**Event Based Video Analytics**

**By**

**SHARAN CHHUGANI - 18BCE354**

**VIDITA CHUDASAMA - 18BCE355**

******

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**Ahmedabad 382481**

**Event Based Video Analytics**

**Mini Project – III**

Submitted in partial fulfillment of the requirements

For the degree of

**Bachelor of Technology in Information Technology**

By

**SHARAN CHHUGANI - 18BCE354**

**VIDITA CHUDASAMA - 18BCE355**

Guided By

**PROF. VISHAL U PARIKH**

**[DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING]**

******

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**Ahmedabad 382481**

**CERTIFICATE**

This is to certify that the Mini Project -III entitled “Event Based Video Analytics” submitted by Sharan Chhugani (18BCE354) and Vidita Chudasama (18BCE355), towards the partial fulfillment of the requirements for the degree of Bachelor of Technology inInformation Technology/Computer Engineering of Nirma University is the record of work carried out by him/her under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination.

Prof. Vishal U Parikh Dr. Madhuri Bhavsar

Assistant Professor Professor and Head of Department

Department of Computer Science & Engg, Dept. of Computer Science & Engineering,

Institute of Technology, Institute of Technology,

Nirma University, Nirma University,

Ahmedabad Ahmedabad

**ACKNOWLEDGEMENT**

We would like to express our deepest appreciation to all those who provided us the possibility to complete this report. We acknowledge with thanks, the support rendered by Prof. Vishal U Parikh, under whose aegis we were able to complete the task in a given period of time. This report could not be completed without coordination of the team members Sharan Chhugani and Vidita Chudasama. We also appreciate the constructive suggestions given by our friends to further enhance the content of the report.

At the home front, we are extremely grateful to our family members for the support and encouragement we got from them in successfully completing the report.

**ABSTRACT**

The Goal of this project is to understand the concept of Human Activity Recognition and its application while trying to implement it. The initial process was to get acquainted with the technology used to implement Video Analysis such as CNN,YOLO modal, LSTM,RNN and various other aspects. Later we went on to train the modal with various video samples and make the algorithm to classify the activities happening in the video in 400 different classes.

**TABLE OF CONTENTS**

Certificate

Acknowledgement

Abstract

Table of Contents

**List of figures**

Figure 1.1 Detection on subsequent layers of the Neural Network

Figure 1.2 Example of Vertical Edge detection on a grayscale Image.

Figure 1.3 Example of padding where p=2.

Figure 1.4 Example of strided convolution where value of stride s=2.

Figure 1.5 Example of a convolution layer (one layer).

Figure 1.6 Example of Max Pooling.

Figure 1.7 Example of Average Pooling.

Figure 1.8 Example of Fully Connected Layer.

Figure 1.9 Formula of Softmax Function.

Figure 1.10 Formula of Logistic Function.

Figure 1.11 Example of a Convnet.

Figure 2.1 Visual representation of RNN.

Figure 2.2 A LSTM neural network.

Figure 3.1 Hierarchy of project.

Figure 3.2 Code Snippet - 1.

Figure 3.3 Code Snippet - 2.

Figure 3.4 Code Snippet - 3.

Figure 3.5 Code Snippet - 4.

Figure 3.6 Code Snippet - 5.

Figure 3.7 Code Snippet - 6.

Figure 3.8 Code Snippet - 7.

Figure 3.9 Code Snippet - 8.

Figure 3.10 Code Snippet - 9.

Figure 3.11 Code Snippet - 10.

Figure 3.12 Code Snippet - 11.

Figure 3.13 Code Snippet - 12.

Figure 3.14 Code Snippet - 13.

Figure 3.15 Predicted output video preview - 1.

Figure 3.16 Predicted output video preview - 2.

Figure 3.17 Predicted output video preview - 3.

**Chapter 1 Introduction**

1.1 Introduction

1.2 CNN and its Terminologies

1.3 Types of Layers

1.4 Going Deeper into Networks

**Chapter 2 Working on Videos**

* 1. RNN
  2. LSTM

**Chapter 3 Project Explanation**

3.1 Explanation and Working

3.2 Output and Analysis

**Chapter 4 Summary and Conclusion**

4.1 Conclusion

**References**

**Appendix – A List of Useful Websites**

**1. INTRODUCTION**

**1.1 Introduction**

Computer Vision has evolved at a rapid rate nowadays which can solve the problems which were impossible to solve a few years ago.

The Computer Vision Problems includes :

-> Image Classification - By Image Classification we mean to find whether the image is of a specified object or not. For Example, when an Image is given as an Input, It would tell us whether the provided input is of ‘DOG’ or not, considering Dog as our subject.

-> Object Detection - Object Detection can be considered as a higher version of Image Classification because in it, we not only have to find the object but also have to locate the object in the Image and draw a box over it depicting the identified object.

The problem with the Computer Vision applications is the input image. As the size of the input image increases the size of our feature vector x[i] also increases. Let say we have an input image of 100x100 which is a very low resolution coloured image, then the value of our feature vector x[i] will be 100x100x3 = 30,000 for just a low resolution image,then what about high resolution images…..

**1.2 CNN and its Terminologies**

To overcome the problem of the size of the feature vector we will use Convolution Layer. Lets see how…

So as you can see in the below image, Our Network will first detect Edges (such as horizontal, vertical, curve etc) in the Initial Layers, Thereafter in the Intermediate Layers it would detect the facial components such as eyes, nose etc , and in the Later Layers it would detect the face. Hence by this we can detect whether an object is present in the image or not.

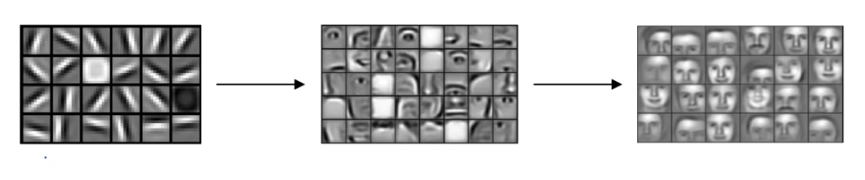


Figure 1.1 Detection on subsequent layers of the Neural Network

So for Edge detection in the input image we would take various filters (also known as kernel). A filter is a small image (Array of values) which consists of the simple edges which can be considered as the building blocks of an object in an image. Now this filter is convoluted along the rows of the input image one by one. We can also convolve more than 1 filter in one go but the input image’s dimension and the filter’s dimension should be the same.

