Capstone Project Report

Technique

We developed three different deep neural network architectures for the task of online video classification. In the first approach, we classified one frame at a time with a convolutional neural network. In this approach, we classified each clip based on a single frame, ignoring the temporal features of the video clip. We used the concept of transfer learning for feature extraction by using an Inception-v3 convolutional neural network pre-trained on the ImageNet dataset. We fine tuned the top dense layers for 10 epochs to ensure that the previous learning was retained. Then we re-trained the top 2 inception blocks with UCF101 dataset.

In the second approach, we extracted features using a convolutional neural network and passed those features to a recurrent neural network to learn the temporal component of video data. We ran each frame of the video clips through a pre-trained Inception-v3 model and extracted a 2048-d vector of features by cutting the network at the final pool layer. We converted these features into sequences to be able to give them as an input to the recurrent neural network. We used a 2048-wide long short term memory (LSTM) layer, followed by a 1024 dense layer, with dropout layers in between for the recurrent neural network.

In the third approach, we extracted features using a convolutional neural network and passed those features to a multilayer perceptron to learn the temporal component of video data. We extracted features by cutting the pre-trained Inception-v3 model like the previous approach. We converted these features into sequences and flattened them into an input vector to be able to give them as an input to a multilayer perceptron. We used a two-layer multilayer perceptron network with 512 neurons per layer for this approach.

Results

Randomly guessing the class of videos is one of the baseline models we considered for comparing the results of our DNN models. Since the UCF101 dataset contains a total of 101 classes, the baseline model gives an accuracy of 0.9%. The one frame at a time convNet gives a top-5 validation accuracy of 65.12% which is much more than the accuracy of the baseline model. This approach does not consider the temporal component of video data. The approach of passing features extracted from convNet to a long short term memory (LSTM) recurrent neural network gives a top-5 validation accuracy of 89.06%. We got a much better accuracy for this architecture as compared to the first one because a recurrent neural network is used

to learn the temporal features of video data. The third approach of passing features extracted from convNet to a multilayer perceptron gives a top-5 validation accuracy of 90.94%.

Architecture	**Training Accuracy**	**Validation Accuracy**	**Top-5 Accuracy**
One frame at a time CNN	52.20%	48.71%	65.12%
ConvNet features passed to LSTM RNN	57.88%	62.97%	89.06%
ConvNet features passed to MLP	84.49%	73.75%	90.94%

Detailed Report –

1. Video classification Problem

We aim to solve the problem of recognizing what actions are being performed in a video by using various deep learning techniques. There are a wide variety of applications of video classification such as surveillance cameras, automated attendance systems, security monitoring systems, etc.

2. Description of the data set

UCF101 is an action recognition data set of realistic action videos, collected from YouTube, having 101 action categories. With 13320 videos from 101 action categories, UCF101 gives the largest diversity in terms of actions and with the presence of large variations in camera motion, object appearance and pose, object scale, viewpoint, cluttered background, illumination conditions, etc, it is the most challenging data set to date. As most of the available action recognition data sets are not realistic and are staged by actors, UCF101 aims to encourage further research into action recognition by learning and exploring new realistic action categories. [1]

3. Data set Summary

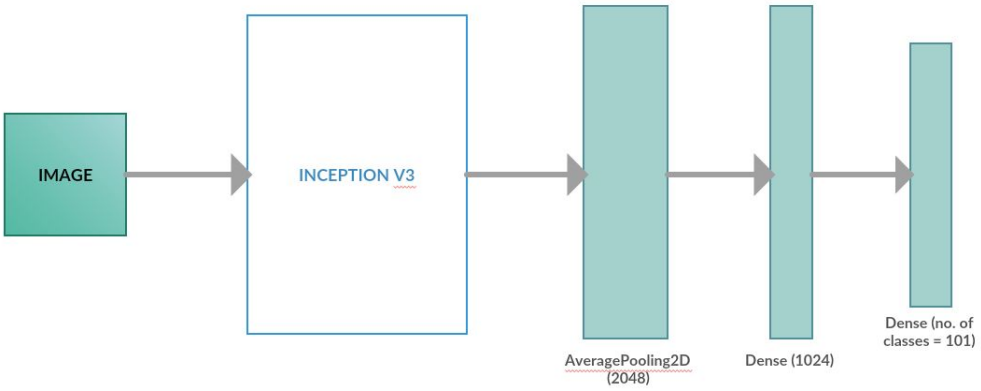
The videos in 101 action categories are grouped into 25 groups, where each group can consist of 4-7 videos of an action. The videos from the same group may share some common features, such as similar background, similar viewpoint, etc. The action categories can be divided into five types:

