WELCOME TO THE

UCL x DeepMind lecture series

In this lecture series, leading research scientists from leading AI research lab, DeepMind, will give 12 lectures on an exciting selection of topics in Deep Learning, ranging from the fundamentals of training neural networks via advanced ideas around memory, attention, and generative modelling to the important topic of responsible innovation.

Please join us for a deep dive lecture series into Deep Learning!

#UCLxDeepMind

General information

Exits:

At the back, the way you came in

Wifi:

UCL guest

TODAY'S SPEAKER

Marta Garnelo

Marta is a research scientist at DeepMind working on deep generative models and meta learning. During her time at DM she has worked on Generative Query Networks as well as Neural Processes and recently her research focus has shifted towards multi-agent systems. In addition she is currently wrapping up her PhD with Prof Murray Shanahan at Imperial College London where she also did an MSc in Machine Learning.

TODAY'S LECTURE

Sequences and Recurrent Networks

In this lecture we will focus on sequential data and how machine learning methods have been adapted to process this particular type of structure. We will start by introducing some fundamentals of sequence modeling including common architectures designed for this task such as RNNs and LSTMs. We will then move on to sequence-to-sequence decoding and its applications before finishing with some examples of recent applications of sequence models.

DeepMind

Sequences and Recurrent Neural Networks

Marta Garnelo

05/03/2020

Roadmap

1

Motivation

Why sequences matter and why the methods we have covered so far don't work on them.

2

Fundamentals

Loss, optimisation and architectures of sequence models.

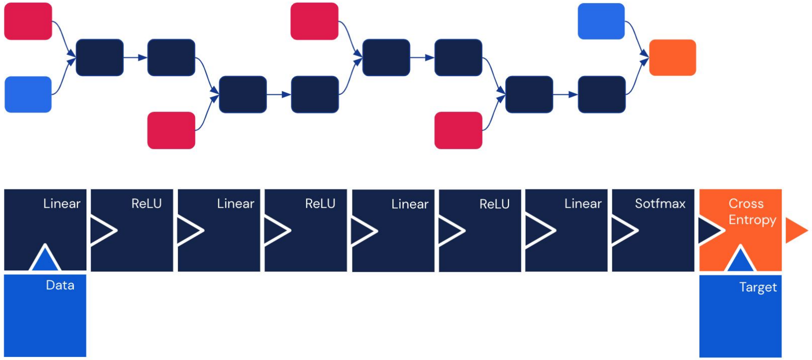
3

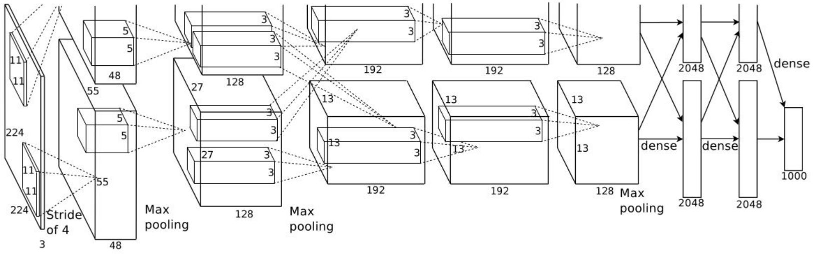
Generation

Applications and examples of sequence modelling.

1 Motivation

So far

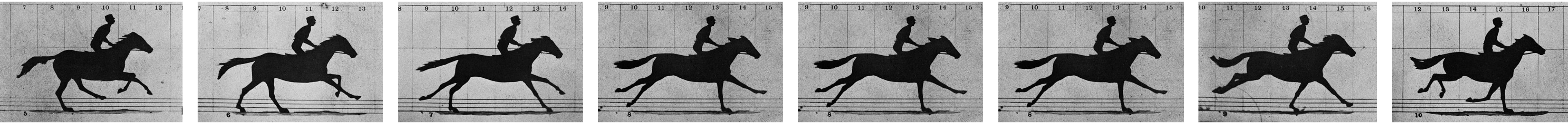




Feed forward networks

Convolutional networks

Sequences



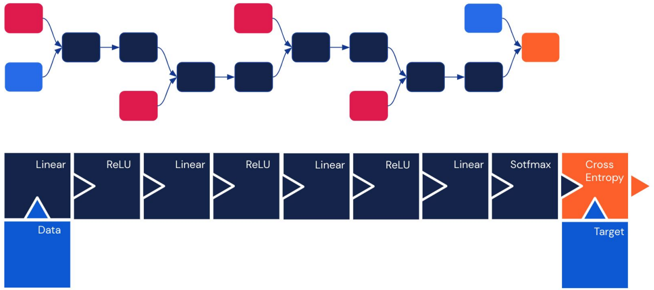
Collections of elements where:

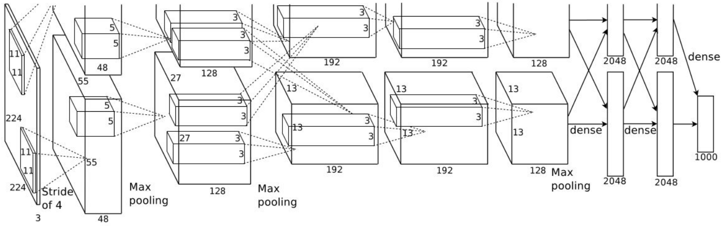
- Elements can be **repeated**
- **Order** matters
- Of **variable** (potentially infinite) length

Modeling sequences

- Elements can be **repeated**
- **Order** matters
- Of **variable** (potentially infinite) length

Models discussed so far don't do well with sequential data.





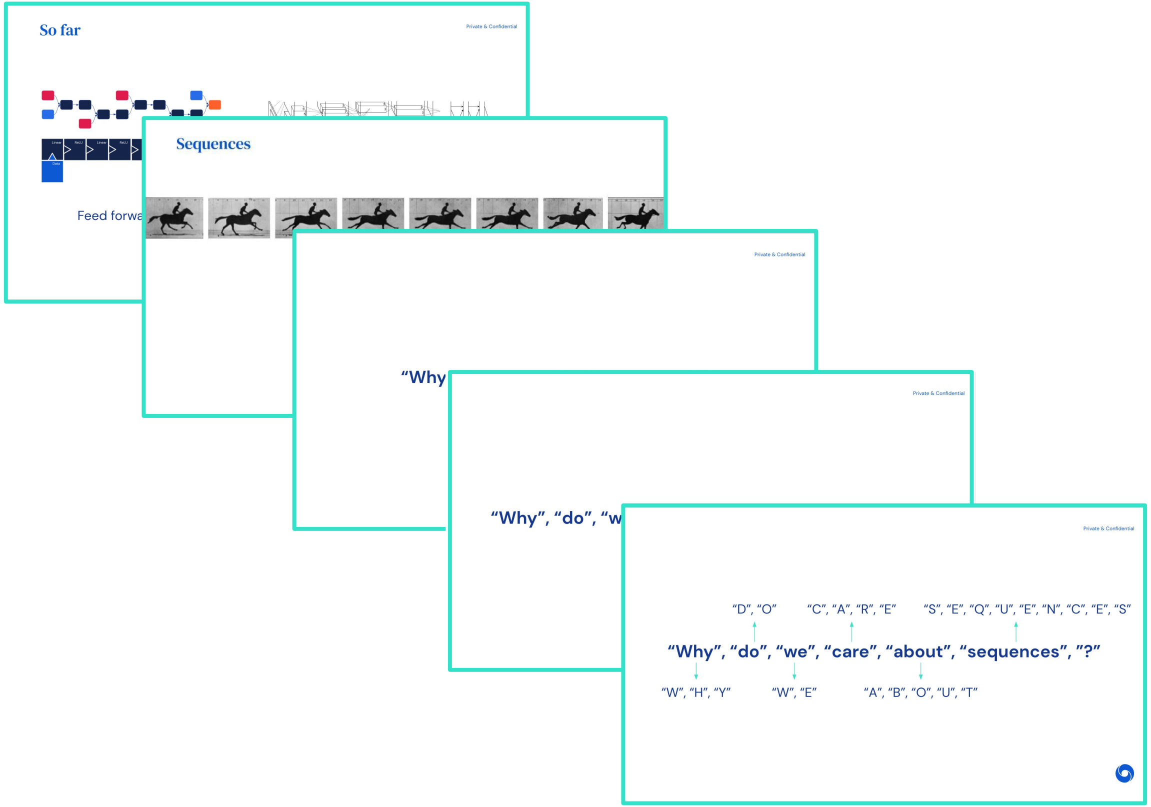
"Why do we care about sequences?"

“Why”, “do”, “we”, “care”, “about”, “sequences”, ”?”

“D”, “O”   “C”, “A”, “R”, “E”   “S”, “E”, “Q”, “U”, “E”, “N”, “C”, “E”, “S”

“Why”, “do”, “we”, “care”, “about”, “sequences”, ”?”

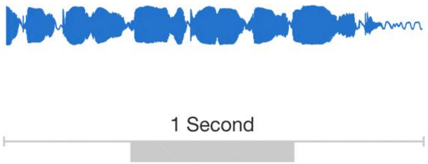
“W”, “H”, “Y”   “W”, “E”   “A”, “B”, “O”, “U”, “T”


So far
Private & Confidential
Feed forwa
Sequences
Private & Confidential
“Why
Private & Confidential
“Why”, “do”, “w
Private & Confidential
“D”, “O”
“C”, “A”, “R”, “E”
“S”, “E”, “Q”, “U”, “E”, “N”, “C”, “E”, “S”
“Why”, “do”, “we”, “care”, “about”, “sequences”, “?”
“W”, “H”, “Y”
“W”, “E”
“A”, “B”, “O”, “U”, “T”

Sequences are everywhere

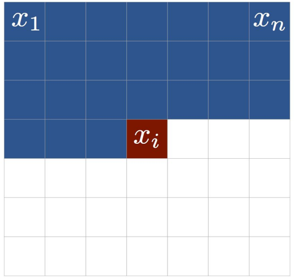
"Sequences really seem to be everywhere! We should learn how to model them. What is the best way to do that? Stay tuned!"

Words, letters



Speech

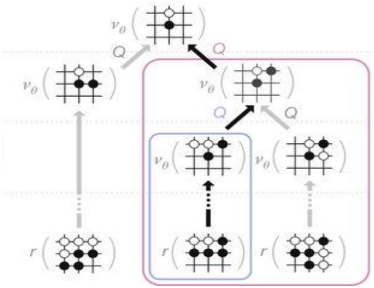
Videos



Images

```python
import numpy as np
import matplotlib.pyplot as plt

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

  dh = 1
  for t in range(T):
    dh = (1-hs[-1-t] ** 2) * w * dh

  return hs[-1], dh

T = 10  # sequence length
wlim = 4  #limit of interval over weights w

results = []
ws = np.linspace(-wlim, wlim, 1000)
for w in ws:
  results.append(forward_backward_prop(w, T))

plt.plot(ws, [r[0] for r in results], label='RNN state')
plt.plot(ws, [r[1] for r in results], label='Gradients')
```

Programs



Decision making

Summary

Sequences are collections of **variable length** where **order matters**

Sequences are widespread across machine learning applications

Not all deep learning models can handle sequential data

DeepMind

2 Fundamentals

Training machine learning models

Supervised learning

Data $$\{x, y\}_i$$

Model $$y \approx f_\theta(x)$$

Loss $$\mathcal{L}(\theta) = \sum_{i=1}^{N} l(f_\theta(x_i), y_i)$$

Optimisation $$\theta^* = \arg\min_\theta \mathcal{L}(\theta)$$

Training machine learning models

	Supervised learning	Sequence modelling
Data	$\{x, y\}_i$	$\{x\}_i$
Model	$y \approx f_\theta(x)$	$p(x) \approx f_\theta(x)$
Loss	$\mathcal{L}(\theta) = \sum_{i=1}^{N} l(f_\theta(x_i), y_i)$	$\mathcal{L}(\theta) = \sum_{i=1}^{N} \log p(f_\theta(x_i))$
Optimisation	$\theta^* = \arg\min_\theta \mathcal{L}(\theta)$	$\theta^* = \arg\max_\theta \mathcal{L}(\theta)$

Modeling p(x)

"Modeling word probabilities is really difficult"

Modeling p(x)

Simplest model:
Assume independence of words

$$p(\mathbf{x}) = \prod_{t=1}^{T} p(x_t)$$

p(“modeling“) × p(“word“) × p(“probabilities“) × p(“is“) × p(“really“) × p(“difficult“)

Word	$p(x_i)$
the	0.049
be	0.028
...	...
really	0.0005
...	...

Modeling p(x)

Simplest model:
Assume independence of words

$$p(\mathbf{x}) = \prod_{t=1}^{T} p(x_t)$$

p(“modeling“) × p(“word“) × p(“probabilities“) × p(“is“) × p(“really“) × p(“difficult“)

Word	$p(x_i)$
the	0.049
be	0.028
...	...
really	0.0005
...	...

However:

Most likely 6-word sentence:

“The the the the the the.”

→ Independence assumption does not match sequential structure of language.

Modeling p(x)

More realistic model:

Assume conditional dependence of words

$$p(x_T) = p(x_T | x_1, \ldots, x_{T-1})$$

Modeling word probabilities is really ?

Context

Target

Target	p(x|context)
difficult	0.01
hard	0.009
fun	0.005
...	...
easy	0.00001

Modeling p(x)

The chain rule

Computing the joint p(**x**) from conditionals

$$p(\mathbf{x}) = \prod_{t=1}^{T} p(x_t | x_1, \ldots, x_{t-1})$$

Modeling — $p(x_1)$

Modeling **word** — $p(x_2 | x_1)$

Modeling word **probabilities** — $p(x_3 | x_2, x_1)$

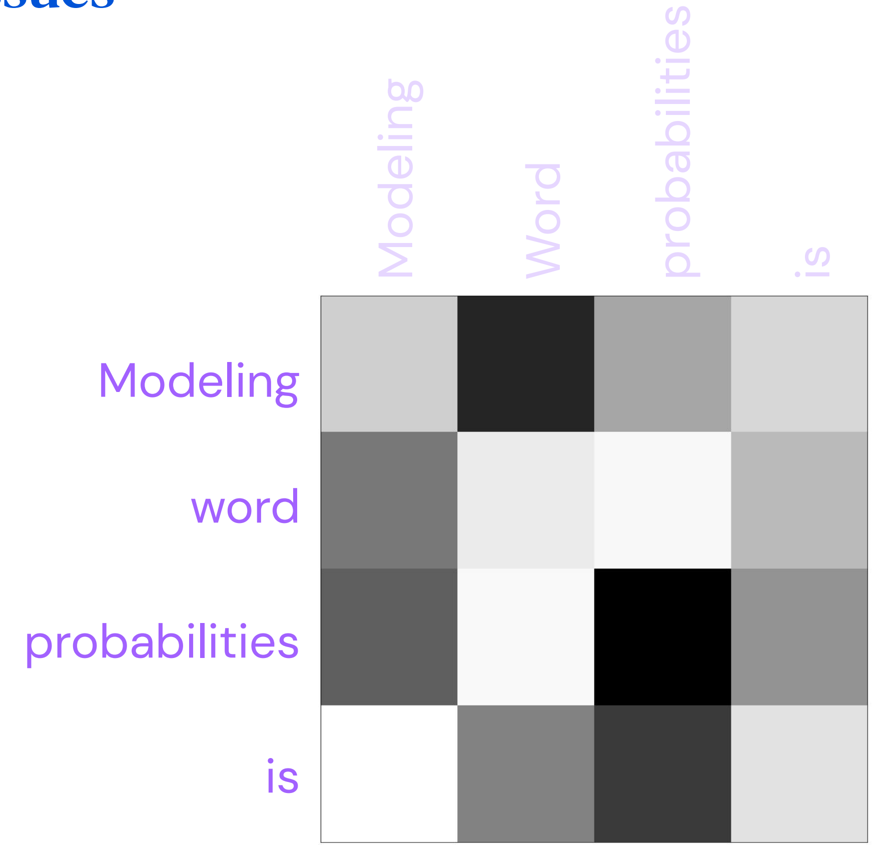
Modeling word probabilities **is** — $p(x_4 | x_3, x_2, x_1)$

Modeling word probabilities is **really** — $p(x_5 | x_4, x_3, x_2, x_1)$

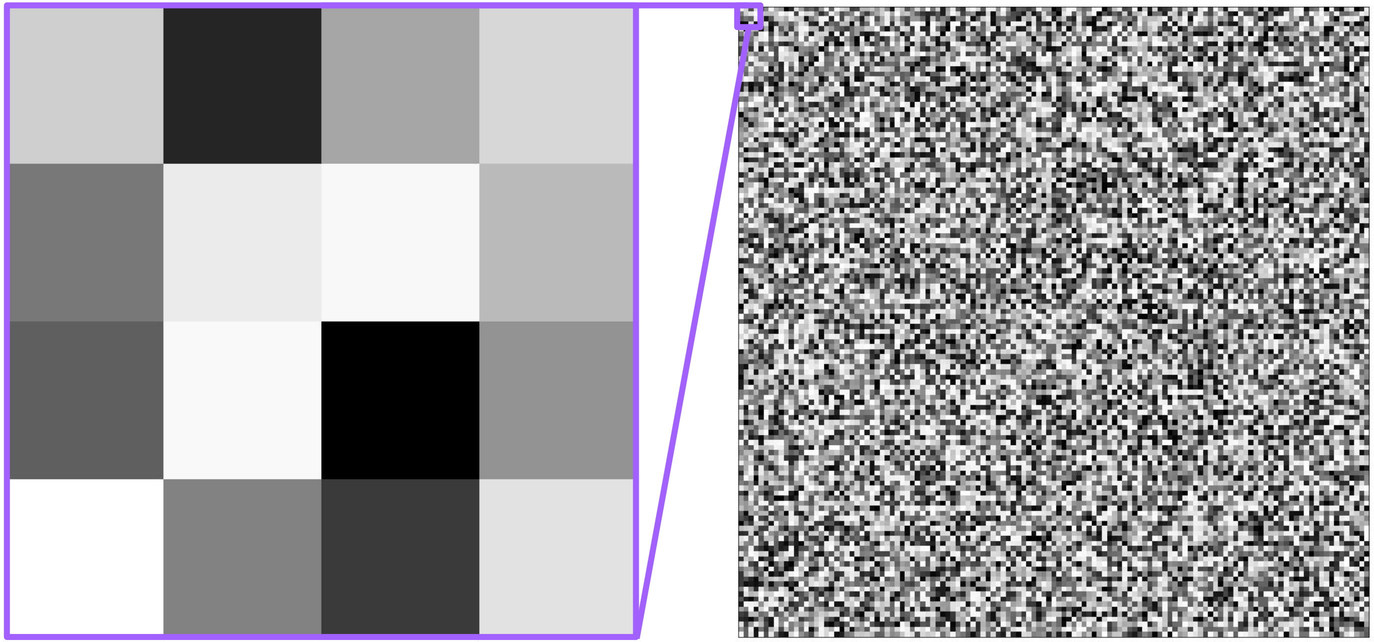
Modeling word probabilities is really **difficult** — $p(x_6 | x_5, x_4, x_3, x_2, x_1)$

