

Two Dimensional Convolutional Neural Network CNN Approach For Detection of Bangla Sign Language

by

Pollock Nag

18101114

Tamim Mahmud Khan

16101116

Shaikh Mehedi Hasan Biplob

18201087

Rachayita Barmon

18201016

MD. Minhaj Rahman

18301072

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 and Engineering

Department of Computer Science and Engineering
BRAC University
September 2022

© 2022. BRAC University
All rights reserved.

Declaration

It is hereby declared that

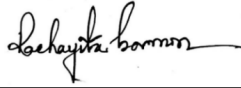
1. The thesis submitted is our own original work while completing degree at BRAC University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material 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:



Pollock Nag

18101114



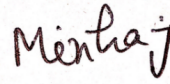
Rachayita Barmon

18201016



Tamim Mahmud Khan

16101116



MD. Minhaj Rahman

18301072



Shaikh Mehedi Hasan Biplob

18201087

Approval

The thesis titled “Two Dimensional Convolutional Network CNN Approach For Detection of Bangla Sign Language” submitted by

1. Pollock Nag (18101114)
2. Tamim Mahmud Khan (16101116)
3. Shaikh Mehedi Hasan Biplob (18201087)
4. Rachayita Barmon (18201016)
5. MD. Minhaj Rahman (18301072)

Of Summer, 2022 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science and Engineering on September 22, 2022.

Examining Committee:

Supervisor:
(Member)



Dewan Ziaul Karim

Lecturer
Department of Computer Science and Engineering
Brac University.

Co-supervisor:
(Member)



Mr. Rafeed Rahman

Lecturer
Department of Computer Science and Engineering
Brac University.

Program Coordinator:
(Member)

Dr. Md. Golam Rabiul Alam

Associate Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chairperson)

Sadia Hamid Kazi

Chairperson
Department of Computer Science and Engineering
Brac University

Abstract

Sign language is known as the primary communication medium for deaf and mute people. But the lack of available resources and a steep learning curve deter the average person from learning it making communication with the mute and deaf difficult. This problem creates an opportune place for the application of machine learning which has given rise to our emerging field. A large number of papers with high accuracy have already been published for English, French, and other languages. But the number of papers on its application for Bangla Sign language is few. Most of the researchers use SVM, ANN or KNN as classifiers. We chose CNN because it is excellent at high accuracy image classification. In this paper we use a large dataset consisting of 30 classes with 500 images each totalling to about 15000 images of bangla sign alphabets. Previous works were done only on 10 classes. We began work on those 10 bangla alphabets and later increased the number of classes to 30. We tested the accuracy's of pre trained CNN models such as DenseNet201, VGG16, InceptionV3, Resnet50, MobileNetV2, InceptionResnet, EfficientnetB2 along with our custom CNN model and were able to achieve 97.97%, 96%, 96.22%, 56.44%, 90%, 94%, 4%, 98.3 % train accuracy and 86.43%, 88%, 88.33%, 54.50%, 60%, 53%, 4.2%, 87% validation accuracy respectively. Our custom CNN model has consistently given better training and validation accuracy than any pre-trained model with lesser layers which in turn require less computations making for a lighter and faster model while maintaining high accuracy.

Keywords: Bangla sign language, CNN, KNN, ANN, VGG16, Resnet50, InceptionV3.

Dedication

The paper is dedicated to the community of mute and deaf people who are always struggling to communicate with others. Moreover, this will work as a bridge between the normal people and deaf mute people as well as make them feel like they are also valuable of our society.

Acknowledgement

At first, With the blessings of God, we have accomplished of our thesis paper so far. Secondly, we are gratified for the genuine support and supervision of our supervisor Dewan Ziaul Karim and co-supervisor Mr. Rafeed Rahman sir, in the field of collecting the literature and data. Moreover, their motivation have made us to state of our model and the successful completion of our paper.

Finally, With the kind support and prayer of our parents We are on the edge of our graduating. It might not be possible without their support and kind help.

Table of Contents

Declaration	i
Approval	ii
Abstract	iv
Dedication	v
Acknowledgment	vi
Table of Contents	vii
List of Figures	ix
List of Tables	1
1 Introduction	2
1.1 Thoughts behind the Topic	2
1.2 Problem Statement	2
1.3 Research Motivation	3
1.4 Research Objectives	4
2 Related Work	5
2.1 Literature Review	5
3 Data Collection and Preprocessing	7
3.1 Working Plan	7
3.2 Data Analysis	8
3.3 Input Data	8
3.4 Data Preprocessing	9
4 Model Architecture Using Classifiers	10
4.1 CNN	10
4.2 Proposed CNN Model	13
4.3 VGG	14
4.4 ResNet50	15
4.5 Inception-v3	18
4.6 Inception-ResNet-v2	20
4.7 DenseNet	21
4.8 EfficientNet	22

4.9	MobileNetV2	23
5	Result And Analysis	25
5.1	Result and Analysis	25
6	Conclusion	37
6.1	Discussion	37
6.2	Conclusion	37
6.3	Future Plan	38
	Bibliography	39

List of Figures

1.1	Bangla Sign Hand Gestures	3
3.1	Working Plan Flowchart	7
3.2	Dataset	8
4.1	Picture representation using a pixel grid	10
4.2	Architecture of a CNN	11
4.3	: Pooling Operation (Source: O'Reilly Media)	12
4.4	Softmax Activation Function	13
4.5	VGG Neural Network Architecture	14
4.6	VGG-16 Architecture of a VGG16 model	15
4.7	ResNet50 Architecture	16
4.8	Skip Connection.	16
4.9	Identity Block.	17
4.10	Convolutional Block.	17
4.11	InceptionV3.	18
4.12	Two 3x3 Convolutions	19
4.13	Asymmetric Convolutions	19
4.14	Auxiliary classifier.	20
4.15	Grid size reduction	20
4.16	Inception ResNetv2	21
4.17	Inception ResNetv2 Process	21
4.18	DenseNet	22
4.19	EfficientNet	22
4.20	MobileNetV2	23
4.21	MobileNetV2 architecture	24
5.1	Custom CNN Accuracy	26
5.2	Custom CNN Loss	26
5.3	DenseNet201 Accuracy	27
5.4	DenseNet201 Loss	27
5.5	VGG16 Accuracy	28
5.6	VGG16 Loss	28
5.7	InceptionV3 Accuracy	29
5.8	InceptionV3 Loss	29
5.9	Resnet50 Accuracy	30
5.10	Resnet50 Loss	30
5.11	MobileNetV2 Accuracy	31
5.12	MobileNetV2 Loss	31

5.13 InceptionResnet Accuracy	32
5.14 InceptionResnet Loss	32
5.15 EfficientNet Accuracy	33
5.16 EfficientNet Loss	33
5.17 Confusion Matrix of Custom CNN model	34
5.18 Train Accuracy	35
5.19 Validation Accuracy	35

List of Tables

4.1	Custom CNN Model.	14
5.1	Accuracy For 30 Classes.	25

Chapter 1

Introduction

1.1 Thoughts behind the Topic

Language constitutes a fundamental building block of society. If people could not communicate they would not be able to coordinate and that would hamper positive growth. The ability to talk is such a commonplace concept that we forget that there are people in the world who cannot communicate vocally whether by disability or accident. Sign language stands as the primary standardized method of communication to many such individuals. Sign language is usually conveyed via hand gestures where one or both hands form a specific shape which represents one of the letters in a particular language. Bangladesh has more than 30 lakh people who are hearing impaired [6]. The Center for Disability and Development or CDD recognized Bangla Sign Language as a standard of communication for the Bangla Deaf Community in 2000 yet the facility and opportunities for learning sign language remains inadequate [6]. In recent years research has been conducted to create a machine learning model for recognizing sign language and translating into letters that the general public can understand. With the help of image recognition we can help close the impairment gap by translating sign languages into both written and spoken forms easing communication. Image recognition is performed by a CNN model trained upon a fixed dataset consisting of images of hand signs and their corresponding labels which are the Bangla alphabet.

1.2 Problem Statement

Language is the medium of communication. But in our society some people unfortunately do not have the ability of speaking and listening. To communicate with those people we need to use sign language. In Sign language we generally use various gestures like hand gestures or symbolic gestures instead of sound. As sign language is quite hard for common people to understand, many people lose their interest in learning sign language.

There is no international standard for sign language. For example, even though citizens from both countries speak English, the sign languages of the United States and the United Kingdom are not the same. Automatic recognition of sign language (ARSL) can be extremely useful in establishing quick communication between general and deaf or mute people. Furthermore, a combination of ARSL and a translator can help deaf and mute people to communicate between similar types of people even

though they are from different nations. As a result, researchers from various fields like NLP and AI are using their knowledge to build a tool that will ease communication with the deaf and the mute. [1]

A lot of research has been done on English language but the amount of researchers on Bangla sing language is very few and most of them uses SVM as a classifier to detect alphabets for different hand gestures. But SVM is a generic classifier in machine learning. It was not designed to work with image data in mind. However, nowadays CNN is frequently using for image classification . It has a higher accuracy for classifying images. However, convolutional neural networks (CNN) do not require manual feature extraction, which is a significant advantage. Using CNN as a classifier, we suggest a method in this research for recognizing Bangla sign language.

1.3 Research Motivation

In this modern world, computer vision is helping in every sector to make our life easier. In order to develop with this modern world, Deaf people frequently use sign language to communicate . For every language in the world there exists its signed counterpart. Such as, Australian Sign Language (AUSLAN),British Sign Language (BSL), etc. In our country deaf people’s language is known as Bangladesh Sign language (BDSL) which is shown in figure 1.1. Sign language is not commonly known making communication with the disabled difficult.This kind of innovation will benefit the deaf people of our country. Fundamental goal of this kind of work is to work as a digital helping device between hearing people and deaf people.

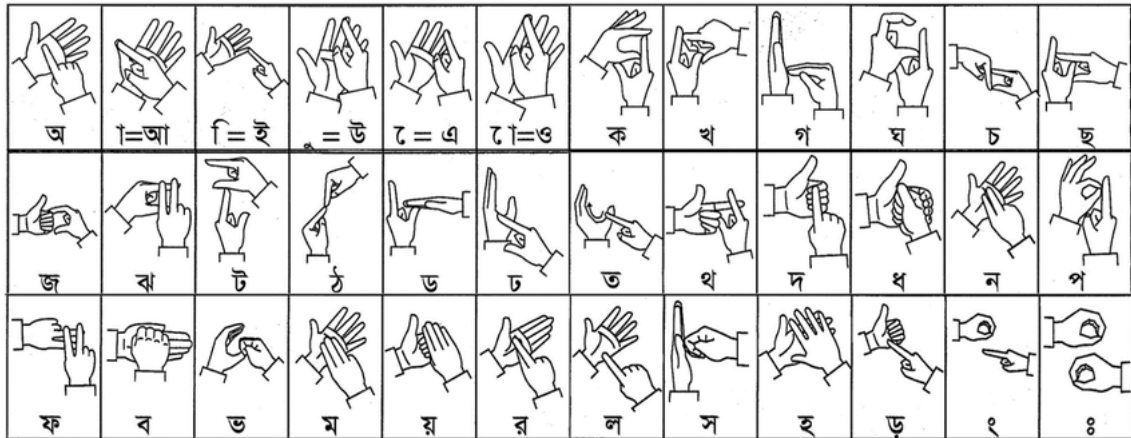


Figure 1.1: Bangla Sign Hand Gestures

1.4 Research Objectives

In order to achieve our goal, we need to do:

1. Find maximum amount of data set, with different backgrounds and lighting conditions to
2. Deeply understand the algorithm model to make the system more efficient.
3. We have to consider all the possible error while making the system.

