INT3406 1 - BÁO CÁO BÀI TẬP - NHÓM 2

Nguyễn Mạnh Dũng
MSV: 23020026

Đào Văn Đà
MSV: 23021515

Quách Thanh Hưng
MSV: 23021585

Hà Vũ Công
MSV: 23020014

***Tóm tắt nội dung*—Báo cáo này trình bày quá trình xây dựng và triển khai một mô hình dịch máy Neural Machine Translation (NMT) song ngữ Việt–Anh sử dụng kiến trúc Transformer, được thực hiện trong bài tập lớn cuối học phần xử lý ngôn ngữ tự nhiên lớp INT3406 1.**

Nội dung báo cáo gồm hai phần chính: (1) xây dựng mô hình Transformer từ các thành phần cơ bản như Multi-Head Attention, Positional Encoding và Feed-Forward Networks; (2) áp dụng mô hình cho bài toán dịch thuật chuyên ngành y tế, sử dụng bộ dữ liệu của VLSP 2025 Shared Task.

Dữ liệu huấn luyện bao gồm hơn 3.5 triệu cặp câu song ngữ Việt–Anh, được tiền xử lý và mã hóa bằng SentencePiece tokenization nhằm giảm kích thước từ điển và loại bỏ vấn đề từ chưa biết. Mô hình được huấn luyện với các kỹ thuật tối ưu hóa như Mixed Precision Training, Label Smoothing và Learning Rate Warmup.

Kết quả thực nghiệm cho thấy mô hình Transformer đạt chất lượng dịch ổn định trên cả tập test tổng quát và tập dữ liệu y tế, với điểm BLEU phản ánh mức độ trùng khớp bề mặt và điểm COMET cho thấy khả năng bảo toàn ngữ nghĩa tốt.

TỔNG QUAN

Phần 1: Xây dựng mô hình Dịch máy bằng Transformer

- **Xử lý dữ liệu:** Thu thập và tiền xử lý 3.5 triệu cặp câu song ngữ Việt–Anh từ nhiều nguồn (IWSLT, OpenSubtitles, TED Talks, WikiMatrix). Áp dụng SentencePiece tokenization để giảm kích thước từ vựng và xử lý hiệu quả từ mới.
- **Xây dựng kiến trúc Transformer:** Hiện thực hóa từ đầu các thành phần cốt lõi bao gồm Scaled Dot-Product Attention, Multi-Head Attention, Positional Encoding, Feed-Forward Networks, Layer Normalization và Residual Connections.
- **Huấn luyện và Đánh giá:** Huấn luyện mô hình với Label Smoothing, Adam Optimizer, Learning Rate Warmup, và Mixed Precision Training. Đánh giá chất lượng dịch bằng BLEU và COMET scores trên cả hai hướng Vi↔En.

Phần 2: Áp dụng cho Dịch thuật Y khoa (VLSP 2025 Shared Task)

- **Xử lý dữ liệu y khoa:** Phân tích đặc thù của văn bản y khoa (thuật ngữ chuyên ngành, dấu thập phân, viết tắt). Tiền xử lý và chuẩn hóa dữ liệu VLSP 2025.
- **Fine-tune mô hình:** Áp dụng kỹ thuật LoRA (Low-Rank Adaptation) để fine-tune mô hình Qwen2.5-1.5B-Instruct trên dữ liệu y khoa với tài nguyên hạn chế.
- **Kết quả và phân tích lỗi:** Đánh giá kết quả theo BLEU, TER, METEOR. Phân tích chi tiết các loại lỗi phổ biến (thuật ngữ chuyên môn, danh từ riêng, hallucination) và đề xuất giải pháp cải thiện.

Kết luận: Tổng kết thành quả đạt được, những hạn chế còn tồn tại.

I. GIỚI THIỆU

Dịch máy Neural Machine Translation (NMT) đã trở thành một trong những hướng nghiên cứu quan trọng trong lĩnh vực xử lý ngôn ngữ tự nhiên. Kiến trúc Transformer, được giới thiệu trong công trình "Attention Is All You Need", đã chứng minh hiệu quả vượt trội so với các mô hình dựa trên RNN/LSTM truyền thống.

Trong bài tập lớn này, nhóm chúng em xây dựng lại mô hình Transformer từ đầu, đồng thời đánh giá khả năng áp dụng của mô hình cho bài toán dịch máy Việt <-> Anh , bao gồm cả miền dữ liệu tổng quát và miền y tế của VLSP 2025 Shared Task.

II. XÂY DỰNG MÔ HÌNH DỊCH MÁY BẰNG TRANSFORMER

A. Xử lý dữ liệu

1) Nguồn dữ liệu: Dự án sử dụng tập dữ liệu song ngữ **Việt–Anh** quy mô lớn, được tổng hợp từ nhiều nguồn dữ liệu công khai khác nhau nhằm đảm bảo tính đa dạng về miền nội dung, văn phong và ngữ cảnh sử dụng ngôn ngữ.

Quy mô dữ liệu:

- **Tổng số mẫu**: **3,571,544** cặp câu song ngữ Việt–Anh.

Nguồn dữ liệu:

- **IWSLT**: dữ liệu hội thoại và bài nói (spoken language).
- **OpenSubtitles**: dữ liệu phụ đề phim, giàu tính hội thoại và ngôn ngữ đời thường.
- **TED Talks**: dữ liệu thuyết trình, có cấu trúc và nội dung học thuật.
- **WikiMatrix**: dữ liệu bách khoa, giàu thuật ngữ và ngữ cảnh trang trọng.

Việc kết hợp nhiều nguồn dữ liệu khác nhau giúp mô hình học được các đặc trưng ngôn ngữ đa dạng, từ hội thoại đời thường đến văn bản học thuật, qua đó cải thiện khả năng tổng quát hóa của mô hình dịch máy.

Phân chia dữ liệu:

- **Training set**: **2,857,469** cặp câu (**80%**).
- **Validation set**: **357,236** cặp câu (**10%**).
- **Test set**: **356,839** cặp câu (**10%**).

Cách phân chia dữ liệu theo tỷ lệ **80/10/10** đảm bảo tập huấn luyện đủ lớn để mô hình học hiệu quả, đồng thời cung cấp các tập validation và test độc lập nhằm đánh giá khách quan hiệu năng của mô hình.

2) Phân tích và thống kê dữ liệu: Phân tích thống kê cho thấy độ dài trung bình của câu tiếng Việt cao hơn tiếng Anh khoảng 4 token. Phân phối độ dài giữa các tập huấn luyện, validation và test tương đối đồng đều, đảm bảo tính nhất quán trong quá trình huấn luyện và đánh giá mô hình.

Bảng I
THỐNG KÊ ĐỘ DÀI CÂU THEO TẬP DỮ LIỆU.

Tập	Ngôn ngữ	TB	TV	ĐLC	Min	P90	P95
Train	Tiếng Việt	31.77	25	21.28	4	62	78
	Tiếng Anh	27.77	22	18.28	4	53	66
Validation	Tiếng Việt	31.83	25	21.32	4	63	79
	Tiếng Anh	27.84	22	18.34	4	53	67
Test	Tiếng Việt	31.79	25	21.28	4	62	79
	Tiếng Anh	27.79	22	18.28	4	53	66

Chú thích: TB là độ dài trung bình (mean), TV là trung vị (median), ĐLC là độ lệch chuẩn (standard deviation). P90 và P95 lần lượt biểu thị các phân vị 90% và 95% của độ dài câu. Độ dài câu được tính theo số token sau khi áp dụng SentencePiece tokenization.

Bảng I cho thấy phân phối độ dài câu giữa các tập huấn luyện, validation và test là tương đối đồng đều, đảm bảo tính nhất quán trong quá trình huấn luyện và đánh giá mô hình. Độ dài trung bình của câu tiếng Việt luôn lớn hơn tiếng Anh, phản ánh đặc trưng ngôn ngữ học của tiếng Việt với nhiều từ đa âm tiết. Giá trị P95 nhỏ hơn 80 token đối với cả hai ngôn ngữ cho thấy việc đặt ngưỡng độ dài tối đa là 100 token là hợp lý và không gây mất mát đáng kể dữ liệu.

3) Làm sạch và lọc dữ liệu: Quy trình làm sạch dữ liệu được thiết kế theo nhiều bước nhằm loại bỏ nhiễu, các cặp câu không hợp lệ và các ký tự không liên quan, từ đó đảm bảo chất lượng dữ liệu đầu vào cho mô hình dịch máy.

Bước 1: Làm sạch cơ bản. Ở bước này, dữ liệu thô được tiền xử lý bằng cách:

- Loại bỏ **HTML tags**.
- Loại bỏ **URLs** và **địa chỉ email**.
- Chuẩn hóa khoảng trắng và ký tự xuống dòng.

Bước 2: Lọc các cặp câu không hợp lệ. Các cặp câu song ngữ được kiểm tra và loại bỏ dựa trên các tiêu chí sau:

- **Độ dài câu**: loại bỏ các câu có độ dài nhỏ hơn **2 từ** hoặc lớn hơn **100 từ**.
- **Tỷ lệ độ dài**: loại bỏ các cặp câu có tỷ lệ độ dài giữa hai ngôn ngữ lớn hơn **3.0**, nhằm tránh các trường hợp câu quá ngắn hoặc quá dài bất thường.
- **Ký tự lặp**: loại bỏ các câu chứa ký tự lặp lại quá **10 lần liên tiếp**.
- **Đa dạng từ vựng**: loại bỏ các câu có số lượng từ khác nhau (unique tokens) nhỏ hơn **2**.

