HANDWRITTEN ENGLISH ALPHABET RECOGNITION USING NEURAL NETWORK

MCA 4th Semester Dissertation Report

Submitted in partial fulfillment of the requirements
for the award of the degree of

MASTER OF COMPUTER APPLICATIONS

(MCA)

by

Pritom Pujari

(Roll No. 22MCA14)

(Registration No. 22090440)

to

DEPARTMENT OF COMPUTER APPLICATIONS

NORTH EASTERN HILL UNIVERSITY

Tura Campus, Tura

Meghalaya, India-794002

Under the Supervision of

Dr. Raj Kumar Goel

Associate Professor

Department of Computer Application

North-Eastern Hill University

Tura Campus

July, 2024

पूर्वोत्तर पर्वतीय विश्वविद्यालय
कंप्यूटर अनुप्रयोग विभाग
तुरा परिसर, पश्चिम गारो हिल्स, तुरा- 794002
मेघालय (भारत)
ईमेल: hodca@nehu.ac.in

North-Eastern Hill University
Department of Computer Application
Tura Campus, West Garo Hills, Tura-794002
Meghalaya (India)
Email: hodca@nehu.ac.in

CERTIFICATE OF APPROVAL

This is to certify that the MCA 4th Semester dissertation work entitled **"HANDWRITTEN ENGLISH ALPHABET RECOGNITION USING NEURAL NETWORK"** carried out by **Pritom Pujari** bearing Roll No: **22MCA14** and Registration No: **22090440** under the guidance of **Dr. Raj Kumar Goel** has been found satisfactory and is approved as a project work carried out and presented in a manner required for its acceptance in partial fulfillment of the requirements for the degree of Master of Computer Applications (MCA) from North-Eastern Hill University, Tura Campus, Meghalaya.

INTERNAL EXAMINER

DATE:

PLACE:

EXTERNAL EXAMINER

DATE:

PLACE:

पूर्वोत्तर पर्वतीय विश्वविद्यालय
कंप्यूटर अनुप्रयोग विभाग
तुरा परिसर, पश्चिम गारो हिल्स, तुरा- 794002
मेघालय (भारत)
ईमेल: hodca@nehu.ac.in

North-Eastern Hill University
Department of Computer Application
Tura Campus, West Garo Hills, Tura-794002
Meghalaya (India)
Email: hodca@nehu.ac.in

CERTIFICATE

FROM

THE HEAD OF DEPARTMENT

This is to certify that the MCA 4th Semester dissertation work entitled **HANDWRITTEN ENGLISH ALPHABET RECOGNITION USING NEURAL NETWORK** is submitted by **Pritom Pujari** bearing Roll No: **22MCA14** and Registration No: **22MCA14** under the guidance of **Dr. Raj Kumar Goel**, in partial fulfillment of the requirements for the degree of Master of Computer Applications (MCA) from North-Eastern Hill University, Tura Campus, Meghalaya.

DATE: _____

PLACE: _____

_____

Head

Dept. of Computer Application,

North-Eastern Hill University

Tura Campus, Meghalaya

पूर्वोत्तर पर्वतीय विश्वविद्यालय
कंप्यूटर अनुप्रयोग विभाग
तुरा परिसर, पश्चिम गारो हिल्स, तुरा- 794002
मेघालय (भारत)
ईमेल: hodca@nehu.ac.in

North-Eastern Hill University
Department of Computer Application
Tura Campus, West Garo Hills, Tura-794002
Meghalaya (India)
Email: hodca@nehu.ac.in

CERTIFICATE FROM THE SUPERVISOR

This is to certify that the MCA 4th Semester dissertation work entitled **HANDWRITTEN ENGLISH ALPHABET RECOGNITION USING NEURAL NETWORK** is a bonafide work carried out by **Pritom Pujari** bearing Roll No: **22MCA14** and Registration No: **22090440** under my supervision and guidance. The report is found to be satisfactory for the partial fulfillment of the requirements for the degree of Master of Computer Applications (MCA) from North-Eastern Hill University, Tura Campus, Meghalaya.

DATE: _____

PLACE: _____

_____

Dr. Raj Kumar Goel
Associate Professor
Dept. of Computer Application,
North-Eastern Hill University
Tura Campus, Meghalaya

पूर्वोत्तर पर्वतीय विश्वविद्यालय
कंप्यूटर अनुप्रयोग विभाग
तुरा परिसर, पश्चिम गारो हिल्स, तुरा- 794002
मेघालय (भारत)
ईमेल: hodca@nehu.ac.in

North-Eastern Hill University
Department of Computer Application
Tura Campus, West Garo Hills, Tura-794002
Meghalaya (India)
Email: hodca@nehu.ac.in

DECLARATION

I hereby declare that the dissertation work entitled **"HANDWRITTEN ENGLISH ALPHABET RECOGNITION USING NEURAL NETWORK"** submitted by **Pritom Pujari** bearing Roll No. **22MCA14** and Registration No. **22090440** as a developer of the project submitted to Department of Computer Application, North-Eastern Hill University, Tura Campus, Meghalaya for the partial fulfillment of the requirements for the degree of Master of Computer Applications (MCA). I further declare that the dissertation work had not been submitted elsewhere for any other degree or diploma before.

DATE: _____

PLACE: _____

_____

Pritom Pujari

Roll No : **22MCA14**

Registration No : **22090440**

Dedicated to

my mother

Momi Pujari

and

my father

Utpal Pujari

ACKNOWLEDGEMENTS

On successful completion of my project, I would like to express my sincere gratitude to everyone who helped and supported me. However, I wish to make special men tion of the following. Firstly, I am highly indebted to my guide **Dr. Raj Kumar Goel** (Associate Professor), Department of Computer Application, North-Eastern Hill University Tura Campus, Meghalaya for the valuable guidance which has promoted my efforts in all stages of this project work. I am also thankful to Head of Department, **Dr. Anindya Halder (Associate Professor)** and **Dr. Ansuman Kumar (Assistant Professor)** who gave me the opportunity to do this project. My thanks and appreciation goes to my friends and staffs as well who willingly helped me without any hesitation.

Regards

Pritom Pujari

Roll No.: **22MCA14**

Reg. No.: **22090440**

Date: .....

ABSTRACT

Handwritten character recognition (HCR) is the detection of characters from images, documents and other sources and changes them in machine-readable shape for further processing. Recent advances in Convolutional neural networks help people get rid of extracting feature sets manually and significantly improve recognition efficiency.In this project, we aim to create a handwriting recognition system that uses deep learning to accurately recognize handwritten text. The dataset used to train and evaluate various deep learning architecture comprises 26 classes, each representing a letter of the English alphabet. We implement and analyze several state-of-the-art convolutional neural network architectures, including LeNet-5, ResNet-50, MobileNet, Inception, ShuffleNet, and EfficientNet. Additionally, we propose four new hybrid models that integrate the strengths of these architectures to enhance recognition accuracy. Among all models, the highest recognition accuracy of 99.69% is achieved using a proposed hybrid model combining MobileNet, ResNet-50, and EfficientNet. The experimental results demonstrate significant improvements in recognition rates for the hybrid model, underscoring its potential for practical applications in optical character recognition (OCR) systems and beyond.

Keywords: Handwritten english alphabet dataset; Lenet5; Resnet50; Mobilenet; Inception; Shufflenet; Efficientnet; hybrid model; Handwrittten character recognition.

Contents

List of Tables

List of Figures

1 Introduction

1.1 Overview

Character recognition is a process that associates symbolic identifiers with visual representations of characters. It involves analyzing images containing letters, symbols, or numerals and assigning them appropriate symbolic meanings. This technology enables the conversion of visual character data into machine-readable text, bridging the gap between human-readable content and digital information systems. Character recognition machines primarily need raw data to carry out the preprocessing step of any recognition system. The following categories can be applied to character recognition systems based on that data collecting process as shown in figure 1.1.

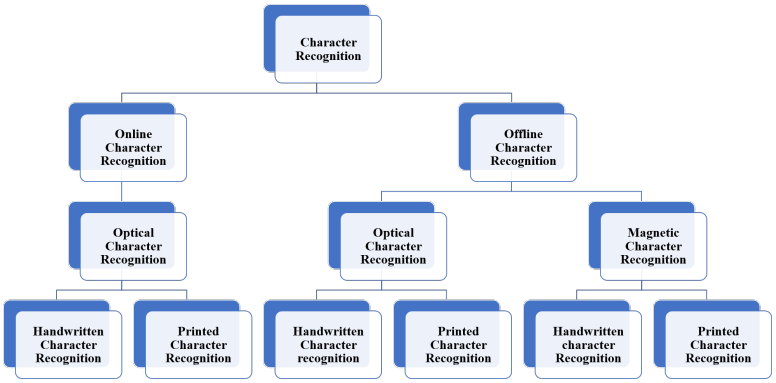


Figure 1.1: Classification of character recognition

- **Offline character recognition** : The method of identifying words that have been digitally recorded in greyscale format after being scanned from a surface, like a sheet of paper, is known as offline handwriting recognition. It

is customary to carry out further processing after storage to enable better recognition. Different methods are used to capture and save the handwriting in digital form for online handwritten character recognition. Typically, an electronic surface is used in conjunction with a specialized pen. The two-dimensional coordinates of subsequent points are recorded in order and are represented as a function of time as the pen advances across the surface.[1, 2]

- **Online character recognition** :- It is known that online character recognition performs better than handwriting recognition offline. This could be explained by the possibility that additional details, such the handwriting's direction, pace, and stroke order, are recorded in the online instance.

The primary distinction between offline and online character recognition is because offline data lacks real-time contextual information, but online character recognition does[2, 3]. There is a notable disparity in processing structures and methods as a result of this difference. Two forms of offline character recognition can be distinguished further:

1.1.1 Handwritten Character Recognition

The process of converting handwritten text into a machine-readable format is known as handwritten character recognition or HCR. The primary challenge facing handwritten character recognition (HCR) systems is the wide range of handwriting styles, which can differ greatly amongst authors. The goal of the handwritten character recognition system is to digitalize and convert handwritten text into machine-readable text by implementing an approachable computer-assisted character representation that will enable the successful extraction of characters from handwritten documents.

In this project, we try to develop a recognition system with the help of some prebuilt models available for english handwritten characters (alphabets only) that can more precisely identify a written character.

1.2 Aims and Objective of the Project

1.2.1 Objective

This objective of the project is to design, develop, and implement a robust system that would accurately recognize English handwritten characters. The basic aim is to overcome the individual style variability in handwriting to give a reliable transcription of handwritten information into a digital representation of the same.

1.2.2 Aims of the project

1. **Develop a Handwritten English Alphabet Recognition System**: To create an advanced system that can accurately recognize handwritten English alphabets using neural network techniques.
2. **Implement and Compare Various CNN Architectures**: To study, implement, and compare the performance of different Convolutional Neural Network (CNN) architectures in recognizing handwritten characters.
3. **Enhance Recognition Accuracy**: To improve the accuracy of handwritten character recognition through fine tuning and transfer learning.
4. **Evaluate Model Performance**: To rigorously evaluate the performance of the developed models using relevant performance metrics such as accuracy, precision, and recall. interfaces

1.3 Motivation of work

The majority of businesses collect consumer information via paperwork. Most of these documents are written by hand. These papers may be checks, forms, etc. Documents are converted to digital formats and kept for simpler retrieval or information gathering. Manually entering the same data into a computer is the standard procedure for handling that information. Managing such records by hand would be laborious and time-consuming. As a result, the need for specialized handwritten character recognition software that can recognise text from

document images automatically develops. Handwritten Character Recognition (HCR) software has made it easier to extract data from handwritten papers and store it in computer formats. sectors include banking, healthcare, and many others where regular use of handwritten papers occurs.

Banking sectors, Health care industries and many such organizations where handwritten documents are used regularly. HCR systems also find applications in newly emerging areas where handwriting data entry is required, such as development of electronic libraries, multimedia database etc.

1.4 Structure of a HCR system

The block diagram of a HCR system [4] is shown below in figure 1.2 The collected

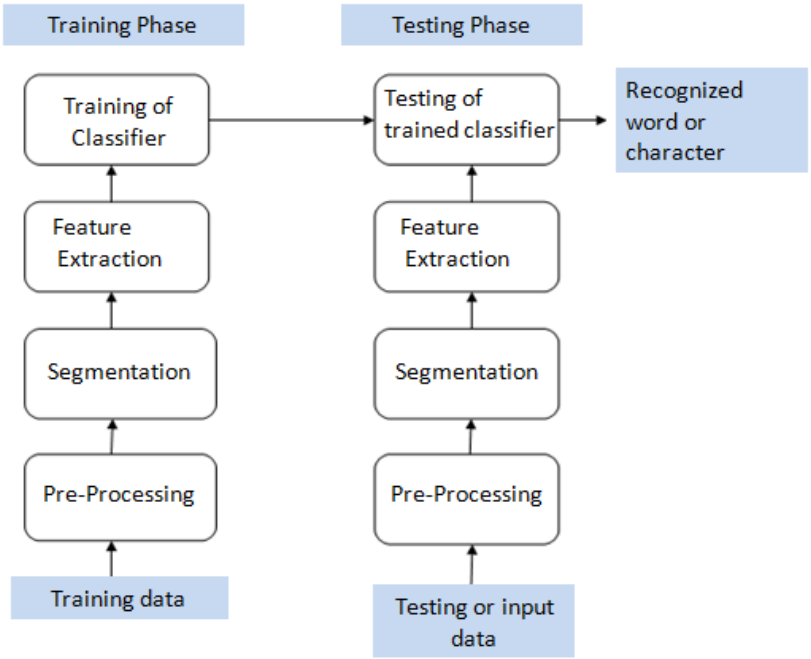


Figure 1.2: Block diagram of HCR

databases are divided into two parts Training data and testing data. Training data (it is also divided in trining and validation data) are used to train the system and this trained system are than used to recognize test data. HCR system consists of the following stages

1.4.1 Preprocessing

The pre-processing is a sequence of operations performed on the scanned input image. It essentially enhances the image making it suitable for further processing. Necessary steps involved in data preprocessing are given below.

- **Image Resizing**: Resize all image to uniform size matching the size that is required as an input by the deep learning model.
- **Numeric conversion**: The neural network cannot directly take image as input, we need to convert the images into numeric values.Converting images into arrays of pixel values. Python libraries such as OpenCV, PIL, NumPy and TensorFlow/Keras can be used to convert images to their numeric representation.
- **Normalization**: Adjusting the pixel values, usually in the range of [0.0, 1.0] or [-1.0, 1.0]. This can be done by dividing the pixel values by 255 (if the original range was 0-255) or normalization through mean and standard deviation of the dataset.

