Lecture 9
Recurrent Neural Networks

"I'm glad that I'm Turing Complete now"

Xinyu Zhou
Megvii (Face++) Researcher
zxy@megvii.com
Nov 2017

Raise your hand and ask,
whenever you have questions...

We have a lot to cover
and
DON’T BLINK

Outline

- RNN Basics
- Classic RNN Architectures
  - LSTM
  - RNN with Attention
  - RNN with External Memory
    - Neural Turing Machine
    - CAVEAT: *don't fall asleep*
- Applications
  - A market of RNNs

RNN Basics

Feedforward Neural Networks

- Feedforward neural networks can fit any bounded continuous (compact) function
- This is called **Universal approximation theorem**

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

Cybenko, George. "Approximation by superpositions of a sigmoidal function." Mathematics of Control, Signals, and Systems (MCSS) 2.4 (1989): 303-314.

Bounded Continuous Function is NOT ENOUGH!

How to solve Travelling Salesman Problem?

Bounded Continuous Function is NOT ENOUGH!

How to solve Travelling Salesman Problem?

We Need to be Turing Complete

RNN is Turing Complete

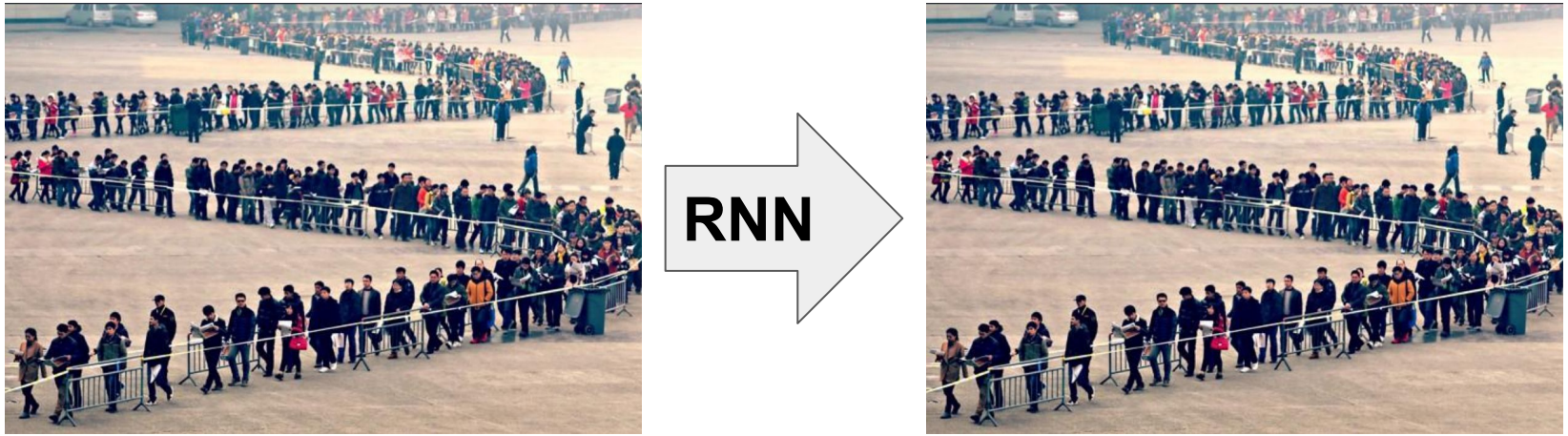
This paper deals with finite size networks which consist of interconnections of synchronously evolving processors. Each processor updates its state by applying a "sigmoidal" function to a linear combination of the previous states of all units. We prove that one may simulate all Turing machines by such nets. In particular, one can simulate any multi-stack Turing machine in real time, and there is a net made up of 886 processors which computes a universal partial-recursive function. Products (high order nets) are not required, contrary to what had been stated in the literature. Non-deterministic Turing machines can be simulated by non-deterministic rational nets, also in real time. The simulation result has many consequences regarding the decidability, or more generally the complexity, of questions about recursive nets.

Siegelmann, Hava T., and Eduardo D. Sontag. "On the computational power of neural nets." Journal of computer and system sciences 50.1 (1995): 132-150.

Sequence Modeling

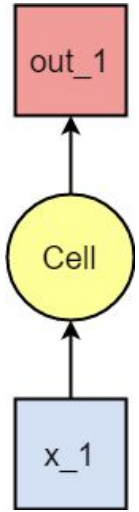
Sequence Modeling

- How to take a **variable length sequence** as **input**?
- How to predict a **variable length sequence** as **output**?



