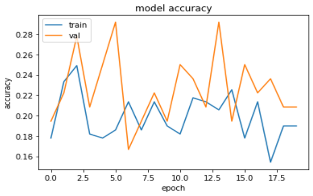
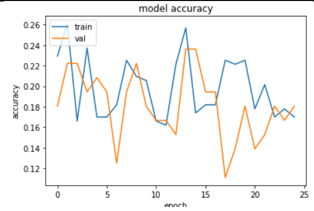
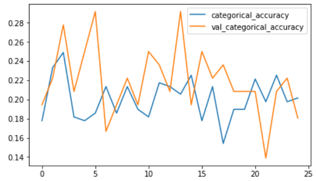
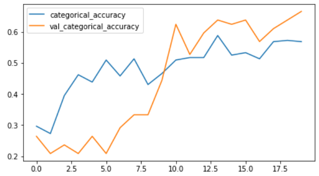
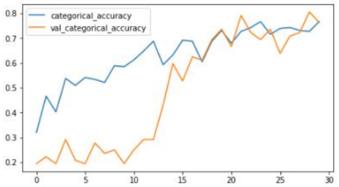
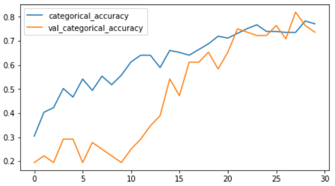
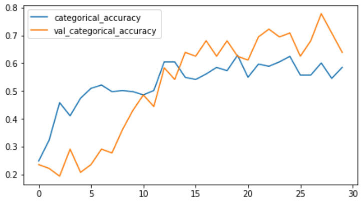
Hand Gesture Recognition With CNN and RNN

H5 file link:

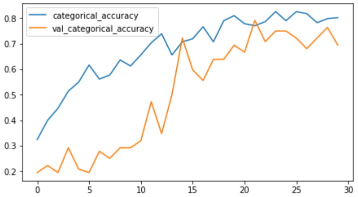
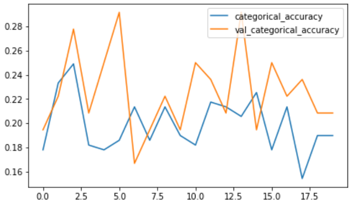
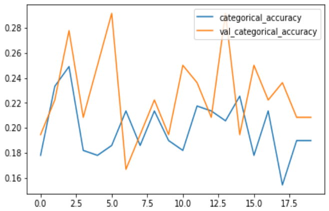
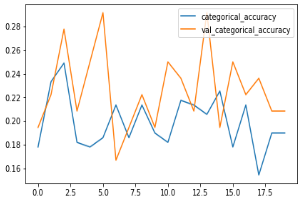
https://drive.google.com/file/d/1Yratzq8Rg7Rra6yR2tld42jd6SeOORS8/view?usp=sharing

Approach Summary:

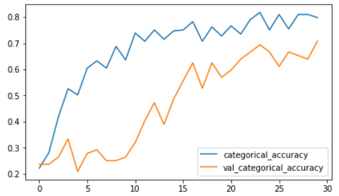
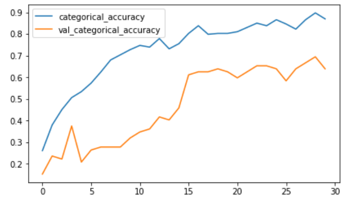
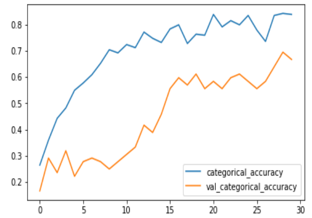
Exp.No.	Model	No. Of Trainable Parameters	Result/Accuracy	Comment
1.	**Conv3D** - **Batch_size=64** - **Activation function = ‘relu’** - **Kernel_size=(3,3,3)** - **Using last 18 image frames**	8,958,629	 categorical_accuracy: 0.1897 val_categorical_accuracy: 0.2083	**Model under-fitting.**
2.	**Conv3D** - **Batch_size=64** - **Activation function = ‘elu’** - **Kernel_size=(3,3,3)** - **Using last 18 image frames** - **Using last 18 image frames**	8,958,629	 categorical_accuracy: 0.1700 val_categorical_accuracy: 0.1806	**Model is under-fitting. Changing the activation function did not improve accuracy,**
3	**Conv3D** - **Batch_size=64** - **Activation function = ‘relu’** - **Kernel_size=(2,2,2)** - **Using last 18 image frames**	9,856,901	 categorical_accuracy: 0.2016 val_categorical_accuracy: 0.1806	**Model Under-fitting. Changing the kernel size did not improve accuracy,**
4.	**Conv3D** - **Batch_size=64** - **Activation function = ‘relu’** - **Kernel_size=(2,2,2)**	9,856,901	**Got ResourceExhaustedError. This means we cannot experiment with batch sizes larger than 64.**	**This means we cannot experiment with batch**

