**Summary on Gesture Recognition case study**

As part of data pre-preprocessing, started off with resizing and cropping the images in order to make sure that the Neural Network only recognizes the gestures appropriately and completely discards all the noise present in the image.

Then, normalization was performed for the RGB values of the image in order to get rid of distortions caused by lights and shadows present in the image.

Later, in order to improve the model’s accuracy, data augmentation was performed in order to make the image more generalizable.

NN Architecture development and training:

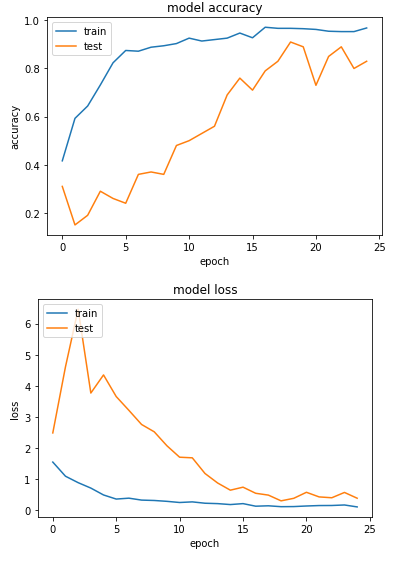
* Experimented with various model configurations and hyper-parameters with different combinations of batch sizes, image dimensions, filter sizes.
* Used ReduceLROnPlateau in order to decrease the learning rate if the metric(val\_loss) remains unchanged during epochs.
* Used BatchNormalization, Pooling and dropout layers when our model started to overfit.

Please find below the observations and a summary of the experiments that were performed as part of the model training.

* The experimentation kicked off with finding the batch size that would overshoot processing power. 60 was the number when it hit the OOM error.
* So did some experiments for Conv3D model to analyze the impact of different batch size, epochs, number of frames and image resolution from '**model\_conv3D\_1**' to '**model\_conv3D\_9**' in order to find a configuration that would use the GPU power appropriately.
* It was found that "image resolution" and "number of frames" in sequence have more impact on training time than batch\_size. So thought of using Batch Size around 15-50 for further major experiments and change the resolution 160 \* 160, 120 \* 120 according to the model performance.
* **Also, note that due to the expiry of the GPU power after few experiments and available time at hand, the GPU power got exhausted and hence the subsequent experiments had relatively slower training time and further experiments couldn’t be performed which can be taken for further experimentation and improvement of this case study.**
* Note that, no. of epochs=25 and batch size=15 was suitable and set for experiments stated below in tabular format.
* Note that dropouts = 0.25 in dense connections.
* Final models are highlighted in yellow along with their accuracy and loss numbers.

|  |  |  |  |
| --- | --- | --- | --- |
| **Experiment Number** | **Model** | **Result** | **Decision + Explanation** |
| **10**  **(4,522,437 parameters)** | **Conv 3D Model using 30 frames per video**  **+ 16, 32, 64, 128 filters conv 3D layers each followed by max pool 3D**  **+ 256 dense nodes**  **+ 128 dense nodes**  **+ image size 120 by 120** | **Training Accuracy: 99%**  **Validation Accuracy: 91%**  **Train loss: 0.1081**  **Val loss: 0.3875** | **Low validation accuracy as compared to training accuracy.** |
| **11**  **(6,881,733 parameters)** | **Conv 3D Model using 30 frames per video**  **+ 16, 32, 64, 128 filters conv 3D layers each followed by max pool 3D**  **+ 256 dense nodes**  **+ 128 dense nodes**  **+ image size 160 by 160** | **Training Accuracy: 93%**  **Validation Accuracy: 83%**  **Train loss: 0.2182**  **Val loss: 0.7458** | **Even lower accuracy compared to the previous experiment.** |
| **12**  **(4,522,437 parameters)** | **Conv 3D Model using 20 frames per video**  **+ 16, 32, 64, 128 filters conv 3D layers each followed by max pool 3D**  **+ 256 dense nodes**  **+ 128 dense nodes**  **+ image size 120 by 120** | **Training Accuracy: 97%**  **Validation Accuracy: 83%**  **Train loss: 0.2182**  **Val loss: 0.7458** | **Again low validation accuracy and hence overfitting** |
| **13**  **(3,693,253 parameters)** | **Mobilenet (retrain all weights)**  **+ GRU (128 cells)**  **+ Dense (128 nodes)**  **+ image size 120 by 120**  **+ 30 images per video** | **Training Accuracy: 99.7%**  **Validation Accuracy: 98%**  **Train loss: 0.2182**  **Val loss: 0.7458** | **Lot of improvement in accuracy and almost perfect model with bit of delta in loss.** |
| **14**  **(3,693,253 parameters)** | **Mobilenet (retrain all weights)**  **+ GRU (128 cells)**  **+ Dense (128 nodes)**  **+ image size 120 by 120**  **+ 30 images per video**  **+ random data transformations on the images** | **Training Accuracy: 100%**  **Validation Accuracy: 97%**  **Train loss: 0.0059**  **Val loss: 0.0538** | **High Training accuracy and considerable good validation accuracy.** |
| **15**  **(3,693,253 parameters)** | **Mobilenet (fine tune after 50th layer)**  **+ GRU (128 cells)**  **+ Dense (128 nodes)**  **+ image size 120 by 120**  **+ 30 images per video** | **Training Accuracy: 100%**  **Validation Accuracy: 93%**  **Train loss: 0.0059**  **Val loss: 0.2062** | **Lesser validation accuracy than previous experiment.** |
| **16**  **(3,693,253 parameters)** | **Mobilenet (retrain all weights)**  **+ LSTM (128 cells)**  **+ Dense (128 nodes)**  **+ image size 120 by 120**  **+ 30 images per video** | **Training Accuracy: 99.25%**  **Validation Accuracy: 99%**  **Train loss: 0.0317**  **Val loss: 0.0658** | **Perfect model.** |
|  |  |  |  |
| **Final Model(Conv 3D)**  **(4,522,437 parameters)** | **Conv 3D Model using 30 frames per video**  **+ 16, 32, 64, 128 filters conv 3D layers each followed by max pool 3D**  **+ 256 dense nodes**  **+ 128 dense nodes**  **+ image size 120 by 120** | **Training Accuracy: 99%**  **Validation Accuracy: 91%**  **Train loss: 0.1081**  **Val loss: 0.3875** | **Low loss and high accuracy with lesser amount of overfitting.** |
| **Final Model using transfer learning**  **(3,693,253 parameters)** | **Mobilenet (retrain all weights)**  **+ LSTM (128 cells)**  **+ Dense (128 nodes)**  **+ image size 120 by 120**  **+ 30 images per video** | **Training Accuracy: 99.25%**  **Validation Accuracy: 99%**  **Train loss: 0.0317**  **Val loss: 0.0658** | **Low loss and high accuracy in both training and validation datasets.** |

**Conv3D best model plot as below:**



**Mobinet + LSTM best model plot as below:**

