Temporal Segment Networks

Anupam Arvind Srivastava (2018A3PS0433P)
Ayush Sharma (2018A3PS0326P)
Samir Patro (2018B3A30746P)

Mentors – **Dr. Surekha Bhanot , Dr. Bijoy .K Mukherjee**
Achleshwar Luthra

Authors

Limin Wang: BS in NJU, PhD in CUHK with Xiaoou Tang, now postdoc in ETHZ.

Yuanjun Xiong: BE in Tsinghua, PhD in CUHK with Xiaoou Tang, now postdoc in CUHK.

Zhe Wang: BE in ZJU, PhD in CUHK with Xiaogang Wang. • Yu Qiao (乔宇): Professor in SIAT.

Dahua Lin: Professor in CUHK. BS in USTC, PhD in MIT.

Xiaoou Tang: Professor in CUHK. BE in USTC, PhD in MIT. • Luc Van Gool: Professor in ETHZ.

Sneak Peek

“Whatever you are studying right now, if you are not getting up to speed on deep learning, neural networks, etc., you lose. We are going through the process where software will automate software, automation will automate automation.”

For image recognition, deep convolutional networks have had a lot of success. However, their superiority over conventional approaches of video action detection is less clear. For learning action models in videos, this paper presents a general and flexible video-level framework.

The temporal segment network (TSN) methodology uses a new segment-based sampling and aggregation module to model long-term temporal structures. TSN can efficiently practise action models by using whole action videos thanks to this one-of-a-kind design.

The learned models could be easily updated for behaviour detection in both trimmed and untrimmed images using simple average pooling and multi-scale temporal window integration.

Why?

Video-based action recognition has drawn considerable attention from the academic community owing to its applications in many areas like security and behavior analysis. Traditionally, to predict actions in a video, dense methods have been used which are computationally intensive tasks.

About our Project

The main idea behind this approach is that consecutive frames are highly redundant, where a sparse and global temporal sampling strategy would be more favorable and efficient in this case.

The TSN framework first extracts short snippets over a long video sequence with a sparse sampling scheme, where the video is first divided into a fixed number of segments and one snippet is randomly sampled from each segment. Then, a segmental consensus function is employed to aggregate information from the sampled snippets.

By this means, temporal segment networks can model long-range temporal structures over the whole video, in a way that its computational cost is independent of the video duration.

Motivation

“All you need is lots and lots of data and lots of information about what the right answer is, and you'll be able to train a big neural net to do what you want.”

Long-term temporal modelling is critical for understanding action in videos. Current deep architectures, such as two-stream ConvNets and 3D convolutional networks, are designed to run on a single frame or a stack of frames (e.g., 16 frames) with small temporal durations.
As a result, these structures lack the ability to incorporate long-term temporal knowledge from images into the learning of action models.

It was observed that although the frames are densely recorded in the videos, the content changes relatively slowly. This motivates us to explore a new paradigm for temporal structure modeling, called segment based sampling. Concerning the property of spareness, only a small number of sparsely sampled snippets would be used to model the temporal structures in a human action. Normally, the number of sampled frames for one training iteration is fixed to a predefined value independent of the durations of the videos. This guarantees that the computational cost will be constant,regardless of the temporal range we are dealing with.On the global property, our segment based sampling ensures these sampled snippets would distribute uniformly along the temporal dimension.

Framework and Formulation

The aim was to use a new segment-based sampling technique to create an effective and efficient video-level structure, dubbed Temporal Segment Network (TSN). Temporal segment networks operates on a series of short clips sampled from the entire recording, rather than a single frame or a short frame stack.

To ensure that these sampled snippets accurately represent the contents of the entire video while maintaining a reasonable computational cost, our segment-based sampling divides the video into several segments of equal duration, and then one snippet is randomly sampled from each segment.

Each snippet in this sequence generates its own snippet-level prediction of the action classes, and a consensus function is used to aggregate these snippet-level predictions into video level scores. Because it captures long-term information from the entire video, this video-level prediction is more reliable and informative than the original snippet-level prediction.The optimization objectives are defined on the video-level predictions during the training process and optimised by iteratively updating the model parameters.

