VIETNAM NATIONAL UNIVERSITY, HANOI
UNIVERSITY OF ENGINEERING AND TECHNOLOGY

Trinh Huu Tan
Nguyen Trung Dung
Nguyen Duc Anh

End Of Term Assignment

Neural Machine Translation

HA NOI - 2023

VIETNAM NATIONAL UNIVERSITY, HANOI
UNIVERSITY OF ENGINEERING AND TECHNOLOGY

Trinh Huu Tan
Nguyen Trung Dung
Nguyen Duc Anh

End Of Term Assignment

Neural Machine Translation

Supervisor: Dr. Nguyen Van Vinh,
Msc. Nguyen Van Minh Cong

HA NOI - 2023

ACKNOWLEDGMENTS

We would like to extend our sincerest appreciation and gratitude to our supervisors, Dr. Nguyen Van Vinh and MSc. Nguyen Van Minh Cong, for their unwavering support and guidance throughout this research project. Their expert knowledge, patient guidance, and enthusiastic encouragement have been invaluable to our work, and we are deeply grateful for their generous contributions of time and effort.

ABSTRACT

In this report, we present our approach to fine-tuning the T5-base model for neural machine translation (NMT) from English to Vietnamese. Our goal was to develop a high-quality NMT model capable of accurately translating English sentences into Vietnamese with a high degree of fidelity. To achieve this goal, we used the iwslt_en_vi dataset and fine-tuned the T5-base model, which is a powerful pre-trained language model that has been shown to perform well on a variety of NLP tasks. Our training process involved several key steps, including data preprocessing, parameter tuning, model initialization, batch processing, and performance evaluation. We used the T5 tokenizer to tokenize and encode the source and target sentences, and formatted the data into input-target pairs. We then fine-tuned the T5-base model using the encoded input-target pairs, adjusting the hyperparameters to optimize the model's performance on the validation set. Our experimental results showed that the fine-tuned T5-base model achieved a BLEU score of 0.278 on the iwslt_en_vi test set. While the BLEU score may not be considered significant, it should be noted that the iwslt_en_vi dataset is a small and challenging dataset, and our results demonstrate the potential effectiveness of fine-tuning the T5-base model for this task. Further experimentation with larger and more diverse datasets may yield more significant improvements. Our results demonstrate the effectiveness of fine-tuning the T5-base model for English-to-Vietnamese machine translation.

Contents

CHAPTER 1. INTRODUCTION

1.1 Motivation

Neural Machine Translation (NMT) has gained significant attention in recent years due to its potential to revolutionize the field of machine translation. Traditional rule-based and statistical machine translation methods have shown limitations in capturing complex language structures, handling long-distance dependencies, and producing fluent and accurate translations. These limitations have motivated researchers to explore and advance the field of NMT.

One of the primary motivations for studying NMT is to improve translation quality. By leveraging the power of neural networks, NMT models have shown promising results in achieving higher translation accuracy, fluency, and adequacy compared to traditional methods. This improvement in translation quality has far-reaching implications for various domains, including cross-cultural communication, localization of content, and accessibility of information in multiple languages.

Another motivation for NMT research lies in addressing linguistic challenges. Language is inherently complex, with nuances, idiomatic expressions, and context-dependent meanings. NMT models aim to capture these linguistic complexities by utilizing neural architectures that can effectively encode and decode sentences, handle syntactic and semantic structures, and generate accurate translations that preserve the intended meaning. Through advanced training techniques and architectural innovations, researchers strive to push the boundaries of NMT and enable more faithful and context-aware translations.

Furthermore, NMT research seeks to address the translation needs of low-resource and rare languages. Many languages lack sufficient parallel training data, posing a significant challenge for traditional statistical approaches. NMT offers potential solutions by exploring transfer learning, unsupervised learning, and techniques to leverage related languages. By developing robust NMT models for low-resource languages, researchers aim to bridge the language divide and make translation accessible to a wider range of languages and communities.