1.4.2 Segmentation

At the segmentation stage, a picture is divided into smaller, more distinct pictures. Line segmentation, or separating a line from a paragraph, is one type of segmentation.

- Word separation from line is known as word segmentation.
- Character segmentation is the process of separating words from characters.If a segmentation-based approach is used for cursive word recognition, character segmentation is carried out.

1.4.3 Feature extraction

The characteristics of the characters that are necessary for categorizing them at the recognition stage are extracted at this point. This is a crucial phase since

it lowers misclassification and increases recognition rate when it operates well. After features are retrieved, a feature vector is constructed, including binary and directional features.Feature extraction techniques are included in these groups.

- **Statistical feature**: It is predicated on hypothesis and probability theory. Variations in writing styles are handled via the statistical distribution of an image's pixels. The statistical distribution of points yields statistical properties. For instance, zoning, crossings, distances, and projections histogram as shown in figure 1.3.

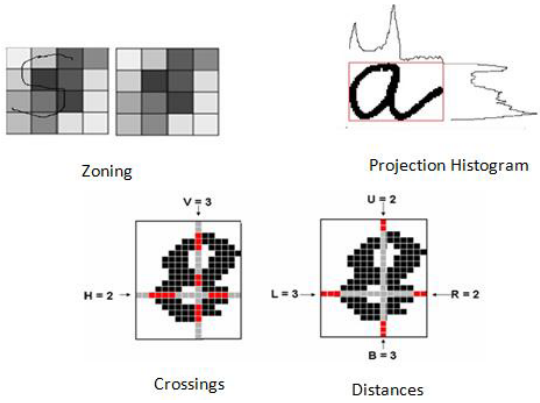


Figure 1.3: Statistical features

- **Structural features**: The image's structure can be inferred from its structural features. Characters with geometric and topological characteristics, such as crossing points, branches, loops, stroke length, stroke breadth, up, down, left, and right projection profiles, are described by structural features[4]. Structural features are depicted in figure 1.4.

- **Classification**: The classification stage is the decision making part of a recognition system and it uses the features extracted in the previous stage. The feature vector is denoted as X where

$$X = (f_1, f_2, ..., f_d) \tag{1.1}$$

where f denotes features and d is the no. features extracted from character. Based on the comparison of feature vector, characters are efficiently classified into appropriate class and recognized.

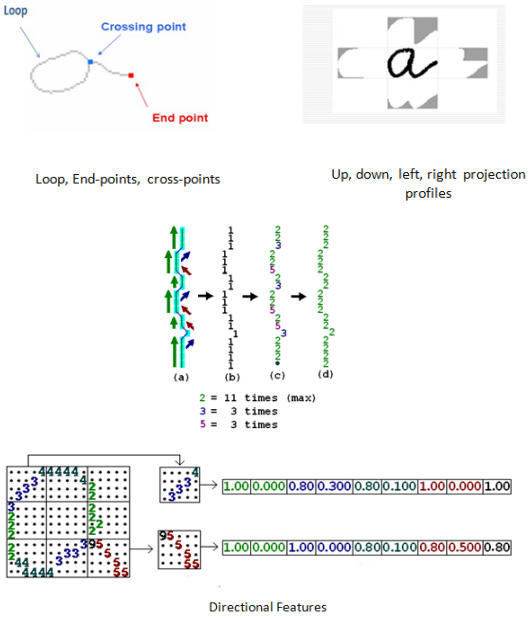


Figure 1.4: Structural features

In deep learning, especially when using convolutional neural networks (CNNs) for image data, explicit feature extraction is often not needed.Because

- Deep learning models, particularly CNNs, are designed to automatically learn and extract features from raw data. Through multiple layers of convolutions, pooling, and non-linear activations, these models can detect hierarchical patterns in data without the need for manual feature engineering.

- Deep learning enables end-to-end learning, where the model learns to map inputs directly to outputs. The intermediate layers automatically extract and refine features as part of the training process.

- The representations learned by deep networks tend to be more complex and informative compared to manually engineered features. Deep networks can capture intricate patterns and dependencies in the data.

In short, explicit feature extraction is generally not required for deep learning models, especially in the context of image data, there are cases where it can be beneficial. The choice depends on the nature of the data, the size of the dataset, and the specific problem .

1.5 Structure of the report

The rest of the report is organized as Chapter 2 discusses existing techniques and research. Chapter 3 explains relevant deep learning concepts.Chapter 4 contains overall process of the handwritten character recognition system. Chapter 5 presents the results, their analysis, and comparisons with previous methods. Chapter 6 summarizes the main findings, addressing the study's objectives and providing final thoughts and recommendations.

2 Literature Review

Al-Mahmud, Asnuva Tanvin, and Sazia Rahman [5]provided a system on Convolutional Neural Networks for handwritten English character and digit recognition. In their paper, a custom-made dataset comprising digits and capital letters was created from the MNIST dataset of handwritten digits and the A-Z Handwritten Alphabet dataset of English capital letters. The authors have therefore proposed a revised framework of CNN inspired by another model introduced by Ranjan Jana and Siddhartha Bhattacharyya, but with modified hyperparameters and the introduction of dropout layers for regularization to prevent overfitting. Along these lines, Al-Mahmud, Tanvin, and Rahman reached an accuracy of 99.47% on the MNIST dataset, thereby outperforming other approaches compared in the paper. For the combined dataset of digits and capital letters, test accuracy achieved was 98.94%. The authors have shown that fine-tuning of hyperparameters actually improves accuracy by trying different learning rates along with various dropouts. They have concluded that although the accuracy of their proposed method in the combined dataset is less as compared to MNIST, it still had very appreciable recognition capability

Shengpei Le [6] introduced a model pertaining to network recognition in English handwriting that generalizes better itself. There is a lightweight convolutional neural network that works on the Chars74K dataset for printed gray images and the EMNIST dataset for handwritten RGB images. The approach involves some steps of preprocessing, including the Hough line detection method, grayscale conversion, normalization, and morphological operations. The architecture used herein is based on LeNet-5, having convolutional and fully connected layers. In this paper, the cross-entropy loss function with Adam optimization

has a dropout rate of 50%. It achieved an accuracy of 93% on the test set with much better recognition accuracy, faster convergences, and great generalization capability compared to traditional models of the network. The findings show a high degree of recognition for well-written words, but somewhat lower—still decent—performance for messy or simply illegible handwritings, from which improvement areas can be drawn.

P. Latchoumy, G. Kavitha, S. Anupriya, and H. Shaila Banu [7] have done one research work in the handwritten recognition domain on the EMNIST dataset. In this paper, Convolutional Neural Networks and Support Vector Machines are applied for classification techniques. The study focuses on training and testing English alphabets from A-Z and digits from 0-9. Results are as high as 95.41% in accuracy, with reduced time taken by computation, thereby proving that CNN and SVM succeed in both increasing accuracy and efficiency for handwriting recognition systems.

Qiu et al.[8] designed a handwritten character recognition system using the Altera DE1 FPGA Kit, which implements linear classification, a fully connected neural network, and a convolutional neural network with Rectified Linear Unit activation layers as three diverse neural network architectures. It makes use of the EMNIST dataset in training and testing, which contains 425,600 images of handwritten characters. The maximum achieved accuracy in recognizing English alphabets and digits in this paper is 95.41%, and the CNN model is deployed to hardware on an FPGA. In this work, Python is used for training with the PyTorch library, and the resulting kernels and weights are saved in hex format files that are transferred to SRAM on the FPGA. The hardware design is based on these various engineering standards: IEEE-754 32-bit floating-point, interface specifications like VGA protocol, UART protocol, and I2C protocol. Also, there will be a self-made 32-bits RISC processor with 5 stages pipeline implemented to control image processing, matrix multiplications, and classifying through a firmware scripted in a custom-designed assembly language.

Priti Singh Rathore and Dr. Pawan Kumar [9] introduced a proposal for a handwritten Hindi character recognition system employing cutting-edge deep

learning methodologies like Convolutional Neural Networks (CNN) and Deep Neural Networks (DNN). The utilization of the Kaggle dataset and a custom dataset comprising grayscale images of handwritten Hindi characters was made. These grayscale images underwent preprocessing, normalization, and labeling from 0 to 47 through one-hot encoding. The CNN algorithm automatically extracted features from the images. To attain promising outcomes, the Adam (Adaptive Moment Estimation) optimization technique was utilized. A 5-layer CNN model was successfully executed, including a convolutional layer, a max-pooling layer, a flattening layer, and two fully connected layers for accurate classification. Activation functions ReLU and Softmax were applied. The suggested system accomplished an accuracy of 87.41% on the validation dataset post-24 epochs, showcasing the efficacy of the CNN model with the Adam optimizer for handwritten Hindi character recognition.

Nazmus Saqib et.al [10]researched CNN-based handwritten character recognition, concentrating on English alphabet characters and digits. Two datasets were employed: Kaggle for alphabet characters and MNIST for digits. Twelve models were formulated by varying hyperparameters, culminating in the selection of the top two models based on their performance. For digit recognition, they utilized the RMSprop optimizer with a learning rate of 0.001, resulting in a peak accuracy of 99.642%, while for alphabet recognition, the ADAM optimizer with a learning rate of 0.00001 was used, achieving an accuracy of 99.563%. The classification reports for these models displayed macro and weighted F1 scores of 0.998 and 0.997 for digits, and 0.992 and 0.996 for alphabets, respectively. Their methodology showcased the efficacy of tailored CNN models in accomplishing high accuracy and dependability for handwritten character recognition, even when dealing with imbalanced datasets, by delivering a comprehensive analysis via precision, recall, specificity, and F1 scores.

A multilayer perceptron neural network with one hidden layer for recognizing handwritten English characters was developed by Anita Pal and Dayashankar Singh.[11] In their research, a total dataset of 500 samples was used—which were equally divided for training and testing. The boundary tracing system coupled

with Fourier descriptors for feature extraction was used to analyze the shape features of the characters. Their system gave an accuracy of 94% on the test set, which is quite a good performance for very small training time.

A study by Bhagyashree P M, L K Likhitha, and D S Rajesh [12] was performed for the performance improvement of handwritten digit recognition using deep learning. Particularly, Convolutional Neural Networks were applied. A single MNIST database was used for training or testing, with 70,000 photos of handwritten numbers in two sets: one for training and another one for testing.Among the proposed architectures are several convolutional layers with max-pooling layers before applying a fully connected layer. After fine-tuning using adaptive moment estimation optimization and categorical cross-entropy function, a classification accuracy of 99.87 was retrieved with a minimum loss of 0.043.

Studies on handwriting recognition for data input using Convolutional Neural Networks were carried out by Y. F. Mustafa, F. Ridho, and S. Mariyah.[13] This paper uses the authors' EMNIST dataset to prove that it needs only categorization by character type to achieve better accuracy results: 89% combined letters and numbers, 95% with letters only, and 99% with numbers only. This methodology consisted of developing a model for handwriting recognition using CNN in order to automate data entry from PAPI questionnaires. On the overall, some 83.33% accuracy showed that the model did well in practical terms against challenges experienced during character segmentation due to continuous lines and broken characters. This basically has pointed to potentials, together with challenges in using CNN-driven handwriting recognition in data-entry processes.

To promote social empowerment through enhanced digital accessibility to Assamese literary works, Naiwrita Borah et al.[14] researched Assamese word recognition. An image collection from the Shankari and Jonaki periods of Assamese literature was hand-picked by them. Various machine learning algorithms based on shape-related variables, including Logistic Regression, Decision Trees, Random Forest, SVM, KNN, Gradient Boosting, and a Convolutional Neural Network (CNN) were also used for recognition. With the highest accuracy of 96.03%, the Gradient Boosting technique was followed closely by SVM with RBF kernel at

approximately 95.6% and Logistic Regression at a similar level. The CNN model outperformed them, achieving 97.3% recognition accuracy.

Meheniger Alam [15] has researched Python frameworks and Convolutional Neural Networks in the field of handwritten character recognition. The principal objects of his research were the MNIST dataset and the English alphabet. Alam came to the conclusion that Statistical SVM was quite adequate for Optical Character Recognition Systems. His CNN setup against the MNIST digit dataset returned an accuracy rate of 98.87%. When this procedure was run on the EMNIST dataset, it had an accuracy for digit and letter recognition across 47 balanced letter classes of 99.79%, comparable to state-of-the-art accuracy of 99.79%.

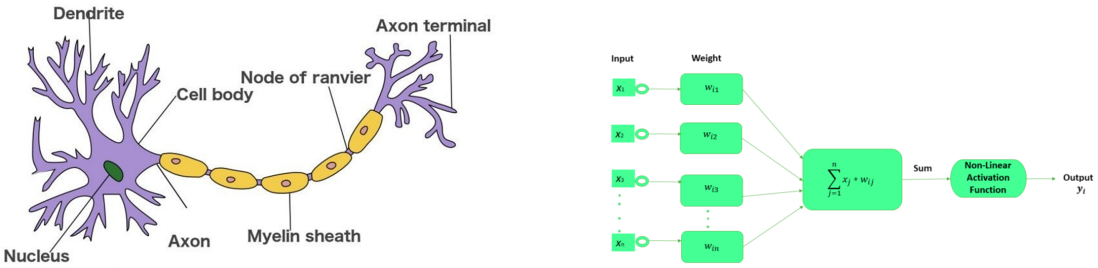
Naveen Garg [3] developed a multi-layered, multi-input neural network for handwritten Gurumukhi character recognition. He applied a dataset containing test sets that equaled ten sets of training photos, each containing the forty characters in different typefaces and handwriting. In this study, the neural network model was based on feedforward backpropagation methodology with definite train settings like 500 epochs and a target of 0.001, and an architecture composed of three layers: input, hidden, and output. In the training, every feature for each character needed to be calculated and then stored. The results showed that the accuracy of recognition was quite variable across samples and went up to 93% accuracy for some character sets. This showed the model's effectiveness and the possible areas for enhancement in training accuracy and memory allocation.

