# **Gesture Recognition Assignment**

**Group Facilitator :** **Name:** **Badari Prasad H**                           **Email ID:**badari.hp@gmail.com  
                             
**Team Member Detail:** **Name: Shantanu Saxena**  
                                **Email ID:**ssaxen@outlook.com

**MLC60 - Batch**

# **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.

* 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.

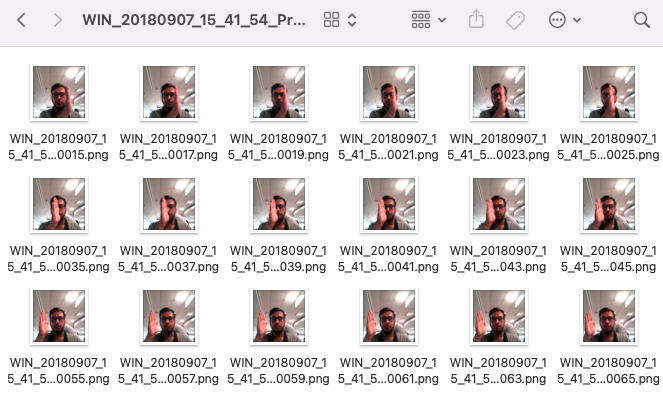
# **Data:**

<https://drive.google.com/uc?id=1ehyrYBQ5rbQQe6yL4XbLWe3FMvuVUGiL>

# **Understanding the provided dataset**

The training data consists of a few hundred videos categorised into one of the five classes. Each video (typically 2-3 seconds long) is divided into a **sequence of 30 frames(images)**. These videos have been recorded by various people performing one of the five gestures in front of a webcam - similar to what the smart TV will use.

# **Sample Data:**



# **Commonly used architecture for analysing videos:**

**Convolutions + RNN**  
The Conv2D network extracts a feature vector from each image, and a sequence of these feature vectors is then fed into an RNN-based network. The RNN's output is a regular softmax layer, suitable for classification problems like this one. Combining Convolutional Neural Networks (CNNs) with Recurrent Neural Networks (RNNs) results in a powerful model for processing sequences of images or videos.

* **Convolutional Layers (CNNs) for Image Feature Extraction**
  + CNNs excel at extracting features from images.
  + They use convolutional layers to detect patterns, shapes, and structures within the images.
  + Convolutional layers capture spatial information, recognizing edges, textures, and complex visual features.
* **RNN Integration for Sequential Understanding**
  + RNNs are designed for sequential data processing.
  + They consider the temporal sequence of images, treating each frame as a time step.
  + RNNs like LSTM or GRU can learn dependencies between sequential frames, capturing motion or changes across time.
* **Combining Convolution and Recurrent Layers**
  + CNNs process individual frames, extracting image features.
  + The output of the CNN is then fed into the RNN as sequential data, preserving spatial information while incorporating temporal understanding.
  + This combined architecture allows the model to learn both spatial features from images and temporal patterns across frames, making it suitable for tasks involving video analysis, action recognition, or sequential image data.

**3D Convolutional Network, or Conv3D**  
3D convolutions extend the concept of 2D convolutions by adding an extra dimension. In 2D convolutions, the filter moves in two directions (x and y). In 3D convolutions, the filter moves in three directions (x, y, and z). For example, if the input to a 3D convolution is a video consisting of 30 RGB images, each with a shape of 100x100x3, the video can be represented as a 4-D tensor with the shape 100x100x3x30. This can also be written as (100x100x30)x3, where 3 is the number of channels.

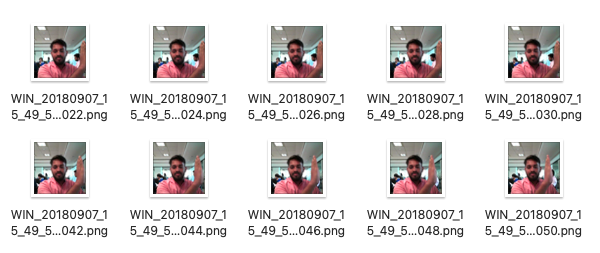
Drawing an analogy from 2D convolutions, where a 2D kernel/filter (a square filter) is represented as (f x f) x c (with f being the filter size and c the number of channels), a 3D kernel/filter (a cubic filter) is represented as (f x f x f) x c. Here, c = 3 since the input images have three channels. This cubic filter will then perform a 3D convolution on each of the three channels of the (100x100x30) tensor.

