# Case Study: Gesture Recognition for Smart TV Control using Conv3D and RNN

**Prem Kumar Subudhi**

Gesture recognition feature in smart TV leverage the advanced technology that interacts with human gestures and interprets using a Model with CNN and RNN design architecture. In this case study, we explore the application of gesture recognition. The goal is to develop a system that allows user to control their smart TV using hand gestures captured by a webcam. Specifically, gestures such as thumbs up, thumbs down, and swiping left or right are recognized to perform actions such as stop, seek forward/backward, volume up and down in a video playback.

The dataset used for this study consists of images captured from recorded videos, comprising instances of each gesture (thumbs up, thumbs down, left swipe, right swipe). In total, the dataset contains 663 images. Each image is labelled according to the corresponding gesture.

Methodology:

The approaches followed are mentioned below.

1. The experiment at Jarvis lab found best w.r.t speed of execution
2. Small data to check if the model over fits or not.
3. Increased the images from 5, 10 and 15 using random sampling from each image folders.

# Data-Pre-processing:

The images resized.

Image normalized.

Data augmentation

Selected 16 images out of 30 for the training for each folders.

# Issues encountered:

Issues encountered running Google Colab / Local.

Executions were too slow.

Jarvis came into rescue:

The models run fine with few modification w.r.t to the environment.

Some of issues encountered are:

OOM Error, No Space left issues, other syntax issues.

Callback\_list used are ModelCheckPoint and ReduceLROnPlateau with save\_best\_only=True

# Model Development:

Initiated with basic Convolution 3D models

Convolution 3D models with different sizes (64,128,256…)

Maxpool3D( stride 2,2,2)

Activation: Relu/softmax to classify the image.

Run with incremental layers.

Run with incremental layers with dropouts.

With different batch\_sizes and epoch numbers.

