Neural Networks Project - Gesture Recognition

Srinivas Yalagam, Satyajit Chaudhuri
PG Scholar, IIITB

Abstract:

Gesture recognition can be seen as a way for computers to begin to understand human body language, thus building a better bridge between machines and humans than older text user interfaces or even GUIs (graphical user interfaces), which still limit the majority of input to keyboard and mouse and interact naturally without any mechanical devices. The following research work focuses on devising a neural network architecture that can recognise five different gestures performed by the user. This architecture is then to be employed by a home electronics company which manufactures state of the art smart televisions to help users control the TV without using a remote.

Introduction:

Myron W. Krieger was the first who proposed gesture recognition as a new form of interaction between human and computer in the mid-seventies. Actually, gesture recognition has been adapted for various other research applications from facial gestures to complete bodily human action. Thus, several applications have emerged and created a stronger need for this type of recognition system. Gesture recognition is an important topic in computer vision because of its wide range of applications, such as HCI, sign language interpretation and visual surveillance. Spatial variation and temporal variation among gesture samples make recognition difficult. For example, different subjects have different hand appearances and thus may sign gestures at different paces. Gesture recognition studies started as early as 1992 when the first frame grabbers for colored video input became available, which enabled researchers to grab colored images in real time. This study signified the start of the development of gesture recognition because color information improves segmentation and real-time performance is a prerequisite for HCI.
Until a few years ago, nearly all work in this area used mechanical devices to sense the shape of the hand. The most common device used was a thin glove mounted with bending sensors of various kinds, such as the data glove. A magnetic field sensor was often used to detect the hand position and orientation. However, this approach was notorious because of the problem of distorting the magnetic field encountered in the electromagnetically noisy environment of most computer laboratories. A primary goal of gesture recognition research is to create a system that can identify specific human gestures and use them to convey information or to control devices.
Some existing applications of gesture recognition are as follows:
(1) interaction with virtual environment, for example, in one of the applications, the user painted on a virtual wall with an extended finger and erased what they had done with their spread open hand,
(2) sign language understanding and
(3) as a part of more traditional computer interfaces such as the use of gesture as a direct mouse replacement.

The following research work deals with recognising five different gestures performed by the user which will help users control the TV without using a remote.
The five gestures are:

Thumbs up:	Increase the volume.
Thumbs down:	Decrease the volume.
Left swipe:	'Jump' backwards 10 seconds.

Right swipe:	'Jump' forward 10 seconds.
Stop:	Pause the movie.

The Key objective of the study is to build neural network architectures and train them on the 'train' folder to predict the action performed in each sequence or video and which performs well on the 'val' folder as well.

Understanding the Dataset:

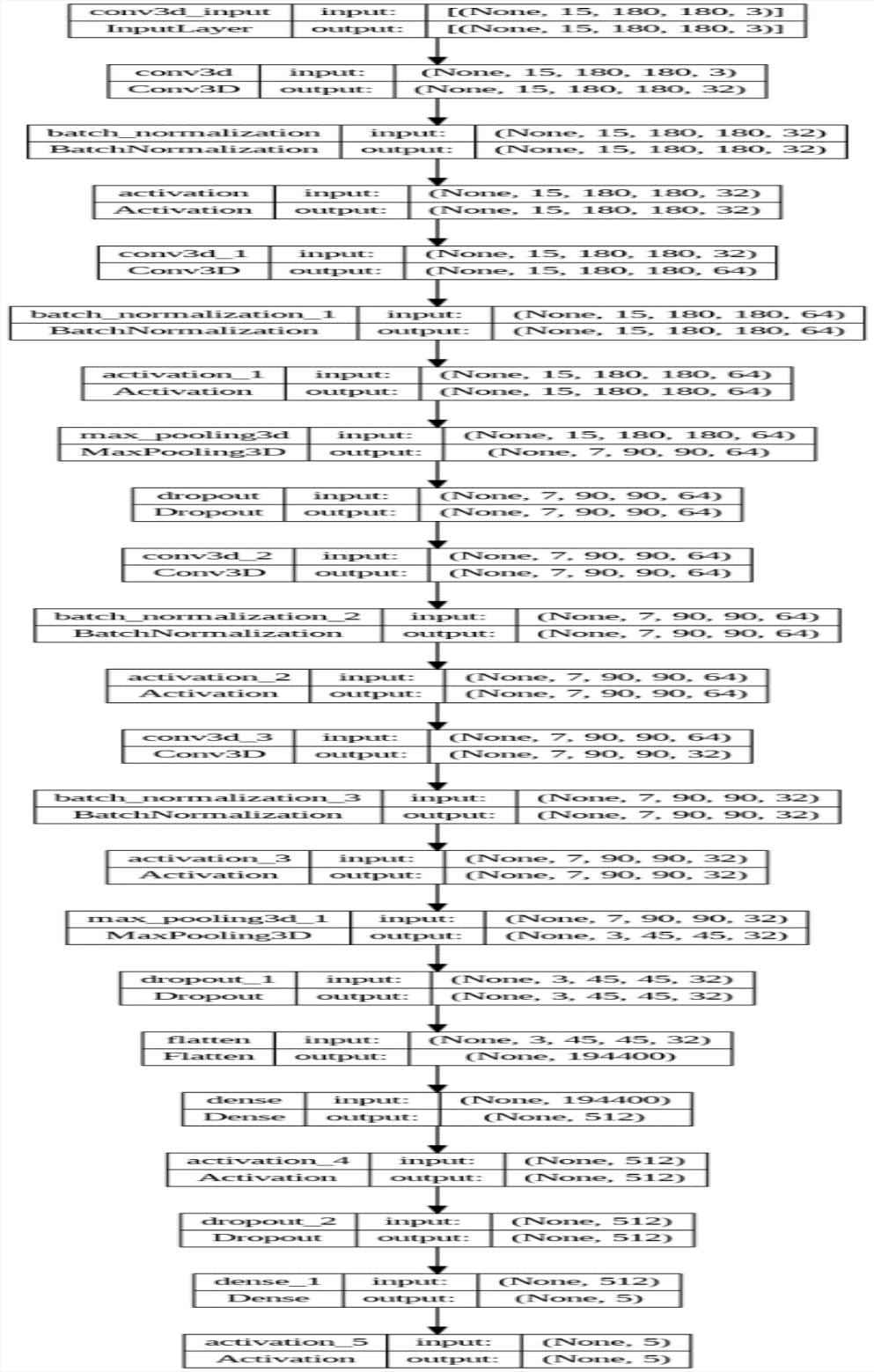
The training data consists of a few hundred videos categorised into one of the five classes. Each video (typically 2-3 seconds long) is divided into a sequence of 30 frames (images). These videos have been recorded by various people performing one of the five gestures in front of a webcam - similar to what the smart TV will use. The data is in a zip file. The zip file contains a 'train' and a 'val' folder with two CSV files for the two folders. These folders are in turn divided into subfolders where each subfolder represents a video of a particular gesture. Each subfolder, i.e. a video, contains 30 frames (or images). Note that all images in a particular video subfolder have the same dimensions but different videos may have different dimensions. Specifically, videos have two types of dimensions - either 360x360 or 120x160 (depending on the webcam used to record the videos). Hence, you will need to do some pre-processing to standardise the videos. Each row of the CSV file represents one video and contains three main pieces of information - the name of the subfolder containing the 30 images of the video, the name of the gesture and the numeric label (between 0-4) of the video.