Multilingual translation and code-switching are also key motivations for NMT research. With globalization and multilingual communication becoming increasingly prevalent, NMT models are designed to handle diverse language pairs, seamlessly switch between languages, and capture the nuances of code-switching. This capability is crucial for various applications, including international business communication, online content translation, and multilingual customer support.

Additionally, NMT research focuses on domain-specific translation, where specialized models are developed to cater to specific domains such as legal, medical, or technical fields. Domain adaptation techniques and fine-tuning methods enable NMT models to generate accurate and domain-specific translations, addressing the unique translation challenges in these specialized domains.

In summary, the motivation for studying NMT stems from the need to overcome the limitations of traditional machine translation methods, improve translation quality, address linguistic complexities, handle low-resource and rare languages, facilitate multilingual communication, enable domain-specific translations, and advance the state-of-the-art in machine translation. By addressing these motivations, NMT research contributes to the development of more accurate, efficient, and accessible translation systems, fostering global understanding and bridging the language barriers in our interconnected world.

1.2 Outlines

The contribution of this work can be summarized as follows:

- Chapter 1: Introduction. This chapter briefly presents our motivation and approach for the missing joints problem in human motion.
- Chapter 2: Related works. This chapter briefly describes the related work associated with recovering missing markers in human motion.
- Chapter 3: Methodology. This chapter describes in detail our method for predicting missing markers.
- Chapter 4: Evaluation Metrics. This chapter presents and discusses the metrics used to assess the performance of the proposed method for predicting missing markers in human motion.
- Chapter 5: Experiments And Results. This chapter illustrates our method performance in a large-scale dataset and compares it to other methods.
- Chapter 6: Conclusion and future works.

CHAPTER 2. RELATED WORK

2.1 Transformer-based Models

Transformer-based models have revolutionized the field of natural language processing (NLP) and achieved state-of-the-art performance across various language tasks. These models, such as OpenAI's GPT-3, are built upon the Transformer architecture, which was first introduced by Vaswani et al. in 2017. This document serves as an introduction to transformer-based models, explaining their key components and highlighting their advantages in NLP applications.

The Transformer architecture is a neural network model that relies on a self-attention mechanism to capture the relationships between words in a sequence. Unlike recurrent neural networks (RNNs) or convolutional neural networks (CNNs), Transformers can process the entire input sequence in parallel, making them more efficient for long-range dependencies.

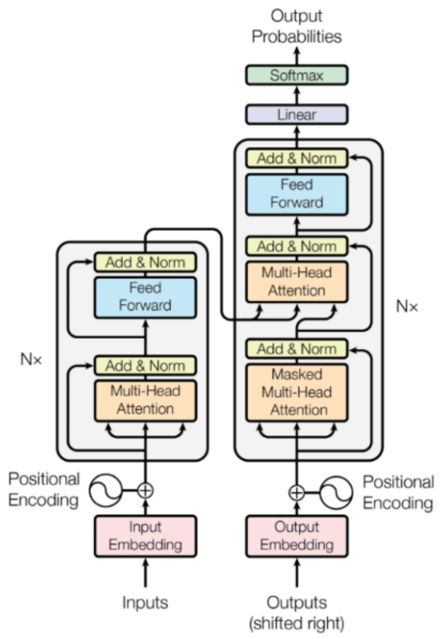

Figure 2.1: Transformer-based Models

The core components of the Transformer architecture are:

Encoder: The encoder processes the input sequence and extracts meaningful representations for each word or token. It consists of multiple layers, each comprising a self-attention mechanism followed by a position-wise feed-forward network.

Decoder: The decoder generates output sequences based on the representations learned by the encoder. It also employs self-attention and additional attention layers to capture both the input sequence and the generated output.

2.2 Convolutional Sequence to Sequence Models

Convolutional Sequence to Sequence (ConvS2S) models are a variant of sequence-to-sequence models that employ convolutional neural networks (CNNs) as the main building blocks. Unlike traditional recurrent neural network (RNN)-based Seq2Seq models, ConvS2S models utilize 1D convolutions to capture dependencies and patterns in the input and output sequences.

