Assignment 1
Of
CS6650W – Smart Sensing For Internet Of Things
On
Smartphone-Based Heart Monitoring Using Video Analysis

Prepared by

Dipendu Ghosh

CS23M509

6th October, 2024

Contents

Project Overview:

This project is focused on developing a smartphone-based photoplethysmography (PPG) system, using the smartphone's camera to monitor blood flow. By analysing video frames from three different scenarios (resting, moderate exercise, and vigorous exercise), this system attempts to estimate heart rate, compute optimal thresholds for classification, and evaluate system performance using Receiver Operating Characteristic (ROC) curves. Our goal was to determine how well the system could classify blood flow events and separate them from background noise.

Assignment Tasks:

A. Warmup - Data Collection

Three 30-second videos were captured using a smartphone camera under three conditions:

1. Resting on bed (minimal blood flow changes).
2. After moderate exercise (increased heart rate).
3. After vigorous exercise (high-intensity exercise with a substantial increase in heart rate).

Assumptions:

- Each video is recorded with consistent settings, including frame rate (FPS), resolution (height and width), and duration.
- The red channel from the RGB video is assumed to carry the most information relevant to blood flow.

Code Explanation:

- The OpenCV "cv2.VideoCapture" method was used to load and process videos frame by frame.
- FPS, frame width, and height were extracted and stored for each video, ensuring consistent metrics are applied in the analysis.

Inference:

The data extraction step is critical as it sets up the raw input data (video frames) to be processed further for intensity and classification analysis. This information is also necessary for visualizing the results later in the project.

Output:

```
ppgvideos/1.mp4, Frame Count: 902.0, FPS: 29.957399824639612, Height: 1920.0, Width: 1080.0
ppgvideos/2.mp4, Frame Count: 967.0, FPS: 29.990396043459366, Height: 1920.0, Width: 1080.0
ppgvideos/3.mp4, Frame Count: 900.0, FPS: 29.990591840263445, Height: 1920.0, Width: 1080.0
```

B. Sensing Metric

This task involves computing the intensity variation with an aggregate statistical measure for each frame from the individual pixel data (R, G, B values) for a given frame where red channel of the video frames has the highest priority. This is because the red channel provides insights into blood flow due to its sensitivity to changes in oxygenated hemoglobin.

Assumptions:

- Blood flow variations are directly related to red pixel intensity changes, which makes the red channel the primary source of information. This is the reason while calculating the intensity the weight of Red is taken as 0.8 and 0.1 for Green and Blue.

- All frames within each video are processed uniformly to maintain consistency in intensity calculations.

Code Explanation:

The red, green and blue pixel values are extracted for each frame, and their sum across the frame is calculated as a proxy for the total blood flow at a given point in time.

Inference:

Significant intensity variations can be observed across the three video conditions. During exercise, the higher heart rate and increased blood flow are reflected by noticeable increases in the red pixel intensity, whereas the resting video shows much more stable intensity.

Output:

```
print(f"Video {videoIndex}, Frame {frameIndex + 1}, Intensity Metric: {intensityMetric:.4f}")
```

Uncommenting the above code will print the frame intensity frame by frame per video.

C. Temporal Intensity Variation

The intensity frame metric is plotted against time for the last 5 second chunk of all three captured videos.

Assumptions:

- The 5 second chunks are extracted from the videos using a random function. So for the same video for different run the chunks will be different
- The find_peaks function is used for peak detection, assuming peaks correspond to heartbeats. Here 0.4 is multiplied with the fps of the current video so that the highest peaks considered are well spaced. Why 0.4? Assuming every heartbeat is of half a second in the best case but while vigorous exercise it may be less. Taking an average.

Observations:

- It is observed that there are noticeable peaks and valleys in the extracted signal. It is also possible to estimate the heart rate by manually counting these peaks.
- Vigorous exercise leads to rapid intensity variations compared to while resting.

Code Explanation:

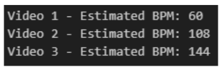
- Extracting Random Chunks: For each video in intensityMetrics, a random 5-second chunk of frames is extracted. The FPS of each video is stored in framesPerSecondValues.
- Peak Detection and BPM Estimation: After extracting the chunks, peaks are detected in the intensity metrics to estimate the number of heartbeats. The total number of peaks is multiplied by 12 (since 5 seconds × 12 = 60 seconds) to calculate BPM.
- BPM Output: The estimated BPM for each video is printed for reference.

