Optimizing American Sign Language Recognition with Binarized Neural Networks: A Comparative Study with Traditional Models

by

Shakeef Ahmed Rakin
20301046
Mohammed Intishar Rahman
20301191
Md.Tahjid Ahsan
20301209
Afif Alamgir
20301199
Md Sifat Mahmud
20101477

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

Department of Computer Science and Engineering
Brac University
May 2023

Declaration

It is hereby declared that

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

Student's Full Name & Signature:

Shakeef Ahmed Rakin
20301046

Tahzid

Mohammed Intishar Rahman
20301191

Intishar

Md.Tahjid Ahsan
20301209

Afif Alamgir
20301199

Sifat

Md Sifat Mahmud
20101477

Approval

The thesis titled “Optimizing American Sign Language Recognition with Binarized Neural Networks: A Comparative Study with Traditional Models” submitted by

1. Shakeef Ahmed Rakin(20301046)
2. Mohammed Intishar Rahman(20301191)
3. Md.Tahjid Ahsan(20301209)
4. Afif Alamgir(20301199)
5. Md Sifat Mahmud(20101477)

Of Spring, 2023 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on May 25, 2023.

Examining Committee:

Supervisor:
(Member)

Md Tanzim Reza
Lecturer
Department of Computer Science and Engineering
Brac University

Co-Supervisor:
(Member)

A.M. Esfar-E-Alam
Senior Lecturer
Department of Computer Science and Engineering
Brac University

Program Coordinator:
(Member)

Md. Golam Rabiul Alam, PhD
Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

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

Abstract

Sign language is essential for effective communication among individuals with hearing or speech impairments. Automated recognition systems for sign language are crucial for facilitating learning and translation across different sign language variants. However, existing systems often struggle with high computational demands and large memory footprints, limiting their applicability in real-time and resource-constrained environments. This research aims to develop an optimized pipeline for American Sign Language (ASL) recognition, focusing on the comparison between Binarized Neural Networks (BNNs) and traditional full-precision neural networks. Using Larq, an open-source Python library for training binarized models, we exploit BNNs' advantages of reduced memory and computational needs, making them ideal for embedded systems and edge devices. The study utilizes a dataset of ASL alphabet images, encompassing a variety of hand configurations and movements. Data augmentation techniques are applied to address challenges like data imbalance and occlusions. Both binarized and traditional models are trained and optimized, and their performance is evaluated using metrics such as accuracy, precision, recall, F1-score, memory footprints and inference times. The results indicate that BNNs offer competitive performance with significantly lower computational requirements, paving the way for more efficient and accessible ASL recognition systems and demonstrating the potential of binarized models in this domain.

Keywords: Artificial Intelligence (AI), Sign Language, Deep Learning, American Sign Language, CNNs, Larq, BNNs, DenseNet121, ResNet50, VGG16, BinaryDenseNet45, BinaryDenseNet37, BinaryResNetE18

Table of Contents

Chapter 1

Introduction

Today's world is interconnected through communication, which plays an essential role in promoting understanding and engagement among individuals. Conventional spoken language can present challenges for the deaf and hard of hearing communities. Sign language, with its gestural and visual aspects, helps to bridge the communication gap between these individuals and others. Understanding the value of inclusive communication, researchers and scientists have explored creative ways to help sign language users connect with the wider population. One promising approach combines deep learning methods with advanced model architectures to improve sign language recognition. Our research aims to study and analyze the performance of Binarized Neural Networks compared to other traditional neural network models for the intent of American Sign Language (ASL) recognition. Some key advantages of BNNs are reduced memory and computation requirements, making them ideally applied to embedded systems for real-time edge applications. Our aim is at creating a highly performing, effective ASL recognition system that recognizes and understands hand movement with good accuracy. To achieve this, we preprocess the ASL dataset by using data augmentation techniques to handle obstacles like imbalanced data, huge intra-class variability in hand positions, and occlusions. With this, there will be a guarantee of enough adaptability of the models to a significant variety of different conditions. Our study is going to involve the training and optimization of both Binarized Neural Network (BNN) models and conventional ones in sequence. We will then compare these two according to various performance measures, including: accuracy, precision, recall, and F1-score. So, this research is going to provide sufficient insight into the strengths and limitations of BNN models over traditional models. Developing more efficiency and cost-effective ASL identification systems that are well suited to real-time applications.

1.1 Background

Sign language is visual language that converts body language into words. Currently in the world, there are over 72 million deaf people according to the World Federation of the Deaf [20]. Furthermore, there are over 300 sign languages around the world with 57.6 million people using them in developing countries. Not all people have the ability to interpret sign language which can cause deaf people a lot of struggles in functioning properly in society. Moreover, sign language can range from simple gestures to entire sequence of gestures which interpret into sentences which allows people to bridge the gap between deaf people and society. Consequently, this makes sign language a steep hill to climb for normal people to learn. Not only that but also, out of the 500,000 deaf people in the US, only 1% use sign language [9]. The importance of the integration of sign language into society can improve of the inclusivity of people with hearing loss into everyday societal functions. Furthermore, with the advent of AI, many machine learning based sign language recognition tools have become available but not yet to the point that they are usable in everyday life due to various limitations and drawbacks. They deploy various techniques such as classification algorithms, feature extraction, data augmentation and deep learning architectures. Some techniques suffer from drawbacks due to not being able to detect hand gestures in unsatisfactory lighting conditions or backgrounds, different skin tones, low quality visual inputs etc. which are all crucial points in enhancing the overall usability and adaptability.

1.2 Research Objective

Our objective is to develop a pipeline for American Sign Language (ASL) recognition with a primary focus on comparing Binarized Neural Networks (BNN) with traditional models. Traditional sign language recognition systems encounter challenges that can be addressed by optimizing deep learning methodologies. For optimization, we will employ Larq, an open-source Python library designed for training binarized models. We will investigate the suitability and analyze sign language gestures, considering the unique advantages of BNN models, such as reduced memory and computational needs. The objective is to effectively discern the unique patterns exhibited by hand movements and shapes that are linked to ASL audibles by means of deep learning methodologies. Additionally, the method tries to reduce the large amount of memory and processing power consumed by regular full-precision neural networks by orders of magnitude, while binarized models are naturally resource-efficient and suited to be enabled in real-time applications on embedded systems and edge devices.

This report seeks to train and test deep learning models that feature a collection of pictures of American Sign Language alphabets. Sufficiency of pictures that show many different hand positions, movements, and signs is critical to the full expression of ASL. To improve the model's ability to generalize, we intend to preprocess the ASL dataset using data augmentation techniques. This will involve resolving issues such as data imbalance, hand position variability, and occlusions. Subsequently, on the ASL dataset, we shall train and optimize deep learning models, including both binarized models and traditional models, to attain optimal recognition accuracy. We will analyze the performance of the developed ASL recognition system using metrics like accuracy, precision, recall, and F1-score, comparing it to existing systems and benchmarks. Eventually, the findings of this study will contribute to the development of more efficient and economical ASL recognition systems.

1.3 Problem Statement

In the field of recognizing sign language using deep learning, traditional models face significant challenges, especially for less complex processes where the nature of those models leads to high computational power and large memory footprints.

- Traditional neural networks use higher bit weights and activations, leading to slower inference times and higher energy usage. This becomes particularly problematic in scenarios requiring real-time processing and immediate feedback, such as sign language recognition. The computational complexity and memory demands of these traditional models make them less suitable for deployment on resource-constrained devices like mobile and embedded systems.
- The large memory footprints of traditional neural networks further limit their practicality for real-time applications. These models often require substantial hardware resources to function effectively, which can be a barrier to widespread adoption, particularly in portable or low-power devices.

1.4 Research Orientation

In the upcoming chapters, we will discuss about some specific topics essential for a comprehensive understanding of our research. These key topics include:

Literature Review

This segment serves as a foundation by establishing the existing knowledge and research on our own research topic. The literature review aids in framing our research inquiries and hypotheses, providing the conceptual framework for our study.

Methodology

The methodology section explains the complexities of our research procedure and allows the readers to look into the reliability and validity of our study's conclusion. This chapter contains various subsections:

- Dataset Used
- Data Preprocessing
- Dataset Distribribitution
- Models Used
- Experimental Setup

Result Analysis

This step includes the compilation and analysis of our findings.We analyze the findings in accordance with the research questions and hypotheses stated in our study. This chapter contains the folowing subsections:

- Performance Measures
- Learning Curves
- Confusion Matrices
- Classification Reports
- Score Analysis
- Model Size, Parameters, and Accuracy Comparison
- Inference Time Analysis
- Advantages of Binarized Models

Future Works

This section serves as a bridge between the current study and prospective possibilities for further exploration and development.

Conclusion

Bringing our paper's results and analyses to a conclusion, this chapter's focus is only on summarizing our findings and results.

Chapter 2

Literature Review

Hein et al [15] proposed a leap motion based model for sign language recognition which is divided into two sections namely the training section and the classification section. Firstly, a RGB color spaced video is used as an input which is further enhanced and transformed into YCbCr. Since it is motion based, the video's component which is the head and the hand is detected and localized after which, the skin component is also trained via machine learning. In the latter section, the sequence of the sign is taken through a leap motion sign recognition and finally, the output is shown. Their proposed system 3 different classes and 38 different hand movement styles which are appropriately labelled. They had access to hand data based on the Myanmar Sign Language Conversations Book and DVD. They created their dataset proposed model by having 35 different signers each performing each sign ten times. The resulting dataset was separated into three groups which had an accuracy of 94.355%, 92.32% and 85.79%. During their testing, they encountered problems regarding detecting unconstrained faces. Furthermore, they will be developing their system by applying YOLO CNN and by having a recognition process based on deep learning.

In the paper of Raval and Gajjar [17], their proposed model consisted of a convolutional neural network with two different sections which are image processing and machine learning. In the first part, the images of the hand are captured and processes by applying a mask on the hand portion of the image which is then normalized by 255 and converted into a row matrix to be saved to the database. Later, it is passed onto the next part which is the CNN algorithm. In the last part, the different features of the hand are identified and classified using CNN where Keras library is used. Moreover, 15% of the dataset is testing and the rest for training which consists of signs of 240 images of 10 images for each alphabet. After the testing, the results displayed an accuracy of 83.79%. Their testing was purposefully done in different background and lighting conditions to test the limits of their model. According to them, their paper can be further developed by converting into a continuous sign language recognition model where successive frames of a sequence of hand signs forming sentences can be detected. Moreover, having a dataset consisting of variety of skin tones and lighting conditions can develop the robustness of their proposed model even more.

The model by Halder and Tayade [14] consisted of a simplified sign language recognition system using MediaPipe's open-source framework alongside machine learning algorithm. MediaPipe is a framework which has support for detecting and tracking human body models which are pre-trained with diverse datasets of Google. Moreover, their model good detect in real-time using SVM. Their proposed model consists of 3 stages where, in the first stage, the images are preprocessed where the multi-hand locations are detected and coordinate points are three-dimension normalized. MediaPipe is then responsible in generating a processed data consisting of different numbered and labelled sections of each hand. During the second stage, null entries are cleaned which can lead to blurry images after which, the data is prepared for splitting where 80% is used for training and the rest for testing. In the final stage, by using machine learning algorithms and SVM, the datasets are tested. In the final results, SVM outperformed all the other algorithms as it was more efficient and better in high-dimensional spaces. Overall, the average accuracy of the proposed model was 99% in most of the sign language datasets which were very diverse. Their proposed model using MediaPipe not only proves to be very robust and cost-effective but also, brings into light, the problem of requiring high computational power in image preprocessing of hand signs. According to the authors, their work can be developed further by introducing word detection from video files using MediaPipe.

Harini et al [7] proposed a model of sign language translation containing four modules namely image capturing, pre-processing, classification and prediction. In their model, the image-capturing module is taken care of by using the OpenCV library and the computer's internal camera. One problem faced by their model was that the image must be captured properly. Around 3000 images for each sign are collected and saved to a csv file with appropriate pixel values for improved accuracy. The next module, pre-processing, focuses on the background condition where the images are gray-scaled and taken through a background subtraction algorithm. This helps in detecting the hand in dynamic background conditions. The classification phase used the CNN algorithm. Finally, the prediction layer displays the accuracy of their proposed model which was 99.91%. According to the author, facial expressions play a vital role in sign language and their model is not suited for said task. Furthermore, the accuracy of their model dipped in low lighting. Future improvements include more dynamic video signs involving facial recognition and expressions.

In this research [10] article Systems for recognizing sign language are being developed to overcome hurdles to communication for the deaf. Traditional computer vision techniques have drawbacks and are expensive. This work suggests a wearable ASL interpretation system based on sensor fusion and deep learning. For dynamic ASL gestures, IMUs affixed to the fingertips and back of the hand obtain a 99.81% recognition rate. The major non-verbal communication method is ASL, which uses hand gestures, body forms, and facial emotions. Due to ASL's language and organization, learning it can be difficult, which makes hearing-impaired people feel alone. The suggested system raises accessibility and standard of living. IMU sensors and a tuned RNN with LSTM are used to assess finger and hand movements for precise classification. Future work will involve healthcare and additional sign languages.

In this review of the research [6], computer vision-based and wearable device-based methods are highlighted as they relate to gesture and activity recognition systems. Gesture recognition powered by Wi-Fi signals has become a viable device-free option, with benefits including greater coverage and the capacity to see items behind walls. Deep learning models are used by Wi-Fi-based recognition systems to examine Channel State Information (CSI) features and achieve high accuracy. The proposed deep learning architecture addresses shortcomings of existing systems in complicated situations with numerous users by leveraging Wi-Fi CSI, CNN, LSTM, and ABLSTM for sign word recognition. The framework employs noise removal and offset correction algorithms to achieve excellent recognition accuracy. The suggested deep learning framework performs better than others in a range of settings, including single-user and multi-user scenarios. Future enhancements to Wi-Fi-based gesture recognition systems may be possible with further development of transfer learning methodologies like ResNet.

This paper [18] introduces H-DNA, a hybrid deep neural architecture for video production, translation, and sign language recognition. To provide results of a high caliber, it makes use of the CNN, LSTM, GRU, and GAN models. The H-DNA system seeks to advance conventional methods by offering real-time translation and recognition through an application with a user interface. It is tested through tests and by humans after being trained on sizable datasets.The first unified deep learning method for translating and recognizing sign language is H-DNA. To produce high-resolution sign films, it combines dynamic GAN, MediaPipe library, and neural machine translation. The framework performs better than earlier methods and achieves great accuracy. Real-time multimodal and multilingual sign gesture identification is made possible, bridging the communication gap between groups with normal abilities and those with impairments.The paper discusses issues including output blurring, self-occlusion, movement epenthesis, ambiguity, noise, and incorrect classification. Using H-DNA is an effective measure in solving the problems mentioned above and is quite reliable and potent. The system effectiveness has been proved by the results of experiments using different sets of signs, and the performance measure shows that this system is efficient. In conclusion, a new and beneficial solution of the H-DNA structure for video generation, translation, and sign language recognition is presented. It overcomes the drawbacks of the existing approaches and also has the potential to significantly improve engagement and interaction among users with hearing or speech challenges.

This research [19] highlights the role played by sign language as a medium of connection for people with speech and hearing impairments. The main focus of the research project revolves around the need to seal the communication gap and offer fair chances to these individuals. This paper looks into video-based systems to record and track hand movements, specifically for sign language identification and understanding. For the 11 often-used words, the developed dataset is made, and the prime focus is on the Indian Sign Language. Unlike other datasets, a more practical and natural approach was taken because no external sensors or smart gloves were utilized to track hand movements. For real-time sign language detection and recognition, the suggested system integrates LSTM and GRU models using a vision-based methodology. To improve ISL recognition accuracy, the authors experiment with different LSTM and GRU layer combinations. The suggested model performs better than the current methods for ISL recognition, especially for terms that are used frequently, according to experimental results. The paper makes recommendations for future developments, including the creation of new datasets, adjustments to camera angle, and the incorporation of wearable technology.

