Use of AI and Video Analytics for Anomaly Detection

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 very thankful to Rishabh Dabral for his constant support and motivation at each and every step of this work. I am also thankful to Sukalyan Bhakat for his insights in our meets. Last but not the least, I would like to thank Rachit Bansal, my batchmate and project partner for constantly being there and 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

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. We try to look into the asynchronous as well as synchronous settings. We see how Video Maximal Marginal Relevance can be used for video summarization and selecting just the anomalous frames which we might be interested in. We also later on try to optimise our algorithm using online submodular maximisation for synchronous or online Proctoring.

Contents

Chapter 1

Introduction

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.

1.1 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 1.1

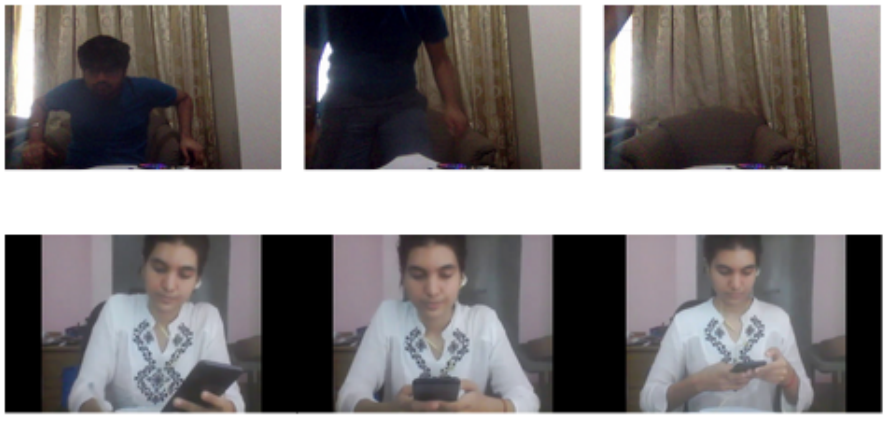


Figure 1.1: Anomalous actions: Walking away from the camera view and Using a mobile phone

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 1.2

Figure 1.2: 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.

1.2 Proposed Pipeline

The flow diagram of the pipeline can be seen in Figure 1.3.

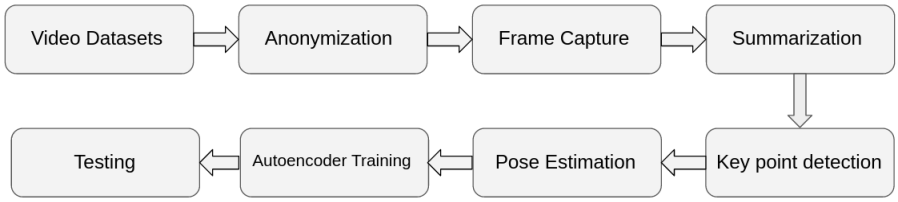


Figure 1.3: Proposed pipeline

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

Chapter 2

Video Datasets and Pre-processing

Collection of Video Datasets were done as follows:

- **Our webcam recordings:** We recorded ourselves doing a few anomalous as well as non-anomalous actions.
- **Google Form responses:** Sent out a google form for collecting videos from voluntary friends and batchmates
- **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. [**ntu**]

2.1 Anonymization

Respecting the student's privacy is very important, hence we ensured that the student's identity is in no way disclosed or used anywhere in the entire project. The data was shared with only the 4 people working on the project, Prof Ganesh Ramakrishnan, Rishabh Dabral, Rachit Bansal and myself, Poorvi Hebbar. We renamed the videos according to the order in which they appeared in the shared folders and also sent the code for renaming the files to all the professors who were willing to share the recordings so that they can do that themselves before sharing. The code is available here. [**rename**]

Frame capture was done at 30fps but every 15th frame was considered thus leading to an average of 2 frames per second being stored.

Chapter 3

Asynchronous and Synchronous Video summarization

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.

3.1 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) [**vmmr**]
- Standard Greedy Algorithm

3.1.1 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

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

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,

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

- 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 5.2

  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 5.2 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