Inference:

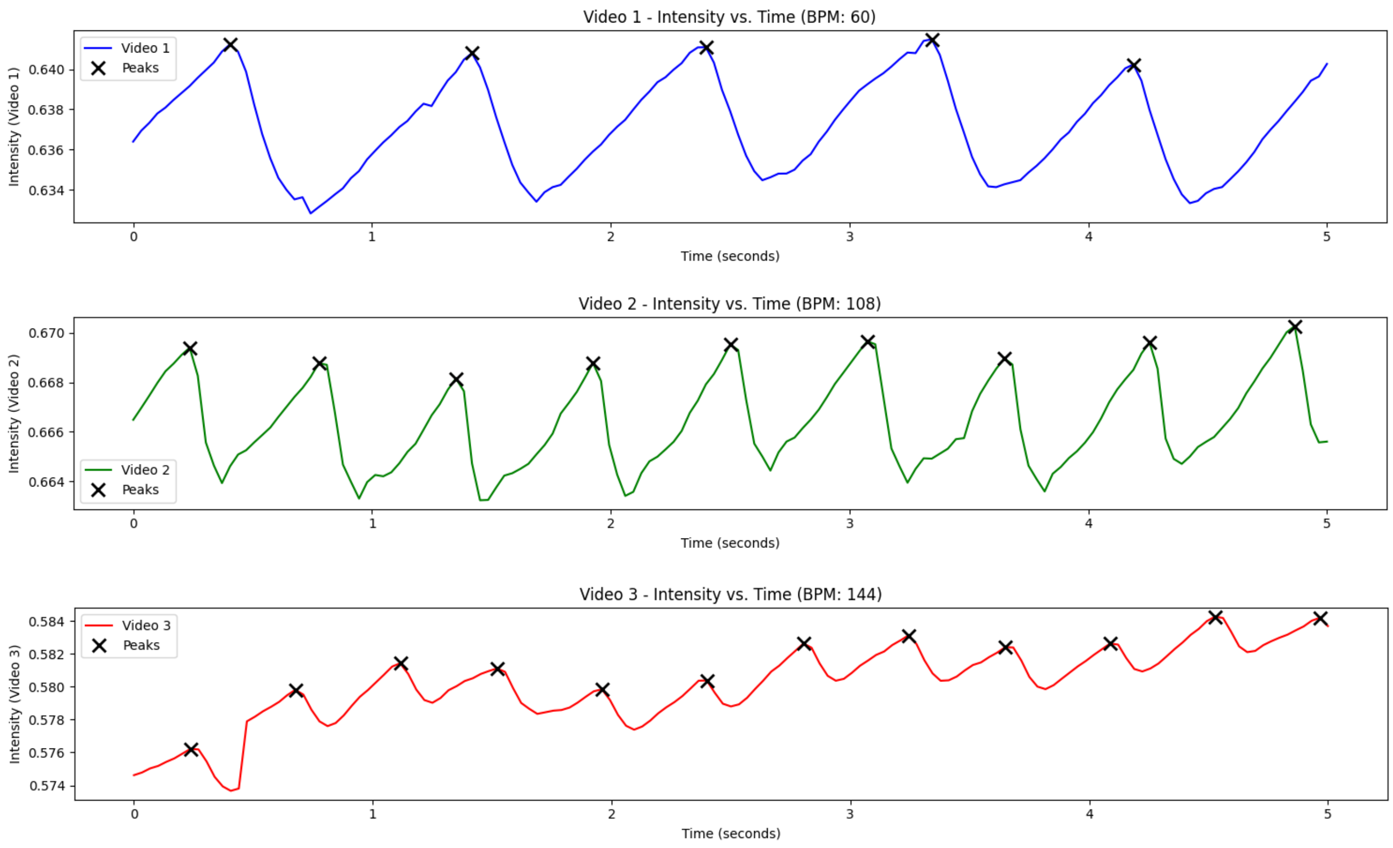
Extracts random 5-second chunks from each video, detects peaks in intensity metrics to estimate BPM, and visualizes the results. This is designed for flexible video input and allows easy estimation of heart rate from video-based PPG signals.

Output:

BPM per video is calculated by finding the number of peaks and multiplying by 12 as described above. Then per video temporal variation of intensity value is plotted per video. This will be random for each run as the chunks selected are different. For the current run the snapshot is provided below.



```
Video 1 - Estimated BPM: 60
Video 2 - Estimated BPM: 108
Video 3 - Estimated BPM: 144
```



D. Likelihood Distributions

20 frames are chosen in the neighbourhood where the sensing metric is close to the local maximum (Case 1) and another 20 frames are chosen where the sensing metric is close to the local minimum (Case 2). Histograms on all 3 channels (R, G and B) are plotted and the distributions are analysed.

Assumptions:

- Consistent Frame Dimensions: Assumes all frames have consistent dimensions.
- RGB Color Space: Assumes frames are in BGR format compatible with OpenCV.
- Extreme Selection: Assumes identifiable minimum and maximum intensity metrics exist in the chunk.
- Frame Count: Assumes enough frames (at least numFrames) are present in the chunk for selection.
- Non-Empty Lists: Assumes maxFrames and minFrames are not empty.

