ĐẠI HỌC QUỐC GIA TPHCM

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

Phân tích bài báo

Đề tài: AST: Audio Spectrogram Transformer

Môn học: Nhập môn học máy

Sinh viên thực hiện:

Tô Hữu Danh (23127336)

Nguyễn Đăng Hưng (23127050)

Lê Phú Cường (23127164)

Nguyễn Bá Đăng Khoa (23127392)

Giáo viên hướng dẫn:

Thầy Võ Nhật Tân

Ngày 30 tháng 11 năm 2025

Mục lục

Danh sách bảng

Danh sách hình vẽ

1 Bài toán giải quyết. Đầu vào, đầu ra

1.1 Mục tiêu chính của bài nghiên cứu

Trong khoảng 1 thập kỷ trước 2021, mô hình CNN được coi là tiêu chuẩn cho việc phân loại âm thanh dựa vào ý tưởng âm thanh thành hình ảnh (phổ âm) rồi sử dụng CNN để xử lý. Tuy nhiên bản thân mô hình CNN tồn tại một số điểm yếu nhất định, đặc biệt là khả năng nhận biệt long-range global context kém với ít lớp tích chập khiến cho kết quả vẫn còn hạn chế.

Khi mô hình Transformer ra đời, một số bài nghiên cứu đã thử nghiệm việc tạo ra mô hình lai giữa CNN và một lớp Attention của mô hình Transformer để khắc phục nhược điểm này. Kết quả là mô hình lai CNN-Attention này đạt được kết quả tốt nhất từ trước tới giờ.

Bài toán đặt ra của bài nghiên cứu: Liệu cơ chế tích chập của CNN có còn cần thiết hay chỉ cần dùng mô hình Transformer là đủ để tạo ra mô hình state-of-the-art trong việc phân loại âm thanh. Kết quả cho ra của bài nghiên cứu này là mô hình Audio Spectrogram Transformer (AST) đạt được kết quả tốt nhất.

Input và Output: Mô hình AST nhận vào âm thanh đã được xử lí thành các ảnh Log-Mel Spectrogram và trả về một vector chứa xác suất mà âm thanh có các loại tiếng (động vật, công trình, đường phố,...) khác nhau. Chi tiết về cách hoạt động của mô hình sẽ được nói ở mục sau.

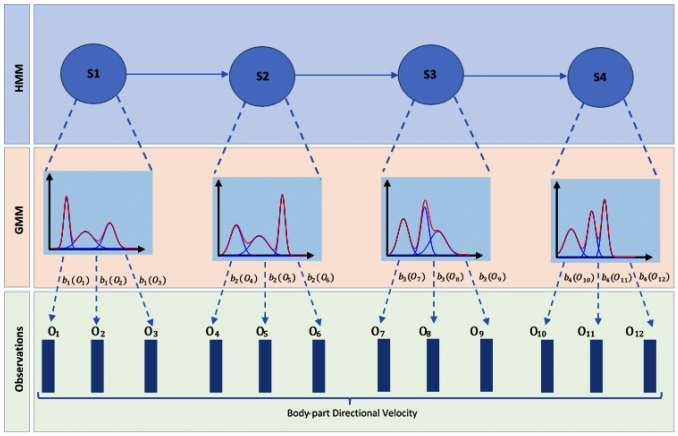
2 Công trình liên quan, quá trình phát triển

2.1 Hidden Markov Models và Gaussian Mixture Models [1]

Là phương pháp phổ biến nhất, mạnh nhất những năm 1980 đến tận trước 2010. Người ta tiền xử lý dữ liệu waveform của audio thủ công bằng các thuật toán như Fourier Transform, Cepstral Analysis để tạo ra các thuộc tính như MFCC (*Mel-Frequency Cepstral Coefficient*).

Về MFCC, đây là một loại đặc trưng âm thanh được sử dụng làm đầu vào cho mô hình GMM/HMM, nó chuyển đổi dữ liệu waveform thành một tập hợp các con số (thường là 12-40 hệ số) mà máy tính có thể phân tích, nhưng được lọc để gắn với cách tai người cảm nhận âm thanh hơn. Quá trình gồm tách khung, biến đổi Fourier (để xem tần số), lọc bằng bộ lọc Mel (làm nổi bật các tần số thấp mà tai người nhạy cảm), biến đổi (logarit, sau đó biến đổi cosin rời rạc (DCT)) sau cùng tạo ra các hệ số MFCC.

