Dissertation on

Classification of Bangla Handwritten Characters using VGG19-based Deep Learning Model

Thesis submitted towards partial fulfilment
of the requirements for the degree of

Master in Multimedia Development

Submitted by
ARGHYA MONDAL

EXAMINATION ROLL NO. : M4MMD24005
UNIVERSITY REGISTRATION NO. : 163786 of 2022-23

Under the guidance of
Dr. SASWATI MUKHERJEE

School of Education Technology
Jadavpur University

Course affiliated to
Faculty of Engineering and Technology
Jadavpur University
Kolkata-700032
India

2024

Master in Multimedia Development
Course affiliated to
Faculty of Engineering and Technology
Jadavpur University
Kolkata, India

CERTIFICATE OF RECOMMENDATION

This is to certify that the thesis entitled "**Classification of Bangla Handwritten Characters using VGG19-based Deep Learning Model**" is a bonafide work carried out by **ARGHYA MONDAL** under our supervision and guidance for partial fulfilment of the requirements for the degree of **Master in Multimedia Development** in **School of Education Technology**, during the academic session 2023-2024.

Dr Saswati Mukherjee
SUPERVISOR
School of Education Technology
Jadavpur University,
Kolkata-700 032

DIRECTOR
School of Education Technology
Jadavpur University,
Kolkata-700 032

DEAN - FISLM
Jadavpur University,
Kolkata-700 032

Master in Multimedia Development
Course affiliated to
Faculty of Engineering and Technology
Jadavpur University
Kolkata, India

CERTIFICATE OF APPROVAL **

This foregoing thesis is hereby approved as a credible study of an engineering subject carried out and presented in a manner satisfactory to warranty its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval, the undersigned do not endorse or approve any statement made or opinion expressed or conclusion drawn therein but approve the thesis only for the purpose for which it has been submitted.

Committee of final examination for evaluation of Thesis

** Only in case the thesis is approved.

DECLARATION OF ORIGINALITY AND COMPLIANCE OF ACADEMIC ETHICS

I hereby declare that this thesis contains a literature survey and original research work by the undersigned candidate as part of his **Master in Multimedia Development** studies.

All information in this document has been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by this rule and conduct, I have fully cited and referenced all materials and results that are not original to this work.

NAME : ARGHYA MONDAL

EXAMINATION ROLL NUMBER : M4MMD24005

REGISTRATION NUMBER : 163786 of 2022-23

THESIS TITLE : Classification of Bangla Handwritten Characters using VGG19-based Deep Learning Model

SIGNATURE: DATE:

ACKNOWLEDGEMENT

I feel fortunate to have the opportunity to present this dissertation at the School of Education Technology, Jadavpur University, Kolkata, as part of the requirements for the degree of **Master in Multimedia Development**.

I would like to express my sincere gratitude to my mentor, **Dr. Saswati Mukherjee**, for her invaluable guidance, support, advice, constructive criticism, and insightful perspectives throughout this dissertation. Her assistance has been instrumental in shaping my work.

I extend my heartfelt thanks to **Prof. Matangini Chattopadhyay**, Director of the School of Education Technology, for her continuous encouragement, support, and guidance. I am also grateful to Mr. Joydeep Mukherjee for his unwavering support during the course of my work. I would also like to acknowledge my classmates from the Master in Multimedia Development and M.Tech IT (Courseware Engineering).

I wish to express my appreciation to all the departmental support staff and everyone else who contributed to this dissertation.

Lastly, I would like to convey my special gratitude to my parents, whose unwavering support and sacrifices have played a significant role in helping me reach this milestone.

Dated :
Place :
Regards,

Arghya Mondal
Examination Roll No. M4MMD24005
Registration No. 163786 of 2022-23
Master in Multimedia Development
School of Education Technology
Jadavpur University
Kolkata-700032

Contents

Topic	Page No.
Lists of Figures	i
Executive Summary	ii
1. Introduction 1.1. Overview 1.2. Problem Statement 1.3. Objective	1-2
2. Background Concept 2.1. Convolution Neural Network 2.2. VGG19 2.3.Performance Measure	3-8
3. Literature Survey	9-11
4. Proposed Methodology 4.1. Dataset Description 4.2. Data Preprocessing 4.3. Model Architecture	12-15
5. Experiments and Results	16-18
6. Comparative Analysis	19-20
7. Conclusion and Future Scope	21
Reference	22-24
Appendix	25-28

List of Figures

Figure 1: CNN Architecture

Figure 2: Max pooling operation

Figure 3: Average pooling operation

Figure 4: Confusion matrix

Figure 5: A handwritten Bangla character written by different individuals

Figure 6: Modified VGG19 architecture

Figure 7: Accuracy Curve

Figure 8: Loss Curve

Executive Summary

In the past decade, many deep learning models have been developed to address handwritten character classification in various languages, including English, Chinese, Arabic, Japanese, and Russian. Despite these advancements, classifying Bangla handwritten characters from document image datasets remains a challenging and open problem. The complexity of the Bangla script, along with the inherent variability in individual handwriting styles, presents unique difficulties that demand sophisticated solutions. Nonetheless, advancements in neural network technology have led to the development of numerous promising models that improve classification performance.

The Bangla language is characterised by a rich set of characters, including numerals, basic characters, and complex compound and modifier characters. This diversity in character forms, coupled with the intricate and often cursive nature of the script, complicates the classification process. Individual handwriting styles vary significantly, further adding to the challenge. As a result, effective classification systems must be capable of handling a wide range of variations in both character forms and writing styles.

Convolutional Neural Networks (CNNs) have emerged as powerful tools for image classification tasks, including handwritten character classification. In this dissertation, a popular CNN model, VGG19, has been used to classify Bangla handwritten characters. The dataset used for this work is 'BanglaLekha-Isolated,' a standard dataset that includes a variety of Bangla characters, comprising 50 basic characters, 10 numerals, and 24 frequently used compound characters.

The modified VGG19 model's performance is compared with traditional classifier-based approaches. The experimental results demonstrated that the VGG19 significantly outperforms the traditional methods. Specifically, the VGG19 model achieved impressive validation accuracy on the Bangla-Lekha-Isolated dataset. Additionally, the model obtained a high F1 Score, indicating its reliability in classifying Bangla handwritten characters.

1. Introduction

1.1. Overview

In the modern era of science and technology, the preference for digitalisation has become omnipresent. Handwritten character classification plays a crucial role in converting handwritten characters into digital text files, as well as in applications like vehicle license plate number classification, ID number classification, parking lot management, banking, etc [1]. By eliminating the labour-intensive and time-consuming process of manual data entry, this technique significantly enhances human-computer interaction. This technique significantly enhances human-computer interaction and is now employed in various important industries.

