Mô hình nhận diện cử chỉ tay

1 Tổng quan

Để đưa ra được dự đoán từ bàn tay, chúng ta có thể tìm thấy rất nhiều cách nhưng 1 trong những cách dễ thực hiện nhất là dùng API Mediapipe từ Google để đưa ra dữ liệu về vị trí các khớp tay, qua đó chúng ta sẽ dự đoán. Đây có thể hiểu là 1 bài toán Classification thông thường, tuy nhiên các khớp tay thật chất các thông số sẽ rất dễ nhầm lẫn do chúng quá sát nhau, việc có thể tập trung vào 1 số đốt quan trọng là cần thiết nhằm đưa ra quyết định. Mô hình được xây dựng dựa trên giả thiết sự tập trung vào các đốt liệu có cải thiện thêm về kết quả. Qua đó, em sẽ trình bày lại 1 ít về lý thuyết cơ sở sau đó sẽ trình bày về những sự tinh chỉnh cho phù hợp với bài toán hiện tại. Nội dung phần trình bày sẽ gồm các phần chính: giải thích về cơ chế Multihead Self-Attention, Xây dựng lại mô hình từ mô hình gốc, Trình bày các kết quả so sánh giữa các mô hình.

2 Multihead Self-Attention

Multihead Self-Attention là bài toán con của Transformer và dần trở thành 1 trong những hướng giải quyết chủ yếu trong các vấn đề về xử lý ngôn ngữ từ 2018 đến nay với việc phân tích các feature(Feature Extraction) từ dữ liệu đầu vào. Em xin phép được trích lại bài toán ở Hình 1.

Hình 1. Multihead Self-Attention Problems

Self-Attention

What is self-attention? Self-attention calculates a weighted average of feature representations with the weight proportional to a similarity score between pairs of representations. Formally, an input sequence of n tokens of dimensions d, $X \in \mathbf{R}^{n \times d}$, is projected using three matrices $W_Q \in \mathbf{R}^{d \times d_q}$, $W_K \in \mathbf{R}^{d \times d_k}$, and $W_V \in \mathbf{R}^{d \times d_v}$ to extract feature representations Q, K, and V, referred to as query, key, and value respectively with $d_k = d_q$. The outputs Q, K, V are computed as

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V. \tag{1}$$

So, self-attention can be written as,

$$S = D(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_q}}\right)V, \tag{2}$$

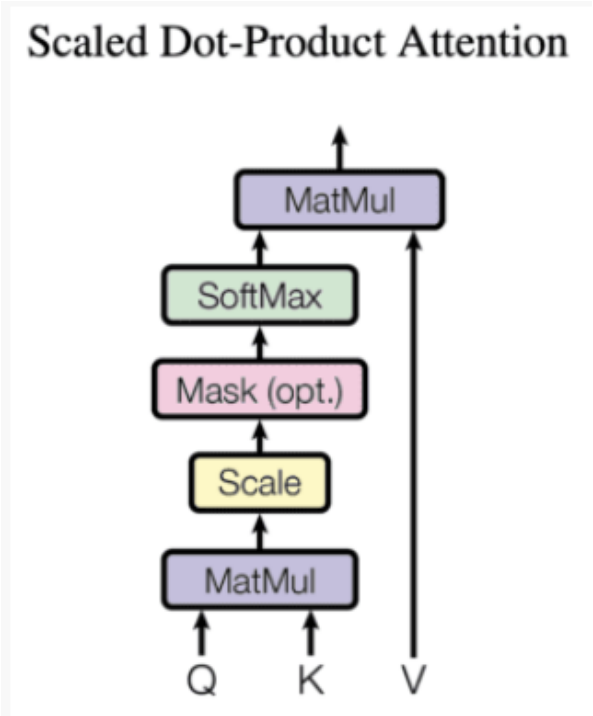
where softmax denotes a *row-wise* softmax normalization function. Thus, each element in S depends on all other elements in the same row.

Đây là 1 bài toán được đưa ra nhằm giải thích rõ hơn về cơ chế tập trung phân tích feature với việc dựa trên sư so sánh về tương đồng của 1 token với các token khác trong 1 sequence qua đó giúp model có thể học nhiều hơn. Cơ chế này thực chất không khác gì so với Multihead Attention thông thường, nhưng có quy định nghiêm ngặt hơn về chiều của $d_k v ă d_q$.

2.1 Scale-Dot Product

Tuy không phải là chương chính, nhưng trong phần này khái niệm về Query, Key và Value sẽ được trình bày, đây là 3 thành phần cấu tạo nên và lý giải về sự tập trung của mô hình.

