

Thesis No. : CSER-22-15

HANDWRITTEN NUMERAL RECOGNITION USING TRANSFER LEARNING TECHNIQUE

By

Kazi Ziaul Hassan

Roll No: 1607061

&

Md. Golam Kaochhar

Roll No: 1607100



Department of Computer Science and Engineering

Khulna University of Engineering & Technology

Khulna 9203, Bangladesh

March, 2022

HANDWRITTEN NUMERAL RECOGNITION USING TRANSFER LEARNING TECHNIQUE

By

Kazi Ziaul Hassan

Roll No: 1607061

&

Md. Golam Kaochhar

Roll No: 1607100

A thesis submitted in partial fulfillment of the requirements for the degree of
“Bachelor of Science in Computer Science and Engineering”

Supervisor:

Mohammad Insanur Rahman Shuvo

Assistant Professor

Signature

Department of Computer Science and Engineering

Khulna University of Engineering & Technology Khulna, Bangladesh

Department of Computer Science and Engineering

Khulna University of Engineering & Technology

Khulna 9203, Bangladesh

March, 2022

Acknowledgement

At first, we would like to thank Almighty God for showering all his blessings on us whenever we needed. It is our great pleasure to express our indebtedness and deep sense of gratitude to our Supervisor Mohammad Insanur Rahman Shuvo, Assistant Professor, Dept. of Computer Science and Engineering (CSE), Khulna University of Engineering & Technology (KUET) for his continuous encouragement, constant guidance, great and enthusiastic support, both during the implementation phase and the writing process and keen supervision throughout the course of this study. His prudence and ethics have been a great inspiration for the work. We also like to remember the inspiration, supports and encouragement of our loving family.

We are extremely grateful to all the faculty members of the Dept. of CSE, KUET to have their privilege of intensive, in-depth interaction and suggestions for the successful completion of our Bachelor of Science degree.

March, 2022

Kazi Ziaul Hassan & Md. Golam Kaochhar

Abstract

Numerals are an important aspect of every language, and individuals routinely write numerals by hand for a variety of reasons, including bank checks, postal codes, and so on. Handwritten numeral recognition (HNR) has significant challenges since any numeral combination is technically valid; HNR is a sensitive task and system must be absolutely accurate for real-life use. Technically, HNR is a process which considers images of handwritten numerals as input and classifies the images into different numeral categories. Challenges of HNR are language dependent based on shape, similarity and other complexities in the numeral sets of languages. Convolutional neural networks (CNN) are used as a classifier to recognize handwritten numerals for classification purposes. For Bengali and Devanagari, benchmark HNI datasets developed by the Computer Vision and Pattern Recognition (CVPR) Unit of the Indian Statistical Institute (ISI) [18] are utilized to verify the proposed system's performance. For Bengali and Devanagari numerals, the CVPR, ISI dataset offers a significant number of training and test samples. In this work, preprocessing is done on 18000 ($=1800 \times 10$) training set images and utilized in training in both Bengali and Devanagari. The test sets for Bengali and Devanagari numerals are 4000 and 3750 images, respectively. Using the transfer learning method and CNN as a classifier, the proposed system misclassified only 11 test samples out of 4000 test sample, achieving 99.73 percent recognition accuracy for Bengali numeral and 14 test samples out of 3750 test sample, achieving 99.63 percent recognition accuracy for Devanagari numeral.

Contents

	PAGE
Title Page	i
Acknowledgement	ii
Abstract	iii
Contents	iv
List of Tables	vi
List of Figures	vii
Nomenclature	ix
 CHAPTER I	
Introduction	1
1.1 Overview of Handwritten Numeral Recognition (HNR)	1
1.2 Overview of Transfer Learning Technique	2
1.3 Motivation	3
1.4 Literature Review	4
1.4.1 HNR using Traditional Methods	4
1.4.2 HNR using Deep Learning Method	4
1.4.3 Previous Works from the Department of CSE, KUET	5
1.5 Observations on Existing Methods	5
1.6 Objectives of the Thesis	6
1.7 Organization of the Thesis	6
 CHAPTER II	
Background	8
2.1 Basic steps of HNR	8
2.1.1 Pre-Processing	8
2.1.2 Feature Extraction	9
2.1.3 Classification	10
2.2 Basic Structure of CNN	10
2.3 Challenges of HNR	14
2.4 Scope of the Research	15

CHAPTER III	Handwritten Numeral Recognition Using Transfer Learning Technique	16
	3.1 Basic Structure of the Proposed HNR System	16
	3.2 Basic Structure of the Overall Systems	17
	3.3 Basic Structure of the Existing Models	18
	3.4 Transfer Learning Strategies	23
	3.5 Development of the Proposed Transfer Learning Technique	23
	3.6 Significance of the Proposed System	29
CHAPTER IV	Experimental Studies	31
	4.1 Description about Pre-Trained Models Dataset	31
	4.2 Benchmark Dataset and Pre-Processing	32
	4.3 Experimental Setup	33
	4.4 Experimental Results and Analysis	34
	4.5 Comparison with the Existing Methods	41
CHAPTER V	Conclusions	44
	5.1 Achievements	44
	5.2 Future Perspectives	45
REFERENCES		46

List of Tables

Table No.	Description	Page
3.1	Standard structure of resnet50 model.	22
3.2	Partial summary of pre-trained model vgg16.	24
3.3	Partial summary of pre-trained model vgg19.	26
3.4	Training parameter with transfer learning approaches for vgg16	27
3.5	Training parameter with transfer learning approaches for vgg19	28
4.1	Pre-trained model testing accuracies	35
4.2	Testing accuracies for CVPR Bengali dataset with batch size 64	35
4.3	Testing accuracies for CVPR Devanagari dataset with batch size 64	36
4.4	Testing accuracies for CVPR Bengali dataset with batch size 32	36
4.5	Testing accuracies for CVPR Devanagari dataset with batch size 32	36
4.6	Confusion matrix for test samples of handwritten Bengali numeral dataset	39
4.7	Confusion matrix for test samples of handwritten Devanagari numeral dataset	40
4.8	Sample misclassified handwritten Bengali and Devanagari numerals by HNRUTL.	41
4.9	A comparative description of proposed HNR with some prominent methods for Bengali handwritten numerals.	42
4.10	A comparative description of proposed HNR with some prominent methods for Devanagari handwritten numerals.	43

List of Figures

Figure No.	Caption of the Figure	Page
2.1	Block diagram of a typical HNR system.	8
2.2	Basic structure of Convolutional Neural Network (CNN).	11
2.3	Convolved kernel/filter on input image.	12
2.4	Applying max-pooling	13
2.5	Flattening and fully connected layer	13
2.6	Sample handwritten numeral images to understand the challenges of HNR: (a) Similar shaped numerals; and (b) Same numeral look very different in size, shape, and orientation	14
3.1	Structure of the proposed handwritten numeral recognition using transfer learning.	16
3.2	An illustration of how all the systems has been categorized.	18
3.3	Vgg16 model block diagram	18
3.4	Vgg19 model block diagram	19
3.5	Residual block	20
3.6	Identity block and convolution block	21
3.7	Pre-trained model structure with vgg16	24
3.8	Pre-trained model structure with vgg19	25
3.9	Pre-trained model structure with resnet50	26
3.10	Proposed method structure for vgg16, with and without fine tuning	27
3.11	Proposed method structure for vgg19, with and without fine tuning	28
3.12	Proposed method structure for resnet50, with and without fine tuning	29
4.1	Samples of handwritten English numerals from MNIST dataset.	31
4.2	Samples of handwritten Bengali numerals from CVPR, ISI dataset.	34
4.3	Samples of handwritten Devanagari numerals from CVPR, ISI dataset.	34

4.4	Comparison among vgg16, vgg19 and resnet50 for the Bengali CVPR, batch size 32	37
4.5	Comparison among vgg16, vgg19 and resnet50 for the Devanagari, CVPR, batch size 32	38
4.6	Comparison among vgg16, vgg19 and resnet50 for the Bengali, CVPR, batch size 64	38
4.7	Comparison among vgg16, vgg19 and resnet50 for the Devanagari, CVPR, batch size 64	39

Nomenclature

CNN	Convolutional Neural Network
DNN	Deep Neural Network
FN	Fine Tuning
HNR	Handwritten Numeral recognition
HNRUTL	Handwritten Numeral recognition using transfer learning
MNIST	Modified National Institute of Standards and Technology
SVM	Support Vector Machine
CVPR	Computer Vision and Pattern Recognition
ANN	Artificial Neural Network
ISI	Indian Statistical Institute
BS	Batch size
TL	Transfer learning
DNN	Deep Neural Network

CHAPTER I

Introduction

Handwritten numeral recognition (HNR) is a procedure that takes handwritten numerical images as input and categorizes them into different numeral categories. Handwritten numbers can be found on bank checks, postal codes, and a variety of other items. Automatic recognition of handwritten digits has become a hot research area in this digital age. HNR is traditionally thought of as a pattern recognition problem; therefore getting high accuracy in HNR is difficult. In practice, very few people train a full Convolutional Network from start because a sufficiently large dataset is uncommon. Instead, it is typical to pre-train a ConvNet on a big dataset (for example, ImageNet, which has 1.2 million images with 1000 categories) and then utilize the ConvNet as an initialization or fixed feature extractor for the job at hand. Transfer learning can be defined as the concept of overcoming the isolated learning paradigm and applying knowledge obtained for one task to solve related problems. Here, we used multiple models such as VGG16, VGG19, and RESNET50 with minimal modifications that are required for dataset to create our pre-trained model with handwritten English numeral MNIST dataset. Then, using a transfer learning strategy, we were able to distinguish CVPR Bengali and Devanagari numerals from 0 to 10 using CVPR Bengali and Devanagari numerals. These two ways to transfer learning are employed here: without fine tuning and with fine tuning. Finally, we analyze and contrast existing approaches. When comparing the suggested system's recognition accuracy value to that of other well-known methods for recognizing Bengali handwritten numerals, it is clear that the proposed system beats all other systems and demonstrates its superiority.

1.1 Overview of Handwritten Numeral Recognition (HNR)

Bengali numbers are an essential aspect of the Bengali language and play a significant role in daily life. Bengali numerals are now utilized for a variety of reasons in everyday life, both printed and handwritten. Handwritten numeral recognition (HNR) is a procedure that takes handwritten numerical images as input and categorizes them into different numeral categories. Unlike letter or word recognition, handwritten numerical recognition poses

significant challenges. Recognition errors caused by a system can be validated using grammar rules with in instance of character or word recognition. However, because any numeral combination is mathematically legal, such an error detection rule or process is not feasible for numeral recognition. As a result, recognizing Bengali handwritten numerals are a complex method, and the system must be 100% accurate for real-world use, as Bengali is one of the world's languages with the most complicated numeral patterns.

1.2 Overview of Transfer Learning Technique

Humans are born with the ability to transfer knowledge from one work to another. What we learn as knowledge while learning about one task, we apply to other tasks in the same way. The easier it is for us to transfer or cross-utilize our knowledge, the more correlated the jobs are. For example, if you can ride a motorbike, it will be much easier for you to learn to drive a car. When we try to learn new things in the examples above, we don't have to start from scratch. We use what we've learnt in the past to transfer and utilize our expertise! Transfer learning is the concept of breaking free from the isolated learning paradigm and applying what you've learned to solve related problems. Transfer learning is the process of taking a model that has already been trained in one field and applying it to another. Starting from scratch to train a model is time-consuming and requires a large amount of data. When your dataset contains insufficient data to train a full-scale model from scratch, transfer learning is typically used. The Transfer Learning process is as follows: (a) Take layers from a previously learned model. (b) Freeze them to prevent any of the knowledge they contain from being lost during subsequent training rounds. (c) On top of the frozen layers, add some new, trainable layers. They'll figure out how to turn existing features into predictions on a new dataset, then (d) train the new layers with your dataset. Transfer learning is useful when you have insufficient data for a new domain you want handled by a neural network and there is a large pre-existing data pool that can be transferred to your problem. So, even if you just have 1,000 images of horses, you can obtain a lot of low-level and mid-level feature definitions by tapping into an existing CNN, such as ResNet or ImageNet that has been trained with over 1 million images. Transfer learning can save us training time, reduce the risk of over-fitting because we aren't starting from scratch, and give us a new means of applying past recognition expertise to something of a similar type. We'll need a lot of training data to construct a

new algorithm that can recognize a frown. Our model will first need to learn how to recognize faces, and only then will it be able to recognize expressions like frowns. Instead, we may get the same result with significantly less data if we take a model that has already learnt to detect faces and retrain it to detect frowns. Transfer learning prepares a model to perform well with data it hasn't been trained on, i.e., models generalize better.

1.3 Motivation

Handwritten Numeral Recognition technique is a complex and convoluted task. If it is a talk about dimension then it can be said that it deals with the tasks that are of higher dimension. Different people write in different ways. They do not use the same pattern while writing. While using the classifier to classify the numerals that people generally write in different ways, complexity arises. When a numeral is seen from a different angle, distortion occurs. Distortion happens when the pattern of an image is rotated. A novel Handwritten Numeral Recognition system is introduced in this thesis that follows an underlying hypothesis. The hypothesis is how humans learn writing. In this chapter, a conclusion will be drawn like a short passage of some starred points of this work and some directions on the basis of the outcome of the related works will be mentioned for how to do future research. Despite its high challenges, a notable number of researches have been conducted in recent years that showed satisfactory recognition accuracy while working over Roman, Chinese, Arabian [19] , English [20], Indian and Bengali [20] [9]–[13] languages. K. Tushar et. al [1] introduced a convolutional neural network-based transfer learning technique for Bengali, Hindi, and Urdu numeric characters. Here, they considered numerals of one language as the source data and numerals of another language as the target data. This research outcome depicts that model pre-trained on one numeric character type can perform well on another type of numeric characters with minimal retraining. D. C. Ciresan et. al [3] proposed a transfer learning approach for Latin and Chinese character recognition with convolutional neural networks. In this study, they trained a CNN on Latin digits and transferred it for recognizing uppercase Latin letters. They conducted a similar analysis taking Chinese characters as source data and Latin characters as target data and vice-versa. Sakib reza et. al [4] proposed a new transfer learning approach for classifying Bengali handwritten characters using Ekush dataset [5]. They trained a CNN on Basic characters and transferred it for recognizing compound characters. Also conduct

experiment considering the digits as the source data and the basic characters as the target data.

