Table of Contents

[Problem Statement 2](#_Toc3712215)

[**Some important fact about data:** 2](#_Toc3712216)

[Objective: 3](#_Toc3712217)

[**1.** **Generator** 3](#_Toc3712218)

[**2.** **Model** 3](#_Toc3712219)

[**3.** **Output** 3](#_Toc3712220)

[Experiment Results & Explanation: 4](#_Toc3712221)

[**Preprocessing:** 4](#_Toc3712222)

[**Crop & Resize** 4](#_Toc3712223)

[**Normalization:** 4](#_Toc3712224)

[**Augmentation:** 5](#_Toc3712225)

[**Training/Model tuning:** 5](#_Toc3712226)

[**Convolution 3D:** 5](#_Toc3712227)

[**Convolution 2D + RNN:** 6](#_Toc3712228)

[**Results:** 6](#_Toc3712229)

[**Convolution 3D:** 7](#_Toc3712230)

[**Convolution 2D + RNN:** 15](#_Toc3712231)

# Problem Statement

A home electronics company that manufactures state of the art smart televisions want to develop a cool feature in the smart-TV that can **recognize five different gestures** performed by the user which will help users control the TV without using a remote. Let's have professor Raghavan introduce you to the problem statement:

The gestures are continuously monitored by the webcam mounted on the TV. Each gesture corresponds to a specific command:

* 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

## **Some important fact about data:**

1. **Each video** is a sequence of **30 frames** (or images).
2. There are **666** videos provided as **training** data and **100** videos provided as **validation** data
3. **all images in a particular video** subfolder have the **same** **dimensions**
4. different videos may have different dimensions. Specifically, videos have two types of dimensions - either **360x360** or **120x160**
5. There are **two csv**(one for train, one for validation) files having path of videos

Our task is to train a model on the 'train' folder which performs well on the 'val' folder as well

Thus, there are two types of architecture commonly used for analysing videos, both explained below.

1. **Convolutions + RNN**
2. **3D Convolutional Network, or Conv3D**

# Objective:

## **Generator**

Write code for generator function so, that it can provide data in batch while training model.

Also, write code for preprocessing image files in generator function

## **Model**

Create models with two architectures

1. Convolution 3D
2. Convolution 2D + RNN
   1. RNN with LSTM
   2. RNN with GRU

## **Output**

Tune model to achieve good accuracy on train as well as validation data

Provide all metric with changes done in model to tune it in the write up

# Experiment Results & Explanation:

## **Preprocessing:**

As there are two different dimensions of frames provided 360 X 360 and 140 X 160, we need to resize and crop images.

We plotted some of the images for both dimensions and tried below operations on them

### **Crop & Resize**

* 1. Crop the images with 360 X 360 dimensions by 20 and 10 pixels one by one:
     + In this case, the images with swipe gestures the object was being cropped . so network might not able to learn the gesture properly. Hence, we drop out this idea
  2. We tried to crop the 120 X 160 images for 20 pixels and make it 120 X 120.
     + In this case too the object to recognize gesture was being removed from pictures.
  3. We trued to rescale/resize the images to 120 X120.
     + In this case the 120 X 140 images was not looking great and seemed it might not work well with the model
  4. **Final Approach for crop & resize**
     + We took img[0:120,10:150] for 120 X 140 size images and whole image for 360 X360 images
     + Then resized it to 120 X 120.
     + This images were looking good for both the dimensions

### **Normalization:**

We tried below normalization techniques

1. Image/255 this want giving good results.
2. Thinking that skin generally having more red color, keep red color more in the picture. But this wasn’t improving picture quality for gesture we want to recognize.
3. So, finally we did it with,

Having percentile normalization for each channel. percentile is specific to channel

Formula used:

**finalimage =img - np.percentile(img,15)/ np.percentile(img,85) - np.percentile(img,15)**

### **Augmentation:**

The data is not huge and hence, with existing data we were not able to achieve more than 30% in Conv3D network. Hence, we added flip & affine transform to generate more data.

