Classification of Video for Action Recognition

Venkatesh Babu

Sreevatsan A

Overview

Recognizing activities in videos has become a hot topic over the last years due to the continuous increase of video cameras devices and online repositories. This large amount of data requires an automatic indexing to be accessed after capture. The recent advances in video coding, storage and computational resources have boosted research in the field towards new and more efficient solutions for organizing and retrieving video content.
In this project we will use CNN to extract the feature and RNN for sequence to sequence classification.

Goal

The main goal is to observe and analyse human activities and to interpret ongoing events successfully. This is done by obtaining the probabilities of the activity for the whole video as the mean of each activity output through the whole video sequence.

Use Case:

It can be used in assisted living in system for smart homes, Health care monitoring application and surveillance system for indoor and outdoor activities

Data Set:

UCF101 is an action recognition data set of realistic action videos, collected from YouTube
Link: http://crcv.ucf.edu/data/UCF101.php
Data contains:

- 101 action categories
- 13320 videos

Data Pre-processing:

- We split the data into training, testing and validation data set
    1. Training Data = 8596 videos
    2. Testing Data = 1706 videos
    3. Validation Data = 1712 videos
- Using ffmpeg we converted all the videos into frames
- Saved all the frames in the .jpg file format in the respective folders
- Wrote the video file name and number of frames generated into CSV

Approach 1: Using Image (CNN and LRCN)

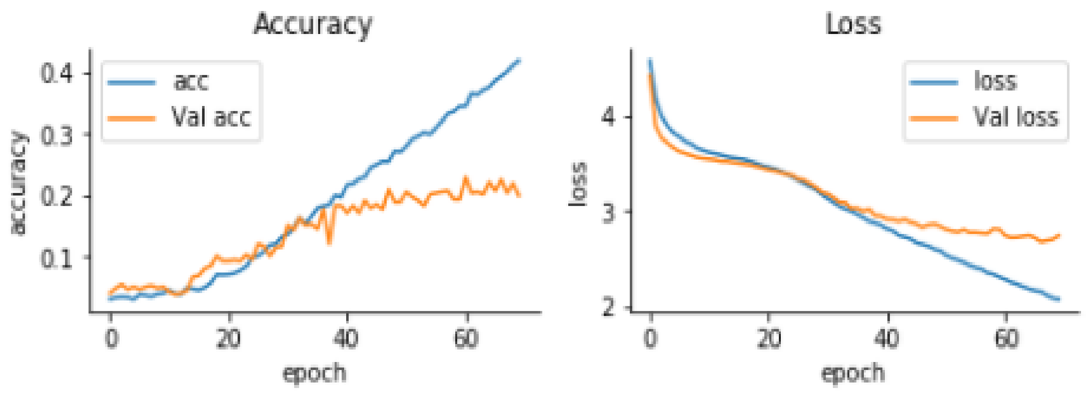
Steps to Follow:

- We gathered all the .jpg file related to the videos
- We rescaled the list of .jpg file in such away we get a same number of image for all the videos
- Build the image sequence with the rescaled list of images
- One hot encode the final Y test and train

