

Neural Machine Translation in Linear Time

Nal Kalchbrenner Lasse Espeholt Karen Simonyan Aäron van den Oord Alex Graves Koray Kavukcuoglu
Google Deepmind, London UK
nalk@google.com

Abstract

We present a novel neural network for processing sequences. The ByteNet is a one-dimensional convolutional neural network that is composed of two parts, one to encode the source sequence and the other to decode the target sequence. The two network parts are connected by stacking the decoder on top of the encoder and preserving the temporal resolution of the sequences. To address the differing lengths of the source and the target, we introduce an efficient mechanism by which the decoder is dynamically unfolded over the representation of the encoder. The ByteNet uses dilation in the convolutional layers to increase its receptive field. The resulting network has two core properties: it runs in time that is linear in the length of the sequences and it sidesteps the need for excessive memorization. The ByteNet decoder attains state-of-the-art performance on character-level language modelling and outperforms the previous best results obtained with recurrent networks. The ByteNet also achieves state-of-the-art performance on character-to-character machine translation on the English-to-German WMT translation task, surpassing comparable neural translation models that are based on recurrent networks with attentional pooling and run in quadratic time. We find that the latent alignment structure contained in the representations reflects the expected alignment between the tokens.

1. Introduction

In neural language modelling, a neural network estimates a distribution over sequences of words or characters that belong to a given language (Bengio et al., 2003). In neural machine translation, the network estimates a distribution over sequences in the target language conditioned on a given sequence in the source language. The network can be thought of as composed of two parts: a *source network* (the encoder) that encodes the source sequence into a representation and a *target network* (the decoder) that uses the

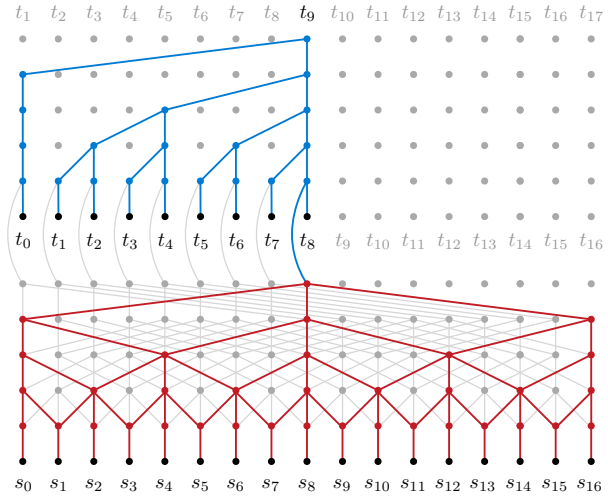


Figure 1. The architecture of the ByteNet. The target decoder (blue) is stacked on top of the source encoder (red). The decoder generates the variable-length target sequence using dynamic unfolding.

representation of the source encoder to generate the target sequence (Kalchbrenner & Blunsom, 2013).

Recurrent neural networks (RNN) are powerful sequence models (Hochreiter & Schmidhuber, 1997) and are widely used in language modelling (Mikolov et al., 2010), yet they have a potential drawback. RNNs have an inherently serial structure that prevents them from being run in parallel along the sequence length during training and evaluation. Forward and backward signals in a RNN also need to traverse the full distance of the serial path to reach from one token in the sequence to another. The larger the distance, the harder it is to learn the dependencies between the tokens (Hochreiter et al., 2001).

A number of neural architectures have been proposed for modelling translation, such as encoder-decoder networks (Kalchbrenner & Blunsom, 2013; Sutskever et al., 2014; Cho et al., 2014; Kaiser & Bengio, 2016), networks with attentional pooling (Bahdanau et al., 2014) and two-dimensional networks (Kalchbrenner et al., 2016a). Despite the generally good performance, the proposed models

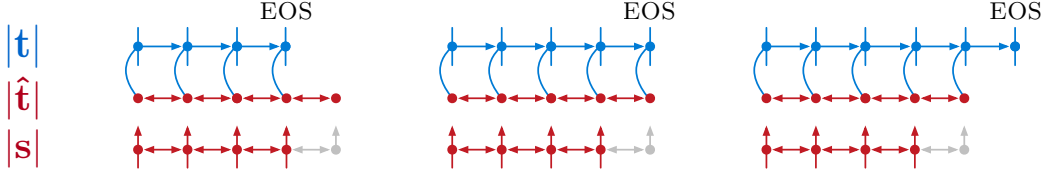


Figure 2. Dynamic unfolding in the ByteNet architecture. At each step the decoder is conditioned on the source representation produced by the encoder for that step, or simply on no representation for steps beyond the extended length $|\hat{\mathbf{t}}|$. The decoding ends when the target network produces an end-of-sequence (EOS) symbol.

either have running time that is super-linear in the length of the source and target sequences, or they process the source sequence into a constant size representation, burdening the model with a memorization step. Both of these drawbacks grow more severe as the length of the sequences increases.