There are some people who work on this kind of project in Virtual reality and they apply a leapmotion control to detect hand gestures. Some people use an artificial neural network to recognize signals by employing CNN. However none of them functioned in real-time. While researching CNN based object identification systems we find about faster R-CNN. This sends only the necessary regions to convolutional network .So by using these simple architectures we can accomplish our goal and make the lives of deaf people easier.

Chapter 2

Related Work

2.1 Literature Review

In a project, Sohalia Rahman, Naureen Fatema, M. Rokonzaman et al.,[1] use a dotted glove to identify the gesture of hands. The system collects the dots and matches with the pre-prepared charts. The researcher uses image processing on the input image. By this way, they identify numerical letters of BDSL. Using gloves is not an effective method.

Oishee Binte Hoque, Mohammad Imrul jubair, Md. Saiful Islam, Al-Farabi Akash, Alvin sachine Paulson et al., [2] proposed a paper using faster (R-CNN) which generally makes maps and a network regional proposal network (RPN). This method processes the input picture with a high possibility of containing the desired hand gesture. After this stage, ROI pooling reduces the maps into the identical shape. After dividing the feature map input into a set numeral of roughly equivalent areas it applies max-pooling into each zone. By this way, they said the accuracy of their project

Md. Sanzidul Islam, Sadia Sultana Sharmin Mousumi, Nazmul A. Jessan, AKM Shahariar Azad Rabby, Sayed Akhter Hossain brought in their paper named “Ishara-Lipi: The First Complete Multipurpose Open Access Dataset of Isolated Characters for Bangla Sign Language” [3] that by using ADAM optimizer they got a rate 0.001. Their model of CNN contains 9-layers. For their sign character database, they kept the data for testing 15% and for training 85%. They assert that after 50 epochs, their accuracy on training set and validation was 92.65% and 94.74%.

Md. Islam et al., [4] suggested a paper using Convolutional Neural Network (CNN) to recognize BdSL digits. Firstly they converted the images in jpg format in 128 by 128 pixels for the dataset. Then the dataset was resized by 28 into 28 pixels converted into gray level pixels. Then converted into binary colored pictures given the labels. The model achieved 95.5% training accuracy 94.88% validation accuracy.

Shirin Sultana Shanta, Saif Taifur Anwar et al.,[5] proposed a paper using Convolutional Neural Network (CNN) and SIFT to recognize BdSL. They implement skin masking technique to crop only region of interest(ROI). Then extract feature descriptor using SIFT (Scale Invariant Feature Transform), Use k-means clustering

to obtain features as clustered descriptor, use bag of features to represent the features in histogram of visual vocabulary , Input the data in CNN as histogram and check output accuracy is 98.20%.

Md. Islam et al.[6] suggested a paper using CNN in order to recognize Bangla Sign Language. Firstly, they convert the image into a gray-scale image then they normalize those images. For normalizing the images they divide gray pixels by max gray level value which is 255. Then, images were reshaped in 64 by 64 pixels for exploration. Finally, they input these images in CNN algorithm where the number of convolution, pooling and fully connected layers are six, three and two (one for input and one for output) accordingly. For basic characters, numerals, and their combined use, they were able to attain accuracy of 99.83 percent, 100 percent, and 99.80 percent, respectively.

Consisting of 24168 samples, the authors Hossain et al., [7] state in their proposed model that they got the highest accuracy from digit detection by putting some extra layers such as convolution layers selected number, max pooling, dropout etc. they also claimed that with 30FPS, their model can give better performance. Moreover, by adding extra layers they solved their detection problem of having a kind of similar input.

To recognize Bangla sign language Lutfun nahar, Nanziba Basnin, and MD Shaha-dat Hossain et al.,[8] makes use of CNN with LSTM. The background subtraction creates a foreground mask. Gray-scale conversion ensures that only a single channel is used, speeding up the learning process. Morphological erosion removes the noise. The image is then run through a median filter and resized. This image is sent to the CNN resulting in a testing accuracy of 88.5%.

F. M. Javed Mehedi Shamrat et al., [9] proposed a paper using Convolutional Neural Network (CNN) and SIFT to recognize BdSL. The system applies transformation on image then applies logarithmic replace technique to control extra light. Every pixel measurement is replaced with the logarithmic grade. Then the LBP is applied to the image.

The paper named Bangla sign Language Recognition using hand Gestures: A Deep Learning Approach by T.B. Das and M.J. Islam et al., [10] proposes a CNN variant -1 with 6 layers. The images are taken with a webcam where the background is segmented through flipping, gray-scaling and blurring it without the hand first. Then the hand is introduced which is gray-scale, blurred and using a threshold is separated from the image with bit-wise AND operation. This results in high accuracy from the model.

Chapter 3

Data Collection and Preprocessing

3.1 Working Plan

In total there are 36 characters available in Bangla sign language. So, we collected all 36 letters and rearrange them properly. To avoid the chance of desultory, the data sets will be categorized in different folders. The folders will be labeled according to the numeric naming convention. For making the datasets usable with machine learning models, we resized the height and width of the images and converted them to gray-scale. We resized them by 128 * 128 pixels. Then, we split the data for training and testing purposes. We trained our dataset with multiple layers of CNN to try and produce the most accurate results.

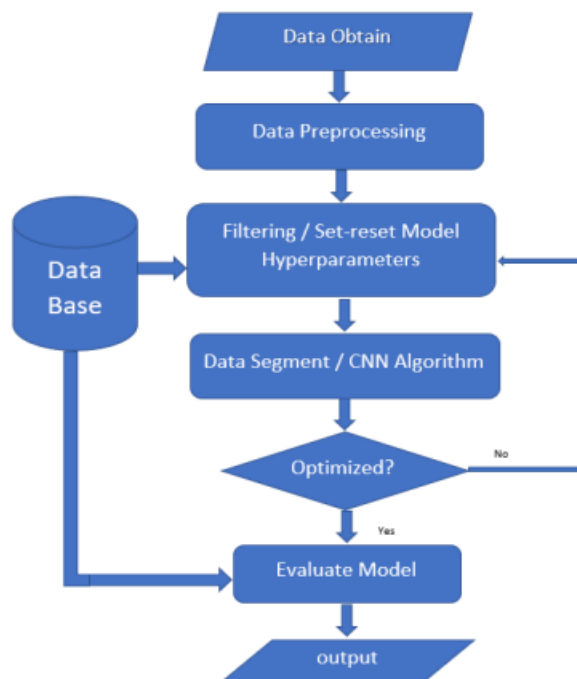


Figure 3.1: Working Plan Flowchart

3.2 Data Analysis

Our research topic has plenty of work done already. So acquiring multiple datasets was easy. We collected multiple datasets from the internet with different backgrounds and lighting effects. As, making our own dataset is time consuming we decided to go with pre-existing ones. How we collected our input data and how we pre-processed it are briefly explained below.

3.3 Input Data

There are two types of bangla sign languages available in our country. They can be one-handed and two-handed however one-handed sign language is used frequently. So we decided to work on one-hand bangla sign language.



Figure 3.2: Dataset

There are a total of 30 classes for one-hand representation of Bangla Sign Language. In 30 classes there are total approximately 15000 which is split into 80 percent training and 20 percent test purpose. All images are in RGB which is illustrated in figure 3. Some images are collected from a project paper paper: "Real Time Bangladeshi Sign Language Detection using Faster R-CNN" by authors Oishee Bintey Hoque and Mohammad Imrul Jubair and Md. Siful islam and Al Farabi Akash and some images are collected from Kaggle and <https://www.kaggle.com/datasets/kanchonkantipodder/bdsl1500> and the paper author is Kanchon Kanti Podder.

3.4 Data Preprocessing

We make use of the python's augmenter library to augment our data.

A) All the input images are resized from their original sizes to a matrix of size 224 by 224. The input dataset contains images with a variety of sizes which cannot be passed through our model due to size mismatch. Resizing allows us to control the size of each layer of our model while allowing the flexibility of having a dataset consist of multiple image sizes. A size of 224 by 224 is a good middle ground between the processing speed and model accuracy.

B) The resulting images are then flattened. This reduces the number of channels that need to be processed from 3 to 1 resulting in a decrease in time required for each step from 24s to 8s each and the total time for each epoch from 600s to 200s roughly which is a 3x increase in processing speed from running the same data in rgb.

Chapter 4

Model Architecture Using Classifiers

4.1 CNN

CNN: Data that can be split into a grid-like pattern, such as a digital image, is evaluated using convolutional neural networks (CNNs). A digital picture may be conceptualized as a large grid of binary values that each represent one primary hue or a mix of the three. Pixel values specify the brightness and color of each individual pixel.

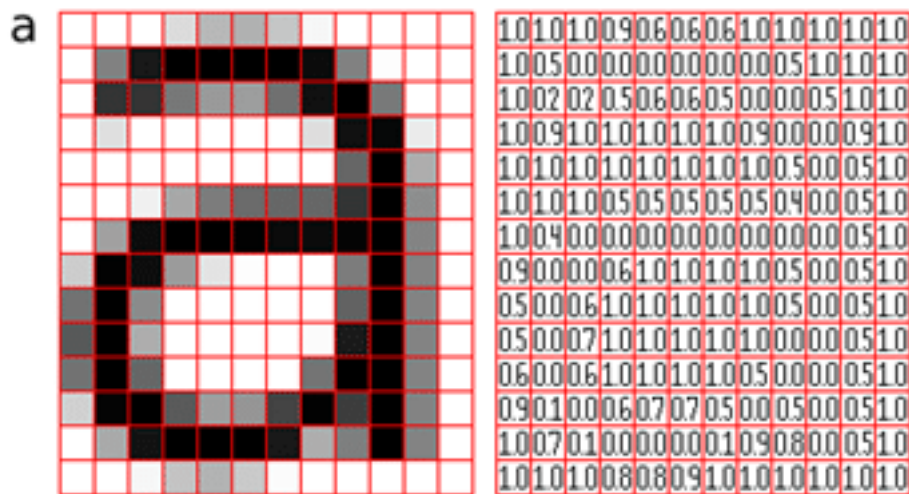


Figure 4.1: Picture representation using a pixel grid

Three layers of CNN are convolutional, pooling, and fully connected.