But same was not true with CNN + RNN. The flip image was decreasing the accuracy. Hence, for CNN + RNN we used only affine transform to generate more data

## **Training/Model tuning:**

We tried with two architectures Convolution 3D and Convolution 2D + RNN

We tried with Ablation first and having 5 epochs only. And then started with full data and 5 up to 25 epochs.

The **batchsize** should be power of 2. And considering GPU limit, we were not able to load lot of data at once… many times got resourceexausted error. Hence, considering the facts, kept the batch size of 8 which is generally used.

We kept the number of neurons(feature maps) in all Convolution layer as power of 2 e.g. 8,16,32, 64 etc…

For Architectures:

only approach is noted with high level explanation. More detailing about networks tried is under Results section.

### **Convolution 3D:**

We started with simple architecture 3 layers, all followed with batchnormalization, activation(relu) and maxpooling 3D.

We added dense layers and drop outs as well after convolution layers.

The kernel sizes tried with (3,3,3), (2,2,2) and (1,3,3).

We tried with ablation first and after achieve some accuracy we tried further experiments with all data.

We tried with four layers as well.

We tried to add more feture maps, add layers, remove batchnormalization, remove dropout, add augmentation, add more epochs etc…

### **Convolution 2D + RNN:**

We started with 5 time distributed conv 2d layers, , activation(relu) and most of them followed by maxpooling 2D.

We added batch normalization, dense layers and drop outs as well after convolution layers.

it also has LSTM layer

The kernel sizes tried with (5,5), (3,3).

Feature map used are 8,16,32,64 and 128

We reduced layers as well and finally only 4 conv 2d layer kept, also we used GRU instead of LSTM to reduce number of trainable parameters.

We also tried data with and without augmentation and running more number of epochs to get the best result.

## **Results:**

**Notes & abbreviation**:

* Conv3d : Convolution 3D layers
* MaxPooling3D(2,2,2) : here (2,2,2) is poolsize
* Dense layers: other than last layer all are having activation=”relu”
* Trial: Experiment
* Categorical CrossEntropy loss is used for all
* Metrics used are: categorical accuracy and validation accuracy to tune network
* Result mentioned in table has below format:
  + Categorical loss
  + Validation loss
  + Categorical accuracy
  + Validation accuracy
* Result is displaying the best result with logic having highest categorical and then validation accuracy from all epochs.
  + Its accuracy order by categorical, validation

When accuracy improved from previous experiment either categorical or validation. We make that cell background **Green else blue**