According to the authors of this paper [4], the survey of research emphasizes the value of people detection in a range of applications, including surveillance, counting people, human-computer interaction, and people tracking, particularly in outdoor settings of smart city environments.The great variety of factors like temperature, light, and backdrop, as well as the need for dispersed sensing and intelligence across big spaces, make it particularly difficult to detect people outside.This research focuses on resource-constrained, low-cost IoT end devices that can identify persons outside while maintaining privacy. The method makes use of thermopile arrays, a cutting-edge sensor technology, and low-resolution thermal cameras, notably the Grid-EYE device.The research examines the distinctions in classification accuracy between the weight representations used by the CMSIS-NN library, which are 8-bit fixed-point and 32-bit floating-point, respectively. It is described how to use CMSIS-NN to transfer the trained classifier from TensorFlow to the micro-controller.To illustrate the idea, a prototype consisting of a thermopile-array-based sensor and an evaluation board is constructed. The micro-controller runs a 3-layer CNN successfully, enabling real-time thermal image categorization at 10 Hz with little power usage.The results demonstrate that the micro-controller-based implementation for persons detection utilizing low-resolution thermal pictures provides efficient processing and low power consumption. Low precision (8-bit fixed-point) format is used in the system with minimum performance impact, making it appropriate for deployment in memory-constrained devices with limited resources.

The growing population of people with hearing or listening impairments—466 million as of early 2018 and expected to rise to 400 million by 2050—is discussed in this analysis . The deaf community uses sign language, a nonverbal communication system with several sign languages developed for various countries. For the purpose of enhancing communication between hearing-impaired and hearing-normal people, trained sign language interpreters are crucial. America, Canada, and other nations all make extensive use of American Sign Language (ASL). Convolutional neural networks (CNNs) have significantly improved image identification compared to shallow neural networks, which were previously used to study automatic ASL recognition. The suggested CNN model from this paper [5] (SLRNet-8) outperforms existing techniques and recognizes ASL signs with high accuracy. The study made use of openly accessible ASL datasets and assessed the model's performance across several datasets. The model may be used in the future to recognize continuous words at the sentence level in ASL or to recognize ASL from video data.Overall, the study shows how well the suggested CNN model performs automatic ASL detection and its potential to improve hearing-impaired communication interface.

In this paper [12], Numerous sectors, including automobile user interfaces, health and medical systems, crisis management, entertainment, and human-computer/robot interaction, have found extensive use for sign language based on hand gestures. Researchers have created a variety of deep learning hand gesture recognition models to aid in efficient communication in these contexts. These models use convolutional neural networks (CNN), 2D CNN, 3D CNN, and recurrent neural networks (RNN) to recognize static and moving hand gestures. These models have been demonstrated to have high classification accuracies ranging from 85.46% to 98.81%, making them useful for correctly identifying hand gestures. The effectiveness of these models has been enhanced using a variety of techniques, such as data augmentation, morphological filters, Gaussian mixture models, and shape and texture evidence. The developments in this area could revolutionize contactless communication in the fields of entertainment, healthcare, and other industries by enabling efficient interaction and control.

As a crucial means of communication for the deaf and hard of hearing, sign language recognition has received a lot of attention in the field of computer vision. Similarities in gesture meaning, uncontrolled surroundings, complicated backgrounds, and lighting circumstances all contribute to the difficulty of sign language recognition. For effective feature extraction and classification on large sign language databases. In this paper [8], deep learning models, in particular convolutional neural networks (CNNs), have been intensively researched. Promising results have been achieved in real-time sign language recognition through the use of techniques such as grid-based fragmentation for recognizing hand poses, Haar cascade classifiers for detecting hand regions, Histogram of Oriented Gradients (HOG) feature extraction, and vision-based feature fusion. However, during training, CNN-based techniques require a lot of resources, making them inappropriate for embedded platforms with little hardware. A solution given to this problem is the use of 1-bit quantized binary weights and activations in binary neural networks (BNNs) for real-time recognition of two-hand motions against complex backdrops. The BNN architecture has been proven superior in experimental assessments on the Two hands Indian Sign Language (ISL) database, where it has achieved an overall accuracy of 98.8 percent while also exceeding conventional CNNs in terms of training speed and memory needs. Future research will focus on implementing the real-time system on software-programmable FPGAs, as well as addressing missclassifications and expanding the BNN architecture to enable a greater number of gesture classes.

A well-known dynamic gesture recognition system in the entertainment industry is the Nintendo Wii, which requires high-resolution cameras to function in real-time. Higher image quality, however, necessitates more computational resources and expense, necessitating rigorous field testing to identify the most advantageous cost-effective approach. When employing thermal cameras with infrared (IR) lighting, image recognition systems can overcome the difficulties brought on by the lighting and background fluctuations in RGB images. Thermal cameras produce streamlined, less-detailed images while operating in complete darkness. In order to perform hand gesture identification using thermal pictures, this work [13] presents two datasets: one with clean backdrops for accurate classification and the other with background noise and objects for added complexity and texture. In order to classify gestures, a Convolutional Neural Network (CNN) model is suggested. The datasets were collected using a thermal camera module. For offline training or online upgrades with new gestures, the CNN model is intended to be extendable. The suggested model outperforms benchmark models in terms of accuracy and inference times. Future study will examine ways to process live stream data more quickly using CNN models like MobileNet V3, add more features, collect data from more people, and investigate whether good recognition could need object detection.

The authors of this paper [22] present a technique for simultaneously predicting hand motions, handedness (left or right hand), and hand keypoints (fingertip and wrist points) using infrared imaging. With three task-specific branches and shared encoder-decoder layers, they offer a deep multi-task learning architecture. The model is developed and validated using a proprietary dataset composed of thermal imaging data from 24 people doing 10 hand gestures. The results show good accuracy for localizing wrist points and high accuracy for categorizing gestures, identifying handedness, and localizing digits. The recommended approach for providing accurate hand gesture detection combines the benefits of deep learning with infrared imaging.

A real-time word-level sign language recognition system for Indian Sign Language (ISL) is developed in this study using deep neural networks. The goal is to improve communication between deaf or mute persons and hearing people. This study [21] covers challenges of ISL recognition, such as finger angle fluctuations and sign similarities. The proposed method employs a CNNs trained on an ISL dataset to recognize hand gestures captured by a camera. The dataset is preprocessed using techniques including grayscale conversion, image segmentation, edge detection, and thresholding. With 99% accuracy, CNN recognizes the images and extracts features. The paper reviews previous research in the field and contrasts various models. The recommended fix focuses on character-level recognition and provides a practical deep neural network-based technique for ISL recognition.

The dataset of this particular paper [16] is pre-processed using methods including edge detection, threesholding, picture segmentation, and gray scale conversion. CNN identifies the photos and extracts features with 99% accuracy. The study compares several models and discusses earlier research in the area. The suggested remedy emphasizes character-level recognition and offers an efficient deep neural network-based method for ISL recognition.A system that translates sign language into spoken language using cutting-edge techniques like transformers and graph neural networks is covered in the literature that is supplied. The process involves eliminating the human skeletons from the movies, encoding them using transformers and graph convolution, pooling the encoded graphs, translating the language, and finally improving the model by adding different losses. The technology performs a fantastic job of recognizing sign language and translating it into spoken English.

RGB image processing, while standard, is limited by computational and memory constraints. Conventional deep learning models, though effective, often struggle with real-time performance on resource-constrained devices. Binarized Neural Networks (BNNs) offer a promising solution, providing computational efficiency and reduced memory usage. This makes them suitable for real-time applications on limited hardware. However, BNNs may be sensitive to noise and complex data, necessitating further research to improve binarization techniques and maintain accuracy.
The study aims to explore the application of Binarized Neural Networks (BNNs) for RGB-based sign language recognition, aiming to overcome traditional deep learning's limitations. By leveraging advancements in BNNs, the research seeks to develop efficient systems that operate effectively on standard RGB inputs. This approach addresses memory management and processing speed challenges, contributing to robust and cost-effective solutions for improved communication within the deaf and hard-of-hearing communities. The focus on RGB image processing underscores the potential to enhance sign language recognition on widely accessible devices, promoting practical applications in real-world settings.

Chapter 3

Methodology

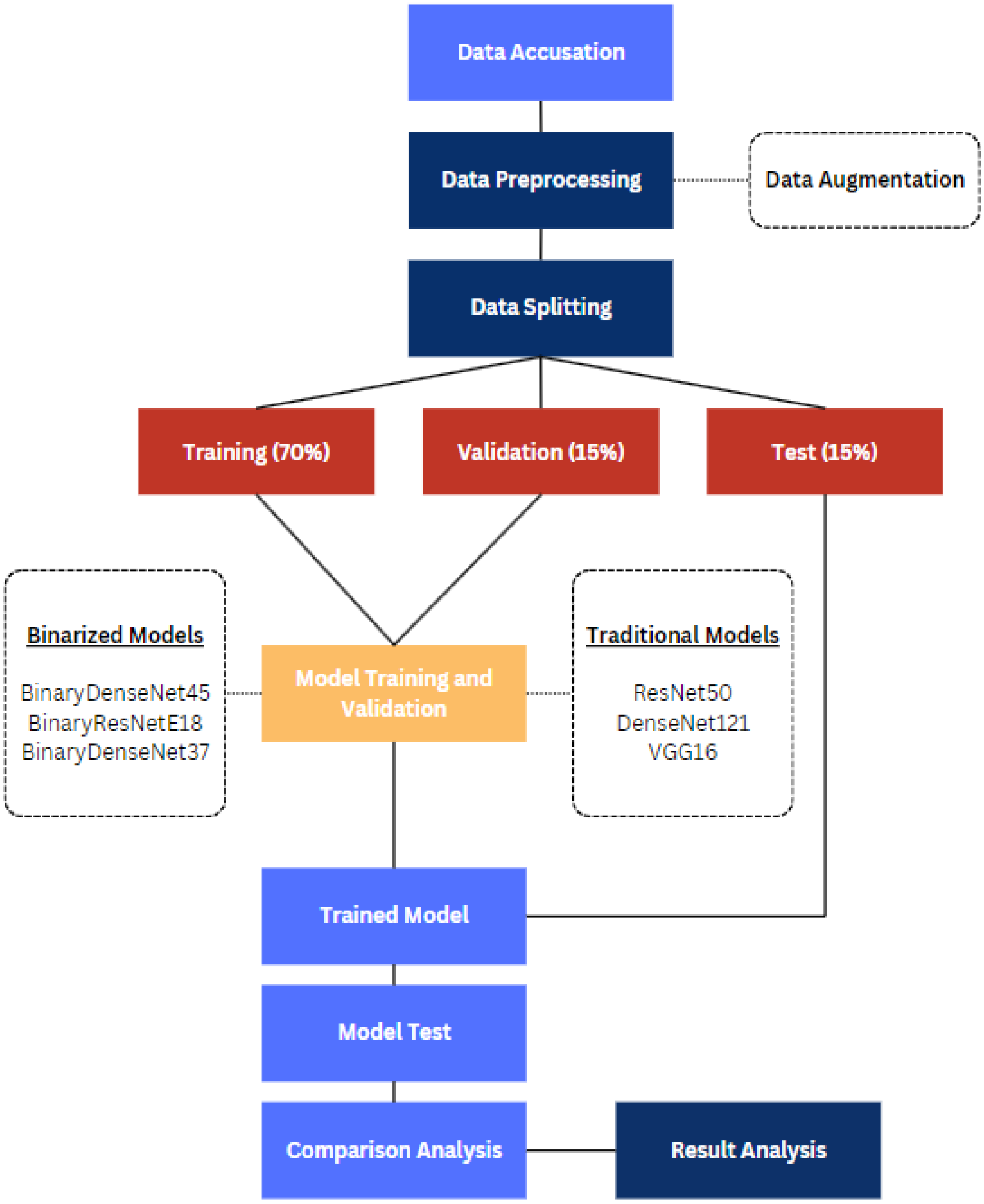


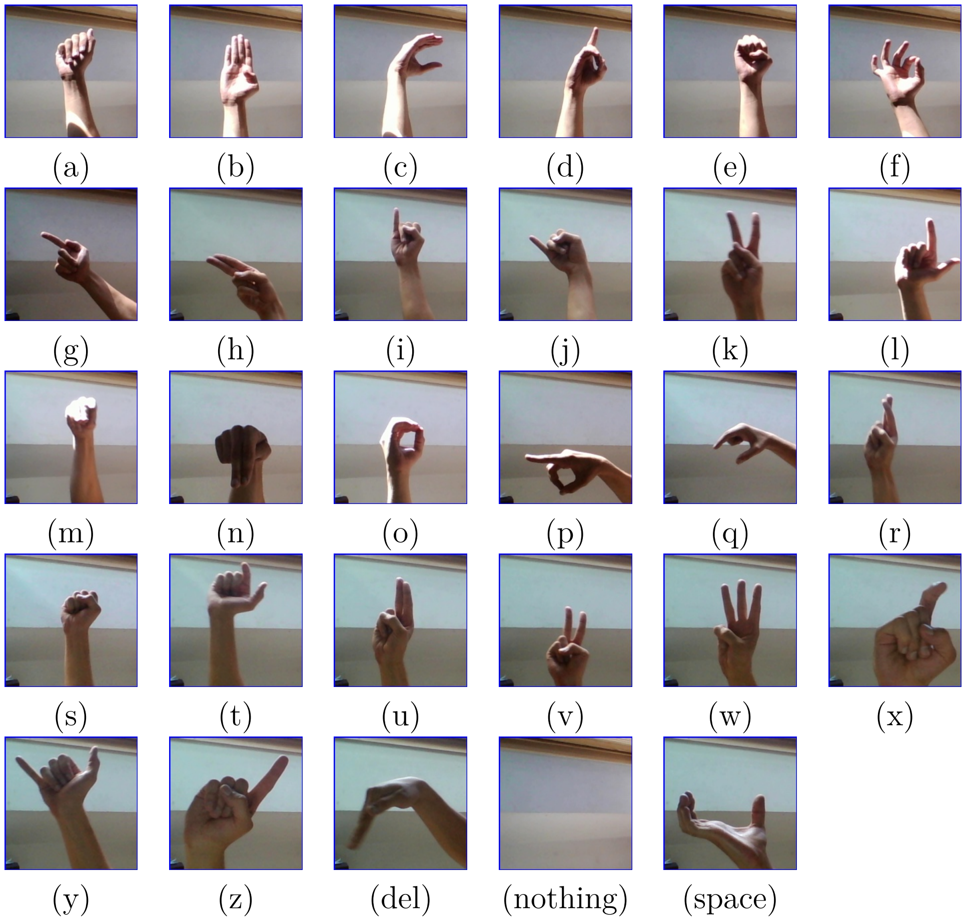
Figure 3.1: Research Methodology

3.1 Dataset Used

We have collected the Image dataset for alphabets in the American Sign Language dataset from Kaggle submitted by Akash Nagaraj [24]. These images are an essential component of our study and have been pivotal in training and evaluating the performance of our selected models on large datasets.

The dataset we have used is a compilation of 87,000 images of the alphabets according to the American Sign Language which are segmented into 29 folders to represent the classes. Each image measures a resolution of 200×200 pixels. Out of the 29 classes, the letters A-Z are separated into 26 classes and the remaining 3 classes include SPACE, DELETE and NOTHING. In real-time application and classification, these 3 classes are very necessary and helpful.

- Folder ”A” contained 3000 RGB images representing letter A.
- Folder ”B” contained 3000 RGB images representing the letter B.
- The other directories contain the rest of the 3000 images per class, totalling 87000 images.



