCR) is a technique that converts the text into a format that computers can understand. In the modern world, OCR is useful for digitizing typewritten materials as well as handwritten ones. It makes it much easier to find important information in a pile of documents and files, saving a lot of manual searching.

In optical character recognition (OCR), the features of characters and categorizing them based on patterns are two important parts[12]. There's a specific area within OCR called Handwritten Optical Character Recognition, which has gained a lot of interest. This area is divided into two types: offline and online systems, depending on how the input is given. Offline systems work with static input, like scanned pictures, while online systems are more dynamic and rely on the movement of a pen, considering factors like speed, angle, position, and path. Online systems are seen as more advanced because they address the issue of input overlap, which offline systems struggle with.

With the development of technology, one of the first OCR systems was created in the 1940s[12]. With advancements, OCR technology evolved to recognize both printed and handwritten characters. Eventually leading to OCR machines being sold commercially. A significant milestone was the introduction of the "IBM 1287" reading a device at the 1965 New York World Fair. It was the first optical reader capable of recognizing handwritten numbers.

In the past ten years [12], researchers have attempted to enhance the accuracy of OCR systems by fusing with image processing techniques. Researchers worked on various techniques like support vector machines, random forests, k-nearest neighbors, decision trees, and neural networks. In recent years, scientists have mostly concentrated on creating deep learning-based algorithms for digitizing handwritten materials.

Few-shot learning is an important and challenging area in deep learning, which focuses on learning from a limited number of labeled examples. This approach is gaining attention because deep learning typically requires large-labeled datasets, which are not always available for various reasons. Learning from a few examples is a human-like learning process. Few-shot learning has shown significant potential in various machine learning applications, especially in image classification. Since it is a relatively new technique, many researchers are exploring its foundational concepts using image datasets like Mini-ImageNet and Omniglot. Additionally, few-shot learning offers opportunities, to work with [18] low-resource languages. Prototypical networks, a popular approach to few-shot learning, are often used as a baseline in these studies. Prototypical networks are an effective method in few-shot learning,

representing each class with a prototype, which is the mean of the feature embeddings from the support set. During inference, a query sample is classified by measuring its distance to these class prototypes. This approach is computationally efficient and avoids complex meta-learning strategies, making it widely used in areas like computer vision and natural language processing.

1.3 Research Objective

After studying Bangla character recognition research papers and trying to figure out a novel approach in this field, we chose to work with a few-shot learning approach. Few-shot learning is a machine learning technique that can recognize new objects using a small amount of labeled data. The following is a list of the objectives of this research:

1. Traditional deep learning requires huge datasets, which is an obstacle for my research.
2. Making a research path for low-resource language datasets which have complex characteristics like the Bangla language.
3. Experiment with one of the few-shot approaches; here, it is the prototypical network.
4. To experiment how the prototypical network fits with image classification models like MobileNet, VGG16, ResNet50, DenseNet121, InceptionV3.
5. To find out the strengths and limitations of the few-shot approach with prototypical network.

Chapter 2

Literature Review

There have been many types of research conducted in this character recognition field using the machine learning approaches.

Pramanik et al. suggested a cutting-edge shape decomposition-based segmentation method for breaking down compound features into distinct shape components in their paper[4]. This form of decomposition enhances recognition accuracy while reducing categorization difficulty by requiring fewer classes to recognize. The segmentation area, where two fundamental shapes are combined to create a compound character, is where the deconstruction takes place. The classifier employed by the authors to classify data was based on a multi-layer perceptron model and used backpropagation learning. They conducted their experiment using the CMATERdb dataset and the ICDAR 2013 Segmentation dataset. These datasets are objective and contain a variety of components in this particular field. The investigation used 10,240 photos of compound characters. For training reasons, they have employed 12,300 basic character graphics. They not only have a high recognition rate, but they also employ fewer classes to produce such precise results. Classes for compound characters no longer require separate definitions. Their research will pave the way for a novel approach to utilising the compound characters' shapes for Bangla character recognition.

The authors of this paper [1] suggested an approach that normalises the written character pictures before using CNN to classify specific characters. It does not use any feature extraction techniques. In this work, 20000 handwritten characters in various shapes and variations are employed. The author's approach performed better than several other important new approaches. For 50 isolated Bangla characters, they prepared a sizable handwritten dataset. For handwritten character scripts, they took into account about 30 people of various ages and educational levels. Their prepared dataset size is 20000 with 400 samples per character. Some of these character illustrations have extremely intricate shapes and are closely related to one another. In their paper on handwritten characters, the authors explored the efficiency of deep convolutional neural networks (DCNNs) [3]. performance on the CMATERdb dataset.The authors of this paper [17] use the SE-ResNeXt to present a deep Convolutional Neural Network model. The proposed model could identify Bangla handwritten compound characters with an average accuracy of 99.82%. Also, the suggested model exhibits higher outcomes than the most advanced models, out-

performing them.

To recognise Bangla handwritten letters, Das et al. suggested an extended convolutional neural network (CNN) model [14]. Their model is evaluated using the "BanglalLekha-Isolated" dataset, which has 39 classes of consonants and 11 classes of vowels. That model indicates that for Bangla numbers, 99.50% of the identification accuracy is accurate; vowels got 93.18%, consonants got 90.00%, and combined classes got 92.25%. Three convolutional layers, one picture input layer, and three fully connected layers make up the architecture of their suggested model. In another paper [16], the authors used face mapping to upgrade the recognition of Bangla handwritten characters. The results of their handwriting recognition were surprisingly good using a basic machine learning technique. They achieved a 94% accuracy for individual characters and an average of 91% for all characters. They focused on eleven vowel characters for better identification and used Scikit-Learn along with an additional SVM classifier on a matrix dataset for their solution.

For the recognition of Bangla handwritten characters, Rizvi [9] et al. concentrated on two distinct architecture. First, a hybrid SVM classifier model trained these features. Second, they take into account ResNet, a multilayer CNN that employs residual learning and automatically picks up the information required for recognition.

The authors used two datasets, BanglaLekha-Isolated and Ekush. These datasets contain collections of distinct, handwritten Bangla characters. They utilized 700 samples from BanglaLekha-Isolated and 3000 samples from Ekush for each unique character to create their dataset. The dataset was divided into three groups: basic characters (including vowels and consonants), numbers, and compound characters. Their findings showed that ResNet-18, a CNN-based model, outperformed conventional classifier-based methods.

The authors [15] addressed numerous methods for comparing and identifying handwriting characters in image documents in this paper. In addition, discussions have been done about the statistical approach and the SVM classifier network method in comparison to the structural pattern recognition, and graphical methods. When combined with a machine learning strategy, statistical SVM produces good results for OCR system performance. A training section with a variety of fashionable letters and digits was used to test the suggested model's ability to learn with more accuracy.

Li et al. [15] proposed a straightforward and powerful method for recognizing irregular text in scenes using readily available neural network components and only word-level annotations. Their approach includes a 2D attention module, a 31-layer ResNet, and an encoder-decoder framework based on LSTM. It achieves state-of-the-art performance on irregular text recognition benchmarks and performs equally well on datasets with regular text. For training, they used large synthetic datasets like Syn90k and SynthText. The system is composed of various pre-built neural network modules, such as ResNet CNN, an LSTM encoder-decoder, and a customized 2D attention module. Without additional guidance, the model can select local features for character decoding. Their method works effectively with both regular and

irregular text and accommodates various text layouts.

Baldominos et al. report the contributions for handwritten digit recognition on the MNIST dataset [7] in their paper. This dataset has been used in computer vision to verify new methods. Researchers have studied convolutional neural networks (CNNs) and other deep learning approaches with the help of this dataset. This essay distinguishes between works that employ various forms of data augmentation. For this dataset, a sizable number of works have achieved a test error rate of less than 1%, which is no longer hard. Towards the middle of 2017, a new dataset called EMNIST was released. It has both digits and letters and contains a larger amount of data that was obtained from a different source than MNIST. Moreover, EMNIST is described, and certain findings are examined in this paper.

Baldominos et al. employed MobileNet to recognise handwritten characters in this paper [7]. A CNN architecture called MobileNet is optimised for cell phone because it uses less processing power. The accuracy rates were 96.46% for 231 categories, 96.17% for 171 compound characters, 98.37% for 50 basic characters, and 99.56% for 10 numeral characters.They collected photos for 231 classes using the CMATERdb 3.1.1, 3.1.2, and 3.1.3 databases.

In this paper [11], the authors discuss using a combination of a convolutional neural network and an Error Correcting Output Code classifier for OCR. They used CNN for extracting features and ECOC for classification. They trained and tested this CNN-ECOC model on the NIST handwritten character image dataset, and the results showed that it achieved higher accuracy compared to the regular CNN classifier.

Jaume et al. have created a new dataset called FUNSD[8], designed for understanding and extracting text from scanned forms. This dataset contains 199 real scanned forms with detailed annotations. It can be used for various tasks like labeling entities, optical character recognition (OCR), and identifying text in different applications.

In this study, Zharikov et al. showcase the Distorted Document Images dataset [13] (DDI-100) and show how valuable it is for a variety of document analysis issues. The DDI-100 dataset is a made-up dataset with more than 100,000 augmented images that are based on 7000 document pages. In order to validate the DDI-100 dataset, different TD and OCR models that exhibit high-quality performance on real data were used.

According to Rabby et al., main characters, digits, modifiers, and compound characters printed in Bangla handwritten characters can be recognised [5]. Because of this problem, each person writes handwritten characters in their own unique way and opts for handwritten scripts. Handwritten Bangla character can be recognised using the model Ekushnet, proposed by the authors. The Ekush dataset was used to train the suggested model, validate it, and cross-validate it using the CMATERdb dataset. Their proposed approach achieved a recognition accuracy of 97.73% on the Ekush dataset and a cross-validation accuracy of 95.01% on the CMATERdb dataset.

In this paper [6], Rabby et al. suggested a straightforward, lightweight CNN model. Using 50 fundamental Bangla characters, their model can categorise Bangla handwriting characters . The Banglalekha-Isolated, CMATERdb, and ISI datasets were used in their experiment. Their character recognition model, BomoNet, achieved validation accuracy scores of 98% on the CMATERdb dataset, 96.81% on the ISI dataset, 95.71% on the Banglalekha-Isolated dataset, and 96.40% on the mixed dataset.

In this paper, the authors conduct a comparative investigation of various character recognition methods in various languages [10]. Several languages, including English, Bangla, Marathi, Devanagari, Oriya, Chinese, Latin, and Arabic, are the focus of this analysis. Based on these variables, such as the dataset and approach employed, this paper provides a comparative comparison of machine learning, and deep learning strategies for handwritten character recognition. Due to the variety of characters in various languages, handwritten character recognition is a difficult task. Segmentation, feature extraction, and classification algorithms are the main topics of this research.