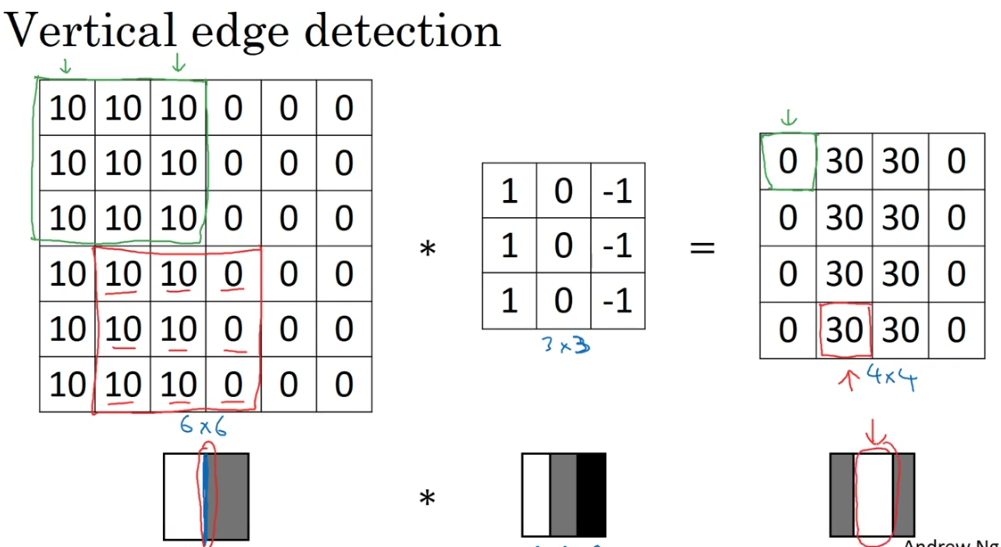


Figure 1.2 Example of Vertical Edge detection on a grayscale Image.

The above image shows the example of the convolution of a 3x3x1 filter on a 6x6x1 image which would result in a 4x4x1 image.

-> Output size :

n x n \* f x f ⇒ n-f+1 x n-f+1

The major problem with it is that what should be the value of the filter and what should be the dimension of the filter by which we want to convolute also how many filters should be used , which filters should be used…

Thus, Instead of trying different values, we can let the neural network learn those filters by itself, by starting off with random values and then back propagating the error. This turns out to be more robust. But learning takes time and also a high end device is needed.

When we apply filters for convolution, the dimension of the original image shrinks which leads to loss of information. Also when the filter convolutes through the input image, the pixels situated in the corners are used much less in the output as compared to the ones in the middle. Thus for the solution of such problems padding is used.

-> Output size :

Padding of p pixels on [n x n] \* f x f ⇒ (n+2p-f+1) x (n+2p-f+1)

-> Types : Depending upon padding we have 2 types of convolutions…

Valid - No padding

Same - Pad so that the output size is same as the input size

For output to be same as input : n+2p-f+1 = n

Thus, p=(f-1)/2

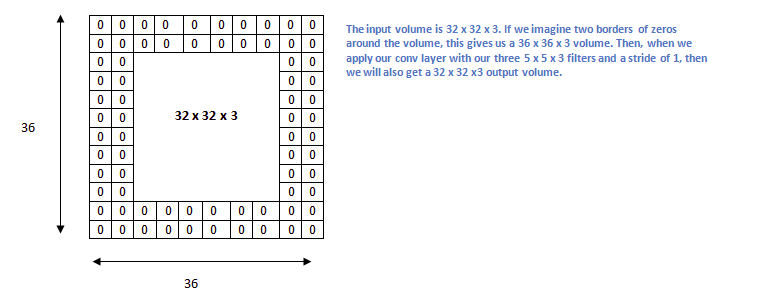


Figure 1.3 Example of padding where p=2.

Stride is the number of pixels shifted while convoluting. Despite of sliding 1 pixel we can slide by s pixels, which would give the output size as follows :

-> Output size :

Padding of p[n x n] \* stride of s[f x f] ⇒ [(n+2p-f)/s +1] x [(n+2p-f)/s +1]

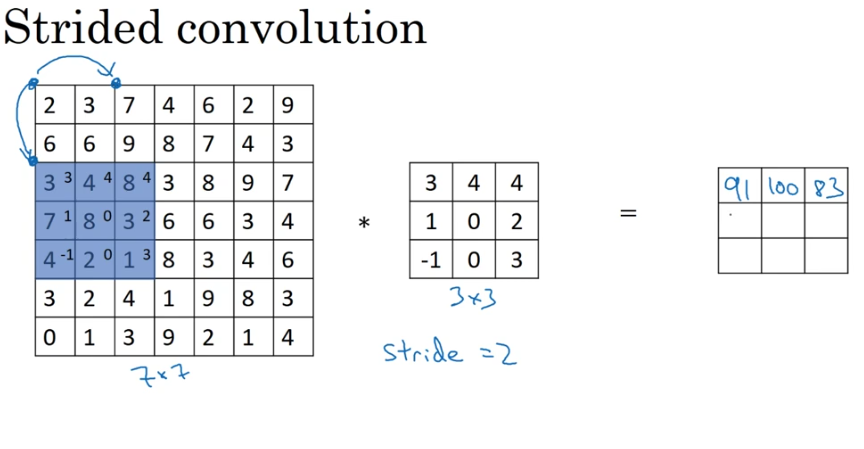


Figure 1.4 Example of strided convolution where value of stride s=2.

After applying the filter the bias value is added and the output is passed through the non-linear function RELU and thus the final output is obtained.

**1.3 Types of Layers**

The below are the layers which can be observed in a Convnet.

-> Convolutional Layer (CONV) :

As we saw earlier, In Convolutional Layer we convolve the Filter on Input image and add bias value and pass through the non-linear Function to get the output.

Here we need to learn the values of filter pixels and the bias values. They are known as parameters. For this, we initially provide random values for filters and biases and then backpropagate for several epochs to get the correct parameter values. These parameter values are then stored in a file to use it when feeding the input to the network for the prediction.

For Example:

If we have 1 layer having input image dim 6x6x3 and the filter size is 3x3x3 having 10 filters, the total parameters to be learnt for that layer would be - 3x3x3x10 + 10 = 280 parameters and total multiplications for obtaining these parameters would be - 108x160 = 17,280 for such a small resolution image and for just one layer.

Why do we use Convolutional Layer :

-- Parameter sharing: A feature detector (such as a vertical edge detector) that's useful in one part of the image is probably useful in another part of the image.

-- Sparsity of connections: In each layer, each output value depends only on a small number of inputs.

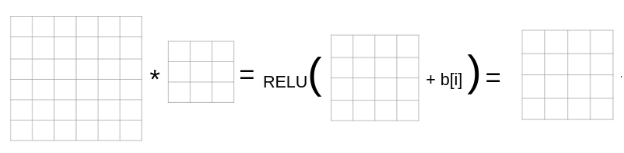


Figure 1.5 Example of a convolution layer (one layer).

-> Pooling Layer :

Why do we use Pooling layer :

-- It detects features more robustly.

-- makes the computational part easier.

-- To make the network learn faster.

-- There are no parameters to learn.

