Neural Machine Translation in Linear Time

Neural Machine Translation in Linear Time

- 저자: Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, Koray Kavukcuoglu (Google Deepmind, London UK)
- 딥마인드 (말 ~~다했죠~~)

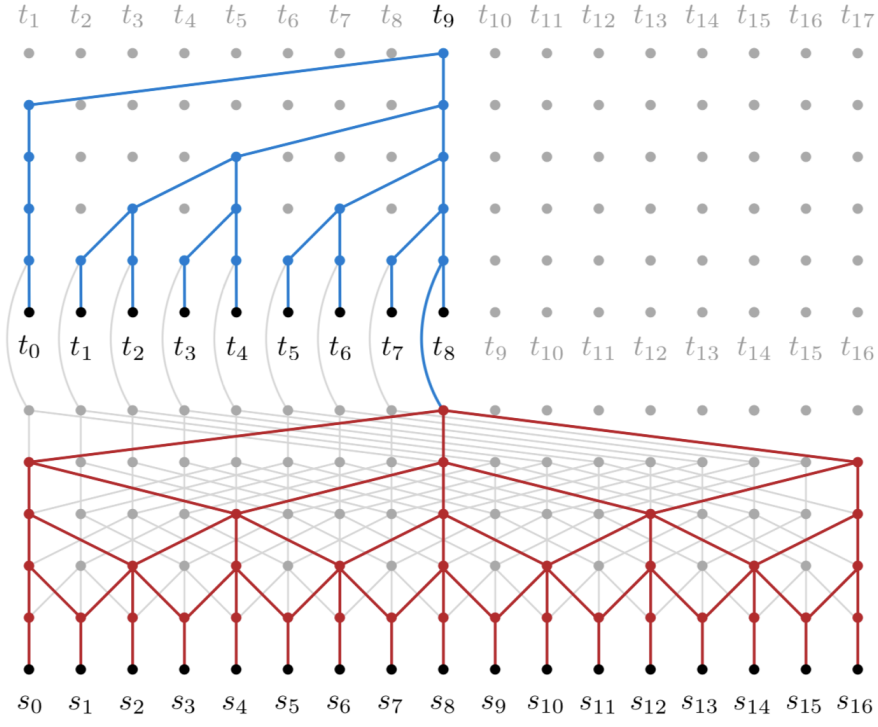


Figure 1. The architecture of the ByteNet. The target decoder (blue) is stacked on top of the source encoder (red). The decoder generates the variable-length target sequence using dynamic unfolding.

Who is an Author?

Nal Kalchbrenner

팔로우

내 프로필 만들기

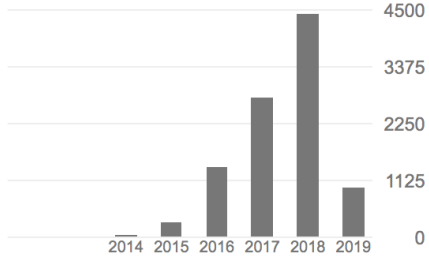
Google Brain Amsterdam
google.com의 이메일 확인됨 - 홈페이지

Deep Learning Generative Models Machine Translation Audio Synthesis

제목	인용	연도
Mastering the game of Go with deep neural networks and tree search D Silver, A Huang, CJ Maddison, A Guez, L Sifre, G Van Den Driessche, ... nature 529 (7587), 484	4501	2016
A convolutional neural network for modelling sentences N Kalchbrenner, E Grefenstette, P Blunsom arXiv preprint arXiv:1404.2188	1696	2014
WaveNet: A generative model for raw audio. A Van Den Oord, S Dieleman, H Zen, K Simonyan, O Vinyals, A Graves, ... SSW 125	737	2016
Recurrent continuous translation models N Kalchbrenner, P Blunsom Proceedings of the 2013 Conference on Empirical Methods in Natural Language …	654	2013
Pixel recurrent neural networks A Oord, N Kalchbrenner, K Kavukcuoglu arXiv preprint arXiv:1601.06759	594	2016
Conditional image generation with pixelcnn decoders A Van den Oord, N Kalchbrenner, L Espeholt, O Vinyals, A Graves Advances in neural information processing systems, 4790-4798	490	2016
Full resolution image compression with recurrent neural networks G Toderici, D Vincent, N Johnston, S Jin Hwang, D Minnen, J Shor, ... Proceedings of the IEEE Conference on Computer Vision and Pattern …	257 *	2017
Grid long short-term memory N Kalchbrenner, I Danihelka, A Graves arXiv preprint arXiv:1507.01526	200	2015
Neural machine translation in linear time N Kalchbrenner, L Espeholt, K Simonyan, A Oord, A Graves, ... arXiv preprint arXiv:1610.10099	168	2016

