



**HAL**  
open science

## Supervised Learning of Procedures from Tutorial Videos

S. Arunima, Amlan Sengupta, Aparna Krishnan, D. Venkata Vara Prasad,  
Lokeswari Y. Venkataramana

► **To cite this version:**

S. Arunima, Amlan Sengupta, Aparna Krishnan, D. Venkata Vara Prasad, Lokeswari Y. Venkataramana. Supervised Learning of Procedures from Tutorial Videos. 5th International Conference on Computational Intelligence in Data Science (ICCIDS), Mar 2022, Virtual, India. pp.356-366, 10.1007/978-3-031-16364-7\_28 . hal-04381287

**HAL Id: hal-04381287**

**<https://inria.hal.science/hal-04381287v1>**

Submitted on 9 Jan 2024

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution 4.0 International License



This document is the original author manuscript of a paper submitted to an IFIP conference proceedings or other IFIP publication by Springer Nature. As such, there may be some differences in the official published version of the paper. Such differences, if any, are usually due to reformatting during preparation for publication or minor corrections made by the author(s) during final proofreading of the publication manuscript.

# Supervised Learning of Procedures from Tutorial Videos

Arunima S<sup>1</sup>[0000-0002-9278-8717], Amlan Sengupta<sup>2</sup>, Aparna Krishnan<sup>3</sup>, D. Venkata Vara Prasad<sup>4</sup> and Lokeswari Y Venkataramana<sup>5</sup>

<sup>1,2,3,4,5</sup> Sri Sivasubramaniya Nadar College of Engineering, Department of Computer Science and Engineering, Kalavakkam, Chennai, 603110.

arunima17016@cse.ssn.edu.in<sup>1</sup>, amlan17008@cse.ssn.edu.in<sup>2</sup>,  
aparna17012@cse.ssn.edu.in<sup>3</sup>, dvvprasad@ssn.edu.in<sup>4</sup>,  
lokeswariyv@ssn.edu.in<sup>5</sup>

**Abstract.** Online educational platforms and MOOCs (Massive Open Online Courses) have made it so that learning can happen from anywhere across the globe. While this is extremely beneficial, videos are not accessible by everyone, due to time constraints and lower bandwidths. Textual step-by-step instructions will serve as a good alternative, being less time-consuming to follow than videos, and also requiring lesser bandwidth. In this work, the authors aim to create a system that extracts and presents a step-by-step tutorial from a tutorial video, which makes it much easier to follow as per the user's convenience. This can be mainly accomplished by using Text Recognition for academic videos and Action Recognition for exercise videos. Text Recognition is accomplished using the Pytesseract package from Python which performs OCR. Action Recognition is performed with the help of a pre-trained OpenPose model. Additional information is extracted from both exercise and academic videos with the help of speech recognition. The information extracted from the videos are then segmented into procedural instructions which is easy to comprehend. For exercise videos, a frame-wise accuracy of 79.27% is obtained. Additionally, for academic videos, an accuracy of 78.47% is obtained.

**Keywords:** Openpose, OCR, Exercise Videos, Academic Videos, Speech Recognition, Action Recognition, Text Recognition

## 1 Introduction

Knowledge and the dissemination of knowledge has become truly global with the advent of the Internet. Therefore, tutorial videos have become one of the primary media through which knowledge is imparted. However, video streaming is resource-intensive and not always possible in remote areas with poor connectivity or due to time constraints. In such situations, having a system that creates a procedural set of instructions from the tutorial videos, which can be read and reread as per the user's convenience. Text Recognition and Action Recognition can be used to interpret the information relayed in tutorial videos and segmented into a procedural set of instructions. With longer, more detailed tutorial videos this system is the most impactful as it

can turn a complicated video into an easy-to-follow textual tutorial.