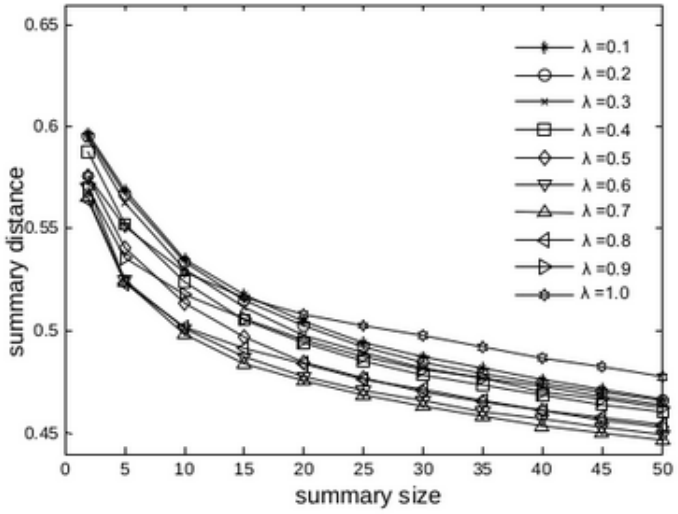


Figure 1. SRC of AM-Video-MMR

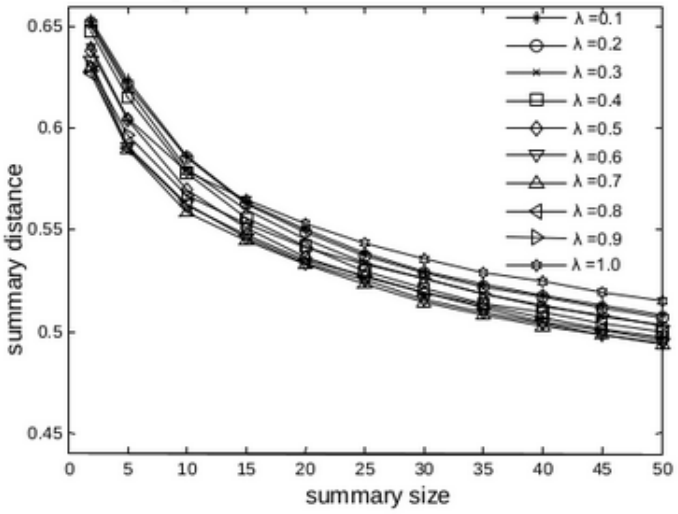


Figure 2. SRC of GM-Video-MMR

Figure 3.1: 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.

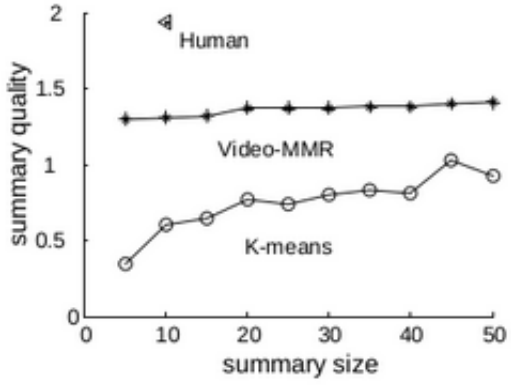


Figure 4. $QC_{Video-MMR}$, $QC_{K-means}$ and QC_{human}

3.1.2 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: 2^N $\longrightarrow$ 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

3.2 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 [**sieve**]

3.2.1 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 ← ∅.
foreach element u_i revealed do
    if i ≤ k then
        Let S_i ← 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}) ≥ c · f(S_{i−1})/k then
            Let S_i ← S_{i−1} + u_i − u'_i.
        else
            Let S_i ← 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.

3.2.2 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]ˆN $\longrightarrow$ 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

foreach *element* u_i *revealed* **do**
  Choose a uniformly random threshold $\theta_i \in [0, 1]$.
  Let $\mathcal{N}_i \leftarrow \{u_1, u_2, \ldots, u_i\}$.
  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

3.2.3 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.

3.2.3.1 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 $\texttt{OPT} \geq v \geq \alpha$ $\texttt{OPT}$

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

3.2.3.2 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.

3.2.3.3 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

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

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

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

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

Chapter 4

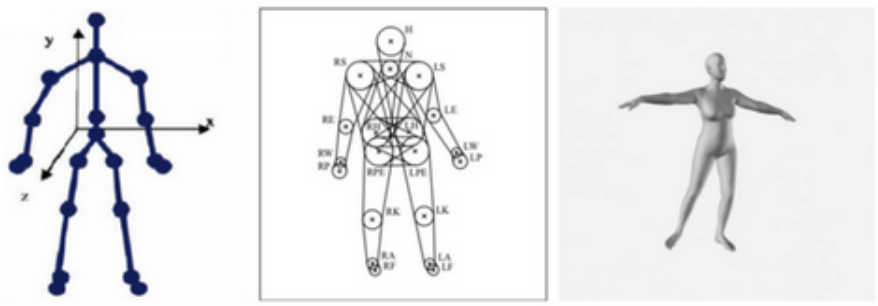
Human Pose Estimation

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

Single-person vs Multi-person Pose Estimation.

