

# **Computer Vision Lab**

## **Project Report**

On

**Five-class classification of heart sound signals using  
the short-term Fourier transform features.**

Submitted by:

Kanhaiya Agarwal (121CS0134)

Yash Agarwal (121CS0135)

Suzen Akhtar (121CS0136)

Khair Alanam (121CS0137)

Submitted to:

Dr. Puneet Kumar Jain



Department of Computer Science and Engineering  
NATIONAL INSTITUTE OF TECHNOLOGY ROURKELA  
APRIL, 2024

## **Abstract:-**

### **Background:**

- Cardiovascular disease (CVD) is a leading cause of death globally, responsible for about 17.5 million deaths annually.
- Cardiac auscultation, analyzing heart sounds, is crucial for diagnosing heart disorders.
- Electronic stethoscopes capture heart sounds digitally, producing phonocardiograms (PCG), and aiding in diagnosis.

### **Objective:**

- Develop a Convolutional Neural Network (CNN) model to classify heart sound signals into five categories: normal and four abnormal conditions (AS, MVP, MS, MR).
- Utilize the Yaseen Khan Dataset, comprising heart sound signals sampled at 8000Hz, with each category containing 100 normal signals and 400 signals for each abnormal category.

### **Methodology:**

- Train and evaluate the proposed CNN model on the Yaseen Khan Dataset.
- Implement signal processing and machine learning techniques on PCG signals for efficient study and diagnosis of cardiac conditions.

### **Dataset Details:**

- Yaseen Khan Dataset contains heart sound signals sampled at 8000Hz.
- Categories: Normal (100 signals), AS (400 signals), MVP (400 signals), MS (400 signals), MR (400 signals).

### **CNN Model Development:**

- Train the CNN model to accurately classify heart sound signals into normal and abnormal categories.

- Leverage signal processing and machine learning techniques to enhance model performance.

### Evaluation:

- Assess the performance of the CNN model using appropriate metrics such as accuracy, precision, recall, and F1-score.
- Evaluate the model's ability to distinguish between normal and abnormal cardiac conditions.

### Significance:

- Findings contribute to the advancement of automated cardiac diagnostic systems.
- Enhance the efficiency and accuracy of heart disorder diagnosis in clinical settings.
- Provide promising results for the classification of heart sound signals, aiding in the early detection and treatment of cardiac conditions.

### List of Figures:-

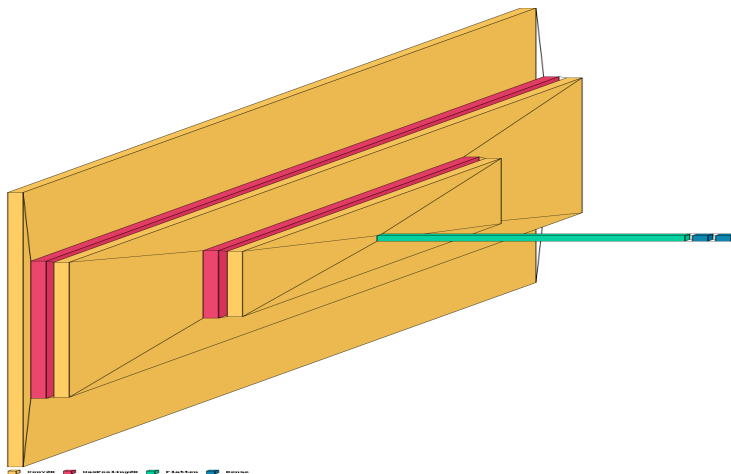
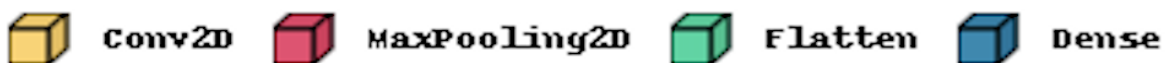


