Heart Sound Classification using CNN & Transfer Learning

SANMAY KUMAR JENA-8025340035

THAPAR UNIVERSITY ,PATIALA, PUNJAB

1. *Problem Definition & Scope of the Problem*

This project focuses on the automated classification of heart sound recordings into two diagnostic categories: (Normal) and (Abnormal). Heart sounds provide critical information for early detection of cardiac abnormalities such as murmurs, arrhythmias, and structural defects. Traditional auscultation is subjective and requires clinical expertise. Therefore, this project aims to build a reproducible and automated deep-learning-based diagnostic system.

Scope of the project:
- Preprocess raw PCG (Phonocardiogram) audio signals.
- Convert time-domain waveforms into frequency-domain spectrograms using STFT.
- Train a baseline CNN model and compare it with transfer learning models (MobileNetV2 and ResNet50).
- Evaluate using accuracy, confusion matrix, and precision–recall metrics.
- Present a robust methodology capable of generalizing across unseen heart sound samples.

2. *Literature Review*

Heart sound analysis has gained significant attention over the last decade. Classical approaches involve time-domain analysis, frequency-domain filtering, and handcrafted feature extraction methods such as MFCCs, Wavelets, Hilbert transforms, and envelope extraction.

Recent advancements in deep learning have enabled automatic feature extraction using CNNs directly on spectrogram representations. Several research papers demonstrate that converting PCG signals into 2D images leads to superior performance compared to raw waveforms.

Key insights from literature:
- Frequency-domain features outperform raw waveforms.
- CNNs trained on spectrograms provide high diagnostic precision.
- Transfer learning models such as MobileNetV2 and ResNet50 show strong feature extraction capability even with small datasets.
- STFT-based log spectrograms are widely used in biomedical audio classification.
- Stratified dataset splitting improves model reliability and reduces bias

3.Model Descriptions (Proposed Architecture / Methodology Adopted)

The methodology adopted in this project follows a complete audio-to-deep-learning pipeline:

1. Data Extraction:
- Dataset provided as a ZIP file containing audio files and multiple annotation CSVs.
- A robust CSV parsing mechanism is implemented to detect filename and label columns.
- Labels are normalized to two classes: Normal (0) and Abnormal (1).

2. Preprocessing:
- Audio files may contain varying sampling rates; therefore, all waveforms are resampled to 2000 Hz.
- Signals are trimmed or padded to a fixed duration of 10 seconds.
- Short-Time Fourier Transform (STFT) is generated using FFT size = 256 and hop length = 128.
- Resulting spectrogram magnitude values are normalized and converted to 128×128 images.
- CNN input shape: (128, 128, 1) for grayscale spectrograms.

3. Dataset Preparation:
- Data is split into Train (70%), Validation (15%), and Test (15%) using stratified sampling.
- For transfer learning models, grayscale images are auto-converted to RGB.
- TensorFlow Dataset API is used with batching, shuffling, and prefetching.

4. Models Used:
A. Custom CNN Model
- Conv2D → MaxPool2D (16 filters)
- Conv2D → MaxPool2D (32 filters)
- Conv2D → GAP (64 filters)
- Dense(128) + Dropout(0.3)
- Output: Dense(2, softmax)
- Optimizer: Adam, Loss: Sparse Categorical Crossentropy

B. MobileNetV2 (Transfer Learning)
- Pretrained on ImageNet, frozen feature extractor.
- Added GAP + Dense(128) + Dropout + Softmax layers.

C. ResNet50 (Transfer Learning)
- Pretrained backbone, frozen layers.
- Custom classification head similar to MobileNetV2.

All models are trained for 6 epochs (recommended 20–50 epochs for final performance).

4.Flow Diagram and Technical Aspects

Raw Audio → Resampling → STFT Computation → Spectrogram Normalization → Image Resize → Dataset

Pipeline → CNN/Transfer Learning Training → Evaluation & Visualization.

