Human Pose Estimation
BTP Report

A dissertation

submitted in partial fulfillment of requirements of the degree of

BACHELOR OF TECHNOLOGY

in

Computer Science and Engineering

by

POORVI HEBBAR

(***170050094***)

Under the supervision of

Prof. Ganesh Ramakrishnan

Department of Computer Science and Engineering
Indian Institute of Technology Bombay
Powai, Mumbai – 400076

Declaration

I declare that this written submission represents my ideas in my own words and where others ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Poorvi Hebbar
Roll no. 170050094
IIT Bombay

Acknowledgments

I would like to express my sincere gratitude to Prof. Ganesh Ramakrishnan for his valuable guidance and support throughout this project. I would like to thank him for his insightful suggestions which helped me in smooth and successful completion of the project. I am also thankful to Prof. Arjun Jain for his insights in our meets and also for constant guidance. I am very grateful to Rishabh Dabral for his constant support and motivation at each and every step of this work. and help in getting results. Last but not the least, I would like to thank Srijon Sarkar for constantly being there for any kind of help and for giving valuable feedbacks. I would also like to thank my family and friends for their constant moral support. This work is dedicated to all of them.

Poorvi Hebbar

Abstract

The first part of this report consists of my work on my BTP1: Use of AI and Video Analytics for Anomaly Detection in Proctored Videos. Any behaviour that is considered abnormal or doesn't fit in can be considered an anomaly. Anomalies in Proctored Videos are detected in case a student's behaviour is suspicious which could imply a high chance of cheating in the examinations. In an attempt to detect anomalous behaviours in online exams and implement a way of Automated Proctoring, we hereby try to design an auto-encoder model based on human pose features which would give an idea how anomalous the video clip is.

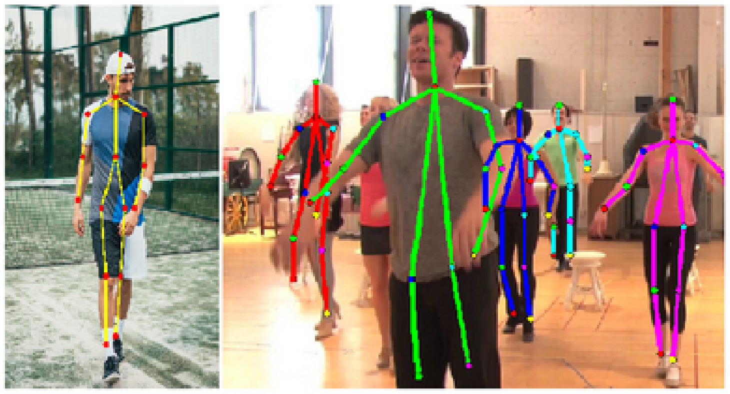
The next part of this report mainly contains a comprehensive summary of the different approaches taken towards Human Pose Estimation, along with recent developments using self-supervision. Human Pose estimation is a Computer Vision problem which estimates the pose of humans either from images or videos, in 2D or 3D. It presents the advantages of using such self-supervised networks as compared to CNN based approaches commonly used. We also look into some experiments using a combination of pre-training and fine tuning with BERT models for Future Pose Prediction, which paves the way for future works in this field.

Contents

Chapter 1

Introduction

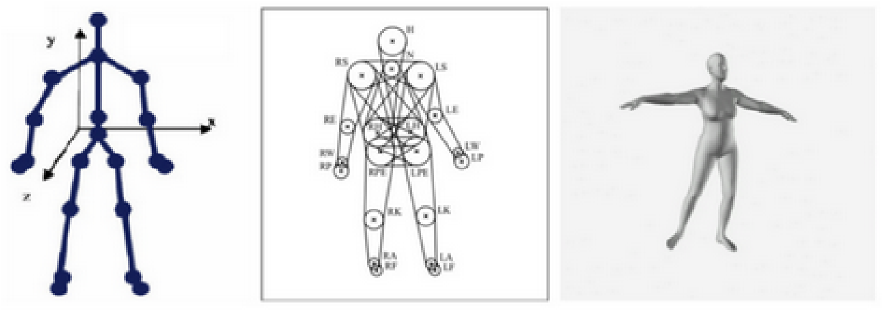
Pose estimation is a computer vision task that infers the pose of a person or object in an image or video. We can also think of pose estimation as the problem of determining the position and orientation of a given person or object relative to the camera. This is done by identifying, and tracking a number of keypoints on a given object or person. For objects, this could be corners or other significant features. When working with humans, these keypoints represent major joints like elbows, knees, wrists, etc. This is referred to as human pose estimation.



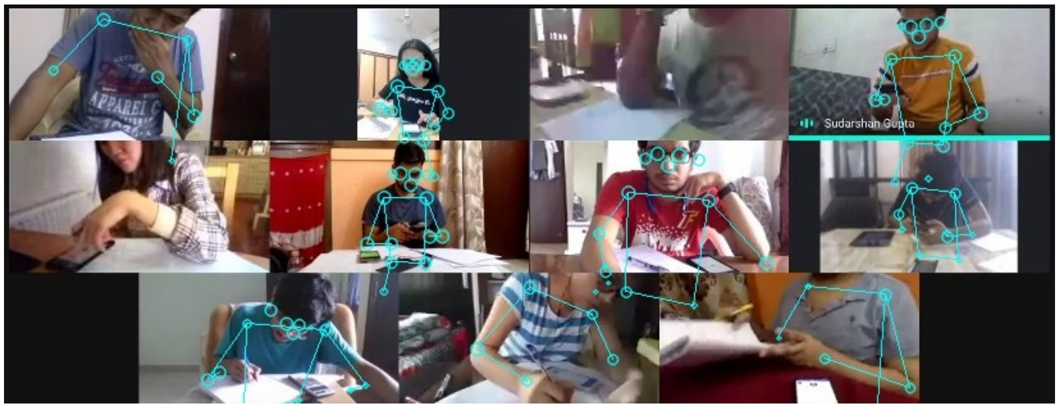
Single-person vs Multi-person Pose Estimation.

Human Pose Estimation is the task of determining the location of a fixed number of joints of a person present in an image.Now this task may be extended to perform pose estimation for multiple people present in the an image,or to estimate their joint locations in 3D. There is a distinction between detecting one or multiple people in an image or video. These two approaches can be referred to as single person and multi person pose estimation. Furthermore, in 3D this notion of pose estimation canalso be extended to the task of shape estimation.

Applications may including video surveillance, assisted living, and sport analysis. The goal of our machine learning model is to track the human body keypoints in images and videos.


Kinematic Model vs Shape-based Model vs Mesh-based Model

Meanwhile the body model formalizes the problem of human pose estimation into that of estimating the body model parameters or the coordinates of the key points. We use a simple N-joint rigid skeleton model (N in our case was 17). These models can be represented as a graph, where each vertex V represents a joint. The edges E can encode constraints or prior beliefs about the structure of the body model.



Any behaviour that is considered abnormal or doesn't fit in can be considered ananomaly. Anomalies in Proctored Videos are detected in case a student's behaviour is suspicious which could imply a high chance of cheating in the examinations. In an attempt to detect anomalous behaviours in online exams and implement a way of Automated Proctoring, in my BTP-1 we tried to design an auto-encoder model based on human pose features which would give an idea how anomalous the video clip is. This report is a summary of experiments conducted in the fields of learning Human Pose Estimation and Future Pose Prediction.

Chapter 2

Anomaly Detection (BTP I)

In this project, we try to detect anomalies in proctored videos of students giving exams using an auto-encoder model based on human pose features.

2.1 What's an Anomaly?

Anomalies here imply any kind of suspicious behaviour which is away from what is expected. The anomalies in proctored videos are mainly instances of students cheating in the exams. Possible actions of students giving exams can be classified as anomalous and non-anomalous Some examples of anomalous actions can be using a phone, standing up and walking away from the laptop screen, or another person entering the camera's view etc as shown in Figure 2.1

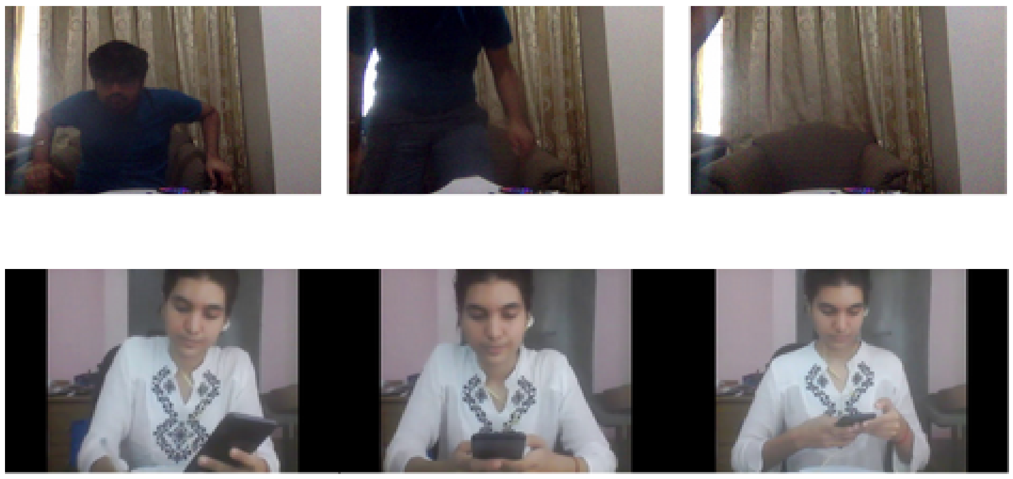


Figure 2.1: Anomalous actions: Walking away from the camera view and Using a mobile phone

Non-anomalous or normal behaviour expected from the students giving exams would be a few hand or neck movements, drinking water, reading or writing, checking their watch for time etc as shown in Figure 2.2

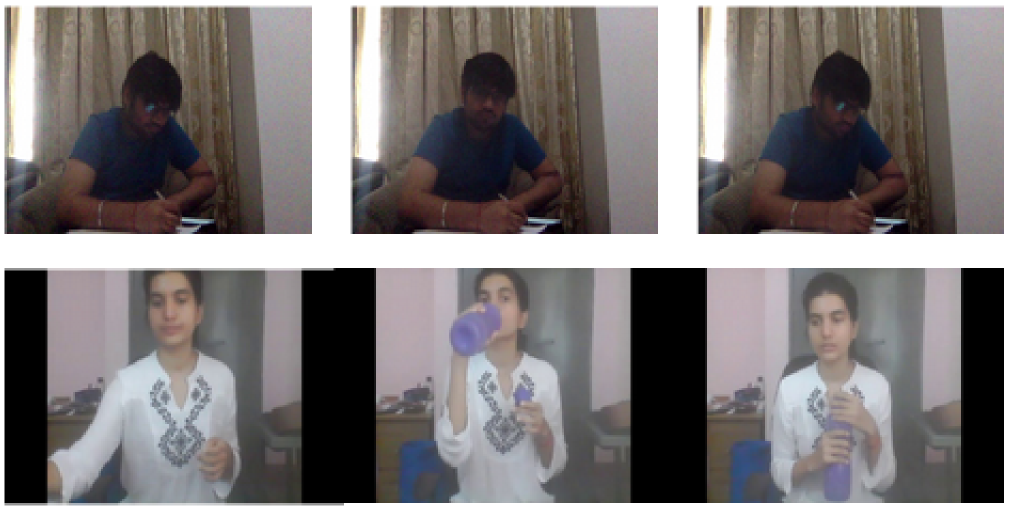


Figure 2.2: Non-Anomalous actions: Writing and Drinking water

Given the different modes in which examinations are conducted the classification of anomalies can be different for each of them. For example, in SAFE exams, the students would be using their phone for the question paper but a phone call would still be anomalous. In case of closed moodle exams typing shouldn't be considered as an anomaly but reading from a different book might be considered as an anomaly.

2.2 Proposed pipeline

The flow diagram of the pipeline can be seen in Figure 2.3

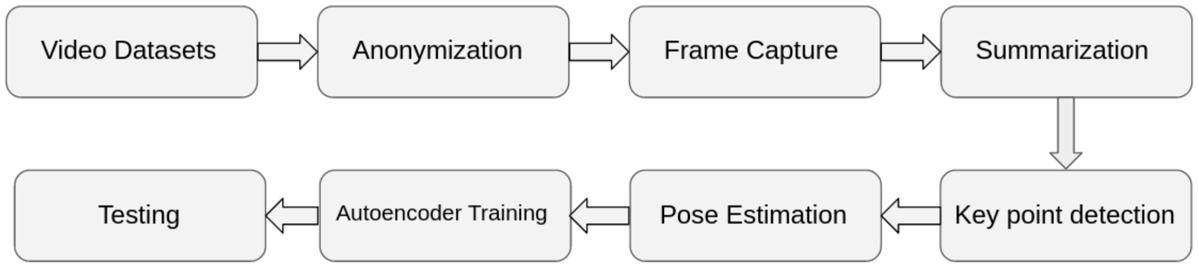


Figure 2.3: Proposed Pipeline

We will now look into each step of the pipeline in detail.

2.2.1 Video Datasets

Collection of Video Datasets were done as follows:

- **Web cam recordings:** Volunteering students performing a few anomalous as well as non-anomalous actions
- **Proctoring videos:** Formally requested the Institute ethics committee for sharing the proctored videos of IITB students giving proctored quizzes and midsems. (** no longer being used)
- **NTU-RGB Dataset:** Available on the Rose lab (Rapid Rich Object Search Lab) page of NTU. This assumes correct pose estimation and ground truth.

2.2.2 Video Summarization

The goal of Video summarization is to identify a small number of keyframes or video segments which contain as much information as possible of the original video. This step was included later as we noticed that pose estimation takes a lot of time and was a bottleneck for the entire process. Thus this would summarize the entire video and give us a subset of frames which we would be interested in and which would contain a jist of the entire video i.e while containing as much info as possible. As the number of frames is decreased to a great extent, the pose estimation step would be a bit faster.

Asynchronous Summarization: The entire video is available beforehand and any frame can be accessed in any order whatsoever. It's offline in nature and the summarization of the proctoring videos is done after the entire video is stored in a local machine. Previous work on asynchronous Video summarization includes:

- Video Maximal Marginal Relevance (Video-MMR) [1]
- Standard Greedy Algorithm

Synchronous Summarization: We have an unpredictable sequence of frames streaming in and we get 1 frame in each iteration. It's online in nature and the frames are selected/ rejected as soon as they enter & the subset of selected frames keeps getting updated in every iteration. Previous work on synchronous Video summarization:

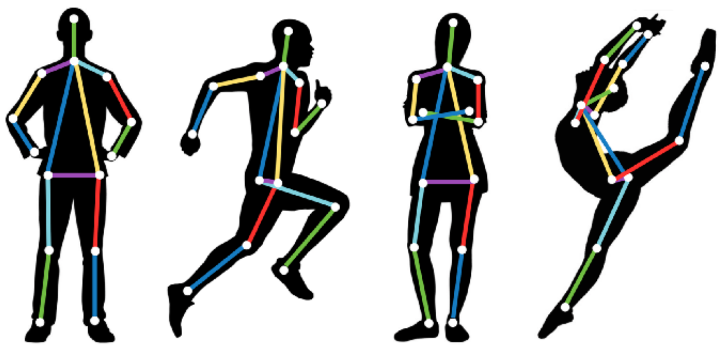
- Stream Greedy Algorithm
- Preemption Streaming
- Sieve Streaming [2]

For the details and results of each algorithm, please refer to the Appendix A.

Chapter 3

Human Pose Estimation

A key element of Scene Understanding is perception and interpretation of humans and the associated interactions. While human perception typically involves inferring the physical attributes about the humans (detection, poses, shape, gaze etc.), interpreting humans involves reasoning about the finer details relating to human activity, behaviour, human-object visual relationship detection, and human-object interactions. This can be broadly classified into 2 domains -Image and Video settings. The goal is to identify the objects interacting with the humans while also estimating the kind of interaction, eg., holding the cup, placing the bowl, moving the furniture, etc. We explore both of these in our works.