Bengali, or Bangla, is the second most widely spoken language in the Indian subcontinent. Globally, it ranks as the fifth most-spoken native language and the seventh most-spoken language overall, with approximately 300 million native speakers and an additional 37 million second-language speakers. Bangla is the official and national language of Bangladesh, with 98% of Bangladeshis speaking it as their first language. Additionally, Bangla is also spoken by significant populations in the Indian states of West Bengal, Tripura, and Assam. It is the official language of the states of West Bengal and Tripura and is also used in the Andaman and Nicobar Islands. A slightly modified version is used for the Assamese writing format [2].

Bangla is also essential for its rich cultural and literary heritage. Many people prefer to write documents in the Bangla language, emphasising the need to preserve these important handwritten documents using digital technology. However, without proper digital maintenance, these valuable documents cannot be effectively preserved. In recent years, significant work has been done in handwritten character classification for the Bangla language. Nonetheless, substantial challenges remain in developing an effective system that yields accurate results in Bangla character classification.

1.2. Problem Statement

Classification of Bangla handwritten characters using VGG19-based deep learning model

1.3. Objective

The objectives are as follows.

a) Develop a deep learning-based CNN model to classify Bangla handwritten characters.
b) Utilize the VGG19 model to train and evaluate a large dataset of diverse Bangla characters.
c) Enhance classification accuracy by adjusting key network parameters of the VGG19 model.

2. Background Concept

2.1. Convolution Neural Network

There are various machine-learning techniques available for classifying Bangla handwritten characters. Some popular techniques include clustering, feature extraction, pattern matching, and artificial neural networks. However, the most effective method is using convolutional neural network (CNN) algorithms. CNNs are a type of deep learning network that directly learns from data. They are particularly useful for identifying image patterns to classify objects, classes, and categories [3]. Additionally, CNNs can be quite effective for classifying audio, time series, and signal data. This specialised deep learning algorithm takes input images and performs a mathematical operation called convolution to differentiate between them. The CNN architecture is presented in Figure 1.

CNN Architecture

A Convolutional Neural Network consists of multiple layers, such as the input layer, Convolutional layer, Pooling layer, and fully connected layers.

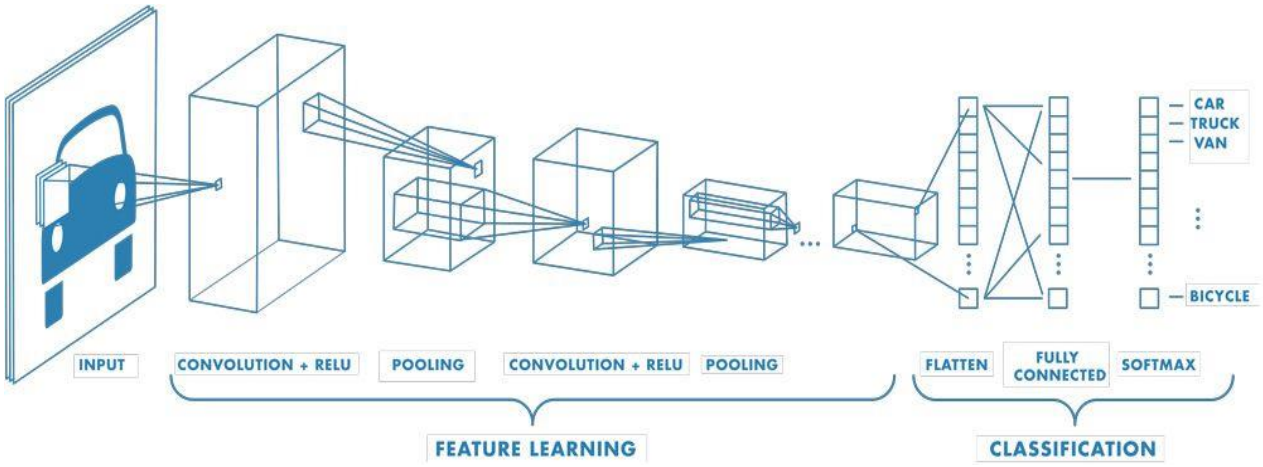

Figure 1: CNN Architecture

Key Components

a) Input Layer

The input to the model is given in this layer. In CNN, generally, an image or a sequence of images will be the input. The raw input of the image, with a width of 32, height of 32, and depth of 3, is held in this layer.

b) Convolutional Layer

- Filters/Kernels: This layer extracts the feature from the input dataset. It applies a set of learnable filters known as the kernels to the input images. The filters/kernels are smaller matrices, usually 2×2, 3×3, or 5×5 shape.
- Stride: The step size with which the filter moves across the input image.
- Padding: Adding borders to the input image to control the spatial size of the output feature map during the convolution operation. It helps prevent information loss at the edges and plays a vital role in the architecture and performance of convolutional neural networks.

c) Activation Layer

Commonly used activation functions include ReLU (Rectified Linear Unit), which introduces non-linearity by setting all negative values to zero while retaining positive values.

d) Pooling Layer

- Max Pooling: It reduces the size of the volume, which speeds up the computation, reduces memory usage, and prevents overfitting. As the filter moves over the image, it selects the maximum pixel value. These selected values make the output array, as shown in Figure 2.

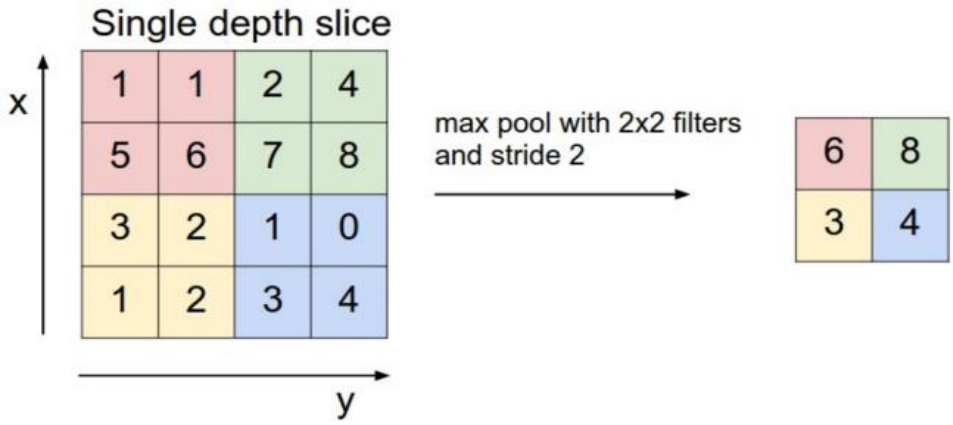

Figure 2: Max pooling operation

- Average Pooling: Reduces the dimensions by taking the average of values in a specified window. As the filter moves over the image, the average value is calculated and sent to the output array, as shown in Figure 3.

