Neural Machine Translation like it's 2015

Didier Blach-Laflèche[1,3] , **Alexandre Dorais**[2] and **Maxime Moreau**[1,3]
[1]Department of Engineering Physics, École Polytechnique Montréal, Canada
[2]Department of Mathematics, École Polytechnique Montréal, Canada
[3]Department of Electrical Engineering, École Polytechnique Montréal, Canada
{dider.blach-lafleche, alexandre.dorais, maxime.moreau}@polymtl.ca

Abstract

Neural Machine Translation (NMT) is a fast-paced field in constant evolution. These days, it seems that BERT-like models based on transformers are the option of choice when training an NMT network, but if you travel five years back, these models didn't exist yet. However, the intuition that attention mechanisms provide massive improvements in NMT models was already budding. In this paper, we aim to prove this intuition by conducting an ablation study on an LSTM recurrent network with a global or a local attention mechanism. Global attention assigns weights to all words in the source sentence, whereas local attention only concentrates on a relevant window of the source sentence. We also include various tricks to improve the quality of the translation, such as reversing the source sentence before running the model, using dropout, and feeding the attentional vectors as inputs to the next timestep. We train our models using the Multi30k dataset, and we evaluate all of the models on the test set using the BLEU-4 score. We find that global attention increases the BLEU score by 5.74 over the model without an attention mechanism, and local attention further increases the BLEU score by 1.91. These results confirm that attention is crucial to improving the performance of NMT models.

1 Introduction

Whether you are a Professor of AI at Polytechnique needing to translate 100 slides per week from English to French, or a courageous traveler in a small foreign town needing to ask for directions without knowing the local language, then accurate machine translation might be exactly what you're looking for. Indeed, machine translation cloud services like Google Translate are equivalent to having a professional translator in your pocket at all times. Of course, that's only true if the machine translation model is as good as a professional translator. However, it is never quite so simple to solve complex cognitive tasks programmatically, and that is where AI and neural networks come in.

1.1 Previous works

Machine translation has always required two phases : encoding the input sentences from the source language and decoding them in the target language. In the years 2013 and 2014, multiple papers [Kalchbrenner and Blunsom, 2013; Sustkever *et al.*, 2014; Cho *et al.*, 2014b] proposed using neural networks to learn the conditional distribution between a target sentence and its source sentence. In that manner, they used a recurrent neural network (RNN) as the encoder and another RNN as the decoder. Such an approach called neural machine translation (NMT) possesses many advantages, notably its portability and its ease of training. It is more portable than previous machine translation (MT) models, because it does not need to memorize phrase tables and huge language models [Luong *et al.*, 2015], and it is easier to train because it can be trained with backpropagation over the whole model instead of requiring tuning sub-components separately [Bahdanau *et al.*, 2015]. However, one issue with this model from [Cho *et al.*, 2014b] called RNN Encoder-Decoder is that the encoder compresses all of the information from the input sentence into a fixed-length vector before decoding the output. This limits the amount of information used for translation and is a problem when trying to translate longer sentences, as was shown by [Cho *et al.*, 2014a]. In order to fix this problem, [Bahdanau *et al.*, 2015] invent RNNSearch. RNNSearch combines a bi-directional RNN with a soft alignment model, which effectively implements an attention mechanism. By allowing the decoder to possess an attention mechanism, the decoder can "see" more information, even if the encoder still produces fixed-length vectors. The theory behind attention mechanisms will be explored further in section 2, but it should already be clear that they are an important key to developping better NMT models.

1.2 Contributions

Building on the work from [Bahdanau *et al.*, 2015] who invented RNNSearch, [Luong *et al.*, 2015] seek to :

1. simplify attention-based recurrent neural networks like RNNSearch,
2. investigate which attention mechanisms are most suited to the NMT task.

They create an improved model in Matlab for NMT using an English-German translation dataset, and perform an ablation study showing which parts of their model improve performance the most.

The objective of our research is to produce a similar ablation study of their model, rewritten using the Pytorch library, on an English-French dataset. We use a different dataset, so we do not expect the same scores for our model, but we should still be able to show the usefulness of the attention mechanism in an NMT model with our ablation study.

In section 2 we lay the theoretical foundations necessary for understanding this paper, in section 3 we detail the experiments performed, in section 4 we present the results and analysis of the experiments, and in section 5 we criticize our own approach to learning about the field of NMT.

