BỘ GIÁO DỤC VÀ ĐÀO TẠO TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN



KHOA CÔNG NGHỆ THÔNG TIN

NHẬP MÔN HỌC MÁY

**BÁO CÁO BÀI TẬP MÔN HỌC**

**NỘI DUNG: PRACTICE COMPETITION PHASE 03- ADVANCED ARCHITECTURES NEURAL NETWORK**

**Giảng viên hướng dẫn** :Nguyễn Thanh Tình Huỳnh Lâm Hải Đăng

**Lớp** :CQ2022/24

**Sinh viên thực hiện** :Đinh Viết Lợi-22120188 Nguyễn Trần Lợi- 22120190 Nguyễn Nhật Long-22120194

*Hồ Chí Minh, ngày 7 tháng 6 năm 2025*

# MỤC LỤC

Contents

[MỤC LỤC 1](#_Toc200272849)

[PHẦN 1: TỔNG QUAN ĐỒ ÁN 3](#_Toc200272850)

[I. Tổng quan đồ án 3](#_Toc200272851)

[1. Giới thiệu đồ án 3](#_Toc200272852)

[2. Yêu cầu đồ án 3](#_Toc200272853)

[II. Kết quả đạt được 4](#_Toc200272854)

[PHẦN 2: BÁO CÁO NHÓM 5](#_Toc200272855)

[I. Thông tin nhóm 5](#_Toc200272856)

[II. Quy trình thực hiện 5](#_Toc200272857)

[PHẦN 3: MODEL DEVELOPMENT 6](#_Toc200272858)

[I. Phân tích bộ dữ liệu 6](#_Toc200272859)

[II. Chi tiết quá trình xây dựng mô hình 6](#_Toc200272860)

[1. Tiền xử lý dữ liệu 6](#_Toc200272861)

[2. Transfomer 6](#_Toc200272862)

[3. Convolutional neural network (CNN) 6](#_Toc200272863)

[4. Recurrent neural network (RNN) 7](#_Toc200272864)

[III. Các phương pháp xử lý khác 7](#_Toc200272865)

[PHẦN 4: MODEL EVALUATION 8](#_Toc200272866)

[I. Các bộ chỉ số đánh giá 8](#_Toc200272867)

[II. Đánh giá và cải thiện bộ chỉ số 9](#_Toc200272868)

[1. Transformer 9](#_Toc200272869)

[2. Convolutional neural network (CNN) 9](#_Toc200272870)

[3. Recurrent neural network (RNN) 9](#_Toc200272871)

[PHẦN 5: ĐÁNH GIÁ PHƯƠNG PHÁP 9](#_Toc200272872)

[I. Phương pháp tiền xử lý dữ liệu 9](#_Toc200272873)

[II. Phương pháp lựa chọn và xây dựng mô hình 10](#_Toc200272874)

[1. Transfomer 10](#_Toc200272875)

[2. Convolutional neural network (CNN) 10](#_Toc200272876)

[3. Recurrent neural network (RNN) 10](#_Toc200272877)

[PHẦN 6: TÀI LIỆU THAM KHẢO 10](#_Toc200272878)

[I. Tài liệu nhóm 10](#_Toc200272879)

[II. Tài liệu tham khảo 10](#_Toc200272880)

# PHẦN 1: TỔNG QUAN ĐỒ ÁN

## Tổng quan đồ án

## Giới thiệu đồ án

* + - Đồ án Practice Competition là đồ án kết thúc môn học của bộ môn Nhập môn Học máy của lớp CQ2022/24 Trường Đại học Khoa học Tự nhiên học kỳ II năm học 2024-2025.
    - Đồ án tập trung vào thực hành quy trình xây dựng các mô hình học máy phục với 3 giai đoạn chính: Exploratory Data, Model Development, and Model Evaluation. Trọng tâm quy trình sẽ nằm ở hai giai đoạn sau.
    - Hình thức thực hiện đồ án được tổ chức dưới dạng một cuộc thi trên Kaggle với thể thức cá nhân hoặc theo nhóm. Các thông tin về cuộc thi được trình bày tại Kaggle competition.
    - Bài toán được đặt ra bởi cuộc thi sẽ được yêu cầu giải quyết trong vòng 3 tuần với 3 giai đoạn (phase) khác nhau. Tuần thứ ba cũng là tuần cuối cùng của cuộc thi được yêu cầu sử dụng các kiến trúc deep learning nâng cao hơn như CNN, RNN, Transformer.

