Case Study Submission by Saptarshi Ghosh and Nitin Balaji Srinivasan, Cohort 58 - AI and ML

Gesture Recognition

Problem Statement & Objective

In this group project, we 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.

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

Objective

We have to build 3D Convolution models (Conv3D) and the standard (CNN + RNN based) models that will be able to predict the 5 gestures correctly.

Approach

For analysing videos using neural networks, two types of architectures are used commonly.

- Standard CNN + RNN architecture in which you pass the images of a video through a CNN which extracts a feature vector for each image, and then pass the sequence of these feature vectors through an RNN.
- CNNs - a 3D convolutional network.

In this project, we will develop models in both these architectures, that will be able to predict the 5 gestures correctly.

Input data understanding

The data consists of 663 training video sequences (each video sequence in a folder made of 30 frames/images per sequence) and 100 validation testing video sequences.

These video sequences capture the five gestures that are required to be classified through this case study.

Generators

As part of our approach to building the models, we have to use a “generator” to yield a batch of data to the models. This is the most important part of the code.

For this project, we have developed our own batch data generator to perform the following.

1. Parameters - the source path for training and validation, folder list, batch size (to change during the experimentation process), the number of images to sample and a parameter to switch on/off augmentation.

2. Read a set of images from the source folder, depending on the batch size and the number of images to sample in each video sequence

3. Preprocess the images to do the following;

    a. With data augmentation mode turned on (for training only);
        i. Double the folder size to allow for some original images to pass through without any augmentation
        ii. Randomly perform one of the following types of augmentation (Adjusting brightness/contrast, affine transformation, converting to grayscale, allow some images to pass through without augmentation)
        iii. Cropping the images to focus on the key gestures to avoid noise

    b. For all images (with or without data augmentation):
        i. Resizing to the preferred height and width and;
        ii. Normalisation using min/max method

4. Depending on the batch size, read the remaining set of images and apply the same preprocessing depending on with or without augmentation

Please note that in the latest version of the Keras models that we have implemented, the model.fit() function fits the generator as well.

Model building and experimentation

We have built 3D Convolution models (Conv3D) and the standard (CNN + RNN based) models to predict the 5 gestures.

In experimenting the models, we have tried various model definitions, hyper-parameters and combinations of batch sizes and filter sizes.

We have used ModelCheckpoint to track all the improved versions of each model (monitoring validation accuracy “val_categorical_accuracy”). We have also used “ReduceLRonplateau” to reduce the learning rate when the validation accuracy reaches a plateau for 5 consecutive epochs. If the validation accuracy does not improve after 7 consecutive epochs, then early stopping kicks in.

We have also utilised ‘Transfer Learning’ from “Mobilenet” plus LSTM.

The following table provides the details of models built as part of the experimentation process, the relevant input and output metrics for each of the models and our rationale for the best model chosen.