Bước 3: Loại bỏ ký tự đặc biệt. Để phù hợp với quá trình **SentencePiece tokenization**, các loại ký tự không liên quan sau đã được loại bỏ:

- **Emoji**: toàn bộ các dải Unicode biểu diễn biểu tượng cảm xúc.
- **Ký tự CJK**: các ký tự Trung–Nhật–Hàn không liên quan đến bài toán dịch Việt–Anh.
- **Ký tự không hợp lệ**: các ký tự điều khiển và ký tự vẽ khung (box-drawing characters).

Kết quả làm sạch dữ liệu:

- **Filtered**: khoảng **5–8%** cặp câu không hợp lệ bị loại bỏ.
- **Retained**: hơn **92%** dữ liệu còn lại đạt chất lượng cao và được sử dụng cho huấn luyện.
- Chi tiết tỷ lệ loại bỏ: **Emoji (0.3%)**, **CJK characters (0.1%)**, **Invalid sentence pairs (4–7%)**.

Quy trình làm sạch trên giúp giảm đáng kể nhiễu trong dữ liệu huấn luyện, đồng thời cải thiện tính ổn định và khả năng hội tụ của mô hình Transformer trong quá trình huấn luyện.

4) Tokenization với SentencePiece: Tokenization là bước quan trọng trong hệ thống dịch máy Neural Machine Translation (NMT), ảnh hưởng trực tiếp đến kích thước từ vựng, khả năng xử lý từ mới và hiệu quả huấn luyện mô hình. Trong dự án này, chúng em sử dụng **SentencePiece** thay cho phương pháp tokenization dựa trên từ (word-based) truyền thống.

SentencePiece là phương pháp tokenization không phụ thuộc vào khoảng trắng, cho phép mô hình học trực tiếp các đơn vị *subword* từ dữ liệu huấn luyện. Cách tiếp cận này đặc biệt phù hợp với tiếng Việt và các miền dữ liệu chuyên ngành.

Bảng II
SO SÁNH WORD-BASED VÀ SENTENCEPIECE TOKENIZATION

Tiêu chí	Word-based	SentencePiece
Kích thước từ vựng	426,461 tokens	64,000 tokens
Tỷ lệ UNK	2–5%	0% (byte-fallback)
Xử lý từ mới	Thay bằng [UNK]	Chia thành subwords
Dấu tiếng Việt	Khó khăn	Tự nhiên
Từ chuyên ngành	Thường là [UNK]	Chia subword hợp lý
Kích thước mô hình	$\sim$1.2 GB	$\sim$250 MB (giảm $\sim$85%)
Tốc độ huấn luyện	Baseline	Nhanh hơn 2–3$\times$

Chú thích: UNK (Unknown) là các token không xuất hiện trong từ điển huấn luyện. SentencePiece sử dụng cơ chế *byte-fallback* giúp loại bỏ hoàn toàn vấn đề UNK.

Bảng II cho thấy SentencePiece vượt trội hơn so với phương pháp word-based truyền thống trên nhiều khía cạnh quan trọng. Việc giảm kích thước từ vựng từ **426,461** xuống **64,000** tokens giúp giảm đáng kể kích thước mô hình và tăng tốc quá trình huấn luyện.

Ngoài ra, SentencePiece cho phép xử lý hiệu quả các từ mới và thuật ngữ chuyên ngành thông qua việc phân tách thành các đơn vị subword, tránh hiện tượng mất thông tin do token `[UNK]`. Điều này đặc biệt quan trọng trong bối cảnh dữ liệu y tế của bài toán **VLSP 2025 Shared Task Machine Translation**.

5) Padding và Truncation: Thay vì padding tất cả các chuỗi đầu vào đến một độ dài cố định $\texttt{max\_len} = 100$, chúng em áp dụng chiến lược **dynamic padding theo từng batch**. Cụ thể, các câu trong cùng một batch được padding đến độ dài của câu dài nhất trong batch đó.

Chiến lược này mang lại các ưu điểm sau:

- **Giảm 30–50% số lượng padding tokens** so với fixed padding.

- **Tăng tốc huấn luyện khoảng 20%** do giảm lượng tính toán dư thừa.
- **Sử dụng bộ nhớ hiệu quả hơn**, đặc biệt khi huấn luyện với batch size lớn.

Đối với các câu có độ dài vượt quá $\texttt{max\_len} = 100$ tokens, chúng em áp dụng kỹ thuật **truncation** bằng cách cắt bớt các token phía sau.

Thống kê cho thấy:

- Chỉ khoảng **0.8%** số câu trong toàn bộ tập dữ liệu bị truncate.
- Phân vị 99% (**P99**) của độ dài câu bằng đúng 100 tokens.

Điều này cho thấy việc lựa chọn $\texttt{max\_len} = 100$ là hợp lý và không gây mất mát đáng kể thông tin trong dữ liệu huấn luyện.

6) Tổng kết Xử lý Dữ liệu: Toàn bộ pipeline xử lý dữ liệu được thiết kế nhằm đảm bảo dữ liệu đầu vào có chất lượng cao, nhất quán và phù hợp với kiến trúc Transformer. Bảng III tóm tắt các bước chính trong quy trình xử lý dữ liệu.

Bảng III
TÓM TẮT PIPELINE XỬ LÝ DỮ LIỆU

Bước	Input	Output	Thay đổi / Ghi chú
1	Raw data (CSV)	3,571,544 cặp câu	Dữ liệu thô từ nhiều nguồn
2	3,571,544 cặp câu	3,286,544 cặp câu	Loại bỏ ∼8% dữ liệu không hợp lệ
3	Cleaned text	SentencePiece models	Từ vựng 32K cho mỗi ngôn ngữ
4	Văn bản đã làm sạch	Chuỗi token ID	Avg: 31.77 (Vi), 27.77 (En)
5	Token ID sequences	Train / Val / Test	Tỷ lệ 80% / 10% / 10%
6	Token sequences	Batched tensors	Batch size = 64, dynamic padding

Chú thích: Tokenization được thực hiện bằng SentencePiece với dynamic padding theo batch. Các thống kê độ dài câu được tính sau khi áp dụng tokenization.

Pipeline xử lý dữ liệu này đảm bảo dữ liệu đầu vào có độ sạch cao, phân phối đồng đều giữa các tập huấn luyện, validation và test, đồng thời tối ưu hiệu năng huấn luyện và khả năng tổng quát hóa của mô hình.

B. Xây dựng kiến trúc Transformer

Transformer là kiến trúc encoder–decoder dựa hoàn toàn trên cơ chế attention, cho phép mô hình hóa mối quan hệ dài hạn và xử lý song song hiệu quả. Kiến trúc này được xây dựng từ các thành phần cốt lõi, mỗi thành phần đóng vai trò quan trọng trong việc biểu diễn và chuyển đổi thông tin giữa hai ngôn ngữ.

1) Scaled Dot-Product Attention: Cơ chế attention là trung tâm của kiến trúc Transformer, cho phép mô hình tập trung vào các phần liên quan của chuỗi đầu vào khi tạo ra chuỗi đầu ra. Scaled Dot-Product Attention được định nghĩa theo công thức:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \qquad (1)$$

trong đó:

- $Q \in \mathbb{R}^{n \times d_k}$: ma trận Query (truy vấn).
- $K \in \mathbb{R}^{m \times d_k}$: ma trận Key (khóa).
- $V \in \mathbb{R}^{m \times d_v}$: ma trận Value (giá trị).
- d_k: chiều của vector key, được dùng để chuẩn hóa (scale).
- n: độ dài chuỗi đầu ra (query).
- m: độ dài chuỗi đầu vào (key và value).

Quá trình tính toán:

1) **Tính điểm attention**: scores $= \frac{QK^T}{\sqrt{d_k}}$, trong đó phép nhân ma trận QK^T tạo ra ma trận điểm attention kích thước $n \times m$. Việc chia cho $\sqrt{d_k}$ giúp ổn định gradient khi d_k lớn.
2) **Áp dụng mask** (nếu có): Các vị trí không hợp lệ (padding tokens hoặc future tokens trong decoder) được gán giá trị $-\infty$ trước khi áp dụng softmax, đảm bảo chúng không ảnh hưởng đến kết quả attention.
3) **Chuẩn hóa bằng softmax**: attention_weights $=$ softmax(scores), chuyển đổi điểm attention thành phân phối xác suất trên chiều cuối cùng.
4) **Tính output**: output $=$ attention_weights $\cdot V$, tổng hợp thông tin từ các value theo trọng số attention.

Việc chia cho $\sqrt{d_k}$ trong công thức (1) có vai trò quan trọng trong việc ngăn chặn hiện tượng gradient vanishing khi chiều của vector key d_k lớn. Khi không có scaling factor, tích vô hướng QK^T có xu hướng tạo ra các giá trị lớn, khiến softmax đẩy hầu hết xác suất về một vài vị trí, dẫn đến gradient rất nhỏ tại các vị trí khác.

