Automated Classification of Cardiac Arrhythmia using Deep Convolutional Neural Networks (CNN) on ECG Images

Project Report

DECEMBER 15, 2025

Table of Contents

Introduction

Cardiac arrhythmia is an unusual heartbeat that can result in serious complications, such as stroke or sudden cardiac death if not detected early. The diagnosis of arrhythmias by recording the electrical activity of the heart has been conducted using an electrocardiogram. However, manual ECG interpretation is time-consuming, subjective, and susceptible to human errors, especially for brief or infrequent abnormal beats . This limitation becomes critical when there is a high volume of ECGs being processed or access to expert cardiologists is very limited.

In contrast, deep learning can be a reliable alternative for automated ECG analysis. CNNs, the most used architecture in medical image analysis, have shown outstanding performance in ECG signal classification. Recent works demonstrate that ECG segment representation as 2D images with the application of CNNs allow for high diagnostic accuracy .

Heartbeat can be classified into **six** types: Normal (N), Supraventricular ectopic (S), Ventricular ectopic (V), Fusion (F), Unclassifiable (Q), and Myocardial Infarction **(M)** using deep CNNs on image-based ECG data. Based on transfer learning, the approach leverages a pre-trained model, for instance, VGG16, together with a large public dataset derived from the MIT-BIH and PTB databases .

Problem Definition

Although manual analysis of cardiac arrhythmias in electrocardiogram signals is a tradition in medical practice, it has proven to have major drawbacks. Such analysis is very time-consuming and largely reliant on human expertise, which may easily overlook an arrhythmia intermittent or unusual in ECG signals recorded over a long span of time

To overcome these obstacles, it is evident that a need arises for a reliable and efficient automated system with the capability to classify arrhythmia with a high degree of accuracy without requiring continuous human interaction. To address this objective, this project proposes a 2D image processing technique in which ECG signals are translated into normalized images and processed via deep learning algorithms. The proposed system takes a single-channel ECG image, for instance 224x224 pixels, and maps it to a specific output class among **six** normalized heartbeat classes described in the AAMI EC57 guidelines: Normal heartbeat beats (N), Supraventricular ectopic

heartbeat beats (S), Ventricular ectopic heartbeat beats (V), Fusion heartbeat beats (F), Unclassifiable heartbeat beats (Q), and Myocardial Infarction beats (M). The proposed system proposes using an image-to-output class format to accomplish efficient arrhythmia analysis.

Objectives

1. To preprocess and prepare a large-scale (100,000+ images) ECG image dataset, establishing a robust data pipelines for a deep learning model.
2. To implement Transfer Learning by adapting a state-of-the-art pre-trained model (e.g., VGG16 or ResNet50) for the task of ECG classification.
3. To fine-tune the model by adding a new, custom classification head (Dense layers) and training it on the ECG image data.
4. To rigorously evaluate the final model's performance on an unseen test dataset using standard metrics, including Accuracy, Precision, Recall, F1-Score, and a detailed Confusion Matrix.

Dataset Description

The project utilizes a publicly available ECG image dataset on Kaggle, which is constructed using two famous datasets in the medical realm: MIT-BIH Arrhythmia Database and PTB Diagnostic ECG Database . The dataset comprises over 100,000 2D images in the grayscale format, with each image representing an individual heartbeat.

The images are grouped into five folders, which represent each of five heartbeat classes specified in AAMI EC57 guideline standards:

- N: Normal beat
- S: Supraventricular ectopic beat
- V: Ventricular ectopic beat
- F: Fusion beat
- Q: Unclassifiable or unknown beat
- **M:** Myocardial Infarction beat

Each image serves as the input feature (pixel values), and the folder name provides the ground-truth label (output class). The images are already preprocessed into visual representations of ECG signals, making them suitable for image-based deep learning models. This structure allows the use of standard computer vision tools and transfer learning techniques without requiring raw signal processing.

Methodology

Preprocessing

The dataset can be downloaded from Kaggle using the Kaggle API. The code begins with the initialization of the API token and Kaggle python environment setup. Then, it downloads a zip file named ecg-image-data.zip and extracts it in the working directory.

The images were organized into folders based on class (N, S, V, F, Q). A way to load these images without loading all of them into memory at a time is using ImageDataGenerator from TensorFlow. This utility loads images in a batch without requiring loading them from the image folders.

