FINDING HEADACHE MOMENTS FROM YOUTUBE VIDEOS USING WEAK SUPERVISION

A Thesis

by

JAGADISH KUMARAN JAYAGOPAL

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

MASTER OF SCIENCE

Chair of Committee,	Anxiao (Andrew) Jiang
Committee Members,	Ruihong Huang
	Tie Liu
Head of Department,	Scott Schaefer

August 2021

Major Subject: Computer Science

ABSTRACT

Smart Monitoring of actions of elderly people have become more important these days since they mostly live by themselves. Majority of them have some kind of illness and require monitoring in their daily life. Computer Vision and Deep Learning can play a vital role in monitoring critical actions of the elderly people and the detected information can be very useful for their primary doctors and their kith and kin to care of them. To build a strong deep learning model that can detect 'headache' moments from videos, the biggest bottleneck is huge amount of labeled training data. The reality is that hand labeled datasets are expensive and may take many months or in some cases years to create. The practical deployment of deep learning is hindered by the cost and intractability of hand labeling such datasets. This bottleneck has led to many machine learning systems use some form of weak supervision. The present study aims to use multiple weak supervision sources such as a pretrained deep learning model and several handcrafted heuristic rules; integrate and model them using Snorkel [1] which helps to programmatically build training datasets without manual labeling, from YouTube videos. A False Positive Rate (FPR) of 0.08% and False Negative Rate of 0.01% were achieved using the DNN model.This research study has shown that the accuracy of the combined weak supervision model is superior than the single pretrained model for programmatically building training dataset consisting of headache moments.

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

1.1 Topic

In this project, a 2D skeleton based deep learning model has been trained; hand crafted heuristic rules have been built using 2D key-points extracted using OpenPose from each frame of a video. The trained deep learning model and the heuristic rules are considered as the weak supervision sources. These weak supervision sources are passed on Snorkel to generate a unified weak supervision model that programmatically labels the YouTube clips as "Headache" or "Not Headache".

1.2 Motivation

Headache, a common, disabling neurologic problem is presumably a primary disorder in older adults [5]. In older adults, the prevalence of headache has been reported to range from 12% to 50% [6] [7]. Frequent headache (more than 2 times per month) occurs in up to 17% of people older than age 65 years [8]. Older adults tend to have co-morbid medical conditions and hence early diagnosis and treatment of headache is very critical. Cluster headache is short-lasting and tends to occur during the early morning hours. So, physicians seldom witness an attack. Whenever they have headache, they tend to clutch the affected side of head with their hands [9].The action of touching head falls under the group of alerting situations. In this project, a 2D skeleton based Deep Learning model and multiple heuristic rules will be developed to read videos of people and detect if they have the hand on their head. The objective of this project is to develop a good unified weak supervision model to detect moments of headache in YouTube videos more accurately, efficiently, and collect a large dataset of such "moments". If a pre-trained deep learning model (single weak supervision source) alone is used to extract ‘headache’ moments from YouTube videos, the quality of the dataset is restricted by the accuracy of the deep learning model. In

this study, the trained model along with heuristics are used as multiple weak supervision sources and generate a unified model using Snorkel and show that the integrated labels will have a better quality. The main reason for using YouTube videos to extract headache moments is that the headache videos in YouTube are wilder than videos recorded in a lab environment where subjects enact the action of headache. Most of the videos in YouTube represent real people suffering from headache or migraine. Hence a dataset for headache built using YouTube videos will be a good representation of actual people suffering from some kind of headache.

There are quite a few challenges in this research. The first challenge is that 3D keypoints cannot be used in this research since 3D keypoints cannot be estimated for YouTube videos. It was found that when HCN model [2] was trained and validated on NTU dataset [10], the accuracy dropped by around 10% when 3D keypoints estimated using Microsoft Kinect v2 were replaced by 2D keypoints estimated using OpenPose [3]. The second challenge is the accuracy of 2D landmarks estimated by OpenPose on YouTube videos. OpenPose works well when the whole body is visible in a frame of the video. Many times, it fails to estimate 2D key points when only half of the body is visible in a frame. Also when there is dynamic movements of the person in a video, the accuracy of OpenPose drops. This directly affects the accuracy of the deep learning model as well as the heuristic rules. This in turn will affect the quality of labels obtained from the weak supervision model.

1.3 Related Works

In this section, a brief summary of the research works which have developed deep learning models for detection of hand touch head (headache) have been provided. Some of the research works have used image data as input and a few others have used skeleton data as input to their models for the action detection.

[2] has employed a CNN model to extract features from skeleton data. Joint co-occurrences and temporal evolutions are passed on to the deep learning network as two streams of inputs and are later fused by concatenation along channels after a specific convolution layer. Although the branches of network identical architecture, their parameters are learned in an isolated fashion. Their model's performance on NTU dataset [10] and PKU-MMD dataset [11] have shown to be better than the state of the art methods.

[12] has implemented Graph Convolution Networks on skeleton data. They have proposed to use higher-order spatial and temporal features learnt from skeleton data and a multi-stream feature fusion method to fuse these higher order features. Their model has achieved state-of-the-art performance on NTU dataset [10] and NTU-120 dataset [13]. They have shown that the multi-feature fusion method wins over the single-feature-based method.

[14] has proposed a deep learning based real-time multi-person action recognition system. They have used Inflated 3D ConvNet (I3D) proposed by [15] for action recognition. This network architecture takes only RGB images as input. They have used video segments of a window size of 16 frames from the previous stage as input to I3D. Every video segment produces a recognition class and a corresponding confidence score. Non-maximum suppression (NMS) is used to obtain a robust decision of multiple object detection. They have experimentally proved that their proposed method can perform multiple-person action recognition in real-time viable for smart home monitoring application.

The paper [16] focuses on using user-defined programmatic heuristics to detect complex objects such as cyclists and by extension, situations, based on the output of existing object detection algorithms.

The paper [17] has proposed a rule-based NLP algorithm to automatically generate labels for the training data, and then use the pre-trained word embeddings as deep repre-

sentation features for training machine learning models.

In [18], they have used Snorkel technique to encode domain knowledge as labeling functions and then applied to unlabeled MRI sequences to generate integrated labels for the task of classifying aortic valve malformations. They have experimentally proved for this task that deep learning models trained with hand labeled dataset were outperformed by models trained with labels generated using weak supervision.

1.4 Research Roadmap

Firstly, the Hierarchical Co-occurrence Network (HCN) model [2] will be trained on 946 headache videos as positive samples and 960 videos from other actions as negative samples from NTU-RGBD dataset [10]. The trained deep learning model is first evaluated on 322 headache videos from PKU-MMD dataset [11] and then tested on YouTube videos. Next several heuristic rules are developed to detect 'headache' moments in YouTube videos and integrated along with the pretrained deep learning model (another weak supervision source) and a unified weak supervision model is generated using Snorkel. Snorkel automatically estimates the accuracies and correlations of the weak supervision sources, re-weight and combine their labels, and produce the final set of clean, integrated labels.

Heuristic rules will be built based on features extracted using 2D skeletal points for each frame in the video.OpenPose will be used to used to extract 2D keypoints for each frame in a video. Features would be defined based on the 2D keypoints. These features will be used to build the heuristic rules to detect 'headache'.

A threshold will be determined for each feature upon analysis. Given a YouTube video, it will be split into multiple video clips in an overlapping sliding window fashion where the overlap is 50 frames and the window size is 100 frames. Then OpenPose will be run on each of the clip and2D skeletal data is extracted for each clip. Finally the above mentioned features are calculated using the 2D skeletal data of each clip. The heuristic rules will be

applied for the features calculated for each clip. The general rule for each heuristic will be that if the feature values are within a desired threshold for more than 50 frames in a clip, then the clip will get the label 'headache' and otherwise.

Finally it is shown that the accuracy of the unified weak supervision model is better than the accuracy of the standalone pretrained deep learning model for finding 'headache' moments in YouTube videos.

2. DATASET

NTU-RGBD dataset [10] and PKU-MMD dataset [11] are two publicly available datasets which has sample data for the action of "hand touch head" (headache). These two datasets will be used for training and evaluating our deep learning model. PKU-MMD dataset is also used for evaluating the weak supervision model's performance.

2.1 NTU-RGBD Dataset

NTU-RGBD dataset has 948 samples of headache. On an average, each video is five minutes long. This dataset contains RGB videos, depth map sequences, 3D skeletal data, and infrared (IR) videos for each sample. It is captured by three Kinect V2 cameras concurrently. The resolutions of RGB videos are 1920x1080, depth maps and IR videos are all in 512x424, and 3D skeletal data contains the 3D coordinates of 25 body joints at each frame.

2.2 PKU-MMD Dataset

PKU-MMD is a large-scale dataset focusing on long continuous sequences action detection and multi-modality action analysis. This dataset is also captured via the Kinect v2 sensor.This dataset also provides multi-modality data sources, including RGB, depth, Infrared Radiation and Skeleton. Depth maps are sequences of two dimensional depth values in millimeters. The resolution is 512×424.Joint information consists of 3-dimensional locations of 25 major body joints for detected and tracked human bodies in the scene. RGB videos are recorded in the provided resolution of 1920×1080. Infrared sequences are also collected and stored frame by frame in 512×424. There are 322 samples of headache. .

