Heart Attack Detection

Kartik Venkataraman

February 12, 2021

Abstract

The research topic I'm working on is Heart Attack Detection. I initially designed a model which was a combination of two models - one for detecting Hand-on-Chest and the other for detecting Facial Expressions. I used this model to label YouTube videos and to identify "moments" in YouTube videos. I was able to get FPR around 15-20% and FNR about 3-5% depending on the videos. To reduce the FPR even further, I used a novel approach which uses Snorkel technique which is a "weak supervision" technique. I wrote a couple of heuristic functions as Snorkel labeling functions and used them alongwith the pretrained model to detect labels for Heart Attack. Using this technique, I was able to get significantly better results than the baseline pretrained model. Now, the FPR is around 8-10% and FNR is about 3-5%.

1 Topic

The research topic I'm working on is Heart Attack Detection. This project presents a proposal to identify people with an apparent heart attack by detecting characteristic postures of heart attack. The method of identifying infarcts (in this case, possible heart attack) makes use of convolutional neural networks. I have built a CNN model and trained it on a GPU-enabled server (Google Colab) to recognize heart-attack moments from a video. I have trained the model with a set of images from the internet and also self-made videos that contain people simulating a heart attack.

2 Motivation

Heart Attacks are one of the leading cause of death throughout the world, according to World Health Organization (WHO) [2]. Heart problems appear as a person gets older. Moreover, the average age is increasing worldwide according to the World Bank [1]. In some countries like Italy, Greece, and Japan, the population older than 65 years exceeds 20% of the total, and the percentage tends to increase. Moreover, many older people worldwide live independently at home rather than move to a nursing centre. However, the decision to live alone increases the likelihood of not receiving timely assistance during an emergency, and even more so when a person lives in remote locations.

For this reason, many researchers have been developing methods and mechanisms to automatically detect abnormal events over the last decades [9]. A heart attack is an example of a situation in which timely care makes the difference between life and death. When a person experiences a heart attack, one main symptom is a strong pain in the chest [5]. A person living alone will find it very difficult to ask for help because of the pain caused by the infarct. Therefore, it is necessary to provide mechanisms that automatically detect events that affect a person's health as in the case of heart attacks.

The strong pain in the chest due to a heart attack, present in most cases, leads to a position in which the person brings their hands to the chest, falling down or upper body moving forward and pain or fear expression on the face. This posture could be useful to detect a heart attack using computer vision techniques. However, as far as we know, there is no project based on human postures and gestures capable of detecting a possible infarction by processing a single image.

So, I would be working on proposing a method for detecting heart attack using images and frames from a video. This is called a non-invasive method, i. e., it does not require a person to wear any device to directly monitor the patient or the person to operate a system in some way. This project is a real smart-home application which will be very helpful to detect a probable heart attack in a timely manner.

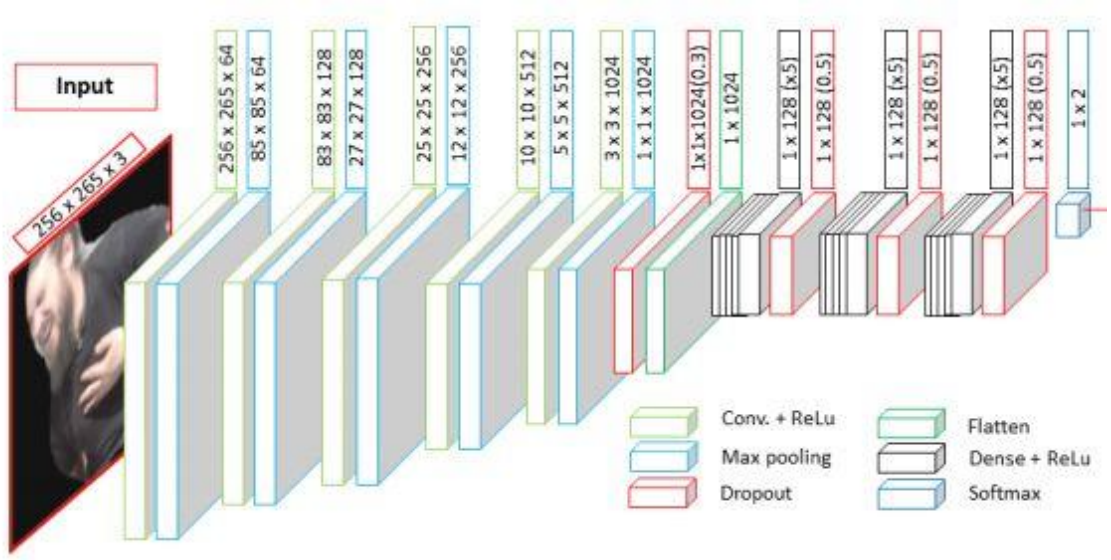

Figure 1: Neural Network Architecture

3 Related Works

A lot of research is going on to solve various challenges in health care through human activity recognition [6]. This has opened the door for the detection of abnormal events autonomously. Fall detection is by far the most commonly faced challenge [3] [18]. But there are other challenges like monitoring Parkinson's disease [8], Alzheimer's or even recognising emotional states [14].

It is possible to classify all the research work into two principal approaches according to the acquisition sensors of the input data, non-visual and visual. For non-visual sensors, there is an extensive use of accelerometers and gyroscopes [17] [11]. Although these devices provide high precision, their main drawback is that they require that people wear them all the time, which is not always possible.

For visual approach, cameras are used and it is always based on computer vision to analyse the captured images or videos. The use of the Microsoft Kinect device is a solution often found in the literature [19] [15]. This simplifies some tasks like background subtraction and skeleton generation. But the accuracy of Kinect to capture decreases after a few metres of distance. Another approach is analysing the image information directly by using artificial intelligence techniques. In fact, the use of neural networks has obtained excellent classification results in identifying particular events in recent years [13] [21]. The most traditional approach to identify specific human events by analysing images is the identification of the human posture [16]. This can be achieved by obtaining the skeleton [22] [20] or processing the complete information of the person's image, which is the approach I'm using.

4 Proposed Model

My model is a combination of two existing models - Hands on Chest from color images and Facial Expression Recognition, which are described below.

4.1 Hands on Chest

4.1.1 Architecture

The model that I started working with is shown in Figure 1.

At the beginning of the network, the architecture has five convolutional blocks, where each block is firstly composed of a convolution layer to highlight the general features in the image. Then, a max pooling layer is provided to keep the number of variables of the network low, in this way maintaining a size easy to compute.

```
NET = Sequential()
NET.add(Convolution2D(64, kernel_size=(3 ,3), padding ="same", input_shape=(256, 256, 3), activation='relu'))
NET.add(MaxPooling2D((3,3), strides=(3,3)))
NET.add(Convolution2D(128, kernel_size=(3, 3), activation='relu'))
NET.add(MaxPooling2D((3,3), strides=(3,3)))
NET.add(Convolution2D(256, kernel_size=(3, 3), activation='relu'))
NET.add(MaxPooling2D((2,2), strides=(2,2)))
NET.add(Convolution2D(512, kernel_size=(3, 3), activation='relu'))
NET.add(MaxPooling2D((2,2), strides=(2,2)))
NET.add(Convolution2D(1024, kernel_size=(3, 3), activation='relu'))
NET.add(MaxPooling2D((2,2), strides=(2,2)))
NET.add(Dropout(0.3))
NET.add(Flatten())

for _ in range(5):
    NET.add(Dense(128, activation='relu'))
NET.add(Dropout(0.5))

for _ in range(5):
    NET.add(Dense(128, activation='relu'))
NET.add(Dropout(0.5))

for _ in range(5):
    NET.add(Dense(128, activation='relu'))
NET.add(Dropout(0.5))

NET.add(Dense(CLASSES, activation='softmax'))

sgd = SGD(lr=LR, decay=1e-4, momentum=0.9, nesterov=True)

NET.compile(optimizer=sgd,
              loss='binary_crossentropy',
              metrics=['acc', 'mse'])
```

Figure 2: Code snippet for neural network model

Class	Initial Training	Training Augmented	Initial Validation	Validation Augmented	Initial Testing	Testing Augmented	Final
Infarct	532	10,640	114	2280	114	2280	15,960
No Infarct	532	10,640	114	2280	114	2280	15,960
Total	1064	21,280	228	4,560	228	4560	31,920

Figure 3: Data distribution and total images after augmentation

In the middle of the network, just after the convolution blocks, there is a dropout layer that prevents the generated model from presenting an envelope training, mainly due to the limited amount of data. After this, a flatten layer allows changing the 2D design of the convolutional layers to a vectorial one so that the values generated in the previous layers are passed to the traditional neuron layers.

