# "VNExpress Text Classification"

**Đàm Văn Hiển - 22022664, Nguyễn Hoàng Vũ - 22022502, Ngô Huy Hoàn - 22022590, Trần Kim Thành - 22022532, Trần Phạm Hoàng - 22022669, Tạ Nguyên Dũng - 22022546**[1]

[1] *Vietnam National University, Hanoi, 144 Xuan Thuy, Cau Giay, Hanoi , Vietnam*

**Tóm tắt nội dung.** *Phân loại văn bản là một trong những bài toán quan trọng trong lĩnh vực xử lý ngôn ngữ tự nhiên (NLP), đặc biệt với tiếng Việt. Trong bài báo này, chúng em sử dụng tập dữ liệu bao gồm các bài báo từ VnExpress để thực hiện phân loại văn bản thành các danh mục khác nhau. Chúng em triển khai và so sánh hiệu suất của các mô hình học sâu phổ biến bao gồm MLP, CNN for text, LSTM và Transformer. Các mô hình được đánh giá dựa trên độ chính xác, F1-score và thời gian huấn luyện. Kết quả thực nghiệm cho thấy mô hình mBERT đạt kết quả vượt trội so với các mô hình khác, đặc biệt trong việc nắm bắt các đặc điểm ngữ nghĩa phức tạp của văn bản tiếng Việt. Nghiên cứu này cung cấp cái nhìn tổng quan về hiệu quả của các mô hình học sâu trên bài toán phân loại văn bản tiếng Việt, đồng thời đề xuất các hướng cải tiến trong tương lai.*

Keywords: deep learning, text classification, transformer, gru.

## 1. Introduction

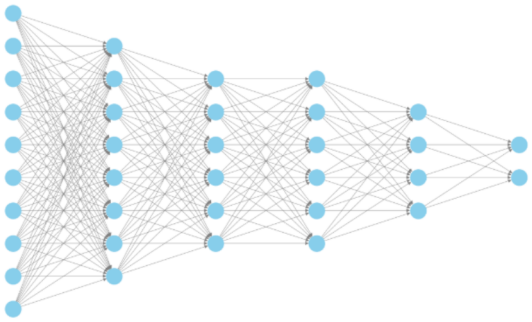
Phân loại văn bản là một bài toán quan trọng trong xử lý ngôn ngữ tự nhiên, được ứng dụng rộng rãi trong các lĩnh vực như phân tích cảm xúc, phát hiện spam, và quản lý nội dung. Với sự phát triển mạnh mẽ của báo chí trực tuyến, việc phân loại các bài báo theo các chuyên mục không chỉ hỗ trợ tổ chức dữ liệu hiệu quả mà còn cải thiện trải nghiệm người dùng. Tuy nhiên, tiếng Việt là một ngôn ngữ có cấu trúc phức tạp, đặt ra nhiều thách thức trong việc xây dựng các mô hình phân loại văn bản chính xác.

Trong bài báo này, chúng em tập trung vào việc áp dụng và so sánh hiệu suất của các mô hình học sâu như MLP, CNN for text, LSTM và Transformer trong bài toán phân loại văn bản tiếng Việt. Tập dữ liệu nghiên cứu được thu thập từ VnExpress, một trang báo điện tử phổ biến tại Việt Nam, với nhiều bài viết thuộc các chuyên mục khác nhau. Mục tiêu của nghiên cứu là đánh giá hiệu quả của từng mô hình trong việc phân loại các bài viết dựa trên ngữ cảnh và đặc điểm ngôn ngữ.

Phần còn lại của bài báo được tổ chức như sau: Phần 2 trình bày các phương pháp nghiên cứu và các mô hình được sử dụng; Phần 3 mô tả thí nghiệm và kết quả đạt được; và Phần 4 thảo luận các kết quả và đưa ra kết luận cùng hướng nghiên cứu tương lai.

