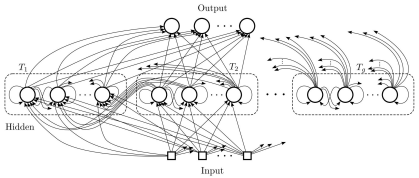
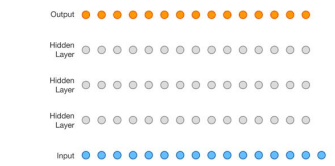
Recurrent Nets and Sequence Generation

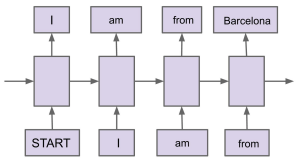
- Lecture topics:
  - Recurrent Neural Networks
  - Long-Short Term Memory (LSTM)
  - (Conditional) Sequence Generation
- Guest Lecturer: **Oriol Vinyals**
  - Joined DeepMind in 2016.
  - Worked in Google Brain from 2013 to 2016.
  - PhD in Artificial Intelligence from UC Berkeley (2009-13). Supervisor: Darrell / Morgan.
  - Current research topics: deep learning, sequence modeling, generative models, distillation, RL/Starcraft, one shot learning.

Temporal Hierarchies

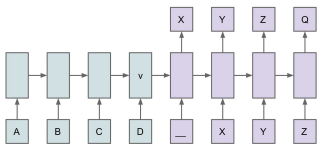




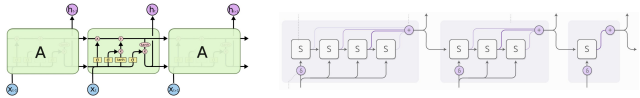
Sequence Prediction



Seq2Seq



Recurrent Architectures



Sequences and Recurrent Neural Nets

Oriol Vinyals // @OriolVinyalsML
Research Scientist
DeepMind
UCL, Feb 2018

Roadmap

I. Deep Learning in (almost) one slide
II. Why should I care about Sequences?

III. Recurrent Neural Networks Fundamentals
  A. The chain rule (loss)
  B. Vanishing gradients (optimization)
  C. LSTMs / GRUs (architecture)

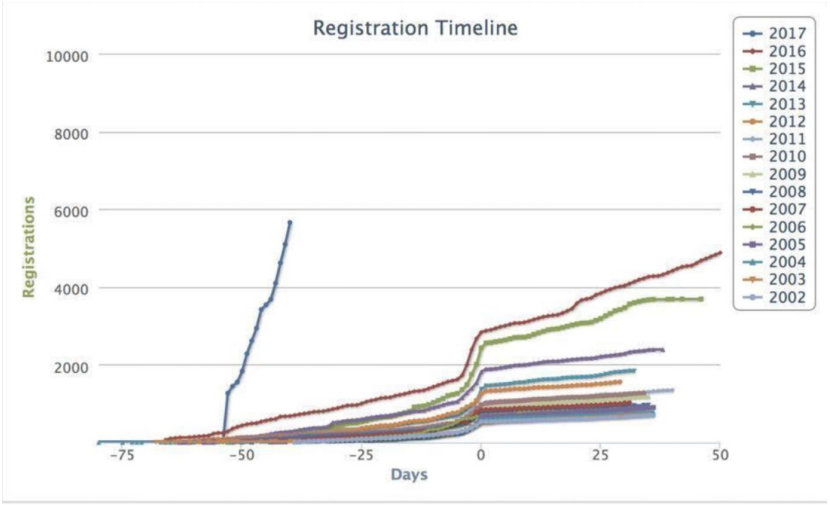
IV. Recurrent Networks as Sequence Decoders
  A. Sequence-To-Sequence Learning & Applications

V. Beyond RNNs

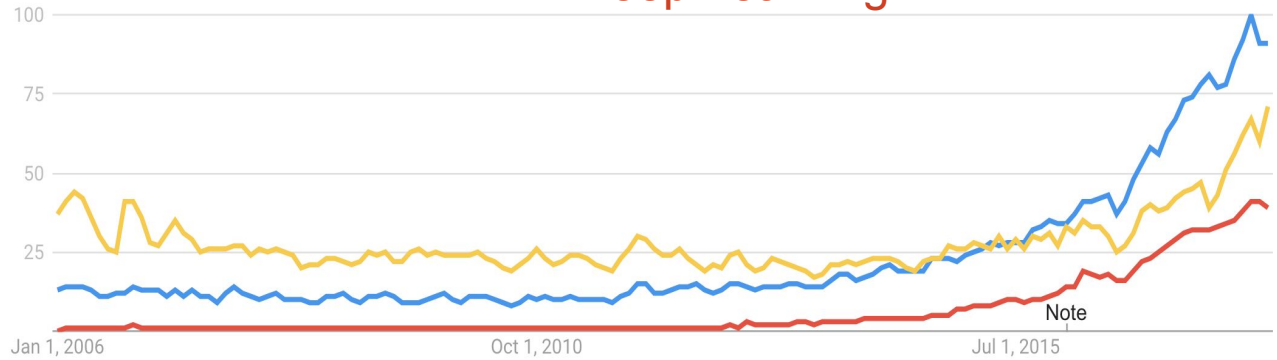
Part I: Deep Learning Overview

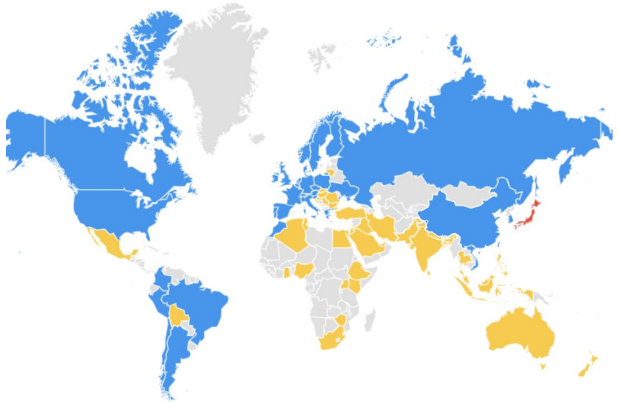
Lots of attention

- Startups / VCs
- Big companies
- Open source efforts
- Press (noise...)
- Universities



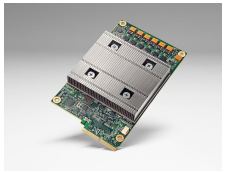
Machine Learning
Artificial Intelligence
Deep Learning





Deep Learning: Zooming Out

Platforms



Microsoft Azure

aws

Google Cloud

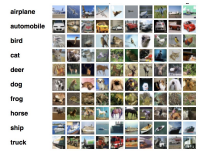
Frameworks

Caffe | Keras

Microsoft CNTK | mxnet | theano

PYTORCH | Caffe2

Datasets



Caltech 101 | MAGENET

LDC Linguistic Data Consortium

UCI Machine Learning Repository
Center for Machine Learning and Intelligent Systems

WMT Workshop 2014

Deep Learning: Zooming In

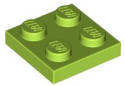
Non-Linearities

Relu
Sigmoid
Tanh
GRU
LSTM
Linear
...

Optimizer

SGD
Momentum
RMSProp
Adagrad
Adam
Second Order (KFac)
...

Connectivity Pattern



Fully connected
Convolutional
Dilated
Recurrent
Recursive
Skip / Residual
Random
...

Loss

Cross Entropy
Adversarial
Variational
Max. Likelihood
Sparse
L2 Reg
REINFORCE
...

Hyper Parameters

Learning Rate
Decay
Layer Size
Batch Size
Dropout Rate
Weight init
Data augmentation
Gradient Clipping
Beta
Momentum

IS THIS A
CAT or DOG?

CAT DOG

OUTPUT LAYER

ACTIVATED NEURONS

INPUT LAYER

DEEP *neural* NETWORK

A simple recipe

1. Lots of data

A simple recipe

1. Lots of data
2. Deep, BIG DNNs/CNNs/RNNs

A simple recipe

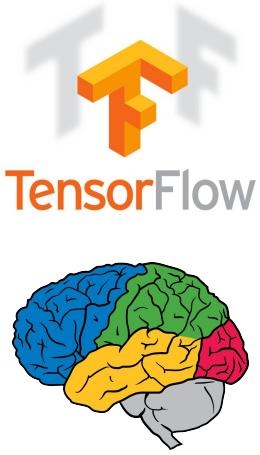
1. Lots of data
2. Deep, BIG DNNs/CNNs/RNNs
3. Lots of GPUs

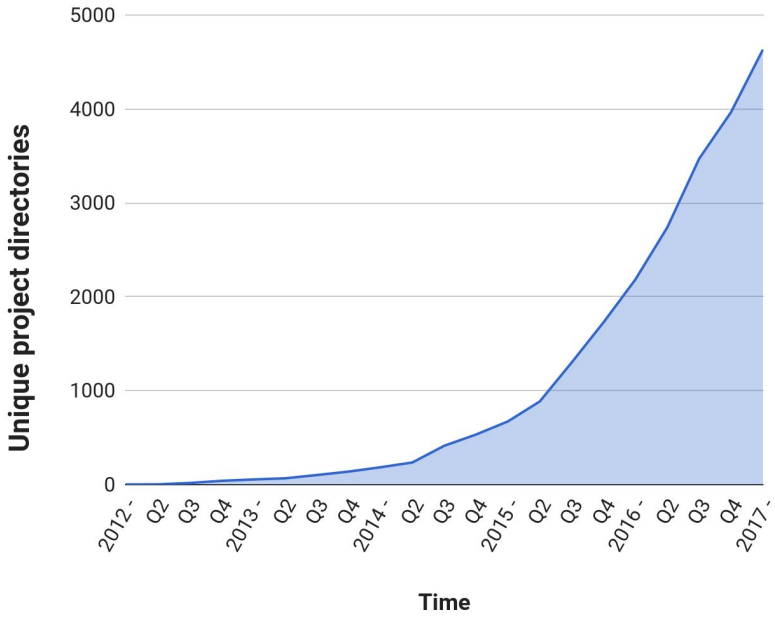
A simple recipe

1. Lots of data
2. Deep, BIG DNNs/CNNs/RNNs
3. Lots of GPUs
4. PROFIT!

Growing Use of Deep Learning at Google

"Google will soon be a big LSTM", Jüergen Schmidhuber



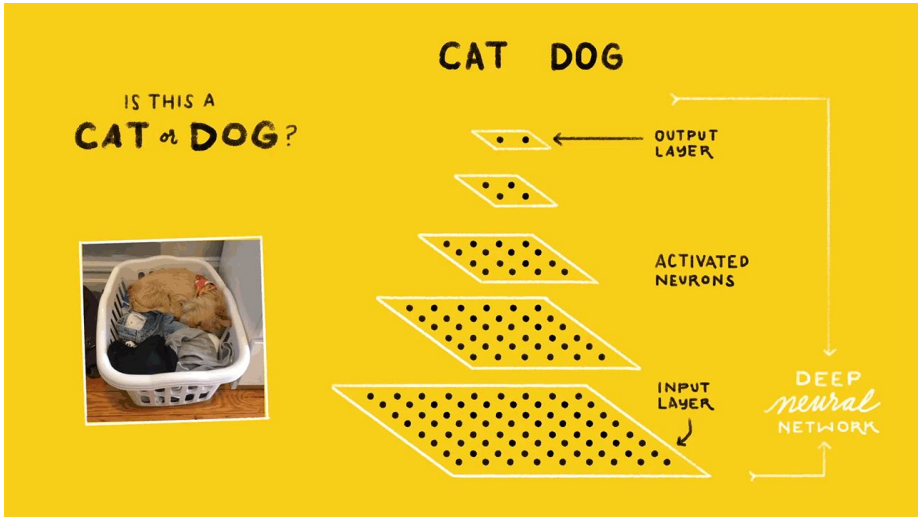


Across many products/areas:

Android
Apps
drug discovery
Gmail
Image understanding
Maps
Photos
Robotics research
Search
Speech
Translate
YouTube
… many others …

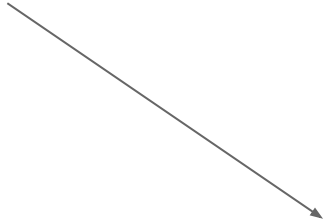
Are we done yet?

- Neural Nets struggle with sequences
- Inputs and outputs must
  be fixed-sized vectors

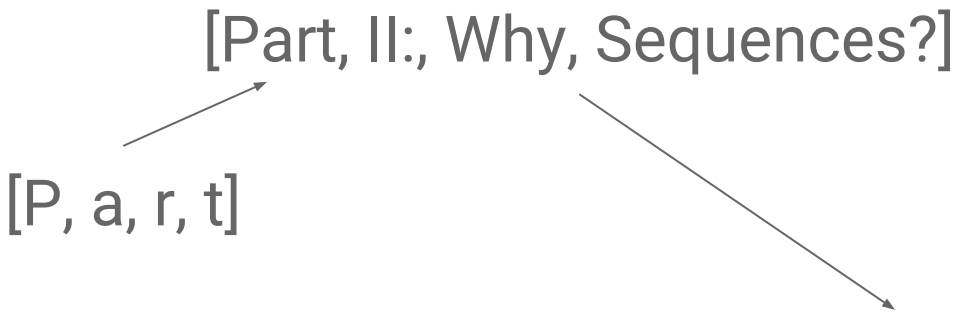


Part II: Why Sequences?

[Part, II:, Why, Sequences?]



Part II: Why Sequences?

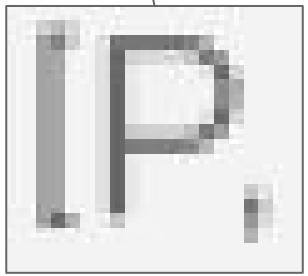


Part II: Why Sequences?

[Part, II:, Why, Sequences?]

[P, a, r, t]

Part II: Why Sequences?



Part II: Why Sequences?

[Part, II:, Why, Sequences?]

Part II: Why Sequences?

[Part, II:, Why, Sequences?]

[P, a, r, t]

Part II: Why Sequences?

[Part, II:, Why, Sequences?]

[P, a, r, t]

Part II: Why Sequences?

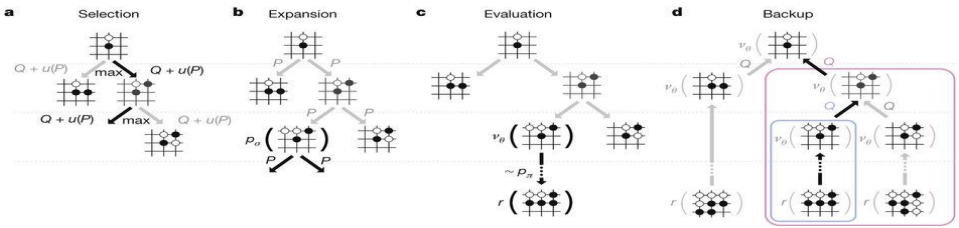
14

Sequences

- Words, Letters
- Speech
- Images, Videos, Touch
- Programs
- Sequential Decision Making (RL)

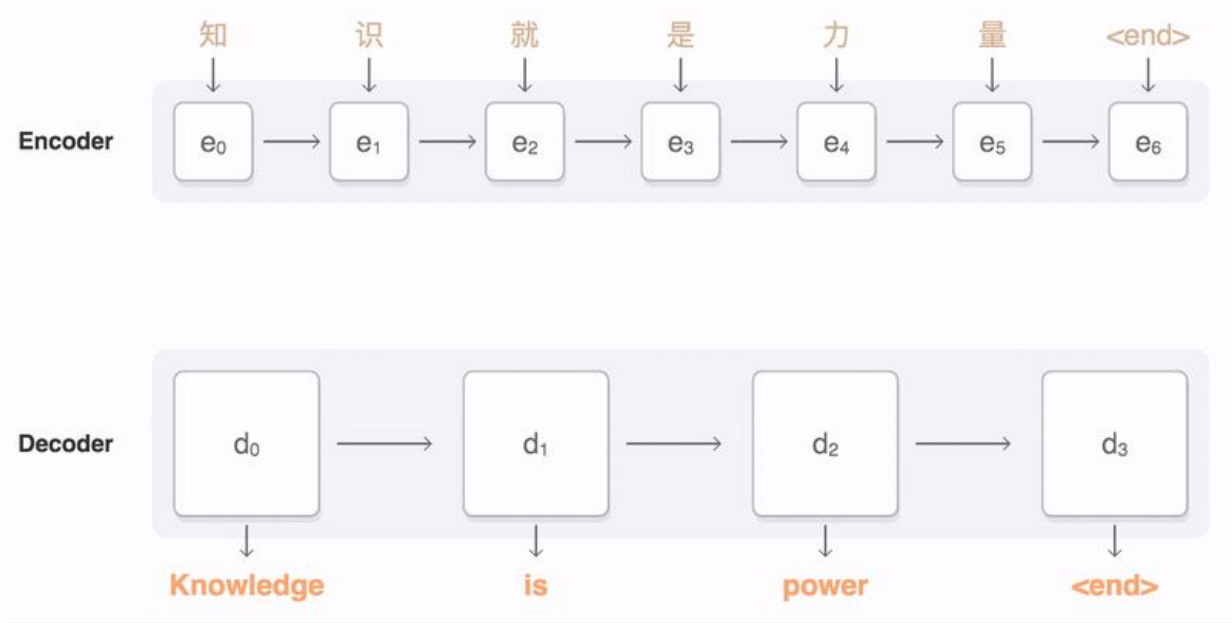
> *50 years ago, the fathers of artificial intelligence convinced everybody that logic was the key to intelligence. Somehow we had to get computers to do logical reasoning. The alternative approach, which they thought was crazy, was to forget logic and try and understand how networks of brain cells learn things. Curiously, two people who rejected the logic based approach to AI were Turing and Von Neumann. If either of them had lived I think things would have turned out differently... now neural networks are everywhere and the crazy approach is winning.*
>
> Geoff Hinton


```
while (*d++ = *s++);
```



Mix and Match: Google Neural MT

Wu et al, 2016 (Kalchbrenner et al, 2013; Sutskever et al, 2014; Cho et al, 2014; Bhadanau et al, 2014; ...)



Math / Code [Karpathy, 2015]

Proof. Omitted. □

Lemma 0.1. *Let $\mathcal{C}$ be a set of the construction.*

Let $\mathcal{C}$ be a gerber covering. Let $\mathcal{F}$ be a quasi-coherent sheaves of $\mathcal{O}$-modules. We have to show that

$$\mathcal{O}_{\mathcal{O}_X} = \mathcal{O}_X(\mathcal{L})$$

.

Proof. This is an algebraic space with the composition of sheaves $\mathcal{F}$ on $X_{\acute{e}tale}$ we have

$$\mathcal{O}_X(\mathcal{F}) = \{morph_1 \times_{\mathcal{O}_X} (\mathcal{G}, \mathcal{F})\}$$

where $\mathcal{G}$ defines an isomorphism $\mathcal{F} \to \mathcal{F}$ of $\mathcal{O}$-modules. □

Lemma 0.2. *This is an integer $\mathcal{Z}$ is injective.*

Proof. See Spaces, Lemma ??. □

Lemma 0.3. *Let S be a scheme. Let X be a scheme and X is an affine open covering. Let $\mathcal{U} \subset \mathcal{X}$ be a canonical and locally of finite type. Let X be a scheme. Let X be a scheme which is equal to the formal complex.*

The following to the construction of the lemma follows.

Let X be a scheme. Let X be a scheme covering. Let

$$b : X \to Y' \to Y \to Y \to Y' \times_X Y \to X.$$

be a morphism of algebraic spaces over S and Y.

Proof. Let X be a nonzero scheme of X. Let X be an algebraic space. Let $\mathcal{F}$ be a quasi-coherent sheaf of $\mathcal{O}_X$-modules. The following are equivalent

(1) $\mathcal{F}$ is an algebraic space over S.
(2) If X is an affine open covering.

Consider a common structure on X and X the functor $\mathcal{O}_X(U)$ which is locally of finite type. □

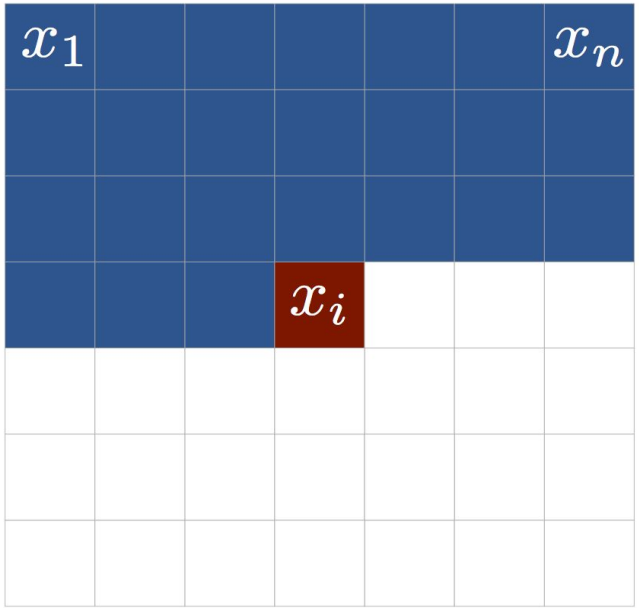
```
static void num_serial_settings(struct tty_struct *tty)
{
  if (tty == tty)
    disable_single_st_p(dev);
  pci_disable_spool(port);
  return 0;
}

static void do_command(struct seq_file *m, void *v)
{
  int column = 32 << (cmd[2] & 0x80);
  if (state)
    cmd = (int)(int_state ^ (in_8(&ch->ch_flags) & Cmd) ? 2 : 1);
  else
    seq = 1;
  for (i = 0; i < 16; i++) {
    if (k & (1 << 1))
      pipe = (in_use & UMXTHREAD_UNCCA) +
        ((count & 0x0000000fffffff8) & 0x000000f) << 8;
    if (count == 0)
      sub(pid, ppc_md.kexec_handle, 0x20000000);
    pipe_set_bytes(i, 0);
  }
  /* Free our user pages pointer to place camera if all dash */
  subsystem_info = &of_changes[PAGE_SIZE];
  rek_controls(offset, idx, &soffset);
  /* Now we want to deliberately put it to device */
  control_check_polarity(&context, val, 0);
  for (i = 0; i < COUNTER; i++)
    seq_puts(s, "policy ");
}
```

Human: A young girl asleep on the sofa cuddling a stuffed bear.

NIC: A close up of a child holding a stuffed animal.

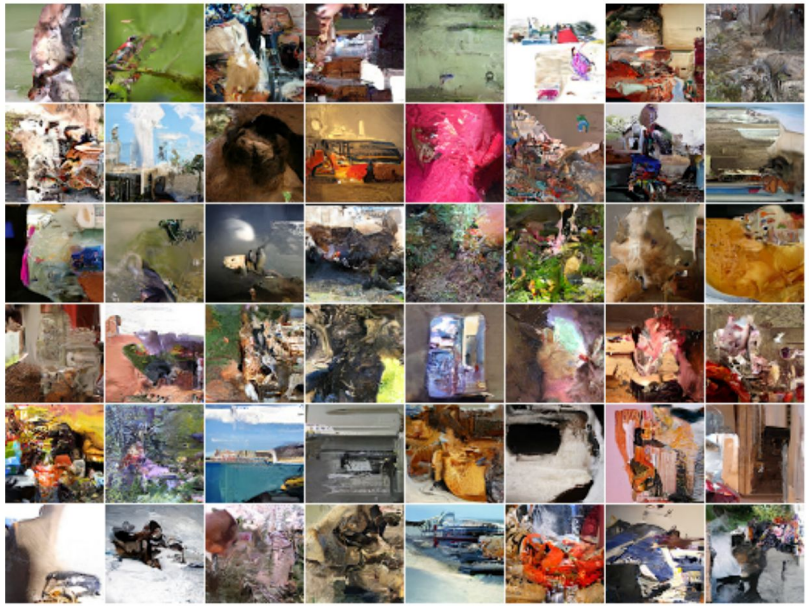
NIC: A baby is asleep next to a teddy bear.

Pixels as a sequence



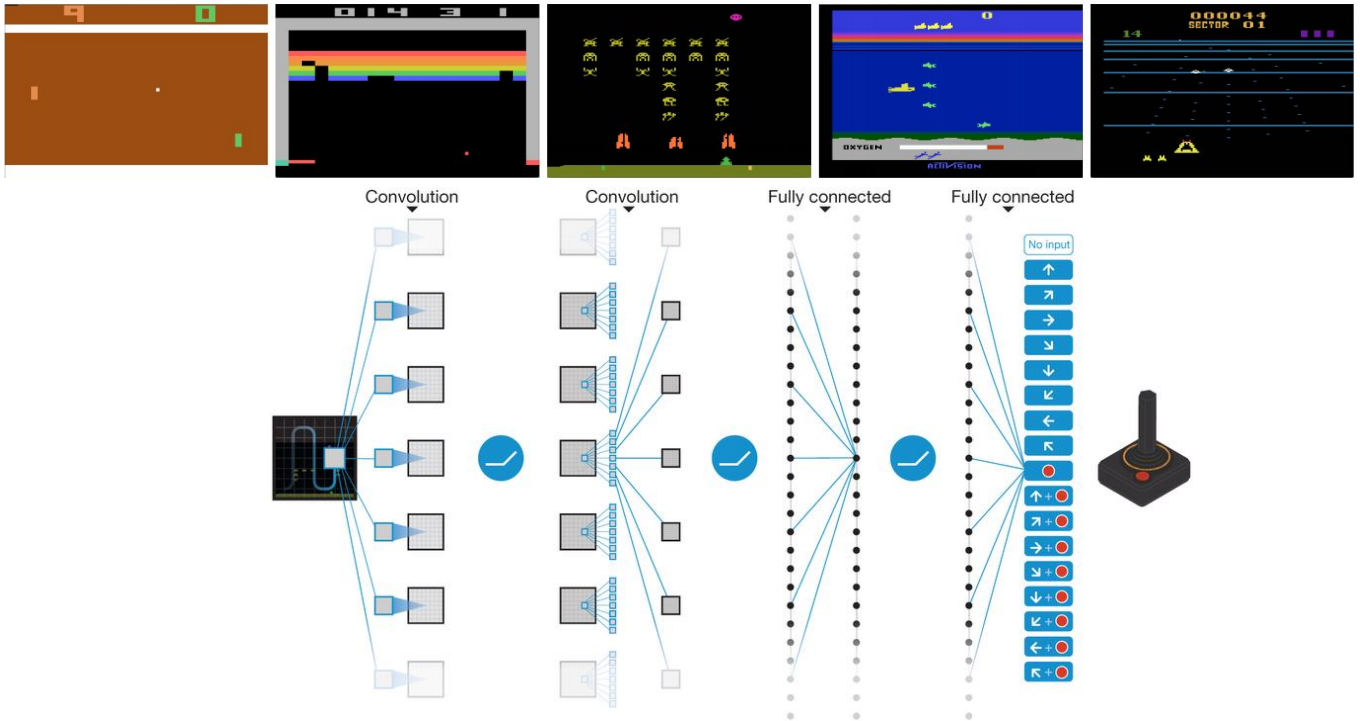
$$p(\mathbf{x}) = \prod_{i=1}^{n^2} p(x_i | x_1, ..., x_{i-1})$$

The Chain Rule (more on it later)



Deep Reinforcement Learning

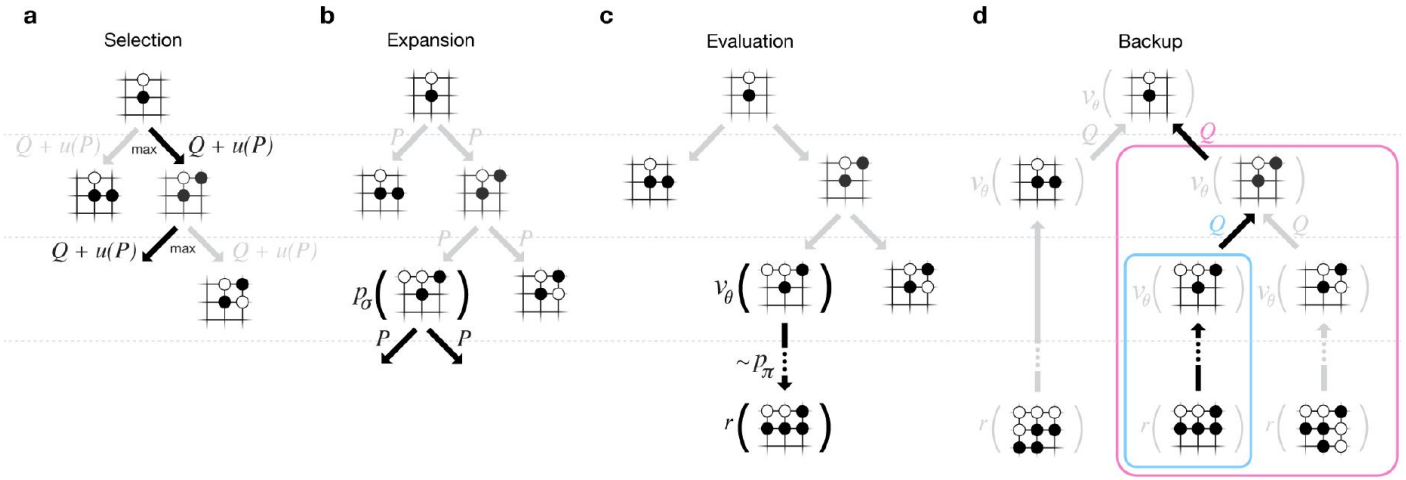
Human-level control through deep reinforcement learning, Mnih et al, Nature 2015



Deep Reinforcement Learning

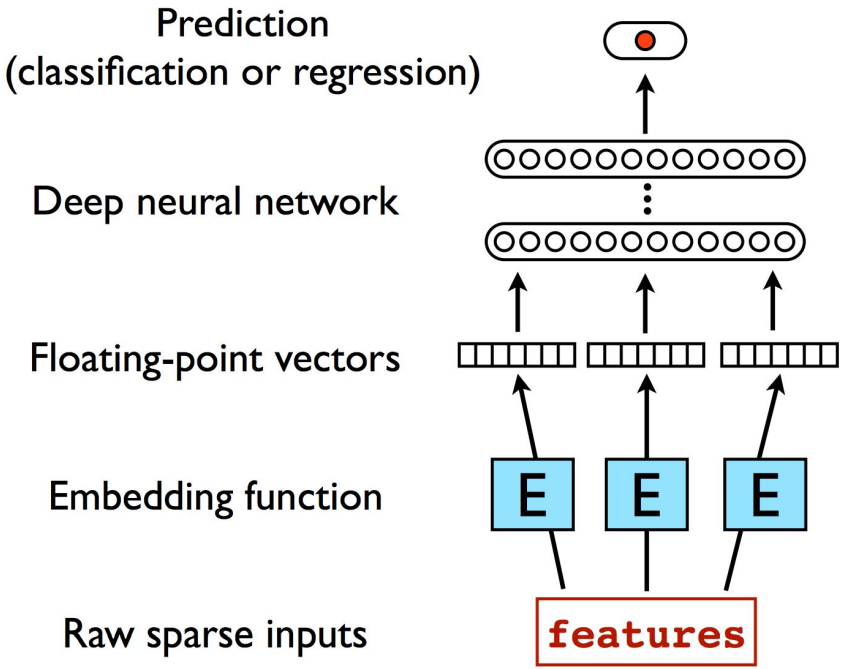
Mastering the game of Go with deep neural networks and tree search.