The recognition and interpretation of human or robot-induced actions and activities have gained considerable interest in computer vision, robotics, and AI communities and as more and more videos are being made, and more students refer to them for their studies, there is an increase in the need to properly recognize the text content in the video. This is the base of text recognition. Text recognition is mainly carried out using Optical Character Recognition (OCR) techniques. The modeling and learning of the extracted features are a critical part of the action recognition, in improving its accuracy. Some popular techniques include optical motion detection, 3-D volume representation, temporal modeling, Hidden Markov Model (HMM) training, Dynamic Time Warping (DTW), multi-view subspace learning, and Pose estimation.

The motivation behind building such a system is to make the dissemination of knowledge over the Internet, especially in these times of online learning, easier and more accessible. The objective of this system is to ensure that students are not deterred by bandwidth restrictions or videos that are too long to be comprehensible. This is the first step towards building systems that make knowledge truly universal

## 2 Related Work

Several research work has been reported in the field of procedure extraction from demonstration videos. The authors [1] proposed the method of temporal clustering and latent variable modeling, and developed a general pipeline for procedure extraction. Temporal clustering consists of aggregating similar frames or images and removing duplicates from the pipeline. This method is further evaluated using improved metrics like temporal purity and temporal completeness are proposed.

Another approach called Video Instance Embedding (VIE) framework [2] which trains deep nonlinear embedding on video sequence inputs is introduced. The authors feel that a two-pathway model with both static and dynamic processing pathways is optimal as it provides analyses indicating how the model works.

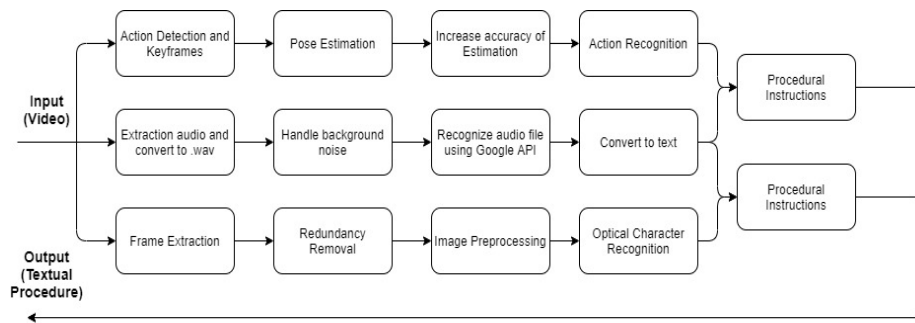
DPC [3] is used to predict a slowly varying semantic representation based on the recent past. A video clip is partitioned into multiple non-overlapping blocks, with each block containing an equal number of frames. An aggregation function is then applied to the blocks and this is used for creating a context representation which helps identify the actions.

The concept of using CompILE [4], is a framework for learning reusable, variable-length segments of hierarchically-structured behavior from demonstration data is also introduced. CompILE utilizes a novel unsupervised, fully-differentiable sequence segmentation module to learn hidden encodings of sequential data that can be re-composed and executed to perform new tasks. Once trained, the model generalizes to sequences of longer length and from environment instances not seen during training. CompILE has been tested in 2D multi-task environments successfully.

There are many approaches to action recognition. A key aspect of using machines to have an understanding of people in images and videos lies on Realtime multi-person 2D pose estimation [5]. This work is used to detect the poses of multiple people in an image. This method uses Part Affinity Fields (PAFs), to learn to associate the values that represent body parts to the individual people identified in the image. Since it is a bottom-up approach, it obtains higher accuracy and real time performance, regardless of the number of people in the image.

### 3 Methodology

The dataset [7] for this project has been majorly taken from YouTube. For exercise videos, a compilation of different exercises along with various angles for each exercise has been given for training. For academic videos, simple tutorials have been used. The proposed methodology employs two different approaches for exercise and academic videos as shown in Figure 1.



**Fig. 1.** Flow of Exercise Cycle

The overall process is divided into three separate flows of recognition: video, audio, text.