Framework and Formulation

Formally, given a video V, we divide it into K equal-length segments S_1, S_2,.... S_K. One snippet T_k is chosen at random from its corresponding segment Sk. The temporal segment network then models a series of snippets (T_1, T_2,...,T_K) as follows:

$$\mathrm{TSN}(T1, T2, \cdots , TK) = \text{Ⱨ}((\text{Ŧ}(T1;W), \text{Ŧ}(T2;W), \cdots , \text{Ŧ}(TK;W))).$$

Here, Ŧ(T_K;W) is the function representing a ConvNet with parameters W which operates on the short snippet T_K and produces class scores over all the classes. The segmenta consensus function G combines the outputs from multiple short snippets to obtain a consensus of class hypothesis among them. Based on this consensus, the prediction function H predicts the probability of each action class for the whole video. Here we choose the widely used Softmax function for Ⱨ.

Aggregate function Analysis

Our temporal segment network framework relies heavily on the consensus (aggregation) feature. The form of the consensus function G is very important in this temporal segment network context, since it should have a high modelling capability while also being differentiable or having subgradients. The ability to efficiently aggregate snippet-level prediction into video-level scores is referred to as high modelling power, and the differentiability enables our temporal segment network architecture to be easily optimised using backpropagation.

In this paper five aggregation functions are proposed and discussed in detail:

- Max Pooling
- Average Pooling
- Top-K Pooling
- Weighted average
- Attention weighting

TSN Training

Existing human annotated datasets for action recognition are limited in terms of sizes. In practice, training deep ConvNets on these datasets are prone to overfitting. To mitigate this issue, several strategies to improve the training in the temporal segment network framework.

Cross Modality Initialization

Regularisation - Batch Normalization

Data Augmentation

Cross Modality Initialization

Pre-training the network parameters on large-scale image recognition datasets, such as ImageNet, has turned out to be an effective remedy when the target dataset does not have enough training samples. As spatial networks take RGB images as inputs, it is natural to exploit models trained on the ImageNet as initialization.

For other input modalities such as optical flow and RGB difference,a cross modality initialization strategy was used.Specifically,first discretize optical flow fields into the interval of 0 to 255 by linear transformation. Then,average the weights of pretrained RGB models across the RGB channels in the first layer and replicate the mean by the channel number of temporal network input.

Finally, the weights of remaining layers of the temporal network are directly copied from the pretrained RGB networks.

Regularization - Batch Normalization

Batch Normalization is able to deal with the problem of covariate shift by estimating the activation mean and variance within each batch to normalize these activation values. This operation speeds up the convergence of training, but also increases the risk of over-fitting in the transfer learning process, due to the biased estimation of mean and variance from a limited number of training samples in target dataset.

Therefore, after initialization with pretrained models,the mean and variance parameters of all Batch Normalization layers except the first one was freezed. As the distribution of optical flow is different from the RGB images, the activation value of first convolution layer will have a distinct distribution and to re-estimate the mean and variance accordingly. This is called this strategy partial BN. Meanwhile, we add an extra dropout layer with high dropout ratio (set as 0.8 in experiment) after the global pooling layer to further reduce the effect of over-fitting.

Data Augmentation

In the original two-stream ConvNets, random cropping and horizontal flipping are employed to augment training samples.Two new data augmentation techniques were exploited: corner cropping and scale jittering. In the corner cropping technique, the extracted regions are only selected from the corners or the center of an image to avoid implicitly focusing more on the center area.

In the multi-scale cropping technique, we adapt the scale jittering technique used in ImageNet classification to action recognition. We present an efficient implementation of scale jittering.Fixed input size as 256 × 340, and the width and height of cropped regions are randomly selected from {256, 224, 192, 168}. Finally, these cropped regions will be resized to 224 × 224 for network training. In fact, this implementation not only contains scale jittering, but also involves aspect ratio jittering.