3. PROPOSED DEEP LEARNING MODEL

3.1 Hierarchical Co-occurrence Network

In this study,Hierarchical Co-occurrence Network (HCN) [2] has been used. The model architecture involves a two-stream framework and takes skeletal sequence and skeletal motion as inputs. The two branches of the network share the same architecture but their parameters are learned in an isolated fashion. The features are learned hierarchically. In the first stage of the network, point-level features are enciphered with two convolution layers where the kernel sizes along the joint dimension are kept 1. This enables the point-level features to learn point-level representation from 3D coordinates for each joint independently. Then the feature maps are transposed with parameter (0,2,1) so that the joint dimension is moved to channels of the tensor. In the second stage of the network, global co-occurrence features are extracted from the ensuing convolution layers. The global co-occurrence features are learned through aggregating co-occurrence features globally. Since the feature maps (tensors) are transposed, the information from one dimension can be aggregated globally if it is specified as channels while the other two dimensions encipher local context. In the final stage, the feature maps are flattened into a vector and are passed on through two fully connected layers for final classification. This network has only 0.8 million parameters (extremely small when compared to VGG19 pretrained on Image-Net) and allows us to easily train the network from scratch without having to need a pretrained model. The model architecture is shown in Figure 6.4. The detection framework is shown in Figure 3.2 to make the action prediction. pretrained on NTU dataset [10].

3.2 Optimal Hyperparameter Values

The optimal values of hyperparameters tested are shown in Table 3.1.

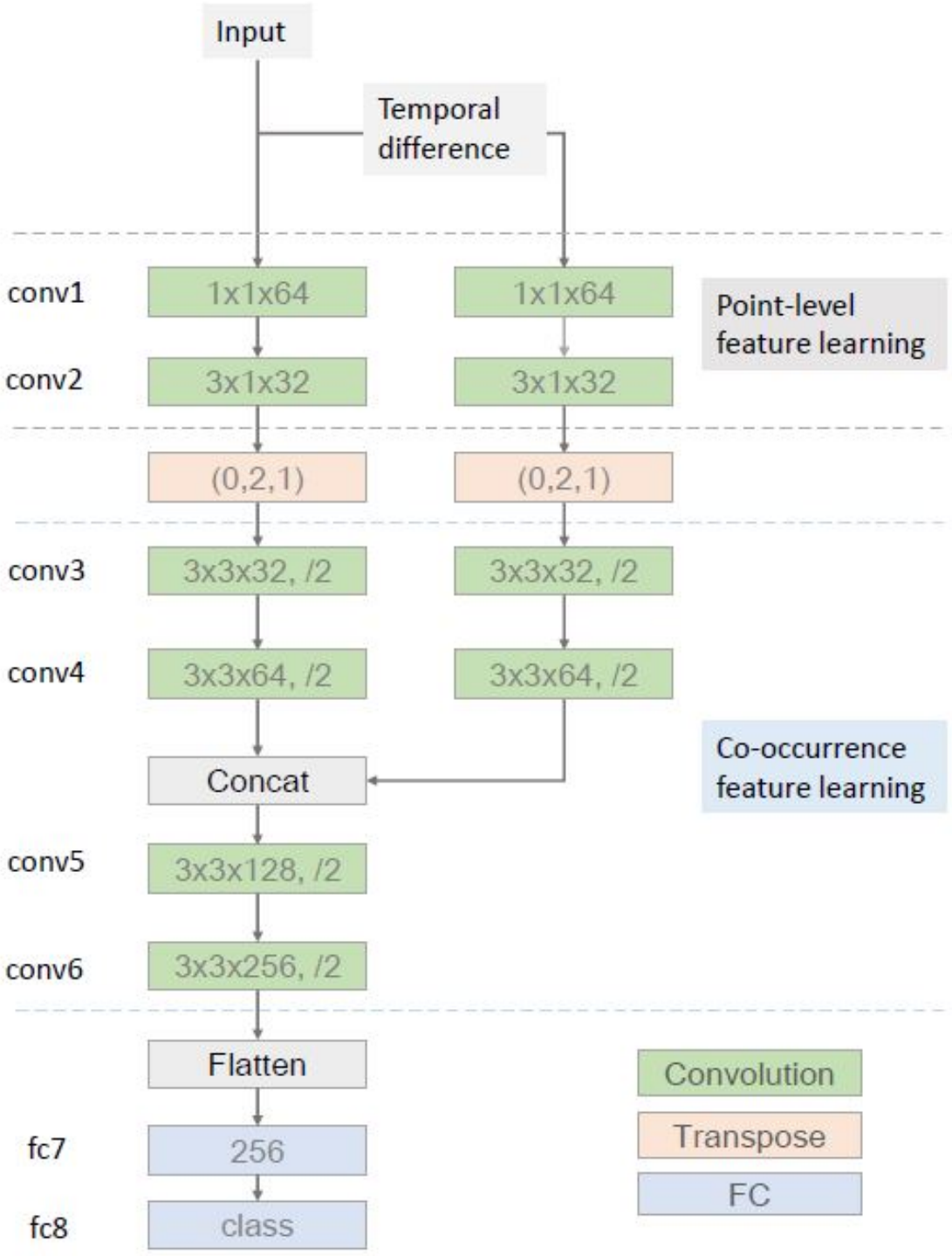


Figure 3.1: Model Architecture Overview [2]. Green blocks are convolution layers, where the last dimension denotes the number of output channels. A trailing "/2" means an appended Max-Pooling layer with stride 2 after convolution. A Transpose layer permutes the dimensions of the input tensor according to the order parameter. ReLU activation function is appended after conv1, conv5, conv6 and fc7 to introduce non-linearity

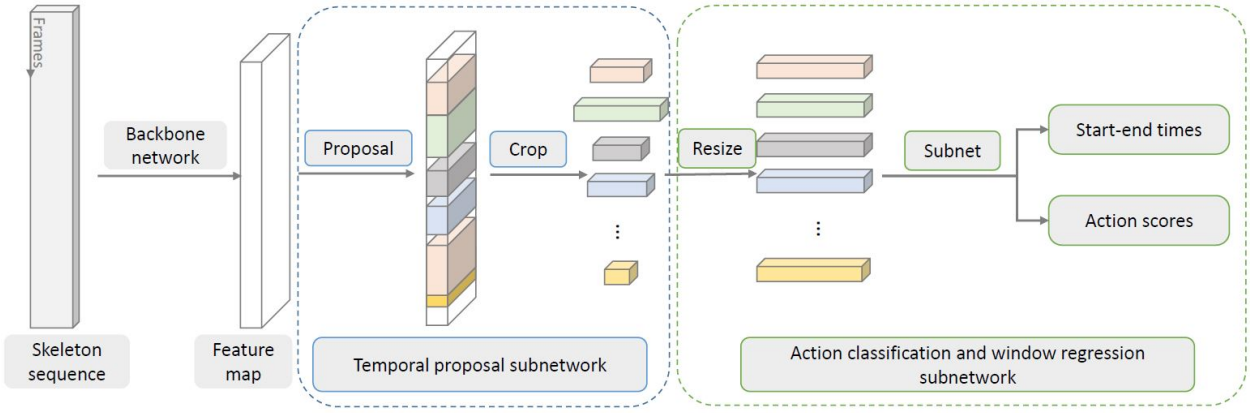


Figure 3.2: Temporal action detection framework [2]. Two sub-networks are designed for temporal proposal segmentation and action classification respectively.

Hyperparameters	Optimal Value
Patience	20
Optimizer	Adam
Dropout	0.5
Batch size	64
Learning rate	0.001
Number of epochs	400

Table 3.1: Optimal Hyperparameter values

3.3 Testing Performance

HCN model was trained with the dataset consisting 946 videos with action "hand touch head" in the NTU dataset [10] as positive samples and 960 videos from the 24 action classes in the NTU dataset [10] consisting of 40 samples from each of the 24 classes as negative samples. The ignored classes in the NTU dataset were (1, 2, 3, 4, 7, 14, 15, 18, 19, 20, 21, 28, 31, 32, 33, 34, 37, 38, 39, 41, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60). The test data consisted of 322 short videos with action "hand touch head" (headache) extracted from the PKU-MMD dataset [11]. 2D skeletal from these 322 videos were extracted using OpenPose. Raw skeleton data was processed and then the trained model was tested on this new data. The classification metrics for the cross-view evaluation and cross-subject evaluation are shown in Table 3.2 and Table 3.3.