1)Human-Object Interaction
2) Body-Motion Only
3) Human-Human Interaction
4) Playing Musical Instruments
5) Sports

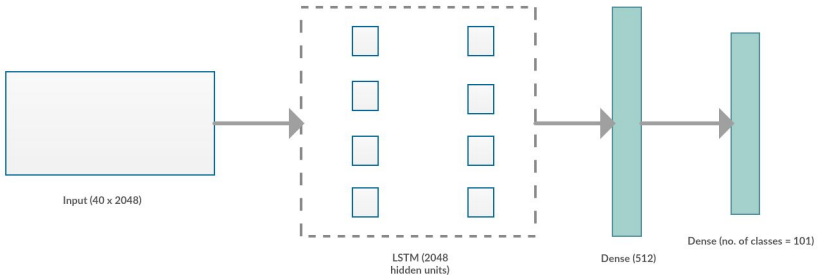
Actions (Classes)	101
Clips	13320
Groups per Action	25
Clips per Group	4-7
Mean Clip Length	7.21 sec
Max Clip Length	71.04 sec
Min Clip Length	1.06 sec
Total Duration	1600 mins
Frame Rate	25 fps
Resolution	320×240
Audio	Yes (51 actions)

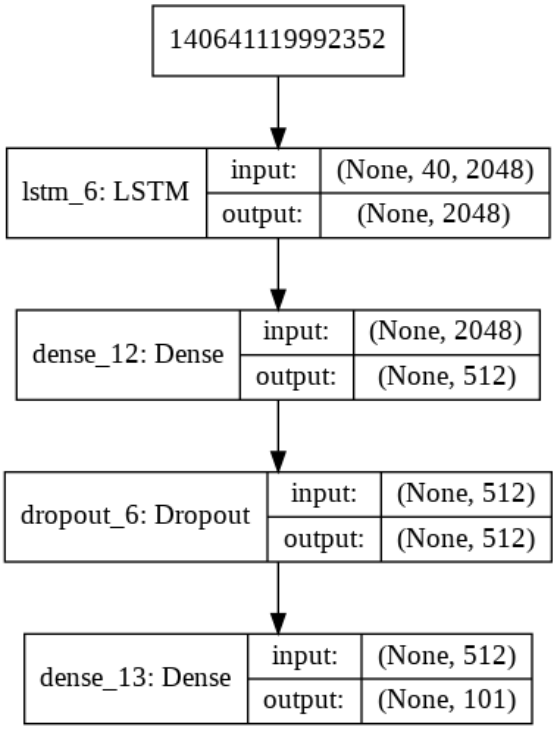
4. Architecture Visual Graphs

1. CNN Architecture

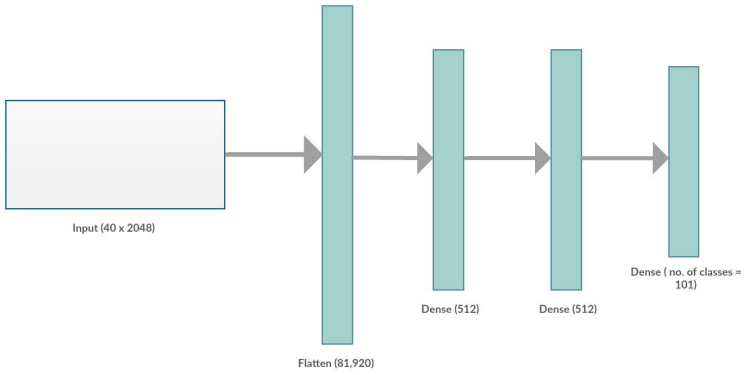