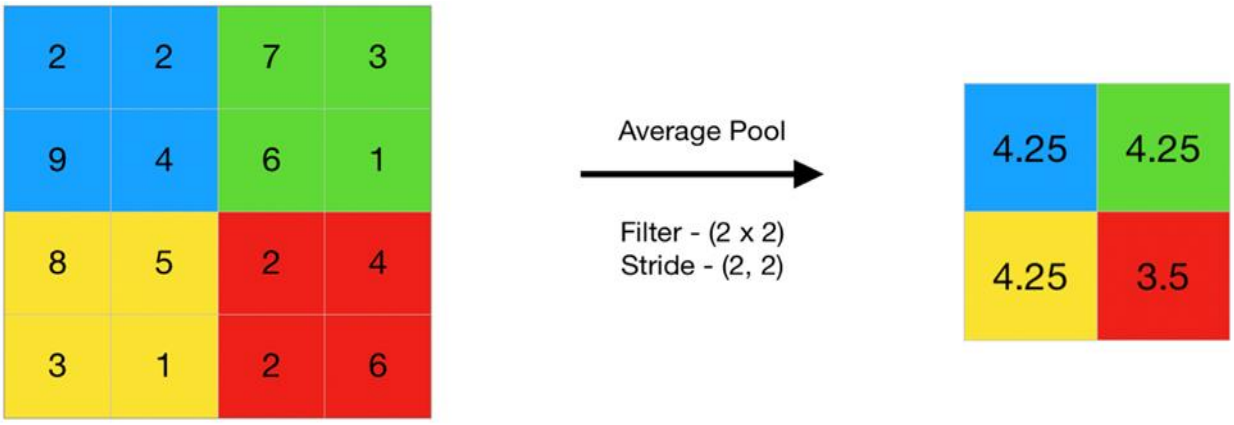

Figure 3: Average pooling operation

e) Flatten

The flatten layer appears after the convolutional and pooling layers. The flatten layer makes the multidimensional input one-dimensional, commonly used in transitioning from the convolution layer to the fully connected layer.

f) Fully Connected Layer

Neurons in these layers are connected to all activations from the previous layer, like traditional neural networks, enabling the combination of extracted features for final classification.

g) Batch Normalization Layer

- Normalizes the activations of the previous layer to improve the stability and speed up training.
- Helps in mitigating the vanishing or exploding gradients problem.

h) SoftMax

The SoftMax function is a mathematical function that converts a vector of real numbers into a probability distribution and gives the output between 0 and 1. It is applied just before the output layer.

i) Output Layer

The output layer uses a SoftMax activation function for classification tasks, providing the probabilities of each class.

2.2. VGG19

The VGG19 model is a deep Convolutional Neural Network (CNN) architecture developed by the Visual Geometry Group at the University of Oxford. It is part of the VGG family of models, which are known for their simplicity and effectiveness in image classification tasks. VGG19 is an extension of the VGG16 model [4], which consists of 19 layers, including 16 convolutional layers, 3 fully connected (dense) layers, and 5 max-pooling layers. VGG19 has been successfully applied to various image classification tasks, including handwritten character classification. The model's depth and capability to capture intricate features make it suitable for classifying complex and diverse handwriting styles. Here's how VGG19 [5, 6] can be specifically applied to handwritten character classification, focusing on Bangla characters.

Layers

a. Input Layer

Takes an image of size 32x32x3 (height, width, and colour channels).

b. Convolutional Layers

- Two convolutional layers with 64 filters each, followed by max pooling.
- Two convolutional layers with 128 filters each, followed by max pooling.
- Four convolutional layers with 256 filters each, followed by max pooling.
- Four convolutional layers with 512 filters each, followed by max pooling.
- Four convolutional layers with 512 filters each, followed by max pooling.

c. Activation Function

Each convolutional layer is followed by a Rectified Linear Unit (ReLU) activation function, introducing non-linearity to the model.

d. Pooling Layers

Max-pooling layers with a 2x2 filter and a stride of 2 are used after each set of convolutional layers to reduce the spatial dimensions and retain the most significant features.

e. Fully Connected Layers

- After the convolutional and pooling layers, the network includes three fully connected (dense) layers:
- Two layers with 4096 neurons each.
- One layer with 1000 neurons, corresponding to the number of classes.

f. Output Layer

The final layer uses the SoftMax activation function to distribute probability over the classes.

2.3. Performance Metrics

A Convolutional Neural Network (CNN) is evaluated using performance metrics similar to those in other machine learning models. However, these measures are specifically adapted for tasks such as image classification, object detection, and segmentation. Here's a detailed explanation of these key performance metrics, including the relevant formulas and variables:

- **Accuracy:** Accuracy is used to measure the performance of the model. It is the ratio of Total correct instances to the total instances. Accuracy can be calculated as Accuracy = (TP + TN)/(TP + TN + FP + FN)
- **Confusion Matrix:** Confusion Matrix is a performance measurement for the machine learning or CNN classification problems where the output can be two or more classes. A typical confusion matrix looks like the Figure 4.

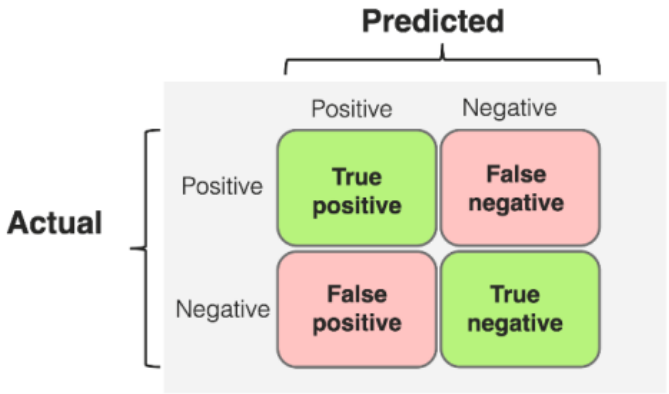

Figure 4. Confusion Matrix

In general, the table is divided into four terminologies, which are as follows:

1. True Positive (TP): In this case, the prediction outcome is true, and it is true in reality, also.
2. True Negative (TN): in this case, the prediction outcome is false, and it is false in reality, also.
3. False Positive (FP): In this case, prediction outcomes are true, but they are false in actuality.
4. False Negative (FN): In this case, predictions are false, and they are true in actuality.

- **Precision:** It measures how accurate the model is in identifying positive instances. It measures the accuracy of positive predictions. It can be calculated as Precision= TP/(TP+FP)
- **Recall:** Recall measures the effectiveness of a classification model in identifying all relevant instances from a dataset. It can be calculated as Recall = TP/(TP + FN)
- **F1-Score:** It provides a balance between precision and recall. It can be calculated as F1-Score = (2 x Precision x Recall)/(Precision+Recall)

3. Literature Survey

Rabby et al. [7] developed a multiclass CNN model called Borno. They evaluated this model using an assembled dataset of 1,069,132 images and reported an accuracy of 91.88%. The Borno model consists of four convolutional layers with 32, 64, 128, and 256 filters, respectively. Each convolutional layer is followed by a batch normalisation layer, which is then connected to a max-pooling layer and a dropout layer.

Rahman et al. [8] found that a CNN model could classify only simple Bangla characters (50 classes) with a testing accuracy of 85.36%. The model's performance improves with more iterations, reaching close to 300. However, despite the convolutional and other layers used, the model still shows a significant limitation: its testing accuracy (85.36%) is noticeably lower than its training accuracy (93.93%) which indicates that the model is overfitting to the training data.