Scalability issues

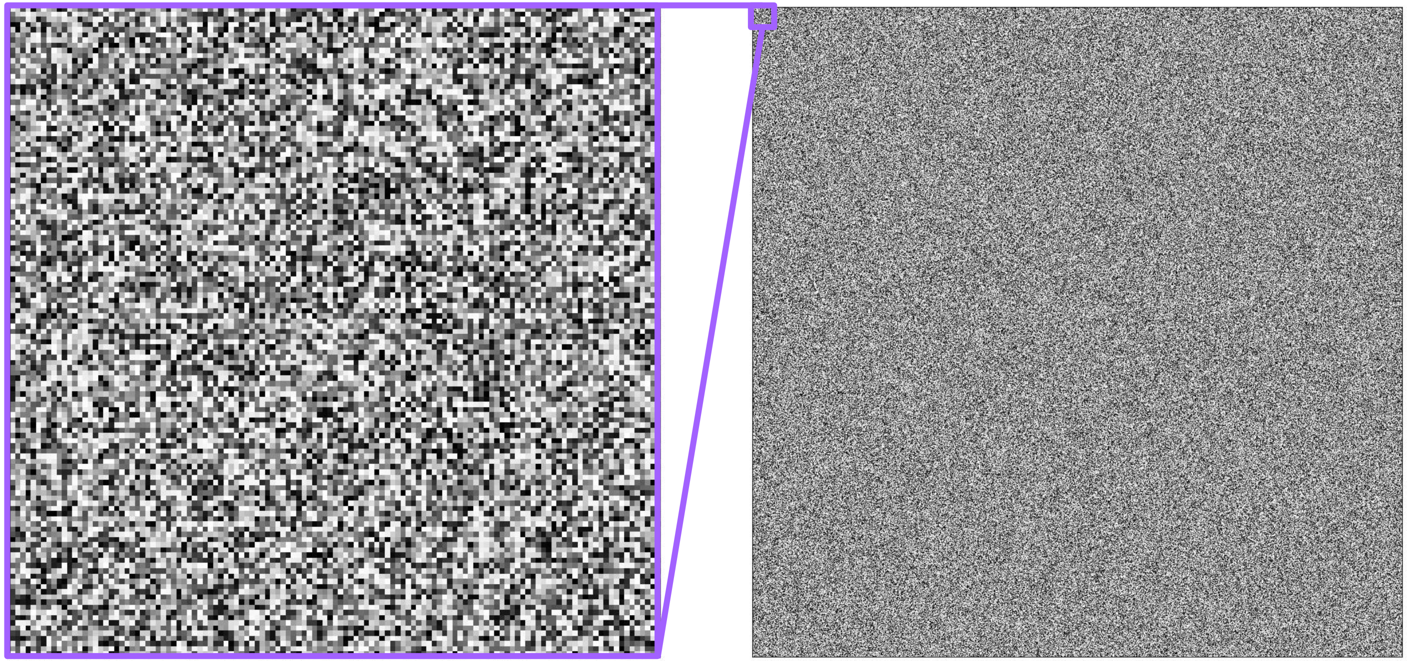
$$p(x_2|x_1)$$



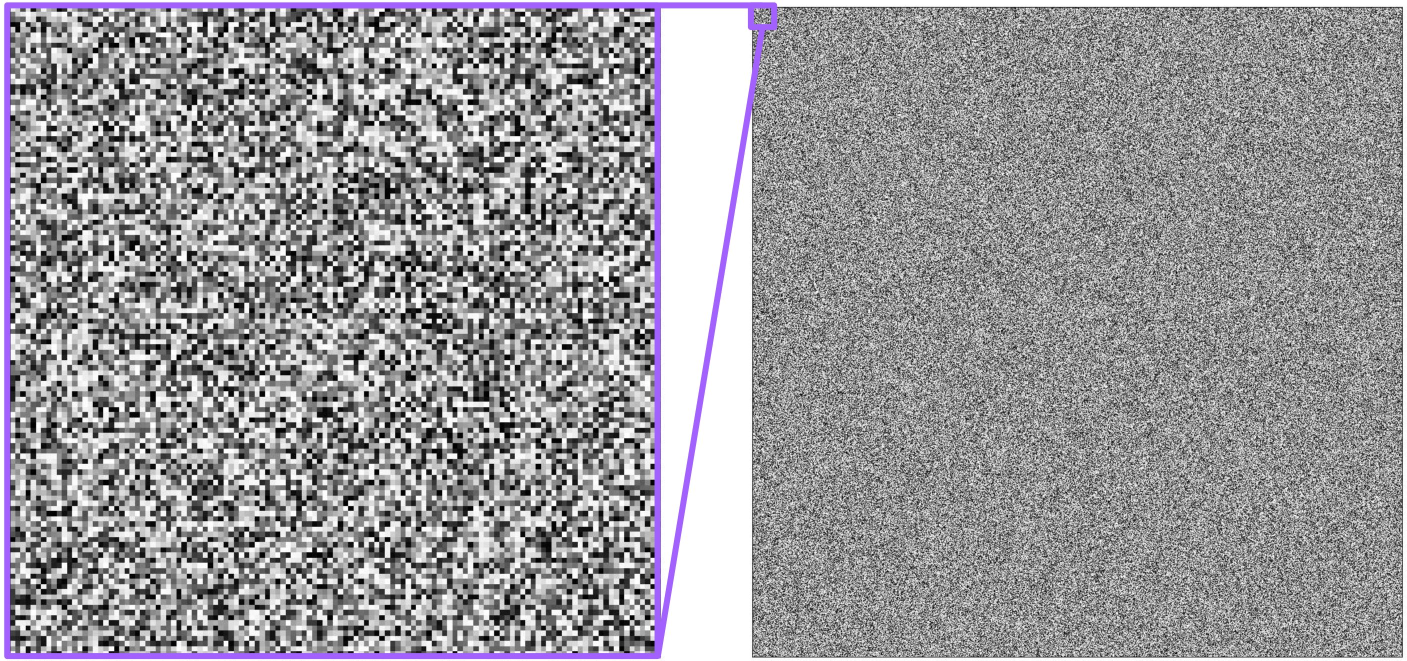
Scalability issues



Scalability issues



Scalability issues



These images are only for context of size N=1!

The table size of larger contexts will grow with **vocabulary**^{N}

Fixing a small context: N-grams

Only condition on N previous words

$$p(\mathbf{x}) \approx \prod_{t=1}^{T} p(x_t | x_{t-N-1}, \dots, x_{t-1})$$

Modeling

Modeling **word**

Modeling word **probabilities**

word probabilities **is**

probabilities is **really**

is really **difficult**

$$p(x_1)$$
$$p(x_2|x_1)$$
$$p(x_3|x_2, x_1)$$
$$p(x_4|x_3, x_2)$$
$$p(x_5|x_4, x_3)$$
$$p(x_6|x_5, x_4)$$

Downsides of using N-grams

1 Doesn't take into account words that are more than N words away

2 Data table is still very, very large

All Our N-gram are Belong to You

Thursday, August 3, 2006

Posted by Alex Franz and Thorsten Brants, Google Machine Translation Team

Here at Google Research we have been using word n-gram models for a variety of R&D projects, such as statistical machine translation, speech recognition, spelling correction, entity detection, information extraction, and others. While such models have usually been estimated from training corpora containing at most a few billion words, we have been harnessing the vast power of Google's datacenters and distributed processing infrastructure to process larger and larger training corpora. We found that there's no data like more data, and scaled up the size of our data by one order of magnitude, and then another, and then one more - resulting in a training corpus of *one trillion words* from public Web pages.

We believe that the entire research community can benefit from access to such massive amounts of data. It will advance the state of the art, it will focus research in the promising direction of large-scale, data-driven approaches, and it will allow all research groups, no matter how large or small their computing resources, to play together. That's why we decided to share this enormous dataset with everyone. We processed 1,024,908,267,229 words of running text and are publishing the counts for all 1,176,470,663 five-word sequences that appear at least 40 times. There are 13,588,391 unique words, after discarding words that appear less than 200 times.

https://ai.googleblog.com/2006/08/all-our-n-gram-are-belong-to-you.html

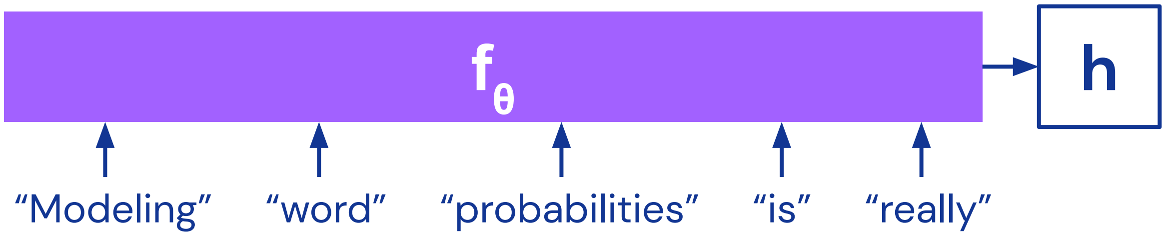
Summary

Modeling probabilities of sequences scales badly given the non-independent structure of their elements

Can this probability estimation be learned from data in a more efficient way?

Learning to model word probabilities

1. Vectorising the context

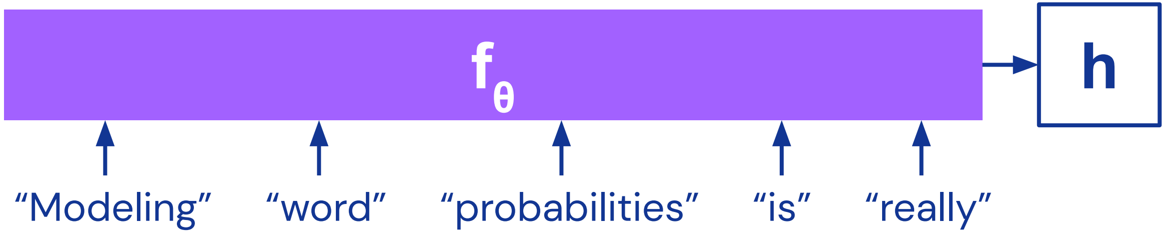


f_θ summarises the context in $\boxed{h}$ such that:

$$p(x_t | x_1, \ldots, x_{t-1}) \approx p(x_t | h)$$

Learning to model word probabilities

1. Vectorising the context



Desirable properties for f_θ:

- Order matters
- Variable length
- Learnable (differentiable)
- Individual changes can have large effects (non-linear/deep)

Desirable properties

Order matters

Variable length

Differentiable

Pairwise encoding

Preserves long-term

N-grams

$$p(\mathbf{x}) \approx \prod_{t=1}^{T} p(x_t | x_{t-N-1}, \ldots, x_{t-1})$$

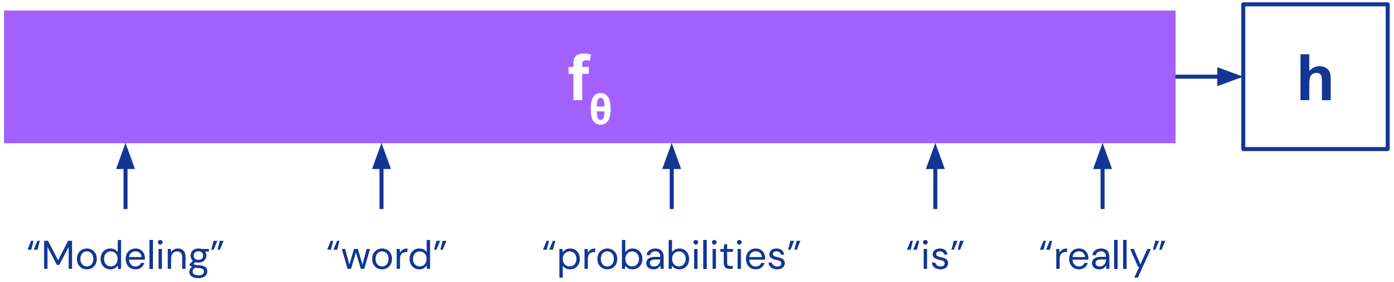
Modeling	$p(x_1)$
Modeling word	$p(x_2 \| x_1)$
Modeling word probabilities	$p(x_3 \| x_2, x_1)$
word probabilities is	$p(x_4 \| x_3, x_2)$
probabilities is really	$p(x_5 \| x_4, x_3)$
is really difficult	$p(x_6 \| x_5, x_4)$

f_θ concatenates the N last words

Properties of N-grams as f_θ

	N-gram
Order matters	✔
Variable length	✘
Differentiable	✘
Pairwise encoding	✔
Preserves long-term	✘

Addition



Properties of addition as f_θ

	N-gram	Addition
Order matters	✔	✘
Variable length	✘	✔
Differentiable	✘	✔
Pairwise encoding	✔	✘
Preserves long-term	✘	✔

Learning to model word probabilities

2. Modeling conditional probabilities

Learning to model word probabilities

2. Modeling conditional probabilities

Desirable properties for g_θ :

Individual changes can have large effects (non-linear/deep)

Returns a probability distribution

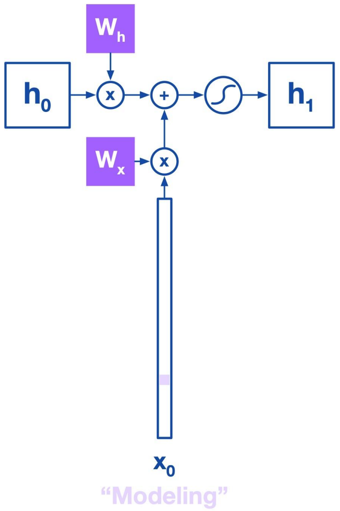
Summary

N-grams and simple aggregation do not meet the requirements for modeling sequences

How can we build a deep network that meets our requirements?

Recurrent Neural Networks (RNNs)

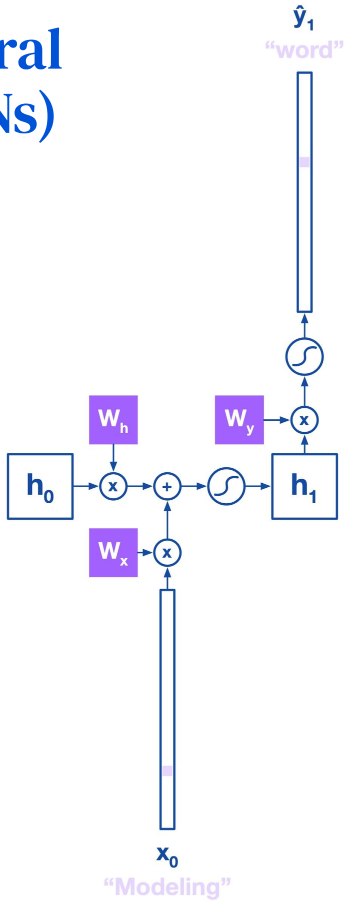
Persistent state variable **h** stores information from the context observed so far.



$$\mathbf{h}_t = \tanh(\mathbf{W}_h \mathbf{h}_{t-1} + \mathbf{W}_x \mathbf{x}_t)$$

Elman (1991)

Recurrent Neural Networks (RNNs)



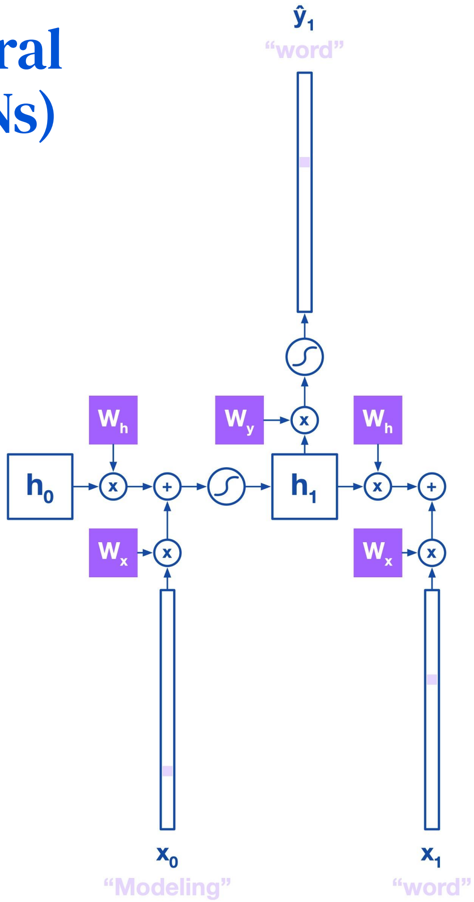
RNNs predict the target **y** (the next word) from the state **h.**

$$p(\mathbf{y_{t+1}}) = softmax(\mathbf{W}_y \mathbf{h}_t)$$

Softmax ensures we obtain a distribution over all possible words.