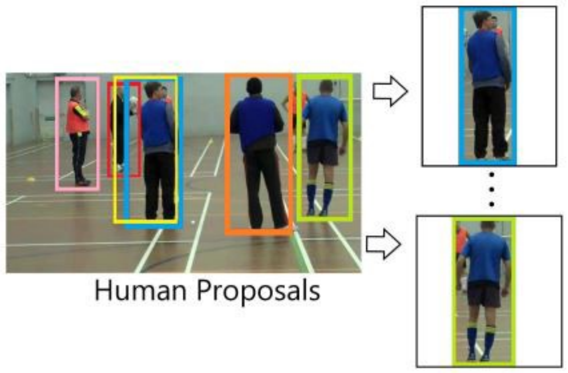
Human Pose Estimation is a vital application of computer vision, which allows it to unlock it's true potential in the field of robotics, VR, gaming, animation and so on. It is the task of determining the location of a fixed number of joints of a person present in an image. Now this task may be extended to perform pose estimation for multiple people present in the an image,or to estimate their joint locations in 3D. Furthermore, in 3D this notion of pose estimation can also be extended to the task of shape estimation.

A major breakthrough that propelled the use of convolutional neural networks (CNN) incomputer vision was the advent of deep learning. Eversince, CNN models have been widely deployed in most computer vision tasks, and unsurpris-ingly, almost all models developed for human pose estimation are based on CNNs. CNN based approaches have topped all image recognition and pose estimation challenges.

3.1 Pipeline

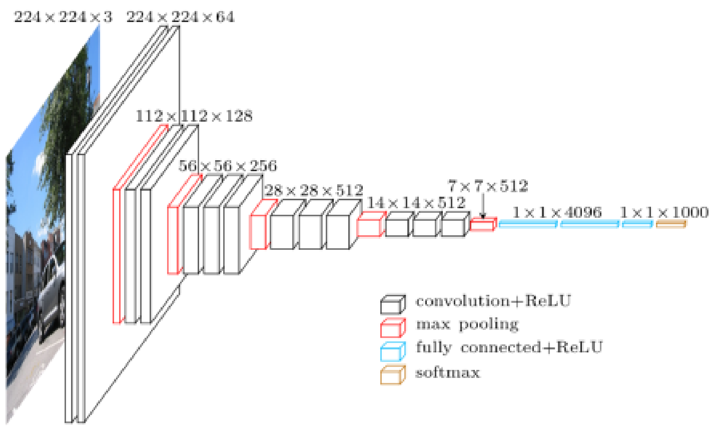
The brief pipeline for pose estimation has 4 steps[3]

- **Pre-processing:** Background removal might be required for segmenting the humans from the background and some algorithms especially the ones used for multi person pose estimation, create bounding boxes for every human present in the image.



Bounding Box creation. Image courtesy Fang et al. (2017)

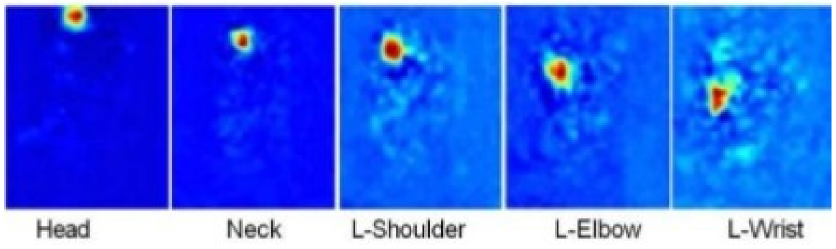
- **Feature Extraction:** This refers to deriving some values from raw data (such as an image or video in our case), that can be used as input to a learning algorithm. The features are implicit and the approach uses an encoder-decoder architecture. Instead of estimating keypoint coordinates directly, the output from the decoder creates heatmaps or confidence maps representing the likelihood that a keypoint is found in a given pixel or region of an image.



VGG16 : A CNN based feature extraction and image classification architecture

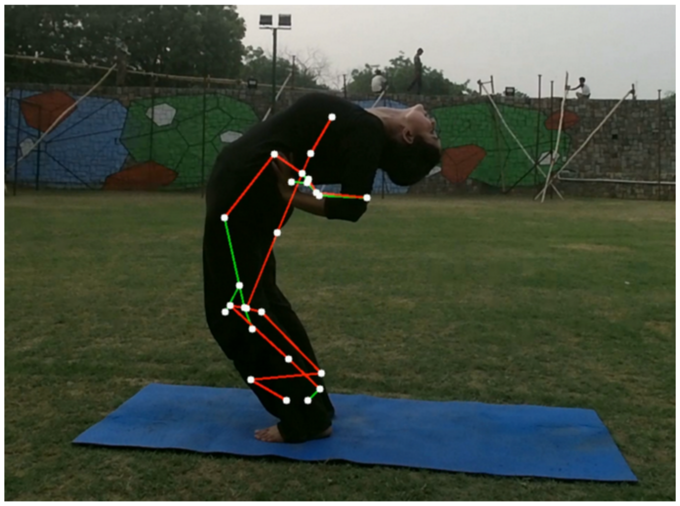
We use a pre defined MobileNet-based architecture for the encoder. This architecture has depthwise separable convolutions that require fewer parameters and less computation but still provides solid accuracy.

- **Inference:** We use confidence maps for predicting joint locations. Confidence map is the probability distribution over the image, representing the confidence of a particular joint location at every pixel. The implementation we used had a top down approach, ie the network first uses an object detector to draw a box around each person, then estimates the keypoints within each cropped region.



Confidence map examples

- **Post-Processing:** The final output is a single set of heatmaps, and sometimes predicting joint positions from an input image does not reject or correct any unnatural human pose. This can sometimes lead to weird Human Pose Estimation like for example the photo of the person doing a yoga pose.



Pose Estimation using Kinect containing weird and unnatural pose

Postprocessing algorithms reject unnatural human poses. The output pose from any Pose Estimation pipeline is passed through a learning algorithm which scores every pose based on its likeliness. Poses that get scores lower than a threshold are ignored.

3.2 Frames to posenets

The pose score for an individual determines the likelihood of that pose and the pose score for each keypoint represents the likelihood of the joint being at that pixel. We used Google's pytorch implementation of the posenet model[4] to build our own code[5]. So with N=17 key points, there were 2*17 coordinates and each joint had a pose score, apart from the individual's entire pose score, thus resulting in a 52 dimensional vector for every person in the frame.

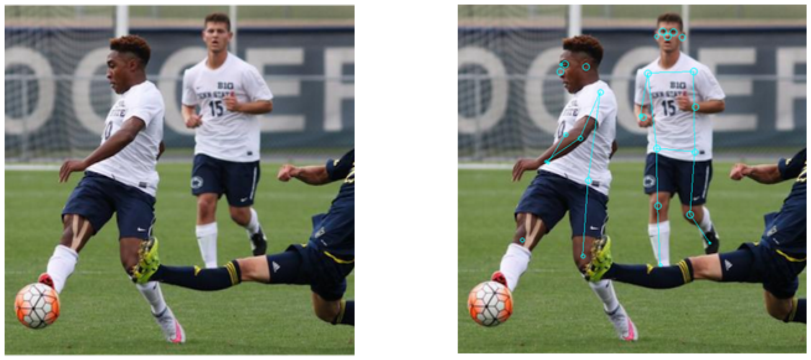
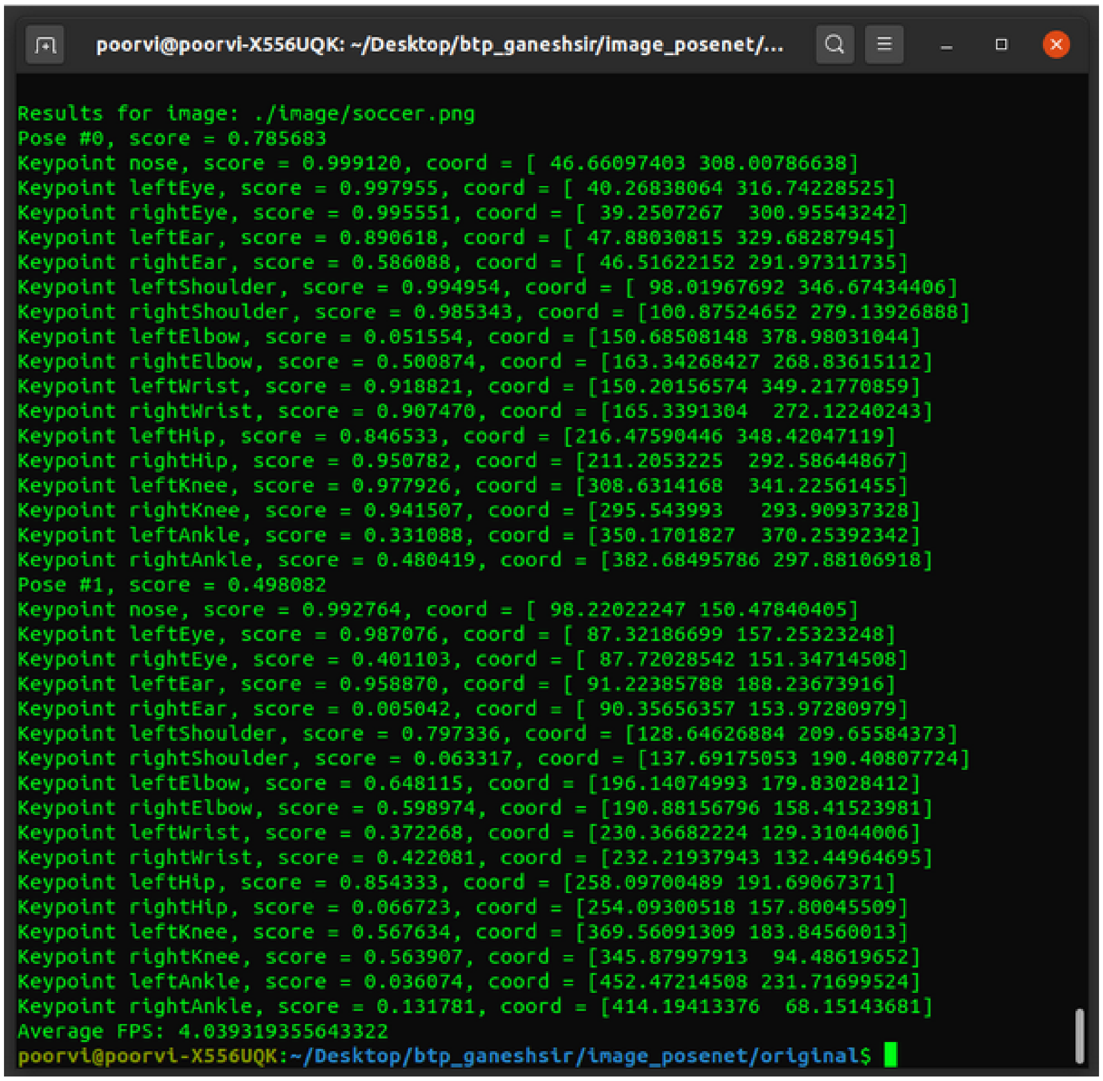

(a) original (b) posenet

Figure 3.1: Example of posenet from original soccer image

An example of this can be seen in the soccer image, Figure 3.1. Because there were 2 people in the frame, there would be 2 sets of keypoints, their coordinates and their respective pose scores. The keypoints, their coordinates and the corresponding pose scores are shown in the terminal output.

A key step toward understanding people in images and video is accurate pose estimation. Given a single RGB image, we wish to determine the precise pixel location of important keypoints of the body. Achieving an understanding of a person's posture and limb articulation is useful for higher level tasks like action recognition, and also serves as a fundamental tool in fields such as human-computer interaction and animation.

As a well established problem in vision, pose estimation has plagued researchers with a variety of formidable challenges over the years. A good pose estimation system must be robust to occlusion and severe deformation, successful on rare and novel poses, and invariant to changes in appearance due to factors like clothing and lighting.
The combined vectors of all the frames in a particular video were then passed on to to the auto-encoder with their corresponding labels.


poorvi@poorvi-X556UQK: ~/Desktop/btp_ganeshsir/image_posenet/...

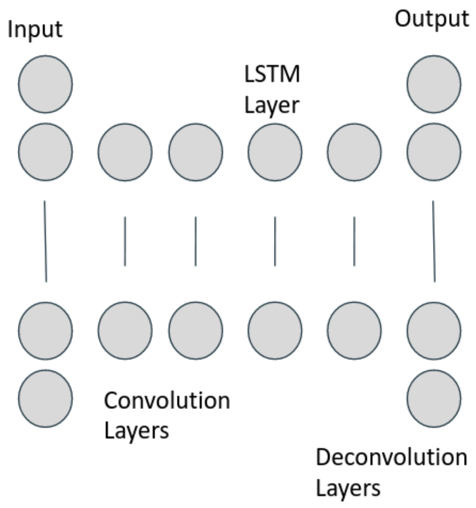
```
Results for image: ./image/soccer.png
Pose #0, score = 0.785683
Keypoint nose, score = 0.999120, coord = [ 46.66097403 308.00786638]
Keypoint leftEye, score = 0.997955, coord = [ 40.26838064 316.74228525]
Keypoint rightEye, score = 0.995551, coord = [ 39.2507267  300.95543242]
Keypoint leftEar, score = 0.890618, coord = [ 47.88030815 329.68287945]
Keypoint rightEar, score = 0.586088, coord = [ 46.51622152 291.97311735]
Keypoint leftShoulder, score = 0.994954, coord = [ 98.01967692 346.67434406]
Keypoint rightShoulder, score = 0.985343, coord = [100.87524652 279.13926888]
Keypoint leftElbow, score = 0.051554, coord = [150.68508148 378.98031044]
Keypoint rightElbow, score = 0.500874, coord = [163.34268427 268.83615112]
Keypoint leftWrist, score = 0.918821, coord = [150.20156574 349.21770859]
Keypoint rightWrist, score = 0.907470, coord = [165.3391304  272.12240243]
Keypoint leftHip, score = 0.846533, coord = [216.47590446 348.42047119]
Keypoint rightHip, score = 0.950782, coord = [211.2053225  292.58644867]
Keypoint leftKnee, score = 0.977926, coord = [308.6314168  341.22561455]
Keypoint rightKnee, score = 0.941507, coord = [295.543993   293.90937328]
Keypoint leftAnkle, score = 0.331088, coord = [350.1701827  370.25392342]
Keypoint rightAnkle, score = 0.480419, coord = [382.68495786 297.88106918]
Pose #1, score = 0.498082
Keypoint nose, score = 0.992764, coord = [ 98.22022247 150.47840405]
Keypoint leftEye, score = 0.987076, coord = [ 87.32186699 157.25323248]
Keypoint rightEye, score = 0.401103, coord = [ 87.72028542 151.34714508]
Keypoint leftEar, score = 0.958870, coord = [ 91.22385788 188.23673916]
Keypoint rightEar, score = 0.005042, coord = [ 90.35656357 153.97280979]
Keypoint leftShoulder, score = 0.797336, coord = [128.64626884 209.65584373]
Keypoint rightShoulder, score = 0.063317, coord = [137.69175053 190.40807724]
Keypoint leftElbow, score = 0.648115, coord = [196.14074993 179.83028412]
Keypoint rightElbow, score = 0.598974, coord = [190.88156796 158.41523981]
Keypoint leftWrist, score = 0.372268, coord = [230.36682224 129.31044006]
Keypoint rightWrist, score = 0.422081, coord = [232.21937943 132.44964695]
Keypoint leftHip, score = 0.854333, coord = [258.09700489 191.69067371]
Keypoint rightHip, score = 0.066723, coord = [254.09300518 157.80045509]
Keypoint leftKnee, score = 0.567634, coord = [369.56091309 183.84560013]
Keypoint rightKnee, score = 0.563907, coord = [345.87997913  94.48619652]
Keypoint leftAnkle, score = 0.036074, coord = [452.47214508 231.71699524]
Keypoint rightAnkle, score = 0.131781, coord = [414.19413376  68.15143681]
Average FPS: 4.039319355643322
poorvi@poorvi-X556UQK:~/Desktop/btp_ganeshsir/image_posenet/original$
```

3.3 Autoencoder Model

We use an autoencoder[6] to learn the non-anomalous behavior as autoencoders act as powerful feature detectors. The weightage given to anomalous data is set to be much less than the weightage given to non-anomalous data, while training the autoencoder. We experimented with various forms of the autoencoder by replacing fully connected layers with convolutional layers; adding LSTM, max-pooling layers; altering activation function etc. to find an autoencoder which gives the minimum reconstruction loss. The reconstruction loss also decreased substantially after normalising the vectors.