In this paper [2] , Snell et al. experimented on the Omniglot and miniImageNet datasets, demonstrating that prototypical networks achieve state-of-the-art results, significantly outperforming other models. They reported 98.8% accuracy for 1-shot 5-way classification and 99.7% for 5-shot 5-way classification on Omniglot. Similarly, on miniImageNet, they achieved 49.42% accuracy for 1-shot 5-way and 68.20% for 5-shot 5-way classification. Prototypical Networks tackle the challenge of few-shot learning, where a classifier must generalize to new classes with limited examples, by learning a metric space where classification is based on distances to class prototypes.

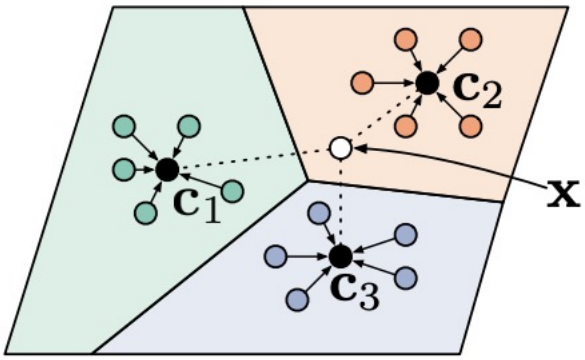

Figure 2.1: Prototypical Networks in the few shop scenario

This approach simplifies the learning process and proves to be more effective than complex meta-learning techniques such as Matching Networks and LSTM-based models. Prototypical Networks compute class prototypes as the mean of support examples in an embedding space generated by a neural network. Classification is performed by finding the nearest prototype using Euclidean distance. Episodic

training, which mimics the few-shot task, enhances generalization. The model also excels in zero-shot learning, outperforming existing methods on the CUB dataset. Prototypical Networks offer a simple, efficient, and a highly effective solution for few-shot and zero-shot learning, with potential for further optimization by exploring different distance metrics and improving the embedding function.

In this [19] research, Sahay et al. discussed the challenges of recognizing handwritten Urdu characters using an enhanced Prototypical Network architecture for few-shot learning. Due to the limited availability of clean, annotated data for Urdu—a language that is context-sensitive and written in a bi-directional script—traditional OCR methods struggle to achieve high accuracy. The proposed solution involves a Prototypical Network that learns Euclidean embeddings of images and uses class prototypes for classification. This study demonstrates the effectiveness of Prototypical Networks in recognizing Urdu characters with minimal examples, paving the way for improved OCR models for low-resource languages.

Shaffi et al. explored the application of Deep Siamese Convolutional Neural Networks (Siamese CNNs) for Tamil handwritten character recognition within the context of few-shot [18] learning. Given the challenge of insufficient training samples in many optical character recognition (OCR) applications, the authors leveraged Siamese CNNs' capability to perform well with minimal data. Their model employs a twin CNN architecture for feature extraction, using cross-entropy loss to validate the learning process. The Siamese CNN achieved an optimal accuracy of 83.39% in a 40-shot learning scenario. Through extensive experimentation on the uTHCD database, which includes a diverse set of Tamil handwritten characters, the study demonstrates the model's robustness and effectiveness in recognizing characters despite the limited data availability. This research sets a new benchmark for developing efficient Indic OCR models using Siamese networks and highlights the potential for further improvements.

Chapter 3

Dataset Description and Preprocessing

3.1 Dataset Description

For my thesis project, I utilized the BanglaLekha-Isolated image dataset, which I downloaded from Mendeley Data. This dataset contains 84 distinct characters, including 50 basic Bangla characters, 10 Bangla numerals, and 24 selected compound characters. Each character has 2,000 handwritten samples that were collected, digitized, and pre-processed. After eliminating errors and scribbles, the dataset comprises a total of 166,105 handwritten character images. All images within each character folder are uniquely labeled.

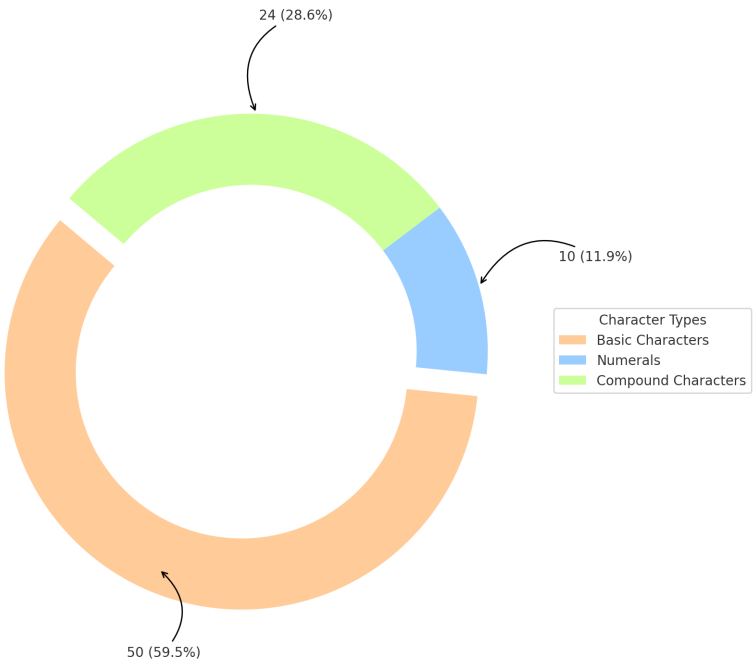

Figure 3.1: Pie Chart with different types of Characters

In few-shot learning, the main goal is to achieve better results with fewer samples.

To align the dataset with this technique, I reduced its size to make it suitable for few-shot learning. I selected 50 basic Bangla characters, 10 Bangla numerals, and 12 compound characters, totaling 72 characters. For each character, I chose 20 image samples. The images for each character are organized into separate classes.

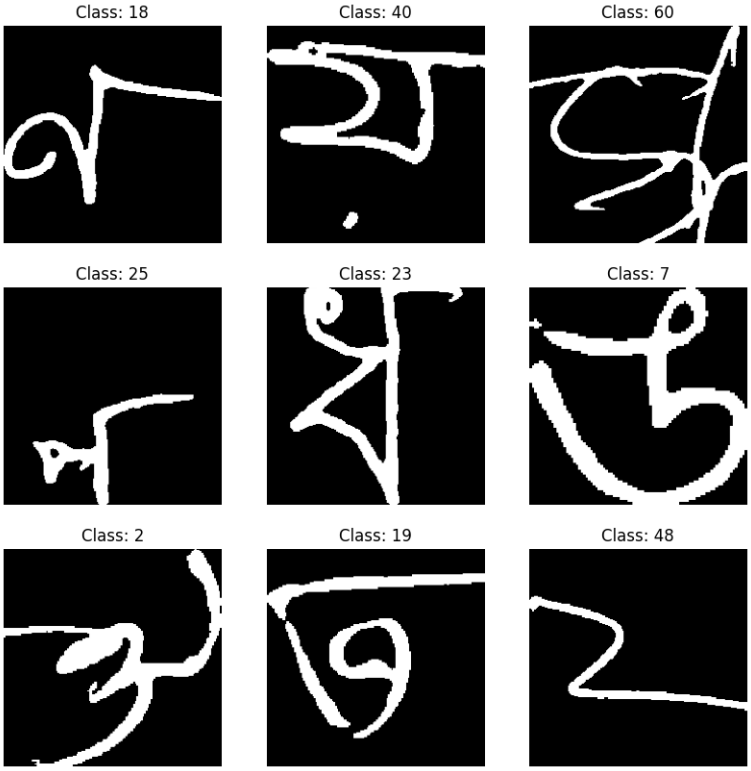

Figure 3.2: Characters divided in classes

3.2 Dataset Preprocessing

The dataset of handwritten characters is loaded and resized to a standard dimension of 224x224 pixels suitable for input into the image classification models I experimented with. Each character class is represented by a folder, with images specific to that class inside. The images are loaded from the dataset directory, converted into numerical arrays, and assigned labels corresponding to their class. After loading, the images are normalized by scaling their pixel values to the range [0, 1], which improves model performance by standardizing the input values. Then, the dataset is split into training and testing sets, with 80% of the data used for training and 20% for testing, while maintaining a balanced distribution of classes using stratified sampling.

The dataset is divided into a support set and a query set to implement a prototypical network for few-shot learning. Each class contains 20 images, with 5 images randomly selected to form the support set, while the remaining 15 images are used as the query set. The support set is used to compute class prototypes, which are the mean embeddings of the support images for each class. By calculating the distance between the query image embeddings and the prototypes, and enabling classifica-

tion based on the nearest prototype, the model is evaluated by the query set. This approach allows the model to generalize to unseen examples by leveraging only a few samples in the support set for each class.

```
Class 25: 20 images
Sample image path: /content/drive/MyDrive/new dataset/25/01_0001_0_11_0916_1918_25.png
Class 14: 20 images
Sample image path: /content/drive/MyDrive/new dataset/14/01_0001_0_11_0916_1918_14.png
Class 49: 20 images
Sample image path: /content/drive/MyDrive/new dataset/49/01_0001_0_12_0916_1912_49.png
Class 5: 20 images
Sample image path: /content/drive/MyDrive/new dataset/5/01_0001_0_12_0916_1905_5.png
Class 2: 20 images
Sample image path: /content/drive/MyDrive/new dataset/2/01_0001_0_12_0916_1932_2.png
Class 71: 20 images
Sample image path: /content/drive/MyDrive/new dataset/71/01_0001_0_11_0916_1986_71.png
Class 47: 20 images
Sample image path: /content/drive/MyDrive/new dataset/47/01_0001_0_11_0916_1917_47.png
Class 22: 20 images
Sample image path: /content/drive/MyDrive/new dataset/22/01_0001_0_11_0916_1928_22.png
Class 13: 20 images
Sample image path: /content/drive/MyDrive/new dataset/13/01_0001_0_11_0916_1913_13.png
Class 40: 20 images
Sample image path: /content/drive/MyDrive/new dataset/40/01_0001_0_11_0916_1922_40.png
Class 24: 20 images
Sample image path: /content/drive/MyDrive/new dataset/24/01_0001_0_12_0916_1914_24.png
Class 48: 20 images
Sample image path: /content/drive/MyDrive/new dataset/48/01_0001_0_11_0916_1923_48.png
Class 70: 20 images
Sample image path: /content/drive/MyDrive/new dataset/70/01_0001_0_11_0916_1913_70.png
Class 46: 20 images
Sample image path: /content/drive/MyDrive/new dataset/46/01_0001_0_11_0916_1924_46.png
Class 41: 20 images
Sample image path: /content/drive/MyDrive/new dataset/41/01_0001_0_12_0916_1912_41.png
```