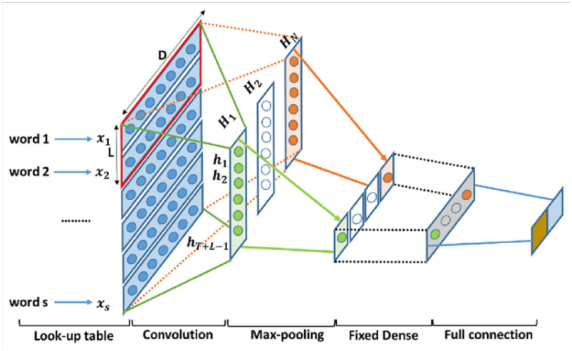
## 2. Models

### *2.1. MLP*



Mô hình MLP được thiết kế với cấu trúc gồm các thành phần chính như sau: Dữ liệu đầu vào của mô hình là các vector embedding có kích thước 768, được trích xuất từ PhoBERT. Để học các đặc trưng phức tạp trong dữ liệu, mô hình sử dụng năm lớp ẩn có kích thước lần lượt là hidden dim1, hidden dim2, hidden dim3, hidden dim4 và hidden dim5, với hàm kích hoạt LeakyReLU được áp dụng ở mỗi lớp. Để giảm thiểu hiện tượng overfitting, nhóm đã áp dụng phương pháp dropout với các tỷ lệ lần lượt là 0.2, 0.3 hoặc 0.5 giữa các lớp. Cuối cùng, lớp đầu ra của mô hình có kích thước 15, tương ứng với số nhãn phân loại cần dự đoán.

### *2.2. CNN*

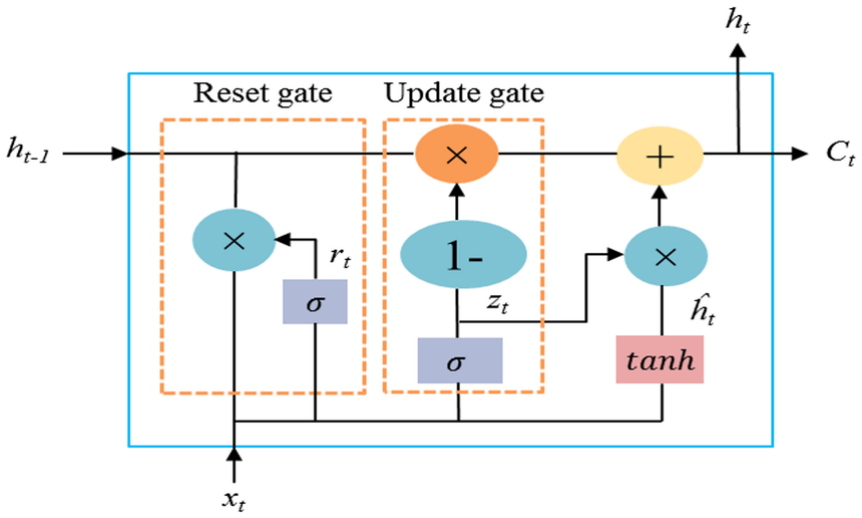



Mô hình CNN được xây dựng để xử lý các chuỗi văn bản với các biến chính gồm: vocab size là số lượng từ vựng trong tập dữ liệu, embedding dim là số chiều của embedding, và padding idx đại diện cho độ dài tối đa của một câu, được giới hạn ở mức 256. Kích thước của lớp ẩn được xác định bởi hidden dim, trong khi num classes thể hiện số lớp phân loại mà mô hình sẽ học. Để giảm thiểu hiện tượng overfitting, mô hình sử dụng kỹ thuật dropout với tỷ lệ được điều chỉnh thông qua biến drop out.

Mô hình bao gồm năm lớp chính: lớp Embedding giúp biểu diễn các từ dưới dạng vector embedding; lớp Conv1 là mạng nơ-ron tích chập 1 chiều, phù hợp để xử lý chuỗi văn bản; lớp fc (fully-connected) thực hiện dự đoán kết quả từ các đặc trưng đã được trích xuất; lớp dropout giúp

loại bỏ một số kết nối giữa các neuron để tăng tính tổng quát của mô hình; lớp relu được sử dụng làm hàm kích hoạt để tạo phi tuyến tính; cuối cùng, lớp pool thực hiện max pooling nhằm giảm kích thước đặc trưng, tăng hiệu quả tính toán và giảm thiểu nhiễu.