The preprocessing operations performed on the data were:

- Resizing: The size of all images was changed to 224x224 pixels to correspond with the pre-trained VGG16 model image size.
- Normalization: The intensity values were normalized by dividing by 255. The scaling factor is 1/255.
- Data Splitting: The training folder is split into a training set and a validation set using validation_split=0.20.
- Augmentation (Training Only): Small random modifications such as rotation (+/- 10°), width/height movement (+/- 10% shifts), and scaling (+/- 10% zoom) were applied to improve generalization. **Notably, horizontal flip was disabled to preserve the temporal direction of the ECG signals.**

No augmentation were performed on the validation/test datasets. The code validated that a total of 79,362 images were used for training, 19,837 for validation, and 24,799 testing images, all of which belonged to 6 classes.

```python
import os


os.environ['KAGGLE_API_TOKEN'] = 'KGAT_9961122a3f9b3f39bc3fb69666f3ce8'

!pip install -U -q kaggle


!kaggle datasets download -d erhmrai/ecg-image-data

!unzip -q ecg-image-data.zip
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 86.4/86.4 kB 4.5 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 256.4/256.4 kB 14.0 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.8/1.8 MB 68.3 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.6/13.6 MB 146.2 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 159.3/159.3 kB 17.2 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 189.0/189.0 kB 19.4 MB/s eta 0:00:00
Dataset URL: https://www.kaggle.com/datasets/erhmrai/ecg-image-data
License(s): CC-BY-NC-SA-4.0
Downloading ecg-image-data.zip to /content
 99% 846M/858M [00:08<00:00, 136MB/s]
100% 858M/858M [00:08<00:00, 107MB/s]
```

Figure 1 Dataset loading from Kaggle API.

```
Training directory found at: ECG_Image_data/train
Testing directory found at: ECG_Image_data/test

--- Loading Training Set (with Augmentation) ---
Found 79362 images belonging to 6 classes.

--- Loading Validation Set ---
Found 19837 images belonging to 6 classes.

--- Loading Test Set ---
Found 24799 images belonging to 6 classes.
```

Figure 2 Loaded data.

```python
train_datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.20,
    rotation_range=10,
    width_shift_range=0.1,
    height_shift_range=0.1,
    zoom_range=0.1,
    horizontal_flip=False,
    fill_mode='nearest'
)
```

Figure 3 Data augmentation steps.

Model Design

A transfer learning approach using VGG16 model architecture was used to construct this model. The base VGG16 model had weights transferred using images from the ImageNet database. The learnable parameters of all layers in the base model were set to non-trainable parameters by using trainsabled= False.

Based on VGG16, a custom classification head was added on top of it:

- A GlobalAveragePooling2D layer for reducing spatial dimensions.
- A Dense layer with a ReLU activation function and 256 units, with L2 regularization of 0.01.
- A Dropout layer (rate=0.5) to further promote regularization.
- A final Dense layer with 6 output units and a softmax activation function.

```
base_model = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224, 3))

base_model.trainable = False



model_vgg_regulated = models.Sequential([
    base_model,

    layers.GlobalAveragePooling2D(),

    layers.Dense(256, activation='relu', kernel_regularizer=regularizers.l2(0.01)),

    layers.Dropout(0.5),

    layers.Dense(6, activation='softmax')
])
```

Figure 4 Defining the model.

Training Details

The model has been compiled with the Adam optimizer, with a learning rate of 0.0001 and using categorical cross-entropy loss. The main metric used when training was accuracy.

Training was conducted for 5 epochs using the training generator, while the validation data came from the validation generator. The model progressively improved on both training accuracy and validation accuracy over the course of the epochs to an eventual 97.38% training accuracy and 98.61% validation accuracy at the end.

After training, the model was saved in .h5 format for later use.