Figure 3.3: Class wise image path

Query Set (15 Images) for Character 52

Figure 3.4: Query Set

Figure 3.5: Support Set

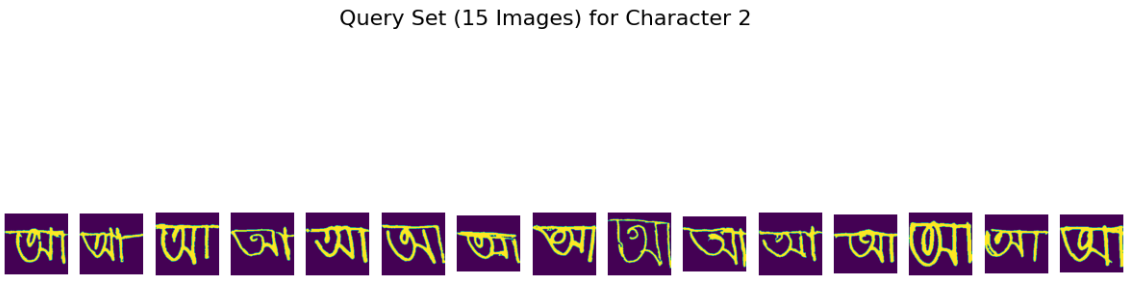

Figure 3.6: Query Set

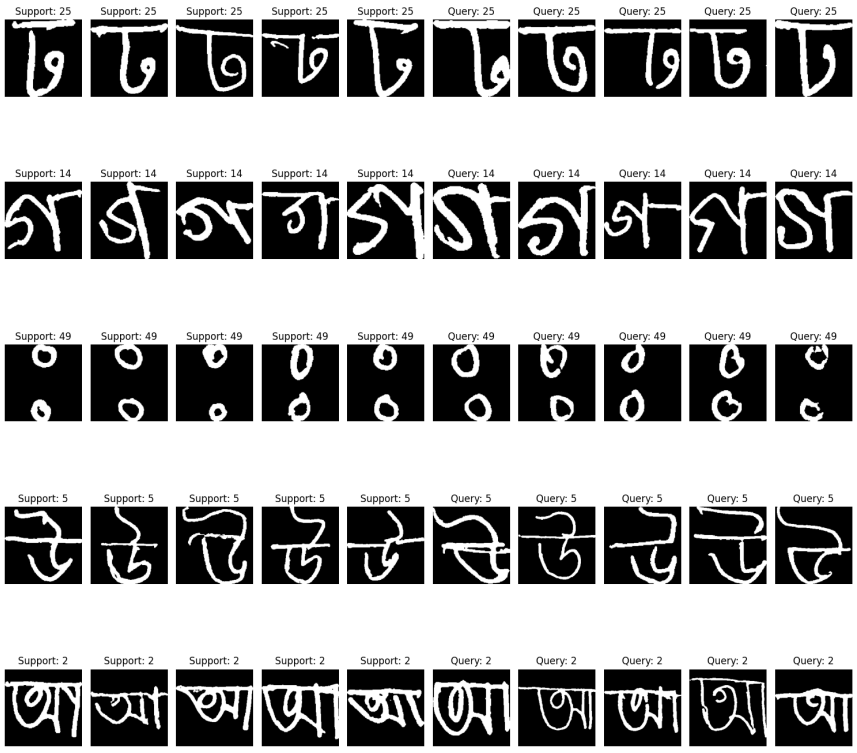

Figure 3.7: Support and Query Set

Chapter 4

Model Description and Result Analysis

4.1 ResNet50

ResNet50 is a deep convolutional neural network consisting of 50 layers, designed to overcome the problem of vanishing gradients that often arise in very deep models. It employs a unique architecture called residual learning, where shortcut connections are added to skip the layers. These shortcuts enable the network to retain important information and make it easier to train, even as the model deepens. ResNet50 is known for its ability to extract complex features from images through multiple layers of convolution, pooling, and activation functions. It is widely used for image classification tasks and is particularly effective when combined with techniques like transfer learning. The ResNet50 model takes images of size 224x224 as input and output feature vectors that represent the key characteristics of the image, which can then be used for classification or other tasks. However, in the context of few-shot learning, this step is modified or replaced as part of the prototypical network setup.

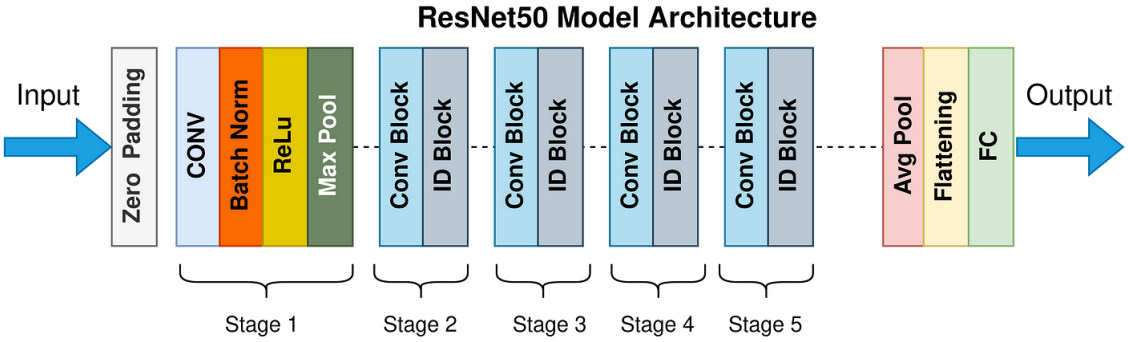

Figure 4.1: ResNet50 Architecture

Here, the ResNet50 backbone is combined with a prototypical network approach. The ResNet50 backbone is used to extract embeddings (feature vectors) for both the support set and query set images. The embeddings of the query set images are

then compared to the class prototypes using a distance metric, which is Euclidean distance in this case. Each query image is assigned to the class whose prototype is closest in the feature space. Based on the distance to the prototypes, the query set images are classified.

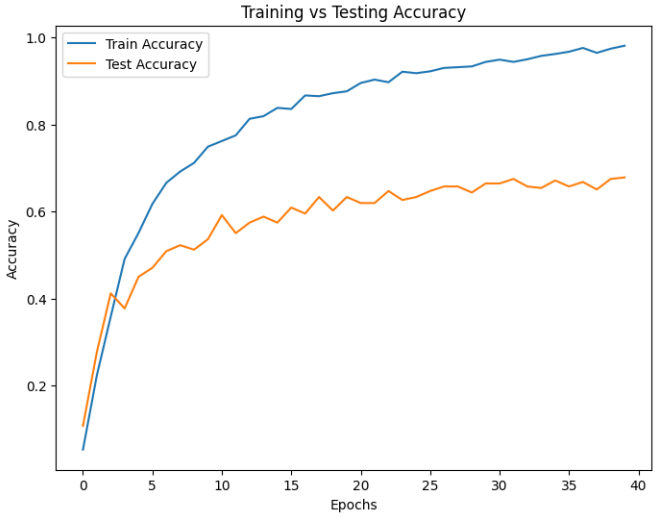

Figure 4.2: ResNet50 Training Vs Testing Accuracy

The training accuracy steadily increases while the testing accuracy increases initially but drops after 0.6 after 10 epochs.

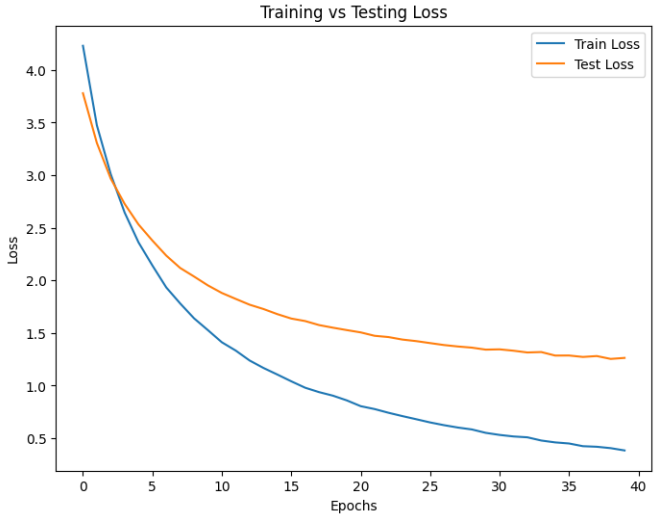

Figure 4.3: ResNet50 Training Vs Testing Loss

The training loss decreases and testing loss remains higher, which means the model performs better on training data than on testing data.

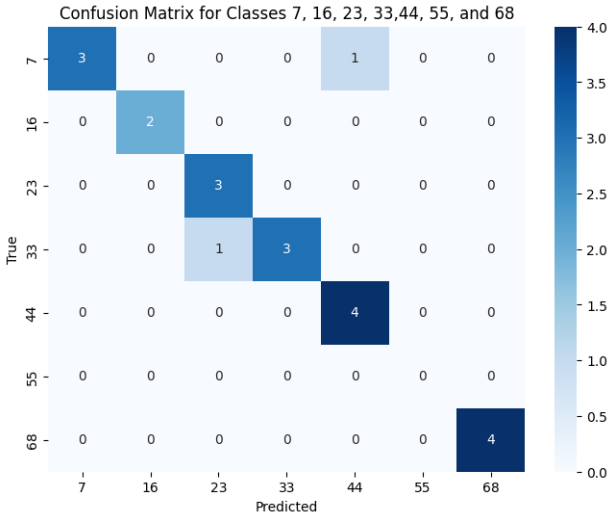

Figure 4.4: ResNet50 Confusion Matrix

The confusion matrix shown above represents the classification performance of the model for six distinct classes: 7, 16, 23, 33, 44, 55, and 68.

