# CHEST PAIN DETECTION IN YOUTUBE VIDEOS

A Thesis

by

YIPENG LU

Submitted to the Graduate and Professional School of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

December 2021

Major Subject: Electrical and Computer Engineering

## ABSTRACT

The topic of this research is to detect chest pain action in YouTube videos. Chest pain detection is very important in smart home applications. However, chest pain detection in YouTube videos is very challenging due to the dissimilarities between YouTube videos and the training set.

In this research, we implemented 5 promising network architectures for chest pain detection and compared their performance. We proposed both frame detectors based on a single frame and clip detectors based on a sequence of frames. Both human skeleton data, as well as RGB information, were extracted as the input feature of the models. We adopted a wide range of network architectures for detection, such as Inception Resnet, simple feed-forward network, RNN, faster RCNN, and I3D. The proposed network architectures were trained on NTU RGB+D which is a clip-wise-labeled dataset containing a wide range of human actions, including chest pain. We implemented APIs of our detectors that feed the input videos to our trained models and visualize the inference results by drawing bounding boxes and confidence scores directly on the input videos. The performance of the detectors was evaluated on both the labeled dataset and the challenging YouTube videos, and promising results were obtained. In the end, we explored the temporal action localization architectures and discussed their viability to be trained on the current dataset.

## DEDICATION

To my parents.

# ACKNOWLEDGMENTS

I would like to express the deepest appreciation to my thesis advisor, Professor Anxiao (Andrew) Jiang, who has the attitude and the substance of a genius. Without his guidance and persistent help, this thesis would not have been possible.

I would also like to thank my committee members, Professor Nicholas Duffield, Professor Andreas Klappenecker, and Professor Tie Liu for serving the committee.

## CONTRIBUTORS AND FUNDING SOURCES

### Contributors

This work was supported by a thesis committee consisting of Professor Anxiao (Andrew) Jiang[advisor], Professor Nick Duffield from the Department of Electrical and Computer Engineering, Professor Andreas Klappenecker from the Department of Computer Science and Engineering and Professor Tie Liu from the Department of Electrical and Computer Engineering.

All other work conducted for the thesis was completed by the student independently.

### Funding Sources

No other outside source of funding was provided.

# NOMENCLATURE

| | |
|---|---|
| CNN | Convolutional Neural Network |
| ConvNet | Convolutional Network |
| RNN | Recurrent neural network |
| FPS | Frames per Second |
| IOU | Intersection over Union |
| RCNN | Region Convolutional Neural Networks |
| YOLO | You Look Only Once (object detection model) |
| SSD | Single Shot Detector |
| I3D | Two-Stream Inflated 3D ConvNet |
| NN | Neural Network |

TABLE OF CONTENTS

LIST OF FIGURES

# LIST OF TABLES

# 1. INTRODUCTION

There has been a lot of remarkable research work in the field of computer vision nowadays. Many novel neural networks with novel architectures that can perform new tasks with better accuracy keep coming out. The topic of this research is to train neural networks with various architectures using existing datasets and detect chest pain action in everyday videos.

Chest pain detection is a key component in smart home applications. Chest pain is a dangerous medical condition that can be caused by lots of illnesses including heart attack which happens to millions of people in the United States each year. It can happen in any age group and can happen at any time. When it is happening, the situation could get worse very quickly if without immediate medical treatment. If an algorithm can detect if a person is suffering from chest pain through an RGB camera, and report to the nearest hospital immediately, it could save the person's life.

However, in the meantime, action detection in everyday videos is an extremely challenging task in the field of computer vision. To obtain everyday videos outside of the dataset, we searched for chest pain videos on YouTube. The dissimilarities between the training set and YouTube videos are a big factor for misclassification. Some dissimilarities such as differences in the background and differences in the target person size can be mitigated by training techniques such as data augmentation. However, other dissimilarities can be harder to handle. For example, the training set contains the full human body, while YouTube videos usually only contain part of the human body. The training set is recorded from one angle, while YouTube videos are recorded from a different angle. The training set contains several patterns of the action, while YouTube videos may have a new pattern that has not been seen by the models yet. Also, as we are using only the RGB videos, the input data tends to lack 3D structure which is also critical information for human action detection. Figure 1.1 shows some difficult cases for action detection in YouTube videos.

We implemented both frame detectors based on a single frame and clip detectors based on a sequence of frames. Some actions usually cannot be decided by a single frame. For example, opening the refrigerator and closing the refrigerator. However, in most cases, we can observe the

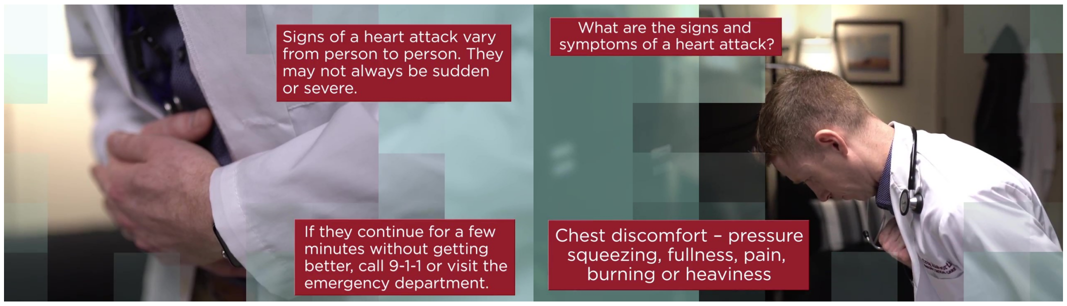
Signs of a heart attack vary from person to person. They may not always be sudden or severe.
What are the signs and symptoms of a heart attack?
If they continue for a few minutes without getting better, call 9-1-1 or visit the emergency department.
Chest discomfort – pressure squeezing, fullness, pain, burning or heaviness

Figure 1.1: YouTube videos usually only contain part of the human body[1]

chest pain action by a single frame. If a person is holding his chest with a painful look, it is almost certain that he is having chest pain. Figure 1.2 is a good example of it. Since chest pain action does not depend too much on a sequence of frames, frame detectors based on a single frame could be good at handling it. In contrast to frame detectors that make detections by just a single frame, clip detectors analyze a sequence of frames and return a confidence score. Usually, clip detectors have better accuracy than frame detectors, but are computational more expensive, and need novel ways to utilize the temporal features in a sequence of spatial feature maps.

Both human skeleton data, as well as RGB information, are extracted as the input feature of the models. The RGB information of an image is a 3-D array of pixel values range from 0 to 255. We, as humans, see and interpret the world with RGB information. The skeleton data contains the position information of some key body parts of humans, such as hands, face, and foot. It contains much less information compared to RGB information which means that detectors will less likely to be misled by non-generalizable features. For example, if we find out the position of the hand is close to the position of the chest, it may indicate that the person is having chest pain.

There are 5 detectors in total implemented in this research, with various network architecture. For the frame detectors with skeleton data, we used a feed-forward network to analyze the extracted features. For the frame detectors with RGB information, we trained two kinds of architectures. One detector only classifies the image with the architecture of the Inception Resnet, while in the other detector, the architecture of the faster RCNN[3] was adopted to both localize and classify

[1]Image from YouTube video `https://www.youtube.com/watch?v=aWn8g7saQKU`

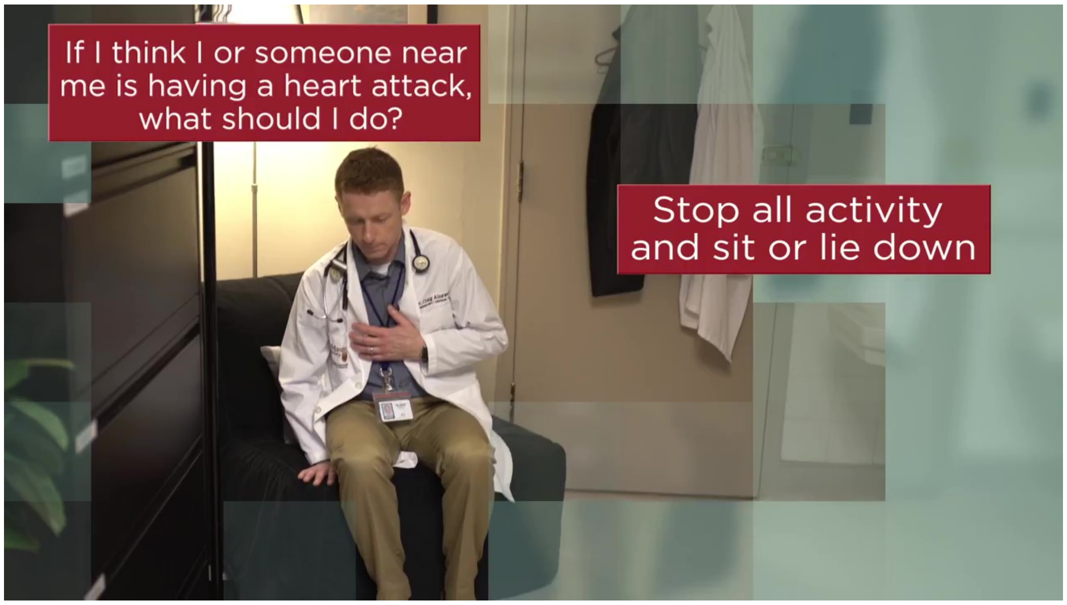
If I think I or someone near me is having a heart attack, what should I do?
Stop all activity and sit or lie down

Figure 1.2: In most cases we can observe the chest pain action by a single frame[2]

the person having chest pain. For the clip detectors with skeleton data, RNN was used to explore temporal patterns among a sequence of feature maps. For the clip detectors with RGB information, we adopted the architecture of I3D[4] which is excellent at handling Spatio-temporal features inside video clips.

We implemented APIs of our detectors that feed the input videos to our trained models and visualize the inference results by drawing bounding boxes and confidence scores directly on the input videos. The API firstly extracts all features from the input video. Then our trained model inferences those features one batch at a time. After that, we concatenate the batch output to obtain the complete output for the input video. The output is a sequence of the confidence score, and some models also output bounding boxes that localize the person having chest pain as well. However, a list of float numbers or tuples is not user-friendly at all. To better visualize our results, we draw bounding boxes and confidence scores directly on the input videos.

We evaluated our detectors on both the labeled dataset and the challenging YouTube videos. On the test dataset, we compared the precision, recall, and accuracy among our detectors. On both

[2]Image from YouTube video `https://www.youtube.com/watch?v=aWn8g7saQKU`

the test dataset and the YouTube videos, we explored cases where each detector is good at and cases where each detector makes mistakes.

# 2. LITERATURE REVIEW

## 2.1 Image classification

Image classification is one fundamental task in the field of computer vision. Given an image and a set of classes, the task is to assign a class to the image.

In order to find a suitable architecture for chest pain detection, we reviewed literature in the field of image classification. Very deep convolution networks turned out to be good at image classification, and achieve top performance on popular image classification datasets, such as ImageNet[5] and CIFAR[6]. ImageNet is one of the most popular datasets in the field of image classification. Its scale is so large that it contains 1.43 million images from 1000 object classes. With such a large-scale dataset, various neural network architectures can test their performance reliably. VGG16[7] uses very small convolution filters with size 3x3 while pushes the depth to 16 layers, and obtains a significant improvement over previous configurations. It demonstrates that deeper architecture can lead to better performance at the cost of more computationally expensive, while the efficiency may not be that satisfying. Inception architecture[8] aims to utilize the computational cost as efficiently as possible and achieves better accuracy with fewer parameters. Residual connections[9] make training deep networks easier, and allow us to optimize substantially deeper networks. Inception Resnet[10] combines inception architecture with residual connections and speeds up the training process of inception networks significantly. If we want better accuracy, we will need to scale up our very deep convolution networks. In Efficientnet[11], a new scaling method is proposed to scale the depth, width, and resolution of the network more efficiently. As a result, the Efficientnet family achieves state-of-the-art accuracy while being much smaller and faster.

Besides classifying images in the current dataset, trained very deep convolution networks also have a wide range of applications. By fine-tuning the model weights, we can obtain a high accuracy even on a small dataset. Also, the feature map extracted from the very deep convolution layers can be used in a lot of areas, such as object detection and pose estimation.

[12] trained a custom CNN on a self-collected dataset to classify between chest pain images and non-chest pain images and achieved 91.75% accuracy and 92.85% sensitivity. Its high performance demonstrates that image classification can be useful for chest pain detection. However, improvements could be made to further increase its performance.

First, its custom CNN is shallow compared to the state-of-the-art image classification architectures, which could lower the performance. In our model, we adopted the architecture of Inception Resnet, which has much more parameters. Second, it trained the custom CNN from scratch, which completely abandoned the benefit of transfer learning. While we fine-tuned the Inception Resnet with the weight pre-trained on ImageNet. Third, it used masked RCNN software to crop the target person in each image during preprocessing, which could increase the runtime and introduce noisy margins. In contrast, we only applied some basic data augmentation, such as horizontal flip and rotation during augmentation.

## 2.2 Object detection

Object detection is an advanced task that is based on image classification techniques. As shown in figure 2.1, given an image and a set of classes, the task is to locate the object with a bounding box and assign a class to the object.

Some popular object detection datasets are PASCAL VOC detection benchmarks[13] and MS COCO dataset[14]. The popular object detection methods nowadays can be approximately classified into two categories: two-stage methods which use two separate models to generate reliable proposals and classify the proposals, such as RCNN[15], fast-RCNN[16] and faster-RCNN[3], and single-stage methods which just use a single big convolution model, such as YOLO[17] and SSD[18]. A proposal is a bounding box that is considered to have a high possibility to contain an object belonging to a set of object classes and is usually obtained by either the selective search algorithm or the region proposal network. Two-stage methods are typically more accurate than single-stage methods at the cost of computational speed.