1.4 Literature Review

Many researches on Handwritten Numeral Recognition have been done in various languages. Various prominent machine learning techniques are used where feature extraction is done before classification. On the contrary, various deep learning techniques are used where no feature extraction is used explicitly. Here below, how the existing studies solve the Handwritten Numeral Recognition problem with the help of the prominent machine learning techniques and deep learning techniques.

1.4.1 HNR Using Traditional Machine Learning Methods

HNR is a pattern recognition problem, whereas extraction of features is a frequent task in classical machine learning approaches. The images are then classified using various classification methods into various numerical categories. Belongie et al. [21] determined the correspondences between points from every two different shapes to calculate the aligning 3 transform and implemented NN classification algorithm for HNR. Kutzner et al. [23] presented a solution to verify user with safe handwritten password using Bayes Net, Naïve Bayes, K-NN and Multi-layer Perceptron (MLP) classifier. Lauer et al. [14] have proposed trainable feature extractors for handwritten digit recognition. Bhattacharya and Chaudhuri [24] proposed an MLP classifier based multistage cascaded architecture using majority voting method for recognition purpose. Nasir and Uddin [12] deployed handwritten Bengali numerals while deploying automated postal system. They build a hybrid system of bayes theorem, k-means clustering and 4 maximum a posteriori to extract features from handwritten numeral images and finally applied SVM for recognition. Khan et al. [25] deployed evolutionary technique while training an ANN for Bengali handwritten digits recognition.

1.4.2 HNR Using Deep Learning Methods

In recent years, different deep learning algorithms have become increasingly popular for HNR since they do not require any additional feature extraction procedures. Akhand et al. [9] proposed a Bengali and Bengali-English Mixed handwritten numeral recognition

scheme where the scanned input images were normalized and used to train CNN which was used for recognition without any feature extraction. Akhand et al. [10] proposed an ensemble system with three convolutional neural networks (CNN) where one CNN was trained with the original training set, the other two CNNs were trained with clockwise and anti-clockwise rotation based generated patterns. Finally, they applied the winner takes all selection mechanism to determine the final classification outcome. They also proposed another method where a single CNN was trained with original and clockwise and anti-clockwise rotation based generated patterns. Then, this single CNN was used to find the final classification outcome.

1.4.3 Previous Works from the Department of CSE, KUET

A Superposition of different shaped, sized and oriented handwritten numerals into same sized printed form is used in [2]. Auto-encoder (AE) and convolutional auto-encoder (CAE) are individually modified and investigated as superposition method to transform images of handwritten numeral into printed numeral. For the classification purpose, both artificial neural network (ANN) or convolutional neural network (CNN) are implied as classifier to recognize the printed numeral. An ensemble system with three convolutional neural networks (CNN) is used in [10] for Bengali handwritten numeral recognition (BHNR) where one CNN was trained with the original training set, the other two CNNs were trained with clockwise and anti-clockwise rotation based generated patterns. A single CNN is also trained with original and clockwise and anti-clockwise rotation based generated patterns for BHNR. Stacked auto-encoder (SAE) which incorporated printed numerals is used in [11] to recognize Bengali handwritten numerals. Sakib reza et. al [4] proposed a new transfer learning approach for classifying Bengali handwritten characters using Ekush dataset [5]. They trained a CNN on Basic characters and transferred it for recognizing compound characters. Also conduct experiment considering the digits as the source data and the basic characters as the target data.

1.5 Observation on Existing Methods

In recent years, a significant number of researchers have employed several classical machine learning approaches as well as deep learning methods to recognize handwritten digits. Using feature extraction techniques, optimal features are extracted from the input

images via traditional methods. The numerals are then classified using several classification methods based on the retrieved attributes. SVM [7], [11], NN algorithm [21], naive Bayes, random tree [17], MLP [22], ANN [25], and other classification algorithms are used to classify the images into different numerical categories. Different deep learning approaches, such as CNN [9], [10] and others, have been employed for HNR in recent years because they do not require any additional feature extraction algorithms. Both classical and deep learning algorithms do not display sufficient performance for similar shaped numerals, according to the existing approaches. Because recognition becomes difficult even for humans when a language contains closely shaped numerals, because the similarities turn out to be very close due to differences in people's writing patterns. Due to the diverse writing techniques of people, the same numeral might appear significantly different in size, shape, and orientation.

1.6 Objectives of the Thesis

The main objective of this work is to recognize Bengali handwritten numerals using the Transfer Learning Method. To achieve the goal, the study will be carried out with the following specific objectives:

- ◆ Study of existing classification techniques for handwritten numeral recognition of different languages.
- ◆ Constructing a pre-trained model, we used the VGG-16, VGG-19, and Resnet50 models with the necessary modifications on the MNIST dataset.
- ◆ Applying VGG-16, VGG-19 and Resnet50 methods on Benchmark CVPR, ISI dataset
- ◆ Performance comparison for HNR using traditional and transfer learning approaches to identify the effectiveness of the proposed system for recognizing handwritten.

1.7 Organization of the Thesis

The thesis has five chapters.

- Chapter I provides an introduction of the HNR problem and inspiration, as well as a description of existing HNR research.
- Chapter II explains the basic steps of HNR as well as challenges of HNR.
- Chapter III explains the proposed architecture for recognizing handwritten numeral in

detail. This chapter also provides an overview of the convolutional neural network (CNN) models used in the proposed HNR, as well as a description of the suggested system's significance. Also, include an explanation of the several models that were used here with minor changes.

- Chapter IV describes the dataset used for the experiments as well as pre-processing, experimental setup and finally reports the experimental results for HNR using VGG16, VGG19, and RESNET50.
- Chapter V is for the conclusions of this thesis along with the outline of future directions of research opened by this work.

CHAPTER II

Background

HNR (handwritten numeral recognition) is a difficult and time-consuming task. Because of its importance in real-world applications, a lot of study has spent into it. This chapter also covers the basic structure of the convolutional neural network (CNN) and the two alternative transfer learning procedures that are used with our suggested system as well as challenges of HNR. This chapter concludes with defining the scope of the study.

2.1 Basic Steps of HNR

Traditionally, HNR has been thought of as a pattern recognition problem; standard tasks include pre-processing images to make them equal sizes, extracting features, training a classifier with those retrieved features, and ultimately evaluating the system. The common phases of HNR are depicted in Figure 2.1, and they include data pre-processing, feature extraction, system training, and evaluation.

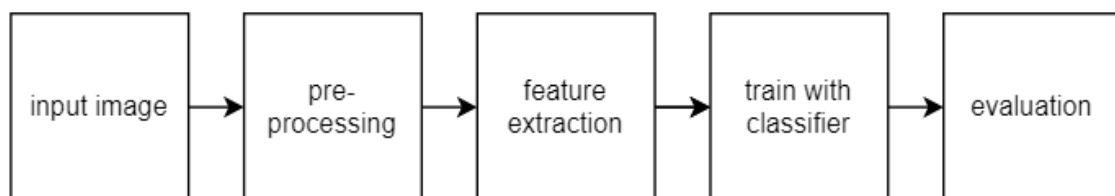


Figure 2.1: Block Diagram of a typical HNR system.

2.1.1 Pre-Processing

The goal of pre-processing is to improve the image's quality so that we can better analyze it. We can reduce unwanted distortions and boost some properties that are important for the application we're working on by preprocessing. Because the input images are typically acquired in a real-world situation using a variety of sensors, there is a lot of variance in image orientation, pixel spacing, and intensity scaling from person to person. All of these aspects are efficiently handled by pre-processing for better recognition. Depending on the application and quality of the data, a variety of strategies may be used in this step. For instance, the data might not have a uniform shape; in this case, the pre-processing step

would be to normalize the data into a uniform shape. Various data augmentation strategies can be used to artificially increase the size of a training dataset by modifying images in the collection. It involves transforming images in the training dataset into transformed copies that are of the same class as the original image. Shifts, flips, zooms, and other transformation operations from the field of image augmentation are included. There may be an outlier or noise in the raw data, in which case noise removal would be the pre-processing step. There are a variety of noise removal techniques available, including Gaussian filtering and smoothing with filters, and other.

2.1.2 Feature Extraction

Feature extraction is a formal process that specifies the key shape information included in a pattern in order to make the work of classifying the pattern easier. Feature extraction is a sort of dimensionality reduction in which a large number of pixels in an image are efficiently represented in such a way that the image's most interesting sections are effectively captured.

A proper feature extractor, which turns a physical sample to be recognized into a set of quantities defining the sample, selects the relevant characteristics. Statistical features and structural characteristics are the two types of features. Pixel density, moment, and mathematical transformation are examples of statistical features. Stroke, contour, number of bifurcation points, number of circles, and so on are structural features. Most studies believe that statistical characteristics may be produced fast using simple approaches and can produce high recognition results, especially in closed testing sets, but they can also be easily influenced by symbol deformation, limiting their application. Structural elements conform more to the intuitive thinking of the human mind, making them more difficult to symbol deformation. However, the recognition algorithm is frequently based on human-summarized principles. When new symbols are added to an application, the cost of revising the algorithm increases [15].

In HNR, feature extraction methods include principal component analysis (PCA), kernel PCA (KPCA) [7], Bayes theorem, k-means clustering and maximum a posteriori [12], convex hull [16], and others. For HNR, various types of characteristics are retrieved from handwritten numerical images. The distribution of dark pixels over image regions can be

used to characterize the shape of individual numerals. The pixel distance between the elements of the picture is counted and the edge of the number is found to characterize the external shape of the numerals. The handwritten images' width and length are crucial factors for recognition. Vertical and horizontal crossings, as well as three feature points, are considered important characteristics. Some important qualities that can be employed for handwritten numeral recognition include projections, zoning, edges, concavities, gradients, and so on.

2.1.3 Classification

After employing an appropriate feature extractor to extract features from the pre-processed data, a suitable classifier is configured for training with those features. By constructing a set of decision functions, the classifier assigns each of its acceptable inputs to one of a finite number of classes or categories in the classification step. However, with even the most sophisticated classifiers, there is always a performance penalty in terms of memory and processing cost.

Deep learning techniques have recently been discovered to be efficient because they mitigate the penalty of using a feature extractor. There are no distinct feature extraction approaches required for these algorithms. Throughout their deep layer-wise structure, they are able to duplicate the inherent features from the image data. These techniques outperform practically every other machine learning model, including traditional state-of-the-art classifiers, in both feature extraction and classification, and this work makes substantial use of some of them. Artificial neural network (ANN) [25] and convolutional neural network (CNN) [9] are being used most frequently in recent times for pattern recognition task.

2.2 Basic Structure of CNN

The convolutional neural network (CNN) is a type of multi-layered neural network that shines at working with two-dimensional data like images. The basic CNN structure for classification tasks consists of an input layer, one or more pairs of convolutional–max-pooling layer(s), one or more dense layer(s), and an output layer, as shown in Figure 2.2. Convolution and max-pooling processes are two of CNN's unique features.

A kernel (2D-filter) convolves over the input picture, called the input feature map (IFM), and generates a convolved feature map (CFM) in a convolution process. As a result, special locality is preserved, and weights are shared among all positions. The hidden layers of a convolutional neural network include convolutional layers. This usually contains a layer that performs multiplication or another dot product, with ReLu as its activation function. Other convolution layers, such as pooling layers, fully connected layers, and normalization layers, are added after that.

By summarizing the existence of features in portions of the feature map, pooling layers provide a method for down sampling feature maps. Average pooling and max pooling are two common pooling algorithms that summarize a feature's average presence and most activated presence, respectively. Convolution and pooling are the first steps in the CNN process, which break down the image into features and analyze them separately. The output of this procedure is fed into a fully connected neural network structure, which is used to make the final classification decision.

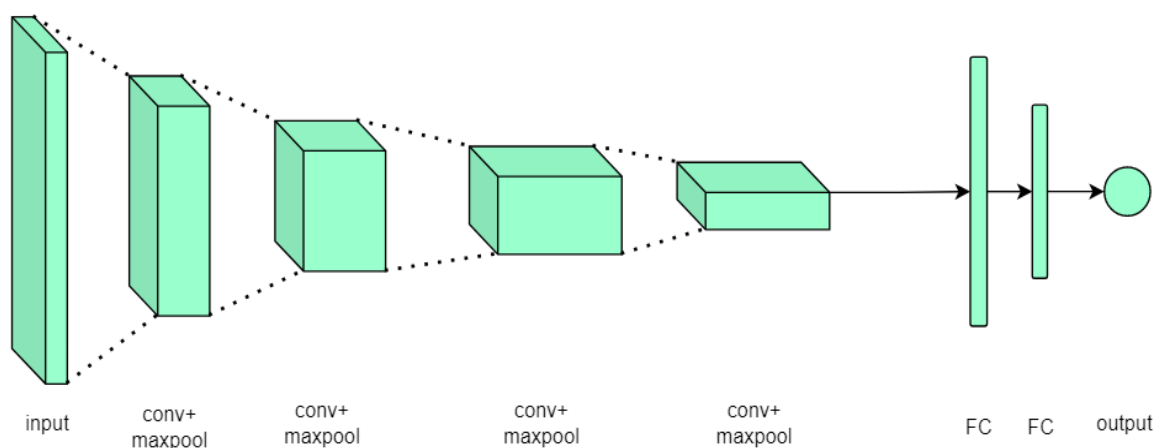


Figure 2.2: Basic structure of Convolutional Neural Network (CNN).