### **Convolution 3D:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Trial** | **Network details**  **First row has detailing, the corresponding rows having only the change made to previous row** | **Data Operations** | **Result (Epoch, datasize,**  **loss, validation loss,**  **categorical accuracy ,validation accuracy)** | **Decision + Explanation** |
| **1** | 1)  Conv3d: 8 maps, kernel(3,3,3)  Batchnormalization  Activation(relu)  Maxpooling3D(2,2,2), strides(1,2,2)  2)  Conv3d: 16 maps, kernel(3,3,3)  Batchnormalization  Activation(relu)  Maxpooling3D(2,2,2), strides(2,2,2)  3)  Conv3d: 16 maps, kernel(3,3,3)  Batchnormalization  Activation(relu)  Dropout(0.25)  Maxpooling3D(2,2,2)  4) Flatten  5) Dense(32), Dropout(0.25)  6) Dense(16), Dropout(0.5)  7) Dense(5), activation=softmax  Batch size: 8 | Crop, resize & normalization | Epoch:5  Train ablation=300  Val ablation=30  loss: 8.9757  val\_loss: 12.8118  accuracy: 0.2380  val\_accuracy: 0.2051 | We started with 5 epoch and ablation so we can see the results quickly and tune the model… once model performance at acceptable level. We’ll run it for all data and then if it works well, we’ll run it with more epochs.  From work experience we always start with batch size 8 then 16 then 32 and so on… or if we have very good configurations then we start with 32 batch size. To check if memory management is good.  Batch size kept to 8 considering that the GPU might not perform well with more data and batch size kept in power of 2 as taught in one of the lecture. |
| **2** | The feature maps for all the Conv3D layers changed from [8,16,32] to [16,32,64]  Dense layers changed from 32 to 64 and 16 to 32 | Crop, resize & normalization | Epoch:5  Train ablation=300  Val ablation=30  loss: 12.5443  val\_loss: 12.1919  accuracy: 0.2199  val\_accuracy: 0.2436 | In previous model we were able to see the loss is decreasing and accuracy is improving.  But the results are not very good.  Hence, we added more features to each layer.  Thinking more features extraction will help network learning more things about image and as a result it might perform better  The accuracy here is decreased but validation accuracy improved. |
| **3** | The feature maps for all the Conv3D layers changed from [16,32,64] to [32,64,128]  Dense layers changed from 64 to 128 and 32 to 64 | Crop, resize & normalization | Epoch:5  Train ablation=300  Val ablation=30  loss: 12.8168  val\_loss: 12.1919  accuracy: 0.2048  val\_accuracy: 0.2436 | In previous experiment the accuracy decreased but validation accuracy improved.  Hence, we thought to try with more features , to see if they are decreasing accuracy or it’ll improve.  But again accuracy is decreased. |
| **4** | Same network as experiment-1. The only change is **removed drop out** after last convolution layer.  So, now the network is as below:  1)  Conv3d: 8 maps, kernel(3,3,3)  Batchnormalization  Activation(relu)  Maxpooling3D(2,2,2), strides(1,2,2)  2)  Conv3d: 16 maps, kernel(3,3,3)  Batchnormalization  Activation(relu)  Maxpooling3D(2,2,2), strides(2,2,2)  3)  Conv3d: 32 maps, kernel(3,3,3)  Batchnormalization  Activation(relu)  Maxpooling3D(2,2,2)  4) Flatten  5) Dense(32), Dropout(0.25)  6) Dense(16), Dropout(0.5) | Crop, resize & normalization | Epoch:5  All Data  loss: 11.6942  val\_loss: 13.0185  accuracy: 0.2231  val\_accuracy: 0.1923 | Adding more featuremaps in convolution layer was decreasing accuracy. Best we got with experiment 1 itself.  Hence, we kept the network same as first experiment and only removed dropout after last layer. The reason to do this is… may be dropping out connections is decreasing accuracy and we should use all connections from last convolution layer.  We also, thought to run the network this time with all data. Thinking the ablation might not be giving good results.  Still accuracy was not improving. In fact it went down. |