Character	precision	recall	f1-score
0	1.00	1.00	1.00
1	0.50	0.25	0.33
2	1.00	0.25	0.40
3	0.50	1.00	0.67
4	0.80	1.00	0.89
5	1.00	1.00	1.00
6	1.00	0.50	0.67
7	1.00	0.75	0.86
8	0.75	0.75	0.75
9	0.00	0.00	0.00
10	1.00	0.75	0.86
11	0.75	0.75	0.75
12	1.00	1.00	1.00
13	0.60	0.75	0.67
14	0.67	0.50	0.57
15	0.75	0.60	0.67
16	0.25	0.25	0.25
17	0.50	1.00	0.67
18	0.20	0.25	0.22
19	0.57	1.00	0.73

20	0.67	1.00	0.80
21	1.00	0.50	0.67
22	0.00	0.00	0.00
23	0.50	0.75	0.60
24	0.75	0.75	0.75
25	0.57	1.00	0.73
26	0.00	0.00	0.00
27	1.00	0.75	0.86
28	1.00	1.00	1.00
29	0.80	1.00	0.89
30	0.75	0.75	0.75
31	0.80	1.00	0.89
32	0.50	0.25	0.33
33	1.00	0.75	0.86
34	1.00	0.50	0.67
35	0.33	0.50	0.40
36	0.25	0.25	0.25
37	0.33	0.50	0.40
38	0.60	0.75	0.67
39	1.00	1.00	1.00
40	0.60	0.75	0.67
41	0.67	0.50	0.57
42	1.00	0.75	0.86
43	0.75	0.75	0.75
44	0.80	1.00	0.89
45	1.00	0.25	0.40
46	0.75	0.75	0.75
47	0.29	0.50	0.36
48	1.00	0.25	0.40
49	0.43	0.75	0.55
50	0.75	0.75	0.75
51	0.67	1.00	0.80
52	0.75	0.75	0.75
53	1.00	1.00	1.00
54	0.67	0.50	0.57
55	0.00	0.00	0.00
56	0.67	0.50	0.57
57	0.80	1.00	0.89
58	0.75	0.75	0.75
59	1.00	0.75	0.86
60	1.00	1.00	1.00
61	0.50	0.75	0.60
62	1.00	1.00	1.00
63	0.67	0.50	0.57
64	0.67	0.50	0.57
65	1.00	1.00	1.00

66	0.50	0.50	0.50
67	1.00	0.75	0.86
68	0.80	1.00	0.89
69	0.67	0.50	0.57
70	0.50	0.75	0.60
71	1.00	1.00	1.00
weighted avg	0.70	0.68	0.66

Table 4.1: Classification report of ResNet50 Model

4.2 VGG16

The VGG16 model is a convolutional neural network commonly used for image classification tasks. It has a total of 16 layers, comprising 13 convolutional layers and 3 fully connected layers. The network employs small 3x3 filters and follows a consistent architectural pattern, which makes it straightforward yet powerful for feature extraction. Pre-trained on the ImageNet dataset, VGG16 is often applied in transfer learning, enabling it to perform well on new tasks with limited training data.

VGG16, a well-known convolutional neural network (CNN), is pre-trained on the ImageNet dataset and has been adapted to create embeddings for the Bangla handwritten character recognition problem. The VGG16 model is used as the base feature extractor. The fully connected layers of VGG16 are removed, keeping only the convolutional layers. This is significant for character recognition.

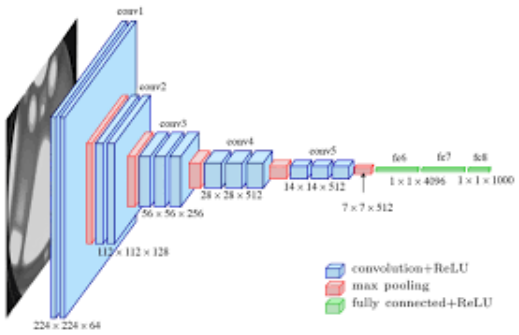

Figure 4.5: VGG16 Model Architecture

The model is initialized with pre-trained ImageNet weights, and its layers are frozen during training to leverage pre-learned features. A GlobalAveragePooling2D layer is added on top of the VGG16 base model, which converts the spatial feature maps into a compact feature vector. This feature vector serves as the embedding for each input image. The embeddings generated by the VGG16 model are used as inputs to the Prototypical Network to compute class prototypes and perform few-shot classification. This architecture combines the robustness of VGG16 for feature extraction with the efficiency of Prototypical Networks for handling limited training samples per class.

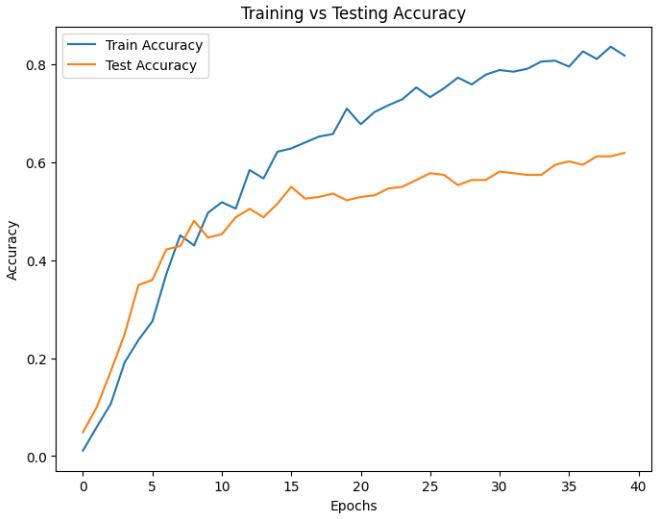

Figure 4.6: VGG16 Model Training Vs Testing Accuracy

The training accuracy increases consistently, reaching around 0.85, while the testing accuracy improves initially but stabilizes at 0.6 after about 10 epochs, indicating overfitting.

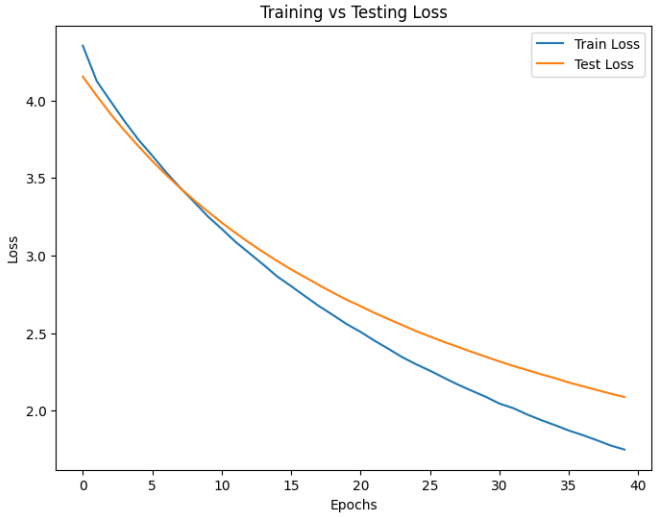

Figure 4.7: VGG16 Model Training Vs Testing Loss

The training loss steadily decreases from around 4.5 to under 2.0, while the testing loss also declines but remains consistently higher.

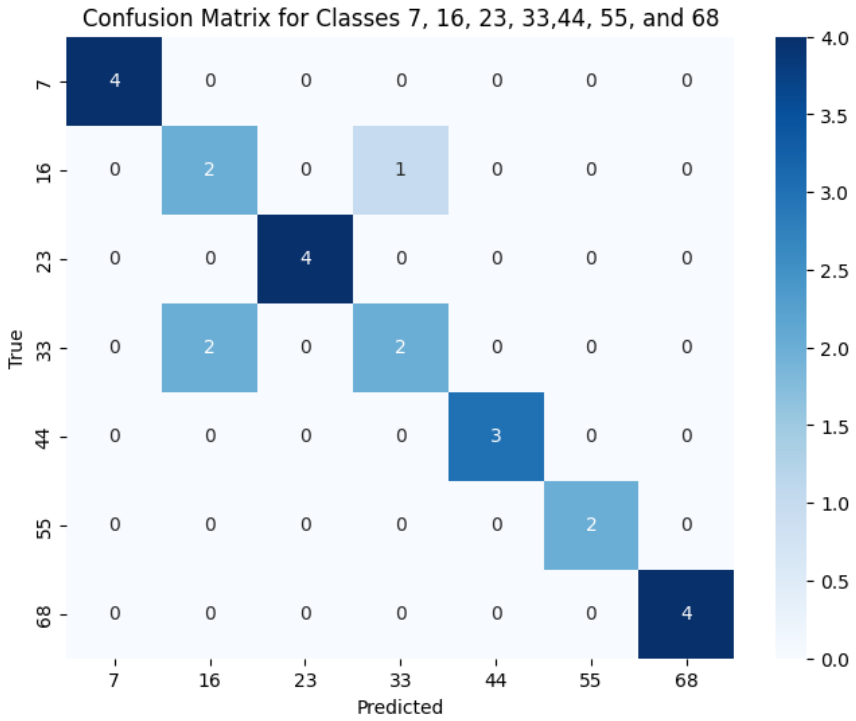

Figure 4.8: VGG16 Confusion Matrix

The confusion matrix shown above represents the model's classification performance for classes 7, 16, 23, 33, 44, 55, and 68. Correct classifications are shown on the diagonal, with perfect predictions for classes 7, 23, and 68. However, misclassifications are observed for classes 16, 33, 44, and 55, such as class 16 being misclassified as class 23 and class 33 as classes 16 and 44.

Character	precision	recall	f1-score
0	0.67	1.00	0.80
1	0.67	0.50	0.57
2	1.00	0.75	0.86
3	1.00	0.25	0.40
4	0.67	1.00	0.80
5	0.40	0.50	0.44
6	0.80	1.00	0.89
7	1.00	1.00	1.00
8	0.33	0.25	0.29
9	0.33	0.25	0.29
10	0.67	0.50	0.57
11	0.57	1.00	0.73
12	0.57	1.00	0.73
13	0.00	0.00	0.00
14	0.60	0.75	0.67

15	0.33	0.25	0.29
16	0.22	0.40	0.29
17	0.75	0.75	0.75
18	0.50	0.75	0.60
19	0.60	0.75	0.67
20	0.40	0.50	0.44
21	1.00	0.50	0.67
22	1.00	0.75	0.86
23	0.67	1.00	0.80
24	0.20	0.25	0.22
25	0.00	0.00	0.00
26	0.57	1.00	0.73
27	0.60	0.75	0.67
28	0.67	0.50	0.57
29	1.00	1.00	1.00
30	0.33	0.25	0.29
31	1.00	0.75	0.86
32	0.60	0.75	0.67
33	0.50	0.50	0.50
34	1.00	0.75	0.86
35	0.67	1.00	0.80
36	1.00	0.25	0.40
37	0.43	0.75	0.55
38	0.50	0.50	0.50
39	0.25	0.25	0.25
40	1.00	1.00	1.00
41	1.00	1.00	1.00
42	0.25	0.25	0.25
43	1.00	0.50	0.67
44	1.00	0.75	0.86
45	1.00	0.75	0.86
46	0.67	0.50	0.57
47	0.33	0.50	0.40
48	1.00	0.50	0.67
49	0.75	0.75	0.75
50	1.00	0.25	0.40
51	0.60	0.75	0.67
52	0.00	0.00	0.00
53	0.67	1.00	0.80
54	0.57	1.00	0.73
55	0.50	0.50	0.50
56	0.00	0.00	0.00
57	1.00	0.25	0.40
58	0.75	0.75	0.75
59	0.67	0.50	0.57
60	0.40	0.50	0.44

61	0.75	0.75	0.75
62	0.00	0.00	0.00
63	1.00	1.00	1.00
64	0.67	1.00	0.80
65	0.67	0.50	0.57
66	0.25	0.25	0.25
67	0.80	1.00	0.89
68	0.80	1.00	0.89
69	0.60	0.75	0.67
70	0.44	1.00	0.62
71	0.75	0.75	0.75
weighted avg	0.62	0.62	0.59

Table 4.2: Classification report of VGG16 Model

4.3 MobileNet

MobileNet is a compact deep learning model optimized for image classification, especially on devices with limited computational resources. It employs a simplified structure using depthwise separable convolutions, which reduces both the number of parameters and computational load, enhancing speed and memory efficiency. Despite its lightweight structure, MobileNet effectively extracts features from images, making it well-suited for transfer learning and applications like few-shot learning, where efficient feature extraction is essential.