The following is a list of the basic components that make up the CNN structure-

Input layer: The input layer is the entire CNN's input. It usually represents the image's pixel in a 2D matrix for gray level image.

Convolutional layer: The image features are extracted using the convolutional layer. The

shallow features are extracted by the low-level convolutional layer (such as edges, lines, and corners). Through the input of low-level information, a high-level convolutional layer learns abstract features. By convolving a convolution kernel of a given size with the previous layer, the convolutional layer obtains numerous feature activation maps. Breaking up the image into tiny pieces is the first stage for a CNN. This is accomplished by selecting a filter's width and height.

Filter/Kernel: Filters, also known as kernels, are pre-selected $m \times n$ matrices that scan the input picture matrix and provide results via matrix multiplication that provide information about various image properties. The filter examines small sections of the image, or patches. The size of these patches is the same as the filter.

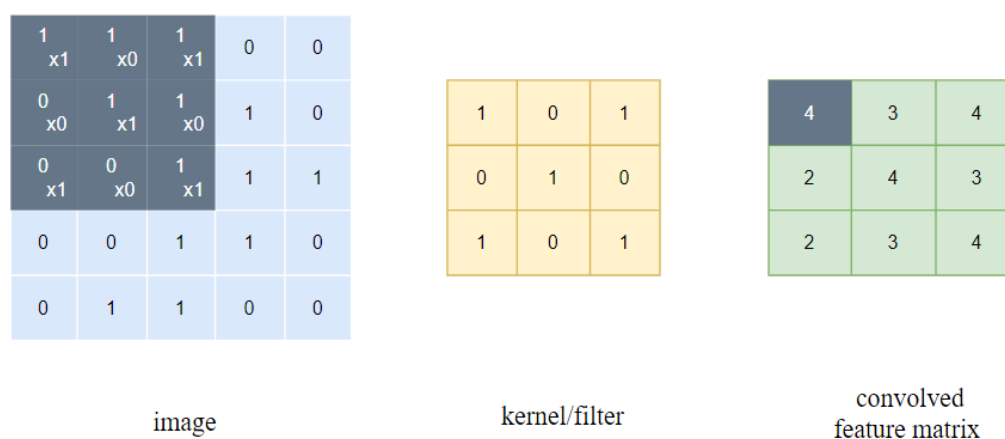


Figure 2.3 Convolved kernel/filter on input image

The light blue color matrix in figure 2.3 is a 5×5 image, whereas the yellow color matrix is a 3×3 kernel/filter. We get convolved feature matrix shown with light green color by computing the kernel on the image matrix. To focus on a different part of the image, simply slide the filter/kernel horizontally or vertically.

Pooling Layers: Pooling layers are commonly used to reduce the number of inputs to speed up computations. Consider the following 4×4 matrix in figure 2.4, when max pooling is applied to this matrix, the result is a 2×2 output.

We take the maximum number for each consecutive 2 X 2 block. We've used a filter with a size of 2 and a stride of 2. These are the pooling layer's hyper-parameters. Apart from max pooling, we may also use average pooling, in which we take the average of the numbers rather than the maximum.

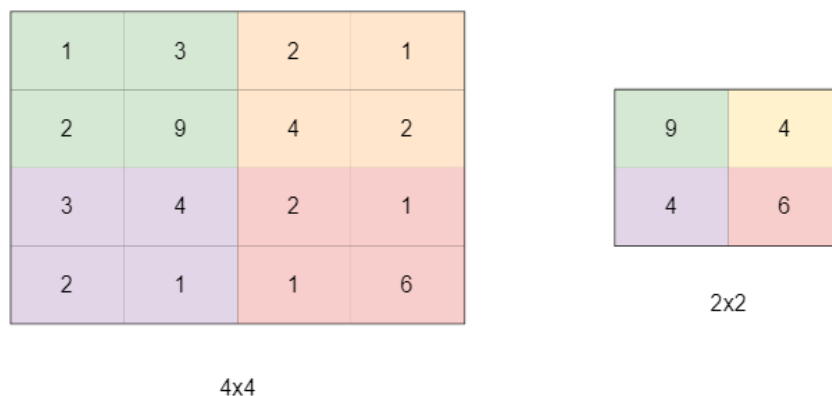


Figure 2.4: Applying max-pooling

Flatten: Flatten is a function that takes a pooled feature map and turns it into a single column that can be supplied to a fully connected layer.

Fully connected layer: Fully Connected Layer is simply, feed forward neural networks. Fully Connected Layers are the network's final layers. The output from the final Pooling or Convolutional Layer, which is flattened and then fed into the fully connected layer, is the input to the fully connected layer. The goal of a fully connected layer is to take the convolution/pooling findings and use them to classify the image into a label. Figure 2.5 show flattening and fully connected layer.

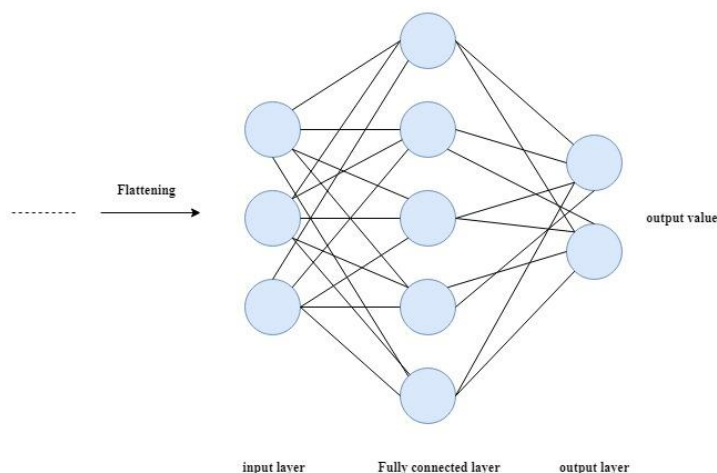


Figure 2.5: Flattening and fully connected layer

2.3 Challenges of HNR

The difficulties of HNR are language-specific. It is based on shape, as well as other complications in different numerals with different language systems. Even for humans, recognition becomes difficult when a language contains similar shaped numerals (such as "৫", and "৬" in Bengali numerals). The challenge of recognition gets challenging even for humans because the similarities turn out to be very near due to differences in people's writing patterns.

Fig. 2.6(a) shows an example of similar shaped numerals. Here, the numeral on the left side of Fig. 2.6(a) is actually "০" but it looks like "৩". On the other hand, the numeral on the right side of Fig. 2.6(a) is actually "৯" but it looks like "৪". As a result, even humans find it challenging to recognize these similarly formed numerals. Due to varied writing methods, the same numeral might appear extremely different in size, shape, and orientation.

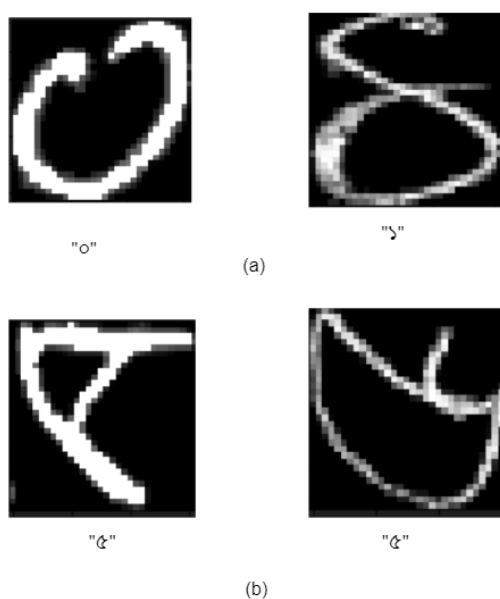


Figure 2.6: Sample handwritten numeral images to understand the challenges of HNR:

(a) Similar shaped numerals; and (b) Same numeral look very different in size, shape, and orientation.

Fig. 2.6(b) shows an example of this. Here, both the images in Fig. 2.6(b) are "৫" in Bengali numeral. But they look completely different in shape and orientation due to different

writing patterns. The presence of curves and holes adds to the difficulty of handwritten numeral numerals. In addition, many noises may occur in the handwritten version, causing the classifier to be unsure while classifying in a given class. As a result, getting high accuracy for HNR is a challenging task.

Again, we used the transfer learning approach for classifying HNR. For this, we need a pre-trained model, which we made by ourselves through a slight modification of an existing model that is needed for our requirements. Finally, in order to achieve the transfer learning approach, we must transfer the weights of the pre-trained model to the suggested model. And this is an extremely important aspect of the method.

2.4 Scope of the Research

Based on existing methodologies, it is evident that an alternative approach to building an HNR system is attainable. Because no training is required from the beginning, the HNR system can be constructed with transfer learning methodologies. Because the classifier has the advantage of employing weights that have already been used for classifying similar tasks, this makes recognition tasks simple. Our proposed method shows that a tiny bit of fine tuning can produce better outcomes than traditional transfer learning procedures, which require removing the top layer from the network and freezing all layers. Image augmentation, which enhances the size of our dataset by using shifting, rotation, zooming, and other techniques, can also help a lot. Again, with transfer learning, training speed grows exponentially while training time is significantly reduced.

CHAPTER III

Handwritten Numeral Recognition Using Transfer Learning Technique

This chapter presents the proposed handwritten numeral recognition through transfer learning in detail in different sections. At first section, the basic structure of the proposed HNR system is described. Then there was a brief description of certain existing models, such as VGG16, VGG19, and RESNET50. Following that, we incorporate and discuss in depth the two procedures that we follow throughout the work. Finally, the significance of the proposed system is described.

3.1 Basic Structure of the Proposed HNR System

The suggested handwritten numeral identification system using the Transfer Learning approach has several important functional phases. To begin, we used multiple models such as VGG-16, VGG-19, and Resnet50 with modest changes for our own objectives to train the MNIST dataset, which is a collection of English handwritten numerals. Then, for each of the models, saved the best weight. The weight is then transferred to our suggested approach for recognizing CVPR, ISI Bengali and Devanagari handwritten numerals from 0 to 9.

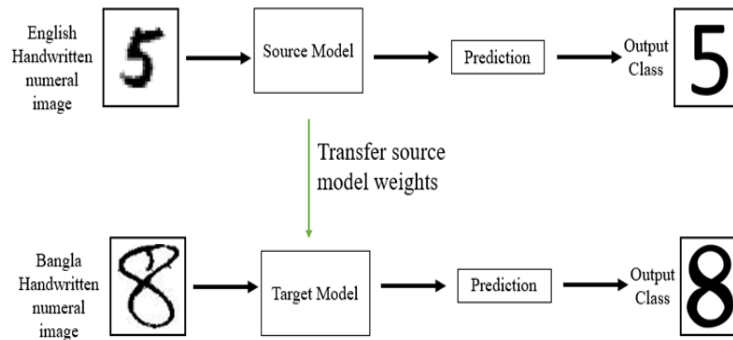


Figure 3.1: Structure of the handwritten numeral recognition using transfer learning

The essential structure of the transfer learning approach is depicted in Figure 3.1. First, we must train the existing model on the English handwritten numbers dataset and keep the best weight. We next apply this weight to our proposed strategy for solving the handwritten numerals problems in Bengali and Devanagari language.

3.2 Basic Structure of the Overall Systems

A pre-trained model is created initially. The MNIST dataset's hand-written English numeral is used to train the pre-trained model. There are two systems in order to ensure that they are both compatible enough to recognize handwritten numerals in various languages. Bengali handwritten numeral recognition using Transfer Learning is one of the systems, while Devanagari handwritten number recognition using Transfer Learning is another. Then, for each of the systems, three architectures were created. VGG-16, VGG-19, and RESNET-50 are the three architectures.

Since there are three architectures, three pre-trained models are required to build up the systems. Both of the systems have been built up into different categories using the same pre-trained models. The testing accuracy of the pre-trained models of VGG-16, VGG-19 and RESNET-50 is 99.57%, 99.61% and 99.40% respectively. These testing accuracies are shown in Table 4.1.

When using transfer learning, each of the models is categorized into two types on the basis of batch size. Batch sizes of 32 and 64 are used here. Then each of the categories is differentiated with respect to whether there is fine tuning or not. All along, the Bengali handwritten numeral recognition system has 12 different unique categories, and so does the Devanagari handwritten numeral recognition system.

The Bengali handwritten dataset of CVPR has been augmented to some certain degree, and so does the Devanagari handwritten dataset of CVPR. Until now, there are three kinds of datasets for both Bengali and Devanagari. There are three types: normal dataset (no rotation), normal dataset with a 10 degree rotation (both clockwise and anticlockwise direction), and normal dataset with a 20 degree rotation (both clockwise and anticlockwise direction).

Each of the 12 different categories of the Bengali handwritten numeral recognition system is trained using transfer learning with the three kinds of datasets, and so does the Devanagari handwritten numeral recognition system. Each of the systems has 36 different categories. That means there are a total of 72 different categories regarding both the Bengali and Devanagari handwritten numeral recognition systems. Figure 3.2 shows an illustration of how all the different systems have been categorized.