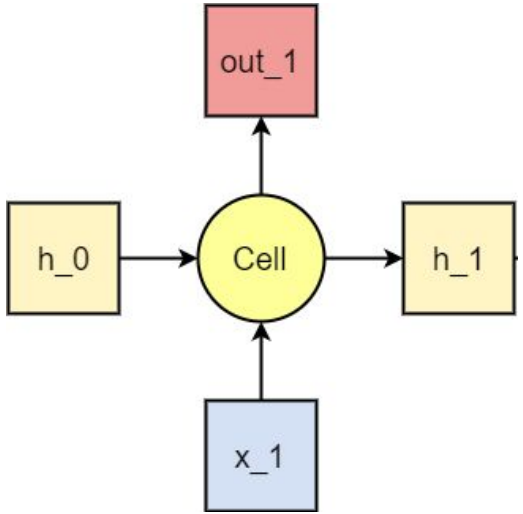
RNN Diagram

A lonely feedforward cell



RNN Diagram

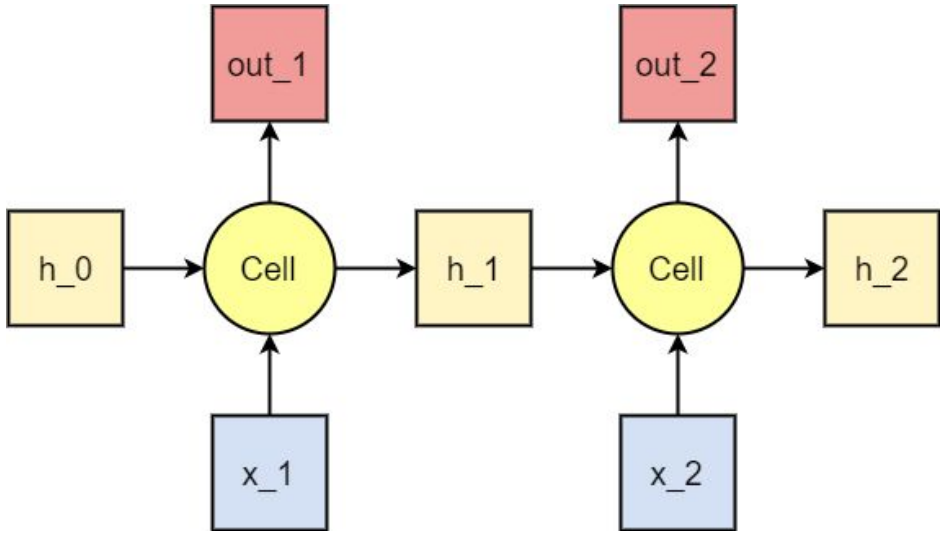
Grows … with more inputs and outputs



RNN Diagram

… here comes a brother

(x_1, x_2) comprises a length-2 sequence

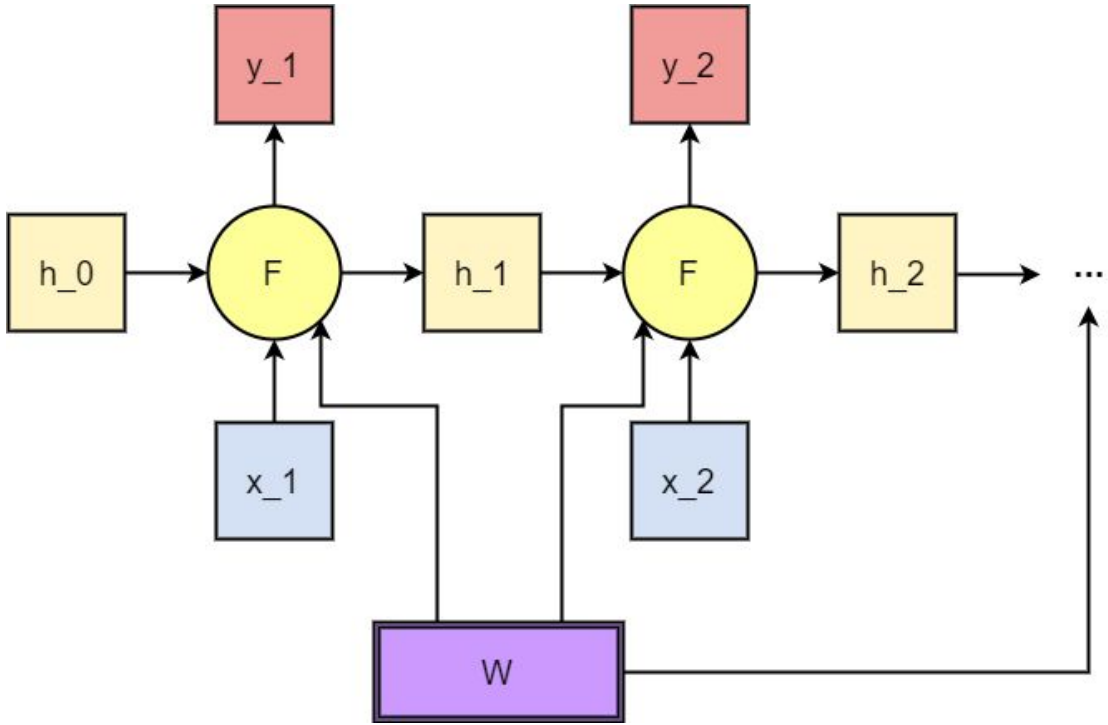


RNN Diagram

… with shared (tied) weights

$$(h_1, y_1) = F(h_0, x_1, W)$$
$$(h_2, y_2) = F(h_1, x_2, W)$$

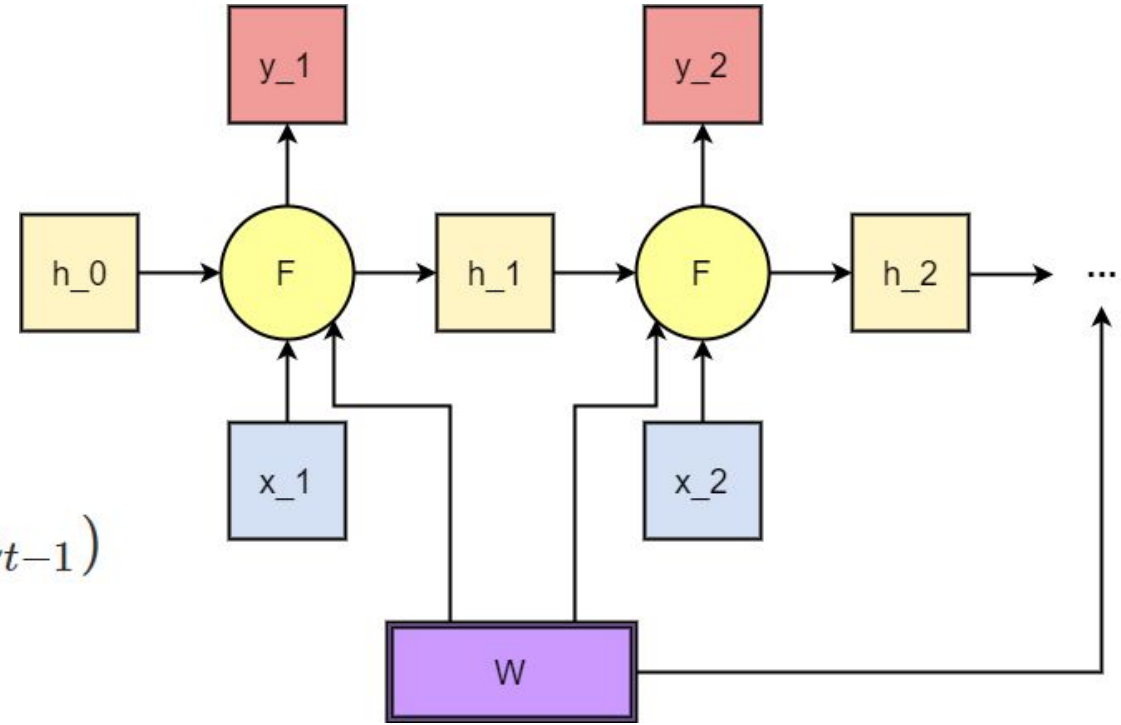
x_i: inputs
y_i: outputs
W: all the same
h_i: internal states that passed along
F: a “pure” function