GMM (*Gaussian Mixture Model*) là mô hình dùng để gán một xác suất cho một âm thanh ví dụ một đoạn âm thanh nghe giống "A" hay "B". HMM chia đoạn âm thanh đó thành nhiều state (số lượng state do người cài đặt), ứng với mỗi state GMM sẽ có một hàm b tương ứng để tính toán xác suất mà một vector MFCC thuộc về state đó (ví dụ b1(O1) là xác suất mà vector O1 thuộc về s1). HMM (*Hidden Markov Model*) là thành phần gom các vector về các state để tổ chức trình tự cho chúng bằng cách tính toán xác suất chuyển đổi từ trạng thái này sang trạng thái khác. Sau đó HMM trả về một bộ xác suất để giúp GMM đoán được chính xác âm thanh kế tiếp. Ví dụ kế âm "M" thường sẽ là âm "ơ", "a", "e" chứ không thể là âm "n", "k", "m".



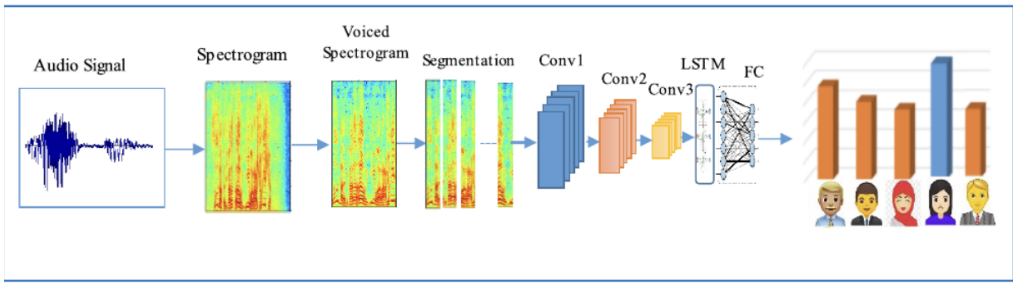
Hình 1: Ví dụ minh hoạ cho GMM-HMM. Bài toán nhận diện hành động của con người. [2]

2.2 Mô hình Hybrid LST-CNN Attention

Nền tảng của CNN (*Convolutional Neural Network*) xuất hiện vào năm 1989 trong bài báo [3], tại đây nhóm tác giả đã huấn luyện thành công các lớp tích chập đầu tiên bằng thuật toán lan truyền ngược, dùng cho bài toán nhận diện chữ viết. Đến năm 1988, kiến trúc LeNet-5 được hoàn thiện và trở thành tiêu chuẩn cho CNN vào thời điểm đó.

Đến năm 2012, AlexNet ra đời và lần đầu tiên triển khai ý tưởng biến âm thanh thành hình ảnh thông qua kỹ thuật Spetrogram rồi sử dụng CNN để xử lý. Mô hình AlexNet với kiến trúc sâu hơn hay nhiều lớp tích chập hơn, sử dụng hàm kích hoạt ReLU thay cho các hàm kích hoạt phổ biến, đồng thời kết hợp thêm cơ chế Dropout để chống overfitting. Từ đó AlexNet cải thiện được đáng kể và đạt tỉ lệ lỗi cực thấp 15.3% trong cuộc thi ILSVRC (*ImageNet Large Scale Visual Recognition Challenge*).

