Lecture 4: Recurrent Neural Networks

课程：机器学习与深度学习

Overview

- Sequence Models
  - Language models
- RNNs
- RNN Extensions
  - Bidirectional RNN
  - LSTM
  - GRU

Sequence Models

- Speech recognition
- Music generation
- Sentiment classification
- DNA sequence analysis
- Video comprehension

Language Models

Language Models

- A language model computes a probability for a sequence of words: $P(w_1, w_2, \ldots, w_T)$
  - Assigning high probabilities to well formed sentences
  - Use for machine translation
    - Word ordering: $p(\text{the baby is coming}) > p(\text{the coming is baby})$
    - Word choice: $p(\text{go to restaurant for food}) > p(\text{go to hospital for food})$
  - Use for speech recognition
  - ……

Probability Language Models

- n^{th} order *Markov Assumption*

$$P(w_1, w_2, \dots, w_m) = \prod_{t=1}^{m} p(w_t | w_1, \dots, w_{t-1}) \approx \prod_{i}^{m} p\left(w_t | w_{t-(n-1)}, \dots, w_{t-1}\right)$$

  - $w_1, \dots, w_{t-1}$ are referred as the context

    e.g. $p(\text{I like deep learning}) = p(\text{I}) \cdot p(\text{like}|\text{I}) \cdot p(\text{deep}|\text{I, like}) \cdot p(\text{learning}|\text{I, like, deep})$

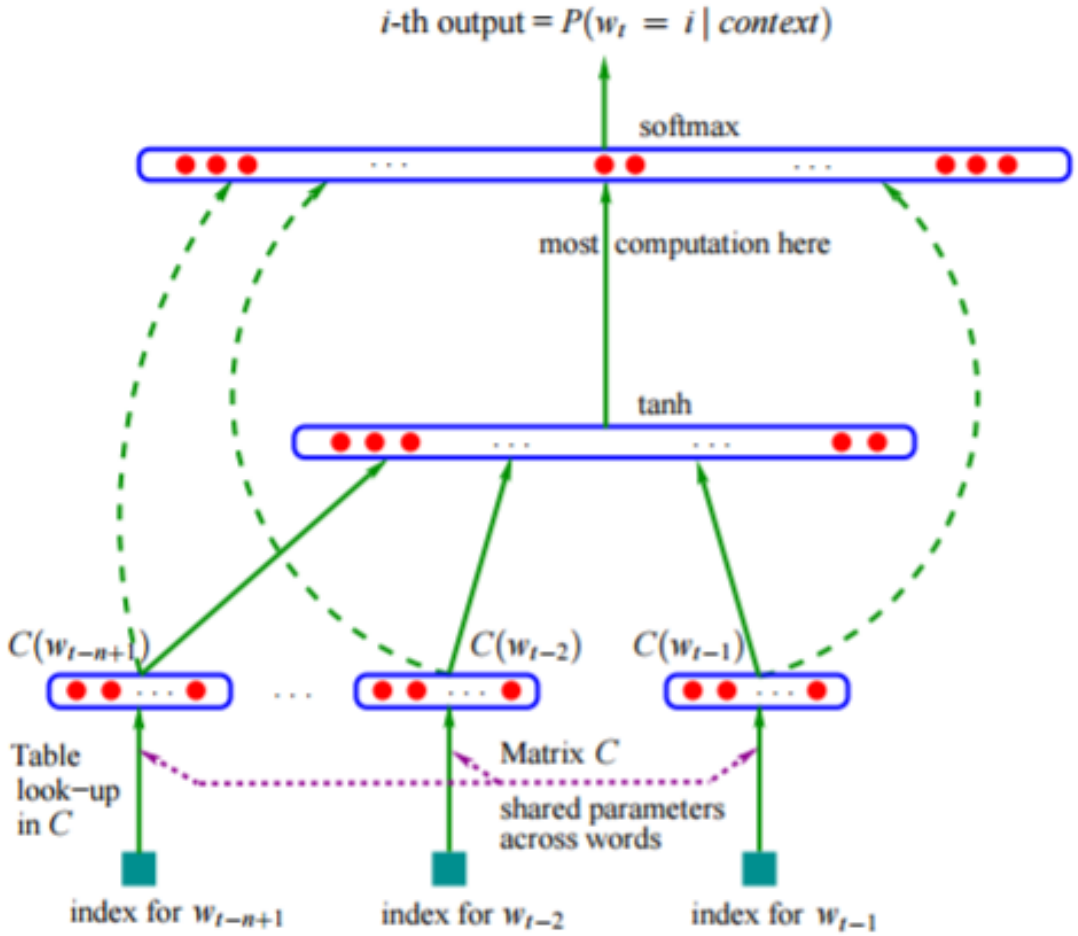
- Estimate conditional probabilities
  - $p(w_2|w_1) = \frac{count(w_1, w_2)}{count(w_1)}$
  - $p(w_3|w_1, w_2) = \frac{count(w_1, w_2, w_3)}{count(w_1, w_2)}$

Probability Language Models

- Performance improves with keeping around higher n-grams counts and doing smoothing and so-called backoff (e.g. if 4-gram not found, try 3-gram, etc)
- There are **A LOT** of n-grams
  - Recent state of the art: Scalable Modified Kneser-Ney Language Model Estimation by Heafield et al., [2013]: “Using one machine with 140 GB RAM for 2.8 days, we built an unpruned model on 126 billion tokens”

Neural Language Model

- Model the conditional distributions with a neural network $p\left(w_t \middle| w_{t-(n-1)}, \dots, w_{t-1}\right)$
  - learn word representations to allow transfer to n-grams not observed in training corpus
  - $\hat{y} = \text{softmax}(W_2 \tanh(W_1 x + b_1) +$



Bengio, Ducharme,Vincent and Jauvin, 2003

Word Representation

- One-hot encoding, e.g. $e(w = 3) = [0,0,1,0,0,0, \dots]$
  - Discrete representation
  - Makes no assumption of word similarity
  - High dimensionality
    - Vulnerability to overfitting
    - Computational expensive

Word Representation by Their Context

- Continuous representation of words
  - Each word w is associated with a real-valued vector $C(w)$
  - The distance $\| C(w_1) - C(w_2) \|$ to reflect meaningful word similarities
  - "word embedding"
- A word is meaningful only when it is with context words

…government debt problems turning into	***banking***	*crises as happened in 2009…*
…saying that Europe needs unified	***banking***	*regulation to replace the hodgepodge…*
…India has just given its	***banking***	*system a shot in the arm…*

Continuous Representation of Words

- Learning representations by back-propagating errors (Rumelhart, Hinton and Williams, 1986)
- A neural probabilistic language model (Bengio et al., 2003)
- Natural Language Processing (Almost) from Scratch (Collobert et al., 2011)
- Efficient Estimation of Word Representations in Vector Space (Mikolov et al., 2013)
- Distributed Representations of Words and Phrases and their Compositionality (Mikolov et al. 2013)
- word2vec Explained: Deriving Mikolov et al.'s Negative-Sampling Word-Embedding Method (Goldberg and Levy, 2014)
- GloVe: Global Vectors for Word Representation (Pennington et al., 2014)
- …

Continuous Representation of Words

- Representing a word by means of its neighbors
  - “You shall know a word by the company it keeps”
  - One of the most successful ideas of modern statistical NLP
- Two types of methods
  - Iteration-based methods, e.g., Word2Vec
  - SVD-based methods, use co-occurrence matrix

Word2Vec: Iteration Based Method

- Two algorithms
- Two training methods: negative sampling and hierarchical sampling

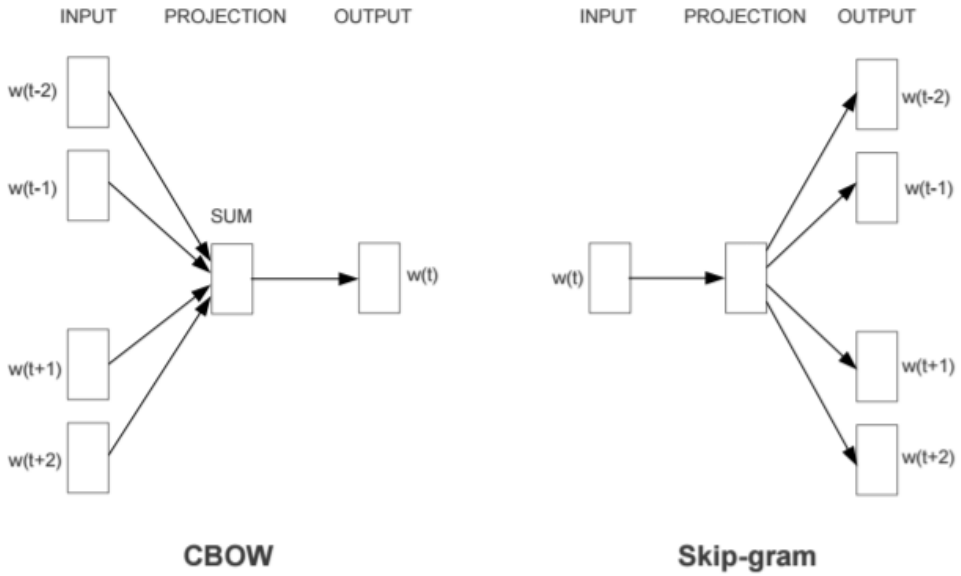
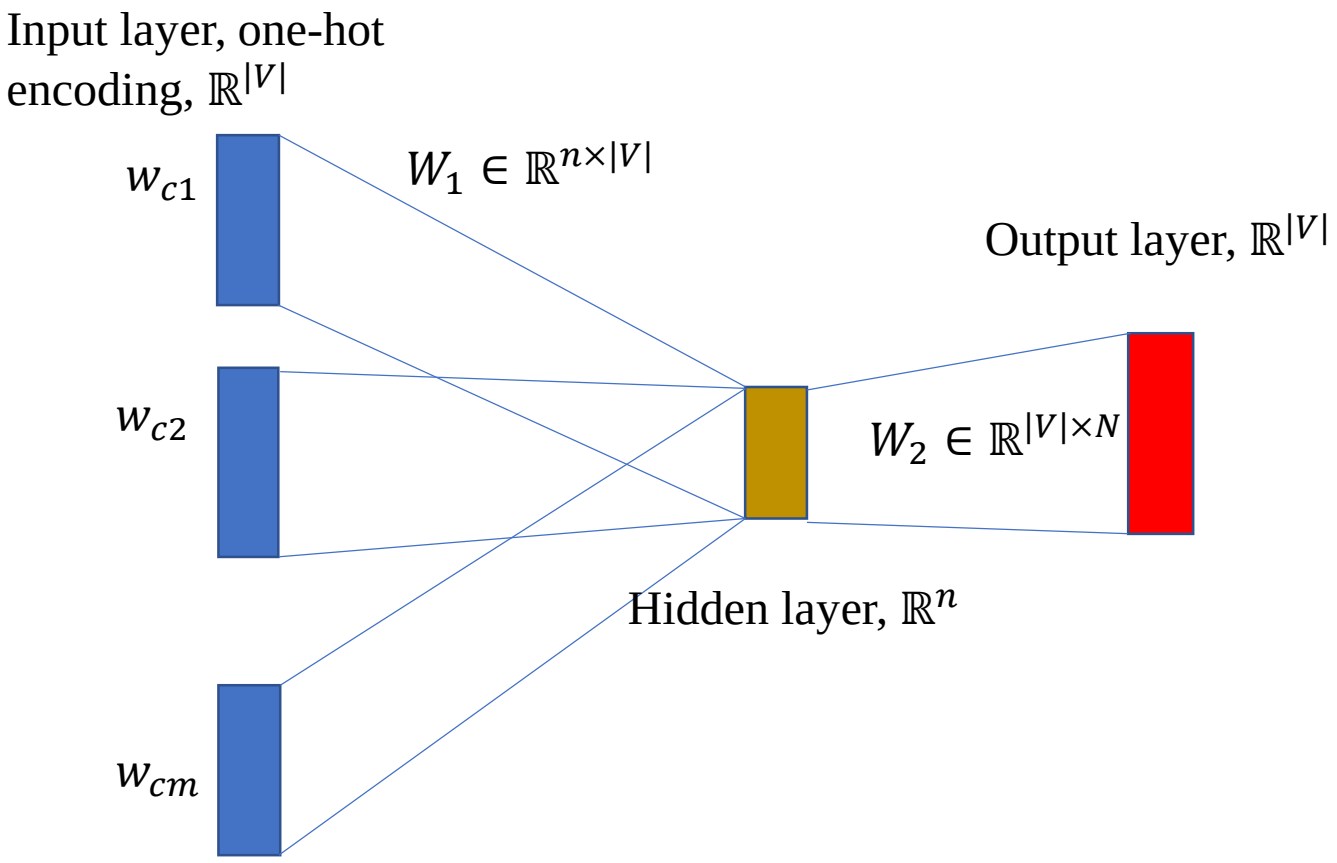


Table 3: *Comparison of architectures using models trained on the same data, with 640-dimensional word vectors. The accuracies are reported on our Semantic-Syntactic Word Relationship test set, and on the syntactic relationship test set of [20]*

Model Architecture	Semantic-Syntactic Word Relationship test set		MSR Word Relatedness Test Set [20]
	Semantic Accuracy [%]	Syntactic Accuracy [%]	
RNNLM	9	36	35
NNLM	23	53	47
CBOW	24	64	61
Skip-gram	55	59	56

CBOW: Continuous Bag-of-Words Model

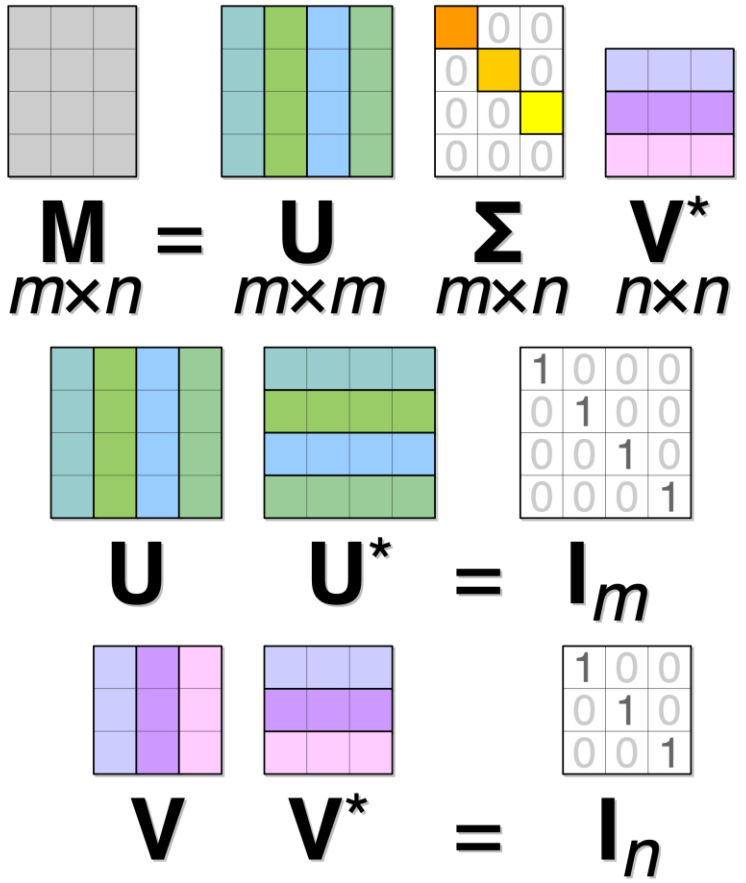


Step1: input encoding for every context word
Step2: multiply embedded word vectors for the context words $W_1 w_{c*}$
Step3: average over the vectors obtained from step2 (hidden layer)
Step4: $\hat{y} = softmax(\mathrm{W}_2\mathrm{h})$

Co-occurance Matrix and SVD

- I like deep learning.
- I like NLP.
- I enjoy flying.

counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0



GloVe

- Combine the advantages of two kinds of models
  - Global matrix factorization
  - Local context window methods, e.g., word2Vec
- Least square objective instead of cross-entropy
  - $J = \sum \sum_w f(X_{ij})(u_j^T v_i - logX_{ij})$

Model	Dim.	Size	Sem.	Syn.	Tot.
ivLBL	100	1.5B	55.9	50.1	53.2
HPCA	100	1.6B	4.2	16.4	10.8
GloVe	100	1.6B	67.5	54.3	60.3
SG	300	1B	61	61	61
CBOW	300	1.6B	16.1	52.6	36.1
vLBL	300	1.5B	54.2	64.8	60.0
ivLBL	300	1.5B	65.2	63.0	64.0
GloVe	300	1.6B	80.8	61.5	70.3
SVD	300	6B	6.3	8.1	7.3
SVD-S	300	6B	36.7	46.6	42.1
SVD-L	300	6B	56.6	63.0	60.1
CBOW†	300	6B	63.6	67.4	65.7
SG†	300	6B	73.0	66.0	69.1
GloVe	300	6B	77.4	67.0	71.7
CBOW	1000	6B	57.3	68.9	63.7
SG	1000	6B	66.1	65.1	65.6
SVD-L	300	42B	38.4	58.2	49.2
GloVe	300	42B	**81.9**	**69.3**	**75.0**

Continuous Representation of Words

- Word representations as input to a neural network
- Can also be updated by gradient descent
  - $C(w) \leftarrow C(w) - \alpha \nabla_{C(w)} J$