## Yêu cầu đồ án

* + - Các sinh viên được mong đợi tuân thủ một quy trình phát triển cho dự án Học máy thực tế, được cấu trúc với ba bước chính: Phân tích Dữ liệu Khám phá (Exploratory Data Analysis), Phát triển Mô hình (Model Development) và Đánh giá Mô hình (Model Evaluation).
    - Bài tập được thực hiện dưới hình thức một cuộc thi Kaggle, sinh viên cần tham gia theo thể thức cá nhân hoặc theo đội nhóm.
    - Dự án được lên kế hoạch diễn ra trong tổng cộng 3 tuần, được cấu trúc thành 3 giai đoạn riêng biệt. Mỗi giai đoạn kéo dài một tuần và bao gồm toàn bộ chu trình của hai bước chính: Phát triển Mô hình, và Đánh giá và Phân tích Lỗi.
    - Mỗi sinh viên/ nhóm sinh viên được yêu cầu nộp quá trình thực hiện hàng tuần bao gồm mã nguồn Jupyter notebook và báo cáo cho từng Phase.
    - Mã nguồn Jupyter Notebook cần có: thông tin sinh viên/ nhóm sinh viên, giải thích chi tiết từng bước bằng hình ảnh minh hoạ, biểu đồ, công thức, chú thích (comment) từng bước xử lý và trực quan, notebook được format tốt.
    - Báo cáo bao gồm: thông tin sinh viên/ nhóm sinh viên, tự đánh giá kết quả, ưu và nhược điểm của các phương pháp xử lý, mô hình và frameworks khác nhau. Thông tin chi tiết mỗi giai đoạn cần đảm bảo:
      * **Model development**: Trình bày việc lựa chọn mô hình, thiết kế kiến trúc và thiết lập huấn luyện cùng với lý do đằng sau các lựa chọn của bạn.
      * **Đánh giá Mô hình:** Báo cáo về hiệu suất mô hình bằng cách sử dụng các chỉ số phù hợp, đồng thời bao gồm các quan sát về thời gian chạy, mức sử dụng tài nguyên tính toán. Cung cấp phân tích chi tiết về bất kỳ loại lỗi nào mà sinh viên và/hoặc nhóm đã xác định là điểm yếu của mô hình.

## Kết quả đạt được

* Quá trình xây dựng mô hình của nhóm được áp dụng mạnh mẽ và đa dạng các thư viện, framwork được hỗ trợ sẵn, tận dụng tối đa được sức mạnh từ các kiến trúc phức tạp để giải quyết bài toán.
* Nhóm áp dụng nhiều các phương pháp xử lý khác nhau nhằm chuẩn bị được bộ dữ liệu tốt nhất, phân tích các đặc điểm của dữ liệu hỗ trợ, gây khó khăn cho quá trình huấn luyện.
* Thực hiện huấn luyện nhiều lần với các bộ siêu tham số, kiến trúc mạng neural và các phương pháp khác nhau nhằm đánh giá quy luật, tìm ra bộ kết hợp tốt nhất để lựa chọn được phương thức huấn luyện mô hình phù hợp.
* Kết quả mô hình cho ra kết quả tốt, có thể được ứng dụng để giải quyết tốt vấn đề được đặt ra.
* Thông tin cụ thể về quá trình làm cũng như động lực được trình bày rõ ràng trong notebook và báo cáo.

# PHẦN 2: BÁO CÁO NHÓM

## Thông tin nhóm

|  |  |
| --- | --- |
| **Thành viên** | **Vai trò** |
| Đinh Viết Lợi | Convolutional Neural Network |
| Nguyễn Trần Lợi | Recurrent Neural Network |
| Nguyễn Nhật Long | Transformer |

## Quy trình thực hiện

Tận dụng số lượng submit có hạn hằng ngày trong thời gian ngắn, nhóm liên tục cập nhật source code hằng ngày để kiểm tra mức độ cải thiện, cũng như xác định độ hiệu quả của các phương pháp qua từng phiên bản:

* + Ngày 1:
    - Xây dựng Transformer bằng thư viện Keras với các kỹ thuật cơ bản.
    - Áp dụng mô hình RNN bằng thư viện Keras với các kĩ thuật cơ bản.
  + Ngày 2:
    - Tối ưu hóa bằng điều chỉnh tham số: số lớp, số epoch, early stopping, batch size,....
    - Áp dụng mô hình CNN cơ bản với 2 chiều tích chập cho bộ dữ liệu.
    - Tối ưu cho mô hình RNN bằng cách chỉnh các tham số, cấu trúc mạng, thêm dropout.
  + Ngày 3:
    - Cải tiến mạng CNN, thêm số lớp, bổ sung dropout, bổ sung kỹ thuật Squeeze-and-Excitation Block.
    - Cải tiến bằng LSTM, chuẩn hóa bằng LayerNormalization, chia và xáo trộn dữ liệu huấn luyện bằng StratifiedShuffleSplit
    - Tối ưu learning rate của RNN bằng ReduceLROnPlateau.
  + Ngày 4:
    - Cải tiến bằng Attention Pooling kết hợp với LayerNormalization.
    - Áp dụng kỹ thuật MultiHeadAttention & LayerNormalization
    - Chạy CNN với nhiều bộ siêu tham số nhằm tìm ra bộ siêu tham số tối ưu, thêm các kỹ thuật mix-up, data augmentation.
    - Tiếp tục tối ưu cho mô hình RNN bằng các kĩ thuật k-fold, regularization, sử dụng các loại lớp khác nhau như LSTM, SpatialDropout, GRU, Bidirectional.
  + Ngày 5:
    - Cải tiến bằng Positional Encoding, Global AveragePooling. Sử dụng TransformerEncoder gọi từ thư viện Keras.
    - Điều chỉnh tham số: num\_heads, intermidiate\_dim, test\_size, random\_state, learning\_rate,.....
    - Chốt kỹ thuật và bộ tham số tối ưu nhất
  + Ngày 6: Tổng hợp phương án tối ưu cho việc huấn luyện mô hình, viết báo cáo tổng hợp.