### *2.3. GRU*



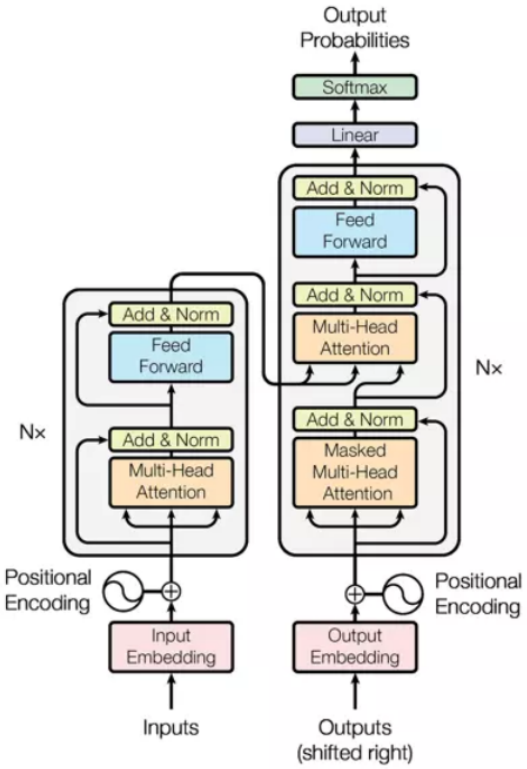

GRU (Gated Recurrent Unit) là một loại mạng nơ-ron hồi tiếp (RNN) được giới thiệu vào năm 2014 bởi Kyunghyun Cho và các cộng sự. GRU được thiết kế để giải quyết vấn đề vanishing gradient (gradient biến mất) trong mạng RNN truyền thống, giúp mạng học được mối quan hệ dài hạn trong dữ liệu tuần tự. Đây là một phiên bản đơn giản hơn của LSTM, với ít tham số hơn, nhưng vẫn mang lại hiệu quả cao.

Cấu trúc của GRU bao gồm hai cổng chính: cổng cập nhật (Update Gate) và cổng quên (Reset Gate). Cổng cập nhật quyết định giữ lại bao nhiêu thông tin từ trạng thái trước đó và thêm bao nhiêu thông tin mới. Trong khi đó, cổng quên xác định lượng thông tin từ trạng thái cũ cần loại bỏ, giúp mạng tập trung vào các thông tin quan trọng tại thời điểm hiện tại.

Nhờ thiết kế đơn giản và hiệu quả, GRU thường được sử dụng trong các bài toán xử lý dữ liệu tuần tự như phân tích chuỗi thời gian, xử lý ngôn ngữ tự nhiên, hoặc dịch máy. So với LSTM, GRU có tốc độ tính toán nhanh hơn và phù hợp với các ứng dụng yêu cầu hiệu năng cao.

### *2.4. Transformer*

Transformer là một kiến trúc deep learning tiên tiến, được giới thiệu trong bài báo nổi tiếng "Attention is All You Need"của Vaswani và cộng sự vào năm 2017. Đây là một kiến trúc dựa trên cơ chế attention và không sử dụng các cơ chế tuần tự như RNN hay LSTM, cho phép xử lý toàn bộ chuỗi dữ liệu song song, thay vì phải thực hiện theo thứ tự. Transformer bao gồm hai phần chính: Encoder và Decoder. Encoder được xây dựng từ  N lớp giống nhau, mỗi lớp gồm ba thành phần chính: cơ chế Multi-Head Self-Attention để tính toán mức độ liên quan giữa tất cả các từ trong chuỗi đầu vào, mạng Feed Forward Neural Network áp dụng độc lập trên từng token để học các đặc trưng phi tuyến tính, và Add Norm kết hợp skip connection với layer normalization để tăng tính ổn định trong huấn luyện. Ngoài ra, Encoder còn tích hợp Positional Encoding để duy trì thông tin thứ tự của các từ trong chuỗi đầu vào.




Về phía Decoder, cấu trúc tương tự Encoder nhưng được bổ sung hai cơ chế đặc biệt là Masked Multi-Head Attention, giúp tính toán attention giữa các từ trong câu đầu ra chỉ với các từ đã xuất hiện trước đó, và Encoder-Decoder Attention, tính toán attention giữa đầu ra của Encoder và đầu vào của Decoder để kết hợp thông tin từ cả hai phần. Các lớp Feed Forward Neural Network và Add Norm trong Decoder vẫn giữ vai trò như ở Encoder.

### 2.5. *Finetune mBERT and BERT*

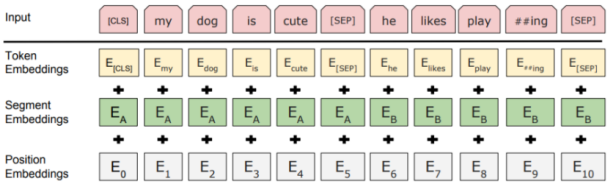



Figure 2: BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings.

