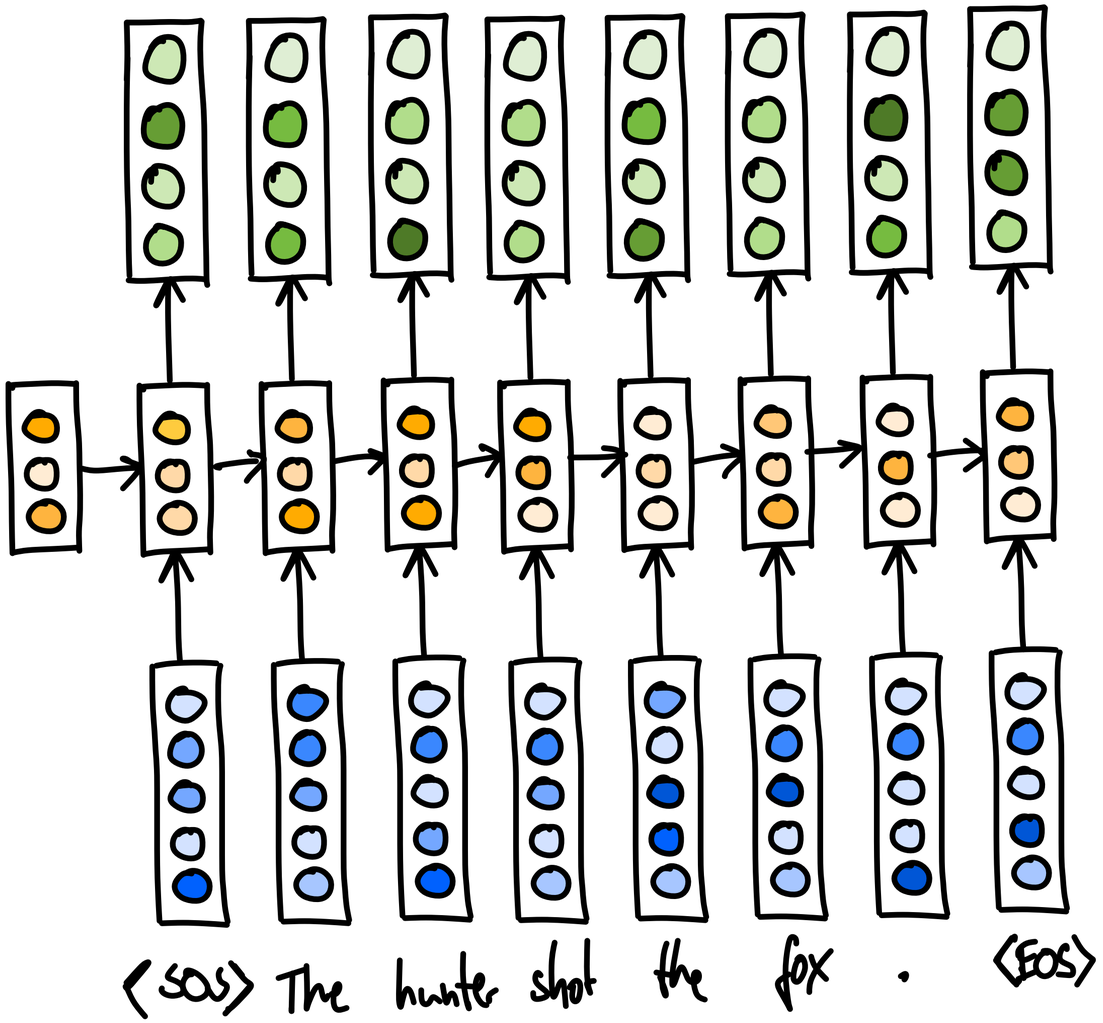
Advanced RNNs

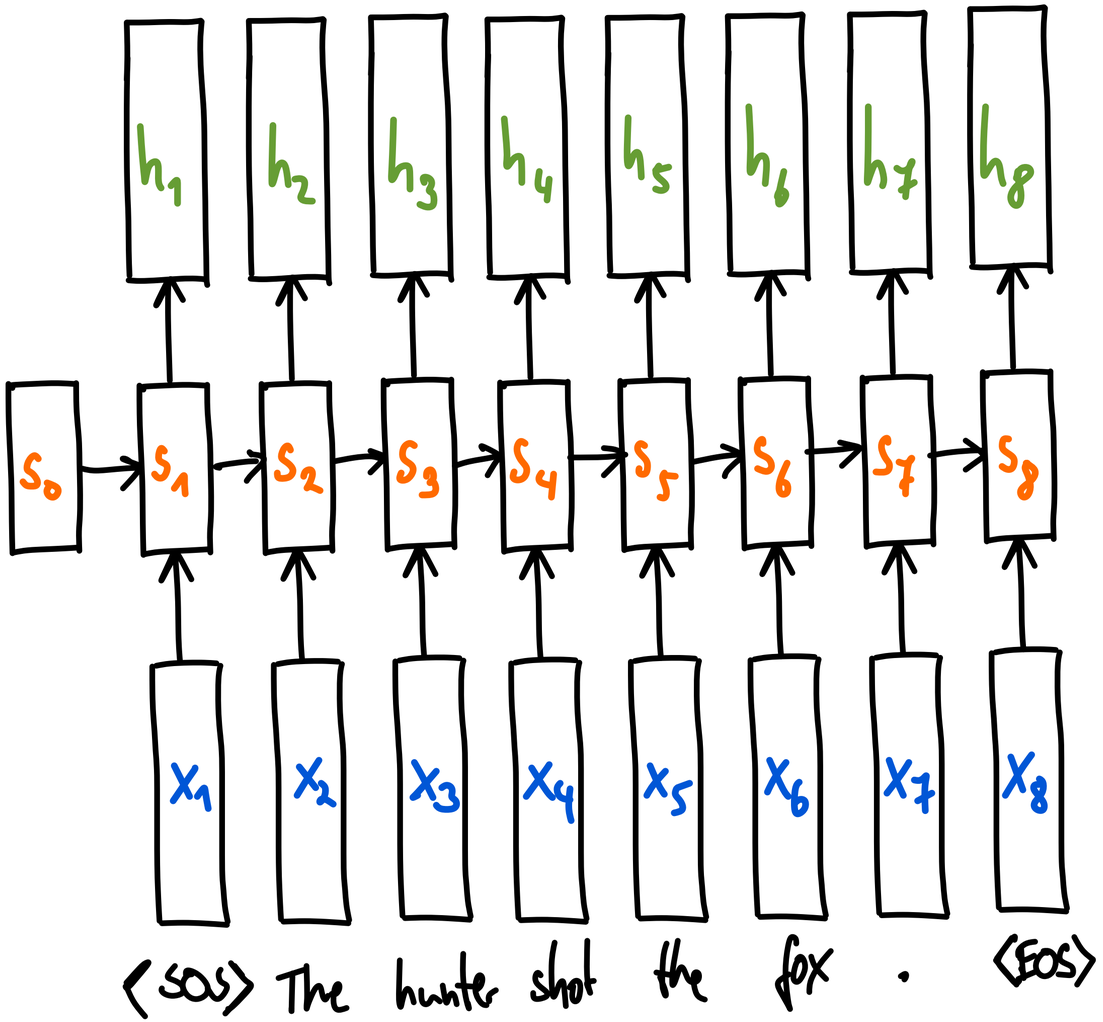
Tim Rocktäschel & Sebastian Riedel

COMP0087 Natural Language Processing

Backpropagation Through Time (BPTT)



Backpropagation Through Time (BPTT)

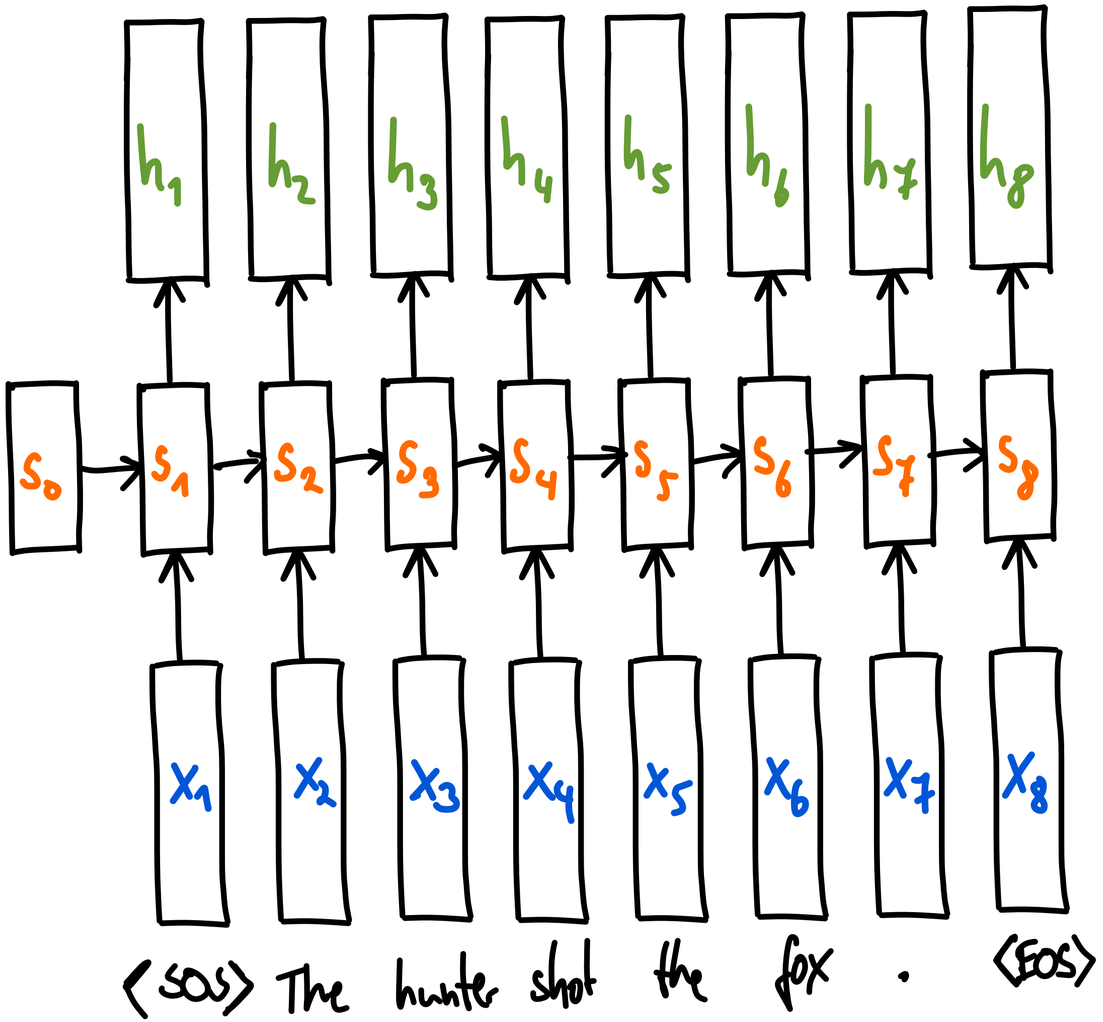


$$\mathbf{s}_t = \tanh(\mathbf{W}[\mathbf{x}_t; \mathbf{s}_{t-1}] + \mathbf{b})$$
$$\mathbf{h}_t = \mathbf{s}_t$$

$$\frac{\partial L(w_N, \hat{w}_N)}{\partial \mathbf{s}_t} = \frac{\partial L(w_N, \hat{w}_N)}{\partial \mathbf{s}_N} \frac{\partial \mathbf{s}_N}{\partial \mathbf{s}_{N-1}} \cdots \frac{\partial \mathbf{s}_{t+1}}{\partial \mathbf{s}_t}$$
$$= \frac{\partial L(w_N, \hat{w}_N)}{\partial \mathbf{s}_N} \prod_{i=t+1}^{N} \frac{\partial \mathbf{s}_i}{\partial \mathbf{s}_{i-1}}$$

$$\frac{\partial L(w, \hat{w})}{\partial \mathbf{s}_t} = \sum_{j=t}^{N} \frac{\partial L(w_j, \hat{w}_j)}{\partial \mathbf{s}_j} \prod_{i=t+1}^{j} \frac{\partial \mathbf{s}_i}{\partial \mathbf{s}_{i-1}}$$

