National University of Computer and Emerging Sciences

Temporal Action Localization in Videos

by

Muteeb Matloob, Musawir Memon, Hassan Zia

Supervised by

Syed Zain-al-Hassan

Co-supervised by

Nida Pervaiz

A report submitted for
the Final Year Project

in the

Department of Computer Science
FAST NUCES Karachi

June 2020

Declaration of Authorship

We, Muteeb/Musawir/Hassan, declare that this report titled, Temporal Action Localization in Videos and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a degree at this University.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this report is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

National University of Computer and Emerging Sciences

Abstract

Department of Computer Science
FAST NUCES Karachi

Bachelors of Computer Science

by Muteeb Matloob, Musawir Memon, Hassan Zia

Supervised by
Syed Zain-al-Hassan
Co-supervised by
Nida Pervaiz

In most web searches, video retrieval and ranking are performed by matching query terms to metadata and other video-level signals. However, we know that videos can contain an array of topics that aren't always characterized by the uploader, and many of these miss localizations to brief but important moments within the video. Temporal localization can enable applications such as improved video search (including search within video), video summarization and highlight extraction, action moment detection, improved video content safety, and many others. We propose a CNN-based approach for the detection of highlights and action moments using the publicly available UCF101 - Action Recognition Data Set YouTube dataset containing videos and labels. We will then compare and validate the results by using the validation set in the dataset.. . .

Contents

List of Figures

Chapter 1

Introduction

Videos are a major part of our society. Advances in analyzing and understanding video may have a substantial impact on all aspects of our current life from entertainment and play to learning and communication. Currently on YouTube, everyday people watch over 1 billion hours of video[1]. Over the internet there is a lot of filler content within a video that can lead to time wastage while searching a specific topic within the video since you have to scrub through the entire video yourself based solely on your intuition. Currently the topics of the video do not give much information about the contents inside the video, this lack of knowledge can lead to multiple problems in various fields. In the past there has been research on video classification models based on their labels, these models classified the videos according to the general topic of the video [2]. We will now be going a step further than before and by temporal localization we will identify the exact moments when the relevant topic is detected within the video using frame level features available in the dataset. This will lead to better understanding and easy access to the content of the video which we can then apply in various practical methods.

1.1 Problem Statement

Currently the retrieval of videos for a query is based on the labelled topic for that specific video, however the topic can be misleading or entirely mislabeled. When a video is uploaded the many events within the video are not always characterized by the uploader, this inability to know the sub-topics of a video can lead to various problems such as time wastage, poor understanding and classification of videos etc. We are unable to gather more information about the video since the labelling is not extensive. If we can improve our understanding of videos and their content more deeply then we can change to more efficient ways of dealing with videos. The detection of continuous actions is

a computationally expensive operation and often yields low accuracies due factors like diverse background and noise, a huge number of classes for actions, and tracking the action through the flow of input segments. There are still a number of methods that this can be achieved and they vary from models and datasets to the identification of specific class actions.

1.2 Research Goal

We want to be able to search within a video for the occurence of a topical entity, we will do this by training a video topic classification model using the available dataset. The model should be able to identify the time stamps at which an event occurs and retrieve them according to the relevant topic it belongs to. There can be multiple topics within a video and multiple occurrences of that topic, we want to be able to distinctly identify different topics and their respective events in order to retrieve the relevant moments according to the search.

Chapter 2

Literature Review

Currently the topics of the video does not give much information about the contents inside the video, this gap in information and knowledge that can lead to multiple problems in various fields has been attempted to be filled in the past by researching on video classification models based on labels among the video dataset.

The detection of continuous actions has proven itself to be a computationally expensive operation and often yields low accuracies due factors like diverse background and noise, a huge number of classes for actions, and tracking the action through the flow of input segments. One of the biggest hurdles in this regard has been the lack of big datasets that have proper features and labels. After the recent availability of large datasets a lot of research has been carried out that have benefited with more accurate and efficient results.

Challenges with the Youtube 8m dataset that have been faced[3] before are that there are only video-level ground-truth labels. It lacks additional information that specifies how the labels are localized within the video, nor their relative prominence in the video, hence inferring their importance for the full video is challenging. Models have been trained with the aim of predicting the main theme of the video using the frame level features as input. Such models have shown competitive results for video-level work in previous studies.

