**Gesture Recognition - Case Study**

In this group project, you are going to build a 3D Conv model that will be able to predict the 5 gestures correctly. Please import the following libraries to get started.

**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

**By:**

* Venki Ramachandran
* Kiran Khairnar (Group Facilitator)

**In this case study we will build multiple models with different experiments with model , CNN layers, Transfer Learnings and image normalization and augmentation techniques.**

**Experiment 1: Generator function image resize**

**Decision:**

* Will use first 10 image frames from each video.
* We will not crop any image, will just resize the images to size 180x180.

**Input:**

* img\_idx = [0,1,2,3,4,5,6,7,8,9,10]
* batch size = 32
* x = 10
* y = 180
* z = 180

**Model:**

In this experiment we will be using 4 layers of Conv3D (Filters: 64,128,256,256). 3 BatchNormalization, 2 Dropout, 2 MaxPooling3D & 2 Dropout layers . Used “relu” as activation function in hidden layers and “softmax” in output layer. The first pooling layer before 2nd Conv3D layer has kernel size(1 ×2 ×2) and stride(1 ×2 ×2). This means that only the spatial pooling is performed on the first Conv3D layer. The second pooling layer has kernel size(2 ×2 ×2) and stride (2×2×2). This means that spatiotemporal pooling is performed on the second Conv3D layer. These two pooling layers make the output size of the 3-D CNN component shrank by the ratios of 4 and 2 on the spatial size and the temporal length respectively. It means that the 3-D CNN component only learns the short-term spatiotemporal features. At the end of model, we have used the 2 Dense layers with filter size 128 & 5(classes).

**Parameters:**

* Optimizer: SGD (lr=0.001)
* Epoch: 10
* Total params: 334,659,845
* Trainable params: 334,658,949
* Non-trainable params: 896

**Result:** Throws an error.

ResourceExhaustedError: OOM when allocating tensor with GPU

**Conclusion:**

*OOM* stands for "out of memory". Your GPU is running out of memory, so it can't allocate memory for this tensor. There are a few things we can do.

* Let’s try reducing the size of images in next experiment.
* Reduce the number of layers
* Use MaxPooling2D layers after convolutional layers
* Use a smaller batch\_size (or increase steps\_per\_epoch and validation\_steps)
* Decrease the number of filters in your Dense, Conv2D layers

**Experiment 2: Generator function image resize & crop**

**Decision:**

* Will use first 10 image frames from each video.
* Per problem statement there are 2 dimensions of images present in data. 360x360 and 120x160.
* We will crop 120x160 image to 120x120 to make it first symmetric in dimensions.
* Resize all input images to 80x80. As Conv3D is prone to error with different shapes in data.

**Input :**

* img\_idx = [0,1,2,3,4,5,6,7,8,9,10]
* batch size = 32
* x = 10
* y = 80
* z = 80

**Model:**

In this experiment we will be using 4 layers of Conv3D (Filters: 64,128,256,256). 3 BatchNormalization, 2 Dropout, 2 MaxPooling3D & 2 Dropout layers . Used “relu” as activation function in hidden layers and “softmax” in output layer. The first pooling layer before 2nd Conv3D layer has kernel size(1 ×2 ×2) and stride(1 ×2 ×2). This means that only the

spatial pooling is performed on the first Conv3D layer. The second pooling layer has kernel size(2 ×2 ×2) and stride (2×2×2). This means that spatiotemporal pooling is performed on the second Conv3D layer. These two pooling layers make

the output size of the 3-D CNN component shrank by the ratios of 4 and 2 on the spatial size and the temporal length respectively. It means that the 3-D CNN component only learns the short-term spatiotemporal features. At the end of model, we have used the 2 Dense layers with filter size 512 & 5.