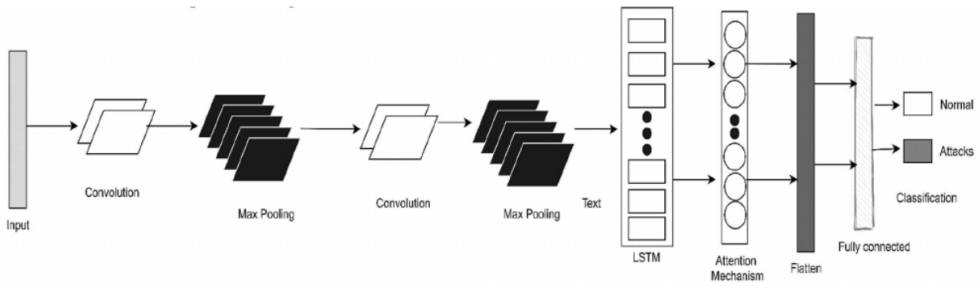
Sau thành công của AlexNet, có rất nhiều nghiên cứu về việc ứng dụng CNN trong bài toán liên quan đến xử lý âm thanh bằng Spectrogram đã được tiến hành và nổi bật nhất phải kể đến CRNN (*Convolutional Recurrent Neural Network*). Về cơ bản, mô hình CRNN chỉ đơn giản thêm một lớp RNN (*Recurrent Neural Network*) vào sau các lớp CNN để có thể xử lý được chuỗi thời gian, quản lý thứ tự của các token - điều vốn rất quan trọng với âm thanh nhưng các bài toán xử lý hình ảnh thuần tuý thường xem nhẹ. RNN được sử dụng nhiều nhất là biến thể LSTM (*Long Short Term Memory*). Vì có LSTM nên mô hình này có thể hiểu được thứ tự và ngữ cảnh của đoạn âm thanh, từ đó có thể tăng được hiệu suất. Tuy nhiên, LSTM cũng chỉ có bộ nhớ giới hạn nên mô hình cơ bản vẫn không nắm được toàn bộ ngữ cảnh của đoạn âm thanh.



Hình 2: Ví dụ minh hoạ cho mô hình CRNN. [4]

Sau cùng, cơ chế Attention đã được thêm vào mô hình CRNN để tạo thành mô hình Hybrid LST-CNN Attention. Cơ chế Attention giúp mô hình có thể tập trung vào những phần quan trọng của

đoạn âm thanh, bỏ qua những phần không cần thiết. Nhờ vậy mà mô hình có thể nắm bắt được ngữ cảnh toàn diện hơn, từ đó cải thiện hiệu suất nhận diện.



Hình 3: Ví dụ minh hoạ cho mô hình LSTM-CNN Attention. [5]

2.3 Mô hình Vision Transformer [6]

Áp dụng Transformer (vốn được xài cho xử lý ngôn ngữ) trực tiếp lên hình ảnh với ít sự thay đổi nhất có thể. Xử lý hình ảnh thành chuỗi các patches, từ đó biến một hình ảnh thành chuỗi các visual words

Quá trình xử lý:

- Nhận vào một hình ảnh, cắt nó thành tập hợp các ô có kích thước cố định 16x16 pixels.
- Duỗi nó thành vector 1D (16x16x3 (kênh màu) = 768).
- Vector đó được nhân với một ma trận trọng số để chuyển thành các dense vector có kích thước cố định (768) (gọi là các patch embeddings). Những embeddings này biểu diễn cho các visual words.
- Mượn ý tưởng từ BERT language model, tác giả không nghĩ việc lấy trung bình các vector output là một cách tốt để nhận về phân loại cho cả image. Thay vào đó, họ xài một vector CLS để prepend vào phần đầu của patch sequence.
- Sau đó, cộng dãy các vector (gồm cả CLS) với vector positional để nhận được đầu vào cuối cùng cho lớp encoder. Vector CLS tương tác với toàn bộ vector còn lại theo cơ chế self-attention của transformer. Xài đầu ra tương ứng của token này (qua lớp encoder) để đưa qua lớp phân loại cuối cùng.

Với cấu trúc như vậy, ViT giải quyết được các vấn đề của CNN gồm:

- Inductive Bias: ViT model xử lý tất cả các patch như các thực thể độc lập nhau, không có mối liên hệ đến các patch khác từ đó giảm được inductive bias so với CNN.
- Receptive field: CNN chỉ xử lý được cục bộ vì các lớp chỉ nhìn thấy một phần nhỏ cục bộ của tấm ảnh, còn ViT xử lý toàn cục vì mỗi patch được liên kết lại với nhau thông qua các lớp đầu.
- Data Hunger: CNN chỉ hoạt động tốt với tập dữ liệu nhỏ vì cách học của nó đã được xây dựng trước, còn cách học của ViT sẽ được xây dựng khi nó phân tích dữ liệu nên nó sẽ học được trên tập dữ liệu lớn hơn.