2.1 Image Recognition and Object Detection

There is a lot of progress in the area of image recognition and the new aspect has been known to identify the moments within the video [4,5,6]. Where we trim the video to our required moment or skipping through our required scene (which we do not know where

is that in the whole video so we keep searching for it), some research has been done on online videos on youtube by the researchers [7]. That was about identifying the moments within the videos. This problem needs a solution because long and un-relevant parts of the videos end up taking a lot of space and time and a video can contain multiple instances of actions at different points in time one video can contain multiple action instances but also include irrelevant background data as well. Say we are detecting something in surveillance video of any place where some event has happened, we can localize it to the part we want to see, or the important information, which can save a lot of time compared to if we see it normally and save a lot of space and computational cost of trimming the video [7]. Mostly we rely on the manual method of trimming the selected feature, but we need to innovate ourselves and adopt methodologies which can help improve our lifestyle and computational cost and time. An important aspect of improving the current methods is to capture motion information, which is important for modeling actions and determining their temporal boundaries.

2.2 Temporal Segments

There are two directions to the temporal localization [8]. When Data to be trained has only video-level category labels and no temporal annotations, researchers articulated this as weakly supervised problems or multiple instance learning problems to identify the main part of the untrimmed or original video and then temporally localize the key moment of the video by selecting key instances. Another way is to do it is through learning from the data when the temporal boundaries have been explained for required moments in original videos. Most of the work done in this problem classifies this as a classification problem and adopts a temporal sliding window approach, where each window of frames is considered as an action candidate subject to classification. Training data of strongly supervised models consists of video clips with constant temporal length frames. Their length can be an arbitrary value. Some researchers discovered that there are several benefits of using video level representations rather than training classifiers directly on frame-level features. After that proceeding with extracting a task-independent and video-level feature vector from the frame-level features for each video[9]. These include the application of standard classifiers since the dimensions of a video is the same across videos. It also helps with compactness since we get a compact representation for the entire video, thereby reducing the training data size exponentially.

	$K_c^p = 1$	$K_c^p = 2$	$K_c^p = 3$	$K_c^p = 4$
CBR-C3D	38.6	**39.6**	39.4	37.8
CBR-TS	42.7	44.5	**45.2**	44.8

FIGURE 2.1: Comparison of cascaded step Kp
c = 1,2,3,4 for temporal proposal generation through regression (AR@F=1.0) on THUMOS-14.

2.3 Classification Models

Classic algorithms like Naive-Bayes [10] and Regression [11] have also been used for the temporal action detection in the past. Though their implementation had to be adjusted since for continuous action detection you have to input features of multiple frames for classification. One way of doing that in the past has been to extract volumetric features spanning multiple frames and multiple spatial areas of the frames. These volumetric features are sent for classification by using a sliding window method. Another approach used is Cascading Boundary Regression in which the output of a frame segment is used as the input for the next segment, hence maintaining the continuity of features that is required for action detection. The comparison between various cascaded steps for temporal action detection can be seen below in Table 1.

2.4 CNN Models

We studied a number of Convolutional Neural Networks to help us identify which architectures are best for action detection in videos. Variations of CNNs such as R-CNN, G-CNN and T-CNN have been previously used for similar purposes [12,13,14].

Recently, 2D Convolutional Neural Networks (2DCNN) trained on ImageNet to perform RGB image classification such as AlexNet have gradually shown their power, but their performance is limited since they can only capture appearance information. In the past various other approaches have also been used like decoding each video at one-frame-per-second, and using a Deep CNN pre-trained on ImageNet to extract the hidden representation immediately prior to sending it to the classification layer. One of the key factors while comparing the CNNs was of time, training and testing, While Faster R-CNN was very efficient when it came to time T-CNN with tube proposal networks gave better result for action detection, specifically for actions that ran through multiple segments of the video [15].

2.4.1 Networks and Layers

It was also observed that multiple stages are introduced for the most efficient and accurate classifications, for example the first Network is usually responsible for identifying potential segments in the video where action is likely to occur [16,17,18] but since the YouTube-8M Dataset provides the start and end times of segments, this won't be necessary. The following Networks will be tasked with feature extraction, the convolutional layers are responsible for identifying regions/segments of interest. After which a DNN using the segment pool and the given classes will classify the actions, localizing the classified actions to the exact time when they occur in the segment. The classes here will be based on the annotations and labels that are present on a frame level in the Dataset.