We present a family of encoder-decoder neural networks that are characterized by two architectural mechanisms aimed to address the drawbacks of the conventional approaches mentioned above. The first mechanism involves the *stacking* of the decoder on top of the representation of the encoder in a manner that preserves the temporal resolution of the sequences; this is in contrast with architectures that encode the source into a fixed-size representation (Kalchbrenner & Blunsom, 2013; Sutskever et al., 2014). The second mechanism is the *dynamic unfolding* mechanism that allows the network to process in a simple and efficient way source and target sequences of different lengths (Sect. 3.2).

The ByteNet is the instance within this family of models that uses one-dimensional convolutional neural networks (CNN) of fixed depth for both the encoder and the decoder (Fig. 1). The two CNNs use increasing factors of dilation to rapidly grow the receptive fields; a similar technique is also used in (van den Oord et al., 2016a). The convolutions in the decoder CNN are masked to prevent the network from seeing future tokens in the target sequence (van den Oord et al., 2016b).

The network has beneficial computational and learning properties. From a computational perspective, the network has a running time that is *linear* in the length of the source and target sequences (up to a constant $c \approx \log d$ where d is the size of the desired dependency field). The computation in the encoder during training and decoding and in the decoder during training can also be run efficiently *in parallel* along the sequences (Sect. 2). From a learning perspective, the representation of the source sequence in the ByteNet is *resolution preserving*; the representation sidesteps the need for memorization and allows for maximal bandwidth between encoder and decoder. In addition, the distance traversed by forward and backward signals between any input and output tokens corresponds to the fixed depth of the networks and is largely independent of the dis-

tance between the tokens. Dependencies over large distances are connected by short paths and can be learnt more easily.

We apply the ByteNet model to strings of characters for character-level language modelling and character-to-character machine translation. We evaluate the decoder network on the Hutter Prize Wikipedia task (Hutter, 2012) where it achieves the state-of-the-art performance of 1.31 bits/character. We further evaluate the encoder-decoder network on character-to-character machine translation on the English-to-German WMT benchmark where it achieves a state-of-the-art BLEU score of 22.85 (0.380 bits/character) and 25.53 (0.389 bits/character) on the 2014 and 2015 test sets, respectively. On the character-level machine translation task, ByteNet betters a comparable version of GNMT (Wu et al., 2016a) that is a state-of-the-art system. These results show that deep CNNs are simple, scalable and effective architectures for challenging linguistic processing tasks.

The paper is organized as follows. Section 2 lays out the background and some desiderata for neural architectures underlying translation models. Section 3 defines the proposed family of architectures and the specific convolutional instance (ByteNet) used in the experiments. Section 4 analyses ByteNet as well as existing neural translation models based on the desiderata set out in Section 2. Section 5 reports the experiments on language modelling and Section 6 reports the experiments on character-to-character machine translation.

2. Neural Translation Model

Given a string \mathbf{s} from a source language, a neural translation model estimates a distribution $p(\mathbf{t}|\mathbf{s})$ over strings \mathbf{t} of a target language. The distribution indicates the probability of a string \mathbf{t} being a translation of \mathbf{s} . A product of conditionals over the tokens in the target $\mathbf{t} = t_0, \dots, t_N$ leads to a tractable formulation of the distribution:

$$p(\mathbf{t}|\mathbf{s}) = \prod_{i=0}^N p(t_i|t_{<i}, \mathbf{s}) \quad (1)$$

Each conditional factor expresses complex and long-range dependencies among the source and target tokens. The strings are usually sentences of the respective languages; the tokens are words or, as in the our case, characters. The network that models $p(\mathbf{t}|\mathbf{s})$ is composed of two parts: a source network (the encoder) that processes the source string into a representation and a target network (the decoder) that uses the source representation to generate the target string (Kalchbrenner & Blunsom, 2013). The decoder functions as a language model for the target language.

A neural translation model has some basic properties. The decoder is autoregressive in the target tokens and the model is sensitive to the ordering of the tokens in the source and target strings. It is also useful for the model to be able to assign a non-zero probability to any string in the target language and retain an open vocabulary.

2.1. Desiderata

Beyond these basic properties the definition of a neural translation model does not determine a unique neural architecture, so we aim at identifying some desiderata.

First, the running time of the network should be *linear* in the length of the source and target strings. This ensures that the model is scalable to longer strings, which is the case when using characters as tokens.

The use of operations that run *in parallel* along the sequence length can also be beneficial for reducing computation time.

Second, the size of the source representation should be linear in the length of the source string, i.e. it should be *resolution preserving*, and not have constant size. This is to avoid burdening the model with an additional memorization step before translation. In more general terms, the size of a representation should be proportional to the amount of information it represents or predicts.