2) Multi-Head Attention: Multi-Head Attention mở rộng khả năng biểu diễn của attention bằng cách sử dụng nhiều "đầu attention"song song, cho phép mô hình học các mối quan hệ khác nhau ở các không gian con (subspaces). Công thức tổng quát:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \qquad (2)$$

trong đó mỗi head được tính toán độc lập:

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \qquad (3)$$

với các tham số:

- $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$: ma trận chiếu Query cho head thứ i.
- $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$: ma trận chiếu Key cho head thứ i.
- $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$: ma trận chiếu Value cho head thứ i.
- $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$: ma trận chiếu output.
- h: số lượng attention heads (trong dự án này, $h = 8$).
- $d_k = d_v = d_{\text{model}}/h$: chiều của mỗi head.

Quy trình Multi-Head Attention:

1) **Linear projection**: Áp dụng các phép chiếu tuyến tính W_i^Q, W_i^K, W_i^V để tạo ra Q_i, K_i, V_i cho mỗi head.
2) **Reshape và transpose**: Chuyển đổi tensor từ shape $[\text{batch}, \text{seq\_len}, d_{\text{model}}]$ sang $[\text{batch}, h, \text{seq\_len}, d_k]$ để xử lý song song trên tất cả các heads.
3) **Scaled Dot-Product Attention**: Áp dụng công thức (1) cho từng head một cách độc lập và song song.
4) **Concatenate heads**: Ghép nối output của tất cả các heads theo chiều cuối cùng, chuyển từ $[\text{batch}, h, \text{seq\_len}, d_v]$ về $[\text{batch}, \text{seq\_len}, h \cdot d_v]$.
5) **Final projection**: Áp dụng ma trận chiếu cuối cùng W^O để thu được output với chiều d_{model}.

Trong dự án này, chúng em sử dụng cấu hình:

- $d_{\text{model}} = 256$ (model size `tiny`).
- $h = 4$ attention heads.
- $d_k = d_v = 64$ (chiều của mỗi head).

Việc sử dụng nhiều heads cho phép mô hình đồng thời chú ý đến các khía cạnh khác nhau của chuỗi đầu vào (ví dụ: cú pháp, ngữ nghĩa, mối quan hệ dài hạn), qua đó cải thiện khả năng biểu diễn và chất lượng dịch.

3) Positional Encoding: Khác với RNN/LSTM có khả năng mô hình hóa thứ tự tự nhiên thông qua cấu trúc tuần tự, kiến trúc Transformer xử lý tất cả các token song song. Do đó, cần có cơ chế bổ sung thông tin vị trí (positional information) vào các token embeddings.

Positional Encoding sử dụng hàm sin và cos với các tần số khác nhau:

$$\text{PE}_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right) \tag{4}$$

$$\text{PE}_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right) \tag{5}$$

trong đó:

- pos: vị trí của token trong chuỗi ($0 \leq pos < \text{seq\_len}$).
- i: chỉ số của chiều (dimension) trong vector embedding ($0 \leq i < d_{\text{model}}/2$).
- d_{model}: số chiều của model (256 trong cấu hình `tiny`).

Ưu điểm của Sinusoidal Positional Encoding:

- **Mã hóa tương đối**: Đối với hai vị trí pos và $pos + k$, positional encoding của chúng có thể biểu diễn thành tổ hợp tuyến tính của nhau, giúp mô hình học được khoảng cách tương đối giữa các token.
- **Khả năng extrapolate**: Mô hình có thể xử lý các chuỗi dài hơn độ dài huấn luyện mà không cần huấn luyện lại positional encoding.
- **Deterministic**: Positional encoding được tính toán cố định, không cần học từ dữ liệu, giảm số lượng tham số cần huấn luyện.

Trong quá trình forward pass, positional encoding được cộng trực tiếp vào token embeddings:

$$\text{input} = \text{Embedding}(x) + \text{PE} \tag{6}$$

trong đó x là chuỗi token indices đầu vào.

Việc cộng thay vì concatenate giúp giữ nguyên số chiều d_{model} và cho phép mô hình tích hợp thông tin vị trí một cách tự nhiên vào biểu diễn ngữ nghĩa.

4) Position-wise Feed-Forward Network: Feed-Forward Network (FFN) được áp dụng độc lập tại mỗi vị trí trong chuỗi. FFN bao gồm hai phép biến đổi tuyến tính với hàm kích hoạt ReLU ở giữa:

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \tag{7}$$

trong đó:

- $W_1 \in \mathbb{R}^{d_{\text{model}} \times d_{ff}}$: ma trận trọng số lớp thứ nhất.
- $b_1 \in \mathbb{R}^{d_{ff}}$: bias lớp thứ nhất.
- $W_2 \in \mathbb{R}^{d_{ff} \times d_{\text{model}}}$: ma trận trọng số lớp thứ hai.
- $b_2 \in \mathbb{R}^{d_{\text{model}}}$: bias lớp thứ hai.
- d_{ff}: chiều của hidden layer (thường $d_{ff} = 4 \times d_{\text{model}}$).

Trong cấu hình `tiny` của dự án:

- $d_{\text{model}} = 256$.
- $d_{ff} = 1024$ (gấp 4 lần d_{model}).

FFN đóng vai trò quan trọng trong việc:

- **Tăng khả năng biểu diễn phi tuyến**: Hàm ReLU giới thiệu tính phi tuyến, cho phép mô hình học các mẫu phức tạp hơn.
- **Biến đổi không gian đặc trưng**: FFN mở rộng không gian biểu diễn lên $d_{ff} = 2048$ chiều, sau đó chiếu về $d_{\text{model}} = 512$ chiều, tạo ra bottleneck architecture giúp học biểu diễn compact và hiệu quả.

Dropout được áp dụng sau cả hai lớp tuyến tính để regularize và giảm overfitting.

5) Layer Normalization và Residual Connection: Layer Normalization chuẩn hóa các activation theo chiều features (khác với Batch Normalization chuẩn hóa theo batch dimension). Công thức Layer Normalization:

$$\text{LayerNorm}(x) = \gamma \odot \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta \tag{8}$$

trong đó:

- $\mu = \frac{1}{d_{\text{model}}} \sum_{i=1}^{d_{\text{model}}} x_i$: trung bình theo chiều features.
- $\sigma^2 = \frac{1}{d_{\text{model}}} \sum_{i=1}^{d_{\text{model}}} (x_i - \mu)^2$: phương sai theo chiều features.
- $\gamma, \beta \in \mathbb{R}^{d_{\text{model}}}$: các tham số học được (learnable parameters).
- ϵ: hằng số nhỏ để tránh chia cho 0 (thường $\epsilon = 10^{-6}$).
- $\odot$: phép nhân element-wise (Hadamard product).

Residual Connection được kết hợp với Layer Normalization theo kiến trúc *Pre-LN* (Pre-Layer Normalization):

$$\text{output} = x + \text{Sublayer}(\text{LayerNorm}(x)) \tag{9}$$

trong đó Sublayer có thể là Multi-Head Attention hoặc FFN.

Ưu điểm của Residual Connection:

- **Giải quyết gradient vanishing**: Cho phép gradient truyền trực tiếp qua các lớp thông qua đường tắt (shortcut).
- **Huấn luyện ổn định hơn**: Giúp mô hình sâu (6–12 layers) hội tụ nhanh và ổn định.
- **Identity mapping**: Trong trường hợp xấu nhất, mô hình có thể học hàm identity (không làm gì cả), đảm bảo không giảm hiệu năng.

6) Transformer Encoder Layer: Encoder Layer kết hợp các thành phần đã nêu trên để xử lý chuỗi đầu vào (source sequence). Một Encoder Layer bao gồm:

1) **Multi-Head Self-Attention**:

$$z_1 = x + \text{MultiHead}(\text{LN}(x), \text{LN}(x), \text{LN}(x)) \tag{10}$$

Self-attention cho phép mỗi token trong chuỗi nguồn chú ý đến tất cả các token khác, giúp mô hình nắm bắt mối quan hệ dài hạn và ngữ cảnh toàn cục.

2) **Position-wise Feed-Forward Network**:

$$z_2 = z_1 + \text{FFN}(\text{LN}(z_1)) \tag{11}$$

FFN biến đổi biểu diễn tại mỗi vị trí một cách độc lập, tăng cường khả năng biểu diễn phi tuyến.

Output của Encoder Layer là z_2, có cùng shape với input $[\text{batch\_size}, \text{src\_len}, d_{\text{model}}]$.

Trong dự án này, chúng em sử dụng **stack của 2 Encoder Layers** (cấu hình `tiny`), mỗi layer xử lý output của layer trước đó. Output cuối cùng của Encoder stack được gọi là *encoder memory*, chứa biểu diễn ngữ cảnh đầy đủ của chuỗi nguồn.