One of the major problems that might arise with such an approach is the loss of temporal identification [19] of the events since some of the CNNs do not maintain the order of the timeline, to counter this we will have to ensure the convolutional layers and the tube proposal networks maintain the flow of input segments by using a tracker proposal network.

Chapter 3

Experiments

3.1 Methodology

3.1.1 Data set

UCF101 is a data set of realistic action videos , sourced via YouTube, it contains 101 categories of action. There are a total of 13320 videos belonging to 101 categories, this data set provides us with the largest diversity. that is the actions it contains and with the help of a large variety of factors like in camera motion, object appearance, scale, perspective, background details, lighting scenario, etc. There are a total of 25 groups which contain the videos from 101 categories, and each of the groups may contain 4-7 videos of a single action. There may be some common features among the videos that belong to the same group, like similar background and viewpoint, etc. There can be 5 types of categories of actions: 1)Human-Object Interaction 2) Body-Motion Only 3) Human-Human Interaction 4) Playing Musical Instruments 5) Sports. Statistics about video length and their categories can be seen in Figure 1 below

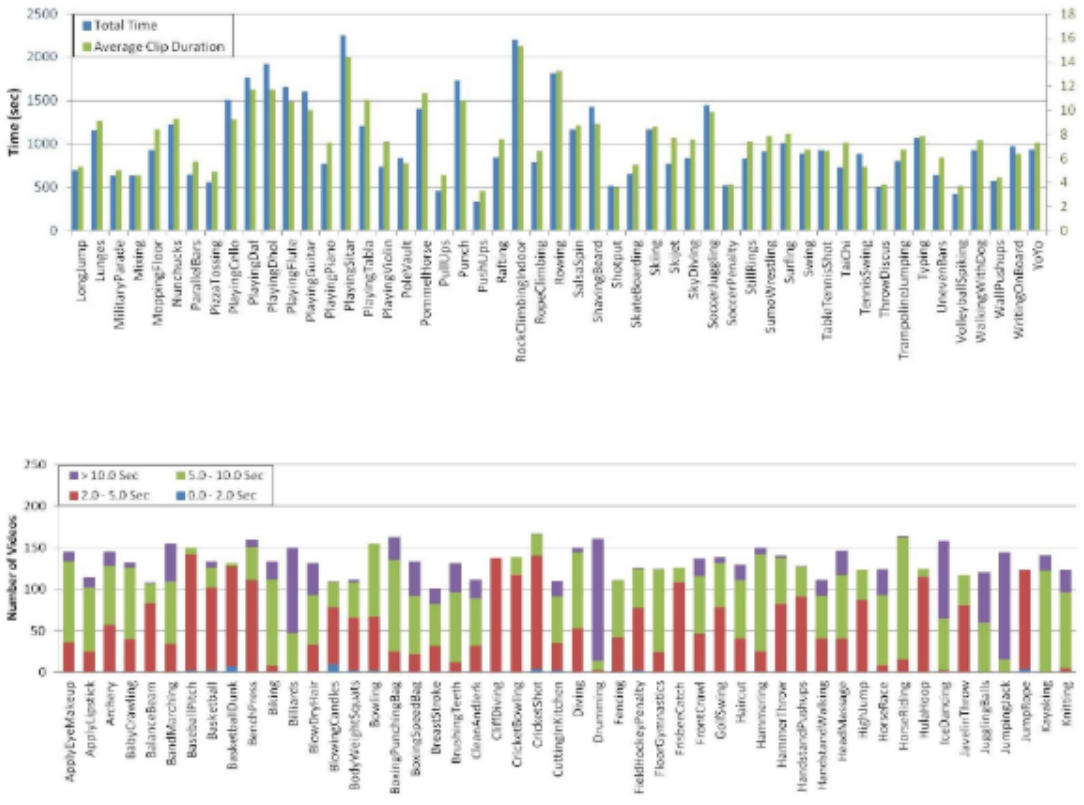

FIGURE 3.1: Data set Statistics

3.1.2 Proposed Framework

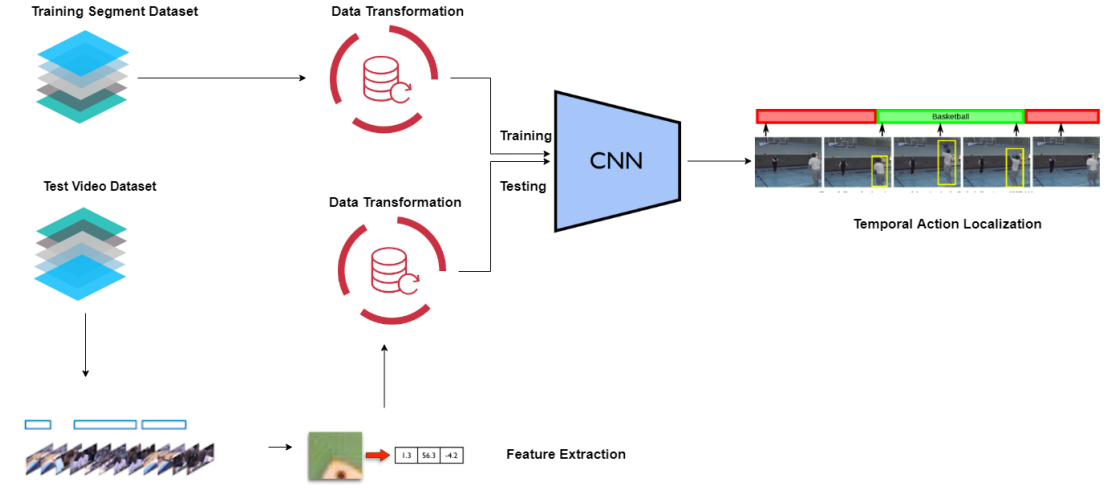

FIGURE 3.2: Proposed Framework

3.2 Experiments

