Group 07: Pandavas

22111007, 22111014, 22111020, 22111033, 22111048
{Allan Robey, Avnish Tripathi, Divyesh Tripathi, Kush Shah, Pulkit Sharma}
Indian Institute of Technology Kanpur (IIT Kanpur)

Abstract

The title of our project is **CardioVision: Non-invasive Contact-less Heart rate detection under adversarial conditions**. This name combines "cardio," which refers to the heart, with "vision," which emphasizes the use of video data to detect the heart rate. The name suggests that the proposed solution will provide a new way of looking at the heart's activity.The project focuses on three key areas: real-time low light video enhancement, real-time video motion handling and automated ROI detection mechanism, and real-time heart rate measurement. To address the challenges associated with measuring heart rate in real-time using different techniques, the project has attempted multiple approaches such as OpenCV, MirNet, Zero-DCE, and VGG16. The project aims to identify the most effective method that provides accurate measurements and reduces computational complexity and inference time required for real-time heart rate detection. The proposed solution leverages deep learning techniques to efficiently handle motion artifacts and automated ROI detection for heart rate detection in real-time videos. A VGG16 based model has been trained on a large and diverse dataset of forehead images to enable more robust and accurate detection of foreheads under the effect of motion artifacts such as head movements. Based on the results obtained from the project, it is evident that some methods are not suitable for medical applications due to high inference time or high mean squared error, while others have limitations in capturing ROI during motion. However, the VGG16-based model has shown promising results and outperforms OpenCV in terms of real-time heart rate measurement, especially in challenging scenarios when the object is in motion. In summary, CardioVision offers a new way of accurately detecting heart rate under adverse conditions, which can potentially be used in medical applications.

1 Introduction

The project aims to develop a deep learning-based system for contactless heart rate detection from real-time video, even under adverse conditions. Traditional methods for measuring heart rate require physical contact with the user, which can be uncomfortable and inconvenient. While non-invasive heart rate measurement methods are available, they may not be performed in real-time or may be affected by light, skin-tone, and motion in videos, making it challenging to achieve accurate heart rate measurements. Hence, the project aims to provide a non-invasive and user-friendly solution for heart rate detection, accurately detecting heart rates from real-time video data captured using various devices such as smartphones, webcams, and security cameras.

Detecting heart rates from real-time video streams presents several challenges. Firstly, the video data may have low-light conditions, which can affect the hue change measurement process and impact the accuracy of heart rate detection. Additionally, the video data may contain motion artifacts that make it difficult to detect the region of interest (ROI) accurately. Accurate detection of ROI is crucial for real-time heart rate measurement, and any inaccuracies may affect the accuracy of the heart rate detection process. Therefore, the developed system will handle such adverse conditions and provide accurate heart rate measurements.

In summary, the project aims to develop a non-invasive, user-friendly solution for heart rate detection using real-time video data. The system will use deep learning-based techniques to accurately detect heart rates even under adverse conditions, such as low-light and motion artifacts in the video stream. Accurate detection of ROI will be a crucial component of the heart rate detection process, which the system will handle efficiently to provide accurate heart rate measurements.

Related work: We have conducted a literature review and identified research gaps in non-invasive heart rate detection from real-time video. Our project addresses these gaps using deep learning techniques, differentiating it from existing implementations.

Proposed Idea: The proposed idea outlines the various methods we have employed to fill the research gaps we identified during our literature review.

Methodology: The methodology section describes the datasets created and used for training our model, as well as the methods employed for low-light enhancement, motion artifact handling, and automated ROI detection. It also discusses the hyperparameters, such as the loss function, used in our model and the experimental setup for training and testing our solution.

Results: The results section outlines how our approach effectively addresses the research gaps identified by discussing the differences in how our various techniques resolve issues such as real-time low-light enhancement, motion artifacts in videos, and ROI detection.

Disscussion and Future Work: In the discussion and future work section, we propose future plans to modify our model for improved results.

Individual Contributions: The individual contributions section highlights the contributions made by each team member in the development of the project.

2 Related Work

1. Heart Rate Measurement Using Face Detection in Video – This paper presents a real-time heart rate measurement system that uses face detection and object tracking techniques to estimate heart rate from live video streams. The system has a reduced computational time but is limited by low-quality videos and the distance between the camera and object affecting the accuracy.[1]

2. Real Time Video based Heart and Respiration Rate Monitoring - This paper proposes a real-time method for heart rate and respiration rate monitoring based on the change in the hue channel in the HSV color space. However, it does not investigate the impact of different skin colors, lighting conditions, and motion artifacts on the proposed technique.[2]

3. Real-Time Webcam Heart-Rate and Variability Estimation with Clean Ground Truth for Evaluation - This paper proposes an unsupervised method for rPPG analysis that achieves high accuracy on several public datasets but faces challenges in extreme cases such as bright or flickering lighting and large head and body movements. Additionally, HR analysis during high facial arousal was marginally challenging.[3]

4. Effects of Lighting and Window Length on Heart Rate Assessment through Video Magnification - This paper explores the accuracy of contactless heart rate measurement using Video Magnification and found that lighting condition and time window duration affected the accuracy of heart rate detection. The study also found that participant movement across videos affected the accuracy of heart rate detection via Video Magnification, suggesting that the method may not be reliable in real-world scenarios where subjects may not remain still.[4]

5. EnlightenGAN: Deep Light Enhancement without Paired Supervision - This paper proposes a GAN-based approach for low-light image enhancement without paired supervision, generating visually pleasing and realistic images with better texture and color information compared to traditional methods such as Retinex and gamma correction.[5]

6. LEIS: A Low-Light Image Enhancement System using Generative Adversarial Networks - This paper proposes a GAN-based approach to low-light image enhancement, effectively preserving the details and edges of the images while reducing noise and enhancing brightness. The proposed method outperforms traditional image enhancement methods, such as gamma correction and histogram equalization, in terms of both visual quality and objective metrics.[6]

7. Multi-level Attention Network for Low-light Image/Video Enhancement" by W. Ren et al - This paper proposes a multi-level attention network that integrates global and local features for low-light image and video enhancement. The proposed method employs a GAN-based adversarial loss to produce visually pleasing results while preserving image details.[7]

8. Assessment of ROI Selection for Facial Video-Based rPPG - This paper investigates the selection of the region of interest (ROI) for accurate heart rate measurement in remote photoplethysmography (rPPG) as the thickness of the skin affects the result. The study concludes that forehead and cheeks provide more accurate results.[8]

9. Deep Video Stabilization with Multi-Grid Warping Transformation Learning - This paper proposes a deep learning-based video stabilization method that learns to perform multi-grid warping transformation for frame alignment. However, the method may not work well on videos with extreme motion or large camera shakes.[9]

10. Heart Rate Measurement Using Facial Videos - This paper implements a methodology for measuring heart rate using a person's facial image. The BBHE technique is applied to minimize low light effects, which may also introduce artifacts and noise.[10]

11. Contact-Less Heart Rate Detection in Low Light Videos - This paper proposes a new approach using a convolutional neural network to analyze time-series color variation data for autonomous heart rate monitoring in low-light conditions. The proposed method has been compared to a heuristic signal processing approach and works well even in low-light conditions.[11]

12. Learning Video Stabilization Using Optical Flow by Yanchao Yang, Deqing Sun, Huaizu Jiang, and MingHsuan Yang (ICCV 2017): This paper proposes a video stabilization method that learns to estimate the camera motion using optical flow. The method includes a CNN-based motion estimation module and a stabilization module that uses the estimated motion to stabilize the frames.[12]

13. Low-Light Video Enhancement Using Generative Adversarial Networks With Channel Attention by Yang Li, Shangwen Liang, and Shiqi Wang (IEEE Access 2019): This paper proposes a GAN-based video enhancement method that includes a channel attention mechanism to improve the contrast and color balance of low-light videos. The method includes a generator network that produces enhanced frames and a discriminator network that distinguishes between the enhanced frames and the ground truth frames.[13]

14. Heart Rate Measurement Combining Motion and Color Information (2022): The paper Combining Motion and Color Information for Heart Rate Estimation Using RGB Camera proposes a method to estimate heart rate using an RGB camera. The method combines motion and color information, achieving an average absolute error of 2.68 BPM and a correlation coefficient of 0.86 between the estimated and ground truth heart rates.[14]

15. Heart rate prediction from facial video with masks using eye location and corrected by convolutional neural networks: The paper proposes a HR detection method that combines traditional methods with deep learning to solve the problem of lack of facial information and unstable output. They also designed a method to create a mask dataset to test the effectiveness of their algorithm and concluded that their proposed algorithm is effective on test datasets.[15]

Research Gaps

1. The current methodologies for heart rate detection in low-light videos lack real-time performance and fail to provide satisfactory results. There is a need for improved techniques to address this issue.

2. Existing methods for heart rate detection in real-time videos often do not consider the impact of motion on accuracy or yield inadequate results. Novel approaches are required to improve the accuracy of heart rate detection in the presence of motion.

3. Although Face Detection combined with object tracking, OpenCV, or segmentation are commonly used for Region of Interest detection in research, a deep-learning based model specialized in forehead detection may provide superior performance. Deep learning models can learn intricate patterns and features from large datasets, enabling them to generalize well to new data and perform well in complex environments, making them a promising approach for healthcare applications.

