PGDMLAI - 2018/19

**Gesture Recognition Project**

Model Experimentation and Fine-Tuning

horizontal line

# 

**Srikanth Srinivasan**

**Aakash Verma**

**Aryan Singh**

**Contents**

[Introduction 3](#_Toc533962771)

[Image Size and Generator Considerations 3](#_Toc533962772)

[Image Augmentation 3](#_Toc533962773)

[LRCN model 4](#_Toc533962774)

[Variants Tried and Corresponding Experiments 5](#_Toc533962775)

[Resnet + GRU : 5](#_Toc533962776)

[Inception V3 + GRU : 6](#_Toc533962777)

[3-D Convolution 9](#_Toc533962778)

[Conclusion 12](#_Toc533962779)

# Introduction

Gesture Recognition project is a use case to automate certain functions on smart TV on the basis of the gestures/actions being presented by the viewer in front of the webcam. Technically, the goal is to run the series of 30 frames from each video through a model with a temporal aspect to it in order to generate the correct label for each video. For this we consider two models:

* CNN-GRU combination called LRCN
* 3-D Convolutional Model

# Image Size and Generator Considerations

Image size was given special attention with ablation experiments with different sizes like 60 x 80, 120 x 120, 197 x 197 and 300 x 300. Finally 300 x 300 gave state of the art results on Inception V3 + GRU and an image size of 60 x 80 gave best results on 3-D convolution.

**Normalisation:**

To make sure we account for the variation in pixel ranges we used min max normalisation instead of dividing by 255 simply.

**Sequence Used:**

We used all the 30 images in the sequence for CNN + GRU combination and alternating frames starting at zero for Conv 3-D as they seem to give the best results after experimentation. One reason is that GRU layer seemed to require all the time steps to learn the dependencies in the model well.

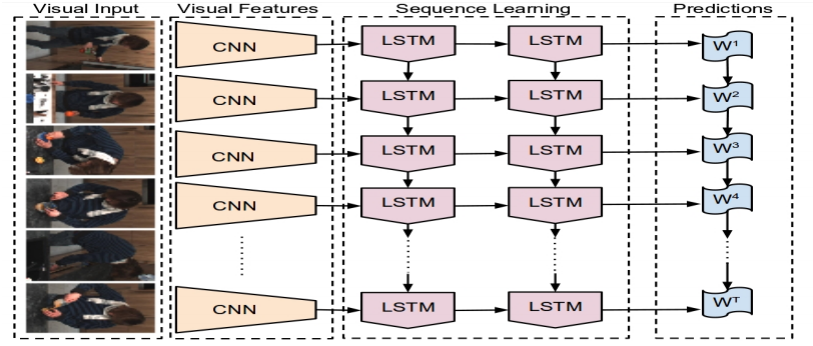
# Image Augmentation

Once the validation accuracy had pretty much **saturated** at around **83 percent** we needed to do some kind of image augmentation to increase the number of training examples so that validation accuracy can be improved and can come nearer to the **training accuracy of 91** percent. Here is what we applied according to the semantic context of the dataset:

1. CropAndPad 2. CropToFixedSize(position=center)  
3. Perspective transform(0.1)  
4. Log Contrast gain(1)  
5. Gaussian Blur(0.25)  
6. Multiply(1.25)  
7. Additive Poisson Noise(lam = 4)

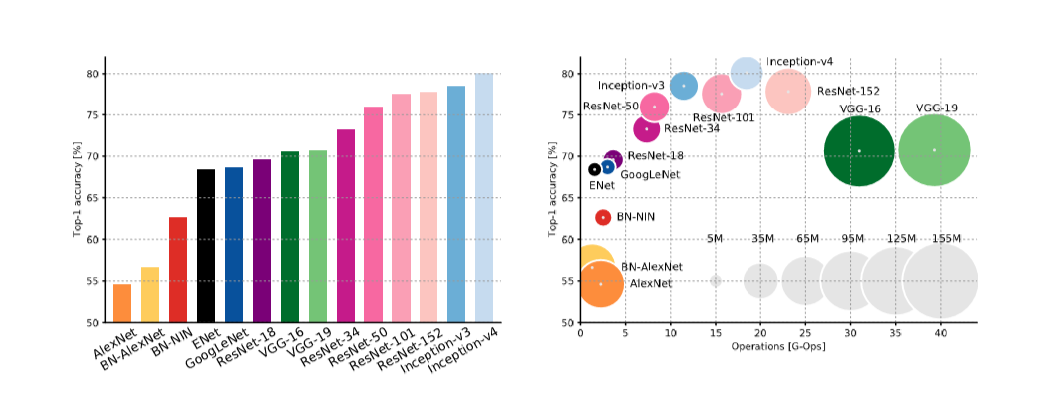
We got a significant jump in validation accuracy after creating more sequences by using image augmentation techniques. The validation accuracy became **90 percent** after image augmentation.