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.

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.

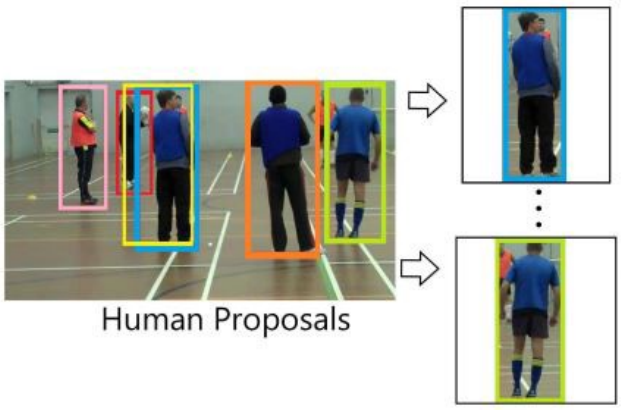


Kinematic Model vs Shape-based Model vs Mesh-based Model

4.1 Pipeline

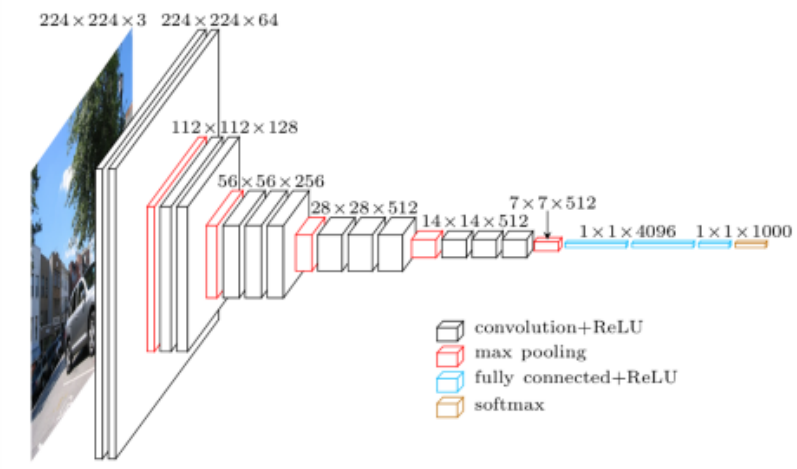
The brief pipeline for pose estimation has 4 steps[**pose**]

- **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)

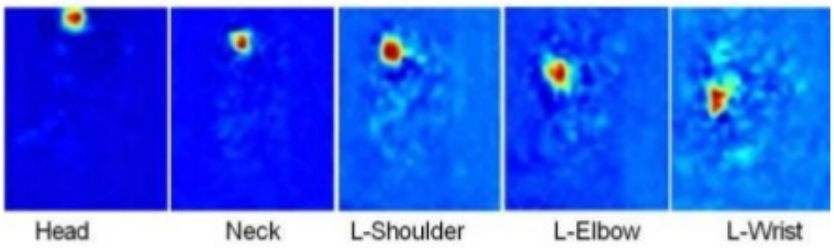
- **Feature Extraction:** This refers to deriving some values from raw data (such as an image or video in our case), that can be used as input to a learning algorithm. The features are implicit and the approach uses an encoder-decoder architecture. Instead of estimating keypoint coordinates directly,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

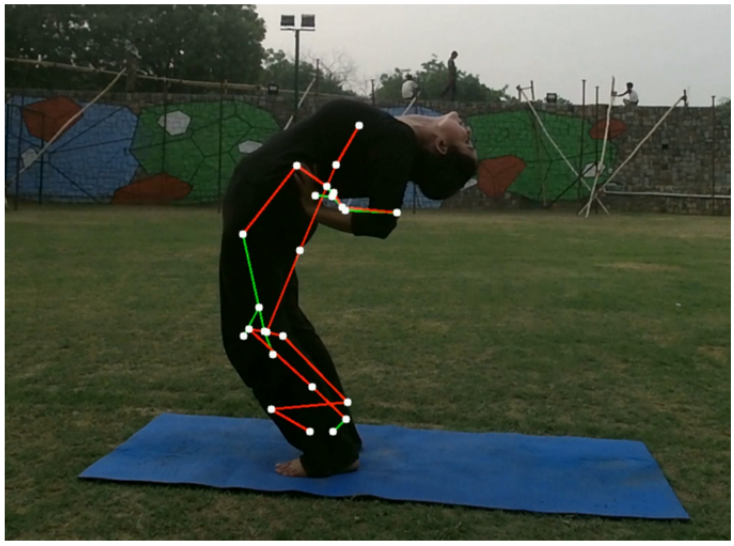
We use a pre defined MobileNet-based architecture for the encoder. This architecture has depthwise separable convolutions that require fewer parameters and less computation but still provide solid accuracy.