Differences in the project from existing implementations / Novelty in the project.

1. In our project, we address the existing gaps in real-time heart rate detection by providing considerable performance under the adverse conditions of low-light and motion in video. While the current methods fail to provide significant accuracy under such conditions, our project uses advanced techniques to ensure reliable and accurate heart-rate measurement in real-time.

2. We have developed a unique model that uses VGG16 as the base feature extractor, which effectively handles motion in videos and automatically determines the ROI accurately even under the impact of large motions such as head movements, exercise moments, etc. This is a novel approach that has not been implemented in existing methods.

3. Furthermore, we have created a dataset specifically designed and labelled for forehead co-ordinates at various orientations in an image, which is a significant contribution to the field as such a dataset did not exist previously. Also, no deep learning model with an IOU for 0.68 is specially designed for forehead detection, and we have developed a model that addresses this gap in the literature.

4. To enhance the existing LOL dataset, we have appended human images to it, which is an additional contribution to the literature.

5. Lastly, we have implemented the Zero-DCE network to address the issue of heart-rate detection under low-light conditions, which is another novel approach in our project. In summary, our project provides significant novelty in terms of the methods used and the contributions made to the literature.

Approach	Effect of low light	Effect of motion	Real time	Comments
combines motion and color information from an RGB camera to estimate heart rate	✗	✗	✓	issue of generalizability due to relatively small dataset of 25 subjects.
signal processing techniques to estimate the heart and respiration rates in real-time by analyzing changes in the hue channel of the HSV color space.	✗	✗	✗	did not investigate the impact of different skin colors on the accuracy
a convolutional neural network (CNN) to analyze time-series color variation data from videos	✓	✗	✓	limited generalizability of the method to populations with different ages, health conditions, and skin tones
heart rate detection method using face detection and object tracking in video streams.	✗	✗	✓	achieves high accuracies on several public datasets, but faces challenges in extreme cases such as overtly bright or flickering lighting and large head and body movements (e.g., during exercising)
unsupervised rPPG analysis method	✗	✓	✓	achieves high accuracies on public datasets, but faces challenges in extreme cases
Our Model: Zero DCE ,VGG16-Automated ROI Detection(using our created data set) capable of handling effect of motion	✓	✓	✓	**a computationally efficient method for real-time heart rate detection in videos, capable of handling adverse conditions such as low-light and motion artifacts.**

Figure 1: Difference in the project from the existing implementation

3 Proposed Idea

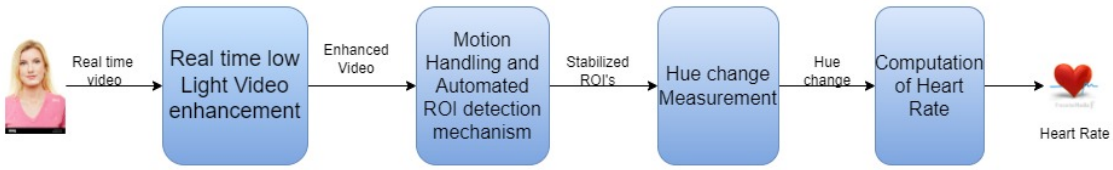


Figure 2: Pipeline of the proposed idea

1. Real-time Low light video enhancement

Heart rate detection from low-light video is a challenging task due to the adverse conditions that affect the accuracy of heart rate measurements. In our project, we have attempted to address this challenge by trying multiple approaches such as OpenCV, MirNet, and Zero-DCE. The aim was to identify the most effective method for accurately measuring heart rate from real-time low-light video.

To evaluate the effectiveness of each approach, we conducted thorough testing and performed a comparative analysis of the obtained results. Our objective was to identify a method that not only provides accurate measurements but also reduces the computational complexity and inference time required for real-time heart rate detection.

2. Real-Time Video Motion Handling and Automated ROI Detection Mechanism

In this project, we aim to develop a novel solution for real-time heart rate measurement that addresses the issues of motion artifacts and instability in video. Existing solutions such as CNNs and RNNs are computationally expensive, making it difficult to handle motion artifacts and instability in real-time heart measurement. We have attempted to address this challenge by trying multiple approaches such as OpenCV and building a model using VGG16 as the feature extractor.

Our proposed solution leverages deep learning techniques to efficiently handle motion artifacts and automated ROI detection for heart rate detection in real-time videos. We have trained a

VGG16 model on a large and diverse dataset of forehead images to enable more robust and accurate detection of foreheads under the effect of motion artifacts such as head movements. This eliminates the need for video stabilization and enhances the accuracy of our heart rate measurement.The current methods for detecting the region of interest (ROI) in videos, such as face detection with object tracking or OpenCV segmentation, are inadequate for detecting ROI in the presence of motion artifacts. To overcome this limitation, we had developed a deep learning-based model specifically designed for forehead detection.

Through thorough testing and comparison with existing solutions, we aim to confirm that our proposed solution outperforms existing solutions for handling the effect of motion artifacts and automated ROI detection in real-time heart rate detection. Our project will contribute to the advancement of this field by providing valuable insights and addressing the limitations of existing solutions for real-time heart rate measurement.

3. **Real-time Heart rate measurement**
The project aims to use a combination of forehead detection and bandpass filtering techniques to accurately measure heart rate from real-time video. The forehead detection model is loaded and used to detect the forehead in each frame of the video, and a region of interest (ROI) is extracted around it. The extracted ROI is then processed using a bandpass filter to isolate the heart rate signal, and the peaks in the filtered signal are detected to calculate the heart rate in beats per minute (BPM). The will then be tested to determine whether the method offers a more efficient and accurate way to measure heart rate in real-time video, particularly in low-light and unstable environments.

4 Methodology

4.1 Dataset

1. **(Low-light) LOL Dataset:** The LOL dataset is composed of 500 low-light and normal-light image pairs and divided into 485 training pairs and 15 testing pairs. The low-light images contain noise produced during the photo capture process. Most of the images are indoor scenes. All the images have a resolution of 400×600.
Furthermore, we have expanded this dataset with an additional 50 images of human subjects to enhance the performance of the Zero-DCE model in low-light image enhancement.

   Reason for incorporating additional 50 images of human subjects to the LOL dataset:
   Adding human images to the LOL dataset for training the Zero-DCE model can improve its performance in low-light image enhancement because it allows the model to learn more about the characteristics of human skin under low-light conditions. This can help the model to better distinguish between noise and the actual features of the skin, resulting in more accurate and reliable heart rate detection in real-time videos. By improving the quality of the low-light images, leading to more accurate heart rate measurement.

2. **Human Faces Dataset:** For our study, we utilized a human faces dataset consisting of 2000 publicly available images and captured an additional 500 images of human subjects in various orientations with varying motion artifacts using a Python script. The entire dataset comprised a total of 2455 images, out of which 2335 images were used for training, 120 for testing, and 60 for validation. To ensure proper labelling, we manually annotated all images using LabelMe, where we marked the bounding box using two coordinates - the upper left and lower bottom points.

   Reason for incorporating additional 500 images of human subjects in various orientations:
   Adding additional human images in different orientations and with varying motion artifacts can help the deep learning model to learn the facial features and variations that occur due to head movements, which may help to handle the effect of motion in videos. By training on a diverse dataset, the model may be better equipped to automatically detect the ROI in any orientation

and under the effect of motion, resulting in more accurate and reliable heart rate measurement in challenging scenarios.

4.2 Pre-Processing on the data

Pre-processing for Zero-DCE

- Initially, a function load-data, reads in an image file from a given file path decodes the image to ensure it has three color channels (red, green, and blue), resizes the image to a standard size of 256x256 pixels, and normalizes the pixel values to be between 0 and 1.
- Then using a data-generator function we take in a list of file paths for the low-light images that will be used to train the model. It creates a TensorFlow dataset object from the file paths, applies the load-data function to each image in parallel, and batches the images into groups of 16 to optimize model training.
- Finally, we define three lists of file paths for the low-light images that will be used for training, validation, and testing. It then creates two dataset objects, train-dataset and val-dataset, using the data-generator function on the respective lists of file paths for training and validation. These dataset objects will be used to train and evaluate the machine learning model.

Pre-processing for MirNet

- Define the image size: Here, a fixed image size of 256x256 pixels have been defined to make the input images compatible with the MirNet model.
- Define the batch size: This step defines the number of images that will be processed in a single forward and backward pass. We have used a batch size of 4 for the MirNet model.
- Load the low-light images: The input images (low-light images) are loaded into the program using a file path. The glob module is used to get the list of file paths of all the low-light images. These images are the ones that we want to enhance using the MirNet model.
- Load the high-quality images: In this step, the high-quality images are loaded into the program. These images are used as the ground truth for training the MirNet model. They are typically taken under normal lighting conditions and are used to compare the enhanced images generated by the model to the actual high-quality images. The file paths of the high-quality images are obtained using the same glob module as used in the previous steps.