Fig-1: This is Model -1 Architecture



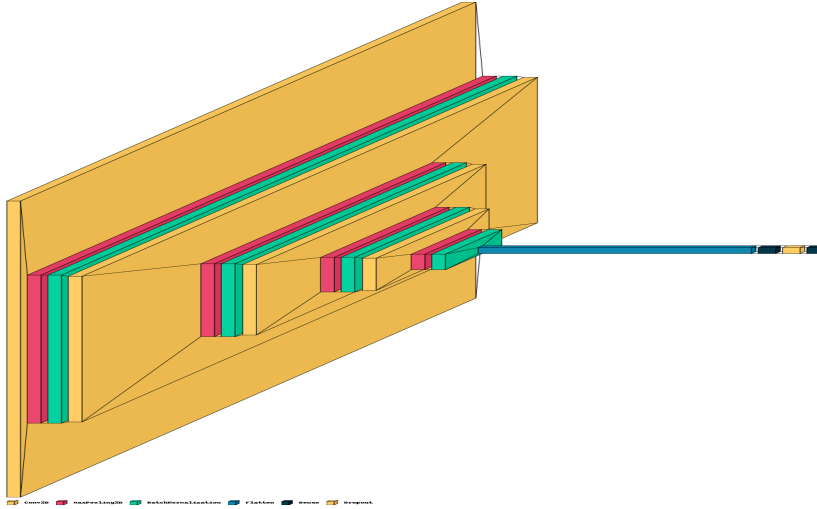
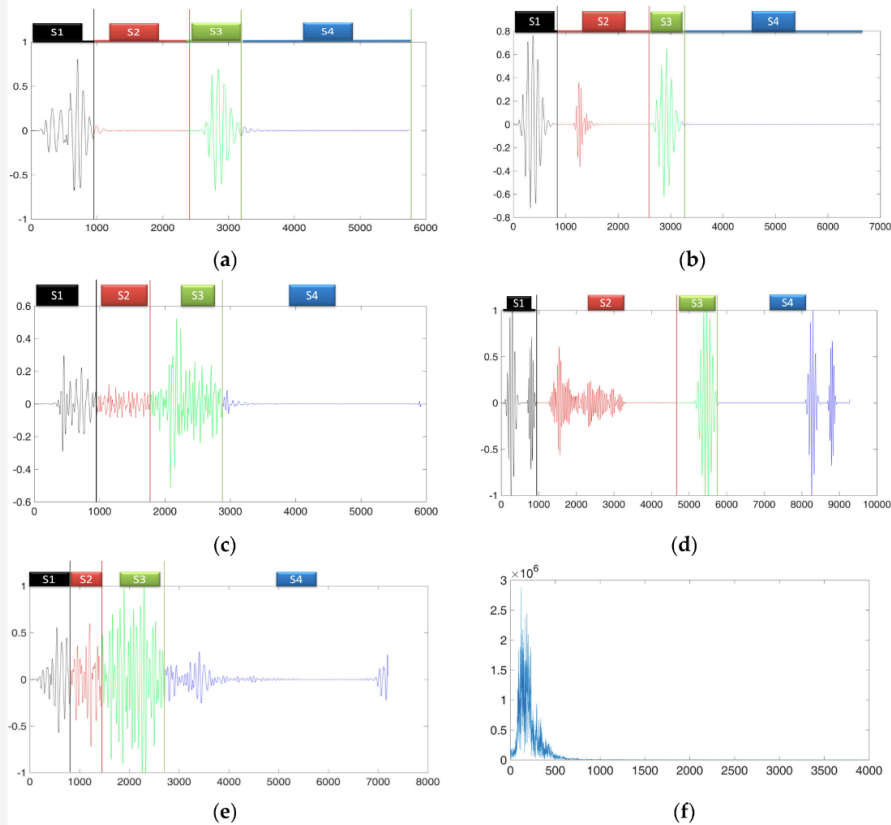


Fig-2: Model-2 Architecture

**Figure 1.** (a) A normal heart sound signal; (b) Murmur in systole (MVP); (c) Mitral Regurgitation (MR); (d) Mitral Stenosis (MS); (e) Aortic Stenosis (AS); (f) Spectrum of a PCG signal.



