Deep Learning
Gesture Recognition

-Sudarshan SoundaraPandian

Problem Statement

Imagine you are working as a data scientist at a home electronics company which manufactures state of the art smart televisions. You want to develop a cool feature in the smart-TV that can recognise five different gestures performed by the user which will help users control the TV without using a remote.

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

Each video is a sequence of 30 frames (or images).

Understanding the Dataset

The training data consists of a few hundred videos categorized 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 standardize 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.

Objective

Our task is to train a model on the 'train' folder which performs well on the 'val' folder as well (as usually done in ML projects). We have withheld the test folder for evaluation purposes - our final model's performance will be tested on the 'test' set.

Given Data

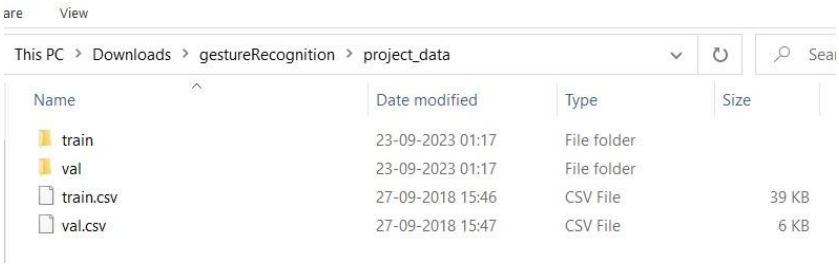
To get started with the model building process, we first need to get the data on your storage.

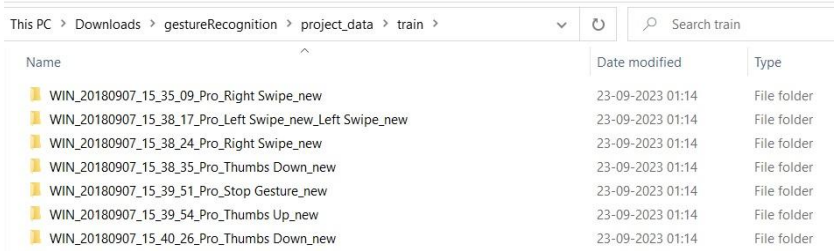
In order to get the data on the storage, perform the following steps in order

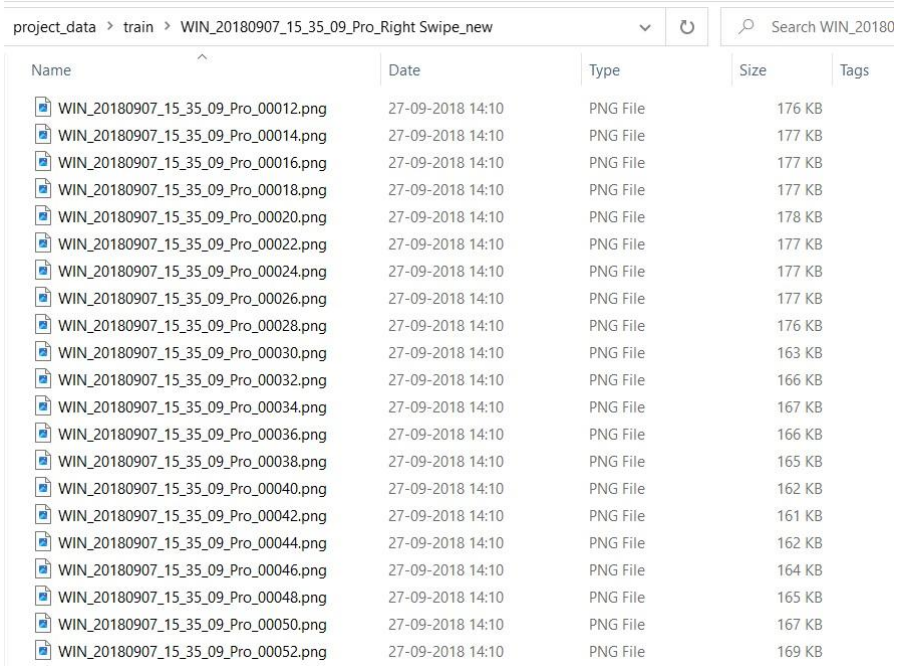
1. Open the terminal
2. g0 down https://drive.google.com/uc?id=1ehyrYBQ5rbQQe6yL4XbLWe3FMvuVUGiL
3. unzip Project_data.zip