Model Type	Model number & Description (ref. notebook)	Model key parameters	Training Accuracy	Validation Accuracy (Highest achieved)	Decision + Explanation - Comments
Conv3D	**Model 1 - Two Conv3D layers and no Dropouts with Augment=True**	Images to sample - 15 Batch size - 33 Trainable parameters - 44,244,405 Includes: 2 Pooling layers Dense layer (512) Augmentation - True Epochs run - 20	74.3% (highest achieved is 86.8)_	71.9%	There is significant overfitting, as evidenced by the training accuracy continuing to improve while the validation accuracy fluctuates and does not improve consistently. Hence decided to experiment with dropouts in the next model.
Conv3D	**Model 2: Experimenting with dropout,L2 Regularizer and decreased Number of neurons in the Dense layer: (augment = True)**	Images to sample - 15 Batch size - 33 Trainable parameters - 22,124,565 Includes: 2 Pooling layers, 3 Drop out layer (0.4) & Dense layer (512) Augmentation - Yes Epochs run - 13	30.59% (highest achieved training accuracy is 37.66)	22.7%	Both Training and validation accuracy is not good and this is due to the fact that it isn't able to learn the features good enough. So now we are proceeding with one additinal layer and see if things change.
Conv3D	**Model 3 - Three Conv3D layers with Dropout and augment=True:**	Images to sample - Batch size - 33 Trainable parameters - 7,396,853 Includes: 2Pooling layers, 1 Drop out layer (0.5) & Dense layer (64) Augmentation - No Epochs run - 16	75.5% (highest achieved training accuracy is also 75.5%)	51.5%	Given that the addition of dropout at a rate of 0.2 has not stabilized the validation curves, it suggests that further adjustments are necessary.
Conv3D	**Model 4 - Conv3D with 3 layers, dropout and no batch normalisation and Augment = False:**	Images to sample - 15 Batch size - 33 Trainable parameters - 7,396,853 Includes: 3 Pooling layers, 3 Drop out layer (0.3) and one Dropout(0.2) & Dense layer (512) Augmentation - No Epochs run - 17	64.3 %(highest achieved training accuracy is 64.3 though)	67.4%	Although there still is overfitting but the gap between the training curve and accuracy curve is highly minimized and Also the training accuracy increases as compared to the baseline 2 layered models without dropouts. Proceeding with Data Augmentation further to enhance the 3 layered model performance.
Conv3D	**Model 5- Conv3D with 3 layers, no Dropout and no batch normalisation and Augment = True:**	Images to sample - 15 Batch size - 33 Trainable parameters - 7,396,853 Includes: 3 Pooling layers Dense layer (512) Augmentation - Yes Epochs run - 13	63.7 %(highest training accuracy is however93.36)	74.2%	Data Augmentation seems to have definitely helped to get better results but there is still overfitting. Hence, using Dropouts finally to deal with overfitting.
Conv3D	**Model 6 - Conv3D with 3 layers, dropout and no batch normalisation and Augment = True:**	Images to sample - 15 Batch size - 33 Trainable parameters - 7,396,853 Includes: 3 Pooling layers, 3 Drop out layer (0.3) and one Dropout(0.2) & Dense layer (512) Augmentation - Yes Epochs run - 13	79.65 %(highest training accuracy is 81.67%)	78.8%	The model performance is significantly better than the rest of the Conv3D models and the validation loss curve is also very similar to the training loss curve. The same can be seen with the validation accuracy as wellwhich is by far the highest. Lets increase the "number of images to sample" by 1 and try CNN + RNN based models

Model Type	Model number & Description (ref. notebook)	Model key parameters	Training Accuracy	Validation Accuracy (Highest achieved)	Decision + Explanation - Comments
CNN + LSTM	**Model 7 - 4 Conv2D layers + LSTM with 64 lstm cells**	Images to sample - 16 Batch size - 33 Trainable parameters - 937,765 Includes: 4 Pooling layers, 1 Drop out layer (0.5) & Dense layer (64) Augmentation - No Epochs run - 16	65% (Highest achieved is 89.6%)	50.0%	There is a significant overfitting issue and the validation accuracy is not high. Lets try with augmentation and with ConvLSTM model
CNN + ConvLSTM	**Model 8 - 3 Conv2D layers + 1 ConvLSTM2D layer**	Images to sample - 16 Batch size - 33 Trainable parameters - 75,301 Includes: 4 Pooling layers, 2 Drop out layer (0.4) & 2 Dense layer (64) Augmentation - No Epochs run - 14	28% (Highest achieved is 49.5%)	54.6%	The ConvLSTM model has low number of parameters but the accuracy is really low. We need to try this with augmentation
CNN + ConvLSTM	**Model 8 - 3 Conv2D layers + 1 ConvLSTM2D layer**	Images to sample - 16 Batch size - 33 Trainable parameters - 75,301 Includes: 4 Pooling layers, 2 Drop out layer (0.4) & 2 Dense layer (64) Augmentation - Yes Epochs run - 16	42.8% (Highest achieved is 49.8%)	36.4%	The accuracy has still not improved but it would be worth trying GRU if it helps improve the accuracy
CNN + GRU	**Model 9 - 4 Conv2D layers + GRU with 64 GRU cells**	Images to sample - 16 Batch size - 33 Trainable parameters - 728,997 Includes: 4 Pooling layers, 1 Drop out layer (0.5) & Dense layer (64) Augmentation - No Epochs run - 24	84.6% (Highest achieved is 96.5%)	51.5%	Overfitting issue, so we need to try GRU with data augmentation

