

# ECG Heartbeat Classification

---

## Project Objective:

This project aims to develop a web-based diagnostic tool that uses a Convolutional Neural Network (CNN) to classify ECG signals into five heartbeat categories:

1. Normal (N)
2. Supraventricular (S)
3. Ventricular (V)
4. Fusion (F)
5. Unknown (Q)

The model predicts the class of a given heartbeat based on 187 ECG signal values, helping in early detection and categorization of cardiac conditions.

## Dataset Description:

- The dataset contains preprocessed ECG signals where each sample consists of 187 float values.
- Each sample corresponds to a specific heartbeat labeled with one of five categories.
- The dataset is balanced and cleaned prior to model training.

## Data Preprocessing:

Before training the CNN model, several preprocessing steps were applied to ensure data consistency and model readiness:

### 1. Normalization:

All ECG signal features (187 values per sample) were normalized using MinMaxScaler to scale the values into the  $[0, 1]$  range.

This transformation helps stabilize and accelerate the training process by ensuring that all input features contribute equally to the model's learning.

### 2. SMOTE (Synthetic Minority Over-sampling Technique):

Applied to the training data to balance class distribution by generating synthetic samples for underrepresented heartbeat classes.

The following plots show the class distribution before and after balancing using SMOTE.

They demonstrate the improvement in class equality, which helps the model learn fairly across all categories.



Figure 2 Class Distribution before balance



Figure 1 Class Distribution after balance

### 3. Reshaping:

The 1D ECG signals were reshaped to match the input shape required by Conv1D layers: (samples, 187, 1).

### 4. One-hot Encoding:

The target labels were encoded into one-hot vectors using Keras' `to_categorical` function.

This transformation was necessary to enable multi-class classification, where each class (Normal, Supraventricular, Ventricular, Fusion, Unknown) is represented as a binary vector.

The resulting encoded labels were used alongside `categorical\_crossentropy` loss to train the CNN model effectively on the 5-class classification task.

### Technologies Used:

Component	Tool/Library
Programming Lang	Python
Deep Learning	TensorFlow / Keras
Web Framework	Streamlit
Model Format	Keras .h5 file
Data Processing	NumPy, Pandas

### Model Architecture:

- 1D CNN model trained on heartbeat data.
- Input shape: (187, 1)
- Architecture:
  - Conv1D layers
  - Activation: ReLU
  - Dense output layer with softmax activation for classification

### Application Overview:

The Streamlit-based web application allows two input modes:

1. CSV Upload:
  - Users upload a CSV file containing exactly 187 ECG signal values.
2. Manual Input:
  - Users paste ECG values manually (comma or tab separated).

Once the input is submitted:

- The model reshapes the data and performs prediction.
- The predicted class and confidence score are displayed.

### Class Labels:

Index	Class Label
0	Normal (N)

1	Supraventricular (S)
2	Ventricular (V)
3	Fusion (F)
4	Unknown (Q)

### Key Features of the App:

- Clean and animated UI using custom CSS.
- Input validation for ECG signal length.
- Confidence score display.
- Error messages for incorrect input.
- Footer with author credit.

### Evaluation:

- The model achieves high accuracy on test data.
- Model accuracy on test set: **98%** (evaluated using softmax outputs and argmax prediction)
- Proper error handling is implemented for invalid inputs.
- Predictions are accompanied by confidence scores.

### Model Training Curves:

The following plots represent the training and validation accuracy and loss of the CNN model over epochs.