Methodology:

For analysing videos using neural networks, two types of architectures are used commonly. One is the standard CNN + RNN architecture in which the images of a video are passed through a CNN which extracts a feature vector for each image, and then the sequence of these feature vectors are passed through an RNN. The other popular architecture used to process videos is a natural extension of CNNs - a 3D convolutional network. A 3D CNN uses a three-dimensional filter to perform convolutions. The kernel is able to slide in three directions, whereas in a 2D CNN it can slide in two dimensions. Based on the very same tenets, seven distinct experimental models have been developed, each with a different parameter or architectural setting. The models have then been trained on the training data and validated consequently. The architecture with the best validation accuracy has finally been selected for evaluation on the test dataset.
In the following sections, we describe each model with intuition behind the model, the model architecture and the performance of the network.

Model 1:

The first model is the base model on which the later models have been built. This model has seen a batch size of 8. The image dimensions are (180,180) and the model is trained on 16 epochs. The kernel size has been kept as (3,3,3) and drop outs have been used after 3D Convolution layers and dense layers. The model uses four convolution layers. The first layer has a filter size of 32, followed by 64, which is again followed by a convolution layer of filter size 64 and 32 for the last layer. The architecture then saw a dense layer of 512 neurons followed by the output layer with 5 neurons. The model architecture is as below:



Model 1: Base Model

Model 1	Loss	Accuracy
Train	0.1743	0.9517
Validation	0.8000	0.8200

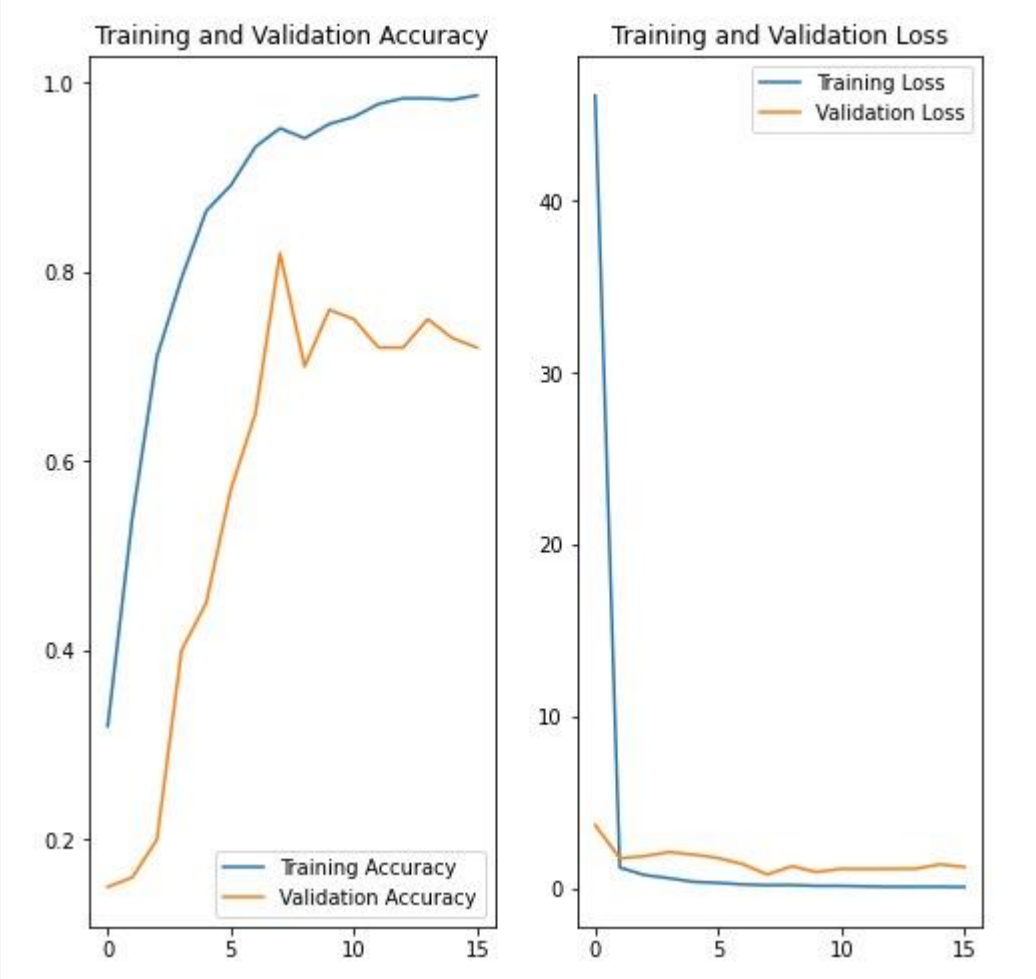
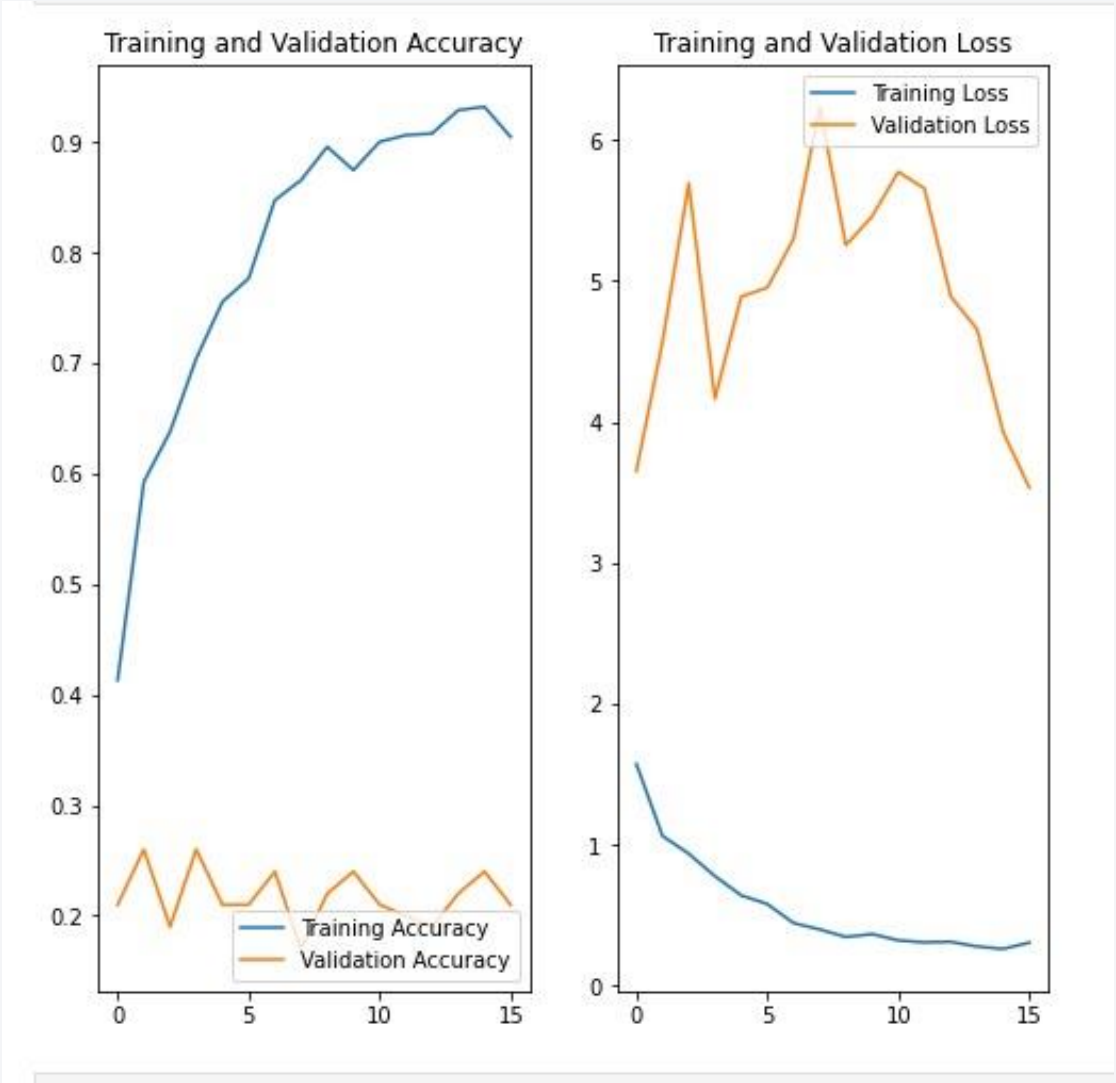


