Convolutional Sequence to Sequence Learning in English to French Neural Machine Translation

Yingke Wang
University of Toronto
yingkewang.wang@mail.utoronto.ca

Jun Xing
University of Toronto
jun.xing@mail.utoronto.ca

Tianyu Zhang *
University of Toronto
tianyutheodosia.zhang@mail.utoronto.ca

Abstract

This paper focuses on improving the training time performance of the Bi-LSTM model in Neural Machine Translation. We used CNN with attention such that it can make full use of parallelization and layers in convolutional neural networks, so that the optimized model will be faster. We trained both Bi-LSTM model and ConS2S model on the WMT'20 English-French dataset. The result shows that not only the training time of ConS2S is much faster than Bi-LSTM model on GPU, ConS2S also outperforms Bi-LSTM on translation accuracy evaluated by the BLEU score.

1 Introduction

Since 2014, deep learning neural networks have successfully reshaped the practices in Machine Translation. We would like to utilize what we learnt in CSC413 and apply it in this task. Traditionally, seq2seq models with RNN Encoder and Decoder were introduced as models usually used to perform neural machine translation tasks. However, RNNs may not be able to process sentences with complicated relationships of information, as it takes linear time complexity to capture long-range dependencies. In the lecture, LSTM is introduced to solve the problem by utilizing activations and weights as short-term and long-term memories. Nevertheless, LSTM are not able to be parallelized, since the temporal unrolling part needs to be done sequentially, which slows down its runtime to a large extent. In our project, we will solve those problems by introducing Convolutional Seq2Seq Learning as another method to address the problem of capturing long-ranged information, since the layers of CNN models make it possible to process dependencies among separated words in sentences. As CNN models can be fully parallelized, it can run much faster. In this paper, we will compare ConS2S model with standard Bi-LSTM model in English to French Neural Machine Translation.

2 Related Works

2.1 Bi-LSTM With Attention

Bi-LSTM is a RNN model that is widely used in speech recognition and translation. Being bidirectional makes it able to get both future information and past information by concatenating two hidden layers of opposite directions to the same output. Compared to vanilla RNN, LSTM is better at

*Authors listed in alphabetical order.

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

handling long sequence of input by utilizing input gate, output gate, and forget gate. Specifically, the input gate is used to update the cell status; the output gate determines the value for the next hidden state, which contains the information on the previous inputs; the forget gate decides what information needs information and what can be ignored. Moreover, we added an attention mechanism to the model, which allows each time step of decoder to focus on the encoder states that are more relevant. However, this model takes long time to train as it is limited by its potential for parallel computation.

3 Methods and Algorithm

Instead of using traditional recurrent components, the paper (Auli et al. [2017]) proposes a Convolutional Sequence to Sequence Model (ConS2S) that uses convolution layers for feature extraction. The encoder encodes the input sentences in source language into a context vector. The decoder decodes the context vector into the target language. In order to capture information from different representation subspaces, Multi-head attention is added to emphasize which encoder state should pay more attention at. Based on our understanding of the paper, we designed and implemented a similar model with that from the paper. An overview of our model as well as its components are displayed in Appendix. As shown in **Figure 2** A.3, the encoder and decoder architecture are almost the same with minor difference in padding and additional attention for decoder. Let x be input elements (a sentence in source language). It first embeds x in distributional space $w = (w_1, w_2, ..., w_m)$ on each words value. It also do a position embedding $p = (p_1, p_2, ..., p_m)$ as a sense of order. Adding up w and p as input element representation $e = (e_1, e_2, ..., e_m)$. Convert e into a a vector of hidden dimension $h = (h_1, h_2, ..., h_3)$. Passing h into A number of Convolution blocks. Encoder and Decoder treat the structure of its CNN blocks and output of them differently.