Model: "sequential"

Layer (type)	Output Shape	Param #
vgg16 (Functional)	(None, 7, 7, 512)	14,714,688
global_average_pooling2d (GlobalAveragePooling2D)	(None, 512)	0
dense (Dense)	(None, 256)	131,328
dropout (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 6)	1,542

Total params: 14,847,558 (56.64 MB)

Trainable params: 132,870 (519.02 KB)

Non-trainable params: 14,714,688 (56.13 MB)

Figure 5 Compiled model.

```
Epoch 1/5
2481/2481 ━━━━━━━━━━━━━━━━━━━━ 1185s 474ms/step - accuracy: 0.7753 - loss: 2.0231 - val_accuracy: 0.9303 - val_loss: 0.4322
Epoch 2/5
2481/2481 ━━━━━━━━━━━━━━━━━━━━ 1161s 468ms/step - accuracy: 0.9321 - loss: 0.4058 - val_accuracy: 0.9546 - val_loss: 0.2910
Epoch 3/5
2481/2481 ━━━━━━━━━━━━━━━━━━━━ 1134s 457ms/step - accuracy: 0.9535 - loss: 0.2900 - val_accuracy: 0.9655 - val_loss: 0.2305
Epoch 4/5
2481/2481 ━━━━━━━━━━━━━━━━━━━━ 1135s 458ms/step - accuracy: 0.9656 - loss: 0.2339 - val_accuracy: 0.9730 - val_loss: 0.1896
Epoch 5/5
2481/2481 ━━━━━━━━━━━━━━━━━━━━ 1135s 457ms/step - accuracy: 0.9738 - loss: 0.1957 - val_accuracy: 0.9861 - val_loss: 0.1600
```

Figure 6 Model training.

Evaluation Metrics

After training, the saved model was reloaded and tested on the unseen test set. The test generator was reset to ensure correct prediction order.

Predictions were made using model.predict(), and the class with the highest probability was selected as the predicted label. True labels were extracted from the test generator.

Performance was evaluated using:

- Accuracy Score (overall correctness)
- Classification Report (precision, recall, F1-score per class)
- Confusion Matrix (visualized using seaborn heatmap)

Results

The final model achieved an overall test accuracy of 98.34%, showing strong performance in classifying ECG heartbeats from images.

✅ final accurcy VGG16 : 98.34%

	precision	recall	f1-score	support
F	0.00	0.00	0.00	161
M	1.00	1.00	1.00	2101
N	0.99	1.00	1.00	18926
Q	0.94	0.99	0.96	1608
S	0.77	0.80	0.79	556
V	0.97	0.92	0.95	1447
accuracy			0.98	24799
macro avg	0.78	0.78	0.78	24799
weighted avg	0.98	0.98	0.98	24799

Figure 7 Classification report.