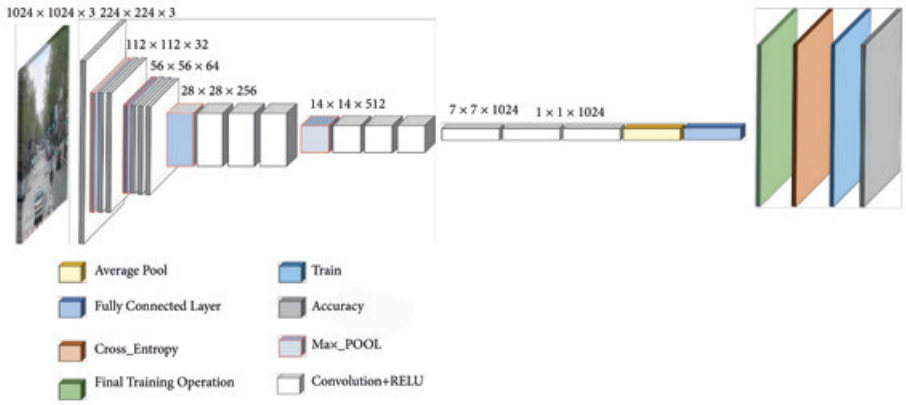

Figure 4.9: MobileNet Model Architecture

The model is built around a pre-trained MobileNet architecture, which is employed for feature extraction. The input size is set to (224, 224, 3) to match the image data, and all images are resized to this shape before processing them by the model. MobileNet is used as the base model in a new architecture, where input images pass through it to produce feature embeddings. These embeddings represent the images

in a lower-dimensional space, enabling comparison in the few-shot learning task. For each class, the feature embeddings of the support images are used to compute a prototype, which is the mean vector of the embeddings. This prototype serves as the class representation in the embedding space. The class whose prototype is closest which has the smallest distance is predicted as the label for the query image. In this experimental setup, MobileNet functions as a feature extractor, generating embeddings that are input into the prototypical network for classifying Bangla characters based on their distance to class prototypes.

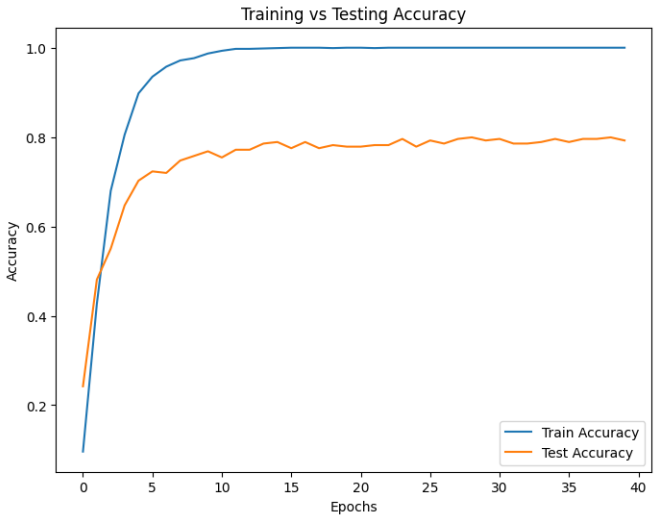

Figure 4.10: MobileNet Model Training Vs Testing Accuracy

The training accuracy increases rapidly while the testing accuracy drops around 80%, which indicates overfitting.

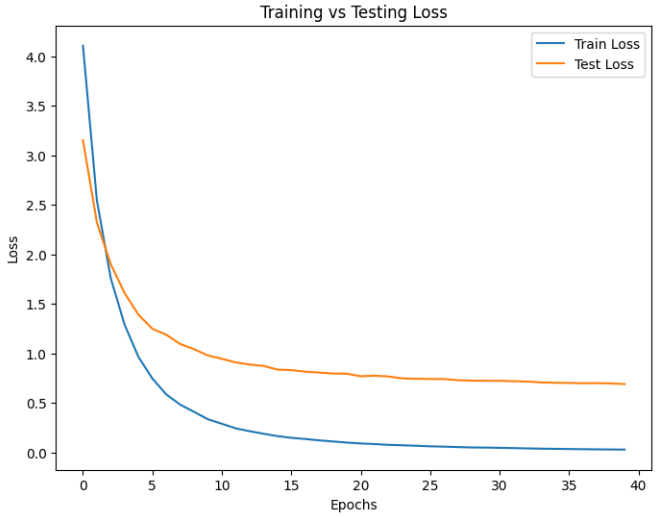

Figure 4.11: MobileNet Model Training Vs Testing Loss

The training loss decreases sharply toward zero, while the testing loss declines more slowly and remains higher, indicating overfitting as the model performs better on the training set than on testing data.

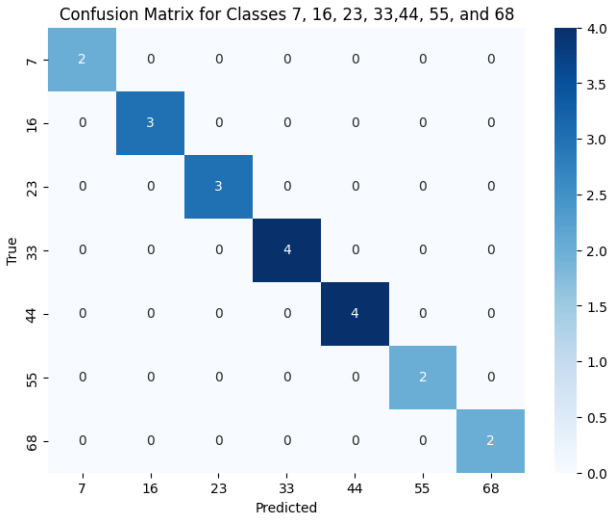

Figure 4.12: MobileNet Confusion Matrix

character	precision	recall	f1-score
0	0.50	0.50	0.50
1	1.00	0.75	0.86
2	0.67	0.50	0.57
3	1.00	1.00	1.00
4	0.80	1.00	0.89
5	1.00	1.00	1.00
6	1.00	0.75	0.86
7	1.00	0.50	0.67
8	0.80	1.00	0.89
9	0.67	0.50	0.57
10	0.67	1.00	0.80
11	1.00	0.75	0.86
12	0.75	0.75	0.75
13	1.00	0.25	0.40
14	0.60	0.75	0.67
15	1.00	0.50	0.67
16	1.00	0.75	0.86
17	1.00	0.75	0.86
18	0.50	0.75	0.60
19	0.80	1.00	0.89
20	1.00	1.00	1.00
21	1.00	0.75	0.86

22	0.60	0.75	0.67
23	1.00	0.75	0.86
24	0.40	0.50	0.44
25	1.00	1.00	1.00
26	0.67	1.00	0.80
27	0.80	1.00	0.89
28	1.00	0.50	0.67
29	1.00	0.75	0.86
30	0.75	0.75	0.75
31	1.00	1.00	1.00
32	1.00	0.75	0.86
33	1.00	1.00	1.00
34	1.00	1.00	1.00
35	0.33	0.25	0.29
36	0.80	1.00	0.89
37	1.00	0.50	0.67
38	1.00	0.75	0.86
39	0.50	0.25	0.33
40	0.80	1.00	0.89
41	0.67	1.00	0.80
42	0.57	1.00	0.73
43	0.60	0.75	0.67
44	1.00	1.00	1.00
45	0.75	0.75	0.75
46	0.75	0.75	0.75
47	0.80	1.00	0.89
48	0.67	1.00	0.80
49	1.00	1.00	1.00
50	0.75	0.75	0.75
51	0.50	0.50	0.50
52	1.00	1.00	1.00
53	1.00	1.00	1.00
54	0.80	1.00	0.89
55	0.67	0.50	0.57
56	1.00	0.75	0.86
57	0.75	0.75	0.75
58	1.00	0.75	0.86
59	0.67	0.50	0.57
60	0.60	0.75	0.67
61	0.56	1.00	0.71
62	0.75	0.75	0.75
63	1.00	1.00	1.00
64	1.00	1.00	1.00
65	1.00	0.75	0.86
66	1.00	1.00	1.00
67	1.00	0.75	0.86

68	0.67	0.50	0.57
69	0.67	1.00	0.80
70	0.80	1.00	0.89
71	0.75	0.75	0.75
weighted avg	0.82	0.79	0.79

Table 4.3: Classification Report of MobileNet

4.4 InceptionV3

The InceptionV3 model is a deep convolutional neural network designed for image classification tasks. It captures important features at multiple scales by using parallel convolutional layers with different filter sizes.The model was pre-trained on large datasets, which allows it to recognize complex patterns in images.