With CNN and LSTM

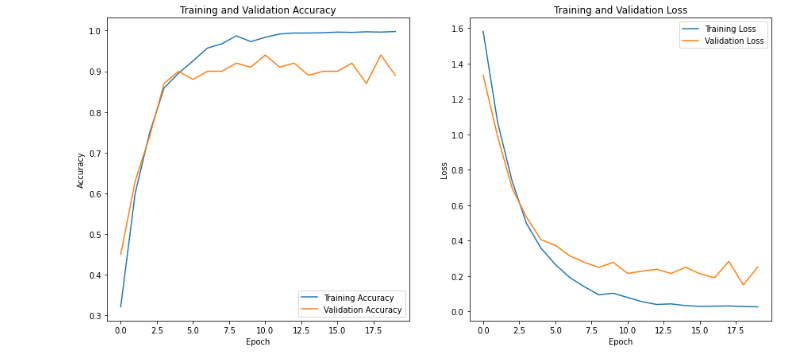
With CNN and GRU

Transfer learning with Mobilenet

# Experiments:

|  |  |  |  |
| --- | --- | --- | --- |
| **Exp No** | **Model** | **Result** | **Decision + Explanation** |
| **1** | Conv3dArchitecture\_  1 Conv3d\_  1 Flatten  2 Dropout(.5)  2 Dense ( 512,5, softmax)  img\_idx = [0,1,2,4,6,8,10,12,14,16,18,20,22,24,26,27,28,29] Total params: 163,848,581  Trainable params: 163,848,453  Non-trainable params: 128 | categorical\_accuracy: 0.6042  val\_categorical\_accuracy: 0.2625 | The result seen is over fitting.  Started with min layers, but with all features like data generators and augmentation implemented. |
| **2** | Conv3dArchitecture\_  2 Conv3d\_  1 Flatten  2 Dropout(.5)  2 Dense ( 512,5, softmax)  img\_idx = [0,1,2,4,6,8,10,12,14,16,18,20,22,24,26,27,28,29]  Total params: 41,190,405  Trainable params: 41,190,021  Non-trainable params: 384 | categorical\_accuracy: 0.5253  val\_categorical\_accuracy: 0.4500 | New set of Conv3D(stride 1,1,1), Batch Normalization, Activation: Relu, Maxpool3D( stride 2,2,2) |
| **3** | Conv3dArchitecture\_  3 Conv3d\_  1 Flatten  2 Dropout(.5)  2 Dense ( 512,5, softmax)  img\_idx = [0,1,2,4,6,8,10,12,14,16,18,20,22,24,26,27,28,29]  Total params: 23,267,589  Trainable params: 23,266,693  Non-trainable params: 896 | categorical\_accuracy: 0.7768  val\_categorical\_accuracy: 0.0750 | Two set of additional Conv3D(stride 1,1,1), Batch Normalization, Activation: Relu, Maxpool3D( stride 2,2,2) Based on the provided training and validation categorical accuracies, it seems that the model is overfitting. |
|  |  |  | Issue of Memory  OOM  h5 could not be saved Cleaned the temporary files ! rm -rf .local/share/Trash/files/\*  When dimension taken 120/120 it failed to run  Later when the image size set to 100/100 , it continued execution. |
| **4** | 4 Sets of Conv 3D(64,128,256,256) (stride 1,1,1), Batch Normalization, Activation: Relu, Maxpool3D( stride 2,2,2) Total params: 7,605,765  Trainable params: 7,604,357  Non-trainable params: 1,408 | categorical\_accuracy: 0.**4915** - val\_categorical\_accuracy: 0.**0500**  Based on the provided training and validation categorical accuracies, it seems that the model is overfitting. | The number of layers have increased , but the result has not improved. |
| **5** | Model: "sequential\_2"  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Layer (type) Output Shape Param #  =================================================================  conv3d\_8 (Conv3D) (None, 18, 100, 100, 64) 5248    batch\_normalization\_8 (Batc (None, 18, 100, 100, 64) 256  hNormalization)    activation\_8 (Activation) (None, 18, 100, 100, 64) 0    max\_pooling3d\_8 (MaxPooling (None, 9, 50, 50, 64) 0  3D)    conv3d\_9 (Conv3D) (None, 9, 50, 50, 128) 221312    batch\_normalization\_9 (Batc (None, 9, 50, 50, 128) 512  hNormalization)    activation\_9 (Activation) (None, 9, 50, 50, 128) 0    max\_pooling3d\_9 (MaxPooling (None, 4, 25, 25, 128) 0  3D)    conv3d\_10 (Conv3D) (None, 4, 25, 25, 256) 884992    batch\_normalization\_10 (Bat (None, 4, 25, 25, 256) 1024  chNormalization)    activation\_10 (Activation) (None, 4, 25, 25, 256) 0    max\_pooling3d\_10 (MaxPoolin (None, 2, 12, 12, 256) 0  g3D)    conv3d\_11 (Conv3D) (None, 2, 12, 12, 256) 1769728    batch\_normalization\_11 (Bat (None, 2, 12, 12, 256) 1024  chNormalization)    activation\_11 (Activation) (None, 2, 12, 12, 256) 0    max\_pooling3d\_11 (MaxPoolin (None, 1, 6, 6, 256) 0  g3D)    flatten\_2 (Flatten) (None, 9216) 0    dropout\_4 (Dropout) (None, 9216) 0    dense\_4 (Dense) (None, 512) 4719104    dropout\_5 (Dropout) (None, 512) 0    dense\_5 (Dense) (None, 5) 2565    =================================================================  Total params: 7,605,765  Trainable params: 7,604,357  Non-trainable params: 1,408  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  None | categorical\_accuracy: 0.4915 - val\_loss: 1.6540 - val\_categorical\_accuracy: 0.1000  Epoch 2/2  11/11 | There is an improvement but not much improvement seen, the model still in highly overfit.  But the number of |
