[AI-Machine Learning] Optimized Sensorless Human Heartrate Estimation for a Dance Workout Application

G. Jeong[1] ; N. Freitas[2]
[1]Gyumin Jeong, Hansung University, Devunlimit
[2]Núria Freitas, Catholic University of Korea, Devunlimit
Seoul Business Agency, Seoul Innovation Challenge 2019

Abstract:- Over the last decade, there has been a great effort to use technology to make exercise more interactive, measurable and gamified. However, in order to improve the accuracy of the detections and measurements needed, these efforts have always translated themselves into multiple sensors including purpose specific hardware, which results in extra expenses and induces limitations on the final mobility of the user. In this paper we aim to optimize a sensorless system that estimates the real-time user heartrate and performs better than the current wearable technology, for further calorie and other vital indicators calculations. The findings here will be applied on a posture correction system for a dance and fitness application.

Keywords:- *Sensorless heartrate estimation – Artificial Intelligence (A.I.) – Machine learning – Posenet – DenseNet – Real-time heartrate estimation – Heartrate at a distance– Dance – K-pop – E-sports – South Korea.*

I. INTRODUCTION

Heartrate is a well-known indicator of the intensity of exercise. From heartrate, other indicators such as burnt calories can be derived. Currently most commercial products on the fitness and diet market require sensors to estimate the heartrate. Common examples for these sensors are smart watches and chest bands.

Due the nature of our platform, we aim to eliminate the necessity of wearable sensors, letting the user free to move and perform fast movements around safely.

Recently, a growing number of studies have been proving the feasibility and accuracy of remote heartrate estimations, as it has been shown that video can capture minute skin color changes that follow the heart pulse and that can't be seen through naked eye.

On this paper we aim to extract vital data by estimating the users' pulse through machine-learning, from RGB signals using the minute skin color change that the face camera can catch, and achieve an equal or higher accuracy rate than the most commonly used smart watch. We also change the used convolutional neural network (CNN) from Resnet to Densenet and tried to improve the accuracy from the researched system.

II. OBSERVATION METHODS AND INDICATORS

The observations on this paper have been done using a real-time video taken with a single regular RGB camera, instead of regular infrared cameras or smart watches' photoplethysmography.

For further calculations we measured the number of frames that have data outputs in a second to set the actual service frame per second (FPS) rate.

Since fast and diverse motion, as well as a high illuminance (lux) could be problematic in the measure of the data, in order to determine whether or not the normal operation is possible within those conditions, we experimented with over the regular values of illuminance and tested over the dataset.

< Overview of Major Performance Indicators >					
Performance Indicator	Unit	Final objective	World's highest level (Holding company / holding country)	Percentage (%)	Measuring organization
1. Data output speed	fps	More than 24fps	15~30 fps (Microsoft, US)	30	TTA accredited certification test
2. Illuminance	lux	More than 5,000lux	Less than 5,000lux (Microsoft, US)	30	TTA accredited certification test
3. Heart rate match rate	%	More than 90%	90%(Apple, US)	20	TTA accredited certification test

<Sample Definition and Measurement Method>			
Performance Indicator	Sample Definition	Number of samples ($n \geq 5$)	Measurement method (standard, environment, result calculation, etc.)
1. Data output speed	1 per System	5	Measured frames per second (fps) in the result data value image taken through the deep learning machine when shooting in real time
2. Illuminance	1 per System	5	Measures illuminance (lux) in the driving environment, and executes in the environment above the standard illuminance value.
3. Heart rate match rate	1 per System	5	Measures the match rate (%) between the predicted value and the actual heart rate using artificial neural network technology through real-time image data (frames with accurate heart rate / full frame)
4. Observation position error	1 per System	5	Measure the accuracy of the skeleton value obtained by the deep learning machine when shooting in real time

Table 1:- Performance indicators, objectives and observation methods.

III. PROCESS AND SCHEDULES

It took 3 months to design and develop a new artificial neural network algorithm for heart rate measurement to measure and provide information about heart rate to application users by using image data captured by RGB camera without specific sensor. Another month to verify and correct errors.

IV. EXPERIMENT

A. *Select Data Set*

For this experiment we choose to use the VIPL-HR Database (Pure Database). The reasons for that choice were because of its large multi-model with 2,378 visible light video (VIS), 752 near infrared video (NIR) recorded from 107 different people that includes various variations such as head movements, lighting variations, and filmed device changes. Also, because we could learn about the proposed spatio-temporal representation used to monitor and estimate the heart rate (RhythmNet) that has very promising results in both public domain and VIPL-HR estimation databases. Lastly, the VIPL-HR Dataset is distributed to universities and research institutes for research purposes and its license is free.

Some of the features of other databases unused in the experiment and the reasons why we do not choose them include:

- HCI Tagging Database: License is free, but the end user cannot use the database for non-academic purposes as a license agreement.
- BU-3DFE (Binghamton University 3D Facial Expression) Database: Available to external parties only under the agreement of the licensing offices of Binghamton University and Pittsburgh University, and only to external parties seeking research for non-profit purposes.
- Oulu Bio-Face (OBF) database: No download link is available.
- Cohface database: only for scientific research.

B. *Data Set Pre-processing*

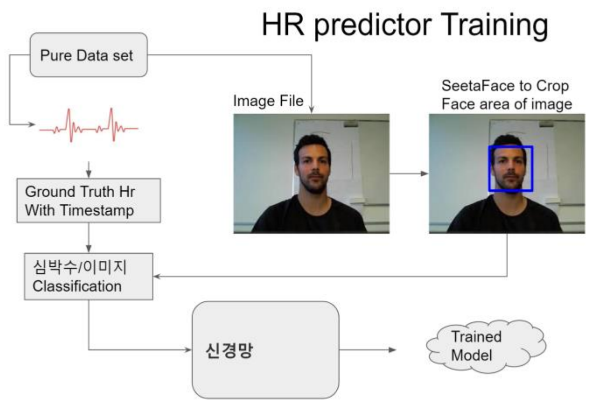
HR predictor Training
Pure Data set
Image File
SeetaFace to Crop Face area of image
Ground Truth Hr With Timestamp
심박수/이미지 Classification
신경망
Trained Model

Fig 1:- Cloud software flow chart for heart rate extraction

➢ *Seetaface Face*

SeetaFace is a facial recognition engine for heart rate extraction and works with three core modules. First, the FaceDetector module is a face detection module. The FaceLandmarker module is the module that identifies the key position of the face. The FaceRecognizer module is a module that extracts and compares facial features. Using the 3-core module Seetaface face recognition engine, after the face in the video is detected, the position of the detected face part is determined. Thereafter, only a face portion is cut for analysis from the predetermined face position. The cropper used with the mentioned engine module was produced by SeetaTech and can be obtained from the link below as an open source under the BSD 2-Clause license.

https://github.com/seetafaceengine/SeetaFace2/blob/master/README_en.md,
https://github.com/seetaface/SeetaFaceEngine2/tree/master/FaceCropper

➢ *Cropped face image classification*

After converting a file containing a large number of frames in avi format in Pure Database into several files in png format and executing detect_face.exe, the human face is detected in the png format file. We stored the coordinates in a csv file. Using face coordinates saved as csv file, execute face_cropper.exe and we cut only the face part into 256x256 size and save it as a png format file. Using csv files containing time stamp information for each frame, heart rate information for each frame's time stamp is linked through comparison in the csv file containing the heart rate information for each time stamp. After that, we moved the png file with only the 256x256 face cut into a folder defined by heart rate.

C. *Training*

➢ Execution Environment (Hardware / software)

CPU	Intel i7-7700k
Memory	DDR4 32GB
Graphic Card	Nvidia GTX1660Ti 6GB
SSD	256 GB

➢ Execution Count / Time

Based on the GPU memory capacity, it is set to batch size 2. 400 times of study, total time required one week.

D. *Verification*

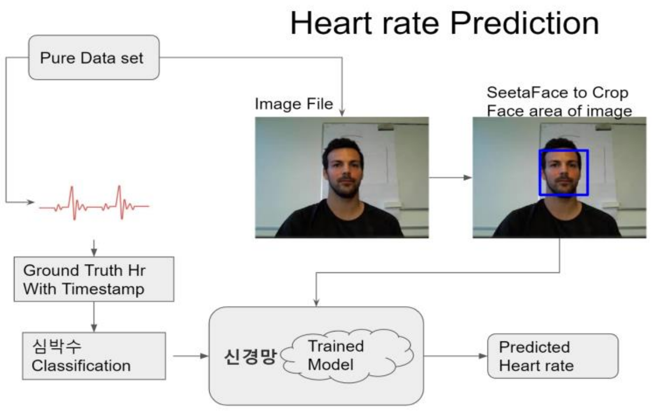

Fig 2

V. PROCEDURES

The test procedure for extracting heart rate is as follows. After installing Ubuntu, install the utilities associated with your development source. Run the following command to install the relevant utility.

```
(1)  > sudo add-apt-repository ppa:deadsnakes/ppa
(2)  > sudo apt-get update
(3)  > sudo apt-get –y upgrade
(4)  > sudo apt-get install -y vim python3.6 python3.6-dev htop gparted make openssh-server curl git
(5)  > curl https://bootstrap.pypa.io/get-pip.py | sudo -H python3.6
(6)  > sudo update-alternatives –install /usr/bin/python python /usr/bin/python3.6 1
(7)  > sudo update-alternatives –install /usr/bin/pip pip /usr/bin/   pip3.6 1
(8)  > sudo su (get root privileges)
(9)  > passwd root (Set root external password)
(10)  > wget http://download.jetbrains.com/python/pycharm-community-2019.1.1.tar.gz
(11)  > tar –zxvf pycharm-community-2019.1.1.tar.gz
```

Next, we install a GPU (including cuda) on Linux. In the same way as above, write the following command to install.

```
(93)      > sh -c ‘echo “deb
http://developer.download.nvidia.com/compute/cuda/repos/ubuntu1604/x86_64/” >> /etc/apt/sources.list.d/cuda.list’
  (2) > sh -c ‘echo “deb http://developer.download.nvidia.com/compute/machine-learning/repos/ubuntu1604/x86_64 /” >>
/etc/apt/sources.list.d/cuda.list’
  (3) > apt-get update
  (4) > apt-cache search nvidia
  (5) > apt-get install nvidia-410
  (6) > apt-ket adv –fetch-keys http://developer.download.nvidia.com/compute/cuda/repos/ubuntu1604/x86_64/7fa2af80.pub
  (7) > apt-get update
(8) > apt-get install cuda-9-0 nvidia-cuda-toolkit
  (9) > cat /usr/local/cuda/version.txt
  (10) > wget http://developer.download.nvidia.com/compute/redist/cudnn/v7.1.4/cudnn-9.0-linux-x64-v7.1.tgz
  (11) > tar xvfz cudnn-9.0-linux-x64-v7.1.tgz
  (12) > cp -P cuda/include/cudnn.h /usr/local/cuda/include
  (13) > cp -P cuda/lib64/libcudnn* /usr/local/cuda/lib64
  (14) > chmod a+r /usr/local/cuda/include/cudnn.h/usr/local/cuda/lib64/libcudnn*
  (15) > apt-get install –y libcupti-dev
  (16) > vi /home/user/.bashrc
export PATH=/usr/local/cuda/bin:$PATH
export
LD_LIBRARY_PATH=”$LD_LIBRARY_PATH:/usr/local/cuda/lib64:/usr/local/cuda/extras/CUPTI/lib64”
export CUDA_HOME=/usr/local/cuda
  (17) > vi ~/.bashrc
export PATH=/usr/local/cuda/bin:$PATH
export
LD_LIBRARY_PATH=”$LD_LIBRARY_PATH:/usr/local/cuda/lib64:/usr/local/cuda/extras/CUPTI/lib64”
export CUDA_HOME=/usr/local/cuda
  (18) > source ~/.bashrc
  (19) > nvcc –version
```

When the installation is complete, install the Python compiler on Linux. Download the pycharm distribution package from https://www.jetbrains.com/pycharm/download/#section=windows and extract the compressed file.

VI. RESULTS AND DISCUSSION

The experiment has tested two items: the accuracy of measured heart rate and the reported heart rate from the dataset and the heart rate analysis FPS.

The test was conducted five times for each item, and it was confirmed whether all of the tests met the final development goals presented in the previous section 2, ‘Observation methods and indicators’. As shown in the test results below, the heart rate accuracy measured was 93.28%, which met the target value of 90% or higher from the commonly used smartwatch standard and even surpassed by 0.66% the current Resnet model, The heart rate analysis FPS was 69.704 to 70.660 fps, which all met the target value of 24 fps or higher.

Index	Tested Images	Successfully Tested Images	Accuracy
1	15,141	14,123	93.28%
2	15,141	14,123	93.28%
3	15,141	14,123	93.28%
4	15,141	14,123	93.28%
5	15,141	14,123	93.28%
Average			93.28%

Table 2:- Heartrate Accuracy with DenseNet.

Index	Tested Images	Analysis Time	Heartrate Image Analysis FPS
1	15,141	214.71 sec	70.522 FPS
2	15,141	217.02 sec	69.768 FPS
3	15,141	214.28 sec	70.660 FPS
4	15,141	215.36 sec	70.306 FPS
5	15,141	217.22 sec	69.704 FPS
Average			70.192 FPS

Table 3:- Heartrate Analysis FPS

REFERENCES

[1]. http://vipl.ict.ac.cn/uploadfile/upload/2018111615545295.pdf
[2]. https://www.cv foundation.org/openaccess/content_cvpr_2016/papers/Tulyakov_Self Adaptive_Matrix_Completion_CVPR_2016_paper.pdf
[3]. https://mahnob-db.eu/hci-tagging/
[4]. http://www.cs.binghamton.edu/~lijun/Research/3DFE/3DFE_Analysis.html
[5]. https://ieeexplore.ieee.org/document/8373836
[6]. https://www.idiap.ch/dataset/cohface