RNN Diagram

… with shared (tied) weights

$$(h_1, y_1) = F(h_0, x_1, W)$$
$$(h_2, y_2) = F(h_1, x_2, W)$$



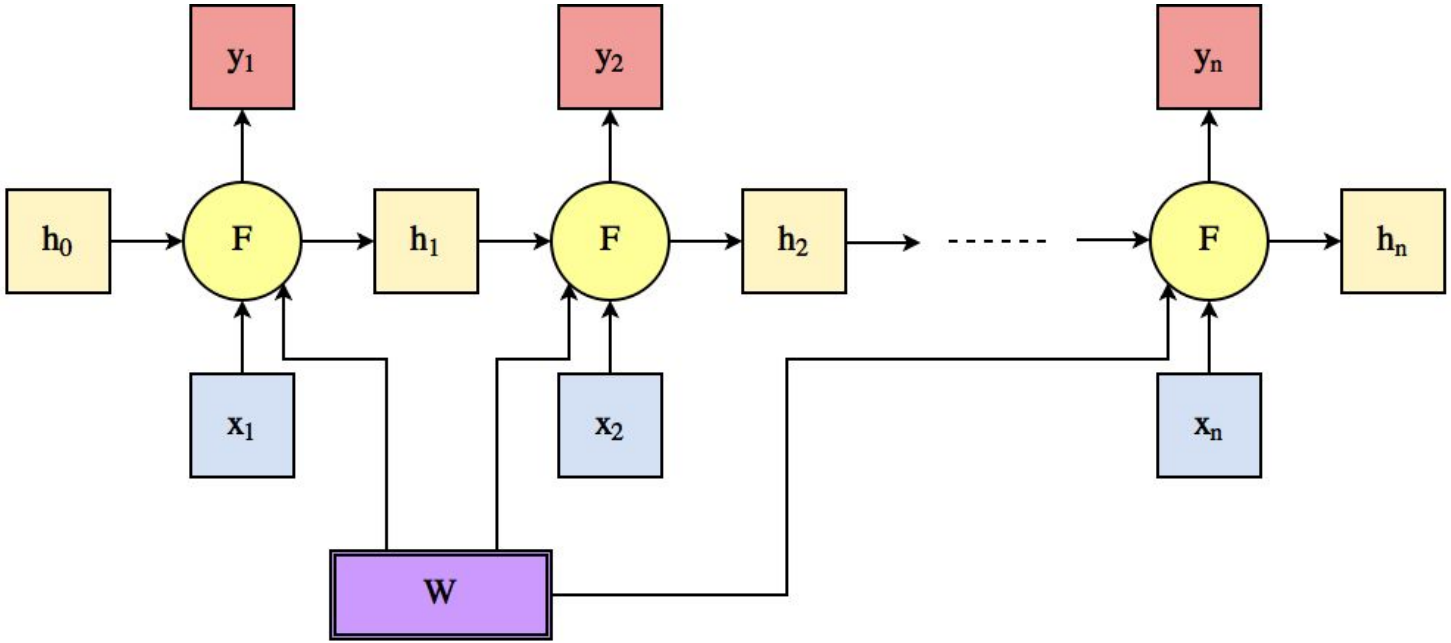
A simple implementation of F

$$h_t = tanh(W_{ih}x_t + W_{hh}h_{t-1})$$
$$y_t = W_{ho}h_t$$

Categorize RNNs by input/output types

