2022

Gesture Recognition

CASE STUDY

SUDEEP K S

Contents

Problem Statement

Develop a cool feature in the smart-TV that can **recognize five different gestures** performed by the user which will help users control the TV without using a remote.

Understanding Dataset

The data is in a zip file. The zip file contains a 'train' and a 'val' folder with two CSV files for the two folders. These folders are in turn divided into subfolders where each subfolder represents a video of a particular gesture. Each subfolder, i.e., a video, contains 30 frames (or images). Note that all images in a particular video subfolder have the same dimensions but different videos may have different dimensions. Specifically, videos have two types of dimensions - either 360x360 or 120x160 (depending on the webcam used to record the videos). Hence, you will need to do some pre-processing to standardize the videos.

Each row of the CSV file represents one video and contains three main pieces of information - the name of the subfolder containing the 30 images of the video, the name of the gesture and the numeric label (between 0-4) of the video.

Objective

In this project, I will be building a model to recognize 5 hand gestures

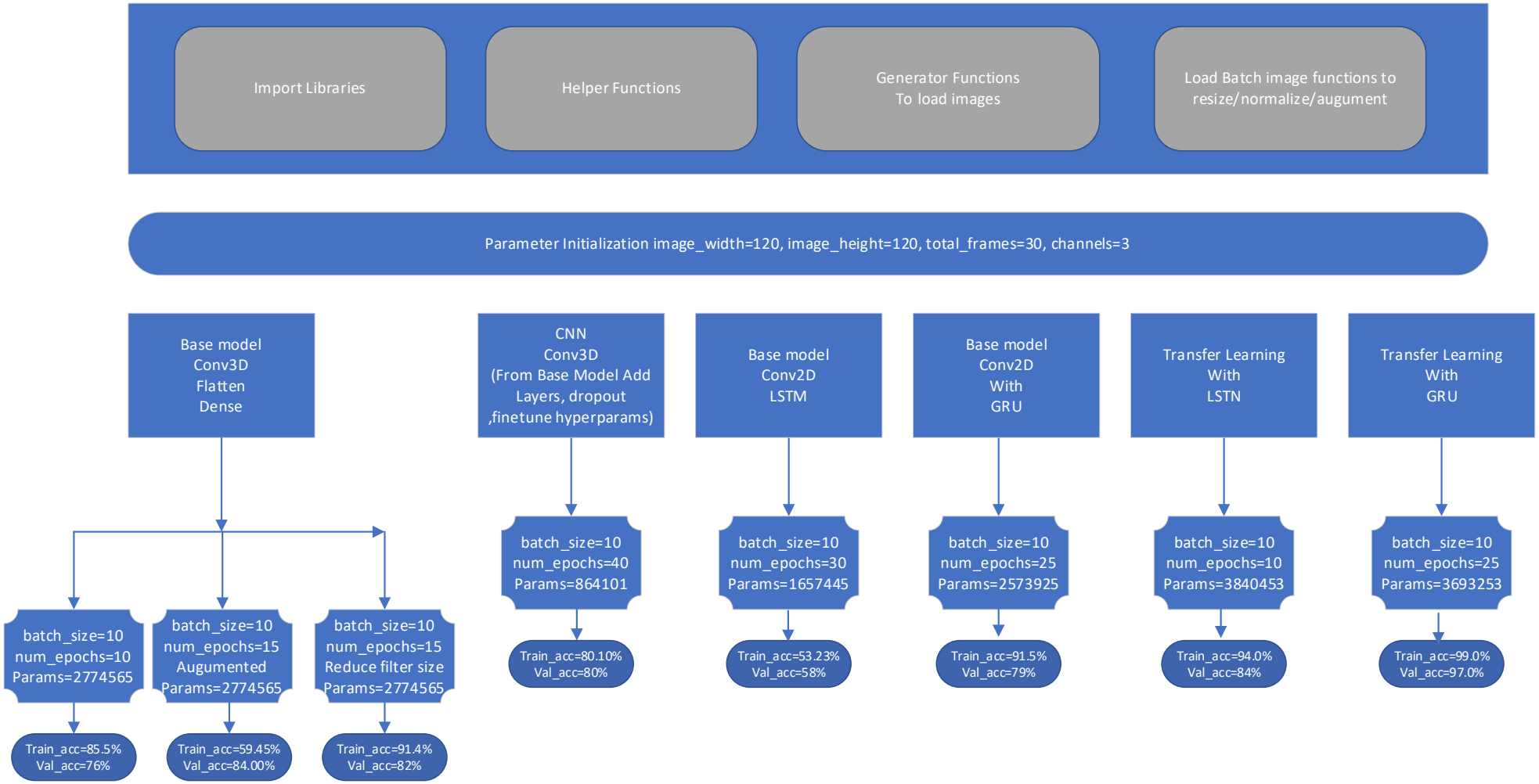
Each gesture corresponds to a specific command:

1. Thumbs up: Increase the volume
2. Thumbs down: Decrease the volume
3. Left swipe: 'Jump' backwards 10 seconds
4. Right swipe: 'Jump' forward 10 seconds
5. Stop: Pause the movie

Each video is a sequence of 30 frames (or images).

Approach

As a part of this case study, I have tried multiple to visualize it I am showing it in a code flow diagram below:



Experiment Number	Model	Result	Decision + Explanation
1	**Base Model 1** **Conv 3D** **Flatten Dense**	batch_size=10 num_epochs=10 Params=2774565 **Train_acc=85.5%** **Val_acc=76%**	**Big difference between Training and validation accuracy.** **Looks like overfitting.**
2	**Base Model 2** **Conv 3D** **Flatten Dense** **Augmented**	batch_size=10 num_epochs=15 Params=2774565 **Train_acc=59.45%** **Val_acc=84.00%**	**By introducing Augmentation Accuracy is declined and not improving**
3	**Base Model 3** **Conv 3D** **Flatten Dense** **Reduce filter size**	batch_size=10 num_epochs=15 Params=2774565 **Train_acc=91.4%** **Val_acc=82%**	**Reducing filter size improved accuracy drastically** **Validation accuracy also improved indicating the overfitting issue has improved**
4	**CNN- Conv3D Added layers** **Dropout Fine-tuned Hyper Params**	batch_size=10 num_epochs=40 Params=864101 **Train_acc=80.10%** **Val_acc=80%**	**Adding Layers in Base models, dropout and hyper parameter tuning this produces a model with decent accuracy, Since Validation accuracy is almost same as training accuracy overfitting issue is resolved** **This model has the least parameters**
5	**Base model Conv2D LSTM**	batch_size=10 num_epochs=30 Params=1657445 **Train_acc=53.23%** **Val_acc=58%**	**Accuracy dropped drastically, no point in pursuing further with this model**
6	**Base model Conv2D With GRU**	batch_size=10 num_epochs=25 Params=2573925 **Train_acc=91.5%** **Val_acc=79%**	**Accuracy increased but there is a significant difference between training and validation accuracy. This points to issue of overfitting.**
7	**Transfer Learning** **With** **LSTN**	batch_size=10 num_epochs=10 Params=3840453 **Train_acc=94.0%** **Val_acc=84%**	**Accuracy has improved a lot but the number of parameters used is huge. The difference between training and validation accuracy is 10% pointing to a possibility of over fitting.**
8	**Transfer Learning With GRU**	batch_size=10 num_epochs=25 Params=3693253 **Train_acc=99.0%** **Val_acc=97.0%**	**Accuracy is near perfect with the difference in training and validation accuracy only 2% which means overfitting issue is resolved.** **But we have used significantly more parameters**

Experiments Done

- A total of 8 models were used for analysis of this case study
- Parameter initialization was done as shown in the figure above
- Augmentation was carried out as a part of calling load image batch function
- Adam optimizer was used in all the above models
- I used SoftMax in the last dense function, as this is a multi-target classification

Conclusion

From the above table I can conclude that **Experiment number 4(CNN Conv3d)** and **experiment number 8(Transfer learning with GRU)** have the best results, we can see that there is no overfitting. **Since CNN CONV3D has the lower params I conclude that this is the best model.**

The loss is decreasing, the accuracy is increasing as shown in the graph below: