Convolutional Sequence to Sequence Learning

Denis Yarats
with Jonas Gehring, Michael Auli, David Grangier, Yann Dauphin
Facebook AI Research
Sequence generation

• Need to model a conditional distribution

\[ Pr(x) = \prod_{t} Pr(x_t | x_{1:t-1}) \]

• Repeatedly predict what will happen next, use your past predictions as if they were real
Sequence generation

- Language modeling
- Machine translation
- Speech generating
- Image generation
- etc.
Sequence generation

• How to model $\Pr(x) = \prod_t \Pr(x_t|x_{1:t-1})$?

• Let's use Recurrent Neural Network
Recurrent Neural Network

- Like feed forward networks, except allows self connections
- Self connections are used to build an internal representation of past inputs
- They give the network memory

Figures from: http://colah.github.io/posts/2015-08-Understanding-LSTMs
Recurrent Neural Network

• Given list of inputs $x_1, \ldots, x_{t-1}, x_t, x_{t+1}, \ldots, x_T$

• At each timestamp do:

$$h_t = \sigma \left( W^{(hh)} h_{t-1} + W^{(hx)} x_{[t]} \right)$$

$$\hat{y}_t = \text{softmax} \left( W^{(S)} h_t \right)$$

• Then: $\hat{P}(x_{t+1} = v_j \mid x_t, \ldots, x_1) = \hat{y}_{t,j}$

Recurrent Neural Network

- $\hat{y} \in \mathbb{R}^{|V|}$ is probability distribution over vocabulary
- To train the network, minimize cross-entropy:

$$J^{(t)}(\theta) = - \sum_{j=1}^{|V|} y_{t,j} \log \hat{y}_{t,j}$$

Recurrent Neural Network

- The notorious vanishing/exploding gradients problem

\[ \frac{\partial E_t}{\partial W} = \sum_{k=1}^{t} \frac{\partial E_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial W} \]

where

\[ h_t = W f(h_{t-1}) + W^{(hx)} x_{[t]} \]

thus

\[ \frac{\partial h_t}{\partial h_k} = \prod_{j=k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}} \]

Long Short Term Memory (LSTM)

- Modification of RNN to have longer memory
- Additional memory cell to store information
- RNN overwrites the hidden state, LSTM adds to the hidden state
- Hochreiter et al.
Sequence to Sequence

- Make NN to read one sequence and produce another
- Use 2 LSTMs
- Sutskever et al. 2014

Sequence to Sequence

- Encoder - encodes input sequence
- Decoder - generates output sequence, conditioning on the input representation

Attention

- Incorporate all hidden states of encoder, rather than the last one
- Bahdahau et al. 2014

Figures from: https://distill.pub/2016/augmented-rnns
Attention

- Use weighted combination of each encoder hidden state
- Alignment mode 
  \[ e_{ij} = a(s_{i-1}, h_j) \]
- Weights: 
  \[ \alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})} \]
- Weighted sum 
  \[ c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j \]

Figures from: https://distill.pub/2016/augmented-rnns
Attention

• Example of attention for machine translation

Figures from: https://distill.pub/2016/augmented-rnns
LSTM based Seq2Seq: Problems

- Still challenging to model long term dependencies
- RNNs are hard to parallelize because of non-homogeneous nature
- Convolutional neural networks to the rescue?
Convolutional Neural Network

- Great success in computer vision: AlexNext, VGG, ResNet, ...
- Efficient implementation on GPU

Figures from: http://cs231n.github.io/convolutional-networks
Convolutional Neural Network

- Each output neuron is a linear combination of input neurons in some spatial neighborhood.
- Unlike a feed forward network, where it is connected to every input neuron.
- Parameter sharing and better scalability.

Figures from: http://cs231n.github.io/convolutional-networks
Convolutional Neural Network

- Convolution can also be used for sequences
- Temporal convolution

Convolutional Neural Network

- Hierarchical processing: bottom-up vs. left-right
- Homogeneous: all elements processed in the same way
- Scalable computation: parallelizable, suited for GPUs
ConvS2S: LSTM based Seq2Seq

Bi-directional LSTM Encoder

Soft Attention

LSTM Decoder
ConvS2S: Convolutional Encoder
ConvS2S: Convolutional Encoder
ConvS2S: Convolutional Decoder

Convolutional Encoder

Soft Attention

LSTM Decoder
ConvS2S: Convolutional Decoder
ConvS2S: Encoder

Convolutional block structure for encoder:

- Gated linear units, residual connections
- $z_0 = \text{embeddings}$
  - word or sub-word embedding
  - plus position embedding: $0,1,2,...$
**ConvS2S: Decoder**

Convolutional block structure for decoder:

- Gated linear units and residual connections
- $h_0 = \text{embeddings (word + pos)}$
- Soft attention pass at every layer
ConvS2S: All together

Putting it all together:

- Here: single-layer encoder and decoder
- High training efficiency due to parallel computation in decoder
ConvS2S: Architecture

- 15 layers in both encoder and decoder
- Convolutional kernel size is 3
- Hidden size gradually increases from 512 to 4096
- Embedding size is 512
ConvS2S: Training