Silver et al, Nature 2016



Part III: Recurrent Neural Networks Fundamentals

Word Vectors Aside



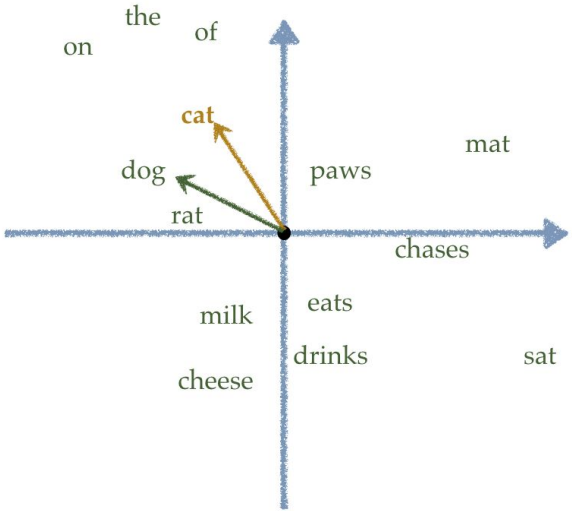
Embedding == Sparse matrix multiplication

Let v= [0 0 0 0 0 0 0 0 1 0 0 0 0 … 0 0 0 0]

each position is a word (length(v) == vocab size)

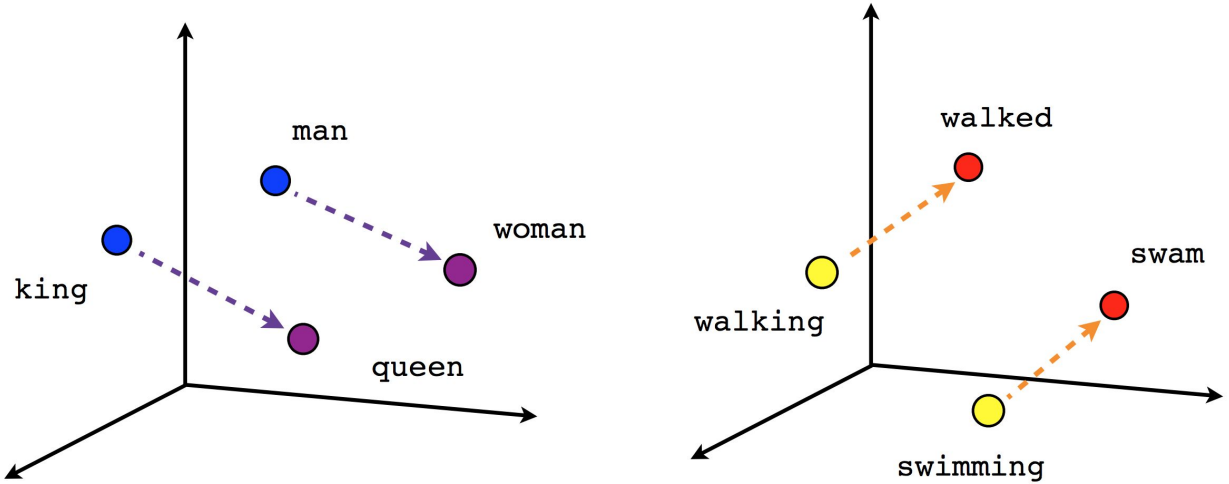
To embed a word we simply do:

e = **W**v



where **W** is a D x vocab size matrix (one D dim. vector for each word)

Vector arithmetic



Solve analogies with vector arithmetic!

$$V(\text{queen}) - V(\text{king}) \approx V(\text{woman}) - V(\text{man})$$

$$V(\text{queen}) \approx V(\text{king}) + (V(\text{woman}) - V(\text{man}))$$

Part III.A The Chain Rule

Loss and Architecture basics

Basics

Supervised Learning: many input / output examples (x, y)

Find mapping *f* s.t. $$\mathbf{y} \approx f(\mathbf{x})$$

Define $$\mathcal{L}(\boldsymbol{\theta}) = \sum_{i=1}^{N} \ell(f_{\boldsymbol{\theta}}, \{\mathbf{x}_i, \mathbf{y}_i\})$$

and learn by optimizing $$\boldsymbol{\theta} = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})$$

Motivating Example: Language

KING LEAR:
O, if you were a feeble sight, the courtesy of your law,
Your sight and several breath, will wear the gods
With his heads, and my hands are wonder'd at the deeds,
So drop upon your lordship's head, and your opinion
Shall be against your honour.

Who wrote these lines?

A. **William Shakespeare**
B. **William Shakespeare's ghostwriter**
C. **Ben Johnson**
D. **Molière (translation)**
E. **Andrej Karpathy's recurrent neural network**

Slide Credit: Piotr Mirowski

n-grams

$$P(w_t|w_{t-1}, \ldots, w_{t-n+1})$$

what to cook with broccoli and _

what to cook with broccoli and **beef**

what to cook with broccoli and **butter**

what to cook with broccoli and **blenders**

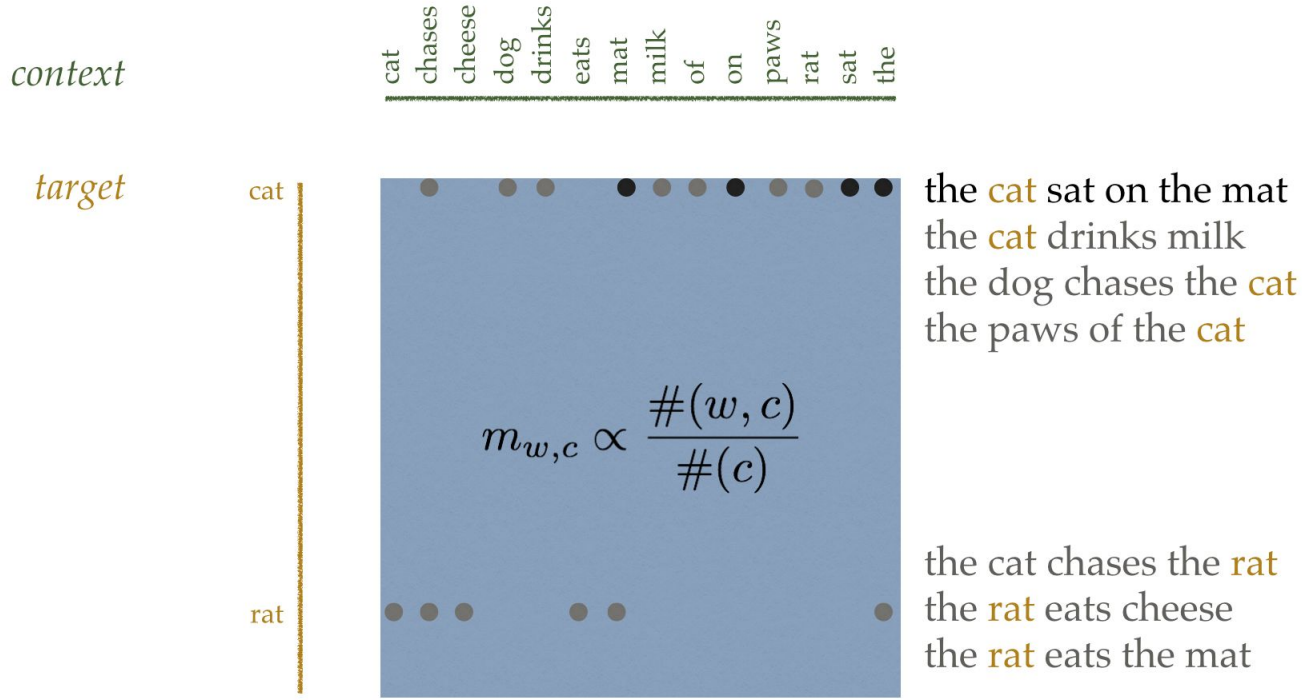
what to cook with broccoli and **boomboxes**

Modelling “P”

context					*target*	$P(w_t \mid w_{t-1}, w_{t-2}, \ldots w_{t-5})$
the	cat	sat	on	the	**mat**	0.15
w_{t-5}	w_{t-4}	w_{t-3}	w_{t-2}	w_{t-1}	w_t	
the	cat	sat	on	the	**rug**	0.12
the	cat	sat	on	the	**hat**	0.09
the	cat	sat	on	the	**dog**	0.01
the	cat	sat	on	the	**the**	0
the	cat	sat	on	the	**sat**	0
the	cat	sat	on	the	**robot**	?
the	cat	sat	on	the	**printer**	?

Slide Credit: Piotr Mirowski

2-grams



Slide Credit: Piotr Mirowski

n-grams

Vocabulary: 10K

Table size: 10000^n

n-grams

$$P(w_1, w_2, \ldots, w_{T-1}, w_T) \approx \prod_{t=1}^{T} P(w_t | w_{t-1}, \ldots, w_{t-n+1})$$

the	cat	sat	on	the	mat	$P(w_1)$
the	**cat**	sat	on	the	mat	$P(w_2 \mid w_1)$
the	cat	**sat**	on	the	mat	$P(w_3 \mid w_2, w_1)$
the	cat	sat	**on**	the	mat	$P(w_4 \mid w_3, w_2)$
the	cat	sat	on	**the**	mat	$P(w_5 \mid w_4, w_3)$
the	cat	sat	on	the	**mat**	$P(w_6 \mid w_5, w_4)$

Slide Credit: Piotr Mirowski

The Chain Rule

$$P(w_1, w_2, \ldots, w_{T-1}, w_T) = \prod_{t=1}^{T} P(w_t | w_{t-1}, w_{t-2}, \ldots, w_1)$$

the	cat	sat	on	the	mat	$P(w_1)$
the	**cat**	sat	on	the	mat	$P(w_2 \mid w_1)$
the	cat	**sat**	on	the	mat	$P(w_3 \mid w_2, w_1)$
the	cat	sat	**on**	the	mat	$P(w_4 \mid w_3, w_2, w_1)$
the	cat	sat	on	**the**	mat	$P(w_5 \mid w_4, w_3, w_2, w_1)$
the	cat	sat	on	the	**mat**	$P(w_6 \mid w_5, w_4, w_3, w_2, w_1)$

Slide Credit: Piotr Mirowski

Proof

$$\sum_{w_1,\ldots,w_T} p(w_1,\ldots,w_T) = 1$$

$$\sum_{w_1,\ldots,w_T} \prod_{t=1}^{T} p(w_t|w_1,\ldots,w_{t-1}) =$$

$$\sum_{w_1,\ldots,w_T} p(w_T|w_1,\ldots,w_{T-1}) \prod_{t=1}^{T-1} p(w_t|w_1,\ldots,w_{t-1}) =$$

$$\sum_{w_1,\ldots,w_{T-1}} p(w_{T-1}|w_1,\ldots,w_{T-2}) \prod_{t=1}^{T-2} p(w_t|w_1,\ldots,w_{t-1}) =$$

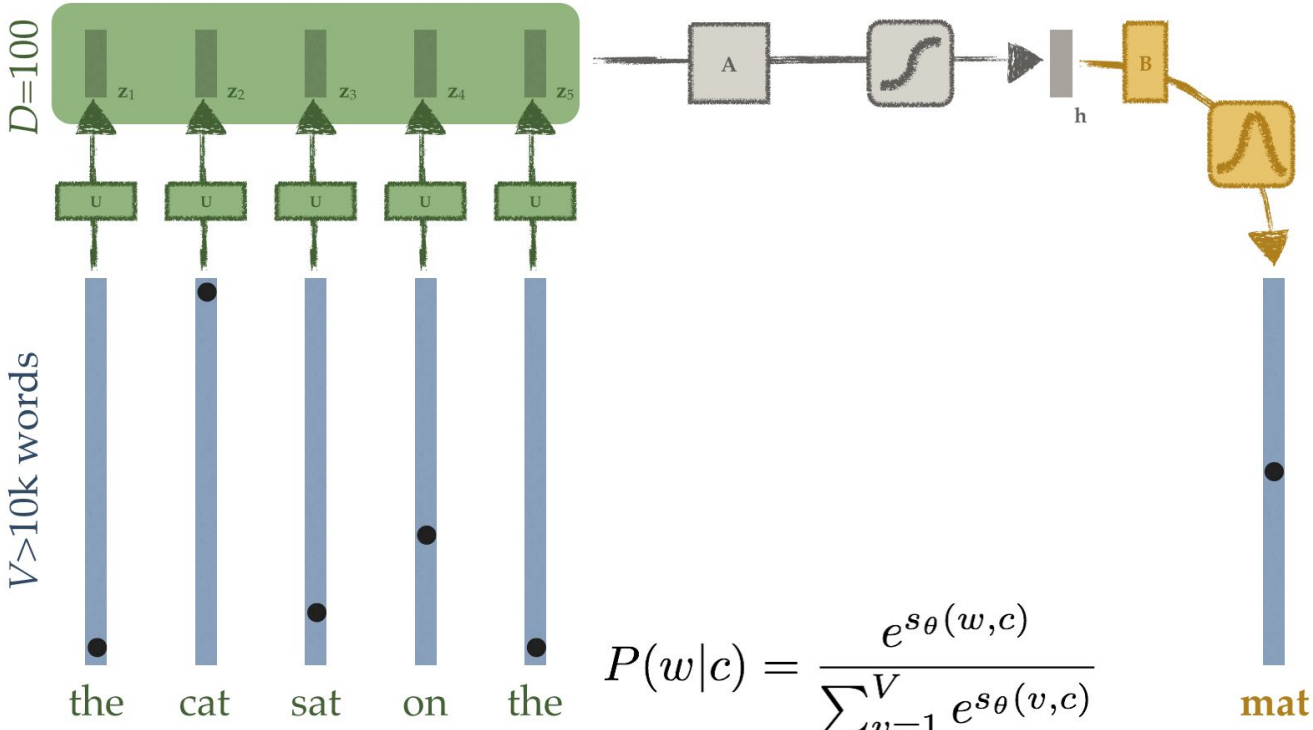
$$\ldots$$

$$\sum_{w_1,\ldots,w_T} \prod_{t=1}^{T} p(w_t|w_1,\ldots,w_{t-1}) = 1$$

A Key Insight: vectorizing context

[Yoshua Bengio et al. (2001, 2003), "A Neural Probabilistic Language Model", *JMLR*;
Andriy Mnih and Geoff Hinton, "Three new graphical models for statistical language modeling", *ICML*]

$$p(w_t|w_1,\ldots,w_{t-1}) = p_\theta(w_t|f_\theta(w_1,\ldots,w_{t-1}))$$

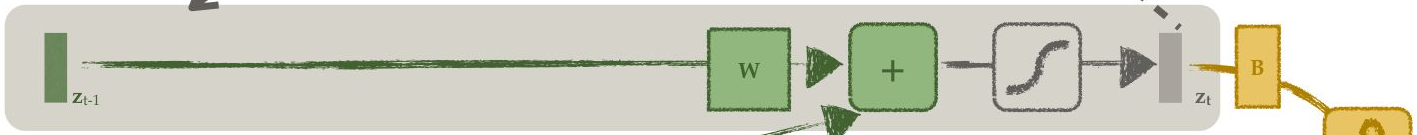


$$P(w|c) = \frac{e^{s_\theta(w,c)}}{\sum_{v=1}^{V} e^{s_\theta(v,c)}}$$

Slide Credit: Piotr Mirowski

Recurrent Neural Network Language Models

[Jeffrey L Elman (1991) "Distributed representations, simple recurrent networks and grammatical structure", *Machine Learning*; Tomas Mikolov et al. (2010) "Recurrent neural network based language model", *INTERSPEECH*]



"persistent memory":
state variable
for arbitrarily
long contexts

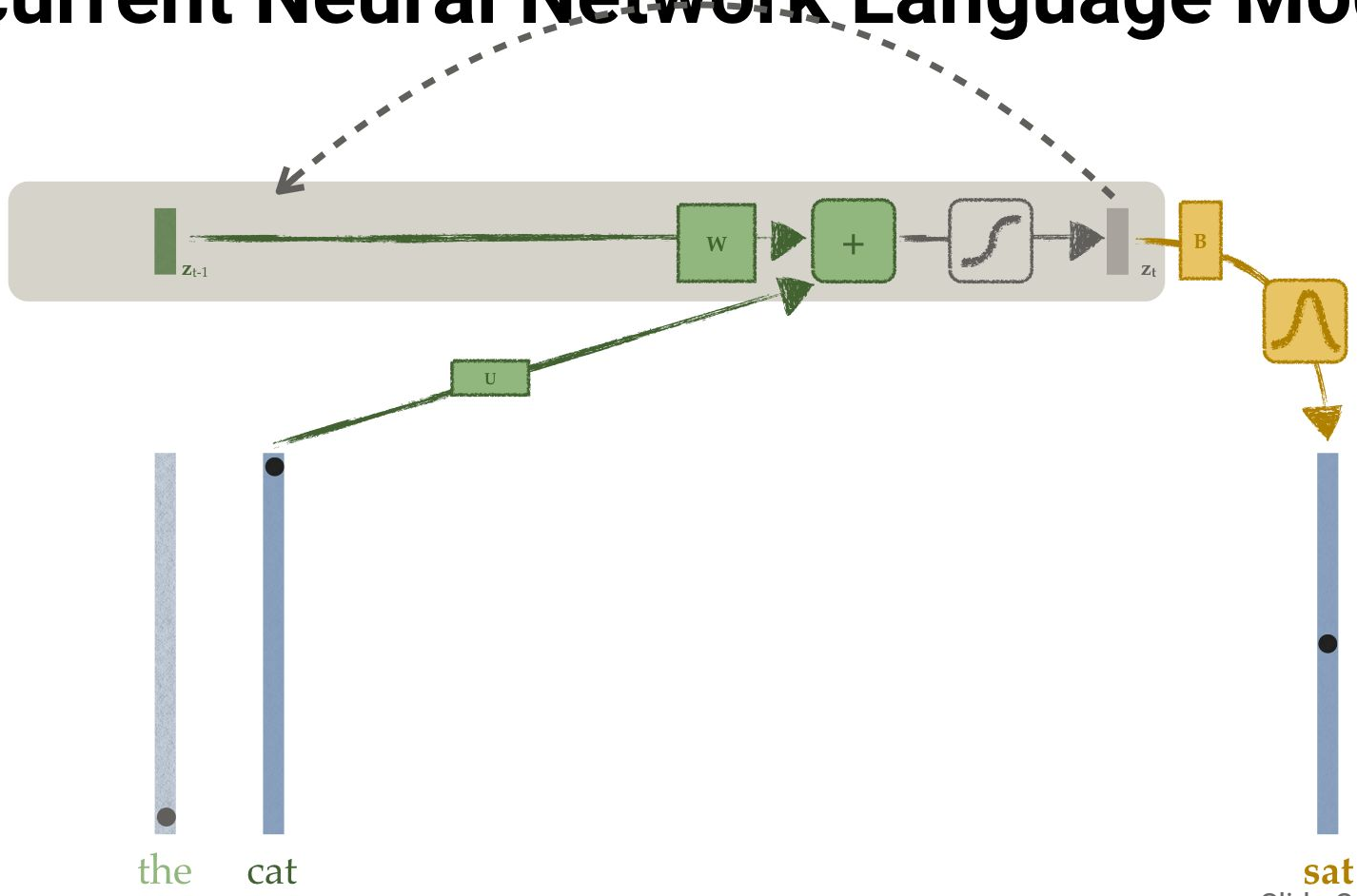
$$z_t = \tanh(W z_{t-1} + U w_t)$$
$$p(w_{t+1}) = softmax(B z_t)$$