Types of Pooling Layer :

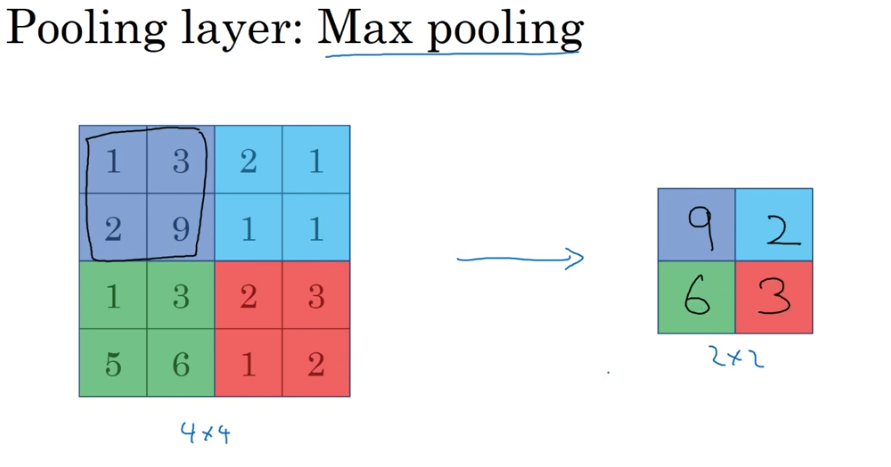


Figure 1.6 Example of Max Pooling.

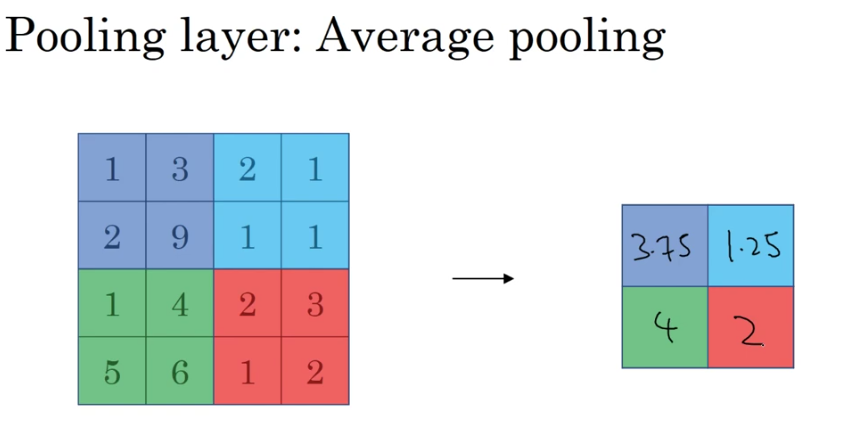


Figure 1.7 Example of Average Pooling.

-> Fully Connected Layer (FC) :

In Fully Connected Layers, all the inputs from one layer are connected to every activation unit of the next layer in the Neural network.

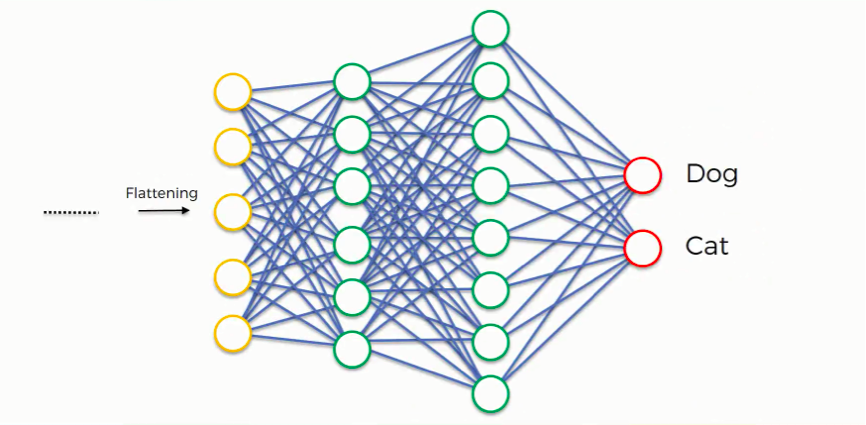


Figure 1.8 Example of Fully Connected Layer.

-> Softmax :

After Fully Connected Layer, Softmax Layer comes through which we get the output probabilities of different classes. The softmax function calculates the probability of each class which when added gives 1.

Formulae :

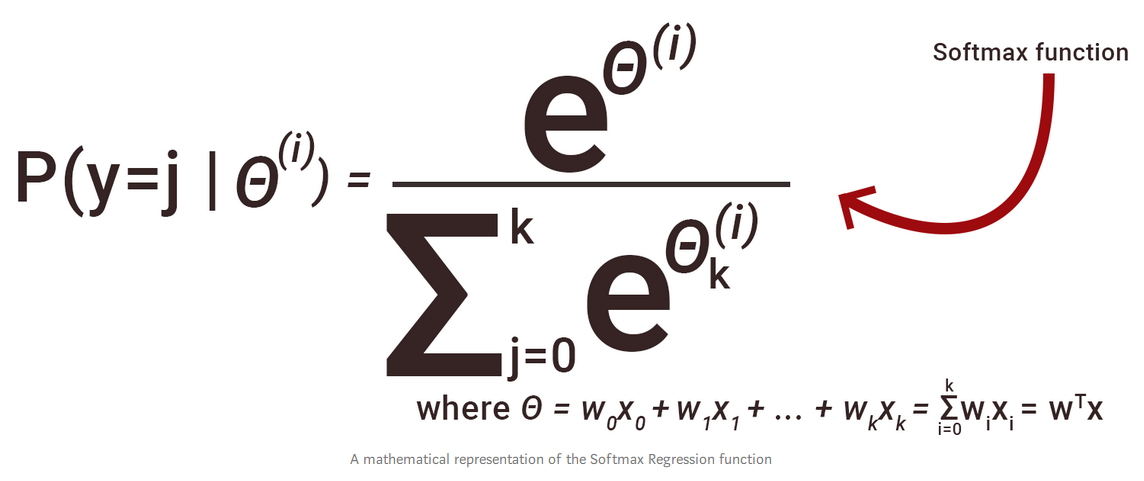


Figure 1.9 Formula of Softmax Function.

-> Logistic :

We can also use Logistic Function instead of Softmax. Logistic function gives the probability of the particular class whether the object of that class is present or not. Using Logistic, the addition of the probabilities of all the classes does not add up to 1, because it calculates the probability of individual class.

Formulae :



Figure 1.10 Formula of Logistic Function.

