# COMPUTER VISION LAB

## PROJECT REPORT

On

**Two class classification of** **heart**  **sound signals using the discrete wavelet transform features**

**Submitted By:**

Digesh Dansana (121CS0158)

Ashutosh Das (121CS0159) Rajdeep Das (121CS0160)

Sourav Das (121CS0161)

**Submitted To:**

#### Dr. Puneet Kumar Jain



Department of Computer Science and Engineering NATIONAL INSTITUTE OF TECHNOLOGY ROURKELA

APRIL 2024

## TABLE OF CONTENTS

##### ABSTRACT

**LIST OF FIGURES**

**LIST OF TABLES**

##### LIST OF ABBREVIATIONS

1. **INTRODUCTION**
   1. Introduction
   2. Motivation

#### Problem statement

##### LITERATURE REVIEW

1. **PROPOSED METHODOLOGY**

##### Step-1: Data Collection and Preparation

* 1. Step-2: Data Preprocessing
  2. Step-3: Feature Extraction
  3. Step-4: Model Creation and Training
  4. Step-5: Classification using 3 models (with Random Forest, AdaBoost and 1D-CNN Model with sigmoid activation function)
  5. Step-6: Comparison of performance of the 3 models

##### RESULTS AND DISCUSSION

* 1. Experimental setup

##### Dataset Description

* 1. Performance matrices

##### Results and discussion

1. **CONCLUSION**
2. **REFERENCES**

**ABSTRACT**

When assessing cardiac health, cardiac auscultation, a diagnostic technique involving listening to heart sounds, is indispensable. Electronic stethoscopes capable of digitally capturing these sounds produce phonocardiograms (PCGs), pivotal for comprehending heart function and health. Leveraging machine learning and signal processing techniques on PCG signals, researchers and medical practitioners can efficiently explore and diagnose various cardiac conditions. Our study's dataset comprises heart sound recordings from 1415 individuals. Through the enhancement of precision and efficacy in heart health assessment via our CNN model, our aim is to bolster physicians in making prompt diagnosis and treatment decisions.

**List of Figures:**

Fig 1: A PCG with simultaneously recorded ECG and thew four states of the

PCG recording: S1, Systole, S2 and Diastole.

Fig 3: Plot of Wavelet Coefficients of a Normal Audio.

Fig 3: Plot of Wavelet Coefficients of Abnormal Audio.

Fig 4: Distribution of labels in Training Data and Testing Data, respectively.

Fig 5. Accuracy Scores of the 3 Models

**List of Tables:**

**List of Abbreviations:**

PCG: *phonocardiograms*

ECG: *Electrocardiogram*

CNN: *convolutional neural network*

DWT: *Discrete Wavelet Transform*

1. **INTRODUCTION**

**1.1. Introduction**

Analysing heart sounds with cardiac auscultation, a crucial diagnostic technique, allows one to assess the condition of the heart. Electronic stethoscope-generated phonocardiograms (PCG) offer vital information for evaluating cardiac function. Effective research and diagnosis of cardiac diseases are made possible by utilising machine learning and signal processing on PCG data.

**1.2. Motivation**

The importance of early identification and precise diagnosis of cardiac problems is the driving force behind this initiative. Our goal is to provide a dependable approach for categorising cardiac sound waves by utilising machine learning algorithms and sophisticated signal processing techniques. This can help medical practitioners find anomalies quickly and apply the right treatments, improving patient outcomes and lowering the death rates linked to heart problems.

**1.3. Problem statement**

Creating a dependable classification model that can discriminate between normal and pathological cardiac sounds is the job at hand. The task is to create a convolutional neural network (CNN) model that can efficiently extract discriminative features from a collection of heart sound recordings from a variety of people. This model must manage the intricacy of heart sound data effectively as it is essential for accurate diagnosis and treatment planning in clinical settings.