the

cat

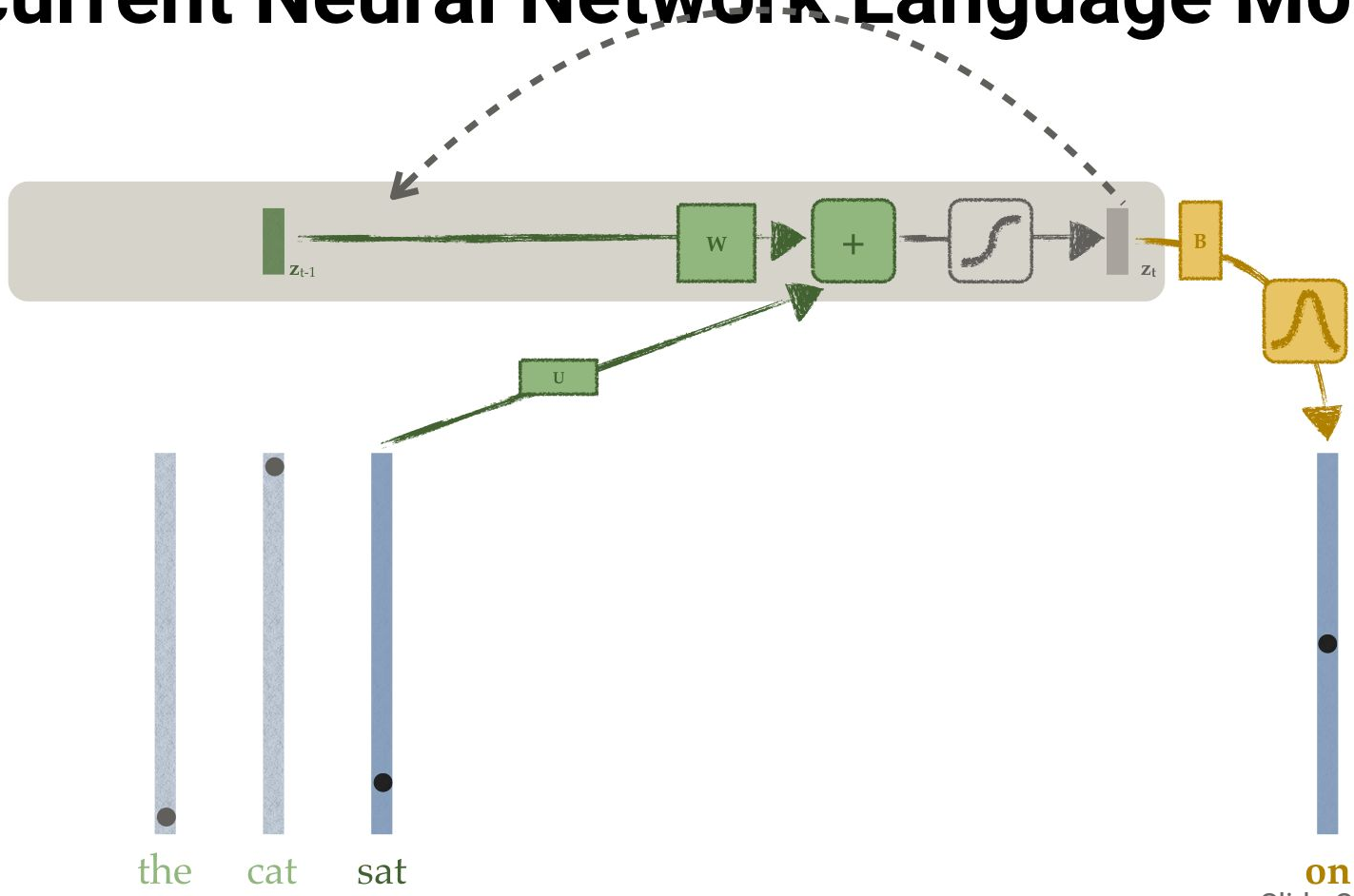
Slide Credit: Piotr Mirowski

Recurrent Neural Network Language Models



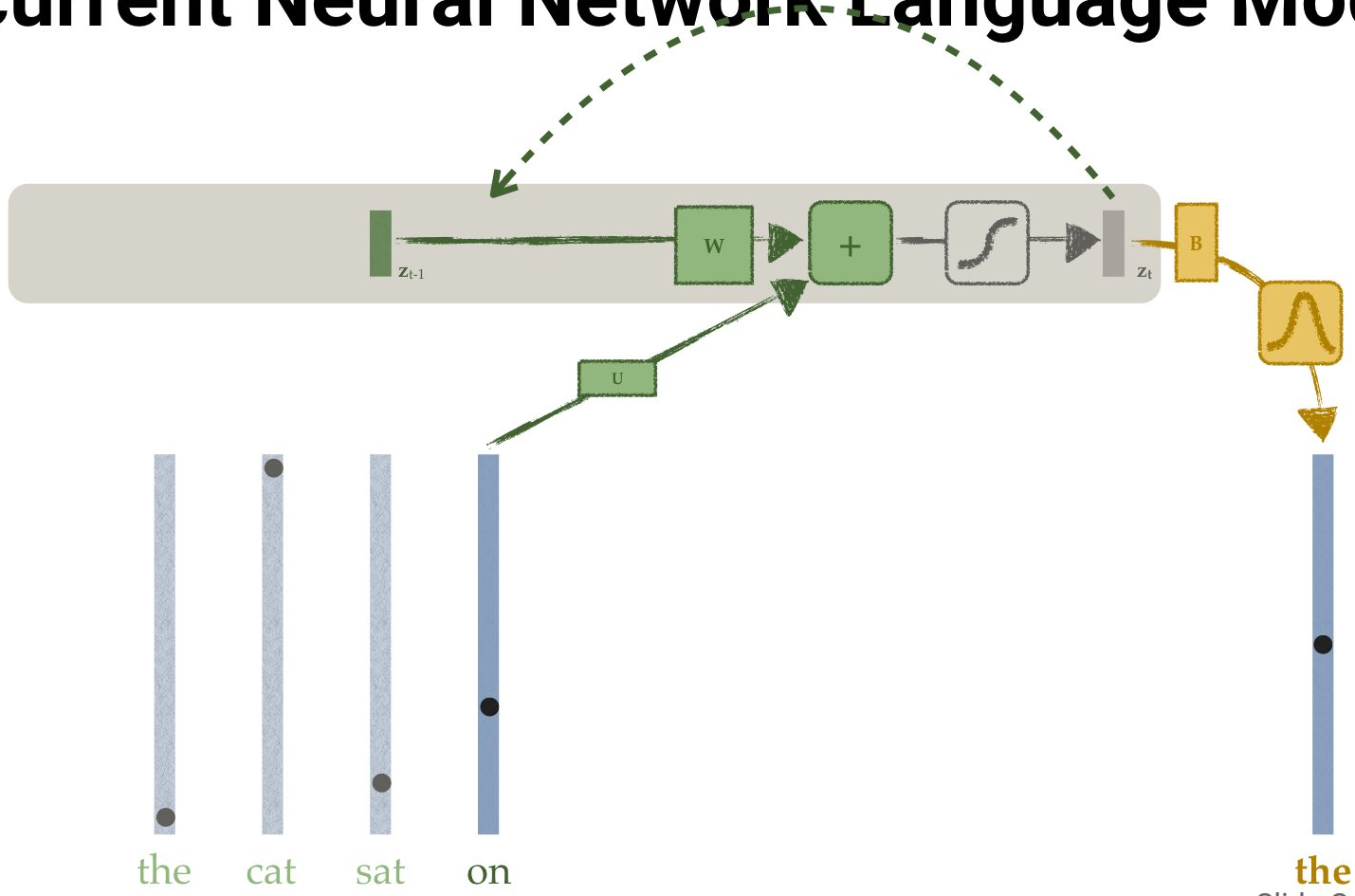
Slide Credit: Piotr Mirowski

Recurrent Neural Network Language Models



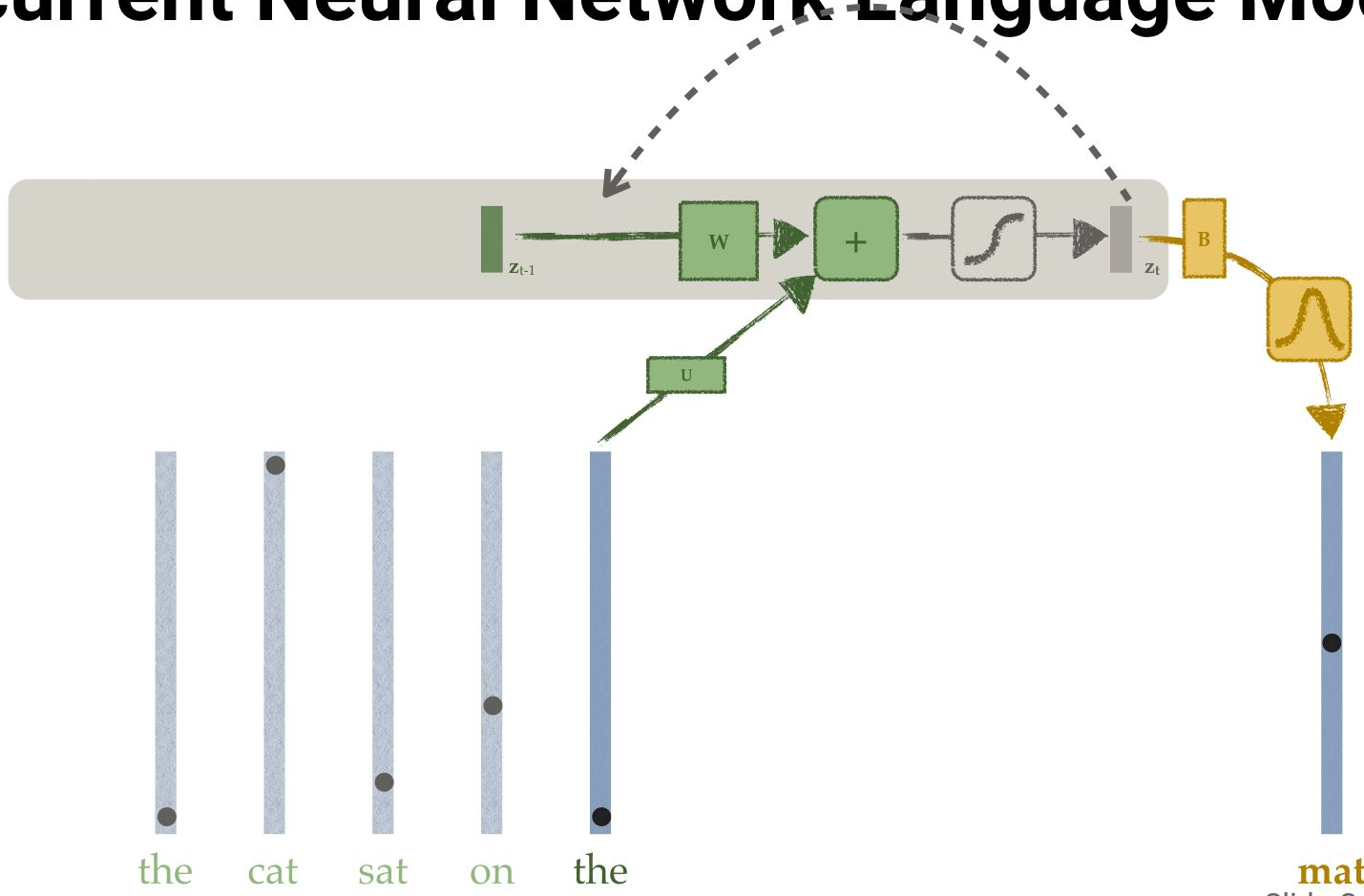
Slide Credit: Piotr Mirowski

Recurrent Neural Network Language Models



Slide Credit: Piotr Mirowski

Recurrent Neural Network Language Models

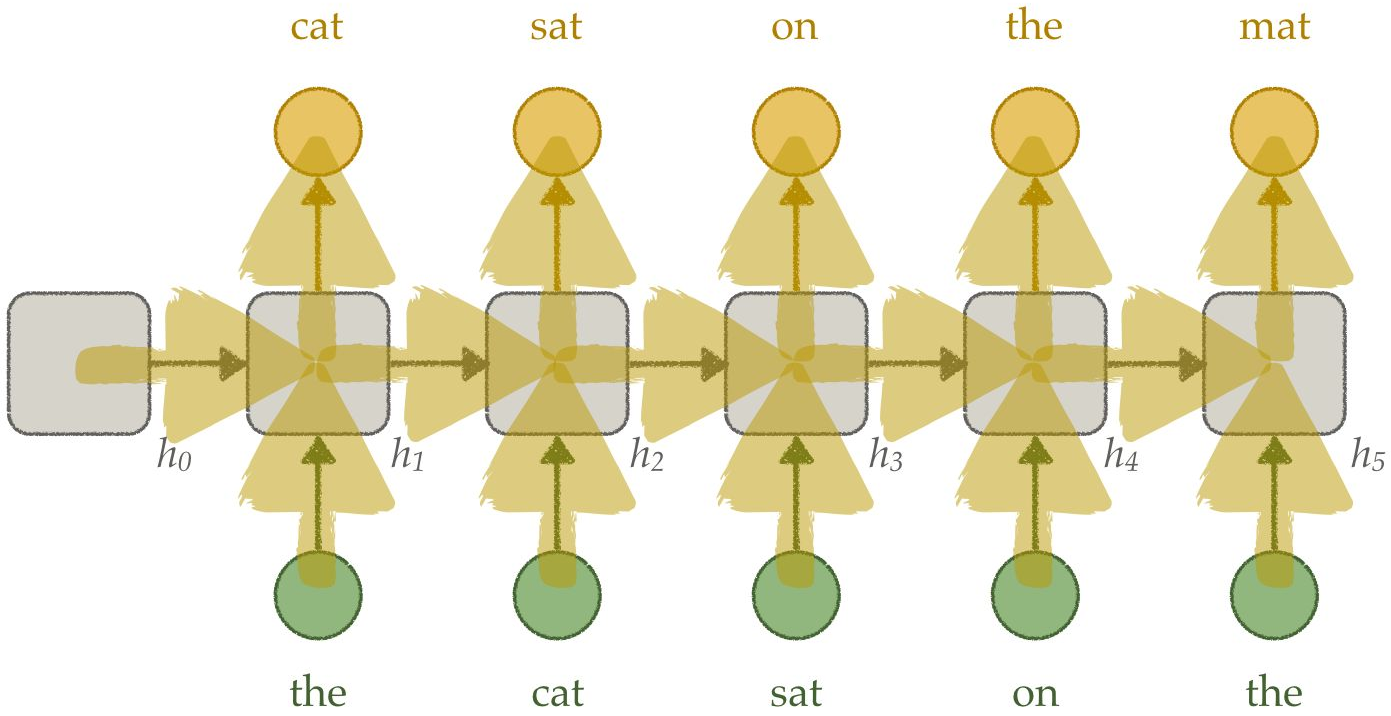


Slide Credit: Piotr Mirowski

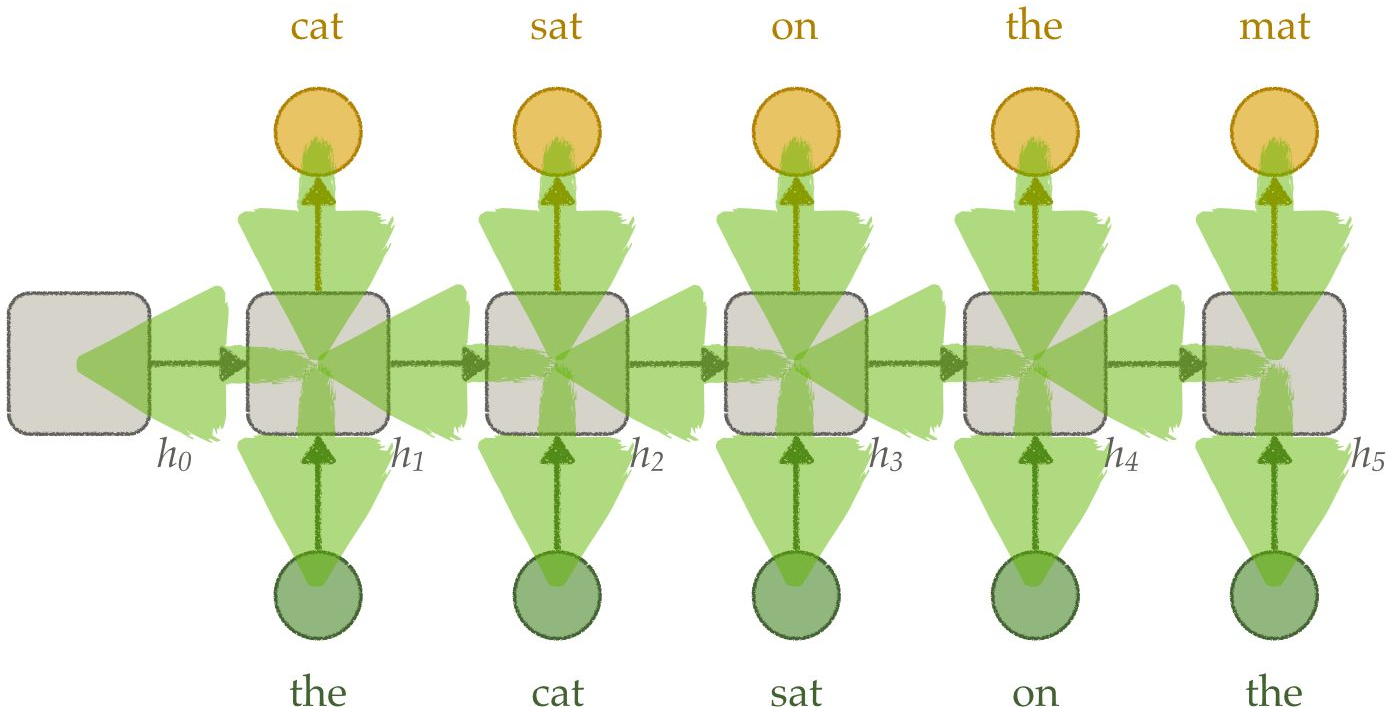
What do we Optimize?

$$\theta^* = \arg\max_{\theta} E_{w \sim data} \log P_\theta(w_1, \ldots, w_T)$$

Recurrent Neural Network Language Models



Recurrent Neural Network Language Models



Deriving BPTT

$$z_t = \tanh(W z_{t-1} + U w_t)$$
$$p(w_{t+1}) = softmax(B z_t)$$

$$L(W, U, B) = -\sum_{t=1}^{T} \log p_{W,U,B}(w_t | w_1, \ldots, w_{t-1})$$

$$L_t = -\log p(w_t | w_1, \ldots, w_{t-1})$$

Deriving BPTT - B (softmax)

$$L_t = -\log p(w_t | w_1, \ldots, w_{t-1})$$

$$\frac{\delta L_t}{\delta B} = \frac{\delta L_t}{\delta p_t} \frac{\delta p_t}{\delta l_t} \frac{\delta l_t}{\delta B}$$

$$\frac{\delta L_t}{\delta B} = (p_t - label_t) z_t^T$$

$$\frac{\delta L}{\delta B} = \sum_{t=1}^{T} (p_t - label_t) z_t^T$$

Deriving BPTT - W (recurrent)

$$z_t = \tanh(W z_{t-1} + U w_t)$$
$$p(w_{t+1}) = softmax(B z_t)$$

$$\frac{\delta L_t}{\delta W} = \frac{\delta L_t}{\delta p_t}\frac{\delta p_t}{\delta z_t}\frac{\delta z_t}{\delta W}$$

$$\frac{\delta L_t}{\delta W} = \sum_{k=1}^{t} \frac{\delta L_t}{\delta p_t}\frac{\delta p_t}{\delta z_t}\frac{\delta z_t}{\delta z_k}\frac{\delta z_k}{\delta W}$$

Part III.B Vanishing Gradients

Optimization Tricks

Vanishing (& exploding) Gradients

[Sepp Hochreiter (1991) “Untersuchungen zu dynamischen neuronalen Netzen”, *Diploma TUM*;

Yoshua Bengio et al. (1994) “Learning Long-Term Dependencies with Gradient Descent is Difficult”, *IEEE Transactions on Neural Networks*]

- n-grams have VERY strong vanishing gradients :)
- RNNs *can* encode long term, but *learning* is hard
- Does long-term ALWAYS matter?
  - Language has strong correlations short term (but also long!)
  - Seq2seq models *must* remember (later)
  - Learning programs

A simple example

$h_t = w * h_{t-1}$

$h_t = w^t * h_0$

$h_t \rightarrow \infty$ if $|w| > 1$

$h_t \rightarrow 0$ if $|w| < 1$

A simple example

$h_t = w * h_{t-1}$

$h_t = w^t * h_0$

$h_t \rightarrow \infty$ if $|w| > 1$

$h_t \rightarrow 0$ if $|w| < 1$

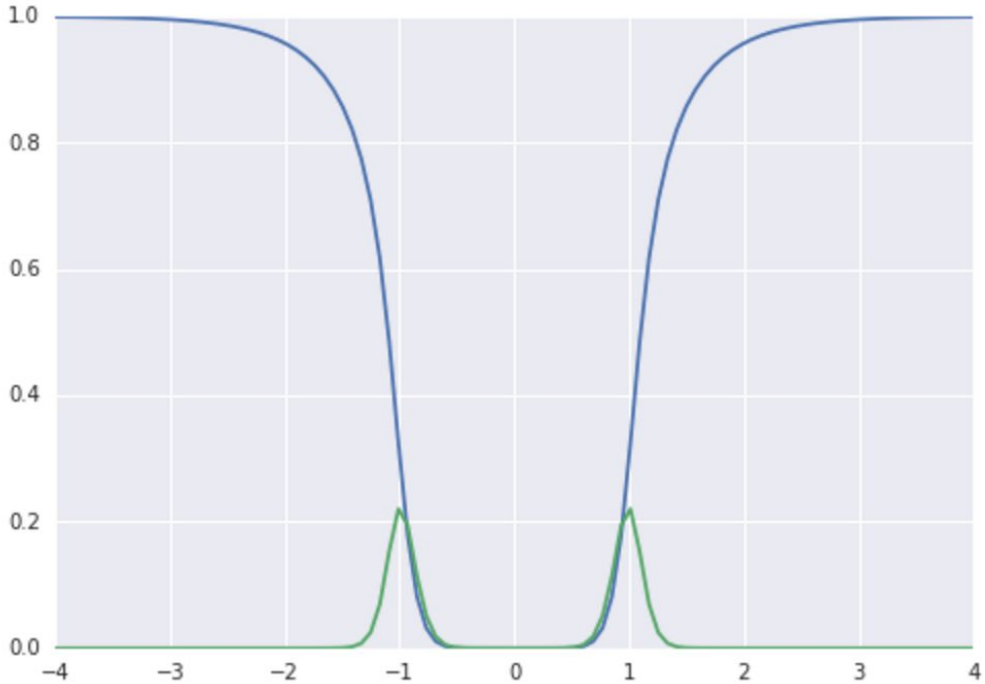
$h_t = \tanh(w * h_{t-1})$

$h_t \rightarrow$ bounded for all w :)

$\delta h_t / \delta h_{t-T} = (1-\tanh^2(w * h_{t-1}))*w*\delta h_{t-1} / \delta h_{t-T}$

A simple example

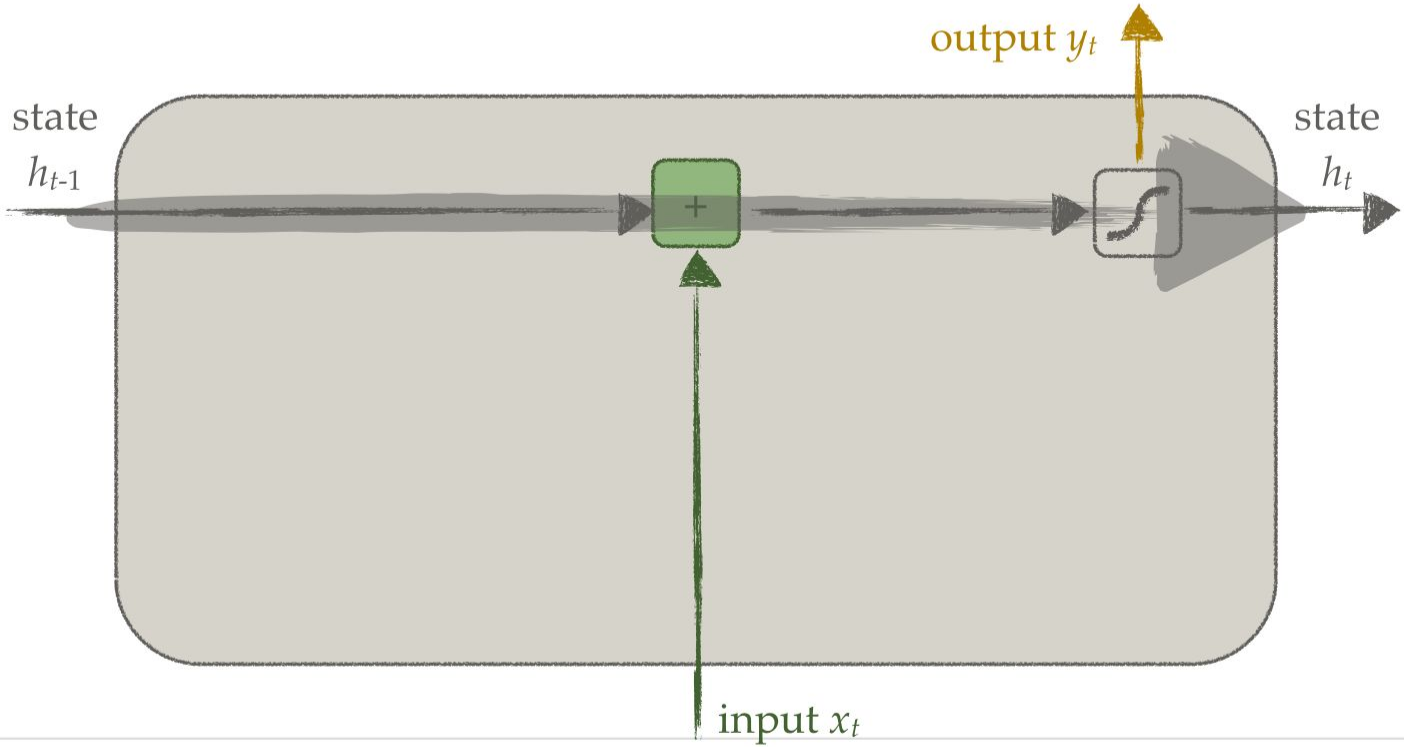
```
import numpy as np
from matplotlib import pyplot

def forward_backward_prop(w, T):
  hs = [0.5]
  for _ in range(T):
    hs.append(np.tanh(w*hs[-1]))
  dh = 1
  for t in range(T):
    dh = (1-hs[-1-t]*hs[-1-t])*w*dh
  return hs[-1], dh

T = 10
wlim = 4

ws = np.linspace(-wlim, wlim, 100)
res = []
for w in ws:
  res.append(forward_backward_prop(w,T))

pyplot.plot(ws, [r[0] for r in res])
pyplot.plot(ws, [r[1] for r in res])
```



Part III.C LSTMs (& GRUs)

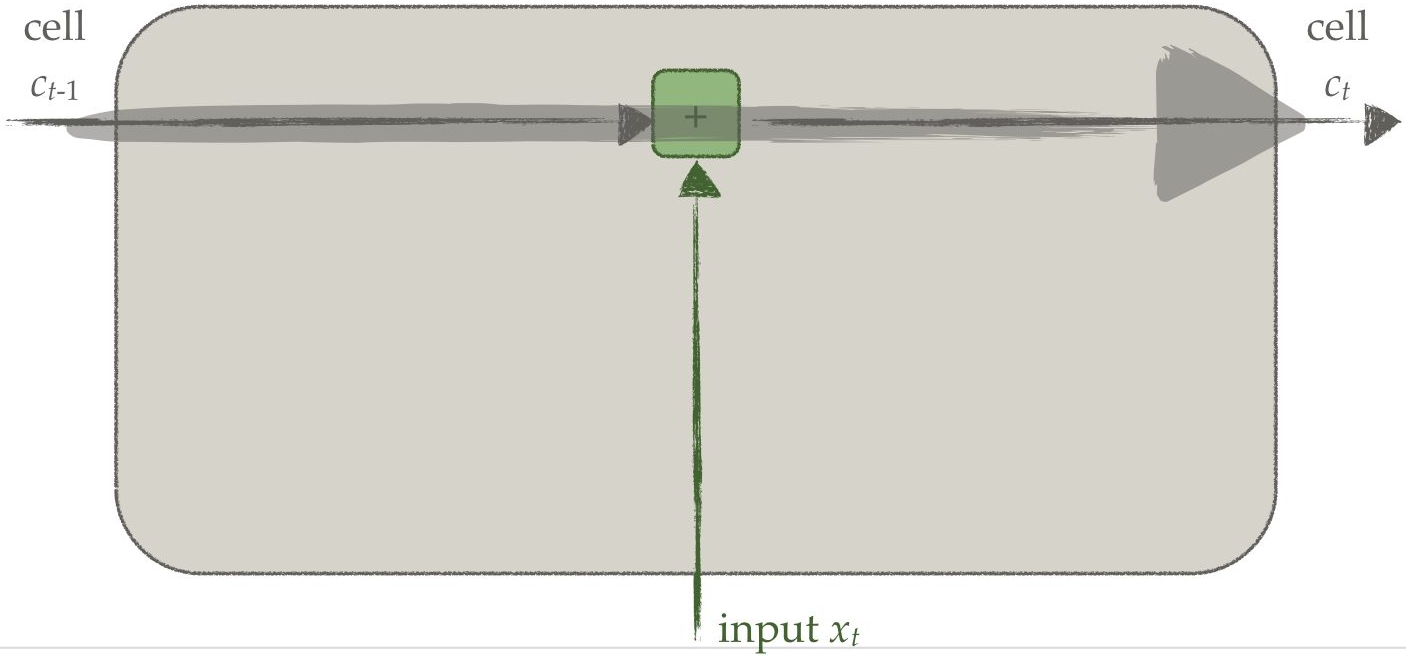
The Model Architecture

LSTMs



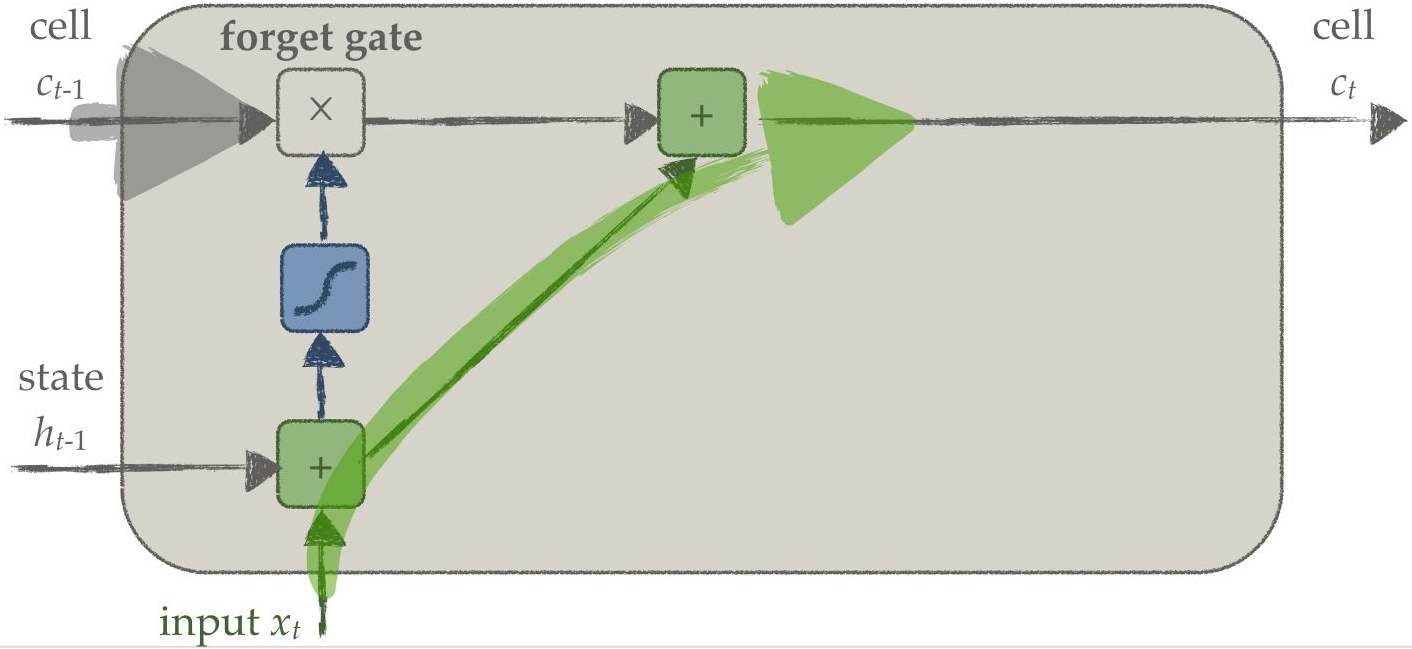
Requirement #1: linear cell

[Sepp Hochreiter and Jürgen Schmidhuber (1997) “Long Short-Term Memory”, *Neural Computation*; Alex Graves (2013a) “Generating sequences with recurrent neural networks”, *arXiv* 1308.0850]

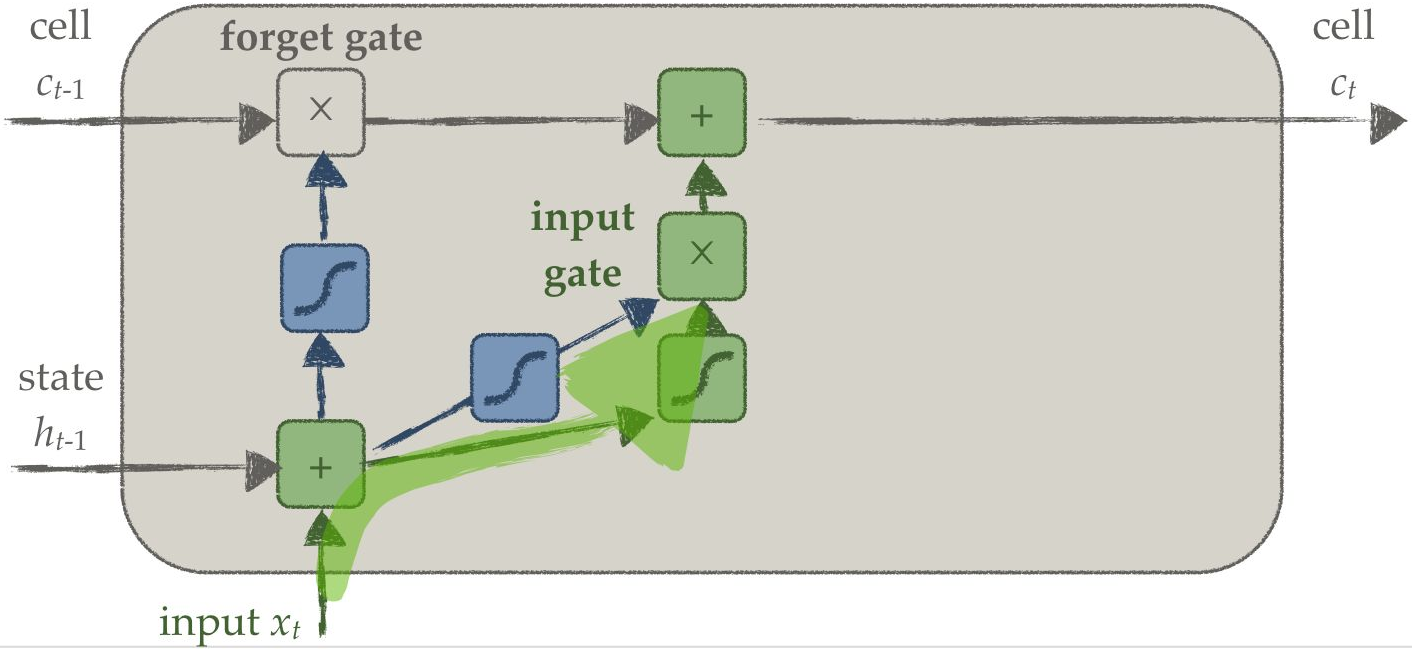


Requirement #2: forget information

[Sepp Hochreiter and Jürgen Schmidhuber (1997) “Long Short-Term Memory”, *Neural Computation*; Alex Graves (2013a) “Generating sequences with recurrent neural networks”, *arXiv* 1308.0850]

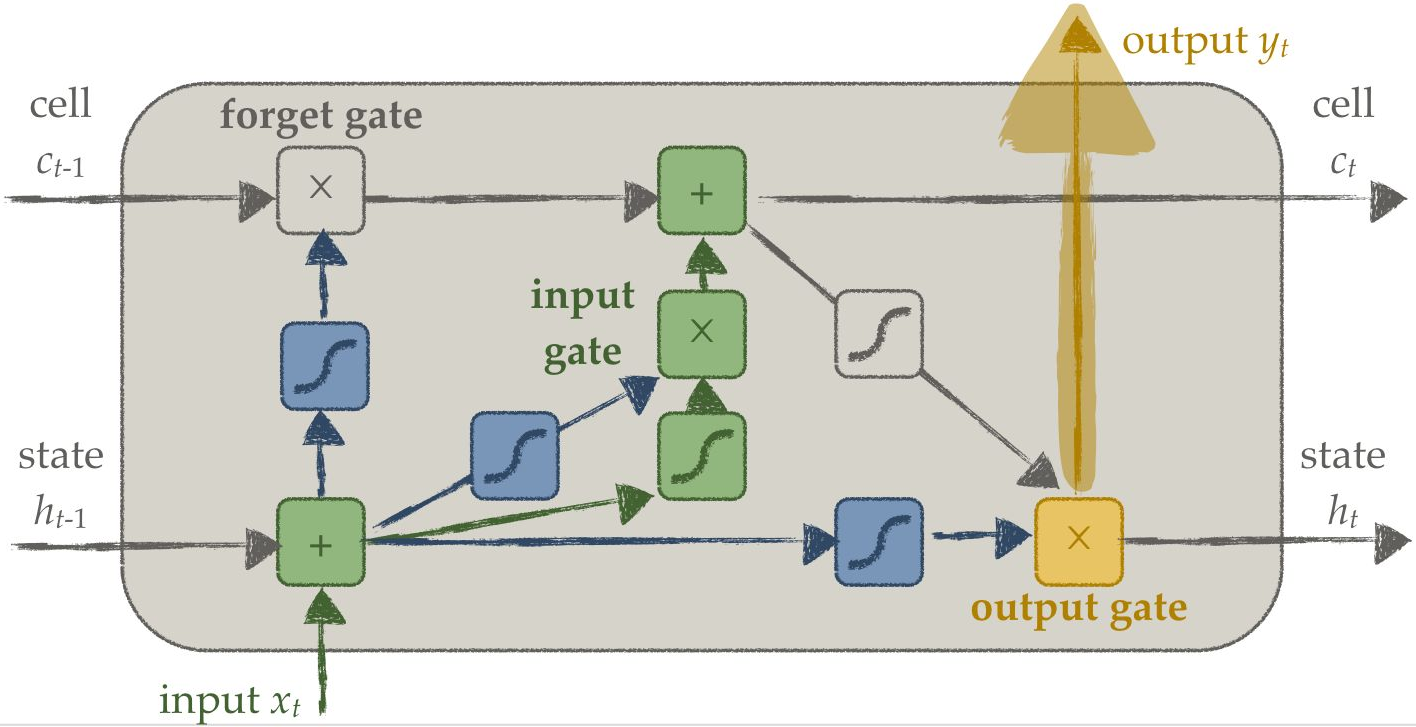


Requirement #3: ignore inputs



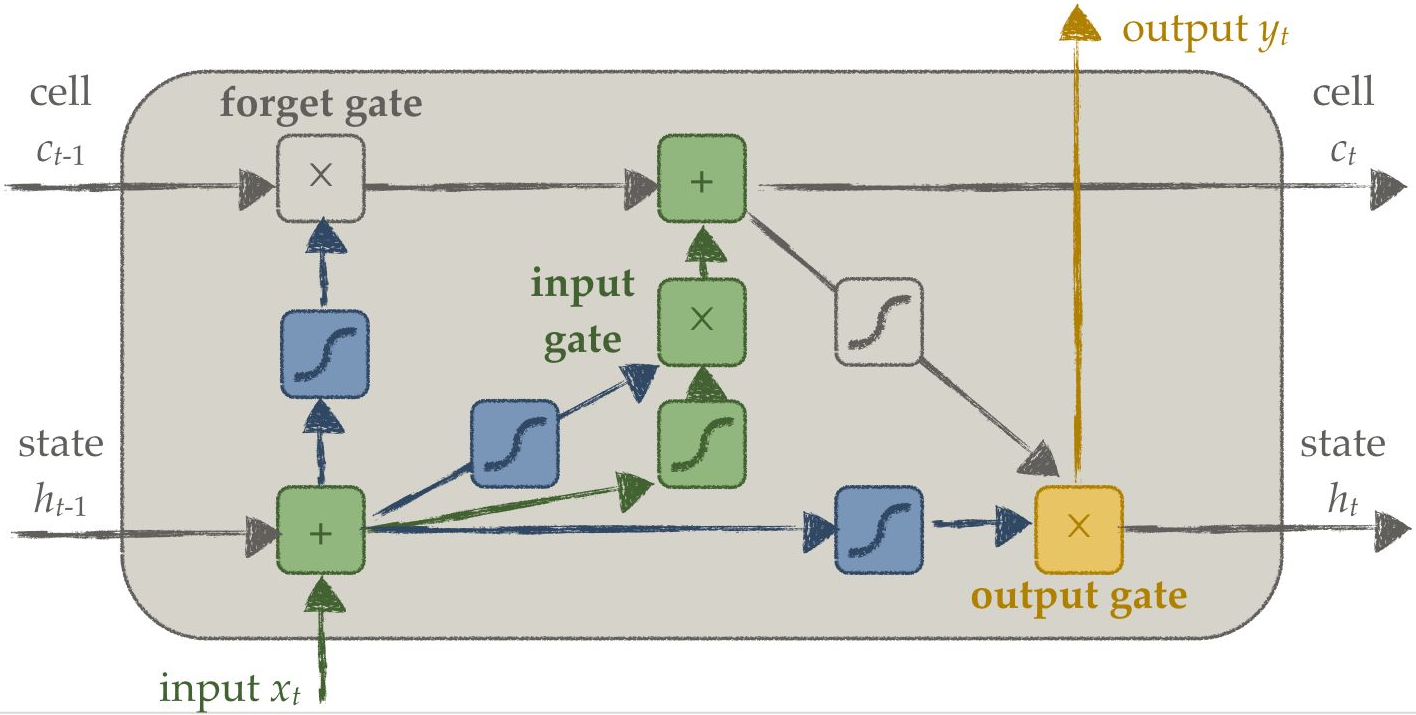
[Hochreiter and Schmidhuber (1997); Graves (2013a)]

Requirement #4: control outputs



[Hochreiter and Schmidhuber (1997); Graves (2013a)]

Long Short-Term Memory (LSTM)



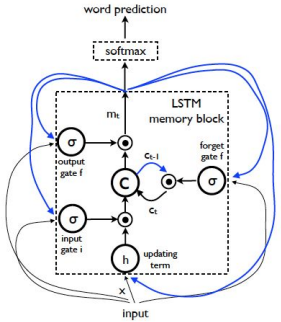
[Hochreiter and Schmidhuber (1997); Graves (2013a)]

LSTM [Hochreiter et al, 1997][Gers et al, 1999]

$$
\begin{aligned}
i_t &= W_{ix}x_t + W_{ih}h_{t-1} + b_i \\
j_t &= W_{jx}x_t + W_{jh}h_{t-1} + b_j \\
f_t &= W_{fx}x_t + W_{fh}h_{t-1} + b_f \\
o_t &= W_{ox}x_t + W_{oh}h_{t-1} + b_o \\
c_t &= \sigma(f_t) \odot c_{t-1} + \sigma(i_t) \odot \tanh(j_t) \\
h_t &= \sigma(o_t) \odot \tanh(c_t)
\end{aligned}
$$