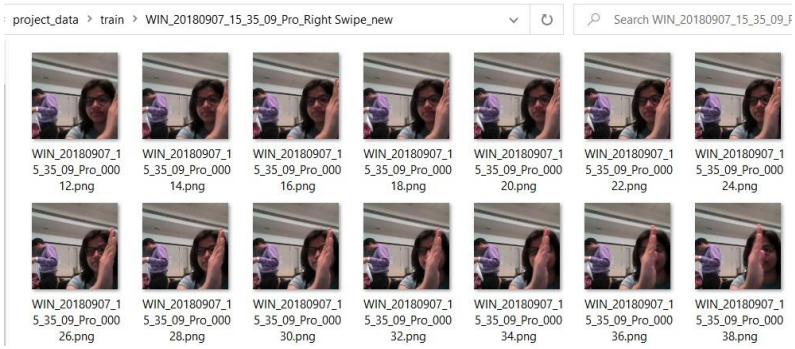
Now that we have got the data on the storage, let's look at the different choices of architectures you can use.

Following are the pictures of the extracted data folder.






project_data > train > WIN_20180907_15_35_09_Pro_Right Swipe_new
Search WIN_20180
Name
Date
Type
Size
Tags
WIN_20180907_15_35_09_Pro_00012.png 27-09-2018 14:10 PNG File 176 KB
WIN_20180907_15_35_09_Pro_00014.png 27-09-2018 14:10 PNG File 177 KB
WIN_20180907_15_35_09_Pro_00016.png 27-09-2018 14:10 PNG File 177 KB
WIN_20180907_15_35_09_Pro_00018.png 27-09-2018 14:10 PNG File 177 KB
WIN_20180907_15_35_09_Pro_00020.png 27-09-2018 14:10 PNG File 178 KB
WIN_20180907_15_35_09_Pro_00022.png 27-09-2018 14:10 PNG File 177 KB
WIN_20180907_15_35_09_Pro_00024.png 27-09-2018 14:10 PNG File 177 KB
WIN_20180907_15_35_09_Pro_00026.png 27-09-2018 14:10 PNG File 177 KB
WIN_20180907_15_35_09_Pro_00028.png 27-09-2018 14:10 PNG File 176 KB
WIN_20180907_15_35_09_Pro_00030.png 27-09-2018 14:10 PNG File 163 KB
WIN_20180907_15_35_09_Pro_00032.png 27-09-2018 14:10 PNG File 166 KB
WIN_20180907_15_35_09_Pro_00034.png 27-09-2018 14:10 PNG File 167 KB
WIN_20180907_15_35_09_Pro_00036.png 27-09-2018 14:10 PNG File 166 KB
WIN_20180907_15_35_09_Pro_00038.png 27-09-2018 14:10 PNG File 165 KB
WIN_20180907_15_35_09_Pro_00040.png 27-09-2018 14:10 PNG File 162 KB
WIN_20180907_15_35_09_Pro_00042.png 27-09-2018 14:10 PNG File 161 KB
WIN_20180907_15_35_09_Pro_00044.png 27-09-2018 14:10 PNG File 162 KB
WIN_20180907_15_35_09_Pro_00046.png 27-09-2018 14:10 PNG File 164 KB
WIN_20180907_15_35_09_Pro_00048.png 27-09-2018 14:10 PNG File 165 KB
WIN_20180907_15_35_09_Pro_00050.png 27-09-2018 14:10 PNG File 167 KB
WIN_20180907_15_35_09_Pro_00052.png 27-09-2018 14:10 PNG File 169 KB


