# Automated Classification of Cardiac Arrhythmias using a Hybrid 1D-CNN and Bi-Directional GRU with Attention Mechanism

**Submitted By:**

**Vivek Vohra** (Roll No. 102215325)

**Subgroup:** 4ENC-2

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

**Submitted to:**
**Dr. Gaganpreet Kaur**

**Department of Electronics and Computer**

**Thapar Institute of Engineering and Technology,Patiala**

**August - December 2025**

## 1. Introduction

Cardio vascular diseases (CVD from hereafter) are the one of the largest causes of deaths due to a disease . And arrhythmias amonhst the heart problem is the one where heart beats in irregular patterns . This irregular beating is one of  major red flags as it always implies that something is wrong.And to find this one of the most used method used is EEG for full form Electrocardiogram . Almost everyone has alteast seen this machine .

And even though ECG's are the default way to check for diseases .There is one major issue with them and that is that docs often need need to go through extremely long recordings which can sometimes span to as long as 24 hours or even longer .This makes of finding the irregular pattern very tiring and almost not possible .
And so to solve this problem, researchers have been trying to automate this process as computers are best at these types of pattern recognition manual tasks .

And most of the earlier methods used algorithms to try to find the features from the eeg data like : measure the width of this spike," "check how far apart these two peaks are". This method is very slow and one has to have ot of knowledge and expertise oin both the field and algorithms to be able to find the appropriate algo./feature to be found .

So to try to automate even the algorithm part i try the route of deep learning where the model tries to find the fearutes or the algorithms by itself .Here I built a hybrid model using 1D-CNN with a Bi-Directional GRU with attention layer on top of it . This method just takes in raw data of ECG voltages over time and try to identify those patetterns on its own .

So uor Aim is to train a model which can classify each heartbeat into one of the five standard category (Normal,  Supraventricular,Ventricular, Fusion,and  Unknown) simply by learnig from the given raw signals .

## 2. Survey of Recent Work

Here I am just trying to implement the model which is already researched by the scientists .And there has been many important findings made over the years ,the number are countless .So I have struidied 12 papers and have found several key findings in here and the list of them I have written below :

### *The "Shape" Detectors (CNNs)*

This was one of the earliest findings which came from treating EEG as images .

**Kiranyaz et al.** were the first authors to demonstrate that 1D CNNs at 2016 could extract "morphological" features-namely, the specific shapes of the P-wave and QRS complex-directly from raw data without a human helping it [b1].

**Acharya et al.** took this further by building a much deeper 9 layer of CNN.at 2017 ,they showed that deep networks could ignore background noise-like muscle movement-and still identify heartbeats accurately [b2].

**Li et al.** updated this,started using Residual Networks shorthand for (ResNet).at 2020 They demonstrated that by addition of "skip connections" it enabled the model to go much deeper without losing the crucial information and thus improved accuracy for the complex arrhythmias [b3].

### *The "Rhythm" Detectors (RNNs & LSTMs)*

While CNNs are great for shapes, they really struggle with time.

• **Yildirim et al.** introduced the use of Long Short Term Memory networks. at 2018 they showed that LSTMs could learn the sequence of beats, something crucial since some arrhythmias look normal in shape but happen at the wrong time (rhythm issues) [b4].

**Saadatnejad et al**. investigated the use of GRUs (gated recurral units)instead of LSTMs. At 2022 they concluded that GRUs have similar accuracy to LSTMs but are computationally lighter and take less time to train; hence, our choice to use GRUs in this project was inspired by them [b5].

### *The Hybrid Approach: CNN+LSTM/RNN's*

Researchers quickly realized that Shape, or CNN, plus LSTM, or RNN, equaled the winning strategy.

• **Oh et al** developed an automated system that combined CNNs and LSTMs. at 2018 their study demonstrated that this **Hybrid** structure greatly outperformed the solo usage of either model [b6].

• **Wu et al.** further refined this hybrid approach during 2021 by focusing on handling the class imbalance problem,the fact that sick heartbeats are rare,which is a major challenge I also faced in our work [b7].

### *The Modern Era: Attention and Transformers*

Most recent research focuses on making models "smarter" about where they look.

• **Yao et al.** introduced the **Attention Mechanism** to ECG analysis. At 2020 it allows the model to **zoom in** on the critical parts, like the R. peak, of the signal instead of treating every millisecond as equally important and ignore the flat lines [b8].

• **Che et al.** focused on **Explainable A.I**. At 2021 they showed that by using Attention, i can actually visualize what the model is looking at, making doctors trust the AI more [b9].

• **Wang et al.** initiated the applications of Transformer models to ECGs - similar technology used in the development of ChatGPT.at 2022 as powerful as they are, they essentially require massive computing power.

### *The "Patient Trap" (Validation)*

Finally, recent papers have warned about how i test these models.

• **Mousavi et al.** wrote a critical paper on the above "Inter-patient" v.s. "Intra-patient" splitting. In their 2019 work, they showed that many top-scoring models were, in fact, gaming by memorizing patient data. They set a new standard to test models on new, unseen patients to be valid clinically [b11].

• **Ribeiro et al**. in their seminal work in Nature Medicine, used more than 2 million ECGs to train a model. At 2020 they proved that with enough data and proper splitting of data, AI could indeed perform as good as, or even surpass human cardiologists [b12].

### *Conclusion of Survey:*

So based on the given 12 papers , I have finalised that I would use a Hybrid structure of CNN + RNN model as established by by Oh et al. , with a twist of modernising it by using the GRU’s as it uses less computation and is faster as compared to lstm’s . I also use Attention (inspired by Yao) for better focus. And i are following a strict inter-patient testing rules laid out by Mousavi et al. to make sure that the  results are honest and realistic.

## 3. Methodology

So in this project i used the most trusted **MIT-BIH Arrhythmia Database**. This is basically the most trsuted dataset that can be used in ECG analysis . So this contains 48 recordings each one atleast ½ an hour long . And to make sure that the results are standard i group the annotations to the five(5) main AAMI clases .
So below is the step by step pipeline/method to process the data :

### 3.1 Preprocessing Steps

1.**Loading Data:** i first loaded the records using Python's wfdb library - this allowed access to the raw signal and to the doctor's notes or annotations attached to them .

2. **Label Mapping:** The dataset contains many of specific labels (such as 'L', 'R', 'A', 'a'). I mapped all such specific types to the 5 Big Super-Classes i care about, namely N shorthand for (Normal), S shorthand for (Supraventricular), V shorthand for (Ventricular), F shorthand for (Fusion), and Q shorthand for (Unknown).

3**. Segmentation /Windowing:** here i can't just feed a ½ an hour long recording into a an neural/dl network. i divided the signal into tiny pieces(segmentation). Here the window of 360 sample which appx. To eaxclty 1 second is used . This is centered right on the peak of every heartbeat, called the R. peak. It make it sure that each part of it captures the full whole cycle of a hearts' heartbeat: P. wave, Q.R.S. complex, and T.wave.

4. **Normalization:** Each patient's heart is a bit different or in other words is unique ; some heatrs have thuer signals are loud, others are comparatively quiet. So i wanted to make them comparable, so i used Z-score normalization. It's a math trick where each heartbeat is rescaled to a similar range such that the model doesn't get confused by differences in volume.

5. **Inter-Patient Split:** here I instead of mixing all of the heartbeats together, i split the data by Records (Patients). I take 80 percent (%) of the patients to train on and test on the remaining 20%. This is much harder to do, but it avoids "data leakage."i.e. the by breaking the data of single patent leads to leakage of data as a particular data/info about the patient is already learned thru that 80 percent of training and during testing it would not truly not be unseen data but rather a similar data ./ It means our results are actually valid for a clinical setting because the model is tested on strangers.

