Gesture Recognition

# Problem Statement

A home electronics company which manufactures state of the art smart televisions, wants to develop a cool feature for a smart TV which is to recognize 5 different hand gestures which helps users control the TV without remote control.

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

# Details of the dataset

There are 663 videos recorded for training, each video is divided into 30 frames. There are 100 videos of 30 frames each to validate the accuracy of the solution.

# Possible Solutions

Using simple machine learning techniques like logistic regression can only be used for image classification, here we are dealing with large volumes of sequential data so we should be using more robust learning techniques which can deal with large amounts of sequential data, like neural networks.

2D Convolutional neural networks which are typically used for image classification cannot be used with sequence/time (the 3rd dimension), hence we should use a sequence learning Neural network architecture like RNN or use 3D convolution techniques. In this project we will use both.

First let us look at some of the pre-processing techniques that are applicable to this project.

# Pre-processing

Preprocessing of data aligns with data oriented approached, systematically cleaning and augmenting the data based on the nature of the problem we are dealing with allow the model to learn better from the data, avoid learning noise and generalize better. Traditionally there are 2 types of transformations that can be applied to images linear transformations (crop, flip etc.) and affine transformations like rotate, transform. We cannot flip the images as the intent of the gesture changes with flip so we will use resize and crop for reduce the training time and affine transformations which is going to squeeze the image a little bit.

In Summary, the pre-processing we apply per image are: crop, resize and transform (conditionally)

# Generators.

Data generators are used to feed a continuous structured stream of data to the models so that they can learn per batch, per epoch. Keras has built in generators but in this case, we are using custom generators which creates stream of batches of configurable batch size, the number of images that can be taken from the folder.

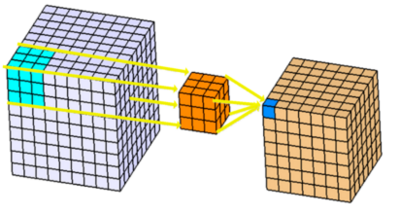
# Experiments.

Experiments are conducted in 3 stages, for each configuration of the model, once the model architecture is fixed, the below process is followed.

* Ablation experiments with small portion of data is done to learn if the model works well and there are no errors while training.
* Overfitting experiments are conducted with small portion (sometimes full data) for a smaller number of epochs to learn if the loss is coming down and if the model can learn from training data. At this point huge variations between training accuracy and validation accuracy are expected.
* Final experiments are conducted with full data with higher epochs, to reduce the learning time, better reporting and make it more robust the following callbacks are created.
  + Early Stopping – The model training stops if there is no improvement in validation loss after 5 epochs.
  + Reduce LR on Plateau – The model’s learning rate is reduced by a factor of 0.1 if the validation loss does not improve after 5 epochs.
  + Checkpoint – The best model based on delta validation loss is saved to disk.

# Solution 1: 3D Convolution

3 D convolution is one of the deep learning architectures that can be used here, it is a natural extension to 2D convolution. In a 2D convolution each filter concentrations on a particular region of the image and generates features, these features used by the next layers to create higher level abstractions. In 3D convolution the same concept is applied, but instead to a batch of images. The below image explains the idea graphically.



(Image source - https://www.mdpi.com/2072-4292/9/11/1139/htm)