Elman (1991)

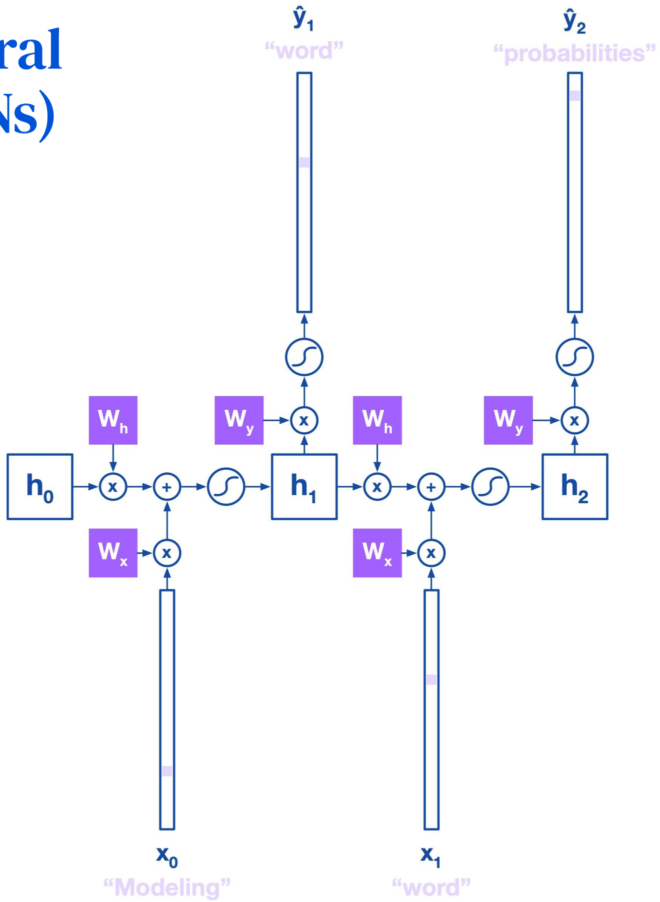
Recurrent Neural Networks (RNNs)



Input next word in sentence x_1

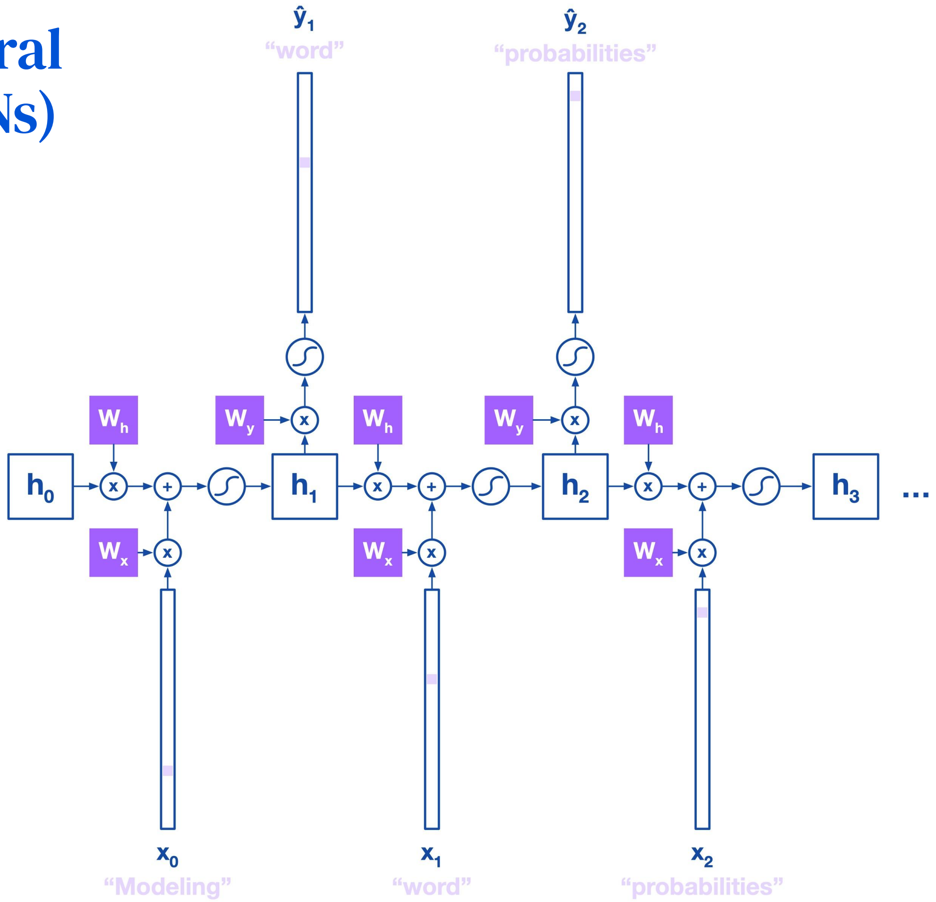
Elman (1991)

Recurrent Neural Networks (RNNs)



Elman (1991)

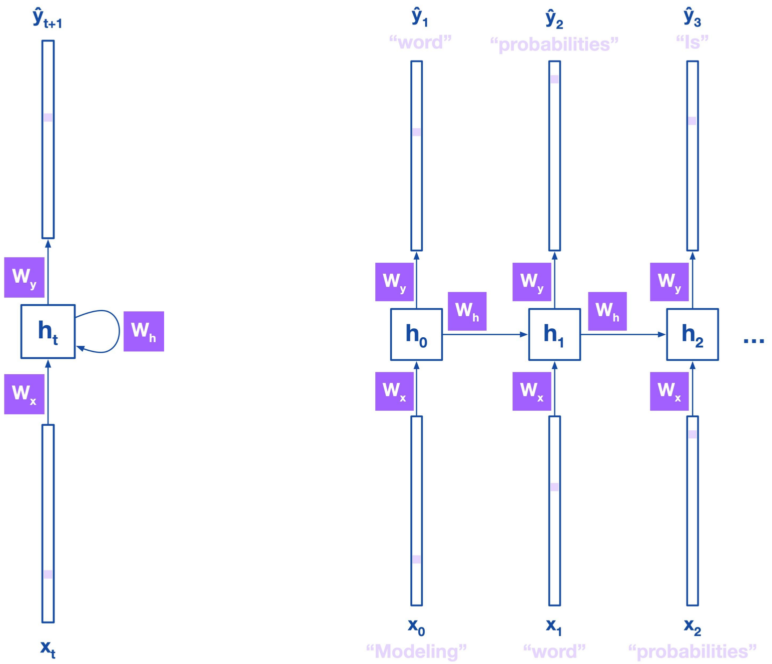
Recurrent Neural Networks (RNNs)



Elman (1991)

Recurrent Neural Networks (RNNs)

Weights are shared over time steps



RNN

RNN rolled out over time

Loss: Cross Entropy

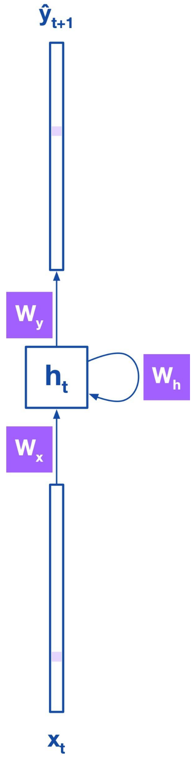
Next word prediction is essentially a classification task where the number of classes is the size of the vocabulary.

As such we use the cross-entropy loss:

For one word: $$\mathcal{L}_\theta(\mathbf{y}, \hat{\mathbf{y}})_t = -\mathbf{y}_t \log \hat{\mathbf{y}}_t$$

For the sentence: $$\mathcal{L}_\theta(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{t=1}^{T} \mathbf{y}_t \log \hat{\mathbf{y}}_t$$

With parameters $\theta = \{\mathbf{W}_y, \mathbf{W}_x, \mathbf{W}_h\}$

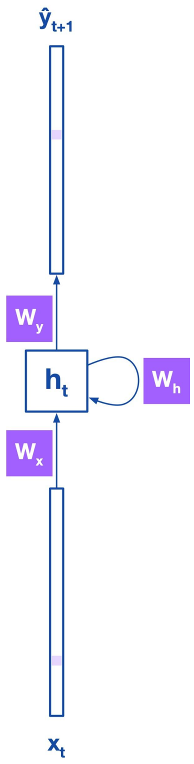


Differentiating wrt W_y, W_x and W_h

$$\mathbf{h}_t = \tanh(\mathbf{W}_h \mathbf{h}_{t-1} + \mathbf{W}_x \mathbf{x}_t)$$

$$p(\mathbf{x}_{t+1}) = softmax(\mathbf{W}_y \mathbf{h}_t)$$

$$\mathcal{L}_\theta(\mathbf{y}, \hat{\mathbf{y}})_t = -\mathbf{y}_t \log \hat{\mathbf{y}}_t$$



Differentiating wrt W_y

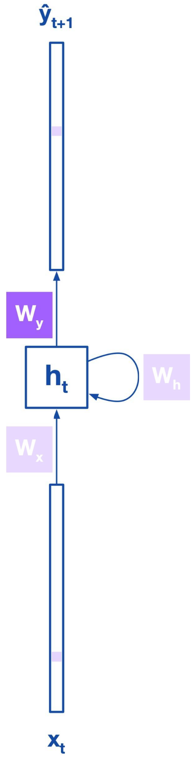
$$\mathbf{h}_t = \tanh(\mathbf{W}_h \mathbf{h}_{t-1} + \mathbf{W}_x \mathbf{x}_t)$$

$$p(\mathbf{x}_{t+1}) = softmax(\mathbf{W}_y \mathbf{h}_t)$$

$$\mathcal{L}_\theta(\mathbf{y}, \hat{\mathbf{y}})_t = -\mathbf{y}_t \log \hat{\mathbf{y}}_t$$

$$\frac{\partial \mathcal{L}_{\theta,t}}{\partial \mathbf{W}_y} = \frac{\partial \mathcal{L}_{\theta,t}}{\partial \hat{\mathbf{y}}_t} \frac{\partial \hat{\mathbf{y}}_t}{\partial \mathbf{W}_y}$$

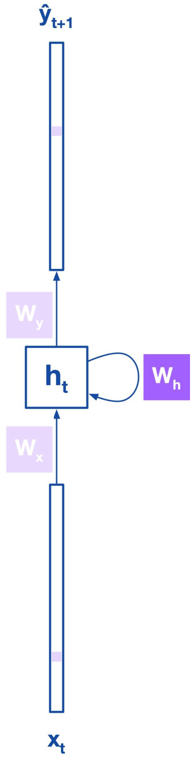
$$= (\mathbf{y}_t - \hat{\mathbf{y}}_t)\mathbf{h}_t$$



Differentiating wrt W_h

$$\mathbf{h}_t = \tanh(\mathbf{W}_h \mathbf{h}_{t-1} + \mathbf{W}_x \mathbf{x}_t)$$
$$p(\mathbf{x}_{t+1}) = softmax(\mathbf{W}_y \mathbf{h}_t)$$
$$\mathcal{L}_\theta(\mathbf{y}, \hat{\mathbf{y}})_t = -\mathbf{y}_t \log \hat{\mathbf{y}}_t$$

$$\frac{\partial \mathcal{L}_{\theta,t}}{\partial \mathbf{W}_h} = \frac{\partial \mathcal{L}_{\theta,t}}{\partial \hat{\mathbf{y}}_t} \frac{\partial \hat{\mathbf{y}}_t}{\partial \mathbf{h}_t} \frac{\partial \mathbf{h}_t}{\partial \mathbf{W}_h}$$



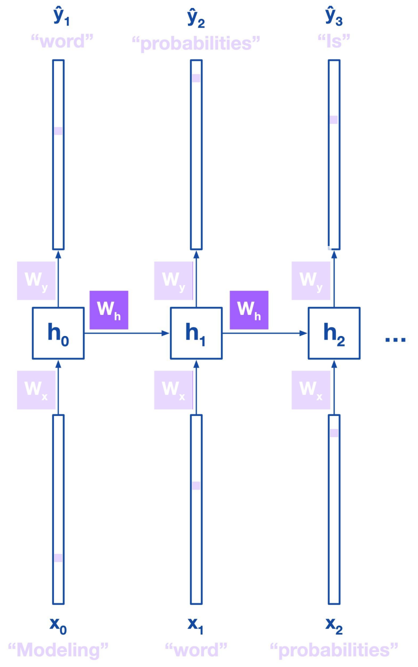
Differentiating wrt W_h

$$\mathbf{h}_t = \tanh(\mathbf{W}_h \mathbf{h}_{t-1} + \mathbf{W}_x \mathbf{x}_t)$$

$$p(\mathbf{x}_{t+1}) = softmax(\mathbf{W}_y \mathbf{h}_t)$$

$$\mathcal{L}_\theta(\mathbf{y}, \hat{\mathbf{y}})_t = -\mathbf{y}_t \log \hat{\mathbf{y}}_t$$

$$\frac{\partial \mathcal{L}_{\theta,t}}{\partial \mathbf{W}_h} = \frac{\partial \mathcal{L}_{\theta,t}}{\partial \hat{\mathbf{y}}_t} \frac{\partial \hat{\mathbf{y}}_t}{\partial \mathbf{h}_t} \frac{\partial \mathbf{h}_t}{\partial \mathbf{W}_h}$$



Back propagating through time

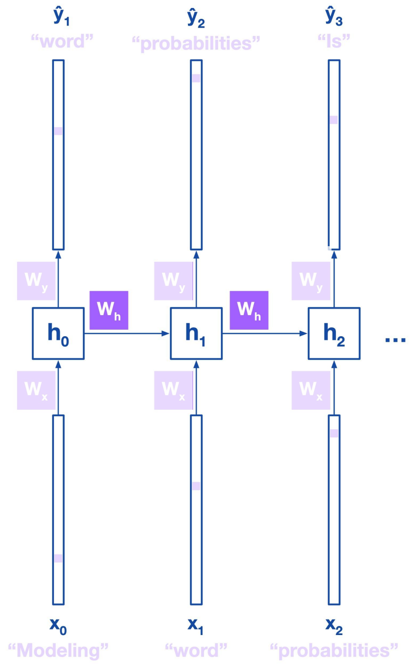
$$
\begin{aligned}
\frac{\partial \mathbf{h}_t}{\partial \mathbf{W}_h} &= \frac{\partial \mathbf{h}_t}{\partial \mathbf{W}_h} + \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_{t-1}} \frac{\partial \mathbf{h}_{t-1}}{\partial \mathbf{W}_h} \\
&= \frac{\partial \mathbf{h}_t}{\partial \mathbf{W}_h} + \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_{t-1}} \left[\frac{\partial \mathbf{h}_{t-1}}{\partial \mathbf{W}_h} + \frac{\partial \mathbf{h}_{t-1}}{\partial \mathbf{h}_{t-2}} \frac{\partial \mathbf{h}_{t-2}}{\partial \mathbf{W}_h} \right] \\
&\dots \\
&= \sum_{k=1}^{t} \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} \frac{\partial \mathbf{h}_k}{\partial \mathbf{W}_h}
\end{aligned}
$$

Differentiating wrt W_h

$$\mathbf{h}_t = \tanh(\mathbf{W}_h \mathbf{h}_{t-1} + \mathbf{W}_x \mathbf{x}_t)$$

$$p(\mathbf{x}_{t+1}) = softmax(\mathbf{W}_y \mathbf{h}_t)$$

$$\mathcal{L}_\theta(\mathbf{y}, \hat{\mathbf{y}})_t = -\mathbf{y}_t \log \hat{\mathbf{y}}_t$$



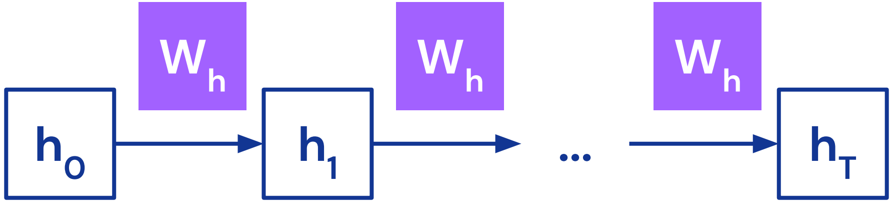
$$\frac{\partial \mathcal{L}_{\theta,t}}{\partial \mathbf{W}_h} = \frac{\partial \mathcal{L}_{\theta,t}}{\partial \hat{\mathbf{y}}_t} \frac{\partial \hat{\mathbf{y}}_t}{\partial \mathbf{h}_t} \frac{\partial \mathbf{h}_t}{\partial \mathbf{W}_h}$$

$$= \sum_{k=1}^{t} \frac{\partial \mathcal{L}_{\theta,t}}{\partial \hat{\mathbf{y}}_t} \frac{\partial \hat{\mathbf{y}}_t}{\partial \mathbf{h}_t} \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} \frac{\partial \mathbf{h}_k}{\partial \mathbf{W}_h}$$

$$\frac{\partial \mathbf{h}_t}{\partial \mathbf{W}_h} = \sum_{k=1}^{t} \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} \frac{\partial \mathbf{h}_k}{\partial \mathbf{W}_h}$$