The initial two-stage method is introduced in RCNN[15]. It uses the selective search algorithm to generate a fixed number of proposals. Then it wraps the image according to each proposal and

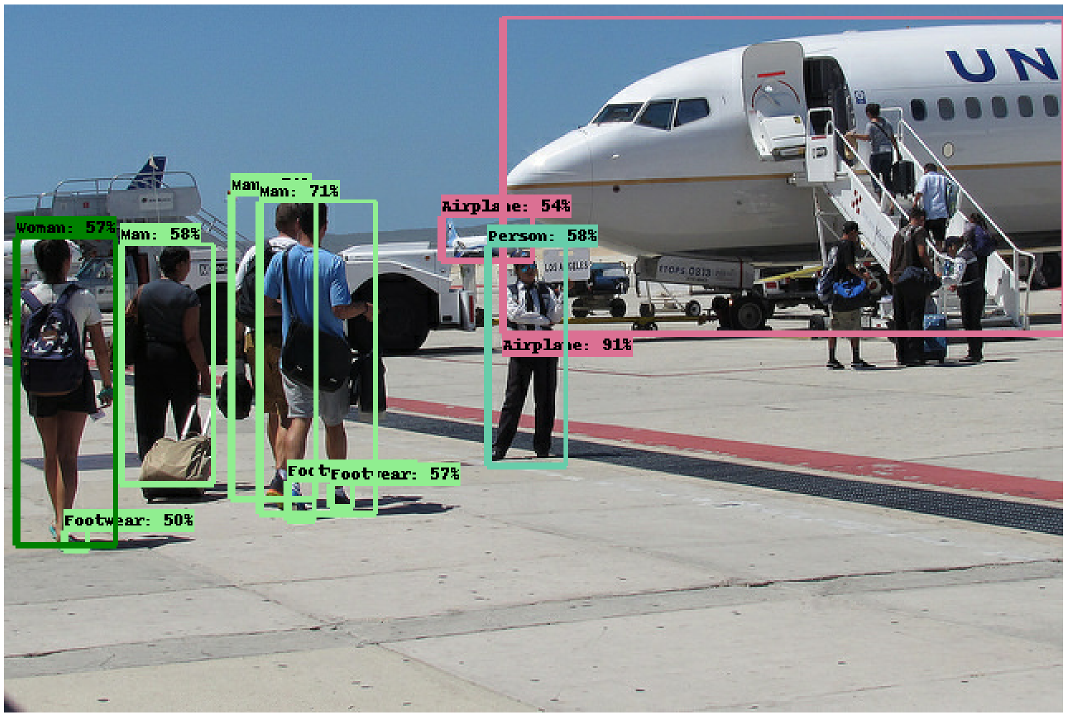

Figure 2.1: Task of object detection: Locate and classify the object[1]

feeds the resized image to a CNN to obtain a fixed-size feature vector. Finally, it uses an SVM to classify the feature vector and assigns a label to each proposal. RCNN outperforms other sliding-window detectors by a large margin, but its computational speed is such a big problem that it takes 47 seconds to process a single image.

The improved version of RCNN[15] is introduced in fast RCNN[16]. Instead of feeding each wrapped image to the CNN separately, faster RCNN extracts feature map from the entire image using the convolution base of the CNN and crop the feature map according to each proposal. Then it uses the RoI pooling layer to get a fixed-size feature vector. After that, instead of just classify the proposal with an SVM, it both classifies the proposal and regresses the bounding box using two separate fully-connected layers. In this way, we do not need to re-compute the feature map for each proposal, and the computational speed is greatly improved. Now we only need 2.3s instead of 49s to process a test image. Also, the boundary is more precise because of boundary regression.

[1]A result figure of the faster RCNN model[3]. Adapted from `https://tfhub.dev/google/faster_rcnn/openimages_v4/inception_resnet_v2/1`

However, the computational speed is still limited by the selective search algorithm which is slow and cannot be trained by training data.

This bottleneck is finally solved in faster RCNN[3]. Faster RCNN replaces selective search algorithm with the region proposal network which is a neural network that can be trained and runs fast. As a result, the time needed to process a test image further reduces from 2.3s to 0.2s. The overall architecture of faster RCNN is shown in figure 2.2.

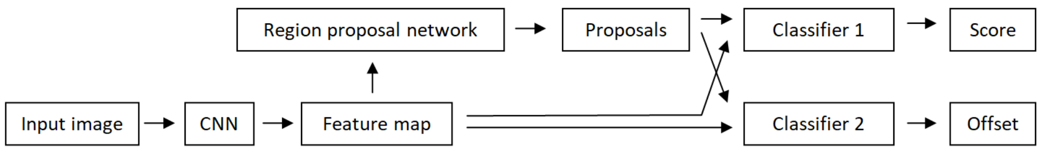

Figure 2.2: Architecture of faster rcnn

[19] trained an SSD on two datasets to classify and identify the vital signs during heart attacks, such as chest pain and achieve a mean average precision of 76.4% and an average recall of 80%. It also deployed the trained model to the Nvidia Jetson Nano for local inferences. Its high performance demonstrates that object detection architectures can not only classify the chest pain action class but also locate the person having chest pain in the image. However, there is still space for improvements.

First, the paper annotates the bounding box of each image manually, which requires a lot of work and limits the size of the dataset to 3k. In our model, we utilized the faster RCNN pre-trained on Open Images V4 to extract bounding boxes automatically and increased the size of the dataset to 80k. Second, the dataset used for training and testing in the paper only contains several vital signs during a heart attack and does not include any normal actions. This could cause the trained model to be weak at distinguishing between normal actions and chest pain actions, which deviates from its motivation. While we include 48 daily action classes apart from chest pain into our dataset to mitigate this issue. Third, as two-stage methods are too complex to deploy on the existing edge embedded devices, it adopted the lightweight single-stage method instead. Because we are not going to deploy our detectors on an edge embedded devices, we selected the two-stage methods which have higher accuracy.

## 2.3 Skeleton-based action recognition

Besides using the RGB information and optical flow which is generated from RGB information, there is another branch of research that focuses on human action recognition using skeleton data. Many kinds of features can be extracted from raw 3D human skeleton data, such as displacement-based feature, orientation-based feature, raw joint positions feature, and multi-modal feature[20].

In our research, because we intended to detect chest pain solely from RGB videos, we have no access to the 3D human skeleton data obtained by some advanced RGB-D cameras such as Microsoft Kinect or Asus Xtion PRO LIVE. Instead, we used the Openpose[2] to extract 2D human skeleton data from RGB videos. However, through extensive literature survey review, we found the study of skeleton-based chest pain detection based on Openpose is lacking. Therefore, we mainly focused on papers that utilize Openpose for skeleton-based action recognition.

About action recognition on a single frame, [21] fine-tuned the Inception Resnet V2 to make fall predictions on the human pose images rendered by Openpose and achieved 91.7% accuracy. However, deep CNNs which are good at exploring rich spatial features in real-world images, are over-complex to deal with pose images which are simply generated by drawing lines between 25 human body key points. Therefore, we used a feed-forward network to process the feature vector generated from the coordinates of 25 human body key points. In this way, the number of model parameters is greatly reduced, which not only mitigated overfitting but also reduced the training and inference time.

About action recognition on a sequence of frames, RNN is usually used to take in the feature and assign an action class to the skeleton data, such as [22], [23] and [24]. [22] trained an LSTM using the de-noised Openpose key point coordinates to classify 5 actions of children with ASD and achieved high accuracy. Its de-noising function consists of scaling the coordinates of the key points to the same unit, removing key points on the head, discarding frames with missing important key points. While in our preprocessing function, we further explored the angle and distance feature between key points and observed a significant accuracy increase.

## 2.4 Video classification

Instead of taking a single image as input in image classification, video classification takes a video clip or a sequence of images into consideration and assigns a label to the video clip. Because usually, human action is a continuous process, video classification usually demonstrates a better accuracy than image classification in the field of action recognition.

Some popular video classification benchmark datasets are HMDB[25], Kinetics[26] and UCF101[27]. One kind of network architecture for video classification combined CNN with RNN, such as [28] and [29]. CNN is especially good at extracting spatial features, while RNN is especially good at capturing temporal patterns. They use the convolutional base of CNN to extract a feature vector for each frame and use RNN to classify the sequence of CNN feature vectors. However, this kind of network architecture has limitations. While being good at capturing high-level variations, it often overlooks the low-level motions, which could damage the performance. Also, it is very expensive to train.

Another kind of network architecture utilizes 3D ConvNets, such as [30], [31] and [32]. With those Spatio-temporal filters, 3D Convnets seem to be a natural way to classify videos. However, as the number of parameters in 3D ConvNets is big, they are hard to train. Also, because it cannot benefit from ImageNet pretraining, its accuracy is not competitive with state-of-the-art.

On the other hand, I3D[4] is one of the most popular and powerful network architectures for video classification. By inflating the successful image classification architecture to 3D ConvNets, it avoids the painstaking process to find a good architecture from scratch. By bootstrapping the weights from the image classification models pre-trained on ImageNet, I3D can benefit from fine-tuning. Moreover, by adopting the two-stream approach which takes in both RGB information and optical flow information calculated using TV-L1 algorithm[33], its accuracy is further improved.

However, although I3D is very promising for video classification, we did not observe any papers using I3D directly for chest pain detection. Although video classification models are good at classifying short video clips containing the entire action sequence, it performs poorly at detecting an action from a long untrimmed video, where most part of the video is background.

When we trimmed the long video into short clips, and classify them with video classification models, a lot of problems arise. First, if we want to reduce the prediction temporal footprint for more sensitive detection, the trimmed short video clips are usually highly overlapped with each other, which causes repeated computation of the 3D convolution feature maps and increases the runtime. Second, lots of the trimmed video clips either only contain part of the action sequence, or contain a large portion of the background, which can easily confuse the video classification models. Therefore, video classification is not suitable for action detection.

On the other hand, the temporal action localization technique, which is based on video classification, is very suitable for action detection.

## 2.5 Temporal action localization

The relationship between temporal action localization and video classification is similar to the relationship between object detection and image classification. The task of temporal action localization is to both locate the action from the video and classify the action. Figure 2.3 is the visualization of temporal action localization on basketball dunk. The image sequence represents the video. For each basketball dunk action in the video, the action segment contains the start time and end time of the action. The blue and green line segments represent the ground truth and predicted action segments.

Some popular temporal action localization datasets are THUMOS'14 detection benchmark[34] and ActivityNet dataset[35]. As object detection is successful by utilizing the successful pre-trained image classifier, it is natural to think that we could achieve temporal action localization using a similar approach. In object detection, we use the convolution base of the image classifier to extract the 2D feature map containing spatial features. While in temporal action localization, we can utilize the convolution base of the video classifier to extract the 1D feature map containing Spatio-temporal features. In object detection, we use 2D ConvNets to process the 2D feature map, while in temporal action localization, we can use 1D ConvNets to process the 1D feature map.

[36], [37] and [38] adopt similar ideas, and get promising accuracy on temporal action localization benchmark dataset. Later, [39] goes one step further and achieves better accuracy by aligning the reception field of each anchor with its span, considering context feature, and applying late fusion on RGB stream and optical flow stream.

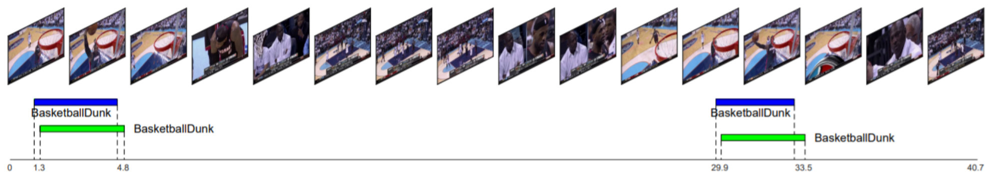

Figure 2.3: Task of temporal action localization: Locate and classify the action[2]

We cannot train a standard temporal action localization model as we did not find a temporal action localization dataset that contains the chest pain class. However, inspired by the fact that most temporal action localization architectures obtain their 1D Spatio-temporal feature map by applying I3D on non-overlapping trimmed video clips, we proposed a novel method that obtains a list of confidence scores of the long video by applying the I3D without final temporal average pooling on non-overlapping trimmed video clips.

[2]TAL-Net's Qualitative examples of the top localized actions on THUMOS'14. Adapted from [39]

# 3. METHODOLOGY

## 3.1 Dataset

The dataset we used was NTU RGB+D dataset[1]. It contains 56,880 video samples which are recorded by 106 subjects under 17 setups using 3 cameras concurrently. Some moments of its video clips are shown in figure 3.1.

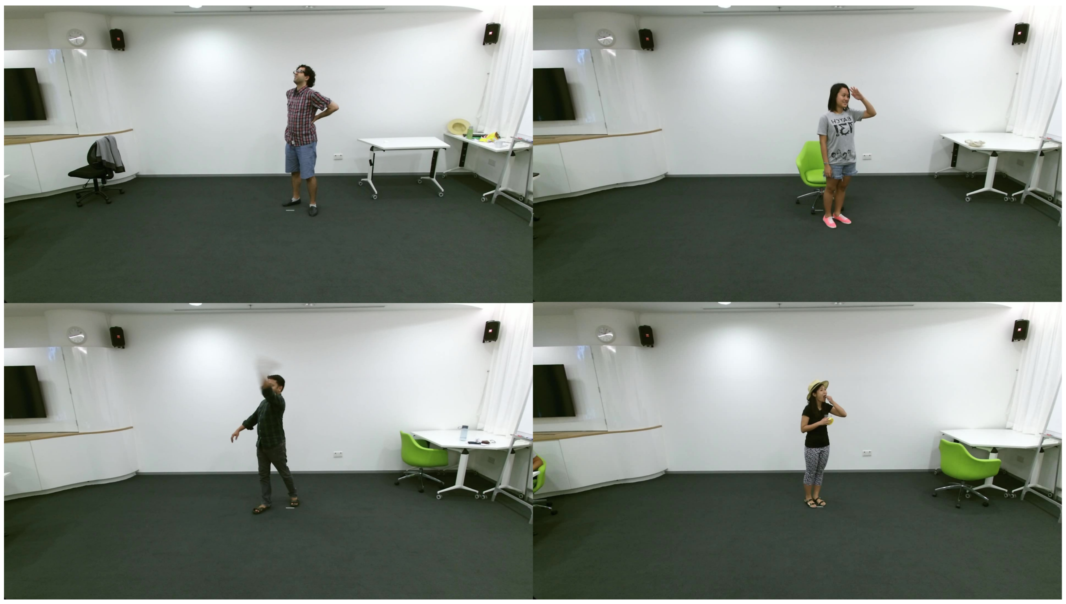
Figure 3.1: Some moments of video clips in NTU RGB+D[1]

There are mainly three types of actions in the dataset: daily actions, mutual actions, and medical condition actions. Our target action—chest pain is included in the medical condition actions. To avoid multi-person distraction during the training process, we excluded all mutual actions and only used the first 49 classes. For each sample, RGB videos, depth map sequences, 3D skeletal data, and infrared (IR) videos are provided. Because our target source is YouTube videos, we only used the RGB videos provided in the dataset. There are two standard types of evaluation for the classification performances, which are cross-subject evaluation and cross-setup evaluation. The subject is the person performing the action, and the setup is the background. To let our model learn more patterns of chest pain, we did the cross-setup evaluation which means that the training set and the testing set will use different recording backgrounds. Videos were split into train set with setup ids 1 to 10, validation set with setup ids 11 to 13, and test set with setup ids 14 to 17.