(a) (b) (c) (d) (e) (f)

(g) (h) (i) (j) (k) (l)

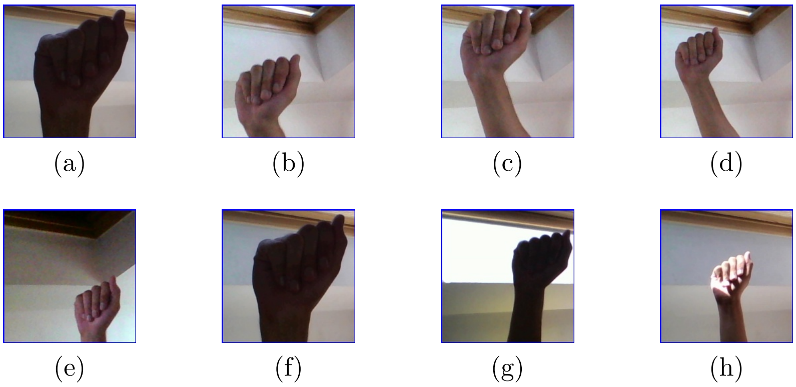
(m) (n) (o) (p) (q) (r)

(s) (t) (u) (v) (w) (x)

(y) (z) (del) (nothing) (space)

Figure 3.2: RGB images corresponding to: (a) Alphabet A .. (space) Space

The large size of this dataset—with 87,000 images of the American Sign Language alphabet broadly spans class representation and is important not only for the lighting condition variation and hand postures but also for the wide-range possibilities that closely resemble the various situations in which ASL can be applied in the real world. The wide range of circumstances in lighting captures dimming and brightening, shading, source refraction, and overall illumination in the process affecting the appearance of the signs in different environments. At the same time, variability in hand positioning in sign language motions captures the originality and thus accounts for a wide range of hand shapes, orientation, and movements that can in turn influence the visual appearance of the signs.


(a) (b) (c) (d)

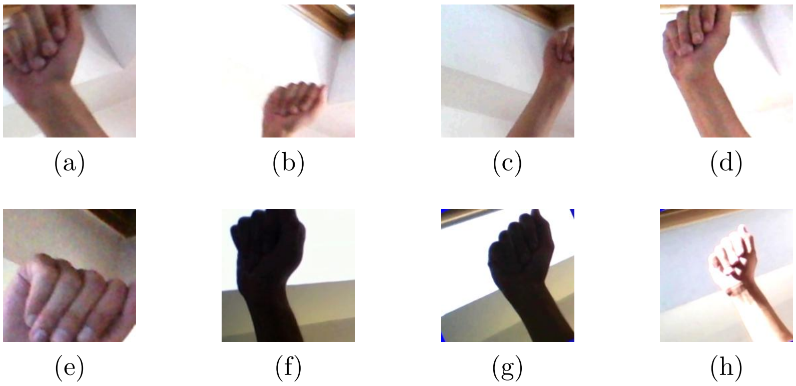
(e) (f) (g) (h)

Figure 3.3: Different RGB images of the alphabet A

3.2 Pre-Processing the Dataset

3.2.1 In-Place Data Augmentation

For the purpose of the present study, we applied data augmentation techniques so as to bring more variability into the current dataset for the purpose of our models' training. This strategy included random rotation by a maximum of 30°, alteration of magnification level within the range of 0.6 to 1, and modification of the level of brightness by a factor of 0.05 to 1.5 times the original intensity level.


(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 3.4: Augmented Images Representing letter A

The data augmentation is for introducing much more variability and diversity into the dataset which is crucial for reducing overfitting and understand the model's performance on unseen data. Having different hand orientations in different lighting conditions, the models will be able to generalize more effectively.

3.3 Dataset Distribution

Training Set: The training set consists of input data and corresponding labels used to teach the model. Through optimization techniques, the model refines its internal parameters to minimize errors, gaining expertise from this data.

Testing Set: The testing set, separate from training, evaluates the model's generalization ability. It assesses how well the model predicts unseen data by comparing its outputs to actual results.

Validation Set: The validation set assists in tuning model parameters. It gauges performance on unseen data, aiding in adjustments to enhance generalization and prevent overfitting.

Dataset	**Number of Images**	**Percentage**
Train	60,900	70%
Test	13,050	15%
Validation	13,050	15%
Total		87,000

Table 3.1: Dataset Split

3.4 Overview of Binarized Models

Binary Neural Networks (BNNs) utilize binary weights and activation function parameters to replace full-precision values, simplifying the complex calculations of Convolutional Neural Networks (CNNs) into simple bitwise operations. This optimization significantly reduces computation and memory storage requirements, resulting in lower area and power consumption compared to full-precision models. Despite these advantages, the binarization process impacts the performance and accuracy of the models [11].

$$\mathrm{sign(x)} = \begin{cases} +1 & \text{if } x \geq 0, \\ -1 & \text{otherwise.} \end{cases}$$

Training BNNs involves forward and backward propagation, similar to traditional neural networks. However, in BNNs, binarization techniques are applied to both weights and activations. During forward propagation, inputs pass through the network, with binary convolutions performed using XNOR and pop-count operations.

In backward propagation, gradients are calculated using techniques like the Straight-Through Estimator (STE) to handle the non-differentiable sign function effectively.

$$\text{Forward: } r_o = \text{sign}(r_i)$$
$$\text{Backward: } \frac{\partial c}{\partial ri} = \frac{\partial c}{\partial ro} \mathbf{1}_{|ri| \leq t_{\text{clip}}}$$

BNNs are applied in various domains, including image classification, object detection, and speech recognition [23]. Common datasets used in these applications include:

- MNIST, CIFAR-10, and SVHN for initial experiments and small-scale tasks.
- ImageNet for large-scale image classification.
- COCO and PASCAL VOC for object detection tasks.

In our application, RGB images of sign language images are not complex which proves to be a good scenario where binary neural networks can prove to be useful. BNNs, with their binary weights and activations, drastically reduce the computational complexity further, making them efficient for processing sign language imagery, even on resource-constrained devices.

3.5 LARQ: Library for Binarized Models

LARQ is a TensorFlow-based library that facilitates the training and deployment of Binary Neural Networks. Developed by Plumerai, LARQ aims to provide a user-friendly interface for building, training, and evaluating BNN models, empowering both researchers and practitioners to leverage the benefits of binary networks in their deep learning workflows [11].

- Efficient Binary Operations: LARQ implements efficient binary operations and custom layers optimized for BNNs, ensuring fast and memory-efficient execution of binary computations.
- Pre-trained Binarized Models: LARQ comes with pre-trained BNN models and comprehensive tutorials to help users get started quickly and explore the capabilities of binary networks.
- Integration with TensorFlow Ecosystem: Being built on TensorFlow, LARQ seamlessly integrates with the broader TensorFlow ecosystem, allowing users to leverage existing TensorFlow functionalities and tools.

Binarrised models would be loaded from Larq Zoo. It is a Python library that comes with pre-trained binary models. They are compared against the full precision TensorFlow models.

3.6 Models Used

Our model selection procedure prioritised the inclusion of both binarized and classical full-precision models, which are well-known and used in image classification. To compare these models on comprehensive ground, we selected models that spanned across a wide variety of topologies and complexities. The most relevant purpose of this comparison is to contrast binarized neural network models against applications with standard full-precision models. We see evidence of equivalently performing binarized models, when compared with traditional models, that could exist with memory and computational efficiency benefits measured by the metrics of accuracy, precision, and recall across diverse classes of domains.

3.6.1 BinaryDenseNet45

BinaryDenseNet-45 is a variation of DenseNet that incorporates binary weights and binary activations. Binary networks aid in reducing the use of memory and lowering computational complexity during inference. These improvements are added to the 45-layer model without losing the dense connectivity architecture, which is of key importance for feature reuse and effective gradient propagation in DenseNet. This architecture achieves deep information extraction efficiently and still keeps amazing classification accuracy, such that it really becomes helpful in resource-constrained conditions [3].

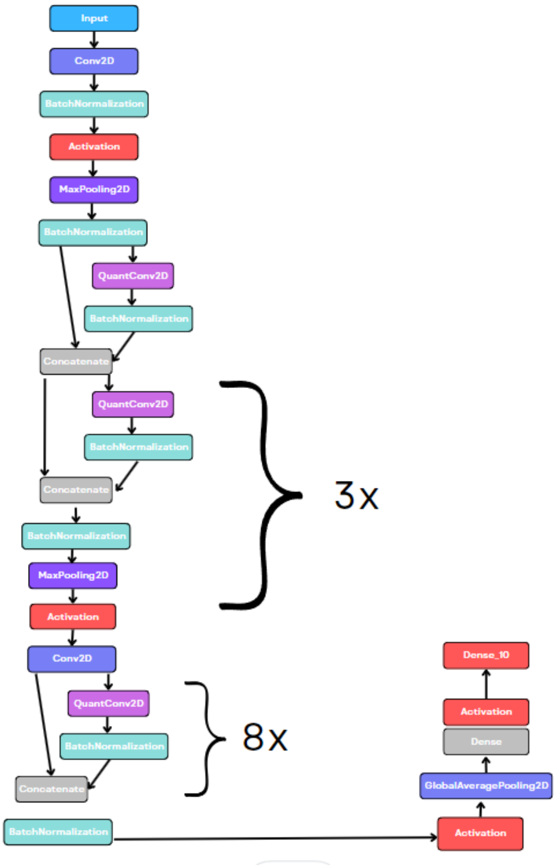


Figure 3.5: BinaryDenseNet45 Model Architecture

3.6.2 BinaryResNetE18

BinaryResNetE18 uses the ResNet architecture, which is a very efficient way to train deep neural networks. On its part, BinaryResNetE18 has 18 layers and maintains residual connections; it uses only binary weights and activations for efficient inference but retains residual connections. Being a balanced network in terms of depth and computational efficiency, BinaryResNetE18 comes in handy in deep learning tasks for it works well for moderately deep networks with high demands on performance. [3].

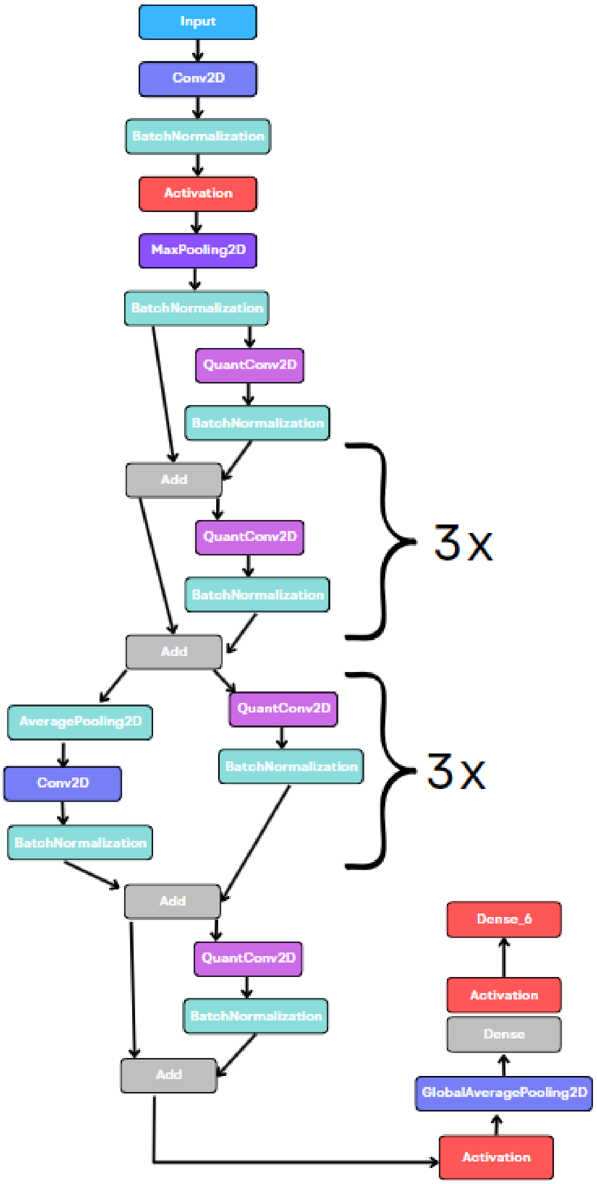


Figure 3.6: BinaryResNetE18 Model Architecture

3.6.3 BinaryDenseNet37

BinaryDenseNet37 is another DenseNet adaptation that uses binary weights and activations to have efficient inference [3]. Similar to the BinaryDenseNet45 model, BinaryDenseNet37 makes use of DenseNet's dense connection structure to enable feature re-use and good gradient flow during training. BinaryDenseNet37 strikes a balance between model depth and computational efficiency. Thus, it forms a good architecture for tasks that desire moderately deep networks with efficient inference abilities.

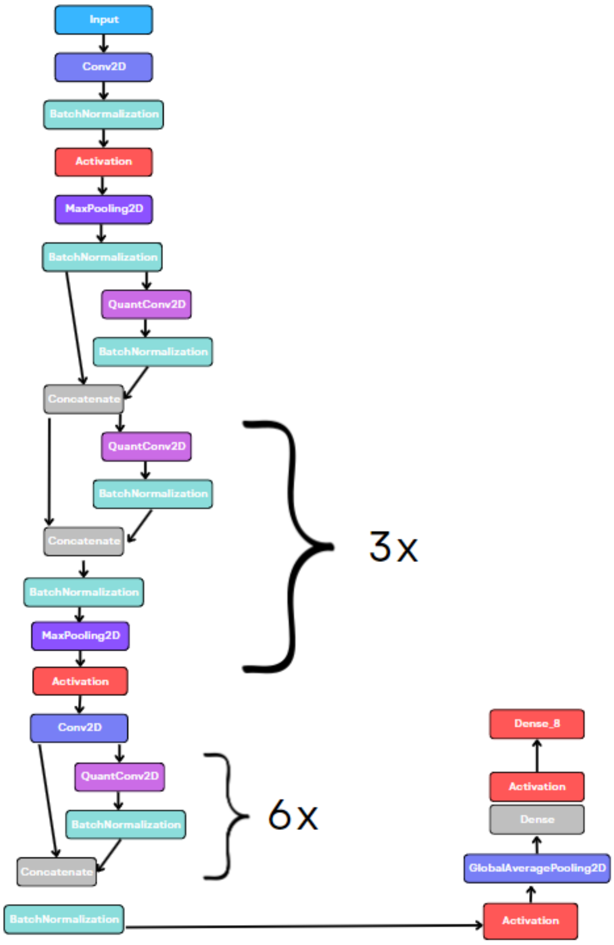


Figure 3.7: BinaryDenseNet37 Model Architecture

3.6.4 DenseNet121

DenseNet121 is an advanced convolutional neural network with dense blocks for image classification. This network has different layers that are connected directly to each other, efficiently interchanging information and features through four dense blocks having a number of layers. It attains accuracy using a growth rate of 32, together with transition layers applied for downsampling, with only 7 million parameters and 121 layers [2]. This architecture is very effective for picture datasets like ImageNet because it is capable of optimizing precision through feature reuse and dense connections.

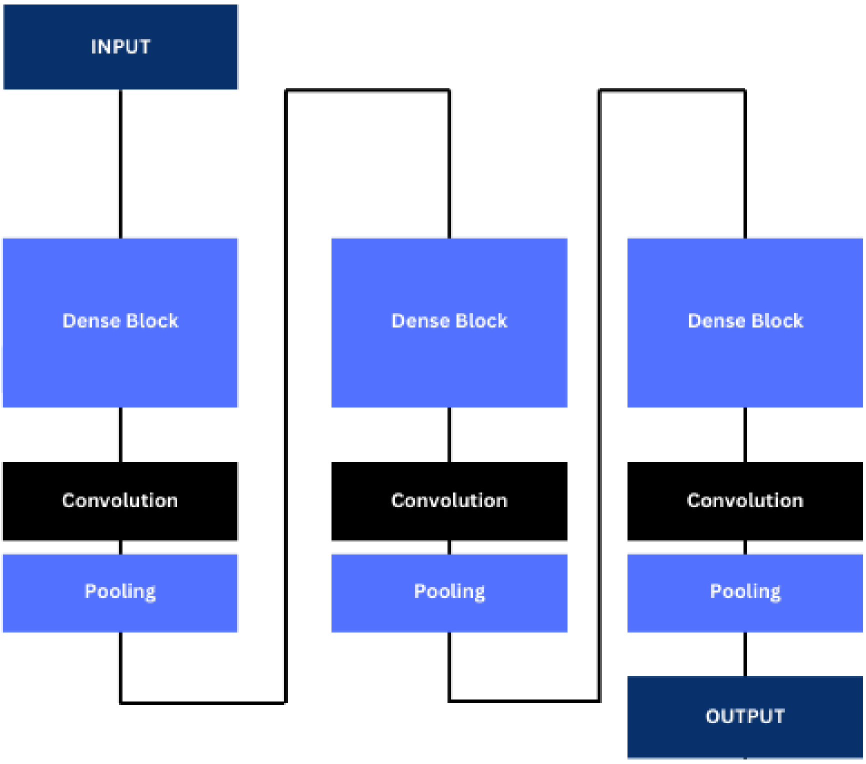


Figure 3.8: DenseNet121 Model Architecture

3.6.5 Resnet50

ResNet50, with its 50-layer deep architecture, is another CNN that is designed specifically to classify images. The uniqueness of the model is in the use of identity shortcuts which are helpful in preventing the vanishing gradient, that simply allows each block to shortcut its layers from an input to an output with identity mapping. ResNet50 uses 7x7 convolutions and max pooling, to begin with, followed by 0 convolutions, which results in reducing dimensionality to the required levels [1]. Batch normalisation, which is introduced before the activation itself, will make the training much faster. The ResNet50 architecture balances complexity with simplicity, having 23 million parameters. That remaining connection goes to help exact feature learning, which can be deployed on an outstanding performance in image recognition applications..