As for encoder convolution block [**Figure 3** A.3], it first add padding = kernel // 2 to keep the dimension same before and after convolution. After convolution, It apply GLU as activation function

$$v([A, B]) = A \odot \sigma(B), \tag{1}$$

and then add residual connection to the result, which is the input of next convolutional block.

After encoder convolution blocks, it change the Output of the last CNN block from hidden dimension to embedding dimension, calling it $OutputLastEncoderCNNBlock$. Adding residual connection from embedding layer, call it $OutputEncoder$. We need to use them in decoder CNN block

As for decoder convolution block[figure 3], it first add padding at front to exclude the target word, so it makes sure that our decoder won't know what is the target word, preventing the model from cheating. It then goes through convolution layer and GLU activation as encoder CNN block. It then calculate the attention on this layer which is

$$Softmax((ConvOut + embedding) \times OutputLastEncoderCNNBlock). \tag{2}$$

Get the attention added output be $Attention \times OutputEncoder$ and change attention added output dimension of embedding back to hidden. Add residual connection and use it as input to next CNN layer.

After decoder convolution blocks, it change the Output of the last CNN block from hidden dimension to embedding dimension, applying dropout for regularization. and change the output from embedding dimension to vocabulary dimension of the target language. Compare to the probability for each of word in vocabulary with the actual target sentence to get the loss.

In paper, it also add scale term to regularize for output of each convolution blocks and residual connection to ensure that the variance throughout the network does not change dramatically.

4 Experiment Setup

4.1 Training Data and Preprocess

We gathered our data from WMT20[2], which regularly releases worldwide standard corpus for Machine Translation tasks. The raw dataset “News Commentary V15” contains 350,728 pairs of English-French sentences, and we randomly sampled 80,000 of training pairs, 20,000 of validation pairs, and 10,000 of test pairs of sentences. The preprocess that we performed on sentence strings mainly includes cleaning, tokenization, and adding special tokens such as <SOS> and <EOS> to the beginning and end of each sentence. In order to convert a sentence string to a vector, we would build two vocabulary dictionaries to index each token that appears in training data.
Hardware Resources NVIDIA GEFORCE RTX 3060Ti (8GB Memory, 1.41 GHz)
Details of our Models and Main Hyperparameters can be found in **Table 2** A.3 in Appendix.

4.2 Evaluation Metric

To compare the performance among different models, we mainly focus on losses, training clock time and Bilingual Evaluation Understudy (BLEU). BLEU is a standard metric which measures the similarity between predicted sentences and ground true sentences. Our implementation of BLEU can be found in the Appendix 1.

5 Results and Analysis

We evaluate the performance of both the Bi-LSTM and the ConS2S models. Trainable parameters, average BLEU and training clock time for the test set are shown in the following table.

Comparison between Bi-LSTM and ConS2S			
Model	# of Trainable Parameters	**BLEU**	Training Time (s)
Bi-LSTM	43,160,425	**29.08**	4,589
ConS2S Model 1	16,326,180	32.95	1,117
ConS2S Model 2	29,463,564	33.22	1,265
ConS2S Model 3	42,045,329	**33.72**	1,841

Table 1: Performance of Bi-LSTM and ConS2S on NVIDIA GEFORCE RTX 3060Ti

5.1 Accuracy

For the entire testing set, all of our three ConS2S models outperforms the Bi-LSTM. Particularly, BLEU scoreConS2S Model 3 outperforms the Bi-LSTM model by 4.64 BLEU score (shown in **Table 1** 5). An example of translation of one example sentence is included in **Table 3** A.3 in the Appendix, from which we observed that all three ConS2S models have higher BLEU scores than the Bi-LSTM model, with an average of 35.68, compared with the BLEU score of the result from Google Translation, which is 35.29. As shown in **Figure 1** 5, ConS2S models generally converges faster than Bi-LSTM. This is not only because there are more trainable parameters in our LSTM model, but also the convolutional layers and muti-head attention extracted better features.

