Quasi-Recurrent Neural Networks

James Bradbury, Stephen Merity, Caiming Xiong, Richard Socher

Outline

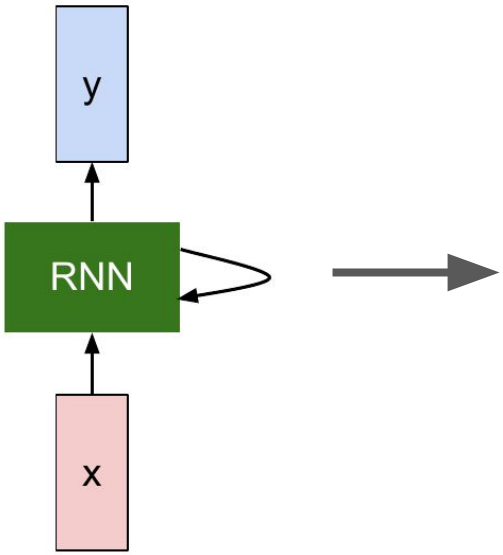
1. Introduction
2. Related Work
3. Architecture
   a. Contributions
   b. Recurrent Pooling
   c. Comparison to LSTM
4. Variations
   a. Zoneout Regularization
   b. DenseNet
   c. Encoder-Decoder Models
5. Evaluation and Results
6. Conclusions

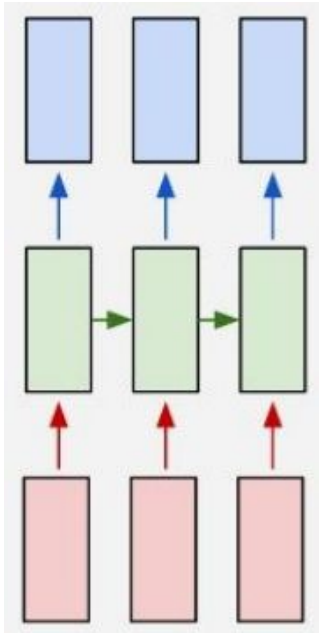
Introduction

- Recurrent Neural Networks limitations:
  - Sequential dependencies to previous computations limits parallelism
  - Difficult to train for longer sequences due to exploding/vanishing gradient
- Motivation: Combining the powers of convolutional networks with recurrent networks.

$$h_t = f_W(h_{t-1}, x_t)$$

$$y = W_h * h_t$$





Related work: Strongly-typed RNNs

- Introduces a “type system” to improve representational power of mainstream RNNs
  - Inspiration from unit type preservation principles in physics and functional programming disciplines
- Motivated from **PCA** analogy when performed on an unrolled RNN points out to a flaw in the traditional RNN update steps:

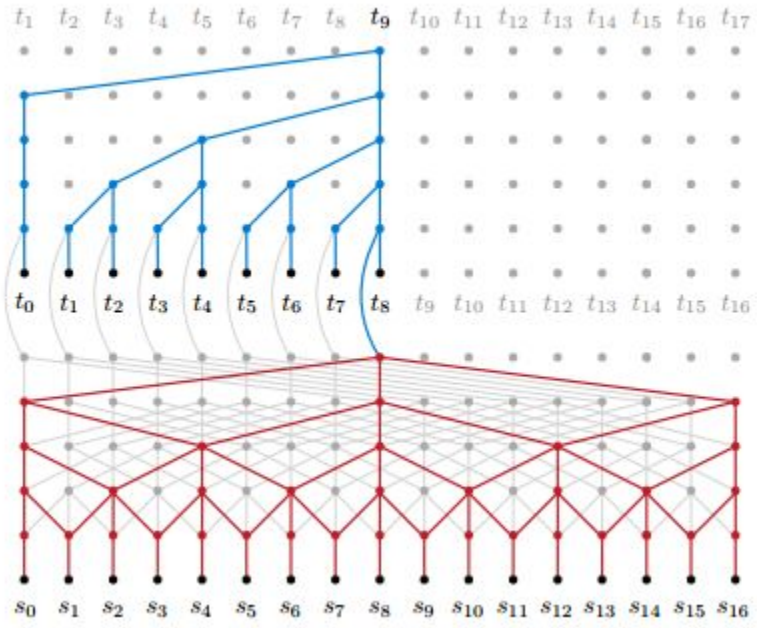
$$\mathbf{h}_t = \sum_{s=1}^{t} \mathbf{V}^{t-s} \cdot \mathbf{z}_s.$$

$$h_t = z_1 * V + z_2 * V^2 + .. \; z_t * V^t$$

- Summation of different units makes the result hard to reason about.