3.3.1 Architecture

We had split the data collected into training and test data. We then sorted the frames according to the reconstruction error. The 52 sized vectors of 20 subsequent frames



are concatenated and given as input to the autoencoder. More the reconstruction error, more is the anomaly in a frame. We analysed the top 5% frames with the maximum reconstruction loss and figured that those frames showed anomaly. Some of anomalous frames had lighting issues, therefore, the student is suggested to sit in a well lit area during the exam.

3.3.2 Results

We obtained the best results after using the shown autoencoder and ReLU activation function. After normalization final loss during training is of the order of 10ˆ-6 Frames with minimum reconstruction loss (in the order 10ˆ-3 to 10ˆ-2) were obtained from the non-anomalous videos while the ones with high reconstruction loss were from anomalous ones. The posenet and the test results for the series of frames shown in the beginning of the report are as follows:

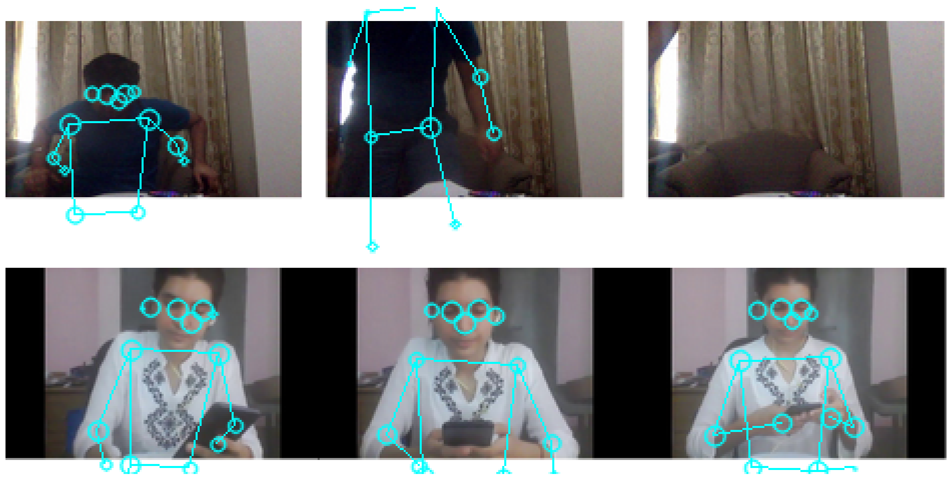


Figure 3.2: Test output label: Anomalous

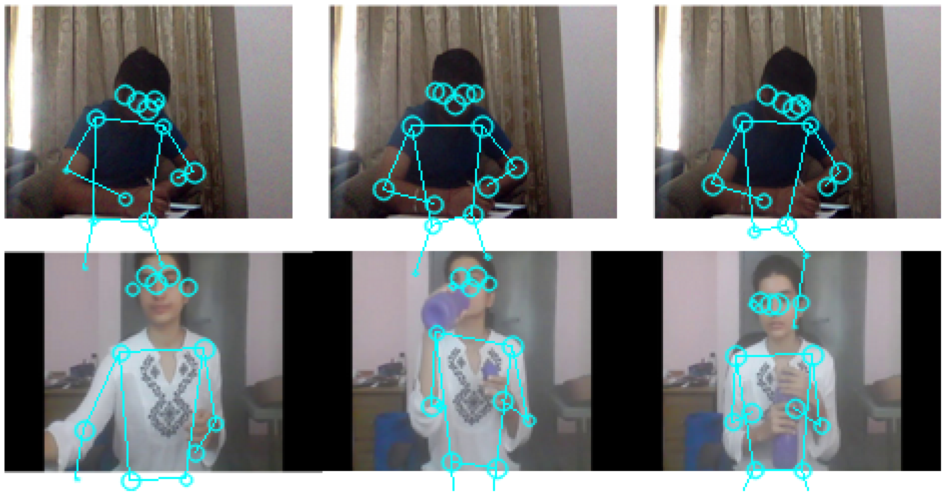


Figure 3.3: Test output label: Non-Anomalous

3.4 NTU-RGB Dataset

As the pose-estimation we did was dependant on features of the person like height, distance and orientation wrt camera, we needed a dataset which is person agnostic. Action Recognition NTU-RGB dataset[7] has a collection of 60 actions with around 56k samples and NTU-RGB 120[8] was its extended version and had a collection of 120 actions with 114k samples. These already have correct pose estimation and are person agnostic i.e we assumed that the ground truth per frame is scale normalised and thus the effect of the physique of the person being considered is minimal.

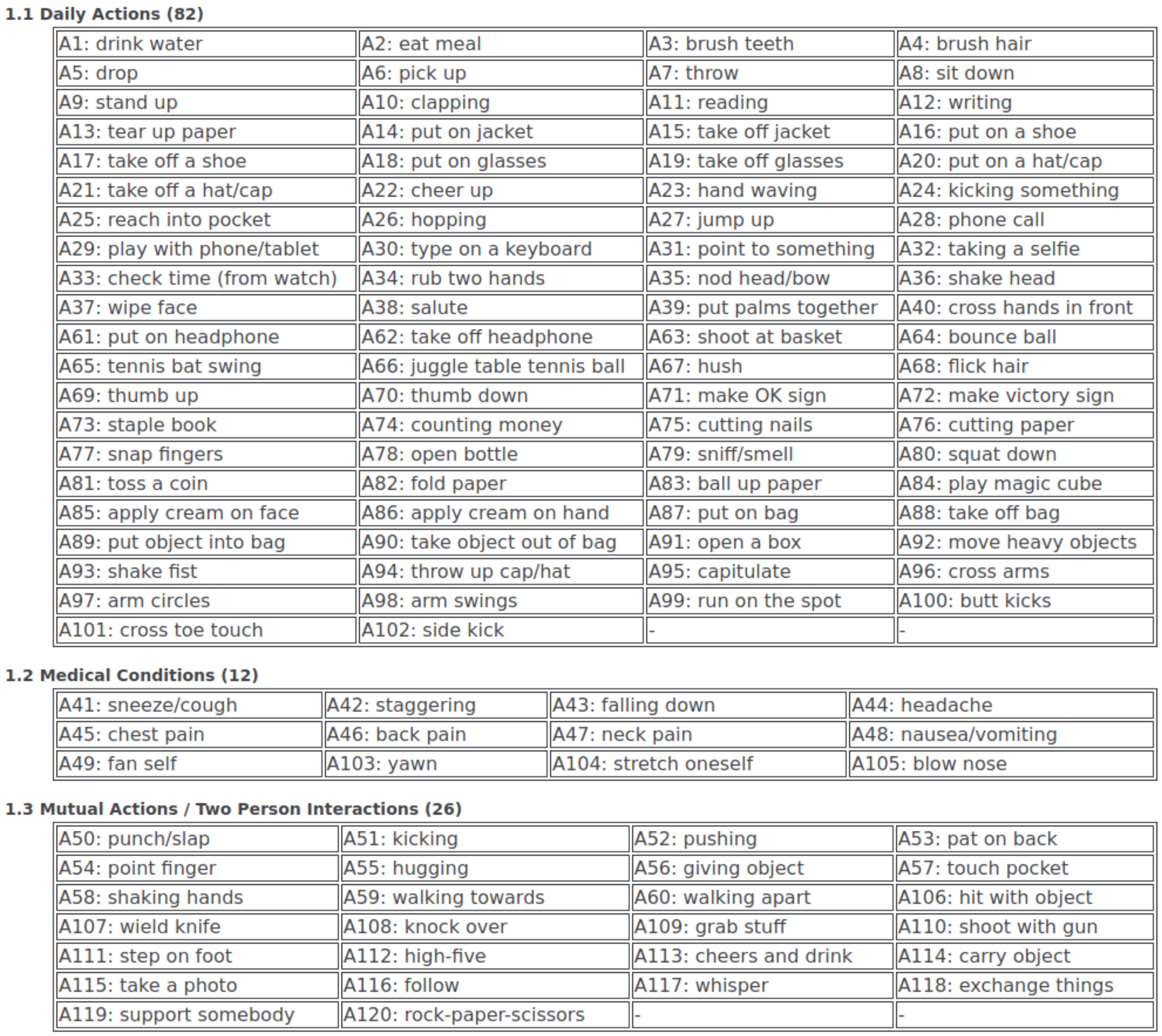
There were 4 different modalities or 4 different types of inputs observed

- **RGB:** The common images we see around us
- **Depth Map Sequences**: Value of a pixel relates to the distance from the camera
- **IR image:** Value of a pixel is determined by the amount of infrared light reflected back to the camera.
- **3D skeletal Data:** 3D coordinates of every single joint in the image

3.4.1 Actions and Rating Labels

Coming to the dataset actions, there were 3 broad classes: daily actions, mutual actions and medical conditions (Figure 3.4). A1 to A60 are contained in NTU-RGB dataset while A1-A120 are in its extended version nturgb 120. We have used the 60 actions NTU-RGB dataset for our project.

We map these actions to 5 ratings. Higher the rating of the action video, lower is its anomaly behaviour. For Example, All 2 person interactions are labelled 1 as they are anomalous in an exam scenario. On the other hand, A1: drink water, A11: reading, A12: writing are non-anomalous and are given a label 5. Similarly a few actions like A25: reach into pocket, A90: take object out of bag are slightly suspicious actions and can be given a lower rating while A9: standing up and A15: take off jacket are considerably less anomalous and could be given a higher rating.



1.1 Daily Actions (82)

A1: drink water	A2: eat meal	A3: brush teeth	A4: brush hair
A5: drop	A6: pick up	A7: throw	A8: sit down
A9: stand up	A10: clapping	A11: reading	A12: writing
A13: tear up paper	A14: put on jacket	A15: take off jacket	A16: put on a shoe
A17: take off a shoe	A18: put on glasses	A19: take off glasses	A20: put on a hat/cap
A21: take off a hat/cap	A22: cheer up	A23: hand waving	A24: kicking something
A25: reach into pocket	A26: hopping	A27: jump up	A28: phone call
A29: play with phone/tablet	A30: type on a keyboard	A31: point to something	A32: taking a selfie
A33: check time (from watch)	A34: rub two hands	A35: nod head/bow	A36: shake head
A37: wipe face	A38: salute	A39: put palms together	A40: cross hands in front
A61: put on headphone	A62: take off headphone	A63: shoot at basket	A64: bounce ball
A65: tennis bat swing	A66: juggle table tennis ball	A67: hush	A68: flick hair
A69: thumb up	A70: thumb down	A71: make OK sign	A72: make victory sign
A73: staple book	A74: counting money	A75: cutting nails	A76: cutting paper
A77: snap fingers	A78: open bottle	A79: sniff/smell	A80: squat down
A81: toss a coin	A82: fold paper	A83: ball up paper	A84: play magic cube
A85: apply cream on face	A86: apply cream on hand	A87: put on bag	A88: take off bag
A89: put object into bag	A90: take object out of bag	A91: open a box	A92: move heavy objects
A93: shake fist	A94: throw up cap/hat	A95: capitulate	A96: cross arms
A97: arm circles	A98: arm swings	A99: run on the spot	A100: butt kicks
A101: cross toe touch	A102: side kick	-	-

1.2 Medical Conditions (12)

A41: sneeze/cough	A42: staggering	A43: falling down	A44: headache
A45: chest pain	A46: back pain	A47: neck pain	A48: nausea/vomiting
A49: fan self	A103: yawn	A104: stretch oneself	A105: blow nose

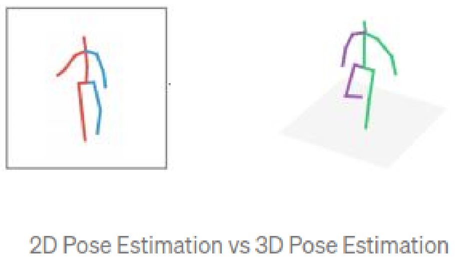
1.3 Mutual Actions / Two Person Interactions (26)

A50: punch/slap	A51: kicking	A52: pushing	A53: pat on back
A54: point finger	A55: hugging	A56: giving object	A57: touch pocket
A58: shaking hands	A59: walking towards	A60: walking apart	A106: hit with object
A107: wield knife	A108: knock over	A109: grab stuff	A110: shoot with gun
A111: step on foot	A112: high-five	A113: cheers and drink	A114: carry object
A115: take a photo	A116: follow	A117: whisper	A118: exchange things
A119: support somebody	A120: rock-paper-scissors	-	-

Figure 3.4: Actions

3.4.2 Parsing the Ground Truth

Each file/folder name in both datasets is in the format of SsssCcccPpppRrrrAaaa (e.g., S001C002P003R002A013), in which sss is the setup number, ccc is the camera ID, ppp is the performer (subject) ID, rrr is the replication number (1 or 2), and aaa is the action class label. We here try to extract just the relevant attributes which are 3d pose skeletal data and 2d pose skeletal data.[9]



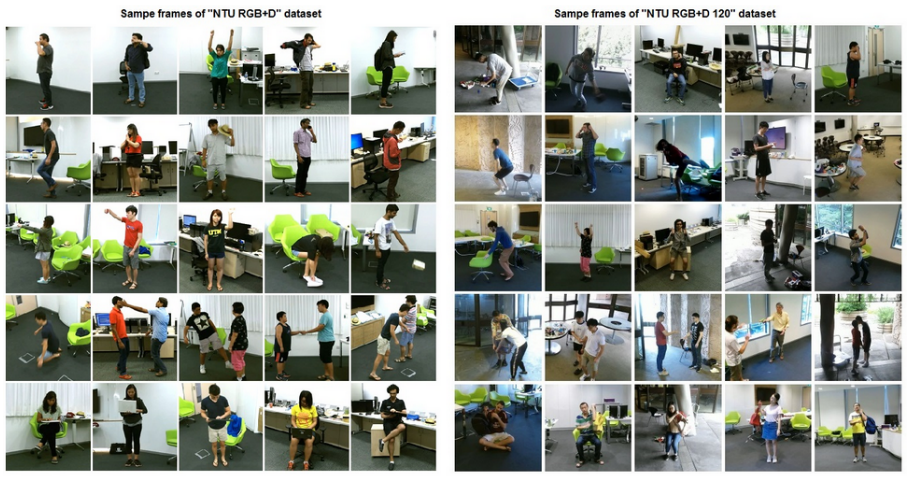

Sampe frames of "NTU RGB+D" dataset
Sampe frames of "NTU RGB+D 120" dataset

Figure 3.5: Sample Frames

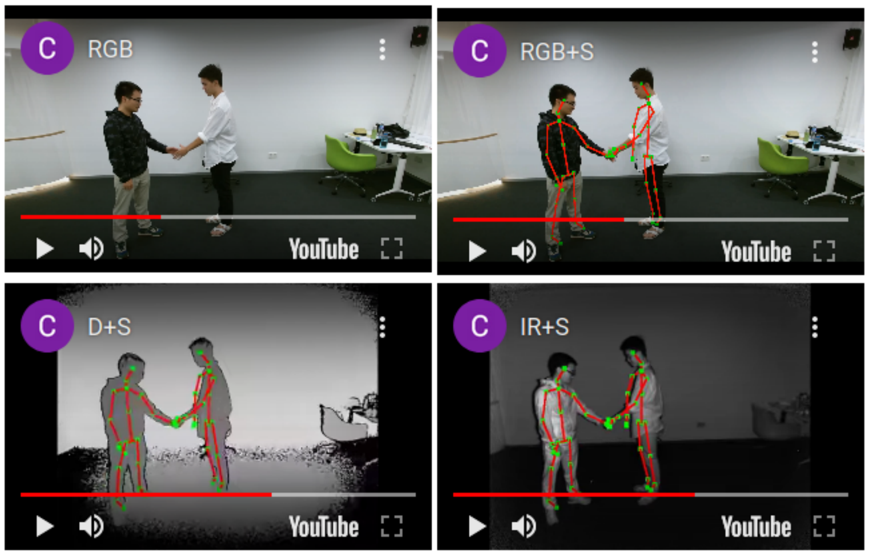


Figure 3.6: Sample Videos

Some sample frames and videos in the dataset are shown in Figure 3.5 and 3.6 respectively.

