Video Classification on UCF101 Dataset
-Suraj Kumar Sahoo, 142202016

1. Introduction

1.1. Objective

The objective of the project is to recognize and classify the action in a video. The labels given in the videos are human actions.

1.2. About the data

The dataset we are provided with is the UCF101 dataset. We shall not be classifying all the classes, instead try to classify videos into a set of 21 classes which were formed after selectively combining 32 classes of the original UCF101 classes. We used the already split train and test data for training and validating respectively. We has a total of 3035 training samples and 1214 testing samples. Note that, none of the videos from training and testing belong to the same group of videos. So if the model is detecting an action correctly, it is doing it correctly mostly because it learnt the action in the video. One can refer to (3) for further knowledge on the dataset.

1.3. Introduction

We know that in order to work with videos we need to look at both the spatial and temporal knowledge of the data(video), where spatial part tells us about the subject in the frame and temporal part justifies the movement of the subject or action of the subject. Therefore, we shall require the use of both CNN and RNN for taking care of the above.

CNN: This network will provide us with the relevant feature maps that will help us in detecting patterns and features of the image that are spatially close to each other.

RNN: This will help us in detecting the sequential knowledge of the data and memorize the information in the previous time step and capture the temporal dependencies between them. We will be using the one of the most popular RNN Model, namely LSTM for our models.

We have also considered Bi-Directional LSTM for these models. Since, remembering what is happening afterwards can also help in better judgement. This is what any normal viewer would do; look at the previous steps after reaching the end, for obtaining a better concept of the work being done.

2. Frame Selection

2.1. Image Histogram

Histogram plotted for pixel intensity values in an image. It shows the number of pixels that share a same intensity value. It is a good way to characterize an image. This can be viewed as a distribution function.

A difference in the distribution function implies that the two images are significantly different. In a video, we can clearly say that 2 consecutive frames do not differ from each other that much and hence their histograms also tend to be equal in nature.

To make this to our use we need to find a measure that finds the difference between two frequency distributions. We can then use this method to find how related two images are.

We opted to use Bhattacharya distance as a measure to fins the same which has been discussed in the next section.

2.2. Histogram Differencing (Bhattacharyya Distance)

Bhattacharyya distance measures the similarity of two probability distribution using the formula given below. The discrete form of the formula is put to use since the pixel probability distribution considers domain as the discrete pixel values ranging from 0 to 255. This formula gives the values considering one channel. We can extend it to 3 channels by summing up the similarity score.

The Bhattacharya distance for 2 distribution P and Q (on the same domain) is given by,

$$D_B(P,Q) = -ln(BC(P,Q)) \tag{1}$$

where,

$$BC(P,Q) = \sum_{x \in X} \sqrt{P(x) * Q(x)} \tag{2}$$

2.3. How did we put this to use?

We calculated the Bhattacharya Distance(BD) between the first 2 consecutive frames. Consider two pointers on the 2

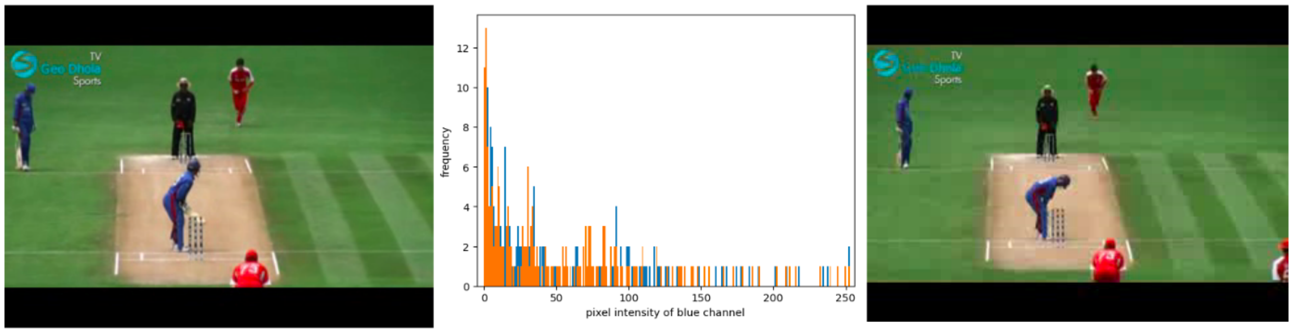


Figure 1. An example to show how histogram changes, with change in frames. This histogram only shows pixel intensity count information on blue channel of the image

