Testing the SliceNet with SentencePiece

Simon Untergasser

Potsdam University
Matr.-Nr. 794948
untergasser@uni-potsdam.de

Abstract. The SliceNet Architecture (Kaiser et al. (2017)) was recently proposed for the task of Neural Machine Translation. In the original paper the authors used a BytePair Encoding following the approach in Sennrich et al. (2015). In this work we investigate the performance of the SliceNet in combination with SentencePiece (Google (2018)). We test the Model with different vocabulary sizes and different embedding sizes and provide a comparison of the results.

1 Introduction and related work

Lately recurrent neural networks have become the state-of-the-art systems for neural machine translation. LSTM-based models like Wu et al. (2016) are widely used and tested in many applications. It is even claimed that the gap between human and machine translation is narrowed significantly. While the results are truly remarkable, due to the recurrent architecture, these models are expensive to train. Even more recent research shows, that autoregressive and attention based convolutional neural networks can perform comparable or even better than previous recurrent models. Fully convolutional architectures (Kalchbrenner et al. (2016)) can be trained in parallel on GPUs and therefore speed up the training manifold. Since normal convolution on high dimensional data require millions of operations the authors of Chollet (2017) showed, that the convolution operation can be separated in two simpler convolutions with less parameters. They showed that using the depthwise separable convolution increases speed while giving similar results to normal convolutions. Inspired by this success the authors of Kaiser et al. (2017) build the SliceNet which uses depthwise separable convolution for the task of machine translation. For their experiments they use the BytePair Encoding described in Sennrich et al. (2015). In this paper the SliceNet architecture is reimplemented, hinting on minor flaws in the original paper. Furthermore the BytePair Encoding is replaced by SentencePiece (Google (2018)) with different vocabulary sizes. The Model is trained on the Europarl English-German corpus. The main focus of this work is on investigating how the vocabulary size of the SentencePiece Encoding in combination with the embedding size in the model effects the quality of the results. For this purpose, experiments with vocabulary sizes of 4000, 8000 and 16000 tokens were conducted. For each of these numbers the embedding size was varied between 300 and 1200 dimensions.

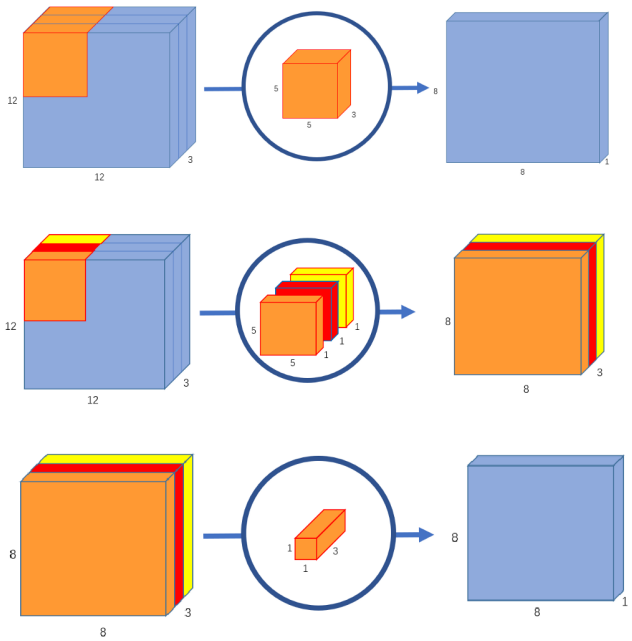

Fig. 1: Normal convolution over 2 spatial and 1 channel dimension

The paper is organized as follows: Chapter 2 describes the architecture of the SliceNet along with the flaws in the original paper. In chapter 3 the experiments are described. The results are presented in chapter 4 and chapter 5 concludes the paper.

2 Separable convolutions, SliceNet and SentencePiece

The core building block of the SliceNet Architecture is the so called Depthwise Separable Convolution described in detail in Chollet (2017). The authors analyze the Inception (Szegedy et al. (2015), Szegedy et al. (2016), Szegedy et al. (2017)) Architecture and argue, that the Inception module in its extremes becomes in its core a depthwise separable convolution.