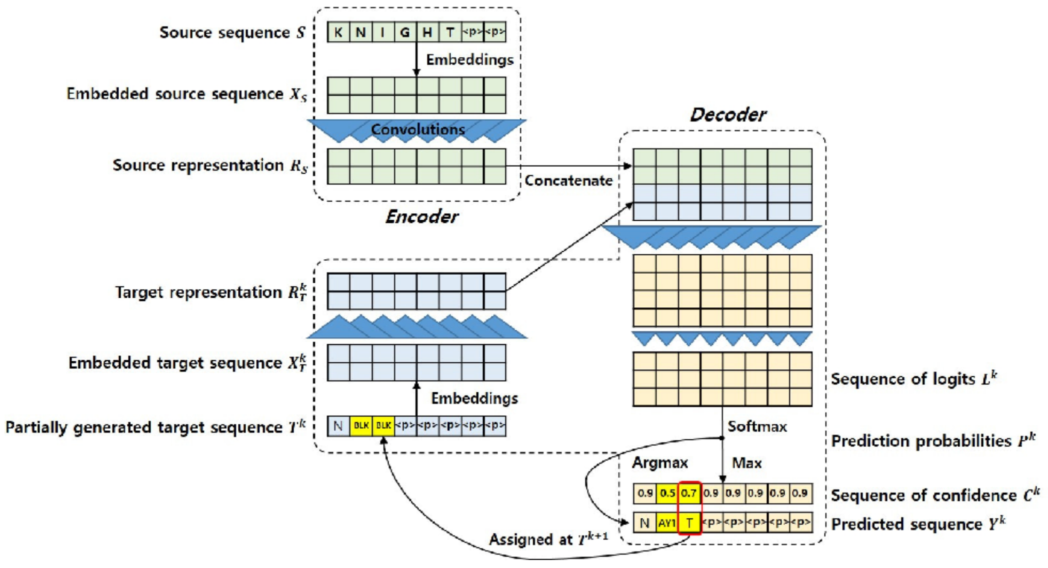
Source sequence S
K N I G H T <p> <p>
Embeddings
Embedded source sequence X_S
Convolutions
Source representation R_S
Concatenate
Encoder
Decoder
Target representation R_T^k
Embedded target sequence X_T^k
Embeddings
Partially generated target sequence T^k
N BLK BLK <p> <p> <p> <p> <p>
Sequence of logits L^k
Softmax
Prediction probabilities P^k
Argmax
Max
0.9 0.5 0.7 0.9 0.9 0.9 0.9 0.9
Sequence of confidence C^k
N AY1 T <p> <p> <p> <p> <p>
Predicted sequence Y^k
Assigned at T^{k+1}

Figure 2.2: Convolutional Sequence to Sequence Models

The architecture of ConvS2S models consists of convolutional layers for both the encoder and the decoder.

Encoder: The encoder in ConvS2S models typically consists of multiple layers of 1D convolutional filters. These filters slide over the input sequence, capturing local patterns and features. The output of the last convolutional layer is a high-dimensional representation that summarizes the input sequence.

Decoder: The decoder in ConvS2S models also employs 1D convolutions, but with a different objective. The decoder takes the high-dimensional representation from the encoder and generates the output sequence. It operates by sliding convolutions over the representation, predicting the next token at each step. The output tokens are usually generated using techniques like softmax or beam search.

CHAPTER 3. METHODOLOGY

3.1 Problem Definition

In this work, we address the task of Neural Machine Translation (NMT), where the goal is to learn a function that maps a source sentence $s \in \mathcal{S}$ to a target sentence $t \in \mathcal{T}$, where $\mathcal{S}$ and $\mathcal{T}$ are the source and target languages, respectively.