- **Inference:** We use confidence maps for predicting joint locations. A 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, and 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 that scores every pose based on its likeliness. Poses that get scores lower than a threshold are ignored.

4.2 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[**github**] to build our own code[**poorvi-github**]. 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.

An example of this can be seen in the soccer image, Figure 4.1. 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) original (b) posenet

Figure 4.1: Example of posenet from original soccer image

```
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$
```

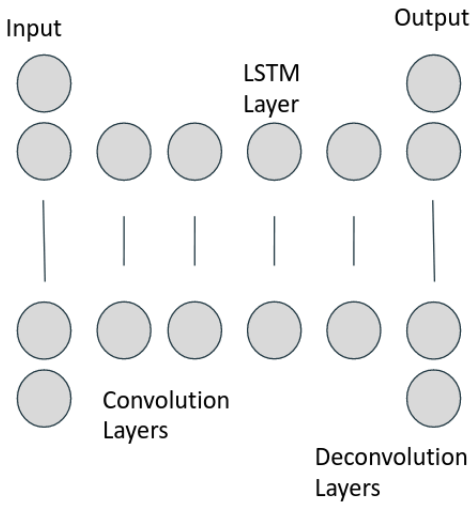
The combined vectors of all the frames in a particular video were then passed on to the auto-encoder with their corresponding labels.

Chapter 5

Autoencoder Model

We use an autoencoder[**autoencoder**] 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 that gives the minimum reconstruction loss. The reconstruction loss also decreased substantially after normalising the vectors.

5.1 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 the anomalous frames had lighting issues, therefore, the student is suggested to sit in a well-lit area during the exam.

5.2 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 at the beginning of the report are as follows:

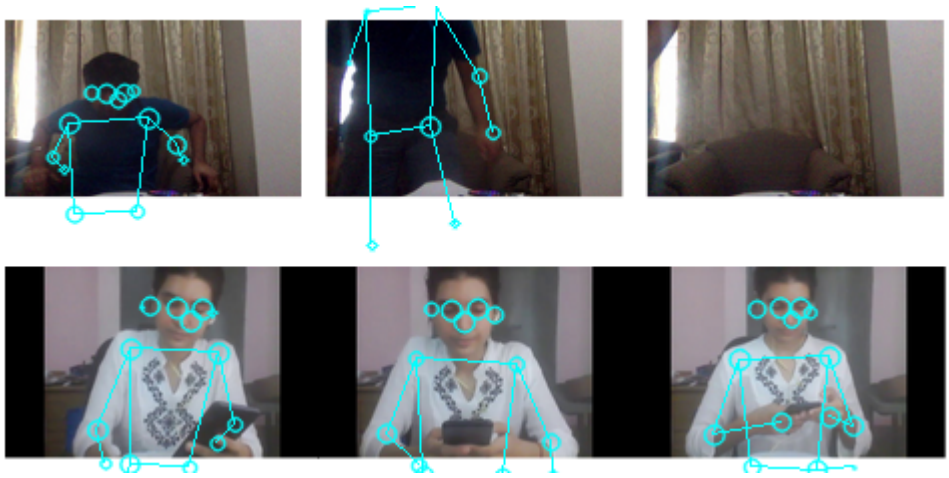


Figure 5.1: Test output label: Anomalous

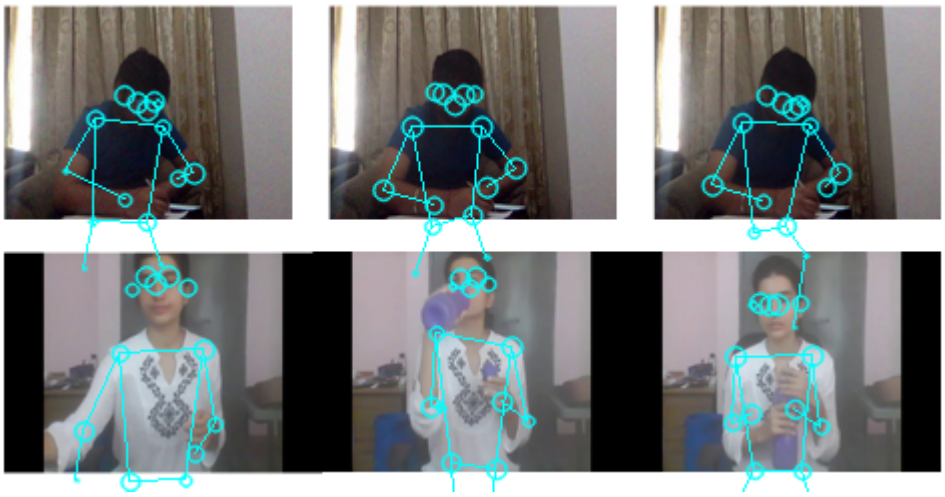


Figure 5.2: Test output label: Non-Anomalous

Chapter 6

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 that is person agnostic. Action Recognition NTU-RGB dataset[**ntu1**] has a collection of 60 actions with around 56k samples and NTU-RGB 120[**ntu2**] 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