| **5** | Same network as previous experiment. The only change is to **add one more convolution layer**  So, now the network is as below:  1)  Conv3d: 8 maps, kernel(3,3,3)  Batchnormalization  Activation(relu)  Maxpooling3D(2,2,2), strides(1,2,2)  2)  Conv3d: 16 maps, kernel(3,3,3)  Batchnormalization  Activation(relu)  Maxpooling3D(2,2,2), strides(2,2,2)  3)  Conv3d: 32 maps, kernel(3,3,3)  Batchnormalization  Activation(relu)  Maxpooling3D(2,2,2)  4)  Conv3d: 64 maps, kernel(3,3,3)  Conv3d: 64 maps, kernel(3,3,3)  Batchnormalization  Activation(relu)  Maxpooling3D(2,2,2)  5) Flatten  6) Dense(32), Dropout(0.25)  7) Dense(16), Dropout(0.5) | Crop, resize & normalization | Epoch:5  Train ablation=400  Val ablation=40  loss: 12.6062  val\_loss: 12.5537  accuracy: 0.2123  val\_accuracy: 0.2212 | Based on previous experiments,  Adding more features, removing dropout or using all data did not improve accuracy.  Hence, we thought to add one more layer.  And work on ablation but this time increased ablation count to train:400 and val:40  Still, we did not get improvement in accuracy. |
| **6** | RemovedBatchnormalization, Activation(relu),Maxpooling3D(2,2,2) between Conv layer 3 and Conv layer 4 | Crop, resize & normalization | Epoch:5  Train ablation=400  Val ablation=40  **Epoch:4**  loss: 12.5974  val\_loss: 13.0185  accuracy: 0.2123  val\_accuracy: 0.1923  **Epoch:5**  loss: 13.2067  val\_loss: 11.9336  accuracy: 0.1792  val\_accuracy: 0.2596 | Based on previous experiments,  Adding more features, adding layers, removing dropout or using all data did not improve accuracy.  We googled and looked at some conv3D networks. And we found people have used two conv3D layers in sequence and then batchnormalization.  Hence, we removed the batchnormalization, activation and maxpooling between layer -3 & layer-4  As we can see here in fifth epoch we got validation accuracy increased. Though the categorical\_accuracy was decreased… |
| **7** | Went back to architecture of experiment -5  But changed the filter for first convolution layer to (5,5,5) And filter of later layer from (3,3,3) to (2,2,2)  So, now the network is as below:  1)  Conv3d: 8 maps, kernel(5,5,5)  Batchnormalization  Activation(relu)  Maxpooling3D(2,2,2), strides(1,2,2)  2)  Conv3d: 16 maps, kernel(3,3,3)  Batchnormalization  Activation(relu)  Maxpooling3D(2,2,2), strides(2,2,2)  3)  Conv3d: 32 maps, kernel(3,3,3)  Batchnormalization  Activation(relu)  Maxpooling3D(2,2,2)  4)  Conv3d: 64 maps, kernel(2,2,2)  Batchnormalization  Activation(relu)  Maxpooling3D(2,2,2)  5) Flatten  6) Dense(32), Dropout(0.25)  7) Dense(16), Dropout(0.5) | Crop, resize & normalization | Epoch:5  Train ablation=400  Val ablation=40  loss: 13.0110  val\_loss: 13.3284  accuracy: 0.1928  val\_accuracy: 0.1731 | Still not much improvement in accuracy.  We thought to change the filter this time.  Considering that the object is occupying good amount of space in image. We can keep first layer filter as (5,5,5) instead (3,3,3)  And filter of later layer from (3,3,3) to (2,2,2)  Now, the accuracy has decreased a lot… so this seemed the worst network.  So, we felt that we need to do augmentation. The more data might improve results. |