To represent the source and target sentences, we use the T5-base model, a pre-trained language model that can encode text into a fixed-length vector representation in $\mathbb{R}^{128}$. Specifically, we tokenize the source sentence s and the target sentence t using the T5 tokenizer, and obtain their vector representations $\mathbf{s} \in \mathbb{R}^{128}$ and $\mathbf{t} \in \mathbb{R}^{128}$, respectively.

Our task is then to learn a function f that takes as input the vector representation of the source sentence $\mathbf{s}$ and produces the vector representation of the target sentence $\mathbf{t}$, as shown in Eq. 3.1:

$$\mathbf{t} = f(\mathbf{s}) \tag{3.1}$$

To learn this function, we use a dataset $\mathcal{D}$ of pairs of source and target sentences, and we minimize a suitable loss function L that measures the discrepancy between the predicted target sentence $\mathbf{t}$ and the ground-truth target sentence $\mathbf{t}'$ for each pair $(s, t') \in \mathcal{D}$, as shown in Eq. 3.2:

$$L(\mathbf{t}, \mathbf{t}') = \text{loss}(\mathbf{t}, \mathbf{t}') \tag{3.2}$$

We train the function f using stochastic gradient descent (SGD) or a suitable variant, and evaluate its performance on a held-out test set of source-target sentence pairs. Our goal is to develop a high-quality NMT model that can accurately translate source sentences from $\mathcal{S}$ to target sentences in $\mathcal{T}$ with a high degree of fidelity.

3.2 Model Architecture

The T5-base model is a variant of the T5 model developed by Google, which uses a text-to-text approach to NMT. This means that both the input and output are in text format, and the model is trained to map one sequence of text to another sequence of text. The overview of our architecture is shown in Figure 3.1.