Purkaystha et al. [9] developed a deep convolutional neural network (DCNN) for recognising Bangla characters. Their model achieved an accuracy of 91.23% for recognising 50 alphabet categories and 89.93% for recognising nearly all Bangla characters across 80 categories. Their approach is relatively complex, utilising more layers in the network.

Rumman et al. [10] utilised the Bangla-Lekha-Isolated dataset and applied a Convolutional Neural Network (CNN) to it, achieving an accuracy of 91.81%. Additionally, by using data augmentation techniques, which involve artificially increasing the size and variability of the training data, they improved the model's performance further, reaching an accuracy of 95.25%. This demonstrates the effectiveness of data augmentation in enhancing the accuracy of CNN models for recognising Bangla characters.

Shaik et al. [11] introduced a novel method for Bangla character classification that utilises a layer-based and view-based approach with a KNN classifier. Another technique for classifying Bangla handwritten characters employs the Euclidean distance measurement technique and the Fourier Transform (FT)

measurement technique. Subsequently, researchers began implementing CNN models for this task.

Alif et al. [12] introduced a modified version of the ResNet-18 architecture tailored for recognising Bangla handwritten characters in their study. Their modification involved integrating dropout layers into the ResNet-18 architecture, enhancing its classification performance. The researchers applied this customised architecture to both the BanglaLekha-Isolated dataset and the CMATERdb dataset. They achieved impressive accuracies of 95.10% and 95.99%, respectively, demonstrating the effectiveness of their approach in accurately recognising Bangla handwritten characters across different datasets.

A reliable model [13] presented a fuzzy technique for segmenting handwritten Bangla word images. Initially, they identify the Matra, the longest straight line connecting multiple characters to form a Bangla word, using fuzzy features extracted from the target word image. Subsequently, they identify segment points within the Matra using three fuzzy features. Their experiment utilised only 210 samples of handwritten Bangla words, yielding an average accuracy of 95.32%. This method showcases a novel approach to segmenting Bangla handwritten words with promising results despite the limited sample size used in the study.

An ensemble strategy [14] is proposed by detecting and correcting any skew in the words. Subsequently, they estimate the headline and segment the words into meaningful pseudo characters. They extract three distinct statistical features, combine them, and apply a CNN-based transfer learning architecture. Following this, they merge the identified pseudo characters to reconstruct the full word. The proposed segmentation methodology achieved an accuracy of 94.01% in recognising Bangla words from images, showcasing promising results in word recognition from handwritten text.

El-Sawy et al. [15] developed a convolutional neural network for detecting handwritten Arabic characters. They used 16,800 images of Arabic characters in their experiment. Their system consists of two convolutional layers, two pooling layers, and two fully connected layers. This system achieved an average accuracy of 94.9%, showing its effectiveness in recognising handwritten Arabic characters.

Yang et al. [16] offer a comprehensive overview of using deep learning techniques in handwritten character recognition. Their study encompasses many deep learning models, notably convolutional neural networks, recurrent neural networks, and deep belief networks.

In a study by Pal et al. [17], the modified quadratic discriminant function (MQDF) was used to recognise Bangla compound characters, resulting in an accuracy of 85.90%. Although MQDF is an advanced classifier for handwriting recognition and fits the training data well, its generalisation performance is poor. To address this limitation, a CNN-based model has been proposed to improve performance, accuracy, and robustness.

4. Proposed Methodology

This work aims to create a CNN-based classification model for identifying Bangla handwritten characters using the VGG19 architecture and the BanglaLekha-Isolated dataset. Specifically, the VGG19 model is fine-tuned better to discern the unique features of handwritten Bangla characters. After the initial training phase, certain layers of the VGG19 model are unfrozen for additional fine-tuning, which extends over 10 epochs using a secondary learning rate scheduler. The dataset, consisting of 84 classes, is pre-processed using augmentation techniques such as rotations, shifts, shearing, and zooming to enhance the model's robustness and prevent overfitting.

To adapt the VGG19 model for this classification task, fully connected layers are appended to the base model. These included two dense followed by BatchNormalization and Dropout layers for overfitting. The final output layer consisted of 84 units, corresponding to the number of classes, with a SoftMax activation function.

During training process, a custom learning rate schedule is implemented for both phases. In the initial phase, the convolutional layers of the VGG19 model are frozen, and only the newly added fully connected layers are trained using the RMSprop optimiser. In the second phase, some of the deeper layers of the VGG19 model are unfrozen for fine-tuning, with a more aggressive learning rate schedule tailored for fine-tuning, starting with a very low learning rate of the initial 10 epochs. The model's performance is evaluated on the validation set using accuracy, precision, recall, and F1-score metrics. This comprehensive evaluation ensures that the developed model effectively captures the intricate features of Bangla handwritten characters and achieves high classification accuracy.

4.1. Dataset Description

The dataset "BanglaLekha-Isolated" comprises a collection of Bangla handwritten isolated character samples, encompassing 50 Bangla basic characters, 10 Bangla numerals, and 24 selected compound characters. Table 1 lists the classes of 84 Bangla basic letters, numerals, and compound characters.

Table 1. Classes of Bangla Characters for classification

Bangla basic characters									
অ [1]	আ [2]	ই [3]	ঈ [4]	উ [5]	ঊ [6]	ঋ [7]	এ [8]	ঐ [9]	ও [10]
ঔ [11]	ক [12]	খ [13]	গ [14]	ঘ [15]	ঙ [16]	চ [17]	ছ [18]	জ [19]	ঝ [20]
ঞ [21]	ট [22]	ঠ [23]	ড [24]	ঢ [25]	ণ [26]	ত [27]	থ [28]	দ [29]	ধ [30]
ন [31]	প [32]	ফ [33]	ব [34]	ভ [35]	ম [36]	য [37]	র [38]	ল [39]	শ [40]
ষ [41]	স [42]	হ [43]	ড় [44]	ঢ় [45]	য় [46]	ৎ [47]	ং [48]	ঃ [49]	ঁ [50]
Bangla numerals									
০ [51]	১ [52]	২ [53]	৩ [54]	৪ [55]	৫ [56]	৬ [57]	৭ [58]	৮ [59]	৯ [60]
Bangla compound characters									
ক্ষ [61]	ব্দ [62]	ঙ্গ [63]	স্ক [64]	স্ফ [65]	স্থ [66]	চ্ছ [67]	ক্ত [68]	ম্ন [69]	ঞ্চ [70]
ম্প [71]	ষ্মা [72]	প্ত [73]	ম্ব [74]	ন্ত [75]	দ্ভ [76]	থ্থ [77]	ষ্ঠ [78]	ল্ল [79]	ম্প [80]
ন্ধ [81]		ন্দ [82]		ম্ম [83]		ণ্ঠ [84]			

Each of the 84 characters has 2000 handwriting samples, totalling 166,105 handwritten character images after digitisation and preprocessing. Mistakes and scribbles are discarded during the preprocessing stage. The dataset includes information regarding the age and gender of the subjects from whom the handwriting samples are collected. Each individual image is mapped to this information in Figure 5. A separate spreadsheet assesses the aesthetic quality of the handwriting samples collected from three independent assessors. This assessment is done on groups of 84 characters, not individual characters.