	- Using last 18 image frames			sizes larger than 64.
Exp.No.	**Model**	**No. Of Trainable Parameters**	**Result/Accuracy**	**Comment**
5.	**Conv3D** - **Batch_size=64** - **Activation function = 'elu'** - **Kernel_size=(3,3,3)** - **Using alternate frames**	9,439,365	 categorical_accuracy: 0.5692 val_categorical_accuracy: 0.6667	**Model Over-fitting. Using alternate frames improved model performance.**
6.	**Conv3D** - **Using (84X84) image frames** - **Updated momentum 0.7 to 0.9 in SGD optimizer**	9,439,365	 categorical_accuracy: 0.7668 val_categorical_accuracy: 0.7639	**No over-fitting or under-fitting. Updating momentum reduced the difference between train and validation accuracy.**
7.	**Conv3D** - **Using (100X100) image frames**	12,322,949	 categorical_accuracy: 0.7708 val_categorical_accuracy: 0.7361	**No over-fitting or under-fitting. Changing image size did not improve accuracy.**
8.	**Conv3D** - **Using (100X100) image frames** - **Learning Rate starting from 0.0001**	12,322,949	 categorical_accuracy: 0.5850 val_categorical_accuracy: 0.6389	**Slight over-fitting.**

Hand Gesture Recognition With CNN and RNN

Exp.No.	**Model**	**No. Of Trainable Parameters**	**Result/Accuracy**	**Comment**
9.	**Conv3D** **-Using (84X84) images** **- Using all 30 frames.**	9,439,365	 categorical_accuracy: 0.8024 val_categorical_accuracy: 0.6944	**Slightly over-fitting.**
10.	**Conv2D + GRU** - **Using last 18 (84X84) images per video** - **Using momentum as 0.7 in SGD optimizer**	1,274,245	 categorical_accuracy: 0.1897 val_categorical_accuracy: 0.2083	**Under-fitting.**
11.	**Conv2D + GRU** - **Adding more layers** - **Using last 18 (84X84) images per video** - **Using momentum as 0.7 in SGD optimizer**	733,957	 categorical_accuracy: 0.1897 val_categorical_accuracy: 0.2083	**Under-fitting.**
12.	**Conv2D + GRU** - **Using 18 last (100X100) images** - **Using momentum as 0.7 in SGD optimizer**	1,004,293	 categorical_accuracy: 0.1897 val_categorical_accuracy: 0.2083	**Under-fitting.**

Hand Gesture Recognition With CNN and RNN

13.	**Conv2D + GRU** **- Using alternative 18 (84X84) images** **- Updating momentum as 0.9**	733,957	 categorical_accuracy: 0.7984 val_categorical_accuracy: 0.7083	**Slightly over-fitting. Using alternative frames improved accuracy.**
Exp.No.	**Model**	**No. Of Trainable parameters**	**Result/Accuracy**	**Comment**
14.	**Conv2D + GRU** **- Using 18 alternative (100X100) images** **- Using momentum as 0.9**	1,004,293	 categorical_accuracy: 0.8696 val_categorical_accuracy: 0.6389	**Over-fitting increased after increasing frame size.**
15.	**Conv2D + GRU** **- Using 30 (100X100) images** **- Using momentum as 0.9**	1,004,293	 categorical_accuracy: 0.8379 val_categorical_accuracy: 0.6667	**Over-fitting. Accuracy did not improve after using all frames.**

Best Model:

We have selected model from experiment 6 as our final model for the following reasons.