## List of Tables:-

Heart Sound Signals	No. of audio files in each heart sound signal
AS (Aortic Stenosis)	100
MVP (Mitral Regurgitation)	100
MS (Mitral Stenosis)	100
MR (Murmur in systole)	100
N (Normal)	100

## Introduction:-

Cardiac auscultation, a diagnostic technique involving the analysis of heart sounds, is crucial in assessing cardiac health. Electronic stethoscopes, capable of recording heart sounds digitally, produce phonocardiograms (PCG), which carry vital information about heart function and health. Researchers and clinicians can effectively study and diagnose various heart disorders by leveraging signal processing and machine learning techniques on PCG signals.

## Motivation:-

Cardiovascular diseases (CVDs) remain a leading cause of mortality worldwide, underscoring the importance of early and accurate diagnosis in improving patient outcomes. Cardiac auscultation, a traditional method for assessing heart health, relies heavily on the expertise of clinicians to interpret subtle variations in heart sounds. However, this process can be subjective and prone to human error, leading to delays in diagnosis and potentially suboptimal treatment strategies.

The emergence of electronic stethoscopes capable of recording heart sounds digitally presents a transformative opportunity in the field of cardiac diagnostics. By converting heart sounds into digital signals, clinicians gain access to a wealth of data that can be analyzed using advanced signal processing and machine learning techniques. This shift towards data-driven approaches offers several compelling motivations:

**1. Enhanced Diagnostic Accuracy:** Machine learning models have the potential to analyze heart sound signals objectively and with high precision, aiding clinicians in detecting subtle abnormalities that might otherwise go unnoticed during traditional auscultation.

**2. Early Detection of Cardiovascular Conditions:** Timely detection of cardiovascular conditions, such as valvular defects and structural anomalies, is crucial for initiating appropriate interventions and preventing disease progression. Machine learning-based diagnostic tools can facilitate early detection, leading to better patient outcomes and reduced healthcare burdens.

**3. Personalized Medicine:** By leveraging large datasets of heart sound recordings, machine learning models can learn to recognize patterns indicative of specific cardiac pathologies. This personalized approach to diagnosis enables tailored treatment plans based on individual patient characteristics, ultimately improving treatment efficacy and patient satisfaction.

**4. Clinical Workflow Optimization:** Automated analysis of heart sound signals using machine learning models has the potential to streamline clinical workflows, allowing clinicians to focus their time and expertise on complex cases while delegating routine screening and triage tasks to automated systems.

**5. Research and Innovation:** Advances in machine learning techniques applied to cardiac diagnostics not only benefit clinical practice but also drive innovation in the broader field of cardiovascular research. By uncovering novel insights from large-scale datasets, researchers can identify new biomarkers, elucidate disease mechanisms, and develop innovative therapeutic strategies.

## **Problem Statement:-**

Cardiac auscultation, a fundamental technique in diagnosing cardiac conditions, involves analyzing heart sounds to assess cardiovascular health. With the advent of electronic stethoscopes capable of recording heart sounds digitally, there's an opportunity to leverage signal processing and machine learning techniques to enhance diagnostic accuracy and efficiency. In this context, the Yaseen Khan Dataset offers a valuable resource, comprising heart sound recordings sampled at 8000Hz and categorized into five distinct classes: one normal and four abnormal categories (AS, MVP, MS, MR).