Word	w	$C(w)$
"the"	1	[0.6762, -0.9607, 0.3626, -0.2410, 0.6636]
"a"	2	[0.6859, -0.9266, 0.3777, -0.2140, 0.6711]
"have"	3	[0.1656, -0.1530, 0.0310, -0.3321, -0.1342]
"be"	4	[0.1760, -0.1340, 0.0702, -0.2981, -0.1111]
"cat"	5	[0.5896, 0.9137, 0.0452, 0.7603, -0.6541]
"dog"	6	[0.5965, 0.9143, 0.0899, 0.7702, -0.6392]
"car"	7	[-0.0069, 0.7995, 0.6433, 0.2898, 0.6359]

Neural Language Model

- Can potentially generalize to contexts not seen in training set
  - Example: “the cat is walking in the bedroom” generalize to “a dog was running in a room”
  - Example: Imagine 4-gram [“ the ”, “ cat ”, “ is ”, “ eating ”] is not in training corpus, but [“ the ”, “ dog ”, “ is ”, “ eating ”] is
    - If the word representations of “ cat ” and “ dog ” are similar, then the neural network will be able to generalize to the case of “ cat ”
    - $P(eating|the, cat, is)$

Performance Evaluation

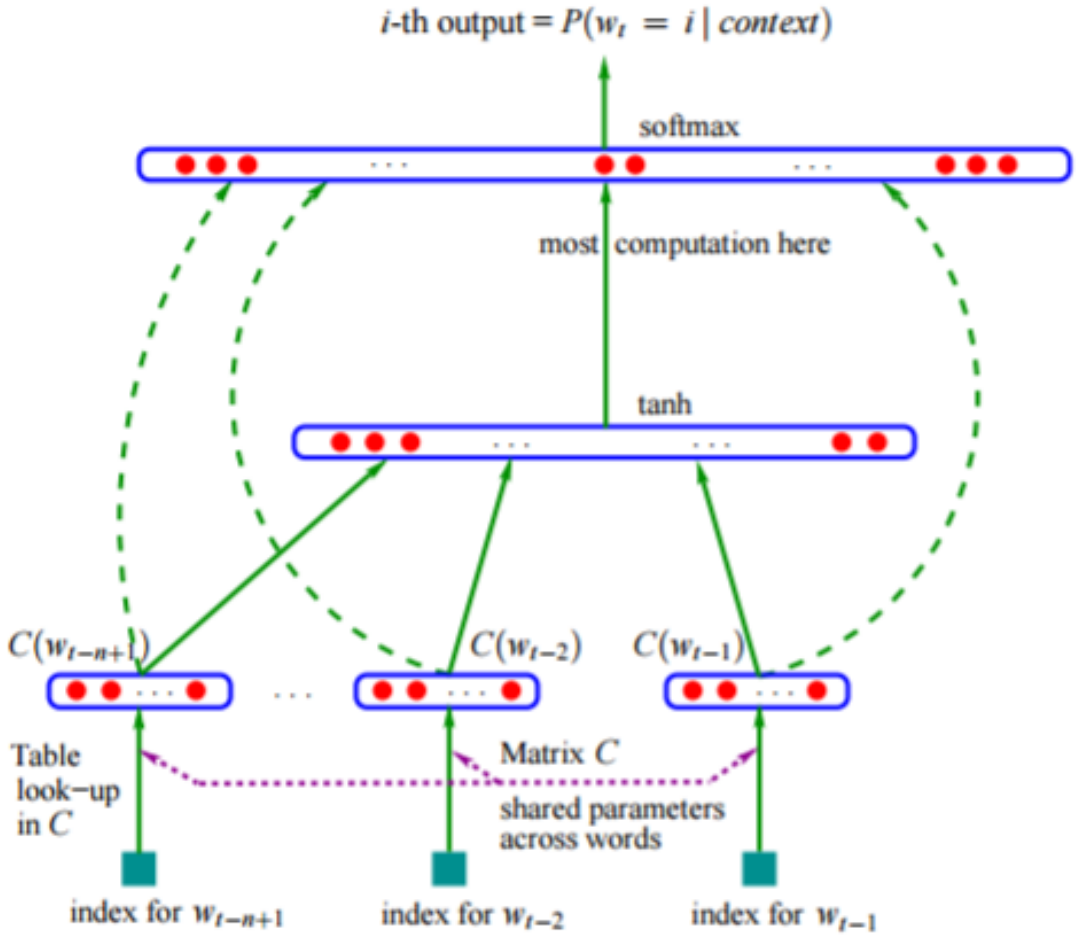
- Perplexity: the exponential of the average negative log-likelihood
  - $2^{-\sum_x p(x) \log_2 p(x)}$
- Evaluation on Brown Corpus
  - n-gram model (Kneser-Ney smoothing): 321
  - neural network language model: 276
  - neural network + n-gram: 252

Bengio, Ducharme,Vincent and Jauvin, 2003

Recurrent Neural Networks (RNN)

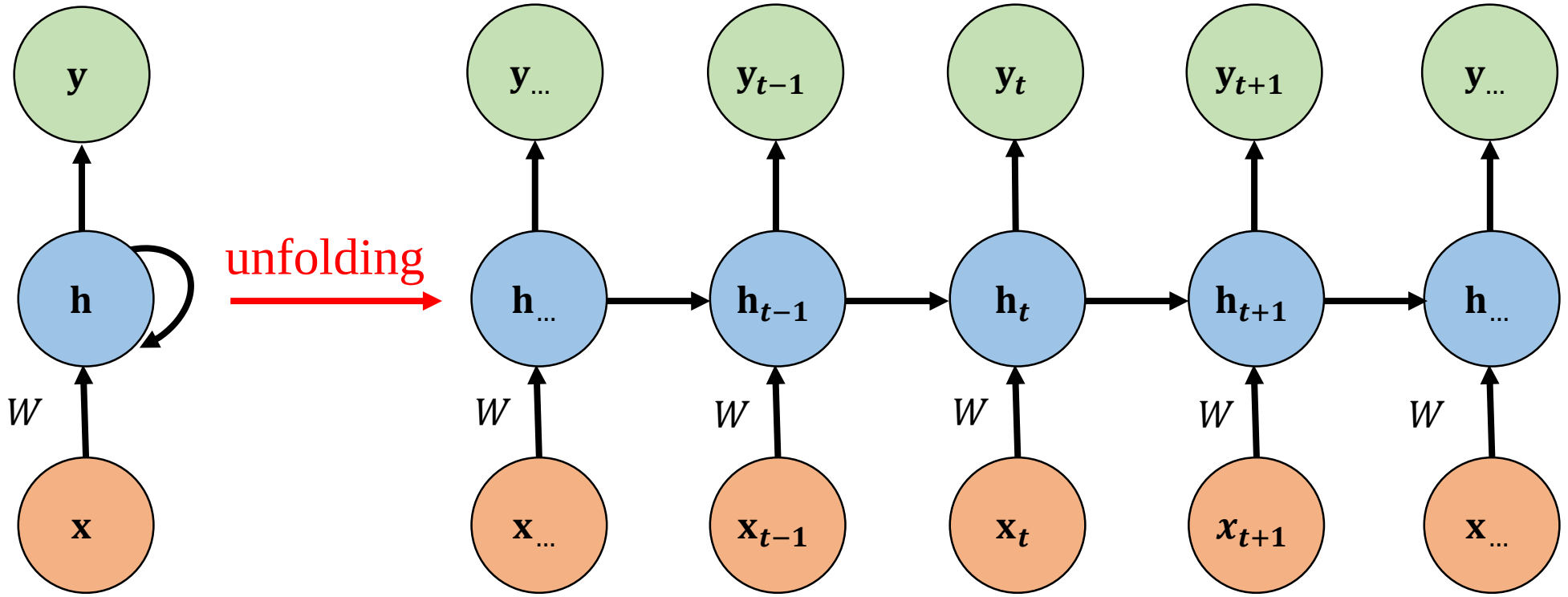
The Problem of Neural Language Model

- Process input of fixed length
- Model size linearly scales with input
- Computation only uses recent information ($n-1$ steps)
- No parameter sharing, only look up table

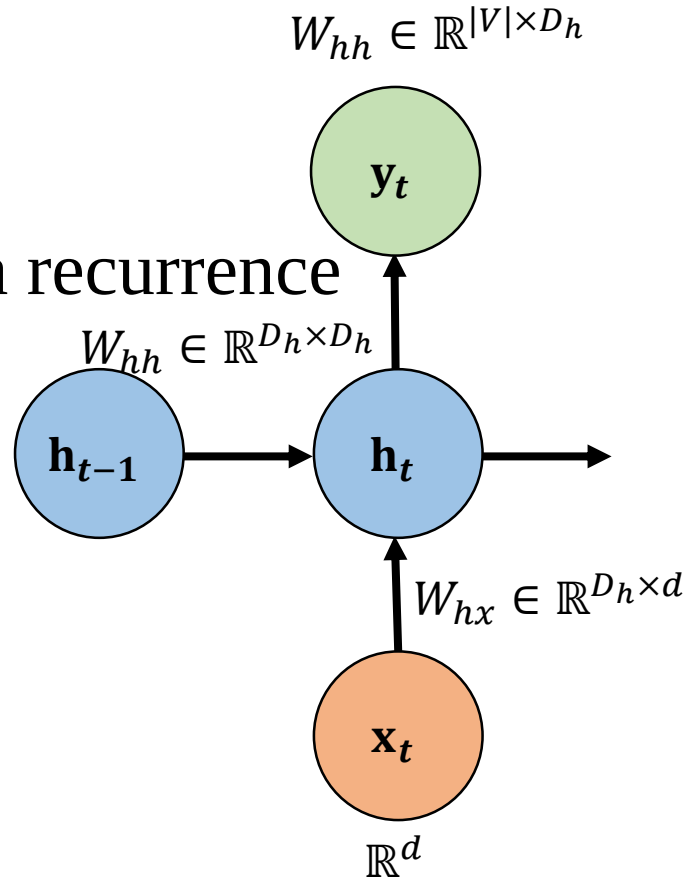


RNN Architecture

- Weights are repeated on the network



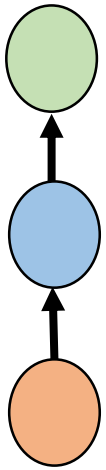
RNN: A Closer Look



- Process a sequence of input vectors **x** by applying a recurrence formula at every time step
- Input vectors: $\mathbf{x_1}, \mathbf{x_2}, \ldots \mathbf{x_t}, \mathbf{x_{t+1}}, \ldots, \mathbf{x_T}$
- At each single time step:
  - $h_t = f_W(h_{t-1}, x_t) = f_{\mathrm{W}}(W_{hh} h_{t-1} + \mathrm{W_{hx}} \mathrm{x_t})$
  - $y_t = f(W_{hy} h_t)$
- Main idea: We use the same set of W weights at all time steps (sharing parameters)

RNN Architecture Types

one to one



one to many

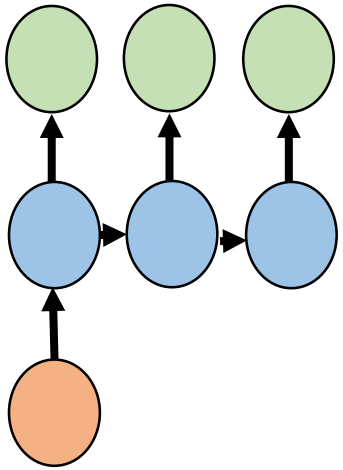
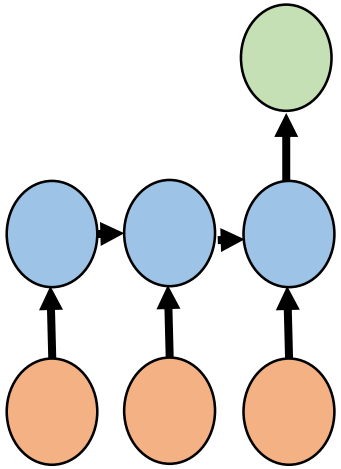


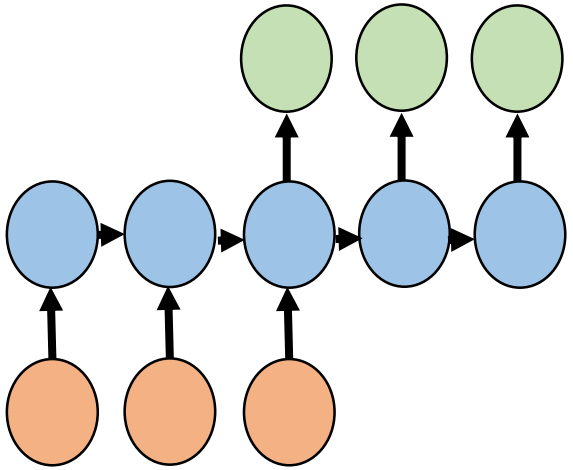
Image captioning: image -> seq. of words

many to one



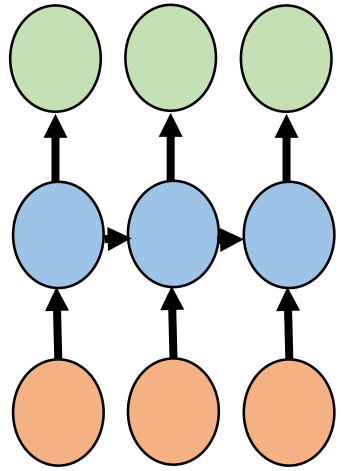
Sentiment classification: seq. of words -> sentiment

many to many



Machine Translation: seq. of words -> seq. of words

many to many

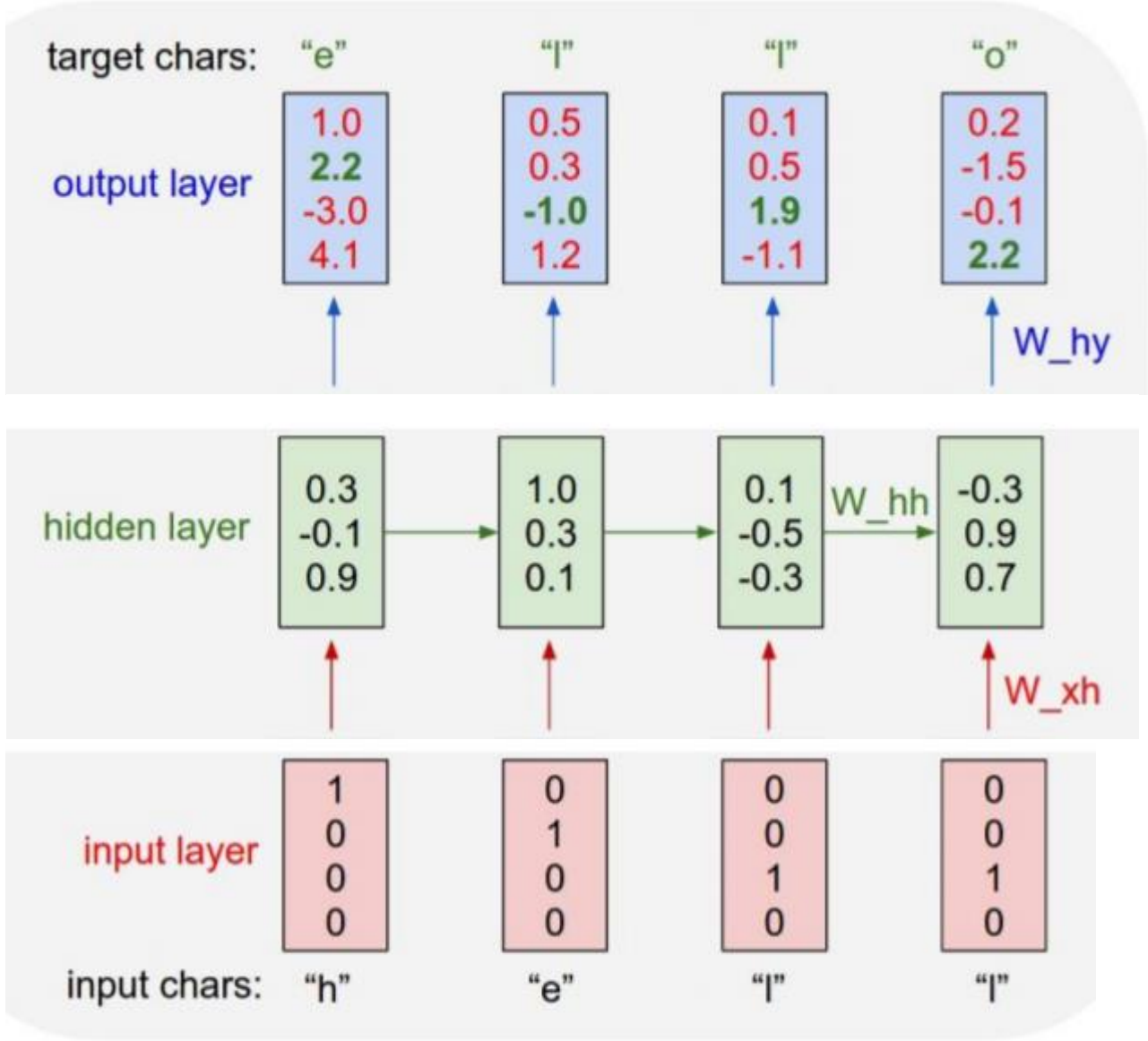


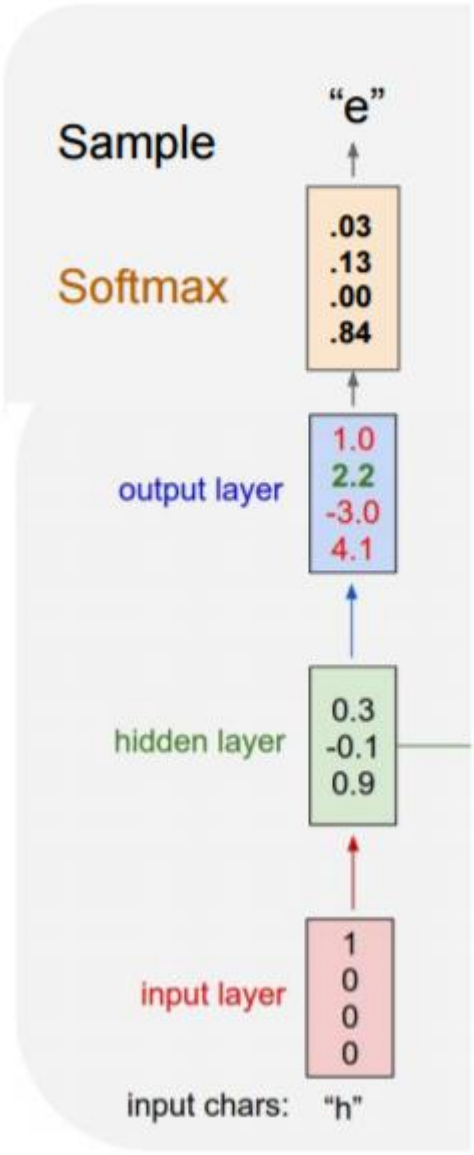
Video classification on frame level

Character-Level Language Model

Vocabulary: [h,e,l,o]

