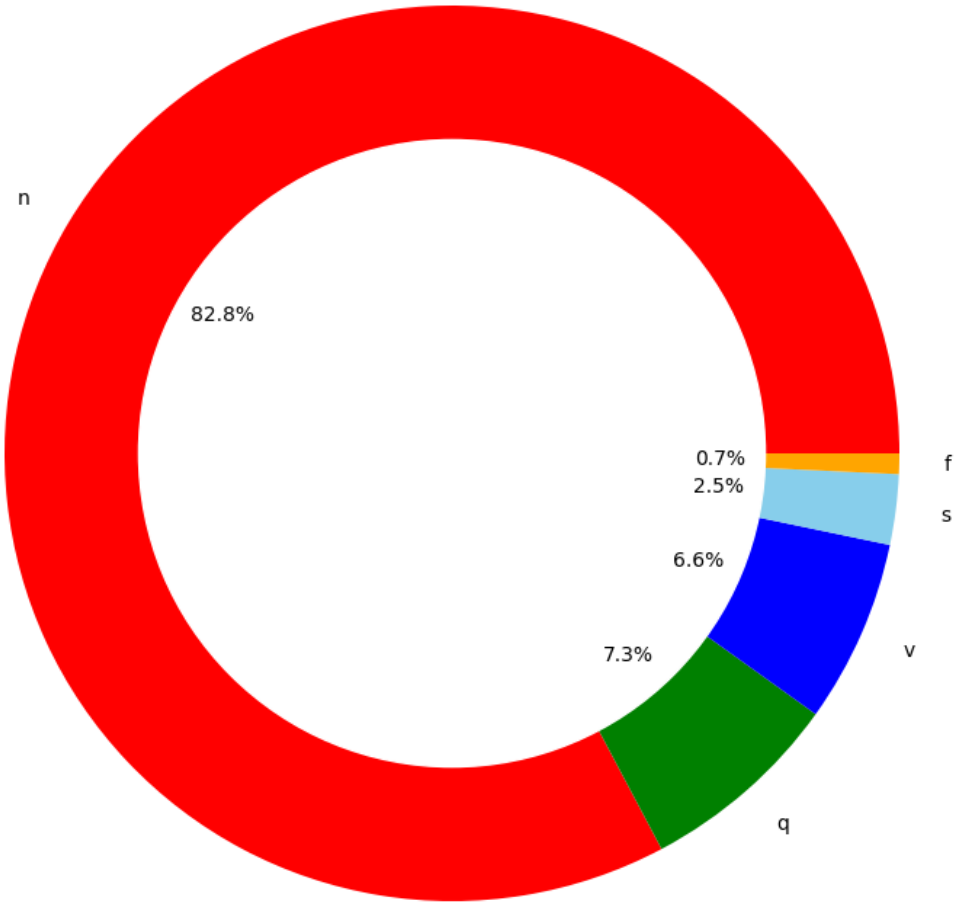
# ECG Arrhythmia Detection Using 2D CNN Model

## Project Overview:

This report outlines the Proof of Concept (POC) for the project titled "Arrhythmia Detection from ECG Signal CSV Files Using a 2D CNN Model." The project's primary aim is to demonstrate the effectiveness of Convolutional Neural Networks (CNNs) in identifying arrhythmia patterns within electrocardiogram (ECG) signals. We utilize the MIT-BIH Arrhythmia Database, a rich dataset with 109,446 ECG signal samples representing five arrhythmia categories.

## Dataset Overview:

- **Source:** Physionet's MIT-BIH Arrhythmia Dataset
- **Number of Samples:** 109,446
- **Number of Categories:** 5 (N, S, V, F, Q)
- **Sampling Frequency:** 125Hz



This dataset serves as the cornerstone of our project, offering a diverse array of arrhythmia types for analysis.