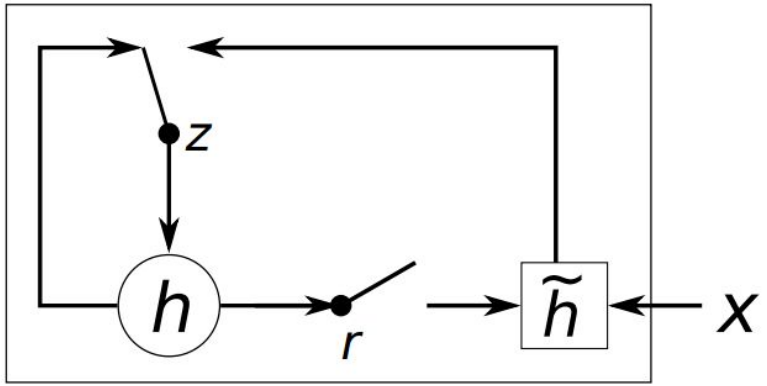
Enables long term dependencies to flow


```
def __call__(self, inputs, state, scope=None):
  """Long short-term memory cell (LSTM)."""
  with vs.variable_scope(scope or type(self).__name__):  # "BasicLSTMCell"
    # Parameters of gates are concatenated into one multiply for efficiency.
    c, h = array_ops.split(1, 2, state)
    concat = linear([inputs, h], 4 * self._num_units, True)

    # i = input_gate, j = new_input, f = forget_gate, o = output_gate
    i, j, f, o = array_ops.split(1, 4, concat)

    new_c = c * sigmoid(f + self._forget_bias) + sigmoid(i) * tanh(j)
    new_h = tanh(new_c) * sigmoid(o)

  return new_h, array_ops.concat(1, [new_c, new_h])
```

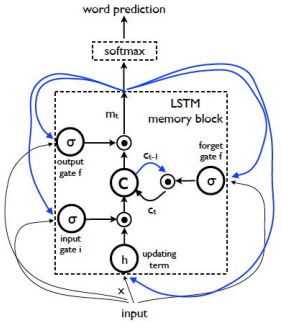


GRU, aka simplified LSTM [Cho et al, 2014]

$$z_t = \sigma(x_t U_z + h_{t-1} W_z + b_z)$$
$$r_t = \sigma(x_t U_r + h_{t-1} W_r + b_r)$$
$$\tilde{h} = \tanh(x_t U_h + (h_{t-1} \odot r_t) W_h + b_h)$$
$$h_t = (1 - z_t) \odot \tilde{h} + z_t \odot h_{t-1}$$



Enables long term dependencies to flow



How about exploding gradients? [Pascanu et al, 2013]

Algorithm 1 Pseudo-code for norm clipping

$\hat{\mathbf{g}} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta}$

if $\|\hat{\mathbf{g}}\| \geq threshold$ **then**

$\hat{\mathbf{g}} \leftarrow \frac{threshold}{\|\hat{\mathbf{g}}\|} \hat{\mathbf{g}}$

end if

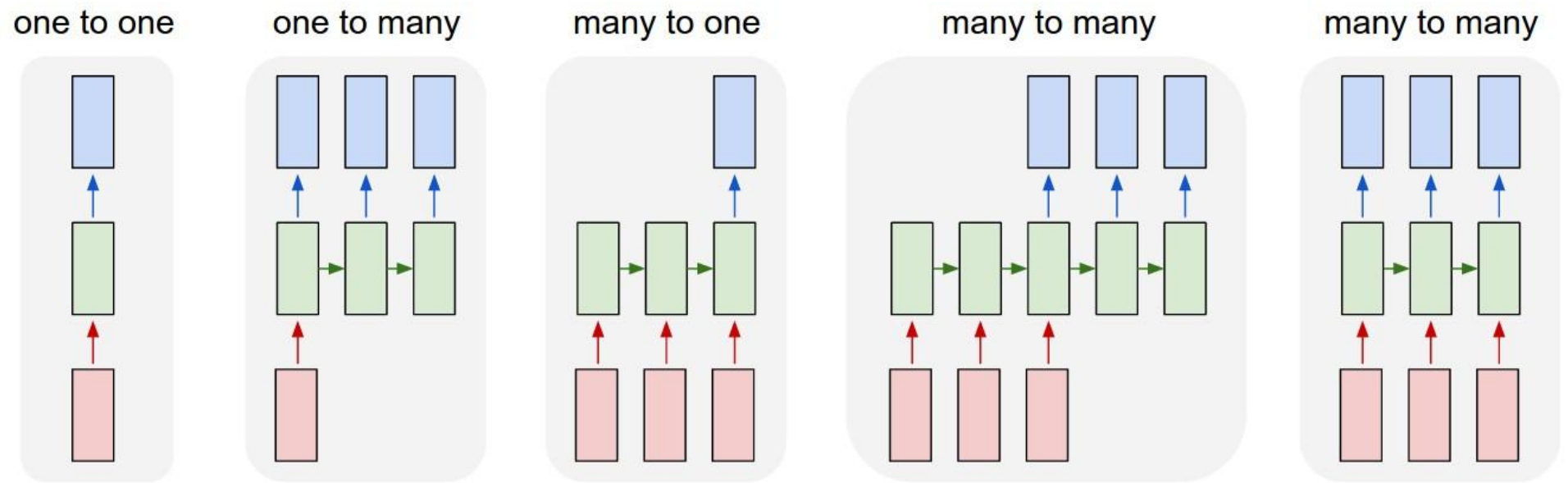
Summary

- Sequences are important, everything is a sequence
- Recurrent Nets are a way forward, but they pose some challenges
  - Chain Rule makes us not assume independence
  - Vectorizing context is key for feasibility / model size
  - Vanishing/Exploding gradients
    - LSTMs is better at this, but by no means “solves” all problems
  - Expensive to compute

Part IV: Recurrent Networks as Sequence Decoders

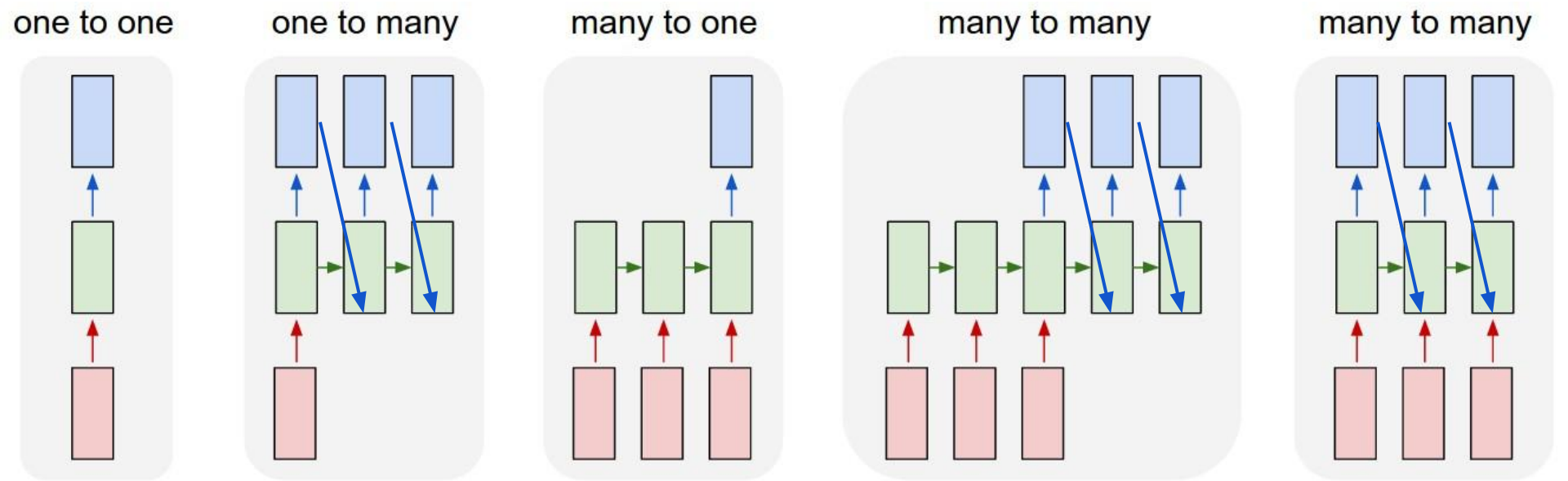
Sequence Models

http://karpathy.github.io/2015/05/21/rnn-effectiveness/



Sequence Models (Autoregressive)

http://karpathy.github.io/2015/05/21/rnn-effectiveness/



Mini-Turing Test

Some of the obese people lived five to eight years longer than others.

Abu Dhabi is going ahead to build solar city and no pollution city.

Or someone who exposes exactly the truth while lying.

VIERA , FLA . -- Sometimes, Rick Eckstein dreams about baseball swings.

For decades, the quintessentially New York city has elevated its streets to the status of an icon.

The lawsuit was captioned as United States ex rel.

Mini-Turing Test

Some of the obese people lived five to eight years longer than others.

Abu Dhabi is going ahead to build solar city and no pollution city.

Or someone who exposes exactly the truth while lying.

VIERA , FLA . -- Sometimes, Rick Eckstein dreams about baseball swings.

For decades, the quintessentially New York city has elevated its streets to the status of an icon.

The lawsuit was captioned as United States ex rel.

Mini-Turing Test

Some of the obese people lived five to eight years longer than others.

Abu Dhabi is going ahead to build solar city and no pollution city.

Or someone who exposes exactly the truth while lying.

VIERA , FLA . -- Sometimes, Rick Eckstein dreams about baseball swings.

For decades the quintessentially New York city has elevated its streets to the status of an icon.

The lawsuit was captioned as United States ex rel.

https://github.com/tensorflow/models/tree/master/lm_1b

Results

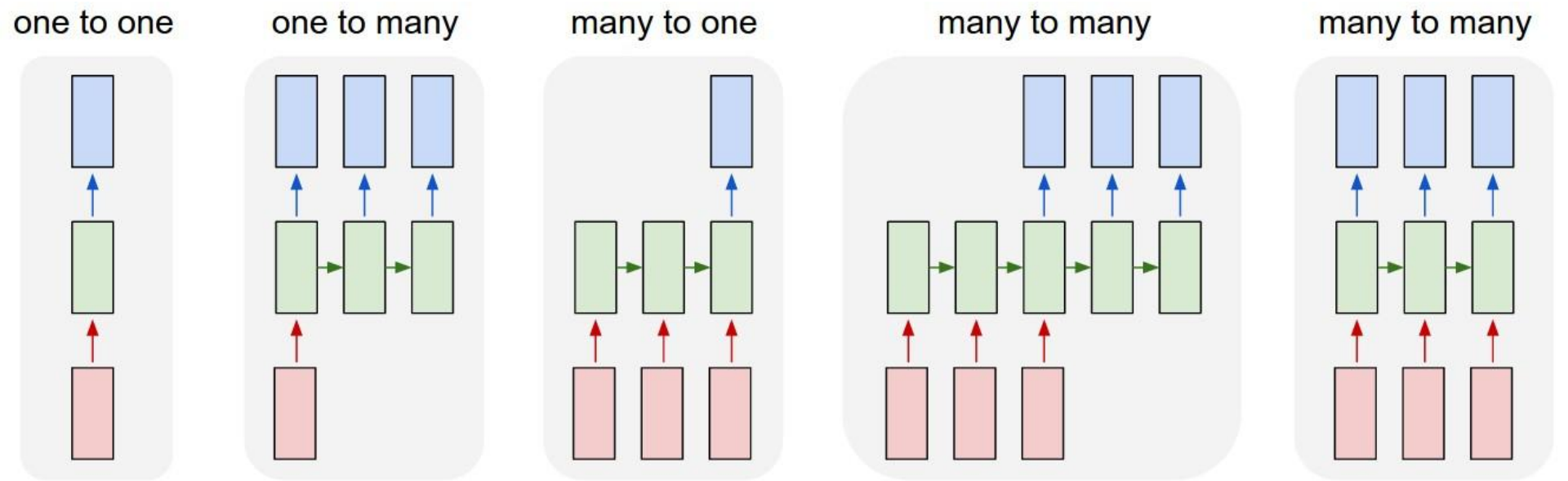
Measure: perplexity (~branching factor (low == good))

Model	Perplexity	#params (Billions)
5-gram (smoothed) (2013)	67.6	1.8
RNN + Maxent (2013)	51.3	20
Big Ensemble (2013)	43.8	LOTS
RNN (ICLR, 2015)	68.3	4.1
RNN + 5-gram (ICLR, 2015)	42.0	LOTS
Sparse NNMF (IS, 2015)	52.9	33 (!)
Ours (LSTM) (2016)	**30.0**	**1.0**
Ours (Ensemble) (2016)	**23.7**	LOTS

HUMANS: ???

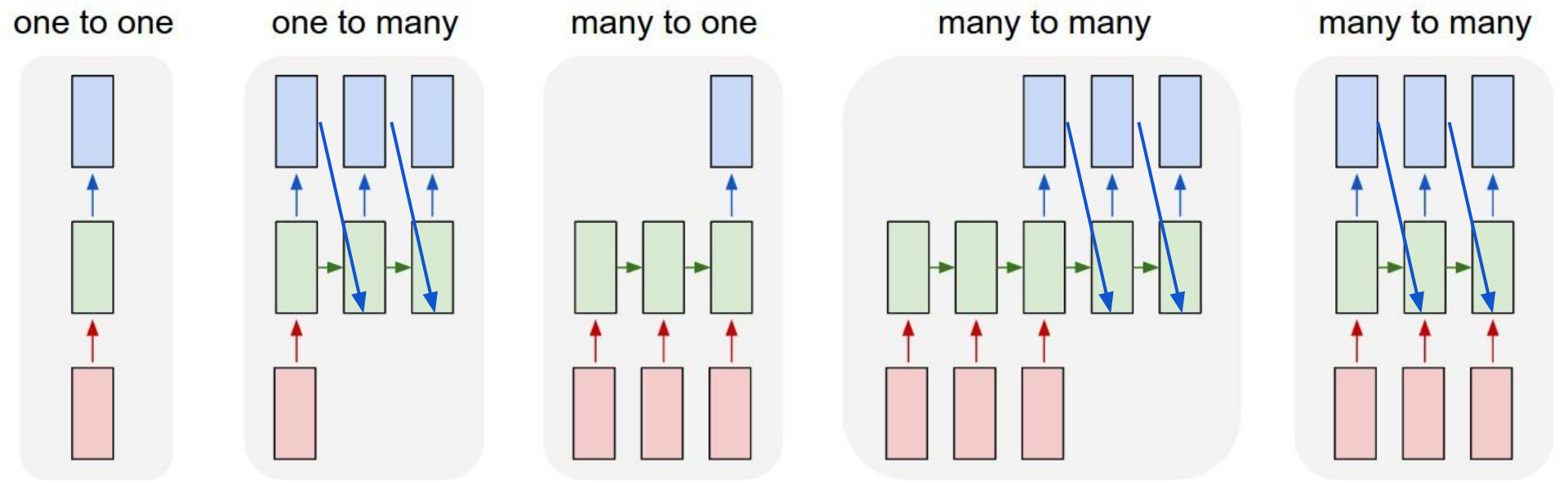
Sequence Models

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

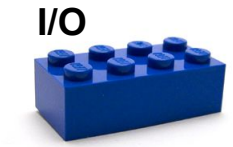


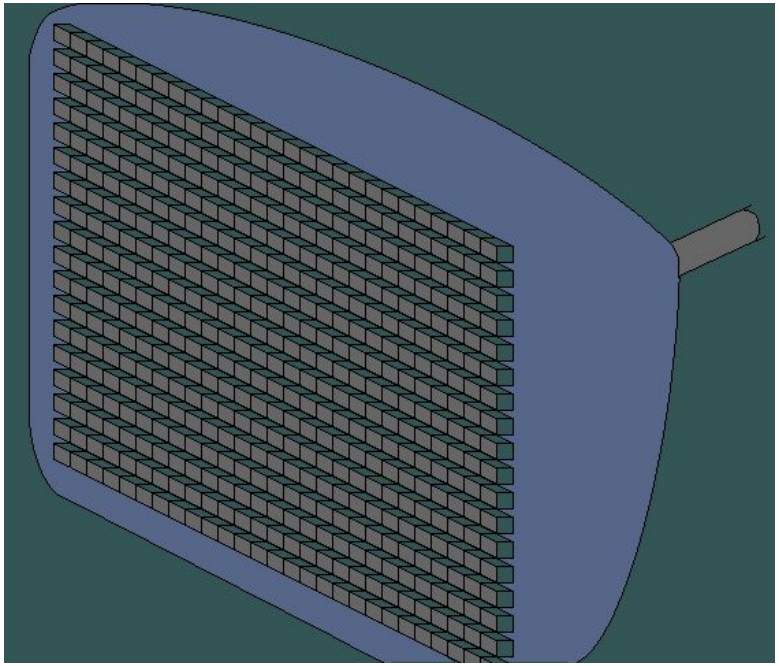
Sequence Models (Autoregressive)

http://karpathy.github.io/2015/05/21/rnn-effectiveness/



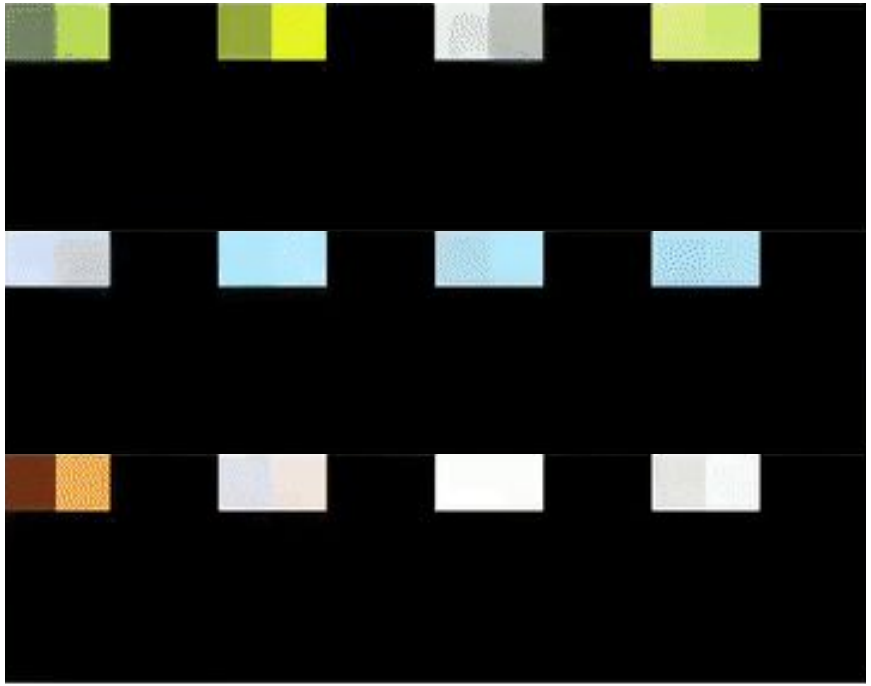
Modeling Images





Pixel-by-pixel

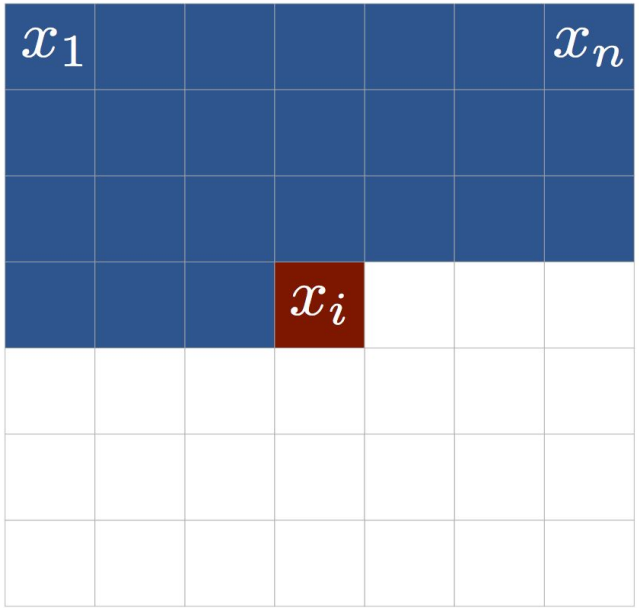
https://giphy.com/gifs/televsion-13epOe3Z06gHba



Group-by-group

Reed et al. “Parallel Multiscale Autoregressive Density Estimation.”

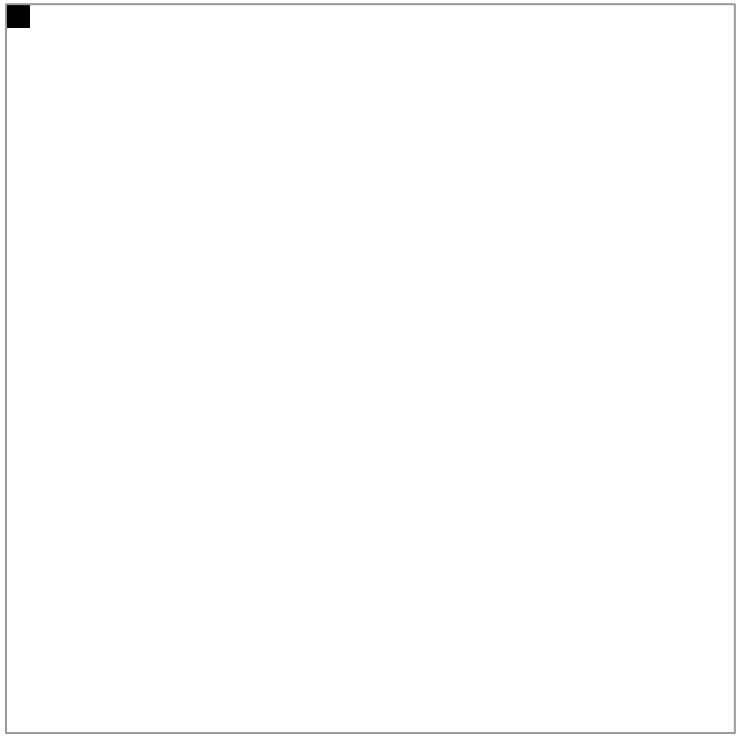
PixelRNN - Model



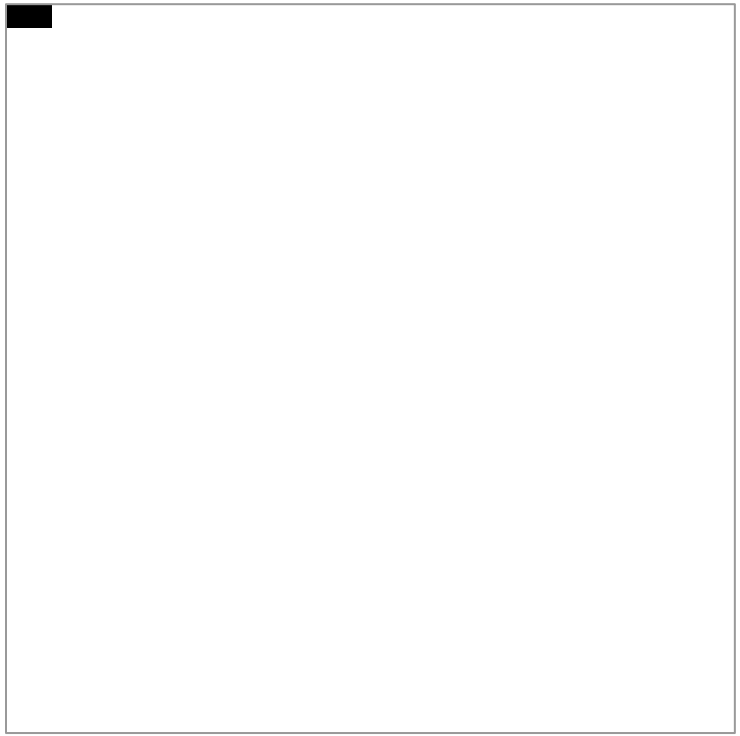
$$p(\mathbf{x}) = \prod_{i=1}^{n^2} p(x_i | x_1, \ldots, x_{i-1})$$

- Fully visible
- Similar to language models with RNNs
- Model pixels with Softmax