Third, the path traversed by forward and backward signals in the network (between input and output tokens) should be short. Shorter paths whose length is largely decoupled from the sequence distance between the two tokens have the potential to better propagate the signals (Hochreiter et al., 2001) and to let the network learn long-range dependencies more easily.

3. ByteNet

We aim at building neural language and translation models that capture the desiderata set out in Sect. 2.1. The proposed ByteNet architecture is composed of a decoder that is *stacked* on an encoder (Sect. 3.1) and generates variable-length outputs via *dynamic unfolding*

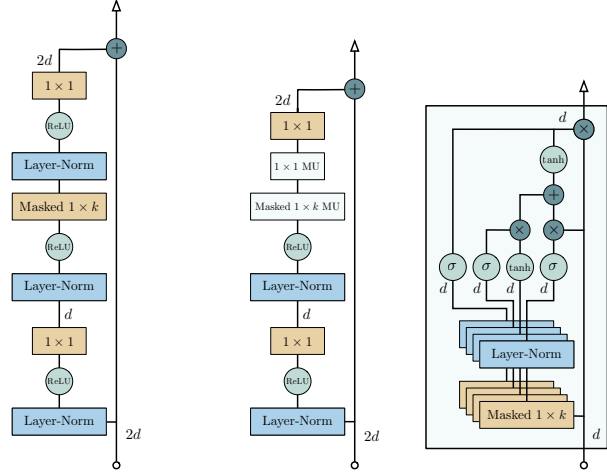


Figure 3. Left: Residual block with ReLUs (He et al., 2016) adapted for decoders. Right: Residual Multiplicative Block adapted for decoders and corresponding expansion of the MU (Kalchbrenner et al., 2016b).

(Sect. 3.2). The decoder is a language model that is formed of one-dimensional convolutional layers that are masked (Sect. 3.4) and use dilation (Sect. 3.5). The encoder processes the source string into a representation and is formed of one-dimensional convolutional layers that use dilation but are *not* masked. Figure 1 depicts the two networks and their combination.

3.1. Encoder-Decoder Stacking

A notable feature of the proposed family of architectures is the way the encoder and the decoder are connected. To maximize the representational bandwidth between the encoder and the decoder, we place the decoder on top of the representation computed by the encoder. This is in contrast to models that compress the source representation into a fixed-size vector (Kalchbrenner & Blunsom, 2013; Sutskever et al., 2014) or that pool over the source representation with a mechanism such as attentional pooling (Bahdanau et al., 2014).

3.2. Dynamic Unfolding

An encoder and a decoder network that process sequences of different lengths cannot be directly connected due to the different sizes of the computed representations. We circumvent this issue via a mechanism which we call dynamic unfolding, which works as follows.

Given source and target sequences \mathbf{s} and \mathbf{t} with respective lengths $|\mathbf{s}|$ and $|\mathbf{t}|$, one first chooses a sufficiently tight upper bound $\hat{|\mathbf{t}|}$ on the target length $|\mathbf{t}|$ as a linear function of the source length $|\mathbf{s}|$:

$$\hat{|\mathbf{t}|} = a|\mathbf{s}| + b \quad (2)$$

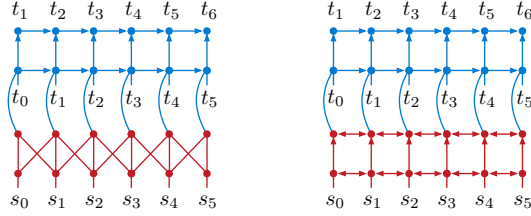


Figure 4. Recurrent ByteNet variants of the ByteNet architecture. Left: Recurrent ByteNet with convolutional source network and recurrent target network. Right: Recurrent ByteNet with bidirectional recurrent source network and recurrent target network. The latter architecture is a strict generalization of the RNN Enc-Dec network.

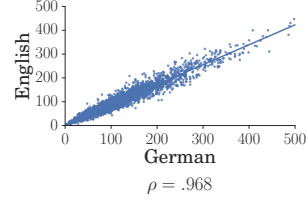


Figure 5. Lengths of sentences in characters and their correlation coefficient for the English-to-German WMT NewsTest-2013 validation data. The correlation coefficient is similarly high ($\rho > 0.96$) for all other language pairs that we inspected.

