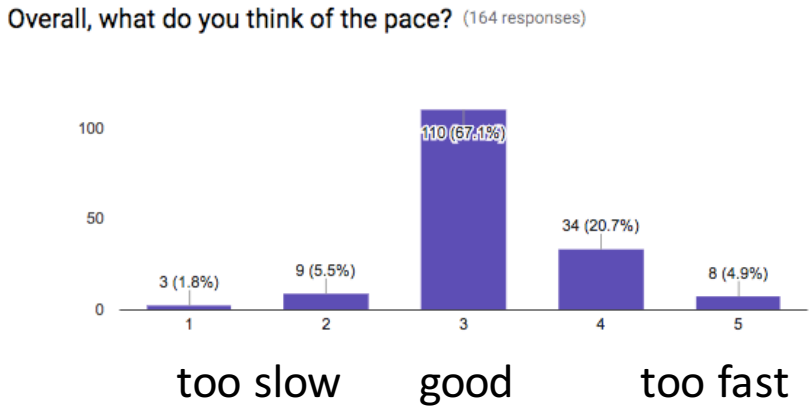
Natural Language Processing with Deep Learning

CS224N/Ling284

Lecture 13:
Convolutional Neural Networks (for NLP)

Christopher Manning and **Richard Socher**

Overview of today



- Organization
- Mini tutorial on Azure and GPUs

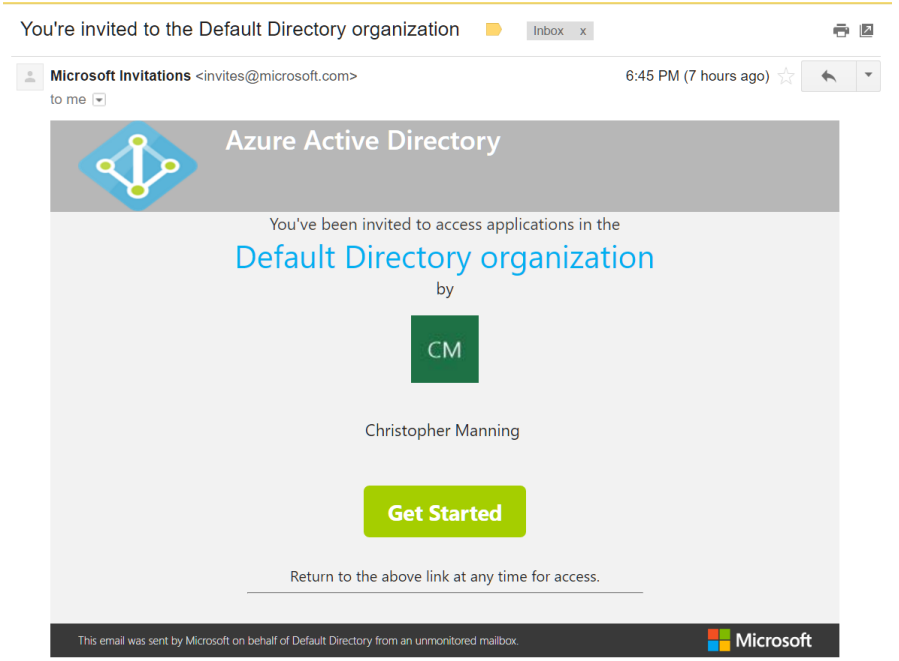
- CNN Variant 1: Simple single layer
- Application: Sentence classification
- Research Highlight: Character-Aware Neural Language Models

- More details and tricks
- Evaluation
- Comparison between sentence models: BoV, RNNs, CNNs
- CNN Variant 2: Complex multi layer

Motivation & Getting Started

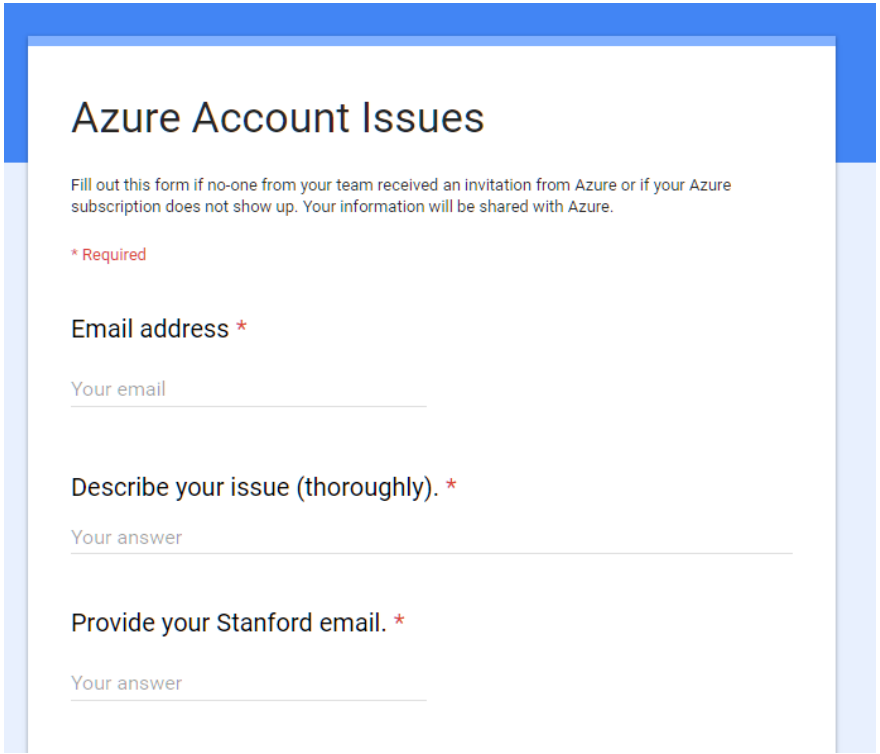
Azure Subscriptions

- Every team should have received an email by now
- Only **38**/311 teams have activated their subscriptions
- Piazza @1830.
  - Subscription issues form
  - Azure representative will be in touch with you shortly



Azure Subscriptions

- Every team should have received an email by now
- Only **38**/311 teams have activated their subscriptions
- Piazza @1830.
  - Subscription issues form
  - Azure representative will be in touch with you shortly



Azure Account Issues

Fill out this form if no-one from your team received an invitation from Azure or if your Azure subscription does not show up. Your information will be shared with Azure.

* Required

Email address *

Your email

Describe your issue (thoroughly). *

Your answer

Provide your Stanford email. *

Your answer

Why should you care?

- We announced a milestone for the final-project/HW4
  - **Worth 2% of your final grade!**
  - Due Sunday (3/5), 11:59PM (!)
  - Expected to have done something with GPU by then!
  - "One paragraph of what you have done"

- Better reason:

GPUs will be much faster to train large models over large data sets

Why should you care?

- Microsoft has offered us NV6 instances (using NVIDIA Tesla M60)
- Specifications:
  - 2048 CUDA cores
  - 8 GB graphics memory
  - **These cost over $3,000 each!**
- Also: *6 CPU cores and 56 GB system memory*

Why should you care?