Vanishing and Exploding Gradients



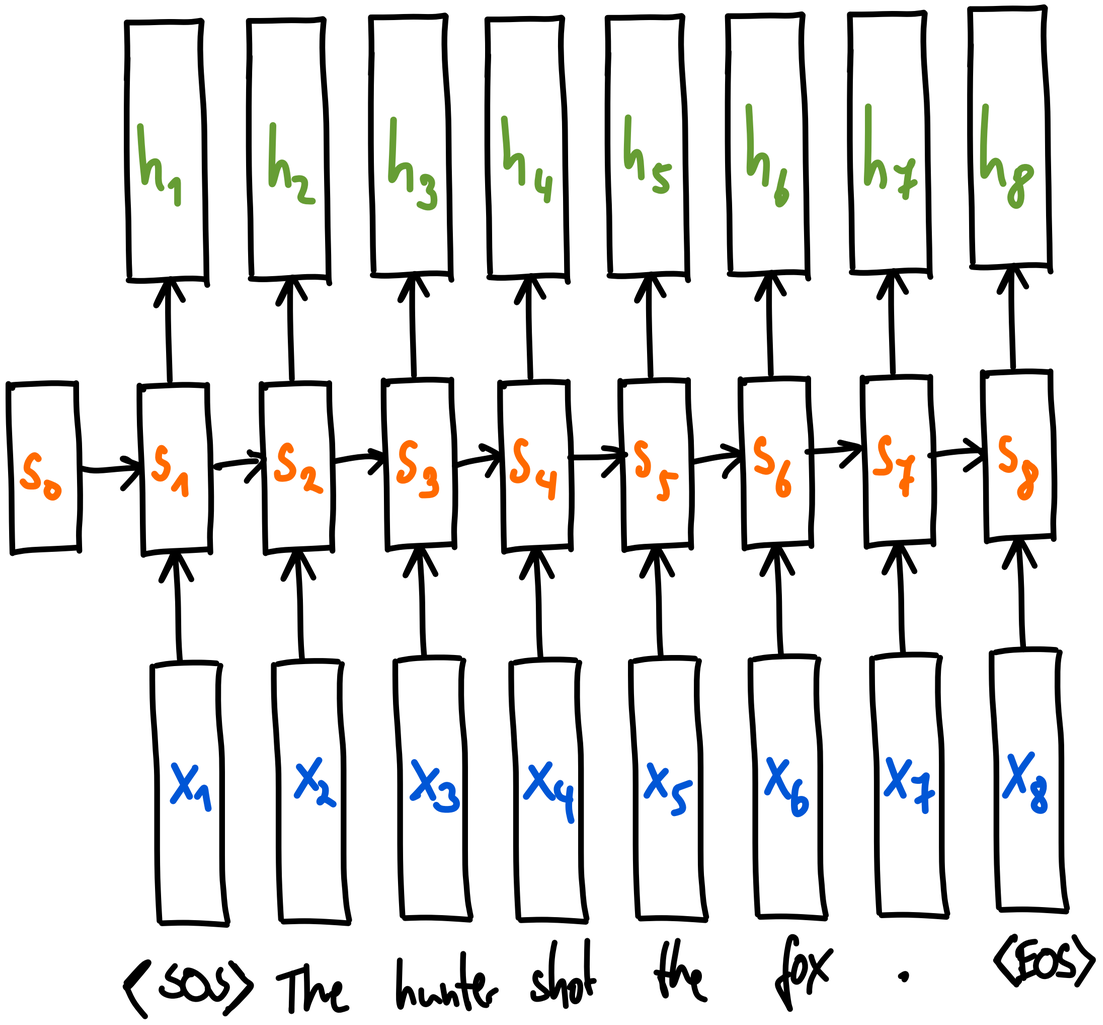
$$\mathbf{u}_t = \mathbf{W}^x \mathbf{x}_t + \mathbf{W}^s \mathbf{s}_{t-1} + \mathbf{b}$$
$$\mathbf{s}_t = \tanh(\mathbf{u}_t)$$
$$\mathbf{h}_t = \mathbf{s}_t$$

$$\frac{\partial L(w_N, \hat{w}_N)}{\partial \mathbf{s}_t} = \frac{\partial L(w_N, \hat{w}_N)}{\partial \mathbf{s}_N} \prod_{i=t+1}^{N} \frac{\partial \mathbf{s}_i}{\partial \mathbf{u}_i} \frac{\partial \mathbf{u}_i}{\partial \mathbf{s}_{i-1}}$$

$$\frac{\partial \mathbf{s}_i}{\partial \mathbf{u}_i} = \tanh'(\mathbf{u}_{i-1})^T, \quad \frac{\partial \mathbf{u}_i}{\partial \mathbf{s}_{i-1}} = \mathbf{W}^s$$

$$\frac{\partial L(w_N, \hat{w}_N)}{\partial \mathbf{s}_t} = \frac{\partial L(w_N, \hat{w}_N)}{\partial \mathbf{s}_N} \prod_{i=t+1}^{N} \tanh'(\mathbf{u}_{i-1})^T \mathbf{W}^s$$

Vanishing and Exploding Gradients



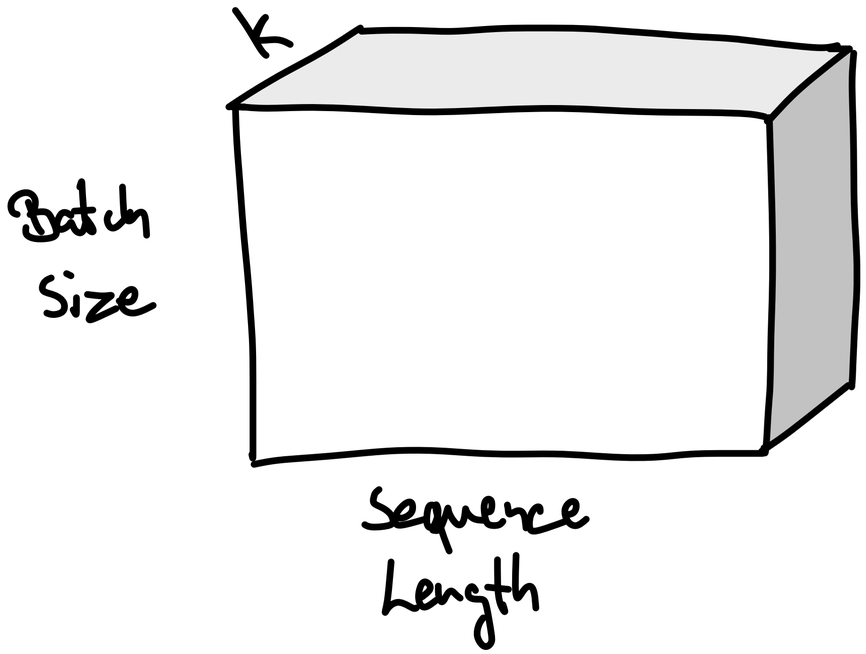
$$\mathbf{u}_t = \mathbf{W}^x \mathbf{x}_t + \mathbf{W}^s \mathbf{s}_{t-1} + \mathbf{b}$$
$$\mathbf{s}_t = \tanh(\mathbf{u}_t)$$
$$\mathbf{h}_t = \mathbf{s}_t$$

$$\frac{\partial L(w_N, \hat{w}_N)}{\partial \mathbf{s}_t} = \frac{\partial L(w_N, \hat{w}_N)}{\partial \mathbf{s}_N} \prod_{i=t+1}^{N} \tanh'(\mathbf{u}_{i-1})^T \mathbf{W}^s$$

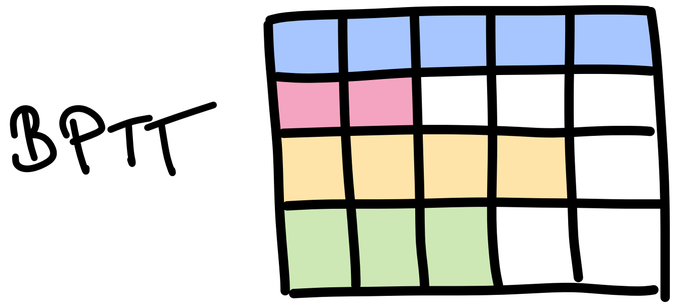
Note the repeated multiplication of $\mathbf{W}^s$
Let λ be the largest eigenvalue of $\mathbf{W}^s$

- **Gradient explodes if λ > 1**
- **Gradient vanishes if λ < 1**
- **Gradient is stable if λ = 1**

Batch Processing

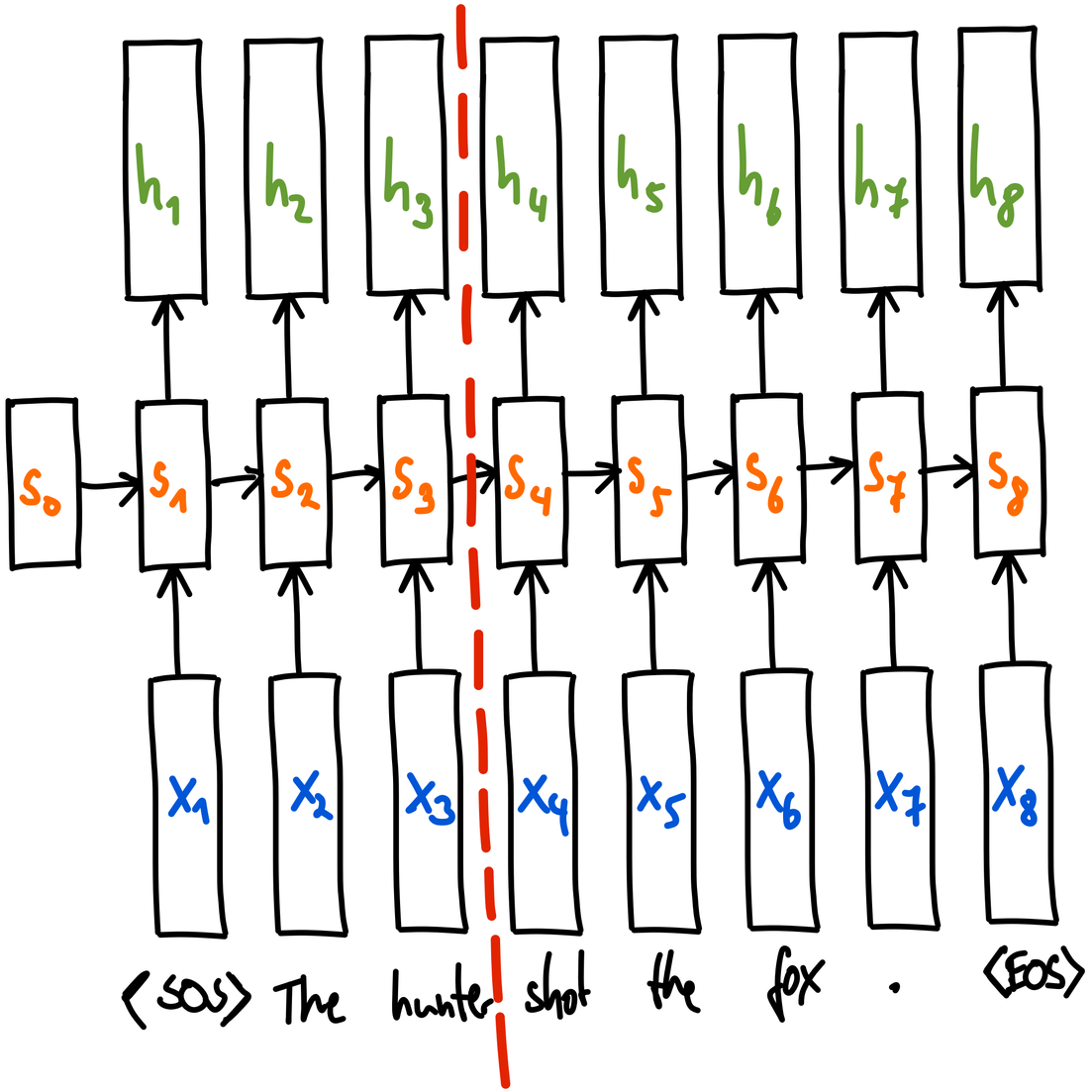


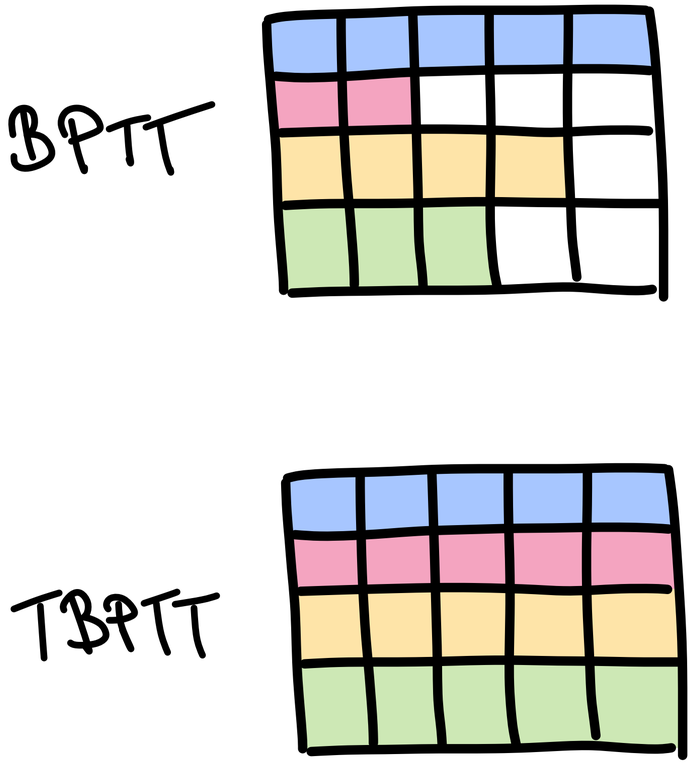
$$\mathbf{S}_t = \tanh(\mathbf{X}_t \mathbf{W}_x^T + \mathbf{S}_{t-1} \mathbf{W}_s^T + \mathbf{b})$$