Hình 2. Scaled Dot-Product Attention



Hình 2 trình bày về tổng quan của 1 bộ Scale-Dot Product. Ban đầu các token trong Query và Key sẽ được Matmul với nhau. Đây là phương thức tìm sự tương đồng giữa Query và Key, các giá trị có độ tương đồng cao sẽ trả về hệ số tập trung cao hơn. Để tránh tình trạng giá trị trở nên quá lớn hoặc quả nhỏ 1 cặp mask và softmax được áp dụng. Ngõ ra sẽ được nhân với value để cho ra kết quả sau cùng của lớp này, là 1 ma trận mới chứa những trọng số ưu tiên. Trọng số này được tạo ra từ ma trận Value nhưng tích hợp thêm quan trọng và liên quan đến query.

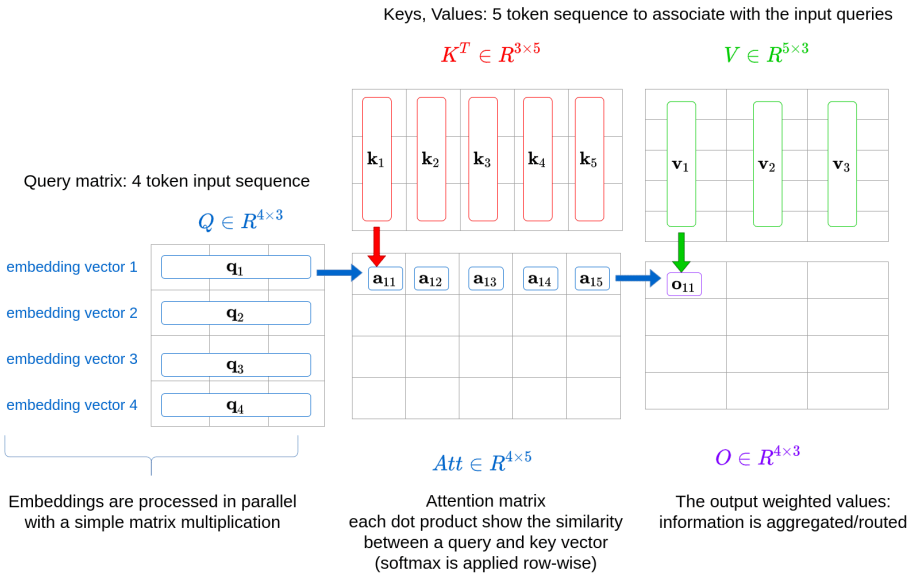
Để tránh việc, giá trị đi vào softmax ra 0, người ta cũng sử dụng 1 lớp mask. Ngoài ra để giá trị trước khi qua softmax và lớp mask cũng được giảm nhỏ lại với việc chia cho căn của số phân tử trong 2 lớp key và query.

Công thức tổng quan có thể viết lại là:

$$attention(Q, K, V) = softmax(\frac{QK^T}{sqrt(d_k)}$$

Hình 3 là 1 ví dụ nhằm giúp chúng ta hiểu rõ hơn về cơ chế Matmul, để trình bày thì em sẽ tạm bỏ qua điều kiện về chiều của K, Q. Ở đây Q,K,V sẽ có lần lượt 4,5,5 token trong 1 sequence và mỗi sequence sẽ 3 vecto nhúng. Ở bảng attention matrix, ta có thể thấy bảng lưu lại sự tương đồng của các token trong Key và Query lại. Sau khi qua xử lý bởi lớp Mask và Softmax, chúng được nhận một lần nữa với ma trận Value, để hình thành bộ hệ số tập trung. Ngõ ra này có chiều giống với lại chiều của Query.

Hình 3. Scale-Dot Illustration



2.2 MultiHead Self-Attention

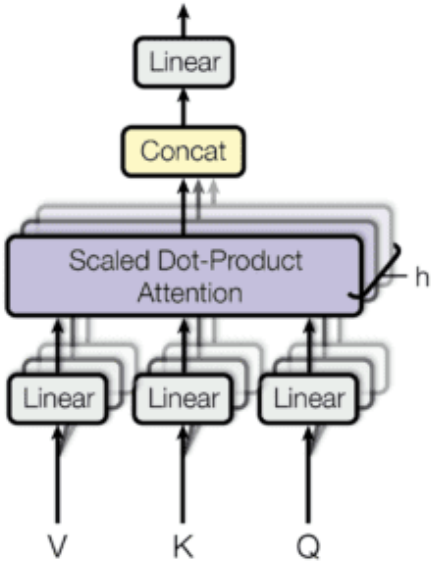
Hình 4 cho chúng ta cái nhìn tổng quát về bộ MultiHeadAttention. Mô hình 3 gồm 3 ngõ vào Value, Key, Query tương tự bộ Scaled Dot-Product Attention. 3 ma trận này được sinh ra từ X ban đầu với ma trận học ban đầu như trong phần lý thuyết. Để đa dạng về dữ liệu, các lớp Q,K,V sẽ được hình thành riêng với h lớp. Sau mỗi lớp, 1 ma trận tập trung mới sẽ được hình thành, và được concatenate lại thành 1 ma trận lớn hơn qua lớp linear sẽ hình thành 1 ma trận đại diện cho các đặc trưng của dữ liệu.