Using 3D convolutions, I did a series of ablation experiments, in which I include only some part of the data and vary the parameters to understand how the model learns and use the best combination based on full data and higher number of epochs. The below table explains each step of the experiment and the results. Please note that we start with a basic architecture and add layers, regularization based on the outcome of each experiment.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Experiment Type** | **Details** | **Validation Accuracy** | **# of trainable parameters** | **Time per epoch** | **What did we learn from experiment?** |
| 2 Conv layers  Ablation Experiment | Architecture: [32 Conv(3,3,3), maxpool(2,2,2), 64 conv(2,2,2), maxpooling(2,2,2), flatten, dense(256), dense(256), softmax  Activation=relu, optimizer=sgd | NA | 27,644,293 | NA | The model looks correct. |
| 2 Conv layers  Overfitting Intentionally | Architecture: [32 Conv(3,3,3), maxpool(2,2,2), 64 conv(2,2,2), maxpooling(2,2,2), flatten, dense(256), dense(256), softmax  Activation=relu, optimizer=sgd | NA |  | NA | The training accuracy increased steadily, and loss decreased. There was overfitting observed. |
| 2 Conv layers  Experiment with full data | Architecture: [32 Conv(3,3,3), maxpool(2,2,2), 64 conv(2,2,2), maxpooling(2,2,2), flatten, dense(256), dense(256), softmax, Activation=relu  Batch size = 30  Epochs = 25  Image dimensions = 120,120  Samples per image = 15 | 64% |  | 23s | The training accuracy = 85%, overfitting observed. The training accuracy should improve further. |
| 3 Conv layers  Overfitting Intentionally | Architecture: [32 Conv(3,3,3), maxpool(2,2,2), 64 conv(3,3,3), maxpooling(2,2,2), 128 conv(2,2,2), maxpool(2,2,2) flatten, dense(256), dense(256), softmax, optimizer = sgd | NA | 5,728,773 | NA | The model learning rate increases steadily, less overfitting observed. |
| 3 Conv layers  Final Experiment | Architecture: [32 Conv(3,3,3), maxpool(2,2,2), 64 conv(3,3,3), maxpooling(2,2,2), 128 conv(2,2,2), maxpool(2,2,2) flatten, dense(256), dense(256), softmax, optimizer=sgd  Batch size = 30  Epochs = 25  Image dimensions = 120,120  Samples per image = 20 | 46% |  | 29s | The training accuracy is 76% which is lesser than 2-layer model. Overfitting observed. |
| 3 Conv layers  Ablation Experiment with Batch Normalization | Architecture: [32 Conv(3,3,3), maxpool(2,2,2), batchnormalization, 64 conv(3,3,3), batchnormalization, maxpooling(2,2,2), 128 conv(2,2,2), ,batchnormalization, maxpool(2,2,2) , flatten, dense(256), dense(256), softmax, optimizer sgd | NA | 12,998,021 | NA | The model learns faster, in 3 epochs training accuracy climbed to 86% but the validation accuracy was too low which means a the model overfit. |
| 3 Conv layers  Full Experiment with Batch Normalization | Architecture: [32 Conv(3,3,3), maxpool(2,2,2), batchnormalization, 64 conv(3,3,3), batchnormalization, maxpooling(2,2,2), 128 conv(2,2,2), ,batchnormalization, maxpool(2,2,2) , flatten, dense(256), dense(256), softmax, optimizer sgd  =======================================  Batch size = 30  Epochs = 25  Image dimensions = 120,120  Samples per image = 30 | 18% |  | 43 | 91% training accuracy, the model has heavily overfit. |
| 3 Conv Layers Ablation Experiment with Batch Normalization and Dropouts (Optimizer changed to ADAM) | Architecture: [32 Conv(3,3,3), maxpool(2,2,2), batchnormalization, 64 conv(3,3,3), batchnormalization, maxpooling(2,2,2), 128 conv(2,2,2), ,batchnormalization, maxpool(2,2,2) , flatten, dense(256), dropout(0.4) dense(256), dropout(0.4) softmax, optimizer = **ADAM**  =======================================  Batch size = 30  Epochs = 25  Image dimensions = 120,120 | NA | 1,760,965 | NA | A series of ablation experiments are conducted using different batch size, images per folder and image dimensions. From the above experiments we learn that the model learns better with increased samples per image, reduced dimensions have less impact on accuracy. Ablation experiments with batch size 30 without augmentation had highest training accuracy of 55. |
| 3 Conv Layers Full Experiment with Batch Normalization and Dropouts (Optimizer changed to ADAM) | Architecture: [32 Conv(3,3,3), maxpool(2,2,2), batchnormalization, 64 conv(3,3,3), batchnormalization, maxpooling(2,2,2), 128 conv(2,2,2), ,batchnormalization, maxpool(2,2,2) , flatten, dense(256), dropout(0.4) dense(256), dropout(0.4) softmax, optimizer = **ADAM**  =======================================  Batch size = 30  Epochs = 30  Image dimensions = 120,120  Samples per image = 25 | 20 | 1,760,741 | 74s | Training accuracy was in high 86, the model has overfit drastically despite dropouts. |
| 3 Conv Layers Full Experiment with Batch Normalization and Dropouts (Optimizer changed to ADAM) | Architecture: [32 Conv(3,3,3), maxpool(2,2,2), batchnormalization, 64 conv(3,3,3), batchnormalization, maxpooling(2,2,2), 128 conv(2,2,2), ,batchnormalization, maxpool(2,2,2) , flatten, dense(256), dropout(0.4) dense(256), dropout(0.4) softmax, optimizer = **ADAM**  =======================================  Batch size = 20  Epochs = 30  Image dimensions = 120,120  Samples per image = 30 | 92 | 1,760,741 | 86s | Good accuracy so far and less overfitting (training accuracy=98) so this is considered as the best model. |