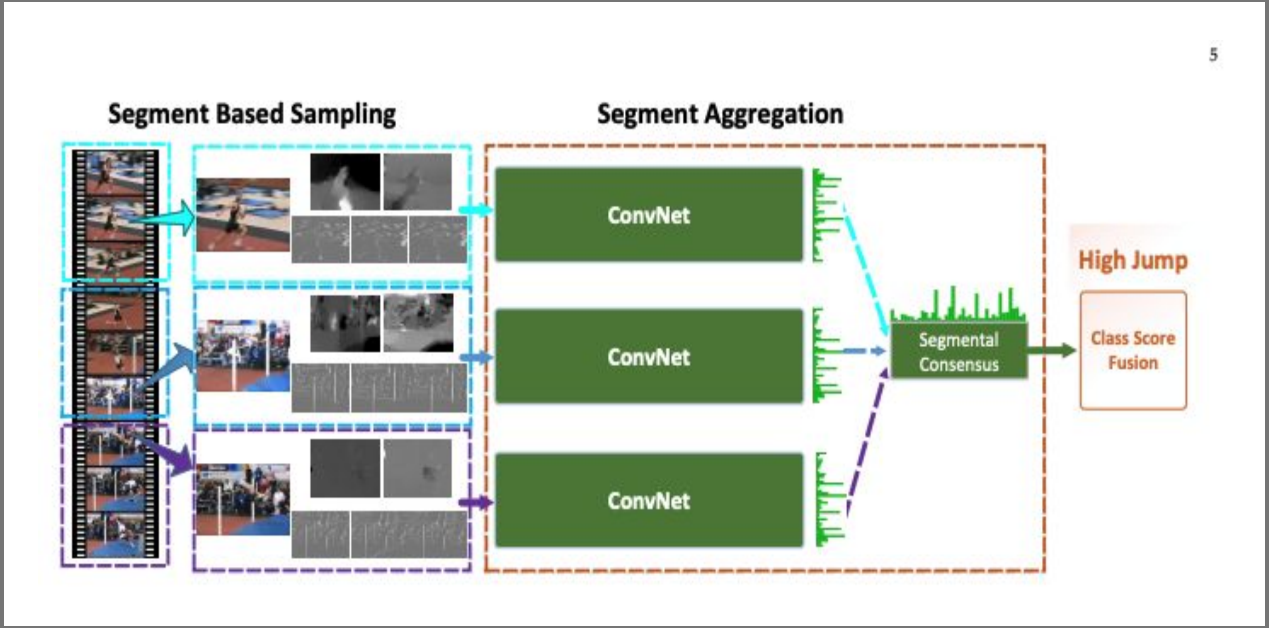
Aggregate function Analysis

Our temporal segment network framework relies heavily on the consensus (aggregation) feature. The form of the consensus function G is very important in this temporal segment network context, since it should have a high modelling capability while also being differentiable or having subgradients. The ability to efficiently aggregate snippet-level prediction into video-level scores is referred to as high modelling power, and the differentiability enables our temporal segment network architecture to be easily optimised using backpropagation.