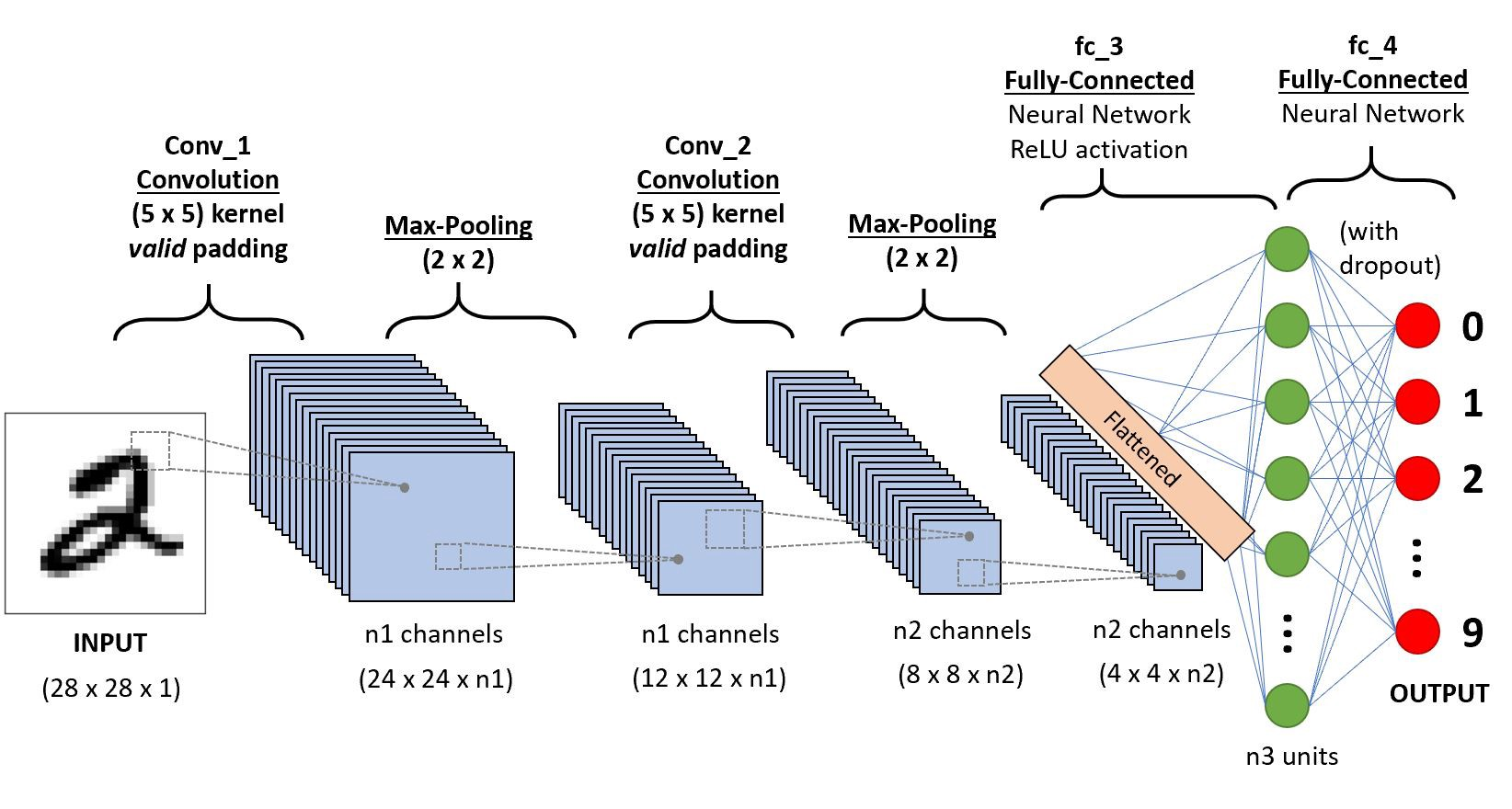


Figure 1.11 Example of a Convnet.

**1.4 Going Deeper into Networks**

A few concepts to be known for understanding the deep neural network.

-> Residual Block :

It is only understood that the deeper and larger the network the higher accuracy it shall achieve. But that only happens in the ideal scenarios. In the practical world, due to vanishing gradient, the accuracy of the network decreases beyond a certain number or layers and the network even becomes incapable of processing the simplest identity function. To solve this problem, residual blocks are constructed which are nothing but giving the output of one layer directly to the layer two or more layers ahead by skipping the ones in between. Residual blocks combine to form a ResNet.

-> Inception module :

The Formation of the inception module was done to reduce the computation of a deep network and also to solve the problem of overfitting. The idea is to make the block wider instead of deeper. Instead of just stacking the filters of various one above the other , all the kernels are kept at the same level immediately getting the input from the last layer. To make computation easier ,the out of there filters are then stacked and sent to the next block. However a 1\*1 filet is introduced after the pooling layer.

-> Data Augmentation :

The aim of Data Augmentation is to deal with the problem of less data. Data Augmentation allows us to make more samples from the existing ones. Many techniques for that exist such as padding , copping , flipping , rotating etc.

-> Object Localisation :

The main difference between object localisation and classification is that object classification tries to identify the single most visible object in the image while the localisation also tries to segment out the area of the image where the object resides by creating a bounding box around it.

-> Sliding Windows :

The sliding window is a window created prior to the convolution and it is convoluted over the image, forming the same size of rectangles and trying to find out if the particular object is present in that area or not.

-> IOU :

Intersection over Union as its name suggests is a technique in which the unionand the intersection of two bounding boxes are calculated and if the division is greater than or equal to 0.5, the object is said to be localised correctly.

-> Anchor Boxes :

Anchor boxes are the borders drawn as an outline on the object which is detected as an output.

**2. WORKING ON VIDEOS**

**2.1 RNN**

RNN stands for Recurrent Neural Network. In RNN the connections between the nodes form a “directed graph” along the temporal data sequence.As the name suggests, in RNN it does not backpropagate ,only once the feedforward is done and the neurons recurrently calculate the weights and update it there only. This can lead towards the dynamic temporal behaviour which can be used in the temporal data such as activity detection. Also the RNN can recurrently pass the information to the Neuron so that we don't need 5 neurons for five inputs. We just need 1 neuron for the input and the parameter list can be variable.