- Optimizer: Nesterov with momentum
- Learning rate 0.25, momentum 0.99
- Gradient clipping if norm exceeds 0.1
- Batch size 64
- Data parallel training: all-reduct gradients after each iteration
- Model is implemented in Torch (PyTorch implementation is coming)
ConvS2S: Tricks

• WeighNorm (Salimans et al. 2016)
• Dropout (Srivastava et. al. 2014)
• Scale outputs of each layer to normalize variance
• Careful weight initialization to ensure normally distributed variance
# ConvS2S: Results

Results on English-German (WMT'14, newstest2014)

<table>
<thead>
<tr>
<th>System</th>
<th>Vocabulary</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>ByteNet v2 (Kalchbrenner et al., 2016)</td>
<td>Characters</td>
<td>23.75</td>
</tr>
<tr>
<td>GNMT (Wu et al., 2016)</td>
<td>Word 80k</td>
<td>23.12</td>
</tr>
<tr>
<td>GNMT (Wu et al., 2016)</td>
<td>Word pieces</td>
<td>24.61</td>
</tr>
<tr>
<td><strong>ConvS2S</strong></td>
<td>BPE 40k</td>
<td>25.16</td>
</tr>
<tr>
<td>Transformer (Vaswani et al., 2017)</td>
<td>Word pieces</td>
<td><strong>28.4</strong></td>
</tr>
</tbody>
</table>
### ConvS2S: Results

Results on English-French (WMT'14, newstest2014)

<table>
<thead>
<tr>
<th>System</th>
<th>Vocabulary</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>GNMT (Wu et al., 2016)</td>
<td>Word 80k</td>
<td>37.90</td>
</tr>
<tr>
<td>GNMT (Wu et al., 2016)</td>
<td>Word pieces</td>
<td>38.95</td>
</tr>
<tr>
<td>GNMT + RL (Wu et al., 2016)</td>
<td>Word pieces</td>
<td>39.92</td>
</tr>
<tr>
<td><strong>ConvS2S</strong></td>
<td><strong>BPE 40k</strong></td>
<td><strong>40.46</strong></td>
</tr>
<tr>
<td>Transformer (Vaswani et al., 2017)</td>
<td>Word pieces</td>
<td>41.0*</td>
</tr>
</tbody>
</table>
## ConvS2S: Speed

Translation speed on English-French (WMT'14, dev set)

<table>
<thead>
<tr>
<th>System</th>
<th>Hardware</th>
<th>BLEU</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GNMT (Wu et al., 2016)</td>
<td>CPU (88 cores)</td>
<td>31.20</td>
<td>1322</td>
</tr>
<tr>
<td>GNMT (Wu et al., 2016)</td>
<td>GPU (K80)</td>
<td>31.20</td>
<td>3028</td>
</tr>
<tr>
<td>GNMT + RL (Wu et al., 2016)</td>
<td>TPU</td>
<td>31.21</td>
<td>384</td>
</tr>
<tr>
<td>ConvS2S, beam=5</td>
<td>CPU (48 cores)</td>
<td>34.10</td>
<td>482</td>
</tr>
<tr>
<td>ConvS2S, beam=5</td>
<td>GPU (K40)</td>
<td>34.10</td>
<td>587</td>
</tr>
<tr>
<td>ConvS2S, beam=5</td>
<td>GPU (GTX-1080ti)</td>
<td>34.10</td>
<td>406</td>
</tr>
<tr>
<td>ConvS2S, beam=1</td>
<td>CPU (48 cores)</td>
<td>33.45</td>
<td>142</td>
</tr>
</tbody>
</table>
ConvS2S: Training

<table>
<thead>
<tr>
<th>Dataset</th>
<th># Pairs (in million)</th>
<th>Hardware</th>
<th>Days</th>
</tr>
</thead>
<tbody>
<tr>
<td>WMT14 EN-DE</td>
<td>4.5</td>
<td>1 x GPU (K40)</td>
<td>18</td>
</tr>
<tr>
<td>WMT14 EN-FR</td>
<td>36</td>
<td>8 x GPU (K40)</td>
<td>37</td>
</tr>
</tbody>
</table>
ConvS2S: Multi-Hop Attention

1st Layer

2nd Layer
ConvS2S: Multi-Hop Attention

3rd Layer

4th Layer
ConvS2S: Multi-Hop Attention

5th Layer

6th Layer
A novel approach to neural machine translation
Convolutional Sequence to Sequence Learning

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, Yann N. Dauphin
(Submitted on 8 May 2017 (v1), last revised 25 Jul 2017 (this version, v3))

The prevalent approach to sequence to sequence learning maps an input sequence to a variable length output sequence via recurrent neural networks. We introduce an architecture based entirely on convolutional neural networks. Compared to recurrent models, computations over all elements can be fully parallelized during training and optimization is easier since the number of non-linearities is fixed and independent of the input length. Our use of gated linear units eases gradient propagation and we equip each decoder layer with a separate attention module. We outperform the accuracy of the deep LSTM setup of Wu et al. (2016) on both WMT'14 English–German and WMT'14 English–French translation at an order of magnitude faster speed, both on GPU and CPU.
Source Code

Facebook AI Research Sequence-to-Sequence Toolkit

- 19 commits
- 1 branch
- 0 releases
- 0 contributors

Branch: master
New pull request
Create new file
Upload files
Find file
Clone or download

BSD-3-Clause