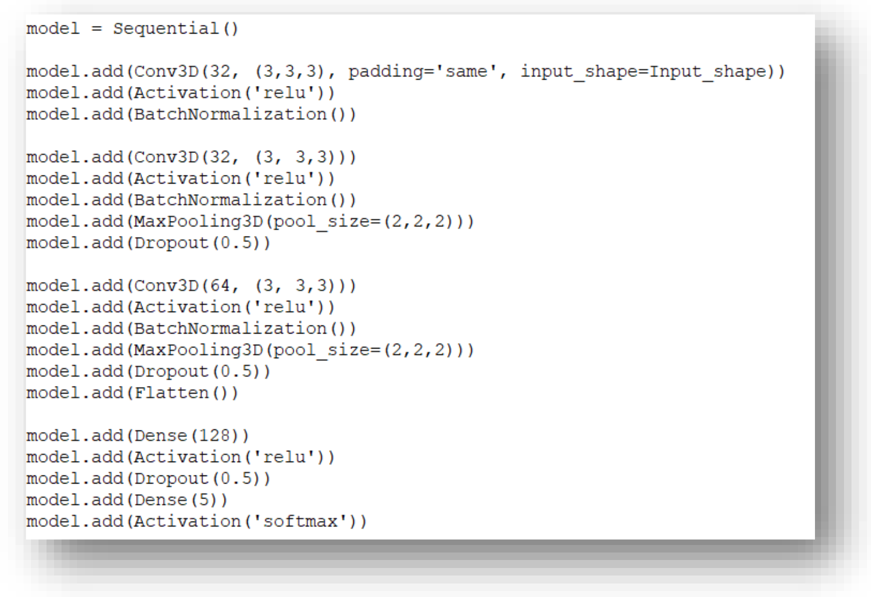
- Using mean subtraction as normalizing technique for the batch gave substantially better performance than dividing pixels by 255. (similar to VGG_ILSVRC_16_layers architecture, where BGR values are subtracted with [103.939, 116.779, 123.68])
- Model is able to capture the gesture from the alternative frames than last 18 consecutive frames.
- We used epoch=30 and batch size=64 for limitation of computational resources.
- Among different values, initial learning rate of 0.001 and momentum = 0.9 gave best accuracy.
- Adding further dropouts is not improving performance.
- Most importantly both training and validation accuracy > 0.75 and very low difference between the 2, signifying that there is no under-fitting or over-fitting.

Hand Gesture Recognition With CNN and RNN

Detailed Approach:

Exp. 1 : Base Conv3D model We started with the base model as below:

- For each video frames from 11th to 29th index are fed to the network.
- Normalization done of by diving every pixel by 255.



```
model = Sequential()

model.add(Conv3D(32, (3,3,3), padding='same', input_shape=Input_shape))
model.add(Activation('relu'))
model.add(BatchNormalization())

model.add(Conv3D(32, (3, 3,3)))
model.add(Activation('relu'))
model.add(BatchNormalization())
model.add(MaxPooling3D(pool_size=(2,2,2)))
model.add(Dropout(0.5))

model.add(Conv3D(64, (3, 3,3)))
model.add(Activation('relu'))
model.add(BatchNormalization())
model.add(MaxPooling3D(pool_size=(2,2,2)))
model.add(Dropout(0.5))
model.add(Flatten())

model.add(Dense(128))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(5))
model.add(Activation('softmax'))
```

Result:

- Model is under-fitting as both training and validation accuracy are poor.
- There are abrupt spikes in the accuracy, indicating unstable nature of the model.

Exp. 2 : Changing Activation Function

Next, we updated activation function ‘elu’ keeping rest same.

Result:

- Model performance is almost similar after changing the activation function.
- Model is under-fitting as both training and validation accuracy are poor.

Exp. 3 : Changing Kernel Size

Next, changing the kernel size from (3,3,3) to (2,2,2)

Result:

- No improvement in model performance.
- Model is under-fitting as both training and validation accuracy are poor.