At the end of the network, ten layers composed of traditional neurons are arranged, each with 128 neurons, which deliver the result of forward propagation to a softmax function with two outputs. These will classify whether there is or is not a person with a heart attack in the image.

The code snippet for constructing the model is shown in Figure 2.

A total of 31,920 images were obtained by adding the augmented images to the original pictures, as shown in Figure 3. It is important to mention that each image generated by the transformations during the data augmentation process was kept in the same set to which the original image belonged, ensuring that both the original image and its transformations belonged to no more that one set.

4.1.2 Input

I chose the 70%–15%–15% for training, validation and testing, which is a typical configuration in many other applications based on neural networks. The number of images and the distribution is given in Figure 4. Once the training images were selected, the model was built using the Tensorflow framework. A precision of 99% was achieved during training, which allowed 91.75% accuracy and 92.85% sensitivity in the test set defining a learning rate of 0.003 and using the gradient descent

Class	Training 70%	Validation 15%	Test 15%	Total
Infarct	532	114	114	760
No Infarct	532	114	114	760
Total	1064	228	228	1520

Figure 4: Images per class and set.

optimiser.

4.1.3 Output

The output shape of each of the hidden layers can be seen in Figure 5.

4.2 Facial Expression Recognition

In the above model implemented, we can see that I haven't taken facial expressions into account. So, it will classify a video of a person laughing with hands on or near the chest as a positive case of a heart attack. This is a case of false positive. So, I worked on detecting facial expressions and then combined the testing of the facial expression recognition model with the existing model.

4.2.1 Architecture

At the beginning of the network, the architecture has three convolutional blocks, where each block is firstly composed of a convolution layer to highlight the general features in the image. Then, an average pooling layer is provided to keep the number of variables of the network low, in this way maintaining a size easy to compute.

In the middle of the network, just after the convolution blocks, there is a dropout layer that prevents the generated model from presenting an envelope training, mainly due to the limited amount of data. After this, a flatten layer allows changing the 2D design of the convolutional layers to a vectorial one so that the values generated in the previous layers are passed to the traditional neuron layers.

At the end of the network, two layers composed of traditional neurons are arranged, each with 1024 neurons, which deliver the result of forward propagation to a softmax function with seven outputs. These will classify the emotion based on the facial expression. The loss is cross-entropy as it is a multi-class classification.

A reference model can be seen in Figure 6. It is not the exact model used. The code snippet for the model is shown in Figure 7

4.2.2 Input

The training set consists of 28,709 examples. The public test set used for the leaderboard consists of 3,589 examples. The final test set, which was used to determine the winner of the competition, consists of another 3,589 examples. So, the distribution chosen here is the 80%–20% for training and testing.

4.2.3 Output

The output for each of the hidden layers and the final output shape is described in

5 Dataset

My model is a combination of two existing models - Hands on Chest from color images and Facial Expression Recognition. Different features needs to be extracted for both the models and so, I had t use two separate datasets, which are explained below.

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 256, 256, 64)	1792
max_pooling2d_1 (MaxPooling2	(None, 85, 85, 64)	0
conv2d_2 (Conv2D)	(None, 83, 83, 128)	73856
max_pooling2d_2 (MaxPooling2	(None, 27, 27, 128)	0
conv2d_3 (Conv2D)	(None, 25, 25, 256)	295168
max_pooling2d_3 (MaxPooling2	(None, 12, 12, 256)	0
conv2d_4 (Conv2D)	(None, 10, 10, 512)	1180160
max_pooling2d_4 (MaxPooling2	(None, 5, 5, 512)	0
conv2d_5 (Conv2D)	(None, 3, 3, 1024)	4719616
max_pooling2d_5 (MaxPooling2	(None, 1, 1, 1024)	0
dropout_1 (Dropout)	(None, 1, 1, 1024)	0
flatten_1 (Flatten)	(None, 1024)	0
dense_1 (Dense)	(None, 128)	131200
dense_2 (Dense)	(None, 128)	16512
dense_3 (Dense)	(None, 128)	16512
dense_4 (Dense)	(None, 128)	16512
dense_5 (Dense)	(None, 128)	16512
dropout_2 (Dropout)	(None, 128)	0
dense_6 (Dense)	(None, 128)	16512
dense_7 (Dense)	(None, 128)	16512
dense_8 (Dense)	(None, 128)	16512
dense_9 (Dense)	(None, 128)	16512
dense_10 (Dense)	(None, 128)	16512
dropout_3 (Dropout)	(None, 128)	0
dense_11 (Dense)	(None, 128)	16512
dense_12 (Dense)	(None, 128)	16512
dense_13 (Dense)	(None, 128)	16512
dense_14 (Dense)	(None, 128)	16512
dense_15 (Dense)	(None, 128)	16512
dropout_4 (Dropout)	(None, 128)	0
dense_16 (Dense)	(None, 2)	258

Total params: 6,633,218
Trainable params: 6,633,218
Non-trainable params: 0

Figure 5: Output shapes of tensors after each layer.

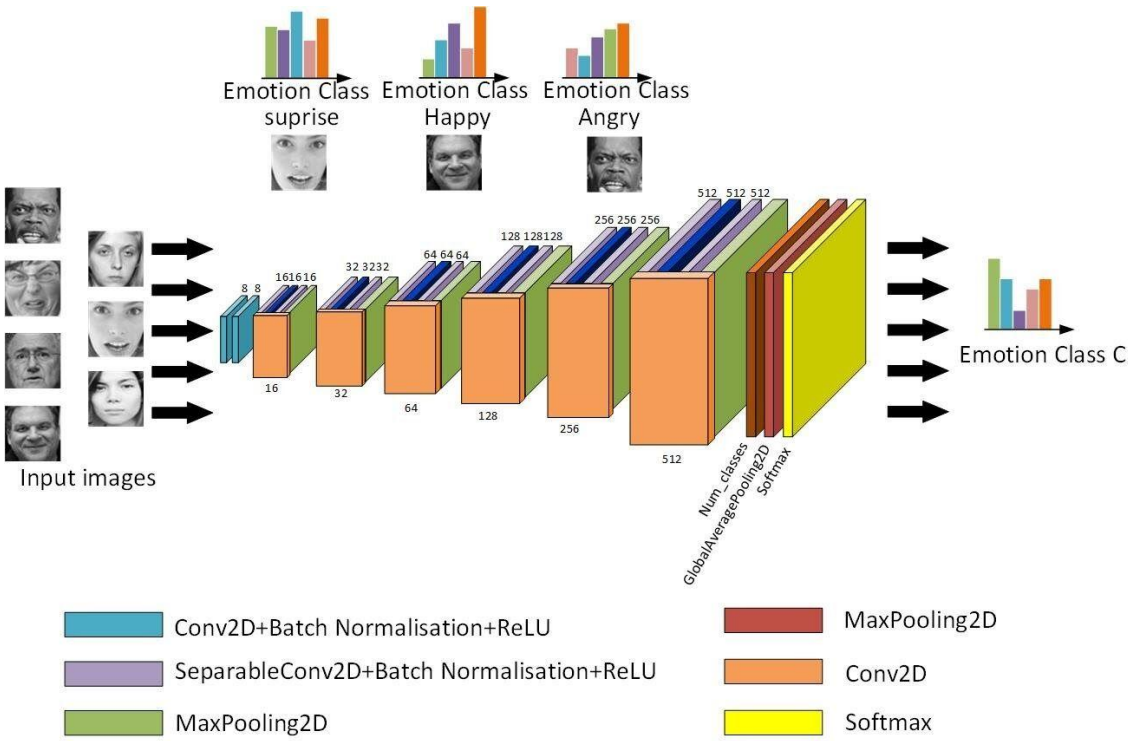

Figure 6: Neural Network Architecture for face expression recognition

```
model = Sequential()
model.add(Conv2D(64, (5, 5), activation='relu', input_shape=(48,48,1)))
model.add(MaxPooling2D(pool_size=(5,5), strides=(2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(AveragePooling2D(pool_size=(3,3), strides=(2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(AveragePooling2D(pool_size=(3,3), strides=(2, 2)))
model.add(Flatten())
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
model.compile(loss='categorical_crossentropy'
    , optimizer=keras.optimizers.Adam()
    , metrics=['accuracy'])
```