**Parameters:**

* Optimizer: SGD (lr=0.001)
* Epoch: 10
* Total params: 265,030,149
* Trainable params: 265,029,253
* Non-trainable params: 896

**Result:** Training Accuracy: **0.78** Validation Accuracy: **0.23**

**Conclusion:**

* With the same model resizing the images from 180x180 to 80x80 has significantly reduced the model parameters . With reduced and 10 epochs we are getting decent 78% validation & 23% training accuracy.
* Accuracy difference between train and validation is high so the model we have is overfitted as we just input 10 images of video. We need to feed more data to the model to generalize.
* We also cropped the images 120x160 to 120x120 to making it properly centered to object striping some pixels from left and right of wider image.

**Experiment 3: Generator function increase number of images in input & reduce filters in model**

**Decision:**

* Will use first 18 image frames from each video.
* Video has total 30 image frames. We divided these 30 images into 3 slots each 10 images. slot1 [0:9], slot2 [10:19], slot3 [20:29] In each slot we will take 6 images will select 2 images in beginning 2 images in ending and 2 images in middle so that in each slot we will have 6 so total 18 images.
* Reduce filters in Conv3D layers.

**Input :**

* img\_idx = [0,1,4,5,8,9,10,11,14,15,18,19,20,21,24,25,28,29]
* batch size = 32
* x = 18
* y = 80
* z = 80

**Model:**

In this experiment we will be using 4 layers of Conv3D (Filters: 32,64,128,256). 3 BatchNormalization, 2 Dropout, 2 MaxPooling3D & 2 Dropout layers . Used “relu” as activation function in hidden layers and “softmax” in output layer. The first pooling layer before 2nd Conv3D layer has kernel size(1 ×2 ×2) and stride(1 ×2 ×2). This means that only the

spatial pooling is performed on the first Conv3D layer. The second pooling layer has kernel size(2 ×2 ×2) and stride (2×2×2). This means that spatiotemporal pooling is performed on the second Conv3D layer. These two pooling layers make

the output size of the 3-D CNN component shrank by the ratios of 4 and 2 on the spatial size and the temporal length respectively. It means that the 3-D CNN component only learns the short-term spatiotemporal features. At the end of model, we have used the 2 Dense layers with filter size 512 & 5.

**Parameters**:

* Optimizer: SGD (lr=0.001)
* Epoch: 10
* Total params: 473,027,973
* Trainable params: 473,027,269
* Non-trainable params: 704

**Result:** Model did not fit due to memory exhaust error.

**Conclusion:**

* Too many parameters not able to train the model. Time taken is huge.
* We will reduce the size of image keeping input images more. Keeping maximum data feed to model.

**Experiment 4: Generator function increase number of images in input , reduce the image size & reduce filters in model**

**Decision:**

* Will use 18 image frames from each video.
* Video has total 30 image frames. We divided these 30 images into 3 slots each 10 images. slot1 [0:9], slot2 [10:19], slot3 [20:29] In each slot we will take 6 images will select 2 images in beginning 2 images in ending and 2 images in middle so that in each slot we will have 6 so total 18 images.
* Reduce the image size from 80x80 to 60x60
* Reduced filters in Conv3D layers.

**Input :**

* img\_idx = [0,1,4,5,8,9,10,11,14,15,18,19,20,21,24,25,28,29]
* batch size = 64
* x = 18
* y = 60
* z = 60

**Model:**

In this experiment we will be using 4 layers of Conv3D (Filters: 32,64,128,256). 3 BatchNormalization, 2 Dropout, 2 MaxPooling3D & 2 Dropout layers . Used “relu” as activation function in hidden layers and “softmax” in output layer. The first pooling layer before 2nd Conv3D layer has kernel size(1 ×2 ×2) and stride(1 ×2 ×2). This means that only the

spatial pooling is performed on the first Conv3D layer. The second pooling layer has kernel size(2 ×2 ×2) and stride (2×2×2). This means that spatiotemporal pooling is performed on the second Conv3D layer. These two pooling layers make

the output size of the 3-D CNN component shrank by the ratios of 4 and 2 on the spatial size and the temporal length respectively. It means that the 3-D CNN component only learns the short-term spatiotemporal features. At the end of model, we have used the 2 Dense layers with filter size 512 & 5.