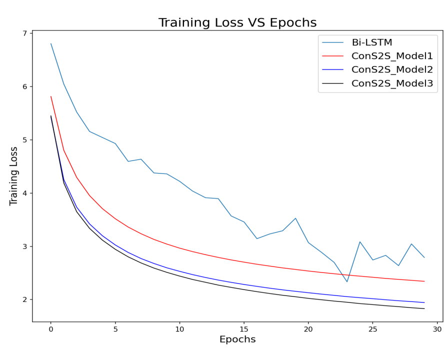

Figure 1: Training Loss v.s. Epochs

[2]https://www.statmt.org/wmt20/translation-task.html

5.2 Training Time

By comparing Bi-LSTM and ConS2S Model 3, which have similar number of trainable parameters trained on the same GPU for the same dataset with same number of epochs. However, the training time for ConsS2S Model 3 is significantly less than that for the Bi-LSTM model. As mentioned before, ConS2S can effectively utilize parallelization, and does not require the unrolling operation contained in standard Seq2seq models. It is cause by the model sees all the word in a sentence at the same time and can parallelly processing them as a regular convolution network. Whereas Bi-LSTM, it feed each word of a sentence one after another in a sequence. Thus, It will cost more time to train.

5.3 Sensitivity of Hyperparameters

Compare ConS2S Model 1 and ConS2S Model 2: The only difference between ConS2S Model 1 and ConS2S Model 2 is the embedding size. Model 1 has embedding size 100. Model 2 has embedding size 256. Based on our finding shown in Table 1, it seems that the BLUE score and training time are very similar even Model 2 has lots more parameters than Model 1. In Figure 1, we can see that, the training loss of Model 2 is always less than Model 1. and after 30 epochs, loss of Model 2 is 0.5 less than loss of Model 1. Therefore, we can see that model 2 converge faster than model 1. Such trend makes sense because when the model has larger embedding size, it can capture the word better since it has more parameter to get represented; hence the model can learn detail and more features during translating at training. However, during testing seeing new data, such minor features may not be necessary to capture since the testing data may coming from a different field and the minor features learned at training may not be useful.

Compare ConS2S Model 2 and ConS2S Model 3: The only difference between ConS2S Model 2 and ConS2S Model 3 is the hidden layer size. Model 2 has hidden layer size 256. Model 3 has hidden layer size 512. Based on our finding shown in Table 1, it seems that the BLUE score is higher and training time is less than Model 3, and it has lots more parameters than Model 2. In Figure 1, we can see that, the training loss of Model 3 is only a little bit less than Model 2. Such trend makes sense because when the Convolutional block has larger hidden size, it will have more parameters to learn during the training. Since the translation pattern is captured inside the CNN blocks; hence, bigger hidden size will increase model's translation capability. Hence, the model can perform better in the test case/ real world translation.

6 Conclusion and Significance

Our experiment results showed that ConS2S is effective in saving the training time of performing Neural Machine Translation tasks on WMT'20 English-French dataset to a large extent, as the CNN architecture enables parallelization. Moreover, the translation results from the ConS2S models also outperforms those from the LSTM model. One limitation of our experiment is that we only trained our models on the dataset of texts from the news. Therefore, the results might not be able to be generalized to translation of sentences from other professions. In the future studies, similar approach can be utilized in other sequence to sequence leanring problems, such as text summarization and image captioning, to address the limitation of unable to be parallelized due to the temporal unrolling part of the traditional Seq2seq models.

References

[1] Auli, M., Dauphin, Y.N., Gehring, J., Grangier, D., & Yarats, D. (2017) Convolutional Sequence to Sequence Learning. arXiv: 1705.03122v3.

[2] Hung, P.T., & Yamanishi, K. (2008) Word2vec Skip-gram Dimensionality Selection via Sequential Normalized Maximum Likelihood. arXiv: 2008.07720.

[3] Yin, Z, & Shen, Y. (2018) On the Dimensionality of Word Embedding. arXiv:1812.04224.