2.3 Mô hình Transformer 'Attention is All You Need'

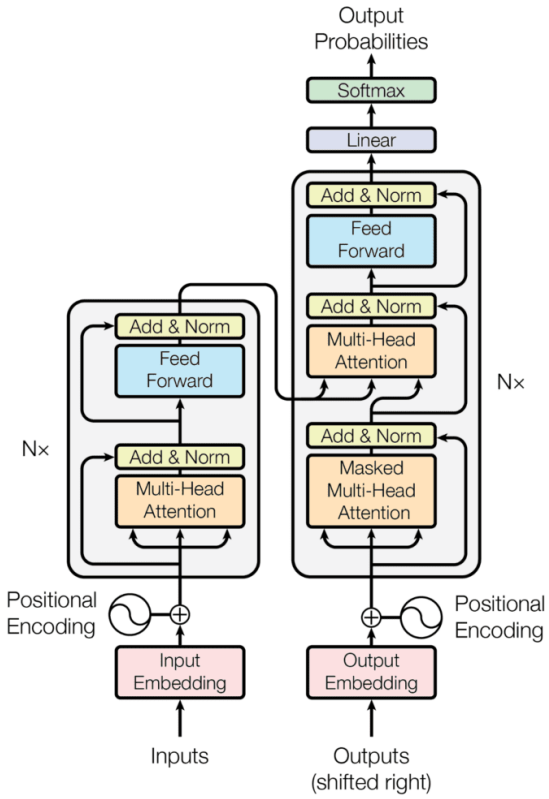
Đây là mô hình thuần túy nhất của Transformer, gồm 2 bộ Encoder và Decoder chịu trách nhiệm lần lượt cho việc chiết suất đặc tính và khởi tạo sequence mới. Mô hình được thể hiện qua hình 5.

Mô hình gồm 1 bộ Positional Encoding chịu trách nhiệm để đặc trưng hóa chuỗi text đầu vào. Có thể hiểu là 1 text đầu vào là 1 sequence gồm các chữ là các token. Mỗi token sẽ được chuyển hóa thành 1 ma trận đặc trưng cho nó. Đặc biệt trong text sẽ khác với các dữ liệu khác ở chỗ mỗi thứ tự về từ ngữ khác nhau sẽ có 1 ý nghĩa khác nhau, do đó các giá trị trong ma trận sẽ được cộng thêm 1 thành phần vị trí nhúng. Trong mô hình còn chứa 1 bộ Add và Normalize giữa các lớp. Đây là 1 lớp hỗ trợ nhằm tránh hệ số trở lên quá lớn hoặc quá nhỏ.

Hình 4. MultiHeadAttention



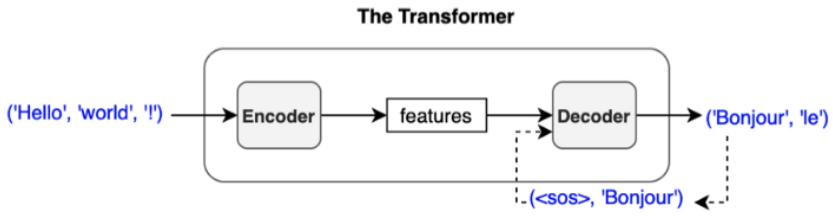
Hình 5. Mô hình Transformer - 'Attention is All You Need'



Tiếp đến, hãy đi sâu để hiểu về cơ chế của Encoder và Decoder. Bộ Encoder phía bên trái của mô hình chịu trách nhiệm cho việc feature extraction. Cụ thể, ngõ vào là sequence đầu vào. Dữ liệu đầu ra bộ Multihead Attention là ma trận chứa các hệ số tập trung, sau bộ feed forward network, chúng hình thành 1 ma trận chứa các feature. Ngõ ra khi này vẫn là ma trận 3 chiều.