**Parameters**:

* Optimizer: SGD (lr=0.001)
* Epoch: 30
* Total params: 268,306,949
* Trainable params: 268,306,053
* Non-trainable params: 896

**Result:** Model did not fit due to memory exhaust error.

**Conclusion:**

* There was significant reduction in model parameters after reducing image size.
* However, model did not fit due high number of images GPU can take. We got the OOM error.
* Let’s reduce the number of images in next experiments.

**Experiment 5(a): Generator function maximize batch size such way that we are able use GPU to full capacity.**

**Decision:**

* Will use 15 image frames from each video.
* Video has total 30 image frames. We will use alternate image from 30 images.
* Will keep image size to 60x60
* Increase batch size to 128 then 96

**Input :**

* img\_idx = [0,2,4,6,8,10,12,14,16,18,20,22,24,26,28]
* batch size = 128 or 96
* x = 15
* y = 60
* z = 60

**Model:**

In this experiment we will be using 4 layers of Conv3D (Filters: 64,128,256,256). 3 BatchNormalization, 2 Dropout, 2 MaxPooling3D & 2 Dropout layers . Used “relu” as activation function in hidden layers and “softmax” in output layer. The first pooling layer before 2nd Conv3D layer has kernel size(1 ×2 ×2) and stride(1 ×2 ×2). This means that only the

spatial pooling is performed on the first Conv3D layer. The second pooling layer has kernel size(2 ×2 ×2) and stride (2×2×2). This means that spatiotemporal pooling is performed on the second Conv3D layer. These two pooling layers make

the output size of the 3-D CNN component shrank by the ratios of 4 and 2 on the spatial size and the temporal length respectively. It means that the 3-D CNN component only learns the short-term spatiotemporal features. At the end of model, we have used the 2 Dense layers with filter size 512 & 5.

**Parameters**:

* Model failed to build, thrown OOM error

**Result:** Throws an error.

OOM when allocating tensor with shape[460800,512] and type float on /job:localhost/replica:0/task:0/device:GPU:0 by allocator GPU\_0\_bfc [Op:Add]

**Conclusion:**

*OOM* stands for "out of memory". Your GPU is running out of memory, so it can't allocate memory for this tensor. There are a few things we can do.

* Use a smaller batch\_size (or increase steps\_per\_epoch and validation\_steps)

**Experiment 5(b): Generator function use batch size greater than 32 and less than 96 and also reduced the filters in Conv3D layers.**

**Decision:**

* Set the batch size to 64
* Set the Epoch to 30

**Input :**

* img\_idx = [0,2,4,6,8,10,12,14,16,18,20,22,24,26,28]
* batch size = 64
* x = 15
* y = 60
* z = 60

**Model:**

In this experiment we will be using 4 layers of Conv3D (Filters: 32,64,128,128). 3 BatchNormalization, 2 Dropout, 2 MaxPooling3D & 2 Dropout layers . Used “relu” as activation function in hidden layers and “softmax” in output layer. The first pooling layer before 2nd Conv3D layer has kernel size(1 ×2 ×2) and stride(1 ×2 ×2). This means that only the

spatial pooling is performed on the first Conv3D layer. The second pooling layer has kernel size(2 ×2 ×2) and stride (2×2×2). This means that spatiotemporal pooling is performed on the second Conv3D layer. These two pooling layers make

the output size of the 3-D CNN component shrank by the ratios of 4 and 2 on the spatial size and the temporal length respectively. It means that the 3-D CNN component only learns the short-term spatiotemporal features. At the end of model, we have used the 2 Dense layers with filter size 256 & 5.

