LANGUAGE MODELING WITH GATED CONVOLUTIONAL NETWORKS

YANN N. DAUPHIN, ANGELA FAN, MICHAEL AULI AND DAVID GRANGIER
FACEBOOK AI RESEARCH

CS 546 Paper Presentation
Jinfeng Xiao
2/22/2018

Intro: Language Models

- Full model:

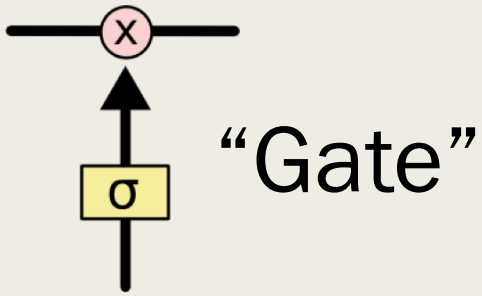
$$P(w_0, \dots, w_N) = P(w_0) \prod_{i=1}^{N} P(w_i | w_0, \dots, w_{i-1})$$

- n-gram model:

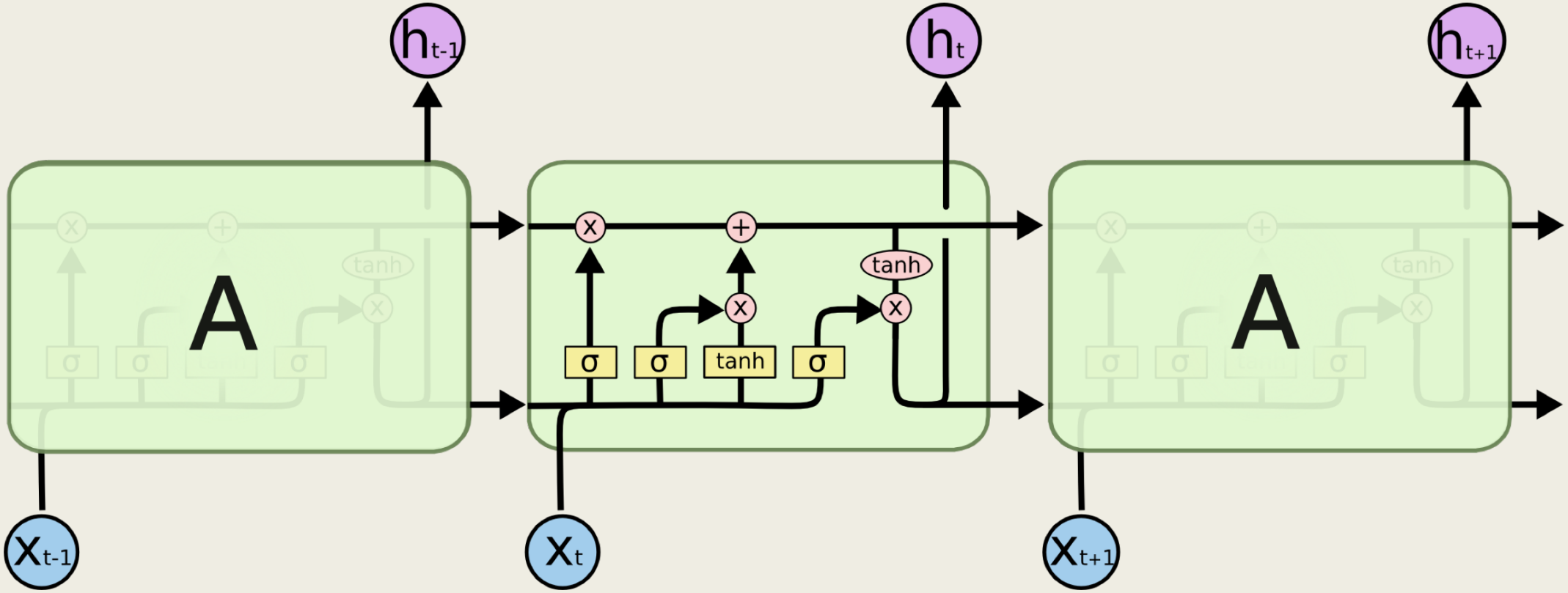
$$P(w_i) = P(w_i | w_{i-n+1}, \dots, w_{i-1})$$

- Hard to represent long-range dependencies, due to data sparsity

Intro: LSTM



"Gate"



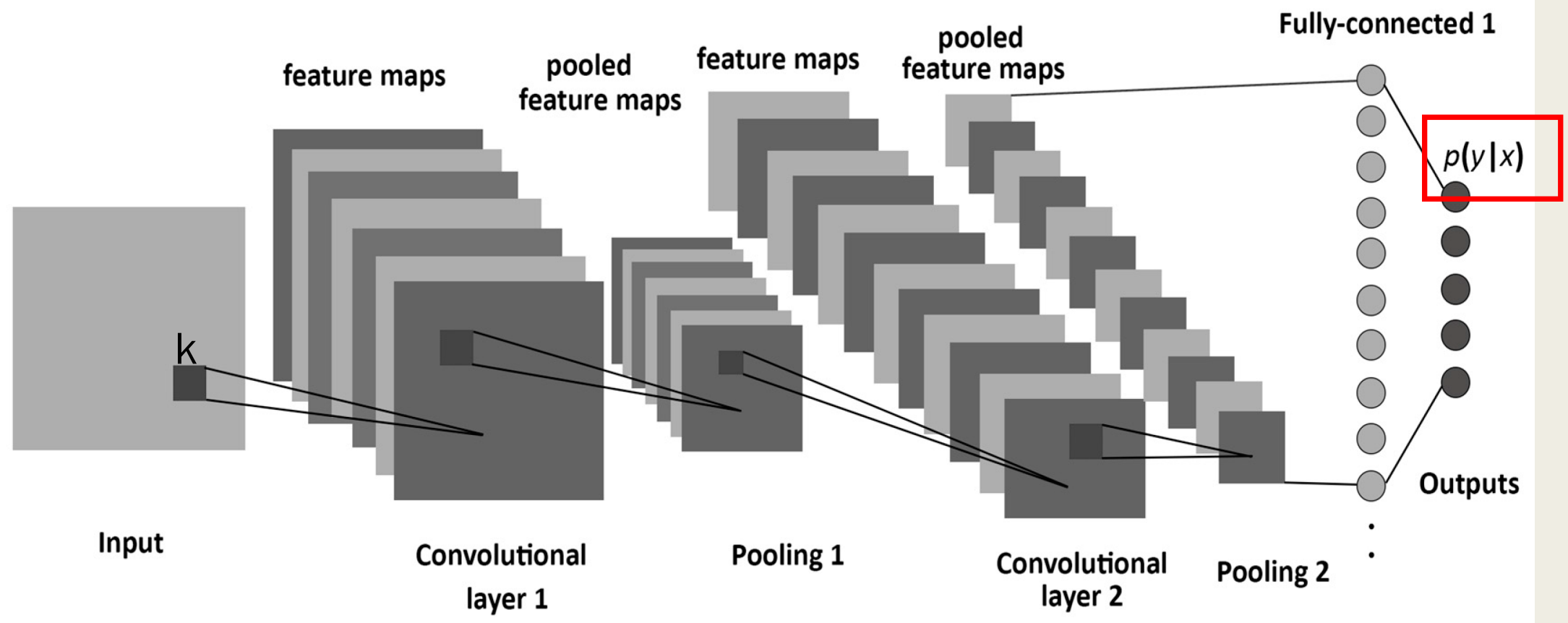
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Intro: LSTM

