Indian Institute of Technology Indore

CS 419/619 - Computer Vision

P I K A C H U

Name: Amey Kiran Patel *Roll number: 160001003*
Name: Amit Kumar Meena *Roll number: 160001004*
Name: Ghanshyam Bairwa *Roll number: 160001022*

June 25, 2020 (Spring 2020)

Indian Institute of Technology
Indore

Contents

1 Introduction

In this project, we create an android application named **'P I K A C H U'**. The name is inspired from the popular TV franchise - Pokemon when Pikachu constantly supports the main protagonist.

This application can be used primarily for medical purposes and to track the blood flow and heart rate of an examinee. Having completed the project early on, we have further integrated functionalities such as OCR to further expand the scope of our project. OCR can be used by elderly patients to identify the medicines that they are short of. This output from the OCR can be sent to another pipeline that sends a notification to the medical shop to replenish the stock for the patient.

The current application is available at the Playstore and can be downloaded for use.

2 Background

2.1 Optical character recognition (OCR)

Optical character recognition (OCR) is the electronic identification and digital encoding of typed or printed text by means of an optical scanner and specialized software. Using OCR software allows a computer to read static images of text and convert them into editable, searchable data. OCR typically involves three steps: opening and/or scanning a document in the OCR software, recognizing the document in the OCR software, and then saving the OCR-produced document in a format of your choosing.

OCR can be used for a variety of applications. In academic settings, it is oftentimes useful for text and/or data mining projects, as well as textual comparisons. OCR is also an important tool for creating accessible documents, especially PDFs, for blind and visually-impaired persons.

2.2 Photoplethysmography(PPG)

Photoplethysmography (PPG) is an uncomplicated and inexpensive optical measurement method that is often used for heart rate monitoring purposes. PPG is a non-invasive technology that uses a light source and a photodetector at the surface of the skin to measure the volumetric variations of blood circulation. Recently, there has been much interest from numerous researchers around the globe to extract further valuable information from the PPG signal in addition to heart rate estimation and pulse oximetry readings.

Wearable health monitoring technologies, including smartwatches and fitness trackers, have attracted considerable consumer interest over the past few years.

2.3 Missing value Identification

OCR is used extensively in the pipeline for building larger products and for larger applications. For example, hospitals, these days, provide unique barcodes to each patient which are used by scanners to identify the patient's history. Essentially, this is a lookup application in a table stored in the backend. The barcode is unique and can be mapped to a tuple in the database which is then retrieved. Similarly, we have implemented a prescription card in which a patient

can enter information about the symptoms or disease he has and this can be used to identify the medicines that one might need to take. This a simple use case of OCR along with its integration to a backend database server.

3 Work done

3.1 Optical Character Recognition(OCR)

Optical Character Recognition is one of the oldest computer vision tasks and some variants do not require any deep learning. Deep learning is very useful in providing state-of-the-art results, we have decided to use this for our final integration. We have also tried OCR with more traditional methods as they would be more aligned to the material that has been covered so far in the course but they did not give good results. An OCR task has to consider the following attributes:

1. Font type and size
2. Character type
3. The density of text
4. Text structure
5. Background noise

3.1.1 The strategy used - METHOD 1

It gets an image to parse and sends it to a server where our trained model is present. It first applies the erosion operation two times and then applies the dilation operation two times. These operations belong to morphological operations.

1. Normally the characters or text exist in very different colors or texture from the background or surroundings. And most of the text remains in the same grey level. So, we can use the histogram here which will give the positions of the pixels which belong to the text.
2. Mostly texts contain two types of grey level, one for text (foreground pixels) and other for the background (background pixels). That is why the histogram will contain two picks, one for text and one for the background. For two grey level image thresholding based segmentation with single thresholding is sufficient. So, we used the single thresholding segmentation method.
3. Then we took every segment obtained from left to right and top to bottom with standard values. Then it finds the percentage of matching (i.e the overlap) with standard characters. For matching first, we find the shape of the character then scale the size of the standard character to scale our segmentation.
4. Then it extracts the character which has the highest percentage of matching with the pixel positions and sends it back to the application.

