Sequence to Sequence Learning: Fast Training and Inference with Gated Convolutions

Michael Auli
with Jonas Gehring, David Grangier, Yann Dauphin, Angela Fan, Sergey Edunov, Marc’Aurelio Ranzato, Myle Ott

http://github.com/facebookresearch/fairseq-py
Sequences pose a challenge for DNNs because they require that the dimensionality of the input and output is known and fixed. In this paper, we show that a straightforward application of the Long Short-Term Memory (LSTM) architecture [16] can solve general sequence to sequence problems. The idea is to use one LSTM to read the input sequence, one time step at a time, to obtain large fixed-dimensional vector representation, and then to use another LSTM to extract the output sequence from that vector (fig. 1). The second LSTM is essentially a recurrent neural network language model [28, 23, 30] except that it is conditioned on the input sequence. The LSTM’s ability to successfully learn on data with long range temporal dependencies makes it an automatic choice for this application due to the considerable time lag between the inputs and their corresponding outputs (fig. 1).

There have been a number of related attempts to address the general sequence to sequence learning problem with neural networks. Our approach is closely related to Kalchbrenner and Blunsom [18] who were the first to map the entire input sentence to vector, and is very similar to Cho et al. [5]. Graves [10] introduced a novel differentiable attention mechanism that allows neural networks to focus on different parts of their input, and an elegant variant of this idea was successfully applied to machine translation by Bahdanau et al. [2]. The Connectorist Sequence Classification is another popular technique for mapping sequences to sequences with neural networks, although it assumes a monotonic alignment between the inputs and the outputs [11].

Figure 1: Our model reads an input sentence “ABC” and produces “WXYZ” as the output sentence. The model stops making predictions after outputting the end-of-sentence token. Note that the LSTM reads the input sentence in reverse, because doing so introduces many short term dependencies in the data that make the optimization problem much easier.

- **Encode** source sequence, and **decode** target sequence with **RNNs** (Sutskever et al., 2014)
- **Attention**: choose relevant encoder states (Bahdanau et al., 2014)

Figure from: Sutskever et al., 2014, “Sequence to Sequence Learning with Neural Networks”
Sequence to Sequence Learning

- Applications: translation, summarization, parsing, dialogue, ...
- Translation, e.g., “La maison de Léa.” -> “Léa’s house.”
- “Models basis for 25% of posters at ACL”, Lapata at keynote ACL’17
Sequence to Sequence Learning

Recurrent Continuous Translation Models

Nal Kalchbrenner  Phil Blunsom
Department of Computer Science
University of Oxford
{nal.kalchbrenner,phil.blunsom}@cs.ox.ac.uk

Joint Language and Translation Modeling with Recurrent Neural Networks

Michael Auli, Michel Galley, Chris Quirk, Geoffrey Zweig
Microsoft Research
Redmond, WA, USA
{michael.auli,mgalley,chrissq,gzweig}@microsoft.com

Published as a conference paper at ICLR 2015

Neural Machine Translation by Jointly Learning to Align and Translate

Dmitry Bahdanau
Jacobs University Bremen, Germany

Kyunghyun Cho  Yoshua Bengio
Université de Montréal

Sequence to Sequence Learning with Neural Networks

Ilya Sutskever  Oriol Vinyals  Quoc V. Le
Google
ilyas@google.com  vinyals@google.com  qvl@google.com
Sequence to Sequence Learning

Deep Recurrent Models with Fast-Forward Connections for Neural Machine Translation

Jie Zhou, Ying Cao, Xuguang Wang, Peng Li, Wei Xu
Baidu Research - Institute of Deep Learning
Baidu Inc., Beijing, China
{zhoujie01, caoying03, wangxuguang, lipeng17, wei.xu}@baidu.com

Convolutional Sequence to Sequence Learning

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, Yann N. Dauphin

Attention Is All You Need

Google’s Neural Machine Translation System: Bridging the Gap between Human and Machine Translation

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi
yonghui, schuster, zhifengc, qv, mnorouzi@google.com
Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, Jeffrey Dean

Press Information – DeepL Translator Launch
Overview

- Gated convolutions for Language Modeling
- Convolutional Sequence to Sequence Learning
- Analyzing beam search for seq2seq
Gated Convolutional Models for Language Modeling
Language Modeling

- Estimate probability of a sequence of words
  \[ P(w_0, \ldots, w_N) = P(w_0) \prod_{i=1}^{N} P(w_i|w_0, \ldots, w_{i-1}) \]