# **Data Generator details:**

The Data Generator is critically important within the codebase. Its main role is to pre-process images, specifically managing images of two different dimensions (360 x 360 and 120 x 160), and to facilitate the creation of video frame batches. A key expectation is that it ensures seamless batch processing of videos without errors. Successful execution involves essential steps such as cropping, resizing, and normalization, which are integral for effective image manipulation.

# **Data pre-processing steps:**

* **Resizing and Cropping:** This step optimizes gesture recognition in neural networks by removing distracting background noise from images. By precisely resizing and cropping the images, the model's focus on gestures is enhanced.
* **Normalization:** Normalizing RGB values in images is a powerful technique to reduce distortions caused by varying lighting conditions and shadows. This process ensures consistency and accuracy in data representation for the model.
* **Data Augmentation for Improved Accuracy:** To enhance the model's precision, data augmentation techniques are employed at advanced stages. This involves slight rotations of pre-processed gesture images, generating additional training data. This augmentation not only improves the model's accuracy but also broadens its ability to generalize across diverse hand positions, addressing scenarios where hand positioning may not consistently align within the camera frame.



# **Approach**

**Model Configuration and Hyper-parameters Experimentation**

* Tried various configurations: batch sizes, image dimensions, filter sizes, padding, and stride length
* Tested different learning rates; used ReduceLROnPlateau to adjust learning rate based on val\_loss

**Optimizer Experimentation**

* Tried SGD() and Adam() optimizers
* Opted for Adam() due to its ability to improve model accuracy by addressing parameter variance

**Usage of Additional Layers**

* Incorporated Batch Normalization, pooling, and dropout layers
* Deployed when model showed signs of overfitting, evident from high training accuracy but poor validation accuracy

**Early Stopping Implementation**

* Utilized early stopping to halt training when val\_loss saturated or model performance plateaued

# **Some Observations**

* **Trainable Parameters Impact Training Time**
  + Increased parameters lead to longer training durations.
* **Batch Size and GPU Memory**
  + Batch size linked to available GPU memory
  + Adjusted batch size to match GPU capacity (NVIDIA RTX A6000)
* **Batch Size vs. Training Time and Model Accuracy**
  + Larger batch size reduces training time but lowers model accuracy
  + Trade-off between speed and accuracy: larger batch for quicker results, smaller for better accuracy
* **Overcoming Overfitting**
  + Data Augmentation and Early Stopping countered initial overfitting issues
* **Transfer Learning Impact (Optional)**
  + MobileNet architecture utilized for enhanced accuracy
  + Chosen for its lightweight design, speed, and low maintenance compared to other architectures like VGG16, AlexNet, GoogleNet
* **Detailed Information**
  + Below table contains comprehensive Observations and Inferences