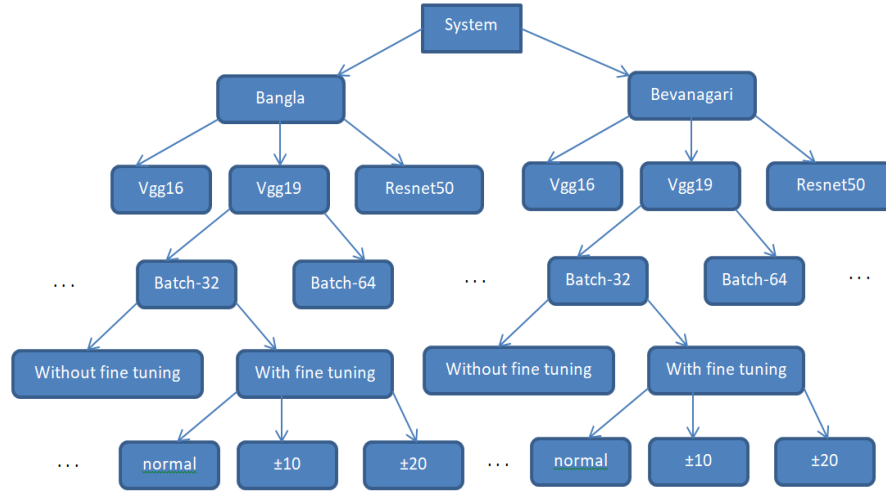


Figure 3.2: An illustration of how all the systems have been categorized

3.3 Basic Structure of Existing Model

For the development of our suggested method, we utilized various existing models. VGG16, VGG19, and RESNET50 are among them. Here is some basic information regarding these models.

VGG16: ImageNet, a massive visual database project utilized in visual object recognition software research, uses VGG16, a simple and widely used Convolutional Neural Network (CNN) Architecture. Karen Simonyan and Andrew Zisserman of the University of Oxford created and launched the VGG16 Architecture in their essay "Very Deep Convolutional Networks for Large-Scale Image Recognition" in 2014. The term 'VGG' stands for Visual Geometry Group, a group of scholars at the University of Oxford who designed this architecture, and the number '16' indicates that there are 16 layers in this architecture.

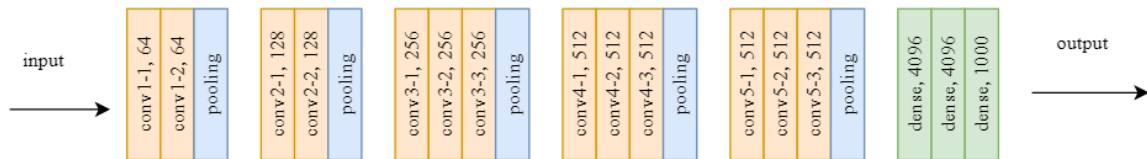


Figure 3.3: Vgg16 model block diagram

The first two layers have the same padding and 64 channels of 3*3 filter size. After a stride (2, 2) max pool layer, the next two layers feature convolution layers of 128 filter size (3, 3). This is followed by a stride (2, 2) max pooling layer, which is the same as the previous layer. Then there are three convolution layers, each with 256 filters of size (3, 3). Following that, there are two sets of three convolution layers, as well as a max pool layer. Each has 512 filters of size (3, 3) and the same padding of stride (2, 2). Figure 3.3 depicts block diagram of vgg16.

VGG19: The VGG-19 network is made up of 19 CNN layers and three fully connected layers, as its name suggests. The VGG-19 network's architecture is depicted in the diagram below with figure 3.4.

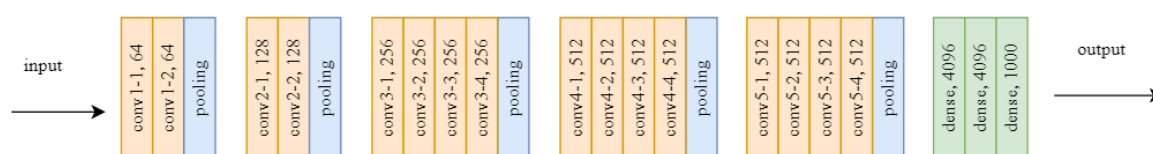


Figure 3.4: Vgg19 model block diagram

As can be seen, the VGG-19 architecture is appropriately depicted in the diagram above. This architecture is made up of three sorts of levels, namely, the convolution layer to extract the feature from the image by employing various numbers and types of filters, the max-pooling layer to reduce the image size and extract the feature from the feature map obtained from these filters present in the convolution layer, the flatten layer to convert the batches of feature maps into 1D tensor, and finally, three fully-connected layers, where the first two have a dense unit of 4096 layers and the final classification layer has 1000 way classification, thus containing one for each class.

Resnet50: ResNet-50 is a 50-layer deep convolutional neural network. ResNet, short for Residual Networks, is a well-known neural network that serves as the foundation for many computer vision tasks. In their 2015 computer vision research paper titled 'Deep Residual Learning for Image Recognition,' Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun proposed a unique neural network. The 'Vanishing Gradient Problem' is a key

drawback of Convolutional Neural Networks. The value of gradient reduces greatly during back-propagation, therefore weights scarcely change. ResNet is used to overcome this.

It employs the technique of "SKIP CONNECTION" depicted in figure 3.5. The term "skip connection" refers to the process of adding the original input to the output of a convolutional block.

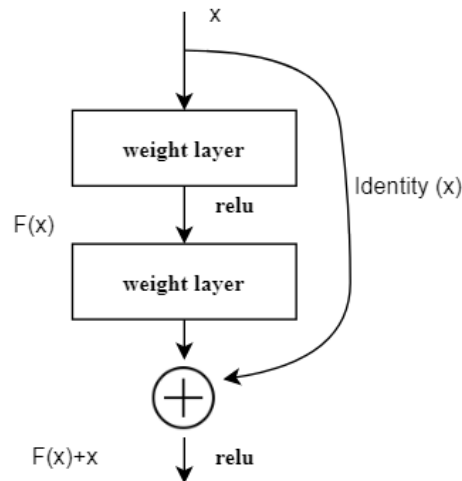


Figure 3.5: Residual block

All algorithms are trained on the output 'Y', whereas ResNet is trained on the output $F(X)$. To put it another way, ResNet aims to make $F(X) = 0$ so that $Y=X$. In general, we don't know the ideal number of layers (or residual blocks) for a neural network, which may vary depending on the dataset's complexity. By adding skip connections to our network, we are allowing the network to skip training for the layers that are not helpful and do not contribute value to overall accuracy, rather than using the number of layers as a key hyper-parameter to tune.

Skip connections, in a sense, make our neural networks dynamic, allowing us to optimally adjust the number of layers during training.

There are two types of blocks in ResNet-50: Identity Block and Convolutional Block depicted in figure 3.6. If and only if the input size equals the output size, the value of 'x' is added to the output layer. If this isn't the case, a 'convolutional block' is added to the shortcut path to make the input and output sizes equal.

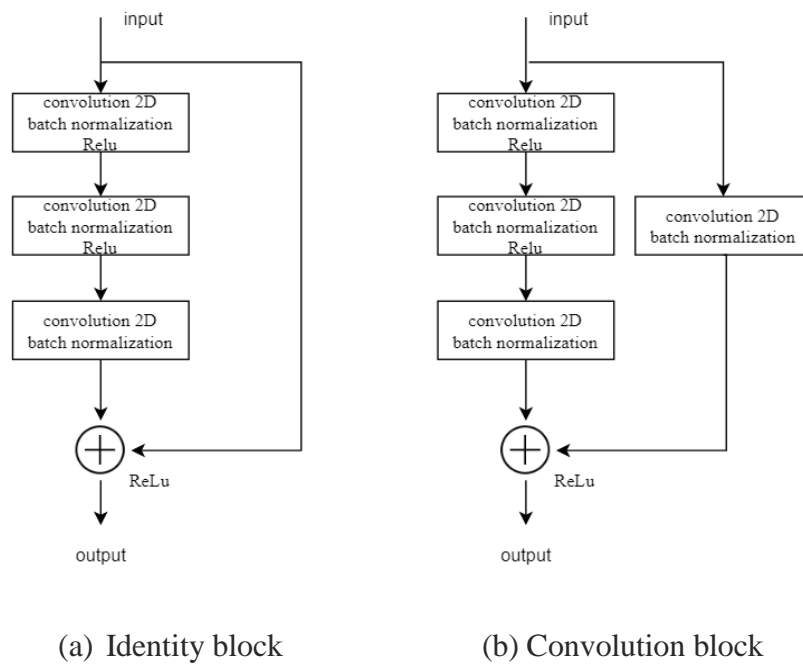


Figure 3.6: Identity block and Convolution block

1. The identity block — The identity block is a standard ResNets block that corresponds to the condition where the input and output activations have the same dimension.
2. The Convolutional block – When the input and output dimensions do not match, we can utilize this sort of block. The shortcut path differs from the identity block in that it includes a CONV2D layer.

Table 3.1: Standard structure of resnet50 model

Layer name	Output size	50 Layers	No of Layers
conv1	112x112	7x7, 64, stride 2 3x3, maxpool, stride 2	1
conv2	56x56	1x1, 64 3x3, 64 1x1, 256	3x3 = 9
conv3_x	28x28	1x1, 128 3x3, 128 1x1, 512	4x3 = 12
conv3_x	14x14	1x1, 256 3x3, 256 1x1, 1024	6x3 = 18
conv5_x	7x7	1x1, 512 3x3, 512 1x1, 2048	3x3 = 9
-	1x1	Average pool, 1000-d, fc, softmax	1

The following element in table 3.1 is found in the resnet50 architecture: We get one layer from a convolution with a kernel size of $7 * 7$ and 64 distinct kernels, all with a stride of size 2. Following that, we have max pooling with a stride size of 2. Following that, we have max pooling with a stride size of 2. There is a $1 * 1$, 64 kernel in the next convolution, followed by a $3 * 3$, 64 kernel, and finally a $1 * 1$, 256 kernel. These three layers are repeated three times in total, giving us 9 layers in this phase. This phase was repeated four times, giving us 12 layers. Next, we see a kernel of $1 * 1$, 128 followed by a kernel of $3 * 3$, 128 and finally a kernel of $1 * 1$, 512. Then there's a $1 * 1$, 256 kernel, followed by $3 * 3$, 256 and $1 * 1$, 1024 kernels, which are repeated six times for a total of 18 layers. Then a $1 * 1$, 512 kernel was added, followed by two more $3 * 3$, 512 and $1 * 1$, 2048 kernels, for a total of 9 layers. After that, we run an average pool and finish with a fully connected layer with 1000 nodes, followed by a softmax function, giving us one layer. The activation functions and the max/average pooling layers are not counted. As a result, we get a Deep Convolutional network with $1 + 9 + 12 + 18 + 9 + 1 = 50$ layers.

3.4 Transfer Learning Strategies

Different transfer learning procedures and approaches can be used depending on the domain, the task at hand, and the data available. The standard procedure includes the following steps:

- (a) Use layers from a model that has already been trained.
 - (b) Freeze them to prevent future training rounds from destroying any of the information they contain.
 - (c) Add some additional, trainable layers on top of the frozen layers.
- They'll figure out how to turn previous features into predictions on a new dataset, and then
- (d) put the new layers to the test on your data.

Fine-tuning is a last, optional step that entails unfreezing the entire model (or a portion of it) and retraining it on new data with a very small learning rate. By incrementally modifying the pre-trained features to the new data, this has the potential to achieve significant improvement.

We apply both procedures to the model that has been proposed.

3.5 Development of the Proposed Transfer Learning Technique

The steps that we followed are listed as follows:

First, choose a model and modify it slightly for our requirements. Then, we trained these models with English handwritten numerals and saved weights. We used it as a pre-trained model for solving related tasks such as recognizing Bengali and Devanagari handwritten numerals. We used the vgg16, vgg19, and resnet50 model architectures as a pre-trained model with the appropriate changes. Figure 3.7 shows the way model vgg16 was used as a pre-trained model. Conv2D with the same padding, batch normalization, Relu activation function, and Maxpooling2D with stride (2x2) make up each block.

The MNIST dataset, which contains English handwritten digits in a 28x28 gray-scale image, was used to train the vgg16 model. We transformed it to 32x32 because images in our Bengali and English datasets are also 32x32. Table 3.1 shows the model summary when trained with the MNIST dataset with the vgg16 architecture. Table 3.2 shows that the flattening layer has an input of (2, 2, 512). If we use the vgg16 architecture without

making any changes, the flatten layer's input will be (1, 1), which is not ideal for our needs. To reduce the over-fitting problem, we utilized several dropout layers once more. We employed two dense layers with a total of 2048 and 1024 neurons in each. Finally, we classified handwritten Bengali and Devanagari numerals from 0 to 9 using the softmax function.