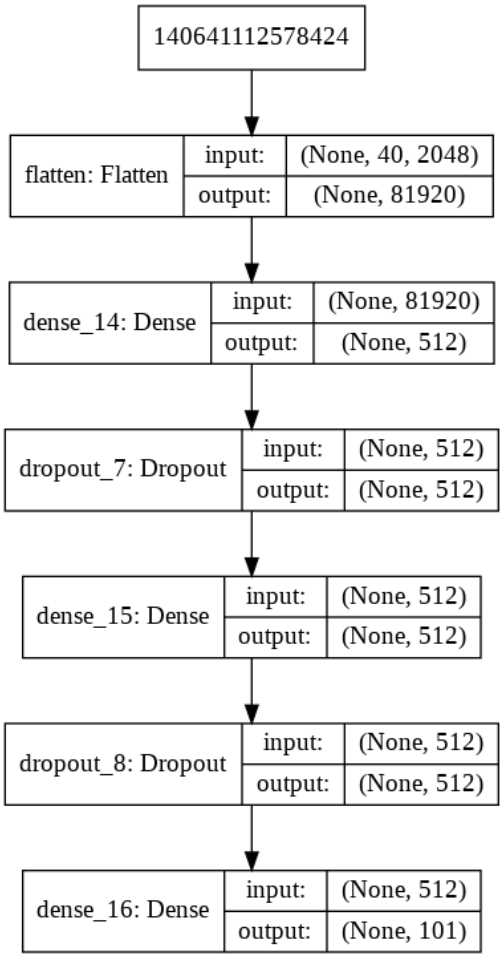
2. LSTM Architecture





3. MLP Architecture



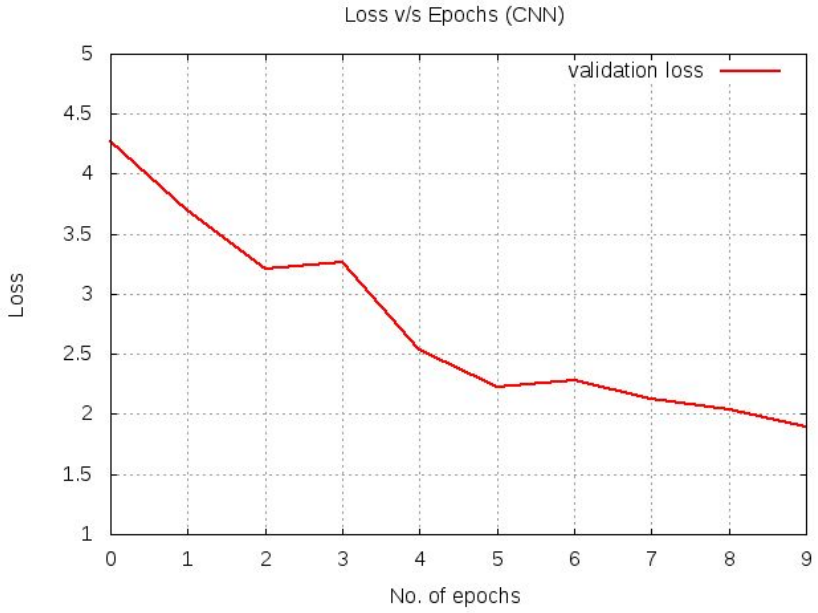


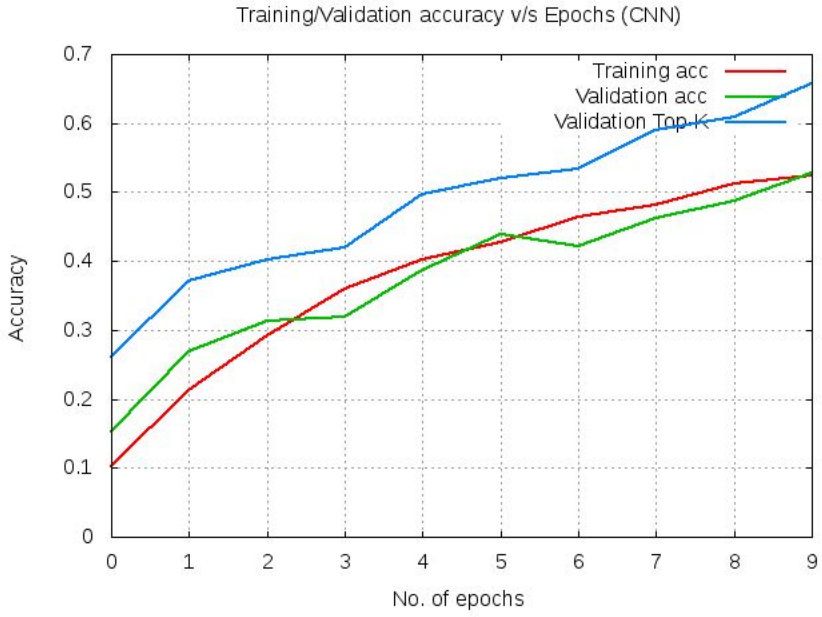
5. Architecture Hyperparameters

Hyperparameter	**Architecture 1 – CNN**	**Architecture 2 – LSTM**	**Architecture 3 – MLP**
Activation Function	relu	relu	linear
# Hidden Layers	4	2	3
# Units per Layer	1024	2048	512
Optimizer	SGD	Adam	Adam
Learning Rate	10^{-4}	10^{-5}	10^{-5}
Dropout Rate	-	0.5	0.5
Batch Size	16	32	32
# Epochs	10	10	10