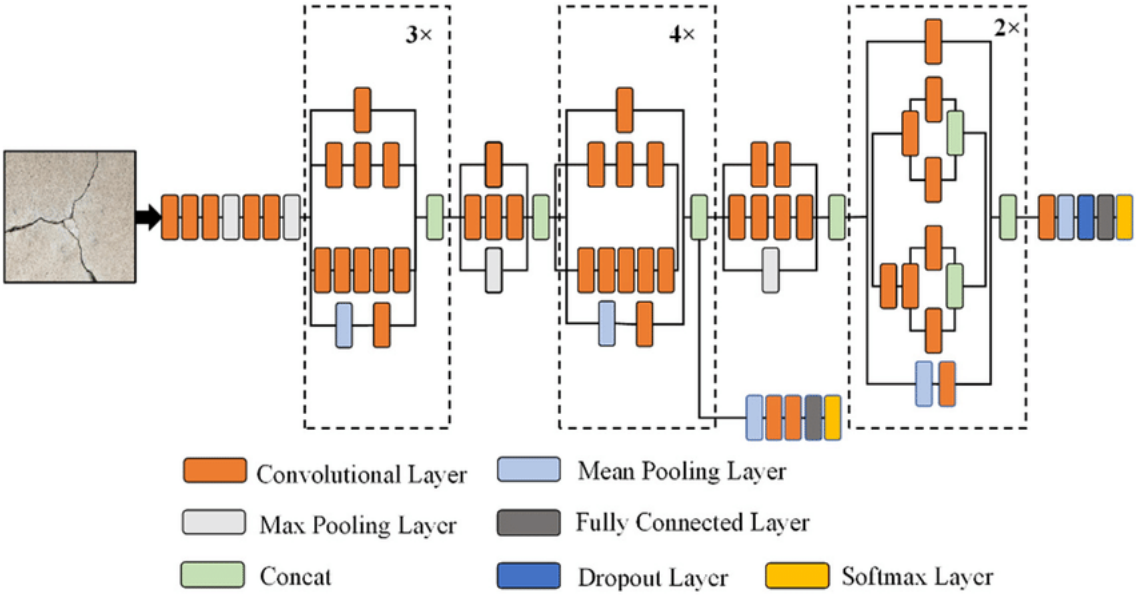

Figure 4.13: InceptionV3 Model Architecture

The convolutional layers from the InceptionV3 model are utilized, and its weights are frozen to retain the pre-trained knowledge. The fully connected layers from the original architecture are removed. The Prototypical Network is employed to perform the few-shot classification. In this setup, the prototypical network leverages the embeddings generated by InceptionV3 to classify images based on their distance from prototype vectors.

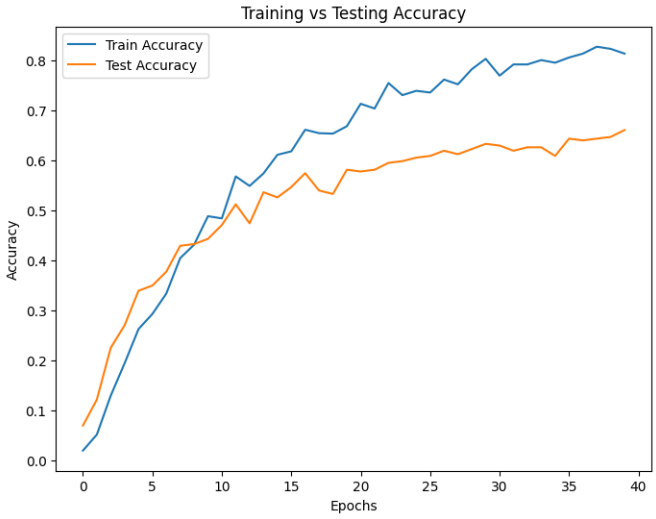

Figure 4.14: InceptionV3 Model Training Vs Testing Accuracy

The training accuracy steadily, while the testing accuracy rises more slowly.

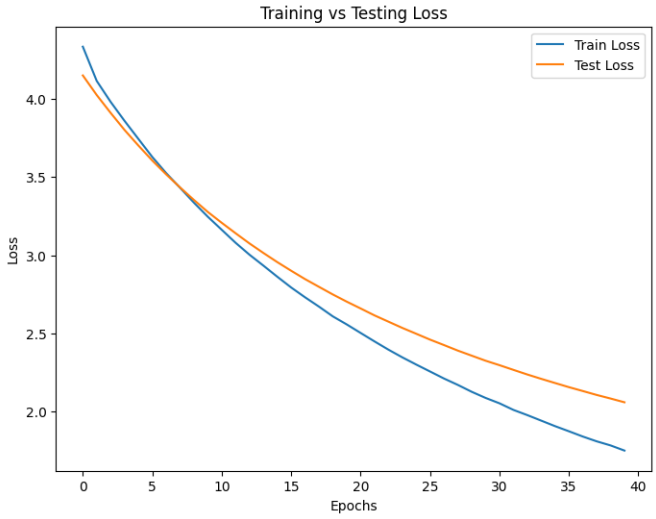

Figure 4.15: InceptionV3 Model Training Vs Testing Loss

The training loss consistently decreases, while the testing loss follows downward but remains higher, indicating overfitting of the model's learning.

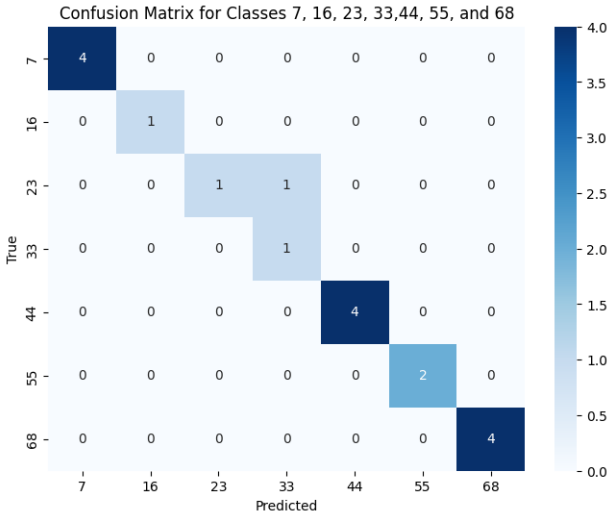

Figure 4.16: InceptionV3 Confusion Matrix

character	precision	recall	f1-score
0	0.80	1.00	0.89
1	0.60	0.75	0.67
2	0.57	1.00	0.73
3	0.00	0.00	0.00
4	0.75	0.75	0.75
5	0.80	1.00	0.89
6	1.00	0.75	0.86
7	0.80	1.00	0.89
8	0.50	0.50	0.50
9	0.80	1.00	0.89
10	0.40	0.50	0.44
11	0.80	1.00	0.89
12	1.00	0.75	0.86
13	0.50	1.00	0.67
14	0.40	0.50	0.44
15	0.30	0.60	0.40
16	1.00	0.25	0.40
17	1.00	0.50	0.67
18	0.33	0.25	0.29
19	0.44	1.00	0.62
20	1.00	0.75	0.86
21	0.50	0.50	0.50
22	0.75	0.75	0.75
23	1.00	0.25	0.40
24	1.00	0.50	0.67
25	0.50	0.50	0.50
26	0.20	0.25	0.22
27	0.75	0.75	0.75
28	0.80	1.00	0.89

29	1.00	0.25	0.40
30	1.00	1.00	1.00
31	1.00	1.00	1.00
32	0.75	0.75	0.75
33	0.25	0.25	0.25
34	0.80	1.00	0.89
35	1.00	0.75	0.86
36	0.00	0.00	0.00
37	0.25	0.25	0.25
38	0.75	0.75	0.75
39	1.00	0.50	0.67
40	1.00	1.00	1.00
41	0.67	1.00	0.80
42	0.67	0.50	0.57
43	1.00	0.75	0.86
44	0.80	1.00	0.89
45	1.00	0.25	0.40
46	0.50	0.25	0.33
47	0.00	0.00	0.00
48	0.33	0.75	0.46
49	0.50	0.75	0.60
50	0.50	0.25	0.33
51	1.00	0.50	0.67
52	1.00	0.75	0.86
53	0.75	0.75	0.75
54	0.67	1.00	0.80
55	0.40	0.50	0.44
56	0.50	0.50	0.50
57	0.57	1.00	0.73
58	0.50	0.75	0.60
59	1.00	0.75	0.86
60	1.00	1.00	1.00
61	0.43	0.75	0.55
62	1.00	1.00	1.00
63	1.00	0.25	0.40
64	0.80	1.00	0.89
65	1.00	0.75	0.86
66	0.00	0.00	0.00
67	0.67	0.50	0.57
68	1.00	1.00	1.00
69	1.00	0.75	0.86
70	0.43	0.75	0.55
71	1.00	1.00	1.00
weighted avg	0.69	0.66	0.64

Table 4.4: Classification Report of InceptionV3

4.5 DenseNet121

The DenseNet121 model is a deep convolutional neural network designed for image classification. It features densely connected layers, with each layer receiving input from all previous layers within the same block.This unique structure allows for efficient feature reuse, improving the flow of information and reducing the number of parameters compared to traditional architectures.

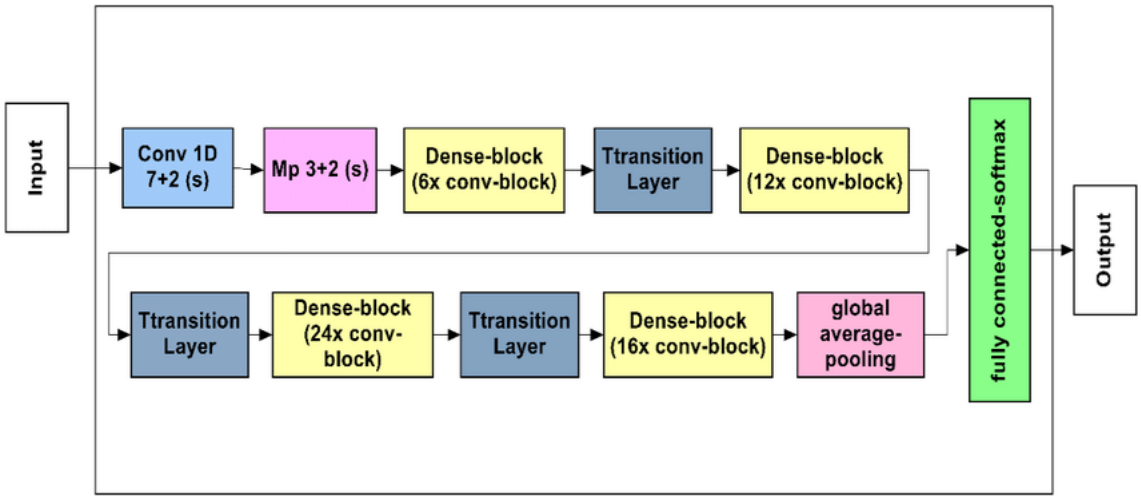

Figure 4.17: DenseNet121 Model Architecture

The model used in this few-shot learning setup is based on DenseNet121, a deep convolutional neural network designed for image classification. In this combination, DenseNet121 extracts meaningful feature vectors from the images, and the prototypical network uses these features to perform a few-shot classification based on the calculated distances between prototypes and query embeddings.

