ISLAMIC UNIVERSITY OF TECHNOLOGY (IUT)

A Simplistic & Fast approach to Bangla Handwritten Digit Recognition using Convolutional Neural Network

Supervisor

Dr. Hasanul Kabir

Professor

Dept. of CSE, IUT

Co-Supervisor

Sabbir Ahmed

Lecturer

Dept. of CSE, IUT

A thesis submitted in partial fulfilment of the requirements

for the degree of B. Sc. Engineering in Computer Science and Engineering

Academic Year: 2019-2020

Department of Computer Science and Engineering (CSE)

Islamic University of Technology (IUT)

A Subsidiary Organ of the Organization of Islamic Cooperation (OIC)

Dhaka, Bangladesh

March 22, 2021

Declaration of Authorship

This is to certify that the work presented in this thesis is the outcome of the analysis and experiments carried out by under the supervision of Dr. Hasanul Kabir, Professor of the Department of Computer Science and Engineering (CSE), Islamic University of Technology (IUT), Dhaka, Bangladesh. It is also declared that neither of this thesis nor any part of this thesis has been submitted anywhere else for any degree or diploma. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given.

Authors:

Eksan Ahmed Emon

____

Student ID - 160041067

Ifta Khairul Adil

____

Student ID - 160041034

Abir Ahbab

____

Student ID - 160041032

Approved By:

Supervisor:

Dr.Hasanul Kabir, PhD

Professor

Department of Computer Science and Engineering (CSE)

Islamic University of Technology (IUT), OIC

Co-Supervisor:

[illegible] 29.03.2021

Sabbir Ahmed

Lecturer

Department of Computer Science and Engineering (CSE)

Islamic University of Technology (IUT), OIC

Acknowledgement

We would like to express our grateful appreciation for **Professor Dr. Hasanul Kabir**, Department of Computer Science & Engineering, IUT for being our adviser and mentor. His motivation, suggestions and insights for this research have been invaluable. Without his support and proper guidance this research would never have been possible. His valuable opinion, time and input provided throughout the thesis work, from first phase of thesis topics introduction, subject selection, proposing algorithm, modification till the project implementation and finalization which helped us to do our thesis work in proper way. We are really grateful to him.

We are also grateful to **Sabbir Ahmed**, Lecturer, Department of Computer Science & Engineering, IUT for his valuable inspection and suggestions on our proposal of Bangla Handwritten Digit Recognition using Deep Learning method.

Abstract

Handwritten digit recognition has consistently a major test because of its variety of shape, size, and composing style.Accurate Handwritten Digit Recognition is becoming challenging and thoughtful to researchers due to its educational and economic values.Most of the To benchmark Bengali digit acknowledgment calculations, a huge openly accessible dataset is required which is liberated from inclinations starting from topographical area, sexual orientation, and age.In light of this point, NumtaDB, a dataset comprising of something else than 85,000 pictures of transcribed Bengali digits, has been amassed.The challenges of NumtaDB data set is that it contains unbiased, unprocessed and augmented images.So for this reason different kinds preprocessing steps were followed to process the available data and A simplistic fast approach to Bangla Handwritten digit recognition using Convolution Neural Network is proposed.

Keywords — Convolutional Neural Network, handwritten digit recognition, Bangla handwritten digits, Image Preprocessing.

Contents

1 Introduction

1.1 Overview

The ability of PCs to recognize human-transcribed digits is known as manually written digit acknowledgment. It's a difficult task for the computer, given that manually written digits aren't particularly appealing and can be rendered in a variety of flavors. The solution to this problem is written by hand digit acknowledgement, which uses an image of a digit and recognizes the digit present in the picture. Because of the variation of form, scale, and composing style, transcribed digit acknowledgment has always been a big challenge. Experts are becoming more interested in precise transcribed recognition as a result of its instructive and monetary value.. There had a few works been as of now done on the Bangla Manually written Acknowledgment, yet at the same time there is no vigorous model grown at this point.

Manually written digit acknowledgment improvement research is quickly advancing and reshaping the must robotization fields like—programmed check perusing, programmed number plate perusing, computerized postal assistance, Optical Picture Perceiving (OCR), and so forth Because of its different parts of employments, PC vision specialists verily feel to chip away at it and improve—quality and execution to be sure. Be that as it may, manually written perceiving is more difficult contrasted with the composed letter. Since various individuals write in an alternate manner and which makes a more significant level of fluctuation in composed style. Likewise, there are a few likenesses between various characters shape. The circumstance of overwriting makes it more trying for precisely grouping the transcribed digit.Nowadays, profound learning, particularly the Convolutional neural organization is working better in the reason for characterizing these sorts of acknowledgment work rather other AI techniques.

The new achievement of profound learning, particularly Convolutional Neural Organization (CNN) for PC vision has motivated numerous analysts to utilize the

CNN to perceive manually written characters and digits as a PC vision task.

A great deal of work has been done as of late on this field. Numerous models have been proposed and showed capable outcome in recognizing transcribed digits. However, a large portion of them has countless boundaries and furthermore sets aside tremendous measure of effort for preparing and anticipating information. Additionally the design of these models are extremely perplexing. So our primary concentration for theory was to thought of a model which will be extremely straightforward yet powerful to distinguish the written by hand digits. We are proposing a straightforward CNN model which produces comparable exactness like other complex model just as has less number of boundaries and burns-through less measure of time.

1.2 Problem Statement

Over the most recent couple of many years, the essential strategy for putting away data has changed from transcribed duplicates of archives to advanced organizations. The advanced configuration of the reports are more dependable and simple to store. Regardless of the switchover to the new type of report stockpiling, an enormous portion of the more established reports are put away in manually written structure. The issue lies in the way toward changing over these archives, where conventional strategies depend on physically composing the entire writing. This measure is repetitive and requires an enormous measure of time to effectively convert the reports, and furthermore requires significant measure of labor to make exact duplicates of the reports. Moreover, Bangla characters have a complex plan of shapes which are more confounded than that of different dialects, for example, English or German. One of the most testing errands in transcribed character order is, managing the tremendous assortment of penmanship styles by various individuals.

The point of convergence of this examination lies on a crucial issue of penmanship acknowledgment, which is, acknowledgment of person characters/letter sets of the Bangla language. By giving a strategy for singular character acknowledgment, this examination opens up the way for additional advancement in the Bangla penmanship acknowledgment area. At the point when joined with suitable picture preparing procedures, Bangla penmanship acknowledgment frameworks may have some commonsense use cases, for example, arranging addresses composed on letters at postal workplaces, transcribed check acknowledgment in the financial area and so on.

1.3 Motivation and scope of research

Bangla hand written digit acknowledgment is a traditional issue in the field of computer vision. There are different sorts of pragmatic utilization of this framework, for example, OCR, postal code acknowledgment, tag acknowledgment, bank checks acknowledgment and so on [1]. Perceiving Bangla digit from reports is turning out to be more significant [2]. The remarkable number of Bangla digits are absolute 10. So the acknowledgment task is to arrange 10 unique classes. The basic assignment of manually written digit acknowledgment is perceiving special transcribed digits. Since each human has his own composition styles. Be that as it may, our commitment is for the really testing task. The difficult errand is tied in with getting hearty execution and high precision for huge, fair-minded, natural, and profoundly enlarged NumtaDB [3] dataset. The dataset contains obscuring, commotion, pivot, interpretation, shear, zooming, tallness/width step, brilliance, distinction, impediments, and superimposition, and was compiled from various sources and at various times.

Although different models have been developed for handwritten digit recognition previously but they were mainly implemented on some particular datasets like MNIST, EMNIST, CMATERdb etc. NumtaDB dataset was published a few days ago and it contains a huge number of data with variation. So our main focus is to implement NumtaDB dataset on our deep CNN model to produce a good outcome.

To simply put the motivation of our work, we can say-

- Firstly,In the field of computer vision, handwritten digit recognition in Bangla is a classic problem.
- Secondly, Publicly available large and unbiased datasets were missing.But NumtaDB dataset is a now available and has a huge number of data with different variety.
- Thirdly, we want to make a simple and fast CNN model which will be able to recognize bangla handwritten digit.
- Finally, It will be simpler and consume less amount of time than the existing model.

1.4 Research Challenges

The main challenging task was to find a suitable and unbiased dataset for our model. Our primary focus was to train our model with large unbiased dataset so that it does not overfit or underfit. For our thesis we chose NumtaDB data set which contains more than 85000 data. There are no inequalities in this dataset due to geographical region, sex, or age.We also performed preprocessing steps like augmentation, normalization, binarization. Another Challenge for our thesis was to find the best hyper parameters for our model like batch size, learning rate ,number of epochs and dividing the dataset into training ,validation and testing set.

1.5 Thesis Outline

In Chapter 1 we have discussed our study in a precise and concise manner. Chapter 2 deals with the necessary literature review for our study and there development so far. In Chapter 3 we have stated the skeleton of our proposed method, proposed algorithm and also the flowchart to provide a detail insight of the working procedure of our proposed method, **A simplistic fast approach to Bangla Handwritten digit recognition using Convolution Neural Network**. Chapter 4 shows the results and comparative analysis of successful implementation of our proposed method. The final segment of this study contains all the references and credits used.

2 Background Study

2.1 Literature Review

Handwritten digit recognition is a large working field in every language. There are several research works based on Bangla handwritten digit recognition using deep learning. Most of their work is very appreciative. They set a high standard for future researchers to work, implement and make improvements to the existing models.

Bangla Handwritten Digit Recognition Using Deep CNN was suggested by Ashadullah Shawon et al. [1]. They used the NumtaDB dataset and pre-processed it with techniques like resizing, grayscaling, interpolation, noise reduction, and so on. Their proposed architecture consists of six convolutional layers and two thick layers that are totally connected. The first two layers have 32 channels, each of which is 55 pixels wide. The central two layers have 128 channels, each of which is 33 pixels wide. The final two layers have 256 channels, each of which is 33 pixels wide. Both layers use the Rectified Linear Unit (ReLu) as an initiation work. After each two layers, Maxpooling layers and Bunch standardization are used. The maxpooling layer has a pool capacity of 22. To speed up learning, group standardization is used. To reduce overfitting, a dropout (20%) layer is inserted after the main thick layer. The first of the two fully connected layers has 64 filters, while the second has 10 filters for the ten digits. The classification's final activation mechanism is a softmax function. For the proposed model, they were able to reach 92.72% accuracy on NumtaDB, which is a fantastic score.