- State-of-the-art neural network approach for language modeling
- + Can theoretically model arbitrarily long dependencies
- -- Not parallelizable; O(N) operations

http://colah.github.io/posts/2015-08-Understanding-LSTMs

Intro: CNN



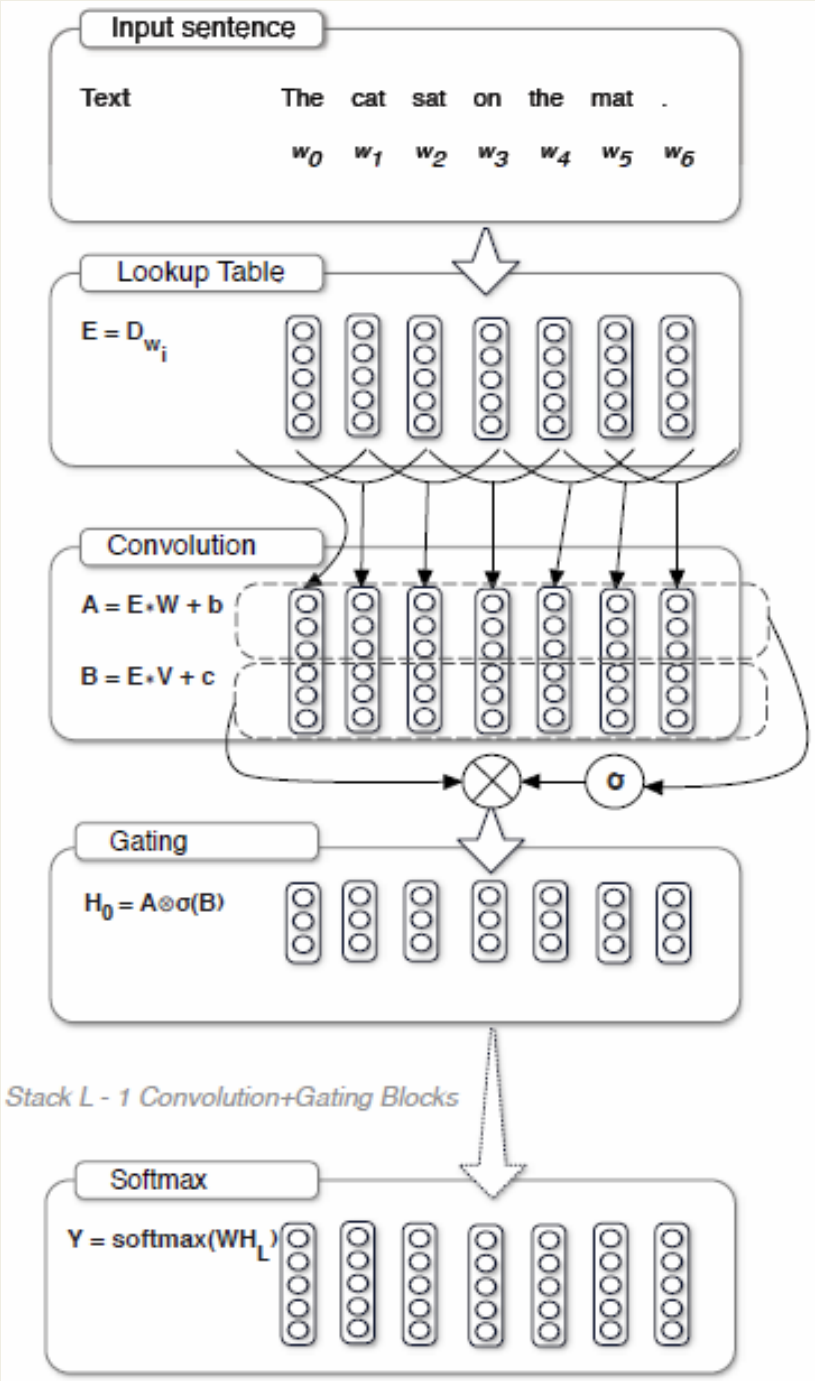
- Predict the current word y with previous words x (i.e. context)
- Model long-term dependencies with O(N/k) operations

This Paper: GCNN

- Gated Convolutional Neural Networks
- Each CNN layer is followed by a gating layer
- Allows parallelization over sequential tokens
- Reduces the latency to score a sentence by an order of magnitude
- Competitive performance on WikiText-103 and Google Billion Words benchmarks