Về phía bộ Decoder phía phải của mô hình, 1 vòng lặp sẽ diễn ra để hình thành toàn bộ các token ngõ ra. Tại mỗi vòng, các token đã được hình thành từ trước được thêm vào bắt đầu với 1 token 'start-of-sequence'. Một lớp mask đươc áp dụng vào để đảm bảo mỗi token được tạo ra từ quá trình học không phụ thuộc vào các token ở các vị trí khác trong tương lai. Sau đó, một cặp giống bộ Encoder xuất hiện để tiến hành phân tích feature. Điều khác biệt là Key và Value của bộ này là ngõ ra từ bộ Encoder. Khi đó bộ matmul đầu tiên giữa Key và Query chính là sự tương quan giữa feature extraction phía encoder và các token đã có ở thời điểm hiên tại, điều này sẽ sinh ra 1 ma trận tập trung qua đó, áp dụng vào lớp Value nhằm hình thành 1 ma trận đặc trưng mới. Sau bộ linear và softmax, mô hình sẽ đưa ra dự đoán tại vị trí hiện tại cho đến khi một 'end-of-sequence' được gửi tới hoặc khi nó đạt tới giới hạn về chiều dài chuỗi. Ngõ ra từ bộ model sẽ được so sánh với target và cải thiện thêm mô hình. Một vòng đời của bộ Decoder có thể trực quan như hình 6.

Hình 6. Decoder Lifecycle



Nhìn chung, mô hình Transformer có bộ Decoder khá phức tạp và nó thường dùng để khởi tạo lần lượt các token. Điều này không quá cần thiết với bài toán Classification. Do đó, trong phần này điều quan trọng nhất cần được học tập và kế thừa chính là kiến trúc của bộ Encoder nhằm đưa ra feature extraction phù hợp.

3 Xây dựng mô hình

Trong chương này, em sẽ đưa ra các mô hình mà em đã thử nghiệm trong thời gian thực hiện đồ án này. Các mô hình này dựa trên 2 bài toán chủ yếu là Sử dụng Neural Network cho bài toán Classification và Transformer.

3.1 Tổng quan về bài toán

Bài toán yêu cầu từ ma trận chứa 21 dữ liệu về các khớp tay ban đầu đưa ra dự đoán về phân loại khớp tay. Để giải quyết bài toán, em sẽ sử dụng 2 phương pháp:

Phương pháp 1: Xây dựng mô hình Classification từ mạng Neural Network

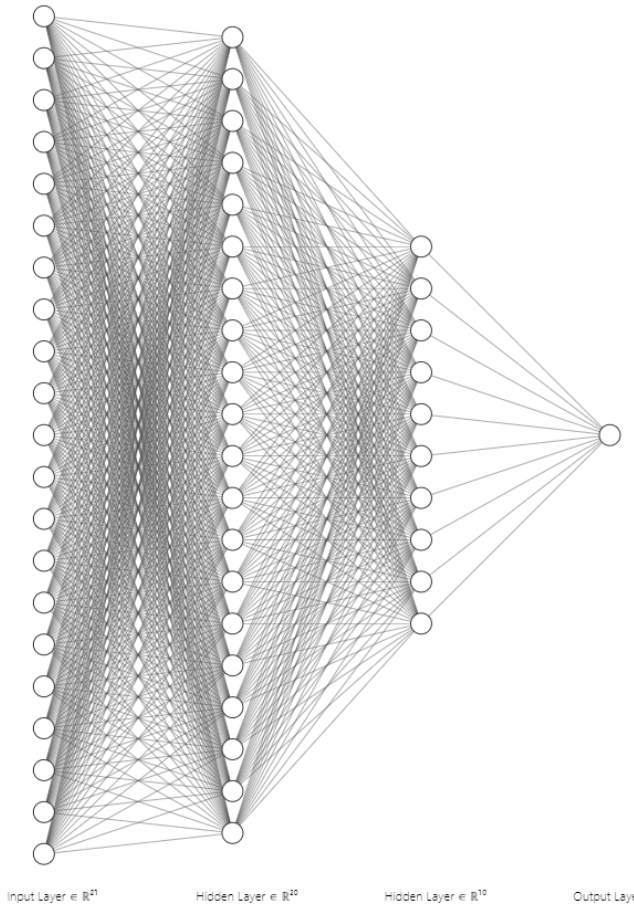
Phương pháp 2: Xây dựng mô hình dựa trên bộ Encoder trong mô hình Transformer kết hợp 1 lớp Softmax sau cùng. Ở phương pháp này, do sự đặc biệt của bài toán là chỉ có 21 giá trị đầu vào. Do đó, để phù hợp với mô hình Transformer em sẽ chia thành 2 kiểu (1,21) (tức

1 token chứa 21 giá trị trong embedded vector) và (21,1) (tức 21 token chứa 1 giá trị trong embedded vector).