6.1 Actions and Rating Labels

Coming to the dataset actions, there were 3 broad classes: daily actions, mutual actions, and medical conditions (Figure 6.1). 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 6.1: Actions

Figure 6.2: Sample Frames

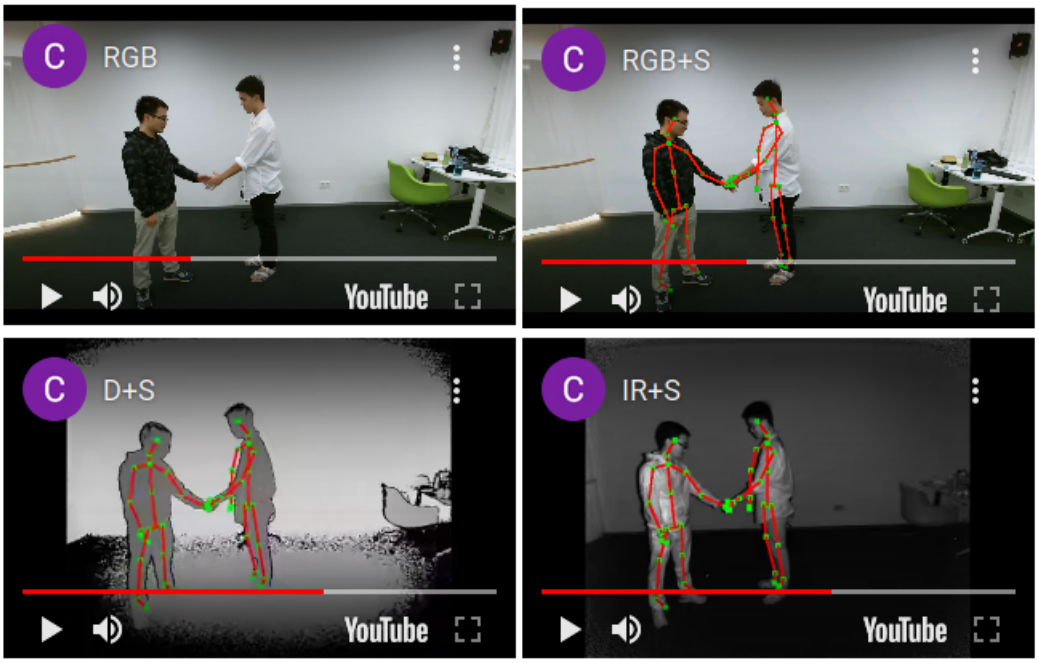


Figure 6.3: Sample Videos

Some sample frames and videos in the dataset are shown in Figure 6.2 and 6.3 respectively.

6.2 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.[**ntu-github**]

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.

2D Pose Estimation vs 3D Pose Estimation

The 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.

6.3 Results

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 10^-2) is more for the dataset with rating 5 (of the order 10^-3). As the reconstruction loss is of the order 10^-5 while training for this dataset, the task of finding better layers has to be done in future.

Future Scope

Autoencoder training and testing for NTU-RGB Dataset with 5 classes of actions needs to optimized and testing should be made into a 5-class classification problem. The other way to work on this would be to rank the frames in descending order of reconstruction error (Rank 1 is most difficult to reconstruct. This would be an efficient way to detect anomalies in a synchronous setting.

Efficient Anomaly detection and Automated Proctoring using posenet features for different modes of exams (SAFE, Moodle etc) needs to be completely implemented. Synchronous Video Summarization has not yet been done and needs to be completed and included in the pipeline. Video Summarization and anomaly detection needs to be done simultaneously in a this set up and the reconstruction loss or video-MMR concept could be a good start for this.

Bibliography

[1] https://towardsdatascience.com/human-pose-estimation-simplified-6cfd885

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

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

[4] http://rose1.ntu.edu.sg/datasets/actionrecognition.asp

[5] https://colab.research.google.com/drive/18DmaHJuTL0qUWbn8kSqOqCA_dEfRz6Od?usp=sharing

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

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

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

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

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

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