2 Theoretical Foundations

The models we investigate are recurrent neural networks based on stacking Long-short-term-memory (**LSTM**) cells. In addition, some models include an **attention** mechanism. Some of the models are subject to an **input-feeding** approach. Some models are trained using **dropout**. All models are scored using the **BLEU score**. These concepts are explained in this section before detailing the experiments in section 3.

2.1 LSTM

LSTM cells (see Figure 1) are an elegant way for recurrent neural networks to remember both short-term and long-term information about sequences of inputs x_t. They do so by introducing the concept of a memory unit m_t which can only be modified through the action of an input gate i_t (write potential input s_t to memory) and a forget gate f_t (delete from memory). Whereas the hidden unit h_{t-1} gets modified at every timestep and thus contains the most recent information about the sequence, the memory unit can preserve information from further in the past following the will of the input and forget gates. The memory unit's knowledge is combined with the output gate unit o_t to generate the hidden unit h_t passed to the next timestep and outputted from the LSTM cell, thus improving the memory of the whole network. The equations of an LSTM cell are as follows [Bouktif *et al.*, 2020], where σ represents the sigmoid function, $\odot$ represents an element-by-element product, and W_i, W_f, W_s, W_o, U_i, U_f, U_s, U_o, b_i, b_f, b_s, b_o are the learnable parameters :

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{1}$$
$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{2}$$
$$s_t = \sigma(W_s x_t + U_s h_{t-1} + b_s) \tag{3}$$
$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{4}$$
$$m_t = f_t \odot m_{t-1} + i_t \odot s_t \tag{5}$$
$$h_t = o_t \odot \tanh(m_t) \tag{6}$$

Each LSTM cell is a part of a recurrent neural network composed of many timesteps from left to right, and it is also possible to feed the outputs of an LSTM network to another LSTM network by stacking them vertically. In that case, they are aptly named stacked LSTM networks, as seen in Figure 2.

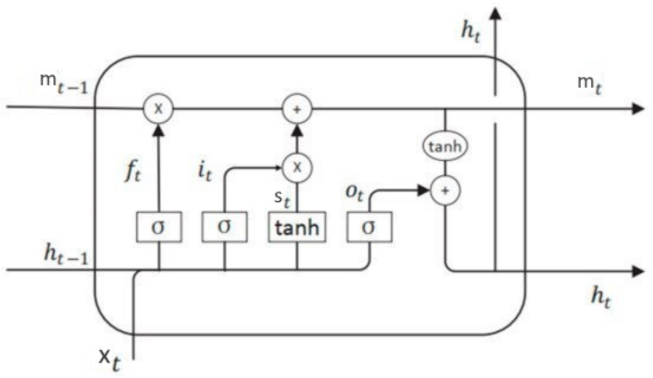

Figure 1: Visual representation of an LSTM cell. Inputs (x_t) come from the bottom, results of previous timestep (m_{t-1}, h_{t-1}) come from the left, results passed to next timestep (m_t, h_t) travel to the right and outputs of the LSTM cell (h_t) travel to the top. Reproduced and modified from [Bouktif *et al.*, 2020].

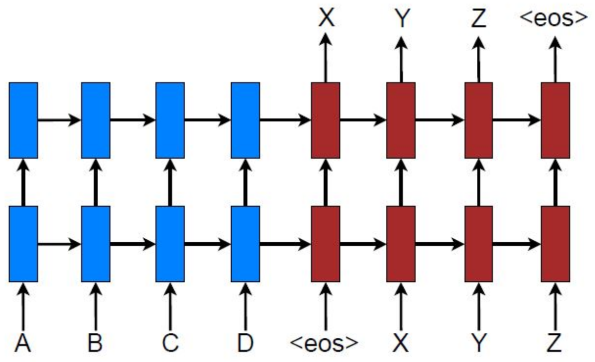

Figure 2: A double-layered stacked LSTM network used for machine translation. Each blue and red box is an LSTM cell. Timesteps in blue correspond to the source sentence and timesteps in red correspond to the target sentence. Reproduced from [Luong *et al.*, 2015].