project_data > train > WIN_20180907_15_35_09_Pro_Right Swipe_new
Search WIN_20180907_15_35_09_F
WIN_20180907_15_35_09_Pro_00012.png
WIN_20180907_15_35_09_Pro_00014.png
WIN_20180907_15_35_09_Pro_00016.png
WIN_20180907_15_35_09_Pro_00018.png
WIN_20180907_15_35_09_Pro_00020.png
WIN_20180907_15_35_09_Pro_00022.png
WIN_20180907_15_35_09_Pro_00024.png
WIN_20180907_15_35_09_Pro_00026.png
WIN_20180907_15_35_09_Pro_00028.png
WIN_20180907_15_35_09_Pro_00030.png
WIN_20180907_15_35_09_Pro_00032.png
WIN_20180907_15_35_09_Pro_00034.png
WIN_20180907_15_35_09_Pro_00036.png
WIN_20180907_15_35_09_Pro_00038.png


train.csv
A
B
C
WIN_20180925_17_Left_Swipe_new 0

Suggested Architectures

After understanding and acquiring the dataset, the next step is to try out different architectures to solve this problem. There are about 2 different architectures that were suggested for a deep learning problem like this.

For analyzing videos using neural networks, **two types of architectures** are used commonly. One is the standard **CNN + RNN architecture** in which you pass the images of a video through a CNN which extracts a feature vector for each image, and then passes the sequence of these feature vectors through an RNN. This is something you are already familiar with (in theory).The other popular architecture used to process videos is a natural extension of CNNs - a **3D convolutional network**. In this project, you will try both these architectures.

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

1. **Convolutions + RNN**
   The conv2D network will extract a feature vector for each image, and a sequence of these feature vectors is then fed to an RNN-based network. The output of the RNN is a regular softmax (for a classification problem such as this one).
2. **3D Convolutional Network, or Conv3D**
   3D convolutions are a natural extension to the 2D convolutions you are already familiar with. Just like in 2D conv, you move the filter in two directions (x and y), in 3D conv, you move the filter in three directions (x, y and z). In this case, the input to a 3D conv is a video (which is a sequence of 30 RGB images). If we assume that the shape of each image is 100x100x3, for example, the video becomes a 4-D tensor of shape 100x100x3x30 which can be written as (100x100x30)x3 where 3 is the number of channels. Hence, deriving the analogy from 2-D convolutions where a 2-D kernel/filter (a square filter) is represented as (fxf)xc where f is filter size and c is the number of channels, a 3-D kernel/filter (a 'cubic' filter) is represented as (fxfxf)xc (here c = 3 since the input images have three channels). This cubic filter will now '3D-convolve' on each of the three channels of the (100x100x30) tensor.

Model Builder Class

The model builder class has all the architecture and code needed to build a model and this is very useful because we won't be writing the same code again and again. Also it's an abstract class so it's easy to derive and make new models as we will be experimenting by changing a few parameters often

Generator Function

This is the important function which will take a sequence of images as input and it should do that without errors. This is the gateway for our input params getting fed into the model. In the generator, you are going to preprocess the images as you have images of 2 different dimensions as well as create a batch of video frames. You have to experiment with `img_idx`, `y`,`z` and normalization such that you get high accuracy.

Data Preprocessing

The following are the data preprocessing steps that will be done before processing:

1. Image Resize and Cropping
2. Normalization of images
3. Image Augmentation

Test Model Structure And Params

Model: "sequential"
Different filter size , image height and width , batch size and number of epochs are chosen.