Metrics	Training	Validation	Testing
Accuracy	92.1%	99%	95.6%

Table 3.2: Classification metrics for cross-view evaluation

Metrics	Training	Validation	Testing
Accuracy	92.1%	99%	97%

Table 3.3: Classification metrics for cross-subject evaluation of deep learning model

To evaluate the retrained model on finding moments of headache, the model was tested on 322 long videos with an average length of 3 minutes where each video has multiple

actions in it. Given a YouTube video, frames per second and total frames in the video using OpenCV and it is split into multiple video clips in an overlapping sliding window fashion where the overlap was 50 frames and the window size was 100 frames. Then OpenPose is run on each of the clip and 2D skeletal data is extracted for each clip. Finally the trained model is run on each clip with its 2D skeletal data as input and makes the action prediction on each clip. The model performed well in finding moments of headache.

4. WEAK SUPERVISION

4.1 Weak Supervision

Weak supervision is getting lower quality labels more efficiently at a higher abstraction level. These noisy, low quality, huge training datasets are built using techniques such as using cheap annotators, programmatic scripts, more creative and high-level input from domain experts etc.Getting higher-level supervision over unlabelled data from subject matter experts can be done using techniques such as heuristics, distant supervision,constraints, expected distributions and invariances. Crowd-sourcing is one example of cheap, low quality supervision. Knowledge bases, pre-trained models are examples of programmatic scripts.

The objective of weak supervision is the same as supervised learning. In the weak supervision setting, instead of a hand labeled training dataset there is unlabeled data and one or more weak supervision sources given by a human subject matter expert. Each weak supervision source has a coverage set over which it is defined and an accuracy which is the expected probability of the true label over its coverage set. The biggest motivation for using weak supervision is that these weak label distributions ensure human supervision is achieved more cheaper and efficient. In this study, pretrained models and heuristics are going to be used as weak supervision sources.

4.2 Learning a Unified Weak Supervision Model

Given a set of multiple weak supervision sources, the key technical challenge is how to unify and de-noise them. This is because the weak supervision sources are noisy, sometimes contradicting or correlated. In general, the task is defining and learning a single weak supervision model defined over the weak supervision sources. In this study, Snorkel [19] has been used which helps us in modeling the weak supervision sources into a unified

weak supervision model.

4.3 Snorkel

Snorkel is a unique system for programmatically developing large training datasets without hand labeling the data. In Snorkel, weak supervision sources are encoded as labeling functions that evinces arbitrary heuristic rules. These heuristics can have uncharted accuracies and correlations. Snorkel denoises the labels generated by the labeling functions with no prior knowledge of ground truth. A class is defined in Snorkel that learns a model (label model) of the labeling functions (LFs)' conditional probabilities of outputting the true (unobserved) label Y, P(LF|Y), and uses this learned model to re-weight and combine their output labels. This class is based on the approach in Training Complex Models with Multi-Task Weak Supervision [4]. Currently this class uses a conditionally independent label model, in which the LFs are assumed to be conditionally independent given Y.

4.4 Training Complex Models with Multi-Task Weak Supervision

In this paper [4], they have proposed a scheme for integrating and modeling the labeling functions (weak supervision sources) by considering them as labeling different related sub-tasks of a problem. This approach is named "Multi-Task Supervision". In this method, an inverse generalized co-variance matrix of the junction tree of a labeling function dependency graph is computed and a matrix completion-style method with respect to these empirical statistics is carried out. The resulting estimated conditional probabilities of the labeling functions (LFs), P(LF|Y) are then fixed as the parameters of the label model used to re-weight and combine the labels output by the LFs. The current version of Snorkel uses a conditionally independent label model as stated in the above paragraph. Here the problem is setup as below. Let X $\in X$ represent a data point and Y = $[Y_1, Y_2, \cdots, Y_t]^T$ represent a vector of categorical task labels, Y_i

$in\ \{1, \cdots, k_i\}$ corresponding to *t* tasks, where (X,Y) is drawn from i.i.d. from a certain distribution *D*. $\mathcal{y}$ represents a feasible set of label vectors such that $\mathrm{Y} \in \mathcal{y}$. This study learns the label model $P_\mu(Y|\lambda)$, parameterized by a vector of source correlations and accuracies μ. This label model takes the noisy labels $\lambda = \lambda_1, \cdots, \lambda_m$ for some subset of *t* tasks outputted by the 'm' labeling functions $s_i \in S$ and provides a single probabilistic label vector $\tilde{Y}$. Let the coverage set $\tau_i \subseteq \{1, \cdots, t\}$ be the fixed set of tasks for which the *i*th source outputs non-zero labels, such that λ_i

$in\ \mathcal{y}_{\tau_i}$. The key technical challenge in this study was to estimate the parameters μ without access to ground truth labels. The dependency structure of the labeling functions is represented as a graph G_{source} = (V,E), where V = $\{Y, \lambda_1, \cdots, \lambda_m\}$. An example of a weak supervision source dependency graph and its junction tree representation is shown in Figure 4.6. Since it is assumed that the labeling functions are conditionally independent, there will be no edges between any λ_i and λ_j for all $i \neq j$ in the graph G_{source}. In order to learn a label model over several sources, they define sufficient statistics over the random variables in G_{source}. The statistic *C* is defined as the set of cliques in G_{source} and an indicator random variable is defined for the event of a clique C $\in$ *C* with a set of values y_C :

$$\psi(C, y_C) = 1\{\cap_{iinC} V_i = (y_C)_i\},$$

where $(y_C)_i \in \mathrm{y}_{\tau_i}$.

$\psi(\mathrm{C}) \in \{0,1\}^{\prod_{i \in \mathrm{C}}(|\mathrm{y}_{\tau_i}|-1)}$ is defined as the indicator random variables vector for all combinations of all except one of the labels given off by each variable in clique C. There-

fore a minimal set of statistics are defined and $\psi(C)$ is defined correspondingly for any set of cliques C $\subseteq$ *C*. Finally $\mu = E[\psi(C)]$ is the sufficient statistics vector for the desired label model that needs to be learned. The biggest issue here is that the true labels Y are not observed. This study analyzes the co-variance matrix of an observable subset of the cliques in G_{source}. This leads to a matrix completion style approach for estimating μ. Two pieces of information are leveraged here. The first one is the observability part of Cov[$\psi(C)$] and the second one is that the inverse co-variance matrix Cov$[\psi(C)]^{-1}$ is structured in line with G_{source}, i.e., if there is no edge between λ_i and λ_j in G_{source}, then the respective values are 0.

Two disjoint subsets of *C* are considered. Firstly, the set of observable cliques, *O* $\subseteq$ *C* (cliques not containing Y) and secondly *S* $\subseteq$ *C* (the separator set cliques of the junction tree). Here they make an assumption that $S = \{Y\}$. The co-variance matrix of the indicator variables *O* $\cup$ *S*, Cov$[\psi(O \cup S)]$ is expressed in block form as:

$$Cov[\psi(O \cup S)] \equiv \Sigma = \begin{bmatrix} \Sigma_O & \Sigma_{OS} \\ \Sigma_{OS}^T & \Sigma_S \end{bmatrix}$$

and its inverse is defined as:

$$K = \Sigma^{-1} = \begin{bmatrix} K_O & K_{OS} \\ K_{OS}^T & K_S \end{bmatrix}$$

Here, $(\Sigma)_O = Cov[\psi(O] \in (R)^{d_O x d_O}$ is the observable block of Σ, where d_O = $\Sigma_{C \in O} \prod_{i \in C} (|y_{T_i}| - 1)$. Then, $\Sigma_{OS} = Cov[\psi(O, \psi(S)]$ is the unobserved block which is a function of μ, the label parameters desired to be determined. Finally $\Sigma_S = Cov[\psi(S)] = Cov[\psi(Y)]$ is a function of the class balance P(Y).The complete form of Σ_S is the co-variance of the $|y|$ - 1 indicator variables for each individual value of Y but one. Since it is assumed that the labeling functions are conditionally independent given Y, only a single

indicator variable for Y is enough. Therefore Σ_S is a scalar. Σ_S is a function of the class balance P(Y). Here it is assumed that it is either known or calculated based on an unsupervised method detailed in Appendix A.3.5 of [4] paper. Therefore given Σ_O and Σ_S, this study desires to estimate the vector Σ_{OS} from which μ can be estimated.

With application of block matrix inversion lemma, it is seen that:

$$K_O = \Sigma_O^{-1} + c(\Sigma_O^{-1}\Sigma_{OS}\Sigma_{OS}^T\Sigma_O^{-1})$$

,

where $c = (\Sigma_S - \Sigma_{OS}^T\Sigma_O^{-1}\Sigma_{OS})^{-1} \in R^+$

They assign $z = \sqrt{\Sigma_O^{-1}\Sigma_{OS}}$

Therefore, $K_O = \Sigma_O^{-1}$ + zz^T

In the above equation the right hand side (RHS) contains an empirically observable term, Σ_O^{-1} and rank-one term zz^T. This RHS can be solved directly to compute μ. For the left hand side, they apply an extension of Corollary 1 from [20] explained in Appendix A.3.2 of [4] and finally determined that K_O has zeros decided by the dependency structure between the sources in G_{source}. This recommends an algorithmic method to estimate z as a matrix completion problem as a means to compute μ (Algorithm 1 in [4]). In order to explain in an detailed manner, they consider Ω to be the set of indices (i,j) where $K_{Oi,j} = 0$, decided by the G_{source}. This yields a scheme of equations:

$0 = (\Sigma_O^{-1})_{i,j} + (zz^T)_{i,j}$ for (i,j) $\in \Omega$

The above system equations is a matrix completion problem. They define $||A||_\Omega$ to be the Frobenius norm of A with entries absent in Ω set to zero. Therefore the above equation

can be rewritten as:

$$||(\Sigma_O)^{-1} + zz^T||_\Omega = 0$$

They solve the above equation to compute z and therefore estimate Σ_{OS}. Finally from Σ_{OS}, the label model parameters μ are computed algebraically. The next two paragraphs are about the proposed condition for checking for identifiability and the proposed source accuracy estimation algorithm.