Pre-processing for VGG16 based motion handling mechanism and automated ROI detection

The initial data processing for dataset creation involves several steps, which are explained below in detail

- Renaming images: The human faces dataset contains many images with the same name, which can create confusion during the further processing. Therefore, we have renamed the images to give them unique names. This is done to avoid any conflicts that may arise due to having the same names for different images.
- Resizing images: The next step is to resize the different sized images to the same size (640, 480). This is done to ensure that all the images have the same dimensions, making it easier to process them. The reason for choosing this particular size is that when we capture frames using cv2, it captures frames in this size. This step ensures consistency across the dataset.
- Manual labelling: After resizing, the images are labelled manually using the Labelme tool. The manual labelling is done to identify the regions of interest (ROI) in the images. In this case, the ROI is the human face, which is required for further processing.
- Data augmentation: The final step of initial data processing involves data augmentation. Data augmentation is the process of artificially expanding the size of the dataset by applying various transformations to the existing images. This is done to increase the diversity of the dataset, which can help improve the accuracy of the machine learning model. In this case, the albumentation module is used for data augmentation, resulting in a final dataset size of 23350 images for

training, 1200 images for testing, and 600 images for validation. The albumentation module provides various image augmentation techniques such as rotation, flipping, scaling, etc. These techniques can help create new images with different variations of the ROI, making the dataset more diverse.

4.3 Computer Vision algorithms and their details

4.3.1 Real-time low-light video enhancement

As part of our methodology, we aimed to tackle this challenge by experimenting with multiple computer vision algorithms, including OpenCV, MirNet, and Zero-DCE.

Using OpenCV

1. In our project, we applied histogram equalization to each color channel (red, green, and blue) separately to enhance the contrast of low-light images.

2. **Histogram equalization** is a technique that redistributes the intensity values of the pixels in an image to improve contrast. The approach works by computing the histogram of the input image, then computing a cumulative distribution function (CDF) from the histogram, and using the CDF as a transformation function to replace each pixel in the image with its corresponding transformed intensity value.

3. We initialized a capture object for the default camera and checked if it was open and ready to read frames. Inside a loop, we read a frame from the camera, extracted the red, green, and blue channels from the frame, and applied the cv2.equalizeHist() function to each channel to improve image contrast. The cv2.equalizeHist() function maps the image intensity distribution to a more uniform distribution, resulting in improved contrast.

4. The three equalized channels were merged back together using cv2.merge(), and the resulting enhanced frame was displayed in a window. This process continued until a stop condition was met.

Using MirNet

1. The MIRNet model is a deep learning architecture that uses a recursive residual design to improve representation learning in image super-resolution tasks. It features a feature extraction model that maintains high-resolution features while computing complementary features across multiple spatial scales. This allows for the preservation of precise spatial details and improved representation learning.

2. The model also includes a mechanism for information exchange that fuses features across multi-resolution branches, progressively improving representation learning. It employs a selective kernel network approach to fuse multi-scale features, dynamically combining variable receptive fields and preserving the original feature information at each spatial resolution.

3. The recursive residual design of the MIRNet model progressively breaks down the input signal, simplifying the overall learning process and enabling the construction of very deep networks. These features make the MIRNet model an effective approach to image super-resolution tasks, particularly in low-light conditions where image quality can be poor.

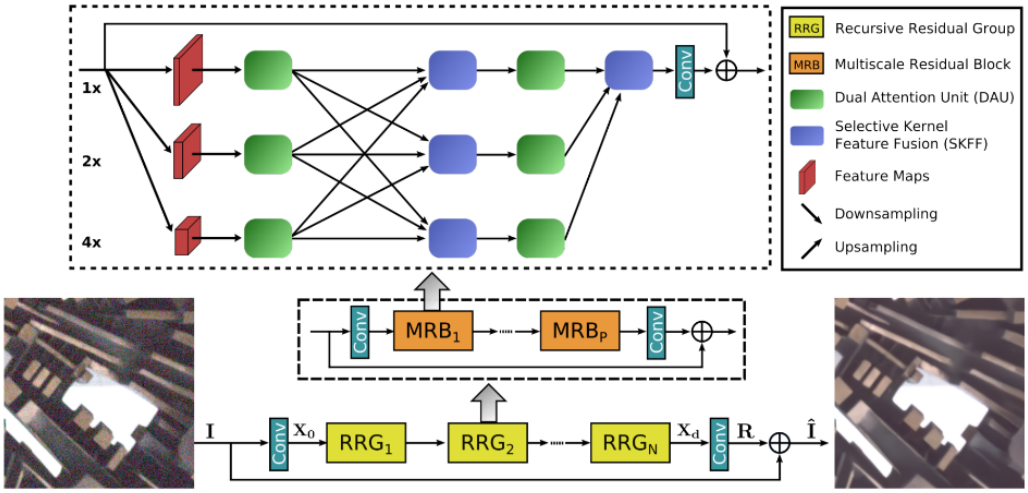


Figure 3: MIRNET Archietecture

4. **Loss function :**

   Charbonnier loss is a smooth and robust loss function commonly used in computer vision tasks, such as image restoration and super-resolution. It is defined as the mean square root of the sum of squared errors and a small constant to avoid numerical instability.

5. **Training and testing:**

   The MIRNet model is trained using the **LoL dataset**, which contains 500 image pairs consisting of low-light input images and well-exposed reference images. The model uses 300 images for training, 185 for validation, and 15 for testing.
   The **Adam optimizer** with a **learning rate of 1e-4** is used for training, with a total of **50 epochs**. The model's performance is evaluated using **Peak Signal Noise Ratio** , a metric that measures the quality of a signal by comparing the maximum possible signal power to the power of corrupting noise.

Using Zero-DCE

1. Zero-DCE is a deep neural network that enhances low-light images by estimating an image-specific tonal curve. It achieves this by training a lightweight deep network, DCE-Net, to estimate pixel-wise and high-order tonal curves for dynamic range adjustment. The output tonal curves are then used to adjust the dynamic range of the input image for enhanced image quality, while preserving the contrast of neighboring pixels. This curve estimation process is inspired by curves adjustment used in photo editing software such as Adobe Photoshop.

2. What makes Zero-DCE stand out is its relaxed assumptions with regard to reference images. It does not require any input/output image pairs during training, which is achieved through a set of carefully formulated non-reference loss functions that guide the training of the network. This makes it a useful tool for enhancing low-light images, without needing any additional reference images for comparison or training.

3. **Zero-DCE Frame work**

   The Zero-DCE framework uses DCE-Net to estimate the best-fitting light-enhancement curves (LE-curves) for a given input image. These curves are used to map all pixels of the input's RGB channels iteratively, resulting in the final enhanced image. A light-enhancement curve is a type of curve that can automatically map a low-light image to its enhanced version, with the curve

parameters being dependent solely on the input image.

When designing a light-enhancement curve, three objectives are taken into account. First, each pixel value of the enhanced image should be in the normalized range of [0,1] to avoid information loss due to overflow truncation. Second, the curve should be monotonous to preserve the contrast between neighboring pixels. Third, the curve's shape should be as simple as possible, and it should be differentiable to allow backpropagation.

The light-enhancement curve is applied separately to the three RGB channels instead of only on the illumination channel. This three-channel adjustment can better preserve the inherent color of the image and reduce the risk of over-saturation. By estimating these curves, Zero-DCE can enhance low-light images without the need for reference images, making it a valuable tool for low-light photography and image enhancement.

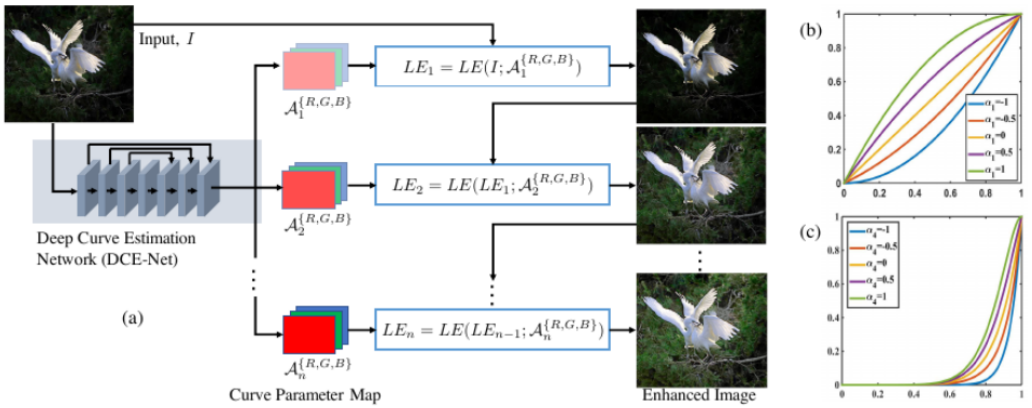


Figure 4: Zero-DCE Archietecture

4. **Loss Functions:**

   - **Color constancy loss:** The color constancy loss function calculates the color variations in an image and helps to reduce the effects of changes in lighting conditions. It works by calculating the mean RGB value of an image and then determining the color difference between the red, green, and blue channels. By minimizing this difference, the loss function helps to maintain consistent color representation in the enhanced image.
   - **Exposure loss:** Exposure loss is a loss function used in image processing to adjust the overall brightness of an image. It measures the difference between the average pixel intensity of the input image and a predefined target value, usually 0.6, and aims to minimize this difference during training.
   - **Illumination smoothness loss:** Illumination smoothness loss is a term used in image enhancement to maintain the smoothness of the illumination across the image. It measures the difference between adjacent pixels in an image and ensures that there is no sudden variation in illumination. The loss function is calculated by taking the sum of squared differences in horizontal and vertical directions and then normalizing it by the total number of pixel pairs.
   - **Spatial consistency loss:** Spatial Consistency Loss is a custom loss function used for image enhancement tasks. It measures the difference in spatial consistency between the original and enhanced images by computing the mean square error of the image gradients in four directions (left, right, up, down).