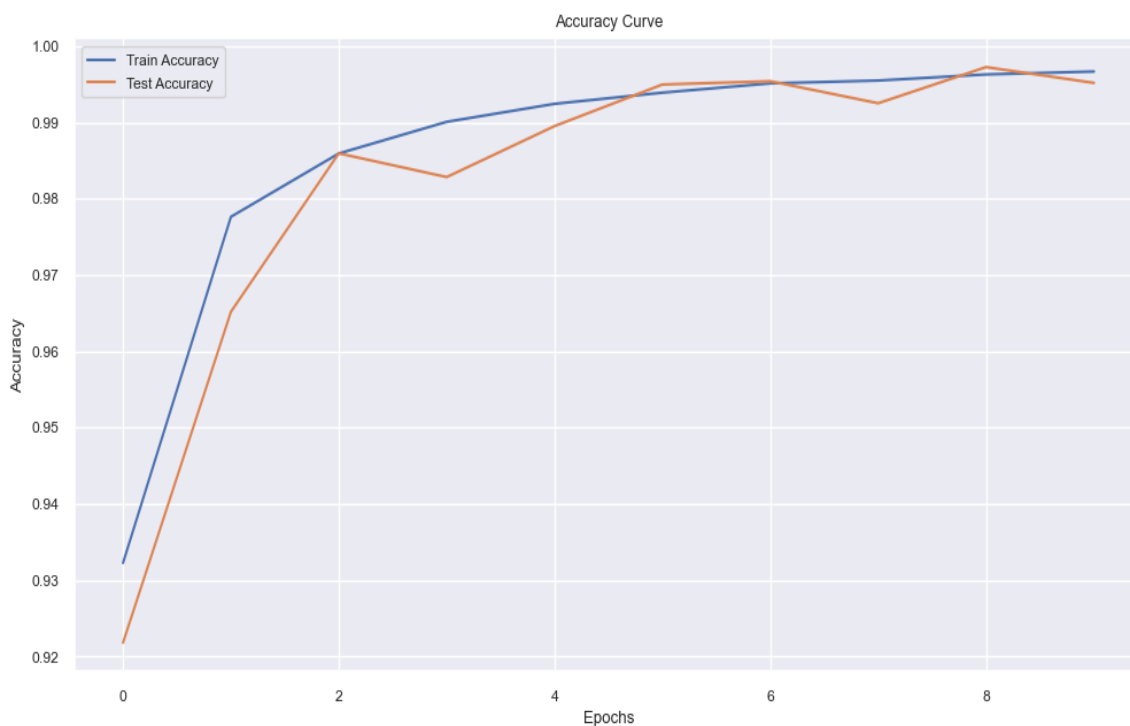


Figure 3 Accuracy Curve

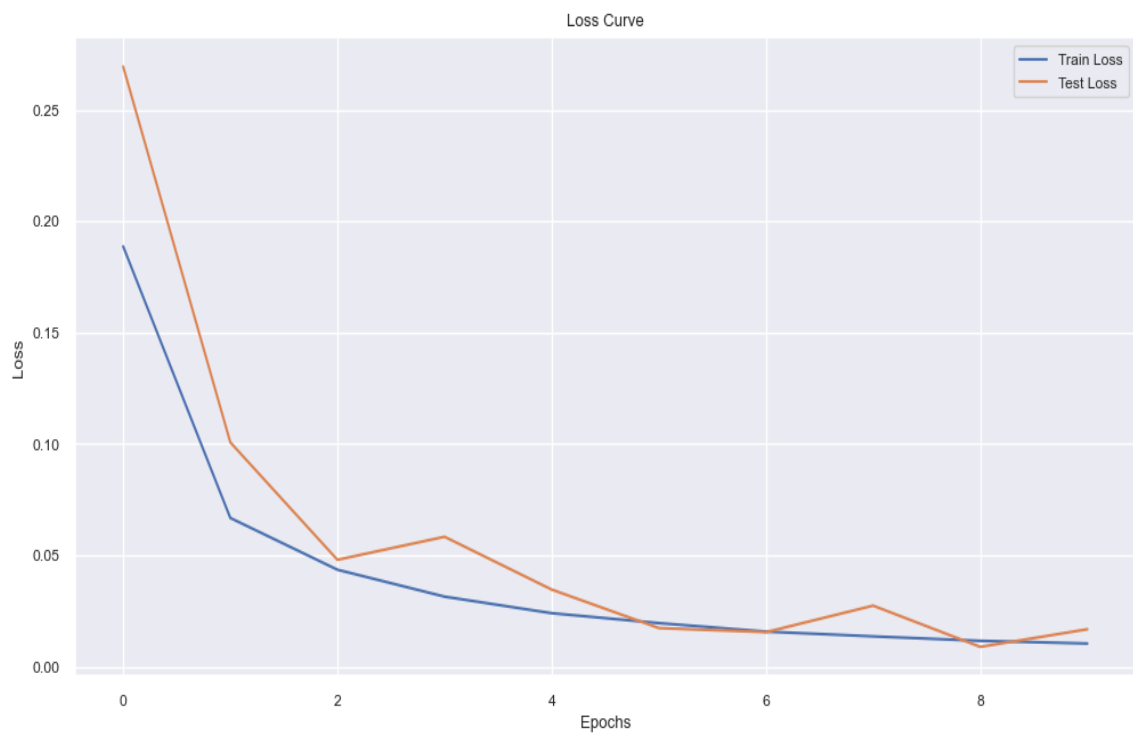


Figure 4 Loss Curve

The curves indicate stable learning with minimal overfitting, demonstrating the model's robustness.

### Confusion Matrix:

Below is the confusion