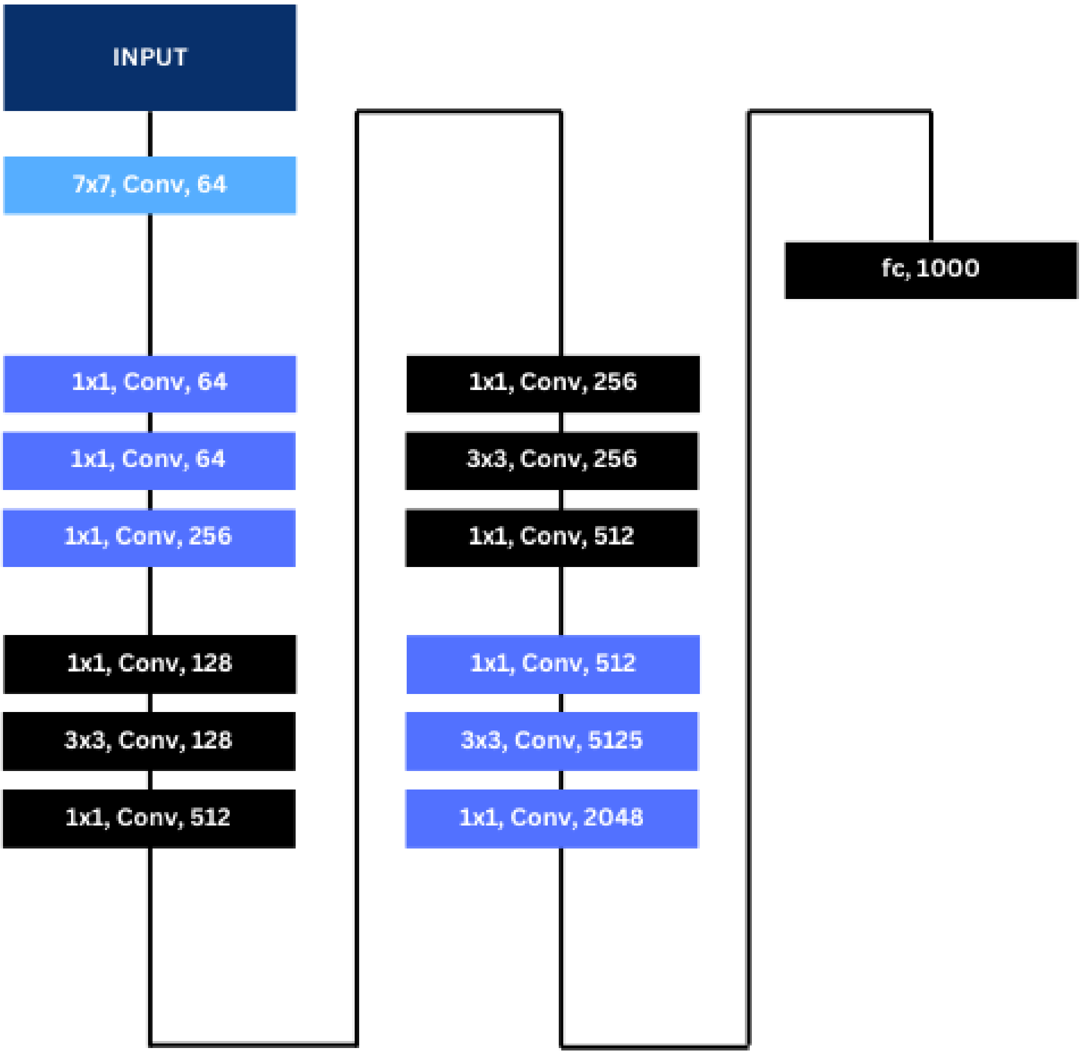


Figure 3.9: ResNet50 Model Architecture

3.6.6 VGG16

VGG16 belongs to those famous designs of convolutional neural networks that perform remarkable in picture classification duties. One of the major characteristics of the VGG16 architecture is its homogeneity, consisting of 16 weight layers arranged in convolutional blocks with max-pooling in between, making it easy to deploy and understand. VGG16 performs the best on many datasets despite its simple architecture. However, its large parameter count, approximately 138 million, makes it computationally expensive compared to more modern architectures.

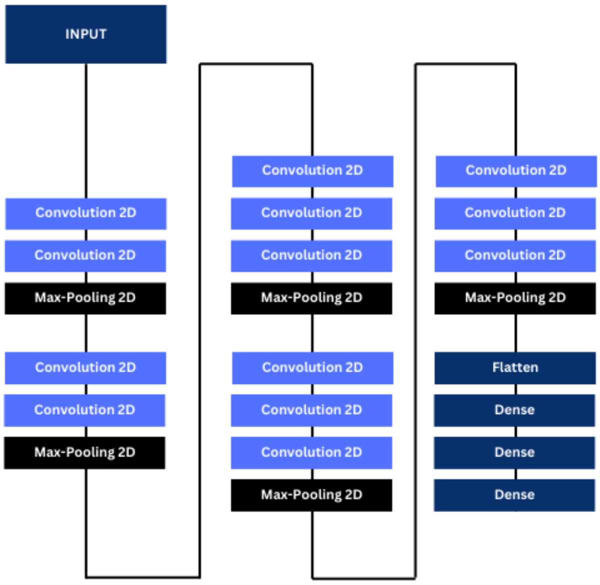


Figure 3.10: VGG16 Model Architecture

3.7 Experimental Setup

3.7.1 Model Training and Evaluation

Model	Batch Size	Epochs	Optimizer	Learning Rate
BinaryDenseNet45	32	30	Adam	1×10^{-5}
BinaryResNetE18	32	30	Adam	1×10^{-5}
BinaryDenseNet37	32	30	Adam	1×10^{-5}
ResNet50	32	30	Adam	1×10^{-5}
DenseNet121	32	30	Adam	1×10^{-5}
VGG16	32	30	Adam	1×10^{-5}

Table 3.2: Parameter Comparison of Different Models

For all models, the following steps were performed:

1. **Model Initialization**: Initialize the respective model architecture with appropriate parameters, including input shape (200x200) and number of classes (29).

2. **Model Compilation**: Compile the model using the mentioned optimizer and loss function, and accuracy as the evaluation metric.

3. **Callback Definition**: Define callbacks, such as EarlyStopping for stopping the training process if the metric stops improving for a certain number of epochs preventing over-fitting (Patience = 5). Also, ModelCheckpoint to save the weights at intervals which allows us to use the best performing model based on validation set performance.

4. **Model Training**: Train the model using the `fit` method with the training data, specifying the number of epochs (30) and batch size (32). Include validation data for monitoring performance during training.

5. **Model Saving**: Save the trained model weights to a file for future use.

6. **Best Model Loading**: Load the the model with the highest performance based on the validation set from the saved checkpoint file.

7. **Performance Visualization**: Visualize the loss and accuracy curves of the validation and training sets, to visualize the model's learning progress over epochs.

3.7.2 PC Configuration

The model's training and testing were run on a mid-range desktop computer with a Ryzen 5 3600 processor, 16 GB of RAM running on 3200 MHz, Nvidia GeForce GTX 1060, and a 64-bit version of Windows 10 professional.

3.7.3 Environment and Software Versions

The following software versions were used in this research:

- Windows 10 Pro Version 10.0.19045 Build 19045
- Python version: 3.8.0
- TensorFlow version: 2.4.0
- NumPy version: 1.22.4
- Matplotlib version: 3.7.4
- Keras version: 2.4.0
- Larq version: 0.13.1
- Larq Zoo version: 2.3.2
- OpenCV version: 4.9.0

Chapter 4

Results

4.1 Performance Measure

Train-Validation curve: Two sets of curves are generated for training and validation. One set represents the loss function, indicating the evolution of the loss value throughout training. The loss function measures the disparity between the model's predicted outputs and the actual target values. The other set illustrates the accuracy function, showcasing how the model's accuracy improves or remains consistent during training. Initially, the model's predictions may deviate significantly from the actual labels, leading to relatively low accuracy. As training progresses and the model adjusts its internal parameters based on labeled data, accuracy gradually improves.

Confusion Matrix: A confusion matrix is a table used to evaluate the performance of a model. It provides a detailed breakdown of the model's predictions compared to the actual ground truth values.

- **True Positive**: Count of positive predictions on positive data.
- **False Positive**: Count of positive predictions on negative data.
- **True Negative**: Count of negative predictions on negative data.
- **False Negative**: Count of negative predictions on positive data.

The matrix makes it more convenient and easier in determining how the model performed in it's testing phase.

Recall: Recall measures the ability of a classification model to correctly identify all relevant instances (true positives) among the total actual positive instances.

$$\text{Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}}$$

Accuracy: Accuracy refers to the proportion of accurately classified cases within a model that classifies variables.

$$\text{Accuracy} = \frac{\text{True Positives (TP)} + \text{True Negatives (TN)}}{\text{Total Instances}}$$

Precision: Precision, in a classification model, is the ability to correctly identify relevant instances (true positives) among the expected positive cases.

$$\text{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}}$$

F1-Score: The F1-score is the balanced harmonic mean of precision and recall.

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Total Parameters, Size and Inference Time: These parameters show the model's memory utilisation and, hence, its suitability for resource-constrained situations or low bandwidth applications. The total number of parameters indicates each model's complexity. Inference time refers to how long it takes for a trained machine learning model to predict new data after deployment.

4.2 Learning Curves

4.2.1 BinaryDenseNet45

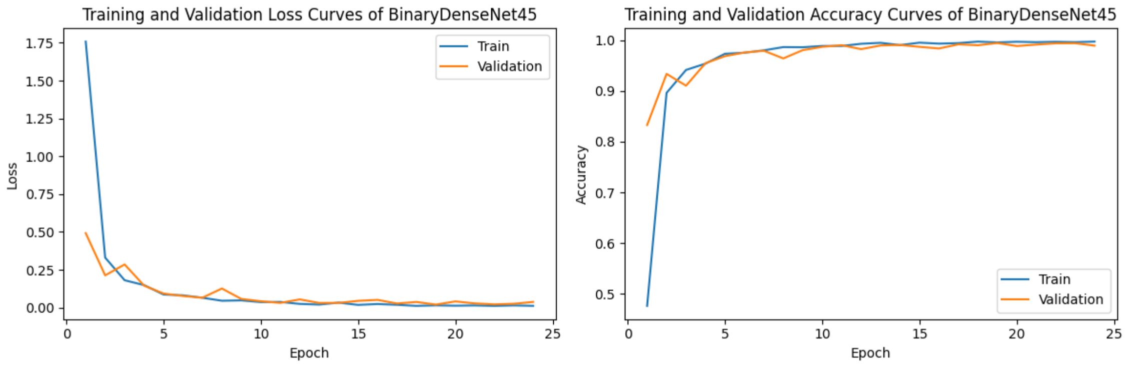


Figure 4.1: Loss and Accuracy curves of BinaryDenseNet45

The learning curves show the performance of the BinaryDenseNet45 model in terms of loss and accuracy during training and validation over epochs. Owing to the large adjustments of model weights by the data, there is a great fluctuation in the curve of loss and accuracy, especially at the initial training stage. For the advanced training process, the fluctuations in both the loss and accuracy tend to become stable, with a tendency of increased convergence to the optimal. The early stopping mechanism halted training at the 24th epoch, triggered by validation loss plateauing or deteriorating beyond a predefined threshold, preventing overfitting and ensuring the model's generalization ability. The total training time of approximately 25278.10 seconds reflects the computational effort required for each epoch, influenced by the model's complexity and the dataset size.

4.2.2 BinaryResNetE18

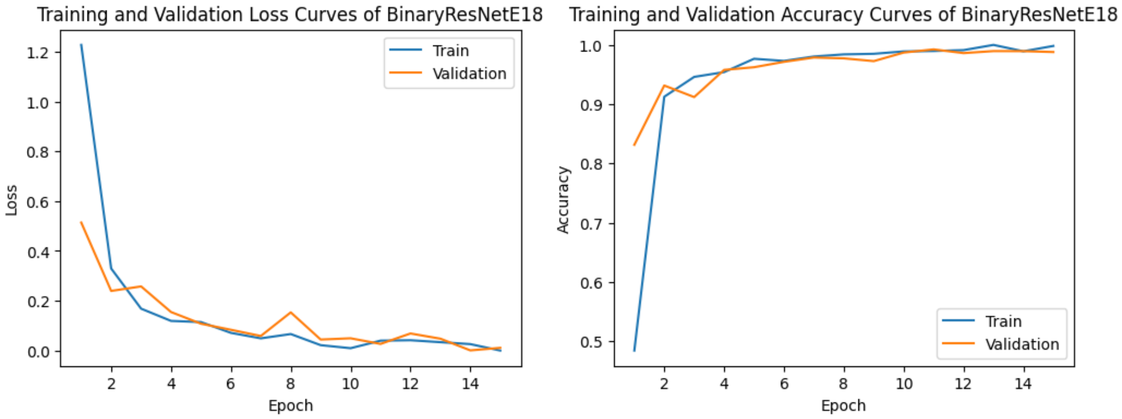


Figure 4.2: Loss and Accuracy curves of BinaryResNetE18

BinaryResNetE18 model exhibit similar characteristics to the previously discussed BinaryDenseNet45. Initially, there are noticeable spikes in both loss and accuracy curves, reflecting the model's rapid adjustments to training data. As training progresses, these fluctuations diminish, indicating improved stability and convergence. The early stopping mechanism intervened at the 15th epoch, likely triggered by validation metrics no longer improving or even worsening, thereby preventing overfitting and ensuring the model's ability to generalize to unseen data. The total training time of approximately 23221.23 seconds.