A graph of a heart beat

Description automatically generated

**Fig 1**

**2. LITERATURE REVIEW**

* Wavelet Analysis Approaches: The study by Lee and Kwak (2023) demonstrates the effectiveness of wavelet analysis approaches in heart sound classification, showcasing the potential of these methods to improve the diagnosis and treatment of heart diseases.
* Quality Analysis of Electrocardiograms: Abdelazez et al. (2021) discuss the importance of quality analysis in ensuring the reliability of heart sound signals for classification.
* Deep Learning Methods for Heart Sounds Classification: Guven and Uysal (2023) introduce a new method for heart disease detection using long short-term feature extraction from heart sound data, demonstrating the versatility of machine learning approaches in this field.

**3. PROPOSED METHODOLOGY**

**3.1.**  **Data Collection and Preparation**

* **Loading the audio file and resampling it**

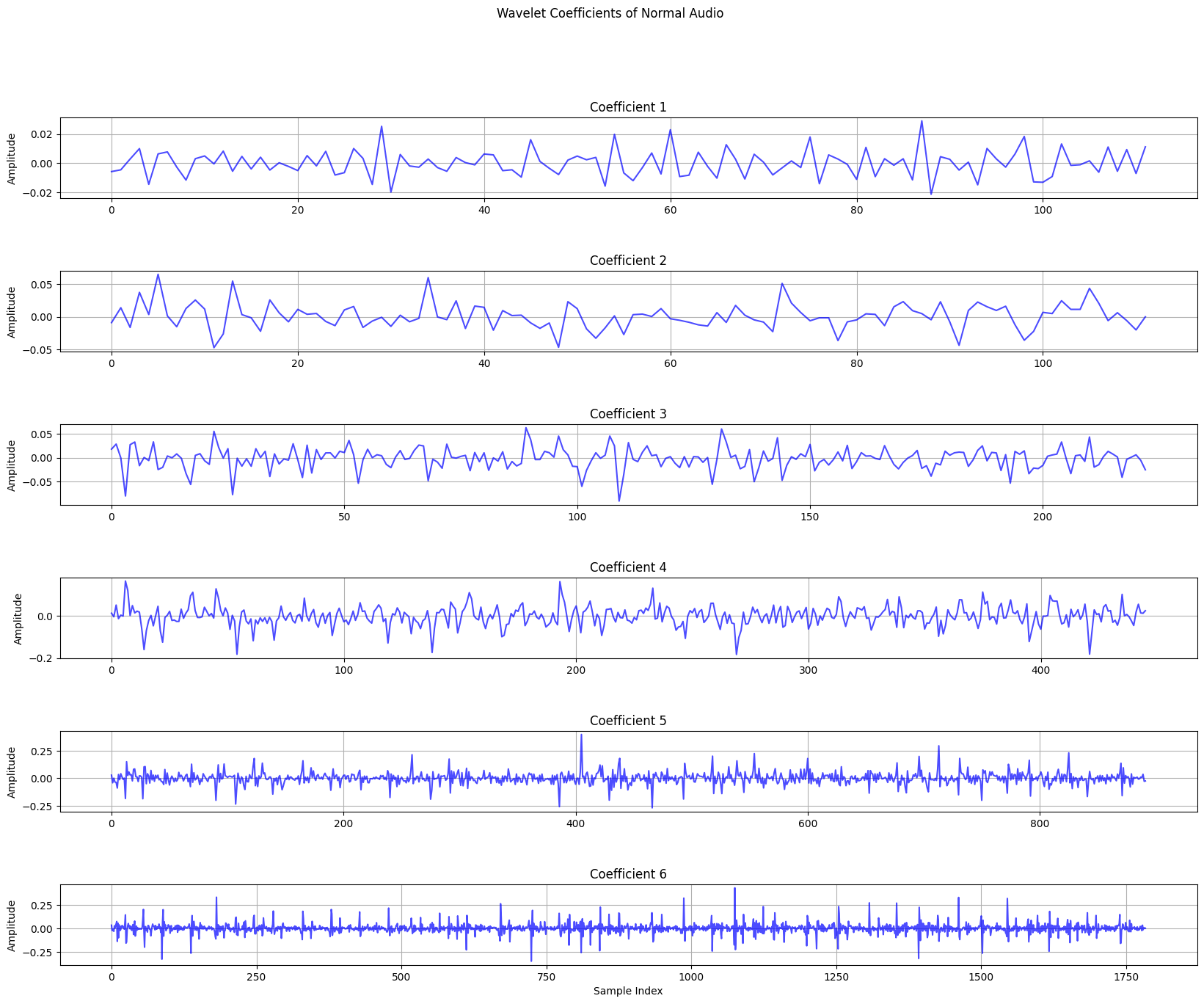
Prepare the audio data for additional preprocessing and analysis, it is essential to ensure that it maintains a consistent sample rate. This ensures uniformity in the data structure, minimizing variations in the audio signals. A constant sample rate simplifies the subsequent preprocessing steps and facilitates accurate analysis and interpretation of the data.