BERT (Bidirectional Encoder Representations from Transformers) là một mô hình ngôn ngữ mạnh mẽ được phát triển bởi Google AI vào năm 2018, dựa trên kiến trúc Transformer. Điểm nổi bật của BERT là khả năng học ngữ cảnh song hướng (bidirectional), tức là hiểu ngữ cảnh của một từ dựa trên cả hai chiều của câu, thay vì chỉ từ trái sang phải hoặc ngược lại như các mô hình

trước đây. BERT được huấn luyện trước (pre-training) trên lượng dữ liệu lớn bằng hai nhiệm vụ: Masked Language Model (MLM), trong đó một số từ bị che đi và mô hình dự đoán từ đó dựa trên ngữ cảnh, và Next Sentence Prediction (NSP), để xác định xem câu thứ hai có phải là câu tiếp theo của câu thứ nhất hay không. Sau đó, BERT có thể được tinh chỉnh (fine-tuning) để áp dụng cho nhiều bài toán như phân loại văn bản, trích xuất thực thể (NER), trả lời câu hỏi (QA).

mBERT (Multilingual BERT) là một phiên bản mở rộng của BERT được phát triển bởi Google, nhằm hỗ trợ nhiều ngôn ngữ khác nhau trong một mô hình duy nhất. Không giống như BERT gốc chỉ được huấn luyện trên dữ liệu tiếng Anh, mBERT được huấn luyện trên 104 ngôn ngữ, bao gồm cả tiếng Việt, dựa trên dữ liệu từ Wikipedia của từng ngôn ngữ. Điều này giúp mBERT trở thành một công cụ mạnh mẽ để xử lý các bài toán ngôn ngữ đa ngữ hoặc các ngôn ngữ không có nhiều dữ liệu. Kiến trúc của mBERT tương tự BERT-Base, với 12 lớp (layers), 12 đầu self-attention (attention heads), và 768 chiều vector ẩn (hidden size). mBERT cũng được huấn luyện trước bằng hai nhiệm vụ: Masked Language Model (MLM) và Next Sentence Prediction (NSP), cho phép mô hình học được ngữ cảnh của các từ trong nhiều ngôn ngữ khác nhau. Mặc dù mBERT không sử dụng bất kỳ thông tin đặc thù nào về ngôn ngữ cụ thể, nó vẫn có thể đạt hiệu suất tốt nhờ khả năng học biểu diễn ngữ nghĩa chung giữa các ngôn ngữ.

## 3. Experiment

### *3.1. Dataset*

#### *3.1.1. Nguồn dữ liệu*

Tập dữ liệu được thu thập từ trang báo điện tử **VnExpress** — một nguồn thông tin đa dạng và phong phú, phù hợp cho việc huấn luyện các mô hình xử lý ngôn ngữ tự nhiên (NLP).

Dữ liệu bao gồm các bài viết thuộc 15 chuyên mục khác nhau, cụ thể là: *Thời sự, Thế giới, Kinh doanh, Bất động sản, Khoa học, Giải trí, Thể thao, Pháp luật, Giáo dục, Sức khỏe, Đời sống, Du lịch, Số hóa, Ôtô - Xe máy, Ý kiến*.

Tập dữ liệu này được sử dụng để huấn luyện và đánh giá hiệu suất của các mô hình học sâu như **Transformer, MLP, CNN** và **GRU** trong bài toán phân loại văn bản.

#### *3.1.2. Thu thập dữ liệu*

Tập dữ liệu bao gồm tổng cộng **30,000 bài viết** từ 15 chuyên mục khác nhau, với **2,000 bài viết** cho mỗi chuyên mục. Phân phối đồng đều này đảm bảo rằng các mô hình có đủ mẫu để học đặc trưng của từng chuyên mục, giảm thiểu nguy cơ mất cân bằng dữ liệu trong quá trình huấn luyện.

#### *3.1.3. Tiền xử lý dữ liệu*

Để chuẩn bị dữ liệu đầu vào cho các mô hình, các bước tiền xử lý sau đã được thực hiện:

- **Loại bỏ dữ liệu không cần thiết**: Xóa bỏ các ký tự đặc biệt, thẻ HTML, hoặc các phần tử không liên quan trong bài viết.
- **Tokenization**: Sử dụng công cụ **ViTokenizer** để tách các từ trong văn bản, giúp nhận diện chính xác các từ đơn và từ ghép trong tiếng Việt.
- **Chuyển đổi văn bản thành token ID**: Áp dụng **AutoTokenizer** từ mô hình `"vinai/phobert-base-v2"` để mã hóa văn bản thành chuỗi các ID token, đây là đầu vào cho các mô hình học sâu.