In the video recognition flow, the action shown is detected and keyframes are extracted. Then, pose estimation is performed using OpenPose on the detected actions. These estimates are further fine-tuned and the action (exercise) performed is identified.

In the audio recognition flow, the audio is converted to a usable format. The background noise is handled appropriately and lessened. Then, Google's Speech-to-Text is used to transcribe the audio file, and the text contained in the audio file is effectively obtained.

In the text recognition flow, video frames are extracted and redundant frames are removed. Each frame is preprocessed to make the text content clearer, and Optical Character Recognition is performed to identify the text.

The outputs from the three flows are combined to form the procedural instructions, which will be the final output of the system.

#### 3.1 Exercise Videos

The actions performed in exercise videos are identified by using OpenPose estimation [8].

**OpenPose Methodology.** OpenPose is divided into three different blocks which are (a) body+foot detection, (b) hand detection, and (c) face detection. The core block is made up of the combined body + foot key point detector. The points recognised in the body and foot loca-

tions help the facial bounding box to be more accurate and also get more key areas like ears, eyes, nose and neck.

Additionally Convolutional Neural Networks (CNNs) i.e 3 layers of convolutions of kernel 3 are used to obtain reliable local observations of body parts. DeepSort algorithm [6] is also used for tracking multiple people. Consequently, the deep neural networks are used to recognise the actions performed by each person based on single frame-wise joints detected from OpenPose.

The single frame-wise joints are given as input to a baseline CNN network like VGG-19. The feature map thus obtained is processed through a multi-stage CNN pipeline which generates the Part Confidence Maps and Part Affinity Field (PAF) [5].

A Confidence Map consists of a 2D representation of the belief that any given pixel can be used to map a particular body part.

$$S = (S_1, S_2, \dots, S_B) \text{ where } S_b \in R^{w \times h}, b \in 1 \dots B \quad (1)$$

The above equation 1 denotes a confidence map where B is the number of body part locations and each map has a dimension of w\*h. This confidence map is used to denote high probabilities where a body part might be in the input image.

The location and orientation of limbs of multiple people is encoded using a set of 2D vector fields known as Part Affinity Fields. The data is encoded in the form of pairwise connections between body parts i.e it points the direction from one limb to another. The following equation 2 represents part affinity fields L, which consists of a set of "N" 2D matrices for each limb in the body. The dimension of L is w \* h \* n.

$$L = (L_1, L_2, \dots, L_N) \text{ where } L_n \in R^{w \times h \times n}, n \in 1 \dots N \quad (2)$$

In case of multiple people detection due to the presence of multiple heads, hands etc., there is a need to map the correct body parts with the correct person. Equation 3 is used to calculate a predicted part affinity which is based on two of the locations of the body parts  $D_{b_1}$  and  $D_{b_2}$ . This predicted PAF value is used to determine  $E_n$  for the set of  $D_b$  values and maximize it.

$$\max_{z_n} E_n = \max_{z_n} \sum_{m \in D_{b_1}} \sum_{n \in D_{b_2}} E_{mn} \cdot Z_{b_1 b_2}^{mn} \quad (3)$$

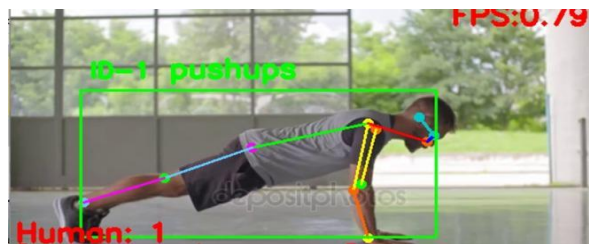
**Action Detection Procedure.** The input consists of a single exercise video from which frames are individually taken and the 36 points corresponding to distinct body parts are identified by the pose estimation algorithm. These points are stored as decimal values in a text file which is further converted into .csv format after adding an additional column i.e. class label. The actions are mapped to their respective class labels in Action function. The csv is then given as input for training the model. The number of cases for each class is given in Table 1.