# PHẦN 3: MODEL DEVELOPMENT

## Phân tích bộ dữ liệu

* + Bộ dữ liệu là tập các record được lưu dưới dạng bảng của loại dữ liệu supervised với nhãn is\_turkey xác định xem liệu một audio âm thanh có phải là tiếng gà tây hay không.
  + Các thuộc tính của dữ liệu bao gồm: các đặc trưng âm thanh của từng frame ~ 1s được embedding thành một features vector gồm 128 chiều thông thường đoạn audio n giây sẽ có ~n frame, các thuộc tính còn lại bao gồm thời gian bắt đầu và kết thúc của video.
  + Quan sát đặc tính của bộ dữ liệu cụ thể là các thuộc tính cho thấy chỉ có thuộc tính audio\_embedding sẽ là trọng tâm xác định nhãn của dữ liệu, các thuộc tính còn lại không đóng vai trò trong việc huấn luyện mô hình.

## Chi tiết quá trình xây dựng mô hình

1. **Tiền xử lý dữ liệu**
   * **Array vector embedding:** giá trị của feature “audio\_embedding” sẽ được giữ nguyên là chuỗi n giây (~0.96s) với mỗi giây được embedding thành 128 chiều. Việc áp dụng mean, max... để rút ngắn số lượng sẽ làm mất đi lượng lớn thông tin của dữ liệu nên sẽ không được áp dụng, đồng thời các kiến trúc nâng cao có thể hỗ trợ tốt với kiểu dữ liệu này giữ nguyên này.
   * **Padding:** Mặc dù hầu hết tất cả thuộc tính ‘audio\_embedding” của các mẫu (1125/1195) đều có độ dài của vector là 10 (video kéo dài 10s) nhưng để tránh mất mát một số mẫu đối với bộ dữ liệu khan hiếm này padding sẽ được sử dụng để “đệm” thêm dữ liệu vào các video ngắn hơn với mục đích đưa tất cả các mẫu về cùng kích thước.

A screen shot of a computer code

AI-generated content may be incorrect.

* + **Chuẩn hóa dữ liệu**: Sau khi rút trích đặc trưng cố định, ta sử dụng **StandardScaler** để chuẩn hóa dữ liệu đầu vào. Phương pháp này chuẩn hóa mỗi đặc trưng sao cho chúng có **trung bình bằng 0 và độ lệch chuẩn bằng 1**, giúp mô hình học máy hội tụ nhanh hơn và tránh bị lệ thuộc vào độ lớn của các đặc trưng.

A screen shot of a computer code

AI-generated content may be incorrect.

* + **Chia tập dữ liệu train và validation:** Dữ liệu sau chuẩn hóa được chia thành hai tập: **tập huấn luyện (training set)** và **tập xác thực (validation set)** để đánh giá khả năng tổng quát hóa của mô hình. Việc chia dữ liệu này đảm bảo mô hình không bị đánh giá trên chính dữ liệu nó đã học. Với dataset cụ thể này, tỉ lệ 70/30 (train/test) cho hiệu quả tốt và hợp lí nhất.