- **Chuyển đổi nhãn văn bản (Topic) sang dạng số**: Các chuyên mục được gán mã số từ 0 đến 14 để thuận tiện cho việc xử lý và tính toán trong quá trình huấn luyện mô hình.

### *3.2. Results*

**Bảng 1.** So sánh kết quả các mô hình trong bài toán phân loại văn bản

| Mô hình | Accuracy |
|---|---|
| MLP | 84.10 |
| CNN for Text | 81.70 |
| GRU | 84.30 |
| Transformer | 84.26 |
| Finetune BERT | 77.87 |
| Finetune mBERT | 86.63 |

### *3.3. Tuning Hyperparameters*

Trong quá trình huấn luyện các mô hình `MLP` và `Transformer`, chúng tôi thực hiện tối ưu hóa các siêu tham số (hyperparameters) để cải thiện hiệu suất mô hình trên tập kiểm thử. Việc điều chỉnh các siêu tham số giúp mô hình học tốt hơn từ dữ liệu và đạt được độ chính xác cao hơn.

*MLP (Multilayer Perceptron)*

Để tối ưu hóa mô hình `MLP`, chúng tôi áp dụng phương pháp `Grid Search` để tìm kiếm các giá trị tối ưu cho các siêu tham số chính. Các tham số điều chỉnh bao gồm:

- **Kích thước các lớp ẩn (Hidden Layer Dimensions)**: Chúng tôi thử nghiệm với các kích thước lớp ẩn khác nhau như 512, 256, 128, 64 và 32, nhằm tìm ra cấu hình phù hợp giúp mô hình học các đặc trưng phức tạp từ dữ liệu.
- **Tỷ lệ dropout**: Tỷ lệ dropout được thử nghiệm trong khoảng 0.2 đến 0.4 để giảm thiểu hiện tượng *overfitting* và tăng khả năng tổng quát của mô hình.
- **Optimizer**: Các optimizer `Adam` và `AdamW` được thử nghiệm để kiểm tra hiệu quả hội tụ của mô hình.
- **Learning rate**: Các giá trị learning rate được thử nghiệm là 1e-3, 1e-4 và 1e-5, nhằm tìm ra mức độ học phù hợp nhất.

**Kết quả Tuning MLP:** Sau khi áp dụng `Grid Search` với tất cả các kết hợp có thể của các tham số trên, mô hình đạt được kết quả tối ưu với các giá trị siêu tham số sau:

- **Kích thước các lớp ẩn**: h1=512, h2=256, h3=128, h4=64, h5=32
- **Optimizer**: AdamW
- **Tỷ lệ dropout**: 0.3
- **Learning rate**: 1e-3

Kết quả này mang lại độ chính xác cao nhất trên tập kiểm thử, cho thấy rằng các siêu tham số này đã giúp mô hình học hiệu quả hơn và giảm thiểu hiện tượng *overfitting*.

*Transformer Model*

Đối với mô hình `Transformer`, chúng tôi cũng thực hiện điều chỉnh các siêu tham số để tối ưu hóa hiệu suất. Các tham số điều chỉnh bao gồm:

- **vocab_size**: Được tính từ tokenizer, xác định kích thước của từ điển từ vựng.
- **embed_dim**: Kích thước không gian embedding, thử nghiệm với giá trị 128.
- **num_heads**: Số lượng đầu attention trong mô hình, thử nghiệm với 8 đầu attention để mô hình có khả năng học các đặc trưng đa dạng từ dữ liệu.
- **hidden_dim**: Kích thước lớp ẩn, thử nghiệm với giá trị 512 để tăng khả năng biểu diễn của mô hình.
- **num_classes**: Được tính theo số lượng lớp trong bài toán phân loại (tính bằng `len(set(labels))`).
- **max_len**: Độ dài tối đa của chuỗi đầu vào, thử nghiệm với giá trị 256.
- **dropout_rate**: Tỷ lệ dropout được thử nghiệm là 0.3 nhằm giảm thiểu overfitting.