The methods that were used are:

- Morphology
- Histogram
- Segmentation

3.1.2 The strategy used - METHOD 2

There are multiple ways to perform OCR. For our task, we get an image from the camera and send it to a go server which has a wrapper over a python class that contains the trained model with pretrained weights. We have used the SSD Keras implementation in https://github.com/pierluigiferrari/ssd_keras. The choices like YOLO and Mask RCNN could have been other suitable alternatives. But on a small dataset that we initially trained on, we found the results of the Single Shot Detection (SSD) to be more promising. We perform fine-tuning on the pretrained weights to avoid having to train the entire model.

The architecture is as follows: SSD has two components: a backbone model and SSD head.

- The Backbone model is a pre-trained image classification network that is used to extract high-level features. It is usually a large residual network like ResNet which is trained on the ImageNet from which the fully connected classification layer, in the end, is removed. We are consequently left with a deep neural network architecture that can remove semantic significance from the information picture while saving the spatial structure of the picture though at the cost of it being lower in resolution. For ResNet34, the backbone results in a 256 7x7 feature map for an input image.
- The SSD head is only one or more convolutional layers added to this backbone and the yields are deciphered as the bounding boxes and classes of objects in the spatial area of the activations of the last layers.

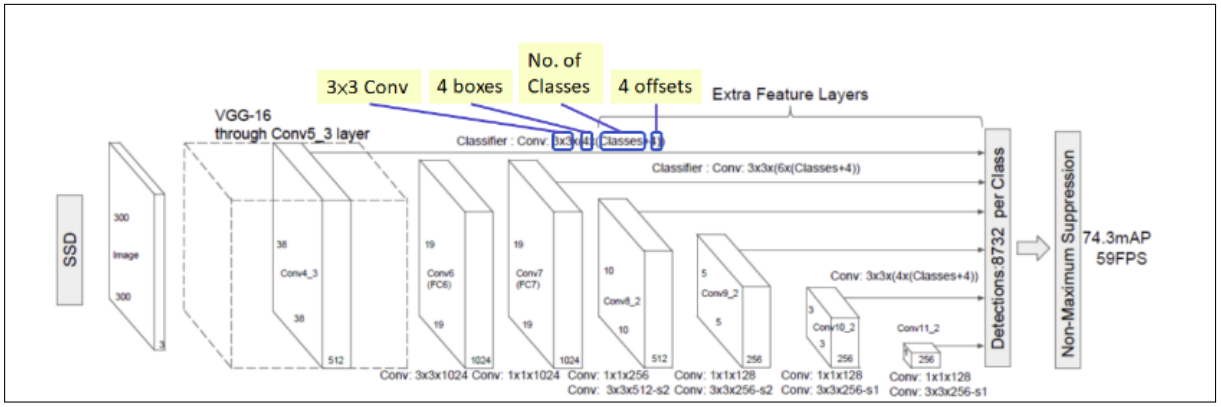


Figure 1: SSD Model Architecture

As should be obvious from the chart over, SSD's design expands on the revered VGG-16 engineering, yet disposes of the completely associated layers. The explanation VGG-16 was utilized as the base system is a direct result of its solid execution in top-notch picture order assignments and its fame for issues where move learning helps in improving outcomes. Rather than the first VGG completely associated layers, a lot of assistant convolutional layers were included, in this manner empowering to extricate highlights at numerous scales and continuously decline the size of the contribution to each ensuing layer.

This model is then fine-tuned on the MSRA Text Detection 500 Database (MSRA-TD500).

