Machine Translation

10: Advanced Neural Machine Translation Architectures

Rico Sennrich

University of Edinburgh

Today's Lecture

so far

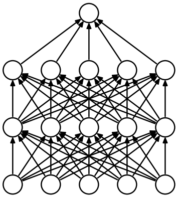
- we discussed RNNs as encoder and decoder
- we discussed some architecture variants:
  - RNN vs. GRU vs. LSTM
  - attention mechanisms

today

- some important components of neural MT architectures:
  - dropout
  - layer normalization
  - deep networks
- non-recurrent architectures:
  - convolutional networks
  - self-attentional networks

1. General Architecture Variants
2. NMT with Convolutional Neural Networks
3. NMT with Self-Attention

Dropout

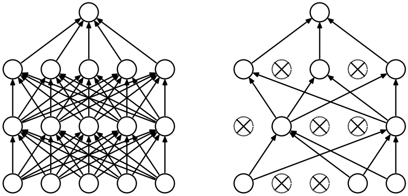


(a) Standard Neural Net (b) After applying dropout.

- wacky idea: randomly set hidden states to 0 during training
- motivation: prevent "co-adaptation" of hidden units
  $\rightarrow$ better generalization, less overfitting

[Srivastava et al., 2014]

Dropout



(a) Standard Neural Net (b) After applying dropout.

- implementation:
  - for training, multiply layer with "dropout mask"
  - randomly sample new mask for each layer and training example
  - hyperparameter p: probability that state is retained
    (some tools use p as probability that state is dropped)
  - at test time, don't apply dropout,
    but re-scale layer with p to ensure expected output is the same
  - (you can also re-scale by $\frac{1}{p}$ at training time instead)

[Srivastava et al., 2014]

Dropout and RNNs

for recurrent connections, applying dropout at every time step blocks information flow

solution 1: only apply dropoput to feedforward connections

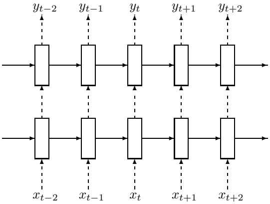


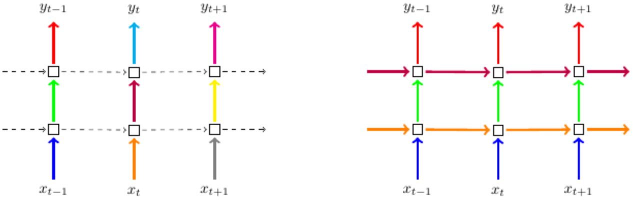
Figure 2: Regularized multilayer RNN. The dashed arrows indicate connections where dropout is applied, and the solid lines indicate connections where dropout is not applied.

[Zaremba et al., 2014]

Dropout and RNNs

for recurrent connections, applying dropout at every time step blocks information flow

solution 2: variational dropout: use same dropout mask at each time step



[Gal, 2015]

Layer Normalization

- if input distribution to NN layer changes, parameters need to adapt to this **covariate shift**
- especially bad: RNN state grows/shrinks as we go through sequence
- normalization of layers reduces shift, and improves training stability
- re-center and re-scale each layer $\mathbf{a}$ (with H units)
- two bias parameters, $\mathbf{g}$ and $\mathbf{b}$, restore original representation power

$$\mu = \frac{1}{H}\sum_{i=1}^{H} a_i$$

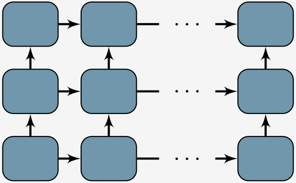
$$\sigma = \sqrt{\frac{1}{H}\sum_{i=1}^{H}(a_i-\mu)^2}$$

$$\mathbf{h} = \left[\frac{\mathbf{g}}{\sigma} \odot (\mathbf{a}-\mu) + \mathbf{b}\right]$$

Deep Networks

- increasing model depth often increases model performance
- example: stack RNN:

$$h_{i,1} = g(U_1 h_{i-1,1} + W_1 x_i)$$
$$h_{i,2} = g(U_2 h_{i-1,2} + W_2 h_{i,1})$$
$$h_{i,3} = g(U_3 h_{i-1,3} + W_3 h_{i,2})$$



Deep Networks

- often necessary to combat vanishing gradient: residual connections between layers:

$$
\begin{aligned}
h_{i,1} &= g(U_1 h_{i-1,1} + W_1 x_i) \\
h_{i,2} &= g(U_2 h_{i-1,2} + W_2 h_{i,1}) + \mathbf{h_{i,1}} \\
h_{i,3} &= g(U_3 h_{i-1,3} + W_3 h_{i,2}) + \mathbf{h_{i,2}}
\end{aligned}
$$

Layer Normalization and Deep Models: Results from UEDIN@WMT17

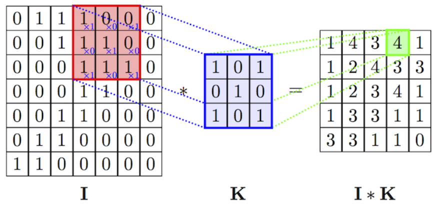
system	CS→EN 2017	DE→EN 2017	LV→EN 2017	RU→EN 2017	TR→EN 2017	ZH→EN 2017
baseline	27.5	32.0	16.4	31.3	19.7	21.7
+layer normalization	28.2	32.1	17.0	32.3	18.8	22.5
+deep model	28.9	33.5	16.6	32.7	20.6	22.9