4.2.3 BinaryDenseNet37

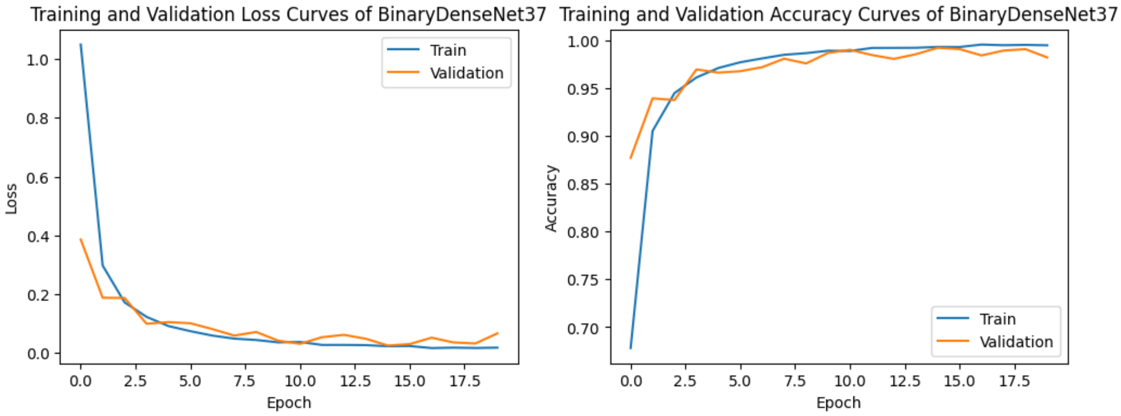


Figure 4.3: Loss and Accuracy curves of BinaryDenseNet37

The learning curves for the BinaryDenseNet37 model depict a similar pattern as observed in the previous model discussions. Initially, there are noticeable fluctuations in both loss and accuracy curves during early epochs, indicative of the model's rapid adaptation to the training data. As training progresses, these fluctuations stabilize, demonstrating improved convergence and consistency in performance. At the 20th epoch, early stopping was activated since the validation loss stopped its improvement in order to prevent overfitting and assure the generalization of the model to new data. The total training time amounted to approximately 15652.22

4.2.4 ResNet50

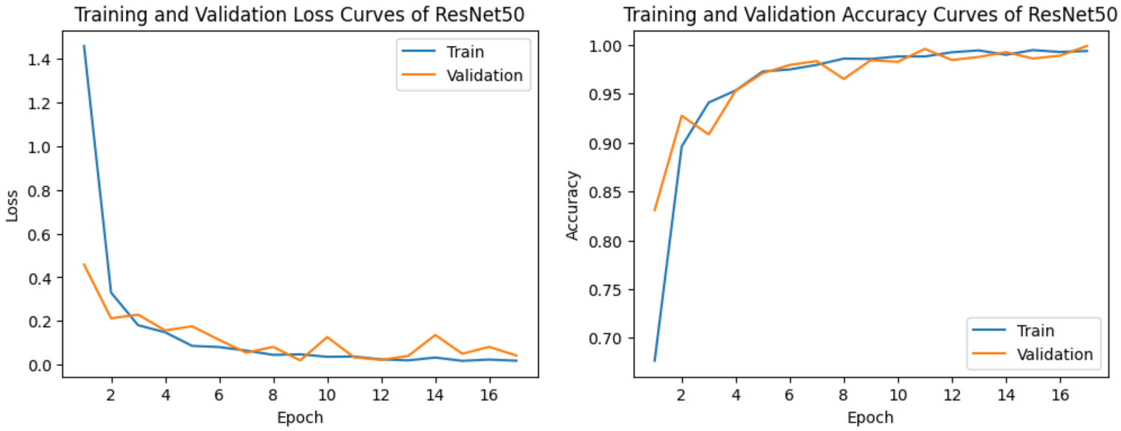


Figure 4.4: Loss and Accuracy curves of ResNet50

Learning curves for the ResNet50 model are regular. The validation loss and accuracy curves show oscillations at the beginning: these are usual behaviors in the learning of a model when it adapts to patterns in the training data. These initial oscillations gradually settle and finally show better convergence and consistency in the performance measurements. Early stopping was incorporated during the 17th epoch of training to effectively control overfitting and generalize the model. Total time around 17151.21 seconds for training.

4.2.5 DenseNet121

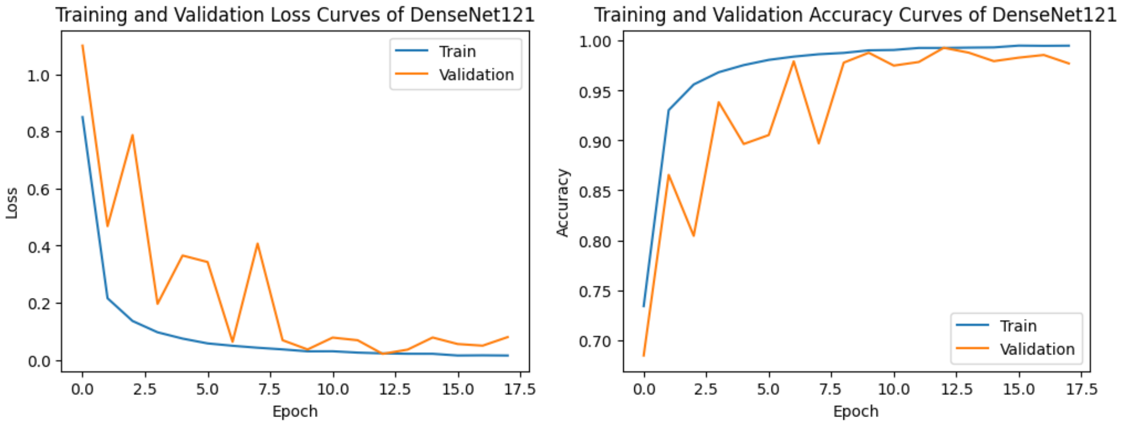


Figure 4.5: Loss and Accuracy curves of DenseNet121

The learning curves of DenseNet121 show a typical training behavior of a neural network; they fluctuate in both validation metrics. The high rate of validation loss and accuracy in the early epochs is because the model is highly sensitive to early changes with respect to model training data and learning dynamics. After that, the spikes become less frequent and, generally, the training tends to stabilize and thus finally move in the direction of ideal performance. Early stopping was triggered at the 18th epoch as the validation metrics became flat or degraded above a preset tolerance limit. This helped in reducing the chances of overfitting and increasing generalization. The training took about 14593.10 seconds

4.2.6 VGG16

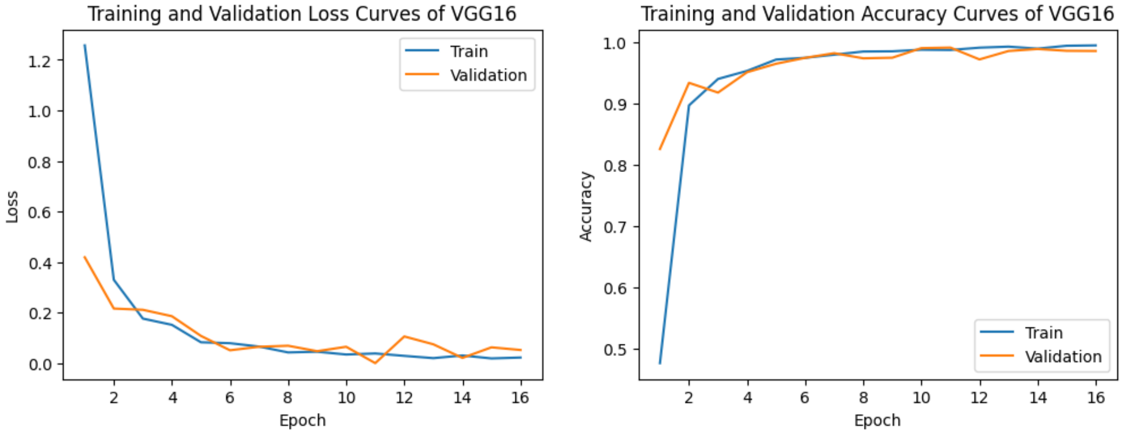
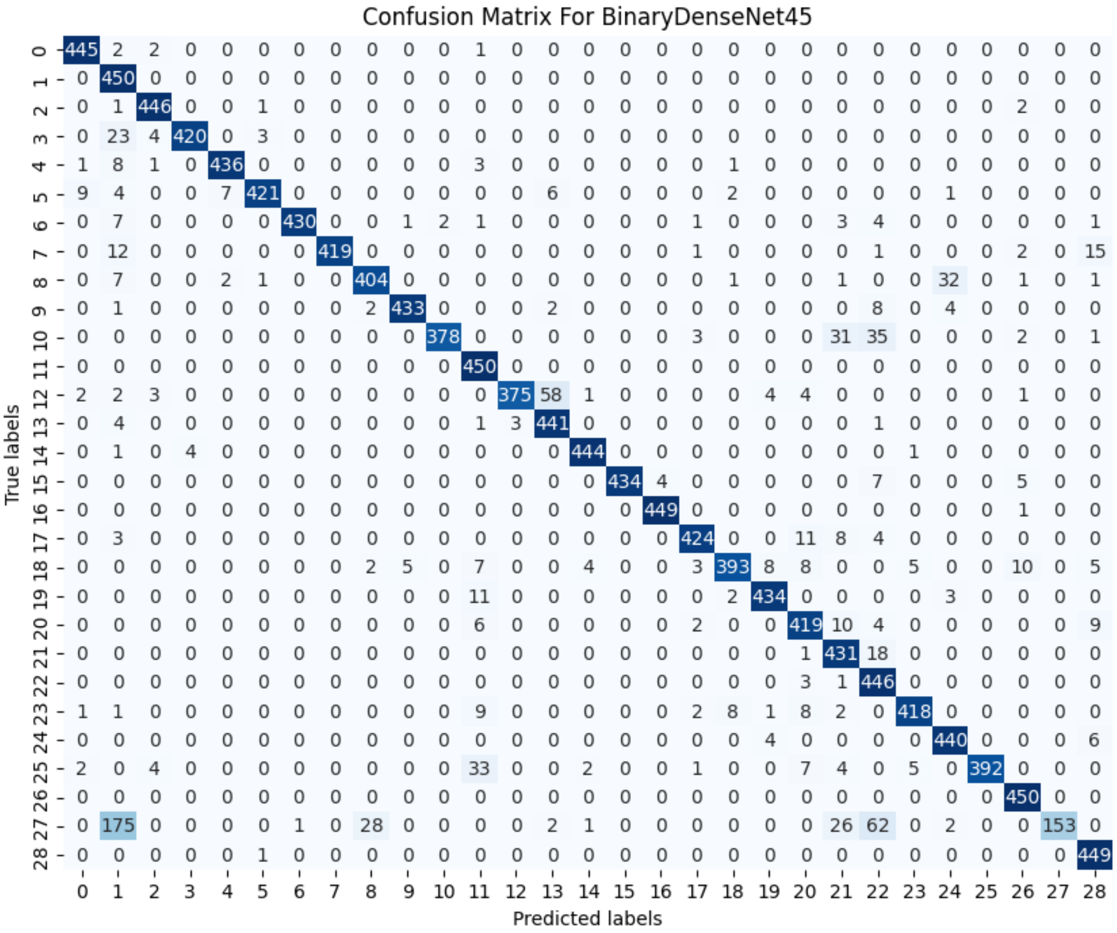


Figure 4.6: Loss and Accuracy curves of VGG16

The initial epochs show relatively larger variations in loss and accuracy, which is common as the model begins to learn and adapt to the dataset. As training progresses, these fluctuations decrease, indicating the model's improved stability and convergence towards optimal performance. Early stopping was initiated at the 16th epoch. The total training time of approximately 19213.21 seconds reflects the computational resources consumed per epoch, influenced by the model's architecture complexity and dataset size.

4.3 Confusion Matrices

4.3.1 BinaryDenseNet45

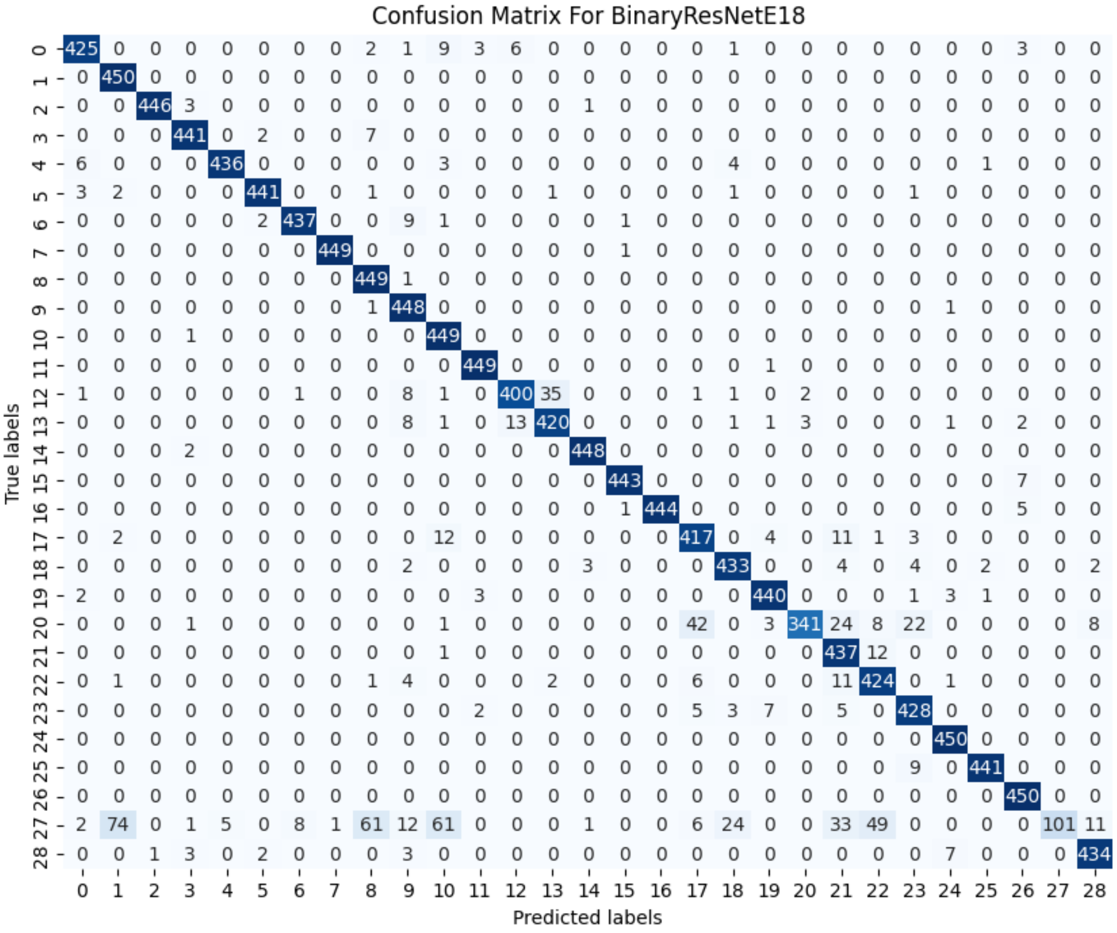


	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
0	445	2	2	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	450	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	1	446	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0
3	0	23	4	420	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	1	8	1	0	436	0	0	0	0	0	0	3	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
5	9	4	0	0	7	421	0	0	0	0	0	0	0	6	0	0	0	0	2	0	0	0	0	0	1	0	0	0	0
6	0	7	0	0	0	0	430	0	0	1	2	1	0	0	0	0	0	1	0	0	0	3	4	0	0	0	0	0	1
7	0	12	0	0	0	0	0	419	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	2	0	15
8	0	7	0	0	2	1	0	0	404	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	32	0	1	0	1
9	0	1	0	0	0	0	0	0	2	433	0	0	0	2	0	0	0	0	0	0	0	0	8	0	4	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	378	0	0	0	0	0	0	3	0	0	0	31	35	0	0	0	2	0	1
11	0	0	0	0	0	0	0	0	0	0	0	450	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	2	2	3	0	0	0	0	0	0	0	0	0	375	58	1	0	0	0	0	4	4	0	0	0	0	0	1	0	0
13	0	4	0	0	0	0	0	0	0	0	0	1	3	441	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
14	0	1	0	4	0	0	0	0	0	0	0	0	0	0	444	0	0	0	0	0	0	0	0	1	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	434	4	0	0	0	0	0	7	0	0	0	5	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	449	0	0	0	0	0	0	0	0	0	1	0	0
17	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	424	0	0	11	8	4	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	2	5	0	7	0	0	4	0	0	3	393	8	8	0	0	5	0	0	10	0	5
19	0	0	0	0	0	0	0	0	0	0	0	11	0	0	0	0	0	0	2	434	0	0	0	0	3	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	2	0	0	419	10	4	0	0	0	0	0	9
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	431	18	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	1	446	0	0	0	0	0	0
23	1	1	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0	2	8	1	8	2	0	418	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	440	0	0	0	6
25	2	0	4	0	0	0	0	0	0	0	0	33	0	0	2	0	0	1	0	0	7	4	0	5	0	392	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	450	0	0
27	0	175	0	0	0	0	1	0	28	0	0	0	0	2	1	0	0	0	0	0	0	26	62	0	2	0	0	153	0
28	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	449

Figure 4.7: Confusion Matrix for BinaryDenseNet45

There are 9387 True Positives, 11296 True Negatives, 306 False Positives and 1011 False Negatives. The model performs well for most classes, with high numbers along the diagonal. However, there are notable misclassifications, particularly for class 28 (second to last row), where many instances are incorrectly classified as class 2, 9, 22, and 27. Class 13 also shows some confusion with class 12. Classes 11, 12, 16, 26, and 29 have perfect or near-perfect classification.