2.2 Attention

In general terms, attention is a mechanism by which semantic links between elements of a vector can be highlighted. The specific equations for attention used in this paper are described in section 3 for each model. In this section, we explore the intuition behind attention mechanisms through an example. Many applications of attention exist, but in this paper we are only interested in computing attention between each word of the target sentence and the entire source sentence. For example, examine the following sentence and its translation in French, where the colors identify the correspondence between translated words :

English :

A	little	girl	climbing	into	a
0	1	2	3	4	5
wooden	treehouse.				
6	7				

French :

Une 0	petite 1	fille 2	grimpant 3	dans 4	une 5
maisonnette 6	en 7	bois. 8			

A recurrent machine translation model translates each word of the target sentence in sequence. To translate each word accurately, the model needs to identify which words from the source sentence are relevant, and it can do so with an attention mechanism. In the example, the attention for the second word of the target sentence *petite*, which means *little*, needs to be on *little* **and** *girl*, because *little* can be translated as a masculine or feminine word in French, but in this case *girl* indicates that it must be feminine. In simpler cases such as for the word *grimpant* in position 3, the attention only needs to be on the word *climbing*, which is also in position 3 of the source sentence, because it is a direct translation with no other dependencies. In this manner, the model proceeds to compute attention for each of the target sentence's words with respect to the source sentence. Attention is especially useful when the order of words in the target sentence are switched with respect to the source sentence, as we can see from *wooden treehouse* being translated to *maisonnette en bois*.

In recurrent models with attention, for each word of the **target** sentence, the model calculates an attentional vector of the same length as the **source** sentence or smaller depending on whether global attention or local attention is used respectively. For the models studied in this paper, attention implies calculating a context vector $\mathbf{c}_t$ for each target word by using the source sentence. This context vector can be concatenated with the target hidden state produced by the LSTM and passed through a **tanh** layer with parameters $\mathbf{W}_c$ to produce the attentional vector $\tilde{\mathbf{h}}_t$:

$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}_c[\mathbf{c}_t; \mathbf{h}_t]) \quad (7)$$

Each translated word is predicted by passing the attentional vector $\tilde{\mathbf{h}}_t$ through a softmax layer with parameters $\mathbf{W}_s$.

2.3 Input-feeding

The input-feeding approach requires feeding the attentional vectors $\tilde{h}_t$ (Notice the ˜ distinguishing the attentional vectors from the hidden LSTM units) as inputs to the next timestep by concatenating them with the other inputs (see Figure 3) The goal of such a maneuver is to inform the model about past alignment decisions [Luong *et al.*, 2015].

2.4 Dropout

Dropout is a common technique to prevent machine learning models from overfitting. It consists of masking some model parameters randomly during training by forcefully setting them to zero (temporarily) with probability p. This forces the model to create stable pathways for the information flow, so that even if some information is lost through these masked parameters, the model can still make accurate decisions. In the case of LSTM recurrent networks, however, it is crucial to apply dropout only between LSTM layers, not between successive timesteps, as shown in Figure 4. Otherwise, in the case of very long sequences, most of the information would be corrupted by dropout before reaching the end of the sequence, rendering the training of the model impossible [Zaremba *et al.*, 2014].

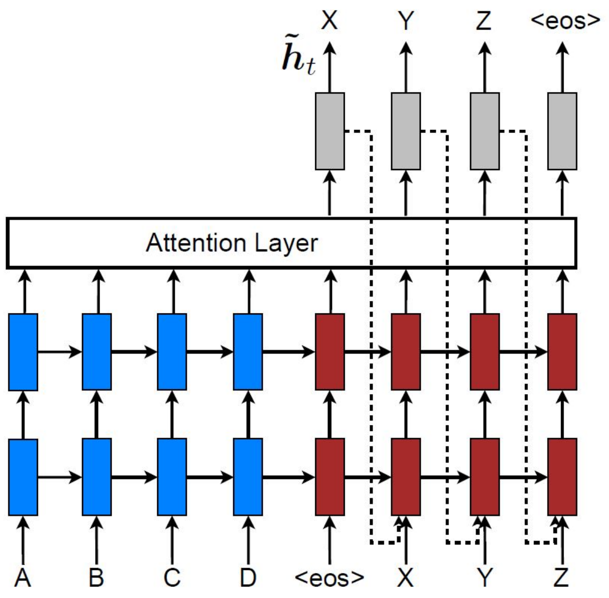

Figure 3: Input-feeding approach. Attentional vectors $\tilde{h}_t$ are fed as inputs to the next timesteps. Reproduced from [Luong *et al.*, 2015].