| **6** | Model: "sequential\_3"  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Layer (type) Output Shape Param #  =================================================================  conv3d\_12 (Conv3D) (None, 18, 100, 100, 64) 5248    batch\_normalization\_12 (Bat (None, 18, 100, 100, 64) 256  chNormalization)    activation\_12 (Activation) (None, 18, 100, 100, 64) 0    max\_pooling3d\_12 (MaxPoolin (None, 9, 50, 50, 64) 0  g3D)    conv3d\_13 (Conv3D) (None, 9, 50, 50, 128) 221312    batch\_normalization\_13 (Bat (None, 9, 50, 50, 128) 512  chNormalization)    activation\_13 (Activation) (None, 9, 50, 50, 128) 0    max\_pooling3d\_13 (MaxPoolin (None, 4, 25, 25, 128) 0  g3D)    conv3d\_14 (Conv3D) (None, 4, 25, 25, 256) 884992    batch\_normalization\_14 (Bat (None, 4, 25, 25, 256) 1024  chNormalization)    activation\_14 (Activation) (None, 4, 25, 25, 256) 0    max\_pooling3d\_14 (MaxPoolin (None, 2, 12, 12, 256) 0  g3D)    conv3d\_15 (Conv3D) (None, 2, 12, 12, 256) 1769728    batch\_normalization\_15 (Bat (None, 2, 12, 12, 256) 1024  chNormalization)    activation\_15 (Activation) (None, 2, 12, 12, 256) 0    max\_pooling3d\_15 (MaxPoolin (None, 1, 6, 6, 256) 0  g3D)    flatten\_3 (Flatten) (None, 9216) 0    dropout\_6 (Dropout) (None, 9216) 0    dense\_6 (Dense) (None, 512) 4719104    dropout\_7 (Dropout) (None, 512) 0    dense\_7 (Dense) (None, 5) 2565    =================================================================  Total params: 7,605,765  Trainable params: 7,604,357  Non-trainable params: 1,408  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  None | categorical\_accuracy: **0.6184 -** val\_loss: 3.1264 - val\_categorical\_accuracy: **0.3300** - lr: 1.0000e-05  Epoch 1/20  21/21 | 20 epoch improved the val accuracy to .33 from .1, but the cat accuracy is improved to .61 from .49  Earlier one was:  categorical\_accuracy: 0.4915 - val\_loss: 1.6540 - val\_categorical\_accuracy: **0.1000**  It still needs lots of improvement. |
| **7** | **Same above Model with**   1. Image resolution =(80 , 80)   Total params: 6,163,973  Trainable params: 6,162,565  Non-trainable params: 1,408 | categorical\_accuracy: 0.5958  val\_categorical\_accuracy: 0.3000 | No improvement seen compared to last run.  **Decision:** Set the image resolution (120,120), will reduce the batch size as well. |
| **8** | batch\_size = 16  num\_epochs = 20  Image resolution =(80,80)  Normalization method /255.  Total params: 6,163,973  Trainable params: 6,162,565  Non-trainable params: 1,408 | categorical\_accuracy: 0.6094  val\_categorical\_accuracy: 0.7300 | **Could see good improvement in accuracy.**  **Next will try with**  **num\_epochs = 30**  **batch\_size = 32**  **y,z= 100,100** |
| **9** | **num\_epochs = 30**  **batch\_size = 32**  **y,z= 100,100** | categorical\_accuracy: 0.6184 - val\_loss: 1.1684 - val\_categorical\_accuracy: 0.5600 | **The validation has come down though there is marginal increase in categorical accuracy.** |
| **10** | **Conv 3D:, S-1,32,64,128,128, F, DRP,Dense 256, Drp .5, Dense-5**  **Model( VGG16: Dense-64 with RELU): TimeDistributed,**  **freeze layers, LSTM (64,32) Dense 8: relu, Dense 5: softmax.**  **num\_epochs = 30 # choose the number of epochs**  batch\_size = 32  img\_idx = [0,1,2,4,6,8,10,12,14,16,18,20,22,24,26,27,28,29]  x=len(img\_idx) # Number of images in sequence  y = 100  z = 100 | - categorical\_accuracy: 0.5385  - val\_categorical\_accuracy: 0.5100 | **The accuracy of cat and val quite close.**  **But it requires additional improvement in accuracy. GRU** |
| **11** | **Conv 3D:, S-1,32,64,128,128, F, DRP,Dense 256, Drp .25, Dense-5**  **Model( VGG16: Dense-64 with RELU): TimeDistributed,**  **freeze layers,**  **GRU (128,64)**  **Dense 64: relu, Dense 5: softmax.**  **num\_epochs = 30 # choose the number of epochs**  batch\_size = 32  img\_idx = [0,1,2,4,6,8,10,12,14,16,18,20,22,24,26,27,28,29]  x=len(img\_idx) # Number of images in sequence  y = 100  z = 100  **adam = Adam(learning\_rate=0.0001)** | categorical\_accuracy: 0.9894 –  - val\_categorical\_accuracy: 0.8300 | **The val lass has decreased tremendously and the scores of both have improved quite well and impressive.** |
| **12** | **cnn\_gru(x,y,z, 0.5)**  **Dropout set to .5 of above model**  Total params: 15,125,893  Trainable params: 411,205  Non-trainable params: 14,714,688 | categorical\_accuracy: 0.9563  - val\_categorical\_accuracy: 0.8300 | Almost same in accuracy but cat accuracy is decreased now.  **Decision reduce the epoch and increase the batch\_size**  **num\_epochs = 15 # choose the number of epochs**  **batch\_size = 32** |