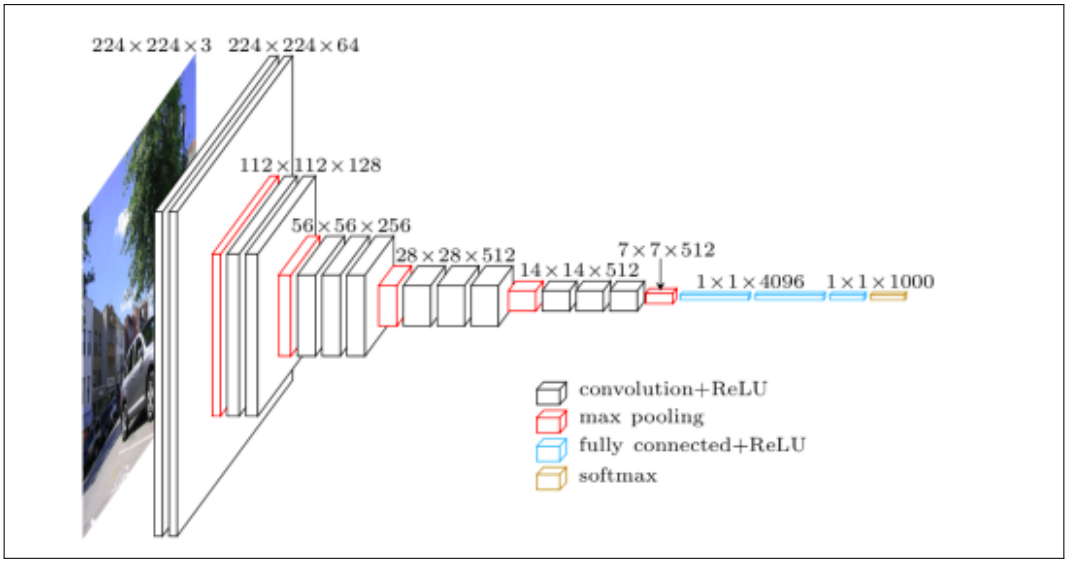


Figure 2: VGG-16 Model Architecture

Important parameters in SSD

1. Grid cell: SSD divides the image using fields and each of these cells is responsible for detecting objects ie. predicting the class and location of the object within the region. If there is no object in the grid cell, the location is ignored. A cell may have multiple objects.

2. Anchor box: Each of the grid cells can be assigned multiple anchor boxes. These boxes are already defined and are responsible for the size and shape of objects within a grid cell. Consider, the swimming pool image corresponding to the taller anchor box while the building image is represented by the wider box. SSD utilizes a matching stage while training, to coordinate the proper grapple box with the bounding boxes of each ground truth object inside a picture. Basically, the anchor box with the furthest extent of cover with an item is liable for predicting that object's class and its location. This property is utilized for training the system and for predicting the objects once the system has been trained. Practically speaking, each anchor box is determined by an aspect ratio and a zoom level.

3. Aspect ratio: Many objects are not square in shape and we need to use anchor boxes with different ratios.

4. Zoom level:This parameter specifies how much we need to scale the anchor boxes as it may be needed that the anchor box has a different size than the grid cell.

5. Receptive Field: It is defined as the region in the input space that a particular CNN's feature is looking at. Because of the property of convolution, features at different layers represent different sizes of the input image. The deeper we go, the larger image size that is represented by the feature. The image below, shows a series of convolution operations applied sequentially. The middle layer feature (green) represents a 3x3 region of the input layer. Similarly, the top layer (orange) represents a 7x7 region in the input space. These arrays (green and orange) are called feature maps which are nothing but the features that are created by applying the convolution operation at different locations of the input map in a sliding window fashion.

Receptive field is the foundational concept of the SSD architecture as it allows us to detect objects at various scales and put a bounding box that is the best fit. A very approach is used in YOLO architecture for object detection. As the shallow layers represent smaller receptive field, they can be used to represent smaller objects. It is precisely the predictions from these shallow layers that help in the prediction of small sized objects.

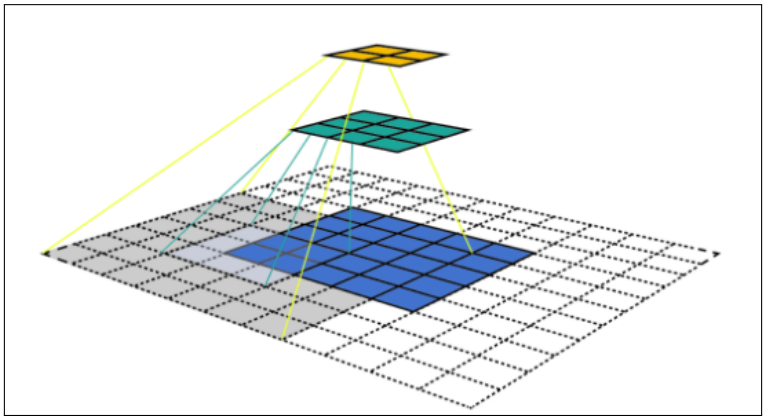


Figure 3: Visualizing CNN feature maps and receptive field

3.2 Photoplethysmography(PPG)

PPG stands for photoplethysmogram. This is a plethysmogram which is generated optically. This can be used to detect the changes in the volume of blood in the tissues of a human body. This technique has gained ground recently in the detection of diabetes. Many wearables including Fitbit work on this principle. By illuminating a blood tissue, we can measure the variance in the reflected light and use this data to find the variation in blood flow. The information about the blood flow can be further extrapolated to calculate the heartbeat of a person.