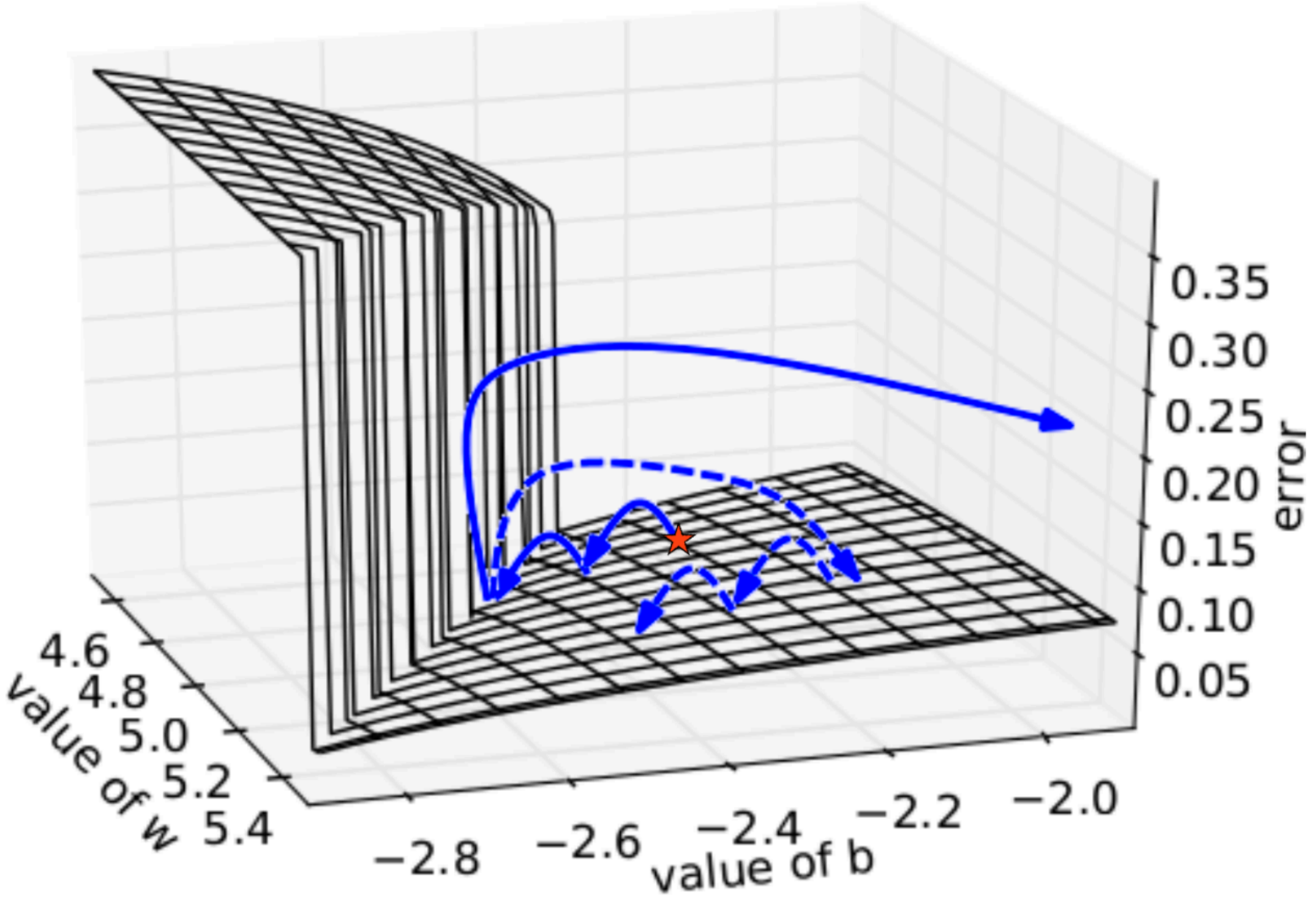

```
def f(self, X, S):
  X = torch.cat([X, S])  # -- [batch_size x 2*k]
  S = torch.tanh(torch.einsum("ij,kj->ki", [self.W, x]) + self.b)  # -- [batch_size x k]
  H = S
  return H, S
```

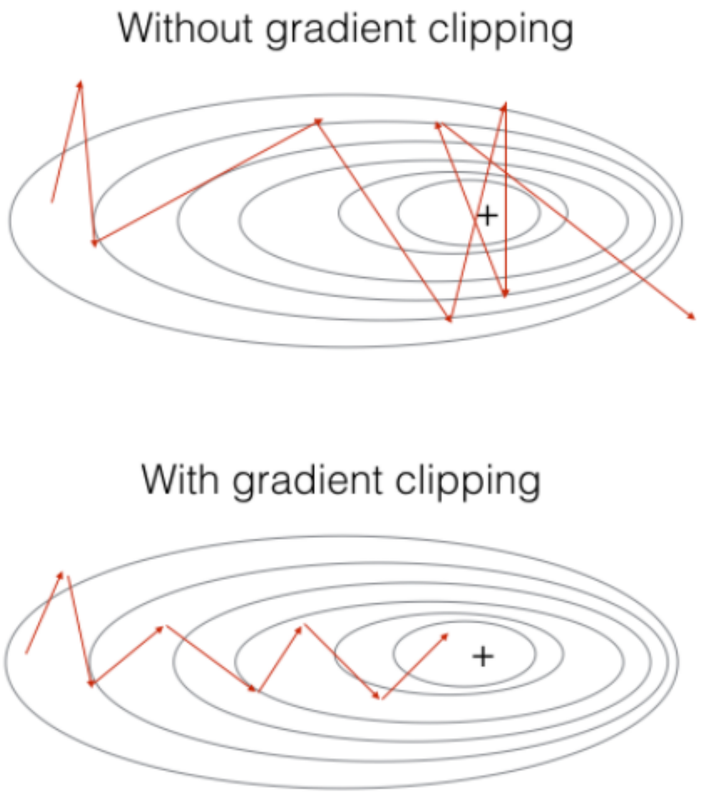
Truncated BPTT





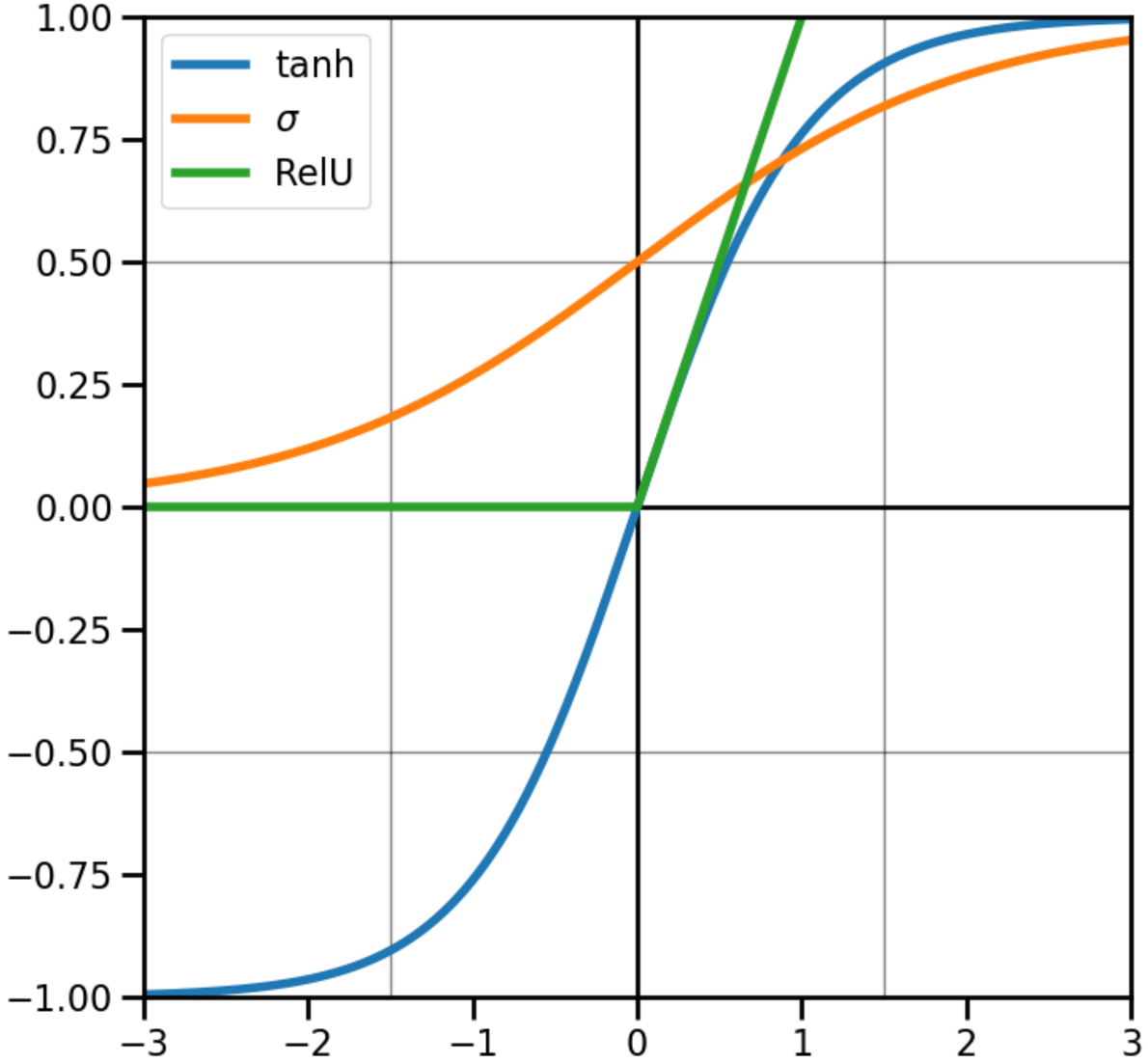
Gradient Clipping





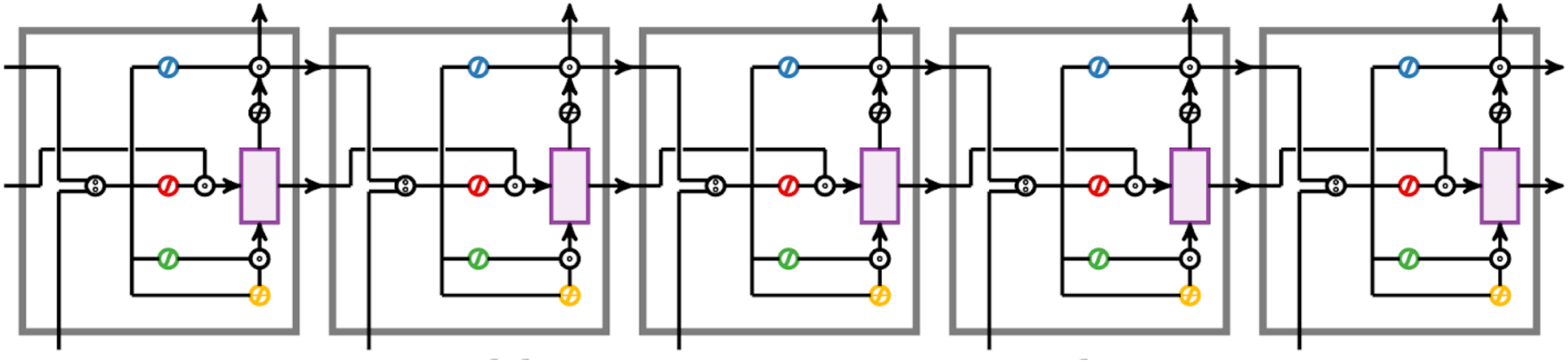
https://www.cnblogs.com/hezhiyao/p/8488888.html

Activation Functions



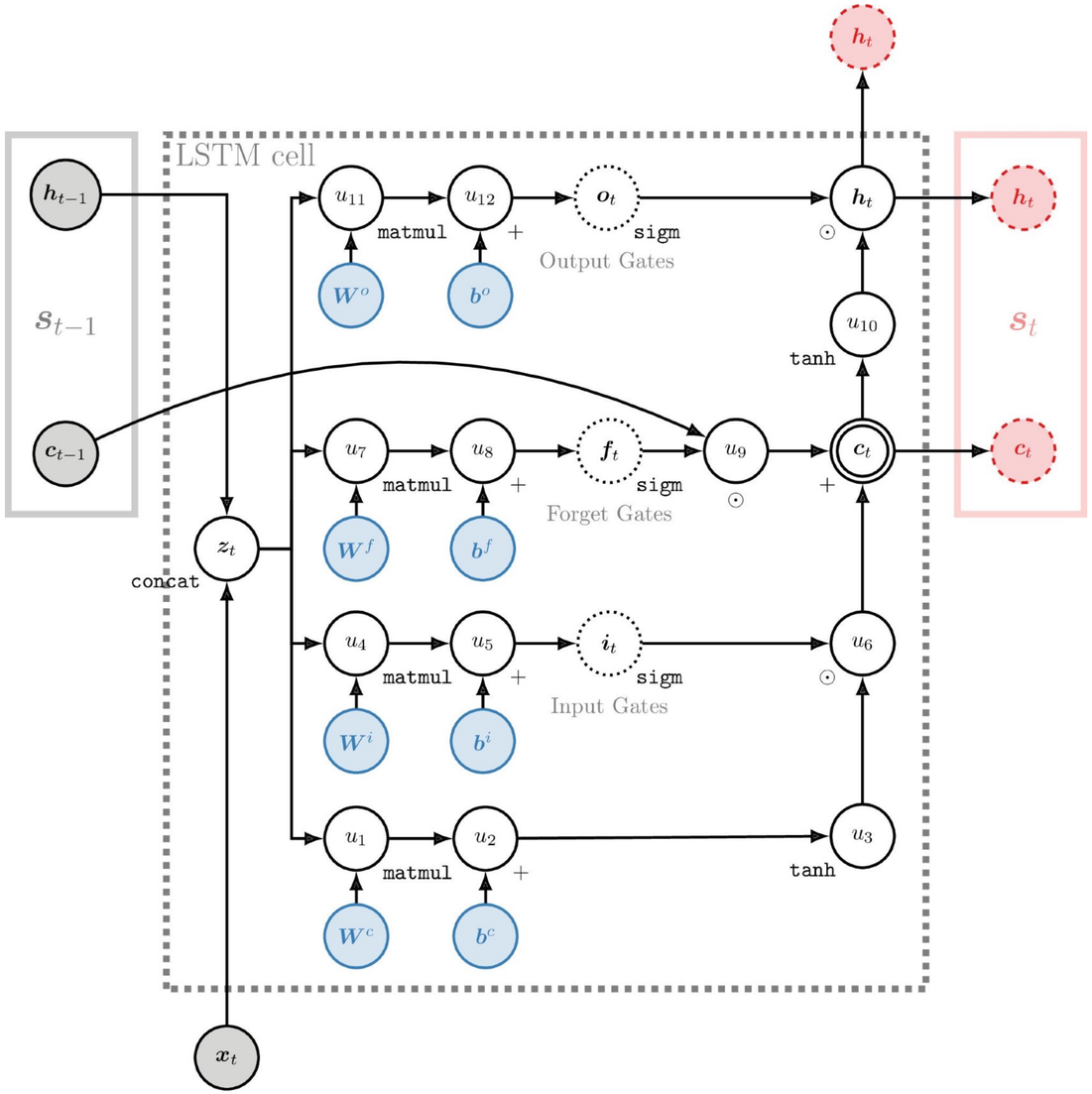
Long Short-term Memory

- Dedicated **memory cell**
- **Input gate**, **forget gate** and **output gate** control flow **into** and out of memory



Hochreiter and Schmidhuber. "Long short-term memory." *Neural computation* 9.8 (1997): 1735-1780.

