**Neural Networks Project-Gesture recognition:**

Submitter

* Pankaj Dixit

**Problem Statement:**

Problem statement for the **Neural Networks Project-Gesture recognition** activity is as below:

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 recognize 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

**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 - similar to what the smart TV will use. Videos have two types of dimensions - either 360x360 or 120x160. 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.

**Activity Flow:**

Train different models on the 'train' folder to identify the correct command in each sequence of frames and which also performs well on the 'val' folder. Final model's performance will be tested on the 'test' set which has been withheld for evaluation purpose.

**Two Architectures for Video data analysis: 3D Convs and CNN-RNN Stack:**

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

1. **CNN + 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).

1. **3D Convolutional Network, or Conv3D**

3D convolutions are a natural extension to the 2D convolutions. 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.

**Models covered as part of the activity:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Exp No** | **Model** | **Result** | **Decision + Explanation** |
| **1** | **Conv3D**   * Batch Size : 32 * Image Size:160 X 160 * Frames: 30 * Epochs: 15 | This model overfits as the training accuracy is quite high while validation accuracy is too low  **Training accuracy:** 95.78%  **Validation accuracy:** 20.00%  **Total params:** 1,117,061 | Reduce the hyperparameters |
| **2** | **Conv3D**   * Batch Size : 16 * Image Size : 160 X 160 * Frames: 30 * Epochs: 15 | This model also overfits as the training accuracy is still quite high while validation accuracy is low  **Training accuracy**: 94.57%  **Validation accuracy**: 77%  **Total params**: 1,117,061 | Reduce the batch size |
| **3** | **Conv3D**   * Batch Size : 16 * Image Size : 120 X 120 * Frames: 30 * Epochs: 15 | This model overfits and the train & test accuracy decrease a bit as well.  **Training accuracy**: 90.20%  **Validation accuracy**: 70.0%  **Total params**: 699,269 | Reduce the image size |
| **4** | **Conv3D**   * Batch Size : 16 * Image Size : 120 X 120 * Frames: 20 * Epochs: 15 | This model overfits as well but there is a slight improvement in accuracy.  **Training accuracy**: 93.36%  **Validation accuracy**: 78.00%  **Total params**: 699,269 | Increase number of frames. |
| **5** | **Conv3D**   * Batch Size : 16 * Image Size : 120 X 120 * Frames: 20 * Epochs: 25 | This model overfits as well and there is a no significant improvement in accuracy. There is no effect of increasing Epochs.  **Training accuracy**: 93.21%  **Validation accuracy**: 77.00%  **Total params**: 699,269 | Increase the number of epochs.  Try some other metrics/strategy eg: DataAugmentation |
| **6** | **Conv3D**   * Batch Size: 16 * Image Size: 120 X 120 * Frames: 20 * Epochs: 15 * Data Augmentation : Enabled | This model overfits as well and there is a significant decrease in Validation accuracy.  **Training accuracy**: 92.08%  **Validation accuracy**: 35.00%  **Total params**: 699,269 | Enabled data Augmentation |
| **7** | **Conv3D**   * Batch Size : 16 * Image Size : 100 X 100 * Frames : 16 * Epochs : 15 * Filter size : 2X2X2 * Data augmentation : True | This model overfits as well and there is a no significant change accuracy.  **Training accuracy**: 85.97%  **Validation accuracy**: 39.00%  **Total params**: 387,573 | Image Size and Frames reduced. Filter size also reduced to 2x2x2 with data Augmentation enabled. |
| **8** | **Conv3D**   * Batch Size: 16 * Image Size: 120 X 120 * Frames: 16 * Epochs: 15 * Filter size: 2X2X2 * dense\_neurons: 128 * Data Augmentation: Enabled | This model overfits as well and there is a significant increase in validation accuracy.  **Training accuracy**: 93.59%  **Validation accuracy**: 77.00%  **Total params**: 1,113,925 | Increased dense\_neurons to 128 |
| **9** | **Conv3D**   * Batch Size : 16 * Image Size : 120 X 120 * Frames : 16 * Epochs : 15 * Filter size: 2X2X2 * Dense\_neurons: 256 * Dropout: 0.5 * Data Augmentation : True | For this model the train and val accuracy gap narrows down so it is better model than previous ones.  **Training accuracy**: 86.27%  **Validation accuracy**: 77.00%  **Total params**: 1,762,613 | increased dense neurons to 256 and dropout to 0.5 |
| **10** | **Conv3D Model complexity increase with more layers added** | This model overfits and validation accuracy decreases significantly on increasing model complexity.  **Training accuracy**: 86.05%  **Validation accuracy:** 25.00%  **Total params:** 1,937,893 | Increasing number of layers in the model.  Model complexity makes the model overfit so complexity reduction is needed. |
| **11** | **Conv2D+ LSTM** | CNN + LSTM  model overfits and validation accuracy is too low.  **Training accuracy:** 97.59%  **Validation accuracy:** 26.00%  **Total params:** 3,392,869 | Conv2D+ LSTM used |
| **12** | **Conv2D+ GRU** | CNN + LSTM  model also overfits and validation accuracy is still too low.  **Training accuracy:** 97.89%  **Validation accuracy:** 41.00%  **Total params:** 2,573,925 | Conv2D+ GRU used |
| **13** | **Transfer Learning : mobilenet with trainable layer disabled** | Mobilenet with trainable layer disabled also overfits. But there is a significant improvement in accuracy.  **Training accuracy:** 99.32%  **Validation accuracy:** 78.00%  **Total params:** 3,693,253 | Transfer Learning using mobilenet ( trainable = False) |
| **14** | **Transfer Learning : mobilenet with trainable layer enabled** | Mobilenet with trainable layer enabled performs well both on train and val accuracy and the numbers are quite high.  **Training accuracy:** 99.70%  **Validation accuracy:** 95.00%  **Total params:** 3,693,253 | Transfer Learning using mobilenet ( trainable = True) |

**Observations:**

* + Mobilenet with trainable layer enabled (Model#14) performs well on both the train and val data. Also, the accuracy is quite high. So, it is the best model out of all the models covered under the above experiment. Its metrics are as below:
* Training accuracy: 99.70%
* Validation accuracy: 95.00%
* Total params: 3,693,253
* Transfer learning models perform better than conv3d and CNN+GRU or CNN+LSTM models.
* Although Model#14 gives us best accuracy numbers but the model params are on the higher side (3,693,253) as compared to conv3d and CNN+GRU or CNN+LSTM models.
* Increasing the model complexity further degrades the model performance as it increases overfitting as is evident in Model#10
* Change in model performance was not significant while enabling Data augmentation