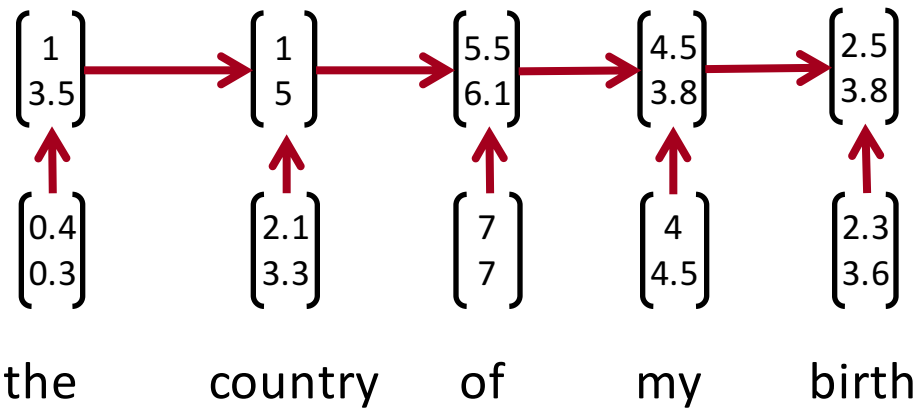
10-20x more powerful than your average CPU, laptop GPU, or desktop GPU for Deep Learning!

Conclusions

- Please do get started on Azure
  - Fill out the help form if you run into subscription issues
  - Come to office hours or file support tickets if you run into technical issues
  - See our Azure-Step-by-Step-Guide
- Homework 4
  - **Decent models will take 1+ hours per epoch on a beefy GPU**
  - Using only a CPU will take a week for several epochs
- Final project
  - Several posts on Piazza already about large datasets
  - If your project is sufficiently challenging for a good final project, you probably want to use a GPU as well.

From RNNs to CNNs

- Recurrent neural nets cannot capture phrases without prefix context
- Often capture too much of last words in final vector



- Softmax is often only at the last step

From RNNs to CNNs

- Main CNN idea:
- What if we compute vectors for every possible phrase?

- Example: “the country of my birth” computes vectors for:
  - the country, country of, of my, my birth, the country of, country of my, of my birth, the country of my, country of my birth

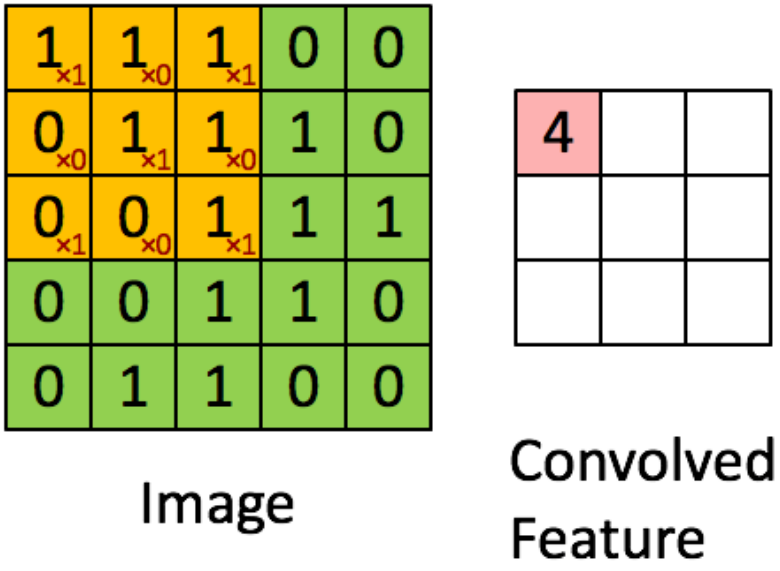
- Regardless of whether phrase is grammatical
- Not very linguistically or cognitively plausible

- Then group them afterwards (more soon)

What is convolution anyway?

- 1d discrete convolution generally: $(f * g)[n] = \sum_{m=-M}^{M} f[n-m]g[m].$
- Convolution is great to extract features from images

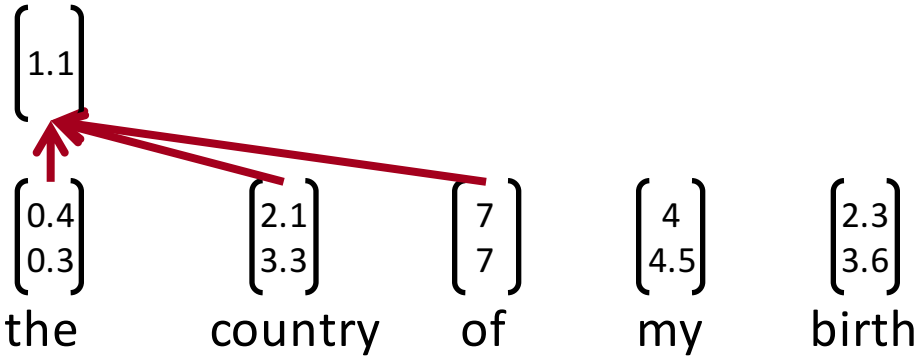
- 2d example →
- Yellow and red numbers show filter weights
- Green shows input



Stanford UFLDL wiki

Single Layer CNN

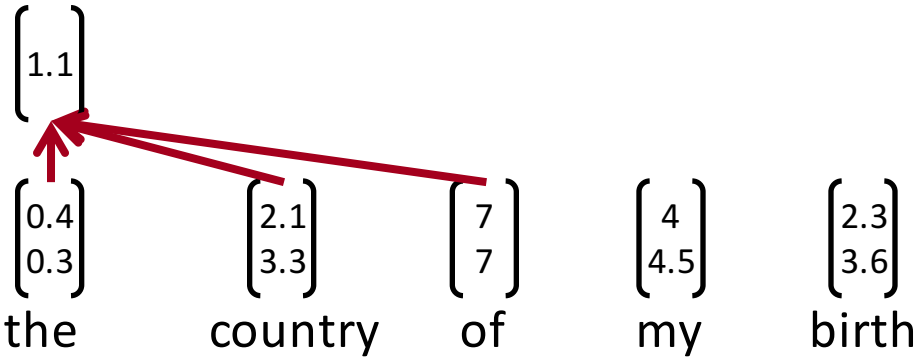
- A simple variant using one convolutional layer and **pooling**
- Based on Collobert and Weston (2011) and Kim (2014) "Convolutional Neural Networks for Sentence Classification"
- Word vectors: $\mathbf{x}_i \in \mathbb{R}^k$
- Sentence: $\mathbf{x}_{1:n} = \mathbf{x}_1 \oplus \mathbf{x}_2 \oplus \ldots \oplus \mathbf{x}_n$ (vectors concatenated)
- Concatenation of words in range: $\mathbf{x}_{i:i+j}$
- Convolutional filter: $\mathbf{w} \in \mathbb{R}^{hk}$ (goes over window of h words)
- Could be 2 (as before) higher, e.g. 3:



Single layer CNN

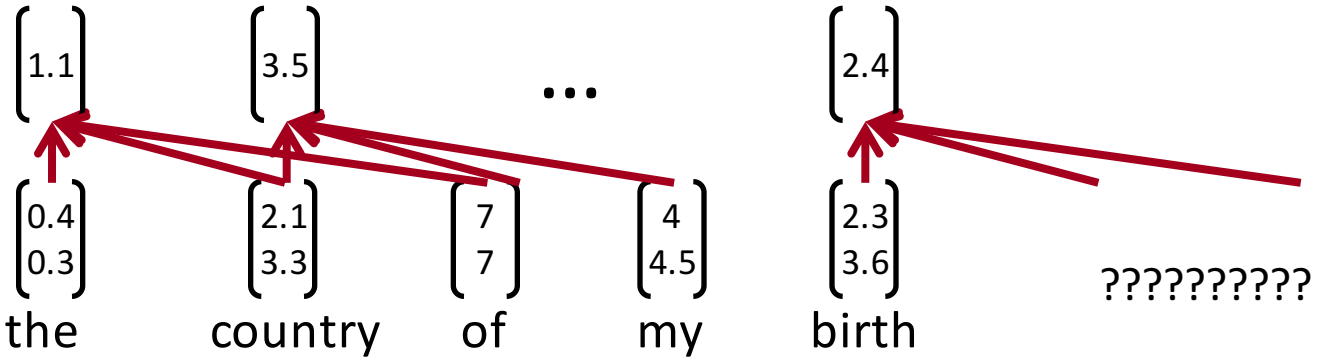
- Convolutional filter: $\mathbf{w} \in \mathbb{R}^{hk}$ (goes over window of h words)
- Note, filter is vector!
- Window size h could be 2 (as before) or higher, e.g. 3:
- To compute feature for CNN layer:

$$c_i = f(\mathbf{w}^T \mathbf{x}_{i:i+h-1} + b)$$



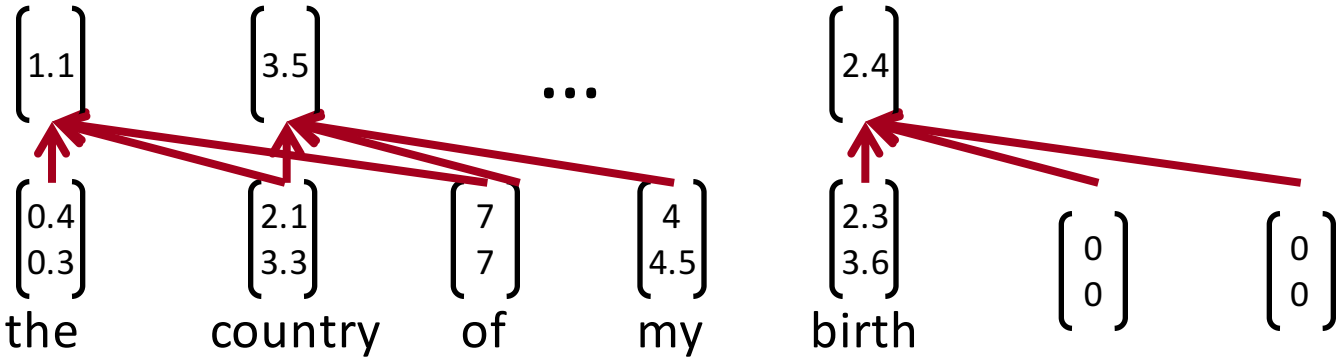
Single layer CNN

- Filter w is applied to all possible windows (concatenated vectors)
- Sentence: $\mathbf{x}_{1:n} = \mathbf{x}_1 \oplus \mathbf{x}_2 \oplus \ldots \oplus \mathbf{x}_n$
- All possible windows of length h: $\{\mathbf{x}_{1:h}, \mathbf{x}_{2:h+1}, \ldots, \mathbf{x}_{n-h+1:n}\}$
- Result is a feature map: $\mathbf{c} = [c_1, c_2, \ldots, c_{n-h+1}] \in \mathbb{R}^{n-h+1}$



Single layer CNN

- Filter w is applied to all possible windows (concatenated vectors)
- Sentence: $\mathbf{x}_{1:n} = \mathbf{x}_1 \oplus \mathbf{x}_2 \oplus \ldots \oplus \mathbf{x}_n$
- All possible windows of length h: $\{\mathbf{x}_{1:h}, \mathbf{x}_{2:h+1}, \ldots, \mathbf{x}_{n-h+1:n}\}$
- Result is a feature map: $\mathbf{c} = [c_1, c_2, \ldots, c_{n-h+1}] \in \mathbb{R}^{n-h+1}$



Single layer CNN: Pooling layer

- New building block: Pooling
- In particular: max-over-time pooling layer
- Idea: capture most important activation (maximum over time)

- From feature map $\mathbf{c} = [c_1, c_2, \ldots, c_{n-h+1}] \in \mathbb{R}^{n-h+1}$

- Pooled single number: $\hat{c} = \max\{\mathbf{c}\}$

- But we want more features!

Solution: Multiple filters

- Use multiple filter weights w
- Useful to have different window sizes h
- Because of max pooling $\hat{c} = \max\{\mathbf{c}\}$, length of **c** irrelevant

$$\mathbf{c} = [c_1, c_2, \ldots, c_{n-h+1}] \in \mathbb{R}^{n-h+1}$$

- So we can have some filters that look at unigrams, bigrams, tri-grams, 4-grams, etc.