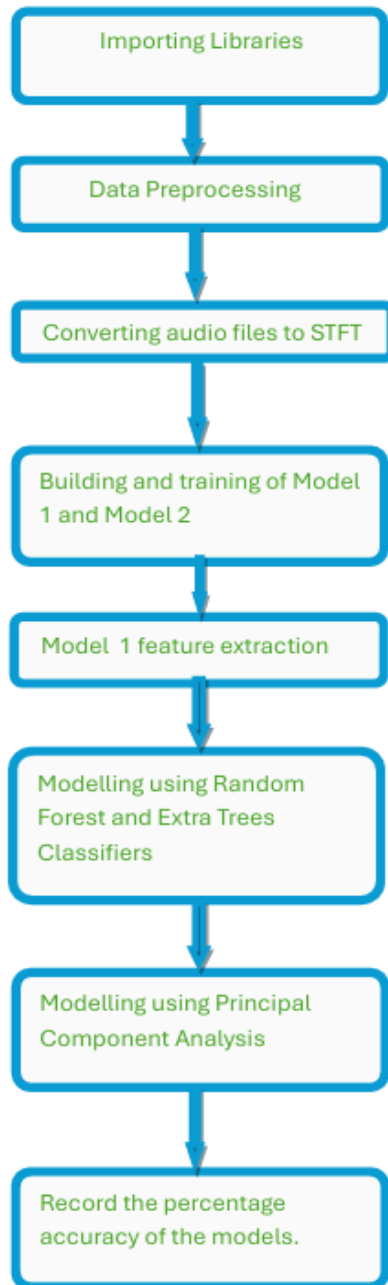
## **Literature Review:-**

Long before the relatively new advent of artificial intelligence algorithms to classify heart sounds, various statistical studies have been performed in healthcare analytics. However, due to the great importance and the special view created by both engineers and physicians on this issue, significant progress has been made in a short period. In this section, the activities performed in heart sound classification and heart disease diagnosis are explicitly examined. Traditional cardiac auscultation has been a mainstay in diagnosing cardiovascular diseases for centuries. Clinicians rely on their expertise to interpret subtle variations in heart sounds using stethoscopes. However, this method is subjective and dependent on the skill and experience of the practitioner, leading to variability in diagnostic accuracy. Despite its widespread use, traditional auscultation faces several challenges, including the difficulty in distinguishing between normal and abnormal heart sounds, the limited sensitivity in detecting subtle abnormalities, and the reliance on subjective interpretation, which can lead to inter-observer variability.

The advent of electronic stethoscopes has revolutionized cardiac auscultation by enabling the recording of heart sounds in digital format. Electronic stethoscopes offer advantages such as improved sound quality, amplification, and the ability to visualize and store recordings for further analysis.

## Proposed Methodologies:-

### Block Diagram of the experimental steps :-





## Explanation of the source code:-

```
# Import libs

import os
import numpy as np
import librosa

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier, ExtraTreesClassifier, BaggingClassifier, VotingClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from tensorflow.keras.models import Model
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

from keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout, BatchNormalization
from keras.callbacks import ModelCheckpoint
```

This part of the code imports necessary libraries for audio classification tasks using machine learning and deep learning models. There are basic libraries, audio processing library (Librosa), Machine learning, and Deep learning libraries. These libraries provide a comprehensive set of tools for building, training, and evaluating machine learning and deep learning models for audio classification tasks. They cover various algorithms and techniques that can be employed depending on the specific requirements and characteristics of the dataset.

```
# Utility functions

def audio_to_stft(audio_file):
    data, sample_rate = librosa.load(audio_file, sr=44100)
    stft_data = np.abs(librosa.stft(data))
    return stft_data
```

This utility function 'audio\_to\_stft' converts an audio file into its Short term Fourier Transform (STFT) representation. First, load the audio file using librosa and set the sample rate to 44100 Hz then compute the STFT of the audio data. It takes the absolute value of STFT data. The function returns the magnitude of the STFT data, which can be used as input features for training the audio classification model.

```

# Collect STFT features and labels

stft_features = []
labels = []
n_frames = 250

for root, dirs, files in os.walk("heart_dataset"):
    for directory in dirs:
        label = os.path.basename(directory)
        category_dir = os.path.join(root, directory)
        for audio_file in os.listdir(category_dir):
            labels.append(label)
            audio_path = os.path.join(category_dir, audio_file)
            stft_data = audio_to_stft(audio_path)
            stft_features.append(stft_data)

```

This code segment is responsible for collecting STFT features and their corresponding labels from audio files stored in the directory under “heart\_dataset”.

```

# Reshape STFT data
new_stft_features = []
for stft_data in stft_features:
    current_frames = stft_data.shape[1]

    if current_frames < n_frames:
        padded_data = np.pad(stft_data, ((0, 0), (0, n_frames - current_frames)), mode='constant')
        new_stft_features.append(padded_data)
    elif current_frames > n_frames:
        trimmed_data = stft_data[:, :n_frames]
        new_stft_features.append(trimmed_data)
    else:
        new_stft_features.append(stft_data)

new_stft_features = np.array(new_stft_features)

```

For each STFT data, it checks the no. of frames. If no. of frames is less than the desired number, then pad the STFT data with zeros to match the desired no. of frames. If no. of frames is greater than the desired number then trim the STFT data to the desired no. of frames else if it is equal then convert the list into numpy array.