The tight upper bound $|\hat{\mathbf{t}}|$ is chosen in such a way that, on the one hand, it is greater than the actual length $|\mathbf{t}|$ in almost all cases and, on the other hand, it does not increase excessively the amount of computation that is required. Once a linear relationship is chosen, one designs the source encoder so that, given a source sequence of length $|\mathbf{s}|$, the encoder outputs a representation of the established length $|\hat{\mathbf{t}}|$. In our case, we let $a = 1.20$ and $b = 0$ when translating from English into German, as German sentences tend to be somewhat longer than their English counterparts (Fig. 5). In this manner the representation produced by the encoder can be efficiently computed, while maintaining high bandwidth and being resolution-preserving. Once the encoder representation is computed, we let the decoder unfold step-by-step over the encoder representation until the decoder itself outputs an end-of-sequence symbol; the unfolding process may freely proceed beyond the estimated length $|\hat{\mathbf{t}}|$ of the encoder representation. Figure 2 gives an example of dynamic unfolding.

3.3. Input Embedding Tensor

Given the target sequence $\mathbf{t} = t_0, \dots, t_n$ the ByteNet decoder embeds each of the first n tokens t_0, \dots, t_{n-1} via a look-up table (the n tokens t_1, \dots, t_n serve as targets for the predictions). The resulting embeddings are concatenated into a tensor of size $n \times 2d$ where d is the number of inner channels in the network.

3.4. Masked One-dimensional Convolutions

The decoder applies masked one-dimensional convolutions (van den Oord et al., 2016b) to the input embedding tensor that have a masked kernel of size k . The masking ensures that information from future tokens does not affect the prediction of the current token. The operation can be implemented either by zeroing out some of the weights of a wider kernel of size $2k - 1$ or by padding the input map.

3.5. Dilation

The masked convolutions use dilation to increase the receptive field of the target network (Chen et al., 2014; Yu & Koltun, 2015). Dilation makes the receptive field grow exponentially in terms of the depth of the networks, as opposed to linearly. We use a dilation scheme whereby the dilation rates are doubled every layer up to a maximum rate r (for our experiments $r = 16$). The scheme is repeated multiple times in the network always starting from a dilation rate of 1 (van den Oord et al., 2016a; Kalchbrenner et al., 2016b).

3.6. Residual Blocks

Each layer is wrapped in a residual block that contains additional convolutional layers with filters of size 1×1 (He et al., 2016). We adopt two variants of the residual blocks: one with ReLUs, which is used in the machine translation experiments, and one with Multiplicative Units (Kalchbrenner et al., 2016b), which is used in the language modelling experiments. Figure 3 diagrams the two variants of the blocks. In both cases, we use layer normalization (Ba et al., 2016) before the activation function, as it is well suited to sequence processing where computing the activation statistics over the following future tokens (as would be done by batch normalization) must be avoided. After a series of residual blocks of increased dilation, the network applies one more convolution and ReLU followed by a convolution and a final softmax layer.

4. Model Comparison

In this section we analyze the properties of various previously introduced neural translation models as well as the ByteNet family of models. For the sake of a more complete analysis, we include two recurrent ByteNet variants (which we do not evaluate in the experiments).

Model	Net _S	Net _T	Time	RP	Path _S	Path _T
RCTM 1	CNN	RNN	$ S S + T $	no	$ S $	$ T $
RCTM 2	CNN	RNN	$ S S + T $	yes	$ S $	$ T $
RNN Enc-Dec	RNN	RNN	$ S + T $	no	$ S + T $	$ T $
RNN Enc-Dec Att	RNN	RNN	$ S T $	yes	1	$ T $
Grid LSTM	RNN	RNN	$ S T $	yes	$ S + T $	$ S + T $
Extended Neural GPU	cRNN	cRNN	$ S S + S T $	yes	$ S $	$ T $
Recurrent ByteNet	RNN	RNN	$ S + T $	yes	$\max(S , T)$	$ T $
Recurrent ByteNet	CNN	RNN	$c S + T $	yes	c	$ T $
ByteNet	CNN	CNN	$c S + c T $	yes	c	c

Table 1. Properties of various neural translation models.

4.1. Recurrent ByteNets

The ByteNet is composed of two stacked encoder and decoder networks where the decoder network dynamically adapts to the output length. This way of combining the networks is not tied to the networks being strictly convolutional. We may consider two variants of the ByteNet that use recurrent networks for one or both of the networks (see Figure 4). The first variant replaces the convolutional decoder with a recurrent one that is similarly stacked and dynamically unfolded. The second variant also replaces the convolutional encoder with a recurrent encoder, e.g. a bi-directional RNN. The target RNN is then placed on top of the source RNN. Considering the latter Recurrent ByteNet, we can see that the RNN Enc-Dec network (Sutskever et al., 2014; Cho et al., 2014) is a Recurrent ByteNet where all connections between source and target – except for the first one that connects s_0 and t_0 – have been severed. The Recurrent ByteNet is a generalization of the RNN Enc-Dec and, modulo the type of weight-sharing scheme, so is the convolutional ByteNet.