3 Background Study

3.1 Artificial Neural Network

Artificial neural networks are popular deep learning techniques that simulate the mechanism of learning in biological organisms. The cells of the human nervous system are called neurons. The neurons are connected to one another with the use of axons and dendrites, and the connecting regions between axons and dendrites are referred to as synapses. These connections are illustrated in Figure 3.1a. The strengths of synaptic connections often change in response to external stimuli. This change is how learning takes place in living organisms. This biological mechanism is simulated in artificial neural networks, which contain computation units that are referred to as neurons. The computational units are connected to one another through weights, which serve the same role as the strengths of synaptic connections in biological organisms. Each input to a neuron is scaled with a weight, which affects the function computed at that unit. This architecture is illustrated in Figure 3.1b. An artificial neural network computes a function of the inputs by propagating the computed values from the input neurons to the output neuron(s) and using the weights as intermediate parameters. Learning occurs by changing the weights connecting the neurons. Similar to how learning in biological organisms requires external stimuli, in artificial neural networks the external stimulus comes from training data that shows instances of input-output pairings for the function that has to be learned. The training data provides feedback to the correctness of the weights in the neural network depending on how well the predicted output (e.g., probability of carrot) for a particular input matches the annotated output label in the training data. The weights are changed

carefully in a mathematically justified way so as to reduce the error and make the predictions more correct in future iterations[16].



(a) Biological Neuron [Source [17]] (b) Artificial Neural Network

Figure 3.1: Biological Neuron and Artificial Neural Network

3.2 Working of a neural network

Neural networks are complex systems that mimic some features of the functioning of the human brain. It is composed of an input layer, one or more hidden layers, and an output layer made up of layers of artificial neurons that are coupled as shown in figure 3.2[Source [18]]. The two stages of the basic process are called backpropagation and forward propagation.

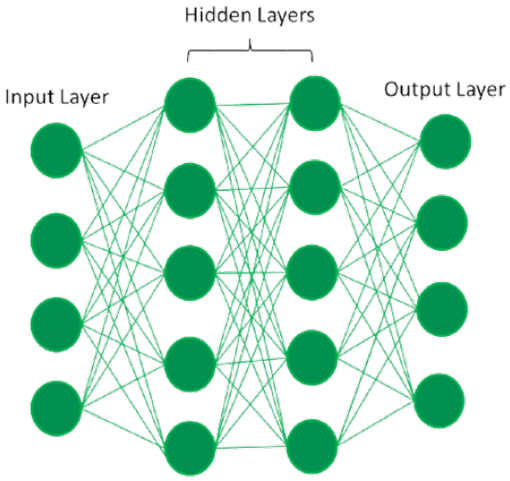


Figure 3.2: A Fully Connected Neural Network, Source

3.2.1 Forward propagation

- **Input Layer** :Each feature in the input layer is represented by a node on the network, which receives input data.

- **Weights and connections** :The weight of each neuronal connection indicates how strong the connection is. Throughout training, these weights are changed.

- **Hidden Layers**: Each hidden layer neuron processes inputs by multiplying them by weights, adding them up, and then passing them through an activation function. By doing this, non-linearity is introduced, enabling the network to recognize intricate patterns.

- **Output**: The final result is produced by repeating the process until the output layer is reached.

3.2.2 Backpropagation

- **Loss calculation**: The network's output is evaluated against the real goal values, and a loss function is used to compute the difference. For a regression problem, the Mean Squared Error (MSE) is commonly used as the cost function. The Loss function 3.1 is as follows:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3.1}$$

- **Gradient Descent**: Gradient descent is then used by the network to reduce the loss. To lower the inaccuracy, weights are changed based on the derivative of the loss with respect to each weight.

- **Adjusting weights**: The weights are adjusted at each connection by applying this iterative process, or backpropagation, backward across the network.

- **Training**: During training with different data samples, the entire process of forward propagation, loss calculation, and backpropagation is done iteratively, enabling the network to adapt and learn patterns from the data.

- **Activation functions**: Model non-linearity is introduced by activation functions like the rectified linear unit (ReLU) or sigmoid. Their decision on whether to "fire" a neuron is based on the whole weighted input.

3.3 Learning of a Neural Network

A machine can learn by supervised, unsupervised and reinforcement learning process.

3.3.1 Supervised Learning

- Supervised learning is a machine learning method in which models are trained using labeled data. In supervised learning, models need to find the mapping function to map the input variable (X) with the output variable (Y) as shown in function 3.2.

$$y = f(x) \tag{3.2}$$

- Supervised learning can be used for two types of problems: Classification and Regression.

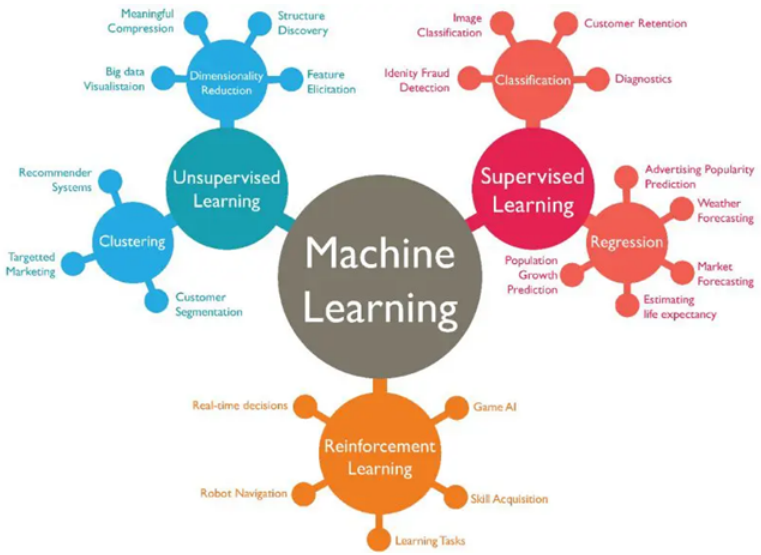


Figure 3.3: Types of machine learning process, Source [19]

3.3.2 Unsupervised Learning

- Unsupervised learning is another machine learning method in which patterns inferred from the unlabeled input data. The goal of unsupervised

learning is to find the structure and patterns from the input data. Unsupervised learning does not need any supervision. Instead, it finds patterns from the data by its own.

- Unsupervised learning can be used for two types of problems: Clustering and Association.

3.3.3 Reinforcement Learning

- Reinforcement learning is a type of machine learning method where an intelligent agent (computer program) interacts with the environment and learns to act within that.

- During this time, the actions made by the agent are followed by rewards or penalizations, leading to adaptation of its future activities through trial and error. The learning process involves exploration into the variants of activities and an exploitation of known manners that return high rewards.

3.4 Types of Neural Network

There are various types of neural networks, each designed for specific tasks and types of data. Some of the most common types of neural networks are

1. Feedforward Neural Network
2. Convolutional Neural Network
3. Recurrent Neural Network
4. Long Short term memory Networks
5. Generative Adversarial Networks (GANs)
6. Autoencoders
7. Radial Basis Function Networks (RBFNs)
8. Deep Belief Networks (DBNs)

9. Trasformer Networks

In this project context, we will focus on Convolutional Neural Networks (CNNs), a specialized type of artificial neural network that has revolutionized the field of computer vision and image processing. We will explore the architecture, key components, and fundamental principles that make CNNs particularly effective for tasks involving structured grid data, such as images.

3.5 Convolutional Neural Network

Convolutional Neural Network (CNN) is the extended version of artificial neural networks (ANN) which is predominantly used to extract the feature from the grid-like matrix dataset. Such a network can be composed of convolutional layers, pooling layers, fully connected layers and loss layers. In a typical CNN architecture, each convolutional layer is followed by a Rectified Linear Unit, then a Pooling layer then one or more convolutional layer and finally one or more fully connected layer. A characteristic that sets apart the CNN from a regular neural network converts the input in a one-dimensional array which makes the trained classifier less sensitive to positional changes. The detailed architecture [Fig 3.4]of a typical CNN is given below.

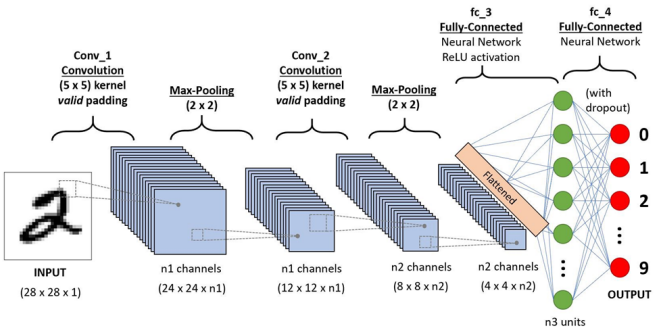


Figure 3.4: Architecture of a CNN, Source [20]

3.5.1 Components of CNN

- **Input layers**: Through this layer input data is inserted to the model. In CNN, Generally, the input will be an image or a sequence of images. This

layer holds the raw input of the image.

- **Convolutional layers**: This is the layer, which is used to extract the feature from the input dataset [Fig 3.5]. It applies a set of learnable filters known as the kernels to the input images. The filters/kernels are smaller matrices usually 2×2, 3×3, or 5×5 shape. It slides over the input image data and computes the dot product between kernel weight and the corresponding input image patch. The output of this layer is referred as feature maps. Suppose we use a total of 12 filters for this layer for an image of dimension 28x28x1 we'll get an output volume of dimension 28 x 28 x 12.

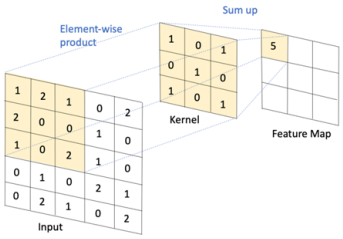


Figure 3.5: Convolutional Operation, Source [21]

The formula [22] to find the output feature map size after convolution operation as below

$$h' = \left\lfloor \frac{h - f + p}{s} + 1 \right\rfloor \tag{3.3}$$

$$w' = \left\lfloor \frac{w - f + p}{s} + 1 \right\rfloor \tag{3.4}$$

Where h' denotes the height of the output feature map, w' denotes the width of the output feature map, h denotes the height of the input image, w denotes the width of the input image, f is the filter size, p denotes the padding of convolution operation and s denotes the stride of convolution operation.

- **Pooling Layers**: The pooling layers are used to sub-sample the feature maps (produced after convolution operations), i.e. it takes the larger size feature maps and shrinks them to lower sized feature maps. While shrinking the feature maps it always preserve the most dominant features (or information) in each pool steps.These are used after convolutional layers to reduce

the spatial dimensions of the feature map, hence reducing the computational load and overfitting. The most common one is Max Pooling, which takes the maximum value in every patch of a feature map. For example, a 2x2 max-pooling layer will take the 32x32 feature map and bring it down to 16x16. This reduces the dimensionality of representations and makes them more computationally manageable. Another variant is the Average Pooling, that calculates the average of each patch; it is not as commonly used as Max Pooling.

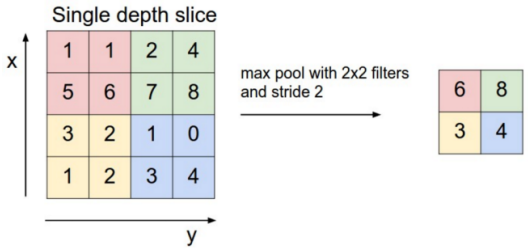


Figure 3.6: Pooling operation [Source [23]]

- **Activation Layers** :By adding an activation function to the output of the preceding layer, activation layers add nonlinearity to the network. it will apply an element-wise activation function to the output of the convolution layer. Some common activation functions are relu, sigmoidal, leaky relu, softmax and tanh.

- **Fully Connected layer**: After multiple convolutional and pooling layers are done, the top level reasoning in the neural network is usually done through fully connected layers. These are the identical layers to standard feedforward neural network layers, where each neuron is fully connected to the previous layer. These layers add all features that the model extracts to predict an outcome. Classifications output a softmax, applied after the final fully connected layer, to convert the output into a probability distribution over class labels.

- **Batch Normalization**: Another important building block of many CNNs is batch normalization, which enables stabilizing a training process of a CNN and accelerating it considerably. This is a technique normalizing the

input to each layer to have a mean of zero and a standard deviation of one to mitigate issues related to internal covariate shift.

The overall summary for any ordinary CNN architecture would be: an input layer that receives the image data, succeeded by a number of convolutional and activation layer pairs, usually followed by ReLU optional succeeded by pooling. After that come one or more fully connected layers outputting the final classification probabilities. Some regularization techniques, including Dropout, may be applied to prevent overfitting by randomly setting during training a fraction rate of input units to zero.

3.5.2 Activation Functions

In CNN architecture, after each learnable layers (layers with weights, i.e. convolutional and FC layers) non-linear activation layers are used. The non-linearity behavior of those layers enables the CNN model to learn more complex things and manage to map the inputs to outputs non-linearly. The important feature of an activation function [22] is that it should be differentiable in order to enable error backpropagation to train the model. The most commonly used activation functions in deep neural networks (including CNN) are described below.

3.5.2.1 Rectified Linear Unit (ReLu)

It [22] is the most commonly used activation function[Fig 3.8] in Convolutional Neural Networks. It is used to convert all the input values to posititve numbers. The advantage of ReLU is that it requires very minimal computation load compared to others. The mathematical representation (3.5) of ReLU is:

$$f(x) = max(0, x) \tag{3.5}$$

3.5.2.2 Sigmoidal

The sigmoid activation function [Fig 3.7]takes real numbers as its input and bind the output in the range of [0,1]. The curve of the sigmoid function is of 'S' shaped.

The mathematical representation of sigmoid is:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3.6}$$

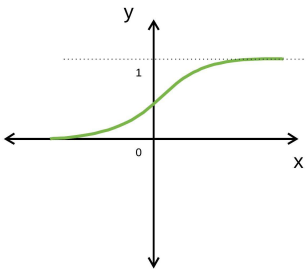


Figure 3.7: Sigmoid

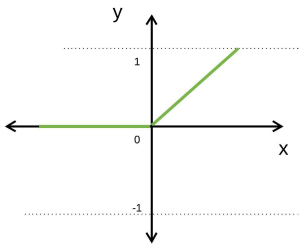


Figure 3.8: ReLU

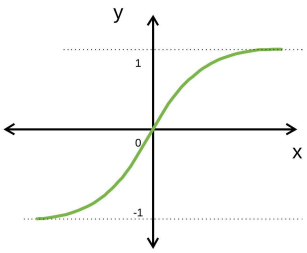


Figure 3.9: Tanh

3.5.2.3 Tanh

The Tanh activation function [Fig 3.9] is used to bind the input values (real numbers) within the range of [-1,1]. The mathematical representation of Tanh is:

$$f(x) = \tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{3.7}$$

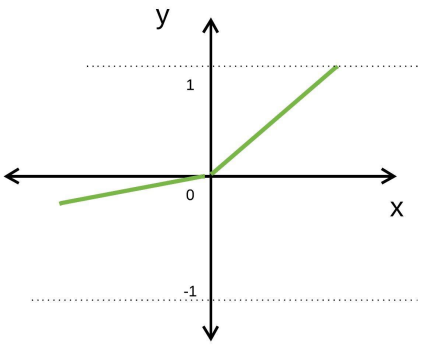


Figure 3.10: Leaky ReLU

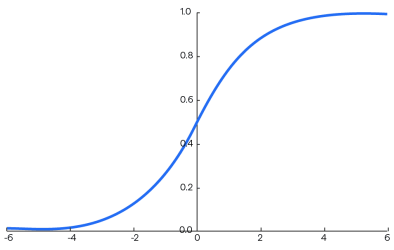


Figure 3.11: Softmax

3.5.2.4 Leaky Relu

Unlike ReLU, a Leaky ReLU [22] activation function does not ignore the negative inputs completely, rather than it down-scaled those negative inputs [Fig

3.10]. Leaky ReLU is used to solve Dying ReLU problem. The mathematical representation of Leaky ReLU is:

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha x & \text{if } x \leq 0 \end{cases} \tag{3.8}$$

3.5.2.5 Softmax

The softmax function [24] is used in the output layer for multi-class classification problems [3.11]. It converts the raw output scores of the network into probabilities, defined as:

$$f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{3.9}$$

3.6 Cross Entropy Loss functions

Cross-entropy loss, also called log loss function is widely used to measure the performance of CNN model.It is widely used as an alternative of squared error loss function in the multi-class classiffcation problems. It uses softmax activations in the output layer to generate the output within a probability distribution.

