Gesture Recognition Assignment using neural networks:

Group Members:

- Rohan Bhale
- Rahul Agarwal

Problem statement:

Given a sequence of images representing a video classify the gesture being used to control a smart tv functionality to 5 given class:

1. Thumbs up: Increase the volume
2. Thumbs down: Decrease the volume
3. Left swipe: Jump backwards 10 seconds
4. Right swipe: Jump forwards 10 seconds
5. Stop: Pause the movie

Image preprocessing:

- Image size: 360*360:
    - Scaled down to 100*100
- Image size: 120*160
    - Horizontal crop: 20 from left and right each
    - Scaled down to 100 * 100

Conv3D base model:

- Start Batch size: 100, Time steps used: 15(img_idx generated using range(0,30,2))
- Architecture:
    - Input: (15, 100,100,3)
    - Conv3d: 32 kernels of size 2, padding same, activation relu
    - MaxPool3D
    - Conv3d: 16 kernels of size 2, padding same, activation relu
    - MaxPool3D
    - Conv3d: 8 kernels of size 2, padding same, activation relu
    - MaxPool3D
    - Flatten
    - Dense: 512, activation relu
    - Dense: 256, activation relu
    - Dense: 5, activation: softmax ~~~~Output
- Starting learn rate: 0.001, Optimizer used: Adam

Experiment Number	Model	Result	Decision + Explanation
1	Conv3D Tried overfitting on 50% data. Chose batch size of 100	Throws Resource Exhausted error	Reduce batch size to 50
2	Conv3D Tried overfitting on 50% data. Chose batch size of 50	Throws Resource Exhausted error	Reduce batch size to 25
3	Conv3D Tried overfitting on 50% data. Chose batch size of 50	Worked. Model overfitted. Train-Val accuracy: 96-70	Increase the amount of trainable data
4	Conv3D Tried overfitting on 100% data.	Model overfitted. Convergence was quicker with 100% data for train Train-Val accuracy: 96-70	Lets try reduce the params. For this lets replace the Dense layer of 256 with 64. We choose this because it reduces the param count moderately. We want to slowly reduce the params if needed
5	Conv3D	Model still overfitted. Train-Val accuracy: 96-70	As accuracies remain almost the same, lets try adding dropouts of 0.2 for the Dense layer inputs.
6	Conv3D	Reduced the Overfitting. Train-Val accuracy: 96-80 Val accuracy plateaued at 80	Increase the dropouts to 0.25
7	Conv3D	Reduced the Overfitting. Train-Val accuracy: 96-85 Val accuracy plateaued around 85	Add Batch Normalization for the Dense layer

8	**Conv3D**	**The convergence was quicker reducing the epochs required to reach train-val accuracy of 92-86 to 13 epochs**	**The overfitting doesn't seem to reduce from now onwards. So lets pick the best model and fine tune with a low learning rate. We tried different lr and 0.00001 turned out to be good.**
10	**Conv3D**	**After fine tuning with lr of 0.00001 for 11 more epochs we got the best train-val accuracy of 96-96**	**We now stopped the experiments here for Conv3D as we got a good model. The observed size of the final model saved in h5 file was 7.7 Mb**

<u>Conv2D + GRU base model:</u>

- Start Batch size: 25, Time steps used: 15(img_idx generated using range(0,30,2))
- Architecture:
    - Input: (15, 100,100,3)
    - TimeDistributed
        - Input: (100, 100, 3)
        - Conv2d: 64 kernels of size 2, padding same, activation relu
        - MaxPool2D
        - Conv2d: 32 kernels of size 2, padding same, activation relu
        - MaxPool2D
        - Conv2d: 16 kernels of size 2, padding same, activation relu
        - MaxPool2D
        - Flatten
    - GRU: 128 cells
    - GRU: 64 cells
    - GRU: 32 cells
    - Dense: 512, activation relu
    - Dense: 256, activation relu
    - Dense: 5, activation softmax ~ ~ Ouput
- Starting learn rate: 0.001, Optimizer used: Adam

Experiment Number	**Model**	**Result**	**Decision + Explanation**
1	**Conv2D + GRU**	**Overfitted on the data**	**Now try overfitting on the full train set.**

	Tried overfitting on 50% data.		
2	**Conv2D + GRU**	**Model overfitted. Convergence was quicker with 100% data for train** **Train-Val accuracy: 98-70**	**As accuracies remain almost the same, lets try adding dropouts of 0.2 for the GRUs with 128 and 64 cells.**
3	**Conv2D + GRU**	**Model still overfitted.** **Train-Val accuracy: 98-70**	**As accuracies remain almost the same, lets try adding dropouts of 0.2 for the Dense layer inputs.**
6	**Conv2D + GRU**	**Reduced the Overfitting.** **Train-Val accuracy: 98-85 in 22 epochs** **Val accuracy plateaued at 85**	**Lets reduce params by removing the Dense layer with 256 neurons. We selected this layer as this moderately reduces the params.**
7	**Conv2D + GRU**	**Reduced the Overfitting.** **Train-Val accuracy: 96-92 in 24 epochs.**	**We now stopped the experiments here for Conv2D + GRU as we got a good model. The observed size of the final model saved in h5 file was 12.2 Mb**

Comparisons:

Sr. No	**Model**	**Accuracy(Train_Val)**	**H5 File size**
1	Conv3D	96-96	7.7 MB
2	Conv2D + GRU	97-92	12.2 MB

Summary:

- Conv3D is lighter and provides a descent level of accuracy. The Conv2d + GRU also provides high accuracy but the size is 12.2 MB. For our case as the model will be loaded on the smart tv with memory constraints, its better to go with the **Conv3D model.**
- So our model of choice is **Conv3D**.