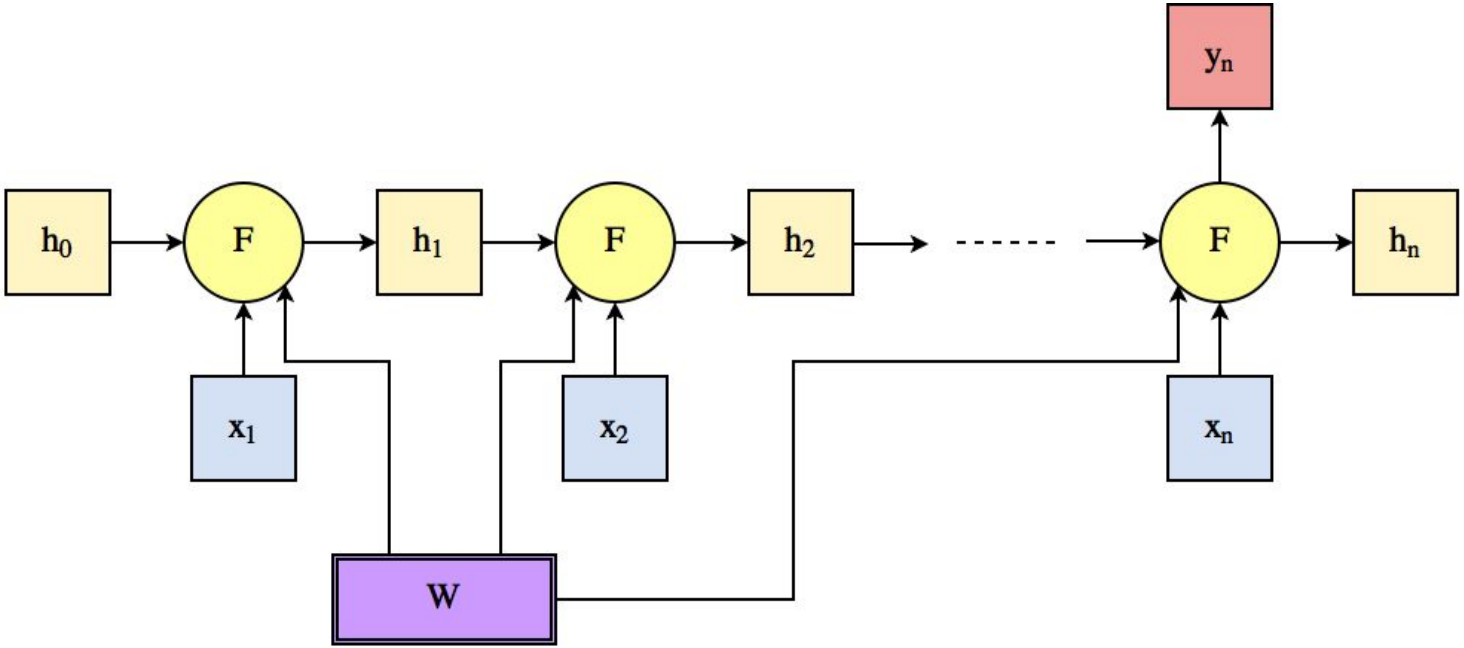
Categorize RNNs by input/output types

Many-to-many



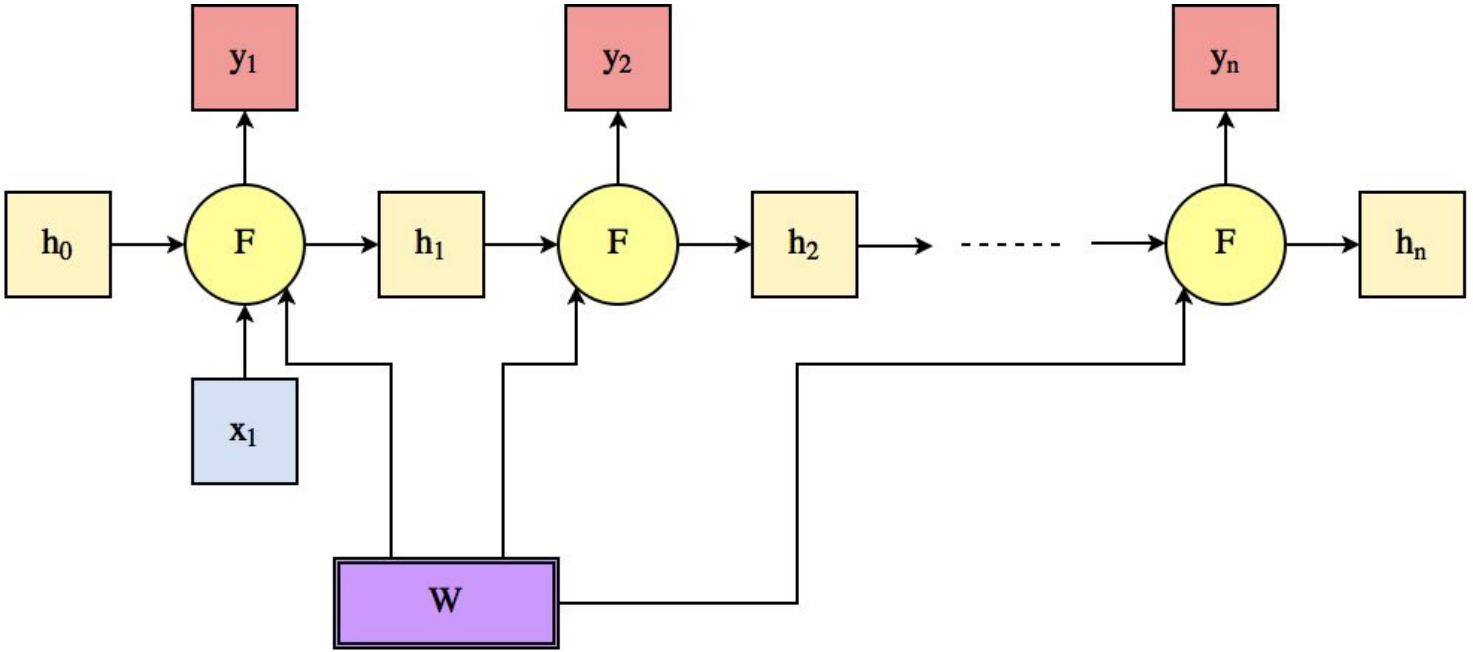
Categorize RNNs by input/output types

Many-to-one



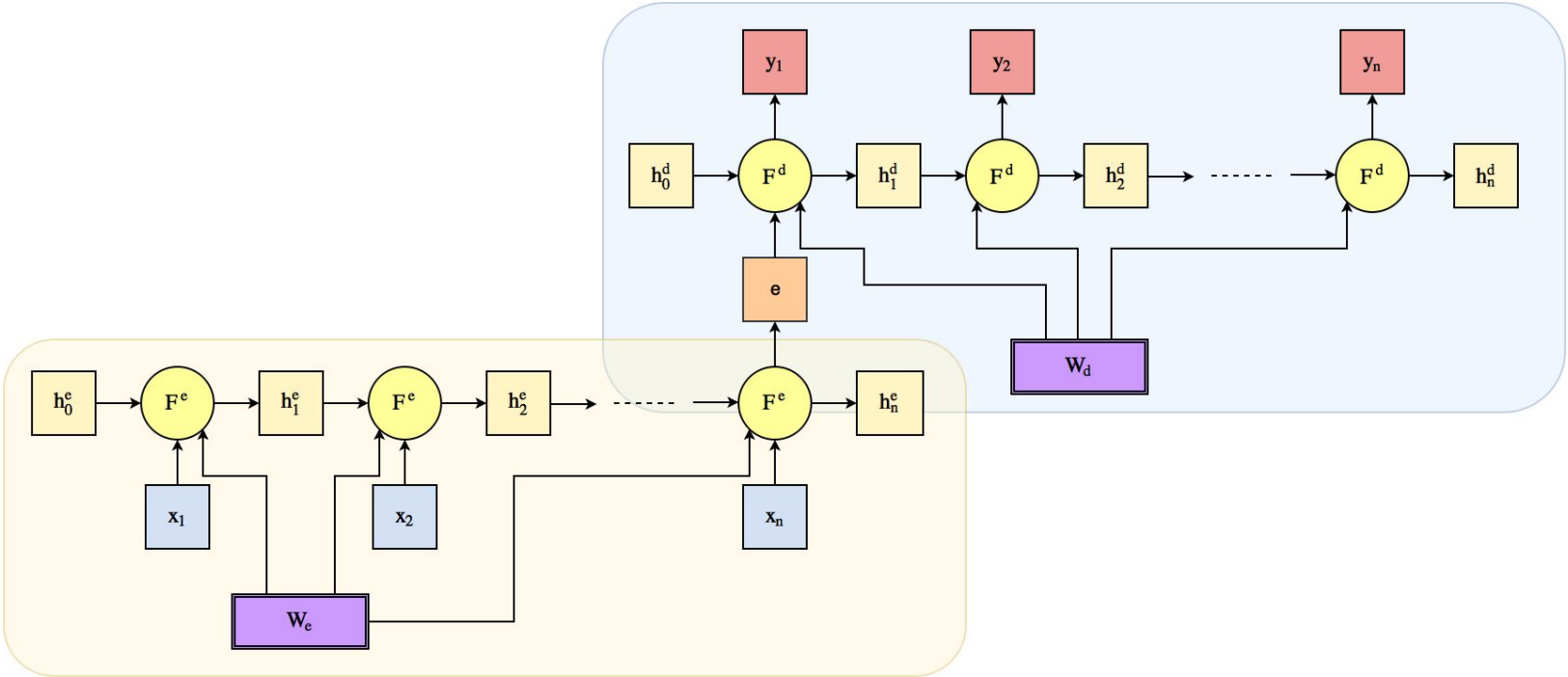
Categorize RNNs by input/output types

One-to-Many



Categorize RNNs by input/output types