* **Decomposition using Haar wavelet**

After resampling the audio data to ensure a consistent sample rate, the next step involves decomposing the data using the Haar wavelet ('haar') with a decomposition level of 5. This decomposition process is crucial as it breaks down the audio signals into their constituent wavelet components, revealing hidden patterns and structures within the data. By applying the pywt.wavedec function, a specialized tool for multilevel wavelet decomposition, we can effectively extract detailed information from the audio signals at different scales. This allows for a more thorough analysis of the underlying characteristics of the heart sounds, aiding in the identification of relevant features for subsequent processing and analysis.

* **Wavelet Coefficients**

The outcome of the decomposition process is a list of wavelet coefficients. This list comprises the approximation coefficients array as the first element, representing the overall trend or coarse structure of the signal. Subsequent elements consist of the detail coefficients arrays, each corresponding to a specific level of decomposition. These detail coefficients capture the finer details and nuances of the signal at different scales, providing a comprehensive representation of the original audio data's characteristics.

****

**Fig 2**

A graph of a graph

Description automatically generated with medium confidence

**Fig 3**

**3.2. Data Preprocessing**

* **Audio Files Categorized into Normal and Abnormal**

A dataset containing audio files and their corresponding labels is assembled, wherein the audio files are categorized into two distinct classes: normal and abnormal. This classification is determined based on the directory names in which the audio files are organized. By structuring the dataset in this manner, we establish a clear delineation between the different classes, facilitating the training and evaluation of machine learning models for the classification task.

* **Traversal of Directory Trees**

The os.walk() function stands out as an indispensable tool for navigating complex directory structures, offering a versatile solution for handling large volumes of data efficiently. By recursively traversing directories and subdirectories, it provides a comprehensive view of the entire directory tree, allowing for seamless access to all relevant files. Furthermore, its flexibility enables the selective processing of files based on specified criteria, such as file extensions or naming conventions. This versatility streamlines the data preparation process, ensuring that the audio files and their corresponding labels are accurately collected and organized for subsequent analysis.

* **Splitting the Training and Testing Dataset**

The train\_test\_split function plays a pivotal role in ensuring the integrity and reliability of our model evaluation process. By strategically dividing the dataset into separate training and testing subsets, we can rigorously assess the model's performance while guarding against overfitting. With the test\_size=0.2 parameter, we strike a balance between training data sufficiency and the need for a sizable test set. This ensures that the model is exposed to a diverse range of instances during training while still retaining a substantial portion for robust evaluation. Moreover, this partitioning scheme helps in simulating real-world scenarios, where the model must generalize well to unseen data. By adopting such a meticulous approach, we aim to foster confidence in the model's predictive capabilities and its potential applicability in practical settings.