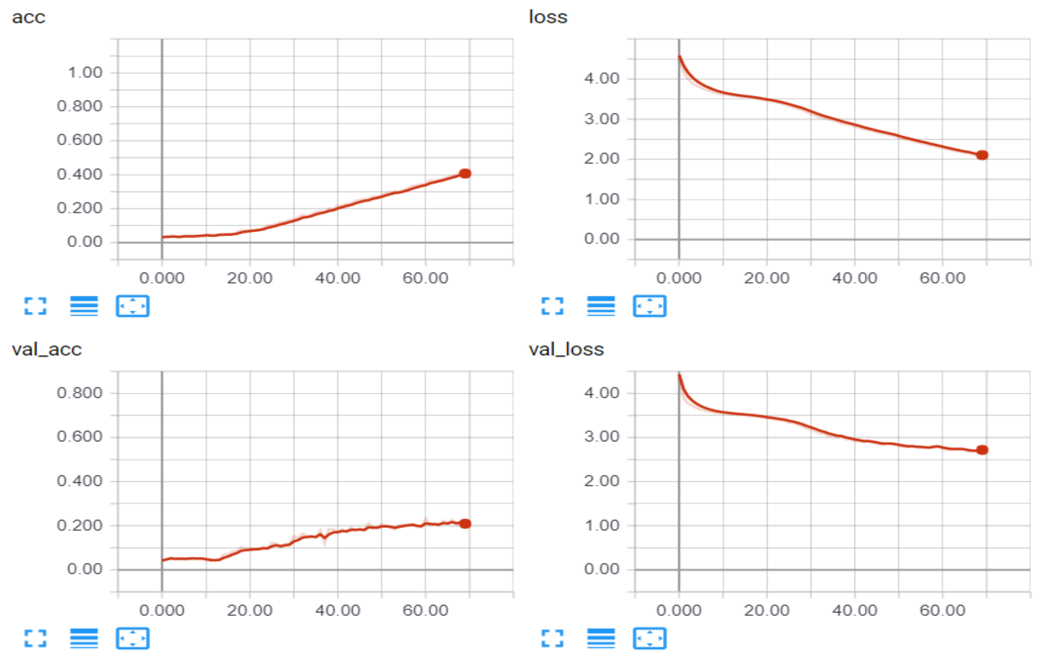
Approach 1 - Model 1: LRCN

- Built a 10-layer convolution 2D layer
- Activation function used Relu and SoftMax
- We are using loss as categorical_crossentropy and optimizer Adam
    - Test loss: 2.7
    - Test accuracy: 0.21

Graph – (Accuracy and Loss)



Graph – (Accuracy and Loss) TensorBoard



Approach 2: Using Features (LSTM and MLP)

Steps to Follow:

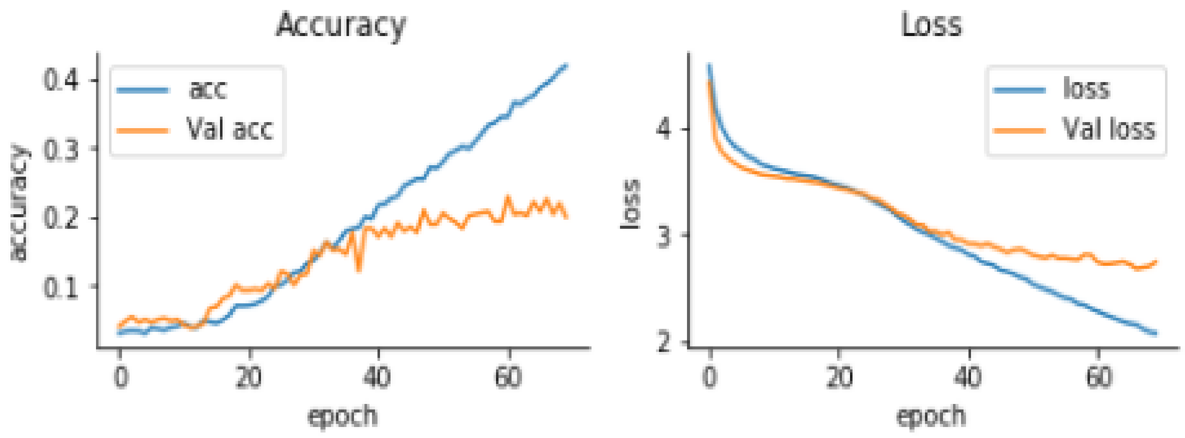
- We downloaded the Inception model with pretrained weights using InceptionV3 package
- We gathered all the .jpg file related to the videos
- We rescaled the list of .jpg file in such away we get a same number of image for all the videos
- Using the inception model we extracted the feature of each image and combine then into a sequence of feature and saved then in .npy file format
- One hot encode the final Y test and train