4.3.2 BinaryResNetE18

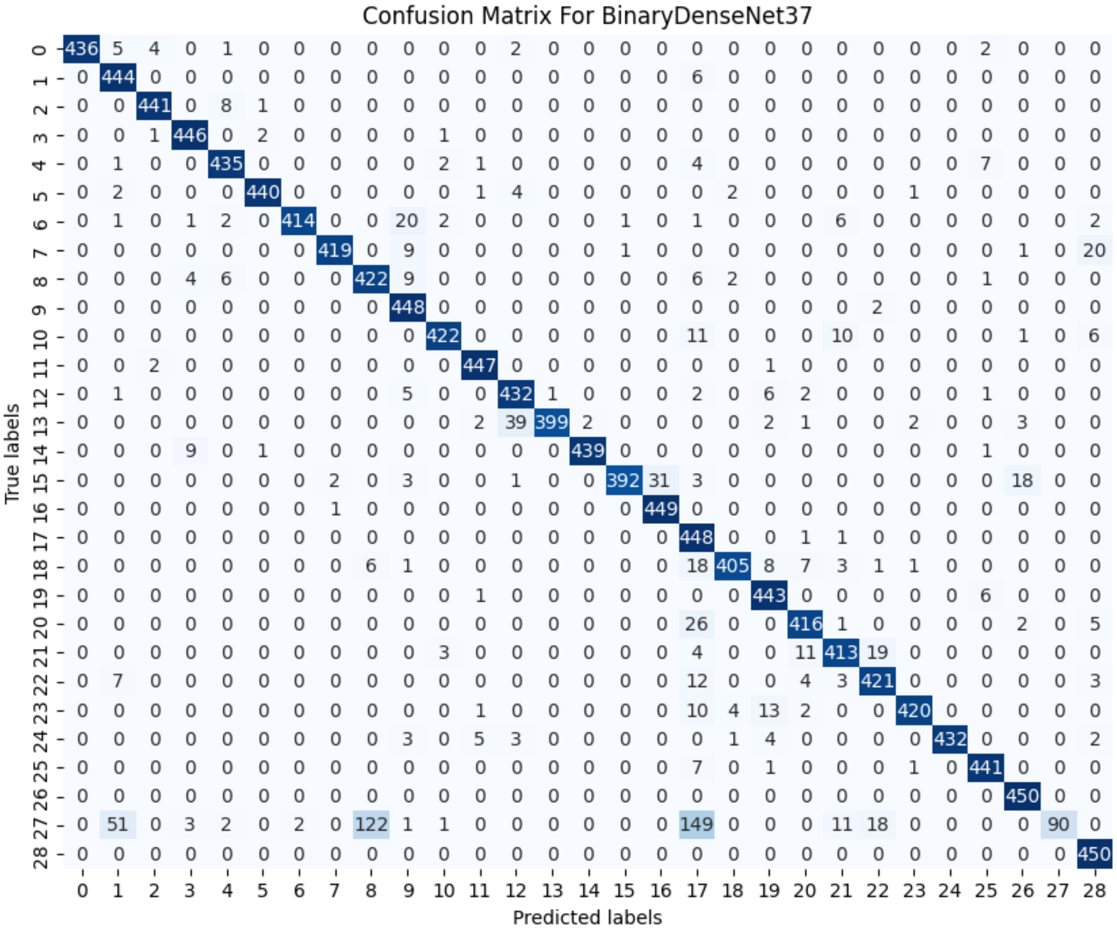


	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
0	425	0	0	0	0	0	0	0	2	1	9	3	6	0	0	0	0	0	1	0	0	0	0	0	0	0	3	0	0
1	0	450	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	446	3	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	441	0	2	0	0	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	6	0	0	0	436	0	0	0	0	0	3	0	0	0	0	0	0	0	4	0	0	0	0	0	0	1	0	0	0
5	3	2	0	0	0	441	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0
6	0	0	0	0	0	2	437	0	0	9	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	449	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	449	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	1	448	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
10	0	0	0	1	0	0	0	0	0	0	449	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	449	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
12	1	0	0	0	0	0	1	0	0	8	1	0	400	35	0	0	0	1	1	0	2	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	8	1	0	13	420	0	0	0	0	1	1	3	0	0	0	1	0	2	0	0
14	0	0	0	2	0	0	0	0	0	0	0	0	0	0	448	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	443	0	0	0	0	0	0	0	0	0	0	7	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	444	0	0	0	0	0	0	0	0	0	5	0	0
17	0	2	0	0	0	0	0	0	0	0	12	0	0	0	0	0	0	417	0	4	0	11	1	3	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	2	0	0	0	0	3	0	0	0	433	0	0	4	0	4	0	2	0	0	2
19	2	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	440	0	0	0	1	3	1	0	0	0
20	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	42	0	3	341	24	8	22	0	0	0	0	8
21	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	437	12	0	0	0	0	0	0
22	0	1	0	0	0	0	0	0	1	4	0	0	0	2	0	0	0	6	0	0	0	11	424	0	1	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	5	3	7	0	5	0	428	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	450	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	441	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	450	0	0
27	2	74	0	1	5	0	8	1	61	12	61	0	0	0	1	0	0	6	24	0	0	33	49	0	0	0	0	101	11
28	0	0	1	3	0	2	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7	0	0	0	434

Figure 4.8: Confusion Matrix for BinaryResNetE18

There are 16725 True Positives, 16563 True Negatives, 323 False Positives and 389 False Negatives. The BinaryResNetE18 model shows strong diagonal elements, indicating good overall accuracy. However, there are notable misclassifications, particularly in the last two rows and columns. Class 27 (third from last) shows significant confusion with several other classes, suggesting it may be a challenging category to distinguish. The model performs exceptionally well for some classes, achieving perfect classification (e.g., classes 1, 24, and 26). There's also noticeable confusion between classes 12 and 13, as well as among classes 20, 21, 22, and 23. The varying levels of misclassification across different class pairs suggest that some categories are more easily distinguishable than others for this model.

4.3.3 BinaryDenseNet37



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
0	436	5	4	0	1	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0
1	0	444	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0	0	0	0	0
2	0	0	441	0	8	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	1	446	0	2	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	1	0	0	435	0	0	0	0	0	2	1	0	0	0	0	0	4	0	0	0	0	0	0	0	7	0	0	0
5	0	2	0	0	0	440	0	0	0	0	0	1	4	0	0	0	0	0	2	0	0	0	0	1	0	0	0	0	0
6	0	1	0	1	2	0	414	0	0	20	2	0	0	0	0	1	0	1	0	0	0	6	0	0	0	0	0	0	2
7	0	0	0	0	0	0	0	419	0	9	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	20
8	0	0	0	4	6	0	0	0	422	9	0	0	0	0	0	0	0	6	2	0	0	0	0	0	0	1	0	0	0
9	0	0	0	0	0	0	0	0	0	448	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	422	0	0	0	0	0	0	11	0	0	0	10	0	0	0	0	1	0	6
11	0	0	2	0	0	0	0	0	0	0	0	447	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
12	0	1	0	0	0	0	0	0	0	5	0	0	432	1	0	0	0	2	0	6	2	0	0	0	0	1	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	2	39	399	2	0	0	0	0	2	1	0	0	2	0	0	3	0	0
14	0	0	0	9	0	1	0	0	0	0	0	0	0	0	439	0	0	0	0	0	0	0	0	0	0	1	0	0	0
15	0	0	0	0	0	0	0	2	0	3	0	0	1	0	0	392	31	3	0	0	0	0	0	0	0	0	18	0	0
16	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	449	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	448	0	0	1	1	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	6	1	0	0	0	0	0	0	0	18	405	8	7	3	1	1	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	443	0	0	0	0	0	6	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	26	0	0	416	1	0	0	0	0	2	0	5
21	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	4	0	0	11	413	19	0	0	0	0	0	0
22	0	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	0	0	4	3	421	0	0	0	0	0	3
23	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	10	4	13	2	0	0	420	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	3	0	5	3	0	0	0	0	0	1	4	0	0	0	0	432	0	0	0	2
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7	0	1	0	0	0	1	0	441	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	450	0	0
27	0	51	0	3	2	0	2	0	122	1	1	0	0	0	0	0	0	149	0	0	0	11	18	0	0	0	0	90	0
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	450

Figure 4.9: Confusion Matrix for BinaryDenseNet37

There are 11801 True Positives, 11961 True Negatives, 345 False Positives and 343 False Negatives. the BinaryDenseNet37 model seems to perform well, with most classes having high correct classification counts along the diagonal. However, there are some notable misclassifications, particularly for class 27, which is frequently confused with classes 8 and 17. Class 13 also shows some confusion with class 12. The matrix reveals that certain classes (e.g., 1, 2, 3, 10, 16, 26, 28, 29) have very high accuracy, while others (e.g., 27) have more room for improvement.

4.3.4 ResNet50

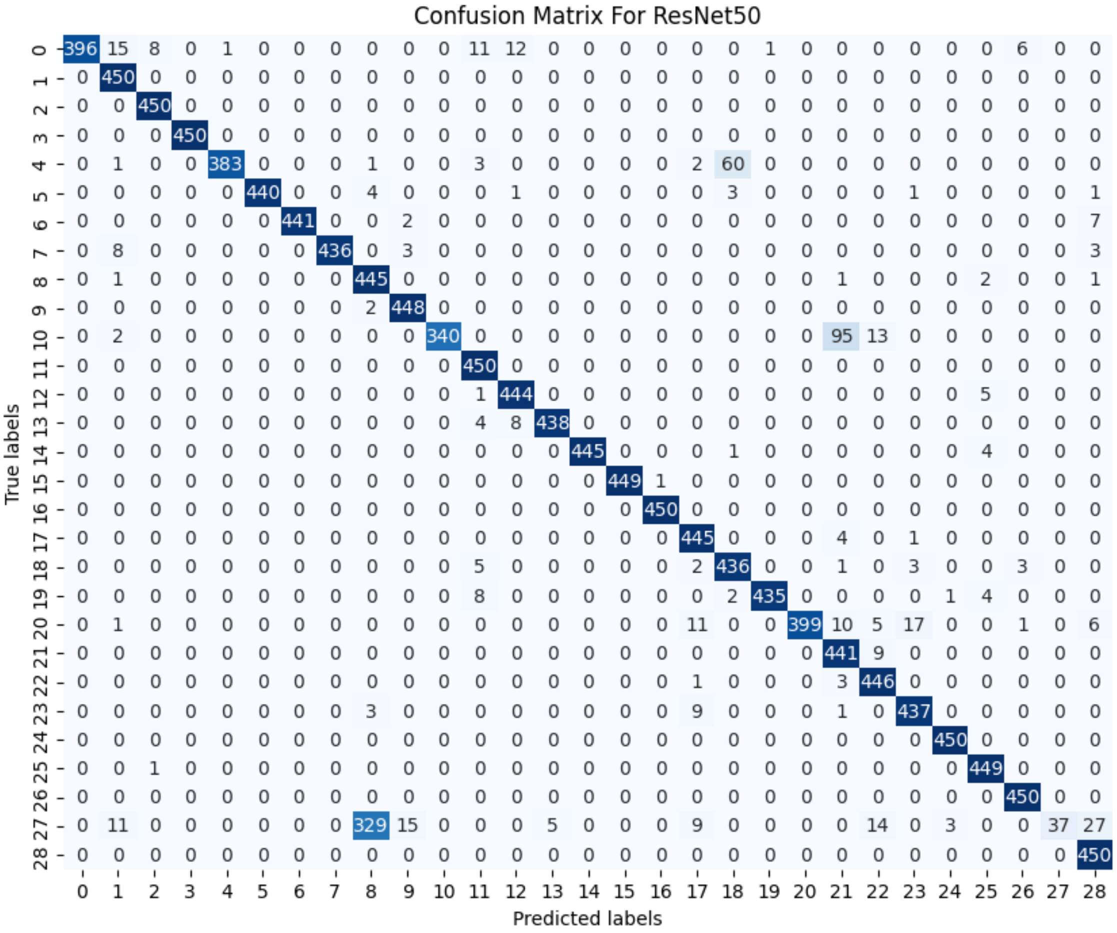
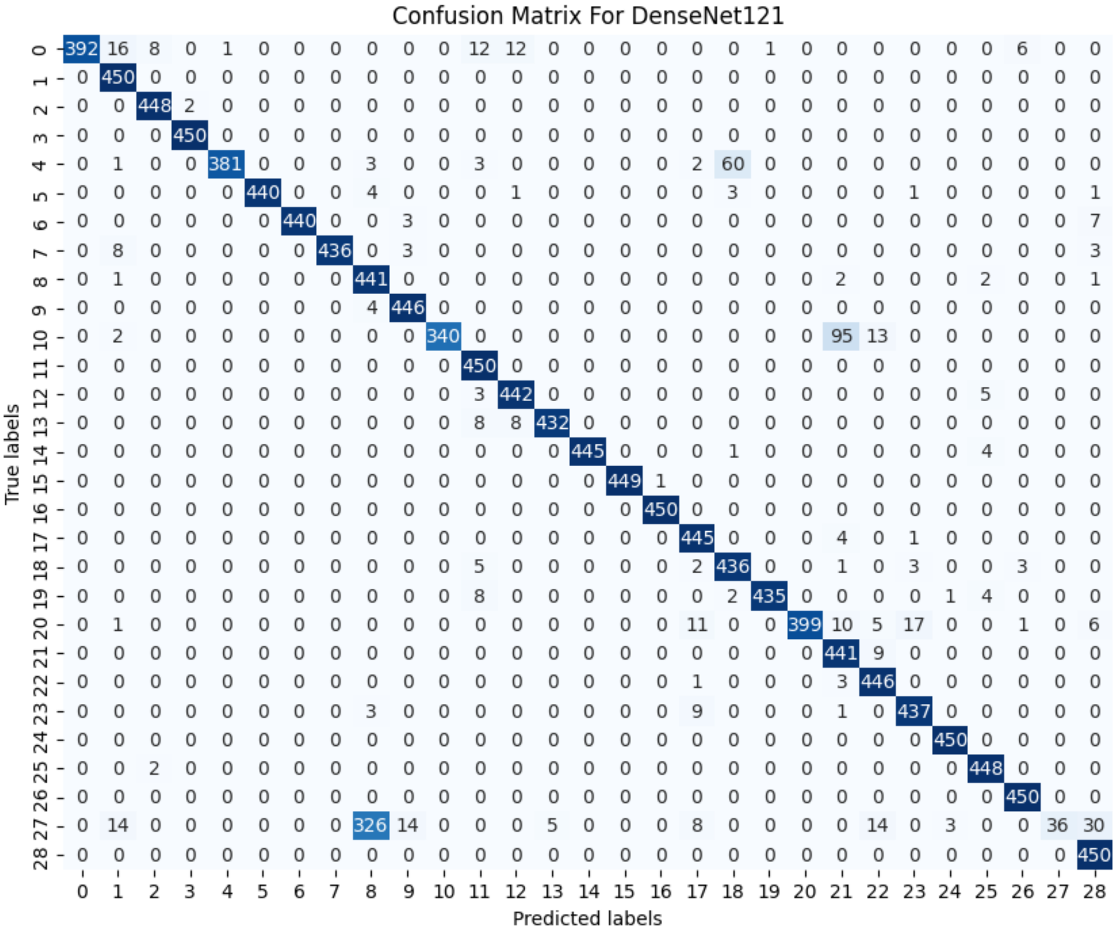


Figure 4.10: Confusion Matrix for ResNet50

There are 12945 True Positives, 12970 True Negatives, 105 False Positives and 30 False Negatives. The ResNet50 model shows strong performance overall, with high values along the diagonal indicating correct classifications. However, there are a few notable areas of confusion. Class 27 seems particularly problematic, with significant misclassifications spread across several other classes, especially classes 8 and 17. There's also some confusion between classes 13 and 12, and between classes 15 and 16. Classes 1, 2, 3, 4, 10, 17, 25, 26, and 29 show very high accuracy with few or no misclassifications. The last row and column (class 29) show perfect classification.