Observations:

- For all three datasets and almost all the three channels, there is a significant if not complete overlap between the distributions. This is because even for one frame there is quite significant variation in pixel values in all 3 channels. Thus the information encoded in a single pixel is not enough to capture and classify the frame as a positive/negative event.
- With regard to separability of distributions it is difficult to conclude because of the overlap in distributions

Code Explanation:

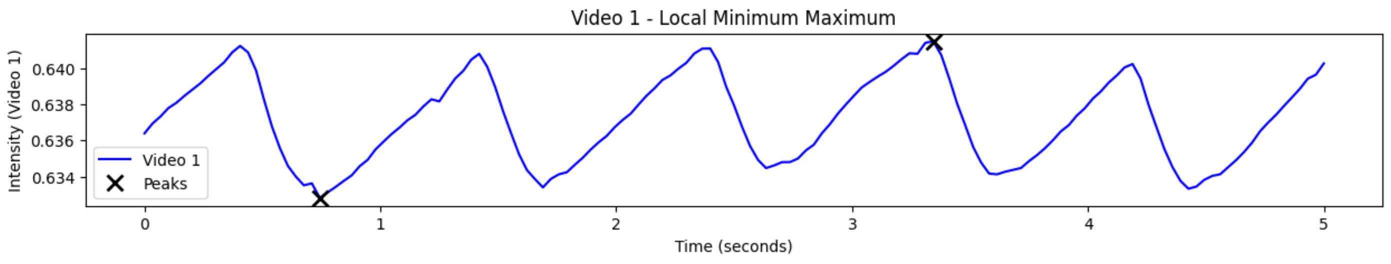
- select_extreme_frames:
  This function identifies frames with the minimum and maximum intensity metrics from a chunk of intensity data. It returns the frames surrounding these extreme values, along with their respective indices.
- extract_rgb_values_histogram_1D:
  This function extracts RGB color values from the selected frames and flattens them into one-dimensional arrays.It retrieves each frame from the video using the videoIndex and splits it into its RGB components.
- plot_local_min_max:
  This function visualizes the local minimum and maximum intensity metrics over time for a specific video.It plots intensity values against time, highlighting the locations of identified minima and maxima.
- plot_histograms:
  This function generates histograms showing the RGB intensity distributions for both maximum and minimum frames. It plots the histograms in a way that allows for easy comparison between the maximum and minimum intensities for each color channel.

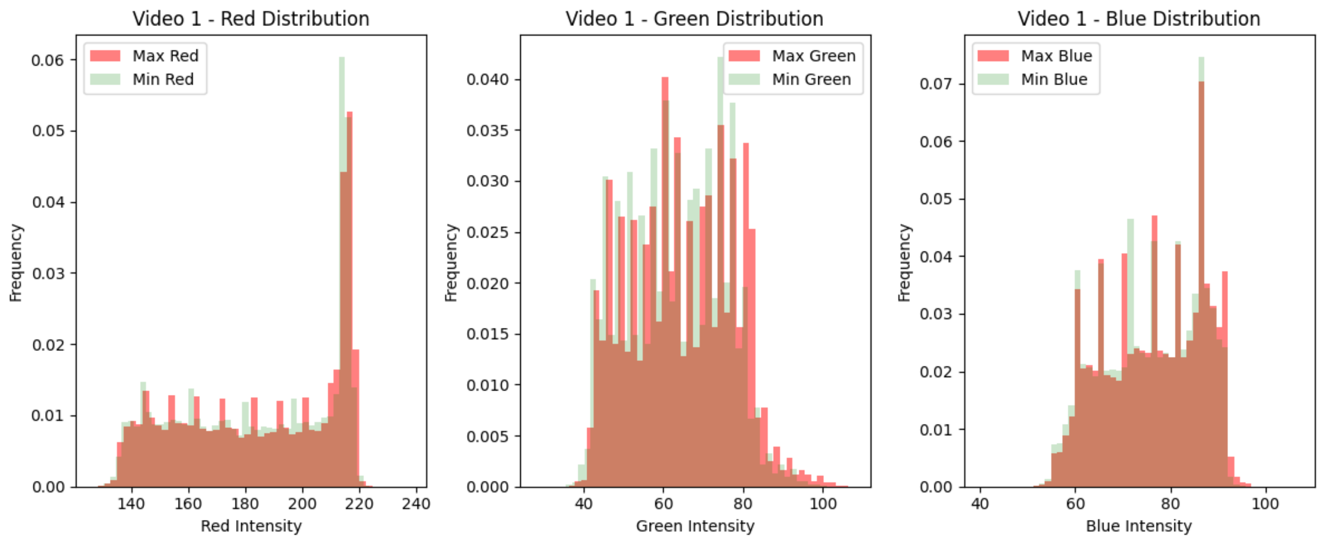
Inference:

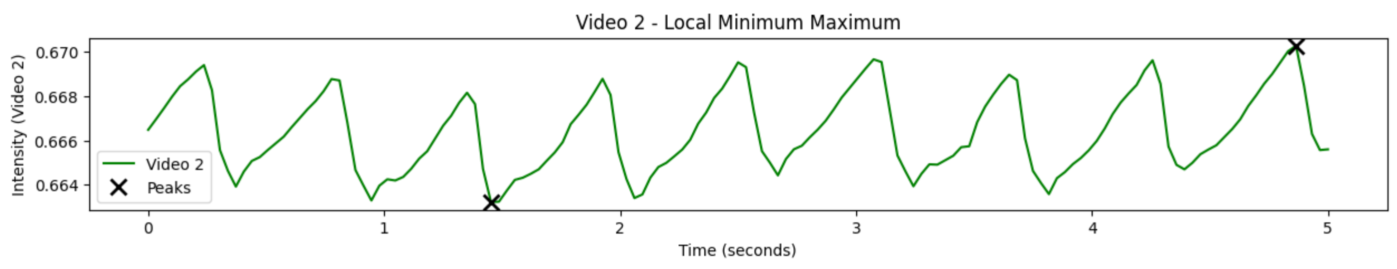
- The results are displayed through plots that show the relationship between intensity metrics over time and the local maximum and minimum per channel per video.
- The distribution of RGB colour intensities in the extracted frame chunks, facilitating a comprehensive analysis of video content.