2.5 BLEU Score

All of the models are evaluated and compared with the Bilingual Evaluation Understudy (BLEU) score [Papineni *et al.*, 2002]. The BLEU score's calculation is based on comparing n-grams from the source and target sentences, where an n-gram is a subsequence of n consecutive words. To calculate the BLEU-N score, it is first necessary to calculate the 1-gram precision, the 2-gram precision, . . . , the N-gram

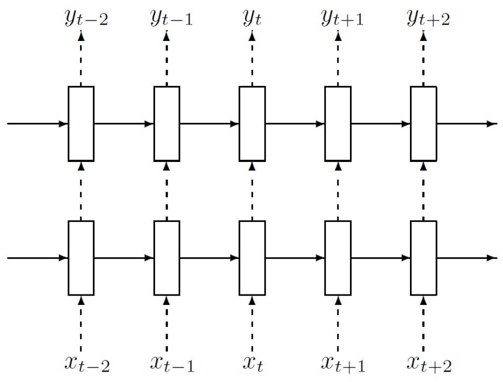

Figure 4: Dropout in a double-layered stacked recurrent network. Weights represented with dotted lines are subject to dropout whereas solid lines aren't. Reproduced from [Zaremba *et al.*, 2014].

precision for some value of N, and then perform geometric averaging over these precisions. As each of these n-gram precisions lie between 0 and 1 since they are normalized by the total number of words in the sentence, the final BLEU score also lies between 0 and 1. It is nevertheless common to report this score after multiplying it by 100 for easier readability.

3 Experiments

The models are trained and tested on the Multi30k dataset, using the BLEU-4 score for performance evaluation. This dataset is composed of 29 000 training sentences translated in four languages (English, Deutsch, French and Czech), 1014 validation sentences, and 1000 test sentences [Elliott *et al.*, 2016]. All sentences combined contain approximately 400k distinct words in English and 437k in French. The words, symbols, and punctuation are tokenized with the advanced natural language processing library spaCy [Honnibal *et al.*, 2020]. In the following sections, we describe the specificities of each subsequent model considered in the ablation study, detailing their additions to previous models.

3.1 Base LSTM

The base model is a single-layered LSTM network with 1024 hidden units. Contrary to models used in [Luong *et al.*, 2015], we do not employ a stacking architecture, because we found that it tends to overfit on our comparatively smaller dataset. The inputs to the model are embedded using 300 dimensions. We train the model for 25 epochs, using a batch size of 32 and the ADAM optimizer with a learning rate of 3×10^{-4}.

3.2 Reverse

The source sequence is input in reverse order: the last token is input first and the first token is input last (See Figure 5). The idea is to circumvent the usual long-term memory loss problem of recurrent networks by bringing the beginning of the source sentence closer to the beginning of the target sentence. In doing so, not only do the first few words of a sentence get translated more accurately, but so should the rest of the sentence. [Sustkever *et al.*, 2014]

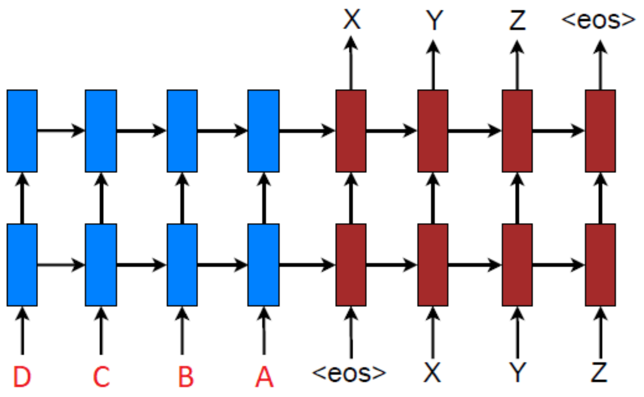

Figure 5: In this doubled-layered stacking LSTM network, the sequence of the input sentence is fed in backward. Reproduced and modified from [Luong *et al.*, 2015].