Usually, we can observe a painful look and moan from a person suffering from chest pain. However, as the chest pain clips from the dataset are not real enough to contain these elements, we used the pose of the person as the only criteria for classification. Also, as there are stomachache videos mixed in the chest pain class of the dataset, we were detecting chest pain and stomachache at the same time.

### 3.1.1 Dataset for frame detectors

To generate the frame dataset for frame detectors, we extracted frames from the latter half part of chest pain clips as positive samples, as the latter half part of chest pain clips is usually where the target action is happening. Although we might introduce incorrectness into the dataset when the target action does not happen in every frame of the latter half part of the chest pain clips, the neural network can still learn the correct pattern when the majority of the training samples are labeled correctly. We extracted one single frame randomly from every video of the other 49 classes as the negative sample. The frame dataset has about 80K samples and is balanced between positive and negative samples. Each sample in the frame dataset is a 1920*1080 resolution image.

### 3.1.2 Dataset for clip detectors

To generate the clip dataset for clip detectors, We used all the chest pain clips in the NTU RGB+D as our positive samples. We randomly selected the same number of clips from the other 49 classes as our negative samples. The clip dataset has about 1800 samples and is balanced between positive and negative samples.

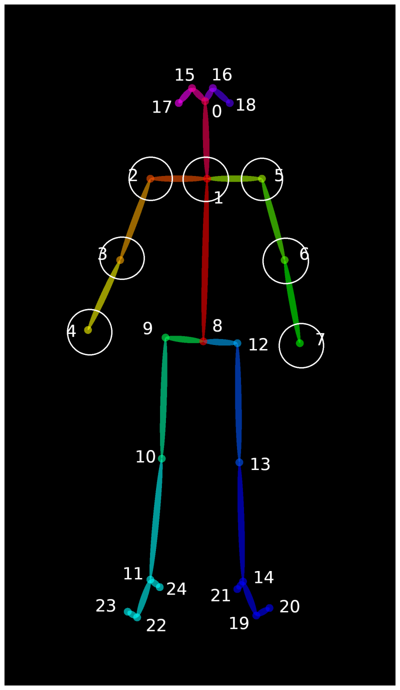

Figure 3.2: The visualization of keypoints output of Openpose[2], with the used keypoints circled[1]

## 3.2 Skeleton-based frame detector

### 3.2.1 Feature extraction

Since YouTube videos do not come with skeleton, data, we did not use the 3D skeleton data provided in the dataset. Instead, we use an excellent pose estimation application to extract raw skeleton data. Openpose[2] is a great application that can extract person keypoints information from the image. The person's key points information is the coordinates of 25 essential body parts of the person, such as head, shoulder, elbow, and hand. All the frames that Openpose failed to find a person will be labeled as false automatically, and will not be used for training. Inspired by [40], we tried to use as few key points as possible. Since we were detecting chest pain, we only focused on the upper body part and use the coordinates of body parts 1 to 7. The used body parts are circled in figure 3.2. Experiments showed that using coordinates of more body parts did not increase the accuracy of the model on the dataset.

[1]The visualization plot of keypoints adapted from `https://github.com/ArtificialShane/OpenPose/blob/master/doc/output.md`

Also inspired by [40], which proposed the pose feature and motion feature consist of joint-joint Euclidean distance and joint-joint orientation, we used the angle and distance information instead of the raw keypoint coordinates as our feature tensor, as shown in figure 3.3. We observed a significant increase in accuracy when we changed from the raw keypoint coordinates to the angle and distance information.

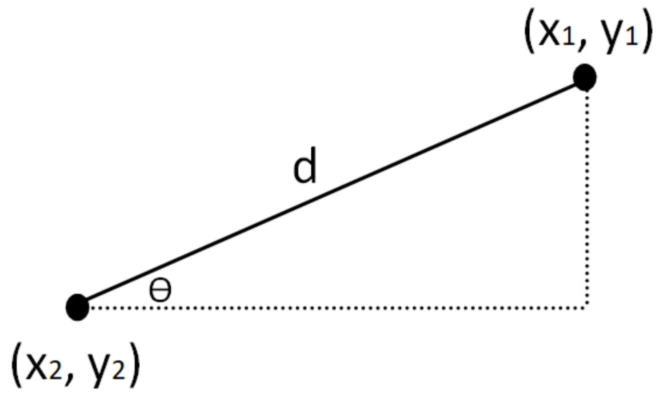

Figure 3.3: Angle and distance information of the keypoint pair

The angle and distance information was obtained by concatenating the angle information and distance information of each keypoint pair. Since we used 7 key points, we had 21 keypoint pairs in total. Given a keypoint pair $(x_1, y_1)$ and $(x_2, y_2)$, the angle information of this keypoint pair is a 2-dimensional vector $(\theta 1, \theta 2)$ that can be calculated as follows.

$$\theta 1 = \frac{x_1 - x_2}{\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}}$$

$$\theta 2 = \frac{y_1 - y_2}{\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}}$$

Although we could get $\theta 2$ directly from $\theta 1$, experiments showed that using both $\theta 1$ and $\theta 2$ can yield a better accuracy on the dataset.

The distance information of this keypoint pair is a 1-dimensional vector ($d$) that can be calculated as follows, where $d_{max}$ is the maximum distance among 21 keypoint pairs.

$$d = \frac{\sqrt{(x_1 - x2)^2 + (y_1 - y_2)^2}}{d_{max}}$$

For each person, there are 21 keypoint pairs. For each keypoint pair, the angle information is 2-dimensional and the distance information is 1-dimensional. Therefore, the feature vector after concatenation is 63-dimensional.

The angle and distance feature is invariant to the location and size of the person, which is easier to generalize. In the latter section we will show that training with our proposed feature yields better validation accuracy, test accuracy and faster convergence speed.

### 3.2.2 Network architecuture

The classifier is simply a sequential model with 2 dense layers. The architecture of the classifier is shown in figure 3.4.

### 3.2.3 Training

During training, we used adam optimizer with a 1e-3 learning rate to minimize the binary cross-entropy loss. There are 733 steps within an epoch with batch size set to 64. The runtime of each step is 26ms on google colab.

The selection of parameters such as batch size and learning rate can greatly impact the model's accuracy. For example, if the batch size is too low, then the training speed will be very slow, and the gradient update will have a lot of uncertainty. If the batch size is too high, then training requires a lot of memory, and the model might not be able to escape from the local minima.

We saved the model with the best validation accuracy after each epoch. The plots of validation accuracy and loss are shown in figure 3.5. After 20 epochs of training, the best model achieves 87.39% validation accuracy and 85.39% test accuracy. The high accuracy demonstrates that both the frame dataset and our proposed model are working.

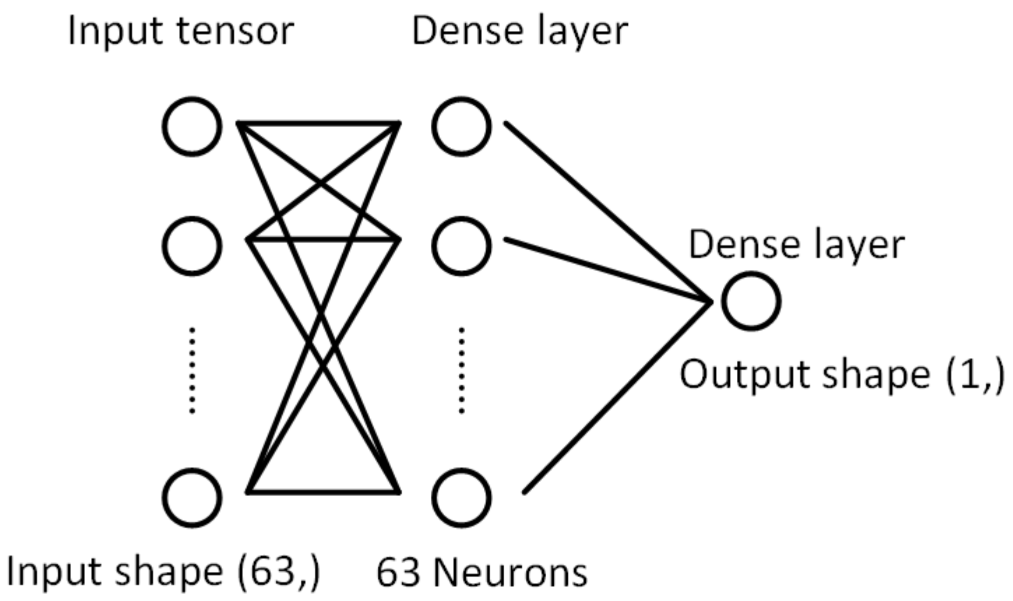

Figure 3.4: The architecture of the classifier for skeleton based frame detector

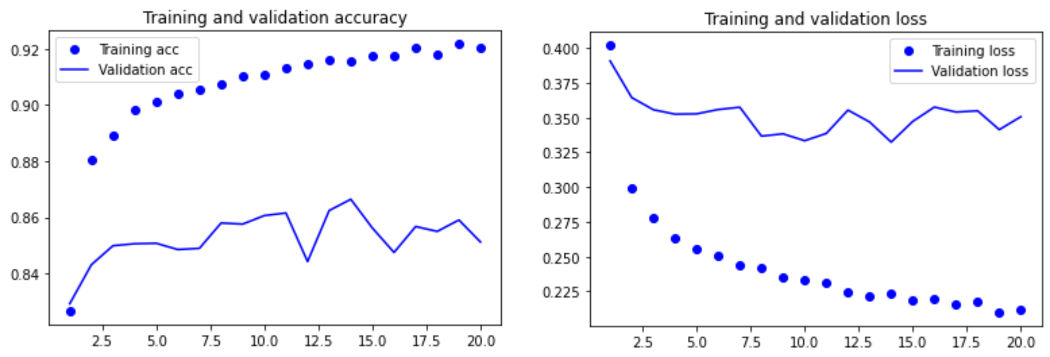

Figure 3.5: The plots of training accuracy vs. validation accuracy(left) and training loss vs. validation loss(right) for skeleton-based frame detector

### 3.2.4 Detector API

After training, we have a trained model that takes the skeleton data of a person as input and returns a confidence score. However, what we want is that we take in a video and return a result video containing the visualized results. Therefore, we need to implement a detector API to do that.

The flow of the detector API is shown in figure 3.6. Multiple-person chest pain detection is supported since Openpose[2] automatically detects every person in the frame. Because we only had the key points of each person in each frame, we drew the bounding box to just large enough to include all the key points of the person. Therefore, the drawn bounding boxes are usually smaller than the ground truth bounding boxes. For each person in each frame, we extract key points with Openpose. We generate feature tensor and infer with our pre-trained model to get a confidence score. We approximate a bounding box with the key points. Finally, we draw the bounding box and score on the frame.

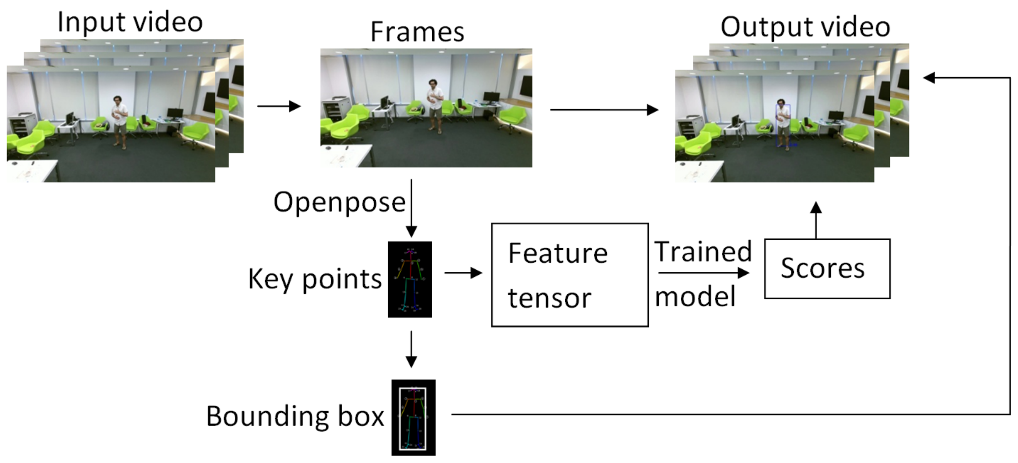

Figure 3.6: The flow of the Skeleton based frame detector API[2]

## 3.3 RGB-based classification frame detector

### 3.3.1 Feature extraction

We used the RGB pixel array of shape (1080,1920,3) as our input feature. We normalize the pixel value of each element in the RGB array from (0,255) to (-1,1) using the standard Inception Resnet V2[10] preprocess function. During training, we augmented the training data by randomly applying horizontal flip and rotation up to 20 degrees.

[2]Frame image adapted from NTU RGB+D dataset, visualization plot of key points adapted from `https://github.com/ArtificialShane/OpenPose/blob/master/doc/output.md`

### 3.3.2 Network architecture

The network architecture is shown in figure 3.7. We used the convolution base of Inception Resnet V2 for spatial feature extraction, then followed by a simple fully connected layer with a sigmoid activation function that acts as the classifier.

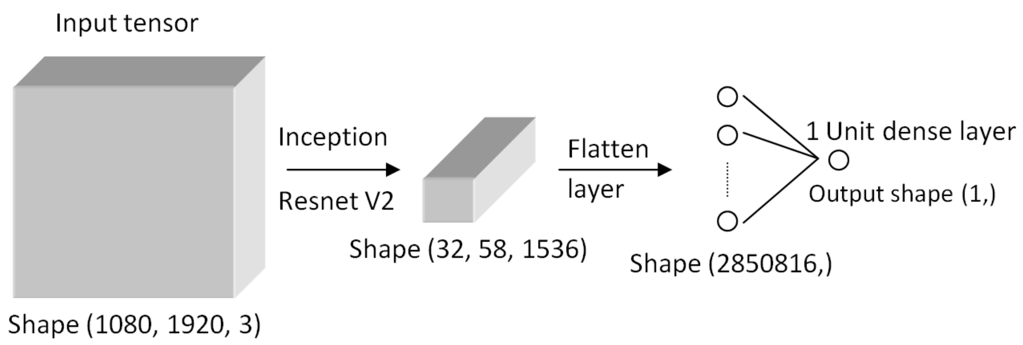

Figure 3.7: The network architecture of the RGB-based classification frame model

### 3.3.3 Training

We trained the weights of the pre-trained Inception Resnet V2[10] and the simple classifier jointly.