Figure 7: Code snippet of the Neural Network model for face expression recognition

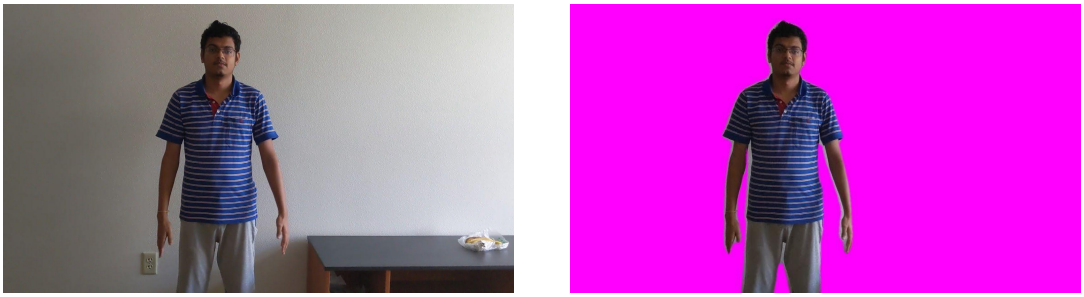

Figure 8: Instance segmentation and background removal

Figure 9: Data Augmentation

5.1 Hands on Chest

The dataset is made up of images created by myself and also downloaded from the internet. I created this image dataset on the basis of infarct and non-infarcts [4]. In images labelled as an infarct, all images show a person's posture in which they have one or both hands on the chest. As regards to no infarct situations, images where people are performing daily activities were used. The initial image data set consisted of a total of 1520 images, 760 images of class "Infarct" and 760 images of class "No Infarct". For better learning and extracting of features, the number of images in the data set is increased using data augmentation [10].

The following steps were performed:

- Each image was scaled to a maximum size of 256×256 pixels, maintaining the original proportion, both for "Infarct" and "No Infarct" images. This is in order to reduce the amount of data to be processed during the augmented data technique.
- After that, the images were classified into two categories, "Infarct" and "No Infarct". Furthermore, each category was split into the three subcategories of training, validation, and testing, as shown in Figure 3.
- As the CNNs only have to infer a possible heart attack, people were extracted from the background of the image by reducing the noise caused by the variation of the background in order to improve the training set. People were automatically located in each image. For this purpose, I performed the instance segmentation and background removal using Mask R-CNN [12] and replaced the pixels with magenta color so as to be a contrast to the person. This is shown in Figure 8.
- Data augmentation is a process for generating new samples by transforming training data. In this case, each original picture generated 20 more different images. For this, six transformations were combined and applied to each image (rotation, increase/decrease in width or height, zoom, horizontal flip, and brightness change). This is shown in Figure 9.

5.2 Facial Expression Recognition

Obviously, I couldn't use the same dataset for facial expression recognition that I created previously as this dataset needs to be more focused on the facial pixels. And there was not much time to

Hyper-parameters	Batch Size	Epochs	Steps/epoch training	Steps/epoch validation	Dropout	Learning rate
Experimentation	32, 48, 64	5, 10, 20	100, 200, 500	20, 50, 100	0.3, 0.5	0.001, 0.003
Optimal	48	10	500	100	0.5	0.003

Figure 10: Range of Hyperparameters tried and optimal values.

```
======= ======== ========
Found 4821 images belonging to 2 classes.
loss: 0.3153, acc: 0.9175, mse: 0.0764
======= ======== ========
```

Figure 11: Test Performance

create a new dataset. So, I used an existing dataset. It is known as the FER-2013 dataset.

- The data consists of 48x48 pixel grayscale images of faces. The faces have been automatically registered so that the face is more or less centered and occupies about the same amount of space in each image. The emotion shown in the facial expression is in one of seven categories (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral).
- train.csv contains two columns, "emotion" and "pixels". The "emotion" column contains a numeric code ranging from 0 to 6, inclusive, for the emotion that is present in the image. The "pixels" column contains a string surrounded in quotes for each image. The contents of this string are space-separated pixel values in row major order. test.csv contains only the "pixels" column and your task is to predict the emotion column.
- The training set consists of 28,709 examples. The public test set used for the leaderboard consists of 3,589 examples. The final test set, which was used to determine the winner of the competition, consists of another 3,589 examples.

6 Model Training and Performance

I trained both the models separately - Hands on Chest from color images and Facial Expression Recognition. As they could not be trained together as each had its own network and datasets. The performance of each model can be seen below -

6.1 Hands on Chest

6.1.1 Hyperparameters

In this project, there are many hyperparameters such as batch size, epochs, steps in training and validation, dropout and learning rate. I trained with different values of hyperparameters to see which gives the best results. The experimentation is documented in Figure 10.

6.1.2 Testing Performance

I also varied the number of epochs which you can see in the graphs I plotted of the training accuracy and training loss versus the number of steps and shown in Figure 12 and Figure 13.

6.2 Facial Expression Recognition

6.2.1 Hyperparameters

For this model, hyperparameters I experimented are: batch size, epochs and dropout. The experimentation can be seen in Figure 14