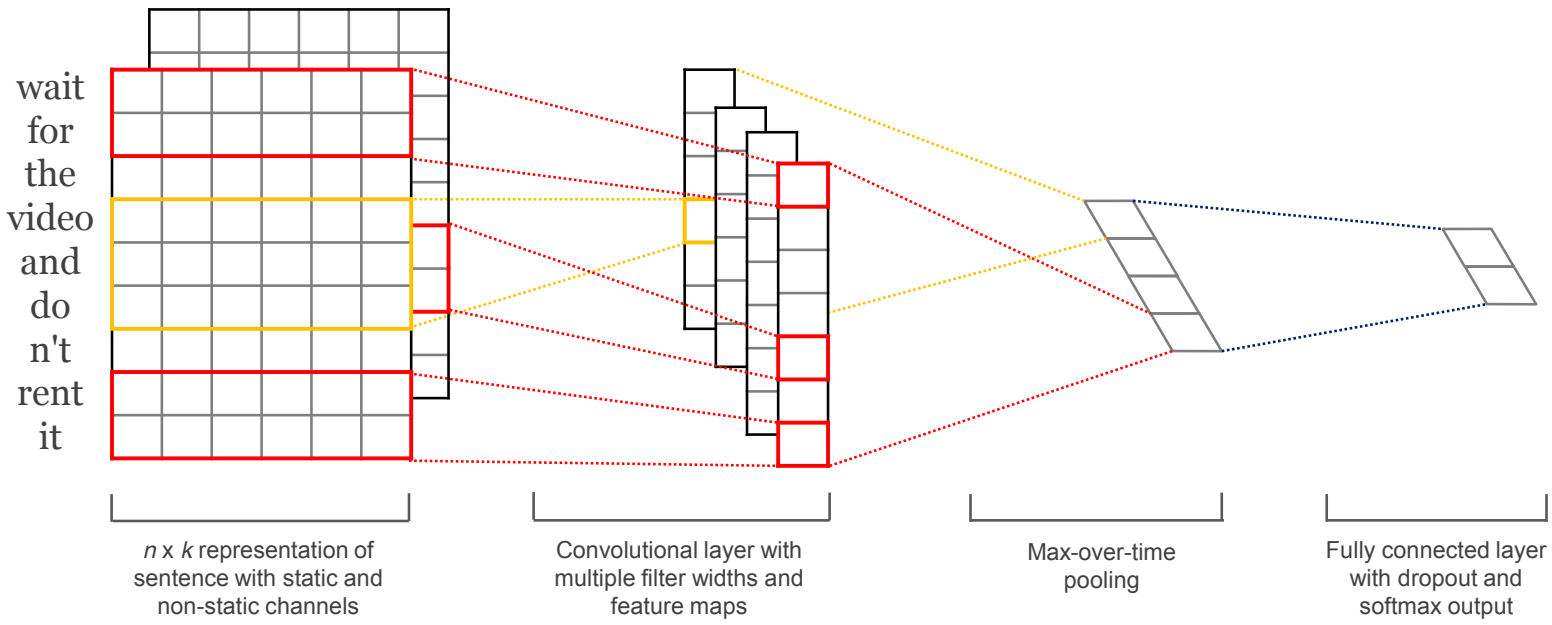
Multi-channel idea

- Initialize with pre-trained word vectors (word2vec or Glove)
- Start with two copies
- Backprop into only one set, keep other “static”
- Both channels are added to c_i before max-pooling

Classification after one CNN layer

- First one convolution, followed by one max-pooling
- To obtain final feature vector: $\mathbf{z} = [\hat{c}_1, \ldots, \hat{c}_m]$ (assuming m filters w)
- Simple final softmax layer $y = softmax\left(W^{(S)} z + b\right)$

Figure from Kim (2014)



n words (possibly zero padded) and each word vector has k dimensions

Character-Aware Neural Language Models

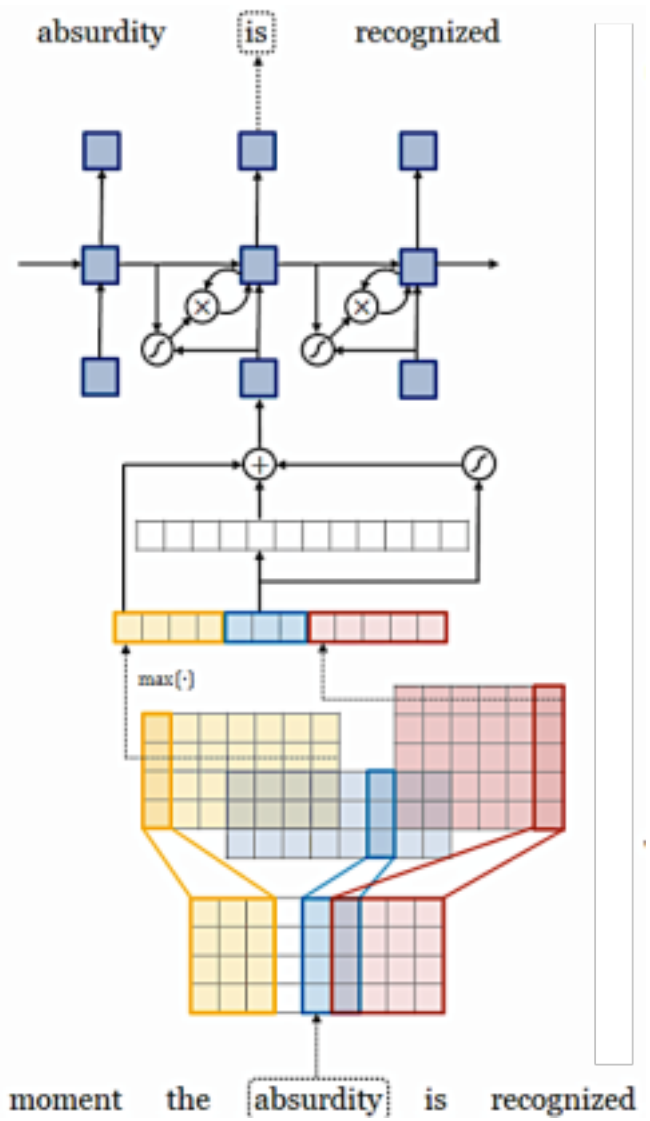
Yoon Kim, Yacine Jernite, David Sontag, Alexander M. Rush

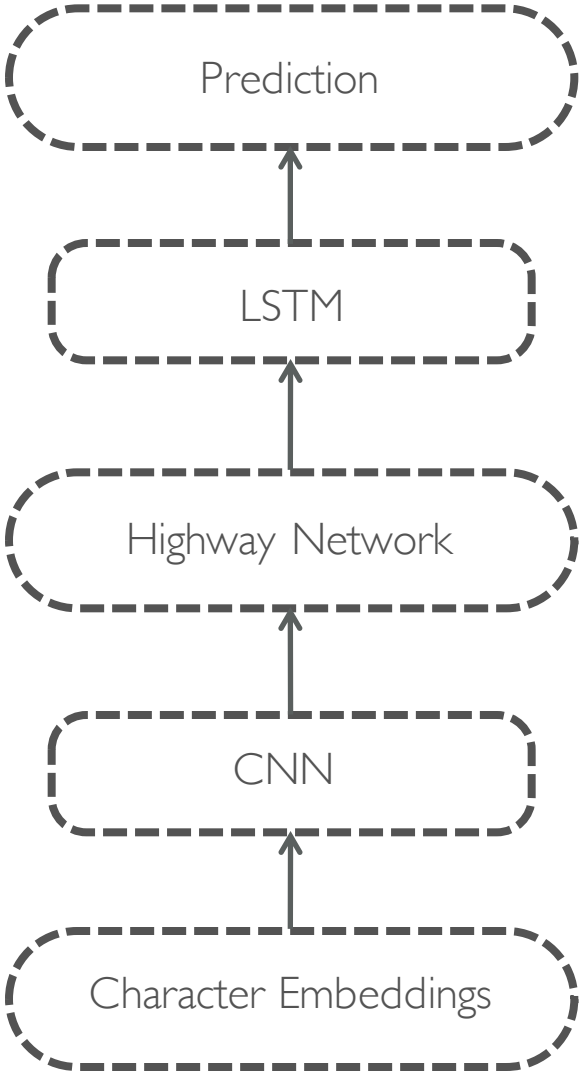
Amani V. Peddada
amanivp@cs.stanford.edu