4.2. Comparison of Properties

In our comparison we consider the following neural translation models: the Recurrent Continuous Translation Model (RCTM) 1 and 2 (Kalchbrenner & Blunsom, 2013); the RNN Enc-Dec (Sutskever et al., 2014; Cho et al., 2014); the RNN Enc-Dec Att with the attentional pooling mechanism (Bahdanau et al., 2014) of which there are a few variations (Luong et al., 2015; Chung et al., 2016a); the Grid LSTM translation model (Kalchbrenner et al., 2016a) that uses a multi-dimensional architecture; the Extended Neural GPU model (Kaiser & Bengio, 2016) that has a convolutional RNN architecture; the ByteNet and the two Recurrent ByteNet variants.

Our comparison criteria reflect the desiderata set out in Sect. 2.1. We separate the first (computation time) desider-

atum into three columns. The first column indicates the time complexity of the network as a function of the length of the sequences and is denoted by **Time**. The other two columns **Net_S** and **Net_T** indicate, respectively, whether the source and the target network use a convolutional structure (CNN) or a recurrent one (RNN); a CNN structure has the advantage that it can be run in parallel along the length of the sequence. The second (resolution preservation) desideratum corresponds to the **RP** column, which indicates whether the source representation in the network is resolution preserving. Finally, the third desideratum (short forward and backward flow paths) is reflected by two columns. The **Path_S** column corresponds to the length in layer steps of the shortest path between a source token and any output target token. Similarly, the **Path_T** column corresponds to the length of the shortest path between an input target token and any output target token. Shorter paths lead to better forward and backward signal propagation.

Table 1 summarizes the properties of the models. The ByteNet, the Recurrent ByteNets and the RNN Enc-Dec are the only networks that have linear running time (up to the constant c). The RNN Enc-Dec, however, does not preserve the source sequence resolution, a feature that aggravates learning for long sequences such as those that appear in character-to-character machine translation (Luong & Manning, 2016). The RCTM 2, the RNN Enc-Dec Att, the Grid LSTM and the Extended Neural GPU do preserve the resolution, but at a cost of a quadratic running time. The ByteNet stands out also for its **Path** properties. The dilated structure of the convolutions connects any two source or target tokens in the sequences by way of a small number of network layers corresponding to the depth of the source or target networks. For character sequences where learning long-range dependencies is important, paths that are sub-linear in the distance are advantageous.

Model	Inputs	Outputs	WMT Test '14	WMT Test '15
Phrase Based MT (Freitag et al., 2014; Williams et al., 2015)	phrases	phrases	20.7	24.0
RNN Enc-Dec (Luong et al., 2015)	words	words	11.3	
Reverse RNN Enc-Dec (Luong et al., 2015)	words	words	14.0	
RNN Enc-Dec Att (Zhou et al., 2016)	words	words	20.6	
RNN Enc-Dec Att (Luong et al., 2015)	words	words	20.9	
GNMT (RNN Enc-Dec Att) (Wu et al., 2016a)	word-pieces	word-pieces	24.61	
RNN Enc-Dec Att (Chung et al., 2016b)	BPE	BPE	19.98	21.72
RNN Enc-Dec Att (Chung et al., 2016b)	BPE	char	21.33	23.45
GNMT (RNN Enc-Dec Att) (Wu et al., 2016a)	char	char	22.62	
ByteNet	char	char	23.75	26.26

Table 2. BLEU scores on En-De WMT NewsTest 2014 and 2015 test sets.

Model	Test
Stacked LSTM (Graves, 2013)	1.67
GF-LSTM (Chung et al., 2015)	1.58
Grid-LSTM (Kalchbrenner et al., 2016a)	1.47
Layer-normalized LSTM (Chung et al., 2016a)	1.46
MI-LSTM (Wu et al., 2016b)	1.44
Recurrent Memory Array Structures (Rocki, 2016)	1.40
HM-LSTM (Chung et al., 2016a)	1.40
Layer Norm HyperLSTM (Ha et al., 2016)	1.38
Large Layer Norm HyperLSTM (Ha et al., 2016)	1.34
Recurrent Highway Networks (Srivastava et al., 2015)	1.32
ByteNet Decoder	1.31

Table 3. Negative log-likelihood results in bits/byte on the Hutter Prize Wikipedia benchmark.

5. Character Prediction

We first evaluate the ByteNet Decoder separately on a character-level language modelling benchmark. We use the Hutter Prize version of the Wikipedia dataset and follow the standard split where the first 90 million bytes are used for training, the next 5 million bytes are used for validation and the last 5 million bytes are used for testing (Chung et al., 2015). The total number of characters in the vocabulary is 205.

The ByteNet Decoder that we use for the result has 30 residual blocks split into six sets of five blocks each; for the five blocks in each set the dilation rates are, respectively, 1, 2, 4, 8 and 16. The masked kernel has size 3. This gives a receptive field of 315 characters. The number of hidden units d is 512. For this task we use residual multiplicative blocks (Fig. 3 Right). For the optimization we use Adam (Kingma & Ba, 2014) with a learning rate of 0.0003 and a weight decay term of 0.0001. We apply dropout to the last ReLU layer before the softmax dropping units with a probability of 0.1. We do not reduce the learning rate during training. At each step we sample a batch of sequences of 500 characters each, use the first 100 characters as the minimum context and predict the latter 400 characters.