frames, i and j. if the distance is more than a threshold(which is adaptive to the videos) we choose both the frames and move the first pointer, i, to the second frame, and the second pointer(j) to the third frame. Otherwise, pointer i remains where it was and we move the second pointer to the next frame and again find the BD between the two.

Iteratively, we swap until we reach the end of the video frames. If the number of frames is not yet the number we wanted we lower the threshold and again try to find all the frames that are 'this threshold' apart.

At the end we will end up with a set of frames that are as apart from each other as they can be considering the number of frames we are considering. We shall go with 20 number of frames for all our experiments.

The algorithm for the same is given in [24].

If we look at the images and their histograms given in [1] we can clearly see that they don't differ much except for a few pixel intensity. This is solely because of the movement of the subject in the image and hence change in the intensity at different parts of the image. We try to quantify that change using BD and take necessary steps for finding the best significant frames.

2.4. Other Frame Selection Methods

We also chose to check how did our models work with other type of choices of the frames, including $skip\ interval$ method, and $random\ selection$ method. In the skip interval method, we chose frames after skipping a bunch of frames, and this number of frames is nothing but the value of $\frac{F}{k}$ where F is the number of frames and k is the no of frames required to fetch from the video.

3. CNN + LSTM

To start with video classification we chose to first try out a simple CNN + LSTM model. CNN Model consists of a series of Convolution Layers including MaxPooling/AvgPooling, followed by an LSTM Layer. The results of these experiments are given at the end.

4. ConvLSTM

This model architecture was first introduced in (1), which incorporated the architecture of LSTM with images by transforming the inputs and hidden states as images/matrices and replaced element wise multiplication with convolution operations in both input to state and state to state transitions. This particular model is suitable for temporal data involving images, therefore, suitable for videos. They have used this model for precipitation now-casting using an Encoder-Prediction Structure. We used this model for our classification purposes. The detailed algorithm of the ConvLSTM (which is almost similar to that of an LSTM architecture) can be seen in (1).

NOTE: The results that we obtained in these two models were quite unsatisfactory yet possible because of the depth of the model being very less, and since the training set we have, can only train the model on a very fixed ranges of images. The test/validation set is a very different image altogether, which results in not so significant feature maps that can capture good knowledge of the frames. On increasing the depth we will face the problem of increase in number of parameter and less number of data, that will only worsen the situation. hence we opted for Transfer Learning, discussed in the next section.

5. Transfer Learning

On realising the limitations of custom CNN networks and ConvLSTM, we chose to use the CNN models that were trained on a larger dataset so as to give us a better feature maps and for this purpose we chose a variety of learnt models including VGG16, ResNet50, Xception and EfficientNetV2L. The choice of EffNetV2L is based on the fact

Algorithm 1 Feature Selection using histogram distance

```
1: Consider $BD(f_i, f_j)$ be the function that gives the Bhattacharyya Distance between frames $i$ and $j$.
2: Let, $F$ be the number of frames in the video, $N$ be the required number of frames that one wants to get from the video.
3: $i = 1, j = 2$                    ▷ initialising the pointers
4: $\epsilon = \epsilon_0, \eta = \eta_0$                    ▷ initialising the threshold and its updation term
5: $SigFrames = [f_i]$                    ▷ array containing all the significant frames
6: while True do
7:     while $j < F$ do
8:         $bd = BD(f_i, f_j)$
9:         if $bd > \epsilon$ then
10:            $SigFrames \leftarrow f_j$
11:            $i \leftarrow j, j+ = 1$
12:        else
13:            $j+ = 1$
14:        end if
15:    end while
16:    if $|SigFrames| < N$ then
17:        $\epsilon \leftarrow \epsilon - \eta$          ▷ one can choose $\eta$ adaptively so that one receives the required number of frames , $N$
18:    else if $|SigFrames| > N$ then
19:        $\epsilon \leftarrow \epsilon + \eta$
20:    else
21:        Steps to make sure we get frames from starting till end so that, we don't loose on information from later part of the video
22:        $break$
23:    end if
24: end while
```