Recognition of Bengali Handwritten Digits Using Convolutional Neural Network Architectures was suggested by Mahmudul Hasan et al. [2]. They've also used a variety of pre-processing methods to improve the accuracy of their model. They've gone through trial and error process to find the best possible pre-processing techniques for the model. They've also cleaned and augmented data from NumtaDB dataset to make a new dataset that consists of even more images for their model. ResNet is able to train deep networks with hundreds or even thousands of layers and accomplishes fascinating performance with the idea of 'identity shortcut connection'. As such, they picked two different ResNets to train six different models. They've used four ResNet34 and a ResNet50 and ensemble them to get the final result. They've used weighted voting from the individual model predictions to get the final prediction. For weights of a model, they used the normalized form of their validation accuracy. They used even number of models instead of odd numbers because even number ensembles are less likely to predict wrong on a particular data due to wrong predicted models having higher weights for a particular data. Their model got around 99.34% accuracy.

Rouhan Noor et al. [3] suggested digit recognition method Using Ensembling of Convolutional Neural Network. In this method, they've experimented with different CNN models and taken two of their best-performing models (Model A and Model B) for ensembling. Model A is the deepest and best-performing model among all with 96.333% accuracy. The output layers of the two models are averaged for the final output. The first convolutional layers consist of 32 filters with 5x5 kernel size generally extract low-level features like vertical horizontal edges at greater extend followed by second layers consist of 32 filters with 3x3 kernel size in both models. After that, Maxpool layers with 2x2 kernel size and strides of 2 are employed to reduce the features by taking maximum value which greatly cut the computation curve and overfitting. Similarly, two convolutional layers consist of 64 filters each with 3x3 kernel size, and Maxpool layers are added similar to the previous Maxpool layer's configuration. Experimenting with different configurations, they have eventually found that, using a slightly wider convolutional layer at the end of Model A offers a slight accuracy boost in Model A. So, instead of deploying 3 same layers consist of 512 filters with 3x3 kernel, They used a bit wider layer consists of 521 filters, and then two layers consist of 512 filters each in

Model A. Rectified Linear Unit (ReLu) activation function is used in every layer including fully connected (FC) layers except the final FC layers in both models. The convolution feature maps are flattened and connected with FC layer with 64 neurons. Dropout layers are added before the final FC layers with the value of 0.2 which randomly cut off the weights of some neurons for preventing overfitting problems in both models. Finally, FC layers of 10 neurons are added with Softmax activation function for classification of 10 classes and ensembled the models by averaging the final output layers. The same padding configuration is used in all convolutional layers in both models. This model performs with around 96% accuracy.

Md Zahangir Alom et al [4] suggested a CNN model which was tested on the CMATERdb where, the CNN with global features with dropout yields the best accuracy. Bishwajit Purkaystha et al [5] also suggested a CNN model to find begali handwritten character recognition, which yields around 89.93% accuracy, proving that a CNN is effective for recognizing handwritten characters. Rabby et al [6] presented a CNN model to find better accuracy with less computation time and with fewer epochs compared to other models. It gives good accuracy for ISI dataset and CMATERDB dataset, but unfortunately they couldn't test it on the NumtaDB dataset.

A digit recognition Approach with an Ensemble of Deep Residual Networks was suggested by Mamunur Rahaman Mamun et al. [7]. They've also experimented with different pre-processing methods in order to improve their model's efficiency. To train their model, they used an Xception network. They selected ResNets because they assume that assembling shallow networks improves results. Xception is known for achieving cutting-edge efficiency with the fewest possible criteria. On the ImageNet dataset, it has a best 1 exactness of 0.790 and a top-5 accuracy of 0.945, with just 22 million boundaries having a depth of 126 layers. As grayscale images were used, the number of input channels was changed. Input resolution of 71×71 was used which is the least possible resolution for Xception architecture. They also trained images of higher resolution (139×139) with the same hyper-parameter set but it didn't help to increase accuracy significantly. The model was also trained with pre-trained weights but it suffered from overfitting with a validation accuracy of around 73%. To address the problem of overfitting, they used dropout layers and regularization. The dropout rate was 25% and it was applied right before every residual layer. L2 regularization was applied to residual layers and to the input layer. The addition of dropout layers compelled them to train the model from scratch without using pre-trained weights. Then they ensemble the model, which also achieved good precision, reaching an accuracy of 98.35%.

Ovi Paul [8] suggested various pre-processing techniques that can be applied to make the model perform better. Using the techniques, the models got a good amount of accuracy from the same dataset, which can be improved by tweaking the models.

Hasib Zunair et al. [9] worked with a VGG16 architecture in their proposed research. VGG16 architecture achieved the state of the art accuracy in the ImageNet Challenge where a model had to classify around 1,000 different object categories in context. The model was trained with more than a million images that a person can see from day-to-day activity – mostly. One major drawback of the VGG16 is that it is slow and the model weight itself is large. But still, it is used for most image classification tasks. Since the pre-trained weights of the model are easily accessible; it has been used for different classification tasks using data set which are not in the distribution of its input. In this paper, a VGG16 model which was pre-trained on Imagenet data was used for transfer learning. The last layer of the existing VGG16 model was discarded and replaced with a Fully Connected layer of 10 neurons with the Softmax activation function. The model is trained using backpropagation and Adadelta is used as the optimizer to maximize the loss function. The loss Categorical Cross entropy is used to compile the model.

Sharif Amit Kamran et al.[10] presented various models and their comparisons which helped with understanding the current models and their performances.

Samiul Alam et al. [11] was used as a base dataset for evaluating their model as it had a lot of images and a lot of variety, which was a very rich dataset. Aminul Islam et al. [12] also released a benchmark for the currently used models like CNN, MLP, SVM, RF, and KNN and provided an analysis on which model performs the best among these and proves that CNN performs much better than other methods. CNN algorithm could focus on the local patterns, and for that reason, CNN was able to extract important features of an image more easily than a classic multilayer perceptron. One of the core building blocks of CNN is the convolution layer which identified the low-level features. In the model, max-pooling was used to reduce the dimensionality of the feature and controlling the overfitting. Chandrika Saha et al. [13] used CMATERdb dataset which was first trained and tested on a LeNet5 model and then their proposed model which was seven-layered CNN giving 97.6% accuracy, better accuracy than the 5 layered LeNet. Talha et al. [14] used a KNN model with GDP and LDP to recognize the Bangla Handwritten Numeral character using directional Pattern. In this paper, an ensemble method containing the directional pattern is proposed. The feature extraction process is performed through the micro-pattern of LDP and GDP. Then KNN and SVM classifiers are used to generate the ensemble approach. Finally, the maximum voted class is considered for the predicted class of the test sample. The overall process had two steps: features extraction through directional pattern and ensemble approach for classification and recognition, getting around 95.62% accuracy in the end. Golam Rabbani et al.[15] suggested bangladeshi license plate detection and recognition with Morphological Operation and Convolution Neural Network, where they detected the license plate, segmented the characters and recognized them using a convolutional neural network, getting an overall accuracy of 95.75%. Abu Sufian et al [16] proposed a CNN network that performs very well on the ISI dataset, but it takes a lot of time and epochs to reach high accuracy on that dataset.

Rabeya Basri et al. [17] investigates the performance of some state-of-the-art deep CNN techniques for the recognition of handwritten digits. It considers four deep CNN architectures, such as AlexNet, MobileNet, GoogLeNet (Inception V3), and CapsuleNet models. These four deep CNNs have experimented on a large, unbiased, and highly augmented standard dataset, NumtaDB. After investigating the individual performance, it can be observed that among the four models, AlexNet provides the highest recognition accuracy of 99.01% over the normal and augmented data, which can be said the best recognition accuracy compared to previous related works. The implementation is a much-time consuming process, requiring high configurable machine with GPU. The future endeavor for this research is to implement these deep CNN models through a transfer learning approach using an inference rule in order to improve accuracy. Md. Abdullah Al Nasim et al[18] also provided a comparative analysis on Bengali Handwritten Digit recognition where they mentioned that the CNN models are extremely significant for augmented and non augmented images.

Md. Masudur Rahman et al.[19] choose VGG-16 network because of its simplicity and depth. Their dataset contains handwritten images Therefore, the intuition was to basically find some key geometric shapes. Therefore, the 3X3 filters of the VGG-16 network should be enough for their problem. Moreover, VGG-16 has 16 layers -which is deep enough for their dataset for finding sophisticated features. The classification layer was changed and the model was fine-tuned to get the final result. They've tried to attain various things from their research and did get a decent result.

[20] Md. Moklesur Rahman et al They looked at four different models (LeNet-5, ResNet-50, VGG-11, and VGG-16) as well as their proposed VGG-11M model for transcribed Bengali numerals recognition at three different goals: 16 16, 32 32, and 64 64. Despite the addition of bunch standardization and zero-cushioning layers, the proposed VGG-11M has fewer channels in the convolutional layers and neurons in the fully connected layers than VGG-11. These changes to the VGG-11 model result in a drastic reduction in the total number of boundaries from 28, 148, 235 (for VGG-11) to 7, 717, 258 (for VGG-11) (for VGG-11M).However, the VGG-11M model was discovered to be more proficient than the other five CNN

models, and it provides the most drastic precision (99.80%) for the ISI dataset at a target of 32 × 32, which is the best exhibition on the Bengali transcribed numeral recognition system to date. It also has excellent accuracy for the following two datasets (CMATERDB and NUMTADB).