Vanishing gradients

A simple example



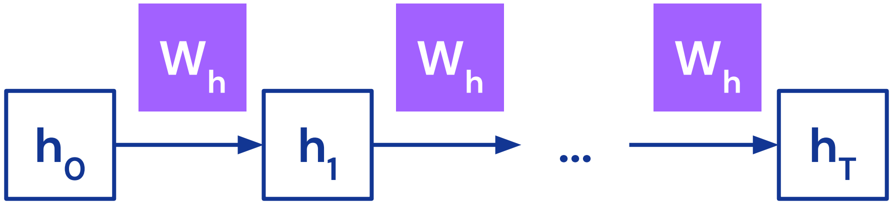
$$\mathbf{h}_t = \mathbf{W}_h \mathbf{h}_{t-1}$$
$$\mathbf{h}_t = (\mathbf{W}_h)^t \mathbf{h}_0$$

$$\mathbf{h}_t \rightarrow \infty \text{ if } |\mathbf{W}_h| > 1$$
$$\mathbf{h}_t \rightarrow 0 \text{ if } |\mathbf{W}_h| < 1$$

Hochreiter (1991), Bengio et al. (1994)

Vanishing gradients

A simple example



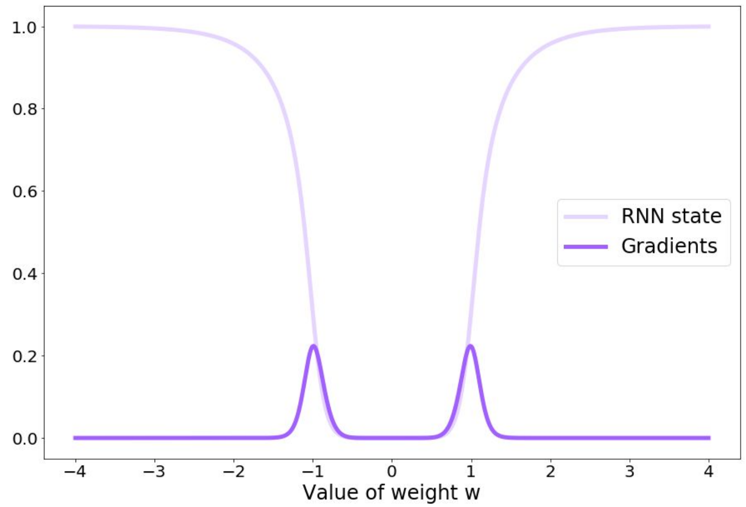
But RNNs bound h with a tanh!

$$\mathbf{h}_t = \mathbf{W}_h \mathbf{h}_{t-1} \longrightarrow \mathbf{h}_t = \tanh(\mathbf{W}_h \mathbf{h}_{t-1})$$

Vanishing gradients

$$\mathbf{h}_t = \tanh(\mathbf{W}_h \mathbf{h}_{t-1})$$

```
import numpy as np
import matplotlib.pyplot as plt

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

  dh = 1
  for t in range(T):
    dh = (1-hs[-1-t] ** 2) * w * dh

  return hs[-1], dh

T = 10  # sequence length
wlim = 4  #limit of interval over weights w

results = []
ws = np.linspace(-wlim, wlim, 1000)
for w in ws:
  results.append(forward_backward_prop(w, T))

plt.plot(ws, [r[0] for r in results], label='RNN state')
plt.plot(ws, [r[1] for r in results], label='Gradients')
```



$$\frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_{t-T}} = (1 - \tanh^2(\mathbf{W}_h \mathbf{h}_{t-1}))\mathbf{W}_h \frac{\partial \mathbf{h}_{t-1}}{\partial \mathbf{h}_{t-T}}$$

Properties of RNNs as f_θ

	N-gram	Addition	RNN
Order matters	✓	✗	✓
Variable length	✗	✓	✓
Differentiable	✗	✓	✓
Pairwise encoding	✓	✗	✗
Preserves long-term	✗	✓	✗

Summary

Recurrent neural networks can model sequences of variable length and can be trained via back-propagation

They do, however suffer from the vanishing gradients problem, which stops them from capturing long-term dependencies

Long term dependencies are important

… Finally, Tim was planning to visit France on the final week of his journey. He was quite excited to try the local delicacies and had lots of recommendations for good restaurants and exhibitions. His first stop was, of course, the capital where he would meet his long-time Friend Jean-Pierre. In order to arrive for breakfast he took the early 5 AM train from London to …

Long term dependencies are important

… Finally, Tim was planning to visit **France** on the final week of his journey. He was quite excited to try the local delicacies and had lots of recommendations for good restaurants and exhibitions. His first stop was, of course, the **capital** where he would meet his long-time Friend Jean-Pierre. In order to arrive for breakfast he took the early 5 AM train from London to …

Long term dependencies are important

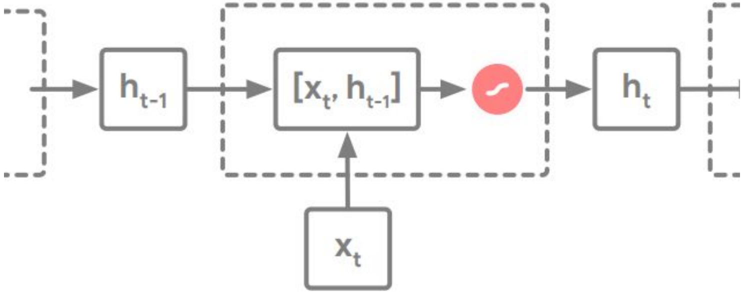
… Finally, Tim was planning to visit **France** on the final week of his journey. He was quite excited to try the local delicacies and had lots of recommendations for good restaurants and exhibitions. His first stop was, of course, the **capital** where he would meet his long-time Friend Jean-Pierre. In order to arrive for breakfast he took the early 5 AM train from London to …

PARIS!

How can we capture long-term dependencies?

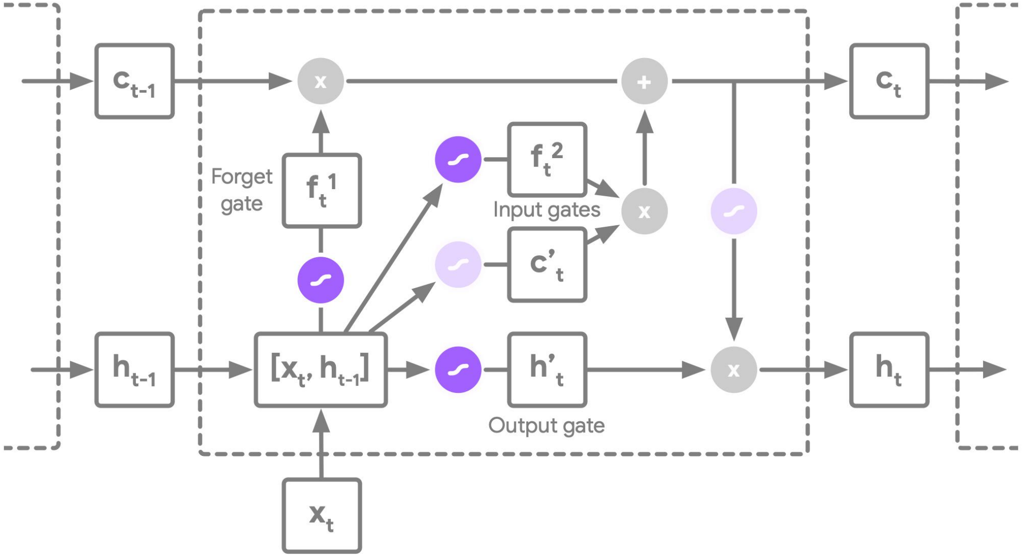
Long Short-Term Memory (LSTM) networks

RNN state update



Long Short-Term Memory (LSTM) networks

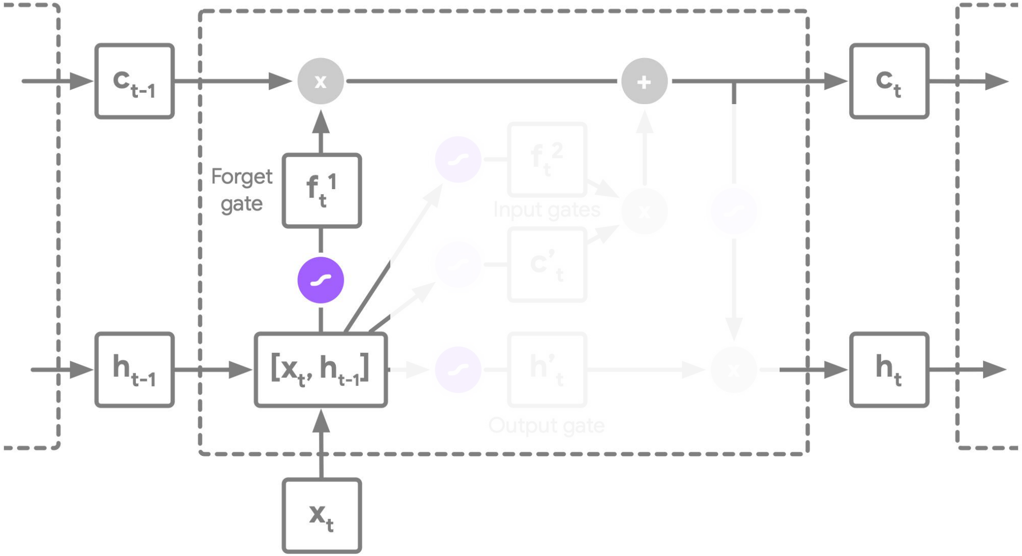
LSTM state update



Hochreiter (1997), Gers et al. (1999)

Long Short-Term Memory (LSTM) networks

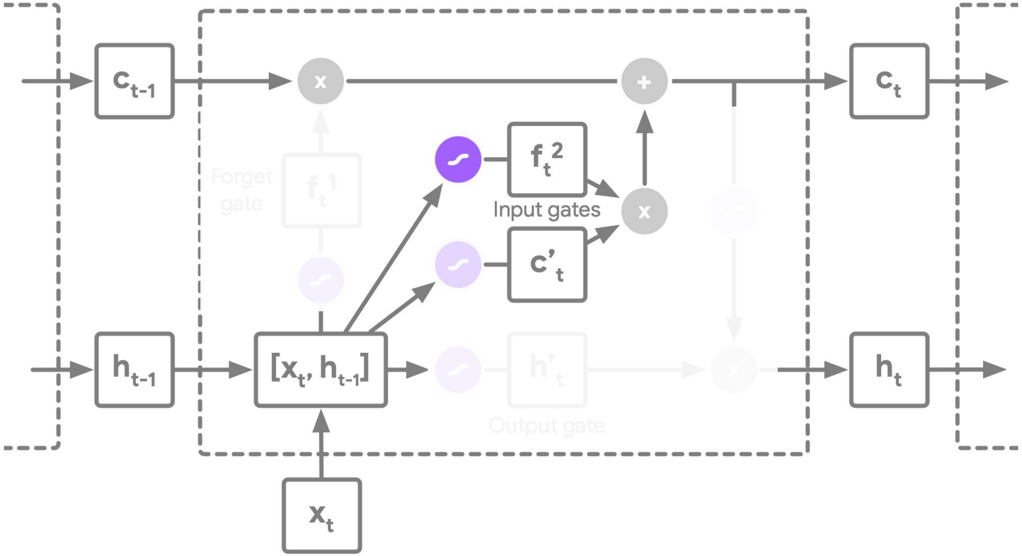
1) Forget gate



$$f_t^1 = \sigma(\mathbf{W}_{f^1} \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_{f^1})$$

Long Short-Term Memory (LSTM) networks

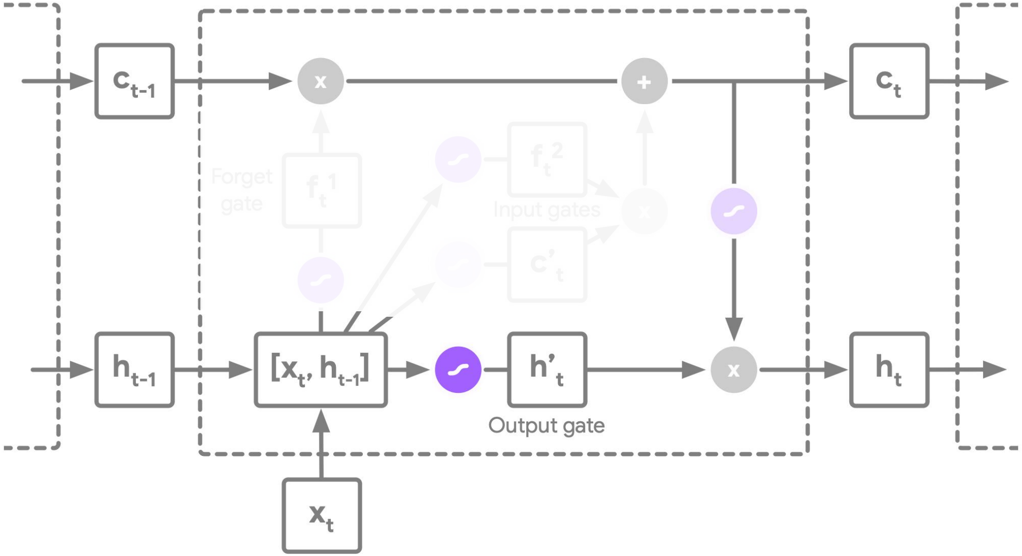
2) Input gates



$$f_t^2 = \sigma(\mathbf{W}_{f^2} \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_{f^2}) \odot \tanh(\mathbf{W}_{f^2} \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_{f^2})$$

Long Short-Term Memory (LSTM) networks

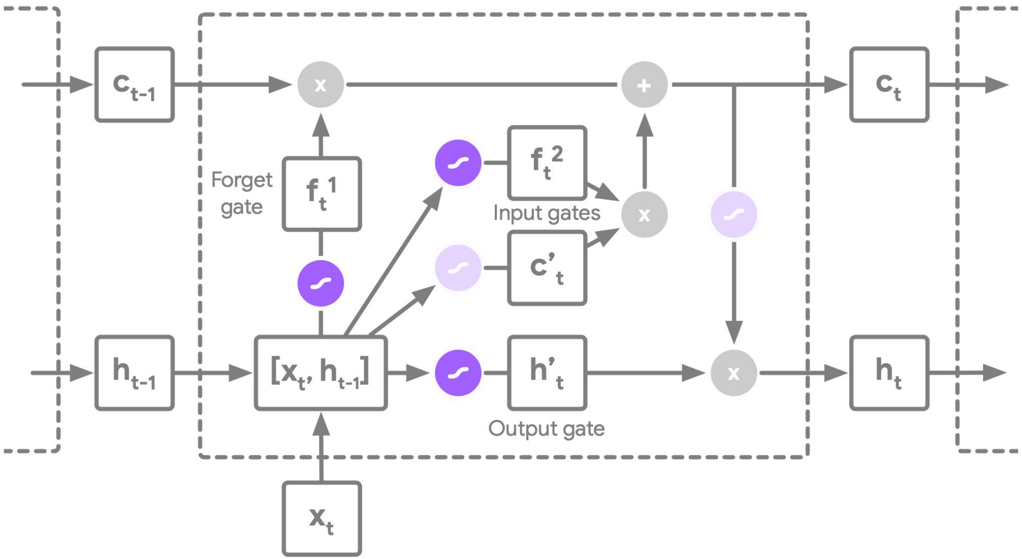
3) Output gate



$$h'_t = \sigma(\mathbf{W}_{h'_t} \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_{h'_t}) \odot \tanh(\mathbf{c}_t)$$

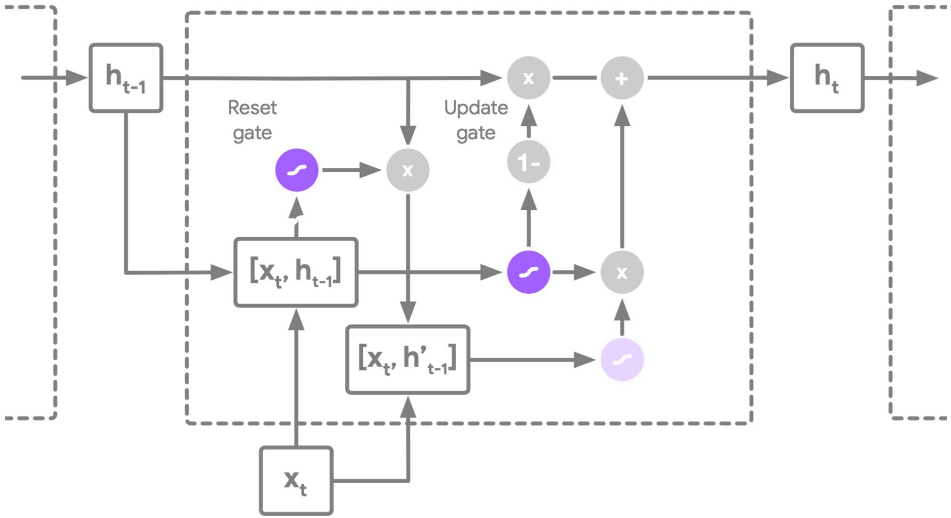
Long Short-Term Memory (LSTM) networks

LSTM state update



Gated Recurrent Unit nets

GRU state update



GRU can be seen as a simplified LSTM.