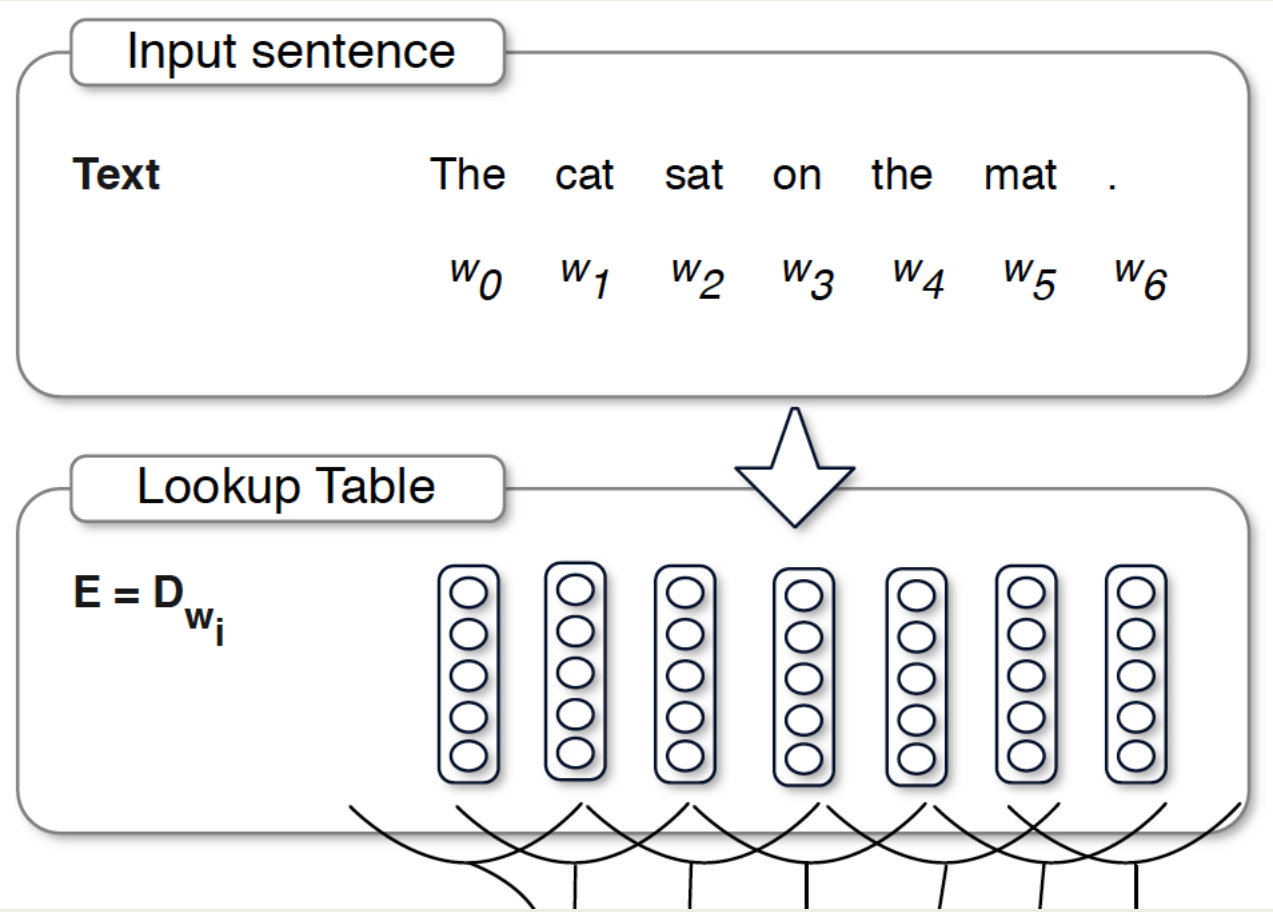
Architecture

- Word Embedding +
- CNN +
- Gating



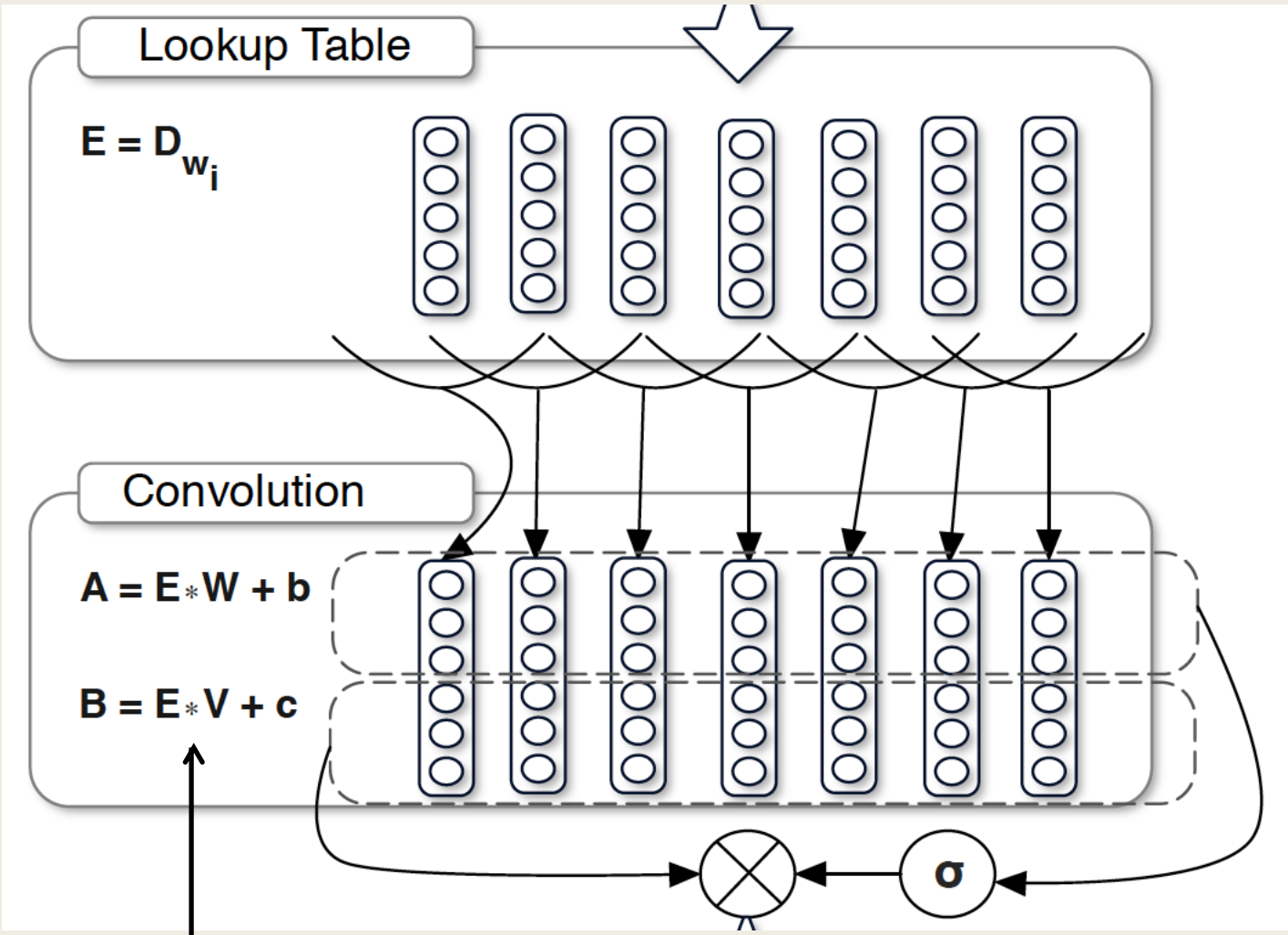
Architecture

- **Word Embedding +**
- CNN +
- Gating



Architecture

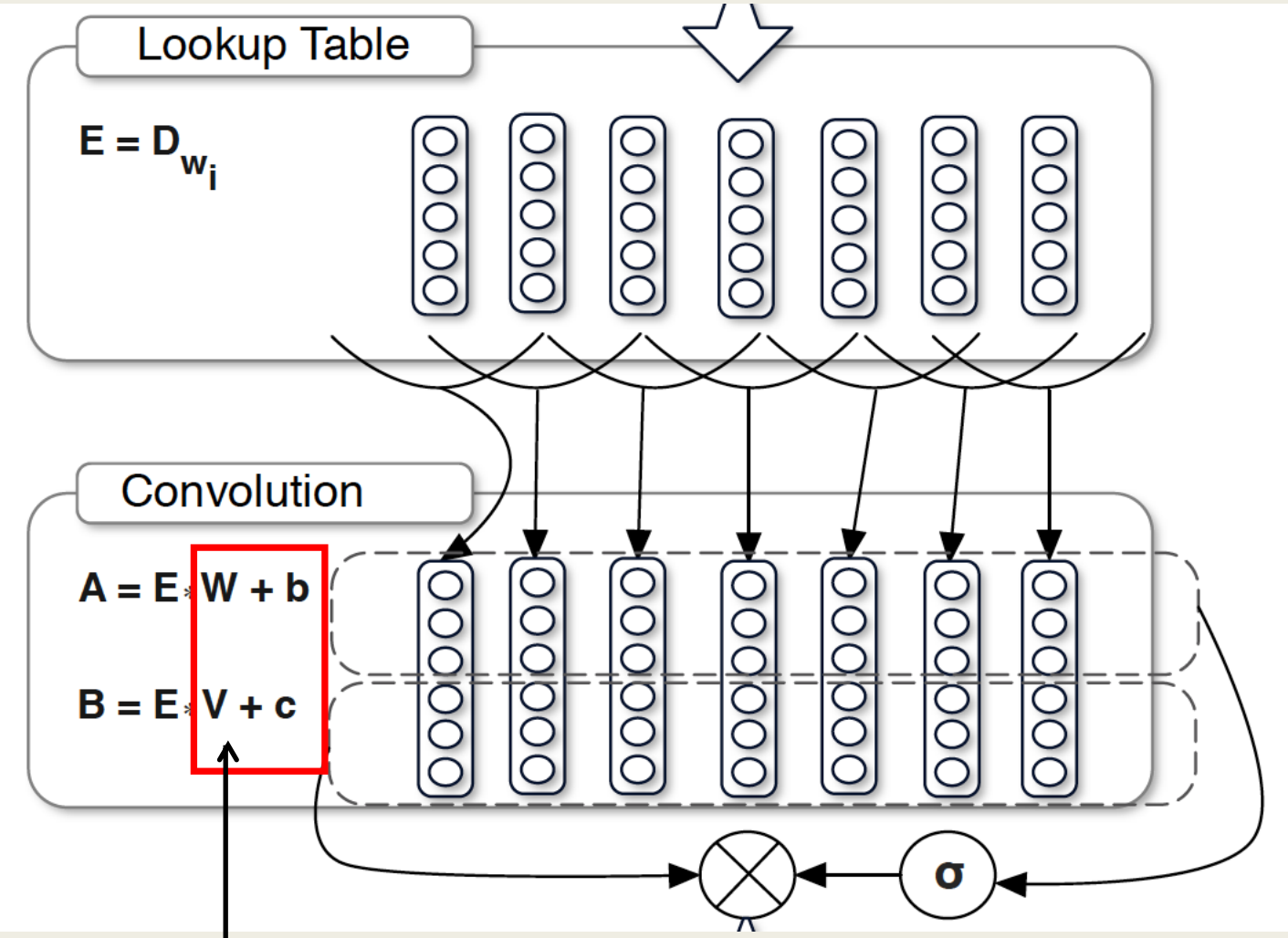