Md. Fahim Sikder [21] proposed a model that recognizes digits after going through various blocks and fully connected layers. Unfortunately, they didn't test it on popular datasets like CMATERdb and NumtaDB. Md Zahangir Alom et al. [22] used CNN network to evaluate the proposed model with other models. This had like 2 million parameters where the other models had 4-8million parameters. They also achieved very good accuracy on the CMATRdb dataset. Yao Zhu et al. [23] proposed a generative model with multistage PCA. Md Nazmul Hoq et al.[24] proposed Bangla Handwritten OCR: an overview of the state of the art classification algorithm with a new dataset, where they used ANN, RF, DT, LDA, SVM, LR, and KNN for character recognition, where ANN performed best. Zinnia Khan Nishat et al. [25] also tried to generate Bangla characters using the conditional generative adversarial network.

Due to its outstanding success in solving linear inseparable problems, Md. Jahid Hasan et al. [26] suggested SVM with Radial base function (RBF) kernel as a classifier for handwritten digit recognition. By mapping low-dimensional input into a higher-dimensional feature space, it finds the best separating hyperplane and performs a strong generalization. Many of the features extracted with CNN were flattened using the 'Global average pooling' function before being used to train the classifier, with one-fifth of the extracted features kept separate for potential use. The hyper-parameters of SVM were optimized using a grid-search algorithm to achieve reasonable classification accuracy. Finally, the classification accuracy of the classifier was predicted using the previously separated features. The proposed CNN+SVM model has over 99% accuracy on the NumtaDB dataset.

Tanvir et al. [27] proposed a DCNN model to recognize Bangla handwritten digits. Each convolutional deep neural network is consisting of two parts, one is the feature extractor part consisting of convolutional layers and the other is the classifier part consisting of dense layers. To train these large parameters, a large database such as NumtaDB was selected where they used regularization techniques to reduce overfitting as well as getting over 99% accuracy on the dataset. Soulib Ghosh et al. [28] suggested a Point-Light Source-based Shadow to detect handwritten numerals. Where Bipul Hossain et al. [29] suggested a CNN to detect sign language. Forrest N. Iandola et al. [30] suggested a new architecture that performs as well as complex networks like AlexNet with 50 times fewer parameters which inspired us to look for simpler models and test them out.

2.2 Architectures

2.2.1 AlexNet

There are eight layers in the design: five convolutional layers and three fully associated layers. However, this isn't what distinguishes AlexNet[31] below are a few of the highlights that are novel approaches to convolutional neural networks:

- **ReLU Nonlinearity:** The use of ReLU(Rectified Linear Unit) Nonlinearity is a key component of the AlexNet. The standard approach for preparing a neural network model was tanh or sigmoid actuation capacities. AlexNet demonstrated that deep CNNs could be prepared much faster using ReLU nonlinearity than using immersing initiation capacities like tanh or sigmoid. AlexNet could achieve a 25% preparation error rate using ReLUs(solid bend) several times faster than an equivalent network using tanh, as seen in the figure below from the article.

- AlexNet takes multi-GPU preparation into account by placing half of the model's neurons on one GPU and the other half on a different GPU. This not only allows for a larger model to be prepared, but it also reduces the time required for planning.

- Max Pooling layers are commonly used to down sample the tensors' width and tallness while maintaining the same depth. The adjacent windows over which the limit is figured cover one another, so Covering Peak Pool layers are similar to Maximum Pool layers. Pooling windows of size 3×3 with a 2 stage between adjoining windows is used by the designers. As compared to using non-covering pooling windows of size 2×2 with a move of 2, which would provide equivalent yield calculations, this covering design of pooling reduced the best 1 mistake rate by 0.4 percent and the top-5 blunder rate by 0.3 percent.

The size of a Neural Network determines its ability to learn, so if one isn't careful, it will try to recall the models in the planning data without comprehending the concept. As a result, the Neural Network will perform admirably on the preparation data, but they will fail to grasp the true concept. It will fail to perform admirably when dealing with fresh and inconspicuous test data. This is referred to as overfitting. AlexNet reduces overfitting using two to three different methods.

- **Data Augmentation:** To make their material more shifted, the developers used a name safeguarding update. They created image representations and level reflections in particular, which increased the planning collection by a factor of 2048. They also used Principle Component Analysis (PCA) on the RGB pixel values to adjust the RGB channel forces, which reduced the best 1 error rate by more than 1%.
- **Dropout:** This approach entails "turning off" neurons with a predetermined probability (for example 50%). This means that each loop uses a different representation of the model's borders, allowing each neuron to have more robust features that can be used for other abnormal neurons. Dropout, on the other hand, increases the time spent preparing for the model's assembly.

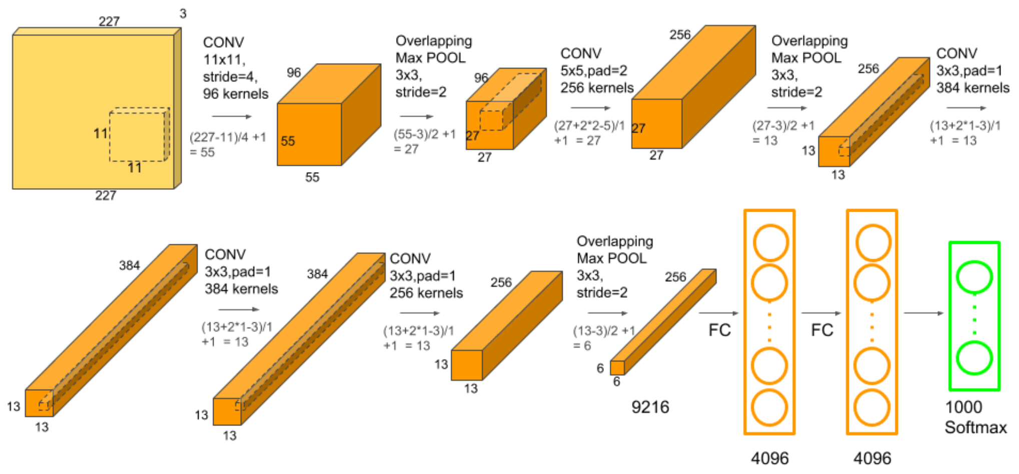


Figure 1: AlexNet Architecture

2.2.2 ResNet

ResNet is a false neural organization (ANN) that draws on constructs identified from pyramidal cells in the cerebral cortex. Skip connections, or simple routes to jump over such layers, are used by leftover neural networks to accomplish this. Average ResNet models are actualized with two- or three-layer avoids in the middle that have nonlinearities (ReLU) and clump standardization. HighwayNets are a form of extra-weight lattice that can be used to familiarize yourself with skip loads. DenseNets are models of a small number of identical skips. A non-leftover network can be thought of as a plain network when it comes to residual neural organizations. Hundreds or thousands of convolutional layers were to be empowered as a result of the engineering. ResNet will connect an immense amount of layers for a strong exhibition, whereas previous CNN models saw a drop off in the feasibility of extra layers. ResNet is a creative solution to the "evaporating slope" problem. Backpropagation interaction, which is based on angle plunge, trains neural networks by lowering the misfortune ability to discover the loads that limit it. When there are a lot of layers, rehashed augmentation makes the angle smaller and smaller until it "vanishes," causing execution to soak or even corrupt with each additional layer."Personality alternative way associations" are used by ResNet to fix this. ResNet stacks character mappings, or textures that don't do much at first, and skirts through them, reusing actuations from previous layers. Skipping the first layer compresses the organization into a few layers, allowing for faster learning. As the network plans again, all layers are expanded, and the "leftover" parts of the enterprise explore a larger portion of the component space of the source image.

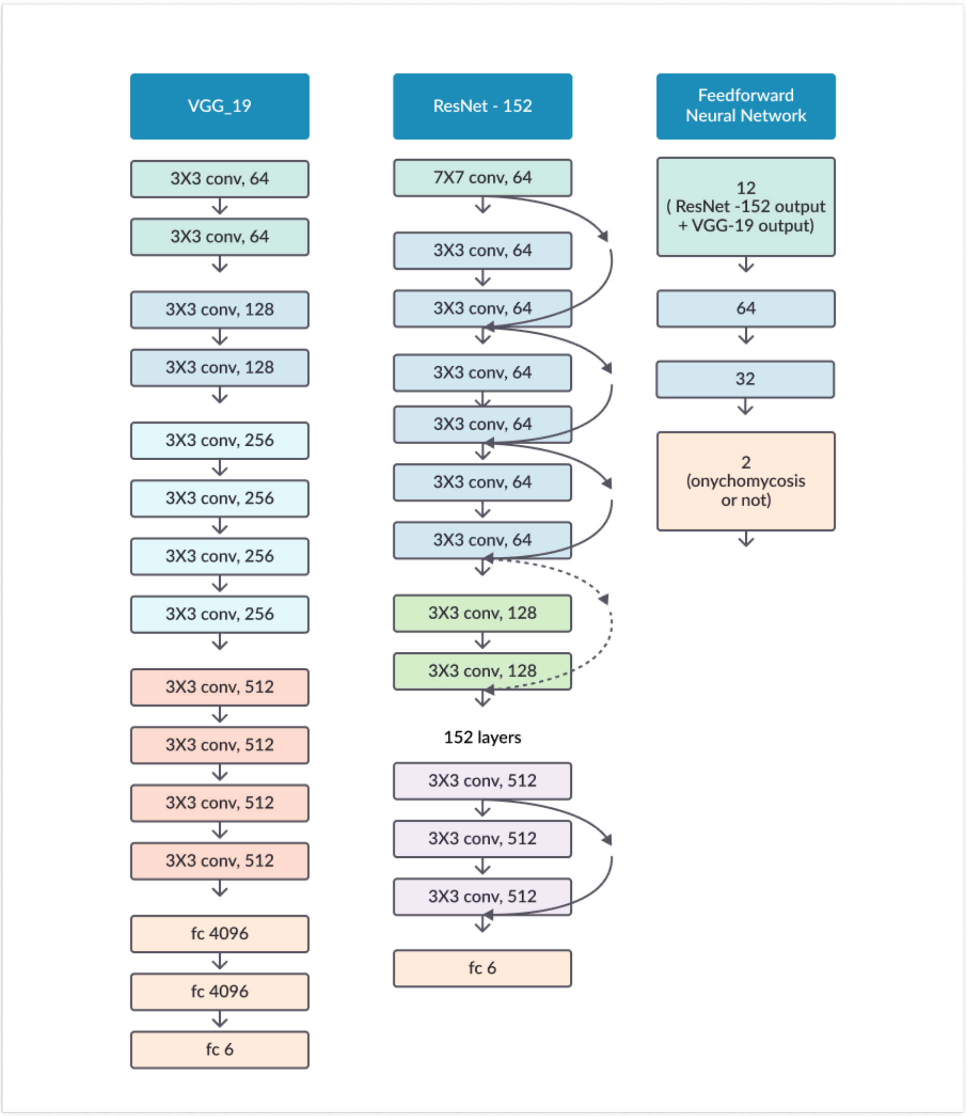


Figure 2: ResNet Architecture

2.2.3 GoogLeNet

The GoogLeNet architecture employs a variety of techniques, such as 1×1 convolution and global average pooling, to allow it to make further design decisions. Any of these methods include:

- **1×1 convolution:** 1×1 convolution is used in the construction of the inception architecture. These convolutions is used to reduce the number of architecture limits (loads and inclinations). By reducing the limits, we can also increase the depth of the architecture.
- **Global Average Pooling :** National average pooling is used at the end of the network in the GoogLeNet architecture.This layer takes a part guide with a value of 7×7 and converts it to a value of 1×1. This also reduces the number of teachable boundaries to 0 and increases top 1 accuracy by 0.6%.
- **Inception Module:** In the Inception module, 1×1, 3×3, 5×5 and 3×3 max-pooling all worked in the same way on the data, and the yields were stacked together to generate the final yield. The idea is that convolution filters of varying sizes can help work with artifacts of various scales.
- **Auxiliary Classifier for Training:** Any transitional classifier branches were used in the engineering of the Inception architecture, and these branches are used throughout the preparation process.A 5×5 standard pooling layer with a phase of 3, an 1×1 convolutions with 128 channels, two fully associated layers of 1024 yields and 1000 yields, and a softmax grouping layer make up these divisions. With a load of 0.3, the resultant depletion of these layers adds up to misfortune. These layers aid in the battle against inclination evaporation and even provide regularization.

Architecture: There are 22 levels in total in the overall architecture. The engineering was created with the intention of remembering numerical efficiency. The idea is that the architecture can be run on single devices with limited computing resources. Two assistant classifier layers are also used in the architecture, and are linked to the yield of the Inception (4a) and Inception (4d) layers.

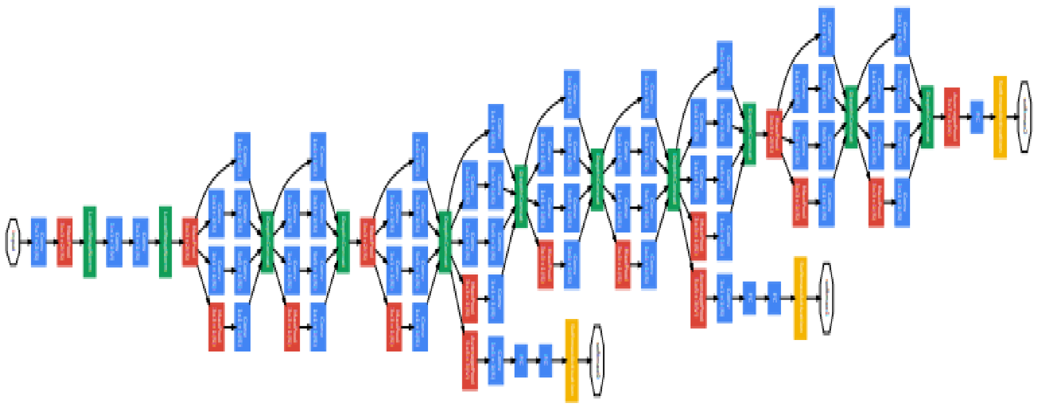


Figure 3: GoogleNet Architecture

2.2.4 CapsuleNet

The CapsNet architecture is made up of two encoders and a decoder, each of three layers. The encoder has three layers: a convolutional layer, a Primary Caps layer, and a Digit Caps layer, whereas the decoder has three layers that are totally connected.

- **Encoder Network:** Two convolutional layers and one fully associated layer make up an encoder. Conv1 is the first convolutional layer, with 256 9×9 convolutional bits and a ReLU actuation work. This layer is in charge of transferring pixel forces to surrounding element finder exercises, which are then handled by the PrimaryCaps layer. The PrimaryCaps layer is a convolutional layer with 32 channels of convolutional 8-D cases (each container has 8 convolutional units with a 9×9 bit and a phase of 2) and 32 channels of convolutional 8-D cases (each container has 8 convolutional units with a 9×9 bit and a step of 2). Essential containers carry out converse designs, which ensures they determine the true image age period.The case adds eight 9×9×256 bits to the 20×20×256 info volume, resulting in a yield tensor of 6×6×8. Due to the fact that there are 32 8-D cases, the yield will be 6×6×8×32. Each class in the DigitCaps layer has 16 D classes, with the low-level case contributing to each case. The weight network used for relative shift against each 8-D case is the 8×16 Wij. The guiding part in question is already present between two container layers on a consistent basis (say, among PrimaryCaps and DigitCaps). Finally, the launch boundaries are encoded using a replication misfortune.For each planning model, the misfortune is calculated against each of the yield groups. The total misfortune is the sum of all the digit containers' misfortunes.

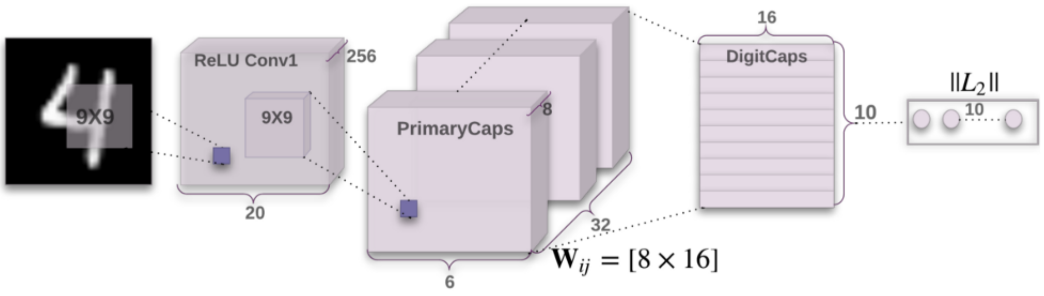


Figure 4: CapsuleNet Encoder Architecture

- **Decoder Network:** The correct 16-D digit container is read by a decoder, which converts it to an image. None of the wrong-digit scenarios was considered. Finding the Euclidean interval between the info picture and the remade picture determines the misfortune. A decoder has three layers that are all connected. The first layer contains 512 neurons, the second layer contains 1024 neurons, and the third layer contains 784 neurons.

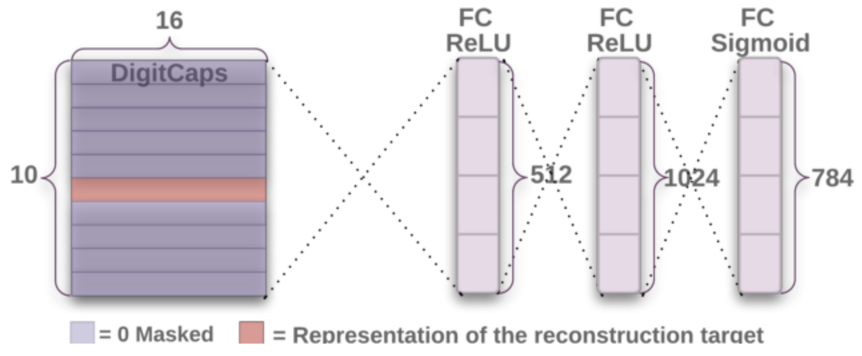


Figure 5: CapsuleNet Decoder Architecture

2.2.5 Xception Net

Xception is a sophisticated convolutional neural network architecture with Depthwise Divisible Convolutions. The data first flows through the segment source, then through the middle stream, which is rehashed many times, and finally through the leave stream. Notice that bunch Standardization follows both Convolution and SeparableConvolution layers. In most old-style grouping problems, Xception engineering outperformed VGG-16, ResNet, and Commencement V3. Xception is a productive network that is based on two main principles:

- Depthwise Separable Convolution
- Shortcuts between Convolution blocks as in ResNet

2.2.6 MobileNet

MobileNet is a smoothed-out architecture that develops lightweight profound convolutional neural organizations using highly informative distinct convolutions and provides an important model for mobile and installed vision applications. MobileNet relies on depthwise distinguishable channels for its construction. Depthwise distinguishable convolution channels combine depthwise and point convolution channels to create depthwise distinguishable convolution channels. The point convolution channel joins the yield of depthwise convolution directly with 1x1 convolutions, while the depthwise convolution channel performs a single convolution for each knowledge channel.