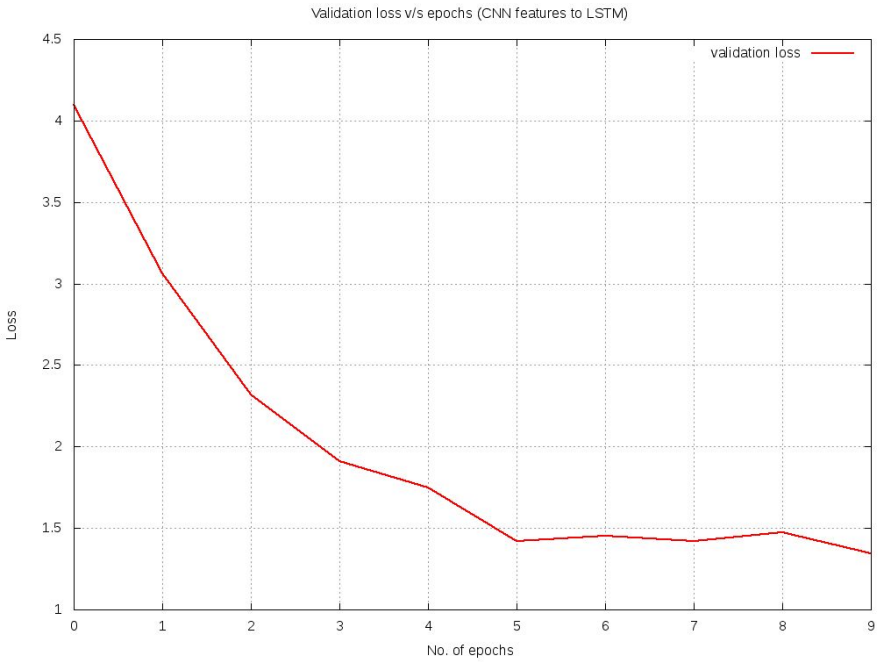
6. Loss and Accuracy Graphs

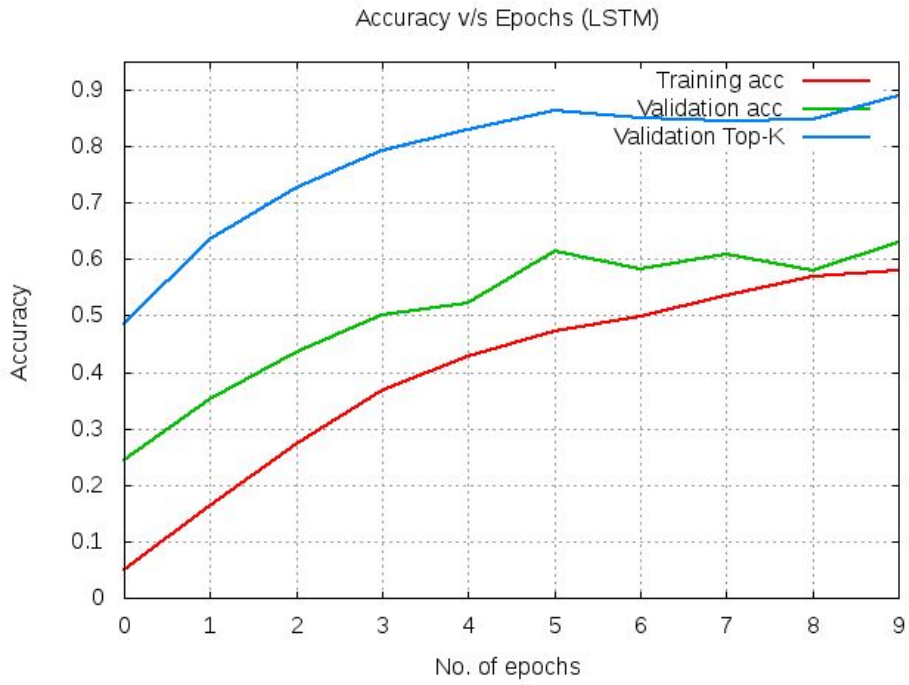
1. CNN Approach



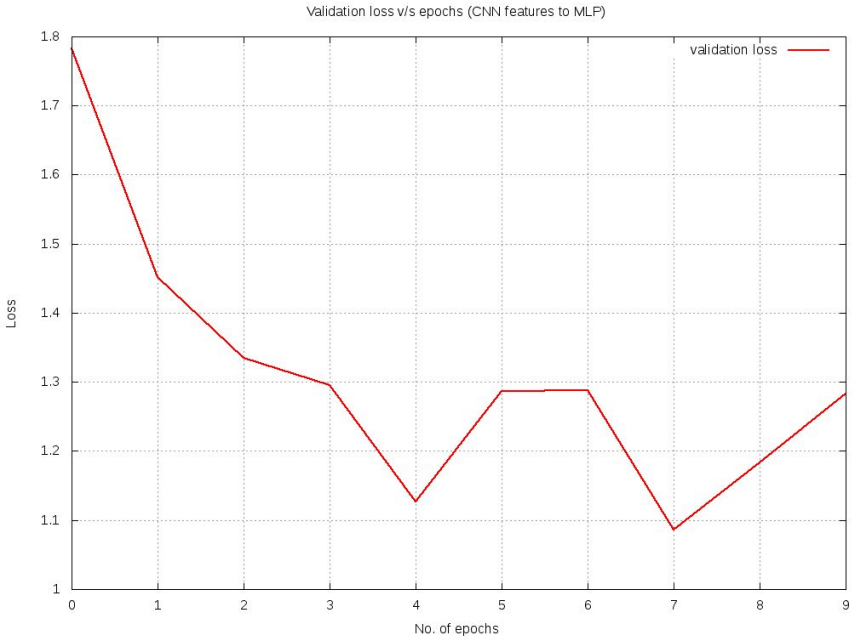


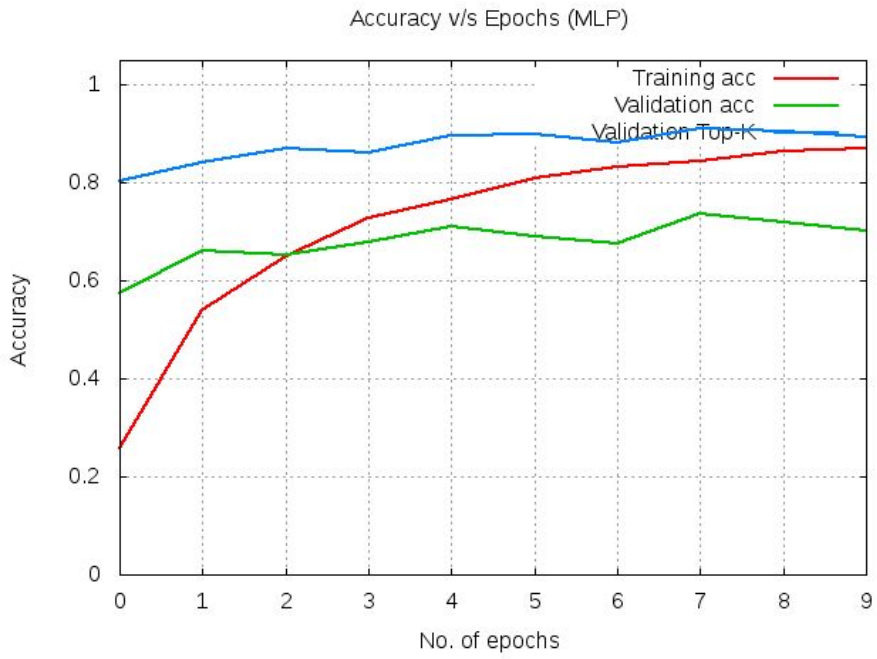
2. LSTM Approach





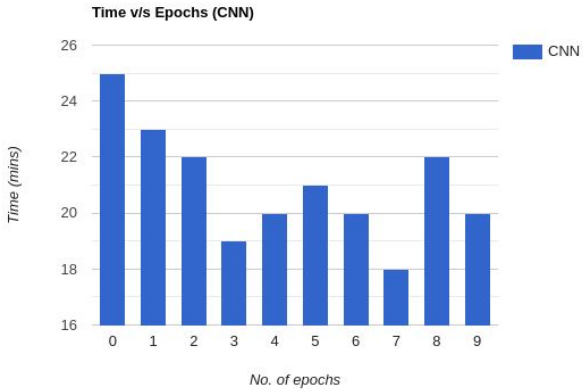
3. MLP Approach



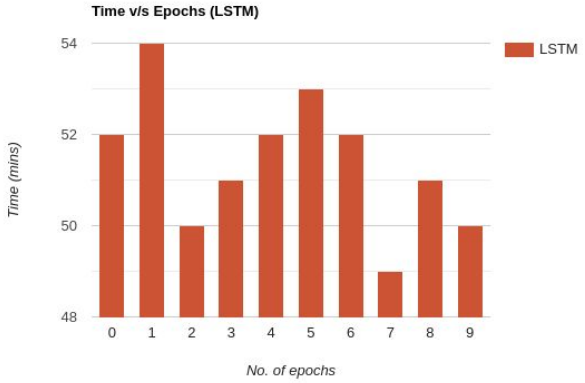


7. Time Graphs

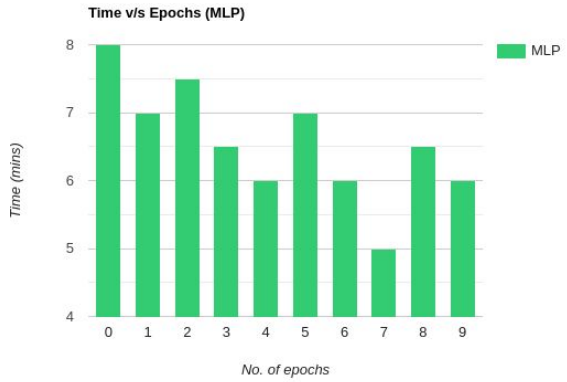
1. CNN Approach



2. LSTM Approach



3. MLP Approach



8. Hyperparameter Tuning

- **Learning Rate**

If the learning rate is too small, it takes a longer time (hundreds or thousands) of epochs to reach the ideal state. On the other hand, if the learning rate is much larger than optimal value then it would overshoot the ideal state and our algorithm might not converge. For the CNN model, we started with 0.01, which is the default for SGD optimizer, but the algorithm did not converge. [2] We then reduced it to 0.0001 which yielded better results. For LSTM and MLP models, we started with the default learning rate of ADAM optimizer - 0.001 but again the algorithm did not converge. We then chose 0.00001.

- **Optimizer**