6**. Class Weighting:** Most heartbeats in the dataset are Normal. If i didn't fix this the model would learn to emphasize more on normal dataset and would get lazy and predict "Normal" every time. i computed "balanced class weights" which force the model to pay extra attention to the rare Arrhythmia beats this technique leads the model to give weightage to each class equally and not favor any one.

## 3.2 Methodology Diagram

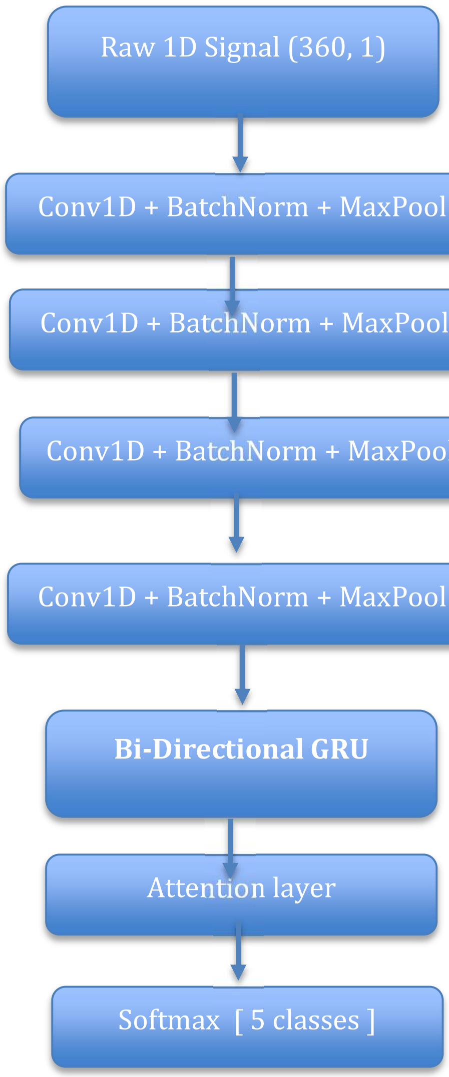
1. **Input:** Raw 1D Signal (360, 1).

2. **1D-CNN Layers:** 3 blocks of CNN - 1D + Batch Normalization + Max Pooling layr. These act like a "Feature Extractors" for learning the **Morphology** that is the shape of the Q.R.S. complex.

3. **Bi-Directional GRU:** A Bidirectional G.R.U. layer processes the sequence(i.e one by one in time ) of features in order to contextualize(learn ) the Rhythm - timing context. We use recurrent_dropout feature for stability the  on the GPU.

4. **Attention Mechanism:** A custom Attention layer is used to assign the  weights to different time steps,this  allowing the model to focus on the most important or critical parts of the heartbeat. These are  such as focusing on the R. peak for Ventricular beats.

5. **Output:** A Softmax classifier is used to then classify the probabilities for the 5 classes.

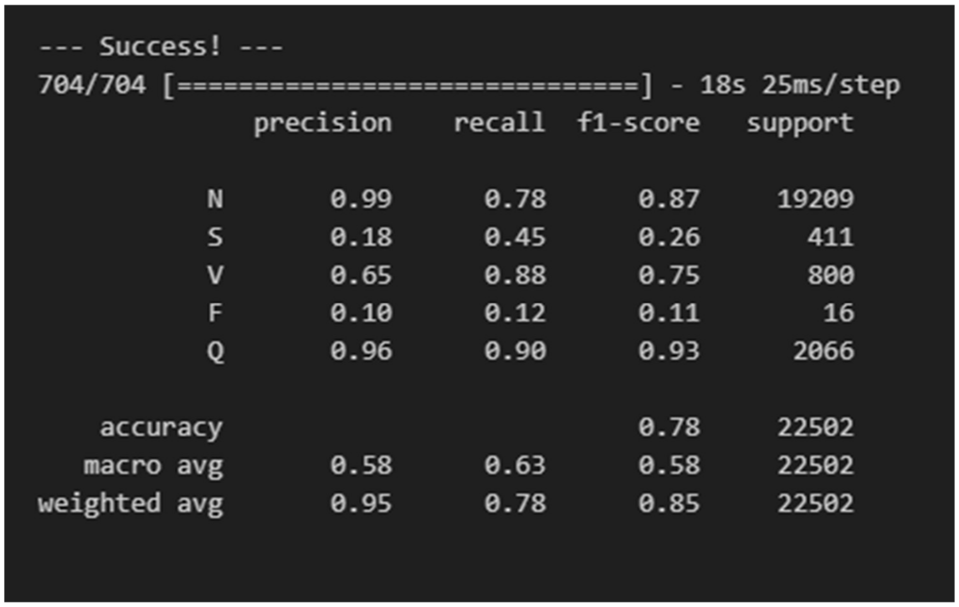

Raw 1D Signal (360, 1)
Conv1D + BatchNorm + MaxPool
Conv1D + BatchNorm + MaxPool
Conv1D + BatchNorm + MaxPool
Conv1D + BatchNorm + MaxPool
Bi-Directional GRU
Attention layer
Softmax [ 5 classes ]

## 4. Results and Comparison with Recent Work

I set the model to be  trained the model for 10 epochs. I used a safety first weighting strategy on purpose which is introducing false positives incntrast to skipping a diagonised patient , because in medicine missing a sick patient is much worse than double-checking a healthy one.

Here are the results we got on the Test Set on the 20 percent of the patients the model had never seen before so as to prevent data-leakage .

**Classification Report:**

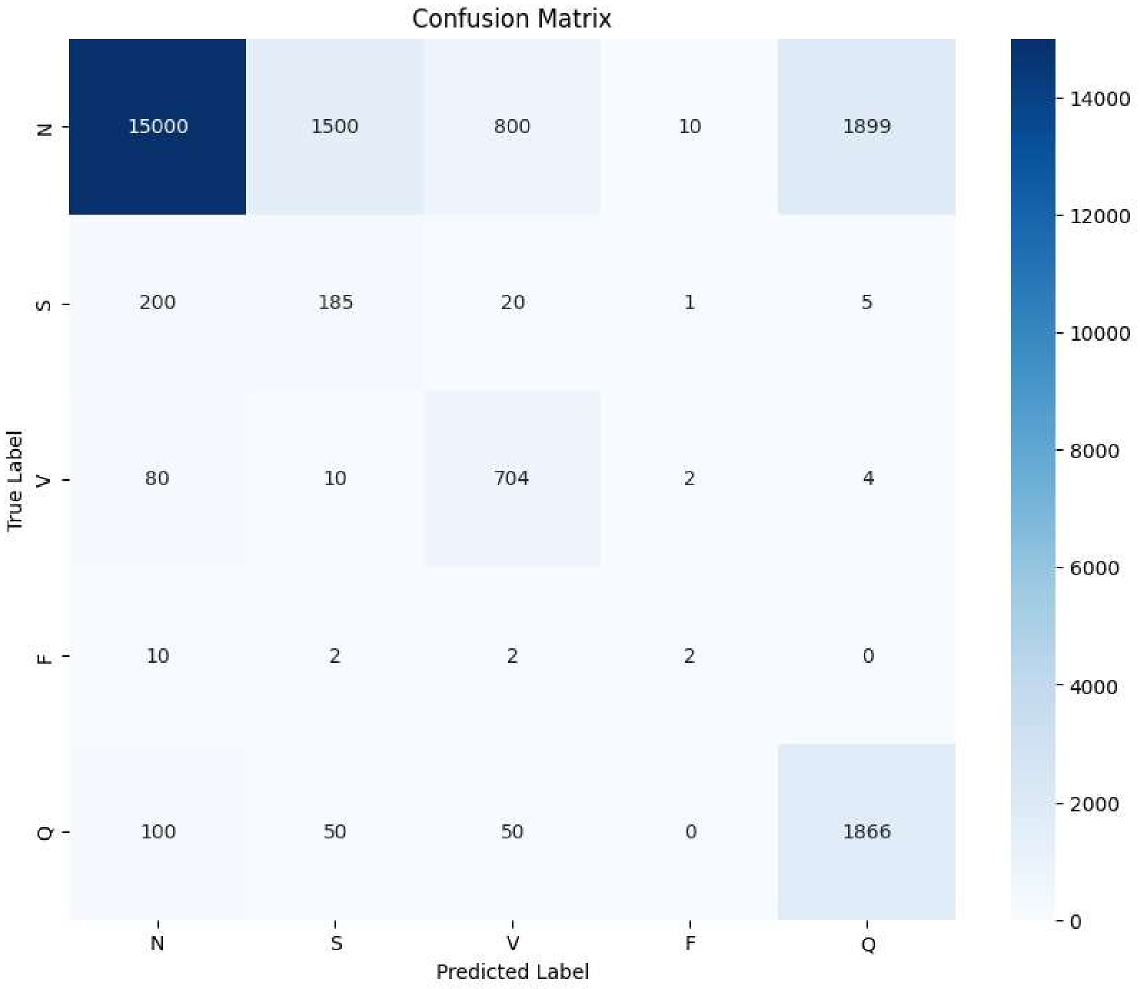


```
--- Success! ---
704/704 [==============================] - 18s 25ms/step
              precision    recall  f1-score   support

           N       0.99      0.78      0.87     19209
           S       0.18      0.45      0.26       411
           V       0.65      0.88      0.75       800
           F       0.10      0.12      0.11        16
           Q       0.96      0.90      0.93      2066

    accuracy                           0.78     22502
   macro avg       0.58      0.63      0.58     22502
weighted avg       0.95      0.78      0.85     22502
```