2D pose estimation simply estimates the location of keypoints in 2D space relative to an image or video frame. The model estimates an X and Y coordinate for each keypoint. 3D pose estimation works to transform an object in a 2D image into a 3D object by adding a z-dimension to the prediction which is estimated from the depth map. 3D pose estimation allows us to predict the actual spatial positioning of a depicted person or object.

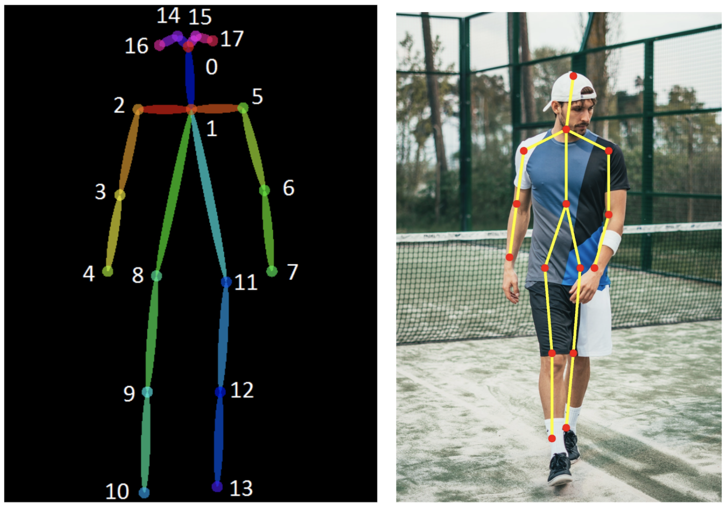


Figure 3.7: An example of Single Person Pose Estimation

Number of keypoints being considered for the ntu dataset is 25, so if the number of frames in the action video is f, then the size of the 3d skeletal data is f*25*3 while that of 2d is f*25*2. If there is more than 1 person in the image, sets of these data are collected. Corresponding to every action, we already had defined the rating and this would be the label of the video going into the autoencoder.

3.4.3 Results

For the NTU-RGB Dataset, we trained our autoencoder on the datasets with different labels separately and found that the reconstruction loss was lesser while training for the dataset which is non-anomalous. On the trained autoencoder, the error for the dataset with rating 1 (of the order 10ˆ-2) is more for the dataset with rating 5 (of the order 10ˆ-3). As the reconstruction loss is of the order 10ˆ-5 while training for this dataset, the task of finding better layers has to be done in future.

Chapter 4

Future Pose Prediction

4.1 Introduction

A vital part of Human Motion modelling is the task of future pose prediction. Pose prediction is to predict future poses given a window of previous poses. Machines that can observe and interact with moving humans, whether in actual or virtual surroundings, need to understand how people move. Because human motion is the product of both physical constraints as well as the intentions of humans, motion modeling is a complex task that should be ideally learned from observations.

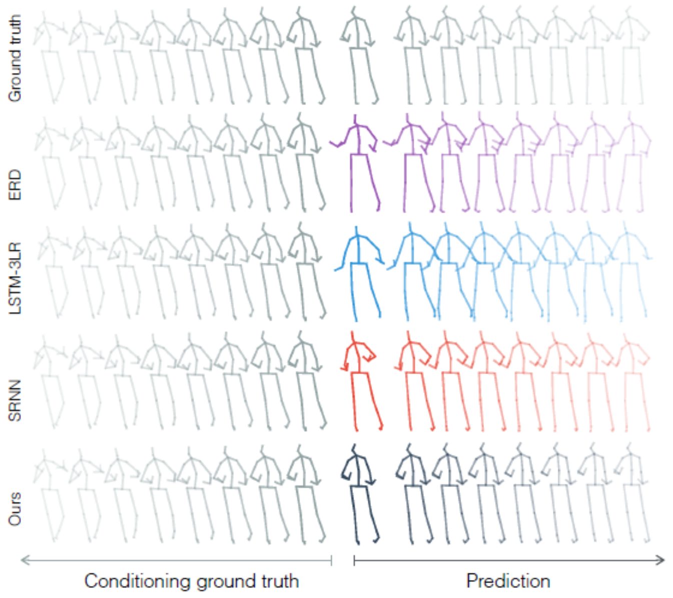


Figure 4.1: An example of models predicting human poses. The seed sequence is on the left, while the model predictions are on the right. Source: [31]

In this chapter, we try to predict 3D human poses in an auto-regressive fashion, i.e. given their previous motion information, we try to predict the next pose, and then feed this pose as input, and so on. Our approach is inspired by the recent success of transformer based models in NLP tasks such as language generation, which is similar to our task of future pose prediction. Language generation is to predict the next sequence of words given a seed sentence, which is infact quite similar to predicting poses. We therefore make use of OpenAI GPT-2 [39] architecture to model human motion. The GPT2 model is based on the decoder part of the transformer model. We use this instead of the BERT model which is based on the encoder because GPT2 uses masked self-attention. As a result, GPT2 computes attention only on timesteps before the current timestep, as opposed to BERT which computes attention over the entire input sequence. And since in motion modelling we want to predict the i^th pose conditioned on all timesteps before it, we use the GPT model.

4.2 Dataset

We use Human 3.6M dataset [13], which is the largest dataset on motion capture data. There are 3.6 million 3D human poses and images corresponding to 17 scenarios (discussion, smoking, taking photo, talking on the phone, etc..) We use the same pose representation as [[31], [28], [18]] and also evaluate using the same method, for fair comparison. Pose is represented as an exponential map of each joint, with special processing of global translation and rotation [19]. Loss evaluation is done by computing euclidean distance between predicted and ground truth poses in angle-space. Although the number of dimensions in exponential map of a pose is 99, we model only 54 of them [31] because the rest of the dimensions either do not vary, or they vary by a very small fraction.

4.3 Related Work

Conventional techniques used in this problem commonly involve imposing knowledge about motion through Markovian assumptions([10], [36]), smoothness or low dimensional embeddings[27]. More recent attempts at solving this problem make use of deep learning based architectures, like RNN's, LSTM's and GRU's[[31], [28], [18]]. These temporal aggregators have not been known to perform well when tested on both qualitative as well as quantitative evaluations of their results. It is often seen that while trying to predict longer sequences of motion, certain unrealistic poses tend to slip in, while in short-term results, there exists a clear discontinuity between the seed sequence and the predicted poses(figure 4.1). With our GPT2 based model, we hope to predict more realistic poses over a longer time horizon, thanks to the self-attention feature of the transformer-decoder.

4.4 Our Model

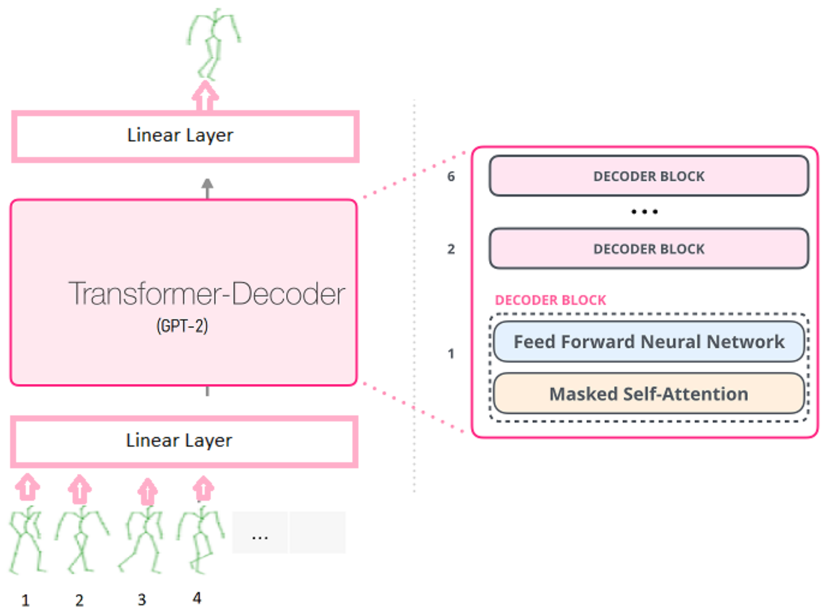


Figure 4.2: A picture to demonstrate the working of our Pose-GPT model. The seed sequence(poses) is fed into the OpenAI GPT-2 model, Source: this blog

We make use of the OpenAI GPT2 [39] model architecture for modelling human motion. We test 2 variants of the model, one with 2 and the other with 4 decoder layers. Embedding dimension used is 768, and the input pose is transformed through a linear layer into the pose embedding. Another linear layer on the output of the GPT2 model transforms the predicted pose embedding into the predicted pose.

4.5 Performance Metrics

Pose estimation requires a different set of metrics to evaluate its performance as compared to other vision tasks. They are as follows:

1. **Percentage of Correct Keypoints(PCK):**

   This metric calculates the percentage of correctly identified joints. A joint is correctly inferred if it lies within a certain threshold distance from the ground truth joint position.

2. **Percentage of Correct Keypoints head (PCKh):**

   This metric also calculates the percentage of correctly identified joints, but the difference with PCK is that here the threshold distance is relative to the length of the head bone link(distance between head and neck joint). A common metric used is PCKh@0.5, which specifies the threshold distance as half of the head bone link.

3. **Mean Per Joint Position Error (MPJPE):**

   This metric is used in 3D pose problems. It is the mean error over all the joint locations, and the error is the euclidean distance between ground truth and predicted joint location. Since 3D pose is in an unconstrained space, the ground truth pose is first aligned with the predicted pose (generally at the pelvis), after which the error is calculated. The root mentioned in the formula is the pelvis joint.

$$MPJPE(\hat{x}, x) = \frac{1}{N}\sqrt{\sum_i ((\hat{x}_{root} - \hat{x}_i) - (x_{root} - x_i))^2}$$

4. **Mean Average Precision (mAP):**

   Datasets like COCO require results to be declared on mAP, which is usually done for object detection tasks. mAP is the average of P scores(Precision) over different IoU thresholds, generally from 0.5 to 0.95 at jumps of 0.05. This accuracy metric is not strictly used for pose tasks.

5. **OKS based Mean Average Precision (mAP):**

   Object Keypoint Similarity (OKS) is the pose equivalent of IoU in object detection. It determines the level of similarity between ground truth pose and the predicted pose. It is scored as follows:

$$OKS = \frac{\sum_i \exp(\frac{-d_i^2}{2s^2k_i^2})\delta(v_i > 0)}{\sum_i \delta(v_i > 0)}$$

   Here d_i is the euclidean distance between the ground truth location and predicted location, v_i is the level of visibility of that joint, s is the object scale and k_i is a per keypoint constant that controls falloff.

4.6 Experiments

These experiments are mainly performed by Srijon and have been included in this report for reference. For fair comparison, we keep the setting of the experiment same as that of [31]. The seed sequence is kept to be 2 sec, and for short term prediction we predict upto 400ms, and for longer prediction we predict upto 1 sec. We train for 1000 epochs with a learning rate of 5e-5 and use Adam optimizer. Loss used here is simple Euclidean loss between predicted pose and ground truth pose. The results for the same are tabulated in the table below.

Motion	Walking				Eating				Smoking				Discussion			
milliseconds	80	160	320	400	80	160	320	400	80	160	320	400	80	160	320	400
ZeroV [[31]]	0.39	0.68	0.99	1.15	0.27	0.48	0.73	0.86	0.26	0.48	0.97	0.95	0.31	0.67	0.94	1.04
ERD [[18]]	0.93	1.18	1.59	1.78	1.27	1.45	1.66	1.80	1.66	1.95	2.35	2.42	2.27	2.47	2.68	2.76
Lstm3LR [[18]]	0.77	1.00	1.29	1.47	0.89	1.09	1.35	1.46	1.34	1.65	2.04	2.16	1.88	2.12	2.25	2.23
SRNN [[28]]	0.81	0.94	1.16	1.30	0.97	1.14	1.35	1.46	1.45	1.68	1.94	2.08	1.22	1.49	1.83	1.93
DropAE [[21]]	1.00	1.11	1.39	/	1.31	1.49	1.86	/	0.92	1.03	1.15	/	1.11	1.20	1.38	/
Samp-loss [[31]]	0.92	0.98	1.02	1.20	0.98	0.99	1.18	1.31	1.38	1.39	1.56	1.65	1.78	1.80	1.83	1.90
Res-sup [[31]]	0.27	0.46	0.67	0.75	0.23	0.37	0.59	0.73	0.32	0.59	1.01	1.10	0.30	0.67	0.98	1.06
CSM [[29]]	0.33	0.54	0.68	0.73	0.22	0.36	0.58	0.71	0.26	0.49	0.96	0.92	0.32	0.67	0.94	1.01
TP-RNN [[14]]	0.25	0.41	0.58	0.65	0.20	0.33	0.53	0.67	0.26	0.47	0.88	0.90	0.30	0.66	0.96	1.04
QuaterNet [[15]]	0.21	0.34	0.56	0.62	0.20	0.35	0.58	0.70	0.25	0.47	0.93	0.90	0.26	0.60	0.85	0.93
AGED [[23]]	0.21	0.35	0.55	0.64	0.18	0.28	0.50	0.63	0.27	0.43	0.81	0.83	0.26	0.56	0.77	0.84
BiHMP-GAN [[26]]	0.33	0.52	0.63	0.67	0.20	0.33	0.54	0.70	0.26	0.50	0.91	0.86	0.33	0.65	0.91	0.95
Skel-TNet [[24]]	0.31	0.50	0.69	0.76	0.20	0.31	0.53	0.69	0.25	0.50	0.93	0.89	0.30	0.64	0.89	0.98
VGRU-r1 [[22]]	0.34	0.47	0.64	0.72	0.27	0.40	0.64	0.79	0.36	0.61	0.85	0.92	0.46	0.82	0.95	1.21
Sym-GNN [[30]]	0.17	0.31	0.50	0.60	0.16	0.29	0.48	0.60	0.21	0.40	0.76	0.80	0.21	0.55	0.77	0.85
Pose-GPT	0.393	0.537	0.636	0.696	0.356	0.443	0.604	0.736	0.447	0.636	0.959	0.971	0.439	0.731	1.003	1.061

Table 4.1: Comparisons of MAEs between Pose-GPT and state-of-the-art methods for short-term motion prediction on 4 representative actions of H3.6M(walking, eating, smoking, discussion)

Motion	Walking		Eating		Smoking		Discussion	
milliseconds	560	1000	560	1000	560	1000	560	1000
ZeroV [[20]]	1.35	1.32	1.04	1.38	1.02	1.69	1.41	1.96
ERD [[18]]	2.00	2.38	2.36	2.41	3.68	3.82	3.47	2.92
Lstm3LR [[18]]	1.81	2.20	2.49	2.82	3.24	3.42	2.48	2.93
SRNN [[28]]	1.90	2.13	2.28	2.58	3.21	3.23	2.39	2.43
DropAE [[21]]	1.55	1.39	1.76	2.01	1.38	1.77	1.53	1.73
Res-sup. [[31]]	0.93	1.03	0.95	1.08	1.25	1.50	1.43	1.69
CSM [[29]]	0.86	0.92	0.89	1.24	0.97	1.62	1.44	1.86
TP-RNN [[14]]	0.74	0.77	0.84	1.14	0.98	1.66	1.39	1.74
AGED [[23]]	0.78	0.91	0.86	0.93	1.06	1.21	1.25	1.30
BiHMP-GAN [[26]]	/	0.85	/	1.20	/	1.11	/	1.77
Skel-TNet [[24]]	0.79	0.83	0.84	1.06	0.98	1.21	1.19	1.75
Sym-GNN [[30]]	0.75	0.78	0.77	0.88	0.92	1.18	1.17	1.28
Pose-GPT	0.76	0.84	0.89	1.27	1.04	1.62	1.38	1.71

Table 4.2: Comparisons of MAEs across models for longer motions(>400ms) on the 4 representative actions of H3.6M.