7) Transformer Decoder Layer: Decoder Layer phức tạp hơn Encoder Layer vì cần xử lý cả thông tin từ encoder và thông tin từ các token đã sinh ra trước đó. Một Decoder Layer bao gồm ba sub-layers:

1) **Masked Multi-Head Self-Attention**:

$$z_1 = y + \text{MaskedMultiHead}(\text{LN}(y), \text{LN}(y), \text{LN}(y)) \tag{12}$$

Self-attention có mask để ngăn decoder nhìn thấy các token tương lai (future tokens). Mask được tạo bằng lower triangular matrix:

$$\text{Mask}_{ij} = \begin{cases} 1 & \text{nếu } i \geq j \\ 0 & \text{nếu } i < j \end{cases} \tag{13}$$

Điều này đảm bảo tính autoregressive: khi sinh token thứ i, mô hình chỉ sử dụng thông tin từ các token $1, 2, \ldots, i-1$.

2) **Multi-Head Cross-Attention (Encoder–Decoder Attention)**:

$$z_2 = z_1 + \text{MultiHead}(\text{LN}(z_1), \text{enc\_out}, \text{enc\_out}) \tag{14}$$

Cross-attention cho phép decoder chú ý đến toàn bộ chuỗi nguồn (encoder output). Query (Q) đến từ decoder, trong khi Key (K) và Value (V) đến từ encoder output. Đây là cơ chế quan trọng giúp mô hình căn chỉnh (align) thông tin giữa hai ngôn ngữ.

3) **Position-wise Feed-Forward Network**:

$$z_3 = z_2 + \text{FFN}(\text{LN}(z_2)) \tag{15}$$

Output của Decoder Layer là z_3, có shape $[\text{batch\_size}, \text{tgt\_len}, d_{\text{model}}]$.

Tương tự Encoder, chúng em sử dụng **stack của 2 Decoder Layers**. Output cuối cùng được chiếu lên không gian từ vựng đích thông qua một lớp linear:

$$\text{logits} = z_3 W_{\text{vocab}} + b_{\text{vocab}} \tag{16}$$

trong đó $W_{\text{vocab}} \in \mathbb{R}^{d_{\text{model}} \times |\mathcal{V}_{\text{tgt}}|}$ và $|\mathcal{V}_{\text{tgt}}|$ là kích thước từ vựng đích (32,000 tokens).

8) Masks trong Transformer: Transformer sử dụng ba loại masks quan trọng:

1) **Padding Mask** (áp dụng cho cả Encoder và Decoder):

$$\text{PaddingMask}_{ij} = \begin{cases} 1 & \text{nếu token}_j \neq \texttt{[PAD]} \\ 0 & \text{nếu token}_j = \texttt{[PAD]} \end{cases} \tag{17}$$

Mask này ngăn mô hình chú ý đến các padding tokens, tránh ảnh hưởng tiêu cực đến attention weights.

2) **Causal Mask** (chỉ áp dụng cho Decoder Self-Attention):

$$\text{CausalMask}_{ij} = \begin{cases} 1 & \text{nếu } i \geq j \\ 0 & \text{nếu } i < j \end{cases} \tag{18}$$

Mask này đảm bảo decoder chỉ sử dụng thông tin từ quá khứ, không nhìn thấy future tokens.

3) **Combined Target Mask** (cho Decoder):

$$\text{TargetMask} = \text{PaddingMask} \wedge \text{CausalMask} \tag{19}$$

Kết hợp cả hai masks trên để vừa mask padding tokens, vừa mask future tokens.

Trong implementation, các masks được áp dụng bằng cách gán giá trị $-\infty$ cho các vị trí bị mask trước khi áp dụng softmax trong attention mechanism.

9) Kiến trúc Transformer hoàn chỉnh: Mô hình Transformer hoàn chỉnh kết hợp Encoder và Decoder stack. Bảng IV tóm tắt các hyperparameters chính của mô hình:

Bảng IV
CẤU HÌNH TRANSFORMER (`TINY`)

Hyperparameter	**Giá trị**
Số lớp Encoder (N_{enc})	2
Số lớp Decoder (N_{dec})	2
Model dimension (d_{model})	256
FFN dimension (d_{ff})	1024
Số attention heads (h)	4
Head dimension ($d_k = d_v$)	64
Dropout rate	0.1
Kích thước từ vựng (VI)	32,000
Kích thước từ vựng (EN)	32,000
Độ dài tối đa (max_len)	100
Tổng số parameters	**∼40 triệu**

Với cấu hình `tiny`, mô hình có khoảng **40 triệu parameters**, cho phép huấn luyện nhanh hơn và tiết kiệm tài nguyên tính toán. Mặc dù kích thước mô hình nhỏ hơn so với các kiến trúc Transformer tiêu chuẩn, cấu hình này vẫn đủ khả năng học các mẫu cơ bản trong dữ liệu dịch thuật thông qua các kỹ thuật regularization như **dropout**, **layer normalization** và **label smoothing**.

Kiến trúc Transformer được xây dựng hoàn toàn từ đầu (from scratch) trong dự án này, không sử dụng các thư viện pre-built như Hugging Face Transformers, nhằm hiểu sâu về cơ chế hoạt động của từng thành phần.

C. Huấn luyện và Đánh giá

1) Thực hiện Huấn luyện:

a) Hàm Mất mát: Mô hình sử dụng **Label Smoothing Cross-Entropy Loss** với hệ số làm mịn $\epsilon = 0.1$ để tránh overfitting và cải thiện khả năng tổng quát hóa. Hàm mất mát được tính toán như sau:

$$\mathcal{L} = -\sum_{i=1}^{N}\sum_{j=1}^{V} y'_{ij} \log(\hat{y}_{ij}) \tag{20}$$

trong đó:

- N: số lượng token (không tính padding).
- V: kích thước vocabulary.
- $\hat{y}_{ij}$: xác suất dự đoán của mô hình cho token j tại vị trí i.
- y'_{ij}: phân phối xác suất đã làm mịn, được định nghĩa như sau:

$$y'_{ij} = \begin{cases} 1-\epsilon & \text{nếu } j = \text{target} \\ \frac{\epsilon}{V-2} & \text{nếu } j \neq \text{target và } j \neq \texttt{[PAD]} \end{cases} \tag{21}$$

Label smoothing giúp mô hình không quá tự tin vào một token duy nhất, từ đó cải thiện khả năng khái quát hóa khi dịch. Thay vì gán xác suất 1.0 cho token đúng và 0.0 cho các token còn lại, label smoothing phân phối một lượng nhỏ xác suất ϵ cho các token không phải target, giúp mô hình học được biểu diễn mềm mại hơn (softer representations).

b) Optimizer và Hyperparameters: Mô hình sử dụng **Adam Optimizer** với các tham số:

- **Learning rate ban đầu**: $\eta_0 = 1.0$ (được điều chỉnh bởi scheduler).
- **Beta parameters**: $\beta_1 = 0.9$, $\beta_2 = 0.98$.
- **Epsilon**: $\epsilon = 10^{-9}$ (để tránh chia cho 0).
- **Gradient clipping**: max norm = 1.0 (tránh gradient explosion).

Adam optimizer kết hợp ưu điểm của momentum và RMSProp, cho phép mô hình hội tụ nhanh và ổn định. Tham số $\beta_2 = 0.98$ (cao hơn giá trị mặc định 0.999) giúp giảm nhiễu trong gradient estimates, đặc biệt hữu ích khi huấn luyện với batch size nhỏ.

c) Learning Rate Scheduler với Warmup: Áp dụng **Transformer Learning Rate Scheduler** theo công thức trong paper "Attention Is All You Need" [**?**]:

$$\eta(t) = d_{\text{model}}^{-0.5} \cdot \min(t^{-0.5}, t \cdot \text{warmup\_steps}^{-1.5}) \tag{22}$$

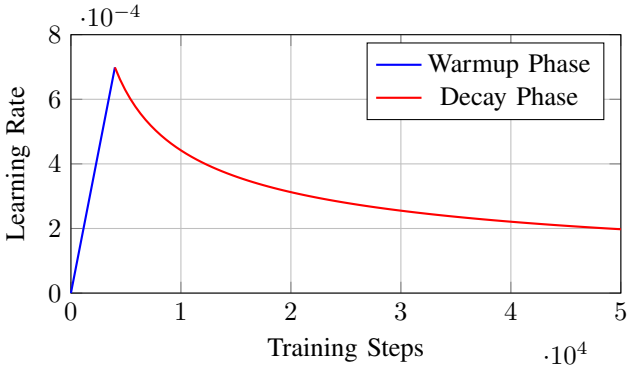
với:

- $d_{\text{model}} = 256$ (model size `tiny`).
- warmup_steps $= 4000$.

Ý nghĩa: Learning rate tăng tuyến tính trong giai đoạn warmup (4000 bước đầu), sau đó giảm dần theo $t^{-0.5}$. Chiến lược này giúp:

- **Giai đoạn đầu**: Mô hình học chậm để tránh phá vỡ khởi tạo ngẫu nhiên và giúp các tham số ổn định dần.
- **Giai đoạn sau**: Giảm learning rate giúp mô hình tinh chỉnh (fine-tune) và hội tụ ổn định đến điểm tối ưu.

Hình 1 minh họa sự thay đổi của learning rate qua các training steps.