1. **Transfomer**
   * **StratifiedShuffleSplit:**
     + Kỹ thuật chia tập dữ liệu thành train/test sao cho tỷ lệ của các lớp (label) giống nhau ở cả hai tập.
     + Shuffle (xáo trộn) dữ liệu trước khi chia để tránh hiện tượng dữ liệu có thứ tự cố định.
     + Đảm bảo mẫu đại diện tốt hơn cho từng lớp, đặc biệt hữu ích với dữ liệu mất cân bằng.
   * **LayerNormalization**
     + Chuẩn hóa giá trị của từng mẫu (sample) theo từng feature trên 1 layer.
     + Giúp giảm sự phụ thuộc vào phân phối đầu vào, tăng tốc độ học và ổn định mạng.
   * **MultiHeadAttention**
     + Cơ chế Attention dùng nhiều "đầu" (head) song song để học các mối liên hệ khác nhau trong dữ liệu.
     + Mỗi head có thể tập trung vào một kiểu thông tin riêng biệt (vị trí, ngữ nghĩa, v.v.)
     + Kết quả từ các head được hợp nhất để cải thiện khả năng biểu diễn của mô hình.
   * **LSTM**
     + Mạng nơ-ron hồi tiếp (RNN) có các "cửa" (gate) giúp ghi nhớ hoặc quên thông tin trong chuỗi.
     + Khắc phục nhược điểm mất thông tin khi xử lý chuỗi dài so với RNN thường
     + Được dùng nhiều trong NLP, chuỗi thời gian, v.v.
   * **Attention Pooling**
     + Cơ chế gán trọng số (attention weight) cho các phần của đầu vào khi tổng hợp (pooling) thông tin.
     + Thay vì trung bình hoặc max, mô hình học trọng số để nhấn mạnh thông tin quan trọng
     + Giúp tăng tính linh hoạt và hiệu quả biểu diễn.
   * **Global Average Pooling**
     + Hàm pooling tính trung bình tất cả giá trị trên mỗi channel (toàn bộ không gian).
     + Giảm chiều dữ liệu (ví dụ từ ảnh thành vector), giữ lại thông tin toàn cục
     + Thường dùng sau các CNN để làm đầu vào cho lớp fully-connected.
   * **TransformerEncoder** 
     + Thành phần encoder của Transformer, gồm nhiều lớp attention + feedforward.
     + Mã hóa thông tin đầu vào (chuỗi) thành biểu diễn có ngữ cảnh giàu thông tin.
     + Không tuần tự như RNN, nên xử lý song song rất nhanh.
   * **Positional Encoding**
     + Kỹ thuật thêm thông tin vị trí vào embedding khi dùng Transformer (vốn không có thứ tự).
     + Các vector vị trí (sin/cos hoặc learned) cộng vào embedding của token.
     + Giúp mô hình nhận biết thứ tự trong chuỗi.
   * **Kiến trúc mạng:** 
     + Input
       - inputs = Input(shape=(seq\_len, d\_model))
       - Đầu vào có shape (batch\_size, seq\_len, d\_model):  
          → 1 chuỗi có seq\_len bước, mỗi bước có vector d\_model chiều (embedding vector hoặc feature vector).
     + PositionalEncoding:
       - x = PositionalEncoding()(inputs)
       - Mục đích: Thêm thông tin về vị trí của từng bước trong chuỗi.
       - Transformer không có cơ chế "hiểu thứ tự", nên bạn cộng thêm vector Positional Encoding vào embedding.
       - Positional Encoding được tính bằng sin/cos với tần số khác nhau cho từng chiều → giúp mô hình biết bước thứ mấy trong chuỗi.
     + for \_ in range(2):
       - x = keras\_nlp.layers.TransformerEncoder(...)
       - Có 2 lớp TransformerEncoder, mỗi lớp gồm:
       - Multi-head self-attention: cho phép từng bước trong chuỗi "nhìn" mọi bước khác trong chuỗi để học quan hệ.
       - Feedforward network: xử lý từng bước sau attention.
       - Dropout: chống overfitting.
       - Residual connection + LayerNorm (ẩn trong TransformerEncoder).
       - Kết quả sau 2 lớp → embedding của từng bước đã được "ngữ cảnh hóa": mỗi bước biết thông tin từ cả chuỗi.
     + GlobalAveragePooling:
       - x = GlobalAveragePooling1D()(x)
       - Mục đích: Tóm tắt toàn bộ chuỗi thành 1 vector duy nhất (per sample).
       - Thay vì chỉ lấy bước cuối hay đầu, ta tính trung bình tất cả các bước → giữ thông tin toàn cục.
     + Drop out & Dense
       - x = Dropout(0.5)(x)
       - x = Dense(256, activation="relu")(x)
       - x = Dropout(0.5)(x)
       - Ngẫu nhiên tắt bớt ½ neuron để chống overfitting
     + Output
       - outputs = Dense(1, activation="sigmoid")(x)
       - Trả về xác xuất, phù hợp với phân loại nhị phân.
2. **Convolutional neural network (CNN)**
   * **Data augmentaion:** thêm nhiễu Gaussian (noise) vào đầu dữ liệu với mục đích chống nhiễu cho mô hình, tạo thêm mẫu và giúp mô hình học được hiệu quả hơn với độ nhiễu nhẹ.

A computer screen shot of white text

AI-generated content may be incorrect.

* + **Kiến trúc mạng Neural:**
    - Transpose đầu vào: từ (batch\_size, time\_steps, 128) → (batch\_size, 128, time\_steps)
    - Block 1:
      * Conv1D: 128 → 64, kernel=3, padding=1
      * ReLU
      * SEBlock(64):
        + AdaptiveAvgPool1D(1) → Linear(64 → 4 → 64) → Sigmoid
        + Nhân lại với đầu vào theo kênh
  + Dropout(0.3)
    - Block 2:
      * Conv1D: 64 → 128, kernel=3, padding=1
      * ReLU
      * SEBlock(128) (tương tự SEBlock trên)
      * Dropout(0.3)
    - Global Average Pooling: AdaptiveAvgPool1d(1) → gom toàn bộ thời gian
    - Flatten: squeeze(-1) → còn lại (batch\_size, 128)
    - Fully Connected: Linear(128 → fc\_out) (mặc định là 1)
  + SEBlock
    - "Squeeze": gom mỗi kênh thành 1 giá trị qua AdaptiveAvgPool1d
    - "Excitation": dùng MLP nhỏ để tính mức quan trọng của mỗi kênh → giống attention theo kênh
    - Cuối cùng: x × attention weight → làm nổi bật kênh quan trọng hơn
* **Khởi tạo mô hình**- Quá trình huấn luyện diễn ra nhanh chóng do lượng dữ liệu thấp nên có thể áp dụng learning\_rate nhỏ, thêm tham số weigh\_decay chống overfitting.

A computer screen shot of white text

AI-generated content may be incorrect.