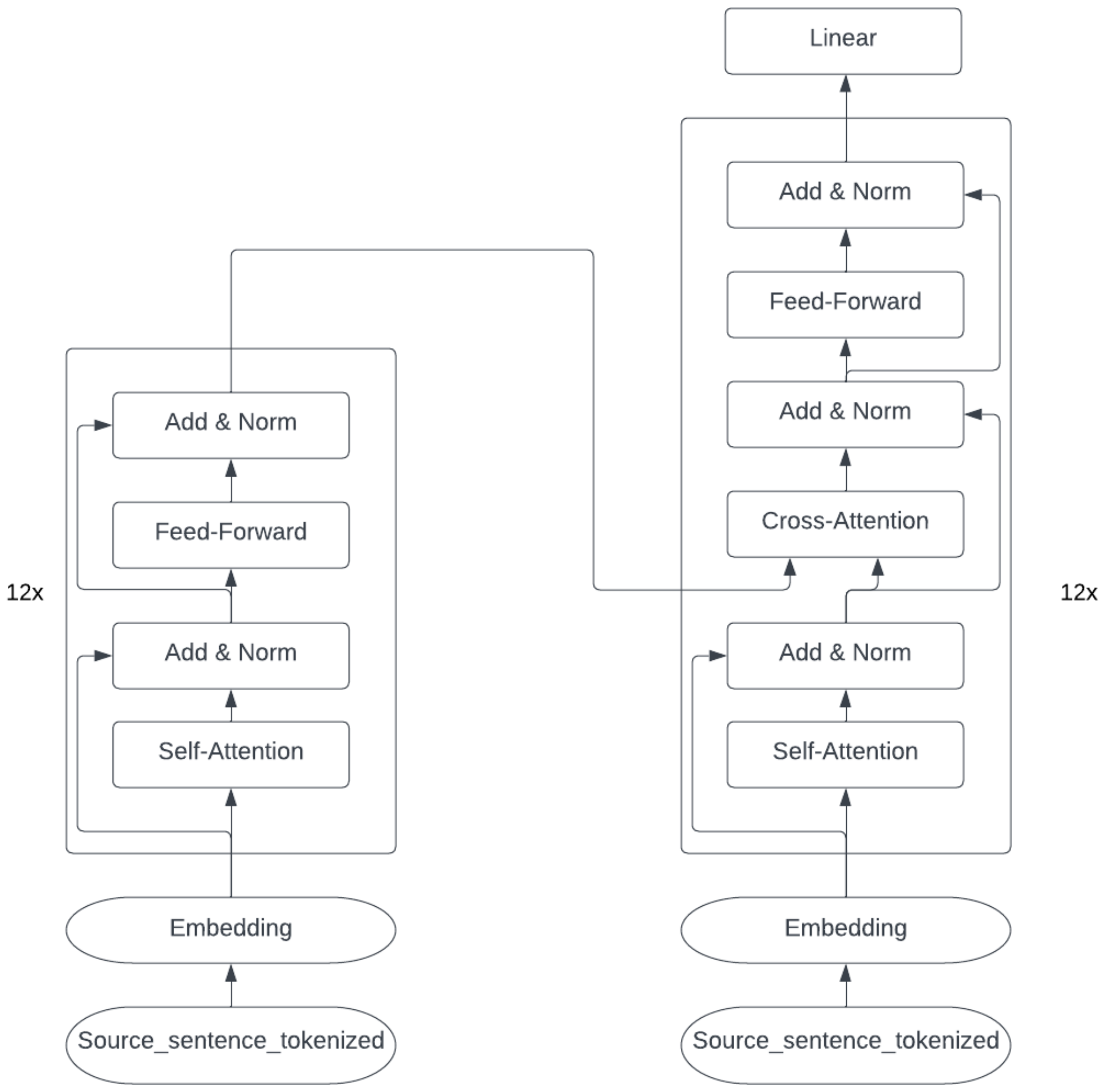

Figure 3.1: Overview Of The Proposed Architecture

3.2.1 Encoder Stack

The encoded input is then passed through a stack of 12 identical Transformer encoder layers. Each layer consists of a self-attention mechanism that allows the model to attend to different parts of the input sequence, followed by a position-wise feedforward network that applies a non-linear transformation to the hidden states.

3.2.2 Decoder Stack

The decoder stack is similar to the encoder stack, but includes an additional cross-attention mechanism that allows the model to attend to the encoded input. The decoder stack also has 12 identical layers, and each layer consists of a self-attention mechanism followed by a position-wise feed-forward network.

3.2.3 Output generation

The output sequence is generated using an auto-regressive approach with beam search. Beam search allows for considering multiple possible output sequences simultaneously. It starts by initializing a beam search queue with the initial input tokens. The beam width determines how many partial sequences are considered at each step.

To generate the output sequence, the model predicts the next token based on the previously generated tokens. The predicted token is then added to each partial sequence in the beam search queue, resulting in new candidate sequences. The probabilities of these candidates are calculated, and the top-k candidates with the highest probabilities are retained.

This process continues iteratively until the maximum desired output length is reached or a stopping criterion is met. The final output sequence is selected based on the sequence with the highest overall probability among the candidates in the beam search queue.

3.2.4 Overview

Overall, T5 encoder-decoder Transformer implementation closely follows its originally proposed form [9]. First, an input sequence of tokens is mapped to a sequence of embeddings, which is then passed into the encoder. The encoder consists of a stack of "blocks", each of which comprises two subcomponents: a self-attention layer followed by a small feed-forward network. Layer normalization [1] is applied to the input of each subcomponent. We use a simplified version of layer normalization where the activations are only rescaled and no additive bias is applied. After layer normalization, a residual skip connection [3] adds each subcomponent's input to its output. Dropout [8] is applied within the feed-forward network, on the skip connection, on the attention weights, and at the input and output of the entire stack. The decoder is similar in structure to the encoder except that it includes a standard attention mechanism after each self-attention layer that attends to the output of the encoder. The self-attention mechanism in the decoder also uses a form of autoregressive or causal selfattention, which only allows the model to attend to past outputs. The output of the final decoder block is fed into a dense layer with a softmax output, whose weights are shared with the input embedding matrix. All attention mechanisms in the Transformer are split up into independent "heads" whose outputs are concatenated before being further processed.

