Final Project Report
CS 584 Fall 2022

Attention in Translation
OLADAPO OGUNNAIKE A20435197

1. Problem Statement:

Deep neural networks have achieved great success in a variety of natural language processing (NLP) tasks, including language modeling, paraphrase detection, and word embedding extraction. In this project, we focus on the task of machine translation.
Machine translation is the task of automatically translating text from one language to another. There are three main types of machine translation: rule-based machine translation (RBMT), statistical machine translation (SMT), and neural machine translation (NMT). RBMT systems use hand-crafted rules to translate text, while SMT systems use statistical models to learn the relationships between words and phrases in different languages. NMT systems use deep neural networks to learn the relationships between words and phrases in different languages.
Attention mechanisms have been shown to be effective for improving the performance of NMT systems. Attention mechanisms allow NMT systems to focus on different parts of the input sentence when generating the output sentence. This is important because different parts of the input sentence may be more relevant to different parts of the output sentence.
In this project, we will study the attention mechanism in detail. We will discuss the different types of attention mechanisms, their theoretical properties, and recent works on attention mechanisms. We will also implement an attention mechanism in a neural machine translation system and evaluate its performance on a machine translation task.
We believe that this project will make a significant contribution to the field of machine translation. By understanding the attention mechanism, we can design more effective machine translation systems that are able to produce more accurate and fluent translations.
.

2. Background work:

Most machine translation models are encoder-decoder models. The encoder reads the input sentence and produces a fixed-length vector representation of the sentence. The decoder then takes this vector representation and produces the output sentence.

The goal of the encoder-decoder model is to maximize the probability of generating the correct output sentence given the input sentence.
One potential issue with encoder-decoder models is that they do not perform well on long sentences. This is because the encoder has to compress all of the information in the input sentence into a fixed-length vector. This can lead to information loss, which can impact the quality of the output sentence.
To address this issue, Bahdanau et al. (2014) introduced the attention mechanism. The attention mechanism allows the decoder to focus on different parts of the input sentence when generating the output sentence. This is important because different parts of the input sentence may be more relevant to different parts of the output sentence.
The attention mechanism was originally implemented using recurrent neural networks (RNNs). However, RNNs can be computationally expensive, especially for long sentences. To address this issue, Vaswani et al. (2017) proposed the Transformer, a model architecture that only relies on the attention mechanism. The Transformer is able to achieve state-of-the-art performance on machine translation tasks, while also being more computationally efficient than RNN-based models.
In summary, attention mechanisms are a powerful tool for improving the performance of machine translation models. They allow the decoder to focus on the most relevant parts of the input sentence when generating the output sentence. This can lead to significant improvements in translation accuracy, especially for long sentences.

3. RNN Encoder-Decoder:

RNN Encoder-Decoder proposed by Cho, K. et al [2]. Given an input sentence as a sequence of vectors $x = (x_1, x_2, \ldots, x_{T_x})$, the encoder produces a context vector c such that:

$$c = q(\{h_1, h_2, \ldots, h_{T_x}\}) \text{ given } h_t = f(x_t, h_{t-1}) \quad (1)$$

Where h_t is a hidden state at time t and f, q are some non-linear functions defined through the training process.

Given this context vector c and the previously predicted words $\{y_1, y_2, \ldots, y_{t'-1}\}$, the decoder predicts the next word $y_{t'}$ with the probability:

$$p(y) = \prod_{t=1}^{T} p(y_t|\{y_1, \ldots, y_{t-1}\}, c) \quad (2)$$

Where each conditional probability is computed by training a non-linear function g such that with the hidden state s_t of the decoder:

$$p(y_t|\{y_1, \ldots, y_{t-1}\}, c) = g(y_{t-1}, s_t, c) \quad (3)$$

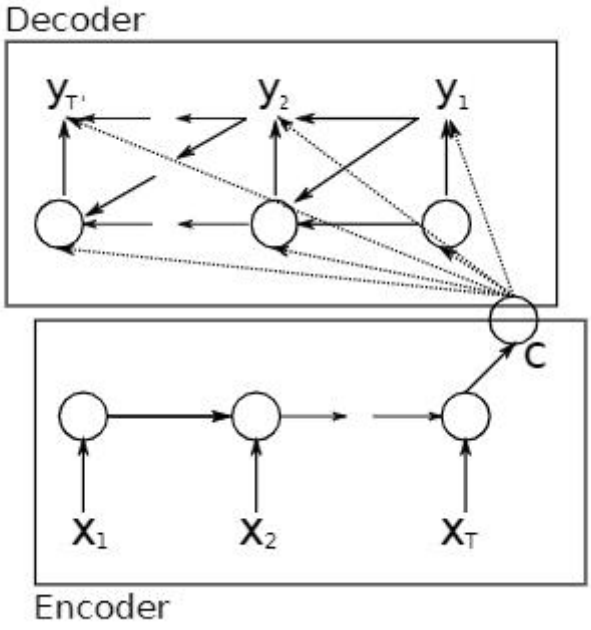

Figure 1: RNN Encoder-Decoder architecture

4. Attention mechanism:

Bahdanau et al. (2014) introduced the attention mechanism, particularly soft attention. Soft attention is a weighted sum of the encoder's hidden states, where the weights are computed based on how well each encoder state aligns with the current decoder state. The alignment score is computed using a feed-forward neural network called the alignment model.

The attention mechanism can be implemented as follows:

1. The decoder's hidden state *hi* is passed to the alignment model.
2. The alignment model outputs an alignment score *aij* for each encoder state *hj* .
3. The alignment scores are normalized using the softmax function to produce a set of weights *aij* .
4. The weighted sum of the encoder's hidden states is computed as *ci* =∑*j* *aij* *hj* .

5. The decoder's hidden state is updated as $hi' = hi + ci$.

The attention mechanism allows the decoder to focus on different parts of the input sentence when generating the output sentence. This is important because different parts of the input sentence may be more relevant to different parts of the output sentence. For example, when translating the sentence "I am a student" into Spanish, the decoder may need to focus on the word "student" when generating the Spanish word "estudiante." The attention mechanism allows the decoder to do this by assigning a higher weight to the encoder state corresponding to the word "student."

The attention mechanism has been shown to be effective for improving the performance of machine translation models. It has also been used in other natural language processing tasks, such as text summarization and question answering.

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T_x} exp(e_{ik})} \quad (5)$$

Where $e_{ij} = a(s_{i-1}, h_j)$, a is the alignment model (6)

Finally, the context vector c_i is defined as the weighted sum of the annotations h_j with weight α_{ij}:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j \quad (7)$$

α_{ij} measures the importance of the annotation h_j with respect to the hidden state s_{i-1} in deciding the next state s_i and generating y_i . With the attention mechanism implemented in it, the decoder decides which part to pay attention to. This means that the encoder now does not have to encode the sequences into fixed-length vectors. The figure below displays the overall architecture:

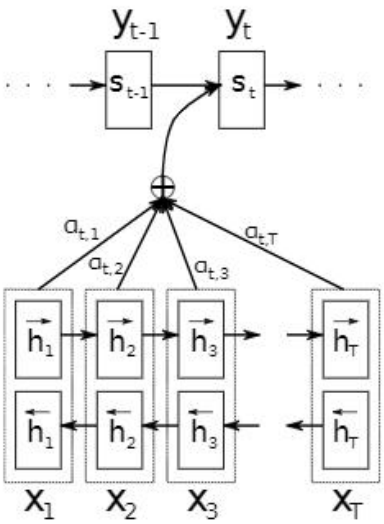

Figure 2: Proposed RNN Encoder-Decoder architecture with attention mechanism

5. Alternative Interpretation: Scaled Dot-Product Attention:

The decoder state can be thought of as a query, and the encoder states can be thought of as keys. The goal of the attention mechanism is to compute a weight for each query-key pair, and then normalize these weights using the softmax function.