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Model Experimentation Table and Model Results** | | | | | | | | | | | | | |
| **S.No** | **Model** | **Filter size** | **frame\_height \* frame\_width** | **Batch**  **size** | **No.of**  **Frames** | **Epochs** | **Augmentation** | **Normalize** | **Optimizer** | **Total params** | **Trainable params** | **Non-trainable params** | **Model result** |
| 1 | Conv3D | (3,3,3) | 120 X 120 | 32 | 16 | 20 | FALSE | TRUE | Adam | 11,30,565 | 11,29,829 | 736 | Training Accuracy: 98%  Validation Accuracy: 34%  Clearly overfitting.  Trying with fine tuning in next step with change in filter size. |
| 2 | (2,2,2) | 120 X 120 | 32 | 16 | 20 | FALSE | TRUE | Adam | 9,25,365 | 9,24,629 | 736 | Training Accuracy: 98%  Validation Accuracy: 16%  Clearly overfitting.  Trying with fine tuning in next step with change in Augmentation |
| 3 | (3,3,3) | 120 X 120 | 32 | 16 | 20 | TRUE | TRUE | Adam | 7,03,493 | 7,02,885 | 608 | Training Accuracy: 97.73%  Validation Accuracy: 31.99%  Clearly overfitting.  Trying with fine tuning in next step with change in Normalization |
| 4 | (3,3,3) | 120 X 120 | 32 | 16 | 20 | TRUE | FALSE | Adam | 7,03,493 | 7,02,885 | 608 | Training Accuracy : 99%, Validation Accuracy: 86%  Parameter count is too high.  Accuracy looks good.  Trying with fine tuning in next step with change in optimizer to sgd. |
| 5 | (3,3,3) | 120 X 120 | 32 | 16 | 20 | TRUE | FALSE | SGD | 7,03,493 | 7,02,885 | 608 | Training Accuracy : 97.5%, Validation Accuracy: 82%  Parameter count is too high with sgd.  Accuracy looks good.  Trying with fine tuning in next step with change in frame size as 100x100. |
| 6 | (3,3,3) | 100 X 100 | 32 | 16 | 20 | TRUE | FALSE | Adam | 5,96,997 | 5,96,389 | 608 | Training Accuracy : 99%, Validation Accuracy: 70%  Accuracy looks good .  Parameter count is bit decreased compared to earlier models  Try with fine tuning by change in optimizer as sgd |
| 7 | (3,3,3) | 100 X 100 | 32 | 16 | 20 | TRUE | FALSE | Sgd | 5,96,997 | 5,96,389 | 608 | Training Accuracy : 98%, Validation Accuracy: 75%  Accuracy looks good .  No Major change as compared to previous model.  Try with fine tuning with Augmentation and Normalization set to “FALSE”. |
| 8 | (3,3,3) | 120 X 120 | 32 | 20 | 30 | FALSE | FALSE | Adam | 16,66,789 | 16,66,677 | 112 | Training Accuracy : 99%, Validation Accuracy: 78%.  Accuracy looks good with frame size as 120x120 and will consider it as baseline model since most of the previous models were considered with 120x120 as frame size.  Sub Optimal Model #4  with Conv3d  Try with fine tuning by changing activation function using “LeakyRelu” and change in frame size as 100x100 and number of frames as 15. |
| 9 | Conv3D with LeakyRelu | (3,3,3) | 100X100 | 32 | 15 | 30 | TRUE | TRUE | Adam | 7,72,42,693 | 7,72,42,373 | 320 | Training Accuracy : 99%, Validation Accuracy: 43%,  Parameter count is too high which will impact model performance.  Clearly Overfitting  Try with fine tuning by changing optimizer as sgd and Augmentation to “FALSE”. |
| 10 | (3,3,3) | 100X100 | 32 | 15 | 30 | FALSE | TRUE | Sgd | 7,72,42,693 | 7,72,42,373 | 320 | Training Accuracy : 99%, Validation Accuracy: 76%,  Parameter count is huge and not much improvement as earlier model.  Clearly Overfitting  Try with fine tuning by changing Augmentation and Normalization to “FALSE”. |
| 11 | (3,3,3) | 100X100 | 32 | 15 | 30 | FALSE | FALSE | Sgd | 7,72,42,693 | 7,72,42,373 | 320 | Training Accuracy : 99%, Validation Accuracy: 71%  Accuracy looks good  Parameter count is huge and will not be considering with frame size as 100x100.  Try with fine tuning by other architectures involving transfer learning / GRU. |
| **S.No** | **Model** | **Filtersize** | **frame\_height \* frame\_width** | **Batch**  **size** | **No.of**  **Frames** | **Epochs** | **Augmentation** | **Normalize** | **Optimizer** | **Total params** | **Trainable params** | **Non-trainable params** | **Model result** |
| 12 | Time distributed Conv2D + LSTM | (2,2) | 120X120 | 20 | 20 | 100 | FALSE | FALSE | Adam | 2,12,807 | 2,11,943 | 864 | Training Accuracy : 98.3%, Validation Accuracy: 89%,  Low params with good accuracy  Optimal model2 with less params and good accuracy than compared to Conv3D which is declared as Optimal fit as earlier [Refer to S. No 8].  Try with fine tuning using GRU and changing no.of frames as 16. |
| 13 | CNN + GRU | (2,2) | 120X120 | 50 | 16 | 100 | FALSE | FALSE | Adam | 3,27,029 | 3,25,653 | 1,376 | Training Accuracy : 99%, Validation Accuracy: 85%,  GRU can be considered as one of Optimal model.  Optimal Model3  Try with fine tuning using Transfer learning architecture with activation function as “tanh”. |
| 14 | Transfer learning with Mobile net + GRU |  | 120X120 | 40 | 20 | 100 | FALSE | TRUE | Adam | 34,46,725 | 34,22,789 | 23,936 | Training Accuracy : 99%, Validation Accuracy: 93%,  Optimal mode1l with good accuracy rate.  Try with fine tuning with other transfer learning architecture VGG16 along with LSTM for checking more improved model. |
| 15 | Transfer learning with VGG16 + LSTM |  | 120X120 | 50 | 18 | 100 | FALSE | TRUE | Adam | 1,49,76,933 | 26,22,053 | 1,23,54,880 | Training Accuracy : 50%, Validation Accuracy: 70%  Too high parameter count  Clearly underfitting. |