After executing this code segment, ‘new\_stft\_features’ will contain the reshaped STFT features with consistent dimensions. This feature can be used for training the audio classification model.

```
# Convert labels to categorical
label_encoder = LabelEncoder()
labels_encoded = label_encoder.fit_transform(labels)
labels_categorical = to_categorical(labels_encoded)
```

In this code segment labels are encoded and converted to categorical format.

```
# Split data into train and test sets
X_train, X_test, Y_train, Y_test = train_test_split(new_stft_features, labels_categorical, test_size=0.2,
random state=42)
```

Here we are splitting the data into training and testing sets. We are using 20 percent of the data for the testing set and the rest will be used for training.

After executing this code segment, you will have:

X\_train = Training set features (STFT data)

X\_test = Testing set features.

Y\_train = Training set features (categorical format)

Y\_test = Testing set labels.

```
# Update the number of classes in the output layer
num_classes = len(label_encoder.classes_)

# Define model
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(1025, n_frames, 1)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    Flatten(),
    Dense(64, activation='relu'),
    Dense(num_classes, activation='softmax')
])

model.summary()
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

In this code segment, the number of classes in the output layer is updated based on the number of unique classes obtained from the label encoder.

The model architecture is defined sequentially using the Keras Sequential API. Three convolutional layers (Conv2D) with ReLU activation are used for feature extraction, two MaxPooling2D layers are added to reduce spatial dimensions. The convolutional layers are followed by a Flatten layer to convert the 2D feature maps into a 1D vector. Two fully connected (Dense) layers are added for classification, where the last dense layer has num\_classes units and uses softmax activation to output class probabilities.

The model summary is printed to provide an overview of the model architecture. Finally, the model is compiled with the Adam optimizer, categorical cross-entropy loss function, and accuracy metric for evaluation during training.

```
import visualkeras

visualkeras.layered_view(model, to_file='output_model1.png', legend = True)
```

This code snippet utilizes the VisualKeras library to create a layered visualization of the Model 1 architecture.

```
# Train the model
history = model.fit(
    np.array(X_train),
    np.array(Y_train),
    epochs=10, batch_size=32,
    validation_data=(np.array(X_test), np.array(Y_test))
)

# Evaluate the model
loss, accuracy = model.evaluate(np.array(X_test), np.array(Y_test))
print("Test Accuracy:", accuracy * 100, "%")
```

Training and evaluation of the 'Model' is done using training and testing data.

```
# Define Model 2
model2 = Sequential([
    Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(1025, n_frames, 1)),
    MaxPooling2D(pool_size=(2, 2)),
    BatchNormalization(),
    Conv2D(64, kernel_size=(3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),
    BatchNormalization(),
    Conv2D(128, kernel_size=(3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),
    BatchNormalization(),
    Conv2D(128, kernel_size=(3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),
    BatchNormalization(),
    Flatten(),
    Dense(256, activation='relu'),
    Dropout(0.5),
    Dense(num_classes, activation='softmax')
])

model2.summary()
model2.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

The second model architecture is defined sequentially using the Keras Sequential API. Four convolutional layers (Conv2D) with ReLU activation are used for feature extraction of different filters like 32, 64, 128, etc, four MaxPooling2D layers are added with a pool size of (2, 2) to reduce spatial dimensions, and Batch Normalization is added after each convolutional layer to stabilize and speed up the training process and allows higher learning rates. The convolutional layers are followed by a Flatten layer to convert the 2D feature maps into a 1D vector. Two fully connected (Dense) layers are added for classification, where the last dense layer has num\_classes units and uses softmax activation to output class probabilities and a dropout layer with a dropout rate of 0.5 to prevent overfitting.