Long Short-term Memory



$$\mathbf{h}_{t-1}, \mathbf{c}_{t-1} = \mathbf{s}_{t-1}$$

$$\mathbf{z}_t = [\mathbf{x}_t; \mathbf{h}_{t-1}]$$

$$\mathbf{i}_t = \sigma(\mathbf{W}^i \mathbf{z}_t + \mathbf{b}^i)$$

$$\mathbf{f}_t = \sigma(\mathbf{W}^f \mathbf{z}_t + \mathbf{b}^f)$$

$$\mathbf{o}_t = \sigma(\mathbf{W}^o \mathbf{z}_t + \mathbf{b}^o)$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tanh(\mathbf{W}^c \mathbf{z}_t + \mathbf{b}^c)$$

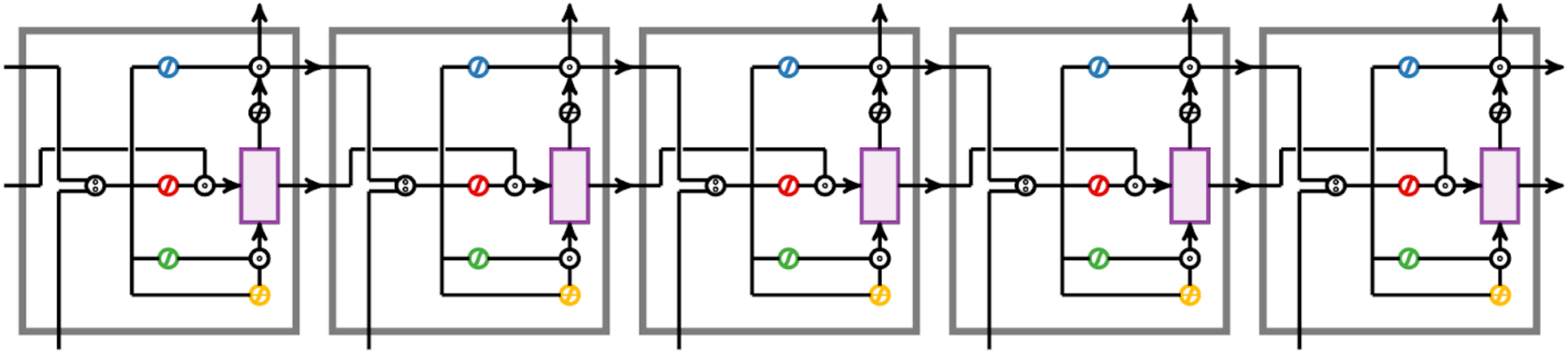
$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t)$$

$$\mathbf{s}_t = [\mathbf{h}_t; \mathbf{c}_t]$$

Hochreiter and Schmidhuber. "Long short-term memory." *Neural computation* 9.8 (1997): 1735-1780.

Long Short-term Memory

- Dedicated **memory cell**
- **Input gate**, **forget gate** and **output gate** control flow **into** and out of memory

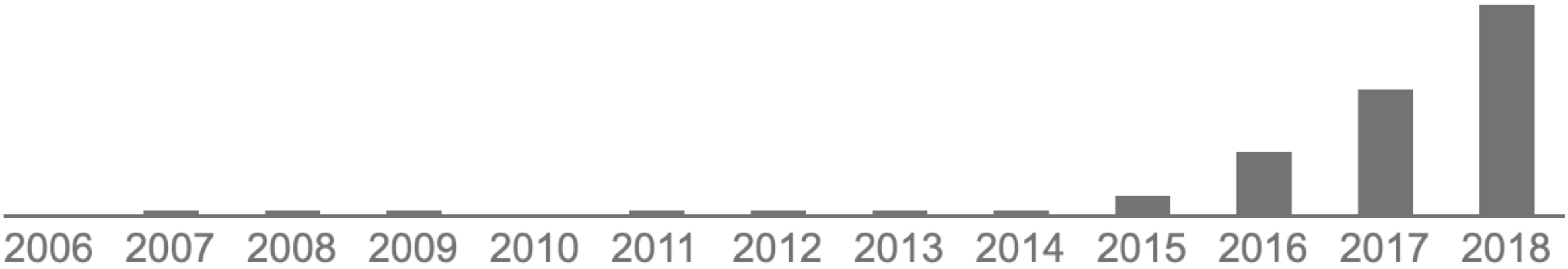


$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tanh(\mathbf{W}^c \mathbf{z}_t + \mathbf{b}^c)$$

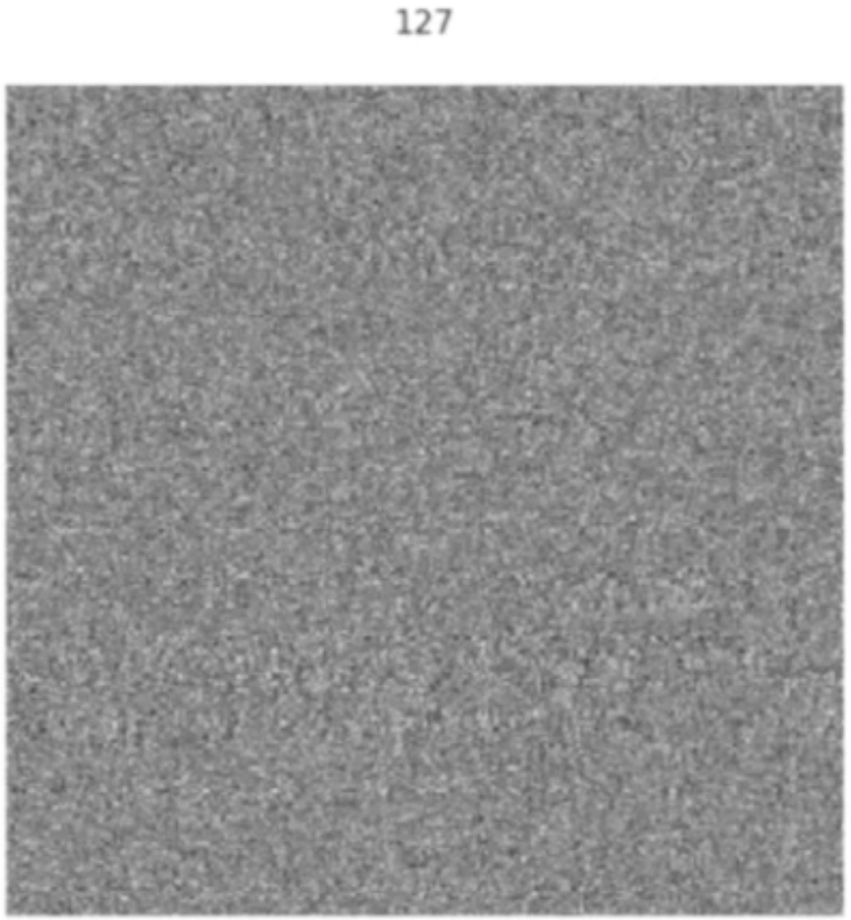
$$\frac{\partial \mathbf{c}_t}{\partial \mathbf{c}_{t-1}} = \frac{\partial}{\partial \mathbf{c}_{t-1}} \left[\mathbf{f}_t \odot \mathbf{c}_{t-1}\right] + \frac{\partial}{\partial \mathbf{c}_{t-1}} \left[\mathbf{i}_t \odot \tanh(\mathbf{W}^c \mathbf{z}_t + \mathbf{b}^c)\right]$$

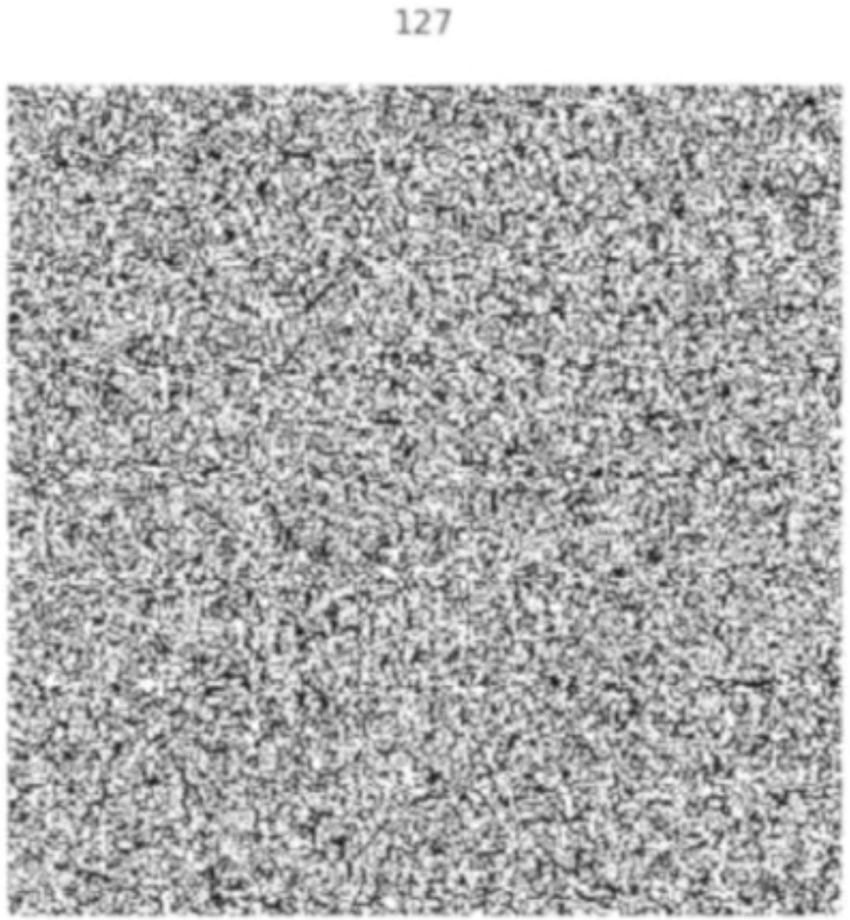
Long Short-term Memory

Cited by 15513



Mitigates Vanishing and Exploding Gradients





LSTM Efficiency

$$
\begin{aligned}
\mathbf{h}_{t-1}, \mathbf{c}_{t-1} &= \mathbf{s}_{t-1} \\
\mathbf{z}_t &= [\mathbf{x}_t; \mathbf{h}_{t-1}] \\
\mathbf{i}_t &= \sigma(\mathbf{W}^i \mathbf{z}_t + \mathbf{b}^i) \\
\mathbf{f}_t &= \sigma(\mathbf{W}^f \mathbf{z}_t + \mathbf{b}^f) \\
\mathbf{o}_t &= \sigma(\mathbf{W}^o \mathbf{z}_t + \mathbf{b}^o) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tanh(\mathbf{W}^c \mathbf{z}_t + \mathbf{b}^c) \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \\
\mathbf{s}_t &= [\mathbf{h}_t; \mathbf{c}_t]
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{h}_{t-1}, \mathbf{c}_{t-1} &= \mathbf{s}_{t-1} \\
\mathbf{z}_t &= [\mathbf{x}_t; \mathbf{h}_{t-1}] \\
\mathbf{u}_t &= \mathbf{W}\mathbf{z}_t + \mathbf{b} \\
\mathbf{i}_t &= \sigma(\mathbf{u}_t[1 \,..\, k]) \\
\mathbf{f}_t &= \sigma(\mathbf{u}_t[k+1 \,..\, 2k]) \\
\mathbf{o}_t &= \sigma(\mathbf{u}_t[2k+1 \,..\, 3k]) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tanh(\mathbf{u}_t[3k+1 \,..\, 4k]) \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \\
\mathbf{s}_t &= [\mathbf{h}_t; \mathbf{c}_t]
\end{aligned}
$$

Gated Recurrent Unit

$$\mathbf{z}_t = \sigma(\mathbf{W}^z[\mathbf{x}_t; \mathbf{s}_{t-1}] + \mathbf{b}^z)$$

