**Gesture Recognition Project**

This project is meant to recognise 5 hand gestures. We have tested different models including Conv3D . The main model that we experimented with different parameters was the Conv3D model. The model was trained using a Generator that selects batches of different gestures where each gesture consists of 30 frames.

**Generator:** The Generator was created to handle the full train and val data sets. The datasets were divided into batches of fixed size. The remaining datapoints after selecting the full batches were handled using an extra batch of less size. The generator also handled images to standardise the size and the selected frames….

**Images Preprocessing:** After analysing the images and frames, we found the following:

1. All Gestures are made of 30 frames (images).
2. All images are of 2 sizes (160x120x3) or (360x360x3). We resized all images to a common standard size of 120x120x3 after analysing the images .
   1. Images of 360x360 were resized to 120x120
   2. Images of 160x120x3 we cropped the images based on the gesture
   3. We Normalised all images dividing the pixels values by 255 as the images are RGB.
   4. we have intentionally reduced the image size for training 80\*80 for most part of training when we know that there would huge difference in train accuracy and validation accuracy but because of the reason that system keep crashing every time we considered 120\*120 or 160\*160 shape the the data would size would increase.

**Batches:**

We have tested the generator with different batch sizes from ranges 16 to 128 . The maximum batch size that didn’t crash the system was 128. Thus, we used this size batch\_size to maximise the utilisation of the GPU.

**Model:**

We have tested the model with different hidden layers, convolutions, and epochs. The best was achieved using 2 hidden Conv3D layers, each followed by a Maxpool3D layer, followed by flatten and 2 Dense layers. We added 2 dropouts of 0.15 and 0.1 and handle overfitting. This increased the val accuracy to over 80%.

For epochs > 15 we noticed that the model starts to suffer overfitting and seems to memorise the training images. Best accuracy was achieved with epochs 15..

Here is a summary of some of the etst.

|  |  |  |  |
| --- | --- | --- | --- |
| **Experiment Number** | **Model** | **Result** | **Decision + Explanation** |
| **1** | **Conv3D (image size =160\*160)** | **System crashes; Out of memory error** | **Reduced the image size and tested the image soze 80\*80 was the max that didn’t cause memory error, so we used batch\_size =64 to maximise GPU utilization and parallelization.** |
| **2** | **Conv3D: 2 Conv3D layers with 16, 32 filters and frame to select as 5 ,epochs 15** | **dint crash Trained well, but suffered overfitting; Very high train accuracy, but val accuracy was less** | **Add dropouts layers to handle overfitting** |
| **3** | **Conv3D: 2 Conv3D layers with 32 and 64 filters, epochs 5** | **decreased accuracy but validation was slightly increase** | **not better than previous model** |
| **4** | **Conv3D: 2 Conv3D layers with 32 and 64 ,128 filters, epochs 15** | **system crahsed** | **conv3D with dense layer 64 ,image 80\*80,frame to select 15 and epochs 15 ran fine for some tiem and crahsed** |
| **Final Model** | **Conv3D** | **Accuracy: 88%** |  |

*Note:out of the limited models we trained and resources we had model 2 was performing better in terms of accuracy ,when ever we try to experiment with other combinations by increasing the image , frames decreasing the batch size adding more layers ,adding dropouts system would crash. highest accuracy we got is around 94% but with over fitted model.*