Seq2Seq ICML Tutorial

Oriol Vinyals and Navdeep Jaitly

@OriolVinyalsML | @NavdeepLearning

Site: https://sites.google.com/view/seq2seq-icml17

Sydney, Australia, 2017

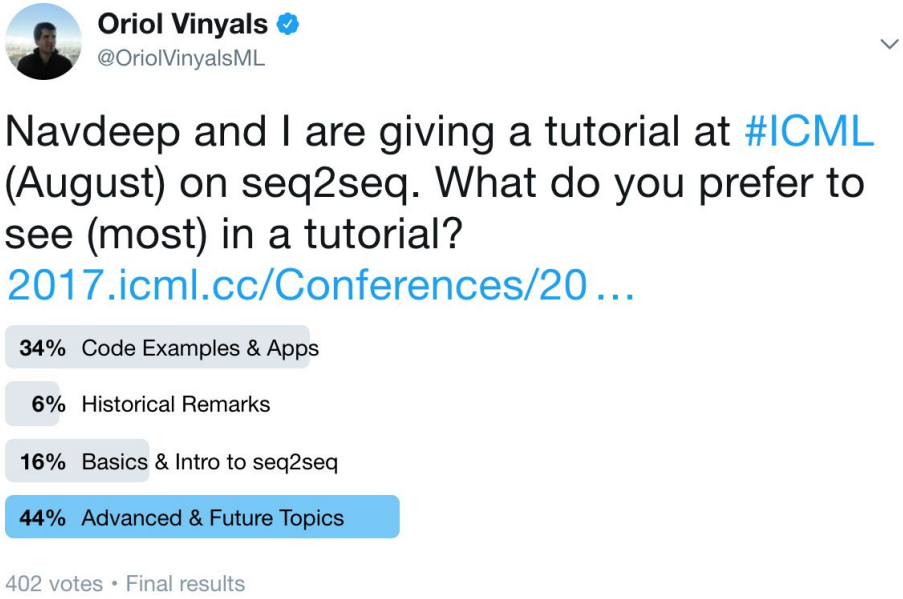
Outline

- An Overview of seq2seq (~25 minutes)
  - Applications
- Attention (~20 minutes)
  - Applications
- Break (15 mins)
- Loss functions (~15 minutes)
- Advanced Topics
  - Autoregressive Models (~5 mins)
  - Online Seq2Seq (~10 mins)
- New Architectures (~10 mins)
- Questions, discussion, etc. (15 minutes)

Seq2Seq Overview

What we WON'T cover

- Machine Learning Basics
- Deep Learning Basics
- (too many) Historical Remarks



Related Tutorials:

https://sites.google.com/site/acl16nmt/

Sequences!

- Words, Letters

> *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

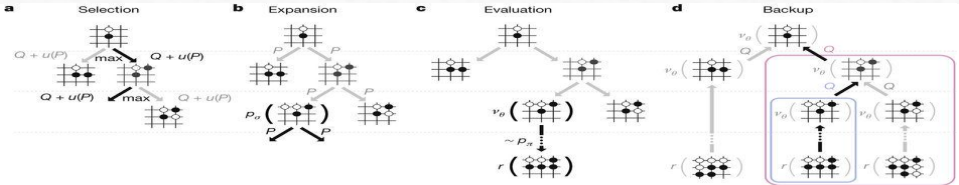
- Speech

- Images, Videos

- Programs

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

- Sequential Decision Making (RL)



Classical Models for Sequence Prediction

- Sequence prediction was classically handled as a structured prediction tasks
  - Most were built on conditional independence assumptions
  - Other such as DAGGER were based on supervisory signals and auxiliary information

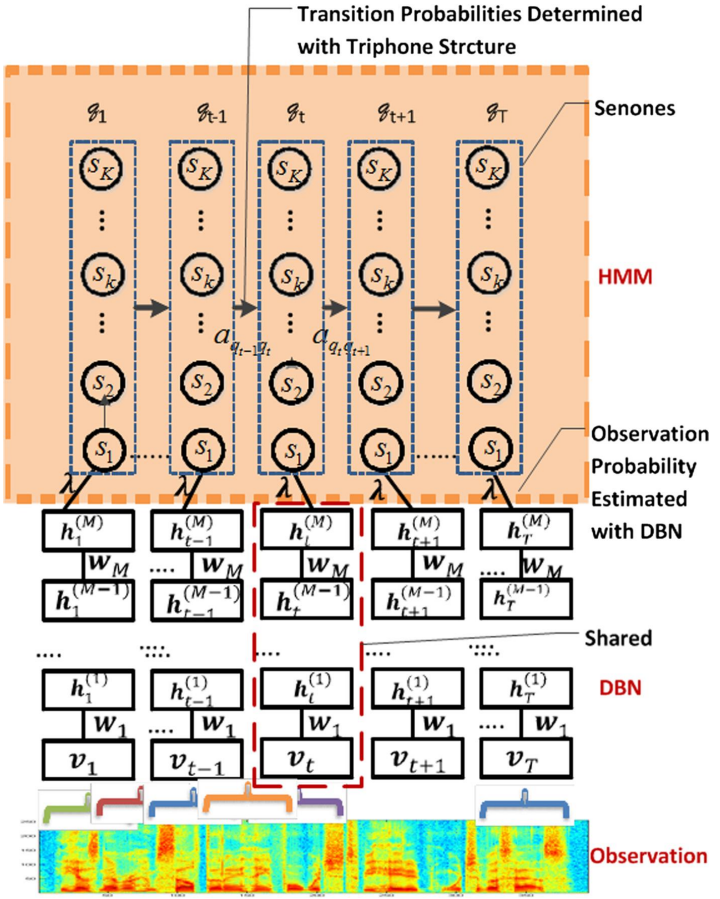
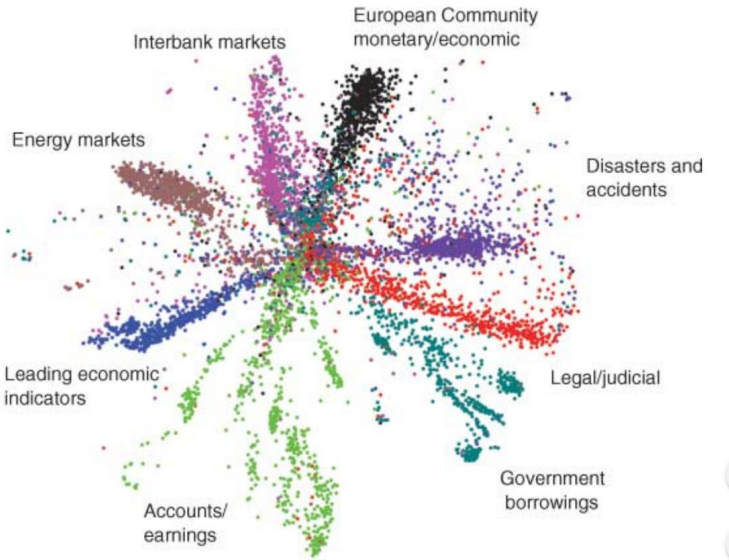


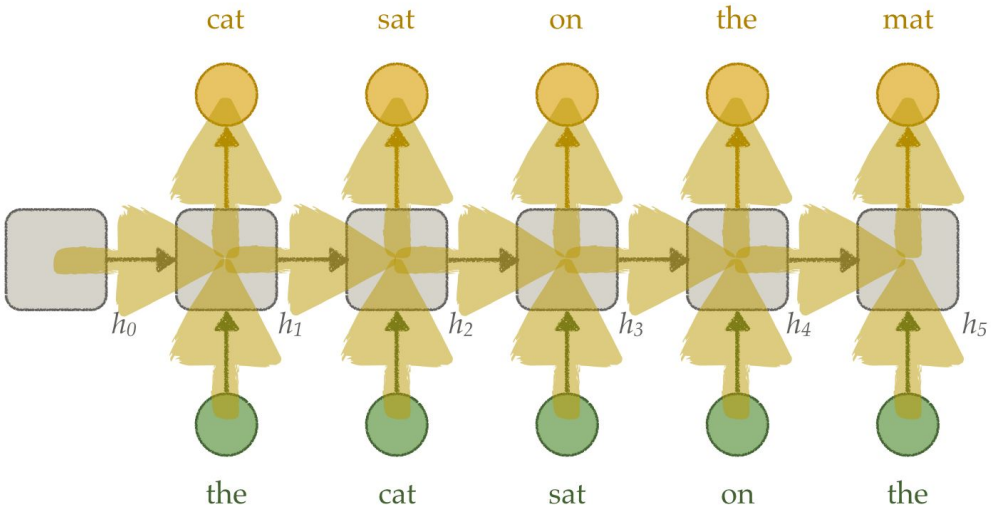
Figure credit: Li Deng

Two Key Ingredients

Neural Embeddings



Recurrent Language Models



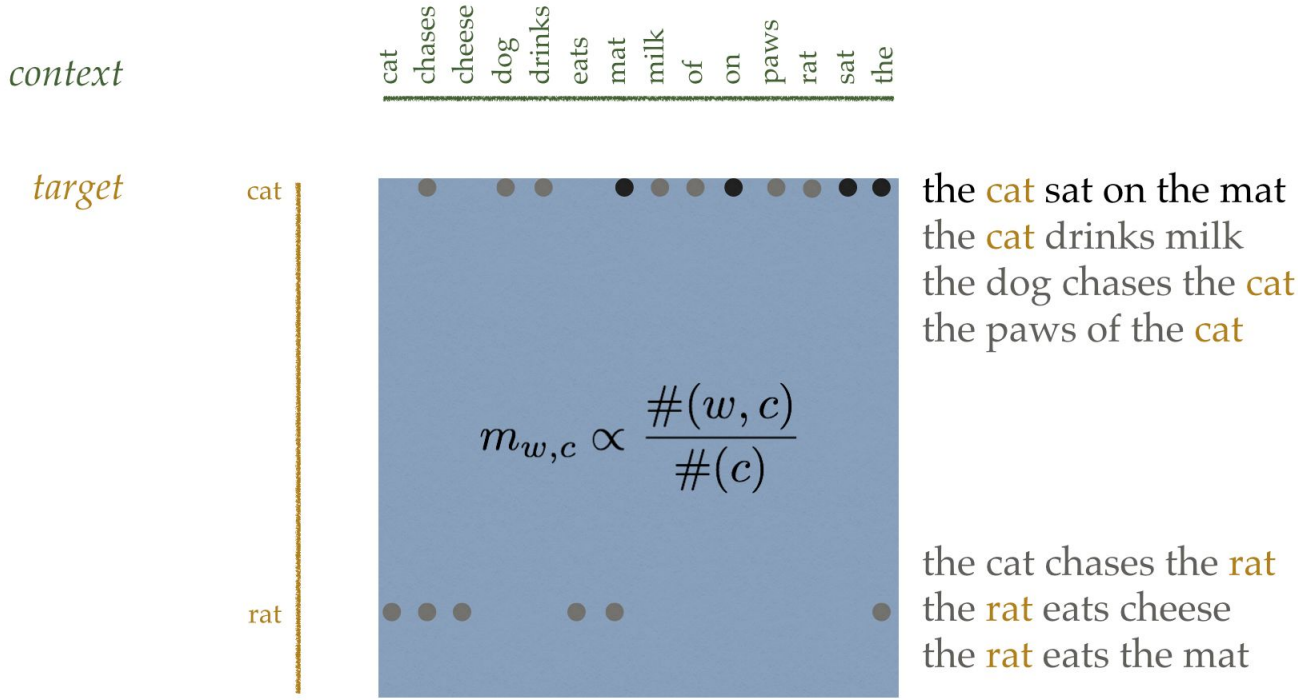
1. Hinton, G., Salakhutdinov, R. "Reducing the Dimensionality of Data with Neural Networks." *Science (2006)*
2. Mikolov, T., et al. "Recurrent neural network based language model." *Interspeech (2010)*

Language Models

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

n-grams



Slide Credit: Piotr Mirowski

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

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

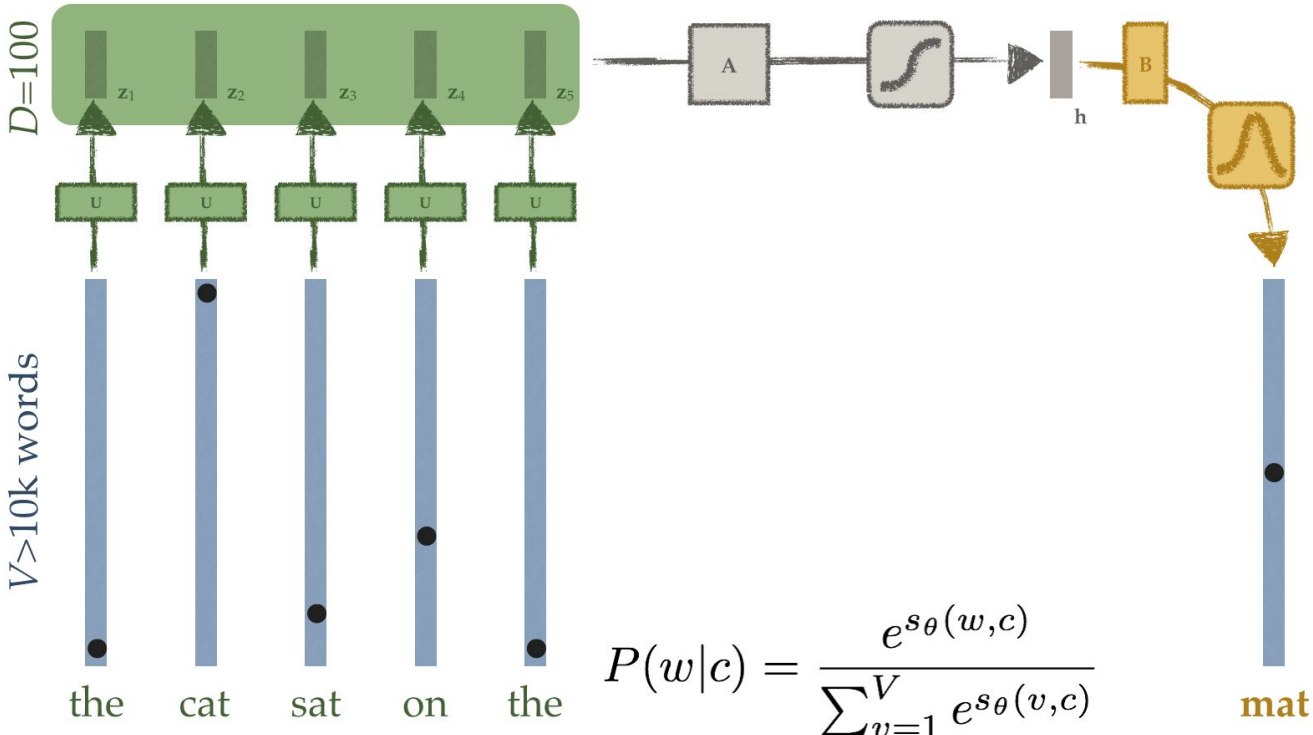
the	cat	sat	on	the	mat	$P(w_1)$
the	**cat**	sat	on	the	mat	$P(w_2 \mid w_1)$
the	cat	**sat**	on	the	mat	$P(w_3 \mid w_2, w_1)$
the	cat	sat	**on**	the	mat	$P(w_4 \mid w_3, w_2, w_1)$
the	cat	sat	on	**the**	mat	$P(w_5 \mid w_4, w_3, w_2, w_1)$
the	cat	sat	on	the	**mat**	$P(w_6 \mid w_5, w_4, w_3, w_2,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

Slide Credit: Piotr Mirowski

A Key Insight: vectorizing context

Bengio, Y. et al., "A Neural Probabilistic Language Model", *JMLR (2001, 2003)*
Mnih, A., Hinton, G., "Three new graphical models for statistical language modeling", *ICML 2007*

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

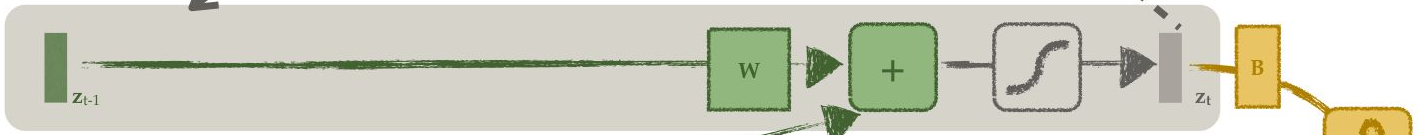


$$P(w|c) = \frac{e^{s_\theta(w,c)}}{\sum_{v=1}^{V} e^{s_\theta(v,c)}}$$

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

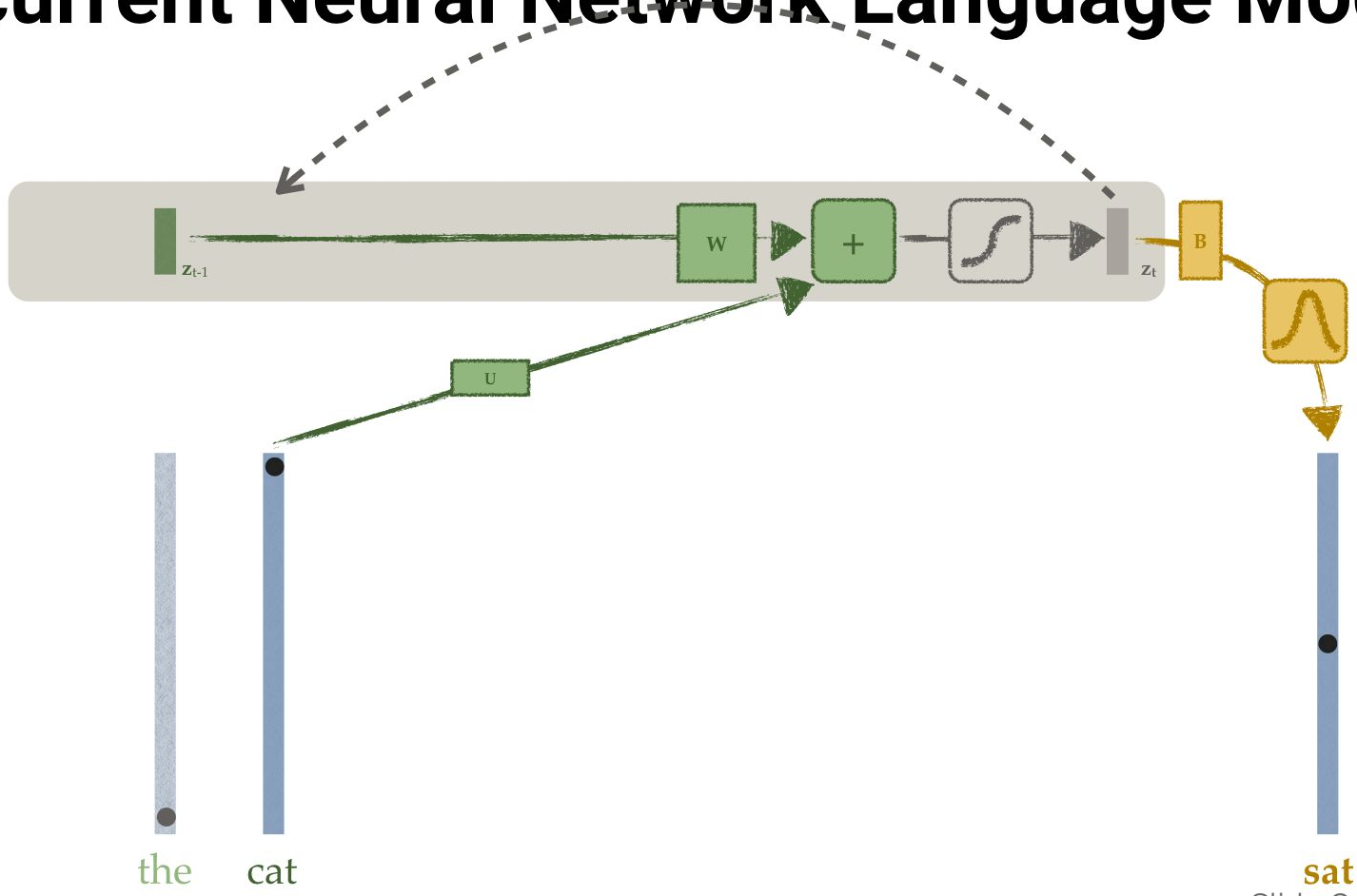
$$z_t = \tanh(W z_{t-1} + U w_t)$$
$$p(w_{t+1}) = softmax(B z_t)$$

the

cat

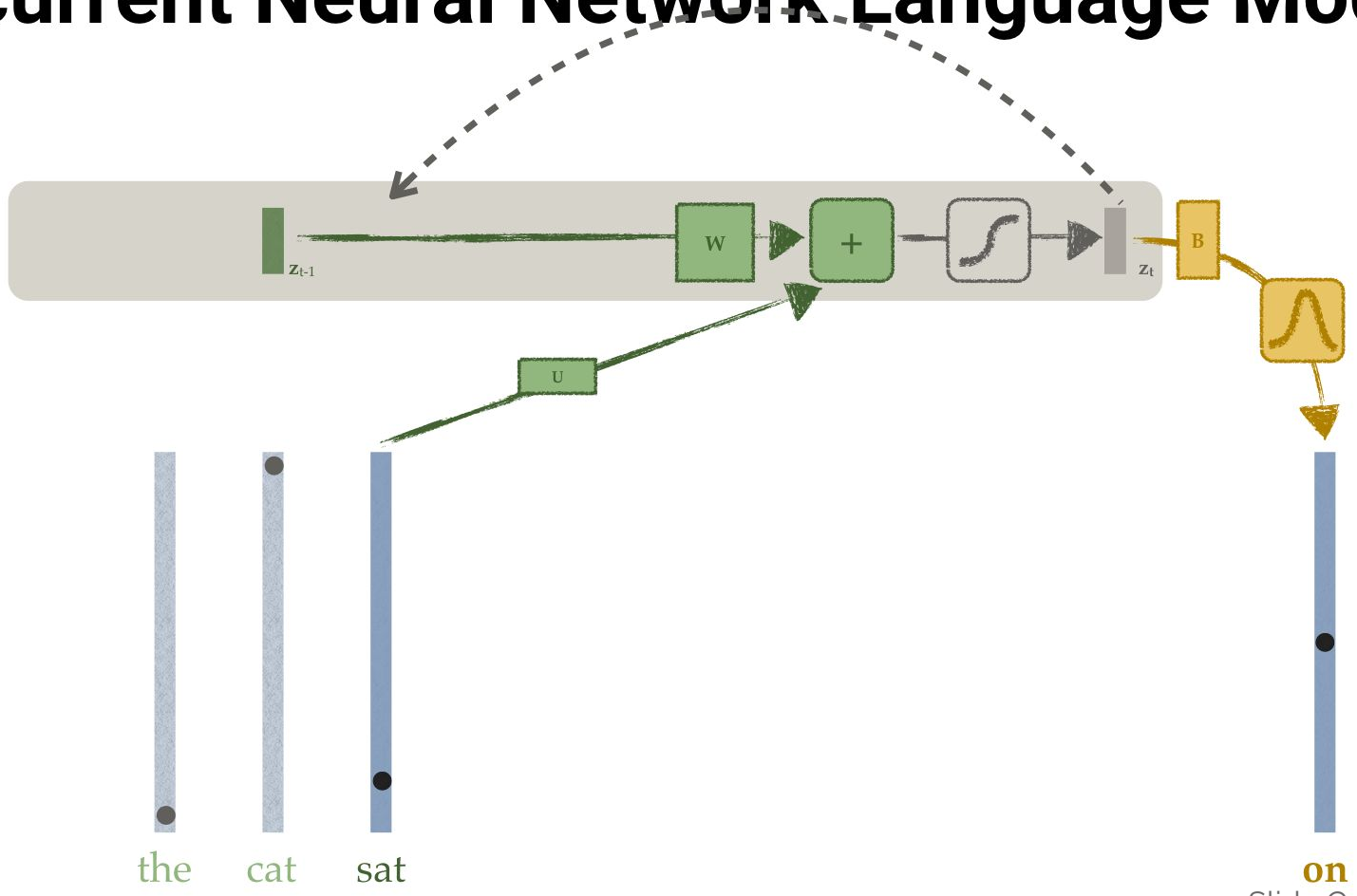
Slide Credit: Piotr Mirowski

Recurrent Neural Network Language Models



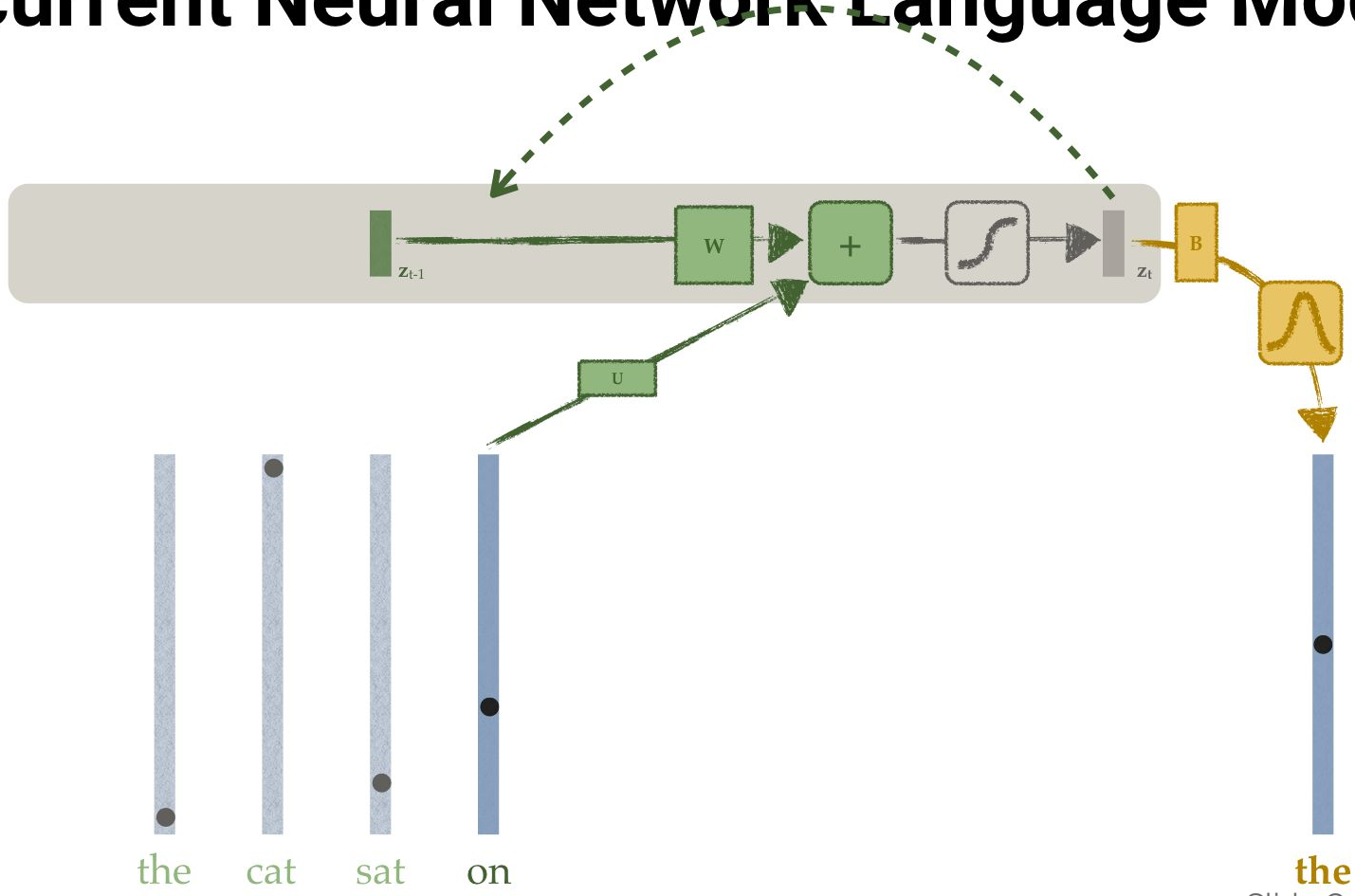
Slide Credit: Piotr Mirowski

Recurrent Neural Network Language Models



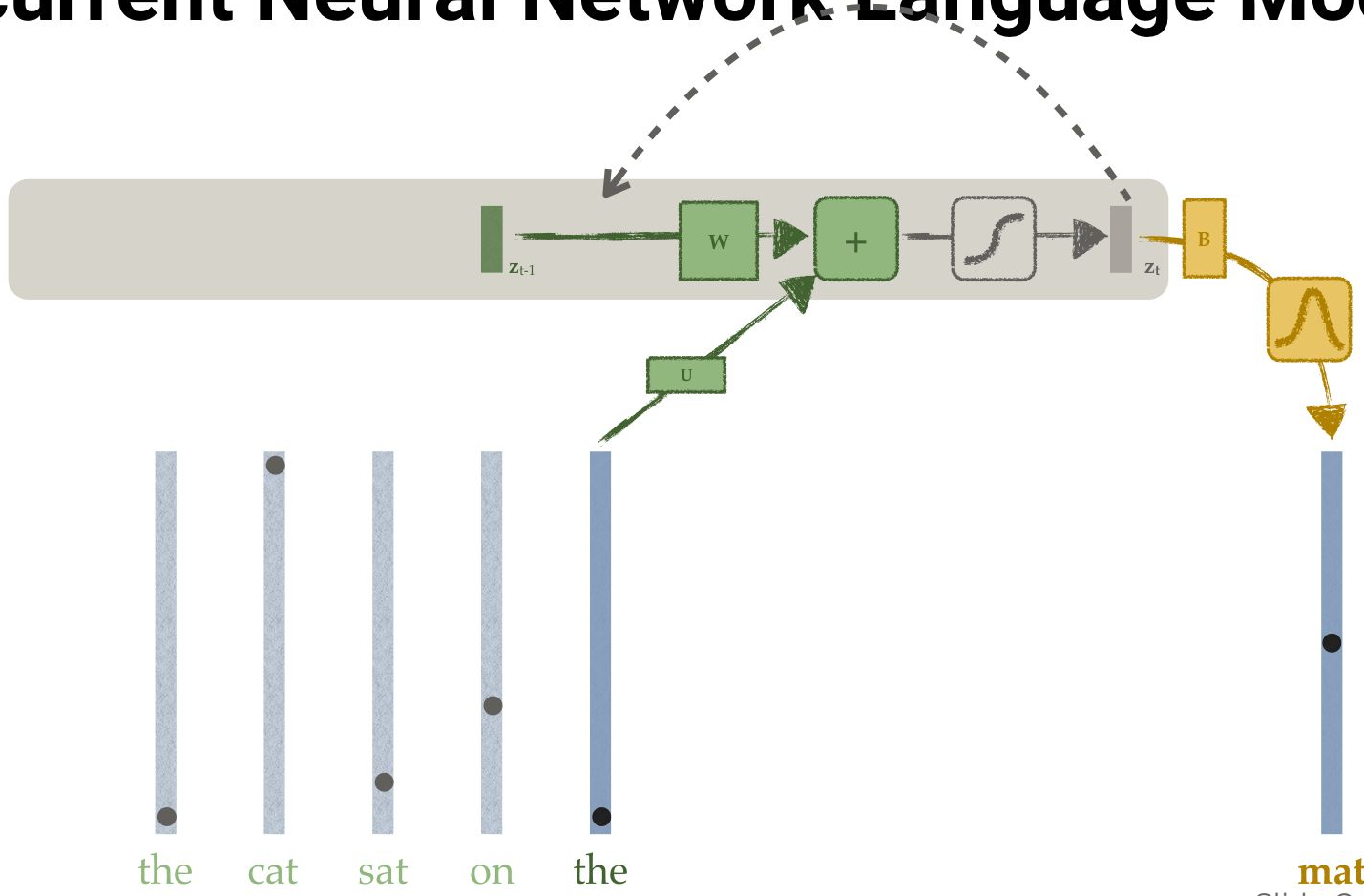
Slide Credit: Piotr Mirowski

Recurrent Neural Network Language Models



Slide Credit: Piotr Mirowski

Recurrent Neural Network Language Models

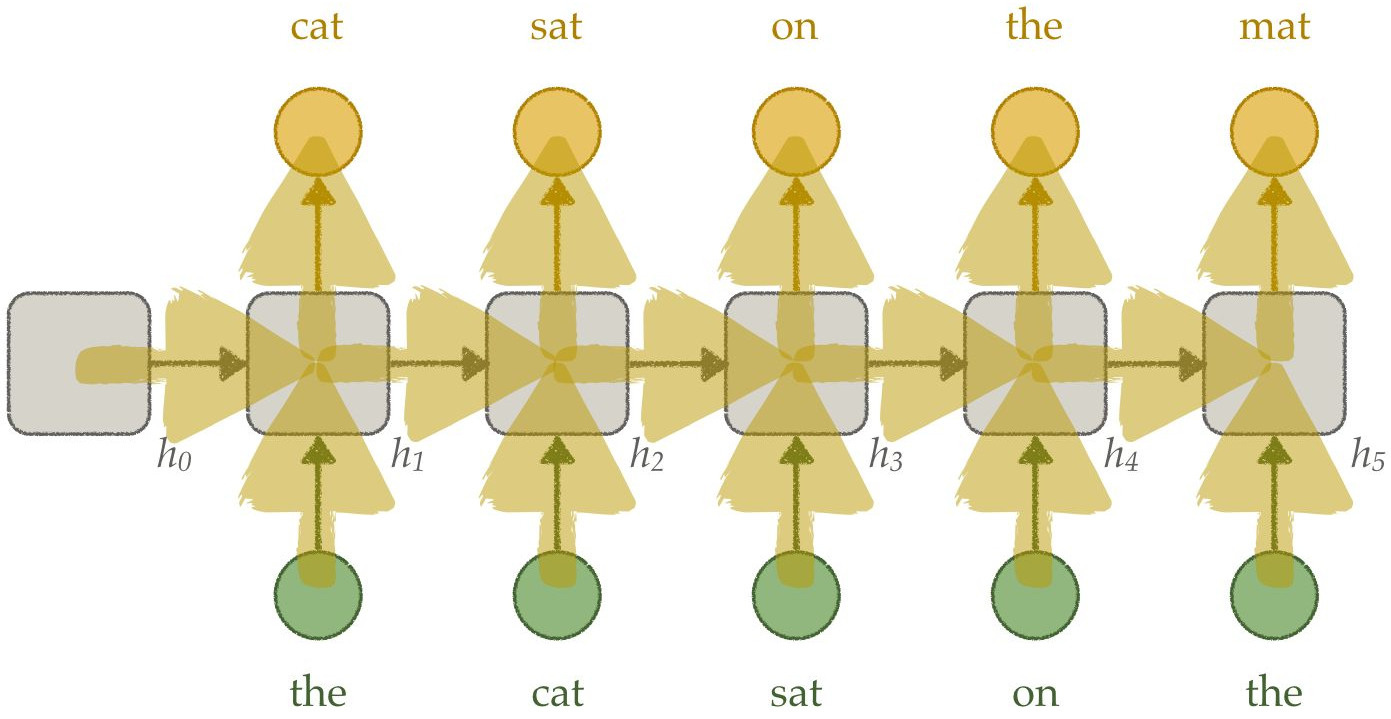


Slide Credit: Piotr Mirowski

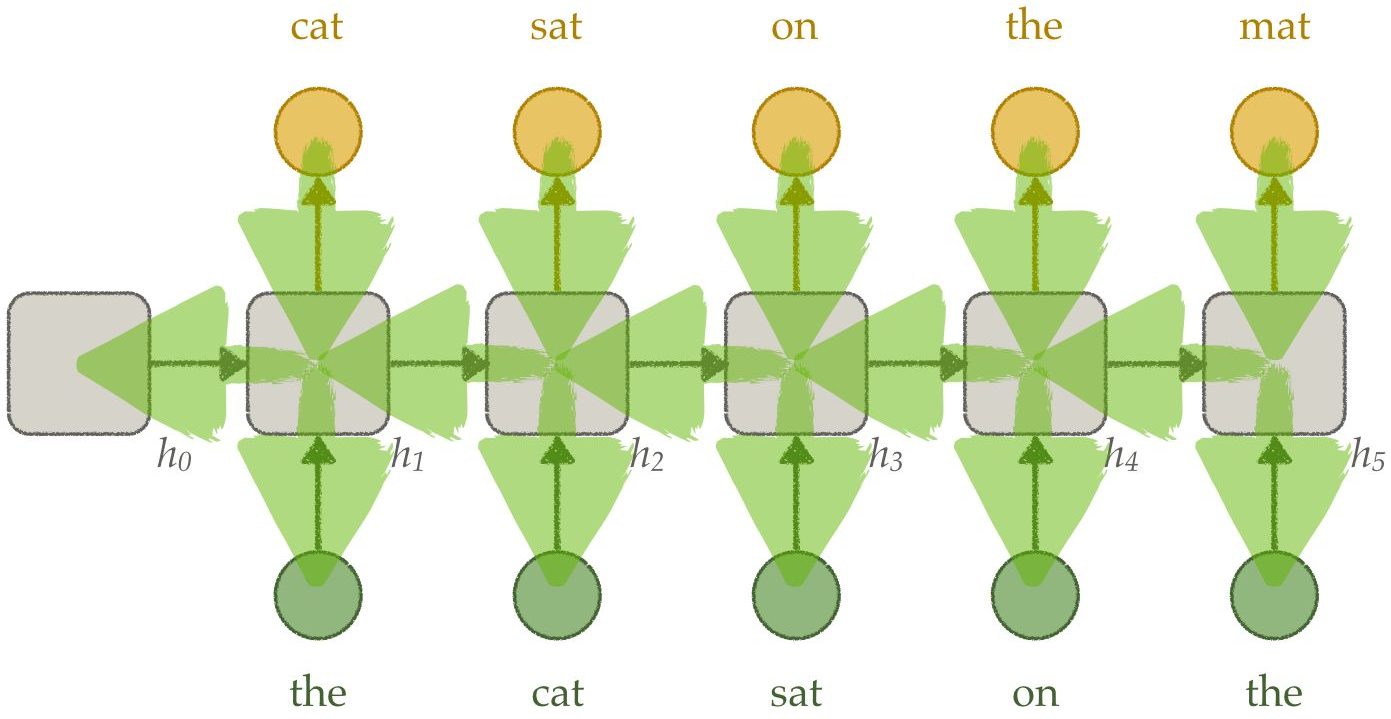
What do we Optimize?

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

Recurrent Neural Network Language Models



Recurrent Neural Network Language Models



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.

Seq2Seq

Joint Language and Translation Modeling with Recurrent Neural Networks

Michael Auli, Michel Galley, Chris Quirk, Geoffrey Zweig
Microsoft Research
Redmond, WA, USA
{michael.auli,mgalley,chrisq,gzweig}@microsoft.com

Abstract

We present a joint language and translation model based on a recurrent neural network which predicts target words based on an unbounded history of both source and target words. The weaker independence assumptions of this model result in a vastly larger search space compared to related feed-forward-based language or translation models. We tackle this issue with a new lattice rescoring algorithm and demonstrate its effectiveness empirically. Our joint model builds on a well known recurrent neural network language model (Mikolov, 2012) augmented by a layer of additional inputs from the source language. We show competitive accuracy compared to the traditional channel model features. Our best results improve the output of a system trained on WMT 2012 French-English data by up to 1.5 BLEU, and by 1.1 BLEU on average across several test sets.

1 Introduction

Recently, several feed-forward neural network-based language and translation models have achieved impressive accuracy improvements on statistical machine translation tasks (Allauzen et al., 2011; Le et al., 2012b; Schwenk et al., 2012). In this paper we focus on recurrent neural network architectures, which have recently advanced the state of the art in language modeling (Mikolov et al., 2010; Mikolov et al., 2011a; Mikolov, 2012), outperforming multi-layer feed-forward based networks in both perplexity and word error rate in speech recognition (Arisoy et al., 2012; Sundermeyer et al., 2013). The major attraction of recurrent architectures is their potential to capture long-span dependencies since predictions are based on an *unbounded history* of previous words. This is in contrast to feed-forward networks as well as conventional n-gram models, both of which are limited to fixed-length contexts. Building on the success of recurrent architectures, we base our joint language and translation model on an extension of the recurrent neural network language model (Mikolov and Zweig, 2012) that introduces a layer of additional inputs (§2).

Most previous work on neural networks for speech recognition or machine translation used a rescoring setup based on n-best lists (Arisoy et al., 2012; Mikolov, 2012) for evaluation, thereby side stepping the algorithmic and engineering challenges of direct decoder-integration.[1] Instead, we exploit *lattices*, which offer a much richer representation of the decoder output, since they compactly encode an exponential number of translation hypotheses in polynomial space. In contrast, n-best lists are typically very redundant, representing only a few combinations of top scoring arcs in the lattice. A major challenge in lattice rescoring with a recurrent neural network model is the effect of the unbounded history on search since the usual dynamic programming assumptions which are exploited for efficiency do not hold up anymore. We apply a novel algorithm to the task of rescoring with an unbounded language model and empirically demonstrate its effectiveness (§3).

The algorithm proves robust, leading to significant improvements with the recurrent neural network language model over a competitive n-gram baseline across several language pairs. We even observe consistent gains when pairing the model with a large n-gram model trained on up to 575 times more

[1] One notable exception is Le et al. (2012a) who rescore reordering lattices with a feed-forward network-based model.

1044

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1044–1054, Seattle, Washington, USA, 18-21 October 2013. ©2013 Association for Computational Linguistics

Recurrent Continuous Translation Models

Nal Kalchbrenner Phil Blunsom
Department of Computer Science
University of Oxford
{nal.kalchbrenner,phil.blunsom}@cs.ox.ac.uk

Abstract

We introduce a class of probabilistic continuous translation models called Recurrent Continuous Translation Models that are purely based on continuous representations for words, phrases and sentences and do not rely on alignments or phrasal translation units. The models have a generation and a conditioning aspect. The generation of the translation is modelled with a target Recurrent Language Model, whereas the conditioning on the source sentence is modelled with a Convolutional Sentence Model. Through various experiments, we show first that our models obtain a perplexity with respect to gold translations that is > 43% lower than that of state-of-the-art alignment-based translation models. Secondly, we show that they are remarkably sensitive to the word order, syntax, and meaning of the source sentence despite lacking alignments. Finally we show that they match a state-of-the-art system when rescoring n-best lists of translations.