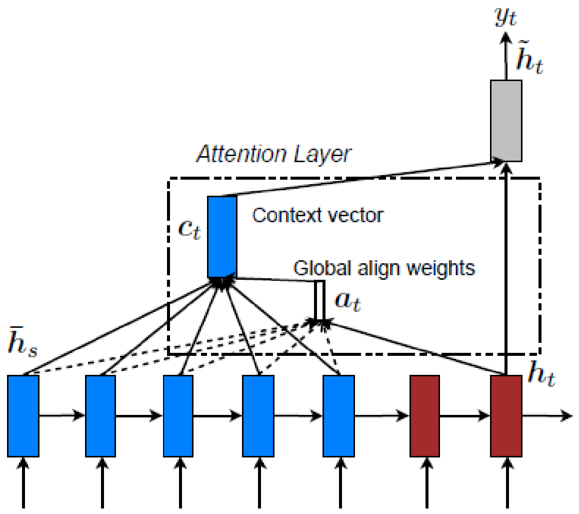

Figure 6: Architecture of a global attention layer for a stacking LSTM network. Reproduced from [Luong *et al.*, 2015].

3.3 Dropout

Three models with dropout probabilities $p = [0.05,\ 0.1,\ 0.2]$ are trained to soft optimize this hyper-parameter.

3.4 Global attention

The global attention model (see Figure 6) takes as input the current target hidden state $\mathbf{h}_t$ and all source hidden states $\bar{\mathbf{h}}_s$ and creates an alignment weight vector α_t. With $\mathbf{v}_a$ and $\mathbf{W}_a$ being learnable parameters, and $[\ \cdot\ ;\ \cdot\ ]$ representing a concatenation, the alignment weight vector is given by:

$$\alpha_t(s) = \frac{\exp\left(\text{score}(\mathbf{h}_t, \bar{\mathbf{h}}_s)\right)}{\sum_{s'} \exp\left(\text{score}(\mathbf{h}_t, \bar{\mathbf{h}}_{s'})\right)}, \tag{8}$$

$$\text{score}(\mathbf{h}_t, \bar{\mathbf{h}}_s) = \mathbf{v}_a^\intercal \tanh\left(\mathbf{W}_a[\mathbf{h}_t; \bar{\mathbf{h}}_s]\right) \tag{9}$$

The context vector $\mathbf{c}_t$ is then calculated as the weighted average of the source states $\bar{\mathbf{h}}_s$ using the alignment weight vector α_t. As explained in section 2.2, this context vector generates an attentional vector $\tilde{\mathbf{h}}_t$ using equation (7), and this attentional vector is in turn passed to a softmax layer to output the predicted word.

3.5 Local attention

The local attention model (see Figure 7) takes a window of the source hidden states $\bar{\mathbf{h}}_s$ instead of accounting for all of them. The main idea is to first predict the alignment between the target sentence and the source sentence with p_t, then account for nearest words more heavily than farther words using Gaussian weights. The parameter p_t is called the predictive alignment and is defined as follows:

$$p_t = S \cdot \text{sigmoid}(\mathbf{v}_p^\intercal \tanh \mathbf{W}_p\mathbf{h}_t)). \tag{10}$$

The local alignment vector $\boldsymbol{a}_t(s)$ is defined using the

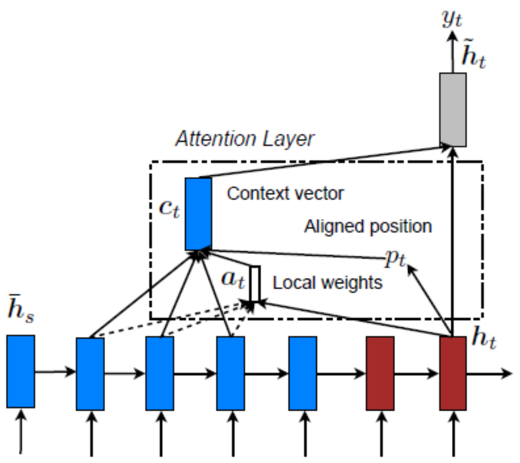

Figure 7: Architecture of a Local Attention on a stacking LSTM network. Reproduced from [Luong *et al.*, 2015].

global alignment vector $\alpha_t(s)$ from equation (8) and the predictive alignment p_t:

$$\boldsymbol{a}_t(s) = \text{align}\left(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s\right) \exp\left(-\frac{(s-p_t)^2}{2\sigma^2}\right). \qquad (11)$$

The context vector $\mathbf{c}_t$ is calculated as the weighted average of the source states $\bar{\mathbf{h}}_s$ using the local alignment weight vector $\boldsymbol{a}_t$. The hyper-parameter σ in equation (11) defines the width of the Gaussian distribution and the tested values are [1, 3, ..., 19, 21]. Since p_t is derivable, the gradient can still be backpropagated through the model during learning.

