**Tài Liệu Hướng Dẫn Huấn Luyện Mô Hình Phân Loại Ảnh Điện Tâm Đồ ECG Tim Mạch**

**ECG Classification using Convolutional Neural Network (CNN)**

**I. Giới thiệu**

Tài liệu này hướng dẫn chi tiết quy trình **xây dựng, huấn luyện, đánh giá và triển khai** mô hình học sâu (*Deep Learning*) sử dụng **mạng nơ-ron tích chập (Convolutional Neural Network – CNN)** để **phân loại tín hiệu điện tim (ECG)**.

Mục tiêu của mô hình là **phân biệt tín hiệu tim bình thường (Normal)** và **bất thường (Abnormal)**, giúp hỗ trợ bác sĩ trong việc **phát hiện sớm rối loạn nhịp tim, rung nhĩ, hoặc các vấn đề tim mạch khác**.

CNN có khả năng tự động học các đặc trưng (feature) của tín hiệu, giảm thiểu việc trích xuất thủ công, giúp cải thiện độ chính xác và tốc độ xử lý so với các phương pháp truyền thống.

**II. Tổng quan về tín hiệu ECG**

**1. ECG là gì?**

ECG (Electrocardiogram) là tín hiệu điện ghi lại hoạt động của tim theo thời gian.  
Một chu kỳ ECG thường bao gồm các thành phần:

* **Sóng P:** Biểu thị sự khử cực của tâm nhĩ.
* **Phức bộ QRS:** Thể hiện sự khử cực của tâm thất.
* **Sóng T:** Biểu thị quá trình tái cực của tâm thất.

**2. Ý nghĩa**

Phân tích tín hiệu ECG giúp phát hiện:

* Nhịp tim nhanh hoặc chậm bất thường.
* Nhịp ngoại tâm thu, rung nhĩ.
* Thiếu máu cơ tim hoặc nhồi máu cơ tim.

**3. Đặc điểm tín hiệu**

* Là tín hiệu **1 chiều (1D)**, dạng chuỗi thời gian.
* Thường có **nhiễu và biến thiên biên độ**, nên cần tiền xử lý kỹ.
* Tần số lấy mẫu phổ biến: **360 Hz hoặc 500 Hz**.

**III. Dữ liệu huấn luyện (Dataset)**

**1. Nguồn dữ liệu**

Có thể sử dụng các tập dữ liệu chuẩn như:

* **MIT-BIH Arrhythmia Database**
* **PTB Diagnostic ECG Database**
* **PhysioNet Challenge Dataset**

Mỗi mẫu dữ liệu gồm:

* Tín hiệu ECG (chuỗi giá trị cường độ điện theo thời gian).
* Nhãn (Label): Normal / Abnormal hoặc nhiều lớp bệnh lý khác.

**2. Chuẩn bị dữ liệu**

* Nạp dữ liệu bằng **Pandas hoặc NumPy**.
* Kiểm tra giá trị thiếu (NaN) và loại bỏ nhiễu.
* Chuẩn hóa giá trị tín hiệu về **[0,1]** để ổn định quá trình huấn luyện.
* Cắt (hoặc zero-padding) tín hiệu để có độ dài cố định (ví dụ 187 điểm).

**3. Chia tập train/test**

Sử dụng hàm:

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

**IV. Kiến trúc mạng CNN cho tín hiệu ECG**

CNN (Convolutional Neural Network) là mạng nơ-ron đặc biệt phù hợp cho **dữ liệu ảnh và chuỗi thời gian** nhờ khả năng **tự động học đặc trưng cục bộ** qua các lớp tích chập (convolution layers).

**1. Lý do chọn CNN cho ECG**

* Có khả năng học **pattern theo thời gian** của tín hiệu.
* Giảm số lượng tham số so với mạng fully connected.
* Chống overfitting tốt khi kết hợp Dropout & Pooling.

**2. Cấu trúc phổ biến**

Input (187, 1)

↓

Conv1D(32, kernel\_size=3, activation='relu')

↓

MaxPooling1D(2)

↓

Conv1D(64, kernel\_size=3, activation='relu')

↓

MaxPooling1D(2)

↓

Flatten

↓

Dense(128, activation='relu')

↓

Dropout(0.5)

↓

Dense(2, activation='softmax')

**3. Giải thích các lớp**

* **Conv1D:** Lọc tín hiệu để phát hiện mẫu cục bộ (như QRS complex).
* **MaxPooling1D:** Giảm kích thước dữ liệu, giữ đặc trưng chính.
* **Flatten:** Chuyển tensor 3D thành vector 1D.
* **Dense:** Lớp kết nối đầy đủ để phân loại.
* **Softmax:** Tính xác suất cho từng lớp đầu ra.