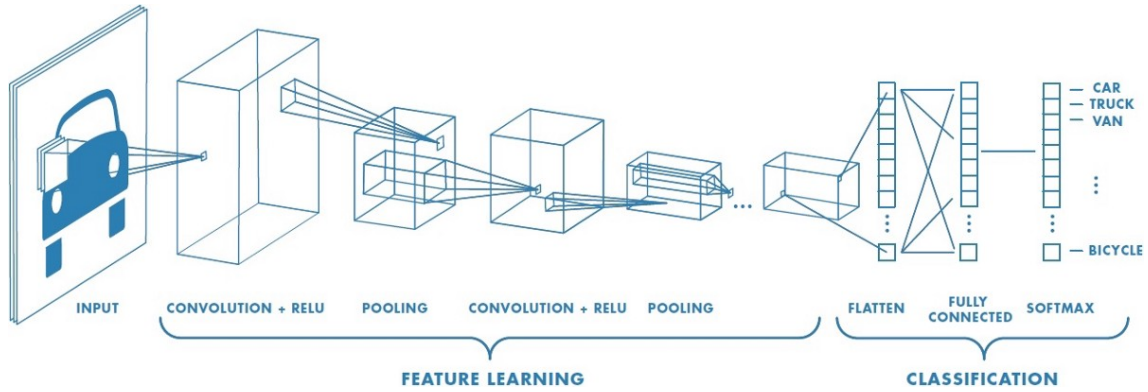


Figure 4.2: Architecture of a CNN

Convolution Layer: The main factor of CNN is the convolution layer. The network's computational capacity is primarily its responsibility. This layer includes two matrices: the limited portion of the receptive field and a group of trainable parameters called kernel, and another part is the confined the receptive field's portion. While the kernel is smaller than an image, it is deeper. This implies that the kernel's height and breadth are modest if the picture has three (RGB) channels but the depth is large. The kernel shifts the height and width of the image throughout a forward run, providing a visual description of the accessible area. At each spatial place in the image, a two-dimensional representation of the kernel's response generates an activation map. A stride refers to the kernel's sliding size. Assuming, our input size is $W \times W \times D$ and,

F = with a spatial dimension a D_{out} number of kernels,

S = stride

P = padding amount

Now, to calculate the size of the output volume the formula is:

$$W_{out} = \frac{W - F + 2P}{S} + 1 \quad (4.1)$$

Pooling Layer: At some locations, the output of the network is replaced with a summary of nearby outputs by the pooling layer. This lessens the size and hence the quantity of calculations needed. Each representational slice is handled independently throughout the pooling process. Examples of pooling functions include the rectangular neighborhood L2 norm, rectangular neighborhood average, and a sample mean calculated based on the distance to the central pixel. Max pooling, the most popular technique, gives the maximum output for the neighborhood.

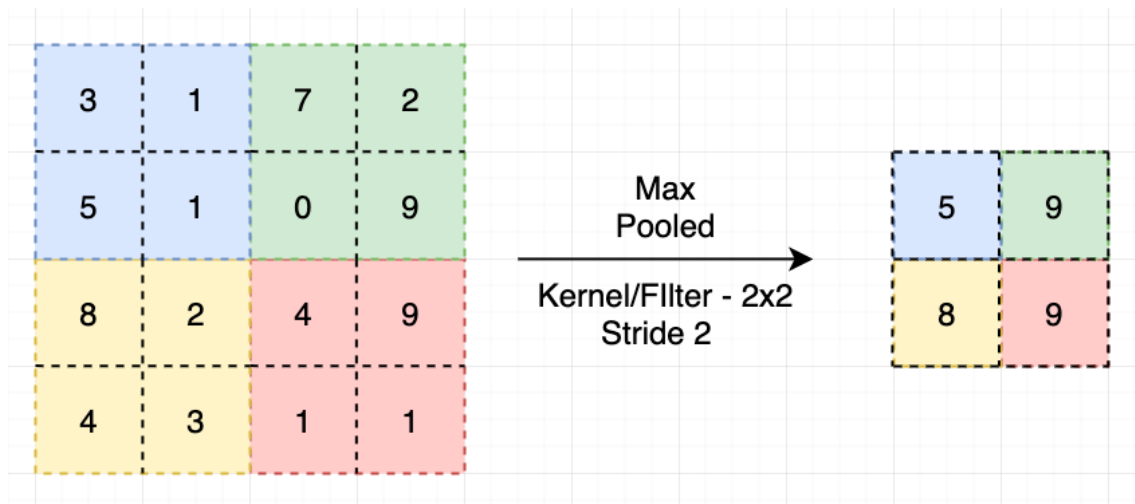


Figure 4.3: : Pooling Operation (Source: O'Reilly Media)

The following formula can be used to compute the output volume:

$$W_{out} = W - FS + 1 \quad (4.2)$$

Here, The activation map's dimensions in this case are $W \times W \times D$, where F is a spatially-sized pooling kernel and S is stride. Pooling provides some uniform translation in all cases, for instance an object may be identified no matter in which it occurs on the panel.

Fully Connected Layer: As suggested by the name, every neuron in this layer as well as the layer below it is linked. This makes it possible to compute it using bias effect and matrix multiplication.

The completely linked layer is constantly able to accommodate the input and output mapping.

Activation Function:

- i. Sigmoid: Mathematical version of sigmoid nonlinearity is $\text{sigmoid}(x) = 1 / (1 + e^{-x})$. A real-valued number is "squashed" into a value between 0 and 1.
- ii. ReLU: In any CNN architecture, activation functions play a critical role in determining which node should be fired. The function ReLU can be represented mathematically as $\text{ReLU}(x) = \max(0, x)$ (1), where x is the input to a neuron
- iii. Softmax: The Softmax Activation Function is a fascinating activation function that takes real-number vectors as input and normalizes them into a probability distribution proportional to the exponential of the numbers.

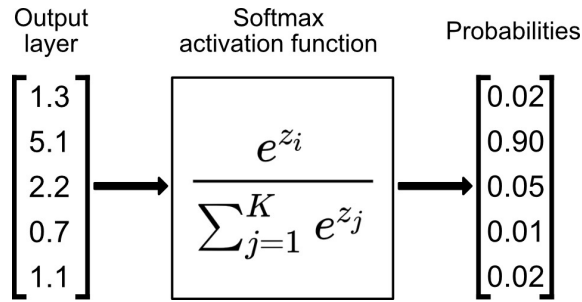


Figure 4.4: Softmax Activation Function

Output Layer: In CNN output layer is a fully connected layer that flattens and sends the input from other in order to change the output into the number of classes requested by the network.

4.2 Proposed CNN Model

Configuration 1: Dropout and Batch Normalization: Dropout layers were a common recommendation for reducing over fitting. So we decided to try creating a model with drop out. Batch normalization would ensure our values kept within a certain range. In this model we use 5 layers. (2 Convolution layers with max pooling, 3 dense layers). All layers include batch normalization with axis= -1.All layers include a Dropout layer of 0.2.

Configuration 2: Simpler CNN with 12 layers: It was apparent from testing the pre-trained models that we would require a simple model for our dataset. So one of the CNN's we tried was one with 12 very simple layers. In this model we have 12 layers which are 2 Convolution layers, 2 max-polling layers, 4 batch-normalization layers, 1 flatten layer and 3 dense layer. However, we have tried few others configurations as well. Among them configuration 2 gives us better accuracy. So the model summary of configuration 2 is attaching below:

Layer (type)	Output Shape	Param
conv2d ₄ (Conv2D)	(None, 222, 222, 32)	896
max_pooling2d ₄ (MaxPooling2D)	(None, 111, 111, 32)	0
batch_normalization ₈ (BatchNormalization)	(None, 111, 111, 32)	128
conv2d ₅ (Conv2D)	(None, 109, 109, 32)	9248
max_pooling2d ₅ (MaxPooling2D)	(None, 54, 54, 32)	0
batch_normalization ₉ (BatchNormalization)	(None, 54, 54, 32)	128
flatten ₂ (Flatten)	(None, 93312)	0
dense ₆ (Dense)	(None, 512)	47776256
batch_normalization ₁₀ (BatchNormalization)	(None, 512)	2048
dense ₈ (Dense)	(None, 256)	131328
batch_normalization ₁₁ (BatchNormalization)	(None, 54, 54, 256)	1024
dense ₁₀ (Dense)	(None, 30)	7710

Table 4.1: Custom CNN Model.

4.3 VGG

VGG : VGG architecture is widely used for Object recognition now a days. Other than imageNet, VGGNet outperforms baseline on a range of tasks and datasets. It is one of the most commonly utilized image recognition architectures available constructed from small convolutional filters. In total it VGG-16 has thirteen convolution layers and three fully linked layers.

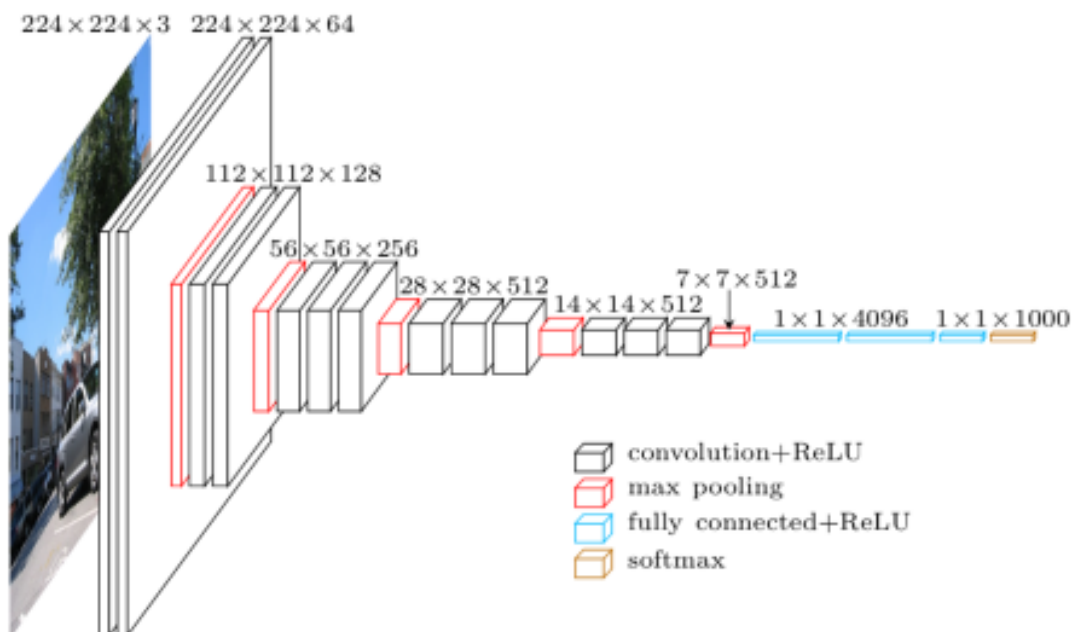


Figure 4.5: VGG Neural Network Architecture

Details of VGG layer is mentioning below:

Input Layer: Images having a resolution of 224×224 pixels are accepted by the VGGNet.