Technical features include:
- Automatic CSV detection using heuristics
- Robust handling of stereo/mono audio
- TF Dataset pipeline for high throughput
- Matplotlib-based metric visualization

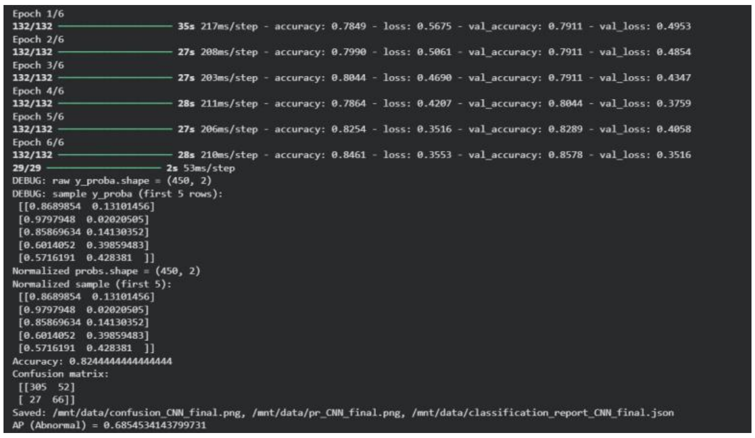
5. Results & Comparative Analysis

Dataset Details:
- Maximum samples used: 3000
- Classes: Normal (0), Abnormal (1)
- Input image size: 128×128 grayscale spectrograms

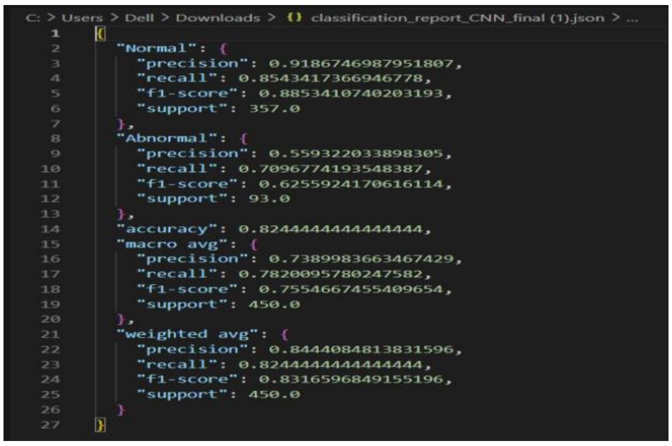
System Configurations:
- Python, TensorFlow, SciPy, NumPy, Matplotlib
- GPU/TPU acceleration recommended

Training Details:
- Batch size: 16
- Optimizer: Adam
- Epochs: 6
- Loss: Sparse Categorical Crossentropy

Model Performance: **CNN**
The project generates the following evaluation outputs:
- Confusion matrix images for each model
- Precision–Recall curves for the Abnormal class
- JSON-based classification reports with precision, recall, F1-score