After exploring the dataset and splitting it into training and validation sets. The training set was used for training the model whereas the evaluation set was used to evaluate the results of the model. First we have extracted frames from all the videos in both of the sets. Then we have applied preprocessing on these frames and then trained a model using the frames in the training set. Once we were satisfied with the performance on the validation set, we used the trained model to classify new videos.

After splitting the dataset into training and testing sets, for every video in both of the sets we added a tag.Next the frames were extracted from the training videos which were used to train the model. After the extraction of frames from all the training videos they were saved along with their corresponding tags. Then we trained our model which will be used to predict the tags for videos in the test set. We proceeded with reading all the frames that we extracted earlier for the training images. Next up was, defining the architecture of our model, for this research at this stage and this particular dataset we started with VGG-16 (a convolutional neural network model) pre-trained model which can be seen in Figure 3 below alongside the model we ended up with. Figure 4 is a representation of a time distributed model with which we experimented initially.

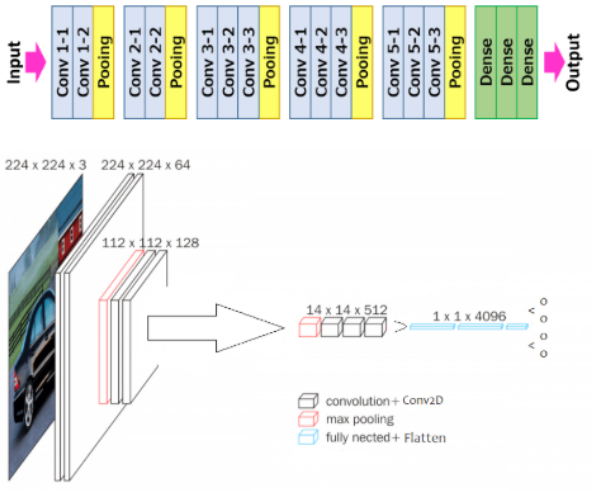

FIGURE 3.3: Working and Architecture of our Model

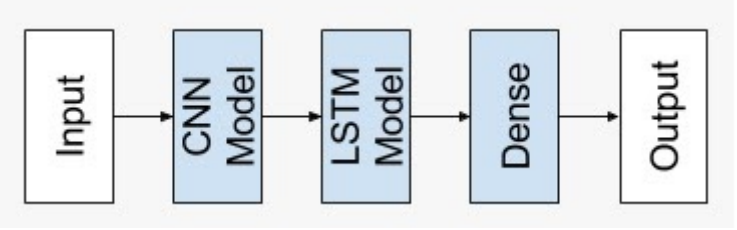

FIGURE 3.4: Diagrammatic Representation of a time distributed model

We then started to train our model, which consisted of four main layers. The convolutional layer, the pooling layer, the flattening layer and finally the classification layer. The training frames are used to train our model while the validation frames are used to validate the performance of our model. In order to avoid retraining the model everytime the weights are saved.