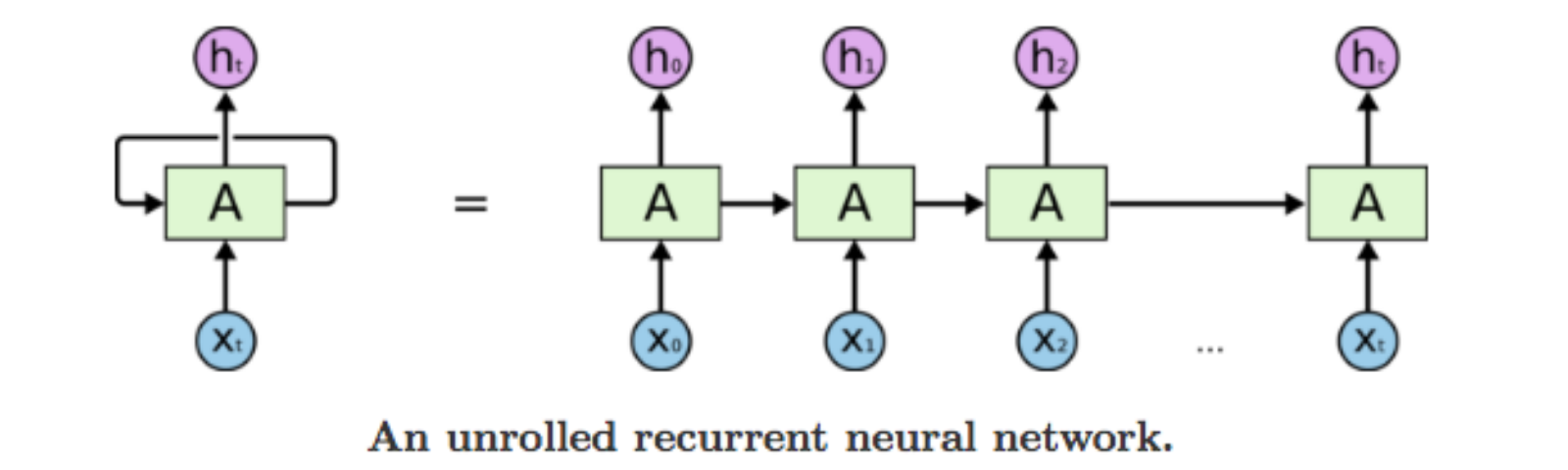


Figure 2.1 Visual representation of RNN.

The other feature of RNN is parameter sharing. Here as we saw we can have an array of parameter inputs in the same neuron, the weight assigned with the neuron will be the same, which means that all the inputs to that neuron will share the weight. Due to the parameter sharing feature, we can get rid of size limitations.

**2.2 LSTM**

The need of the situation is to remember the context of the module for as far as possible and in theory, it is the ideal condition for RNN's to work. However in practise the RNN made by using a simple tanh layer in the recurung module can only remember the context to a certain length. To solve this problem LSTM i.e Long Short Term Memory ) was introduced which is the reason for major advancements like speech recognition etc.

All recurrent neural networks have the form of a chain of repeating modules of neural networks. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer.

LSTMs also have this chain-like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way.

Combining a single sigmoid convolutional layer and the pointwise multiplication operation, gates are introduced in LSTM which allow or restrict information from entering the module. They also enable LSTM to add or delete information in each cell.

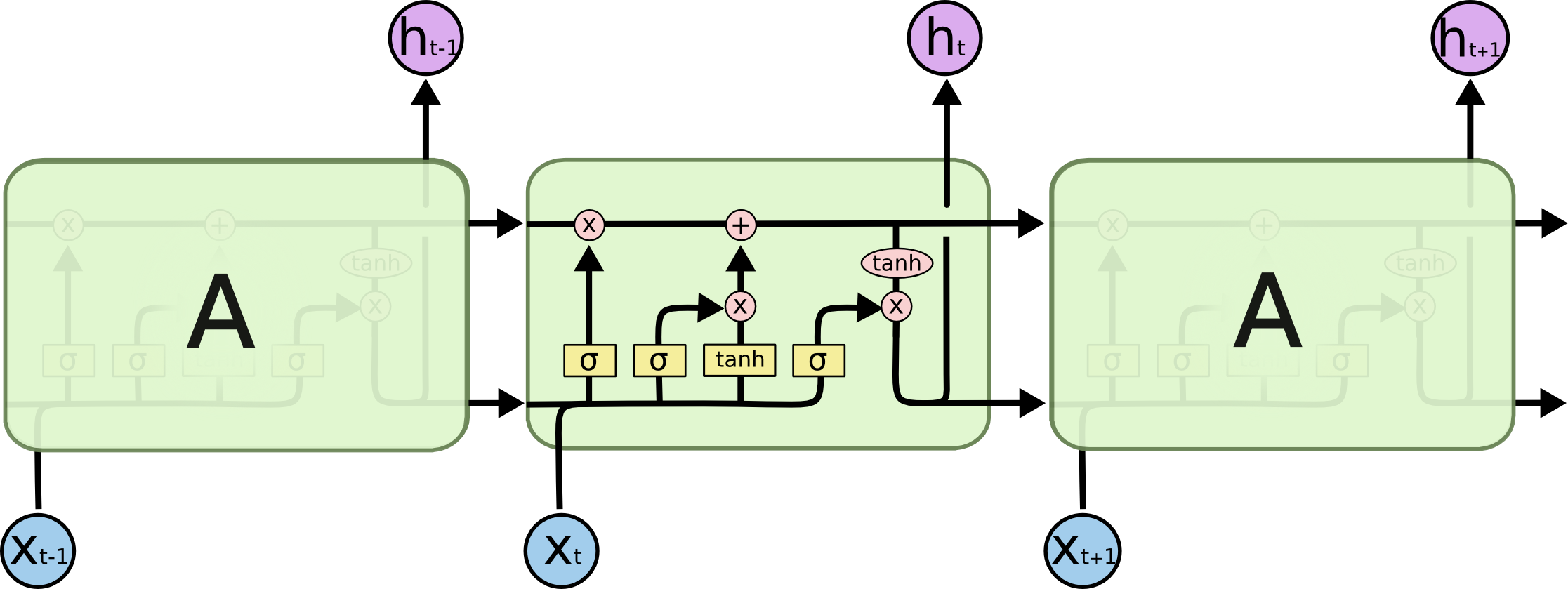


Figure 2.2 A LSTM neural network.

**3. PROJECT EXPLANATION**

**3.1 Explanation and working**

So, the project is about predicting the activity occurring in the video and then writing the predicted action on the video frames.

The Neural Network used here is trained on kinetics dataset. This dataset consists of 400 classes and we have at least 400 video clips per class, A total number of 300,000 videos.

To understand the modifications and other details of the network do refer this paper - <https://arxiv.org/abs/1711.09577>

Moving Ahead, so the hierarchy of the project is as follows :

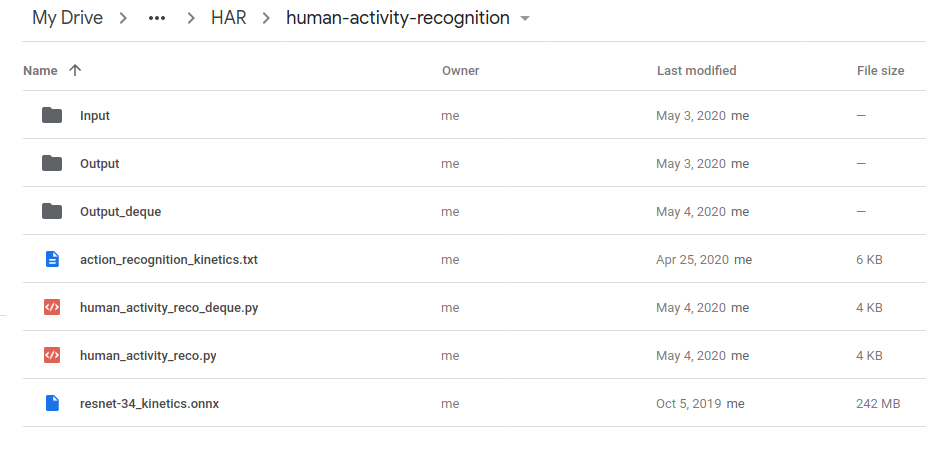


Figure 3.1 Hierarchy of project.