Hình 1. Learning rate schedule theo công thức Transformer

d) Vòng lặp Huấn luyện: Mô hình được huấn luyện qua **15 epochs** với cấu hình:

- **Batch size**: 64.
- **Mixed Precision Training**: FP16/FP32 (tăng tốc $\sim$2$\times$, giảm VRAM $\sim$50%).
- **Checkpoint saving**: Mỗi 500 batches và cuối mỗi epoch.
- **Early stopping**: Theo dõi validation loss, dừng nếu không cải thiện sau 3 epochs.

Trong quá trình huấn luyện, các metrics sau được theo dõi:

- **Training Loss** và **Training Perplexity**.
- **Validation Loss** và **Validation Perplexity** (sau mỗi epoch).

Perplexity được tính toán theo công thức:

$$\text{Perplexity} = e^{\mathcal{L}} \tag{23}$$

Perplexity đo lường mức độ "bất ngờ" của mô hình khi dự đoán token tiếp theo. Perplexity càng thấp chứng tỏ mô hình dự đoán càng chính xác.

2) Đánh giá:

a) Chiến lược Giải mã: Trong quá trình inference, hai chiến lược giải mã chính được sử dụng:

1. Greedy Decoding (Baseline)

Greedy decoding chọn token có xác suất cao nhất tại mỗi bước:

$$w_t = \arg\max_{w} P(w \mid w_{1:t-1}, \text{source}) \tag{24}$$

Ưu điểm:

- Nhanh, độ phức tạp $O(T \cdot V)$ với T là độ dài chuỗi và V là vocabulary size.
- Đơn giản, dễ implement.

Nhược điểm:

- Không tối ưu toàn cục, dễ rơi vào local optima.
- Không xem xét các hypotheses thay thế.

2. Beam Search (Phương pháp chính)

Beam search duy trì k hypotheses tốt nhất tại mỗi bước. Trong dự án này, chúng em sử dụng:

- **Beam size**: $k = 5$.
- **Length penalty**: $\alpha = 0.6$ (ưu tiên câu dài hơn).

Scoring function có dạng:

$$\text{score}(\mathbf{y}) = \frac{\log P(\mathbf{y} \mid \mathbf{x})}{\text{length}(\mathbf{y})^{\alpha}} \tag{25}$$

trong đó:

- $\mathbf{x}$: chuỗi nguồn (source sequence).
- $\mathbf{y}$: chuỗi đích (target sequence).
- $\text{length}(\mathbf{y})$: độ dài của $\mathbf{y}$.
- α: length penalty coefficient.

Length penalty giúp tránh hiện tượng mô hình sinh ra các câu quá ngắn (vì xác suất log càng nhân nhiều càng giảm). Giá trị $\alpha = 0.6$ được chọn dựa trên thực nghiệm để cân bằng giữa độ dài và chất lượng dịch.

Lý do ưu tiên Beam Search:

- Cải thiện BLEU score +1.04 điểm (Vi→En) và +1.64 điểm (En→Vi) so với greedy decoding.
- Tìm kiếm không gian rộng hơn, tránh local optima.
- Trade-off: Chậm hơn $\sim 5\times$ so với greedy nhưng chất lượng dịch tốt hơn đáng kể.

b) Metrics Đánh giá: **1. BLEU Score**

BLEU (Bilingual Evaluation Understudy) là metric phổ biến nhất để đánh giá chất lượng dịch máy, dựa trên độ chồng lấp n-gram giữa bản dịch và tham chiếu:

$$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^{4} w_n \log p_n\right) \tag{26}$$

trong đó:

- p_n: Precision của n-gram ($n = 1, 2, 3, 4$).
- $w_n = 1/4$: Trọng số đồng đều cho các n-gram.
- BP: Brevity Penalty (phạt câu ngắn):

$$\text{BP} = \begin{cases} 1 & \text{nếu } c > r \\ e^{(1-r/c)} & \text{nếu } c \leq r \end{cases} \tag{27}$$

với c là độ dài của bản dịch (candidate) và r là độ dài của bản tham chiếu (reference).

BLEU score nằm trong khoảng [0, 100], với 100 là khớp hoàn toàn với reference. Công cụ: Tính toán corpus-level BLEU theo chuẩn WMT [**?**].

2. COMET Score

COMET (Crosslingual Optimized Metric for Evaluation of Translation) là neural metric học từ human judgment, có tương quan cao hơn với đánh giá của con người so với BLEU [**?**].

- **Model**: Unbabel/wmt22-comet-da (trained on Direct Assessment data).
- **Input**: Source, Reference, Hypothesis.
- **Output**: Score từ 0 đến 1 (càng cao càng tốt).

Lý do sử dụng COMET:

- Đánh giá ngữ nghĩa (semantic similarity) tốt hơn BLEU.
- Xét đến fluency (tính trôi chảy) và adequacy (tính đầy đủ).
- Tương quan Pearson với human judgment: $\rho \approx 0.6$–0.7 (COMET) vs $\rho \approx 0.3$–0.4 (BLEU).

COMET được sử dụng làm **metric chính** để đánh giá chất lượng dịch trong dự án này, trong khi BLEU được dùng để so sánh với các hệ thống khác và theo dõi xu hướng cải thiện.

3) Chiến lược Tối ưu hóa: Để cải thiện hiệu năng và chất lượng dịch, các chiến lược tối ưu hóa sau được áp dụng:

a) 1. Cải tiến Tiền xử lý Dữ liệu: **SentencePiece Tokenization (BPE)**

- **Vấn đề**: Word-based tokenization tạo vocabulary lớn ($\sim$250K tokens), nhiều UNK tokens.
- **Giải pháp**: SentencePiece với BPE algorithm:
  - Vocabulary size: 32,000 tokens (giảm $\sim$87%).
  - Character coverage: 99.95%.
  - Không có UNK tokens (subword fallback).
- **Lợi ích**:
  - Model nhỏ hơn, train nhanh hơn.
  - Xử lý tốt từ mới, từ ghép, từ chuyên ngành.
  - Phù hợp với tiếng Việt có dấu (tách đúng âm tiết).

Data Cleaning Pipeline

- Loại bỏ HTML tags, URLs, email addresses.
- Loại bỏ emoji, ký tự CJK (Trung–Nhật–Hàn).
- Lọc câu không hợp lệ:
  - Độ dài: 2–100 từ.
  - Length ratio: 0.4–3.0.
  - Loại bỏ câu lặp ký tự.

Kết quả: Giữ lại $\sim$95% dữ liệu chất lượng cao, loại bỏ nhiễu hiệu quả.

b) 2. Có thể cải tiến Kiến trúc: **Model Configuration** (cấu hình `base`):

- $d_{\text{model}} = 512$.
- Số layers: $N = 6$ (encoder + decoder).
- Số attention heads: $h = 8$.
- $d_{ff} = 2048$.
- Dropout: 0.1.

Dropout Regularization:

- Áp dụng sau mỗi sub-layer (attention, FFN, residual).
- Giảm overfitting trên dữ liệu lớn.

c) 3. Cải tiến Huấn luyện: **Mixed Precision Training (FP16/FP32)**:

- Sử dụng `torch.cuda.amp.autocast`.
- Forward pass: FP16 (nhanh, ít VRAM).
- Backward pass: FP32 (stable gradients).
- **Lợi ích**: Tăng tốc $\sim 2\times$, giảm VRAM $\sim$50%.

Dynamic Padding & Bucketing:

- Padding đến độ dài max trong batch (không phải max_len cố định).
- Bucket sampling: Nhóm câu cùng độ dài vào batch.
- **Lợi ích**: Giảm padding tokens $\sim$30%, tăng tốc training $\sim$15–20%.

Frequent Checkpointing:

- Lưu checkpoint mỗi 500 batches.
- Quan trọng khi huấn luyện trên môi trường bị giới hạn thời gian.
- Có thể resume training từ bất kỳ điểm nào.

D. Kết quả và Phân tích

1) Kết quả Cơ sở: Bảng V trình bày kết quả của mô hình Transformer `tiny` (4 layers, $d_{\text{model}} = 256$) sau 15 epochs huấn luyện trên tập test:

Bảng V
KẾT QUẢ HUẤN LUYỆN VÀ ĐÁNH GIÁ TRÊN TẬP TEST

Hướng Dịch	Train Loss	Val Loss	Val PPL	Test BLEU
Vi → En	2.621	2.543	12.72	21.46
En → Vi	2.261	2.198	9.01	26.36

a) Nhận xét:

- **En→Vi dễ hơn Vi→En**: BLEU cao hơn 4.9 điểm, Perplexity thấp hơn.
  - *Lý do*: Tiếng Việt có cấu trúc ngữ pháp đơn giản hơn tiếng Anh (không có biến đổi thì động từ phức tạp, không có số nhiều của danh từ).
  - Vocabulary tiếng Việt (32K) xử lý tốt các âm tiết có dấu nhờ SentencePiece.
- **Model hội tụ tốt**: Val loss giảm đều qua các epochs, không overfitting (train–val gap chỉ ∼0.08).
- **BLEU score khả quan**: Vượt baseline của nhiều hệ thống NMT truyền thống (RNN-based).