**Table 1.** Training data for action recognition

Exercise	No. of Lines	Class
Squat	445	0
Jumping Jacks	715	1
Push Ups	351	2
High Knees	744	3
<b>Total</b>	<b>2255</b>	

A sequential keras model with relu activation function is used for training. A classifier trained for specific action is saved. The classifier is loaded along with a pre-trained VGG model [9] to perform action recognition.

Figure 2 is an example of how the points are mapped in an image. A text file is generated containing the sequence of distinct exercises performed throughout the video as shown in Figure 3.

**Fig. 1.** Pushups

```

output - Notepad
File Edit Format View Help
Exercises to be performed are:
highknees
squats
pushups
jumpingjacks

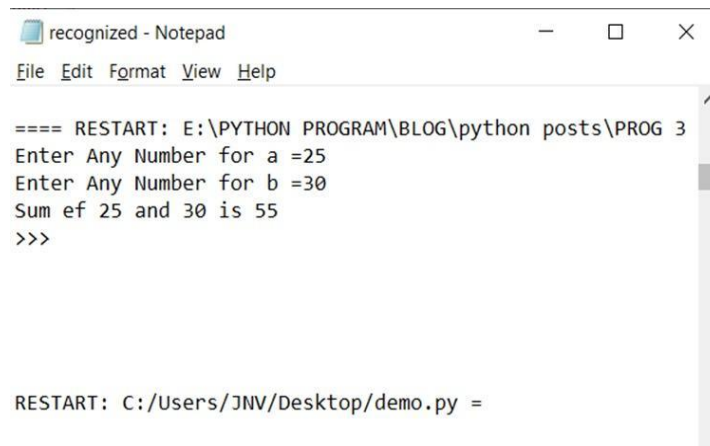
```

**Fig. 2.** Output for Activity Recognition

### 3.2 Academic Videos

The input video is split into various frames at regular intervals using packages like OpenCV [10]. To remove similar and repetitive frames, similarity measures like Mean Squared Error (MSE) and Structural Similarity Index (SSIM) are used. Structural Similarity Index (SSIM) is a metric used to measure the similarity between two images. The similarity between the two images is represented by a value between -1 and +1. A SSIM of +1 denotes that the two images are very similar. A value of -1 denotes the images have very less similarity. The system uses 3 aspects to measure similarity: Luminance, Contrast and Structure. The more similar images are, the more likely they are to be removed. The final collection of images has less similar images thereby making it easier for image pre-processing and recognition of text. For pre-processing, the colorspace of the images is changed to grey-scale using the `cvtColor()` method, and a Gaussian Blur is added using low-pass filters using the `GaussianBlur()` method. The text in the image is detected by covering the textual area with a rectangle and OCR is performed using Pytesseract [11] on that area.

A single academic tutorial video is given as input from which frames are extracted at 3 second intervals. Redundant frames are removed using SSIM [12]. Image preprocessing is performed on the images which includes converting to grayscale, blurring the background and thresholding. OCR using Pytesseract is performed to extract text from the preprocessed images. A text file is generated for recognized text as shown in Figure 4.



```
recognized - Notepad
File Edit Format View Help

==== RESTART: E:\PYTHON PROGRAM\BLOG\python posts\PROG 3
Enter Any Number for a =25
Enter Any Number for b =30
Sum of 25 and 30 is 55
>>>

RESTART: C:/Users/JNV/Desktop/demo.py =
```