4.4.1 Checking for Identifiability

The problem is to determine which dependency structures of G_{source} drive to singular solutions for μ. Here they define G_{inv} to be the inverse of G_{source}. Here Ω represents the set of edges in G_{inv} widened to comprise of all indicator variables $\psi(C)$. Then they assume M_Ω to be a matrix with $|\Omega|$ x d_O as dimensions such that each row in M_Ω compares to a pair (i,j) $\in \Omega$ with ones in positions i and j and zeros elsewhere. Now log of the squared entries in the above equation leads to a system of linear equations $M_\Omega l = q_\Omega$ where $l_i = log(z_i^2)$ and $q_{(i,j)} = log(((\Sigma_O^{-1})_{i,j})^2)$. They solve this system by adding sources and uniquely compute z_i^2. This shows that the label model is identifiable up to sign. Given the estimates of z_i^2, it is observed that the sign of a single z_i decides the sign of all other z_j reachable from z_i in G_{inv}. Therefore in order to get a unique solution, a sign needs to picked for each connected component in G_{inv}. Here, since it is assumed that the sources are independent, this study selects the sign of the z_i that leads to higher mean accuracies of the sources.

4.4.2 Source Accuracy Estimation Algorithm

It has been shown that with a set of sources with correlation structure G_{source} which is identifiable, yielding a singular z, the accuracies μ can be computed using Algorithm 1 of [4]. This algorithm uses the function ExpandTied which is a simple algebraic expansion

of tied parameters (μ) per the model with conditional independence assumption. Thus the label model learns its parameters without ground truth labels and combines the labels from different sources into a final set of integrated labels.

4.5 Heuristics

The preliminary results shown above are pertained to the 2D skeletal based deep learning model. The next steps in this research are to develop several heuristic rules to detect 'headache' moments in YouTube videos, integrate along with the pretrained deep learning model (another weak supervision source) and generate a unified weak supervision model using Snorkel. Snorkel automatically estimates the accuracies and correlations of the weak supervision sources, re-weight and combine their labels, and produce the final set of clean, integrated labels.

Heuristic rules have been built based on features extracted using 2D skeletal points for each frame in the video.OpenPose is used to used to extract 2D keypoints for each frame in a video. (There are totally 25 body keypoints and 21 hand keypoints as shown in Figures 4.1 and 4.2 respectively. The following features are defined based on the 2D keypoints:

1. *Shoulder angle*: Consider the 25 body pose landmarks as illustrated in Figure 4.1. For $i = 0, 2, \cdots, 24$, let (x_i, y_i) denote the coordinates of the ith body pose landmark in the image. Shoulder angle ($\mathcal{S}_L$ for left hand and $\mathcal{S}_R$ for right hand) is defined as the below formula:

$$\mathcal{S}_L = \frac{\overrightarrow{BA} \cdot \overrightarrow{BC}}{|\overrightarrow{BA}| \cdot |\overrightarrow{BC}|},$$

where

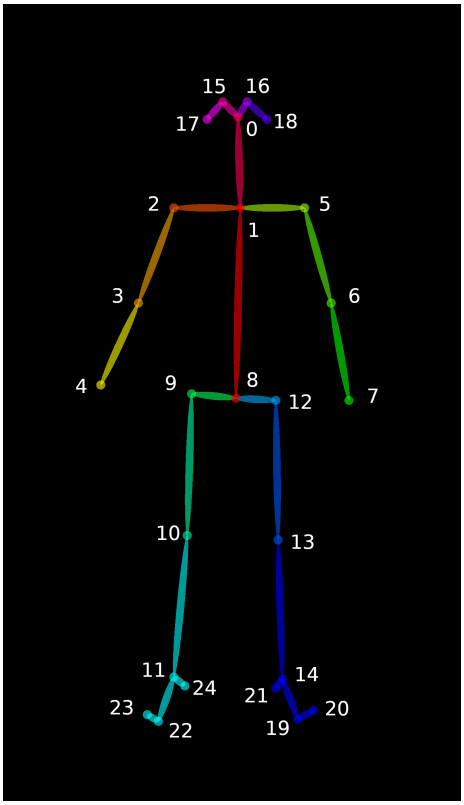


Figure 4.1: An illustration of the 25 body pose landmarks for a human [3].

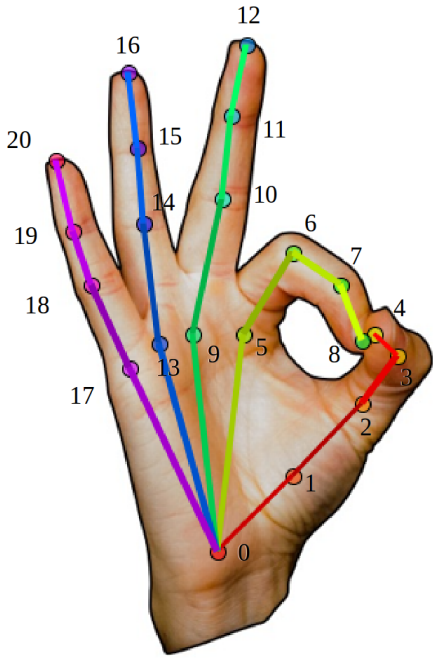


Figure 4.2: An illustration of the 21 hand landmarks for a human [3].

$$A = (x_1, y_1), B = (x_5, y_5), C = (x_6, y_6)$$

$$|\overrightarrow{BA}| = \sqrt{(x_1 - x_5)^2 + (y_1 - y_5)^2}$$

$$|\overrightarrow{BC}| = \sqrt{(x_6 - x_5)^2 + (y_6 - y_5)^2}$$

$$\mathcal{S}_R = \frac{\overrightarrow{QP} \cdot \overrightarrow{QR}}{|\overrightarrow{QP}| \cdot |\overrightarrow{QR}|},$$

where

$$P = (x_1, y_1), Q = (x_2, y_2), R = (x_3, y_3)$$

$$|\overrightarrow{QP}| = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

$$|\overrightarrow{QR}| = \sqrt{(x_3 - x_2)^2 + (y_3 - y_2)^2}$$

2. *Hand to Head distance ratio*: Hand to Head distance ratio ($\mathcal{D}_L$ for left hand and $\mathcal{D}_R$ for right hand) is defined as the below formula:

$$\mathcal{D}_L = \frac{\sqrt{(x_0 - x_7)^2 + (y_0 - y_7)^2}}{\sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2}}$$

$$\mathcal{D}_R = \frac{\sqrt{(x_0 - x_4)^2 + (y_0 - y_4)^2}}{\sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2}}$$

3. *Hand to Eye distance ratio*: Hand to Eye distance ratio ($\mathcal{E}_L$ for left hand and $\mathcal{E}_R$ for right hand) is defined as the below formula:

$$\mathcal{E}_L = \frac{\sqrt{(x_{18} - x_7)^2 + (y_{18} - y_7)^2}}{\sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2}}$$

$$\mathcal{E}_R = \frac{\sqrt{(x_{17} - x_4)^2 + (y_{17} - y_4)^2}}{\sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2}}$$

An illustration of 2D keypoints superimposed on a human with his left hand on head is shown in Figure 4.3. It is observed that the vertical distance between the end of fore arm and the head also varies with respect to the position of the fore arm. Also the angle between the shoulder and the upper arm varies with the position of the upper arm. Therefore it was decided to use $\mathcal{S}$, $\mathcal{E}$ and $\mathcal{D}$ as features to build the heuristic rules to detect 'headache'. A threshold has been determined for each heuristic based on analysis of the trends of each of the features. The threshold has been fixed as '1.5' for the hand to head distance ratio and hand to head distance ratio heuristics. Hand to head heuristic will label the video clip as 'headache' if the value of the $\mathcal{D}$ feature is less than 1.5 for at least 35 frames of the video clip and 'not headache' otherwise. Similarly, Hand to eye heuristic will label the video clip as 'headache' if the value of the $\mathcal{E}$ feature is less than 1.5 for at least 35 frames of the video clip and 'not headache' otherwise. the For the shoulder angle heuristic the threshold has been set as 120°. Similarly, this heuristic will label the video clip as 'headache' if the value of the $\mathcal{S}$ feature is greater than 120° for at least 35 frames of the video clip and 'not headache' otherwise. Given a YouTube video, it will be split into multiple video clips in an overlapping sliding window fashion where the overlap is 50 frames and the window

size is 100 frames. Then OpenPose will be run on each of the clip and2D skeletal data is extracted for each clip. Finally the above mentioned features for each frame in the clip are calculated using the 2D skeletal data of each clip. The heuristic rules will be applied for the features calculated for each clip. The general rule for each heuristic will be that if the feature values are within a desired threshold for more than 35 frames in a clip, then the clip will get the label 'headache' and 'not headache' otherwise.