3 Mô hình đề xuất của tác giả

Mô hình được đề xuất kế thừa kiến trúc *Vision Transformer (ViT)* đã được pretrain trên ImageNet, sau đó được điều chỉnh để phù hợp với bài toán phân loại âm thanh. Ý tưởng chính là chuyển đổi tín hiệu âm thanh từ miền thời gian sang miền tần số để tạo ra biểu diễn dạng ảnh (spectrogram), từ đó tận dụng khả năng học đặc trưng mạnh mẽ của Transformer.

3.1 Kiến trúc tổng thể

Audio Spectrogram Transformer (AST) được xây dựng theo kiến trúc **pure Transformer**, hoàn toàn không sử dụng bất kỳ lớp Convolutional Neural Network (CNN) nào. Mô hình khai thác cơ chế *multi-head self-attention* để học các quan hệ phụ thuộc dài hạn trong spectrogram.

3.2 Biểu diễn đầu vào

Đầu vào của mô hình là ảnh **Log-Mel Spectrogram** được trích xuất từ tín hiệu waveform. Spectrogram được tính theo quy trình:

- Windowing: Hamming window 25 ms,
- Hop size: 10 ms,
- Số lượng filterbank Mel: 128.

Với đoạn âm thanh dài t giây, spectrogram thu được có kích thước:

$$128 \times 100t.$$

Spectrogram sau đó được chuẩn hóa và điều chỉnh kích thước để phù hợp với đầu vào ViT.

3.3 Patching

Spectrogram được chia thành các patch kích thước 16×16. AST sử dụng **stride = 10**, dẫn đến hiện tượng overlap 6 điểm theo cả trục tần số và trục thời gian.
Số lượng patch được tính bởi:

$$N = 12 \times \left\lceil \frac{100t - 16}{10} \right\rceil.$$

Trong đó 12 là số patch theo trục tần số. Mỗi patch được flatten để tạo thành vector 1 chiều.

3.4 Patch Embedding

Mỗi patch sau khi flatten được đưa qua một lớp Linear Projection để thu được embedding vector kích thước 768 (embedding dimension của ViT-Base).
Vì ViT gốc được pretrain trên ảnh RGB (3-channel), còn spectrogram chỉ có 1 channel, trọng số đầu vào được khởi tạo bằng cách **average-weighting** từ 3 channel của ViT sang 1 channel.

3.5 Positional Embedding

Để bảo toàn thông tin vị trí trên không gian thời gian–tần số, mỗi patch embedding được cộng thêm một vector positional embedding (learnable).
Do ViT gốc hoạt động trên ảnh vuông (ví dụ 384×384 với positional grid 24×24), còn spectrogram lại có dạng hình chữ nhật, nhóm tác giả đề xuất phương pháp **"crop-and-bilinear-interpolate"** đối với positional embedding:

- Crop số hàng từ 24 xuống 12 để phù hợp chiều frequency,
- Bilinear interpolate số cột từ 24 lên chiều thời gian tương ứng (ví dụ 100).

Cách này giúp positional embedding trở nên tương thích hoàn toàn với kích thước spectrogram.

3.6 Transformer Encoder

Chuỗi token đầu vào có dạng:

$$[\text{CLS}], x_1, x_2, \ldots, x_N.$$

Bộ *Transformer Encoder* trong AST được cấu hình như sau:

- 12 Transformer layers,
- 12 attention heads,

- Hidden size: 768.

Tất cả trọng số đều được khởi tạo từ ViT pretrained trên ImageNet để tận dụng khả năng nhận diện cấu trúc hình học đã học trước.

3.7 Classification Head

Sau Transformer Encoder, embedding của token `[CLS]` được dùng làm global representation của toàn bộ tín hiệu. Token này được truyền qua một lớp Linear mới (thay thế classification head của ViT) và một hàm kích hoạt Sigmoid để thực hiện phân loại đa nhãn.
Kết quả cuối cùng là vector xác suất tương ứng với từng loại âm thanh như tiếng chó, mèo, gà, chim, chuột, ...