The attention weight for a query-key pair is computed as the dot product of the query and key vectors. This dot product measures how similar the query and key vectors are. The softmax function is then used to normalize the attention weights so that they sum to 1. This ensures that the decoder only attends to a subset of the encoder states, and that the attention weights are consistent with each other.

The attention mechanism allows the decoder to focus on different parts of the input sentence when generating the output sentence. This is important because different parts of the input sentence may be more relevant to different parts of the output sentence. For example, when translating the sentence "I am a student" into Spanish, the decoder may need to focus on the word "student" when generating the Spanish word "estudiante." The attention mechanism allows the decoder to do this by assigning a higher weight to the encoder state corresponding to the word "student."

The attention mechanism has been shown to be effective for improving the performance of machine translation models. It has also been used in other natural language processing tasks, such as text summarization and question answering.

This idea is defined as Scaled Dot-Product Attention by Vaswani et al. [1]. Specifically, given a set of queries and keys of dimension d_k and values of dimension d_v. The

attention weight is computed by taking the dot product between the query with all the keys, dividing each by $\sqrt{d_k}$, and applying the Softmax function. A vectorization equation is defined below with a set of queries packed in matrix Q, and matrix K, V containing the keys and values respectively:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \quad (8)$$

The paper introduces a scaled factor $\frac{1}{\sqrt{d_k}}$ to counteract the situation that as for large d_k or as the size of queries and keys increase, the dot product grow large in magnitude which pushes the Softmax function into regions where it has extremely small gradients.

6. Multi-head Attention:

The proposed Transformer model in [1] also utilizes the notion of Multi-head Attention which applies the attention function multiple times on different, learned linear projections of the keys, values, and queries. Each function produces a d_v -dimensional output. These values are then concatenated and projected again, resulting in the final output.
This process is shown in the figure below:

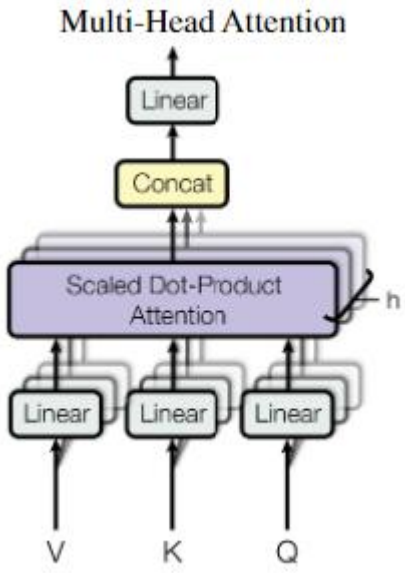

Figure 3: Multi-head attention.

7. Transformers:

The Transformer model is a network within the family of encoder-decoders for NMT tasks. Unlike previous network architecture, It does not use RNN and relies entirely on the attention mechanism.

The encoder consists of N identical layers each containing 2 sub-layers: The Multi-Head Attention layer and a feed-forward fully-connected network. In addition, there is a residual connection around each sub-layers and a layer normalization.
The decoder also consists of N identical layers containing the 2 sub-layers similar to the encoder plus another Multi-Head Attention layer with residual connection and layer normalization that takes the output from the encoder. The network also implements masking on the self-attention sub-layer to only consider input preceding the current time step.

The network utilizes cross-attention which refers to the interaction between encoder and decoder mentioned in previous sections. It also uses self-attention for a position in the encoder to attend to all the positions in previous layers of the encoder and for a position in the decoder to attend to all the positions in the decoder up to that position. Finally, the paper for the Transformer model [1] also introduces positional encoding which simply is a mechanism for the network to be aware of the word order in the sentences.