	WMT Test '14	WMT Test '15
Bits/character	0.521	0.532
BLEU	23.75	26.26

Table 4. Bits/character with respective BLEU score achieved by the ByteNet translation model on the English-to-German WMT translation task.

Table 3 lists recent results of various neural sequence models on the Wikipedia dataset. All the results except for the ByteNet result are obtained using some variant of the LSTM recurrent neural network (Hochreiter & Schmidhuber, 1997). The ByteNet decoder achieves 1.31 bits/character on the test set.

6. Character-Level Machine Translation

We evaluate the full ByteNet on the WMT English to German translation task. We use NewsTest 2013 for validation and NewsTest 2014 and 2015 for testing. The English and German strings are encoded as sequences of characters; no explicit segmentation into words or morphemes is applied to the strings. The outputs of the network are strings of characters in the target language. We keep 323 characters in the German vocabulary and 296 in the English vocabulary.

The ByteNet used in the experiments has 30 residual blocks in the encoder and 30 residual blocks in the decoder. As in the ByteNet Decoder, the residual blocks are arranged in sets of five with corresponding dilation rates of 1, 2, 4, 8 and 16. For this task we use the residual blocks with ReLUs (Fig. 3 Left). The number of hidden units d is 800. The size of the kernel in the source network is 3, whereas the size of the masked kernel in the target network is 3. For the optimization we use Adam with a learning rate of 0.0003.

Each sentence is padded with special characters to the nearest greater multiple of 50; 20% of further padding is ap-

Director Jon Favreau, who is currently working on Disney's forthcoming Jungle Book film, told the website Hollywood Reporter: "I think times are changing."

Regisseur Jon Favreau, der derzeit an Disneys bald erscheinenden Dschungelbuch-Film arbeitet, sagte gegenüber der Webseite Hollywood Reporter: "Ich glaube, die Zeiten ändern sich."

Regisseur Jon Favreau, der zur Zeit an Disneys kommendem Jungle Book Film arbeitet, hat der Website Hollywood Reporter gesagt: "Ich denke, die Zeiten ändern sich".

Matt Casaday, 25, a senior at Brigham Young University, says he had paid 42 cents on Amazon.com for a used copy of "Strategic Media Decisions: Understanding The Business End Of The Advertising Business."

Matt Casaday, 25, Abschlussstudent an der Brigham Young University, sagt, dass er auf Amazon.com 42 Cents ausgegeben hat für eine gebrauchte Ausgabe von "Strategic Media Decisions: Understanding The Business End Of The Advertising Business."

Matt Casaday, 25, ein Senior an der Brigham Young University, sagte, er habe 42 Cent auf Amazon.com für eine gebrauchte Kopie von "Strategic Media Decisions: Understanding The Business End Of The Advertising Business".

Table 5. Raw output translations generated from the ByteNet that highlight interesting reordering and transliteration phenomena. For each group, the first row is the English source, the second row is the ground truth German target, and the third row is the ByteNet translation.

plied to each source sentence as a part of dynamic unfolding (eq. 2). Each pair of sentences is mapped to a bucket based on the pair of padded lengths for efficient batching during training. We use *vanilla* beam search according to the total likelihood of the generated candidate and accept only candidates which end in a end-of-sentence token. We use a beam of size 12. We do not use length normalization, nor do we keep score of which parts of the source sentence have been translated (Wu et al., 2016a).

Table 2 and Table 4 contain the results of the experiments. On NewsTest 2014 the ByteNet achieves the highest performance in character-level and subword-level neural machine translation, and compared to the word-level systems it is second only to the version of GNMT that uses word-pieces. On NewsTest 2015, to our knowledge, ByteNet achieves the best published results to date.

Table 5 contains some of the unaltered generated translations from the ByteNet that highlight reordering and other phenomena such as transliteration. The character-level aspect of the model makes post-processing unnecessary in principle. We further visualize the sensitivity of the ByteNet's predictions to specific source and target inputs using gradient-based visualization (Simonyan et al., 2013). Figure 6 represents a heatmap of the magnitude of the gradients of the generated outputs with respect to the source and target inputs. For visual clarity, we sum the gradients for all the characters that make up each word and normalize the values along each column. In contrast with the attentional pooling mechanism (Bahdanau et al., 2014), this general technique allows us to inspect not just dependencies of the outputs on the source inputs, but also dependencies of the outputs on previous target inputs, or on any other neural network layers.