Example training sequence: “hello”

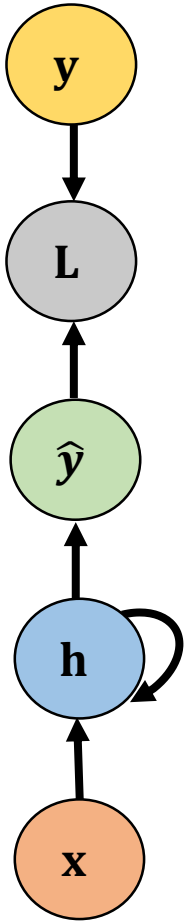




From cs231n: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10.pdf

Training RNN

- Forward propagation: computing loss
  - Recall: input vectors: $\mathbf{x_1}, \mathbf{x_2}, \ldots \mathbf{x_t}, \mathbf{x_{t+1}}, \ldots, \mathbf{x_T}$
  - Output: predicted: $\hat{y}_1 \hat{y}_2 \ldots, \hat{y}_T$
  - Ground-truth: $\mathbf{y_1}, \mathbf{y_2}, \ldots \mathbf{y_t}, \mathbf{y_{t+1}}, \ldots, \mathbf{y_T}$
  - $L = \frac{1}{T} \sum_{t=1}^{T} y_t log p_{model}(\widehat{y_t})$ (cross − entroy error)
- Backward propagation through time BPTT



- Training RNN is hard for two reasons
  1. Vanishing and exploding gradient problems
     - [Pascanu et al, 2013, On the difficulty of training recurrent neural networks]
  2. RNN is inherently sequential, can’t compute the states for different parts of the sequence in parallel.

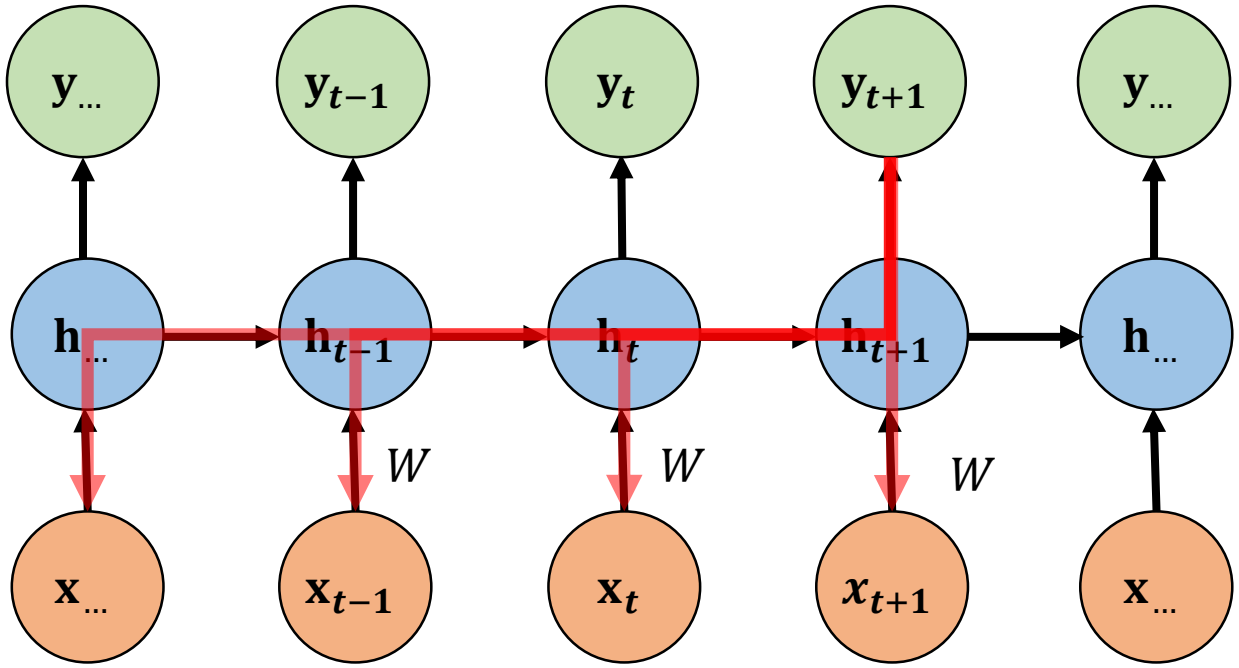
The Vanishing and Exploding Gradient

- In the case of language modeling or question answering words from time steps far away are not taken into consideration when training to predict the next word
- Example:
  - Jane walked into the room. John walked in too. It was late in the day. Jane said hi to _John
  - Jane walked into the room. John walked in too. It was late in the day, and everyone was walking home after a long day at work. Jane said hi to _John

Backpropagation Through Time

$$\frac{\partial L}{\partial W} = \sum_t \frac{\partial L_t}{\partial W}$$

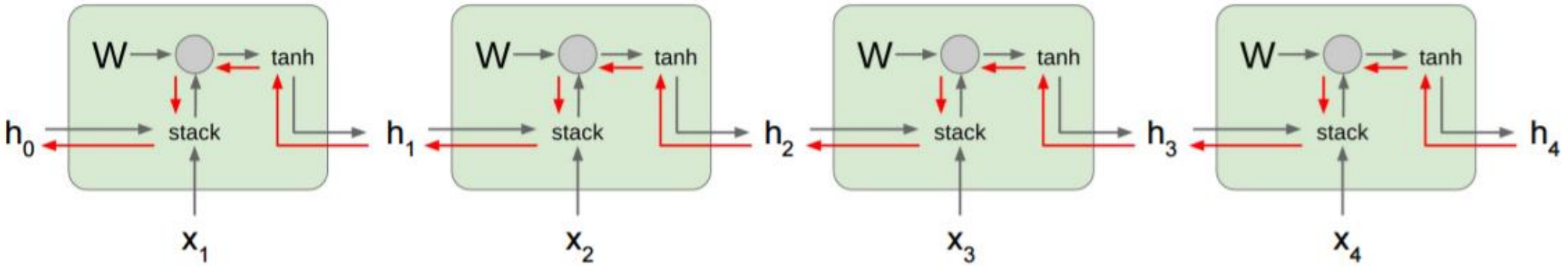
$$= \sum_t \frac{\partial L_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \left(\sum_k \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial W} \right)$$



The Vanishing and Exploding Gradient

- $\frac{\partial L}{\partial W} = \sum_t \frac{\partial L_t}{\partial W} = \sum_t \frac{\partial L_t}{\partial y_t}\frac{\partial y_t}{\partial h_t}\left(\sum_k \frac{\partial h_t}{\partial h_k}\frac{\partial h_k}{\partial W}\right)\frac{\partial h_t}{\partial h_k} = \prod_{j=k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}}$
- $\| \frac{\partial h_t}{\partial h_k} \| \leq (\beta_W \beta_h)^{t-k}$
- $\beta's$ as the upper bound of the corresponding norms
- Thus, the gradients can become very large and small quickly [bengio et al., 1994], and the locality assumptions of gradient descent breaks down
  - The error at a time step ideally can tell a previous time step from many steps away to change during backprop
  - Weight initialization $U[-\frac{\sqrt{3}}{\sqrt{n}}, \frac{\sqrt{3}}{\sqrt{n}}]$

RNN Gradient Flow



Computing gradient of h_0 involves many factors of W (and repeated tanh)

Largest singular value > 1:
Exploding gradients

Largest singular value < 1:
Vanishing gradients

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10.pdf

Training Tricks: Exploding Gradient

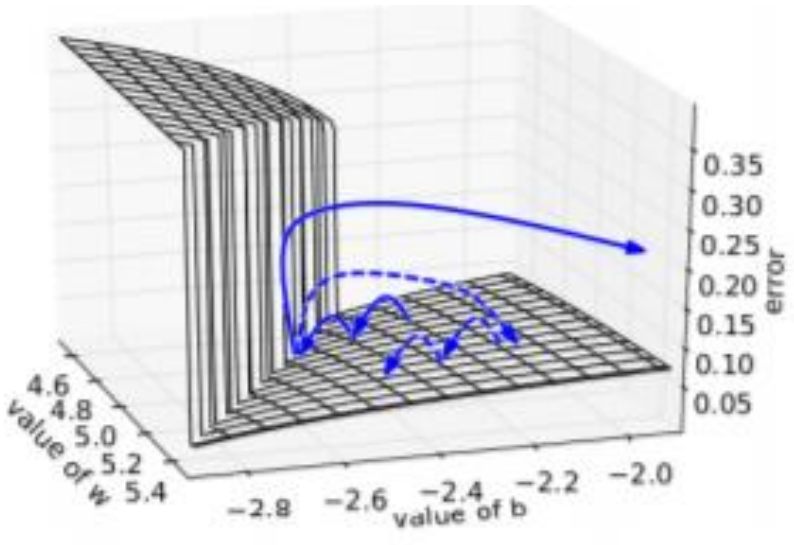
- **Clipping trick**: to clip gradients to a maximum value (first introduced by Mikolov) to address the exploding gradient

  If $||g_t|| \geq threshold$ then

  $$g_t \leftarrow \frac{threshold}{||g_t||} g_t$$

  end if

- L2 or L1 penalty



Training Tricks: Vanishing Gradients

- **Initialization + ReLUs** (to address vanishing gradient)
  - Initialize W_{hh} to identity matrix I
  - Use ReLUs
  - Socher et al. 2013, Le et al. 2015
- Regularizer term [Pascanu et al, 2013] enforces constant backwards error flow
- Replace hidden layers with gated units

$$\Omega = \sum_k \Omega_k = \sum_k \left(\frac{\left\| \frac{\partial \mathcal{E}}{\partial \mathbf{x}_{k+1}} \frac{\partial \mathbf{x}_{k+1}}{\partial \mathbf{x}_k} \right\|}{\left\| \frac{\partial \mathcal{E}}{\partial \mathbf{x}_{k+1}} \right\|} - 1 \right)^2$$

$$\begin{aligned} \frac{\partial^+ \Omega}{\partial \mathbf{W}_{rec}} &= \sum_k \frac{\partial^+ \Omega_k}{\partial \mathbf{W}_{rec}} \\ &= \sum_k \frac{\partial^+ \left(\frac{\left\| \frac{\partial \mathcal{E}}{\partial \mathbf{x}_{k+1}} \mathbf{W}_{rec}^T diag(\sigma'(\mathbf{x}_k)) \right\|}{\left\| \frac{\partial \mathcal{E}}{\partial \mathbf{x}_{k+1}} \right\|} - 1 \right)^2}{\partial \mathbf{W}_{rec}} \end{aligned}$$

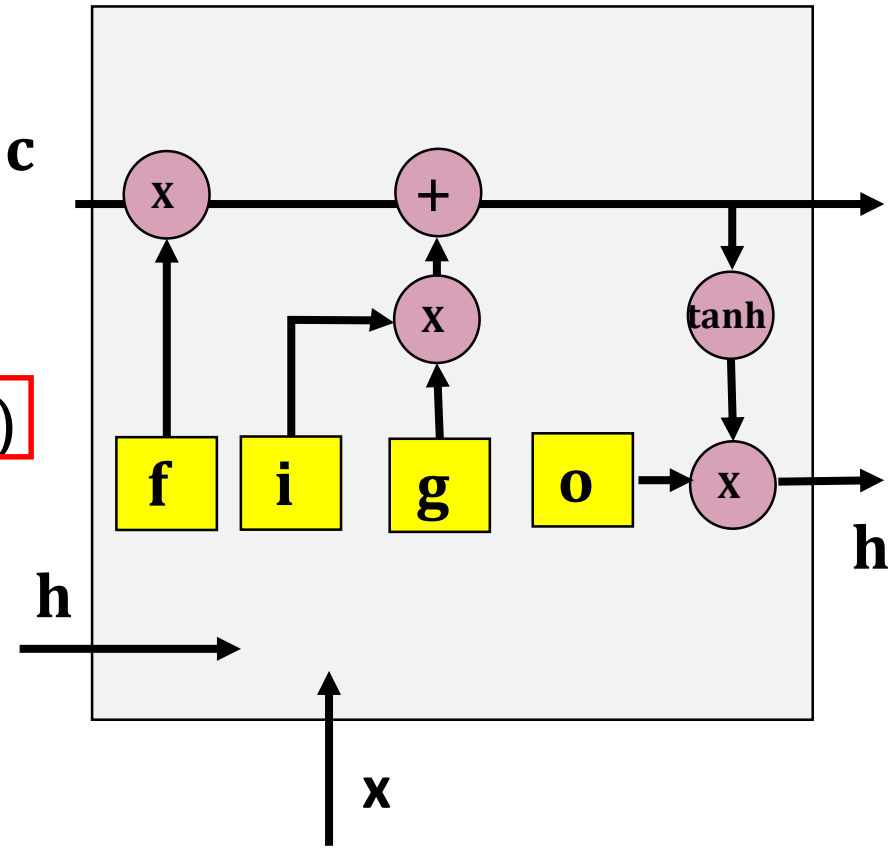
RNN Extensions

Long Short-Term Memories (LSTM)

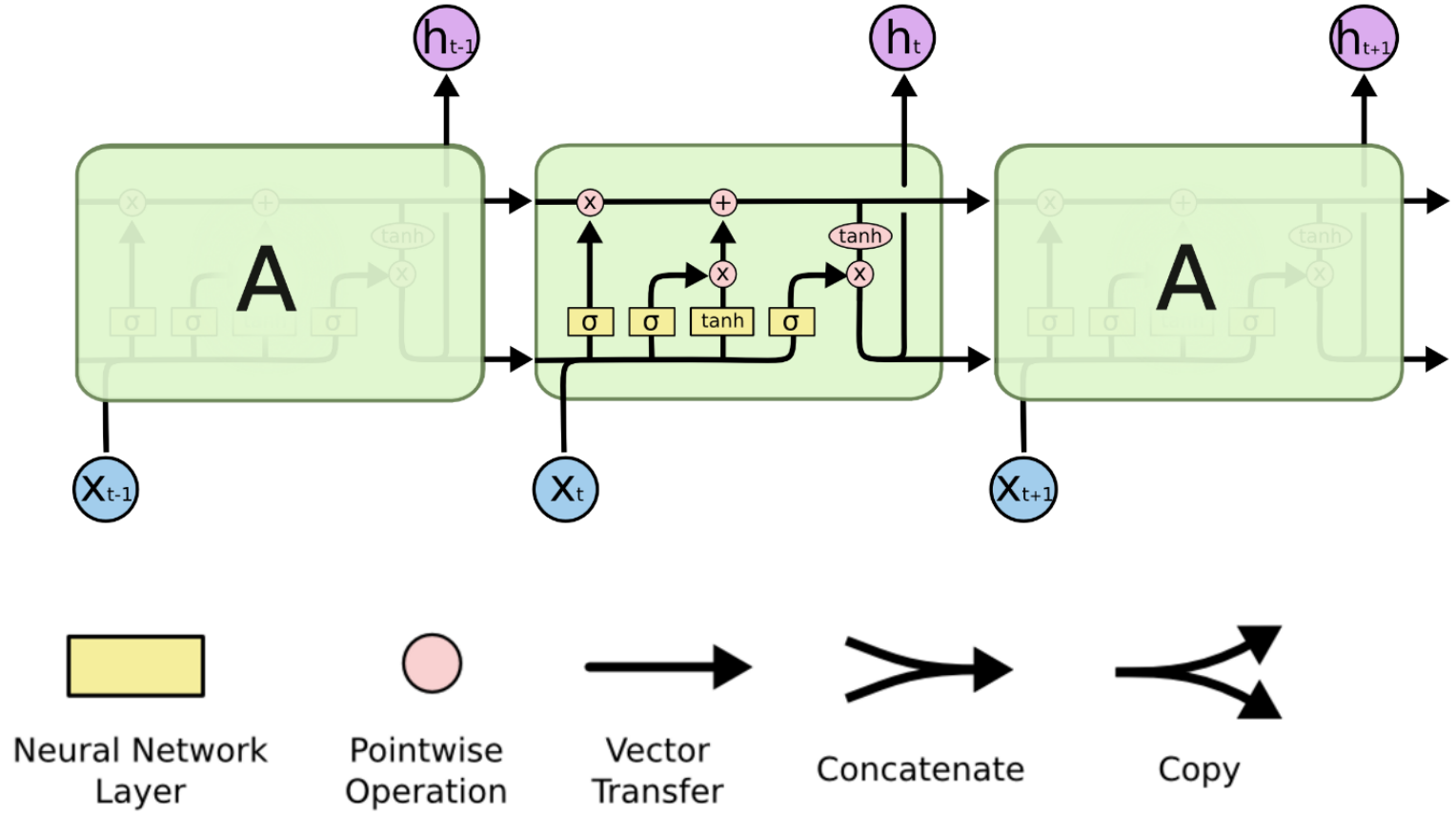
- Hochreiter and Schmidhuber (1997)
- “how can we achieve constant error flow through a single unit with a single connection to itself [i.e., a single piece of isolated information]?”
- To ensure the integrity of our messages in the real world, we write them down
- Gates as a mechanism for selectivity