Since self-attention is order-independent (i.e. it is an operation on sets), it is common to provide an explicit position signal to the Transformer. While the original Transformer used a sinusoidal position signal or learned position embeddings, it has recently become more common to use relative position embeddings [6, 4]. Instead of using a fixed embedding for each position, relative position embeddings produce a different learned embedding according to the offset between the "key" and "query" being compared in the self-attention mechanism. We use a simplified form of position embeddings where each "embedding" is simply a scalar that is added to the corresponding logit used for computing the attention weights. For efficiency, we also share the position embedding parameters across all layers in our model, though within a given layer each attention head uses a different learned position embedding. Typically, a fixed number of embeddings are learned, each corresponding to a range of possible key-query offsets. In this work, we use 32 embeddings for all of our models with ranges that increase in size logarithmically up to an offset of 128 beyond which we assign all relative positions to the same embedding. Note that a given layer is insensitive to relative position beyond 128 tokens, but subsequent layers can build a sensitivity to larger offsets by combining local information from previous layers. To summarize, our model is roughly equivalent to the original Transformer proposed by Vaswani et al. (2017) with the exception of removing the Layer Norm bias, placing the layer normalization outside the residual path, and using a different position embedding scheme. Since these architectural changes are orthogonal to the experimental factors we consider in our empirical survey of transfer learning, we leave the ablation of their impact for future work.

3.3 Training

During training model, Cross-entropy loss is used when adjusting model weights. The aim is to minimize the loss, i.e, the smaller the loss the better the model. Cross-entropy is described as follows

$$l_{CE} = \sum_{i=1}^{n} t_i \log p_i, \qquad \text{for n classes}$$

where t_i is the truth label and p_i is the Softmax probability for the i^{th} class.

3.4 Implementation

The training data for our NMT model was taken from the iwslt_en_vi dataset. We utilized the BPE tokenizer from the Hugging Face's transformers library to preprocess the data, which involved segmenting the text into subword units and converting it into a numerical format. The advantage of using BPE is that it can effectively handle out-of-vocabulary words by breaking them down into smaller subwords and then reconstructing them during decoding. This helps the model to better generalize to unseen words and improves the overall performance of the system.

Model Architecture: We used the T5ForConditionalGeneration model from the transformers library, which is based on the Transformer architecture. The model has

12 layers each in the encoder and decoder.
Optimization: We used the Adafactor optimizer [7] with a learning rate of 3e-4, epsilon values of (1e-30, 1e-3), and clip threshold of 1.0. We also set the decay rate to -0.8, and weight decay to 0.0. We used the get_linear_schedule_with_warmup function to schedule the learning rate, with a warm-up period of 500 steps.

Training: We trained the model for a total of 20 epochs. We used a batch size of 16, and the training process involved iterating over the training data and updating the model parameters using backpropagation. During training, we monitored the average training loss per epoch and the BLEU score on the validation set.

CHAPTER 4. EVALUATION METRICS

4.1 Definition

In this project, we used BLEU score (Bilingual Evaluation Understudy score) as evaluation metric for Neural Machine Translation problem. The BLEU score is a commonly used metric for evaluating the quality of machine translation systems. It measures the similarity between a machine-translated sentence and one or more reference translations. Bleu Scores are between 0 and 1, with higher values indicating better translation quality. The BLEU score is calculated based on the precision of n-gram matches between the machine-translated sentence and the reference translations, with a penalty for shorter machine translations.

The BLEU score is implemented in the MultilingualMT-UET-KC4.0 project, which is an open-source project developed by the UETNLPLab group. (Github: `https://github.com/KCDichDaNgu/KC4.0_MultilingualNMT`).