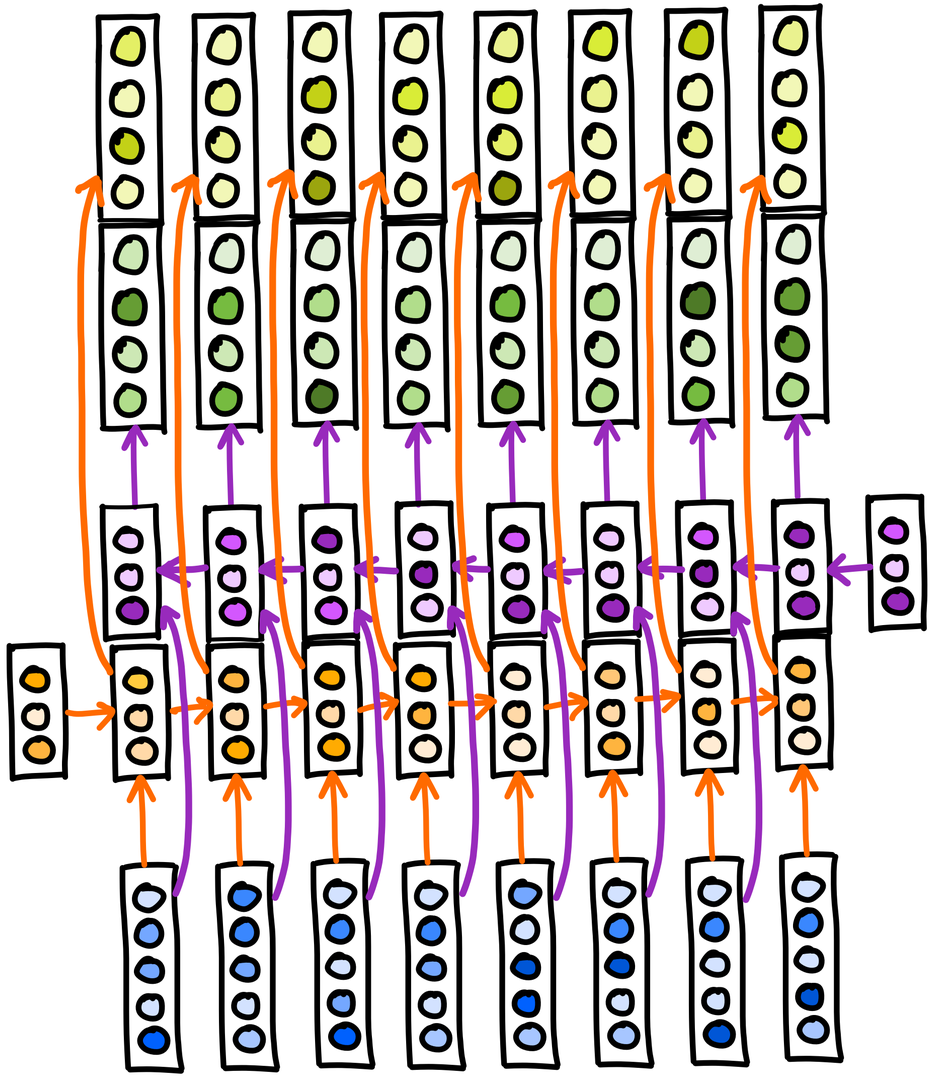
$$\mathbf{r}_t = \sigma(\mathbf{W}^r[\mathbf{x}_t; \mathbf{s}_{t-1}] + \mathbf{b}^r)$$

$$\mathbf{s}_t = (1 - \mathbf{z}_t) \odot \mathbf{s}_{t-1} + \mathbf{z}_t \odot \sigma(\mathbf{W}^x[\mathbf{x}_t; \mathbf{r}_t \odot \mathbf{s}_{t-1}] + \mathbf{b}^x)$$

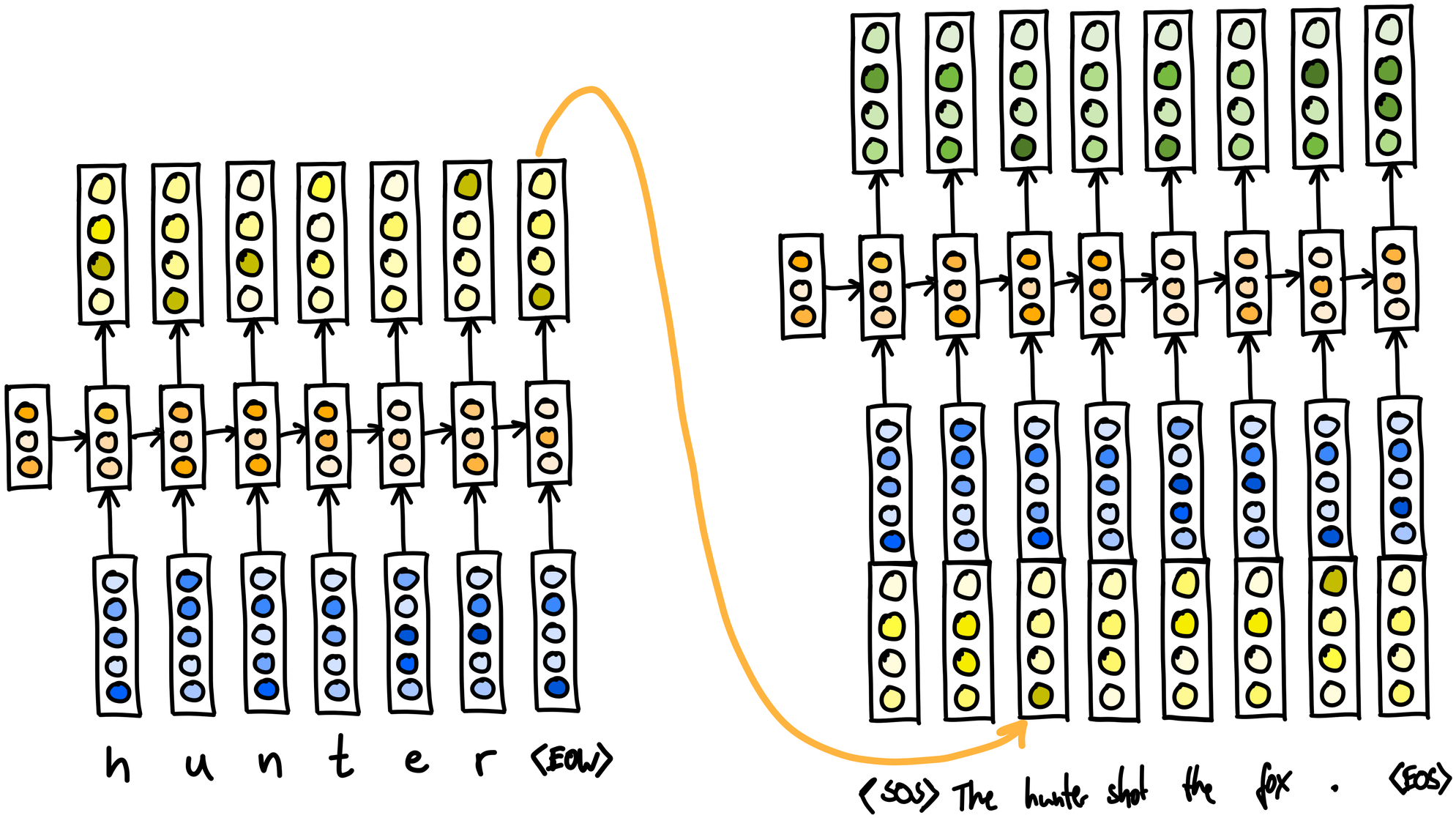
$$\mathbf{h}_t = \mathbf{s}_t$$

Cho et al. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. 2014.