LSTM

- At each time step to modify
  - Input gate: $i_t = \sigma(W_i x_t + U_i h_{t-1})$
  - Forget gate: $f_t = \sigma(W_f x_t + U_f h_{t-1})$
  - Output gate: $o_t = \sigma(W_o x_t + U_o h_{t-1})$
  - New memory cell: $g_t = \tanh(W_g x_t + U_g h_{t-1})$
- Final memory cell: $c_t = f_t \odot c_{t-1} + i_t \odot g_t)$
- Final hidden state: $h_t = o_t \odot \tanh(c_t)$



LSTM



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

RNN VS. LSTM

Vanilla RNN

$$h_t = \tanh\left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

LSTM

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

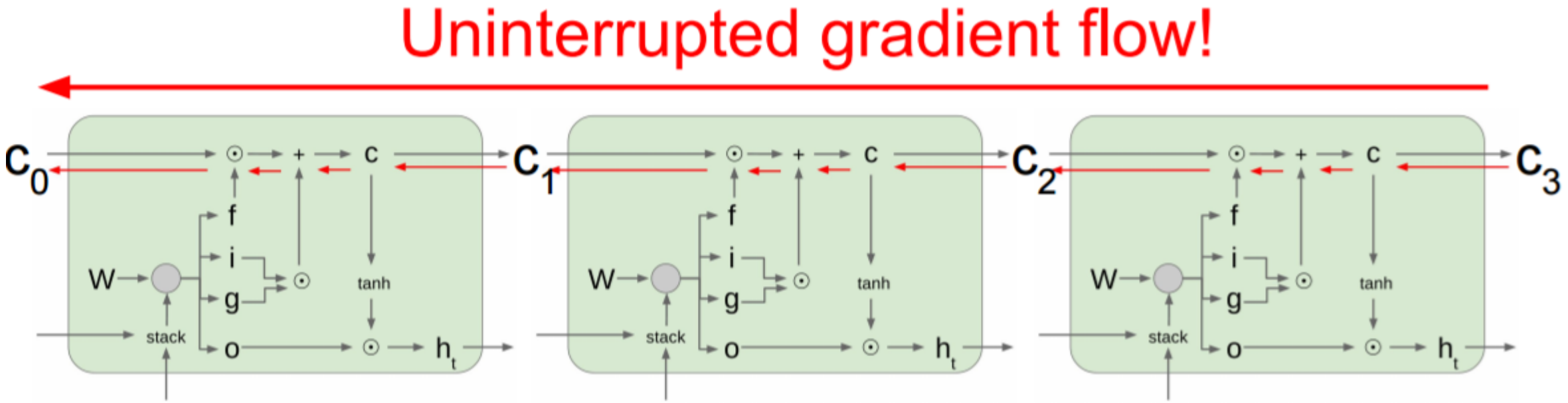
$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10.pdf

LSTM Gradient Flow

http://imgur.com/gallery/vaNahKE



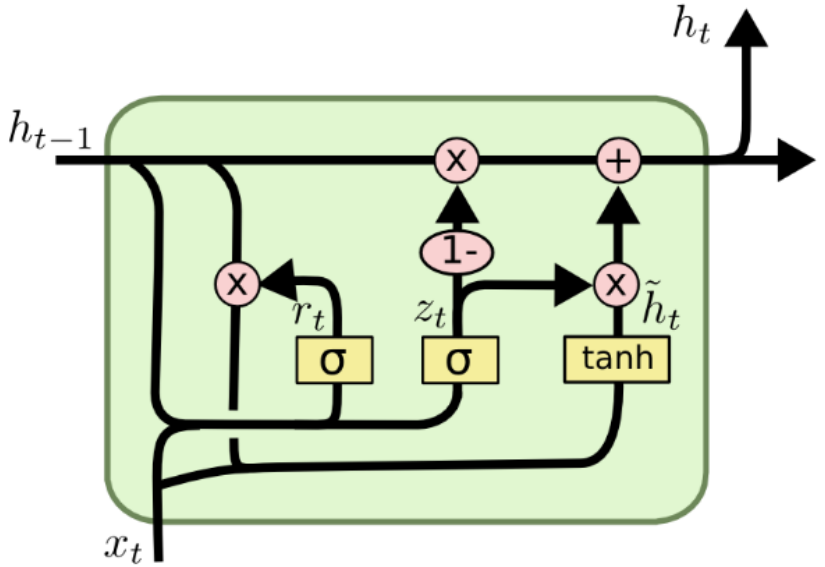
Backpropagation from C_t to C_{t-1} only element-wise multiplication by f, no matrix multiply

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10.pdf

Gated Recurrent Units (GRU)

- Cho et al. (2014)
- A simplified variant of LSTM
- Main idea
  - Keep around memories to capture long distance dependencies
  - Allow error messages to flow at different strengths depending on the inputs
- Reset Gates and Update Gates
  - Vectors with (0,1) entries to perform convex combination of hidden states and others.

GRU



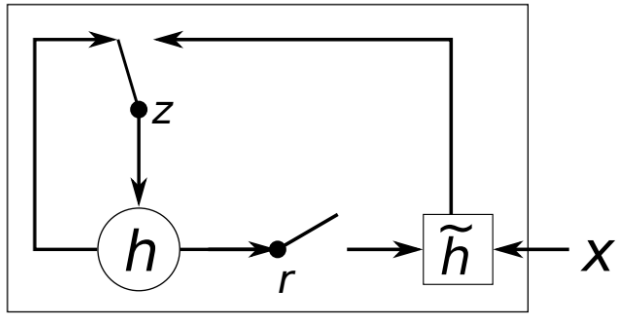
- Update gate
  - $z_t = \sigma(W_{xz}x_t + W_{hz}h_{t-1} + b_z)$
- Reset gate
  - $r_t = \sigma(W_{xr}x_t + W_{hr}h_{t-1} + b_r)$
- Candidate memory content:

$$\tilde{h}_t = \tanh(W_{xh}x_t + W_{hh}(r_t \odot h_{t-1}) + b_h)$$

  - If reset gate ~0, ignores previous memory, only store new information
- Final memory: $h_t = z_t \odot h_{t-1} + (1 - \mathrm{z_t}) \odot \tilde{h}_t$

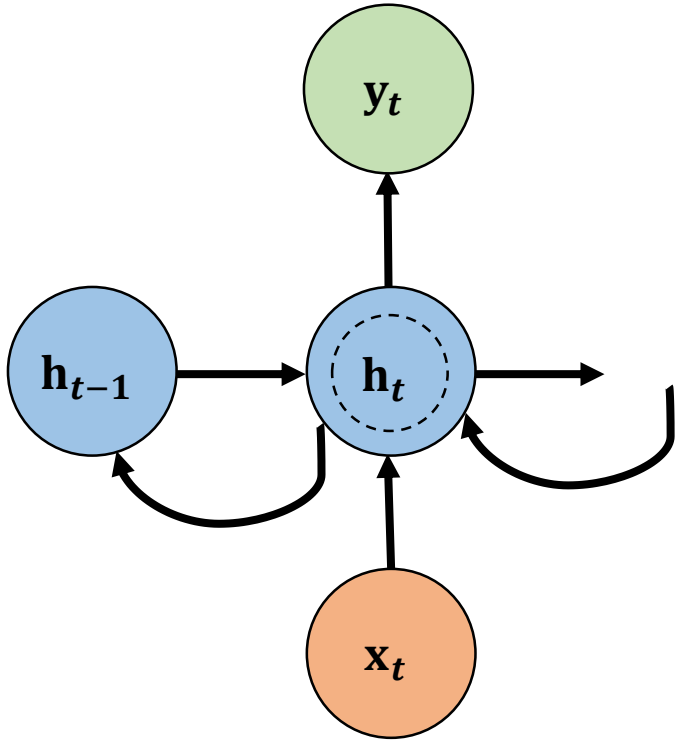
GRU Intuition

- If reset gate r is close to 0, ignore previous hidden status
  - Allows model to drop information that is irrelevant in the future
- Update gate z controls how much of past state should matter now
  - If z close to 1, then we can copy information in that unit through many time steps (less vanishing gradient)
- Units with short-term dependencies often have reset gates very active
- Units with long term dependencies have active update gates



Bidirectional RNNs

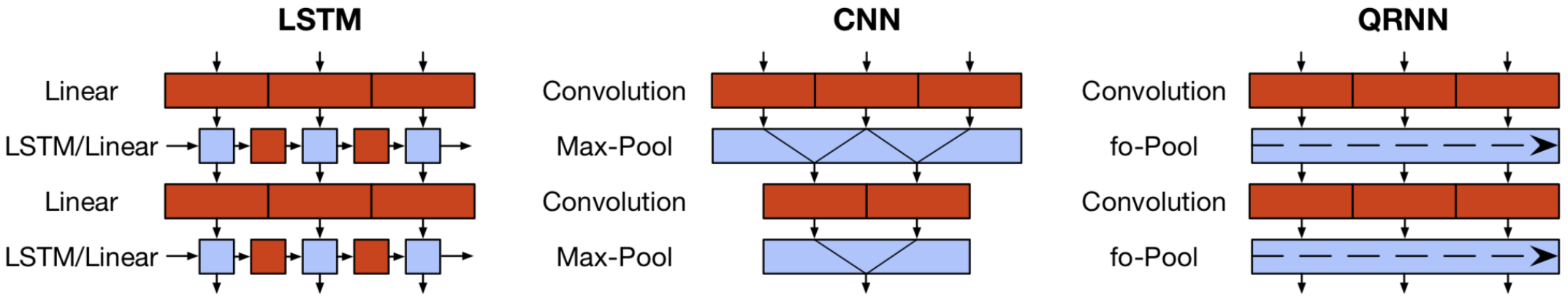
- The one-directional RNN only captures the causal relationship that past affects the present
- However, sometimes we might need to check the future to better explain the current status
  - E.g. opinion mining, sentiment classification
- At each single time step
  - $h_t^{\text{past}} = \text{f}_\text{W}\left(W_{h_{t-1}h_t} h_{t-1} + \text{W}_{\text{xh}}^{\text{past}} \text{x}_\text{t}\right)$
  - $h_t^{\text{future}} = f_W(W_{h_{t+1}h_t} \text{h}_{\text{t+1}} + \text{W}_{\text{xh}}^{\text{future}} \text{x}_\text{t})$
  - $y_t = f(h_t^{\text{past}}, h_t^{\text{future}})$



Training RNN

- Due to their inherently sequential nature, training RNN is difficult.
- Some ideas to accelerate RNN tasks:
  - Make computation in RNN parallelizable
  - Model selection, skip computation and units

Quasi RNN



Parallelize across time-step and minibatch dimensions

$$\mathbf{Z} = \tanh(\mathbf{W}_z * \mathbf{X}) \qquad \mathbf{z}_t = \tanh(\mathbf{W}_z^1 \mathbf{x}_{t-1} + \mathbf{W}_z^2 \mathbf{x}_t)$$

$$\mathbf{F} = \sigma(\mathbf{W}_f * \mathbf{X}) \qquad \mathbf{f}_t = \sigma(\mathbf{W}_f^1 \mathbf{x}_{t-1} + \mathbf{W}_f^2 \mathbf{x}_t)$$

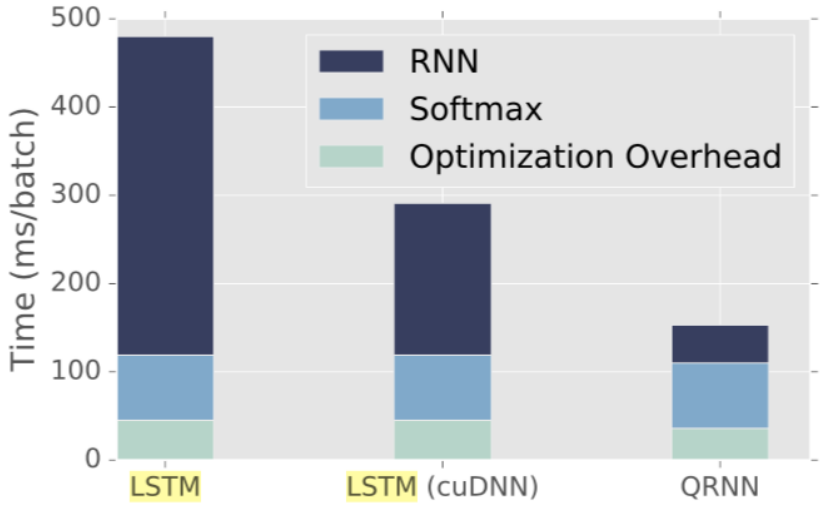
$$\mathbf{O} = \sigma(\mathbf{W}_o * \mathbf{X}), \qquad \mathbf{o}_t = \sigma(\mathbf{W}_o^1 \mathbf{x}_{t-1} + \mathbf{W}_o^2 \mathbf{x}_t).$$

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

Bradbury, J., Merity, S., Xiong, C., & Socher, R. (2016). Quasi-recurrent neural networks. *arXiv preprint arXiv:1611.01576*.

Quasi RNN

Fast and accurate



		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**

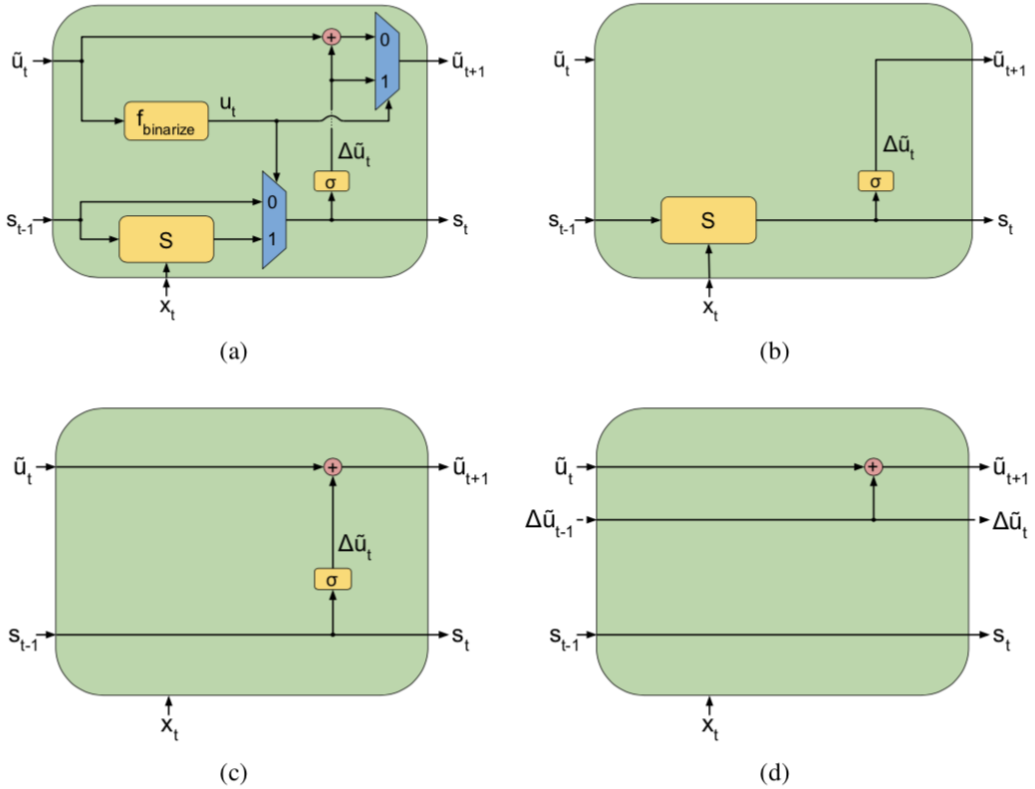
Model	Parameters	Validation	Test
LSTM (medium) (Zaremba et al., 2014)	20M	86.2	82.7
Variational LSTM (medium, MC) (Gal & Ghahramani, 2016)	20M	81.9	79.7
LSTM with CharCNN embeddings (Kim et al., 2016)	19M	–	78.9
Zoneout + Variational LSTM (medium) (Merity et al., 2016)	20M	84.4	80.6
Our models			
LSTM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

Skip RNN

- Learn to skip state updates and shortens the effective size of the computational graph

$$u_t = f_{binarize}(\tilde{u}_t)$$
$$s_t = u_t \cdot S(s_{t-1}, x_t) + (1 - u_t) \cdot s_{t-1}$$
$$\Delta\tilde{u}_t = \sigma(W_p s_t + b_p)$$
$$\tilde{u}_{t+1} = u_t \cdot \Delta\tilde{u}_t + (1 - u_t) \cdot (\tilde{u}_t + \min(\Delta\tilde{u}_t, 1 - \tilde{u}_t))$$

u_t is generated by Bernoulli or rounding

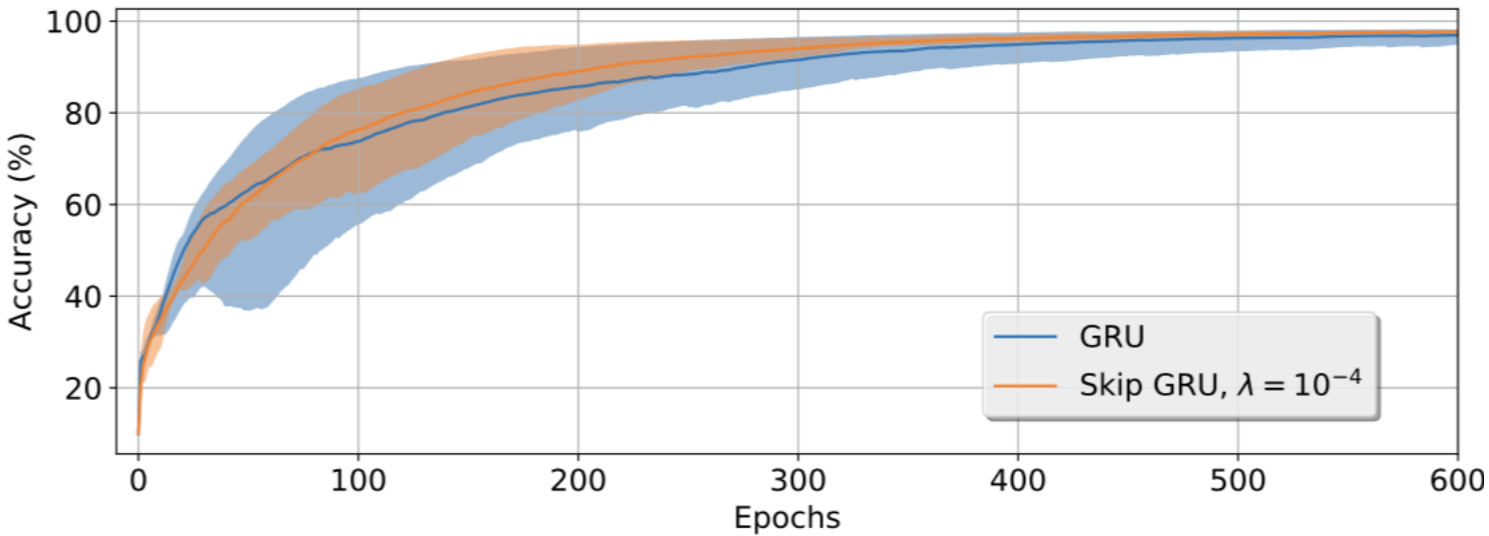


Campos, V., Jou, B., Giró-i-Nieto, X., Torres, J., & Chang, S. F. (2017). Skip rnn: Learning to skip state updates in recurrent neural networks. *arXiv preprint arXiv:1708.06834*.