인용

	전체	2014년 이후
서지정보	10052	10018
h-index	19	19
i10-index	20	20



공동 저자

모두 보기

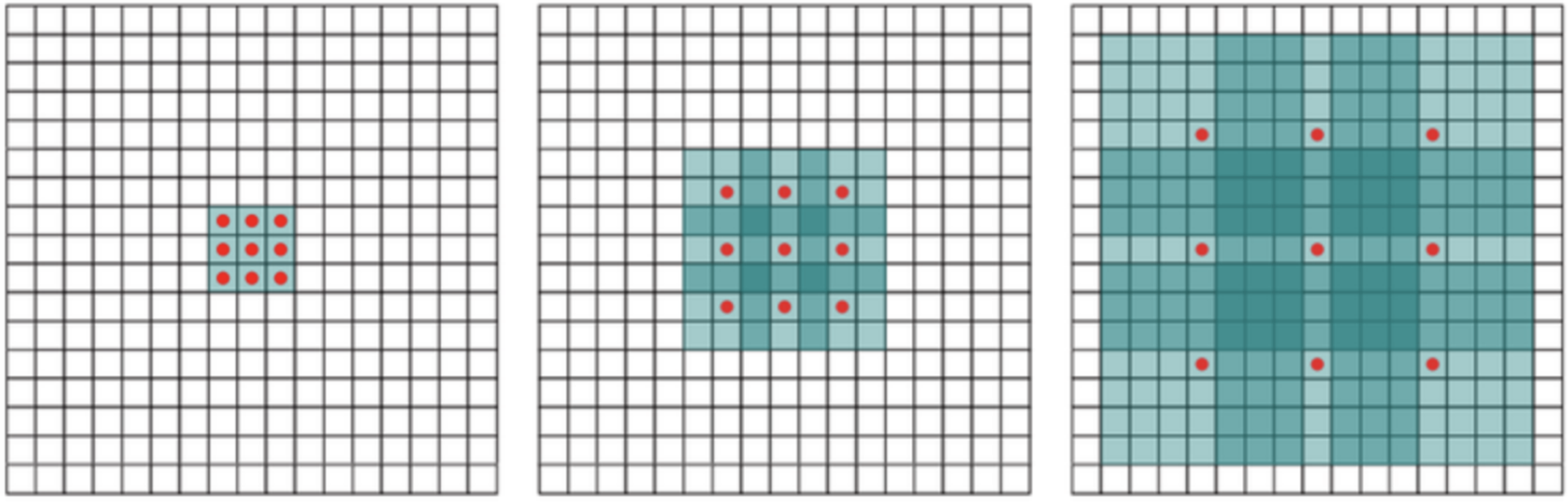
- koray kavukcuoglu — DeepMind
- Aäron van den Oord — Google DeepMind
- Sander Dieleman — Research Scientist, DeepMind
- Phil Blunsom — DeepMind and Oxford University
- Alex Graves — University of Toronto
- Karen Simonyan — Google DeepMind

Abstract

- 이 당시엔 `ByteNet` 은 Novel architecture였다
  (2016년, 후에 `Attention is all you need` 논문에서도 인용함)
- `ByteNet` 이라고 부름
- one-dimensional CNN이고 Encoder-Decoder구조인데 Decoder를 Encoder위에 Stacking한 구조임
  - sequence의 temporal resolution 보존
    (Resolution Preserving; RP)
  - source와 target의 lengths가 다른 걸 잘 처리함
- diliation방식 (스케일을 넓힐 때 주로 사용함 자세한 내용은 블로그 참고)의 CNN을 encoder(not masked), decoder(maksed)에서 사용함

Abstract

- diliation방식 (스케일을 넓힐 때 주로 사용함 자세한 내용은 블로그 참고)의 CNN을 decoder 에서 사용함



Abstract

- Contribution
  - sequence length에 linear한 runtime을 자랑함
    (단점은 메모리가 좀 더 많이 필요함, ~~layer를 더 쌓아서 그런건가~~)
  - character-level LM에서 SOTA기록함
    (이전 SOTA였던 RNN 프레임워크 성능 깸)
  - character-to-character MT (English-to-German WMT translation task)에서도 SOTA기록함 (attenton계열의 quadratic runtime 을 갖는 RNN 계열 프레임워크의 성능도 깸)
  - 우리가 만든 representation이 토큰간의 latent alignment structure 정보를 갖고 있음을 알아냄 (히트맵 참고)

1. Introduction

- NMT에서 NeuralNet은 주어진 source lang의 시퀀스의 분포로부터 target lang의 시퀀스 분포를 estimate함
- Network는 크게 Encoder, Decoder 구조로 구성됨 (Kalchbrenner & Blunsom, 2013)
- RNN은 시퀀스 모델링에 파워풀하고 (Hochreiter & Schmidhuber, 1997), language modeling에 많이 사용되지만 (Mikolov et al., 2010) 잠재적 단점들이 있음
  - RNN 구조상 parallel하게 실행이 안됨
  - Forward & Backword 할때도 path의 full distance에 대한 조건이 요구됨 (전체 path를 다 봐야함)
  - larger distance? -> dependency 배우기 어려워짐 (Hochreiter et al., 2001)
- 몇가지 네트워크가 계속 제안되었음
  - encoder-decoder networks
  - networks with attention pooling
  - two dimensional networks
  - 이런 모델들은 성능이 좋지만 running time이 sequence length에 `super-linear` 하고 (실제로 이렇게 적혀있음, linear보다 기울기가 높은 함수를 말함) source sequence를 contant size의 representation으로 바꾸거나 무거운 memorization step을 사용한다든지 하는 단점이 있음 (sequence 길이가 증가할수록 심해짐)

1. Introduction

- 제안 모델은 encoder-decoder 구조를 사용하면서 2가지 방법으로 위에서 말한 단점을 극복하고자함
    - 첫번째는 encoder representation 위에 decoder를 쌓는 것임. 이렇게 하면 시퀀스의 temporal resolution이 보존됨(dependency 고려가 쉬워짐)
        - fixex-size에 source sequence의 representation 저장하는 한계를 어느정도 극복할 수 있게됨 (Kalchbrenner & Blunsom, 2013; Sutskever et al., 2014)
    - 두번째방법은 `dynamic unfolding` mechanism을 사용하는 것임
        - source와 target의 길이가 다른 걸 간단하고 효율적으로 대응하게 해줌
- ByteNet(제안모델)에서 사용하는 CNN은 fixed-dpeth 형태의 one-dimensional CNN임
    - The two CNNs (Encoder&Decoder) use `increasing factors of dilation to grow the receptive fields`
    - Decoder에서 사용된 CNN은 target sequence에서의 future tokens을 보는걸 막기 위해 `masking` 처리가 되어있음(~~transformer 아이디어와 비슷하네,~~ 본래의 이 아이디어는 `Pixel recurrent neural networks` 논문에서 먼저 사용됨, 본 논문의 저자가 2저자로 참여) (van den Oord et al., 2016b).