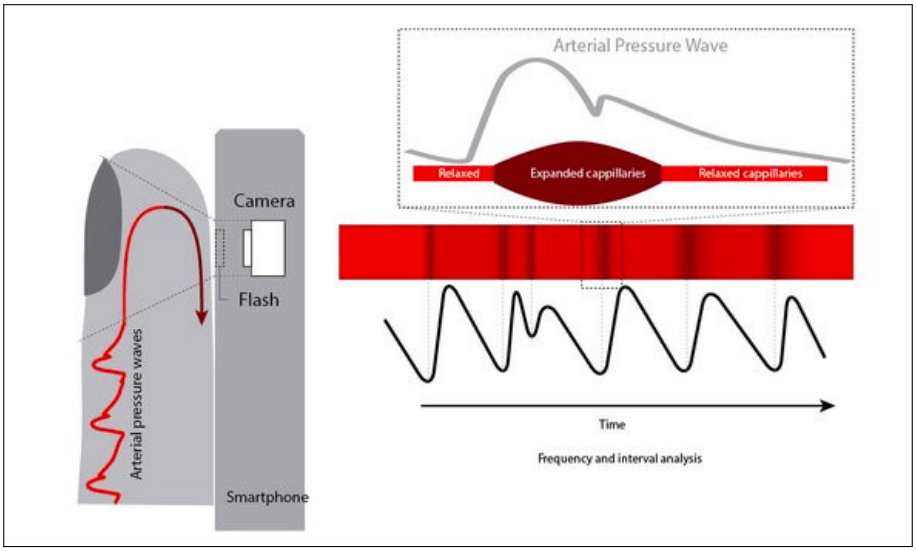


Figure 4: PPG principle by smartphone.

3.2.1 Brief overview

The examinee is supposed to place a finger on the flashlight. By measuring the intensity of the reflected light using the phone's camera, we can process the image taken by the camera. This can be used to calculate the instantaneous heat beat. By continuously performing this procedure, we are able to compute the blood flow of the person.

3.2.2 Libraries used

We use the following dependencies to visualize data and to get access to the sensors of the application.

1. Charts_flutter - Data visualization library that is written in Dart.
2. wavelock - plugin to toggle screen wavelock and prevents screen from turning off automatically.
3. Camera - plugin to allow access to the phone's camera

3.2.3 Image Processing

The function_scanImage() is responsible for the processing of the image. It computes the average of the camera image's red channel, which is represented as the first channel in the set of (R, G, B) channels. This is added to the collection of points that are displayed in the application finally. We impose a restriction on the number of data points that we display to 50 so that our displayed graph is not overpopulated. It is not necessary to process every frame. Hence we have chosen a sampling rate of 30 samples a second. This is done by setting the variable _processing to true for the first 1/30 seconds after the _scanImage() function is called. It is then set to false and we stop collecting samples for this particular function call. This simple methodology is used by leading fitness wearable devices around the globe. This algorithm is given in figure 5 (Algorithm 1).

```
_scanImage(CameraImage image) {
  double _avg =
      image.planes.first.bytes.reduce((value, element) => value + element) /
          image.planes.first.bytes.length;
  if (_data.length >= 50) {
    _data.removeAt(0);
  }
  setState(() {
    _data.add(SensorValue(DateTime.now(), _avg));
  });
  Future.delayed(Duration(milliseconds: 1000 ~/ 30)).then((onValue) {
    setState(() {
      _processing = false;
    });
  });
}
```

Figure 5: Algorithm 1

3.2.4 Heart Beat Estimation

The above procedure measures the variance in the volume of blood flow and is used to display the chart in the application. Our next task is to compute the heart rate.

```
Heart rate = Frequency of the plotted signal.
```