Skip RNN

- Faster and reduced variance
- Random skip is worse than guided skip

Model	Accuracy	State updates	Inference FLOPs
LSTM	0.910 ± 0.045	784.00 ± 0.00	3.83×10^7
LSTM ($p_{skip} = 0.5$)	0.893 ± 0.003	392.03 ± 0.05	1.91×10^7
Skip LSTM, $\lambda = 10^{-4}$	0.973 ± 0.002	379.38 ± 33.09	1.86×10^7
GRU	0.968 ± 0.013	784.00 ± 0.00	2.87×10^7
GRU ($p_{skip} = 0.5$)	0.912 ± 0.004	391.86 ± 0.14	1.44×10^7
Skip GRU, $\lambda = 10^{-4}$	0.976 ± 0.003	392.62 ± 26.48	1.44×10^7



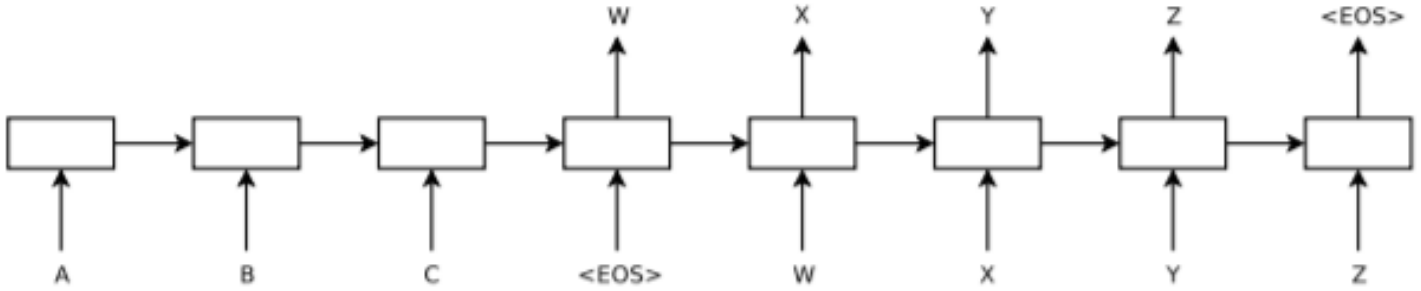
Summary

- RNNs allow a lot of flexibility in architecture design
- Vanilla RNNs are simple but don’t work very well because of vanishing/exploding gradients
- Gradient exploding is controlled with gradient clipping; vanishing is controlled with additive interactions (LSTM)
- Better/simpler architectures are a hot topic of current research
- Better understanding (both theoretical and empirical) is needed

RNN Applications

Machine Translation

- Translation model + Language model
- Sequence to sequence modeling (seq2seq)



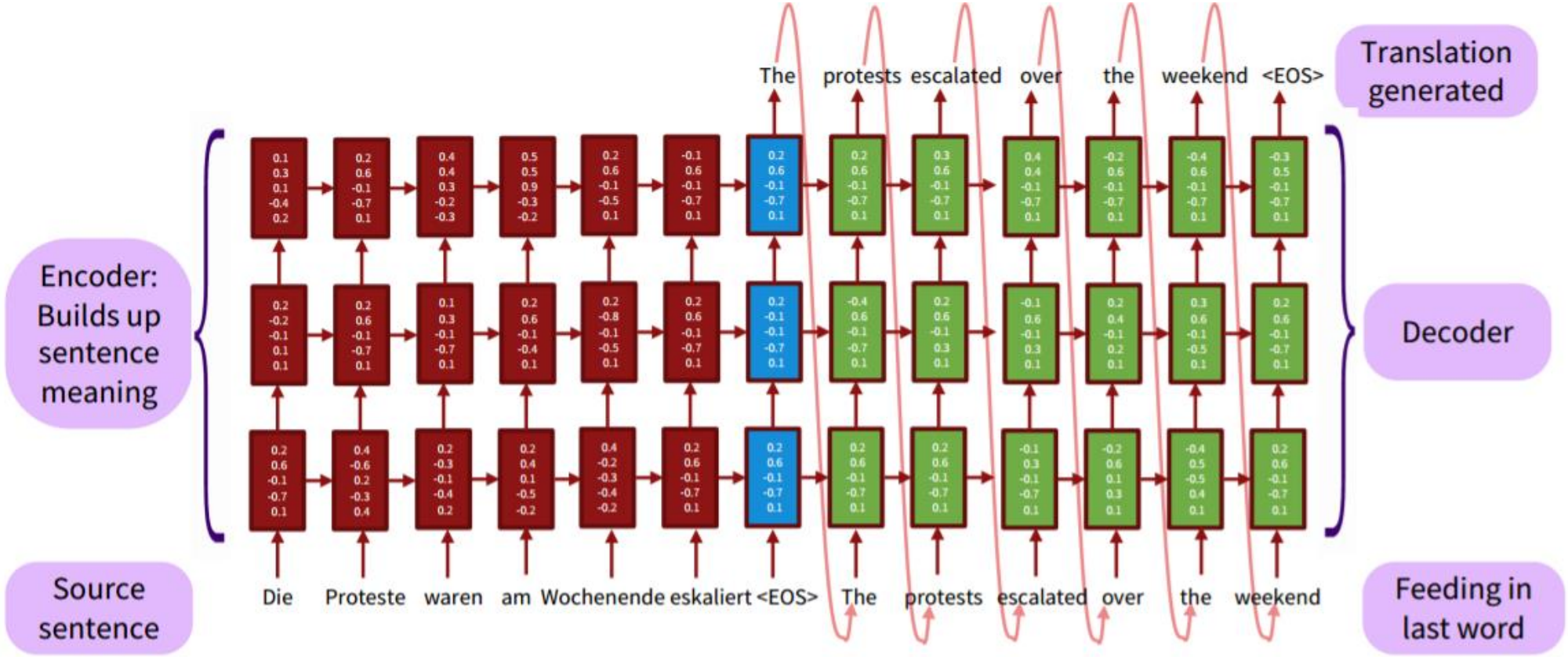
I love deep learning!

Amo el aprendizaje profundo!

Sutskever et al. 2015, Sequence to Sequence Learning with Neural Networks
Johnson et al., 2016, Google's Multilingual Neural Machine Translation System: Enabling Zero-shot Translation

Machine Translation: Seq2Seq

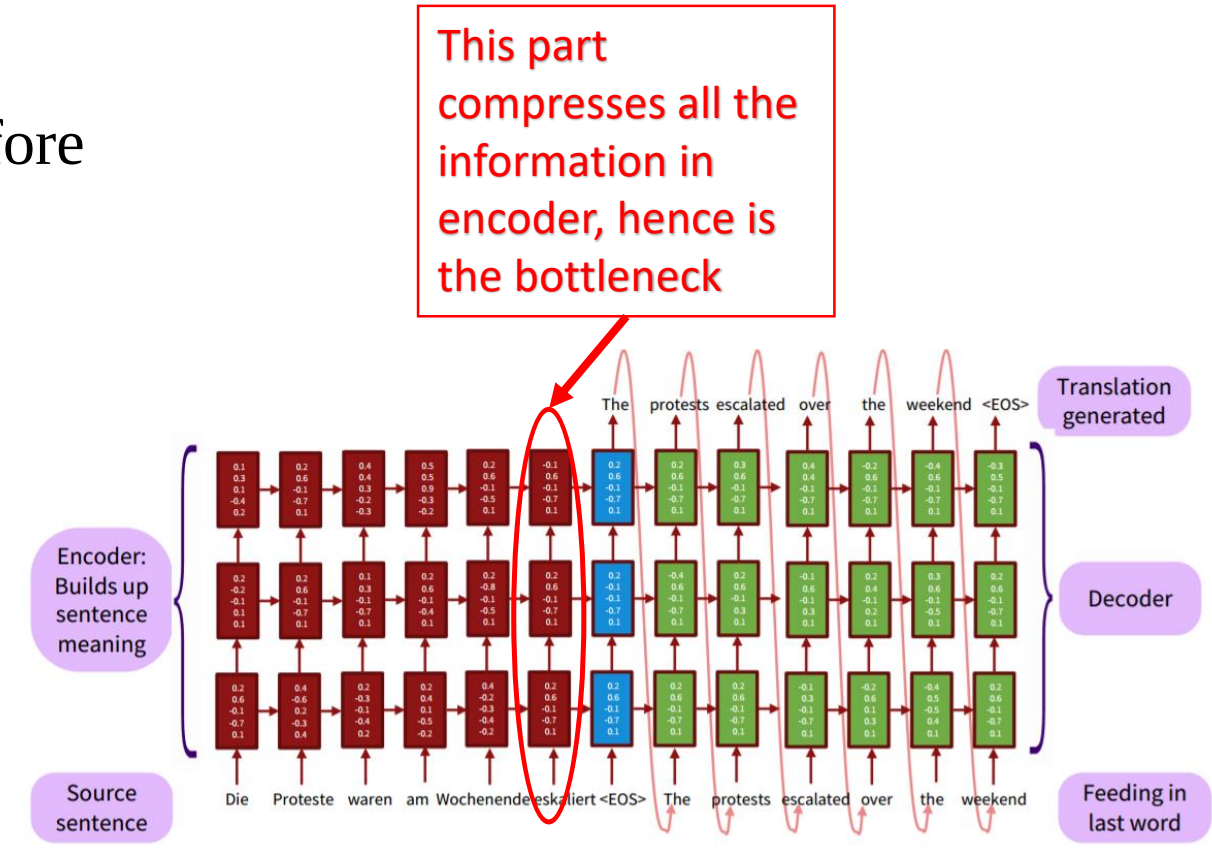
- An encoder: encodes input sequence into a fixed-sized context vectors
- A decoder: use context vector as input to generate an output sequence



http://web.stanford.edu/class/cs224n/lectures/cs224n-2017-lecture10.pdf

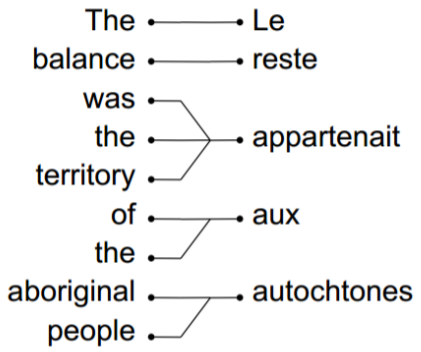
Seq2Seq

- Pros:
  - Better use of context: see enough before translating
- Cons:
  - Encoder has to encapsulate all the information, becoming information bottleneck



Machine Translation + Attention Mechanism

- Encoder: bidirectional RNN
- Decoder: different context vectors
  - The first word of output based on the first few words
  - The last word of output based on the last few words
  - $s_i = f(s_{i-1}, y_{i-1}, c_i)$
  - Similar to “alignment” in phrase-based MT



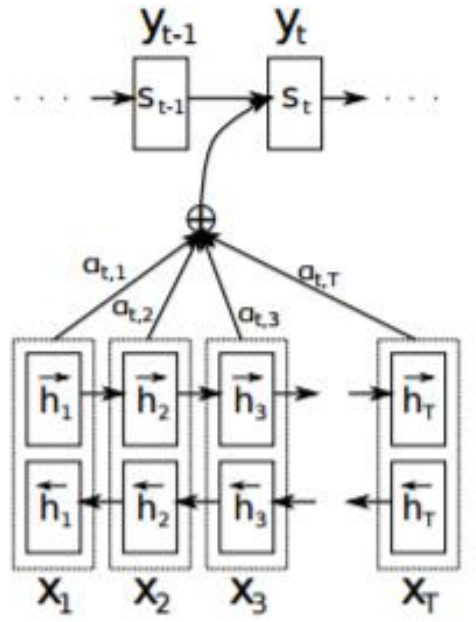


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.

Bahdanau et al., 2014, Neural Machine Translation by Jointly Learning to Align and Translate
Luong et al., 2015, Effective Approaches to Attention-based Neural Machine Translation

Machine Translation + Attention Mechanism

Connect the encoder to focus on a particular part of the source sequence

c_t: a context vector to average all source states according to the weight a_t

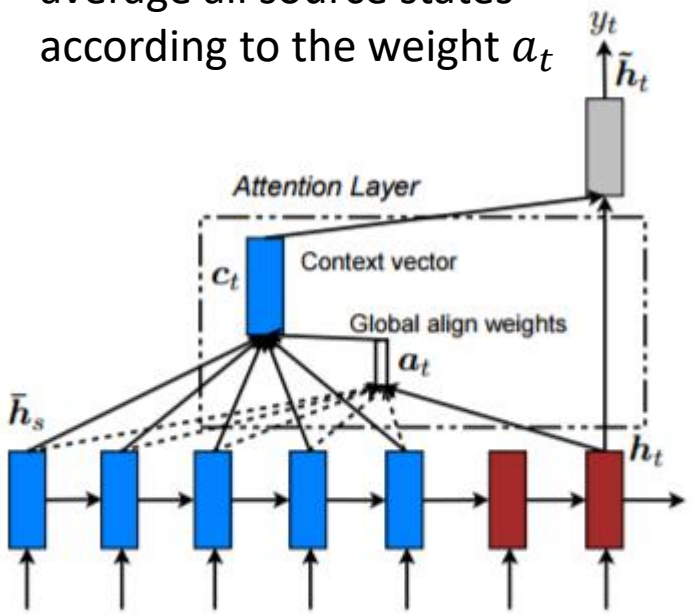


Figure 2: **Global attentional model** – at each time step t, the model infers a *variable-length* alignment weight vector $\boldsymbol{a}_t$ based on the current target state $\boldsymbol{h}_t$ and all source states $\bar{\boldsymbol{h}}_s$. A global context vector $\boldsymbol{c}_t$ is then computed as the weighted average, according to $\boldsymbol{a}_t$, over all the source states.

p_t: predict aligned position

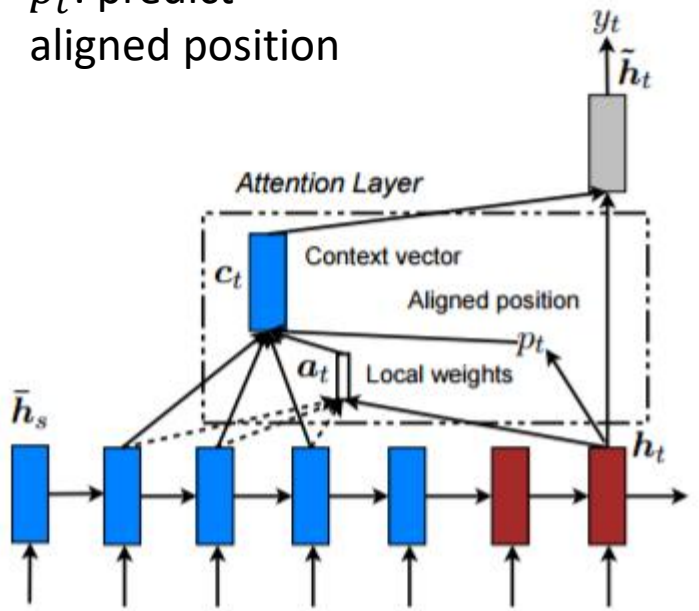


Figure 3: **Local attention model** – the model first predicts a single aligned position p_t for the current target word. A window centered around the source position p_t is then used to compute a context vector $\boldsymbol{c}_t$, a weighted average of the source hidden states in the window. The weights $\boldsymbol{a}_t$ are inferred from the current target state $\boldsymbol{h}_t$ and those source states $\bar{\boldsymbol{h}}_s$ in the window.