- layer normalization and deep models generally improve quality
- layer normalization also speeds up convergence when training (fewer updates needed)
- dropout used for low-resource system (TR→EN)

Convolutional Networks

core idea: rather than using fully connected matrix between two layers, repeatedly compute dot product with small *filter* (or *kernel*)

2d convolution with 3x3 kernel

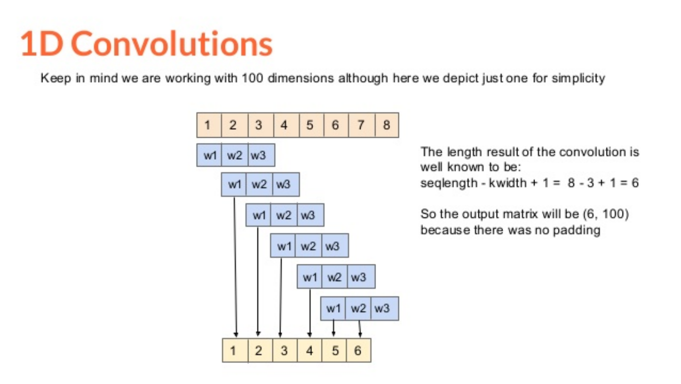


https://cambridgespark.com/content/tutorials/convolutional-neural-networks-with-keras/index.html

Convolutional Networks

when working with sequences, we often use 1d convolutions

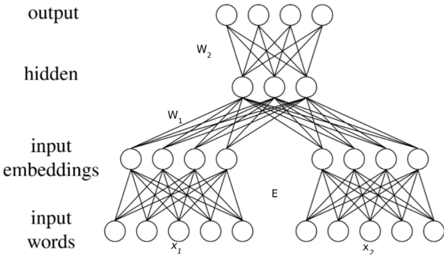
1d convolution with width-3 kernel



https://www.slideshare.net/xavigiro/recurrent-neural-networks-2-d2l3-deep-learning-for-speech-and-language-upc-2017

Convolutional Networks

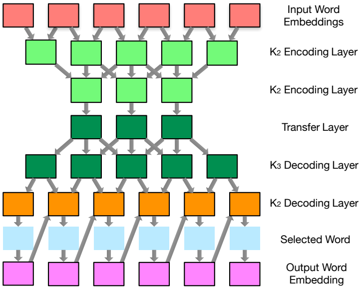
(this is similar to how we obtained hidden state for n-gram LM)



[Vaswani et al., 2013]

Convolutional Neural Machine Translation

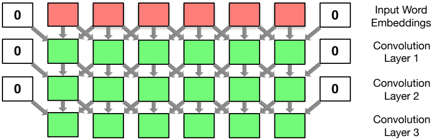
convolutional encoder actually predates RNN encoder



architecture of [Kalchbrenner and Blunsom, 2013], as illustrated in P. Koehn, Neural Machine Translation

Convolutional Neural Machine Translation with Attention

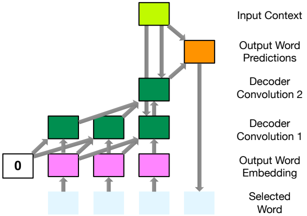
- to keep representation size constant, use padding
  $\rightarrow$ similar variable-size representation as RNN encoder
- kernel can be applied to all windows in parallel



architecture of [Gehring et al., 2017], as illustrated in P. Koehn, Neural Machine Translation

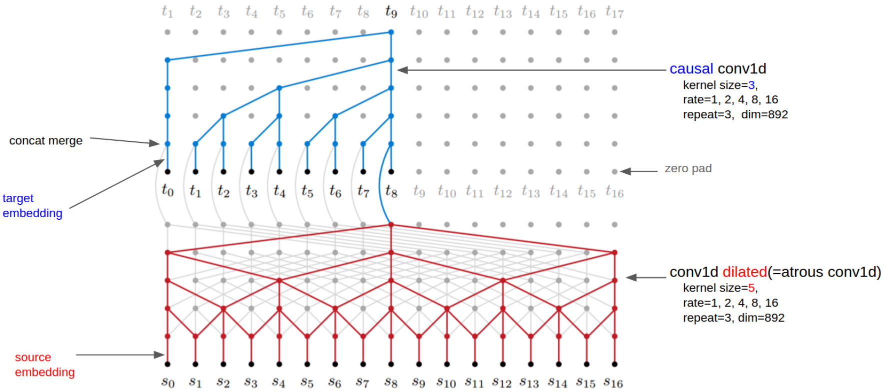
Convolutional Neural Machine Translation with Attention

- use your favourite attention mechanism to obtain input context
- in decoder, information from future tokens is masked during training
- effective context window depends on network depth and kernel size



architecture of [Gehring et al., 2017], as illustrated in P. Koehn, Neural Machine Translation