Related work: ByteNet

- Uses CNN to perform sequential tasks
- A binary tree as the network architecture, encoder-decoder units
- Improved parallelism and the gradient distribution against RNNs



Related work: PixelCNN

- Uses *masked convolutions* for sequence prediction
  - Shifting input signal to change so that convolution output corresponds to the next pixel
  - Essentially allows using existing implementations of the convolutional units
- Proposes generative model using CNN and RNNs

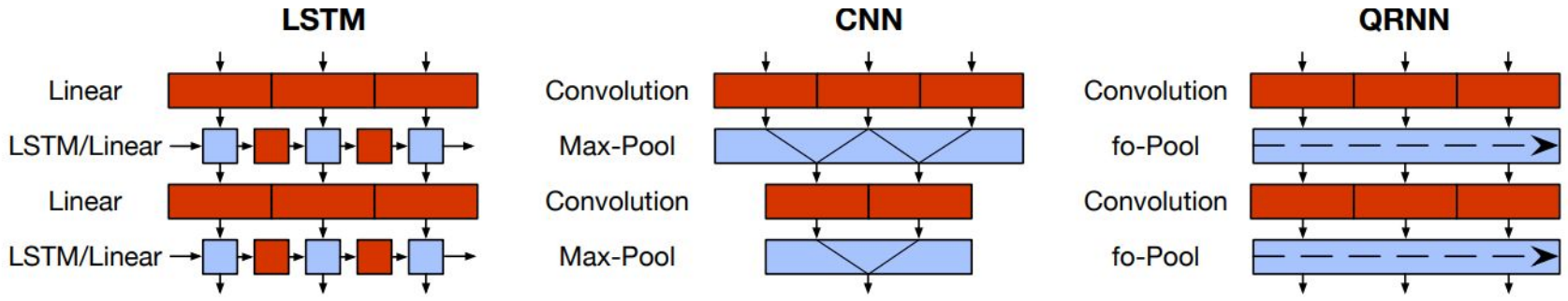
Contributions

- Learn parameters to a recurrent neural network using convolutional filters
  - Three options *f-pooling, fo-pooling, ifo pooling*
- Propose a generalization to the T-RNN approach
  - No hyperparameters in the recurrent step
- Applying recently techniques:
  - Zoneout regularization []
  - Dense Convolutional Networks []
  - Encoder-Decoder Models

Architecture: Main Idea

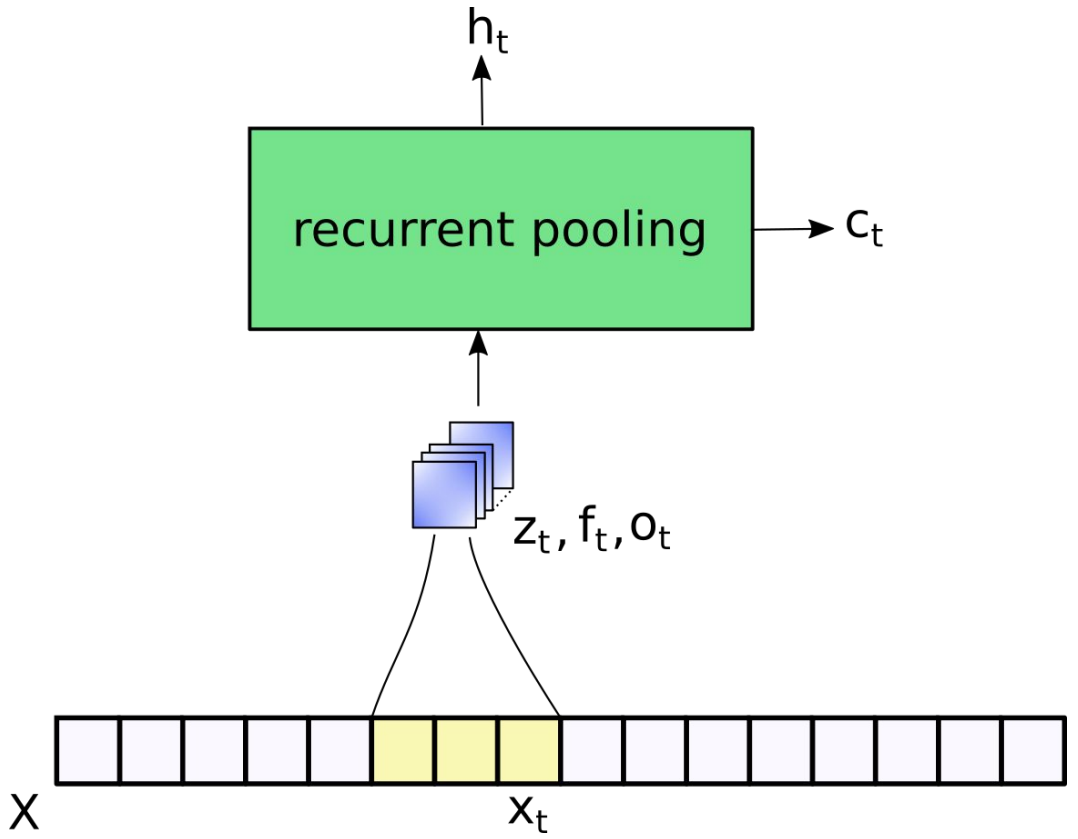
- Learn weights to be used in the recurrent step using convolutional filters
- Using convolutions to compute *n-gram* features