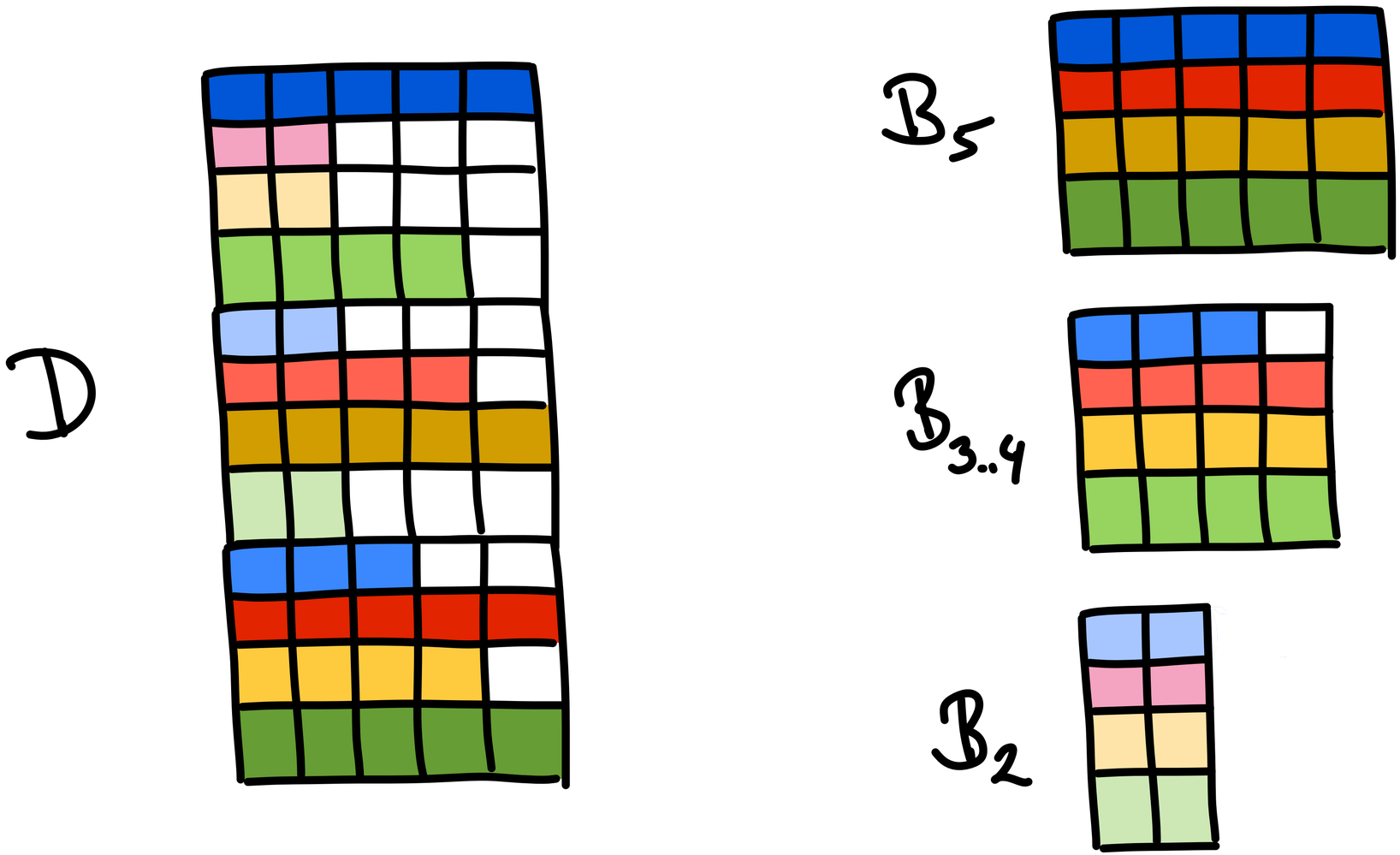
Bidirectional RNNs



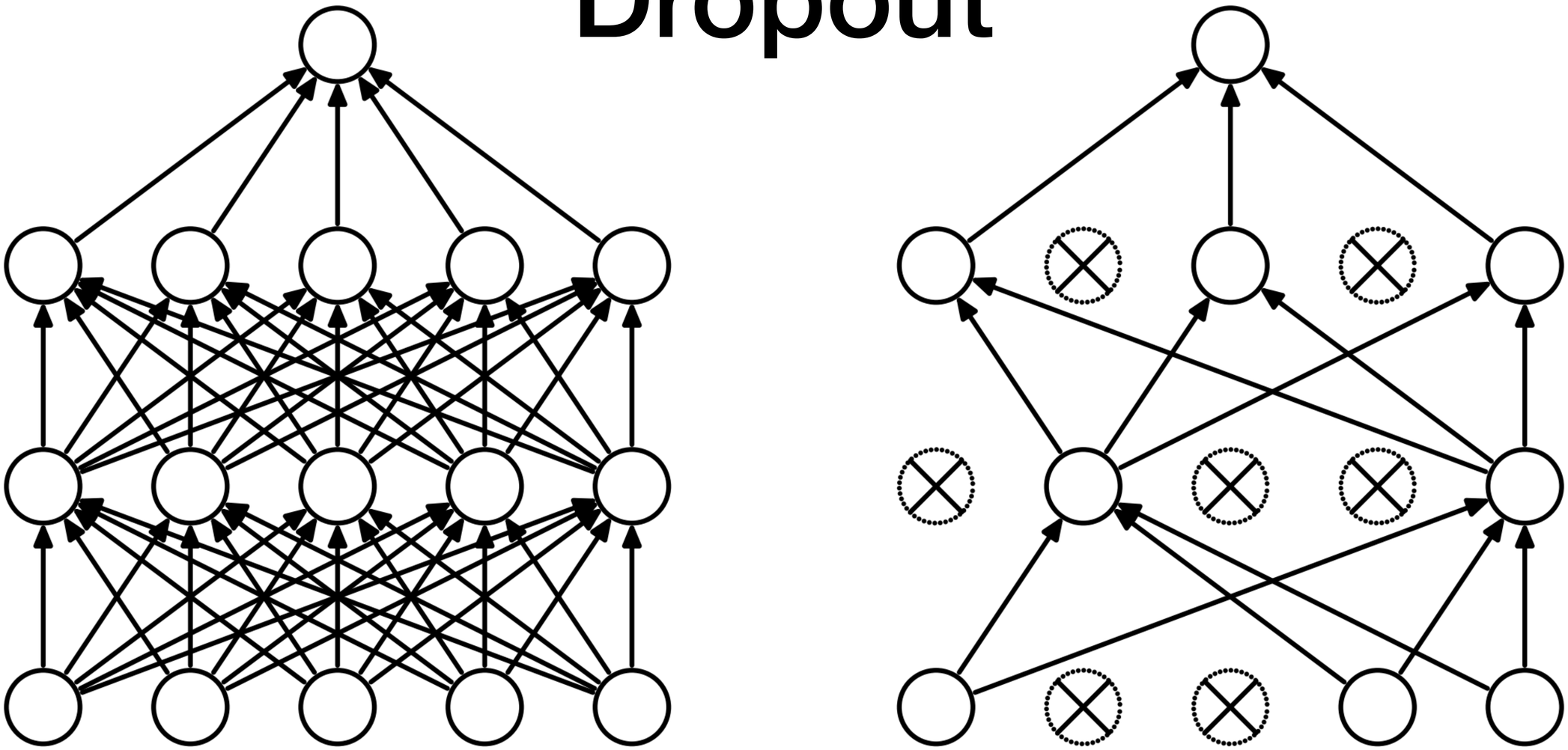
Character-level Models



Bucketing



Dropout

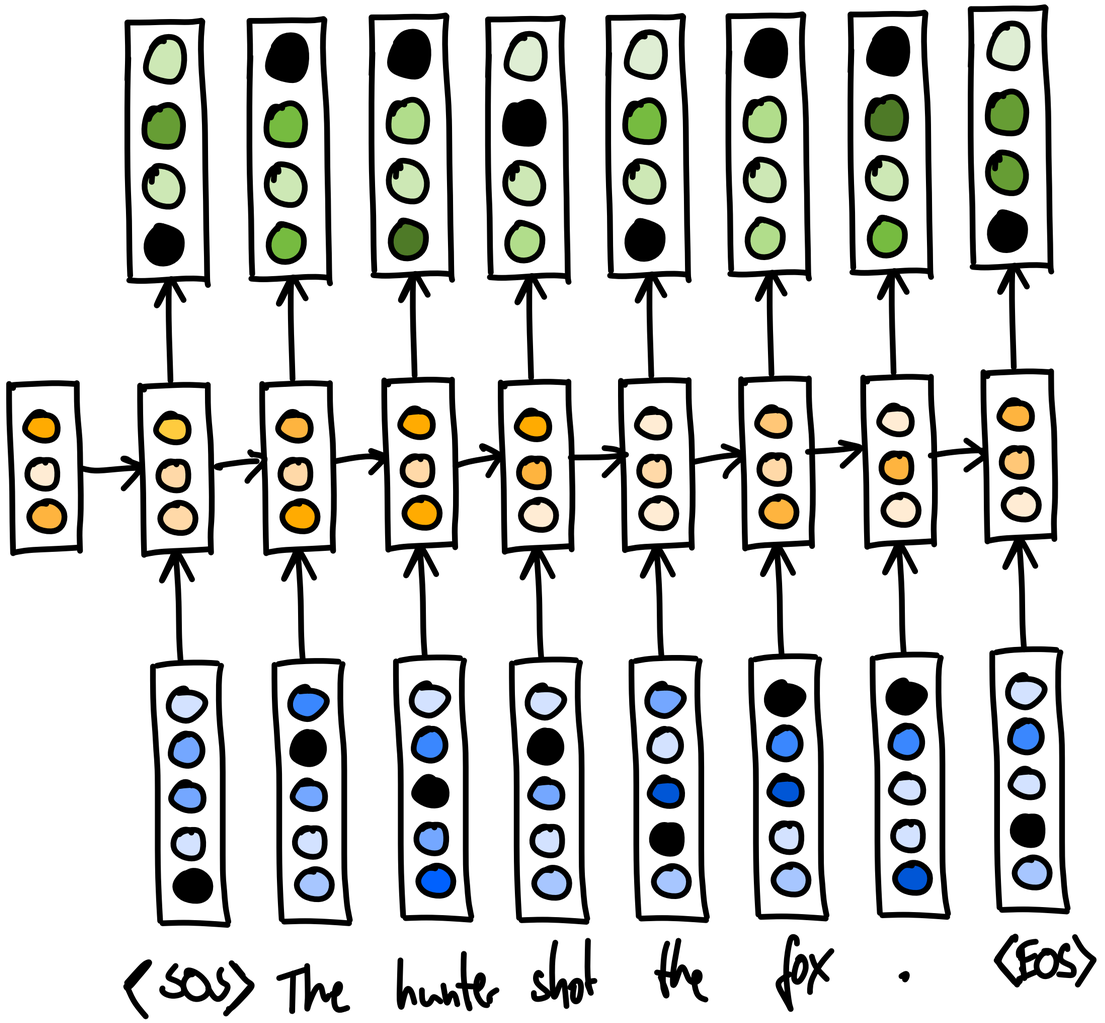


(a) Standard Neural Net

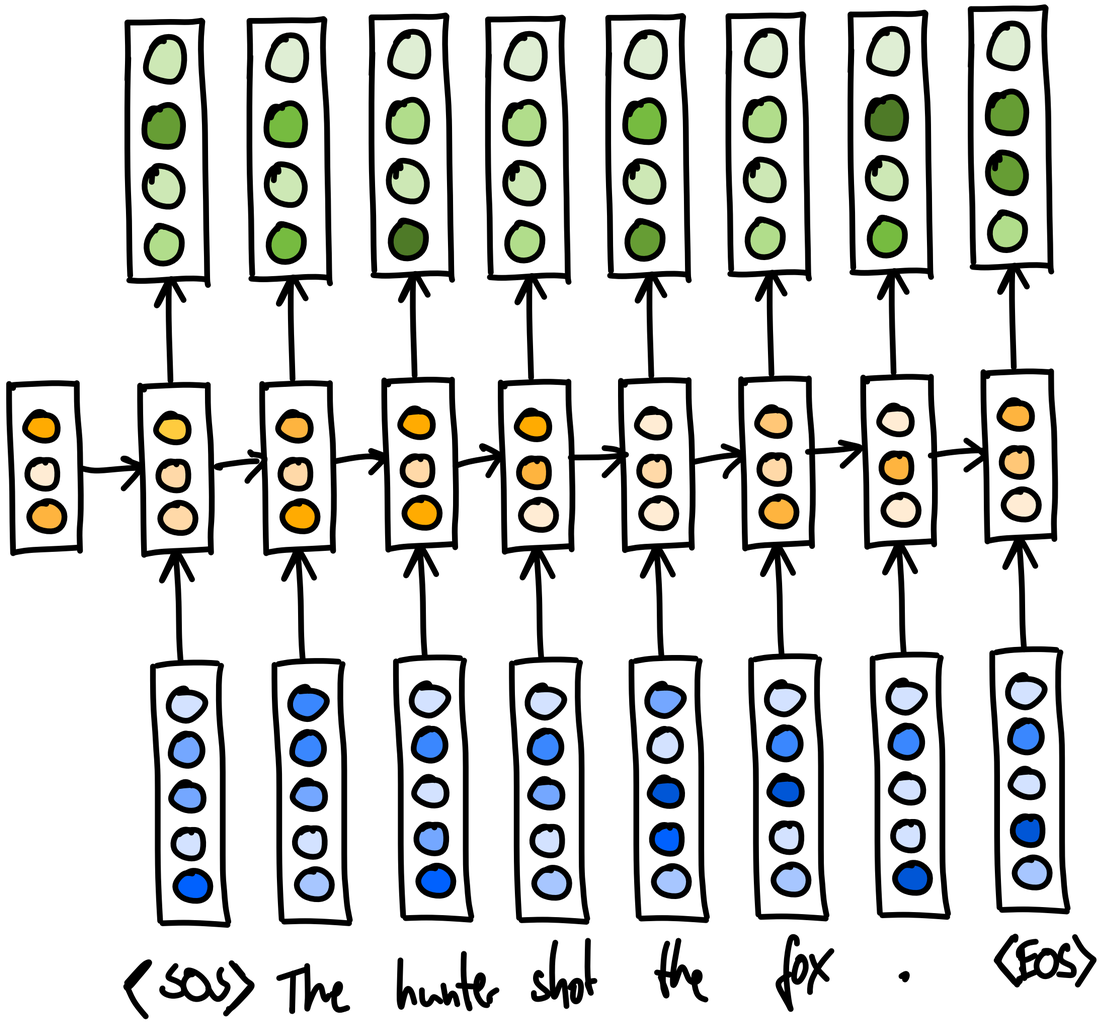
(b) After applying dropout.

Srivastava et al. Dropout: A Simple Way To Prevent Neural Networks From Overfitting. JMLR 2014

Dropout



Recurrent Dropout?



Gal and Ghahramani. A Theoretically Grounded Application of Dropout in Recurrent Neural Networks. NIPS 2016.

Applications

- Language Modeling
- Recognizing Textual Entailment
- Named Entity Recognition
- Machine Translation
- Question Answering
- Dialog Modeling
- Language Generation
- Sentence Summarization
- Paraphrasing
- Sentiment Analysis
- …

Recognizing Textual Entailment

A man is crowd surfing at a concert

- The man is at a football game **Contradiction**
- The man is drunk **Neutral**
- The man is at a concert **Entailment**

A wedding party is taking pictures

- There is a funeral **Contradiction**
- They are outside **Neutral**
- Someone got married **Entailment**

State-of-the-Art Before 2015

[Lai and Hockenmaier, 2014, Jimenez et al., 2014, Zhao et al., 2014, Beltagy et al., 2015 etc.]

- Engineered natural language processing pipelines
- Various external resources
- Specialized subcomponents
- Extensive manual creation of features:
  - Negation detection, word overlap, part-of-speech tags, dependency parses, alignment, unaligned matching, chunk alignment, synonym, hypernym, antonym, denotation graph...

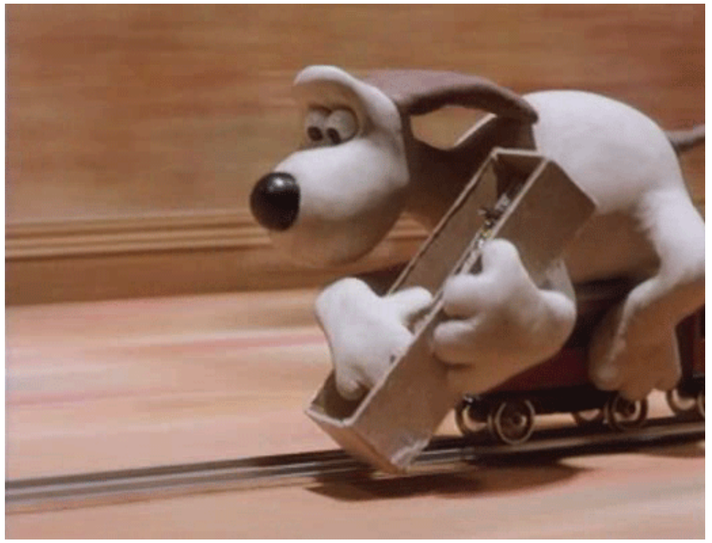
Data Matters!

Neural Networks on SICK corpus
[Marelli et al., SemEval 2014]

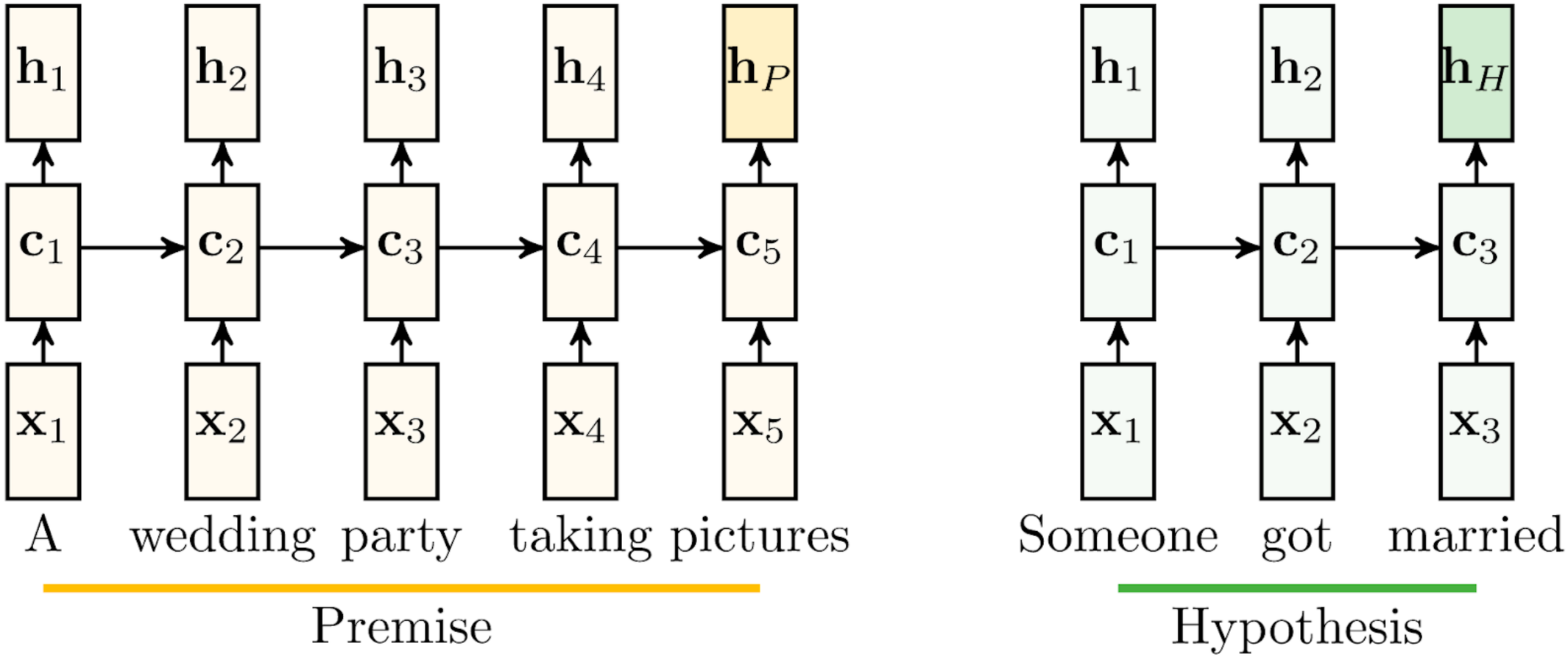
10k sentence pairs, partly synthetic

Neural Networks on SNLI corpus
[Bowman et al., EMNLP 2015]

570k sentence pairs from Mechanical Turkers
EMNLP 2015 “best data set or resource” award!