- Word Embedding +
- CNN +
- Gating



*: Convolution operation

Architecture

- Word Embedding +
- **CNN** +
- Gating



learned parameters

Example: Convolution

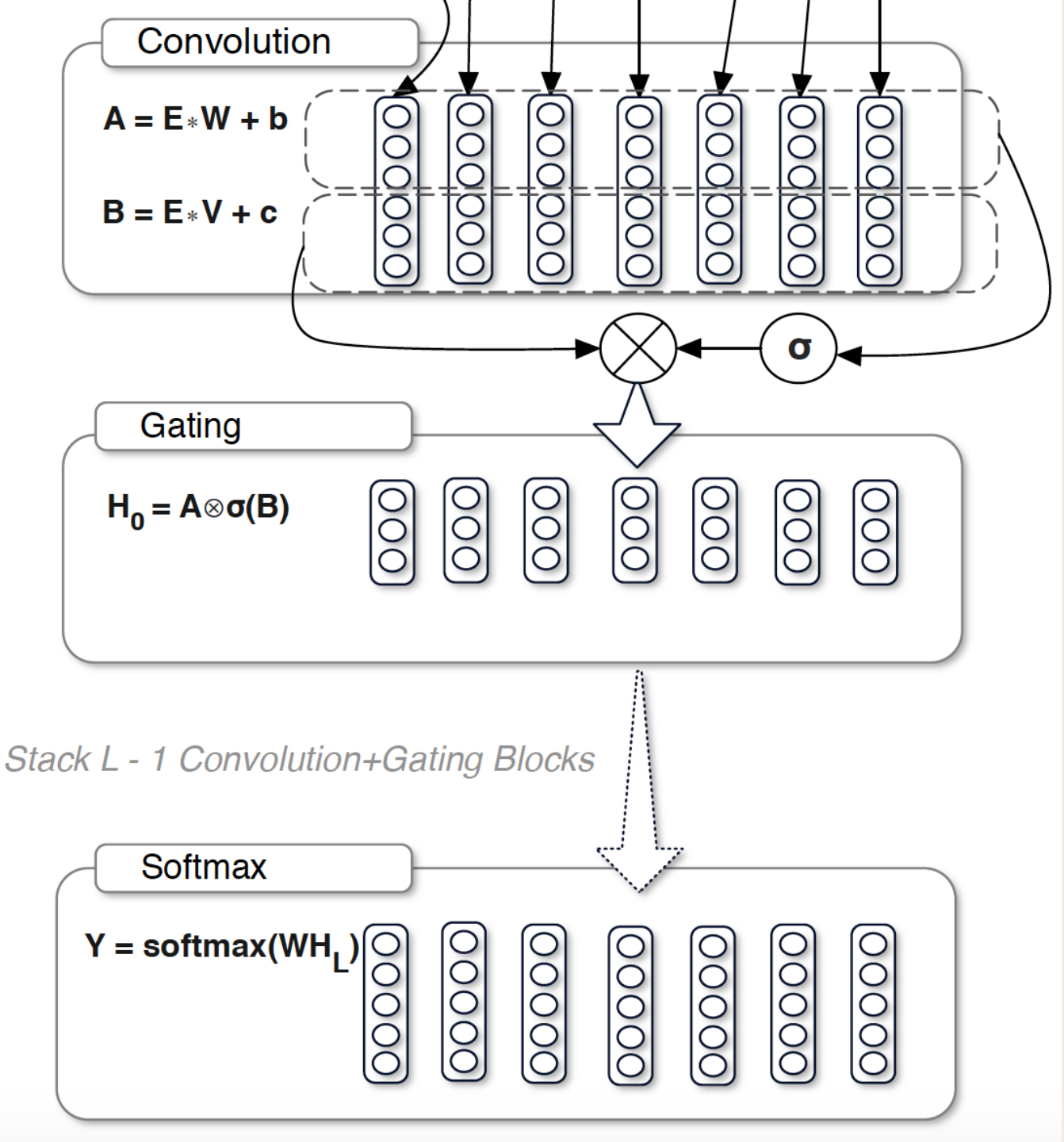
$$\left(\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} * \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}\right)[2,2] = (i \cdot 1) + (h \cdot 2) + (g \cdot 3) + (f \cdot 4) + (e \cdot 5) + (d \cdot 6) + (c \cdot 7) + (b \cdot 8) + (a \cdot 9).$$