**Kết quả Tuning Transformer:** Sau khi điều chỉnh các siêu tham số, mô hình `Transformer` đạt được kết quả tốt với các tham số tối ưu như sau:

- **vocab_size**: len(tokenizer)
- **embed_dim**: 128
- **num_heads**: 8
- **hidden_dim**: 512
- **num_classes**: Số lượng lớp, tính bằng len(set(labels))
- **max_len**: 256
- **dropout_rate**: 0.3

Các tham số trên giúp mô hình `Transformer` đạt được hiệu suất tối ưu và giảm thiểu hiện tượng overfitting.

#### *3.3.1. Kết luận*

Việc điều chỉnh các siêu tham số là một bước quan trọng trong quá trình tối ưu hóa hiệu suất của các mô hình học máy. Cả hai mô hình `MLP` và `Transformer` đều đạt được kết quả tốt sau khi điều chỉnh các tham số chính, giúp cải thiện độ chính xác và khả năng tổng quát của mô hình.

## 4. Limitation and Future Work

### *4.1. Nhược điểm chung của các mô hình*

#### *4.1.1. Overfitting*

Một trong những nhược điểm lớn nhất của các mô hình là hiện tượng **overfitting**. Biểu hiện rõ nhất là độ chính xác trên tập huấn luyện (train accuracy) cao hơn nhiều so với tập kiểm định (validation/test accuracy). Ví dụ, với mô hình MLP, train accuracy đạt 94% trong khi validation accuracy chỉ đạt 84%. Điều này cho thấy mô hình học quá kỹ trên tập huấn luyện, nhưng lại kém khái quát khi đánh giá trên dữ liệu mới. Ngoài ra, validation loss có xu hướng dao động và không giảm đều đặn, đặc biệt rõ ràng trên các mô hình như MLP, GRU và BERT. Điều này có thể là dấu hiệu cho thấy mô hình chưa tối ưu tốt hoặc có thể cần phải điều chỉnh lại kiến trúc và các siêu tham số.

#### *4.1.2. Hiệu suất không đồng đều giữa các lớp*

Mặc dù tổng thể mô hình có thể đạt được độ chính xác cao, nhưng hiệu suất giữa các lớp lại không đồng đều. Một số lớp có F1-Score thấp hơn hẳn so với các lớp khác. Ví dụ, trong mô hình Transformer, các lớp như Class 0, Class 10 và Class 14 có F1-Score lần lượt là 0.76, 0.67 và 0.78, thấp hơn đáng kể so với các lớp còn lại. Nguyên nhân có thể do sự mất cân bằng dữ liệu giữa các lớp. Khi các lớp có số lượng mẫu nhỏ hơn, mô hình sẽ khó học được các đặc trưng quan trọng của chúng, dẫn đến hiệu suất kém hơn.

#### *4.1.3. Phức tạp và tốn tài nguyên*

Các mô hình lớn như Transformer và BERT có hàng triệu tham số, yêu cầu tài nguyên tính toán rất lớn (RAM, GPU) và thời gian huấn luyện dài. BERT với 12 encoder layers và 768 hidden dimensions là một trong những mô hình nặng nề nhất. Ngoài ra, các mô hình tuần tự như GRU cũng yêu cầu nhiều tài nguyên vì chúng phải xử lý lần lượt từng bước thời gian. Điều này làm cho thời gian huấn luyện và suy luận của GRU lâu hơn so với các mô hình không tuần tự như MLP và CNN.

#### *4.1.4. Giới hạn trong việc xử lý ngữ cảnh dài*

Một số mô hình có hạn chế trong việc nắm bắt ngữ cảnh dài. MLP chỉ học được các đặc trưng cục bộ và không tận dụng ngữ cảnh tuần tự. GRU và CNN có khả năng tốt hơn nhưng vẫn chủ yếu học được ngữ cảnh ngắn. Mặc dù Transformer và BERT được thiết kế để học ngữ cảnh dài, nhưng chúng yêu cầu bộ nhớ và thời gian tính toán lớn, đặc biệt khi chiều dài chuỗi tăng lên. Điều này khiến việc xử lý văn bản dài trở thành một thách thức lớn.