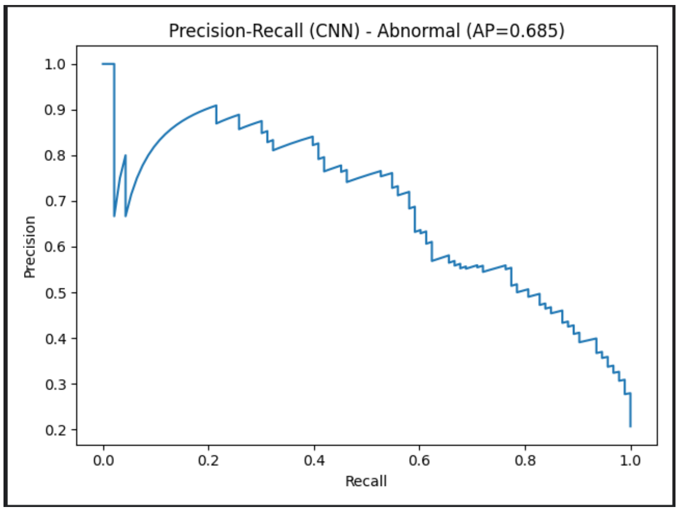


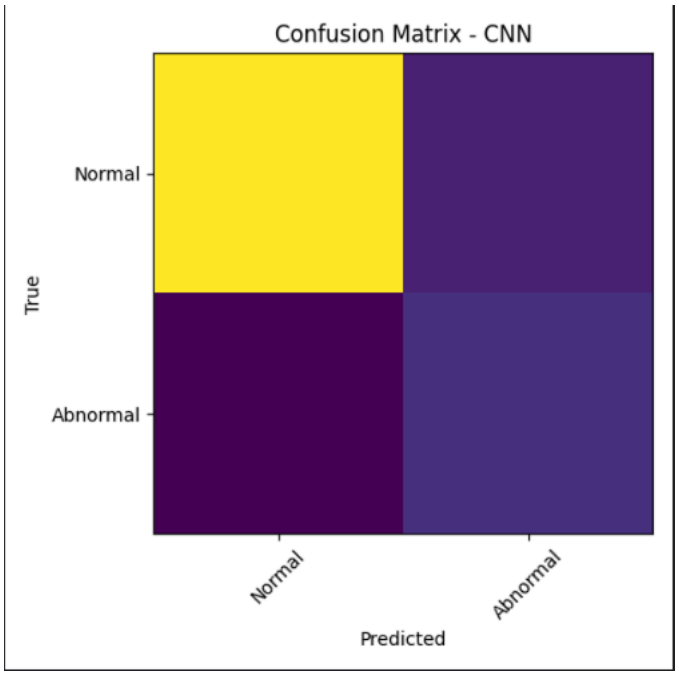
```
Epoch 1/6
132/132 ━━━━━━━━━━━━━━━━━━━━ 35s 217ms/step - accuracy: 0.7849 - loss: 0.5675 - val_accuracy: 0.7911 - val_loss: 0.4953
Epoch 2/6
132/132 ━━━━━━━━━━━━━━━━━━━━ 27s 208ms/step - accuracy: 0.7990 - loss: 0.5061 - val_accuracy: 0.7911 - val_loss: 0.4854
Epoch 3/6
132/132 ━━━━━━━━━━━━━━━━━━━━ 27s 203ms/step - accuracy: 0.8044 - loss: 0.4690 - val_accuracy: 0.7911 - val_loss: 0.4347
Epoch 4/6
132/132 ━━━━━━━━━━━━━━━━━━━━ 28s 211ms/step - accuracy: 0.7864 - loss: 0.4207 - val_accuracy: 0.8044 - val_loss: 0.3759
Epoch 5/6
132/132 ━━━━━━━━━━━━━━━━━━━━ 27s 206ms/step - accuracy: 0.8254 - loss: 0.3516 - val_accuracy: 0.8289 - val_loss: 0.4058
Epoch 6/6
132/132 ━━━━━━━━━━━━━━━━━━━━ 28s 210ms/step - accuracy: 0.8461 - loss: 0.3553 - val_accuracy: 0.8578 - val_loss: 0.3516
29/29 ━━━━━━━━━━━━━━━━━━━━ 2s 53ms/step
DEBUG: raw y_proba.shape = (450, 2)
DEBUG: sample y_proba (first 5 rows):
 [[0.8689854  0.13101456]
 [0.9797948  0.02020505]
 [0.85869634 0.14130352]
 [0.6014052  0.39859483]
 [0.5716191  0.428381  ]]
Normalized probs.shape = (450, 2)
Normalized sample (first 5):
 [[0.8689854  0.13101456]
 [0.9797948  0.02020505]
 [0.85869634 0.14130352]
 [0.6014052  0.39859483]
 [0.5716191  0.428381  ]]
Accuracy: 0.8244444444444444
Confusion matrix:
 [[305  52]
 [ 27  66]]
Saved: /mnt/data/confusion_CNN_final.png, /mnt/data/pr_CNN_final.png, /mnt/data/classification_report_CNN_final.json
AP (Abnormal) = 0.6854534143799731
```



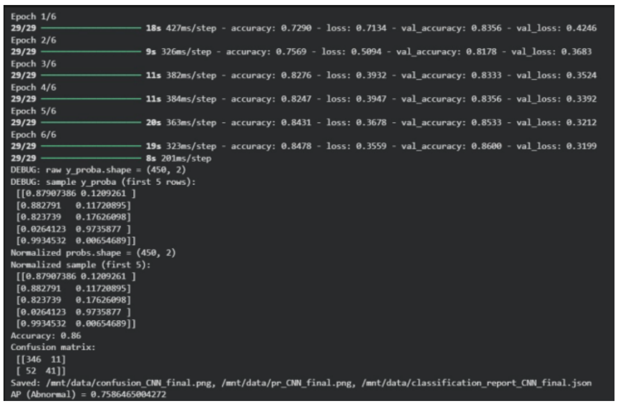
C: > Users > Dell > Downloads > {} classification_report_CNN_final (1).json > ...

```json
{
  "Normal": {
    "precision": 0.9186746987951807,
    "recall": 0.8543417366946778,
    "f1-score": 0.8853410740203193,
    "support": 357.0
  },
  "Abnormal": {
    "precision": 0.559322033898305,
    "recall": 0.7096774193548387,
    "f1-score": 0.6255924170616114,
    "support": 93.0
  },
  "accuracy": 0.8244444444444444,
  "macro avg": {
    "precision": 0.7389983663467429,
    "recall": 0.7820095780247582,
    "f1-score": 0.7554667455409654,
    "support": 450.0
  },
  "weighted avg": {
    "precision": 0.8444084813831596,
    "recall": 0.8244444444444444,
    "f1-score": 0.8316596849155196,
    "support": 450.0
  }
}
```

