Deep Learning Course Project- Gesture Recognition

* Jishna M – Group Facilitator
* Kiran

1. Problem Statement

* 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

1. 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)**.
* 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.

1. Objective

* Train a model on the 'train' folder which performs well on the 'val' folder as well (as usually done in ML projects)
* There are two types of architecture commonly used for analyzing videos:
  + **Convolutions + RNN**
    - **Use transfer learning in the 2D CNN layer rather than training your own CNN**
    - **GRU can be a better choice than an LSTM since it has lesser number of parameters**
  + **3D Convolutional Network, or Conv3D**

1. Data Generators

* Write own generators from scratch to feed batches of videos, not images.
* In the generator, we are going to pre-process the images as we have images of 2 different dimensions (360 x 360 and 120 x 160) as well as create a batch of video frames.
* The generator should be able to take a batch of videos as input without any error. Steps like cropping, resizing and normalization should be performed successfully.

1. Data Pre-processing

* **Resizing and cropping of the images**. This was mainly done to ensure that the NN only recognizes the gestures effectively rather than focusing on the other background noise present in the image.
* **Normalization of the images**. Normalizing the RGB values of an image can at times be a simple and effective way to get rid of distortions caused by lights and shadows in an image.
* **Augmentation.** At the later stages for improving the model’s accuracy, we did data augmentation to rotate the pre-processed images of the gestures to bring in more accurate data for the model to train on.

1. Models

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Exp No. | Model | Result | Decision + Explanation | Parameters |
| 1 | Conv3D | CPU Session Crashed | Reduce the batch size and Reduce the number of neurons in Dense layer |  |
| 2 | Conv3D  (Model 1) | Training Accuracy 0.88  Validation Accuracy 0.27 | * Val\_loss didn’t improve from 1.5671, so early stopping the training process. * Let’s add some Dropout Layers to Model 2. | 1,116,325 |
| 3 | Conv3D  (Model 2) | Training Accuracy: 0.79  Validation Accuracy: 0.31 | * Increase the amount of trainable data/ reduce the filter size. * Learning Rate reduces to 0.0002 * Val\_loss didn’t improve from 1.8505, so early stopping. | 3,638,981 |
| 4 | Conv3D  (Model 3 & 4) | Training Accuracy: 0.83  Validation Accuracy: 0.25 | * Reduced the amount of trainable data. * Add some Dropout Layers. * Overfitting is considerably high, not much improvement. * No improvement in Val Loss, lets switch to TimeDistributed Conv2D + GRU. | 1,336,629 |
| 5 | TimeDistributed Conv2D + GRU  (Model 5) | Training Accuracy: 0.76  Validation Accuracy: 0.25 | * Overfitting is considerably high, not much improvement. * Adding Dropout layers to feed in next model (Model 6) | 482,693 |
| 6 | TimeDistributed Conv2D + GRU with dropout  (Model 6) | Training Accuracy: 0.20  Validation Accuracy: 0.18 | * No Loss in validation and training dataset. * But there is not much improvement in accuracy too. | 482,693 |
| 7 | CNN+LSTM  (Model 7) | Training Accuracy: 0.27  Validation Accuracy: 0.32 | * Overfitting is not present but accuracy is still not good. | 6,817,829 |
| 8 | CNN+LSTM  (Model 8) | Training Accuracy: 0.81  Validation Accuracy: 0.62 | * Accuracy is improving with each epoch * Also, Loss is decreasing for both training and Validation data. | 625,029 |
| 9 | CNN LSTM + GRU  (Model 9) | Training Accuracy: 0.39  Validation Accuracy: 0.23 | * Validation Loss is increasing. * Accuracy is not as per expectation. * CNN+LSTM is still the best model till now. | 1,00,0293 |
| 10 | Transfer Learning with GRU  (Model 10) | Training Accuracy:0.99  Validation Accuracy:0.94 | Seems better result. This will be final model. | 3,69,3253 |
| Final Model | CNN+LSTM  (Model 8) | Training Accuracy: 0.81  Validation Accuracy: 0.62 | Loss is decreasing whereas Accuracy is increasing for both training and Validation data. | 625,029 |

1. Observations

* CNN+LSTM based model with GRU cells had better performance than Conv3D.
* Model training time increase with the increase in
  + Number of trainable parameters increase.
  + Decreasing the batch size.
* A batch size greater than 50 can throw GPU Out of memory error, and thus we played with the batch size between 30-40.

**Batch size ∝ GPU memory / available compute.**

* Increasing the batch size greatly reduces the training time but this also has a negative impact on the model accuracy. To make model more accurate choose lower batch size.
* Data Augmentation and Early stopping greatly helped in overcoming the problem of overfitting which our initial version of model was facing.
* At last, Transfer learning boosted the overall accuracy of the model. We made use of the [MobileNet](https://arxiv.org/abs/1704.04861) Architecture due to it’s light weight design and high speed performance coupled with low maintenance as compared to other well-known architectures like VGG16, AlexNet, GoogleNet etc.

1. Result

* Model 8– CNN+LSTM, which performed well.
  + (Training Accuracy: 81%, Validation Accuracy: 62%)
  + Number of Parameters (625,029) lower than other model’s parameters.
  + Learning rate gradually decreasing whereas Accuracy increasing after some Epochs.