There are multiple algorithms that can be used to carry out the computation of this metric. The algorithm, given below in figure 6 (Algorithm 2), that we used measures the average and the maximum of the data points in our data window and sets the threshold to the mean of those values. We then detect the peaks above the threshold and update the BPM value with a coefficient called the attenuation coefficient to avoid abrupt changes.

```
_updateBPM() async {
  List<SensorValue> _values;
  double _avg;
  int _n;
  double _m;
  double _threshold;
  double _bpm;
  int _counter;
  int _previous;
  while (_toggled) {
    _values = List.from(_data);
    _avg = 0;
    _n = _values.length;
    _m = 0;
    _values.forEach((SensorValue value) {
      _avg += value.value / _n;
      if (value.value > _m) _m = value.value;
    });
    _threshold = (_m + _avg) / 2;
    _bpm = 0;
    _counter = 0;
    _previous = 0;
    for (int i = 1; i < _n; i++) {
      if (_values[i - 1].value < _threshold &&
          _values[i].value > _threshold) {
        if (_previous != 0) {
          _counter++;
          _bpm +=
              60000 / (_values[i].time.millisecondsSinceEpoch - _previous);
        }
        _previous = _values[i].time.millisecondsSinceEpoch;
      }
    }
    if (_counter > 0) {
      _bpm = _bpm / _counter;
      setState(() {
        _bpm = (1 - _alpha) * _bpm + _alpha * _bpm;
      });
    }
    await Future.delayed(Duration(milliseconds: (1000 * 50 / 30).round()));
  }
}
```

Figure 6: Algorithm 2

3.3 Prescription Card

- This functionality allows the patient to recognize which medicine they need to take for a given set of symptoms or disease. This is basically a look-up table in the back-end that

is dependent on the scanning and OCR on the image.

- It takes the input image and sends it to the back-end server. The back-end server extracts the text using OCR and performs a simple search that matches it to the most similar image. If there are no missing values (represented by '?'), the contents of this image is saved in the database for future retrieval. If there are missing values, the matching algorithm is run and we retrieve the tuple in the database with which it matched perfectly and return the value of the attributes that are '?' with the actual values stored in the database for the login card it matched with. Note that the image must be in a specific format. Any image not in the format required will be deemed invalid. The format of the image required is provided in the results section.

4 Dataset

The dataset used for OCR is MSRA Text Detection 500 Database (MSRA-TD500). Detection 500 Database (MSRA-TD500) is collected and released publicly as a benchmark to evaluate text detection algorithms, for the purpose of tracking the recent progress in the field of text detection in natural images, especially the advances in detecting texts of arbitrary orientations.

The MSRA Text Detection 500 Database (MSRA-TD500) contains 500 natural images, which are taken from indoor (office and mall) and outdoor (street) scenes using a pocket camera. The indoor images are mostly signs, doorplates, and caution plates while the outdoor images are mostly guided boards and billboards in complex backgrounds. The resolutions of the images vary from 1296 x 864 to 1920 x 1280.

No dataset is required for the computation of PPG as it is performed by processing images obtained on the fly and requires no training.

5 Results

5.1 OCR

Training of the OCR model : -
The picture below captures the training loss and validation loss vs the training steps. As only fine-tuning is performed, the training is completed relatively faster than what it would have required if we had to train the entire model architecture.

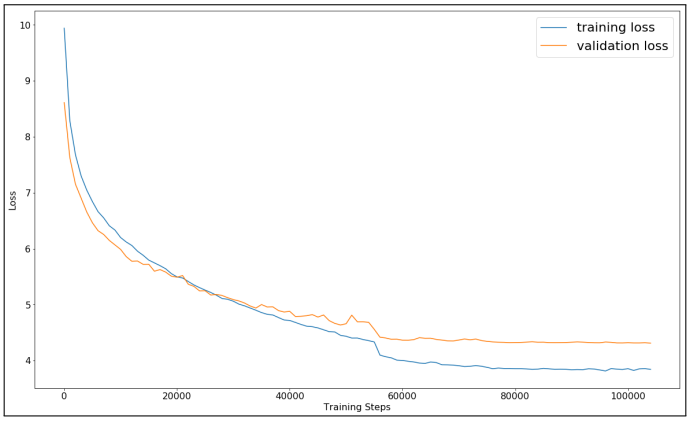
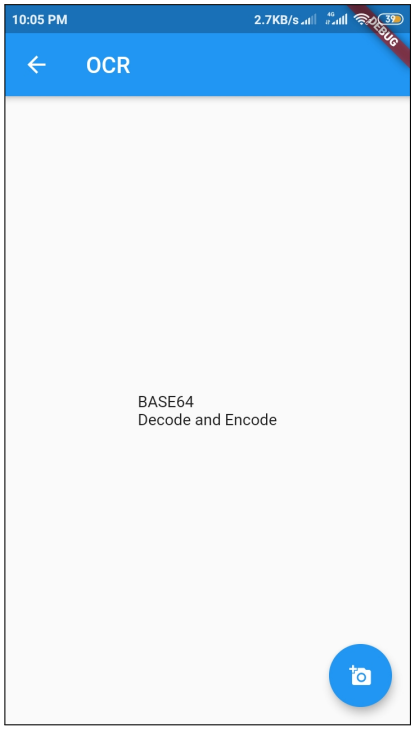


Figure 7: Loss vs Training steps

To make use of the OCR functionality of the application, click on the button at the bottom right of the screen to upload an image. The algorithm runs in the backend and extracts the text from the image. Below is an example of the same. Here, we have an input image with text against a background. As you would notice, the application is correctly able to process the image to obtain the text.