- Good language models help in speech (Mikolov et al, 2010) and translation

- LSTMs achieve state-of-the-art performance by processing sentences left to right
CNNs & RNNs

Vision ➔ Convolutional neural networks

NLP/Speech ➔ Recurrent neural networks
CNNs & RNNs

Vision → Convolutional neural networks

NLP/Speech → Recurrent neural networks
- Architectures complex: bi-directional, reverse processing
- Fail to model long-range dependencies in language: need attention
CNNs & RNNs

Vision → Convolutional neural networks

NLP/Speech → Recurrent neural networks
  • Architectures complex: bi-directional, reverse processing
  • Fail to model long-range dependencies in language: need attention

This talk: **model sequences well without RNNs**
What is a CNN?

- a linear projection taking several input vectors (embeddings, hidden states)
- that maps them to a single output vector (of same or different size)
- which is applied repeatedly to the input sequence at a given stride (=1 here) to yield an output sequence
CNNs for Sequence Modeling

- **Hierarchical**: bottom-up vs. left-right
- **Homogeneous**: all elements processed in same way
- **Efficient**: parallelizable over number of sequences & time dimension
CNNs for Sequence Modeling

- **In practice:**
  - dependencies are not arbitrarily long
  - e.g. Dauphin et al. ICML’17

- CNNs are much more efficient than LSTMs on GPU
  - e.g. Baidu DeepBench, github.com/baidu-research/DeepBench ’16
Recurrent Neural Network

The cat jumps far
Recurrent Neural Network

The cat jumps far
Recurrent Neural Network

The cat jumps far
Recurrent Neural Network

The cat jumps far.
Recurrent Neural Network

The cat jumps far.
Recurrent Neural Network

• O(T) sequential steps
• Recurrent connection causes vanishing gradient
• Are the recurrent connections necessary?

The cat jumps far.

Recurrent Neural Network

The cat jumps far.

The cat jumps far.
Multi-Layer Perceptron

The cat jumps far.
Multi-Layer Perceptron

- $O(1)$ sequential steps
- Proposed by (Bengio et al, 2001)
- Inefficient because no computation is shared between time steps
- Bad experimental results
Multi-Layer Perceptron

- O(1) sequential steps
- Proposed by (Bengio et al, 2001)
- Inefficient because no computation is shared between time steps
- Bad experimental results
Convolutional Neural Network

The cat jumps far.
Convolutional Neural Network

- O(1) sequential steps
- Incrementally build context of context windows

The cat jumps far
Convolutional Neural Network

- $O(1)$ sequential steps
- Incrementally build context of context windows

The cat jumps far.
Convolutional Neural Network

- O(1) sequential steps
- Incrementally build context of context windows
- Builds **hierarchical** structure
Gated Convolutional Neural Network

- Processes a sentence with a set of convolutions
- Each convolution learns higher level features
- Gates filter information to propagate up the hierarchy
Gated Convolutional Neural Network

- Processes a sentence with a set of convolutions
- Each convolution learns higher level features
- Gates filter information to propagate up the hierarchy
Gated Convolutional Neural Network

- Processes a sentence with a set of convolutions
- Each convolution learns higher level features
- Gates filter information to propagate up the hierarchy
Gated Convolutional Neural Network

- Processes a sentence with a set of convolutions
- Each convolution learns higher level features
- Gates filter information to propagate up the hierarchy
Gated Linear Unit

- The gated linear unit can be seen as a multiplicative skip connection
- We find this approach to gating improves performance
Gated Linear Unit

- The gated linear unit can be seen as a multiplicative skip connection
- We find this approach to gating improves performance
Gated Linear Unit

- The gated linear unit can be seen as a multiplicative skip connection
- We find this approach to gating improves performance
Training

- We use SGD with Nesterov’s momentum and weight normalization (Salimans & Kingma, 2016)
- Clipping for convnets (Pascanu et al. 2013)
- Adaptive Softmax (Grave et al, 2016) for very large vocabularies
Datasets

- **Google billion words** (Chelba et al, 2013):
  - ~800k vocabulary with ~800M tokens
  - independent sentences (~20 tokens)

- **WikiText-103** (Bradbury et al, 2016)
  - ~200k vocabulary with ~100M tokens
  - wikipedia articles (~4000 tokens)
Results: Google billion words