- “Average” over a small patch around an element

http://colah.github.io/posts/2015-08-Understanding-LSTMs

Architecture

- Word Embedding +
- CNN +
- **Gating**



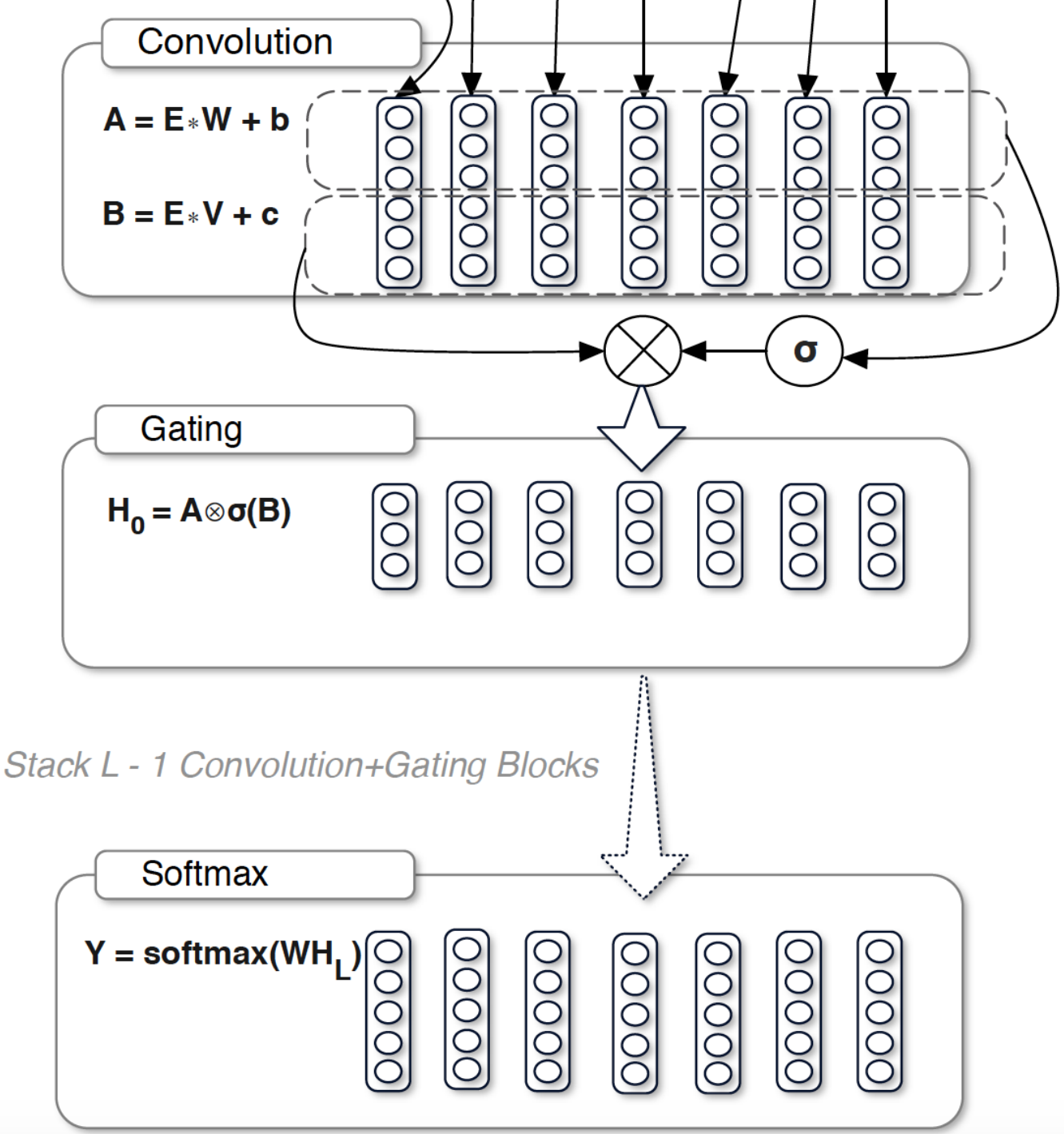
Two Gating Mechanisms

- Gated linear units (GLU)

$$h_l(\mathbf{E}) = (\mathbf{E} * \mathbf{W} + b) \otimes \sigma(\mathbf{E} * \mathbf{V} + c)$$

- Gated tanh units (GTU)

$$h_t(\mathbf{E}) = \tanh(\mathbf{E} * \mathbf{W} + b) \otimes \sigma(\mathbf{E} * \mathbf{V} + c)$$



Evaluation Metric: Perplexity

- The perplexity of a discrete probability distribution p is

$$e^{\frac{1}{N}\sum_{i=1}^{N} -\log p(w_i|\ldots,w_{i-1})}$$

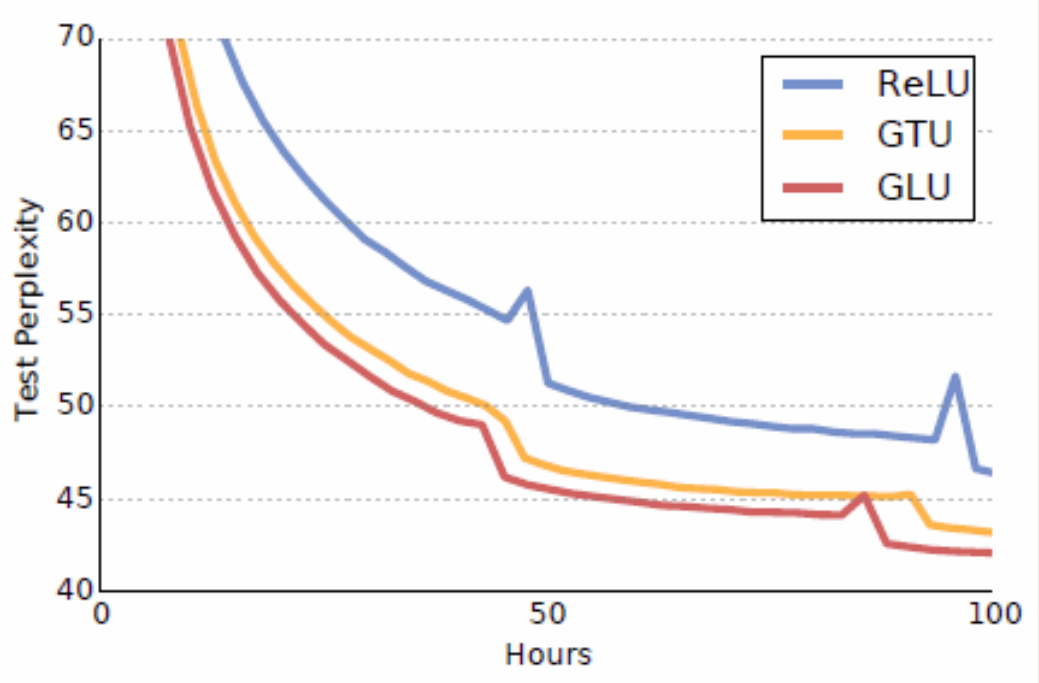
- It measures how well our model matches the held out test data set.
- The smaller, the better.