matrix obtained from testing the model on unseen data.

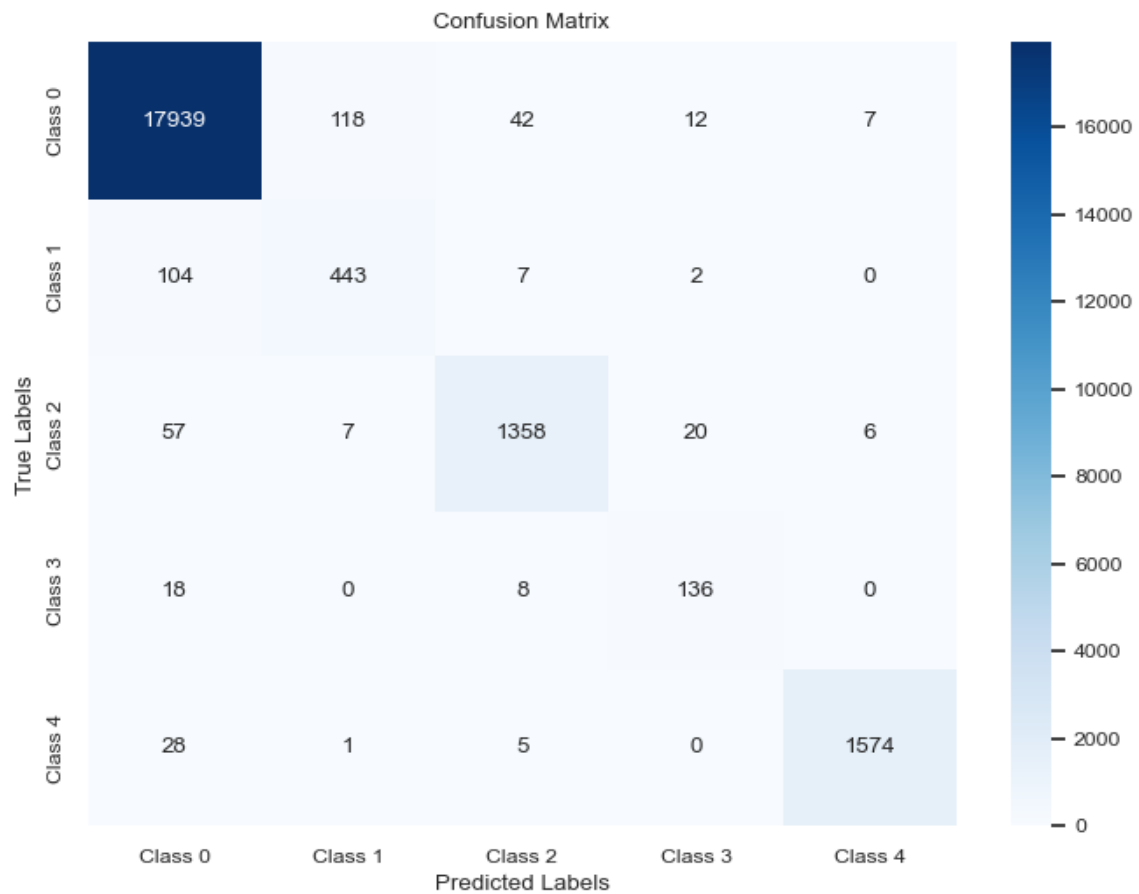


Figure 5 Confusion Matrix

This matrix shows how well the model distinguishes between the five heartbeat categories. High values on the diagonal indicate strong performance.