3.6.1 Binary Cross Entropy

Binary cross-entropy [25, 26] is used for binary (dual class) classification tasks. It is defined as:

$$L = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)] \tag{3.10}$$

where N is the number of samples, y_i is the true label, and $\hat{y}_i$ is the predicted probability.

3.6.2 Categorical Cross Entropy

Categorical cross-entropy [25, 26]is used for multi-class classification tasks. It is defined as:

$$L = -\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij}\log(\hat{y}_{ij}) \tag{3.11}$$

where N is the number of samples, C is the number of classes, y_{ij} is the binary indicator (0 or 1) if class label j is the correct classification for sample i, and $\hat{y}_{ij}$ is the predicted probability for class j for sample i.

3.7 Optimizers

A CNN model will choose the learning algorithm in the learning process and make some improvements. The key concept as a whole is the minimization of error based on learnable parameters like weights and biases. Gradient-based learning methods naturally go with the learning of CNNs, because model parameters are updated at each training epoch while finding a locally optimal solution.

3.7.1 Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent (SGD) [27] updates the parameters θ of the model for each training separately by moving them in the direction of the negative gradient of the loss function $J(\theta)$ with respect to the parameters as shown in equation 3.12:

$$\theta = \theta - \eta\nabla_{\theta}J(\theta) \tag{3.12}$$

where η is the learning rate.

3.7.2 RMSprop

RMSprop (Root Mean Square Propagation) [28] is an adaptive learning rate method that divides the learning rate by an exponentially decaying average of squared gradients as expressed in equation 3.13:

$$v_t = \beta v_{t-1} + (1-\beta)g_t^2 \tag{3.13}$$

The parameters are updated as follows:

$$\theta_t = \theta_{t-1} - \eta \frac{g_t}{\sqrt{v_t} + \epsilon} \tag{3.14}$$

where η is the learning rate, β is a decay rate, and ϵ is a small constant

3.7.3 Adam

Adam (Adaptive Moment Estimation)The adaptive moment estimation is Adam [22] which is a method of learning that computes a varying learning rate gradient in the neural network such as combining momentum with RMSprop benefits. It maintains two moving averages, m_t and v_t:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{3.15}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{3.16}$$

The parameters are updated as follows:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{3.17}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{3.18}$$

$$\theta_t = \theta_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{3.19}$$

where η is the learning rate, β_1 and β_2 are hyperparameters, and ϵ is a small constant to prevent division by zero.

3.8 Deep trasfer learning models

In image classiffication, we assume that the input image contains a single object and then we have to classify the image into one of the pre-selected target classes by using CNN models.In this project six pre-trained trasfer learning models under deep CNN architecture are studied, LeNet5, ResNet50, MobileNet, InceptionV1, ShuffleNet and EfficientNet.

3.8.1 LeNet5

The LeNet-5 [22] is one of the earliest CNN architecture. The LeNet-5 has 5 weighted (trainable) layers, that is, three convolutional layer and two FC layers. Among them, each of first two convolution layer is followed by a max-pooling layer (to sub-sample the feature maps) and afterward, the last convolution layer is followed by two fully connected layers. The last layer of those fully connected layers is used as the classiffier.. The architecture of LeNet-5 is shown in Fig 5.1

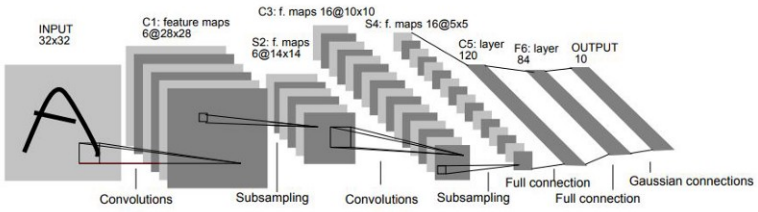


Figure 3.12: LeNet5

3.8.2 ResNet50

A deep neural network model, ResNet50 performs image feature extraction using convolutional layers. Batch normalization and ReLu activation follow these layers thereby extracting contours, textures and patterns on the photographs. Identity block as well as convolution block are essential components; the identity block learns residual functions while convolution block diminishes filters.The fully connected layers are the final part of ResNet50 and deal with the input/output information for predicting the class probabilities at the end. The final fully connected layers' output goes through a softmax activation function to give us the final class probabilities. This architecture is intended to transform the input image effectively. The following figure 3.13, (Source [29]) describes the architecture of ResNet50.

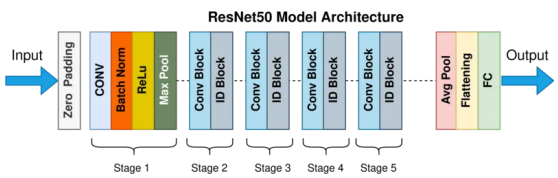


Figure 3.13: ResNet50 Architecture

3.8.3 MobileNetV1

MobileNetV1 is one of the best known convolutional neural network designs which was made for deployment easily on smartphones and other small devices, thanks to its two new ideas: depth wise separable convolutions where full convolutions are broken down into depth wise and point wise convolutions, which greatly minimize model's computational cost and size.The structure begins with one first fully-convolutional layer; then, it contains 13 depthwise separable convolution blocks followed by average pooling together fully connected layer before the softmax classifier. Simply put, MobileNetV1 combines two crucial but cheap computational hyperparameters; width multiplier and resolution features; showing flexibility in trading off between precision, size, and performance.Sockets consist of a single fully convolutional layer immediately followed by thirteen depth-wise separable convolution layers on a separate occasion before average-pooled across width then densely connected via the softmax classifier neuron at its end [Fig 3.14 Source [30]].

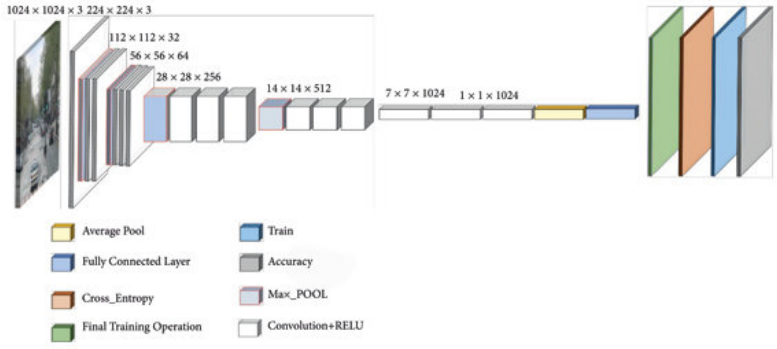


Figure 3.14: MobileNet Architecture

3.8.4 EfficientNet

Besides, EfficientNet is a family of neural convolutional networks designed to achieve the best performance with reduced computational resources. First proposed by Google Research in 2019, it basically invented a new method of scaling up, generally referred to as the compound scaling method, but which scales network depth, width, and input resolution using a compound coefficient. It

has a stem, a body, and a head. The stem is an initial convolution, the body a series of MBConv blocks with dep separable convolutions and squeeze-and-excitation layers; finally, the head contains final convolution and classification layers. EfficientNet-B0 forms the base model; six scaled larger models based on it are available: B1-B6. Compound scaling method balances scaling of width, depth, and resolution. Thus, for every doubling of depth of the network, increasing its width by about 1.1 times, while resolution by approximately 1.15 times may be handy eventually. Compound scaling jointly with depth-wise separable convolutions and new squeeze-and-excitation modules made it easy for EfficientNet to bring state-of-the-art-accuracy with huge reductions in model size and computational cost compared with prior models [Fig 3.15, Source [31]].

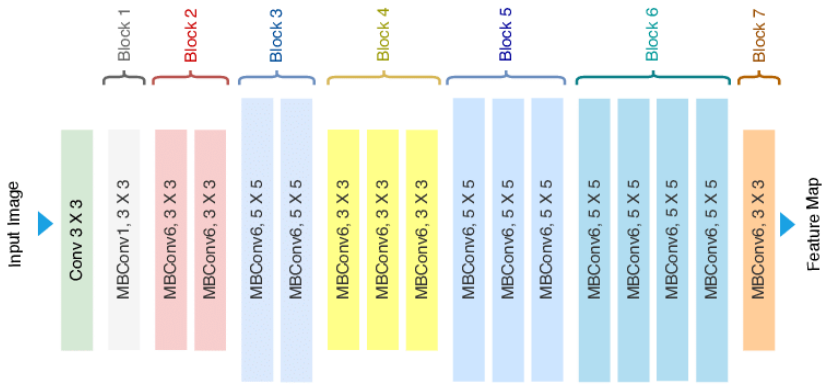


Figure 3.15: EfficientNet Architecture

3.8.5 InceptionV1

Inception v1 was introduced in 2014. This architecture came to push the community of convolutional neural networks [32] further with their new "inception modules." These modules are parallel paths of convolutions of filter sizes 1x1, 3x3, and 5x5, also a max-pool layer whose outputs are concatenated. Another important notion here is businesses of 1x1 convolutions before larger convolutions for reducing dimensions, greatly reducing computational cost. The network's depth is 22 layers, but considering the pooling layers, it goes as deep as 27 layers. It is perfectly balanced in terms of depth versus width. This network starts off, like any other standard network, with some convolutional and max-pooling layers; then some modules proposed in this architecture are passed through; and

then ends with average global pooling, avoiding fully connected layers to trim down the number of parameters. Here, the vanishing gradient problem in this deep network is handled through the introduction of two auxiliary classifiers into intermediate layers only during training. For the final model, we have linearly stacked 9 Inception modules with occasional max-pool layers for spatial dimension reduction. This meant that GoogLeNet could only achieve state-of-the-art performance on the ILSVRC 2014 classification challenge with 5 million parameters, a lot shallower than the former architectures of AlexNet with 60 million and VGGNet with 180 million parameters; hence, this would turn out to be a very accurate and computationally efficient objective.The architecture is as shown in below. [Fig 3.16, Source [33]].

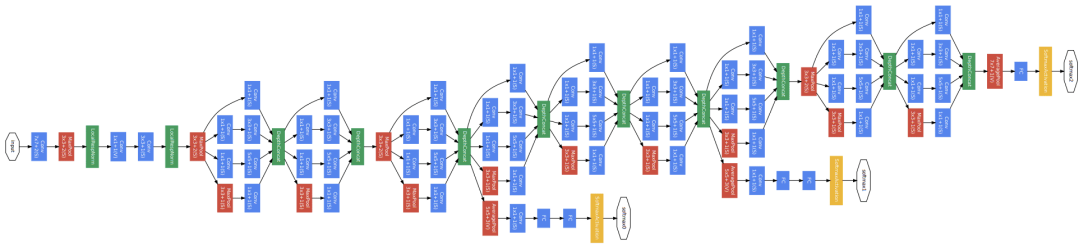


Figure 3.16: InceptionV1 architecture

3.8.6 ShuffleNet

ShuffleNet is a light weight network which applies a special block called ShuffleNet unit [34]. This contains a 1 x 1 group convolution, then a channels interchange spike, followed by a 3 x 3 depth-wise convolution and finally, a 1 x 1 group convolution to recover the original channel dimension. In addition, it has three main stages where spatial dimensions are shrunk in the first units of each and the number of channels rises up to two times before passing through other stages.Standard 1x1 convolutions reduce computation when compared to point-wise group convolutions and channel shuffle operationsremain effective in allowing the network to learn complex features. ShuffleNet can also be adjusted to varying sizes by simply changing the number of channels across the whole network. Owing to this kind of architecture, ShuffleNet is able to leverage a higher number of channels for a set computational cost compared to other light-weight networks. The architecture

of shufflenet unit [Fig 3.17, source [35]] and channel shuffle operation [Fig 3.18, Source [35]] are as shown in below.