2.1 Separable convolution

The normal convolution operates on one (for text processing) or on two (for image processing) spatial dimensions and on one channel dimension. The filter therefore have to learn spatial and cross-channel correlations (see Figure 1 top). But the spatial dimensions and the channel dimensions often have little or no correlation at all. So the main idea behind the Inception module was, to split the

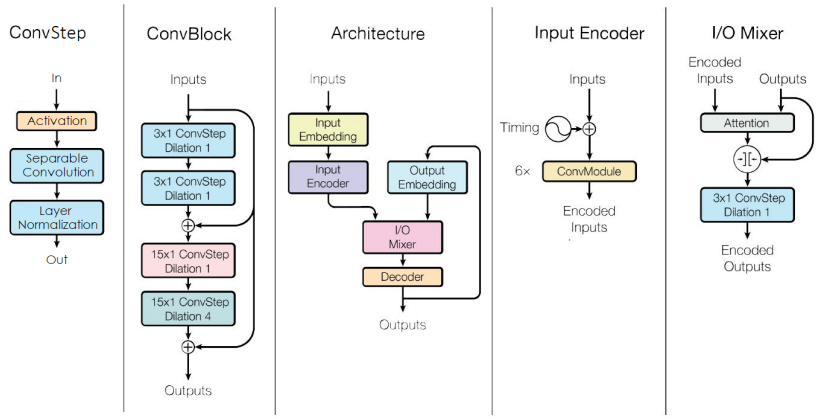

Fig. 2: The SliceNet architecture with all the building blocks

one operation on spatial and channel dimensions into two separate operations. The first operation being a pointwise convolution over all channels pictured in figure 1 middle row. This operation handles the cross-channel correlations. The second operation is the depthwise convolution shown at the bottom of figure 1. These filters learn the spatial correlations.

2.2 SliceNet

In Kaiser et al. (2017) the authors built a so called `ConvStep` around the separable convolution as shown in figure 2 on the left side. It consist of a ReLu activation followed by the Separable Convolution followed by Layer normalization. This `ConvStep` is then used in the `ConvBlock` also called `ConvModule`. As can be seen in in figure 2, the `ConvBlock` consists of 4 `ConvSteps` with different filter sizes and dilation rates. While the first 2 `ConvSteps` have rather small filters of size 3x1 the second 2 filters have 15x1. With the last filter having additionally a dilation rate of 4 its receptive field reaches 60 items. The `ConvBlock` also adds residual connections to the architecture. The overall model consists of an encoder structure, which consists of the embedding, a positional encoding and 6 `ConvBlocks`. The encoded input then goes into the Input/Output-Mixer. There the input and the decoder output are mixed over an attention module. The details and formulas can be found in Kaiser et al. (2017) and are not included here due to space constrains. The decoder on the other hand needs a little bit more description since the figure in the original SliceNet paper contains a minor flaw, which complicates the implementation when not handled carefully. In the original figure it looks like the decoder handles only one input which is the output of the I/O-Mixer. But the formulas state:

$$AttnConvModule(x, source) = ConvModule(x) + attention(source, x)$$
$$Decoder(x) = AttnConvModule^4(x, InputEncoder(inputs))$$

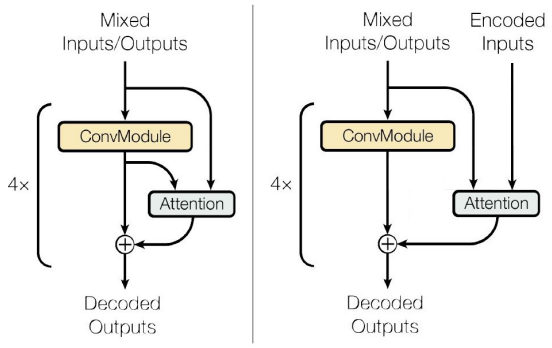

Fig. 3: The decoder figure from the original paper on the left and the corrected version on the right

Here one can see, that the Decoder consists of a concatenation of a ConvBlock and the attention module. The ConvBlock takes the I/O-Mixer output and the attention module takes the same but *additionally* the output of the encoder. The corrected version along with the original is shown in figure 3.

2.3 SentencePiece

SentencePiece was developed by members of google and was presented in 2018. It is an unsupervised text tokenizer based on subword units like byte-pair-encoding or an unigram language model. It can handle raw sentences and is language independent. For the training of the SentencePiece model, one can choose a vocabulary size, the subword algorithm and some extra options like predefined ids for special tokens. Additionally the whitespace is treated as a basic symbol.

3 Experiments and results

This section presents the conducted experiments and the corresponding results. The goal of this work was, to test different vocabulary sizes for SentencePiece with varying embedding sizes in the SliceNet model. Table 1 gives an overview over the chosen numbers. Next the data and the preprocessing is described. Then the results are presented.

3.1 Data and preprocessing

For all the experiments the Europarl.v7 english-german corpus (Koehn (2005)) was used. It consists of 1,929,209 parallel sentences. All these sentences are used to train a SentencePiece model with shared vocabulary. That means, after the training there is one vocabulary for both languages. The raw sentences are then

Vocab Size	Embed Size	# Parameters
4000	300	8,454,700
	500	19,288,500
	800	43,339,200
	1000	64,573,000
8000	300	12,058,700
	500	25,292,500
	800	52,943,200
	1000	76,577,000
16000	300	19,266,700
	500	37,300,500
	800	72,151,200
	1000	100,585,000

Table 1: Vocabulary sizes and embedding sizes and corresponding parameter counts for the models used in the experiments

encoded using the SentencePiece model and all sentences longer than 40 tokens are excluded to reduce training time.