5. **Training and Testing :**

   The Zero-DCE low light model is trained using the **LOL dataset** which includes 500 low light

images. Additionally, 50 more images were added to the dataset. The dataset was split into training and testing sets with 485 and 15 images, respectively, for the original LOL dataset and 40 and 10 images for the additional dataset. We use 300 low-light images from the LOL Dataset training set for training, and we use the remaining 185 low-light images for validation and 30 image from our own additional dataset for training ,10 images for validation. The model was trained for **100 epochs** with a **learning rate of 0.001**.

4.3.2 Real-time Motion Handling Mechanism and Automated ROI Detection

Using CV2 for Video Stabilization

1. The OpenCV library provides a function called cv2.VideoCapture() that can be used to capture frames from a camera or a video file. Once we have captured a frame, we can use it to stabilize the video.

2. To do this, we first need to initialize some variables. These include prev-frame, prev-pts, max-pts, stability-threshold, and max-unstable-frames. prev-frame is the previous frame that we will use to stabilize the current frame. prev-pts are the points that were tracked from the previous frame. max-pts is the maximum number of points that we want to track. stability-threshold is the threshold below which we consider a point to be unstable. max-unstable-frames is the maximum number of unstable frames we allow before resetting prev-frame and prev-pts.

3. Next, we use optical flow to track points between the previous frame and the current frame. We then filter out any unstable points based on our stability threshold. If the current frame is considered unstable for a number of consecutive frames (based on max-unstable-frames), we reset prev-frame and prev-pts.

4. Once we have identified stable points in both the previous and current frames, we estimate an affine transform between them using cv2.estimateAffinePartial2D(). We then use this transform to stabilize the current frame using cv2.warpAffine().

5. Finally, we display both the original and stabilized frames using cv2.imshow().

Using OpenCV based ROI Detection

1. The Haar Cascade classifier is created using the 'haarcascade-frontalface-default.xml' file. This classifier is trained to detect frontal faces in images or videos.Then, we initialize a video capture object by calling the 'VideoCapture' function and passing it the index of the camera. In this case, '0' is passed, which refers to the default camera on the device.

2. The while loop is initiated to read frames from the camera until it is stopped by the user.Inside the while loop, the 'cap.read()' function is used to read a frame from the video capture object. The 'ret' variable returns a boolean value indicating whether the read was successful, and 'img' variable contains the actual image.If the 'ret' value is false, meaning the video capture object is not able to read the frame, the loop is exited.

3. The next step is to convert the color image to grayscale using the 'cv2.cvtColor' function.

4. The 'face-cascade.detectMultiScale' function is used to detect the faces in the grayscale image. It takes the grayscale image as input, and two parameters - 'scaleFactor' and 'minNeighbors'. These parameters control the sensitivity and accuracy of face detection. The function returns a list of faces in the form of rectangles.

5. The loop iterates over each face detected and calculates the position of the forehead based on the face coordinates. The 'forehead-y' variable is set to 0.25 times the height of the face rectangle, and the 'forehead-h' variable is set to 0.2 times the height of the face rectangle.

6. A green rectangle is drawn around the forehead using the 'cv2.rectangle' function. The rectangle is drawn on the original color image, not the grayscale one.

7. Finally, the 'cv2.imshow' function is used to display the image with the forehead rectangle.The video capture object is released and all windows are destroyed using 'cv2.destroyAllWindows'.

8. It is important to note that this approach is unable to reliably detect the forehead under the effect of motion in videos.

Using our own VGG16 based Model

In this project, we have developed a model capable of accurately detecting and localizing a region of interest (ROI) in images even under the effect of motion artifacts. Thus, eliminating the need of different models to handle the impact of motion in videos and ROI(forehead) detection.

```
Model: "vgg16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, None, None, 3)]   0

 block1_conv1 (Conv2D)       (None, None, None, 64)    1792
 block1_conv2 (Conv2D)       (None, None, None, 64)    36928
 block1_pool (MaxPooling2D)  (None, None, None, 64)    0

 block2_conv1 (Conv2D)       (None, None, None, 128)   73856
 block2_conv2 (Conv2D)       (None, None, None, 128)   147584
 block2_pool (MaxPooling2D)  (None, None, None, 128)   0

 block3_conv1 (Conv2D)       (None, None, None, 256)   295168
 block3_conv2 (Conv2D)       (None, None, None, 256)   590080
 block3_conv3 (Conv2D)       (None, None, None, 256)   590080
 block3_pool (MaxPooling2D)  (None, None, None, 256)   0

 block4_conv1 (Conv2D)       (None, None, None, 512)   1180160
 block4_conv2 (Conv2D)       (None, None, None, 512)   2359808
 block4_conv3 (Conv2D)       (None, None, None, 512)   2359808
 block4_pool (MaxPooling2D)  (None, None, None, 512)   0

 block5_conv1 (Conv2D)       (None, None, None, 512)   2359808
 block5_conv2 (Conv2D)       (None, None, None, 512)   2359808
 block5_conv3 (Conv2D)       (None, None, None, 512)   2359808
 block5_pool (MaxPooling2D)  (None, None, None, 512)   0

=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
_________________________________________________________________
```

Figure 5: Model Summary

1. **Dataset Preparation**
   To train and test our model, we created a dataset of 2455 images and manually labeled them with bounding box annotations using LabelMe. Additionally, we used a Python script to capture an additional 500 images with varying motion artifacts and orientations. We then augmented

the dataset resulting in a final dataset size of 23350 images for training, 1200 images for testing, and 600 images for validation. The dataset was split into training, testing, and validation sets with the sizes of 23350, 1200, and 600, respectively.

2. **Model Architecture**
   We used the VGG16 model as a feature extractor, and the output of the last convolution layer with dimensions of None,None,None,512 was used as input to the subsequent layers. A Global Max Pooling 2D layer was employed to obtain a feature vector of dimensions [512,1]. The extracted features were fed into two separate models: a classification model and a regression model.

3. **Classification Model**
   The output of the Global Max Pooling 2D layer was fed into a classification model, which consisted of 2048 dense nodes with a sigmoid activation function. The goal of the classification model was to determine whether an object was present in the image or not. The sigmoid activation function produced output in the form of 0 and 1 classes, which can be interpreted as the confidence level of the model in its prediction.

4. **Regression Model**
   The feature vector obtained from the Global Max Pooling 2D layer was also fed into a regression model. The regression model consisted of 2048 dense nodes, and its output was in the form of four co-ordinates: the center co-ordinates of the bounding box, as well as its height and width. The goal of the regression model was to accurately localize the ROI within the input image by predicting its bounding box co-ordinates.

5. **Hyperparameters**
   We set the hyperparameters to Batch Size = 8, Epoch = 50, and Learning Rate = 0.0001, and used the Adam optimizer. These values were chosen through experimentation and tuning.

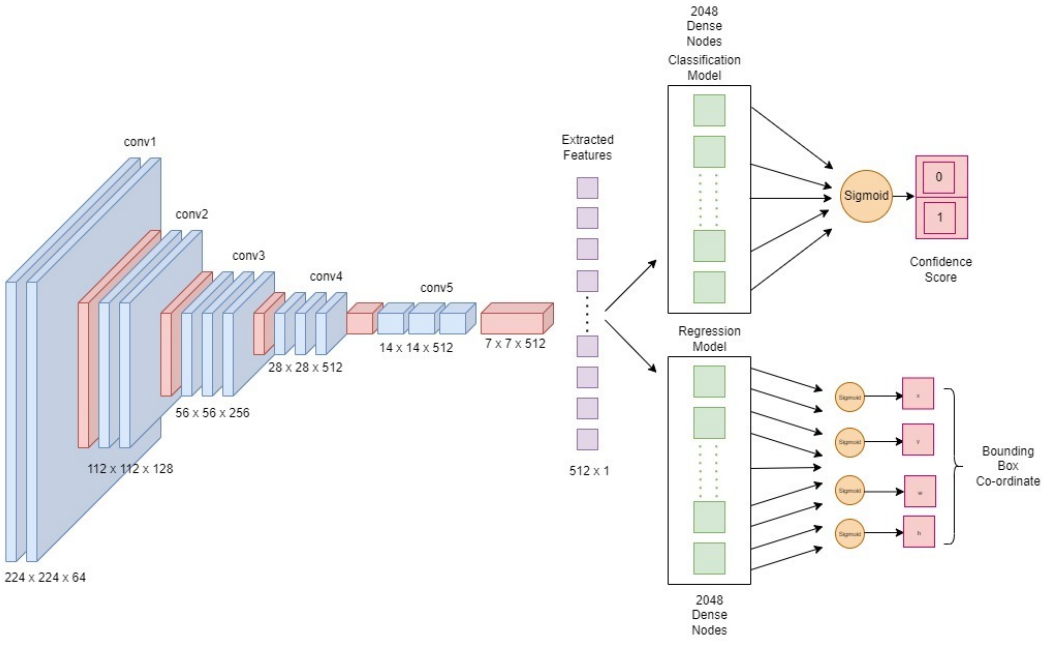


Figure 6: VGG16 based Architecture for motion handling and automated ROI detection

Thus, we developed a model to accurately detect and localize a region of interest in images even under the effect of motion artifacts. Our approach used the VGG16 model to extract features from input images, which were then passed through separate classification and regression models to produce the

desired output. By utilizing these models, our approach was able to accurately detect the ROI even under the effect of motion.