3.2 Classification Neural Network Solution

Trong phương pháp này, em sẽ dùng 2 bộ Feed Forward Network và 1 bộ softmax ở ngõ ra để đưa ra dự đoán. Cụ thể sơ đồ khối của mô hình như sau:

Hình 7. Classification Neural Network

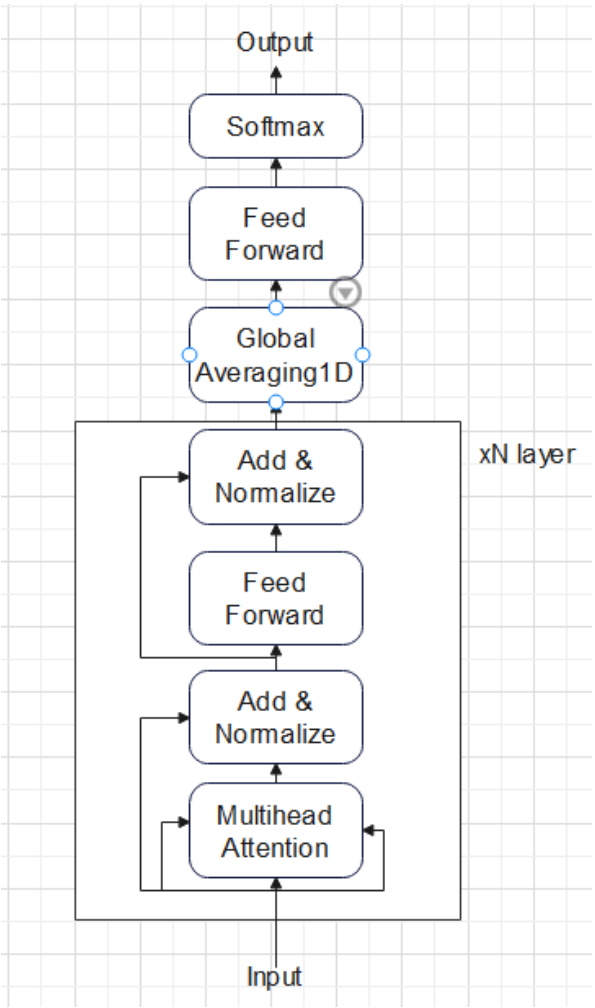


Cụ thể ban đầu gồm 1 ma trận 21 giá trị sẽ qua 1 lớp DenseLayer với hàm kích hoạt RelU đưa về ma trận 20 giá trị. Sau đó, 1 lớp Dense Layer sẽ áp dụng để thu về ma trận 8 ngõ ra và được tính toán softmax lại nhằm thu được giá tri dự đoán cuối cùng của bài toán.

3.3 Transformer Solution

Cả 2 phương pháp mà em trình bày trong phần này đều sử dụng chung 1 sơ đồ khối được xây dựng kết hợp mô hình Transformer và mô hình Classification. Điểm khác biệt nằm ở cách xây dựng ngõ vào của hệ thống.

Hình 8. Transformer solution



Input đầu vào sau khi qua bộ Multihead Attention sẽ hình thành 1 ma trận chứa trọng số tập trung có chiều tương đương ma trận đầu vào. Để đảm bảo tránh hiện tượng vanishing gradient, khi 1 vài trọng số trước đó trở nên cực kì nhỏ, 1 lớp add được thêm vào để đảm bảo tránh được điều đó. 1 lớp normalize cũng được xuất hiện, ngược lại với lớp add, lớp normalize tránh việc các trọng số trở nên quá lớn và dẫn đến mô hình học lệch hoàn toàn. Tiếp đến 1 lớp Feed Forward dùng để chuyển ngõ ra chỉ đơn thuần là những neural thể hiện sự tập trung thành 1 ma trận chứa feature, lớp add và normalize cũng được áp dụng ở đây.

Tiếp đến, do ngõ ra của bộ Encoder là 1 ma trận 3 chiều (batch, token, embedded), vì vậy để làm tiếp các bước kế tiếp, em sẽ sử dụng 1 bộ Global Averaging 1D để đưa về 2 chiều (batch, feature) từ đây 1 lớp feed forward và softmax được áp dụng như model Classification để đưa ra dự đoán.

3.3.1 Sử dụng ngõ vào là ma trận có chiều (21,1)

Ở đây, mỗi token chỉ có 1 giá trị trong embedded vector, và điều kì vọng của em là qua lớp Dense Layer ứng với từng ma trận Q,K,V sẽ hình thành nên 1 ma trận Q,K,V mới có số lượng phần tử trong embedded vector lớn hơn, đây là 1 thử thách lớn khi để cho máy tự sinh ra các đặc tính của 1 token ban đầu.

3.3.2 Sử dụng ngõ vào là ma trận có chiều (1,21)

Đây là 1 phương pháp có thể nói là làm mất đi ý nghĩa của mô hình Transformer ban đầu, khi chỉ sử dụng 1 token duy nhất, điều này khiến cho đầu ra từ softmax(QK) luôn bằng 1 và khiến cho khối Multihead Attention trở thành khối Feed-Forward Network. Tuy nhiên, đây lại là phương pháp mang lại hiệu quả rất cao, điều này vô hình chung giúp em đánh giá khối Encoder giải quyết rất tốt vấn đề Classification tuy nhiên, mô hình Attention có vẽ không phù hợp với bài toán input chỉ là 1 ma trận 1 chiều duy nhất.