For predictions, two lists have been created – one to store the predictions and the other to store the actual tags. Then, we proceeded to extract frames from the videos belonging to the test set and store it. After reading all the frames, the features were extracted for these frames.

The accuracy score was calculated using the predicted tags and the actual tags of each video

For further work we can use various networks of the CNN, like the architecture in Figure 3, to work in sequence in order to get the best results. We will keep exploring for better options, if found, throughout the research.

3.3 Results

3.3.1 1st Phase

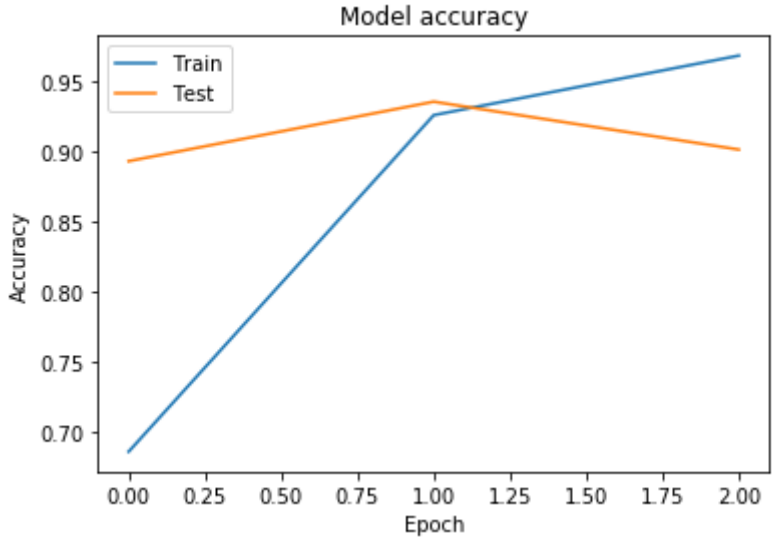

FIGURE 3.5: Model Validation Accuracy Phase 1

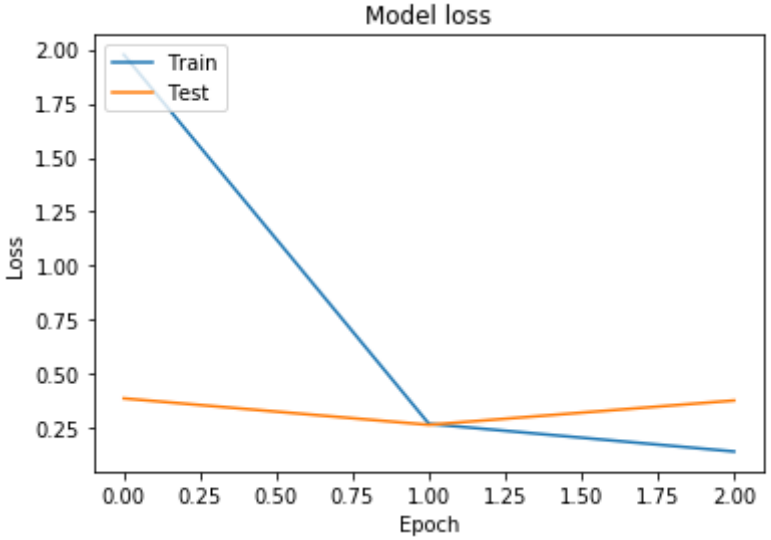

FIGURE 3.6: Model Validation Loss Phase 1

Training of our model at 2 epochs gave a validation accuracy 90.18We predicted 3398 frames of various videos belonging to different human actions, to which the accuracy of our model's prediction was 45.32