Approach 2 - Model 1: LSTM

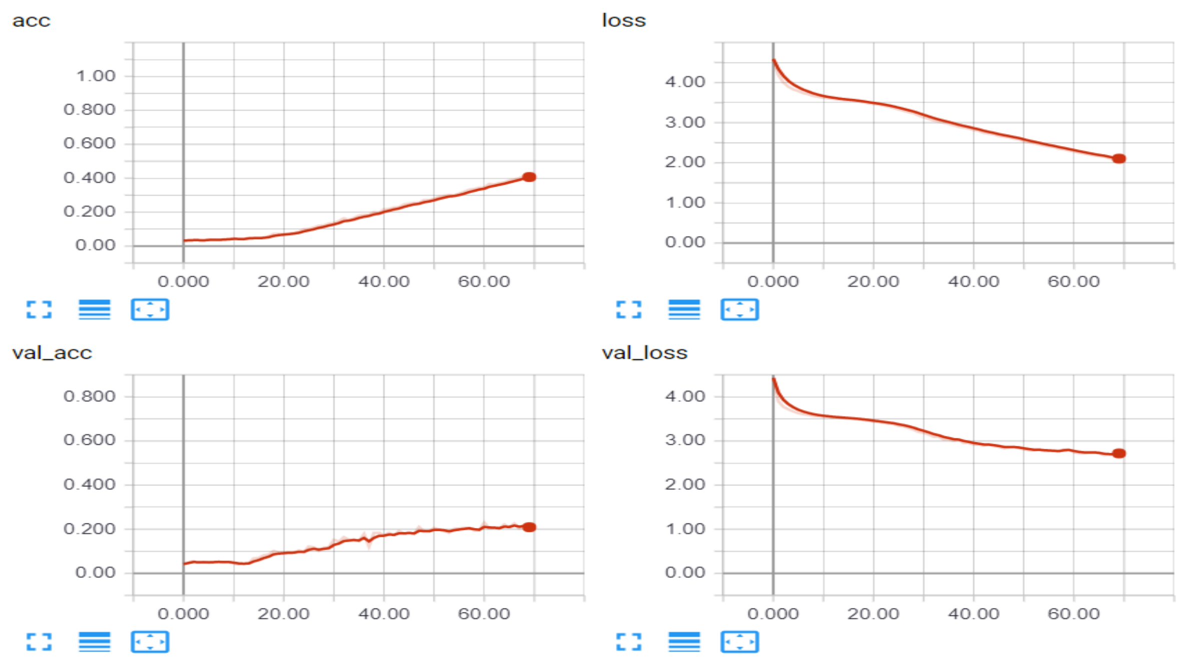
- Built a 3 -layer model with 0.5 percent dropout
- Activation function used Relu and SoftMax
- We are using loss as categorical_crossentropy and optimizer Adam

    - Test loss: 2.7
    - Test accuracy: 0.21

Graph – (Accuracy and Loss)



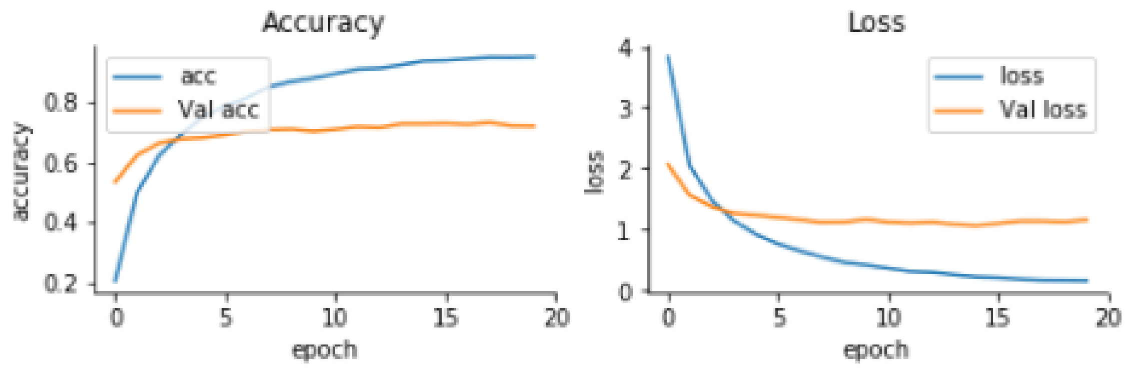
Graph – (Accuracy and Loss) from TensorBoard