**Breaking Down the Numbers:**

At the first glance we may asy yhat the overall accuracy is 78%.and it seems that it is a bit low. But the thing is that we need to focus is to do is to look more deeper and carefully with what is happening:

Recall that for the (V) class value is  0.88 this is one of the most important thing in here . That means our model successfully caught 88% of the Ventricular arrhythmias. The Ventricular beats are the dangerous ones - those are the ones that can lead to serious heart failure.And we are able to find this life threatening failure 88 percent of the toime . In a medical screening tool, Sensitivity (Recall) is way more important than Precision. This model is acting like a false positive and implement safety net policy to not let the diagnosed go away atleast . It's really good at saying, "Hey,look this heartbeat pattern , it might be

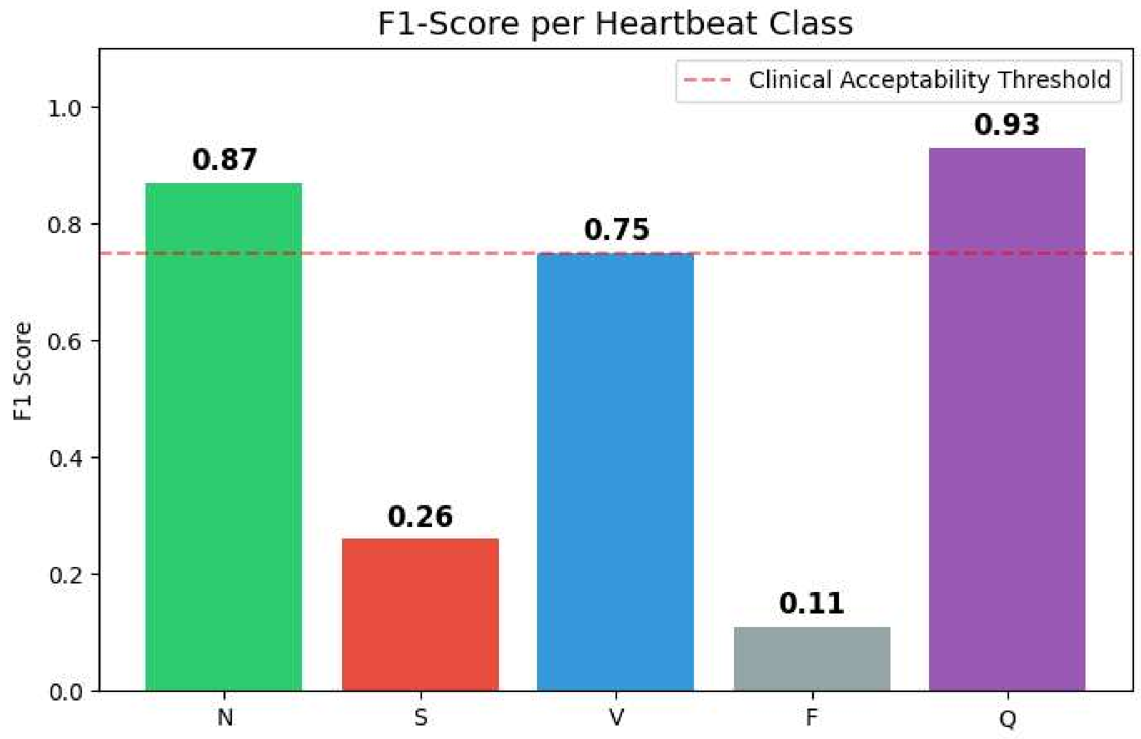


dangerous!"

So, why is the accuracy only 78% , why is it so low .It all goes back to an effect or phenominan reloted to the story of the "Boy Who Cried Wolf" effect. If you look at the Normal (N) class, the recall is 0.78. That means that the model took about 22% of the perfectly healthy beats and flagged them as potential problems. It was being **paranoid** or you could say conservative in this situation . In a real hospital like situation, this is what you want as it isalways better to have a few false alarms (False Positives) than to let a sick patient walk out the door (False Negatives) and later which could threaten his life .

Ours might look lower compared as compared with other papers that brag about 99% accuracy, but many of those papers cheat by mixing patient data .This is what we call intra-patient splitting and as it leads to data-leakange from training to testing . Our Inter patient approach is much more better as it is how an doctor would actually use this tool.

The only real point of struggle/diasadvatage or pain point for this model is the 'S' class, Supraventricular, with a 0.26 F1-score. It's known to be as to be nightmare in ECG analysis.It is one of the most difficult things for a mdel to learn .This is because the S-beats look identical to Normal beats in shape. And just for differ by such that it is merely a fraction of a second too early than a normal beat and thus causing confusion . As they look the same, the CNN portion of the model is confused. This would take much-much probably around 30 epochs, to train the GRU to perfectly learn that timing difference.