Motivation

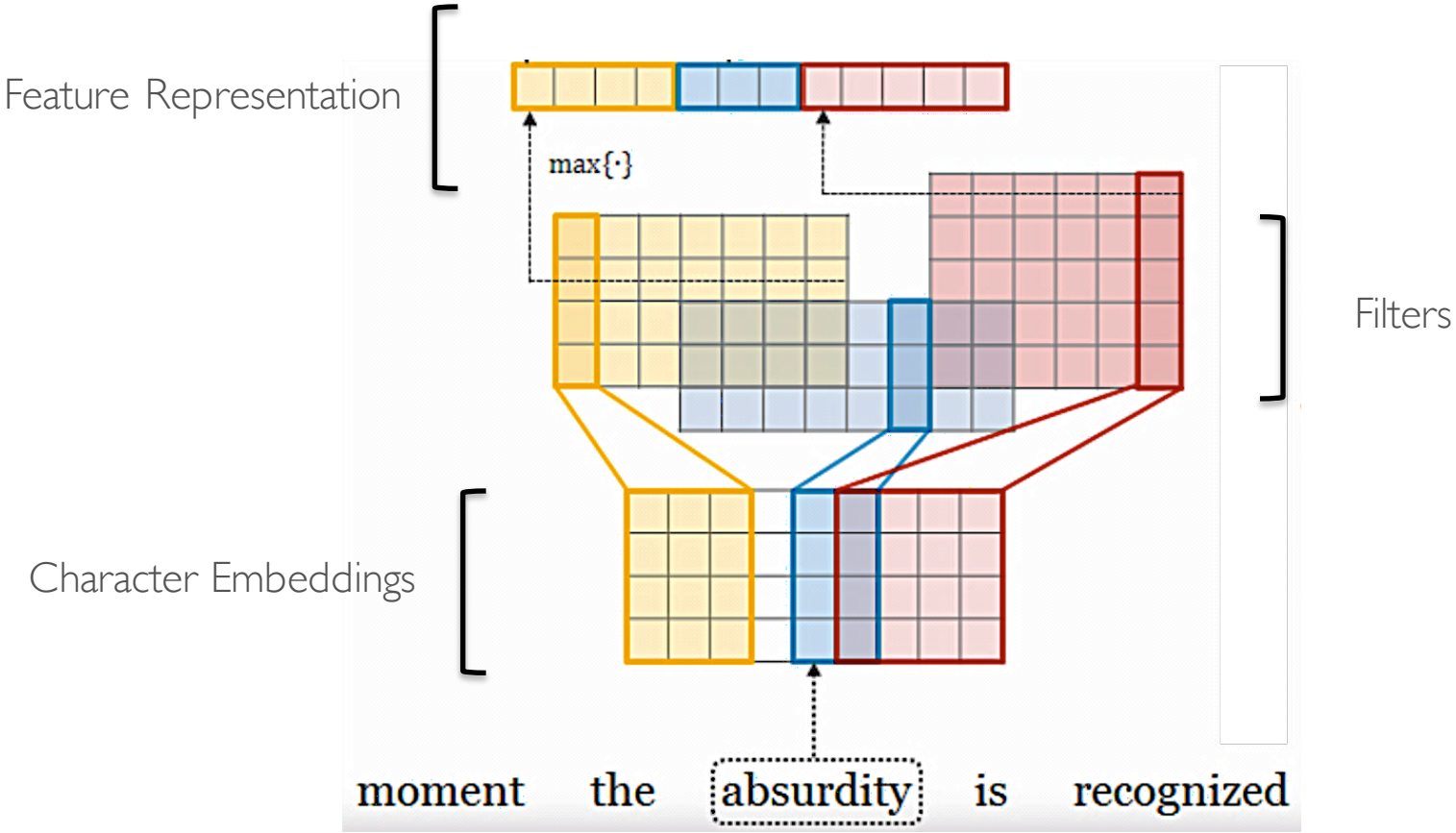
- Derive a powerful, robust language model effective across a variety of languages.
- Encode subword relatedness: *eventful, eventfully, uneventful*…
- Address rare-word problem of prior models.
- Obtain comparable expressivity with fewer parameters.

Technical Approach





Convolutional Layer



- Convolutions over character-level inputs.
- Max-over-time pooling (effectively n-gram selection).

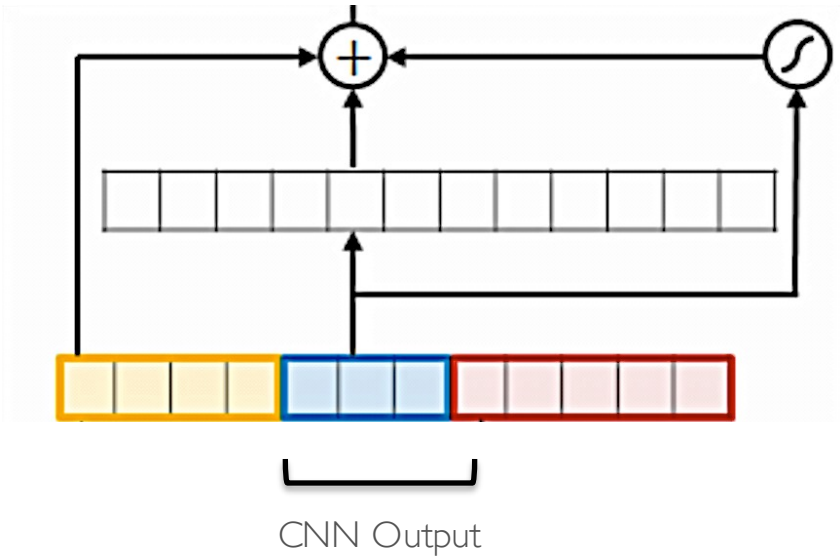
Highway Network (Srivastava et al. 2015)

- Model *n*-gram interactions.
- Apply transformation while carrying over original information.
- Functions akin to an LSTM memory cell.

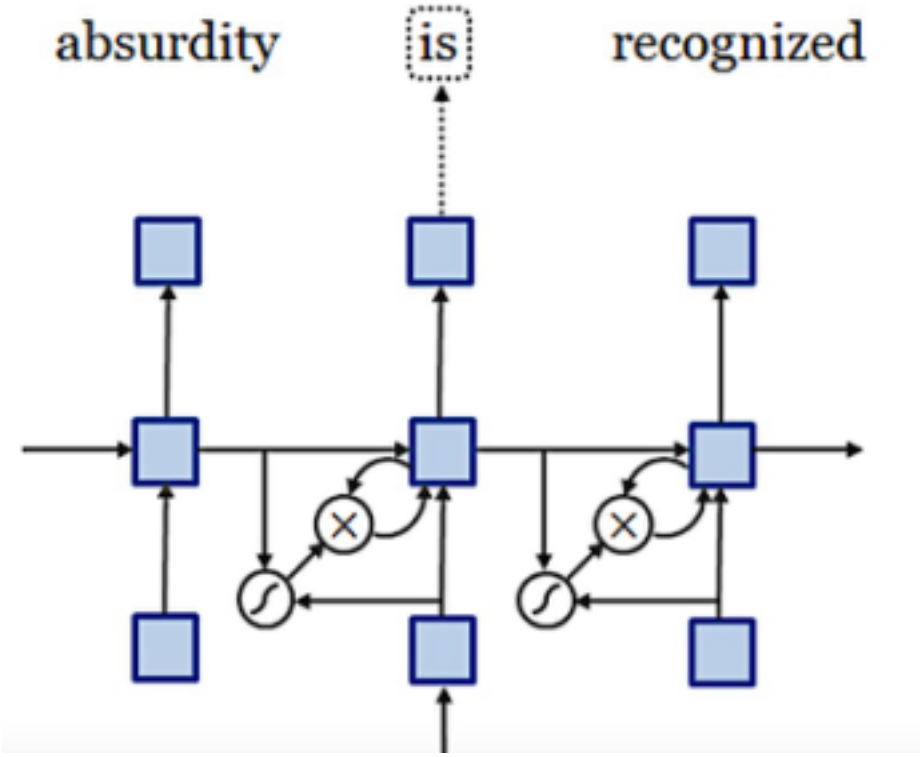
$$\mathbf{t} = \sigma(\mathbf{W}_T\mathbf{y} + \mathbf{b}_T)$$

$$\mathbf{z} = \mathbf{t} \odot g(\mathbf{W}_H\mathbf{y} + \mathbf{b}_H) + (\mathbf{1} - \mathbf{t}) \odot \mathbf{y}$$

Transform Gate Input Carry Gate



Long Short-Term Memory Network



Highway Network Output

- Hierarchical Softmax to handle large output vocabulary.
- Trained with truncated backprop through time.