1. Introduction

- 이 네트워크는 learning과 computational 관점에서 이점이 있음
  - computational 관점에서는 source & target sequence 길이에 linear한 runtime을 가짐
    - encoding 부분과 decoding부분을 training시에 `parallel` 하게 돌릴 수 있음 (Sec2 참고)
  - learning 관점에서는 source sequence의 representation이 `resolution preserving` 됨
    - 단, 메모리는 더 필요함
    - 덕분에 encoder-decoder 사이의 bandwidth도 최대가 됨

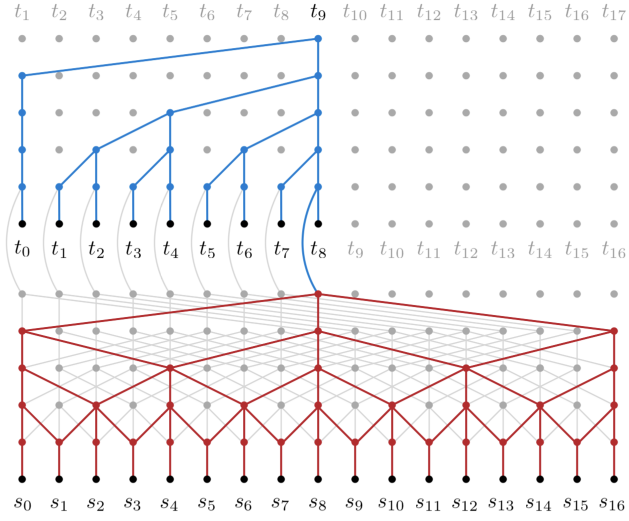


Figure 1. The architecture of the ByteNet. The target decoder (blue) is stacked on top of the source encoder (red). The decoder generates the variable-length target sequence using dynamic unfolding.

1. Introduction

- 본 논문에서는 ByteNet을 `character-level langauge modeling` 과 `character-to-character machine translation` 에 적용함
  - `Hutter Prize Wikipedia task` 로 decoder network에 대해 평가함
    (이 당시 SOTA는 `1.31` bits/character임)
  - `English-to-German WMT benchmark` 로 평가함
    (이 당시 SOTA는 BLEU score: `22.82` (0.38 bits/character), `25.53` (0.389 bits/character) on 2014, 2015 test sets)
    - 그 당시 SOTA인 GNMT 모델(char-level) 보다 더 나은 결과를 보여줌

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.

Neural Machine Translation in Linear Time

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.

2. Neural Translation Model

- NMT model은 다음의 분포를 estimate함

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

- s: source toekns, t: target tokens
- 토큰은 단어가 될수도, 캐릭터가 될 수도 있음
- source network(encoder)와 targer network(decoder)로 source representation을 target string이 되게 만들 것임
- 문장을 만들어내는 거니까 당연히 decoder가 language model처럼 동작한다고 볼 수 있음

2. Neural Translation Model

- NMT의 basic properties
  - autoregressive: 자기 자신의 값을 사용함 (decoder)
  - source와 target token의 order에 민감함
  - `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`
- 보통의 NMT가 이러한 basic properties를 갖지만 본 논문에서의 모델은 이것 이외의 properties를 갖게 하고자함
  - linear run time (parallel computing)
  - size of source representation (fixed -> `in the length of the source string`); `resolution preserving`, not constant size
  - 더 짧은 signal path (network 내의 forward, backward signal);
    아마 층이 쌓일 수록 멀리 있는 것까지 고려해주는 계산 방식(dilation)이 있어서 그런듯