$$\mathbf{Z} = \tanh(\mathbf{W}_z * \mathbf{X})$$
$$\mathbf{F} = \sigma(\mathbf{W}_f * \mathbf{X})$$
$$\mathbf{O} = \sigma(\mathbf{W}_o * \mathbf{X}),$$



Architecture: Main Idea

- Learn weights to be used in the recurrent step using convolutional filters



Architecture: Recurrent Pooling Step

- Convolutional features are multiplied *element-wise*
- Basic model *f*-pooling (*averaging the input*):

$$\mathbf{h}_t = \mathbf{f}_t \odot \mathbf{h}_{t-1} + (1 - \mathbf{f}_t) \odot \mathbf{z}_t,$$

- fo-pooling (output gate):

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + (1 - \mathbf{f}_t) \odot \mathbf{z}_t$$
$$\mathbf{h}_t = \mathbf{o}_t \odot \mathbf{c}_t.$$

- ifo-pooling (independent input and forget gate):

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{z}_t$$
$$\mathbf{h}_t = \mathbf{o}_t \odot \mathbf{c}_t.$$

Architecture: Comparison to an LSTM cell

- Although the structure looks similar not exactly same as the LSTM:
  - Hidden states are not used when computing the gate signals
  - Hence the name *quasi-recurrent*

QRNN

$$\mathbf{z}_t = \tanh(\mathbf{W}^1_z \mathbf{x}_{t-1} + \mathbf{W}^2_z \mathbf{x}_t)$$
$$\mathbf{f}_t = \sigma(\mathbf{W}^1_f \mathbf{x}_{t-1} + \mathbf{W}^2_f \mathbf{x}_t)$$
$$\mathbf{o}_t = \sigma(\mathbf{W}^1_o \mathbf{x}_{t-1} + \mathbf{W}^2_o \mathbf{x}_t).$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + (1 - \mathbf{f}_t) \odot \mathbf{z}_t$$
$$\mathbf{h}_t = \mathbf{o}_t \odot \mathbf{c}_t.$$

LSTM

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \tanh \end{pmatrix} W^l \begin{pmatrix} h^{l-1}_t \\ h^l_{t-1} \end{pmatrix}$$

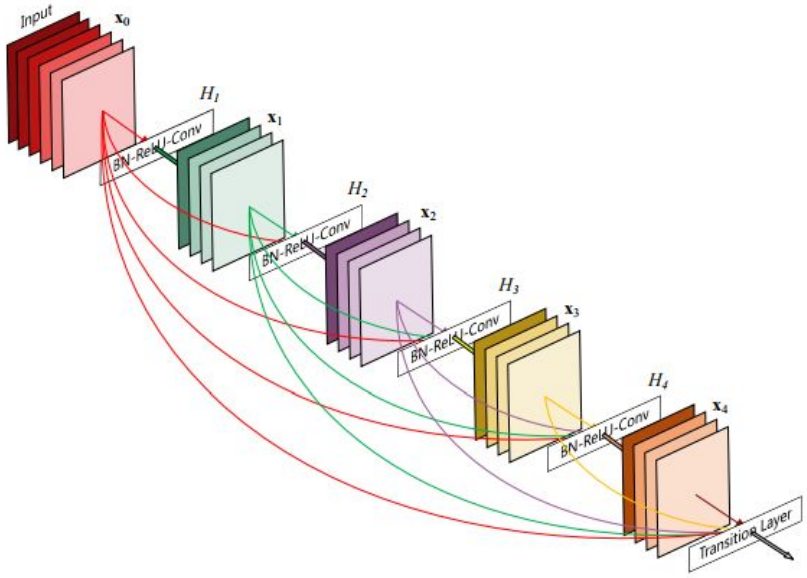
$$c^l_t = f \odot c^l_{t-1} + i \odot g$$
$$h^l_t = o \odot \tanh(c^l_t)$$