<table>
<thead>
<tr>
<th>Model</th>
<th>Test PPL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sigmoid-RNN-2048 (Ji et al., 2015)</td>
<td>68.3</td>
</tr>
<tr>
<td>Interpolated KN 5-Gram (Chelba et al., 2013)</td>
<td>67.6</td>
</tr>
<tr>
<td>Sparse Non-Negative Matrix LM (Shazeer et al., 2014)</td>
<td>52.9</td>
</tr>
<tr>
<td>RNN-1024 + MaxEnt 9 Gram Features (Chelba et al., 2013)</td>
<td>51.3</td>
</tr>
<tr>
<td>LSTM-2048-512 (Jozefowicz et al., 2016)</td>
<td>43.7</td>
</tr>
<tr>
<td>2-layer LSTM-8192-1024 (Jozefowicz et al., 2016)</td>
<td>30.6</td>
</tr>
<tr>
<td>LSTM-2048 (Grave et al., 2016a)</td>
<td>43.9</td>
</tr>
<tr>
<td>2-layer LSTM-2048 (Grave et al., 2016a)</td>
<td>39.8</td>
</tr>
<tr>
<td>GCNN-13</td>
<td>38.1</td>
</tr>
<tr>
<td>GCNN-14 Bottleneck</td>
<td>31.9</td>
</tr>
</tbody>
</table>

- GatedCNN manages to match the LSTM with comparable output approximation and computational budget for training.
Results: Google billion words

<table>
<thead>
<tr>
<th>Model</th>
<th>Test PPL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sigmoid-RNN-2048 (Ji et al., 2015)</td>
<td>68.3</td>
</tr>
<tr>
<td>Interpolated KN 5-Gram (Chelba et al., 2013)</td>
<td>67.6</td>
</tr>
<tr>
<td>Sparse Non-Negative Matrix LM (Shazeer et al., 2014)</td>
<td>52.9</td>
</tr>
<tr>
<td>RNN-1024 + MaxEnt 9 Gram Features (Chelba et al., 2013)</td>
<td>51.3</td>
</tr>
<tr>
<td>LSTM-2048-512 (Jozefowicz et al., 2016)</td>
<td>43.7</td>
</tr>
<tr>
<td>2-layer LSTM-8192-1024 (Jozefowicz et al., 2016)</td>
<td>30.6</td>
</tr>
<tr>
<td>LSTM-2048 (Grave et al., 2016a)</td>
<td>43.9</td>
</tr>
<tr>
<td>2-layer LSTM-2048 (Grave et al., 2016a)</td>
<td>39.8</td>
</tr>
<tr>
<td>GCNN-13</td>
<td>38.1</td>
</tr>
<tr>
<td>GCNN-14 Bottleneck</td>
<td>31.9</td>
</tr>
</tbody>
</table>

- GatedCNN manages to match the LSTM with comparable output approximation and computational budget for training
### Results: Google billion words

<table>
<thead>
<tr>
<th>Model</th>
<th>Test PPL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sigmoid-RNN-2048 \cite{Ji2015}</td>
<td>68.3</td>
</tr>
<tr>
<td>Interpolated KN 5-Gram \cite{Chelba2013}</td>
<td>67.6</td>
</tr>
<tr>
<td>Sparse Non-Negative Matrix LM \cite{Shazeer2014}</td>
<td>52.9</td>
</tr>
<tr>
<td>RNN-1024 + MaxEnt 9 Gram Features \cite{Chelba2013}</td>
<td>51.3</td>
</tr>
<tr>
<td>LSTM-2048-512 \cite{Jozefowicz2016}</td>
<td>43.7</td>
</tr>
<tr>
<td>2-layer LSTM-8192-1024 \cite{Jozefowicz2016}</td>
<td>30.6</td>
</tr>
<tr>
<td>BIG GLSTM-G4 \cite{Kuchaiiev2017}</td>
<td>23.3*</td>
</tr>
<tr>
<td>LSTM-2048 \cite{Grave2016a}</td>
<td>43.9</td>
</tr>
<tr>
<td>2-layer LSTM-2048 \cite{Grave2016a}</td>
<td>39.8</td>
</tr>
<tr>
<td>GCNN-13</td>
<td>38.1</td>
</tr>
<tr>
<td>GCNN-14 Bottleneck</td>
<td>31.9</td>
</tr>
</tbody>
</table>

- GatedCNN manages to match the LSTM with comparable output approximation and computational budget for training
## Results: Wikitext-103

- SOTA accuracy despite limited context size (25 & 32 words)