Convolutional Layers: A tiny receptive field (3×3) to record up or down and left or right movements is used by the convolutional layers of VGG. In addition, 11 convolution filters produce a linear function of the input. The ReLU unit, a significant AlexNet creation significantly diminishes training time. The stride is set at 1 pixel to support the spatial resolution of subsequent convolution.

Hidden Layers: The VGG network uses ReLU in every hidden layer.

Fully-Connected Layers: Three completely connected layers make up the VGGNet. Two layers in the beginning have 4096 channels each, whereas the third layer has 1000 channels, one per class.

VGG16: VGG16 has 16-layer deep neural network (VGGNet). with 138 million parameters. The network's appeal comes from its simplicity. It has a uniform architecture. We can use around 64 alternatives to the number of filters, where we can extend to approximately 128 and then to 256. We can utilize 512 filters in the last phases.

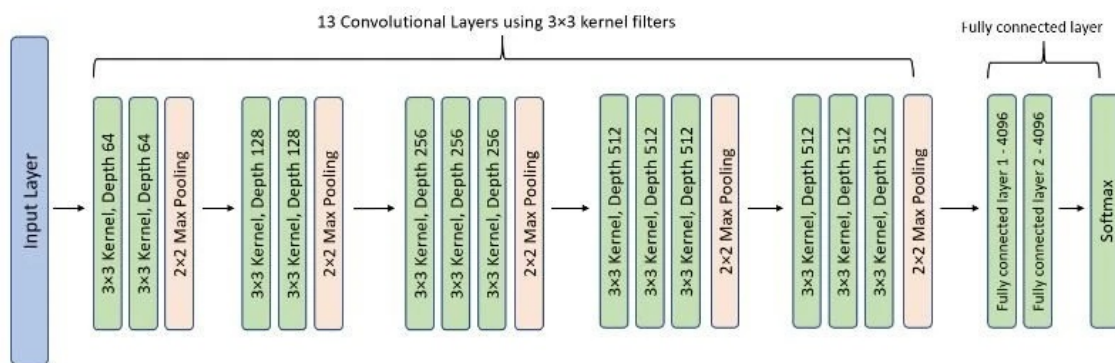


Figure 4.6: VGG-16 Architecture of a VGG16 model

4.4 ResNet50

ResNet50: Resnet50 is a 50-layer deep convolutional neural network (CNN) consisting of 48 layers of convolution, 1 MaxPool and 1 Average Pool layer. A residual Neural Network(ResNet) is a sort of Artificial Neural Network (ANN) which constructs a network by pilling up residual blocks on top of one another. In this method, a network is pre-trained on over a million photographs and is stored in the imageNet database. A $224 \times 224 \times 3$ image is used as the input, followed by a MaxPooling layer with a 3×3 filter. Then there are 16 leftover blocks. Each residual block is made up of three convolutional layers that change the data's dimensionality. The leftover blocks

are divided into four groups. The remaining blocks in a group have comparable layers. In the first group, for example, we have three residual blocks, each of which has three convolutional layers that perform the following convolutions. ResNet50 helps to train ultra-deep neural networks which means it can contain hundreds or thousands of layers but it still performs well. In short, Resnet is the most well-known neural network that is used to solve a variety of computer vision problems as it can classify pictures into over a thousand different items types which includes monitor, book, pen and a variety of animals.

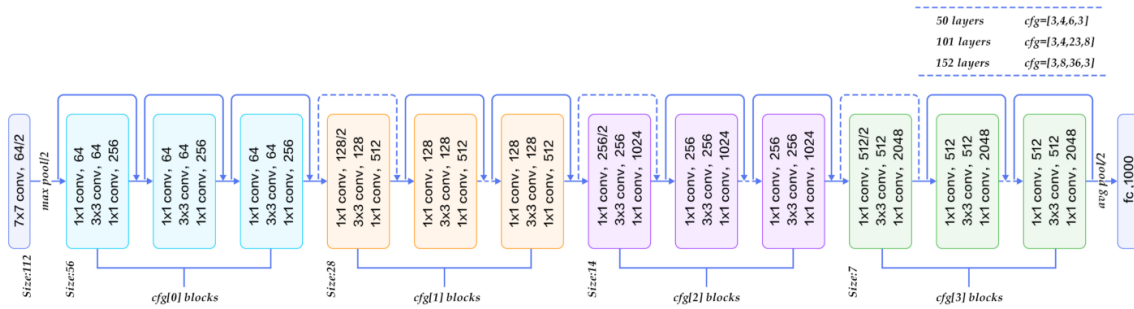


Figure 4.7: ResNet50 Architecture

The framework can also be used to improve accuracy in non-computer vision activities. Backpropagation reduces the gradient which in turn slows the network’s learning rate. This is called the ‘Vanishing Gradient Problem’ and is one of the key drawbacks of CNNs. ResNet overcomes this problem with ”SKIP CONNECTION.” Skip connection concatenates the original input to the output of the convolution block. When we multiply numerous numbers less than or greater than 1 together, the output shrinks or grows exponentially with each term added to the multiplication yet we need to conduct more multiplications the more layers we have. This has been empirically demonstrated that when using typical architectures, the performance of very deep neural networks degrades and accuracy saturates as the network converges. To eliminate this kind of problem, skip connection is necessary.

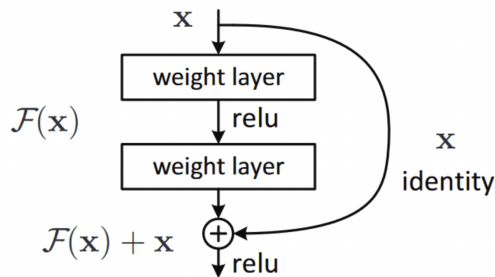


Figure 4.8: Skip Connection.

A skip connection is a straight connection that bypasses some of the model’s tiers. As a consequence of the skip connection, it results in various kinds of output. Without the skip connection, input ‘X is multiplied by the layer’s weights, then a bias

term is added.

Although all algorithms are trained on the output 'Y,' ResNet is trained on $F(X)$. To put it another way, ResNet attempts to make $F(X) = 0$ so that $Y = X$.

The activation function, $F(x)$, produces the following output:

$$F(w * x + b) = F(X); \tag{4.3}$$

The output of the skip connection technique, on the other hand, is:

$$F(X) + x; \tag{4.4}$$

There are two kinds of input blocks in ResNet50. One is identity block and another is convolutional block. The value of 'x' is added to the output if the,

$$Inputsize == Outputsize; \tag{4.5}$$

However, a convolutional block is attached to the shortest way to make the size of the input equal to the output if the input size is not equivalent to the output size.

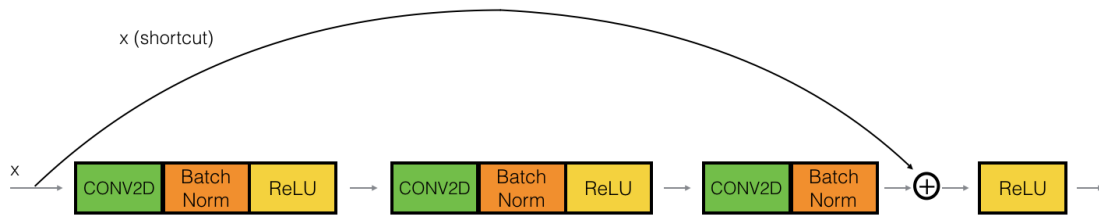


Figure 4.9: Identity Block.

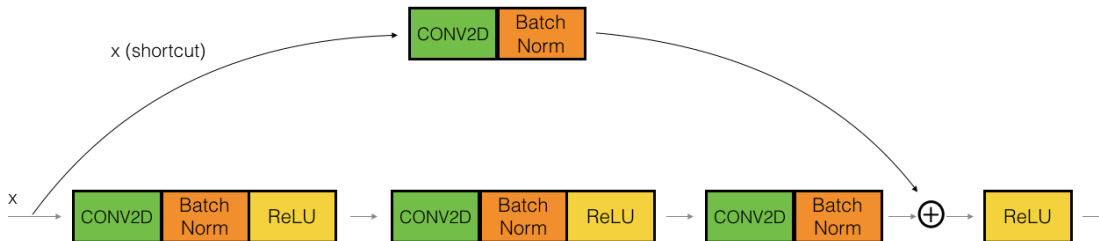


Figure 4.10: Convolutional Block.

To make the input size equal to the output size, there are two methods,
i) Padding the input volume,

$$\lceil (n + 2p - f) / s + 1 \rceil^2; \tag{4.6}$$

Where,

n = input size of image,

p = padding,

f= number of filters

s= stride.

ii) Performing 1x1 convolutions,

$$\lceil (n/2) * (n/2) \rceil; \tag{4.7}$$

Where,

n = input size of image.

4.5 Inception-v3

Inception-v3: Inception-v3 is design from the inception family that performs factorized 7*7 convolutions, label smoothing, and includes an additional classifier to send label information farther down the network. InceptionV3 model is the outcome of several concepts that different scholars have refined over time. The model has a variety of elements, including symmetric and asymmetric buildings, max pooling, concatenations, dropouts, and completely connected layers. This model heavily relies on batch normalization, and Softmax is utilized to compute loss.

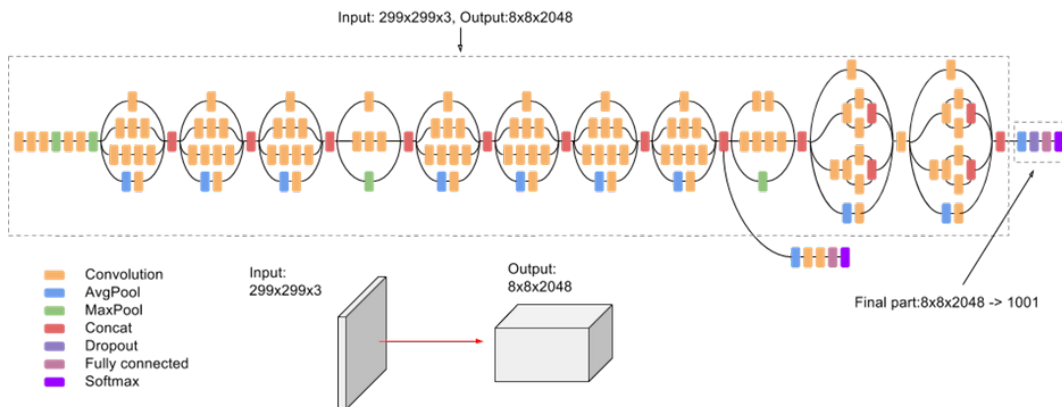


Figure 4.11: InceptionV3.