Figure 1: Model -1 Performance Evaluation.

Model 2:

As seen above, it is quite evident that the model does not perform very well on the validation set. Hence the study proceeds by developing a new network. The batch size is increased to 20 and the image size is reconfigured to (160,160). The architecture used eight convolutions with kernel size of (3, 3, 3). This is then followed by two dense neural networks. The first convolution network has 16 filters, the second and third network has 32 filters, the fourth, fifth and sixth convolution layers have 64 filters and the last two convolution layers have 128 filters. The activation functions used in the convolution nets is “relu”. This is then followed by two dense networks of 64 neurons. The dropouts are being introduced after each dense layer. The final dense layer has 5 output neurons. The model performance is much poor then the base model itself. The performance is depicted below:

Model 2	Loss	Accuracy
Train	0.3074	0.9040
Validation	3.530	0.2100

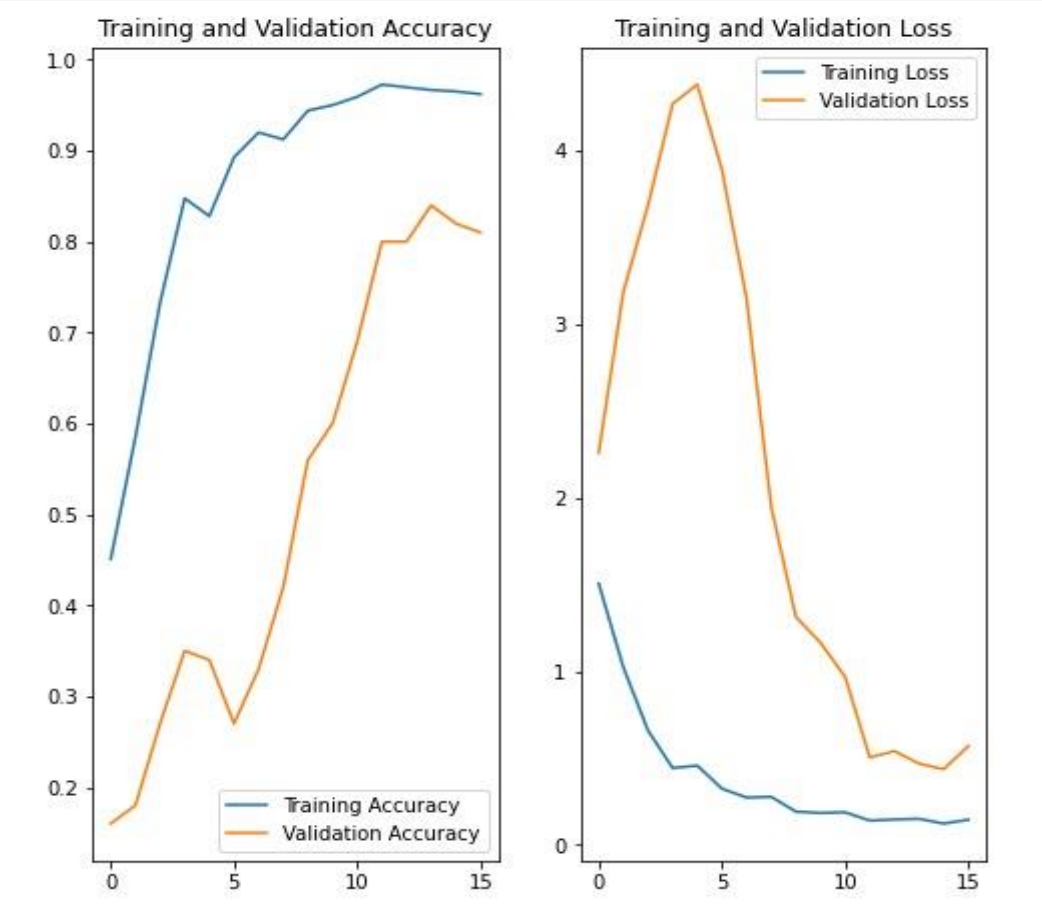