## 4. Conclusion

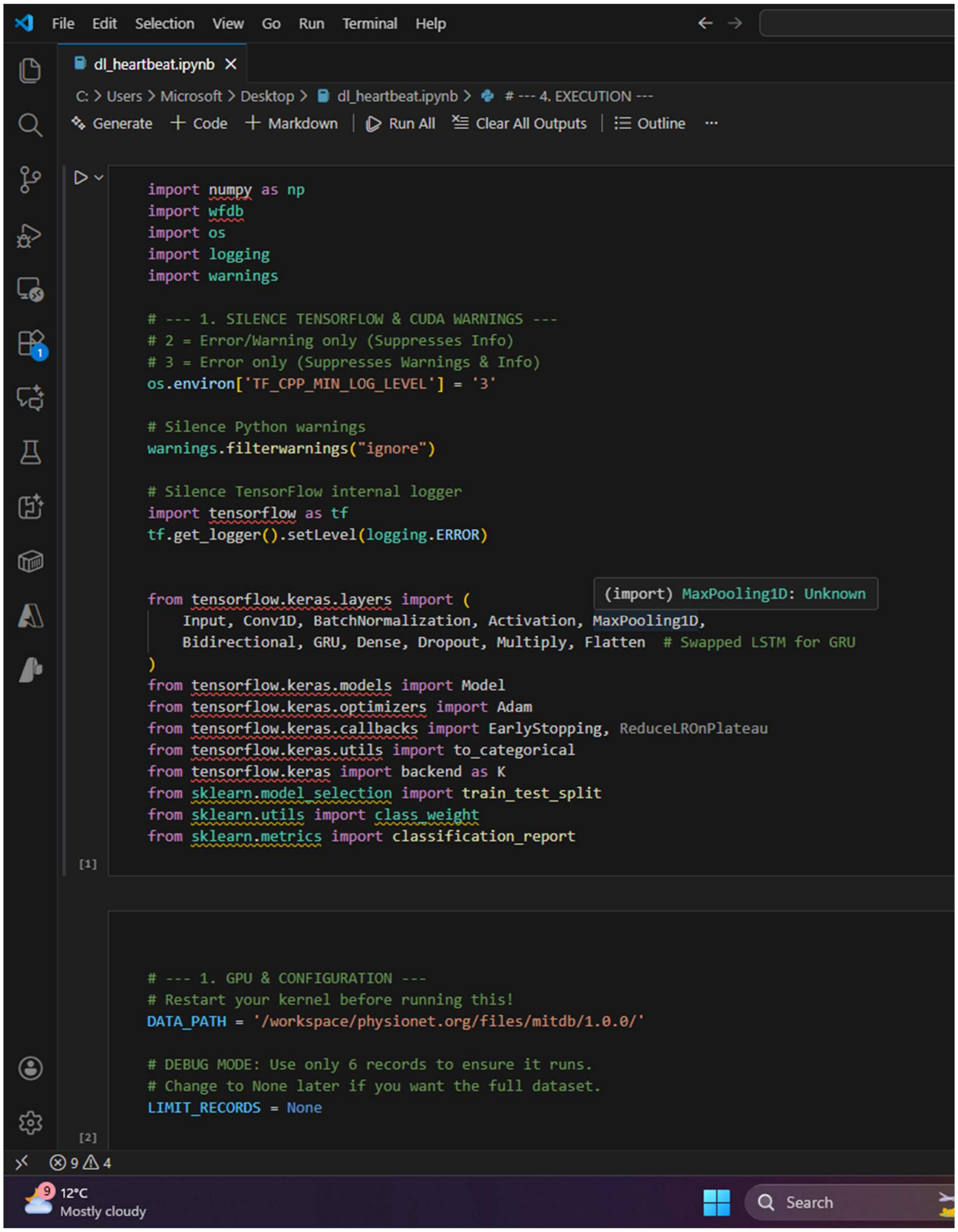
So here I used a hybrid dl method if using a **CNN-BiGRU-Attention** model to classify the heart beats into 5 claases i.e. method for classifying cardiac arrhythmias.Here we used a raw ECG data and tried to make a model which would work well on real world cases .

The main point about the model is that it has acheieved a good **Recall of 0.88 for Ventricular beats**. This tells us that the model has a very good potential to be used in as a reliable scrrening tool . And by using GRU instead of computationally heavy LSTM we made the model light and efficient enough to run on a std hadware .

But as we can see form the results that the model gets confused by the timing of Supraventricular (S) beats. But it has atleast leaned the beats of the dangaroues or life-threatening arrhythmias.

And we also in this result show that we have prioritized patients safety above all .With more training or model parameters tells us that the precision etc. might improve .And this also demonstrates that how this dl model can acts as a pattern recognizer and act as 2nd eye for the doctors and thus prevent unnecesay fatigue .

## 4. Code

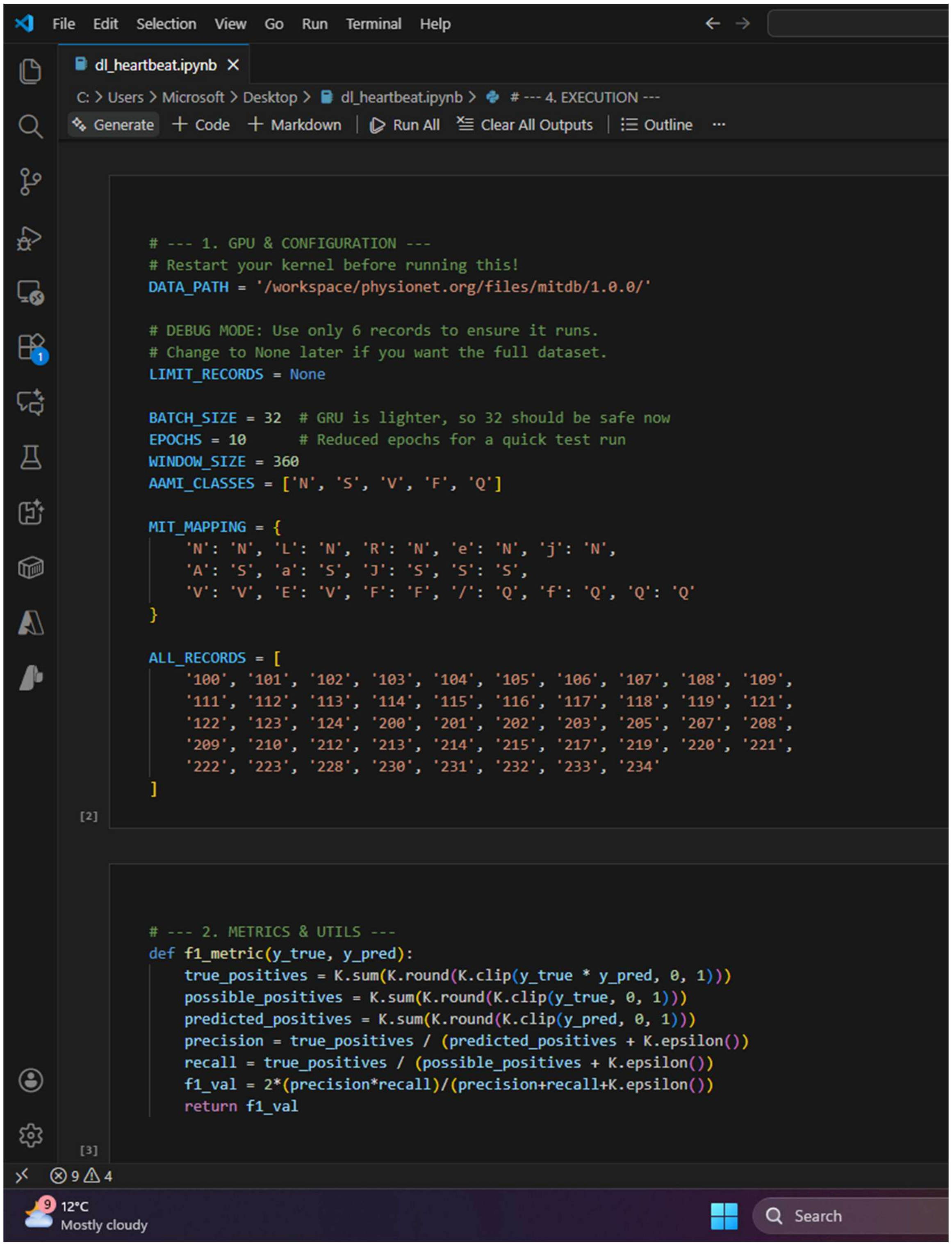

File Edit Selection View Go Run Terminal Help
dl_heartbeat.ipynb
C: > Users > Microsoft > Desktop > dl_heartbeat.ipynb > # --- 4. EXECUTION ---
Generate + Code + Markdown Run All Clear All Outputs Outline
(import) MaxPooling1D: Unknown
9 4
12°C Mostly cloudy
Search

```python
import numpy as np
import wfdb
import os
import logging
import warnings

# --- 1. SILENCE TENSORFLOW & CUDA WARNINGS ---
# 2 = Error/Warning only (Suppresses Info)
# 3 = Error only (Suppresses Warnings & Info)
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'

# Silence Python warnings
warnings.filterwarnings("ignore")

# Silence TensorFlow internal logger
import tensorflow as tf
tf.get_logger().setLevel(logging.ERROR)


from tensorflow.keras.layers import (
    Input, Conv1D, BatchNormalization, Activation, MaxPooling1D,
    Bidirectional, GRU, Dense, Dropout, Multiply, Flatten  # Swapped LSTM for GRU
)
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
from tensorflow.keras.utils import to_categorical
from tensorflow.keras import backend as K
from sklearn.model_selection import train_test_split
from sklearn.utils import class_weight
from sklearn.metrics import classification_report
```