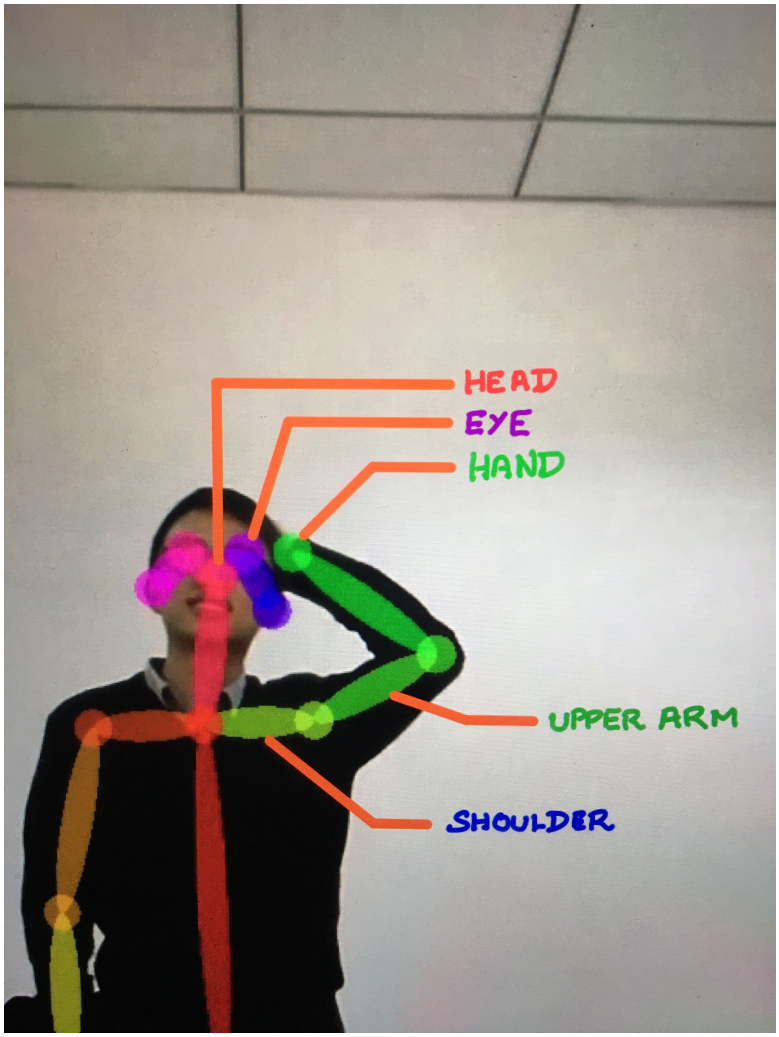


Figure 4.3: An illustration of 2D keypoints superimposed on a human.

4.6 Dataset for Weak Supervision

Weak supervision is used to label unlabeled dataset. Therefore, Weak supervision technique 'Snorkel' is applied on YouTube videos.

4.7 Testing Performance

For preliminary evaluation of the unified weak supervision model, 316 videos of headache from the PKU-MMD dataset has been chosen. 2D key-points were extracted from these videos using OpenPose. Then $\mathcal{S}$ and $\mathcal{D}$ features were computed for all the videos. The labels generated by testing the trained deep learning model on this dataset and the labels generated by applying the heuristic rules on this dataset were given as input to Snorkel. Snorkel generated the unified weak supervision model by combining these weak supervision sources (trained model and heuristic rules) and this unified weak supervision model generated the final integrated labels for this dataset. The classification metrics of the trained deep learning model (DL) and the unified weak supervision model (WM) are shown in Table 4.1. It was observed that the accuracy has been increased from 98.4% to 99.7% when weak supervision is applied.

As the final step, the weak supervision technique was applied to YouTube videos to detect 'headache' moments. A label matrix was generated which consisted of data points (videos in this case) along the rows and labels from the three heuristics and the trained deep learning model along each of the columns. This label matrix was as input to Snorkel. Snorkel generated a unified weak supervision model by combining these weak supervision sources and output the final integrated labels. A couple of examples where the pretrained model failed to predict headache moment but the weak supervision model generated by Snorkel classified them as 'headache' are shown in Figures 4.4 and 4.5. This shows heuristic rules defined above have helped especially in cases where the pretrained model has failed to predict correctly. The classification metrics of the trained deep learn-

ing model (DL) and the unified weak supervision model (WM) on YouTube videos are shown in Table4.2. It was observed that the accuracy has been increased from 82.5% to 94.7% when weak supervision technique is applied to extract 'headache' moments from YouTube videos.

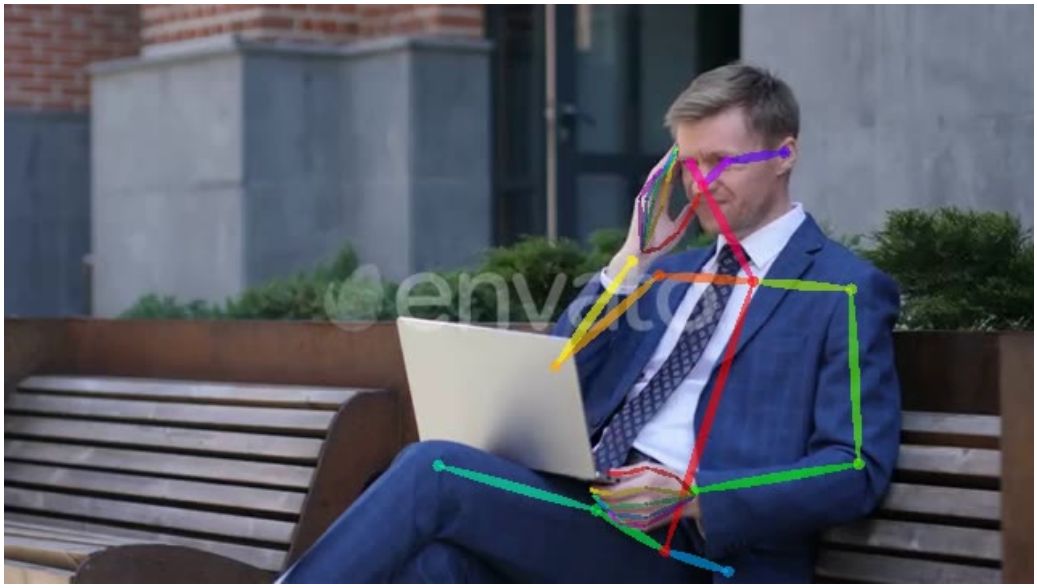


Figure 4.4: Example 1

Finally it is observed that the performance metrics of the unified weak supervision model is better than that of the standalone pretrained deep learning model for finding 'headache' moments in YouTube videos.

Figure 4.5: Example 2

Figure 4.6: An example of a weak supervision source dependency graph G_{source} (left) and its junction tree representation (right), where Y is a vector-valued random variable with a feasible set of values, Y
$in\ y$. Here, the output of sources 1 and 2 are modeled as dependent conditioned on Y. This results in a junction tree with singleton separator sets, Y. Here, the observable cliques are $O = \{_1, \lambda_2, \lambda_3, \lambda_4, \{\lambda_1, \lambda_2\}\} \subset C$ [4]

Metrics	Accuracy	Sensitivity	Specificity
DM	98.4%	98.4%	100%
WM	99.7%	99.7%	100%

Table 4.1: Classification metrics of trained deep learning model and unified weak supervision model

Metrics	Accuracy	Sensitivity	Specificity
DM	82.5%	67.9%	93.5%
WM	94.7%	98.8%	91.7%

Table 4.2: Classification metrics of trained deep learning model and unified weak supervision model on YouTube Videos

5. PERFORMANCE ON YOUTUBE VIDEOS

5.1 How to detect “moments” of target action/emotion

The model is designed and trained to take 2D skeletal data of a video (maximum of 300 frames) as input and makes the prediction. Here in a YouTube video, moments of the target action may be present in some parts of the video or the entire video might contain the target action. Given a YouTube video, frames per second and total frames in the video are computed using OpenCV and it is split into multiple video clips in an overlapping sliding window fashion where the overlap was 50 frames and the window size was 100 frames. Then OpenPose is run on each of the clip and 2D skeletal data is extracted for each clip. Finally the trained model is run on each clip with its 2D skeletal data as input and makes the action prediction on each clip.

5.2 View Found “Moments” in iLab Website

The label for the target action is "headache". 80 true positive moments have been predicted by the trained deep learning model so far and the videos with labels have been added to the iLab website. They can be viewed by searching for the label "headache".

