AUTOMATED BLEEDING DETECTION IN WIRELESS CAPSULE ENDOSCOPY IMAGES

Foundations of Machine Learning (EC346) Project Report

Submitted in partial fulfillment of the requirements for the degree of

BACHELOR OF TECHNOLOGY in

ELECTRONICS AND COMMUNICATION ENGINEERING

by

Kaliki Venkata Srinanda (211EC117)

Suhas R P (211EC153)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING,

NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA,

SURATHKAL, MANGALORE -575025

November 2023

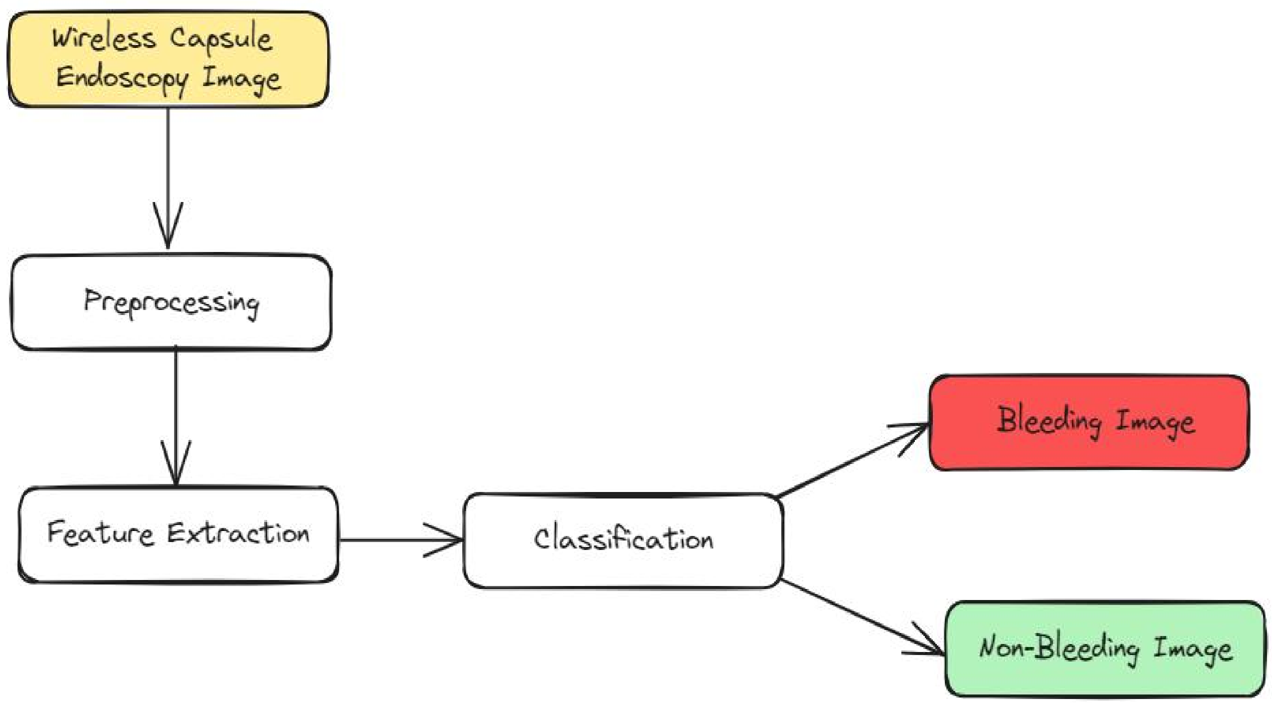
INDEX

CONTENTS	Page Number

OBJECTIVE

Wireless Capsule Endoscopy (WCE) is a non-invasive medical imaging technology that allows the visualization of the digestive tract using a small, swallowable capsule equipped with a camera. In our project, we focus on a dataset comprising of WCE images, aiming to distinguish between bleeding and non-bleeding images. Early and accurate detection of bleeding conditions can lead to improved diagnosis and more effective treatment. By leveraging feature extraction techniques and ensemble machine learning models, this project aims to enhance the accuracy and reliability of bleeding detection in WCE images.