| **13** | Conv 3D:, S-1,32,64,128,128, F, DRP,Dense 256,  Drp .25, Dense-5  Model( VGG16: Dense-64 with RELU): TimeDistributed,  freeze layers,  GRU (128,64)  Dense 64: relu, Dense 5: softmax.  num\_epochs = 30 # choose the number of epochs  batch\_size = 32  img\_idx = [0,1,2,4,6,8,10,12,14,16,18,20,22,24,26,27,28,29]  x=len(img\_idx) # Number of images in sequence  y = 100  z = 100  adam = Adam(learning\_rate=0.0001)  **num\_epochs = 15 # choose the number of epochs**  **batch\_size = 32** | categorical\_accuracy: 0.8733 - val\_loss: 0.6824 - val\_categorical\_accuracy: 0.7300 | **Not improved much.** |
| **14** | **Same above model with**  **num\_epochs = 20 # choose the number of epochs**  **batch\_size = 32**  **y = 100**  **z = 100** | categorical\_accuracy: 0.9955 - val\_loss: 0.6963 - val\_categorical\_accuracy: 0.7600 | **Impressive** 0.9955 for cat accuracy |
| **15** | **num\_epochs=15** | categorical\_accuracy: 0.9517 - val\_loss: 0.7139 - val\_categorical\_accuracy: 0.7500 | **No improvement.**  **Img\_augment attempted.**  **from tensorflow.keras.optimizers import SGD** |
|  | **image\_augment**  **generator\_with\_aug**  **Conv 3D:, S-1,64,128,256,256**, F, DRP,Dense 256,  Drp .25, Dense-5  Model( VGG16: Dense-64 with RELU): TimeDistributed,  freeze layers,  GRU (128,64)  Dense 64: relu, Dense 5: softmax.  num\_epochs = 30 # choose the number of epochs  batch\_size = 32  img\_idx = [0,1,2,4,6,8,10,12,14,16,18,20,22,24,26,27,28,29]  x=len(img\_idx) # Number of images in sequence  y = 100  z = 100  adam = Adam(learning\_rate=0.0001)  **num\_epochs = 20# choose the number of epochs**  **batch\_size = 20** | categorical\_accuracy: 0.6003 - val\_loss: 0.9865 - val\_categorical\_accuracy: 0.6300 | **Reduced the accuracy ☹** |
| **16** | **Same with batch\_size = 32** | categorical\_accuracy: 0.9819 - val\_loss: 0.6993 - val\_categorical\_accuracy: 0.7600 | **Seen improvement in result.**  **Decision to use MobileNet** |
| **17** | **MobileNet**  **TD**  **MAXPool**  **GRU 128**  **Dense 128**  **Dense 5**  **num\_epochs = 20 # choose the number of epochs**  **batch\_size = 16**  **ht/wdth=100, 100** | categorical\_accuracy: 0.9962 - val\_loss: 0.2364 - val\_categorical\_accuracy: 0.9000 | **Wow an impressive improvement in accuracy**  **Mobilenet seems a good pic for this model.** |
|  |  | 42/42 [==============================] - 68s 2s/step - loss: 0.0358 - categorical\_accuracy: 0.9925 - val\_loss: 0.2719 - val\_categorical\_accuracy: 0.8800 - lr: 1.0000e-05 |  |
|  |  | Epoch 00015: ReduceLROnPlateau reducing learning rate to 2.499999936844688e-05.  42/42 [==============================] - 70s 2s/step - loss: 0.0396 - categorical\_accuracy: 0.9940 - val\_loss: 0.2396 - val\_categorical\_accuracy: 0.9100 - lr: 5.0000e-05 |  |
| **18** | **Above same model with num\_epochs = 5**  **batch\_size = 32** | Epoch 2/5  21/21 [==============================] - ETA: 0s - loss: 0.4016 - categorical\_accuracy: 0.9148  Epoch 00002: val\_loss improved from 0.80686 to 0.40047, saving model to model\_init\_2024-05-0206\_10\_26.550223/model-00002-0.40163-0.91478-0.40047-0.92000.h5  21/21 [==============================] - 69s 3s/step - loss: 0.4016 - categorical\_accuracy: 0.9148 - val\_loss: 0.4005 - val\_categorical\_accuracy: 0.9200 - lr: 1.0000e-04  Best result is  model-00002-0.40163-0.91478-0.40047-0.92000.h5 | **In one of epoch**  Epoch 2/5  **This model found best but the end of epoch**  Epoch 5/5  21/21 [==============================] - ETA: 0s - loss: 0.0785 - categorical\_accuracy: 0.9955  Epoch 00005: val\_loss improved from 0.28068 to 0.25274, saving model to model\_init\_2024-05-0206\_10\_26.550223/model-00005-0.07846-0.99548-0.25274-0.89000.h5  21/21 [==============================] - 70s 3s/step - loss: 0.0785 - categorical\_accuracy: 0.9955 - val\_loss: 0.2527 - val\_categorical\_accuracy: 0.8900 - lr: 1.0000e-04  categorical\_accuracy: 0.9955  val\_categorical\_accuracy: 0.8900 |
| **19** | **Model checkpoint files stored in** | model-00002-0.40163-0.91478-0.40047-0.92000.h5 |  |
|  |  | model-00013-0.04772-0.99170-0.23017-0.90000.h5 |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Plots for the models