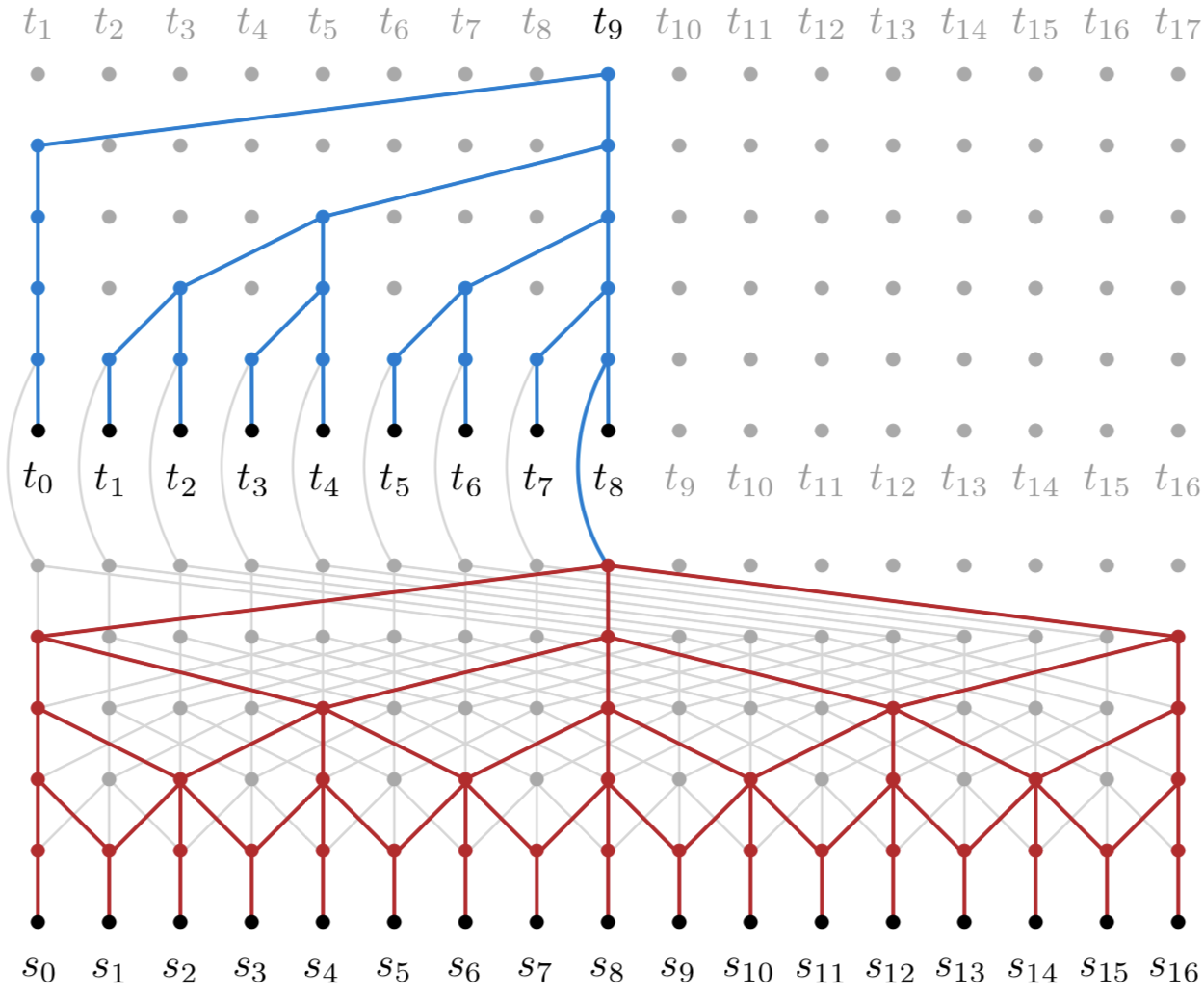


Figure 1. The architecture of the ByteNet. The target decoder (blue) is stacked on top of the source encoder (red). The decoder generates the variable-length target sequence using dynamic unfolding.

3. ByteNet

- [encoder] + [decoder stacked on an encoder]
- Generate `variable-length outputs` via `dynamic unfolding`
- masking은 decoder 부분에만 적용함
- decoder 부분엔 Residual block이 적용됨

3. ByteNet

3.1. Encoder-Decoder Stacking

- 제안하는 논문의 특징 중 하나가 바로 Encoder와 Decoder를 연결하는 방식이 새롭다는 것 (이 당시 기준)
- To maximize `the representational bandwidth` between the encoder and the decoder
- fixed-size vector 또는 attentional pooling (Bahdanau et al., 2014 including 조경현교수님)과 같은 방법과는 대조된다고 말함
    - 그렇지만 decoder가 encoder의 state까지 참조하는건 아님, encoder output의 representation까지만 참조함
    - 단순히 참조하는 state가 한개가 아니라 여러개로 늘어났다는게 의의가 있을듯..

3. ByteNet

3.1. Encoder-Decoder Stacking

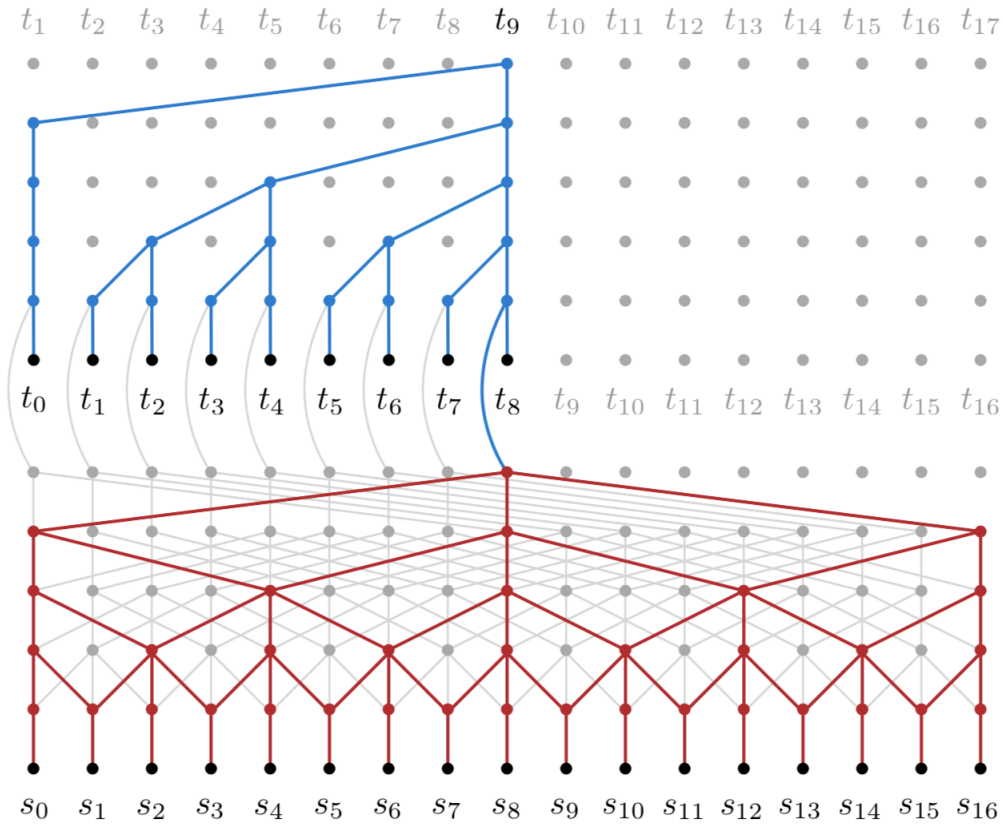


Figure 1. The architecture of the ByteNet. The target decoder (blue) is stacked on top of the source encoder (red). The decoder generates the variable-length target sequence using dynamic unfolding.