that it had the best Top1 and Top5 Accuracy. The rest were chosen to check our dataset on a larger spectrum of models. We incorporated a LSTM layer after extracting the feature maps off the videos. We did the training on all the 3 types of frame choices. The results are given in the results section.

6. Attention

Since we are dealing with a number of frames we might as well can take into account the possibility of attending more the important frames that take part in decision making. For this we introduced an attention layer into the model.

This layer can be expressed as a series of steps:

- Transforming the feature maps(for each frame) to $query$ and key
- Finding similarity between them to get the attention values for each feature map/ frame.(matrix multiply the two transformations)
- using softmax at the end for making the similarity to weights that sum up to one.
- Outputs the attention-weighted sum of the inputs.

The attention layer is depicted in the figure [2].

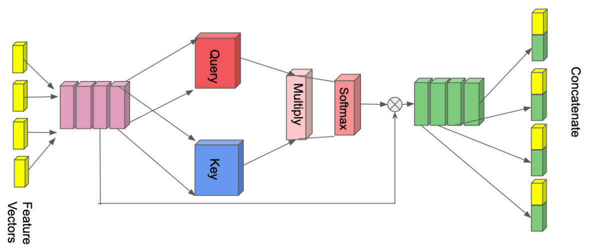


Figure 2. Attention Architecture

After obtaining the feature maps from a CNN model, we passed it through the attention layer which is followed by the BiDirectional LSTM layer that takes in input the concatenation of both the actual feature map that we obtained from the CNN and the attention layer's output. This, theoretically should improve the performance of the model. The results are presented in the results section.

7. Positional Encoding and Multi-head Attention

We also tried to discover how well does our model learn, if the temporal learning, done earlier with LSTM or BiDirectional LSTM and simple attention is switched with po-