The model summary is printed to provide an overview of the model architecture. Finally, the model is compiled with the Adam optimizer, categorical cross-entropy loss function, and accuracy metric for evaluation during training.

```
import visualekera

visualekera.layered_view(model2,to_file='output_model2.png',legend = True)
```

This code snippet utilizes the VisualKeras library to create a layered visualization of the Model 2 architecture. The 'legend= True' argument is employed to incorporate a legend in the visualization, providing additional details about the layers and their parameters.

```
# Train Model 2
history2 = model2.fit(
    np.array(X_train),
    np.array(Y_train),
    epochs=10, batch_size=32,
    validation_data=(np.array(X_test), np.array(Y_test))
)

# Evaluate Model 2
loss, accuracy = model2.evaluate(np.array(X_test), np.array(Y_test))
print("Test Accuracy:", accuracy * 100, "%")
```

Training and evaluation of 'Model 2' is done using training and testing data.

```
feat_extract = Model(
    inputs=model.inputs,
    outputs=model.layers[-2].output
)

# Predict the features
feat_train = feat_extract.predict(X_train)
feat_test = feat_extract.predict(X_test)

# Model with different ensemble methods
ensemble_methods = {
    "Random Forest Classifier": RandomForestClassifier(n_estimators=100, random_state=42),
    "Extra Trees Classifier": ExtraTreesClassifier(n_estimators=100, random_state=42)
}

# Model and predict using different ensemble methods
for method in ensemble_methods:
    print(f"Using {method}:")
    ensemble_methods[method].fit(feat_train, Y_train)

    method_predict = ensemble_methods[method].predict(feat_test)
    accuracy = accuracy_score(Y_test, method_predict)
    f1 = f1_score(Y_test, method_predict, average='weighted')
    print(f"Accuracy using {method} = {accuracy * 100}%")
    print(f"F1 score using {method} = {f1}")
```

This code performs feature extraction using a pre-trained model (model 1) and utilizes the extracted features to train and evaluate different classifiers within the ensemble methods, such as the Random Forest Classifier and Extra Trees Classifier.

1. feat\_extract = Model(inputs=model.inputs, outputs=model.layers[-2].output): This creates a new model (feat\_extract) that takes the same input as the original model (model) but outputs the features from the second-to-last layer of the model. These features are extracted for both the training and testing data.
2. feat\_train = feat\_extract.predict(X\_train): The features are extracted from the training data (X\_train) using the feat\_extract model.
3. feat\_test = feat\_extract.predict(X\_test): The same process is applied to the testing data (X\_test).

Classifiers for Ensemble Methods: Two ensemble methods are instantiated, namely Random Forest Classifier and Extra Trees Classifier, each with 100 estimators.

#### Model and Predict using Different Ensemble Methods:

The classifiers within the ensemble methods are trained on the extracted features from the training data ('feat\_train') and then used to predict the labels for the extracted features from the testing data ('feat\_test'). The accuracy and F1 score of each ensemble method are calculated and printed.

This code demonstrates the process of utilizing extracted features from pre-trained deep learning models to train and evaluate different classifiers within ensemble methods for audio classification. Each ensemble method's accuracy and F1 score are printed, allowing for a comparison of their performance.

```
# Model using PCA

from sklearn.decomposition import PCA

test accuracies = []
test_f1_scores = []
pca = PCA(n_components=10)
pca.fit(X_train_vgg_features)
feat_train_reduced = pca.transform(X_train_vgg_features)
feat_test_reduced = pca.transform(X_test_vgg_features)
```

Here we import the PCA module from scikit-learn, which is used for dimensionality reduction. PCA is instantiated with 10 principal components. PCA is fit on the training features and the training features are transformed into a lower-dimensional space using PCA. The testing features are transformed into the same lower-dimensional space using PCA.

```
for i in range(1, 16):
    test accuracies = []
    test_f1_scores = []
    pca = PCA(n_components=i)
    pca.fit(X_train_vgg_features)
    feat_train_reduced = pca.transform(X_train_vgg_features)
    feat_test_reduced = pca.transform(X_test_vgg_features)
```

### Training and Evaluation:

A loop runs for 'n\_components' ranging from 1 to 15. Within each iteration, PCA is instantiated with the current principal components (n\_components=i) and fitted on the training features. The training and testing features are transformed into the lower-dimensional space using PCA.

