Human Related-Health Actions Detection using Android Camera based on TensorFlow Object Detection API

Fadwa Al-Azzo[a, 1], Arwa Mohammed Taqi[a, 2], Mariofanna Milanova[b, 3]
[a]System Engineering Department, [b]Computer Science Department
University of Arkansas at Little Rock, Arkansas, USA

Abstract*—A new method to detect human health-related actions (HHRA) from a video sequence using an Android camera. The Android platform works not only to capture video images through its camera, but also to detect emergency actions. An application for HHRA is to help monitor unattended children, individuals with special needs or the elderly. The application has been investigating based on TensorFlow Object Detection Application Program Interface (API) technique with Android studio. This paper fundamentally focuses on the comparison, in terms of improving speed and detection accuracy. In this work, two promising new approaches for HHRA detection has been proposed: SSD Mobilenet and Faster RCNN Resnet models. The proposed approaches are evaluated on the NTU RGB+D dataset, which it knows as the present greatest publicly accessible 3D action recognition dataset. The dataset has been split into training and testing dataset. The total confidence scores detection quality (total mAP) for all the actions classes are 95.8% based on the SSD-Mobilenet model and 93.8% based on Faster-R-CNN-Resnet model. The detection process is achieved using two methods to evaluate the detection performance using Android camera (Galaxy S6) and using TensorFlow Object Detection Notebook in terms of accuracy and detection speed. Experimental results have demonstrated valuable improvements in terms of detection accuracy and efficiency for human health-related actions identification. The experiments have executed on Ubuntu 16.04LTS GTX1070 @ 2.80GHZ x8 system.

***Keywords*—Android camera; TensorFlow object detection API; emergency actions; detection accuracy**

I. Introduction

Generally, the elderly, children and the people with special needs are considered to be in need of care and supervision of their behavior all the time. Safety is a priority especially when their caregivers are not available to prevent accidents like falling, passing out or nausea due to an existing medical conditions or unforeseen dangers to prevent such accidents as in falling, nausea This led us to build an application to detect the abnormal actions through the Android platform and in the future, it can be manipulated to send a alerting message to the observer to provide the necessary assistance as soon as possible. The Object detection is a common concept for computer vision methods for definition and labeling objects. Human action detection is one of the top massively studied topics that can be used for surveillance camera. Most of the techniques include substantial limitations when it comes to the specific form of computational resources, the dependence of the motion of the objects, disability to differentiate one object from another, the absence of proper data analysis of the measured trained data, and a major interest is over the speed of the movement and illumination. Therefore, drafting, applying and recognizing new methods of detection that manage the present limitations, are much needed. The techniques of object detection can be applied both to still images or video images. The objective of human action detection is to detect each appearance of a specific action in a video and to localize every detection identified together in space and time. Recently, the video action recognition performance has improved based on Deep Learning (DL) approaches. Human action detection task is challenging compared to the human action recognition because of the change in size of the human and in addition to the spatio-temporal location. Generally, deep learning has been connected with data centers and large clusters of great-powered GPU machines. Nonetheless, it can be highly expensive and time-consuming to transfer all of the data in the device and deliver it across a network connection. Implementation on a mobile makes it possible to deliver real interacting applications using a method that is not possible when you should wait for a network round trip. The smartphone is being added to video surveillance systems almost everywhere at any time. In other words, the video surveillance system for mobile has spread and expanded significantly in recent times so you can monitor your home or business when you are away. Smartphone devices with a camera are a massive part of our daily lives. However, there is a growing interest in the interaction between the users and their devices. The interaction between the user, the phone, and real-world objects [1] represents the many variations of smart device applications.

Objects detection utilizing a mobile camera has many functions like video surveillance, object stability, and collision revocation.Most techniques have been used to detect tracking objects from a non-stable platform. Whereas these techniques require the movement parameters of the camera to be recognized, that is frequently not easily obtainable or are incapable of object detection if the tracking object’s size is small. In this paper, we aim to present a new detection application for video images of human health-related actions using Android phone’s camera. The software is based on Deep learning system running on TensorFlow’s Object Detection API using Android platform. A robust tool gives it straight to construct, train, and use object detection models. In most of the cases, training a complete convolutional network from scratch is time-consuming and needs massive datasets. Accordingly, this problem can be resolved by utilizing the power of transfer learning with a pre-trained model [2] using the TensorFlow API.

An Android platform has been used toward the object recognition applications [1], which works on images taken through a built-in camera. Android is gradually growing to be the commonly utilized platform amongst the smartphone technologies as a monitor. The user can have access to the correct human action from the object by which the required action is marked upon the detected object. The recognition of the object detection in a video frames on an android is completely developed. Once the object is appropriately detected, it could be saved and accessible for future applications.

The contributions of this study are as follows.

- Videos of human actions are selected based on health needs and it defined emergency actions that must be taken consideration because it is important to detect them quickly and with high accuracy.
- Most advanced methods have been utilized for training the various human action classes in order to guarantee that it learns all the samples of the entire video frames.
- Model performance has been evaluation through detection accuracy by measuring the confidence source mAP for each action class and total mAP for all the classes.
- The number of steps of the training and evaluation processes can be controlled to get the least error (classification loss) to make the classification of the action class correct and also make the location of the human action as the lowest error (localization loss) to bounding detection box on the right place.
- By using the Android camera, the efficiency and power of the model detector can be enhanced. That is because; the Android detection results were compared to the detection process using TensorFlow Object Detection Notebook technique.

The paper is organized as follows. Section II discusses the related works. Details of object detection technique is presented in Section III. Section IV explains TensorFlow object detection API technique. The Feature extraction algorithm defined with details in Section V. Section VII presents the proposed technique that includes the experimental settings, training and evaluation processes and detection approach with experimental results. Discussion of the significance of this work is discussed in Section IX. Finally, Section X concludes this work.

II. Related Works

Human action detection using image-processing methods on smartphones as Android device is a developed conception. Wherefore only, a meager published literature was usable. Most of the obtainable literature focuses on the desktop applications. The topic studied in [3] showcases a collection of critically trained elements in a star model that is called Pictorial Structure. It can be said that this paper could be viewed as a 2-layer model, consisting of the first layer and the second layer will include the star model. The research in [4] utilizes the common field information which was related to [5] which was based on the manually designed Histogram of Gradients (HOG) descriptors. [6] represents the objectness measure by simply utilizing 8 × 8 binarized normed gradients (BING) features. This process is easy and quick and it is done by calculating the objectness of each image boundary box at any scale by using a few atomic operations. The paper [7] presents a design for bidirectional retrieval of images and sentences by using a deep, multi-modal embedding of visual and natural language data. In addition, they also display a structured max-margin objective, which enables their model to incorporate these parts over modalities. A design that creates natural language descriptions of images and their regions is shown in research [8] while approaching supported datasets that include images and their sentence information in order to learn about the inter-modal correlation between language and visual images. Their model is set up on an aggregate of Convolutional Neural Networks over image regions and bidirectional Recurrent Neural Networks over sentences. The work in [9] examines the conclusion of disfiguring part models from 2D images to 3D spatio-temporal volumes in order to study their efficiency for action detection in video.

Action detection model presented in [10] , first decomposes human actions towards temporal key poses and then to spatial action parts. Precisely, they began by clustering cuboids around every human joint to dynamic-pose lets by using a new descriptor. The structure of paper [11] merges strong computer vision techniques for forming bottom-up region with modern improvements in learning high performance of convolutional neural networks. The resulting system based on R-CNN is called: Regions with CNN features.

III. Object Detection Technique

Generally speaking, the object detection methods apply an image classifier to an object detection function which is becoming efficient methods; it involves changing the size and a position of the object in the test image and then utilizing the classifier to recognize the object. The multi- bounding box method [12] is a familiar model that has been reported in. Over the past few years, a method covering the extraction of several filter regions of objects utilizing region proposals as performed by R-CNN. Thereafter, the process of making classification decision [13] with filter regions utilizing classifiers has been described. However, the R-CNN method could be time consuming because it needs a larger amount of crops, which will result to duplicate the calculation from overlapping crops. Such calculation verbosity was resolved by the use of a Fast R-CNN [14] that inserts the completed image once within a feature extractor so the crops would end up sharing the calculation amount of feature extraction.

In this work has used the developed software tool for the Android camera; it focuses on two recent TensorFlow object detection API models: SSD_Mobilenet framework, and the Faster R-CNN-Resent framework. The algorithm proposed for HHRA detection model is important to understand how efficient the framework performs.

IV. TENSORFLOW OBJECT DETECTION API

TensorFlow Object Detection API's package is a process to resolve object detection problems. This is a technique of detecting real-time objects in an image. In agreement with the documentation and the paper [2] that shows the library, what makes it exclusive is that it is capable to trading accuracy for speed and memory application (vice-versa). Therefore, you can modify the model to satisfy your requirements and your platform, like a smartphone. The Tensorflow Object Detection API library contains multiple out-of-the-boxes object detection structures like SSD (Single Shot Detector), Faster R-CNN (Faster Region-based Convolutional Neural Network), and R-FCN (Region-based Fully Convolutional Networks).

In addition to different feature extractors such as MobileNet, Inception, and Resnet; those extractors are actually important since they represent a major part in the speed/achievement trade-off of the framework. In fact, training from scratch to cover an entire convolutional network is time consuming and requires hugs datasets. To avoid this problem, transfer learning is applied with a pre-trained model relating the TensorFlow API. The transfer learning [15] is a machine learning approach where a model advanced for a function is reused as the outset point for a model on a second function. It is a common strategy in deep learning where pre-trained models are adopted as the starting point on computer vision processing function given the large calculation and time resources wanted to develop neural network models. The benefit of utilizing a pre-trained model is that alternatively of creating the model from scratch, a model trained for a similar problem can be applied as a starting point for training the system. In this work, the experiments have used the pre-trained model/checkpoints SSD MobileNet and Faster R-CNN- Resent [16] from the TensorFlow Zoo.