1 Introduction

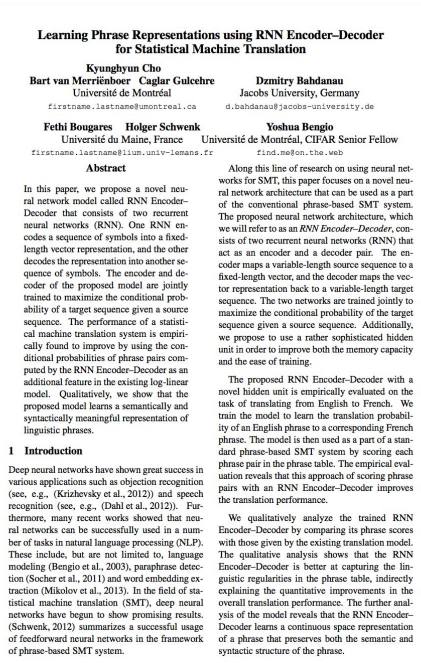
In most statistical approaches to machine translation the basic units of translation are phrases that are composed of one or more words. A crucial component of translation systems are models that estimate translation probabilities for pairs of phrases, one phrase being from the source language and the other from the target language. Such models count phrase pairs and their occurrences as distinct if the surface forms of the phrases are distinct. Although distinct phrase pairs often share significant similarities, linguistic or otherwise, they do not share statistical weight in the models' estimation of their translation probabilities. Besides ignoring the similarity of phrase pairs, this leads to general sparsity issues. The estimation is sparse or skewed for the large number of rare or unseen phrase pairs, which grows exponentially in the length of the phrases, and the generalisation to other domains is often limited.

Continuous representations have shown promise at tackling these issues. Continuous representations for words are able to capture their morphological, syntactic and semantic similarity (Collobert and Weston, 2008). They have been applied in continuous language models demonstrating the ability to overcome sparsity issues and to achieve state-of-the-art performance (Bengio et al., 2003; Mikolov et al., 2010). Word representations have also shown a marked sensitivity to conditioning information (Mikolov and Zweig, 2012). Continuous representations for characters have been deployed in character-level language models demonstrating notable language generation capabilities (Sutskever et al., 2011). Continuous representations have also been constructed for phrases and sentences. The representations are able to carry similarity and task dependent information, e.g. sentiment, paraphrase or dialogue labels, significantly beyond the word level and to accurately predict labels for a highly diverse range of unseen phrases and sentences (Grefenstette et al., 2011; Socher et al., 2011; Socher et al., 2012; Hermann and Blunsom, 2013; Kalchbrenner and Blunsom, 2013).

Phrase-based continuous translation models were first proposed in (Schwenk et al., 2006) and re-

1700

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1700–1709, Seattle, Washington, USA, 18-21 October 2013. ©2013 Association for Computational Linguistics



Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation

Kyunghyun Cho
Bart van Merriënboer Caglar Gulcehre Dzmitry Bahdanau
Université de Montréal Jacobs University, Germany
firstname.lastname@umontreal.ca d.bahdanau@jacobs-university.de

Fethi Bougares Holger Schwenk Yoshua Bengio
Université du Maine, France Université de Montréal, CIFAR Senior Fellow
firstname.lastname@lium.univ-lemans.fr find.me@on.the.web

Abstract

In this paper, we propose a novel neural network model called RNN Encoder–Decoder that consists of two recurrent neural networks (RNN). One RNN encodes a sequence of symbols into a fixed-length vector representation, and the other decodes the representation into another sequence of symbols. The encoder and decoder of the proposed model are jointly trained to maximize the conditional probability of a target sequence given a source sequence. The performance of a statistical machine translation system is empirically found to improve by using the conditional probabilities of phrase pairs computed by the RNN Encoder–Decoder as an additional feature in the existing log-linear model. Qualitatively, we show that the proposed model learns a semantically and syntactically meaningful representation of linguistic phrases.

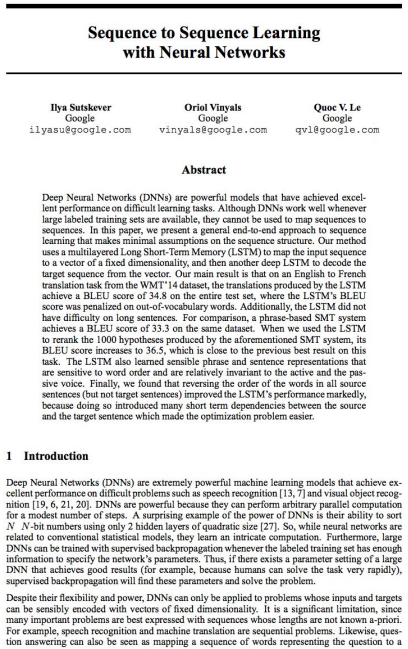
1 Introduction

Deep neural networks have shown great success in various applications such as objection recognition (see, e.g., (Krizhevsky et al., 2012)) and speech recognition (see, e.g., (Dahl et al., 2012)). Furthermore, many recent works showed that neural networks can be successfully used in a number of tasks in natural language processing (NLP). These include, but are not limited to, language modeling (Bengio et al., 2003), paraphrase detection (Socher et al., 2011) and word embedding extraction (Mikolov et al., 2013). In the field of statistical machine translation (SMT), deep neural networks have begun to show promising results. (Schwenk, 2012) summarizes a successful usage of feedforward neural networks in the framework of phrase-based SMT system.

Along this line of research on using neural networks for SMT, this paper focuses on a novel neural network architecture that can be used as a part of the conventional phrase-based SMT system. The proposed neural network architecture, which we will refer to as an *RNN Encoder–Decoder*, consists of two recurrent neural networks (RNN) that act as an encoder and a decoder pair. The encoder maps a variable-length source sequence to a fixed-length vector, and the decoder maps the vector representation back to a variable-length target sequence. The two networks are trained jointly to maximize the conditional probability of the target sequence given a source sequence. Additionally, we propose to use a rather sophisticated hidden unit in order to improve both the memory capacity and the ease of training.

The proposed RNN Encoder–Decoder with a novel hidden unit is empirically evaluated on the task of translating from English to French. We train the model to learn the translation probability of an English phrase to a corresponding French phrase. The model is then used as a part of a standard phrase-based SMT system by scoring each phrase pair in the phrase table. The empirical evaluation reveals that this approach of scoring phrase pairs with an RNN Encoder–Decoder improves the translation performance.

We qualitatively analyze the trained RNN Encoder–Decoder by comparing its phrase scores with those given by the existing translation model. The qualitative analysis shows that the RNN Encoder–Decoder is better at capturing the linguistic regularities in the phrase table, indirectly explaining the quantitative improvements in the overall translation performance. The further analysis of the model reveals that the RNN Encoder–Decoder learns a continuous space representation of a phrase that preserves both the semantic and syntactic structure of the phrase.



Sequence to Sequence Learning with Neural Networks

Ilya Sutskever Oriol Vinyals Quoc V. Le
Google Google Google
ilyasu@google.com vinyals@google.com qvl@google.com

Abstract

Deep Neural Networks (DNNs) are powerful models that have achieved excellent performance on difficult learning tasks. Although DNNs work well whenever large labeled training sets are available, they cannot be used to map sequences to sequences. In this paper, we present a general end-to-end approach to sequence learning that makes minimal assumptions on the sequence structure. Our method uses a multilayered Long Short-Term Memory (LSTM) to map the input sequence to a vector of a fixed dimensionality, and then another deep LSTM to decode the target sequence from the vector. Our main result is that on an English to French translation task from the WMT'14 dataset, the translations produced by the LSTM achieve a BLEU score of 34.8 on the entire test set, where the LSTM's BLEU score was penalized on out-of-vocabulary words. Additionally, the LSTM did not have difficulty on long sentences. For comparison, a phrase-based SMT system achieves a BLEU score of 33.3 on the same dataset. When we used the LSTM to rerank the 1000 hypotheses produced by the aforementioned SMT system, its BLEU score increases to 36.5, which is close to the previous best result on this task. The LSTM also learned sensible phrase and sentence representations that are sensitive to word order and are relatively invariant to the active and the passive voice. Finally, we found that reversing the order of the words in all source sentences (but not target sentences) improved the LSTM's performance markedly, because doing so introduced many short term dependencies between the source and the target sentence which made the optimization problem easier.

1 Introduction

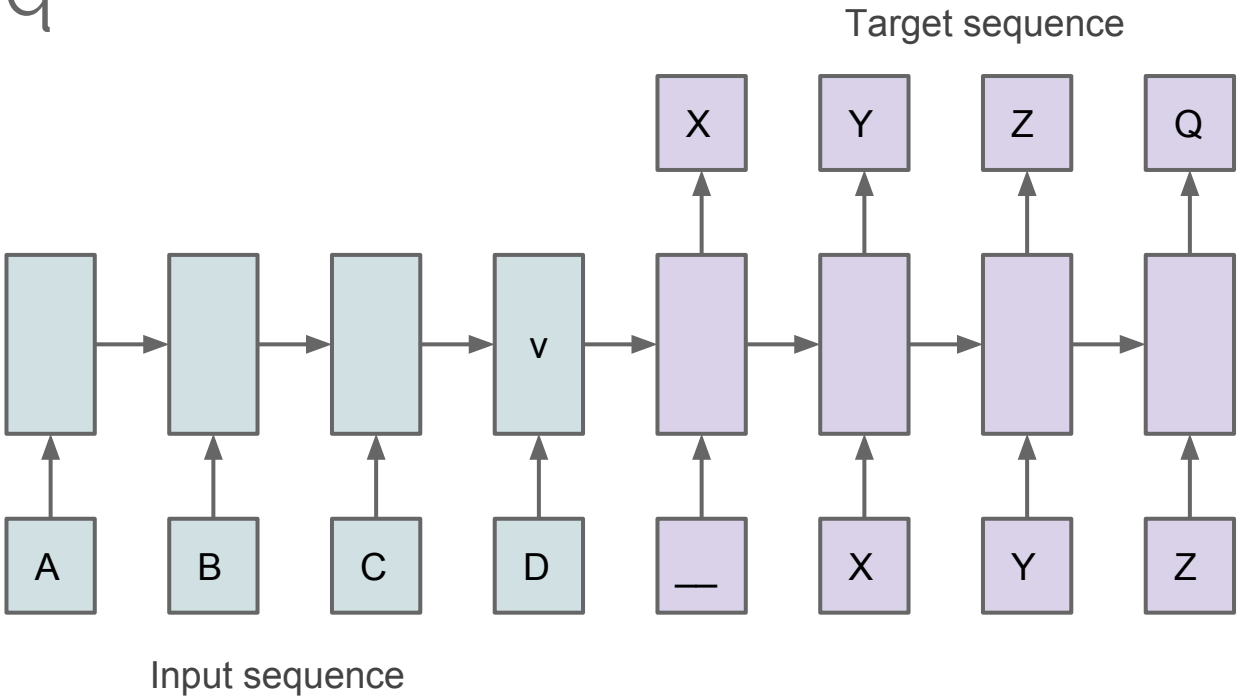
Deep Neural Networks (DNNs) are extremely powerful machine learning models that achieve excellent performance on difficult problems such as speech recognition [13, 7] and visual object recognition [19, 6, 21, 20]. DNNs are powerful because they can perform arbitrary parallel computation for a modest number of steps. A surprising example of the power of DNNs is their ability to sort N N-bit numbers using only 2 hidden layers of quadratic size [27]. So, while neural networks are related to conventional statistical models, they learn an intricate computation. Furthermore, large DNNs can be trained with supervised backpropagation whenever the labeled training set has enough information to specify the network's parameters. Thus, if there exists a parameter setting of a large DNN that achieves good results (for example, because humans can solve the task very rapidly), supervised backpropagation will find these parameters and solve the problem.

Despite their flexibility and power, DNNs can only be applied to problems whose inputs and targets can be sensibly encoded with vectors of fixed dimensionality. It is a significant limitation, since many important problems are best expressed with sequences whose lengths are not known a-priori. For example, speech recognition and machine translation are sequential problems. Likewise, question answering can also be seen as mapping a sequence of words representing the question to a

1

1. Auli, M., et al. "Joint Language and Translation Modeling with Recurrent Neural Networks." *EMNLP (2013)*
2. Kalchbrenner, N., et al. “Recurrent Continuous Translation Models.” *EMNLP (2013)*
3. Cho, K., et al. “Learning Phrase Representations using RNN Encoder-Decoder for Statistical MT.” *EMNLP (2014)*
4. Sutskever, I., et al. “Sequence to Sequence Learning with Neural Networks.” *NIPS (2014)*

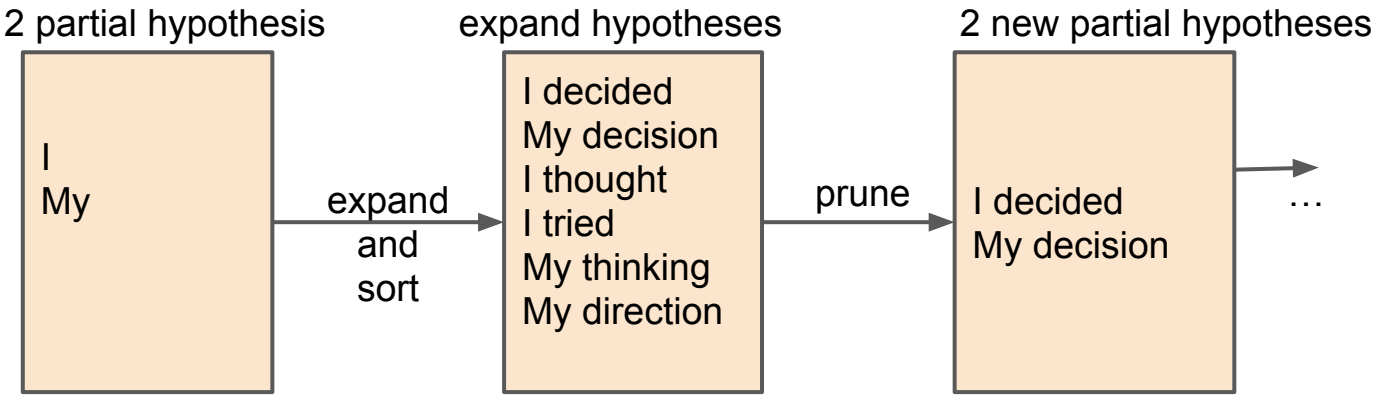
Seq2Seq



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

Decoding in a Nutshell (Beam Size 2)

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



Code

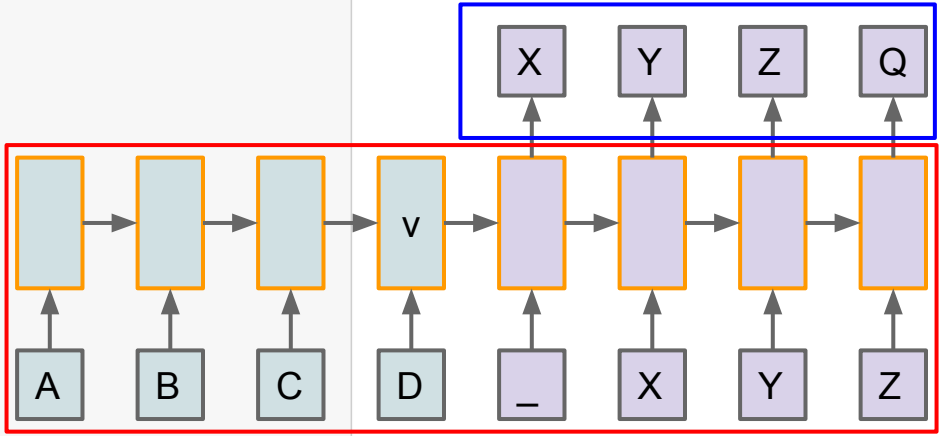
Source: https://github.com/keveman/tensorflow-tutorial/blob/master/PTB%20Word%20Language%20Modeling.ipynb

```
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)
```

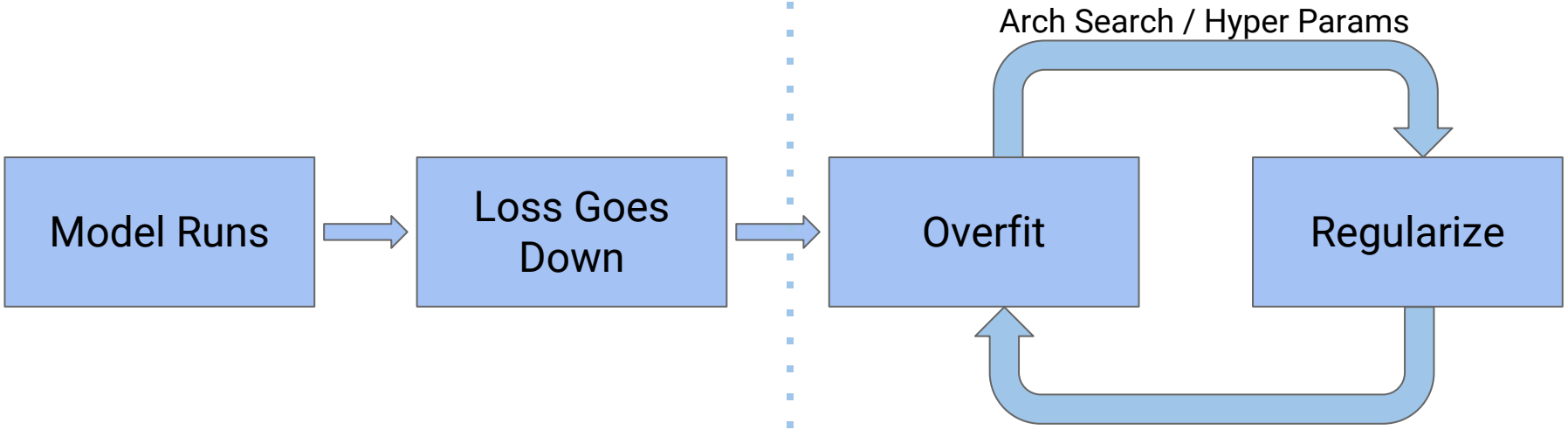
Code

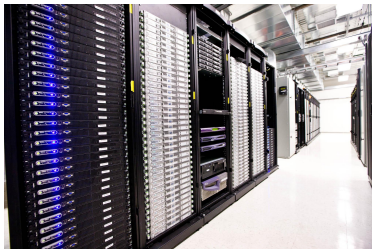
```
# words and targets are placeholders for [batch_size, num_steps]
# tensor of word and target ids
words = tf.placeholder(tf.int64, name='words')
targets = tf.placeholder(tf.int64, name='targets')

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
```

```
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)
```



Deep Learning Vicious Cycle





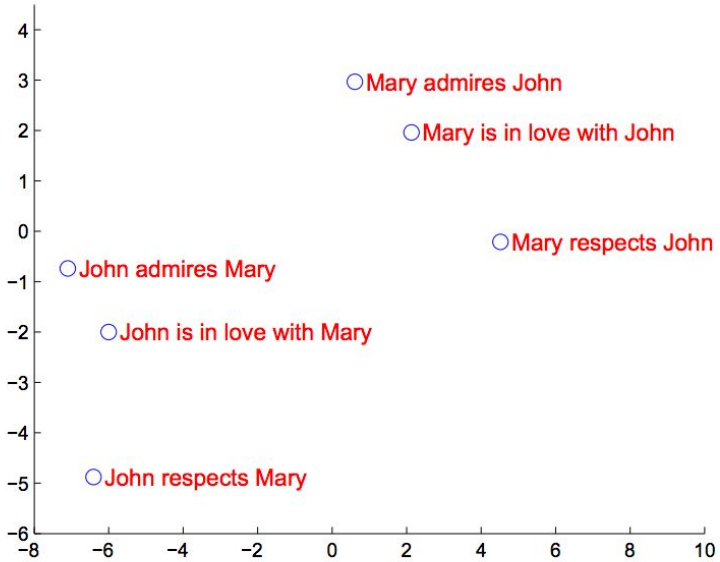
(Some) Tricks of the Trade

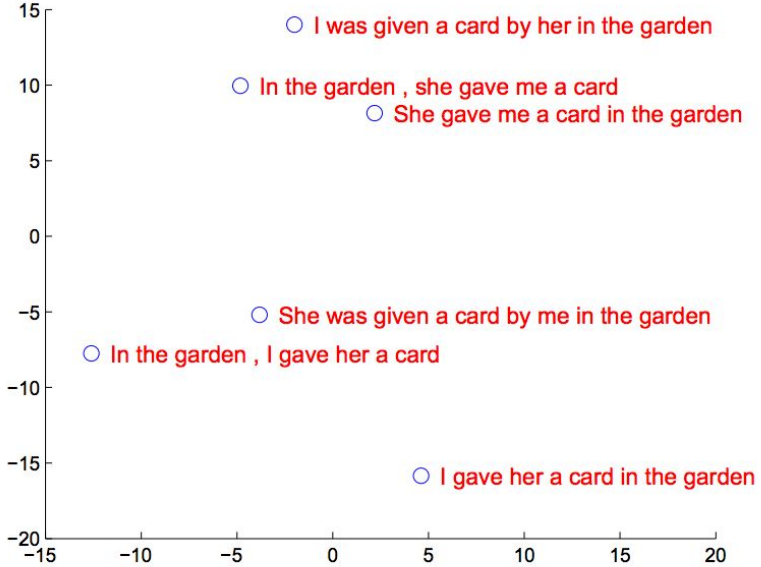
- Long sequences?
  - Attention
  - Bigger state
- Can't overfit?
  - Bigger hidden state
  - Deep LSTM + Skip Connections
- Overfit?
  - Dropout + Ensembles
- Tuning
  - Keep calm and decrease your learning rate
  - Initialization of parameters is critical (in seq2seq we used U(-0.05, 0.05))
  - Clip the gradients!
    - E.g. if ||grad|| > 5: grad = grad/||grad|| * 5

Applications

Machine Translation

Method	test BLEU score (ntst14)
Bahdanau et al. [2]	28.45
Baseline System [29]	33.30
Single forward LSTM, beam size 12	26.17
Single reversed LSTM, beam size 12	30.59
Ensemble of 5 reversed LSTMs, beam size 1	33.00
Ensemble of 2 reversed LSTMs, beam size 12	33.27
Ensemble of 5 reversed LSTMs, beam size 2	34.50
Ensemble of 5 reversed LSTMs, beam size 12	**34.81**

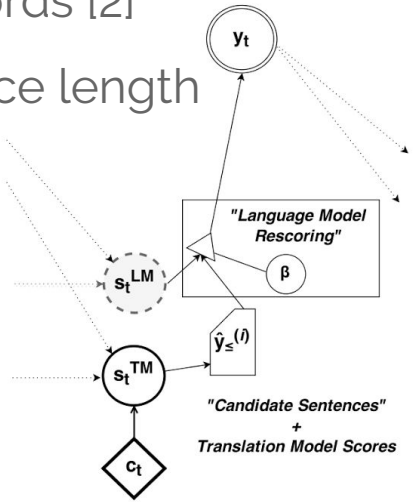




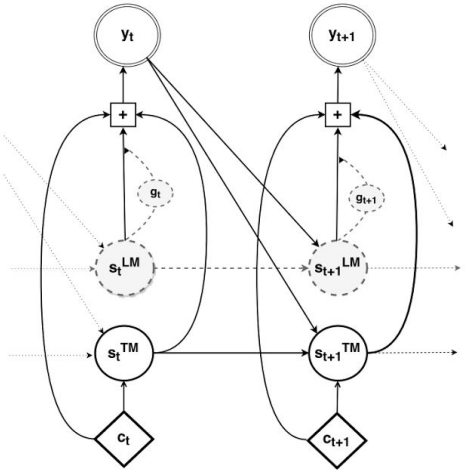
Sutskever, I., et al. “Sequence to Sequence Learning with Neural Networks.” *NIPS (2014)*

Machine Translation - other concerns

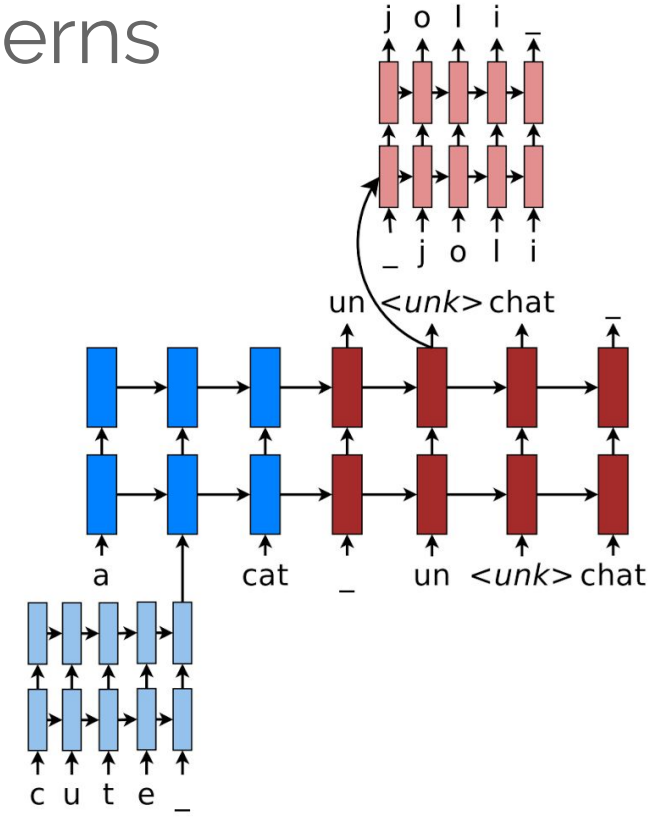
- Using Language Models [1]
- OOV words [2]
- Sequence length



(a) Shallow Fusion (Sec. 4.1)



(b) Deep Fusion (Sec. 4.2)



1. Gulcehre, C., et al. "On using monolingual corpora in neural machine translation." *arXiv* (2015).
2. Luong, T., and Manning, C. "Achieving open vocabulary neural MT with hybrid word-character models." *arXiv* (2016).

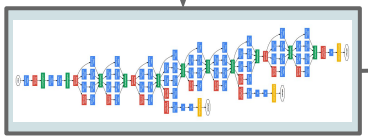
Image Captioning

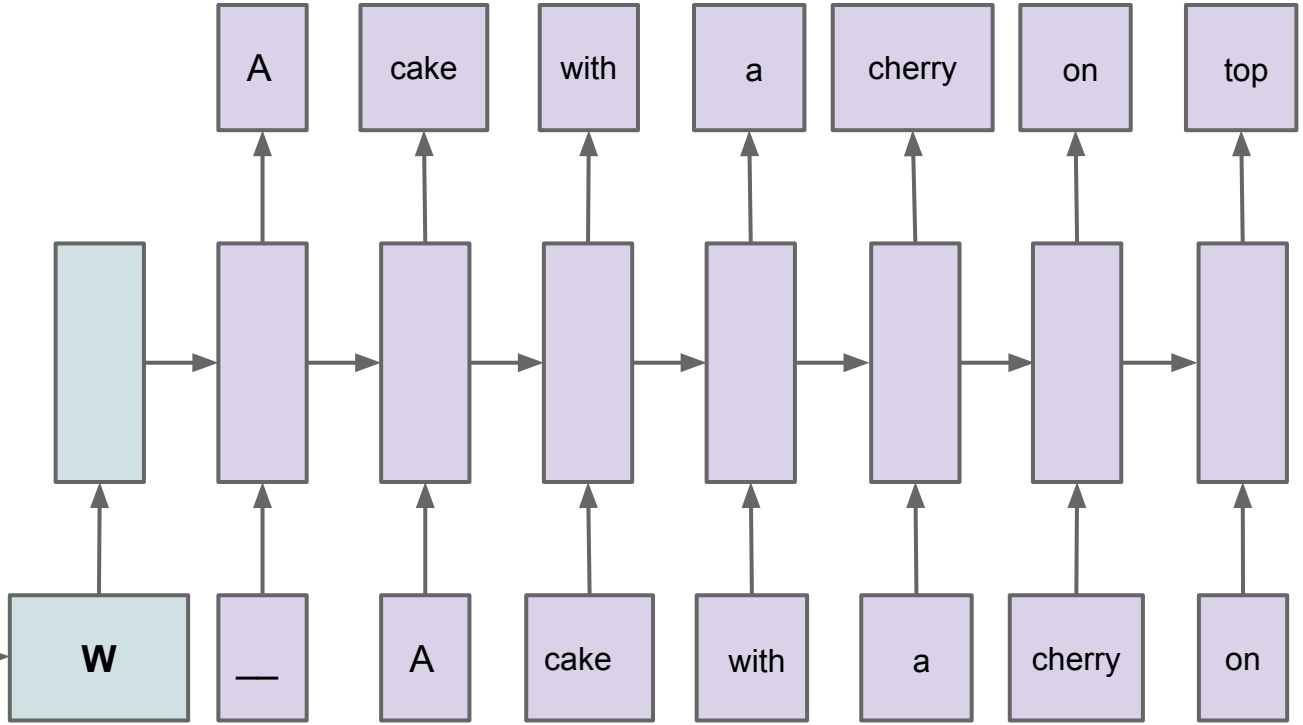
p(English | French)

p(English | Image)

1. Vinyals, O., et al. "Show and Tell: A Neural Image Caption Generator." *CVPR* (2015).
2. Mao, J., et al. "Deep captioning with multimodal recurrent neural networks (m-rnn)." *ICLR (2015).*
3. Karpathy, A., Li, F., "Deep visual-semantic alignments for generating image descriptions." *CVPR (2015)*.

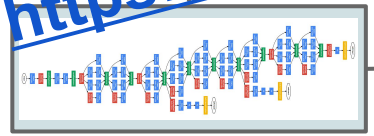
Image Captioning

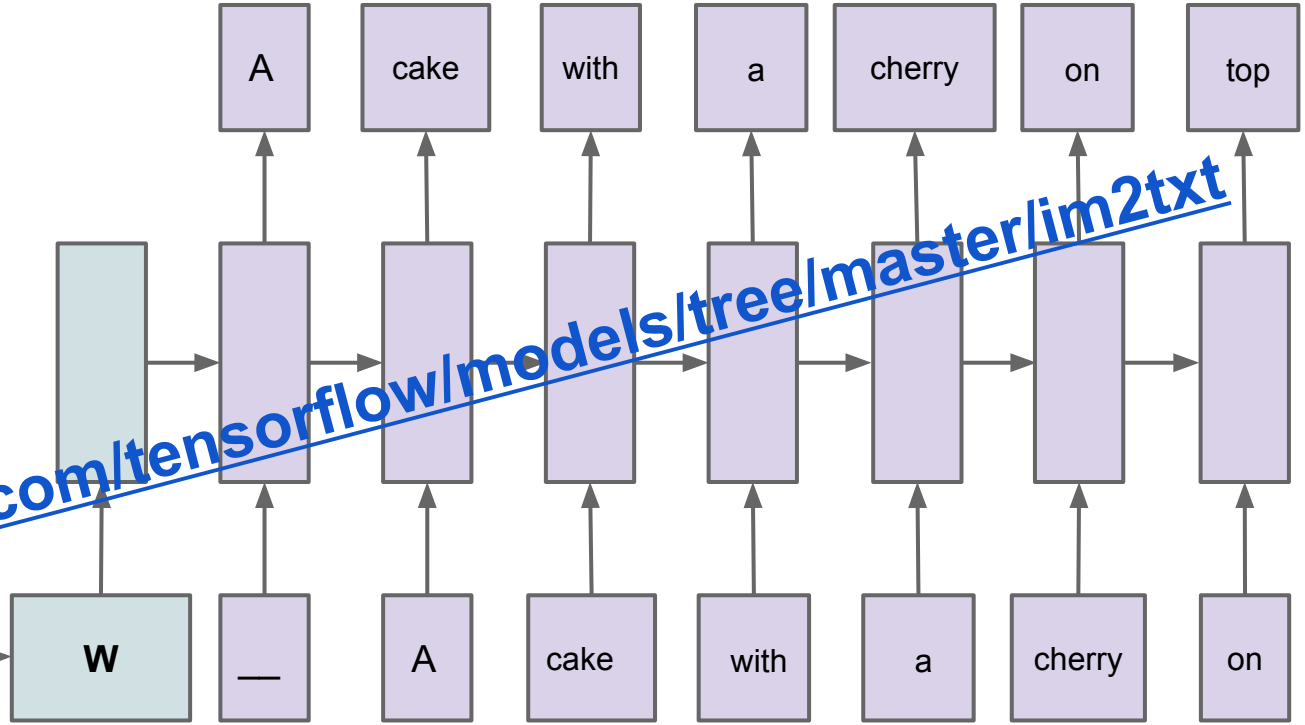




$$\theta^\star = \arg\max_\theta p(S|I)$$

Image Captioning





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

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

Image Captioning

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.

Image Captioning

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.

Image Captioning

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.

Image Captioning

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.

Image Captioning

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.

Learning To Execute [Zaremba & Sutskever, 2014]

- One of the first (modern) examples of learning algorithms
- 2014--??? “era of discovery” → Apply seq2seq to *everything*

Input:

```
j=8584
for x in range(8):
  j+=920
b=(1500+j)
print((b+7567))
```

Target: 25011.

Input:

```
i=8827
c=(i-5347)
print((c+8704) if 2641<8500 else 5308)
```

Target: 12184.

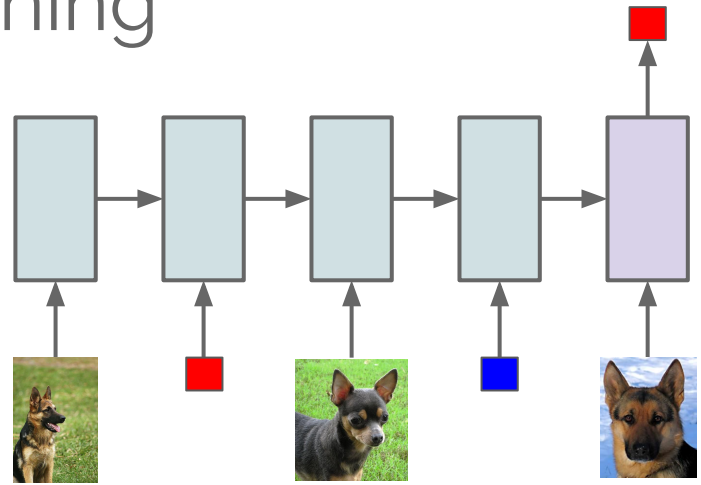
Input:

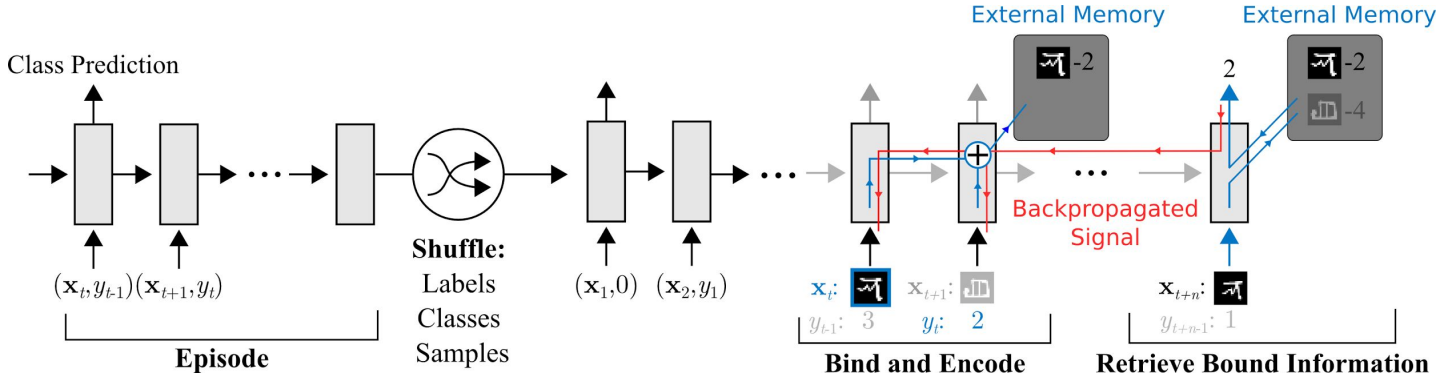
```
vqppkn
sqdvfljmnc
y2vxdddsepnimcbvubkomhrpliibtwztbljipcc
```

Target: hkhpg

1. Zaremba, W., Sutskever, I. “Learning To Execute.” *arxiv (2014)*.

Meta Learning

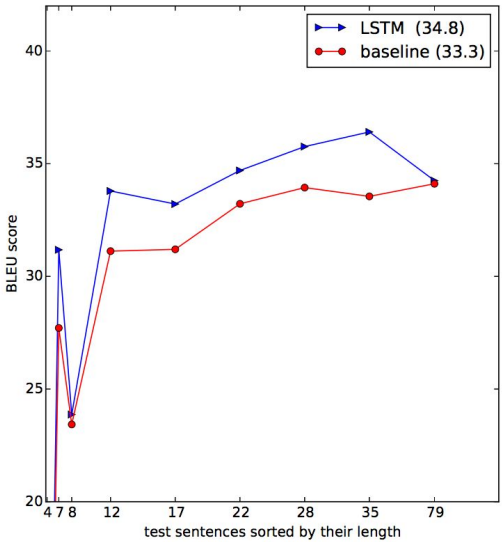


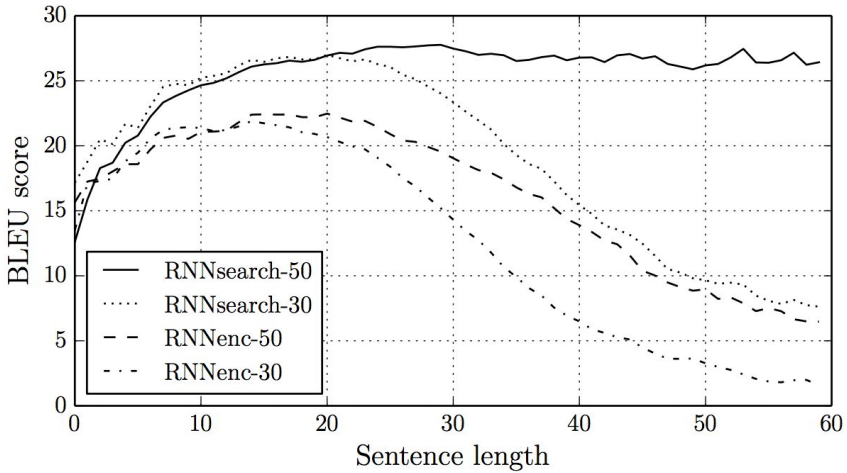


1. Young, A., et al. “Meta-learning with backpropagation.” *Neural Nets (2001).*
2. Santoro, A., et al. “Meta-Learning with Memory-Augmented Neural Networks.” *ICML (2016)*.
3. Vinyals, O., et al. “Matching Networks for One Shot Learning.” *NIPS (2016).*

Seq2Seq - Limitations

- Fixed Size Embeddings are easily overwhelmed by long inputs or long outputs





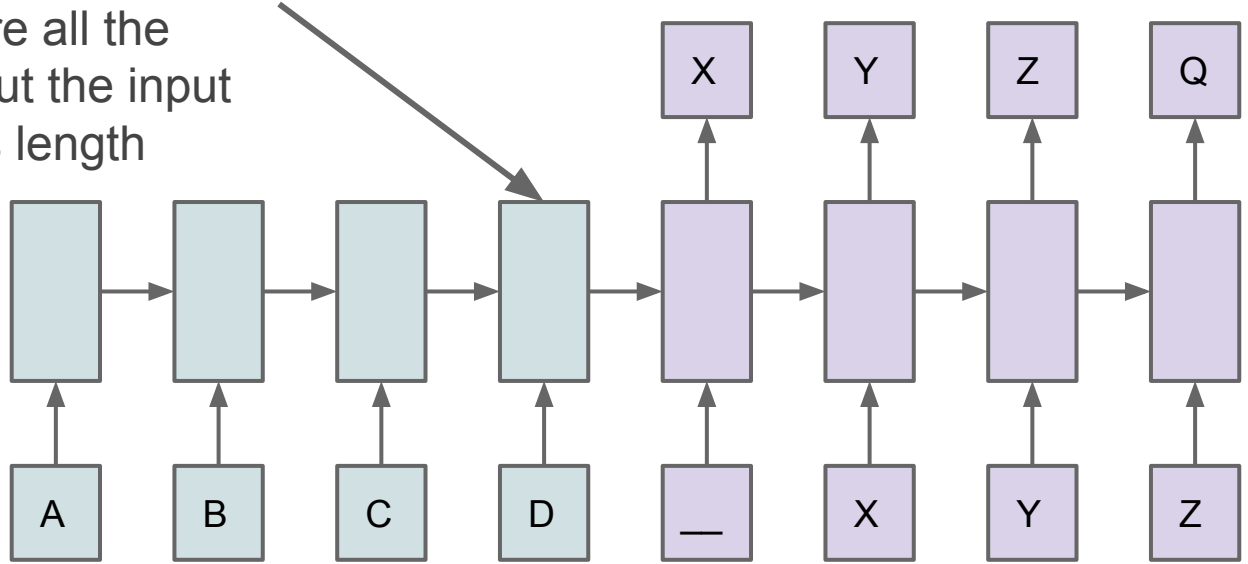
1. Sutskever, I., et al. “Sequence to Sequence Learning with Neural Networks.” *NIPS (2014)*
2. Bahdanau, D., et al. “Neural Machine Translation by Jointly Learning to Align and Translate.” *ICLR (2015)*

Attention

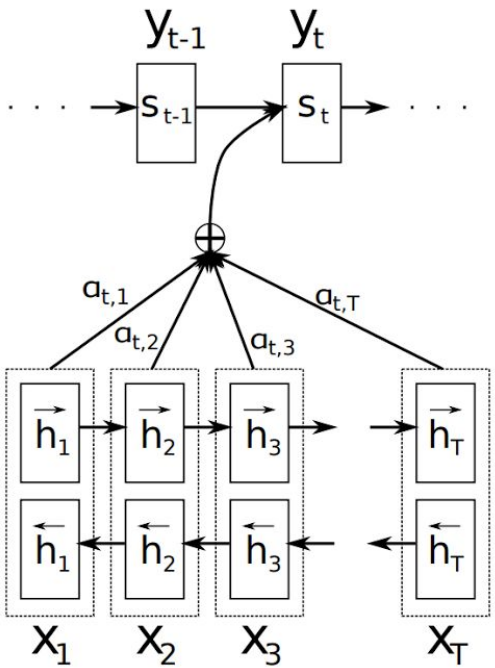
Seq2Seq -- The issue with long inputs

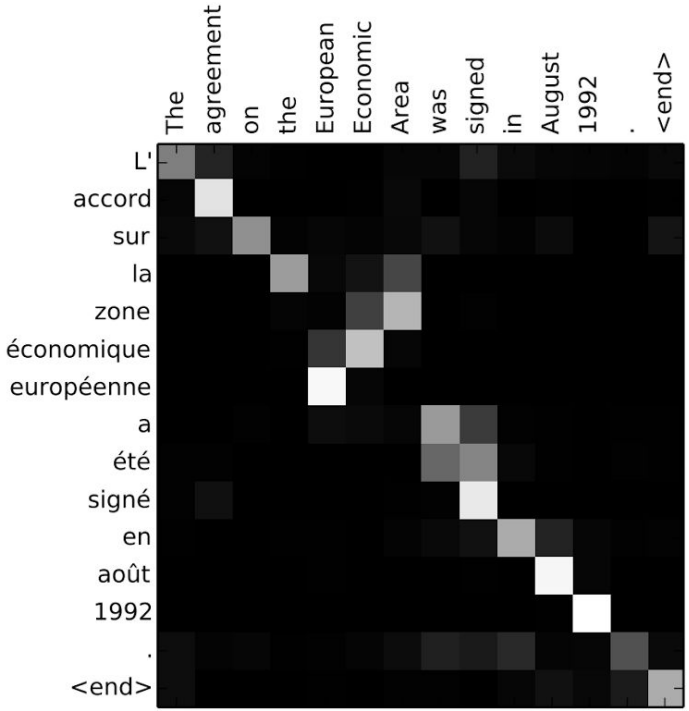
- Same embedding informs the entire output
- Needs to capture all the information about the input regardless of its length

Is there a better way to pass the information from encoder to the decoder ?