4 Cài đặt thực nghiệm của tác giả

4.1 Thí nghiệm trên AudioSet

4.1.1 Bộ dữ liệu AudioSet

- AudioSet gồm hơn 2 triệu đoạn âm thanh, mỗi đoạn dài 10 giây, cắt ra từ video youtube và được gán nhãn theo 527 loại âm thanh.
- Bộ dữ liệu được chia thành:
  - Balanced training set: khoảng 22K mẫu
  - Full training set: khoảng 2M mẫu
  - Evaluation set: khoảng 20K mẫu
- Tác giả sử dụng mean Average Precision(mAP) làm chỉ số chính để đánh giá chất lượng mô hình trên AudioSet

4.1.2 Thiết lập thử nghiệm

Trong phần thực nghiệm với bộ dữ liệu AudioSet [7, 8], tác giả tuân theo đúng quy trình huấn luyện được mô tả trong bài báo gốc. AudioSet là tập hơn 2 triệu đoạn âm thanh dài 10 giây được cắt từ các video YouTube, gắn nhãn theo tập 527 loại âm thanh. Các tập con dùng trong thí nghiệm gồm:

- Tập huấn luyện cân bằng (balanced training set): khoảng 22 000 mẫu.
- Tập huấn luyện đầy đủ (full training set): khoảng 2 000 000 mẫu.
- Tập đánh giá (evaluation set): khoảng 20 000 mẫu.

Cấu hình huấn luyện chung.

- Mô hình được khởi tạo từ trọng số pretrained trên ImageNet (theo Mục 2.2 của bài báo gốc).
- Batch size bằng 12.
- Tối ưu bằng thuật toán Adam [9] với hàm mất mát binary cross-entropy cho bài toán gán nhãn đa nhãn (multi-label).

- Pipeline huấn luyện:
  - Với thí nghiệm trên *full set*: dùng balanced sampling để cân bằng phân bố nhãn.
  - Data augmentation:
    - Mixup [10] với tỉ lệ trộn (mixup ratio) bằng 0.5.
    - Spectrogram masking [11] với độ dài mặt nạ theo thời gian tối đa 192 frame và theo tần số tối đa 48 bin.
  - Model aggregation:
    - Weight averaging [12].
    - Ensemble nhiều mô hình [13].
- Thước đo chính dùng để đánh giá là mean Average Precision (mAP) trên tập đánh giá AudioSet.

Lịch huấn luyện cho từng thiết lập.

- **Balanced set:**
  - Mô hình được huấn luyện trong 25 epoch.
  - Learning rate khởi tạo là 5×10^{-5}.
  - Learning rate được giảm một nửa sau mỗi 5 epoch kể từ sau epoch thứ 10 (tức là giảm tại các mốc 15, 20, ...).
- **Full set:**
  - Mô hình được huấn luyện trong 5 epoch.
  - Learning rate khởi tạo là 1×10^{-5}.
  - Learning rate được giảm một nửa sau mỗi epoch kể từ sau epoch thứ 2 (tức là tại các epoch 3, 4, 5).

4.1.3 Kết quả thí nghiệm

Với balanced AudioSet (chỉ khoảng 1% full train), AST vẫn giữ được ưu thế rõ rệt. Một mô hình AST đơn lẻ với weight averaging đã đạt khoảng 0.347 mAP, vượt mức của các mô hình CNN+attention

tốt nhất trước đó. Khi dùng ensemble-S và ensemble-M, kết quả tiếp tục tăng lên khoảng 0.363 và 0.378. Điều này cho thấy ngay cả khi dữ liệu huấn luyện rất ít, kiến trúc Transformer của AST vẫn học đặc trưng âm thanh tốt hơn và tổng quát hóa tốt hơn so với các mô hình CNN-attention hybrid.

4.1.4 Ảnh hưởng của ImageNet pretraining

- Thử nghiệm ablation cũng cho thấy vai trò gần như "bắt buộc" của ImageNet pretraining: nếu huấn luyện AST từ đầu, mAP trên balanced và full set chỉ quanh 0.148 và 0.366, trong khi khi khởi tạo từ trọng số ImageNet thì nhảy vọt lên khoảng 0.347 và 0.459. Khoảng chênh lệch này cho thấy ImageNet pretraining đóng vai trò rất quan trọng trong việc giúp AST đạt hiệu năng cao trên AudioSet.

- Khi thay đổi loại backbone để khởi tạo (ViT-Base, ViT-Large, DeiT có/không distillation), kết quả cho thấy việc chỉ tăng kích thước mô hình (ViT-Large 307M tham số) không quan trọng bằng chất lượng pretraining. Phiên bản DeiT có distillation, với số tham số tương đương ViT-Base (87M), lại cho mAP cao nhất trên balanced AudioSet (0.347). Điều này gợi ý rằng chiến lược pretrain thông minh (distillation) còn quan trọng hơn việc "phóng to" mạng, và phù hợp hơn làm điểm xuất phát cho AST trên tác vụ audio tagging.