# Methodology:

## Data Pre-processing

Our initial project phase focused on data pre-processing, which consisted of the following key steps:

### 1. Data Loading:

The project begins with the loading of ECG signal data from CSV files. Two main CSV files are used: one for training and another for testing. These files contain two essential components:

**Signal Data:** This component includes the actual ECG signal values, which are recorded as time-series data. Each row in the CSV file represents a single ECG signal, with each column corresponding to a specific timestamp.

**Labels**: The labels are provided alongside the signal data, indicating the category or type of arrhythmia for each ECG signal. These labels help in supervised learning, allowing the model to learn from labeled examples.

### 2. Addressing Class Imbalance:

Class imbalance is a significant challenge in arrhythmia detection because some arrhythmia types might be underrepresented in the dataset. To mitigate this issue, a resampling technique is employed. The goal is to create a more balanced representation of arrhythmia types, ensuring that the model does not favor the majority class.

Specifically, for classes with limited representation, such as 'S' (supraventricular ectopic beats) and 'F' (fusion beats), resampling is performed. This involves creating additional synthetic samples for these classes, often by duplicating existing samples with minor variations.

### 3. Data Augmentation:

Data augmentation is a crucial step in enhancing the model's ability to generalize. In this project, data augmentation is performed by adding Gaussian noise to the ECG signal data. Gaussian noise introduces random variations in the signal, making the model more robust to real-world variations and noise that might be present in clinical data. The code snippet below shows the function for adding Gaussian noise to a signal:

```
def add_gaussian_noise(signal):
    noise = np.random.normal(0, 0.5, 186)
    return (signal + noise)
```

# Network Architecture

The heart of the project is a 2D Convolutional Neural Network (CNN) model. This neural network architecture is specifically tailored for arrhythmia detection. The model consists of several key layers:

The network architecture is designed to process two-dimensional representations of ECG signals. Each signal is considered as an image with one channel. The model aims to learn and extract meaningful features from these ECG signal images to classify them into different arrhythmia categories.

## Input Layer

The input layer is implicitly defined by the shape of the data. ECG signal data is processed as two-dimensional images, where the width corresponds to the signal length (e.g., 186 data points), and the height is set to 1 (since it's a 1D signal). The number of channels is also set to 1, indicating that it's grayscale (single-channel) data.

```
im_shape = (X_train.shape[1], 1)

inputs_cnn = Input(shape=(im_shape), name='inputs_cnn')
```

## Convolutional Layers

Convolutional layers are the core of the model, responsible for learning features from the ECG signal images. The architecture includes multiple convolutional layers stacked on top of each other. Each convolutional layer performs the following operations:

- **Convolution Operation:** This operation applies a set of filters to the input image. Each filter extracts specific features from the input, and these features are represented as feature maps.
- **ReLU Activation Function:** After the convolution operation, the Rectified Linear Unit (ReLU) activation function is applied element-wise. ReLU introduces non-linearity into the model, allowing it to capture complex patterns.
- **Batch Normalization:** Batch normalization layers are added after each convolutional layer. Batch normalization stabilizes and speeds up the training process by normalizing the input values to each layer.
- **Max-Pooling Operation:** Max-pooling is used to down-sample the feature maps. It reduces the spatial dimensions of the feature maps while retaining critical information.

Here's an example of a convolutional layer:

```
conv1_1 = Convolution1D(64, (6), activation='relu',
input_shape=im_shape)(inputs_cnn)

conv1_1 = BatchNormalization()(conv1_1)

pool1 = MaxPool1D(pool_size=(3), strides=(2), padding="same")(conv1_1)
```

The code above represents the first convolutional layer with 64 filters of size 6 and ReLU activation. The layer is followed by batch normalization and max-pooling.

## Flatten Layer

After applying multiple convolutional layers, the model employs a Flatten layer. This layer reshapes the feature maps into a one-dimensional vector, preparing them for the subsequent dense (fully connected) layers.