# Results:

The trained Conv3D-RNN model achieves promising results in gesture recognition accuracy. Evaluation metrics such as precision, recall, and F1-score demonstrate the effectiveness of the proposed approach in accurately identifying various hand gestures. Real-time testing on webcam input shows robust performance, enabling users to control smart TV functions seamlessly through intuitive hand movements.

The best model found during experiments is:

## With LSTM:

21/21 [==============================] - 41s 2s/step - loss: 0.2601 –

categorical\_accuracy: **0.9713** - val\_loss: 0.6913 –

val\_categorical\_accuracy: **0.7500** - lr: 5.0000e-05

## With GRU:

**model-00020-0.02472-0.99698-0.17416-0.94000.h5**

categorical\_accuracy: **0.99698**

val\_categorical\_accuracy: **0.9400** - lr:

num\_epochs = 20

batch\_size = 32

img\_idx = [0,1,2,4,6,8,10,12,14,16,18,20,22,24,26,27,28,29]

Height: y = 100

Width: z = 100

Model: "sequential"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

time\_distributed (TimeDistr (None, 18, 3, 3, 1024) 3228864

ibuted)

time\_distributed\_1 (TimeDis (None, 18, 3, 3, 1024) 4096

tributed)

time\_distributed\_2 (TimeDis (None, 18, 1, 1, 1024) 0

tributed)

time\_distributed\_3 (TimeDis (None, 18, 1024) 0

tributed)

gru (GRU) (None, 128) 443136

dropout (Dropout) (None, 128) 0

dense (Dense) (None, 128) 16512

dropout\_1 (Dropout) (None, 128) 0

dense\_1 (Dense) (None, 5) 645

=================================================================

Total params: 3,693,253

Trainable params: 3,669,317

Non-trainable params: 23,936

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## H5 Models

Some good results Model file with .H5 saved are :

model-00002-0.40163-0.91478-0.40047-0.92000.h5

model-00020-0.02472-0.99698-0.17416-0.94000.h5

### Highest accuracy found:

**model-00018-0.02757-0.99774-0.17660-0.94000.h5**

# Conclusion:

In conclusion, this case study demonstrates the feasibility and effectiveness of employing Conv3D and RNN models for gesture recognition in the context of smart TV control. By leveraging deep learning techniques, we have developed a system capable of accurately interpreting hand gestures captured by a webcam in real-time, thereby enhancing the user experience of smart TV interaction. Future work may involve further refinement of the model architecture, exploration of additional gestures, and integration with advanced smart TV functionalities.

References: [upGrad | Learning Platform](https://learn.upgrad.com/course/4705/segment/47956/290759/884746/4421564)

User guide : [upGrad+Online+Lab+Guide.pdf](https://cdn.upgrad.com/uploads/production/815eddc1-1819-4bf0-8936-8bbe3c097bbd/upGrad%2BOnline%2BLab%2BGuide.pdf)

<https://jarvislabs.ai/docs/docs/>

Kaggle sites <http://kaggle.com>

GitHub <https://github.com/>

Keywords: Gesture recognition, Smart TV control, Conv3D, Recurrent Neural Network, Deep Learning.