We believe this is a good start towards making an efficient temporal action detection model. On the UCF101 data set's information page the best models gave an accuracy of 45

3.3.2 2nd Phase

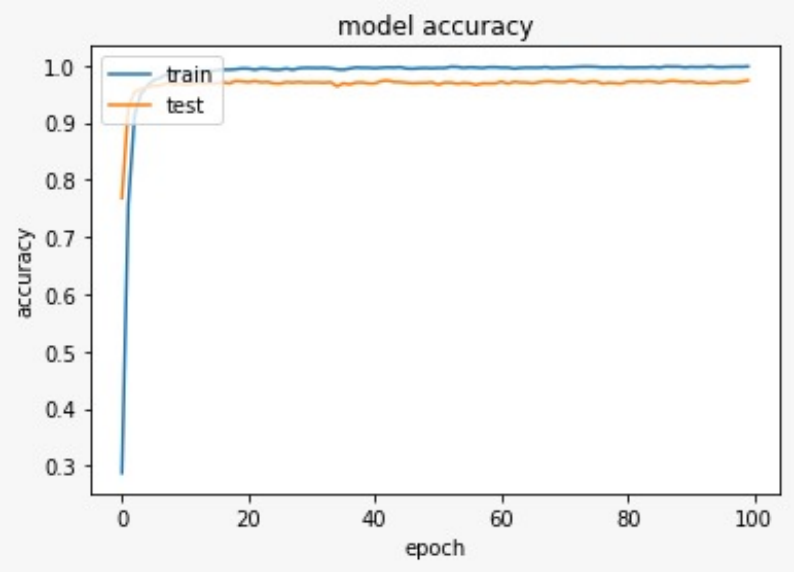

FIGURE 3.7: Model Validation Accuracy Phase 2

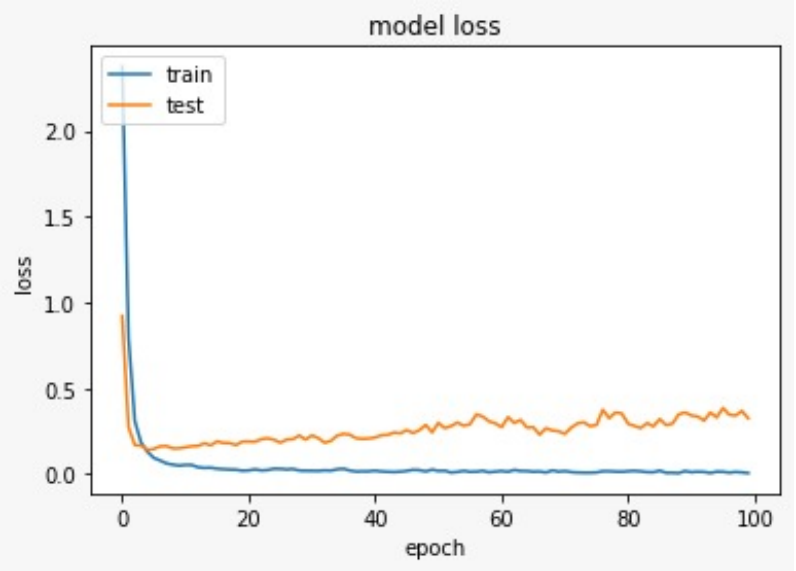

FIGURE 3.8: Model Validation Loss Phase 2

This time we trained our model using 100 epochs based on a transfer learning model. For that purpose we used pre-trained weights of ImageNet (more details and reference). We focused on fine tuning our model on the UCF-101 data set to maximize the accuracy. Total of 23 action categories were used to train this model, we desired to and tried to include more classes but ran into constraints caused by limited resources, primarily the RAM available to us. The overall accuracy was 47.5

Further research can be done as a comparative study towards other such experiments where different data sets and variations of CNN models have been used. We are hoping

to excel the overall understanding of video sub-topic classification by experimenting and deducing efficient methods and models.