While our model does not perform the best at short time intervals(refer to table 4.1), it is competitive at the longer time intervals (refer to table 4.2). On the 4 representative actions of Human 3.6M, our results on Walking action is the closest to the SOTA model on the longer time intervals (560ms and 1000ms). We include results for the other 11 actions in table 4.3 for short time intervals. The better numbers on longer intervals could be attributed to the self-attention feature of GPT model that allows it to generate more plausible poses on the long run.

4.7 Conclusion

This chapter included a set of experiments on the task of future pose prediction. We achieve competitive results on the longer time intervals actions in Human 3.6M, but fall short on on the smaller intervals test. The initial hypothesis of self attention to aid pose prediction over larger time intervals is mildly corroborated by the results, although further investigation into the attention weights will provide useful insight

Motion	Directions				Greeting				Phoning				Posing			
milliseconds	80	160	320	400	80	160	320	400	80	160	320	400	80	160	320	400
ZeroV [[20]]	0.39	0.59	0.79	0.89	0.54	0.89	1.30	1.49	0.64	1.21	1.65	1.83	0.28	0.57	1.13	1.37
Res-sup [[18]]	0.41	0.64	0.80	0.92	0.57	0.83	1.45	1.60	0.59	1.06	1.45	1.60	0.45	0.85	1.34	1.56
CSM [[29]]	0.39	0.60	0.80	0.91	0.51	0.82	1.21	1.38	0.59	1.13	1.51	1.65	0.29	0.60	1.12	1.37
TP-RNN [[14]]	0.38	0.59	0.75	0.83	0.51	0.86	1.27	1.44	0.57	1.08	1.44	1.59	0.42	0.76	1.29	1.54
AGED [[23]]	0.23	0.39	0.62	0.69	0.54	0.80	1.29	1.45	0.52	0.96	1.22	1.43	0.30	0.58	1.12	1.33
Skel-TNet [[24]]	0.36	0.58	0.77	0.86	0.50	0.84	1.28	1.45	0.58	1.12	1.52	1.64	0.29	0.62	1.19	1.44
Sym-GNN [[30]]	0.23	0.42	0.57	0.65	0.35	0.60	0.95	1.15	0.48	0.80	1.28	1.41	0.18	0.45	0.97	1.20
Pose-GPT	0.60	0.80	0.85	0.94	0.64	0.92	1.28	1.45	0.74	1.03	1.58	1.70	0.74	0.90	1.50	1.77

Motion	Purchases				Sitting				Sitting down				Taking photo			
ZeroV [[20]]	0.62	0.88	1.19	1.27	0.40	1.63	1.02	1.18	0.39	0.74	1.07	1.19	0.25	0.51	0.79	0.92
Res-sup [[18]]	0.58	0.79	1.08	1.15	0.41	0.68	1.12	1.33	0.47	0.88	1.37	1.54	0.28	0.57	0.90	1.02
CSM [[29]]	0.63	0.91	1.19	1.29	0.39	0.61	1.02	1.18	0.41	0.78	1.16	1.31	0.23	0.49	0.88	1.06
TP-RNN [[14]]	0.59	0.82	1.12	1.18	0.41	0.66	1.07	1.22	0.41	0.79	1.13	1.27	0.26	0.51	0.80	0.95
AGED [[23]]	0.46	0.78	1.00	1.07	0.41	0.75	1.04	1.19	0.33	0.61	0.97	1.08	0.23	0.48	0.81	0.95
Skel-TNet [[24]]	0.58	0.84	1.17	1.24	0.40	0.61	1.01	1.15	0.37	0.72	1.05	1.17	0.24	0.47	0.78	0.93
Sym-GNN [[30]]	0.40	0.60	0.97	1.04	0.24	0.41	0.77	0.95	0.28	0.60	0.89	0.99	0.14	0.32	0.53	0.64
Pose-GPT	0.73	0.92	1.24	1.31	0.71	0.91	1.24	1.40	0.95	1.18	1.43	1.54	0.44	0.66	0.94	1.05

Motion	Waiting				Walking Dog				Walking Together				Average			
ZeroV [[20]]	0.34	0.67	1.22	1.47	0.60	0.98	1.36	1.50	0.33	0.66	0.94	0.99	0.39	0.77	1.05	1.21
Res-sup. [[18]]	0.32	0.63	1.07	1.26	0.52	0.89	1.25	1.40	0.27	0.53	0.74	0.79	0.40	0.69	1.04	1.18
CSM [[29]]	0.30	0.62	1.09	1.30	0.59	1.00	1.32	1.44	0.27	0.52	0.71	0.74	0.38	0.68	1.01	1.13
TP-RNN [[14]]	0.30	0.60	1.09	1.28	0.53	0.93	1.24	1.38	0.23	0.47	0.67	0.71	0.37	0.66	0.99	1.11
AGED [[23]]	0.25	0.50	1.02	1.12	0.50	0.82	1.15	1.27	0.23	0.42	0.56	0.63	0.33	0.58	0.94	1.01
Skel-TNet [[24]]	0.30	0.63	1.17	1.40	0.54	0.88	1.20	1.35	0.27	0.53	0.68	0.74	0.36	0.64	0.99	1.02
Sym-GNN [[30]]	0.22	0.48	0.87	1.06	0.42	0.73	1.08	1.22	0.16	0.33	0.50	0.56	0.26	0.49	0.79	0.92
Pose-GPT	0.48	0.72	1.14	1.31	0.67	0.91	1.24	1.34	0.39	0.58	0.75	0.78	0.59	0.79	1.09	1.20

Table 4.3: Comparisons of MAEs of Pose-GPT vs other methods for short motions on the remaining 11 actions of H3.6M.

regarding the short time interval results. We leave further experiment and ablation studies to future works.

Chapter 5

Self Supervised Pose Estimation

5.1 Introduction

Neural networks benefit from large quantities of labeled training data, i.e Deep learning methods are always data intensive, and the first step to building a good model for 3D pose estimation would require large amount of 3D pose annotated data.This however, has always been lacking. In many settings labeled data is much harder to come by than unlabeled data, 3D pose annotation requires the use of costly motion capture devices which makes it less accessible. Due to this, in-the-wild pose-annotated datasets do not exist, and hence most works in this field make use of 2D in-the-wild data and 3D datasets like Human3.6M and MPI-INF. These 3D pose annotated datasets are set in controlled environments and hence don't generalize well to real-world settings.

Self-supervised training is a topic which might lend itself useful to this problem. In this regime, training is split into 2 halves, pre-training and fine-tuning. In the pre-training stage, no labelled data is required and the model learns a feature representation from the data itself. In the fine-tuning stage, this representation is training against labelled data for whatever downstream task is applicable for the problem. Self-supervised training has led to great success in the field of Natural Language Processing, where benchmarks have been set using magnitudes lesser labelled data.

5.2 Related Work

Self-Supervised Learning: Multiple works in NLP have used this training regime to success with transformer models [[17], [37], [38]]. With BERT [[17]], a simple pre-training method allowed it to learn a word representation strong enough for multiple downstream tasks. In speech domain, [12] makes use of self-supervision to learn speech representations from thousands of hours of unlabelled data. In the vision domain, [40, 25, 35] use self-supervision to learn a geometry aware embedding in their pre-training stage, and later finetune with limited 3D data. All these methods make use of multi-view synchronized videos of human motion. Our approach is similar to these as we don't use an 2D data, but in contrast we only train on monocular videos without any annotations, which is abundantly available.

3D Pose Estimation: Most supervised methods utilise both 2D and 3D pose annotated data to learn a mapping between 2D pose and 3D pose[[32, 33, 11, 16, 41]]. These generally try to overcome the lack of large scale in-the-wild 3D pose annotated data, by using 2D real world data and learning a complimentary network for lifting/ predicting the depth of the poses. These methods require large amount of 2D annotated data as well as some 3D annotated data, while we use only small amount of 3D labels.

Negative mining: When using contrastive loss or its variants like InfoNCE and metric learning, selecting good negatives becomes imperative to learn a good feature representation. [35] uses the Hardnet framework [34] to perform hard negative mining; essentially it maximises the distance between the closest positive and closest negative sample in the batch. Since we work with videos, we choose negatives from the same. We discuss our methods for negative mining in the Our Model section.

5.3 Wav2vec2 model

We utilise self-supervision to learn a pose representation from monocular videos without annotations, and then perform fine-tuning with labelled data. Our work is inspired by the success of wav2vec2 ([12]) in the speech domain, where self supervision along with contrastive loss allows it to set a new benchmark in speech recognition. This paper proved that learning powerful representations from speech audio alone followed by fine-tuning on transcribed speech can outperform the best semi-supervised methods while being conceptually simpler. Wav2vec 2.0 masks the speech input in the latent space and solves a contrastive task defined over a quantization of the latent representations which are jointly learned.

In here, the pre-training task becomes that of "contrasting" the true latent against that of multiple negatives, which are basically embeddings representing some other speech signals. We can use either the poses or the images as input for our model. In each case, we again have the option of using or abandoning the quantizer. The models with poses as input with and without a quantizer are demonstrated as follows:

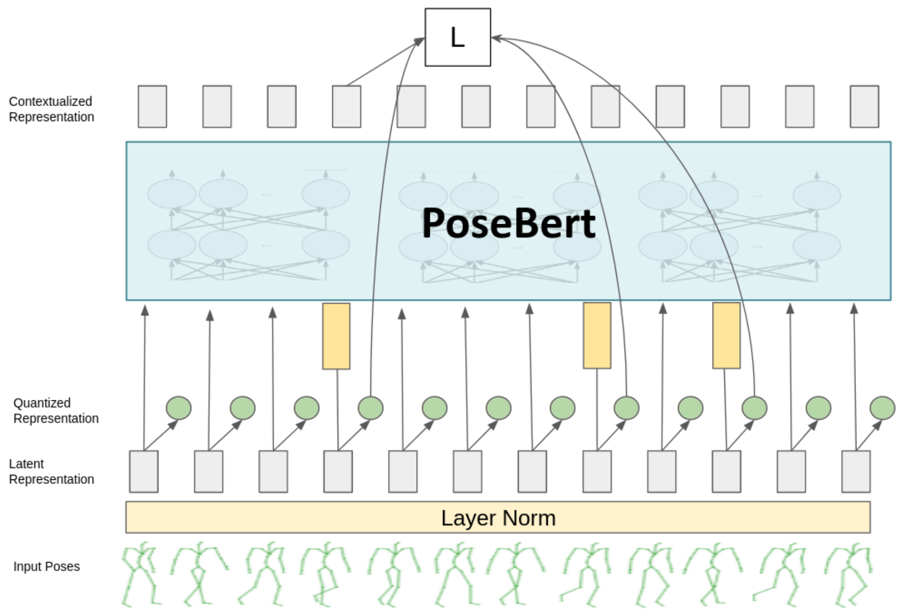


Figure 5.1: Pose Input with quantizer

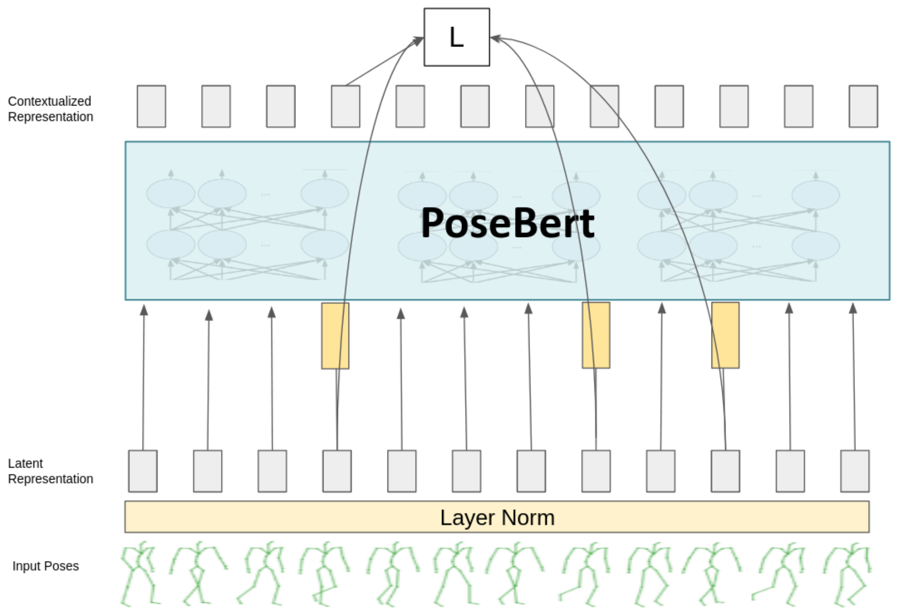


Figure 5.2: Pose Input with no quantizer

5.4 Our Model

Our model consists of a pretrained Resnet-50 feature encoder which takes as input raw images and outputs latent representations for each time-step. They are then fed to a Transformer-Encoder network to build representations capturing information from the entire sequence. The output of the feature encoder is discretized with a quantization module to represent the targets in the self-supervised objective.