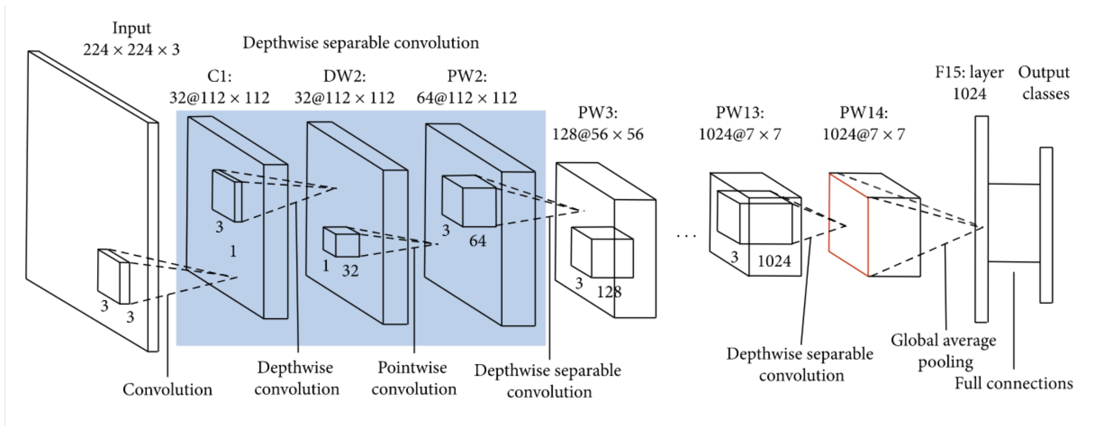
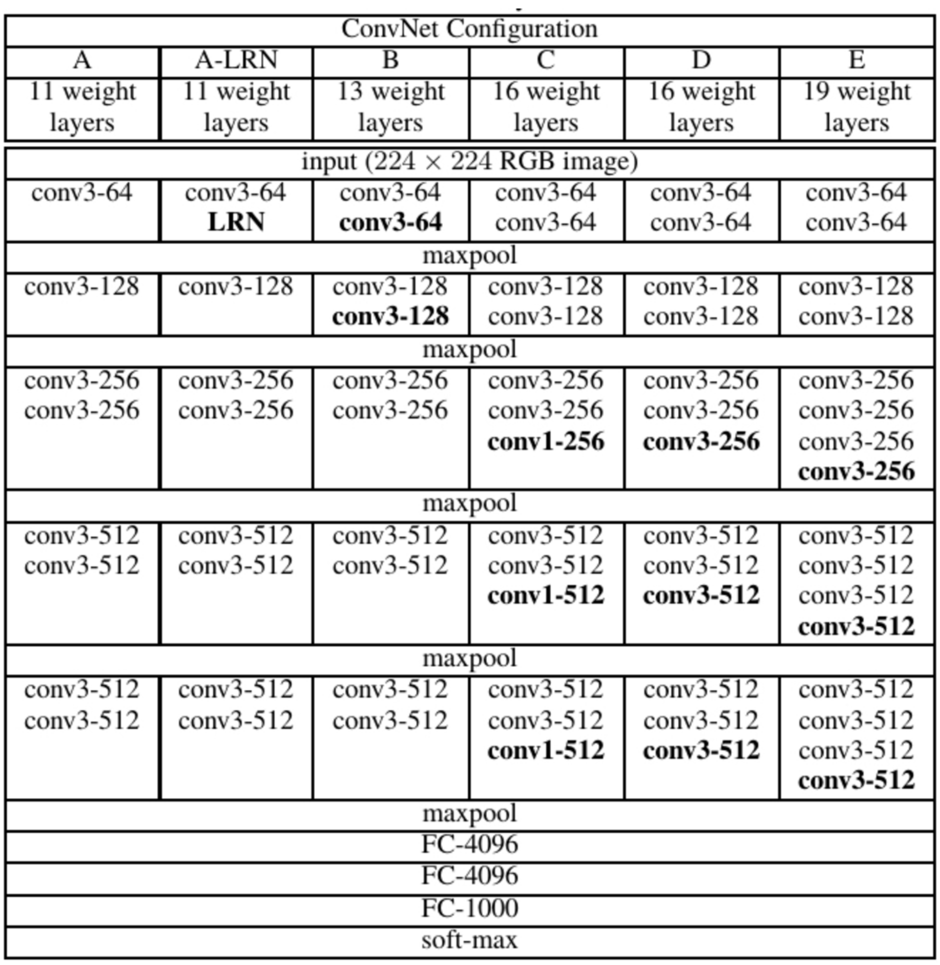


Figure 6: MobileNet Architecture

2.2.7 VGG

The acronym VGG stands for Visual Geometry Group. VGG entered the picture when it relates to the depth of CNNs. It's a pre-built model based on a dataset that has loads that discuss the highlights of the dataset it was built on. When you use a pre-made model, you save time. Effectively, enough time and computation resources have been used to achieve proficiency with a large number of highlights, and the model would most likely benefit from it. The number after the watchword denotes the model's weighted layer count. VGG models use a 224 x 224-pixel image as input; this image should be in RGB format.If the reader wonders why only 224 pixels out of the RGB range of 0 to 255 were chosen to maintain a constant picture size, the answer is simple. Currently, I'm rattling off the layers of the VGG model, which also serve as the foundation for VGG-11.

Convolutional Layer: The kernel size is 3 x 3 in this case. The size of the convolutional channel is denoted by the 3 x 3 symbol. One by one, network elements are set aside in order to carry out the specific operation. Study on ReLU actuation is being considered here. The highest pooling layer recovers the most reward pixel from a group of pixels within a bit. The phase size of the convolution is set to 1 pixel. The watchword phase here refers to the convolutional network's progression. The judgment of enactment work is important here; this is a non-straight shift that characterizes the yield of a neural network).ReLU is chosen because it has a simpler measurement, making it particularly suitable for large neural networks because it reduces planning and estimation times, as well as the possibility of soaking when the information is less than 0, which is prevented by faulty ReLU. It also reduces the computing cost of setting up an organisation. This allows for the development of larger nets with more boundaries at the same computational cost. VGG 11 has 11 fully connected convolutional layers, while VGG 16 has 16 convolutional layers with thick layers.



ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64	conv3-64	conv3-64	conv3-64	conv3-64
	LRN	**conv3-64**	conv3-64	conv3-64	conv3-64
maxpool					
conv3-128	conv3-128	conv3-128	conv3-128	conv3-128	conv3-128
		conv3-128	conv3-128	conv3-128	conv3-128
maxpool					
conv3-256	conv3-256	conv3-256	conv3-256	conv3-256	conv3-256
conv3-256	conv3-256	conv3-256	conv3-256	conv3-256	conv3-256
			conv1-256	**conv3-256**	conv3-256
					conv3-256
maxpool					
conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
			conv1-512	**conv3-512**	conv3-512
					conv3-512
maxpool					
conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
			conv1-512	**conv3-512**	conv3-512
					conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure 7: VGG11 Architecture

3 Proposed Approach

3.1 Dataset Collection:

For Handwritten Numerals different dataset are available like ISI, Ekushe, CMARTdb, EMNIST etc. These dataset contains a lot of handwritten digit images. But these datasets does not have enough data to train the model or good quality images. Recently a dataset called **NumtaDB** was created for this purpose. This is a large and unbiased dataset which contains more than **85000** images. It contains qualityful real world images. It was created from six different sources.The dataset is split into eight subfolders, with six named testing-a, testing-b, testing-c, testing-d, testing-e, and testing-f, and two named testing-auga and testing-augc. This two folders were created by applying augmentation process on the images so that the model could be trained with variety of images. This saves the model from overfitting and underfitting.

For our work we used the NumtaDB dataset. The dataset was divided into two part one for training and one for validation.80% for training and 20% for testing.

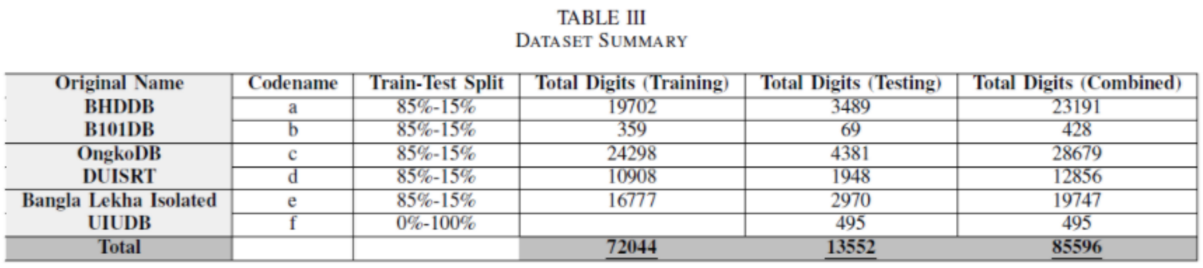


TABLE III
DATASET SUMMARY

Original Name	Codename	Train-Test Split	Total Digits (Training)	Total Digits (Testing)	Total Digits (Combined)
BHDDB	a	85%-15%	19702	3489	23191
B101DB	b	85%-15%	359	69	428
OngkoDB	c	85%-15%	24298	4381	28679
DUISRT	d	85%-15%	10908	1948	12856
Bangla Lekha Isolated	e	85%-15%	16777	2970	19747
UIUDB	f	0%-100%		495	495
Total			**72044**	**13552**	**85596**

Figure 8: NumtaDB Dataset

3.2 Data Preprocessing:

On the videos, several preprocessing steps were added before training the model with the dataset. Some filters were used to reduce noise from the photographs and to eliminate the blur effect. The image's scale was updated on a regular basis.

The photographs were resized to 28 * 28 pixels after being translated to gray scale.

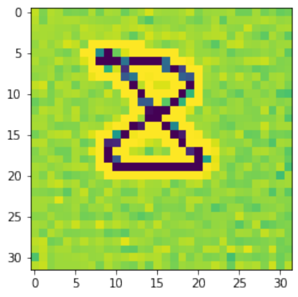


Figure 9: MobileNet Architecture

In Figure 9 we can see a sample of augmented image from the dataset.

3.3 Convolutional Neural Network:

A convolutional neural network (CNN) is a form of phony neural network used in image recognition and processing that is specifically designed to deal with pixel data.

CNNs are excellent image handling, artificial intelligence (AI) systems that use deep learning to execute both generative and clear tasks, often using machine vision, which includes image and video recognition, as well as recommender applications and common language preparation (NLP).

A neural network is a collection of hardware and software that is modeled after the behavior of neurons in the human brain. Traditional neural networks are not suitable for image preparation and can only be used with images of small target bits. CNN's "neurons" are somewhat like those of the frontal projection, the area responsible for managing sensory enhancements of humans and other animals. The layers of neurons are arranged in such a manner that they occupy the entire visual field, avoiding the problem of piecemeal image preparation that plagues traditional neural networks.

A CNN employs a multilayer perceptron-like architecture that has been optimized for reduced handling requirements. An knowledge layer, a yield layer, and a hidden layer compose the layers of a CNN, which include various convolutional layers, pooling layers, fully connected layers, and standardization layers. The removal of impediments and increase in efficiency for picture preparation results in a process that is undeniably more efficient, more simple to train for picture handling and daily language preparation.