4 Tiến hành train và đánh giá kết quả

Em sẽ kiểm tra bài toán với ngõ ra ít nhất là 8 loại ứng với 3 led on/off. Em sẽ đánh giá thông qua 3 yếu tố là Loss, accuracy của tập test; biểu đồ hàm loss và train trong quá trình huấn luyện; confussion matrix giữa các class

Để kiểm định kết quả của 3 mô hình được đề ra, em sẽ kiểm tra trong 3 trường hợp để đánh giá khả năng của mô hình:

TH 1: Kiểm tra trong nội bộ những kết quả đúng với tay trái.

TH 2: Kiểm tra khi có thêm 1 class chứa các TH sai với 1 tay trái.

TH3: Kiểm tra có 1 class sai và nhận diện 2 tay.

4.1 Kiểm tra nội bộ những kết quả đúng với tay trái

4.1.1 Mô hình Classification Neural Network

Train với đầu vào 42 (2 lớp ảnh), 2 lớp relu 32-16 neural, 1 lớp softmax 8 class. Một số thông số huấn luyện khác bao gồm: Dropout giữa các lớp là 0.4, EarlyStopping với patience là 20 và verbose là 1, learning rate là 1e-4, sử dụng hàm loss sparse categorical crossentropy, đây

sẽ là hàm loss dùng chung cho tất cả model do em tìm hiểu được nó phù hợp với bài toán classification. Train với 1500 epoch, mỗi epoch có batch size là 128.

Kết quả kiểm định:

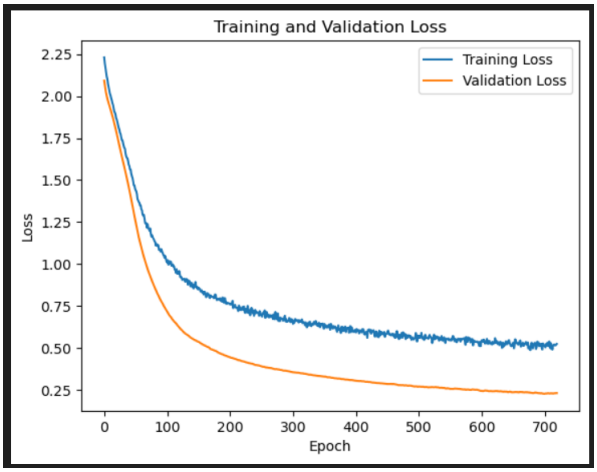
Mô hình early stop sau 720 epoch.

Kết quả từ tập test: accuracy: 85

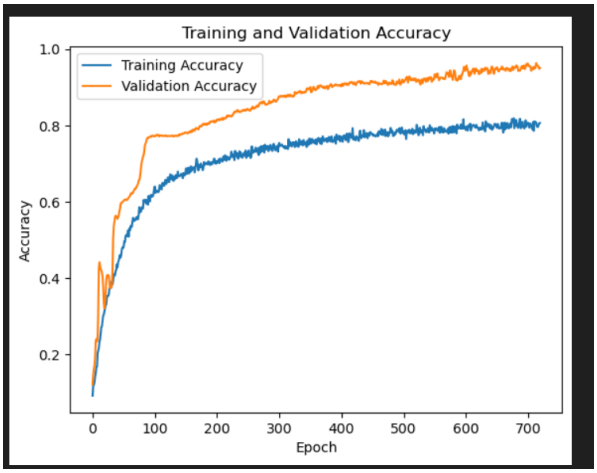
Đường cong học của dữ liệu theo hàm loss ở hình 9 và theo accuracy ở hình 10.

Kết quả của ma trận nhầm lẫn được trình bày ở hình 11.

Hình 9. Training and Validation Loss of Neural Network



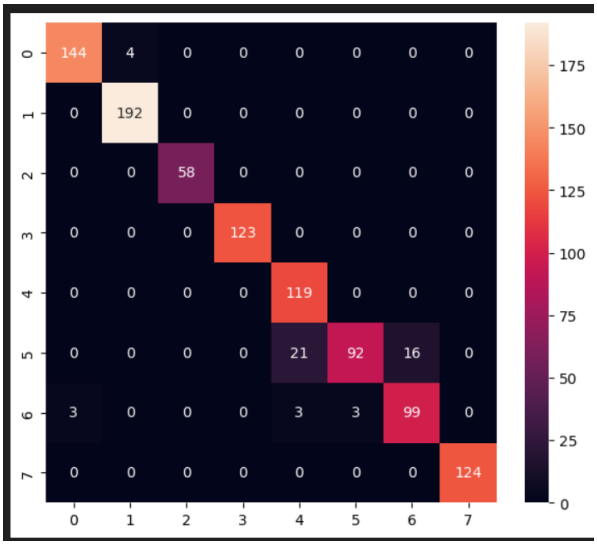
Hình 10. Training and Validation Accuracy of Neural Network