Inception v3 network built in five ways.

i) Factorized Convolutions: As the number of frameworks in a network is reduced, it plays an important role to upgrade the computational productivity. Furthermore, the factorized convolutional monitors the network's efficiency.

ii) Smaller Convolutions: It is used for replacing bigger convolutions with smaller. Because of this reason, it leads to faster training. For instance, to reduce the computational cost 5×5 layer is replaced by two 3×3 convolutional layers.

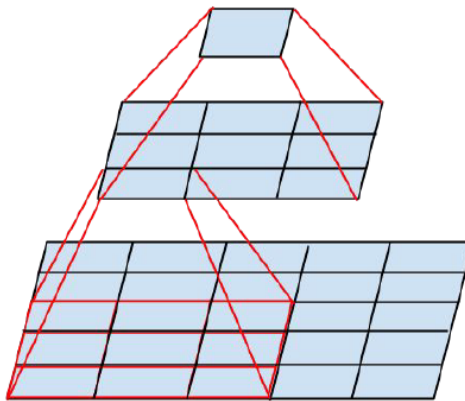


Figure 4.12: Two 3×3 Convolutions

iii) Asymmetric Convolutions: Beside a bigger convolutions are factored into small pieces, the asymmetric convolution is the best option for making the model more productive. For instance, we can replace a 3×3 convolution by a 1×3 convolution after another 3×1 convolution.

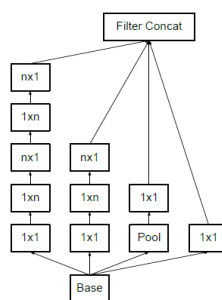


Figure 4.13: Asymmetric Convolutions

iv) Auxiliary Classifier: Inception v3 aims to use an auxiliary classifier so that it can enhance the convergence of very deep neural networks. As an additional classifier during training, a little CNN is inserted between layers, and the loss it suffers is contributed to the loss of the primary network. Prior to Inception v3, auxiliary classifiers used deeper networks in Google Net to their advantage; as a result, Inception v3 uses auxiliary classifiers as regularizers.

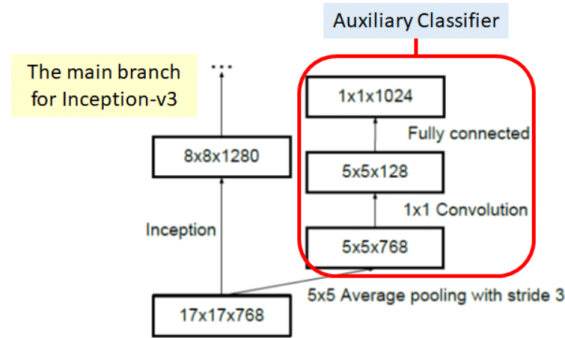


Figure 4.14: Auxiliary classifier.

v) Reduced grid size: Basically, the reduction of grid size is done by max pooling and average pooling. Moreover, a more efficient strategy is presented to tackle computational cost bottlenecks. For instance, a $d \times d$ grid with n filters after reduction it sums up in $d/2 \times d/2$ with $2n$ filters.

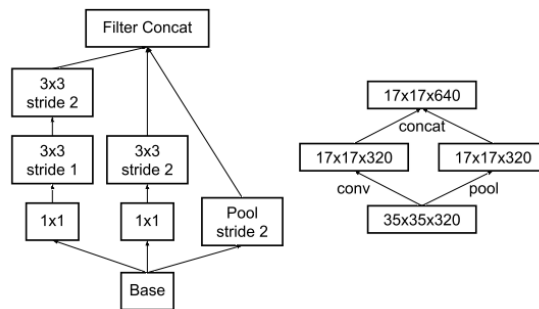


Figure 4.15: Grid size reduction

4.6 Inception-ResNet-v2

Inception-ResNet-v2: ResNet and Inception provide the best performance in image classification in relation to the computational cost necessary. Inception-ResNet as the name implies combines the Inception and Residual architectures to obtain the best of both worlds. The network consists of 164 layers and is capable of classifying images into 1000 different categories from keyboard and mice to many animals.

The essential feature of the Inception-ResNet is output of inception module is attached to the input from the preceding layer. This is accomplished by utilizing three separate stem modules and reduction blocks. In order to maintain this functionality factorization is utilised to match the input and output dimensions from the previous layer and the inception module respectively.

However, more research revealed that when there are more than 1000 convolution filters, the network fails. The problem of the dying network is addressed by activation

Inception Resnet V2 Network

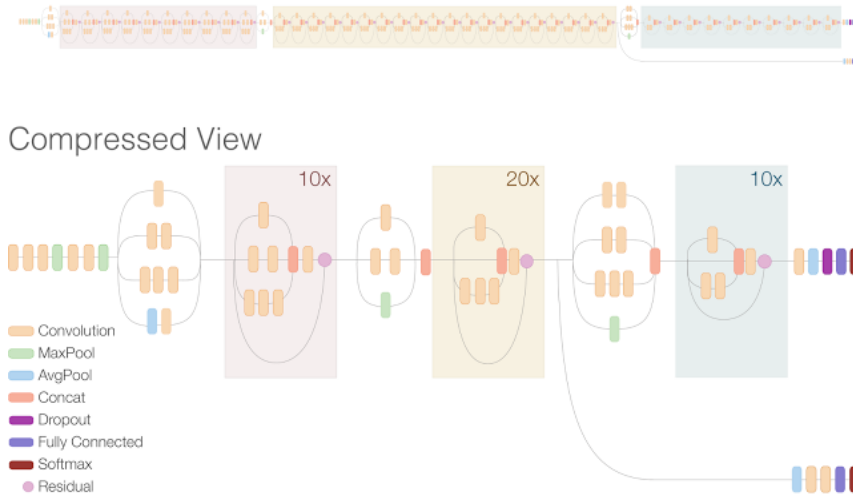


Figure 4.16: Inception ResNetv2

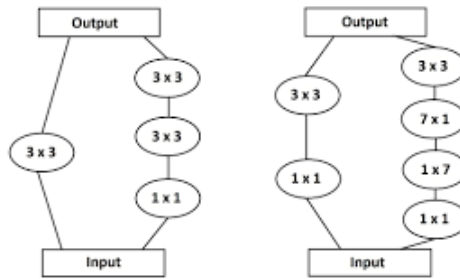


Figure 4.17: Inception ResNetv2 Process

scaling, which is then introduced.

4.7 DenseNet

DenseNet: By concatenating the output feature maps of the layer with the input feature maps rather than calculating their total, DenseNets streamline the connection pattern between layers introduced in previous designs. Due to the lack of duplicate feature maps, DenseNets may operate with fewer parameters than a comparable classical CNN.

$$x_l = H_l([x_0, x_1, \dots, x_{l-1}])$$

DenseBlocks and Transition Layers are the two components of DenseNets. The feature maps dimensions are consistent inside each Denseblock. Between them are the Transition Layers, which provide batch normalization, 1x1 convolution, and 2x2 pooling layers to handle downsampling. This channel dimension is becoming bigger at every layer since we are concatenating feature maps. We can generalize for the l -th layer if we set up H_l to generate k feature maps each time:

$$k_l = k_0 + k * (l - 1)$$

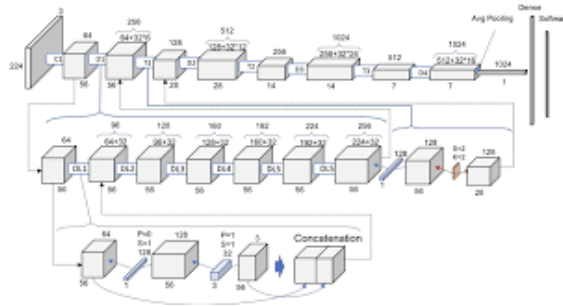


Figure 4.18: DenseNet

The growth rate is this hyperparameter k . The amount of information added to the network at each tier is controlled by the growth rate.

Every layer has access to the layer above it and, as a result, to all of the information. Then, as concrete k feature maps of information, each layer is contributing fresh information to this overall collection.

4.8 EfficientNet

EfficientNet: Based on the baseline network created by the neural architecture search utilizing the AutoML MNAS framework, EfficientNet was created. The network is optimized to achieve maximum accuracy, but when the network takes a long time to produce predictions, it is punished for both being computationally demanding and having a sluggish inference time. Due to the increase in FLOPS, the architecture employs a mobile inverted bottleneck convolution that is bigger than MobileNetV2.

EfficientNet uses a technique called compound coefficient to scale up models in a simple but effective manner. Compound scaling uniformly scales each dimension with a certain fixed set of scaling coefficients. The baseline model is compound scaled up to obtain the family of EfficientNet

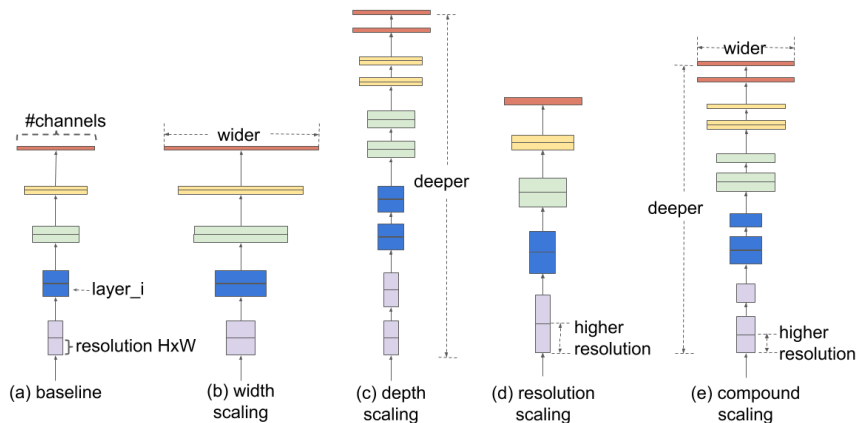


Figure 4.19: EfficientNet

4.9 MobileNetV2

MobileNetV2: MobileNetV2 is CNN group of general-purpose computer vision neural networks built for mobile devices to provide categorization, detection, and other functions. The capability to operate deep networks on personal mobile devices increases user experience by providing anytime, everywhere access, as well as extra security, privacy, and energy conservation benefits. As new applications arise allowing users to interact with the actual world in real time, so does the demand for ever more efficient neural networks. Here, the MobileNetV2 meets the need of the requirement of efficient neural network.