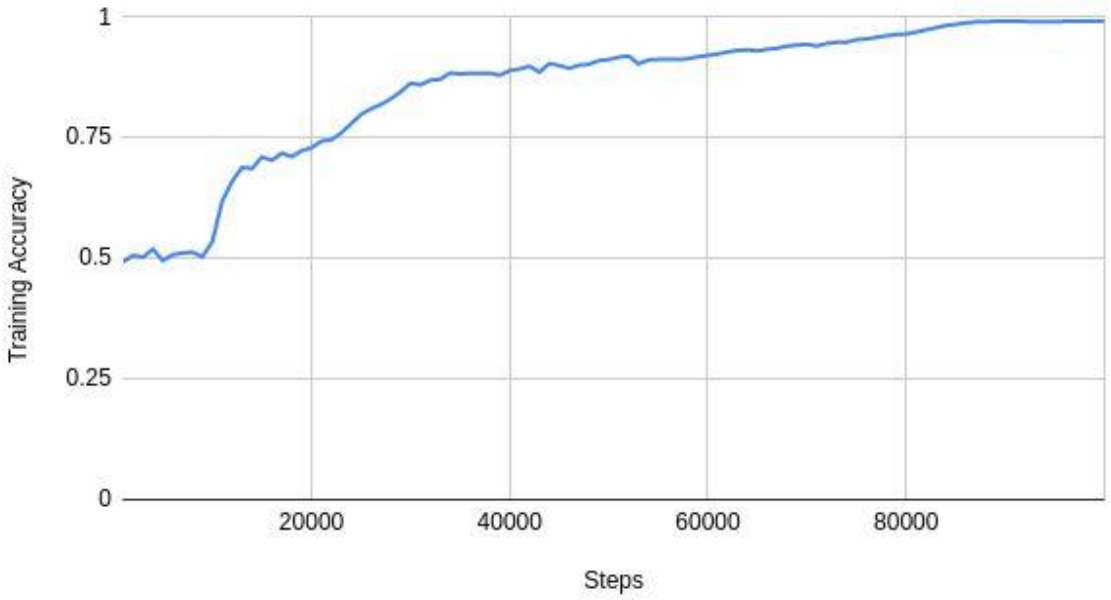

Figure 12: Training Accuracy vs. Steps

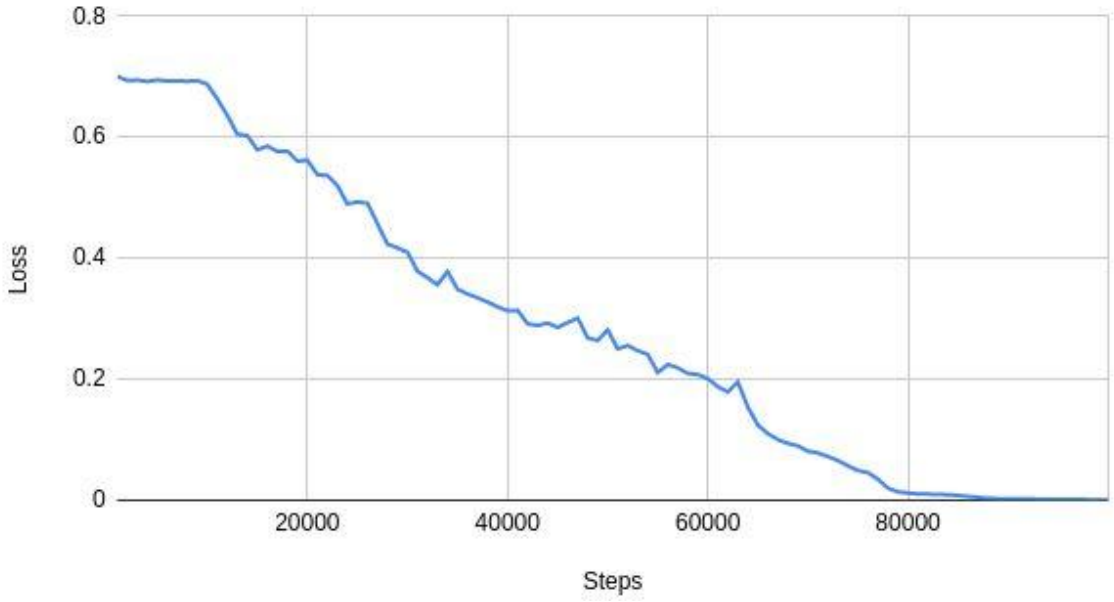

Figure 13: Training Loss vs. Steps.

Hyperparameters	Batch Size	Epochs	Dropout
Experimentation	128, 256	2, 3, 5	0.1, 0.2
Optimal	256	5	0.2

Figure 14: Range of Hyperparameters tried and optimal values

```
Train loss: 0.223031098232
Train accuracy: 92.0512731201
==============================
Test loss: 2.27945706329
Test accuracy: 57.4254667071
```

Figure 15: Training and Testing performance

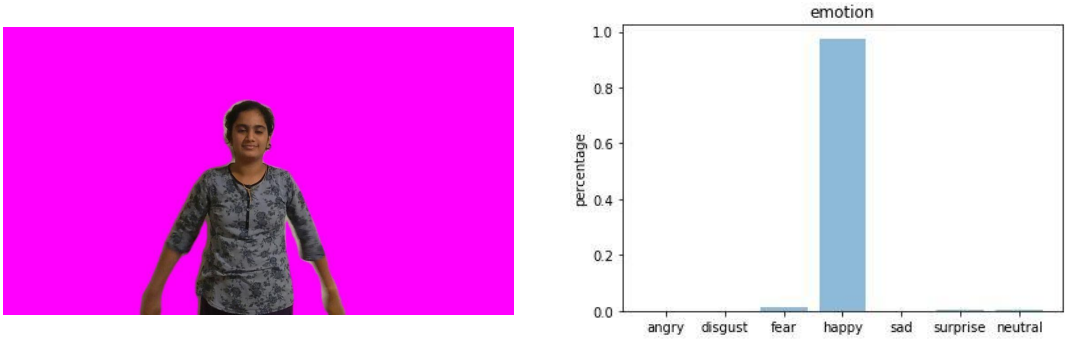

Figure 16: Facial Expression Model output visualization

6.2.2 Testing

The training and testing performance can be seen in Figure 15

For our custom images, emotions are stored as numeric labels from 0 to 6. Keras would produce an output array including these 7 different emotion scores. We can visualize each prediction as a bar chart as seen in Figure 16.

But sometimes, the model cannot clearly identify which emotion the person is experiencing as it can't be exactly identified from the expression. This can be seen in Figure 17

As there are seven classes, the accuracy does not express the right impression for multi class classification problems as some emotions have very little change in the facial expressions as we see above. So, I tried to calculate the confusion matrix to get a better understanding. This is shown in Figure 18

The rows represent actual labels whereas columns state predictions. That means that there are 467 angry instances in testset. We can classify 214 angry items correctly. On the other hand, we classified 9 items as disgust but these items are actual angry ones. So, as we can see, the model is not very accurate but facial expression detection is still a difficult and open problem.

For our project of heart attack detection, I decided to take into account all the emotions except "happy" and "surprise". So, when the person holds the chest initially, if his emotion is happy or surprised, it won't label it as heart-attack. After a few frames with hand-on-chest detection, even when there is no hand on the chest, the system will classify as heart-attack if the emotion is sad, fear, angry, disgust and even neutral.

This is a very important addition to the project as the person might not have his hands on the chest always but might be having a heart attack. That's where facial expression detection plays an important role.

7 Performance on Youtube Videos

I tested the model on a few videos. The videos can be found here. The model ran quite well and found moments recognising Heart Attack. The figures can be seen here.

7.1 How to detect "moments" of target action/emotion

My model runs frame-by-frame. So, initially, for each video, I found the frames where my model predicted Heart Attack and saw the graph. Those graphs had spikes in some points, especially when hands go from in front of the chest and not holding the chest or facial expression model giving wrong prediction for a few frames.

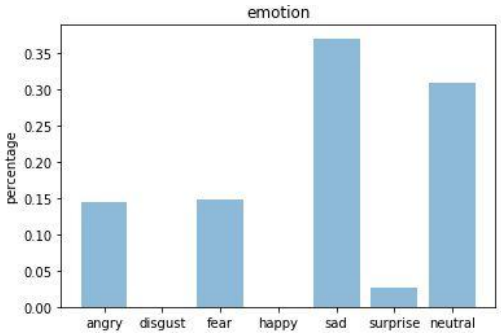

Figure 17: Facial Expression Model output ambiguity

	Angry	Disgust	Fear	Happy	Sad	Surprise	Neutral
Angry	214	9	53	30	67	8	86
Disgust	10	24	9	2	6	0	5
Fear	45	2	208	29	89	45	78
Happy	24	0	40	696	37	18	80
Sad	65	3	83	56	285	10	151
Surprise	7	1	42	27	9	303	26
Neutral	45	2	68	65	88	8	331