4.1.5 Ảnh hưởng của Positional Embedding Adaptation

Tác giả sử dụng cách cut và nội suy song tuyến (bi-linear interpolation) cho positional embedding, rồi so sánh với trường hợp mô hình AST đã được pretrain nhưng positional embedding được khởi tạo lại ngẫu nhiên. Kết quả cho thấy khi reinitialize positional embedding, mAP trên balanced set còn 0.305; trong khi đó, nếu dùng nearest-neighbor interpolation thì mAP đạt 0.346, và với bi-linear interpolation (thiết lập được sử dụng trong mô hình chính) thì mAP đạt 0.347. Từ các kết quả này, tác giả ghi nhận rằng việc khởi tạo lại positional embedding không làm mô hình pretrain hoàn toàn bị phá vỡ, vì vẫn tốt hơn mô hình bị khởi tạo ngẫu nhiên toàn bộ(mAP = 0.148), nhưng vẫn gây ra suy giảm hiệu năng đáng kể so với cách thích ứng được đề xuất; đồng thời, bi-linear interpolation và nearest-neighbor interpolation không tạo ra khác biệt lớn về hiệu năng.

4.1.6 Ảnh hưởng của Patch Split Overlap

Tác giả so sánh hiệu năng của các mô hình AST được huấn luyện với các mức patch split overlap khác nhau. Kết quả cho thấy khi không dùng overlap, mô hình có 512 patch và đạt mAP 0.336 trên balanced set, 0.451 trên full set; với overlap-2 (657 patch) mAP tăng lên 0.342 / 0.456; với overlap-4 (850 patch) là 0.344 / 0.455; và với overlap-6 (1212 patch, cấu hình được sử dụng trong mô hình chính) là 0.347 / 0.459. Như vậy, hiệu năng có xu hướng cải thiện khi tăng độ chồng lấn patch cho cả balanced và full set, nhưng đồng thời số patch lớn hơn cũng làm chuỗi đầu vào cho Transformer dài hơn, dẫn đến chi phí tính toán tăng theo bậc hai. Tác giả cũng nhấn mạnh rằng ngay cả trong trường hợp không dùng patch split overlap, AST vẫn vượt hệ thống tốt nhất trước đó.

4.1.7 Ảnh hưởng của hình dạng và kích thước Patch

Tác giả so sánh cách chia spectrogram thành patch khác nhau. Spectrogram được chia thành các patch vuông 16×16, nên chuỗi patch đầu vào Transformer không theo thứ tự thời gian, và positional embedding được kỳ vọng sẽ mã hóa được thông tin không gian 2D. Tác giả so sánh với cách chia patch chữ nhật 128×2, cắt theo đúng thứ tự thời gian; khi giữ cùng diện tích patch = 256, mô hình dùng patch 128×2 (512 patch, không pretrain, mAP = 0.154) cho kết quả tốt hơn patch 16×16 (512 patch, không pretrain, mAP = 0.143). Tuy vậy, do không có mô hình ImageNet pretrained nào dùng patch 128×2, trong khi cấu hình 16×16 có thể tận dụng pretraining (mAP = 0.336), nên 16×16 vẫn là lựa chọn tối ưu hiện tại. Ngoài ra, so với patch 32×32 (128 patch, mAP = 0.139 khi không pretrain), các kết quả cho thấy patch nhỏ hơn thường cho hiệu năng tốt hơn.

4.2 Thí nghiệm trên ESC-50

4.2.1 Bộ dữ liệu ESC-50

Bộ dữ liệu ESC–50 [14] gồm 2 000 đoạn âm thanh môi trường dài 5 giây, được tổ chức thành 50 lớp.

4.2.2 Thiết lập thí nghiệm cho ESC-50

Tác giả so sánh AST với các mô hình SOTA trong hai kịch bản:

- **AST-S**: mô hình AST chỉ sử dụng trọng số pretrained trên ImageNet.

- **AST-P**: mô hình AST sử dụng trọng số pretrained trên ImageNet và AudioSet.