The Train & Validation performance of Model 2

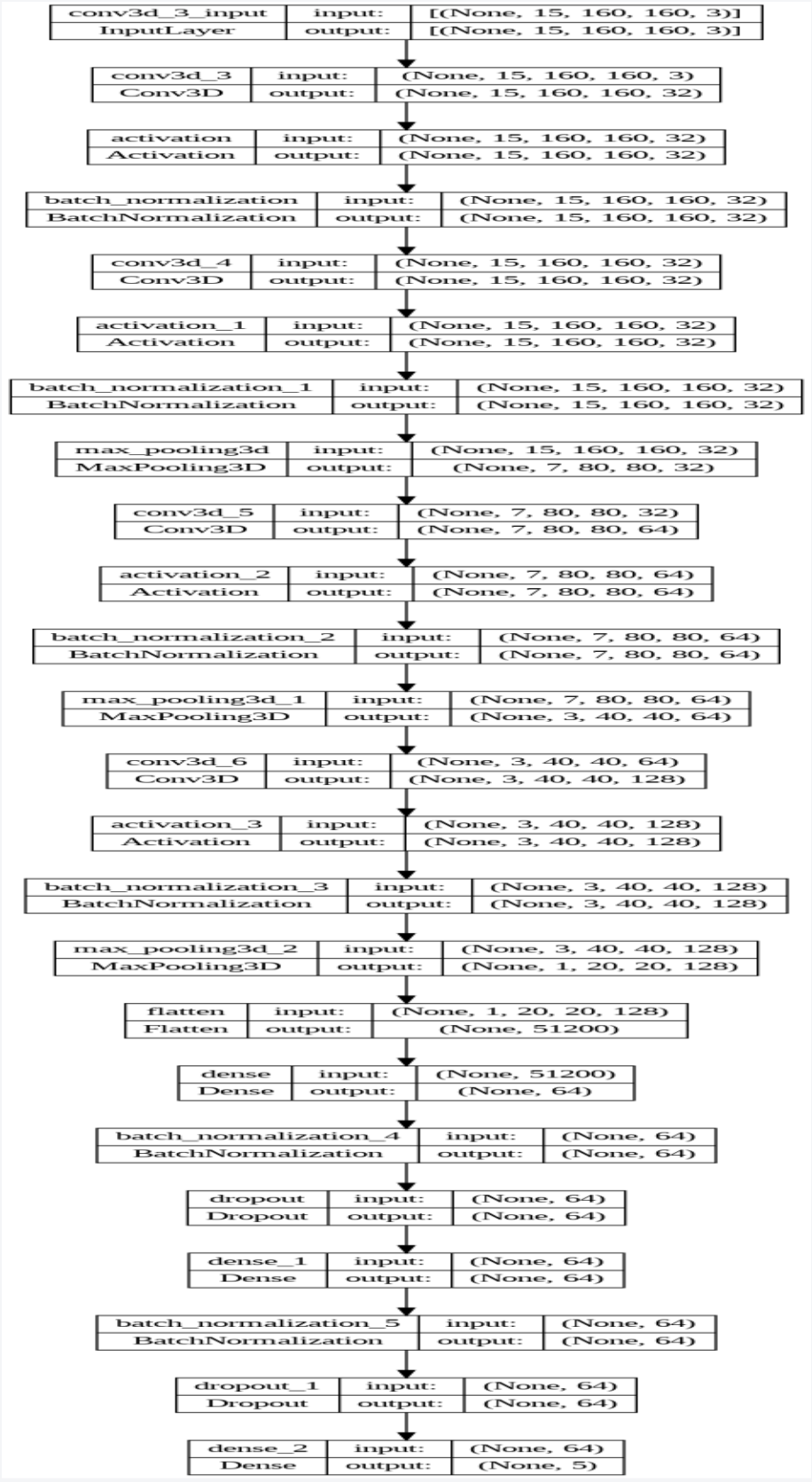
Model 3:

As seen in the previous model, increasing the number of convulations layers did not result into increase in validation accuracy and hence in this trial the number of layers have been reduced keeping the batch size and the image size same. The kernel size and the activation functions have also been kept same. The Batch Normalization has been applied after each convulation layer. The max pooling layers of pool size (2, 2, 2) have been used after the second, third and fourth convulation layers. The number of epochs is maintained at 16. The model is composed of four convulation layers. The first and second layer has 32 filters, the next has 64 filters, the last layer has 128 filters. The dense networks and the dropouts have been kept the same as in the previous model.

Model 3	Loss	Accuracy
Train	0.1219	0.9653
Validation	0.4350	0.8200



The Train & Validation performance of the Model 3

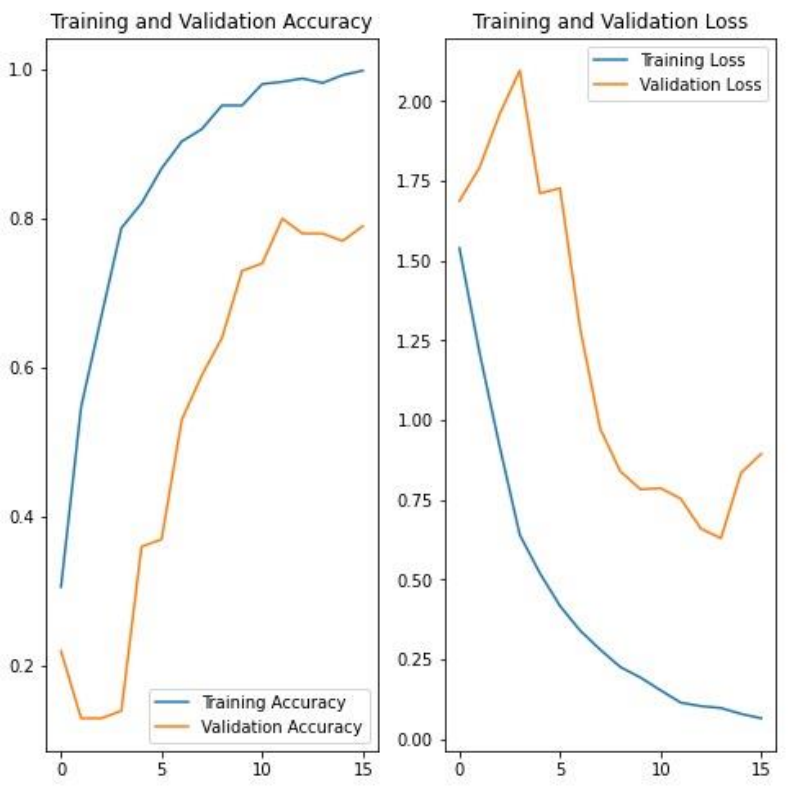


Model 3: The architecture of the third model.