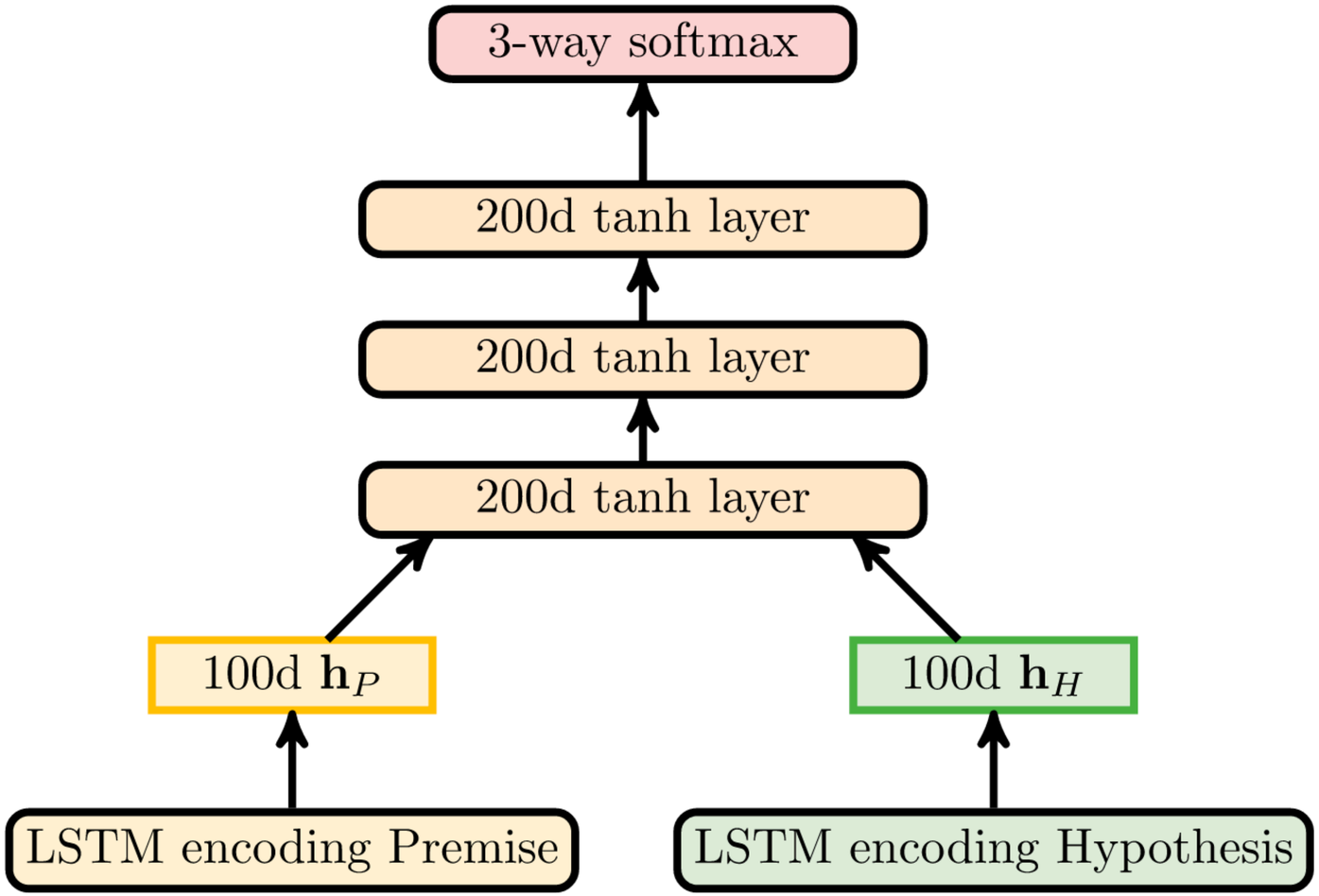


Independent Sentence Encoding

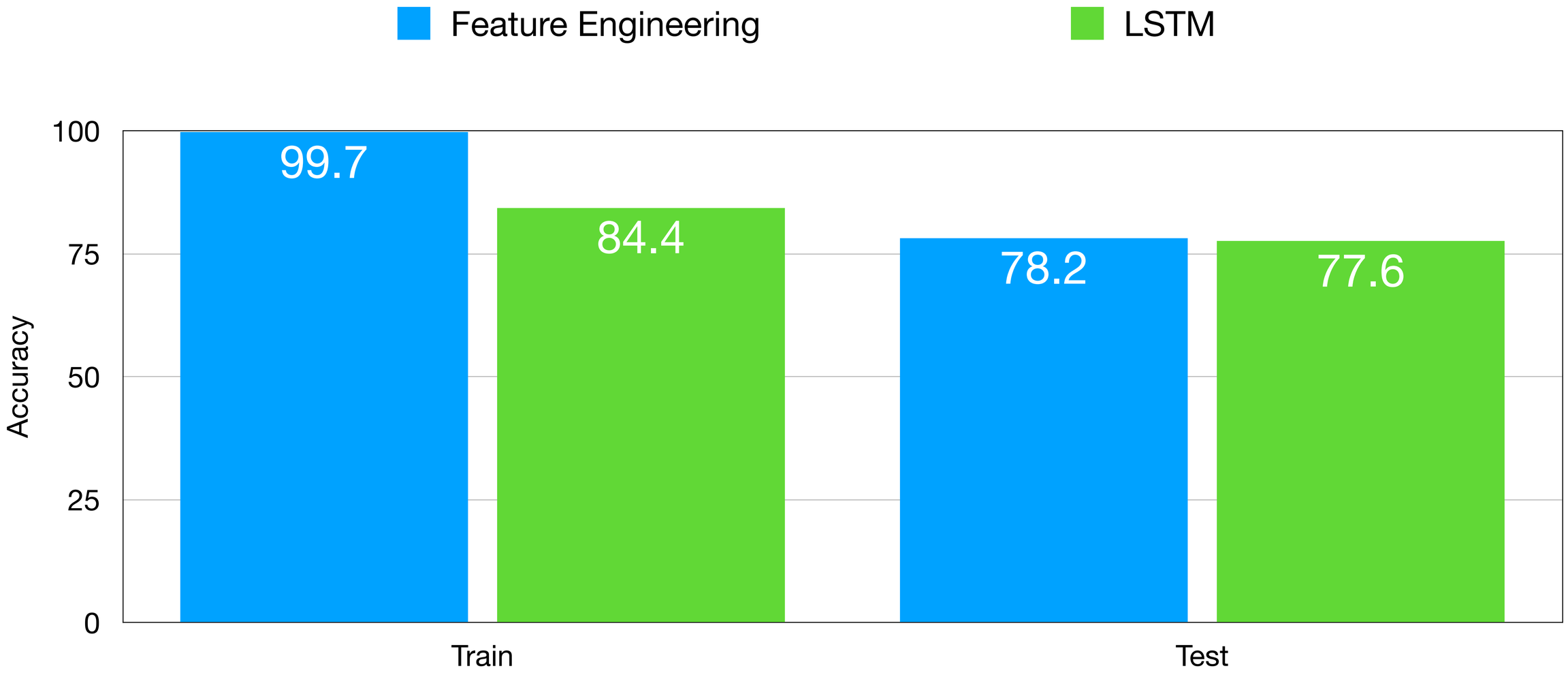


Bowman et al. A large annotated corpus for learning natural language inference. EMNLP 2015.

Independent Sentence Encoding



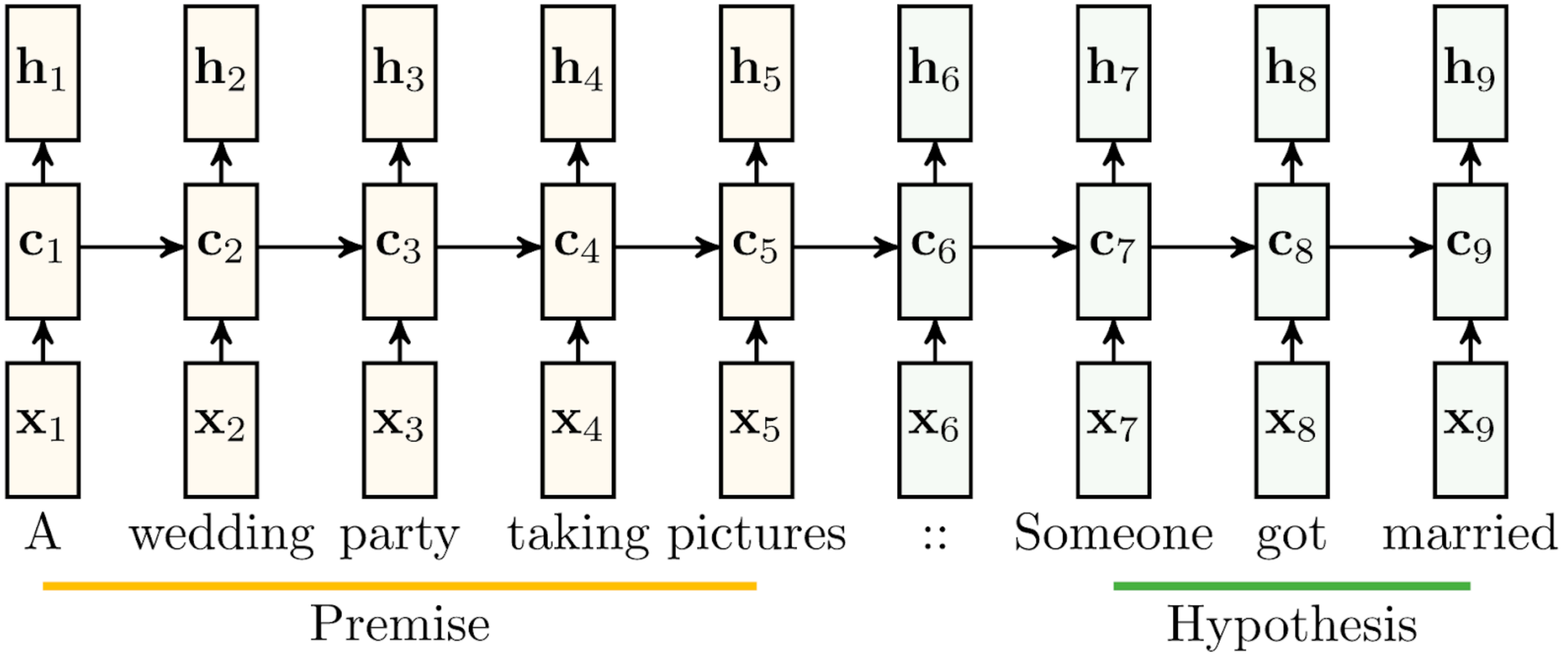
Results


Feature Engineering
LSTM
100
75
50
25
0
Accuracy
99.7
84.4
78.2
77.6
Train
Test

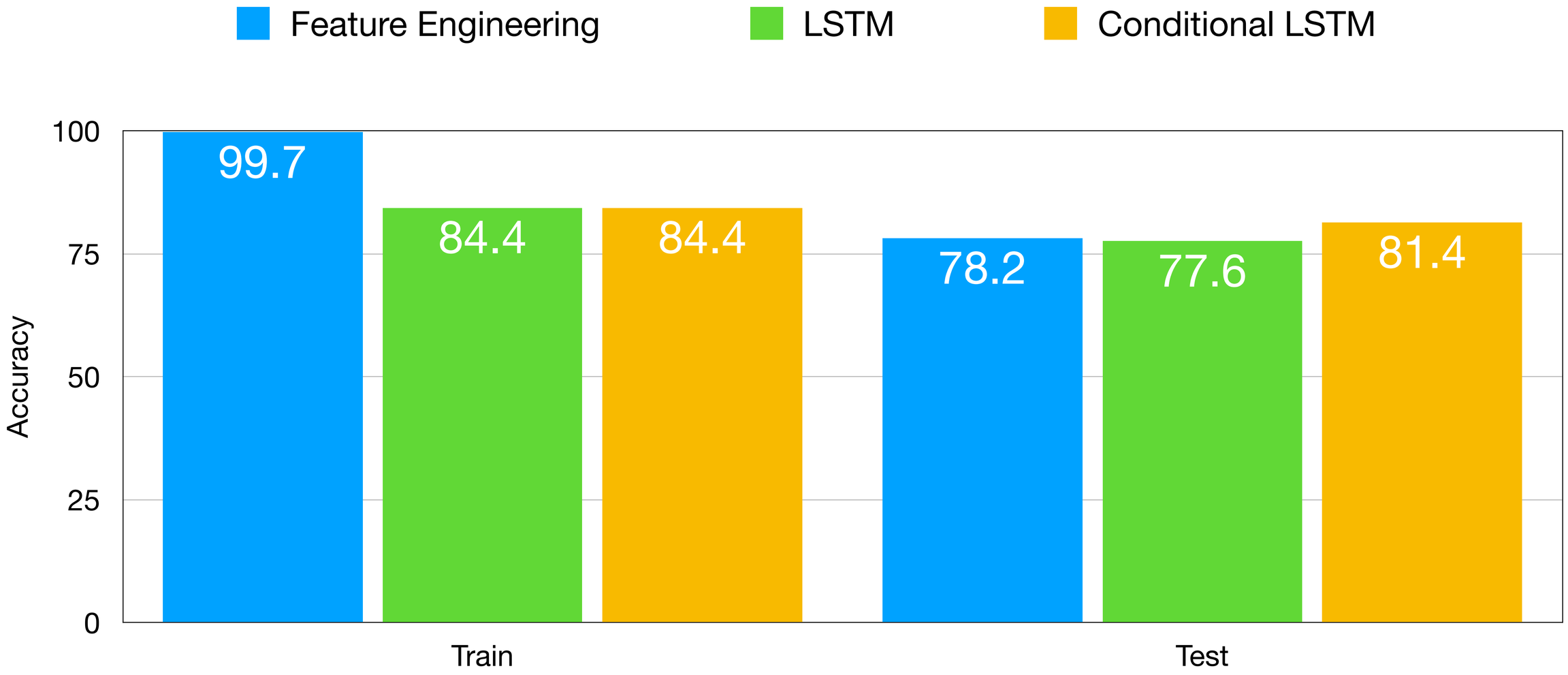
“You can’t cram the meaning of a whole %&!$# sentence into a single $&!#* vector!”