PROPOSED APPROACH AND IMPLEMENTATION



A block diagram explaining the proposed approach

Pre-processing:

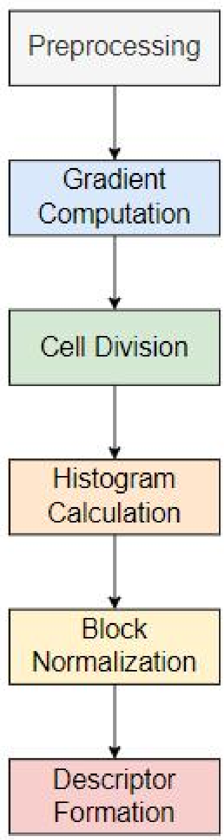
Grayscale conversion is a common pre-processing step in image analysis, simplifying the data while preserving essential information for various computer vision tasks. The **load_and_preprocess_images** function (in the code) takes a folder path as input and processes each image file within that folder. For each image, it reads the file using **OpenCV** (cv2.imread), converts it to grayscale using the **rgb2gray** function from the **scikit-image library**, and appends the resulting grayscale image to a list. The function then returns a list containing all the preprocessed grayscale images from the specified folder.

Feature Extraction:

We have extracted 1 feature to train the model. The feature is **Histogram of Gradients (HOG).**

The Histogram of Oriented Gradients, commonly referred to as **HOG**, is a feature extraction technique used in computer vision and image processing. We perform the preprocessing step of converting the images into grayscale format to implement HOG. It also includes other steps as follows:

- Gradient computation: HOG calculates the gradient magnitudes and orientations of image pixels. This step helps in identifying edges and texture boundaries.
- Cell construction: The image is divided into small, overlapping cells. Typically, each cell covers a region of 8x8 pixels.
- Histogram calculation: For each cell, a histogram of gradient orientations is computed. The orientations are quantized into bins, and the histogram represents the distribution of gradient orientations within the cell.
- Block normalization: Cells are grouped into larger blocks (usually consisting of 2x2 or 3x3 cells). Normalization is applied within each block to enhance the algorithm's robustness to changes in lighting and contrast.
- Descriptor formation: The normalized histograms from all blocks are concatenated to form the final HOG descriptor for the image. This descriptor captures the spatial distribution of gradients and their orientations.



HOG Workflow

We implemented HOG with the help of **sci-kit learn library** and used the following parameters in the code.

Parameters Used in the Code:

- **block_norm='L2-Hys': This** parameter defines the block normalization method. 'L2-Hys' indicates that the block normalization is done using L2 norm followed by clipping and normalization.
- **pixels_per_cell=(8, 8):** Specifies the size of each cell in pixels. Gradients are calculated within each cell, and the histograms are created based on these gradients.
- **cells_per_block=(2, 2): Defines** the number of cells in each block. The histograms from each block are then concatenated to form the final feature vector.

We also tried using other features such as Gabor Filter and Local Binary Patterns to train the model. However, they gave lower accuracy when combined with HOG. Hence, we decided to drop them and proceed with HOG as the only feature.

Models for Classification:

Before training the model, the bleeding and non-bleeding images were assigned labels of 1 and 0. The dataset was split in the ratio 80:20 for training and testing.