# LRCN model



LRCN is a model in which a time distributed CNN acts as a feature extractor for the LSTM layer(GRU in our case) and passes on the feature over each time step to the LSTM cells. Once the LSTM learn the sequence the final output at last timestep is driven through a dense layer with 5 neurons and a softmax activation to get the prediction.

## Variants Tried and Corresponding Experiments



The reason to choose **Resnet** and **Inception V3** is because they have lesser number of parameters and operations as compared to the accuracy they give.

### Resnet + GRU :

First network that we tried was Resnet and a GRU combination.

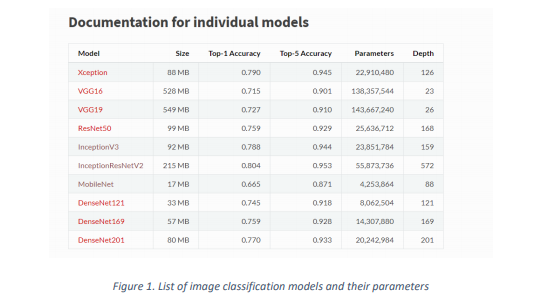
**Experiments:**

1. **Training CNN layers**: Firstly we tried by freezing all the layers of CNN and trying to train only the GRU and fully connected layers of the model. This did not give us very good accuracy with validation loss being stuck around 1.7 and accuracy around 25 percent. On training the convolution layer as well the accuracy did improve to around 30 percent.
2. **Learning Rate**: We started off with a learning rate of 0.01 and ran it for around 50 epochs. But since this did not give very good results we moved on to a learning rate of 0.1 to achieve faster convergence but that did not give very good results as well with accuracy still languishing around 25 percent for the frozen layers and 30 percent when unfreezing it.
3. **Batch Size**: A larger batch size of 10 did not lead to convergence even after 50 epochs. When the batch size was reduced to 3 we got better convergence but still the accuracy languished around 35 percent for the Resnet based model.
4. **Image Size**: We tried with minimum image size of Resnet which is 197 x 197 but overall with resnet even after increasing the image size result were not appropriate as accuracy was around 30 percent and validation loss was around 30 percent.
5. **Optimizer**: We tried **Adam, Nadam** and **SGD** optimisers. Out of the three SGD gave us early convergence and better results so we went ahead with SGD for rest of the iterations.
6. **Reduce LR on Plateau**: We started with the **val\_loss** as reduce LR criterion. Then we tried to use the combination of **categorical\_accuracy** and **val\_loss** but finally best result was obtained on using only the **categorical\_accuracy** as the criterion.
7. **Model Architecture**: We used TIme Distributed Resnet + one **GRU** layer with 1028 cells followed by a dense layer with 5 neurons and a softmax activation.
8. **Trainable Layers**