Exp. 4 : Changing Batch Size

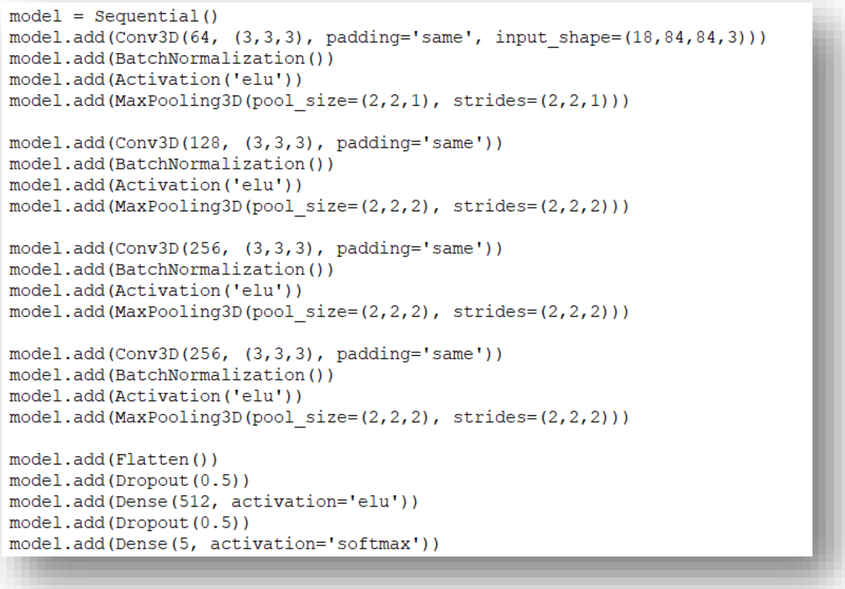
Next, changing the batch size from 64 to 70

Result: Got ResourceExhaustedError. This means we cannot experiment with batch sizes larger than 64.

Exp. 5 : Changing Input Frame and Normalization

- Instead of taking all consecutive frames from last half of video, taking alternative frames throughout the video as input to the network.
- Used mean subtraction as normalizing technique for the batch. (similar to VGG_ILSVRC_16_layers

architecture, where BGR values are subtracted with [103.939, 116.779, 123.68])
- Added extra layers to the network.
- Trained for 20 epochs.



```
model = Sequential()
model.add(Conv3D(64, (3,3,3), padding='same', input_shape=(18,84,84,3)))
model.add(BatchNormalization())
model.add(Activation('elu'))
model.add(MaxPooling3D(pool_size=(2,2,1), strides=(2,2,1)))

model.add(Conv3D(128, (3,3,3), padding='same'))
model.add(BatchNormalization())
model.add(Activation('elu'))
model.add(MaxPooling3D(pool_size=(2,2,2), strides=(2,2,2)))

model.add(Conv3D(256, (3,3,3), padding='same'))
model.add(BatchNormalization())
model.add(Activation('elu'))
model.add(MaxPooling3D(pool_size=(2,2,2), strides=(2,2,2)))

model.add(Conv3D(256, (3,3,3), padding='same'))
model.add(BatchNormalization())
model.add(Activation('elu'))
model.add(MaxPooling3D(pool_size=(2,2,2), strides=(2,2,2)))

model.add(Flatten())
model.add(Dropout(0.5))
model.add(Dense(512, activation='elu'))
model.add(Dropout(0.5))
model.add(Dense(5, activation='softmax'))
```

Result:
- Training accuracy and validation accuracy consistently improved over the epochs.
- Low difference between train and validation accuracy and accuracy>0.60 for both confirms no over-fitting/under-fitting happening.
- Using alternative frames helped the network to recognize gesture better.
- Adding more layers to the previous network helped improving the performance as well.

Exp. 6 : Changing Momentum of learning rate

Using the same model with updated momentum (from 0.7 to 0.9).
optimiser = optimizers.SGD(lr=0.001, decay=1e-6, momentum=0.9, nesterov=True)
Result:- Training accuracy and validation accuracy consistently improved over the epochs.
- Accuracy score was best for momentum = 0.9

Exp. 7 : Changing Image Size

Using the same model with (100X100) image frames
Result: Performance is almost same.