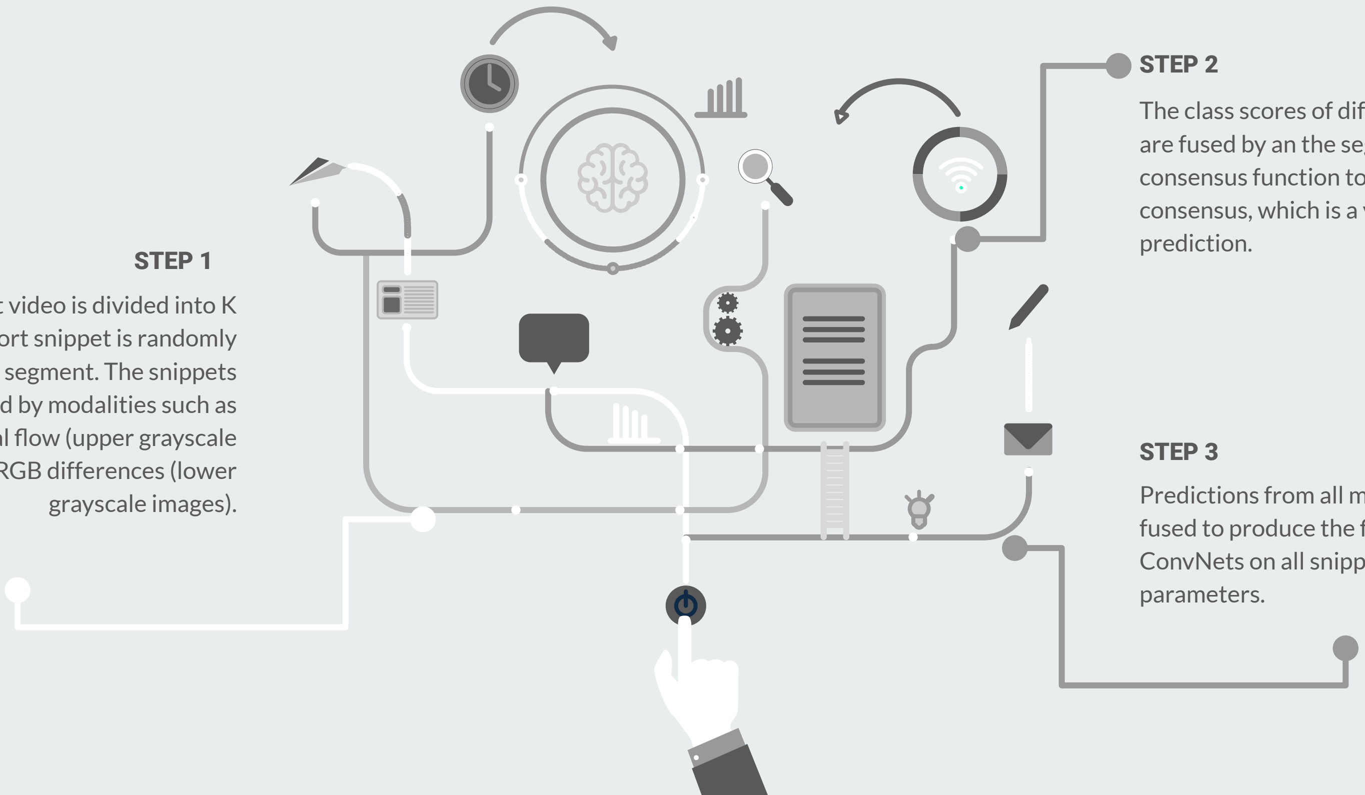
In this paper five aggregation functions are proposed and discussed in detail:

- Max Pooling
- Average Pooling
- Top-K Pooling
- Weighted average
- Attention weighting

Brief look of TSN



TSN STAGES



STEP 1

One input video is divided into K segments and a short snippet is randomly selected from each segment. The snippets are represented by modalities such as RGB frames, optical flow (upper grayscale images), and RGB differences (lower grayscale images).

STEP 2

The class scores of different snippets are fused by an the segmental consensus function to yield segmental consensus, which is a video-level prediction.

STEP 3

Predictions from all modalities are fused to produce the final prediction. ConvNets on all snippets share parameters.

Let’s dive into code !

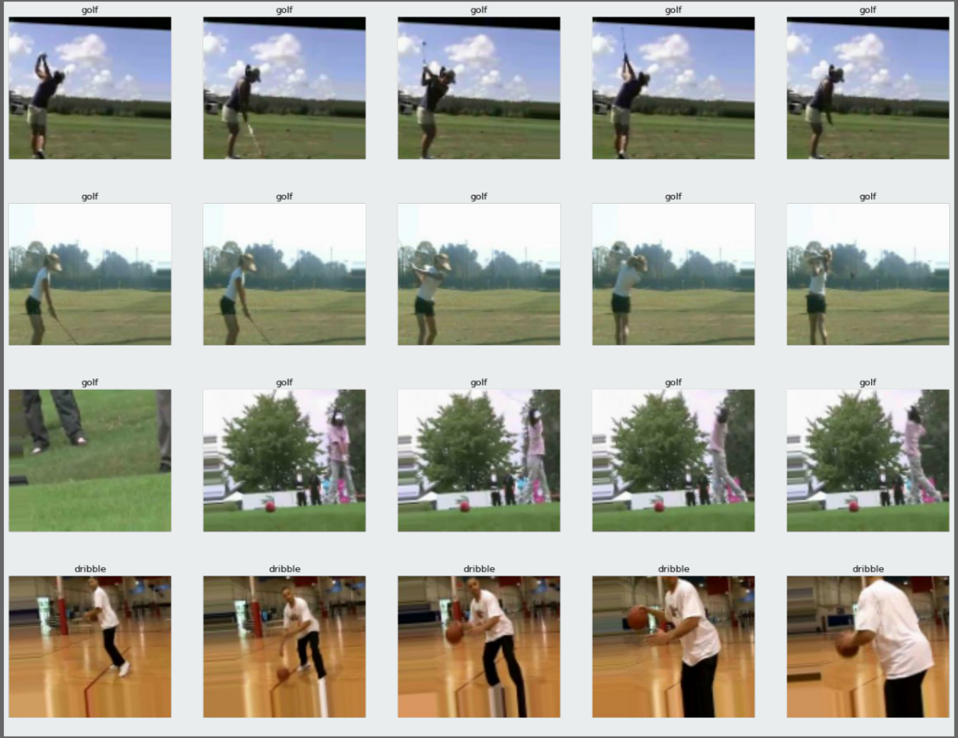
“I am telling you, the world’s first trillionaires are going to come from somebody who masters AI and all its derivatives,and applies it in ways we never thought of.”