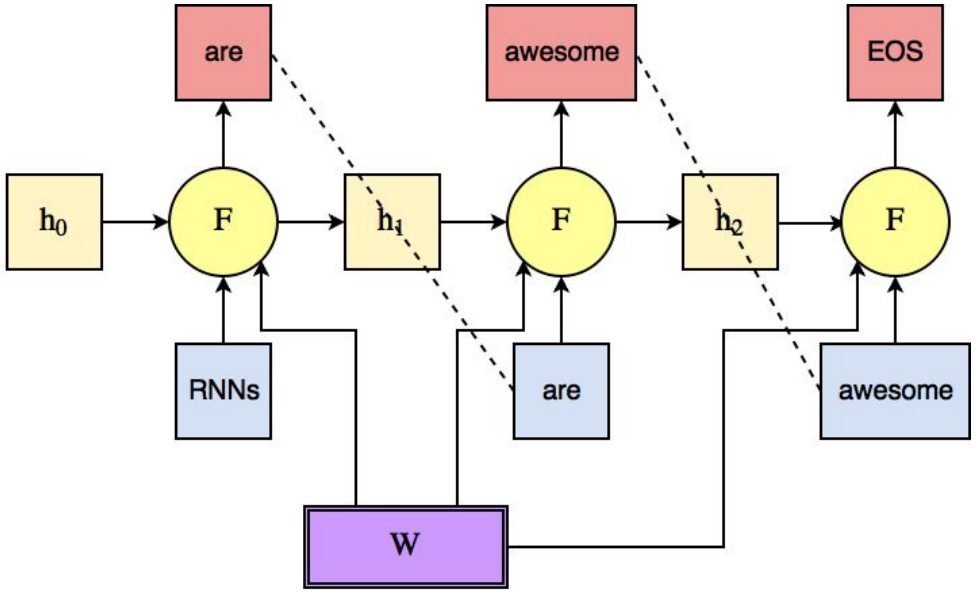
Many-to-Many: Many-to-One + One-to-Many



Many-to-Many Example

Language Model

- Predict next word given previous words
- “h” → “he” → “hel” → “hell” → “hell



$$P(w_1, w_2, \ldots, w_t) = P(w_1)P(w_2|w_1)P(w_3|w_{1:2}) \ldots P(w_t|w_{1:t-1})$$