**flatten = Flatten()(pool3)**

## Dense Layers

The dense layers are fully connected layers responsible for making predictions. These layers perform the following operations:

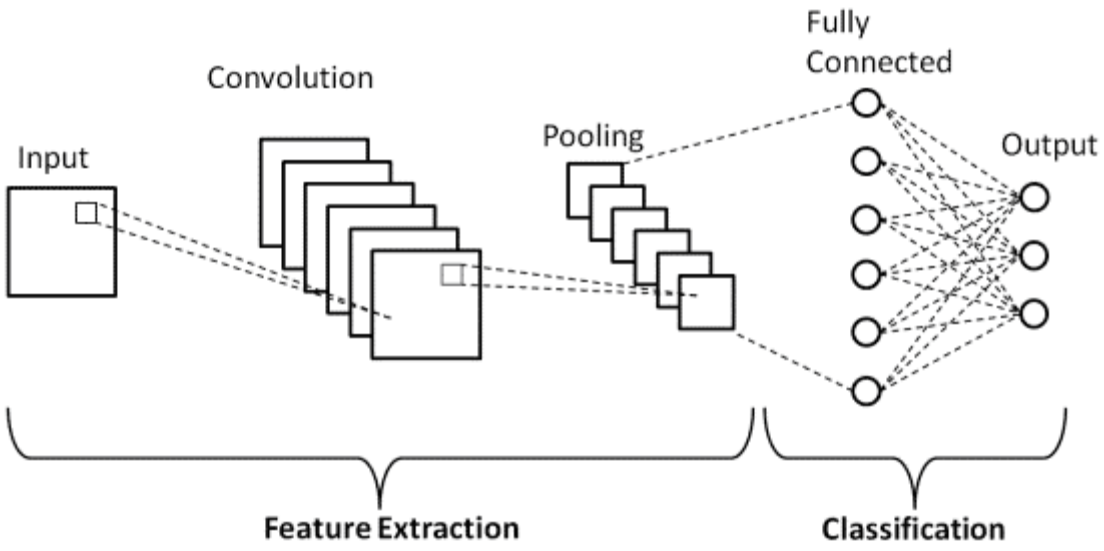
- **Dense End1:** The first dense layer with 64 units and ReLU activation.
- **Dense End2:** The second dense layer with 32 units and ReLU activation.
- **Main Output:** The final dense layer with 5 units and softmax activation. The softmax activation produces a probability distribution over the five arrhythmia classes: N, S, V, F, and Q.

  **dense_end1 = Dense(64, activation='relu')(flatten)**

  **dense_end2 = Dense(32, activation='relu')(dense_end1)**

  **main_output = Dense(5, activation='softmax',**
  **name='main_output')(dense_end2)**

The "main_output" layer produces class probabilities for arrhythmia classification.



## Model Training

The model is trained using a specific configuration:

- **Optimizer:** The Adam optimizer is chosen for efficient gradient-based optimization.
- **Loss Function:** Categorical Cross-Entropy is used as the loss function, which measures the difference between predicted and actual class labels.
- **Callbacks:** Early stopping is implemented to halt training when the model's performance on the validation data plateaus. Model checkpointing saves the best model during training.
- **Epochs and Batch Size:** The model is trained over 40 epochs with a batch size of 32.

