

# Inference

May 1, 2025

## 1 Importing Necessary Libraries

```
[82]: import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import random
```

## 2 Loading the Model and the Data

```
[85]: model = tf.keras.models.load_model("../Models/Model.h5")
```

```
[87]: X, y = np.load("../Features.npy"), np.load("../Labels.npy")
```

### 2.1 Splitting the Data

```
[90]: _, X_test, _, y_test = train_test_split(X, y, test_size = 0.2, shuffle = True,
↳ random_state = 30)
```

```
[92]: X_test.shape, y_test.shape
```

```
[92]: ((21884, 250, 2), (21884,))
```

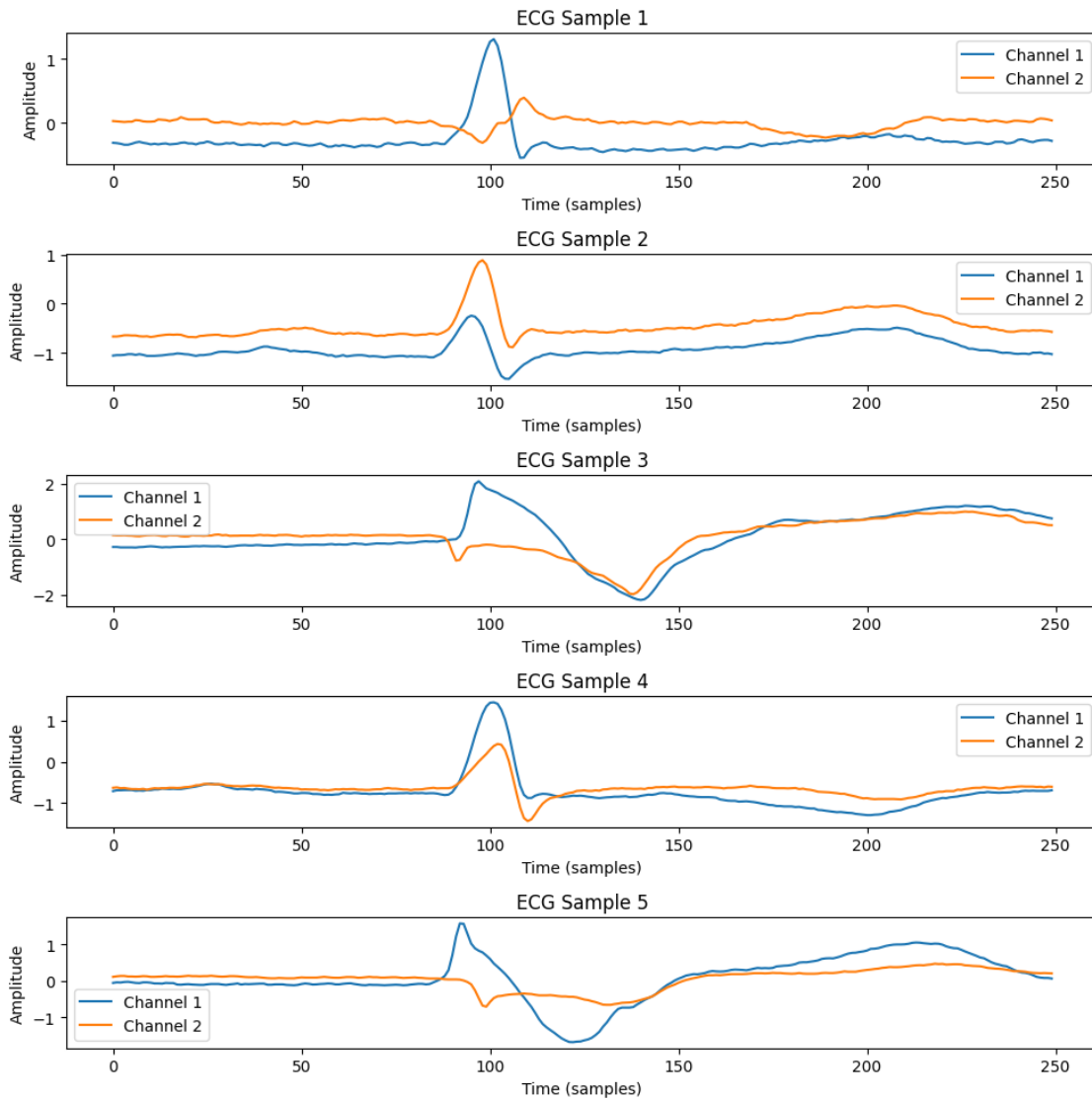
### 2.2 Plotting Random Samples

```
[57]: num_samples = 5
fig, axs = plt.subplots(num_samples, 1, figsize=(10, 10))

for i in range(num_samples):
    axs[i].plot(X_test[i, :, 0], label='Channel 1')
    axs[i].plot(X_test[i, :, 1], label='Channel 2')
    axs[i].set_title(f'ECG Sample {i + 1}')
    axs[i].set_xlabel('Time (samples)')
    axs[i].set_ylabel('Amplitude')
    axs[i].legend()

plt.tight_layout()
```

```
plt.show()
```



### 3 Inference

```
[94]: symbol, code = [], []

with open("../Remapped_Symbol_Classes.txt", "r", encoding = "utf-8") as file:
    for line in file:
        if ">" in line:
            s, c = line.strip().split(">")
            symbol.append(s.rstrip())
            code.append(c.lstrip())
```

```
[25]: symbol, code

[25]: ([('/', 'A', 'E', 'F', 'J', 'L', 'N', 'R', 'S', 'V', 'a', 'e', 'f', 'j'],
      ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12', '13'])

[27]: # Abnormal Beat Symbols
      abnormal = ['L', 'R', 'V', '/', 'A', 'f', 'F', 'j', 'a', 'E', 'J', 'e', 'S']

      # Normal Beat Symbols
      normal = ['N']

[29]: i = random.randint(0, X_test.shape[0] + 1)

[31]: temp = np.expand_dims(X_test[i, :, :], axis = 0)

[33]: idx = np.argmax(model.predict(temp))

1/1 [=====] - 1s 836ms/step

[34]: idx

[34]: 9

[39]: plt.plot(X_test[i, :, :])
      plt.title(f"Random ECG Sample | Prediction: {'Normal' if idx == 6 else 'Abnormal'} | Annotation: {symbol[idx]}")
      plt.grid()
      plt.xlabel("Time")
      plt.ylabel("Amplitude")
      plt.show()
```