$$Loss = \sum_{i=1..t} \text{cross-entropy}(y_i, y_i^*)$$

Language Modeling

- Tell story
-
- “Heeeeeel”
- ⇒ “Heeeloolllell”
- ⇒ “Hellooo”
- ⇒ “Hello”

```
tyntd-iafhatawiaoihrdemot  lytdws  e ,tfti, astai f ogoh eoase rrranbyne 'nhthnee e
plia tklrgd t o idoe ns,smtt   h ne etie h,hregtrs nigtike,aoaenns lng
```

train more

```
"Tmont thithey" fomesscerliund
Keushey. Thom here
sheulke, anmerenith ol sivh I lalterthend Bleipile shuwy fil on aseterlome
coaniogennc Phe lism thond hon at. MeiDimorotion in ther thize."
```

train more

```
Aftair fall unsuch that the hall for Prince Velzonski's that me of
her hearly, and behs to so arwage fiving were to it beloge, pavu say falling misfort
how, and Gogition is so overelical and ofter.
```

train more

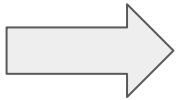
```
"Why do what that day," replied Natasha, and wishing to himself the fact the
princess, Princess Mary was easier, fed in had oftened him.
Pierre aking his soul came to the packs and drove up his father-in-law women.
```

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

Language Modeling

- Write (nonsense) book in latex

```
\begin{proof}
We may assume that $\mathcal{I}$ is an abelian sheaf on
$\mathcal{C}$.
\item Given a morphism $\Delta : \mathcal{F} \to \mathcal{I}$
is an injective and let $\mathfrak q$ be an abelian sheaf on $X$.
Let $\mathcal{F}$ be a fibered complex. Let $\mathcal{F}$ be a
category.
\begin{enumerate}
\item \hyperref[setain-construction-phantom]{Lemma}
\label{lemma-characterize-quasi-finite}
Let $\mathcal{F}$ be an abelian quasi-coherent sheaf on
$\mathcal{C}$.
Let $\mathcal{F}$ be a coherent $\mathcal{O}_X$-module. Then
$\mathcal{F}$ is an abelian catenary over $\mathcal{C}$.
\item The following are equivalent
\begin{enumerate}
\item $\mathcal{F}$ is an $\mathcal{O}_X$-module.
\end{lemma}
```



To prove study we see that $\mathcal{F}|_U$ is a covering of $\mathcal{X}'$, and $\mathcal{T}_i$ is an object of $\mathcal{F}_{X/S}$ for $i > 0$ and $\mathcal{F}_p$ exists and let $\mathcal{F}_i$ be a presheaf of $\mathcal{O}_X$-modules on $\mathcal{C}$ as a $\mathcal{F}$-module. In particular $\mathcal{F} = U/\mathcal{F}$ we have to show that

$$\widetilde{M}^\bullet = \mathcal{I}^\bullet \otimes_{\mathrm{Spec}(k)} \mathcal{O}_{S,s} - i_X^{-1}\mathcal{F})$$

is a unique morphism of algebraic stacks. Note that

$$\mathrm{Arrows} = (Sch/S)^{opp}_{fppf}, (Sch/S)_{fppf}$$

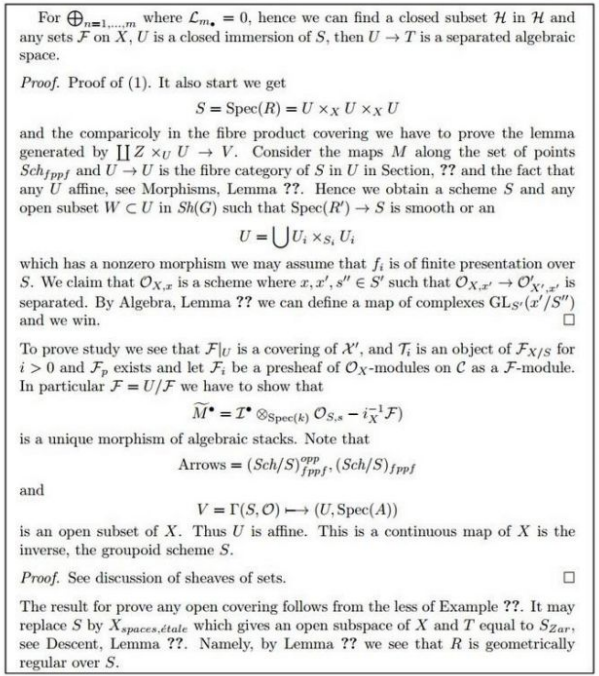
and

$$V = \Gamma(S, \mathcal{O}) \longmapsto (U, \mathrm{Spec}(A))$$

is an open subset of X. Thus U is affine. This is a continuous map of X is the inverse, the groupoid scheme S.

Language Modeling

- Write (nonsense) book in latex



For $\bigoplus_{n=1,\ldots,m}$ where $\mathcal{L}_{m_\bullet} = 0$, hence we can find a closed subset $\mathcal{H}$ in $\mathcal{H}$ and any sets $\mathcal{F}$ on X, U is a closed immersion of S, then $U \to T$ is a separated algebraic space.