Additional notes:

* The total time for experiment does not matter much as we are using the callbacks for optimizing the run. The training will stop if the targets are hit, hence in the table above only time per epoch is mentioned.
* Tried the image resolution 120x120 and 80x80, the accuracy dropped for 80x80 while there was only less difference in the time taken – 3sec per step. Accuracy is considered higher priority than time hence switched back to 120x120. No change in accuracy for 180x180 beyond that the kernel crashed.
* Activation function **sigmoid, tanh** for the 2 layered networks did not overfit, hence dropped.
* The data augmentation was not done initially, after learning that the model is not learning beyond 65% accuracy the data augmentation techniques were applied.

# Solution 2: CNN + RNN

The second type of architectures that can be used in this kind of problems is a combination of CNN + RNN or combinedly called CRNN. In this type of architecture CNN is a typical 2D CNN model which is responsible for learning the features from each image, this model does not consider the sequence but the out of this model is fed into an RNN model as a time distribution which learns and generalizes the gesture. The loss is computed at the output of the RNN and propagated all the way back to the CNN.

In order to build a CNN model which can learn from each image, there are 2 approaches, we can use any existing pre baked models like ResNET, MobileNet, VGGNet etc, this approach is called transfer learning. In transfer learning we download a pre-trained model, once download we can either reuse the model entirely or make the model unlearn the weights in the last layers (the last layers hold high level abstractions like object identification, the beginning layers contain lower-level abstractions like edges, curves detection which are common for any image classification). For this problem I used mobilenet and resnet50, mobilenet is more compact and since we need a model for an edge device like smart TV it is more suitable due to less layers and portability.

RNN models traditionally suffer from vanishing or exploding gradients a matured architecture like LSTM / GRU is usually preferred. In these experiments I tried both LSTM and GRU, GRU is preferable than LSTM due to lesser number of parameters.