_____

Total params: 1736389 (6.62 MB)
Trainable params: 1735525 (6.62 MB)
Non-trainable params: 864 (3.38 KB)

_____

Test Model Observations:

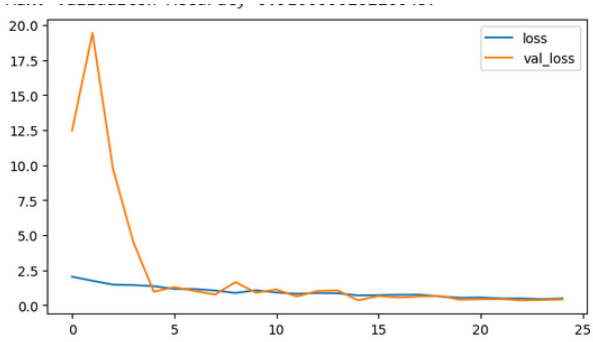
1. Image size of 120,120 seems to be performing well.
2. Bigger batch size is crashing by going over the maximum ram usage. So I decided to keep the batch size to 5.

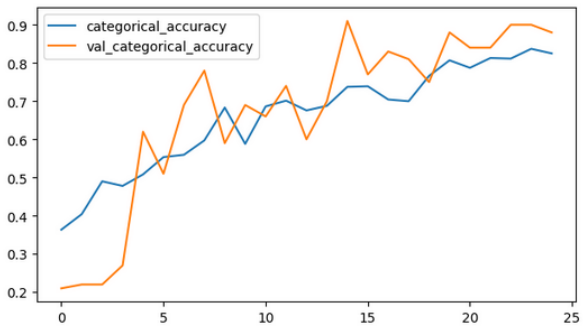
Model 1 : Structure And Params

_____

frames_to_sample=20
batch_size=5
num_epochs=25
image_height=120
image_width=120
dropout=0.50%
Total params: 1762613 (6.72 MB)
Trainable params: 1761109 (6.72 MB)
Non-trainable params: 1504 (5.88 KB)

_____





Model 1 Observations:

- Max. Training Accuracy 0.837104082107544
- Max. Validation Accuracy 0.9100000262260437
- Model seems to be underfitting

Model 2 : Structure And Params

Lets try Model 1 with Augmentation

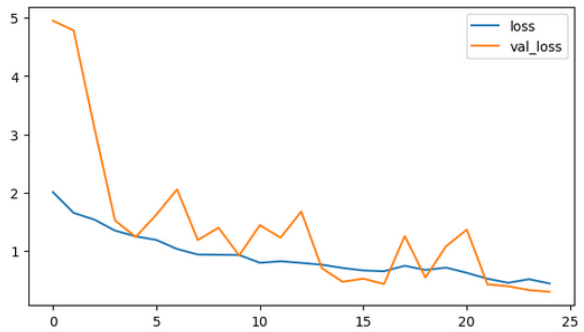
Model: "sequential_8"

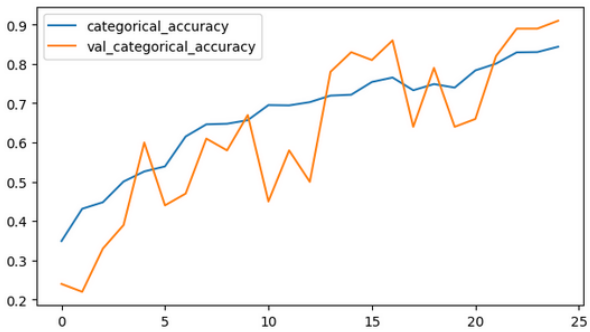
---

frames_to_sample=20
batch_size=5
num_epochs=25
image_height=120
image_width=120
dropout=0.50%
Total params: 1762613 (6.72 MB)

Trainable params: 1761109 (6.72 MB)

Non-trainable params: 1504 (5.88 KB)





Model 2 Observations:

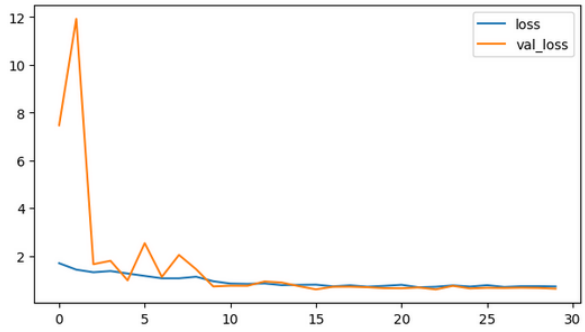
- Max. Training Accuracy 0.837104082107544
- Max. Validation Accuracy 0.9100000262260437
- Good results but still there is too much oscillation on loss during training

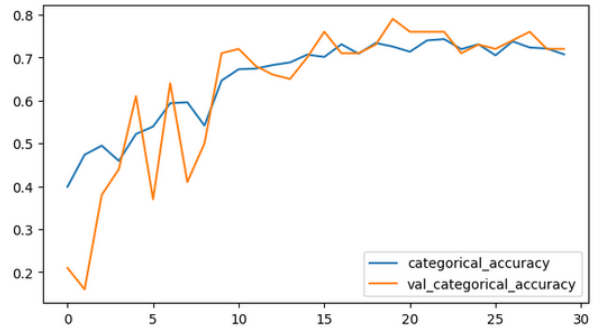
Model 3 : Structure And Params

Lets reduce frame samples and train for longer periods

---

frames_to_sample=16
batch_size=5
num_epochs=30
image_height=120
image_width=120
dropout=0.25%
Total params: 909637 (3.47 MB)
Trainable params: 908645 (3.47 MB)
Non-trainable params: 992 (3.88 KB)





Model 3 Observations:

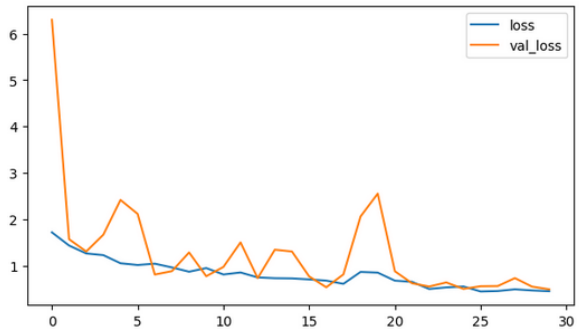
- Max. Training Accuracy 0.7428355813026428
- Max. Validation Accuracy 0.7900000214576721
- Both val and train accuray is not enough

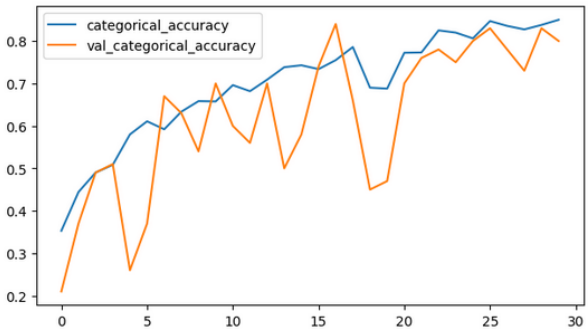
Model 4 : Structure And Params

Lets Adjust a few drop layers and custom filter sizes for each layer

---

frames_to_sample=16
batch_size=5
num_epochs=30
image_height=120
image_width=120
dropout=0.25
Total params: 504709 (1.93 MB)
Trainable params: 503973 (1.92 MB)
Non-trainable params: 736 (2.88 KB)





Model 4 Observations:

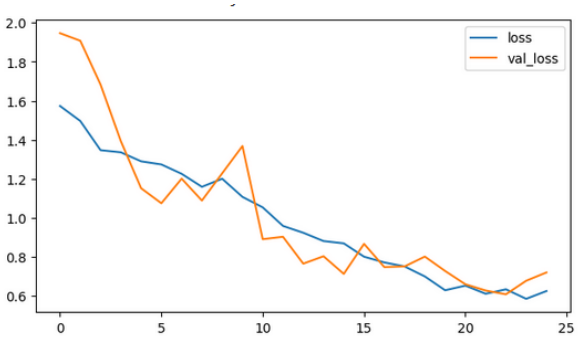
- Max. Training Accuracy 0.8499245643615723
- Max. Validation Accuracy 0.8399999737739563
- Model seems to be good candidate for submission
- The number of params is very less which is great