4.2 Calculating Bleu Score

4.2.1 Geometric Average Precision

We calculated the cumulative 4-gram BLEU score, also called BLEU-4. The weights for the BLEU-4 are 1/4 (25%) or 0.25 for each of the 1-gram, 2-gram, 3-gram and 4-gram scores. To calculate BLEU score, first we need to calculate Precision Score for 1-gram to 4-gram by using the Clipped Precision method. Then using Geometric Average to calculate the Average Precision Scores

$$\begin{aligned}\textbf{Geometric Average Precision }(N) &= \exp \sum_{n=1}^{N} w_n \log p_n && (4.3)\\ &= \prod_{n=1}^{N} P_n^{w_n} && (4.4)\\ &= (p_1^{\frac{1}{4}}).(p_2^{\frac{1}{4}}).(p_3^{\frac{1}{4}}).(p_4^{\frac{1}{4}}) && (4.5)\end{aligned}$$

which p_n is Precision n-gram.

4.2.2 Brevity Penalty

To penalize sentences that are too short, we calculate a Brevity Penalty

$$\textbf{Brevity Penalty} = \begin{cases} 1 & \text{if } c > r \\ e^{1-\frac{r}{c}} & \text{if } c \leq r \end{cases} \tag{4.6}$$

which **c** is predicted length = number of words in the predicted sentence and **r** is target length = number of words in the target sentence

4.2.3 Bleu Score

Finally, to calculate the Bleu Score, we multiply the Brevity Penalty with the Geometric Average of the Precision Scores

$$\textbf{BLEU } (N) = \textbf{Brevity Penalty } \times \textbf{ Geometric Average Precision Scores } (N) \tag{4.7}$$

which $N = 4$ in our project.

CHAPTER 5. EXPERIMENTS AND RESULTS

In this chapter, we evaluate the performance of our model. We first describe the dataset and experiment setup and then continue with the qualitative and quantitative results on reconstructing human motion samples. Besides, we discuss the strengths and limitations of the baseline and proposed method.

5.1 Dataset and Preprocessing

The iwslt_en_vi dataset is a parallel corpus consisting of English-Vietnamese sentence pairs. This dataset is commonly used for machine translation research. A summary of the dataset is shown in **Table 4.1**.

Dataset	**iwslt_en_vi**
Number of training samples	133,317
Number of validating samples	1,553
Number of testing samples	1,268

Table 5.1: Dataset Statistics

To prepare the iwslt_en_vi dataset for use with the T5 model, we first tokenize and encode the source and target sentences using the T5 tokenizer. The tokenizer breaks up the input text into individual tokens, and encodes each token as a unique integer. This process allows the model to work with discrete inputs, and enables the use of techniques such as positional encoding and attention.

Next, we format the data into input-target pairs that can be used to train the T5 model. Specifically, we take each source sentence in the dataset and pair it with its corresponding target sentence, such that the T5 model is trained to generate the target sentence given the source sentence as input. We also add special tokens to the beginning and end of each target sentence to indicate the start and end of the output sequence.

Finally, we feed these input-target pairs into the T5 model using the tokenizer of pretrained ('t5-base') model. This model has been pre-trained on a large amount of text data using a sequence-to-sequence pre-training objective, and is fine-tuned on the iwslt_en_vi dataset using a cross-entropy loss function. During training, the model learns to generate high-quality translations by minimizing the difference between its predicted output and the true target sentence.

Overall, the use of the iwslt_en_vi dataset and T5 model with tokenization and encoding preprocessing enables us to effectively train an NMT model to translate English sentences to Vietnamese.

5.2 Experiments and Result

We conduct 2 experiments in order to compare our model with other competitive models. The result is summarized in **Table 4.2**.

5.2.1 Baselines Settings

To better verify the performance of our model, we chose several methods for comparison, i.e., Transformer, KC4.0_MultilingualNMT, Transformer-Fairseq, ConvS2S. The hyperparameters settings are described as follows:

Transformer [9]: The Transformer model has been shown to achieve promising performance on several NMT benchmarks and is widely used in research and industry. We implement a network containing 3 layers of Encoders and Decoders, d_model = 512, num_heads = 8.