Cho et al. (2014)

Properties of LSTMs as f_θ

	N-gram	Addition	RNN	LSTM
Order matters	✔	✘	✔	✔
Variable length	✘	✔	✔	✔
Differentiable	✘	✔	✔	✔
Pairwise encoding	✔	✘	✘	✘
Preserves long-term	✘	✔	✘	✔

Summary

LSTMs and GRUs overcome the vanishing gradient problem by making use of sophisticated gating mechanisms

As a result these models are ubiquitous across machine learning research

DeepMind

3 Generating Sequences

Using a trained model

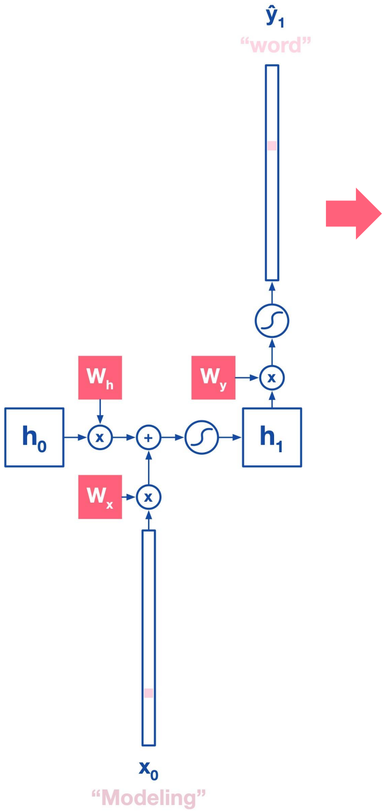
During training we focussed on optimising the **log probability** estimates produced by our model.

$$\boxed{\mathcal{L}_\theta(\mathbf{y}, \hat{\mathbf{y}})_t = -\mathbf{y}_t \log \hat{\mathbf{y}}_t}$$

This means at test time we can use it to evaluate the probability of a new sentence. This, however, is arguably not very interesting.

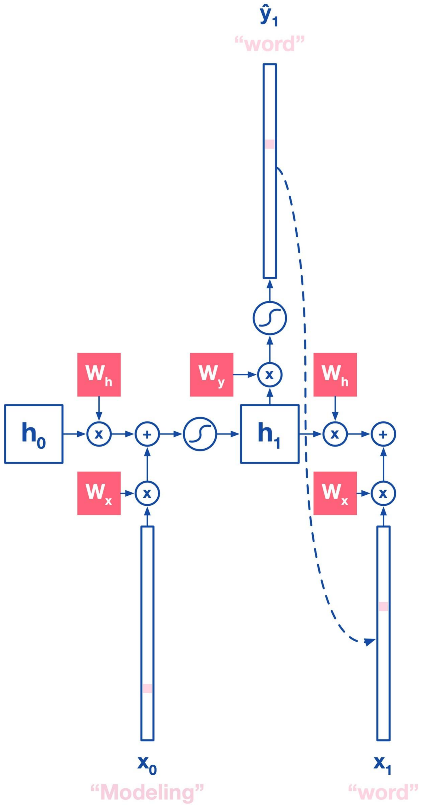
An alternative use case of our trained model is sequence **generation**.

Generating sequences with RNNs



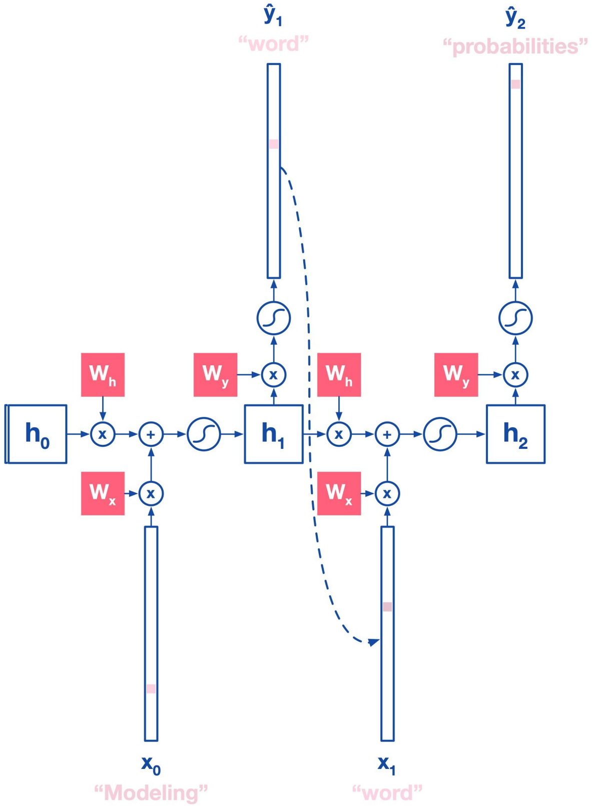
$\hat{y}$ is a probability distribution over possible words that we can sample from.

Generating sequences with RNNs

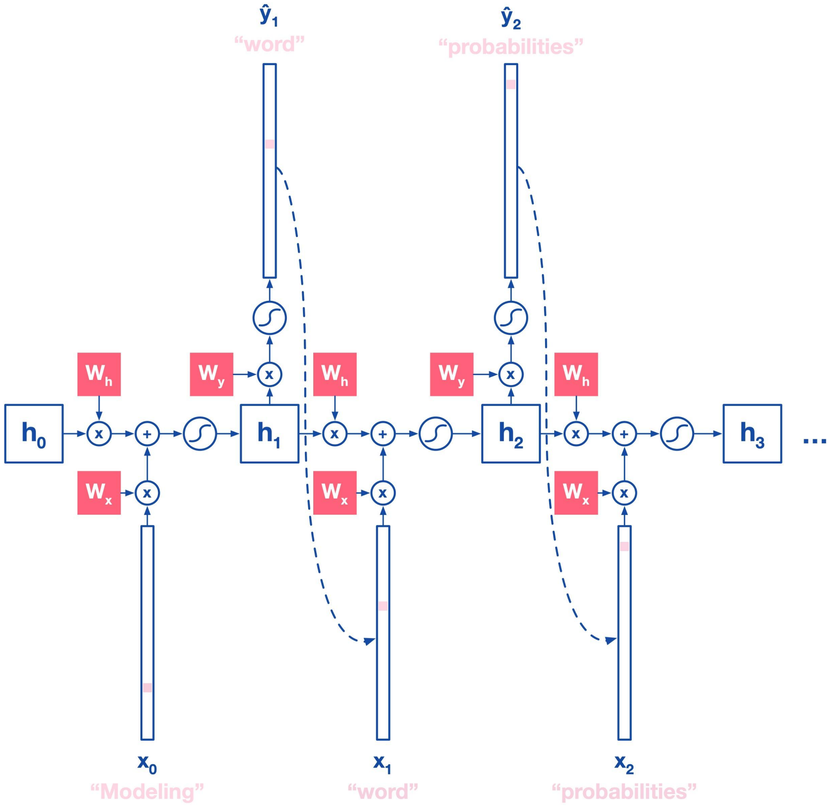


The sampled ŷ is the input to the next iteration of the network.

Generating sequences with RNNs

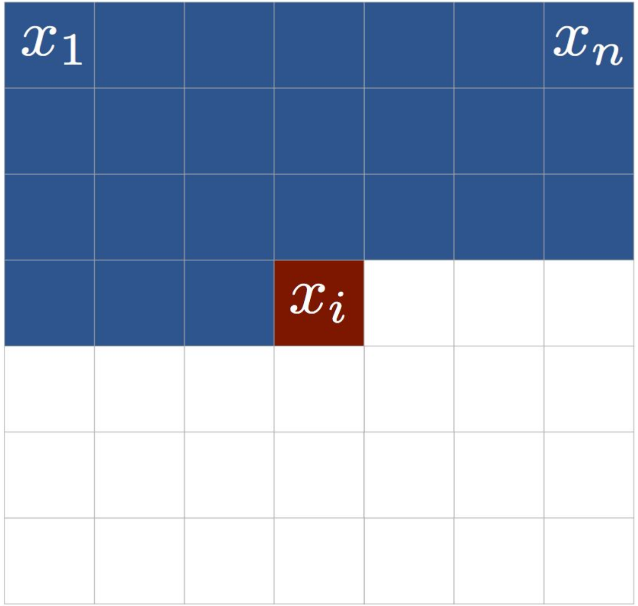


Generating sequences with RNNs



Images as sequences

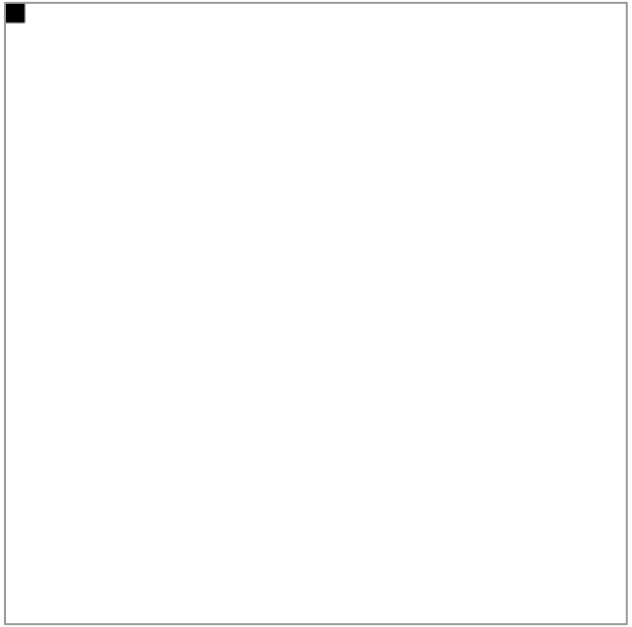
Images as sequences: PixelRNN

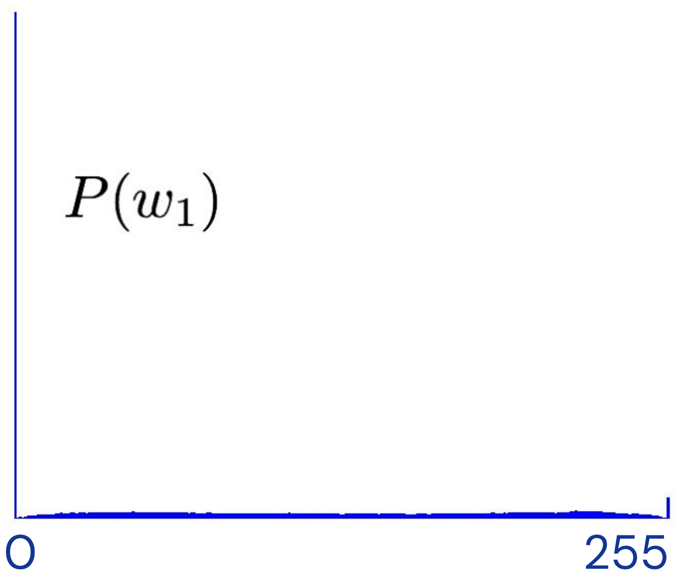


$$p(\mathbf{x}) = \prod_{i=1}^{n^2} p(x_i | x_1, \ldots, x_{i-1})$$

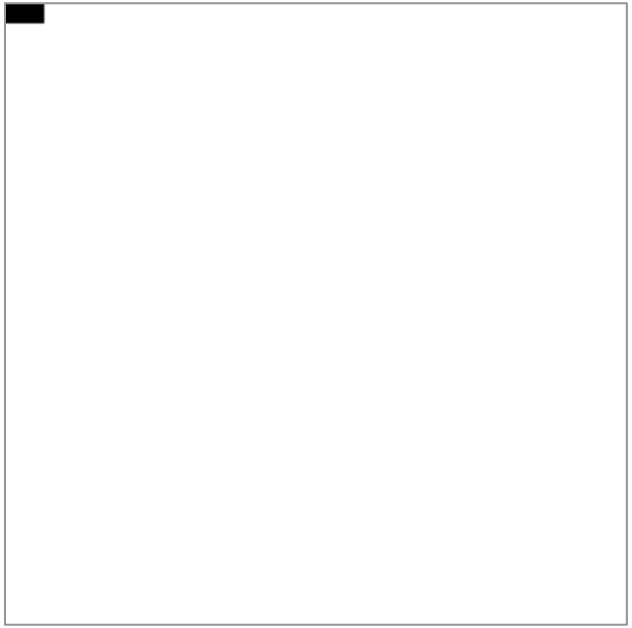
Van den Oord et al. (2016)

Softmax Sampling





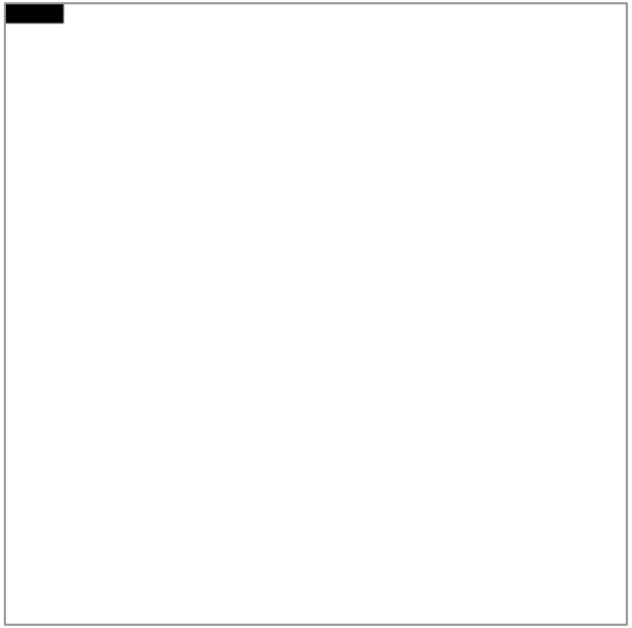
Softmax Sampling



$$P(w_1)$$

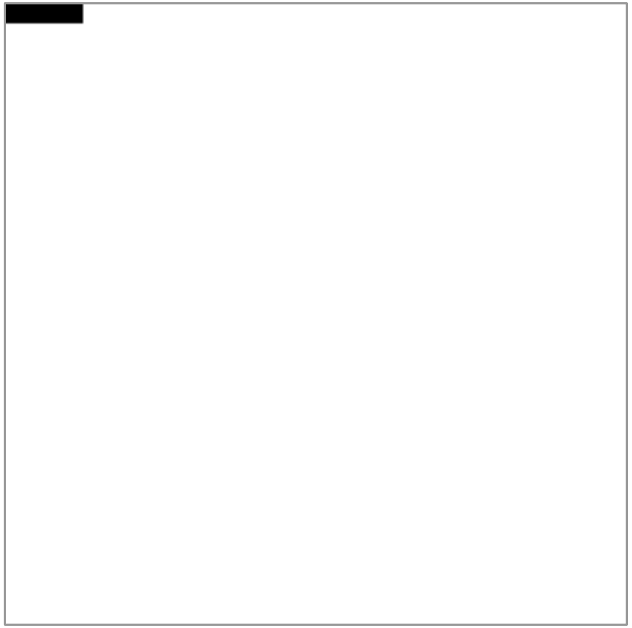
$$P(w_2|w_1)$$

Softmax Sampling



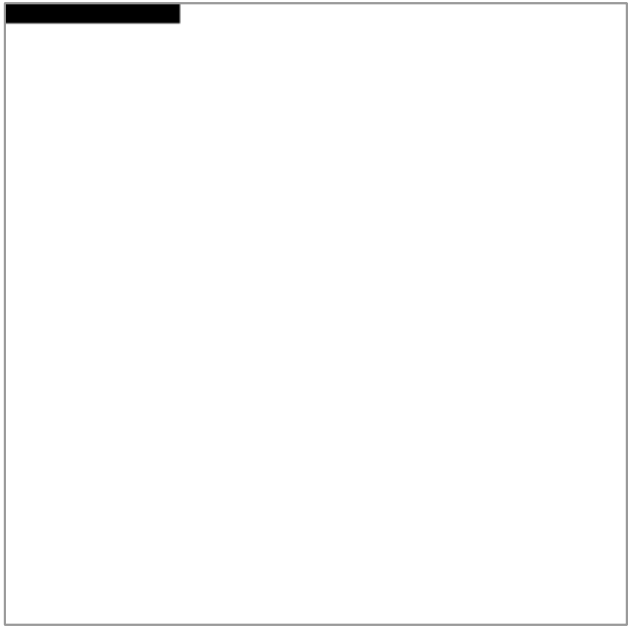
$$P(w_1)$$
$$P(w_2|w_1)$$
$$P(w_3|w_2, w_1)$$