3.4 Proposed CNN Model:

- **Architecture of the Model:** Our proposed model has five convolution layers and two thick layers that are totally connected. Each of the first two convolution layers has eight filters with a 3X3 filter dimension. Then we use batch normalization and maximum pooling. The maximum pooling capacity is 2x2. After that, two convolution layers are used. The first one has 16 filters with a 3x3 filter ratio. The second one has 32 filters with a 3x3 filter ratio. Then, to avoid the model from overfitting, we use dropout(50%). At each update of the training process, Dropout operates by setting the outgoing edges of hidden layers to 0 at random. Then we use maximum pooling for a 2x2 pool size. Then, for each mini-batch, we use batch normalization to normalize the inputs to a layer. As a result, the learning algorithm can follow a moving goal. The final convolution layer has 32 filters with a 3x3 filter dimension. Dropout (20%) and batch normalization are used once more. And we add two thick layers on top of that. The first has 32 nodes, while the second has just 10. Both layers except the last use the Rectified Linear Unit (ReLU) as an activation function. The softmax activation mechanism is used in the final dense layer. To change the weights, we use Adam optimizer. The nature of our proposed architecture is depicted in Figure 7. Any configuration is properly labelled from input to output.

  Here in the figure we can see the architecture of our proposed model.We can see the 5 concolutional layer and maxpooling and batch normalization after every two convolutional layer then after the 5th convolutional layer we have

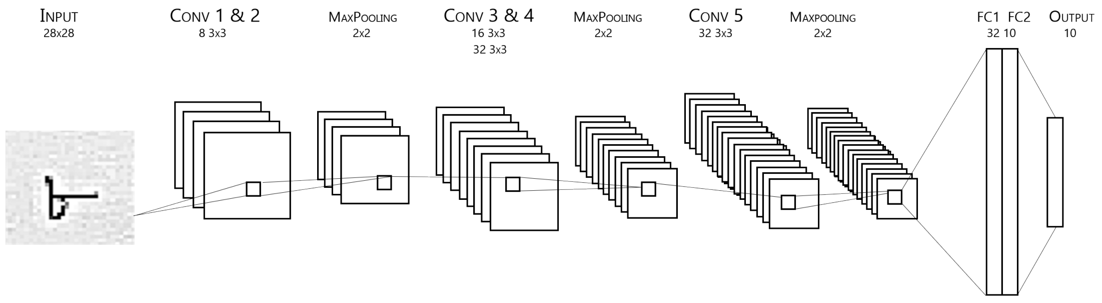
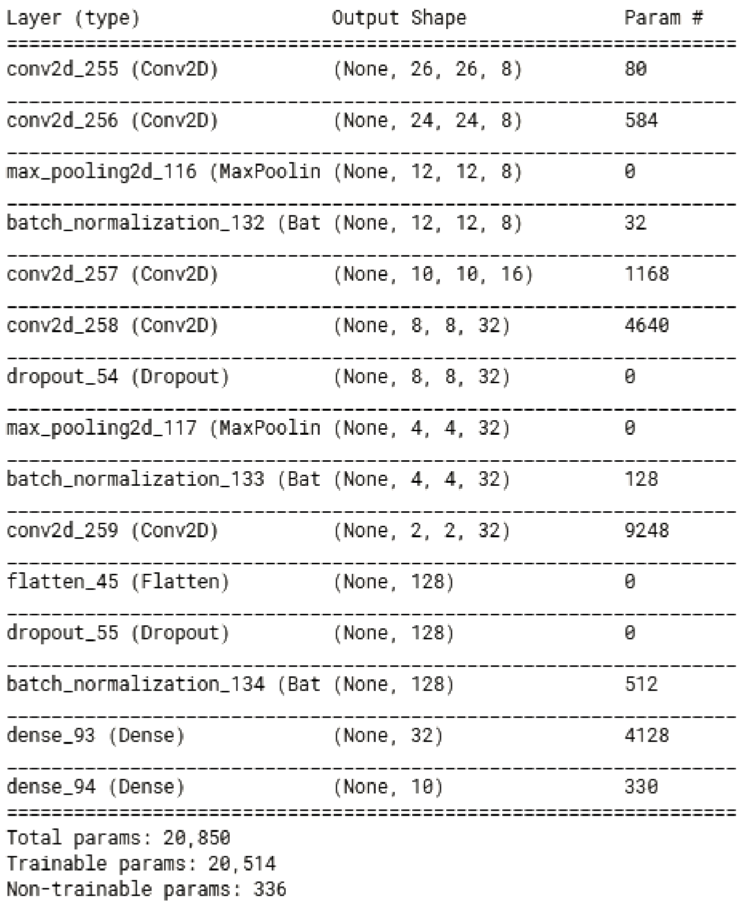


Figure 10: Proposed Model Architecture

a maxpooling layer then fully connected dense layer and a output layer with 10 class label for the ten digits (0 to 9).



```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_255 (Conv2D)          (None, 26, 26, 8)         80
_________________________________________________________________
conv2d_256 (Conv2D)          (None, 24, 24, 8)         584
_________________________________________________________________
max_pooling2d_116 (MaxPoolin (None, 12, 12, 8)         0
_________________________________________________________________
batch_normalization_132 (Bat (None, 12, 12, 8)         32
_________________________________________________________________
conv2d_257 (Conv2D)          (None, 10, 10, 16)        1168
_________________________________________________________________
conv2d_258 (Conv2D)          (None, 8, 8, 32)          4640
_________________________________________________________________
dropout_54 (Dropout)         (None, 8, 8, 32)          0
_________________________________________________________________
max_pooling2d_117 (MaxPoolin (None, 4, 4, 32)          0
_________________________________________________________________
batch_normalization_133 (Bat (None, 4, 4, 32)          128
_________________________________________________________________
conv2d_259 (Conv2D)          (None, 2, 2, 32)          9248
_________________________________________________________________
flatten_45 (Flatten)         (None, 128)               0
_________________________________________________________________
dropout_55 (Dropout)         (None, 128)               0
_________________________________________________________________
batch_normalization_134 (Bat (None, 128)               512
_________________________________________________________________
dense_93 (Dense)             (None, 32)                4128
_________________________________________________________________
dense_94 (Dense)             (None, 10)                330
=================================================================
Total params: 20,850
Trainable params: 20,514
Non-trainable params: 336
```

Figure 11: Model Summary

Here we can see the summary of our proposed simple CNN model with total parameters of 20,084 where 20,514 parameters are trainable and 336 parameters are non trainable. We can also see the filter size and dimensions of the filters in each layer of the Convolutional Neural Network.We have used two dropout layer.

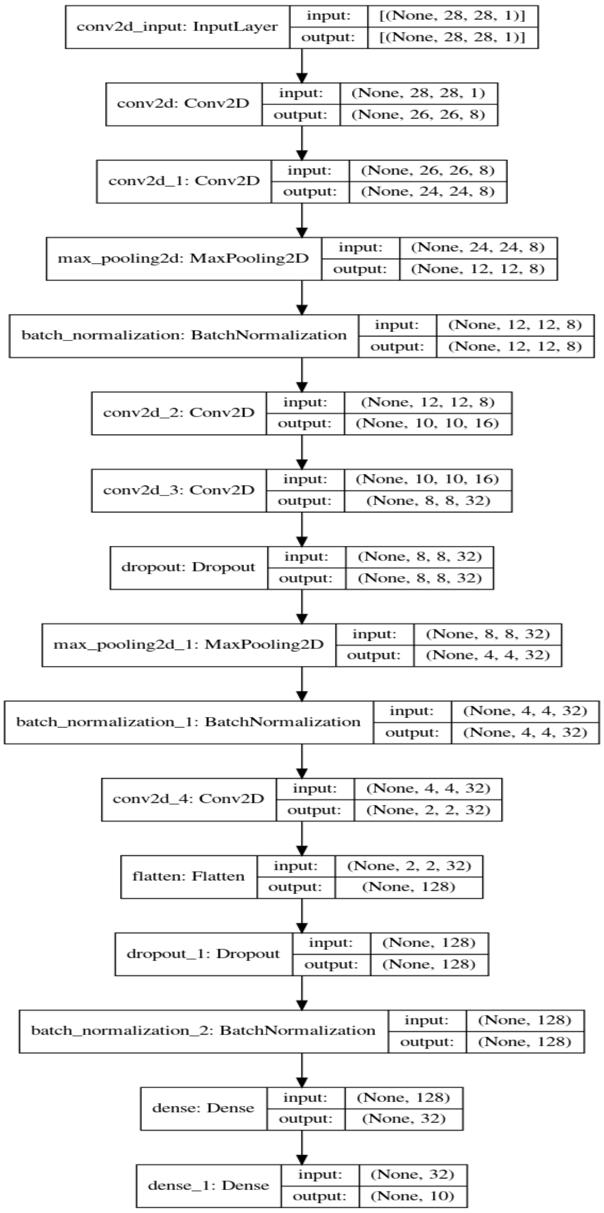

conv2d_input: InputLayer | input: [(None, 28, 28, 1)] | output: [(None, 28, 28, 1)]
conv2d: Conv2D | input: (None, 28, 28, 1) | output: (None, 26, 26, 8)
conv2d_1: Conv2D | input: (None, 26, 26, 8) | output: (None, 24, 24, 8)
max_pooling2d: MaxPooling2D | input: (None, 24, 24, 8) | output: (None, 12, 12, 8)
batch_normalization: BatchNormalization | input: (None, 12, 12, 8) | output: (None, 12, 12, 8)
conv2d_2: Conv2D | input: (None, 12, 12, 8) | output: (None, 10, 10, 16)
conv2d_3: Conv2D | input: (None, 10, 10, 16) | output: (None, 8, 8, 32)
dropout: Dropout | input: (None, 8, 8, 32) | output: (None, 8, 8, 32)
max_pooling2d_1: MaxPooling2D | input: (None, 8, 8, 32) | output: (None, 4, 4, 32)
batch_normalization_1: BatchNormalization | input: (None, 4, 4, 32) | output: (None, 4, 4, 32)
conv2d_4: Conv2D | input: (None, 4, 4, 32) | output: (None, 2, 2, 32)
flatten: Flatten | input: (None, 2, 2, 32) | output: (None, 128)
dropout_1: Dropout | input: (None, 128) | output: (None, 128)
batch_normalization_2: BatchNormalization | input: (None, 128) | output: (None, 128)
dense: Dense | input: (None, 128) | output: (None, 32)
dense_1: Dense | input: (None, 32) | output: (None, 10)

Figure 12: Proposed Model Flowchart

Here we have given the detailed flowchart of our proposed simple Convolutional Neural Network model.