Quantitative Results

		DATA-S					
		CS	DE	ES	FR	RU	AR
Botha	KN-4	545	366	241	274	396	323
	MLBL	465	296	200	225	304	–
Small	Word	503	305	212	229	352	216
	Morph	414	278	197	216	290	230
	Char	401	260	182	189	278	196
Large	Word	493	286	200	222	357	172
	Morph	398	263	177	196	271	**148**
	Char	**371**	**239**	**165**	**184**	**261**	**148**

		DATA-L					
		CS	DE	ES	FR	RU	EN
Botha	KN-4	862	463	219	243	390	291
	MLBL	643	404	203	227	**300**	273
Small	Word	701	347	186	202	353	236
	Morph	615	331	189	209	331	233
	Char	**578**	**305**	**169**	**190**	313	**216**

Comparable performance with fewer parameters!

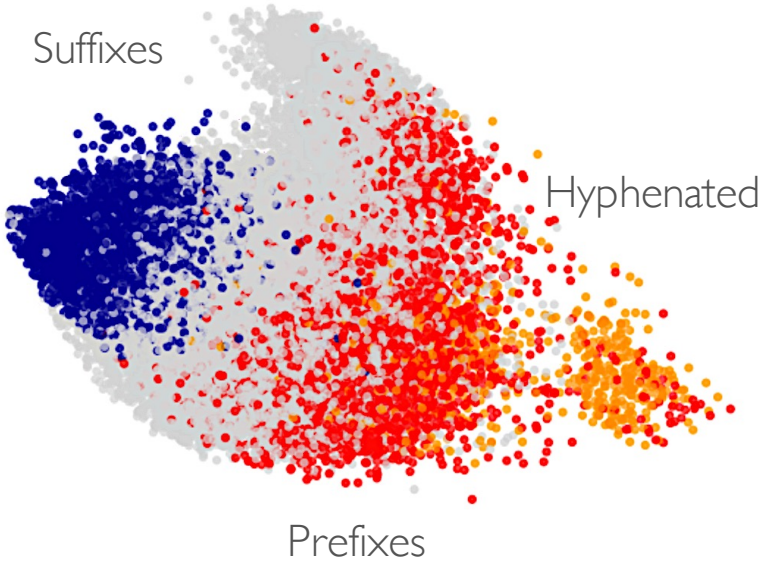
	PPL	Size
LSTM-Word-Small	97.6	5 m
LSTM-Char-Small	92.3	5 m
LSTM-Word-Large	85.4	20 m
→ LSTM-Char-Large	78.9	19 m
KN-5 (Mikolov et al. 2012)	141.2	2 m
RNN† (Mikolov et al. 2012)	124.7	6 m
RNN-LDA† (Mikolov et al. 2012)	113.7	7 m
genCNN† (Wang et al. 2015)	116.4	8 m
FOFE-FNNLM† (Zhang et al. 2015)	108.0	6 m
Deep RNN (Pascanu et al. 2013)	107.5	6 m
Sum-Prod Net† (Cheng et al. 2014)	100.0	5 m
LSTM-1† (Zaremba et al. 2014)	82.7	20 m
→ LSTM-2† (Zaremba et al. 2014)	78.4	52 m

Qualitative Insights

	In Vocabulary				
	while	*his*	*you*	*richard*	*trading*
LSTM-Word	*although*	*your*	*conservatives*	*jonathan*	*advertised*
	letting	*her*	*we*	*robert*	*advertising*
	though	*my*	*guys*	*neil*	*turnover*
	minute	*their*	*i*	*nancy*	*turnover*
LSTM-Char (before highway)	*chile*	*this*	*your*	*hard*	*heading*
	whole	*hhs*	*young*	*rich*	*training*
	meanwhile	*is*	*four*	*richer*	*reading*
	white	*has*	*youth*	*richter*	*leading*
LSTM-Char (after highway)	*meanwhile*	*hhs*	*we*	*eduard*	*trade*
	whole	*this*	*your*	*gerard*	*training*
	though	*their*	*doug*	*edward*	*traded*
	nevertheless	*your*	*i*	*carl*	*trader*

Qualitative Insights

Out-of-Vocabulary		
computer-aided	*misinformed*	*looooook*
–	–	–
–	–	–
–	–	–
–	–	–
computer-guided	*informed*	*look*
computerized	*performed*	*cook*
disk-drive	*transformed*	*looks*
computer	*inform*	*shook*
computer-guided	*informed*	*look*
computer-driven	*performed*	*looks*
computerized	*outperformed*	*looked*
computer	*transformed*	*looking*



Conclusion

- Questions the necessity of using word embeddings as inputs for neural language modeling.
- CNNs + Highway Network over characters can extract rich semantic and structural information.
- Key takeaway: can compose "building blocks" to obtain more nuanced or powerful models!

Tricks to make it work better: Dropout

- Idea: randomly mask/dropout/set to 0 some of the feature weights z
- Create masking vector r of Bernoulli random variables with probability p (a hyperparameter) of being 1

- Delete features during training:

$$y = softmax\left(W^{(S)}(r \circ z) + b\right)$$

- Reasoning: Prevents co-adaptation (overfitting to seeing specific feature constellations)

Tricks to make it work better: Dropout

$$y = softmax\left(W^{(S)}(r \circ z) + b\right)$$

- At training time, gradients are backpropagated only through those elements of z vector for which $r_i = 1$

- At test time, there is no dropout, so feature vectors z are larger.
- Hence, we scale final vector by Bernoulli probability p

$$\hat{W}^{(S)} = pW^{(S)}$$

- Kim (2014) reports **2 – 4% improved accuracy** and ability to use very large networks without overfitting

Another regularization trick

- Constrain l_2 norms of weight vectors of each class (row in softmax weight $W^{(S)}$) to fixed number s (also a hyperparameter)

- If $\|W_{c\cdot}^{(S)}\| > s$, then rescale it so that: $\|W_{c\cdot}^{(S)}\| = s$

- Not very common

All hyperparameters in Kim (2014)

- Find hyperparameters based on dev set
- Nonlinearity: reLu
- Window filter sizes h = 3,4,5
- Each filter size has 100 feature maps
- Dropout p = 0.5
- L2 constraint s for rows of softmax s = 3
- Mini batch size for SGD training: 50
- Word vectors: pre-trained with word2vec, k = 300

- During training, keep checking performance on dev set and pick highest accuracy weights for final evaluation