Model Type	Model number & Description (ref. notebook)	Model key parameters	Training Accuracy	Validation Accuracy (Highest achieved)	Decision + Explanation - Comments
CNN + LSTM	**Model 7 - 4 Conv2D layers + LSTM with 64 lstm cells**	Images to sample - 16 Batch size - 33 Trainable parameters - 937,765 Includes: 4 Pooling layers, 1 Drop out layer (0.5) & Dense layer (64) Augmentation - No Epochs run - 16	65% (Highest achieved is 89.6%)	50.0%	There is a significant overfitting issue and the validation accuracy is not high. Lets try with augmentation and with ConvLSTM model
CNN (Mobilenet Transfer learning) + LSTM	**Model 10 - Transfer learning + LSTM with 64 lstm cells**	Images to sample - 16 Batch size - 33 Trainable parameters - 285,317 Includes: 1 Batch Normalisation, 1 Pooling layer, 1 Drop out layer (0.25) & Dense layer (64) Augmentation - No Epochs run - 21	95.9% (Highest achieved is 96.1%)	59.1%	Increase drop out and run this with image augmentation to see if it helps to reduce "overfitting" improves the validation accuracy
CNN + LSTM	**Model 11 - 4 Conv2D layers + LSTM with 64 lstm cells with Augment=True**	Images to sample - 16 Batch size - 33 Trainable parameters - 937,765 Includes: 4 Pooling layers, 1 Drop out layer (0.5) & Dense layer (64) Augmentation - Yes Epochs run - 16	79.4% (Highest achieved is 91.7%)	51.5%	The data augmentation has only marginally improved Model 7. Lets try augmentation on the Transfer Learning model
CNN (Mobilenet Transfer learning) + LSTM	**Model 12 - Transfer learning + LSTM with 64 lstm cells with Augment=True**	Images to sample - 16 Batch size - 33 Trainable parameters - 283,269 Includes: 1 Pooling layer, 1 Drop out layer (0.4) & Dense layer (64) Augmentation - Yes Epochs run - 31	96.7% (Highest achieved is 96.7%)	87.1%	**Great result. We will choose this as the best model as this achieves a good level of accuracy with reduced overfitting and with relatively low number of trainable parameters**
CNN + GRU	**Model 13 - 4 Conv2D layers + GRU with 64 GRU cells with augment = True**	Images to sample - 16 Batch size - 33 Trainable parameters - 728,997 Includes: 4 Pooling layers, 1 Drop out layer (0.5) & Dense layer (64) Augmentation - Yes Epochs run - 11	45.7% (Highest achieved is 71.4%)	62.9%	The validation accuracy has improved following data augmentation but it is not high enough, though there is no overfitting!

Key Observations

- Initial experimentation of multiple batch sizes showed that with a batch size of 33, there was a reasonable trade-off between training time and overall accuracy. Therefore we have subsequently executed all the models with the batch size of 33.
- As the number of trainable parameters increase, the model takes longer to train.
- Increasing dropouts and using data augmentation were instrumental in reducing overfitting.
- CNN + GRU model had better computational performance than CNN + LSTM based on the trainable parameters, but both the models did not achieve a high level of accuracy based on the experiments that we have taken.
- Transfer Learning model produced a good accuracy level (training accuracy of 97%, validation accuracy of 87%) and the number of trainable parameters was comparatively lower than the other models, as expected.

Best Model:

We will choose "Model 12 - Transfer learning + LSTM with 64 LSTM cells with Augment=True" as the best model as this achieves a good level of accuracy with reduced overfitting and with relatively low number of trainable parameters