https://en.wikipedia.org/wiki/Perplexity

Benchmark: Google Billion Word

Average Sequence Length = 20 Words

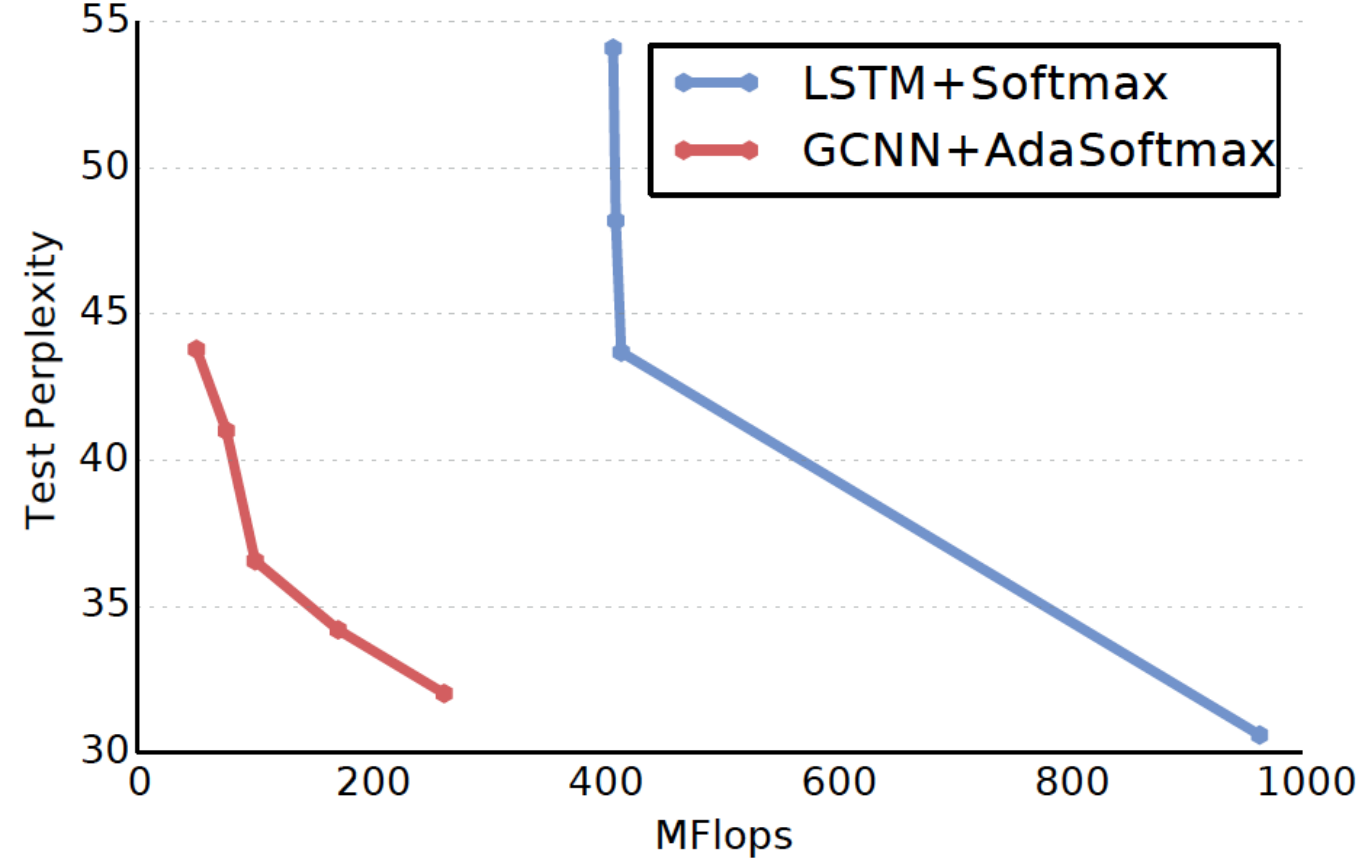
Model	Test PPL	Hardware
Sigmoid-RNN-2048 (Ji et al., 2015)	68.3	1 CPU
Interpolated KN 5-Gram (Chelba et al., 2013)	67.6	100 CPUs
Sparse Non-Negative Matrix LM (Shazeer et al., 2014)	52.9	-
RNN-1024 + MaxEnt 9 Gram Features (Chelba et al., 2013)	51.3	24 GPUs
LSTM-2048-512 (Jozefowicz et al., 2016)	43.7	32 GPUs
2-layer LSTM-8192-1024 (Jozefowicz et al., 2016)	30.6	32 GPUs
BIG GLSTM-G4 (Kuchaiev & Ginsburg, 2017)	23.3*	8 GPUs
LSTM-2048 (Grave et al., 2016a)	43.9	1 GPU
2-layer LSTM-2048 (Grave et al., 2016a)	39.8	1 GPU
GCNN-13	38.1	1 GPU
GCNN-14 Bottleneck	31.9	8 GPUs