(a) A random input image for OCR

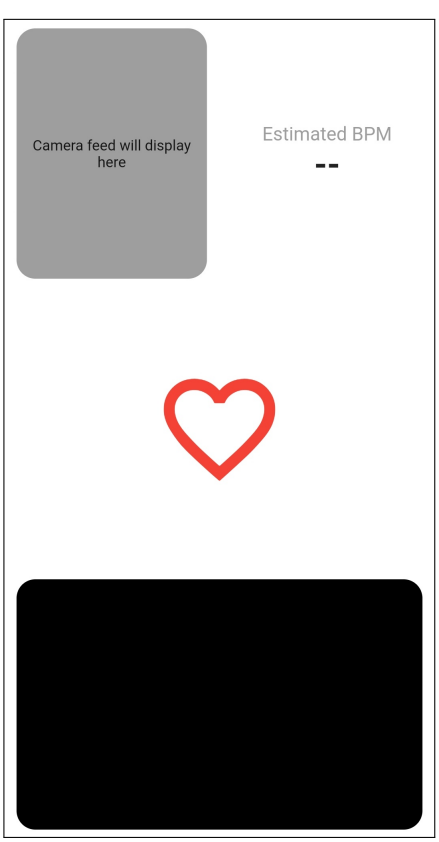


(b) Output of image (a) in OCR

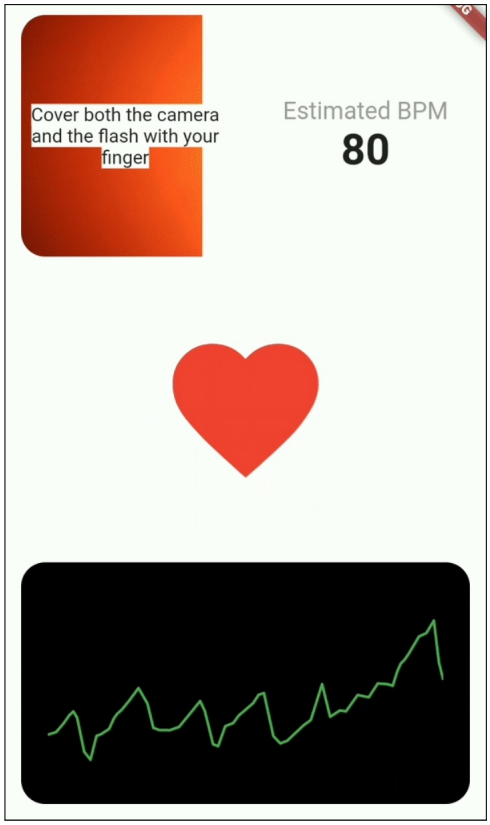
Figure 8: For OCR

5.2 PPG

To make use of this functionality, you have to first click the heart icon at the middle. Having clicked the icon, place your finger to cover the flashlight as well as the camera. The reflected light is captured by the camera and is used by the sensors of the phone to help calculate the blood flow volume. This blood flow is depicted in the graph at the bottom. We also display the BPM (beats per minute) of the person.



(a) Start screen. Click on heart icon.

