Gesture Recognition with Neural Network

- **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

Experiment Number	Model	Result	Decision + Explanation
1	**3D-CNN** • Kernal Size = (3,3,3) • Batch Size = 32 • Frames = 30 • Convolutions = 32, 64, 128 • Dense = 128, 64, 5 • Epochs = 5	**Train Accuracy:** 36.09% **Test Accuracy:** 43.75%	This is a starting model, but it is underfitting.
2	**3D-CNN** • Kernal Size = (3,3,3) • Batch Size = 32 • Frames = 30 • Convolutions = 16, 32, 64, 128 • Dense = 64, 64, 5 • With padding • Epochs = 5	**Train Accuracy:** 36.69% **Test Accuracy:** 37.50%	Add extra convolutional layer with 16 filters since above model is underfitting. But performance is still not much improved.
3	**3D-CNN** • Kernal Size = (2,2,2) • Batch Size = 32 • Frames = 30 • Convolutions = 32, 32, 64, 64 • Dense = 512 • Epochs = 5	**Train Accuracy:** 25.71% **Test Accuracy:** 7.00%	In this model I decreased the pool size as well as batch size. Also, similar type of convolutions is used here but the model is overfitting.

4	**3D-CNN** • Kernal Size = (2,2,2) • Batch Size = 32 • Frames = 30 • Convolutions = 32, 64, 128 • Dense = 128, 64, 5 • Batch Normalization • Epochs = 5	**Train Accuracy:** 41.92% **Test Accuracy:** 25.00%	Since the above models are not performing well therefore in this model, I decreased the kernel size and batch size. Batch normalization layer is also used. Now the model is overfitting.
5	**3D-CNN** • Kernal Size = (2,2,2) • Batch Size = 32 • Frames = 30 • Convolutions = 32, 64, 128 • Dense = 128, 64, 5 • Epochs = 15	**Train Accuracy:** 89.64% **Test Accuracy:** 87.50%	Remove batch normalization from the above model. From Now the model performance is increased significantly. But a small amount of overfitting is still there. Let's jump to CNN+RNN architecture.
6	**ResNet152 + GRU** • Batch Size = 10 • Time Distributed ResNet152 • Time Distributed Dense = 64 and 256 • GRU = 128 • Dense = 256, 5 • Epochs = 5	**Train Accuracy:** 66.51% **Test Accuracy:** 37.50%	This is a starter model with transfer learning. The model accuracy is good, but overfitting is there.
7	**ResNet50 + GRU** • Batch Size = 10 • Time Distributed ResNet50 • Time Distributed Dense = 64 and 256 • GRU = 128 • Dense = 256, 5 • Epochs = 5	**Train Accuracy:** 63.54% **Test Accuracy:** 37.50%	To remove overfitting, I use the less complex CNN model, now overfitting has removed but the model is under perform on train data.
8	**ResNet152 + GRU** • Batch Size = 10 • Time Distributed ResNet152 • Time Distributed Dense = 64 and 256 • GRU = 128 • Dense = 256, 128, 5 • Epochs = 5	**Train Accuracy:** 62.85% **Test Accuracy:** 31.25%	In experiment number 5, I have added an extra dense layer of 128 neurons. No effect on model's test accuracy.

9	**MobileNet + GRU** • Batch Size = 64 • Time Distributed MobileNet • Time Distributed Batch Normalization • GRU = 256 • Dense = 256, 5 • Epochs = 15	**Train Accuracy:** 92.12% **Test Accuracy:** 87.50%	Try another transfer learning model with large batch size, this time model is performing extremely well.

Contributor:

Vishrut Mishra