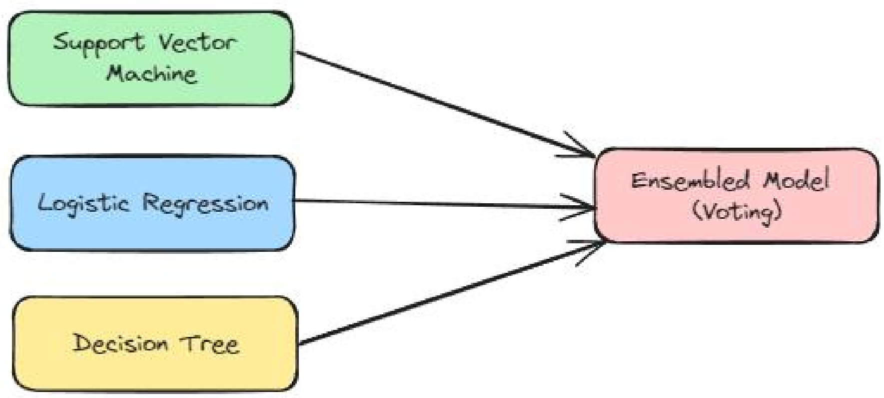
We have used 3 base models to build the classifier. They are as follows:

- **Support Vector Machine (SVM)**: Support Vector Machines (SVM) is a supervised machine learning algorithm used for classification and regression tasks. Its primary objective is to find a hyperplane that best separates the data points into different classes. The kernel trick allows SVM to implicitly map the input data into a higher-dimensional space, making it possible to find a hyperplane that separates non-linearly separable data. SVMs (Support Vector Machines) are effective in high-dimensional spaces and versatile due to the kernel trick. We have utilized SVM with the help of **sci-kit learn library** and used a **polynomial kernel of degree 3** and the **hyperparameter 'C'=1.**

- **Logistic Regression:** Logistic Regression is a widely employed supervised machine learning algorithm utilized for binary classification tasks. Its primary aim is to model the probability of an instance belonging to a particular class, employing the logistic function to constrain the output between 0 and 1. Logistic Regression estimates the parameters of the hyperplane that best separates the classes, allowing for probabilistic predictions. Logistic Regression is interpretable, computationally efficient, and effective in multiple scenarios. In this project, we employed Logistic Regression using the **sci-kit learn library**, employing the **default logistic function.** A few parameters include using L2 regularization, maximum iterations as 100 and tolerance for stopping criteria as 1e-4.

- **Decision Tree:** A decision tree is one of the most powerful tools of supervised learning algorithms used for both classification and regression tasks. It builds a flowchart-like tree structure where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label. It is constructed by recursively splitting the training data into subsets based on the values of the attributes until a stopping criterion is met. In this project, we implemented Decision Tree using the **sci-kit learn library** utilizing the **default parameters.** It uses Gini impurity as the criterion and uses the best splitter strategy.

Ensembled Model:



Block Diagram for Ensembled Model

Ensemble methods are a category of machine learning techniques that aim to improve the accuracy and robustness of models by combining the predictions of multiple base models. It offers more generalization and diversity.

We have employed the **'Voting'** method to ensemble the SVM, Logistic Regression and Decision Tree models. Voting refers to the process by which multiple models contribute their predictions, and the final decision is made based on a certain criterion. There are two main types of voting: hard voting and soft voting.

In **hard voting,** each model in the ensemble casts a **single "vote"** for a specific class label, and the class with most votes becomes the final prediction. This approach is usually said to be effective when the individual models have high accuracy in their predictions. **Soft voting** involves combining the **predicted probabilities** from each model and selecting the class with the highest average probability. This method is suitable when the base models can provide probability estimates.

We have implemented **Hard Voting** while ensembling the models since the models had high accuracy.

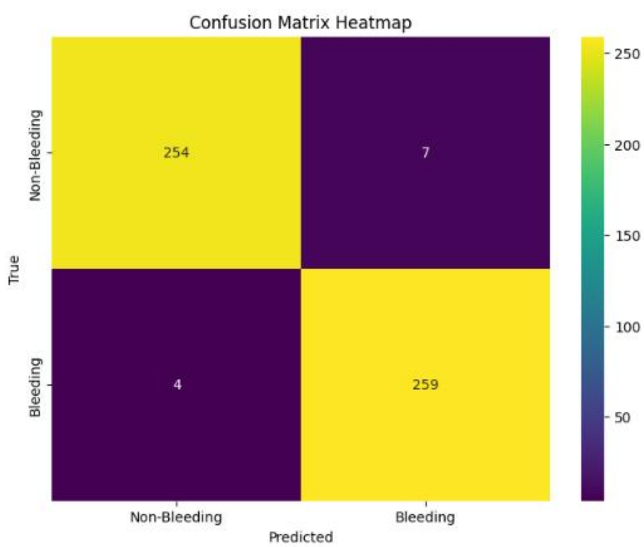
RESULTS AND DISCUSSIONS

Individual Models:

1.SVM:

```
Accuracy: 0.9790076335877863
              precision    recall  f1-score   support

         0.0       0.98      0.97      0.98       261
         1.0       0.97      0.98      0.98       263

    accuracy                           0.98       524
   macro avg       0.98      0.98      0.98       524
weighted avg       0.98      0.98      0.98       524

Confusion Matrix:
True Negative (TN): 254
False Negative (FN): 4
False Positive (FP): 7
True Positive (TP): 259
[[254   7]
 [  4 259]]

Evaluation Metrics:
Accuracy 0.9790076335877863
Precision: 0.9736842105263158
Recall: 0.9847908745247148
F1 Score: 0.9792060491493384
```

Classification Report and Confusion Matrix of SVM Model



Confusion Matrix Heatmap of SVM Model

Key Evalution Metrics of the SVM Model:

Accuracy: 0.9790

Precision: 0.9736

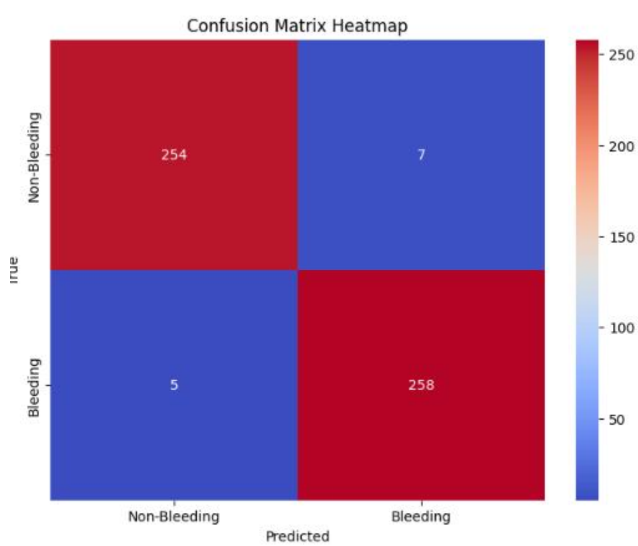
Recall: 0.9847

F1 Score: 0.9792

2.Logistic Regression:

```
Accuracy: 0.9770992366412213
              precision    recall  f1-score   support

         0.0       0.98      0.97      0.98       261
         1.0       0.97      0.98      0.98       263

    accuracy                           0.98       524
   macro avg       0.98      0.98      0.98       524
weighted avg       0.98      0.98      0.98       524

Confusion Matrix:
True Negative (TN): 254
False Negative (FN): 5
False Positive (FP): 7
True Positive (TP): 258
[[254   7]
 [  5 258]]

Evaluation Metrics:
Accuracy 0.9770992366412213
Precision: 0.9735849056603774
Recall: 0.9809885931558935
F1 Score: 0.9772727272727272
```

Classification Report and Confusion Matrix of LR Model



Confusion Matrix Heatmap of LR Model

Key Evalution Metrics of the Logistic Regression Model:

Accuracy 0.9770

Precision: 0.9735

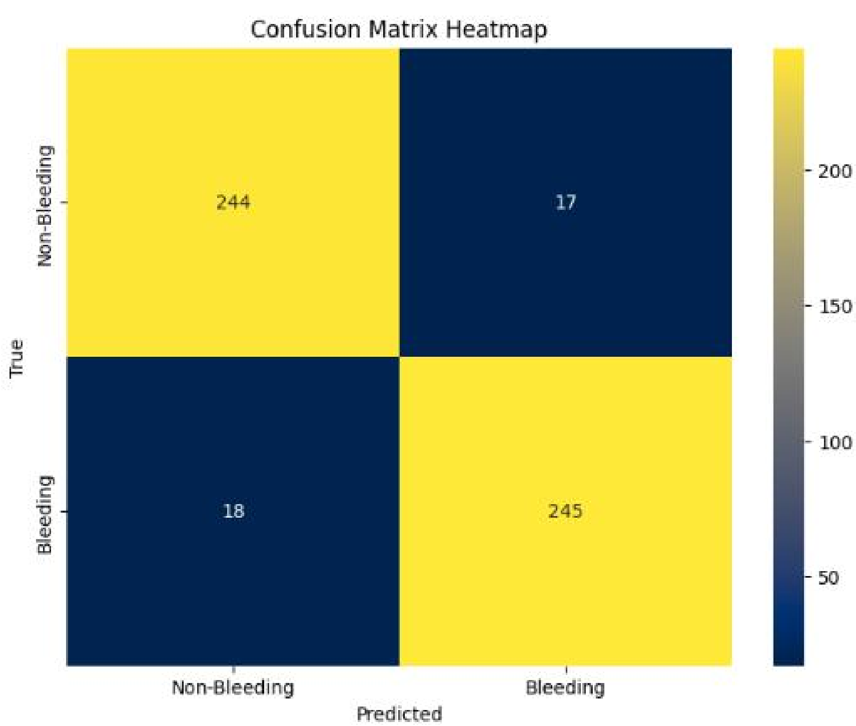
Recall: 0.9809

F1 Score: 0.9772

3.Decision Tree:

```
Accuracy: 0.933206106870229
              precision    recall  f1-score   support

         0.0       0.93      0.93      0.93       261
         1.0       0.94      0.93      0.93       263

    accuracy                           0.93       524
   macro avg       0.93      0.93      0.93       524
weighted avg       0.93      0.93      0.93       524

Confusion Matrix:
True Negative (TN): 244
False Negative (FN): 18
False Positive (FP): 17
True Positive (TP): 245
[[244  17]
 [ 18 245]]

Evaluation Metrics:
Accuracy 0.933206106870229
Precision: 0.9351145038167938
Recall: 0.9315589353612167
F1 Score: 0.9333333333333333
```

Classification Report and Confusion Matrix of Decision Tree Model



Confusion Matrix Heatmap of Decision Tree Model

Key Evalution Metrics of the Decision Tree Model:

Accuracy 0.9332

Precision: 0.9351

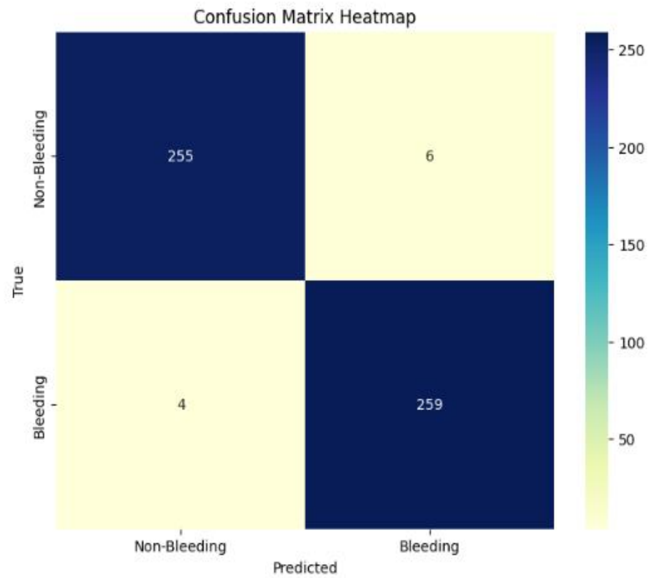
Recall: 0.9315

F1 Score: 0.9333

Ensembled Model:

```
Accuracy: 0.9809160305343512
              precision    recall  f1-score   support

         0.0       0.98      0.98      0.98       261
         1.0       0.98      0.98      0.98       263

    accuracy                           0.98       524
   macro avg       0.98      0.98      0.98       524
weighted avg       0.98      0.98      0.98       524

Confusion Matrix:
True Negative (TN): 255
False Negative (FN): 4
False Positive (FP): 6
True Positive (TP): 259
[[255   6]
 [  4 259]]

Evaluation Metrics:
Accuracy 0.9809160305343512
Precision: 0.9773584905660377
Recall: 0.9847908745247148
F1 Score: 0.9810606060606061
```