Here, action\_recognition\_kinetics.txt contains the names of the 400 classes in which the activities can be classified.

Input folder contains the videos which can be given as input to the neural network for the processing.

Output folder contains the videos given as an output by the file Human\_activity\_reco.py and Output\_deque contains the output videos generated by human\_activity\_reco.py.

Resnet-34\_kinetics.onnx file contains our neural network.

Ok, So there are 2 files namely human\_activity\_reco.py and human\_activity\_reco\_deque.py. The first file processes frames normally and in the second file we use Queue data structure to store frames. Let's see how...

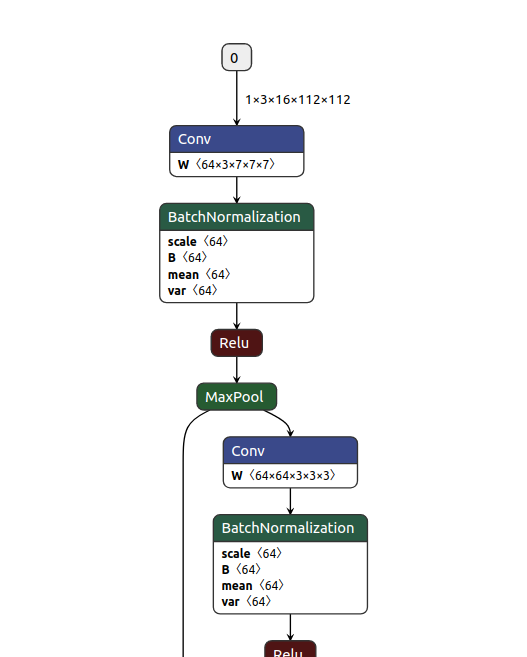


Figure 3.1 Hierarchy of project.

-> Human\_activity\_reco.py :

So firstly, the libraries are imported and some arguments are taken as input.

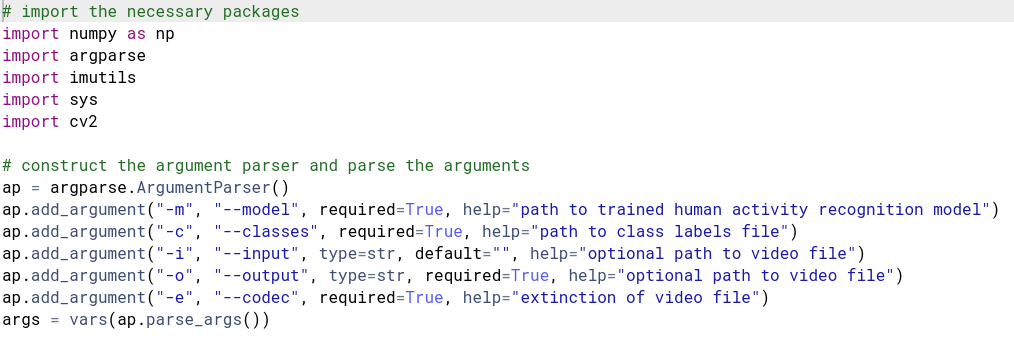


Figure 3.2 Code Snippet - 1.

Thereafter we will store the classes, taken as input, in a variable and initialize needed variables. The sample duration is 16 and the sample size is 112 because our network needs that (refer figure 3.2).

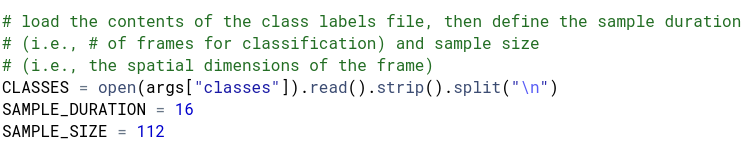


Figure 3.3 Code Snippet - 2.

Thereafter we will load the neural network and initialize a pointer to the video stream.

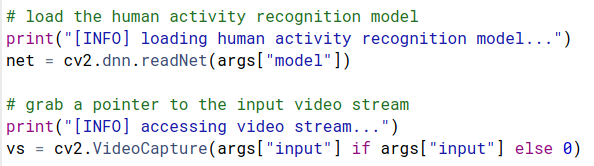


Figure 3.4 Code Snippet - 3.

Thereafter the fps of the input video is found and stored in variable fps and the writer is initialized. Through the writer we will write the processed frames to the output video in our drive.

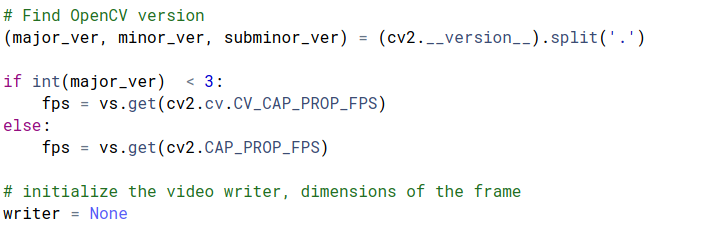


Figure 3.5 Code Snippet - 4.

Now in the while loop we will grab frames from the input video until it's over. Now in it we will take a for loop grabbing 16 frames as the value of sample duration is 16. And then will resize these frames and store those 16 frames into frames[] variable.

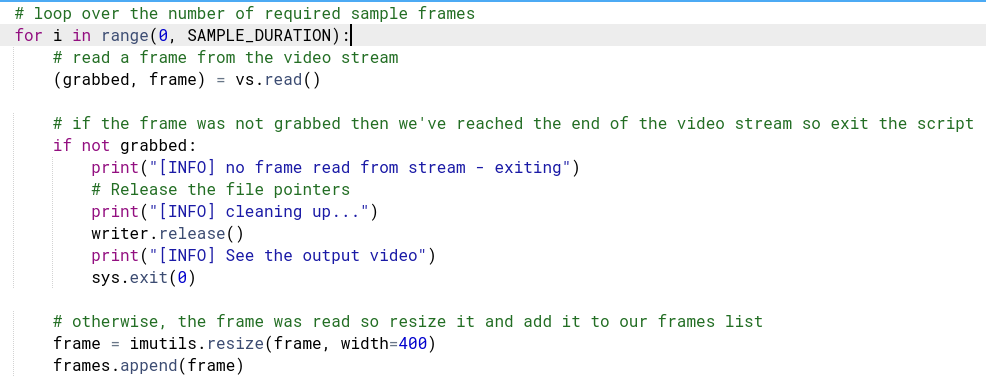


Figure 3.6 Code Snippet - 5.