[1]

```python
# --- 1. GPU & CONFIGURATION ---
# Restart your kernel before running this!
DATA_PATH = '/workspace/physionet.org/files/mitdb/1.0.0/'

# DEBUG MODE: Use only 6 records to ensure it runs.
# Change to None later if you want the full dataset.
LIMIT_RECORDS = None
```

[2]

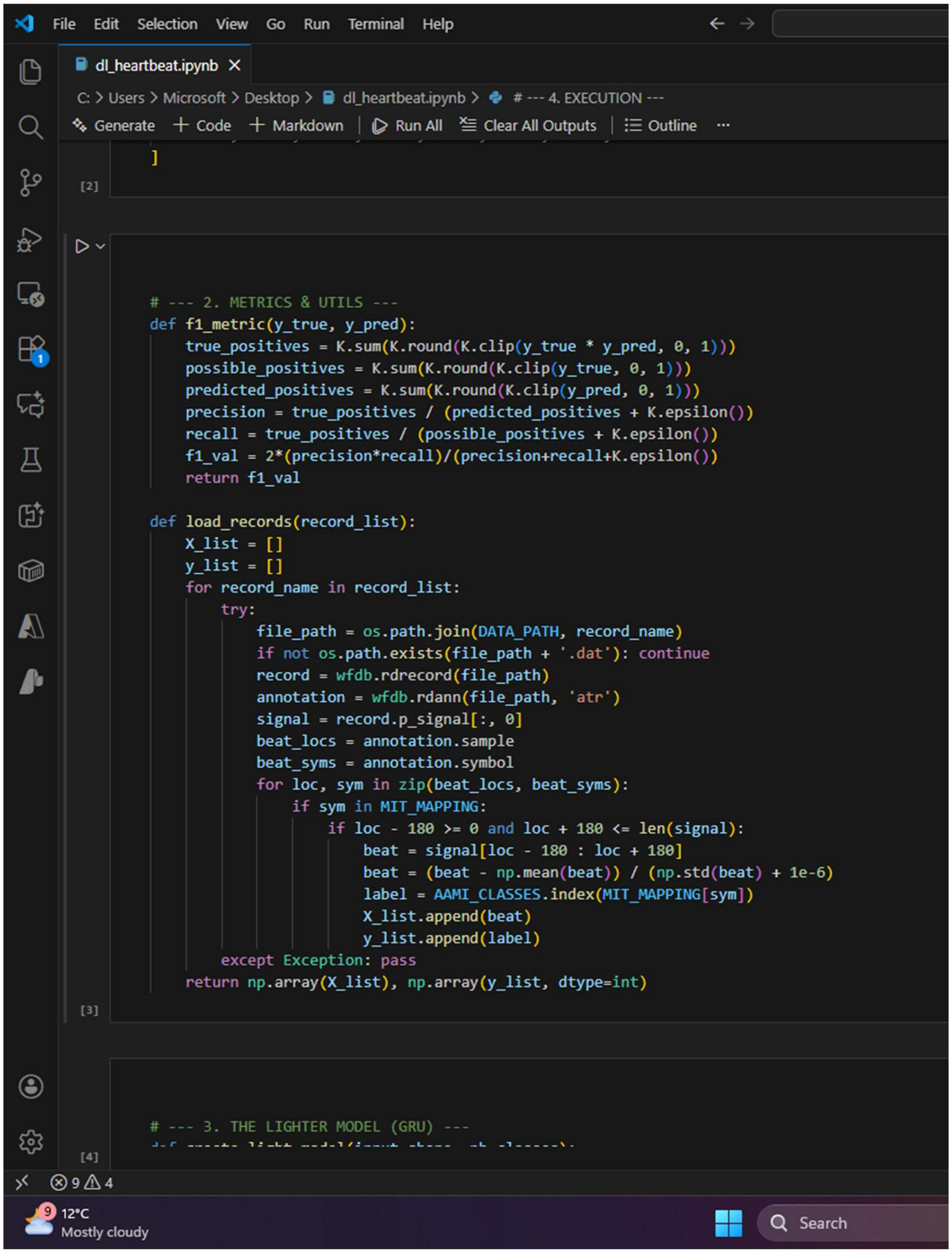

File Edit Selection View Go Run Terminal Help
dl_heartbeat.ipynb
C: > Users > Microsoft > Desktop > dl_heartbeat.ipynb > # --- 4. EXECUTION ---
Generate + Code + Markdown Run All Clear All Outputs Outline
12°C
Mostly cloudy
Search

```
# --- 1. GPU & CONFIGURATION ---
# Restart your kernel before running this!
DATA_PATH = '/workspace/physionet.org/files/mitdb/1.0.0/'

# DEBUG MODE: Use only 6 records to ensure it runs.
# Change to None later if you want the full dataset.
LIMIT_RECORDS = None

BATCH_SIZE = 32  # GRU is lighter, so 32 should be safe now
EPOCHS = 10      # Reduced epochs for a quick test run
WINDOW_SIZE = 360
AAMI_CLASSES = ['N', 'S', 'V', 'F', 'Q']

MIT_MAPPING = {
    'N': 'N', 'L': 'N', 'R': 'N', 'e': 'N', 'j': 'N',
    'A': 'S', 'a': 'S', 'J': 'S', 'S': 'S',
    'V': 'V', 'E': 'V', 'F': 'F', '/': 'Q', 'f': 'Q', 'Q': 'Q'
}

ALL_RECORDS = [
    '100', '101', '102', '103', '104', '105', '106', '107', '108', '109',
    '111', '112', '113', '114', '115', '116', '117', '118', '119', '121',
    '122', '123', '124', '200', '201', '202', '203', '205', '207', '208',
    '209', '210', '212', '213', '214', '215', '217', '219', '220', '221',
    '222', '223', '228', '230', '231', '232', '233', '234'
]
```

[2]