**Parameters**:

* Optimizer: SGD (lr=0.001)
* Epoch: 30
* Total params: 59,706,629
* Trainable params: 59,706,181
* Non-trainable params: 448

**Result:** Training Accuracy: **0.82** Validation Accuracy:  **0.32**

**Conclusion:**

* We tried batch size 128 we got the OOM error on GPU.
* We tried batch size 96 we got the OOM error on GPU.
* So, decided to use batch size 64 which is greater than 32 which worked with out any issue and less than 96.
* We wanted to use high batch size as much as possible. Larger the batch size better would be approximation. The advantage of using large batch size is it would make evaluation/prediction process much faster.
* From this experiment we see good accuracy in training datasets but less accuracy in validation again model is not generalized.
* However, it’s too early to say the model is good because we could use only 15 image frames out of 30.

**Experiment 6: CCN + RNN Model with LSTM.**

**Decision:**

* Will use 15 image frames from each video.
* Video has total 30 image frames. We will use alternate image from 30 images.
* Will keep image size to 60x60
* batch size will be 64.
* The recurrent connections often offer advantages. They make every unit to use their context information and especially in image recognition tasks this is very helpful. As the time steps increase, the unit gets influenced by larger and larger neighborhood. With that information recurrent networks can watch large regions in the input space. In CNN this ability is limited to units in higher layers.
* Recurrent connections yield to an ability of handling sequential data.
* We will be using LSTM to solve vanishing gradient problem or exploding gradient problem.

**Input :**

* img\_idx = [0,2,4,6,8,10,12,14,16,18,20,22,24,26,28]
* batch size = 64
* x = 15
* y = 60
* z = 60

**Model:**

In this experiment we will be using 4 layers of Conv2D (Filters: 16,32,64,128). 4 BatchNormalization, 2 Dropout, 4 MaxPooling2D & 2 Dropout layers . Used “relu” as activation function in hidden layers and “softmax” in output layer. Using filter-size (3,3) in Conv2D and (2,2) in MAxPooling2D layers. At the end of model, we have used the 2 Dense layers with filter size 64 & 5. Will be using 64 LSTM cells in LSTM layer.

**Parameters**:

* Optimizer: SGD (lr=0.001)
* Epoch: 30
* Total params: 414,437
* Trainable params: 413,957
* Non-trainable params: 480

**Result:** Training Accuracy: **0.42** Validation Accuracy:  **0.18**

**Conclusion:**

* In 30 epochs we did not get any good accuracy on train and validation datasets.
* Model parameters are not high in numbers.
* We will also increase the image size this time to 120x120 and if the increasing image size helps improving accuracy

**Experiment 7: CNN + RNN Model(LSTM) with increased image size.**

**Decision:**

* Will use 15 image frames from each video.
* Video has total 30 image frames. We will use alternate image from 30 images.
* Will increase image size to 80x80
* batch size will be 64.
* We will use LSTM in this model like experiment 6.

**Input :**

* img\_idx = [0,2,4,6,8,10,12,14,16,18,20,22,24,26,28]
* batch size = 64
* x = 15
* y = 80
* z = 80

**Model:**

In this experiment we will be using 4 layers of Conv2D (Filters: 16,32,64,128). 4 BatchNormalization, 2 Dropout, 4 MaxPooling2D & 2 Dropout layers . Used “relu” as activation function in hidden layers and “softmax” in output layer. Using filter-size (3,3) in Conv2D and (2,2) in MAxPooling2D layers. At the end of model, we have used the 2 Dense layers with filter size 64 & 5.Will be using 64 LSTM cells in LSTM layer.

**Parameters**:

* Optimizer: SGD (lr=0.001)
* Epoch: 30
* Total params: 938,725
* Trainable params: 938,245
* Non-trainable params: 480

**Result:** Training Accuracy: **0.55** Validation Accuracy:  **0.34**

**Conclusion:**

* In 30 epochs we did not get any good accuracy on train and validation datasets.
* Increase image size did not helped much improvement in model accuracy.
* Model parameters are higher than compare to last experiment 6.
* In next experiments let’s try using GRU.