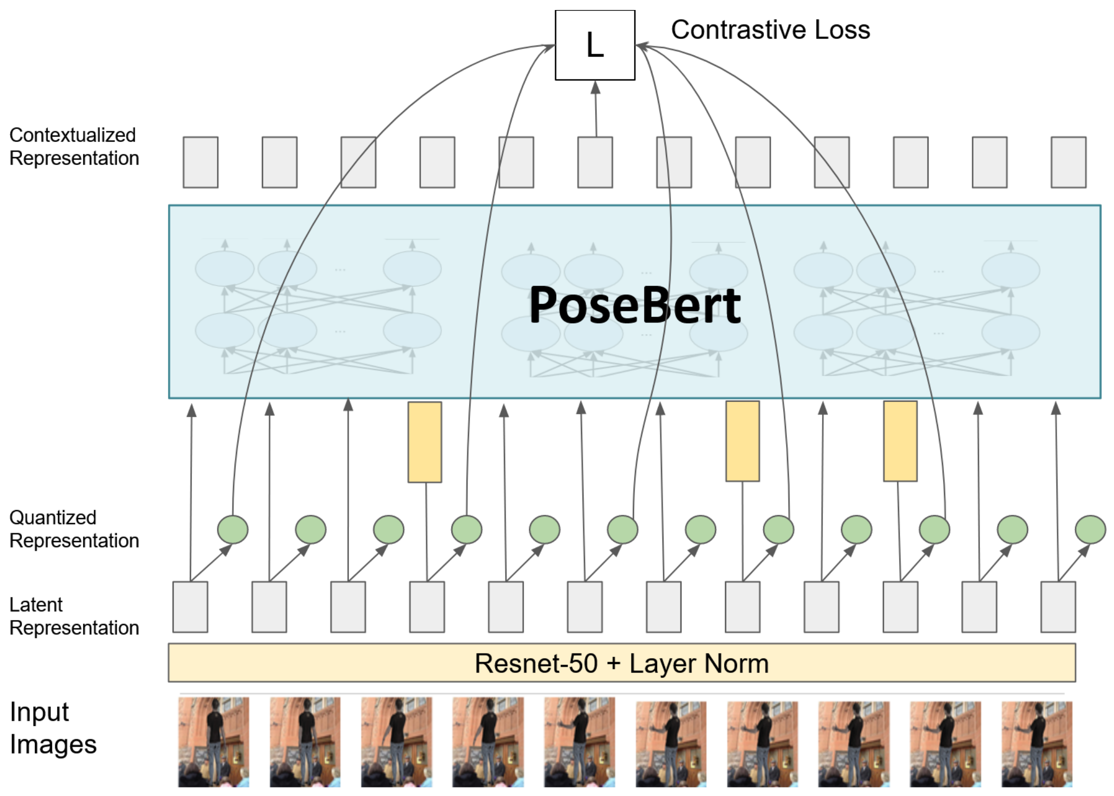


Figure 5.3: A picture to demonstrate the working of our PoseBERT model. The images are passed through a pretrained Resnet-50 feature extractor to obtain the latent representation. The objective of contrastive loss is to differentiate between the quantizer entry for the anchor image against the negatives sampled from the same. We try multiple methods of sampling negatives.

Around 30% of the frames in an input video are masked before being passed into the Transformer-Encoder network. We make use of relative positional embedding, instead of absolute positional embedding which encodes absolute positions. This is implemented via a sliding 1-D convolution layer.
We use product quantization to discretize the output of the feature encoder into a limited set of representations for self-supervised training, just as in [12]. Basically, product quantization chooses quantized representations from the codebooks and concatenates them. Although we can have multiple codebooks, we use only one. Choosing the quantized representation is done via Gumbel softmax, and is fully differentiable. For a codebook g, and a representation v, the probability of being selected is as follows:

$$p_{g,v} = \frac{exp(l_{g,v} + n_v)/\tau}{\Sigma_{k=1}^{V} exp(l_{g,k} + n_k)/\tau}$$

Here, τ is a non-negative temperature, $n = -\log(-\log u)$ and u are uniform samples from $U(0,1)$.

In order to sample negatives from the video, there are multiple options. **?** uses random sampling of negatives from a temporal sequence, however that is acceptable in speech since speech tokens do not repeat frequently. Hence, the chosen negative is more likely to represent a different speech signal. However, in our task, random sampling may not work the same as the set of poses in a video may be limited, and randomly picking timesteps may lead to similar poses being contrasted against each other.

Due to this, we need to use something to pick dissimilar pose timesteps. The simplest way to due it would be to use the ground truth 3D pose. But then we would be using the pose annotations even during pre-training. Another way would be to use cosine similarity between the frames of a video to pick dissimilar poses. This assumes that the camera angle is able to capture variation in poses of the subject in question. This however, requires no pose annotations, and can work monocular videos. We try both of the negative selection methods and compare them in the experiments section.

The loss used for training is contrastive loss:

$$L_m = -\log \frac{exp(sim(c_t, q_t)/\kappa)}{\Sigma_{\tilde{q}\sim Q_t} exp(sim(c_t, \tilde{q}/\kappa))}$$

Here, $\tilde{q}$ are distractors sampled from the quantizer representations. The c_t is the anchor embedding, and it needs to identify the true latent quantized representation from the distractors.

Apart from contrastive loss, we also use a diversity loss which ensures that maximum entries from the codebook are utilised. The diversity loss is implemented as follows:

$$L_d = \frac{GV - \Sigma_{g=1}^{G} exp(-\Sigma_{v=1}^{V} p_{gv} \log p_{gv})}{GV}$$

Here, G is number of codebooks, and V stands for number of codebook entries. $p_g v$ is the probability of selecting the $v^t h$ entry from $g^t h$ codebook.

So the total loss is

$$L = L_m + 0.1 * L_d$$

5.5 Datasets

5.5.1 Human3.6M

We use the Human3.6M dataset for both pre-training and fine-tuning tasks here. This dataset is ideal for us since it contains a large set of actors(7) performing diverse set of actions (11 actions). Although these videos are shot using multiple synchronized cameras, we don't make use of multi-view data, and simply rely on a single camera only. We used masked out images here, as we feel it is necessary for the encoder network as well as the contextual network to focus on the pose only.

5.5.2 MPI-INF

MPII Human Pose Dataset is a state of the art benchmark for evaluation of articulated human pose estimation. The dataset includes around 25k images containing over 40k people with annotated body joints. The images were collected from YouTube videos, covering daily human activities with complex poses and image appearances. The dataset covers 410 human activities and each image is provided with an activity label.

A few other relevant datasets are listed below (all of which haven't been used).

Dataset	3D Human Pose	Actions	Size
NTU	True	True	56k*100
Human3.6	True	False	3.12l
TikTok	False	False	1l
MPI-3D HP	True	False	6l
Kinetics	False	True	400k*100
VidHOI	False	Hoi	7.3M frames

5.6 Experiments

In this section we consider the retrieval and the pose estimation experiments on both the H3.6 and MPI-INF Datasets. We consider 4 sampling methods for distractors:

- Random sampling
- Pose-Based sampling: Sample the farthest n samples based on the MPJPE with the anchor.
- Distance-Based sampling: Uniform sampling from time steps which are at least D time steps away from the anchor. (Not yet implemented Yet)
- Cosine Similarity Based sampling: Use cosine similarity between image embeddings to determine dissimilar images from anchor, from which negatives are sampled.

5.6.1 BG Mask

Comparing the Top1 retieval MPJPE scores and the results for fine tuning for pose estimation with and without BG mask:

Negative Sampling Strategy	With BG Mask?	Top-1 Retrieval Score (MPJPE in mm)	Pose-Estimation (MPJPE in mm)
GT-pose	Yes	259	144
GT-pose	No	238	152
Image feature similarity	Yes	273	149
Image feature similarity	No	241	150
Random-Initialization	Yes	231	149
Random-Initialization	No	268**	153

Table 5.1: Comparisons of retrieval scores as well as fine-tuning for pose estimation task across multiple variants of PoseBERT on H3.6 (without quantizer). In GT-Pose, pose is used for sampling negatives, while in Image feature similarity, the cosine-similarity between frames is used.

We see that with background mask, retrieval scores are much better for each sampling strategy. (**except for the random initialization, experiment to be repeated). But for pose-estimation, we see that the our immediate pretraining (with contrastive loss) doesn't seem to help much. We also see that that random initialization yields better results after pretraining with contrastive loss. This might be because either the contrastive loss is not a representative of the retrieval objective or maybe because of some bad architecture/bug.

To verify if contrastive loss is not the right representative, we conducted another supervised experiment with just pose-based loss (no contrastive or diversity loss). A fresh BERT model was created and no pretraining was done. In this, the validation MPJPE for pose estimation came out to be 160.64mm which is quite high compared to the ones mentioned in 5.1.

For improving it's performance, we added a residual block and thus skipped the connection. We got an MPJPE of 164.06mm for the fine tuning task, which is not an improvement compared to 164.64. This might be due to overfitting owing to the lots of extra parameters introduced (embedding size=2048* num of joints=16* dimensions=3). More parameterization turned out to be ineffective. This thread of experiments need to be looked into, in the future so as to understand the unexpectated random-initialization scores.

5.6.2 Quantizer

In order to evaluate the contribution of the quantizer, we run experiments both with and without the it. Same frame from multiple views are considered positives while unrelated frames are negatives. When the quantizer is not present, the negatives are sampled from the output embeddings of the PoseBERT model.

Evaluation:

- Test the embeddings from the pre-training stage by using the retrieval metric.
- Construct a database of train video embeddings
- Retrieve the closest train embedding for each test embedding
- Compare their MPJPE scores and report fine-tuning results on the pose estimation tasks 5.2.

Our PoseBERT model consists of 4 layers of transformer-encoder layers, each of them having 4 attention heads and embedding dimension 2048. We training it using Adam optimizer, learning rate of 1e-4 with lr decay. We use 1 codebook in the quantizer experiments, with 1024 codebook entries.

Negative Sampling Strategy	Quantizer Used	Retrieval Score (MPJPE)	Pose-Estimation (MPJPE)
GT-pose	Yes	277	148
GT-pose	No	259	144
Cosine-sim	Yes	289	149
Cosine-sim	No	273	149
Random	No	231	149

Table 5.2: Comparisons of retrieval scores as well as fine-tuning for pose estimation task across multiple variants of PoseBERT on H3.6 (with BG mask). In GT-Pose, pose is used for sampling negatives, while in Cosine-sim, the cosine-similarity between frames is used

As we can see, the retrieval scores for our models are quite high. Also, we notice that the experiments using quantizer tend to perform worse than their counterparts without it. More importantly, we observe that the choice of pre-training does not affect the final metric of the fine-tuning task. The pose estimation results are very close to each other, irrespective of the choice of negative sampling of the use of quantizer.

5.6.3 Multiview Setting

In order to be comparable with other papers in the same space, we also apply our method to multi-view setting. In this case, the positive for the anchor is selected from another view of the same video. Since we work with Human 3.6M dataset, multiple views of the same video are available. The architecture for the same is given in Figure 5.4.

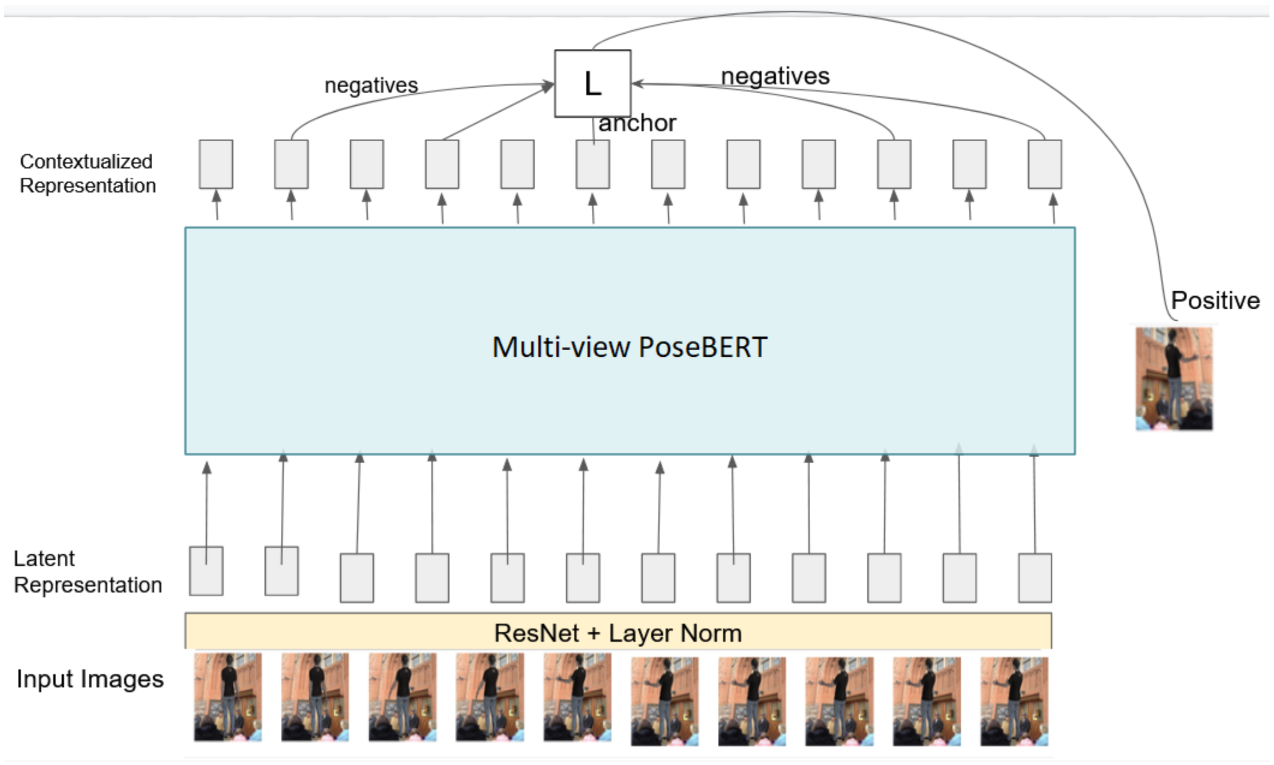


Figure 5.4: Our Multi-view PoseBERT model. Each batch contains multiple views of the same video. While the negatives are still being sampled from the same video, the positive sample is from another view of the anchor frame.

However, with multi-view information, our retrieval results actually get even worse (200mm with transformer and 233mm without transformer for pose-estimation task). We figure that the retrieval metric is not the ideal metric for evaluation in our case, since for 2 contextualized embeddings to be similar, not only do the both video sequences represented by the embedding need to have similar poses, they also need to have them in the same sequence. Now that is hard to achieve in H3.6M, as here the actors have a fair amount of leeway in acting out the actions. Hence, one actor's sequence of poses does not exactly match another actor, even for the same action. It is due to this reason that we feel the retreival MPJPE numbers are quite high.

5.6.4 MPI-Inf

In the retrieval experiment, given the pose embeddings, we need to retrieve the closest pose from the dataset.

Dataset	Input	Sampling Strategy (MPJPE)	Retrieval Score (MPJPE)
MPI	Pose	Uniform Sampling	152
H3.6	Pose	Uniform Sampling	135
MPI	Image	GT-pose based	332
H3.6	Image	GT-pose based	259
MPI	Image	Cosine-similarity	333
H3.6	Image	Cosine-similarity	273

Table 5.3: Comparisons of retrieval scores on Mpi-inf and H3.6 dataset

We see that MPI gives worse results as compared to H3.6 in case of Image as well as Pose inputs. Apart from this, not using the quantizer again gives better results on MPI dataset.

5.7 Conculsion

In this chapter we performed self-supervised learning of poses from monocular videos. We tested the embeddings from the contrastive loss based pre-training, followed by fine-tuning on pose estimation task. Quantization as well as multi-tasking seems to be backfiring. Therefore we need to investigate the initialization of the GT-pose based sampling with Image similarity based sampling's weights and vice-versa to understand the reason behind. Apart from this, using a BG mask seems to help for the retrieval scores. Also, we concluded that our immediate pretraining doesn't seem to help much in pose estimation. Apart from this we observed that Random Initialization yields better results after pretraining with contrastive loss. The reason behind the un-expected retrieval scores needs to be looked into, in the future.

Chapter 6

Conclusion and Future Scope

The first part of this report was on Use of AI and Video Analytics for Anomaly Detection in Proctored Videos. In an attempt to detect anomalous behaviours in online exams and implement a way of Automated Proctoring, we tried to design an auto-encoder model based on human pose features which would give an idea how anomalous the video clip is. We collected the Video datasets required, and after the required preprocessing, summarized the videos, detected the poses in frames, and finally trained an autoencoder using the labels. The results came out to be pretty good with reconstruction errors as low as 10^{-3}.