During training, we used adam optimizer with a 1e-5 learning rate to minimize the binary cross-entropy loss. There are 20702 steps within an epoch with batch size set to 2, as a higher batch size would cause the resource exhaust error. The runtime of each step is 1s on google colab. After 2 epochs of training, the model achieves 88.09% validation accuracy and 87.83% test accuracy. It demonstrates that the Inception Resnet V2 can capture the unique features of chest pain.

## 3.4 RGB-based localization frame detector

### 3.4.1 Feature extraction

We used the pre-trained faster RCNN[3] model to obtain the bounding boxes of the person as we found it to be very accurate both on the dataset and youtube videos. We did not fine-tune the weight of faster RCNN on our small dataset for the fear that it may harm the generalizability of the model. All the frames that the faster RCNN failed to find a person will be labeled as false automatically, and will not be used for training. During training, we augmented the training data by randomly cutting off the bottom half part of the bounding boxes because usually the useful information to infer chest pain is located at the upper body of the person. By augmenting the data, the trained model showed better performance when classifying a person with partial body parts. We also extracted the RGB array of the original image as our input feature. The bounding box is a 4-dimensional vector representing the coordinates of the upper right and lower left of the bounding box. The RGB array is of shape (1080,1920,3). In the end, we normalize the pixel value of each element in the RGB array from (0,255) to (-1,1) using the standard Inception Resnet V2[10] preprocess function.

### 3.4.2 Network architecture

The model architecture follows the classifier part of the faster RCNN[3], as shown in figure 3.8. We first extract the feature map from the RGB array using the Inception Resnet V2[10] pre-trained on ImageNet. Then we crop the feature map with the bounding box and apply an RoI layer to get a fixed size feature. Finally, we use a simple classifier that only consists of one dense layer to make a decision.

### 3.4.3 Training

We trained the weights of the pre-trained Inception Resnet V2[10] and the simple classifier jointly, as experiments showed it yields better performance compared to only training the simple classifier.

During training, we used adam optimizer with a 1e-5 learning rate to minimize the binary

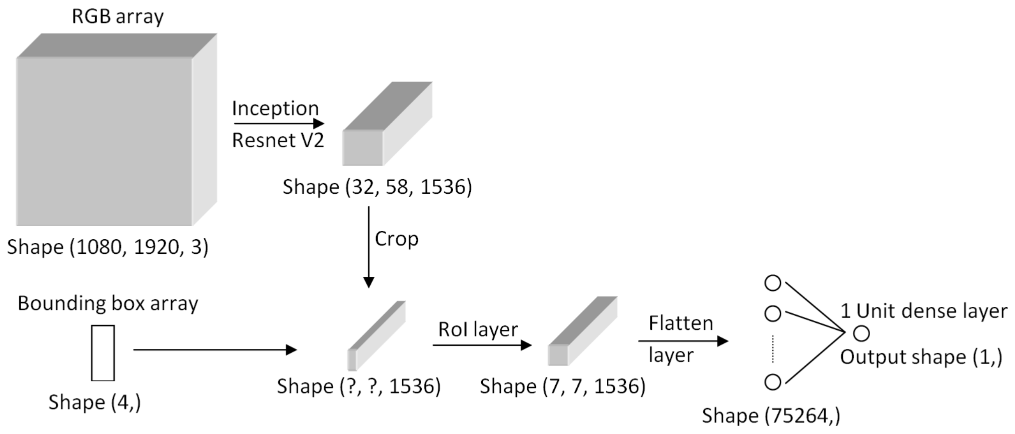

Figure 3.8: The architecture of the model for RGB localization-based frame detector

cross-entropy loss. There are 23455 steps within an epoch with batch size set to 2, as a higher batch size would cause the resource exhaust error. The runtime of each step is 961ms on google colab. After 1 epoch of training, the model achieves 84.75% validation accuracy and 79.41% test accuracy. Its accuracy is lower than the previous model, which shows that cropping the feature map does affect the performance to some extent.

### 3.4.4 Detector API

The flow of the detector API is shown in figure 3.9. Multiple-person chest pain detection is supported since the faster RCNN[3] automatically detects every person in the frame. For each human in each frame, we use the pre-trained faster RCNN to extract the human bounding box. Our custom model takes in the RGB array and the human bounding box and generates a score. Finally, we draw the bounding box and score on the frame.

## 3.5 Skeleton-based clip detector

### 3.5.1 Feature extraction

Remember that in the skeleton-based frame detector, we extract the 63-dimensional feature vector containing angle and distance information from the skeleton data of one person in one frame. The feature extraction of the skeleton-based clip detector is nearly the same as the skeleton-

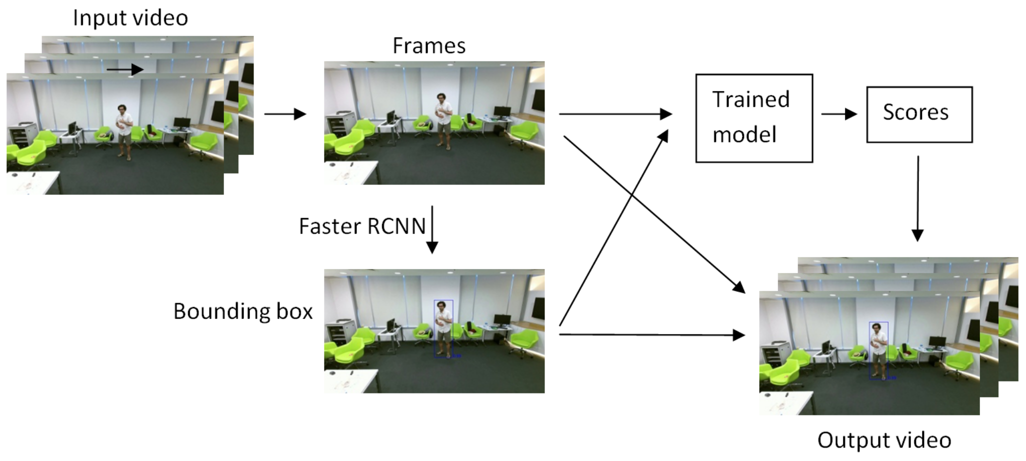

Figure 3.9: The flow of the RGB based localization frame detector API[3]

based frame detector, except that we concatenated the feature vector of each frame in the end. We first extracted frame sequences from the video clips at 10fps, then we fixed the length of frame sequences to 30. To fix the length of the frame sequence to 30, we first remove all frames that Openpose[2] fails to find a person. If the length of the frame sequence is 0, the clip will be labeled as false automatically, and will not be used for training. If the length of the frame sequence is less than 30, we repeat the frame sequence multiple times. If the length of the frame sequence exceeds 30, we take the 30 frames at the center of the frame sequence. Then we calculated the angle and distance information of each frame in the sequence and concatenated them together. The shape of the output feature is (30,63).

### 3.5.2 Network architecture

Inspired by [41] whose RNN architecture achieves high accuracy on NTU RGB+D dataset using the 3D skeleton data provided in the dataset, the architecture of our model is a sequential model that consists of two 100-unit stacked bi-directional GRU layers for temporal feature extraction, a dropout layer with 0.25 dropout rate to reduce overfitting, and two dense layers for classification, as shown in figure 3.10.

[3]Frame image adapted from NTU RGB+D dataset

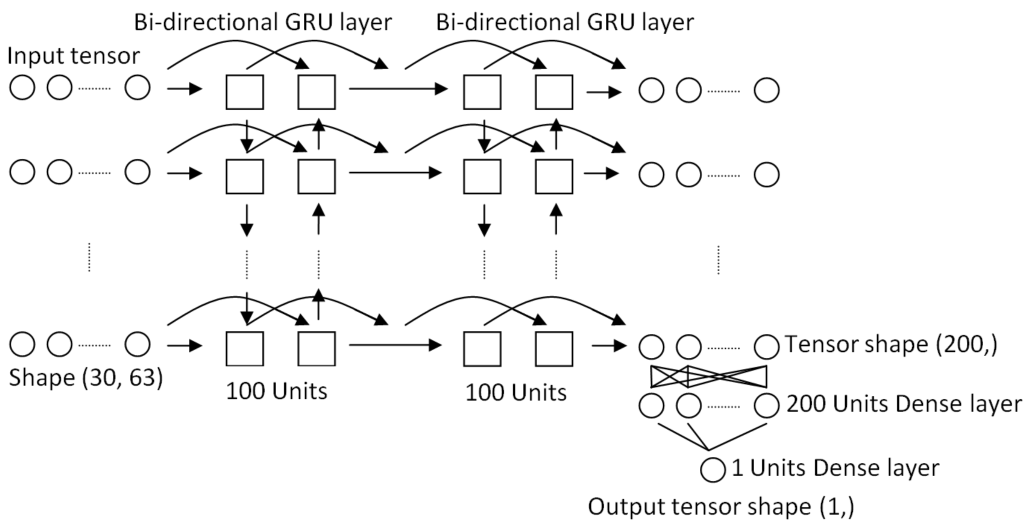

Figure 3.10: The architecture of the model for skeleton-based clip detector

### 3.5.3 Training

During training, we use adam optimizer with a 1e-4 learning rate to minimize the binary cross-entropy loss. There are 33 steps within an epoch with batch size set to 32. The runtime of each step is 11ms on google colab. The plots of validation accuracy and loss are shown in figure 3.11. After 80 epochs of training, the model achieves 87.3% validation accuracy and 84.5% test accuracy. Surprisingly, we do not observe a significant accuracy increase from the frame models to clip models.

### 3.5.4 Detector API

The flow of the detector API is the same as the frame detector. We used the sliding window approach to obtain the feature array of each time step. The length of the sliding window is 3s. For example, the prediction for time step 3s is the inference result of the video clip from 1.5s to 4.5s. As a result, the detector may pre-respond or post-respond to the chest pain action. Despite that the inference result of a time step is obtained by looking at a sequence of surrounding frames, the bounding box is still calculated from the key point information of the current frame. Because of that, there are cases where the inference result is positive while Openpose[2] fails to find the

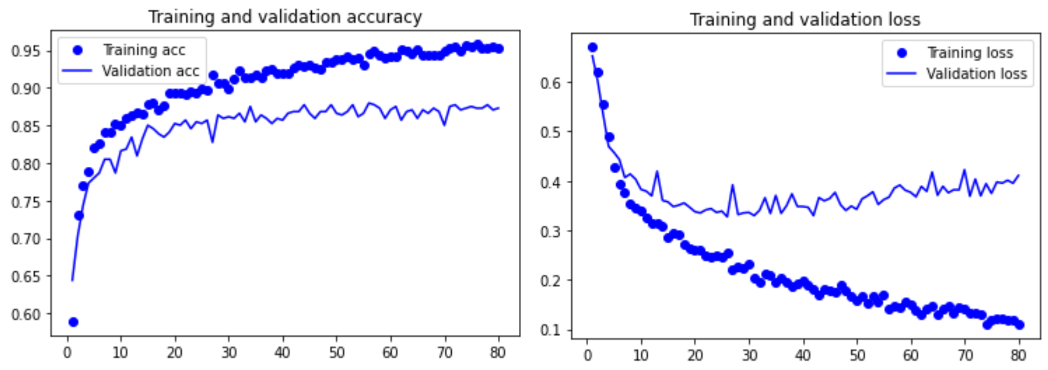

Figure 3.11: The plots of training accuracy vs. validation accuracy(left) and training loss vs. validation loss(right) for skeleton-based clip detector

person for the current frame. In this case, we would still label this time step as negative. Also, multiple person detection is not supported by the program, since Openpose does not have a tracking functionality yet. As a result, currently, we just tracked the keypoint sequence by selecting the person with the highest confidence score.

## 3.6 RGB-based clip detector

### 3.6.1 Feature extraction

We extracted the RGB array from each frame, resize the RGB array from 1920x1080 to 453x256, and concatenated them together to form our initial input feature.

During training, we augmented the training data by applying the 224x224 random space crop, the random temporal crop which cropped out up to one-tenth frames, the random rotation up to 10 degrees, and the random horizontal flip. During validation, testing, and inferring, we only applied 224x224 center space crop.

Next, we fixed the length of the sequence to 30. If the length of the sequence is less than 30, we pad the sequence with arrays of 0s. If the length of the sequence exceeds 30, we take the 30 frames at the center of the sequence.

In the end, we normalize the RGB pixel values in the input feature array from (0,255) to (-1,1) by dividing 128 and subtracting 1. The shape of the finalized input feature is (30,224,224,3).

### 3.6.2 Network architecture

The network architecture is slightly different from the original I3D since we wanted to output a list of confidence scores instead of a single confidence score to minimize our temporal footprint which is the interval between two consecutive predictions. Figure 3.12 shows a simplified network architecture of I3D.

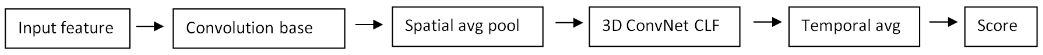

Figure 3.12: Simplified network architecture of I3D

As discussed in the previous subsection, the finalized input feature is (30,224,224,3). The convolution base output of I3D has shape (4,7,7,1024) which is a temporal sequence of spatial feature maps. After the spatial 3D average pooling layer with (2,7,7) filter size, the features of different spatial locations in the spatial feature map are averaged together, and the output shape becomes (3,1,1,1024). Then a 3D convolution layer with 1 filter of size (1,1,1) is used as a classifier to make a decision for each spatial-averaged feature map. Its output has shape (3,1,1,1) which is a list of predictions. Finally, the list of predictions is averaged and a single prediction is obtained.

In [39], the spatial 3D average pooling layer output of I3D was extracted as the Spatio-temporal feature, and a well-performed temporal action localization model was trained on top of it. Inspired by its idea, we extracted the 3D convolution classifier output of I3D as our prediction result. Figure 3.13 shows the architecture of the model for RGB-based clip detector.

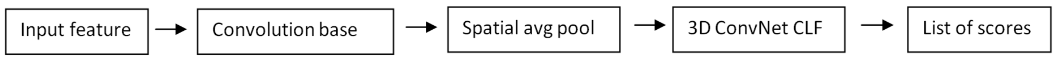

Figure 3.13: The architecture of the model for RGB-based clip detector

### 3.6.3 Training

During training, we were still fine-tuning the original I3D architecture. We used adam optimizer with a 1e-4 learning rate to minimize the binary cross-entropy loss. There are 69 steps within an epoch with batch size set to 16. After 11 epochs of training, the model achieves 90.7% validation accuracy and 86.2% test accuracy. The plots of validation accuracy and loss are shown in figure 3.14.