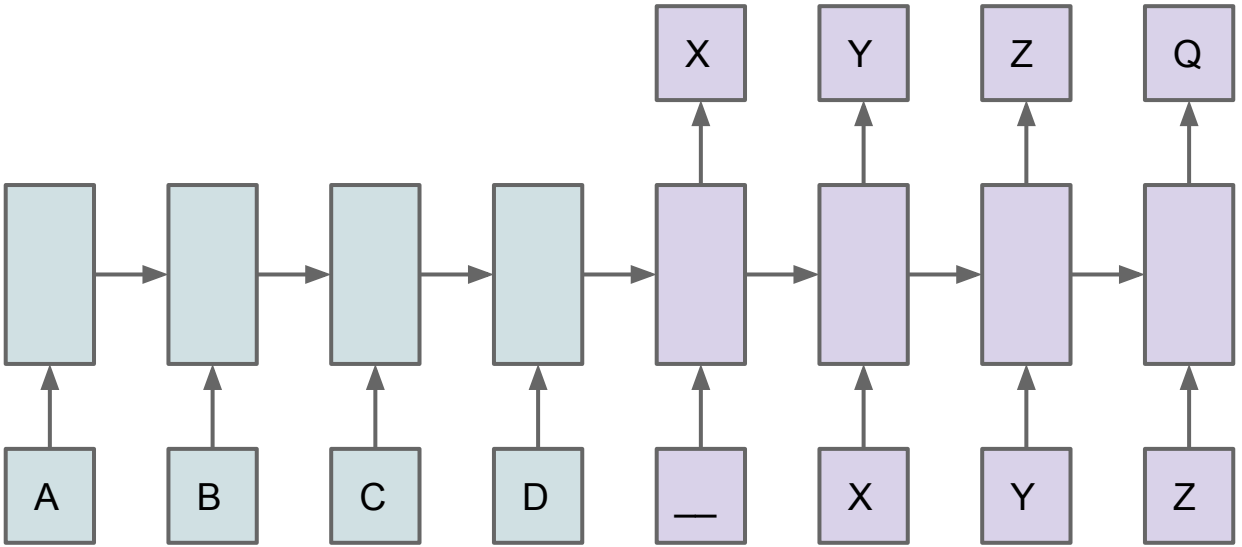
Seq2Seq with Attention





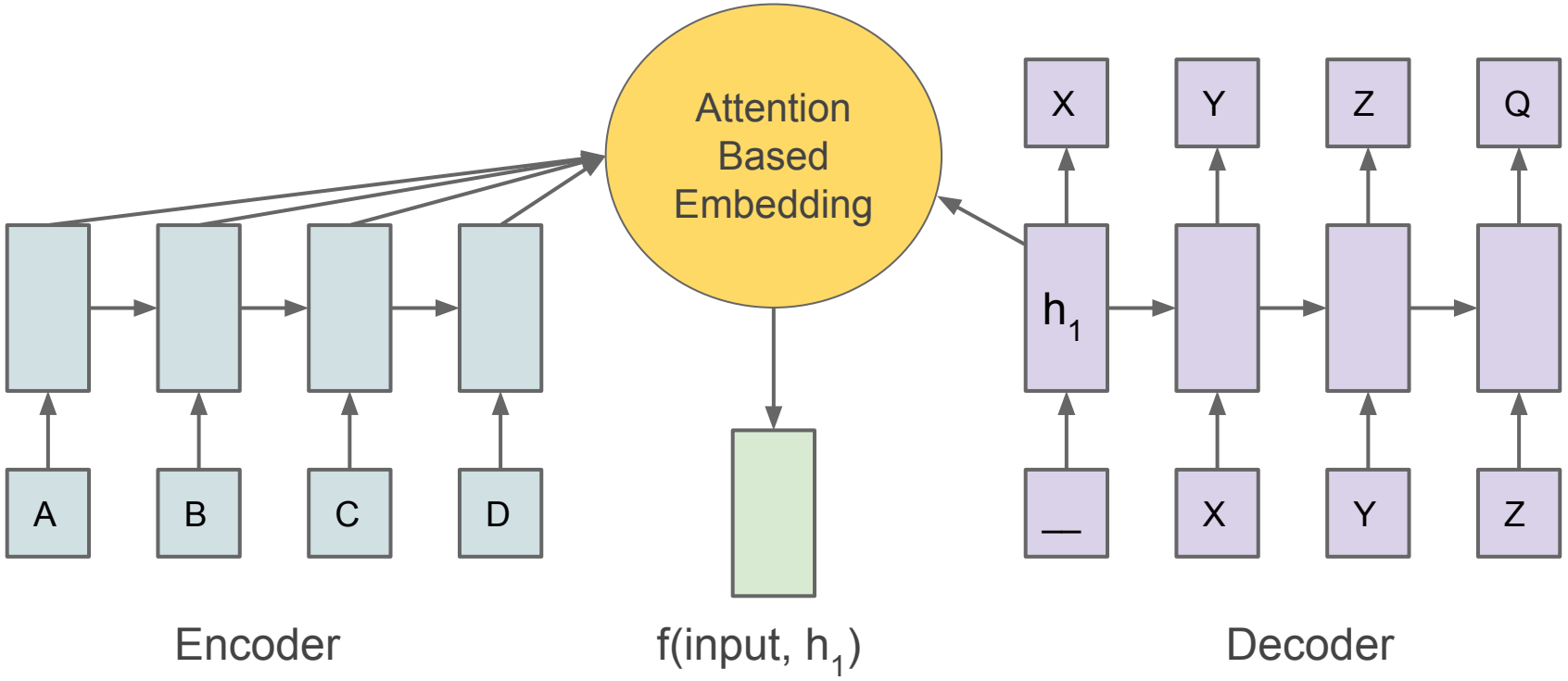
Bahdanau, D., et al. “Neural Machine Translation by Jointly Learning to Align and Translate.” *ICLR (2015)*

Seq2Seq



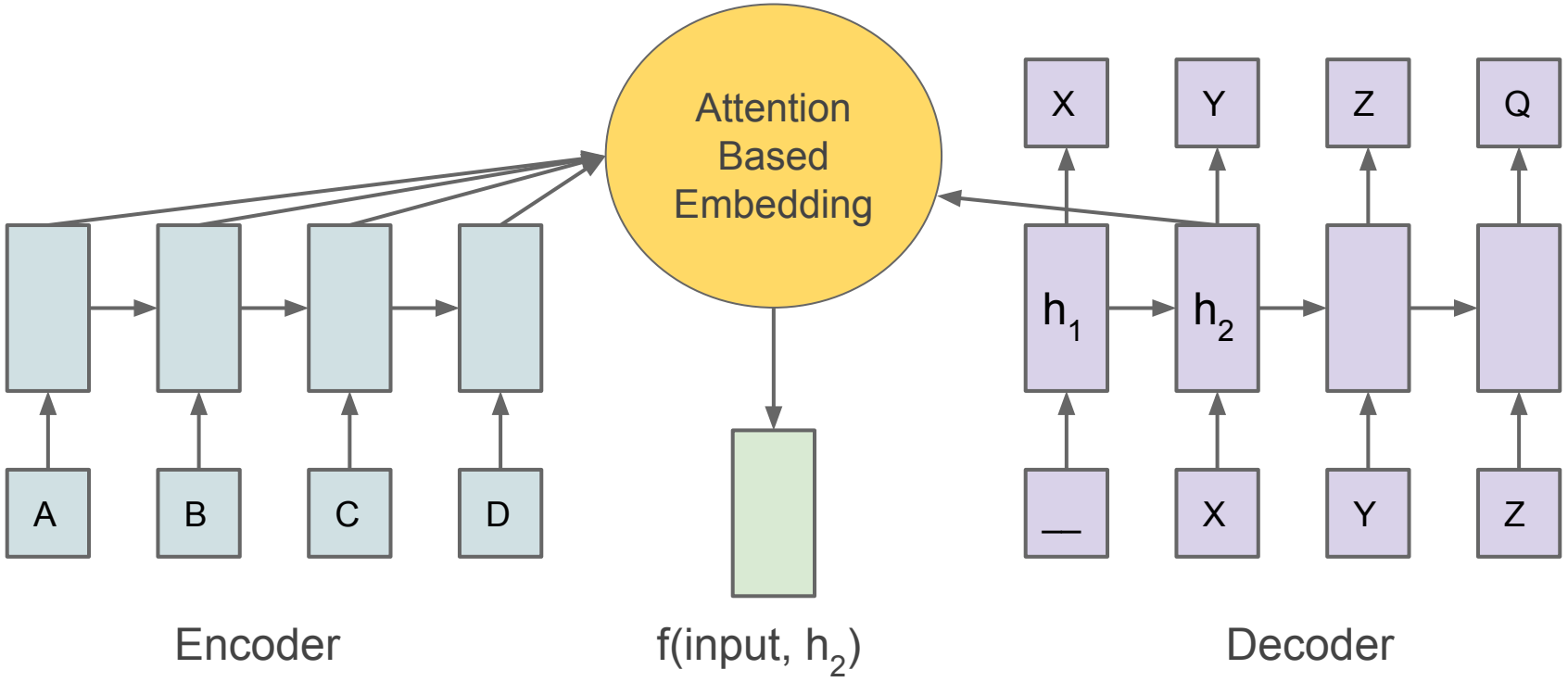
Seq2Seq with Attention

- A different embedding computed for every output step



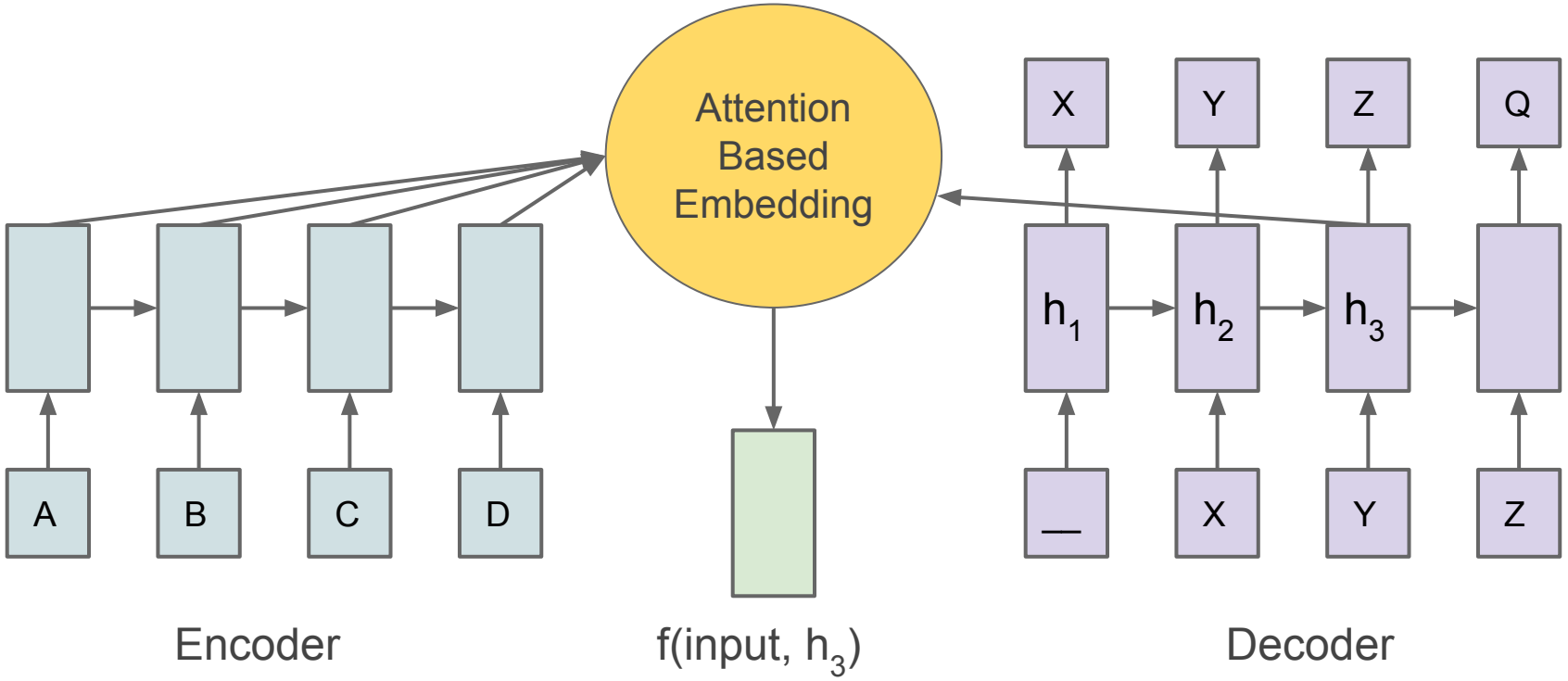
Seq2Seq with Attention

- A different embedding computed for every output step



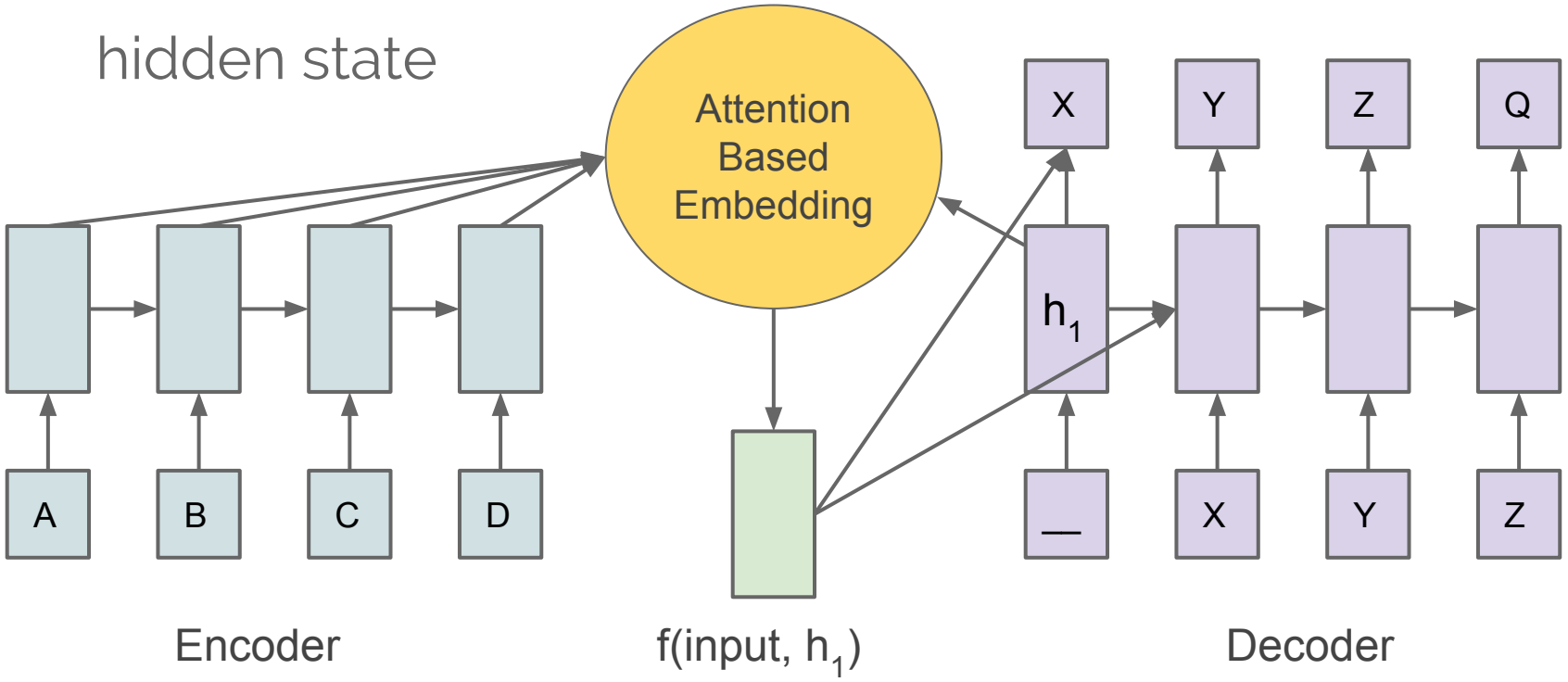
Seq2Seq with Attention

- A different embedding computed for every output step



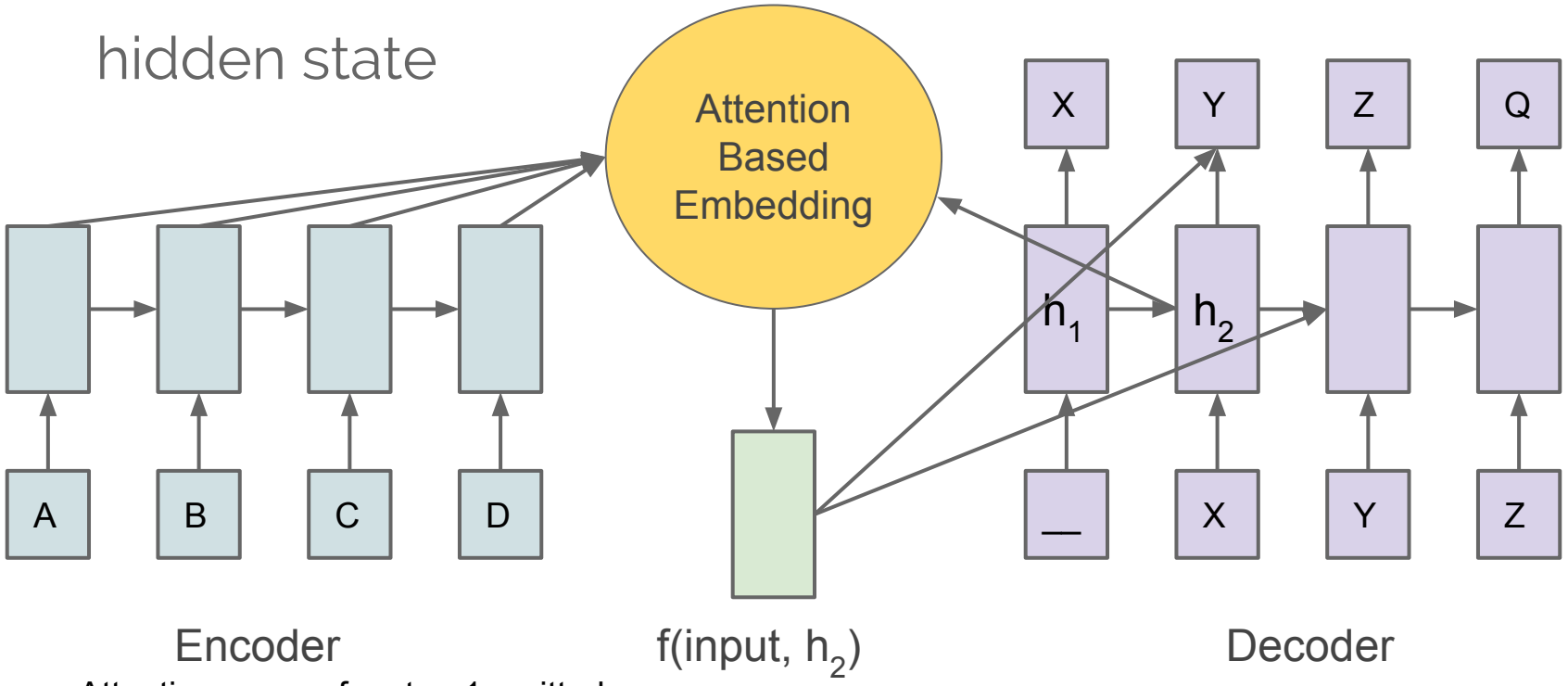
Seq2Seq with Attention

- Embedding used to predict output, and compute next hidden state



Seq2Seq with Attention

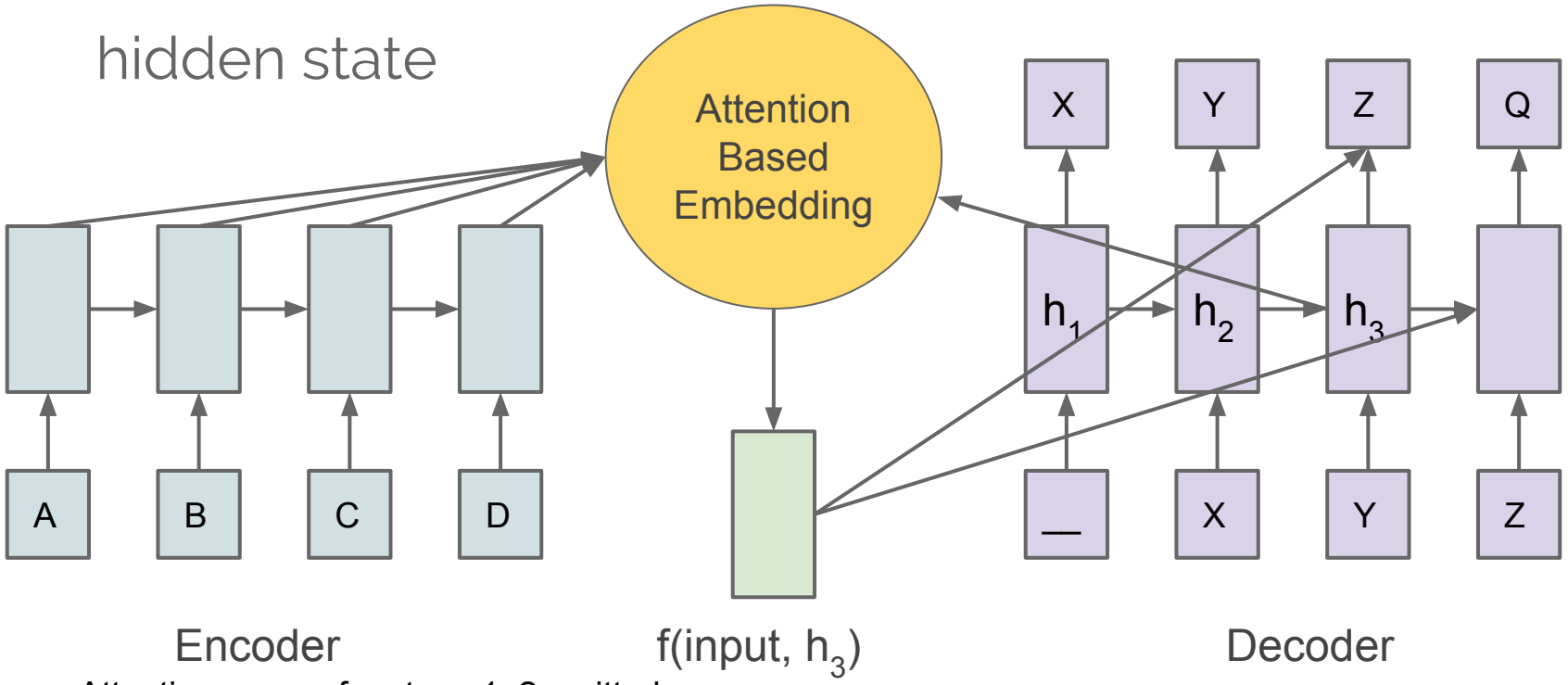
- Embedding used to predict output, and compute next hidden state



- Attention arrows for step 1 omitted

Seq2Seq with Attention

- Embedding used to predict output, and compute next hidden state



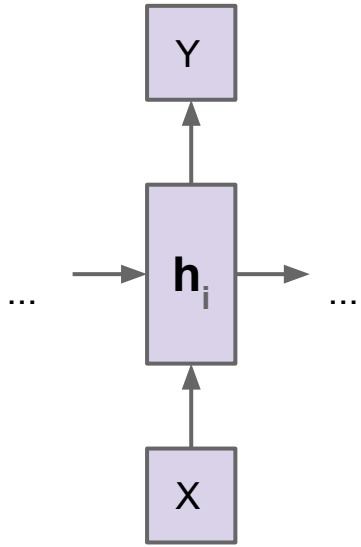
- Attention arrows for steps 1, 2 omitted

Attention Based Embedding

- Linear blending of embedding RNN states $\mathbf{e}_1$ $\mathbf{e}_2$ $\mathbf{e}_3$ $\mathbf{e}_4$ is a natural choice
- How to produce the coefficients (attention vector) for blending ?
  - Content based coefficients based on query state $\mathbf{h}_i$ and embedding RNN states $\mathbf{e}_1$ $\mathbf{e}_2$ $\mathbf{e}_3$ $\mathbf{e}_4$

Dot product Attention

- Inputs: “I am a cat.”
- Input RNN states: $\mathbf{e}_1$ $\mathbf{e}_2$ $\mathbf{e}_3$ $\mathbf{e}_4$
- Decoder RNN state at step i (query): $\mathbf{h}_i$
- Compute scalars $\mathbf{h}_i^T\mathbf{e}_1, \mathbf{h}_i^T\mathbf{e}_2, \mathbf{h}_i^T\mathbf{e}_3, \mathbf{h}_i^T\mathbf{e}_4$ representing similarity / relevance between encoder steps and query.
- Normalize $[\mathbf{h}_i^T\mathbf{e}_1, \mathbf{h}_i^T\mathbf{e}_2, \mathbf{h}_i^T\mathbf{e}_3, \mathbf{h}_i^T\mathbf{e}_4]$ with softmax to produce attention weights, e.g. [0.0 0.05 0.9 0.05]



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 [1] & Memory Networks [2]:
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

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

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

1. Graves, A., et al. “Neural Turing Machines.” *arxiv (2014)*
2. Weston, J., et al. “Memory Networks.” *arxiv (2014)*

Other strategies for attention models

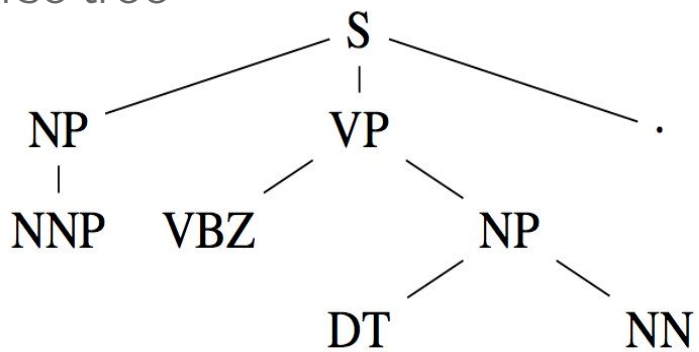
- Tensored attention
  - Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. “Effective Approaches to Attention-based Neural Machine Translation.” EMNLP’15.
- Multiple heads
- Pyramidal encoders
  - William Chan, Navdeep Jaitly, Quoc Le, Oriol Vinyals. “Listen Attend and Spell”. ICASSP 2015.
- Hierarchical Attention
  - Andrychowicz, Marcin, and Karol Kurach. "Learning efficient algorithms with hierarchical attentive memory." *arXiv preprint arXiv:1602.03218* (2016).
- Hard Attention
  - Xu, Kelvin, et al. “Show, attend and tell: Neural image caption generation with visual attention.” ICML 2015

Applications

Sentence to Constituency Parse Tree

1. Read a sentence
2. Flatten the tree into a sequence (adding (,))
3. “Translate” from sentence to parse tree

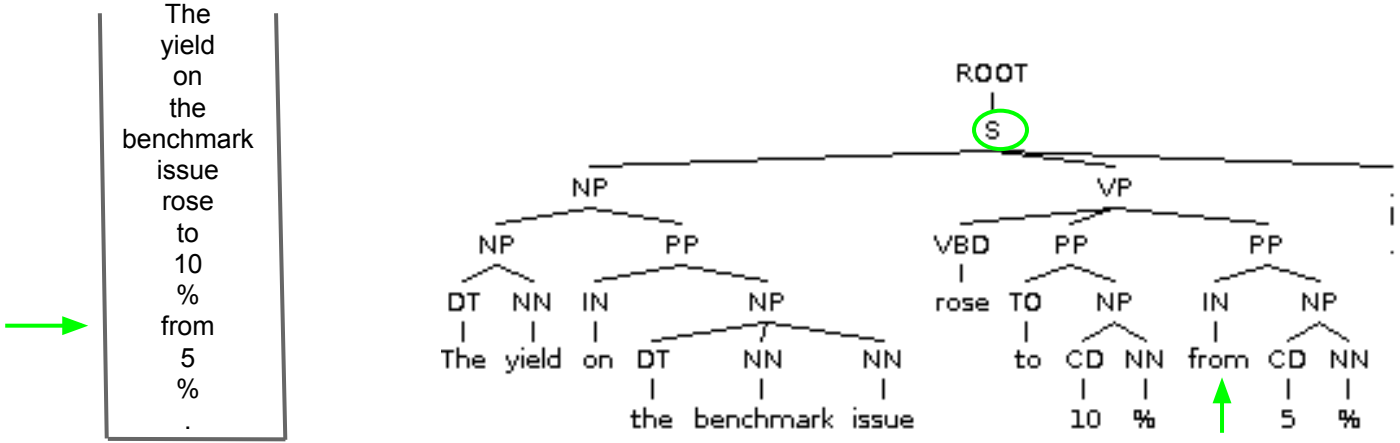
John has a dog . →



John has a dog . → (S (NP NNP)$_{NP}$ (VP VBZ (NP DT NN)$_{NP}$)$_{VP}$.)$_{S}$

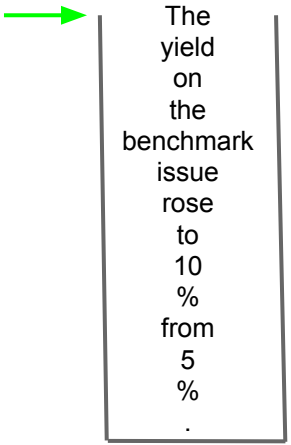
Vinyals, O., et al. “Grammar as a foreign language.” *NIPS (2015).*

Parsing Example

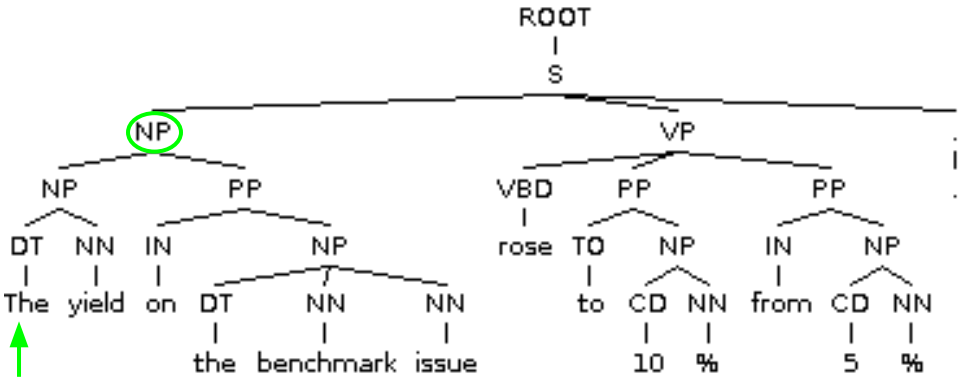


Vinyals, O., et al. “Grammar as a foreign language.” *NIPS (2015).*

Parsing Example

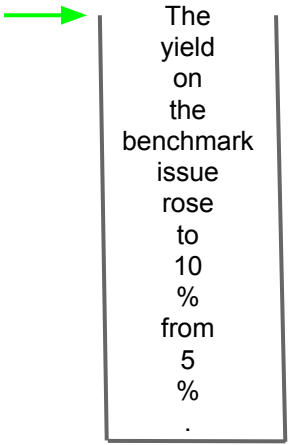


Encoder LSTM
(reverse input)

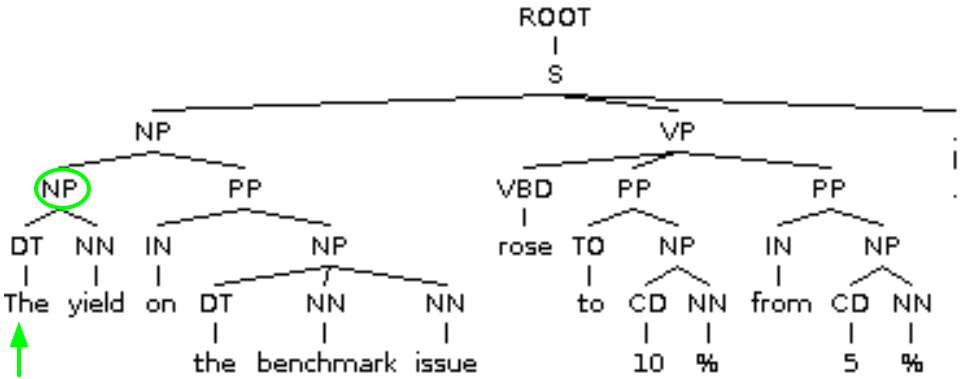


Vinyals, O., et al. “Grammar as a foreign language.” *NIPS (2015).*

Parsing Example

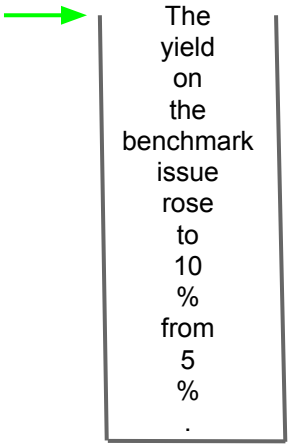


Encoder LSTM
(reverse input)

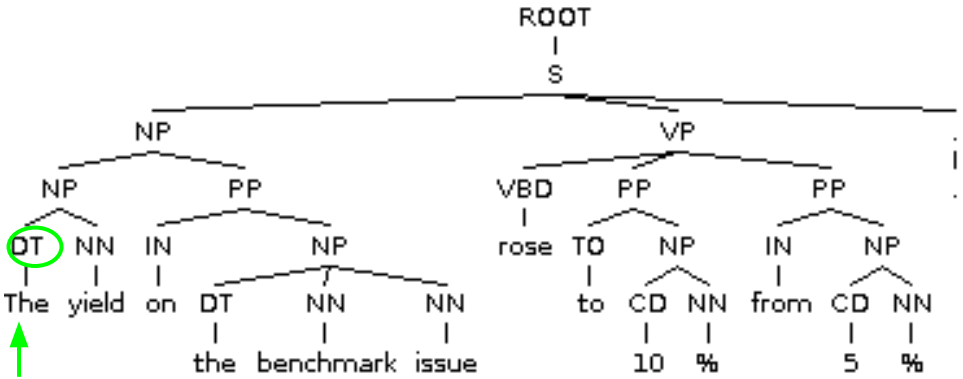


Vinyals, O., et al. “Grammar as a foreign language.” *NIPS (2015).*

Parsing Example

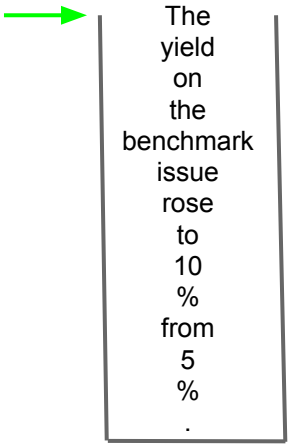


Encoder LSTM
(reverse input)

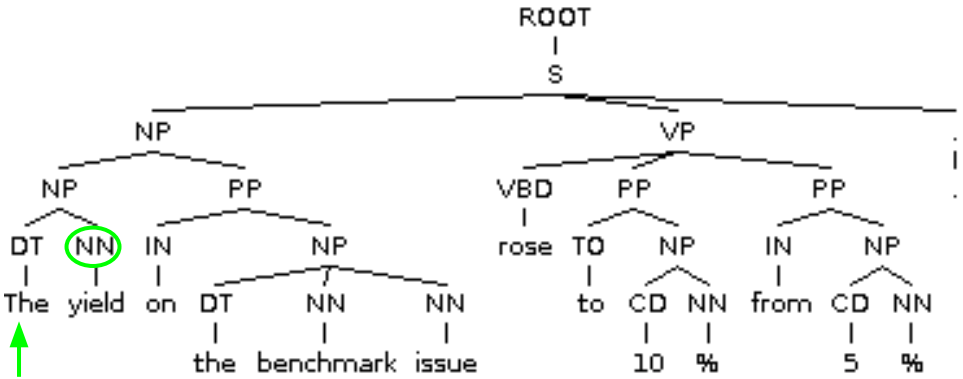


Vinyals, O., et al. “Grammar as a foreign language.” *NIPS (2015).*

Parsing Example

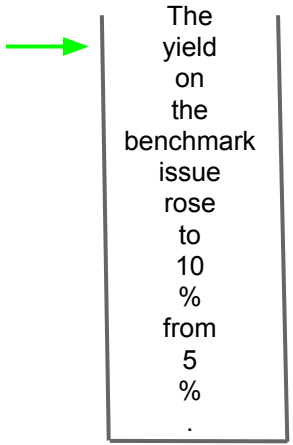


Encoder LSTM
(reverse input)

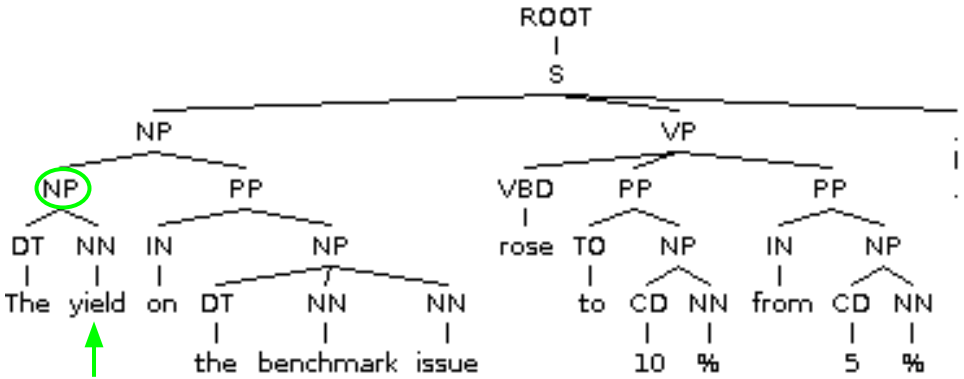


Vinyals, O., et al. “Grammar as a foreign language.” *NIPS (2015).*

Parsing Example

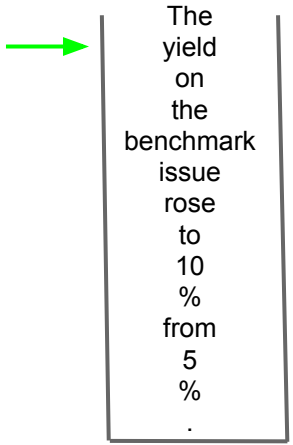


Encoder LSTM
(reverse input)

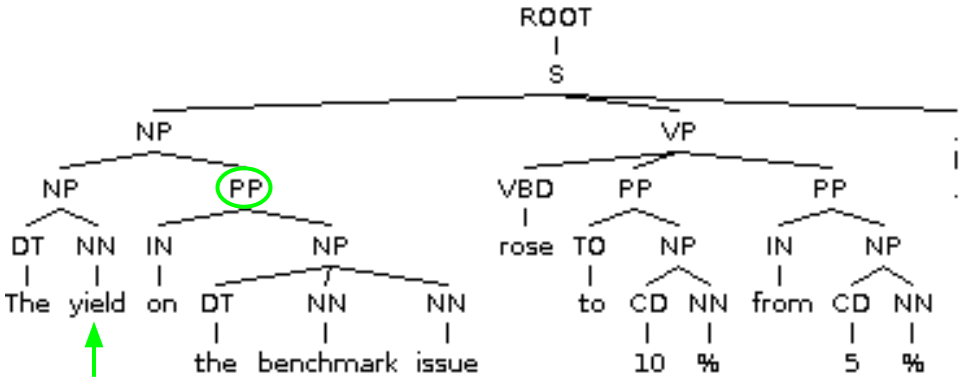


Vinyals, O., et al. “Grammar as a foreign language.” *NIPS (2015).*

Parsing Example

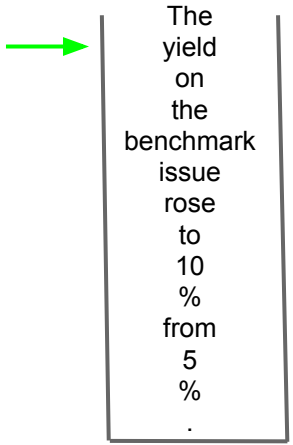


Encoder LSTM
(reverse input)