IOU: 0.682

Reason for not incorporating the dense layers of VGG16: In deep learning models, dense layers are computationally expensive, which means that they can slow down the prediction time of the model. This is because dense layers have a large number of parameters that need to be learned during the training process. During prediction, the model needs to perform calculations on these parameters to produce the final output.

In the given approach, there are two separate models, a classification model and a regression model, both of which use dense layers. By reducing the number of dense layers used in these models, the overall prediction time of the model can be reduced.

However, reducing the number of dense layers may come at a cost of reduced accuracy in the model's predictions. This is because dense layers are able to capture complex non-linear relationships between the input features and the output predictions. By reducing the number of dense layers, the model may not be able to capture these relationships as effectively, resulting in a decrease in accuracy.

Therefore, a trade-off needs to be made between prediction time and accuracy when deciding on the number of dense layers to use in the model. If prediction time is a critical factor, reducing the number of dense layers may be a viable option. However, if accuracy is of utmost importance, it may be necessary to use more dense layers to capture the necessary complexity in the model.

Also, VGG16 was originally designed for image classification tasks with 1000 output classes, which meant that the last layer had 1000 neurons for classification. However, in our case, we only needed to detect a single class, which is the presence of an object in a specific region of interest. Using a model with 1000 output classes and neurons in the last layer would not be an optimal solution as it would require unnecessary computation and training time. Therefore, developing our own model that is tailored to our specific task was a more efficient and effective approach.

4.3.3 Real-time Heart rate measurement

1. **Load the forehead detection model:** The code loads the trained model for detecting foreheads from the 'forehead.h5' file using load-model function from TensorFlow Keras.

2. **Capture video frames:** The code captures video frames using the default camera by creating a cv2.VideoCapture object.

3. **Detect the forehead:** The code uses the loaded forehead detection model to detect the forehead in the captured video frames. If the forehead is detected, the code computes the bounding box around it.

4. **Extract the ROI:** The code extracts a square region of interest (ROI) from the captured frame, centered on the forehead point.

5. After extracting the region of interest (ROI) from the video frame, we applied a band-pass filter to isolate the heart rate signal. The band-pass filter is designed using the butter function from SciPy, and it passes only the frequencies in a specific range (low-cutoff to high-cutoff). Then, the filtered signal is centered around its mean value to remove any DC offset. Finally, the code uses the find-peaks function to detect the peaks in the signal, which correspond to the heartbeats.

6. Then we computed the heart rate in beats per minute (BPM) by measuring the time between two consecutive peaks in the signal. Specifically, it divides 70 by the time difference (in seconds) between the two consecutive peaks and multiplies it by the number of samples per second (buffer-size/sample-rate) to get the BPM value.

7. **Formula for calculating heart rate from signal:**
The formula for calculating heart rate (BPM) using the time difference between consecutive peaks (Δt) and sampling frequency (Fs) is:

$$BPM = \frac{70}{\Delta t \times Fs} \tag{1}$$

where Δt is the time difference (in seconds) between two consecutive peaks, and F_s is the sampling frequency (samples per second). The constant 70 comes from the fact that the average heart rate for adults is around 70 beats per minute.

4.4 Experimental Set-Up

- The heart rate detection experiment was conducted on an HP Pavilion laptop equipped with a built-in webcam. The webcam was capable of capturing 30 frames per second at a resolution of 640*480 pixels. The implementation was done in Python version 3.9.12.
- An initial delay of 2 seconds was required for HR estimation due to the duration of each heartbeat, which ranges from 0.45 to 1.25 seconds. The average of all detected HR within the last second was considered after the initial estimation, and the stable values were obtained by averaging the 20 most recent HR.
- The heart rate detection results were compared with those obtained from two contact-based devices: a Samsung Watch and a Boat Watch 6.

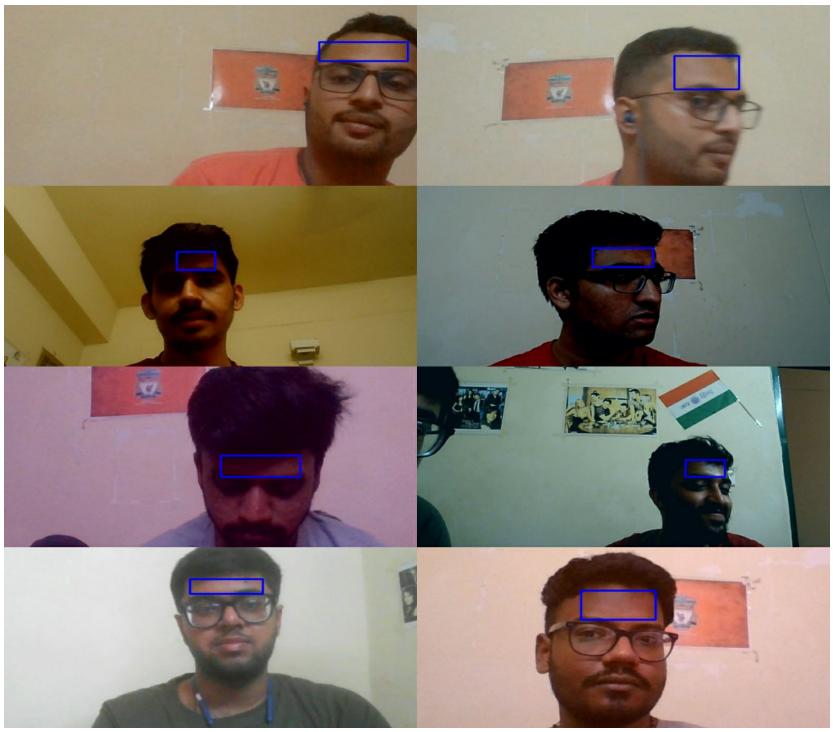
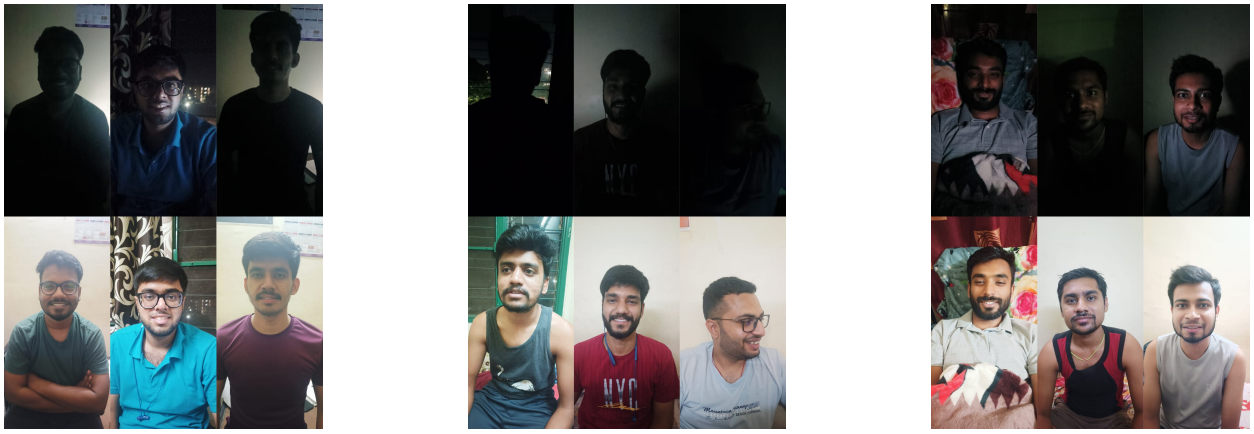


Figure 7: Data set for motion handling and Automated ROI detection



(a) (b) (c)

Figure 8: Data set low low light enhancement

4.5 Implementation of the Project

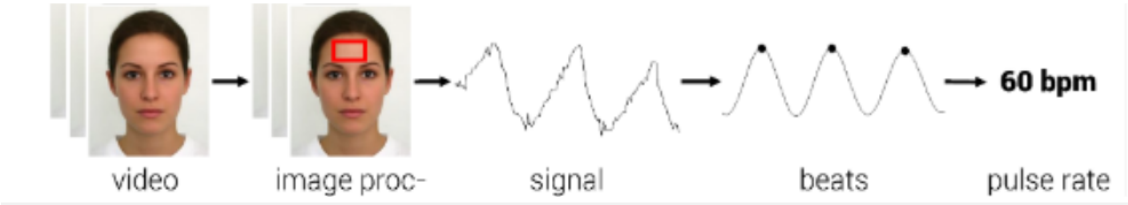


Figure 9

To overcome the challenge of low-light video, our methodology involved experimenting with multiple computer vision algorithms, namely OpenCV, MirNet, and Zero-DCE. We compared the accuracies of these approaches and their corresponding inference times, and ultimately selected Zero-DCE due to its high accuracy and intermediate inference time.With respect to motion handling and automated ROI detection, our VGG16 based model gave superior performance by accurately detecting ROI under different orientations.