4.3.5 DenseNet121

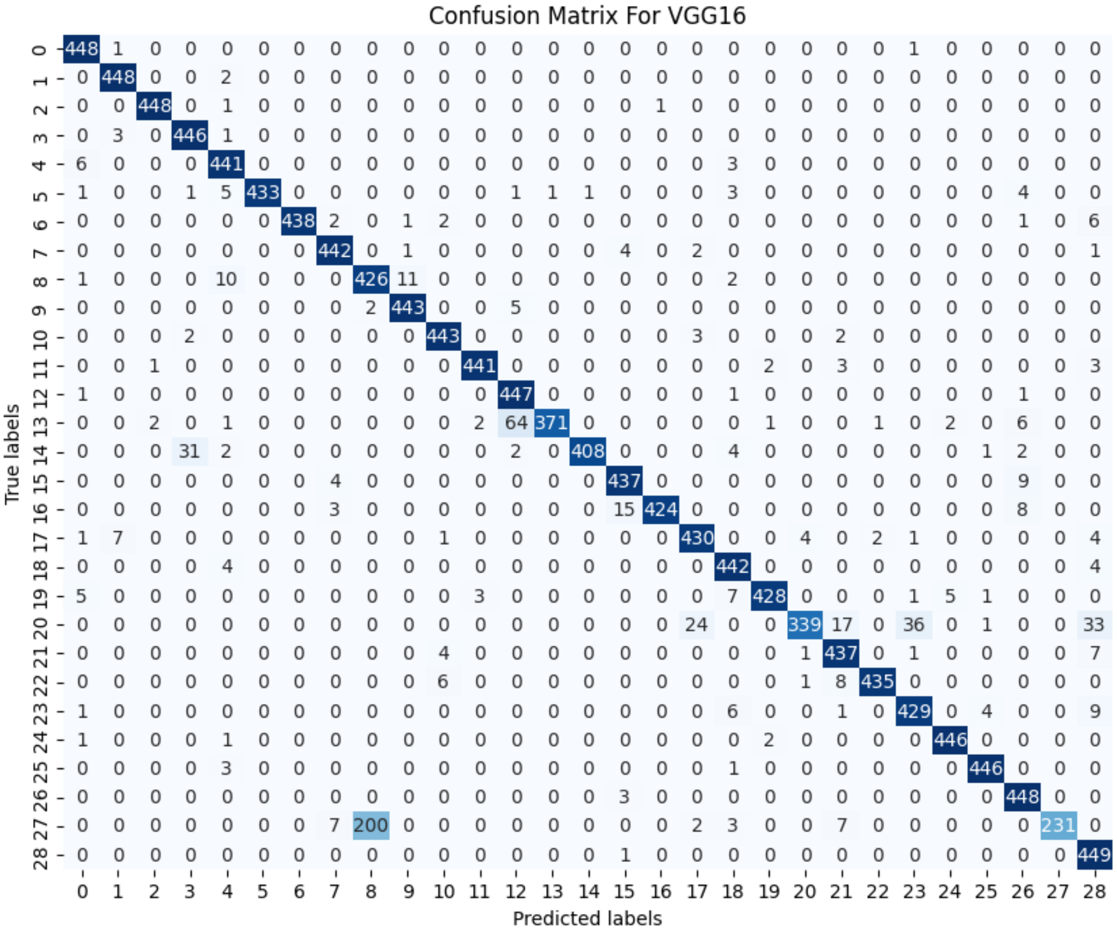


	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
0	392	16	8	0	1	0	0	0	0	0	0	12	12	0	0	0	0	0	0	1	0	0	0	0	0	0	6	0	0
1	0	450	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	448	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	450	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	1	0	0	381	0	0	0	3	0	0	3	0	0	0	0	0	2	60	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	440	0	0	4	0	0	0	1	0	0	0	0	0	3	0	0	0	0	1	0	0	0	0	1
6	0	0	0	0	0	0	440	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7
7	0	8	0	0	0	0	0	436	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3
8	0	1	0	0	0	0	0	0	441	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	2	0	0	1
9	0	0	0	0	0	0	0	0	4	446	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	2	0	0	0	0	0	0	0	0	340	0	0	0	0	0	0	0	0	0	0	95	13	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	450	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	3	442	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	8	8	432	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	445	0	0	0	1	0	0	0	0	0	0	4	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	449	1	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	450	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	445	0	0	0	4	0	1	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	2	436	0	0	1	0	3	0	0	3	0	0
19	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	2	435	0	0	0	0	1	4	0	0	0
20	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0	0	399	10	5	17	0	0	1	0	6
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	441	9	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	3	446	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	9	0	0	0	1	0	437	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	450	0	0	0	0
25	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	448	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	450	0	0
27	0	14	0	0	0	0	0	0	326	14	0	0	0	5	0	0	0	8	0	0	0	0	14	0	3	0	0	36	30
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	450

Figure 4.11: Confusion Matrix for DenseNet121

There are 11910 True Positives, 12219 True Negatives, 273 False Positives and 648 False Negatives. The DenseNet121 model shows strong diagonal elements, indicating good overall performance across most classes. However, there are some notable areas of confusion. Class 0 (first row) shows some misclassifications spread across multiple other classes. There's significant confusion between classes 10 and 21, with 95 instances of class 10 being misclassified as class 21. Classes 4 and 18 also show some mutual confusion. The model performs perfectly on several classes (e.g., 1, 3, 24, 26, 28) with all 450 instances correctly classified. Some classes like 27 show scattered misclassifications across multiple other classes.

4.3.6 VGG16



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
0	448	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
1	0	448	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	448	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
3	0	3	0	446	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	6	0	0	0	441	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0
5	1	0	0	1	5	433	0	0	0	0	0	0	1	1	1	0	0	0	3	0	0	0	0	0	0	0	4	0	0
6	0	0	0	0	0	0	438	2	0	1	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	6
7	0	0	0	0	0	0	0	442	0	1	0	0	0	0	0	4	0	2	0	0	0	0	0	0	0	0	0	0	1
8	1	0	0	0	10	0	0	0	426	11	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	2	443	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	2	0	0	0	0	0	0	443	0	0	0	0	0	0	3	0	0	0	2	0	0	0	0	0	0	0
11	0	0	1	0	0	0	0	0	0	0	0	441	0	0	0	0	0	0	0	2	0	3	0	0	0	0	0	0	3
12	1	0	0	0	0	0	0	0	0	0	0	0	447	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0
13	0	0	2	0	1	0	0	0	0	0	0	2	64	371	0	0	0	0	0	1	0	0	1	0	2	0	6	0	0
14	0	0	0	31	2	0	0	0	0	0	0	0	2	0	408	0	0	0	4	0	0	0	0	0	0	1	2	0	0
15	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	437	0	0	0	0	0	0	0	0	0	0	9	0	0
16	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	15	424	0	0	0	0	0	0	0	0	0	8	0	0
17	1	7	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	430	0	0	4	0	2	1	0	0	0	0	4
18	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	442	0	0	0	0	0	0	0	0	0	4
19	5	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	7	428	0	0	0	1	5	1	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	24	0	0	339	17	0	36	0	1	0	0	33
21	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	1	437	0	1	0	0	0	0	7
22	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0	0	0	1	8	435	0	0	0	0	0	0
23	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0	1	0	429	0	4	0	0	9
24	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	446	0	0	0	0
25	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	446	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	448	0	0
27	0	0	0	0	0	0	0	7	200	0	0	0	0	0	0	0	0	2	3	0	0	7	0	0	0	0	0	231	0
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	449

Figure 4.12: Confusion Matrix for VGG16

There are 11455 True Positives, 11447 True Negatives, 207 False Positives and 941 False Negatives. The VGG16 model performs well, with most classes having high correct classification counts (400+) along the diagonal. However, there are some notable confusions, such as between classes 13 and 14 (64 instances of class 14 misclassified as class 13), and between classes 20 and 23 (36 instances of class 23 misclassified as class 20). Classes 7, 16, and 17 show some mutual confusion. The model struggles most with class 21, which has the lowest correct classification count (339) and is often confused with classes 18, 23, and 29. Despite these issues, the model generally demonstrates strong performance across most classes.

4.4 Classification Reports

4.4.1 BinaryDenseNet45

Class	Precision	Recall	F1-score	Support
A	0.97	0.99	0.98	450
B	0.64	1.00	0.78	450
C	0.97	0.99	0.98	450
D	0.99	0.93	0.96	450
E	0.98	0.97	0.97	450
F	0.99	0.94	0.96	450
G	1.00	0.96	0.98	450
H	1.00	0.93	0.96	450
I	0.93	0.90	0.91	450
J	0.99	0.96	0.97	450
K	0.99	0.84	0.91	450
L	0.86	1.00	0.93	450
M	0.99	0.83	0.91	450
N	0.87	0.98	0.92	450
O	0.98	0.99	0.98	450
P	1.00	0.96	0.98	450
Q	0.99	1.00	0.99	450
R	0.97	0.94	0.96	450
S	0.97	0.87	0.92	450
T	0.96	0.96	0.96	450
U	0.91	0.93	0.92	450
V	0.83	0.96	0.89	450
W	0.76	0.99	0.86	450
X	0.97	0.93	0.95	450
Y	0.91	0.98	0.94	450
Z	1.00	0.87	0.93	450
del	0.95	1.00	0.97	450
nothing	1.00	0.34	0.51	450
space	0.92	1.00	0.96	450
Accuracy			0.93	13050
Macro avg	0.94	0.93	0.93	13050
Weighted avg	0.94	0.93	0.93	13050

Table 4.1: Classification report for BinaryDenseNet45

4.4.2 BinaryResNetE18

Class	**Precision**	**Recall**	**F1-score**	**Support**
A	0.97	0.94	0.96	450
B	0.85	1.00	0.92	450
C	1.00	0.99	0.99	450
D	0.98	0.98	0.98	450
E	0.99	0.97	0.98	450
F	0.99	0.98	0.98	450
G	0.98	0.97	0.98	450
H	1.00	1.00	1.00	450
I	0.86	1.00	0.92	450
J	0.90	1.00	0.95	450
K	0.83	1.00	0.91	450
L	0.98	1.00	0.99	450
M	0.95	0.89	0.92	450
N	0.92	0.93	0.93	450
O	0.99	1.00	0.99	450
P	0.99	0.98	0.99	450
Q	1.00	0.99	0.99	450
R	0.87	0.93	0.90	450
S	0.93	0.96	0.94	450
T	0.96	0.98	0.97	450
U	0.99	0.76	0.86	450
V	0.83	0.97	0.90	450
W	0.86	0.94	0.90	450
X	0.91	0.95	0.93	450
Y	0.97	1.00	0.99	450
Z	0.99	0.98	0.99	450
del	0.96	1.00	0.98	450
nothing	1.00	0.22	0.37	450
space	0.95	0.96	0.96	450
Accuracy			0.94	13050
Macro avg	0.95	0.94	0.93	13050
Weighted avg	0.95	0.94	0.93	13050

Table 4.2: Classification report for BinaryResNetE18

4.4.3 BinaryDenseNet37

Class	**Precision**	**Recall**	**F1-score**	**Support**
A	1.00	0.97	0.98	450
B	0.87	0.99	0.92	450
C	0.98	0.98	0.98	450
D	0.96	0.99	0.98	450
E	0.96	0.97	0.96	450
F	0.99	0.98	0.98	450
G	1.00	0.92	0.96	450
H	0.99	0.93	0.96	450
I	0.77	0.94	0.84	450
J	0.90	1.00	0.94	450
K	0.98	0.94	0.96	450
L	0.98	0.99	0.98	450
M	0.90	0.96	0.93	450
N	1.00	0.89	0.94	450
O	1.00	0.98	0.99	450
P	0.99	0.87	0.93	450
Q	0.94	1.00	0.97	450
R	0.63	1.00	0.77	450
S	0.98	0.90	0.94	450
T	0.93	0.98	0.95	450
U	0.94	0.92	0.93	450
V	0.92	0.92	0.92	450
W	0.91	0.94	0.92	450
X	0.99	0.93	0.96	450
Y	1.00	0.96	0.98	450
Z	0.96	0.98	0.97	450
del	0.95	1.00	0.97	450
nothing	1.00	0.20	0.33	450
space	0.92	1.00	0.96	450
Accuracy			0.93	13050
Macro avg	0.94	0.93	0.92	13050
Weighted avg	0.94	0.93	0.92	13050

Table 4.3: Classification report for BinaryDenseNet37

4.4.4 ResNet50

Class	Precision	Recall	F1-score	Support
A	0.96	1.00	0.98	450
B	0.98	1.00	0.99	450
C	0.99	1.00	0.99	450
D	0.93	0.99	0.96	450
E	0.94	0.98	0.96	450
F	1.00	0.96	0.98	450
G	1.00	0.97	0.99	450
H	0.97	0.98	0.97	450
I	1.00	0.95	0.97	450
J	0.97	0.98	0.98	450
K	0.97	0.98	0.98	450
L	0.99	0.98	0.98	450
M	0.86	0.99	0.92	450
N	1.00	0.82	0.90	450
O	1.00	0.91	0.95	450
P	0.95	0.97	0.96	450
Q	1.00	0.94	0.97	450
R	0.93	0.96	0.94	450
S	0.94	0.98	0.96	450
T	0.99	0.95	0.97	450
U	0.98	0.75	0.85	450
V	0.92	0.97	0.94	450
W	0.99	0.97	0.98	450
X	0.91	0.95	0.93	450
Y	0.98	0.99	0.99	450
Z	0.98	0.99	0.99	450
del	0.94	0.99	0.96	450
nothing	0.95	0.20	0.19	450
space	0.87	1.00	0.93	450
Accuracy			0.96	13050
Macro avg	0.96	0.96	0.96	13050
Weighted avg	0.96	0.96	0.96	13050

Table 4.4: Classification report for ResNet50

4.4.5 DenseNet121

Class	**Precision**	**Recall**	**F1-score**	**Support**
A	1.00	0.88	0.94	450
B	0.92	1.00	0.96	450
C	0.98	1.00	0.99	450
D	1.00	1.00	1.00	450
E	1.00	0.85	0.92	450
F	1.00	0.98	0.99	450
G	1.00	0.98	0.99	450
H	1.00	0.97	0.98	450
I	0.57	0.99	0.72	450
J	0.96	1.00	0.98	450
K	1.00	0.76	0.86	450
L	0.93	1.00	0.97	450
M	0.95	0.99	0.97	450
N	0.99	0.97	0.98	450
O	1.00	0.99	0.99	450
P	1.00	1.00	1.00	450
Q	1.00	1.00	1.00	450
R	0.93	0.99	0.96	450
S	0.87	0.97	0.92	450
T	1.00	0.97	0.98	450
U	1.00	0.89	0.94	450
V	0.79	0.98	0.88	450
W	0.92	0.99	0.95	450
X	0.95	0.97	0.96	450
Y	0.99	1.00	1.00	450
Z	0.97	1.00	0.98	450
del	0.98	1.00	0.99	450
nothing	1.00	0.08	0.15	450
space	0.91	1.00	0.95	450
Accuracy			0.94	13050
Macro avg	0.95	0.94	0.93	13050
Weighted avg	0.95	0.94	0.93	13050

Table 4.5: Classification report for DenseNet121

4.4.6 VGG16

Class	Precision	Recall	F1-score	Support
A	0.97	0.89	0.93	450
B	0.93	0.99	0.96	450
C	0.97	0.99	0.98	450
D	1.00	0.98	1.00	450
E	1.00	0.83	0.91	450
F	1.00	0.97	0.99	450
G	0.99	0.96	0.98	450
H	0.99	0.98	0.99	450
I	0.56	0.98	0.71	450
J	0.97	0.99	0.98	450
K	0.98	0.78	0.87	450
L	0.92	0.99	0.95	450
M	0.94	1.00	0.97	450
N	0.98	0.95	0.97	450
O	0.99	0.96	0.97	450
P	1.00	1.00	1.00	450
Q	1.00	1.00	1.00	450
R	0.96	0.98	0.97	450
S	0.88	0.96	0.92	450
T	0.99	0.98	0.99	450
U	1.00	0.91	0.95	450
V	0.81	0.96	0.88	450
W	0.95	0.98	0.97	450
X	0.94	0.96	0.95	450
Y	1.00	1.00	1.00	450
Z	0.97	0.99	0.98	450
del	0.99	0.99	0.99	450
nothing	1.00	0.09	0.16	450
space	0.92	1.00	0.96	450
Accuracy			0.94	13050
Macro avg	0.96	0.94	0.94	13050
Weighted avg	0.96	0.94	0.94	13050

Table 4.6: Classification Report for VGG16

4.5 Score Analysis

Model	Precision	Recall	F1-Score	Accuracy
BinaryDenseNet45	0.94	0.93	0.93	0.93
BinaryResNetE18	0.95	0.94	0.93	0.94
BinaryDenseNet37	0.94	0.93	0.92	0.93
ResNet50	0.96	0.96	0.96	0.96
DenseNet121	0.95	0.94	0.93	0.94
VGG16	0.96	0.94	0.94	0.94

Table 4.7: Comparison of Precision, Recall, F1-Score, and Accuracy

The table above demonstrates that binarized neural networks (BNNs) like BinaryDenseNet45 and BinaryResNetE18 achieve impressive precision, recall, F1-score, and accuracy metrics, closely trailing behind traditional models such as ResNet50 and VGG16. For instance, BinaryResNetE18 maintains a high precision of 0.95 and an accuracy of 0.94, comparable to DenseNet121 and VGG16. Overall, it shows that Binarized Neural Networks (BNNs) such as BinaryDenseNet45, BinaryResNetE18, and BinaryDenseNet37 maintain high performance metrics close to their non-binarized counterparts like ResNet50, DenseNet121, and VGG16.