3.2 Model parameters

As stated above, the purpose of this work is to compare the performance of the SliceNet architecture with different vocabulary and embedding sizes. The used numbers are listed in Table 1. Of course, these parameters influence the model dimensions greatly. The same table gives an overview over the parameter counts depending on model dimensions.

All models are trained for 40 epochs with 5% of the training data used for validation. Since model dimensions in terms of parameter counts are very high, the training data was reduced to 500,000 parallel sentences. Still the computing time for one epoch ranged from 621 seconds for the smallest model (`vocab_size=4000`, `embed_size=300`) to 4260 seconds (1.18 hours) for the biggest model (`vocab_size=16000`, `embed_size=1000`).

3.3 Results

All models reached a training accuracy of about 70%. This score is way higher than the score reported in Kaiser et al. (2017). They reported accuracies of about 64%. The reported parameter counts are also higher. This is probably due to the restricted sentence length o f only 40 tokens used in this work. Their best model reached a BLEU score of 26.1 with a parameter count of 253 million. The BLEU scores reached in this work are presented in table 2. While not in the

Vocab Size	Embed Size	# Parameters
4000	300	4.6
	500	5.9
	800	4.5
	1000	3.2
8000	300	7.8
	500	9.2
	800	7.4
	1000	7.1
16000	300	7.7
	500	10.7
	800	6.7
	1000	5.0

Table 2: BLEU scores of all models

range of the original model, one can clearly see the trend of increased BLEU score with increasing vocabulary size. Generally the model with embedding size of 500 seems to perform better than the other sizes.

With increasing embedding size, the BLEU score slightly decays, which could be an hint on to less training time. In general, the models were not fully converged after 40 epochs, but due to time restrictions this number was fixed.

4 Conclusion and Future Work

The results indicate, that using SentencePiece for the SliceNet architecture is indeed a feasible approach and could be investigated further. The model reaches a high accuracy and BLEU scores that promise some potential. Still there are some problems which will be addressed in the following. The Implementation of the SliceNet was not straight forward as the author of this paper expected. The original paper lacks a lot of details, so that experimenting with parameters and architecture decisions was necessary. In the end, time restrictions permitted only to train all models for 40 epochs. From the results one can clearly see, that the performance of the model does not reach the performance of the original. On the one hand, this is due to smaller sentence length, using only part of the training data and restricted length of training. On the other hand, the tested models all have the same architecture, meaning for example, the dilation rate was fixed as shown in figure 2. The big dilation rate of 4 together with the window size of 15 gives the model a receptive field of 60 tokens. This makes not much sense with a sentence size restricted to 40 tokens. Future work could include using a smaller dilation rate inspired by the numbers in the original SliceNet paper. They also tried models with all dilations set to 1. The next improvement would be to train the model until convergence. For this purpose, it makes sense to only train one model and evaluate the results.

Bibliography

Chollet, F. (2017), Xception: Deep learning with depthwise separable convolutions, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 1251–1258.

Google (2018), 'Sentencepiece'. [Online; accessed February 18, 2019]. `https://github.com/google/sentencepiece`

Kaiser, L., Gomez, A. N. & Chollet, F. (2017), 'Depthwise separable convolutions for neural machine translation', *arXiv preprint arXiv:1706.03059* .

Kalchbrenner, N., Espeholt, L., Simonyan, K., Oord, A. v. d., Graves, A. & Kavukcuoglu, K. (2016), 'Neural machine translation in linear time', *arXiv preprint arXiv:1610.10099* .

Koehn, P. (2005), Europarl: A parallel corpus for statistical machine translation, *in* 'MT summit', Vol. 5, pp. 79–86.

Sennrich, R., Haddow, B. & Birch, A. (2015), 'Neural machine translation of rare words with subword units', *arXiv preprint arXiv:1508.07909* .

Szegedy, C., Ioffe, S., Vanhoucke, V. & Alemi, A. A. (2017), Inception-v4, inception-resnet and the impact of residual connections on learning, *in* 'Thirty-First AAAI Conference on Artificial Intelligence'.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. & Rabinovich, A. (2015), Going deeper with convolutions, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 1–9.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. & Wojna, Z. (2016), Rethinking the inception architecture for computer vision, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 2818–2826.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K. et al. (2016), 'Google's neural machine translation system: Bridging the gap between human and machine translation', *arXiv preprint arXiv:1609.08144* .