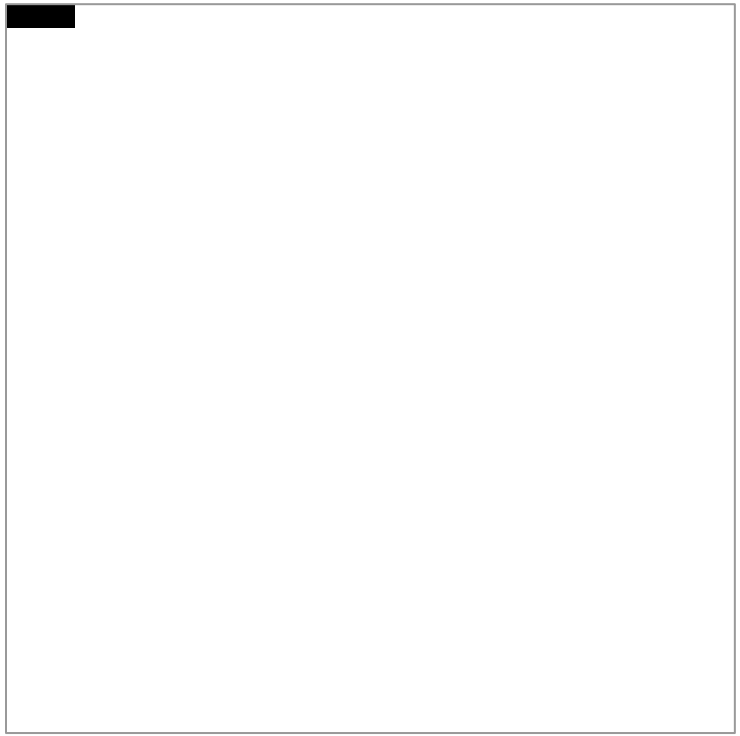
Softmax Sampling



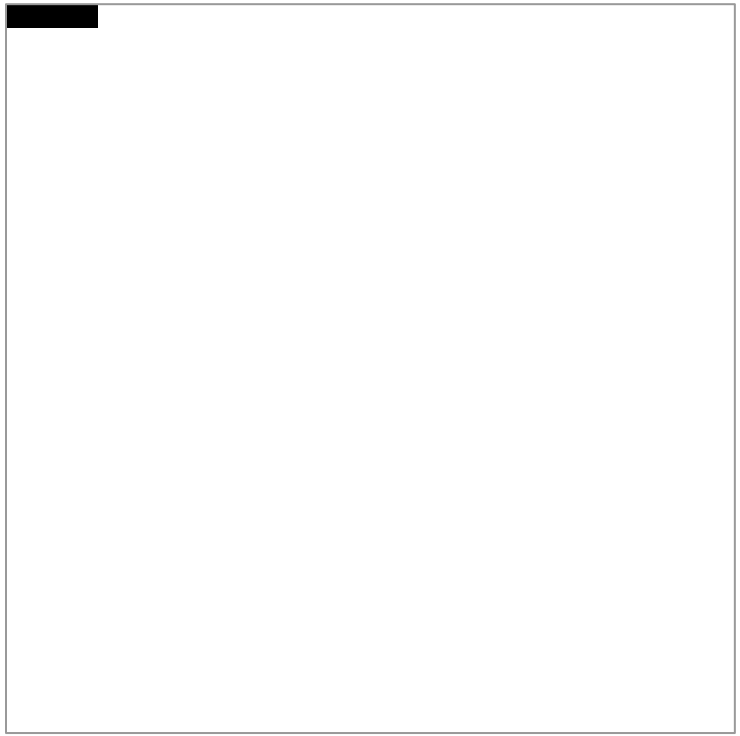
Softmax Sampling



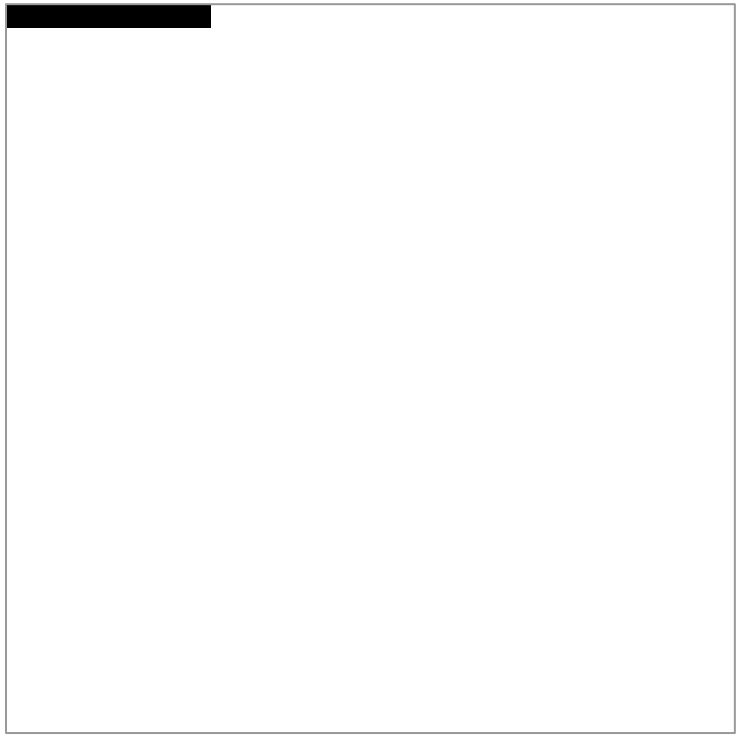
Softmax Sampling



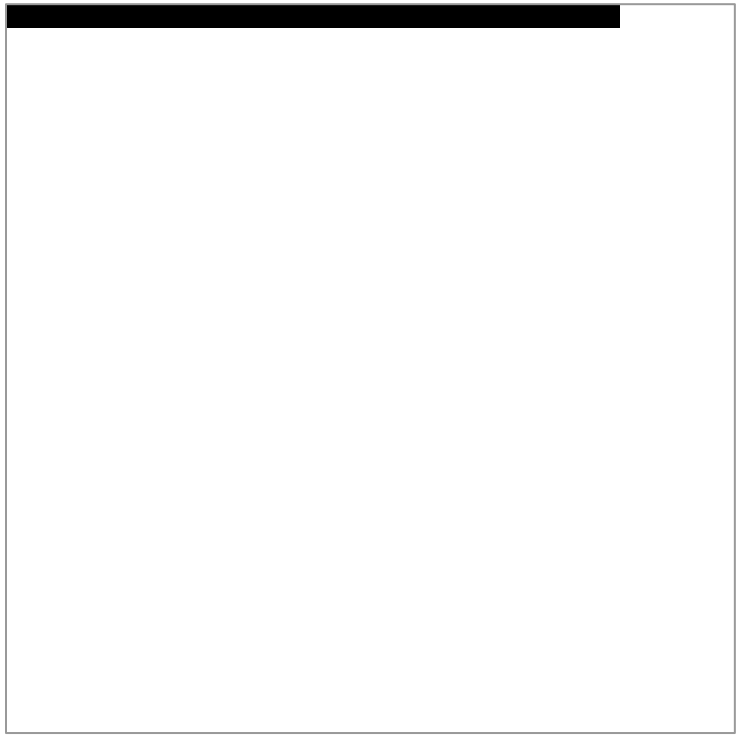
Softmax Sampling



Softmax Sampling



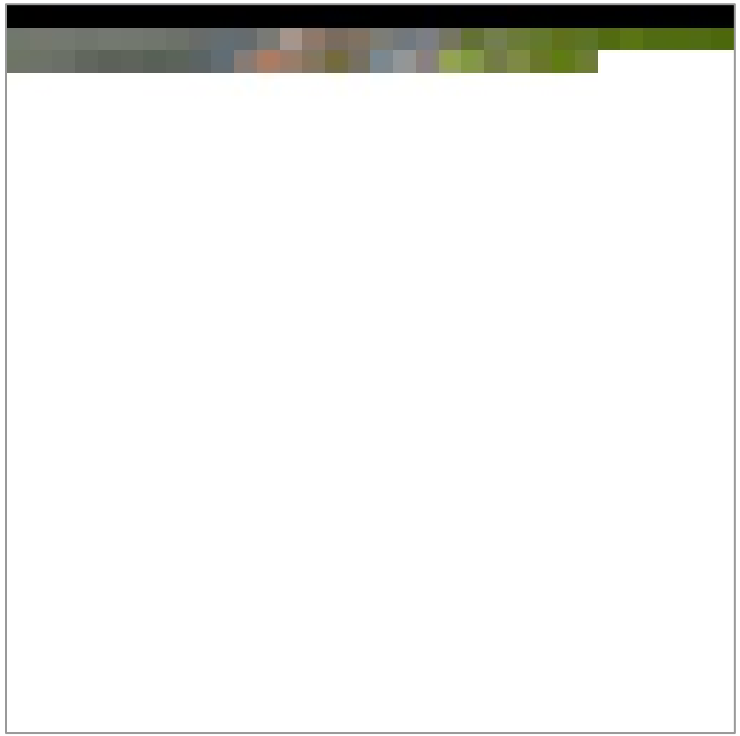
Softmax Sampling



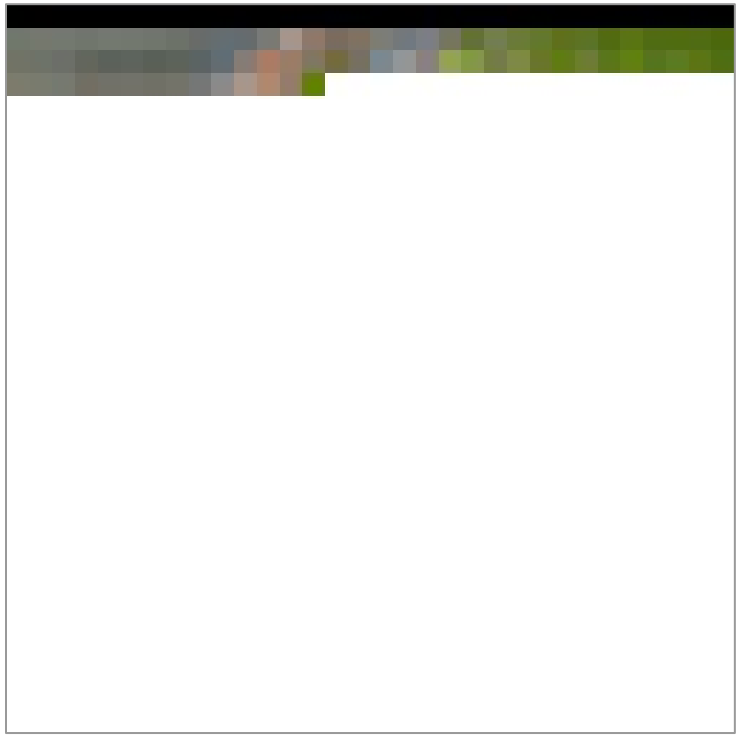
Softmax Sampling

Softmax Sampling

Softmax Sampling

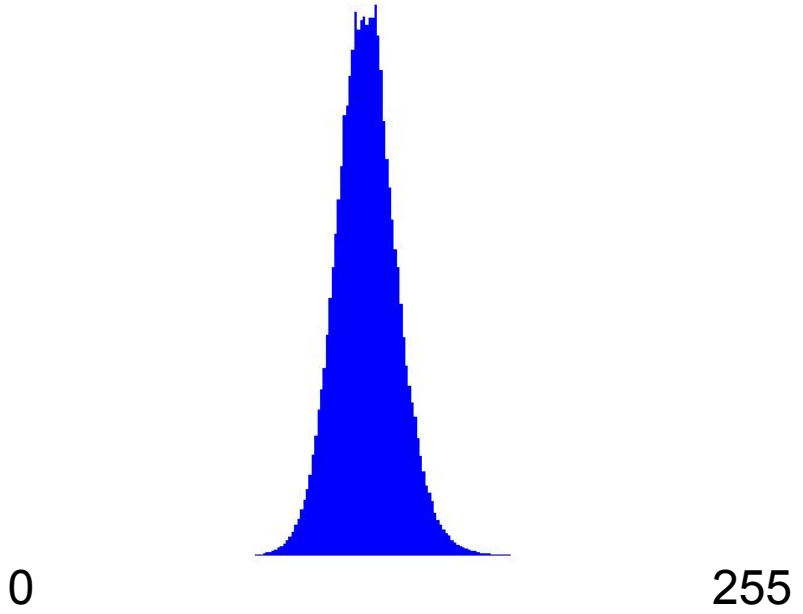


Softmax Sampling

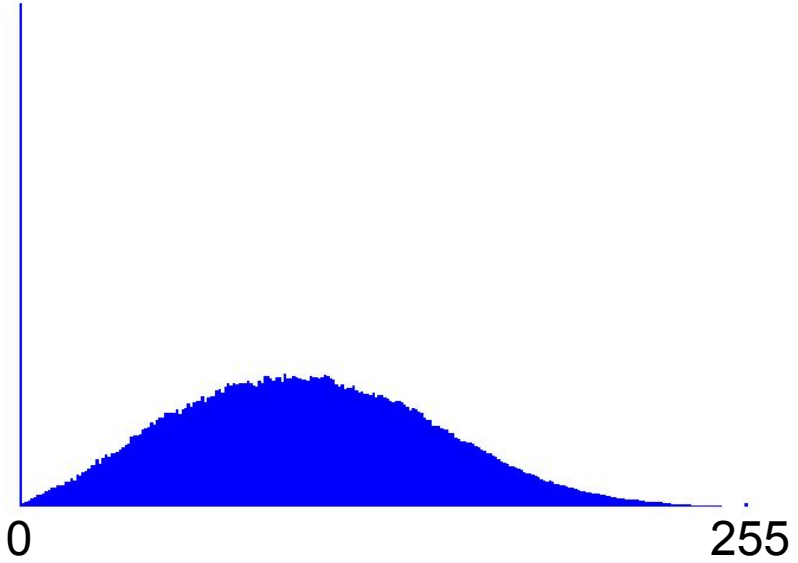


Softmax Sampling

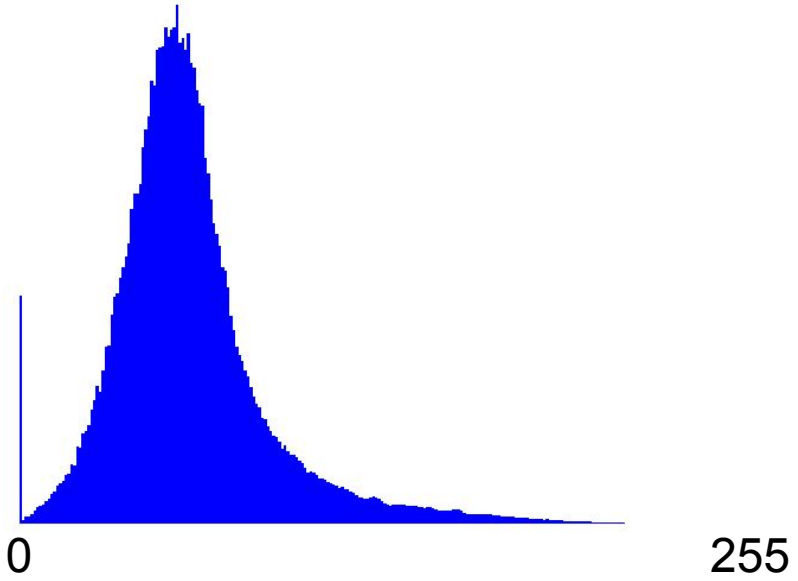
Softmax Sampling



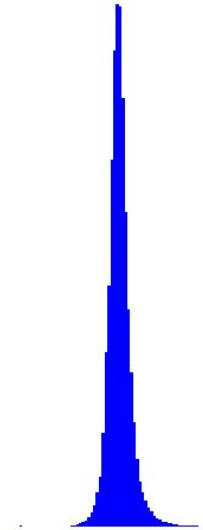
Softmax Sampling



Softmax Sampling



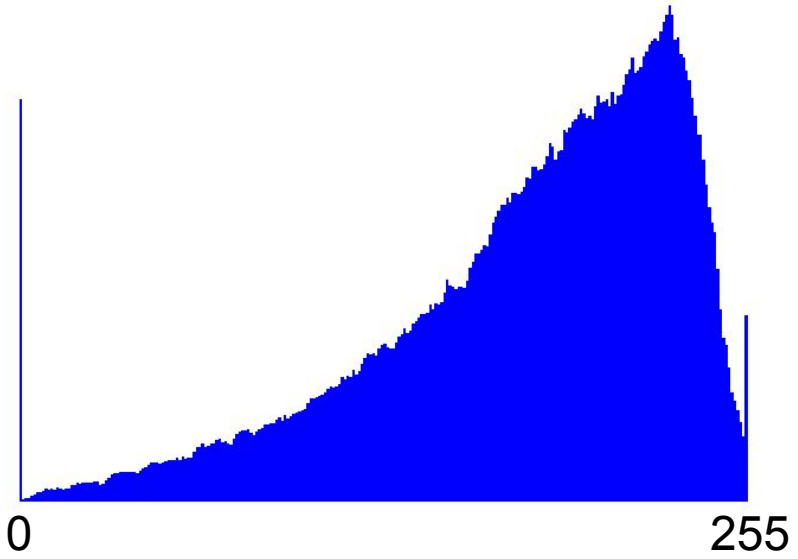
Softmax Sampling



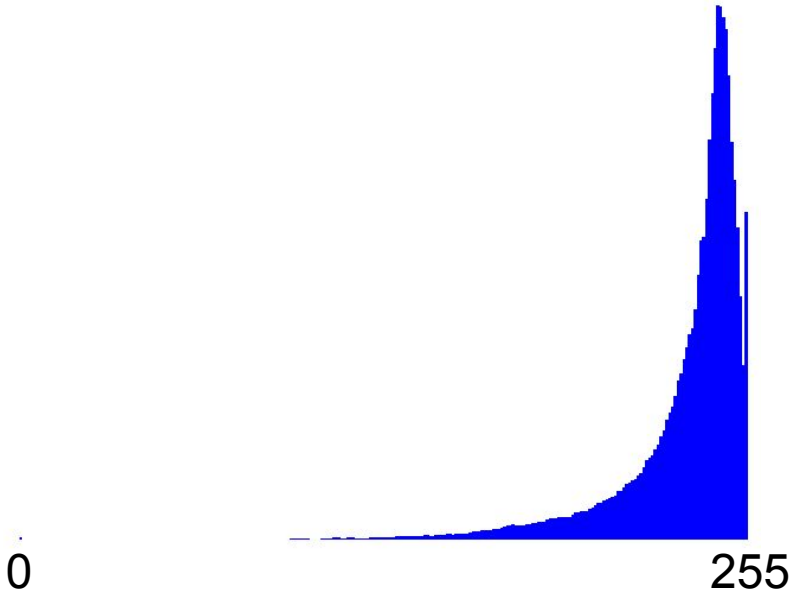
0

255

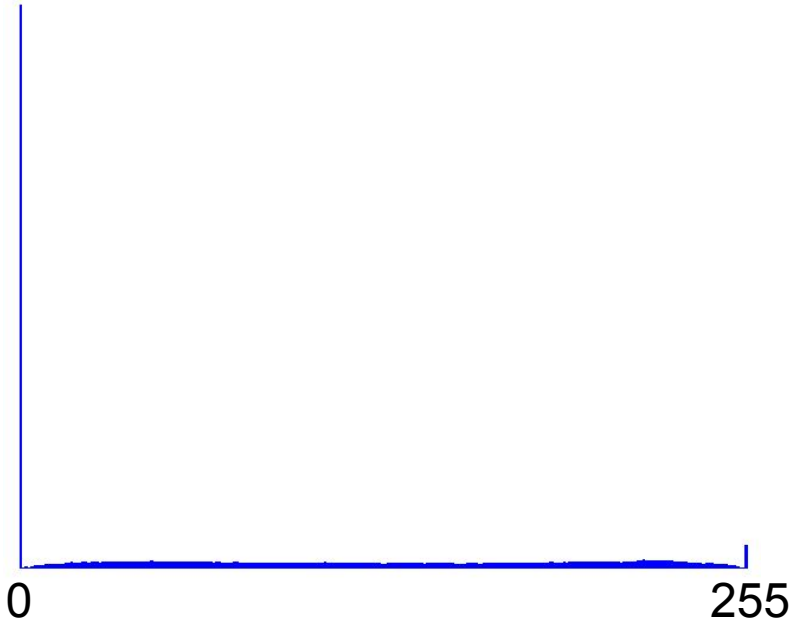
Softmax Sampling



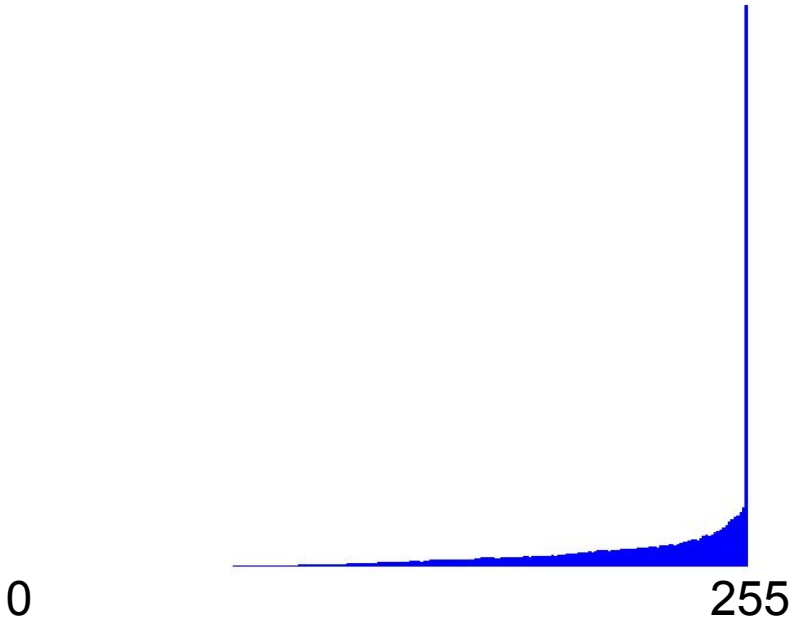
Softmax Sampling



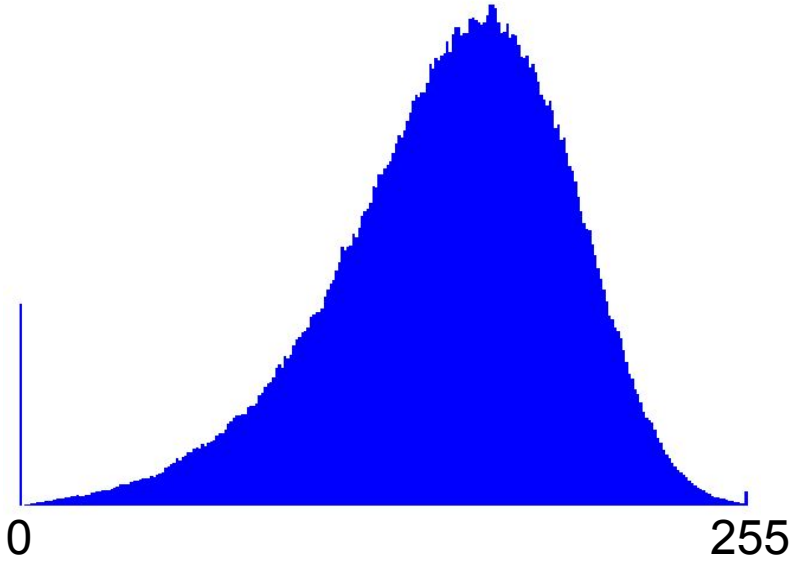
Softmax Sampling



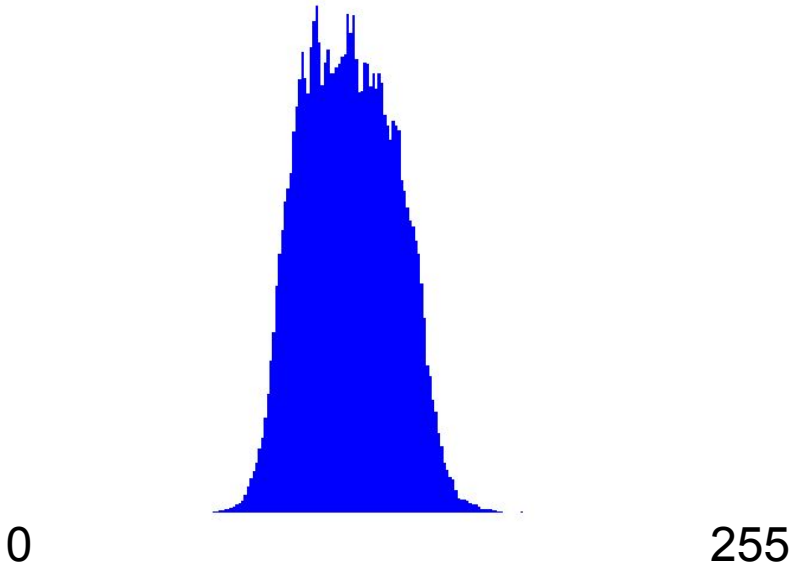
Softmax Sampling



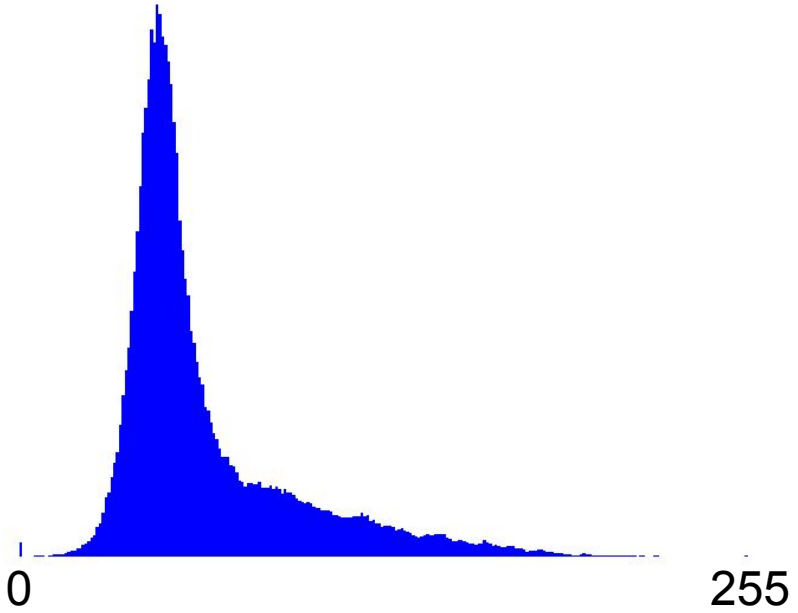
Softmax Sampling



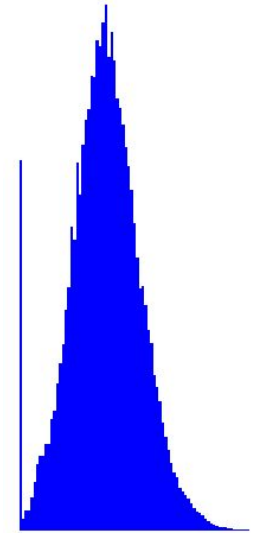
Softmax Sampling



Softmax Sampling



Softmax Sampling

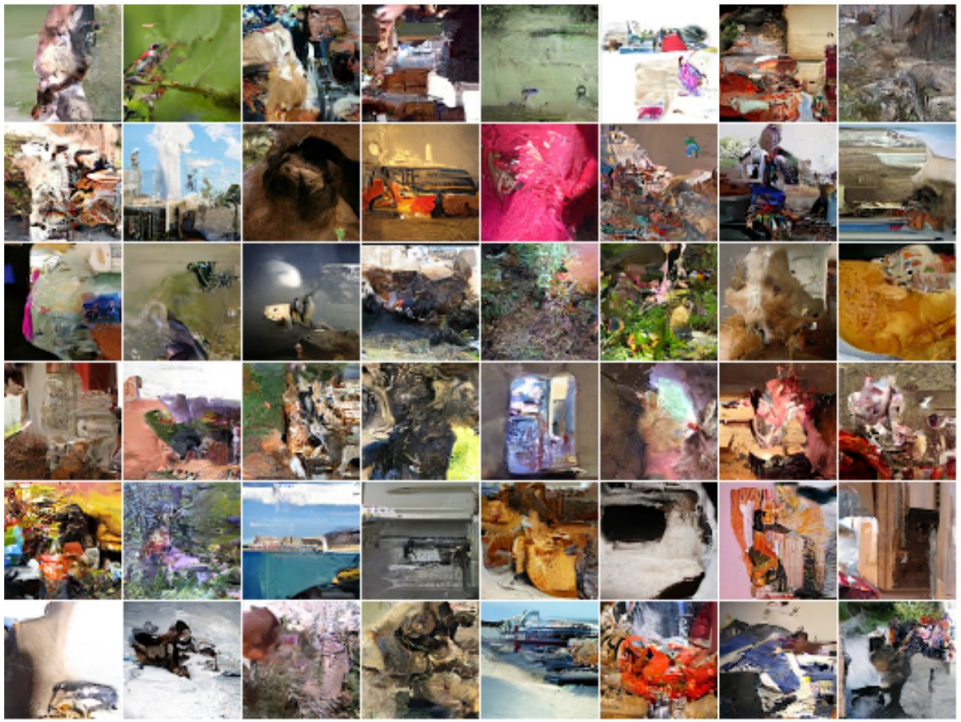


0

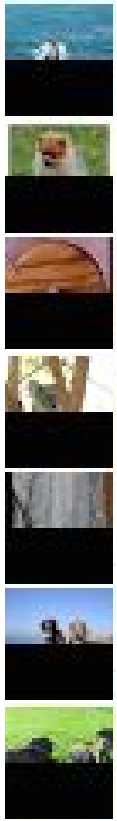
255

Pixel RNN [van den Oord, Kalchbrenner, Kavukcuoglu, ICML16]

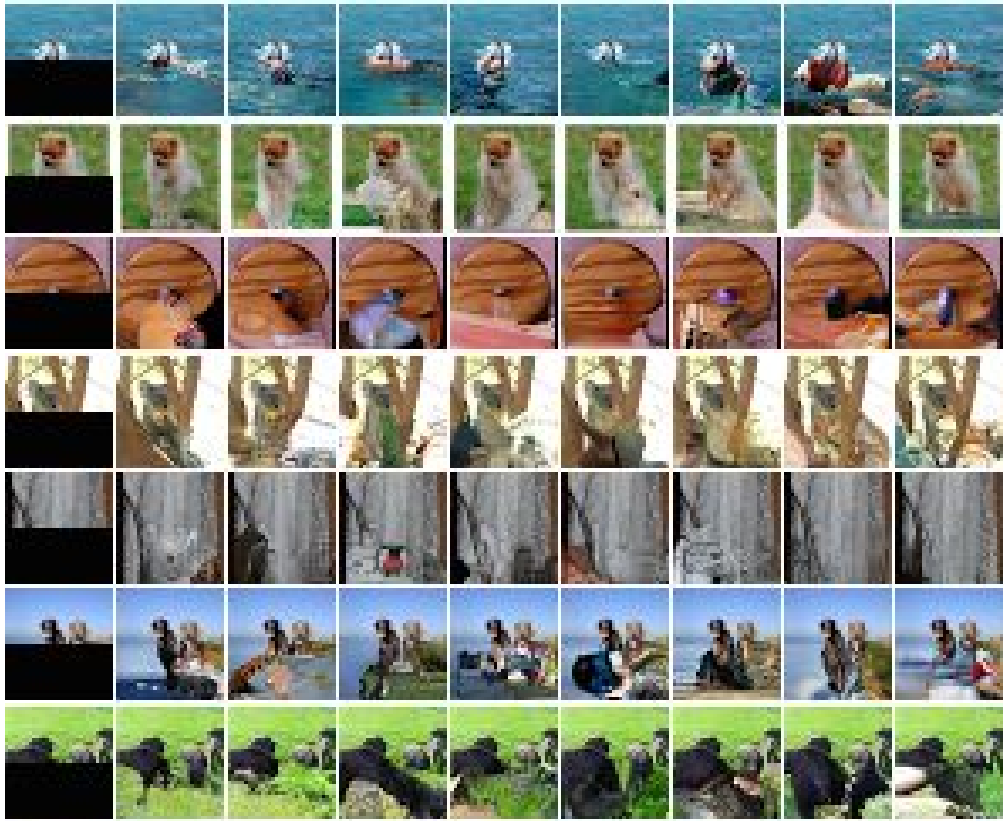
Sequence of Words == Sequence of Pixels



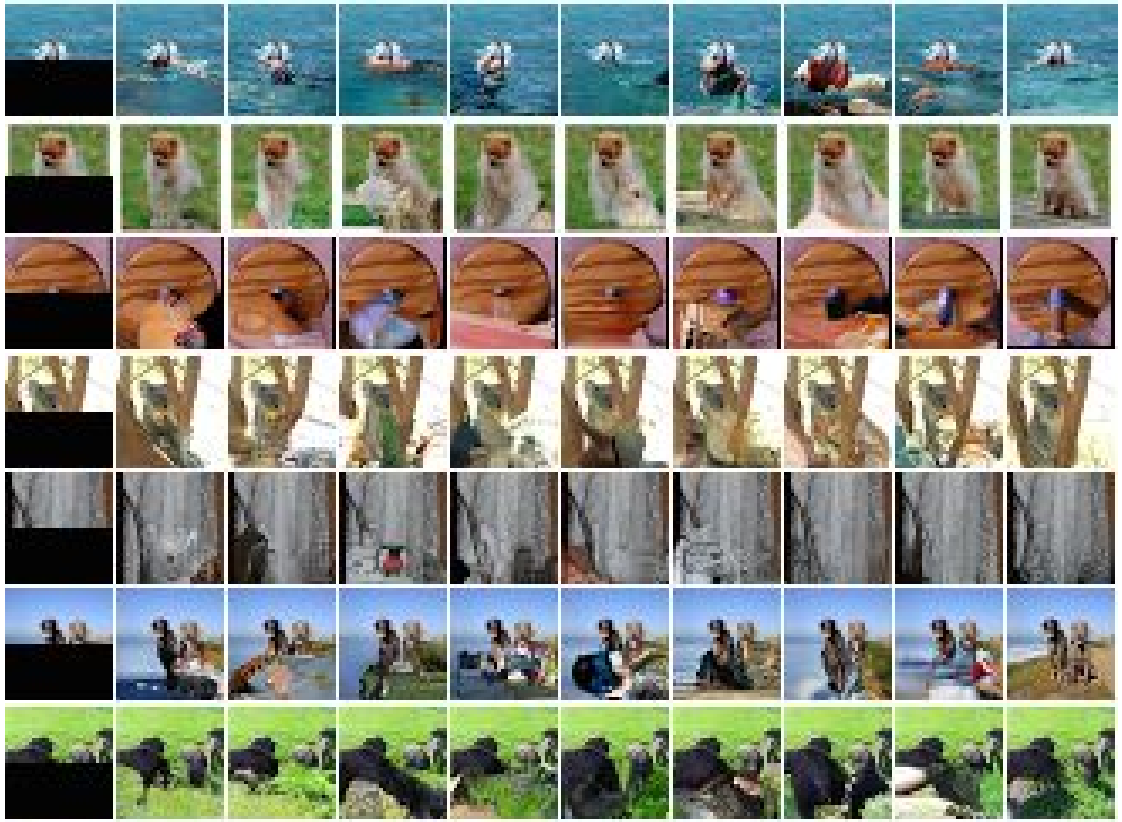
occluded



occluded completions

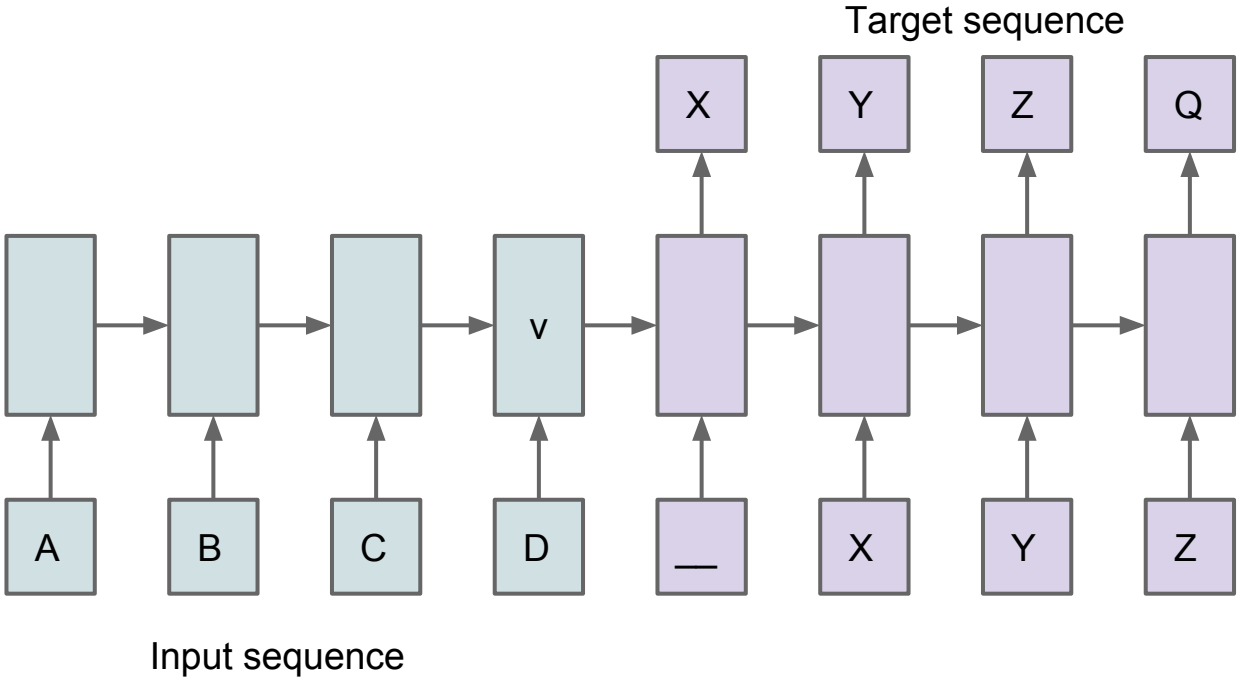


occluded completions original



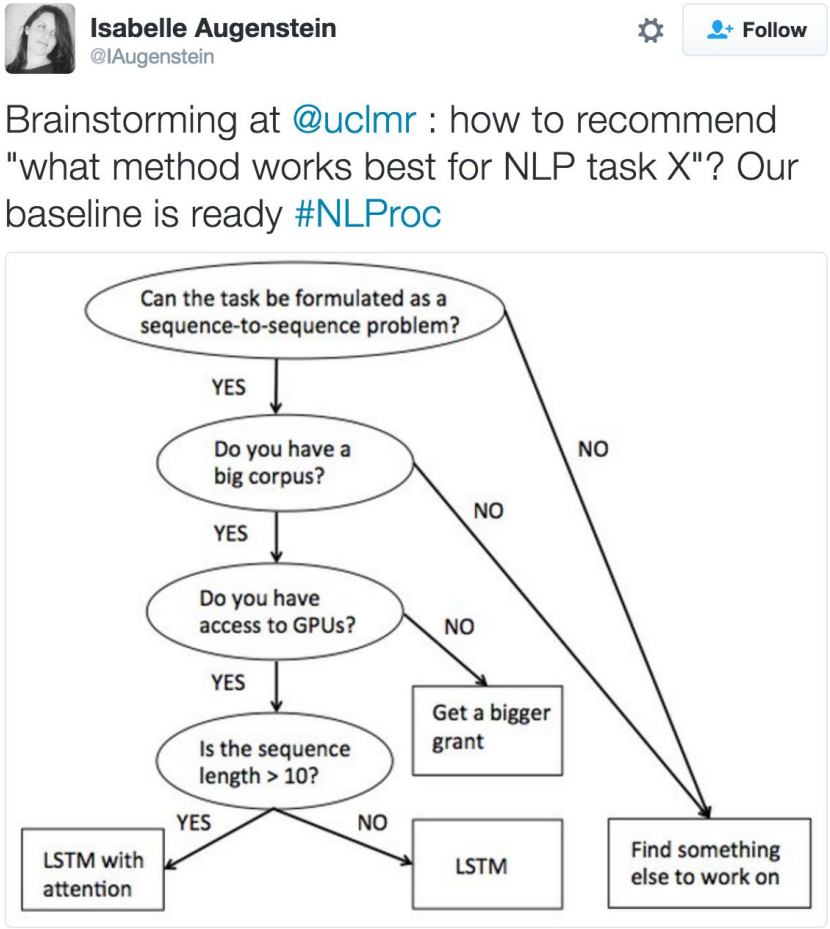
Part IV.A: Sequence-to-Sequence & Applications