3. ByteNet

3.2. Dynamic Unfolding

- Encoder와 Decoder가 처리하는 sequence의 length가 다르다면 둘은 directly connected 될 수 없음
- 이를 해결하기 위해 Dynamic Unfolding이라는 메커니즘을 제안함

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

3. ByteNet

3.2. Dynamic Unfolding

- 원리는 간단함. `encoder의 output의 representaiton`이 적당한 길이 `|t^|`(`|s|`에 linear relationship)를 갖도록 생성함; source sequence length `|s|`와 target sequence length `|t|`로 ~~적절하게(?)~~ 생성
- 대게 `|t|` 보다는 `|t^|`가 길게 나옴 (a=1.20, b=0으로 논문에서 설정함)

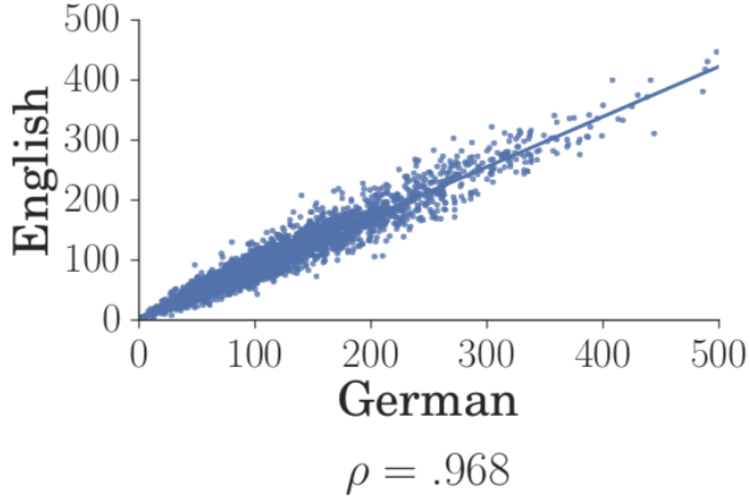


Figure 5. Lengths of sentences in characters and their correlation coefficient for the English-to-German WMT NewsTest-2013 validation data. The correlation coefficient is similarly high ($\rho > 0.96$) for all other language pairs that we inspected.

3. ByteNet

3.2. Dynamic Unfolding

- 이 적절한 길이의 `encoder output representation` 으로 decoder에서 상황에 맞게 (decoding되는 길이에 맞게) 가져다 쓰며 decoding함

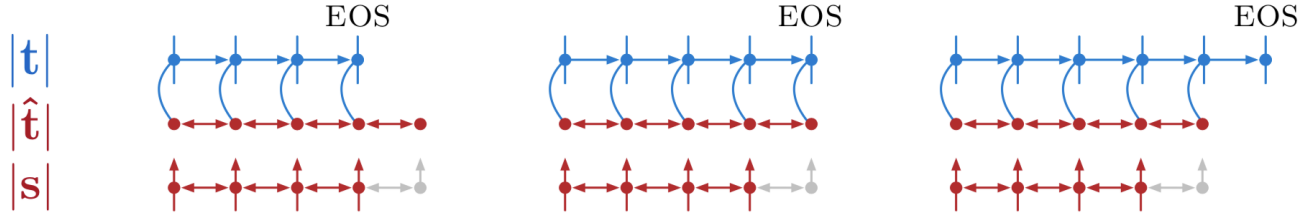


Figure 2. Dynamic unfolding in the ByteNet architecture. At each step the decoder is conditioned on the source representation produced by the encoder for that step, or simply on no representation for steps beyond the extended length $|\hat{t}|$. The decoding ends when the target network produces an end-of-sequence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

- EOS가 미리 나오면 그 time step까지만 쓰고, `|t^|` 보다 길게 나오면 encoder output representation 없이 EOS 나올때 까지 decoding 진행함
- source에 padding을 추가해서 생성함 (논문에 자세히는 안나옴)

3. ByteNet

3.3 Input Embedding Tensor

- target sequence에서 처음 n개만 우선 input으로 임베딩시킴 `(0~n-1)`
- `(1~n)` 번까지 prediction을 위한 target으로 셋팅
- input으로 임베딩된 텐서 + prediction으로 사용되는 tensor가 concat되면서 n x 2*d* 차원을 가짐 (*d* == number of inner channels in the network)
- 아래는 Recurrent ByteNet 경우에 대한 예시임(이해를 돕기 위해가져옴)

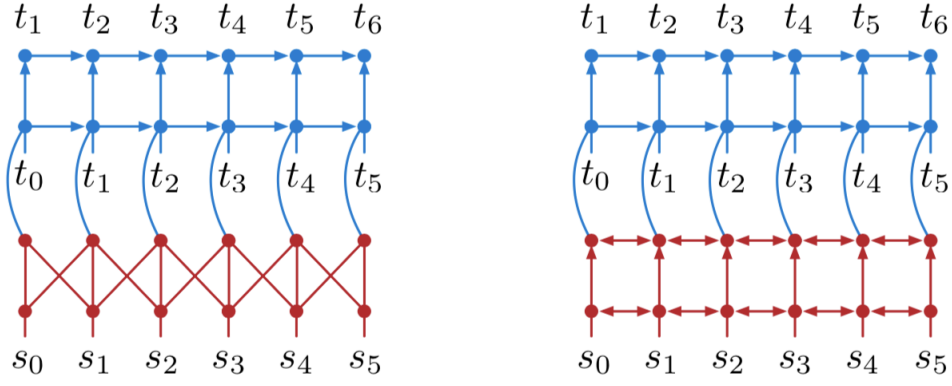


Figure 4. Recurrent ByteNet variants of the ByteNet architecture. Left: Recurrent ByteNet with convolutional source network and recurrent target network. Right: Recurrent ByteNet with bidirectional recurrent source network and recurrent target network. The latter architecture is a strict generalization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