Classification Report and Confusion Matrix of Ensembled Model



Confusion Matrix Heatmap of Ensembled Model

<u>Key Evalution Metrics of the Ensembled Model:</u>

Accuracy: 0.9809

Precision: 0.9773

Recall: 0.9847

F1 Score: 0.9810

We also performed additional experiments using inbuilt ensemble methods such as Random Forest Classifier and XGBoost Classifier.

Accuracy of Random Forest Classifier: 0.9637

Accuracy of XGBoost Classifier: 0.9770

```
Accuracy: 0.9637404580152672
              precision    recall  f1-score   support

         0.0       0.99      0.94      0.96       261
         1.0       0.94      0.99      0.96       263

    accuracy                           0.96       524
   macro avg       0.96      0.96      0.96       524
weighted avg       0.96      0.96      0.96       524

Confusion Matrix:
True Negative (TN): 245
False Negative (FN): 3
False Positive (FP): 16
True Positive (TP): 260
[[245  16]
 [  3 260]]

Evaluation Metrics:
Accuracy 0.9637404580152672
Precision: 0.9420289855072463
Recall: 0.9885931558935361
F1 Score: 0.9647495361781075
```

Results of Random Forest Classifier

```
Accuracy: 0.9770992366412213
              precision    recall  f1-score   support

         0.0       0.99      0.96      0.98       261
         1.0       0.96      0.99      0.98       263

    accuracy                           0.98       524
   macro avg       0.98      0.98      0.98       524
weighted avg       0.98      0.98      0.98       524

Confusion Matrix:
True Negative (TN): 251
False Negative (FN): 2
False Positive (FP): 10
True Positive (TP): 261
[[251  10]
 [  2 261]]

Evaluation Metrics:
Accuracy 0.9770992366412213
Precision: 0.9630996309963099
Recall: 0.9923954372623575
F1 Score: 0.9775280898876405
```

Results of XGBoost Classifier

As an additional exercise, we implemented a simple Convolutional Neural Network (CNN) using TensorFlow for classify images. It performed extremely well and gave an accuracy of 100%.

```
Epoch 1/10
66/66 [==============================] - 3s 22ms/step - loss: 0.0136 - accuracy: 0.9981 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
Epoch 2/10
66/66 [==============================] - 1s 17ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
Epoch 3/10
66/66 [==============================] - 1s 17ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
Epoch 4/10
66/66 [==============================] - 1s 16ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
Epoch 5/10
66/66 [==============================] - 1s 16ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
Epoch 6/10
66/66 [==============================] - 1s 18ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
Epoch 7/10
66/66 [==============================] - 1s 20ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
Epoch 8/10
66/66 [==============================] - 1s 21ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
Epoch 9/10
66/66 [==============================] - 1s 22ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
Epoch 10/10
66/66 [==============================] - 1s 21ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
17/17 [==============================] - 0s 7ms/step - loss: 0.0000e+00 - accuracy: 1.0000
Test accuracy: 1.0
```

Results of Convolutional Neural Network

REFERENCES

1. https://www.geeksforgeeks.org/support-vector-machine-algorithm/
2. https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052
3. https://www.ibm.com/topics/logistic-regression
4. https://towardsdatascience.com/hog-histogram-of-oriented-gradients-67ecd887675f
5. https://en.wikipedia.org/wiki/Local_binary_patterns
6. https://towardsdatascience.com/ensemble-of-classifiers-voting-classifier-ef7f6a5b7795