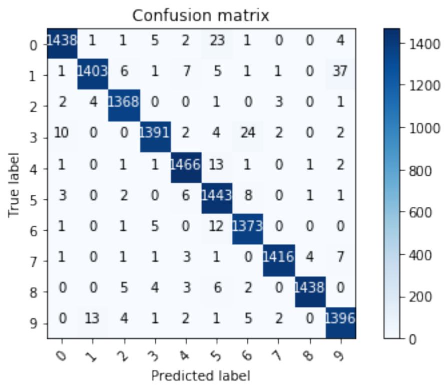
Training Model Parameters: The training model of our architecture is dependent on some parameters. The parameters of the training model are given in the table below:

Parameter	**Value**
Epoch	100
Learning Rate	10^{-2}
Batch Size	64
Total Parameters	20,850

Table 1: Training Parameters

4 Result Analysis

For a model that has a little over 20 thousand parameters, our model performs very well in the short amount of time it takes to train the number of parameters that it has. But still, it's not perfect, as there are some miss-classifications. From figure 12's confusion matrix, we can see that it classifies some 1 digits as 9. 3 is sometimes classified as 6, and 6 is classified as 5 sometimes. We can also see that 0 is classified as 5 in various cases as they look very similar.



True \ Predicted	0	1	2	3	4	5	6	7	8	9
0	1438	1	1	5	2	23	1	0	0	4
1	1	1403	6	1	7	5	1	1	0	37
2	2	4	1368	0	0	1	0	3	0	1
3	10	0	0	1391	2	4	24	2	0	2
4	1	0	1	1	1466	13	1	0	1	2
5	3	0	2	0	6	1443	8	0	1	1
6	1	0	1	5	0	12	1373	0	0	0
7	1	0	1	1	3	1	0	1416	4	7
8	0	0	5	4	3	6	2	0	1438	0
9	0	13	4	1	2	1	5	2	0	1396

Figure 13: Confusion Matrix

We can see the loss of the model become smaller with each epoch, having a lot of loss at first and slightly getting better every time and finally getting a very good accuracy of over 98% with around 100 epochs.

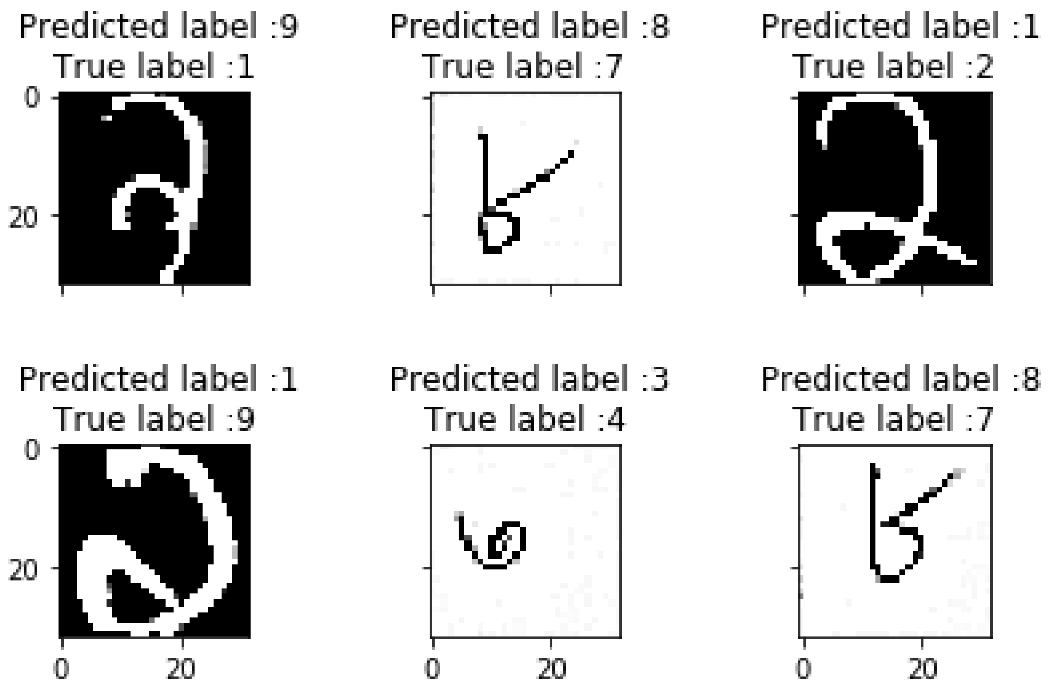


Figure 14: Sample Error

Here we can see some error sample predicted by our model. It predicted 9 as 1 and 1 as 9 sometime and also some other digits were predicted wrong.If we hyper tune the parameter more efficiently and also provide good dataset images for training then it can reduce the error and predict more accurately.

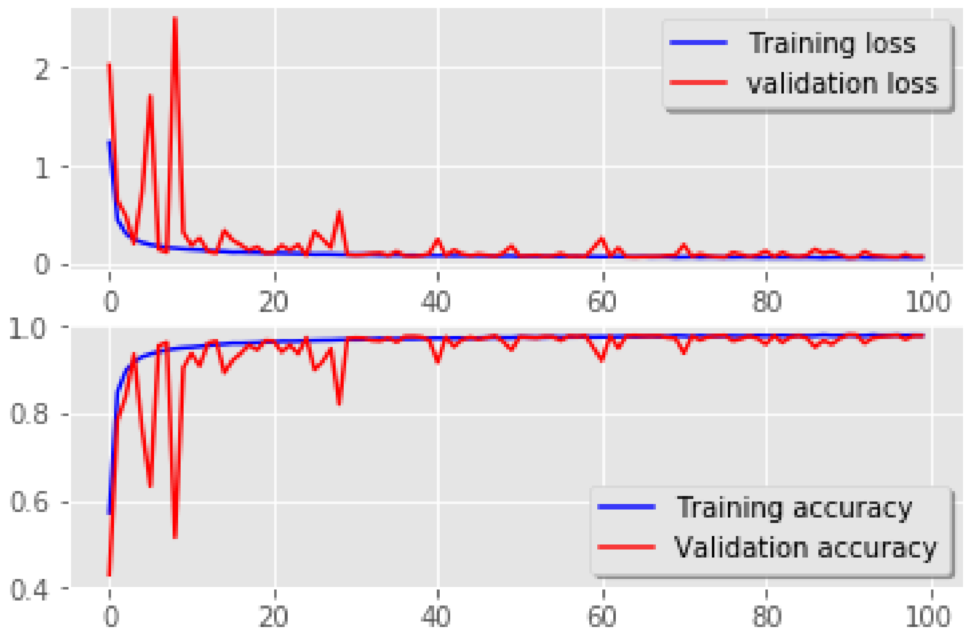


Figure 15: Loss and Accuracy

Here we can see the training and validation loss as well as accuracy of our proposed model. Initially the model gives large loss but as epoch increases it performs better and improves the accuracy of the model.

```
57636/57636 [==============================] - 4s 66us/step
Train loss : 0.022862118108309088
Train accuracy : 99.37192032757305 %


14409/14409 [==============================] - 1s 65us/step
Test loss : 0.0634159583949213
Test accuracy : 98.25109306683323 %
```

Figure 16: Accuracy of the model

we can see that our proposed model has achieved about 99.37% accuracy on training and 98.25% accuracy on testing. So we can say that our model gets similar accuracy rate like the existing model but has a lot less number of parameters and also requires less amount of computational time.

Models	Parameters	Time(sec)
VGG16 Hasib et al.[9] (97.09%)	47075162	35000
ResNet Hasan et al.[2] (99.34%)	44708202	27000
CNN Mahmud et al.[27] (99.43%)	15734890	16300
CNN Noor et al.[3] (96.79%)	9483613	11000
CNN Shawon et al.[1] (92.72%)	1360810	1170
Proposed Model (98.25%)	20850	800

Table 2: Comparison of Models

We compare our model with currently existing models to see how it performs against other models. From table 1, we can see that each one of those models have very good accuracy, but they also have a lot of parameters. As such, they need a lot of time to be trained. As the parameters reduce, their accuracy reduces as well, where our model performs a lot better considering the small number of parameters.

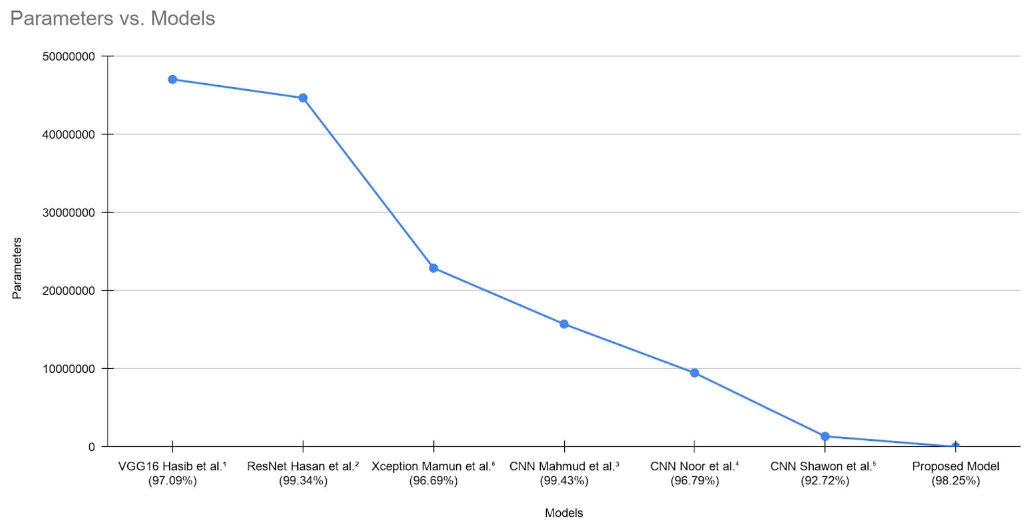


Figure 17: Parameter Comparison

In Figure 17 we have shown the graphichal comparison of our proposed model with the existing models according to the total number of parameters. Our proposed model has huge less number of parameters than the others as well as achieved same amount of accuracy like the existing models.