Fig. 3. Output for Text Recognition

### 3.3 Speech Recognition

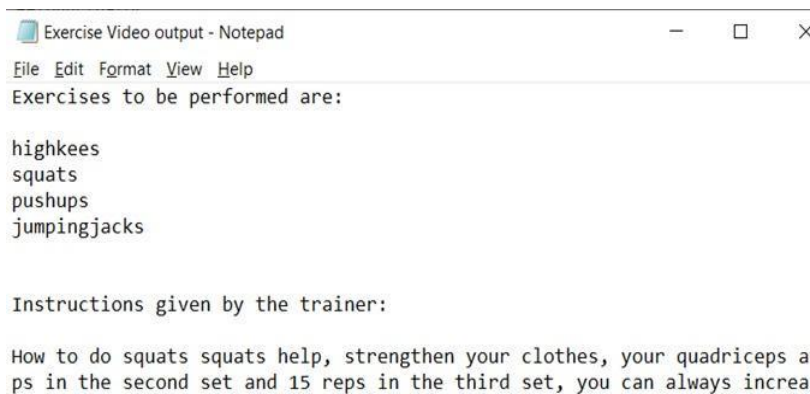
Speech recognition is used to note down the additional information given in the videos. For example, instructions given by the trainer in exercise videos and important points discussed by the teacher in academic videos. Therefore, speech has been included in both types of videos. The audio from the input video is extracted and chunked to sub-clips using the `extract_subclip()` method from the `moviepy` python package. Each sub-clip is converted to .wav format and written back to the sub-clip files. Using each sub-clip as the source, the `adjust-for-ambient-noise()` method is used to remove noise from the sub-clip. Once each sub-clip has been pre-processed, the `recognize-google()` Speech Recognition method from Google is used to recog-



nize the text from each clip. The chunked clips from which the text is recognized are later joined. The text obtained from speech recognition is punctuated using the python library punctuator and then added to the final output for both types of videos.

## 4 Results

For exercise videos, an output file (Figure 5) is generated which contains the sequence of exercises performed throughout the video along with the instructions spoken by the trainer.



```

Exercise Video output - Notepad
File Edit Format View Help
Exercises to be performed are:

highkees
squats
pushups
jumpingjacks

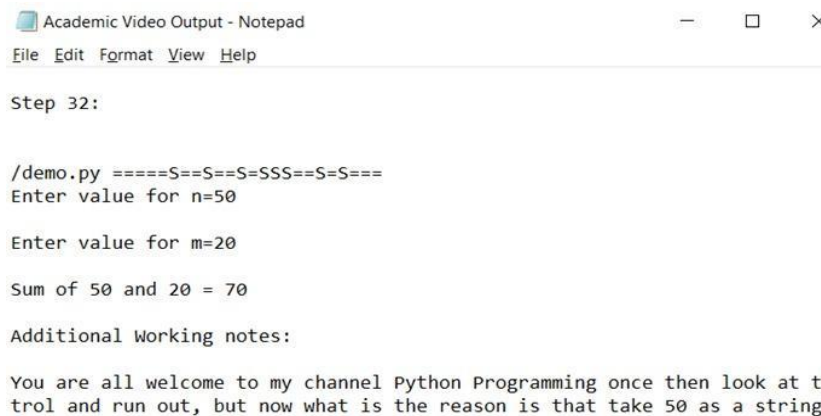
Instructions given by the trainer:

How to do squats squats help, strengthen your clothes, your quadriceps a
ps in the second set and 15 reps in the third set, you can always increa