Setting-Up time distribution

```
cap = cv.VideoCapture(video)
total_frames = self.count_frames(cap, video,
force_no_headers)
orig_total = total_frames
if total_frames % 2 != 0:
    total_frames += 1
frame_step = floor(total_frames/(nbframe -1))
```

Get the RGB frames

```
if frame_i == 1 or frame_i % frame_step == 0 or frame_i == orig_total:
# resize
frame = cv.resize(frame, shape)
# use RGB or Grayscale ?
if self.nb_channel == 3:
frame = cv.cvtColor(frame, cv.COLOR_BGR2RGB)
else:
frame = cv.cvtColor(frame, cv.COLOR_RGB2GRAY)
# to np
frame = img_to_array(frame) * self.rescale
# keep frame
frames.append(frame)
if len(frames) == nbframe:
break
```

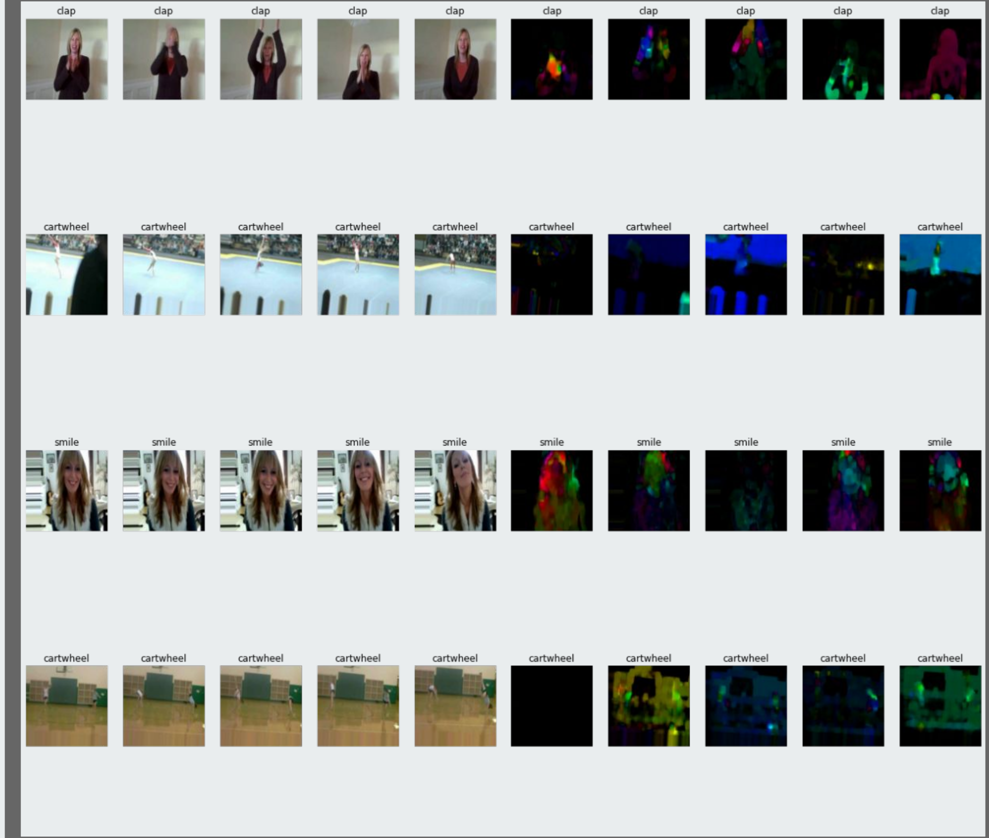


Sample RGB Dataset

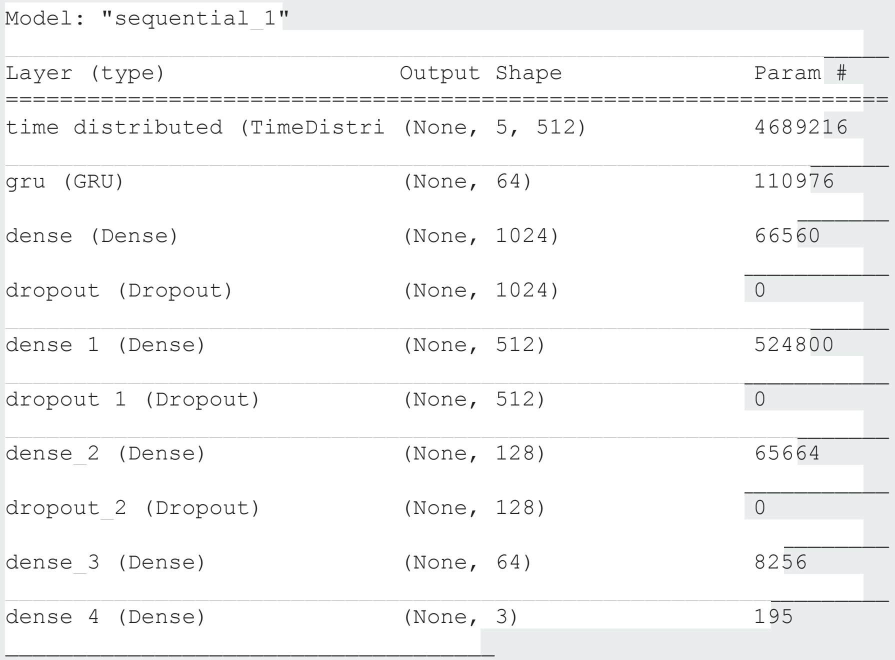
Get the Optical frames

```
 now = cv.cvtColor(frame1,cv.COLOR_BGR2GRAY)
           frame_i += 1
           if frame_i == 1 or frame_i % frame_step == 0 or frame_i == orig_total:
               flow = cv.calcOpticalFlowFarneback(prvs,now, None, 0.5, 3, 15, 3,
5, 1.2, 0)
               mag, ang = cv.cartToPolar(flow[...,0], flow[...,1])
               hsv[...,0] = ang*180/np.pi/2
               hsv[...,2] = cv.normalize(mag,None,0,255,cv.NORM_MINMAX)
               bgr = cv.cvtColor(hsv,cv.COLOR_HSV2BGR)
               k = cv.waitKey(30) & 0xff
               if k == 27:
                 Break
 # resize as done with RGB frames
           prvs=now
```

Sample Optical Dataset



Model



```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
time distributed (TimeDistri (None, 5, 512)            4689216
_________________________________________________________________
gru (GRU)                    (None, 64)                110976
_________________________________________________________________
dense (Dense)                (None, 1024)              66560
_________________________________________________________________
dropout (Dropout)            (None, 1024)              0
_________________________________________________________________
dense 1 (Dense)              (None, 512)               524800
_________________________________________________________________
dropout 1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 128)               65664
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_3 (Dense)              (None, 64)                8256
_________________________________________________________________
dense 4 (Dense)              (None, 3)                 195
____________________________________
```