Proof. Proof of (1). It also start we get

$$S = \mathrm{Spec}(R) = U \times_X U \times_X U$$

and the compariciy in the fibre product covering we have to prove the lemma generated by $\coprod Z \times_U U \to V$. Consider the maps M along the set of points Sch_{fppf} and $U \to U$ is the fibre category of S in U in Section, ?? and the fact that any U affine, see Morphisms, Lemma ??. Hence we obtain a scheme S and any open subset $W \subset U$ in $Sh(G)$ such that $\mathrm{Spec}(R') \to S$ is smooth or an

$$U = \bigcup U_i \times_{S_i} U_i$$

which has a nonzero morphism we may assume that f_i is of finite presentation over S. We claim that $\mathcal{O}_{X,x}$ is a scheme where $x, x', s'' \in S'$ such that $\mathcal{O}_{X,x'} \to \mathcal{O}'_{X',x'}$ is separated. By Algebra, Lemma ?? we can define a map of complexes $\mathrm{GL}_{S'}(x'/S'')$ and we win. □

To prove study we see that $\mathcal{F}|_U$ is a covering of $\mathcal{X}'$, and $\mathcal{T}_i$ is an object of $\mathcal{F}_{X/S}$ for $i > 0$ and $\mathcal{F}_p$ exists and let $\mathcal{F}_i$ be a presheaf of $\mathcal{O}_X$-modules on $\mathcal{C}$ as a $\mathcal{F}$-module. In particular $\mathcal{F} = U/\mathcal{F}$ we have to show that

$$\widetilde{M}^\bullet = \mathcal{I}^\bullet \otimes_{\mathrm{Spec}(k)} \mathcal{O}_{S,s} - i_X^{-1}\mathcal{F})$$

is a unique morphism of algebraic stacks. Note that

$$\mathrm{Arrows} = (Sch/S)^{opp}_{fppf}, (Sch/S)_{fppf}$$

and

$$V = \Gamma(S, \mathcal{O}) \longmapsto (U, \mathrm{Spec}(A))$$

is an open subset of X. Thus U is affine. This is a continuous map of X is the inverse, the groupoid scheme S.

Proof. See discussion of sheaves of sets. □

The result for prove any open covering follows from the less of Example ??. It may replace S by $X_{spaces,étale}$ which gives an open subspace of X and T equal to S_{Zar}, see Descent, Lemma ??. Namely, by Lemma ?? we see that R is geometrically regular over S.

Lemma 0.1. *Assume (3) and (3) by the construction in the description.*

Suppose $X = \lim |X|$ *(by the formal open covering* X *and a single map* $\underline{Proj}_X(\mathcal{A}) = \mathrm{Spec}(B)$ *over* U *compatible with the complex*

$$Set(\mathcal{A}) = \Gamma(X, \mathcal{O}_{X,\mathcal{O}_X}).$$

When in this case of to show that $\mathcal{Q} \to \mathcal{C}_{Z/X}$ *is stable under the following result in the second conditions of (1), and (3). This finishes the proof. By Definition ?? (without element is when the closed subschemes are catenary. If* T *is surjective we may assume that* T *is connected with residue fields of* S*. Moreover there exists a closed subspace* $Z \subset X$ *of* X *where* U *in* X' *is proper (some defining as a closed subset of the uniqueness it suffices to check the fact that the following theorem*

(1) f *is locally of finite type. Since* $S = \mathrm{Spec}(R)$ *and* $Y = \mathrm{Spec}(R)$.

Proof. This is form all sheaves of sheaves on X. But given a scheme U and a surjective étale morphism $U \to X$. Let $U \cap U = \coprod_{i=1,\ldots,n} U_i$ be the scheme X over S at the schemes $X_i \to X$ and $U = \lim_i X_i$. □

The following lemma surjective restrocomposes of this implies that $\mathcal{F}_{x_0} = \mathcal{F}_{x_0} = \mathcal{F}_{\mathcal{X},\ldots,0}$.

Lemma 0.2. *Let* X *be a locally Noetherian scheme over* S, $E = \mathcal{F}_{X/S}$. *Set* $\mathcal{I} = \mathcal{J}_1 \subset \mathcal{I}'_n$. *Since* $\mathcal{I}^n \subset \mathcal{I}^n$ *are nonzero over* $i_0 \leq \mathfrak{p}$ *is a subset of* $\mathcal{J}_{n,0} \circ \overline{A}_2$ *works.*

Lemma 0.3. *In Situation ??. Hence we may assume* $\mathfrak{q}' = 0$.

Proof. We will use the property we see that $\mathfrak{p}$ is the mext functor (??). On the other hand, by Lemma ?? we see that