### *4.2. Định hướng cải thiện chung*

#### *4.2.1. Giảm Overfitting*

Để giảm overfitting, có thể áp dụng nhiều phương pháp như **regularization, early stopping và data augmentation**. Một cách đơn giản là tăng tỷ lệ **dropout** từ 0.5 lên 0.6 hoặc 0.7 cho các mô hình MLP, CNN, GRU và BERT. Ngoài ra, **weight decay** có thể được thêm vào optimizer (AdamW) để kiểm soát giá trị của các trọng số và ngăn chúng trở nên quá lớn. **Early stopping** là kỹ thuật phổ biến, ngừng huấn luyện khi validation loss không giảm trong một số epoch liên tiếp. Cuối cùng, sử dụng **data augmentation** (hoán đổi từ đồng nghĩa, xáo trộn từ) cũng giúp tăng tính đa dạng của dữ liệu huấn luyện và tránh overfitting.

#### *4.2.2. Giảm độ phức tạp của mô hình*

*Giảm số lớp ẩn và số lượng tham số.* Để giảm độ phức tạp, ta có thể giảm số lớp ẩn và số tham số trong mô hình. Đối với MLP, có thể giảm số lớp ẩn từ 5 xuống 3-4 lớp, giảm số neurons trong các lớp ẩn. Tương tự, trong GRU, số lớp có thể giảm từ 3 còn 2 lớp, và với BERT, có thể sử dụng các biến thể nhẹ hơn như **DistilBERT** hoặc **TinyBERT** để tiết kiệm tài nguyên và giảm thời gian huấn luyện.

#### *4.2.3. Tăng khả năng tổng quát hóa*

*Pooling cải tiến.* Để tăng khả năng tổng quát hóa của mô hình, ta có thể cải thiện cách lấy đặc trưng từ dữ liệu. Thay vì chỉ lấy `out[:, -1, :]` (kết quả từ bước thời gian cuối) từ GRU, ta có thể dùng **mean pooling** hoặc **attention pooling** để tận dụng tất cả các bước thời gian. Trong MLP và BERT, ta có thể kết hợp cả **mean pooling và max pooling** để lấy thông tin đa dạng hơn từ chuỗi

đặc trưng. Ngoài ra, thay GRU thành **Bidirectional GRU (BiGRU)** để học ngữ cảnh từ cả hai phía của chuỗi.

#### *4.2.4. Cân bằng dữ liệu và cải thiện các lớp yếu*

**Weighted CrossEntropy Loss** có thể được sử dụng để điều chỉnh trọng số cho từng lớp nhằm cải thiện hiệu suất của các lớp yếu. Ngoài ra, **oversampling** (tăng số mẫu của các lớp nhỏ) và **undersampling** (giảm lớp lớn hơn) có thể giúp dữ liệu cân bằng hơn. Việc tăng kích thước tập dữ liệu cũng sẽ cải thiện khả năng học của mô hình, đặc biệt đối với các lớp có số lượng mẫu nhỏ.

#### *4.2.5. Tối ưu hóa hyperparameters*

Việc tối ưu hóa các hyperparameters như **batch size, learning rate, dropout** sẽ giúp mô hình hoạt động tốt hơn. Sử dụng **ReduceLROnPlateau** để giảm learning rate động khi validation loss không cải thiện. Ta cũng có thể sử dụng **Cosine Annealing** để giảm learning rate từ từ theo thời gian. Tăng batch size để giảm nhiễu trong quá trình cập nhật gradient. Các kỹ thuật tìm kiếm như **Grid Search** và **Bayesian Optimization** có thể được áp dụng để tìm bộ hyperparameters tốt nhất.

### *4.3. Nhược điểm và định hướng cải thiện cụ thể theo mô hình*

#### *4.3.1. MLP (Multi-layer perceptron)*

*Nhược điểm.* MLP thường dễ bị **overfitting** vì không tận dụng ngữ cảnh tuần tự. Với 5 lớp ẩn, mô hình có thể trở nên phức tạp và cần nhiều tài nguyên.
*Định hướng cải thiện.* Giảm số lớp ẩn từ 5 xuống 3-4, tăng dropout từ 0.5 lên 0.7 và thêm weight decay để ngăn quá khớp.

#### *4.3.2. GRU*

*Nhược điểm.* GRU chậm hơn MLP do phải xử lý tuần tự, dễ bị **overfitting** và kém hiệu quả khi chỉ lấy `out[:, -1, :]` làm đặc trưng.
*Định hướng cải thiện.* Thay thế **GRU bằng BiGRU** và sử dụng mean pooling hoặc attention pooling để lấy đặc trưng.

