# **GESTURE RECOGNITION:**

# Develop a 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.

* 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

1. **Time Distributed Conv 2D vs Conv2D:** Time Distributed Conv2D allows us to work on sequence of frames(images) that make up the video. It gives an extra dimension to the tensor and makes the learning images as sequence in a video for gesture recognition possible.
2. **Conv 3D:** Conv3D extends the concept of 2D convolutions to three dimensions, allowing it to process spatial and temporal information simultaneously. [This makes it particularly effective for tasks like video analysis, where both spatial and temporal features are important](https://openaccess.thecvf.com/content/ACCV2020/papers/Manttari_Interpreting_Video_Features_A_Comparison_of_3D_Convolutional_Networks_and_ACCV_2020_paper.pdf). Also Conv3D requires less parameters than GRU and LSTM.
3. **GRU/LSTM:** 
   1. LSTMs use three gates (input, forget, and output) to control the flow of information. GRUs use two gates (reset and update) and merge the cell state and hidden state into a single state.
   2. LSTM has cell state that captures long term information but requires more memory. GRU are simpler and faster to train.
   3. GRUs are more memory-efficient, which can be beneficial for tasks with limited computational resources. We will go with GRU.
4. **Cropping Logic**
   1. We will try two logic. Crop 10 pixel from all sides
   2. Crop based on aspect ratio. Get the current and target aspect ratio. If current aspect ratio is higher than target aspect ratio we adjust the width otherwise we adjust height. If current and target aspect ratio are same we do not crop.
   3. Crop from top as most of the images do not have any info in the top. But bottom, left and right have info so we won’t crop much there.
5. **Generator Logic:** 
   1. We will be running an infinite loop that runs until all the batches are exhausted.
   2. We will be cropping, resizing and normalizing all the images.
   3. We will normalize each channel of the image separately instead of the entire image. The goal is to ensure that the pixel values within each channel have similar scales. By doing this, we make sure that the network doesn’t get confused by vastly different ranges of pixel values across channels.