Figure 18: Confusion Matrix of Facial Expression Recognition Model

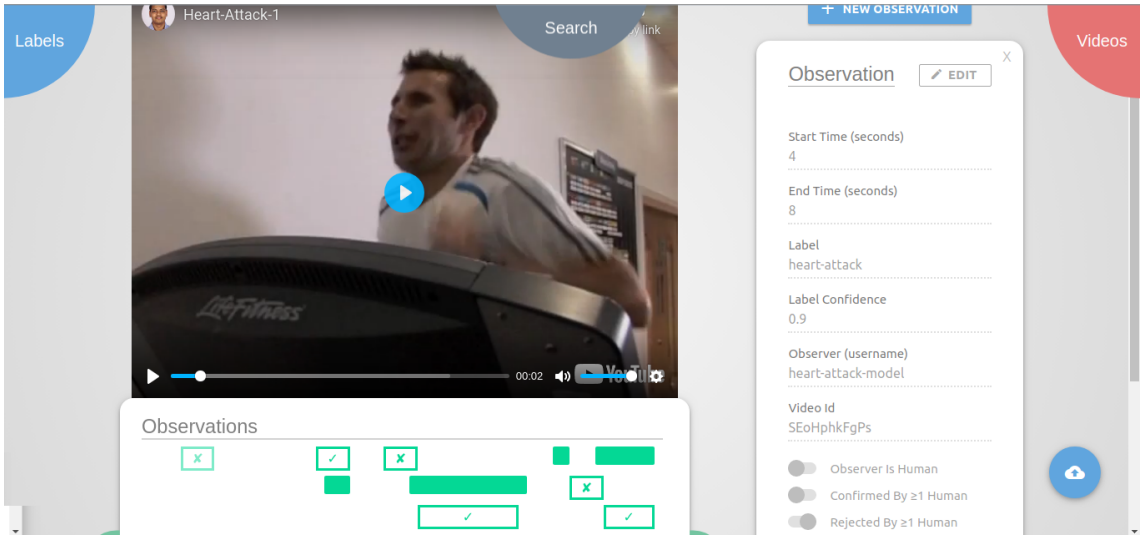

Figure 19

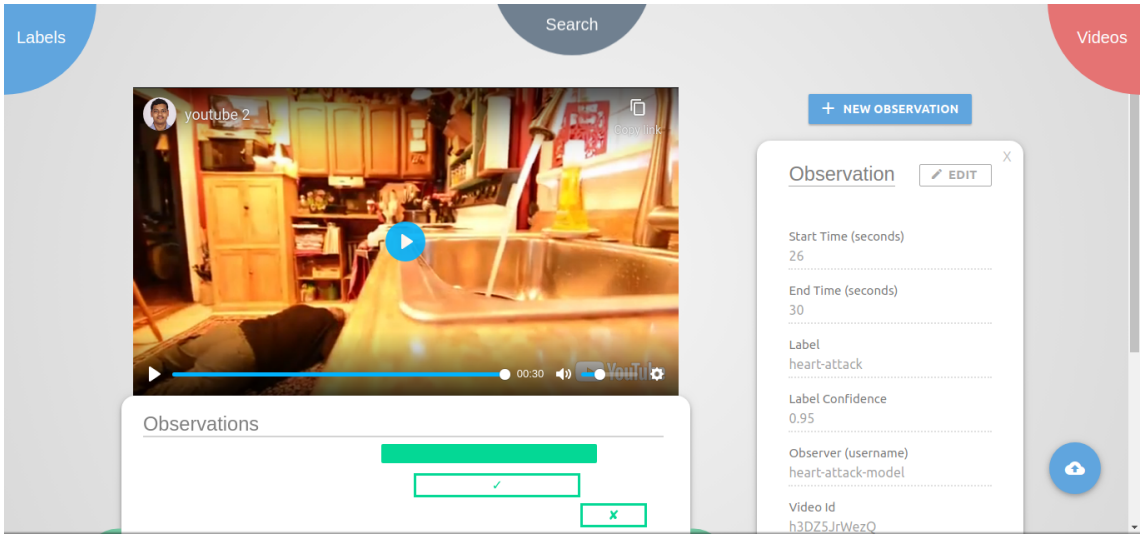

Figure 20

So, to avoid these spikes and find the actual moments, I then decided to split the videos into small sub-clips of 3-4 seconds ($\tilde{1}$00 frames/sub-clip). The frame-rate of YouTube videos is around 25-30 fps. So, I tried to group by 100 frames together, i.e. around 3-4 sec clip of the video and give a label for that sub-clip. I also had an overlap of 50 frames so that I can more accurately determine the exact moments.

So, for 100 frames, if 30 frames return a probability of heart attack as more than 0.5, that sub-clip is labeled as positive and recognised as a positive moment of Heart Attack.

7.2 View Found "Moments" in iLab Website

As I mentioned above, initially I was generating json for each frame and identifying label for each frame. But then I created sub-clips and generated labels for each sub-clip. The label I identified was whether Heart Attack is present in that sub-clip or not. I created the json file in the appropriate format and uploaded the results for the videos I obtained from YouTube. You can see the results in Figures 19, 20, 21, 22. The results are satisfactory.

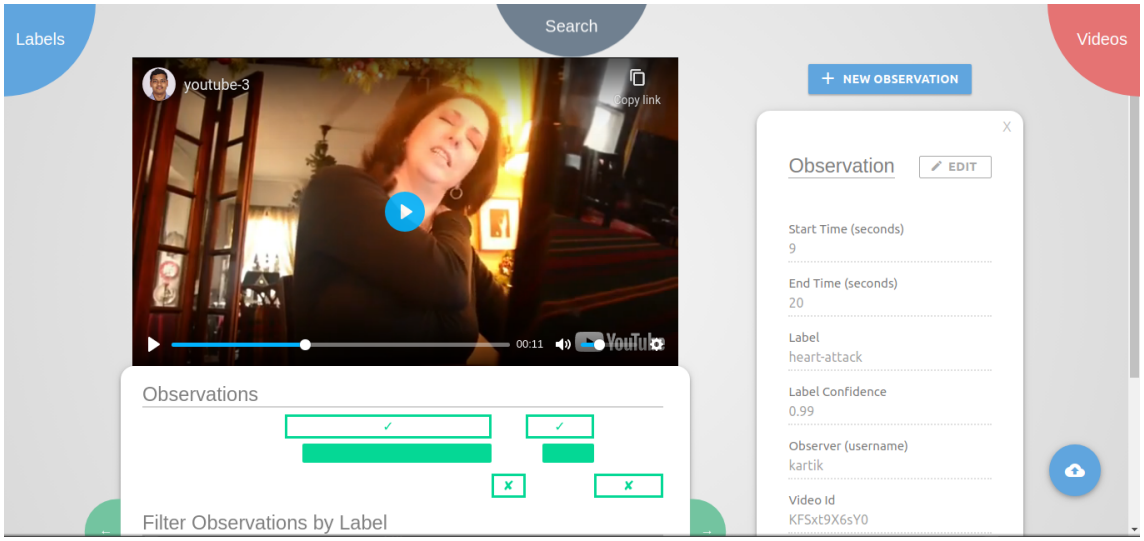

Figure 21

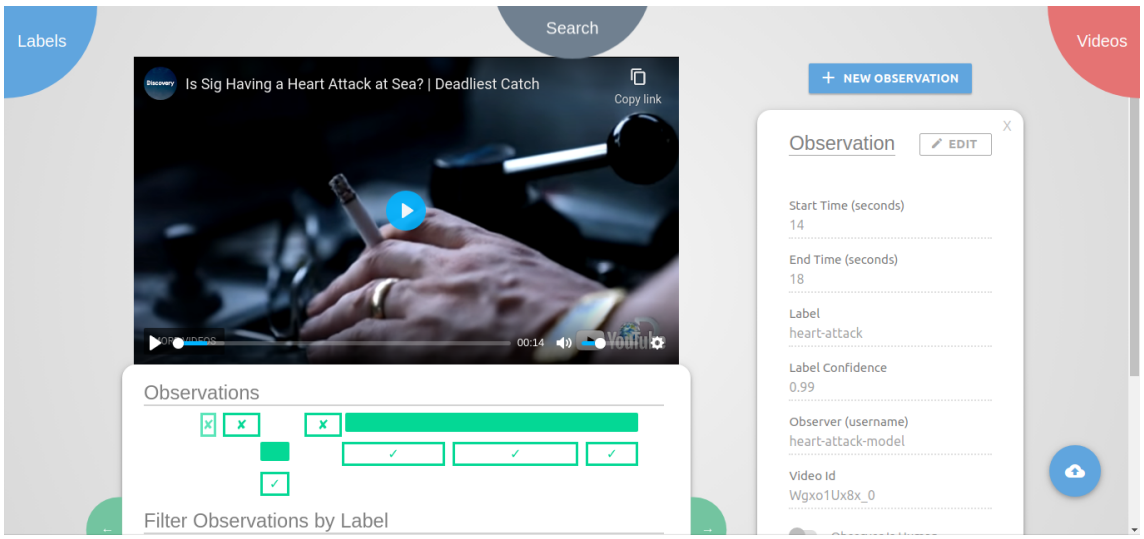

Figure 22

YouTubeID	Accuracy	FPR	FNR
SEoHphkFgPs	76.67%	20%	3.33%
h3DZ5JrWezQ	77.78%	11.11%	11.11%
KFSxt9X6sY0	72.23%	27.77%	0%
Wgxo1Ux8x_0	82.85%	17.85%	0%

Figure 23: Result of model on YouTube videos

7.3 Performance: Accuracy

Let's measure the accuracy of the model when it is applied to Youtube videos. We can measure it in two ways: False-Positive Rate (FPR), and False-Negative Rate).