Figure 5: A handwritten Bangla character written by different individuals

4.2. Data Preprocessing

Data Augmentation

Data augmentation is a method used to increase the size of a dataset without collecting additional data. For image data, various techniques, such as cropping, rotating, zooming, and shifting, are used to create new data from the existing dataset. In this work, a comprehensive set of transformations is employed to augment the dataset of Bangla handwritten characters, thereby bolstering the robustness and performance of the CNN-based classification model. The augmentation techniques implemented include width and height shifting,

rotation by 10 degrees, a shear range of 0.1, and a zooming range of 0.5. Each of these transformations serves a specific purpose in diversifying the dataset and simulating real-world scenarios. For instance, rotating the images allows the model to classify characters from different angles, while width and height shifting introduces variations in the characters' positions within the image. Shearing distorts the characters, simulating various writing styles, and zooming enhances the model's ability to identify characters at different scales. These augmentations help prevent overfitting and improve the model's generalisation of unseen data. Additionally, rescaling ensures that pixel values are standardised, making the training process more efficient. Overall, data augmentation plays a crucial role in training CNN-based models for character classification tasks, contributing to their accuracy and reliability.

After augmentation, the dataset size remains unchanged, as augmentation is primarily aimed at enhancing diversity and robustness rather than increasing the number of images. For instance, the dataset initially contains 166,105 images. The image dimensions remain consistent, typically resized to a fixed size, such as 32x32 pixels, to maintain uniformity. Pixel intensity after pixel value standardisation is adjusted to ensure consistency across all images, typically ranging from 0 to 1.

4.3. Model Architecture

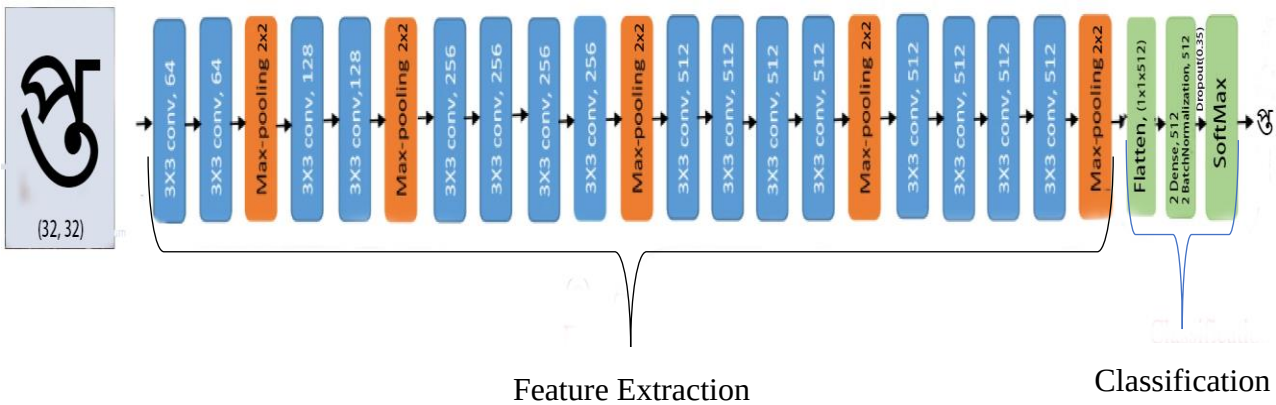

Figure 6: Modified VGG19 architecture

The model architecture depicted in Figure 6, represents a modified version of the VGG19 convolutional neural network (CNN) tailored for Bangla handwritten character classification. The model consists of multiple convolutional layers followed by max-pooling layers to extract hierarchical features from input

images. These layers are responsible for capturing intricate patterns and structures present in the handwritten characters.

Following the convolutional layers, two dense layers are appended to the model to perform classification based on the extracted features. Each dense layer allows the model to learn complex representations of the input data. BatchNormalization layers are inserted after the dense layers to improve the training stability and accelerate convergence. Dropout layers are included in the model as a form of regularization to mitigate overfitting.

The dense layer contains 84 units, corresponding to the number of classes in the BanglaLekha-Isolated dataset, with a SoftMax activation function. This layer computes the probability distribution over the classes, enabling the model to make predictions. Throughout the training process, these parameters are optimized to minimize classification errors and enhance model performance.

Overall, the modified VGG19 model is implemented to effectively capture the distinctive features of Bangla handwritten characters and facilitate accurate classification. Through fine-tuning and training on the BanglaLekha-Isolated dataset, the model aims to achieve high accuracy and robustness in character recognition tasks.

5. Experiments and Results

In this work, Python (version 3.12.3) served as the primary programming language, supported by a selection of indispensable libraries, including Numpy, Pandas, Seaborn, Matplotlib, and Math. Given the dataset's substantial size of 166,105 images, Google Colaboratory is utilised for its rapid processing capabilities. The open-source libraries TensorFlow and Keras are used to implement the VGG19 model.

The dataset comprises Bangla handwritten characters, partitioned into training and validation sets with an 80:20 split. In the training set, there are 132,914 images spread across 84 classes, while the validation set contains 33,191 images from the same classes. The classification model adopts the VGG19 architecture, initially pre-trained on ImageNet, featuring 16 convolutional layers followed by 5 max-pooling layers. Additionally, the model is augmented with two dense layers, each containing 512 units.

To combat overfitting, the model incorporates batch normalization and dropout layers, with a dropout rate set to 0.35. The batch size is configured to 128 to optimize training efficiency. In third dense layer encompasses 84 classes from the BanglaLekha-Isolated dataset, employing a SoftMax activation function to compute the probability distribution over the classes. In the first phase, the model training continues for 10 epochs, maintaining a constant learning rate of 0.0001 for the first 5 epochs. Subsequently, the learning rate is reduced to 0.00005 for the remaining epochs.

Following the initial phase, fine-tuning extends over 10 epochs, accompanied by a learning rate scheduler. The scheduler initiates with a minimal learning rate of 0.0000001, gradually escalating to 0.000005 until the 45th epoch, and then declining to 0.000001 thereafter. The RMSprop optimizer oversees the optimization of model parameters throughout the training process. These meticulously orchestrated configurations, coupled with dataset augmentation and fine-tuning strategies, synergistically fortify the model's resilience and efficacy in accurately discerning Bangla handwritten characters. Table 2 enumerates all the parameters required to establish proposed model for classifying Handwritten Bangla alphabets, numerals, and compound characters.