3.6 Feed Input

The final component added to the network is the input-feeding mechanism explained in section 2.3 and figure 3.

4 Results

The source code of our experiments can be found on this Github repository. The experiments are split into two parts, 1) Pre-processing the data and 2) Training the Neural Machine Translation models. The first loads the Multi30k dataset and tokenizes the sentences. This process takes about 30 minutes. The tokenized dataset is then saved in the same repository as *.pt* files, and loaded for the second part. This part implements the experiments detailed in section 3 using Pytorch and torchtext.

Dropout	BLEU score
0.05	34.61
0.10	30.66
0.20	29.78

Table 1: BLEU score for different dropout probabilities on the LSTM network with reverse input.

Table 2 presents the performance of the experiments and the results show a progressive increase in BLEU score for most add-ons to the models.

The Reverse does not show a great improvement on this dataset, but it has shown significant improvement in [Luong *et al.*, 2015]. This can be explained by the length of sentences in the dataset. We use the Multi30k dataset, where sentences contain an average of 15 tokens, whereas in the original paper, they use the WMT'14 dataset, containing much longer sentences. In their case, the need for long-term memory is an issue which can be resolved using Reverse, but since our sentences are so short, Reverse does not impact performance.

Dropout with probability ($p = 0.05$) increases the BLEU score (+ 2.51), but the model is very sensitive to the dropout probability. A small change in p significantly changes the BLEU score as shown in table 1.

The attention mechanisms, both global and local increase the BLEU scores. The global attention increases the BLUE score by (+5.74) over the model with Dropout, as expected.

Input-feeding shows a decrease in BLEU score (-0.33), but in [Luong *et al.*, 2015], the BLEU score increases. It is possible that the BLEU score in table 2 is not precise enough, since it is calculated from 1000 sentences, which might not be enough for a strong conclusion. It is also possible that for shorter sentences, letting the model know about past alignment decisions doesn't provide much of an improvement.

By using local attention instead of global attention, the BLEU score is increased by (+1.91) over the previous model. The standard deviation used for the Gaussian is $\sigma = 15$ which is larger than the average sentence length. This means that local attention focuses on the whole input sentence, but gives more attention near the predictive alignment position p_t. For that reason, it would be more apt to call it *gaussian global attention* or *modified global attention*, since it should act similar to global attention.

Figure 8 presents the validation curves of the models, and shows a great reduction of loss between the models with attention and the ones without. This clearly illustrates the power of attention in NMT. It also highlights that the models start to overfit the data between 5 and 10 epochs.

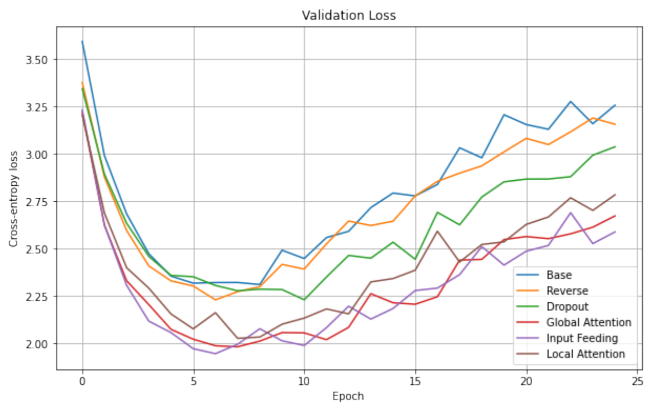

Figure 8: Validation Loss of the models on the Multi30k validation set

NMT model	BLEU score
Base	32.09
Base + Reverse	32.10
Base + Reverse + Dropout	34.61
Base + Reverse + Dropout + Global Attention	40.35
Base + Reverse + Dropout + Global Attention + Feed Input	40.03
Base + Reverse + Dropout + Local Attention + Feed Input	41.94

Table 2: Flick2016 French-English results - BLEU score of the various models on 1000 sentences. The Dropout probability is 0.05 and the standard deviation is 15 when applicable.