The next part of this report mainly contains a comprehensive summary of the different approaches taken towards Human Pose Estimation, along with recent developments using self-supervision. We did multiple experiments on Future pose prediction and acheived competitive results on the longer time intervals actions in Human 3.6M, but fell short on the smaller intervals test. Coming to self-supervised pose estimation from monocular videos, we realized that quantization as well as multi-tasking seems to be back firing, which needs to be looked into, in the future. We also concluded that our immediate pretraining doesn't seem to help much in pose estimation, which again through modification in experiments needs to be taken care of. While neither of the evaluations have led to State-of-the-Art numbers, it is possible to take our model performing relatively well for a particular task, and add a few modifications to it to enhance its performance.

Bibliography

[1] https://ieeexplore.ieee.org/document/5617655.

[2] http://www.cs.cornell.edu/~ashwin85/docs/frp0328-badanidiyuru.pdf.

[3] https://towardsdatascience.com/human-pose-estimation-simplified-6cfd88542ab3.

[4] https://github.com/rwightman/posenet-pytorch.

[5] https://github.com/poorvirhebbar/anomaly-detection-in-proctored-videos.

[6] https://colab.research.google.com/drive/19PLR-Bj7bBJ9G9gj6KP-Qqswc9uv4TCj?usp=sharing.

[7] https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Shahroudy_NTU_RGBD_A_CVPR_2016_paper.pdf.

[8] https://arxiv.org/pdf/1905.04757.pdf.

[9] https://github.com/shahroudy/NTURGB-D/tree/master/Python.

[10] P. V. Gehler A. M. Lehrmann and S. Nowozin. "A nonparametric bayesian network prior of human pose". In: *ICCV*. 2013.

[11] Mihai Zanfir Alin-Ionut Popa and Cristian Sminchisescu. "Deep multitask architecture for integrated 2d and 3d human sensing". In: *CVPR*. 2017.

[12] Baevski et al. "wav2vec 2.0: A framework for self-supervised learning of speech representations". In: *arXiv preprint arXiv:2006.11477* (2020).

[13] Vlad Olaru Catalin Ionescu Dragos Papava and Cristian Sminchisescu. "Human3.6M: Large Scale Datasets and Predictive Methods for 3D Human Sensing in Natural Environments". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2014.

[14] H. Chiu et al. "Actionagnostic human pose forecasting". In: *CoRR*. 2018.

[15] D. Grangier D. Pavllo and M. Auli. "Quaternet: A quaternion based recurrent model for human motion". In: *BMVC*. 2018.

[16] Chris Russell Denis Tome and Lourdes Agapito. "Lifting from the deep: Convolutional 3d pose estimation from a single image". In: *CVPR*. 2017.

[17] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: 2018. URL: https://arxiv.org/abs/1810.04805.

[18] K. Fragkiadaki et al. "Recurrent network models for human dynamics". In: *CVPR*. 2015.

[19] G. E. Hinton G. W. Taylor and S. T. Roweis. "Modeling human motion using binary latent variables". In: *NIPS*. 2006.

[20] G. E. Hinton G. W. Taylor and S. T. Roweis. "Modeling human motion using binary latent variables". In: *NIPS*. 2006.

[21] P. Ghosh et al. "Learning human motion models for long-term predictions". In: *CoRR*. 2018.

[22] A. Gopalakrishnan, A. Mali, and D. Kifer. "A neural temporal model for human motion prediction". In: *CVPR*. 2019.

[23] L. Gui et al. "Adversarial geometryaware human motion prediction". In: *ECCV*. 2018.

[24] X. Guo and J. Choi. "Human motion prediction via learning local structure representations and temporal dependencies". In: *AAAI*. 2019.

[25] Mathieu Salzmann Helge Rhodin and Pascal Fua. "Unsupervised geometry-aware representation for 3d human pose estimation". In: *ECCV*. 2018.

[26] M. Gor J. Kundu and R. Babu. "Bihmp-gan: Bidirectional 3d human motion prediction gan". In: *AAAI*. 2019.

[27] A. Hertzmann J. Wang and D. M. Blei. "Gaussian process dynamical models". In: *NIPS*. 2005.

[28] A. Jain et al. "Structural-RNN: Deep learning on spatio-temporal graphs". In: *CVPR*. 2016.

[29] C. Li et al. "Convolutional sequence to sequence model for human dynamics". In: *CVPR*. 2018.

[30] Maosen Li et al. "Symbiotic Graph Neural Networks for 3D Skeleton-based Human Action Recognition and Motion Prediction". In: *CVPR*. 2019.

[31] Martinez, Michael J. Black Julieta, and Javier Romero. "On human motion prediction using recurrent neural networks". In: *CVPR*. 2017.

[32] Julieta Martinez et al. "A simple yet effective baseline for 3d human pose estimation". In: *CVPR*. 2017.

[33] Dushyant Mehta et al. “Vnect:Real-time 3d human pose estimation with a single rgb camera”. In: *TOG*. 2017.

[34] Anastasiya Mishchuk et al. “Working hard to know your neighbor′s margins:Local descriptor learning loss”. In: *NIPS*. 2018.

[35] Rahul Mitra et al. “Multiview-Consistent Semi-Supervised Learning for 3D Human Pose Estimation”. In: *CVPR*. 2020.

[36] J. M. Rehg P. Pavlovic and J. MacCormick. “Learning switching linear models of human motion”. In: *NIPS*. 2000.

[37] M. E. Peters et al. “Deepcontextualized word representations”. In: *ACL*. 2018.

[38] A. Radford et al. “Improving language understanding by generative pre-training”. In: 2018.

[39] Alec Radford et al. “Language Models are Unsupervised Multitask Learners”. In: (2019).

[40] Helge Rhodin et al. “Neural scene decomposition for multi-person motion capture”. In: *CVPR*. 2018.

[41] Xiao Sun et al. “Integral human pose regression”. In: *ECCV*. 2018.

Appendix A

Video Summarization Techniques (BTP I)

The goal of Video summarization is to identify a small number of keyframes or video segments that contain as much information as possible of the original video. This step was included later as we noticed that pose estimation takes a lot of time and was a bottleneck for the entire process. Thus this step would summarize the entire video and give us a subset of frames which we would be interested in and which would contain a gist of the entire video i.e while containing as much info as possible. As the number of frames is decreased to a great extent, the pose estimation step would be a bit faster.

A.1 Asynchronous summarization

The entire video is available beforehand and any frame can be accessed in any order whatsoever. It's offline in nature and the summarization of the proctoring videos is done after the entire video is stored in a local machine. Previous work on asynchronous Video summarization includes:

- Video Maximal Marginal Relevance (Video-MMR) [1]
- Standard Greedy Algorithm

A.1.1 Video-MMR

If the summary is say S and the entire set of video frames is represented by V, then a distance measure can be introduced to quantify the relation between S and V or how well S represents V.

If fj is a frame belonging to V, we iterate over all frames g belonging to S, to maximise the similarity between fj and g. This frame g would be closest to fj and

$$d(S,V) = \frac{1}{n} \sum_{j=1}^{n} \min_{f_j \in V, g \in S} [1 - sim(f_j, g)] \qquad \hat{S} = \underset{S}{argmin}[d(S,V)]$$

then the 1 - their similarity would give some kind of a measure of the distance of fj from g. Extending it to the entire set, and averaging over all frames fj belonging to V, we get d(S,V). The best Summary would be the one that minimizes this quantity

In the proctoring scenario, our intuition is to construct the summary such that its visually similar to the content of the videos but at the same time it differs from the frames which are already selected in S. Following our intuition, we define a new measure Video-MR(fi) given by

$$\begin{aligned} Video\text{-}MR(f_i) = \lambda\ Sim_1(f_i, V \backslash S) \\ - (1-\lambda) \max_{g \in S} Sim_2(f_i, g) \end{aligned}$$

Maximising Video-mr(fi) would mean maximising the similarity between fi and the frames not yet selected while minimising the similarity between fi and the ones already selected. Therefore at every step, we chose fi which maximises this quantity. We can make a set S of some fixed cardinality by following the update rule shown here.

$$S_{k+1} = S_k \cup \underset{f_i \in V \backslash S_k}{argmax} \begin{pmatrix} \lambda\ Sim_1(f_i, V \backslash S_k) \\ - (1-\lambda) \max_{g \in S_k} Sim_2(f_i, g) \end{pmatrix}$$

where Sim1 is a similarity measure between 2 frames while sim2 is that for a frame and a set of frames. Sim2 can be considered as an average of sim1s between the frame being considered and every other frame in the set of frames. This average again can be of 2 types. Arithmetic mean or geometric mean and we will consider both the variants later on while testing their performance.

So the exact steps to be followed are shown here.

- S1 is initialised with a frame that has the best average similarity with all the other frames in the set. This frame would be that 1 frame which would be the most similar to the visual content of the video

$$f_1 = arg \max_{f_i, f_i \neq f_j} \left(\prod_{j=1}^{n} Sim(f_i, f_j)\right)^{\frac{1}{n}}$$

- We then find the frame which maximises Video-MR(fi), i.e is most similar to the rest of the video frames and most dissimilar to the frames already selected,

$$f_k = arg\ max_{f_i \in V \setminus S_{k-1}} \begin{pmatrix} \lambda\ Sim_1(f_i, V \setminus S_{k-1}) \\ -(1-\lambda) \max_{g \in S_{k-1}} Sim_2(f_i, g) \end{pmatrix}$$

- Set S is updated is with f, and this continues till the cardinality constraint, size of set S = k is reached.

$$S_k = S_{k-1} \cup \{f_k\}.$$

Performance:

- **AM vs GM variant:** AM video mmr performed better than the GM variant, i.e the arithmetic mean would be a better way of getting sim2, the similarity between the frame and a given set of frames as we can see from the SRC (Summary Reference Comparison) graphs, Figure A.1

  The ds of AM are comparatively lower than that of GM as they start from somewhat around 0.6 and end at 0.45 while the GM d values start from 0.65 and end at approx 0.5.

- **Tradeoff due to λ :** The lambda term decides the trade off between the relevance i.e the similarity to visual content and the novelty i.e how unusual or abnormal the frame is compared to the ones already selected in the summary. This in fact also can be used to detect anomalies as the ones with an anomaly are the ones with abnormal behavior and high reconstruction losses. Thus for the anomaly detection problem keeping the lambda low would be beneficial if we are interested in choosing just the abnormal frames. In the paper, for the video dataset which they considered, the best performance was obtained for lambda=0.7 as can be seen in Figure A.1 and here the performance comparison is done wrt human evaluation.

- **Comparison with standard kmeans algorithm:** Kmeans algorithm clusters the similar scenes and then chooses the cluster centres. We have a few advantages of video mmr compared to this:

  - **Stability:** more stable as there are no randomised initial centres

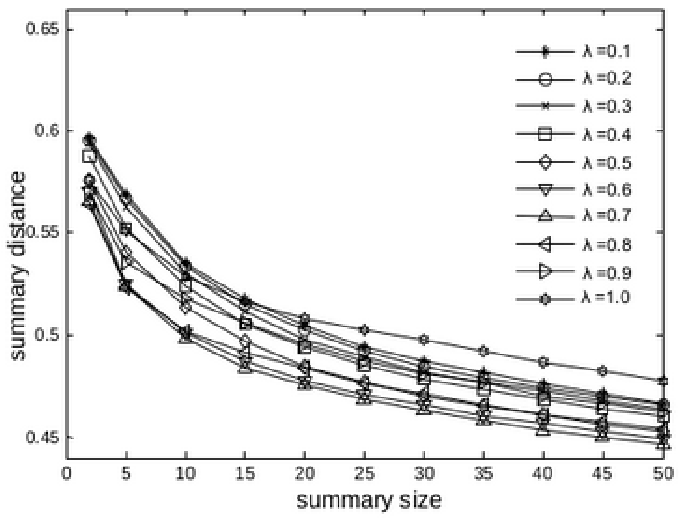


Figure 1. SRC of AM-Video-MMR

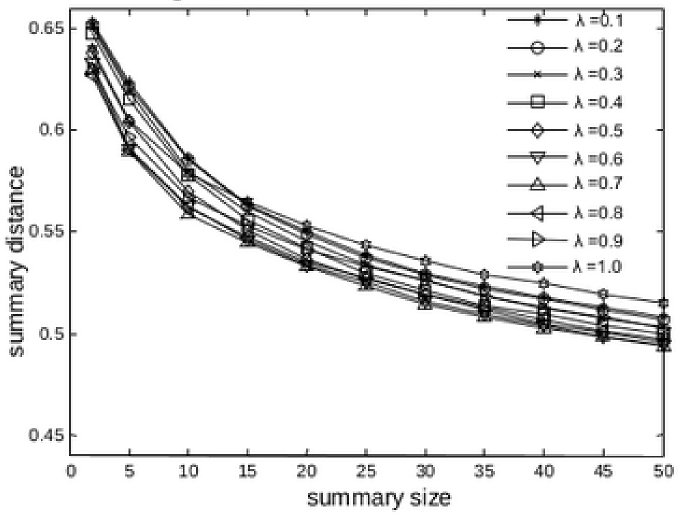


Figure 2. SRC of GM-Video-MMR

Figure A.1: Summary distance vs Summary size

- **Dynamical Summarization:** we can summarize dynamically by maximising the similarity of the frame to be chosen with the remaining frames of the video while minimising similarity with the ones already selected. This can't be done by Kmeans algo as the k centres are kinda fixed and the entire algo needs to be run again in case new frames are given.

Quality wrt Human choice: QC here is the summary quality of a particular algo wrt human choice. As we can see the summary subset, got from video mmr, is much closer to human choice as compared to kmeans.

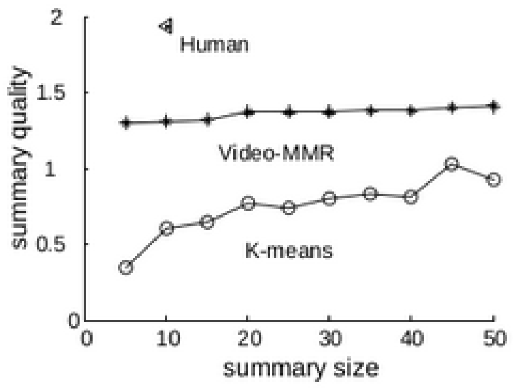


Figure 4. $QC_{Video-MMR}$, $QC_{K-means}$ and QC_{human}

A.1.2 Standard Greedy Algorithm

Lots of video summarization techniques use **Online submodular maximization**. Selecting the best summary subset can be considered as selecting a subset of data elements optimizing a utility function that quantifies "representativeness" of the selected set. These objective functions satisfy submodularity, an intuitive notion of **diminishing returns**, stating that selecting any given element earlier helps more than selecting it later. Data summarization requires maximizing submodular set functions subject to the cardinality constraint i.e the size of the summary is fixed.

If we define a monotonic submodular set function f: 2ˆN $\longrightarrow$ R where N is the finite total number of frames. Therefore if V is the entire set of frames in the video, the domain of f is the powerset of V. For every possible keyframe subset S $\subset$ V, f(S) quantifies the utility of set S, i.e how well S represents V. This equation here quantifies the increase in utility by adding e to set S and this is called the **marginal utitilty** of e wrt S.

$$\triangle_f\ (e|S) \doteq f(S \cup \{e\}) - f(S)$$

Observe that f is monotone implies $\triangle$f $\geq$0 for all e and S. i.e adding any frame to the subset S is gonna increase the utility. And f is submodular implies if A is a subset of B and e doesn't belong to A and B, Adding an element to A is more helpful than adding it to B.

$$\triangle_f\ (e|A) \geq \triangle_f(e|B).$$

Algorithm
Greedy algorithm starts with the empty set, and iteratively locates the element with maximal marginal benefit, i.e at iteration i   This requires random access to the data.

$$S_i = S_{i-1} \cup \{\arg\max_{e \in V} \triangle_f(e|S_{i-1})\}.$$

Hence, while it can easily be applied if the data fits in the main memory, it is impractical if the data is arriving over time at a fast pace as we won't be knowing the marginal utility of those frames in comparison to the frames we have already seen.