```

**Fig. 4.** Output for Exercise Videos

For academic videos, an output file (Figure 6) containing the procedural information from the video along with the additional information spoken by the tutor is generated.



```

Academic Video Output - Notepad
File Edit Format View Help
Step 32:

/demo.py =====S==S==S=SS==S=S===
Enter value for n=50

Enter value for m=20

Sum of 50 and 20 = 70

Additional Working notes:

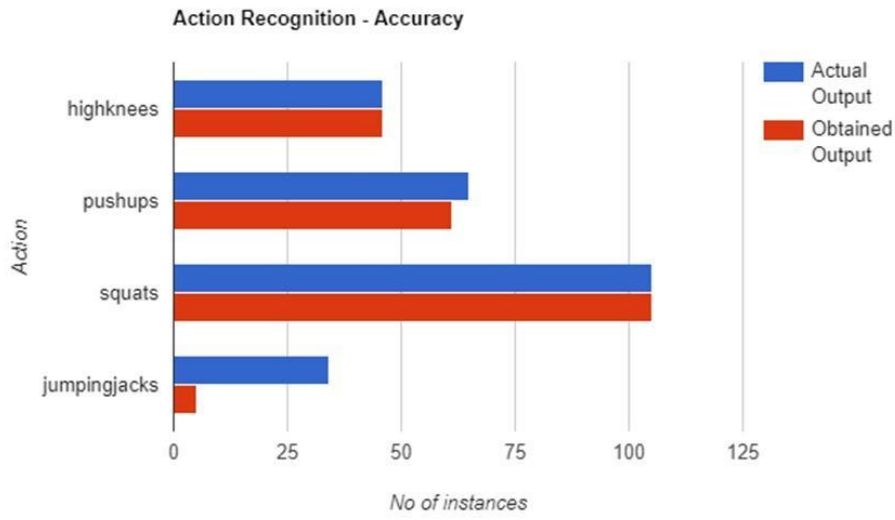
You are all welcome to my channel Python Programming once then look at t
trol and run out, but now what is the reason is that take 50 as a string

```

**Fig. 5.** Output for Academic Videos

#### 4.1 System Performance

The performance and accuracy of the first part of the system is measured in terms of the percentage of exercise instances classified correctly in the case of Exercise Videos. The following graph in Figure 7 shows the accuracy of OpenPose when run over our data.

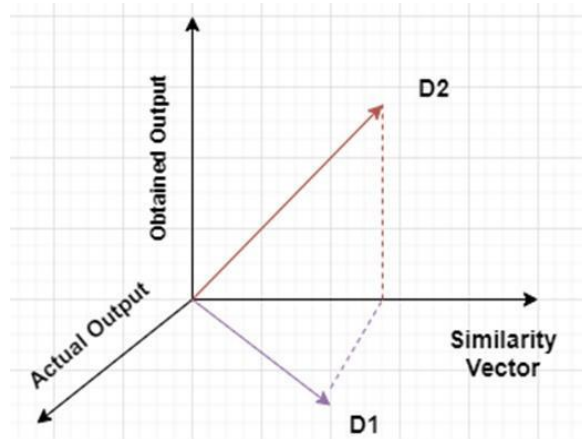


**Fig. 6.** OpenPose Accuracy

The test video consisted of 304 frames, out of which 292 keyframes were detected and action was performed on these keyframes. Out of the 292 keyframes, 51 were classified incorrectly thereby leading to 241 frames classified accurately. This provided the model with a frame-wise accuracy of 79.27%.

The model involves using OpenPose specifically for exercise videos when compared to other models which include a multitude of actions. It is able to accommodate as many exercises as needed. The training is short when compared to other deep learning methods as the pose estimation points are extracted during the length of the video. The classifier containing the pose estimation points for each exercise is trained using a keras sequential model with relu activation. The training time is optimal as only points are given as inputs rather than images.

For Text Recognition, the accuracy is measured using Document Similarity as shown in Figure 8 and 9. Document similarity is calculated by calculating document distance. Document distance is a concept where words (documents) are treated as vectors and is calculated as the angle between two given document vectors. Document vectors are the frequency of occurrences of words in a given document.



**Fig. 7.** Document Similarity - Accuracy of Text Detection

```
PS C:\Users\Aparna Krishnan\Desktop> python simi.py
File actual.txt :
840 lines,
187 words,
41 distinct words
File output.txt :
6149 lines,
1187 words,
261 distinct words
The distance between the documents is: 0.678415 (radians)
PS C:\Users\Aparna Krishnan\Desktop> █
```

**Fig. 8.** Document Similarity - Output

In the case of Academic Videos, the percentage similarity between the gold label document and the output document determines the system performance as shown in Table 2.