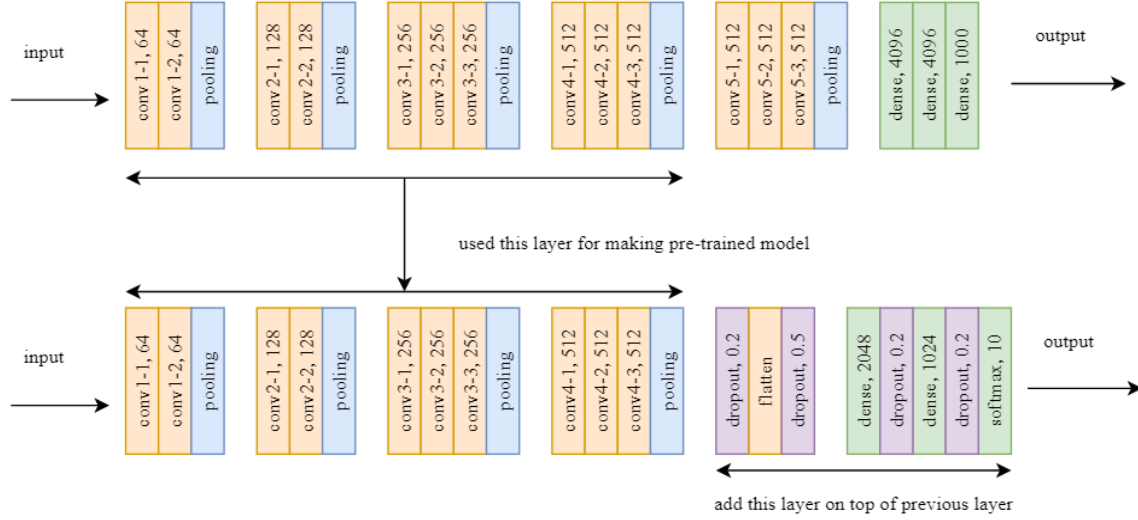


Figure 3.7: Pre-trained model structure with vgg16

Table 3.2: Partial summary of pre-trained model vgg16

Layer(Type)	Output shape
input_1 (Input Layer)	(32, 32, 1)
zero_padding2d(Zero Padding 2D)	(38, 38, 1)
block1_conv1 (Conv2D)	(38, 38, 64)
block1_conv2 (Conv2D)	(38, 38, 64)
block1_pool (MaxPooling2D)	(19, 19, 64)
block2_conv1 (Conv2D)	(19, 19, 128)
block2_conv2 (Conv2D)	(19, 19, 128)
block2_pool (MaxPooling2D)	(9, 9, 128)
...	...
block4_conv1 (Conv2D)	(4, 4, 512)
block4_conv2 (Conv2D)	(4, 4, 512)
block4_conv3 (Conv2D)	(4, 4, 512)
block4_pool (MaxPooling2D)	(2, 2, 512)
flatten (Flatten)	(2048)
fc1 (Dense)	(2048)
fc2 (Dense)	(1024)
predictions (Dense)	(10)

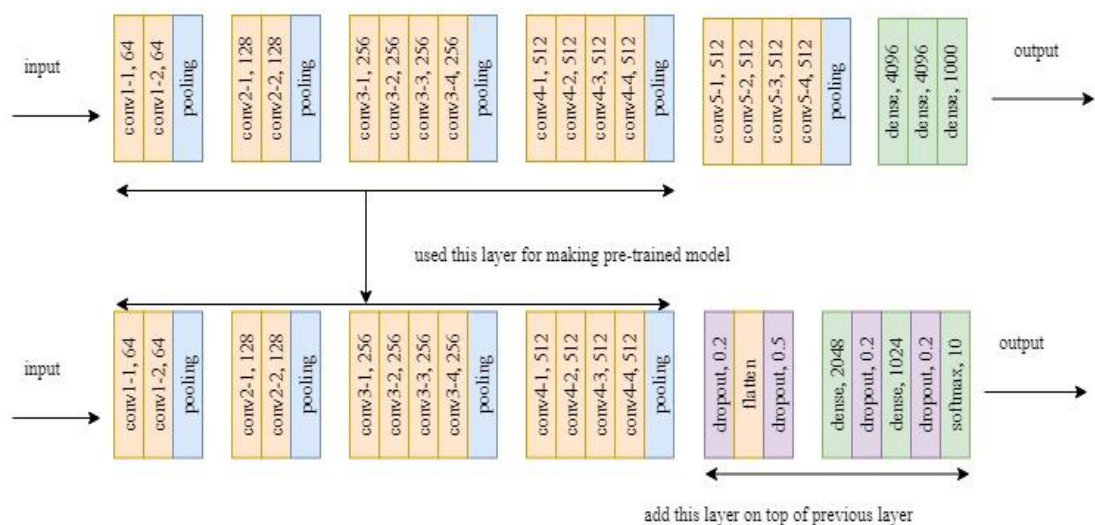


Figure 3.8: Pre-trained model structure with vgg19

Figure 3.8 displays a pre-trained model based on a modified version of vgg19. For our needs, we change the normal design of vgg19. We'll leave the first four blocks untouched. The 2D array is then converted to 1D using flattening. The output is sent into two layers that are fully connected. Finally, for multiclass classification, we employed softmax classification with ten neurons. We placed a dropout layer between these layers to assist decrease the problem of over-fitting. Two convolutions and one max-pooling layer are found in the first two blocks. There are 64 and 128 neurons in each convolution layer, respectively. Three convolutions and one max-pooling operation make up the next two blocks. Each convolution in the third and fourth blocks has 256 and 512 neurons, respectively. All of them have the same 3x3 kernel size. Conv2D with the same padding, batch normalization, relu activation function, and Maxpooling2D with stride (2x2) make up each block. The output of a node is defined by the activation function of that node given an input or group of inputs. The rectified linear activation function, or ReLU for short, is a piecewise linear function that, if the input is positive, outputs it directly; else, it outputs zero. Batch normalization is the process of adding more layers to a deep neural network to make it faster and more reliable. The standardizing and normalizing procedures are performed by the new layer on the input of a previous layer.

Table 3.1 shows the model summary when trained with the MNIST dataset with the vgg19 architecture. Table 3.3 shows that the flattening layer has an input of (2, 2, 512). If we use the vgg19 architecture without making any changes, the flatten layer's input will be (1, 1),

which is not ideal for our needs. To reduce the over-fitting problem, we utilized several dropout layers once more. We employed two dense layers with a total of 2048 and 1024 neurons in each. Finally, we classified handwritten Bengali and Devanagari numerals from 0 to 9 using the softmax function.

Table 3.3: Partial summary of pre-trained model vgg19

Layer(Type)	Output shape
input_1 (Input Layer)	(32, 32, 1)
zero_padding2d (Zero Padding 2D)	(38, 38, 1)
block1_conv1 (Conv2D)	(38, 38, 64)
block1_conv2 (Conv2D)	(38, 38, 64)
block1_pool (MaxPooling2D)	(19, 19, 64)
...	...
block3_conv3 (Conv2D)	(9, 9, 256)
block3_pool (MaxPooling2D)	(4, 4, 256)
block4_conv1 (Conv2D)	(4, 4, 512)
block4_conv2 (Conv2D)	(4, 4, 512)
block4_conv3 (Conv2D)	(4, 4, 512)
block4_conv4 (Conv2D)	(4, 4, 512)
block4_pool (MaxPooling2D)	(2, 2, 512)
dropout (Dropout)	(2, 2, 512)
flatten (Flatten)	(2048)
dropout_1 (Dropout)	(2048)
fc1 (Dense)	(2048)
fc2 (Dense)	(1024)
predictions (Dense)	(10)

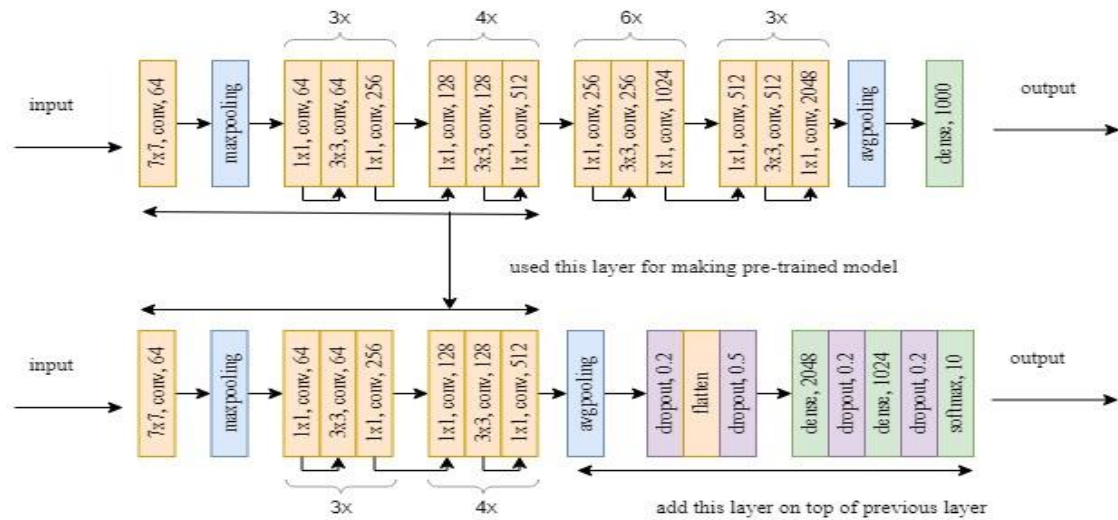


Figure 3.9: Pre-trained model structure with resnet50

The second phase in our proposed strategy is that we took our pre-trained model and tweaked it somewhat to fit our needs. We use two types of transfer learning procedures in this study: one without fine tuning and the other with fine tuning.

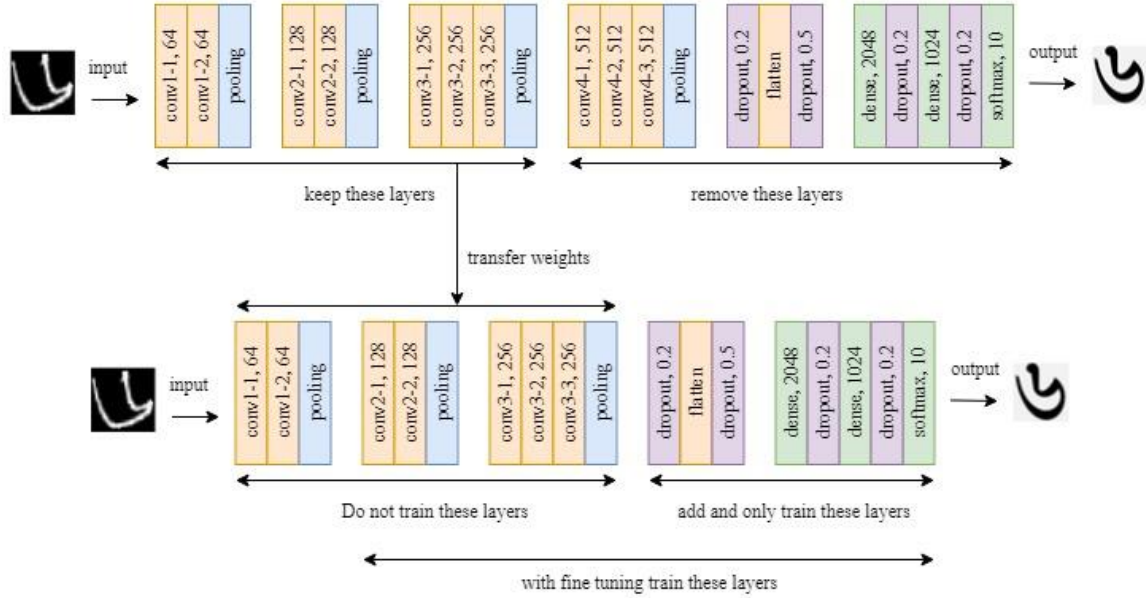


Figure 3.10: Proposed method structure for vgg16, with and without fine tuning

Figure 3.10 shows a structural depiction of our transfer learning methodologies with vgg16. We left the first three blocks of the pre-trained model alone and built a flat and dense layer on top of them. Without fine-tuning, we froze these three blocks, i.e., we did not train these layers. Only the top levels of these layers are subjected to training. When training with CVPR Bengali and Devanagari handwritten numerals, pre-trained model weights from when training with English handwritten numerals are used. This makes weight initialization go much more smoothly than random initialization. We just froze the first blocks with fine-tuning, i.e., we did not train these layers. On top of these layers, the rest of the layers are trained.

Table 3.4: Training parameter with transfer learning approaches for vgg16

Type of Parameter	Without Fine Tuning	With Fine Tuning
Total	13,949,642	13,949,642
Trainable	6,304,778	13,832,586
Non-trainable	7,644,864	117,056

Transfer learning can reduce the number of training parameters, as shown in Table 3.4. Without fine tuning techniques, the total number of non-trainable parameters using the vgg16 architecture is 7,644,864. This accounts for more than half of the total number of parameters. As a result, it will undoubtedly assist us in accelerating the training process.

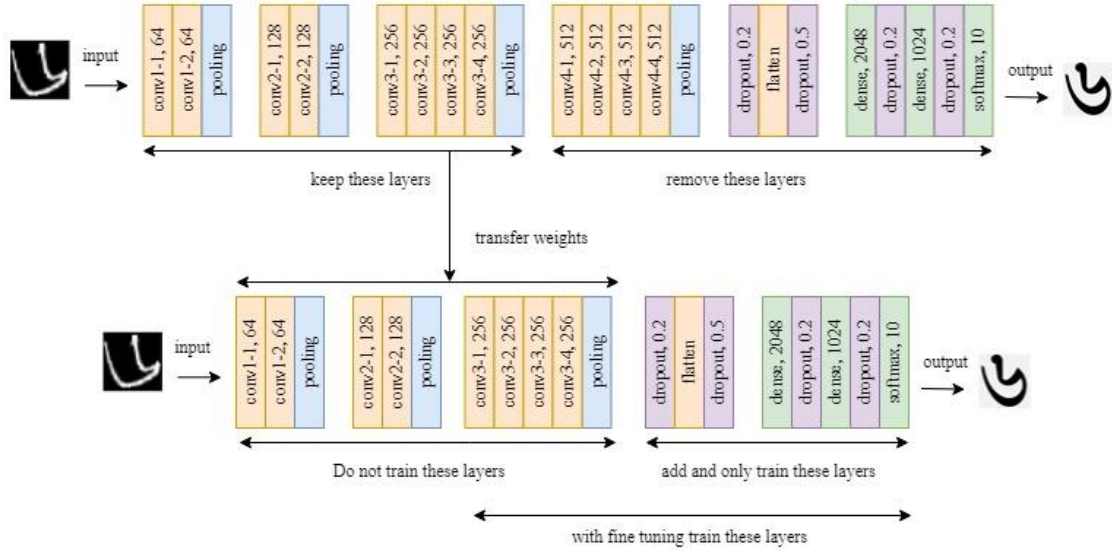


Figure 3.11: Proposed method structure vgg19, with and without fine tuning

Figure 3.11 shows a structural depiction of our transfer learning methodologies with vgg19. We left the first three blocks of the pre-trained model alone and built a flat and dense layer on top of them. Without fine-tuning, we froze these three blocks, i.e. we did not train these layers. Only the top levels of these layers are subjected to training. When training with CVPR Bengali and Devanagari handwritten numerals, pre-trained model weights from when training with English handwritten numerals are used. This makes weight initialization go much more smoothly than random initialization. We just froze the first blocks with fine-tuning, i.e., we did not train these layers. On top of these layers, the rest of the layers are trained.