**Experiment 8: CNN + RNN Model with GRU**

**Decision:**

* Will use 15 image frames from each video.
* Video has total 30 image frames. We will use alternate image from 30 images.
* Will increase image size to 120x120
* batch size will be 64.
* We will use GRU in this model.
* Finally, you briefly saw an LSTM variant - the gated recurrent unit (GRU). A GRU network consists of GRU layers which consist of GRU cells which are like LSTM cells. However, the GRU has fewer parameters than an LSTM network. A GRU has three weight matrices as compared to the four in an LSTM layer.
* This means that a GRU has 3x parameters than a vanilla RNN layer.

**Input :**

* img\_idx = [0,2,4,6,8,10,12,14,16,18,20,22,24,26,28]
* batch size = 64
* x = 15
* y = 120
* z = 120

**Model:**

In this experiment we will be using 4 layers of Conv2D (Filters: 16,32,64,128). 4 BatchNormalization, 2 Dropout, 4 MaxPooling2D & 2 Dropout layers . Used “relu” as activation function in hidden layers and “softmax” in output layer. Using filter-size (3,3) in Conv2D and (2,2) in MAxPooling2D layers. At the end of model, we have used the 2 Dense layers with filter size 64 & 5.Will be using 64 GRU cells in GRU layer.

**Parameters**:

* Optimizer: SGD (lr=0.001)
* Epoch: 30
* Total params: 1,319,781
* Trainable params: 1,319,301
* Non-trainable params: 480

**Result:** Training Accuracy: **0.89** Validation Accuracy:  **0.38**

**Conclusion:**

* In 30 epochs we did not get any good accuracy on validation datasets.
* Model is overfitting.
* Model parameters were higher than previous experiment.
* In next experiments let’s try GRU with transfer learning VGG16 model.

**Experiment 9: Transfer Learning with VGG16 and GRU.**

**Decision:**

* Will use 15 image frames from each video.
* Video has total 30 image frames. We will use alternate image from 30 images.
* Will use image shape to 120x120
* batch size will be 64.
* We will use heavy duty transfer learning architectures like VGG16 with GRU in this model.

**Input :**

* img\_idx = [0,2,4,6,8,10,12,14,16,18,20,22,24,26,28]
* batch size = 64
* x = 15
* y = 120
* z = 120

**Model:**

In this experiment we will be using heavy duty VGG16 architecture. At the end of model, we have used the 2 Dense layers with filter size 8 & 5.Will be using 2 GRU layers in each layer we will be using 32 & 16 GRU cells respectively.

**Parameters**:

* Optimizer: SGD (lr=0.001)
* Epoch: 30
* Total params: 15,021,653
* Trainable params: 306,965
* Non-trainable params: 14,714,688

**Result:** Training Accuracy: **0.26** Validation Accuracy:  **0.27**

**Conclusion:**

* In 30 epochs we did not get any good accuracy on validation and train datasets. But it’s underfitting.
* Model trainable parameters were lower than compare to experiment 8.
* In next experiments let’s try MobileNet transfer learning with LSTM.

**Experiment 10: Transfer Learning with MobileNet and LSTM.**

**Decision:**

* Will use 15 image frames from each video.
* Video has total 30 image frames. We will use alternate image from 30 images.
* Will use image shape to 120x120
* batch size will be 64.
* We will use light weight transfer learning architectures like MobileNet with LSTM in this model.

**Input :**

* img\_idx = [0,2,4,6,8,10,12,14,16,18,20,22,24,26,28]
* batch size = 64
* x = 15
* y = 120
* z = 120

**Model:**

In this experiment we will be using light weight MobileNet architecture. At the end of model, we have used the 2 Dense layers with filter size 64 & 5.Will be using LSTM layers with 64 LSTM cells.