- For defining False-Positive Rate (FPR), I used the true labels (manual labeling) for the video and compared with the labels I got from the model for each sub-clip in the video.

  To understand it better, let's take an example. Let's say a video has Heart-Attack from 10-20 seconds, and I have sub-clips of 4 seconds each with 2 seconds overlap, then my target labels (defined manually) will be -
  Target [[6-10, 0], [8-12, 1], [10-14, 1], [12-16, 1], [14-18, 1], [16-20, 1], [18-22, 1], [20-24, 0]]

  Now, let's say my model based on the above method of giving labels to sub-clips rather than each frame gives the following labels -
  Pred [[6-10, 1], [8-12, 1], [10-14, 1], [12-16, 1], [14-18, 1], [16-20, 1], [18-22, 1], [20-24, 1]]

  So, False Positive Rate is the ratio where the model predicts as 1 but the actual label is 0. So, in above case - 25%. As for 2 sub-clips out of 8, the model predicts 1 when actual label is 0.

- For determining False Negative Rate, I used a similar logic.

  Now, let's say my model based on the above method of giving labels to sub-clips rather than each frame gives the following labels -
  Pred [[6-10, 1], [8-12, 1], [10-14, 0], [12-16, 1], [14-18, 1], [16-20, 1], [18-22, 1], [20-24, 1]]

  So, False Negative Rate is the ratio where the model predicts as 0 but the actual label is 1. So, in above case - 12.5%. As for 1 sub-clips out of 8, the model predicts 1 when actual label is 0.

Based on the above mentioned definition of FPR and FNR, the results for the four YouTube videos uploaded to iLab website are shown in Figure 23

8 Improve Accuracy by using Snorkel (Novel Approach)

When we apply a trained model to practical videos (represented by Youtube videos), the accuracy of the model goes a little down. So, there is a possibility to increase the accuracy.

To improve the accuracy of your model, beside improving its architecture or training method, we can also provide more labeled dataset. But it is be time consuming to label images/videos.

The approach I used is to use the Snorkel project, *https* : *//www.snorkel.org*, which is a "weak supervision" technique. It allows to label data via efficient programming, instead of laborious manual labeling.

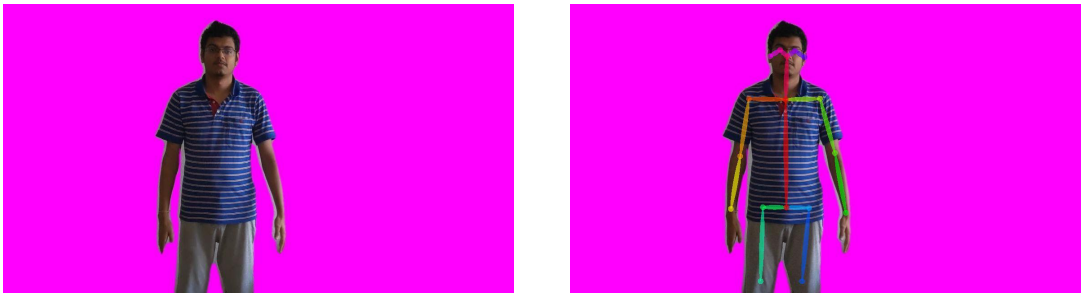

Figure 24: OpenPose body landmarks generation


```
{"version":1.3,"people":[{"person_id":[-1],"pose_keypoints_2d":
[0,0,0,286.293,158.43,0.780511,351.587,151.862,0.702645,411.577,255.048,0.816693,348.959,273.287,0.333809,217.168,162.315,0
.710143,172.756,275.905,0.856764,183.268,360.714,0.819845,292.876,372.486,0.38953,333.341,371.167,0.348826,0,0,0,0,0,0,251.
078,372.48,0.351259,0,0,0,0,0,0,0,0,0,0,0,0,316.355,87.9865,0.868645,252.405,94.4511,0.805801,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0],"face_keypoints_2d":[],"hand_left_keypoints_2d":[],"hand_right_keypoints_2d":[],"pose_keypoints_3d":
[],"face_keypoints_3d":[],"hand_left_keypoints_3d":[],"hand_right_keypoints_3d":[]}]}
```


Figure 25: Body landmarks json file

But instead of using it to increase the dataset size, I used it as a kinf of "Ensemble technique" by defining a few more heuristic functions based on body landmark keypoints to label the sub-clips and predict based on the combination of heuristic functions and my trained model.

Let's see how it works.

8.1 OpenPose

Openpose [7] is a tool for generating facial/body landmarks for humans in video. It is a realtime multi-person 2D pose estimation is a key component in enabling machines to have an understanding of people in images and videos. The method uses a nonparametric representation, referred to as Part Affinity Fields (PAFs), to learn to associate body parts with individuals in the image. This bottom-up system achieves high accuracy and real time performance.

I installed OpenPose in Google Colab and used the BODY 25 model as it is best suited for this project and gives the best results.

Figures 25 shows the json file of the body landmarks that are generated for each frame. To understand how this json file is created and how to interpret this, please see here

The strong pain in the chest due to a heart attack, present in most cases, leads to a position in which the person brings their hands to the chest, falling down or upper body moving forward and pain or fear expression on the face. This posture could be detected using the body landmarks.

When the person brings their hands near to the chest, we can identify that action by extracting the following features using body landmarks -

- The distance between the wrist and neck keypoints.
- The distance between the wrist and hip keypoints.
- The angle created by the arm i.e. the shoulder-elbow-wrist angle.

As a human being, we can in general say the following things when the hands are near to the chest -

- The distance between the wrist and neck keypoints will be less than or equal to distance between the wrist and hip keypoints.
- The distance between the wrist and neck will be less than or equal to half of the distance between the neck and hip keypoints.
- The angle created by the arm i.e. the shoulder-elbow-wrist angle will be an acute angle. It can't be a right angle or obtuse angle.

The above generalizations will have a lower threshold as well as the angle cannot be too acute as it will then go above the chest and same for the distance between the wrist and neck keypoints.

Thresholds for heuristics in Snorkel	Angles(min, max) - count threshold above which it gives the label as Heart Attack for each subclip	Arm ratio (wrist-to-neck/neck-to-hip) (min, max) - count threshold above which it gives the label as Heart Attack for each subclip
Ranges tested on my own video and professor videos	(50, 90) - 30, (50, 90) - 40 (40, 80) - 30, (40, 80) - 40 (40, 90) - 30, (40, 90) - 40	(0.4, 0.9) - 30, (0.4, 0.9) - 40 (0.4. 0.8) - 30, (0.4, 0.8) - 40 (0.3, 0.8) - 30, (0.3, 0.8) - 40 (0.3, 0.7) - 30, (0.3, 0.7) - 40
Optimal	(50, 90) - 30	(0.3, 0.7) - 40

Figure 26: Thresholds for Hueristic functions

8.2 Snorkel Labeling Functions

Labeling functions can be used to represent many heuristic and/or noisy strategies for labeling data, often referred to as weak supervision. The basic idea of labeling functions, and other programmatic operators in Snorkel, is to let users inject domain information into machine learning models in higher level, higher bandwidth ways than manually labeling thousands or millions of individual data points. The key idea is that labeling functions do not need to be perfectly accurate, and can in fact even be correlated with each other. Snorkel will automatically estimate their accuracies and correlations in a provably consistent way, and then reweight and combine their output labels, leading to high-quality training labels.