Softmax Sampling

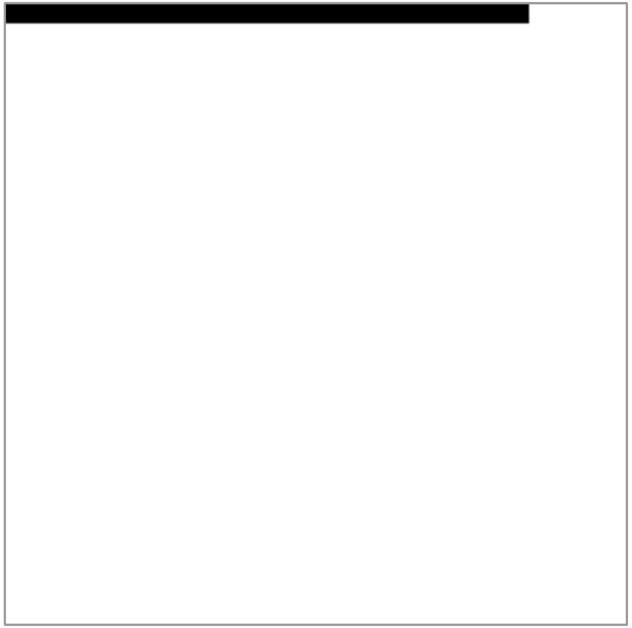


$$P(w_1)$$
$$P(w_2 | w_1)$$
$$P(w_3 | w_2, w_1)$$
$$P(w_4 | w_3, w_2, w_1)$$

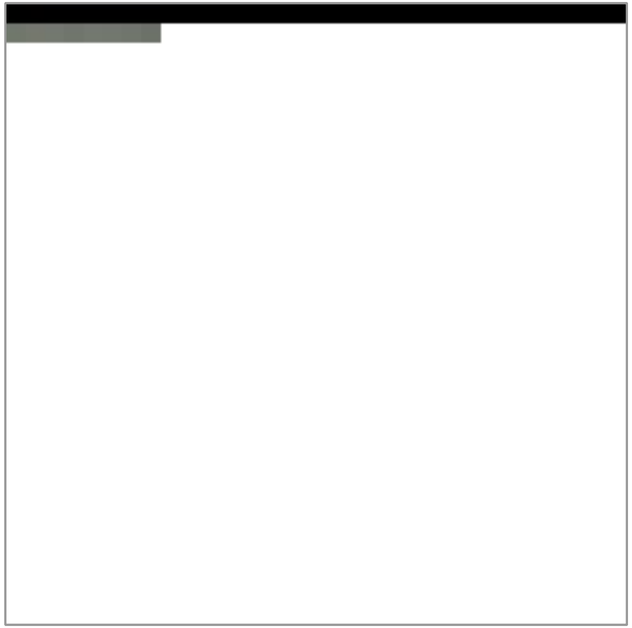
Softmax Sampling



Softmax Sampling



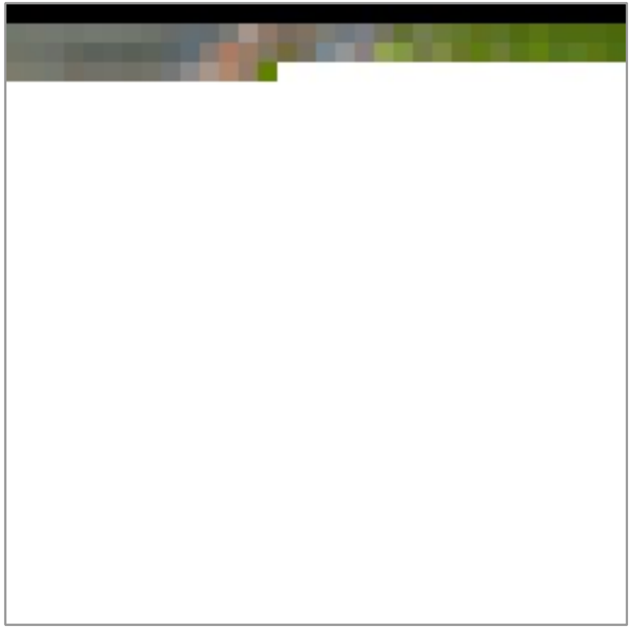
Softmax Sampling



Softmax Sampling

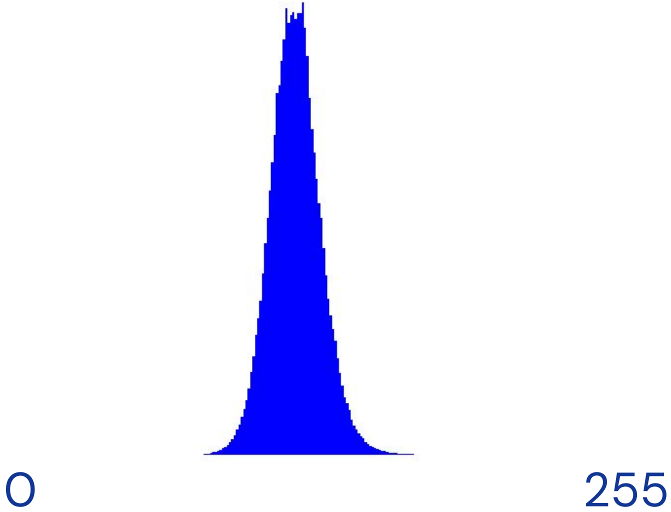
Softmax Sampling

Softmax Sampling

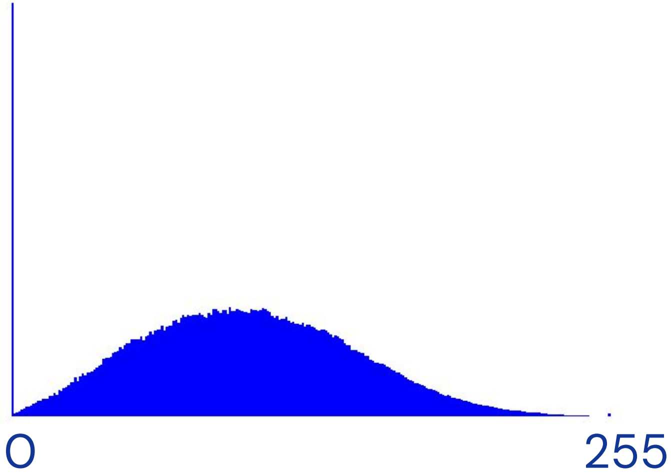


Softmax Sampling

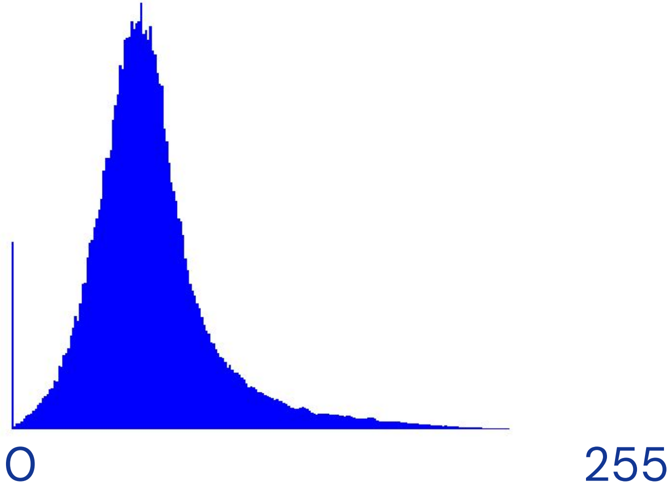
Softmax Sampling



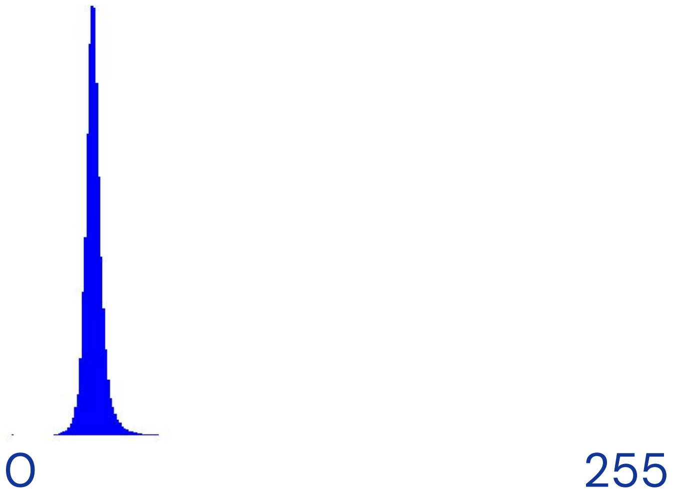
Softmax Sampling



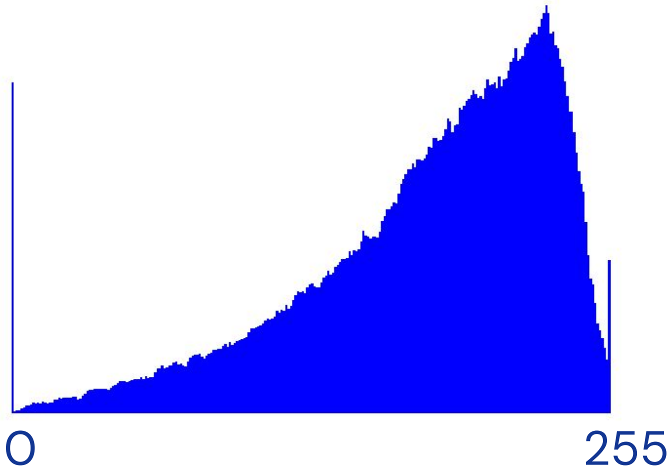
Softmax Sampling



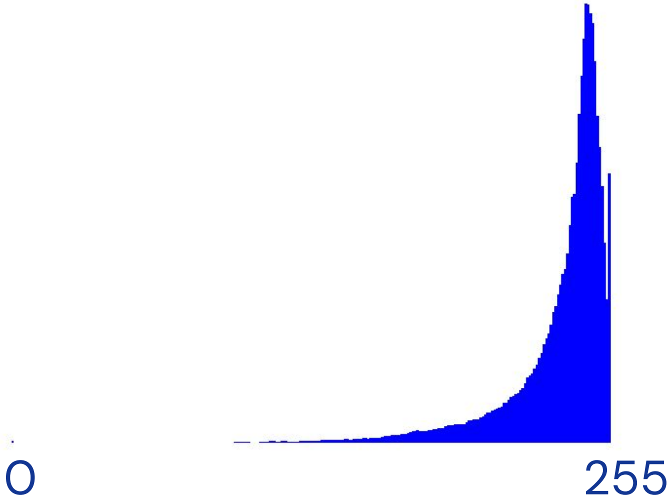
Softmax Sampling



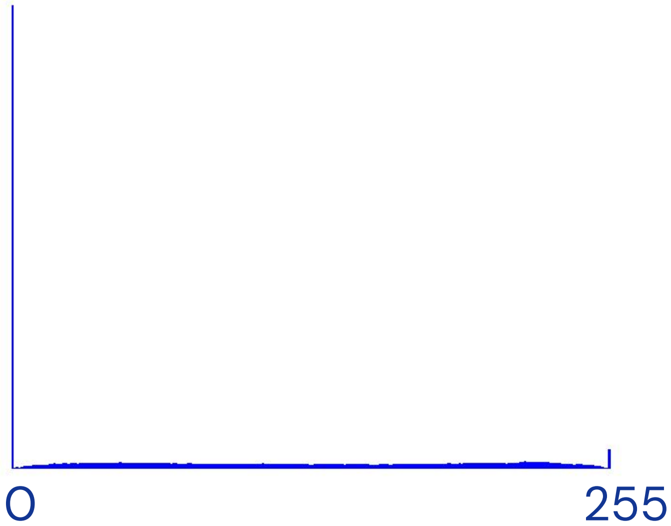
Softmax Sampling



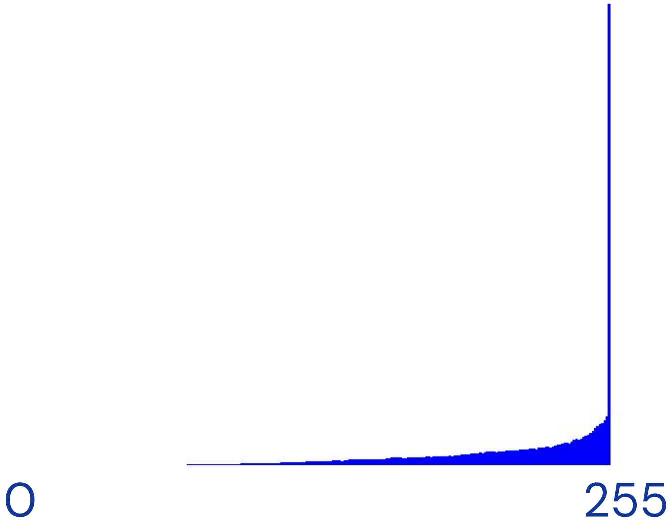
Softmax Sampling



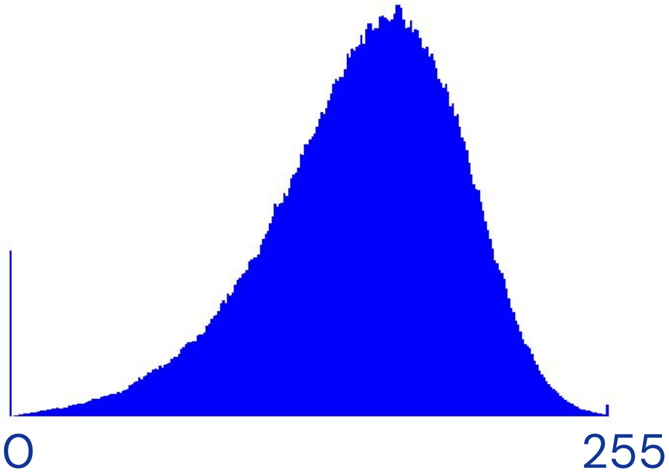
Softmax Sampling



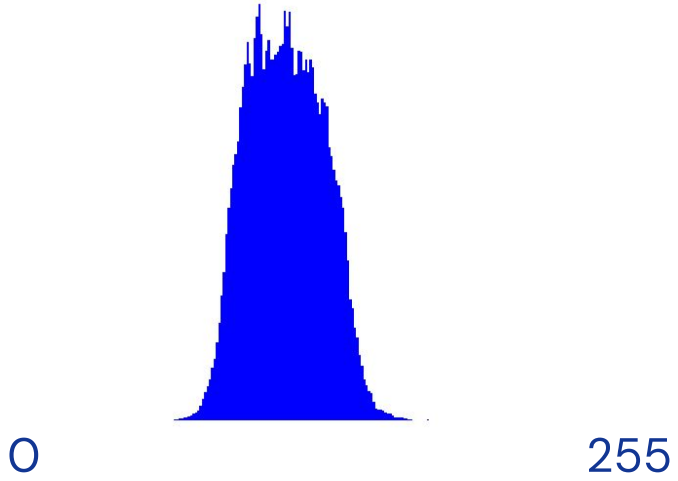
Softmax Sampling



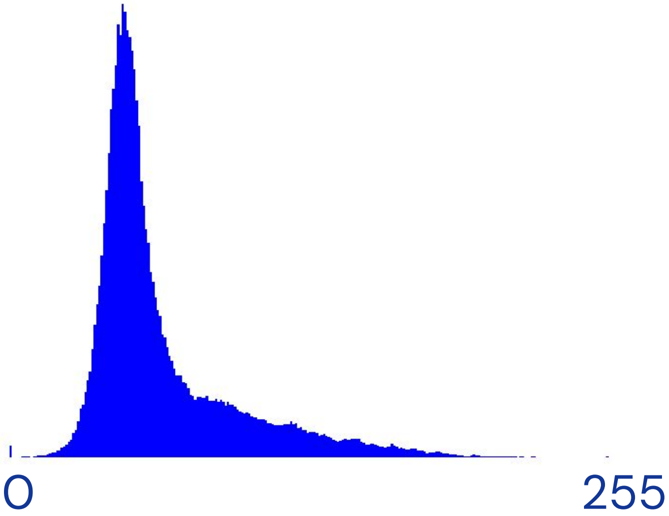
Softmax Sampling



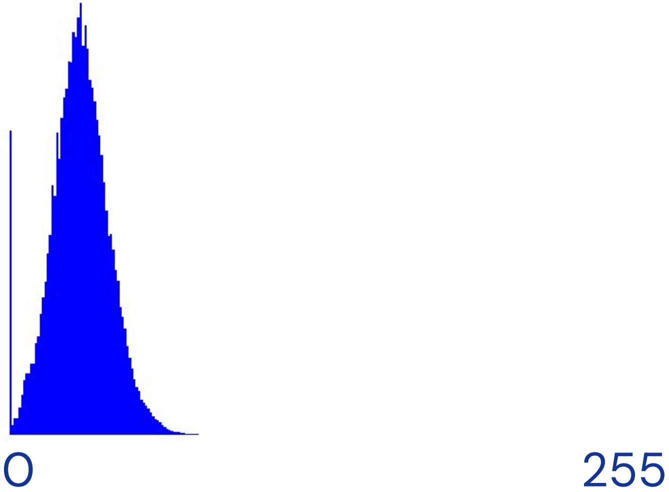
Softmax Sampling



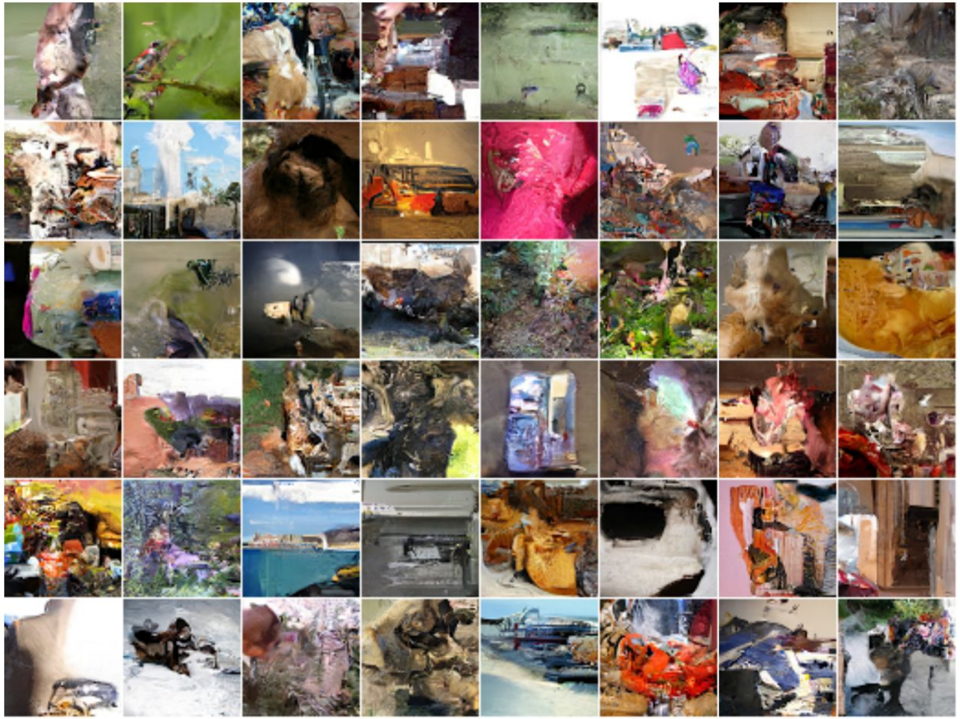
Softmax Sampling



Softmax Sampling

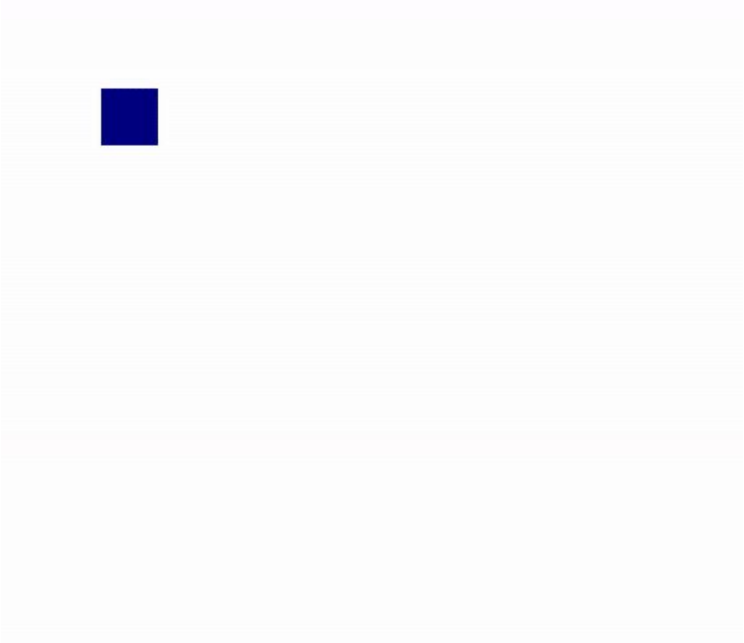


Images as sequences: PixelRNN

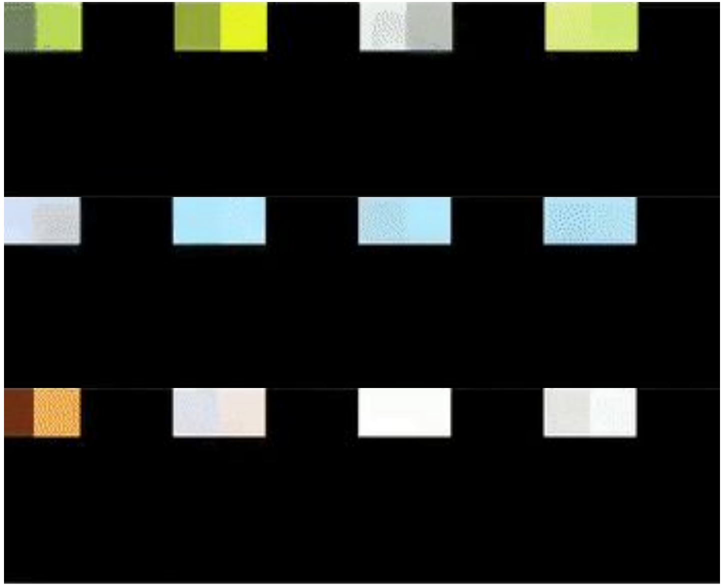


Van den Oord et al. (2016)

Images as sequences



Pixel-by-pixel

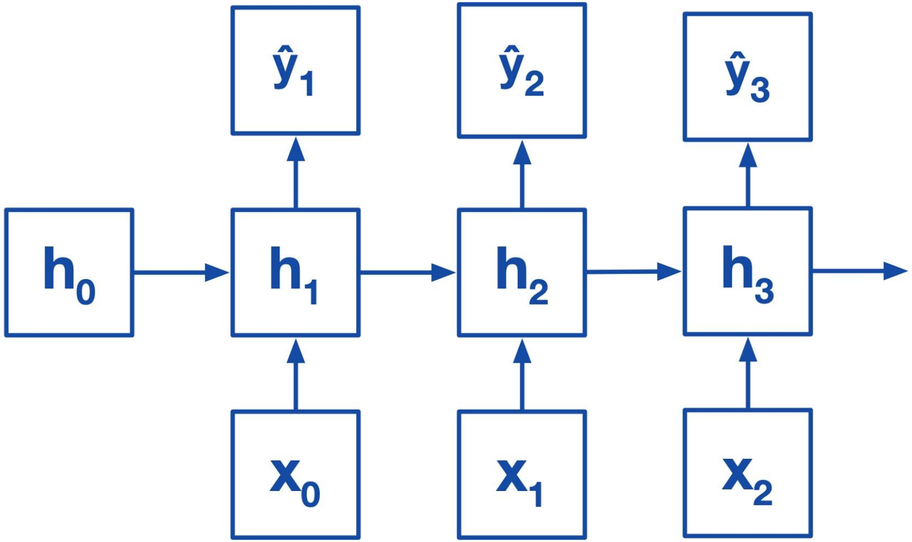


Group-by-group