Thiết lập huấn luyện:

- Data augmentation: frequency/time masking [11].
- Batch size: 48.
- Tối ưu: Adam [9].
- Số epoch: 20.
- Learning rate:
  - AST-S: learning rate khởi tạo 1×10^{-4}.
  - AST-P: learning rate khởi tạo 1×10^{-5}.
  - Cả hai: giảm learning rate với hệ số 0.85 sau mỗi epoch kể từ sau epoch thứ 5.
- Đánh giá theo chuẩn 5-fold cross-validation; mỗi thí nghiệm được lặp lại 3 lần và báo cáo giá trị trung bình và độ lệch chuẩn của accuracy.

4.2.3 Kết quả thí nghiệm

AST-S đạt 88.7 ± 0.7% và AST-P đạt 95.6 ± 0.4%, đều vượt các mốc SOTA tương ứng. Tác giả cũng lưu ý rằng mỗi fold chỉ có khoảng 1.600 mẫu huấn luyện nhưng AST vẫn hoạt động tốt ngay cả khi không dùng tiền huấn luyện AudioSet.

4.3 Thí nghiệm trên SpeechCommands

4.3.1 Bộ dữ liệu SpeechCommands

SpeechCommands V2 gồm 105.829 đoạn ghi âm dài 1 giây của 35 lệnh thoại, được chia thành tập huấn luyện, tập validation và tập kiểm tra

4.3.2 Thiết lập thử nghiệm

Tác giả cũng đánh giá hai biến thể: AST-S (pretrained ImageNet) và AST-P (pretrained ImageNet + AudioSet). Thiết lập huấn luyện:

- Nhiệm vụ: phân loại 35 lớp lệnh nói.
- Data augmentation: frequency & time masking [11], random noise và mixup [10].
- Batch size: 128.
- Tối ưu: Adam [9].
- Số epoch tối đa: 20.
- Learning rate:
  - Learning rate khởi tạo 2.5×10^{-4}.
  - Giảm learning rate với hệ số 0.85 sau mỗi epoch kể từ sau epoch thứ 5.
- Mô hình tốt nhất được chọn dựa trên tập validation, sau đó báo cáo accuracy trên tập test.
- Mỗi cấu hình được lặp lại 3 lần và báo cáo trung bình và độ lệch chuẩn của accuracy.

4.3.3 Kết quả thí nghiệm

AST-S đạt $98.11 \pm 0.05\%$, vượt cả mô hình SOTA không pretrain thêm và mô hình CNN được tiền huấn luyện với 200M audio; AST-P đạt $97.88 \pm 0.03\%$, và tác giả kết luận rằng tiền huấn luyện thêm với AudioSet là không cần thiết cho bài toán phân loại lệnh thoại này.

4.4 Nhận xét chung

Tác giả tổng kết rằng, dù độ dài đầu vào khác nhau (1 giây cho SpeechCommands, 5 giây cho ESC-50, 10 giây cho AudioSet) và nội dung trải từ speech đến non-speech, chúng vẫn dùng một kiến trúc AST cố định cho cả ba bộ dữ liệu và đều đạt kết quả SOTA, cho thấy tiềm năng sử dụng AST như một bộ phân loại âm thanh tổng quát.

5 Cài đặt thực nghiệm của nhóm

5.1 Dataset

Nhóm chúng em dự kiến sử dụng bộ dataset **UrbanSound8K** để thực hiện train và test mô hình AST. Một số đặc điểm của dataset:

- **Số lượng:** 8732 file âm thanh .wav
- **Độ dài:** Tối đa 4 giây, có thể ngắn hơn
- **Phân loại** thành 10 lớp âm thanh đô thị: Tiếng máy lạnh, Còi xe, Trẻ em chơi đùa, Chó sủa, Tiếng khoan, Động cơ nổ máy, Tiếng súng, Tiếng búa máy, Còi hụ, Nhạc đường phố.
- **Cấu trúc:** Được chia thành 10 folds để train, validate và test.