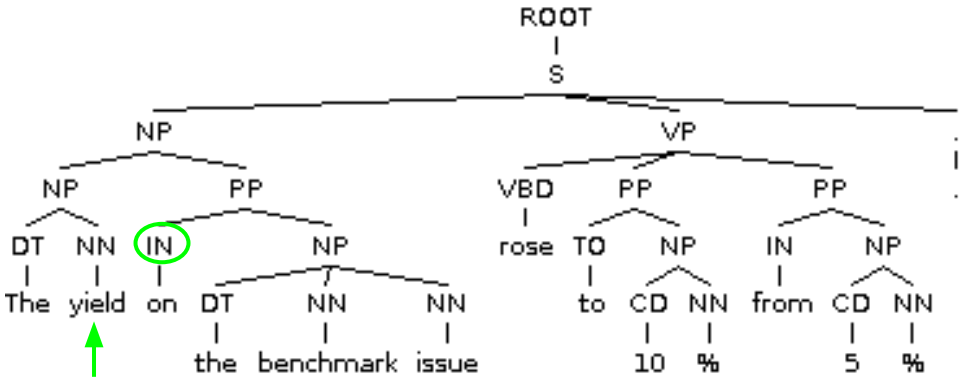


Vinyals, O., et al. “Grammar as a foreign language.” *NIPS (2015).*

Parsing Example

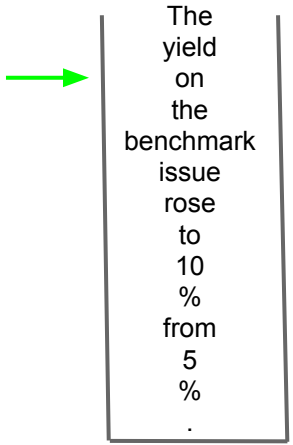


Encoder LSTM
(reverse input)

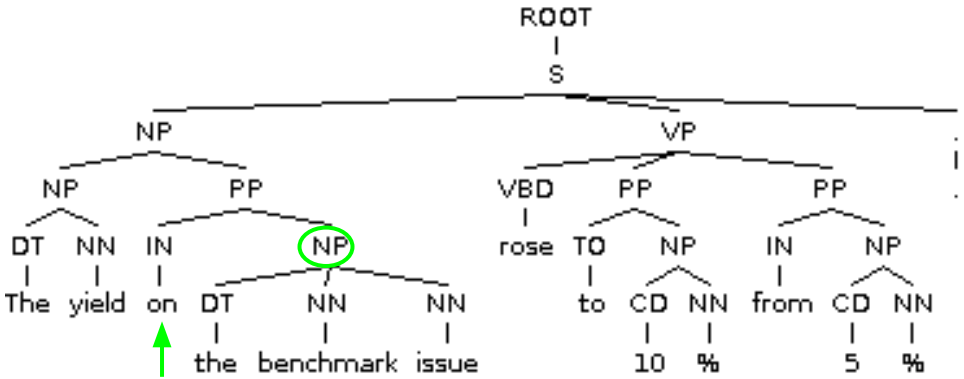


Vinyals, O., et al. “Grammar as a foreign language.” *NIPS (2015).*

Parsing Example

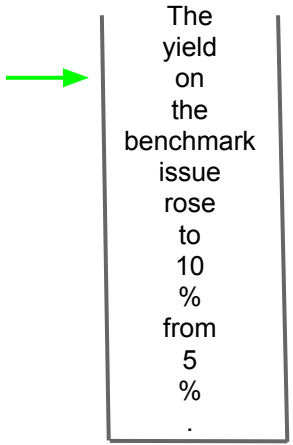


Encoder LSTM
(reverse input)

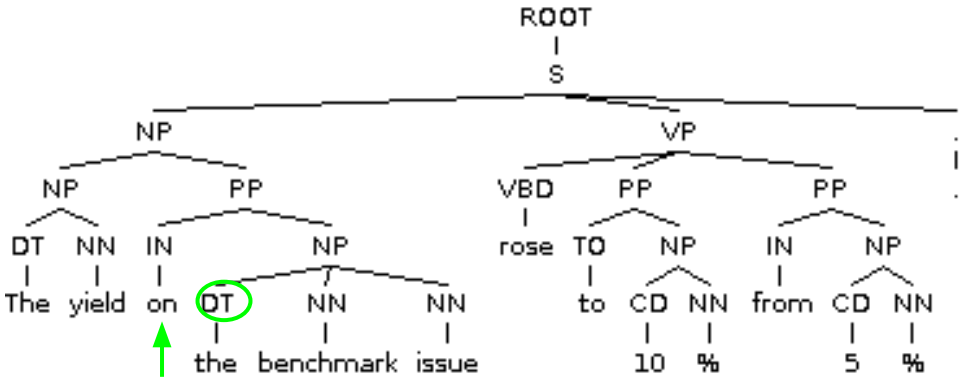


Vinyals, O., et al. “Grammar as a foreign language.” *NIPS (2015).*

Parsing Example

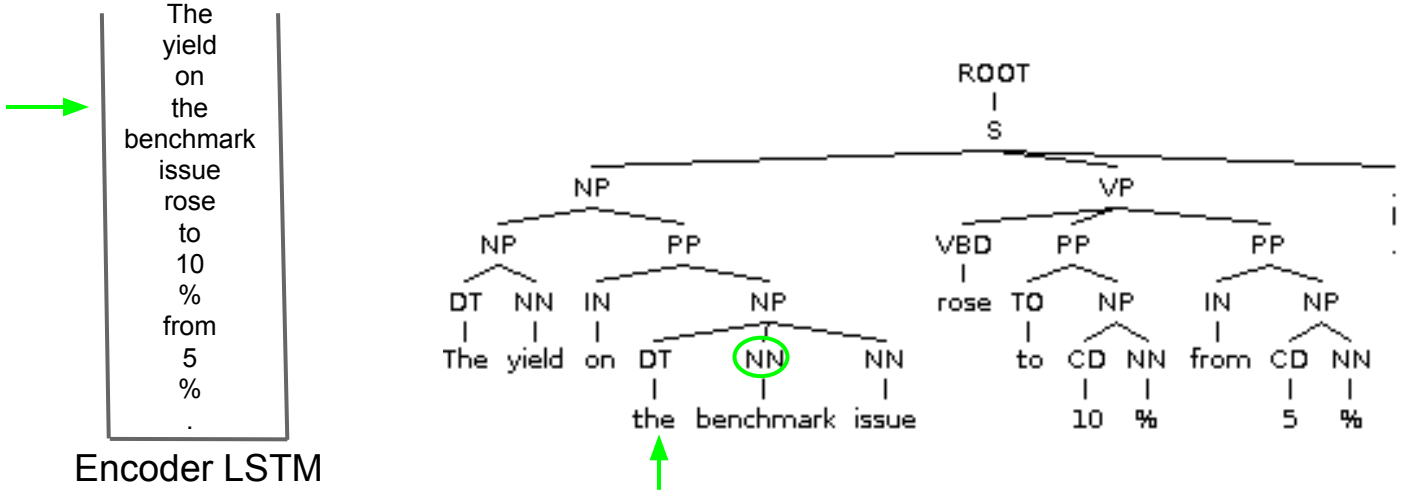


Encoder LSTM
(reverse input)



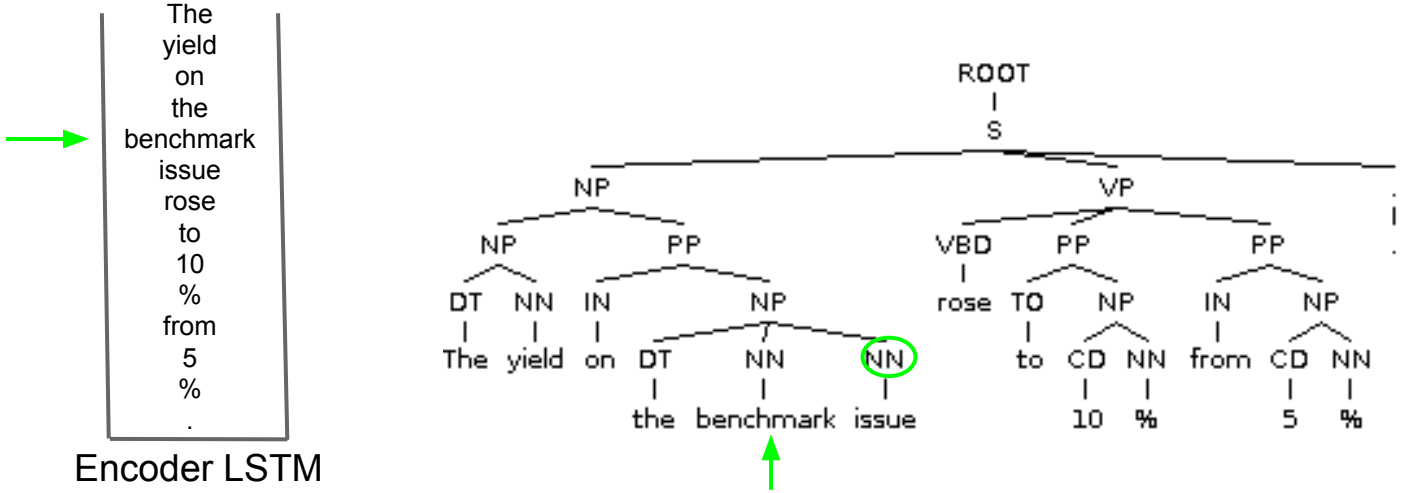
Vinyals, O., et al. “Grammar as a foreign language.” *NIPS (2015).*

Parsing Example



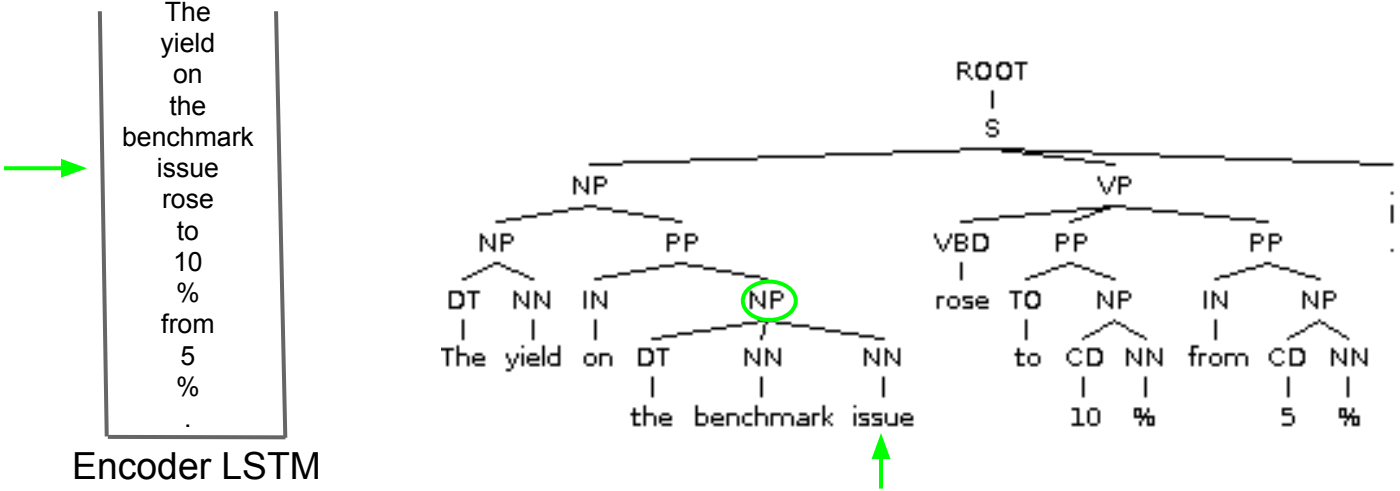
Vinyals, O., et al. “Grammar as a foreign language.” *NIPS (2015).*

Parsing Example



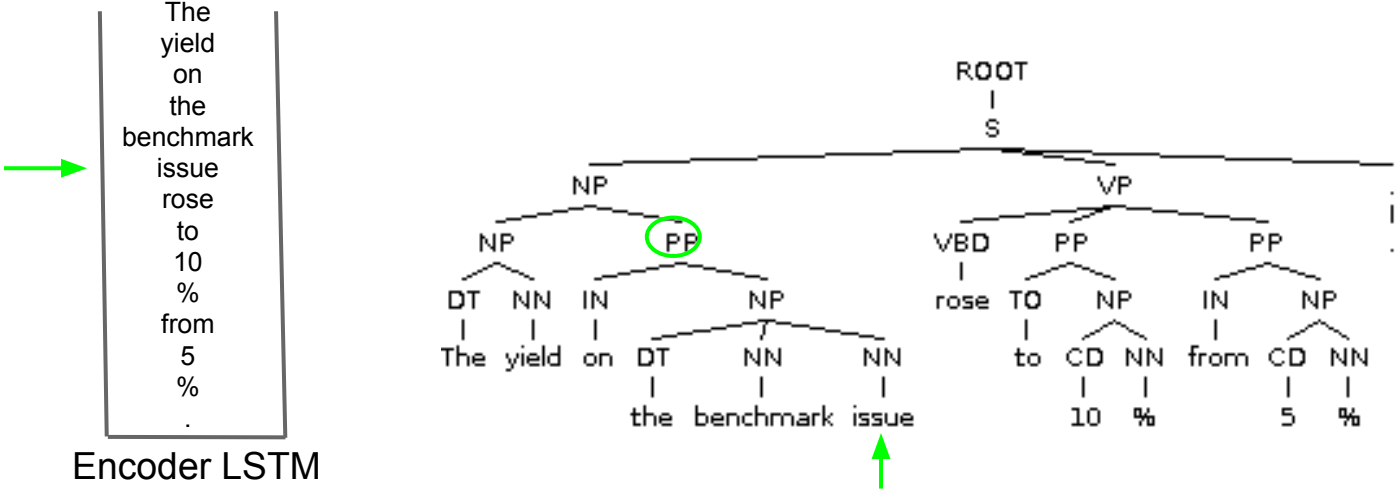
Vinyals, O., et al. “Grammar as a foreign language.” *NIPS (2015).*

Parsing Example



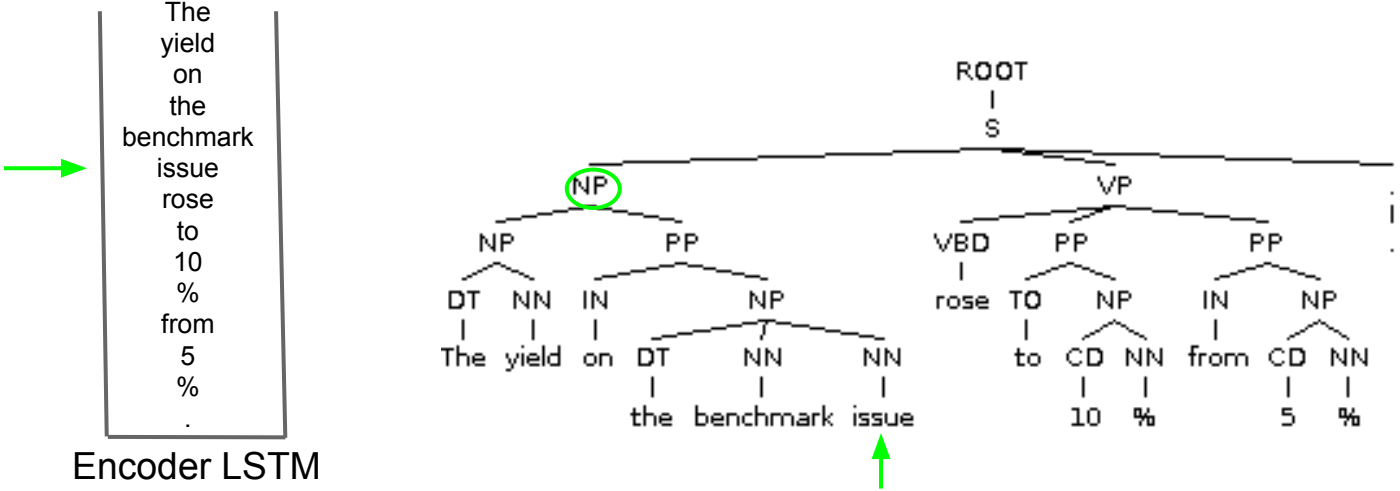
Vinyals, O., et al. “Grammar as a foreign language.” *NIPS (2015).*

Parsing Example



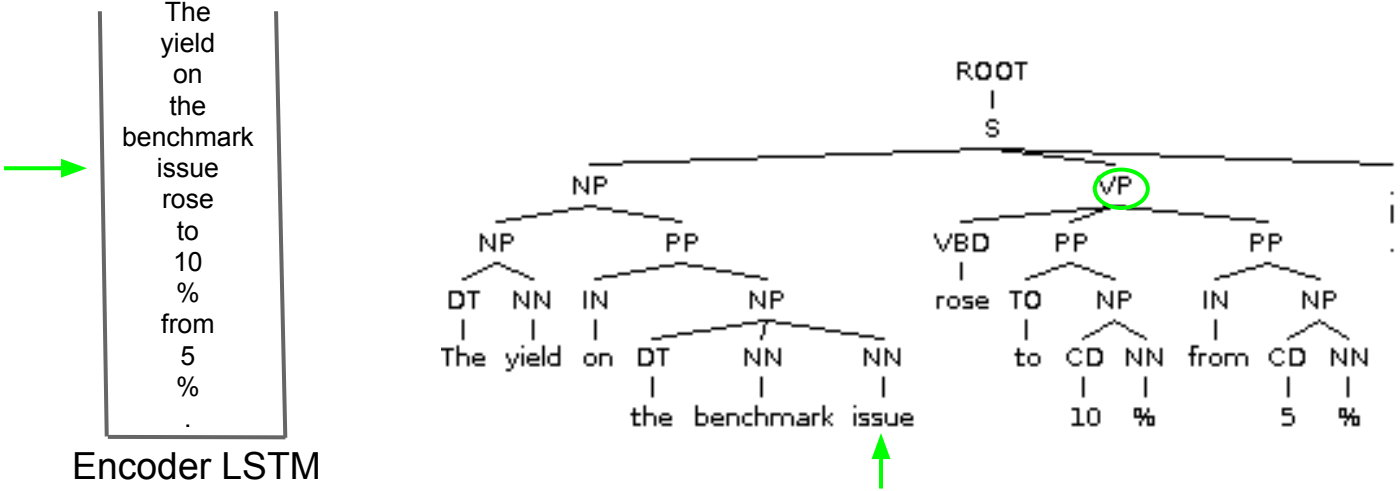
Vinyals, O., et al. “Grammar as a foreign language.” *NIPS (2015).*

Parsing Example



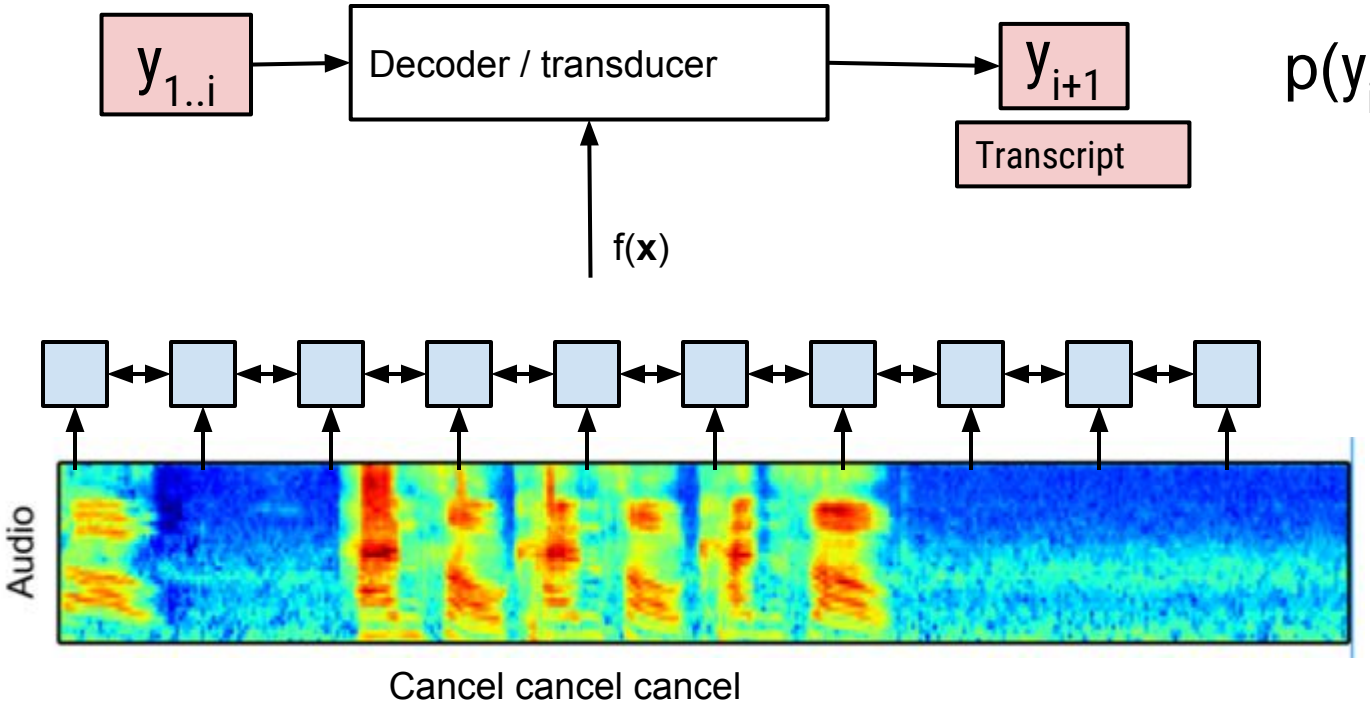
Vinyals, O., et al. “Grammar as a foreign language.” *NIPS (2015).*

Parsing Example



Vinyals, O., et al. “Grammar as a foreign language.” *NIPS (2015).*

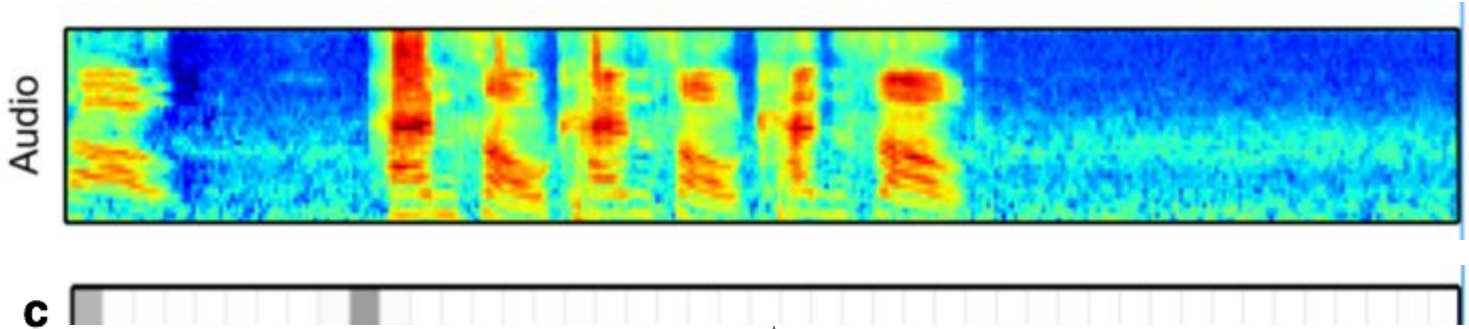
Speech Recognition



$p(y_{i+1}|y_{1..i}, x)$

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

Attention Example



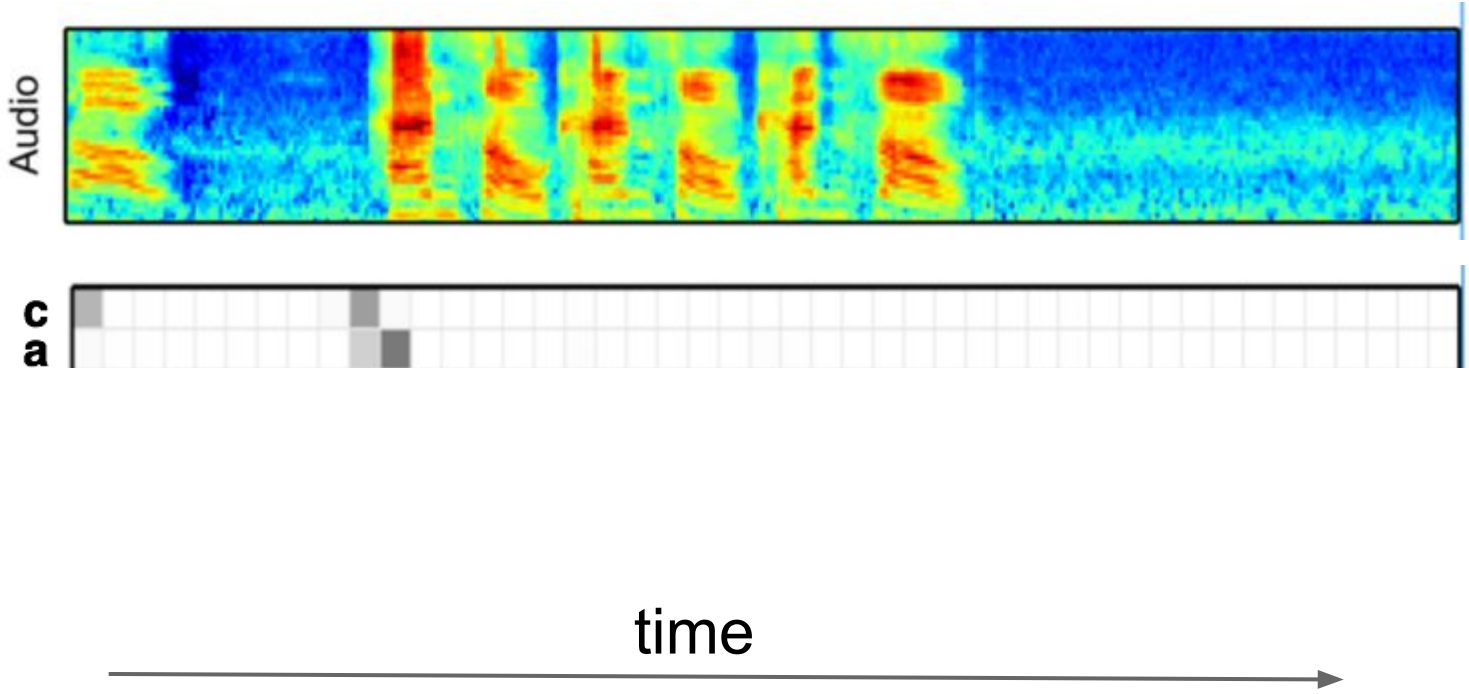
prediction derived from "attending" to segment of input

Attention vector - where the model thinks the relevant information is to be found

time

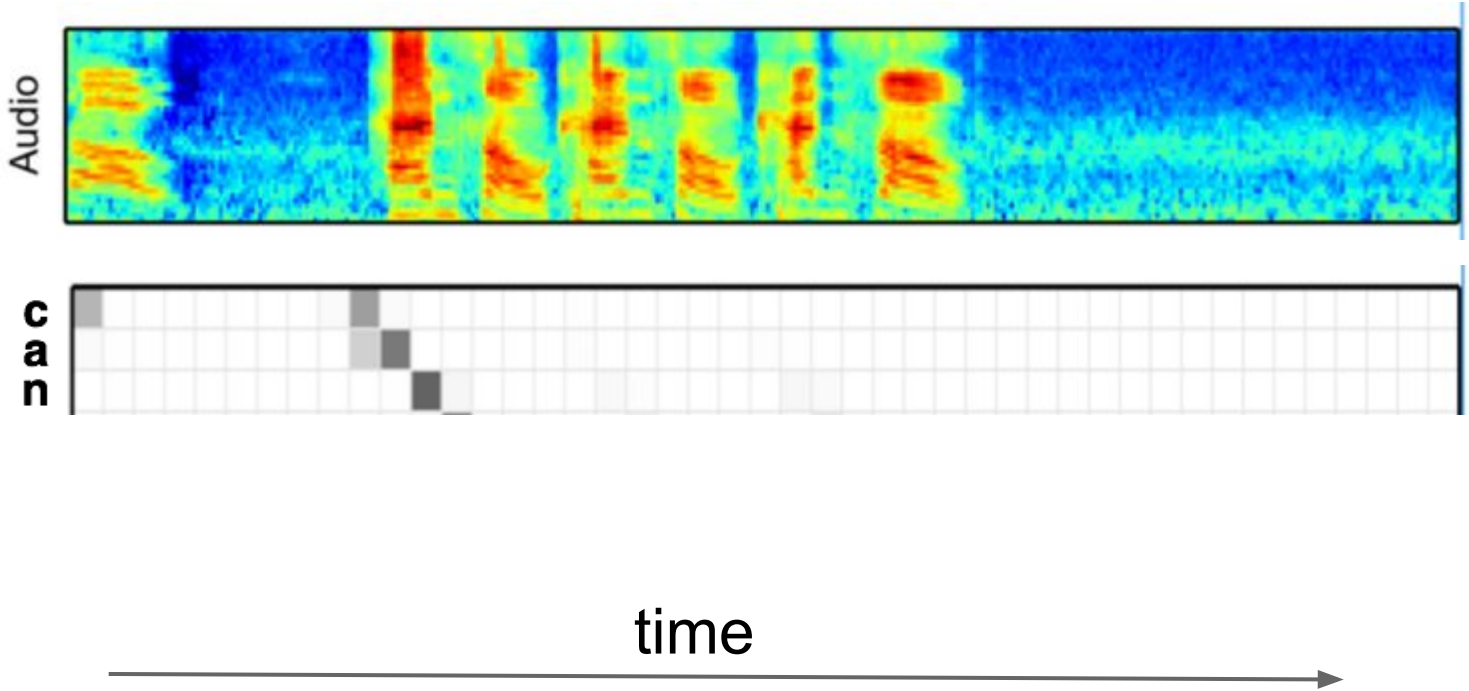
Chan, W., Jaitly, N., Le, Q., Vinyals, O. "*Listen Attend and Spell*." *ICASSP (2015).*

Attention Example



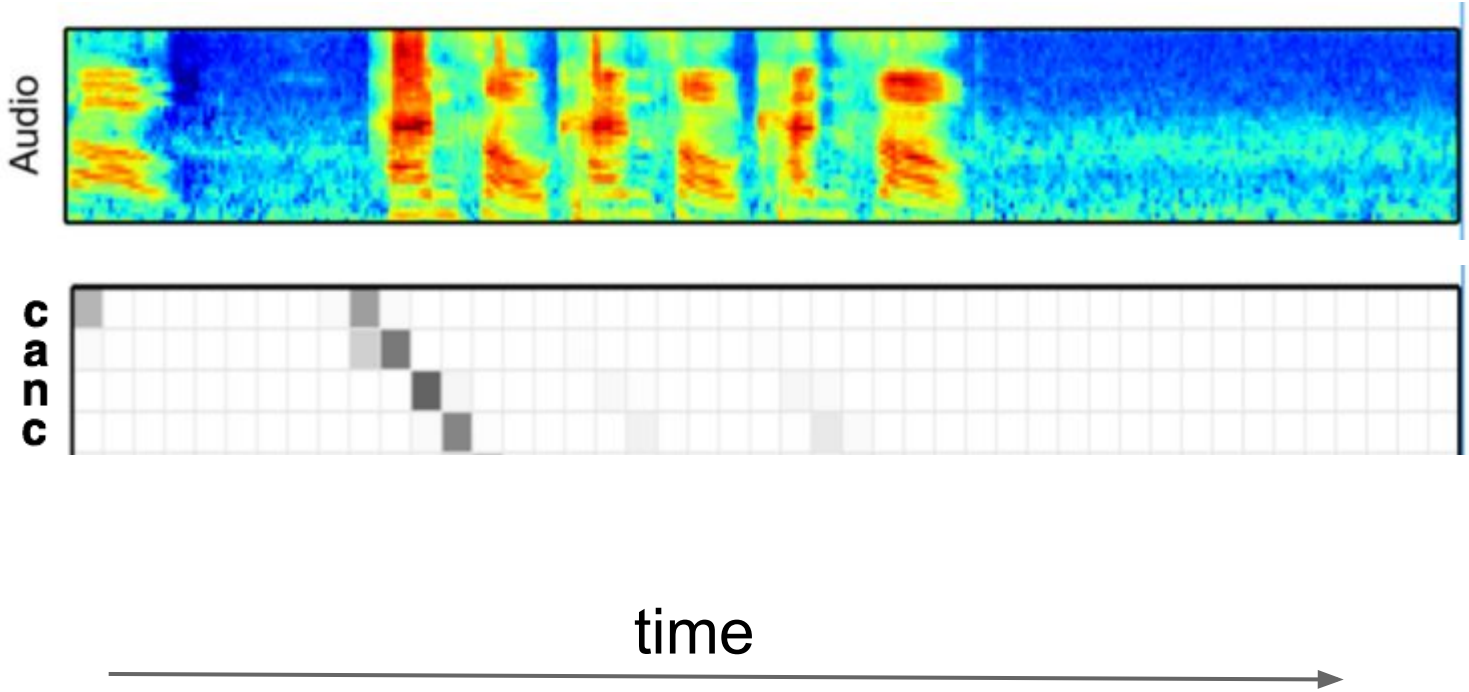
Chan, W., Jaitly, N., Le, Q., Vinyals, O. "*Listen Attend and Spell*." *ICASSP (2015).*

Attention Example



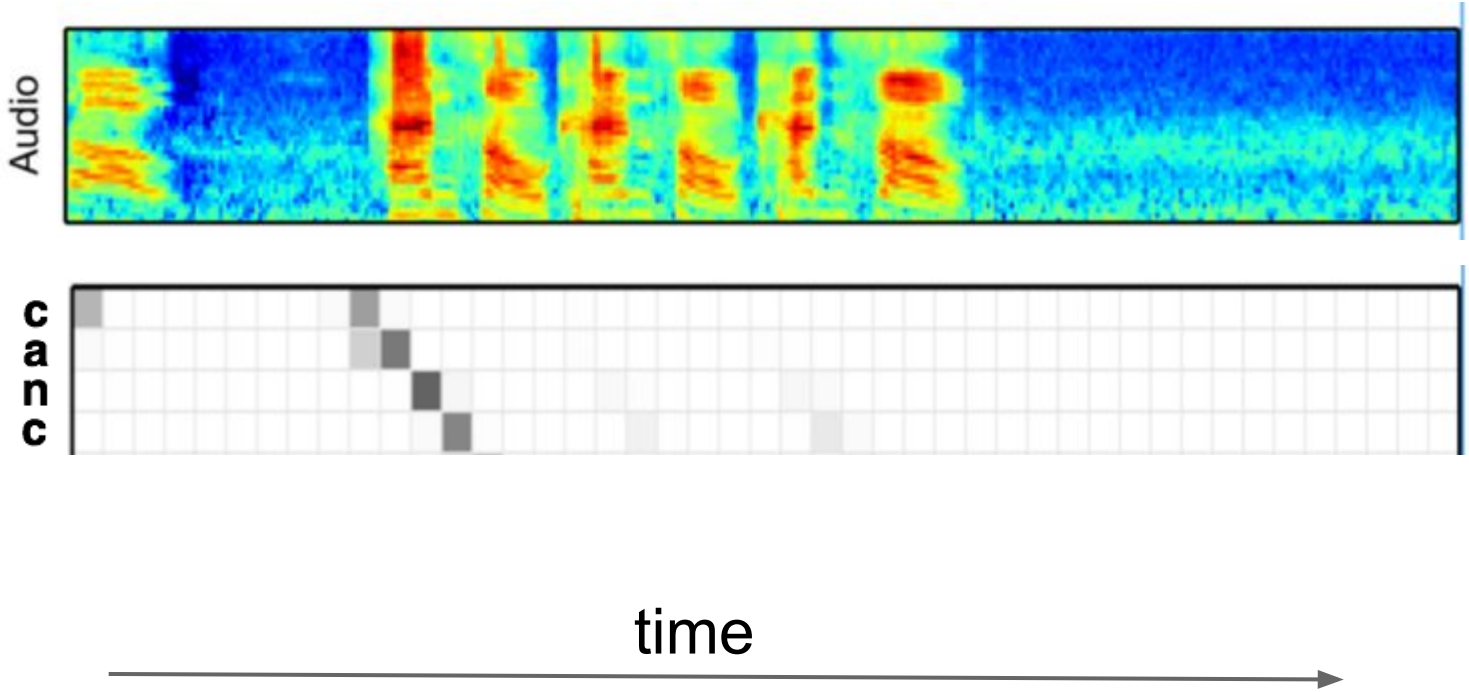
Chan, W., Jaitly, N., Le, Q., Vinyals, O. "*Listen Attend and Spell*." *ICASSP (2015).*

Attention Example



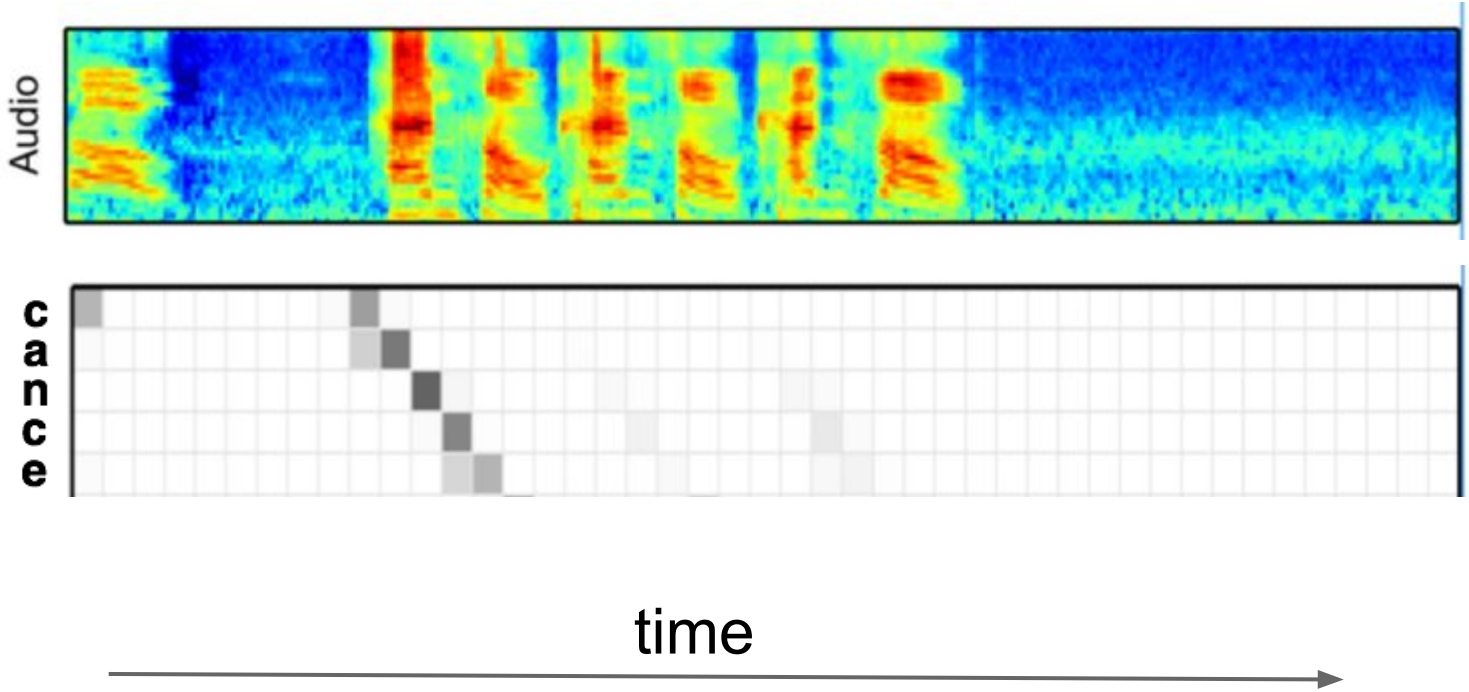
Chan, W., Jaitly, N., Le, Q., Vinyals, O. "*Listen Attend and Spell*." *ICASSP (2015).*

Attention Example



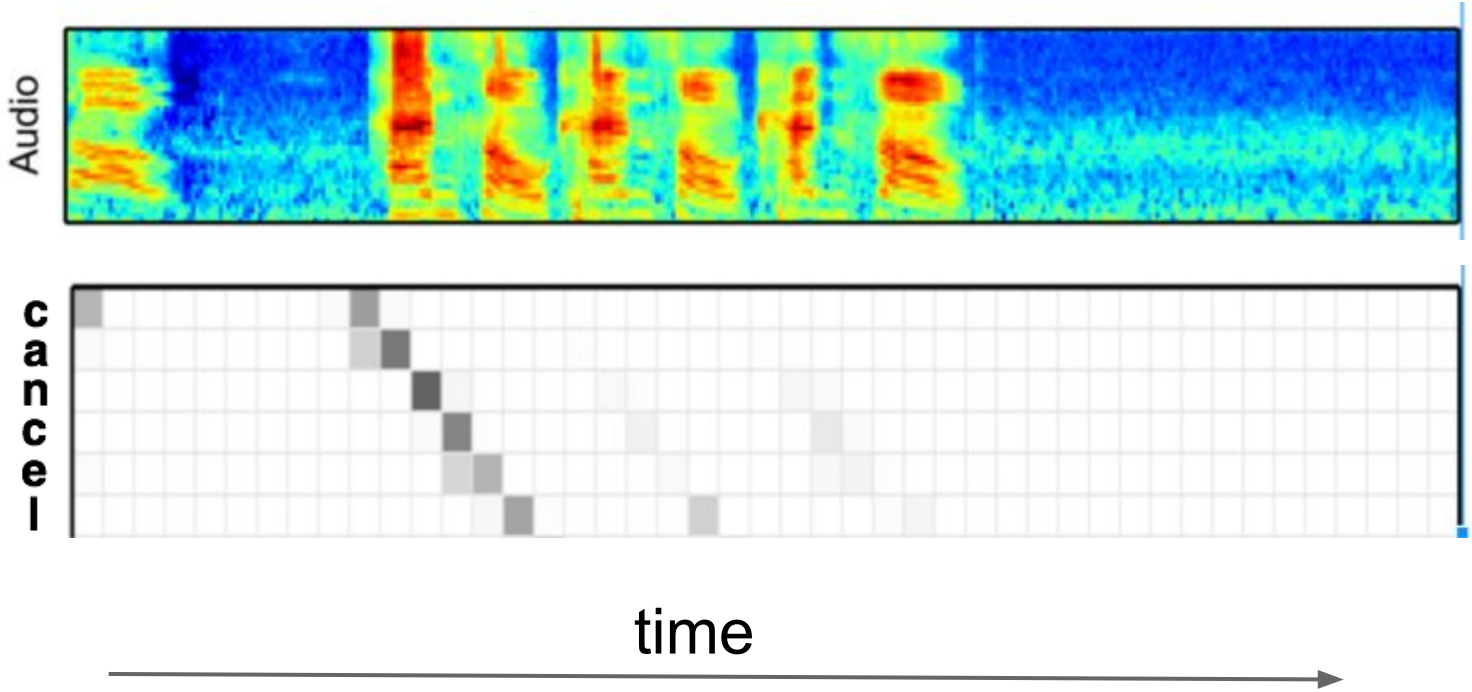
Chan, W., Jaitly, N., Le, Q., Vinyals, O. "*Listen Attend and Spell*." *ICASSP (2015).*