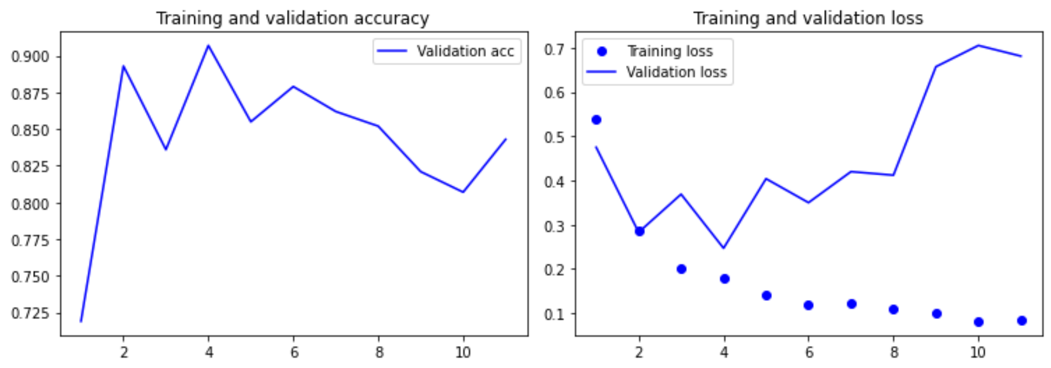

Figure 3.14: The plots of training accuracy vs. validation accuracy(left) and training loss vs. validation loss(right) for RGB based clip detector

### 3.6.4 Detector API

About how we used the trained I3D to obtain a list of confidence scores, as shown in figure 3.15, we separated the input long video into non-overlapping 3s video clips. After inference, we concatenate each output confidence scores list with length 3 to form our final confidence score list. For example, if the input video is 9s long, it will be divided into 3 video clips 0-3s, 3-6s, and 6-9s. The inference output of the video clip 3-6s is a list of confidence scores with length 3. The first element in the list is the label for 3-4s part of the original long video, the second element in the list is the label for 4s-5s part of the original long video and so on.

To obtain the Spatio-temporal feature for temporal action localization, [39] cut the long video into non-overlapping video clips, extracted 1024-d feature vector for each video clip using I3D,

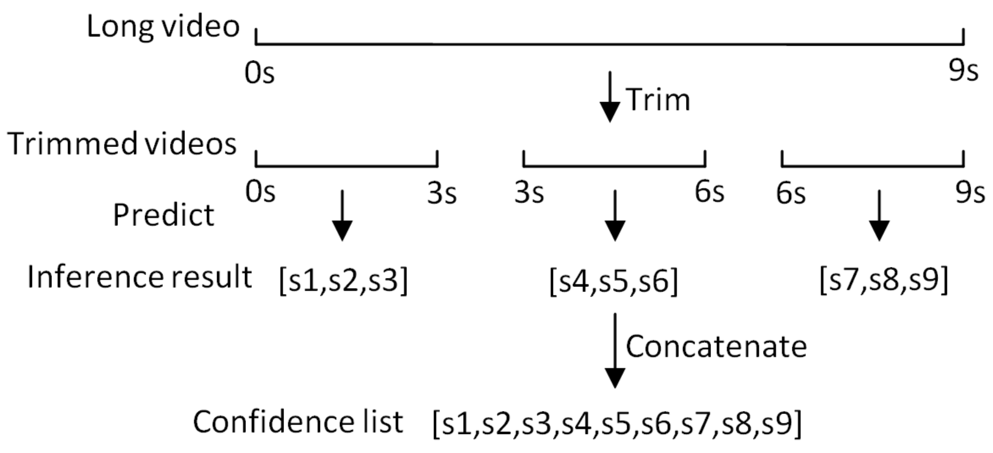

Figure 3.15: The flow of the RGB-based clip detector API

and concatenated all feature vectors. Therefore, we thought it is reasonable to get the confidence scores vector of the long input video using a similar approach.

Because I3D output includes neither bounding boxes nor key point information, we could only draw the confidence scores without the bounding boxes on our output video. Because each element in the output confidence scores list has a reception field much longer than 1s, the detector may pre-respond or post-respond to the chest pain action just like the skeleton-based clip detector.

# 4. PERFORMANCE EVALUATION

## 4.1 Performance evaluation of the trained models on the dataset

### 4.1.1 Statistic evaluation

Table 4.1 shows the accuracy, precision, and recall of 5 trained models on the dataset. The first three models are evaluated on the test set of the dataset for frame detectors, while the last two models are evaluated on the test set of the dataset for clip detectors. All the test sets are balanced between positive and negative samples.

| Model Name | Accuracy | Precision | Recall |
|---|---|---|---|
| Skeleton-based frame feedforward NN | 85.2% | 86.3% | 83.6% |
| RGB-based classification frame Inception Resnet | 87.8% | 96.6% | 78.4% |
| RGB-based localization frame faster RCNN[3] | 79.3% | 98.4% | 59.6% |
| Skeleton-based clip RNN | 84.2% | 86.0% | 83.3% |
| RGB-based clip I3D[4] | 87.8% | 80.3% | 100% |

Table 4.1: Accuracy, precision, and recall of 5 trained models

For the frame models, we can see that the RGB-based models all share impressive precision and relatively low recall, while the skeleton-based feedforward NN is balanced between precision and recall. We can also see that the RGB-based localization model has lower accuracy than the RGB-based classification model, mainly because of the information lost during cropping the spatial feature map.

For the clip models, the RGB-based I3D has higher accuracy than the skeleton-based RNN. The skeleton-based RNN is again balanced between precision and recall. The RGB-based I3D has an impressive 100% recall, which means that all the chest pain cases are classified correctly.

We cannot directly compare the results of frame models and clip models, since they are using different datasets. However, we can get a sense that frame models are working almost as well as clip models, which confirms our assumption that chest pain can usually be recognized by a single frame.

Table 4.2 shows the accuracy and the number of epochs to convergence for the skeleton-based models that apply the proposed feature extraction method or only the basic normalization which normalizes all coordinates to the range of [0,1]. It demonstrates that by applying the proposed feature extraction method, the skeleton-based models have better accuracy and faster convergence speed.

| Model Name | Accuracy | Epochs | Feature extraction | Learning rate |
|---|---|---|---|---|
| Skeleton based frame feedforward NN | 75.0% | 32 | false | 1e-3 |
| Skeleton based frame feedforward NN | 85.2% | 14 | true | 1e-3 |
| Skeleton based clip RNN | 72.1% | 302 | false | 1e-4 |
| Skeleton based clip RNN | 84.2% | 56 | true | 1e-4 |

Table 4.2: The accuracy and the number of epochs to convergence for the skeleton based models that apply or do not apply the proposed feature extraction method

Table 4.3 shows the accuracy and the largest training batch size possible on google Colab for the Inception Resnet with different input image resolutions. It demonstrates that training and inference are much faster when reducing the resolution of the input image at the cost of accuracy.

| Resolution | Accuracy | Batch size |
|---|---|---|
| 640x360 | 76.3% | 16 |
| 1280x720 | 87.1% | 4 |
| 1920x1080 | 87.8% | 2 |

Table 4.3: The accuracy and the largest training batch size possible on google Colab for the Inception Resnet with different input image resolutions

### 4.1.2 False cases analysis

#### *4.1.2.1 Skeleton based models*

Because skeleton-based models all rely on Openpose for skeleton data extraction, we analyzed the false cases of frame feedforward NN and clip RNN together. We first analyzed the false-negative cases, then false-positive cases.

About the false-negative cases, we calculated the ratio of frames that chest pain is not detected in a chest pain video clip to evaluate the frame feedforward NN since the two models are originally evaluated on different datasets. There are 7589 chest pain frame samples in the test set of the frame dataset extracted from 192 chest pain video clips. 101 out of 192 video clips have all the chest pain frame samples correctly classified, which means that the chest pain is correctly detected. 91 out of 192 video clips have at least one misclassified frame. Figure 4.1 is a histogram that shows the distribution of the ratio of undetected frames among all the 192 video clips in the test set.

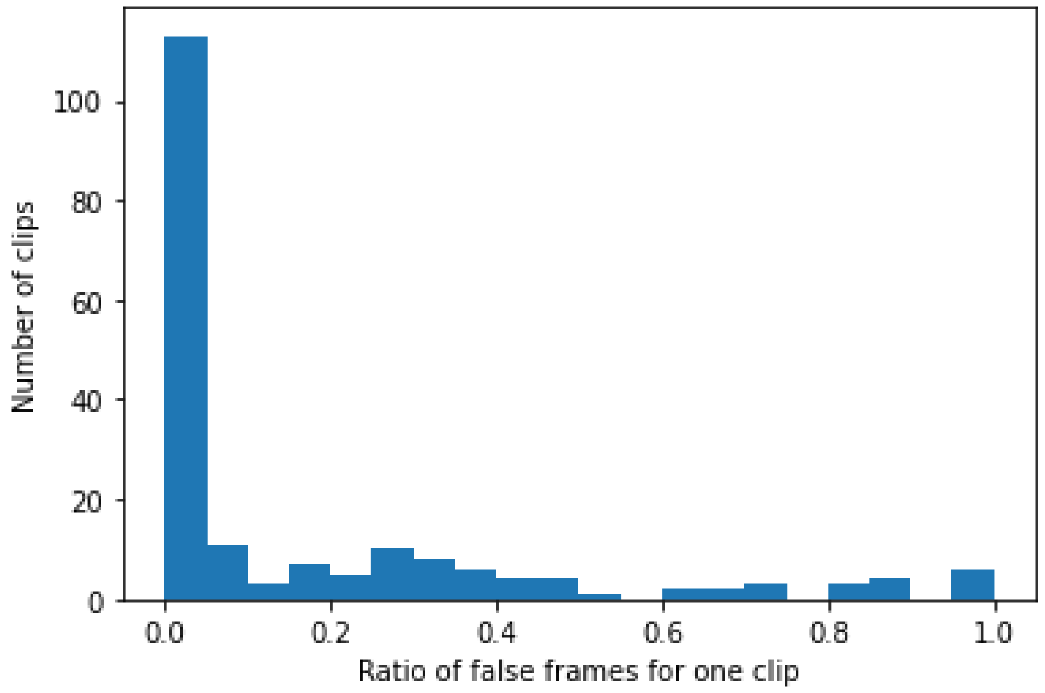

Figure 4.1: The distribution of the ratio of wrongly classified frames for Skeleton-based frame feedforward NN.

From the histogram, we can see that most of the video clips only have a small ratio of chest pain frames undetected, and only very few 'hard' clips have nearly all chest pain frames undetected. If we make a simple decision by assigning the video clip chest pain class if chest pain is detected over have of its frames, we would get 89.1% recall, even higher than the recall of the clip RNN model. It makes sense since we only extracted the frames of the latter half part of the chest pain clips for the frame feedforward NN, and the former half part of the chest pain clips might confuse the clip RNN.

We also observed some correlations between the chest pain cases that the two models perform well and cases that the two models perform poorly. As we have stated before, there are 101 chest pain clips that the frame feedforward NN successfully classified every frame. Among those 101 chest pain clips, 96 of them are also correctly classified by the clip RNN. It demonstrates that there are lots of universal easy clips between the frame model and the clip model. There are 6 chest pain clips that the frame feedforward NN fails on all the frames. 5 of them were also misclassified by the clip RNN model with a very low confidence score(less than 5.7%). It demonstrates that there are some universal hard chest pain clips between the frame model and the clip model, which can be caused by some factors other than the network architecture. Figure 4.2 shows the original frame and the frame with the skeleton of one of the universal hard clips.

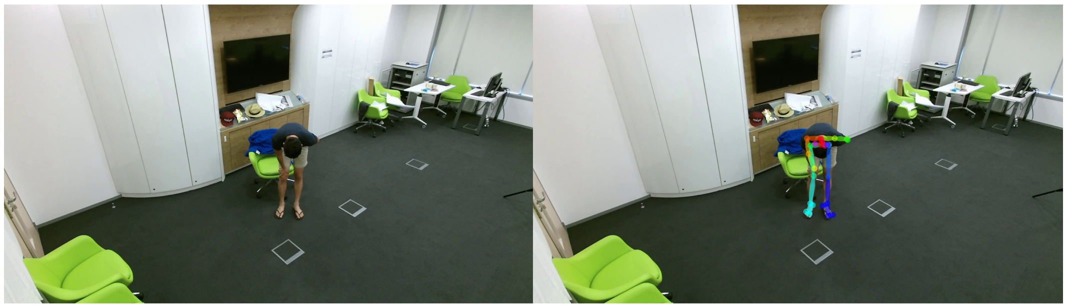
Figure 4.2: Original frame(left) and the frame with skeleton(right) for a false negative case of both skeleton-based models[1]

[1]Images from dataset NTU RGB+D

In the frame, the man is bending over with his left hand pressing the chest. Due to the recording angle, his left elbow and left hand which is the critical clue to detecting chest pain are occluded. The output of Openpose shows that the confidence of the left elbow is 39.8%, the confidence of the left hand is as low as 6.7%, and the position of the left hand is also incorrect. Therefore, incorrect Openpose output is one of the main factors that cause the universal hard clips for skeleton-based models.

There are also some chest pain clips that one of the two models classifies correctly while the other model fails. It could be because the model fails to learn some chest pain patterns during the training process.

We also looked into some false positive cases where a negative clip is classified as chest pain. Figure 4.3 shows the original frame and the frame with the skeleton of one of them. In the frame, the person is going to write on paper. Because we only used the skeleton data, we cannot detect the paper and pen in the frame which is critical information for classification. Also, we cannot decide whether his left hand is pressing on the chest or hovering above the chest since the skeleton data is 2D. Therefore, useful information that the 2D skeleton data missed, such as action-associated objects and 3D information, is one of the main factors that caused the misclassification.