* **Huấn luyện mô hình:**
  + - Sử dụng batch\_size =64 giúp mô hình học với tốc độ phù hợp.
    - Sử dụng epoch = 300, phù hợp khi learning\_rate nhỏ.
    - Patience=5, tham số giúp early\_stopping xảy ra khi quá trình học không có dấu hiệu cải thiện chất lượng.
* **Kiểm tra các thông số độ đo với tập validation.**
* **Dự đoán trên tập test.**

1. **Recurrent neural network (RNN)**

* **Kiến trúc mạng:**
  + - Đầu vào với kích thước (10,128). Mỗi dòng dữ liệu là một ma trận có kích thước 10x128 tương ứng với 10 frame và 128 đặc trưng.
    - Masking với mask\_value = 0.0
    - SpatialDropout1D với tỉ lệ 0.1
    - Block 1:
      * LSTM (128, return\_sequences = False)
      * Dropout(0.1)
    - Block 2:
      * Dense(32, activation = ‘relu’)
      * Dropout(0.1)
    - Output: Dense(1, activation = ‘sigmoid’)
* **Khởi tạo callback:** 
  + - EarlyStopping với monitor='val\_loss', patience=5, restore\_best\_weights=True.
    - ReduceLROnPlateau với monitor='val\_loss', patience=3, factor=0.5, verbose=1.
* **Huấn luyện mô hình:**
  + - Sử dụng batch\_size =64 giúp mô hình học với tốc độ phù hợp.
    - Sử dụng epoch = 50, đủ với quy mô của mô hình.
    - Callback gồm EarlyStopping và ReduceLROnPlateau đã khai báo ở trên.
* **Kiểm tra các thông số độ đo với tập validation.**
* **Dự đoán trên tập test.**

## Các phương pháp xử lý khác

* **Data Regularization:** Một hiện tượng phổ biến xảy ra khi xây dựng mô hình là việc thường xuyên gặp phải overfitting với hầu hết các cấu hình siêu tham số hay kiến trúc mạng neural với nguyên nhân dễ thấy nhất là việc thiếu dữ liệu. Nhóm áp dụng một số phương pháp data regularization như Data Augmentation, Mixup, Soft Voting,... với mục đích tạo thêm dữ liệu huấn luyện, giúp mô hình tránh bị overfitting.
* **Trích xuất đặc trưng cố định từ audio\_embedding:** Do mỗi audio\_embedding có số lượng frame khác nhau (tức là chiều thứ 2 của embedding không cố định), để chuẩn hóa đầu vào cho mô hình học máy, ta thực hiện việc lấy trung bình theo trục thời gian cho từng audio\_embedding. Cụ thể, với mỗi mẫu âm thanh, ta tính trung bình theo từng cột (tức là trung bình vector theo từng chiều đặc trưng), kết quả là mỗi mẫu sẽ được biểu diễn bởi một vector có chiều cố định.

# PHẦN 4: MODEL EVALUATION

## Các bộ chỉ số đánh giá

* **AUC:**
  + - Ý nghĩa: AUC đo lường khả năng của mô hình trong việc phân biệt giữa hai lớp. AUC càng gần 1 thì mô hình phân biệt càng tốt. AUC = 0.5 tương đương với mô hình ngẫu nhiên.
    - Dữ liệu đầu vào: Nhãn thật (y\_true) và xác suất dự đoán (y\_prob).
    - AUC xét toàn bộ các giá trị ngưỡng từ 0 đến 1, không cố định tại một giá trị nào.
    - Thư viện hỗ trợ:

from sklearn.metrics import roc\_auc\_score auc = roc\_auc\_score(y\_true, y\_prob)

* + Accuracy:
    - Ý nghĩa: Tỷ lệ dự đoán đúng trên tổng số mẫu. Phản ánh mức độ tổng quát mà mô hình dự đoán chính xác.
    - Dữ liệu đầu vào: Nhãn thật (y\_true) và xác suất dự đoán (y\_prob).
    - Cần chọn ngưỡng (ví dụ: 0.5) để chuyển xác suất (y\_prob) thành nhãn (y\_pred).
    - Thư viện hỗ trợ:

from sklearn.metrics import accuracy\_score accuracy = accuracy\_score(y\_true, y\_pred)

* + Precision:
    - Ý nghĩa: Tỷ lệ các mẫu được mô hình dự đoán là dương (positive) mà thực sự là dương. Phản ánh mức độ tin cậy khi mô hình dự đoán 1.
    - Dữ liệu đầu vào: Nhãn thật (y\_true) và xác suất dự đoán (y\_prob).
    - Cần chọn ngưỡng (ví dụ: 0.5) để chuyển xác suất (y\_prob) thành nhãn (y\_pred).
    - Thư viện hỗ trợ:

from sklearn.metrics import precision\_score precision = precision\_score(y\_true, y\_pred)

* + Recall:
    - Ý nghĩa: Tỷ lệ các mẫu thực sự dương được mô hình dự đoán đúng là dương. Quan trọng trong các bài toán nhạy cảm như y tế, an ninh.
    - Dữ liệu đầu vào: Nhãn thật (y\_true) và xác suất dự đoán (y\_prob).
    - Cần chọn ngưỡng (ví dụ: 0.5) để chuyển xác suất (y\_prob) thành nhãn (y\_pred).
    - Thư viện hỗ trợ:

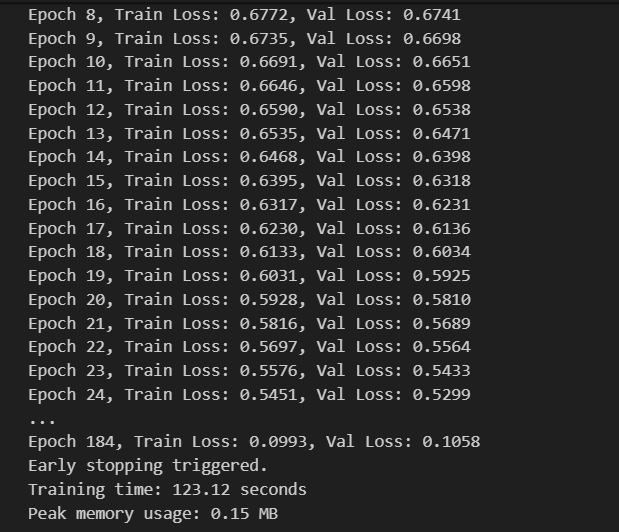
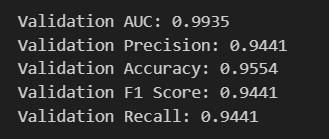
from sklearn.metrics import recall\_score recall = recall\_score(y\_true, y\_pred)

* + F1-score:
    - Ý nghĩa: Là chỉ số cân bằng giữa Precision và Recall. F1-score cao nghĩa là mô hình vừa chính xác vừa phát hiện tốt lớp dương.
    - Dữ liệu đầu vào: Nhãn thật (y\_true) và xác suất dự đoán (y\_prob).
    - Cần chọn ngưỡng (ví dụ: 0.5) để chuyển xác suất (y\_prob) thành nhãn (y\_pred).
    - Thư viện hỗ trợ:

from sklearn.metrics import f1\_score f1 = f1\_score(y\_true, y\_pred)

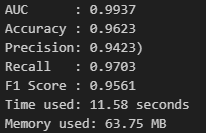
## Đánh giá và cải thiện bộ chỉ số

1. **Transformer**
2. **Convolutional neural network (CNN)**



* **AUC Score = 0.9935:** gần như đạt giá trị tuyệt đối, cho thấy mô hình phân biệt rất tốt giữa các lớp.
* **Accuracy = 0.9441:** độ chính xác cao, phản ánh tỷ lệ dự đoán đúng tổng thể.
* **Precision = 0.9554:** tỷ lệ dương tính dự đoán đúng rất cao, nghĩa là mô hình rất ít dự đoán sai các mẫu tiếng gà.
* **Recall = 0.9441:** mô hình bắt được gần như đầy đủ các trường hợp dương tính (false negative thấp).
* **F1-score = 0.9441:** cân bằng tốt giữa precision và recall.
  + Tổng thể, mô hình có độ chính xác và độ tin cậy cao, có thể triển khai hoặc ứng dụng thực tế.
  + Nhờ các phương pháp chống overfitting, mô hình cũng đạt được kết quả ấn tượng đối với tập test với score lên tới 0.9912.
  + Tuy nhiên mô hình trên cần lượng lớn thời gian để huấn luyện, điều này được giải thích bởi số lượng epoch lớn trong khi learning\_rate nhỏ, mô hình tốn rất lâu để hội tụ.
  + Pooling giúp cho mô hình đơn giản hoá dữ liệu, chỉ tập trung vào những đặc trưng quan trọng và không cần nhớ những giá trị đứng trước nó khiến cho việc học tốn rất ít không gian cho bộ nhớ.

1. **Recurrent neural network (RNN)**



* **AUC Score = 0.9937:** gần như đạt giá trị tuyệt đối, cho thấy mô hình phân biệt rất tốt giữa các lớp.
* **Accuracy = 0.9623:** độ chính xác cao, phản ánh tỷ lệ dự đoán đúng tổng thể.
* **Precision = 0.9423:** tỷ lệ dương tính dự đoán đúng rất cao, nghĩa là mô hình rất ít dự đoán sai các mẫu tiếng gà.
* **Recall = 0.9703:** mô hình bắt được gần như đầy đủ các trường hợp dương tính (false negative thấp).
* **F1-score = 0.9561:** cân bằng tốt giữa precision và recall.
* Thời gian huấn luyện 11~13 giây.
* Bộ nhớ sử dụng là 30~64MB. Vì quá trình học của RNN nhớ thông tin của các bước trước đó nên bộ nhớ sử dụng cao hơn nhiều so với CNN.

# PHẦN 5: ĐÁNH GIÁ PHƯƠNG PHÁP

## Phương pháp tiền xử lý dữ liệu

* + Trích xuất đặc trưng bằng mean:
    - **Ưu điểm:** Đơn giản, nhanh, dễ tính toán. Rút gọn được chiều dữ liệu → giảm kích thước đầu vào cho mô hình.
    - **Nhược điểm:** Thiếu cơ sở lý thuyết làm mất thông tin chi tiết theo từng đặc trưng. Có thể làm mất các mẫu âm thanh đặc trưng quan trọng.
  + Chuẩn hoá dữ liệu:
    - **Ưu điểm:** Giúp đưa các đặc trưng về cùng một thang đo, giúp quá trình lan truyền tiến (forward) và lan truyền ngược (backpropagation) trong mạng MLP trở nên ổn định hơn. Việc này hỗ trợ tăng tốc độ hội tụ, giúp mô hình học hiệu quả hơn và tránh hiện tượng gradient biến mất.
    - **Nhược điểm:** Nhạy cảm với outliers nếu tồn tại nhiều outliers sẽ dẫn đến hiệu suất kém cho mô hình.
  + Mix-up là một phương pháp phổ biến để tránh overfitting nhưng thực tế khi ứng dụng với bộ dữ liệu này lại không mang lại kết quả tốt, điều này có thể do phương pháp tạo thêm dữ liệu mới không phù hợp, pha trộn và sáng tạo nhưng không thể đánh nhãn phù hợp cho dữ liệu.
  + Gaussian noise mang lại kết quả khá tốt khi giúp tăng lượng lớn được dữ liệu thông qua việc làm nhiễu nhẹ nhưng không ảnh hưởng mạnh tới quá trình học.