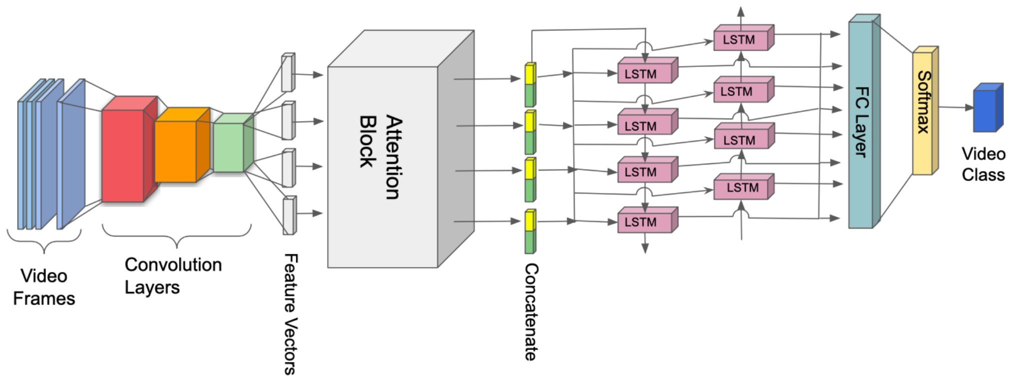
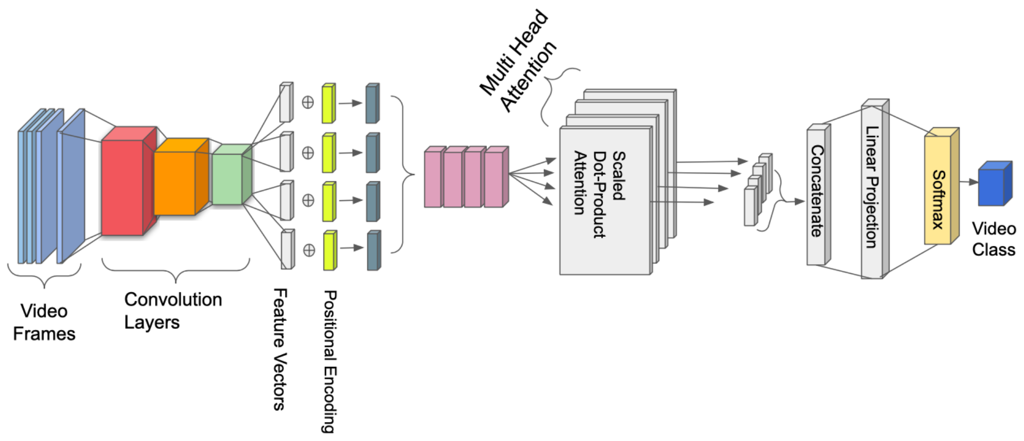
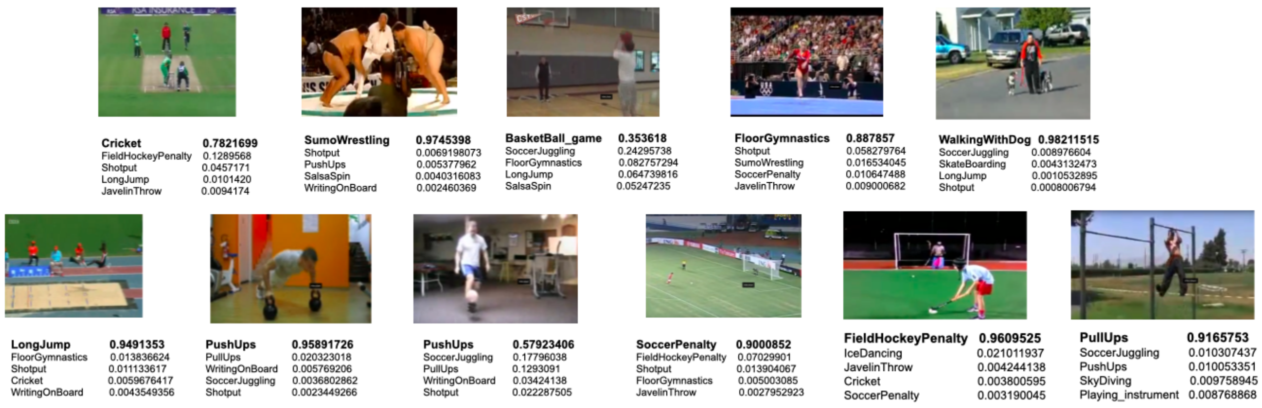


Figure 3. CNN+ Attention + BiDirectional LSTM Architecture





Cricket	**0.7821699**
FieldHockeyPenalty	0.1289568
Shotput	0.0457171
LongJump	0.0101420
JavelinThrow	0.0094174

SumoWrestling	**0.9745398**
Shotput	0.0069198073
PushUps	0.005377962
SalsaSpin	0.0040316083
WritingOnBoard	0.002460369

BasketBall_game	**0.353618**
SoccerJuggling	0.24295738
FloorGymnastics	0.082757294
LongJump	0.064739816
SalsaSpin	0.05247235

FloorGymnastics	**0.887857**
Shotput	0.058279764
SumoWrestling	0.016534045
SoccerPenalty	0.010647488
JavelinThrow	0.009000682

WalkingWithDog	**0.98211515**
SoccerJuggling	0.008976604
SkateBoarding	0.0043132473
LongJump	0.0010532895
Shotput	0.0008006794

LongJump	**0.9491353**
FloorGymnastics	0.013836624
Shotput	0.011133617
Cricket	0.0059676417
WritingOnBoard	0.0043549356

PushUps	**0.95891726**
PullUps	0.020323018
WritingOnBoard	0.005769206
SoccerJuggling	0.0036802862
Shotput	0.0023449266