A simple stochastic gradient descent optimizer worked well for the CNN model. For LST and MLP, we chose a very popular optimiser specifically designed for tuning deep neural networks called Adam. Adam stands for adaptive moment estimation and is another way of using past gradients to calculate current gradients. Adam also utilizes the concept of momentum by adding fractions of previous gradients to the current one. [3]

- **Dropout Rate**

A dropout layer specifies the probability at which outputs of the layer are dropped out, or inversely, the probability at which outputs of the layer are retained. The dropout rate is between 0 and 1. Higher values retain inputs from the visible layer and a value closer to 0.5 retains outputs in each hidden layer. [4] We initially chose a dropout rate of 0.5 for the LSTM and MLP models and we found this worked well so we did not change it.

- **Batch Size**

A larger batch size allows computational boosts that utilizes matrix multiplication in the training calculations but that comes at the expense of needing more memory for the training process. A smaller minibatch size induces more noise in their error calculations and often more useful in preventing the training process from stopping at local minima. [2] We first tried using a batch size of 32 but this resulted in the GPU running out of memory. Therefore, we used batch size 16.

- **Epochs**

We had computational constraints given the massive size of the data and therefore we limited all models to train for 10 epochs.

References

[1] “UCF 101 Action Recognition Data Set” [Online] Available : https://www.crcv.ucf.edu/data/UCF101.php [Accessed: 03 May 2019]

[2] “Hyperparameters in Deep Learning”, [Online] Available : https://towardsdatascience.com/hyperparameters-in-deep-learning-927f7b2084dd [Accessed: 03 May 2019]

[3] “Introduction to Optimizers”, [Online] Available : https://blog.algorithmia.com/introduction-to-optimizers/ [Accessed: 03 May 2019]

[4] Jason Brownlee, “A Gentle Introduction to Dropout for Regularizing Deep Neural Networks” [Online] Available : https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/ [Accessed: 03 May 2019]

[5] Matt Harvey, “Five video classification methods implemented in Keras and TensorFlow” [Online] Available : https://blog.coast.ai/five-video-classification-methods-implemented-in-keras-and-tensorflow-99cad29cc0b5 [Accessed: 03 May 2019]