| **8** | We added augmentation for data | Crop, resize & normalization  **+**  Flip & Affine transform | Epoch:5  Train ablation=400  Val ablation=40  loss: 0.8602  val\_loss: 0.8550  accuracy: 0.6571  val\_accuracy: 0.6955 | Yay!!!  We got good accuracy improvement this time.  The augmentation was must for Conv3D architecture.  But the accuracy was still not in acceptable range.  Hence, we thought to do more experiments with all data to see accuracy on whole dataset. |
| 9 | Ran same network with all data instead ablation | Crop, resize & normalization  **+**  Flip & Affine transform | Epoch:5  All Data  loss: 1.3285  val\_loss: 1.7207  accuracy: 0.4511  val\_accuracy: 0.4744 | Accuracy is decreased but it is not bad as earlier models.  Hence, augmentation is working.  We need to tune some other things now. |
| 10 | Removed forth convolution layer.  Also, changed kernel size of third convolution later from (3,3,3) to (1,3,3) | Crop, resize & normalization  **+**  Flip & Affine transform | Epoch:5  All Data  loss: 1.3555  val\_loss: 2.6211  accuracy: 0.5489  val\_accuracy: 0.3814 | Earlier we saw adding more layers was not improving accuracy.  Hence, we thought to add forth layer we added in experiment-5.  We thought to change the kernel size of last layer to extract more detailed features  The categorical accuracy improved here. But validation accuracy went down. |
| 11-13 | Removed batch normalization from layer-1 and layer-2 | Crop, resize & normalization  **+**  Flip & Affine transform | Epoch:5  All Data  **epoch:4**  loss: 1.0327  val\_loss: 1.2022  accuracy: 0.5812  val\_accuracy: 0.5577  **epoch:5**  loss: 1.0535  val\_loss: 0.9313    accuracy: 0.5797  val\_accuracy: 0.6314 | Earlier we saw that removing batchnormalization from one layer was not improving accuracy but did not decreased it too.  Hence, we thought to remove batchnormalization from layer -1 and layer -2  We got again that removing batchnormalization improves validation accuracy.  Also, categorical accuracy improved from previous experiment |
| 12 | Changed normalization of images to image/255. | Crop, resize & normalization  **+**  Flip & Affine transform | Epoch:5  All Data  loss: 1.8379  val\_loss: 1.5345  accuracy: 0.2324  val\_accuracy: 0.3750 | Now, we thought to change normalization to improve accuracy. We thought may be percentile is not working well. Dividing image by 255 keeps image in same way and reduces the pixel values only.  Hence, tried with normalized\_image = image/255.  But, here with each epoch the accuracy was decreasing and loss was increasing. |
| 13 | Ran network from Experiment -11 with 15 epoch. | Crop, resize & normalization  **+**  Flip & Affine transform | Epoch:15  loss: 0.3922  val\_loss: 0.6091  accuracy: 0.8474  val\_accuracy: 0.8103 | Obviously, normalization changes didn’t work well.  Hence, we changed the normalization formula to earlier one  So, we took same network from experiment-11 and ran it with 15 epoch.  This time we got acceptable accuracy. |
| 14 | Take alternate frame from videos and run with same network architecture.  Also, increased epochs to 25 | Crop, resize & normalization  **+**  Flip & Affine transform | Epoch:25  loss: 0.3152  val\_loss: 0.4300  accuracy: 0.8942  val\_accuracy: 0.8667 | We thought to experiment with frames this time.  We saw that if we take alternate frame the video is still good. That will use less memory will be faster to process as well.  So we tried previous network with alternate frame. And 25 epochs.  And we got really good results… the accuracy is almost **90%** and validation accuracy is also almost **87%**. That is good accuracy till now. |