Chapter 4

References

[1] Chen Sun, Sanketh Shetty, Rahul Sukthankar, Ram Nevatia, Temporal Localization of Fine-Grained Actions in Videos by Domain Transfer from Web Images, University of Southern California, Los Angeles, CA, USA, 2015

[2] Rohit Gandhi, R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Understanding Object Detection Algorithms, towardsdatascience.com, 2018

[3] Sami Abu-El-Haija, Nisarg Kothari, Joonseok Lee, Paul Natsev, George Toderici, Balakrishnan Varadarajan, Sudheendra Vijayanarasimhan, YouTube-8M: A Large-Scale Video Classification Benchmark, Google, 2016

[4] Zheng Shou, Dongang Wang, Shih-Fu Chang, Columbia University, Temporal Action Localization in Untrimmed Videos via Multi-stage CNNs, New York, NY, USA, 2016

[5] Yu-Wei Chao1, Sudheendra Vijayanarasimhan, Bryan Seybold, David A. Ross, Jia Deng, Rahul Sukthankar, Rethinking the Faster R-CNN Architecture for Temporal Action Localization ,University of Michigan, 2015

[6] Alexander Kläser, Marcin Marszałek, Cordelia Schmid, Andrew Zisserman, Human Focused Action Localization in Video, Engineering Science University of Oxford, UK, 2010

[7] Sudheendra Vijayanarasimhan, David Ross, Capturing Special Video Moments with Google Photos, Google AI Blog, 2019

[8] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition, TPMAI, 2013

[9] Ke Yang, Xiaolong Shen, Peng Qiao, Shijie Li, Dongsheng Li, Yong Dou, Exploring Frame Segmentation Networks for Temporal Action Localization, Cornell, 2019

[10] Yan Ke, Rahul Sukthankar, Martial Hebert, Efcient Visual Event Detection using Volumetric Features, Carnegie Mellon, 2016

[11] Jiyang Gao, Zhenheng Yang, Ram Nevatia, Cascaded Boundary Regression for Temporal Action Detection, USC, 2017

[12] Rui Hou, Chen Chen and Mubarak Shah, Tube Convolutional Neural Network (T-CNN) for Action Detection in Videos, International Conference in Computer Vision (ICCV), 2017.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, Imagenet classification with deep convolutional neural networks, NIPS, 2012

[14] Tuan-Hung Vu, Anton Osokin, Ivan Laptev, Tube-CNN:Modeling temporal evolution of appearance for object detection in video, INRIA/ENS, Paris, France, 2018

[15] K. Simonyan and A. Zisserman, Two-stream convolutional networks for action recognition in videos, NIPS, 2014

[16] K. Soomro, H. Idrees, and M. Shah, Action localization in videos through context walk, ICCV, 2015

[17] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, Learning spatiotemporal features with 3d convolutional networks, ICCV, 2015

[18] P. Weinzaepfel, Z. Harchaoui, and C. Schmid, Learning to track for spatio-temporal action localization, ICCV, 2015