**V. Môi trường và thư viện cần thiết**

**1. Cấu hình phần mềm**

* Python 3.8+
* Jupyter Notebook hoặc Google Colab

**2. Cài đặt thư viện**

pip install numpy pandas matplotlib seaborn scikit-learn tensorflow keras

**3. Thư viện chính**

| **Thư viện** | **Chức năng** |
| --- | --- |
| NumPy | Xử lý ma trận & vector tín hiệu |
| Pandas | Nạp và xử lý dữ liệu |
| TensorFlow / Keras | Xây dựng và huấn luyện CNN |
| Matplotlib / Seaborn | Trực quan hóa dữ liệu |
| Scikit-learn | Chia dữ liệu & đánh giá mô hình |

**VI. Tiền xử lý dữ liệu ECG**

**1. Chuẩn hóa và định dạng lại**

X = X / np.max(X)

X = X.reshape(X.shape[0], X.shape[1], 1)

**2. One-hot encoding nhãn**

from tensorflow.keras.utils import to\_categorical

y = to\_categorical(y)

**3. Chia train/test**

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2)

**VII. Huấn luyện mô hình CNN**

**1. Xây dựng mô hình**

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv1D, MaxPooling1D, Flatten, Dense, Dropout

model = Sequential([

Conv1D(32, kernel\_size=3, activation='relu', input\_shape=(187,1)),

MaxPooling1D(pool\_size=2),

Conv1D(64, kernel\_size=3, activation='relu'),

MaxPooling1D(pool\_size=2),

Flatten(),

Dense(128, activation='relu'),

Dropout(0.5),

Dense(2, activation='softmax')

])

**2. Compile mô hình**

model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

**3. Huấn luyện**

history = model.fit(X\_train, y\_train, epochs=50, batch\_size=32, validation\_data=(X\_test, y\_test))

**4. Theo dõi huấn luyện**

Quan sát biểu đồ accuracy và loss để xác định sự hội tụ.

**VIII. Đánh giá và trực quan hóa kết quả**

**1. Đánh giá mô hình**

loss, acc = model.evaluate(X\_test, y\_test)

print("Test Accuracy:", acc)

**2. Trực quan hóa quá trình huấn luyện**

import matplotlib.pyplot as plt

plt.plot(history.history['accuracy'], label='Train Accuracy')

plt.plot(history.history['val\_accuracy'], label='Validation Accuracy')

plt.legend()

plt.title('Model Accuracy')

plt.show()

**3. Ma trận nhầm lẫn**

from sklearn.metrics import confusion\_matrix, classification\_report

y\_pred = model.predict(X\_test)

y\_pred\_classes = np.argmax(y\_pred, axis=1)

y\_true = np.argmax(y\_test, axis=1)

print(confusion\_matrix(y\_true, y\_pred\_classes))

print(classification\_report(y\_true, y\_pred\_classes))

**IX. Lưu và retrain mô hình**

**1. Lưu mô hình**

model.save('ecg\_cnn\_model.h5')

**2. Tải mô hình đã lưu**

from tensorflow.keras.models import load\_model

loaded\_model = load\_model('ecg\_cnn\_model.h5')

**3. Retrain với dữ liệu mới**

* Bước 1: Nạp dữ liệu ECG mới.
* Bước 2: Tiền xử lý tương tự (chuẩn hóa, reshape, one-hot).
* Bước 3: Gọi model.fit() lại để tiếp tục học.
* Bước 4: Lưu mô hình mới (model.save('ecg\_cnn\_updated.h5')).

**X. Kết luận và hướng phát triển**

**1. Kết luận**

Mô hình CNN cho phân loại tín hiệu ECG đạt hiệu quả cao (accuracy thường trên 95%) nhờ khả năng học tự động đặc trưng từ tín hiệu thô.  
Việc áp dụng mô hình giúp **tự động hóa phân tích ECG**, **giảm tải công việc cho bác sĩ**, và **nâng cao độ chính xác chẩn đoán**.

**2. Hướng phát triển**

* Sử dụng **mạng kết hợp CNN + LSTM** để học phụ thuộc theo thời gian dài.
* Huấn luyện trên **ECG đa kênh (12-lead)** để mô phỏng thực tế lâm sàng.
* Áp dụng kỹ thuật **Explainable AI (XAI)** để giải thích quyết định mô hình.
* Tích hợp mô hình vào **thiết bị IoT hoặc ứng dụng y tế di động.**