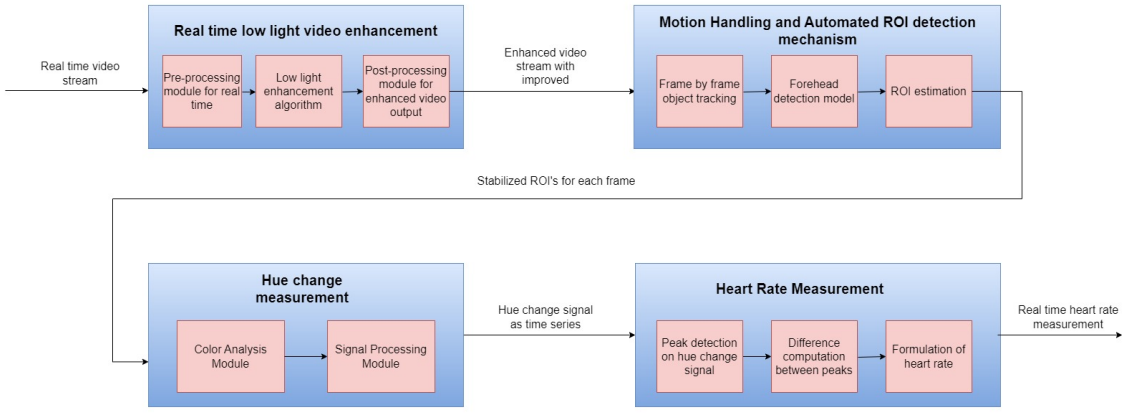


Figure 10: Pipeline of the Project

1. Real-time low-light video enhancement system includes a pre-processing module, a low-light video enhancement algorithm like Zero-DCE, and a post-processing module. it receives Real-time video stream as input . The pre-processing module resizes, crops, and normalizes the

real-time video stream input, which is then passed to the low-light video enhancement algorithm. The algorithm enhances the video frames using deep convolutional neural networks, and the post-processing module finalizes the output by performing color correction, contrast adjustment, and denoising. The system produces an enhanced video stream that offers improved visibility in low-light conditions, making it useful for applications such as surveillance, security, and night time photography.

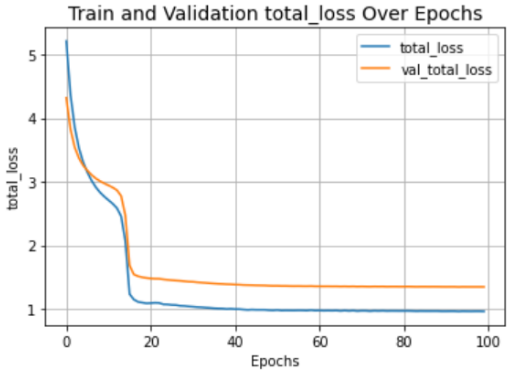
2. The real-time motion handling mechanism and automated ROI detection component includes an object tracking algorithm, a forehead detection model, and an image processing module. The object tracking algorithm, such as KCF or MOSSE, is used for motion handling, allowing the system to track objects in real-time. The forehead detection model, based on VGG16 architecture, detects the region of interest (ROI) in each frame. The image processing module stabilizes the ROIs to reduce unwanted motion and improve the visual quality of the output. The input for this component is an enhanced video stream, and the output is a stabilized ROI for each frame.

3. The hue change measurement component consists of a color analysis module and a signal processing module. The color analysis module measures the hue change in the stabilized ROIs for each frame using techniques such as color histogram analysis. The signal processing module then extracts the heart rate signal from the hue change signal using methods like Fourier transform. The input for this component is the stabilized ROIs for each frame, and the output is a heart rate signal as a time series.

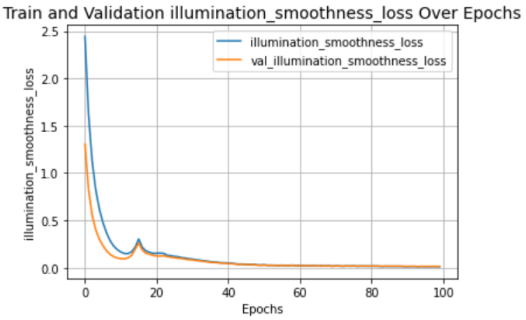
4. The heart rate detection component includes a heart rate measurement algorithm and a display module. The heart rate measurement algorithm, which can use peak detection on the heart rate signal, analyzes the time series of the heart rate signal to determine the frequency and amplitude of the heartbeats. The display module then shows the real-time heart rate measurement results to the user. The input for this component is the heart rate signal as a time series, and the output is a real-time heart rate measurement

5 Results

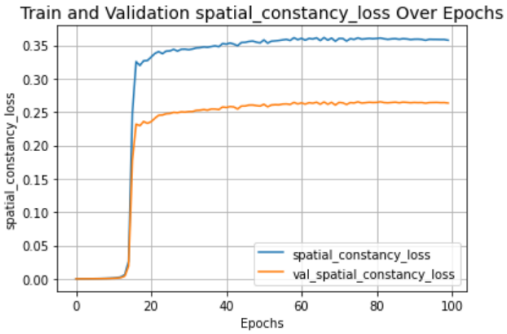
5.1 Zero-DCE Train and Validation Loss over Epochs



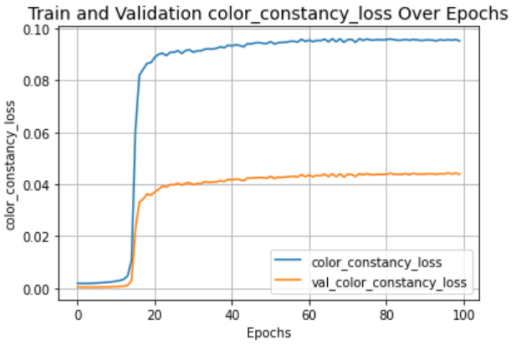
(a)



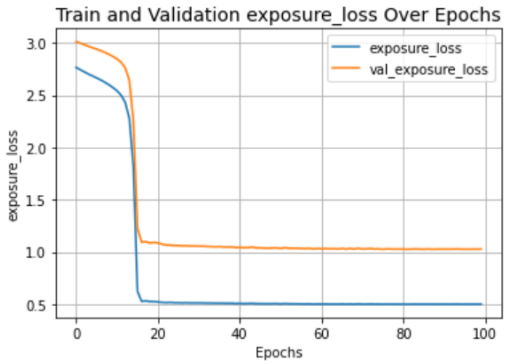
(b)



(c)



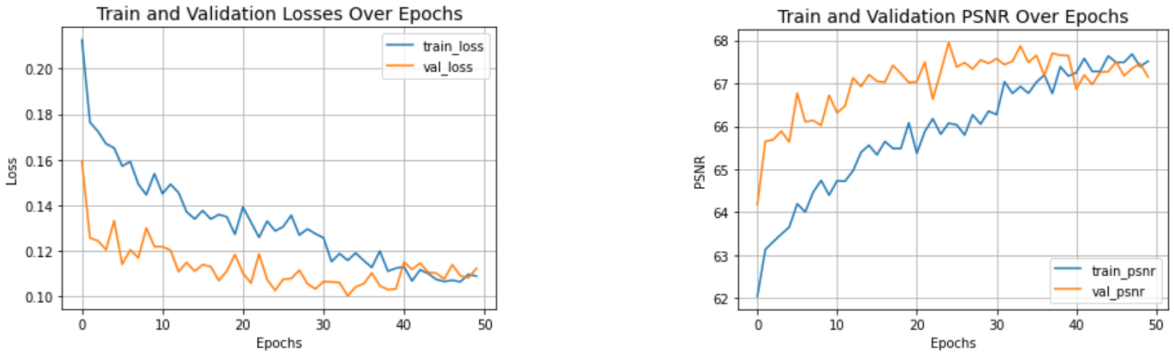
(d)



(e)

Figure 11: Zero-DCE Train and Validation Loss over Epochs

5.2 MirNet Train and Validation Loss over Epochs



(a) (b)

Figure 12: MirNet Train and Validation Loss over Epochs

5.3 Performance of different approaches for low light enhancement

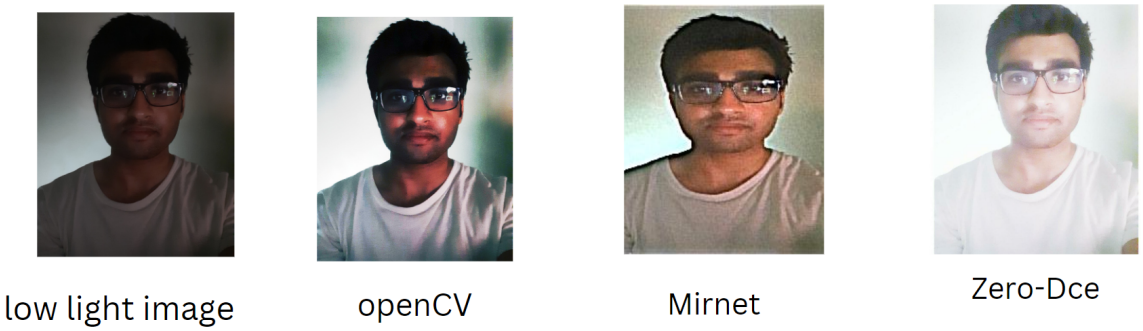


Figure 13: Performance of different approaches for low light enhancement

Thus, from the above figure it can be concluded that Zero-DCE gives better performance.