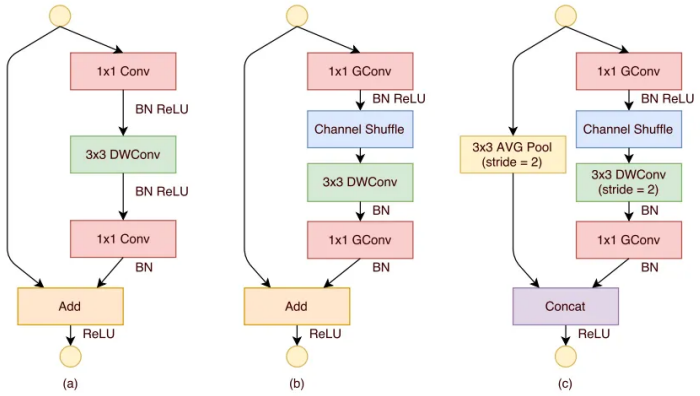


Figure 3.17: (a) A bottleneck unit with depthwise convolution (3x3 DWConv), (b) A ShuffleNet unit with pointwise group convolution and channel shuffle, (c) A ShuffleNet unit with stride of 2

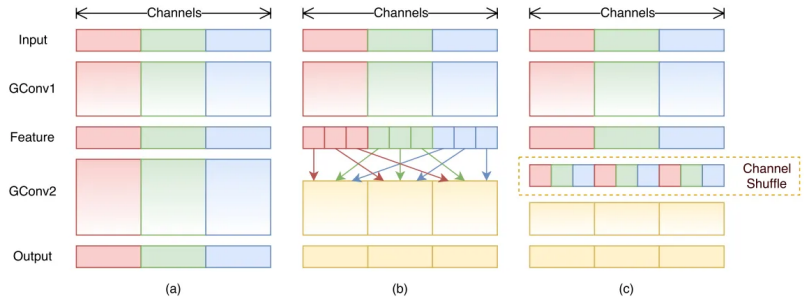


Figure 3.18: (a) Illustrates a situation of two stacked group convolution layers, (b) Shows that the group convolution is allowed to obtain input data from different groups, (c) Sets up the feature map from the previous group layer, which is then implemented by a channel shuffle operation

4 Methodology

4.1 Proposed Work

In Figure 4.1 overall workflow of the proposed work is shown. To achieve accurate recognition, the Convolutional Neural Network (CNN) architecture is employed. CNN models are well suited for image classification tasks due to their ability to capture local patterns and spatial heirarchies. In addition to CNN (LeNet5), five state-of-the-art architectures, ResNet50, MobileNetV1, InceptionV1, ShuffleNet and EfficientNet are also incorporated to enhance the performance of the system. The training process involves the feeding of the preprocessed dataset

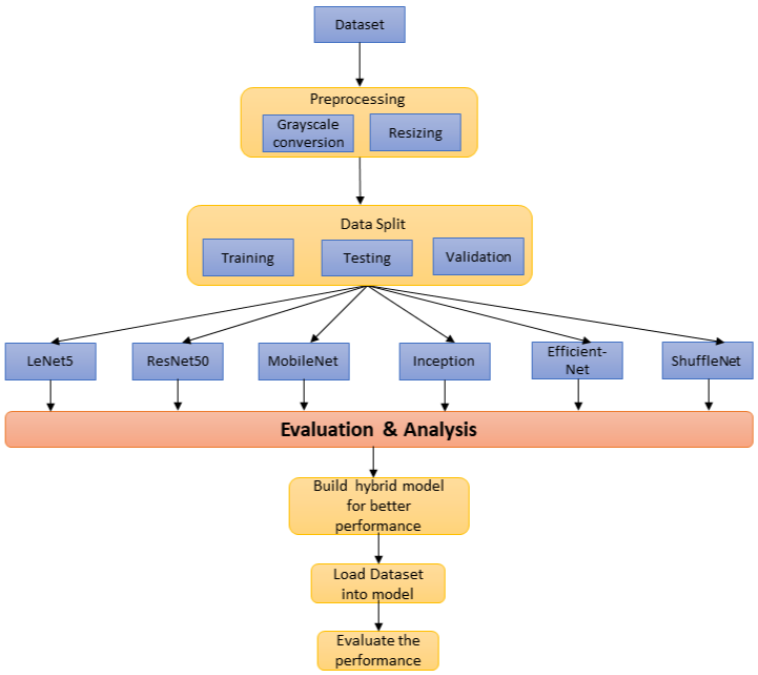


Figure 4.1: Workflow diagram of the proposed work

through a number of varied deep learning models. Through many iterations, they become proficient in these complex features and patterns of images. Transfer learning enforces the use of previously pre-trained weights of such models: LeNet5, ResNet50, MobileNet, Inception, ShuffleNet, and EfficientNet; the latter have weights pre-trained on large datasets of images. A hybrid model is also developed referring to the related best outcomes with pre-trained models, including the incorporation of some extra layers for working better. In this way, this approach is able to achieve better generalization and fast convergence.

After the training phase has been completed, the trained models are examined using another test set so that their performance can be assessed, with evaluation measures such as accuracy rate, precision rate and recall rate, among others, being used.

The proposed handwritten character recognition system using hybrid model of ResNet50, mobilenet and shufflenet demonstrates excellence in terms of accuracy and efficiency. the system's ability to accurately classify handwritten alphabets can greatly help in documentation, baking sectors, healthcare industries and many more organizations where handwritten documents are used regularly. It is modular; therefore, flexibility and scalability in view of the envisaged future improvements will also find applications in a wide spectrum of uses.

4.2 Proposed Model Architecture

After analyzing results of pretrained models, four hybrid models are constructed using MobileNet, ResNet, ShuffleNet, and Efficientnet, four hybrid models are built after these model's performances are examined. The architecture of proposed models are shown below.

4.2.1 Hybrid of MobileNet and ResNet50

The model combines two pre-trained models, MobileNet and ResNet50, starting from a single input layer. The outputs of these models are joined together through concatenation layer and then processed through several dense (fully connected) layers. Between these dense layers, batch normalization and dropout layers are used to improve the model's performance and prevent overfitting. It is depicted in Figure 4.2.

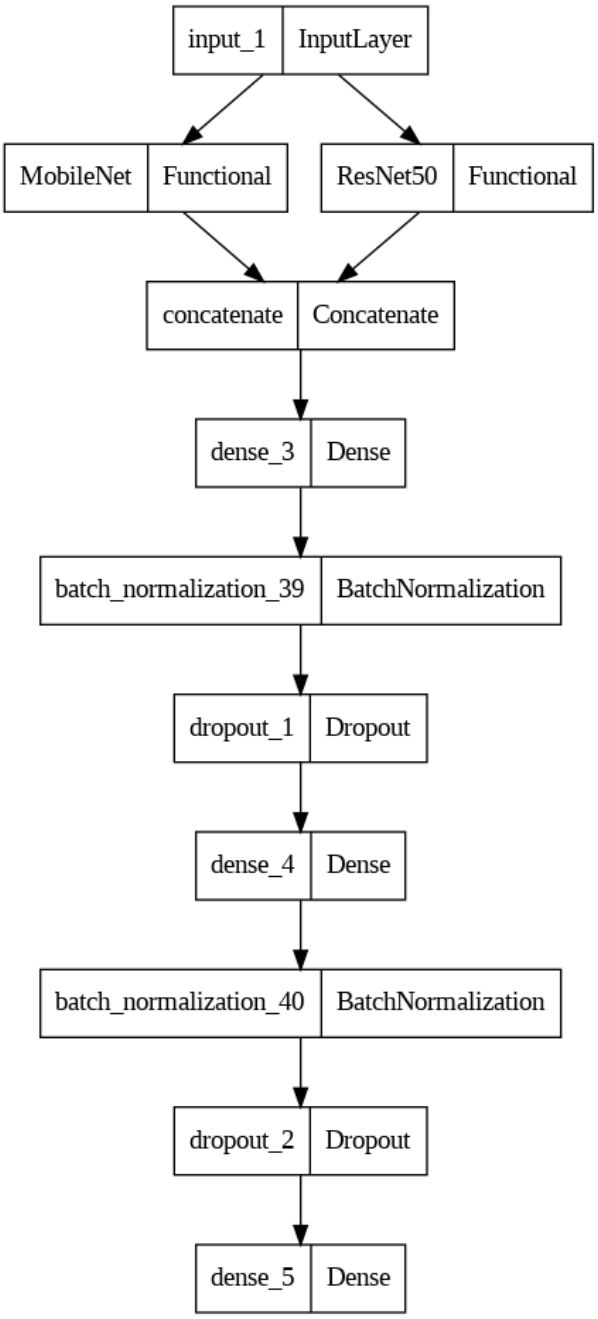


Figure 4.2: Hybrid of MobileNet and ResNet50

4.2.2 Hybrid of MobileNet nad ShuffleNet

This neural network model starts with a single input layer and uses two pre-trained models, MobileNet and ShuffleNet. Each model processes the input separately, and their outputs are passed through dense (fully connected) layers. The results from these layers are then combined into one. This combined output goes through a series of dense layers, with batch normalization and dropout layers. Figure 4.3 shows the above explanation.

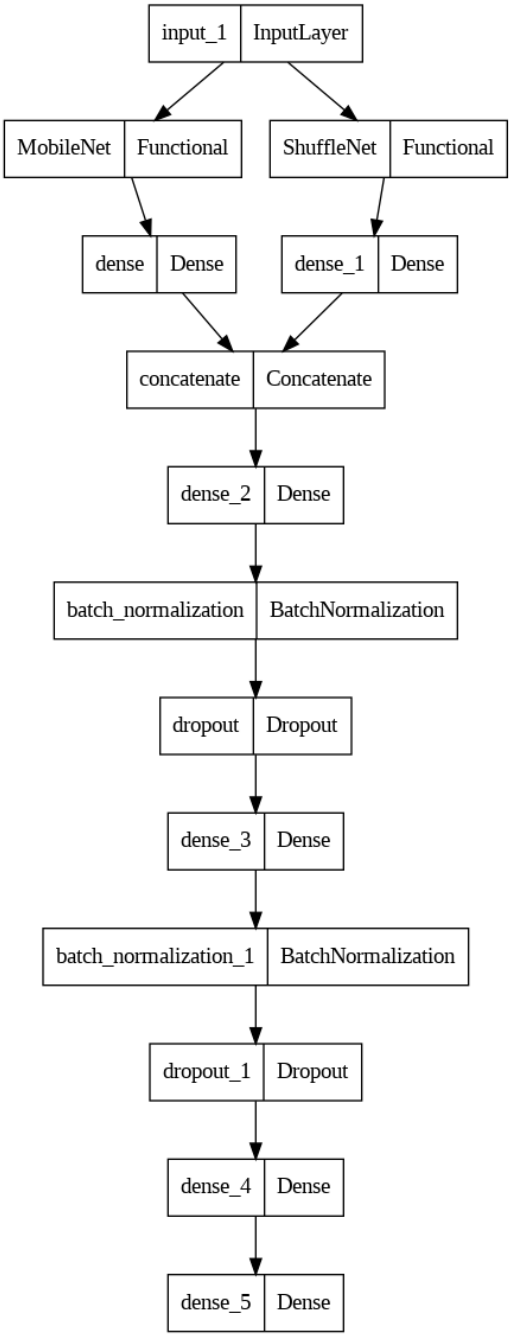


Figure 4.3: Hybrid of ResNet50 and ShuffleNet

4.2.3 Hybrid of MobileNet, EfficientNet, and ResNet50

This model combines MobileNet, EfficientNet, and ResNet50, three powerful convolutional neural networks, each followed by Functional and Flatten layers as shown in Figure 4.4. Their outputs are concatenated and fed through alternating Dense, BatchNormalization, and Dropout layers.The alternating pattern of Dense, BatchNormalization, and Dropout layers suggests a careful approach to managing the learning process, reducing overfitting, and improving generalization.

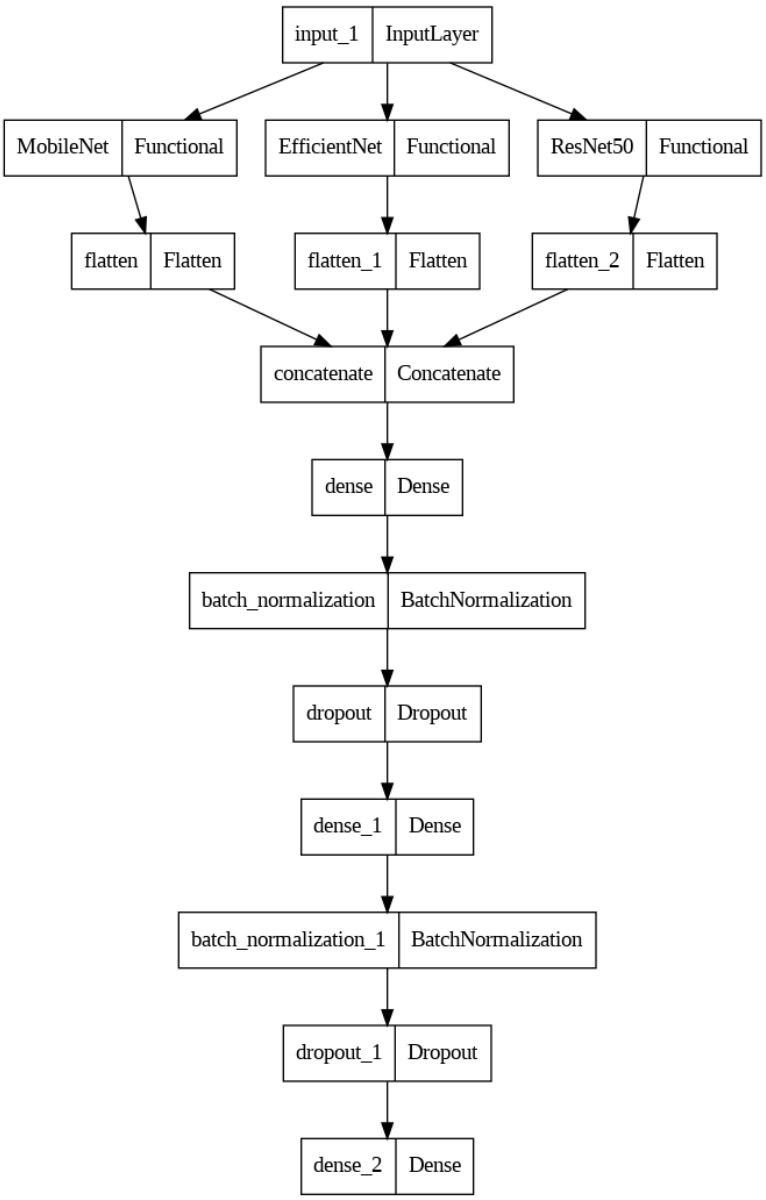


Figure 4.4: Hybrid of MobileNet, EfficientNet, and ResNet50

4.2.4 Hybrid of MobileNet,ShuffleNet, and ResNet50

This model presents an ensemble model that utilizes MobileNet, ShuffleNet, and ResNet50 as its core feature extractors is shown in Figure 4.5. The overall structure mirrors the first image, with each network followed by Functional and Flatten layers before concatenation. The subsequent layers follow a similar pattern of Dense, BatchNormalization, and Dropout layers.