**Table 2.** Accuracy for Text Recognition

File 1	File 2	Percentage Similarity
AT	OF	69.42
AT	OF-RR	78.47
OF	OF-RR	89.72

Key for Table 2:

1. *AT*: Actual text that is manually extracted from the video
2. *OF*: Output obtained without performing redundancy removal
3. *OF-RR*: Output obtained by using redundancy removal on the frames
4. *File 1*: The first file taken for calculating similarity
5. *File 2*: The second file taken for calculating similarity

The *actual.txt* file is manually created to act as the gold labels for checking the accuracy of the files that are obtained as output from the OCR function. The output without redundant frames being removed when compared to *actual.txt* is comparatively lesser than the accuracy obtained from comparing *actual.txt* and the output obtained from the frames where redundancy removal has been performed. The similarity between the two generated output files is quite high since only redundant and repetitive frames have been removed.

## 5 Source Code and Computational Resources

The source code for implementing this system has been uploaded in this GitHub repository [13]. The resources used for implementing the system were a 2.60 GHz Intel core i7 CPU and Intel UHD Graphics 630 4 GB GPU.

## 6 Conclusion

This work is the first step in making education accessible to all. With the world going increasingly online, education is now more global than ever before in history. The hope and end goal is to contribute to this success with this system.

This system creates a set of readable instructions that are much more accessible and less resource intensive than videos. The many advantages of such a system will be most felt in places with low internet connectivity, and for longer videos with many steps and sub steps. Based on the results of the first iteration, this system can be extended to other classes of tutorial videos. The accuracy of recognition can also be exponentially improved with each iteration.

As a part of future work, the authors intend to apply this system to all kinds of tutorial videos. The authors also aspire to make this system multilingual, thereby providing unfettered access to tutorial videos globally. Using advanced computing resources such as a dedicated Graphical Processing Unit (GPU), the accuracy of real-time text and action recognition can be increased manifold. This is due to the fact that the model can be trained on a larger data set and for many more epochs using a GPU. With higher computing power and a more diverse data set, the accuracy and efficiency of this model, and the ease of extension to other classes of videos will increase exponentially.

## 7 References

1. Karan Goel and Emma Brunskill. 2018. Unsupervised learning of procedures from demonstration videos.
2. Chengxu Zhuang, Tianwei She, Alex Andonian, Max Sobol Mark, and Daniel Yamins. 2020. Unsupervised learning from video with deep neural embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9563–9572.
3. Tengda Han, Weidi Xie, and Andrew Zisserman. 2019. Video representation learning by dense predictive coding. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 0–0.
4. Thomas Kipf, Yujia Li, Hanjun Dai, Vinicius Zambaldi, Alvaro Sanchez-Gonzalez, Edward Grefenstette, Pushmeet Kohli, and Peter Battaglia. 2019. Compile: Compositional imitation learning and execution. In *International Conference on Machine Learning*, pages 3418–3428. PMLR.
5. Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. 2019. Openpose: Realtime multi-person 2d pose estimation using part affinity fields.
6. Nicolai Wojke, Alex Bewley, and Dietrich Paulus. 2017. Simple online and realtime tracking with a deep association metric.
7. <https://github.com/arunimasundar/Supervised-Learning-of-Procedures/tree/master/dataset>
8. <https://github.com/CMU-Perceptual-Computing-Lab/openpose>
9. Karen Simonyan and Andrew Zisserman. 2015. Very deep convolutional networks for large-scale image recognition.
10. <https://pypi.org/project/opencv-python/>
11. <https://pypi.org/project/pytesseract/>
12. <https://www.mathworks.com/help/images/ref/ssim.html>
13. <https://github.com/arunimasundar/Supervised-Learning-of-Procedures>