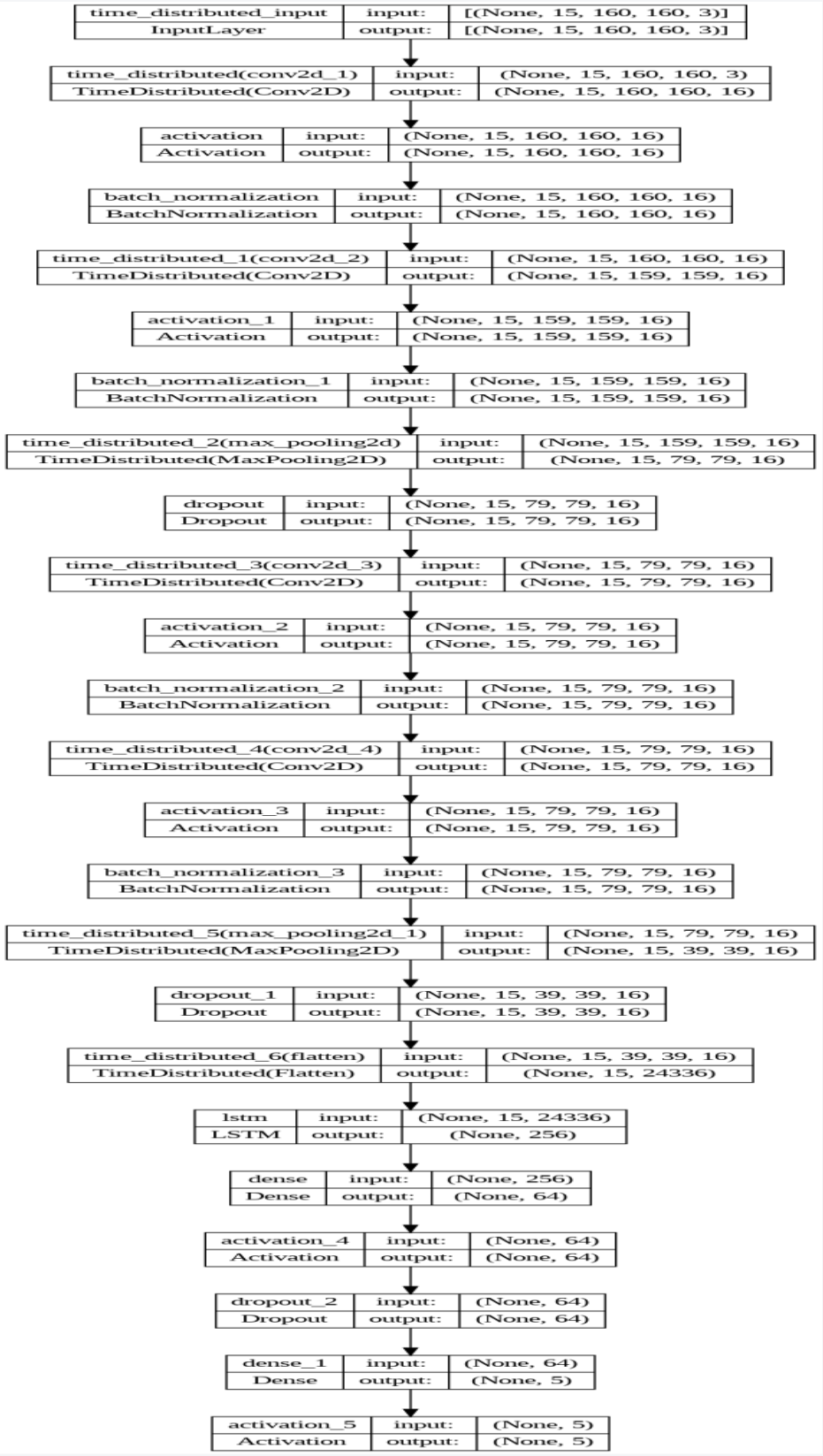
Model 4:

In this fourth experiment, the basic framework is being shifted from Convulation 3D to a CNN+RNN based approach. In this model, the architecture uses the Long Short Term Memory Cells. The model uses four two-dimensional convulation layers having 16 filters and kernel size of (2,2). The number of epochs have been kept at 16 like previous models only. The activation functions employed here is rectified linear unit. Batch Normalization has been used after each convulation layer. The max pooling layers have been used after the first two layers and a dropout layer of 0.2 dropout. The next max pooling and dropouts have been employed after the third and fourth layer.
All these layers are time distributed networks in line with the RNN tenets. After the convulation layers, the LSTM cell with 256 neurons cells have been employed with a dropout of 0.5. This is followed by a dense neuron network of 64 neurons and dropout of 0.25. The last layer is the output with 5 output neurons and softmax activation.
The model performances are as follows:

Model 4	Loss	Accuracy
Train	0.0969	0.9819
Validation	0.6290	0.7800



Train and Validation Results for Model 4.