5.2 Kế hoạch thực nghiệm

Nhóm sẽ sử dụng 2 model được cung cấp trong source là AST-S (mô hình AST với pretrain trên ImageNet) và AST-P (AST pretrain trên ImageNet và AudioSet) để train và đánh giá. Nhóm dự định xử lý dữ liệu từ bộ UrbanSound8K bằng cách:

- Resampling tất cả các file âm thanh về 16kHz, đây là sample mẫu được dùng trong các phần thực nghiệmc của bài báo.
- Xử lí độ dài: AST có cơ chế interpolation để tiếp nhận độ dài âm thanh khác nhau, tuy nhiên để cho ra kết qua tốt nhất, nhóm có thể điều chỉnh độ dài của các file âm thanh sang cùng mức 10s bằng cách cho lặp lại hoặc để padding cho phần còn thiếu.
- Chuyển thành Spectrogram: Các file âm thanh sau khi xử lý sample và độ dài sẽ được đổi thành log-Mel Spectrogram để train và test Model.

Sau khi thực hiện preprocess dữ liệu, nhóm sẽ tiến hành train và đánh giá. Phương pháp được dùng là 10-fold cross validation (train trên fold 1 - 9, test trên fold 10, lặp lại tới khi test đủ 10 fold), các hyperparameter sẽ được dùng giống như cách tác giả dùng trong bài báo với tập ESC-50:

- **Optimizer**: Adam

- **Learning rate:** 1e-5
- **Epochs:** 10 - 20 vòng
- **Batch size:** 24 - 48
- **Augmentation:** Bật SpecAugment (Che thời gian/tần số) để chống học vẹt.

Sau khi có kết quả cho từng fold, nhóm sẽ tính $mean \pm std$ để so sánh với bài báo gốc.

6 Tài liệu tham khảo

[1] Jonathan Hui. *Speech Recognition — GMM, HMM*. Medium. Truy cập: 2025-11-29. Sept. 2019. URL: https://jonathan-hui.medium.com/speech-recognition-gmm-hmm-8bb5eff8b196.

[2] Sid Talha, Anthony Fleury, and Sebastien Ambellouis. "Human Action Recognition from Body-Part Directional Velocity Using Hidden Markov Models". In: *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2017, pp. 1123–1128. DOI: 10.1109/ICMLA.2017.00185. URL: https://ieeexplore.ieee.org/document/8260778.

[3] Y. LeCun et al. "Backpropagation Applied to Handwritten Zip Code Recognition". In: *Neural Computation* 1.4 (1989), pp. 541–551. DOI: 10.1162/neco.1989.1.4.541.

[4] Ali Hamid Meftah et al. "Speaker Identification in Different Emotional States in Arabic and English". In: *IEEE Access* 8 (2020), pp. 60070–60083. DOI: 10.1109/ACCESS.2020.2983029.

[5] Pendukeni Phalaagae et al. "A Hybrid CNN-LSTM Model With Attention Mechanism for Improved Intrusion Detection in Wireless IoT Sensor Networks". In: *IEEE Access* 13 (2025), pp. 57322–57341. DOI: 10.1109/ACCESS.2025.3555861.

[6] Alexey Dosovitskiy et al. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=YicbFdNTTy.

[7] Jort F. Gemmeke et al. "Audio Set: An ontology and human-labeled dataset for audio events". In: *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*. 2017. URL: https://static.googleusercontent.com/media/research.google.com/vi//pubs/archive/45857.pdf.

[8] Yuan Gong, Yu-An Chung, and James Glass. "AST: Audio Spectrogram Transformer". In: *Proc. Interspeech*. 2021. URL: https://arxiv.org/abs/2104.01778.

[9] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *arXiv preprint arXiv:1412.6980* (2014). URL: https://arxiv.org/abs/1412.6980.

[10] Hongyi Zhang et al. "mixup: Beyond Empirical Risk Minimization". In: *Proc. Int. Conf. on Learning Representations (ICLR)*. 2018. URL: https://arxiv.org/abs/1710.09412.

[11] Daniel S. Park et al. "SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition". In: *Proc. Interspeech*. 2019. URL: https://arxiv.org/abs/1904.08779.

[12] Pavel Izmailov et al. "Averaging Weights Leads to Wider Optima and Better Generalization". In: *Proc. UAI*. 2018. URL: https://arxiv.org/abs/1803.05407.

[13] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. "Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017. URL: https://arxiv.org/abs/1612.01474.

[14] Karol J. Piczak. "ESC: Dataset for Environmental Sound Classification". In: *Proceedings of the 23rd ACM International Conference on Multimedia*. 2015, pp. 1015–1018. DOI: 10.1145/2733373.2806390. URL: https://dl.acm.org/doi/10.1145/2733373.2806390.