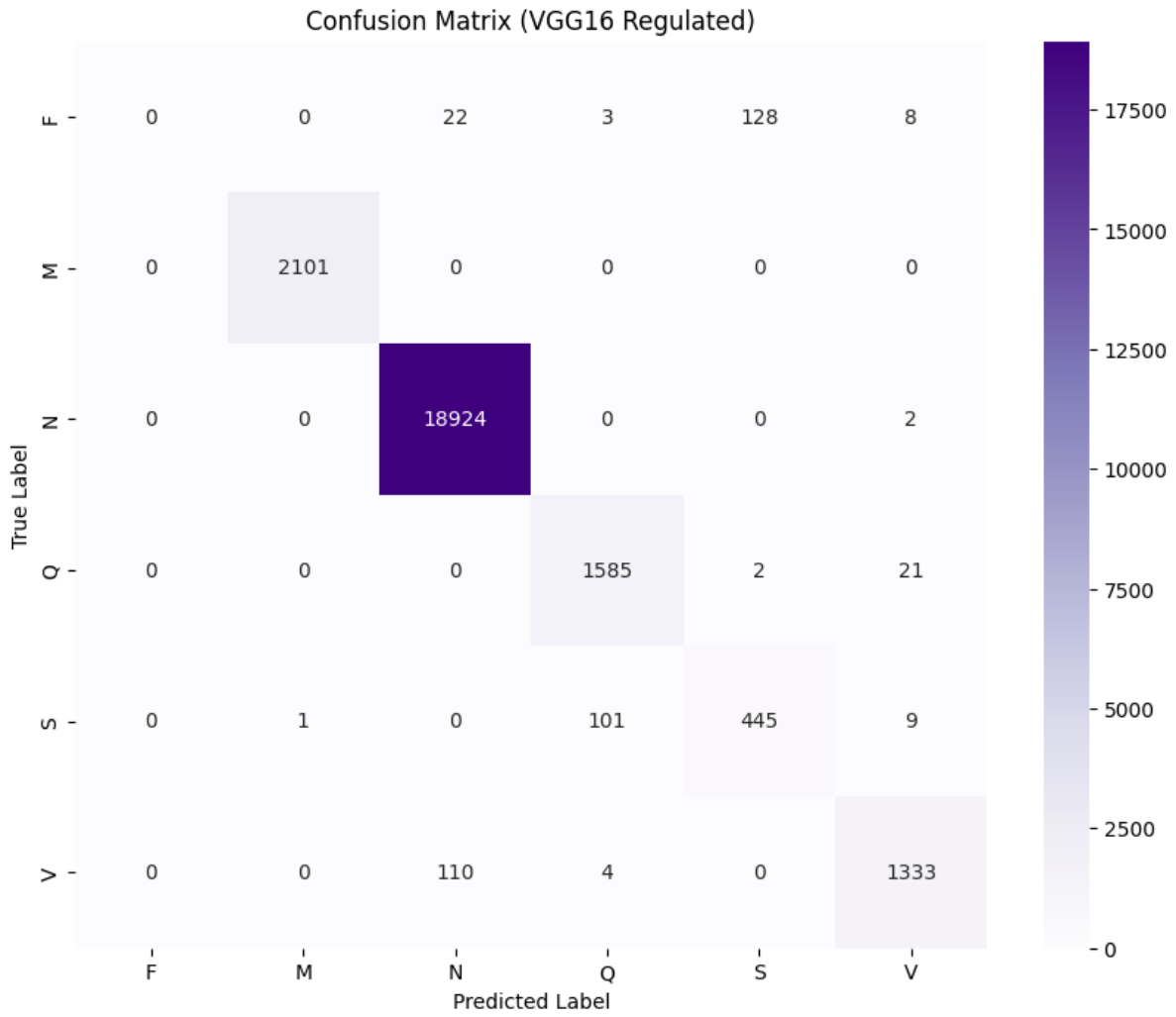

Figure 8 Confusion matrix.

Discussion

The output shows that the VGG16-CNN model is very efficient in classifying most of the ECG heartbeat signals with a weighted average F1 score of 0.98. The high accuracy rate shows effectiveness in making predictions on unseen samples, particularly in major categories such as N and M.

However, performance differs greatly across different classes. The F class (fusion beats) is greatly underrepresented with a mere 161 samples. This class could not be classified at all by the model with zero samples predicted into this class. A similarly underrepresented class with a mere 556 samples in the S class (supraventricular ectopic beats) demonstrates lower recall and precision

relative to other classes. These patterns strongly suggest that class imbalance is a major factor limiting performance on minority classes. Although transfer learning with VGG16 is effective when dealing with a classes having large number of instances, learning with a small number of samples in those classes can lead to poor performance.

To address this, future research can focus on:

- Class weighing (class_weight='balanced' in Keras) for weighing of misclassification of less frequently observed classes.
- Oversampling minority classes using methods such as duplication and synthetic generation.

Nevertheless, despite these shortcomings, it can be confirmed that using CNN on 2D ECG images can successfully classify arrhythmia with a high degree of accuracy. Such a technique holds immense potential in medical screening, remote analysis, and early warning systems in medical institutions.

Conclusion

A deep learning approach using a deep CNN with VGG16 architecture proved successful in classifying ECG heartbeat images into standard arrhythmic categories with a high weighted F1-score of 0.98. The deep model worked well on common classes such as Normal(N) and M class, confirming the effectiveness of transfer learning for large-scale ECG image analysis. Moreover, it proved inefficient in identifying Fusion(F) class instances, which were very uncommon in ECG images, along with lower accuracy in classifying Supraventricular(S) heartbeats, mainly because of class imbalance in a given dataset. Results show that deep learning techniques can become a game changer in automating ECG analysis with refinement in class weights, augmentation strategies, or bettering the existing dataset.

Ref. paper: Barry, K. A., Manzali, Y., & El Far, M. (2025). Deep Learning for ECG Classification Using Convolutional Neural Networks and VGG16. 2025 International Conference on Circuit, Systems and Communication (ICCSC), 1-5.

Dataset ref.: Kaggle Dataset: https://www.kaggle.com/datasets/erhmrai/ecg-image-data