5.4 Performance of different approaches for Motion Handling

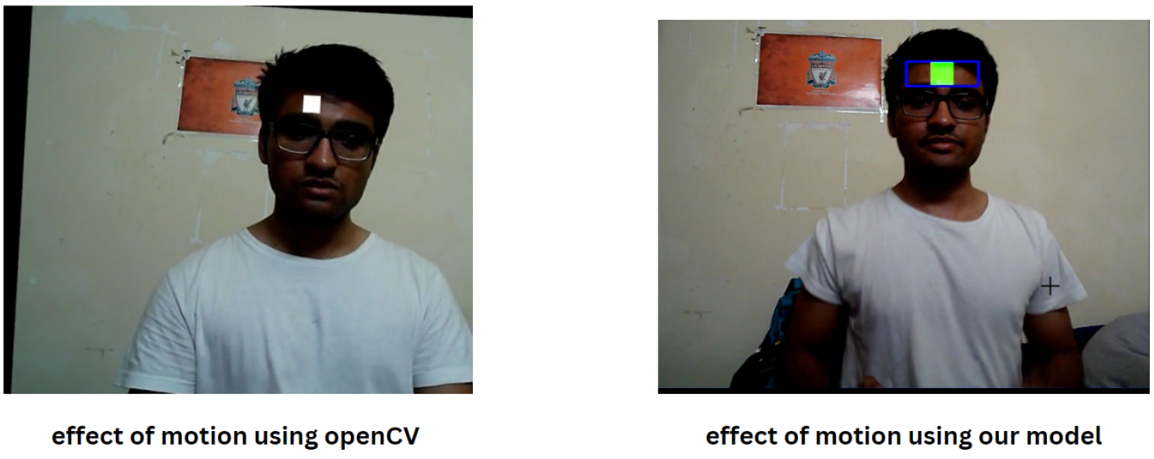


Figure 14: Performance of different approaches for Motion Handling

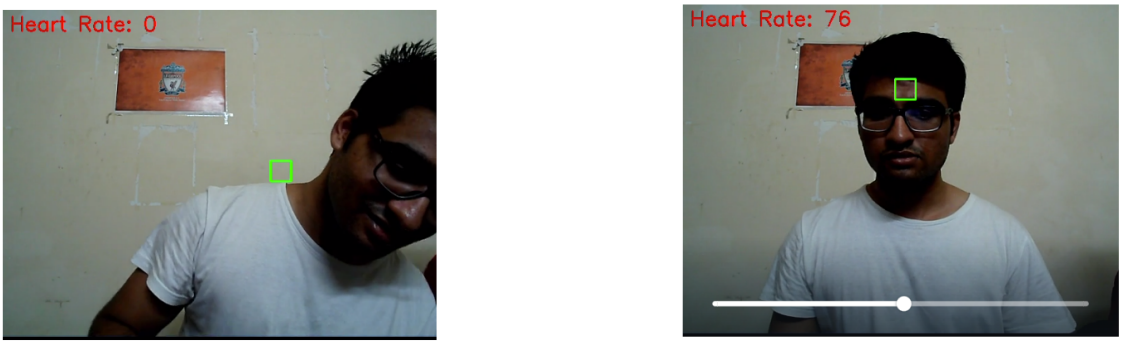
Thus, our VGG16 based model is able to detect ROI and handle the effect of motion under different orientations.

5.5 Performance of different approaches for ROI Detection

Figure 15: Performance of different approaches for ROI Detection

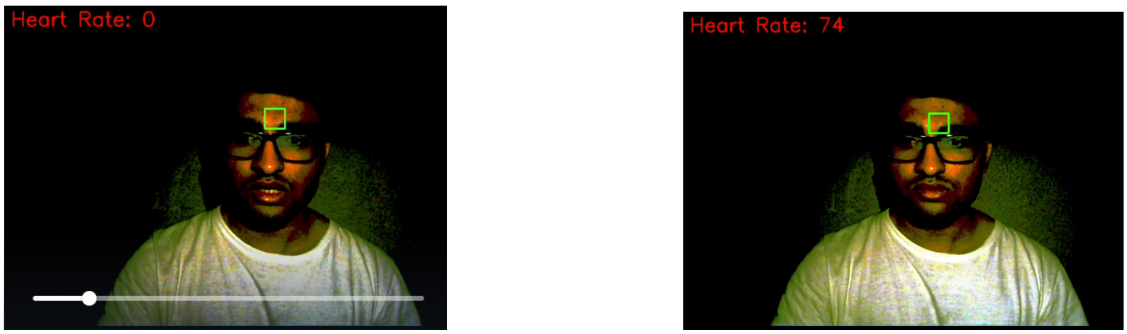
Thus, our VGG16 based model is able to obtain stabilized ROI's unlike OpenCV approach which is unable to obtain the ROI under such adverse conditions.

5.6 Heart Rate Measurement



(a) (b)

Figure 16: Heart Rate Measurement under ideal condition



(a) (b)

Figure 17: Heart Rate Measurement under low light condition

We have used a confidence score as a threshold according to which if and only if forehead is detected in the frame then only the heart rate is computed else heart rate will not be computed.
Figure 15-a shows that heart rate is not computed as forehead is not detected in the frame.
Figure 15-b shows accurate heart rate computation on detection of forehead.
Figure 16-a and 16-b shows heart-rate detection with some inaccuaries

5.7 Comparison of Inference time and FPS

5.7.1 Low-light video ehancement

Approach	**Inference Time (in sec)**	**FPS**
OpenCV	0.0995	10.050
MirNet	9.523	0.1173
Zero DCE	5.392	0.1854

Table 1: Low-light video ehancement

Zero DCE is an intermediate method between OpenCV and MirNet for heart rate measurement, providing more accurate results than MirNet and OpenCV. However, Zero DCE has a relatively longer inference time than OpenCV, but is faster than MirNet.

5.7.2 Motion Handling and ROI detection

Approach	**Inference Time (in sec)**	**FPS**
OpenCV	4 x 10^{-2}	25
VGG-16	7.5 x 10^{-2}	13.33

Table 2: Motion Handling and ROI detection

It can be seen that OpenCV for ROI detection have an inference time of 4×10^{-2} seconds and an FPS of 25. However, OpenCV is unable to detect ROI when the object is in motion, while VGG-16 base model can detect ROI in motion as well, but with a relatively higher time of 7.5×10^{-2} seconds seconds and an FPS of 13.33. Thus, our VGG16 based model is able to detect the ROI effectively even under the presence of motion in videos.

5.7.3 Inference Time and FPS for different Pipeline for Heart rate measurement

Pipelines	Inference Time (in sec)	FPS
OpenCV for low light enhancement + VGG16 for motion handling and ROI detection + Heart Rate measurement	0.7995	1.25
Zero-DCE for low light enhancement + VGG16 for motion handling and ROI detection + Heart Rate measurement	6.079	0.164
MirNet for low light enhancement + VGG16 for motion handling and ROI detection + Heart Rate measurement	9.497	0.105

Table 3: Inference Time and FPS for different Pipeline for Heart rate measurement

Based on the above table, it can be concluded that OpenCV is the fastest method for low light conditions and ROI detection, with an inference time of 0.7995 and FPS of 1.25. However, the combination of Zero DCE and VGG-16 provides better accuracy for low light conditions and ROI detection but with a relatively higher inference time of 6.079 and FPS of 0.164. MirNet with VGG-16 is the slowest method for low light conditions and ROI detection, with an inference time of 9.497 and FPS of 0.105.

5.8 Average RMES under different Conditions

Conditions	Boat smart watch	Samsung smart watch
Ideal conditions	4.02	5.69
Low light conditions	7.39	10.81
Video with motion	5.36	5.76

Table 4: Average RMES under different Conditions using Boat and Samsung smart watches.

Based on the root mean square error (RMSE) values obtained from your experiment, it can be concluded that heart rate measurement varies depending on the type of smartwatch and the environmental conditions.

Under ideal conditions, the Boat Smart Watch and the Samsung Smart Watch both performed relatively well, with RMSE values of 4.02 and 5.69, respectively. This suggests that in ideal lighting conditions, our model can be used for accurate heart rate measurement.

However, in low-light conditions, the RMSE values for both watches increased significantly, with the Boat Smart Watch having an RMSE of 7.39 and the Samsung Smart Watch having an RMSE of 10.81. This indicates that it may not be suitable to accurately measure heart rate in low-light conditions.

In the case of motion video,the Boat Smart Watch and the Samsung Smart Watch both performed relatively well than low light . This suggests that our model can be better suited for heart rate measurement during physical activity or when there is motion involved.