We stopped at 14th experiment with Convolution 3D. and started with Conv 2D + RNN Network architecture.

### **Convolution 2D + RNN:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Trial** | **Network details**  **First row has detailing, the corresponding rows having only the change made to previous row** | **Data Operations** | **Result (Epoch,**  **loss,  validation loss,**  **categorical accuracy ,validation accuracy)** | **Decision + Explanation** |
| **1** | 1. TimeDistributed(Conv2d): 8 maps, kernel(5,5), Stride(2,2), Activation(relu) 2. TimeDistributed(Conv2d): 8 maps, kernel(3,3), Activation(relu) kernel\_initializer(he\_normal)) 3. TimeDistributed(Maxpooling2D(2,2), strides(2,2)) 4. TimeDistributed(Conv2d): 16 maps, kernel(3,3)), Activation(relu) ) 5. TimeDistributed(Maxpooling2D(2,2), strides(2,2)) 6. TimeDistributed(Conv2d): 32 maps, kernel(3,3), Activation(relu)) 7. TimeDistributed(Maxpooling2D(2,2), strides(2,2)) 8. TimeDistributed(Conv2d): 64 maps, kernel(2,2), Activation(relu)) 9. TimeDistributed(Maxpooling2D(2,2), strides(2,2)) 10. TimeDistributed(BatchNormalization) 11. Dropout(0.25) 12. TimeDistributed(Flatten) 13. Dense(128) 14. Dropout(0.25) 15. Dense(64) 16. Dropout(0.5) 17. LSTM(256) 18. Dense(5), activation=softmax | Crop, resize & normalization | Epoch:5    loss: 1.1045  val\_loss: 1.1340    accuracy: 0.5762  val\_accuracy: 0.5000 | We started with 5 epoch. We’ll run it for all data and then if it works well, we’ll run it with more epochs.  From work experience we always start with batch size 8 then 16 then 32 and so on… or if we have very good configurations then we start with 32 batch size. To check if memory management is good.  Batch size kept to 16 by gaining the experience from previous conv 3d model experience. |
| 2 | Layer 2 has been removed  Dropout reduced from 0.5 to 0.25 in Layer 16  New network   1. TimeDistributed(Conv2d): 8 maps, kernel(5,5), Stride(2,2), Activation(relu) 2. TimeDistributed(Maxpooling2D(2,2), strides(2,2)) 3. TimeDistributed(Conv2d): 16 maps, kernel(3,3)), Activation(relu) ) 4. TimeDistributed(Maxpooling2D(2,2), strides(2,2)) 5. TimeDistributed(Conv2d): 32 maps, kernel(3,3), Activation(relu)) 6. TimeDistributed(Maxpooling2D(2,2), strides(2,2)) 7. TimeDistributed(Conv2d): 64 maps, kernel(2,2), Activation(relu)) 8. TimeDistributed(Maxpooling2D(2,2), strides(2,2)) 9. TimeDistributed(BatchNormalization) 10. Dropout(0.25) 11. TimeDistributed(Flatten) 12. Dense(128) 13. Dropout(0.25) 14. Dense(64) 15. Dropout(0.25) 16. LSTM(256) 17. Dense(5), activation=softmax | Crop, resize & normalization | Epoch:7    loss: 0.9320  val\_loss: 1.1607    accuracy: 0.6598  val\_accuracy: 0.6429 | Number of layer has been reduced to do some experimentation.  Drop out reduced from 0.5 to 0.25, the reason to do this is… may be dropping out connections is decreasing accuracy and we should use all connections from last convolution layer. |
| **3** | No Change in network, Augmentation added | Crop, resize & normalization  **+**  Flip & Affine transform | Epoch:7  loss: 0.9336  val\_loss: 1.1067    accuracy: 0.5775  val\_accuracy: 0.6395 | We added augmentation to increase the data size so, model can be trained with more data.  But this reduced accuracy. |
| **4** | 16th layer changed from LSTM to GRU | Crop, resize & normalization  **+**  Flip & Affine transform | Epoch:7    loss: 0.6245  val\_loss: 0.7559    accuracy: 0.7550  val\_ accuracy: 0.6871 | We thought to reduce the complexity of network by using GRU instead of LSTM, also considering the realtime result it would be better to have less number of trainable parameters in network.  This improved the accuracy and using GRU instead of LSTM is right decision. |
| **5** | No Change in network, augmentation is removed | Crop, resize & normalization | Epoch:7    loss: 0.7917  val\_loss: 0.5524    accuracy: 0.7427  val\_ accuracy: 0.7347 | Experiments are done using removing augmentation, this give better results as both validation and normal accuracy are closer. |