## Phương pháp lựa chọn và xây dựng mô hình

1. **Transfomer**
   * Transformer ra đời vào năm 2017, trong bài báo Attention is All you Need, là một kiến trúc mạng nơ-ron tiên tiến dùng trong xử lý chuỗi, đặc biệt nổi bật trong NLP.
     + Nó không dùng RNN hay CNN mà dựa vào cơ chế Attention để học quan hệ giữa các phần tử trong chuỗi.
     + Có thể xử lý song song toàn bộ chuỗi, giúp huấn luyện nhanh hơn so với RNN.
     + Cấu trúc gồm 2 phần: Encoder (mã hóa đầu vào) và Decoder (tạo đầu ra).
     + Transformer là bước đột phát, là nền tảng cho các mô hình mạnh như BERT, ROBERTA, GPT, T5,.....
   * Kiến trúc Transformer dùng để:
     + Encoder (N lớp Encoder giống nhau)- Mỗi lớp Encoder gồm:
       - Multi-Head Self-Attention Layer → Cho phép mỗi từ "nhìn" toàn bộ các từ khác trong câu để học quan hệ.
       - Add & Norm → Cộng residual (shortcut) và chuẩn hóa LayerNormalization.
       - Feed Forward Network (FFN) → 2 lớp Dense: nonlinear transformation.
       - Add & Norm → Cộng residual + LayerNorm lần nữa. → Encoder đầu vào: embedding + positional encoding.
     + Decoder (N lớp Decoder giống nhau) - Mỗi lớp Decoder gồm:
       - Masked Multi-Head Self-Attention → Attention nhìn các từ trước đó (dự đoán tuần tự).
       - Add & Norm
       - Multi-Head Attention (Encoder-Decoder Attention) → Attention nhìn qua kết quả của Encoder để liên kết thông tin đầu vào.
       - Add & Norm
       - Feed Forward Network (FFN)
       - Add & Norm
     + Output Layer : Dense layer + Softmax để sinh xác suất cho từng từ (trong Decoder).
2. **Convolutional neural network (CNN)**
   * + **CNN** (Convolutional Neural Network – Mạng nơ-ron tích chập) là một kiến trúc mạng nơ-ron chuyên xử lý dữ liệu có cấu trúc không gian, phổ biến nhất là hình ảnh và video. CNN mô phỏng cách thị giác con người hoạt động – từ nhận diện các cạnh, góc, rồi đến hình khối, và cuối cùng là toàn bộ vật thể.
     + **Kiến trúc của CNN bao gồm:** 
       - Convolution Layer (Lớp tích chập): Sử dụng các **kernel (bộ lọc)** để quét qua ảnh và **trích xuất đặc trưng cục bộ** như đường thẳng, họa tiết, màu sắc...
       - **Activation Layer (thường là ReLU):** giúp mạng học được các mối quan hệ phức tạp thông qua phi tuyến.
       - **Pooling Layer (Lớp giảm kích thước):** Giảm số lượng tính toán và trích xuất đặc trưng quan trọng (ví dụ: MaxPooling).
       - **Fully Connected Layer (Lớp kết nối đầy đủ):** giống như MLP.
   * Thực tế các CNN thường được sử dụng rộng rãi hơn đối với các bài toán liên quan đến ảnh do đặc tính không gian của ảnh, cần trích xuất đặc trưng theo nhiều chiều. Tuy nhiên trong trường hợp này, âm thanh đã được embedding bằng Vggish, mỗi giây (~0.96s) lúc này được chuyển thành vector 128 chiều, khi sắp xếp các âm thanh này thành một khối 10\*128, ta cũng có thể có được một ma trận không gian. Điểm đặc biệt là ta có thể có thể thực hiện phép tích chập theo chiều dọc, tức là chiều thời gian.
   * Kiến trúc cơ bản của mô hình sẽ dần đi qua các lớp tích chập 1D, mục đích là tích chập theo chiều thời gian.
   * Do số lượng phần tử trong mỗi mẫu lớn lên tới 10\*128 mẫu nên cần chọn ra những mẫu quan trọng, được xác định thông qua SEBlock, cho phép mô hình xác định các kênh quan trọng.
   * Dropout, pooling, FC giúp giảm overfitting và ra quyết định cuối.
   * Quá trình lựa chọn bộ tham số:
     + - Bước 1: Chạy vòng lặp với learning\_rate lớn (0.01), epoch thấp (20) cho ra kết quả underfitting.
       - Bước 2: Thay đổi số lượng batch\_size để cải tiến quá trình học, đối với CNN batch\_size =64 là tốt nhất.
       - Bước 3: Thực nghiệm cho thấy mô hình đang cho kết quả tốt nhất với learning\_rate bé (0.00001), điều này có thể tạm chấp nhận vì bộ dữ liệu cũng chỉ có kích thước nhỏ.
       - Bước 4: Weight\_decay = 0.001 là phù hợp để cho mô hình không bị overfitting lẫn underifitting.
       - Bước 5: Kết quả chạy nhiều lần cho thấy kết quả không lệch quá nhiều, cho thấy bộ tham số này là phù hợp, không dựa trên may rủi.
3. **Recurrent neural network (RNN)**
   * **RNN (Recurrent Neural Network - Mạng nơ-ron hồi quy)** là một loại mạng nơ-ron nhân tạo được thiết kế để xử lý dữ liệu tuần tự. Khác với mạng nơ-ron thường (Feedforward Neural Network), RNN có khả năng ghi nhớ thông tin từ các bước trước đó trong chuỗi, nhờ vào cơ chế trạng thái ẩn (hidden state).
   * **Kiến trúc của RNN:**
     + - **Input Layer (Lớp đầu vào):** Nhận một chuỗi đầu vào dạng nhiều bước thời gian (timesteps).
       - **RNN Cell (Khối hồi tiếp**): Thành phần cốt lõi, xử lý tuần tự từng bước đầu vào. Tại thời điểm t, trạng thái ẩn hₜ được tính dựa trên đầu vào xₜ và trạng thái trước đó hₜ₋₁. RNN có thể "nhớ" hoặc "quên" thông tin tùy thuộc vào loại cell được sử dụng (SimpleRNN, LSTM, GRU...).
       - **Activation Function (thường là tanh hoặc ReLU**): Áp dụng phi tuyến để mô hình hóa các mối quan hệ phức tạp trong chuỗi.
       - **Output Layer (Lớp đầu ra):** Sau khi xử lý toàn bộ chuỗi, trạng thái ẩn cuối cùng hoặc toàn bộ chuỗi trạng thái được đưa vào lớp Dense để tạo ra đầu ra như phân loại, dự đoán,...
     + RNN được sử dụng rộng rãi trong các bài toán như: nhận dạng tiếng nói, dịch máy, phân tích cảm xúc văn bản, dự báo chuỗi thời gian (stock, IoT...). Trong trường hợp này, chuỗi audio\_embedding đã được trích xuất từ mô hình VGGish (1 vector 128 chiều mỗi ~0.96 giây). Khi xếp 10 vector lại, ta có ma trận (10, 128) – về bản chất là chuỗi âm thanh, nên RNN sẽ học mối quan hệ theo chiều thời gian giữa các đoạn này.
     + Các biến thể phổ biến:
       - **SimpleRNN:** dạng cơ bản nhất, dễ bị quên thông tin cũ (vanishing gradient).
       - **LSTM (Long Short-Term Memory):** có cơ chế "quên" và "nhớ" tốt hơn, học được phụ thuộc dài hạn.
       - **GRU (Gated Recurrent Unit):** nhẹ hơn LSTM, hiệu quả tương đương trong nhiều bài toán.
       - **Bidirectional RNN:** học ngữ cảnh cả trước và sau mỗi bước – đặc biệt hữu ích trong xử lý ngôn ngữ tự nhiên hoặc chuỗi âm thanh.
     + Dùng EarlyStopping và ReduceLROnPlateau để thực hiện huấn luyện.