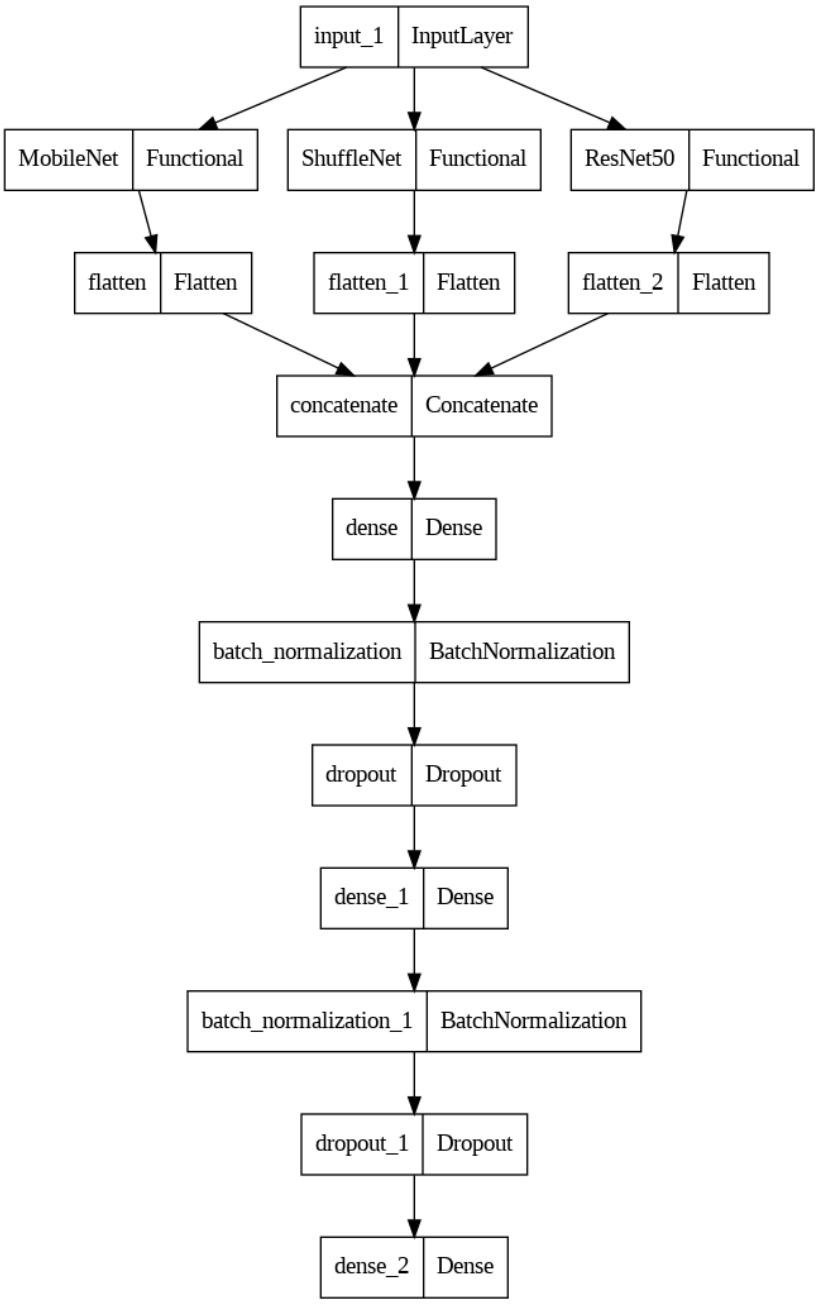


Figure 4.5: Hybrid of MobileNet, ShuffleNet, and ResNet50

The working of the additional layers in each model is as follows

- **Input Layer**: The models start with an input layer, which processes images in grayscale that are of dimension 28x28 pixels. The models are dynamic in terms of batch size; hence, it is processing a different number of images in every batch during training or inference.

- **Concatenation**: This layer Combines the feature maps from the respective pretrained models into a single tensor. This allows the model to leverage the strengths of both architectures by considering features learned from all respective networks.

- **Flatten Layers**: This layer Transforms multi-dimensional inputs into a one-dimensional vector. This is often used before the dense layers to convert the 2D feature maps into a format that can be processed by fully connected layers.

- **Dense Layer**: These are fully connected layers that perform classification or regression tasks based on the features learned in previous layers. They adjust weights during training to minimize the error in predictions.

- **Batch Normalization**: This layer standardizes the inputs to a layer for each mini-batch, stabilizing the learning process and reducing the number of training epochs needed. This normalization helps in maintaining the mean output close to 0 and the output standard deviation close to 1.

- **Dropout**: It reduces overfitting by randomly setting a fraction of input units to zero during each update cycle. This prevents the network from becoming too dependent on specific neurons, thereby promoting generalization.

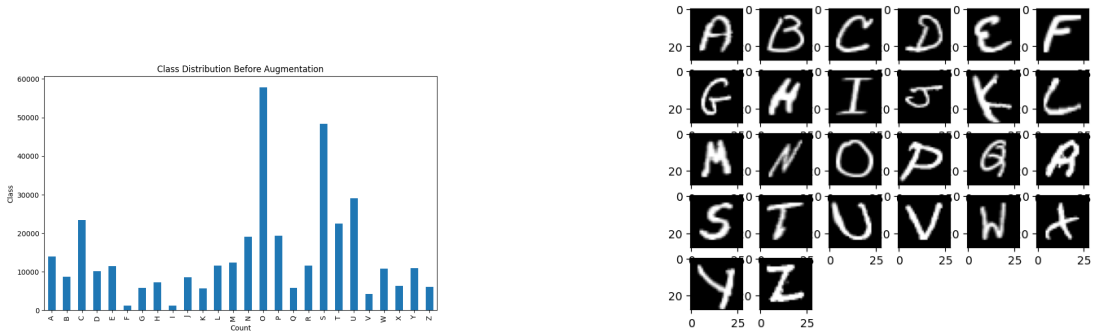
- **Final Dense Layer**: The final dense layer in each model performs class classification task. Most likely, such a layer uses the softmax activation function to output a probability distribution over the respective possible classes for a particular dataset.

4.3 Dataset Used

The dataset used for training and evaluation of models is collected from Kaggle [36]. Each letter is of uniform size i.e. 28 x 28. It is shown in Table 4.1. The dataset is visualised in Figure 4.6

Table 4.1: Dataset of alphabets

Dataset	**Total samples**	**Classes**	**Training Samples**	**Validation Samples**	**Testing Samples**
Kaggle Alphabet	26	3,71,490	2,97,000	37,245	37,245



(a) (b)

Figure 4.6: (a) Class distribution and (b) Sample images of dataset

4.4 Data Splitting

Data splitting is the process of splitting the data into two or more datasets. Typically, three parts is used for training, validation and testing the dataset and the other part for validating the dataset.

4.4.1 Training

Training dataset is used to train the model. Here we are using 80 percent for training dataset.

4.4.2 Validation

In validation, a portion of the data is set aside to evaluate the model's performance on unseen examples. It helps detect overfitting, guide hyperparameter tuning, and assess the model's ability to generalize. Here we are using 10 percent for validation dataset.

4.4.3 Testing

Testing dataset is used to test the model. Here we are using 10 percent for testing dataset.

After the data pre-processing and data splitting all the data are normalized such that the attribute values (pixel values) of the images lie between 0 to 1. Data normalization allows the pre-trained model to converge to better weights and then leads to a accurate model.

4.5 Hyperparameter Tuning

In deep learning ,hyperparameters define the structures of a model and training rules are defined before the process of training, moderated by a user, and do not emerge during the process of learning from data. Hyperparameters used for conducting experiments for this report are:

- **Learning rate**: Controls the step size during optimization. In this process, we have used a learning rate of 0.001.
- **Batch size**: It is the number of training examples used in one iteration. The batch size used in this process is 128.
- **Number of epochs**: It defines the number of times the model processes the entire dataset. For our experiments, number of epochs vary modelwise.
- **Activation functions**: ReLU is an activation function that outputs the input if it's positive, otherwise zero, introducing non-linearity and mitigating the vanishing gradient problem. Softmax: It is an activation function

that converts logits into a probability distribution, used in the output layer of classification models.

- **Adam**: It is an optimizer algorithm for training deep learning models. It's popular for its ability to handle noisy data and achieve good results quickly, often requiring less hyperparameter tuning than other optimizers.

- **Dropout Rate**: The dropout rate is the fraction of neurons randomly set to zero during each training iteration to prevent overfitting. The droput rate used to train our model is 0.5.

5 Result Analysis and Discussion

5.1 Experimental setup and Evaluation Measures

In this phase, the experimental setup, result and evaluation measure used for result will be discussed. The following describes the experimental setup used for the project.

5.1.1 Hardware Requirements

Table 5.1: Hardware Requirements

CPU	Intel Core i5-11300H 3.10GHz 3.11 GHz
RAM	16GB or above
GPU	Nvidia Geforce GTX 1650 graphics processor

5.1.2 Software Requirements

Table 5.2: Software Requirements

Operating System	Windows 11 (64-bit)
Package Installation	Jupyter Notebook(IDE),Python Libraries (numpy, pandas, scikit-sklearn, matplotlib, tensorflow, seaborn)
Programming Language	Python

5.2 Evaluation Measures

For multiclass classification validity measures such as:

i. **Accuracy**: Accuracy is the proportion of correct predictions across all classes. It is defined in Equation 5.1.

$$\text{Accuracy} = \frac{\sum_{i=1}^{N} TP_i}{\sum_{i=1}^{N} (TP_i + FP_i + FN_i + TN_i)} \times 100 \tag{5.1}$$

ii. **Precision**: Precision is the proportion of correct predictions for a specific class among all instances predicted as that class. It is defined in Equation 5.2.

$$\text{Precision}_i = \frac{TP_i}{TP_i + FP_i} \tag{5.2}$$

iii. **Recall**:Recall is the proportion of correct predictions for a specific class among all actual instances of that class. It is defined in Equation 5.3.

$$\text{Recall}_i = \frac{TP_i}{TP_i + FN_i} \tag{5.3}$$

iv. **F1 Score**: It is the harmonic mean of precision and recall. It is defined in Equation 5.4.

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5.4}$$

Where:

- TP_i (True Positive): Correctly identified positive samples for class i.
- TN_i (True Negative): Correctly identified negative samples for class i.
- FP_i (False Positive): Negative samples incorrectly identified as positive for class i.
- FN_i (False Negative): Positive samples incorrectly identified as negative for class i.

Experiments are carried out for all the six pretrained models and four hybrid models. Average results in terms of accuracy, precision, recall and F1 score are summarised in the Tables. From the Table 5.4 we have seen that out of all the models, the hybrid of MobileNet, EfficientNet and ResNet50 gives highest accuracy i.e. 99.69 percent. In case of performance metrics from Table 5.6, two hybrid model and ResNet50 provide highest precision, recall and F1 score.

Table 5.3: Accuracy Metrics for pretrained models

Model	Training Accuracy (%)	Validation Accuracy (%)	Testing Accuracy (%)
LeNet5	99.01	95.45	96.21
ResNet50	99.48	99.35	99.15
MobileNet	99.36	99.32	99.32
ShuffleNet	98.07	98.92	98.65
InceptionV1	98.19	98.66	98.64
EfficientNet	98.96	98.82	98.82

Table 5.4: Accuracy Metrics for Hybrid models

Base models	Training Accuracy (%)	Validation Accuracy (%)	Testing Accuracy (%)
MobileNet & ResNet50	99.86	99.62	98.65
MobileNet & ShuffleNet	99.09	98.90	98.90
MobileNet, EfficientNet & ResNet50	**99.88**	**99.66**	**99.69**
Hybrid of MobileNet, ShuffleNet & ResNet50	99.79	99.59	99.65

Table 5.5: Performance Metrics for pretrained models

Model	Precision	Recall	F1 score
LeNet5	0.97	0.97	0.97
ResNet50	1	1	1
MobileNet	0.99	0.99	0.99
ShuffleNet	0.99	0.99	0.99
InceptionV1	0.99	0.99	0.99
EfficientNet	0.99	0.99	0.99
Hybrid of MobileNet, ShuffleNet & ResNet50	**1**	**1**	**1**

Table 5.6: Performance Metrics for Hybrid models

Model	Precision	Recall	F1 score
MobileNet & ResNet50	**1**	**1**	**1**
MobileNet & ShuffleNet	0.99	0.99	0.99
MobileNet, EfficientNet & ResNet50	**1**	**1**	**1**
Hybrid of MobileNet, ShuffleNet & ResNet50	**1**	**1**	**1**

5.3 Accuracy and Loss Curves of the models

To train each model, we perform epochs with a batch size of 128. The figures shows Accuracy and Loss curve with respect to epochs on the dataset of kaggle letters by diffrenet models.