| **6** | New Timedistributed conv2d layer added with 128 feature maps and maxpooling added after that  **New network**   1. TimeDistributed(Conv2d): 8 maps, kernel(5,5), Stride(2,2), Activation(relu) 2. TimeDistributed(Maxpooling2D(2,2), strides(2,2)) 3. TimeDistributed(Conv2d): 16 maps, kernel(3,3)), Activation(relu) ) 4. TimeDistributed(Maxpooling2D(2,2), strides(2,2)) 5. TimeDistributed(Conv2d): 32 maps, kernel(3,3), Activation(relu)) 6. TimeDistributed(Maxpooling2D(2,2), strides(2,2)) 7. TimeDistributed(Conv2d): 64 maps, kernel(2,2), Activation(relu)) 8. TimeDistributed(Maxpooling2D(2,2), strides(2,2)) 9. TimeDistributed(Conv2d): 128 maps, kernel(2,2), Activation(relu)) 10. TimeDistributed(Maxpooling2D(2,2), strides(2,2)) 11. TimeDistributed(BatchNormalization) 12. Dropout(0.25) 13. TimeDistributed(Flatten) 14. Dense(128) 15. Dropout(0.25) 16. Dense(64) 17. Dropout(0.25) 18. GRU(256) 19. Dense(5), activation=softmax | Crop, resize & normalization | Epoch:7    loss: 0.9372  val\_loss: 1.2150    accuracy: 0.6175  val\_ accuracy: 0.5102 | Some more experiments by adding new layer with 128 feature maps  but its bad idea as making network more complex is reducing the accuracy. |
| **7** | **Removed the added layers in last step (i.e. conv2d with 128 feature maps and max pooling)**  **Added back**  TimeDistributed(Conv2d): 8 maps, kernel(3,3), Activation(relu) kernel\_initializer(he\_normal)) | Crop, resize & normalization | Epoch:7    loss: 0.7028  val\_loss: 0.5433    accuracy: 0.7159  val\_ accuracy: 0.7551 | Removed the layer added with 128 feature maps and added back the layer at start of the network to extract more feature.  But doing all this experiments is not increasing accuracy more then 75 % ☹ |
| **8** | **Removed layer 2**  TimeDistributed(Conv2d): 8 maps, kernel(3,3), Activation(relu) kernel\_initializer(he\_normal))  **Filter changed for layer 1 to (3,3) from (5,5)**  **GRU feature map changed from 256 to 128**   1. TimeDistributed(Conv2d): 8 maps, kernel(3,3), Stride(2,2), Activation(relu) 2. TimeDistributed(Maxpooling2D(2,2), strides(2,2)) 3. TimeDistributed(Conv2d): 16 maps, kernel(3,3)), Activation(relu) ) 4. TimeDistributed(Maxpooling2D(2,2), strides(2,2)) 5. TimeDistributed(Conv2d): 32 maps, kernel(3,3), Activation(relu)) 6. TimeDistributed(Maxpooling2D(2,2), strides(2,2)) 7. TimeDistributed(Conv2d): 64 maps, kernel(2,2), Activation(relu)) 8. TimeDistributed(Maxpooling2D(2,2), strides(2,2)) 9. TimeDistributed(Conv2d): 128 maps, kernel(2,2), Activation(relu)) 10. TimeDistributed(Maxpooling2D(2,2), strides(2,2)) 11. TimeDistributed(BatchNormalization) 12. Dropout(0.25) 13. TimeDistributed(Flatten) 14. Dense(128) 15. Dropout(0.25) 16. Dense(64) 17. Dropout(0.25) 18. GRU(128) 19. Dense(5), activation=softmax | Crop, resize & normalization | Epoch:7    loss: 0.4983  val\_loss: 0.4523    accuracy: 0.8155  val \_accuracy: 0.8367 | Let’s do some more changes   1. Get rid of added layer at the start 2. Change filter from (5,5) to (3,3) to extract more feature at the start 3. Reduce the complexity of GRU i.e. feature map of 128 from 256   And guess what we can see increased accuracy and we are now in 80% range.  So, overall by reducing the complexity accuracy is getting increased. |
| **9** | Ran previous network with 25 epoch | Crop, resize & normalization | Epoch:25    loss: 0.1755  val\_loss: 0.0886    accuracy: 0.9603  val\_accuracy: 0.9796 | As accuracy was better with the model we decided it to run for larger epochs and get the best result at epoch 17  So, we are able to get the model with more than 95 % accuracy as we trained model with more epochs. |
| **10** | **Added affine transform** | **Crop, resize & normalization**  **+**  **Affine transform** | **Epoch:25**    **loss: 0.0454**  **val\_loss: 0.0063**    **accuracy: 0.9960**  **val\_ accuracy: 1.0000** | **We hit 100%**  **Overall we thought to keep augmentation as it may improve the test accuracy.**  **There was debate over which model we should pick a model which gave us accuracy of close to 97-98% or the model which is giving us accuracy of more than 99%, as we did not have test data we finally used the model which is giving us the best result and did not considered that this model could be overfitting.** |