A. Faster R-CNN

A Faster R-CNN network [12] uses as input a whole image and a group of object proposals. Accordingly, the first processes the whole image including some convolutional (conv.) and max pooling layers to generate a conv. feature map. Next, for every object proposal, a region of interest (RoI) extracts a fixed-length feature vector of the feature map. Every feature vector is supplied to a concatenation of fully connected (fc) layers, which eventually branch for two relationship output layers: one that creates softmax probability rating through N object categories and addition to taking all "background" category plus adding a layer that produces outputs four real-valued numbers for each of the N object categories. Each collection of 4 values encodes filtered bounding-box positions for one of the N categories. The RoI pooling layer utilizes max pooling to change the features under each actual region of interest in a slight feature map by a fixed spatial range of $H \times W$, where H and W denote a layer hyper-parameters. A RoI in [12] is a rectangular window in a conv. feature map. Every RoI is determined over a four-variable (r, c, h, w) where it defines its top-left corner (r, c) and its height and width (h, w). The RoI max-pooling operates through dividing the $h \times w$ RoI window into an $H \times W$ grid of sub-windows of estimated size ($h/H \times w/W$). Thereafter the max-pooling values in every sub-window within the corresponding output grid cell. Faster R-CNN has two processes for object detection. First process, images are processed utilizing a feature extractor model (e.g., MobileNet, VGG,) named the Region Proposal Network (RPN), and at the same time several medium-level layers are applied to expect the category bounding box proposals. Second process, the box proposals are utilized to crop features from the same medium feature map, which are then inputted to the rest of the feature extractor model with a view for predicting a category label and its bounding box will improve for every proposal. Lastly, the Faster R-CNN does not crop proposals straight from the image; instead, it runs the crops again over the feature extractor, which will drive to more replicated computations.

B. SSD

A Single Shot Multibox Detector (SSD) [17] was presented in 2016 by researchers from Google. The SSD is a rapid single-shot object detector for multiple classes. It utilizes a one feed-forward convolutional network to assume classes straightforward and anchor stabilizer without needing another step for each proposal classification process. The important feature of SSD is the employ of multi-scale convolutional bounding box outputs connected to various feature maps at the highest of the network. The VGG-16 was applied as the core network because of its effective performance in high-quality image classification functions and transfer-learning training in order to enhance the results. The bounding box technique of SSD is driven by Szegedy's project [18] on MultiBox, and an Inception mode convolutional network is used. The MultiBox's loss task mixed two significant parts that performed their path to SSD. The first part is confidence loss that measures how confident the system is of the objects of the calculated bounding box. However, the second part is location loss, which measures the distance of the network's predicted bounding boxes from the ground truth ones through the training process.

The SSD applies smooth L1-Norm [19] to determine the location loss. Regarding classification process, the SSD performs object classification. Therefore, for every predicted bounding box, collections of N categories predictions are calculated for each likely category in the dataset. Furthermore, feature maps are a description of the interest features of the image at various scales, hence working MultiBox on multiple feature maps rises the likelihood of any object whether large or small which to be eventually detected, localized and properly classified.

V. FEATURE EXTRACTION

The aim of feature extraction is to decrease a variably sized image to a packed set of visual features. Typically, image classification models are built by applying strong visual feature extraction techniques. While they depend on either conventional computer vision approaches, (e.g. filter based methods, histogram techniques, etc.) or on deep learning approaches, they all have the same goal: extract features from the input of image which are appropriate for the task, and apply these features in a classification process to define the class of the image. In object detection systems, a convolutional feature extractor is a base network that is implemented in the input data to obtain advanced features. The collection of the feature extractor is supposed to be highly significant, which is because of the number and types of layers, the number of parameters and other characteristics, which immediately influence the execution of the detector. In this paper, two feature extractors which have been selected are the most used in the area of computer vision.

A. *Resnet*

Deep learning networks have been developed and moved to a high level of sophistication in detection applications when a Microsoft Research released for a Deep residual networks (Resnets) [20]. The Resnets that are above 100-layer deep have demonstrated state-of-the-art accuracy for challenging recognition functions on ImageNet [21] and MS COCO [22] competitions that included object detection, image classification, and semantic segmentation. The validity of Resnets has been confirmed by multiple visual recognition applications and by non-visual applications including speech and language. Deep Network is of significant interest in neural network architectures, however, deeper networks are further challenging to train. The residual learning structure helps the training of these networks and allows them to be deeper activate performance in both visual and non-visual functions. In the deeper network, the additional layers much better approximate the mapping than its conventional counterpart and decrease the error.

The residual function utilize $F(x) = H(x) - x$, that can be reorganized into $H(x) = F(x) + x$, where $F(x)$ and x performs the accumulated non-linear layers and the identity function (input=output) respectively. Moreover, the main concept of Resnet is to inserting a termed “identity shortcut connection” that jumps one or more layers. Generally, Resnets involves several stacked “Residual Units”, where each unit can be represented in a common form:

$$y_l = h(x_l) + F(x_l, W_l), \quad (1)$$

$$x_{l+1} = f(y_l), \quad (2)$$

Where x_l and x_{l+1} are input and output of the l-th unit, and F represents a residual function. In [20] h(x_l) = x_l represents an identity mapping and f is a ReLU [23] function. The basic concept of Resnets is to learn the collective residual function F with regard to $h(x_l)$, with an important option of applying an identity mapping $h(x_l) = x_l$, this is achieved by linking an identity overstep connection (“shortcut”).

B. *Mobilenet*

Mobilenet [24] was created for active inference in different mobile vision functions. The network structure of Mobilenet depends on depthwise separable convolution. It is an advanced state of the inception module, wherever parted spatial convolution for each channel is used which indicated as depthwise convolutions [25]. The 1x1 convolution with all the channels to combines the output indicated as pointwise convolutions are applied. As a result, the division in depthwise and pointwise convolution increase the efficiency performance furthermore, it enhances the accuracy, while a cross-channel and spatial correlations mapping is learned independently. The MobileNet has been displaying to reach an accuracy identical to VGG-16 on ImageNet with exclusive 1/30th of the calculation cost and model magnitude. Its structuring blocks set are depthwise separable convolutions which factorize a standard convolution for a depthwise convolution and a 1 × 1 convolution as shown in Fig.1ultimately, decrease the pair of the calculation cost and a number of hyper-parameters.

VI. TENSORFLOW IN ANDRIOD

The Android model of the TensorFlow library [26] is a single project that builds and installs four sample applications (TF Detect, TF Classify, TF Stylize, and TF Speech) [27], which all use the same underlying code. The sample applications all take input video from a phone's camera. The TF Detect app is going to be used in this work. The base of the TensorFlow is written in C++ to begin towards building the process for the Android. In order to establish for Android, JNI (Java Native Interface) has to be applied to call the C++ functions such as loadModel, obtain predictions, etc. A shared object (.so) file will be built, that is a C++ compiled file and a jar file that will involve JAVA API, which calls for the native C++, and the JAVA API will be called to make things achieved simply. There are software, dependencies, and packages required: Android Studio, Android SDK and Android NDK. Android Studio import a new project using the directory from the TensorFlow repo called “Android”.

Essentially, NDK (Native Development Kit) [28] is a great tool in the evolution of mobile applications. Particularly if you need to improve a multiplatform application, the NDK is excelled in this field. Since the same code written in C++ for Android can be ported and run the same way on the iOS, Windows or any other platform without changing the original code. This really keep a lot of time in the development of applications, which are advanced for doing run on multiple platforms; as games and other traditional applications.

SDK (software development kit) [29] is a tool with more tool applications, data files, and model code. The SDK supports you in developing code, which uses a special system such as extension code for utilizing features of an operating system (Windows SDK), drawing 3D graphics by a specific system (DirectX SDK), or writing a code to make a device like a mobile phone perform what you need.

VII. PROPOSED TECHNIQUE

In this paper, the proposed model aim to detect human health-related actions from videos of N frames by using Android camera. The dataset splits into training and testing dataset. First step is to label the dataset by drawing a bounding box (ground truth) around each video frame of human health-related action. Then, save them as XML file. The XML file converts to a CSV file and thereafter, TensorFlow that is called “TFRecords” converts the CSV file into a format that is readable. The proposed HHRA model uses two per-trained models of TensorFlow object detection API that are Faster R-CNN-Resnet and SSD-Mobilenet for training the dataset. Model evaluation will be done during and after training. Lastly, it will demonstrate how to export the model to Android for detect the action type using Android camera. Under the object detection algorithm utilizing deep learning, there are many parameters that are learned from the data. Fig.1 describes the basic block diagram for proposed the HHRA model in this paper. The setting of TensorFlow object detection API of the models that used in this work as following as

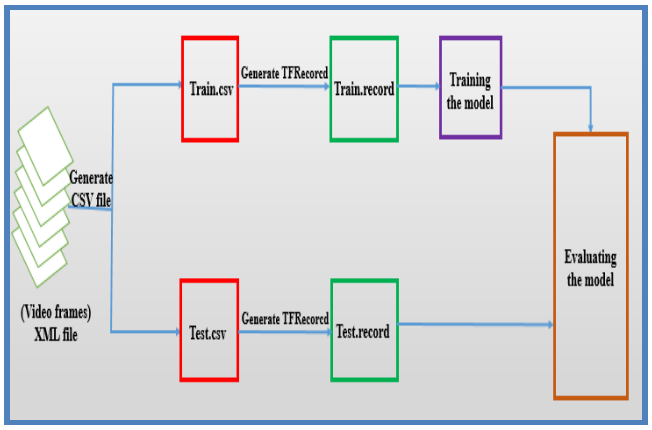


Fig. 1. Basic Block Diagram for Proposed the HHRA Model.

A. Faster R-CNN-Resnet

The methodology mentioned in the [12] [20] is used in this work. It has been using the weights by an abbreviate normal distribution with a standard deviation of 0.01. The intial_learning_ rate is 0.003 and a momentum _optimizer value is 0.9 with batch size equals 1. A loss function procedures the softmax function for classification loss and the smooth L1 function for localization loss. A function is used as the activation function. The input for each video frame was resized to 606 × 1024 pixels.

B. SSD-Mobilenet

Following the methodology mentioned in the papers [17] [24] it has been initializing the weights by an abbreviate normal distribution with a standard deviation of 0.03. The intial_learning_ rate is 0.003 with a learning_ rate_ decay of 0.9997 and a momentum _optimizer_value is 0.9 with batch size equals 32. A loss function uses the sigmoid function for classification loss and the smooth L1 function for localization loss. A ReLU function is used as the activation function. The input for each video frame was resized in this framework to 300 × 300 pixels.

VIII. Experimental Results and Analysis

A. Dataset and Preprocessing

The proposed detection method are evaluated on the NTU RGB+D dataset [30], which is defined as the present biggest widely obtainable 3D action recognition dataset. The dataset includes more than 56k action videos and contents 4 million frames. In this paper, seven different human health-related actions of video sequences were selected from NTU RGB+D dataset (Falling, Nausea, Headache, Neck pain, Sneezing, Staggering, and Stomachache) presented as frames sequences for training using TensorFolw object detection models and testing using Android's camera. The training dataset was created by manually tagging the human action in the video frames using LabelImg software [31] because it is needed to have a ground truth of what exactly the object is. In other words, it is important to draw a bounding box around the person with his action as shown in Fig.2, so the system knows that this "action" inside the box is the actual human action. Each person with his action was tagged as a name of the human activities related to the type of action (a health-related action). LabelImg saves the annotations as XML-files in PASCAL VOC format is prepared for creating TFRcords (Tensor Flow record format). Each dataset requires a label map connected with it, which represents a mapping from string class names to integer class IDs. Label maps should always start from ID1. In this work, there are seven IDs related to seven human health-related actions. The 1920x1080 sized video frame (and the corresponding annotation files) were later resized to improve the model training efficiency. Once all the frames were labelled, the next step was to split the dataset into a train and test the dataset.

B. Training and Evaluation

In this work, the pre-trained with one of the models (SSD_Mobilenet or Faster R-CNN-Resnet) was fine-tuned for NTU RGB+D dataset using manually labeled video frames of HHRA saved this tagged data as an XML file and adapted this XML file to a CSV file. Next, the CSV file is converted to TFRecord file by satisfying the similar specifications as shown in Fig.1. The entire training process is addressed by a configuration file recognized as the "pipeline". The pipeline is split into several essential structures that are responsible for determining the model, the training and evaluation process parameters, and both the training and evaluation dataset inputs. Actually, the TensorFlow advises that the training should apply one of their own and already trained models as an outset point. The idea behind this is that training a completely new model from scratch might need an excessive amount of time. Therefore, the TensorFlow gives various configuration files, which only needs a number of changes that correspond to a new training environment. The results of the training and the evaluation stages can be observed by applying TensorFlow's visualization platform, TensorBoard [32]. This tool can observe various metrics such as the training time, total loss, number of steps and much more. The TensorBoard also runs while the model is being trained, making this an excellent tool to confirm that the training is going in the right direction. The given checkpoint file for the models is applied as a beginning point for the fine-tuning process.

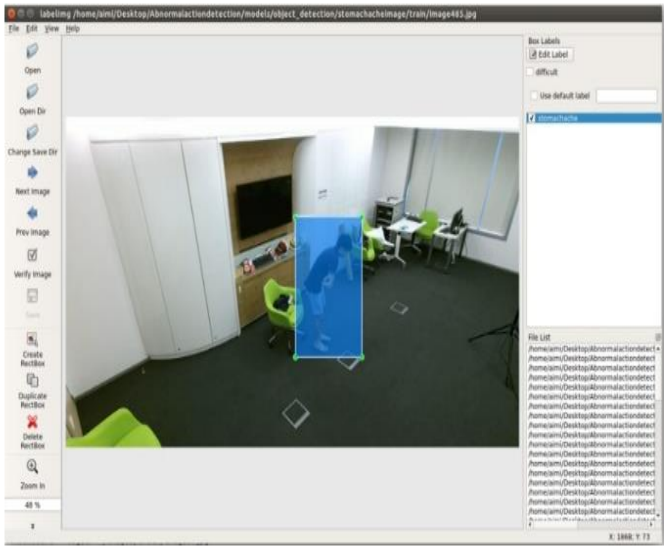


Fig. 2. Sample of Stomachache Action Frame Surrounded by a Bounding Box.

The changes are made so that the variable num_classes, num_steps to pause the training earlier, fine_tune_checkpoint to point to the location of the model downloaded, and the input_path and label_map_path variables of the train_input_reader and eval_input_reader to point to the training and testing dataset, as well as the labels map. Primarily, training data as well as evaluation data are needed during the training process. The training data is necessary to learn the model on these data, while the evaluation data is required to evaluate the accuracy of the trained model if the model has learned all the video frames for each class. Region proposals are clustering-based method, which tries to group pixels and produce proposals based on the created clusters.

In the evaluation stage, the mAP measures the trained model percentage of correct predictions for all seven actions labels. The IoU is particular to object detection models and a case for Intersection-over-Union. This measurement represents the overlap between the bounding box generated by HHRA model and the ground truth-bounding box, described as a percentage. The mAP graph is averaging the percentage of correct bounding boxes and labels of the HHRA model resumed with “correct” in this case relating to bounding boxes that had 50% or more overlap with their corresponding ground truth boxes.

The HHAR model is improved for loss functions which combining two functions (classification and localization):

u: True class label, $u \epsilon$ 0,1,. . . ., K; by convention, , the catch all background class has u=0.

p: Discrete probability distribution (per RoI) over k+1 classes:

$p = (p_0, \ldots, p_k)$, computed by a softmax over the k+1 outputs of a fully connected layer.

v: True bounding box $v = (v_x, v_y, v_w, v_h)$.

t^u: Predicated bounding box correction, $t^u = (t_x^u, t_y^u, t_w^u, t_h^u)$.

The loss function sums up the rate of classification and bounding box prediction: $\mathcal{L} = \mathcal{L}_{cls} + \mathcal{L}_{box}$. For “background” RoI, $\mathcal{L}_{box}$ is avoided by the indicator function $1[u \geq 1]$, describe as:

$$1[u \geq 1] = \begin{cases} 1 & if\ u \geq 1 \\ 0 & otherwise \end{cases} \quad (3)$$

The overall loss function is:

$$\mathcal{L}(p, u, t^u, v) = \mathcal{L}_{cls}(p, u) + 1[u \geq 1]\, \mathcal{L}_{box}(t^u, v) \quad (4)$$

$$\mathcal{L}_{cls}(p, u) = -\log p_u \quad (5)$$

$$\mathcal{L}_{box}(t^u, v) = \sum_{i \epsilon \{x,y,w,h\}} L_1^{smooth}(t_i^u - v_i) \quad (6)$$

The bounding box loss L_{box} measure the difference between t_i^u and v_i applying a robust loss function. The smooth L_1 loss[8] is implemented here and it is supposed to be less sensitive to outliers.

$$L_1^{smooth}(x) = \begin{cases} 0.5x^2 & if\ |x| < 1 \\ |x| - 0.5 & otherwise \end{cases} \quad (7)$$

C. Evaluation the Model

To evaluate the model during the training and after it, each time the training produces a new checkpoint, the evaluation tool will perform predictions using the video frames available in a given directory.

Evaluation metrics: the greatest significant evaluation metrics for such implementation are precision, recall, F1-score and mAP. Precision represents how applicable detection results are (Eq.8):

$$Precision = \frac{TP}{(TP+FP)} \quad (8)$$

Where TP = true positive, FP = false positive.

Recall: represents the percentage of objects, which are detected including the detector. (Eq. 9):

$$Recall = \frac{TP}{(TP+FN)} \quad (9)$$

Where FN = false negative.

It is important to mention that which the there is an opposite correlation between precision and recall and which these metrics are dependent on the model score threshold [33].

In the light of this, the model detector has the power to detect a big percentage of objects in an image; however, it as well produces a large number of false positives. While the model detector by a big threshold for detection only generates a small false positive, but, it likewise quits a greater percentage of objects, which remain undetected. The best equivalence between these two depends on the applicability.

D. Localization and Intersection over Union

Intersection over Union (IoU) is an evaluation metric applied to estimate the accuracy of an object detector on a relevant database as shown in Fig.3. Concerning to evaluate database as shown in Fig.3. Concerning to evaluate the model on the function of object localization, it must determine how strong the model predicted the location of the object as shown Fig.4. Ordinarily, this is accomplished including drawing a bounding box around the object of interest. The localization function is normally evaluated on the Intersection over Union threshold (IoU).

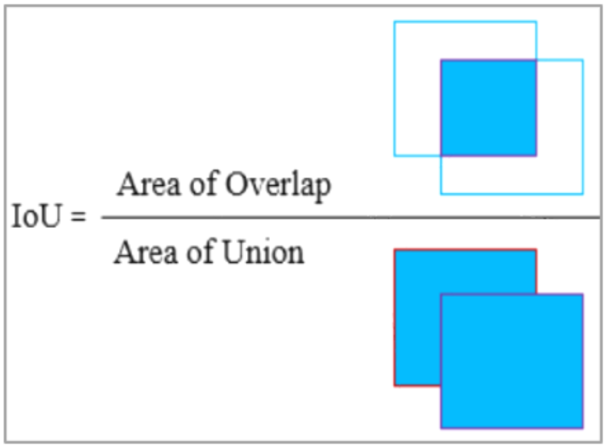


Fig. 3. Calculate IoU between Ground-Truth and Predicted Bounding Boxes.

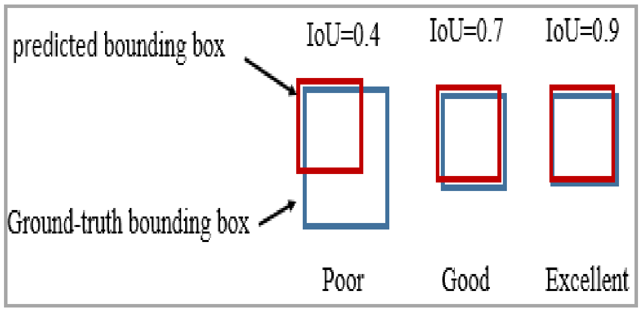


Fig. 4. An Instance of Calculating IoU for Different Bounding Boxes.

The TensorFlow Object Detection API usages “PASCAL VOC 2007 metrics” [34] location an estimated instance is described as a TP while Intersection over Union (IoU) is above 50% [33] (Eq.10):

$$IoU = \frac{area(groundtruth\ bounding\ box \cap predicted\ bounding\ box)}{area(ground\ truth\ bounding\ box \cup predicted\ bounding\ box)} > 0.5 \quad (10)$$

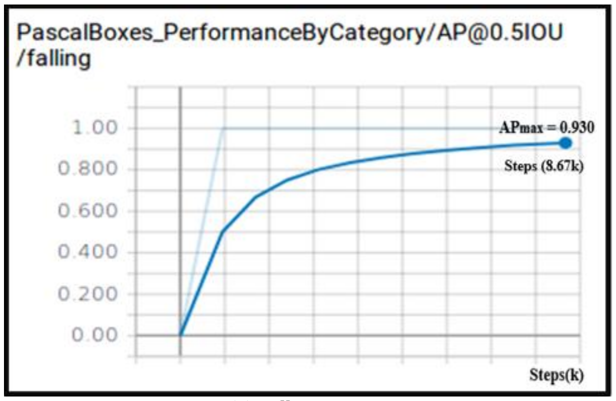
One object can be associated with only one bounding box; however, if some bounding boxes are predicted for an object, one is regarded as TP and the others FP. However, if an object is without a predicted bounding box, which is associated with it, then it is recognized as an FN.

E. Mean Average Precision (mAP)

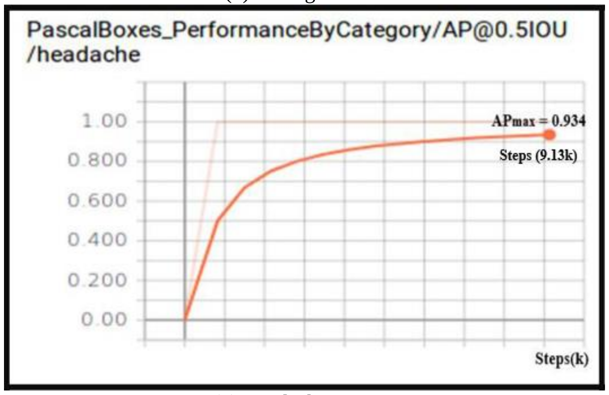
The mAP is the result of precision and recall “precision-recall” on detecting bounding boxes. It is a satisfactory way for a measuring of how the network to understands the objects of importance and how it evades invalid information. The greater the score of the mAP, the more accurate is the network. Generally, the mAP enhance the information in “precision-recall” curve a single number. Thereafter, for each prediction, a recall rate and a precision rate is estimated. The average precision (AP) is the average of class predictions measured over various thresholds, in PASCAL metrics, the thresholds are from scale [0, 0.1, . . ., 1], i.e., its average of precision values for various recall levels. It is an effort to achieve characteristics of the detector in a single number. The AP is defined as the region under the precision-recall curve. In this work, the AP curves for each human health-related action class based on both the per-trained models (Faster R-CNN-Resnet and SSD-Mobilenet) (see Figs.5-6)). A vertical axis represents the AP values while a horizontal axis represents the steps (epochs). Table1 lists the results for maximum AP values at 0.5IoU with last number of steps being each human health-related action.

TABLE I. MAXIMUM AP VALUES AT 0.5IOU WITH THE NUMBER OF STEPS (K=1000) FOR EACH HUMAN HEALTH-RELATED ACTION BASED ON FASTER R-CNN-RESNET AND SSD-MOBILENET MODELS

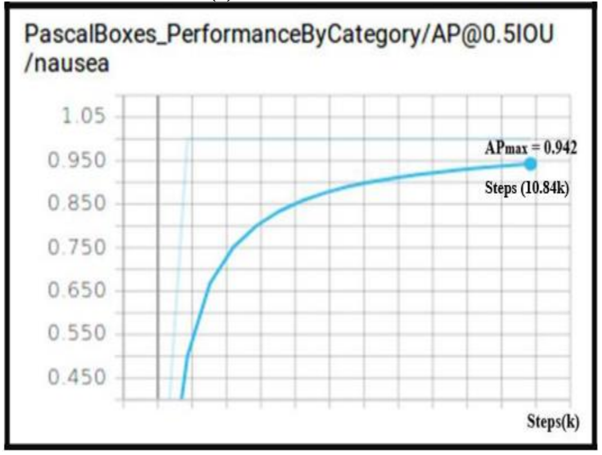
	Faster-R-CNN-Resnet		**SSD-Mobilenet**	
	AP@ 0.5 IoU	**Steps(k)**	**AP@ 0.5 IoU**	**Steps(k)**
falling	**0.930**	**8.67**	**0.975**	**47.66**
headache	**0.934**	**9.13**	**0.917**	**50.65**
nausea	**0.942**	**10.84**	**0.979**	**55.68**
neck pain	**0.934**	**10.30**	**0.967**	**58.34**
sneeze	**0.938**	**10.20**	**0.973**	**44.74**
staggering	**0.934**	**11.04**	**0.959**	**52.31**
stomachache	**0.953**	**15.07**	**0.965**	**41.09**



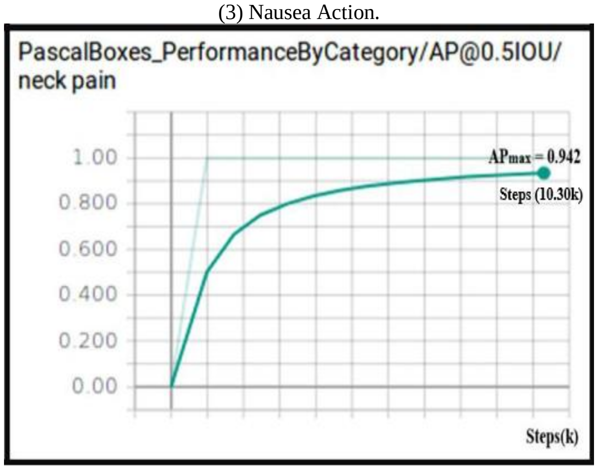
(1) Falling Action.



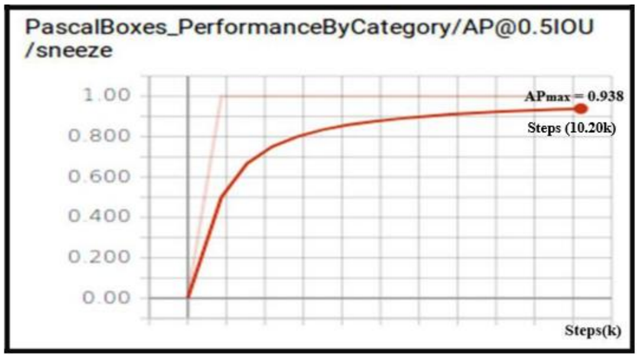
(2) Headache Action.



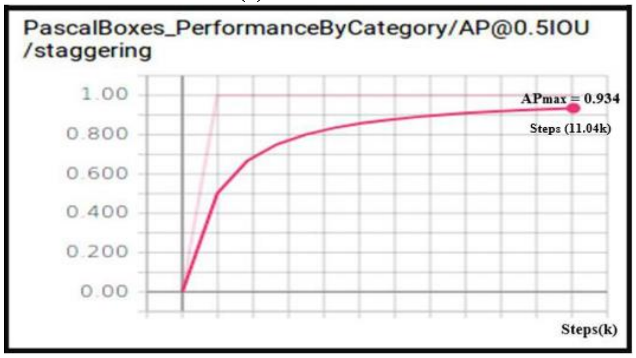
(3) Nausea Action.



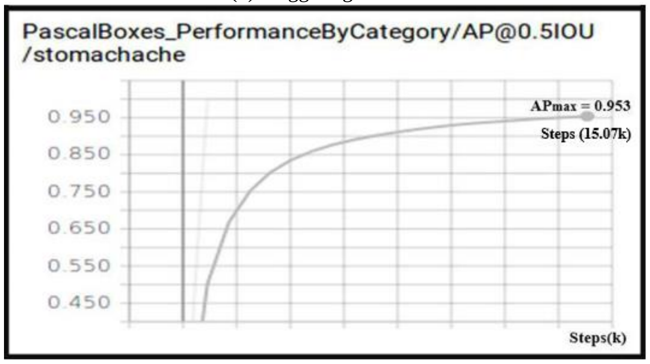
(4) Neck Pain Action.



(5) Sneeze Action.



(6) Staggering Action.

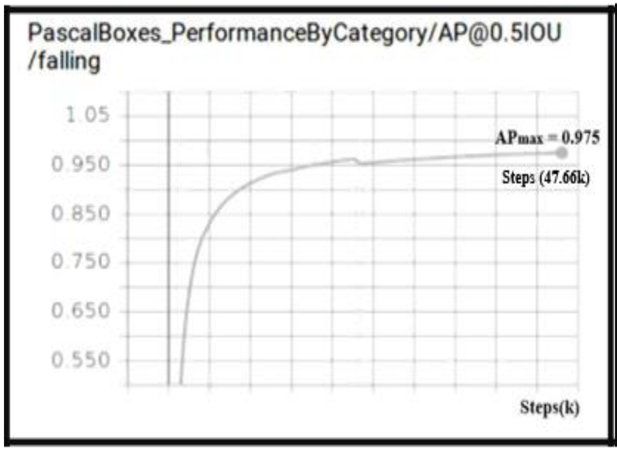


(7) Stomachache Action

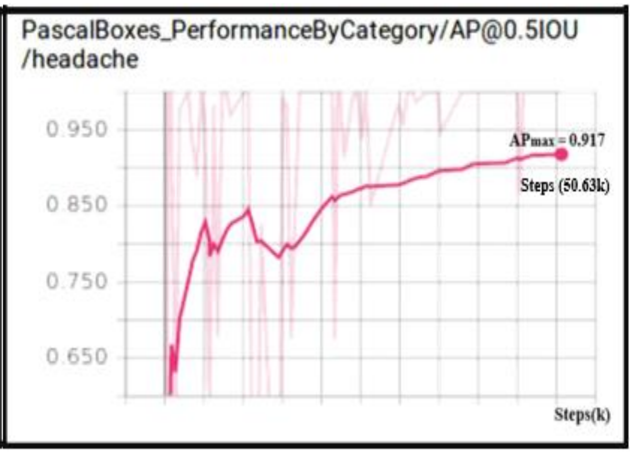
Fig. 5. AP Curves for Each Human Health-Related Action. Maximum AP Values at 0.5IoU with Last Number of Steps (k=1000) are based on Faster R-CNN-Resnet Model.

F. The Results

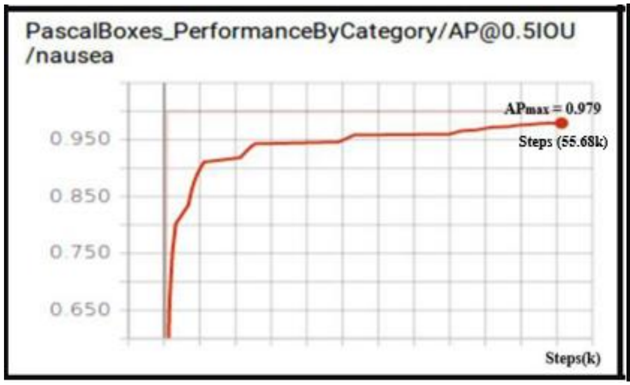
As the training process progresses, the expectation is to reduce total loss (errors) to its possible minimum (about a value of 1 or less). By observing the tensorboard graphs for total loss for Faster R-CNN-Resent and SSD-Mobilenet models (see Fig.6), it should be possible to get an idea of when the training process is complete (total loss decrease with further iterations/steps(epochs)). The parameter num_steps defines how many training steps they will run before stopping. This number actually depends on the size of the dataset along with how long the user wants to perform the training of the model. The utilized metric for achievement is a mean average precision (mAP) which is a single number used to summarize the area under the precision-recall curve. The mAP is a measure of how well the model generates a bounding box that has at least a 50% overlap with the ground truth bounding box in the test dataset. The mAP value reached to higher confidence at 0.5IoU (see Fig.7) for each action class of both pre-trained models. The higher the mAP values the higher the detection accuracy (the higher the better). However, the SSD-Mobilenet takes a long time to reach the high mAP value compared to mAP's Faster-R-CNN-Resnet time. The classification loss curve in Fig.8 indicates the validation of human action class which is classified and matched with the previous trained class. As the values of the classification loss decrease to zero, it shows that the classification accuracy is high and the efficiency of the detector performance becomes more advanced. The Fig.9 (a,b) displays the results for both models where there are seven classification loss curves corresponding to each human action health-related class. Generally, all actions have consistent decreasing classification loss values, which give the power of the model performance in the classification of each video frame of the similar class type. As for the localization loss curve in Fig.10, it describes the predicate bounding box that matches with the ground-truth bounding box. As the loss value is reduced, less error will be present in the action detection and the Intersection over Union will be high, which all indicates that the detection of the action is in the right direction. The smoothed L1 loss is used for localization and is weighted.



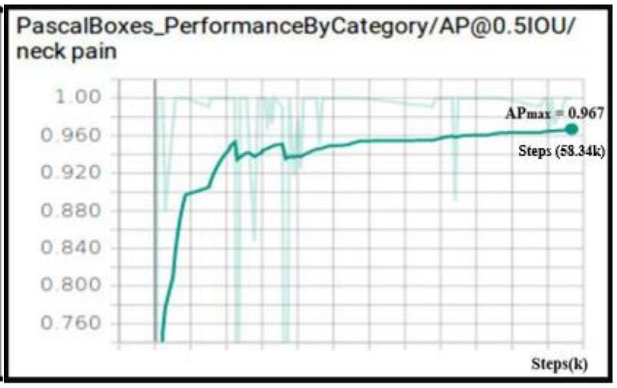
(1) Falling Action.



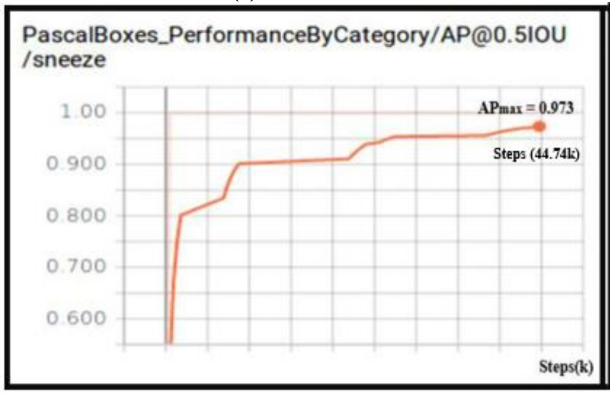
(2) Headache Action.



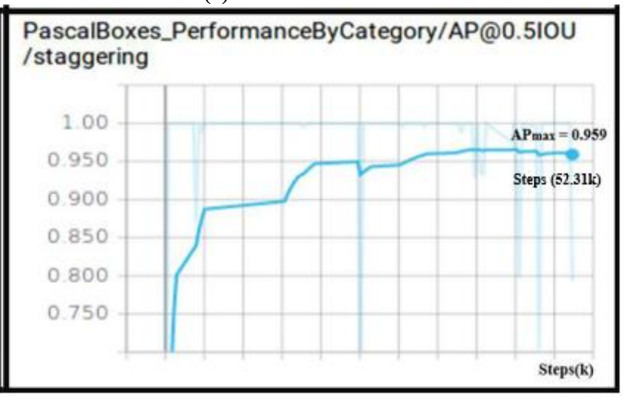
(3) Nausea Action.



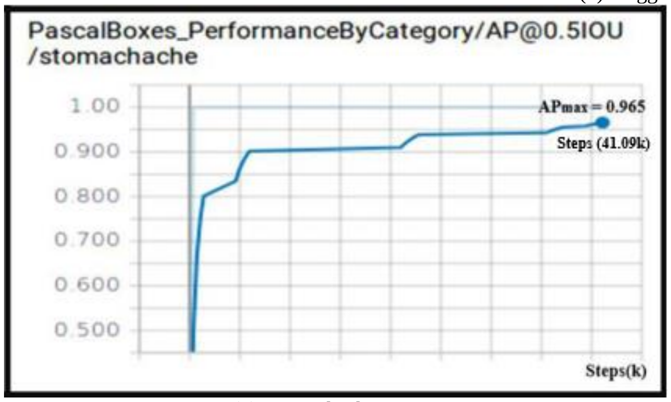
(4) Neck Pain Action.



(5) Sneeze Action.

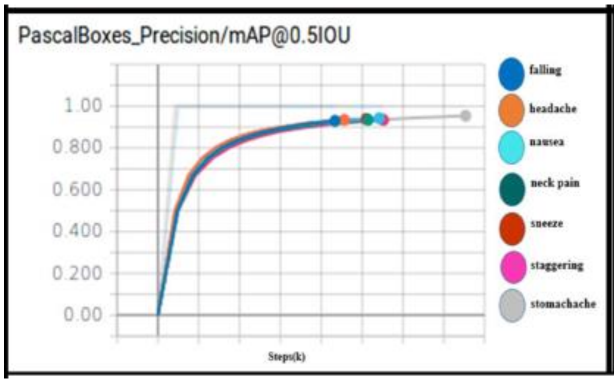


(6) Staggering Action.



(7) Stomachache Action

Fig. 6. AP Curves for each Human Health-Related Action. Maximum AP Values at 0.5IoU with Last Number of Steps (k=1000) are based on SSD-Mobilenet Model.



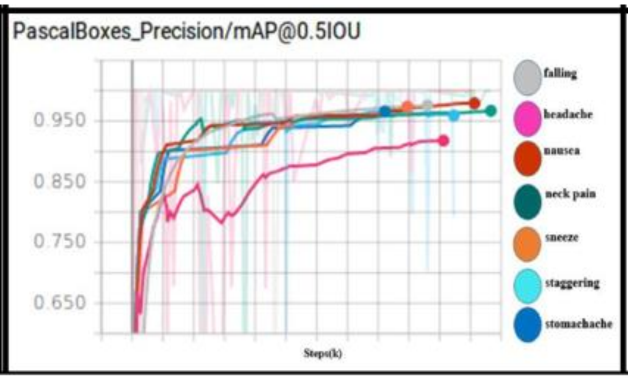


Fig. 7. Maximum mAPs @0.5IoU for Seven Human Heath Related Actions Classes are based on Two Per-Trained Models (the left (Faster R-CNN-Resnet),the Right (SSD-Mobilenet)) respectively.

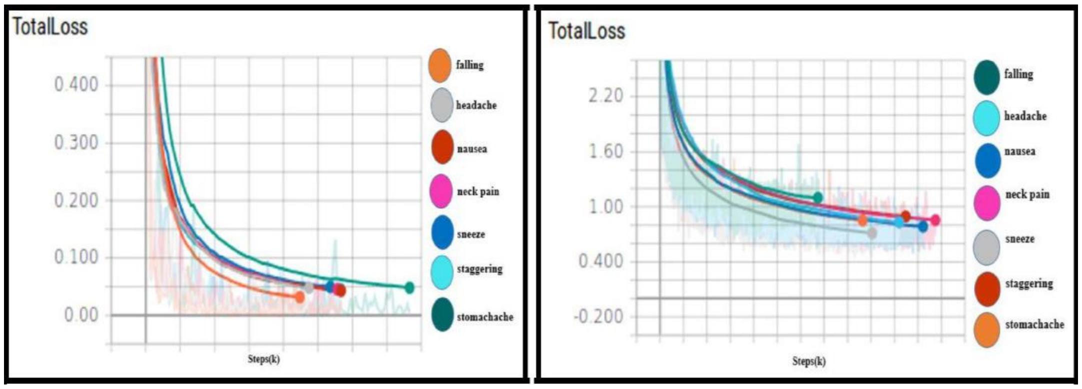


Fig. 8. Minim Total Losses for Seven Human Heath Related Actions Classes are based on Two Per-Trained Models (the left (Faster R-CNN-Resnet), the Right (SSD-Mobilenet)) respectively.

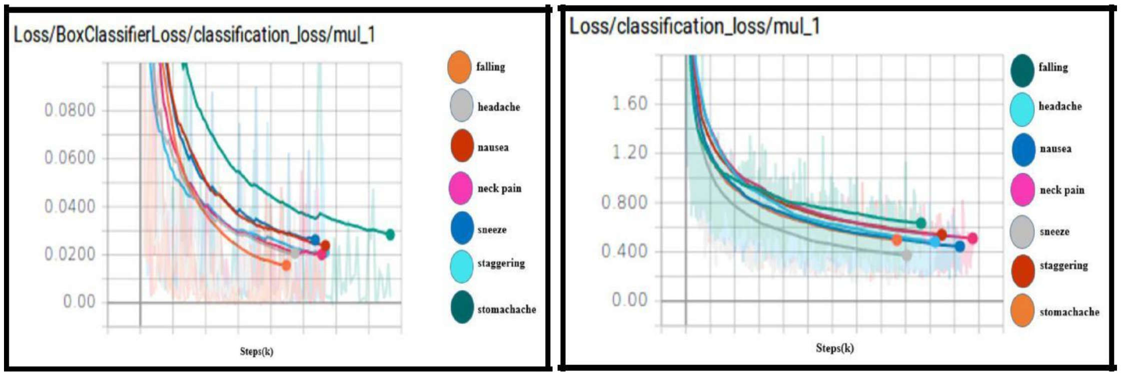


Fig. 9. Minim Classification Losses for Seven Human Heath Related Actions Classes are based on Two Per-Trained Models (the Left (Faster R-CNN-Resnet), the Right (SSD-Mobilenet)) respectively.

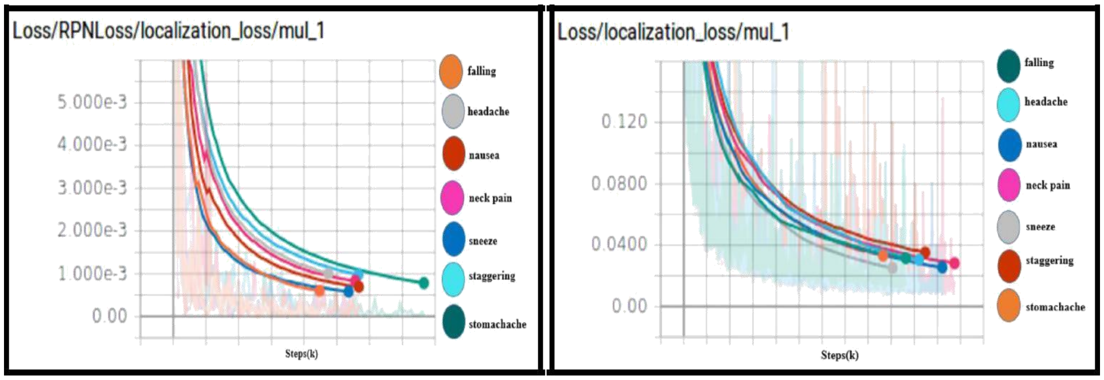


Fig. 10. Minim Localization Losses for Seven Human Heath Related Actions Classes are based on Two Per-Trained Models (the Left (Faster R-CNN-Resnet), the Right (SSD-Mobilenet)) respectively.

TABLE II. THE RESULTS FOR EACH ACTION OF THE PRE-TRAINED MODEL (FASTER R-CNN-RESNET). STEPS (K=1000), TIME (H=HOUR, M=MINUTES, S=SECOND)

Actions	**mAP@ 0.5 IoU**	**Total loss**	**Classification loss**	**Localization loss**	**Steps(k)**	**Time(h-m-s)**
falling	**0.9300**	**0.0312**	**0.0156**	**6.0513e-4**	**8.44**	**2-13-58**
headache	**0.9347**	**0.0479**	**0.0206**	**1.0070e-3**	**9.47**	**2-25-57**
nausea	**0.9425**	**0.0429**	**0.0237**	**6.9610e-4**	**11.34**	**2-47-57**
neck pain	**0.9437**	**0.0451**	**0.0201**	**8.3558e-4**	**11.12**	**2-31-57**
sneeze	**0.9389**	**0.0493**	**0.0261**	**5.8604e-4**	**10.37**	**2-37-57**
staggering	**0.9347**	**0.0455**	**0.0210**	**9.7959e-4**	**11.32**	**2-23-57**
stomachache	**0.9539**	**0.0480**	**0.0283**	**7.8347e-4**	**15.32**	**3-23-57**

TABLE III. THE RESULTS FOR EACH ACTION OF THE PRE-TRAINED MODEL (SSD-MOBILENET). STEPS (K=1000), TIME (D=DAY, H=HOUR, M=MINUTES, S=SECOND)

Actions	mAP@ 0.5 IoU	Total loss	Classification loss	Localization loss	Steps(k)	Time(d-h-m-s)
falling	0.9758	0.9872	0.6332	0.03147	48.19	1d-2h-13m-45s
headache	0.9178	0.8323	0.4829	0.03056	51.04	18h-19m-34s
nausea	0.9793	0.7855	0.4448	0.02550	56.11	20h-11m-44s
neck pain	0.9670	0.8548	0.5094	0.02826	58.75	1d-0h-29m-45s
sneeze	0.9737	0.7144	0.3718	0.02514	45.26	16h-3m-45s
staggering	0.9590	0.8949	0.5391	0.03517	52.38	1d-0h-11m-45s
stomachache	0.9657	0.8515	0.4973	0.03313	43.22	19h-49m-46s

TABLE IV. TOTAL MAP FOR ALL ACTIONS BASED ON TWO PRE-TRAINED MODELS

Total mAP	
Faster R-CNN-Resnet	SSD-Mobilenet
93.8%	95.8%

Tables 2 and 3 summarize all the parameters (mAP, Total loss, Classification loss, Localization loss) of the training and evaluation processes in addition to the number of the steps with the time that they consumed to achieve these requirements. Table 4 labels the mean average precision (total mAP) for all the actions classes of the both per-trained models. The total mAP that belongs to the SSD-Mobilenet model is somewhat higher than the total mAP of the Faster-R-CNN-Resnet model.

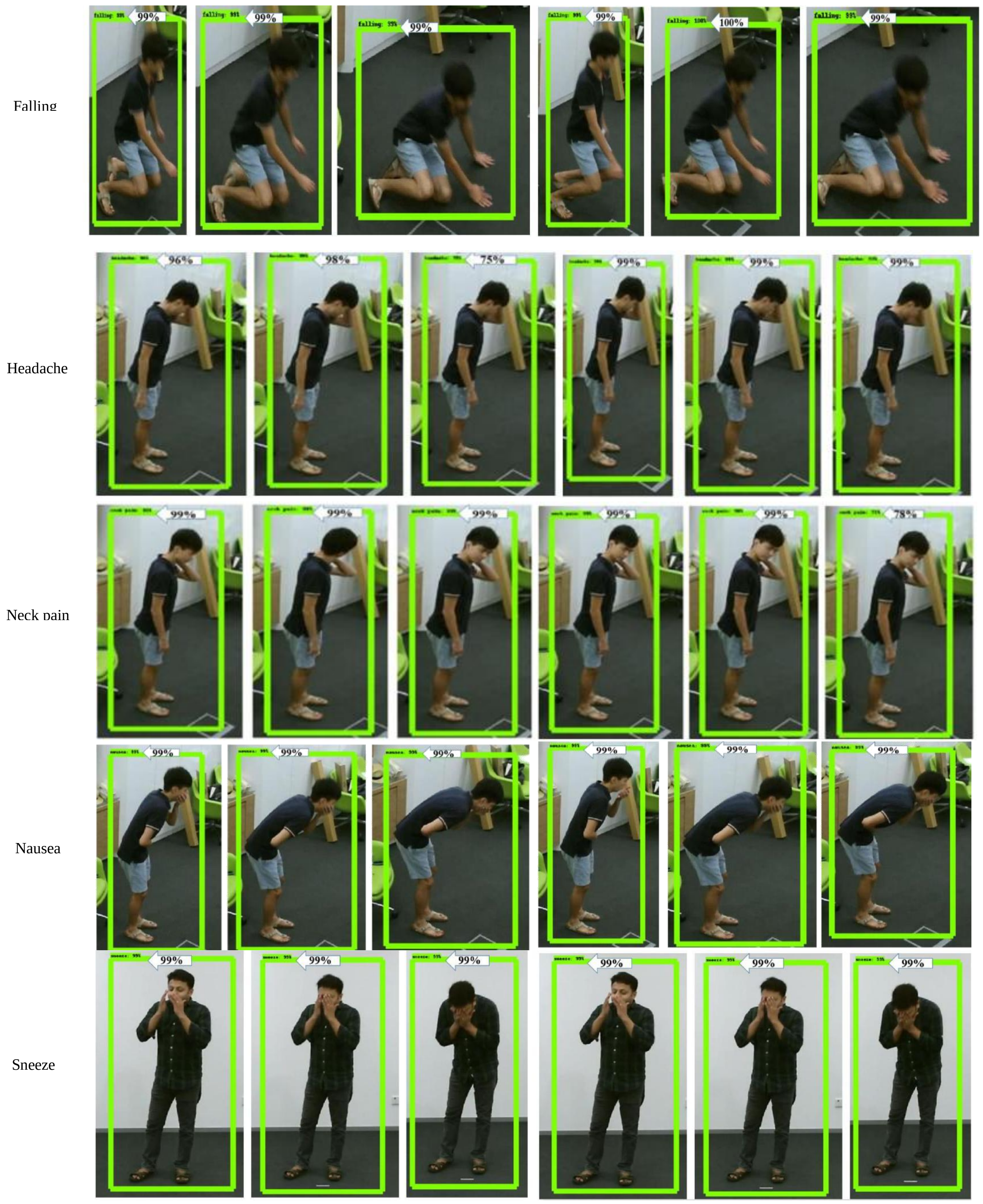
G. Testing (Detection) and Results

To validate the HHRA model's performance for action detection in videos, comprehensive examinations have been achieved on a special type of human action that has related on the health from the NTU RGB+D datasets. In order to accomplish the greatest predictable detection accuracy, sets of human actions frames with different health-related actions and various environments are tested. In this work, the detection process for testing the video frames of seven different human health-related actions is implemented in two ways to the evaluation of the detection performance using the TensorFlow Object Detection Notebook and the Android camera in terms of accuracy and detection speed. The first way is by trying out the TensorFlow Object Detection Notebook with couple pre-trained models (Faster R-CNN-Resnet and SSD-Mobilenet). While the detection using Android camera only uses the SSD-Mobilenet model because the TensorFlow in Android does not support the Faster-R-CNN-Resnet model yet. To perform the detection process in both ways, must export the model as a static inference graph trained on the human health-related dataset, as well as the corresponding label map. The TensorFlow object detection API library provides the script, named export_inference_graph with using the latest checkpoint number at the last step that stopped the training process. It has used the 16.04LTS GTX1070@2.80GHZ x8 system to run the object detector on each frame from seven different human health-related actions of the NTU RGB+D dataset to detect the action type. The detection process has been applied on 50 frames of each video action for seven different actions.

1) TensorFlow's object Detection Notebook: As mentioned above after the requirements is completed, the detection process is accomplished by using TensorFlow's Object Detection Notebook. In pre-trained Faster R-CNN-Resnet model, the bounding detection boxes for each frame from seven different human health-related actions consumed around 120 seconds over all the 50 frames of the testing dataset. While, the bounding detection boxes for each frame in pre-trained SSD-Mobilenet model are finalized within a 95 seconds time span. The detection results in two pre-trained Faster R-CNN-Resnet and SSD-Mobilenet models for samples of frames of each human health-related action are shown in Fig.11. The results displayed a high detection accuracy for all the actions. According to the SSD-Mobilenet model, the results include diverse detection values ranging from intermediate to high values of different action frames.However, there are misdetections in the headache action where SSD-Moblienet failed to demonstrate the bounding detection box in some frames and misdetection that shows the bounding detection boxes in a wrong action placement as shown in Fig.12.

2) TensorFlow in andriod: Once the requirements are completed, as mentioned in sections (VI and VIII-G), the model will be imported to an Android phone (Galaxy S6). This is the time for testing the video frames based on pre-trained SSD-Mobilenet model by capturing them from the phone's camera also bounding detection boxes have been visualize for each frame with the name of the action class and detection percentage accuracy as described in Fig.13. In order to evaluate the impact of the TensorFlow in Android for the detection results, the detection accuracy is improved up to high values and it speed up the detection time.

The action detection from the phone's camera consumes roughly 25 seconds for all 50 video frames. In the headache action, there are two misdetections, however, when the detection using the Android camera, these two problems have been solved. The action detection for all frames come to be all properly visible and the bounding detection boxes have become in the correct action location.


Falling
99%
99%
99%
99%
100%
99%
Headache
96%
98%
75%
99%
99%
99%
Neck pain
99%
99%
99%
99%
99%
78%
Nausea
99%
99%
99%
99%
99%
99%
Sneeze
99%
99%
99%
99%
99%
99%

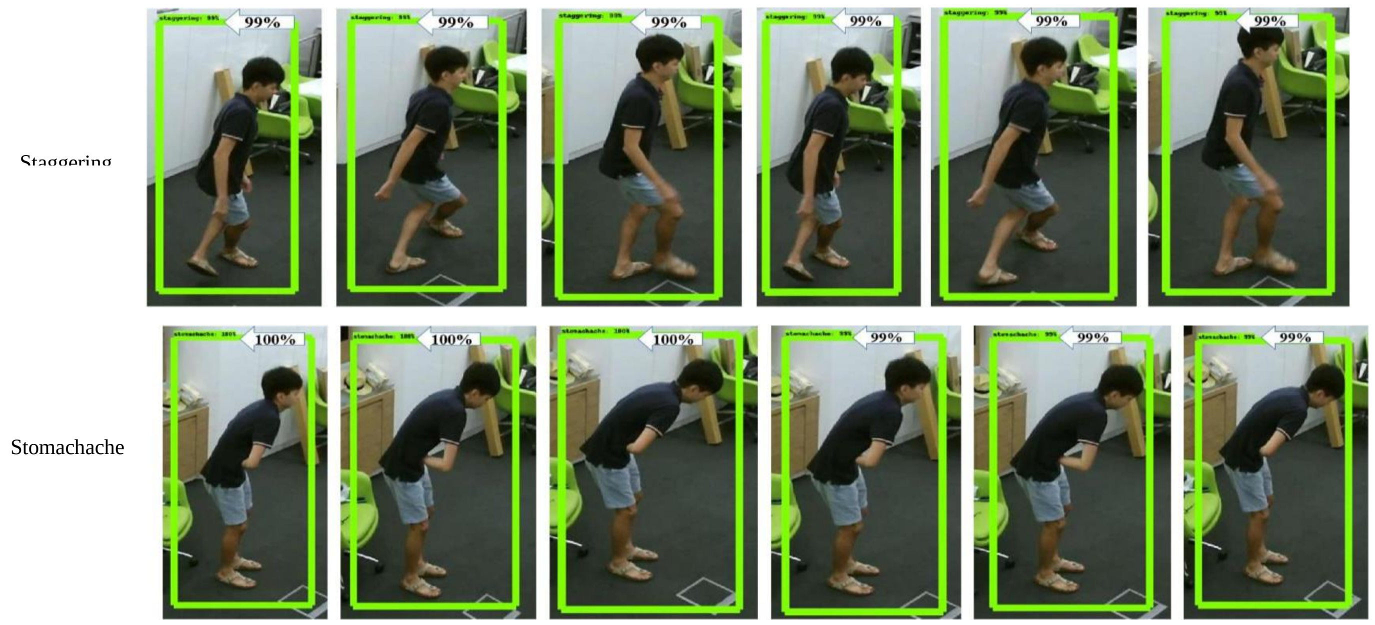


Fig. 11. Samples of The Detection Results that used TensorFlow Object Detection Notebook Technique are based on Two Pre-Trained Models (the First Three Images from the Left are based on Faster R-CNN-Resnet Model, while the Last Three Images from the Right are based on SSD-Mobilenet)) respectively. The Results Include the Bounding Detection Box for Every of the Seven Human Health-Related Actions Including (Name of the Action Class & Detection Percentage Accuracy).

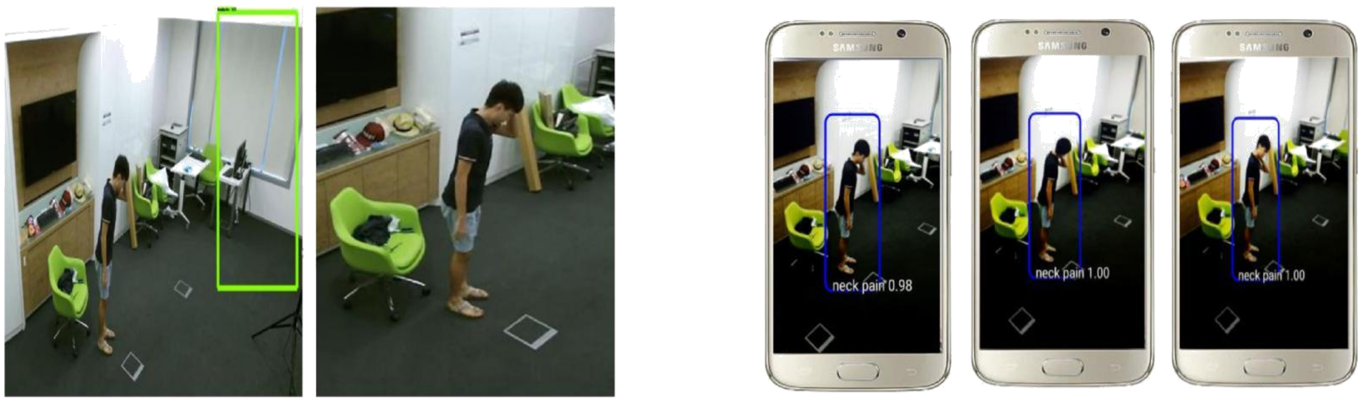


Fig. 12. Misdetections for Some Frames of the Headache Action are based on the Pre-Trained SSD-Mobilenet (the left (Bounding Detection Box Showed in the Wrong Location), the Right (No Bounding Detection Box in the Frame).

IX. Discussion

In this paper, human health related action videos have been detected by using the implemented TensorFlow object detection API technique. The two new pre-trained (Faster R-CNN-Resnet and SSD-Mobilenet) models have been applied for training human actions dataset. The average precision (AP) is the average of class predictions estimated over several thresholds. The detection accuracy (mAP) at 0.5IoU is a high value with different num_steps. This is due to the fact that the network deals with the video images, therefore it takes a long time to train the entire samples of each frame for every action and depend on the type of the model's architecture. The parameter num_steps determines how many training steps they will run before stopping. This number certainly depends on the size of the dataset along with how long the user wants to train the model. Localization loss represents the predicted bounding box which matches with the ground-truth bounding box. However, the loss value is decreased and the Intersection over Union is high, which means that the detection of the action is in the correct location. The classification loss shows the effectiveness of human action class which is classified and matched with the previous trained class. When the classification loss values decrease and are near zero that means that the classification accuracy is outstanding and the performance of the detector becomes more high-level. The Android detection results were compared to the detection process TensorFlow Object Detection Notebook technique. These two distinct processes were used to examine which one does a better job in measuring the detection speed and how accurate the detection is. In the end, utilizing the Android smartphone's camera revealed that the seven types of human health-related actions were precisely detected with high accuracy and reasonable detection speed rate.