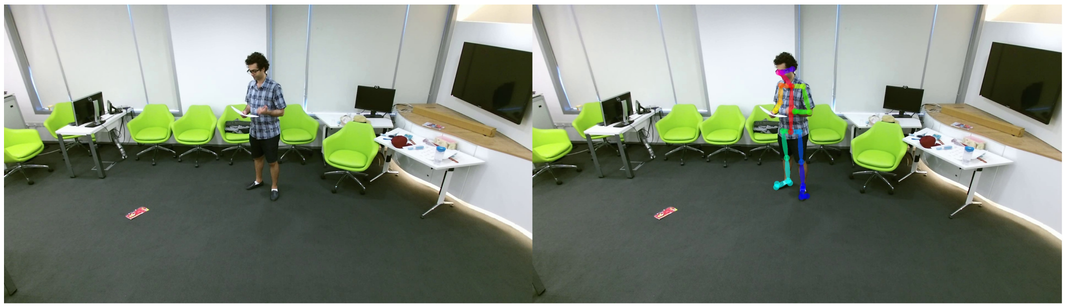

Figure 4.3: The original frame(left) and the frame with skeleton(right) for a false positive case of Skeleton based models[2]

[2]Images from dataset NTU RGB+D

#### *4.1.2.2 RGB-based frame models*

Because RGB-based frame models all rely on Inception Resnet for spatial feature map extraction, we analyzed their false cases together. There are 7589 chest pain frame samples in the test set of the frame dataset extracted from 192 chest pain video clips. 95 out of 192 video clips have all the chest pain frame samples correctly classified by the classification model, while 67 out of 192 video clips have all the chest pain frame samples correctly classified by the localization model. Figure 4.4 is a histogram that shows the distributions of the ratio of undetected frames among these 192 video clips.

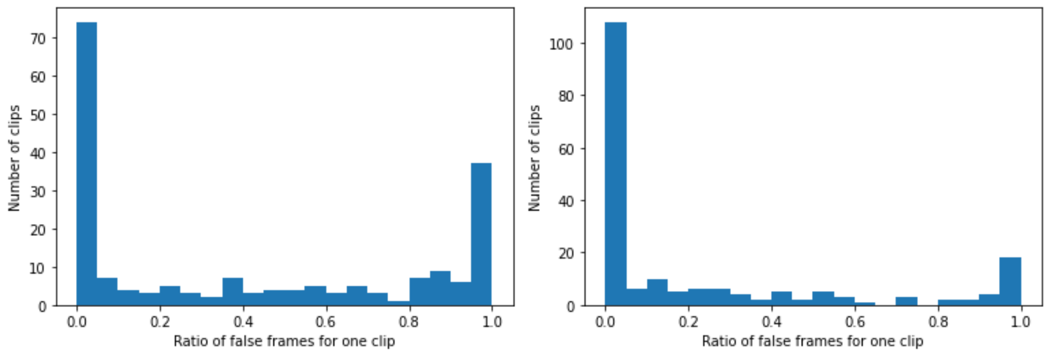

Figure 4.4: Distribution of ratio of wrongly classified frames for RGB-based frame faster RCNN(left) and Inception Resnet(right)

From the histogram, we can see that the majority of chest pain clips either have almost all their frames successfully classified or have almost all their frames misclassified. 17 out of 192 video clips have all the chest pain frame samples misclassified by the localization model, while 31 out of 192 video clips have all the chest pain frame samples misclassified by the localization model. 60 video clips have all the chest pain frame samples correctly classified by both models, while 16 video clips have all the chest pain frame samples misclassified by both models. The fact that almost all video clips missed by the classification model are missed by the localization model, and almost all video clips detected by the localization model are detected by the classification model demonstrates there is a huge correlation between the prediction result of the two models, and the classification model is better on the test set of the frame dataset. About the 5 universal hard clips for skeleton-based models that we previously discussed, 4 of them have over half of their frames misclassified by the two RGB based frame models, which proves that universal hard clips for skeleton-based models are still hard for the RGB based frame models, and body parts missing still greatly affected the accuracy of the RGB based models. Also, one thing to note is that in the previous model, incorrect Openpose output is one of the main factors that cause false-negative cases. While in the current model, there is no such concern since the pre-trained faster RCNN can give the correct bounding box to almost all frames.

On the other hand, the RGB-based frame models have a lot fewer false-positive cases. Figure 4.5 shows some false positive sample frames with bounding boxes. The left figure shows a nauseous person. It is possible that the model misinterprets the depth location of his left hand, and thinks it is pressing the chest instead of hovering above the chest. Many of the false-positive cases are nausea class. The right figure shows a person hopping. Perhaps the model takes the upper body into consideration while overlooks the lower body which contains the critical information.

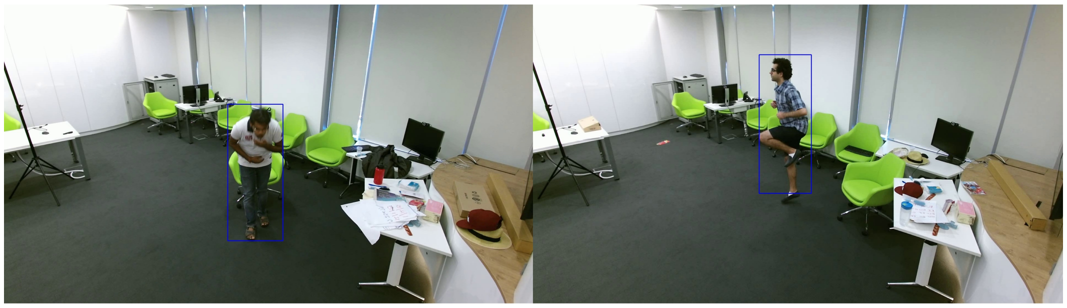

Figure 4.5: Some false positive sample frames of RGB based frame faster RCNN with bounding boxes[3]

[3]Images from dataset NTU RGB+D

## 4.2 Performance evaluation of the detectors

### 4.2.1 Detectors on the dataset

#### *4.2.1.1 Frame detectors*

For the frame detectors, we found that they can precisely locate the chest pain action in most of the clips in the test set. Figure 4.6 shows the plots of time vs. confidence score on a chest pain clip in the test set plotted by the frame detectors.

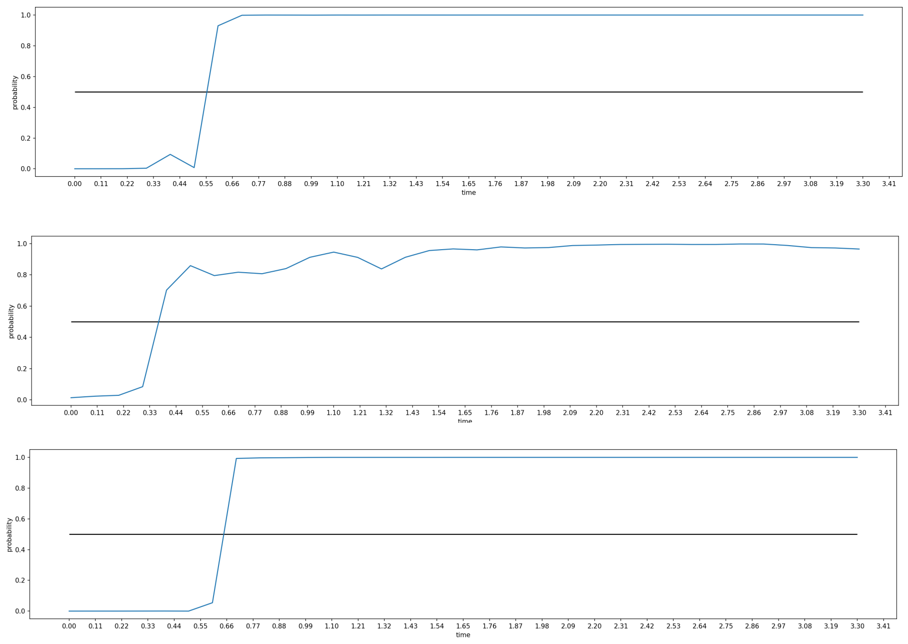

Figure 4.6: Plot of the RGB-based classification frame detector(top), the plot of the skeleton-based frame detector(middle) and the plot of the RGB-based localization frame detector(bottom)

The ground truth chest pain starting time of this clip is about 0.5s. We can see that all the frame detectors find the approximately correct start time of chest pain, with the start time of the RGB-based frame detectors a little bit later probably because the hand is blurred when the person is raising his hand to the chest as shown in figure 4.7. It could be because Openpose is better to identify the blurred body parts due to motion than our trained RGB-based detectors. The result

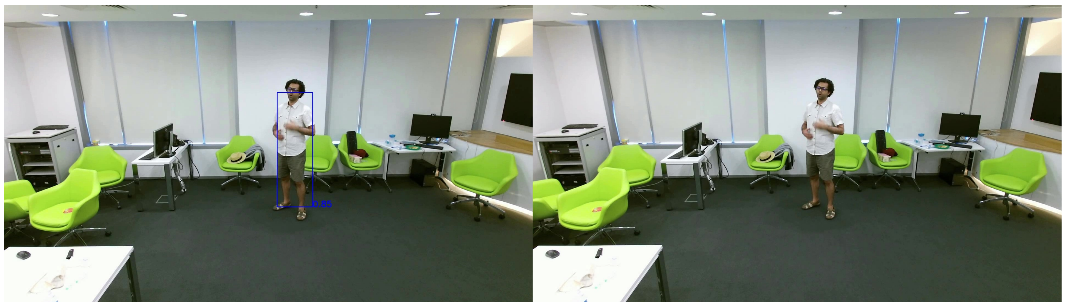

Figure 4.7: The person with a blurred hand is detected by the skeleton-based frame detector(left) and missed by the RGB-based frame detectors(right)[4]

is impressive since our dataset is only labeled clip-wise and we trained our detectors based on the assumption that the latter half part of the clip contains chest pain action. It demonstrates that the frame detectors can be very accurate under good conditions.

#### *4.2.1.2 Clip detectors*

On the other hand, the clip detectors cannot locate the chest pain action in the dataset. The average length of the chest pain clips is about 3s, and the part before the chest pain action is usually shorter than 0.5s. Both detectors look at a 3-s clip at a time, so they cannot even notice the part before the chest pain action.

### 4.2.2 Detectors on YouTube videos

The YouTube videos are generally hard for the detectors, as we have discussed in the first section. The level of difficulty of each YouTube video also differs significantly from each other. Some might be nearly as easy as the clips in the dataset, while others might never be correctly classified with solely the knowledge of the 900+ chest pain clips in the dataset. Moreover, the chest pain videos are rare on YouTube since nobody enjoys watching a person having chest pain. Due to the above reasons, the framewise or clip-wise accuracy, precision, or recall usually cannot reflect the real performance of the detectors. Due to the above reasons, we are not going to focus on the framewise or clip-wise accuracy, precision, or recall as they usually cannot reflect the real

[4]Images from dataset NTU RGB+D

performance of the detectors.

#### *4.2.2.1 Skeleton-based frame detector*

Figure 4.8 is one frame extracted from one result YouTube video of the skeleton-based frame detector. From it, we can see that multiple-person detection is supported by the detector as we have stated in the detector API section. The detection of the man in gray at the center of the frame shows that although we trained our model on the dataset with full human body parts, we can still correctly detect a person with partial body parts, although the correctness may suffer from time to time. In the meantime, the two false detections on the right side of the frame show that the detector is very weak at handling persons that do not face front to the camera, since Openpose would not get a correct location for the essential body parts such as hands and shoulders.

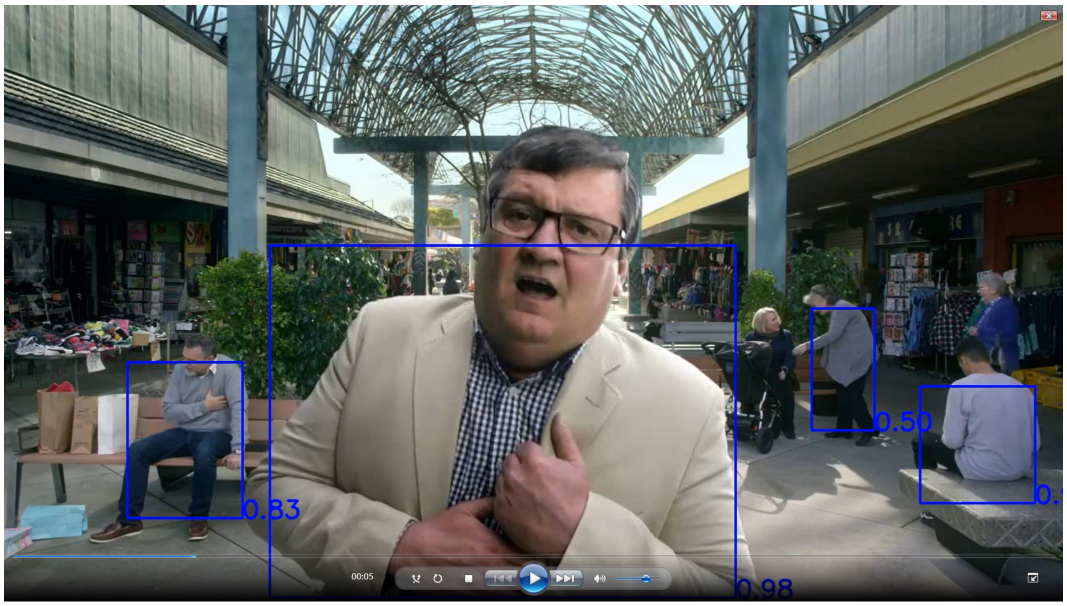

Figure 4.8: One frame extracted from one result YouTube video of the skeleton-based frame detector[5]

Figure 4.9 shows the comparison between the resulting plot and the ground truth plot labeled by humans of one YouTube video. From it, we can see that the detector covers almost all the chest

[5]Image from YouTube video `https://www.youtube.com/watch?v=5CGRe6doF4k`

pain action instances, but is very unstable and has low precision.

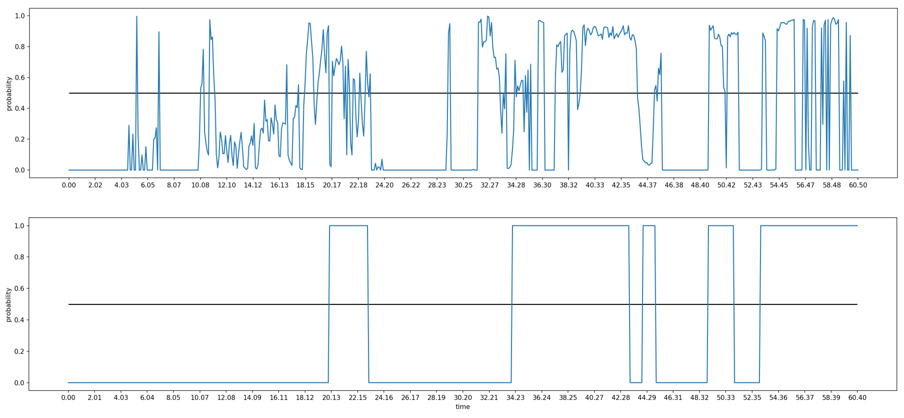

Figure 4.9: The comparison between the resulting plot(top) and the ground truth plot labeled by humans(bottom) of one YouTube video

The runtime of the detector mainly depends on the skeleton data extraction speed of Openpose which is very fast. For a 174s long youtube video with 1280x720 resolution, the runtime of the detector is 176s when setting fps to be 10.