A Appendix

A.1 Contribution

Yingke Wang ConS2S implementation, ConS2S training and testing, Methods and Algorithm, results and analysis, summary

Jun Xing Bi-LSTM implementation, Bi-LSTM training and testing, data preprocessing, experiment setup, results and analysis, summary

Tianyu Zhang Bi-LSTM implementation, Bi-LSTM training and testing, abstract, introduction, related works, results and analysis, summary

A.2 Algorithm

A.2.1 BLEU Score

Algorithm 1 BLEU Score

For each candidate, find the reference with **the most similar in length**
$c_i \leftarrow$ the length of the i^{th} candidate
$r_i \leftarrow$ the nearest length among the references
$brevity_i \leftarrow \frac{f_i}{c_i}$
if $brevity_i < 1$ **then**
| $BP_i = 1$
else
| $BP_i = e^{1-brevity_i}$
end
$BLEU_C \leftarrow BP_C \times (p_1 p_2 \ldots p_n)^{\frac{1}{n}}$ ▷ p_n is the n-gram precision

A.3 Figures and Tables

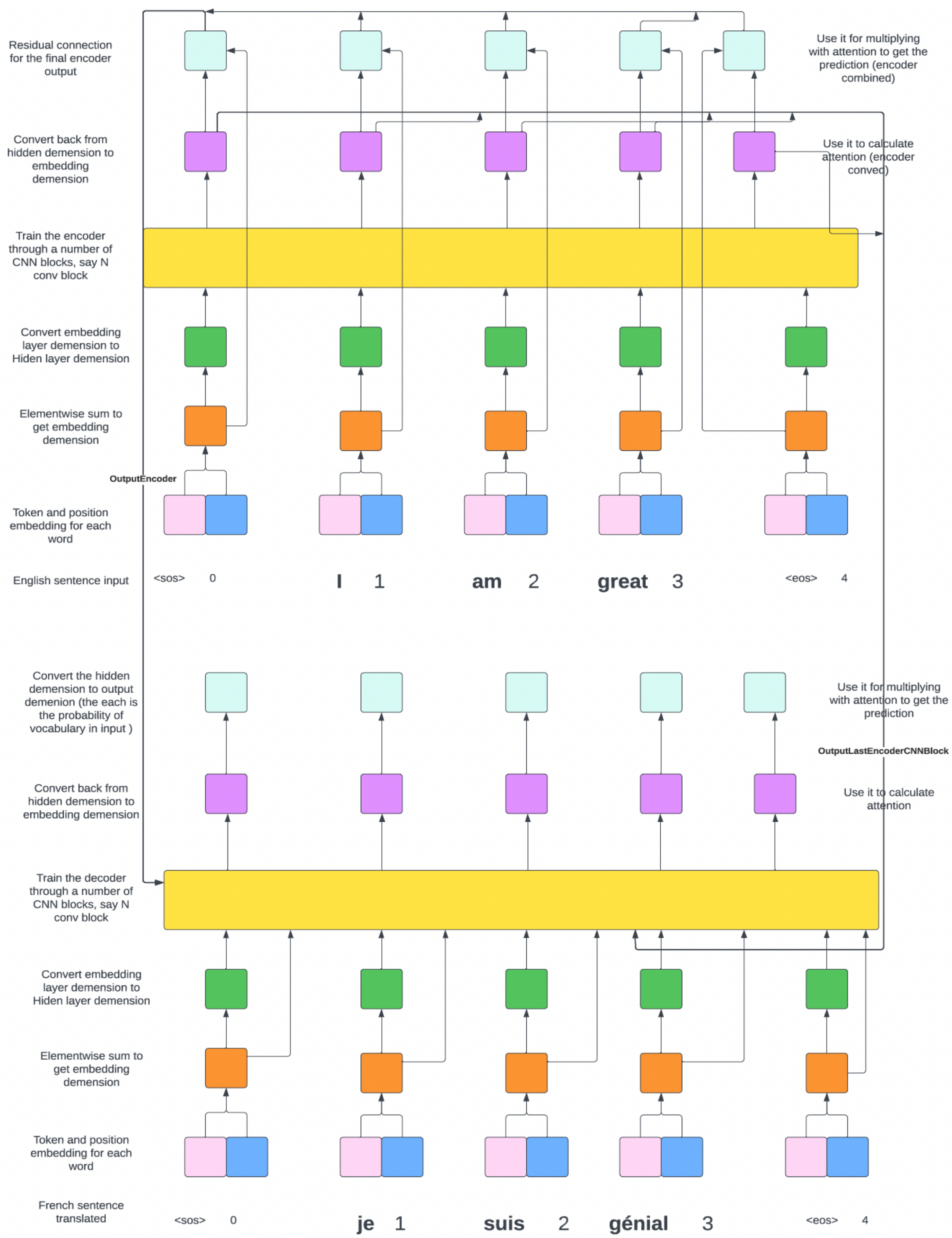

Figure 2: ConS2S Model Architecture

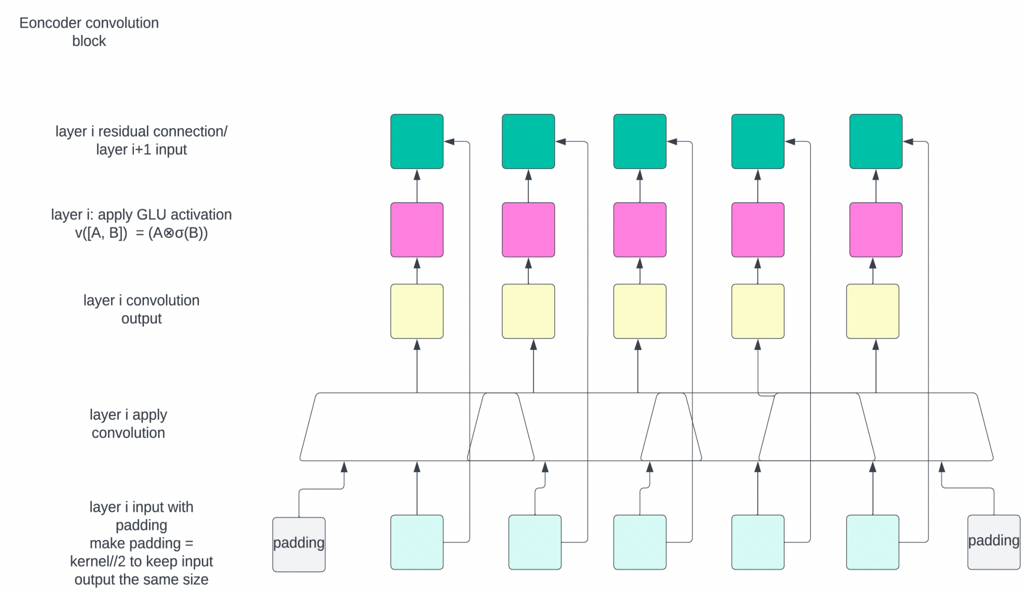