Table 2. Internal parameter of proposed VGG19

Type of layer	Output Shape	Parameters
input_1 (InputLayer)	(None, height=32, width=32, filter size=3)	0
block1_conv1 (Conv2D)	(None, height=32, width=32, filter size =64)	1792
block1_conv2 (Conv2D)	(None, height=32, width=32, filter size =64)	36928
block1_pool (MaxPooling2D)	(None, height=16, width=16, filter size =64)	0
block2_conv1 (Conv2D)	(None, height=16, width=16, filter size =128)	73856
block2_conv2 (Conv2D)	(None, height=16, width=16, filter size =128)	147584
block2_pool (MaxPooling2D)	(None, height=8, width=8, filter size =128)	0
block3_conv1 (Conv2D)	(None, height=8, width=8, filter size =256)	295168
block3_conv2 (Conv2D)	(None, height=8, width=8, filter size =256)	590080
block3_conv3 (Conv2D)	(None, height=8, width=8, filter size =256)	590080
block3_conv4 (Conv2D)	(None, height=8, width=8, filter size =256)	590080
block3_pool (MaxPooling2D)	(None, height=4, width=4, filter size =256)	1180160
block4_conv2 (Conv2D)	(None, height=4, width=4, filter size =512)	2359808
block4_conv3 (Conv2D)	(None, height=4, width=4, filter size =512)	2359808
block4_conv4 (Conv2D)	(None, height=4, width=4, filter size =512)	2359808
block4_pool (MaxPooling2D)	(None, height=2, width=2, filter size =512)	0
block5_conv1 (Conv2D)	(None, height=2, width=2, filter size =512)	2359808
block5_conv2 (Conv2D)	(None, height=2, width=2, filter size =512)	2359808
block5_conv3 (Conv2D)	(None, height=2, width=2, filter size =512)	2359808
block5_conv4 (Conv2D)	(None, height=2, width=2, filter size =512)	2359808
block5_pool (MaxPooling2D	(None, height=1, width=1, filter size =512)	0
flatten (Flatten)	(None, filter size =512)	0
dense (Dense)	(None, filter size =512)	262656
batch_normalization (Batch Normalization)	(None, filter size =512)	2048
dropout (Dropout)	(None, filter size =512)	0
dense_1 (Dense)	(None, filter size =512)	262656
batch_normalization_1 (Batch Normalization)	(None, filter size =512)	2048
dropout_1 (Dropout)	(None, filter size =512)	0
dense_2 (Dense)	(None, filter size =84)	43092

5.1. Accuracy and Loss Curves

Figure 7 depicts the training and validation accuracy for different epochs. It shows whether the model is overfitting or underfitting. The modified VGG19 model had a training and validation accuracy of 96.63% and 96.79% respectively. Figure 8 shows the training vs. validation loss. The loss curve indicates that both losses are consistently minimal and decreasing, indicating that the models fit well with the data.

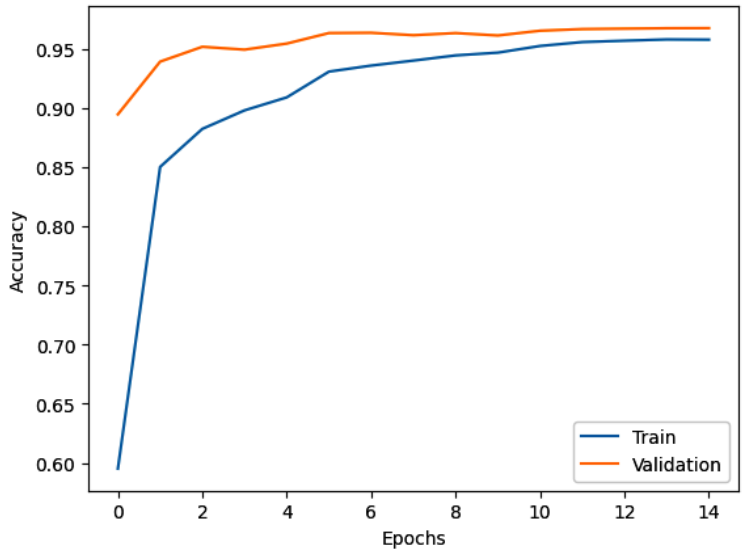

Figure 7: Accuracy Curve

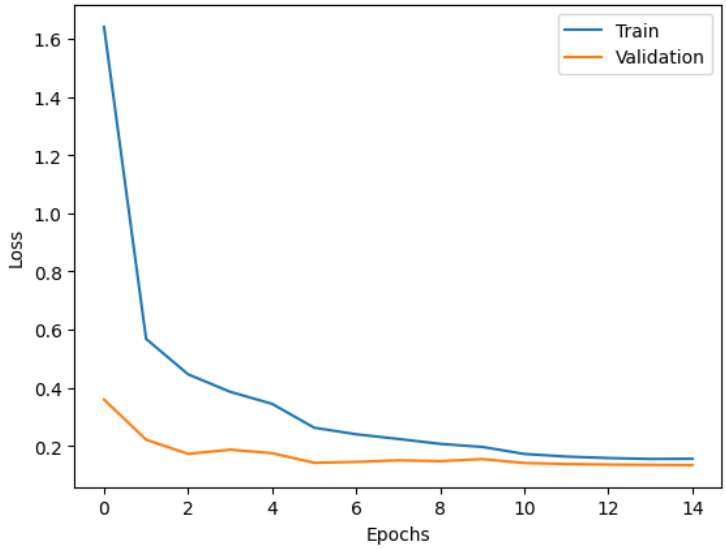

Figure 8: Loss Curve

In evaluating the performance of the model, several key metrics are utilised. The model achieves an validation accuracy of 96.79%, indicative of its overall effectiveness. The modified VGG19 model achieved a high precision of 96.84% which signifies a low rate of false positives, bolstering the model's reliability. Recall, on the other hand, evaluates the model's ability to identify all relevant instances within the dataset, resulting in a recall rate of 96.79%. The F1-Score, which balances precision and recall, yields a score of 96.79%, underscoring the model's robust performance across both dimensions.

6. Comparative Analysis

Table 3 presents a comprehensive evaluation of several deep learning models employed for Bangla handwritten character classification. This evaluation encompasses models such as Convolutional Neural Networks (CNN), modified quadratic discriminant function (MQDF), and ResNet-18, previously proposed by other researchers. Alongside these established models, the study also assesses the modified VGG19 model, providing a detailed comparison regarding accuracy and various limitations. The works aims to identify which model performs best under specific conditions by presenting its accuracy rates.

The comparative analysis of various models for Bangla handwritten character classification reveals a range of validation accuracies and inferential insights across different architectures. It is observed that modified VGG19 model achieved a validation accuracy of 96.79% with 84 classes, outperforming a CNN model [8] which had a validation accuracy of 85.36% with high iterations. Additionally, other existing methods [17, 18, 19, 12, 20, 21] achieved validation accuracies of 85.90%, 89.30%, 93.2%, 95.99%, 96.40%, and 96.40%, respectively. These methods, however, face limitations such as character constraints, overfitting issues, high complexity, and a large number of weights, which contribute to their lower accuracies. In contrast, the modified VGG19 model demonstrates superior accuracy, addressing these challenges more effectively.