3. ByteNet

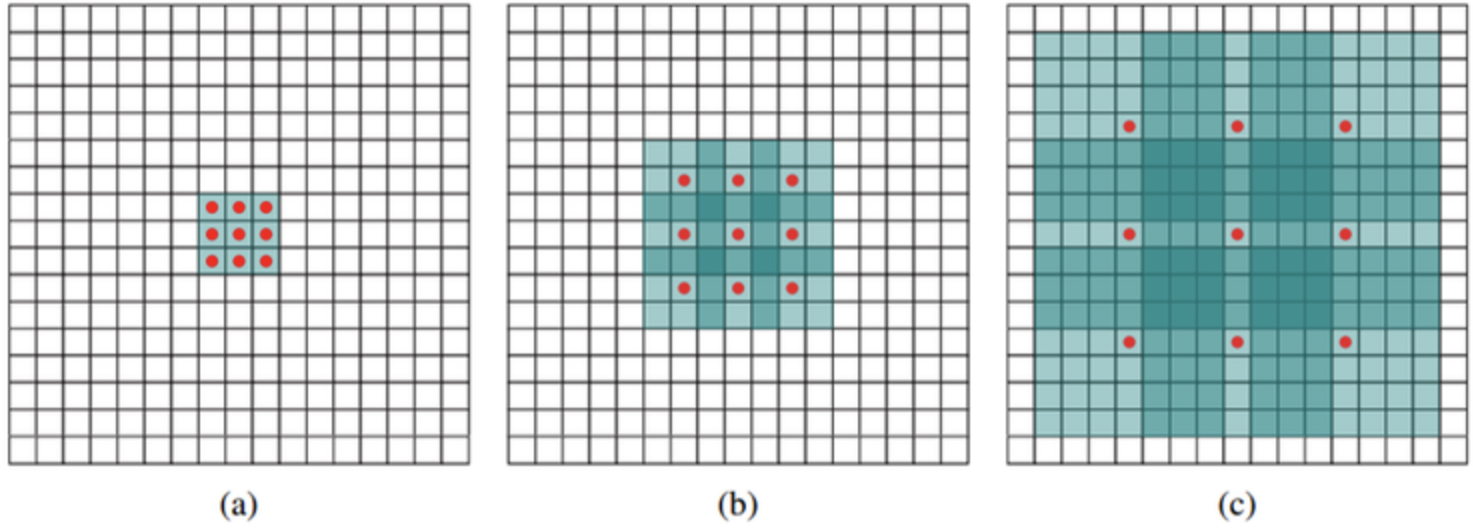
3.4 Masked One-dimensional Convolutions

- decoder에서는 `Pixel Recurrent Neural Networks (DeepMind, 2016)`에서 사용한 `masked one-dimensional convolution`을 `maksed kernel size *k*`로 `input embedding tensor`에 대해서 적용함
- Masking은 future tokens에 대해서 적용함 (prediction시에 current token이 영향받지 않도록!, Loss 계산을 생각해보면 좋을듯)
- kernel weight를 zero로하거나 input map에 padding처리해서 구현함

3. ByteNet

3.5 Dilation

- masked convolution에 적용됨
- target network (decoder)의 receptvie field 크기 넓히려고 사용함
- Network depth 커질수록, Dilation을 통한 receptive field도 exponentially 커짐 (layer depth에 double로 증가, dilation rates r = 16으로 사용함, 1부터 시작해서 16까지 커짐)
- dilation방식 (스케일을 넓힐 때 주로 사용함 자세한 내용은 블로그 참고)



3. ByteNet

3.6 Residual Blocks

- 각 레이어는 residual block으로 래핑됨
- residual block은 additional convolutional layer가 있고 filter size는 1 x 1 임
- 본 논문에서는 2가지 버전의 residual blocks을 사용함
  - 하나는 ReLU(NMT에서 많이 씀)
  - 나머지 하나는 Multiplicative Units(Language Modeling에서 많이씀)을 활성함수로 씀
    - Multiplicative Units(MU)는 Video pixel Networks (Kalchbrenner et al., 2016b; 본인이 썼던 논문) 에서 가져온 개념임
- 두 버전 모두 다 활성함수 전에 Layer-Norm 사용함

3. ByteNet

3.6 Residual Blocks

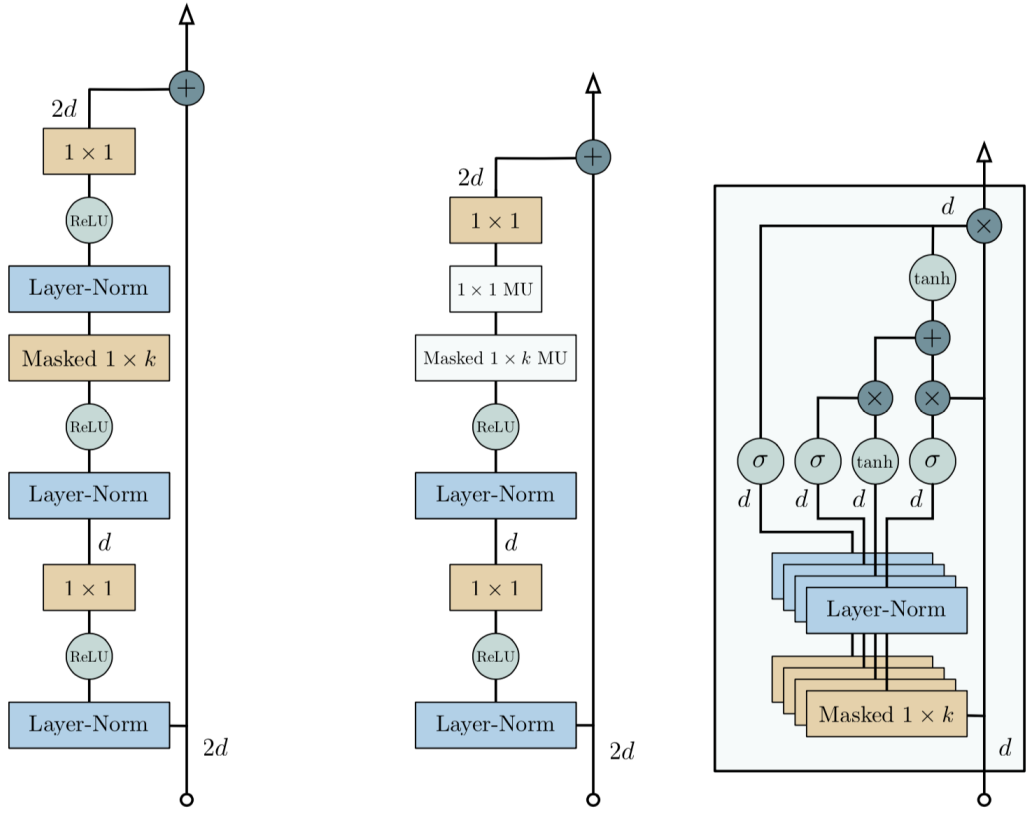


Figure 3. Left: Residual block with ReLUs (He et al., 2016) adapted for decoders. Right: Residual Multiplicative Block adapted for decoders and corresponding expansion of the MU (Kalchbrenner et al., 2016b).

