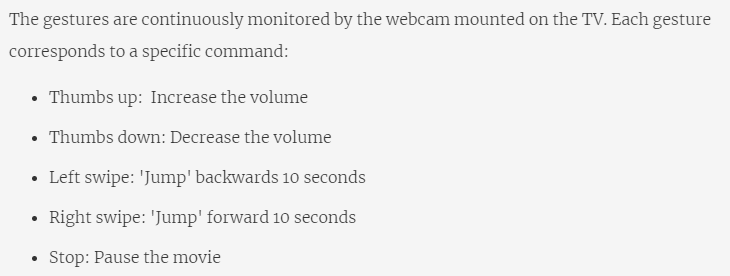
Deep Learning Course Project - Gesture Recognition

Problem Statement:

As a data scientist at a home electronics company which manufactures state of the art smart televisions. 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.



Understanding the Dataset:

The training data consists of a few hundred videos categorized into one of the five classes. Each video (typically 2-3 seconds long) is divided into a sequence of 30 frames (images). These videos have been recorded by various people performing one of the five gestures in front of a webcam - like what the smart TV will use.

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 standardise 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.

Our task is to train a model on the 'train' folder which performs well on the 'val' folder as well (as usually done in ML projects). We have withheld the test folder for evaluation purposes - your final model's performance will be tested on the 'test' set.

for downloading the dataset [Click here](https://drive.google.com/uc?id=1ehyrYBQ5rbQQe6yL4XbLWe3FMvuVUGiL).

**Solution**

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

Convolutions + RNN:

The conv2D network will extract a feature vector for each image, and a sequence of these feature vectors is then fed to an RNN-based network. The output of the RNN is a regular softmax (for a classification problem such as this one).

3D Convolutional Network or Conv3D:

3D convolutions are a natural extension to the 2D convolutions you are already familiar with. Just like in 2D conv, you move the filter in two directions (x and y), in 3D conv, you move the filter in three directions (x, y and z). In this case, the input to a 3D conv is a video (which is a sequence of 30 RGB images). If we assume that the shape of each image is 100x100x3, for example, the video becomes a 4-D tensor of shape 100x100x3x30 which can be written as (100x100x30)x3 where 3 is the number of channels. Hence, deriving the analogy from 2-D convolutions where a 2-D kernel/filter (a square filter) is represented as (fxf)xc where f is filter size and c is the number of channels, a 3-D kernel/filter (a 'cubic' filter) is represented as (fxfxf)xc (here c = 3 since the input images have three channels). This cubic filter will now '3D-convolve' on each of the three channels of the (100x100x30) tensor.

Note:

* We can use transfer learning in the 2D CNN layer rather than training our own CNN
* GRU can be a better choice than an LSTM since it has lesser number of gates (and thus parameters)

Generators:

In most deep learning projects, you need to feed data to the model in batches. This is done using the concept of generators.

Creating data generators is probably the most important part of building a training pipeline. Although libraries such as Keras provide builtin generator functionalities, they are often restricted in scope and we must write your own generators from scratch.

In our generator function we pre-processed the data such as resizing and normalizing. Resizing the data is important because we have data of two sizes, 360\*360 and 120\*160 so we get all images to same size and normalization to remove distortions in the images.

Model Building:

We have built our models as a function, so that we can experiment with the models just by giving parameters to the functions. Without changing model architecture every time.

The parameters given to the model are hyperparameters such as Batch size, Image shape, number of epochs and number of frames.

We also made a function to visualise the history of the model such as Train and validation loss, Train and validation accuracy.