$$\mathrm{ReLU}(\mathbf{X}) = \mathbf{X} \otimes (\mathbf{X} > 0)$$

GCNN Is Faster

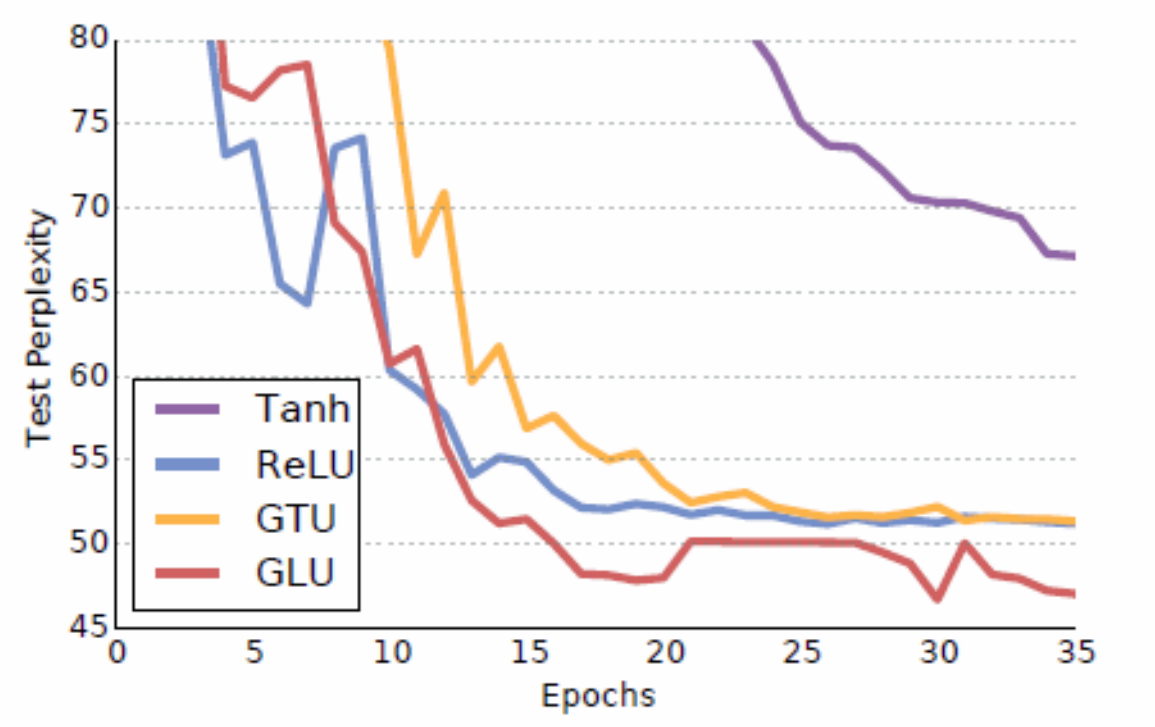
On Google Billion Words



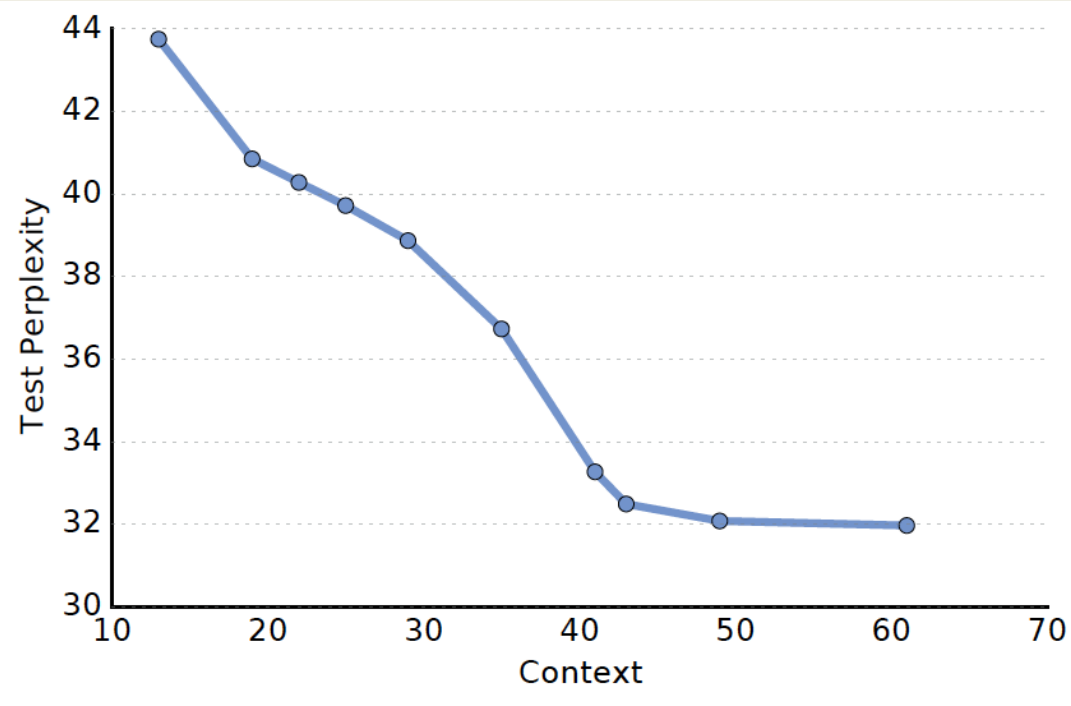
Benchmark: WikiText-103

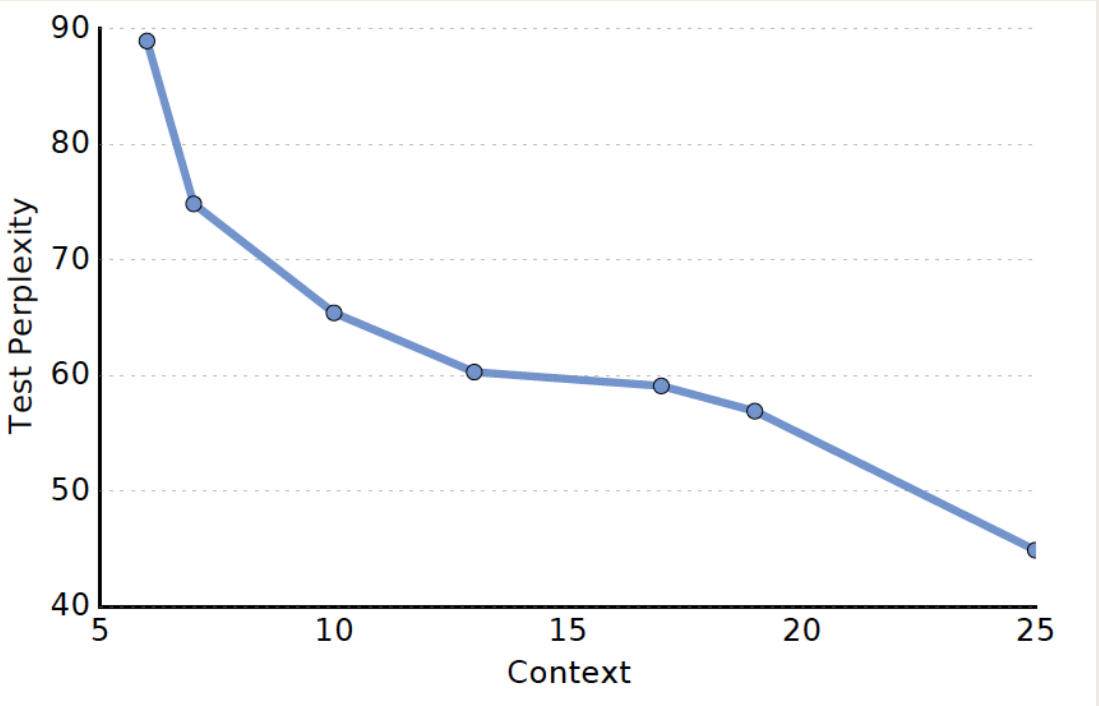
Average Sequence Length = 4,000 Words

Model	Test PPL	Hardware
LSTM-1024 (Grave et al., 2016b)	48.7	1 GPU
GCNN-8	44.9	1 GPU
GCNN-14	37.2	4 GPUs



Short Context Size Suffices





Google Billion Word

Avg. Text Length = 20

Wiki-103

Avg. Text Length = 4,000

Summary

- GCNN: CNN + Gating
- Perplexity is comparable with the state-of-the-art LSTM
- GCNN converges faster and allows parallelization over sequential tokens
- The simpler linear gating (GLU) works better than LSTM-like tanh gating (GTU)