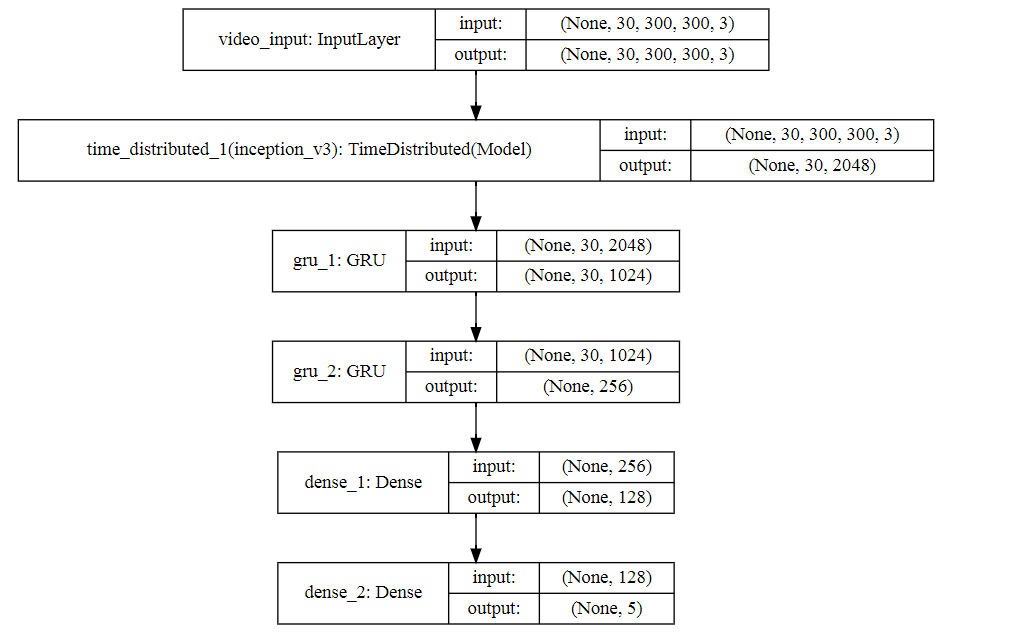
**End Result + Verdict**: The model with Resnet and CNN was undefitting the data so we decided to try **Inception V3 with more GRU as well as Dense layers.**

### Inception V3 + GRU :

Second network that we tried was Time Distributed Google Inception V3 and a GRU combination. The reason to use this network was better **accuracy to parameters** ratio and better **accuracy to operations** ratio.



Following was the model architecture:



**Experiments:**

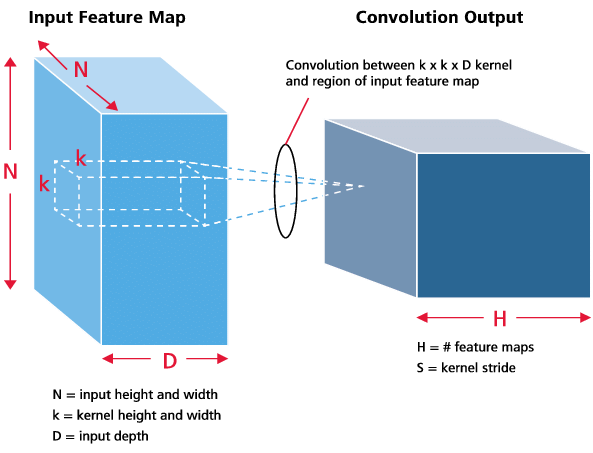
1. **Model Architecture**: Initially we started of with one GRU layer followed by a fully connected layer with 5 neurons but that did not give very good results with validation loss of around 1.2 and accuracy languishing at 40 percent. But on adding one more GRU and dense layer state of the art results were obtained with both validation and training accuracy going above **90 percent**.
2. **Training CNN layers**: In the case of Inception V3 freezing the layers did not help in terms of accuracy so we trained all the layers of CNN. On doing this we were able to get decent accuracy of about 95 percent on training data and around 86 percent on the validation data. The validation loss went as down as 0.59.
3. **Learning Rate**: As mentioned earlier the learning rate of 0.01 with SGD optimiser gave good results on the problem.
4. **Batch Size**: Initially we started with a batch size of one to achieve early convergence and see if the model is working at all. Once that was achieved but the results were still suboptimal with the validation accuracy languishing around 76 percent, we tested it with batch size of 2,3,4 and 3 gave us the best result out of all with validation accuracy going up to 83 percent.
5. **Image Size**: Initially image size 0f 120,120 did not give very good results with accuracy languishing around 30 percent and loss around 1.5. But after research and ablation we discovered that increasing image size does add to the accuracy. Thus we got state of the art results with an image size of 300x300 with 86 percent validation accuracy and 95 percent training accuracy.
6. **Optimizer**: We tried **Adam, Nadam** and **SGD** optimisers. Out of the three SGD gave us early convergence and better results so we went ahead with SGD for rest of the iterations.
7. **Dropouts**: Dropouts added a lot of value in terms of increasing the validation accuracy and preventing overfitting. Firstly, we tried adding the dropout of 20 percent on each of the GRU layers as well as both dense layers. But this dented the performance a bit too much by lowering both the validation and training accuracy down to nearly 80 percent. On restricting the drop out to 20 percent on the two GRU layers we got state of the art results with around **90 percent** validation accuracy and **91 percent** training accuracy with validation loss nearing almost **0.2**.
8. **Gradient Clipping**: The gradient clipping via a clipnorm of 0.7 helped along with Dropouts to get excellent validation accuracy.
9. **Number Of Epochs**: Initially with a batch size of 1 even 10 epochs were sufficient to achieve good fit. But with a batch size of 3 around 20 epochs were required to get the best results.
10. **Reduce LR on Plateau**: Reduce LR on plateau on the categorical accuracy with a factor of 0.2 and patience of 5 worked best the best for the model by modifying the learning rate on stagnation. Earlier we tried with a reduce LR on plateau on validation loss with a factor of 0.1 but the loss started to stagnate after a point and thus we went with the factor of 0.2 and with categorical accuracy as criterion to achieve the best possible results.
11. **Trainable Layers**: For the best possible results we needed to train all the layers of CNN as well as otherwise validation accuracy was languishing around 60 percent.