Table 3. Comparison of the modified VGG19 model with other models

Literature	**Models**	**Classes**	**Validation Accuracy**	**Inference**
Rahman et al. [8]	CNN	50	85.36%	Low accuracy, requires high iterations
Pal et al. [17]	MQDF	110	85.90%	Limited to compound characters, generalisation performance is not encouraging
R. Jadhav et al. [18]	CNN	84	89.30%	Overfitting when iterations increase
Hossain et al. [19]	CNN	60	93.2%	Overfitting and underfitting
Alif et al. [12]	ResNet-18	84	95.99%	High complexity, a large number of weights, not

				suitable for this specific research domain
Saha et al. [20]	BBCNet-15	50	96.40%	6 layers of convolution, 6 layers of pooling, and 2 dense layers; more experimental results needed
Roy [21]	ResNet50	84	96.40%	12 convolutional layers, 4 pooling layers, and 5 fully connected layers require high-capacity devices and longer processing time
Modified VGG19 Model	VGG19	84	96.79%	20 convolutional layers, 5 pooling layers, and 4 fully connected layers outperforms existing methods with respect to fewer epochs

7. Conclusion and Future Scope

The evaluation presented in this work highlights the efficacy of various deep learning models for Bangla handwritten character classification, focusing on VGG19, which achieved an impressive validation accuracy of 96.79%. The performance of VGG19 can be attributed to its deep architecture, effective convolutional layers, and robust training methodologies, including data augmentation and transfer learning. This work not only underscores the potential of advanced deep learning models like VGG19 in achieving high accuracy but also provides a detailed comparison with other models such as CNN, MQDF, and ResNet-18. The experimental findings demonstrate that while traditional models have their strengths, the deep learning approach, particularly with architectures like VGG19, significantly improves accuracy and reliability for Bangla handwritten character classification.

In the future, this work aims to build a larger dataset for Bangla handwritten digits and characters, as having more examples enhances the model's learning capability.

Reference

[1] A. Taufique, F. Rahman, I.K. Pranta, N.A. Zahid and S.S. Hasan, "Handwritten Bangla Character Recognition using Inception Convolutional Neural Network," International Journal of Computer Applications, vol. 181 – no. 17, 2018.

[2] C.J. Kumar, G. Singh, R. Rani and R. Dhir, "Handwritten Segmentation in Bangla Script: A Review of Offline Techniques," International Journal of Advanced Research in Computer Science and Software Engineering, vol. 3, Issue 1, 2013, pp. 135-140.

[3] S.M.A. Hakim and Asaduzzaman, "Handwritten Bangla numeral and basic character recognition using deep convolutional neural network," International Conference on Electrical, Computer and Communication Engineering (ECCE), 2019, pp. 1–6. Available: https://doi.org/10.1109/ECACE.2019.8679243

[4] S. Islam, S.I.A. Khan, M.M. Abedin, K.M. Habibullah and A.K. Das, "Bird Species Classification from an Image Using VGG-16 Network," Proceedings of the 2019 7th International Conference on Computer and Communications Management (ICCCM 2019), 2019, pp. 38-42. Available: https://doi.org/10.1145/3348445.3348480

[5] M. Bansal, M. Kumar, M. Sachdeva and A. Mittal, "Transfer learning for image classification using VGG19: Caltech-101 image data set", Journal of Ambient Intelligence and Humanized Computing and Springer Nature 2021, vol. 14, 2021, pp. 3609-3620. Available: https://doi.org/10.1007/s12652-021-03488-z

[6] A. Narayan and R. Muthalagu, "Image Character Recognition using Convolutional Neural Networks," Seventh International conference on Bio Signals, Images, and Instrumentation (ICBSII), IEEE, 2021. Available: https://ieeexplore.ieee.org/abstract/document/9445136

[7] A.K.M.S.A. Rabby, M. Islam, N. Hasan, J. Nahar and F. Rahman, "Borno: Bangla handwritten character recognition using a multiclass convolutional neural network," Proceedings of the Future Technologies Conference (FTC), vol. 1, 2020, pp. 457-472.

[8] M. Rahman, M.A.H. Akhand, S. Islam, P.C. Shill and M.M.H. Rahman, "Bangla Handwritten Character Recognition using Convolutional Neural Network,"

International Journal of Image, Graphics and Signal Processing, vol. 7, issue 8, 2015, pp. 42 – 49. Available: https://www.scinapse.io/papers/755956977

[9] B. Purkaystha, T. Datta and S. Islam, “Bangla handwritten character recognition using deep convolutional neural network,” 20th IEEE International Conference of Computer and Information Technology 2017 (ICCIT '17), 2017, pp. 01-05. Available: https://ieeexplore.ieee.org/abstract/document/8281853

[10] R.R. Chowdhury, M.S. Hossain, R. Islam, K. Andersson and S. Hossain, "Handwritten Character Recognition using Convolutional Neural Network with Data Augmentation," 2019 Joint 8th International Conference on Informatics, Electronics & Vision (ICIEV), 2019, pp. 318-323.

[11] S.H Shaikh, M. Tabedzki, N. Chaki and K. Saeed, “Bangla Printed Character Recognition – A New Approach,” Computer Information Systems and Industrial Management, 2013, pp. 129-140. Available: http://dx.doi.org/10.1007/978-3-642-40925-7_13

[12] M.A.R Alif, S. Ahmed and M.A. Hasan, “Isolated Bangla handwritten character recognition with convolutional neural network,” 20th IEEE International Conference of Computer and Information Technology 2017 (ICCIT '17), 2017, pp. 01-06. Available: https://ieeexplore.ieee.org/abstract/document/8281823

[13] S. Basu, R. Sarkar, N. Das, M. Kundu, M. Nasipuri and D.K. Basu “A fuzzy technique for segmentation of handwritten bangla word images,” International Conference on Computing: Theory and Applications (ICCTA’07), IEEE, 2007, pp. 427–433.

[14] R. Pramanik and S. Bag, “Segmentation-based recognition system for handwritten bangla and devanagari words using conventional classification and transfer learning,” IET Image Processing, 2020. Available: https://doi.org/10.1049/iet-ipr.2019.0208

[15] A. El-Sawy, M. Loey and H. El-Bakry, “Arabic handwritten characters recognition using convolutional neural network,” WSEAS Transactions on Computer Research, vol. 5, 2017, pp. 11-19.

[16] X. Yang, X. Ren, Y. Zhou, Z. Huang, J. Sun, and K. Chen, “A Novel Text Structure Feature Extractor for Chinese Scene Text Detection and Recognition,” IEEE, vol. 5, 2017, pp. 3193–3204.