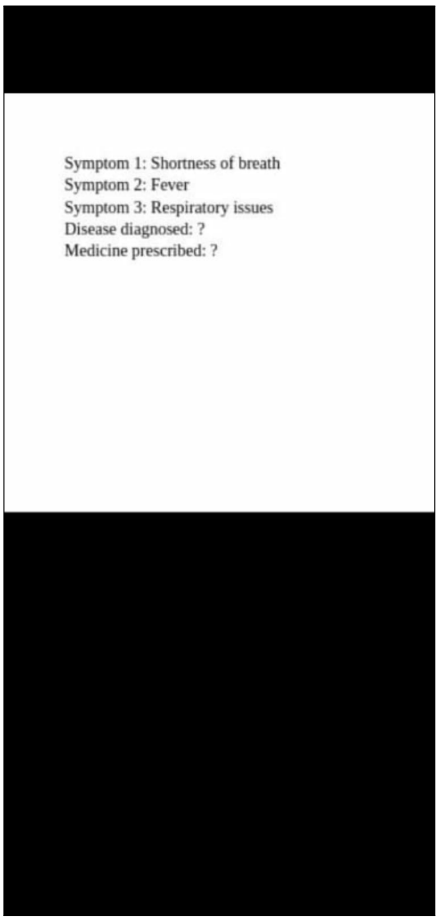


(b) Computation of heart rate and blood flow.

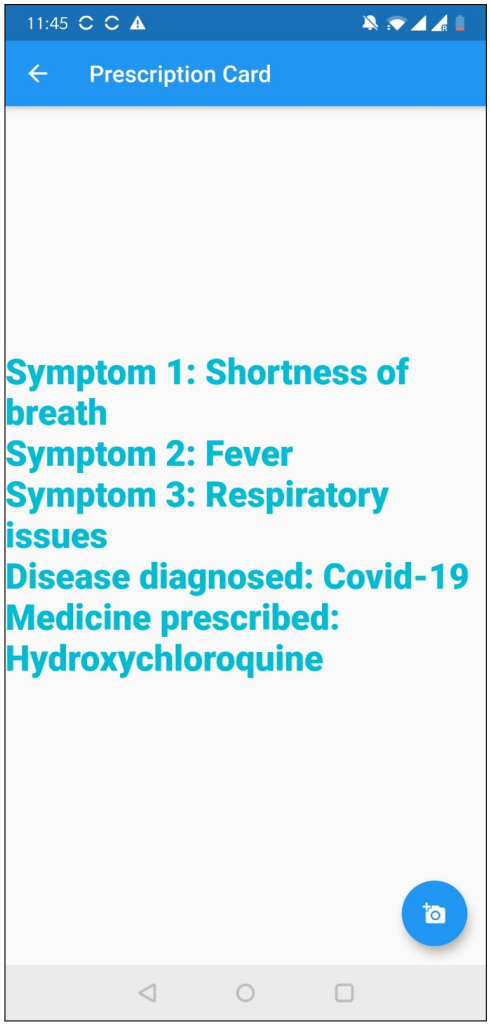
Figure 9: For PPG

5.3 Prescription Card

In this functionality, the user uploads the image of a written text containing the symptoms they are facing. This is read by the application and the application fills any missing values such as the disease diagnosed or the medicine to be prescribed using the data that has been collected. The images below show its working.



(a) Query



(b) Output.

Figure 10: For PPG

6 Direction to use the App

1. Download the app.
2. Give permission which required
3. Click on OCR
    a. Upload an image which has some text.
    b. Then it will generate the estimated texts and show on screen.
    c. We can upload another image also.
    d. Then go back to main menu
4. Click on PPG
    a. Click on the blank heart tab.
    b. It will automatically switch on the flash if permission is granted.

c. Put the tip of the finger on the camera and flashlight.

d. It will show estimated BMP and its graph.

5. Click on Prescription Card

   a. You have to upload the prescription cards.

   b. If the selected card has missing values, it checks the backend database for the closest match and fills the missing values.

   c. If it does not have any missing values and does not conflict with any other card, it will be uploaded in the database.

   d. Then go back to main menu

7 Conclusion

To summarize our work, we implemented a photoplethysmogram (PPG) that computes the blood flow volume. Using this data, we can compute the heart rate of a person. Infact, this is the technology used by leading phone applications and wearable devices. More sophisticated algorithms can also be used to check for diabetes.

Further we implemented Optical Character Recognition (OCR). For this, we used 2 algorithms. The first one was based on traditional techniques but failed to produce promising results. Hence, we shifted to a deep learning model and used transfer learning to train it on the MSRA-TD500 which is a collection of images taken from a camera. We use this model to recognize the characters.

We further extend the scope of OCR to implement a Prescription Card. This takes certain information in fixed format and fills the missing values by referring to the backend database as a look-up table. One intriguing use case for patients is to find out which medicine they need to purchase given certain symptoms. For future work, this can be extended to send a notification message to vendors when a patient scans the image of their medicine cabinet and the application automatically identifies the missing medicines.