Variations: Regularization and Network Architectures

- Adopting Zoneout regularization from []:
  - Apply dropout to *f* gate such that:

$$\mathbf{F} = 1 - \text{dropout}(1 - \sigma(\mathbf{W}_f * \mathbf{X}))$$

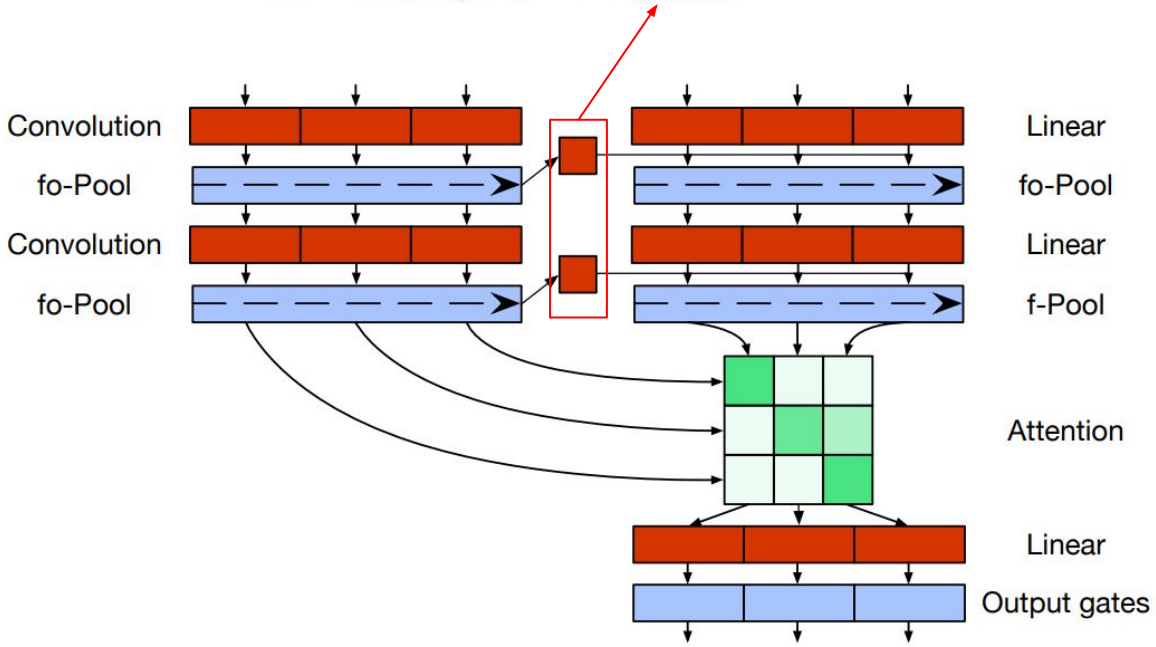
- Densely Connected Networks
  - An extensions to Residual Network
  - Connecting outputs of **every two** layers instead of just one at a time
  - **Concatenate** previous inputs instead of **adding**



Variations: Encoder-Decoder Models

- Encoder-decoder variant uses hidden states of the last encoder layer in the decoder

$$\mathbf{Z}^\ell = \tanh(\mathbf{W}_z^\ell * \mathbf{X}^\ell + \mathbf{V}_z^\ell \tilde{\mathbf{h}}_T^\ell)$$
$$\mathbf{F}^\ell = \sigma(\mathbf{W}_f^\ell * \mathbf{X}^\ell + \mathbf{V}_f^\ell \tilde{\mathbf{h}}_T^\ell)$$
$$\mathbf{O}^\ell = \sigma(\mathbf{W}_o^\ell * \mathbf{X}^\ell + \mathbf{V}_o^\ell \tilde{\mathbf{h}}_T^\ell),$$



Evaluation and Results

- Performance is measured in three tasks
  - Sentiment Analysis
  - Language Modelling
  - Neural Machine Translation
- Overall performs more accurately and trains faster than LSTM with same number of parameters

Experiments: Sentiment Classification

- Sentiment analysis on IMDb movie review dataset
- Better results than equal sized LSTM in shorter time
- L2 and Zoneout regularizations, dense connections