Attention Example



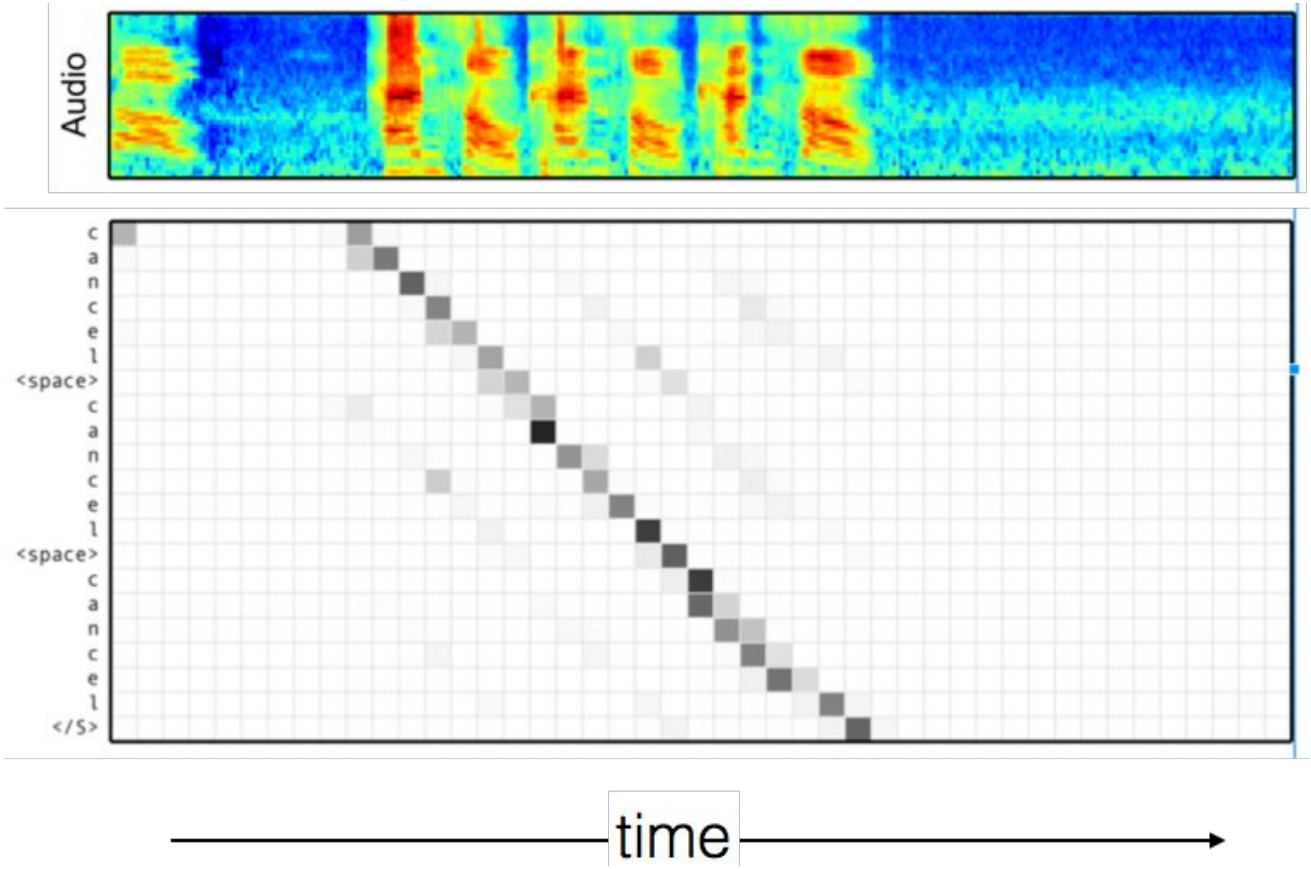
Chan, W., Jaitly, N., Le, Q., Vinyals, O. "*Listen Attend and Spell*." *ICASSP (2015).*

Attention Example



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

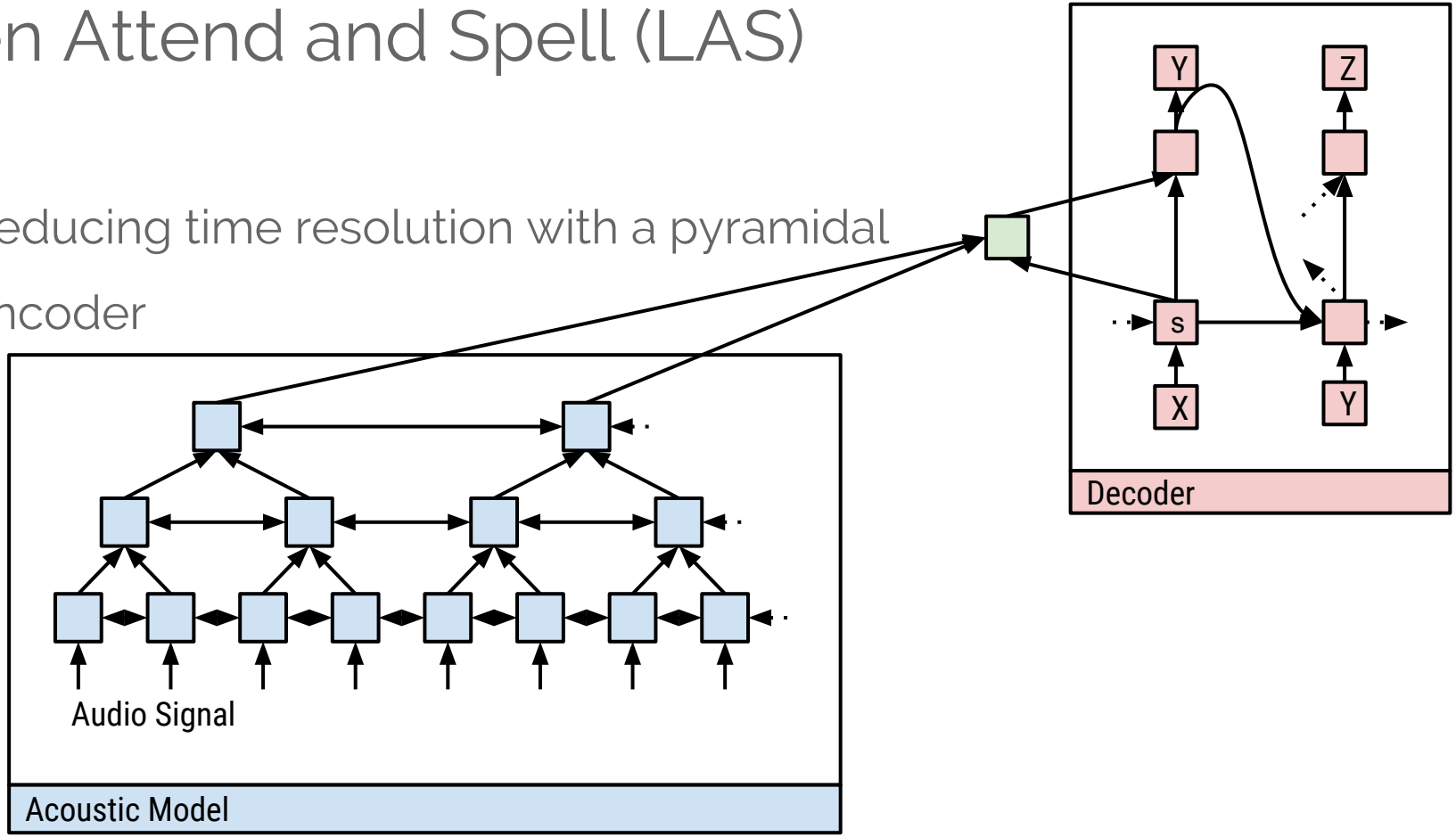
Attention Example



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

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).*

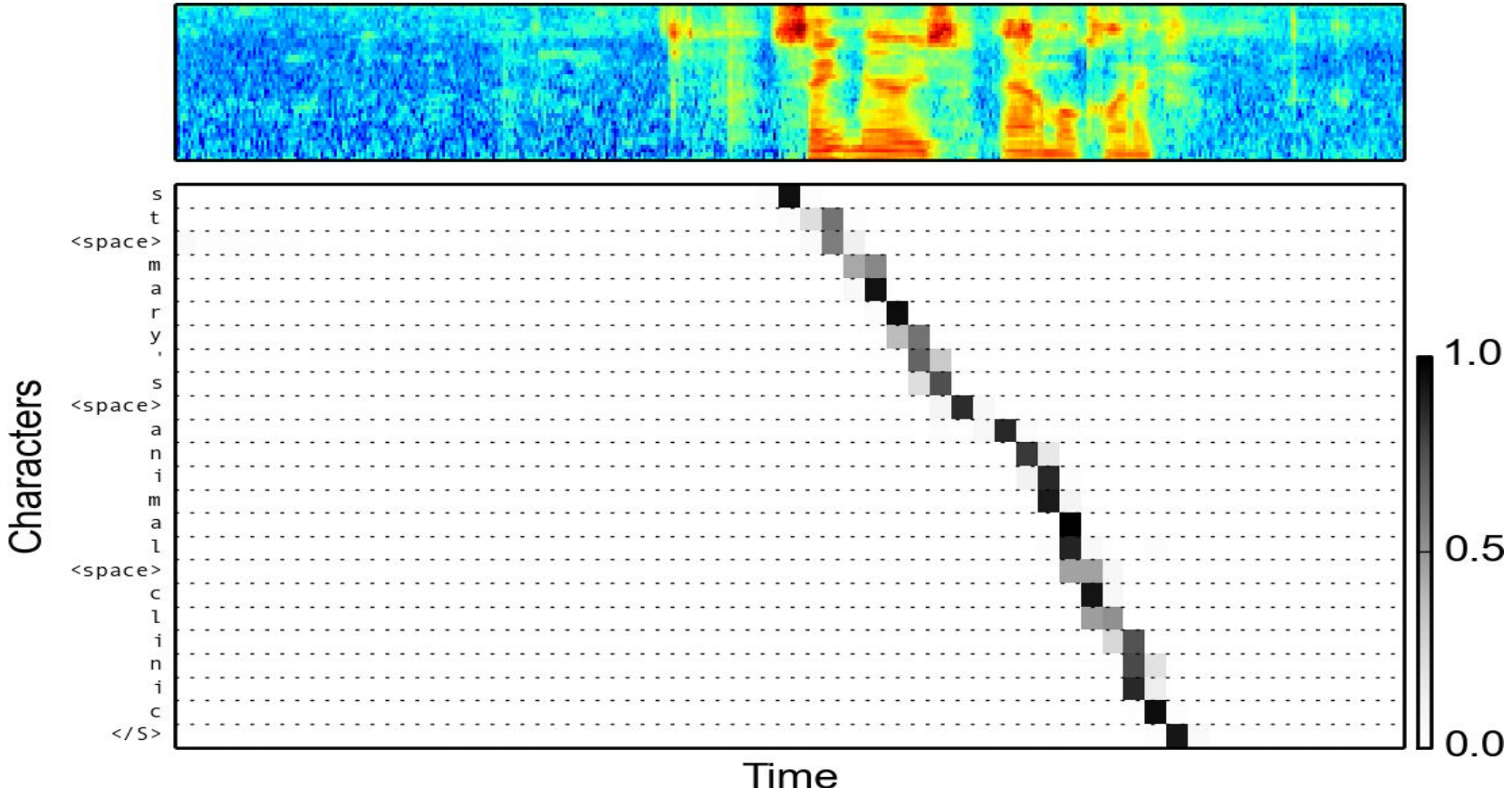
LAS - Multimodal outputs

Beam	Text	LogProb	WER
Truth	call aaa roadside assistance	-	-
1	call aaa roadside assistance	-0.5740	0.00
2	call triple a roadside assistance	-1.5399	50.0
3	call trip way roadside assistance	-3.5012	50.0
4	call xxx roadside assistance	-4.4375	25.0

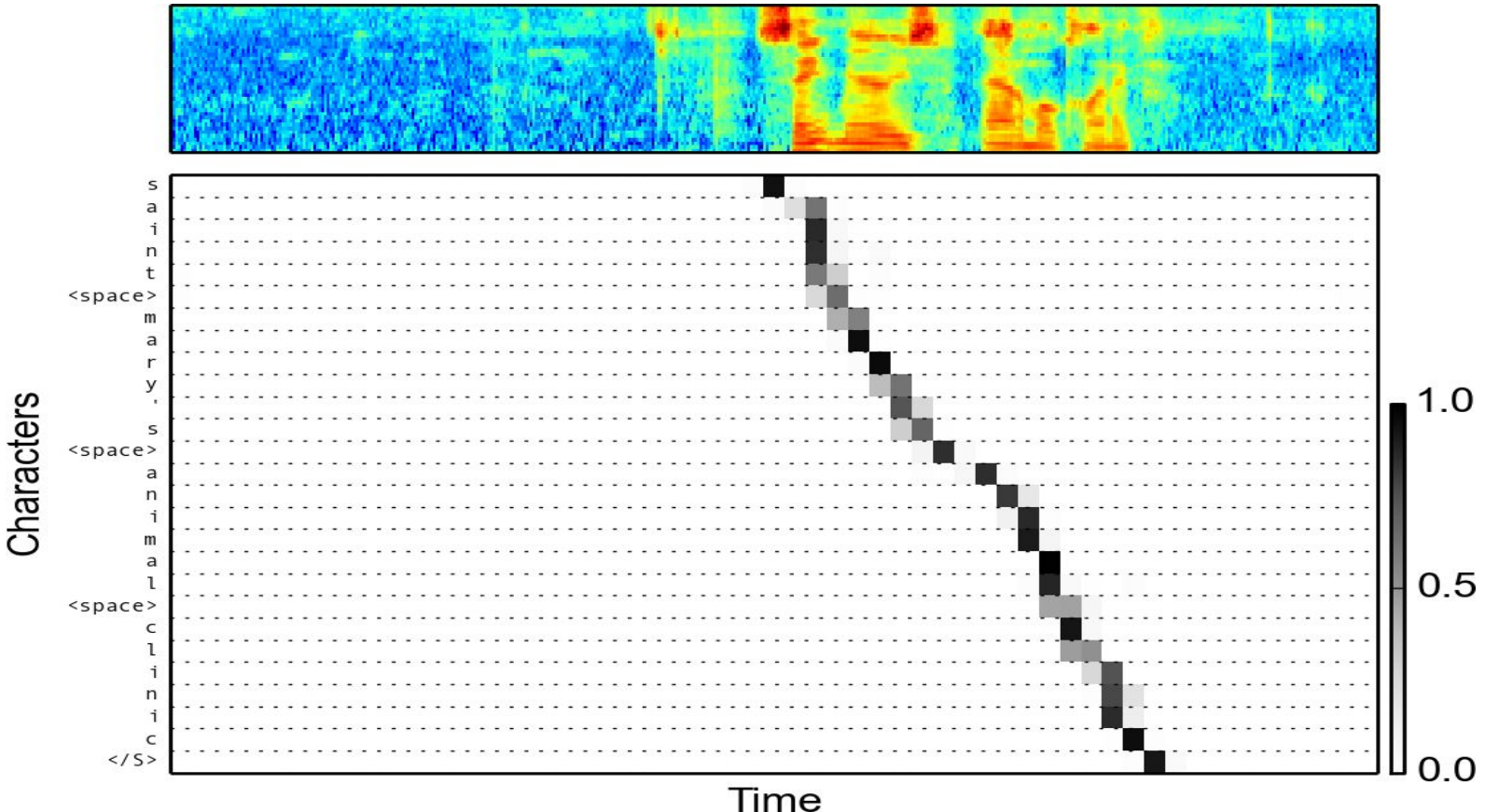
Very different outputs for same input!

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

LAS - Impact of Autoregressive Inputs on Attention



LAS - Impact of Autoregressive Inputs on Attention



LAS - Results

Model	Clean WER	Noisy WER
CLDNN-HMM (baseline)	8	8.9
Listen Attend and Spell (LAS)	14	16.5
LAS + external language model	10.3	12.0

Comparable to some of our best models without extensive engineering !

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

Lip Reading

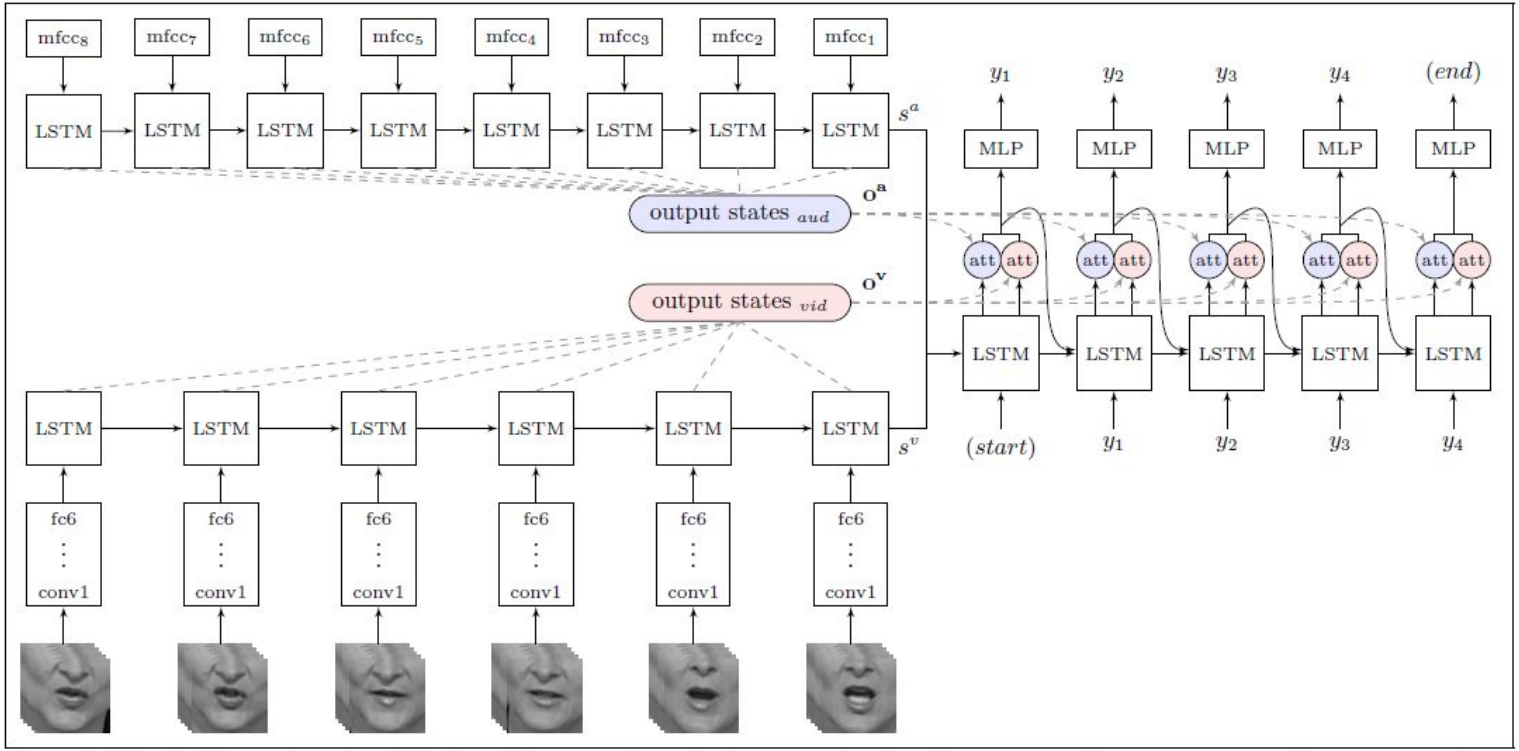
Channel	Series name	# hours	# sent.
BBC 1 HD	News†	1,584	50,493
BBC 1 HD	Breakfast	1,997	29,862
BBC 1 HD	Newsnight	590	17,004
BBC 2 HD	World News	194	3,504
BBC 2 HD	Question Time	323	11,695
BBC 4 HD	World Today	272	5,558
All		4,960	118,116

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).

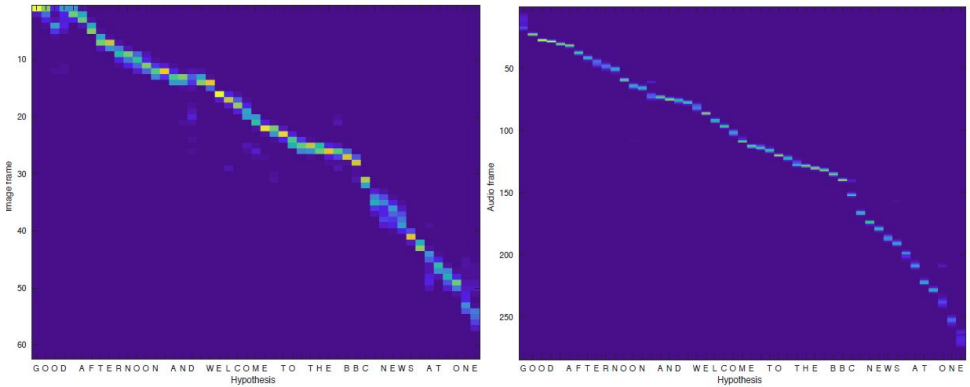
Lip Reading

Separate embedding and attention for audio and visual streams



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

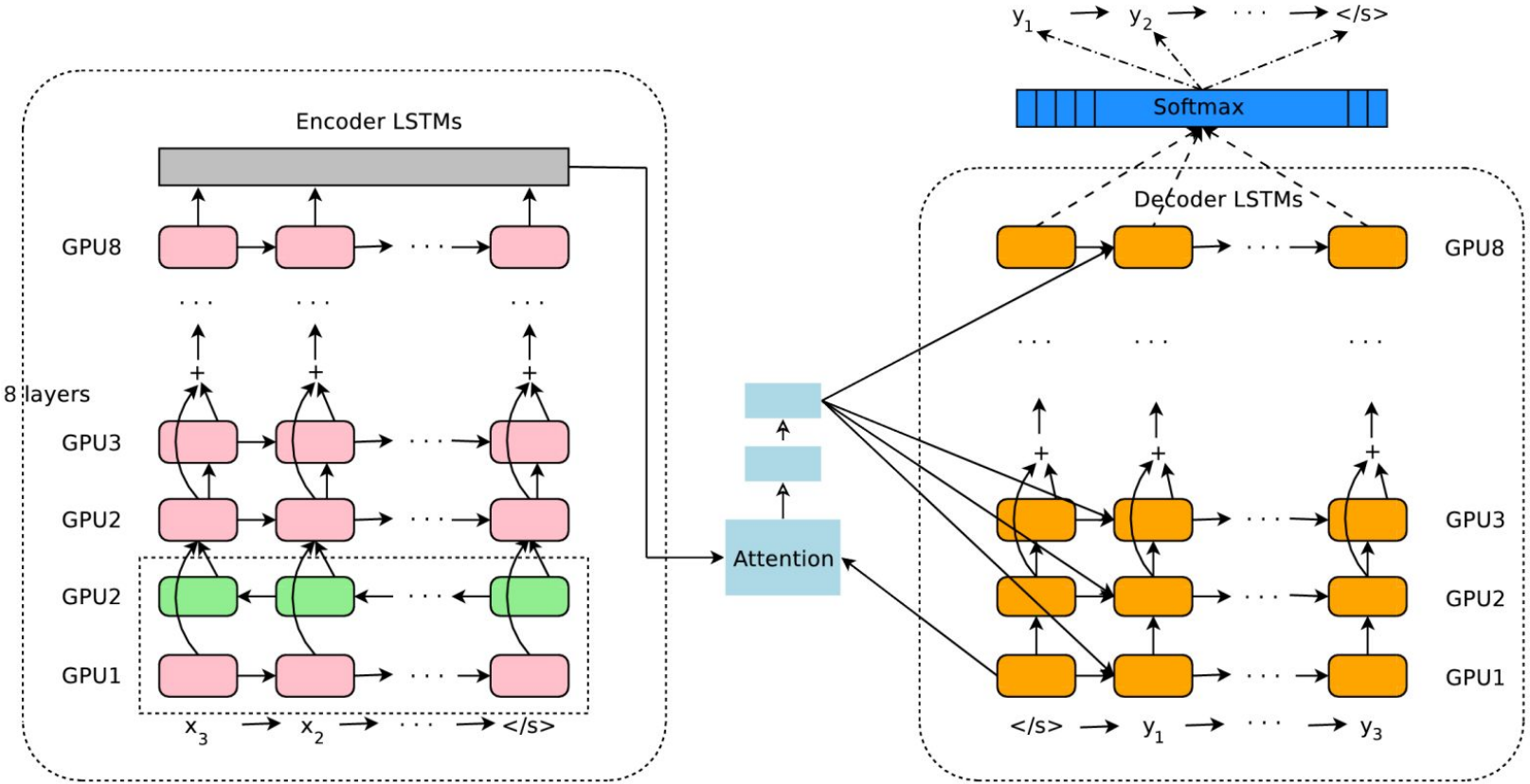
Lip Reading



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

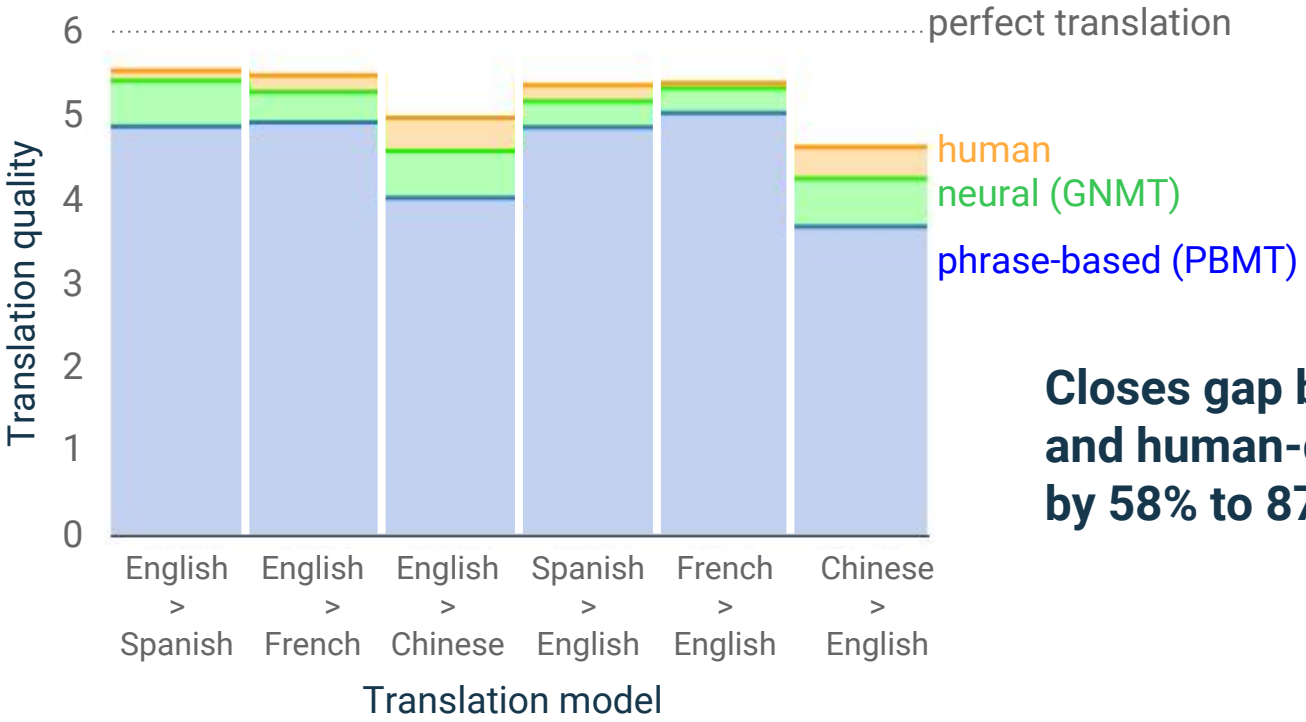
Chung, J., et al. "Lip reading sentences in the wild." *CVPR* (2017).

Google Neural Machine Translation System



Wu, Y., et al. "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation." *arxiv (2016).*

Google Neural Machine Translation

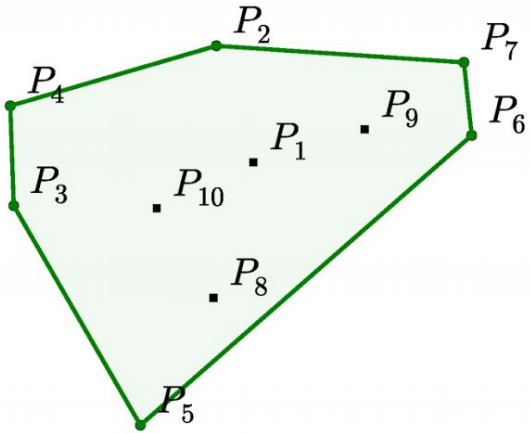


Closes gap between old system and human-quality translation by 58% to 87%

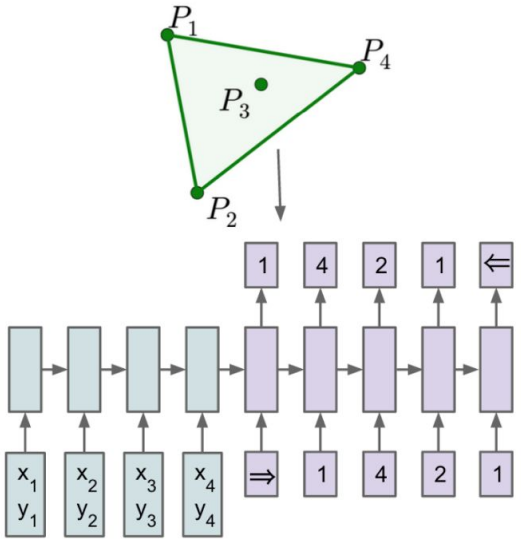
research.googleblog.com/2016/09/a-neural-network-for-machine.html

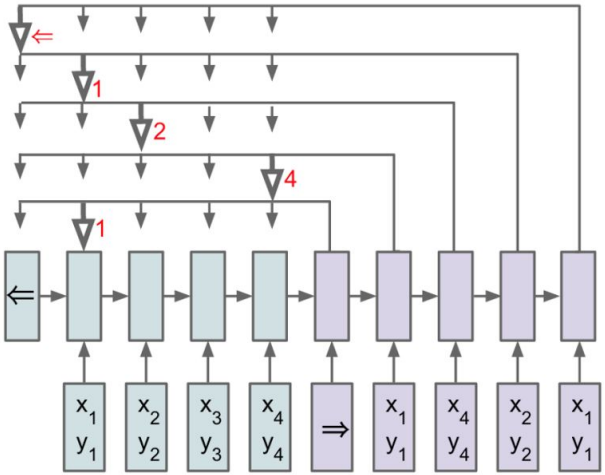
Pointer Networks

- Adaptation of sequence to sequence with attention for self referential outputs



Convex Hull

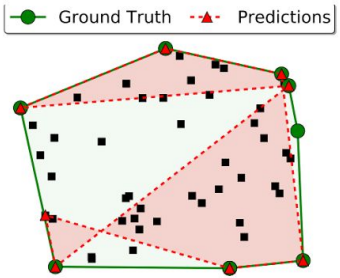




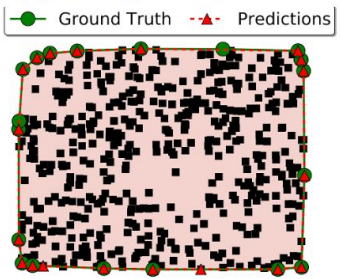
Vinyals, O., Fortunato, M., Jaitly. N. “Pointer networks.” *NIPS (2015).*

Pointer Networks

Convex hull

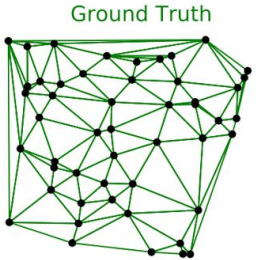


(a) LSTM, m=50, n=50

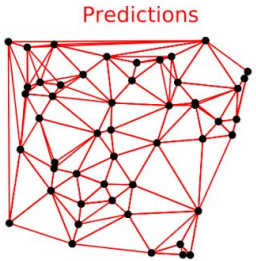


(d) Ptr-Net, m=5-50, n=500

Delauney Triangulation

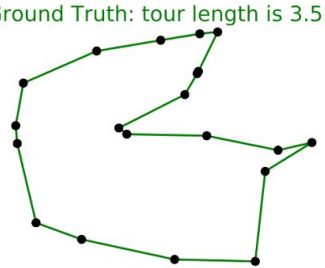


(b) Truth, n=50

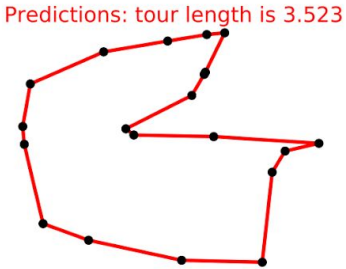


(e) Ptr-Net , m=50, n=50

Travelling Salesman Problem



(c) Truth, n=20



(f) Ptr-Net , m=5-20, n=20

- Merity, Stephen, et al. "Pointer sentinel mixture models." *arXiv preprint arXiv:1609.07843* (2016).
- Kadlec, Rudolf, et al. "Text understanding with the attention sum reader network." *arXiv preprint arXiv:1603.01547* (2016).
- Gulchere, C., et. al. “Pointing the unknown words” *ACL (2016)*

Break

Loss Function

Loss Functions

- Cross Entropy
- Scheduled Sampling [1]
- Expected Loss [2]
- Augmented Loss [3]
- Sequence to Sequence as a beam search optimization [4]
- Learning decoders with different loss function [5]

1. Bengio, S., et al. “Scheduled sampling for sequence prediction with recurrent neural networks.” *NIPS (2015).*
2. Ranzato, M., et al. “Sequence level training with recurrent neural networks.” *ICLR (2016).*
3. Norouzi, M., et al. “Reward augmented maximum likelihood for neural structured prediction.” *NIPS (2016).*
4. Wiseman, S., Rush, A. “Sequence-to-sequence learning as beam-search optimization.” *EMLP (2016).*
5. Gu, J, Cho, K and Li, V.O.K. "Trainable greedy decoding for neural machine translation." *arXiv preprint arXiv:1702.02429* (2017).

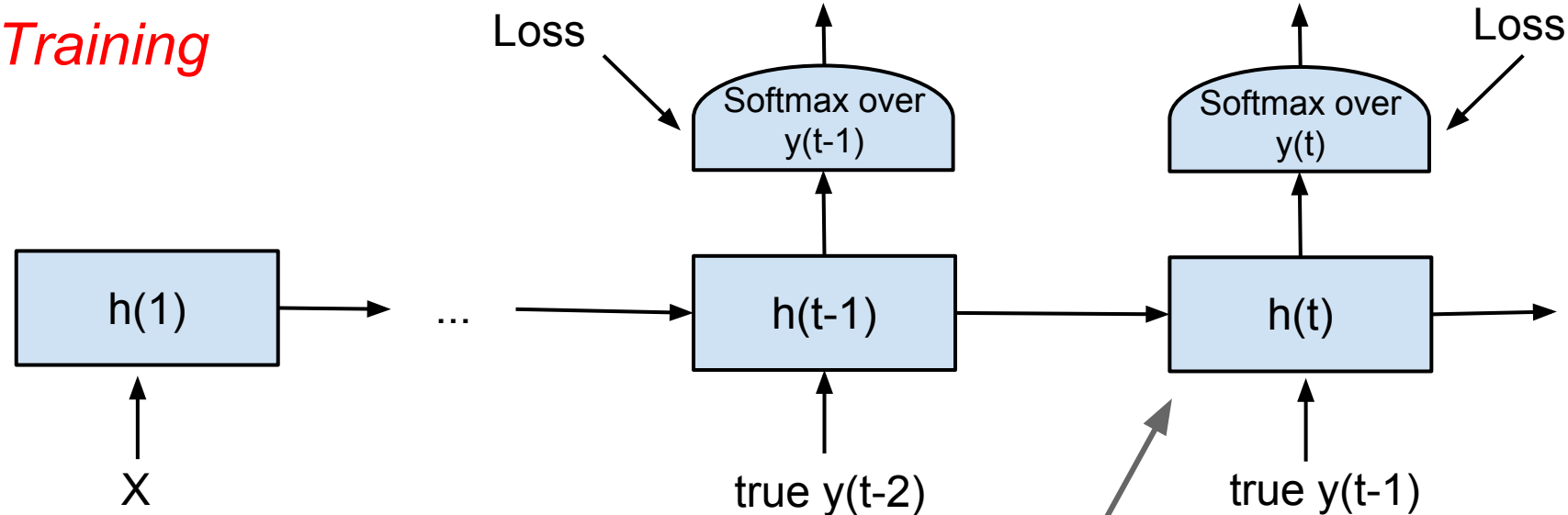
Cross Entropy (Negative Log Likelihood) Loss

- Log Likelihood, by chain rule is sum of next step log likelihoods $\log p(\mathbf{y}|\mathbf{x}) = \sum_{i=1}^{N} \log p(y_i|y_{<i}, \mathbf{x})$

- Supervised classification for each time step
  - depends on input, past outputs, which are known during training

Training and Inference Mismatch

Training

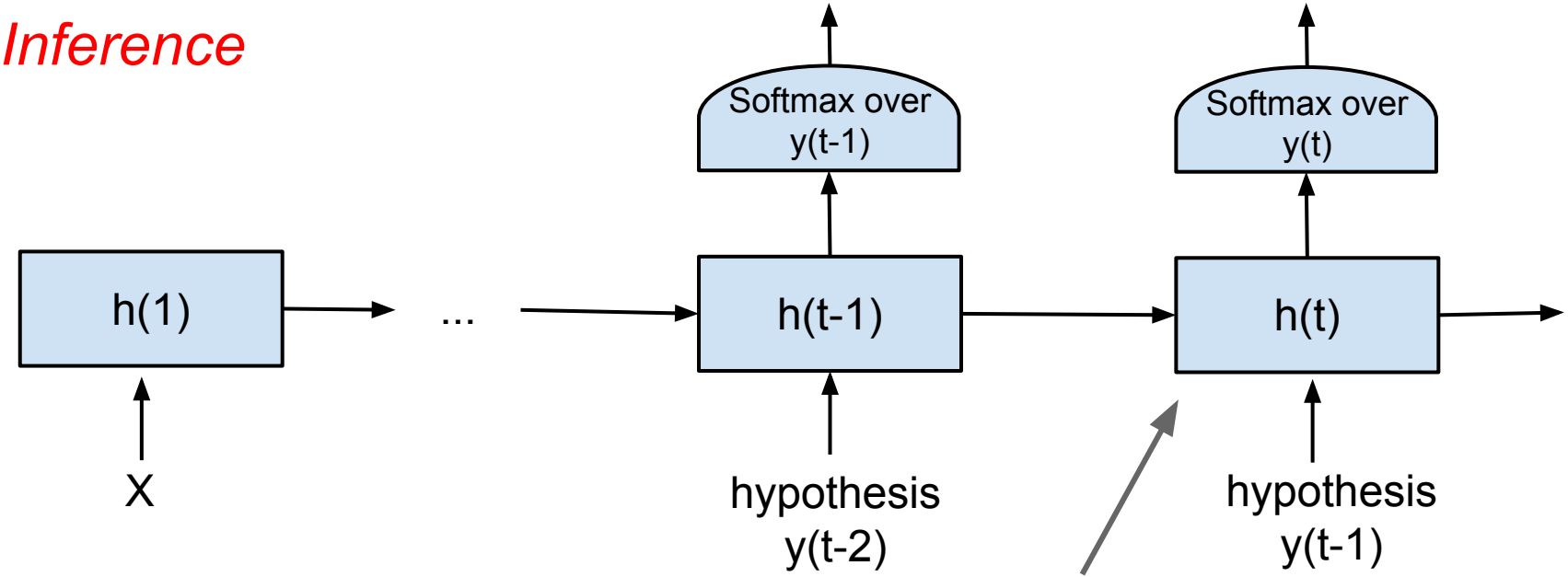


$$P(y_t|h_t) \text{ with } h_t = f(h_{t-1}, y_{t-1}; \theta)$$

Bengio, S., et al. “Scheduled sampling for sequence prediction with recurrent neural networks.” *NIPS (2015).*

Training and Inference Mismatch

Inference

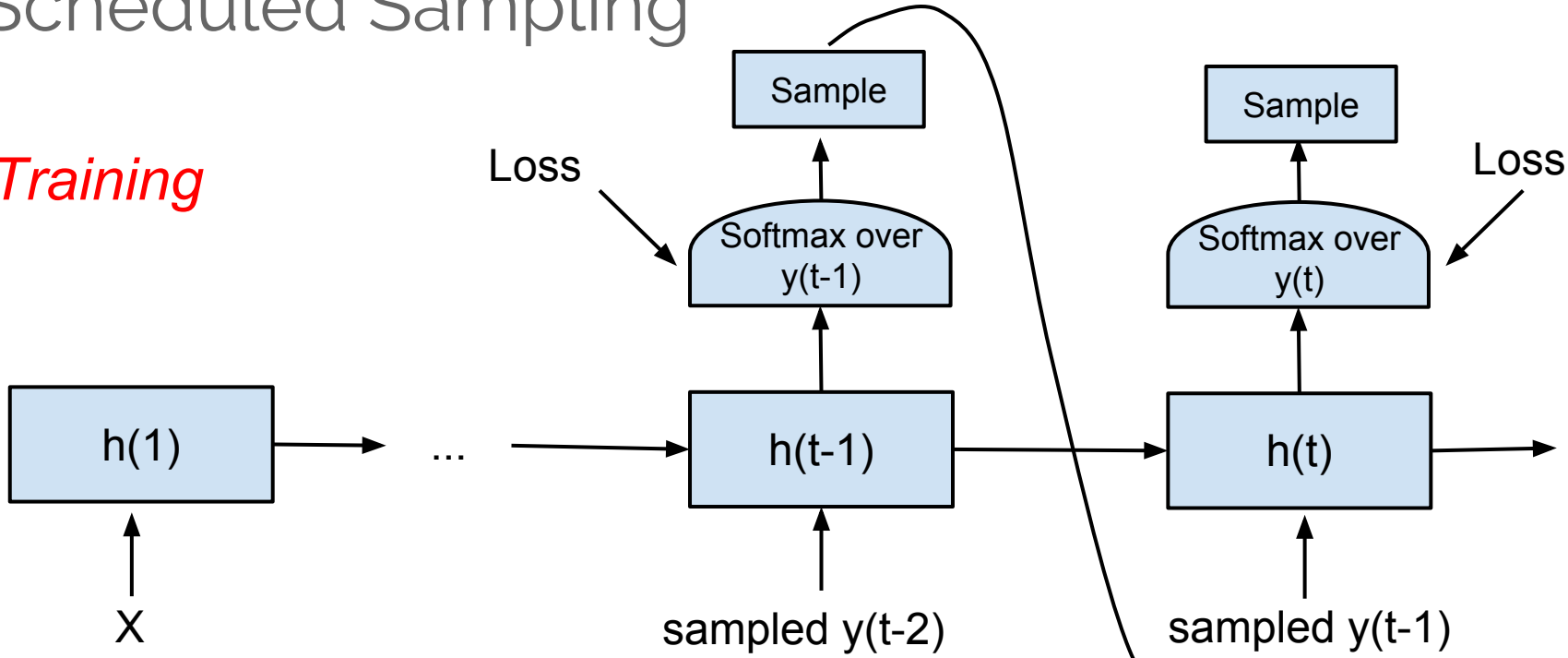


$$P(y_t|h_t) \text{ with } h_t = f(h_{t-1}, y_{t-1}; \theta)$$

Bengio, S., et al. “Scheduled sampling for sequence prediction with recurrent neural networks.” *NIPS (2015).*

Scheduled Sampling

Training



$$P(y_t|h_t) \text{ with } h_t = f(h_{t-1}, \hat{y}_{t-1}; \theta)$$

Bengio, S., et al. “Scheduled sampling for sequence prediction with recurrent neural networks.” *NIPS (2015).*

Scheduled Sampling

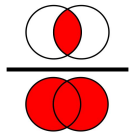
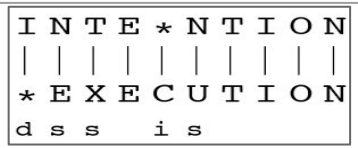
Machine Translation Model	Bleu-4	Meteor	Cider
Baseline	28.8	24.2	89.5
Baseline with dropout	28.1	23.9	87.0
Scheduled sampling	**30.6**	**24.3**	**92.1**

Parsing Model	F1
Baseline LSTM with dropout	87.00
Scheduled sampling with dropout	**88.68**

Speech Recognition Model	WER
LAS + LM Rescoring	12.6
LAS + **Sampling** + LM Rescoring	10.3

Bengio, S., et al. “Scheduled sampling for sequence prediction with recurrent neural networks.” *NIPS (2015).*

Rewards (-loss) used in Structured Prediction

TASK	REWARD	
Classification	0/1 rewards	$r(\mathbf{y}, \mathbf{y}^*) = \mathbb{1}[\mathbf{y} = \mathbf{y}^*]$
Segmentation	Intersection over Union	$r(\mathbf{y}, \mathbf{y}^*) = \cap(\mathbf{y}, \mathbf{y}^*) / \cup(\mathbf{y}, \mathbf{y}^*)$ 
Speech Recognition	Edit Distance	$r(\mathbf{y}, \mathbf{y}^*)$ = (#*d* + #*i* + #*s*) 
Machine Translation	BLEU	

Expected reward (-loss)

Given a dataset of input output pairs, $\mathcal{D} \equiv \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)*})\}_{i=1}^{N}$

learn a conditional distribution $p_\theta(\mathbf{y} \mid \mathbf{x})$ that minimizes

expected loss: $$\mathcal{L}_{\mathrm{RL}}(\boldsymbol{\theta}) = \sum_{(\mathbf{x},\mathbf{y}^*)\in\mathcal{D}} -\sum_{\mathbf{y}\in\mathcal{Y}} p_\theta(\mathbf{y} \mid x)\; r(\mathbf{y}, \mathbf{y}^*)$$

*Sample from the **model** distribution*

Difficult / Impossible to train from scratch!!