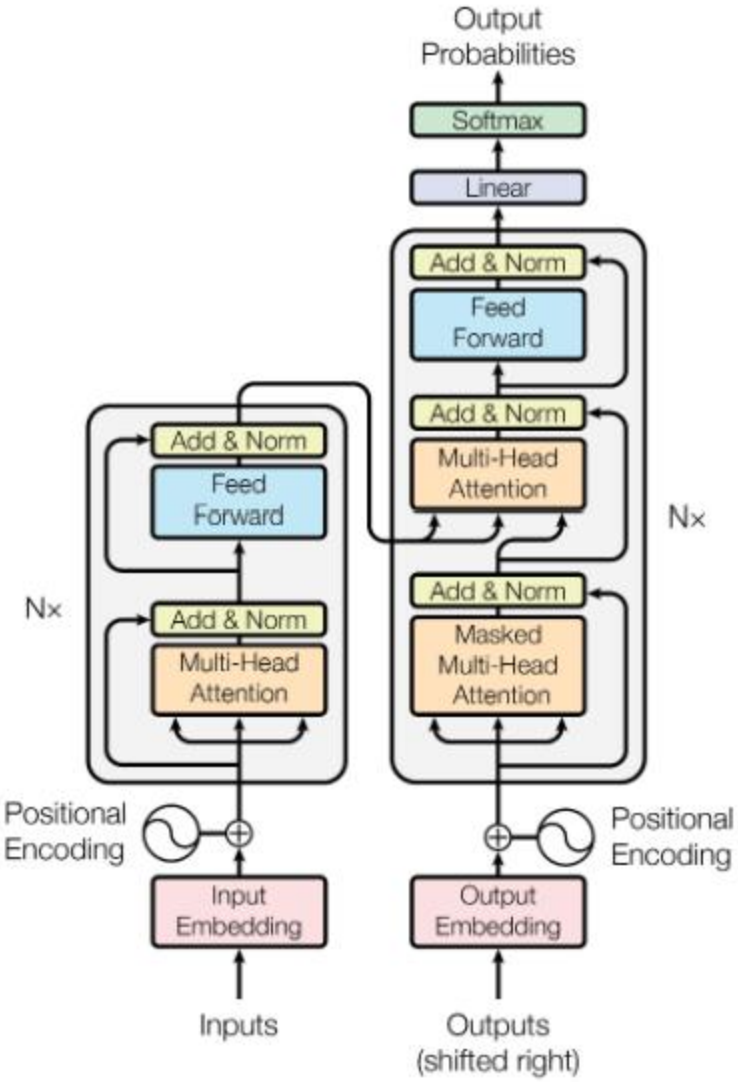

Figure 4: Transformer network architecture [1]

8. BLEU Score

To evaluate the performance of the machine translation model, the Bilingual Evaluation Understudy Score, or BLEU is used. This metric was first proposed by Kishore Papineni, et a [3]. BLEU is widely used as it is quick, inexpensive, and language-independent. It correlates well with human evaluation.

BLEU Score consists of 2 parts: the Geometric Average Precision Score and the Brevity Penalty BP. The equation for the metric is shown below:

$$BLEU = BP\ exp(\sum_{n=1}^{N} w_n log(p_n)) \quad (9)$$

BP is defined as:

$$\mathrm{BP} = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases} \quad (10)$$

Where r is effective corpus length, and c is the length of the candidate translation

The remaining part is the Geometric Average Precision Score computed based as the exponent of the weighted sum of the n-gram precisions p_n with weight w_n across n-grams up to N.