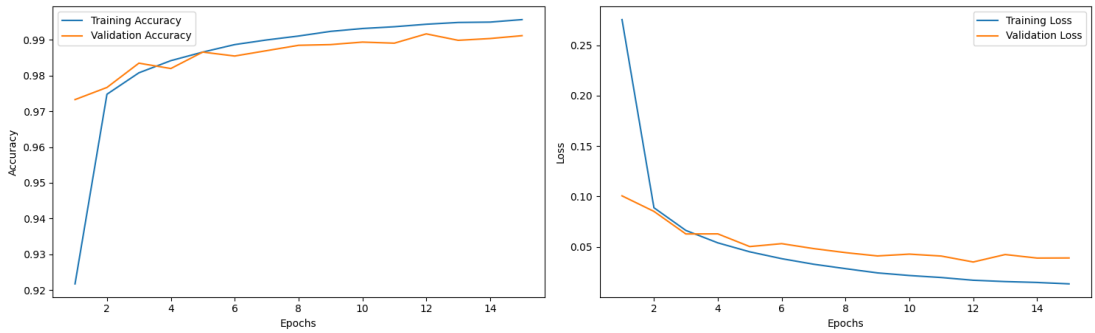


Figure 5.1: Accuracy and Loss of LeNet5

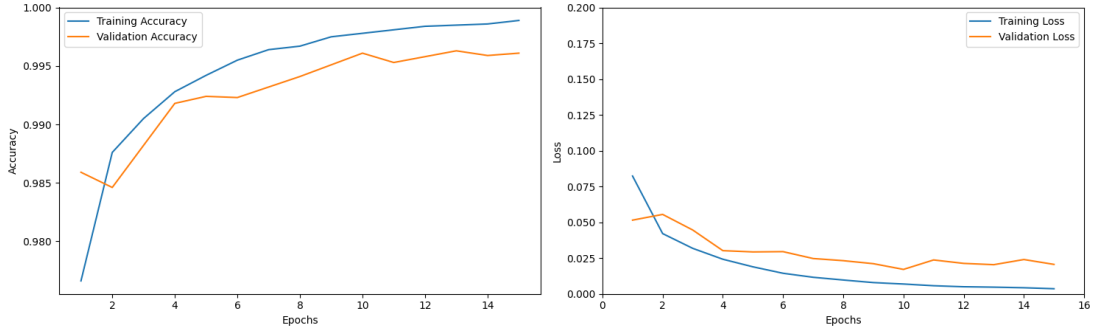


Figure 5.2: Accuracy and Loss of ResNet50

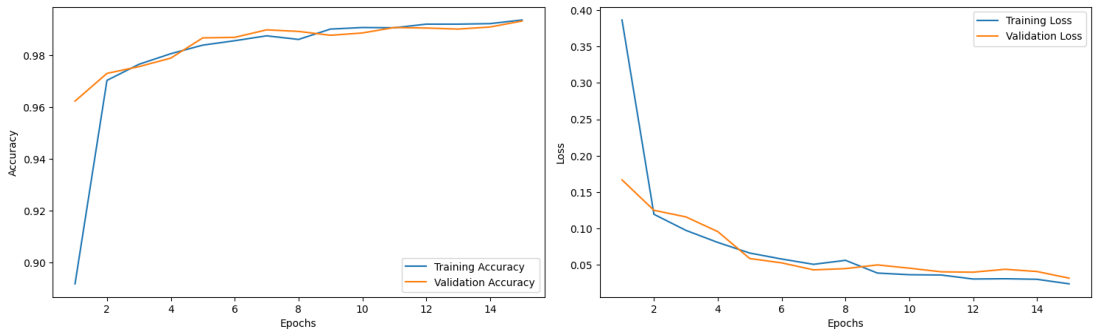


Figure 5.3: Accuracy and Loss of MobileNet

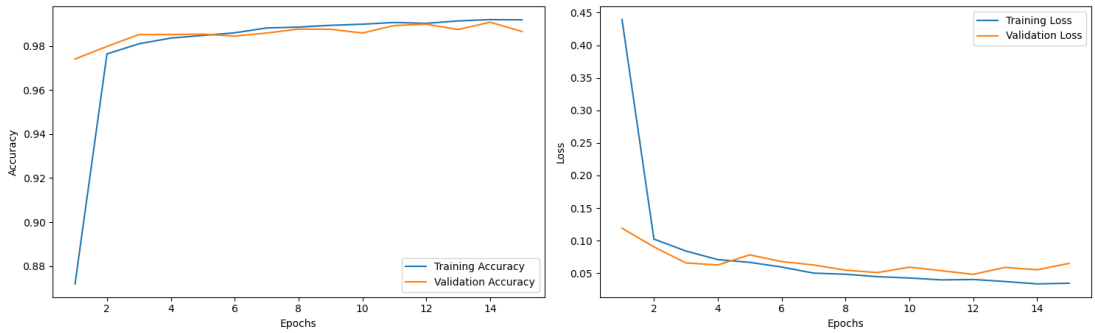


Figure 5.4: Accuracy and loss of InceptionV1

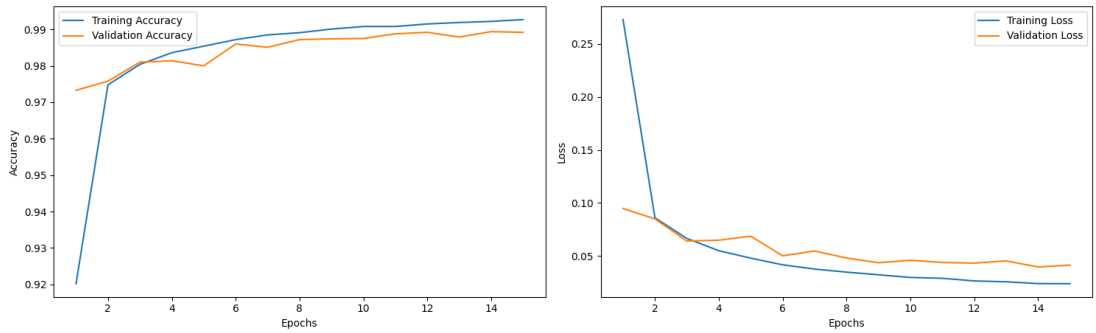


Figure 5.5: Accuracy and Loss of ShuffleNet

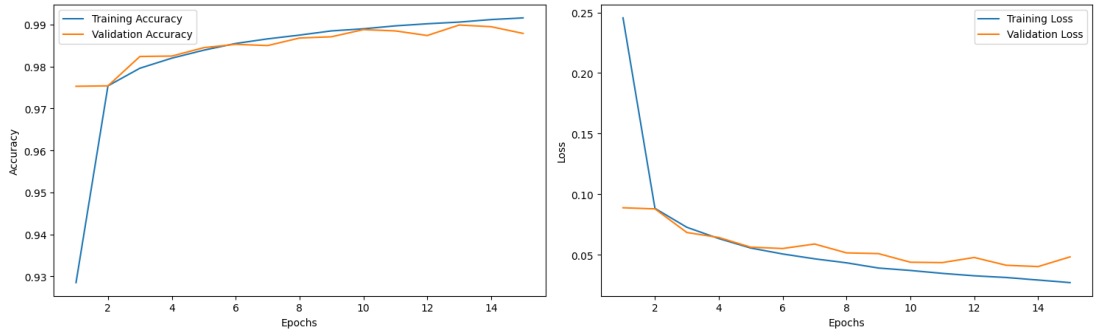


Figure 5.6: Accuracy and Loss of EfficientNet

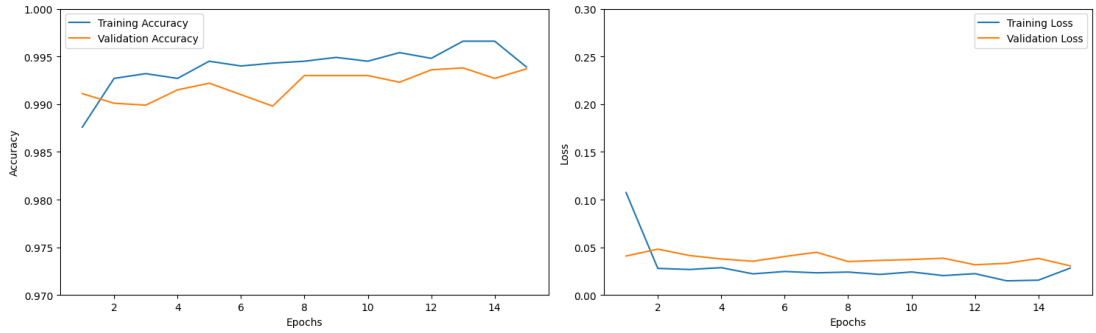


Figure 5.7: Accuracy and Loss of Mobilenet and ShuffleNet

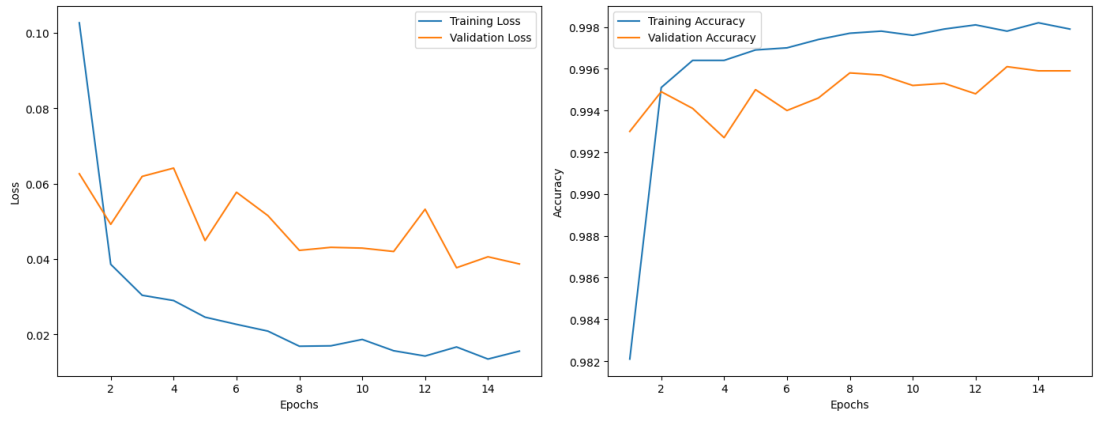


Figure 5.8: Accuracy and Loss of MobileNet, Shufflenet, and Resnet50 hybrid

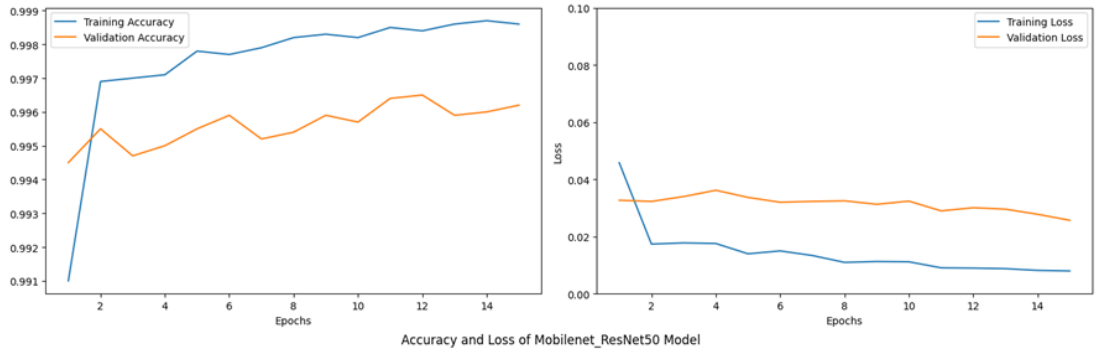


Figure 5.9: Accuracy and Loss of Mobilenet and Resnet50

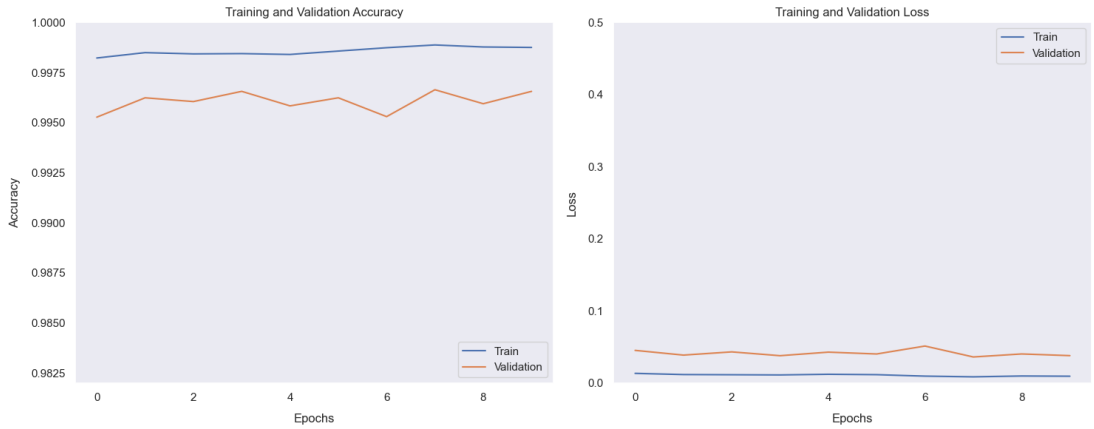
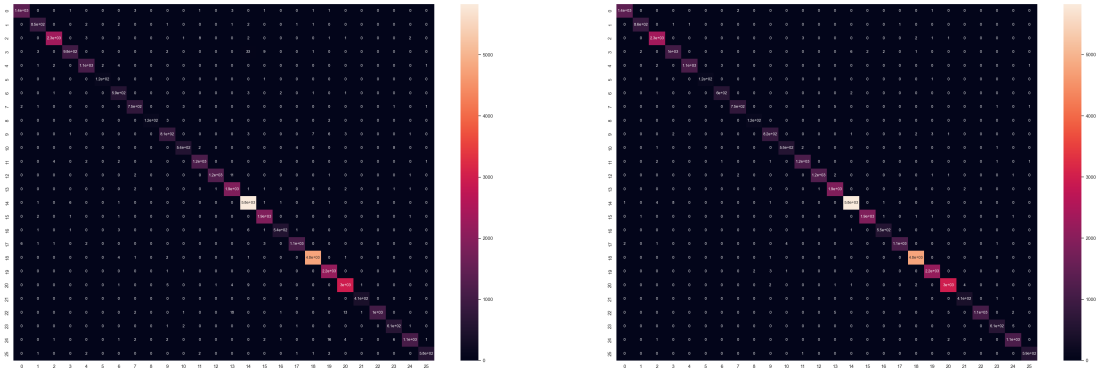


Figure 5.10: Accuracy and Loss of Mobilenet, Efficientnet, and Resnet50 hybrid

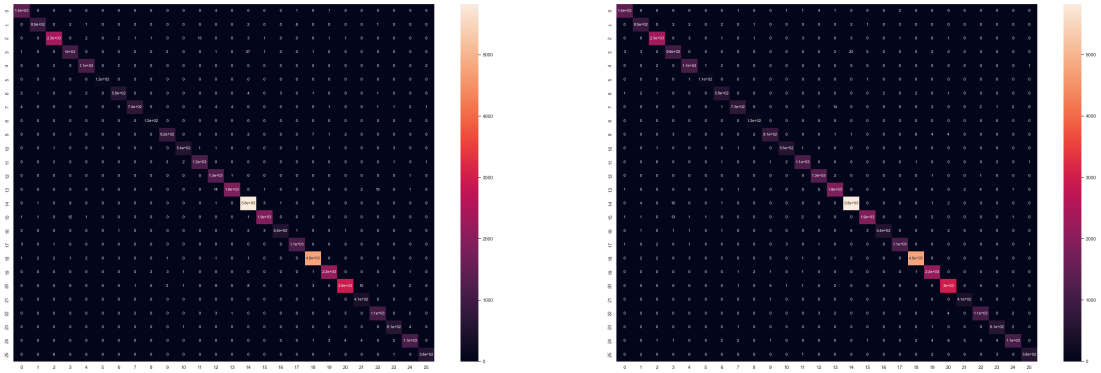
5.4 Confusion Matrix

A confusion matrix is a tabular summary of the number of correct and incorrect predictions made by a classifier. It is used to measure the performance of a classification model. It can be used to evaluate the performance of a classification model through the calculation of performance metrics like accuracy, precision, recall, and F1-score. Experiments are carried out for 26 classes dataset for each model as shown in figures below.