7. Conclusion

We have introduced the ByteNet, a neural translation model that has linear running time, decouples translation from memorization and has short signal propagation paths for tokens in sequences. We have shown that the ByteNet decoder is a state-of-the-art character-level language model based on a convolutional neural network that outperforms recurrent neural language models. We have also shown that the ByteNet generalizes the RNN Enc-Dec architecture and achieves state-of-the-art results for character-to-character machine translation and excellent results in general, while maintaining linear running time complexity. We have revealed the latent structure learnt by the ByteNet and found it to mirror the expected alignment between the tokens in the sentences.

References

- Ba, Lei Jimmy, Kiros, Ryan, and Hinton, Geoffrey E. Layer normalization. *CoRR*, abs/1607.06450, 2016.
- Bahdanau, Dzmitry, Cho, Kyunghyun, and Bengio, Yoshua. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.
- Bengio, Yoshua, Ducharme, Réjean, Vincent, Pascal, and Jauvin, Christian. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.
- Chen, Liang-Chieh, Papandreou, George, Kokkinos, Iasonas, Murphy, Kevin, and Yuille, Alan L. Semantic image segmentation with deep convolutional nets and fully connected crfs. *CoRR*, abs/1412.7062, 2014.
- Cho, Kyunghyun, van Merriënboer, Bart, Gülçehre, Çağlar, Bougares, Fethi, Schwenk, Holger, and Bengio, Yoshua.

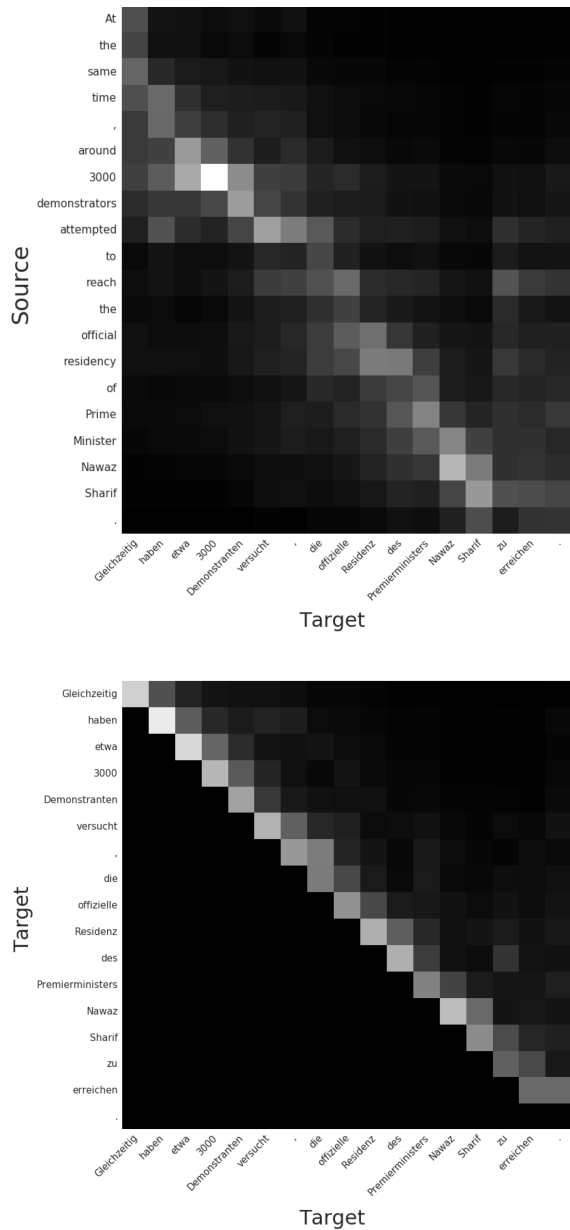


Figure 6. Magnitude of gradients of the predicted outputs with respect to the source and target inputs. The gradients are summed for all the characters in a given word. In the bottom heatmap the magnitudes are nonzero on the diagonal, since the prediction of a target character depends highly on the preceding target character in the same word.

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

Chung, Junyoung, Gülçehre, Caglar, Cho, Kyunghyun, and Bengio, Yoshua. Gated feedback recurrent neural networks. *CoRR*, abs/1502.02367, 2015.

Chung, Junyoung, Ahn, Sungjin, and Bengio, Yoshua. Hierarchical multiscale recurrent neural networks. *CoRR*, abs/1609.01704, 2016a.

Chung, Junyoung, Cho, Kyunghyun, and Bengio, Yoshua. A character-level decoder without explicit segmentation for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016*, 2016b.

Freitag, Markus, Peitz, Stephan, Wuebker, Joern, Ney, Hermann, Huck, Matthias, Sennrich, Rico, Durrani, Nadir, Nadejde, Maria, Williams, Philip, Koehn, Philipp, Hermann, Teresa, Cho, Eunah, and Waibel, Alex. Eu-bridge mt: Combined machine translation. In *ACL 2014 Ninth Workshop on Statistical Machine Translation*, 2014.