9. Data:

Manythings.org anki French-English corpus:

The zip file (fra-eng.zip) from [4] is downloaded and placed at the top level folder. It is then unzipped, and the resulting file fra.txt contains the sentence pairs. The create_manythings_dataset function handles reading the extracted file and returns the sentence pairs to corresponding sentence types: input and output. These variables contain the sentences in English and French respectively which can be split to get the training and test set. The sentences also need to be cleaned by removing unwanted characters, and symbols, lowercasing the letters, and adding <start>, and <end> flags to mark the start and end of the sentence.

```
Input Language; index to word mapping
1 ----> <start>
4 ----> i
30 ----> m
46 ----> at
28 ----> my
306 ----> place
3 ----> .
2 ----> <end>
```

```
Target Language; index to word mapping
1 ----> <start>
4 ----> je
30 ----> suis
5 ----> a
13 ----> la
112 ----> maison
3 ----> .
2 ----> <end>
```

Figure 5: Example of encoded sentence pairs

Europarl French English corpus:

The Europarl French-English dataset consists of 2 main files europarl-v7.fr-en.en and europarl-v7.fr-en.fr which represents the English and French sentences respectively. These files are loaded into their corresponding variable and cleaned similarly to (3.1.1).

For reuse, the cleaned data is saved into pkl files. The data from the pkl files are then used to generate training and testing datasets.

Tokenization of the corpus data:

Similar to other Natural Language Processing tasks, there is a need to tokenize the text data. A tokenizer is constructed for each language using the Tokenizer from the Keras library. Each tokenizer encodes the texts to create a mapping between indexes and words. Each sentence in the training and testing data is then encoded with the corresponding mapping and padded to have the size of the longest sequence.

10. Results and discussion:

The implementation of the experiments in this project is in this repository.

To get started, run the notebook files which contain the implementations of all the network architectures. The repository also contains some python scripts which contain the implementation of the RNN Encoder-Decoder with attention network, but due to some hardware limitations, it was then moved to the notebook files to be used on Google colab. The python scripts can be run as below after installing the packages from requirements.txt and downloading the datasets:

python train.py

- **'--dataset'**, type = str, default = 'manythings': Dataset used for training, either 'manythings' or 'europarl'
- **'--path_manythings'**, type = str, default = 'fra.txt': Path to the dataset file if the dataset 'manythings' is used
- **'--input_data',** type = str, default = 'english.pkl': Path to the list of input sentences if 'europarl' dataset is used.
- **'--output_data',** type = str, default = 'french.pkl': Path to the list of target sentences if 'europarl' dataset is used.
- **'--num_samples'**, type = int, default = 50000: Number of sentence pairs to train a model on.
- **'--batch_size',** type = int, default = 64: Batch size
- **'--embedding_dim'**, type = int, default = 512: Embedding dimension
- **'--units'**, type = int, default = 512: Number of units in the encoder and decoder
- **'--attention_units'**, type = int, default = 10: Number of units in the attention layer

- **'–epochs'**, type = int, default = 15: Number of epochs.
- **'–train',** type = bool, default = True: Whether to train the model or load a trained model to do inference.

Comparison of Transformers and normal RNN Encoder-Decoder networks:

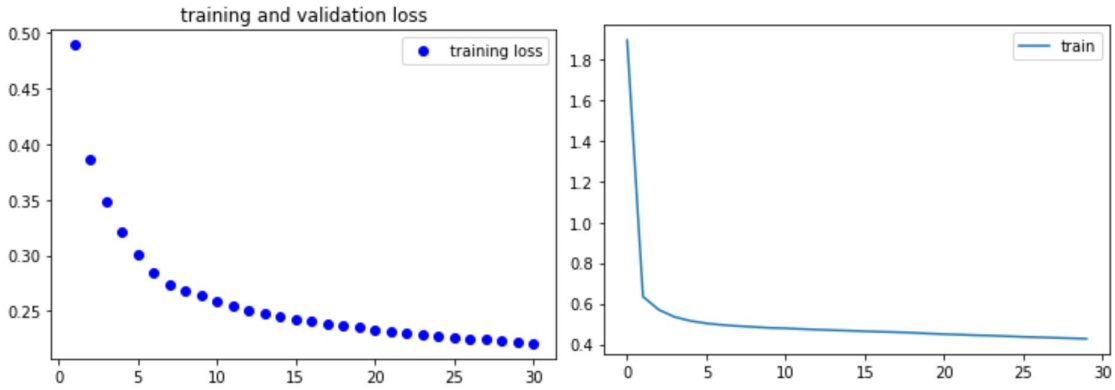

Figure 6: Comparison of training loss with the Transformer network (left) and simple RNN Encoder-Decoder network (right)