Performance
If k is the size of the summary needed and n is the total number of frames in the video

- O(k) memory
- O(nk) computation time
- 1-1/e approximation to the optimal solution

A.2 Synchronous summarzation

Here, unlike the asynchronous setting, we have an unpredictable sequence of frames streaming in and we get 1 frame in each iteration. It's online and the frames are selected or rejected as soon as they enter and the subset of selected frames keeps getting updated in every iteration. Relevant previous work on synchronous Video summarization includes:

- Stream Greedy Algorithm
- Preemption Streaming
- Sieve Streaming [2]

A.2.1 Stream Greedy

Here, too like in the standard greedy algorithm, we start with the empty set and for the first k iterations, we store all the incoming frames.

For iterations after k, we check whether switching the incoming frame with one in the already selected subset will increase the value of the utility function f by more than some relative threshold. If so a, we switch it with the one that maximizes the utility. Otherwise the summary remains the same.

```
Algorithm 3: Greedy with Theshold($c$)
Let $S_0 \leftarrow \varnothing$.
foreach element $u_i$ revealed do
    if $i \leq k$ then
        Let $S_i \leftarrow S_{i-1} + u_i$.
    else
        Let $u_i'$ be the element of $S_{i-1}$ maximizing $f(S_{i-1} + u_i - u_i')$.
        if $f(S_{i-1} + u_i - u_i') - f(S_{i-1}) \geq c \cdot f(S_{i-1})/k$ then
            Let $S_i \leftarrow S_{i-1} + u_i - u_i'$.
        else
            Let $S_i \leftarrow S_{i-1}$.
```

Performance
If k is the size of the summary needed

- O(k) memory
- O(k) queries per frame
- 1-1/ε approximation to the optimal solution where ε is c/k.

A.2.2 Preemption Streaming

This is one of the most simple algorithms for choosing the frames coming through an online stream but is computationally very expensive. It allows the algorithm to reject (preempt) previously accepted elements.

Consider the multilinear extension of this function F:[0,1]^N $\longrightarrow$ R and say, for every N dimensional binary vector x, F(x) gives a real valued positive number corresponding to how well choosing the frames chosen according to x would approximate the entire video. "1" at a particular dimension i in the vector x implies that the ith frame is chosen in the keyframe subset. N here is the total number of frames in the video

For making sure the uth frame is selected in the key frames subset, consider a N dimensional vector u, such that all values except for the uth coordinate are 0s. Now (x $\vee$ u), the coordinate-wise union of x and u would have 1 as the uth coordinate and hence would surely have u as one of the selected frames.

The vector N is say a N-dimensional vector with 1s at each coordinate, then for the vector (N-u), all values except for the uth coordinate are 1s. Therefore (x $\wedge$ (N-u)), the coordinate wise intersection of x and (N-u) would surely have a 0 at the uth

coordinate.
Thus the marginal utility of adding frame u can be given by

$$\partial_u F(x) = F(x \vee u) - F(x \wedge (\mathcal{N} - u))$$

Algorithm
For every frame ui received at the ith iteration, we define a threshold θi on the marginal benefit. If Ni is the set of all frames received till now, we define the subset Si containing the frames with marginal benefit more than θi and the other frames are rejected.
This way if θi is $>$ than θi-1, previously selected frames can be rejected too and that explains the term "pre-emption".

```
Algorithm 1: Marginal Choice
1 foreach element $u_i$ revealed do
2     Choose a uniformly random threshold $\theta_i \in [0, 1]$.
3     Let $\mathcal{N}_i \leftarrow \{u_1, u_2, \ldots, u_i\}$.
4     Let $S_i \leftarrow \{u_j \in \mathcal{N}_i \mid \partial_{u_j} F(\theta_j \cdot \mathcal{N}_i) \geq 0\}$.
```

Performance

- O(k) memory and O(k) queries per frame
- 1/4 approximation to the optimal solution

A.2.3 Sieve Streaming

This final optimum algorithm requires only a single pass through the data, and memory independent of data size. The challenge in this setting is that, when we receive the next element from the stream, we must immediately decide whether it has “sufficient” marginal value. The approach builds on three key ideas:

- a way of knowing the utility value of the optimum summary subset and using it to make a threshold for the marginal values of the incoming frames
- As optimum utility value is difficult to guess, we later on guess the threshold based on the maximum marginal value of a singleton element or a single frame
- lazily running the algorithm for different thresholds when the maximum marginal utility observed till now keeps getting updated

As our final algorithm is a careful mixture of these ideas, we showcase each of them by making certain assumptions and then removing each assumption to get the final algorithm.

A.2.3.1 Assumption1: Knowing OPT helps

OPT is the maximum utility value of the function for any subset of size k. Greedy algorithms work because at every iteration, an element is identified and it reduces the "gap" to the optimal solution by a significant amount.

We can easily see that if Si is the subset selected by following the greedy algo i.e if Si is obtained using the following update rule,

$$S_i = S_{i-1} \cup \{\arg\max_{e \in V} \triangle_f(e|S_{i-1})\}.$$

for the next element ei+1 to be added to Si, the marginal benefit is $\geq$ (OPT-f(Si))/(k-|Si|) as marginal benefits of the chosen frames keeps on diminishing.

Our challenge in sieve streaming is to immediately decide whether an incoming frame has sufficient marginal value. This will require us to compare it to OPT in some way which gives the intuition that knowing OPT should help. If we follow this intuition, for the first element to be chosen, we have the threshold for marginal benefit as OPT/k. This however won't work if there is a single element just above OPT/k at the end of the series while the rest of the elements with marginal value just below OPT/k appear towards the beginning of the stream. Our algorithm would have then rejected these elements with marginal value just below OPT/k and can never get their value.

Therefore instead of keeping the threshold as OPT/k, we keep it as β*OPT/k and chose a suitable β. Say β=½.

Now suppose we know OPT uptil a constant factor $0 < \alpha < 1$; i.e we know v st v $\in$ [α*OPT, OPT]. Then at the ith iteration, if Si is the selected subset and |Si| $<$ k, the algorithm adds a given new frame only if its marginal value is $>$ (v/2-f(Si))/(k-|Si|) (as beta=½, the v/2 factor appears)

```
Algorithm 1 SIEVE-STREAMING-KNOW-OPT-VAL
Input: $v$ such that OPT $\geq v \geq \alpha$ OPT
1: $S = \emptyset$
2: for $i = 1$ to $n$ do
3:     if $\triangle_f(e_i \mid S) \geq \frac{v/2 - f(S)}{k - |S|}$ and $|S_v| < k$ then
4:         $S := S \cup \{e_i\}$
5: return $S$
```

A.2.3.2 Assumption2: Knowing max f(e) suffices

The second step is to assume that we know m, ie the maximum utility value of the function for just a single element or frame. Knowing OPT is difficult as we don't have the solution beforehand. In order to get a very crude estimate on OPT, it is enough to know the maximum value of any singleton element m = max(f(e), e $\in$ V We observe m $\geq$ OPT $\geq$ k*m. (Reason: Greedy algorithm requires us to pick frames with the highest marginal values in descending order. As the first maximum value is m, the total sum for the entire summary would always be less than k*m)

Now if we consider the set

$$O = \{(1+\epsilon)^i | i \in \mathbb{Z}, m \leq (1+\epsilon)^i \leq k \cdot m\}.$$

For atleast one of the thresholds v $\in$ O, v would be a good estimate of OPT, i.e (1-ϵ)*OPT $\geq$ v $\geq$ OPT. So, we could run the previous Algorithm 1 once for each value v $\in$ O, requiring multiple passes over the data and then output the best solution obtained. Since the algorithm does not know which value v is a good estimate for OPT, it simulates Algorithm 1 for each of these values v $\in$ O.

The size of set O is |O| = O((log k)/ϵ), so we need that many passes of Algo1.

Algorithm 2 SIEVE-STREAMING-KNOW-MAX-VAL

```
Input: $m = \max_{e \in V} f(\{e\})$
1: $O = \{(1+\epsilon)^i | i \in \mathbb{Z}, m \leq (1+\epsilon)^i \leq k \cdot m\}$
2: For each $v \in O, S_v := \emptyset$
3: for $i = 1$ to $n$ do
4:     for $v \in O$ do
5:         if $\triangle_f(e_i \mid S_v) \geq \frac{v/2 - f(S_v)}{k - |S_v|}$ and $|S_v| < k$ then
6:             $S_v := S_v \cup \{e_i\}$
7: return $\text{argmax}_{v \in O_n} f(S_v)$
```

Obtaining "m" of all singletons still requires one pass over the full data set, and thus this results in a two-pass algorithm, one for finding m and the other for selecting the frames and running algo1 for every possible v $\in$ O. But our final algo should need just one pass over the incoming stream data.

A.2.3.3 Lazy updates: The final Algorithm

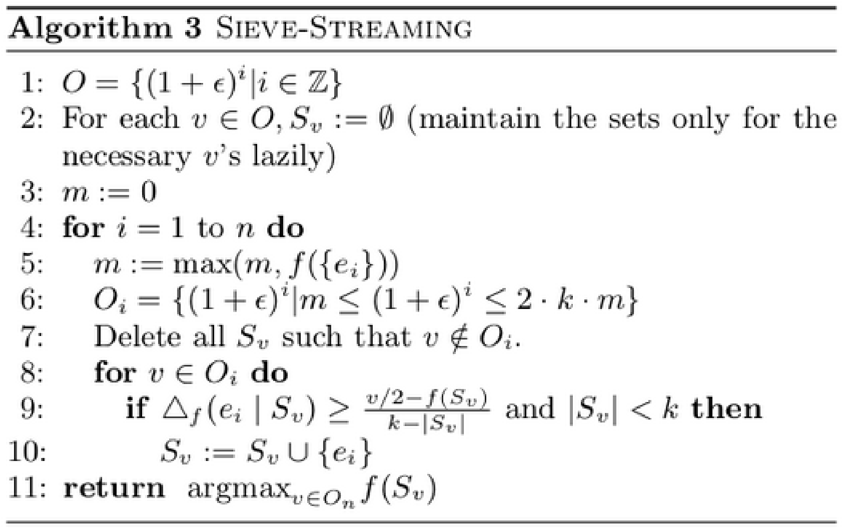
One idea for reducing the previous algorithm to a one-pass algorithm is to maintain an auxiliary variable m which holds the current maximum singleton element after observing each incoming element and lazily instantiate the thresholds v $\in$ O defined previously for this m.

This however won't work as m would be an underestimate of the actual value of max(f(e)), and hence v chosen above could be much smaller than real OPT. This implies v/2k, the threshold marginal value for choosing the first frame would be small and with fixed cardinality, only the first few frames would get selected leaving the others unattended. Therefore, we increase the range of v to

$$v = (1+\epsilon)^i, m \leq (1+\epsilon)^i \leq 2 \cdot k \cdot m.$$

Thus the final algo maintains an auxiliary variable m that holds the current maximum singleton element after observing each element ei . Whenever m gets updated, the algorithm lazily instantiates the set Oi and deletes all thresholds outside Oi or all vs $\notin$ Oi.

We then consider Sv for each v which keeps getting updated with ei+1 in the ith iteration if the marginal value of ei+1 is $\geq$ (v/2-f(Svi))/(k-|Svi|) and |Svi| $\leq$ k. Finally, we output the best solution among Svs for different vs $\in$ final Oi.



```
Algorithm 3 SIEVE-STREAMING
1: $O = \{(1+\epsilon)^i | i \in \mathbb{Z}\}$
2: For each $v \in O, S_v := \emptyset$ (maintain the sets only for the
   necessary $v$'s lazily)
3: $m := 0$
4: for $i = 1$ to $n$ do
5:     $m := \max(m, f(\{e_i\}))$
6:     $O_i = \{(1+\epsilon)^i | m \leq (1+\epsilon)^i \leq 2 \cdot k \cdot m\}$
7:     Delete all $S_v$ such that $v \notin O_i$.
8:     for $v \in O_i$ do
9:         if $\triangle_f(e_i \mid S_v) \geq \frac{v/2 - f(S_v)}{k - |S_v|}$ and $|S_v| < k$ then
10:            $S_v := S_v \cup \{e_i\}$
11: return $\mathrm{argmax}_{v \in O_n} f(S_v)$
```

Performance

- one pass over the entire dataset and O(klogk/ϵ) memory
- (½-ϵ) approximation to OPT
- O(nlogk/ϵ) complexity whereas classical greedy had O(nk) complexity

Limitations

- f depends on the entire datastream V
- |S| can be smaller than k

To overcome the above limitations, we define the evaluation of the utility function f restricted to a subset W $\subseteq$ V and not the entire V

$$\cancel{f(S) = \frac{1}{|V|}\sum_{e\in V} f_e(S),} \qquad f_W(S) = \frac{1}{|W|}\sum_{e\in W} f_e(S).$$

as long as W is large enough and its elements are randomly chosen, the value of the empirical mean fw(S) will be a very good approximation of the true mean f(S).

PROPOSITION 6.1. *Assume that all of $f_e(S)$ are bounded and w.l.o.g. $|f_e(S)| \leq 1$. Moreover, let W be uniformly sampled from V. Then by Hoeffding's inequality we have*

$$\Pr(|f_W(S) - f(S)| > \xi) \leq 2\exp\left(\frac{-|W|\xi^2}{2}\right).$$

There are at most $|V|^k$ sets of size at most k. Hence, in order to have the RHS $\leq \delta$ for *any* set S of size at most k we simply (using the union bound) need to ensure

$$|W| \geq \frac{2\log(2/\delta) + 2k\log(|V|)}{\xi^2}. \tag{8}$$

Also, this W acts as a reservoir and can be used to fill up |S| in case it's smaller than k. Therefore to get the subset W, we just need to sample uniformly at random from a data stream once and then use our stream algo, resulting in a 2-pass algorithm but without the previous limitations.

Algorithm 4 SIEVE-STREAMING + Reservoir Sampling

```
1: Go through the data and find a reservoir $W$ of size (8)
2: Run SIEVE-STREAMING by only evaluating $f_W(\cdot)$
```

Competitive Analysis: The performance of each algorithm can be done by comparing them to an optimal offline algorithm that can view the sequence of frames in advance. Listing down the results of the relevant algorithms for comparison:

Algorithm	Approximation Ratio	Memory	Queries per Element	Stream	Reference
Greedy	$1 - 1/\exp(1)$	$\mathcal{O}(K)$	$\mathcal{O}(1)$	✗	(Nemhauser and others 1978)
StreamGreedy	$1/2 - \varepsilon$	$\mathcal{O}(K)$	$\mathcal{O}(K)$	✗	(Gomes and Krause 2010)
PreemptionStreaming	$1/4$	$\mathcal{O}(K)$	$\mathcal{O}(K)$	✓	(Buchbinder, Feldman, and Schwartz 2019)
Sieve-Streaming	$1/2 - \varepsilon$	$\mathcal{O}(K \log K/\varepsilon)$	$\mathcal{O}(\log K/\varepsilon)$	✓	(Badanidiyuru et al. 2014)
Sieve-Streaming++	$1/2 - \varepsilon$	$\mathcal{O}(K/\varepsilon)$	$\mathcal{O}(\log K/\varepsilon)$	✓	(Kazemi et al. 2019)
Salsa	$1/2 - \varepsilon$	$\mathcal{O}(K \log K/\varepsilon)$	$\mathcal{O}(\log K/\varepsilon)$	✓	(Norouzi-Fard et al. 2018a)
ThreeSieves	$1/2 - \varepsilon$ with prob. $(1-\alpha)^K$	$\mathcal{O}(K)$	$\mathcal{O}(1)$	✓	this paper