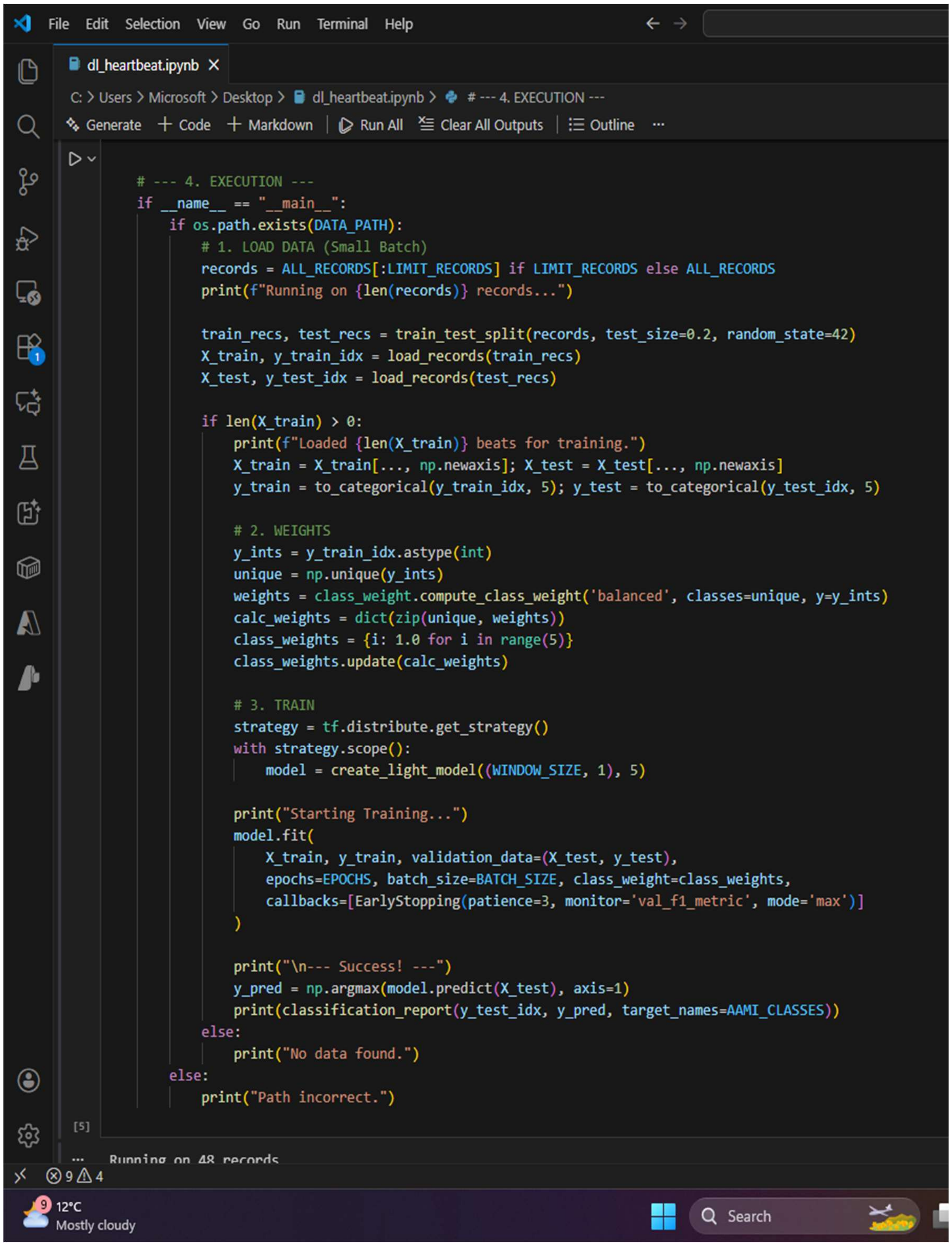
```
# --- 2. METRICS & UTILS ---
def f1_metric(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
    precision = true_positives / (predicted_positives + K.epsilon())
    recall = true_positives / (possible_positives + K.epsilon())
    f1_val = 2*(precision*recall)/(precision+recall+K.epsilon())
    return f1_val
```

[3]


File Edit Selection View Go Run Terminal Help
dl_heartbeat.ipynb
C: > Users > Microsoft > Desktop > dl_heartbeat.ipynb > # --- 4. EXECUTION ---
Generate + Code + Markdown Run All Clear All Outputs Outline
[2]
[3]
[4]
9 4
12°C
Mostly cloudy
Search

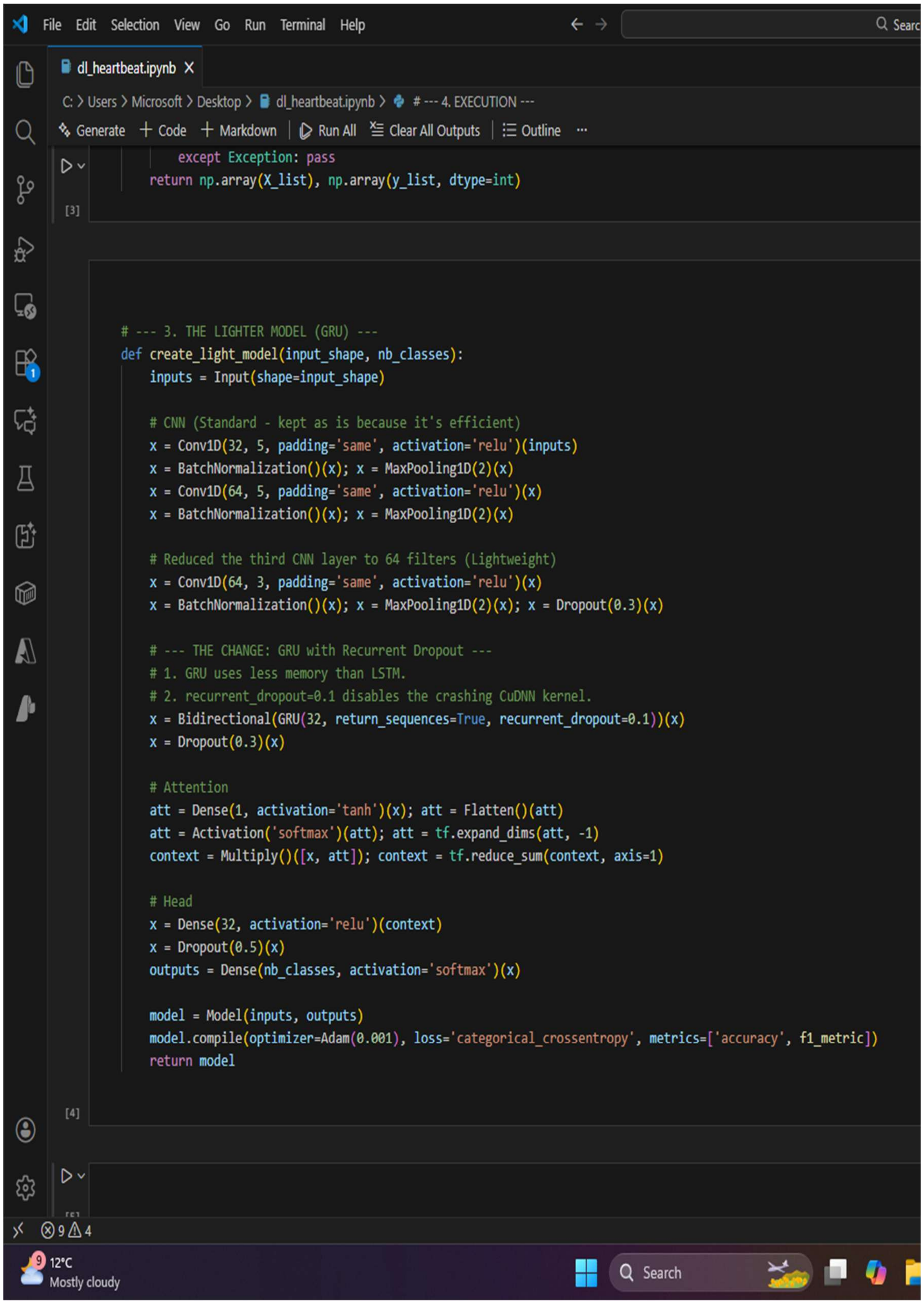
```
    ]

# --- 2. METRICS & UTILS ---
def f1_metric(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
    precision = true_positives / (predicted_positives + K.epsilon())
    recall = true_positives / (possible_positives + K.epsilon())
    f1_val = 2*(precision*recall)/(precision+recall+K.epsilon())
    return f1_val

def load_records(record_list):
    X_list = []
    y_list = []
    for record_name in record_list:
        try:
            file_path = os.path.join(DATA_PATH, record_name)
            if not os.path.exists(file_path + '.dat'): continue
            record = wfdb.rdrecord(file_path)
            annotation = wfdb.rdann(file_path, 'atr')
            signal = record.p_signal[:, 0]
            beat_locs = annotation.sample
            beat_syms = annotation.symbol
            for loc, sym in zip(beat_locs, beat_syms):
                if sym in MIT_MAPPING:
                    if loc - 180 >= 0 and loc + 180 <= len(signal):
                        beat = signal[loc - 180 : loc + 180]
                        beat = (beat - np.mean(beat)) / (np.std(beat) + 1e-6)
                        label = AAMI_CLASSES.index(MIT_MAPPING[sym])
                        X_list.append(beat)
                        y_list.append(label)
        except Exception: pass
    return np.array(X_list), np.array(y_list, dtype=int)
```

```
# --- 3. THE LIGHTER MODEL (GRU) ---
```