Experiments

Model	MR	SST-1	SST-2	Subj	TREC	CR	MPQA
CNN-rand	76.1	45.0	82.7	89.6	91.2	79.8	83.4
CNN-static	81.0	45.5	86.8	93.0	92.8	84.7	**89.6**
CNN-non-static	**81.5**	48.0	87.2	93.4	93.6	84.3	89.5
CNN-multichannel	81.1	47.4	**88.1**	93.2	92.2	**85.0**	89.4
RAE (Socher et al., 2011)	77.7	43.2	82.4	–	–	–	86.4
MV-RNN (Socher et al., 2012)	79.0	44.4	82.9	–	–	–	–
RNTN (Socher et al., 2013)	–	45.7	85.4	–	–	–	–
DCNN (Kalchbrenner et al., 2014)	–	48.5	86.8	–	93.0	–	–
Paragraph-Vec (Le and Mikolov, 2014)	–	**48.7**	87.8	–	–	–	–
CCAE (Hermann and Blunsom, 2013)	77.8	–	–	–	–	–	87.2
Sent-Parser (Dong et al., 2014)	79.5	–	–	–	–	–	86.3
NBSVM (Wang and Manning, 2012)	79.4	–	–	93.2	–	81.8	86.3
MNB (Wang and Manning, 2012)	79.0	–	–	**93.6**	–	80.0	86.3
G-Dropout (Wang and Manning, 2013)	79.0	–	–	93.4	–	82.1	86.1
F-Dropout (Wang and Manning, 2013)	79.1	–	–	**93.6**	–	81.9	86.3
Tree-CRF (Nakagawa et al., 2010)	77.3	–	–	–	–	81.4	86.1
CRF-PR (Yang and Cardie, 2014)	–	–	–	–	–	82.7	–
SVM_S (Silva et al., 2011)	–	–	–	–	**95.0**	–	–

Problem with comparison?

- Dropout gives 2 – 4 % accuracy improvement
- Several ‘’baselines’’ didn’t use dropout

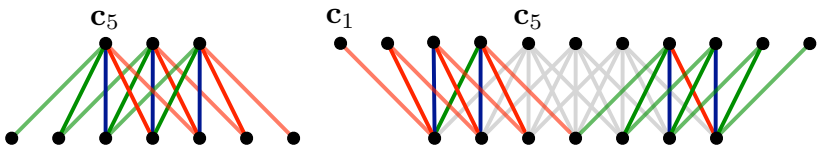
- Still remarkable results and simple architecture!

- Difference to window and RNN architectures we described in previous lectures: pooling, many filters and dropout
- Some of these ideas can be used in RNNs too

- Tree-LSTMs obtain better performance on sentence datasets

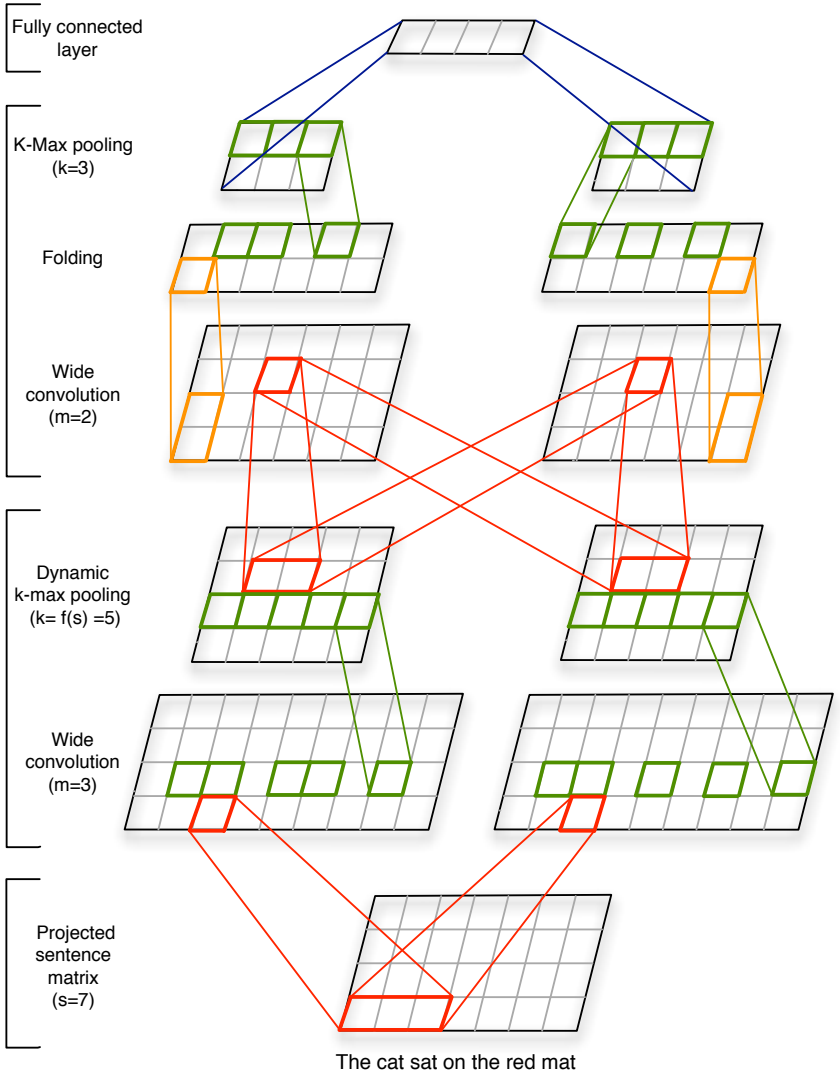
CNN alternatives

- Narrow vs wide convolution



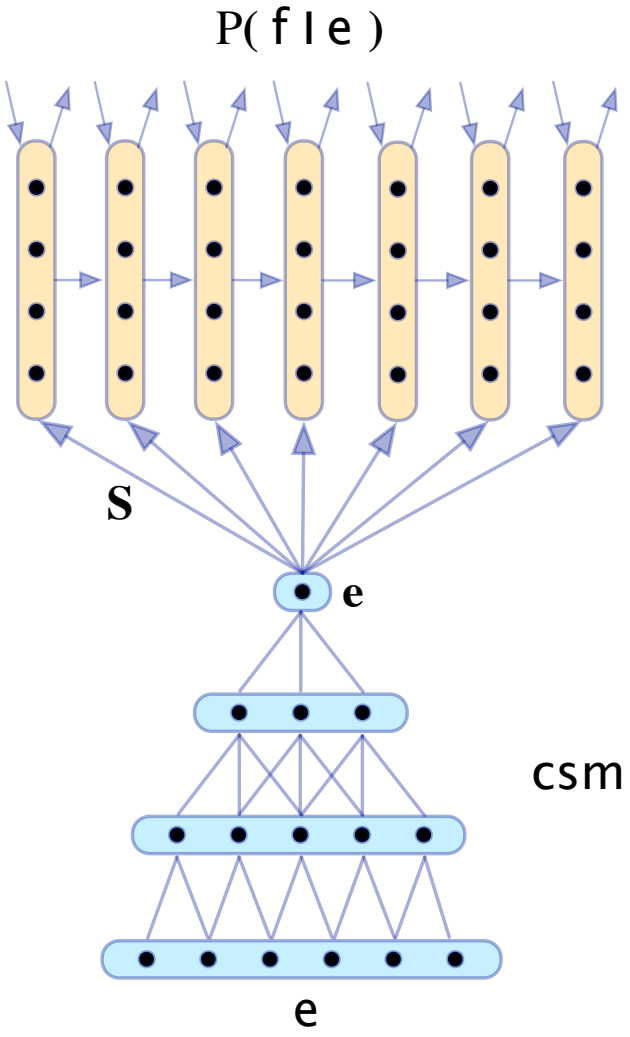
- Complex pooling schemes (over sequences) and deeper convolutional layers