The 2 models are trained on the same manythings.org/anki dataset using around 45000 sentence pairs. The transformer network contains 8 heads, an embedding dimension of 256, and a latent dimension of 2048. On the other hand, the RNN encoder-decoder network initially utilizes LSTM layers with 512 units. The figure shows that the Transformer outperforms the RNN encoder-decoder network.

Another RNN Encoder-Decoder network is constructed with LSTM layers but with 2048 units. The updated figure below shows that it can perform better than our transformer implementation. However, this is at the cost of longer training time, and the number of learned parameters is a lot larger than the transformer model (161M compared to 46M):

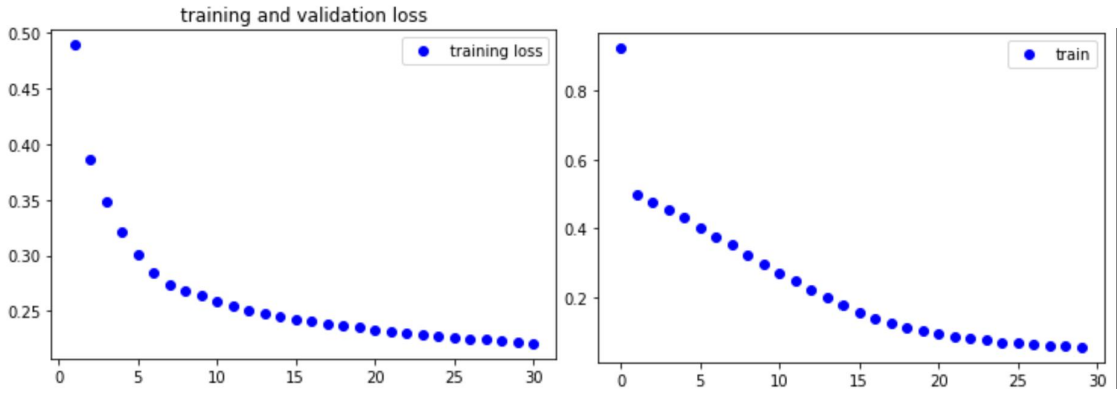

Figure 7: Comparison of training loss with the Transformer network (left) and simple RNN Encoder-Decoder network (right)

BiRNN Encoder and RNN Encoder in Encoder-Decoder networks with attention:

In this project, first, the network architecture with BiRNN Encoder and attention is trained and compared against baseline RNN Encoder-Decoder but with attention. Due to the hardware limitations, the former model is trained with 512 units in the encoder and 1024 units in the decoder. The latter is trained with 1024 units in the encoder and 1024 units in the decoder. The results are shown below:

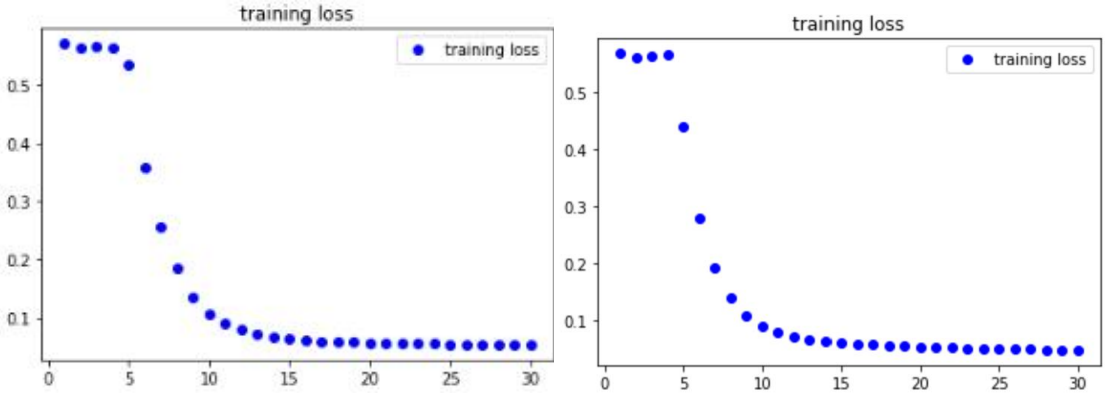

Figure 8: Training process of the proposed architecture with BiRNN Encoder (left) or RNN Encoder (right)

Based on the figure, it seems that the performances of the 2 models are comparable with each having roughly a Categorical Cross-Entropy loss value of 0.05. These models also show a better performance than the model trained with RNN Encoder-Decoder without attention in figure 6 showing the effect of the mechanism. Even an improved model with over 100M parameters only has roughly equivalent performance.

Effects of sentence length on the model performance:

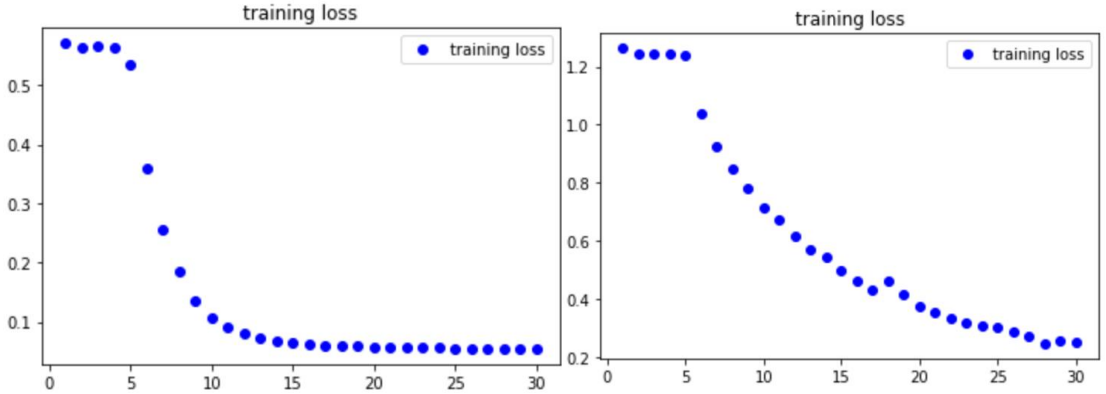

Figure 9: Training process of the proposed model architecture on the manythings.org (left) and Europarl (right) datasets.

It is intuitive that with longer sentences, the model trained on the Europarl dataset has a lower performance. However, due to the memory issue, this model is only trained with 256 encoder units and 512 decoder units which is just half as many units used in the other model. If trained on the same number of units and a few epochs longer, this model can achieve closer performance to the other model. This also reinforces the authors' intention that the models with the proposed architecture cope well with longer sentences.

BLEU Score evaluation:

BLEU-1	0.784
BLEU-2	0.681
BLEU-3	0.582
BLEU-4	0.426

Table 1: BLEU scores of the trained model

The trained model with the architecture in [2] is evaluated using the BLEU scores. The BLEU scores are applied to the sentence pairs of the test sets. The rows in the table above represent the BLEU scores for 1-gram, 2-gram, 3-gram, and 4-gram respectively. The model performs well with 1-gram, but its performance decreases as the n-grams increase.