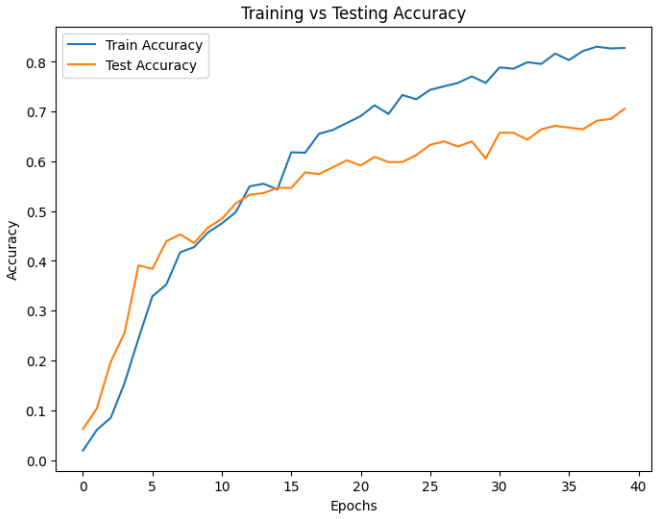

Figure 4.18: DenseNet121 Model Training Vs Testing Accuracy

The training accuracy increases quickly and stabilizes after 20 epochs, while the testing accuracy follows the same but remains slightly lower.

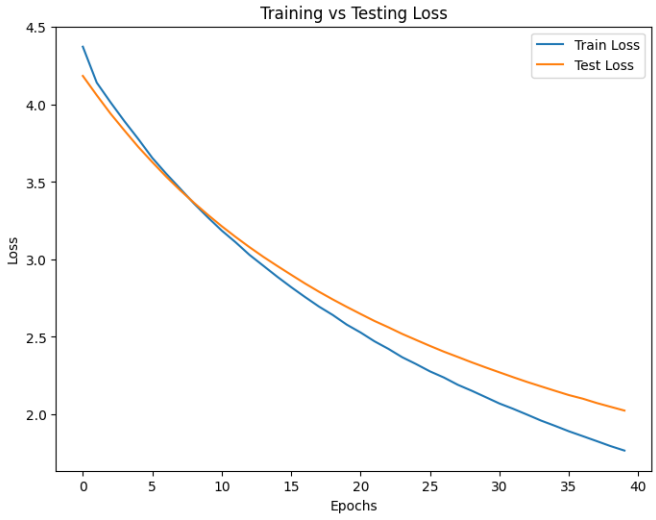

Figure 4.19: DenseNet121 Model Training Vs Testing Loss

The training loss decreases sharply, while the testing loss declines more gradually but remains higher, reflecting the model's ability to reduce errors over time. Both losses show a steady reduction, indicating improved performance across the epochs.

character	precision	recall	f1-score
0	1.00	1.00	1.00
1	0.67	0.50	0.57
2	0.57	1.00	0.73

3	0.80	1.00	0.89
4	0.80	1.00	0.89
5	1.00	1.00	1.00
6	1.00	0.75	0.86
7	0.67	1.00	0.80
8	0.60	0.75	0.67
9	0.50	0.50	0.50
10	0.60	0.75	0.67
11	1.00	1.00	1.00
12	1.00	1.00	1.00
13	0.50	1.00	0.67
14	0.60	0.75	0.67
15	0.33	0.60	0.43
16	0.67	0.50	0.57
17	0.50	0.25	0.33
18	1.00	0.75	0.86
19	0.67	0.50	0.57
20	0.80	1.00	0.89
21	1.00	1.00	1.00
22	1.00	0.50	0.67
23	0.50	0.25	0.33
24	0.75	0.75	0.75
25	0.60	0.75	0.67
26	0.00	0.00	0.00
27	1.00	1.00	1.00
28	1.00	1.00	1.00
29	0.50	0.25	0.33
30	1.00	0.50	0.67
31	0.67	1.00	0.80
32	0.67	0.50	0.57
33	0.25	0.50	0.33
34	0.75	0.75	0.75
35	0.50	0.25	0.33
36	1.00	0.50	0.67
37	0.67	0.50	0.57
38	1.00	0.75	0.86
39	1.00	0.75	0.86
40	0.67	0.50	0.57
41	0.67	1.00	0.80
42	0.75	0.75	0.75
43	0.60	0.75	0.67
44	0.80	1.00	0.89
45	0.75	0.75	0.75
46	1.00	0.75	0.86
47	0.33	0.25	0.29
48	0.75	0.75	0.75

49	0.43	0.75	0.55
50	0.67	0.50	0.57
51	1.00	0.75	0.86
52	1.00	1.00	1.00
53	1.00	0.75	0.86
54	0.75	0.75	0.75
55	0.40	0.50	0.44
56	0.00	0.00	0.00
57	0.57	1.00	0.73
58	0.75	0.75	0.75
59	0.80	1.00	0.89
60	0.80	1.00	0.89
61	1.00	0.75	0.86
62	0.75	0.75	0.75
63	0.50	0.25	0.33
64	0.50	0.25	0.33
65	1.00	0.75	0.86
66	1.00	0.50	0.67
67	0.75	0.75	0.75
68	1.00	1.00	1.00
69	0.67	0.50	0.57
70	0.57	1.00	0.73
71	0.80	1.00	0.89
weighted avg	0.72	0.71	0.69

Table 4.5: Classification Report of DenseNet121

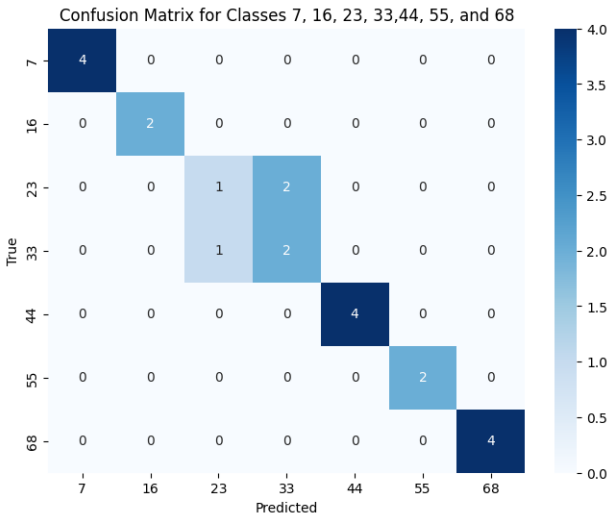

Figure 4.20: DenseNet121 Confusion Matrix

```
Query image 19: Predicted - 5, True - 5
Query image 20: Predicted - 5, True - 5
Query image 21: Predicted - 47, True - 47
Query image 22: Predicted - 47, True - 47
```

Figure 4.21: Character Prediction from Query Set

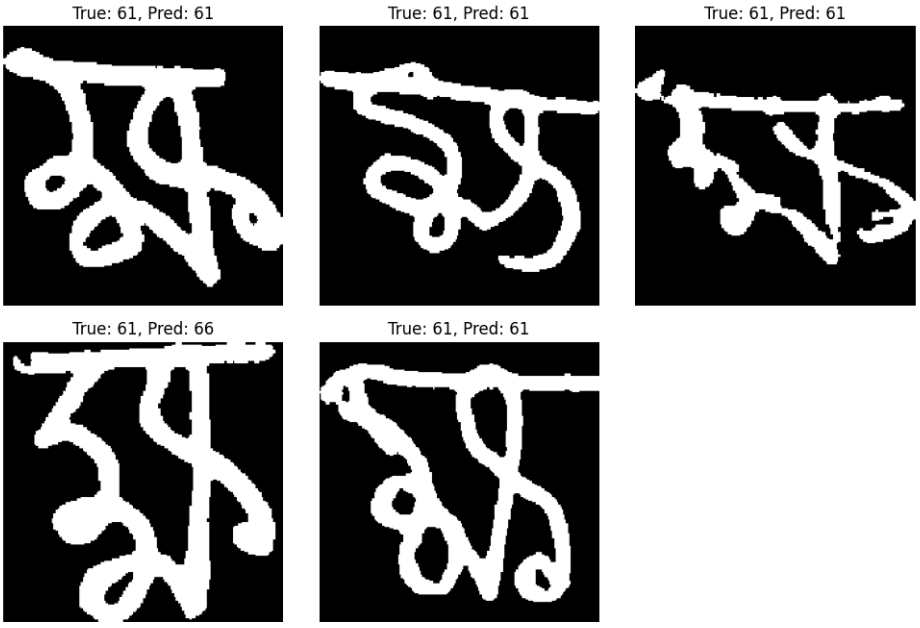

Figure 4.22: Output character Prediction from Query Set(1)

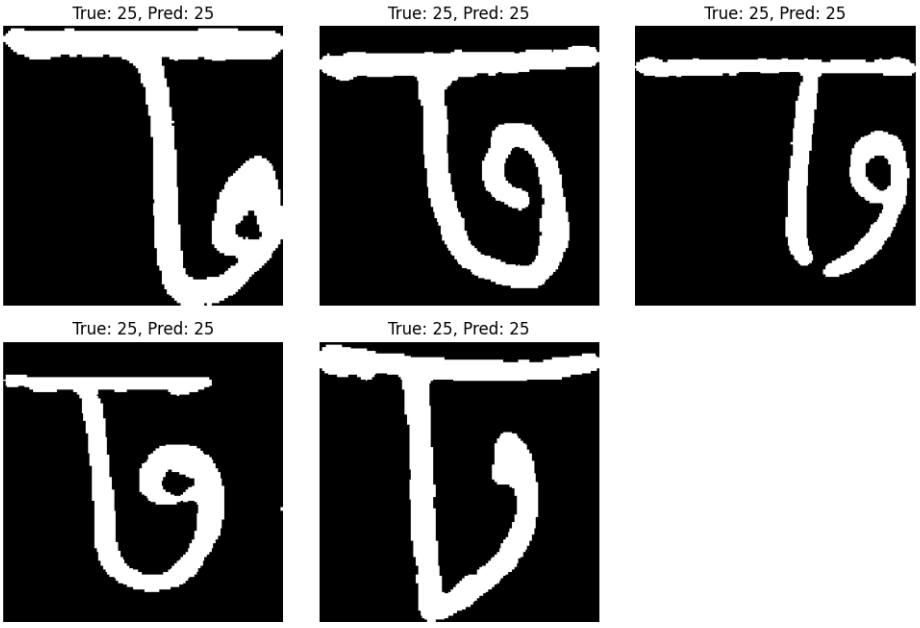

Figure 4.23: Output character Prediction from Query Set(2)