	Tag	Sentence
1	Source	Un homme avec un chapeau orange regardant quelque chose.
	Target	A man in an orange hat starring at something.
	Translated	A man with an orange hat looking at something.
2	Source	Un terrier de boston court sur l' herbe verdoyante devant une clôture blanche.
	Target	A boston terrier is running on lush green grass in front of a white fence.
	Translated	A marathon - colored runs on grass grass in front of a white fence.
3	Source	Une fille en tenue de karaté brisant un bâton avec un coup de pied.
	Target	A girl in karate uniform breaking a stick with a front kick.
	Translated	A girl in a karate uniform is a a stick with a kick .
4	Source	Des gens réparent le toit d'une maison.
	Target	People are fixing the roof of a house.
	Translated	People people are standing around the roof of a house.

Table 3: Examples of translated sentences from our best model.

The best model is tested on the Flickr 2016 French-English test dataset and some examples of translated sentences are shown in table 3. The translation is not always exact, but the inaccurate translations are most likely due to the lack of certain words in the training dataset (e.g. ”boston terrier” or ”fixing”). Nonetheless, the results are quite impressive for the small dataset we used to train our model.

5 Discussion

The first step we took to approach Neural Machine Translation was to thoroughly read Luong's paper [Luong *et al.*, 2015] to understand how to replicate their results. This led us to read the main references of the article, including Bahdanau's paper [Bahdanau *et al.*, 2015], Cho's papers [Cho *et al.*, 2014a; Cho *et al.*, 2014b] and others. From these papers we were able to understand the various models and get a grasp on NMT.

Coding the NMT model was a bigger challenge, therefore we started by watching an online tutorial by Alladin Perrson on sequence-to-sequence translation for a Deutsch-English dataset and we adapted it to our project.

Another part of the project was to determine what dataset to select to train and test the models. We started with the Europarl corpus dataset [Koehn, 2005], but loading the dataset and tokenizing it was too long for a project of this scale. This dataset has 2 million sentences, which is quite larger than Multi30k, the dataset we ended up choosing for this project which has 30 thousand sentences.

6 Conclusion

The results obtained show that an attention based mechanism significantly improves the BLEU score of Neural Machine Translation on a French-English dataset. Our models yield an increase of more then 5 BLEU over non-attention models. A small change that could increase the performance would be to add an adaptive learning rate such as halving it every few epochs. These models could also be trained on a larger dataset and compared to other papers, but the time restriction and the computational limitation restrained further research.

References

[Bahdanau *et al.*, 2015] Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, (November), 2015.

[Bouktif *et al.*, 2020] Salah Bouktif, Ali Fiaz, Ali Ouni, and Mohamed Adel Serhani. Multi-sequence lstm-rnn deep learning and metaheuristics for electric load forecasting. *Energies*, 13(2), 2020.

[Cho *et al.*, 2014a] KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *CoRR*, abs/1409.1259, 2014.

[Cho *et al.*, 2014b] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and

Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014.

[Elliott *et al.*, 2016] Desmond Elliott, Stella Frank, Khalil Sima'an, and Lucia Specia. Multi30k: Multi-lingual english-german image descriptions. *CoRR*, abs/1605.00459, 2016.

[Honnibal *et al.*, 2020] Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. spaCy: Industrial-strength Natural Language Processing in Python, 2020.

[Kalchbrenner and Blunsom, 2013] Nal Kalchbrenner and Phil Blunsom. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

[Koehn, 2005] Philipp Koehn. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86. Citeseer, 2005.

[Luong *et al.*, 2015] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective Approaches to Attention-based Neural Machine Translation. *Conference Proceedings - EMNLP 2015: Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, 2015.

[Papineni *et al.*, 2002] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics.

[Sustkever *et al.*, 2014] Ilya Sustkever, Oriol Vinyals, and Quoc V. Le. Sequence to Sequence Learning with Neural Networks. In *NIPS*, 2014.

[Zaremba *et al.*, 2014] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *CoRR*, abs/1409.2329, 2014.