# **Top 3 Chosen optimal models – Summary Table (sorted by trainable parameters which we used)**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **S.No** | **Model** | **Filter**  **size** | **frame\_height \* frame\_width** | **Batch**  **size** | **No.of**  **Frames** | **Epochs** | **Augmentation** | **Normalize** | **Optimizer** | **Total params** | **Trainable params** | **Non-trainable params** | **Model result** |
| 1 | Time distributed Conv2D + LSTM | (2,2) | 120X120 | 20 | 20 | 100 | FALSE | FALSE | Adam | 2,12,807 | 2,11,943 | 864 | Optimal model2  Training Accuracy : 98.3%, Validation Accuracy: 89%,  Least params with good accuracy |
| 2 | CNN + GRU | (2,2) | 120X120 | 50 | 16 | 100 | FALSE | FALSE | Adam | 3,27,029 | 3,25,653 | 1,376 | Optimal Model3  Training Accuracy : 99%, Validation Accuracy: 82.99%,  Next least params out of top 4 models we have |
| 4 | Transfer learning with Mobile net + GRU |  | 120X120 | 40 | 20 | 100 | FALSE | TRUE | Adam | 34,46,725 | 34,22,789 | 23,936 | Optimal mode1  Training Accuracy : 99%, Validation Accuracy: 93%,  Highest accuracy, but higher no. of params. |

# **Top 3 Chosen optimal models and inferences**

**#1. Transfer learning with Mobile net + GRU**

This model achieved the highest training accuracy at 99% and a validation accuracy of 93%. However, it also had the highest number of parameters among the top three models we evaluated. Therefore, in terms of accuracy, this is our optimal model #1. However, if the number of parameters is prioritized, we should consider the other models listed below.

H5 file name for this model: model-00017-0.01374-0.99849-0.17060-0.93000.h5

**#2. Time distributed Conv2D + LSTM**

This model achieved a respectable training accuracy of 98.3% and a validation accuracy of 88.99%, while having the fewest parameters among the top three models we evaluated. Therefore, in terms of accuracy, this is our optimal model #2

H5 file name for this model: model-00025-0.07085-0.98341-0.37390-0.89000.h5

**#3. CNN + GRU**

This model achieved a solid training accuracy of 99% and a validation accuracy of 82.99%, while having fewer parameters compared to the top four models we evaluated. Therefore, in terms of accuracy, this is our optimal model #3

H5 file name for this model: model-00024-0.03856-0.99246-0.53267-0.85000.h5

# **Further Options which can explored apart from what we experimented:**

* Transfer Learning Implementation
  + Utilize pre-trained models such as ResNet50, ResNet152, or Inception V3.
  + Extract initial feature vectors for gesture identification.
  + Pass these features to an RNN for sequence processing and a softmax layer for classification.
* Consideration of GRU Model
  + Consider GRU as an alternative to LSTM due to its fewer trainable parameters and potentially faster computations.
  + Assess its impact on validation accuracies to validate its effectiveness.

* Enhanced Data Understanding
  + Explore diverse backgrounds, lighting conditions, individuals, and various recording cameras within the dataset.
  + A deeper exploration can enhance dataset diversity, aiding the generator function for improved model stability and accuracy.

* Hyperparameter Tuning, Experiment with different hyperparameter combinations:
  + Activation functions (ReLU, Leaky ReLU, tanh, sigmoid)
  + Filter size, padding, stride length, batch normalization, dropouts, etc.
  + Aim to refine and optimize model performance for better accuracy.