Mixed Incremental Cross-Entropy Reinforce (MIXER)

- Gradually interpolate from Cross-Entropy to Expected Loss

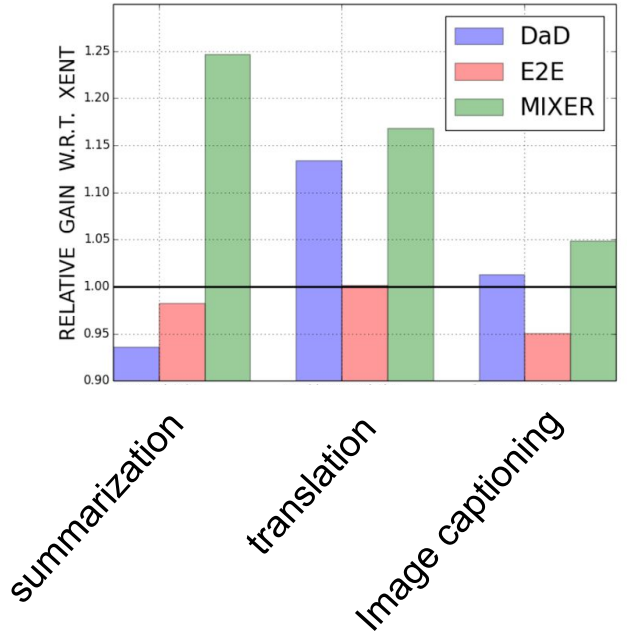
Data: a set of sequences with their corresponding context.
Result: RNN optimized for generation.
Initialize RNN at random and set N^{XENT}, $N^{\text{XE+R}}$ and Δ;
for $s = T, 1, -\Delta$ **do**
  if $s == T$ **then**
    train RNN for N^{XENT} epochs using XENT only;
  else
    train RNN for $N^{\text{XE+R}}$ epochs. Use XENT loss in the first s steps, and REINFORCE (sampling from the model) in the remaining $T - s$ steps;
  end
end

More expected loss optimization as training proceeds

Ranzato, M., et al. “Sequence level training with recurrent neural networks.” *ICLR (2016).*

Mixed Incremental Cross-Entropy Reinforce (MIXER)

TASK	XENT	DAD	E2E	MIXER
summarization	13.01	12.18	12.78	**16.22**
translation	17.74	20.12	17.77	**20.73**
image captioning	27.8	28.16	26.42	**29.16**



Ranzato, M., et al. “Sequence level training with recurrent neural networks.” *ICLR (2016).*

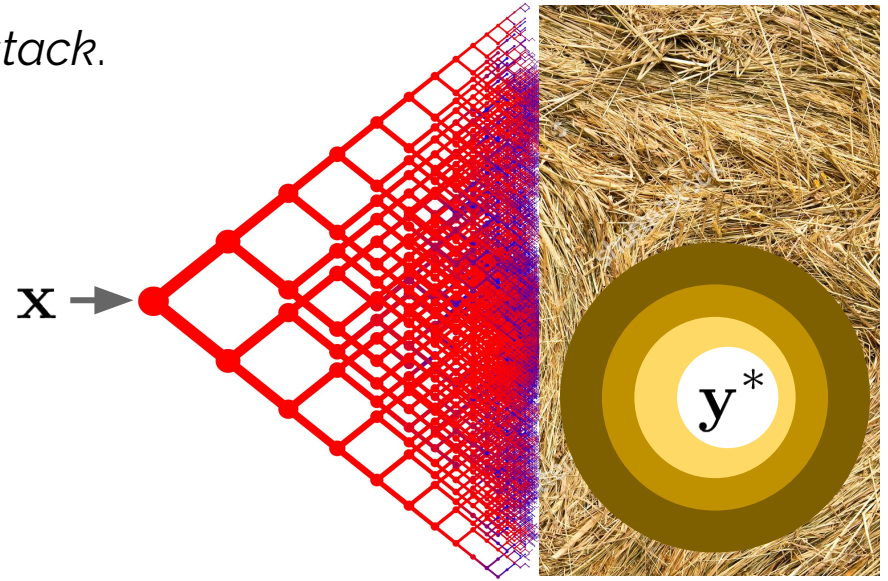
Reward Augmented Maximum Likelihood (RML)

Finding the *right output sequence*, for tasks like speech recognition or machine translation is like finding *a needle in a haystack*.

It is very risky to shoot *only* for the *true target*.

What if we expand the targets to make learning easier?

E.g. by inserting, deleting random words…



Norouzi, M., et al. “Reward augmented maximum likelihood for neural structured prediction.” *NIPS (2016).*

Reward augmented maximum likelihood (RML)

$$\mathcal{L}_{\mathrm{RML}}(\boldsymbol{\theta}; \tau) = \sum_{(\mathbf{x}, \mathbf{y}^*) \in \mathcal{D}} \left\{ - \sum_{\mathbf{y} \in \mathcal{Y}} q(\mathbf{y} \mid \mathbf{y}^*; \tau) \log p_\theta(\mathbf{y} \mid \mathbf{x}) \right\}$$

Optimal $p_\theta(\mathbf{y} \mid \mathbf{x})$:

$$q(\mathbf{y} \mid \mathbf{y}^*; \tau) = \frac{1}{Z(\mathbf{y}^*, \tau)} \exp\{r(\mathbf{y}, \mathbf{y}^*)/\tau\}$$

*Sample from the **reward** distribution, irrespective of the model*

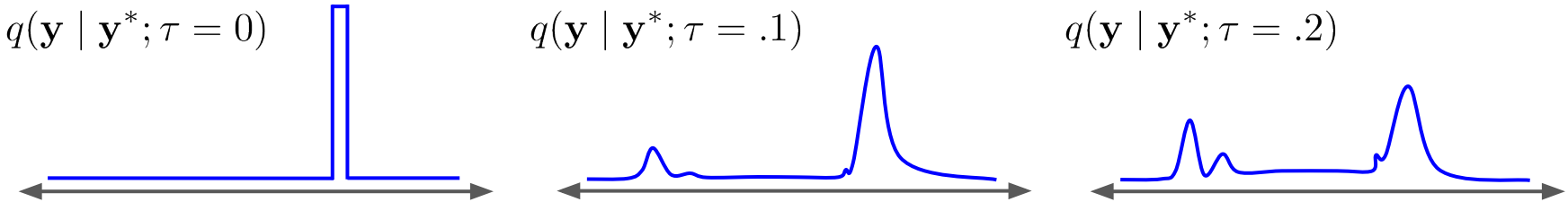
$$\mathcal{L}_{\mathrm{RML}}(\boldsymbol{\theta}; \tau) = \sum_{(\mathbf{x}, \mathbf{y}^*) \in \mathcal{D}} D_{\mathrm{KL}}\left(q(\mathbf{y} \mid \mathbf{y}^*; \tau) \parallel p_\theta(\mathbf{y} \mid \mathbf{x})\right) + \text{constant}$$

Norouzi, M., et al. “Reward augmented maximum likelihood for neural structured prediction.” *NIPS (2016).*

RML - Impact of temperature τ

Temperature impacts spread of distribution we sample from

Cross Entropy Targets → More spread

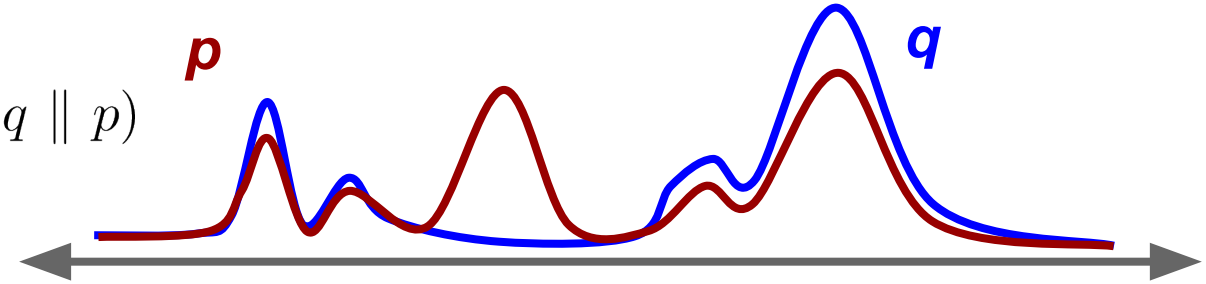


$$\mathcal{L}_{\mathrm{RML}}(\boldsymbol{\theta}; \tau) = \sum_{(\mathbf{x}, \mathbf{y}^*) \in \mathcal{D}} D_{\mathrm{KL}}\left(q(\mathbf{y} \mid \mathbf{y}^*; \tau) \parallel p_\theta(\mathbf{y} \mid \mathbf{x})\right) + \text{constant}$$

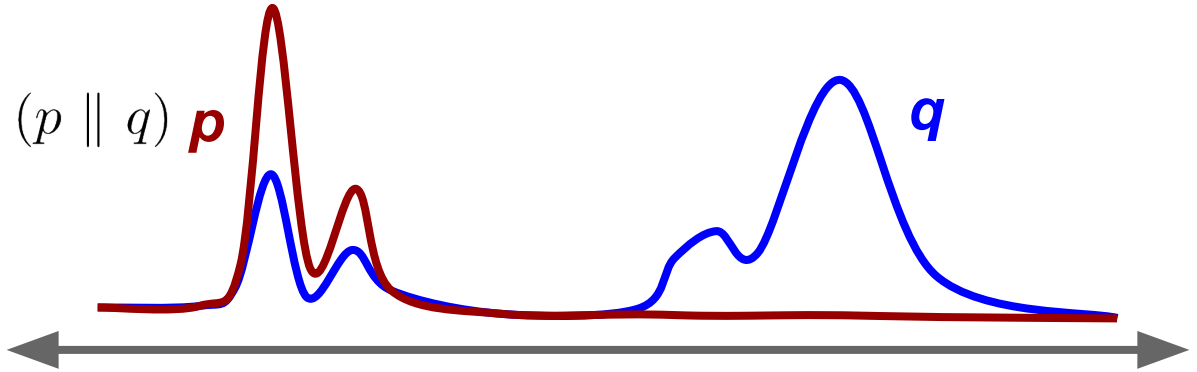
Norouzi, M., et al. “Reward augmented maximum likelihood for neural structured prediction.” *NIPS (2016).*

Contrasting RML with RL

RML $D_{\mathrm{KL}}(q \parallel p)$



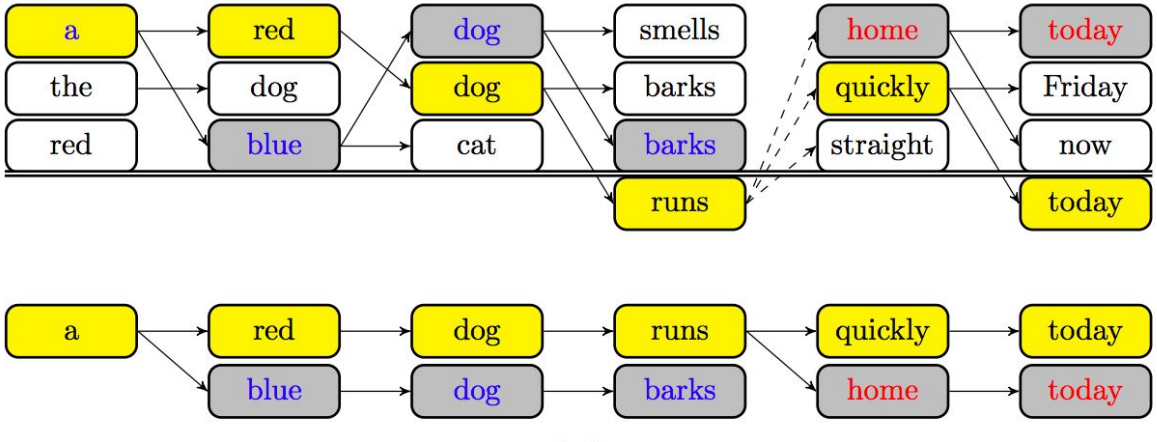
RL $D_{\mathrm{KL}}(p \parallel q)$



Norouzi, M., et al. “Reward augmented maximum likelihood for neural structured prediction.” *NIPS (2016).*

Margin Loss

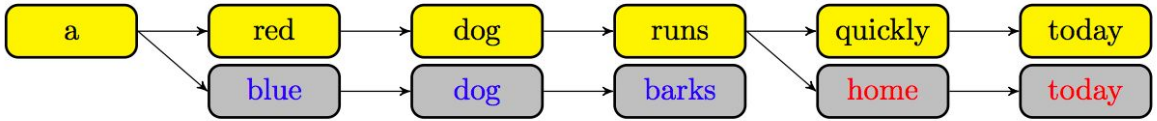
- Perform beam search until correct hypothesis falls out of the beam
- Restart beam whenever there is a violation
- Extract correct hypothesis and competing hypotheses



Wiseman, S., Rush, A. “Sequence-to-sequence learning as beam-search optimization.” *EMLP (2016).*

Margin Loss

- Add a margin score for all time steps where the correct hypothesis is not better than the Kth best hypothesis by a certain margin



$$\mathcal{L}(f) = \sum_{t=1}^{T} \Delta(\hat{y}_{1:t}^{(K)}) \left[1 - f(y_t, \boldsymbol{h}_{t-1}) + f(\hat{y}_t^{(K)}, \hat{\boldsymbol{h}}_{t-1}^{(K)})\right]$$

Loss for error; 0 when margin constraint is satisfied

Score function for prediction of current output

Score function for prediction of K^{th} best output

Wiseman, S., Rush, A. “Sequence-to-sequence learning as beam-search optimization.” *EMLP (2016).*

Margin Loss

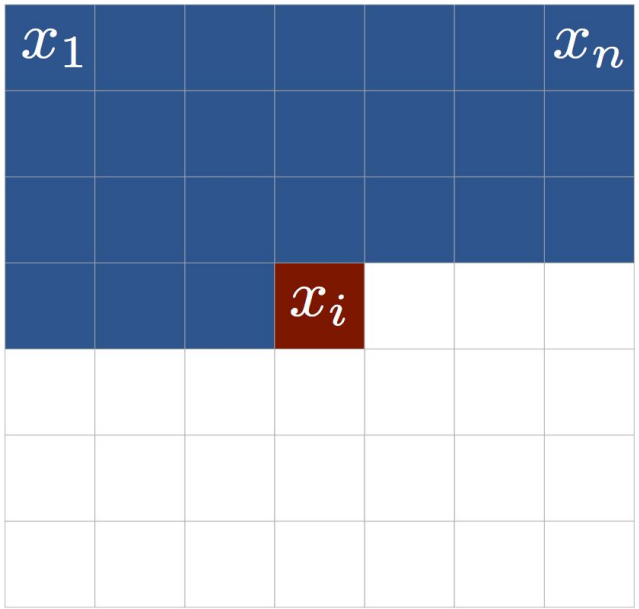
	Machine Translation (BLEU)		
	$K_{te} = 1$	$K_{te} = 5$	$K_{te} = 10$
seq2seq	22.53	24.03	23.87
BSO, SB-Δ	**23.83**	**26.36**	**25.48**
XENT	17.74	20.10	20.28
DAD	20.12	22.25	22.40
MIXER	20.73	21.81	21.83

Wiseman, S., Rush, A. “Sequence-to-sequence learning as beam-search optimization.” *EMLP (2016).*

Advanced Topics

Autoregressive Generative Models

PixelRNN - Model

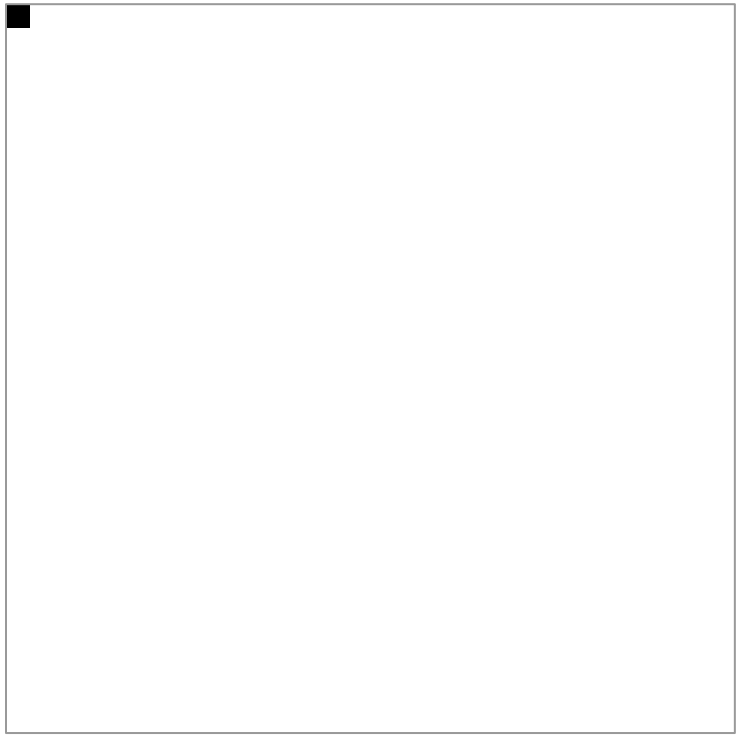


$$p(\mathbf{x}) = \prod_{i=1}^{n^2} p(x_i | x_1, \ldots, x_{i-1})$$

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

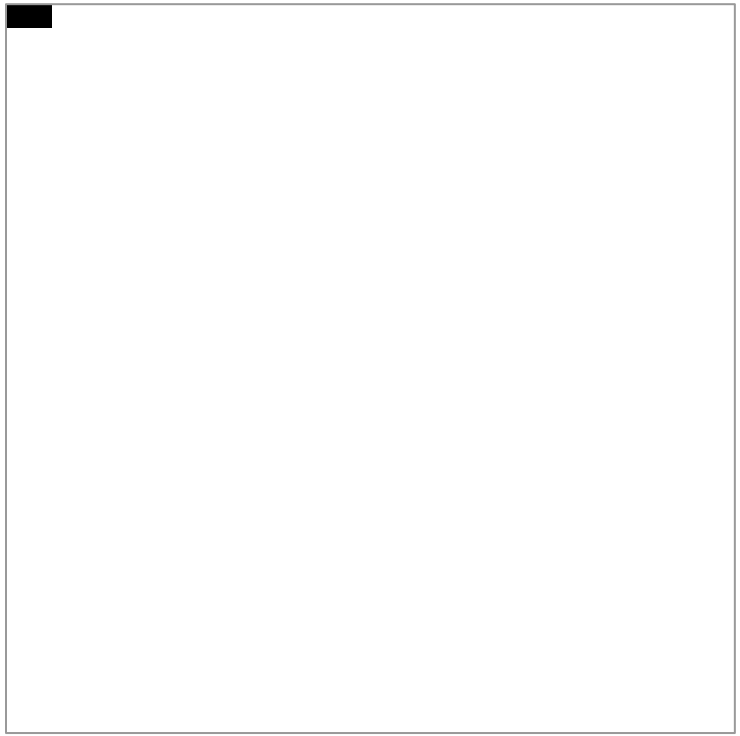
van den Oord, A., et al. “Pixel Recurrent Neural Networks.” *ICML (2016).*

Softmax Sampling



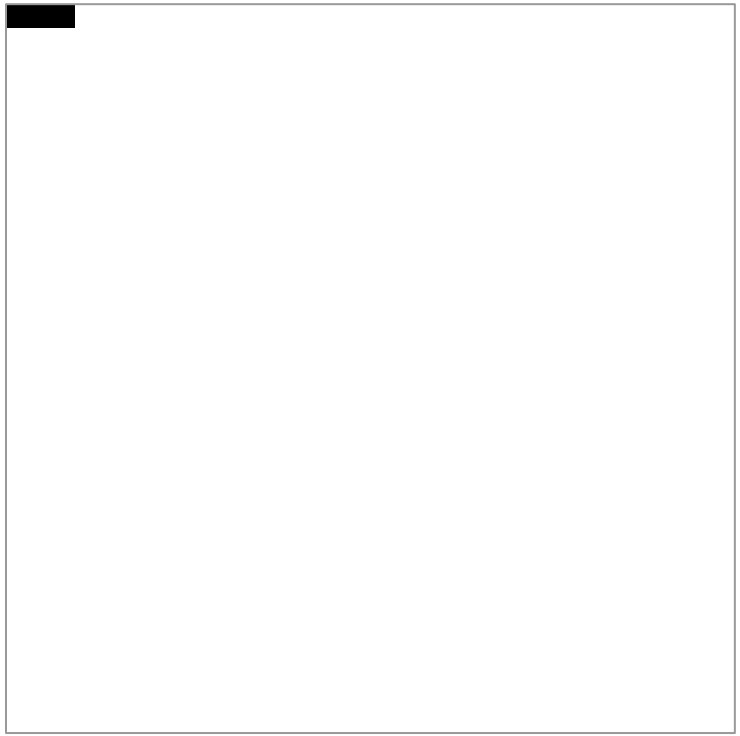
van den Oord, A., et al. “Pixel Recurrent Neural Networks.” *ICML (2016).*

Softmax Sampling



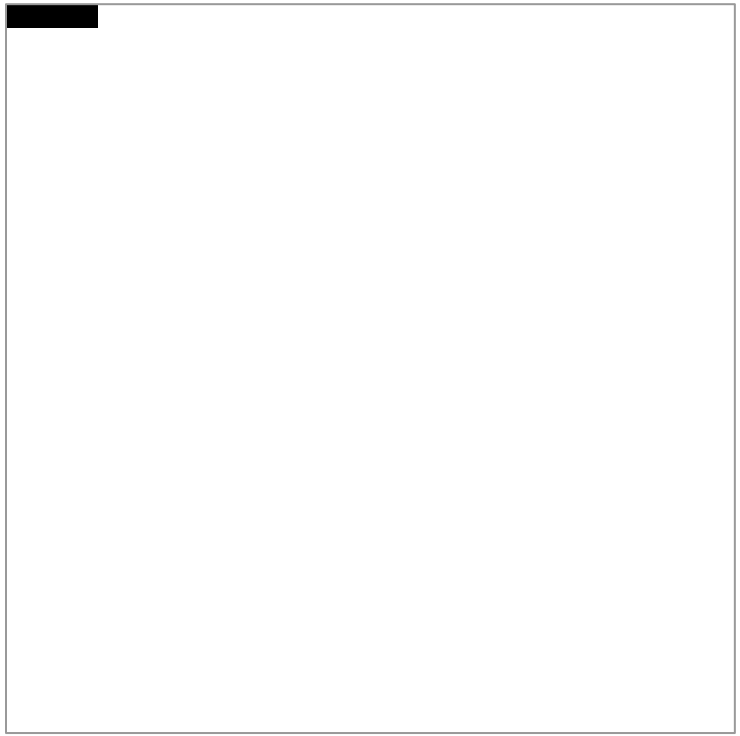
van den Oord, A., et al. “Pixel Recurrent Neural Networks.” *ICML (2016).*

Softmax Sampling



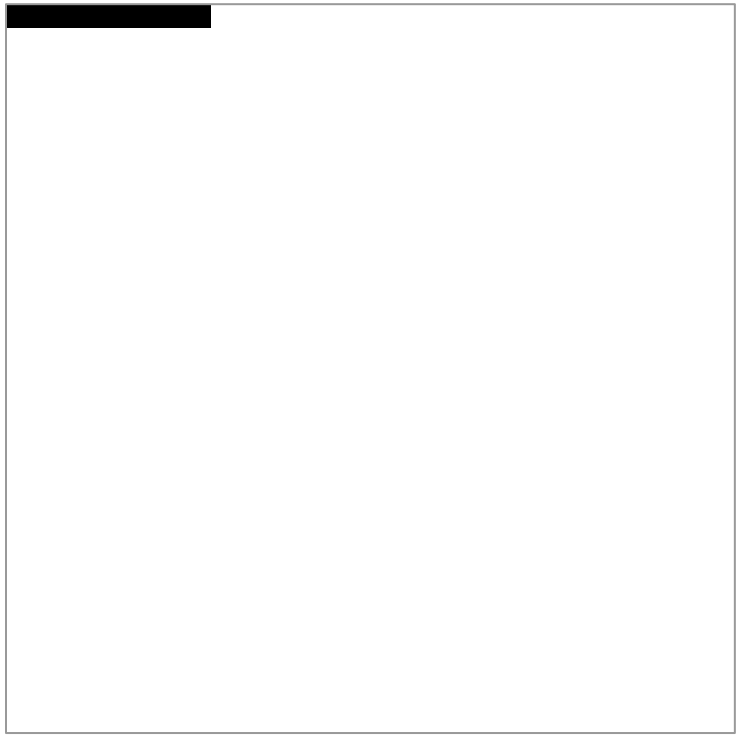
van den Oord, A., et al. “Pixel Recurrent Neural Networks.” *ICML (2016).*

Softmax Sampling



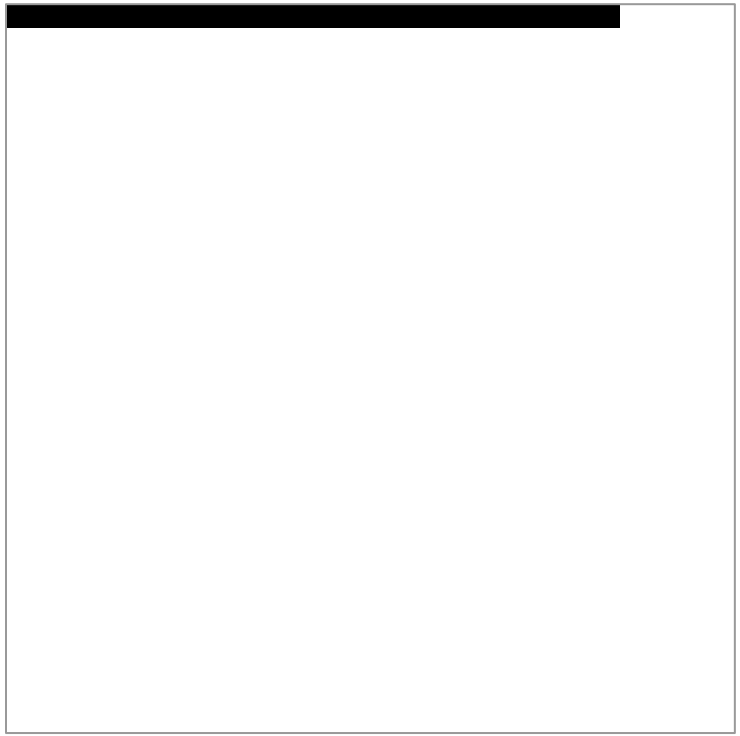
van den Oord, A., et al. “Pixel Recurrent Neural Networks.” *ICML (2016).*

Softmax Sampling



van den Oord, A., et al. “Pixel Recurrent Neural Networks.” *ICML (2016).*

Softmax Sampling



van den Oord, A., et al. “Pixel Recurrent Neural Networks.” *ICML (2016).*

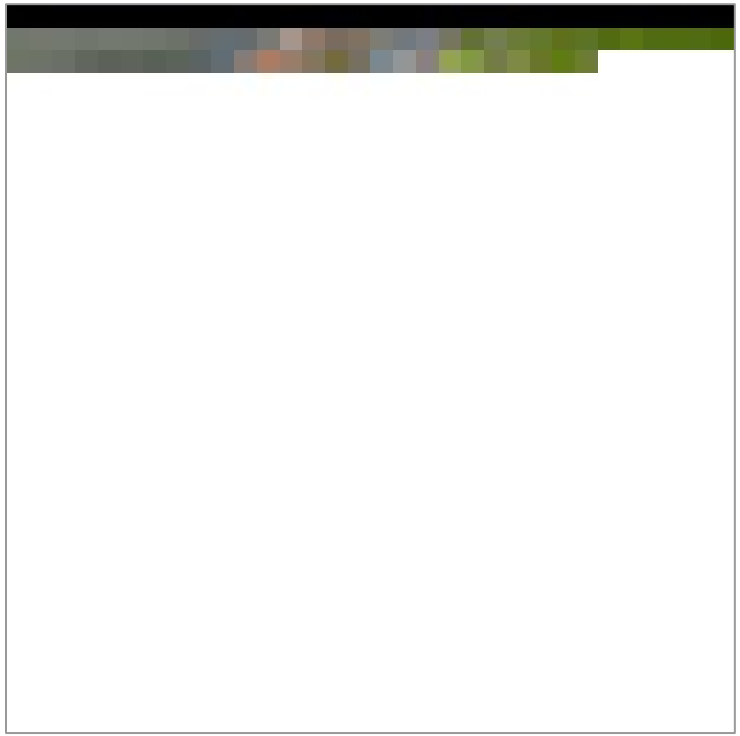
Softmax Sampling

van den Oord, A., et al. “Pixel Recurrent Neural Networks.” *ICML (2016).*

Softmax Sampling

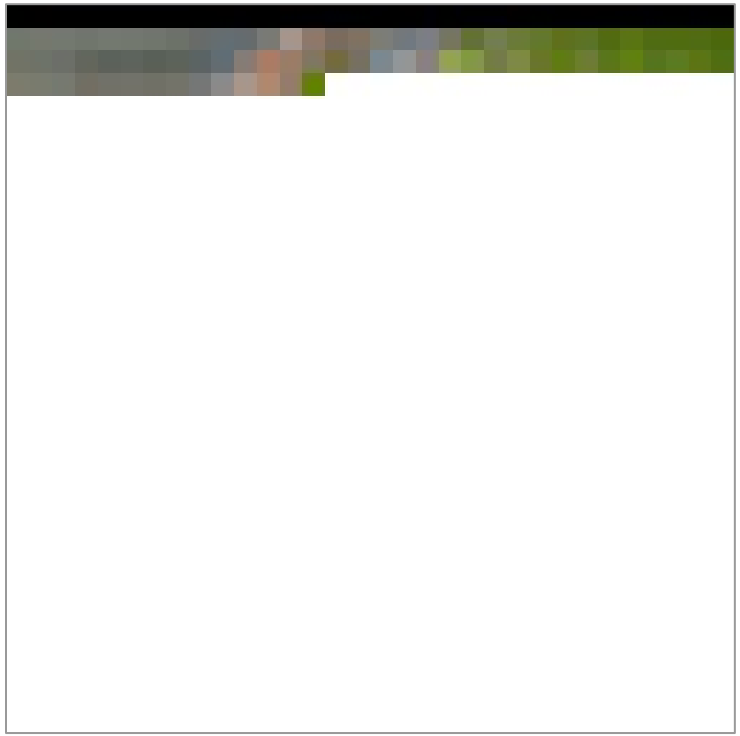
van den Oord, A., et al. “Pixel Recurrent Neural Networks.” *ICML (2016).*

Softmax Sampling



van den Oord, A., et al. “Pixel Recurrent Neural Networks.” *ICML (2016).*

Softmax Sampling

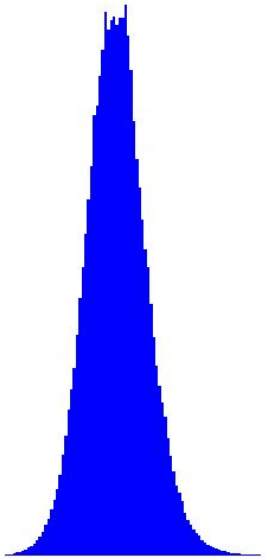


van den Oord, A., et al. “Pixel Recurrent Neural Networks.” *ICML (2016).*

Softmax Sampling

van den Oord, A., et al. “Pixel Recurrent Neural Networks.” *ICML (2016).*

Softmax Sampling

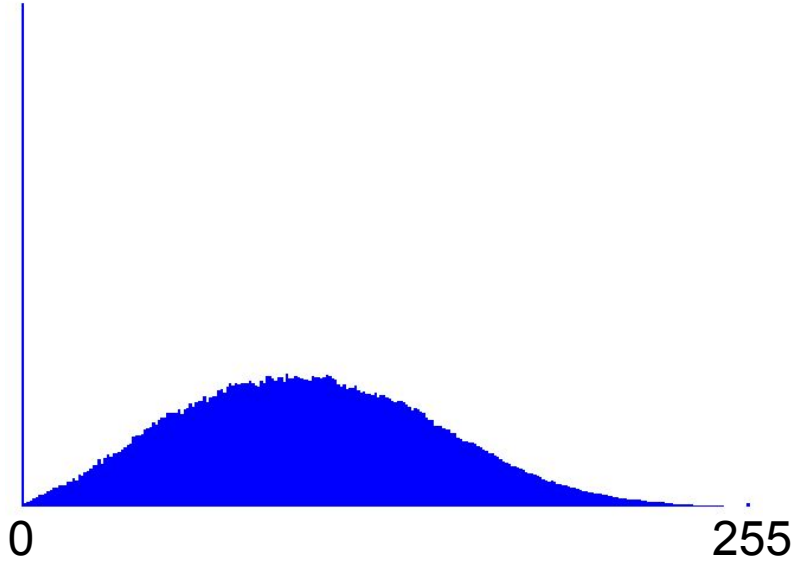


0

255

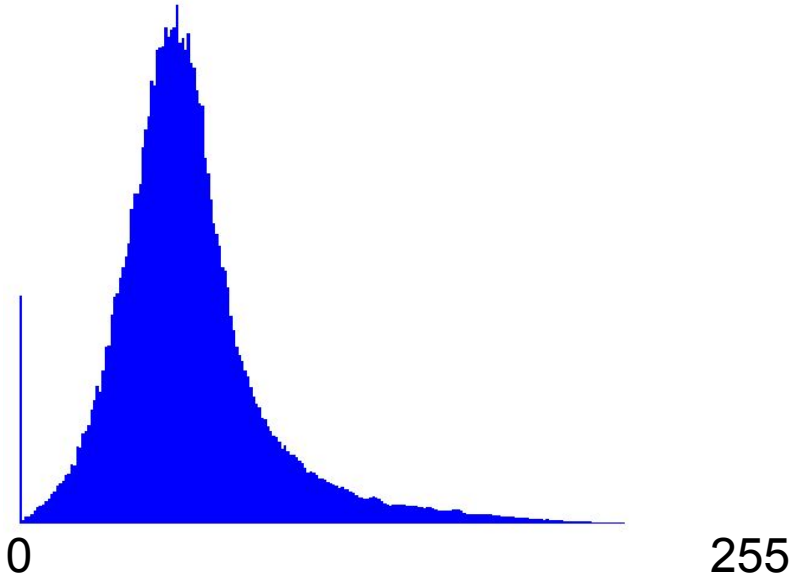
van den Oord, A., et al. “Pixel Recurrent Neural Networks.” *ICML (2016).*

Softmax Sampling



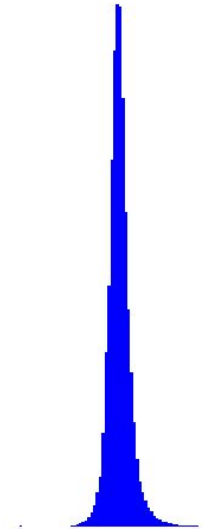
van den Oord, A., et al. “Pixel Recurrent Neural Networks.” *ICML (2016).*

Softmax Sampling



van den Oord, A., et al. “Pixel Recurrent Neural Networks.” *ICML (2016).*

Softmax Sampling

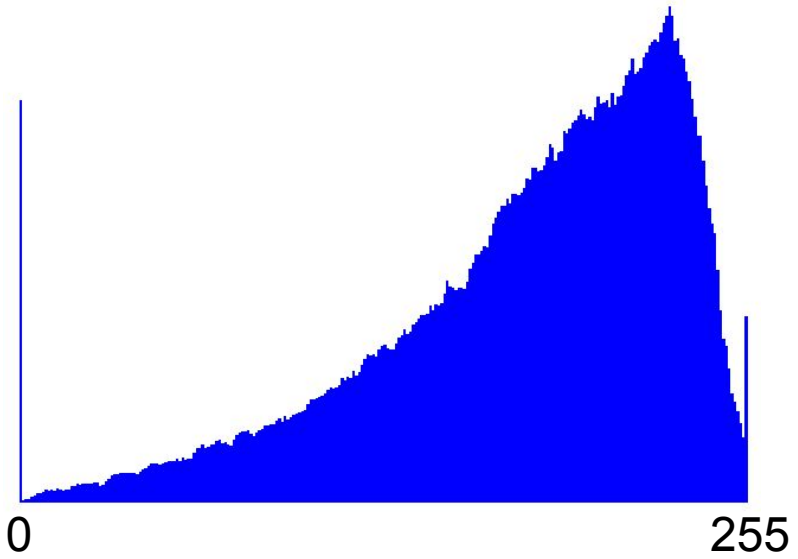


0

255

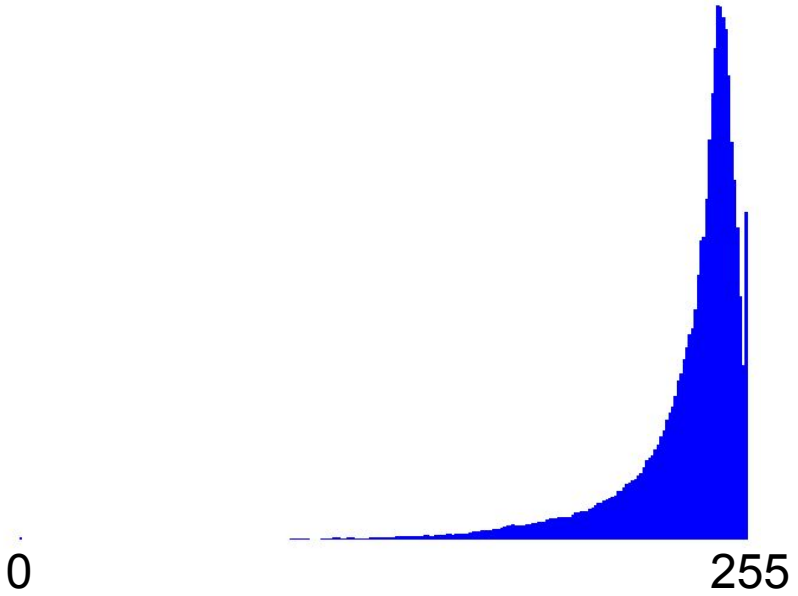
van den Oord, A., et al. “Pixel Recurrent Neural Networks.” *ICML (2016).*