3. ByteNet

3.6 Residual Blocks

```
def layer_normalization(x, name, epsilon=1e-8, trainable = True):
    with tf.variable_scope(name):
        shape = x.get_shape()
        beta = tf.get_variable('beta', [ int(shape[-1])],
            initializer=tf.constant_initializer(0), trainable=trainable)
        gamma = tf.get_variable('gamma', [ int(shape[-1])],
            initializer=tf.constant_initializer(1), trainable=trainable)

        mean, variance = tf.nn.moments(x, axes=[len(shape) - 1], keep_dims=True)

        x = (x - mean) /  tf.sqrt(variance + epsilon)

        return gamma * x + beta

def byetenet_residual_block(input_, dilation, layer_no,
    residual_channels, filter_width,
    causal = True, train = True):
        block_type = "decoder" if causal else "encoder"
        block_name = "bytenet_{}_layer_{}_{}".format(block_type, layer_no, dilation)
        with tf.variable_scope(block_name):
            input_ln = layer_normalization(input_, name="ln1", trainable = train)
            relu1 = tf.nn.relu(input_ln)
            conv1 = conv1d(relu1, residual_channels, name = "conv1d_1")
            conv1 = layer_normalization(conv1, name="ln2", trainable = train)
            relu2 = tf.nn.relu(conv1)

            dilated_conv = conv1d(relu2, residual_channels,
                dilation, filter_width,
                causal = causal,
                name = "dilated_conv"
                )
            print dilated_conv
            dilated_conv = layer_normalization(dilated_conv, name="ln3", trainable = train)
            relu3 = tf.nn.relu(dilated_conv)
            conv2 = conv1d(relu3, 2 * residual_channels, name = 'conv1d_2')
            return input_ + conv2
```

- Reference: https://github.com/paarthneekhara/byteNet-tensorflow/blob/master/ByteNet/ops.py

4. Model Comparison

4.1 Recurrent ByteNets

- 비교를 위해 Recurrent 버전까지 추가하겠음
- [1] Decoder를 RNN으로 바꿔보자
- [2] Enocder, Decoder 모두 RNN으로 바꾸되 stacked decoder 유지하자 (RNN Enc-Dec 구조와 가장 유사함)

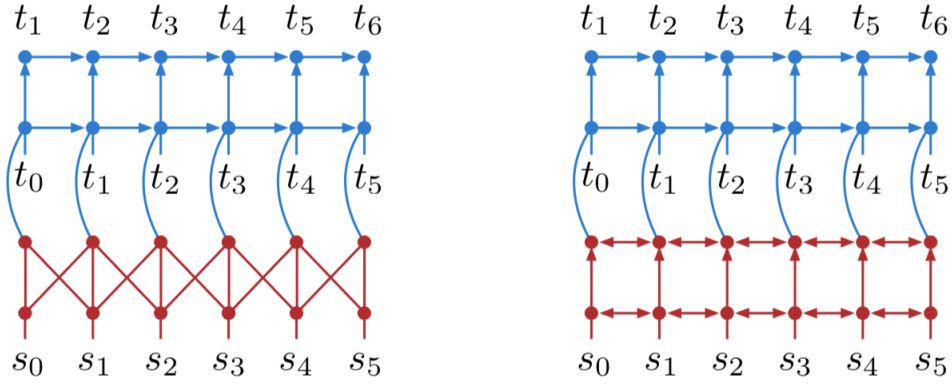


Figure 4. Recurrent ByteNet variants of the ByteNet architecture. Left: Recurrent ByteNet with convolutional source network and recurrent target network. Right: Recurrent ByteNet with bidirectional recurrent source network and recurrent target network. The latter architecture is a strict generalization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

4.2 Comparison of Properties

- Runtime 비교
- 참고용어
  - RCTM: Recurrent Continuous translation
  - RP: Resol4ution Preserving
  - Path_S: length from source token to any output target token
  - Path_T: length from input target token to any output target token
- ByteNet이 linear runtime도 보존 되고, RP도 만족시킴

Model	$\mathbf{Net_S}$	$\mathbf{Net_T}$	**Time**	**RP**	$\mathbf{Path_S}$	$\mathbf{Path_T}$
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.

5. Character Prediction

- Character-level language modelling benchmark에 대해서 평가함
- `Hutter Prize version of the Wikipedia dataset` 사용
  - 90 million bytes: training
  - 5 million bytes: validation
  - 5 million bytes: testing
- ByteNet Decoder's hyper params
  - 30 residual Blocks: 6 sets * 5 blocks
    - 5 blocks's dillation rates: 1, 2, 4, 8 and 16
  - masked kernel size: 3
  - 315 characters를 커버 할 수 있는 receptive field가 됨
  - hidden units *d*: 512
  - 여기선 Residual Multiplicative blocks 사용
  - Optimizer: Adam
    - lr: 0.0003
    - weight decay: 0.001
  - dropout to the last ReLU layer before the softmax
    - drop_rate: 0.1
  - a batch of sequences of character: 500
    - 100: for minimum context
    - 400: for prediction

5. Character Prediction

- Table 3의 결과는 모두 LSTM 기반임
- ByteNet이 성능이 제일 좋음 (현재는 24-layer transformer-XL이 SOTA)
- 링크: http://nlpprogress.com/english/language_modeling.html

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.