MobileNetV2 extends the concepts of MobileNetV1, as a effective building blocks depth-wise separable convolution. MobileNetV2 has two new architectural features. One is linear bottlenecks between layers and another one is shortcut connections between bottlenecks.

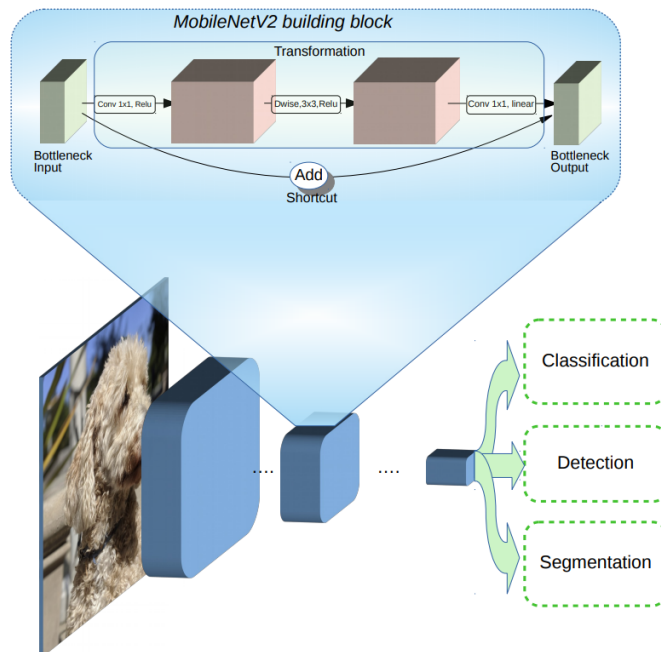


Figure 4.20: MobileNetV2

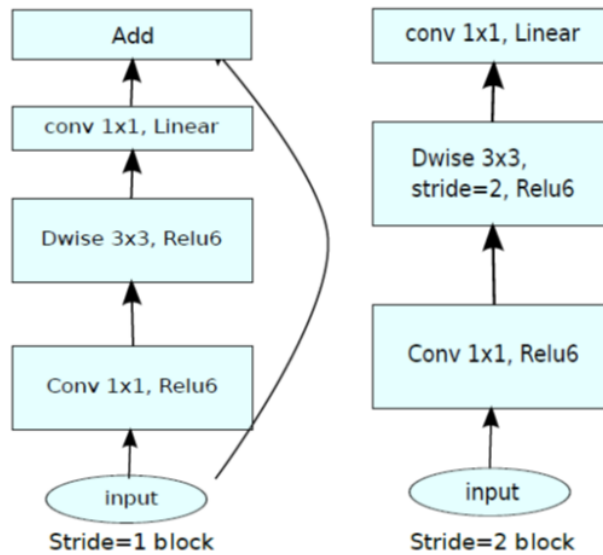


Figure 4.21: MobileNetV2 architecture

MobileNetV2 enhances the performance of mobile models on a variety of workloads and benchmarks, as well as across a range of model sizes. The base of this model architecture is it is a reversed residual form where it goes outside of the tradition and makes the residual block as thin bottleneck of input and output. In order to retain representational strength, non-linearity's in the thin layers must be removed. For this design, it enhances performance and provides the inspiration. From transformation's vent, it allows decoupling of the input or output domains. Moreover, MobileNetV2 is providing a useful foundation for further investigation.

Chapter 5

Result And Analysis

5.1 Result and Analysis

We achieved various accuracy metrics from various models to verify the outcome analysis after the train and test for all the models for 30 classes. We achieved the highest prediction accuracy and validation accuracy for each model for these 30 classes. The following table lists top train accuracy and validation accuracy:

Models Top	Training Accuracy	Validation Accuracy
Custom CNN	98.3 %	87%
Dansenet201	97.97%	86.43%
VGG16	96%	88%
InceptionV3	96.22%	88.33%
Resnet50	56.44%	54.50%
MobileNetV2	90%	60%
InceptionResnet	94%	53%
EfficientnetB2	4%	4.2%

Table 5.1: Accuracy For 30 Classes.

In Table 5.1 we can see that test accuracy, which measures how accurately the models can categorize Bangla Sign Language (30 classes), and validation accuracy, which measures how accurately our trained model predicts on the testing dataset.

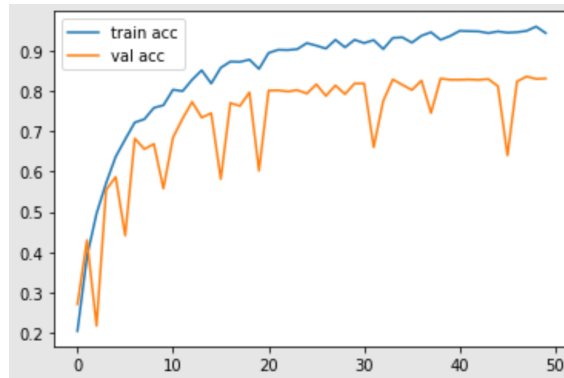


Figure 5.1: Custom CNN Accuracy

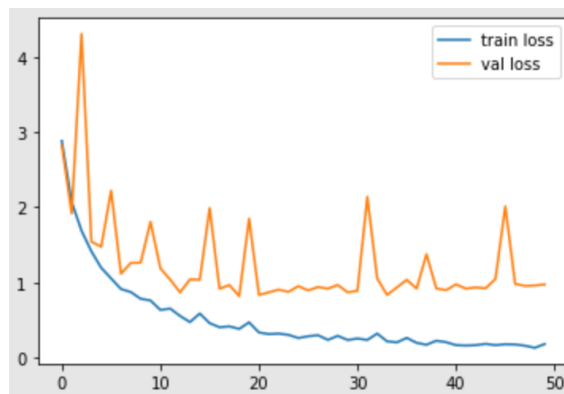


Figure 5.2: Custom CNN Loss

In figure 5.1 and 5.2 accuracy for custom CNN model shoots up quite rapidly crossing 80% by epoch 10. From 10 onwards it very slowly climbs to 98.3% over the remaining 50 epoch. Accuracy in training. Loss follows a similar trend in the opposite direction. It decreases to 0.5 in the first 10 epochs and slowly trickles to around 0.1 by epoch 50 in training.

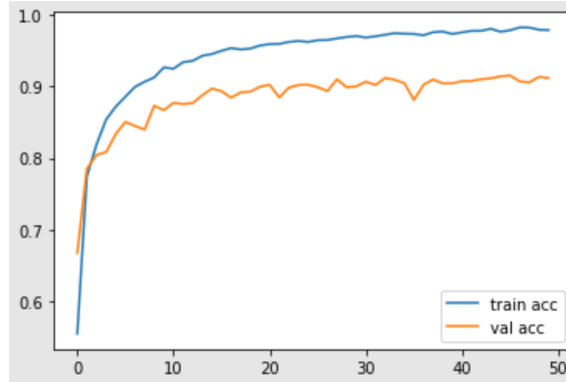


Figure 5.3: DenseNet201 Accuracy

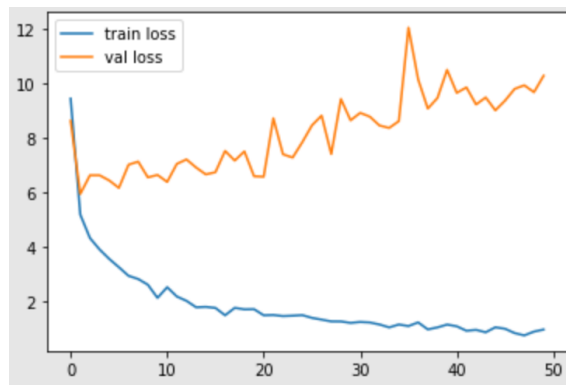


Figure 5.4: DenseNet201 Loss

In figure 5.3 and 5.4 DenseNet model starts off with relatively low loss on both training and validation but as the epochs progress the training loss progressively decreases while the inverse is true for validation. Validation loss is also not a gradual increase. Accuracy for this model starts around 55% and quickly increases to a maxima of 90% within the first 10 epochs. What follows is a gradual but slower progression to a maximum of 97.97% at epoch 50.

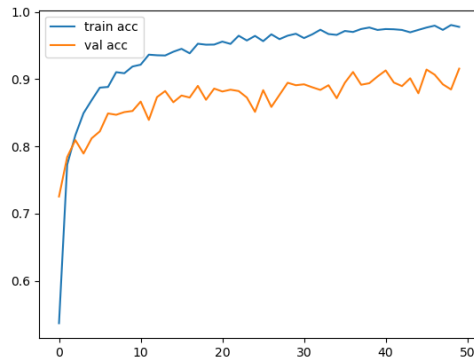


Figure 5.5: VGG16 Accuracy

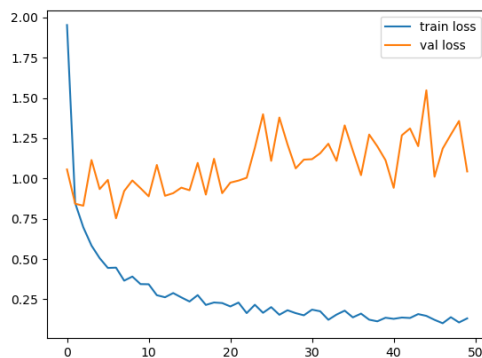


Figure 5.6: VGG16 Loss

In figure 5.5 and 5.6 Vgg16 starts off with poor accuracy but within the first 5 epochs accuracy shoots up to 80% before steadily increasing with time until about 10 epochs have finished. At which the accuracy starts to plateau as it crosses 90%. From 10 epochs onwards the accuracy shifts between 93 - 96% over the course of the next 20 epochs to epoch number 30. As the model approaches epoch 40 the accuracy gradually stabilises and holds position near 96% till it reaches epoch 50. Beyond epoch 50 overfitting occurs. Validation accuracy however is far more unstable across the entire dataset. We used an 80/20 split between the training and validation data. Validation accuracy in contrast begins at 72% gradually increasing to 80 after which it increases in an unstable manner to a maximum of 88% till epoch 50. Loss during training starts of at around a value of 2. Similar to the accuracy over the course of the next 5 epochs it drastically decreases to a value of 0.375. From epoch 5 to 50 loss decreases to 0.125.