Machine Translation + Attention Mechanism

System	Ppl	BLEU
Winning WMT'14 system – *phrase-based + large LM* (Buck et al., 2014)		20.7
Existing NMT systems		
RNNsearch (Jean et al., 2015)		16.5
RNNsearch + unk replace (Jean et al., 2015)		19.0
RNNsearch + unk replace + large vocab + *ensemble* 8 models (Jean et al., 2015)		**21.6**
Our NMT systems		
Base	10.6	11.3
Base + reverse	9.9	12.6 (*+1.3*)
Base + reverse + dropout	8.1	14.0 (*+1.4*)
Base + reverse + dropout + global attention (*location*)	7.3	16.8 (*+2.8*)
Base + reverse + dropout + global attention (*location*) + feed input	6.4	18.1 (*+1.3*)
Base + reverse + dropout + local-p attention (*general*) + feed input	5.9	19.0 (*+0.9*)
Base + reverse + dropout + local-p attention (*general*) + feed input + unk replace		20.9 (*+1.9*)
Ensemble 8 models + unk replace		**23.0** (*+2.1*)

Table 1: **WMT'14 English-German results** – shown are the perplexities (ppl) and the *tokenized* BLEU scores of various systems on newstest2014. We highlight the **best** system in bold and give *progressive* improvements in italic between consecutive systems. *local-p* referes to the local attention with predictive alignments. We indicate for each attention model the alignment score function used in pararentheses.

- Better translation of long sentences

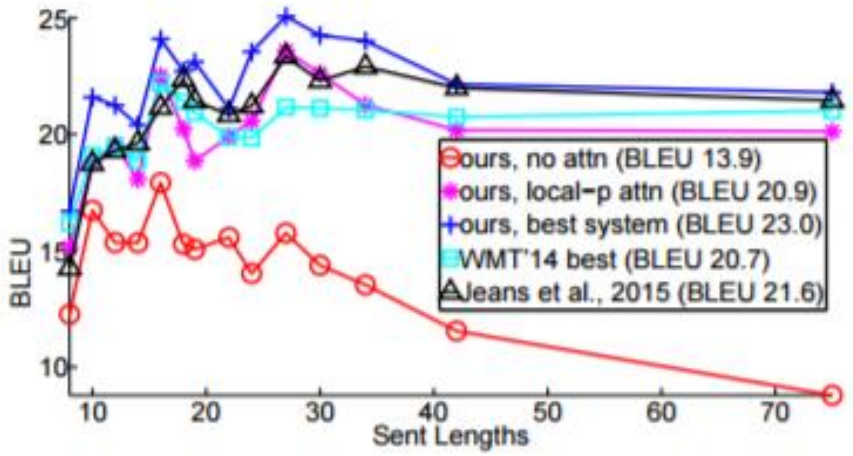
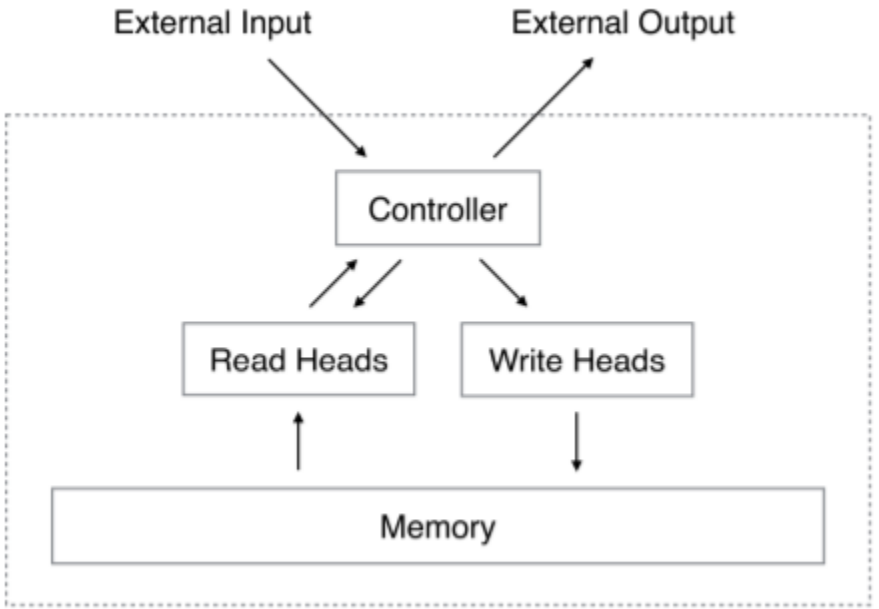


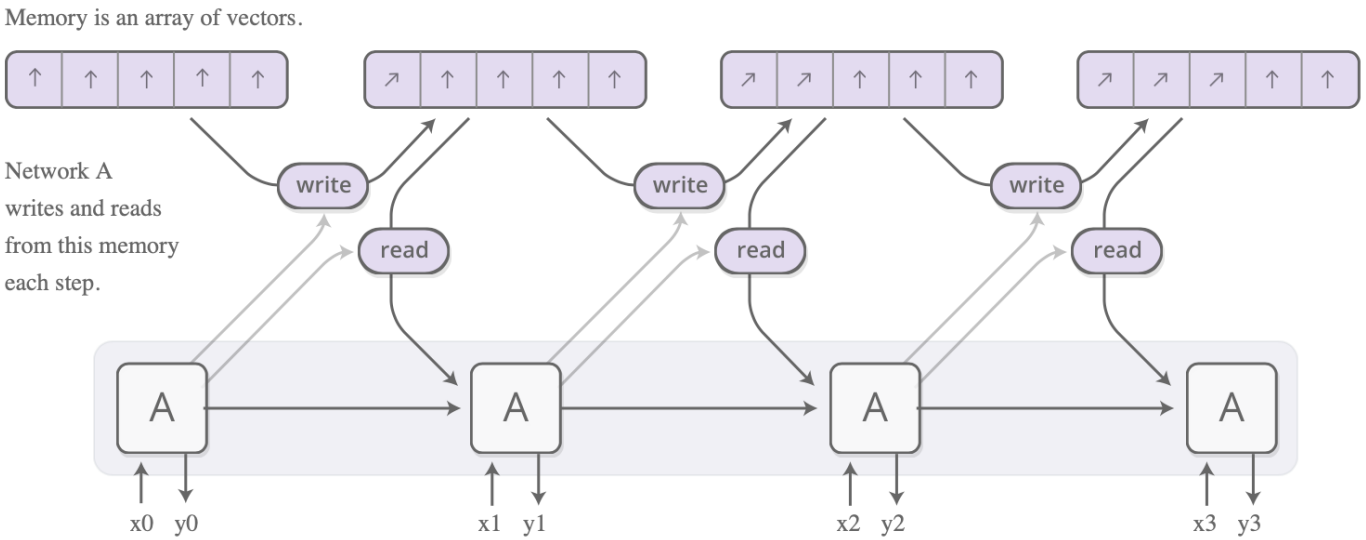
Figure 6: **Length Analysis** – translation qualities of different systems as sentences become longer.

Neural Turing Machines

- RNN + External Memory [graves et al., 2014]
  - Use attention distribution to control read and write access



Neural Turing Machines



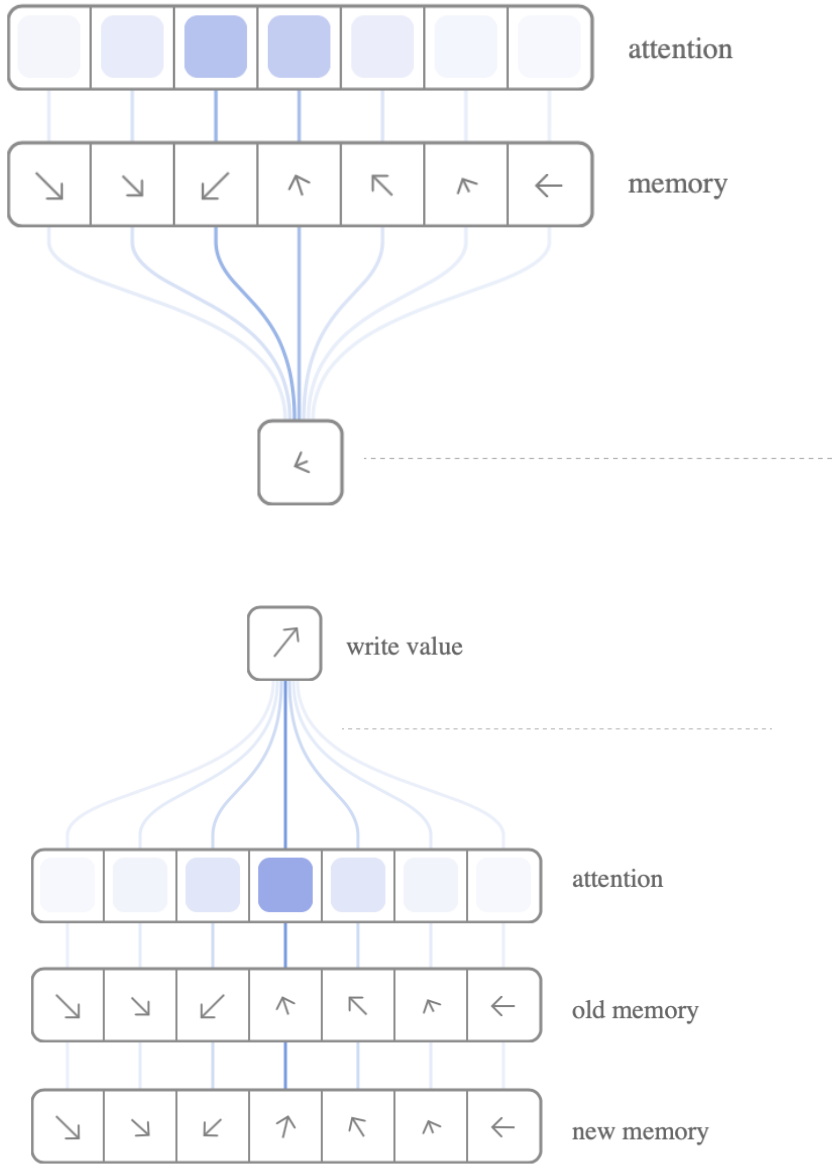


Image Captioning: CNN + RNN

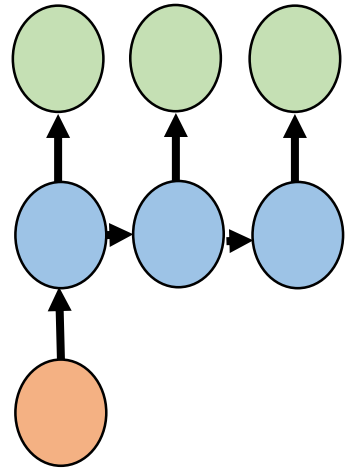
- CNN: feature representation learning
- RNN: language model

Input

Output

A man skiing down the snow covered mountain with a dark sky in the background

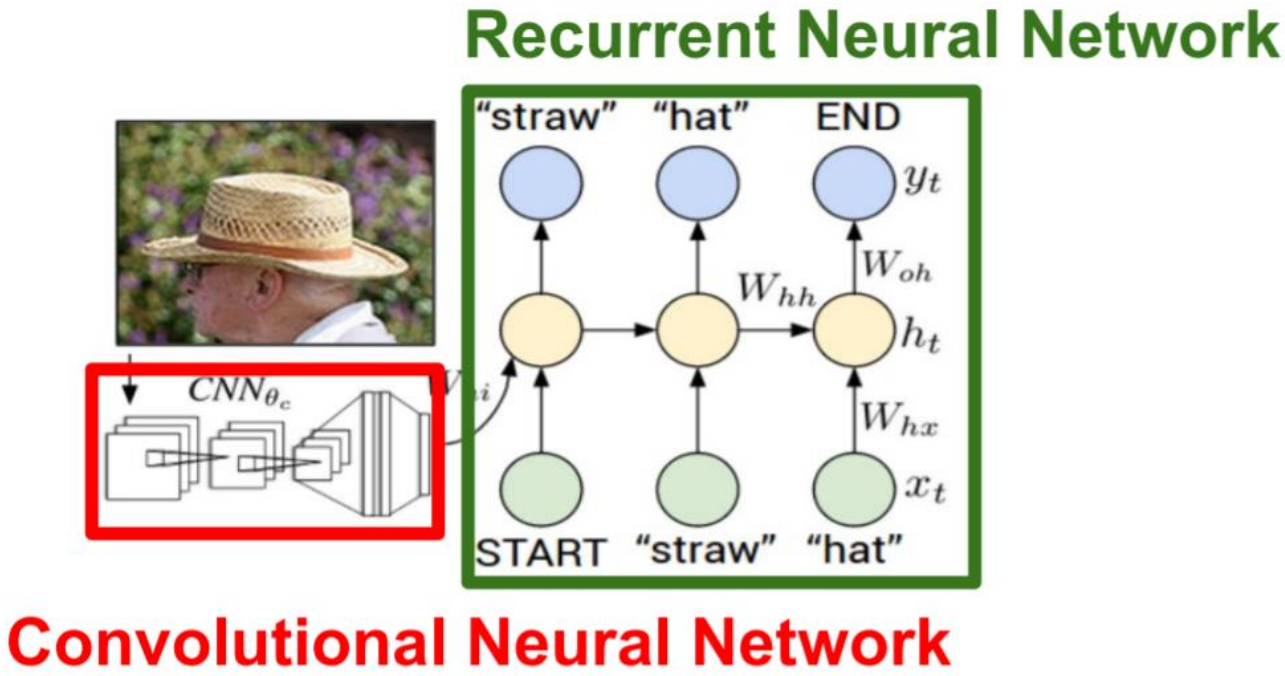
one to many



Picture from http://www.cs.cmu.edu/~rsalakhu/10807_2016/

Image Captioning

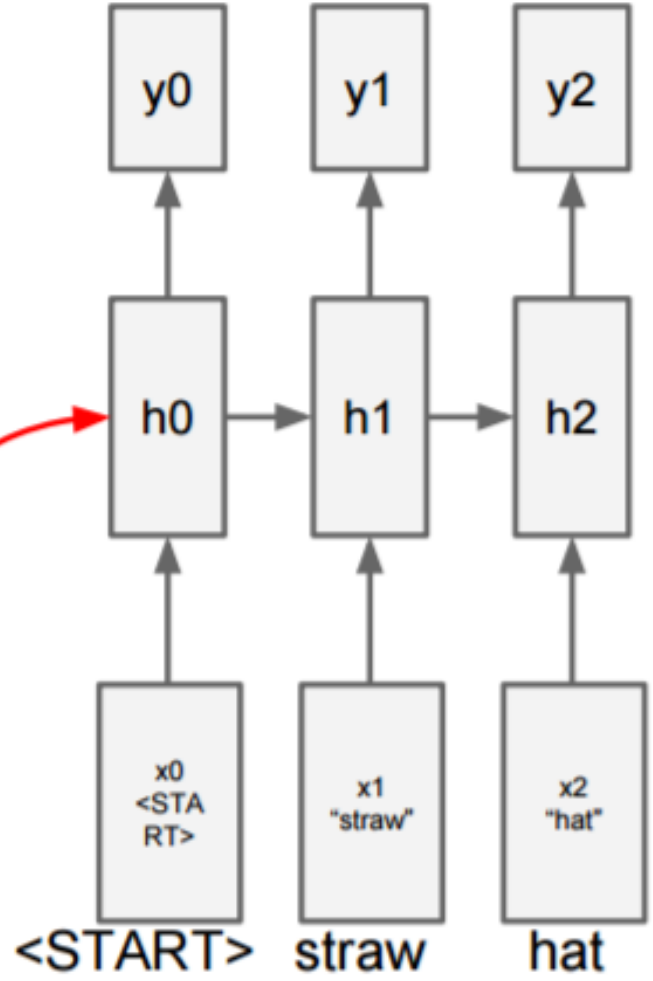
- Use CNN for encoding
  - Express a single differentiable function from raw image pixel values to class probabilities



Karpathy A. and Li F. F., Deep Visual-Semantic Alignments for Generating Image Descriptions