5.3 Performance: Accuracy

In this study, metrics such as False Positive Rate (FPR) and False Negative Rate (FNR) have been used to evaluate the classification model. The definition of the terms used in estimating the metrics are defined below:

- True Positives (TP): The total number of accurate predictions that were ’positive’ or ’0’. In our study, this is the total number of samples correctly predicted as headache.
- False Positives (FP): The total number of inaccurate predictions that were ’positive’

or '0'. In our study, this is the total number of samples incorrectly predicted as headache.

- True Negatives (TN): The total number of accurate predictions that were 'negative' or '1'. In our study, this is the total number of samples correctly predicted as "not headache".

- False Positives (FN): The total number of inaccurate predictions that were 'negative' or '1'. In our study, this is the total number of samples incorrectly predicted as "not headache".

The definition of classification metrics such as False Positive Rate (FPR) and False Negative Rate (FNR) and their mathematical expressions are presented below.

- False Positive Rate (FPR): It is a measure of accuracy for a deep learning model. It is the ratio of the number of false positive classifications to the total number of negative classifications.

  $$FPR = \frac{FP}{FP + TN}$$

- False Negative Rate (FNR): It is the probability that a true positive will be missed by the deep learning model. It is the ratio of the number of false negative classifications to the total number of positive classifications.

  $$FNR = \frac{FN}{FN + TP}$$

YouTube videos with the word 'headache' in their title were chosen for testing. If they were chosen randomly, the FPR and FNR would still be around the same range since the deep learning model has been trained on an extensive dataset consisting of different actions and the heuristics functions have been been designed to correctly detect headache moments. Please note that while testing, the YouTube video is split into multiple video

clips in an overlapping sliding window fashion where the overlap was 50 frames and the window size was 100 frames. For example frames 0 to 100, 50 to 150 and 100 to 150 are considered individual video clips during testing and the classification metrics are calculated based on this. The classification metrics of the trained deep learning model (DL) and the unified weak supervision model (WM) tested on YouTube videos are shown in Table 5.1. The application of weak supervision technique has significantly improved the results.

Metrics	FPR	FNR
DL	0.06%	0.32%
WM	0.08%	0.01%

Table 5.1: Classification metrics for cross-subject evaluation of deep learning model

5.4 Improve Accuracy of Model on YouTube Videos

In order to improve the accuracy of the model on YouTube videos, weak supervision technique was applied. A few heuristics such as $\mathcal{S}$, $\mathcal{E}$ and $\mathcal{D}$ were developed; and used along with the trained deep learning model as weak supervision sources. Snorkel was used to combine these weak supervision sources into a unified weak supervision model and generate the final integrated labels on YouTube videos. The FPR got reduced from 32% to nearly 1% and the FNR increased minimally by 2% after the application of weak supervision technique.

5.5 Code in Github

5.5.1 Install Dependencies

- pandas
- numpy

- scipy
- sklearn
- tensorflow-gpu
- keras
- matplotlib
- snorkel

5.5.2 Execution

1. Download and Split YouTube Videos:
    - Download the YouTube video.
    - Compute frames per second and total frames in the video using OpenCV.
    - Split the video into multiple video clips in an overlapping sliding window fashion where the overlap is 50 frames and the window size is 100 frames.
    - The above operations can be executed by running PyTube.IPYNB
2. Generate landmarks:
    - Videos are submitted to the OpenPose portable execution file to generate JSON files of the body landmarks, and the left and right hand landmarks for each frame in the video.
3. Preprocess data for Heuristics:
    - Read the JSON files (landmarks for each frame in the video) in every folder.
    - Collect the full body (pose), and the left and right hand landmarks.

- Calculate the angle between upper arm and shoulder for each frame.
- Calculate the distance between fore arm and head for each frame.
- Calculate the distance between fore arm and eye for each frame.
- Output the file for each YouTube video.
- The above operations can be executed by running parse_json_head_snorkel.IPYNB

4. Preprocess data Deep Learning model:

    - Read the JSON files (landmarks for each frame in the video) in every folder.
    - Collect the full body (pose) landmarks.
    - Git clone https://github.com/huguyuehuhu/HCN-pytorch.git and install dependencies. This is the HCN model used as one of the weak supervision source.
    - Combine the landmarks generated for all the frames into a single dictionary.
    - Output the file for each video.
    - The above operations can be executed by running 'parse_json_head_model.IPYNB'

5. Deep Learning model:

    - Preprocess the training and test file for each video, combine them and store data and label information.
    - Fit the Deep Learning model on training data.
    - Test the model on testing data.
    - The above operations can be executed by following the instructions given in https://github.com/huguyuehuhu/HCN-pytorch

6. Unified weak supervision model:

- Pass the file for each YouTube with features extracted from 2D landmarks to each heuristic.
- Combine them into a single data-frame. This consists of video IDs along X axis and labels from heuristics on Y axis.
- Append the pretrained model predictions to the above data-frame.
- Pass the data-frame values to Snorkel.
- Output the final integrated labels
- The above operations can be executed by running 'Weak_Supervision_.IPYNB'

5.5.3 Video Link

The link for the Github repository that includes the codes, videos and data can be found in the link https://github.tamu.edu/jug-971990/Finding-Headache-moments-from-YouTube-Videos-using-Weak-Supervison.

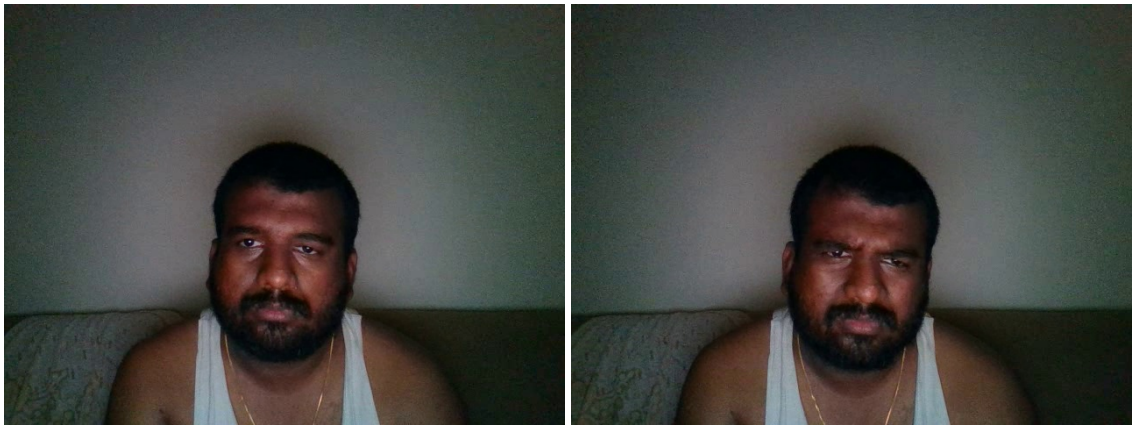
6. OTHER PROJECTS

6.1 Eye Squinting action detection

Eye squinting or narrowing eyes is one of the body languages that may indicate displeasure, uncertainty or evaluation. Eye Squinting involves lowering of the brows and narrowing eyes to a certain extent. This can happen in a fraction of a second. It is said that squinting happens due to emotional or physical pain. Squinting was one of the prominent features observed during online exam videos whenever the questions were hard. The present study developed a squint detection model using 2D facial landmarks. 5 videos of totally 7.5 minutes recorded while the subject took an online exam was considered in this study. Eyes and eyebrow landmarks were detected from each frame of the video using the Face-Alignment facial landmark detector [21]. Features such as eyebrow closeness, eyebrow height from upper eyelid and eye openness served as inputs to a Deep Neural Network (DNN) model. Examples of eye squinting and facial 2D keypoints superimposed on images are shown in Figures 6.1 and 6.2 respectively. A classification accuracy of 93.35%, sensitivity of 97.32% and specificity of 89.46% were achieved using the DNN model. The results suggest that eyebrow closeness, eyebrow height from upper eyelid and eye openness are useful features for the detection of squint in videos.

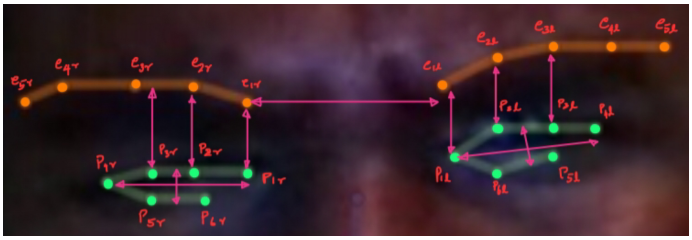
6.2 Hand on Chest action detection

Whenever the old people have any kind of uneasiness in their chest, they tend to rub their hand(s) over their chest. In this project, a Deep Learning model has been used to read videos people and detect if they have the hand on the chest. The model architecture is shown in Figure 6.3 . The whole body pose landmarks were extracted using OpenPose. Features such as angle between forearm and upper-arm and distance between end of forearm and were computed using the 2D key points. The training data-set contained 28 videos

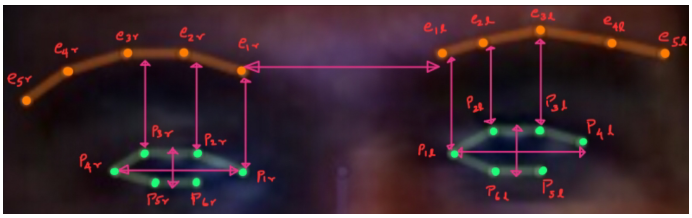


(a) Normal eyes (NE) (b) Squinted eyes (SE)

Figure 6.1: Normal and Squinted eyes



(a) NE with landmarks



(b) SE with landmarks

Figure 6.2: Normal and Squinted eyes with landmarks

and were tested on 9 videos. A classification accuracy of 94.94%, sensitivity of 96.21% and specificity of 85.58% were achieved.