Main idea



$$P(y_1, \ldots, y_{T'} | x_1, \ldots, x_T) = \prod_{t=1}^{T'} p(y_t | v, y_1, \ldots, y_{t-1})$$

Sequence-to-Sequence

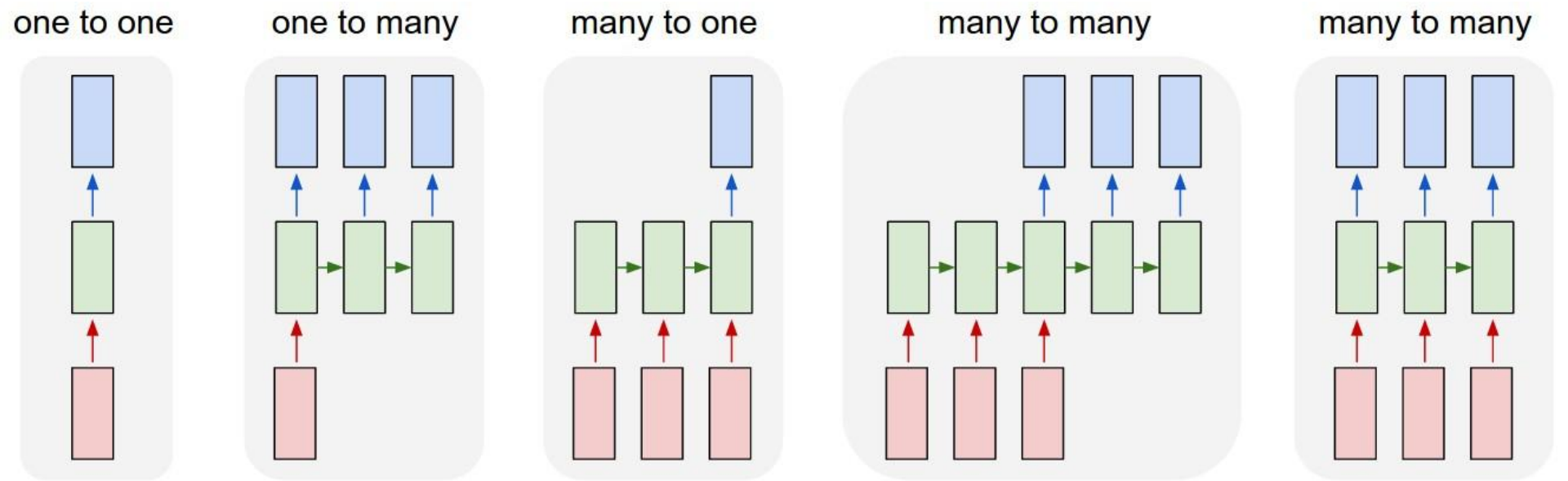


Sequence-to-Sequence

1. MT [Kalchbrenner et al, EMNLP 2013][Cho et al, EMLP 2014]**[Sutskever & Vinyals & Le, NIPS 2014]**[Luong et al, ACL 2015][Bahdanau et al, ICLR 2015]
2. Image captions [Mao et al, ICLR 2015][Vinyals et al, CVPR 2015][Donahue et al, CVPR 2015][Xu et al, ICML 2015]
3. Speech [Chorowsky et al, NIPS DL 2014][Chan et al, arxiv 2015]
4. Parsing [Vinyals & Kaiser et al, NIPS 2015]
5. Dialogue [Shang et al, ACL 2015][Sordoni et al, NAACL 2015][Vinyals & Le, ICML DL 2015]
6. Video Generation [Srivastava et al, ICML 2015]
7. Geometry [Vinyals & Fortunato & Jaitly, NIPS 2015]

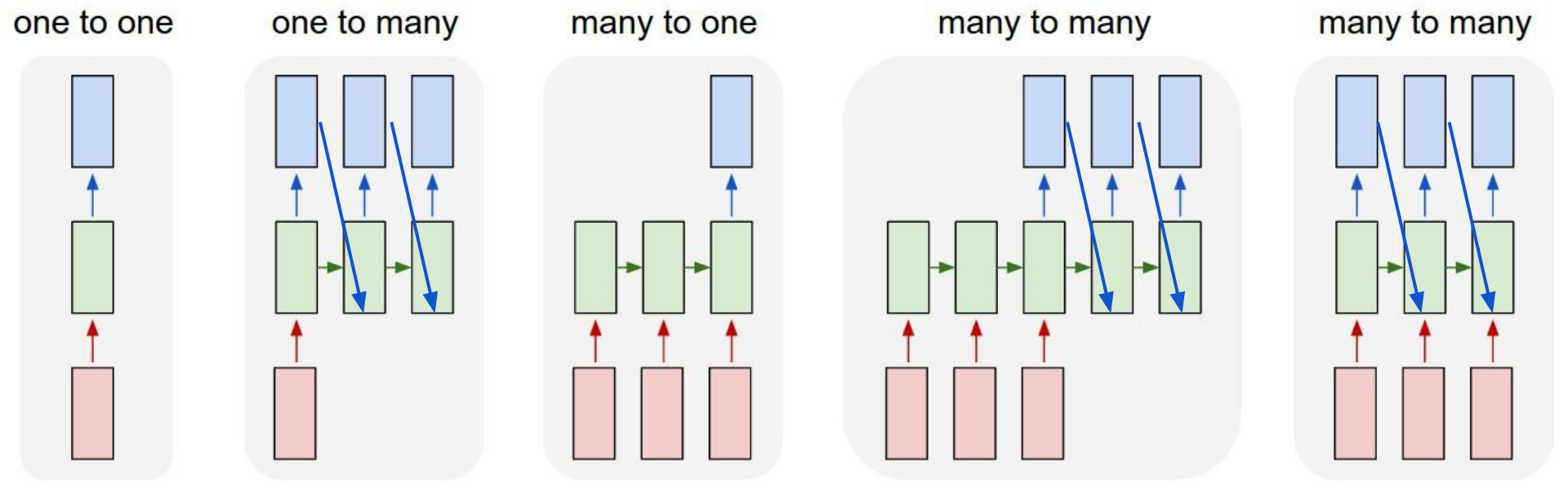
Sequence Models

http://karpathy.github.io/2015/05/21/rnn-effectiveness/



Sequence Models (Autoregressive)

http://karpathy.github.io/2015/05/21/rnn-effectiveness/



Training vs Inference

Training

$$\theta^* = \arg\max_{\theta} P(y_1, \ldots, y_{T'} | x_1, \ldots, x_T)$$

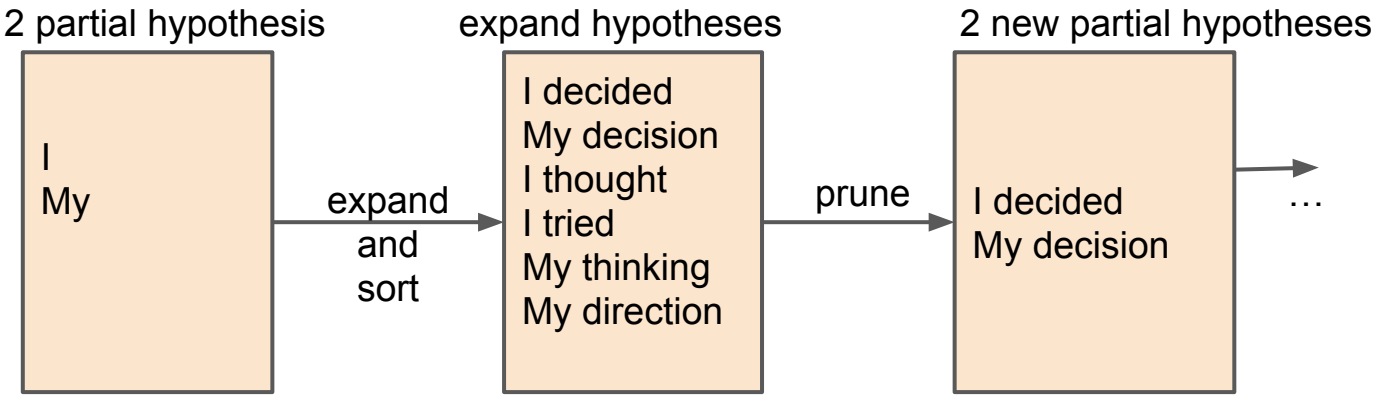
Inference

$$Y^* = \arg\max_{Y} P(Y | x_1, \ldots, x_T)$$

- Used beam search decoding over Y
- Small beam (~20) works fine

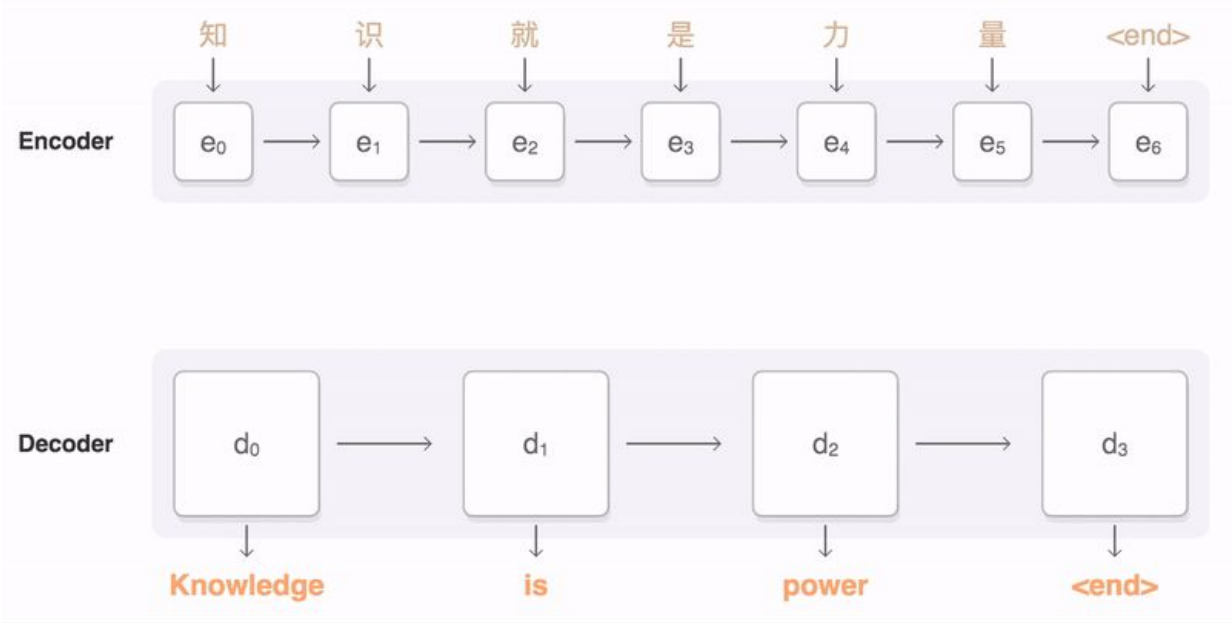
Decoding in a Nutshell (Beam Size 2)

$$y^* = \arg \max_{y_1,\ldots,y_{T'}} P(y_1, \ldots, y_{T'} | x_1, \ldots, x_T)$$

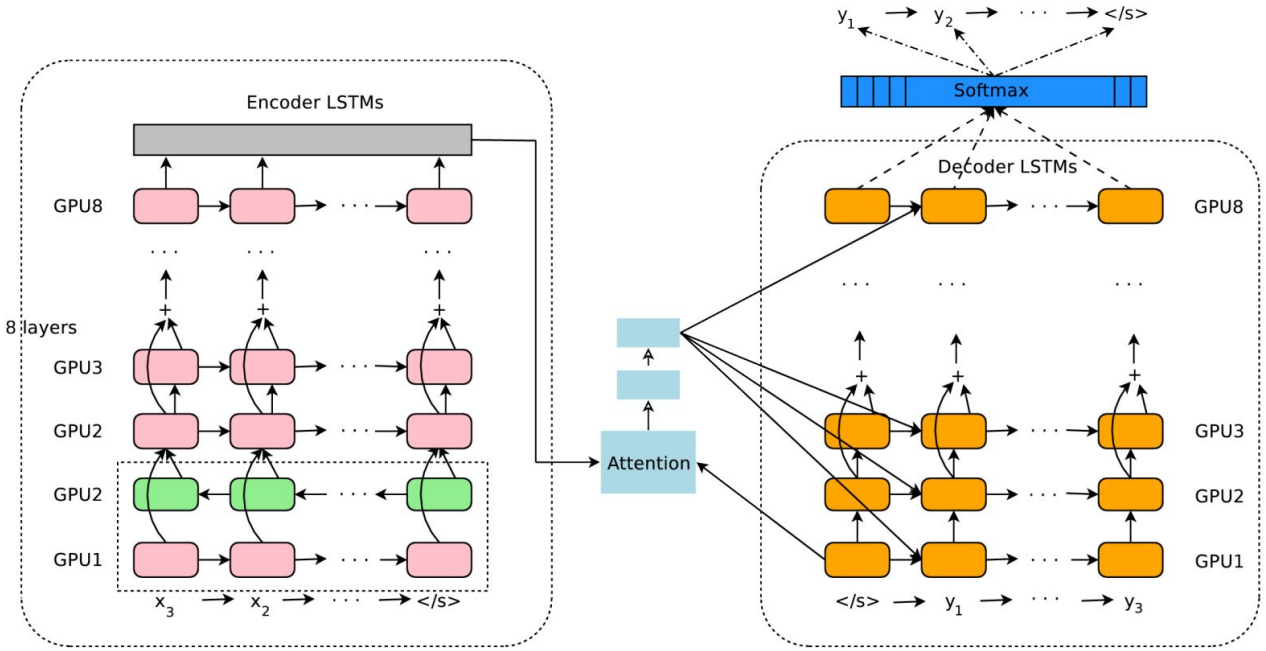


Google Neural MT

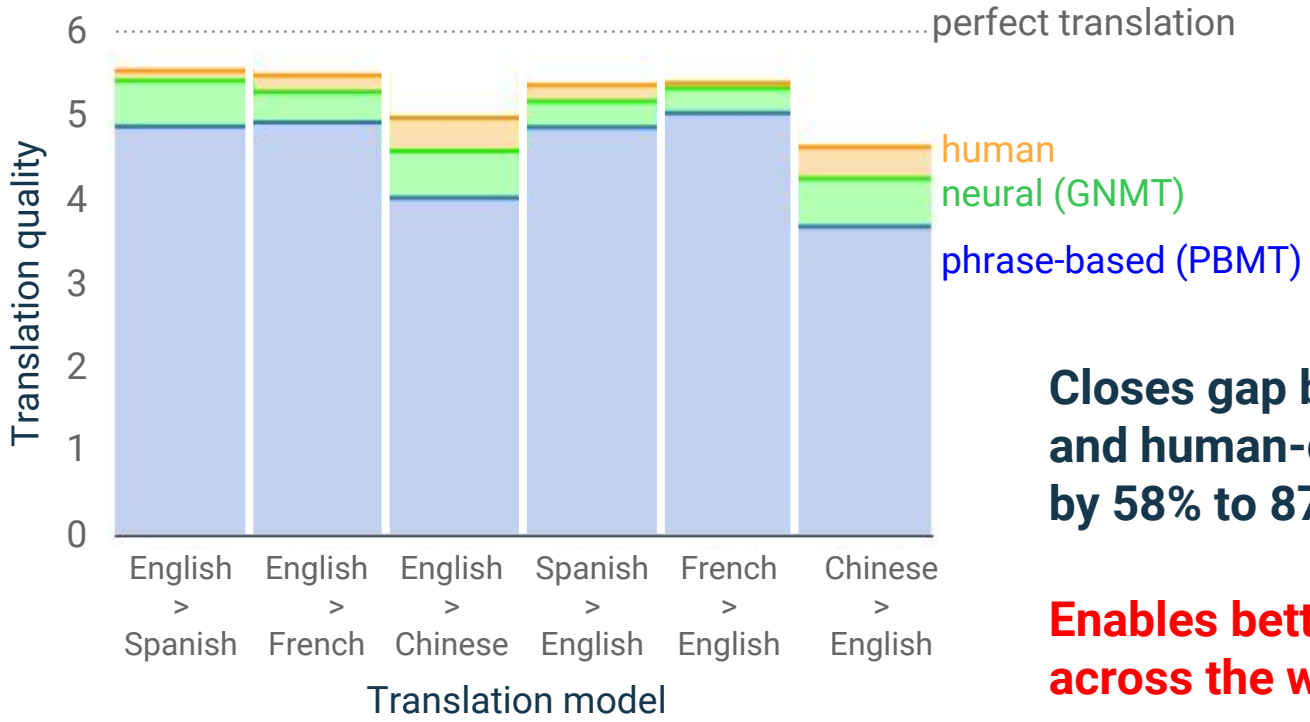
Wu et al, 2016 (Kalchbrenner et al, 2013; Sutskever et al, 2014; Cho et al, 2014; Bhadanau et al, 2014; ...)



Google Neural MT



Google Neural MT



Closes gap between old system and human-quality translation by 58% to 87%

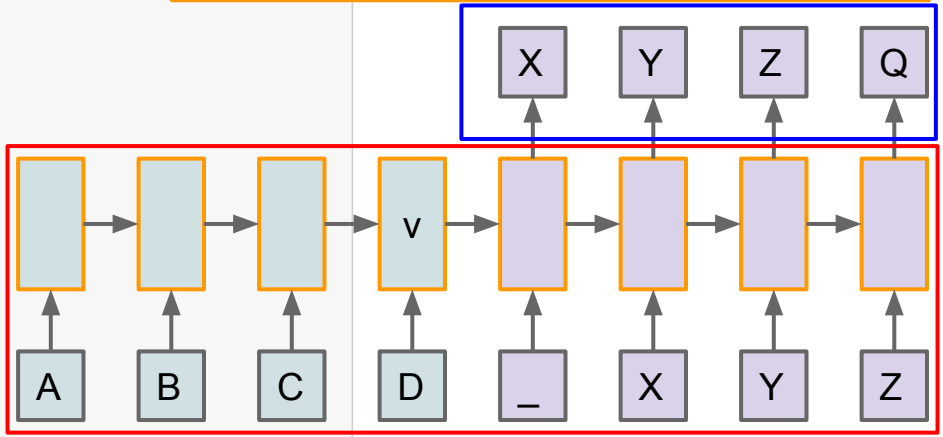
Enables better communication across the world

research.googleblog.com/2016/09/a-neural-network-for-machine.html

Code

```
class LSTMCell(object):
  def __init__(self, state_size):
    self.state_size = state_size
    self.W_f = tf.Variable(self.initializer())
    self.W_i = tf.Variable(self.initializer())
    self.W_o = tf.Variable(self.initializer())
    self.W_C = tf.Variable(self.initializer())
    self.b_f = tf.Variable(tf.zeros([state_size]))
    self.b_i = tf.Variable(tf.zeros([state_size]))
    self.b_o = tf.Variable(tf.zeros([state_size]))
    self.b_C = tf.Variable(tf.zeros([state_size]))
  def __call__(self, x_t, h_t1, C_t1):
    X = tf.concat(1, [h_t1, x_t])
    f_t = tf.sigmoid(tf.matmul(X, self.W_f) + self.b_f)
    i_t = tf.sigmoid(tf.matmul(X, self.W_i) + self.b_i)
    o_t = tf.sigmoid(tf.matmul(X, self.W_o) + self.b_o)
    Ctilde_t = tf.tanh(tf.matmul(X, self.W_C) + self.b_C)
    C_t = f_t * C_t1 + i_t * Ctilde_t
    h_t = o_t * tf.tanh(C_t)
    return h_t, C_t
  def initializer(self):
    return tf.random_uniform([2*self.state_size, self.state_size],
                             -0.1, 0.1)
```

```
# words and targets are placeholders for [batch_size, num_steps]
# tensor of word and target ids
words = tf.placeholder(tf.int64, name='words')
targets = tf.placeholder(tf.int64, name='targets')

def model(batch_size, num_steps):
  output = [tf.zeros([batch_size, state_size])] * 4
  state = [tf.zeros([batch_size, state_size])] * 4
  preds = []
  cost = 0.0
  for i in range(num_steps):
    # Get the embedding for words
    embedding = tf.nn.embedding_lookup(embedding_params, words[:, i])
    # Run the LSTM cells
    output[0], state[0] = lstm[0](embedding, output[0], state[0])
    for d in range(1, 4):
      output[d], state[d] = lstm[d](output[d-1], output[d], state[d])
    # Get the logits
    logits = tf.matmul(output[-1], sm_w) + sm_b
    # Get the softmax predictions
    preds.append(tf.nn.softmax(logits))
    # Cost per step
    cost = cost + tf.reduce_mean(
      tf.nn.sparse_softmax_cross_entropy_with_logits(logits,
                                                     targets[:, i]))
  # Average cost across time steps
  cost = cost / np.float32(num_steps)
  return preds, cost
```



Human: A young girl asleep on the sofa cuddling a stuffed bear.

NIC: A close up of a child holding a stuffed animal.

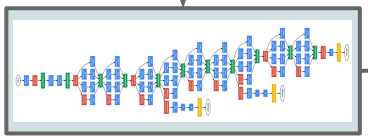
NIC: A baby is asleep next to a teddy bear.

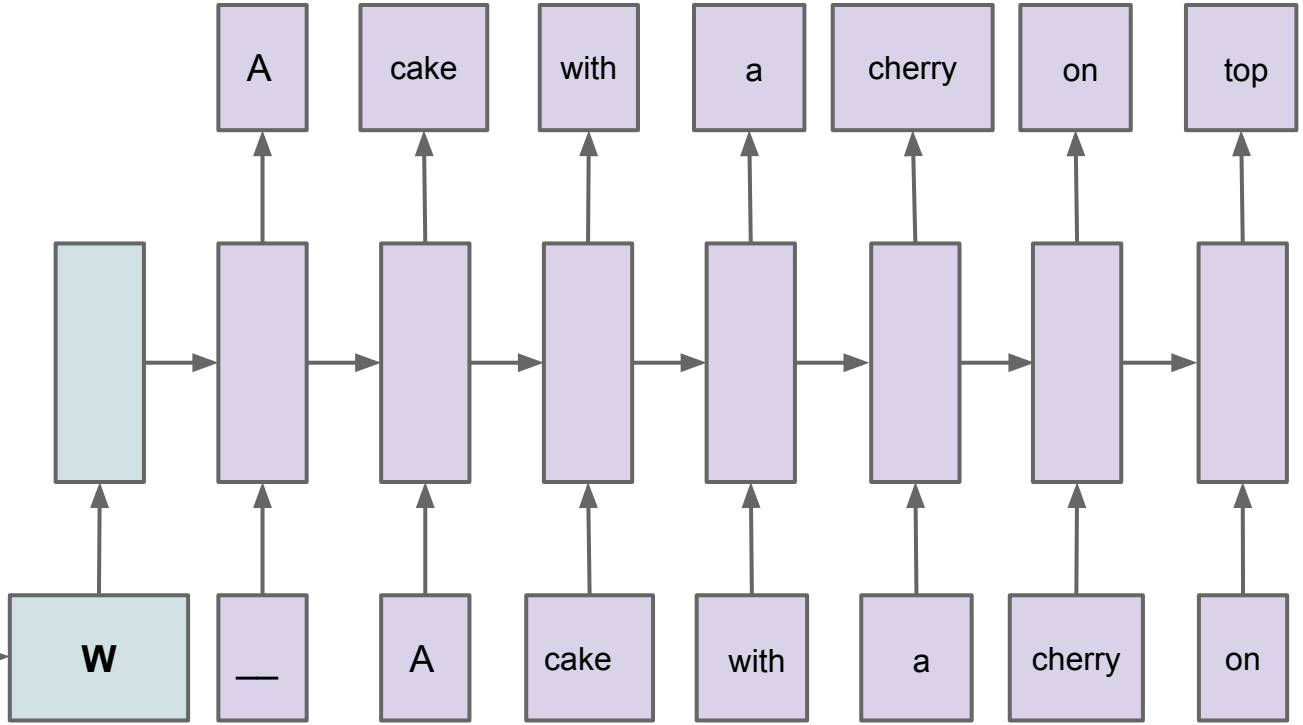
How to do Image Captions?

P(English | French)

P(English | Image)

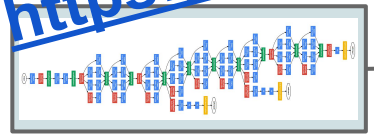
Captioning

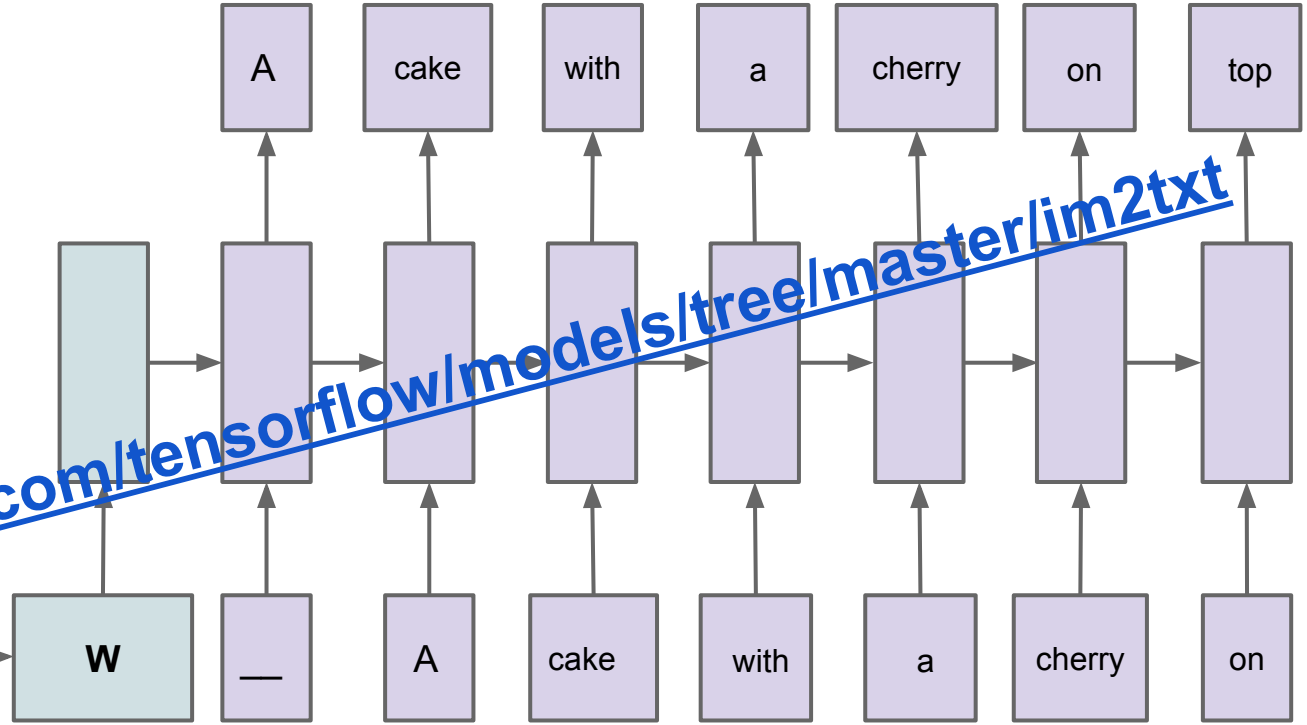




$$\theta^\star = \arg\max_\theta p(S|I)$$

Captioning





https://github.com/tensorflow/models/tree/master/im2txt

$$\theta^\star = \arg\max_\theta p(S|I)$$

Human: A close up of two bananas with bottles in the background.

BestModel: A bunch of bananas and a bottle of wine.

InitialModel: A close up of a plate of food on a table.

Human: A view of inside of a car where a cat is laying down.

BestModel: A cat sitting on top of a black car.

InitialModel: A dog sitting in the passenger seat of a car.

Human: A brown dog laying in a red wicker bed.

BestModel: A small dog is sitting on a chair.

InitialModel: A large brown dog laying on top of a couch.

Human: A man outside cooking with a sub in his hand.

BestModel: A man is holding a sandwich in his hand.

InitialModel: A man cutting a cake with a knife.

Human: Someone is using a small grill to melt his sandwich.

BestModel: A person is cooking some food on a grill.

InitialModel: A pizza sitting on top of a white plate.

Human: A woman holding up a yellow banana to her face.

BestModel: A woman holding a banana up to her face.

InitialModel: A close up of a person eating a hot dog.

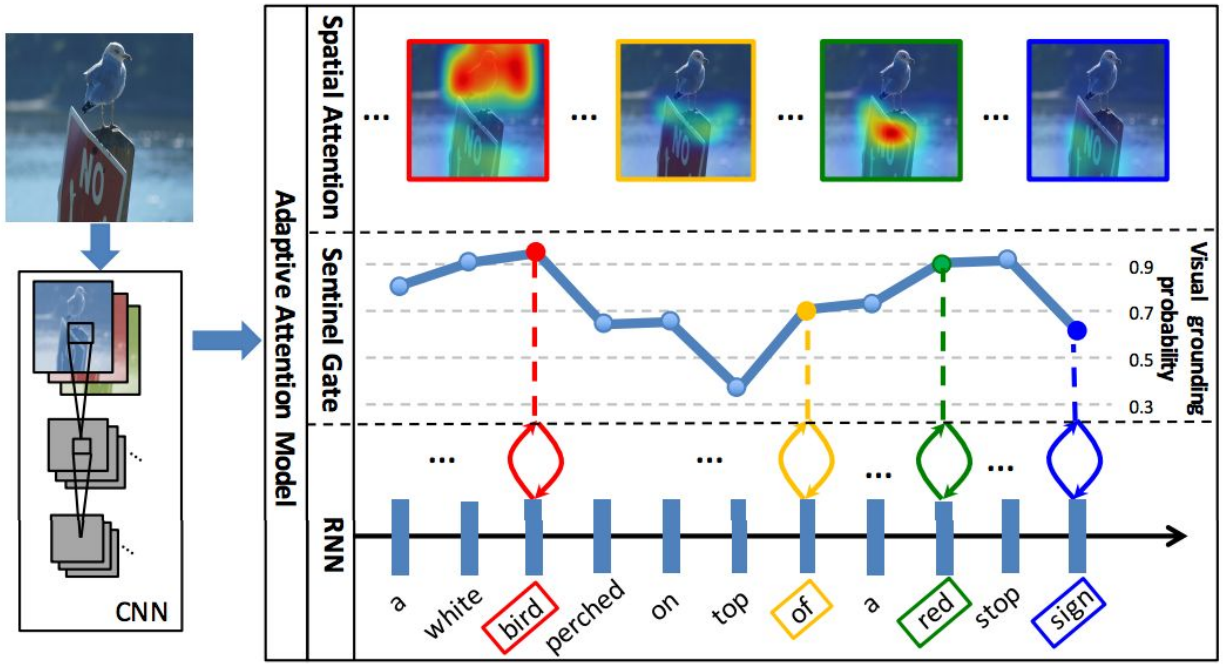
Human: A blue , yellow and red train travels across the tracks near a depot.

BestModel: A blue and yellow train traveling down train tracks.