Note: In the context of the provided code snippet, the VGG features (X\_train\_vgg\_features and X\_test\_vgg\_features) are used as input data for dimensionality reduction using PCA (Principal Component Analysis).

```
for _ in range(5):
    rfc_pca = RandomForestClassifier(n_estimators=100, random_state=42)
    rfc_pca.fit(feat_train_reduced, Y_train)
    pca_predicted = rfc_pca.predict(feat_test_reduced)
    pca_acc = accuracy_score(Y_test, pca_predicted)
    pca_f1 = f1_score(Y_test, pca_predicted, average='weighted')
    test accuracies.append(pca_acc)
    test_f1_scores.append(pca_f1)
pca_mean_acc = np.mean(test accuracies)
pca_mean_f1 = np.mean(test_f1_scores)
print(f"Mean Accuracy with PCA of n_components={i} = {pca_mean_acc * 100}%")
print(f"Mean F1 score with PCA of n_components={i} = {pca_mean_f1}")
print()
```

For each configuration of PCA, the Random Forest Classifier is trained and evaluated 5 times. Here, first Random Forest Classifier is instantiated then the classifier is trained on the reduced training features. The classifier predicts labels



for the reduced testing features then the accuracy of the predictions is computed. The F1 score of the predictions is computed. The mean accuracy and mean F1 score across the 5 iterations are calculated and printed for each n\_components.

We have followed the same experimental procedure with another dataset which contains 1000 audio files.

### **Experimental Setup:-**

Here we have trained and tested 2 models and tried to achieve the maximum accuracy using the Yaseen Khan dataset for experimental setup on Google Colab.

### **Dataset Description:-**

A comprehensive database encompassing heart sound signals (PCG signals) from diverse sources has been curated and comprises five distinct categories. These categories include one normal category and four abnormal categories, facilitating the classification of heart sound signals into normal and abnormal classes. The signals in this dataset are sampled at a frequency of 8000Hz. Total Categories: 5 (1 normal (100), 4 abnormal (400 each 100)) Sampling Frequency: 8000Hz. We have also used one more dataset to train and test the models which consists of 1000 audio files.

### **Results:-**

<b><u>Metrics</u></b>	<b><u>Model 1</u></b>	<b><u>Model 2</u></b>	<b><u>RFC</u></b>	<b><u>ETC</u></b>	<b><u>PCA (n=1)</u></b>
<b><u>Accuracy</u></b>	100%	70.99%	100%	100%	36.1%
<b><u>F1 Score</u></b>	-	-	100%	100%	34.65%
<b><u>Trainable Params</u></b>	60,955,461	26,653,509	-	-	-
<b><u>Non-train</u></b>	0	704	-	-	-

<b><u>able</u></b> <b><u>Params</u></b>					
<b><u>Total</u></b> <b><u>Params</u></b>	60,955,461	26,654,213	-	-	-

## Conclusions:-

PCG signals carry information about the functioning of heart valves during a heartbeat, hence these signals are very important in diagnosing heart problems at an early stage. To detect heart problems with great accuracy, a feature called the Short-Term Fourier Transform feature is used and here we are provided with a Yaseen Khan dataset which is a comprehensive database encompassing heart sound signals (PCG signals) from diverse sources that has been curated and comprises five distinct categories. These categories include one normal category and four abnormal categories, facilitating the classification of heart sound signals into normal and abnormal classes. The signals in this dataset are sampled at a frequency of 8000Hz. The results that have been obtained show clear performance superiority with great maximum accuracy and can help physicians in the early detection of heart disease during auscultation examination.

## References:-

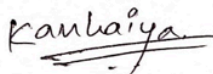
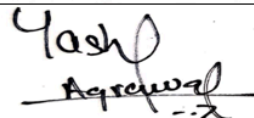

GitHub Repository:

<https://github.com/yaseen21khan/Classification-of-Heart-Sound-Signal-Using-Multiple-Features-/blob/master/README.md>

EDA-Audio Classification Project Using Deep Learning:

<https://www.youtube.com/watch?v=mHPpCXqQd7Y&t=207s>

### Contribution by each team member:-

Name	Tasks			Signature
	Report	Presentation	Source Code	
Kanhaiya Agarwal	10%	5%	40%	
Yash Agarwal	5%	85%	5%	
Suzen Akhtar	80%	5%	5%	
Khair Alanam	5%	5%	50%	