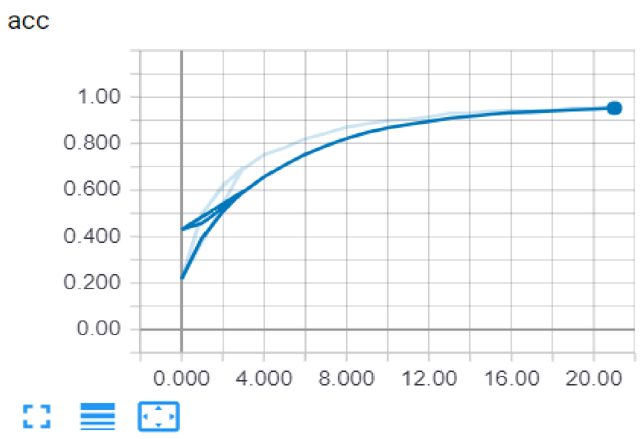
Approach 2 - Model 2: MLP

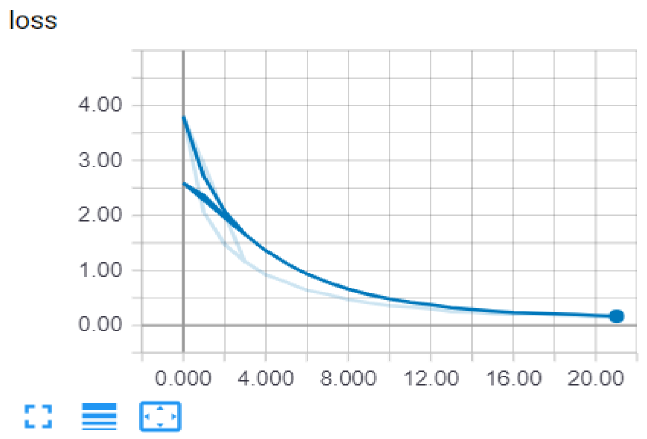
- Built a 2-layer dense model with 0.5 percent dropout
- Activation function used Relu and SoftMax
- We are using loss as categorical_crossentropy and optimizer Adam
    - Test loss: 1.14
    - Test accuracy: 0.72

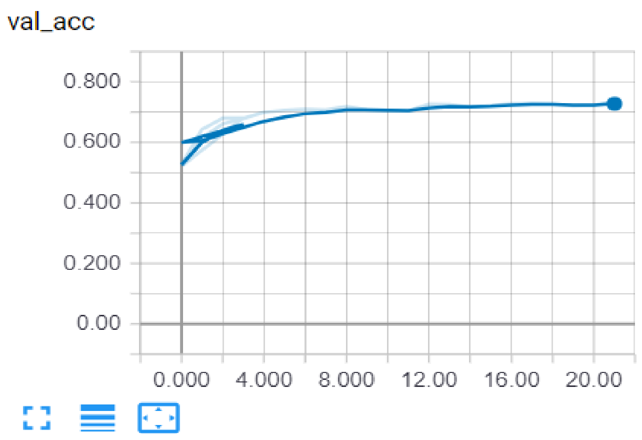
Graph – (Accuracy and Loss)