Table 3.5: Training parameter with transfer learning approaches for vgg19

Type of Parameter	Without Fine Tuning	With Fine Tuning
Total	12,829,130	16,902,602
Trainable	10,499,082	16,635,914
Non-trainable	2,330,048	266,688

Transfer learning can reduce the number of training parameters, as shown in Table 3.5. Without fine tuning techniques, the total number of non-trainable parameters using the vgg19 architecture is 2,330,048. As a result, it will undoubtedly assist us in accelerating the training process.

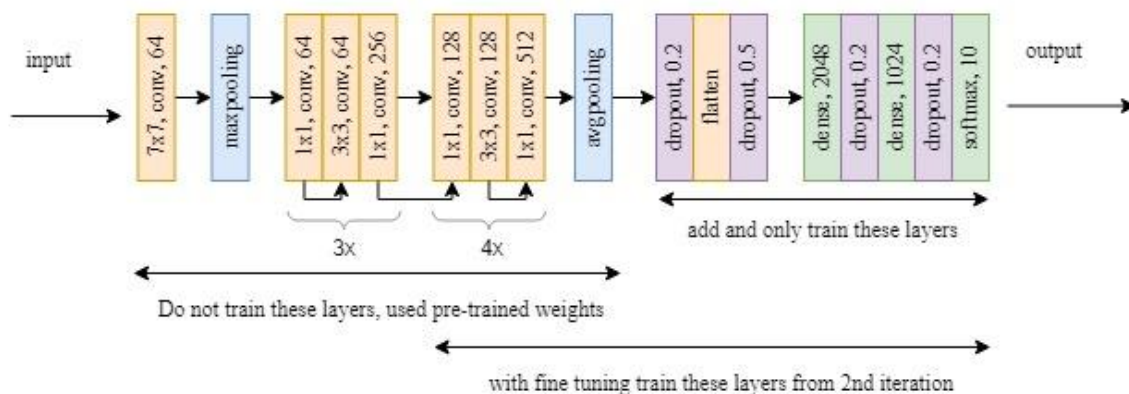


Figure 3.12: Proposed method structure resnet50, with and without fine tuning

The same method applies to the existing resnet50 model depicted in figure 3.9 and 3.12, starting with the creation of a pre-trained model and ending with testing with Bengali and devanagari handwritten numerals. Machine learning professionals add extra layers when working with deep convolutional neural networks to tackle an issue in computer vision. These additional layers aid in the faster resolution of complicated problems since the individual layers can be trained for different jobs to produce highly accurate outcomes. Deep residual nets make use of residual blocks to improve the accuracy of the models. The concept of “skip connections,” which lies at the core of the residual blocks, is the strength of this type of neural network.

3.6 Significance of the Proposed System

There are some noticeable differences between the proposed HNRUTL and typical handwritten numeral recognition systems. To train a neural network from scratch, a lot of data is usually required, but access to that data isn't always possible – this is when transfer learning comes in handy. Because the model has already been pre-trained, a good machine learning model can be generated with relatively little training data using transfer learning. To accurately train a machine learning algorithm, a large amount of data is normally

required. It requires time, effort, and talent to develop labeled training data. Transfer learning reduces the amount of training data needed for new machine learning models because the majority of the model has already been learned. When compared to a target model that learns from scratch, using knowledge from a source model (pre-trained model) may aid in fully learning the target task. As a result, the overall time it takes to develop/learn a model gets shorter. The challenge is broken into two key functional steps: acquiring weights from a pre-trained model for recognizing handwritten English numerical images, and then recognition of related tasks, such as Bengali and Devanagari handwriting numerals, with the proposed model using those weights as an initial value. This provides a number of advantages, the most important of which are reduced training time, improved neural network performance (in most circumstances), and solve the problem of the absence of a large amount of data. Furthermore, this strategy uses a simple pre-processing of the images and does not use any manual feature extraction methods. Traditional methods, on the other hand, employ several feature extraction methods, as well as various pre-processing procedures and recognition with various machine learning tools.

CHAPTER IV

Experimental Studies

The effectiveness and performance of the suggested handwritten numeral recognition system are investigated in this chapter. For the ISI [18] dataset, different experimental analyses are provided. A regulated preprocessing is used to produce the patterns and make them suitable for feeding to a classifier. The suggested BHNRTL's recognition accuracy is evaluated in terms of iterations, batch size, rotation angles, and the confusion matrix. Finally, the suggested system's classification performance is compared to that of other well-known approaches in order to validate its effectiveness. The test set recognition accuracy is used to evaluate the proposed method's performance.

4.1 Description about pre-trained models dataset

The MNIST dataset [26], which stands for Modified National Institute of Standards and Technology, is used to train the model. It contains 60,000 training instances and about 10,000 test cases. It's a subset of NIST's wider collection. The digits were size-normalized and centered in a fixed-size picture of handwritten single digits between 0 and 9 with a square form of 28x28 pixel gray-scale images.

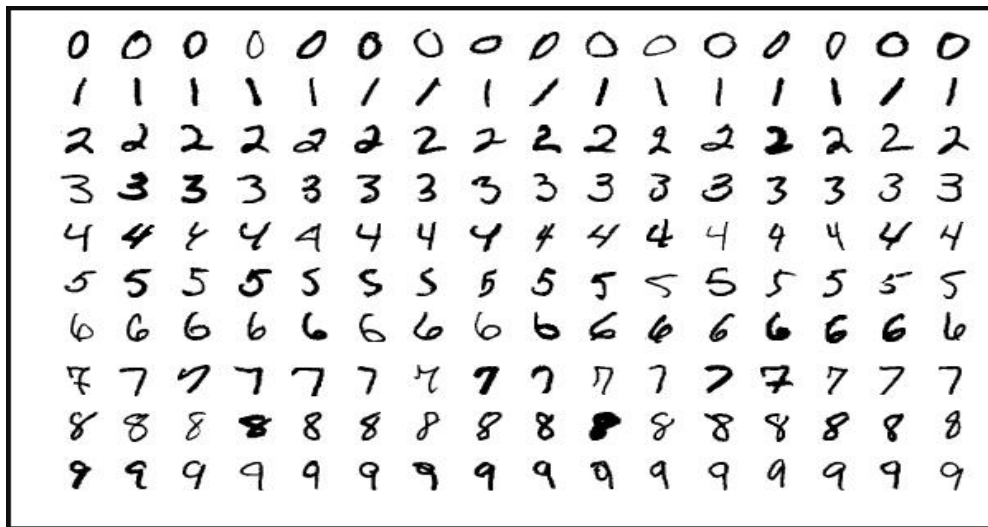


Figure 4.1: Samples of handwritten English numerals from MNIST dataset

4.2 Benchmark Dataset and Pre-processing

For some languages, such as English, the recognition accuracy of handwritten numerals is satisfactory. However, due to inherited difficulties, the accuracy of handwritten numerical identification in some languages, such as Bengali, is low. Even in printed Bengali, certain numerals are remarkably similar to one another. In Bengali, for example, the handwritten and printed forms of ১ and ২ are remarkably similar in shape.

For example, ১ and ২ quite similar in appearance to each other in the Bengali language. For another example, in the Devanagari language १ (in Bengali it is ১, in English it is 1) and २ (in Bengali it is ২, in English it is 2) are quite similar. As a result, a benchmark handwritten Bengali and Devanagari numerals images dataset preserved by the Computer Vision and Pattern Recognition (CVPR) Unit of the Indian Statistical Institute (ISI) [18] is utilized for the tests and to assess the performance of the suggested system. This dataset comprises a huge number of training and test samples for Bengali numerals. The samples are scanned images of postal mail pin numbers from a variety of people of varied sexes, ages, and educational levels. It contains 19392 training and testing image samples of Bengali handwritten numerals for ten digits. There are approximately 1900 training images and 400 testing images for each digit. In this study, 18000 (=1800x10) images were preprocessed and utilized as the training set, while the remaining 4,000 images were used as the test set.

Figure 4.2 depicts a few sample representations of each numeral in the Bengali script, illustrating the degree of uncertainty that makes recognition difficult. Benchmark HNI datasets developed by the Computer Vision and Pattern Recognition (CVPR) Unit of the Indian Statistical Institute (ISI) for Bengali and Devanagari are utilized to verify the proposed system's performance. For Bengali and Devanagari numerals, the CVPR, ISI dataset offers a significant number of training and test samples. The examples are scanned pictures of various people's postal mail pin codes. It includes 19392 and 18793 Bengali and Devanagari training examples for 10 numerals, as well as 4000 and 3763 testing samples. Preprocessing is done on 18000 (=1800x10) training set photos in this study, and the images are used in both Bengali and Devanagari training. This dataset has been used in

various notable efforts for Bengali and Devanagari handwritten number recognition.

Image examples in Bengali and Devanagari databases are available in a range of sizes, resolutions, and formats. Certain preprocessing processes are undertaken to transform the picture data into a standard format that can be exploited by recognition systems. First, to generate the binary images from the input images, automatic thresholding is applied. The background and foreground are then swapped, resulting in a lower computational cost for auto-encoders and classifiers. After that, the images are scaled to 32x32 pixels. To preserve the exact quality of the photos, the double-type matrix is used for shrunk pixels. For English numbers, the well-known MNIST [26] dataset is chosen. This dataset contains 60,000 training data points and 10,000 test data points. All of the photos have the same size, 28x28 pixels wide, with pixel values ranging from 0-255.

Image examples of various sizes, resolutions, and shapes are included in the dataset. Before the picture data is given into the recognition algorithms, a basic preprocessing step is done on it, as though all of the photos had the same shape. Automatic thresholding is done to the photos before they are converted to binary images. Because the numbers are written in black over white in the original pictures, there is more value of 1 in the images and a few zeros are necessary to repeat the pattern, the foreground and background are switched to lessen the time complexity. After the foreground-background switch, binary pictures have fewer ones and more zeros, resulting in decreased computing costs for auto-encoders and classifiers.

After that, the images are scaled to the same dimensions (32x32). For the reduced pixels, the matrix type is kept double to maintain the quality of most of the ideal images.

4.3 Experimental Setup

An initial training pattern is rotated at a predetermined angle in both anticlockwise and clockwise orientations to create two patterns. For simplicity, just 10° and 20° of rotation are evaluated, and this degree of rotation was proven to be beneficial in the previous study [21]. The benchmark dataset's designated test set was not used in any of the training stages to ensure that the proposed system can respond appropriately to unknown input.

English Numeral	Bangla Numeral	Sample Bangla Handwritten Numeral Images					
0	০						
1	১						
2	২						
3	৩						
4	৪						
5	৫						
6	৬						
7	৭						
8	৮						
9	৯						

Figure 4.2: Samples of handwritten Bengali numerals from CVPR, ISI dataset

Zero									
One									
Two									
Three									
Four									
Five									
Six									
Seven									
Eight									
Nine									

Figure 4.3: Samples of handwritten Devanagari numerals from CVPR, ISI dataset

Keras and Tensorflow are used to implement the proposed HNRUTL system in Python. The experiments were run on a desktop in a Colab environment with the following settings: Backend for Python 3 Google Compute Engine (GPU) with dedicated RAM.

4.4 Experimental Results and Analysis

This section evaluates the proposed HNRUTL's performance on the CVPR, ISI dataset. Going through the result analysis is a mandatory job to choose the best two systems among the 72 systems discussed in section 3.2. Obviously, one system is for the Bengali

language, and another is for the Devanagari language. When all the different categories are trained using the transfer learning method with the help of those pre-trained models, satisfactory training accuracy has been found. Early stopping has been used when the training process is ongoing. The training process is stopped as soon as it is observed that the validation losses of the very previous ten epochs are unchanged. The test accuracies of all the categories are also satisfactory. All of the test accuracies for each category are shown in Table 4.2 to 4.5.

The best test accuracies for VGG-16, VGG-19, and RESNET-50 are 99.58%, 99.68%, and 98.93%, respectively, when using the Bengali CVPR dataset with 10 degree rotation (both clockwise and anticlockwise direction) augmentation, 64 batch size, and fine tuning. The best test accuracies for VGG-16, VGG-19, and RESNET-50 are 99.60%, 99.63%, and 99.31%, respectively, when using the Devanagari CVPR dataset with 10 degree rotation (both clockwise and anticlockwise direction) augmentation, 64 batch size, and fine tuning. The best test accuracies for VGG-16, VGG-19, and RESNET-50 are 99.73%, 99.40%, and 98.80%, respectively, when using the Bengali CVPR dataset with 10 degree rotation (both clockwise and anticlockwise direction) augmentation, 32 batch size, and fine tuning. The best test accuracies for VGG-16, VGG-19, and RESNET-50 are 99.52%, 99.55%, and 99.49%, respectively, when using the Devanagari CVPR dataset with 10 degree rotation (both clockwise and anticlockwise direction) augmentation, 32 batch size, and fine tuning.