Thereafter, by using frames variable we will make a blob and pass it to the model and take the Action as output.

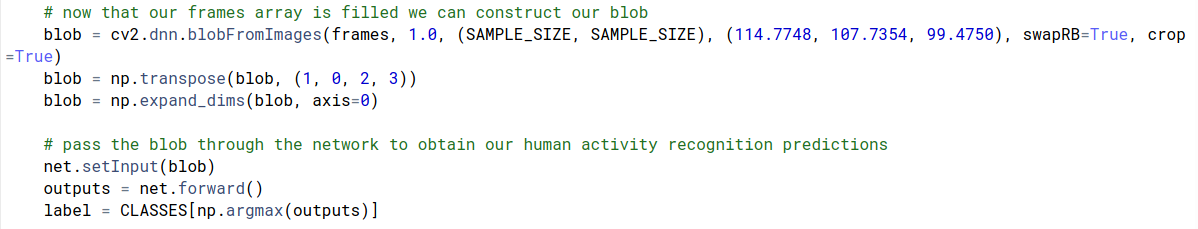


Figure 3.7 Code Snippet - 6.

Then again a for loop is taken to write the predictions on the frames and append those frames to a video.

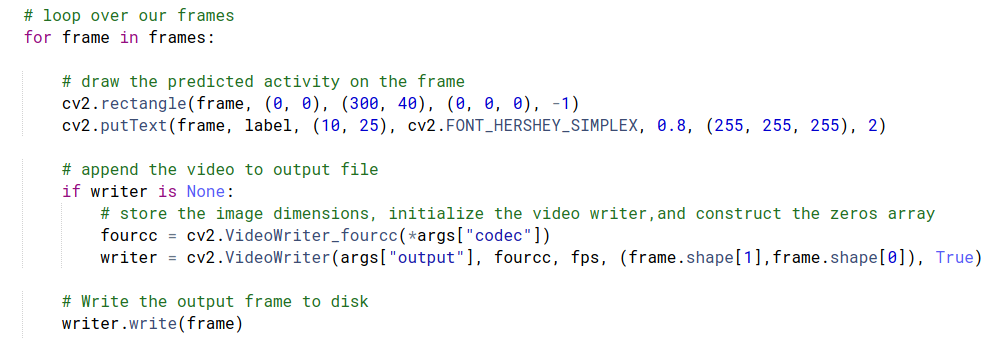


Figure 3.8 Code Snippet - 7.

Now our output video is getting ready. The while loop will continue until the frames in our video is over. And we can view the output video in the drive folder itself.

-> Human\_activity\_reco\_deque.py :

In this file we will use Queue data Structure. The queue size would be 16 as our sample\_duration value is 16.

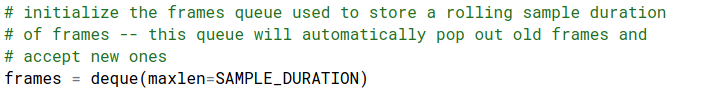


Figure 3.9 Code Snippet - 8.

Else everything is the same, the while loop is taken and frames are grabbed, the below condition is for the starting 15 times when the frames are being filled in our loop.

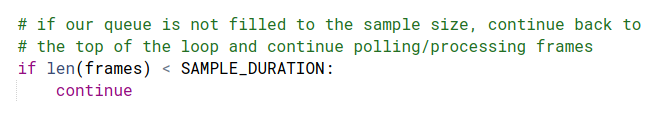


Figure 3.10 Code Snippet - 9.

When the 16th frame arrives the code goes further as the queue is filled with 16 frames and then the blob is made and the same is passed through the network to get the prediction.

And hence the prediction is written on the frame and is then appended to the output video.

When the next frame arrives, the first frame is dropped and the new frame is appended in the queue.

So here as you can see, the same frame is taken for the prediction multiple times. You can think of a window sliding over the input video frames for the prediction.

**3.2 Output and Analysis**

Now as we know what all the files contain, we will upload this folder in the drive so that we can easily access this folder through our colab file.

Firstly we will mount the drive and then reach the location of the folder which we have uploaded.

The corresponding code output is shown in the screenshot itself.

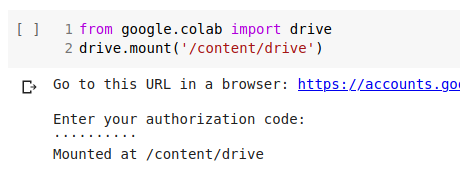


Figure 3.11 Code Snippet - 10.

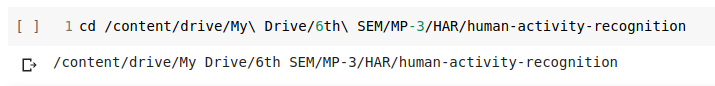


Figure 3.12 Code Snippet - 11.

Now we will pass the required parameters to the python file explained previously and see the output.

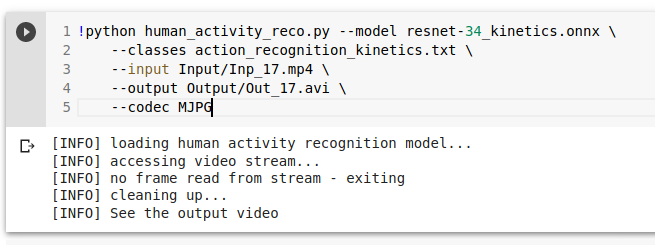


Figure 3.13 Code Snippet - 12.

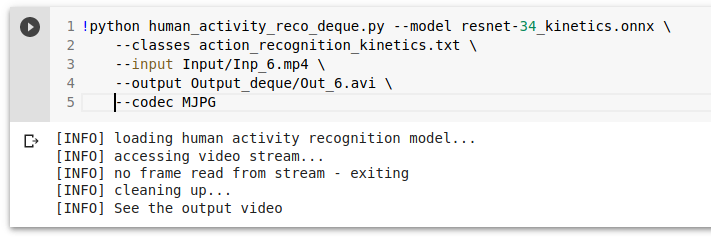


Figure 3.14 Code Snippet - 13.

The output video will be found in the output folder…

Below are some screenshots of the predicted activities in the videos.



Figure 3.15 Predicted output video preview - 1.