Softmax Sampling



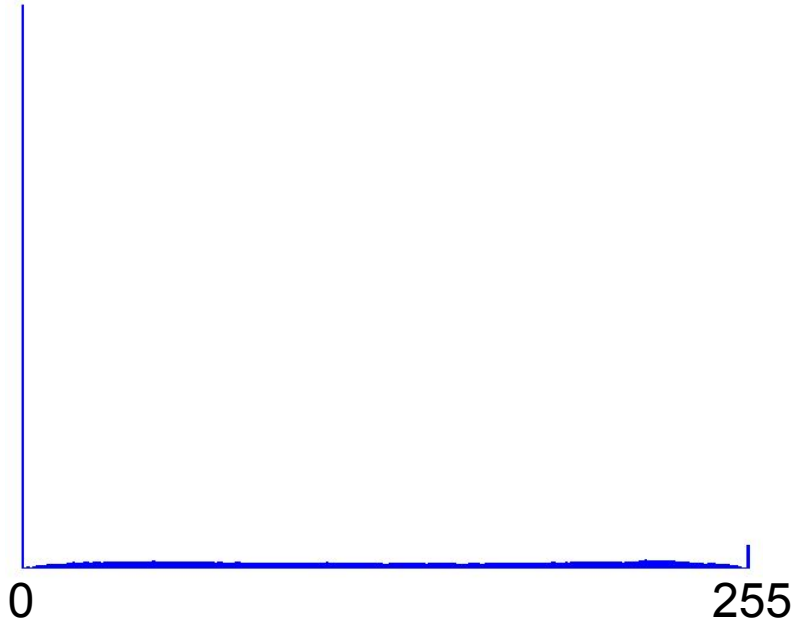
van den Oord, A., et al. “Pixel Recurrent Neural Networks.” *ICML (2016).*

Softmax Sampling



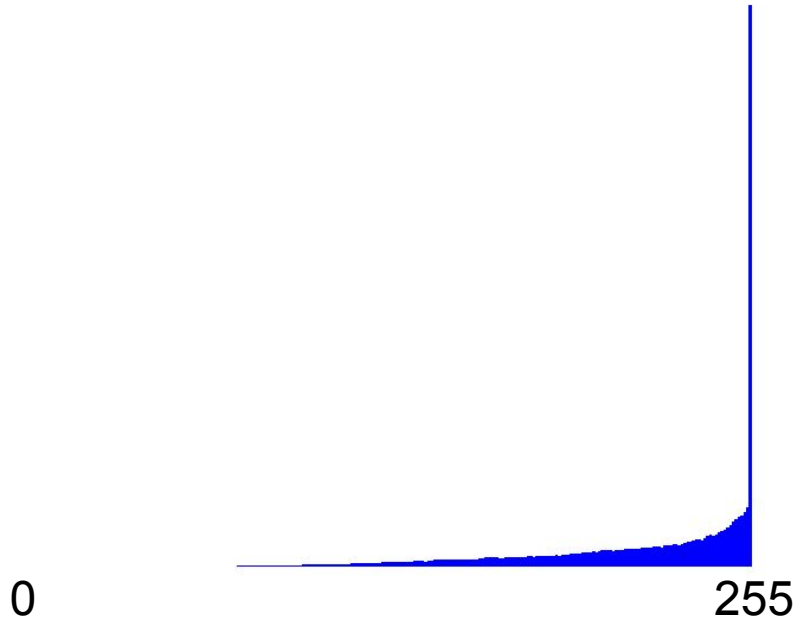
van den Oord, A., et al. “Pixel Recurrent Neural Networks.” *ICML (2016).*

Softmax Sampling



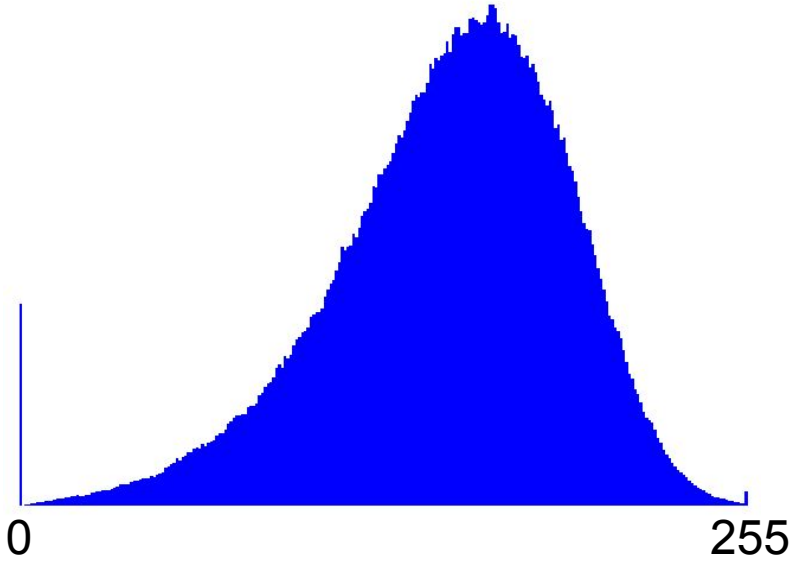
van den Oord, A., et al. “Pixel Recurrent Neural Networks.” *ICML (2016).*

Softmax Sampling



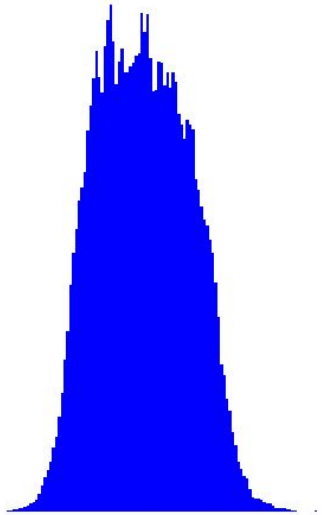
van den Oord, A., et al. “Pixel Recurrent Neural Networks.” *ICML (2016).*

Softmax Sampling



van den Oord, A., et al. “Pixel Recurrent Neural Networks.” *ICML (2016).*

Softmax Sampling

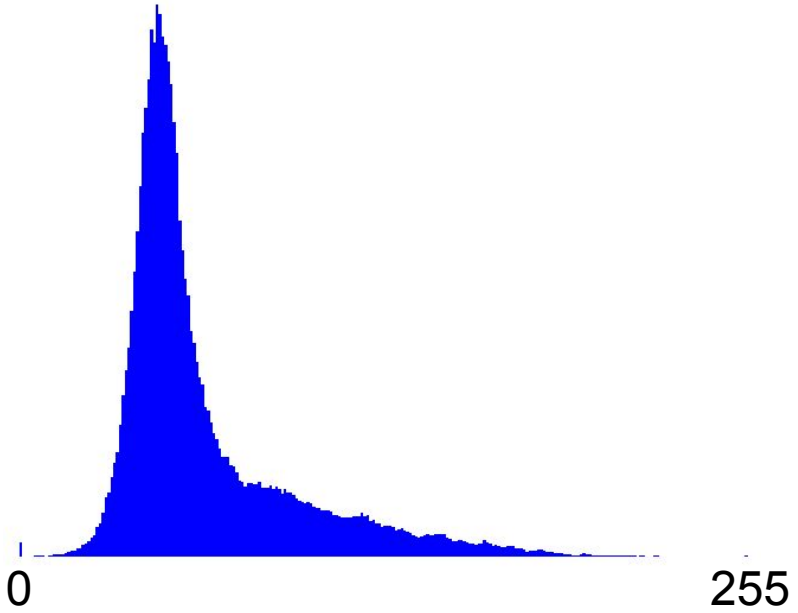


0

255

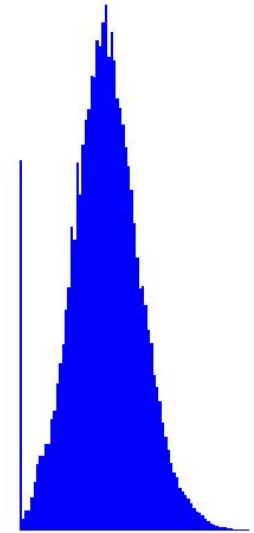
van den Oord, A., et al. “Pixel Recurrent Neural Networks.” *ICML (2016).*

Softmax Sampling



van den Oord, A., et al. “Pixel Recurrent Neural Networks.” *ICML (2016).*

Softmax Sampling



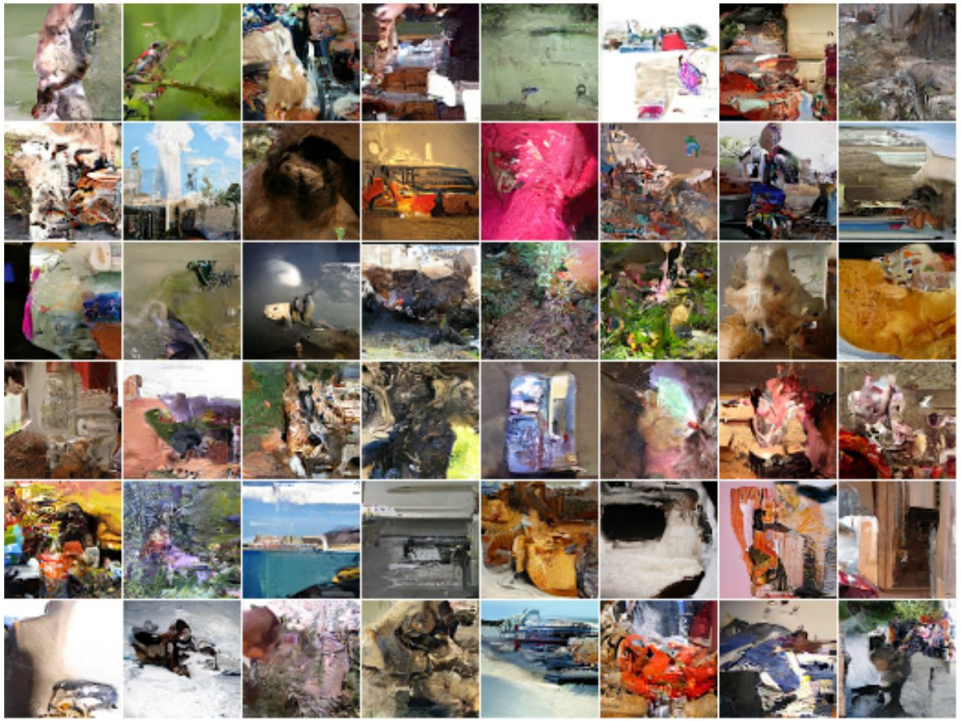
0

255

van den Oord, A., et al. “Pixel Recurrent Neural Networks.” *ICML (2016).*

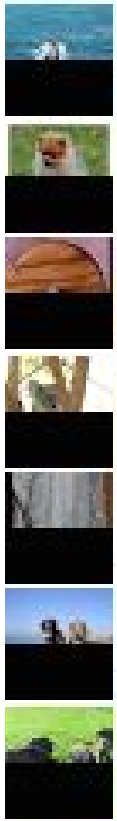
Pixel RNN

Sequence of Words == Sequence of Pixels



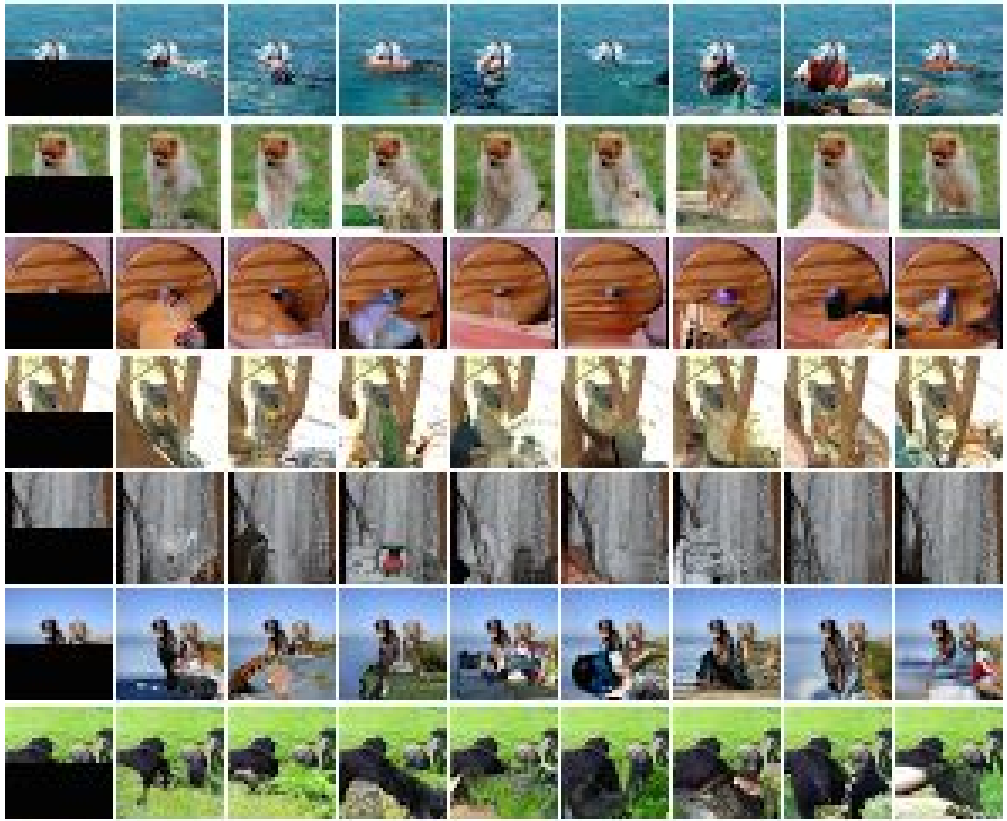
van den Oord, A., et al. “Pixel Recurrent Neural Networks.” *ICML (2016).*

occluded



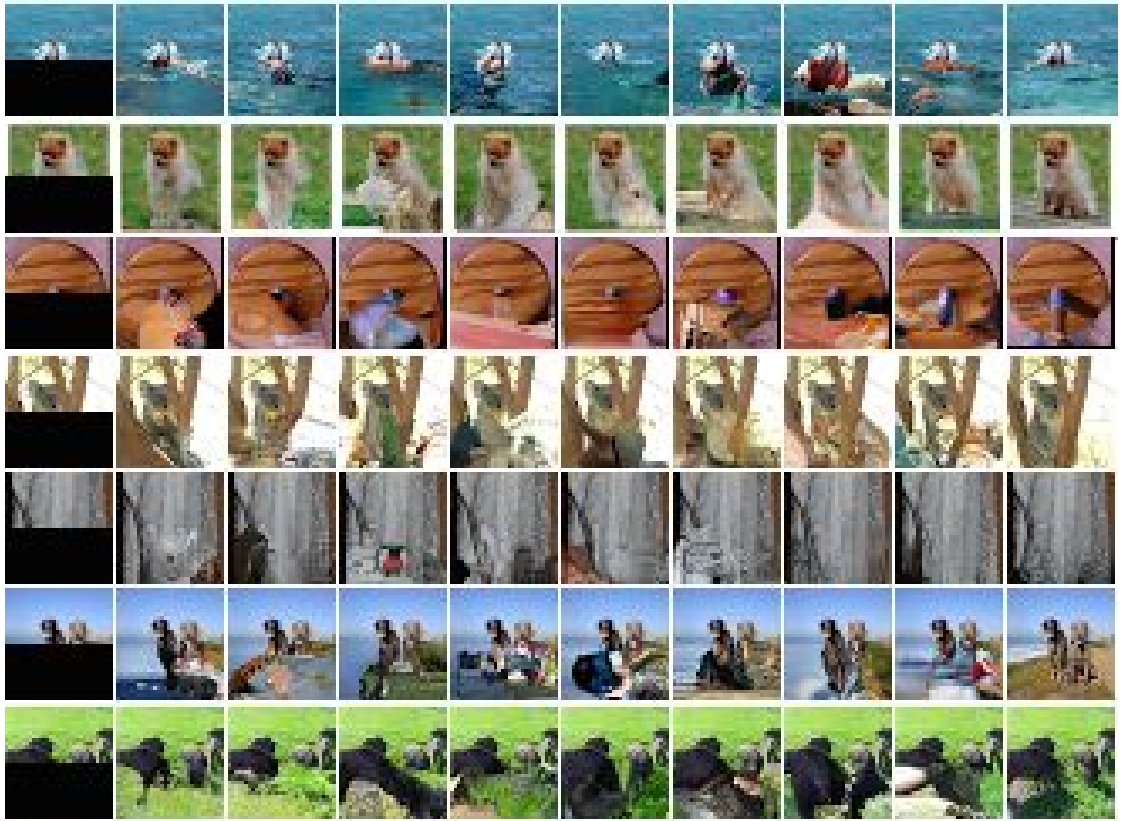
van den Oord, A., et al. “Pixel Recurrent Neural Networks.” *ICML (2016).*

occluded **completions**



van den Oord, A., et al. “Pixel Recurrent Neural Networks.” *ICML (2016).*

occluded completions original

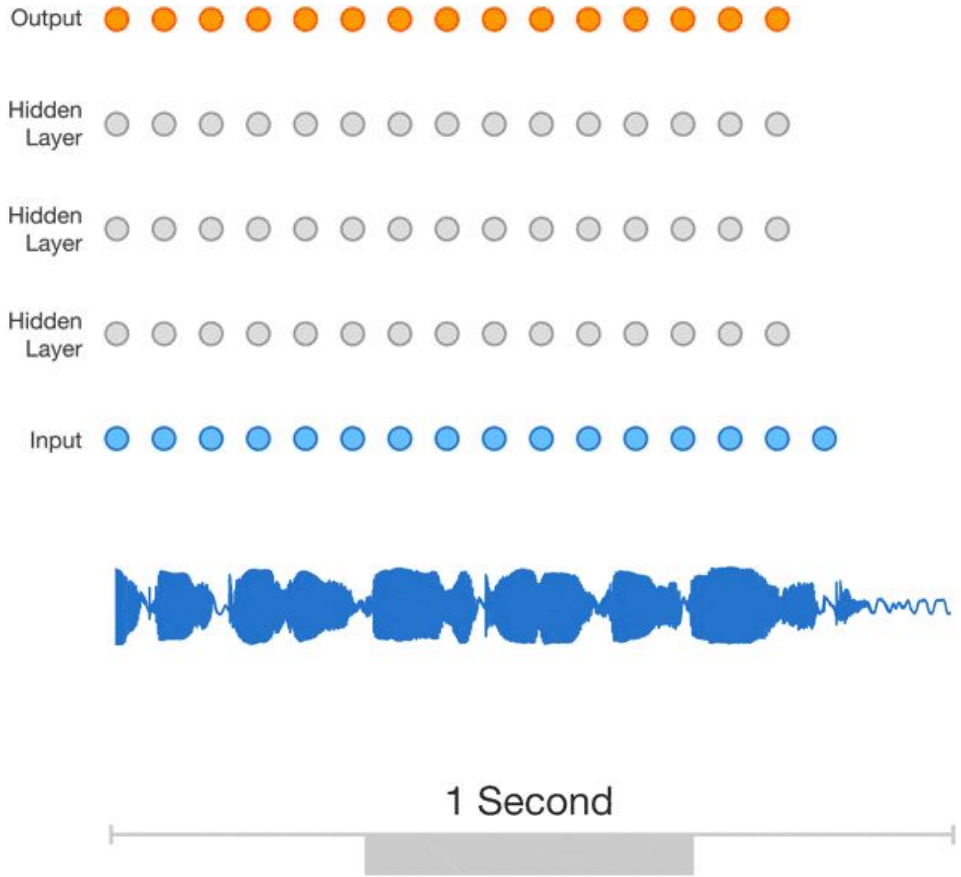


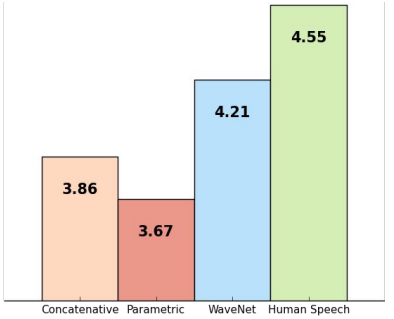
van den Oord, A., et al. “Pixel Recurrent Neural Networks.” *ICML (2016).*

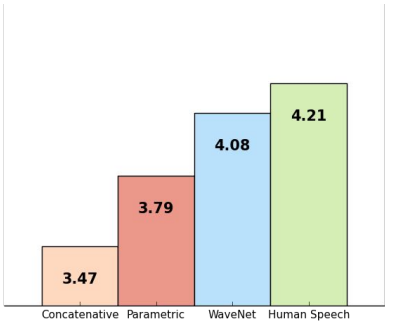
Conditional Pixel CNN

van den Oord, A., et al. "Conditional Pixel CNN." *NIPS (2016).*

WaveNets



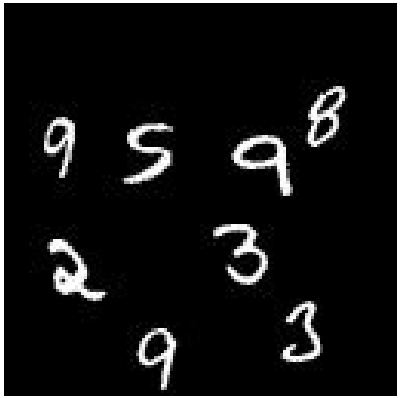


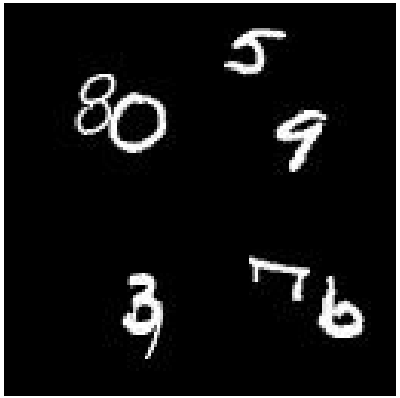


van den Oord, A., et al. “WaveNet: A Generative Model for Raw Audio.” *arxiv (2016).*

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



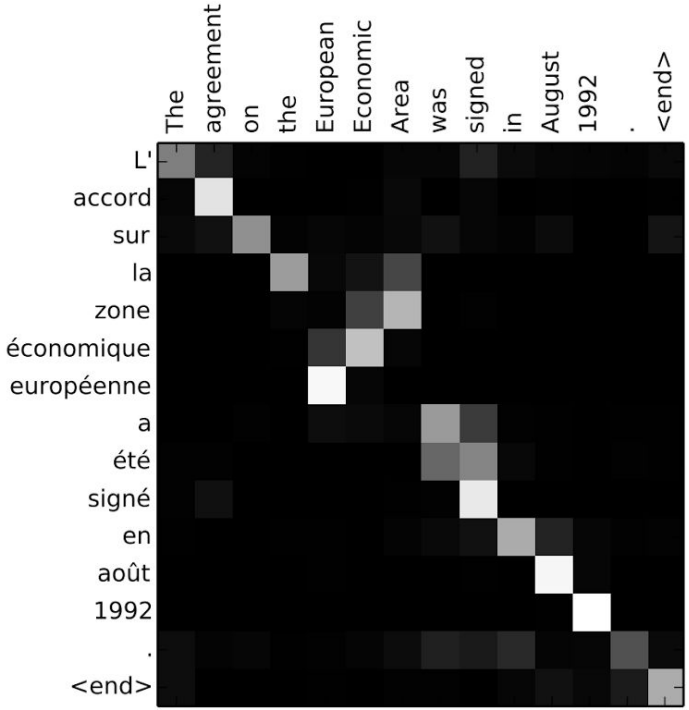


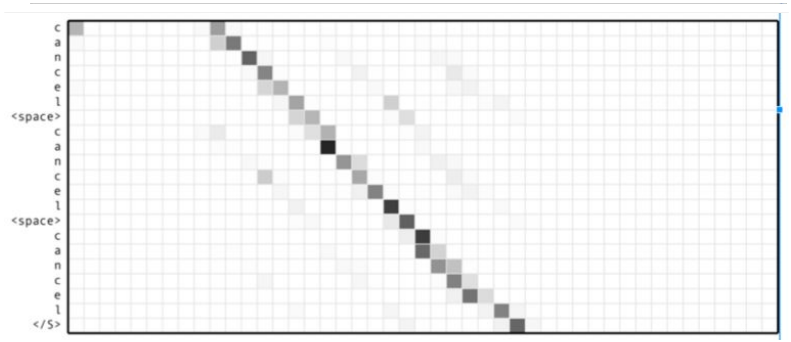
Kalchbrenner, N., et al. “Video Pixel Networks.” ***ICML (2017)***.

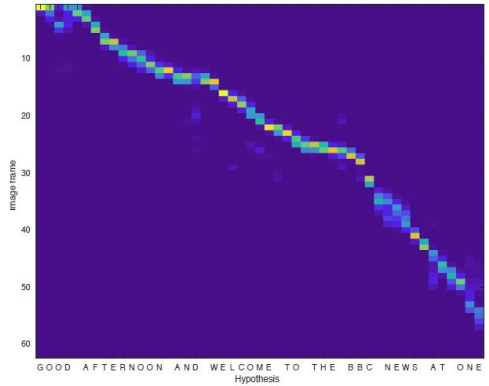
Online Seq2Seq

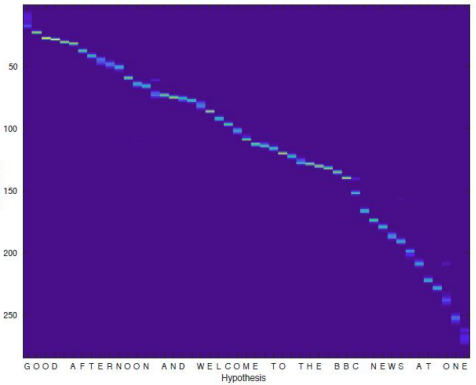
Practical limitations of Online Seq2Seq

- Have to wait for all of the input to arrive
- Yet, attention vector reveals a lot of globally monotonic structure







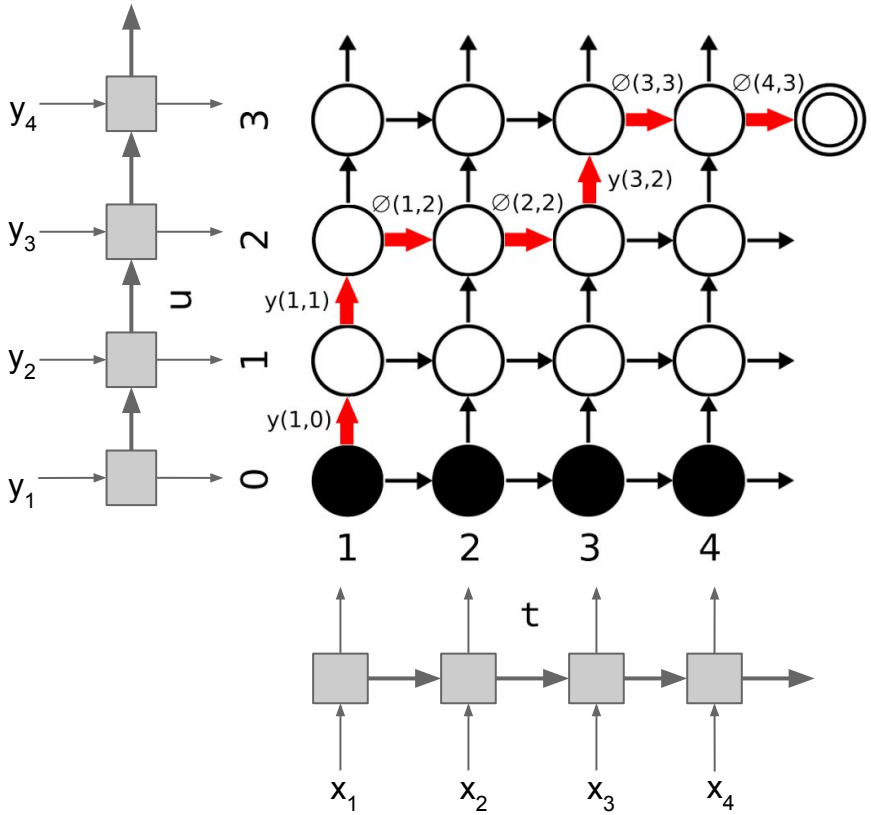


Online Sequence to Sequence Models

- Overcome limitations of sequence to sequence models
  - No need to wait for the entire input sequence to arrive
- Has to solve the following problem:
  - Has enough information arrived for the model to make its next prediction ?

- Yu, L., Buys, J., Blunsom, P. “Online Segment to Segment Neural Transduction.” *EMLP (2016).*
- Jaitly, N., et al. “An Online Sequence-to-Sequence Model Using Partial Conditioning.” *NIPS (2016).*
- Luo, Y., et al. “Learning online alignments with continuous rewards policy gradient.” *ICASSP (2017).*
- Gu, J, Neubig, G, Cho, K, Li, V.O.K.. “Learning to translate in real-time with neural machine translation”. *EACL (2017)*
- Raffel, Colin, et al. "Online and Linear-Time Attention by Enforcing Monotonic Alignments." *ICML 2017 (*Tue Aug 8th 01:30*)*

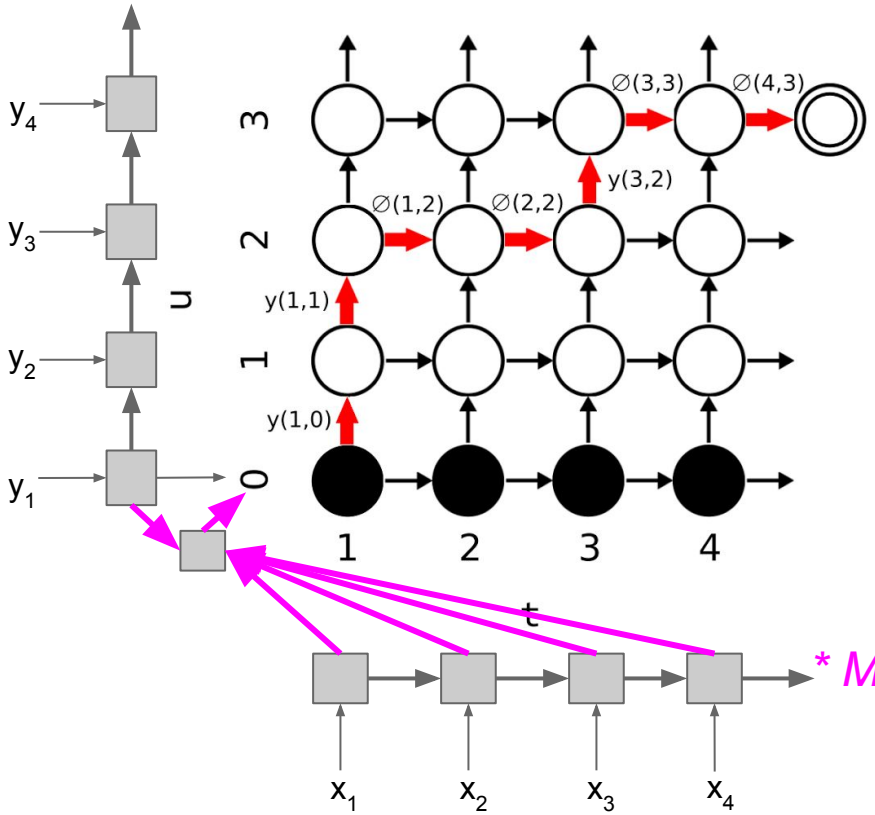
Sequence Transducer



- RNN over input (encoder)
- RNN over output (transducer)
- Right arrow consumes input without producing output
- Up arrow produces output without consuming input
- Edge probabilities computed from softmax that is a sum of logits from the encoder and the decoder

Graves, A. “Sequence transduction with recurrent neural networks.” *ICML (2012).*

Sequence Transducer vs Sequence-to-Sequence

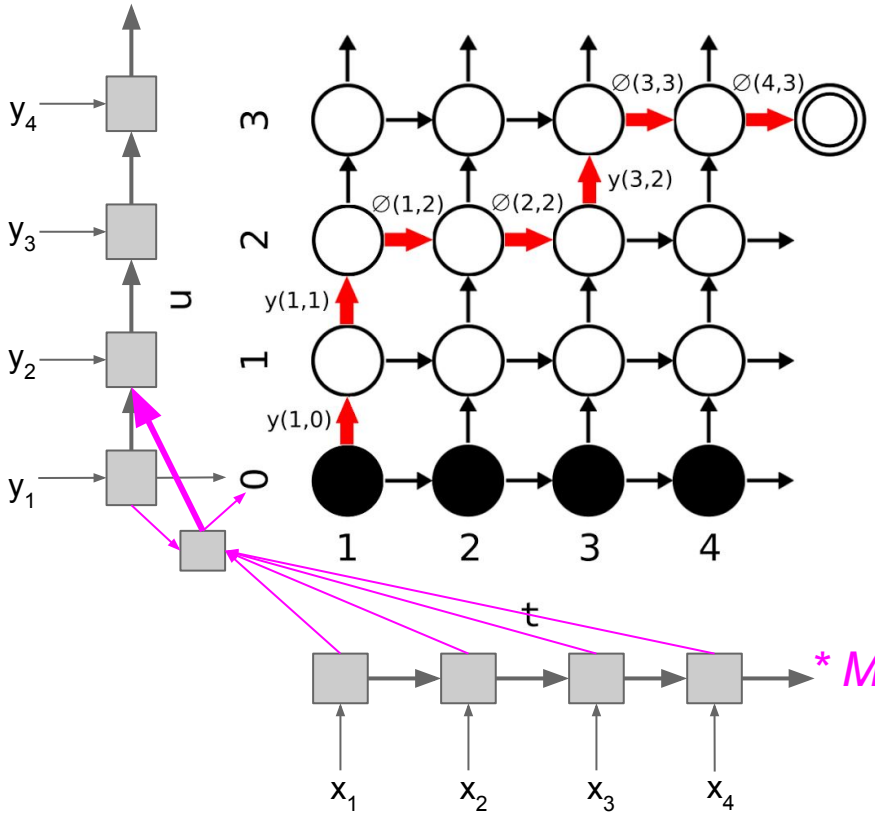


- Transducer step interacts with only one input step

** Missing connections in sequence transducer*

Graves, A. “Sequence transduction with recurrent neural networks.” *ICML (2012).*

Sequence Transducer vs Sequence-to-Sequence

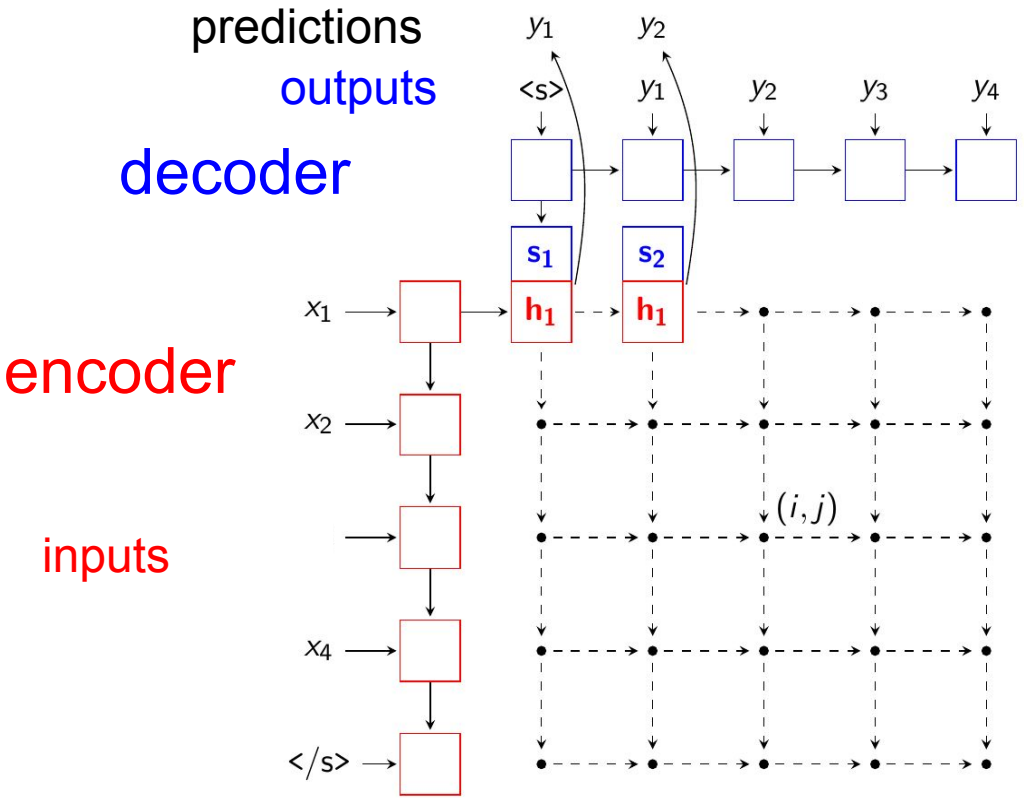


- Transducer output interacts with only one input step
- Transducer states do not depend on path history because context vector connection is missing

** Missing connections in sequence transducer*

Graves, A. “Sequence transduction with recurrent neural networks.” *ICML (2012).*

Online Segment to Segment Neural Transduction



Yu, L., Buys, J., Blunsom, P. "Online Segment to Segment Neural Transduction." *EMLP (2016).*

Online Segment to Segment Neural Transduction

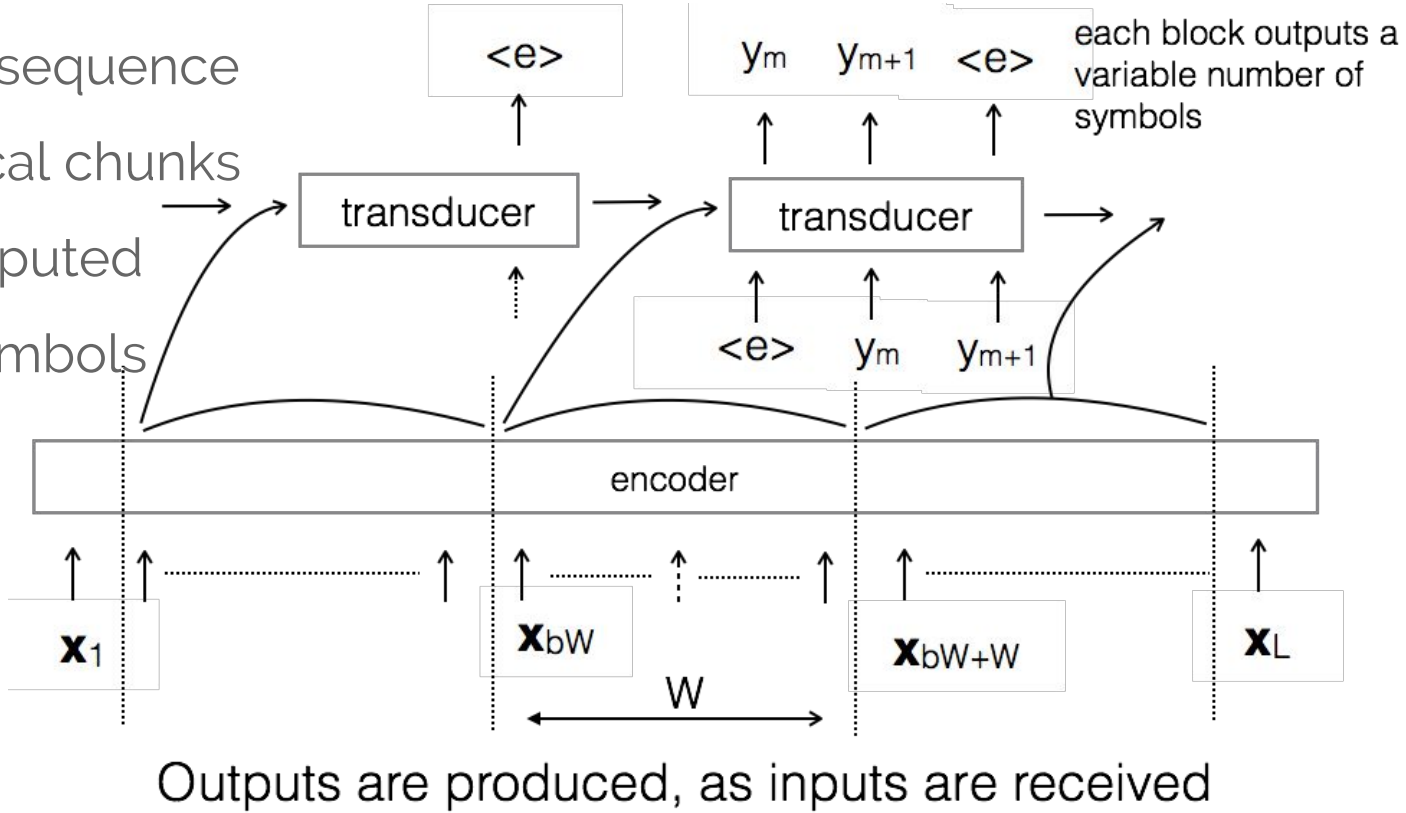
Model	ROUGE-1	ROUGE-2	ROUGE-L
Seq2Seq	25.2	9.1	23.1
Seq2Seq + attention	30.3	14.2	28.5
Neural Transduction	31.2	14.3	28.9
Neural Transduction + bidirectional encoder	31.7	14.7	29.3

F1 values on Abstractive sentence summarisation task

1. Yu, L., Buys, J., Blunsom, P. “Online Segment to Segment Neural Transduction.” *EMLP (2016).*
2. Yu, L., et al. “The Neural Noisy Channel.” *ICLR (2017).*