Automated Image Captioning with ConvNets and Recurrent Nets, https://cs.stanford.edu/people/karpathy/sfmltalk.pdf

training example



W_{ih}

v

x

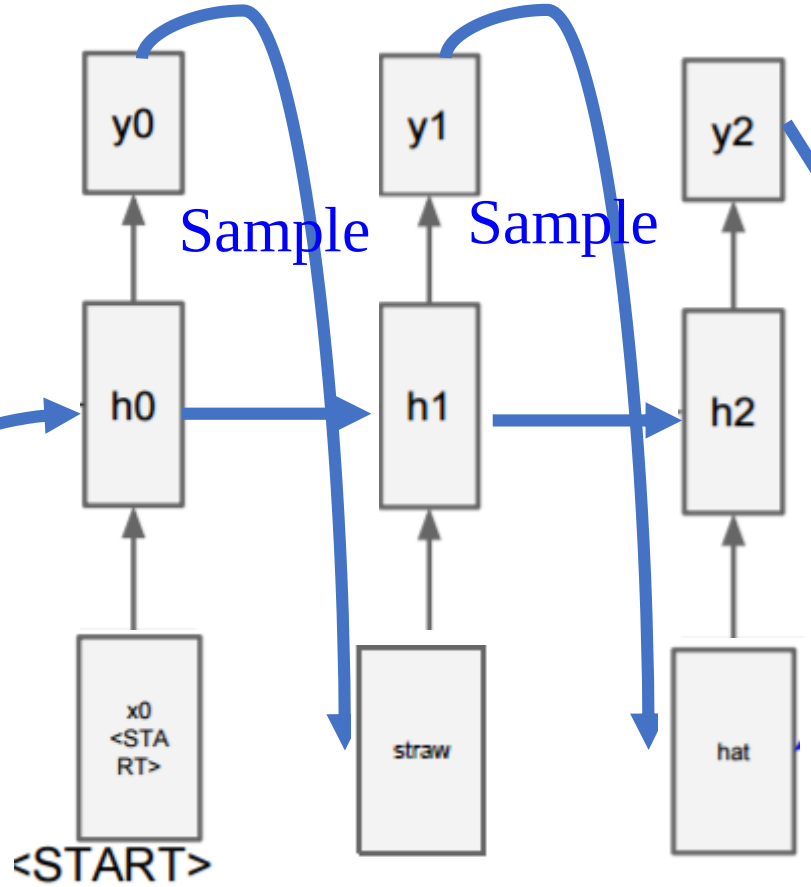
Before:

$$h_0 = \max(0, W_{xh} * x_0)$$

Now:

$$h_0 = \max(0, W_{xh} * x_0 + W_{ih} * v)$$

Test image

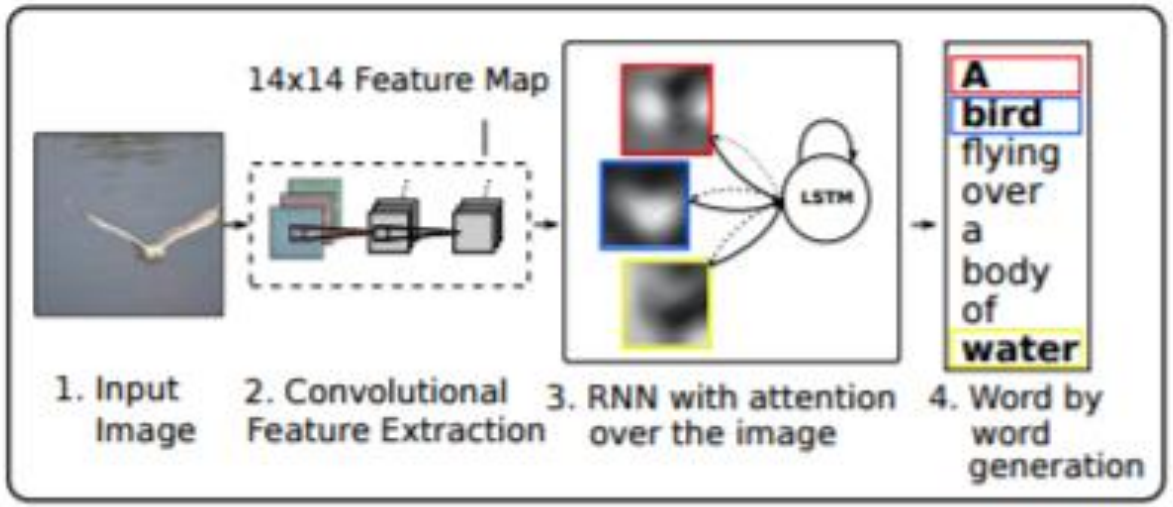


Sample!
<END> token
=> finish

Image Captioning + Attention Mechanism

- RNN with attention on image
  - Focuses its attention at a different spatial location when generating each word

Figure 1. Our model learns a words/image alignment. The visualized attentional maps (3) are explained in section 3.1 & 5.4



Xu et al., 2016, Show, Attend and Tell: Neural Image Caption Generation with Visual Attention

Image Captioning with Attention

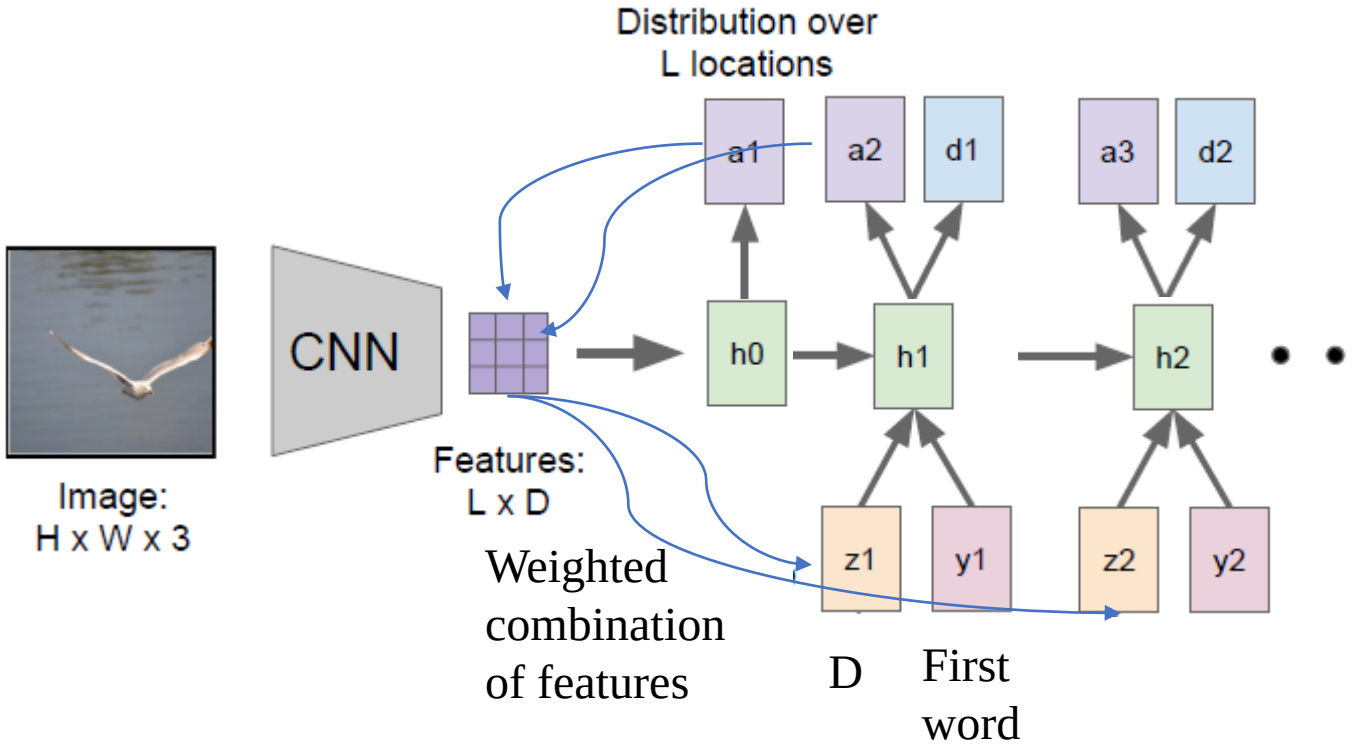


Figure 2. Attention over time. As the model generates each word, its attention changes to reflect the relevant parts of the image. "soft" (top row) vs "hard" (bottom row) attention. (Note that both models generated the same captions in this example.)

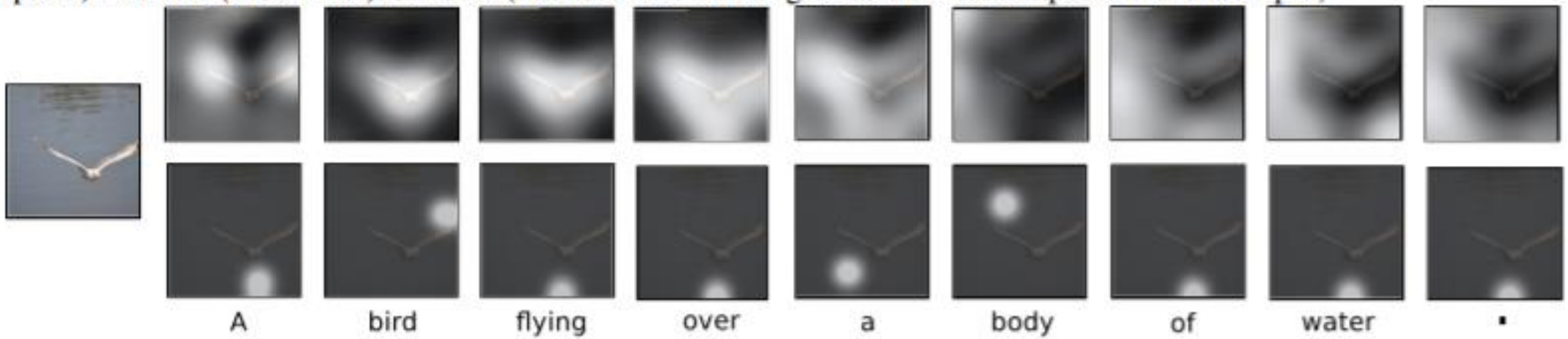


Figure 3. Examples of attending to the correct object (*white* indicates the attended regions, *underlines* indicated the corresponding word)

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

A little girl sitting on a bed with a teddy bear.

A group of people sitting on a boat in the water.

A giraffe standing in a forest with trees in the background.

Figure 5. Examples of mistakes where we can use attention to gain intuition into what the model saw.

A large white bird standing in a forest.

A woman holding a clock in her hand.

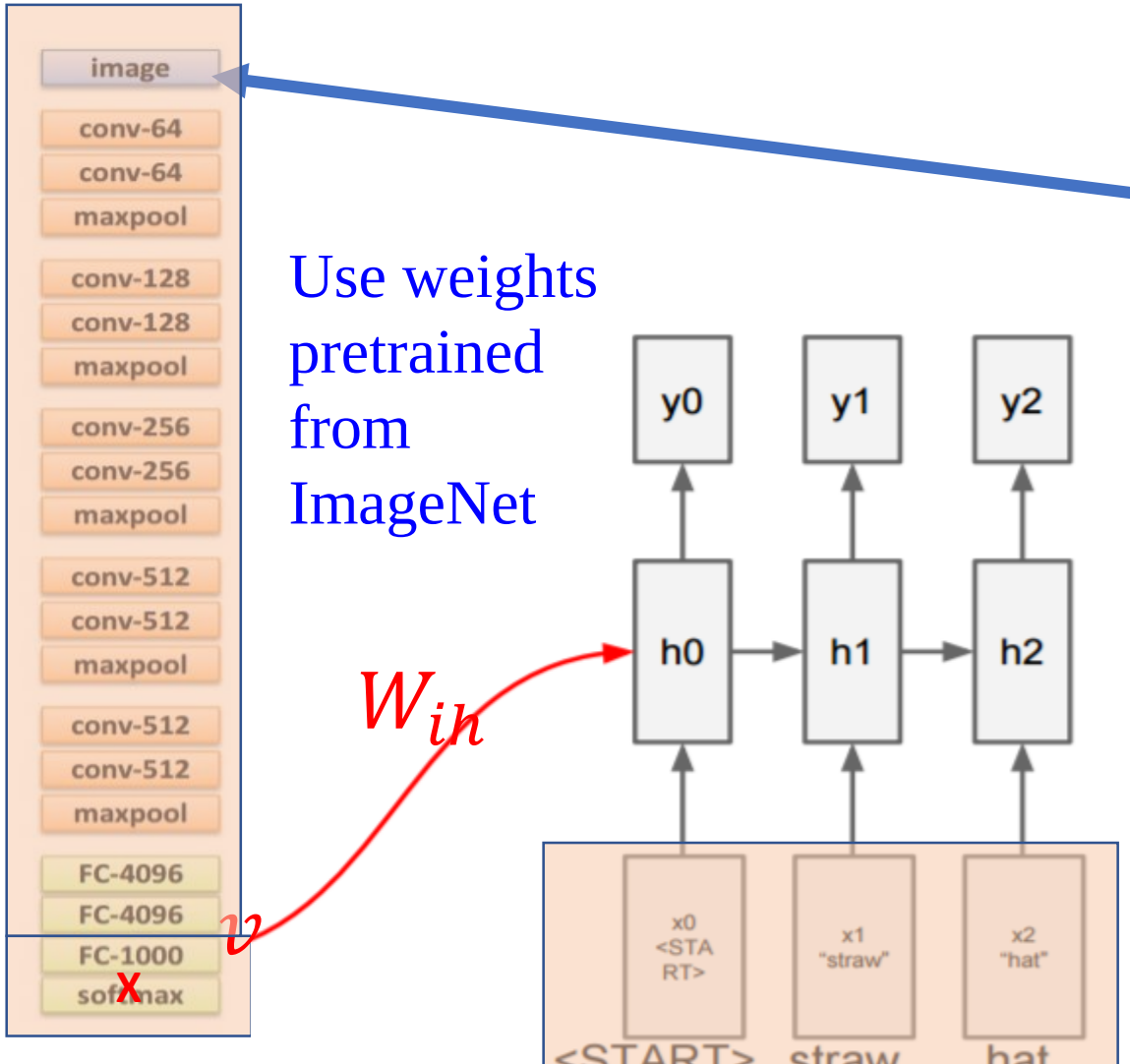
A man wearing a hat and
a hat on a skateboard.

A person is standing on a beach
with a surfboard.

A woman is sitting at a table
with a large pizza.

A man is talking on his cell phone
while another man watches.

Image Captioning + Transfer Learning



Use weights pretrained from ImageNet

training example

Use word vectors pretrained with word2vec

Image Sentences Dataset

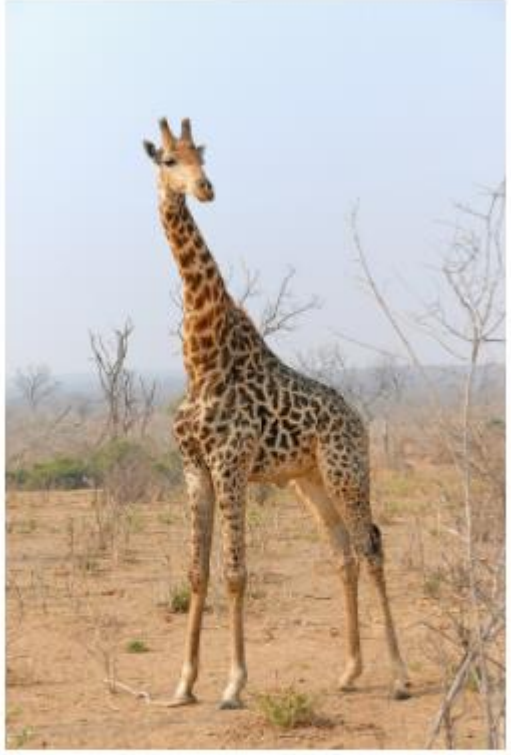
- Microsoft Coco (http://mscoco.org/)
  - ~300K images
  - ~ 5 sentences each

- Three challengers
  - Detection
  - Captioning
  - Keypoints

http://cs.stanford.edu/people/karpathy/deepimagesent/generationdemo/
https://github.com/karpathy/neuraltalk2

a pizza with toppings on a white plate

a woman is playing tennis on a tennis court



a giraffe standing in a field with trees in the background

a train traveling down tracks next to a forest

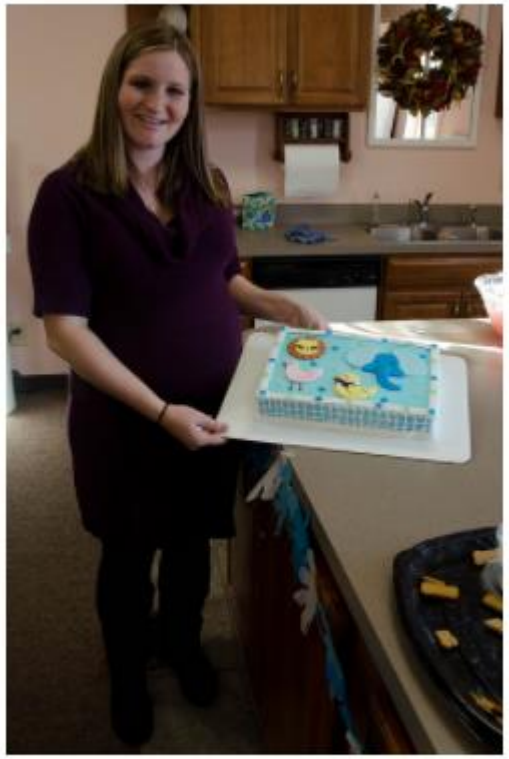
a man riding a wave on top of a surfboard

a cat sitting on a window sill looking out

Reasonable results

a group of people playing frisbee on a beach

a man sitting on a couch with a laptop and a laptop



a man is holding a box of food

a young boy is playing frisbee in the park

a group of elephants standing in a field

Not so good results

a woman is sitting at a table with a plate of food

a small white bird sitting on a green lawn

voman sitting on a
nch with a laptop

a man riding a motorcycle with a dog on the back

a man standing in front of a store with a clock

Failure Cases

Opinion Mining: Sentiment Analysis

- Recursive neural networks

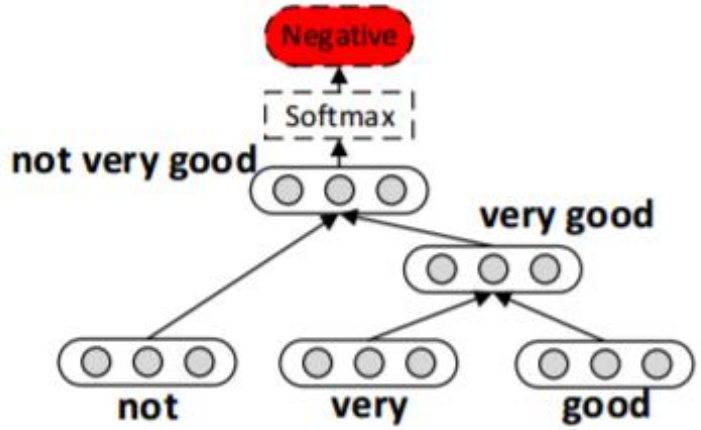
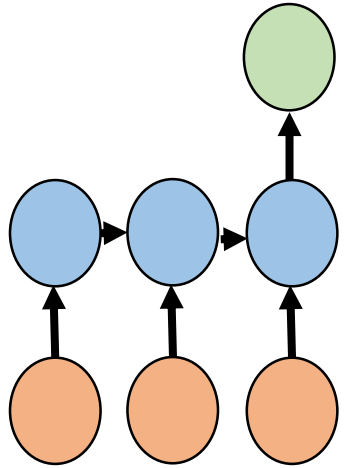


Figure 1: The composition process for "*not very good*" in Recursive Neural Network.