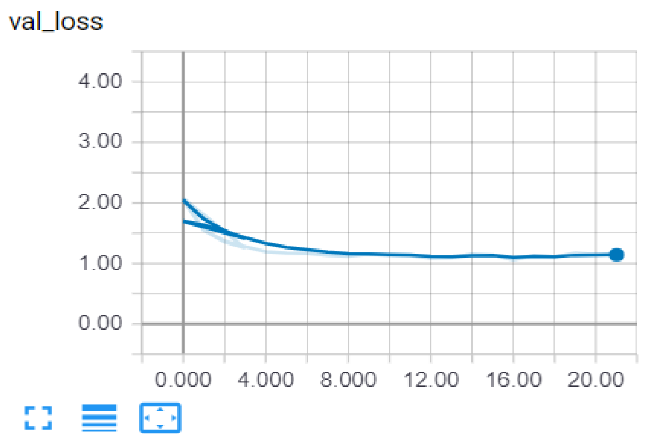


Graph – (Accuracy and Loss) formTensorBoard







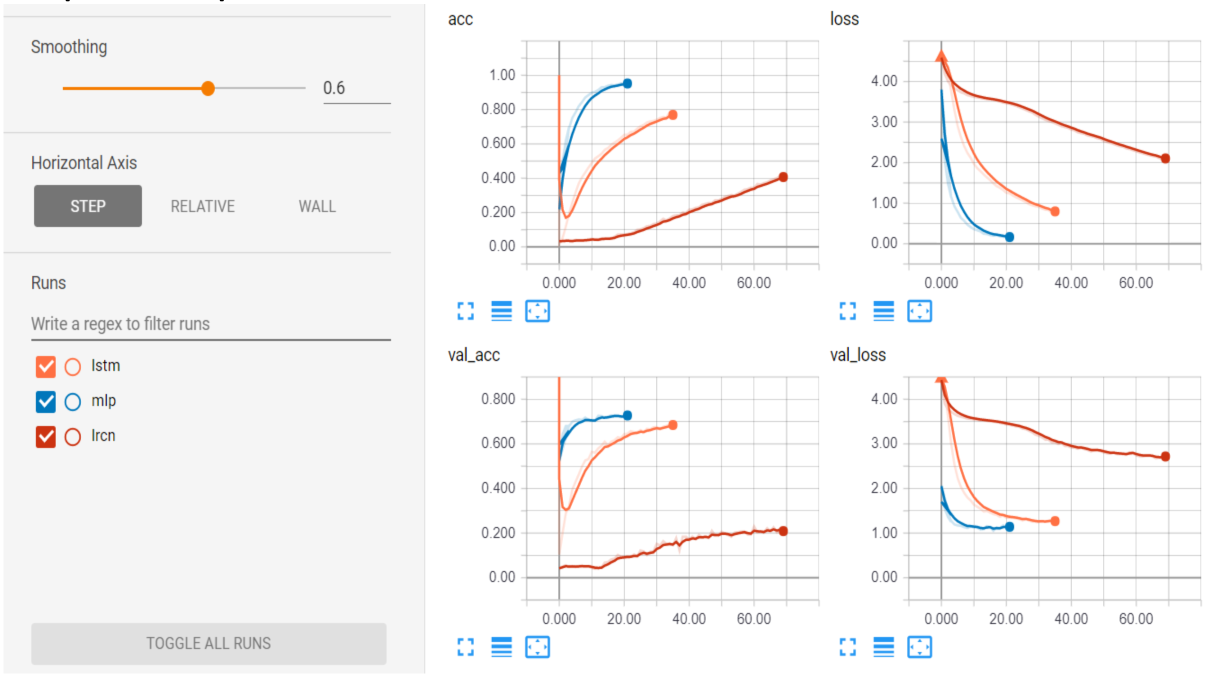


Conclusion:

From the accuracy of the above built model we can say MLP is the best of all the model with 72 percent accuracy

Model	Loss	Accuracy
MLP	1.14	**72**
LSTM	1.21	70
LRCN	2.7	21

Comparison Graph of all three model



Smoothing 0.6

Horizontal Axis: STEP RELATIVE WALL

Runs

Write a regex to filter runs

- ✓ lstm
- ✓ mlp
- ✓ lrcn

TOGGLE ALL RUNS

Reference and Sources:

http://journals.sagepub.com/doi/pdf/10.1177/1550147716665520
http://crcv.ucf.edu/papers/UCF101_CRCV-TR-12-01.pdf
https://arxiv.org/pdf/1406.2199.pdf