4.1.2 Mô hình Transformer với ngõ vào (42,1)

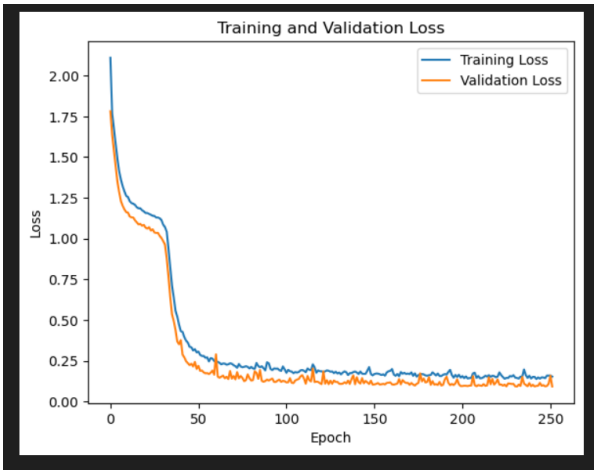
Train với đầu vào (42,1) (2 lớp ảnh), embed dim = 10, số lượng head =2, chiều feed forward trong bộ encode = 20. Neural của lớp Dense trước lớp Softmax là 64. Một số thông số huấn luyện khác bao gồm: Dropout giữa các lớp là 0.3, EarlyStopping với patience là 20 và verbose là 1, learning rate là 1e-3. Train với 1500 epoch, mỗi epoch có batch size là 128.

Hình 11. Confussion Matrix of Neural Network



Kết quả:
Mô hình Early Stopping sau 250 epoch.
Kết quả từ tập test: 93
Đường cong học của dữ liệu theo hàm loss ở hình 12 và theo accuracy ở hình 13.
Kết quả của ma trận nhầm lẫn được trình bày ở hình 14.

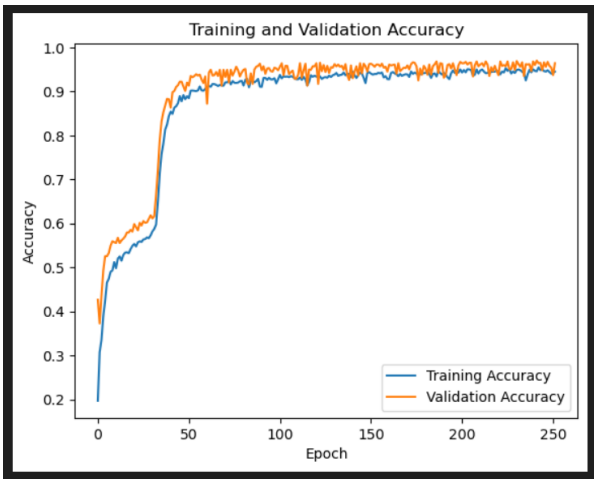
Hình 12. Training and Validation Loss of Transformer(42,1)



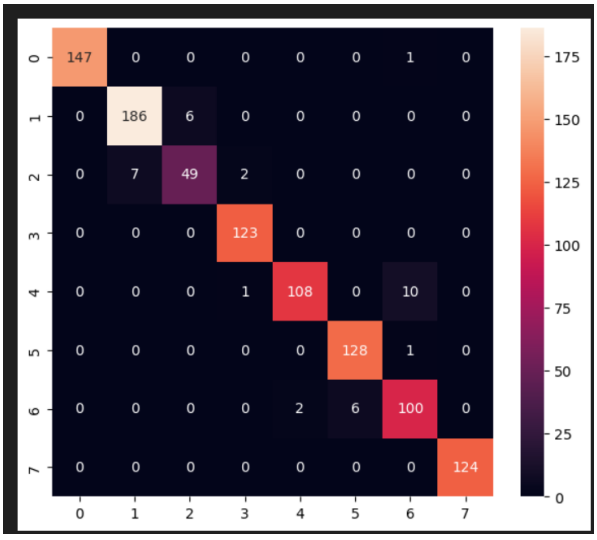
4.1.3 Mô hình Transformer với ngõ vào (1,42)

Train với đầu vào (1,42) (2 lớp ảnh), embed dim = 42, số lượng head =2, chiều feed forward trong bộ encode = 20. Do đây là 1 trường hợp đặc biệt, nên số chiều thứ 3 trong quá trình attention phải chính là chiều của inputs. Neural của lớp Dense trước lớp Softmax là 64. Một số thông số huấn luyện khác bao gồm: Dropout giữa các lớp là 0.3, EarlyStopping với patience

Hình 13. Training and Validation Accuracy of Transformer(42,1)



Hình 14. Enter Caption



là 20 và verbose là 1, learning rate là 1e-4. Train với 1500 epoch, mỗi epoch có batch size là 128.