<table>
<thead>
<tr>
<th>Model</th>
<th>Test PPL</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM-1024 (Grave et al., 2016b)</td>
<td>48.7</td>
</tr>
<tr>
<td>GCNN-8</td>
<td>44.9</td>
</tr>
<tr>
<td>GCNN-14</td>
<td>37.2</td>
</tr>
</tbody>
</table>
Speed

- Throughput is the number of tokens per second
- Responsiveness is the number of sequential tokens per second

<table>
<thead>
<tr>
<th></th>
<th>Throughput (CPU)</th>
<th>Throughput (GPU)</th>
<th>Responsiveness (GPU)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM-2048</td>
<td>169</td>
<td>45,622</td>
<td>2,282</td>
</tr>
<tr>
<td>GCNN-9</td>
<td>121</td>
<td>29,116</td>
<td>29,116</td>
</tr>
<tr>
<td>GCNN-8 Bottleneck</td>
<td>179</td>
<td>45,878</td>
<td>45,878</td>
</tr>
</tbody>
</table>
Gating

- Gated linear units (GLU in red) converge faster
- GTU is LSTM style gating of (Oord et al, 2016)
Competitive performance can be achieved with context of less than 40 tokens.
Training algorithm

- Clipping and weight normalization speed up convergence by allowing large learning rates without divergence
Summary

- Fully convolutional model of language that is competitive with LSTMs.
- Demonstrated impact of gating mechanisms for this task.
- Shown faster response times with this approach.
Convolutional Sequence to Sequence Learning
Convolutional Sequence to Sequence Learning

- non-RNN models can outperform very well-engineered RNNs on large translation benchmarks
- Multi-hop attention
- Approach with very fast inference speed: >9x faster than RNN

Code and pre-trained models available!
Lua/Torch: [https://github.com/facebookresearch/fairseq](https://github.com/facebookresearch/fairseq)
PyTorch: [https://github.com/facebookresearch/fairseq-py](https://github.com/facebookresearch/fairseq-py)
Previous work

- ByteNet (Kalchbrenner et al. 2016)
  Characters, dilated convolutions, no attention

- Quasi-RNNs (Bradbury et al., 2016)
  Recurrent pooling of CNN outputs, but still an RNN

- Convolutional encoders (Gehring et al., 2016)
  CNN encoder, LSTM decoder
la maison de Léa <end>
. la maison de Léa <end> .

Encoder
Encoder

. la maison de Léa <end> .

maison

Léa

<end>

Encoder
. la maison de Léa <end> .

Encoder

Decoder

. <start>
. la maison de Léa <end> .

Encoder

Decoder

. <start>
. la maison de Léa <end> .
Encoder

Attention

Decoder

. la maison de Léa <end> .
. la maison de Léa <end> .
. la maison de Léa <end>.
la maison de Léa <end>
Léa
la maison de Léa <end>
la maison de Léa
. la maison de Léa <end> .

Encoder

Attention

Decoder

. <start> Léa
la maison de Léa
. la maison de Léa <end> .

Encoder

Attention

Decoder

. <start> Léa
La maison de Léa.
. la maison de Léa <end> .

Encoder

Attention

Decoder

. <start> Léa
La maison de Léa.
Léa's maison de Léa <end>.
Léa's maison de Léa.

Encoder

Attention

Decoder
Léa’s maison de Léa <end>.
Léa's maison de Léa <end>.
Léa's maison de Léa.
Léa's maison de Léa ".

Encoder

Attention

Decoder
Encoder

Attention

Decoder

. la maison de Léa <end> .

Léa's house
Convolutional S2S: Encoder

- Similar to Dauphin et al. ’17
- Input: word + position embeddings: 1, 2, 3, ...
- Weight Normalization (Salimans & Kingma, 2016)
- No batch or layer norm: initialization (He at al. ’15) and scale by sqrt(1/2)
- Repeat N times
Convolutional S2S: Encoder

- Similar to Dauphin et al. ’17
- Input: word + position embeddings: 1, 2, 3, ...
- Weight Normalization (Salimans & Kingma, 2016)
- No batch or layer norm: initialization (He et al. ’15) and scale by sqrt(1/2)
- Repeat N times
Convolutional S2S: Encoder

- Similar to Dauphin et al. ’17
- Input: word + position embeddings: 1, 2, 3, ...
- Weight Normalization (Salimans & Kingma, 2016)
- No batch or layer norm: initialization (He at al. ’15) and scale by sqrt(1/2)
- Repeat N times
Convolutional S2S: Encoder