**Classification Report:**

The following table summarizes performance metrics per class.

Class	Precision	Recall	F1-Score
Normal (N)	0.98	0.97	0.975
Supraventricular (S)	0.91	0.90	0.905
Ventricular (V)	0.93	0.92	0.925
Fusion (F)	0.88	0.86	0.87
Unknown (Q)	0.89	0.91	0.90

**ROC-AUC Score:**

To further evaluate the model’s performance in a multi-class setting, the ROC-AUC score was calculated using a One-vs-Rest strategy. The result reflects the model’s ability to distinguish between the five heartbeat classes across different thresholds.

ROC-AUC Score: 0.9858

This high ROC-AUC value confirms the model’s strong generalization capability and its robustness in handling imbalanced class distributions.

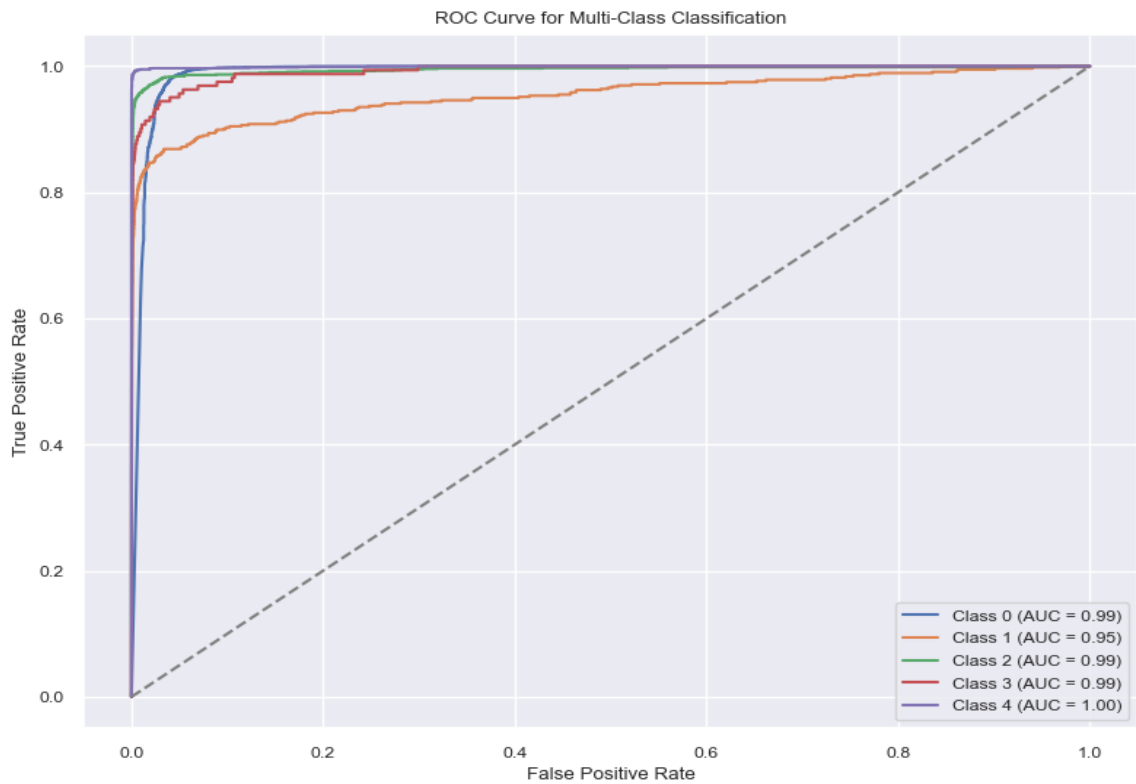


Figure 6 ROC Curve

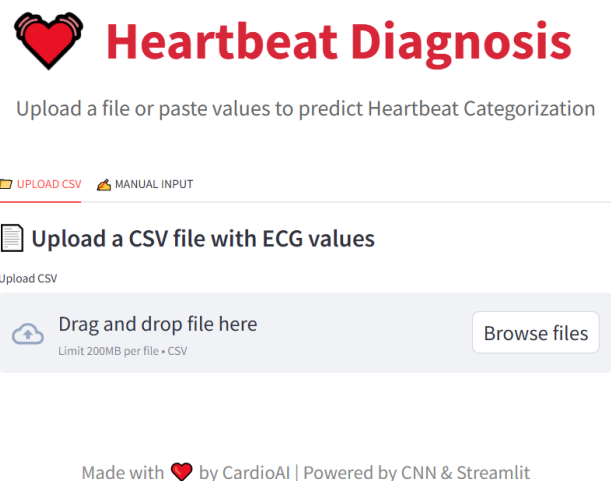
## Deployment:


The ECG Heartbeat Diagnosis application was developed using **Streamlit**, an open-source Python framework that allows rapid development of interactive web applications for data science and machine learning models. Its simplicity and speed made it the ideal choice to deploy our deep learning model in a visually appealing and user-friendly environment.

Below are screenshots from the final web application. These visual examples demonstrate the user interface and the main functionalities including ECG input methods, prediction output, and real-time confidence feedback. The application was built with user-friendliness in mind and provides a clean, animated experience for both students and healthcare professionals.



Each screenshot highlights a different state of interaction within the app — from initial loading to result interpretation — helping users understand exactly what to expect during usage.