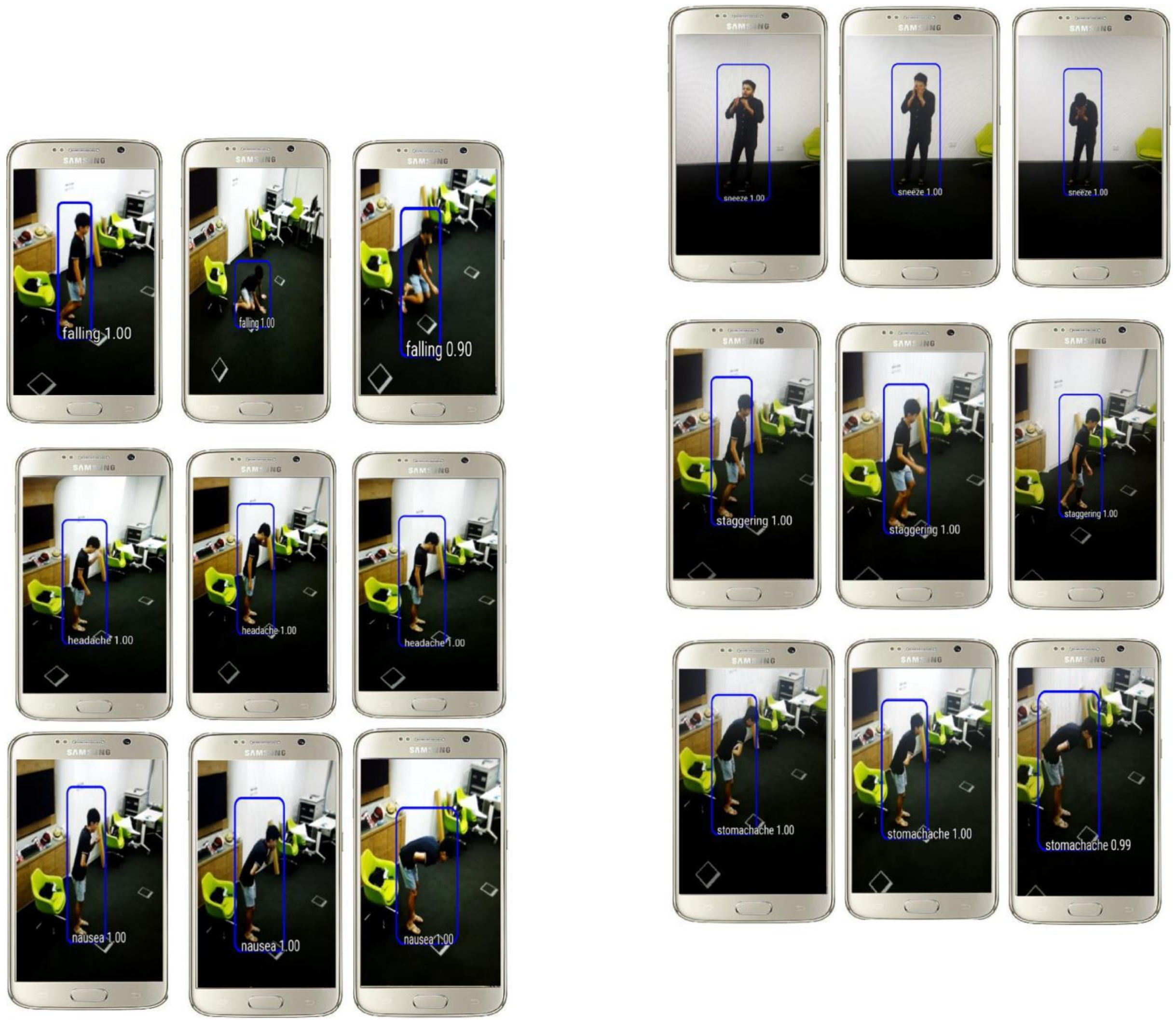


Fig. 13. Samples of The Detection Results that used Android Camera are based on the Pre-Trained SSD-Mobilenet Model. The Results Include the Bounding Detection Box for Seven Human Health- Related Actions Classes Including (Name of the Action Class & Detection Percentage Accuracy).

X. Conclusion

In Conclusion, the health-related actions can be detected with high-speed detection and its great accuracy. It can figure out the correct action required to deal with the appropriate situation using the smartphone camera. A new detector model was built for seven different human health-related video actions using two techniques of TensorFlow object detection API, which are TesnsorFlow object detection notebook and TensoFlow in Android, using the phone's camera. In addition, the HHRA detection model was trained and evaluated using NTU RGB+D dataset based on two pre-trained models (Faster-R-CNN-Resnet and SSD-Mobilenet). According to the results, in the best-detectable category, the mAP total achieved 93.8% for the Faster R-CNN-Resnet and 95.8% for SSD-Mobilenet. In addition, the lowest error was calculated through both losses of classification and localization and the results were satisfactory. Furthermore, the detection speed and the high-performance efficiency have been improved by the use of the smartphone. In the future, we plan to the opportunity to train using Google Cloud to decrease the training and evaluation time. Moreover, new methods can be developed to detect the further human actions classes.

References

[1] K. Matusiak, P. Skulimowski, and P. Strumillo, "Object recognition in a mobile phone application for visually impaired users," IEEE 6th Int. Conf. Hum. Syst. Interact., pp. 479–484, 2013

[2] pkulzc, "TensorFlow Object Detection API repository,"github.com, 2018. [Online]. https://github.com/tensorflow/models/tree/master/research/object_detection. [Accessed: 16-Jul-2018].

[3] P. F. Felzenszwalb and D. P. Huttenlocher, "Pictorial structures for object recognition," Int. J. Comput. Vis.,vol. 61, no. 1, pp. 55–79, 2005.

[4] P. F. Felzenszwalb, R. B. Girshick, D. Mcallester, and D.Ramanan, "Object Detection with Discriminatively Trained Part Based Models," IEEE Trans. Pattern Anal. Mach.Intell., vol. 32, no. 9, pp. 1–20, 2009.

[5] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," Proc. - 2005 IEEE Comput. Soc.Conf. Comput. Vis. Pattern Recognition, CVPR 2005, vol.I, pp. 886–893, 2005.

[6] M. M. Cheng, Z. Zhang, W. Y. Lin, and P. Torr, "BING: Binarized normed gradients for objectness estimation at 300fps," Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., pp. 3286–3293, 2014.

[7] A. Karpathy, A. Joulin, and L. Fei-Fei, "Deep Fragment Embeddings for Bidirectional Image Sentence Mapping," pp. 1–9, 2014.

[8] A. Karpathy and L. Fei-Fei, "Deep Visual-Semantic Alignments for Generating Image Descriptions," IEEE Trans. Pattern Anal. Mach. Intell., vol. 39, no. 4, pp. 664–676, 2017.

[9] Y. Tian, R. Sukthankar, and M. Shah, "Spatiotemporal deformable part models for action detection," Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., pp. 2642–2649, 2013.

[10] L. Wang, Y. Qiao, and X. Tang, "Video Action Detection with Relational Dynamic Poselets," Eccv, 2014.

[11] S. Ren et al., "Rich feature hierarchies for accurate object detection and semantic segmentation," Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., vol. 794, pp. 1–15, 2015.

[12] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," Nips, pp. 91–99, 2015.

[13] C. Sun and K. Murphy, "Speed/accuracy trade-offs for modern convolutional object detectors," vol. 3, 2017.

[14] pkulzc, "TensorFlow Object Detection API repository," github.com, 2018.[Online].Available: https://github.com/tensorflow/models/tree/master/research/object_detection. [Accessed: 16-Jul-2018].

[15] L. Torrey and J. Shavlik, "Transfer Learning," Mach. Learn., pp. 1–22, 2009.

[16] V. Rathod and N. Wu, "Tensorflow detection model zoo," 2017. [Online].Available: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md.[Accessed: 25-Jun-2018].

[17] W. L. B et al., "SSD : Single Shot MultiBox Detector," vol. 1, pp. 21–37, 2016.

[18] C. Szegedy, S. Reed, D. Erhan, D. Anguelov, and S. Ioffe, "Scalable, High-Quality Object Detection," 2014.

[19] Karpathy, "Smooth L1 Loss," mohitjainweb.files.wordpress.com, 2015. [Online].Available: https://mohitjainweb.files.wordpress.com/2018/03/smoothl1loss.pdf. [Accessed: 15-Jul-2018].

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conf. Comput. Vis. Pattern Recognit., pp. 770–778, 2016.

[21] O. Russakovsky et al., "ImageNet Large Scale Visual Recognition Challenge," Int. J. Comput. Vis., vol. 115, no. 3, pp. 211–252, 2015.

[22] T. Y. Lin et al., "Microsoft COCO: Common objects in context," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 8693 LNCS, no. PART 5, pp. 740–755, 2014.

[23] V. Nair and G. E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines," Proc. 27th Int. Conf. Mach. Learn., no. 3, pp. 807–814, 2010.

[24] A. G. Howard et al., "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," 2017.

[25] M. Siam, M. Gamal, M. Abdel-Razek, S. Yogamani, and M. Jagersand, "RTSeg: Real-time Semantic Segmentation Comparative Study," no. 1, 2018.

[26] Tensorflow, "https://github.com/tensorflow," 2016. [Online]. Available: https://github.com/tensorflow/tensorflow/tree/master/tensorflow/examples/[Accessed: 20-Jun-2018].

[27] B. Readme, A. Inception, L. Repre, and R. Third, "TensorFlow Android Camera Demo Prebuilt Components : Building in Android Studio using the TensorFlow AAR from JCenter," pp. 2–5, 2018.

[28] F. Liu, Android Native Development Kit Cookbook. Birmimgham B3 2PB, UK, 2013.

[29] Campbell Scientific, "CSI Software Development Kit Beginner's Guide," s.campbellsci.com,2006.[Online].Available: https://s.campbellsci.com/documents/de/manuals/sdkbeginnersguide.pdf. [Accessed: 20-Jul-2018].

[30] A. Shahroudy, J. Liu, T.-T. Ng, and G. Wang, "NTU RGB+D: A Large Scale Dataset for 3D Human Activity Analysis," 2016.

[31] S. Nadella, L. Manifest, M. Readme, O. S. H. Sierra, and M. Os, "LabelImg,"github.com, [Online].Available: https://github.com/tzutalin/labelImg. [Accessed: 05-Jun-2018].

[32] V. Toolkit, "TensorBoard Key Concepts Summary Ops : How TensorBoard gets data from TensorFlow," github.com, 2018. [Online]. Available: https://github.com/tensorflow/tensorboard. [Accessed: 24-Jul-2018].

[33] P. Mustamo, "Object detection in sports: TensorFlow Object Detection API case study," no. January, 2018.

[34] M. Everingham, L. Van Gool, C. K. I. Williams, and J. Winn, "The P ASCAL Visual Object Classes (VOC) Challenge," Int. J., pp. 303–338, 2010.