6 Discussion and Future Work

6.1 Discussion

- Based on the results obtained from our project, it is evident that there are several challenges associated with measuring heart rate in real-time using different techniques.
- Firstly, it was observed that the Zero-DCE method and the MirNet algorithm were not suitable for real-time heart rate measurement as the inference time was found to be very high and FPS was found to be very low. Therefore, it can be concluded that these methods are not practical for medical applications that require quick and accurate heart rate measurements.
- Secondly, OpenCV was identified as a potential solution for real-time heart rate measurement. However, the enhancement applied by OpenCV makes it difficult to capture the hue change, resulting in a high mean squared error of 15.37. This error is not within the acceptable range for medical applications and hence, cannot be used to measure heart rate in such scenarios.
- Despite these challenges, our VGG16 based model was found to perform well under proper lighting conditions, and even in motion. Unlike OpenCV, your model was able to detect the ROI even when the object was in motion, which is a significant advantage. Therefore, it can be concluded that your model outperforms OpenCV in terms of real-time heart rate measurement, especially in challenging scenarios when the object is in motion.
- In summary, our project highlights the challenges associated with real-time heart rate measurement using different techniques. While some methods may not be suitable for medical applications due to high inference time or high mean squared error, other methods may have limitations in capturing ROI during motion. However, our model has shown promising results, and with further research, it can potentially be used for real-time heart rate measurement in challenging scenarios.

Reason for inadequate heart rate measurement under low-light conditions:

- Under low-light conditions, the amount of light captured by the camera's sensor decreases significantly, leading to a decrease in the quality of the captured image. This decrease in image quality can affect the accuracy of the heart rate detection algorithm, which relies on detecting subtle color changes in the facial region.
- When the lighting conditions are poor, the color changes in the face due to the blood flow become less pronounced, and it becomes more difficult for the algorithm to detect these changes accurately. As a result, the heart rate detection accuracy decreases significantly in low-light conditions.
- Moreover, low light conditions can also introduce noise in the captured image which can further deteriorate the performance of the algorithm. In some cases, the algorithm may even fail to detect any heartbeat, leading to inaccurate heart rate measurements.
- Therefore, it is important to consider lighting conditions while designing and implementing heart rate detection systems, and take necessary steps to improve the lighting conditions for accurate and reliable measurements.
- In addition to the challenge of low-light conditions, heart rate detection can also be impacted by the choice of computer vision algorithm used. For example, OpenCV is a popular computer vision library, but it may not always provide accurate results for heart rate detection. On the other hand, more advanced deep learning models like Zero-DCE and MirNet may provide better accuracy, but they often require high inference time and computational resources.
- In the case of our experiment, we tested multiple computer vision algorithms, including OpenCV, MirNet, and Zero-DCE. While OpenCV was able to detect some heart rate signals, its accuracy was low under low-light conditions. The deep learning models, while showing better accuracy, required significantly more time for inference, especially in the case of Zero-DCE. Therefore, we had to choose a trade-off between accuracy and inference time, and eventually selected Zero-DCE, which had the highest accuracy and intermediate inference time compared to other approaches.

6.2 Future Scope

- To address the problem of fan-induced changes in hue on the face during heart rate detection, one solution is to apply image filtering techniques. Spatial filtering, such as a Gaussian or median filter, can help smooth out any noise caused by the fan by removing high-frequency components in the image signal. Temporal filtering, such as a moving average or low-pass filter, can reduce the effect of the fan on the overall heart rate signal by eliminating rapid changes in hue that are not related to blood flow. These techniques can help improve the accuracy of heart rate detection in the presence of fan-induced changes in hue.
- Using a ResNet model instead of a VGG-16 model as a feature extractor for detecting and measuring heart rate from video signals. ResNet models have been shown to outperform VGG-16 on a variety of computer vision tasks due to their deeper architecture and residual connections. Using a ResNet model could potentially lead to better accuracy and performance, especially when dealing with large, complex datasets. However, implementing a ResNet model may require more computational resources and training time compared to VGG-16, so the trade-offs between accuracy and efficiency should be carefully considered.
- Choose a region of interest (ROI) in the video frames that is less affected by the fan, such as the cheeks, and focus on measuring hue changes in that region only. This could help isolate the heart rate signal from the fan-induced noise.

7 Conclusion

Our project proposed the idea of real-time heart rate measurement from low-light video, which is a challenging task due to adverse conditions that affect the accuracy of heart rate measurements. The project explored multiple approaches for real-time heart rate measurement from low-light video, including OpenCV, MirNet, and Zero-DCE. Our objective was to find a method that not only provides accurate measurements but also reduces the computational complexity and inference time required for real-time heart rate detection. OpenCV was identified as a potential solution, but its enhancement made it difficult to capture hue changes accurately thus ,resulting inaccurate heart rate mesurement. Zero-DCE and MirNet were found to have high inference time and low FPS, making them impractical for real-time heart rate measurement.

our VGG16-based model showed promising results, performing well under proper lighting conditions and even in motion. Unlike OpenCV, our model was able to detect the ROI even when the object was in motion, which is a significant advantage. Our model eliminates the need for video stabilization and enhances the accuracy of our heart rate measurement. We have trained a VGG16 model on a large and diverse dataset of forehead images to enable more robust and accurate detection of foreheads under the effect of motion artifacts such as head movements.

Measuring heart rate in real-time from low-light video is a challenging task due to several factors such as motion artifacts and lighting conditions. Different approaches have their limitations, and it is crucial to choose a method that can provide accurate measurements while reducing computational complexity and inference time.

In conclusion, our project has highlighted the challenges associated with real-time heart rate measurement from low-light video and proposed a solution that leverages deep learning techniques for more accurate and efficient measurement. The VGG16-based model has shown promising results in detecting ROI even under motion, providing a potential solution for real-time heart rate measurement in challenging scenarios. With further research, this approach can contribute to the advancement of the field and help address the limitations of existing solutions for real-time heart rate measurement.

8 Individual Contributions

Group Members	**Individual Contribution**	**Percentage of contribution**
Allan Robey	Low Light enhancement(Zero-DCE, openCV,MirNet) , Project Report , Project Presentation	20%
Avnish Tripathi	Automated ROI Detection and video stablization , Project Report, Project Presentation	20%
Divyesh Tripathi	Automated ROI Detection and Video Stablization , Project Report,Project Presentation	20%
Kush Shah	Low Light Enhancement(Zero-DCE,openCV,MirNet), Project Report , Project Presentation	20%
Pulkit Sharma	Heart Rate Detection, Dataset creation for ROI Detection , Project Report, Project Presentation	20%

References

[1] C. Nadrag, V. Poenaru1, and G. Suciu1, "Heart rate measurement using face detection in video," *IEEE*, 2018.

[2] J. Pourbemany, A. Essa, and Y. Zhu, "Real time video based heart and respirationrate monitoring," *IEEE*, 2021.

[3] A. Gudi, M. Bittner, and J. van Gemert, "Real-time webcam heart-rate and variability estimation with clean ground truth for evaluation," *IEEE*, 2020.

[4] L. Y. Kassab, A. Law, B. Wallace, J. Larivière-Chartier, and R. Goubran, "Effects of lighting and window length on heart rate assessment through video magnification," *IEEE*, 2022.

[5] Y. Jiang, X. Gong, D. Liu, Y. Cheng, C. Fang, X. Shen, J. Yang, P. Zhou, and Z. Wang, "Enlightengan: Deep light enhancement without paired supervision," *IEEE*, 2015.

[6] F. Li, J. Zheng, and Y. fang Zhang, "Generative adversarial network for low-light image enhancement," *IEEE*, 2021.

[7] D.-Y. Kim, K. Lee, and C.-B. Sohn, "Assessment of roi selection for facial video-based rppg," *IEEE*, 2021.

[8] M. Wang, G.-Y. Yang, J.-K. Lin, S.-H. Zhang, A. Shamir, S.-P. Lu, and S.-M. Hu, "Deep online video stabilization with multi-grid warping transformation learning," *IEEE*, 2018.

[9] C. Nadrag, V. Poenaru, and G. Suciu, "Heart rate measurement using face detection in video," *IEEE*, 2018.

[10] T. Chowdhury, S. Chanda, S. Bhattacharya, and S. Biswas, "Contact-less heart rate detection in low light videos," *ACPR*, 2022.

[11] Y. Yang, D. Sun, H. Jiang, and M.-H. Yang, "Learning video stabilization using optical flow," *IEEE*, 2017.

[12] Y. Li, S. Liang, and S. Wang, "Low-light image enhancement based on generative adversarial network," *IEEE*, 2021.

[13] J.-P. Lomaliza, H. Park, and K.-S. Moon, "Heart rate measurement combining motion and color information," *IEEE*, 2020.

[14] K. Zheng, K. Ci, H. Li, L. Shao, G. Sun, J. Liu, and J. Cui, "Heart rate prediction from facial video with masks using eye location and corrected by convolutional neural networks," *IEEE*, 2022.

[15] L. Wang, G. Fu, Z. Jiang, and G. J. and, "Low-light image enhancement with attention and multi-level feature fusion," *IEEE*, 2019.