Exp. 8 : Changing Initial Learning Rate

optimiser = optimizers.SGD(lr=0.0001, decay=1e-6, momentum=0.9, nesterov=True)
Using the same model with (100X100) image frames
Result: Model slightly over-fitted.

Exp. 9 : Using All Image Frames

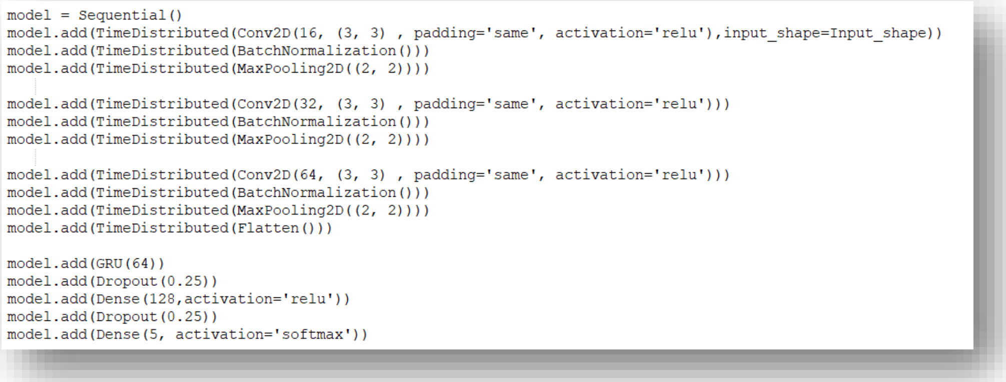
Using the same model with all 30 frames instead of 18 alternative frames.

Result: Model slightly over-fitted.

- This will increase the input size and processing time, however not necessary to improve accuracy.

Exp. 10 : Conv2D+GRU base model

- Using the following model as base for CNN+RNN architecture
- For each video frames from 11th to 29th index are fed to the network.



```
model = Sequential()
model.add(TimeDistributed(Conv2D(16, (3, 3) , padding='same', activation='relu'),input_shape=Input_shape))
model.add(TimeDistributed(BatchNormalization()))
model.add(TimeDistributed(MaxPooling2D((2, 2))))

model.add(TimeDistributed(Conv2D(32, (3, 3) , padding='same', activation='relu')))
model.add(TimeDistributed(BatchNormalization()))
model.add(TimeDistributed(MaxPooling2D((2, 2))))

model.add(TimeDistributed(Conv2D(64, (3, 3) , padding='same', activation='relu')))
model.add(TimeDistributed(BatchNormalization()))
model.add(TimeDistributed(MaxPooling2D((2, 2))))
model.add(TimeDistributed(Flatten()))

model.add(GRU(64))
model.add(Dropout(0.25))
model.add(Dense(128,activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(5, activation='softmax'))
```

-

Normalization done of by diving every pixel by 255.

- optimiser = optimizers.SGD(lr=0.001, decay=1e-6, momentum=0.7)

Result: No improvement in model performance.

- Model is under-fitting as both training and validation accuracy are poor.

Exp. 11 : Adding Layers

- Adding more layer to the previous model to improve learning.

Result: No improvement in model performance.

- Model is under-fitting as both training and validation accuracy are poor.

Exp. 12 : Changing Image Size

- Using (100X100) images

Result:- Model is under-fitting as both training and validation accuracy are poor.

Exp. 13 : Updating Momentum

- Using alternative image frame of size (84X84)
- optimiser = optimizers.SGD(lr=0.001, decay=1e-6, momentum=0.9)

Result:- Model performance improved, slightly over-fitting.

Exp. 14:

- Using alternative image frame of size (100X100)
- optimiser = optimizers.SGD(lr=0.001, decay=1e-6, momentum=0.9)

Result:- Model performance improved, slightly over-fitting.

Exp. 15 : Using All Frame

- Using all 30 image frames.
- optimiser = optimizers.SGD(lr=0.001, decay=1e-6, momentum=0.9)

Hand Gesture Recognition With CNN and RNN

Result:- Model performance improved, slightly over-fitting.