- Kalchbrenner et al. (2014)



CNN application: Translation

- One of the first successful neural machine translation efforts
- Uses CNN for encoding and RNN for decoding
- Kalchbrenner and Blunsom (2013)
  "Recurrent Continuous Translation Models"

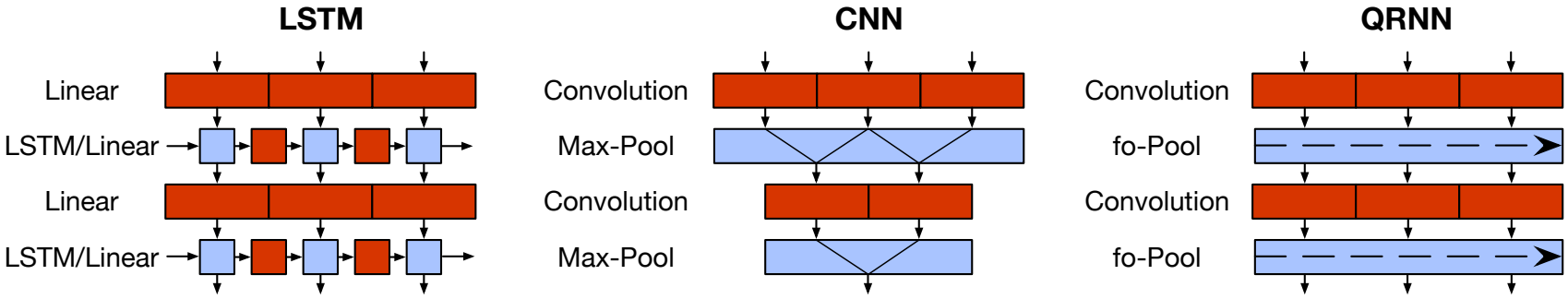


Model comparison

- **Bag of Vectors**: Surprisingly good baseline for simple classification problems. Especially if followed by a few layers!
- **Window Model**: Good for single word classification for problems that do not need wide context
- **CNNs:** good for classification, unclear how to incorporate phrase level annotation (can only take a single label), need zero padding for shorter phrases, hard to interpret, **easy to parallelize on GPUs**
- **Recurrent Neural Networks**: Most cognitively plausible (reading from left to right), not best for classification, slower than CNNs, can do sequence tagging and classification, lots of improvements

Quasi-Recurrent Neural Network

- Combines best of both model families



- Convolutions for parallelism across time:

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

→

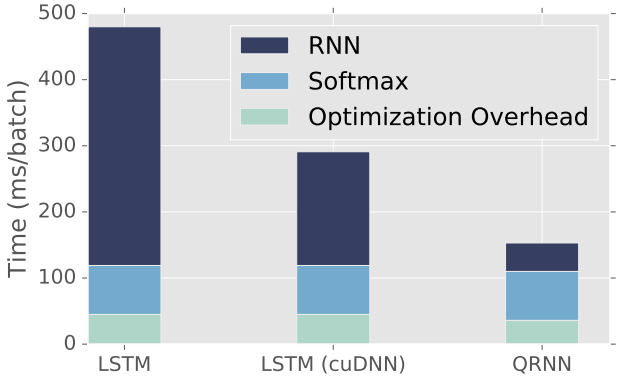
$$\mathbf{Z} = \tanh(\mathbf{W}_z * \mathbf{X})$$
$$\mathbf{F} = \sigma(\mathbf{W}_f * \mathbf{X})$$
$$\mathbf{O} = \sigma(\mathbf{W}_o * \mathbf{X}),$$

- Element-wise gated recurrence for parallelism across channels: $\mathbf{h}_t = \mathbf{f}_t \odot \mathbf{h}_{t-1} + (1 - \mathbf{f}_t) \odot \mathbf{z}_t,$

Q-RNNs for Language Modeling

- Better
- Faster

Model	Parameters	Validation	Test
LSTM (medium) (Zaremba et al., 2014)	20M	86.2	82.7
Variational LSTM (medium) (Gal & Ghahramani, 2016)	20M	81.9	79.7
LSTM with CharCNN embeddings (Kim et al., 2016)	19M	–	78.9
Zoneout + Variational LSTM (medium) (Merity et al., 2016)	20M	84.4	80.6
Our models			
LSTM (medium)	20M	85.7	82.0
QRNN (medium)	18M	82.9	79.9
QRNN + zoneout ($p = 0.1$) (medium)	18M	82.1	78.3



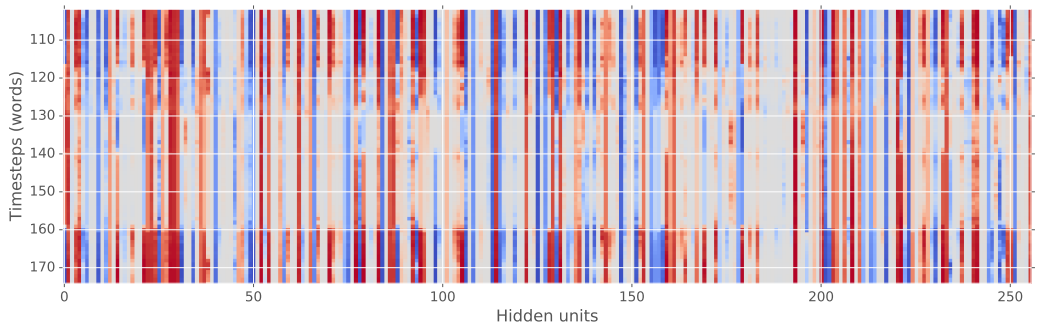
Batch size	Sequence length 32	64	128	256	512
8	**5.5x**	**8.8x**	**11.0x**	**12.4x**	**16.9x**
16	**5.5x**	**6.7x**	**7.8x**	**8.3x**	**10.8x**
32	**4.2x**	**4.5x**	**4.9x**	**4.9x**	**6.4x**
64	**3.0x**	**3.0x**	**3.0x**	**3.0x**	**3.7x**
128	**2.1x**	**1.9x**	**2.0x**	**2.0x**	**2.4x**
256	**1.4x**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

Q-RNNs for Sentiment Analysis

- Better and faster than LSTMs

Model	Time / Epoch (s)	Test Acc (%)
BSVM-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		
Deeply connected 4-layer LSTM (cuDNN optimized)	480	90.9
Deeply connected 4-layer QRNN	150	91.4
D.C. 4-layer QRNN with $k = 4$	160	91.1

- Interpretable



- Example:
- Initial positive review
- *Review starts out positive*
  At 117: "*not exactly a bad story*"
  At 158: "*I recommend this movie to everyone, even if you've never played the game*"
- https://metamind.io/research/new-neural-network-building-block-allows-faster-and-more-accurate-text-understanding/

Next Lecture:

- Last major model: Recursive neural networks

- More parsing and related tasks