# PHẦN 6: TÀI LIỆU THAM KHẢO

## Tài liệu nhóm

Github: [Dzivilord/Machine\_Learning\_Project](https://github.com/Dzivilord/Machine_Learning_Project)

## Tài liệu tham khảo

1. Patrice Guyot, ‘SIMPLECNNANDVGGISHMODEL FORHIGH-LEVELSOUNDCATEGORIZATION WITHINTHEMAKINGSENSEOFSOUNDSCHALLENGE’ [Trực tuyến]. Địa chỉ: [Patrice\_Guyot\_3.pdf](https://cvssp.org/projects/making_sense_of_sounds/site/assets/challenge_abstracts_and_figures/Patrice_Guyot_3.pdf)
2. Scott Zhu, Francois Chollet, ‘Working with RNNs’ [Trực tuyến]. Đường dẫn: [Working with RNNs](https://www.tensorflow.org/guide/keras/working_with_rnns)
3. [Danijar Hafner,](https://danijar.com/) ‘Tips for Training Recurrent Neural Networks’ [Trực tuyến]. Đường dẫn: [Tips for Training Recurrent Neural Networks.](https://danijar.com/tips-for-training-recurrent-neural-networks/)
4. <https://www.sciencedirect.com/science/article/pii/S1051200424005803>
5. Advanced Audio Processing and Recognition with Transformer

Link: <https://www.geeksforgeeks.org/audio-processing-with-transformer/>

1. Khalid Zaman a, Kai Li a, Melike Sah b, Cem Direkoglu c, Shogo Okada a, Masashi Unoki a , ‘Transformers and audio detection tasks: An overview’. Đường dẫn: [Transformers and audio detection tasks: An overview - ScienceDirect](https://www.sciencedirect.com/science/article/pii/S1051200424005803)