[17] U. Pal, T. Wakabayashi and F. Kimura, “Handwritten Bangla Compound Character Recognition Using Gradient Feature,” 10th IEEE International Conference on Information Technology 2007, 2007, pp. 208-213. Available: https://ieeexplore.ieee.org/abstract/document/4418297

[18] R. Jadhav, S. Gadge, K. Kharde, S. Bhere and I. Dokare, “Recognition of handwritten Bangla characters using low cost convolutional neural network,” 2022 Interdisciplinary Research in Technology and Management (IRTM), 2022. Available: https://ieeexplore.ieee.org/abstract/document/9791802

[19] A. Hossain, M.A.F.M.R. Hasan, A.F.M.Z. Abadin and N. Fatta, “Bangla Handwritten Characters Recognition Using Convolutional Neural Network,” Australian Journal of Engineering and Innovative Technology, vol. 4, issue 2, 2022, pp. 27–31. Available: https://universepg.com/journal-details/317

[20] C. Saha, R.H. Faisal and M. Rahman, “Bangla Handwritten Basic Character Recognition Using Deep Convolutional Neural Network,” 8th IEEE International Conference on Informatics, Electronics & Vision (ICIEV '19), 2019, pp. 190-195. Available: https://ieeexplore.ieee.org/abstract/document/8858575

[21] A. Roy, “AKHCRNet: Bangla handwritten character recognition using deep learning,” Computing Research Repository, 2021. Available: https://dblp.org/rec/journals/corr/abs-2008-12995

Appendix

Code

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
!unzip /content/drive/MyDrive/Images.zip
```

```
Streaming output truncated to the last 5000 lines.
inflating: Images/83/02_0002_0_19_1016_1099_83.png
inflating: Images/83/02_0002_0_19_1016_1100_83.png
inflating: Images/83/02_0002_0_19_1016_1132_83.png
inflating: Images/83/02_0002_0_19_1016_1133_83.png
inflating: Images/83/02_0002_0_19_1016_1143_83.png
inflating: Images/83/02_0002_0_19_1016_1144_83.png
```

```
import numpy as np
import random
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.preprocessing.image import
ImageDataGenerator
from tensorflow.keras.layers import Input, Dropout, Flatten,
Dense, BatchNormalization, concatenate
from tensorflow.keras.applications import VGG19
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.callbacks import LearningRateScheduler
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix,
ConfusionMatrixDisplay
from sklearn.metrics import f1_score, precision_score,
recall_score, accuracy_score

%matplotlib inline
```

```
img =
tf.keras.preprocessing.image.load_img('Images/9/02_0002_1_27_1
016_1171_9.png')
img
```

```
# Dataset details
train_data_dir = "Images"
img_width, img_height = 32, 32
input_shape = (img_width, img_height, 3)
batch_size = 128
```

```
# ImageDataGenerator for loading and scaling data
train_datagen = ImageDataGenerator(
    rescale=1.0/255,
    rotation_range=10,
    width_shift_range=0.05,
    height_shift_range=0.05,
    shear_range=0.1,
    zoom_range=0.05,
    fill_mode='constant',
    cval=0,
    validation_split=0.20
)

valid_datagen = ImageDataGenerator(rescale=1.0/255,
validation_split=0.2)
```

```
train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    color_mode="rgb",
    class_mode='categorical',
    subset='training',
    shuffle=True,
    seed=13
)

validation_generator = valid_datagen.flow_from_directory(
    train_data_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    color_mode="rgb",
    subset='validation',
    shuffle=False,
    class_mode='categorical',
    seed=13
)
```

```
Found 132914 images belonging to 84 classes.
Found 33191 images belonging to 84 classes.
```

```
classes = len(train_generator.class_indices)
print("Number of classes:", classes)
```

```
Number of classes: 84
```

```
# Fine-tuning the model by unfreezing some layers
for layer in base_model.layers[:28]:
    layer.trainable = True

# Recompile the model
model.compile(loss='categorical_crossentropy',
optimizer=RMSprop(), metrics=['accuracy'])
# Training the model with the second learning rate scheduler
callback2 =
LearningRateScheduler(decayed_learning_rate_tuned50)
history2 = model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // batch_size,
    epochs=10,
    callbacks=[callback2],
    validation_data=validation_generator,
    validation_steps=validation_generator.samples //
batch_size
)
```

```
Epoch 1/10
1038/1038 [==============================] - 173s 162ms/step - loss: 0.1691
- accuracy: 0.9536 - val_loss:0.1445 - val_accuracy: 0.9639 - lr:1.0000e-07

Epoch 2/10
1038/1038 [==============================] - 170s 163ms/step - loss: 0.1616
- accuracy: 0.9558 - val_loss:0.1415 - val_accuracy: 0.9649 - lr:1.0000e-07

Epoch 3/10
1038/1038 [==============================] - 176s 169ms/step - loss: 0.1565
- accuracy: 0.9569 - val_loss:0.1396 - val_accuracy: 0.9654 - lr:1.0000e-07

Epoch 4/10
1038/1038 [==============================] - 172s 165ms/step - loss: 0.1536
- accuracy: 0.9579 - val_loss:0.1384 - val_accuracy: 0.9658 - lr:1.0000e-07

Epoch 5/10
1038/1038 [==============================] - 181s 175ms/step - loss: 0.1536
- accuracy: 0.9584 - val_loss:0.1373 - val_accuracy: 0.9662 - lr:1.0000e-07

Epoch 6/10
1038/1038 [==============================] - 176s 169ms/step - loss: 0.1437
- accuracy: 0.9609 - val_loss:0.1312 - val_accuracy: 0.9679 - lr:5.0000e-06

Epoch 7/10
1038/1038 [==============================] - 175s 168ms/step - loss: 0.1356
- accuracy: 0.9629 - val_loss:0.1326 - val_accuracy: 0.9677 - lr:5.0000e-06

Epoch 8/10
1038/1038 [==============================] - 174s 168ms/step - loss: 0.1326
- accuracy: 0.9646 - val_loss:0.1330 - val_accuracy: 0.9677 - lr:5.0000e-06
```

```
Epoch 9/10
1038/1038 [==============================] - 178s 172ms/step - loss: 0.1275
- accuracy: 0.9652 - val_loss:0.1335 - val_accuracy: 0.9679 - lr:5.0000e-06

Epoch 10/10
1038/1038 [==============================] - 174s 167ms/step - loss: 0.1230
- accuracy: 0.9663 - val_loss:0.1351 - val_accuracy: 0.9679 - lr:5.0000e-06
```

```
# calculate different metrics for HCR-Net on the test dataset
print('precision_score: ',
precision_score(validation_generator.classes,y_pred,
average="macro"))
print('recall_score   : ',
recall_score(validation_generator.classes,y_pred,
average="macro"))
print('f1_score       : ',
f1_score(validation_generator.classes,y_pred,
average="macro"))
print('accuracy_score : ',
accuracy_score(validation_generator.classes,y_pred,
normalize=True))

print("\nBest accuracy  : ",
max(max(history1.history['val_accuracy']),
max(history2.history['val_accuracy'])))
```

```
precision_score:  0.9684424796559218
recall_score   :  0.9679070369183158
f1_score       :  0.967870358494676
accuracy_score :  0.96794311711006
Best accuracy  :  0.9679355621337891
```