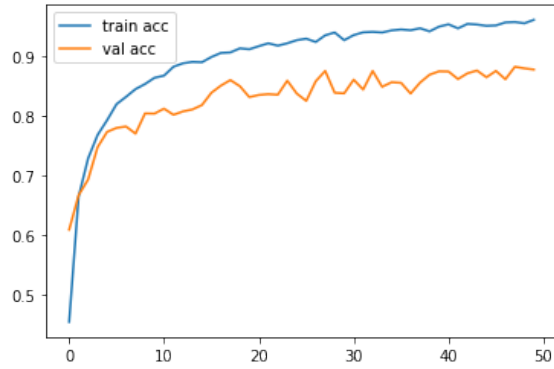


Figure 5.7: InceptionV3 Accuracy

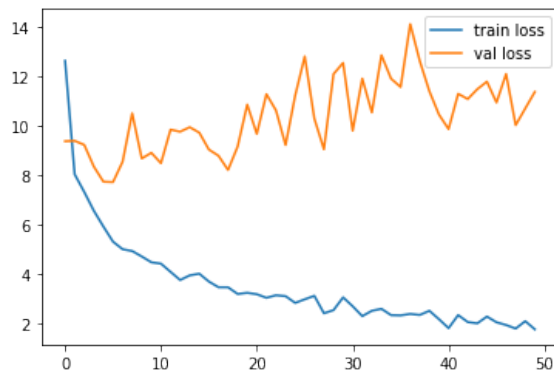


Figure 5.8: InceptionV3 Loss

In figure 5.7 and 5.8 training accuracy for this pre-trained InceptionV3 model starts of faster than the others, quickly shooting up to about 80% by epoch 5. After epoch 5 the two curves split up. Training accuracy continues to increase steadily with some variance upto 96% . Validation accuracy splits off at epoch 5 and with a lot of variance, slowly increases to about 88%.

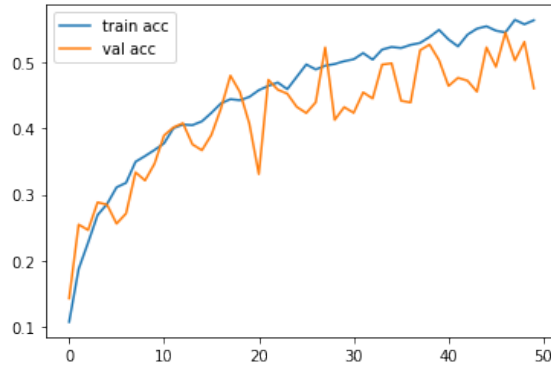


Figure 5.9: Resnet50 Accuracy

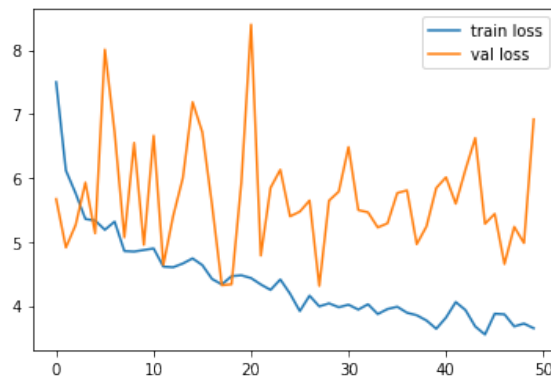


Figure 5.10: Resnet50 Loss

In figure 5.9 and 5.10 Resnet50 model training started off with accuracy 1% which gradually increases to a maximum of around 56% with some variance. Validation accuracy never went above 54% with a large variance in its values. Training loss started from 7.5 and went to 3 over 50 epochs. Validation was incredibly unstable ranging in value from 4.5 all the way up to 8.5 throughout the epochs never holding a consistent value.

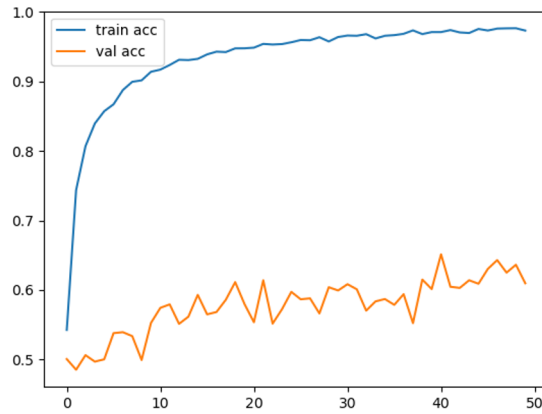


Figure 5.11: MobileNetV2 Accuracy

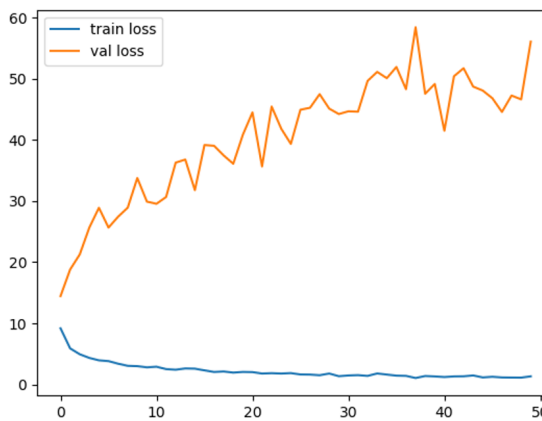


Figure 5.12: MobileNetV2 Loss

In figure 5.11 and 5.12 shows MobileNetV2 had 90% training accuracy and 60% validation accuracy and training loss is fluctuating between 10% to 0% and validation loss is also fluctuating between 11% to 58%.

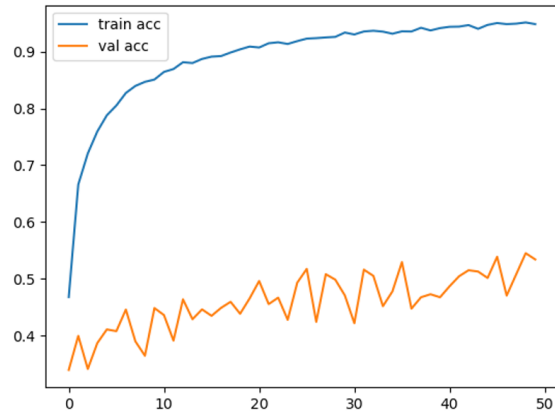


Figure 5.13: InceptionResnet Accuracy

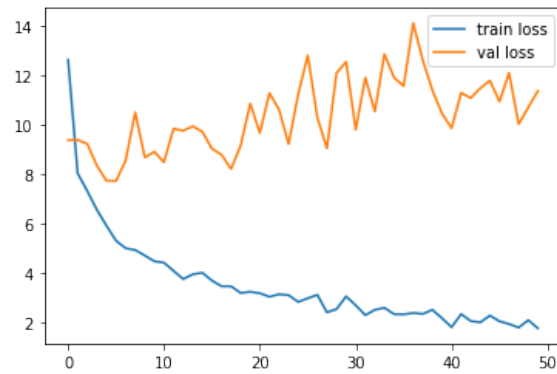


Figure 5.14: InceptionResnet Loss

In figure 5.13 and 5.14 shows InceptionResnet had 94% training accuracy and 53% validation accuracy and training loss is fluctuating between 13% to 2% and validation loss is also fluctuating between 9% to 14%.

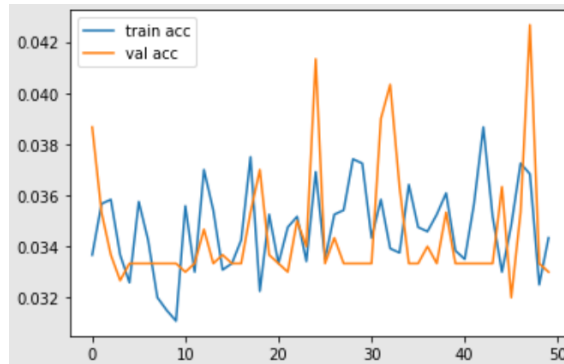


Figure 5.15: EfficientNet Accuracy

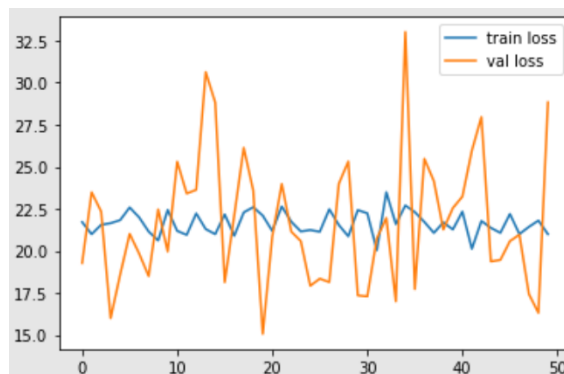


Figure 5.16: EfficientNet Loss

In figure 5.15 and 5.16 EfficientNet had the lowest accuracy by far. Training accuracy never went beyond 4% and did not increase across 50 epochs. Validation accuracy was also erratic and managed a maxima of 4.2% at its peak.

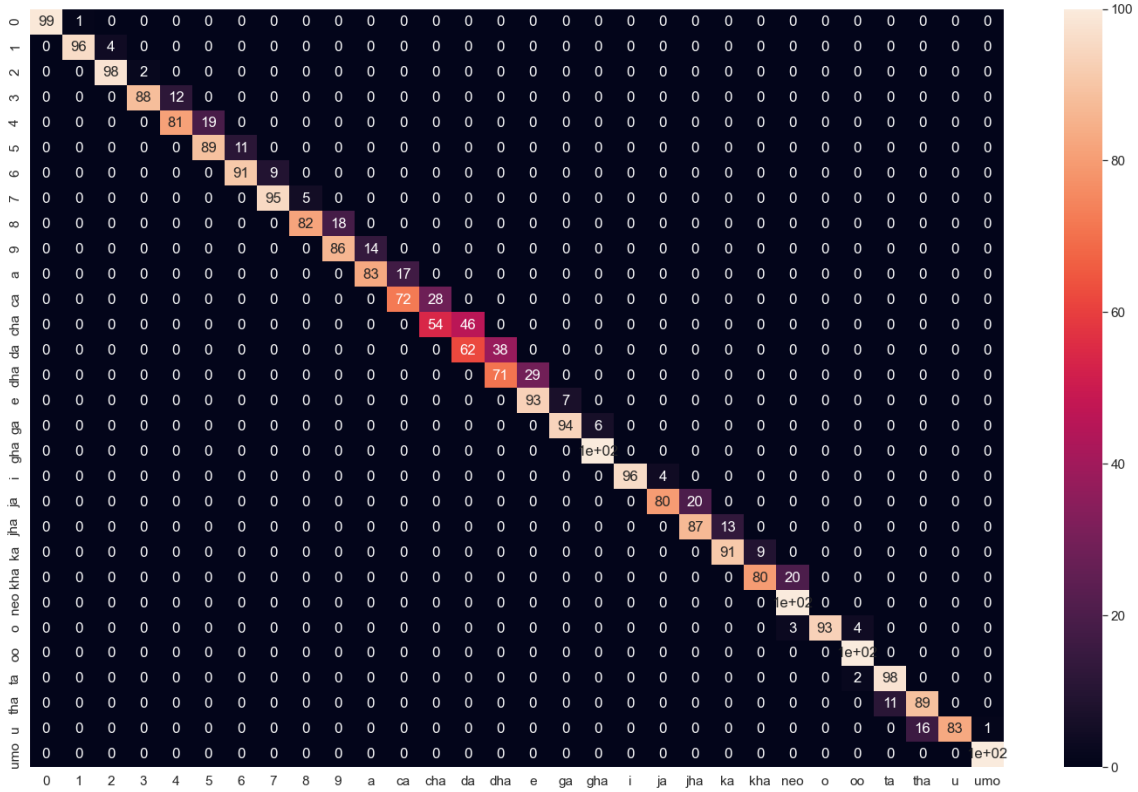


Figure 5.17: Confusion Matrix of Custom CNN model

A confusion matrix summarises the performance of a model for each class in one elaborate yet simple diagram. In figure 5.17 our confusion matrix is a straight diagonal line traveling from top left to down right as a clean line indicating that our model works very well at classifying most of our classes effectively. The problems come from some of the more complex letters that are constructed from a combination of straight and rounded lines and also closely resemble other letters. Notable examples would be *Ca* and *Dha*.