I also tried to build a custom CNN model using Conv2D in combination with GRU.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Experiment Type** | **Details** | **Accuracy** | **# of trainable parameters.** | **Time** | **Analysis.** |
| Mobile Net + GRU  Ablation Experiment | Mobile Net (untrained last 10 layers), Flatten, GRU(8), Dense(256), softmax, optimizer = adam | NA | 3,453,877 | NA | A series of ablation experiments are conducted using different batch size, images per folder and image dimensions. From these experiments we learn that the model learns better with increased samples per image, batch size, reduced dimensions have less impact on accuracy.  The model is correct, less overfitting observed. |
| Mobile Net + GRU  Ablation Experiment with dropouts | Mobile Net (untrained last 10 layers), Flatten, GRU(8),dropout(0.4), Dense(256), dropout(0.4) softmax, optimizer = adam | NA | 18,306,757 | NA | Overfitting reduced. |
| Mobile Net + GRU full experiment with full data with dropouts | Mobile Net (untrained last 10 layers), Flatten, GRU(8),dropout(0.4), Dense(256), dropout(0.4), softmax, optimizer = adam  batch\_size=20  num\_epochs=25  dim=(120,120)  samples=20  rnncells=256 | 88 | 3,453,877 | 62 | Training accuracy is 90% so less overfitting. Another experiment was conducted with increased batch size and samples per image but there was less improvement in accuracy but the training per epoch increased to 80 seconds hence it is rejected. |
| ResnetNet50 + LSTM Ablation Experiments | Resnet50(untrained last 10 layers), Flatten, LSTM(512), Dense(256), softmax, optimizer = adam | NA | 68,292,101 | NA | 2 ablation experiments are conducted, with varying batch size, the experiment with higher batch size and more samples yielded better results.  The model is correct. There is overfitting observed. |
| ResnetNet50 + LSTM Ablation Experiments  With dropouts | Resnet50(untrained last 10 layers), Flatten, LSTM(512), dropout(0.4),Dense(256), dropout(0.4), softmax, optimizer = adam | NA | 68,292,101 | NA | Overfitting reduced. the model with less samples and batch size worked better, |
| Resnet50 + LSTM experiments with full data without dropouts | Resnet50(untrained last 10 layers), Flatten, LSTM(512), Dense(256), softmax, optimizer = adam  ablation\_size=None  batch\_size=20  num\_epochs=25  dim=(120,120)  samples=20  rnncells=512 | 44 | 68,292,101 | 58s | The model did not learn, it underfit. |
| Custom CNN Model + GRU ablation experiments | 32Conv2d(3,3), max2d(2,2), 64conv2d(3,3), max2d(2,2), 128conv2d(3,3), max2d(2,2), flatten, dense(512), dense(256), timedistrubuted, flatten(), gru(512), dense(256), dens(5), oprimizer= adam | NA | 16,286,021 | NA | The model learnt well, and it overfit. |
| Custom CNN Model + GRU + dropouts ablation experiments | 32Conv2d(3,3), max2d(2,2), 64conv2d(3,3), max2d(2,2), dropout(0.25),128conv2d(3,3), max2d(2,2), dropout(0.25), flatten, dense(512), dense(256), timedistrubuted, flatten(), gru(512), dense(256), dens(5), oprimizer= adam | NA | 16,286,021 | NA | The overfitting reduced with dropouts |
| Custom CNN Model + GRU with dropouts experiments with full data | 32Conv2d(3,3), max2d(2,2), 64conv2d(3,3), max2d(2,2), dropout(0.25),128conv2d(3,3), max2d(2,2), dropout(0.25), flatten, dense(512), dense(256), timedistrubuted, flatten(), gru(512), dense(256), dens(5), oprimizer= adam | 95 | 16,286,021 | 68s | Only one full experiment was conducted, the kernel crashed when tried with samples = 30 or batch size = 30 with OOM exception |

# Conclusion

The parameters that are chosen for choosing the best model are – accuracy and size (# of parameters) in the order of priority. Following the priority, the below are the best models.

1. Traditional CNN Model + GRU model with 95% accuracy.

Text

Description automatically generated with low confidence

1. 3D conv model with 92 % accuracy, # of parameters = 1,760,741

Table

Description automatically generated with low confidence

1. Mobile Net + GRU – 87% accuracy, # of parameters = 3,453,877

Text

Description automatically generated

## Predictions using the best model.

The below image shows the final prediction made on a batch randomly picked from validation data set.

Graphical user interface, text, application

Description automatically generated