Dong et al., 2014. Adaptive Recursive Neural Network for Target-dependent Twitter Sentiment Classification

Attention as A General Technique

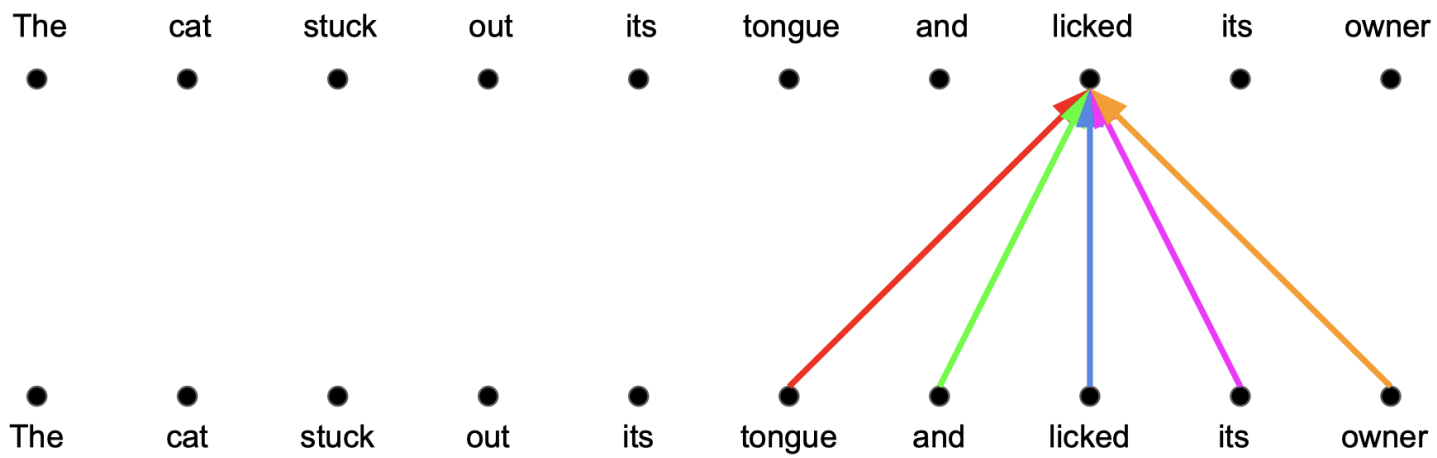
We have seen attention in machine translation, neural Turing machines, etc

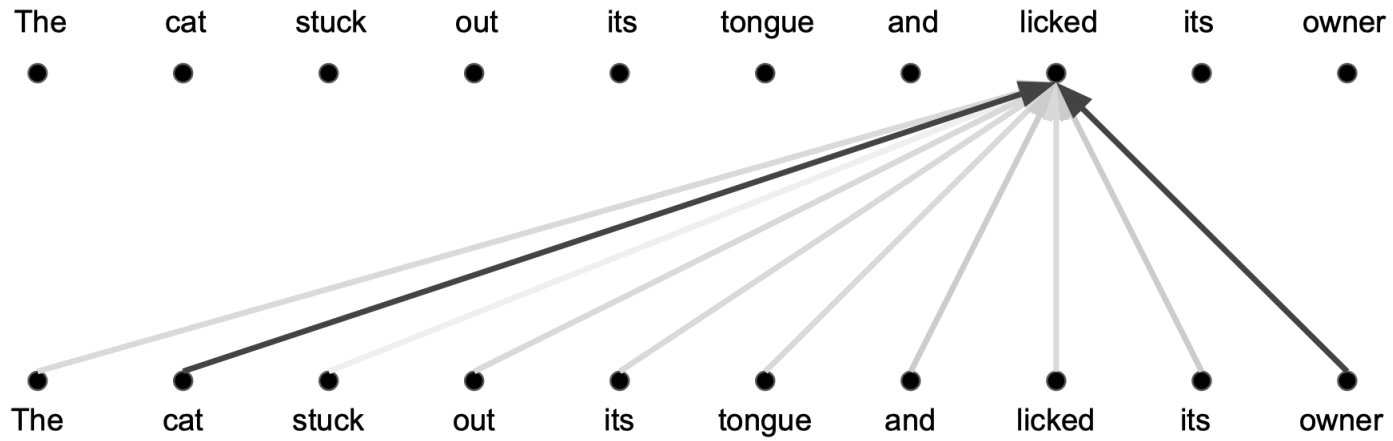
- Attention is a technique to compute a weighted sum of the *values* for a certain *query*, given that we have a vector of candidate values
- Attention is a way to obtain a fixed size representation of an arbitrary set of representations (the values), dependent on some other representation (the query)

Self-Attention

- Disadvantage of existing architectures:
  - RNN are sequential in nature, which inhibits parallelization, long-term dependencies, etc
  - CNN are easy to parallelize, but conv kernel only captures local dependencies, need many layers
- Use attention for representation
  - Constant ‘path length’ between any two positions.
  - Gating/multiplicative interactions.

Convolution vs Attention





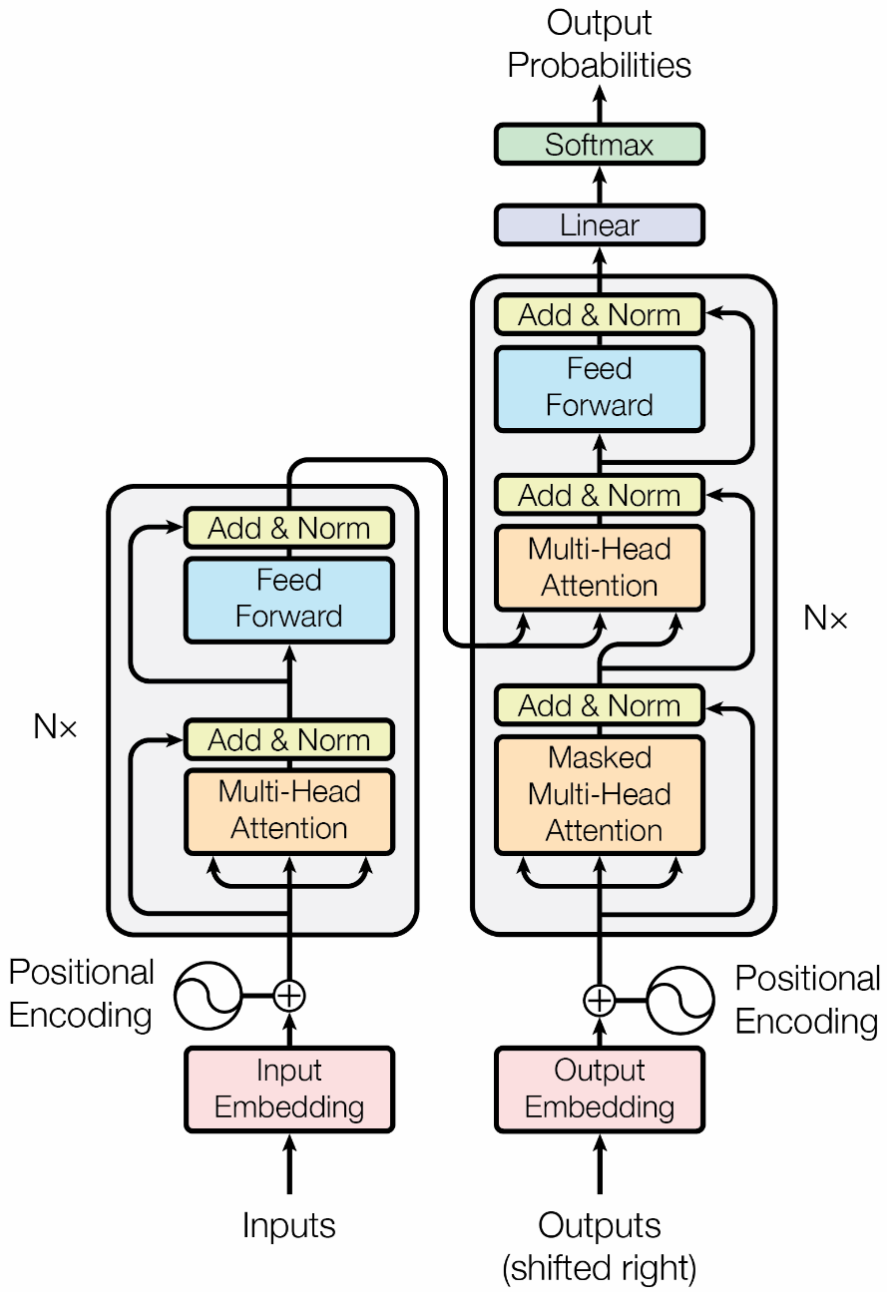
http://cs224n.stanford.edu

“Attention Is All You Need”

- Based solely on attention mechanisms
- Completely remove recurrence and convolutions units

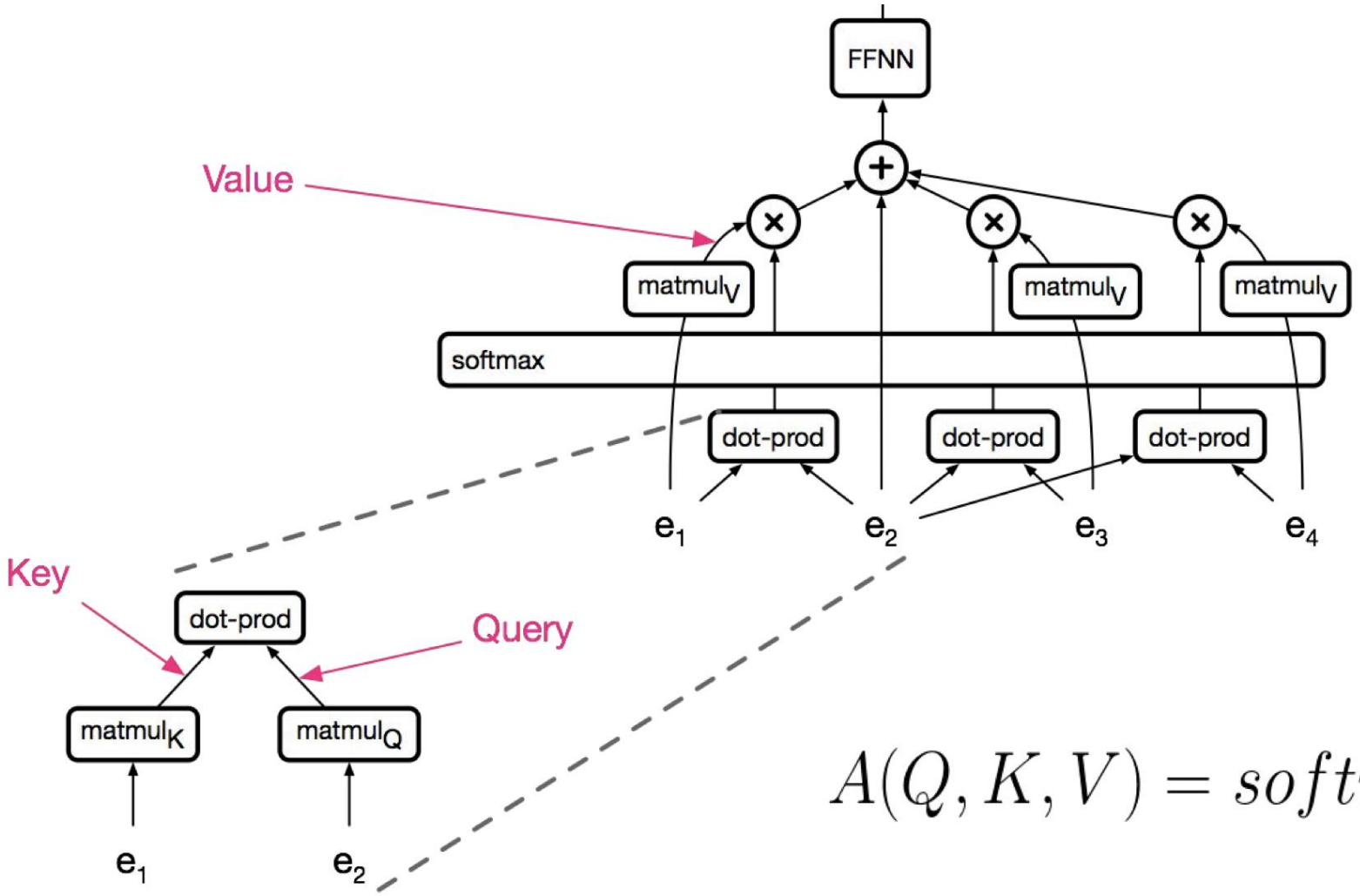
Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	



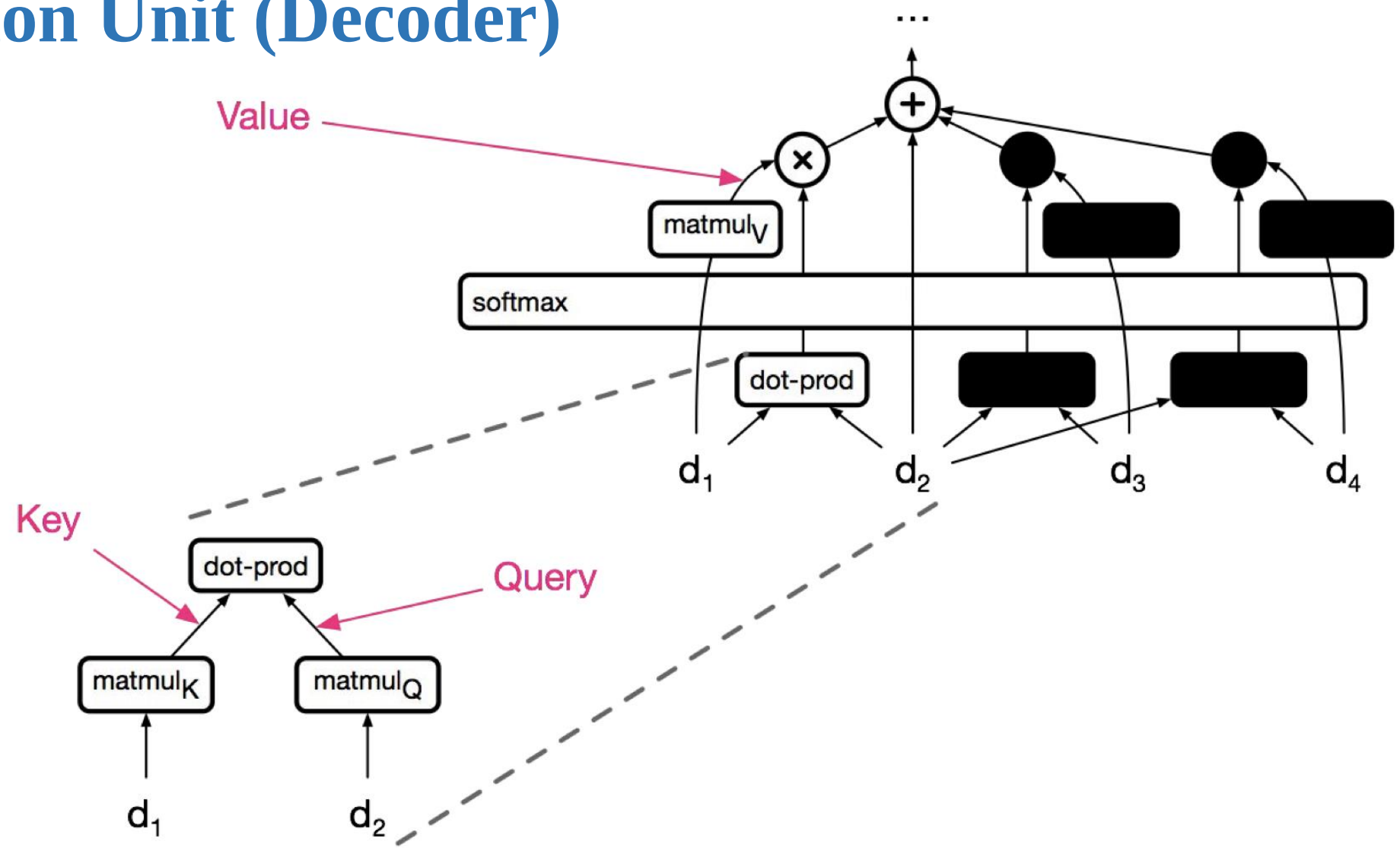
Google’s transformer model

Attention Unit (Encoder)



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

Attention Unit (Decoder)



http://cs224n.stanford.edu

Any Questions?