|  |  |  |  |
| --- | --- | --- | --- |
| **Experiment Number** | **Model** | **Result** | **Decision + Explanation** |
| **1** | 1. Conv2D(Time Distribued) + Batch\_size=100 2. Num\_epochs = 3 3. Train on alternate frames, 15 frames total 4. Cropping logic: 10 pixel from corner 5. Model Details: 1 Conv2D(Time Distributed, 16 filters, 3,3(kernel)), Flatten, 1 GRU(16 filter), Padding = Same, Optimizer = adam   Total params: 7,682,517  Trainable params: 7,682,517  Non-trainable params: 0 | **Training accuracy: 32.7%**  **Validation Accuracy: 30%**  **Train Loss:1.49**  **Val Loss: 1.59** | We see that we were able to achieve val categorical accuracy of 30% on validation data and val\_loss of 1.49 in the ablation experiment. The model did learn, we are in the right direction. see if we can improve on our previous outcome.  **Decision**: Add one more Conv2D layer and num\_epochs = 5 |
| **2** | 1. batch\_size = 32 2. num\_epochs = 5 3. img\_width, img\_height = 100, 100 4. Resize, Crop logic - Crop on corners - 10 pixels 5. Train on 15 frames, alternate frames chosen 6. Model Details: 1 Conv2D(Time Distributed, 16 filters, 3,3(kernel)), 1 Conv2D(Time Distributed, 16 filters, 3,3(kernel)), Flatten, 1 GRU(16 filter), Padding = Same, Optimizer = adam, categorical\_accuracy evaluation   Total params: 30,730,757  Trainable params: 30,730,757  Non-trainable params: 0 | **Training accuracy: 23.9%**  **Validation Accuracy: 24%**  **Train Loss:1.58**  **Val Loss: 1.63** | Let's try changing out crop logic. As we can see after observing random images, top of the image does not contain any info.  **Decision**: We will crop 40 pixels from top, 10 pixels from bottom. We will also change the cropping logic to not maintain aspect ratio when cropping. |
| **3** | 1. batch\_size = 32 2. num\_epochs = 5 3. img\_width, img\_height = 100, 100 4. Resize, Crop logic - Crop on top - 40 pixels, bottom – 10 pixels. No taking in account aspect ratio. 5. Train on 15 frames, alternate frames chosen | **Training accuracy: 22.7%**  **Validation Accuracy: 26%**  **Train Loss:1.6**  **Val Loss: 1.59** | Looks like our resizing did not affect the model positively. We will go back to previous resizing logic but  **Decision**: keep cropping from top only. Also, we will change the frames we will train on. We will get rid of beginning and end 4 frames. |
| **4** | 1. batch\_size = 32 2. num\_epochs = 5 3. img\_width = 100 4. img\_height = 100 5. crop logic = top 40 pixels, crop on basis of aspect ratio 6. img\_idx = drop first 4 and last 4 frames, total 22 frames | **Training accuracy: 15.6%**  **Validation Accuracy: 17%**  **Train Loss:1.6**  **Val Loss: 1.56** | There was no improvement in the model.  **Decision**: We will go back to using alternate frames in img\_idx, keep the batch\_size at 32. We will also add batch normalization for better learning. |
|  | 1. batch\_size = 32 2. img\_idx = alternate frames 3. img\_width, img\_height = 100, 100 4. num\_epochs = 5 5. Add batch\_normalization   Total params: 61,462,757  Trainable params: 61,462,661  Non-trainable params: 96 | **Training accuracy: 47.36%**  **Validation Accuracy: 33%**  **Train Loss:1.31**  **Val Loss: 1.67** | We can see that there in definite improvement in the categorical accuracy in just 5 epochs. However val\_categorical\_accuracy is lower.  **Decision:** We will now try Conv3D layer instead of Conv2D |
| **5** | 1. batch\_size = 32 2. img\_idx = alternate frames 3. img\_width, img\_height = 100, 100 4. num\_epochs = 5 5. Add batch\_normalization 6. Conv 3D layer | **Training accuracy: 32.5%**  **Validation Accuracy: 33%**  **Train Loss:1.49**  **Val Loss: 1.73** | We can see that the gap between the train and val accuracy has reduced.  **Decision:** Add one more GRU layer to improve accuracy numbers |
| **6** | 1. batch\_size = 32 2. img\_idx = alternate frames 3. img\_width, img\_height = 100, 100 4. num\_epochs = 5 5. Add batch\_normalization 6. Conv 3D layer 7. 2 GRU layer instead of 1 | **Training accuracy: 100%**  **Validation Accuracy: 52%**  **Train Loss:1.00**  **Val Loss: 1.09** | Our categorical accuracy took a huge jump by adding a GRU unit but there is a huge gap between val accuracy and train accuracy.  **Decision:** Let's introduce maxpooling and dropout for better generalization and overfitting prevention |
| **7** | 1. batch\_size = 32 2. num\_epochs = 5 3. img\_width, img\_height = 100, 100 4. Add batch\_normalization 5. Conv3d, Conv3d + Maxpooling + Dropout(0.10), TimeDistributed Flatten + GRU + GRU + Dropout(0.10) + Flatten + Dense | **Training accuracy: 98.9%**  **Validation Accuracy: 36%**  **Train Loss:0.17**  **Val Loss: 1.59** | There is still huge gap between training and validation accuracy.  **Decision:** We will add more drop and Maxpooling layer. |
| **8** | 1. batch\_size = 32 2. num\_epochs = 5 3. img\_width, img\_height = 100, 100 4. Add batch\_normalization 5. Conv3d + Maxpooling + Dropout(0.25), Conv3d + Maxpooling + Dropout(0.25), TimeDistributed Flatten + GRU + Dropout(0.25) + GRU + Dropout(0.25) + Flatten + Dense | **Training accuracy: 60%**  **Validation Accuracy: 55%**  **Train Loss:1.09**  **Val Loss: 1.22** | The model is much more balanced now. Training loss and Validation loss are very close to each other. categorical accuracy has dropped but that is to be expected after 5 epochs. Best thing is that validation accuracy and train accuracy are close to each other.  **Decision:** Let's reduce the dropout rate to 10% in Conv3D and keep it at 25% in GRU and see if it helps improve the value of accuracy overall while keep at two accuracies close to each other. |
| **9** | 1. batch\_size = 32 2. num\_epochs = 5 3. img\_width, img\_height = 100, 100 4. Add batch\_normalization 5. Conv3d + Maxpooling + Dropout(0.10), Conv3d + Maxpooling + Dropout(0.10), TimeDistributed Flatten + GRU + Dropout(0.25) + GRU + Dropout(0.25) + Flatten + Dense | **Training accuracy: 75%**  **Validation Accuracy: 51%**  **Train Loss:0.83**  **Val Loss: 1.22** | There is a definite improvement in train accuracy by reducing dropout in the initial layer but validation accuracy is lower than train accuracy by a lot.  **Decision:** Let's introduce regularization and see if it helps close the gap between the training and validation accuracy. We will also increase epochs to 15 to get a real feel of accuracy |
| **10.** | 1. batch\_size = 32 2. num\_epochs = 15 3. img\_width, img\_height = 100, 100 4. Add batch\_normalization 5. Conv3d + Maxpooling + Dropout(0.10) + L2(0.01), Conv3d + Maxpooling + Dropout(0.10) + L2(0.01), TimeDistributed Flatten + GRU + Dropout(0.25)+ L2(0.01) + GRU + Dropout(0.25) + L2(0.01)+ Flatten + Dense + L2(0.01) | **Training accuracy: 99.4%**  **Validation Accuracy: 70%**  **Train Loss:5.56**  **Val Loss: 5.54** | Training Accuracy is at a good point and gap between training and validation accuracy has reduced. Train and Validation loss are also very close. We will stop here. |
| **Final Model** | **We will keep the above model as our final model. There is possibility of further reducing the gap between training and validation by increasing dropouts, epochs, augmentation. But we will end here.** | **………….** | **…………………** |