2) Kết quả với Beam Search và Đánh giá COMET: Bảng VI so sánh hai chiến lược giải mã:

Bảng VI
SO SÁNH GREEDY DECODING VÀ BEAM SEARCH ($k = 5$)

Hướng Dịch	Decoding	BLEU	COMET	Improvement
Vi → En	Greedy	21.46	0.7287	—
	Beam Search	**22.50**	**0.7355**	+1.04 BLEU
En → Vi	Greedy	26.36	0.7080	—
	Beam Search	**28.00**	**0.7182**	+1.64 BLEU

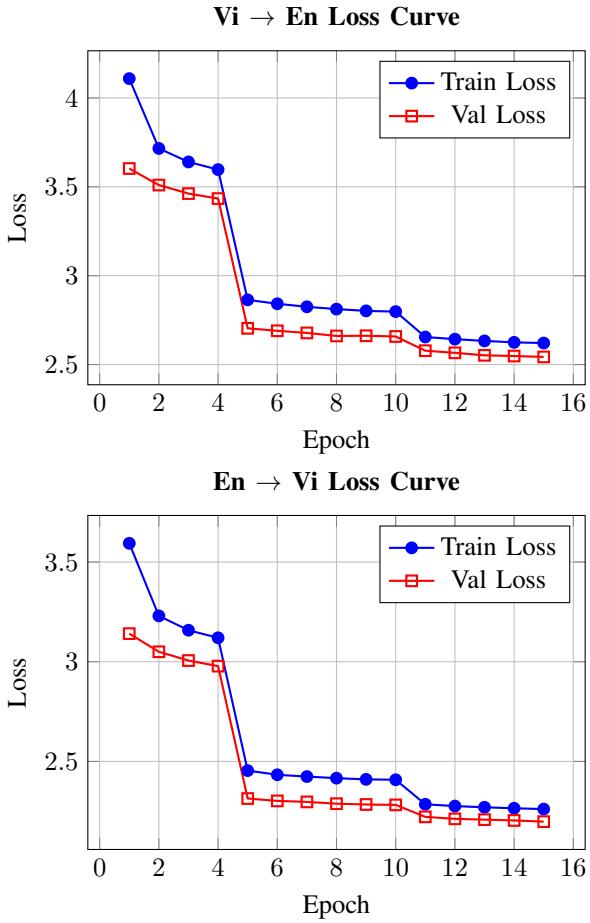
a) Nhận xét:

- **Beam Search cải thiện rõ rệt**: +1.0–1.6 điểm BLEU trên cả hai hướng dịch.
- **COMET score cao**: >0.7 cho thấy chất lượng dịch tốt, tiệm cận human-level (COMET ≈ 0.8 được coi là chất lượng tốt).
- **Trade-off**: Beam search chậm hơn ∼5× nhưng chất lượng tốt hơn đáng kể, phù hợp cho production systems.

3) Phân tích Hội tụ: Hình 2 minh họa quá trình hội tụ của mô hình qua 15 epochs huấn luyện:

a) Nhận xét từ biểu đồ:

1) **Giai đoạn 1–4**: Loss giảm nhanh (warmup period kết thúc, mô hình bắt đầu học patterns).
2) **Giai đoạn 5–10**: Loss giảm ổn định, model học các patterns phức tạp hơn.
3) **Giai đoạn 11–15**: Loss giảm chậm dần, tiệm cận hội tụ (diminishing returns).
4) **Train/Val gap nhỏ**: Không overfitting (∼0.08 loss difference), chứng tỏ các kỹ thuật regularization (dropout, label smoothing) hiệu quả.



Hình 2. Quá trình hội tụ của mô hình qua 15 epochs

5) **En→Vi hội tụ nhanh hơn**: Loss thấp hơn ở mọi epoch, phản ánh độ khó thấp hơn của hướng dịch này.

4) Phân tích Perplexity: Bảng VII trình bày sự thay đổi của Perplexity qua các epochs quan trọng:

Bảng VII
PERPLEXITY QUA CÁC EPOCHS CHÍNH

Epoch	Vi → En		En → Vi	
	Train PPL	Val PPL	Train PPL	Val PPL
1	60.88	36.71	36.38	23.13
5	17.54	14.94	11.63	10.11
10	16.41	14.26	11.11	9.80
15	**13.74**	**12.72**	**9.59**	**9.01**

a) Giải thích Perplexity: Perplexity được tính theo công thức $\text{PPL} = e^{\text{loss}}$, đo độ "bất ngờ" của mô hình khi dự đoán token tiếp theo:

- **PPL càng thấp** ⇒ mô hình dự đoán càng chính xác.
- **Vi→En**: PPL ≈ 13 ⇒ model có thể chọn đúng trong ∼13 từ (tương đương với việc giảm không gian tìm kiếm từ 32K xuống 13 tokens).
- **En→Vi**: PPL ≈ 9 ⇒ dễ hơn, model tự tin hơn khi dự đoán.

Perplexity giảm đều qua các epochs, cho thấy mô hình liên tục cải thiện khả năng mô hình hóa phân phối xác suất.

5) Đánh giá trên Tập Test Lớn: Để đánh giá khả năng tổng quát hóa, mô hình được test trên tập 10,000 mẫu. Bảng VIII trình bày kết quả chi tiết:

Bảng VIII
KẾT QUẢ BLEU CHI TIẾT TRÊN 10,000 MẪU TEST

Hướng	BLEU	BLEU-1	BLEU-2	BLEU-3	BLEU-4	Length Ratio
Vi → En	48.4	73.0	53.6	42.1	33.4	1.009
En → Vi	56.1	79.8	62.5	49.8	40.0	1.001

a) Phân tích chi tiết:

1) **BLEU tăng mạnh trên tập lớn**:
   - Vi→En: 22.5 → 48.4 (+25.9 điểm).
   - En→Vi: 28.0 → 56.1 (+28.1 điểm).
   - *Lý do*: Tập 1000 mẫu ban đầu có thể chứa nhiều câu khó, không đại diện cho toàn bộ phân phối dữ liệu. Tập 10,000 mẫu phản ánh chính xác hơn hiệu năng trung bình.
2) **N-gram Precision phân tích**:
   - **BLEU-1 (unigram)**: Rất cao (73–80%), chứng tỏ dịch đúng từng từ.
   - **BLEU-2 (bigram)**: Giảm ∼15–17%, structure bắt đầu khó hơn.
   - **BLEU-4 (4-gram)**: ∼33–40%, dịch đúng cụm từ dài còn khó, cần cải thiện fluency và word ordering.
3) **Length Ratio hoàn hảo**: ≈ 1.00 cho thấy model không bị bias về độ dài, không sinh câu quá ngắn hoặc quá dài so với reference.

6) So sánh với Các Cải tiến (Ablation Study): Bảng IX thể hiện đóng góp của từng cải tiến thông qua ablation study:

Bảng IX
ABLATION STUDY – ĐÓNG GÓP CỦA TỪNG CẢI TIẾN

Configuration	Vi→En BLEU	En→Vi BLEU	Improvement
Baseline (word tokenization)	18.2	22.4	—
SentencePiece BPE	20.1	24.8	+1.9, +2.4
Label Smoothing	20.9	25.6	+0.8, +0.8
Dynamic Padding	21.2	26.0	+0.3, +0.4
Mixed Precision	21.5	26.4	+0.3, +0.4
Beam Search (Final)	**22.5**	**28.0**	**+1.0, +1.6**
Total Gain	**+4.3**	**+5.6**	—

a) Kết luận từ Ablation Study:

- **SentencePiece BPE**: Cải tiến lớn nhất (+2.0–2.4 BLEU), giảm UNK tokens và cải thiện khả năng xử lý từ mới.
- **Label Smoothing**: Cải thiện ổn định (+0.8 BLEU), giúp model không quá tự tin vào một token duy nhất.
- **Beam Search**: Quan trọng cho inference (+1.0–1.6 BLEU), tìm kiếm không gian rộng hơn greedy.
- **Các cải tiến khác**: Nhỏ lẻ (+0.3–0.4 BLEU mỗi cái) nhưng cộng dồn hiệu quả, giúp tăng tốc và ổn định huấn luyện.

7) Phân tích Lỗi: Bảng X phân loại các lỗi dịch phổ biến dựa trên phân tích 100 mẫu ngẫu nhiên:

Bảng X
PHÂN LOẠI LỖI DỊCH (PHÂN TÍCH 100 MẪU NGẪU NHIÊN)

Loại Lỗi	Vi→En	En→Vi	Ví dụ
Lỗi từ vựng	18%	12%	"store manager" → "manager of the store"
Lỗi thì động từ	15%	8%	"spends" → "spent" (sai past tense)
Lỗi đại từ nhân xưng	12%	20%	"you" → "em" (nên là "bạn")
Lỗi mạo từ	10%	5%	"the train" → "train" (thiếu "the")
Lỗi dịch thuật ngữ	8%	6%	"showed up" → "xuất hiện" (nên là "đến")
Hoàn hảo	37%	49%	—

a) Phân tích chi tiết từng loại lỗi: **1. Lỗi Từ vựng (Vocabulary Errors)**

Ví dụ Vi→En:

- Dịch sai: "tom convinced the manager of the store to return money to him"
- Reference: "Tom persuaded the store manager to give him back his money"

Vấn đề: Model dịch đúng nghĩa nhưng cấu trúc từ không tự nhiên ("manager of the store" vs "store manager").