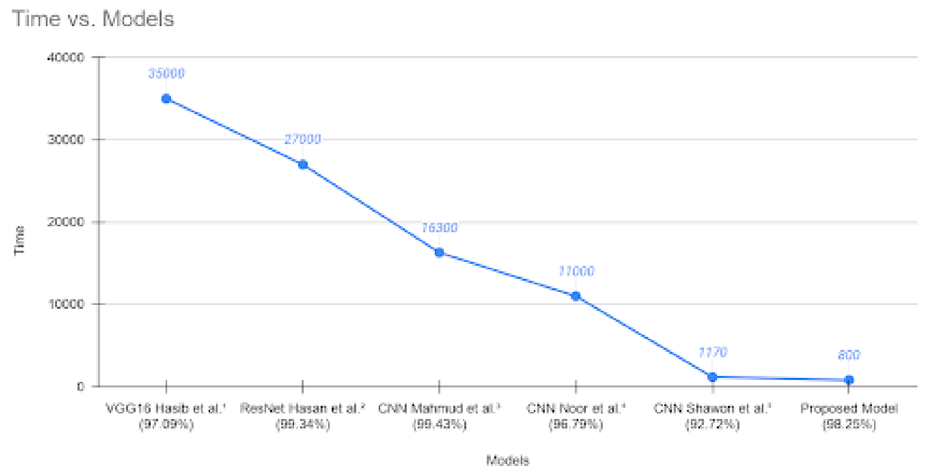


Figure 18: Time Comparison

In Figure 18 we have shown the computational time comparison of our model with the existing models. We can see that our models achieves 98.25% accuracy with the least amount of time that is only in **800 seconds**.

So the graphs and the comparison table shows that our model out performs the existing models in comparison to time and parameters.We only sacrifice about 1.085% accuracy and reduce about 750% parameters compare to the state of the art.

5 Conclusion and Future Work

We can see that our model achieved good accuracy in recognizing bangla handwritten digits. It has achieved about **99.37%** accuracy on training and about **98.25%** accuracy on testing. So we can assume that it is a simple and efficient model for recognizing digits. But as we know everything comes with room of improvements. Our model is not exception to that. We think our model has room for improvements. If we can tune our models hyper-parameters that we can achieve more accuracy and also simplify the model.

For our future work, we are trying to improve our model as much as we can and also apply the other publicly available data sets and see how it performs on them. We are also aiming to extend our model and implement bangla character recognition (OCR).Then it will be able to recognize both handwritten bangla characters as well as handwritten digits.

References

[1] A. Shawon, M. J.-U. Rahman, F. Mahmud, and M. A. Zaman, "Bangla handwritten digit recognition using deep cnn for large and unbiased dataset," in *2018 International Conference on Bangla Speech and Language Processing (ICBSLP)*, pp. 1–6, IEEE, 2018.

[2] M. M. Hasan, M. R. U. Islam, and M. T. Mahmood, "Recognition of bengali handwritten digits using convolutional neural network architectures," in *2018 International Conference on Bangla Speech and Language Processing (ICBSLP)*, pp. 1–6, IEEE, 2018.

[3] R. Noor, K. M. Islam, and M. J. Rahimi, "Handwritten bangla numeral recognition using ensembling of convolutional neural network," in *2018 21st International Conference of Computer and Information Technology (ICCIT)*, pp. 1–6, IEEE, 2018.

[4] M. Z. Alom, P. Sidike, T. M. Taha, and V. K. Asari, "Handwritten bangla digit recognition using deep learning," *arXiv preprint arXiv:1705.02680*, 2017.

[5] B. Purkaystha, T. Datta, and M. S. Islam, "Bengali handwritten character recognition using deep convolutional neural network," in *2017 20th International conference of computer and information technology (ICCIT)*, pp. 1–5, IEEE, 2017.

[6] A. S. A. Rabby, S. Abujar, S. Haque, and S. A. Hossain, "Bangla handwritten digit recognition using convolutional neural network," in *Emerging Technologies in Data Mining and Information Security*, pp. 111–122, Springer, 2019.

[7] M. R. Mamun, Z. Al Nazi, and M. S. U. Yusuf, "Bangla handwritten digit recognition approach with an ensemble of deep residual networks," in *2018 International Conference on Bangla Speech and Language Processing (ICBSLP)*, pp. 1–4, IEEE, 2018.

[8] O. Paul, "Image pre-processing on numtadb for bengali handwritten digit recognition," in *2018 International Conference on Bangla Speech and Language Processing (ICBSLP)*, pp. 1–6, IEEE, 2018.

[9] H. Zunair, N. Mohammed, and S. Momen, "Unconventional wisdom: A new transfer learning approach applied to bengali numeral classification," in *2018 International Conference on Bangla Speech and Language Processing (ICB-SLP)*, pp. 1–6, IEEE, 2018.

[10] S. A. Kamran, A. I. Humayun, S. Alam, R. M. Doha, M. K. Mandal, T. Reasat, and F. Rahman, "Ai learns to recognize bengali handwritten digits: Bengali. ai computer vision challenge 2018," *arXiv preprint arXiv:1810.04452*, 2018.

[11] S. Alam, T. Reasat, R. M. Doha, and A. I. Humayun, "Numtadb-assembled bengali handwritten digits," *arXiv preprint arXiv:1806.02452*, 2018.

[12] A. Islam, F. Rahman, and A. S. A. Rabby, "Sankhya: an unbiased benchmark for bangla handwritten digits recognition," in *2019 IEEE International Conference on Big Data (Big Data)*, pp. 4676–4683, IEEE, 2019.

[13] C. Saha, R. H. Faisal, and M. M. Rahman, "Bangla handwritten digit recognition using an improved deep convolutional neural network architecture," in *2019 International Conference on Electrical, Computer and Communication Engineering (ECCE)*, pp. 1–6, IEEE, 2019.

[14] T. I. Aziz, A. S. Rubel, M. S. Salekin, and R. Kushol, "Bangla handwritten numeral character recognition using directional pattern," in *2017 20th International Conference of Computer and Information Technology (ICCIT)*, pp. 1–5, IEEE, 2017.

[15] G. Rabbani, M. A. Islam, M. A. Azim, M. K. Islam, and M. M. Rahman, "Bangladeshi license plate detection and recognition with morphological operation and convolution neural network," in *2018 21st International Conference of Computer and Information Technology (ICCIT)*, pp. 1–5, IEEE, 2018.

[16] A. Sufian, A. Ghosh, A. Naskar, F. Sultana, J. Sil, and M. H. Rahman, "Bdnet: bengali handwritten numeral digit recognition based on densely connected convolutional neural networks," *Journal of King Saud University-Computer and Information Sciences*, 2020.

[17] R. Basri, M. R. Haque, M. Akter, and M. S. Uddin, "Bangla handwritten digit recognition using deep convolutional neural network," in *Proceedings of the international conference on computing advancements*, pp. 1–7, 2020.

[18] M. A. Al Nasim, R. E. Ferdous, M. A. H. Pantho, and A. I. Chowdhury, "A comparative analysis on bangla handwritten digit recognition with data augmentation and non-augmentation process," in *2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, pp. 1–5, IEEE, 2020.

[19] M. M. Rahman, S. M. Shaiban, S. S. Hasan, M. R. Tanjim, and M. A. Khan, "Forensic analysis of bangla handwritten letters," in *Proceedings of the 8th International Conference on Computer and Communications Management*, pp. 74–78, 2020.

[20] M. M. Rahman, M. S. Islam, R. Sassi, and M. Aktaruzzaman, "Convolutional neural networks performance comparison for handwritten bengali numerals recognition," *SN Applied Sciences*, vol. 1, no. 12, pp. 1–11, 2019.

[21] M. F. Sikder, "Bangla handwritten digit recognition and generation," in *Proceedings of International Joint Conference on Computational Intelligence*, pp. 547–556, Springer, 2020.

[22] M. Zahangir Alom, P. Sidike, M. Hasan, T. M. Taha, and V. K. Asari, "Handwritten bangla character recognition using the state-of-art deep convolutional neural networks," *arXiv e-prints*, pp. arXiv–1712, 2017.

[23] Y. Zhu, S. Suri, P. Kulkarni, Y. Chen, J. Duan, and C.-C. J. Kuo, "An interpretable generative model for handwritten digits synthesis," in *2019 IEEE*

International Conference on Image Processing (ICIP), pp. 1910–1914, IEEE, 2019.

[24] M. N. Hoq, N. A. Nipa, M. M. Islam, and S. Shahriar, "Bangla handwritten character recognition: an overview of the state of the art classification algorithm with new dataset," in *2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT)*, pp. 1–6, IEEE, 2019.

[25] Z. K. Nishat and M. Shopon, "Synthetic class specific bangla handwritten character generation using conditional generative adversarial networks," in *2019 International Conference on Bangla Speech and Language Processing (ICBSLP)*, pp. 1–5, IEEE, 2019.

[26] M. J. Hasan, M. F. Wahid, M. S. Alom, and M. M. A. Mia, "A new state of art deep learning approach for bangla handwritten digit recognition using svm classifier," in *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pp. 1–6, IEEE, 2020.

[27] T. Mahmud, A. R. Hossain, and S. A. Fattah, "Deepbanglanet: A deep convolutional neural network to recognize bengali handwritten digits," in *2020 IEEE Region 10 Symposium (TENSYMP)*, pp. 742–745, IEEE, 2020.

[28] S. Ghosh, A. Chatterjee, P. K. Singh, S. Bhowmik, and R. Sarkar, "Language-invariant novel feature descriptors for handwritten numeral recognition," *The Visual Computer*, pp. 1–23, 2020.

[29] M. B. Hossain, A. Adhikary, and S. J. Soheli, "Sign language digit recognition using different convolutional neural network model," *Asian Journal of Research in Computer Science*, pp. 16–24, 2020.

[30] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size," *arXiv preprint arXiv:1602.07360*, 2016.

[31] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.