Table 4.1: Pre-trained model testing accuracies

Pre-trained model name	Testing Accuracy (MNIST)(%)
VGG16	99.57
VGG19	99.61
RESNET50	99.4

Batch size = 64

Table 4.2: Testing accuracies for CVPR Bengali dataset with batch size 64

Model Used	Normal(%)	Normal FT(%)	± 10 (%)	± 10 FT(%)	± 20 (%)	± 20 FT(%)
VGG16	98.48	99.18	99.15	99.58	98.78	99.58
VGG19	98.88	99.28	98.90	99.68	98.75	99.58
RESNET50	97.50	98.05	98.08	98.93	98.23	98.55

Table 4.3: Testing accuracies for CVPR Devanagari dataset with batch size 64

Model Used	Normal(%)	Normal FT(%)	± 10 (%)	± 10 FT(%)	± 20 (%)	± 20 FT(%)
VGG16	99.25	99.49	99.44	99.60	99.23	98.80
VGG19	98.75	99.28	99.09	99.63	99.04	99.52
RESNET50	98.72	98.91	99.07	99.31	98.85	99.31

Batch size = 32

Table 4.4: Testing accuracies for CVPR Bengali dataset with batch size 32

Model Used	Normal(%)	Normal FT(%)	± 10 (%)	± 10 FT(%)	± 20 (%)	± 20 FT(%)
VGG16	98.38	99.18	98.98	99.73	98.88	99.43
VGG19	98.35	99.48	99.08	99.40	98.53	99.48
RESNET50	97.05	98.25	97.78	98.80	97.88	98.70

Table 4.5: Testing accuracies for CVPR Devanagari dataset with batch size 32

Model Used	Normal(%)	Normal FT(%)	± 10 (%)	± 10 FT(%)	± 20 (%)	± 20 FT(%)
VGG16	99.04	98.77	99.25	99.52	99.15	99.55
VGG19	99.07	99.31	99.33	99.55	99.25	99.60
RESNET50	98.64	99.01	99.01	99.49	98.43	99.31

For the Bengali CVPR dataset with batch size 64, a transfer learning model using the VGG-19 architecture with fine tuning gave the highest testing accuracy of 99.68% when the dataset is normal with 10 degree rotational augmentation. For the Bengali CVPR dataset with batch size 32, a transfer learning model using the VGG-16 architecture with fine tuning gave the highest testing accuracy of 99.73% when the dataset is normal with

10 degree rotational augmentation. For the Devanagari CVPR dataset with batch size 64, a transfer learning model using the VGG-19 architecture with fine tuning gave the highest testing accuracy of 99.63% when the dataset is normal with 10 degree rotational augmentation. For the Devanagari CVPR dataset with batch size 32, a transfer learning model using the VGG-19 architecture with fine tuning gave the highest testing accuracy of 99.55% when the dataset is normal with 10 degree rotational augmentation.

Hence, for the Bengali CVPR dataset with batch size 32, a transfer learning model using the VGG-16 architecture with fine tuning gave the highest testing accuracy of 99.73% when the dataset is normal with 10 degree rotational augmentation. But, for Devanagari CVPR dataset, for batch size 64, the transfer learning models using the VGG-19 architecture with fine tuning gave the highest testing accuracy of 99.63% when the dataset is normal with 10 degree rotational augmentation.

For batch size 32 and the Bengali CVPR dataset, figure 4.4 depicts a comparison of the systems that provided the highest testing accuracy for the VGG-16, VGG-19, and RESNET-50 architectures.

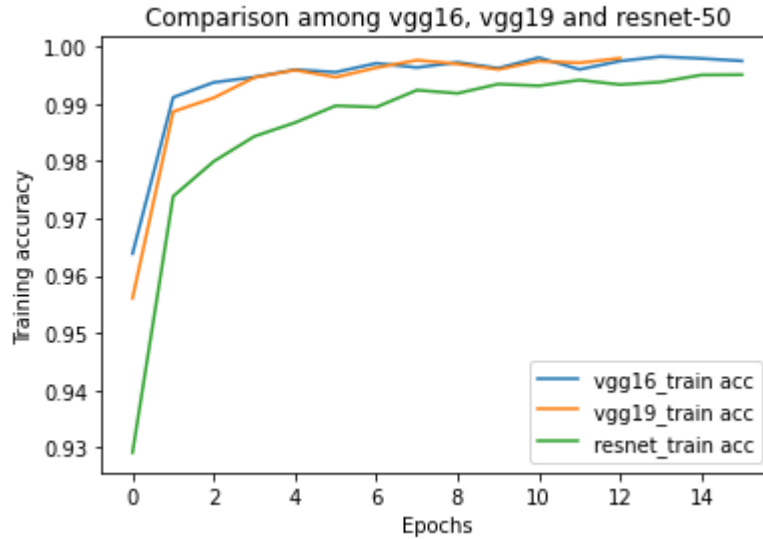


Figure 4.4: Comparison among VGG-16, VGG-19 and RESNET-50 for Bengali CVPR and batch size 32

For batch size 32 and the Devanagari CVPR dataset, figure 4.5 depicts a comparison of the systems that provided the highest testing accuracy for the VGG-16, VGG-19, and

RESNET-50 architectures.

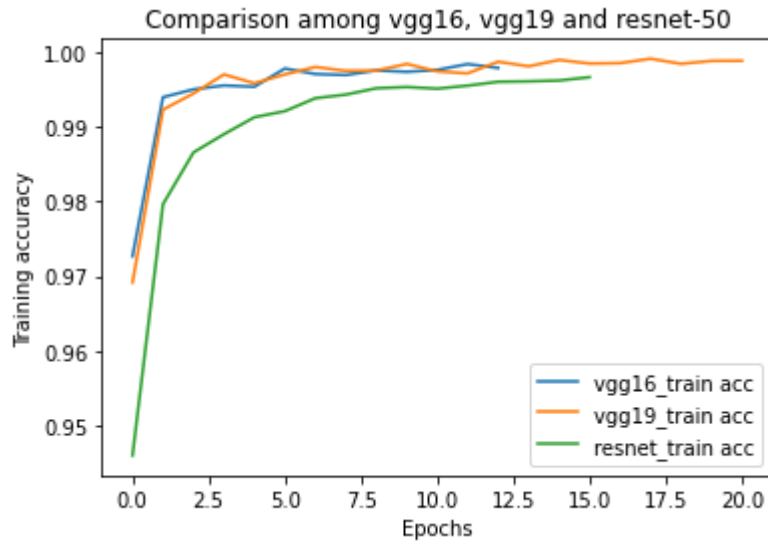


Figure 4.5: Comparison among VGG-16, VGG-19 and RESNET-50 for Devanagari CVPR and batch size 32

For batch size 64 and the Bengali CVPR dataset, figure 4.6 depicts a comparison of the systems that provided the highest testing accuracy for the VGG-16, VGG-19, and RESNET-50 architectures.

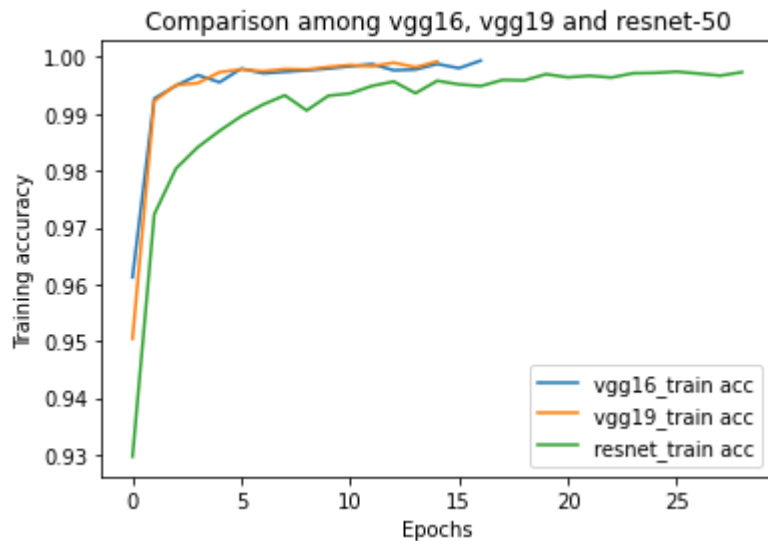


Figure 4.6: Comparison among VGG-16, VGG-19 and RESNET-50 for Bengali CVPR and batch size 64

For batch size 64 and the Devanagari CVPR dataset, figure 4.7 depicts a comparison of the

systems that provided the highest testing accuracy for the VGG-16, VGG-19, and RESNET-50 architectures.

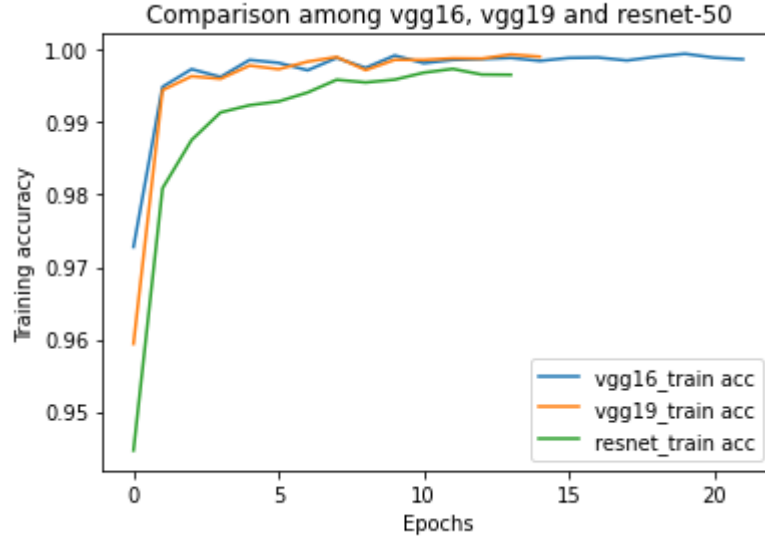


Figure 4.7: Comparison among VGG-16, VGG-19 and RESNET-50 for Devanagari CVPR and batch size 64

Table 4.6: Confusion matrix for test samples of handwritten Bengali numeral dataset.

Bengali Numeral	Number of samples classified as									
	০	১	২	৩	৪	৫	৬	৭	৮	৯
০	399	০	০	১	০	০	০	০	০	০
১	০	399	০	০	০	০	০	০	০	১
২	০	০	400	০	০	০	০	০	০	০
৩	০	০	০	400	০	০	০	০	০	০
৪	০	০	০	০	400	০	০	০	০	০
৫	১	১	০	০	২	396	০	০	০	০
৬	০	০	০	০	০	০	400	০	০	০
৭	০	০	০	০	১	০	০	399	০	০
৮	০	০	০	০	০	০	০	০	400	০
৯	০	৩	০	১	০	০	০	০	০	396

Table 4.7: Confusion matrix for test samples of handwritten Devanagari numeral dataset.













Devanagari Numeral	Number of samples classified as									
	०	१	२	३	४	५	६	७	८	९
०	375	0	0	0	0	0	0	0	0	0
१	0	370	1	0	0	0	0	2	2	0
२	0	0	375	0	0	0	0	0	0	0
३	0	0	1	373	0	0	0	1	0	0
४	1	0	0	0	374	0	0	0	0	0
५	0	0	0	0	0	375	0	0	0	0
६	0	1	0	0	0	0	372	0	0	2
७	0	0	0	0	0	0	0	375	0	0
८	0	0	1	0	0	0	0	0	374	0
९	0	0	0	0	0	0	1	0	1	373

The suggested system's misclassified handwritten numerical pictures are shown in Table 4.8. Because of the enormous variance in writing patterns, these pictures are challenging to correctly detect even for humans, as seen in the table. All 13 misclassified photos were judged to be ambiguous, indicating that the system's categorization is acceptable.

First image prediction as “8” seems accurate considering wrong placement in category in “5”. Similarly, second image classification as “6” rather than “4” is logical as the image is more closed to “6” due to incompleteness and rotation. Similar reason is also considerable for forth image to classify as “4” though it is in category “6” in the dataset. The third image is so ambiguous to place in any numeral category even by human; therefore, its printed form reconstruction is not clear and finally misclassified. Misclassification of rest nine images of category “9” as “5” are also acceptable due to structural similarity between the two. Considering above explanations for all 13 misclassifications (out of 4000 test

images), the proposed system based on the novel idea of handwritten numeral images superposition onto printed form is found a best suited method for HNR.

Table 4.8: Sample misclassified handwritten Bengali (a) and Devanagari (b) numerals by HNRUTL.

Handwritten Bengali numeral image	Predicted value	Actual value	Handwritten Devanagari numeral image	Predicted value	Actual value
	৩	০		७	१
	৪	০		০	४
	২	১		৭	৩
	৬	৩		৭	১
	৫	৬		৮	১
	৬	৫		২	৩

(a)

(b)

Table 4.6 and 4.7 are showing the confusion matrix of respectively the Bengali and Devanagari language. Here we can see, the system for the Bengali language has misclassified a total of 11 numbers and provides the testing accuracy 99.73%. On the other hand, the system for the Devanagari language has misclassified a total of 14 numbers and provides the testing accuracy of 99.63%. The Bengali system misinterpreted between 1 and 9, 0 and 9, 0 and 3, 5 and 6. The Devanagari system misinterpreted between 1 and 7, 2 and 3, 9 and 1, 0 and 4.

4.5 Comparison with the Existing Methods

Table 4.9 and 4.10 shows a comparative description of proposed HNR with some prominent methods for Bengali and Devanagari handwritten numerals. From table 4.9 you can see that our proposed model gains much better accuracy than other prominent methods

with the Bengali dataset. From table 4.10 you can see that our proposed model gains much better accuracy than other prominent methods with the Devanagari dataset.