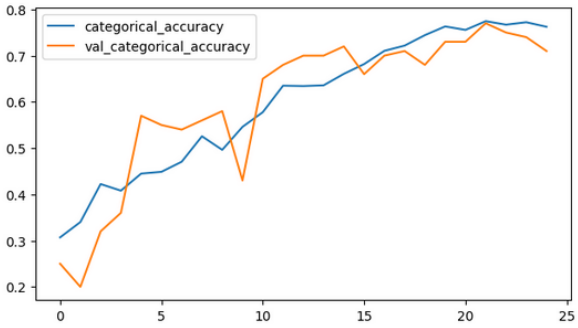
Model 5 : Structure And Params (Cnn Lstm)

Lets build a Cnn Lstm with 50% drop out

---

frames_to_sample=20
batch_size=5
num_epochs=25
image_height=120
image_width=120
dropout=0.50%
Total params: 1657445 (6.32 MB)
Trainable params: 1656453 (6.32 MB)
Non-trainable params: 992 (3.88 KB)





Model 5 Observations:

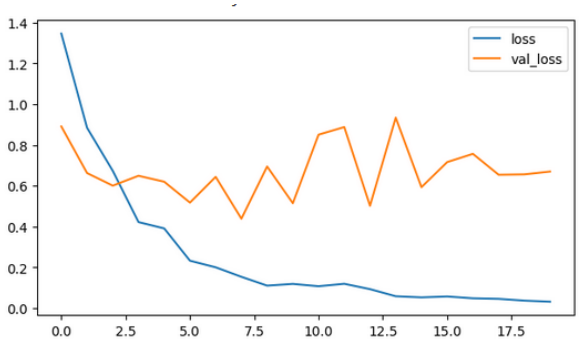
- Max. Training Accuracy 0.7745097875595093
- Max. Validation Accuracy 0.7699999809265137

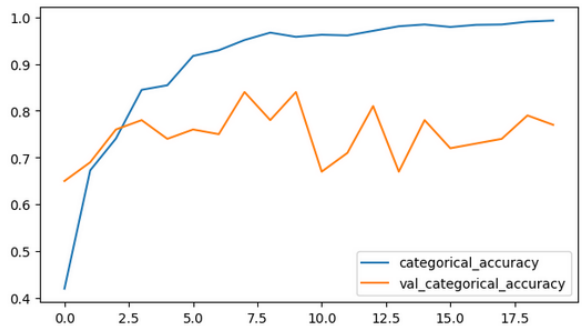
Model 6 : Structure And Params - Transfer Learning

Lets implement transfer learning by also reducing the dropouts and frame samples

---

frames_to_sample=16
batch_size=5
num_epochs=20
image_height=120
image_width=120
dropout=0.25%
Total params: 1762613 (6.72 MB)
Trainable params: 1761109 (6.72 MB)
Non-trainable params: 1504 (5.88 KB)





Model 6 Observations:

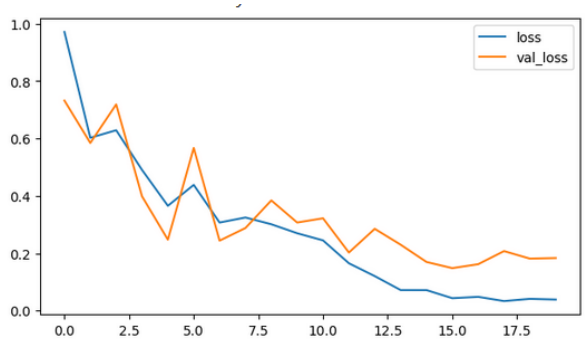
- Max. Training Accuracy 0.9924585223197937
- Max. Validation Accuracy 0.8399999737739563