**Parameters**:

* Optimizer: SGD (lr=0.001)
* Epoch: 30
* Total params: 3,840,453
* Trainable params: 609,541
* Non-trainable params: 3,230,912

**Result:** Training Accuracy: **0.43** Validation Accuracy:  **0.51**

**Conclusion:**

* In 30 epochs we did not get any good accuracy on validation datasets. But it’s not overfitting.
* Model trainable parameters were slightly lower than compare to experiment 9 with VGG16 and GRU.
* Let’s try image augmentation as models in previous experiments are not performing well. Some models are not generalized hence overfitting.

**Experiment 11: CNN + RNN Model(LSTM) with augmentation image size.**

**Decision:**

* Will use 15 image frames from each video.
* Video has total 30 image frames. We will use alternate image from 30 images.
* Will keep image size to 120x120.
* We will use image cropping augmentation in this experiment. Let’s see if this solves the model overfitting problem in experiment 8.
* batch size will be 64.
* We will use LSTM in this model.

**Input :**

* img\_idx = [0,2,4,6,8,10,12,14,16,18,20,22,24,26,28]
* batch size = 64
* x = 15
* y = 120
* z = 120

**Model:**

In this experiment we will be using 4 layers of Conv2D (Filters: 16,32,64,128). 4 BatchNormalization, 2 Dropout, 4 MaxPooling2D & 2 Dropout layers . Used “relu” as activation function in hidden layers and “softmax” in output layer. Using filter-size (3,3) in Conv2D and (2,2) in MAxPooling2D layers. At the end of model, we have used the 2 Dense layers with filter size 64 & 5.Will be using 128 LSTM cells in LSTM layer.

**Parameters**:

* Optimizer: SGD (lr=0.001)
* Epoch: 30
* Total params: 3,392,869
* Trainable params: 3,392,389
* Non-trainable params: 480

**Result:** Training Accuracy: **0.55** Validation Accuracy:  **0.27**

**Conclusion:**

* In 30 epochs I did not get any good accuracy on train and validation datasets.
* Model parameters are higher than compare to last experiment 10. Less parameters didn’t help.
* In next experiments let’s try using GRU with augmentation. Will crop and remove the noise from image.

**Final Experiment 12: Transfer Learning with MobileNet and LSTM with more augmentation.**

**Decision:**

* Will use 15 image frames from each video.
* Video has total 30 image frames. We will use alternate image from 30 images.
* Will keep image size to 120x120.
* We will use more image augmentation like rotation along with cropping and shifting of image in this experiment. Let’s see if this solves the model overfitting problems in previous experiments.
* batch size will be 64.
* We will use LSTM in this model.

**Input :**

* img\_idx = [0,2,4,6,8,10,12,14,16,18,20,22,24,26,28]
* batch size = 64
* x = 15
* y = 120
* z = 120

**Model:**

In this experiment we will be using light weight MobileNet architecture. At the end of model, we have used the 2 Dense layers with filter size 64 & 5.Will be using LSTM layers with 64 LSTM cells.

**Parameters**:

* Optimizer: SGD (lr=0.001)
* Epoch: 20
* Total params: 3,516,229
* Trainable params: 285,317
* Non-trainable params: 3,230,912

**Result:** Training Accuracy: **0.97** Validation Accuracy:  **0.81**

**Conclusion:**

* In 20 epochs we did get good accuracy on train and validation datasets.
* Model parameters are lower than compare to last experiments.
* This is the best model with less parameters and high accuracy.

**Summary:**

**We tried several experiments with high number of images we got the GPU OOM error. We tried a smaller number of images did not help in improving accuracy. We tried reducing the filter sizes to reduce the number of parameters but did not help improving the accuracy either. Interestingly we got decently balance in accuracy with VGG16 & MobileNet. In final we decided to use MobileNet with LSTM that’s gave the best accuracy and least parameters in all the experiments.**

**Final Model Accuracy: Train 97% and Val 81% .**