Graves, Alex. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850, 2013.

Ha, D., Dai, A., and Le, Q. V. HyperNetworks. *ArXiv e-prints*, September 2016.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Identity mappings in deep residual networks. *CoRR*, abs/1603.05027, 2016.

Hochreiter, Sepp and Schmidhuber, Jürgen. Long short-term memory. *Neural computation*, 1997.

Hochreiter, Sepp, Bengio, Yoshua, and Frasconi, Paolo. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In Kolen, J. and Kremer, S. (eds.), *Field Guide to Dynamical Recurrent Networks*. IEEE Press, 2001.

Hutter, Marcus. The human knowledge compression contest. <http://prize.hutter1.net/>, 2012.

Kaiser, Łukasz and Bengio, Samy. Can active memory replace attention? *Advances in Neural Information Processing Systems*, 2016.

Kalchbrenner, Nal and Blunsom, Phil. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 2013.

Kalchbrenner, Nal, Danihelka, Ivo, and Graves, Alex. Grid long short-term memory. *International Conference on Learning Representations*, 2016a.

- Kalchbrenner, Nal, van den Oord, Aaron, Simonyan, Karen, Danihelka, Ivo, Vinyals, Oriol, Graves, Alex, and Kavukcuoglu, Koray. Video pixel networks. *CoRR*, abs/1610.00527, 2016b.
- Kingma, Diederik P. and Ba, Jimmy. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.
- Luong, Minh-Thang and Manning, Christopher D. Achieving open vocabulary neural machine translation with hybrid word-character models. In *ACL*, 2016.
- Luong, Minh-Thang, Pham, Hieu, and Manning, Christopher D. Effective approaches to attention-based neural machine translation. In *EMNLP*, September 2015.
- Mikolov, Tomas, Karafiát, Martin, Burget, Lukás, Cernocký, Jan, and Khudanpur, Sanjeev. Recurrent neural network based language model. In *INTERSPEECH 2010*, pp. 1045–1048, 2010.
- Rocki, Kamil. Recurrent memory array structures. *CoRR*, abs/1607.03085, 2016.
- Simonyan, Karen, Vedaldi, Andrea, and Zisserman, Andrew. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, abs/1312.6034, 2013.
- Srivastava, Rupesh Kumar, Greff, Klaus, and Schmidhuber, Jürgen. Highway networks. *CoRR*, abs/1505.00387, 2015.
- Sutskever, Ilya, Vinyals, Oriol, and Le, Quoc V. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pp. 3104–3112, 2014.
- van den Oord, Aaron, Dieleman, Sander, Zen, Heiga, Simonyan, Karen, Vinyals, Oriol, Graves, Alex, Kalchbrenner, Nal, Senior, Andrew, and Kavukcuoglu, Koray. Wavenet: A generative model for raw audio. *CoRR*, abs/1609.03499, 2016a.
- van den Oord, Aäron, Kalchbrenner, Nal, and Kavukcuoglu, Koray. Pixel recurrent neural networks. In *ICML*, volume 48, pp. 1747–1756, 2016b.
- Williams, Philip, Sennrich, Rico, Nadejde, Maria, Huck, Matthias, and Koehn, Philipp. Edinburgh’s syntax-based systems at WMT 2015. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, 2015.
- Wu, Yonghui, Schuster, Mike, Chen, Zhifeng, Le, Quoc V., Norouzi, Mohammad, Macherey, Wolfgang, Krikun, Maxim, Cao, Yuan, Gao, Qin, Macherey, Klaus, Klingner, Jeff, Shah, Apurva, Johnson, Melvin, Liu, Xiaobing, ukasz Kaiser, Gouws, Stephan, Kato, Yoshikiyo, Kudo, Taku, Kazawa, Hideto, Stevens, Keith, Kurian, George, Patil, Nishant, Wang, Wei, Young, Cliff, Smith, Jason, Riesa, Jason, Rudnick, Alex, Vinyals, Oriol, Corrado, Greg, Hughes, Macduff, and Dean, Jeffrey. Google’s neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016a.
- Wu, Yuhuai, Zhang, Saizheng, Zhang, Ying, Bengio, Yoshua, and Salakhutdinov, Ruslan. On multiplicative integration with recurrent neural networks. *CoRR*, abs/1606.06630, 2016b.
- Yu, Fisher and Koltun, Vladlen. Multi-scale context aggregation by dilated convolutions. *CoRR*, abs/1511.07122, 2015.
- Zhou, Jie, Cao, Ying, Wang, Xuguang, Li, Peng, and Xu, Wei. Deep recurrent models with fast-forward connections for neural machine translation. *CoRR*, abs/1606.04199, 2016.