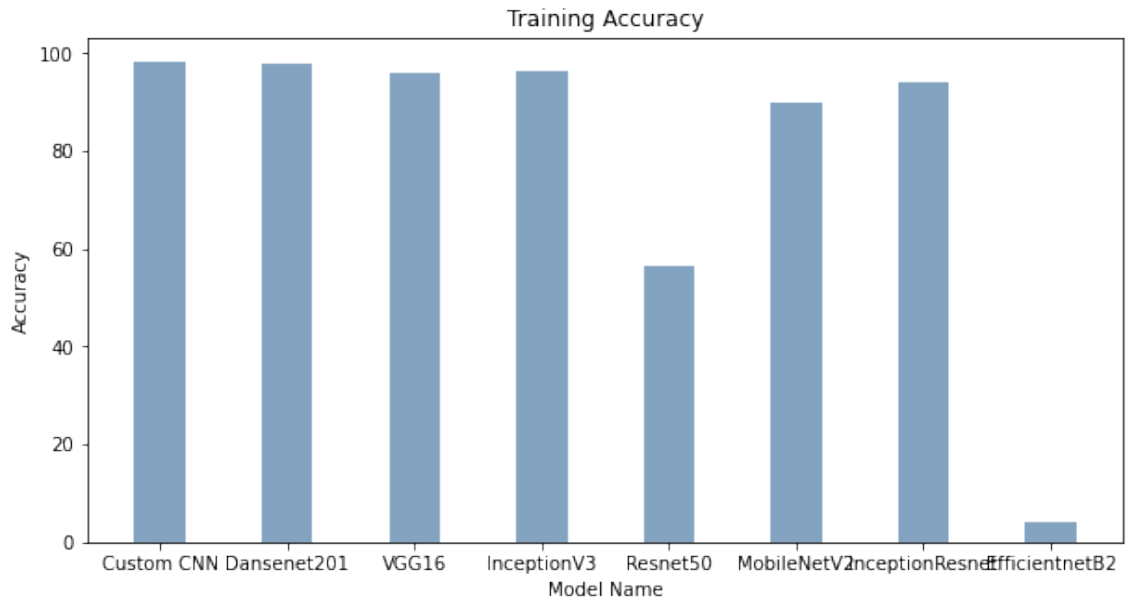


Figure 5.18: Train Accuracy

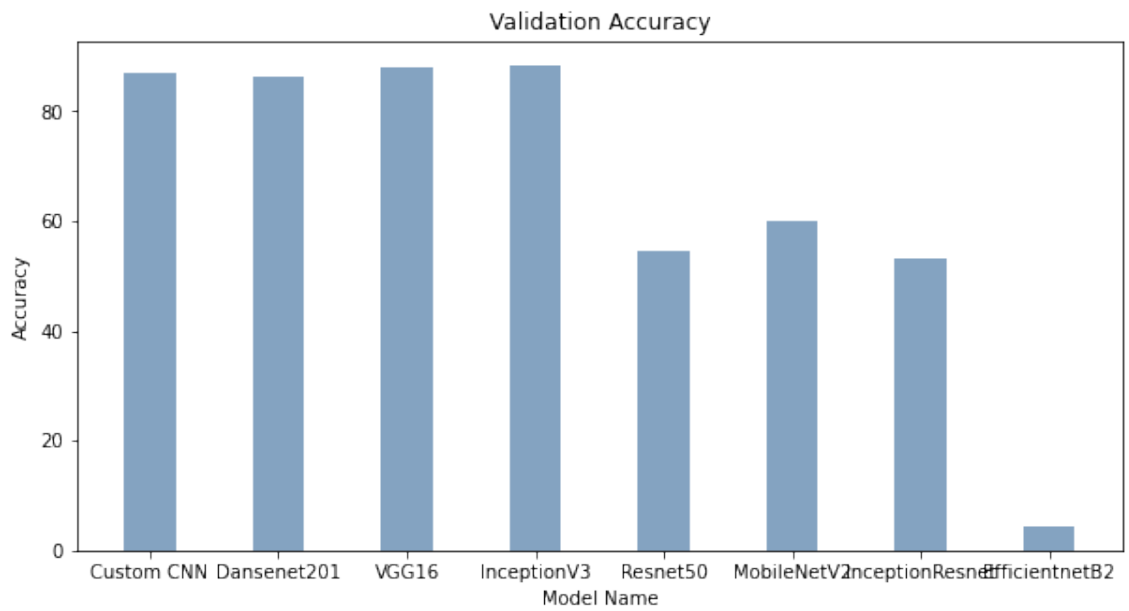


Figure 5.19: Validation Accuracy

In a nutshell, we know that a loss function graph can be used to determine if a model is over-fitting or not. When models rely too much on their training data and lose their capacity to perform effectively on new data, this is known as over-fitting. In other words, the model includes and learns from the random oscillations in the training data. We might have had minor over-fitting concerns, as shown by the loss function graphs of the classic transfer learning models shown above. These over-fitting problems had an effect on the predictability of outcomes. This problem was resolved via normalizing and the addition of few dropout layers, which increased the prediction accuracy of the Bangla Sign Language categorization. Additionally, the total number of layers in our unique CNN model is 12, which makes it lighter and faster to train. Finally, as seen in figure 5.1, our customized CNN model performs better than any other pre-trained models and provides us with greater accuracy.

Chapter 6

Conclusion

6.1 Discussion

We worked on 15000 images of hand gestures. All these images were augmented to increase the number of data. Then it was split into two folders using split-folders library. One is train dataset which contains 80 percent of the image of whole dataset. The validation dataset which contains rest 20 percent images. Then we applied various pre-trained models on this dataset. We used Densenet201, VGG16, InceptionV3, Resnet50, MobileNetV2, InceptionResnet, EfficientnetB2. Among them DenseNet201 gave the highest training and validation accuracy 98 percent and 86 percent accordingly.

The objective of this study was to create a model which can predict Bangla Sign Language as accurate as possible. To satisfy our goal we tried to build a custom CNN model. Which includes in total 12 layers. To improve the accuracy and reduce overfitting issue, we added several dropout layers and normalizing layers. Then we arrived at a model which gave better accuracy (98.3 percent training and 87 percent validation) than any tested pre-trained model ensuring less depth of neurons, lighter model and faster training time.

6.2 Conclusion

Nowadays, sign language is very essential for people who have problems in hearing and talking. This dataset contains Bangla Sign Language which is developed by using convolutional neural networks. We used CNN architecture for faster delivery. This system will be the digital interpreter between deaf and normal people which can be easier for understand the language to deaf and also very friendly for hearing people. Initially, this paper contains Sign language of all the Bangla Alphabets and further we will also work for the Bangla Numerical Signs. Conducting this procedure as friendly as possible for both hearing and non-hearing people is our main objective.

6.3 Future Plan

We intend to work with all Bengali alphabets in the future. But we also want to apply it in real world applications which will be able to identify and differentiate between different bangla sign languages in real time. We had over-fitting problems in our current system. We'll take care of this by enhancing the data.

Bibliography

- [1] S. Rahman, N. Fatema, and M. Rokonzaman, "Intelligent assistants for speech impaired people," in *ICCIT*, 2002.
- [2] O. B. Hoque, M. I. Jubair, M. S. Islam, A.-F. Akash, and A. S. Paulson, "Real time bangladeshi sign language detection using faster r-cnn," in *2018 international conference on innovation in engineering and technology (ICIET)*, IEEE, 2018, pp. 1–6.
- [3] M. S. Islam, S. S. S. Mousumi, N. A. Jessan, A. S. A. Rabby, and S. A. Hossain, "Ishara-lipi: The first complete multipurpose open access dataset of isolated characters for bangla sign language," in *2018 International Conference on Bangla Speech and Language Processing (ICBSLP)*, IEEE, 2018, pp. 1–4.
- [4] S. Islam, S. S. S. Mousumi, A. S. A. Rabby, S. A. Hossain, and S. Abujar, "A potent model to recognize bangla sign language digits using convolutional neural network," *Procedia computer science*, vol. 143, pp. 611–618, 2018.
- [5] S. S. Shanta, S. T. Anwar, and M. R. Kabir, "Bangla sign language detection using sift and cnn," in *2018 9th international conference on computing, communication and networking technologies (ICCCNT)*, IEEE, 2018, pp. 1–6.
- [6] M. S. Islalm, M. M. Rahman, M. H. Rahman, M. Arifuzzaman, R. Sassi, and M. Aktaruzzaman, "Recognition bangla sign language using convolutional neural network," in *2019 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*, 2019, pp. 1–6. DOI: 10.1109/3ICT.2019.8910301.
- [7] S. Hossain, D. Sarma, T. Mittra, M. N. Alam, I. Saha, and F. T. Johora, "Bengali hand sign gestures recognition using convolutional neural network," in *2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA)*, IEEE, 2020, pp. 636–641.
- [8] N. Basnin, L. Nahar, and M. S. Hossain, "An integrated cnn-lstm model for bangla lexical sign language recognition," in *Proceedings of International Conference on Trends in Computational and Cognitive Engineering*, Springer, 2021, pp. 695–707.
- [9] F. Shamrat, S. Chakraborty, M. M. Billah, M. Kabir, N. S. Shadin, and S. Sanjana, "Bangla numerical sign language recognition using convolutional neural networks," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 23, no. 1, pp. 405–413, 2021.
- [10] T. Das and M. Islam, "Bangla sign language alphabet recognition using hand gestures: A deep learning approach,"