[19] G. Gkioxari and J. Malik. Finding action tubes, CVPR, 2015

Code Segment

```
# open .txt file which have names of training videos
f = open("trainlist01.txt", "r")
temp = f.read()
videos = temp.split('\n')

# creating a dataframe having all of the video names
train = pd.DataFrame()
train['video_name'] = videos
train = train[:-1]
print(train)

for i in range(0,train.shape[0]):
  print(train['video_name'][i].split('/'))
  train['video_name'][i]=train['video_name'][i].split('/')[1]
  train['video_name'][i]=train['video_name'][i].split(' ')[0]

print(train)

# Utilities to fetch videos from UCF101 dataset
import math   # for mathematical operations
from google.colab.patches import cv2_imshow
UCF_ROOT = "http://crcv.ucf.edu/THUMOS14/UCF101/UCF101/"
_VIDEO_LIST = None
_CACHE_DIR = tempfile.mkdtemp()
data=[]
labels=[]

def list_ucf_videos():
  """Lists videos available in UCF101 dataset."""
  global _VIDEO_LIST
  if not _VIDEO_LIST:
    index = request.urlopen(UCF_ROOT).read().decode("utf-8")
    videos = re.findall("(v_[\w_]+\.avi)", index)
    _VIDEO_LIST = sorted(set(videos))
  return list(_VIDEO_LIST)

def fetch_ucf_video(video):
  """Fetchs a video and cache into local filesystem."""
  cache_path = os.path.join(_CACHE_DIR, video)
```

```
  if not os.path.exists(cache_path):
    urlpath = request.urljoin(UCF_ROOT, video)
    print("Fetching %s => %s" % (urlpath, cache_path))
    data = request.urlopen(urlpath).read()
    open(cache_path, "wb").write(data)
  return cache_path

def crop_center_square(currentFram):
  labels, data = currentFram.shape[0:2]
  min_dim = min(labels, data)
  Xstart = (data // 2) - (min_dim // 2)
  Ystart = (labels // 2) - (min_dim // 2)
  return currentFram[Ystart:Ystart+min_dim,Xstart:Xstart+min_dim]

def load_video(path, temp, max_frames=0, resize=(224, 224)):
  opened_c = cv2.VideoCapture(path)
  rate_frame = opened_c.get(5) #rate of the frame
  data=1
  while(opened_c.isOpened()):
    id_frame = opened_c.get(1) #current frame
    returned, currentFram = opened_c.read()
    if (returned != True):
      break
    if (id_frame % math.floor(rate_frame) == 0):
      currentFram = crop_center_square(currentFram)
      currentFram = cv2.resize(currentFram, resize)
      currentFram = currentFram[:, :, [2, 1, 0]]
      data.append(currentFram)
      labels.append(temp)
  opened_c.release()

ucf_videos = list_ucf_videos()
for video in ucf_videos:
  if video in train.values:
    video_path = fetch_ucf_video(video)
    temp=video_path.split('/')[3]
    temp=temp.split('_')[1]
    labels.append(temp)
    load_video(video_path, temp)
```

```
#Truncating Labels array to be equal to data samples.
labels=labels[:17154]



data = np.array(data)

trainData, testX, trainLabels, testY = train_test_split(data, labels, random_state=50,
test_size=0.2, stratify = labels)

# creating dummies of target variable for train and validation set
trainLabels = pd.get_dummies(trainLabels)
testY = pd.get_dummies(testY)

model_basic = VGG16(weights='imagenet', include_top=False)

trainData = model_basic.predict(trainData)
trainData.shape

testX = model_basic.predict(testX)
testX.shape

trainData = trainData.reshape(13723, 7*7*512)
testX = testX.reshape(3431, 7*7*512)

# normalizing the pixel values
max = trainData.max()
trainData = trainData/max
testX = testX/max

trainData.shape

#Transfer Learning.
#defining the transferModel architecture
transferModel = Sequential([
  Dense(1024, activation="relu", input_shape=(25088,)),
  Dropout(0.5),
  Dense(512, activation="relu"),
  Dropout(0.5),
  Dense(256, activation="relu"),
  Dropout(0.5),
  Dense(128, activation="relu"),
```

```
  Dropout(0.5),
  Dense(23, activation='softmax'),
])


transferModel.compile(optimizer="Adam",loss="categorical_crossentropy",metrics=["accuracy"])


graph_data = transferModel.fit(
  trainData,
  trainLabels,
  epochs=100,
  validation_data=(testX, testY),
  batch_size=128
)

graph.plot(graph_data.graph_data['accuracy'])
graph.plot(graph_data.graph_data['val_accuracy'])
graph.title('transferModel accuracy')
graph.ylabel('accuracy')
graph.xlabel('epoch')
graph.legend(['train', 'test'], loc='upper left')
graph.show()

graph.plot(graph_data.graph_data['loss'])
graph.plot(graph_data.graph_data['val_loss'])
graph.title('transferModel loss')
graph.ylabel('loss')
graph.xlabel('epoch')
graph.legend(['train', 'test'], loc='upper left')
graph.show()
```

Turnitin Similarity Segment

Report Similarity

Turnitin Originality Report

Temporal localization of topics

From abcd (General)

- Processed on 03-Jun-2020 10:56 PKT
- ID: 1336924187
- Word Count: 2512

Similarity Index
9%
Similarity by Source

Internet Sources:
5%
Publications:
5%
Student Papers:
2%

Code Similarity

Turnitin Originality Report

Processed on: 07-Jun-2020 21:32 PKT
ID: 1339421607
Word Count: 377
Submitted: 1

Code

Similarity Index	Similarity by Source	
16%	Internet Sources:	5%
	Publications:	6%
	Student Papers:	15%