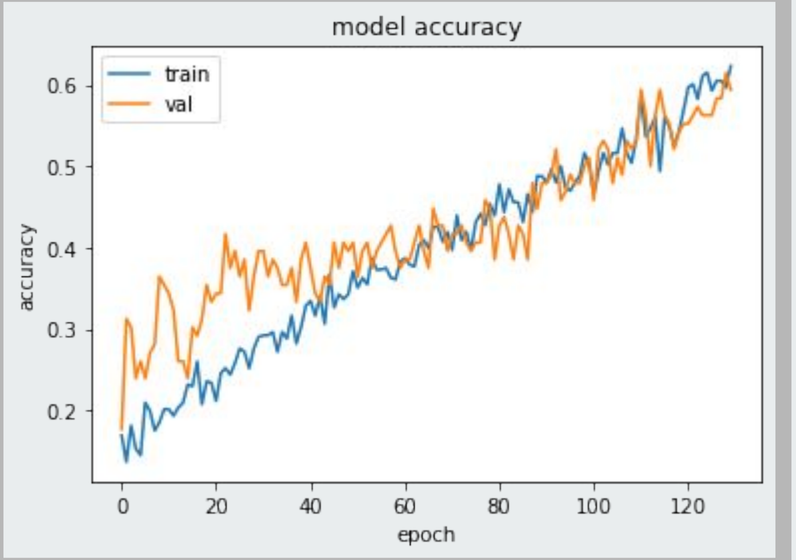
Allotted Tasks

- Use both RGB and Optical Flow for data extraction
- Create loss plots and accuracy plots
- Test the trained model

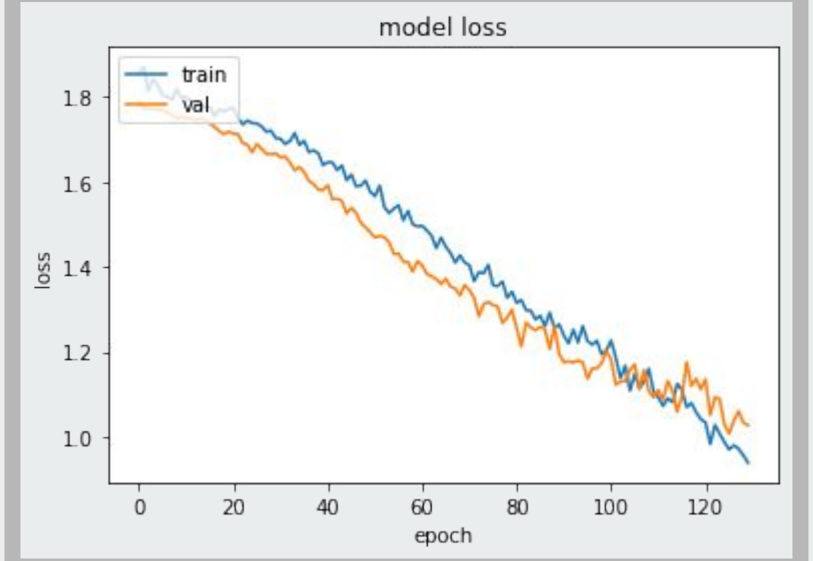
Results

“In the long term, artificial intelligence and automation are going to be taking over so much of what gives humans a feeling of purpose.”

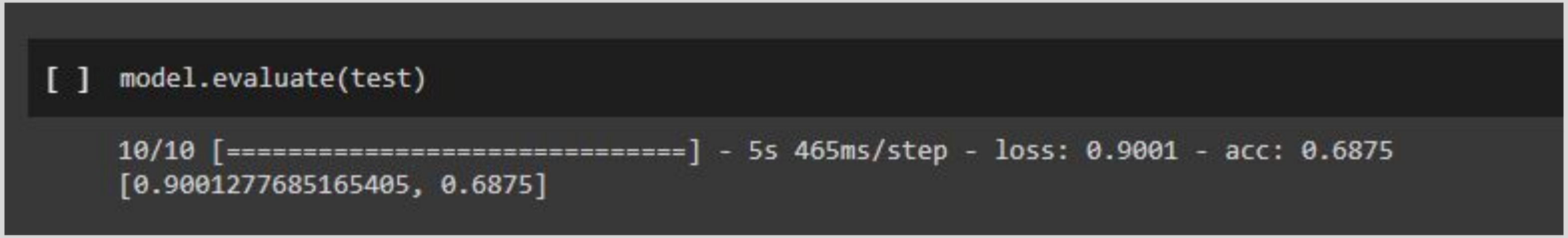
Training Accuracy



Training Loss



Accuracy on Test Set



```
[ ]  model.evaluate(test)

     10/10 [==============================] - 5s 465ms/step - loss: 0.9001 - acc: 0.6875
     [0.9001277685165405, 0.6875]
```