Convolutional Neural Machine Translation (ByteNet)



architecture of [Kalchbrenner et al., 2016]

Attention Is All You Need [Vaswani et al., 2017]

- same criticisms of recurrent architecture:
  recurrent computations cannot be parallelized
- core idea: instead of fixed-width convolutional filter, use attention
- there are different flavours of self-attention
  here: attend over previous layer of deep network

https://nlp.stanford.edu/seminar/details/lkaiser.pdf

Attention Is All You Need [Vaswani et al., 2017]

Transformer architecture

- stack of N self-attention layers
- self-attention in decoder is *masked*
- decoder also attends to encoder states
- *Add & Norm*: residual connection and layer normalization

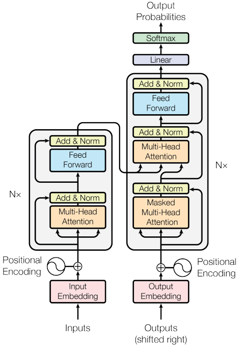


Figure 1: The Transformer - model architecture.

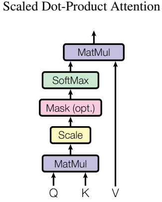
[Vaswani et al., 2017]

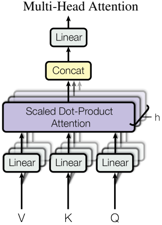
Multi-Head Attention

- basic attention mechanism in AIAYN: Scaled Dot-Product Attention

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

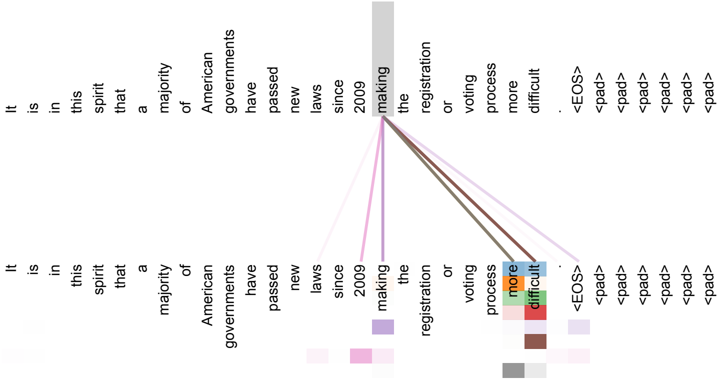
- query Q is decoder/encoder state (for attention/self-attention)
- key K and value V are encoder hidden states
- multi-head attention: use h parallel attention mechanisms with low-dimensional, learned projections of Q, K, and V





Multi-Head Attention

motivation for multi-head attention:
different heads can attend to different states



Comparison

empirical comparison difficult

- some components could be mix-and-matched
  - choice of attention mechanism
  - choice of positional encoding
  - hyperparameters and training tricks
- different test sets and/or evaluation scripts

Comparison

Sockeye [Hieber et al., 2017] (EN-DE; newstest2017)

system	Bleu
deep LSTM	25.6
Convolutional	24.6
Transformer	**27.5**

Marian (EN-DE; newstest2016)

system	Bleu
deep LSTM	32.6
Transformer	**33.4**

https://github.com/marian-nmt/marian-dev/issues/116#issuecomment-340212787

Empiricism vs. Theory

- our theoretical understanding of neural networks lags behind empirical progress
- there are some theoretical arguments why architectures work well... (e.g. self-attention reduces distance in network between words)
- ...but these are very speculative

Further Reading

- required reading: Koehn, 13.7
- consider original literature cited on relevant slides

Bibliography I

Gal, Y. (2015).
A Theoretically Grounded Application of Dropout in Recurrent Neural Networks.
ArXiv e-prints.

Gehring, J., Auli, M., Grangier, D., Yarats, D., and Dauphin, Y. N. (2017).
Convolutional Sequence to Sequence Learning.
CoRR, abs/1705.03122.

Hieber, F., Domhan, T., Denkowski, M., Vilar, D., Sokolov, A., Clifton, A., and Post, M. (2017).
Sockeye: A Toolkit for Neural Machine Translation.
ArXiv e-prints.

Kalchbrenner, N. and Blunsom, P. (2013).
Recurrent Continuous Translation Models.
In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, Seattle. Association for Computational Linguistics.

Kalchbrenner, N., Espeholt, L., Simonyan, K., van den Oord, A., Graves, A., and Kavukcuoglu, K. (2016).
Neural Machine Translation in Linear Time.
ArXiv e-prints.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014).
Dropout: A Simple Way to Prevent Neural Networks from Overfitting.
Journal of Machine Learning Research, 15:1929–1958.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017).
Attention Is All You Need.
CoRR, abs/1706.03762.

Bibliography II

Vaswani, A., Zhao, Y., Fossum, V., and Chiang, D. (2013).
Decoding with Large-Scale Neural Language Models Improves Translation.
In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, pages 1387–1392, Seattle, Washington, USA.

Zaremba, W., Sutskever, I., and Vinyals, O. (2014).
Recurrent Neural Network Regularization.
CoRR, abs/1409.2329.