InitialModel: A train that is sitting on the tracks.

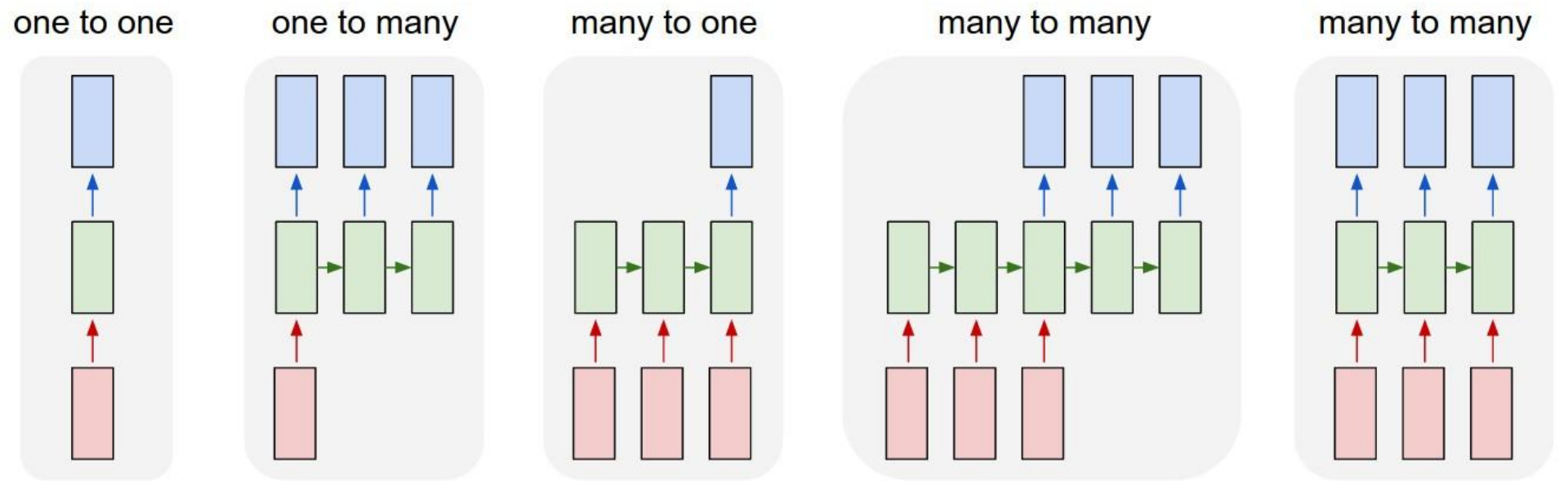
Advanced Networks: Captioning

Figure from [Lu et al, 2016]



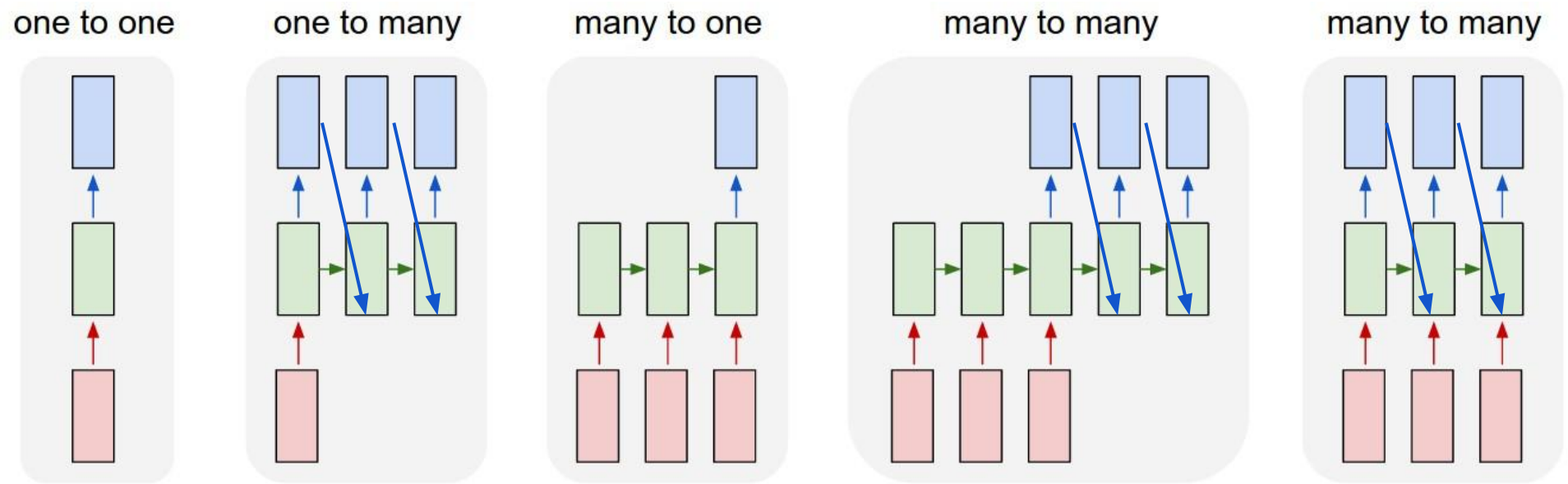
Sequence Models

http://karpathy.github.io/2015/05/21/rnn-effectiveness/



Sequence Models (Autoregressive)

http://karpathy.github.io/2015/05/21/rnn-effectiveness/



Conditional Pixel CNN

[van den Oord, Kalchbrenner, Vinyals, Espeholt, Graves, Kavukcuoglu, NIPS16]

Geyser

Hartebeest

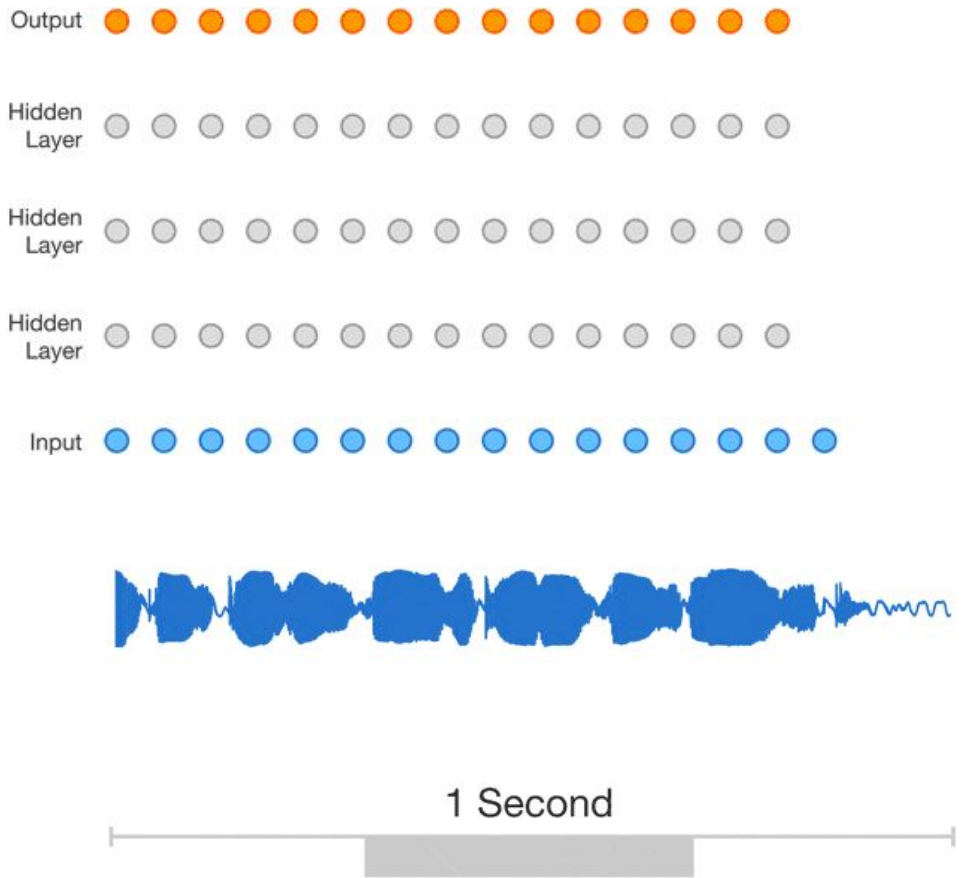
Grey whale

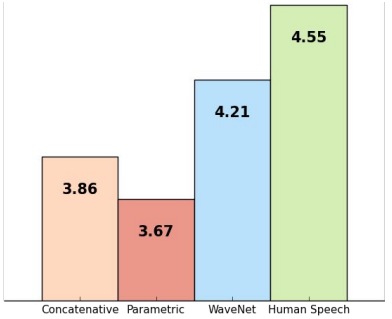
Tiger

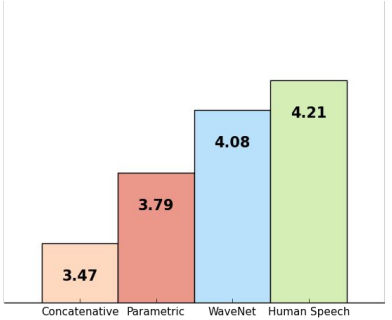
EntleBucher (dog)

Yellow lady's slipper (flower)

WaveNets [van den Oord et al, 2016]

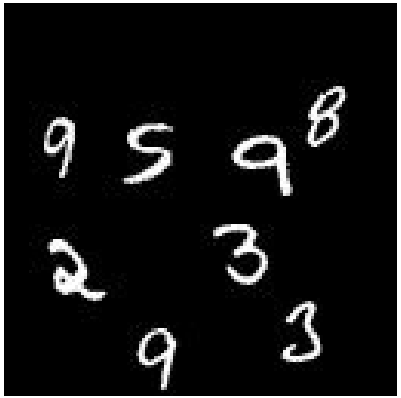


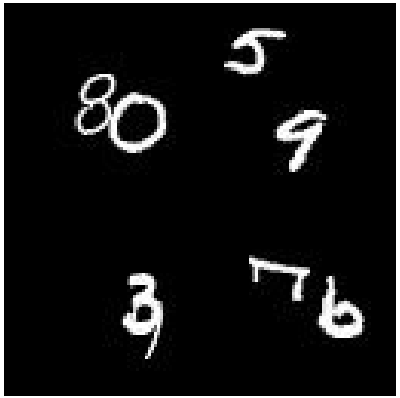




Video Pixel Network (VPN)

Model	Test
(Shi et al., 2015)	367.2
(Srivastava et al., 2015a)	341.2
(Brabandere et al., 2016)	285.2
(Patraucean et al., 2015)	179.8
Baseline model	110.1
VPN	**87.6**
Lower Bound	86.3





Kalchbrenner, N., et al. “Video Pixel Networks.” ***ICML (2017)***.

Neural Conversational Models

- Take movie subtitles (~900M words) or IT HelpDesk chats
- Predict the next dialog from history

i got to go .
no .
i get too emotional when i drink .
have another beer . i 've got to get up early .
no , you don 't . sit down .
i get too emotional when i drink .
will you have another beer ?
i 've got to go !
why ?
i got to get up early in the morning .
you 're drunk .
and emotional !
you got to go .

Conditional Language [Vinyals, Le, ICML DL WS 2015]

Human: what do you think about tesla ? Machine: he ’s a good conductor .

Human: What do you think about messi ? Machine: he ’s a great player .

Human: what do you think about cleopatra ? Machine: oh , she ’s very regal .

Human: who is skywalker ? Machine: he is a hero .

Human: is sky blue or black ? Machine: blue .

Human: does a cat have a tail ? Machine: yes .

Human: does a cat have a wing ? Machine: no

Conditiona

Human: what do

Human: What do

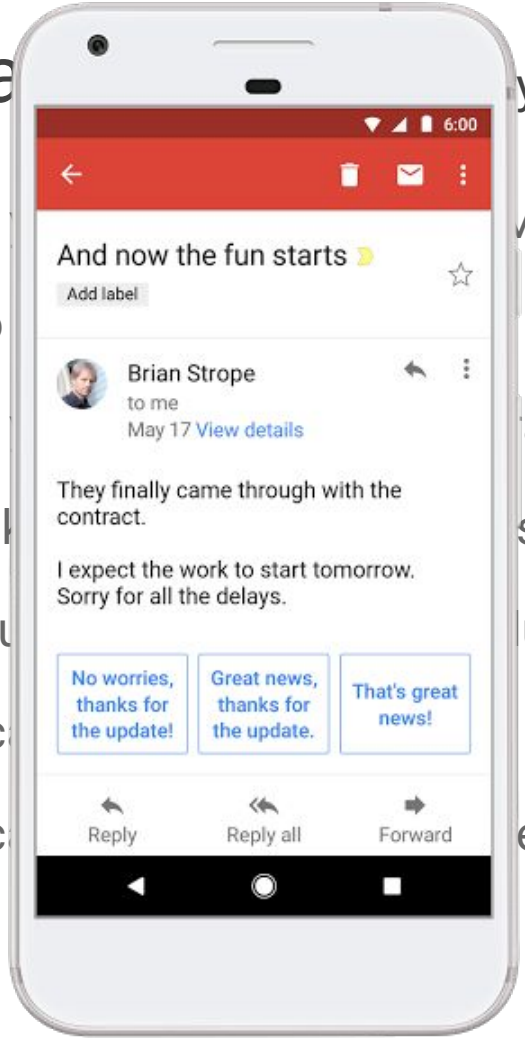
Human: what do

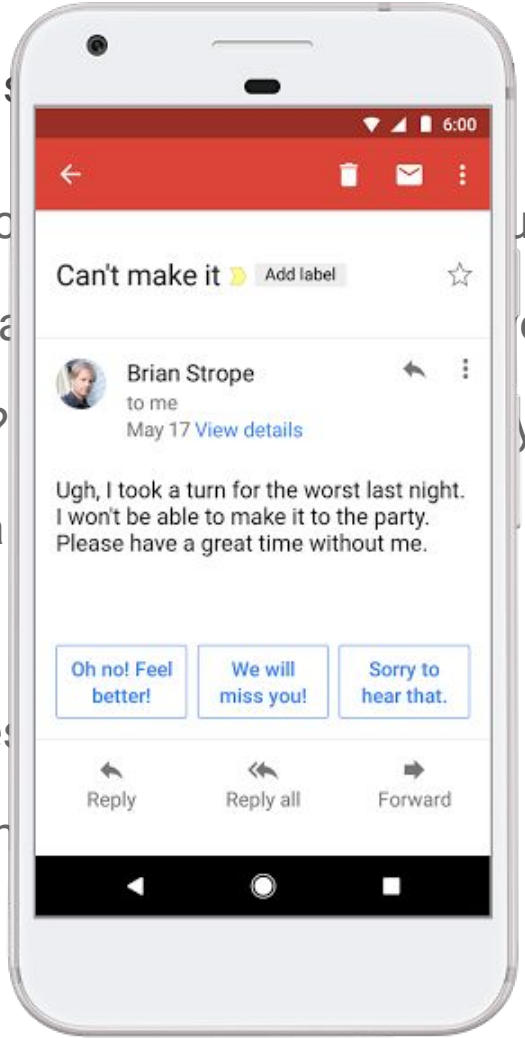
Human: who is sk

Human: is sky blu

Human: does a c

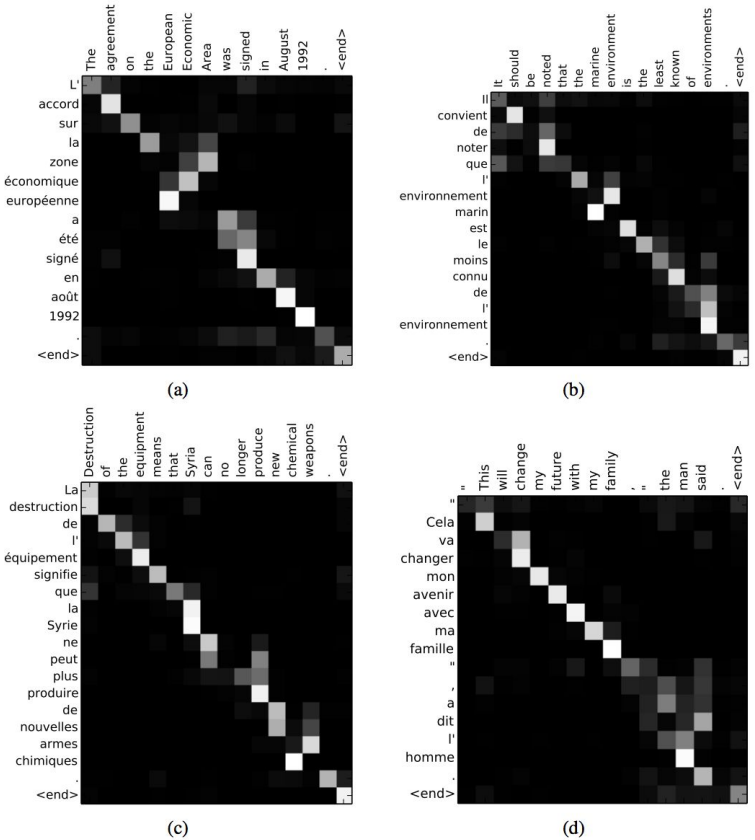
Human: does a c





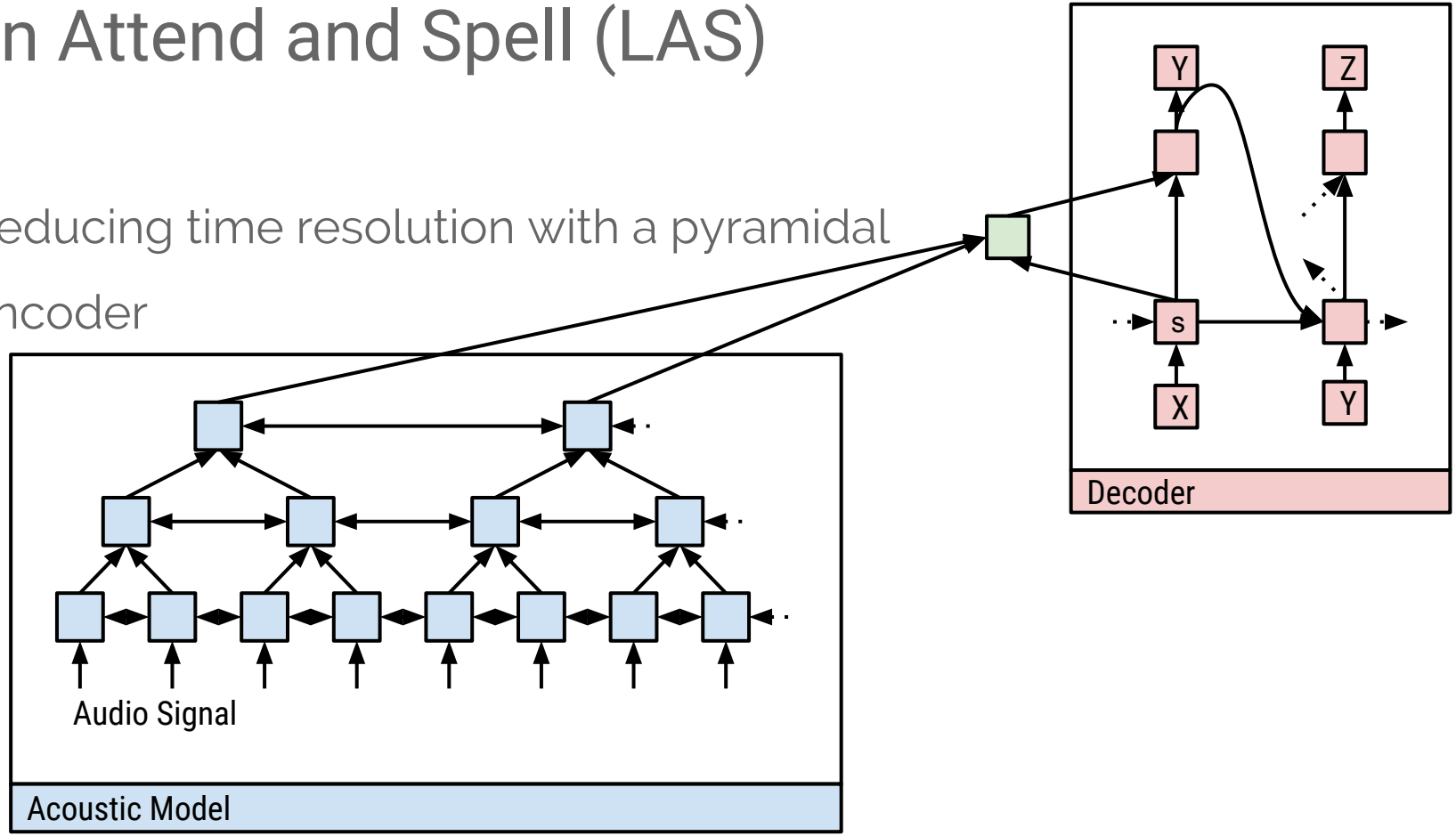
Part V: Beyond RNNs

Attention (Bahdanau et al)



Listen Attend and Spell (LAS)

- Reducing time resolution with a pyramidal encoder



Chan, W., Jaitly, N., Le, Q., Vinyals, O. "*Listen Attend and Spell*." *ICASSP (2015).*

Lip Reading

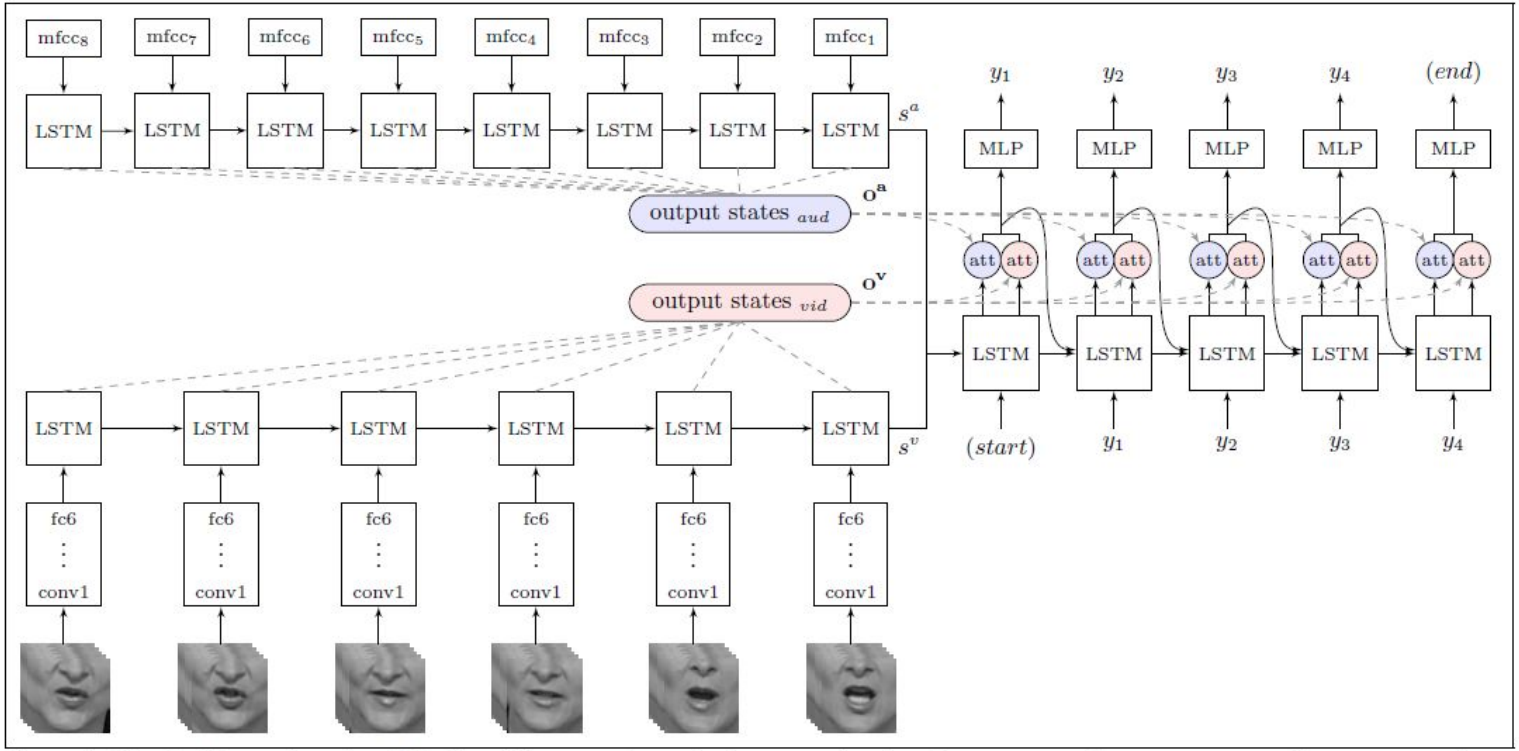
Channel	Series name	# hours	# sent.
BBC 1 HD	News†	1,584	50,493
BBC 1 HD	Breakfast	1,997	29,862
BBC 1 HD	Newsnight	590	17,004
BBC 2 HD	World News	194	3,504
BBC 2 HD	Question Time	323	11,695
BBC 4 HD	World Today	272	5,558
All		4,960	118,116

http://www.robots.ox.ac.uk/~vgg/data/lip_reading/

1. Chung, J., et al. "Lip reading sentences in the wild." *CVPR* (2017).
2. Assael, Y., et al. "Lipnet: Sentence-level lipreading." *arxiv* (2016).

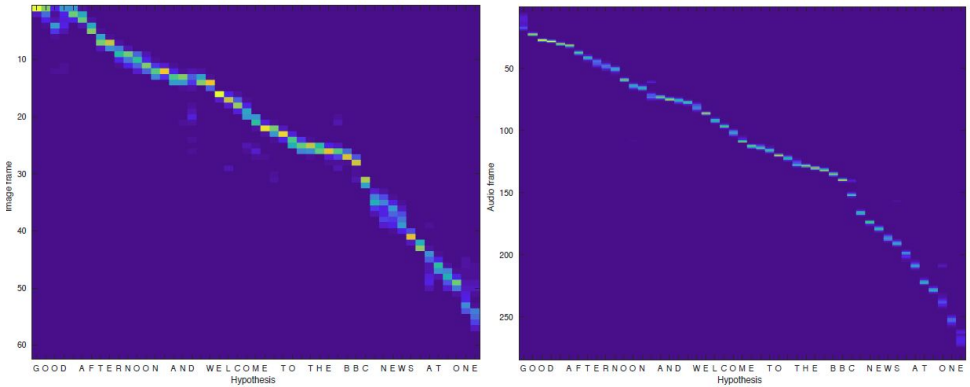
Lip Reading

Separate embedding and attention for audio and visual streams



Chung, J., et al. "Lip reading sentences in the wild." *CVPR* (2017).

Lip Reading

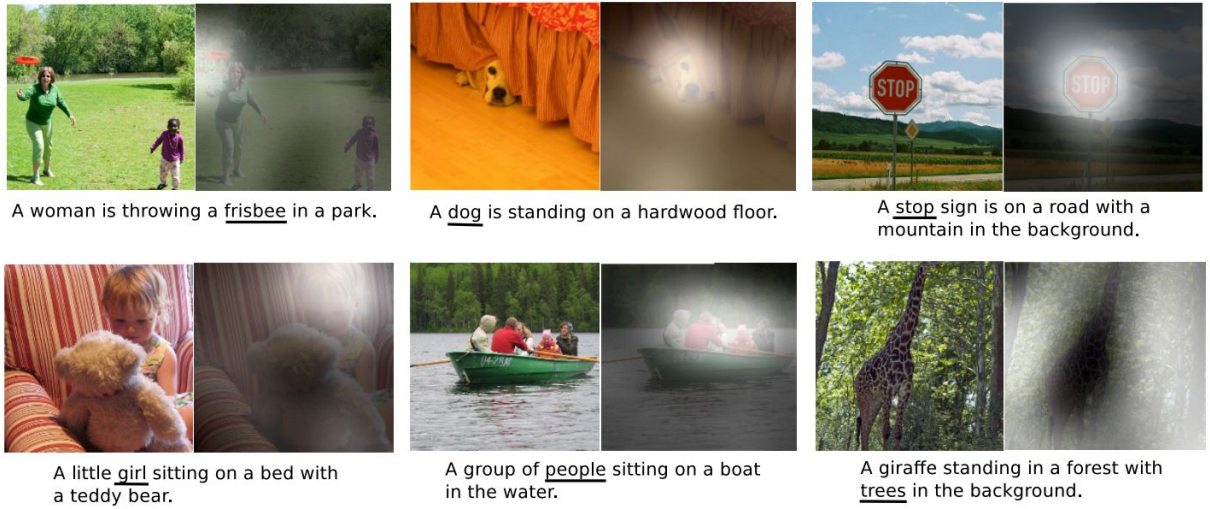


Method	SNR	CER	WER	BLEU†
Lips only				
Professional‡	-	58.7%	73.8%	23.8
WAS	-	59.9%	76.5%	35.6
WAS+CL	-	47.1%	61.1%	46.9
WAS+CL+SS	-	42.4%	58.1%	50.0
WAS+CL+SS+BS	-	39.5%	50.2%	54.9

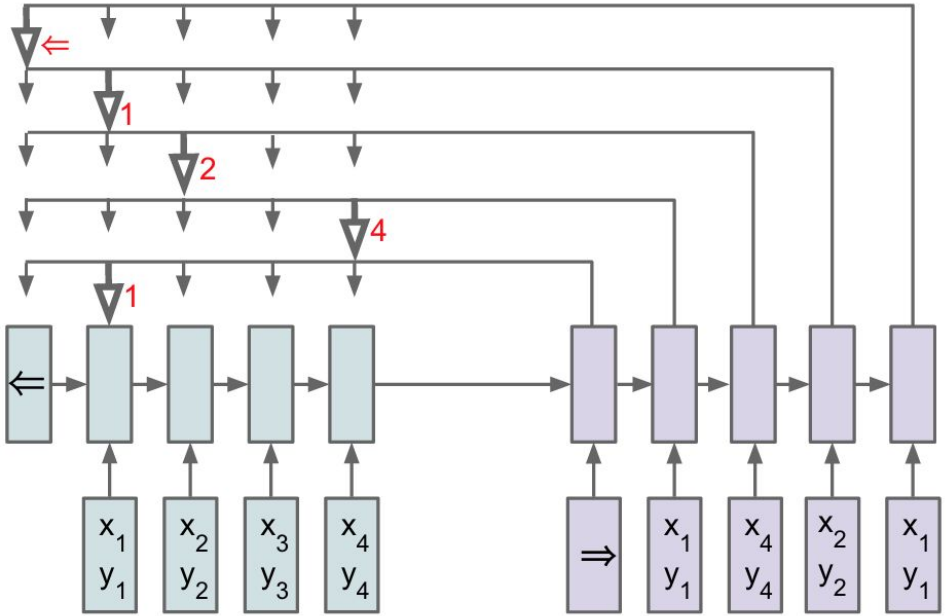
Chung, J., et al. "Lip reading sentences in the wild." *CVPR* (2017).