#### *4.2.2.2 RGB-based classification frame detector*

Because the base model Inception Resnet is an image classification model, we have no way to know the location of the people and which person is having chest pain with the RGB-based classification frame detector. All we can get at one timestamp is just a confidence score, as shown in figure 4.10.

The detector performs especially well on one YouTube video, in which all the chest pain instances are correctly detected. However, it hardly detects anything in other YouTube videos despite its good performance on the frame dataset, as shown in figure 4.11. We can observe that the detector has very strict conditions in order to perform correctly.

The runtime of the detector mainly depends on the computation speed of the deep convolution base of the Inception Resnet. For a 174s long youtube video with 1280x720 resolution, the runtime

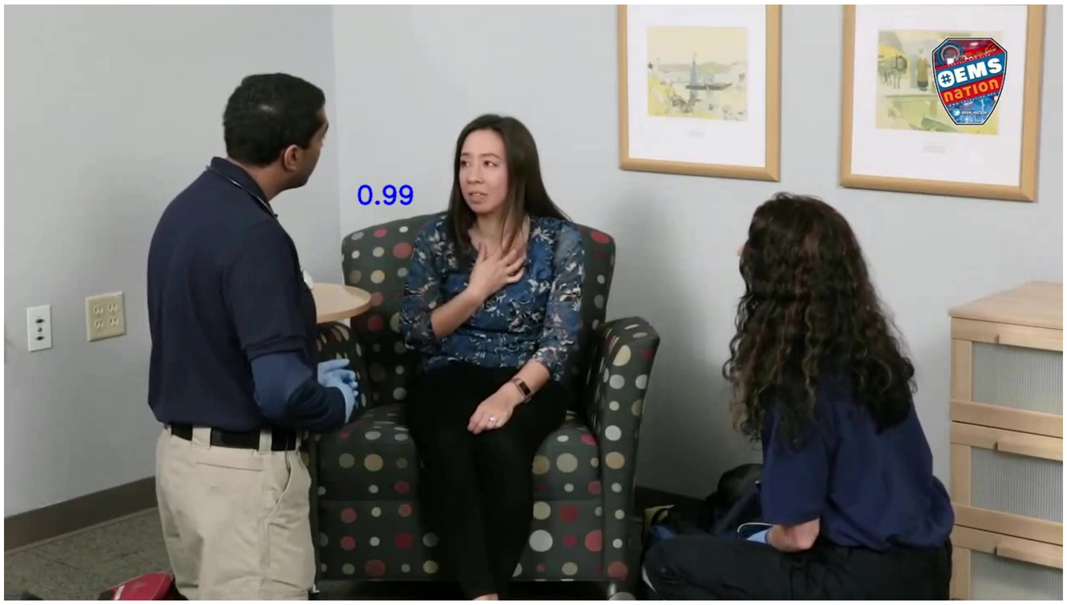

Figure 4.10: One frame extracted from one result YouTube video of the RGB-based classification frame detector[6]

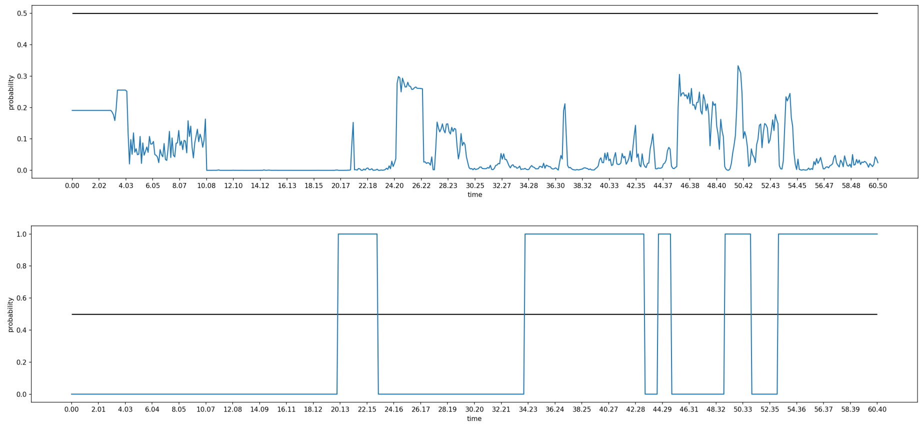

Figure 4.11: The comparison between the resulting plot(top) and the ground truth plot labeled by humans(bottom) of one YouTube video

of the detector is 370s when setting fps to be 10.

[6]Image from YouTube video `https://www.youtube.com/watch?v=K5HKRuKc5F8`

#### 4.2.2.3 *RGB-based localization frame detector*

Figure 4.12 shows two frames extracted from different result videos of the RGB-based localization frame detector. Multiple-person detection is also supported by the detector. In the left figure, the person having chest pain in the middle is detected, and two people from the sides who face back to the camera are not selected, which shows that the detector is less likely to assign a positive label to the person with missing critical body parts. In the right figure, the detector still correctly detects the chest pain action even when the left hand and lower body parts of the person are obscured, which proves that the detector also correctly detects a person with partial body parts. Moreover, the bounding box is more accurate compared to the previous Openpose approach.

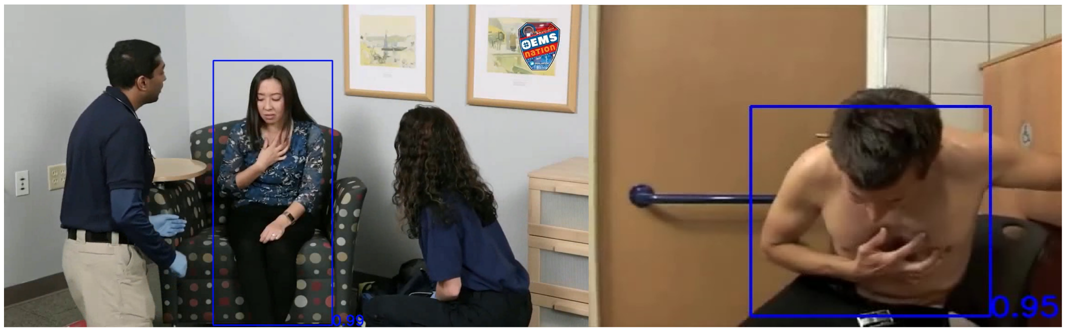

Figure 4.12: Two frames extracted from different result videos of the RGB-based localization frame detector[7]

Figure 4.13 shows the comparison between the resulting plot and the ground truth plot labeled by humans of one YouTube video. From it, we can see that the detector does not cover all the chest pain action instances, but has a much higher precision compared to the skeleton-based frame detector, which makes sense since the RGB-based localization frame model has the highest precision over the 5 models. As a result, the RGB-based localization frame detector seems to be much better on the YouTube videos than the skeleton-based frame detector.

[7]Left image from YouTube video `https://www.youtube.com/watch?v=K5HKRuKc5F8`. Right image from YouTube video `https://www.youtube.com/watch?v=SEoHphkFgPs`

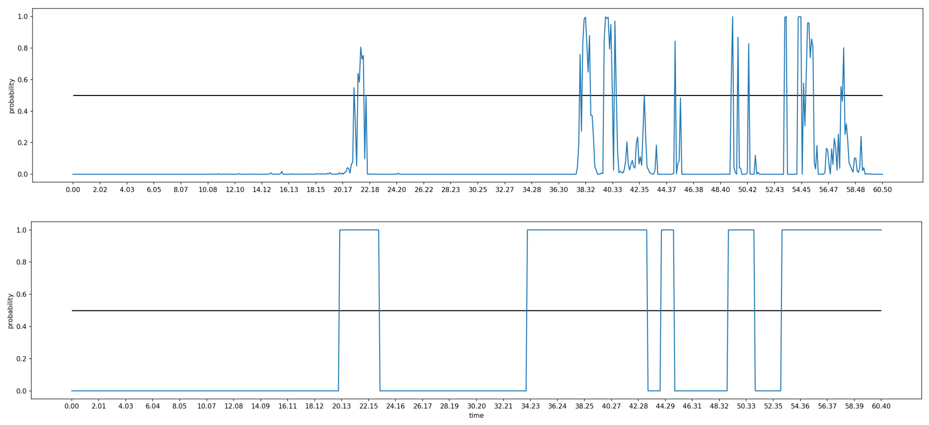

Figure 4.13: The comparison between the resulting plot(top) and the ground truth plot labeled by humans(bottom) of one YouTube video

The runtime of the detector mainly depends on the bounding boxes extraction speed of the pre-trained faster RCNN which is kind of slow since it does not support batching. For a 174s long youtube video with 1280x720 resolution, the runtime of the detector is 2189s when setting fps to be 10.

#### *4.2.2.4 Skeleton-based clip detector*

For the skeleton-based clip detector, we only tested the detector on YouTube videos that contain a single person, since we did not implement a multiple-person tracking algorithm for it. Figure 4.14 shows two frames extracted from a result video of the skeleton-based clip detector. The score is computed by a sequence of frames while the bounding box still depends on the keypoint information of a single frame. From the left figure, we can see that the detector can still detect chest pain correctly with missing body parts. In the right figure, the person is running facing back to the camera. The frame is classified correctly by the detector and is classified incorrectly as chest pain by the skeleton-based frame detector, which shows that the clip detector is better to exclude some actions that depend on multiple frames, such as running.

Figure 4.15 shows the comparison between the resulting plot and the ground truth plot labeled by humans of one YouTube video. From it, we can see that the detector is more stable than the

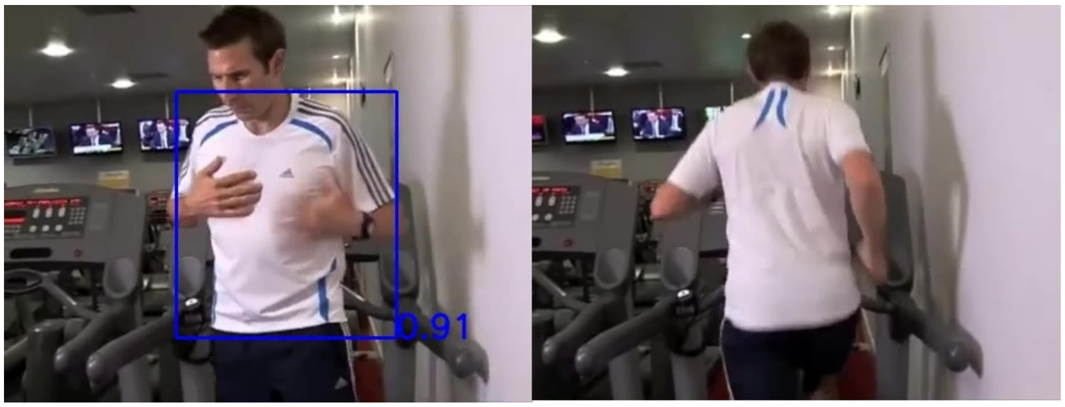

Figure 4.14: Two frames extracted from a result video of the skeleton-based clip detector[8]

skeleton-based frame detector, and has a precision that is between the RGB-based frame detector and skeleton-based frame detector. Because the detector looks at one 3-s clip at a time, it usually overlooks the brief chest pain action instances. Besides that, the detection delay caused by the sliding window approach is not noticeable most of the time.

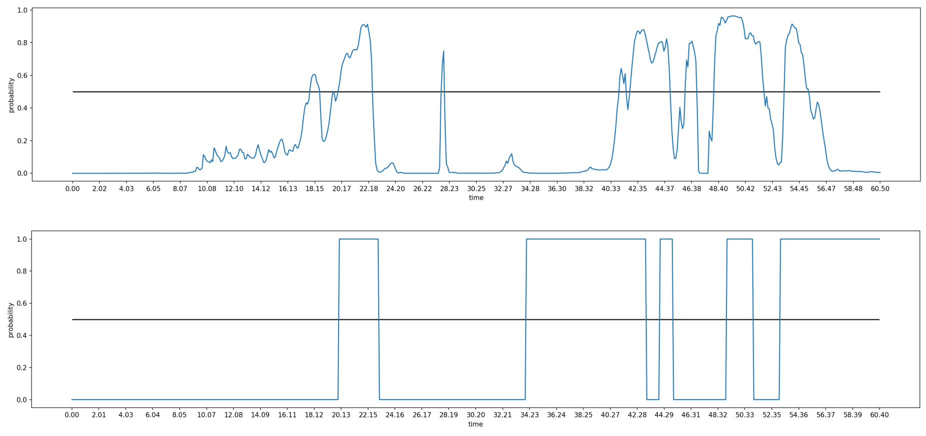

Figure 4.15: The comparison between the resulting plot(top) and the ground truth plot labeled by humans(bottom) of one YouTube video

The runtime of the detector is almost the same as the skeleton-based clip detector as the RNN infers very quickly. For a 174s long youtube video with 1280x720 resolution, the runtime of the

[8]Images from YouTube video `https://www.youtube.com/watch?v=SEoHphkFgPs`

detector is 178s when setting fps to be 10.

#### *4.2.2.5 RGB-based clip detector*

Because the base model I3D is a video classification model, like the RGB-based classification frame detector we have no way to know the location of the people and which person is having chest pain with the RGB-based clip detector. All we can get at one timestamp is just a confidence score, as shown in figure 4.16.