KC4.0_MultilingualNMT: The KC4.0_MultilingualNMT is a repository containing code for a multilingual neural machine translation system that can translate between several languages using a single model, based on the Transformer architecture. Epoch = 10

Transformer-Fairseq [5]: Fairseq is a sequence modeling toolkit that allows researchers and developers to train custom models for translation, summarization, language modeling and other text generation tasks. We used the Transformer model trained by the Fairseq team. Hyperparameters: max-tokens=4096, lr=5e-4, dropout=0.3, beam=5

ConvS2S [2]: Convolutional Sequence to Sequence is a neural network model that combines convolutional neural networks (CNNs) with sequence-to-sequence models. It is used for tasks like machine translation and text generation. We used the model trained by the Fairseq team.Hyperparameters: Lr=0.5, max-token=4000, dropout=0.2, beam=5
Evaluation Metric: The metric we use is the Multi-Bleu Score which is a metric commonly used to evaluate the performance of machine translation systems.

5.2.2 Quantitative results

In our NMT experiments, we set the maximum length of the input sequence to 128, and trained three different models: the Transformer model for 50 epochs, the KC4.0 model for 10 epochs, and the T5-base model for 20 epochs. During the training process, we monitored the multi-BLEU score, training loss, and time per epoch (in seconds). The results of our experiments can be found below, and may be useful in determining which model to use for a given NMT task.

Model	**Multi-Bleu Score**	**Training Loss**	**Time per epoch (s)**
Transformer	22.38	1.377	240
KC4.0_MultilingualNMT	25.76	1.254	708
Transformer-Fairseq	25.03	1.213	255
ConvS2S	21.45	1.507	489
T5-Base	**31.35**	**0.413**	**11500**

Table 5.2: Quantitative comparisons of Multi-Bleu Score between T5-Base and others baselines

CHAPTER 6. CONCLUSION

Although our proposed NMT model has achieved some results, they are still not good enough to applying this model in practice. There are some solutions for future work that could be archived to further improve its performance.

Domain-specific adaptation: Investigate fine-tuning the T5 model on domain-specific datasets, such as medical or technical texts. This can improve translation quality and adapt the model to specific domains.

Contextual information integration: Explore methods to incorporate contextual information, such as transformer-based contextual encoders like BERT or RoBERTa, into the T5 model. This can enhance the model's ability to capture context-specific meanings in translation.

Low-resource language support: Investigate techniques to improve translation performance for low-resource language pairs like Vietnamese-English. This may involve transfer learning, unsupervised or semi-supervised approaches, and data augmentation techniques.

Evaluating: In this study, we focused on evaluating the performance of the T5 model on the prepared dataset. However, it is important to note that we did not evaluate the model on other datasets. To ensure the reliability and robustness of the results, further evaluation of the model on diverse datasets would be an important future research direction.

In conclusion, Neural Machine Translation has revolutionized the field of machine translation and holds great promise for bridging language barriers and facilitating global communication. Continued research and development efforts will further refine and enhance NMT systems, making them indispensable tools for overcoming language barriers in an increasingly interconnected world.

Bibliography

[1] Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. In *Advances in Neural Information Processing Systems*, pages 1647–1655, 2016.

[2] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning, 2017.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[4] Yanping Huang, Youlong Wang, Wei Yang, Chenzhuo Liu, Pengchuan Wang, Xipeng Xu, and Jingjing Liu. Gpipe: Efficient training of giant neural networks using pipeline parallelism. *CoRR*, abs/1811.06965, 2018.

[5] Nathan Ng, Kyra Yee, Alexei Baevski, Myle Ott, Michael Auli, and Sergey Edunov. Facebook fair's wmt19 news translation task submission, 2019.

[6] Peter Shaw and Jakob Uszkoreit. Self-attention with relative position representations. *CoRR*, abs/1803.02155, 2018.

[7] Noam Shazeer and Mitchell Stern. Adapting beam search for efficient training of neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2011–2022, 2018.

[8] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.