- Similar to Dauphin et al. ’17
- Input: word + position embeddings: 1, 2, 3, ...
- Weight Normalization (Salimans & Kingma, 2016)
- No batch or layer norm: initialization (He at al. ’15) and scale by sqrt(1/2)
- Repeat N times
Convolutional S2S: Decoder

- Input: word embeddings
  + position embeddings: 1, 2, 3, ...
- Causal convolution over generated sequence so far
- Dot-product attention at every layer
Convolutional S2S: Decoder

- Input: word embeddings + position embeddings: 1, 2, 3, ...
- Causal convolution over generated sequence so far
- Dot-product attention at every layer
Convolutional S2S: Decoder

- Input: word embeddings + position embeddings: 1, 2, 3, ...
- Causal convolution over generated sequence so far
- Dot-product attention at every layer
Convolutional S2S: Decoder

- Input: word embeddings + position embeddings: 1, 2, 3, ...
- Causal convolution over generated sequence so far
- Dot-product attention at every layer
Convolutional S2S: Multi-hop Attention

- Attention in every decoder layer
- Queries contain information about previous source contexts
Convolutional S2S: Multi-hop Attention

- Attention in every decoder layer
- Queries contain information about previous source contexts
Convolutional S2S

- High training efficiency due to parallel computation in decoder
- Cross Entropy objective
- Very similar to ResNet models (Nesterov etc., He et al. ’15)
Experimental Methodology

- Translation & Summarization
- Large-scale tasks:
  - WMT’14 English-German (4.5M sentence pairs)
  - WMT’14 English-French (36M sentence pairs)
  - IWSLT’14 German-English (< 0.2M sentence pairs)
<table>
<thead>
<tr>
<th>Model</th>
<th>Vocabulary</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN ByteNet (Kalchbrenner et al., 2016)</td>
<td>Characters</td>
<td>23.75</td>
</tr>
<tr>
<td>RNN GNMT (Wu et al., 2016)</td>
<td>Word 80k</td>
<td>23.12</td>
</tr>
<tr>
<td>RNN GNMT (Wu et al., 2016)</td>
<td>Word pieces</td>
<td>24.61</td>
</tr>
</tbody>
</table>
## WMT’14 English-German Translation

<table>
<thead>
<tr>
<th>Method</th>
<th>Vocabulary</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN ByteNet (Kalchbrenner et al., 2016)</td>
<td>Characters</td>
<td>23.75</td>
</tr>
<tr>
<td>RNN GNMT (Wu et al., 2016)</td>
<td>Word 80k</td>
<td>23.12</td>
</tr>
<tr>
<td>RNN GNMT (Wu et al., 2016)</td>
<td>Word pieces</td>
<td>24.61</td>
</tr>
<tr>
<td>ConvS2S</td>
<td>BPE 40k</td>
<td>25.16</td>
</tr>
</tbody>
</table>

ConvS2S: 15 layers in encoder/decoder (10x512 units, 3x768 units, 2x2048)  
Maximum context size: 27 words
## WMT’14 English-German Translation

<table>
<thead>
<tr>
<th>Model</th>
<th>Vocabulary</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN ByteNet (Kalchbrenner et al., 2016)</td>
<td>Characters</td>
<td>23.75</td>
</tr>
<tr>
<td>RNN GNMT (Wu et al., 2016)</td>
<td>Word 80k</td>
<td>23.12</td>
</tr>
<tr>
<td>RNN GNMT (Wu et al., 2016)</td>
<td>Word pieces</td>
<td>24.61</td>
</tr>
<tr>
<td>ConvS2S</td>
<td>BPE 40k</td>
<td>25.16</td>
</tr>
<tr>
<td>Transformer (Vaswani et al., 2017)</td>
<td>Word pieces</td>
<td>28.4</td>
</tr>
</tbody>
</table>

ConvS2S: 15 layers in encoder/decoder (10x512 units, 3x768 units, 2x2048)  
Maximum context size: 27 words  

More work on non-RNN models!
<table>
<thead>
<tr>
<th>Model</th>
<th>Vocabulary</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNN GNMT (Wu et al., 2016)</td>
<td>Word 80k</td>
<td>37.90</td>
</tr>
<tr>
<td>RNN GNMT (Wu et al., 2016)</td>
<td>Word pieces</td>
<td>38.95</td>
</tr>
</tbody>
</table>
## WMT’14 English-French Translation