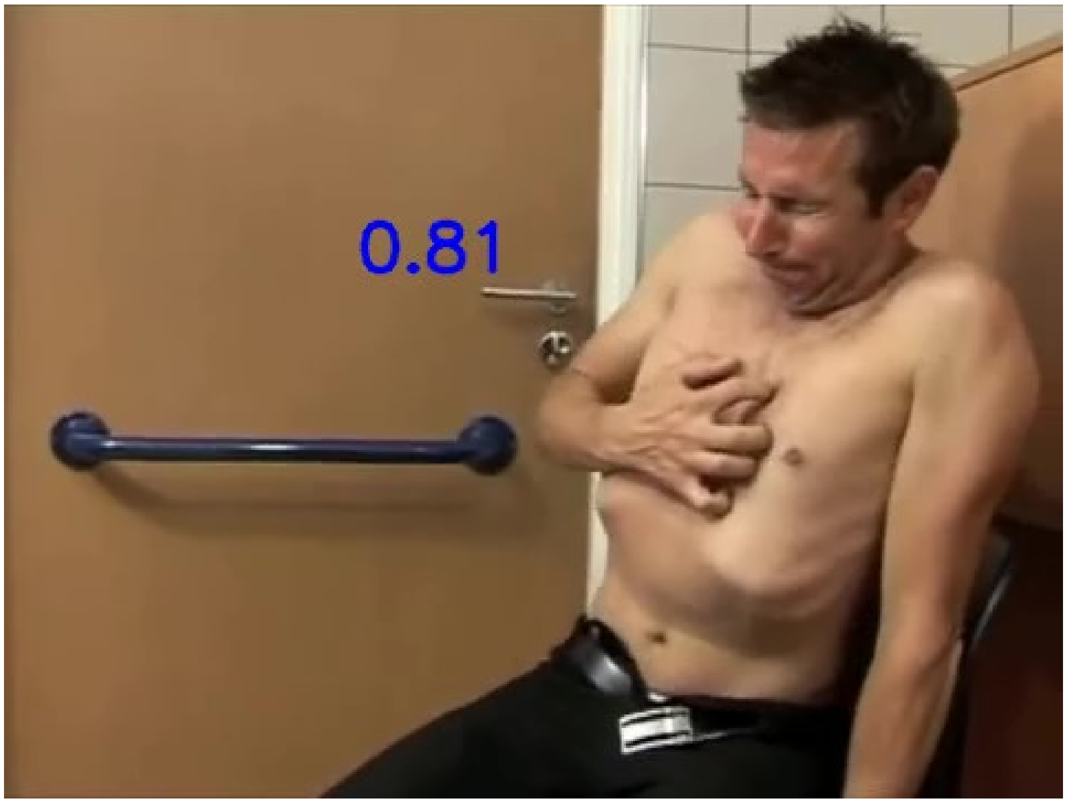

Figure 4.16: One frame extracted from one result YouTube video of the RGB-based clip detector[9]

Figure 4.17 shows the comparison between the resulting plot and the ground truth plot labeled by humans of one YouTube video. From it, we can see that the detector has an extremely low recall, and although it has high precision in this YouTube video, its precision is not high at all in

[9]Image from YouTube video `https://www.youtube.com/watch?v=SEoHphkFgPs`

other YouTube videos. Its temporal footprint is 1s, which is a lot bigger than the temporal footprint of other detectors, which is 0.1s. Moreover, for each 3-s video clip, the last element of the resulting confidence scores list is usually unreasonably low.

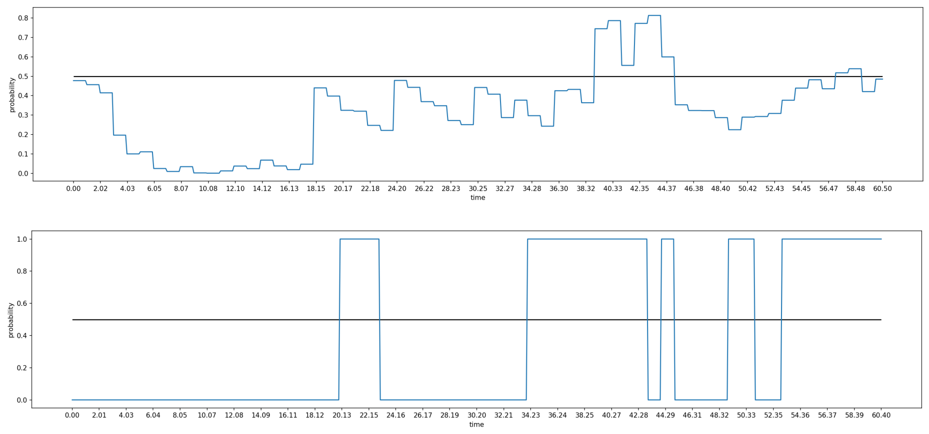

Figure 4.17: The comparison between the resulting plot(top) and the ground truth plot labeled by humans(bottom) of one YouTube video

The runtime of the detector is the quickest since it only does the classification without the localization, and it slices video clips without overlap. For a 174s long youtube video with 1280x720 resolution, the runtime of the detector is 106s.

### *4.2.2.6 Summary*

Table 4.4 summarizes the performance of the detectors on YouTube videos.

| Detector Name | Multi-person | Localization | Precision | Recall | Runtime |
|---|---|---|---|---|---|
| Skeleton-based frame detector | true | true | low | high | fast |
| RGB-based classification frame detector | true | false | high | low | fast |
| RGB-based localization frame detector | true | true | high | moderate | slow |
| Skeleton-based clip detector | false | true | high | moderate | fast |
| RGB-based clip detector | true | false | moderate | low | fast |

Table 4.4: A summary of the performance of the detectors on YouTube videos

# 5. SUMMARY AND CONCLUSIONS

The topic of this research is to train neural network models with various architectures using existing labeled datasets and detect chest pain actions in YouTube videos. Chest pain detection is very important for smart home applications because chest pain could be caused by severe illnesses, and the medical condition of patients could get worse very quickly. However, chest pain detection in YouTube videos is challenging as there are many dissimilarities between YouTube videos and the training set.

We implemented 5 different detectors in total: the skeleton-based frame detector, the RGB-based classification frame detector, the RGB-based localization frame detector, the skeleton-based clip detector, and the RGB-based clip detector. The skeleton-based detectors obtain skeleton data using Openpose and calculate the angle and distance information as input features. The RGB-based detectors use the preprocessed RGB pixel values array as the input feature.

The skeleton-based frame detector uses a feed-forward NN as the classifier, while the skeleton-based clip detector explores temporal patterns with RNN architecture. The RGB-based classification frame detector classifies chest pain with the Inception Resnet architecture, while the RGB-based localization frame detector both locates and classifies the person having chest pain based on the faster RCNN architecture. The RGB-based clip detector utilizes the fine-tuned I3D to analyze the Spatio-temporal features in the video clip.

After training, the RGB-based frame faster RCNN shows an impressive 98.4% precision on the test set of the frame dataset, and the RGB-based clip I3D achieves 100% recall on the test set of the clip dataset. On the dataset, although trained on the clip-wise labeled dataset, the three frame detectors show promising results by precisely classify or locate the chest pain action in most of the video clips. The two clip detectors cannot locate the chest pain action in the clips as the clips in the dataset are too short.

On the YouTube videos, the skeleton-based frame detector suffers from low precision. The RGB-based classification frame detector has strict conditions in order to work properly. The RGB-based localization frame detector and skeleton-based clip detector demonstrate satisfying performance with high precision and moderate recall. The runtime of the RGB-based frame detector is limited by the faster RCNN, which cannot process frames in batch. The RGB-based clip detector has the fastest runtime while suffering from the overall low performance and needs further improvements.

Currently, the performance of the RGB-based clip detector is very limited. It has a long temporal footprint and low precision and recall. The I3D is a video classification model, and we fine-tuned it on a video classification task, therefore it is reasonable that it cannot detect an action well. The temporal action localization technique could be a good solution to this problem. Similar to the faster RCNN which locates and classifies objects in a 2D space, temporal action localization locates and classifies actions in a 1D space which is the time.

To train a temporal action localization model, we need a dataset with timestamp annotation, which records the start time and end time of every action instance. Because our current dataset, NTU RGB+D is labeled clip-wise, we have to obtain the timestamp annotation by ourselves. Besides manually annotate each video clip, we can also utilize the RGB-based frame detector that we have implemented.

However, there are some problems with this approach. To obtain precise annotation, we will have to drop some hard clips that the RGB-based frame detector fails to detect, which makes the small dataset even smaller. Even after dropping some hard clips, the accuracy of the annotation is still limited by the RGB-based frame detector. The chest pain video clips in the NTU RGB+D do not have the ending process of the action, which prevents the model from learning the ending process of chest pain. Moreover, the part before the chest pain action of the clips is usually so short that we cannot generate enough if any negative proposals with a normal proposal length. Due to the above reasons, we should switch to a temporal action localization dataset with chest pain action included, which is hard to find.

## REFERENCES

[1] A. Shahroudy, J. Liu, T.-T. Ng, and G. Wang, "Ntu rgb+d: A large scale dataset for 3d human activity analysis," in *IEEE Conference on Computer Vision and Pattern Recognition*, June 2016.

[2] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh, "Openpose: Realtime multi-person 2d pose estimation using part affinity fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2019.

[3] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," 2016.

[4] J. Carreira and A. Zisserman, "Quo vadis, action recognition? A new model and the kinetics dataset," *CoRR*, vol. abs/1705.07750, 2017.

[5] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F. Li, "Imagenet large scale visual recognition challenge," *CoRR*, vol. abs/1409.0575, 2014.

[6] A. Krizhevsky, "Learning multiple layers of features from tiny images," 2009.

[7] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014.

[8] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *CoRR*, vol. abs/1512.00567, 2015.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

[10] C. Szegedy, S. Ioffe, and V. Vanhoucke, "Inception-v4, inception-resnet and the impact of residual connections on learning," *CoRR*, vol. abs/1602.07261, 2016.

[11] M. Tan and Q. V. Le, “Efficientnet: Rethinking model scaling for convolutional neural networks,” *CoRR*, vol. abs/1905.11946, 2019.

[12] G. Rojas-Albarracín, M. Chaves, A. Fernández-Caballero, and M. T. López, “Heart attack detection in colour images using convolutional neural networks,” *Applied Sciences*, vol. 9, no. 23, 2019.

[13] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, “The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results.” http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[14] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, “Microsoft coco: Common objects in context,” 2015.

[15] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” *CoRR*, vol. abs/1311.2524, 2013.

[16] R. Girshick, “Fast r-cnn,” 2015.

[17] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” 2016.

[18] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “Ssd: Single shot multibox detector,” *Lecture Notes in Computer Science*, p. 21–37, 2016.

[19] H. Mohan, S. Anitha, R. Chai, and S. Ling, “Edge artificial intelligence: Real-time non-invasive technique for vital signs of myocardial infarction recognition using jetson nano,” *Advances in Human-Computer Interaction*, vol. 2021, pp. 1–19, 08 2021.

[20] F. Han, B. Reily, W. A. Hoff, and H. Zhang, “Space-time representation of people based on 3d skeletal data: A review,” *CoRR*, vol. abs/1601.01006, 2016.

[21] Q. Xu, G. Huang, M. Yu, and Y. Guo, “Fall prediction based on key points of human bones,” *Physica A: Statistical Mechanics and its Applications*, vol. 540, p. 123205, 2020.

[22] Y. Zhang, Y. Tian, P. Wu, and D. Chen, “Application of skeleton data and long short-term memory in action recognition of children with autism spectrum disorder,” *Sensors*, vol. 21, no. 2, 2021.

[23] Y. Du, W. Wang, and L. Wang, “Hierarchical recurrent neural network for skeleton based action recognition,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1110–1118, 2015.

[24] W. Zhu, C. Lan, J. Xing, W. Zeng, Y. Li, L. Shen, and X. Xie, “Co-occurrence feature learning for skeleton based action recognition using regularized deep lstm networks,” 2016.

[25] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, “Hmdb: A large video database for human motion recognition,” in *2011 International Conference on Computer Vision*, pp. 2556–2563, 2011.

[26] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, M. Suleyman, and A. Zisserman, “The kinetics human action video dataset,” *CoRR*, vol. abs/1705.06950, 2017.

[27] K. Soomro, A. R. Zamir, and M. Shah, “UCF101: A dataset of 101 human actions classes from videos in the wild,” *CoRR*, vol. abs/1212.0402, 2012.

[28] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, “Long-term recurrent convolutional networks for visual recognition and description,” *CoRR*, vol. abs/1411.4389, 2014.

[29] J. Y.-H. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, “Beyond short snippets: Deep networks for video classification,” 2015.

[30] S. Ji, W. Xu, M. Yang, and K. Yu, “3d convolutional neural networks for human action recognition,” *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 35, pp. 221–231, jan 2013.

[31] G. W. Taylor, R. Fergus, Y. LeCun, and C. Bregler, “Convolutional learning of spatio-temporal features,” in *Computer Vision – ECCV 2010* (K. Daniilidis, P. Maragos, and N. Paragios, eds.), (Berlin, Heidelberg), pp. 140–153, Springer Berlin Heidelberg, 2010.

[32] D. Tran, L. D. Bourdev, R. Fergus, L. Torresani, and M. Paluri, “C3D: generic features for video analysis,” *CoRR*, vol. abs/1412.0767, 2014.

[33] C. Zach, T. Pock, and H. Bischof, “A duality based approach for realtime tv-l<sup>1</sup> optical flow,” in *Proceedings of the 29th DAGM Conference on Pattern Recognition*, (Berlin, Heidelberg), p. 214–223, Springer-Verlag, 2007.

[34] Y.-G. Jiang, J. Liu, A. Roshan Zamir, G. Toderici, I. Laptev, M. Shah, and R. Sukthankar, “THUMOS challenge: Action recognition with a large number of classes.” `http://crcv.ucf.edu/THUMOS14/`, 2014.

[35] F. C. Heilbron, V. Escorcia, B. Ghanem, and J. C. Niebles, “Activitynet: A large-scale video benchmark for human activity understanding,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 961–970, 2015.

[36] X. Dai, B. Singh, G. Zhang, L. S. Davis, and Y. Q. Chen, “Temporal context network for activity localization in videos,” 2017.

[37] J. Gao, Z. Yang, and R. Nevatia, “Cascaded boundary regression for temporal action detection,” 2017.

[38] H. Xu, A. Das, and K. Saenko, “R-c3d: Region convolutional 3d network for temporal activity detection,” 2017.

[39] Y.-W. Chao, S. Vijayanarasimhan, B. Seybold, D. A. Ross, J. Deng, and R. Sukthankar, “Rethinking the faster r-cnn architecture for temporal action localization,” 2018.

[40] H. Pham, L. Khoudour, A. Crouzil, P. Zegers, and S. A. Velastin, “Skeletal movement to color map: A novel representation for 3d action recognition with inception residual networks,” *CoRR*, vol. abs/1807.07033, 2018.

[41] R. Zhao and P. van der Smagt, “Two-stream rnn/cnn for action recognition in 3d videos,” pp. 4260–4267, 09 2017.