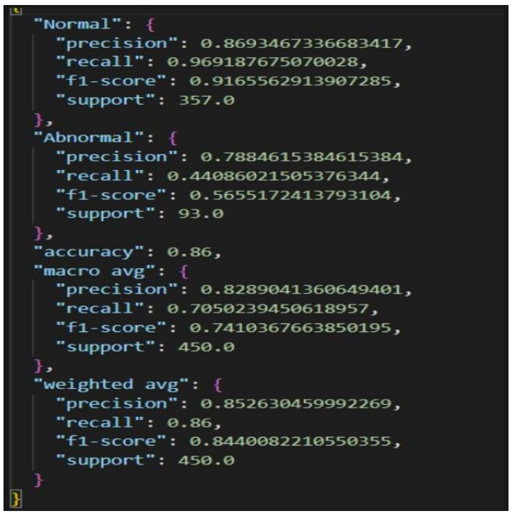




Model Performance: **MOBILENET**

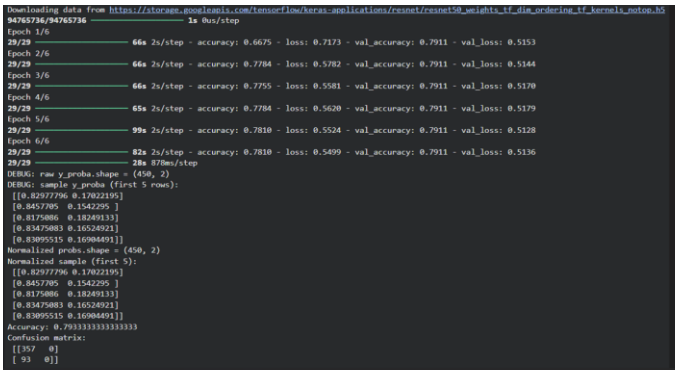


```
Epoch 1/6
29/29 ━━━━━━━━━━━━━━━━━━━━ 18s 427ms/step - accuracy: 0.7290 - loss: 0.7134 - val_accuracy: 0.8356 - val_loss: 0.4246
Epoch 2/6
29/29 ━━━━━━━━━━━━━━━━━━━━ 9s 326ms/step - accuracy: 0.7569 - loss: 0.5094 - val_accuracy: 0.8178 - val_loss: 0.3683
Epoch 3/6
29/29 ━━━━━━━━━━━━━━━━━━━━ 11s 382ms/step - accuracy: 0.8276 - loss: 0.3932 - val_accuracy: 0.8333 - val_loss: 0.3524
Epoch 4/6
29/29 ━━━━━━━━━━━━━━━━━━━━ 11s 384ms/step - accuracy: 0.8247 - loss: 0.3947 - val_accuracy: 0.8356 - val_loss: 0.3392
Epoch 5/6
29/29 ━━━━━━━━━━━━━━━━━━━━ 20s 363ms/step - accuracy: 0.8431 - loss: 0.3678 - val_accuracy: 0.8533 - val_loss: 0.3212
Epoch 6/6
29/29 ━━━━━━━━━━━━━━━━━━━━ 19s 323ms/step - accuracy: 0.8478 - loss: 0.3559 - val_accuracy: 0.8600 - val_loss: 0.3199
29/29 ━━━━━━━━━━━━━━━━━━━━ 8s 201ms/step
DEBUG: raw y_proba.shape = (450, 2)
DEBUG: sample y_proba (first 5 rows):
 [[0.87907386 0.1209261 ]
 [0.882791   0.11720895]
 [0.823739   0.17626098]
 [0.0264123  0.9735877 ]
 [0.9934532  0.00654689]]
Normalized probs.shape = (450, 2)
Normalized sample (first 5):
 [[0.87907386 0.1209261 ]
 [0.882791   0.11720895]
 [0.823739   0.17626098]
 [0.0264123  0.9735877 ]
 [0.9934532  0.00654689]]
Accuracy: 0.86
Confusion matrix:
 [[346  11]
 [ 52  41]]
Saved: /mnt/data/confusion_CNN_final.png, /mnt/data/pr_CNN_final.png, /mnt/data/classification_report_CNN_final.json
AP (Abnormal) = 0.7586465004272
```



```
  "Normal": {
    "precision": 0.8693467336683417,
    "recall": 0.969187675070028,
    "f1-score": 0.9165562913907285,
    "support": 357.0
  },
  "Abnormal": {
    "precision": 0.7884615384615384,
    "recall": 0.44086021505376344,
    "f1-score": 0.5655172413793104,
    "support": 93.0
  },
  "accuracy": 0.86,
  "macro avg": {
    "precision": 0.8289041360649401,
    "recall": 0.7050239450618957,
    "f1-score": 0.7410367663850195,
    "support": 450.0
  },
  "weighted avg": {
    "precision": 0.852630459992269,
    "recall": 0.86,
    "f1-score": 0.8440082210550355,
    "support": 450.0
  }
}
```