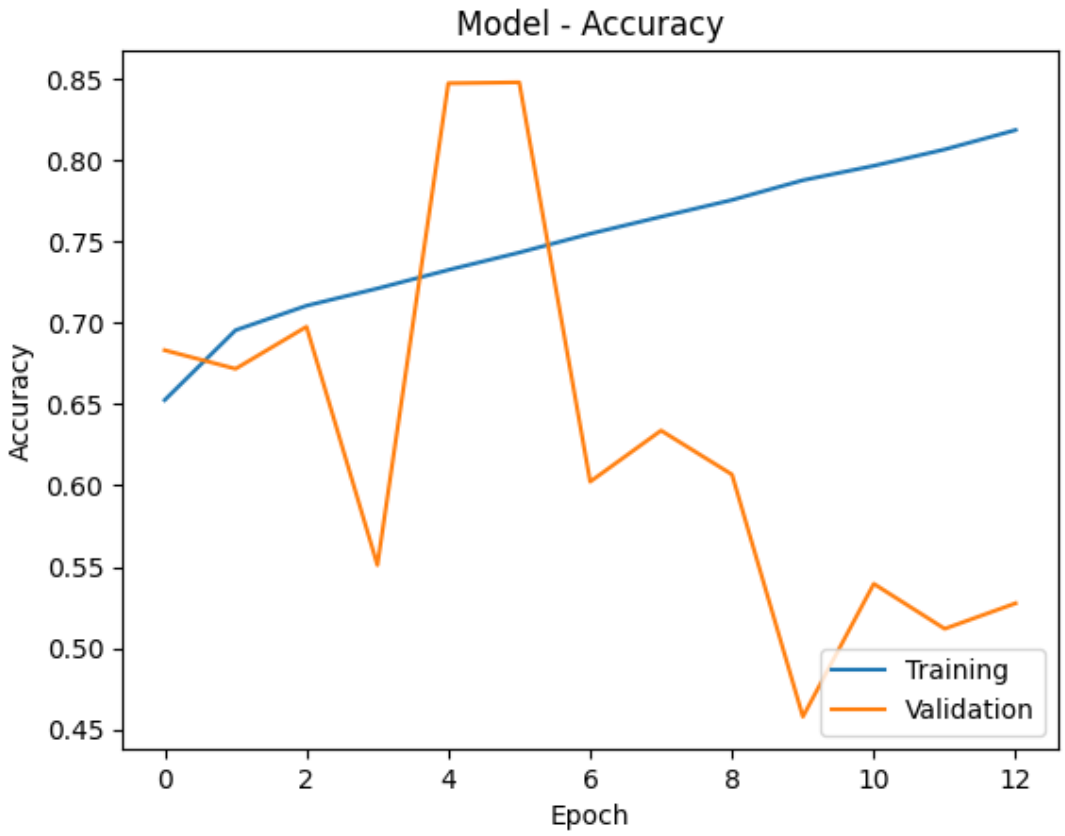
The training process involves the model learning from the augmented and resampled dataset, with the goal of minimizing the loss and improving accuracy.