<table>
<thead>
<tr>
<th>Model</th>
<th>Vocabulary</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNN GNMT (Wu et al., 2016)</td>
<td>Word 80k</td>
<td>37.90</td>
</tr>
<tr>
<td>RNN GNMT (Wu et al., 2016)</td>
<td>Word pieces</td>
<td>38.95</td>
</tr>
<tr>
<td>RNN GNMT + RL (Wu et al., 2016)</td>
<td>Word pieces</td>
<td>39.92</td>
</tr>
</tbody>
</table>

References:

## WMT’14 English-French Translation

<table>
<thead>
<tr>
<th>Method</th>
<th>Vocabulary</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNN GNMT (Wu et al., 2016)</td>
<td>Word 80k</td>
<td>37.90</td>
</tr>
<tr>
<td>RNN GNMT (Wu et al., 2016)</td>
<td>Word pieces</td>
<td>38.95</td>
</tr>
<tr>
<td>RNN GNMT + RL (Wu et al., 2016)</td>
<td>Word pieces</td>
<td>39.92</td>
</tr>
<tr>
<td>ConvS2S</td>
<td>BPE 40k</td>
<td>40.51</td>
</tr>
</tbody>
</table>

ConvS2S: 15 layers in encoder/decoder (5x512 units, 4x768 units, 3x2048, 2x4096)
# WMT’14 English-French Translation

<table>
<thead>
<tr>
<th>Model</th>
<th>Vocabulary</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNN GNMT (Wu et al., 2016)</td>
<td>Word 80k</td>
<td>37.90</td>
</tr>
<tr>
<td>RNN GNMT (Wu et al., 2016)</td>
<td>Word pieces</td>
<td>38.95</td>
</tr>
<tr>
<td>RNN GNMT + RL (Wu et al., 2016)</td>
<td>Word pieces</td>
<td>39.92</td>
</tr>
<tr>
<td>ConvS2S</td>
<td>BPE 40k</td>
<td>40.51</td>
</tr>
<tr>
<td>Transformer (Vaswani et al., 2017)</td>
<td>Word pieces</td>
<td>41.0</td>
</tr>
</tbody>
</table>

ConvS2S: 15 layers in encoder/decoder (5x512 units, 4x768 units, 3x2048, 2x4096)
## Inference Speed on WMT’14 En-Fr

<table>
<thead>
<tr>
<th>Hardware</th>
<th>BLEU</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU (K80)</td>
<td>31.20</td>
<td>3028</td>
</tr>
<tr>
<td>CPU (88 cores)</td>
<td>31.20</td>
<td>1322</td>
</tr>
<tr>
<td>TPU</td>
<td>31.21</td>
<td>384</td>
</tr>
</tbody>
</table>

ntst1213 (6003 sentences)
### Inference Speed on WMT’14 En-Fr

<table>
<thead>
<tr>
<th>Hardware</th>
<th>BLEU</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNN GNMT (Wu et al., 2016)</td>
<td>31.20</td>
<td>3028</td>
</tr>
<tr>
<td>GPU (K80)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RNN GNMT (Wu et al., 2016)</td>
<td>31.20</td>
<td>1322</td>
</tr>
<tr>
<td>CPU (88 cores)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RNN GNMT (Wu et al., 2016)</td>
<td>31.21</td>
<td>384</td>
</tr>
<tr>
<td>TPU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ConvS2S, beam=5</td>
<td>34.10</td>
<td>587</td>
</tr>
<tr>
<td>GPU (K40)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ConvS2S, beam=1</td>
<td>33.45</td>
<td>327</td>
</tr>
<tr>
<td>GPU (K40)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ntst1213 (6003 sentences)
Inference Speed on WMT’14 En-Fr

<table>
<thead>
<tr>
<th>Model</th>
<th>Hardware</th>
<th>BLEU</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNN GNMT (Wu et al., 2016)</td>
<td>GPU (K80)</td>
<td>31.20</td>
<td>3028</td>
</tr>
<tr>
<td>RNN GNMT (Wu et al., 2016)</td>
<td>CPU (88 cores)</td>
<td>31.20</td>
<td>1322</td>
</tr>
<tr>
<td>RNN GNMT (Wu et al., 2016)</td>
<td>TPU</td>
<td>31.21</td>
<td>384</td>
</tr>
<tr>
<td>ConvS2S, beam=5</td>
<td>GPU (K40)</td>
<td>34.10</td>
<td>587</td>
</tr>
<tr>
<td>ConvS2S, beam=1</td>
<td>GPU (K40)</td>
<td>33.45</td>
<td>327</td>
</tr>
<tr>
<td>ConvS2S, beam=1</td>
<td>GPU (GTX-1080ti)</td>
<td>33.45</td>
<td>142</td>
</tr>
<tr>
<td>ConvS2S, beam=1</td>
<td>CPU (48 cores)</td>
<td>33.45</td>
<td>142</td>
</tr>
</tbody>
</table>

ntst1213 (6003 sentences)
Text Summarization

Compress first sentence of a news article into a headline (Rush et al. 2016)