6. Character-Level Machine Translation

- WMT English to German translation Task에 대해서 평가함
- Validation: `NewsTest 2013`
- Testing: `NewsTest 2014`, `NewsTest 2015`
- German character vocab size: 323
- English character vocab size: 296

6. Character-Level Machine Translation

- ByteNet's hyper params & training process
  - 30 residual Blocks: 6 sets * 5 blocks
    - 5 blocks's dillation rates: 1, 2, 4, 8 and 16
  - 여기선 Residual blocks with ReLUs 사용
  - hidden units *d*: 800
  - kernel size in the source network: 3
  - maksed kernel size in the target network: 3
  - Optimizer: Adam
    - lr: 0.0003
  - 각 문장은 special characters로 패딩적용됨; 패딩의 20% 정도는 dynamic unfolding을 적용하기 위해 source sentence에 적용함
  - 학습시 효율적인 배치처리를 위해 패딩된 길이에 맞게 bucketting 사용함
  - Beam search 사용: beam of size 12

6. Character-Level Machine Translation

- Table 2와 Table 4에 따르면 ByteNet이 Character-level과 subword-level NMT와 비교할 때 성능이 제일 좋음 (~~그러나 word-pieces를 사용하는 GNMT보다는 낮은데 word-pieces는 subword라고 생각안하고 word-level이라고 생각하는듯~~)

Neural Machine Translation in Linear Time

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.

	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.

6. Character-Level Machine Translation

- Table 5는 ByteNet의 English-German Translation 결과를 보여줌
- recodering, trasliteration(단어 그대로 갖다 쓰는거)이 일어나는 특징이 있음

Neural Machine Translation in Linear Time

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 gegenber 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.

6. Character-Level Machine Translation

- Figure 6는 gradient를 heatmap으로 시각화한건데, 단어는 단어를 구성하는 characters의 gradient를 합치고, 각 컬럼에 대해서 normalization한 것임
- 두가지 dependency를 표현함
    - source와 output의 dependency
    - target과 previous target input의 dependency도 보여줌

6. Character-Level Machine Translation

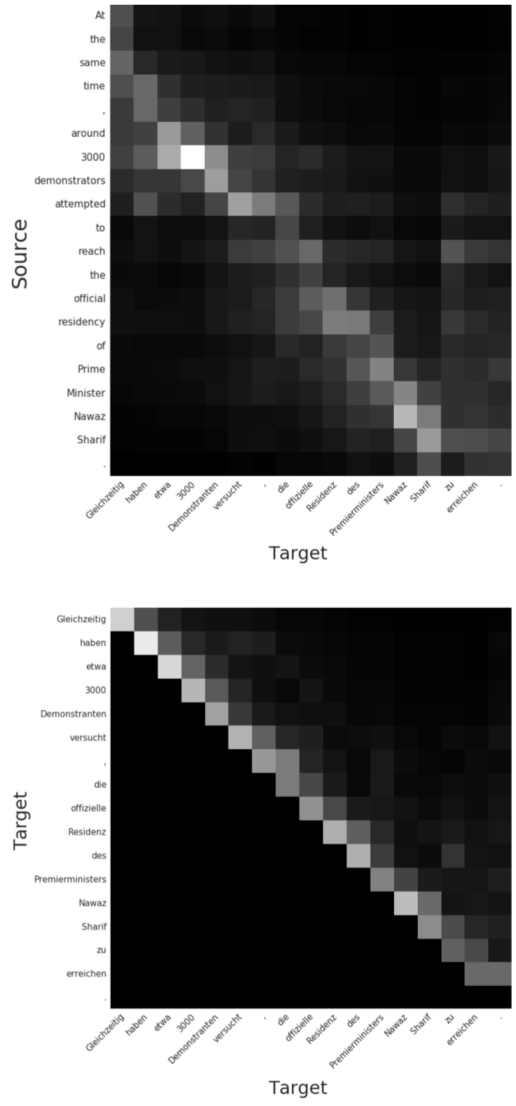


Figure 6. Magnitude of gradients of the predicted outputs with respect to the source and target inputs. The gradients are summed for all the characters in a given word. In the bottom heatmap the magnitudes are nonzero on the diagonal, since the prediction of a target character depends highly on the preceding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

7. Conclusion

- linear running time 갖는 NMT 제안함
- RP 및 signal progation path 길이 줄이는 장점도 있음
- character-level language model에서 SOTA 찍고 RNN 이김
- character-to-character machine translation에서 SOTA 찍음
    - 기대하던대로 tokens간의 alignment도 잘 나옴

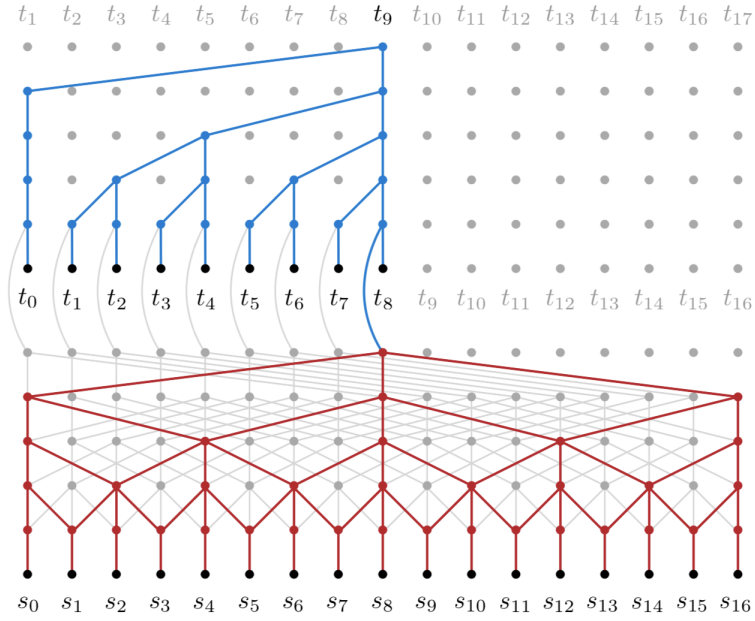


Figure 1. The architecture of the ByteNet. The target decoder (blue) is stacked on top of the source encoder (red). The decoder generates the variable-length target sequence using dynamic unfolding.