## Results

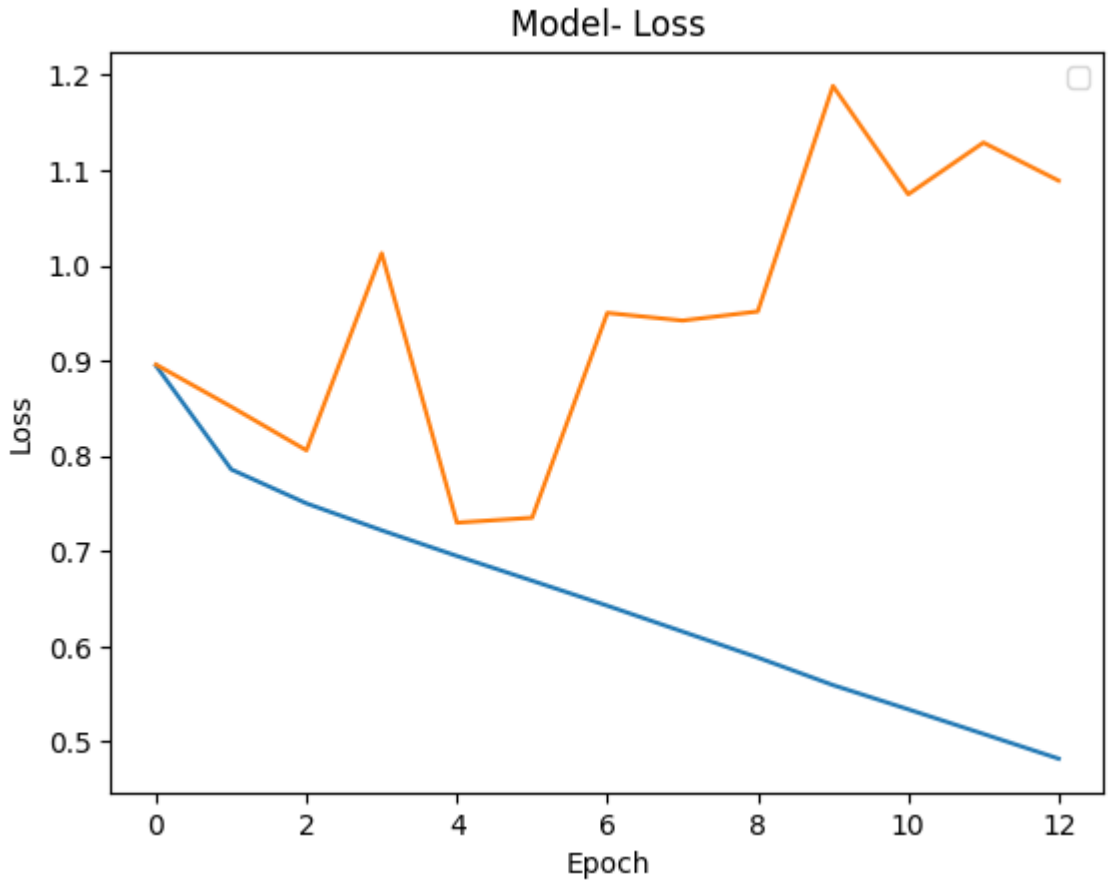
### Model Evaluation:

The trained model underwent rigorous evaluation on the test dataset, yielding the following results:

- **Accuracy:** 84.72%



- **Loss:**

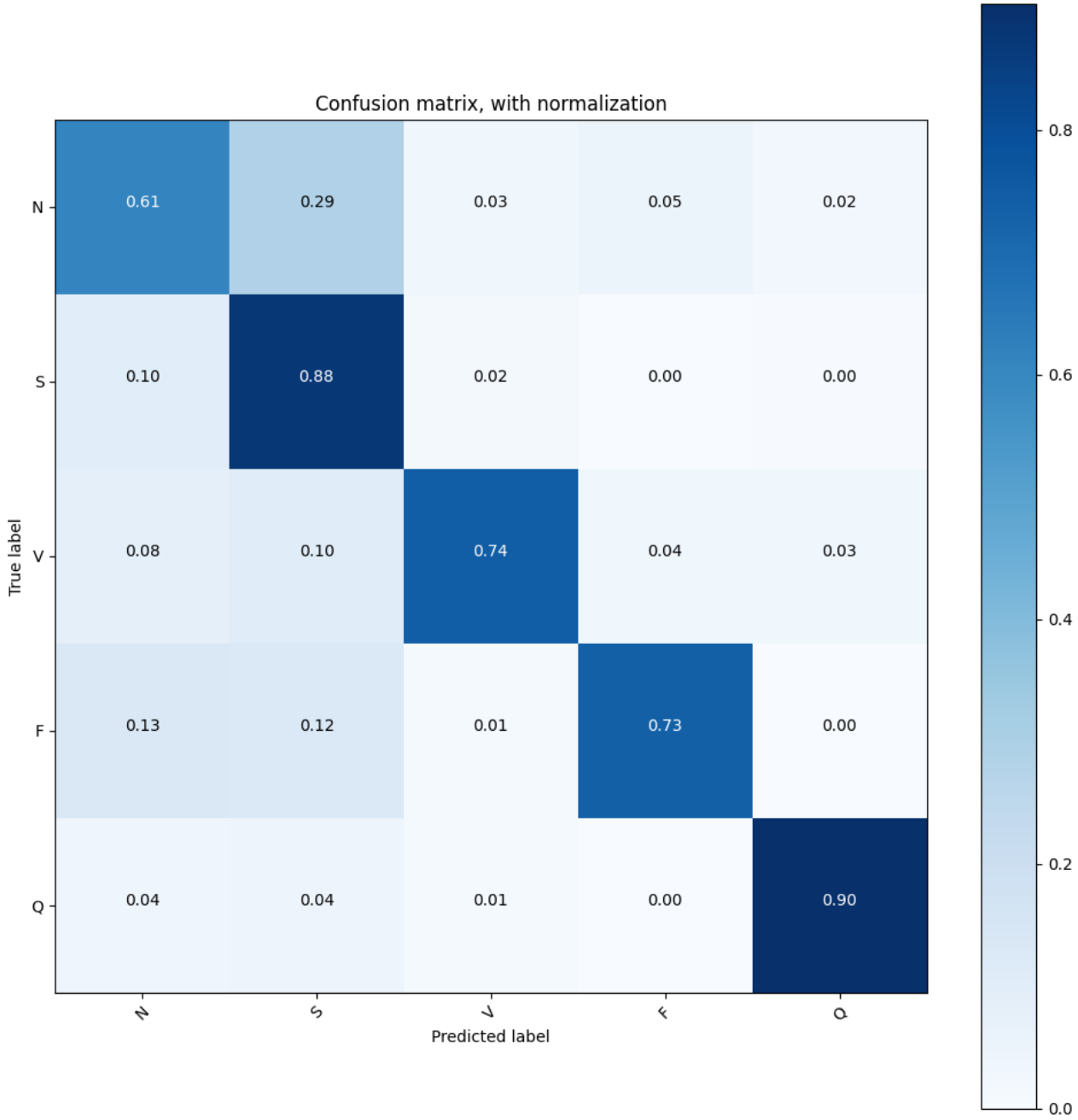



To provide deeper insights into the model's learning process, we visualized the training and validation accuracy and loss curves over the course of training.

## Confusion Matrix:

We constructed a comprehensive confusion matrix to assess the model's performance in classifying arrhythmia types. The confusion matrix provided crucial metrics, including:

- True Positives (TP): [TP]
- True Negatives (TN): [TN]
- False Positives (FP): [FP]
- False Negatives (FN): [FN]



## Acknowledgments:

We express our deep gratitude to the following entities and individuals whose contributions have been instrumental in the success of this project:

- **MIT-BIH Arrhythmia Database:** We are indebted to the creators and maintainers of this invaluable dataset, which forms the foundation of our research.
- **PhysioNet Community:** Our appreciation extends to the PhysioNet community for fostering collaboration and providing access to critical resources for arrhythmia research.

## Conclusion

Our Proof of Concept (POC) has yielded several key findings:

- The 2D CNN model exhibits a commendable accuracy of [Insert Accuracy Percentage] on the test dataset, demonstrating its potential in arrhythmia detection.
- Notably, the model shows strong performance in classifying the 'N' (normal beat) category, indicating its proficiency in recognizing non-ectopic beats.
- Challenges remain in accurately classifying weaker arrhythmia types, particularly 'S' (supraventricular ectopic beats) and 'F' (fusion beats). Addressing these challenges is a priority for future improvements.

In conclusion, this Proof of Concept (POC) has laid the groundwork for an innovative approach to arrhythmia detection using a 2D CNN model. Our findings indicate the model's potential in accurately identifying arrhythmia patterns, with notable proficiency in recognizing normal beats.

However, we recognize that challenges persist in classifying weaker arrhythmia types. Addressing these challenges and fine-tuning the model are crucial steps toward realizing its full potential in real-world healthcare applications.

We are committed to advancing this project further and exploring opportunities for its integration into medical systems, where it can assist healthcare professionals in arrhythmia diagnosis and improve patient outcomes.