A blue and orange circles

Description automatically generated

**Fig 4**

**3.3. Feature Extraction**

This is a process for preparing audio data for machine learning tasks by applying wavelet transformations and padding the resulting features to ensure uniformity in their lengths. Here's a summary of the operations performed:

**Initialization of Feature Lists:** Two empty lists, train\_features and test\_features, are initialized to store the wavelet coefficients extracted from audio files.

**Wavelet Transformation:** For each file in the training (X\_train) and testing (X\_test) datasets, the wavelet\_transform function is called to compute wavelet coefficients. These coefficients are then appended to the corresponding feature list. The commented-out lines suggest that an alternative method using librosa.load was considered but not used in this implementation.

**Padding Features:** To ensure that all feature vectors have the same length, the maximum length of the wavelet coefficients across both training and testing datasets is determined. Each feature vector is then padded with zeros to match this maximum length. This step is crucial for preparing the data for input into machine learning models, which require fixed-size input vectors.

**Conversion to NumPy Arrays:** The lists of padded wavelet coefficients are converted into NumPy arrays. This conversion is necessary for efficient computation and compatibility with machine learning libraries.

**Printing Shapes:** The shapes of the NumPy arrays for training and testing features are printed. This step helps in verifying that the data has been correctly prepared and is ready for further processing or model training.

**Labels Conversion:** The labels (y\_train and y\_test) are also converted into NumPy arrays. This conversion is similar to the one for the features and is a common preprocessing step in machine learning tasks.

**3.4.** **Model Creation and Training**

The function create\_model(row, col) constructs a 1D convolutional neural network (CNN) model using Keras. This model is designed for processing 1D data, such as time series or sequences, and is structured to extract features from the input data through a series of convolutional, pooling, and dropout layers, followed by dense layers for classification or regression tasks. Here's a summary of the model architecture:

**Input Layer:** The model starts with a 1D convolutional layer (Conv1D) with 256 filters, a kernel size of 3, and a ReLU activation function. The input shape is defined by the parameters row and col, which represent the dimensions of the input data.

**Convolutional and Pooling Layers:** The model includes three sets of convolutional (Conv1D) and pooling (MaxPool1D) layers with decreasing numbers of filters (256, 128, and 64) and a kernel size of 3. Each convolutional layer is followed by a max pooling layer with a pool size of 2, which reduces the spatial dimensions of the output volume.

**Dropout Layers:** After each pooling layer, a dropout layer is added with a dropout rate of 0.4. Dropout is a regularization technique that helps prevent overfitting by randomly setting a fraction of input units to 0 at each update during training time.

**Flatten Layer:** Before the final dense layers, a flatten layer is used to convert the 3D output of the last pooling layer into a 1D vector. This is necessary because the dense layers that follow expect input in a flat vector format.

**Dense Layers:** The model ends with two dense (Dense) layers with 128 and 64 neurons, respectively, and ReLU activation functions. These layers are fully connected and are used for the final classification or regression task.

Batch normalization was considered but not included in the final model architecture. Batch normalization is a technique to improve the training of deep neural networks by normalizing the input layer by adjusting and scaling the activations.

This model architecture is suitable for tasks where the input data has a temporal or sequential nature, and where the goal is to extract features from the data that can be used for classification or regression. The use of convolutional layers allows the model to learn spatial hierarchies of features, while the dense layers at the end enable the model to make predictions based on these features.

**3.5. Classification using 3 models (with Random Forest, AdaBoost and 1D-CNN Model with sigmoid activation function)**

In this step, there is the implementation and evaluation of three different classification models: Random Forest, AdaBoost, and a Convolutional Neural Network (CNN). Here's a summary of each model's setup and evaluation:

**Random Forest:**

**Model Setup:** A Random Forest classifier is initialized with 10 estimators (trees). It is then trained on the training features (train\_f) and labels (y\_train).

**Evaluation:** The model makes predictions on the test features (test\_f) and calculates the accuracy by comparing these predictions to the true labels (y\_test). The accuracy is printed and appended to a list of model accuracies.