Observations:

* It was observed that as the Number of trainable parameters increase, the model takes much more time for training.
* Batch size has impact on accuracy and training time as we increase the batch size the training time decreases, and accuracy decreases, so we should choose correct batch size to balance the training time and accuracy.
* Use of *ReduceLROnPlateau* decrease the learning rate if the monitored metrics (*val\_loss*) remains unchanged in between epochs. Helped to build better model.
* Cnn model in combination with rnn works better than a 3d conv model for our application may be due to the parameters we used and can be improved with the correct hyperparameters.
* Use of transfer learning boosted the accuracy, we can use different type of architectures based on requirements such as performance, size of the model e.t.c.,
* For observations of each model refer the table below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Experiment Number** | **Model** | **Parameters**  **used** | **Result** | **Decision + Explanation** |
| **1** | **Conv3D** | **Batch size: 50**  **Epochs: 10**  **Img shape: (120,120)** | **Train\_acc = 46**  **Val\_acc = 47** | **Model is under fitting so increasing the number of epochs and reducing the batch size** |
| **2** | **Conv3D** | **Batch size: 30**  **Epochs: 15**  **Img shape: (120,120)** | **Train\_acc = 85**  **Val\_acc =50** | **Model is overfitting over the data so tried to add drop out layers** |
| **3** | **Conv3D** | **Batch size: 30**  **Epochs: 15**  **Img shape: (120,120)** | **Train\_acc = 56**  **Val\_acc =75** | **Performance of model improved but the problem of overfitting still exists.** |
| **4** | **Conv3D** | **Batch size: 20**  **Epochs: 15**  **Img shape: (120,120)** | **Train\_acc = 74**  **Val\_acc =63** | **Model performance improved again but a little over fitting exists** |
| **5** | **Conv3D** | **Batch size: 20**  **Epochs: 15**  **Img shape: (160,160)** | **Train\_acc = 61**  **Val\_acc =65** | **Over fitting exist and we need to improve accuracy try to reduce image size and see** |
| **6** | **Conv3D** | **Batch size: 20**  **Epochs: 15**  **Img shape: (100,100)** | **Train\_acc = 87**  **Val\_acc =79** | **Accuracy increased but validation accuracy not improved adding dropouts to reduce overfitting** |
| **7** | **Conv3D** | **Batch size: 20**  **Epochs: 15**  **Img shape: (100,100)** | **Train\_acc = 66**  **Val\_acc =70** | **Model performed better without overfitting but accuracy need to be improved trying lstm based model** |
| **8** | **ConvLSTM** | **Batch size: 20**  **Epochs: 15**  **Img shape: (100,100)** | **Train\_acc = 72**  **Val\_acc =63** | **Overfitting is reduced to improve accuracy we are adding more layers** |
| **9** | **ConvLSTM** | **Batch size: 20**  **Epochs: 20**  **Img shape: (100,100)** | **Train\_acc = 75**  **Val\_acc =71** | **Accuracy is increased but not satisfactory so improving batch size** |
| **10** | **ConvLSTM** | **Batch size: 30**  **Epochs: 20**  **Img shape: (100,100)** | **Train\_acc = 73**  **Val\_acc =75** | **Model is over fitting so adding dropout layers** |
| **11** | **ConvLSTM** | **Batch size: 30**  **Epochs: 20**  **Img shape: (100,100)** | **Train\_acc = 79**  **Val\_acc =57** | **Model is overfitting trying to use GRU based model for better performance** |
| **12** | **ConvGRU** | **Batch size: 20**  **Epochs: 20**  **Img shape: (100,100)** | **Train\_acc = 83**  **Val\_acc =71** | **Model is still overfitting we will try to use transfer learning to improve performance** |
| **13** | **ConvLSTM**  **Transfer learning** | **Batch size: 5**  **Epochs: 20**  **Img shape: (120,120)** | **Train\_acc = 89**  **Val\_acc =88** | **A desired model with good accuracy is obtained with transfer learning** |
| **Final Model** | **Conv3D with Dropouts and**  **normalization** | **Batch size: 20**  **Epochs: 15**  **Img shape: (100,100)** | **Train\_acc = 93**  **Val\_acc =88** | **A desired model with good accuracy is obtained with transfer learning** |