#### *4.3.3. CNN (Convolutional Neural Network)*

*Nhược điểm.* CNN chủ yếu học đặc trưng cục bộ, không nắm bắt được ngữ cảnh dài.
*Định hướng cải thiện.* Tăng số lượng kernel size (3, 4, 5) để học được các đặc trưng đa dạng và kết hợp CNN với RNN (GRU hoặc LSTM) để học cả cục bộ và ngữ cảnh dài.

#### *4.3.4. Transformer*

*Nhược điểm.* Dễ bị **overfitting** do độ phức tạp cao và yêu cầu tài nguyên lớn.
*Định hướng cải thiện.* Dùng positional encoding động và sử dụng **attention pooling** thay vì mean pooling.

#### *4.3.5. BERT*

*Nhược điểm.* Mô hình nặng với 12 layers encoder và 768 dimensions, yêu cầu tài nguyên tính toán lớn.
*Định hướng cải thiện.* Thử DistilBERT, ALBERT, hoặc TinyBERT. Tăng batch size và sử dụng learning rate scheduler. Dùng weighted cross-entropy để giảm mất cân bằng dữ liệu. Kết hợp mean pooling và attention pooling từ các lớp ẩn thay vì chỉ dùng CLS token.

## 5. Conclusion

### *5.1. Kết luận tổng hợp*

Dựa trên các kết quả và nhận xét của các mô hình đã triển khai cho bài toán phân loại văn bản, có thể đưa ra các kết luận sau:

#### *5.1.1. Hiệu suất mô hình tổng quan*

BERT và các biến thể như BertForSequenceClassification thể hiện hiệu suất tốt nhất với Test Accuracy đạt 0.8663, vượt trội so với các mô hình khác (GRU, Transformer, MLP, CNN). Điều này chứng minh rằng các mô hình dựa trên transformer hiện đại có khả năng nắm bắt tốt ngữ cảnh và mối quan hệ trong văn bản.

GRUClassifier đạt hiệu suất khá cao với Test Accuracy là 0.843 và F1-Score là 0.8439, cho thấy khả năng xử lý tuần tự của mô hình GRU vẫn rất hiệu quả, đặc biệt khi cần xử lý ngữ cảnh vừa phải.

MLP đạt 0.841 Test Accuracy, gần ngang bằng với GRU nhưng có dấu hiệu overfitting nhẹ. Tuy nhiên, do MLP không có cơ chế xử lý chuỗi tuần tự, mô hình này thường kém hiệu quả hơn khi ngữ cảnh văn bản phức tạp.

Transformer Encoder và CNN có hiệu suất thấp hơn, với Test Accuracy dao động trong khoảng 0.81-0.82, cho thấy các mô hình này có hạn chế khi xử lý ngữ cảnh phức tạp hoặc dài hạn trong văn bản.

#### *5.1.2. Đề xuất lựa chọn mô hình*

Nếu hiệu suất cao là ưu tiên hàng đầu: BERT là lựa chọn tối ưu, với khả năng xử lý ngữ cảnh vượt trội và hiệu suất phân loại tốt nhất.

Cần tối ưu hóa thêm bằng cách sử dụng pooling đa lớp hoặc attention pooling để tận dụng toàn bộ đặc trưng của BERT.

Nếu tài nguyên tính toán hạn chế:

GRUClassifier là lựa chọn phù hợp, với hiệu suất chỉ thấp hơn một chút so với BERT nhưng chi phí tính toán thấp hơn đáng kể.

Nếu bài toán yêu cầu mô hình nhẹ hơn hoặc xử lý nhanh hơn:

Các biến thể của BERT như DistilBERT hoặc ALBERT có thể cân nhắc.

MLP có thể được dùng cho các bài toán đơn giản hoặc khi ngữ cảnh văn bản không phức tạp.

### *5.2. Tổng kết*

BERT thể hiện khả năng vượt trội trong bài toán phân loại văn bản, đặc biệt với các văn bản phức tạp hoặc chứa nhiều ngữ cảnh dài.

GRU Classifier là giải pháp thay thế phù hợp khi cần cân bằng giữa hiệu suất và chi phí tính toán. Các mô hình đơn giản hơn như MLP hoặc CNN phù hợp với bài toán cơ bản hơn nhưng không đủ mạnh trong các bài toán ngữ cảnh phức tạp.

Việc kết hợp các mô hình (Ensemble) hoặc thử nghiệm các kỹ thuật cải tiến như attention pooling, xử lý dữ liệu không cân bằng có thể giúp cải thiện hiệu suất tổng thể trong tương lai.