6.3 Watching Television action detection

In this project, a Deep Learning model was developed to read videos people and detect if they are watching television. The model architecture is shown in Figures 6.4, 6.5 and 6.6. The whole body pose landmarks were extracted using OpenPose. Euclidean distances between the 2D key points were computed. Features were also extracted from Inception-V3 architecture pretrained on Stanford 40 dataset.Stanford40 dataset [8] consists of 9532 images belonging to 40 different actions and one of them was 'watching television'. Features extracted from 2D keypoints and features extracted from the pretrained model were combined and passed on to the deep learning model. The training data-set contained 43 videos and were tested on 9 videos. A classification accuracy of 94.57%, sensitivity of 94.55% and specificity of 71.97% were achieved.

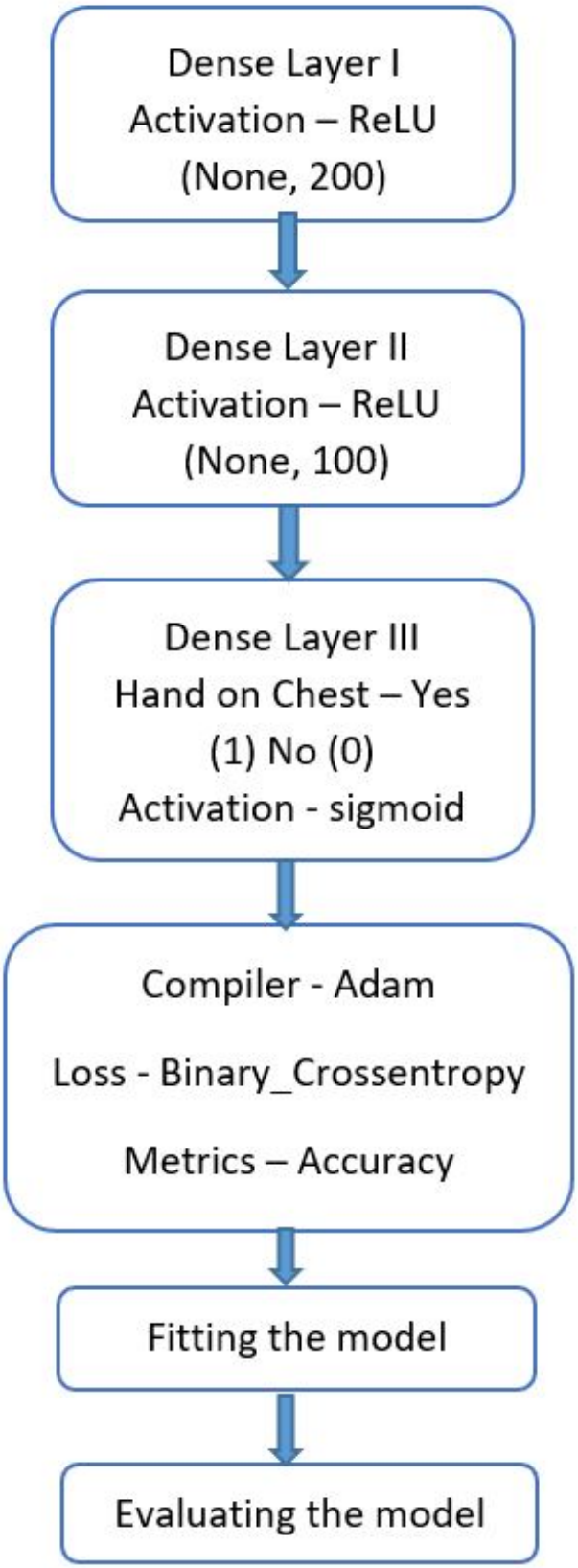


Figure 6.3: Model Architecture for hand on chest action.

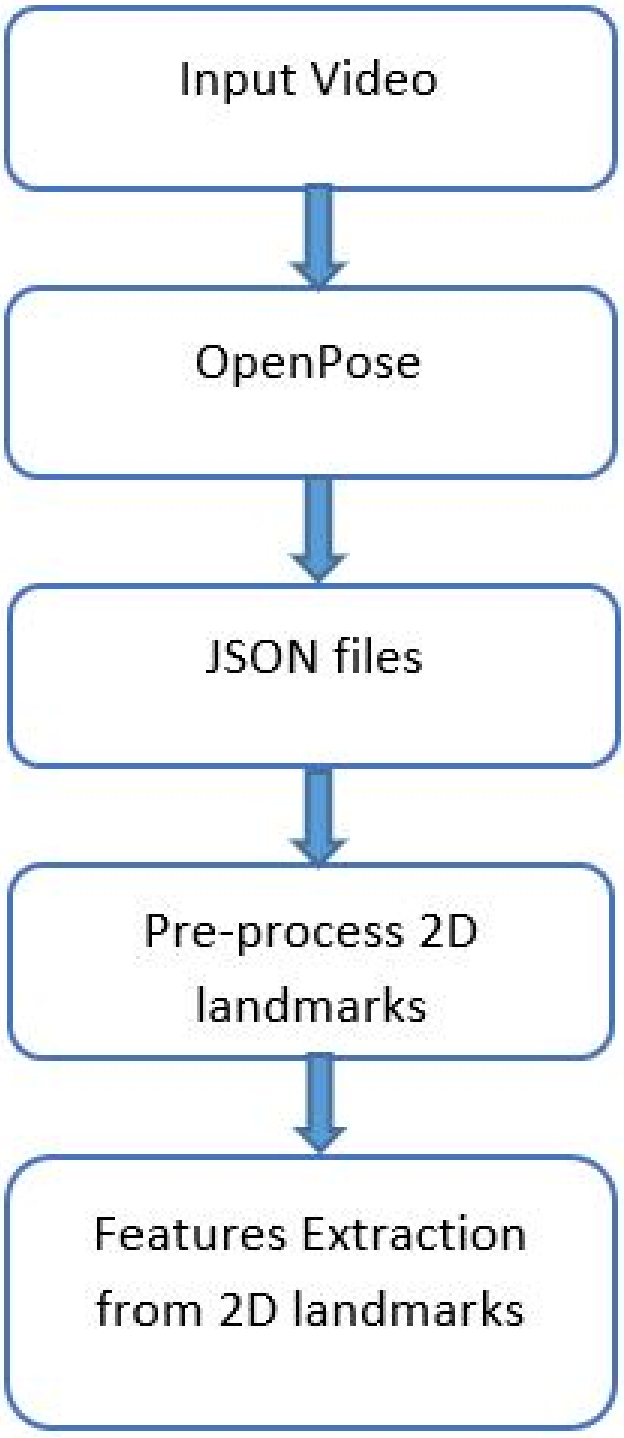


Figure 6.4: Feature extraction from OpenPose.