**End Result + Verdict:** Finally **CNN + GRU model with Inception V3** gave state of the art results with both validation and training **accuracy** above **90 percent** and **loss** of **~0.2**.

# 

# P.T.O

# 3-D Convolution



**Preliminary Over fitting:**

Considered 30 sequences from training to achieve overfitting with the following parameters:

**Batch\_size:** 2

**Frames per sequence:** 15

**Image size:** 120 by 120

**Conv3D layers**: 4 (32, 64, 64, 128 filters)

Dense layer before the end (128)

Dense layer at the end (Softmax with 5 classes)

100% training accuracy was achieved after 20 epochs.

**Model tuning:**

**Image size**: The image size was gradually reduced from 120 by 120 to see when the accuracy was starting to diminish. The accuracy remained comparable until about 60 by 80 size.

**Number of frames per sequence**: The accuracy remained comparable from 30 frames per sequence to 10 frames per sequence. Below that the accuracy deteriorated. However, in ordered to have 4 Conv3D layers and 3 Max pooling layers, we need 17 frames per sequence.

**Number of layers**: 4. Anything lesser than that was not able to extract enough features. We were not able to get more than 65% accuracy with less than 4 layers.

Number of filters: 32, 64, 128, 256 in first, second, third and fourth Conv3D layers respectively. More filters yielded more features and hence higher accuracy.

**Filter Size**: The initial filter size we used was (3,3,3), later we tried smaller (2,2,2) filters and still achieved almost the same results. Thus to keep the number of parameters in control we finalized the filter size as (2,2,2)

**Drop out ratio**: We started with minimal dropout of 0.1 for our architecture the result was a model that was heavily overfitting on the Training dataset. We then increased the dropout to 0.2-0.5 for various the result was a model that was more generic, wasn’t overfitting on training dataset and gave almost same Validation accuracy.

**Dense filters**:

**Batch size**: Owing to limited space on PaperSpace environment, we could not go beyond batch size of 5. For the given model, batch size of 2 gave us more than 70% accuracy.

**ReduceLROnPlateau**: By monitoring val\_loss, we could get validation accuracy of 75% with the above parameters. Patience number of epochs was set to 5 and the reduce factor to 0.3. Any other values took the model longer to converge.

**Batch Normalizing and Gradient Clipping**: To keep the model overfitting in check we also used Batch normalization and Gradient Clipping (0.7).

# Conclusion

In conclusion the CNN + GRU model with Image Augmentation, Regularization(Dropout and gradient clipping) and GRU sequence layers gave very impressive results with **around 90 percent** accuracy and a **loss** of around 0.2 for both training and test datasets.