So, based on the generalizations that a human can make based on the posture, we can write heuristic functions to label the video frames.

- The labeling function for angles will determine if in the video clip that is given as input, if the angle created by the arm is in a specific range, then it will label as Heart Attack.
- The labeling function for the ratio of distance between the wrist and neck and distance between the neck and hip keypoints. If this ratio is in a specific range, then it will label as Heart Attack.
- The labeling function for the ratio of distance between the wrist and neck and distance between the wrist and hip keypoints. If this ratio is in a specific range, then it will label as Heart Attack.
- The labeling function for my model is the labels my pre-trained model predicted for the corresponding sub-clips.

The input to the labeling functions are pandas DataFrames which contains all the features extracted from the keypoints for each video sub-clip. So, the labeling function gives a label for each video sub-clip just like the pretrained model.

8.3 Thresholds for Heuristic Functions

As discussed above, we need to give ranges or thresholds to ratios and angles for the heuristic functions. This is kind of hyperparameter fine-tuning which we need to keep checking based on the training videos. I tested the thresholds on various videos before concluding on the final values. This is shown in Figure 26.

8.4 Combining the Labels and giving the prediction

The next step is to apply the labeling functions we wrote to the unlabeled training data. The result is a label matrix, L_train, where each row corresponds to a data point and each column corresponds to a labeling function. Since the labeling functions have unknown accuracies and correlations, their output labels may overlap and conflict. We use the "LabelModel" to automatically estimate their accuracies and correlations, reweight and combine their labels, and produce our final set of clean, integrated labels.

YouTubeID	Accuracy	FPR	FNR
SEoHphkFgPs	90.00%	6.67%	3.33%
h3DZ5JrWezQ	66.67%	0%	33.33%
KFSxt9X6sY0	77.78%	11.11%	11.11%
Wgxo1Ux8x_0	92.86%	7.14%	0%

Figure 27: Result of Snorkel technique on YouTube videos

So, based on our heuristics and pre-trained model, Snorkel combines the labels given by these functions for each sub-clip in a provably consistent way and give the labels for the whole video.

We can compare the results obtained by using the Snorkel technique against the baseline result of our pretrained model in the next section.

8.5 Improve Accuracy of Model on YouTube Videos

Based on the above mentioned technique and the definitions of FPR and FNR in previous section, the results for the four YouTube videos uploaded to iLab website are shown in Figure 27

8.6 Analysis of Results

Based on the results of the four videos, we can clearly see that the Snorkel results are better than the baseline model. Some notable points are as follows -

1. FPR of all the videos has decreased without any exception. This is because in the model, hands in front of chest were also detected as hands-on-chest. But due to the heuristics of angle and arm ratio, they become false.
2. Accuracy of almost all the videos has increased. There is decrease in the accuracy of only one video as in that video, there is a segment which is zoomed on the face and because I used Facial Expression in the model, it got detected correctly. But when only the face is visible, it is impossible to get body landmarks. This technique won't work in those cases.
3. There is no specific trend in FNR. It remains same, decreases or increases but overall, there is an increase in the accuracy.

I created the json file in the appropriate format for the above generated results and uploaded the results for the videos I obtained from YouTube. You can see the results in Figures 28, 29, 30, 31.

9 Code in Github

The code is available at https://github.com/Kartikvenkat98/Heart-Attack-Detection. It is shared with JaxAI.