A Neural Transducer

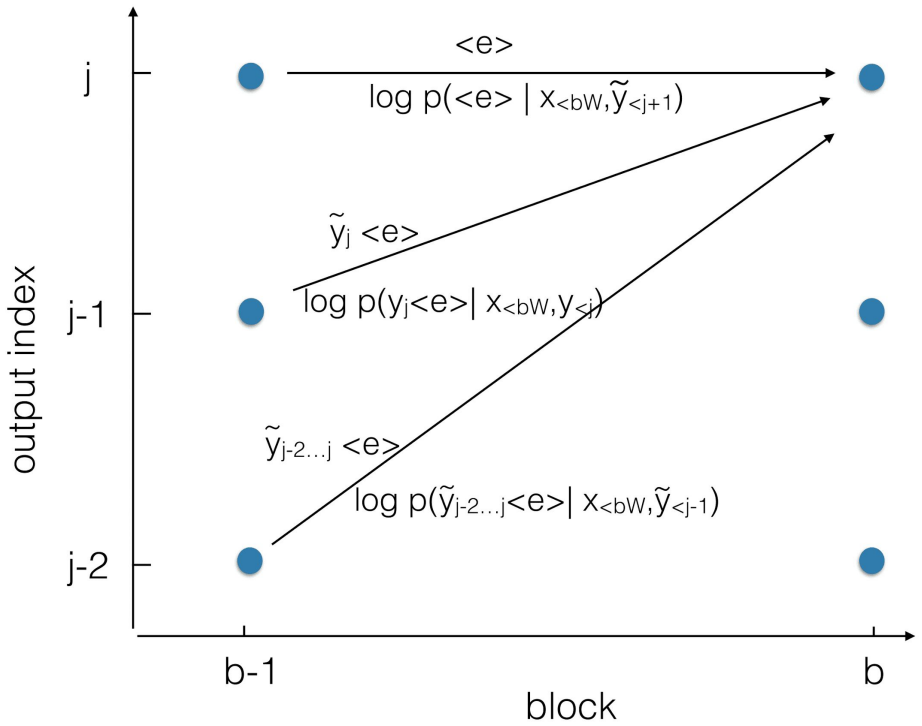
- sequence-to-sequence models on local chunks
- Features computed depend on symbols output so far
- Introduces an alignment problem



Jaitly, N., et al. “An Online Sequence-to-Sequence Model Using Partial Conditioning.” *NIPS (2016).*

A Neural Transducer - Viterbi-like training

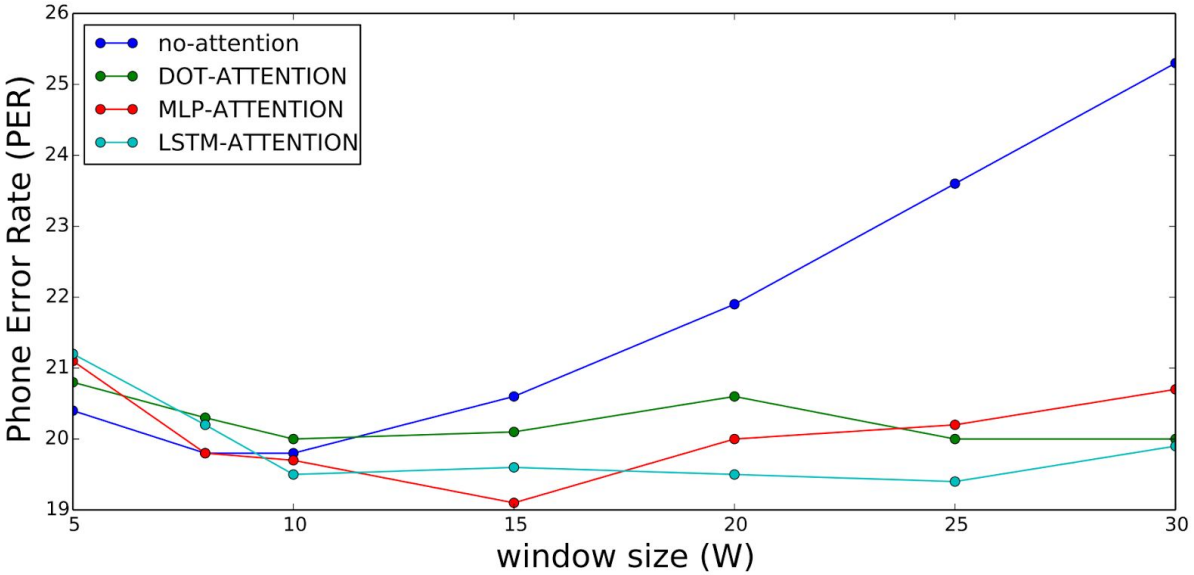
- Naive beam search for Viterbi training fails easily because of corner cases
- Approximate Dynamic programming for finding best alignment works quite well



Jaitly, N., et al. “An Online Sequence-to-Sequence Model Using Partial Conditioning.” *NIPS (2016).*

A Neural Transducer - Results

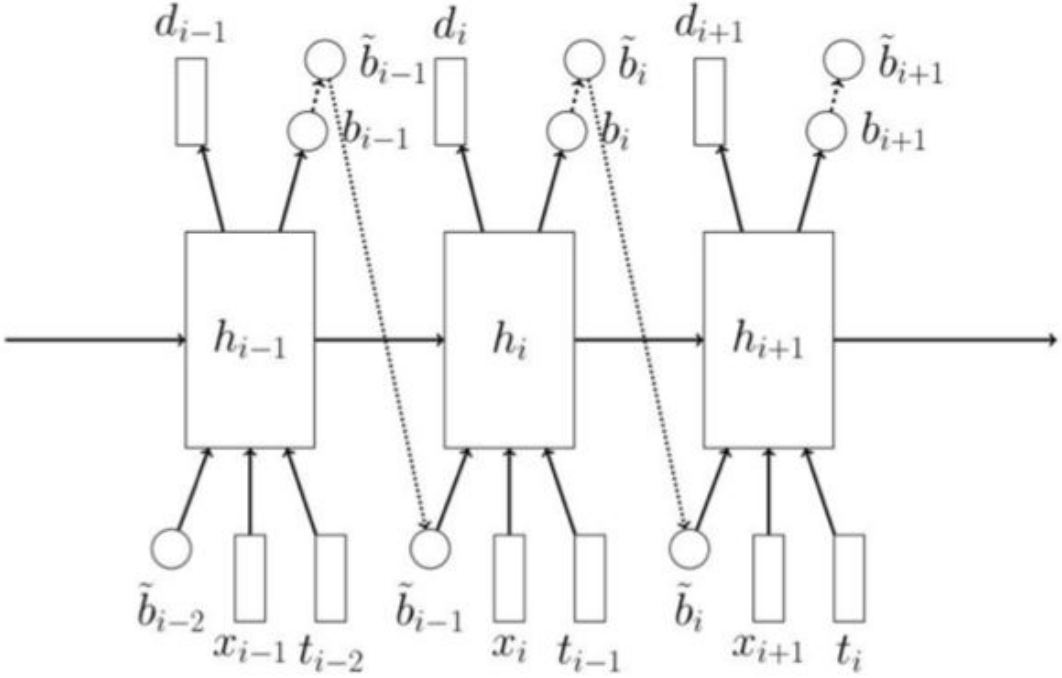
# of layers in encoder / transducer	1	2	3	4
2		19.2	18.9	18.8
3		18.5	**18.2**	19.4



Jaitly, N., et al. “An Online Sequence-to-Sequence Model Using Partial Conditioning.” *NIPS (2016).*

A Stochastic Emissions Model

- Autoregressive at every step

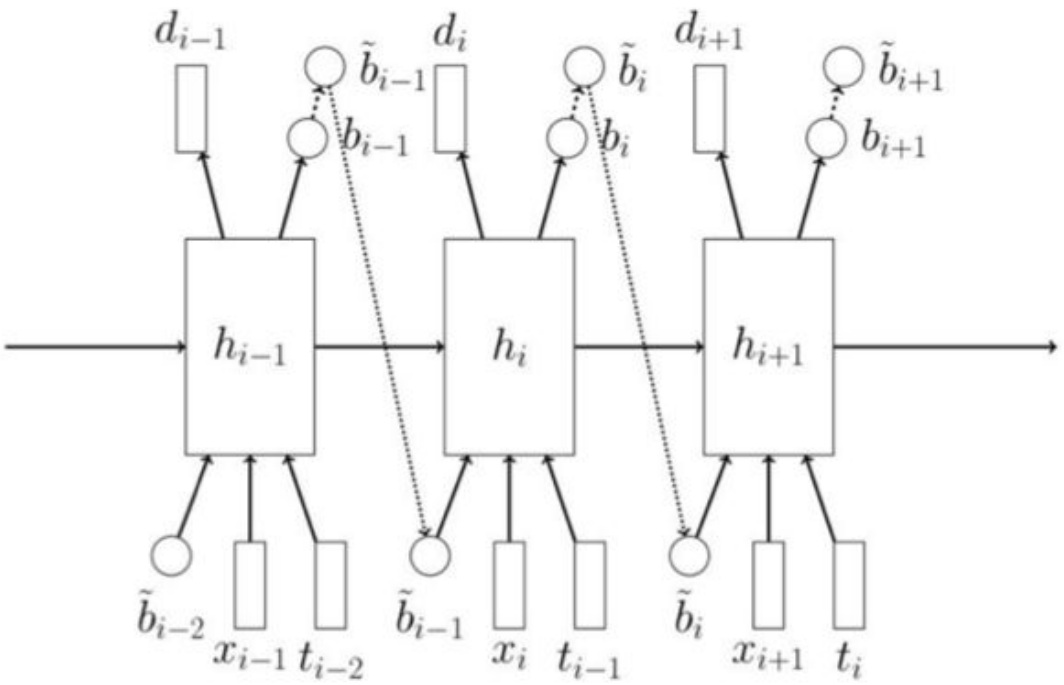


d	distributions over phoneme
b	probabilities to emit
$\tilde{b}$	samplings of *Bernoulli(b)*
h	states of RN
x	input
t	last true label

Luo, Y., et al. “Learning online alignments with continuous rewards policy gradient.” *ICASSP (2017).*

A Stochastic Emissions Model

- Continuous Reward $R_i = \tilde{b}_i \log p(d_i = t_i | \mathbf{x}_{\leq i}, \tilde{\mathbf{b}}_{<i}, \mathbf{t}_{<i})$

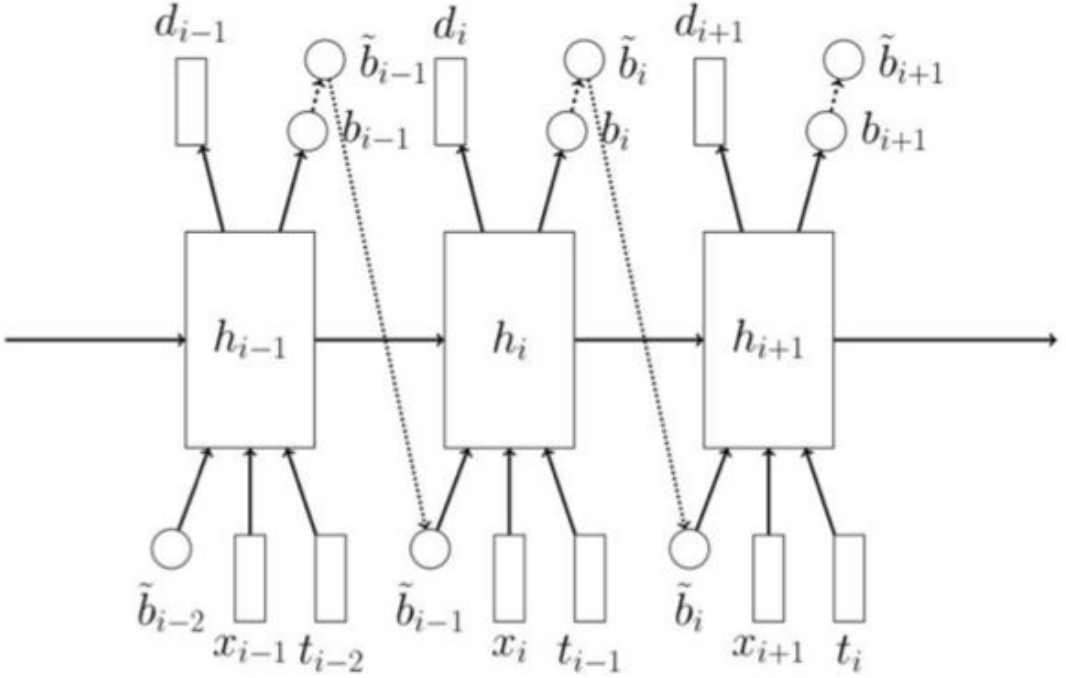


d	distributions over phoneme
b	probabilities to emit
$\tilde{b}$	samplings of *Bernoulli(b)*
h	states of RN
x	input
t	last true label

Luo, Y., et al. “Learning online alignments with continuous rewards policy gradient.” *ICASSP (2017).*

A Stochastic Emissions Model

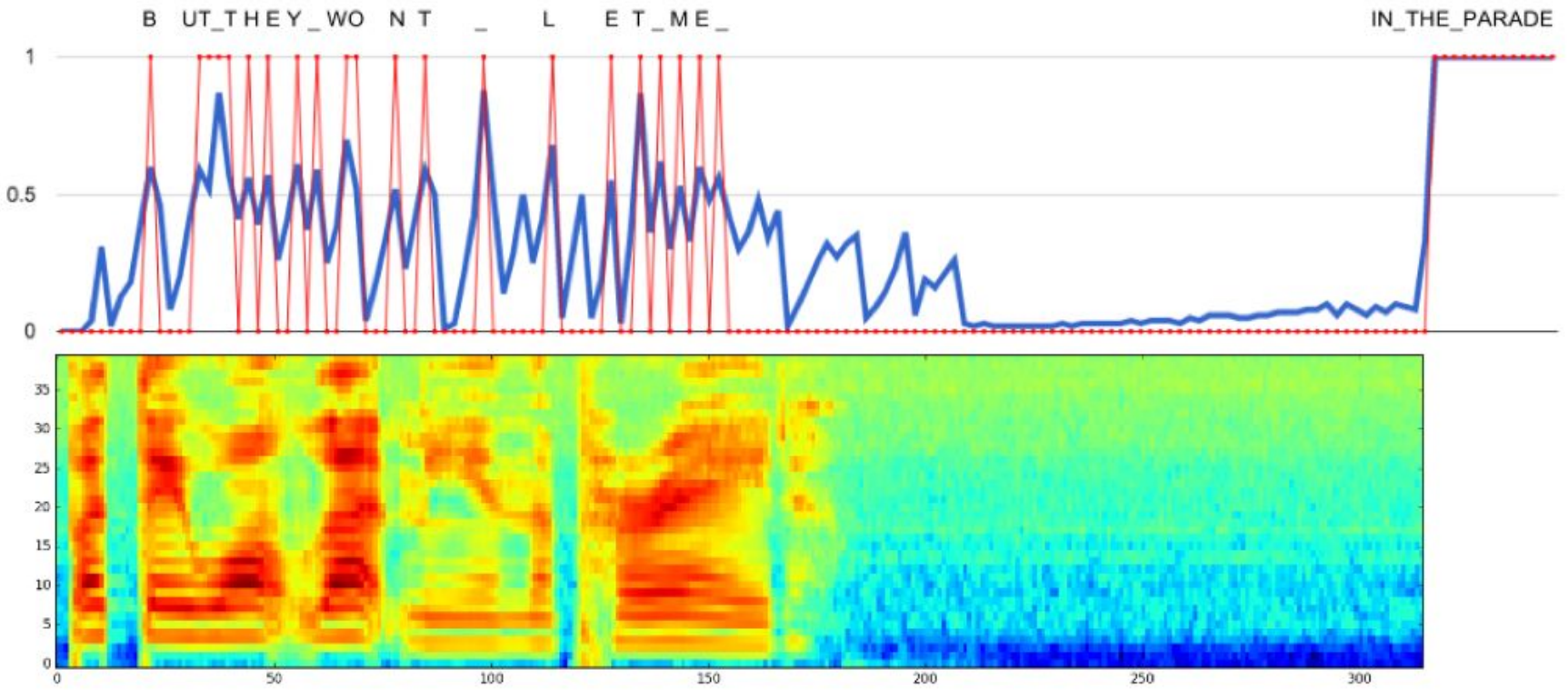
- Optimize for expected reward: $\mathcal{R} = \mathbb{E}_{\tilde{\mathbf{b}}}\left[R(\tilde{\mathbf{b}})\right]$



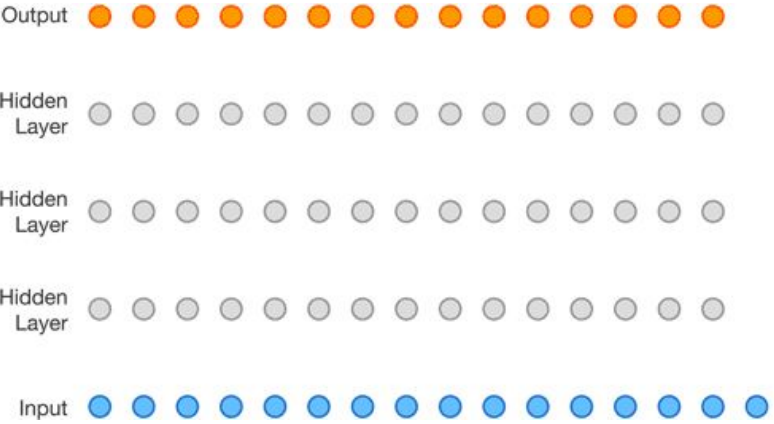
d	distributions over phoneme
b	probabilities to emit
$\tilde{b}$	samplings of *Bernoulli(b)*
h	states of RN
x	input
t	last true label

Luo, Y., et al. “Learning online alignments with continuous rewards policy gradient.” *ICASSP (2017).*

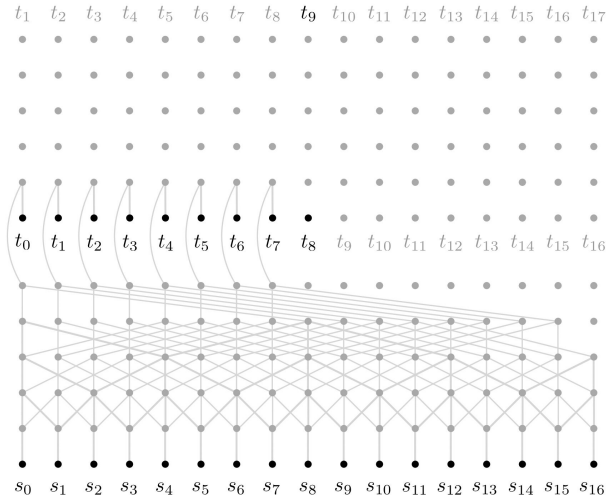
A Stochastic Emissions Model



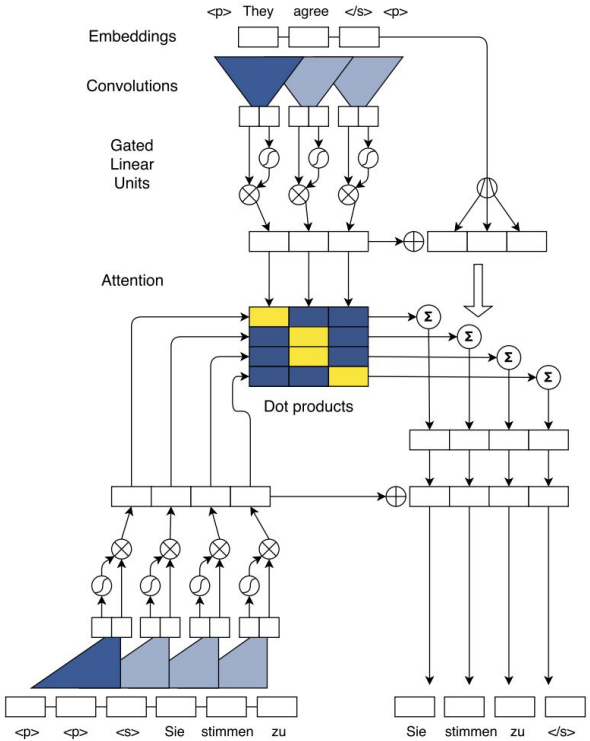
New Architectures



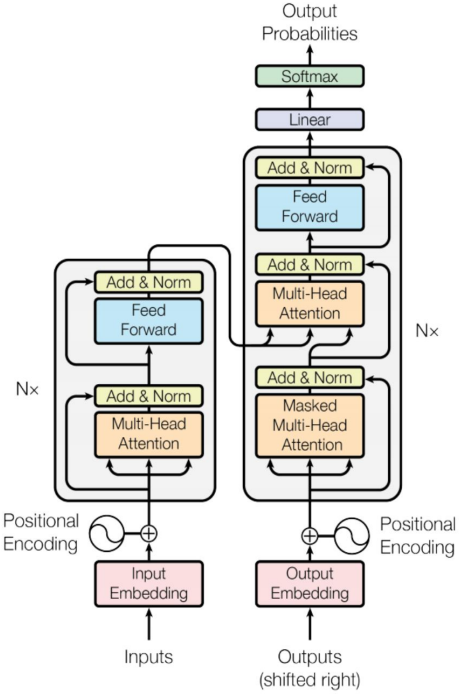
Wavenet, van den Oord, et al, 2016



Bytenet, Kalchbrenner, et al, 2016



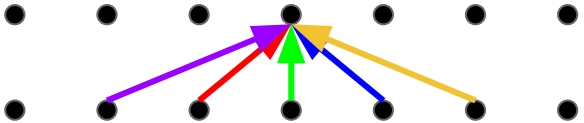
Conv seq2seq, Gehring, et al, 2017



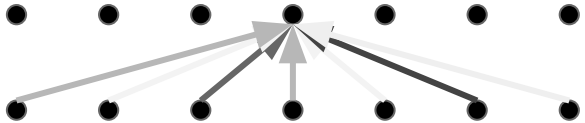
Att is all you need, Vaswani, et al, 2017

Self-Attention

Convolution

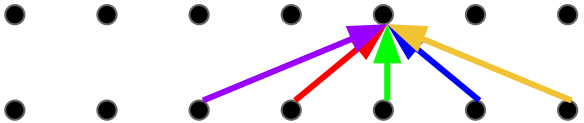


Self-Attention

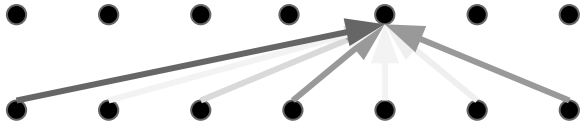


Self-Attention

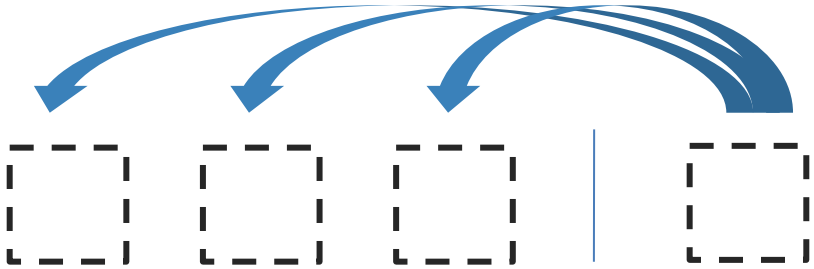
Convolution



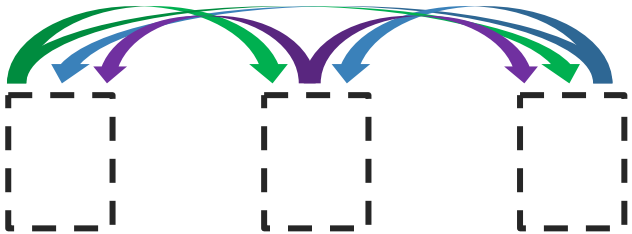
Self-Attention



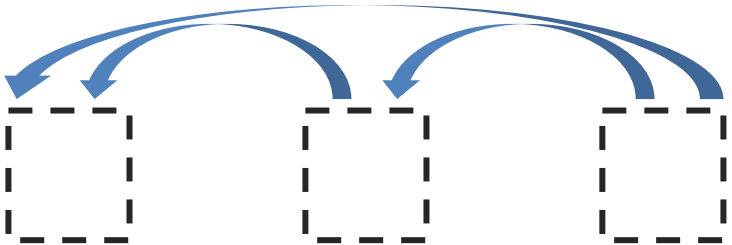
Three ways of attention



Encoder-Decoder Attention



Encoder Self-Attention

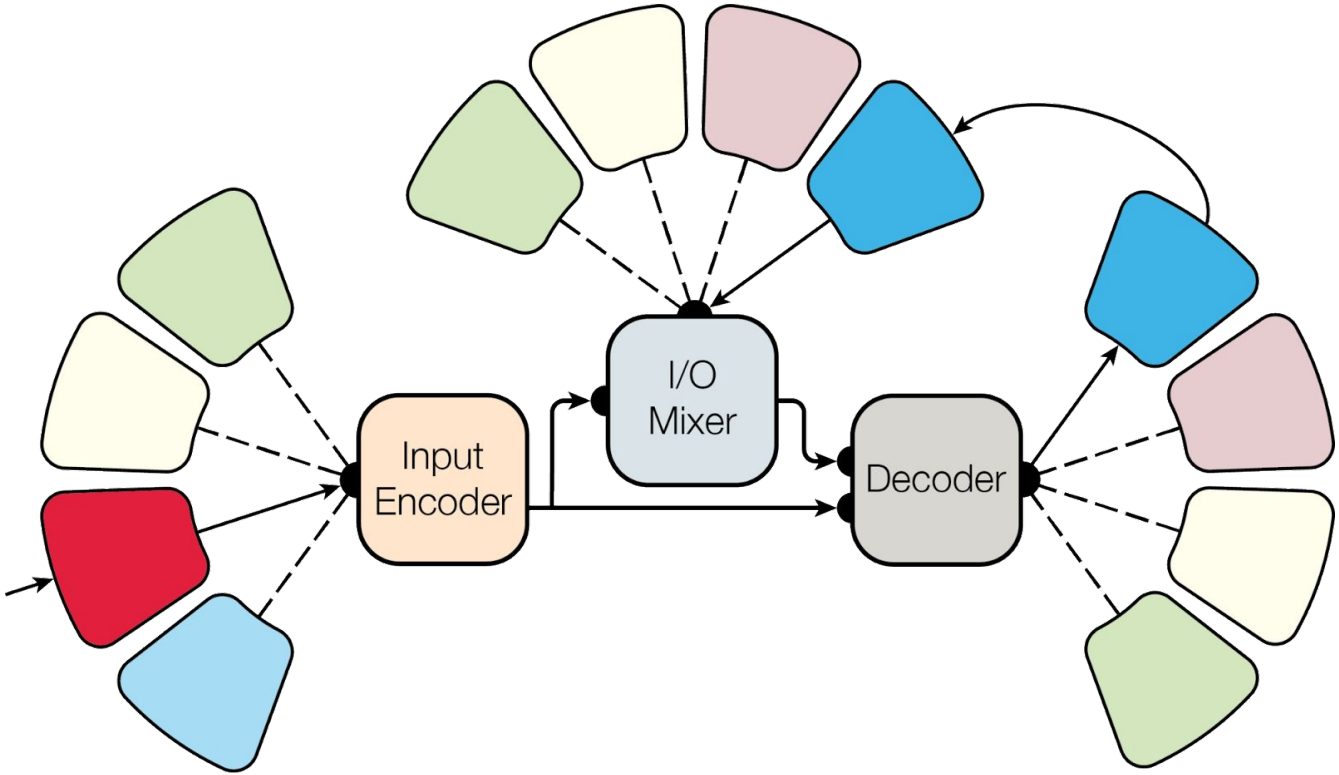


MaskedDecoder Self-Attention

Machine Translation Results: WMT-14

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [17]	23.75			
Deep-Att + PosUnk [37]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [36]	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 [31]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [37]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [36]	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**	**41.0**	$2.3 \cdot 10^{19}$	

MultiModel



Kaiser, L., et al. “One Model To Learn Them All.” *arxiv (2017).*

MultiModel

To English

"A man that is sitting in front of a suitcase"

To Category

Category 127 (Male Human)

"Last week, Kigali raised the possibility of military retaliation after shells..."

To French

"La semaine dernière, Kigali a soulevé la possibilité de représailles militaires aprés avoir débarqué des coquilles..."

"Can you give our readers some details on this?"

To German

"Können Sie unseren Lesern einige Details dazu geben?"

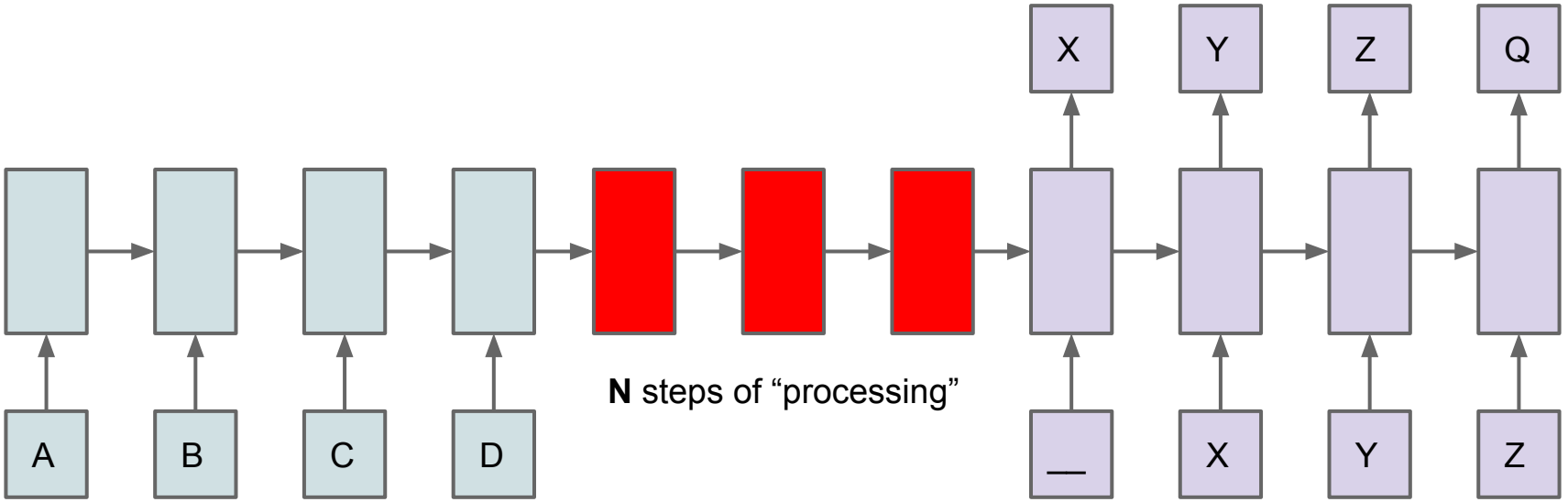
The above represents a triumph of either apathy or civility

To Parse

"S NP DT JJS /NP VP VBZ NP NP DT NN /NP PP IN NP NP NN /NP CC NP NN /NP /NP /PP /NP /VP . /S"

Kaiser, L., et al. "One Model To Learn Them All." *arxiv (2017).*

Processing Depth



1. Vinyals, O., et al. “Order Matters: Sequence to Sequence for sets.” *ICLR (2015).*
2. Sukhbaatar, S., et al. “End-to-End Memory Networks.” *NIPS (2015).*

How to pick N? Learn it of course!

Slide Credit:
Alex Graves

Add a *halting unit h* to the output

$$h_t^n = \sigma\left(W_h s_t^n + b_h\right)$$

Use this to define the *halt probability* p_t^n at ponder step *n*

$$p_t^n = \begin{cases} R(t) \text{ if } n = N(t) \\ h_t^n \text{ otherwise} \end{cases}$$

Where *N(t)* is # updates at *t*

$$N(t) = \min\{n' : \sum_{n=1}^{n'} h_t^n >= 1 - \epsilon\}$$

And *R(t)* is the *remainder* at *t*

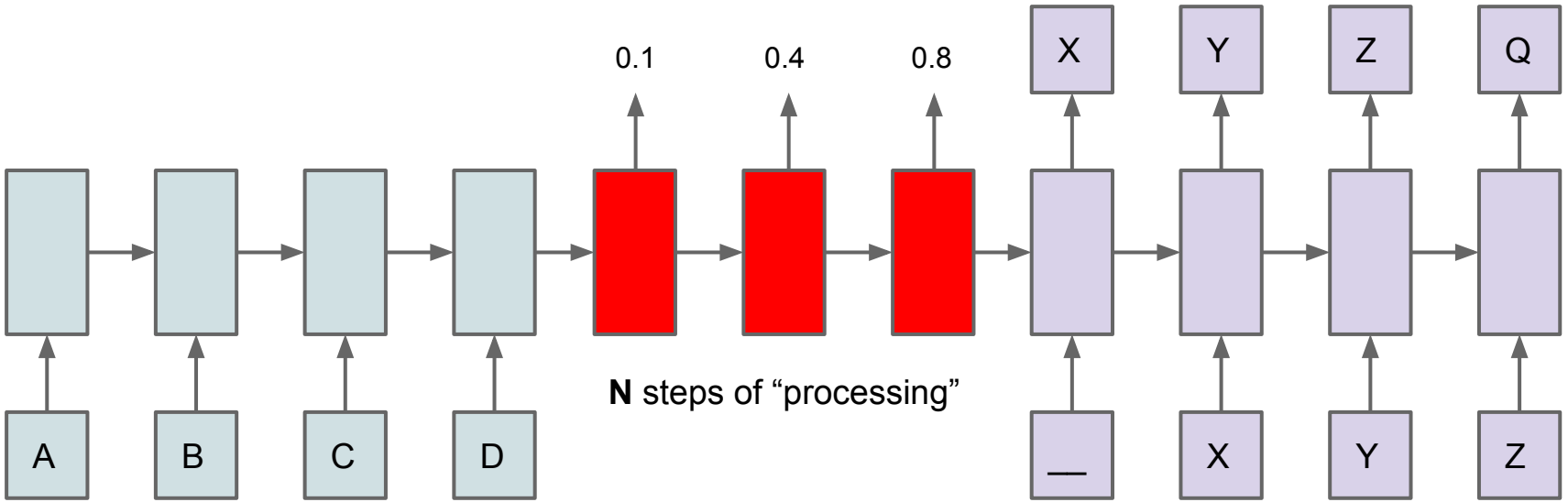
$$R(t) = 1 - \sum_{n=1}^{N(t)-1} h_t^n$$

The final states and outputs at *t* are weighted sums (!)

$$s_t = \sum_{n=1}^{N(t)} p_t^n s_t^n \qquad y_t = \sum_{n=1}^{N(t)} p_t^n y_t^n$$

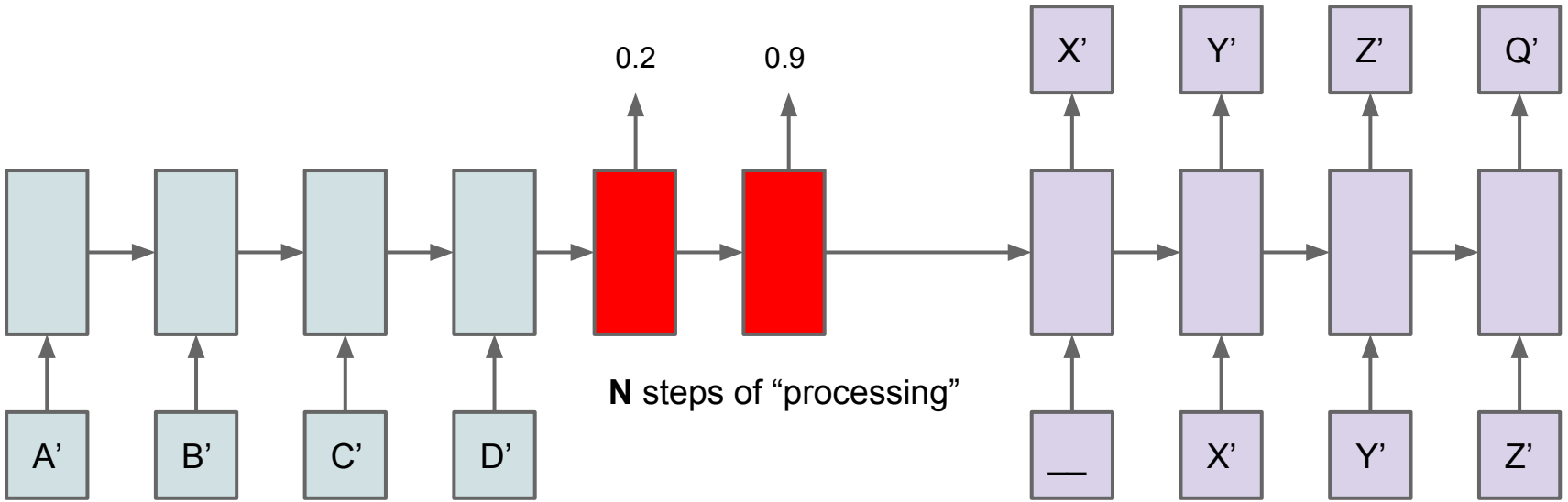
Graves, A. “Adaptive Computation Time for Recurrent Neural Networks.” *arxiv (2016).*

Adaptive Computation Time (ACT)



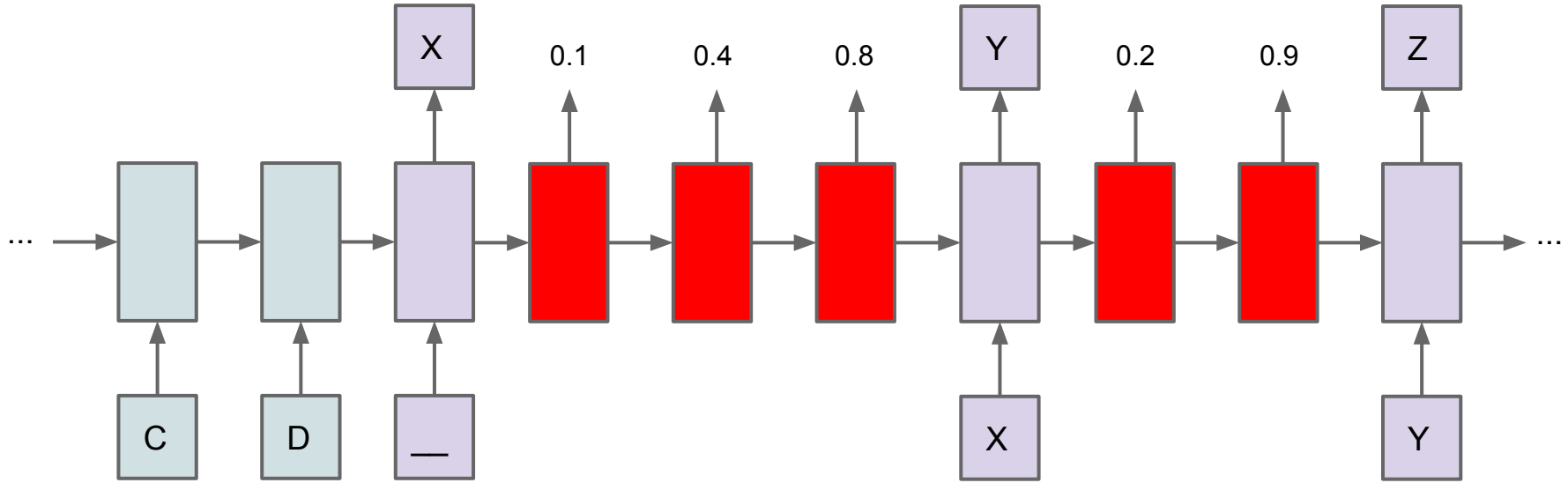
Graves, A. “Adaptive Computation Time for Recurrent Neural Networks.” *arxiv (2016).*

Adaptive Computation Time (ACT) [Graves, 2016]



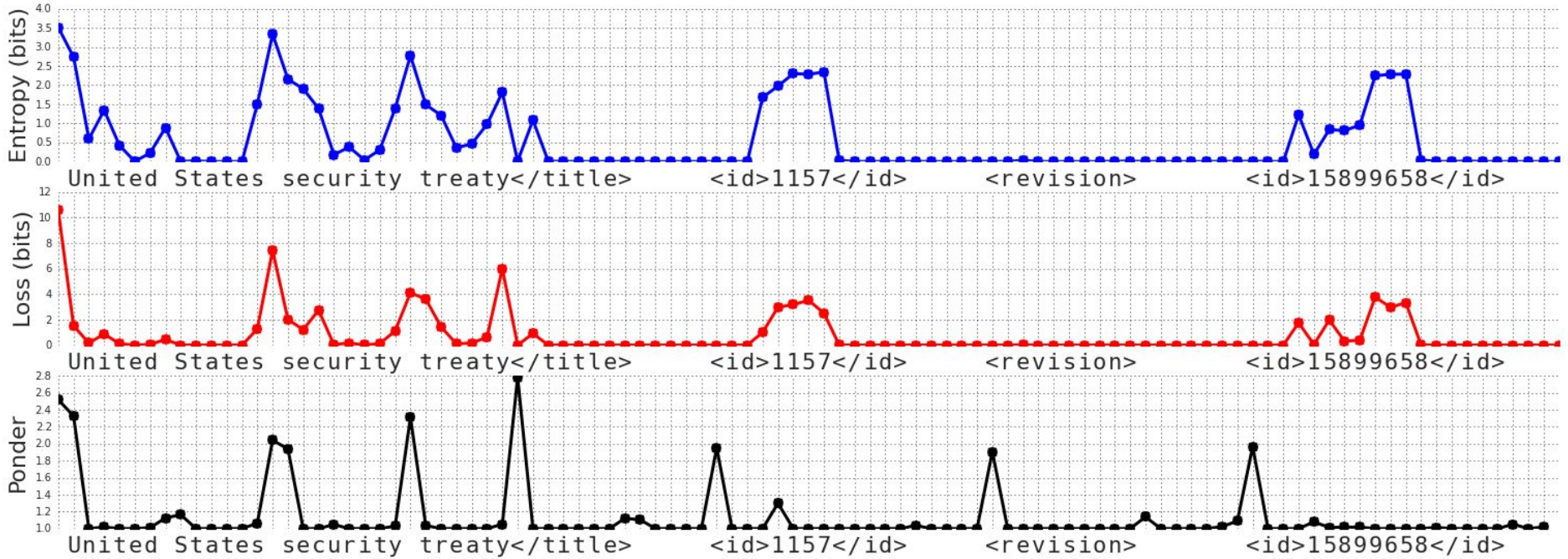
Graves, A. “Adaptive Computation Time for Recurrent Neural Networks.” *arxiv (2016).*

(in)ACT [Graves, 2016]



Graves, A. “Adaptive Computation Time for Recurrent Neural Networks.” *arxiv (2016).*

Character level LM



Graves, A. “Adaptive Computation Time for Recurrent Neural Networks.” *arxiv (2016).*

The End / Questions

Oriol Vinyals and Navdeep Jaitly

@OriolVinyalsML | @NavdeepLearning

Site: https://sites.google.com/view/seq2seq-icml17