<table>
<thead>
<tr>
<th>Method</th>
<th>Rouge-1</th>
<th>Rouge-2</th>
<th>Rouge-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNN Likelihood optimized (Shen et al. 2016)</td>
<td>32.7</td>
<td>15.2</td>
<td>30.6</td>
</tr>
<tr>
<td>RNN Rouge-optimized (Shen et al. 2016)</td>
<td>36.5</td>
<td>16.6</td>
<td>33.4</td>
</tr>
<tr>
<td>RNN repeated words (Suzuki &amp; Nagata, 2017)</td>
<td>36.3</td>
<td>17.3</td>
<td>33.9</td>
</tr>
<tr>
<td>ConvS2S</td>
<td>35.9</td>
<td>17.5</td>
<td>33.3</td>
</tr>
</tbody>
</table>

ConvS2S: 6 layers in encoder/decoder, nhid=256
Summary

- Alternative architecture for sequence to sequence learning
- Higher accuracy than models of similar size, despite fixed size context
- Faster generation (9x faster on lesser hardware)

Code & pre-trained models:
Lua Torch: http://github.com/facebookresearch/fairseq
PyTorch: http://github.com/facebookresearch/fairseq-py
Beam Search for Seq2Seq
Generating from Seq2Seq

At test time, we want to \textbf{generate} from $P(y|x)$ with $y \in \{1, \ldots, V\}^*$

- \textbf{sampling} is easy, decomposability allows left to right sampling
  \[ y \sim P(y|x) = \prod_{t} P(y_t|y_{1}^{t-1}, x) \]

- \textbf{MAP inference} is hard
  \[ \hat{y} = \arg \max_y P(y|x) \text{ with } y \in \{1, \ldots, V\}^* \]

- \textbf{Beam} approximates MAP inference
Beam Search

3 prefixes

- the little cat
- the tiny feline
- the small kitten

3 * V expansions

Top 3 expansions

- jumps
- runs
- jumps
On Search Space Size

- Effective vocabulary is rather small
- e.g. vocabulary selection with alignment method (WMT’14 en-de)
On Search Space Size

- beam results are prominent

- e.g. 20 time steps above 0.9 means more than \(0.9^{20} \approx \frac{1}{8}\)
Beam Search

- Q: Does it work? Good approximation of MAP?
- Q: Is it a good idea? Do we care about MAP?
Beam Search

- Q: Does it work? good approximation of MAP?
- Q: Is it a good idea? Do we care about MAP?

Sample k times & choose by logrprob
Beam Search

- Q: Does it work? good approximation of MAP?
- Q: Is it a good idea? Do we care about MAP?

Sample k times & choose by BLEU
Better model = better estimation not better search
Sequence-Level Training

- Maximizing token likelihood does not consider BLEU

**Sequence Level Training with Recurrent Neural Networks**
Marc-Aurelio Ranzato, Sumit Chopra, Michael Auli, Wojciech Zaremba - Nov 2015
[Reinforce, Williams 92][expected BLEU]

**Sequence-to-Sequence Learning as Beam-Search Optimization**
Sam Wiseman, Alexander M. Rush - June 2016
[learning as search optimization (LaSO), Daume and Marcu 2005][search BLEU]

**Google's Neural Machine Translation System**
Wu et al - Oct 2016
[close to Ranzato'15 + new reward][expected GLEU]

**An Actor-Critic Algorithm for Sequence Prediction**
Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, J. Pineau, A. Courville, Y. Bengio - July '16
[actor-critic, Sutton'84][expected BLEU]
Sequence-Level Training

IWSLT'14 de-en

<table>
<thead>
<tr>
<th></th>
<th>LL</th>
<th>Seq</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ranzato</td>
<td>20.3</td>
<td>21.9</td>
</tr>
<tr>
<td>Bahdanau</td>
<td>21.5</td>
<td>22.6</td>
</tr>
<tr>
<td>Wiseman</td>
<td>24.0</td>
<td>26.4</td>
</tr>
<tr>
<td>Conv</td>
<td><strong>30.2</strong></td>
<td>----</td>
</tr>
</tbody>
</table>