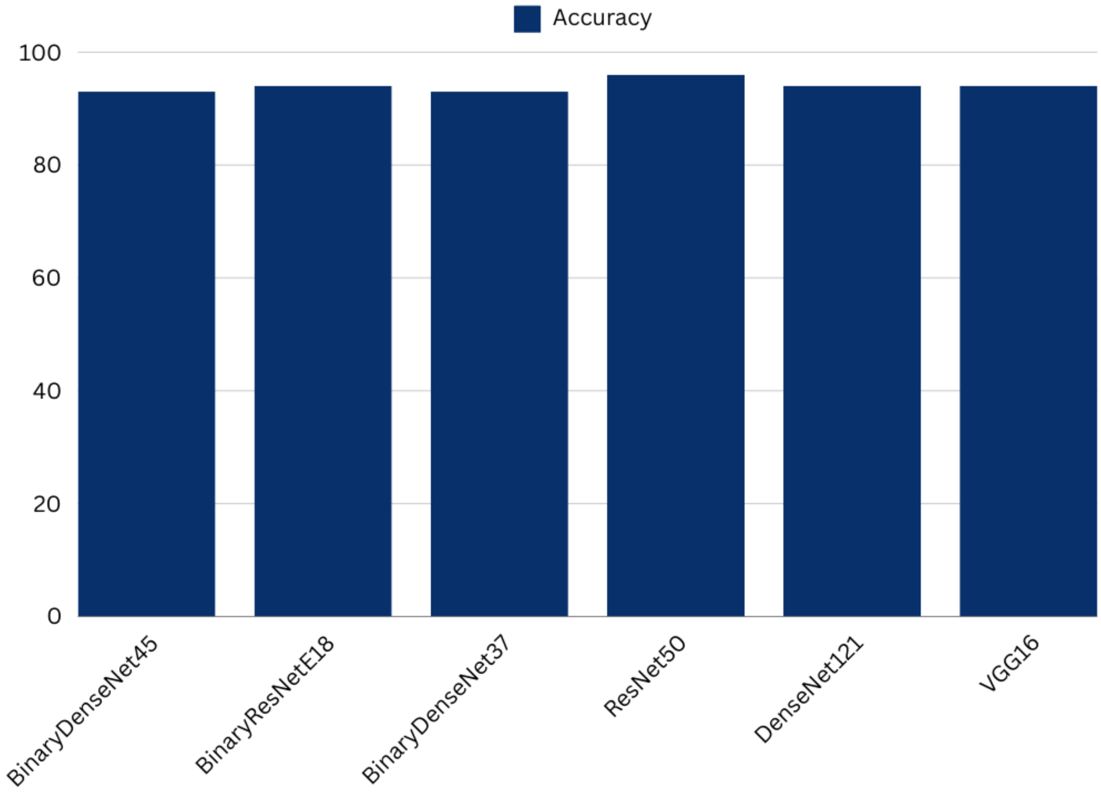


Figure 4.13: Model Accuracies Bar Chart

Despite the marginal differences in precision, recall, and F1-score, the accuracy of BNNs remains competitive as displayed by the barchart above. However, comparing binarized models to their non-binarized counterparts reveals distinct advantages and trade-offs in various performance metrics.

4.6 Model Size, Parameters, and Accuracy Comparison

Model	Model Size (MiB)	Total Parameters	Accuracy
BinaryDenseNet45	7.35	13.9M	0.93
BinaryResNetE18	4.00	11.7M	0.94
BinaryDenseNet37	5.13	8.7M	0.93
ResNet50	93.46	25.6M	0.96
DenseNet121	33.00	8.1M	0.94
VGG16	503.54	138M	0.94

Table 4.8: Comparison of Model Sizes, Total Parameters, and Accuracy

The table above clearly demonstrates the efficiency of BNNs. For instance, BinaryResNetE18, with an accuracy of 94%, has a model size of only 4.00 MiB and 11.7M parameters, compared to DenseNet121 which has the same accuracy but a higher model size of 33.00 MiB. Compared to the VGG16 model, it has a model size of 503.54 MiB and 138M total parameters which also has the same accuracy as the BinaryResNetE18 model. This makes binarized models such as the BinaryResNetE18 highly advantageous for deployment on resource-constrained devices with similar performance. Overall the binarized models like BinaryDenseNet45, BinaryResNetE18, and BinaryDenseNet37 exhibit competitive precision, recall, and F1-score metrics comparable to non-binarized models like ResNet50, DenseNet121, and VGG16. The binarized models exhibit significantly lower model sizes compared to the other models while having very similar performance metrics.

4.7 Inference Time Analysis

Model	Average Inference Time/Image (ms)
BinaryDenseNet45	5.139
BinaryResNetE18	4.223
BinaryDenseNet37	3.999
ResNet50	7.978
DenseNet121	9.056
VGG16	9.126

Table 4.9: Comparison of Average Inference Time per Image

The table compares the average inference time per image across various models, highlighting the efficiency of binarized neural networks (BNNs) compared to traditional architectures. For example, BinaryDenseNet37 and BinaryResNetE18 achieve average inference times of 3.999 ms and 4.223 ms, respectively, significantly outperforming ResNet50, DenseNet121, and VGG16, which require 7.978 ms, 9.056 ms, and 9.126 ms, respectively. Also, BinaryResNetE18 compared to DenseNet121 and VGG16, have the same accuracy but a significantly lower model size and average inference time. This efficiency advantage of BNNs underscores their suitability for real-time applications, where rapid processing of data is crucial, making them an optimal choice for resource-constrained environments.

4.8 Advantages of Binarized Models

Binarized Neural Networks (BNNs) offer several key advantages over traditional neural networks (NNs):

- **Reduced Model Size:** BNNs have significantly smaller model sizes, enabling easier deployment on resource-constrained devices.
- **Lower Computational Requirements:** BNNs require fewer computational resources, leading to faster inference times.
- **Comparable Performance:** Despite their smaller size and lower computational requirements, BNNs maintain competitive performance metrics (precision, recall, F1-score) compared to traditional NNs.

In the context of RGB images of sign language, where the input data is simple, having a smaller model size and fewer parameters can be advantageous. Binarized models are less computationally intensive, allowing them to run efficiently on devices with limited processing capabilities. Moreover, their reduced memory footprint makes them more suitable for deployment on edge devices or embedded systems, enabling real-time detection without relying heavily on cloud-based processing. Specific comparisons, such as BinaryResNetE18 versus VGG16 and DenseNet121, illustrate that BNNs provide a similar level of accuracy and other performance metrics while significantly reducing model size and inference time. This makes BNNs ideal for real-world applications. Binarized Neural Networks (BNNs) can be efficient alternatives to traditional neural networks. The trade-offs in accuracy are minimal, highlighting BNNs as a promising solution for various practical implementations.

Chapter 5

Conclusion

To conclude, this research focuses on the potential of binarized neural networks by comparing them to traditional neural networks on American Sign Language (ASL) recognition using RGB imagery. This combination of deep learning algorithms and RGB images has the potential to overcome the obstacles and communication gaps among hearing-impaired individuals. The study comprises a comprehensive literature review, augmented suitable dataset collection for RGB imagery, and the execution of deep learning models and algorithms for sign language detection. The optimized pipeline we developed demonstrates the potential of binarized neural networks (BNNs) in sign language recognition by interpreting distinct visual patterns associated with hand gestures. Furthermore, the research highlighted the potential of using binarized models, as their accuracies are competitive with those of traditional models. The reduced computational complexity, leading to better efficiency and lower memory requirements, are potential factors for accessibility and more efficient real-time deployments. Further investigation into more advanced binarized network architectures and multimodal approaches can elevate the optimized deep learning pipeline. We hope that this research initiates more investigations, innovations, and advancements into both binarized models and sign language recognition, eventually leading to improved communication experiences for the deaf and hearing-impaired communities.

5.1 Future Works

- **Real-Time Deployment:** Despite numerous claims and research highlighting the resource efficiency, speed, and lightweight nature of binarized models, our models have yet to undergo real-world testing. For practical deployment, especially in real-time applications, it is crucial to validate these models under real-world conditions. This involves extensive testing with a broader dataset to ensure reliability and performance. Moreover, deploying these models in real-world applications will provide insights into their operational effectiveness and identify potential areas for further optimization and improvement.

Bibliography

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015. arXiv: 1512.03385 [cs.CV].

[2] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," 2018. arXiv: 1608.06993 [cs.CV].

[3] J. Bethge, H. Yang, M. Bornstein, and C. Meinel, "Back to simplicity: How to train accurate bnns from scratch?," 2019. arXiv: 1906.08637 [cs.LG].

[4] G. Cerutti, R. Prasad, and E. Farella, "Convolutional neural network on embedded platform for people presence detection in low resolution thermal images," pp. 7610–7614, 2019. DOI: 10.1109/ICASSP.2019.8682998.

[5] M. M. Rahman, M. S. Islam, M. H. Rahman, R. Sassi, M. W. Rivolta, and M. Aktaruzzaman, "A new benchmark on american sign language recognition using convolutional neural network," pp. 1–6, 2019. DOI: 10.1109/STI47673.2019.9067974.

[6] M. Bastwesy, N. Elshennawy, and M. Saidahmed, "Deep learning sign language recognition system based on wi-fi csi," *International Journal of Intelligent Systems and Applications*, vol. 12, pp. 33–45, Dec. 2020. DOI: 10.5815/ijisa.2020.06.03.

[7] R. Harini, R. Janani, S. Keerthana, S. Madhubala, and S. Venkatasubramanian, "Sign language translation," pp. 883–886, 2020. DOI: 10.1109/ICACCS48705.2020.9074370.

[8] M. Jaiswal, V. Sharma, A. Sharma, S. Saini, and R. Tomar, "An efficient binarized neural network for recognizing two hands indian sign language gestures in real-time environment," pp. 1–6, 2020. DOI: 10.1109/INDICON49873.2020.9342454.

[9] S. Lacke, "Do all deaf people use sign language?," 2020. [Online]. Available: https://www.accessibility.com/blog/do-all-deaf-people-use-sign-language#:~:text=That's%5C%20because%5C%20not%5C%20all%5C%20deaf,1%5C%25%5C%20%5C%E2%5C%80%5C%94%5C%20use%5C%20sign%5C%20language..

[10] B. G. Lee, T.-W. Chong, and W.-Y. Chung, "Sensor fusion of motion-based sign language interpretation with deep learning," *Sensors*, vol. 20, no. 21, p. 6256, Nov. 2020, ISSN: 1424-8220. DOI: 10.3390/s20216256. [Online]. Available: http://dx.doi.org/10.3390/s20216256.

[11] T. Bannink, A. Bakhtiari, A. Hillier, L. Geiger, T. de Bruin, L. Overweel, J. Neeven, and K. Helwegen, “Larq compute engine: Design, benchmark, and deploy state-of-the-art binarized neural networks,” 2021. arXiv: 2011.09398 [cs.LG].

[12] D. S. Breland, S. B. Skriubakken, A. Dayal, A. Jha, P. K. Yalavarthy, and L. R. Cenkeramaddi, “Deep learning-based sign language digits recognition from thermal images with edge computing system,” *IEEE Sensors Journal*, vol. 21, no. 9, pp. 10 445–10 453, 2021. DOI: 10.1109/JSEN.2021.3061608.

[13] D. S. BRELAND, “Hand gestures recognition using thermal images,” p. 70, 2021. [Online]. Available: https://uia.brage.unit.no/uia-xmlui/handle/11250/2823720.

[14] A. Halder and A. Tayade, “Real-time vernacular sign language recognition using mediapipe and machine learning,” vol. 2, pp. 9–17, 5 2021. [Online]. Available: https://www.ijrpr.com/uploads/V2ISSUE5/IJRPR462.pdf.

[15] Z. Hein, T. P. Htoo, B. Aye, S. M. Htet, and K. Z. Ye, “Leap motion based myanmar sign language recognition using machine learning,” pp. 2304–2310, 2021. DOI: 10.1109/ElConRus51938.2021.9396496.

[16] J. Kan, K. Hu, M. Hagenbuchner, A. C. Tsoi, M. Bennamounm, and Z. Wang, “Sign language translation with hierarchical spatio-temporalgraph neural network,” 2021. arXiv: 2111.07258 [cs.CV].

[17] J. J. Raval and R. Gajjar, “Real-time sign language recognition using computer vision,” pp. 542–546, 2021. DOI: 10.1109/ICSPC51351.2021.9451709.

[18] N. Bala, R. Elangovan, E. R., K. Kotecha, A. Abraham, L. Gabralla, and S. V, “Development of an end-to-end deep learning framework for sign language recognition, translation, and video generation,” *IEEE Access*, vol. PP, pp. 1–1, Jan. 2022. DOI: 10.1109/ACCESS.2022.3210543.

[19] D. Kothadiya, C. Bhatt, K. Sapariya, K. Patel, A. Gil, and J. Corchado Rodríguez, “Deepsign: Sign language detection and recognition using deep learning,” *Electronics*, vol. 11, p. 1780, Jun. 2022. DOI: 10.3390/electronics11111780.

[20] D. Younis, ““dawwie” in sign language,” 2022. [Online]. Available: https://www.unicef.org/egypt/stories/dawwie-sign-language#:~:text=Not%5C%20all%5C%20words%5C%20are%5C%20expressed,World%5C%20Federation%5C%20of%5C%20the%5C%20Deaf..

[21] M. R. K, H. Kaur, S. K. Bedi, and M. A. Lekhana, “Image-based indian sign language recognition: A practical review using deep neural networks,” 2023. arXiv: 2304.14710 [cs.CV].

[22] S. Li, S. Banerjee, N. K. Banerjee, and S. Dey, “Simultaneous prediction of hand gestures, handedness, and hand keypoints using thermal images,” 2023. arXiv: 2303.01547 [cs.CV].

[23] R. Sayed, H. Azmi, H. Shawkey, A. Khalil, and M. Refky, “A systematic literature review on binary neural networks,” *IEEE Access*, vol. PP, pp. 1–1, Jan. 2023. DOI: 10.1109/ACCESS.2023.3258360.

[24] “Asl alphabet,” DOI: 10.34740/KAGGLE/DSV/29550. [Online]. Available: https://www.kaggle.com/dsv/29550.