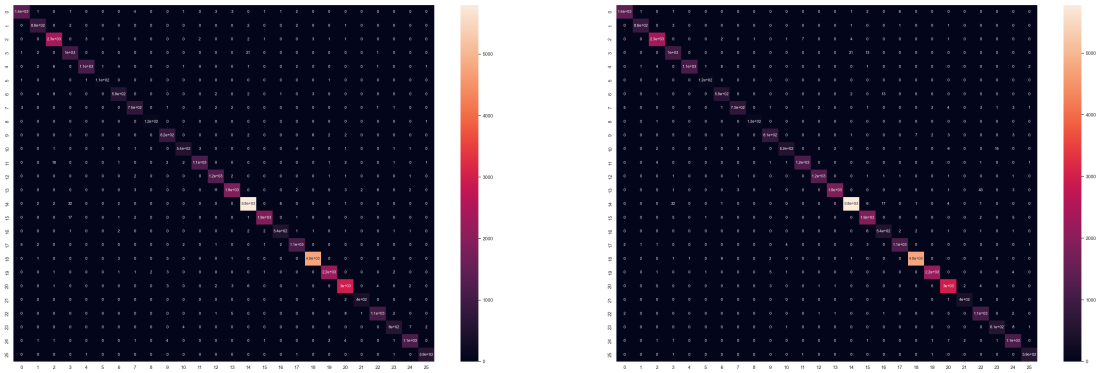
(a) (b)

Figure 5.11: Confusion matrix of (a)Lenet5 and (b)Resnet50



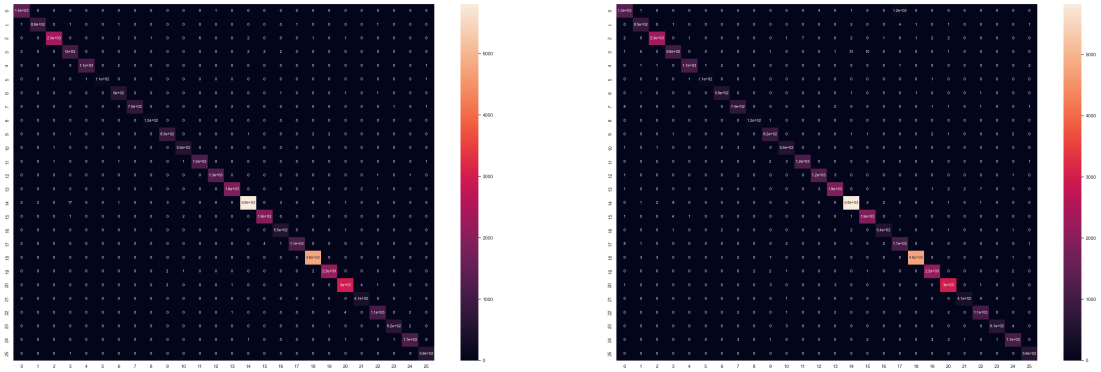
(a) (b)

Figure 5.12: Confusion matrix of (a)Mobilenet and (b)Inception



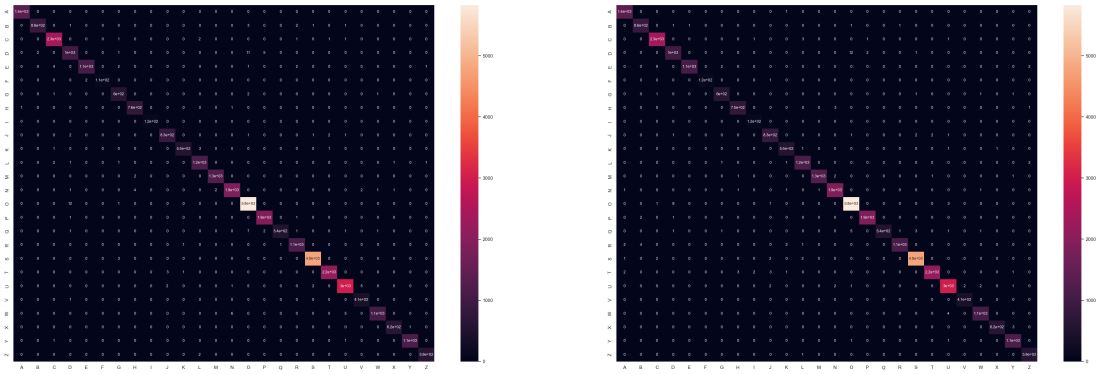
(a) (b)

Figure 5.13: Confusion matrix of (a)Shufflenet and (b)Efficientnet



(a) a (b) b

Figure 5.14: Confusion matrix of (a)Mobilenet and ResNet50 hybrid, (b)Mobilenet and Shufflenet hybrid



(a) a (b) b

Figure 5.15: Confusion matrix of(a)Mobilenet, ResNet50 and Shufflenet hybrid, (b)Mobilenet, Resnet50 and Efficientnet hybrid

5.5 Classification Report of proposed model

To classify the the handwritten alphabets we used 'Adam' optimiser with a learning rate of 0.001. Number of classes is defined in the last fully connected layer, which is the 'fully connected+softmax' layer in each of the proposed models. The following Table 5.7 illustrates the classification of alphabets through the proposed model which gives us highest accuracy i.e. 99.69%.

Table 5.7: Classification Report of the proposed model 4.2.3

Class(A-Z)	Precision	Recall	F1-score	Support
A	1.00	1.00	1.00	1411
B	1.00	0.99	0.99	864
C	1.00	0.99	1.00	2355
D	0.98	0.99	0.99	1030
E	1.00	1.00	1.00	1102
F	1.00	0.97	0.99	116
G	0.99	0.98	0.99	599
H	1.00	0.99	0.99	757
I	1.00	1.00	1.00	120
J	0.99	0.99	0.99	833
K	0.99	0.99	0.99	553
L	0.99	1.00	0.99	1162
M	1.00	1.00	1.00	1256
N	0.99	1.00	1.00	1877
O	1.00	1.00	1.00	5842
P	1.00	0.99	1.00	1892
Q	0.99	0.98	0.98	553
R	1.00	0.99	1.00	1143
S	1.00	1.00	1.00	4776
T	1.00	1.00	1.00	2187
U	0.99	1.00	1.00	2977
V	1.00	0.99	1.00	408
W	1.00	0.99	0.99	1072
X	1.00	1.00	1.00	618
Y	0.99	0.99	0.99	1150
Z	1.00	1.00	1.00	592
Total	1.00	1.00	1.00	37245

5.6 Analysis with existing Records

CNN provides the best accuracy rates for the A-Z handwritten dataset. For this dataset, one of the proposed models (Hybrid of MobileNet, EfficientNet, and ResNet50) is pretty close to 100% accuracy at 99.69%. Comparision among existing results and our prposed model is shown in the Table 5.8 below.

Table 5.8: Comparative analysis of performance among existing works and the proposed model

Authors	**Dataset used**	**Method used**	**Accuracy (%)**
Anita Pal & Dayashankar Singh [11]	A-Z handwritten dataset	Multilayer perceptron with one hidden layer	94.00
Y F Mustafa, F Ridho, S Mariyah[13]	A-Z handwritten dataset	CNN	95.00
Al-Mahmud, Asnuva Tanvin, Sazia Rahman[5]	A-Z handwritten dataset	CNN	98.94
Shengpei Le[6]	A-Z handwritten dataset	CNN	93.00
Nazmus Saqib et al [10].	A-Z handwritten dataset	CNN	99.56
Meheniger Alam [15]	A-Z handwritten dataset	CNN	98.00
Proposed model (Combination of Mobilenet, Efficientnet, and Resnet50)	**A-Z handwritten dataset**	**CNN (Combination of Mobilenet, Efficientnet, and Resnet50)**	**99.69**

6 Conclusion and Future Work

6.1 Conclusion

This paper reports upon work that has conclusively improved the task of handwritten character recognition by creating a complete dataset of handwritten English characters and applying state-of-the-art deep learning models. The total 26 classes of alphabetic characters formed a good dataset used in training and testing different convolutional neural network architectures.

Among all the tested models, the hybrid model consists of MobileNet, EfficientNet, and ResNet50 gained the best performance, with a recognition accuracy of 99.69%. This can be attributed to the fact that the hybrid model combined the architectural advantages for effectively capturing diversified feature representations and enhancing the overall recognition capability. For now, the various architectures are compared under one floor, such as LeNet-5, ResNet-50, MobileNet, Inception, GenNet, ShuffleNet, EfficientNet, and four hybrid models results obtained on this dataset have been shown to give an insight into some of their respective advantages and limitations in respect to applications for handwritten character recognition.

The results obtained in this work show the potential of hybrid models to improve the accuracy and efficiency of OCR systems. However, the improved performance of the envisioned hybrid model in this work will suggest that features combined from different architectures may eventually lead to robust and reliable character recognition systems. This is important for practical implementations:

automated document processing, digital archiving, and assistive technologies.

6.2 Future Work

Building on insights gained from this project, several avenues for future research and development are proposed:

- **Real-Time Implementation**: Developing real-time applications of our predictive models is a crucial next step. This involves optimizing the models for faster processing times and integrating them into real-time systems to enable immediate decision-making and responses.
- **Scalability and Deployment**: Investigating the scalability of our models to handle larger datasets and more complex scenarios is essential. Future research should aim at deploying these models in distributed computing environments to assess their performance and efficiency on a broader scale.
- **Domain-Specific Adaptation**: Adapting our models to specific domains and industries can provide targeted solutions that address unique challenges. Future work should involve customizing the models to meet the needs of different sectors, ensuring their applicability and effectiveness in various contexts.

By focusing on these areas, future research will be well-placed to build on the platform that this report provides, further extending knowledge boundaries and contributing toward progress in predictive modeling. Such aspirations will need to be met with continued dedication to rigorous, innovative, and ethical standards of research.

Bibliography

[1] Réjean Plamondon and Sargur N Srihari. "Online and off-line handwriting recognition: a comprehensive survey". In: *IEEE Transactions on pattern analysis and machine intelligence* 22.1 (2000), pp. 63–84.

[2] Naveen Garg and Karun Verma. "Handwritten Gurumukhi character recognition using neural networks". In: *M. Tech. Theis, Thapar University* (2009).

[3] Atul Negi, Chakravarthy Bhagvati, and B Krishna. "An OCR system for Telugu". In: *Proceedings of Sixth International Conference on Document Analysis and Recognition*. IEEE. 2001, pp. 1110–1114.

[4] Monica Patel and Shital P Thakkar. "Handwritten character recognition in english: a survey". In: *International Journal of Advanced Research in Computer and Communication Engineering* 4.2 (2015), pp. 345–350.

[5] Asnuva Tanvin, Sazia Rahman, et al. "Handwritten English character and digit recognition". In: *2021 International Conference on Electronics, Communications and Information Technology (ICECIT)*. IEEE. 2021, pp. 1–4.

[6] Shengpei Le. "English handwriting recognition based on the convolutional neural network". In: *Applied and Computational Engineering* 5 (June 2023), pp. 146–151. DOI: 10.54254/2755-2721/5/20230550.

[7] P. Latchoumy et al. "Handwriting Recognition using Convolutional Neural Network and Support Vector Machine Algorithms". In: *2022 6th International Conference on Electronics, Communication and Aerospace Technology*. 2022, pp. 1266–1271. DOI: 10.1109/ICECA55336.2022.10009150.

[8] Haining Qiu, Qikun Liu, Eric J Hoffman, et al. "FPGA Implementation of Convolutional Neural Network for Real-Time Handwriting Recognition". In: *arXiv preprint arXiv:2306.13557* (2023).

[9] Priti Singh Rathore and Pawan Kumar. "Handwritten Character Recognition with Neural Network". In: *Ijraset Journal for Research in Applied Science and Engineering Technology* 8 (2022).

[10] Nazmus Saqib et al. "Convolutional-Neural-Network-Based Handwritten Character Recognition: An Approach with Massive Multisource Data". In: *Algorithms* 15.4 (2022). ISSN: 1999-4893. DOI: 10.3390/a15040129. URL: https://www.mdpi.com/1999-4893/15/4/129.

[11] Anita Pal and Dayashankar Singh. "Handwritten English character recognition using neural network". In: *International Journal of Computer Science & Communication* 1.2 (2010), pp. 141–144.

[12] PM Bhagyashree, LK Likhitha, and DS Rajesh. *Handwritten Digit Recognition Using Deep Learning*. 2021.

[13] Yusron Farid Mustafa, Farid Ridho, and Siti Mariyah. "Study of Handwriting Recognition Implementation in Data Entry of Survei Angkatan Kerja Nasional (SAKERNAS) using CNN". In: *Proceedings of The International Conference on Data Science and Official Statistics*. Vol. 2021. 1. 2021, pp. 53–65.

[14] Naiwrita Borah et al. "Efficient Assamese Word Recognition for Societal Empowerment: A Comparative Feature-Based Analysis." In: *IEEE Access* (2023).

[15] Meheniger Alam. "Hand Writing Recognition (English & Digit)". PhD thesis. Comilla University, 2023.

[16] Charu C Aggarwal et al. *Neural networks and deep learning*. Vol. 10. 978. Springer, 2018.

[17] https://www.scienceabc.com/wp-content/uploads/2018/05/Structure-of-a-typical-neuron-768x478.jpg.

[18] https://www.geeksforgeeks.org/introduction-deep-learning/.

[19] https://datasciencedojo.com/wp-content/uploads/ml-ds-algos.jpg.

[20] https://editor.analyticsvidhya.com/uploads/90650dnn2.jpeg.

[21] https://www.researchgate.net/publication/356841798/figure/fig2/AS:1098742765764615@1638972097475/An-example-of-the-convolution-operation.png.

[22] Anirudha Ghosh et al. "Fundamental Concepts of Convolutional Neural Network". In: Jan. 2020, pp. 519–567. ISBN: 978-3-030-32643-2. DOI: 10.1007/978-3-030-32644-9_36.

[23] https://www.geeksforgeeks.org/introduction-convolution-neural-network/.

[24] https://botpenguin.com/glossary/softmax-function.

[25] Jason Brownlee. "A gentle introduction to cross-entropy for machine learning". In: *Machine learning mastery* 20 (2019).

[26] https://en.wikipedia.org/wiki/Cross-entropy.

[27] https://en.wikipedia.org/wiki/Stochastic_gradient_descent.

[28] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[29] https://medium.com/@nitishkundu1993/exploring-resnet50-an-in-depth-look-at-the-model-architecture-and-code-implementation-d8d8fa67e46f.

[30] https://www.researchgate.net/figure/The-architecture-of-MobileNet-V1-30_fig5_357623745.

[31] https://iq.opengenus.org/content/images/2022/11/Architecture-of-EfficientNet-B0-with-MBConv-as-Basic-building-blocks.png.

[32] Christian Szegedy et al. "Rethinking the inception architecture for computer vision". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2818–2826.

[33] https://machinelearningmastery.com/review-of-architectural-innovations-for-convolutional-neural-networks-for-image-classification/.

[34] Xiangyu Zhang et al. "Shufflenet: An extremely efficient convolutional neural network for mobile devices". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 6848–6856.

[35] https://medium.com/syncedreview/shufflenet-an-extremely-efficient-convolutional-neural-network-for-mobile-devices-72c6f5b01651.

[36] https://www.kaggle.com/datasets/sachinpatel21/az-handwritten-alphabets-in-csv-format.