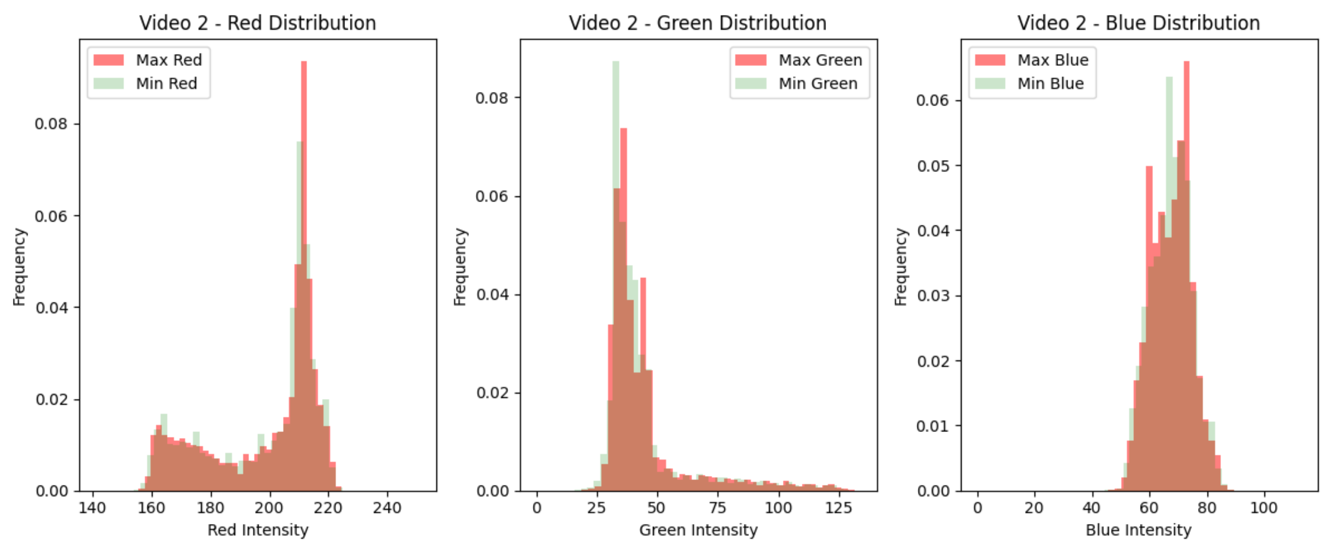
Output:

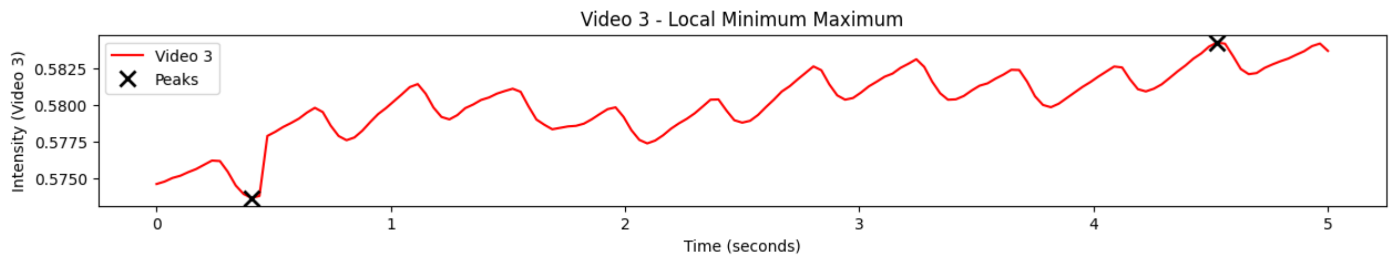
Histogram per channel per video and the local minimum and local maximum. For the Green channel it seems to be more separable.