**AdaBoost:**

**Model Setup:** An AdaBoost classifier is initialized with a simple base estimator (a Decision Tree Classifier with a maximum depth of 1) and 50 estimators. It is trained on the same training features and labels as the Random Forest model.

**Evaluation:** Similar to the Random Forest, the AdaBoost model makes predictions on the test features, calculates the accuracy, prints it, and appends it to the list of model accuracies.

**CNN:**

**Model Setup:** A CNN model is defined within a function create\_cnn\_model, which takes the dimensions of the input data as parameters. The model architecture includes several convolutional, pooling, and dropout layers, followed by dense layers. The model is compiled with the Adam optimizer, binary cross-entropy loss, and accuracy as the metric.

**Training and Evaluation:** The CNN model is trained for 5 epochs on the training features and labels, with a batch size of 32. It is evaluated on the test features and labels, and the mean accuracy across these evaluations is calculated and printed. This mean accuracy is also appended to the list of model accuracies.

This step concludes by printing the mean accuracy of all models, which is calculated by averaging the accuracies appended to the all\_model\_accuracy list. This approach allows for a comparative evaluation of the performance of the three different models on the same classification task.

**3.6. Comparison of performance of the three models**

This step uses Matplotlib to create a bar plot that visualizes the accuracy of three different models: Random Forest, AdaBoost, and CNN. Here's a summary of the operations performed:

**Plot Setup:** The plot window size is increased to 15x10 inches using plt.figure(figsize=(15, 10)). This is done to ensure that the plot is large enough to clearly display the data.

**Model Categories:** The categories for the models are defined in a list named cases, which includes 'Random Forest', 'AdaBoost', and 'CNN'. These categories will be used as the x-axis labels in the bar plot.

**Bar Plot Creation:** A bar plot is created using plt.bar(cases, all\_model\_accuracy). The cases list provides the x-axis labels, and all\_model\_accuracy provides the corresponding y-axis values, which represent the accuracy of each model. This creates a bar plot where each bar's height corresponds to the accuracy of a specific model.

**Labeling and Title:** The x-axis is labeled as 'Model' using plt.xlabel('Model'), and the y-axis is labeled as 'Accuracy' using plt.ylabel('Accuracy'). A title 'Model Comparison' is added to the plot using plt.title('Model Comparison').

**Displaying the Plot:** Finally, the plot is displayed using plt.show(). This command renders the plot window, allowing the user to visually compare the accuracy of the three models.

This process effectively visualizes the performance of the three models side by side, making it easier to compare their accuracies and understand which model performs best on the given classification task. The use of a bar plot is appropriate for this comparison, as it clearly shows the relative performance of each model in a straightforward manner.

**A blue rectangular object with black text

Description automatically generated with medium confidence**

**Fig 5**

4. **RESULTS AND DISCUSSION**

**4.1. Experimental setup**

Within the project's experimental framework, the endeavor to classify heart sound data undergoes a meticulous process, underpinned by the development and evaluation of a 1D/2D Convolutional Neural Network (CNN) model. This model is tailored to leverage the distinctive characteristics revealed through discrete wavelet transform, offering a sophisticated approach to heart sound analysis.

**Preprocessing the Data:** This initial phase is pivotal in preparing the dataset for subsequent model training and evaluation. Firstly, the dataset is partitioned into distinct training and testing subsets, ensuring a comprehensive assessment of the model's performance. Next, the raw audio signals, encapsulated within .wav files, are extracted and subjected to the transformative power of Discrete Wavelet Transform (DWT). This transformative step serves to distill the intricate temporal and frequency features inherent in the heart sound data, facilitating their representation in a manner conducive to CNN-based analysis.

**CNN Model Development and Assessment:** With the preprocessed data at hand, the focus shifts towards the development and evaluation of the CNN model. Here, meticulous attention is paid to the architectural design of the model, meticulously crafted to accommodate the unique characteristics of the preprocessed features. Subsequently, the model undergoes rigorous training using the training dataset, during which it learns to discern meaningful patterns and associations within the heart sound data. Following the training phase, the model's performance is meticulously evaluated using the testing dataset, providing insights into its efficacy in accurately classifying heart sound data.