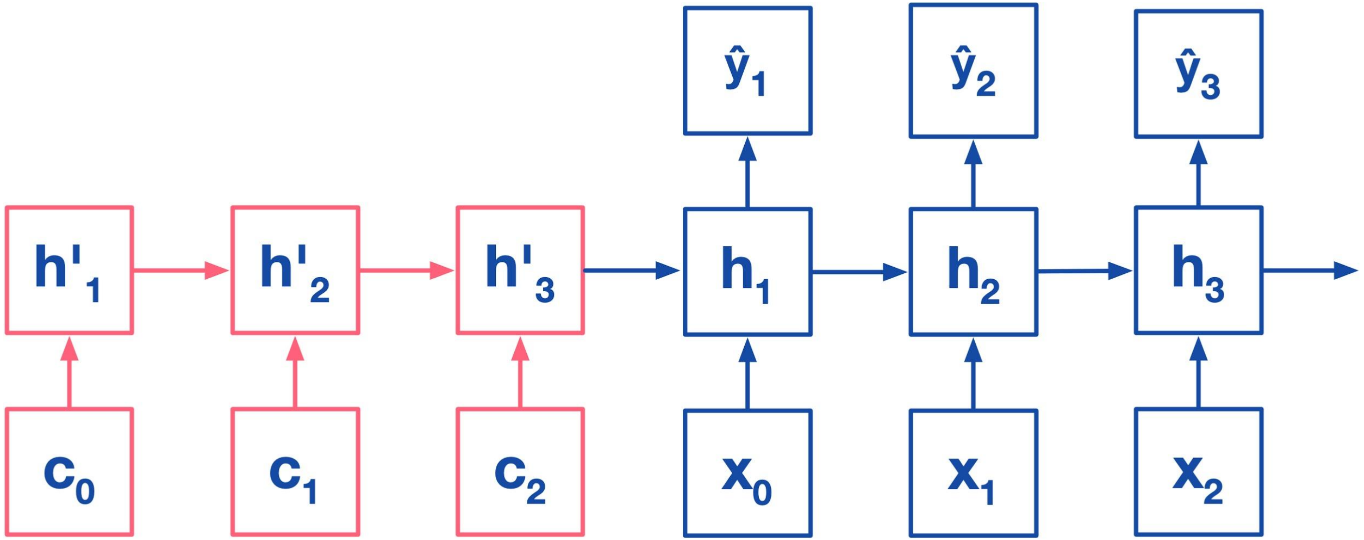
Reed et al. “Parallel Multiscale Autoregressive Density Estimation.”

Natural language as sequences

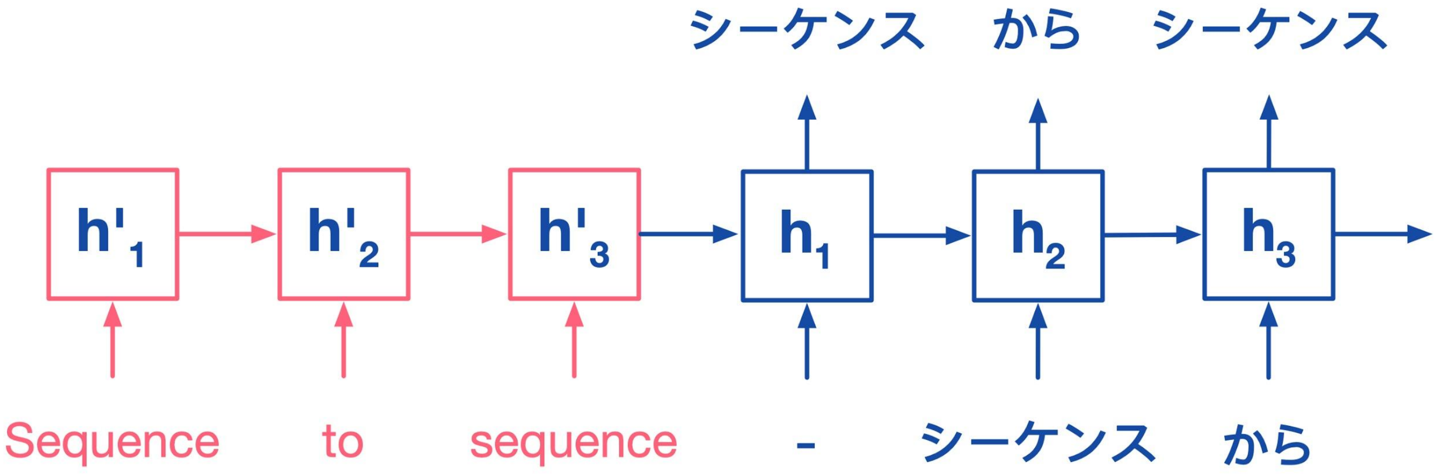
Language as sequences: Sequence-to-sequence models



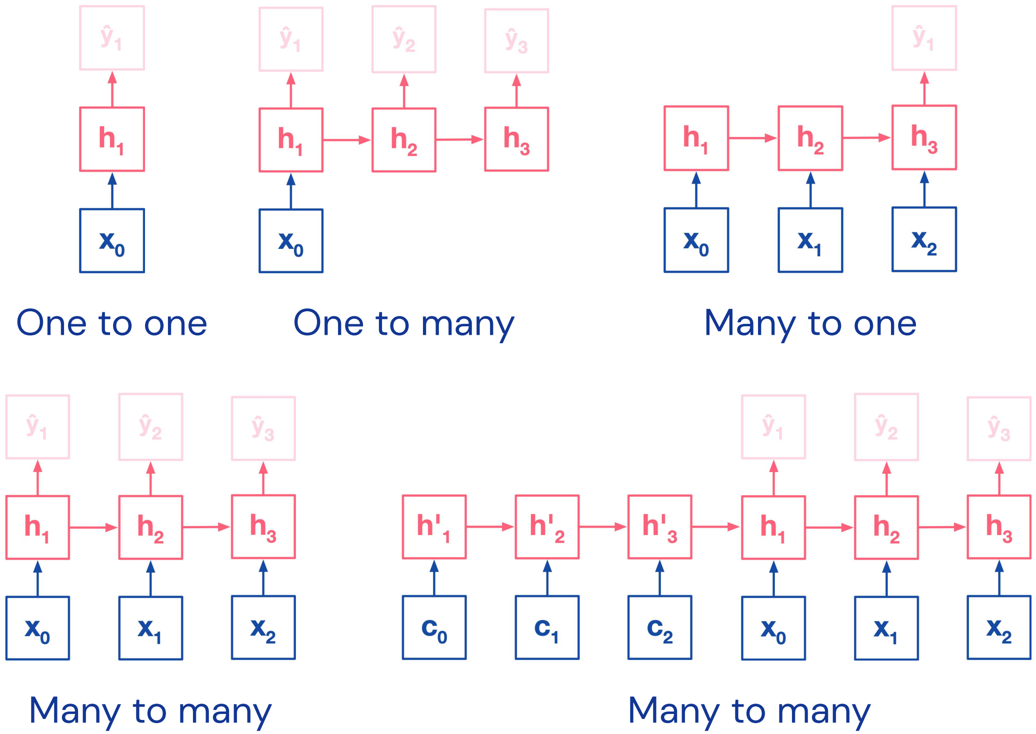
Language as sequences:
Sequence-to-sequence models



Language as sequences:
Sequence-to-sequence models



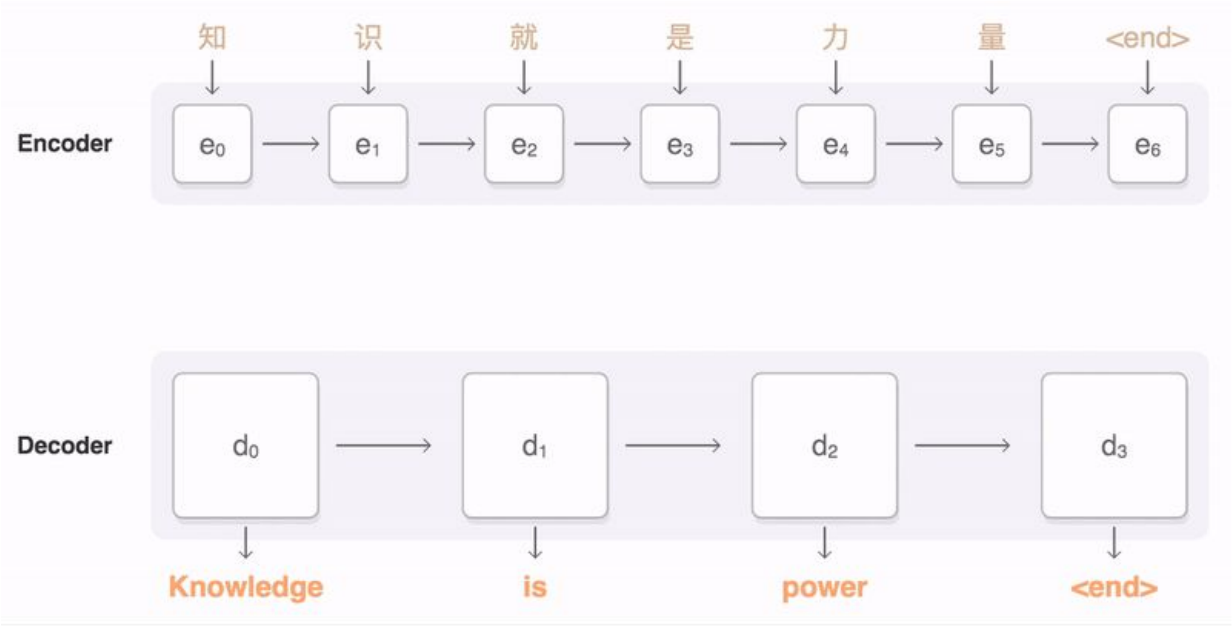
This setup is flexible



Seq2seq has a wide range of applications

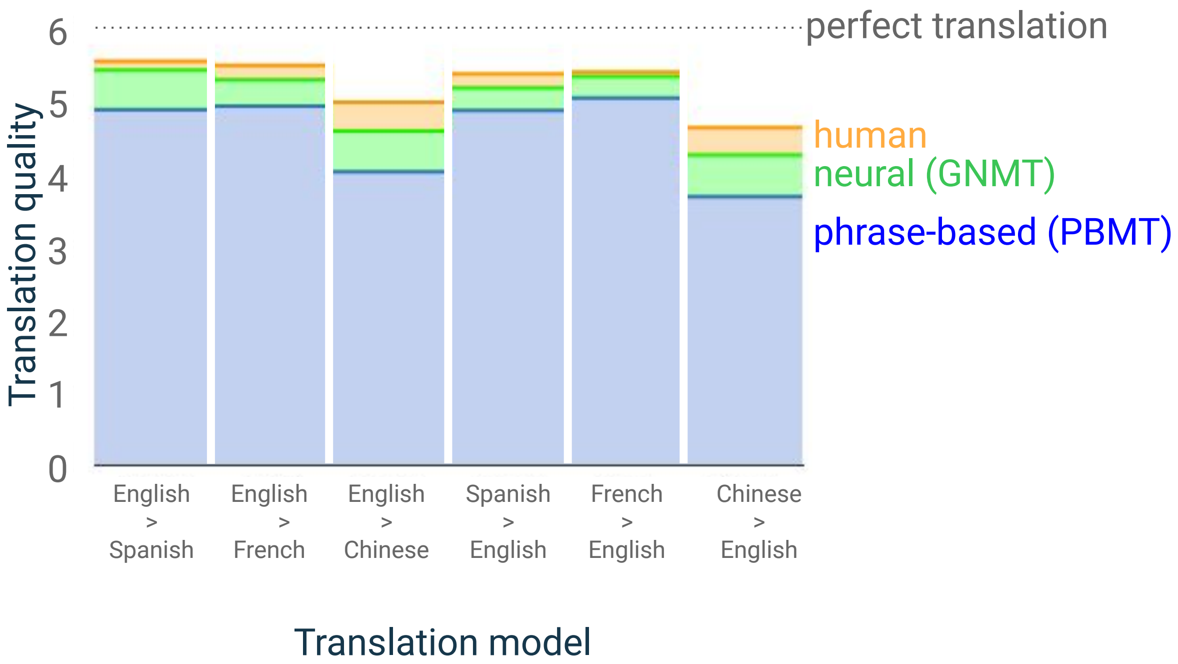
1. **MT** [Kalchbrenner et al, EMNLP 2013][Cho et al, EMLP 2014][Sutskever & Vinyals & Le, NIPS 2014][Luong et al, ACL 2015][Bahdanau et al, ICLR 2015]
2. **Image captions** [Mao et al, ICLR 2015][Vinyals et al, CVPR 2015][Donahue et al, CVPR 2015][Xu et al, ICML 2015]
3. **Speech** [Chorowsky et al,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

Google Neural Machine Translation



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

Google Neural Machine Translation



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

Image captioning

$$p(\text{language}_1 \mid \text{language}_2) \rightarrow p(\text{language}_1 \mid \text{image})$$

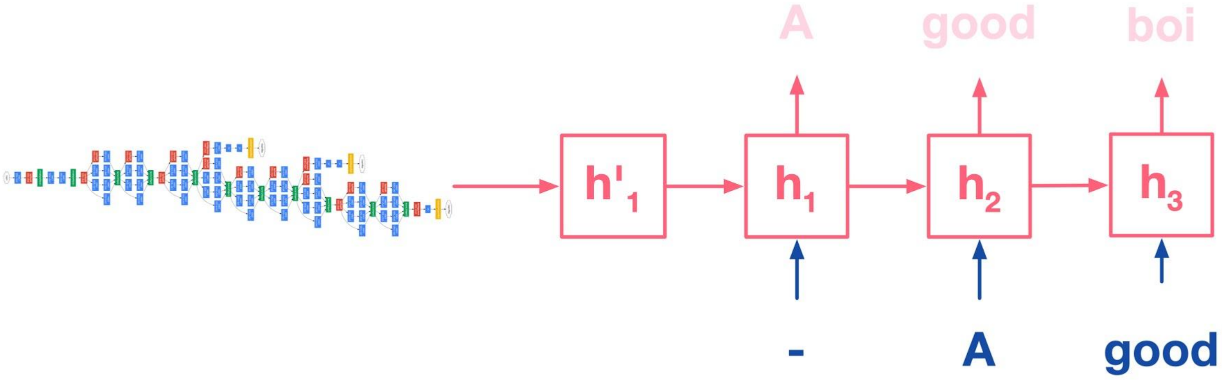


Image captioning

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

Best Model: A small dog is sitting on a chair.