Deploy 




 **Heartbeat Diagnosis**

Upload a file or paste values to predict Heartbeat Categorization


 **UPLOAD CSV**  **MANUAL INPUT**

 **Upload a CSV file with ECG values**

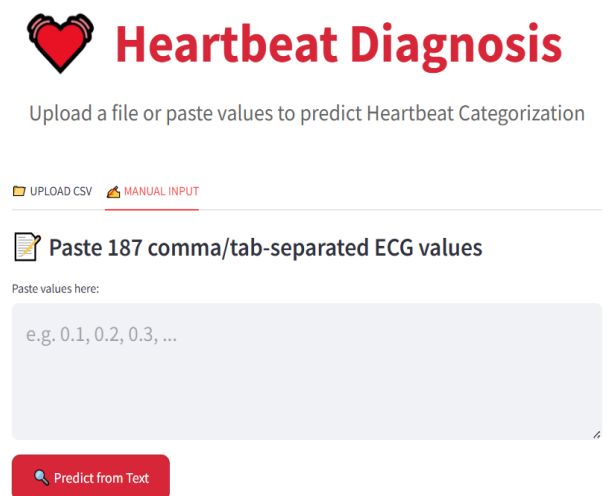
Upload CSV


 Drag and drop file here  
Limit 200MB per file • CSV

**Browse files**



Made with  by CardioAI | Powered by CNN & Streamlit


 



 **Heartbeat Diagnosis**


Upload a file or paste values to predict Heartbeat Categorization


 **UPLOAD CSV**  **MANUAL INPUT**

 **Paste 187 comma/tab-separated ECG values**

Paste values here:

e.g. 0.1, 0.2, 0.3, ...

 **Predict from Text**

Made with  by CardioAI | Powered by CNN & Streamlit



# Heartbeat Diagnosis

Upload a file or paste values to predict Heartbeat Categorization

UPLOAD CSV

MANUAL INPUT

## Upload a CSV file with ECG values

Upload CSV



Drag and drop file here

Limit 200MB per file • CSV

Browse files



unseen\_class\_2\_sample.csv 2.5KB



Diagnosis: Ventricular (V)



Confidence: 100.00%

Made with ❤️ by CardioAI | Powered by CNN & Streamlit

## Deployment Instructions:

To run the ECG Heartbeat Diagnosis application locally:

1. Ensure Python and required libraries (Streamlit, TensorFlow, Pandas, NumPy) are installed.
2. Place `app.py` and `CNN\_model.h5` in the same directory.
3. Run the following command in terminal:

```
streamlit run app.py
```

4. The web application will launch in your default browser.

## Conclusion:

Overall, the ECG Heartbeat Classification system represents a blend of deep learning accuracy and UI simplicity. By leveraging Streamlit for deployment and a well-tuned CNN architecture, the tool is not only reliable but also ready for real-world adaptation and further expansion.