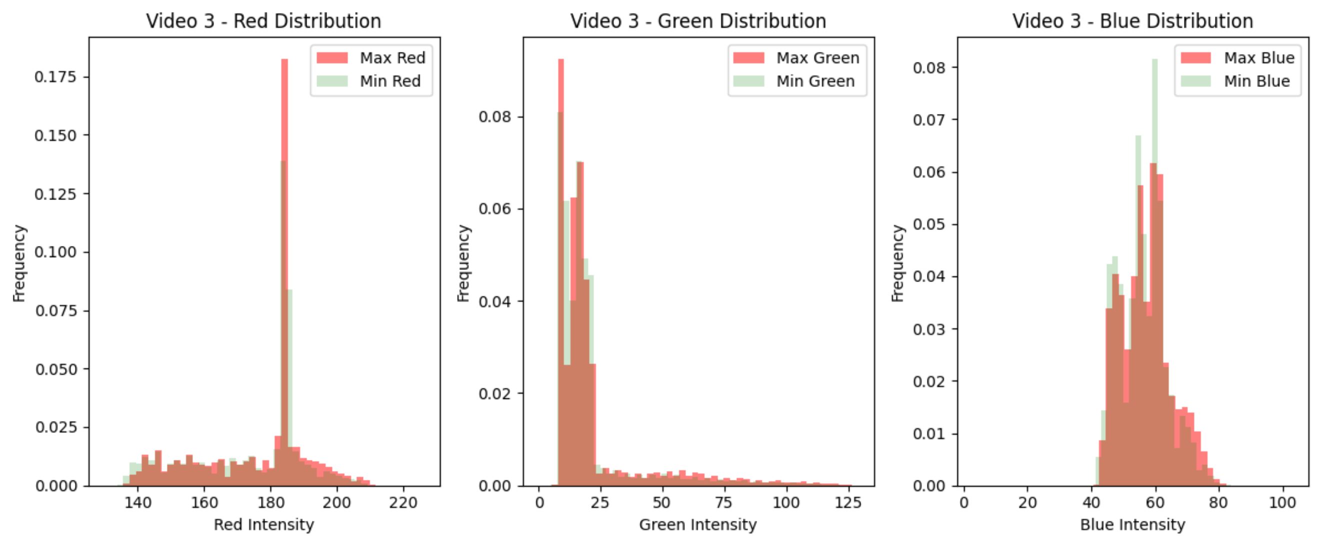












E. Threshold Based Detection and ROC Curve

Only the R-channel is considered and out of each of the 40 (20 + 20) frames, 500 random pixels are chosen and based on the pixels values, a threshold based classification is done to classify the frame as Case 1 (positive event, local maxima) or Case 2 (negative event, local minima). The probability of detection (PD) and probability of false alarm (PFA) is computed and the ROC curve is plotted for various threshold values.

Assumptions:

- The ROC curve is computed based on the true positive rate (Pd) and false positive rate (Pfa), assuming that the classification thresholds can effectively capture the detection rates of maximum intensity pixels while minimizing false alarms from minimum intensity frames.
- The AUC score: The area under the Receiver Operating Characteristic (ROC) curve, called the AUC score, gives a statistical measure on the quality of the prediction model. A model whose predictions are 100% and wrong has an AUC of 0.0; one whose predictions are 100% correct has an AUC of 1.0.
- The code assumes that a random selection of 500 pixels from each frame is representative of the overall frame's intensity distribution. It also assumes that this random selection does not bias the results significantly.
- The classification between the two hypotheses (H0 and H1) is based on a threshold applied to the likelihood ratio of pixel intensities. It assumes that there is a meaningful range of thresholds that differentiate between maximum and minimum intensity frames.

Observations:

- The AUC scores vary across different videos, indicating that some videos are better suited for classification based on pixel intensity differences between maximum and minimum frames. The higher the AUC score, the better the classification performance.
- The ROC curves provide a visual representation of the trade-off between detection (Pd) and false alarms (Pfa) for different thresholds. A video with a higher AUC indicates a better balance between true positive detection and false positive avoidance.
- The sensitivity of classification to the choice of thresholds is demonstrated through the ROC curves. As the threshold increases, the true positive rate (Pd) generally decreases, but so does the false positive rate (Pfa), showing the impact of selecting appropriate thresholds for classification.
- The red channel pixel intensities from frames can be used to distinguish between frames of maximum and minimum intensity, but the effectiveness varies across videos, as seen in the differing AUC scores.
- Random pixel selection from frames seems to provide enough information to generate reliable ROC curves, suggesting that a subset of pixels is sufficient to capture the overall intensity variation in the frames.

Code Explanation:

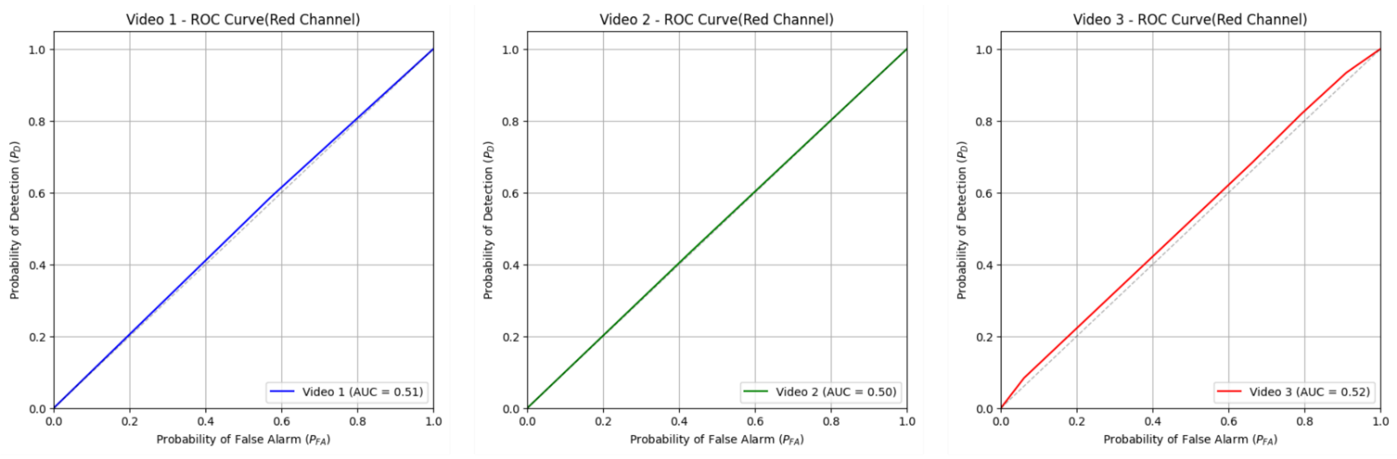
- compute_auc(pfa, pd):
  This function calculates the Area Under the Curve (AUC) for a given Receiver Operating Characteristic (ROC) curve. It takes in probabilities of false alarm (pfa) and probabilities of detection (pd), sorts them, and computes the AUC using the trapezoidal rule for numerical integration.
- compute_roc(redPixelsMax, redPixelsMin, thresholds, videoIndex):
  This function computes the ROC curve values, including true positive rates (Pd) and false positive rates (Pfa), based on the pixel intensities from maximum and minimum intensity frames. It applies a likelihood ratio test with different thresholds to classify pixel values and then calculates Pd, Pfa, and the AUC for each threshold.
- plot_roc_curves(pd, pfa, auc):
  This function generates and plots the ROC curve using the calculated true positive rates (pd), false positive rates (pfa), and AUC score. It also adds a diagonal reference line for random guessing and labels the axes.

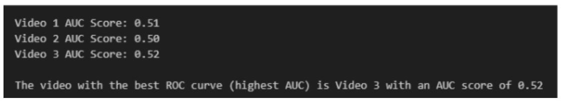
Inference:

- The code serves as a comprehensive framework for analyzing pixel intensity data to assess classifier performance through ROC curves and AUC scores. It combines statistical analysis with visualization, enabling a clear understanding of how well the model discriminates between different classes based on the intensity of red pixels in video frames. This approach is beneficial for applications in video analysis, image classification, and other fields where pixel intensity plays a significant role.

Output:

The ROC curve per video along with the AUC. The best AUC is selected after the plot. This will vary for each run depending on the 5sec chunk selected in C. For the run when the result was taken the outputs are in the snapshot below.





```
Video 1 AUC Score: 0.51
Video 2 AUC Score: 0.50
Video 3 AUC Score: 0.52

The video with the best ROC curve (highest AUC) is Video 3 with an AUC score of 0.52
```

F. Spatial Correlation Analysis

An optimal threshold value is chosen that maximises PD while minimising PFA. For a given video sample, a random frame from the local maximum(positive event) and one from the local minimum (negative event) is chosen. The spatial correlation between the ”good” samples (true positives and true negatives) and ”bad” samples (false positives and false negatives) is analysed.

Assumptions:

- Threshold Sensitivity: It is assumed that the optimal threshold (T_OPT) derived from the Pd and Pfa values accurately reflects the best point for classification in the context of the data. This threshold is critical for determining the sensitivity and specificity of the classification.
- Classification Logic: The logic for classifying the samples into different categories (TP, TN, FP, FN) assumes that the likelihood ratios are sufficient to determine the correctness of each classification. This implies that the likelihood ratio test is a valid method for the specific problem being addressed.

Observations:

- Optimal Threshold Calculation: The find_optimal_threshold function effectively computes the optimal threshold (T_OPT) using the ratio of Probability of Detection (Pd) to Probability of False Alarm (Pfa). This approach helps balance sensitivity and specificity in classification tasks.
- Likelihood Ratio Classification: The classification of samples into True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN) is based on the likelihood ratio test, leveraging normal distribution fitting for pixel values. This is a robust statistical method for binary classification.

Code Explanation:

- Function: find_optimal_threshold:

  This function calculates the optimal threshold (T_OPT) for classification by maximizing the ratio of Probability of Detection (Pd) to Probability of False Alarm (Pfa). The likelihood ratio is computed, and the threshold corresponding to the maximum ratio is selected as optimal.
- Function: classify_samples:

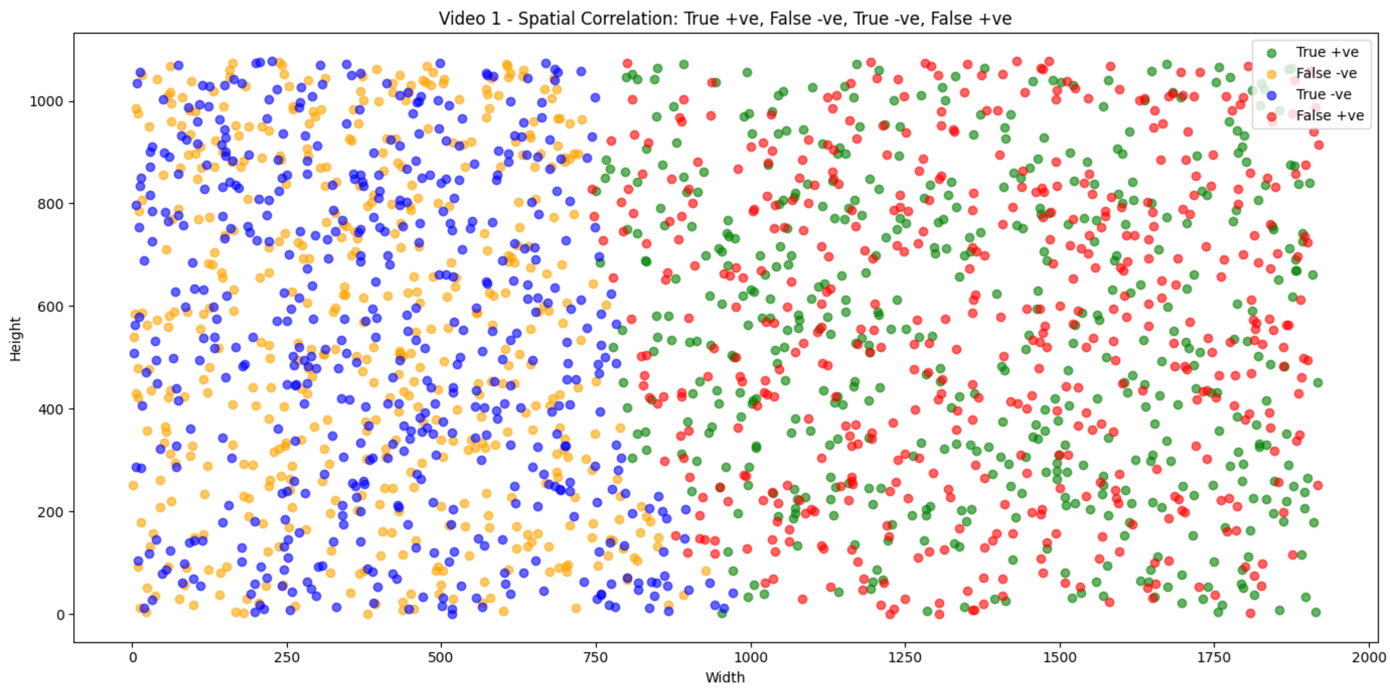
  This function classifies pixel values (from redPixelsMax and redPixelsMin) into True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN) based on the optimal threshold (T_OPT). It fits normal distributions to the red pixel values (for Max and Min) and applies a likelihood ratio test to classify each sample.

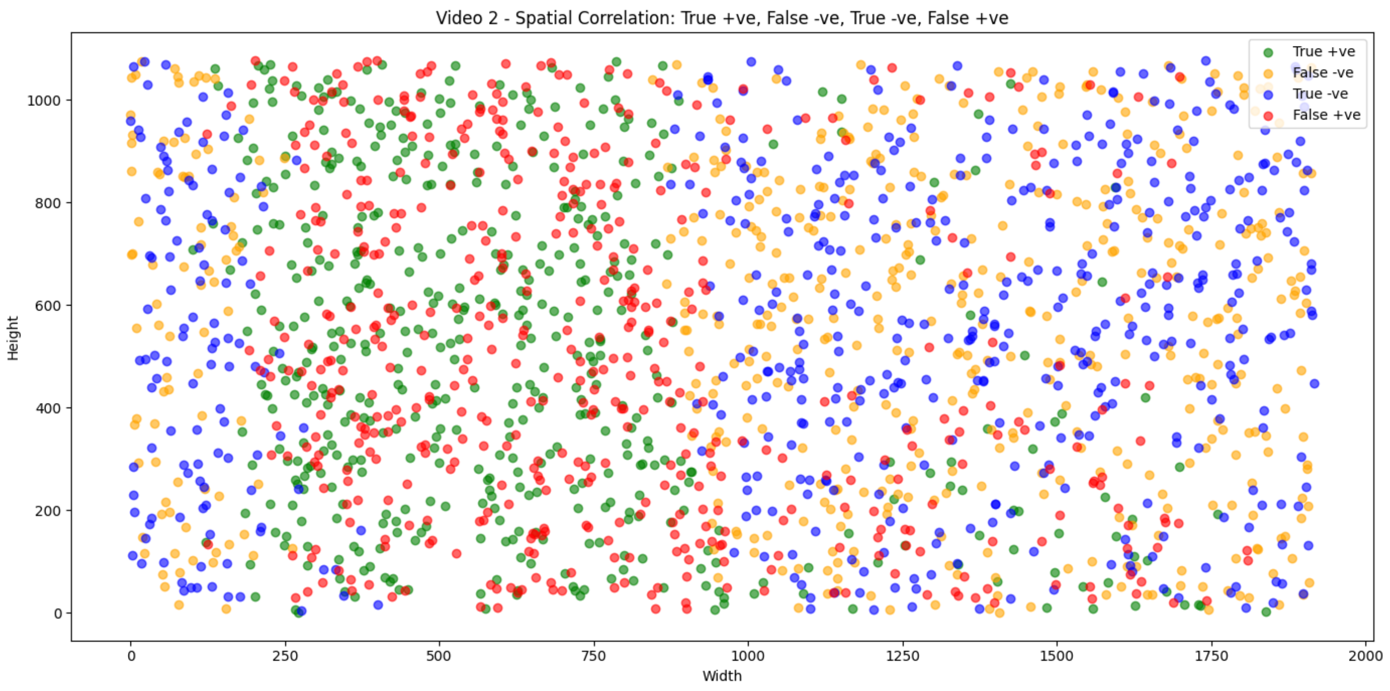
Inference:

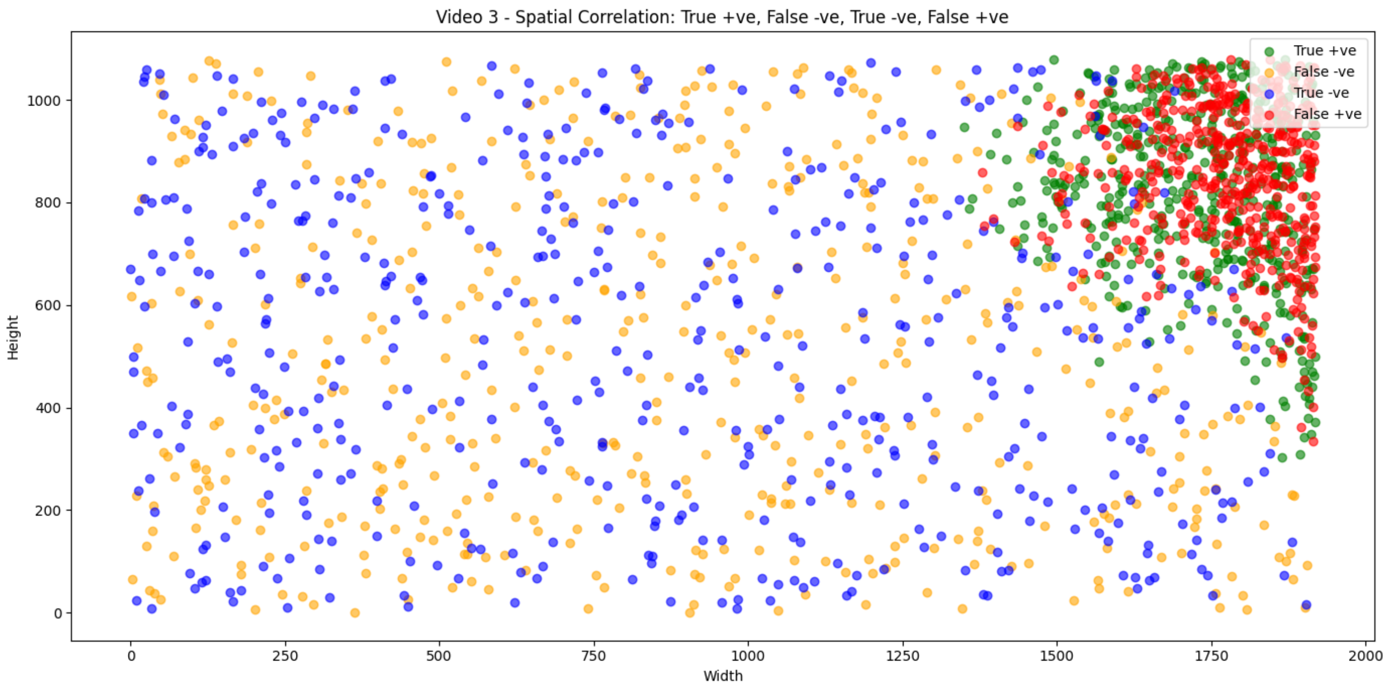
The code provides a reliable method for optimal classification of pixel intensities, visualizes the results effectively, and offers insights into the classifier's spatial performance across video frames.

Output:

The spatial plot per video is provided. For the current run the optimal threshold seems to be around 1 and it can it be hypothesised that the "good" samples (true positives and true negatives) are spatially clustered in certain areas of the frame, compared to the "bad" samples (false positives and false negatives). Here a sample set of 500 for each class is used. This can be modified by changing the value of “sampleSize”. It will plot minimum of “sampleSize” provided and the total number of sample size present per class.







Conclusion:

This project successfully developed a smartphone-based PPG analysis system capable of detecting blood flow using video data. The analysis involved Gaussian fitting, likelihood ratio testing, and ROC curve evaluation, showing promising results in distinguishing true events (blood flow) from noise. The system's performance improved significantly after exercise due to the increase in blood flow, as shown by higher AUC values in the ROC curve.

References:

-https://docs.opencv.org/master/d8/dfe/classcv_1_1VideoCapture.html

- https://en.wikipedia.org/wiki/Receiver_operating_characteristic

- https://en.wikipedia.org/wiki/Normal_distribution

- https://dl.acm.org/doi/pdf/10.1145/3560830.3563722

- https://medium.com/@bgallois/smartphone-based-heart-rate-monitoring-preprocessing-and-analysis-of-ppg-signals-de443473f529