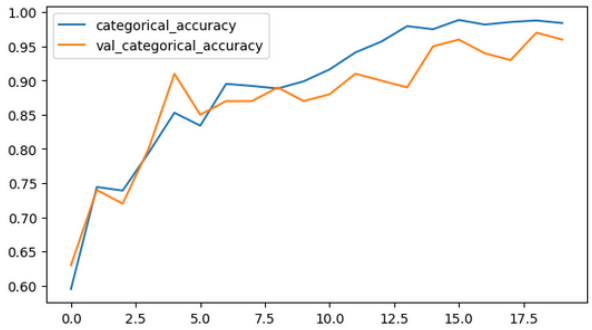
Model 7 : Structure And Params -

Transfer Learning with GRU and training all weights and also increased the image size to 160,160

---

frames_to_sample=16
batch_size=5
num_epochs=20
image_height=160
image_width=160
dropout=0.25%
Total params: 4872901 (18.59 MB)
Trainable params: 4848965 (18.50 MB)
Non-trainable params: 23936 (93.50 KB)





Model 7 Observations:

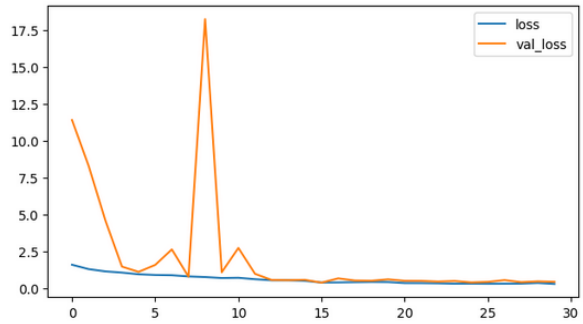
- Max. Training Accuracy 0.9886877536773682
- Max. Validation Accuracy 0.9700000286102295
- Best model performance with regards to accuracy and prediction
- Number of params took to achieve this is high and model took more time to train

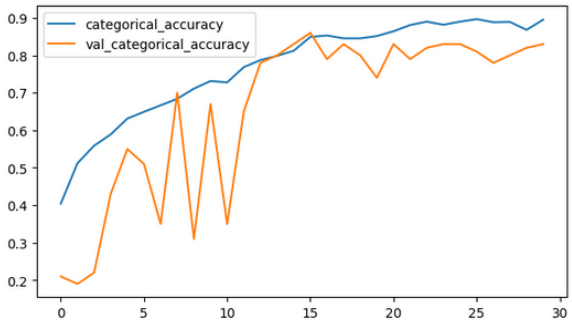
Model 8 : Structure And Params -

Lets try to Optimize Model 2 with less number of dropouts, params , filter size , sample frames and training for longer period with epoch 30

---

frames_to_sample=16
batch_size=5
num_epochs=30
image_height=120
image_width=120
dropout=0.25
Total params: 909637 (3.47 MB)
Trainable params: 908645 (3.47 MB)
Non-trainable params: 992 (3.88 KB)





Model 8 Observations:

- Max. Training Accuracy 0.8966817259788513
- Max. Validation Accuracy 0.8600000143051147
- Model 8 has better performance than model 2 and great accuracy results
- It also has less than 10L parameters
- Because of these performance metrics Iam choosing model 8 as my submission

Further Improvements:

- Higher resolution of images to be used but more powerful computer is needed
- The Training period can also be increased with better computer
- Transfer learning Gru with an all waits model is great and combined with the above two suggestions will make the model more accurate to the point of being perfect.
- More data with more clear parameters can be used to better the model prediction accuracy
- Adagrad and Adadelta could also be used to increase accuracy of the model further.

Model Chosen For Submission: Model 8

- model-00016-0.40387-0.84917-0.39701-0.86000.h5