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

Image captioning

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

Best Model: A man is holding a sandwich in his hand.

Initial Model: A man cutting a cake with a knife.

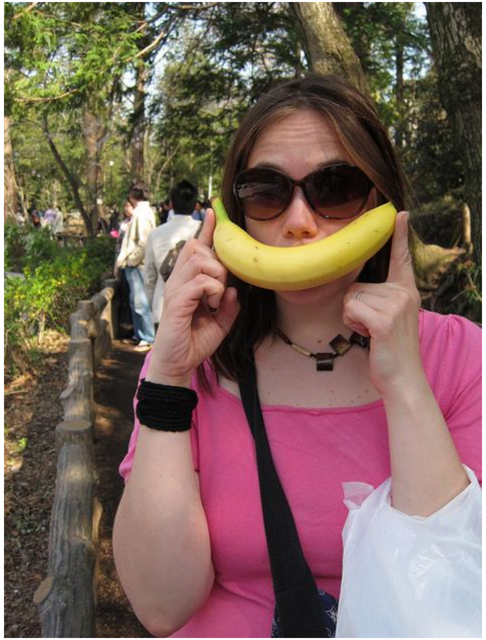
Image captioning

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

Best Model: A person is cooking some food on a grill.

Initial Model: A pizza sitting on top of a white plate.

Image captioning



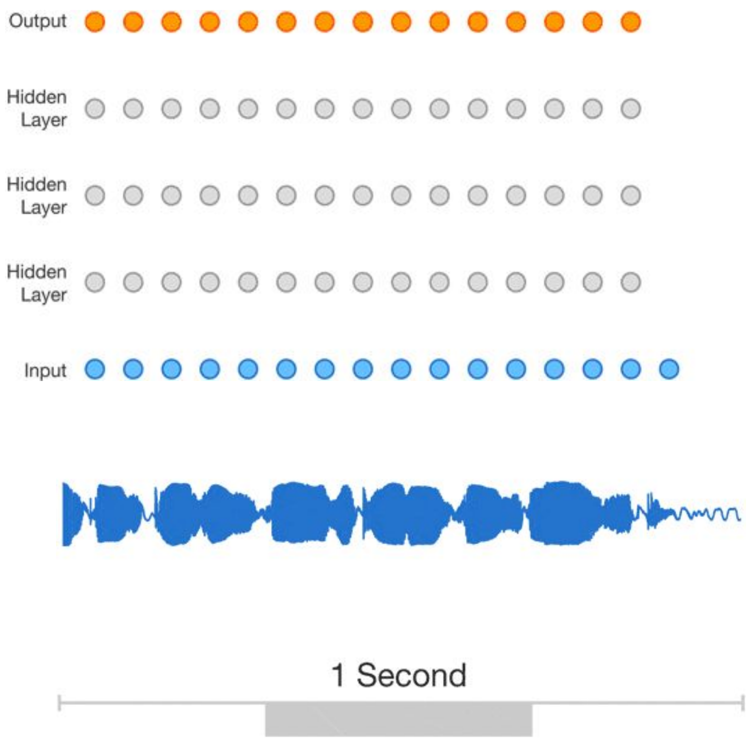
Human: A woman holding up a yellow banana to her face.

Best Model: A woman holding a banana up to her face.

Initial Model: A close up of a person eating a hot dog.

Audio waves as sequences

Audio waves as sequences: convolutions



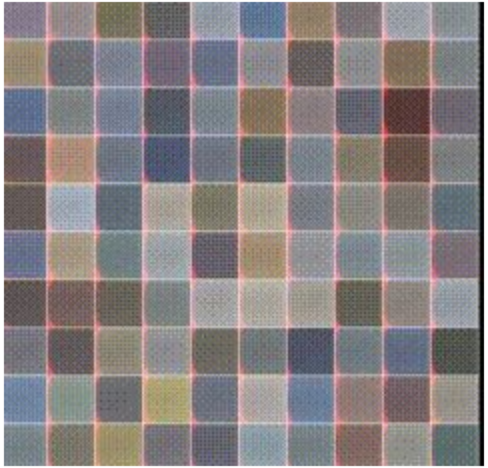
Van den Oord et al. (2016)

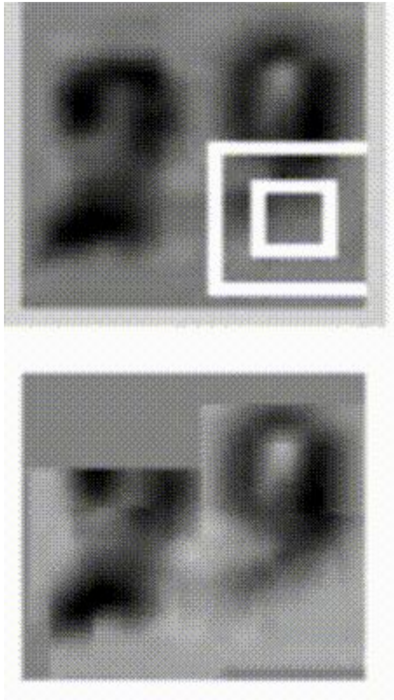
Properties of Convolutions as f_θ

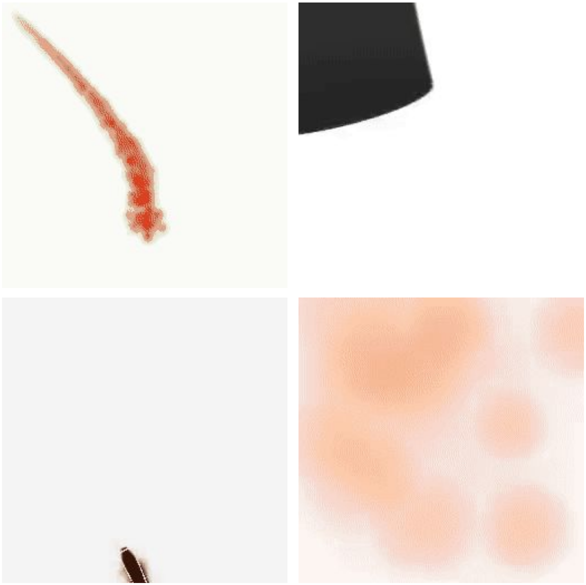
	N-gram	Addition	RNN	LSTM	Conv
Order matters	✔	✘	✔	✔	✔
Variable length	✘	✔	✔	✔	✔
Differentiable	✘	✔	✔	✔	✔
Pairwise encoding	✔	✘	✘	✘	✘
Preserves long-term	✘	✔	✘	✔	✔

Policies as sequences

Policies as sequences





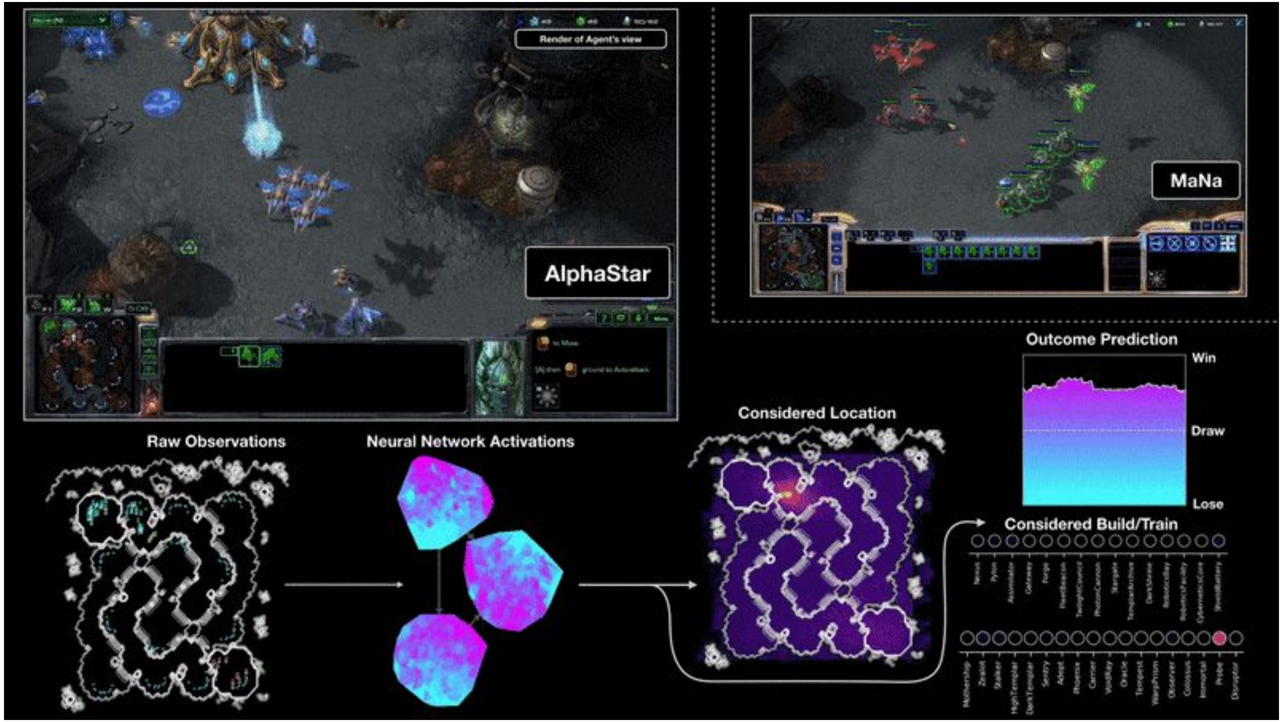


Ba et al. (2014), Gregor et al (2015) , Mellor et al (2019)

OpenAI five

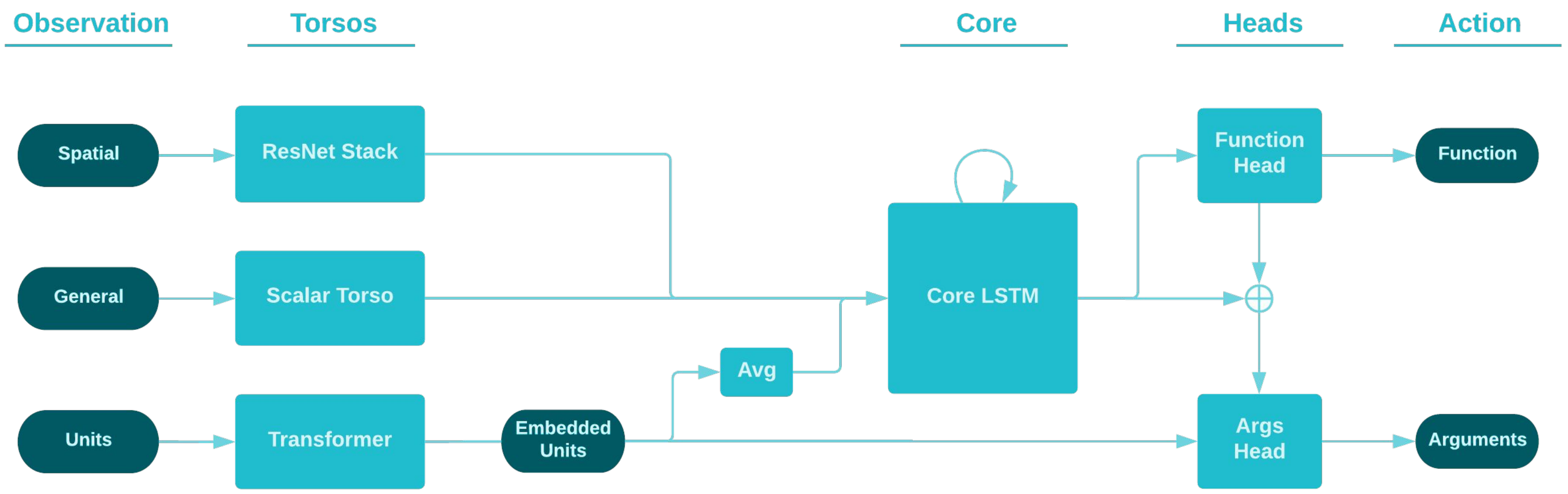
DOTA 2 IGDB Presskit

Alphastar



Render of Agent's view
AlphaStar
MaNa
Outcome Prediction
Win
Draw
Lose
Considered Location
Raw Observations
Neural Network Activations
Considered Build/Train

AlphaStar Architecture



Attention for sequences: transformers

Transformers

Convolution

Transformer

Vaswani et al. (2017)

Transformers

Vaswani et al. (2017)

Transformers

Vaswani et al. (2017)

Transformers

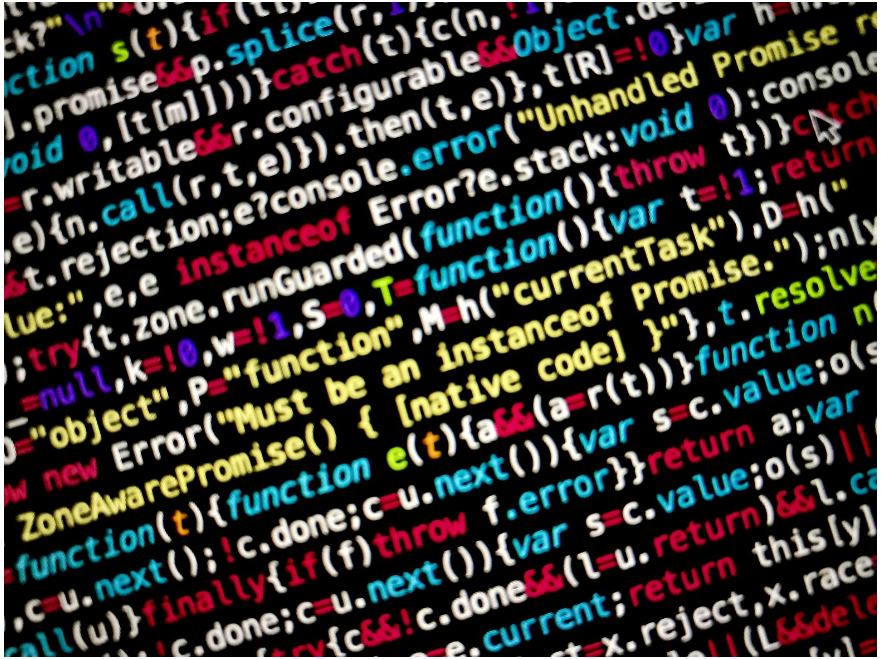
Vaswani et al. (2017)

Transformers

Vaswani et al. (2017)

GPT2

- Transformer-based language model with 1.5 billion parameters for next-word prediction
- Dataset: 40GB of text data from 8M websites
- Adapts to **style** and **content** of arbitrary conditioning input



https://openai.com/blog/better-language-models/#sample1

Radford et al. (2019)

GPT2

SYSTEM PROMPT (HUMAN-WRITTEN)	*In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.*
MODEL COMPLETION (MACHINE-WRITTEN, 10 TRIES)	The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science. Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved. Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow. Pérez and the others then ventured further into the valley. "By the time we reached the top of one peak, the water looked blue, with some crystals on top," said Pérez. Pérez and his friends were astonished to see the unicorn herd. These creatures could be seen from the air without having to move too much to see them – they were so close they could touch their horns.

Properties of Transformers as f_θ

	N-gram	Addition	RNN	LSTM	Conv	Transf.
Order matters	✔	✘	✔	✔	✔	✔
Variable length	✘	✔	✔	✔	✔	✔
Differentiable	✘	✔	✔	✔	✔	✔
Pairwise encoding	✔	✘	✘	✘	✘	✔
Preserves long-term	✘	✔	✘	✔	✔	✔

Evolution of language modeling

Sutskever et al, 2011, RNNs

[ ] while he was giving attention to the second advantage of school building a 2-for-2 stool killed by the Cultures saddled with a halfsuit defending the Bharatiya Fernall 's office

Radford et al, 2019, GPT2

[Miley Cyrus was caught shoplifting from Abercrombie and Fitch on Hollywood Boulevard today.] The singer was wearing a black hoodie with the label 'Blurred Lines' on the front and 'Fashion Police' on the back.

Summary

1. Motivation: Sequences are everywhere but modeling them is hard!
2. Covered different approaches:
   a. N-Grams
   b. RNNs
   c. LSTMs & GRUs
   d. Dilated convolutions and Transformers
3. These models are flexible and can be applied to a wide range of tasks across machine learning

Thank you

Questions