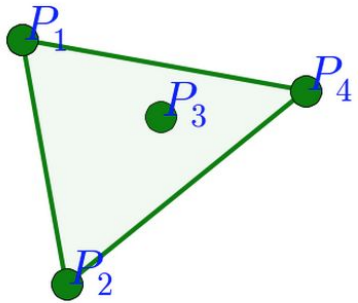
Visual Attention (Xu et al)

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

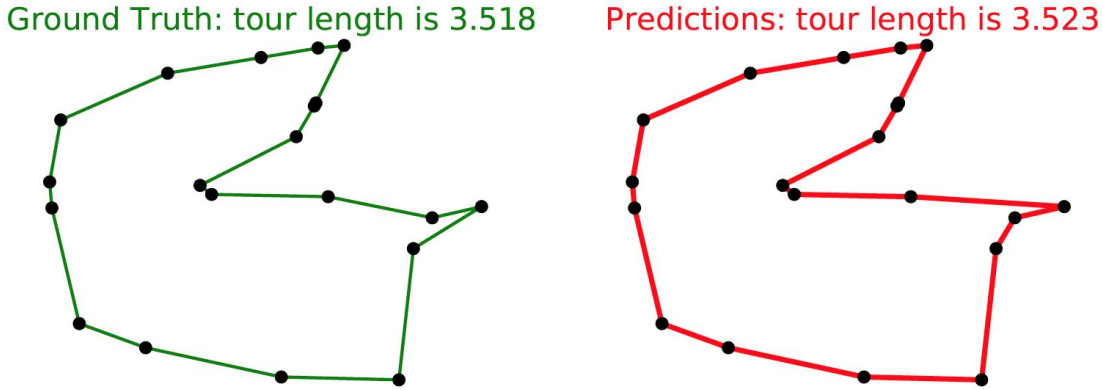


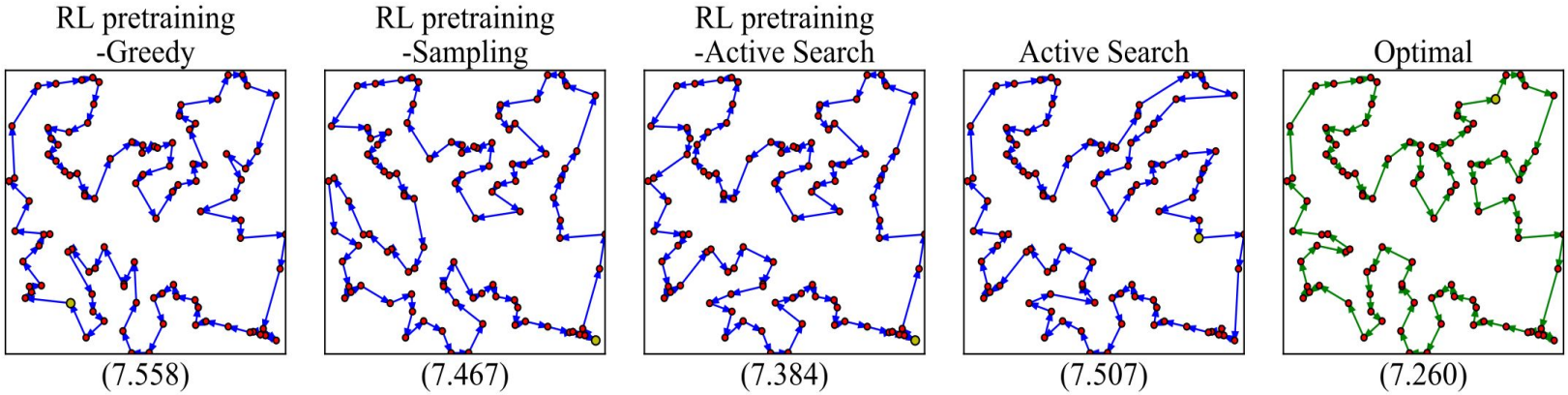
Pointer Networks





TSP: NP hard!


Ground Truth: tour length is 3.518
Predictions: tour length is 3.523


RL pretraining -Greedy
(7.558)
RL pretraining -Sampling
(7.467)
RL pretraining -Active Search
(7.384)
Active Search
(7.507)
Optimal
(7.260)

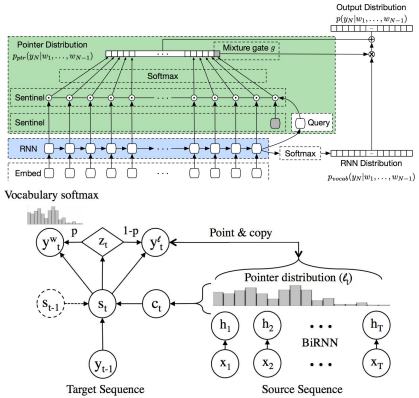
Frontiers: Natural Language **Understanding**

- Good progress on dealing with rarely seen / unknown words

  [Luong et al, 2015][Gulcehre et al, 2016][Merity et al, 2016][Wu et al, 2016]...

- Limited domain (e.g. BABI) tasks increasingly solved

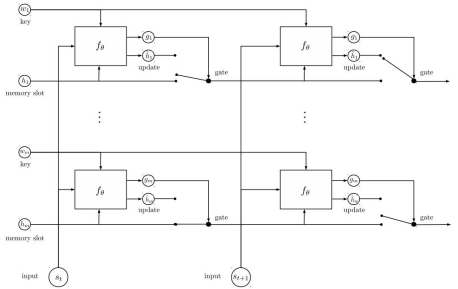
  [Kadlec et al, 2016][Dhingra et al, 2016][Trischler et al, 2016][Henaff et al, 2016]...



- We can’t generate long / coherent text

  [Liu et al, 2018]

- We can’t summarize (or write :)) a book



Frontiers: Complex Decision Making in Games

https://github.com/deepmind/pysc2

DeepMind

BLIZZARD ENTERTAINMENT

MiniGames

Agent	Metric	MoveToBeacon	CollectMineralShards	FindAndDefeatZerglings	DefeatRoaches	DefeatZerglingsAndBanelings	CollectMineralsAndGas	BuildMarines
Random policy	mean	1	17	4	1	23	12	< 1
	max	6	35	19	46	118	750	5
Random search	mean	25	32	21	51	55	2318	8
	max	29	57	33	241	159	3940	46
DeepMind human player	mean	26	133	46	41	729	6880	138
	max	28	142	49	81	757	6952	142
StarCraft GrandMaster	mean	28	177	61	215	727	7566	133
	max	28	179	61	363	848	7566	133
Atari-net	best mean	25	96	49	101	81	3356	< 1
	max	33	131	59	351	352	3505	20
FullyConv	best mean	26	103	45	100	62	3978	3
	max	45	134	56	355	251	4130	42
FullyConv LSTM	best mean	26	104	44	98	96	3351	6
	max	35	137	57	373	444	3995	62

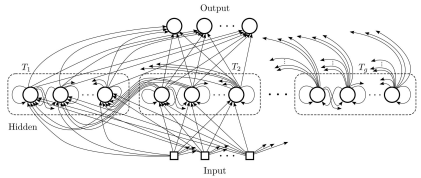
DeepMind

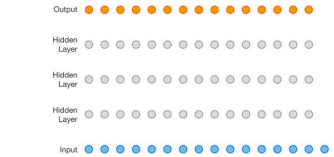
BLIZZARD ENTERTAINMENT

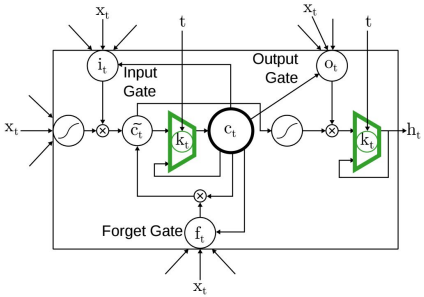
Common Challenge: Long Term Dependencies

- Thousands / Millions of steps
  - Recursive programs
  - Learning to Learn
  - Complex Decision Making
  - NLU

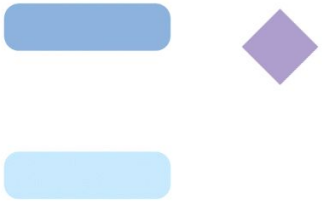
Temporal Hierarchies







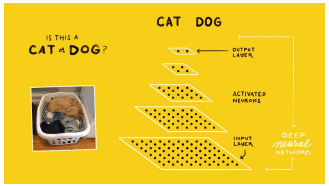
Neil et al, 2016



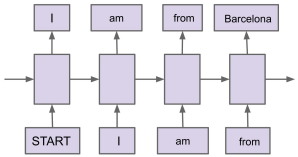
Jaderberg et al, 2016

The Deep Learning Toolbox

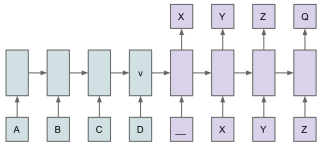
Feed forward models



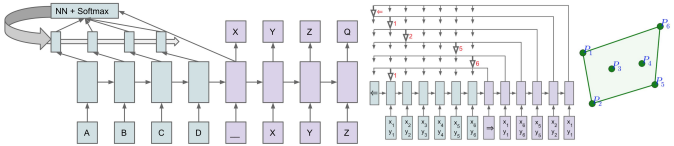
Sequence Prediction



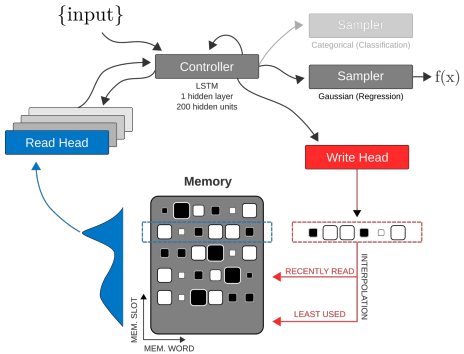
Seq2Seq



Attention/Pointers



Read/Write memories



COSINE DISTANCE

$$K(\mathbf{k}_t, \mathbf{M}_t(i)) = \frac{\mathbf{k}_t \cdot \mathbf{M}_t(i)}{\| \mathbf{k}_t \| \| \mathbf{M}_t(i) \|}$$

READ WEIGHTS

$$w_t^r(i) \leftarrow \frac{\exp\big(K(\mathbf{k}_t, \mathbf{M}_t(i))\big)}{\sum_j \exp\big(K(\mathbf{k}_t, \mathbf{M}_t(j))\big)}$$

READ VECTOR

$$\mathbf{r}_t \leftarrow \sum_i w_t^r(i) \mathbf{M}_t(i)$$

WRITE WEIGHTS

$$\mathbf{w}_t^w \leftarrow \sigma(\alpha) \mathbf{w}_{t-1}^r + (1 - \sigma(\alpha)) \mathbf{w}_{t-1}^{lu}$$

USAGE WEIGHTS

$$\mathbf{w}_t^u \leftarrow \gamma \mathbf{w}_{t-1}^u + \mathbf{w}_t^r + \mathbf{w}_t^w$$

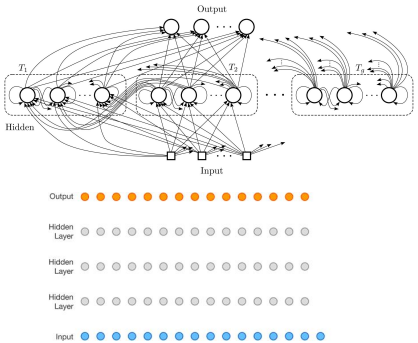
LEAST-USED WEIGHTS

$$w_t^{lu}(i) = \begin{cases} 0 & \text{if } w_t^u(i) > m(\mathbf{w}_t^u, n) \\ 1 & \text{if } w_t^u(i) \leq m(\mathbf{w}_t^u, n) \end{cases}$$

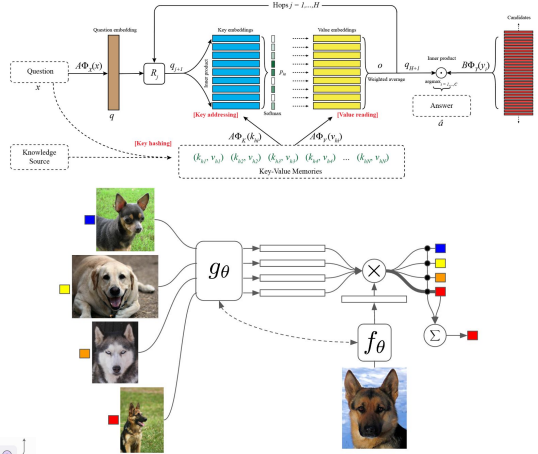
MEMORY UPDATING

$$\mathbf{M}_t(i) \leftarrow \mathbf{M}_{t-1}(i) + w_t^w(i) \mathbf{k}_t, \forall i$$

Temporal Hierarchies



Key,Value memories



Recurrent Architectures

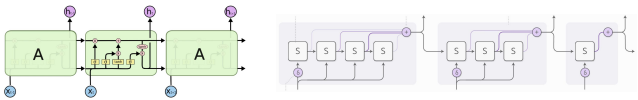


Figure credits: Jeff Dean, Chris Olah, Santoro et al 2016, Koutnik et al 2014, van den Oord et al 2016, Miller et al 2016, Vinyals et al 2016

Summary

- Sequences are important, everything is a sequence
- Recurrent Nets are a way forward, but they pose some challenges
  - Vanishing/Exploding gradients
  - Expensive to compute
- Conditional models are very powerful, and many applications “just worked” when seq2seq and similar were trained on large amounts of data
- Attention/memory, longer term memory… many challenges remain!

Questions?

@OriolVinyalsML // vinyals@google.com

Extra Resources:

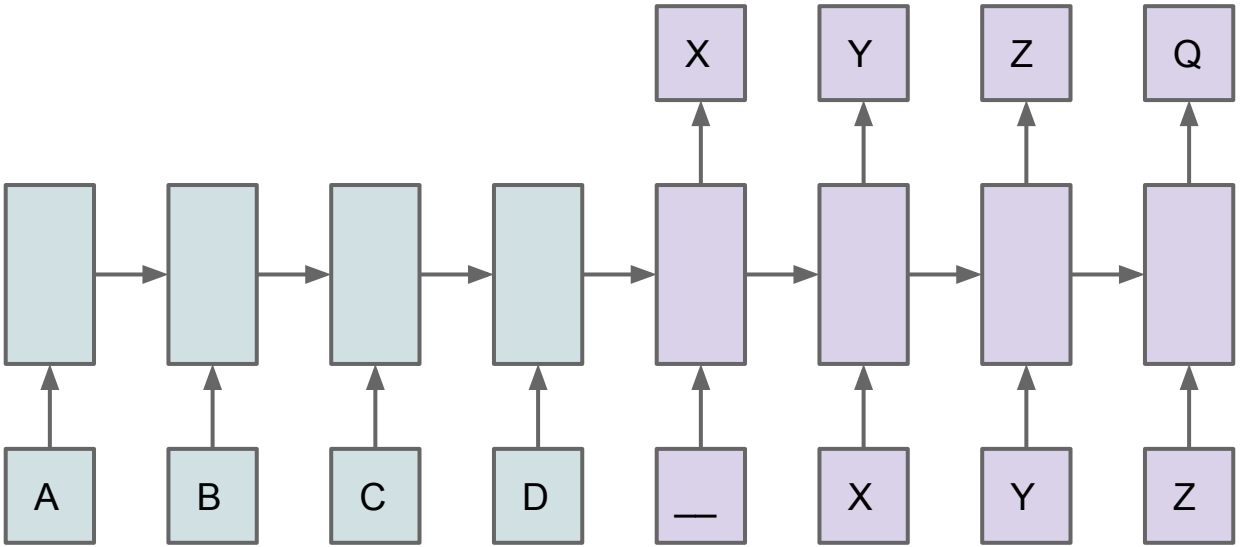
http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Vinyals & Jaitly ICML 2017 Tutorial (Seq2Seq)

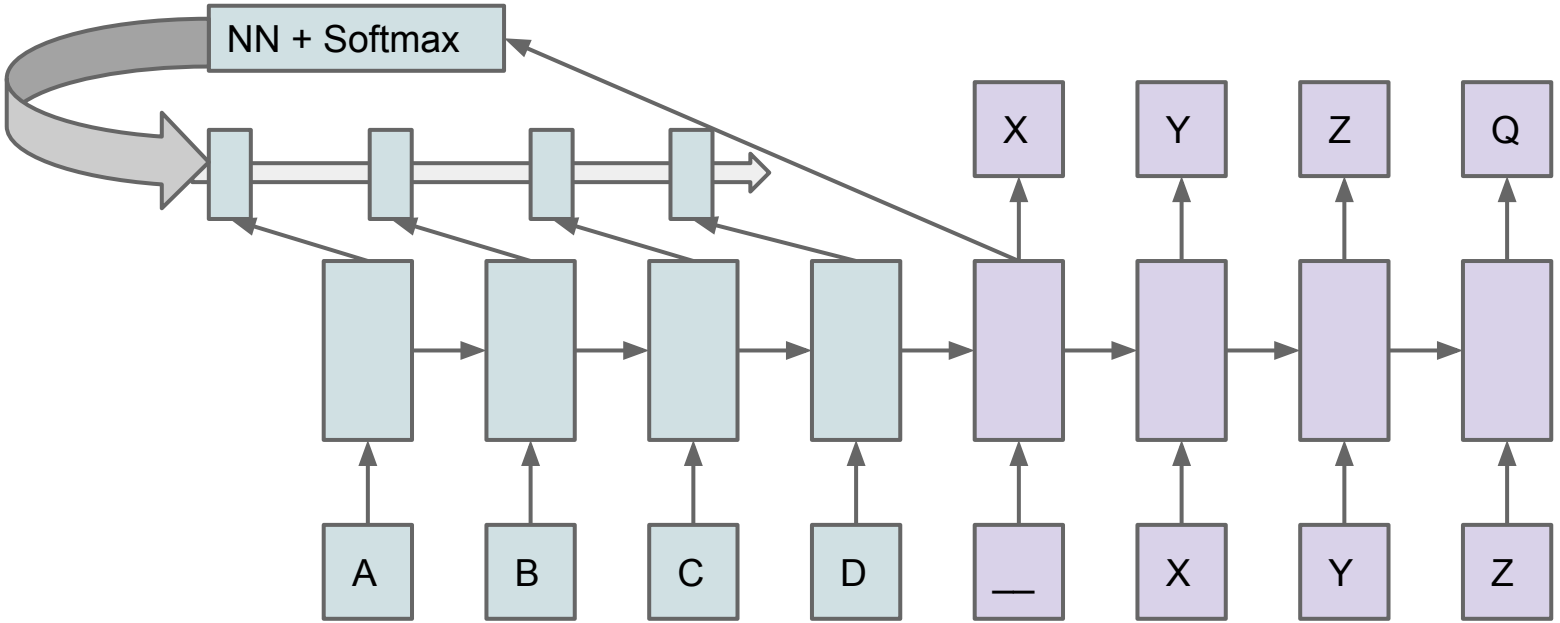
Next lecture...

Beyond image recognition, end-to-end learning, embeddings - Raia Hadsell

Attention



Attention



Position Based Attention

Inputs: “I am a cat.”

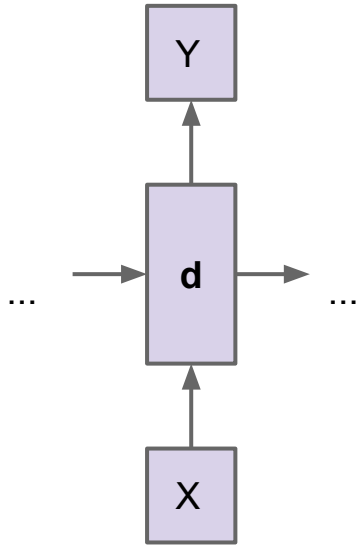
Input RNN states: $\mathbf{e}_1$ $\mathbf{e}_2$ $\mathbf{e}_3$ $\mathbf{e}_4$

Decoder RNN state: **d**

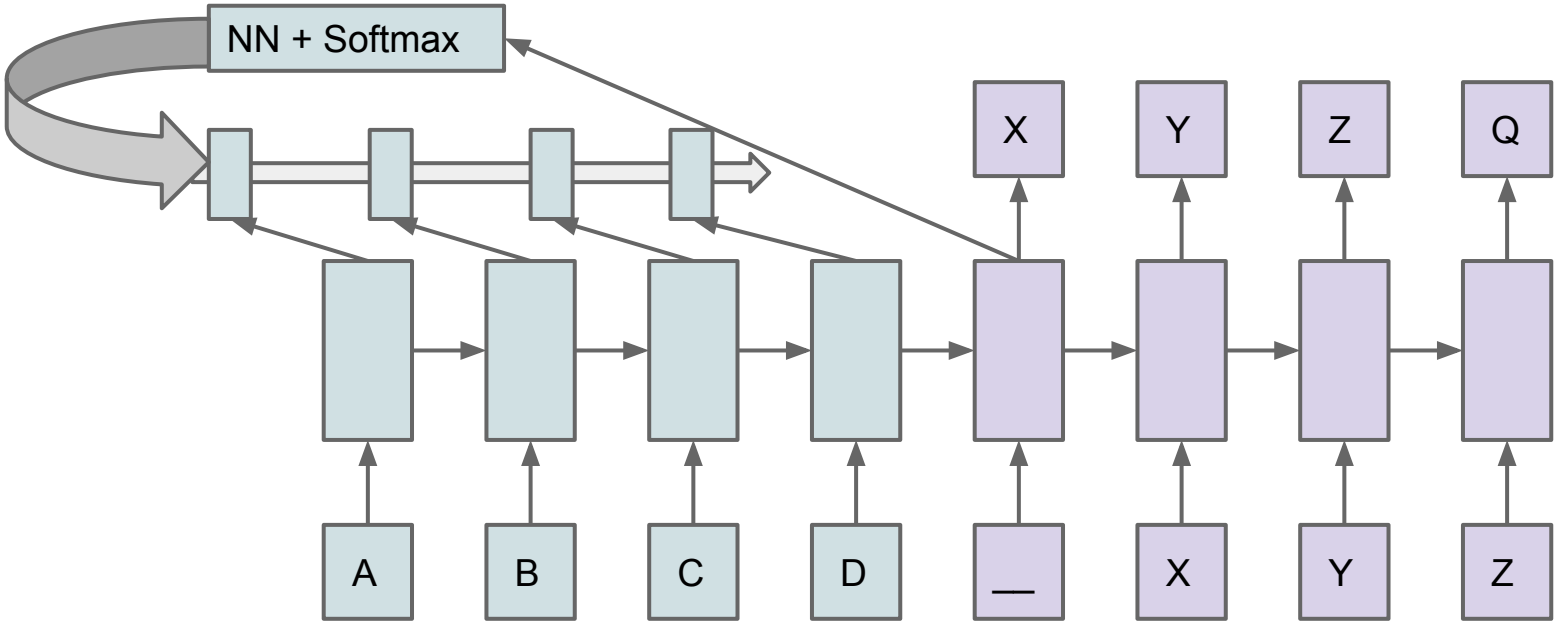
Map **d** to **a** = [0.1 0.9 0.0 0.0] E.g. **a** = s(**Wd**)

Compute: **d’** = 0.1*$\mathbf{e}_1$ + 0.9*$\mathbf{e}_2$ **+** 0*$\mathbf{e}_3$ **+**0*$\mathbf{e}_4$

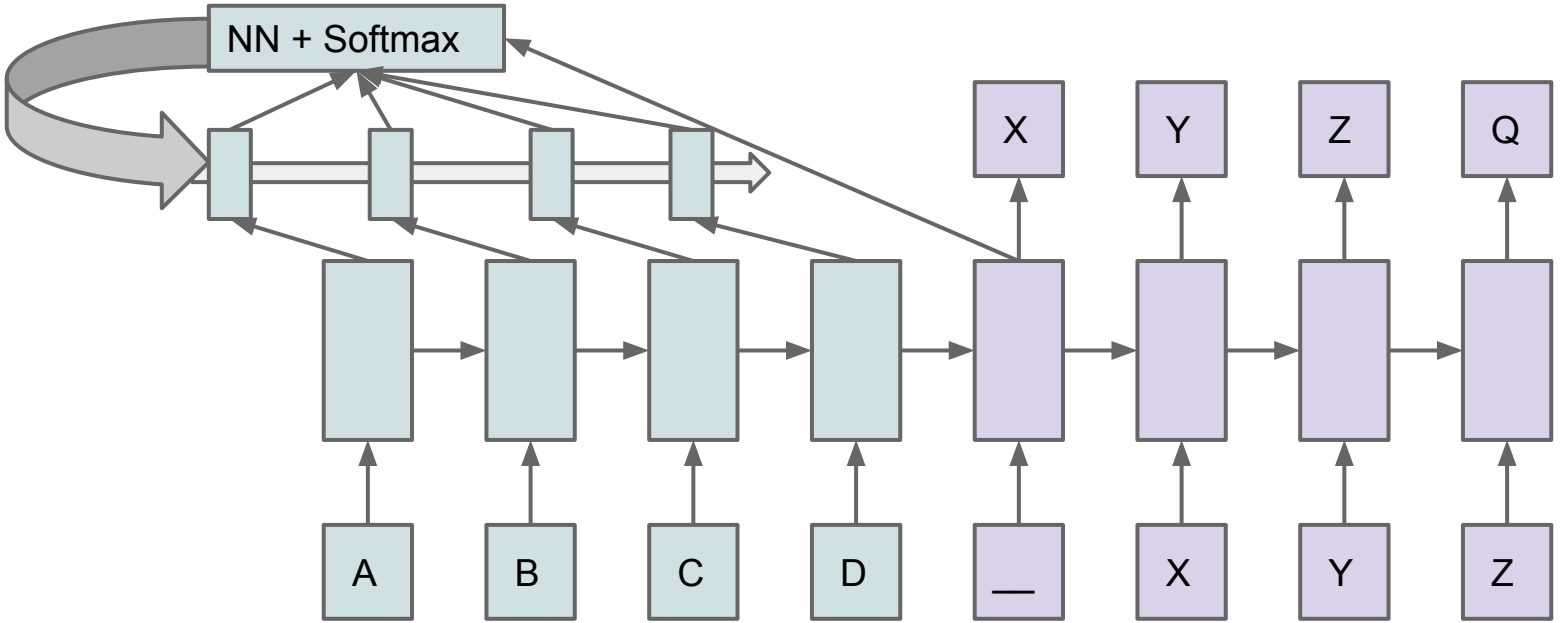
Set **d** = [**d d’**]



Position Based Attention



Content Based Attention



Content Based Attention

Inputs: “I am a cat.”

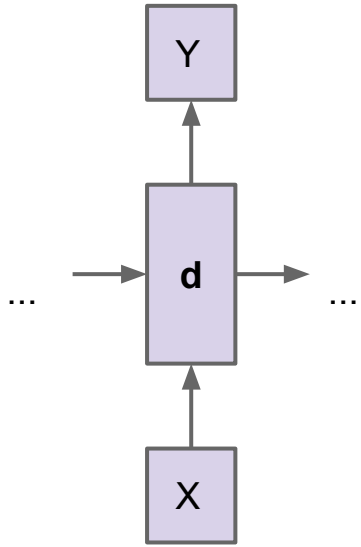
Input RNN states: $\mathbf{e}_1$ $\mathbf{e}_2$ $\mathbf{e}_3$ $\mathbf{e}_4$

Decoder RNN state: **d**

Compute $\mathbf{d}^T\mathbf{e}_i$: [0.0 0.05 0.9 0.05]

Compute: $\mathbf{d'} = 0*\mathbf{e}_1 + 0.05*\mathbf{e}_2 + 0.9*\mathbf{e}_3 + 0.05*\mathbf{e}_4$

Set **d** = [**d** **d'**]



Content Based Attention

Attention [Bahdanau, Cho and Bengio, 2014]

$$u_j = v^T \tanh(W_1 e_j + W_2 d) \quad j \in (1, \ldots, n)$$

$$a_j = \text{softmax}(u_j) \quad j \in (1, \ldots, n)$$

$$d' = \sum_{j=1}^{n} a_j e_j$$

Neural Turing Machine & Memory Networks:
Given a key vector **k** and a strength scalar *s* emitted by the controller, get a weighting (w_1,…,w_N) over memory locations where

$$w_i = \frac{e^{\hat{w}_i}}{\sum_j e^{\hat{w}_j}} \qquad \hat{w}_i = \log(1 + e^s) C(\mathbf{k}, \mathbf{m_i})$$

$$C(\mathbf{k}, \mathbf{m_i}) = \frac{\mathbf{k} \cdot \mathbf{m_i}}{\|\mathbf{k}\| \|\mathbf{m_i}\|}$$

and $C(\mathbf{k}, \mathbf{m}_i)$ is the cosine similarity between the key and the word $\mathbf{m}_i$ at memory location *i*

Frontiers: Learning Programs

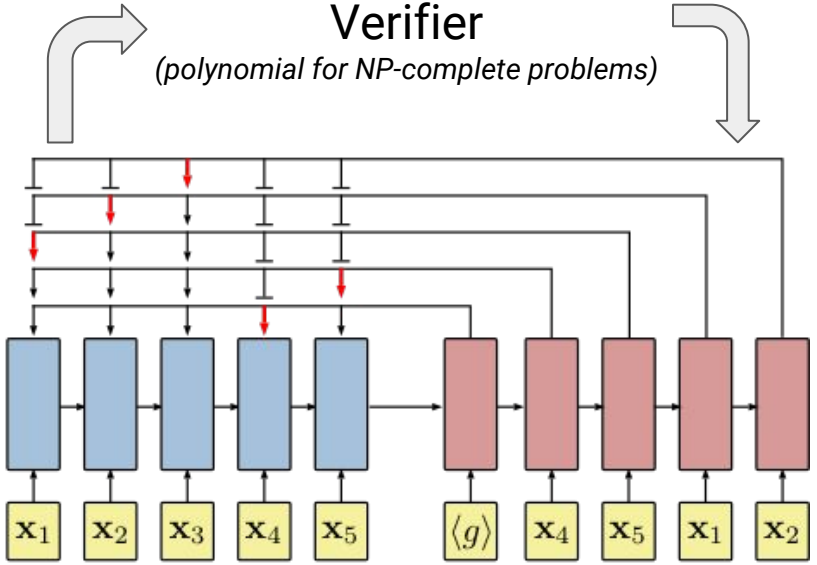
LOTS of related work: [Zaremba et al, 2014][Reed et al, 2015][Neelakantan et al, 2015][Vinyals et al, 2015]...

RL-PtrNets [Bello, Pham, et al 2016]

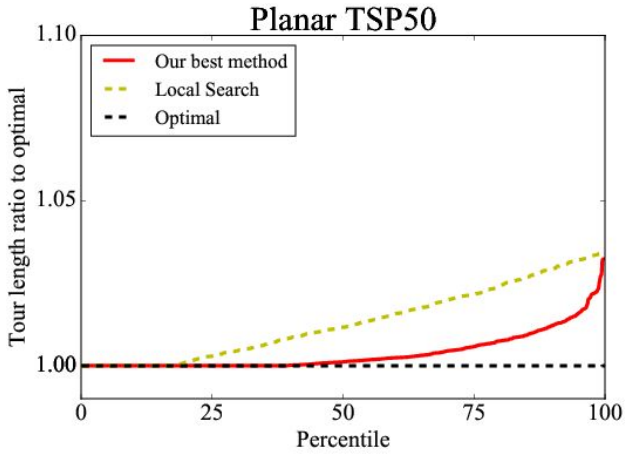
Actor-critic training using the expected combinatorial optimization objective

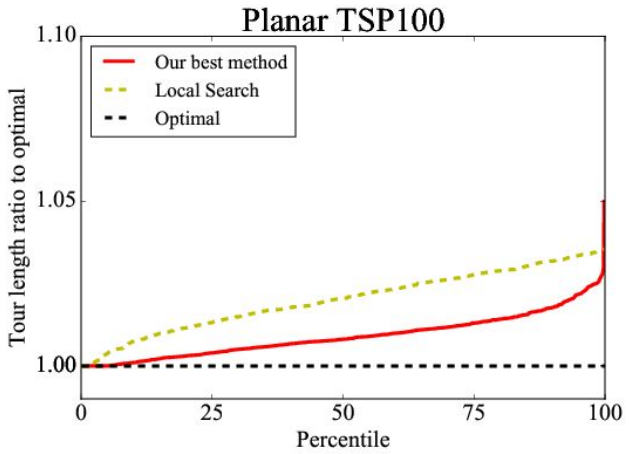
Different configurations:

- *Train* on training set
- *Search* at inference time
- *Refine* parameters at inference time



Frontiers: Learning Programs





Average tour lengths on TSP100 (~ 10^{157} solutions)

Christofides: 9.18 | Local search: 7.99 | RL-PtrNets: 7.83 | Optimal: 7.77