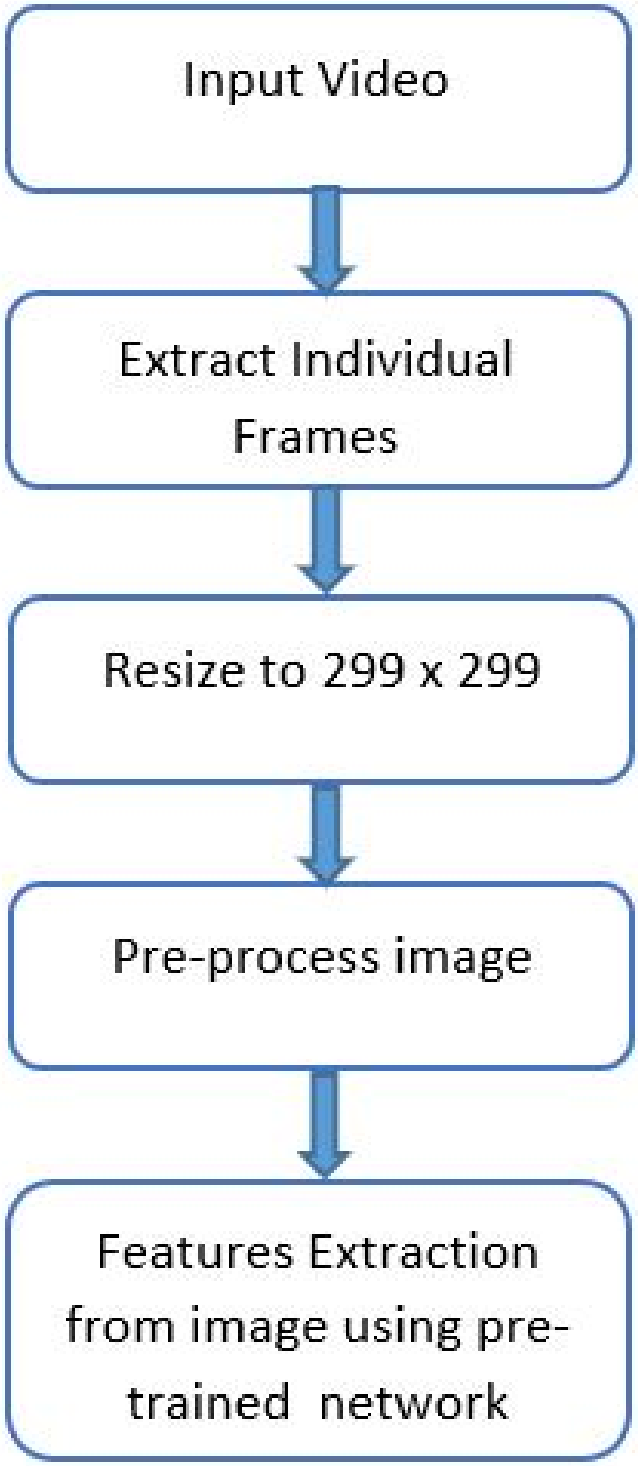


Figure 6.5: Feature extraction from pretrained Inception-V3 model.

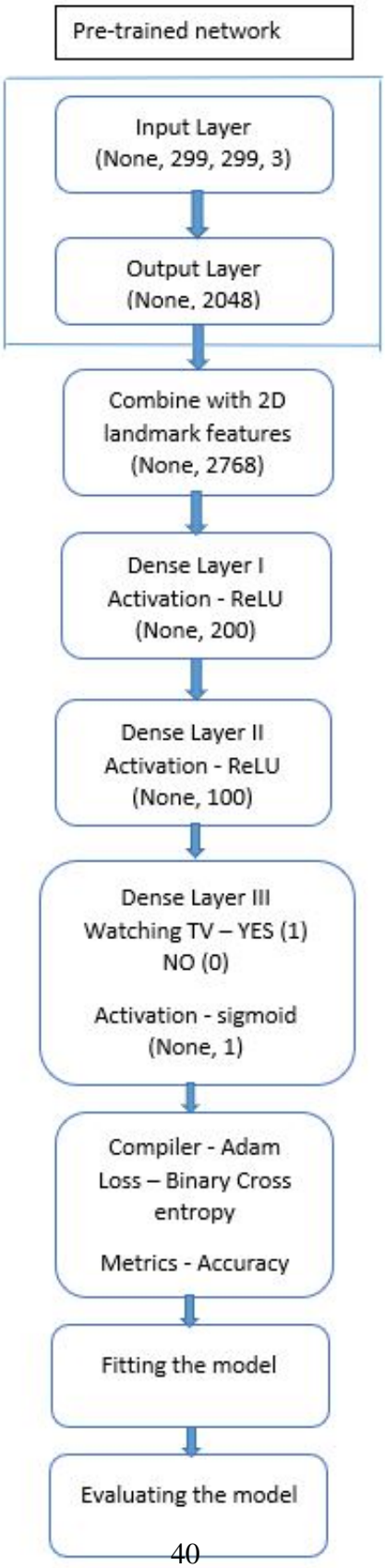


Figure 6.6: Model Architecture for watching television action.

7. CONCLUSIONS AND FUTURE WORK

7.1 Conclusions

Finding high quality 'headache' moments from YouTube videos has lots of challenges. In this research more accurate weak supervision sources such as a 2D skeletal based deep learning model trained on a large labeled dataset (NTU-RGBD) [10], heuristic rules using hand crafted features extracted from 2D key points have been developed and a unified weak supervision model is generated using Snorkel [1] to find accurate 'headache' moments from YouTube videos.

7.2 Future Work

As part of the future research work, firstly, building a deep learning architecture for action detection in videos from scratch will be considered. In this study, the idea of improving the efficiency of the model has not been explored. Future research will involve working on techniques to improve the efficiency of the model. 2D keypoints were used in this study. Estimating 3D keypoints from videos will be explored as part of the future study. In real world scenario, the illumination is not optimal always. Future research will also focus on building model that works even there is poor illumination.

REFERENCES

[1] A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré, "Snorkel: Rapid training data creation with weak supervision," in *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, vol. 11, p. 269, NIH Public Access, 2017.

[2] C. Li, Q. Zhong, D. Xie, and S. Pu, "Co-occurrence feature learning from skeleton data for action recognition and detection with hierarchical aggregation," *arXiv preprint arXiv:1804.06055*, 2018.

[3] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh, "Openpose: realtime multi-person 2d pose estimation using part affinity fields," *arXiv preprint arXiv:1812.08008*, 2018.

[4] A. Ratner, B. Hancock, J. Dunnmon, F. Sala, S. Pandey, and C. Ré, "Training complex models with multi-task weak supervision," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 4763–4771, 2019.

[5] A. J. Starling, "Diagnosis and management of headache in older adults," in *Mayo Clinic Proceedings*, vol. 93, pp. 252–262, Elsevier, 2018.

[6] J. Pascual and J. Berciano, "Experience in the diagnosis of headaches that start in elderly people.," *Journal of Neurology, Neurosurgery & Psychiatry*, vol. 57, no. 10, pp. 1255–1257, 1994.

[7] W. E. Hale, F. E. May, R. G. Marks, M. T. Moore, and R. B. Stewart, "Headache in the elderly: an evaluation of risk factors," *Headache: The Journal of Head and Face Pain*, vol. 27, no. 5, pp. 272–276, 1987.

[8] N. R. Cook, D. A. Evans, H. H. Funkenstein, P. A. Scherr, A. M. Ostfeld, J. O. Taylor, and C. H. Hennekens, "Correlates of headache in a population-based cohort of elderly," *Archives of Neurology*, vol. 46, no. 12, pp. 1338–1344, 1989.

[9] P. Torelli and G. C. Manzoni, "Behavior during cluster headache," *Current pain and headache reports*, vol. 9, no. 2, pp. 113–119, 2005.

[10] 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.

[11] L. Chunhui, H. Yueyu, L. Yanghao, S. Sijie, and L. Jiaying, "Pku-mmd: A large scale benchmark for continuous multi-modal human action understanding," *arXiv preprint arXiv:1703.07475*, 2017.

[12] J. Dong, Y. Gao, H. J. Lee, H. Zhou, Y. Yao, Z. Fang, and B. Huang, "Action recognition based on the fusion of graph convolutional networks with high order features," *Applied Sciences*, vol. 10, no. 4, p. 1482, 2020.

[13] J. Liu, A. Shahroudy, M. Perez, G. Wang, L.-Y. Duan, and A. C. Kot, "Ntu rgb+d 120: A large-scale benchmark for 3d human activity understanding," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[14] J.-K. Tsai, C.-C. Hsu, W.-Y. Wang, and S.-K. Huang, "Deep learning-based real-time multiple-person action recognition system," *Sensors*, vol. 20, no. 17, p. 4758, 2020.

[15] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6299–6308, 2017.

[16] Z. Weng, P. Varma, A. Masalov, J. Ota, and C. Ré, "Utilizing weak supervision to infer complex objects and situations in autonomous driving data," in *2019 IEEE*

Intelligent Vehicles Symposium (IV), pp. 119–125, IEEE, 2019.

[17] Y. Wang, S. Sohn, S. Liu, F. Shen, L. Wang, E. J. Atkinson, S. Amin, and H. Liu, "A clinical text classification paradigm using weak supervision and deep representation," *BMC medical informatics and decision making*, vol. 19, no. 1, pp. 1–13, 2019.

[18] J. A. Fries, P. Varma, V. S. Chen, K. Xiao, H. Tejeda, P. Saha, J. Dunnmon, H. Chubb, S. Maskatia, M. Fiterau, *et al.*, "Weakly supervised classification of aortic valve malformations using unlabeled cardiac mri sequences," *Nature communications*, vol. 10, no. 1, pp. 1–10, 2019.

[19] A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré, "Snorkel: Rapid training data creation with weak supervision," *The VLDB Journal*, vol. 29, no. 2, pp. 709–730, 2020.

[20] P.-L. Loh and M. J. Wainwright, "Structure estimation for discrete graphical models: Generalized covariance matrices and their inverses," *The Annals of Statistics*, pp. 3022–3049, 2013.

[21] A. Bulat and G. Tzimiropoulos, "How far are we from solving the 2d & 3d face alignment problem? (and a dataset of 230,000 3d facial landmarks)," in *International Conference on Computer Vision*, 2017.