— Raymond J. Mooney

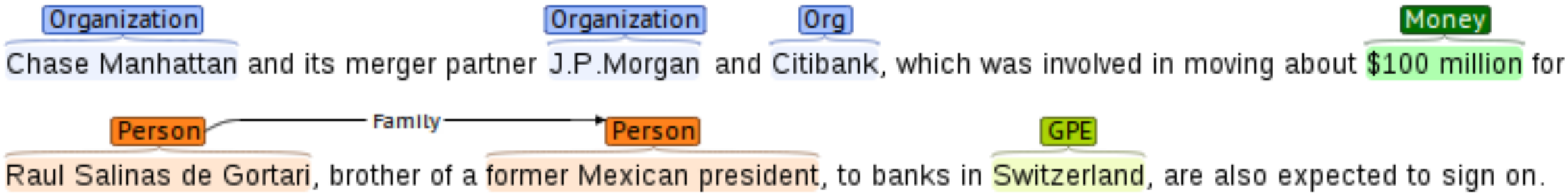
Conditional Encoding



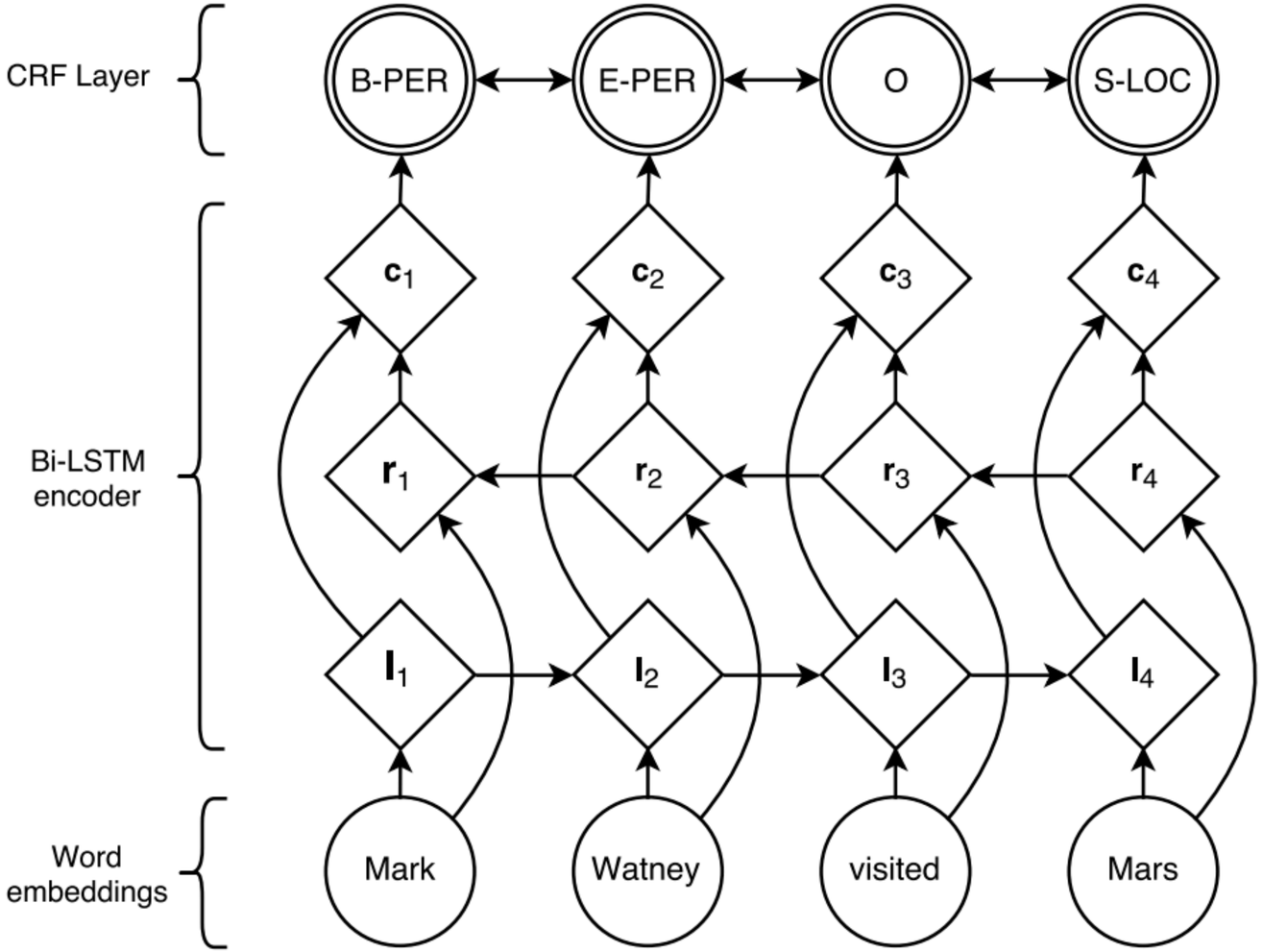
Results

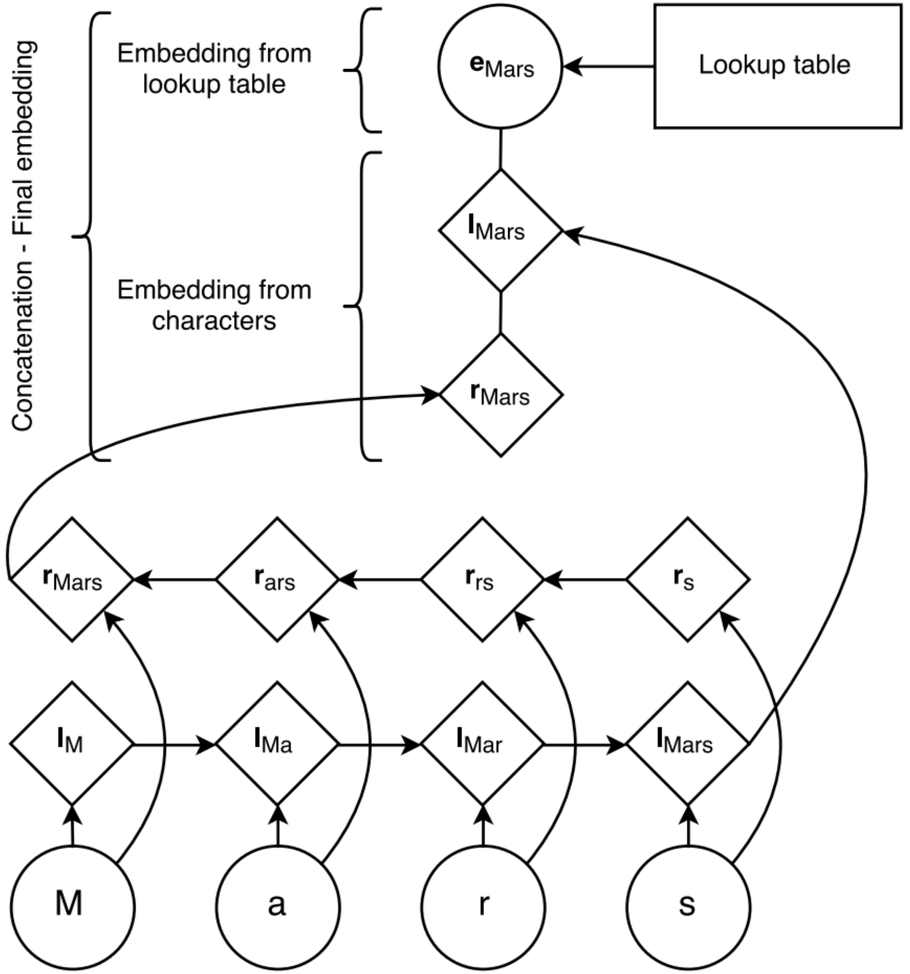

Feature Engineering
LSTM
Conditional LSTM
100
75
50
25
0
Accuracy
99.7
84.4
84.4
78.2
77.6
81.4
Train
Test

Named Entity Recognition



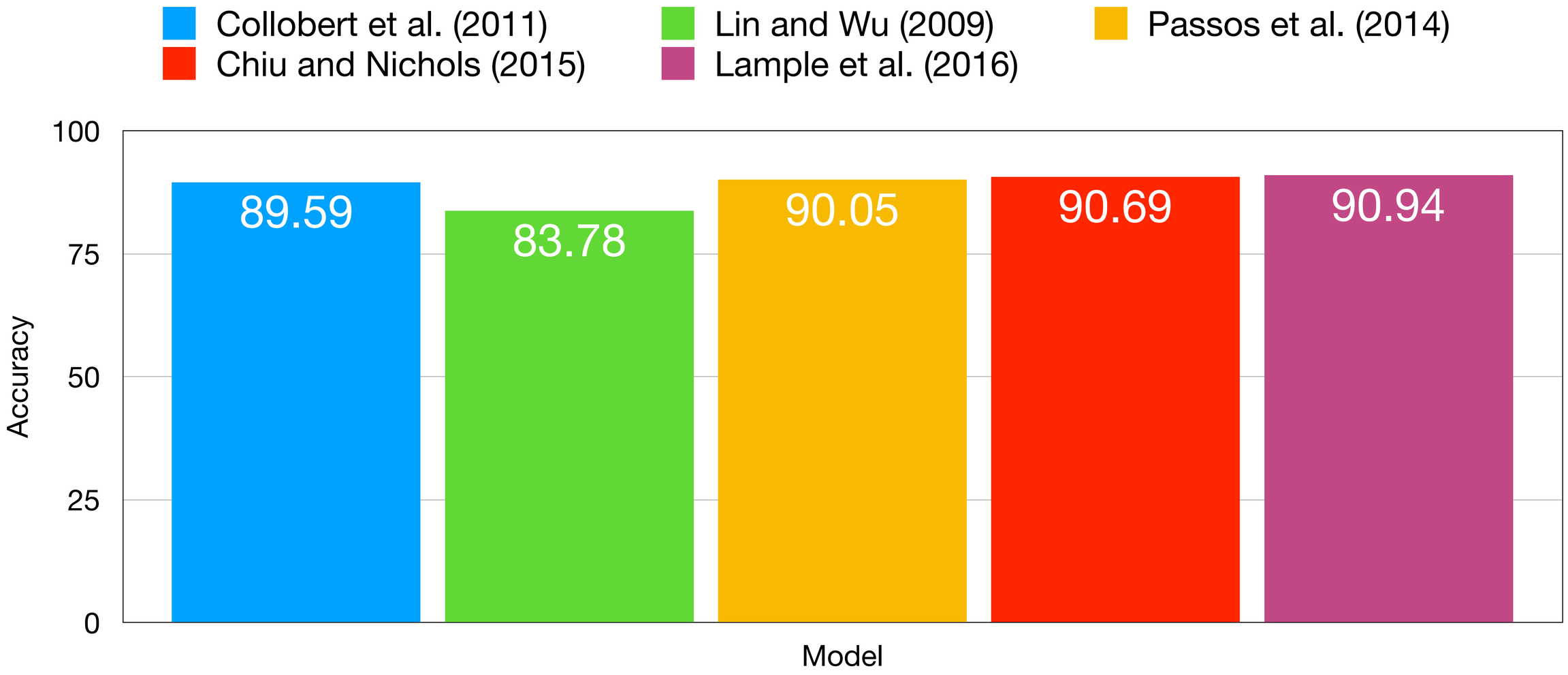
Bidirectional Character-aware LSTM





Lample et al. Neural Architectures for Named Entity Recognition. 2016

Results


Collobert et al. (2011)
Chiu and Nichols (2015)
Lin and Wu (2009)
Lample et al. (2016)
Passos et al. (2014)
100
75
50
25
0
Accuracy
89.59
83.78
90.05
90.69
90.94
Model