Kết quả:

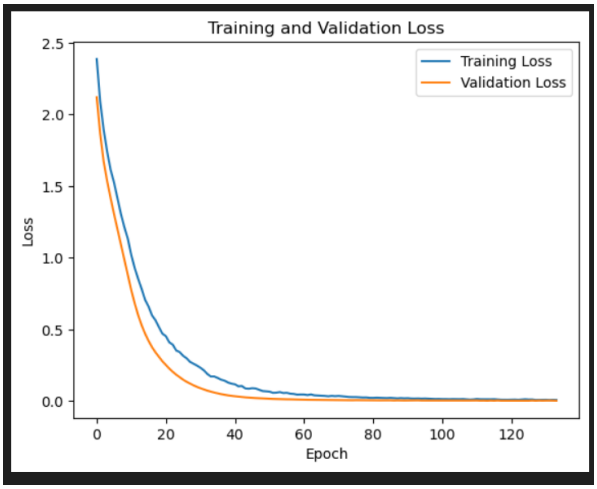
Mô hình Early Stopping sau 134 epoch.

Kết quả từ tập test: 99.90010261535645

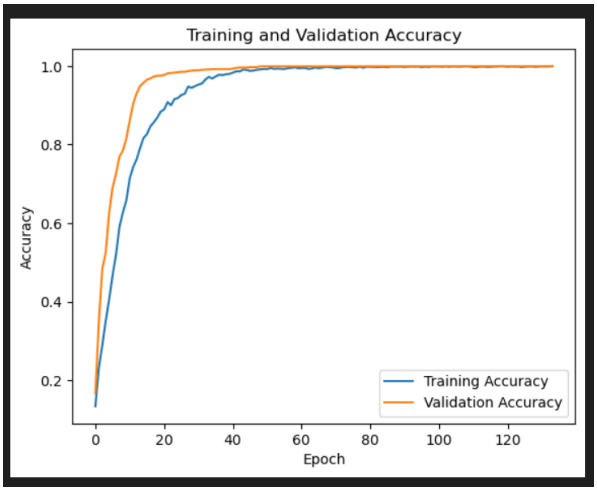
Đường cong học của dữ liệu theo hàm loss ở hình 15 và theo accuracy ở hình 16.

Kết quả của ma trận nhầm lẫn được trình bày ở hình 17.

Hình 15. Training and Validation Loss of Transformer(1,42)



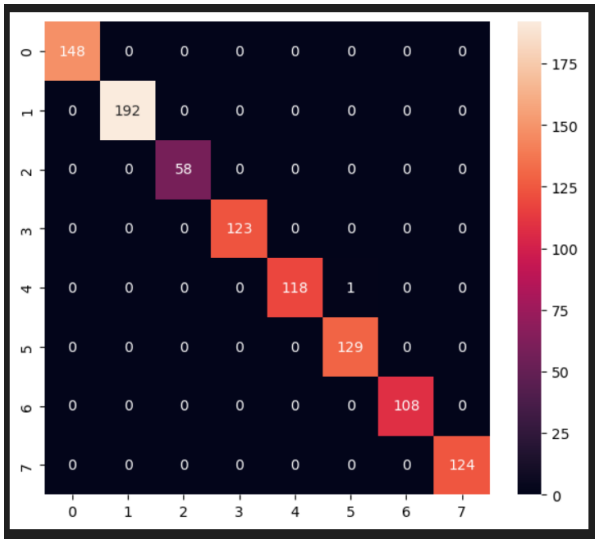
Hình 16. Training and Validation Accuracy of Transformer(1,42)



4.2 Kết luận

Cả 3 mô hình đều cho ra kết quả cao từ 80 - 99 phần trăm. Đặc biệt là 2 mô hình Transformer cho kết quả trên 90 pt. Mô hình với ngõ vào (1,42) tuy không giữ được về mặt lý thuyết cốt lõi của 1 mô hình Transformer nhưng lại cho 1 kết quả rất cao. Điều này giúp em đặt ra 1

Hình 17. Enter Caption



nghi vấn kiến trúc của Transformer giải quyết bài toán Classification khá tốt. Tuy nhiên, bộ multihead attention tuy là cốt lõi nhưng không thực sự hợp với bài toán này.

5 Trích dẫn

Tài liệu

[1] Nikolas Adaloglou. *Why Multihead Self-Attention Work*. https://theaisummer.com/self-attention/.

[2] Google Research/Google Brain. *Attention Is All You Need*. https://papers.nips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[3] Stefania Cristina. *Build Transformer From Scratch*. https://machinelearningmastery.com/the-transformer-model/.

[4] *Transformer Encoder-Decoder*. https://kikaben.com/transformers-encoder-decoder/.