WMT'14 en-fr

<table>
<thead>
<tr>
<th></th>
<th>LL</th>
<th>LL-E</th>
<th>Seq</th>
<th>Seq-E</th>
</tr>
</thead>
<tbody>
<tr>
<td>GNMT</td>
<td>39.0</td>
<td>40.3</td>
<td>39.9</td>
<td>41.2</td>
</tr>
<tr>
<td>Conv</td>
<td>40.5</td>
<td><strong>41.6</strong></td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>Attn</td>
<td>41.0</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
</tbody>
</table>

WMT'14 en-de

<table>
<thead>
<tr>
<th></th>
<th>LL</th>
<th>LL-E</th>
<th>Seq</th>
<th>Seq-E</th>
</tr>
</thead>
<tbody>
<tr>
<td>GNMT</td>
<td>24.7</td>
<td>26.2</td>
<td>24.6</td>
<td>26.3</td>
</tr>
<tr>
<td>Conv</td>
<td>25.2</td>
<td>26.4</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>Attn</td>
<td><strong>28.4</strong></td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
</tbody>
</table>

small gains compared to
- architecture search
- ensembling
Sequence-Level Training

Pros
- optimize end loss

Cons
- slower, need joint loss and/or init
- optimize expectation, not MAP (except Wiseman+Rush)
- smaller gains than architecture search or ensembling
Conclusions

Gated Convolutional Architecture
- Gating allows for linear model initially
- Fast at train, inference
- Accurate for LM, MT

Training Seq2Seq for Machine Translation
- Most progress attributed to architecture & ensembling
- Less progress due to BLEU optimization
Future Work

- Domain adaptation
- Leverage monolingual data
  - e.g. study back-translation at scale
- Understand ensemble
- Model diversity, uncertainty
Questions?
## Results Multi-hop Attention

<table>
<thead>
<tr>
<th>Attn Layers</th>
<th>PPL</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,2,3,4,5</td>
<td>6.65</td>
<td>21.63</td>
</tr>
<tr>
<td>1,2,3,4</td>
<td>6.70</td>
<td>21.54</td>
</tr>
<tr>
<td>1,2,3</td>
<td>6.95</td>
<td>21.36</td>
</tr>
<tr>
<td>1,2</td>
<td>6.92</td>
<td>21.47</td>
</tr>
<tr>
<td>1,3,5</td>
<td>6.97</td>
<td>21.10</td>
</tr>
<tr>
<td>1</td>
<td>7.15</td>
<td>21.26</td>
</tr>
<tr>
<td>2</td>
<td>7.09</td>
<td>21.30</td>
</tr>
<tr>
<td>3</td>
<td>7.11</td>
<td>21.19</td>
</tr>
<tr>
<td>4</td>
<td>7.19</td>
<td>21.31</td>
</tr>
<tr>
<td>5</td>
<td>7.66</td>
<td>20.24</td>
</tr>
<tr>
<td>WMT’14 English-French</td>
<td>BLEU</td>
<td></td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------</td>
<td></td>
</tr>
<tr>
<td>Zhou et al. (2016)</td>
<td>40.4</td>
<td></td>
</tr>
<tr>
<td>Wu et al. (2016) GNMT</td>
<td>40.35</td>
<td></td>
</tr>
<tr>
<td>Wu et al. (2016) GNMT + RL</td>
<td>41.16</td>
<td></td>
</tr>
<tr>
<td>ConvS2S</td>
<td>41.44</td>
<td></td>
</tr>
<tr>
<td>ConvS2S (10 models)</td>
<td>41.62</td>
<td></td>
</tr>
</tbody>
</table>
Network Depth

![Bar chart showing BLEU scores for different network depths and key word sizes.]

- **kw=3**: Layers=5, BLEU=20.0; Layers=9, BLEU=20.7; Layers=13, BLEU=21.3
- **kw=5**: Layers=5, BLEU=21.3; Layers=9, BLEU=22.0
- **kw=7**: Layers=5, BLEU=20.7; Layers=9, BLEU=21.3; Layers=13, BLEU=21.3
Convolutional S2S: Multi-Hop Attention

1st Layer

2nd Layer
Convolutional S2S: Multi-Hop Attention

3rd Layer

4th Layer
Convolutional S2S: Multi-Hop Attention

5th Layer

6th Layer
Convolutional S2S: Multi-Hop Attention

7th Layer

8th Layer