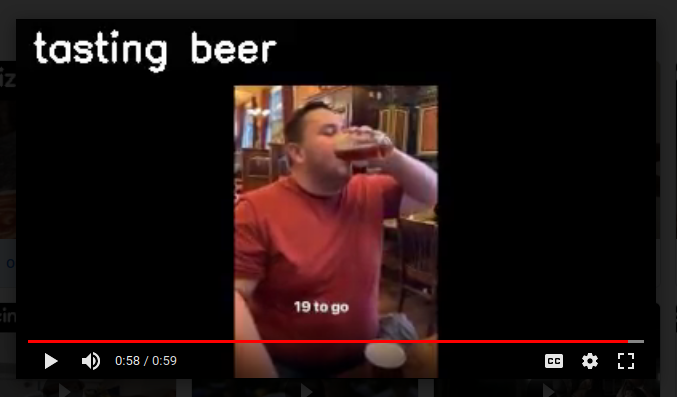


Figure 3.16 Predicted output video preview - 2.



Figure 3.17 Predicted output video preview - 3.

The Neural network works but gives only 75% accuracy as per the analysis in the paper. So it needs more videos of the same class to be fed so that it can learn more, and we can have higher accuracy on our applications. As this problem statement works with the generic activity detection, this accuracy problem will always be there because we need more training videos for all specific activities to achieve higher accuracy.

**4. SUMMARY AND CONCLUSION**

**4.1 Summary**

To summarize the entire work, we studied the concepts of object detection and localisation using CNN and also how to process them in videos using LSTM and RNN. We used the above explained algorithm and it is possible to detect activities happening in the video.

**4.2 Conclusion**

To conclude the entire project, we would like to add that the project is good to work for numerous applications. Also the one glitch that still exists is that the switch between the detection is quite high and that is possibly because the algorithm gets confused when the activity is changing. To rectify this will be the future scope of this project.

**References**

[1] R.M. Bell and Y. Koren. Lessons from the netflix prize challenge.ACM SIGKDD Explorations Newsletter,9(2):75–79, 2007.

[2] A. Berg, J. Deng, and L. Fei-Fei.Large scale visual recognition challenge 2010. www.image-net.org/challenges. 2010.

[3] L. Breiman. Random forests.Machine learning, 45(1):5–32, 2001.

[4] D. Cire ̧san, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification.Arxiv preprint arXiv:1202.2745, 2012.

[5] D.C. Cire ̧san, U. Meier, J. Masci, L.M. Gambardella, and J. Schmidhuber. High-performance neuralnetworks for visual object classification.Arxiv preprint arXiv:1102.0183, 2011.

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale HierarchicalImage Database. InCVPR09, 2009.

[7] J. Deng, A. Berg, S. Satheesh, H. Su, A. Khosla, and L. Fei-Fei.ILSVRC-2012,2012.URL<http://www.image-net.org/challenges/LSVRC/2012/>.

[8] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: Anincremental bayesian approach tested on 101 object categories.Computer Vision and Image Understand-ing, 106(1):59–70, 2007.

[9] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical Report 7694, Cali-fornia Institute of Technology, 2007. URL<http://authors.library.caltech.edu/7694>.

[10] G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R.R. Salakhutdinov. Improving neural net-works by preventing co-adaptation of feature detectors.arXiv preprint arXiv:1207.0580, 2012.

[11] K. Jarrett, K. Kavukcuoglu, M. A. Ranzato, and Y. LeCun. What is the best multi-stage architecture forobject recognition? InInternational Conference on Computer Vision, pages 2146–2153. IEEE, 2009.

[12] A. Krizhevsky. Learning multiple layers of features from tiny images. Master’s thesis, Department ofComputer Science, University of Toronto, 2009.

[13] A. Krizhevsky. Convolutional deep belief networks on cifar-10.Unpublished manuscript, 2010.

[14] A. Krizhevsky and G.E. Hinton. Using very deep autoencoders for content-based image retrieval. InESANN, 2011.

[15] Y. Le Cun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, L.D. Jackel, et al. Hand-written digit recognition with a back-propagation network. InAdvances in neural information processingsystems, 1990.

[16] Y. LeCun, F.J. Huang, and L. Bottou. Learning methods for generic object recognition with invariance topose and lighting. InComputer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the2004 IEEE Computer Society Conference on, volume 2, pages II–97. IEEE, 2004.

[17] Y. LeCun, K. Kavukcuoglu, and C. Farabet. Convolutional networks and applications in vision. InCircuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on, pages 253–256.IEEE, 2010.

[18] H. Lee, R. Grosse, R. Ranganath, and A.Y. Ng. Convolutional deep belief networks for scalable unsuper-vised learning of hierarchical representations. InProceedings of the 26th Annual International Conferenceon Machine Learning, pages 609–616. ACM, 2009.

[19] T. Mensink, J. Verbeek, F. Perronnin, and G. Csurka. Metric Learning for Large Scale Image Classifi-cation: Generalizing to New Classes at Near-Zero Cost. InECCV - European Conference on ComputerVision, Florence, Italy, October 2012.

[20] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. InProc. 27thInternational Conference on Machine Learning, 2010.

[21] N. Pinto, D.D. Cox, and J.J. DiCarlo. Why is real-world visual object recognition hard?PLoS computa-tional biology, 4(1):e27, 2008.

[22] N. Pinto, D. Doukhan, J.J. DiCarlo, and D.D. Cox. A high-throughput screening approach to discoveringgood forms of biologically inspired visual representation.PLoS computational biology, 5(11):e1000579,2009.

[23] B.C. Russell, A. Torralba, K.P. Murphy, and W.T. Freeman. Labelme: a database and web-based tool forimage annotation.International journal of computer vision, 77(1):157–173, 2008.

[24] J. Sánchez and F. Perronnin. High-dimensional signature compression for large-scale image classification.InComputer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, pages 1665–1672. IEEE,2011.

[25] P.Y. Simard, D. Steinkraus, and J.C. Platt. Best practices for convolutional neural networks applied tovisual document analysis. InProceedings of the Seventh International Conference on Document Analysisand Recognition, volume 2, pages 958–962, 2003.

[26] S.C. Turaga, J.F. Murray, V. Jain, F. Roth, M. Helmstaedter, K. Briggman, W. Denk, and H.S. Seung. Con-volutional networks can learn to generate affinity graphs for image segmentation.Neural Computation,22(2):511–538, 2010.

**Appendix - A List of Useful Links**

<https://www.pyimagesearch.com/2019/11/25/human-activity-recognition-with-opencv-and-deep-learning/>

<https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/>

<https://en.wikipedia.org/wiki/Recurrent_neural_network>

<https://towardsdatascience.com/understanding-rnn-and-lstm-f7cdf6dfc14e>

<https://stackoverflow.com/>

<https://www.youtube.com/playlist?list=PLkDaE6sCZn6Gl29AoE31iwdVwSG-KnDzF>

<https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks-Part-2/>

<https://adeshpande3.github.io/adeshpande3.github.io/The-9-Deep-Learning-Papers-You-Need-To-Know-About.html>