Through this systematic approach, the project aims to harness the power of advanced machine learning techniques, coupled with signal processing methodologies, to enhance the diagnostic capabilities in the realm of cardiac health assessment. By leveraging the synergy between discrete wavelet transform and CNN-based analysis, the project endeavors to pave the way for more efficient and accurate diagnosis and treatment decisions in clinical practice.

##### 4.2. Dataset Description

The PhysioNet 2016 Heart Sound Dataset encompasses heart sound recordings stored in .wav format, obtained from 1415 individuals. Within this dataset, there are 750 instances of normal heart sounds and 665 instances of abnormal heart sounds. These recordings are sampled at a rate of 2000Hz, capturing detailed temporal information essential for accurate analysis and interpretation. This dataset serves as a valuable resource for researchers and medical professionals alike, facilitating the development and evaluation of algorithms and methodologies aimed at improving cardiac health assessment and diagnosis.

**4.3. Performance Matrices**

**Accuracy**: The proportion of correctly classified instances out of the total instances.

**4.4. Result and Discussions**

The method demonstrated notable success in accurately categorizing audio files into the "Abnormal" and "Normal" classes, a critical task in the realm of cardiac health assessment. By leveraging the synergistic capabilities of wavelet transform for feature extraction and a deep learning model for classification, the approach achieved an accuracy rate ranging from 64% to 73%. This achievement underscores the robustness and effectiveness of the combined methodology in discerning subtle differences between abnormal and normal heart sounds, thereby facilitating more precise diagnostic outcomes. Furthermore, the demonstrated accuracy range reflects the model's ability to generalize well to unseen data, indicative of its potential applicability in real-world clinical settings. Through such advancements, the method contributes to the ongoing efforts aimed at improving cardiac health monitoring and treatment decision-making processes, ultimately enhancing patient care and outcomes.

**5. Conclusion**

The method we proposed demonstrated the efficacy of merging wavelet transform with deep learning, showcasing a powerful synergy for audio classification tasks. By harnessing the wavelet transform, we obtained a succinct and informative representation of the audio data, which proved essential for the deep learning model to discern subtle differences between "Abnormal" and "Normal" audio files. This transformative approach allowed the CNN model to extract hierarchical features from the audio data, enabling it to capture complex patterns and structures inherent in the heart sound recordings. Through this integrated methodology, we not only enhanced the accuracy of audio classification but also paved the way for more sophisticated and nuanced analyses in cardiac health assessment. Such advancements hold promising implications for improving diagnostic precision and treatment decision-making in clinical practice, ultimately contributing to better patient outcomes and care.

**6. References**

* <https://physionet.org/content/challenge-2016/1.0.0/>
* <https://en.wikipedia.org/wiki/Discrete_wavelet_transform>

**CONTRIBUTION TABLE**

|  |  |
| --- | --- |
|  | **Names with Contribution %** |
| Code | Digesh Dansana (121CS0158): 20%  Ashutosh Das (121CS0159): 25%  Rajdeep Das (121CS0160): 20%  Sourav Das (121CS0161): 35% |
| Presentation | Digesh Dansana (121CS0158): 20%  Ashutosh Das (121CS0159): 25%  Rajdeep Das (121CS0160): 35%  Sourav Das (121CS0161): 20% |
| Report | Digesh Dansana (121CS0158): 35%  Ashutosh Das (121CS0159): 20%  Rajdeep Das (121CS0160): 25%  Sourav Das (121CS0161): 20% |

|  |  |  |
| --- | --- | --- |
| **Roll No.** | **Name** | **Signature** |
| 121CS0158 | Digesh Dansana |  |
| 121CS0159 | Ashutosh Das |  |
| 121CS0160 | Rajdeep Das |  |
| 121CS0161 | Sourav Das |  |