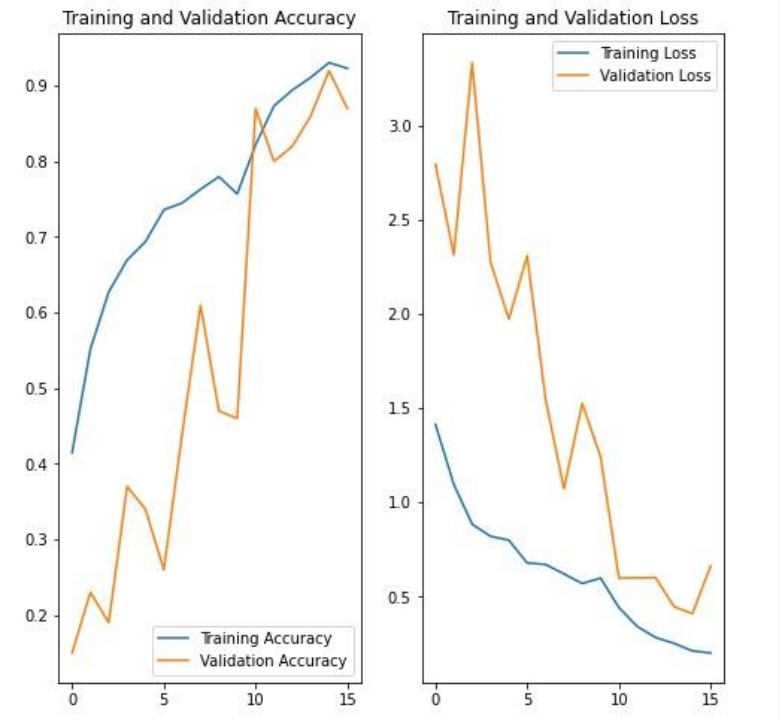


Model 4: The architecture of the fourth

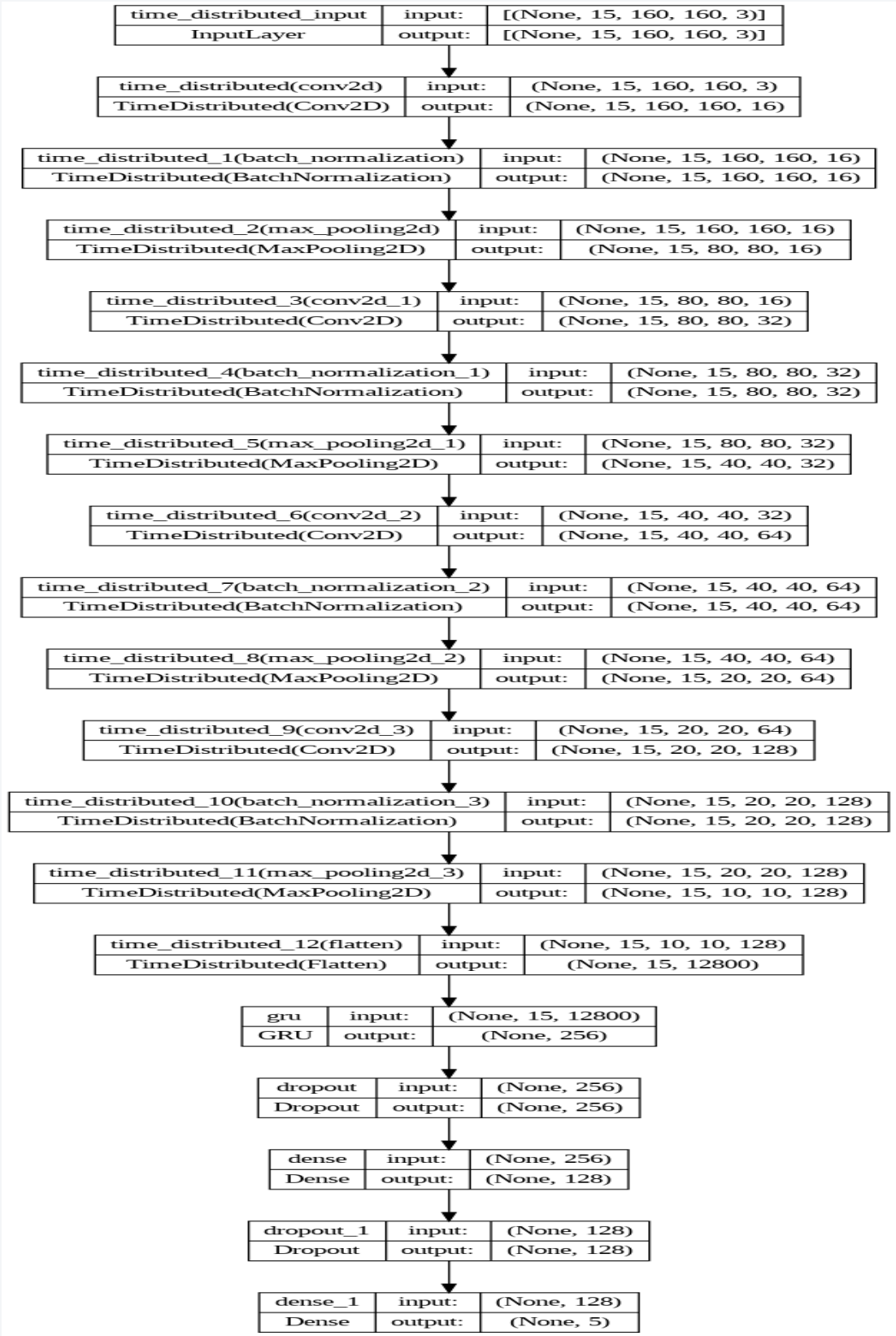
Model 5:

The last model is composed of Gated Recurrent Units. The key difference between GRU and LSTM is that GRU's bag has two gates that are reset and update while LSTM has three gates that are input, output, forget. GRU is less complex than LSTM because it has less number of gates. If the dataset is small then GRU is preferred otherwise LSTM for the larger dataset. Thus the GRU model is being used in this study to see if it is more efficient for this particular task.
In this model, the batch size has been kept at 12 , image sizes (160.160) and number of epochs at 16. The model is built up of 4 convulational units. The first layer has 16 filters, followed by 32 filters. This is being followed up by 64 filters and last CNN layer has 128 filters. The kernel size is kept at (2,2) for first layer and (3,3) for remaining layers. The batch normalization and max pooling has been performed after each CNN layer.This is then folllowed up by the GRU layer of 256 neurons. This is then succedded by a dropout layer. The remaining network is composed of densely connected neural nets finally giving an output of 5 classes.
The model performance of this model is encouraging and listed as follows:

Model 5	Loss	Accuracy
Train	0.2113	0.9300
Validation	0.4090	0.9200



Train and Validation Results for Model 5.

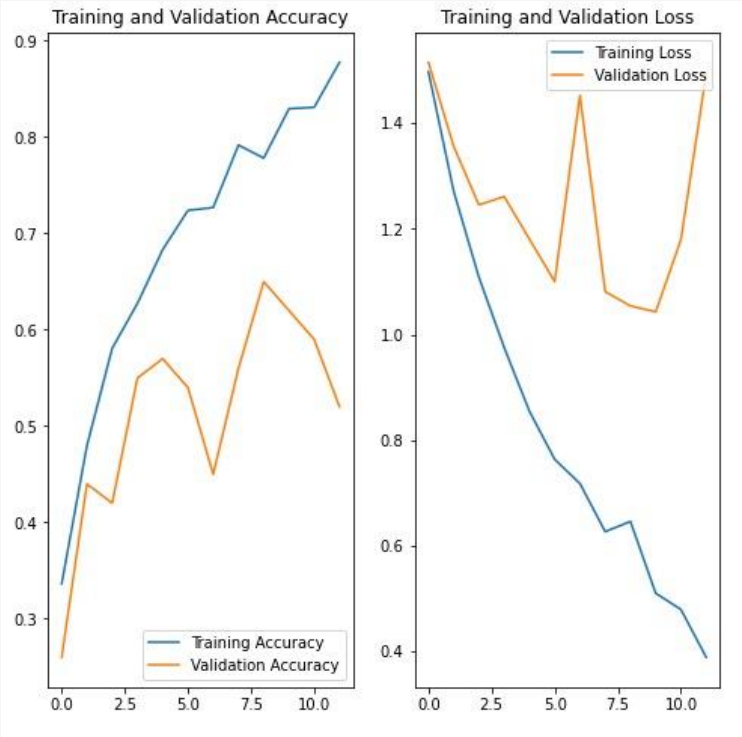


Model 5: The architecture of GRU model.