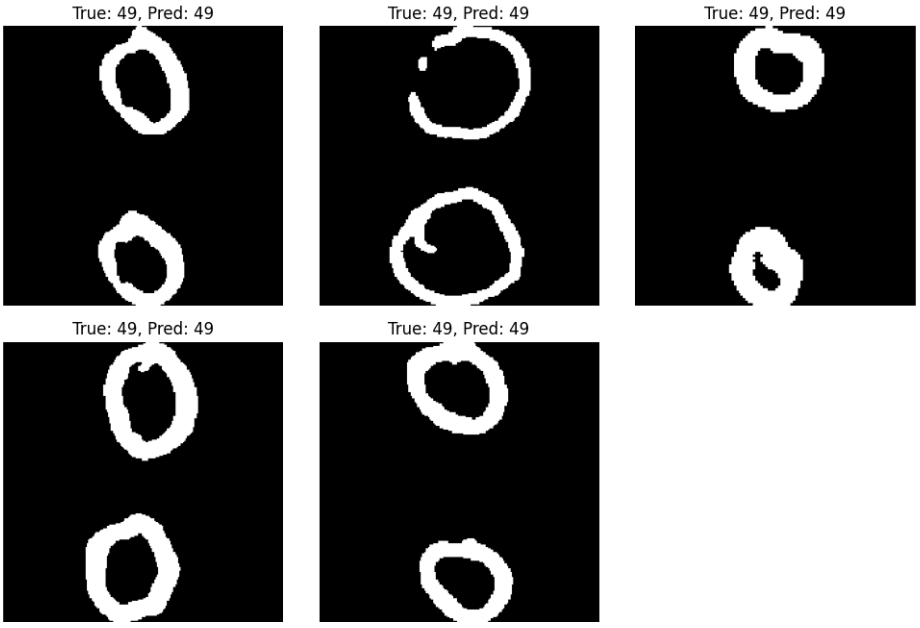

Figure 4.24: Output character Prediction from Query Set(3)

Chapter 5

Comparative Analysis

In this paper,we have experimented with deep learning models: ResNet50, VGG16, MobileNet, InceptionV3, and DenseNet121 using few-shot learning approach with the prototypical network. We evaluated the result using four performance metrics: Precision, Recall, F1 Score, and Accuracy. MobileNet outperforms all the other models, with a precision of 82%, recall of 79%, F1 score of 79%, and accuracy of 79%, making it the best model in this evaluation. On the other hand, VGG16 demonstrates the weakest performance, with a precision of 62%, recall of 62%, F1 score of 59%, and accuracy of 62%, indicating difficulties in handling the dataset for its learning. ResNet50 shows moderate performance, with a precision of 70% and accuracy of 68%, hardly outperforming InceptionV3, which has a precision of 69% than ResNet50 and accuracy of 66%. DenseNet121 performs well, especially in recall and F1 score, achieving 70% and 71% respectively. In conclusion, MobileNet demonstrates the strongest performance, followed by DenseNet121 and ResNet50. VGG16, however, shows the lowest performance in this comparison.

Model	Precision	Recall	F1 Score	Accuracy
ResNet50	0.70	0.68	0.66	0.68
VGG16	0.62	0.62	0.59	0.62
MobileNet	0.82	0.79	0.79	0.79
InceptionV3	0.69	0.66	0.64	0.66
DenseNet121	0.72	0.70	0.70	0.71

Table 5.1: Performance Metrics Table

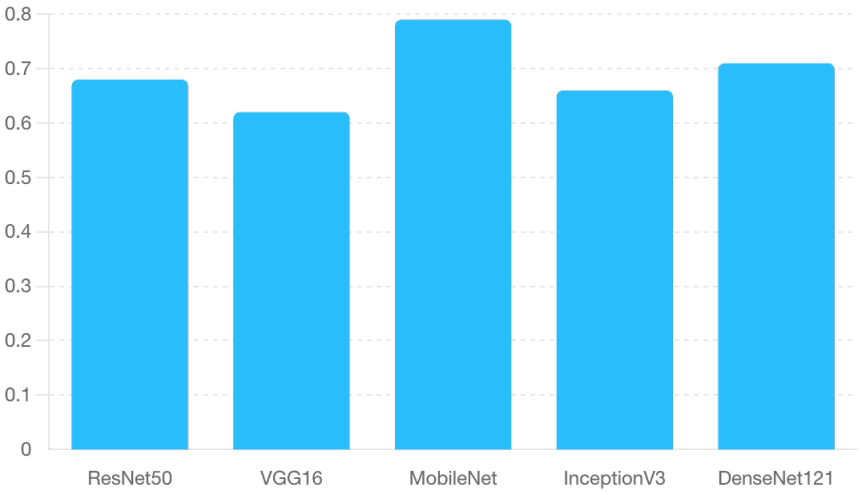

Figure 5.1: Barchart of Accuracy comparison between the implemented algorithm

Chapter 6

Conclusion

6.1 Challenges

One of the key challenges in this research is the lack of prior studies on Bangla character recognition using few-shot learning. While OCR technology with few-shot learning has advanced for many languages, few-shot learning approaches for Bangla characters remain largely unexplored. Additionally, there is a shortage of publicly available resources, such as datasets and pre-trained models with prototypical networks, tailored to languages like Bangla. This scarcity creates a significant obstacle to initiating research in this field, requiring the creation or compilation of suitable datasets, which is time-consuming.
I experimented with various pre-trained models, but they did not perform well on unseen data. Due to time constraints and resource limitations, it was not possible to improve the models' learning accuracy or experiment with additional models.

6.2 Conclusion and Future Work

In this paper, we explored the application of few-shot learning using Prototypical Networks for the handwritten Bangla character recognition. We conducted experiments with several deep learning models, including ResNet50, VGG16, MobileNet, InceptionV3, and DenseNet121, to evaluate their performance using a minimal number of training samples. Among these models, MobileNet demonstrated the best overall performance, followed by DenseNet121 and ResNet50, with VGG16 performing the weakest. The research highlighted the potential of few-shot learning approaches, particularly in scenarios where large datasets are unavailable, which is a common issue in low-resource languages. This work creates opportunities for applying advanced recognition techniques to underrepresented languages, contributing to the development of OCR systems that are inclusive and capable of preserving linguistic diversity. It would further advance the field of character recognition in underrepresented languages, ensuring that digital technologies become more accessible to a wider range of communities for their lifestyle convenience.

Bibliography

[1] M. M. Rahman, M. Akhand, S. Islam, P. C. Shill, M. Rahman, *et al.*, "Bangla handwritten character recognition using convolutional neural network," *International Journal of Image, Graphics and Signal Processing*, vol. 7, no. 8, pp. 42–49, 2015.

[2] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, *et al.*, Eds., vol. 30, Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/cb8da6767461f2812ae4290eac7cbc42-Paper.pdf.

[3] M. Z. Alom, P. Sidike, M. Hasan, T. M. Taha, V. K. Asari, *et al.*, "Handwritten bangla character recognition using the state-of-the-art deep convolutional neural networks," *Computational intelligence and neuroscience*, vol. 2018, 2018.

[4] R. Pramanik and S. Bag, "Shape decomposition-based handwritten compound character recognition for bangla ocr," *Journal of Visual Communication and Image Representation*, vol. 50, pp. 123–134, 2018.

[5] A. S. A. Rabby, S. Haque, S. Abujar, and S. A. Hossain, "Ekushnet: Using convolutional neural network for bangla handwritten recognition," *Procedia computer science*, vol. 143, pp. 603–610, 2018.

[6] A. S. A. Rabby, S. Haque, S. Islam, S. Abujar, and S. A. Hossain, "Bornonet: Bangla handwritten characters recognition using convolutional neural network," *Procedia computer science*, vol. 143, pp. 528–535, 2018.

[7] A. Baldominos, Y. Saez, and P. Isasi, "A survey of handwritten character recognition with mnist and emnist," *Applied Sciences*, vol. 9, no. 15, p. 3169, 2019.

[8] G. Jaume, H. K. Ekenel, and J.-P. Thiran, "Funsd: A dataset for form understanding in noisy scanned documents," in *2019 International Conference on Document Analysis and Recognition Workshops (ICDARW)*, IEEE, vol. 2, 2019, pp. 1–6.

[9] M. RIZVI, K. Deb, M. I. Khan, M. KOWSAR, M. SAKI, and T. KHANAM, "A comparative study on handwritten bangla character recognition," *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 27, no. 4, pp. 3195–3207, 2019.

[10] A. Singh and A. S. Bist, "A wide scale survey on handwritten character recognition using machine learning," *Int J Comput Sci Eng*, vol. 7, no. 6, pp. 124–134, 2019.

[11] M. B. Bora, D. Daimary, K. Amitab, and D. Kandar, "Handwritten character recognition from images using cnn-ecoc," *Procedia Computer Science*, vol. 167, pp. 2403–2409, 2020.

[12] J. Memon, M. Sami, R. A. Khan, and M. Uddin, "Handwritten optical character recognition (ocr): A comprehensive systematic literature review (slr)," *IEEE Access*, vol. 8, pp. 142 642–142 668, 2020. DOI: 10.1109/ACCESS.2020.301254.

[13] I. Zharikov, P. Nikitin, I. Vasiliev, and V. Dokholyan, "Ddi-100: Dataset for text detection and recognition," in *Proceedings of the 2020 4th international symposium on computer science and intelligent control*, 2020, pp. 1–5.

[14] T. R. Das, S. Hasan, M. R. Jani, F. Tabassum, and M. I. Islam, "Bangla handwritten character recognition using extended convolutional neural network," *Journal of Computer and Communications*, vol. 9, no. 03, pp. 158–171, 2021.

[15] Y. B. Hamdan and A. Sathesh, "Construction of statistical svm based recognition model for handwritten character recognition," *Journal of Information Technology and Digital World*, vol. 3, no. 2, pp. 92–107, 2021.

[16] S. Ahsan, T. N. Shah, T. B. Sarwar, M. S. U. Miah, and A. Bhowmik, "A machine learning approach for bengali handwritten vowel character recognition," *IAES International Journal of Artificial Intelligence*, vol. 11, no. 3, p. 1143, 2022.

[17] M. M. Khan, M. S. Uddin, M. Z. Parvez, and L. Nahar, "A squeeze and excitation resnext-based deep learning model for bangla handwritten compound character recognition," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 6, pp. 3356–3364, 2022.

[18] M. Samuel, L. Schmidt-Thieme, D. Sharma, A. Sinamo, and A. Bruck, *Offline handwritten amharic character recognition using few-shot learning*, Oct. 2022.

[19] R. Sahay and M. Coustaty, "An enhanced prototypical network architecture for few-shot handwritten urdu character recognition," *IEEE Access*, vol. 11, pp. 33 682–33 696, 2023.