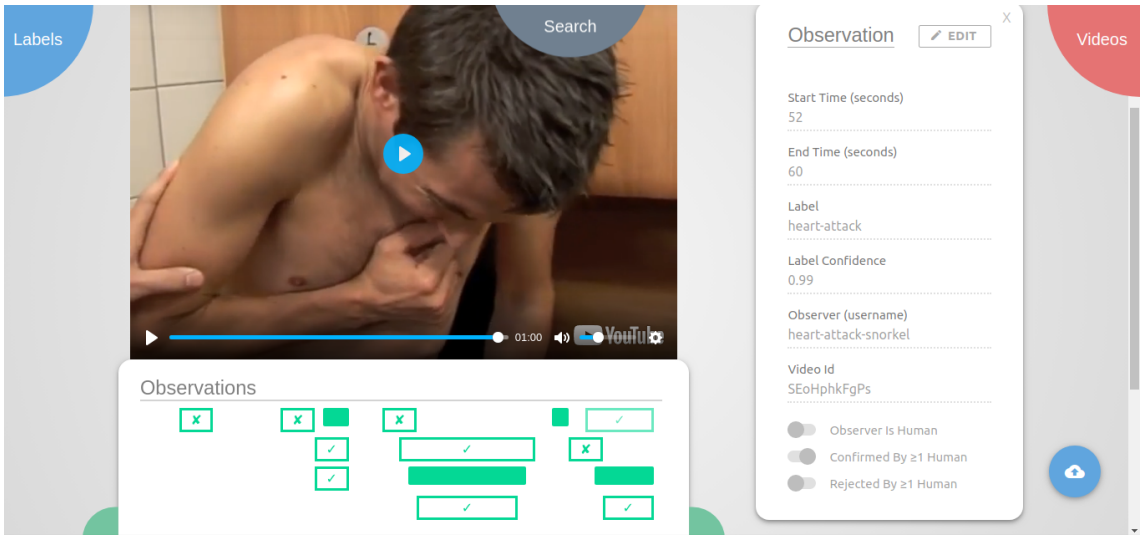

Figure 28

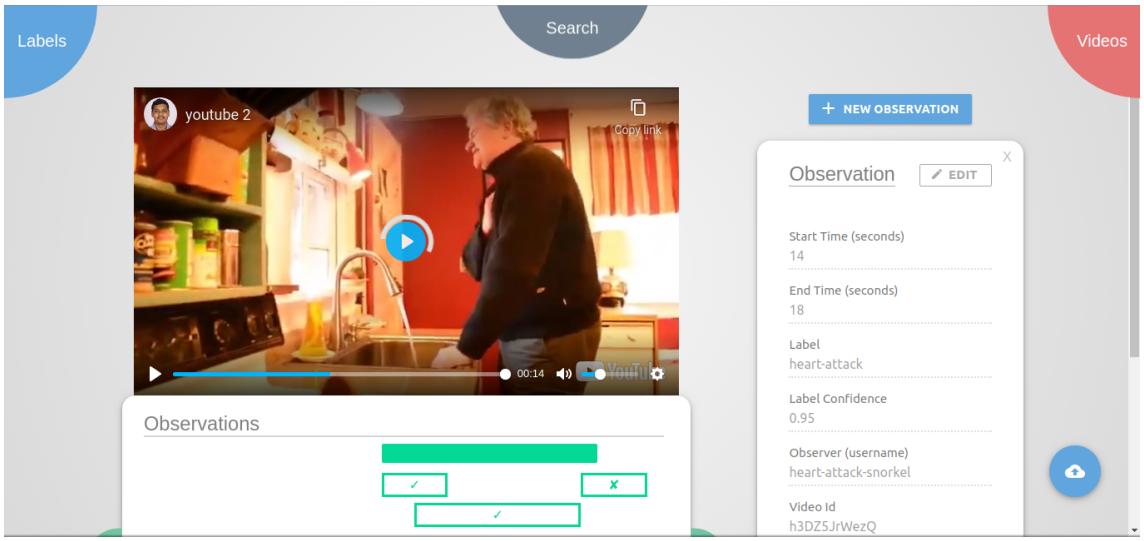

Figure 29

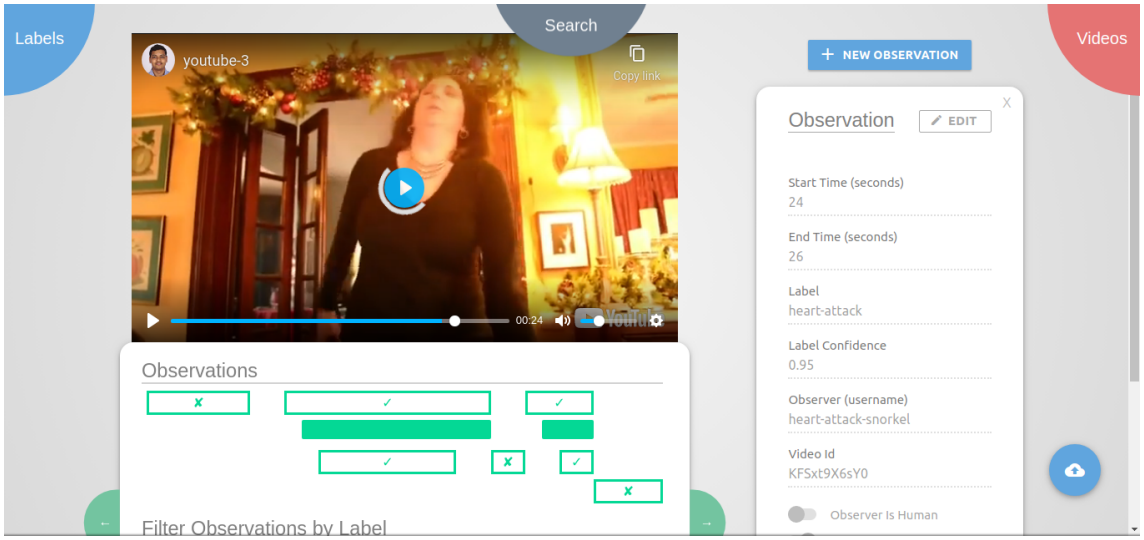

Figure 30

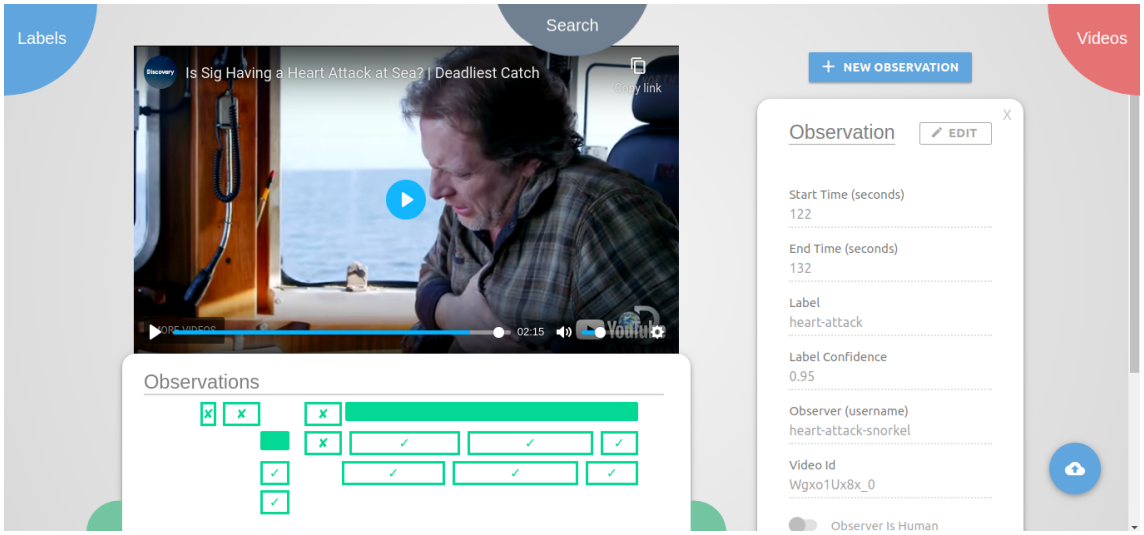

Figure 31

References

[1] The World Bank, Washington, DC, USA, 2017. *Population Ages 65 and above (% of Total).*

[2] World Health Organization (WHO), Geneva, Switzerland, 2018. *The Top 10 Causes of Death.*

[3] R. Alazrai, M. Momani, and M. Daoud. Fall Detection for Elderly from Partially Observed Depth-Map Video Sequences Based on View-Invariant Human Activity Representation, 2017. *Applied Sciences*, Vol.7, 316.

[4] G. R. Albarracin, M. A. Chaves, A. F. Caballero, and M. T. Lopez. Heart Attack Detection in Colour Images using Convolutional Neural Networks, 2019. *Applied Sciences.*

[5] A.Patel, J.Fang, C.Gillespie, E.Odom, C.Luncheon, and C.Ayala. Awareness of heart attack signs and symptoms and calling 9-1-1 among US adults, 2018. In *Journal of the American College of Cardiology*, Vol. 71, Pages 808-809.

[6] S. Ari and S. P. Sahoo. On an algorithm for human action recognition, 2019. *Expert Systems with Applications*, Vol.105, Pages 524-534.

[7] Z. Cao, G. Hidalgo, T. Simon, S. Wei, and Y. Sheikh. OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields, 2019. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition.*

[8] C. W. Cho, W. H. Chao, S. H. Lin, and Y. Y. Chen. A vision-based analysis system for gait recognition in patients with Parkinson's disease, 2009. *Expert Systems with Applications*, Vol.36, Pages 7033-7039.

[9] C. Dhiman and D. K. Vishwakarma. A review of state-of-the-art techniques for abnormal human activity recognition, 2019. In *Engineering Applications of Artificial Intelligence*, Vol. 77, Pages 21-45.

[10] A. Fawzi, H. Samulowitz, D. Turaga, and P. Frossard. Adaptive data augmentation for image classification, 2016. In *Proceedings of the International Conference on Image Processing*, Pages 3688-3692.

[11] Y. Guan and T. Ploetz. Ensembles of deep LSTM learners for activity recognition using wearables, 2017. In *Proceedings of ACM Interact. Mob. Wearable Ubiquitous Technol*, Vol.1, Pages 11:1-11:28.

[12] K. He, G. Gkioxari, P. Dollar, and R. Girshick. Mask r-cnn, 2017. In *Proceedings of the International Conference on Image Processing*, Pages 2980-2988.

[13] W. N. Lie, A. T. Le, and G. H. Lin. Human fall-down event detection based on 2d skeletons and deep learning approach, 2018. In *Proceedings of the International Workshop on Advanced Image Technology*, Pages 1-4.

[14] C. J. Lin, C. H. Lin, S. H. Wang, and C. H. Wu. Multiple Convolutional Neural Networks Fusion using Improved Fuzzy Integral for Facial Emotion Recognition, 2019. *Applied Sciences*, Vol.9, 2593.

[15] H. Y. Lin, Y. L. Hsueh, and W. N. Lie. Abnormal event detection using Microsoft Kinect in a smart home, 2016. In *Proceedings of the International Computer Symposium*, Vol.1, Pages 285-289.

[16] M. M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele. 2D human pose estimation: New benchmark and state of the art analysis, 2014. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Pages 3686-3693.

[17] D. Micucci, M. Mobilio, and P. Napoletano. UniMiB SHAR: A Dataset for Human Activity Recognition using Acceleration data from Smartphones, 2017. *Applied Sciences*, Vol.7, 1101.

[18] M. V. Sokolova, J. Serrano-Cuerda, J. C. Castillo, and A. Fernández-Caballero. A fuzzy model for human fall detection in infrared video, 2013. *Journal of Intelligent and Fuzzy Systems*, Vol.24, Pages 215-228.

[19] E. E. Stone and M. Skubic. Fall detection in homes of older adults using the Microsoft Kinect, 2015. *IEEE Journal of Biomedical and Health Informatics*, Vol.19, Pages 290-301.

[20] J. Sung, C. Ponce, B. Selman, and A. Saxena. Unstructured human activity detection from RGBD images, 2012. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Pages 842-849.

[21] H. Yang, J. Zhang, S. Li, J. Lei, and S. Chen. Attend it Again: Recurrent Attention Convolutional Neural Network for Action Recognition, 2018. *Applied Sciences*, 8, 383.

[22] C. Zhao, M. Chen, J. Zhao, Q. Wang, and Y. Shen. 3D Behavior Recognition based on Multi-Modal Deep Space-Time Learning, 2019. *Applied Sciences*, 9, 716.