68.75 %

Comparing Results

	Results from Paper	Our Results
Number of Classes	51	6
Accuracy	71%	68%

Obstacles faced

“Obstacles are those frightful things you see when you take your eyes off your goal.”

Accessing data from Google drive is not possible above a certain frequency, and the storage on remote collab machine was limited, so large models cannot be trained.

Data availability was an issue sometimes, for example we were unable to find UCF101 dataset as the link was revoked.

Updated libraries keep coming up, and then implementation of a mere 3 year old paper becomes difficult as the dependencies are not solvable in the environment due to lack of support.

Future Scope

An interesting vertical to study is how depth modelling can relate to better video classifications. Current approaches to video classification have to learn that the videos are taken in a 3D environment. Depth forms an important part of our spatial perception. It could be that current approaches have to learn how to express depth in their spatiotemporal modelling of 2D features.

Perhaps using monocular depth estimation networks can aid the current video networks in creating a better understanding of the environment itself. An important observation is that any spatial changes in a video come from two sources: a transformation of an external object we are observing, or the observer itself changing viewing angle or position.

Both these sources of movement have to be learned by the current networks. It would be interesting to investigate how depth fields could be used to model either sources of change.

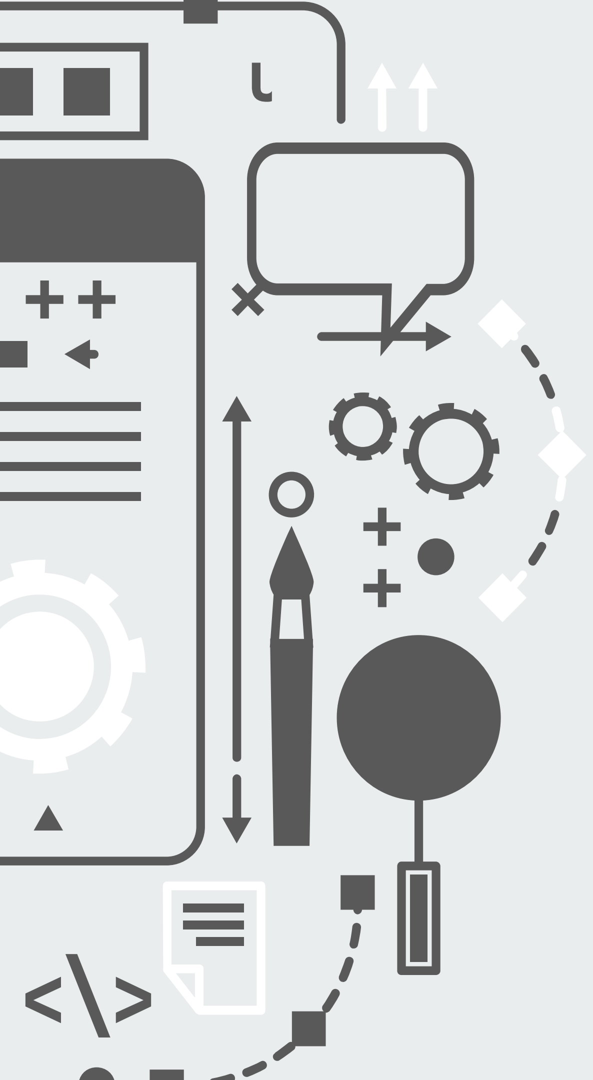
Learning Outcomes

We learned to implement a temporal segment network that can be used to improve various action recognition models.

Understanding dataset and how meaningful data preprocessing can have a huge impact on the results obtained.

Gained technical know-how, through hands on experience, to implement cutting edge technologies in the field of deep learning.

- Data Sampling
- Data augmentation
- Action Recognition Models
- Aggregation Functions
- Regularization



THANK YOU!