Attention plot:

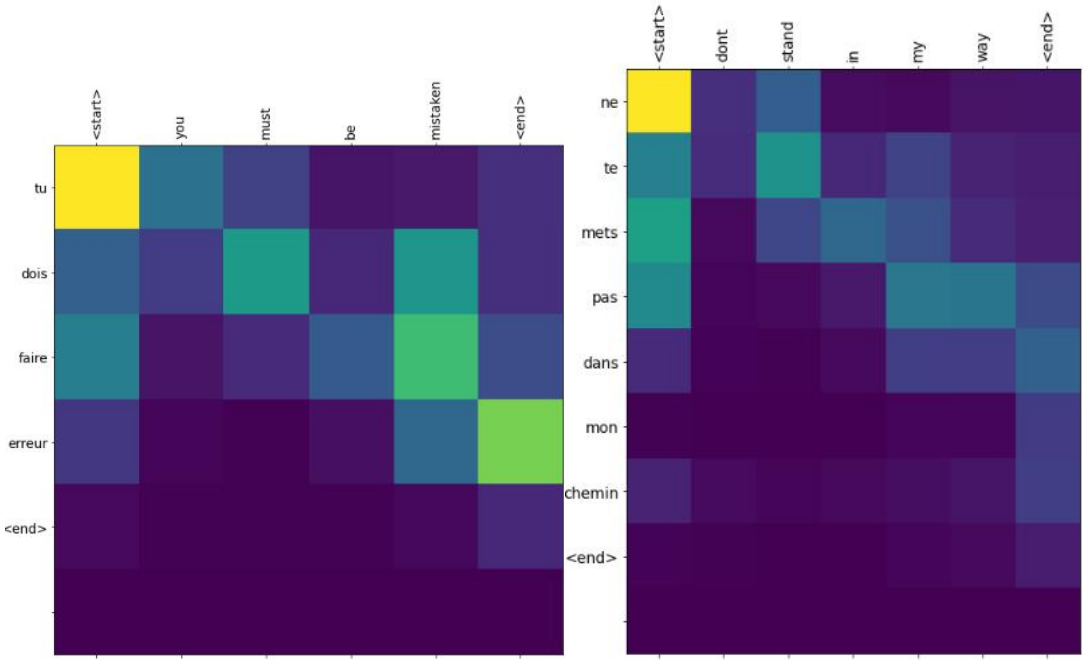

Figure 10: Attention plot of 2 different sentences translation

The plot above visualizes the attention weights of 2 different translations. It shows that the attention weights along the diagonal are relatively larger than in other places which is intuitive. There are other intuitions like the noun should start the sentence as it has a large weight on the top left of the first attention plot.

Inference:

Source	Translation	Target
don't stand in my way	ne te mets pas dans mon chemin	ne te mets pas dans mon chemin
you must be mistaken	tu dois faire erreur	vous devez etre malade
i was dismissed	on ma licencie	jai ete licencie
he has good eyesight	il a une bonne vue	il a une bonne vue

Table 2: Inference table

Based on the table above, the trained model performs quite well on translating the example sentences with the resulting translation being close or exactly the same to the target translation.

11. Conclusion and future works:

This project explores the notion of attention and its properties. It goes over the development of the attention mechanism from being proposed as an extension of the RNN Encoder-Decoder network architecture to being a main component of a state-of-the-art network in Transformers. The applications of such models expand beyond the task of machine translation into other NLP tasks like text summarization and even Computer Vision tasks through the proposal of Vision Transformers.

Personally, it has been a great learning experience for me. As a person invested in the field of Computer Vision, I learned about Vision Transformers a while ago and wanted to know how it came to begin with. This project helped me learn about the fundamentals of one of the most prominent network architectures today in Transformers and gain more experience in working on NLP tasks.

There are quite a few potential future works. It would be interesting to explore about Transformers and how it utilizes parallelization to speed up training. In addition, as mentioned before, the use of attention particularly and transformers as a whole in Computer Vision tasks can be explored further.

References:

[1] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

[2] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

[3] Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

[4] Manythings.org French-English corpus http://www.manythings.org/anki

[5] Machine Translation: What It is, and How It Works, retrieved 30 Sep 2022 from https://phrase.com/blog/posts/machine-translation/

[6] Tab-delimited Bilingual Sentence Pairs, retrieved 30 Oct 2022 from http://www.manythings.org/anki/