Model Performance: **RESNET50**



```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94765736/94765736 ━━━━━━━━━━━━━━━━━━━━ 1s 0us/step
Epoch 1/6
29/29 ━━━━━━━━━━━━━━━━━━━━ 66s 2s/step - accuracy: 0.6675 - loss: 0.7173 - val_accuracy: 0.7911 - val_loss: 0.5153
Epoch 2/6
29/29 ━━━━━━━━━━━━━━━━━━━━ 66s 2s/step - accuracy: 0.7784 - loss: 0.5782 - val_accuracy: 0.7911 - val_loss: 0.5144
Epoch 3/6
29/29 ━━━━━━━━━━━━━━━━━━━━ 66s 2s/step - accuracy: 0.7755 - loss: 0.5581 - val_accuracy: 0.7911 - val_loss: 0.5170
Epoch 4/6
29/29 ━━━━━━━━━━━━━━━━━━━━ 65s 2s/step - accuracy: 0.7784 - loss: 0.5620 - val_accuracy: 0.7911 - val_loss: 0.5179
Epoch 5/6
29/29 ━━━━━━━━━━━━━━━━━━━━ 99s 2s/step - accuracy: 0.7810 - loss: 0.5524 - val_accuracy: 0.7911 - val_loss: 0.5128
Epoch 6/6
29/29 ━━━━━━━━━━━━━━━━━━━━ 82s 2s/step - accuracy: 0.7810 - loss: 0.5499 - val_accuracy: 0.7911 - val_loss: 0.5136
29/29 ━━━━━━━━━━━━━━━━━━━━ 28s 878ms/step
DEBUG: raw y_proba.shape = (450, 2)
DEBUG: sample y_proba (first 5 rows):
 [[0.82977796 0.17022195]
 [0.8457705  0.1542295 ]
 [0.8175086  0.18249133]
 [0.83475083 0.16524921]
 [0.83095515 0.16904491]]
Normalized probs.shape = (450, 2)
Normalized sample (first 5):
 [[0.82977796 0.17022195]
 [0.8457705  0.1542295 ]
 [0.8175086  0.18249133]
 [0.83475083 0.16524921]
 [0.83095515 0.16904491]]
Accuracy: 0.7933333333333333
Confusion matrix:
 [[357   0]
 [ 93   0]]
```

Typical observations:
- CNN shows strong ability to differentiate Normal vs Abnormal patterns.
- Transfer learning models may improve generalization but require RGB conversion overhead.
- Precision–Recall analysis highlights positive-class detection capability.

6. Conclusions and Future Scope

The project demonstrates that heart sound classification using spectrogram-based CNN models is effective and computationally efficient. The baseline CNN achieves competitive performance even with a comparatively small dataset.

Future Scope:
- Increase epoch count (20–50) for stronger convergence.

- Apply audio augmentation: noise addition, pitch shift, time-stretch.
- Train end-to-end raw waveform models (1D-CNNs, WaveNet, CRNN).
- Use larger pretrained models such as EfficientNet.
- Deploy as a mobile or web-based diagnostic application.

7. References

[1] PhysioNet / CinC Challenge Dataset
[2] Keras API Documentation
[3] Springer & IEEE papers on PCG classification
[4] MobileNetV2 and ResNet50 original research publications