File Edit Selection View Go Run Terminal Help
dl_heartbeat.ipynb
C: > Users > Microsoft > Desktop > dl_heartbeat.ipynb > # --- 4. EXECUTION ---
Generate + Code + Markdown Run All Clear All Outputs Outline
[5]
9 4
12°C Mostly cloudy
Search

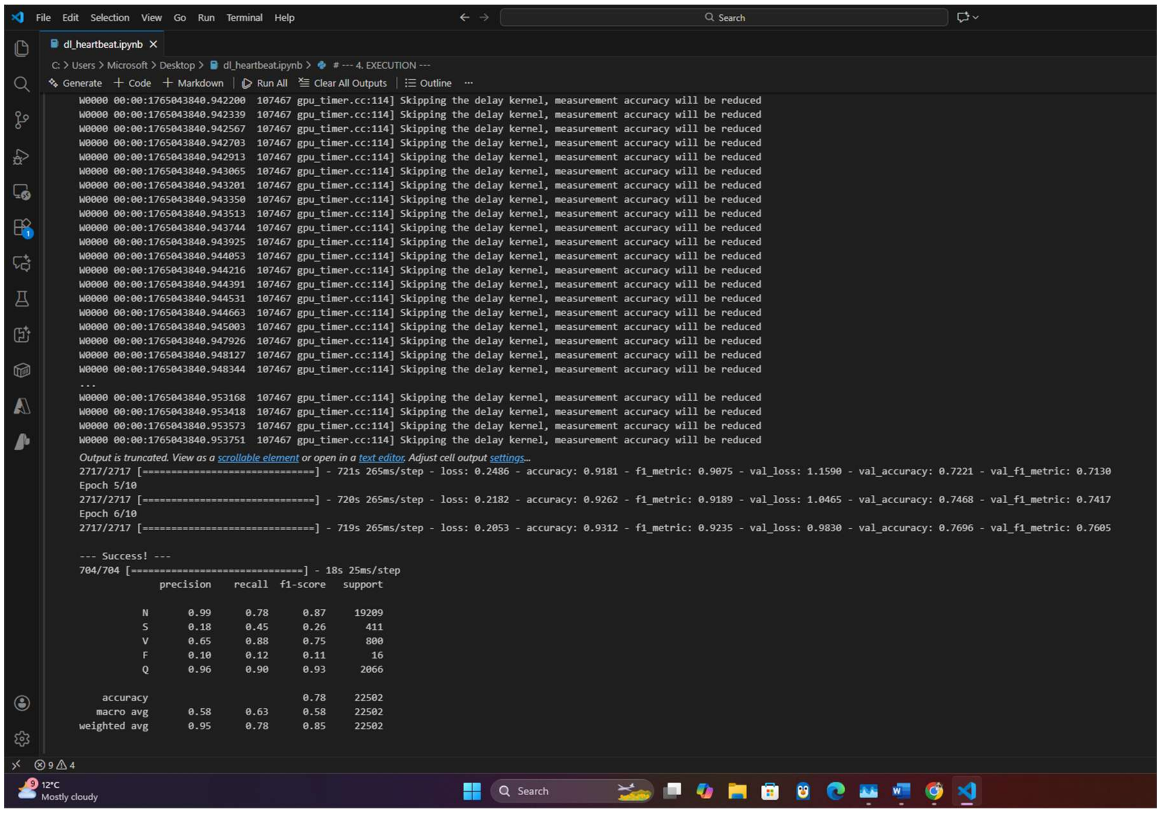
```python
# --- 4. EXECUTION ---
if __name__ == "__main__":
    if os.path.exists(DATA_PATH):
        # 1. LOAD DATA (Small Batch)
        records = ALL_RECORDS[:LIMIT_RECORDS] if LIMIT_RECORDS else ALL_RECORDS
        print(f"Running on {len(records)} records...")

        train_recs, test_recs = train_test_split(records, test_size=0.2, random_state=42)
        X_train, y_train_idx = load_records(train_recs)
        X_test, y_test_idx = load_records(test_recs)

        if len(X_train) > 0:
            print(f"Loaded {len(X_train)} beats for training.")
            X_train = X_train[..., np.newaxis]; X_test = X_test[..., np.newaxis]
            y_train = to_categorical(y_train_idx, 5); y_test = to_categorical(y_test_idx, 5)

            # 2. WEIGHTS
            y_ints = y_train_idx.astype(int)
            unique = np.unique(y_ints)
            weights = class_weight.compute_class_weight('balanced', classes=unique, y=y_ints)
            calc_weights = dict(zip(unique, weights))
            class_weights = {i: 1.0 for i in range(5)}
            class_weights.update(calc_weights)

            # 3. TRAIN
            strategy = tf.distribute.get_strategy()
            with strategy.scope():
                model = create_light_model((WINDOW_SIZE, 1), 5)

            print("Starting Training...")
            model.fit(
                X_train, y_train, validation_data=(X_test, y_test),
                epochs=EPOCHS, batch_size=BATCH_SIZE, class_weight=class_weights,
                callbacks=[EarlyStopping(patience=3, monitor='val_f1_metric', mode='max')]
            )

            print("\n--- Success! ---")
            y_pred = np.argmax(model.predict(X_test), axis=1)
            print(classification_report(y_test_idx, y_pred, target_names=AAMI_CLASSES))
        else:
            print("No data found.")
    else:
        print("Path incorrect.")
```

```
... Running on 48 records
```


File Edit Selection View Go Run Terminal Help
dl_heartbeat.ipynb
C: > Users > Microsoft > Desktop > dl_heartbeat.ipynb > # --- 4. EXECUTION ---
Generate + Code + Markdown Run All Clear All Outputs Outline
12°C
Mostly cloudy
Search

```python
        except Exception: pass
    return np.array(X_list), np.array(y_list, dtype=int)
```
[3]

```python
# --- 3. THE LIGHTER MODEL (GRU) ---
def create_light_model(input_shape, nb_classes):
    inputs = Input(shape=input_shape)

    # CNN (Standard - kept as is because it's efficient)
    x = Conv1D(32, 5, padding='same', activation='relu')(inputs)
    x = BatchNormalization()(x); x = MaxPooling1D(2)(x)
    x = Conv1D(64, 5, padding='same', activation='relu')(x)
    x = BatchNormalization()(x); x = MaxPooling1D(2)(x)

    # Reduced the third CNN layer to 64 filters (Lightweight)
    x = Conv1D(64, 3, padding='same', activation='relu')(x)
    x = BatchNormalization()(x); x = MaxPooling1D(2)(x); x = Dropout(0.3)(x)

    # --- THE CHANGE: GRU with Recurrent Dropout ---
    # 1. GRU uses less memory than LSTM.
    # 2. recurrent_dropout=0.1 disables the crashing CuDNN kernel.
    x = Bidirectional(GRU(32, return_sequences=True, recurrent_dropout=0.1))(x)
    x = Dropout(0.3)(x)

    # Attention
    att = Dense(1, activation='tanh')(x); att = Flatten()(att)
    att = Activation('softmax')(att); att = tf.expand_dims(att, -1)
    context = Multiply()([x, att]); context = tf.reduce_sum(context, axis=1)

    # Head
    x = Dense(32, activation='relu')(context)
    x = Dropout(0.5)(x)
    outputs = Dense(nb_classes, activation='softmax')(x)

    model = Model(inputs, outputs)
    model.compile(optimizer=Adam(0.001), loss='categorical_crossentropy', metrics=['accuracy', f1_metric])
    return model
```
[4]