Table 4.9: A comparative description of proposed HNR with some prominent methods for Bengali handwritten numerals.

Work Reference and Year	Feature Selection	Method	Dataset; Training and Test Samples	Recognition Accuracy
Basu et al. [19], 2005	Shadow feature and Centroid feature	MLP with Dempster-Shafter techniques	Samples from CVPR, ISI [18]; 4000 and 2000	95.10%
Pal et al. [6], 2006	Feature based on reservoir and water overflow	Binary decision tree	Self-prepared, 12000	92.80%
Wen et al. [7], 2007	Principal component analysis (PCA) and Kernel PCA	SVM	Bangladesh postal system; 6000 and 10000	95.05%
Wen and He [8], 2012	Eigenvalues and eigenvectors	Kernel and Bayesian discriminant	Bangladesh postal system; 30000 and 15000	96.91%
Nasir and Uddin [12], 2013	Bayes' theorem, K-means clustering and Maximum Posteriori	SVM	Self-prepared, 300	96.80%
Akhand et al. [9], 2016	No	CNN	CVPR, ISI [18]; 18000 and 4000	98.45%
Mahtab et al. [11], 2016	No	SAE	CVPR, ISI [18]; 18000 and 4000	96.30%
Akhand et al. [10], 2018	No	CNN	CVPR, ISI [18]; 18000 and 4000	98.98%
Shuvo et al. [2], 2019	No	CAE with CNN	CVPR, ISI [18]; 18000 and 4000	99.68%
Proposed Method	No	TL with VGG-16 using FT, BS 32	10 CVPR, ISI [18]; 54000 and 4000	99.73%

From both of the table 4.9 and 4.10, we can see that our technique provides the best accuracy. Until now it has the highest accuracy. For the Bengali language it provides 99.73% and the Devanagari language provides 99.63%.

Other accuracies are shown in the above table 4.2 4.3 4.4 and 4.5. Models except for the model that has provided the best accuracy, some of them exceed the traditional models. But those are not considered or compared anyhow.

Table 4.10: A comparative description of proposed HNR with some prominent methods for Devanagari handwritten numerals.

Work Reference and Year	Method	Dataset, Training and Test Samples	Recognition Accuracy
Arya et al. [27], 2015	SVM and KNN	CVPR, ISI [18]; 19798 and 3763	98.06 %
Singh et al. [28], 2015	Ensemble of NNs	CVPR, ISI [18]; 16794 and 3762	99.37 %
Singh et al. [29], 2016	MLP	CMATERdb 3.2.1 [27]; 2000 and 1000	98.92 %
Singh et al. [33], 2017	MLP	CMATERdb 3.2.1 [27]; 2000 and 1000	97.50 %
Akhand et al. [9], 2018	CNN with data augmentation	CVPR, ISI [18]; 18000 and 3763	98.96 %
Trivedi et al. [30], 2018	CNN with GA	CVPR, ISI [18]; 18784 and 3762	96.41%
Jiang et al. [31], 2020	Edge-TripleNet	CMATERdb [27]; 2400 and 600	97.33%
Gupta et al. [32], 2021	Transfer Learning (MNIST Pre-trained)	CVPR, ISI [18]; 15789 and 2256	99.04%
Proposed Method	TL with VGG-19 using FT, BS 64	10 CVPR, ISI [18]; 54000 and 3750	99.63 %

CHAPTER V

Conclusions

Handwritten Numeral Recognition technique is a complex and convoluted task. If it is a talk about dimension then it can be said that it deals with the tasks that are of higher dimension. Different people write in different ways. They do not use the same pattern while writing. While using the classifier to classify the numerals that people generally write in different ways, complexity arises. When a numeral is seen from a different angle, distortion occurs. Distortion happens when the pattern of an image is rotated. A novel Handwritten Numeral Recognition system is introduced in this thesis that follows an underlying hypothesis. The hypothesis is how humans learn writing. In this chapter, a conclusion will be drawn like a short passage of some starred points of this work and some directions on the basis of the outcome of the related works will be mentioned for how to do future research.

5.1 Achievements

The actual goal of this thesis work is about building up a novel Handwritten Numeral Recognition system that is on the basis of an underlying hypothesis which is how humans learn writing. The hypothesis says that people are familiar with the printed form of numerals on which they practice writing numerals. The only challenge was distortion. If an image can be seen from a different angle, the pattern will also be changed. That is why, while training, the proposed system is fed the dataset on which a certain degree of rotational augmentation is applied. Hence, the challenge is overcome. Thus the effect of distortion is attenuated. The system has also reduced the computational complexity. The traditional CNN models used to take a lot of time while training. This system uses transfer learning. Transfer learning does not train all the CNN layers. It only trains the dense layers which follow the flattened layers. Since it uses a pre-trained model's weight, it takes fewer epochs to reach convergence. When it is a matter of finding a new dataset or a huge number of instances is required, transfer learning works fine in such cases.

5.2 Future Perspectives

From this study, there are some possible, potential future avocations of works. The proposed system tests the numerals of the Bengali and Devanagari language. Compared to the other languages, these two languages gave relatively low accuracy because of its high-dimension and various forms of writing pattern. Here only three architectures have been introduced. They are VGG-16, VGG-19 and RESNET-50. Though these three give a better accuracy than the other related systems, it might be improved if other architectures were tried. In addition, 32x32 input images are used to train the system. A better result may be found if other sizes of images are tried. If proper study is done on this concept, it seems that there exist a huge number of combinations. Hence, required modification might get a better result.

REFERENCES

- [1] A. K. Tushar, A. Ashiquzzaman, A. Afrin and M. R. Islam, "A novel transfer learning approach upon hindi, arabic and bangla numerals using convolutional neural networks," in *Computational Vision an Bio Inspired Computing*. Springer, 2018, pp. 9972-981
- [2] M. I. R. Shuvo, M. A. H. Akhand, and N. Siddique, "Handwritten Numeral Superposition to Printed Form Using Convolutional Auto-encoder and Recognition Using Convolutional Neural Network", *International Joint Conference on Computational Intelligence (IJCCI)*, Dhaka, Bangladesh, 2019. (Accepted).
- [3] D. C. Ciresan, U. Meier and J. Schmidhuber, "Transfer learning for latin and Chinese characters with deep neural networks," in the 2012 International Joint Conference on Neural Networks (IJCNN). IEEE, 2012, pp. 1-6
- [4] Sakib Reza, Ohinda Binte Amin and M. M. A. Hashem "Basic to Compound: A Novel Transfer Learning Approach for Bengali Handwritten Character Recognition" in 2nd International Conference on Bangla Speech and Language Processing (ICBSLP), September 2019
- [5] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 3320–3328.
- [6] U. Pal, B. B. Chaudhuri, and A. Belaid, "A Complete System for Bangla Handwritten Numeral Recognition," *IETE J. Res.*, vol. 52, no. 1, pp. 27–34, Jan. 2006.
- [7] Y. Wen, Y. Lu, and P. Shi, "Handwritten Bangla numeral recognition system and its application to postal automation," *Pattern Recognit.*, vol. 40, no. 1, pp. 99–107, Jan. 2007.
- [8] Y. Wen and L. He, "A classifier for Bangla handwritten numeral recognition," *Expert Syst. Appl.*, vol. 39, no. 1, pp. 948–953, Jan. 2012.
- [9] M. M. H. Akhand, M. A. H.; Ahmed, Mahtab; Rahman, "Convolutional Neural Network based Handwritten Bengali and Bengali-English Mixed Numeral Recognition," *Int. J. Image, Graph. Signal Process.*, vol. 8, no. 9, pp. 40–50, 2016.
- [10] M. A. H. Akhand, M. Ahmed, M. M. H. Rahman, and M. M. Islam, "Convolutional Neural Network Training incorporating Rotation-Based Generated Patterns and Handwritten Numeral Recognition of Major Indian Scripts," *IETE J. Res.*, vol. 64, no. 2, pp. 176–194, Mar. 2018.
- [11] M. Ahmed, A. K. Paul, and M. A. H. Akhand, "Stacked auto encoder training incorporating printed text data for handwritten bangla numeral recognition," in *2016*

19th International Conference on Computer and Information Technology (ICCIT), 2016, pp. 437–442.

- [12] M. K. Nasir, “Hand Written Bangla Numerals Recognition for Automated Postal System,” *IOSR J. Comput. Eng.*, vol. 8, no. 6, pp. 43–48, 2013.
- [13] N. Ilmi, W. T. A. Budi, and R. K. Nur, “Handwriting digit recognition using local binary pattern variance and K-Nearest Neighbor classification,” in *2016 4th International Conference on Information and Communication Technology (ICoICT)*, 2016, pp. 1–5.
- [14] F. Lauer, C. Y. Suen, and G. Bloch, “A trainable feature extractor for handwritten digit recognition,” *Pattern Recognit.*, vol. 40, no. 6, pp. 1816–1824, Jun. 2007.
- [15] L. LI, L. ZHANG, and J. SU, “Handwritten character recognition via direction string and nearest neighbor matching,” *J. China Univ. Posts Telecommun.*, vol. 19, no. 2, pp. 160–196, Oct. 2012.
- [16] M. Das, Nibaran; Pramanik, Sandip; Basu, Subhadip; Saha, Punam; Sarkar, Ram ; Kundu, Mahantapas; Nasipuri, “Recognition of Handwritten Bangla Basic Characters and Digits using Convex Hull based Feature Set,” in *International Conference on Artificial Intelligence and Pattern Recognition (AIPR-09)*, 2009, pp. 380–386.
- [17] S. M. Shamim, M. ; Miah, Badrul Alam Miah, A. Sarker, and A. Al Rana, Masud; Jobair, “Handwritten Digit Recognition using Machine Learning Algorithms,” *Glob. J. Comput. Sci. Technol. D Neural Artif. Intell.*, vol. 18, no. 1, pp. 17–23, 2018.
- [18] B. . Bhattacharya, U.; Chaudhuri, “Offline Handwritten Bangla and Devanagari Numeral Databases. Kolkata: Computer Vision and Pattern Recognition Unit, Indian Statistical Institute.” 2009.
- [19] R. Plamondon and S. N. Srihari, “Online and off-line handwriting recognition: a comprehensive survey,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 1, pp. 63– 84, 2000.
- [20] K. Labusch, E. Barth, and T. Martinetz, “Simple Method for High-Performance Digit Recognition Based on Sparse Coding,” *IEEE Trans. Neural Networks*, vol. 19, no. 11, pp. 1985–1989, Nov. 2008.
- [21] S. Belongie, J. Malik, and J. Puzicha, “Shape matching and object recognition using shape contexts,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 4, pp. 509–522, Apr. 2002.
- [22] T. Kutzner, M. Dietze, I. Bonninger, C. M. Travieso, M. K. Dutta, and A. Singh, “Online handwriting verification with safe password and increasing number of features,” in *2016 3rd International Conference on Signal Processing and Integrated Networks (SPIN)*, 2016, pp. 650–655.
- [23] B. Fallah and H. Khotanlou, “Identify human personality parameters based on

handwriting using neural network,” in 2016 Artificial Intelligence and Robotics (IRANOPEN), 2016, pp. 120–126.

- [24] A. Choudhary, R. Rishi, and S. Ahlawat, “Handwritten Numeral Recognition Using Modified BP ANN Structure,” in International Conference on Computer Science and Information Technology, 2011, pp. 56–65.
- [25] M. M. Khan, M. M. R. ; Shah, Md Arifur Rahman ; Alam, “Bangla handwritten digits recognition using evolutionary artificial neural networks,” in 7th International Conference on Computer and Information Technology (ICCIT 2004), 2004, pp. 26–28.
- [26] Y. C. C. C. J. C. B. LeCun, “THE MNIST DATABASE of handwritten digits”.
- [27] S. Arya, I. Chhabra, and G. S. Lehal, “Recognition of Devnagari Numerals using Gabor Filter,” *Indian J. Sci. Technol.*, vol. 8, no. 27, pp. 1–6, Oct. 2015, doi: 10.17485/ijst/2015/v8i27/81856.
- [28] P. SINGH, A. VERMA, and N. S. CHAUDHARI, “Feature selection based classifier combination approach for handwritten Devanagari numeral recognition,” *Sadhana*, vol. 40, no. 6, pp. 1701–1714, Sep. 2015, doi: 10.1007/s12046-015-0419-x.
- [29] P. K. Singh, R. Sarkar, and M. Nasipuri, “A Study of Moment Based Features on Handwritten Digit Recognition,” *Appl. Comput. Intell. Soft Comput.*, vol. 2016, pp. 1–17, 2016, doi: 10.1155/2016/2796863.
- [30] A. Trivedi, S. Srivastava, A. Mishra, A. Shukla, and R. Tiwari, “Hybrid evolutionary approach for Devanagari handwritten numeral recognition using Convolutional Neural Network,” *Procedia Comput. Sci.*, vol. 125, pp. 525–532, 2018, doi: 10.1016/j.procs.2017.12.068.
- [31] W. JIANG and L. ZHANG, “Edge-SiamNet and Edge-TripleNet: New Deep Learning Models for Handwritten Numeral Recognition,” *IEICE Trans. Inf. Syst.*, vol. E103.D, no. 3, pp. 720–723, Mar. 2020, doi: 10.1587/transinf.2019EDL8199.
- [32] D. Gupta and S. Bag, “CNN-based multilingual handwritten numeral recognition: A fusion-free approach,” *Expert Syst. 26 Appl.*, vol. 165, p. 113784, Mar. 2021, doi: 10.1016/j.eswa.2020.113784.
- [33] M. A. H. Akhand, *Deep Learning Fundamentals - A Practical Approach to Understanding Deep Learning Methods*. Dhaka: University Grants Commission of Bangladesh, 2021.