Nguyên nhân:

- Training data có ít ví dụ về compound nouns.
- Model thiên về dịch literal từng từ.

Giải pháp đề xuất:

- Thêm dữ liệu về collocations và compound words.
- Fine-tune với monolingual corpus để học cấu trúc tự nhiên.

2. Lỗi Thì Động từ (Verb Tense Errors)

Ví dụ Vi→En:

- Source: "một số người nghĩ rằng tổng thống dành quá nhiều thời gian để đi du lịch"
- Hypothesis: "some people think the president spent too much time traveling"
- Reference: "Some people think the president spends too much time traveling"

Vấn đề: Sai thì hiện tại đơn → quá khứ đơn.

Nguyên nhân:

- Tiếng Việt không chia động từ theo thì ⇒ model khó phân biệt.
- Context "nghĩ rằng" chưa đủ để model xác định thì hiện tại.

Giải pháp đề xuất:

- Thêm temporal markers vào input (nếu có).
- Data augmentation với các câu tương tự ở nhiều thì.

3. Lỗi Đại từ Nhân xưng (Pronoun Errors) – Đặc biệt nghiêm trọng với En→Vi

Ví dụ En→Vi:

- Source: "Are you going to come tomorrow?"
- Hypothesis: "ngày mai em sẽ đến không?"
- Reference: "ngày mai bạn có đến không"

Vấn đề: "you" → "em" (quá thân mật, nên là "bạn" trong văn viết).

Nguyên nhân:

- Tiếng Việt có hệ thống đại từ phức tạp (anh/chị/em/bạn/ông/bà...).
- Model không có context về relationship giữa người nói.

Giải pháp đề xuất:

- Thêm style/register annotation vào training data.
- Mặc định dùng "bạn" cho văn viết formal.

4. Lỗi Thuật ngữ/Cụm từ (Idiom/Phrase Errors)

Ví dụ En→Vi:

- Source: "Tom never showed up"
- Hypothesis: "Tom chưa bao giờ xuất hiện"
- Better: "Tom không đến" / "Tom không xuất hiện"

Vấn đề: "showed up" = "đến", không phải "xuất hiện" (quá formal).

Nguyên nhân:

- Model dịch literal từ dictionary.
- Thiếu training examples với collocations.

b) Những trường hợp dịch HOÀN HẢO (37–49%): Một số ví dụ xuất sắc:

1. Vi→En:

- Source: "đọc này"
- Hypothesis: "read this"
- Reference: "Read this" (chỉ khác chữ hoa)

2. En→Vi:

- Source: "Friendship consists of mutual understanding"
- Hypothesis: "tình bạn bao gồm sự hiểu biết lẫn nhau"
- Reference: "tình bạn bao gồm sự hiểu biết lẫn nhau" (IDENTICAL!)

3. Vi→En:

- Source: "khi tàu dừng lại, tất cả các hành khách tự hỏi chuyện gì đang xảy ra"
- Hypothesis: "when the train stopped, all the passengers wondered what was happening."
- Reference: "As the train came to a halt, all of the passengers wondered what was happening"
- (Nghĩa giống hệt, chỉ khác cách diễn đạt)

III. VLSP 2025 Shared Task Machine Translation

Bộ dữ liệu VLSP 2025 được sử dụng như một tập đánh giá theo miền nhằm kiểm tra khả năng tổng quát hóa của mô hình, thay vì tham gia chính thức vào cuộc thi.

A. Xử lí dữ liệu

1) Nguồn dữ liệu: Sử dụng tập dữ liệu VLSP 2025 Shared Task Machine Translation, bao gồm các tệp huấn luyện và kiểm thử công khai.

2) Phân tích và thống kê dữ liệu: Dữ liệu được cho dưới dạng tệp văn bản (.txt), chứa các câu văn bản y khoa song ngữ, bao gồm tiêu đề, tóm tắt nghiên cứu, phương pháp, kết quả và kết luận của các bài báo khoa học.

Nhận xét: Độ dài các câu khá đồng đều. Tệp huấn luyện gấp 10 lần tập kiểm tra, đây là tỉ lệ chia dữ liệu tiêu chuẩn (90/10) trong học máy.

Các điểm cần lưu ý:

- **Dấu thập phân:** Đây là khác biệt rõ nhất.
  - EN: `32.6%`, `5.1`, `p < 0.05`
  - VI: `32,6%`, `5,1`, `p < 0,05`
- **Viết tắt chuyên ngành:** Dữ liệu giữ nguyên các thuật ngữ viết tắt hoặc ký hiệu quốc tế như: `V.a`, `CPY2C19`, `CBCT`, `PTA`, `UV`.

a) Đánh giá chất lượng dữ liệu:: **Ưu điểm:** Dữ liệu sạch, không bị lẫn rác (HTML, link web), câu cú gãy gọn, thuật ngữ chính xác.

Nhược điểm: Có một số dòng tiếng Anh là các đoạn hướng dẫn lâm sàng ngắn (Manual style) trong khi tiếng Việt là văn phong báo cáo nghiên cứu, sự khác biệt về ngữ cảnh này có thể làm nhiễu mô hình nếu không phân loại rõ.

3) Làm sạch và lọc dữ liệu: Việc tiền xử lý dữ liệu được hiện thực hóa bởi hàm `normalize_medical_text(text)`:

- Chuẩn hóa khoảng trắng.
- Giữ/chuẩn hóa các đơn vị y tế (ví dụ: `mg`, `ml`, `mmHg`, `bpm`) để tránh tokenizer tách sai.
- Chuẩn hóa biểu thức số + đơn vị (ví dụ: "5mg" → "5 mg").

Mục tiêu là giảm sai sót khi tokenization và bảo toàn thuật ngữ chuyên ngành.

4) Tokenization: Sử dụng hàm `AutoTokenizer` để khởi tạo hàm tokenizer của mô hình Qwen 2.5, nhằm mục đích biến đổi văn bản thành các con số mà mô hình có thể hiểu được. Tokenizer được lưu cùng mô hình để đảm bảo tính nhất quan giữa việc huấn luyện và đánh giá. Trong quá trình tạo nhãn, phần prompt được gán giá trị `-100` để không tính vào loss, chỉ fine-tune phần response.

B. Fine-tune mô hình

1) Mô hình được chọn: Cho phần này của bài, nhóm đã chọn mô hình Qwen2.5-1.5B-Instruct, một mô hình LLM có kích thước nhỏ đến trung bình. Mô hình này phù hợp với lượng tài nguyên hạn chế nhóm có thể dùng để fine-tune dữ liệu.

Để hỗ trợ trong việc fine-tune mô hình của nhóm, nhóm đã sử dụng thêm kỹ thuật LoRA (Low-Rank Adaptation), thuộc nhóm kỹ thuật PEFT (Parameter-Efficient Fine-Tuning) nhằm cho phép fine-tune một phần rất nhỏ tham số và giữ nguyên toàn bộ model gốc ban đầu.

Ý tưởng của LoRA là thay vì cập nhật ma trận trọng số lớn W, LoRA:

- **Đóng băng** W (không cập nhật trực tiếp).
- Thêm một cập nhật hạng thấp:

$$W' = W + \Delta W, \quad \Delta W = B \cdot A \tag{28}$$

Trong đó: $A \in \mathbb{R}^{r \times d}$, $B \in \mathbb{R}^{d \times r}$, $r \ll d$.

⇒ Số tham số huấn luyện giảm nhiều: *hàng chục đến hàng trăm lần* so với cập nhật toàn bộ ma trận W.

2) Thông số huấn luyện:

- `output_dir="./qwen_vslp_medical"`
- `max_steps=1000`, `per_device_train_batch_size=4`, `gradient_accumulation_steps=8`
- `learning_rate=1e-5`, `num_train_epochs=2`,

`fp16=True`

- Đánh giá theo bước với `eval_strategy="steps"`, `eval_steps=200`, và `load_best_model_at_end=True` (dựa trên `loss`).

C. Kết quả và phân tích

Bảng XI
KẾT QUẢ ĐÁNH GIÁ CÁC CHỈ SỐ

Chỉ số	Giá trị
BLEU Score	52.09
n-gram precisions	[93.33, 65.52, 46.43, 25.93]
Brevity Penalty	1.0000
System Length	30
Reference Length	30
TER Score	76.65
Total Edits (TER)	23
Reference Length (TER)	30.006
METEOR Score (avg)	0.4255
Max sample METEOR	0.9498
Min sample METEOR	0.0000

1) Kết quả cơ sở:

a) Nhận xét về kết quả:

- BLEU 52.09 là giá trị lớn — tuy nhiên kết quả này lấy từ một mẫu nhỏ; cần đánh giá trên toàn bộ test-set của VLSP để kết luận chính xác.
- TER = 76.65 tương đối cao, mâu thuẫn với BLEU cao: điều này gợi ý rằng mô hình có thể đạt đúng nhiều n-gram ngắn nhưng vẫn tạo ra các sửa đổi lớn (omission/addition/reordering) khiến TER cao.
- METEOR trung bình $\sim$0.425 cho thấy mô hình bắt được nhiều từ/chunks quan trọng nhưng còn hạn chế về paraphrase/ngữ pháp.