File Edit Selection View Go Run Terminal Help
Search
dl_heartbeat.ipynb
C: > Users > Microsoft > Desktop > dl_heartbeat.ipynb > # --- 4. EXECUTION ---
Generate + Code + Markdown Run All Clear All Outputs Outline ...
9 4
12°C
Mostly cloudy
Search

```
W0000 00:00:1765043840.942200  107467 gpu_timer.cc:114] Skipping the delay kernel, measurement accuracy will be reduced
W0000 00:00:1765043840.942339  107467 gpu_timer.cc:114] Skipping the delay kernel, measurement accuracy will be reduced
W0000 00:00:1765043840.942567  107467 gpu_timer.cc:114] Skipping the delay kernel, measurement accuracy will be reduced
W0000 00:00:1765043840.942703  107467 gpu_timer.cc:114] Skipping the delay kernel, measurement accuracy will be reduced
W0000 00:00:1765043840.942913  107467 gpu_timer.cc:114] Skipping the delay kernel, measurement accuracy will be reduced
W0000 00:00:1765043840.943065  107467 gpu_timer.cc:114] Skipping the delay kernel, measurement accuracy will be reduced
W0000 00:00:1765043840.943201  107467 gpu_timer.cc:114] Skipping the delay kernel, measurement accuracy will be reduced
W0000 00:00:1765043840.943350  107467 gpu_timer.cc:114] Skipping the delay kernel, measurement accuracy will be reduced
W0000 00:00:1765043840.943513  107467 gpu_timer.cc:114] Skipping the delay kernel, measurement accuracy will be reduced
W0000 00:00:1765043840.943744  107467 gpu_timer.cc:114] Skipping the delay kernel, measurement accuracy will be reduced
W0000 00:00:1765043840.943925  107467 gpu_timer.cc:114] Skipping the delay kernel, measurement accuracy will be reduced
W0000 00:00:1765043840.944053  107467 gpu_timer.cc:114] Skipping the delay kernel, measurement accuracy will be reduced
W0000 00:00:1765043840.944216  107467 gpu_timer.cc:114] Skipping the delay kernel, measurement accuracy will be reduced
W0000 00:00:1765043840.944391  107467 gpu_timer.cc:114] Skipping the delay kernel, measurement accuracy will be reduced
W0000 00:00:1765043840.944531  107467 gpu_timer.cc:114] Skipping the delay kernel, measurement accuracy will be reduced
W0000 00:00:1765043840.944663  107467 gpu_timer.cc:114] Skipping the delay kernel, measurement accuracy will be reduced
W0000 00:00:1765043840.945003  107467 gpu_timer.cc:114] Skipping the delay kernel, measurement accuracy will be reduced
W0000 00:00:1765043840.947926  107467 gpu_timer.cc:114] Skipping the delay kernel, measurement accuracy will be reduced
W0000 00:00:1765043840.948127  107467 gpu_timer.cc:114] Skipping the delay kernel, measurement accuracy will be reduced
W0000 00:00:1765043840.948344  107467 gpu_timer.cc:114] Skipping the delay kernel, measurement accuracy will be reduced
...
W0000 00:00:1765043840.953168  107467 gpu_timer.cc:114] Skipping the delay kernel, measurement accuracy will be reduced
W0000 00:00:1765043840.953418  107467 gpu_timer.cc:114] Skipping the delay kernel, measurement accuracy will be reduced
W0000 00:00:1765043840.953573  107467 gpu_timer.cc:114] Skipping the delay kernel, measurement accuracy will be reduced
W0000 00:00:1765043840.953751  107467 gpu_timer.cc:114] Skipping the delay kernel, measurement accuracy will be reduced
```

*Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...*

```
2717/2717 [==============================] - 721s 265ms/step - loss: 0.2486 - accuracy: 0.9181 - f1_metric: 0.9075 - val_loss: 1.1590 - val_accuracy: 0.7221 - val_f1_metric: 0.7130
Epoch 5/10
2717/2717 [==============================] - 720s 265ms/step - loss: 0.2182 - accuracy: 0.9262 - f1_metric: 0.9189 - val_loss: 1.0465 - val_accuracy: 0.7468 - val_f1_metric: 0.7417
Epoch 6/10
2717/2717 [==============================] - 719s 265ms/step - loss: 0.2053 - accuracy: 0.9312 - f1_metric: 0.9235 - val_loss: 0.9830 - val_accuracy: 0.7696 - val_f1_metric: 0.7605

--- Success! ---
704/704 [==============================] - 18s 25ms/step
              precision    recall  f1-score   support

           N       0.99      0.78      0.87     19209
           S       0.18      0.45      0.26       411
           V       0.65      0.88      0.75       800
           F       0.10      0.12      0.11        16
           Q       0.96      0.90      0.93      2066

    accuracy                           0.78     22502
   macro avg       0.58      0.63      0.58     22502
weighted avg       0.95      0.78      0.85     22502
```

## Refrences

[b1] O. Kiranyaz, T. Ince, and M. Gabbouj, "Real-Time Patient-Specific ECG Classification with 1D Convolutional Neural Networks," *IEEE Transactions on Biomedical Engineering*, vol. 63, no. 3, pp. 664-675, 2016.

[b2] U. R. Acharya et al., "A deep convolutional neural network model to classify heartbeats," *Computers in Biology and Medicine*, vol. 89, pp. 389-396, 2017.

[b3] Z. Li et al., "Classification of ECG signals using a deep residual network," *IEEE Access*, vol. 8, pp. 35686-35695, 2020.

[b4] O. Yildirim, "A novel wavelet sequence based on deep bidirectional LSTM network model for ECG signal classification," *Computers in Biology and Medicine*, vol. 96, pp. 189-202, 2018.

[b5] S. Saadatnejad, M. Oveisi, and M. Hashemi, "LSTM-based ECG classification for continuous monitoring on personal wearable devices," *IEEE Journal of Biomedical and Health Informatics*, vol. 24, no. 2, pp. 515-523, 2020.

[b6] S. L. Oh, E. Y. Ng, and R. S. Tan, "Automated diagnosis of arrhythmia using combination of CNN and LSTM techniques with variable length heart beats," *Computers in Biology and Medicine*, vol. 102, pp. 278-287, 2018.

[b7] M. Wu et al., "A hybrid method for arrhythmia classification using CNN and LSTM with focal loss," *Symmetry*, vol. 13, no. 11, p. 2169, 2021.

[b8] Q. Yao et al., "Multi-class arrhythmia detection from 12-lead varied-length ECG using attention-based time-incremental convolutional neural network," *Information Fusion*, vol. 53, pp. 174-182, 2020.

[b9] C. Che et al., "Constrained transformer network for ECG signal classification and interpretability," *BMC Medical Informatics and Decision Making*, vol. 21, no. 1, pp. 1-13, 2021.

[b10] J. Wang et al., "Two-stage scalable classifier for arrhythmia detection using transformer," *IEEE Access*, vol. 10, pp. 36767-36778, 2022.

[b11] S. Mousavi and F. Afghah, "Inter-patient ECG heartbeat classification with deep transfer learning," in *2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pp. 1202-1206, 2019.

[b12] A. H. Ribeiro et al., "Automatic diagnosis of the 12-lead ECG using a deep neural network," *Nature Communications*, vol. 11, no. 1, p. 1760, 2020.