PushUps	**0.57923406**
SoccerJuggling	0.17796038
PullUps	0.1293091
WritingOnBoard	0.03424138
Shotput	0.022287505

SoccerPenalty	**0.9000852**
FieldHockeyPenalty	0.07029901
Shotput	0.013904067
FloorGymnastics	0.005003085
JavelinThrow	0.0027952923

FieldHockeyPenalty	**0.9609525**
IceDancing	0.021011937
JavelinThrow	0.004244138
Cricket	0.003800595
SoccerPenalty	0.003190045

PullUps	**0.9165753**
SoccerJuggling	0.010307437
PushUps	0.010053351
SkyDiving	0.009758945
Playing_instrument	0.008768868

Figure 5. top 5 predictions on a random set of 11 videos

Table 1. Model Performance

Model(Model Number)	Frame Selection	Trainable parameters	Training Accuracy	Validation Accuracy
CNN+LSTM (2)	skip	97,834	1.0000	0.4659
ConvLSTM (1)	skip	534,792	0.9137	0.4800
	histograms	534,792	0.9200	0.5142
Pretrained Models				
VGG16 + LSTM (3)	skip	330,901	0.9967	0.8180
	random	330,901	0.9687	0.8180
	histograms	330,901	0.9633	0.7817
Xception + LSTM (4)	skip	21,040,277	0.9688	0.7593
	random	21,040,277	0.8218	0.7633
	histograms	21,040,277	0.8835	0.7826
ResNet50 + LSTM (5)	skip	1,117,333	0.8736	0.7083
	random	1,117,333	0.8929	0.8097
	histograms	1,117,333	0.8835	0.7826
EfficientNetV2L + LSTM (6)	skip	724,117	0.9229	0.8229
	random	724,117	0.9555	0.8336
	histograms	724,117	0.9456	0.8394
EfficientNetV2L + BiDirectional LSTM (7)	histograms	1,448,213	0.9901	0.8723
	random	1,448,213	0.9911	0.8756
EfficientNetV2L + Attention + BiDirectional LSTM (8)	skip	1,919,637	0.9921	0.8616
	random	1,919,637	0.9802	0.8707
	histograms	1,919,637	0.9924	0.8789
EfficientNet + Positional Encoding + SelfAttention (9)	histogram	54,742,125	0.9450	0.8328

sitional encoding and Multi-head Attention. We will not require the use of a recurrent network here. We trained this model only with the frames, selected using our adaptive frame selection method only. The basic idea behind the positional encoding is the adding the information of the frame position in the video (we used the embedding of the tensor containing values 1 to number of frames in input). The Multi-head attention layer(of keras, that we used) starts with linearly transforming input into a query, key, value and then, it does dot product of query and keys and is then scaled. This value is then softmaxed to obtain the probability of attention. These are then used to get attended values. These values are then concatenated to single tensor and then projected linearly to required dimension and outputted. This architecture of it is also given in the figure [4]. This part of the model is a derivative of the encoder part of the Transformer model as in (2).

8. Results

We have conducted all the experiments mentioned in the table [1]. We observed the best performance in model with features extracted using transfer learning from EfficientNetV2L with an Attention layer followed by BiDirectional LSTM.

9. Contribution

My contribution to the project are: Building the Adaptive Frame Selection to find the most significant frames , Building and Training models (refer to 1) (1)-(9) on frames selected using the Adaptive Frame Selection Methods.

Combined Contribution: Literature Study, Data handling, pre-processing, Report(Architecture Diagrams, Tables)

10. Reference

References

[1] Xingjian Shi et al.,(2015), Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting, https://arxiv.org/abs/1506.04214

[2] Ashish Vaswani et al., (2017), Attention Is All You Need, https://arxiv.org/abs/1706.03762

[3] The UCF101 Dataset, https://www.crcv.ucf.edu/data/UCF101.php

[4] https://en.wikipedia.org/wiki/Bhattacharyya_distance