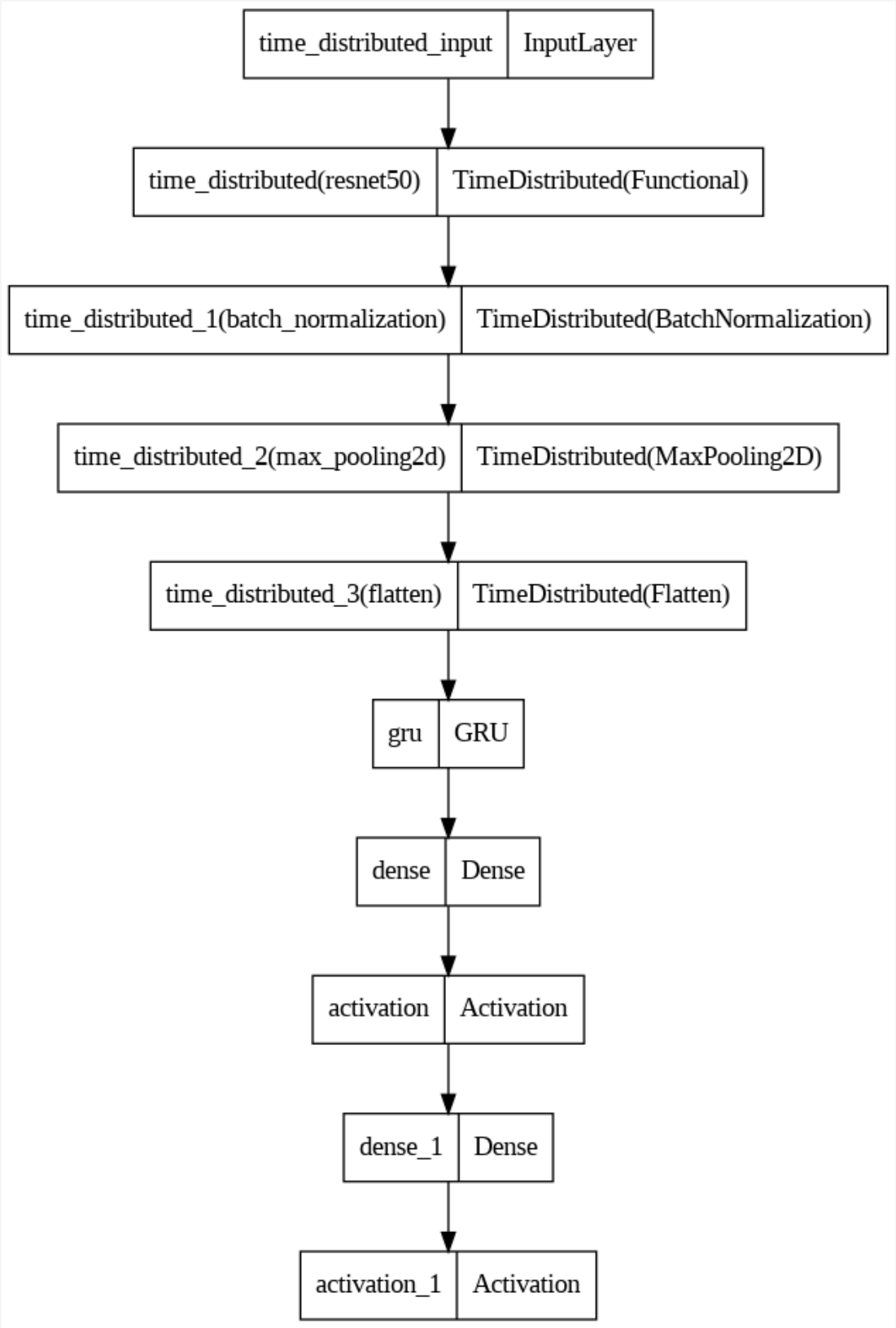
Model 6:

After succesfully testing the CNN+RNN architecture, the experiment focuses on using the transfer learning methodology. The study now focusses on using the Resnet architecture to convolve on the images which is then coupled with the Gated Recurrent Units to make predictions on the validation set.
Here also, the batch sizes are kept at 12m, image sizes at (160,160) and epoch is set to 12. The model uses the weights similar to that of the resnet model. After the CNN layer a Batch Normalization layer and max pooling is applied. This is then followed by GRU units of 128 neurons. This is then followed by dense network of 64 neurons and the final output neuron has 5 output neurons with softmax function.
The model performances are as follows:

Model 6	Loss	Accuracy
Train	0.5090	0.8200
Validation	1.0400	0.6200



The Train & Validation results of Resnet Model

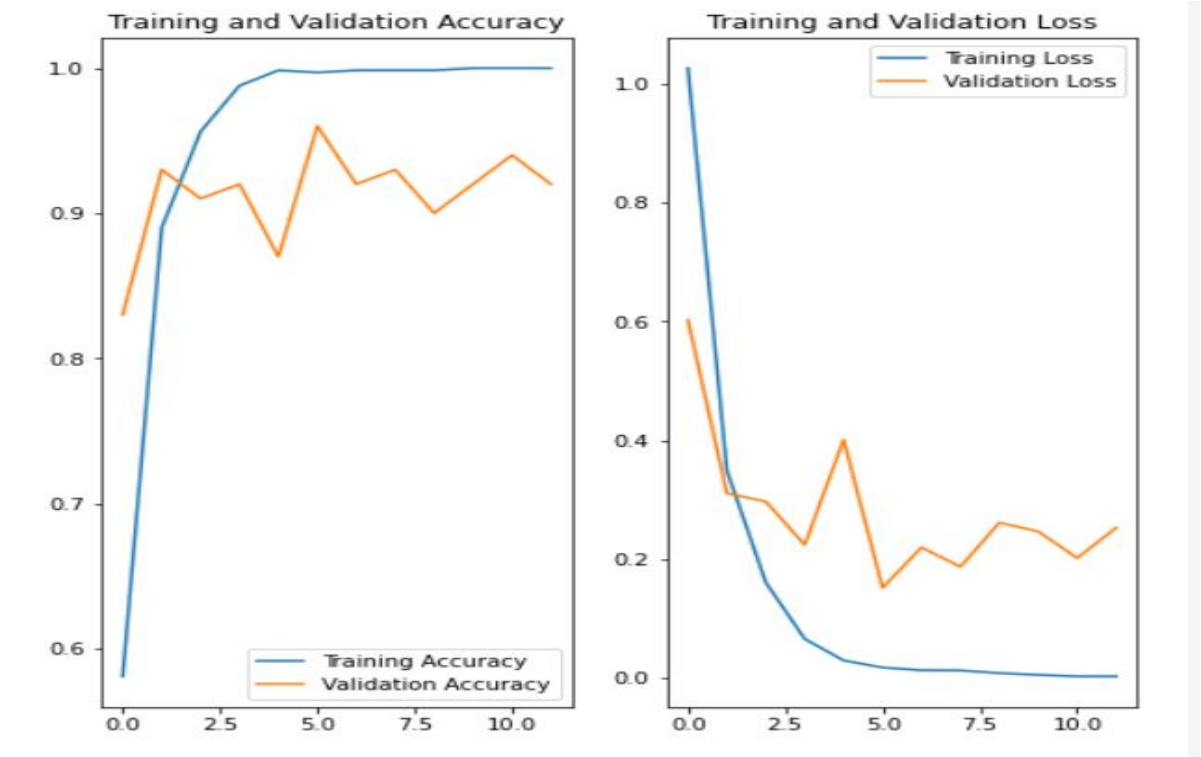


The RESNET + GRU Architecture

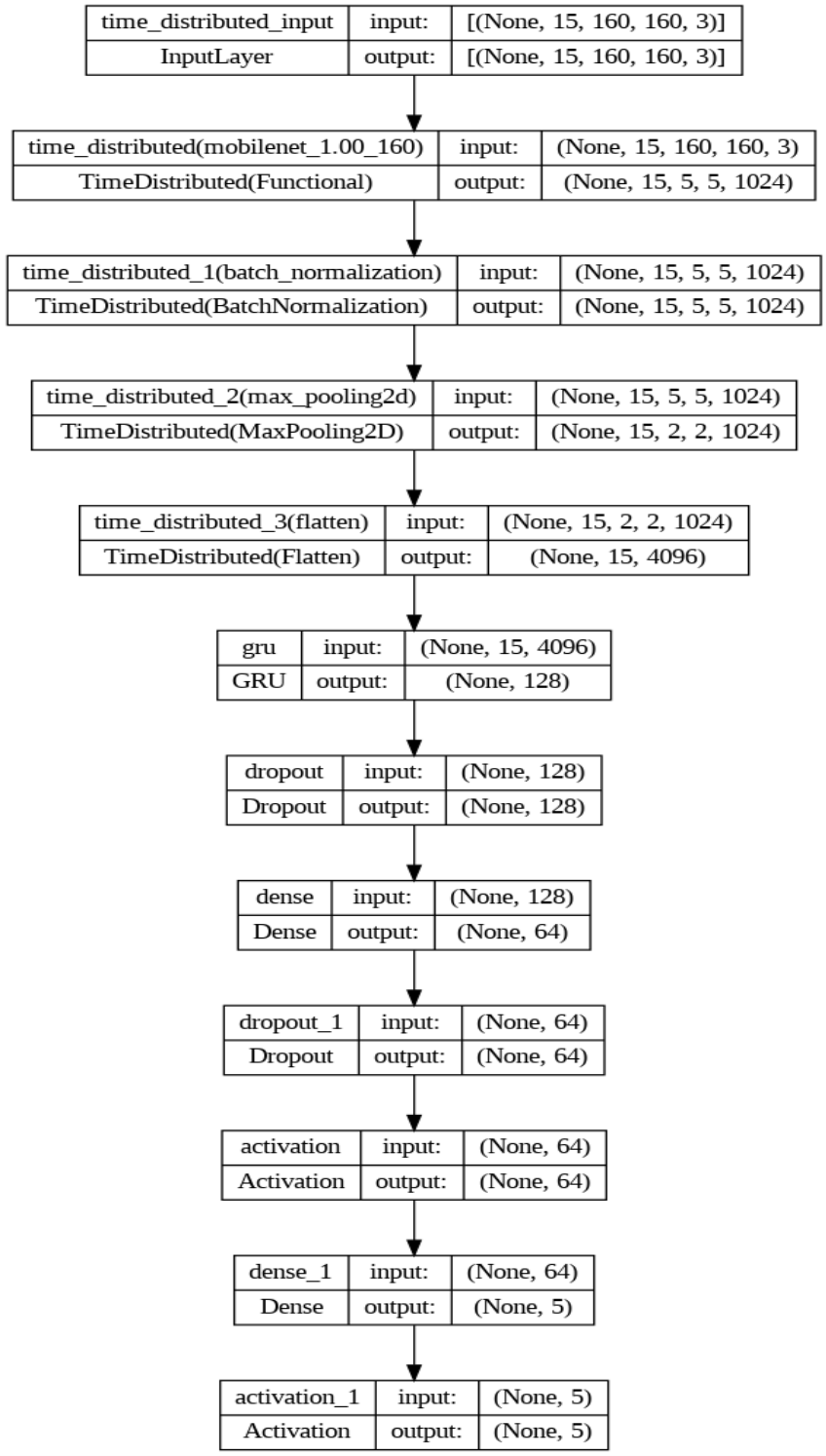
Model 7:

The transfer learning model Mobilenet is now being used to train the model. MobileNet is a streamlined architecture that uses depthwise separable convolutions to construct lightweight deep convolutional neural networks and provides an efficient model for embedded vision applications. Now this advantage of mobilenet is being harnessed to work alongwith the Gated Recurrent Units to test on the validation data.
Here also, the batch sizes are kept at 12m, image sizes at (160,160) and epoch is set to 12. The model uses the weights similar to that of the resnet model. After the CNN layer a Batch Normalization layer and max pooling is applied. This is then followed by GRU units of 128 neurons. This is then followed by dense network of 64 neurons and the final output neuron has 5 output neurons with softmax function.
The model performances are as follows:

Model 7	Loss	Accuracy
Train	0.0176	0.996
Validation	0.151	0.9600



The results of Mobilenet model on Train & Validation set.



The architecture of MOBILENET + GRU Model

Conclusion

In this experimental study, the intial investigations began with creating a 3D CNN model to check for its feasibility on the given dataset. This was followed by introduction of more layers of CNN. In Model-2, eight CNN layers were introduced but this did not effect the performance significantly. Thus the focus was shifted to have the data tested on less deep models. In model-3, the CNN network had fewer layers and the performance was significantly better. This three experiments were done by using the Convolution 3D methodology. Amongst the models, Model 3 performed the best.
Then the focus shifted to the CNN+RNN architectural model. Model-4 saw CNN layers coupled with LSTM model. The model did perform good, but the search was on for better performing architectures and hence, GRU was coupled with CNN and the results did improve in Model-5 from the previous model. The final approach was to use the transfer learning with GRU and hence, RESNET (model-6) & MOBILENET(model-7) were used to train and validate the dataset.The Mobilenet+GRU model was found to be the best performing architecture and this is considered for final submission. This model gave a training accuracy of 99.6% and 96% accuracy for validation set.

Model no.	Model	Description			Train loss	Train Accuracy	Val loss	Val Accuracy
		Batch Size	Image Size	Epochs				
1	3D CNN Base Model	8	(180,180)	16	0.1744	95.17%	0.8000	82.00%
2	3D CNN with increased layers.	20	(160,160)	16	0.3075	90.40%	3.5300	21.00%
3	3D CNN with reduced layers.	12	(160,160)	16	0.1220	96.53%	0.4350	82.00%
4	CNN + LSTM	12	(160,160)	16	0.0970	98.19%	0.6290	78.00%
5	CNN + GRU	12	(160,160)	16	0.2113	93.00%	0.4090	92.00%
6	RESNET + GRU	12	(160,160)	12	0.5090	82.00%	1.0400	62.00%
7	MOBILENET + GRU	12	(160,160)	12	0.0176	99.61%	0.1510	96.00%

Hence, the MOBILENET + GRU architecture developed in the seven model is found to be best fit model for gesture recognition application in the to-be-designed Smart TV's by the company.