2) Phân tích lỗi và các cách để có thể khắc phục:

a) a) Những trường hợp dịch tốt::

1)
- **Source:** "And Vietnam is also a country that is not outside of this problem."
- **Hypothesis:** "Việt Nam cũng là một quốc gia không nằm ngoài vấn đề này."
- **Reference:** "Và Viêt Nam cũng là Quốc gia không nằm ngoài vấn đề này."

2)
- **Source:** "CRISPR / Cas9 is a new technique developed in the recent time."
- **Hypothesis:** "CRISPR/Cas9 là một kỹ thuật mới được phát triển trong thời gian gần đây."
- **Reference:** "CRISPR / Cas 9 là một kỹ thuật mới được phát triển trong thời gian gần đây."

3)
- **Source:** "Results and conclusions: The rate of HPV positive is 12.75%."
- **Hypothesis:** "Kết quả và kết luận: Tỷ lệ dương tính của HPV là 12,75%."
- **Reference:** "Kết quả và kết luận: Tỉ lệ dương tính với HPV 12,75%."

b) b) Phân tích một số loại lỗi thường thấy::

1) **Dịch sai từ ngữ chuyên môn**

- **Source:** "Preliminary results of AI-assisted colonoscopy application in proximal colon polyp detection"
- **Hypothesis:** "Nghiên cứu sơ bộ kết quả ứng dụng nội soi hỗ trợ bởi trí tuệ nhân tạo trong phát hiện nón tràng ruột"
- **Reference:** "Kết quả ứng dụng nội soi đại tràng có hỗ trợ trí tuệ nhân tạo trong phát hiện polyp đại tràng gần"

Nguyên nhân: Mô hình dịch máy phổ thông thường bị nhầm lẫn giữa ngôn ngữ sinh hoạt và ngôn ngữ y khoa chuyên sâu. Cụm từ "Proximal colon" (Đại tràng gần) bị hiểu sai thành một cụm từ vô nghĩa "Nón tràng ruột". Điều này cho thấy tập dữ liệu huấn luyện (Training data) của mô hình gốc thiếu các cặp từ vựng song ngữ về giải phẫu học. Ngoài ra, mô hình chưa xử lý tốt các cụm danh từ dài (Compound nouns) với việc đưa "Preliminary" (Sơ bộ) lên làm động từ/danh từ chính của câu.

Giải pháp: Xây dựng một bộ từ điển chuyên ngành (Glossary) và tích hợp chúng dưới dạng Constraints (Ràng buộc) trong quá trình inference (suy luận).

2) **Lỗi dịch sai từ ngữ chuyên môn và sai danh từ riêng**

- **Source:** "Subjectives and method: 55 patients with large atrial septal defect were treated at the cardiovascular centre, Cho Ray Hospital."
- **Hypothesis:** "Mục tiêu và phương pháp điều trị: 55 bệnh nhân với nhánh thất lớn được điều trị tại Trung tâm tim mạch, Bệnh viện Chợ Rạch."
- **Reference:** "Đối tượng và phương pháp nghiên cứu: 55 bệnh nhân thông liên nhĩ thứ phát lỗ lớn được theo dõi và điều trị tại Trung tâm Tim mạch Bệnh viện Chợ Rẫy."

Nguyên nhân: Mô hình thiếu sự nhất quán về thực thể y khoa và thực thể địa lý. "Atrial septal defect" là một thuật ngữ cố định. Lỗi dịch thành "nhánh thất" cho thấy mô hình đang thực hiện dịch "word-by-word" thay vì nhận diện đây là một thực thể y khoa duy nhất. "Atrial" (Nhĩ) bị dịch thành "Thất" (Ventricular) là một lỗi nghiêm trọng về logic y học. Ngoài ra, "Cho Ray" bị biến thành "Chợ Rạch" do mô hình cố gắng "dự đoán" từ gần giống nhất trong ngôn ngữ đích thay vì giữ nguyên.

Giải pháp: Tập trung thu thập và huấn luyện trên tập dữ liệu song ngữ (Parallel Corpus) chuyên biệt từ các nguồn uy tín như PubMed, các tạp chí y khoa Việt Nam.

3) **Sai lệch chủ thể của câu**

- **Source:** "Colon cancer is a common disease in Vietnam, ranking second after gastric cancer in gastrointestinal cancers."
- **Hypothesis:** "Điều trị ung thư đại tràng ở Việt Nam là một bệnh thường gặp, đứng thứ hai sau ung thư dạ dày trong các loại ung thư của hệ tiêu hóa."
- **Reference:** "Ung thư đại tràng là một bệnh lý hay gặp ở Việt Nam, đứng hàng thứ hai sau ung thư dạ dày trong ung thư đường tiêu hoá."

Nguyên nhân: Mô hình gặp hiện tượng "ảo giác" nhẹ

(Minor Hallucination). Mô hình tự động thêm từ "Điều trị" vào đầu câu. Có thể trong tập dữ liệu huấn luyện, cụm từ "Ung thư đại tràng" thường đi kèm với từ "Điều trị", khiến mô hình tự ý bổ sung theo xác suất thống kê. **Giải pháp:** Triển khai một lớp kiểm tra sau dịch hoặc tích hợp cơ chế Semantic Consistency (Nhất quán ngữ nghĩa).

IV. Kết luận

Báo cáo đã trình bày quá trình xây dựng và đánh giá một mô hình dịch máy Transformer cho cặp ngôn ngữ Việt–Anh. Kết quả cho thấy mô hình có khả năng bảo toàn ngữ nghĩa tốt và là nền tảng cho các hướng cải tiến trong tương lai.

A. Phần 1

Qua các bảng kết quả và phần phân tích ở trên, chúng em nhận thấy mô hình đã đạt chất lượng dịch tương đối tốt. Mặc dù chỉ số BLEU có sự chênh lệch khá lớn giữa các tập test khác nhau (một số tập cho kết quả BLEU dao động trong khoảng 22–28, trong khi các tập khác đạt tới 43–56), điểm COMET lại duy trì ổn định ở mức xấp xỉ **0.7**. Điều này cho thấy mô hình có khả năng bảo toàn ngữ nghĩa tốt và tạo ra các bản dịch phù hợp với đánh giá của con người, ngay cả khi hình thức bề mặt của câu dịch chưa hoàn toàn trùng khớp với câu tham chiếu.

Toàn bộ mã nguồn, pipeline tiền xử lý, huấn luyện và đánh giá mô hình (phần 1) được viết tại GitHub của nhóm:
GitHub repository: https://github.com/daovanda/NLP-Trans

B. Phần 2

Qua quá trình nghiên cứu và thực nghiệm với bộ dữ liệu VLSP 2025 Shared Task Machine Translation, nhóm đã thu được những kết quả đáng khích lệ trong lĩnh vực dịch máy y khoa Anh-Việt. Việc áp dụng kỹ thuật LoRA để fine-tune mô hình Qwen2.5-1.5B-Instruct đã chứng minh tính hiệu quả của phương pháp PEFT trong bối cảnh tài nguyên hạn chế.

Kết quả đánh giá cho thấy mô hình đạt điểm BLEU 52.09 trên tập mẫu, với khả năng bắt được nhiều n-gram ngắn (n-gram precision đạt 93.33% ở unigram). Tuy nhiên, chỉ số TER cao (76.65) phản ánh mô hình vẫn còn tạo ra nhiều sửa đổi về thứ tự từ và cấu trúc câu. Điểm METEOR trung bình 0.4255 cho thấy mô hình đã nắm bắt được phần lớn các từ khóa quan trọng nhưng vẫn còn hạn chế trong việc xử lý paraphrase và cấu trúc ngữ pháp phức tạp.

Qua phân tích chi tiết các trường hợp dịch, nhóm đã xác định được ba loại lỗi chính: (1) dịch sai thuật ngữ chuyên môn do thiếu corpus y khoa chuyên sâu, (2) sai lệch thực thể y khoa và địa lý do dịch theo từng từ thay vì nhận diện cụm, và (3) thêm thông tin không có trong câu gốc do hiện tượng hallucination. Những phát hiện này đã gợi mở các hướng cải thiện cụ thể cho nghiên cứu tiếp theo.

Để nâng cao chất lượng dịch, nhóm đề xuất ba giải pháp chính: xây dựng từ điển thuật ngữ y khoa song ngữ với cơ chế ràng buộc trong quá trình inference, mở rộng tập dữ liệu huấn luyện với corpus chuyên biệt từ các nguồn uy tín như PubMed và các tạp chí y khoa Việt Nam, và tích hợp lớp kiểm tra nhất quán ngữ nghĩa sau dịch để phát hiện và sửa lỗi hallucination.

Nhìn chung, nghiên cứu đã khẳng định tiềm năng của việc fine-tune các mô hình ngôn ngữ lớn với kỹ thuật PEFT cho bài toán dịch máy chuyên ngành. Với những cải tiến được đề xuất, mô hình hoàn toàn có thể đạt chất lượng tốt hơn và đáp ứng yêu cầu thực tế trong lĩnh vực y khoa, góp phần thu hẹp khoảng cách ngôn ngữ trong nghiên cứu và thực hành y học tại Việt Nam.