Model	Time / Epoch (s)	Test Acc (%)
NBSVM-bi (Wang & Manning, 2012)	–	91.2
2 layer sequential BoW CNN (Johnson & Zhang, 2014)	–	92.3
Ensemble of RNNs and NB-SVM (Mesnil et al., 2014)	–	92.6
2-layer LSTM (Longpre et al., 2016)	–	87.6
Residual 2-layer bi-LSTM (Longpre et al., 2016)	–	90.1
Our models		
Densely-connected 4-layer LSTM (cuDNN optimized)	480	90.9
Densely-connected 4-layer QRNN	150	91.4
Densely-connected 4-layer QRNN with $k = 4$	160	91.1

Experiments: Sentiment Classification

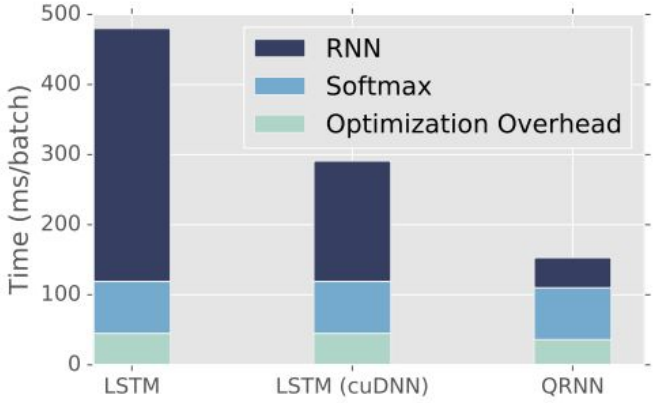
- Sentiment analysis on IMDb movie review dataset
  - L2 and Zoneout regularizations, dense connections
  - 256 untis per layer
  - RMSprop
- Better results than equal sized LSTM in shorter time

Model	**Time / Epoch (s)**	**Test Acc (%)**
NBSVM-bi (Wang & Manning, 2012)	–	91.2
2 layer sequential BoW CNN (Johnson & Zhang, 2014)	–	92.3
Ensemble of RNNs and NB-SVM (Mesnil et al., 2014)	–	92.6
2-layer LSTM (Longpre et al., 2016)	–	87.6
Residual 2-layer bi-LSTM (Longpre et al., 2016)	–	90.1
Our models		
Densely-connected 4-layer LSTM (cuDNN optimized)	480	90.9
Densely-connected 4-layer QRNN	150	91.4
Densely-connected 4-layer QRNN with $k = 4$	160	91.1

Experiments: Sentiment Classification

Timing Comparison to standard LSTM model

- Up to 16x speedup



		Sequence length				
		32	64	128	256	512
Batch size	8	**5.5x**	**8.8x**	**11.0x**	**12.4x**	**16.9x**
	16	**5.5x**	**6.7x**	**7.8x**	**8.3x**	**10.8x**
	32	**4.2x**	**4.5x**	**4.9x**	**4.9x**	**6.4x**
	64	**3.0x**	**3.0x**	**3.0x**	**3.0x**	**3.7x**
	128	**2.1x**	**1.9x**	**2.0x**	**2.0x**	**2.4x**
	256	**1.4x**	**1.4x**	**1.3x**	**1.3x**	**1.3x**

Experiments: Machine Translation

- Encoder-Decoder model is used for German-English translations
  - ~100 characters per sentence
  - 320 units per layer
  - Filter size: 6 chars in the first encoder layer, 2 chars in the next layers
  - Adam optimizer

Model	**Train Time**	**BLEU (TED.tst2014)**
Word-level LSTM w/attn (Ranzato et al., 2016)	–	20.2
Word-level CNN w/attn, input feeding (Wiseman & Rush, 2016)	–	24.0
Our models		
Char-level 4-layer LSTM	4.2 hrs/epoch	16.53
Char-level 4-layer QRNN with $k = 6$	1.0 hrs/epoch	19.41

Conclusions

Pros:

- Proposes a faster and accurate network for sequential learning
- Reduces number of sequentially dependent variables
- Proves the generality by using recent regularization and network design techniques

Cons:

- Seemingly, does not work well without dense connections
- Performs worse than the state-of-the art for neural machine translation

References

- From lecture slides “14 - RNNs”
- James Bradbury, Stephen Merity, Caiming Xiong & Richard Socher Quasi-Recurrent Neural Networks. arXiv: 1611.01576, 2016
- David Balduzzi and Muhammad Ghifary. Strongly-typed recurrent neural networks. In ICML, 2016.
- Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. Neural machine translation in linear time. arXiv preprint arXiv:1610.10099, 2016.
- Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. arXiv preprint arXiv:1601.06759, 2016.