Figure 3: Encoder Convolution Block

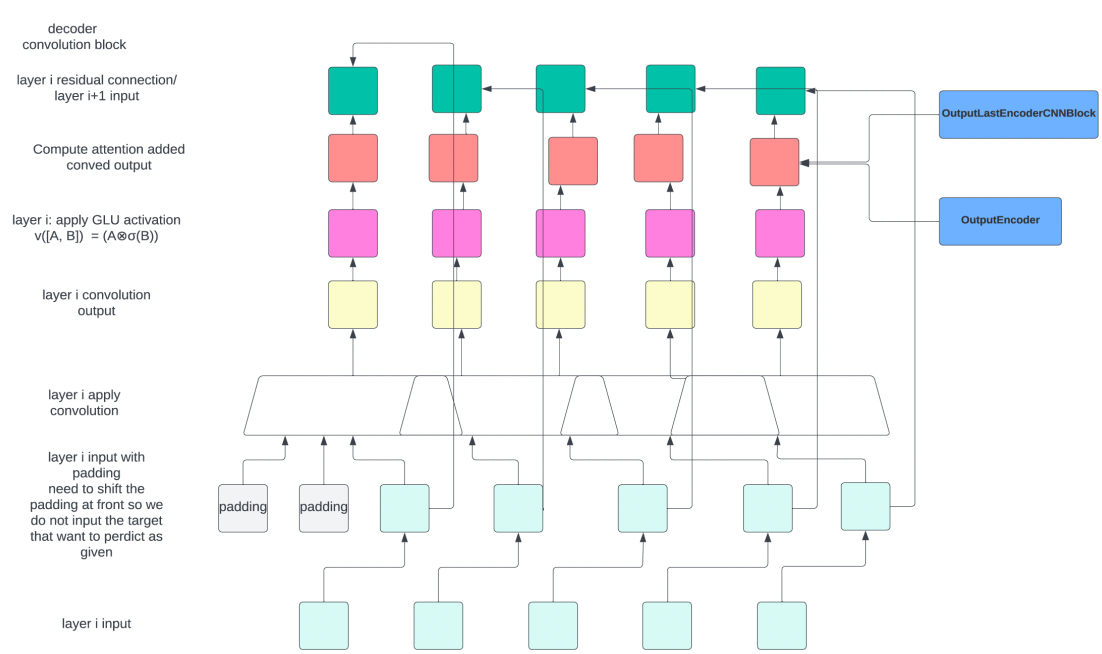

Figure 4: Decoder Convolution Block

Models and Main Hyperparameters	
Model	Hyperparameters
Bi-LSTM	(Embedding size = 256, Hidden size = 128) + Dot Attention
ConS2S Model 1	Embedding size = 100, Hidden size = 256, Number of Hidden Layers = 10, Kernel Size = 3, Dropout = 0.25
ConS2S Model 2	Embedding size = 256, Hidden size = 256, Number of Hidden Layers = 10, Kernel Size = 3, Dropout = 0.25
ConS2S Model 3	Embedding size = 256, Hidden size = 512, Number of Hidden Layers = 10, Kernel Size = 3, Dropout = 0.25

Table 2: Models and Main Hyperparameters

Comparison of Translation Result			
Text Type	Model	Content	**BLEU**
Source (En)		**But everyone would also be in better shape today if Bush had been able to provide the main questions of his interlocutors.**	
Target (Fr)		**Mais les relations seraient aussi meilleures si Bush avait été capable de donner une suite concrète aux principales questions abordées avec ses interlocuteurs.**	
Prediction (Fr)	Bi-LSTM	Mais le monde se trouverait également dans dans mieux plus dans le si Bush avait été capable de répondre à les principaux interlocuteurs de ont pris se sentir avec lui.	**31.27**
	ConS2S Model 1	Mais tout le monde serait aussi en meilleure forme aujourd'hui si bush avait été capable de fournir les principaux questions ses interlocuteurs.	**38.10**
	ConS2S Model 2	Mais tout le monde serait également dans le meilleur moment que bush avait été en mesure de fournir en compte les principaux questions de ses interlocuteurs.	33.45
	ConS2S Model 3	Mais tout le monde serait aussi en train d' être plus forme aujourd'hui si bush avait été capable de fournir les principaux questions de ses interlocuteurs.	35.48
Google Translate (Fr)		Mais tout le monde serait aussi en meilleure forme aujourd'hui si Bush avait pu fournir les principales interrogations de ses interlocuteurs.	**35.29**

Table 3: Comparison of Translation Result