$$D(\mathcal{O}_{X'}) = \mathcal{O}_X(D)$$

where K is an F-algebra where δ_{n+1} is a scheme over S. □

Many-to-One Example

Sentiment analysis

- “RNNs are awesome!” ⇒ 
- “The course project is too hard for me.” ⇒ 

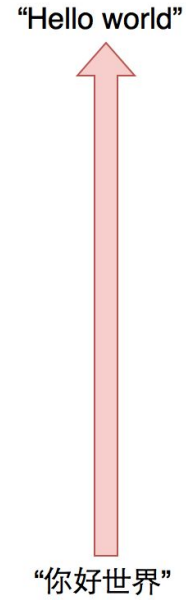
Many-to-One Example

Sentiment analysis

- “RNNs are awesome!” ⇒ 
- “The course project is too hard for me.” ⇒ 

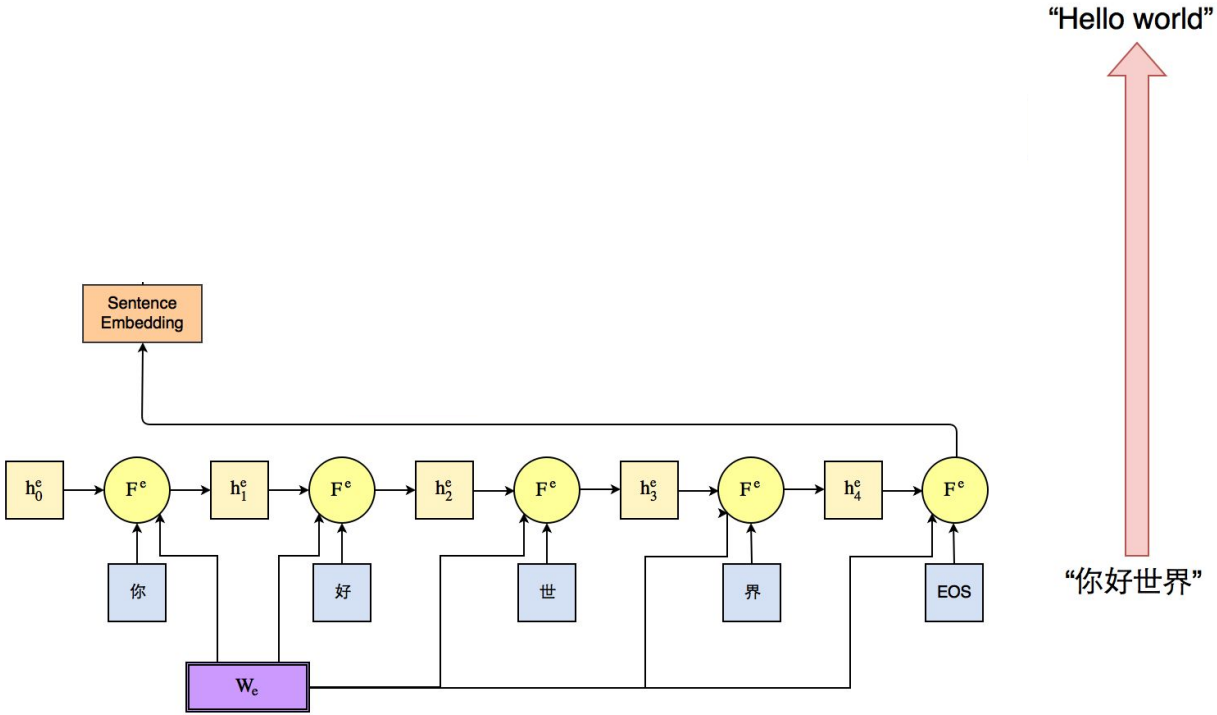
Many-to-One + One-to-Many

Neural Machine Translation



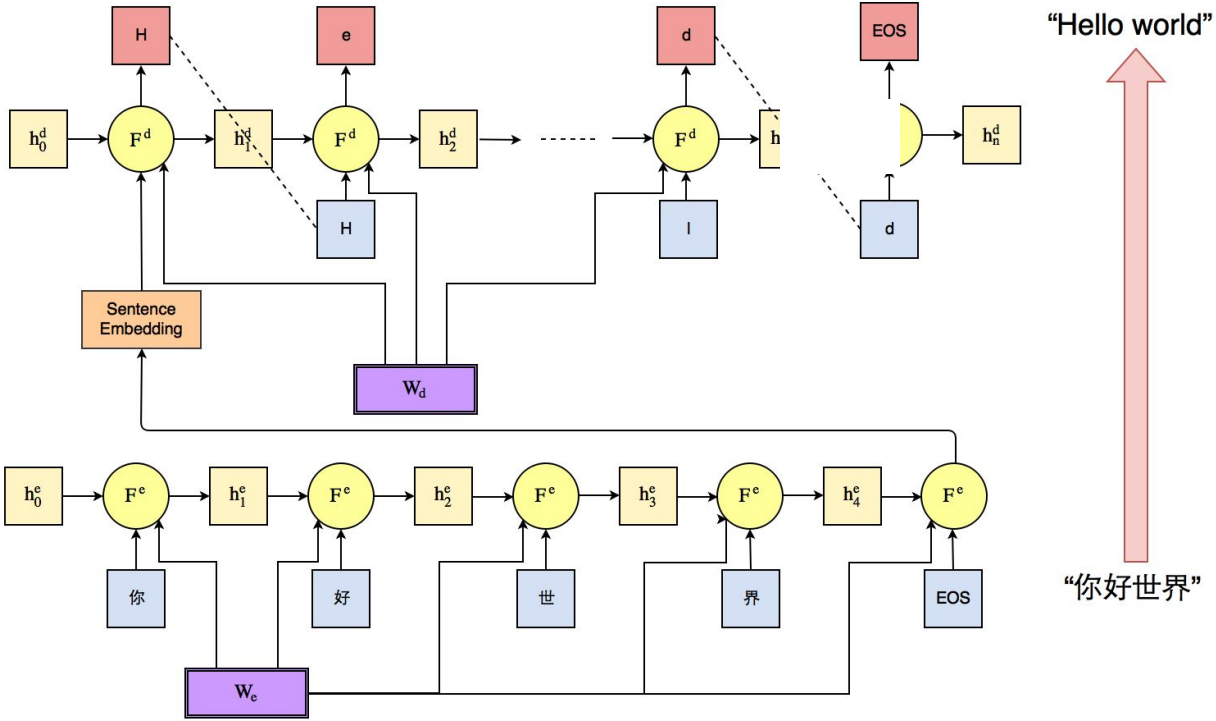
Many-to-One + One-to-Many

Neural Machine Translation



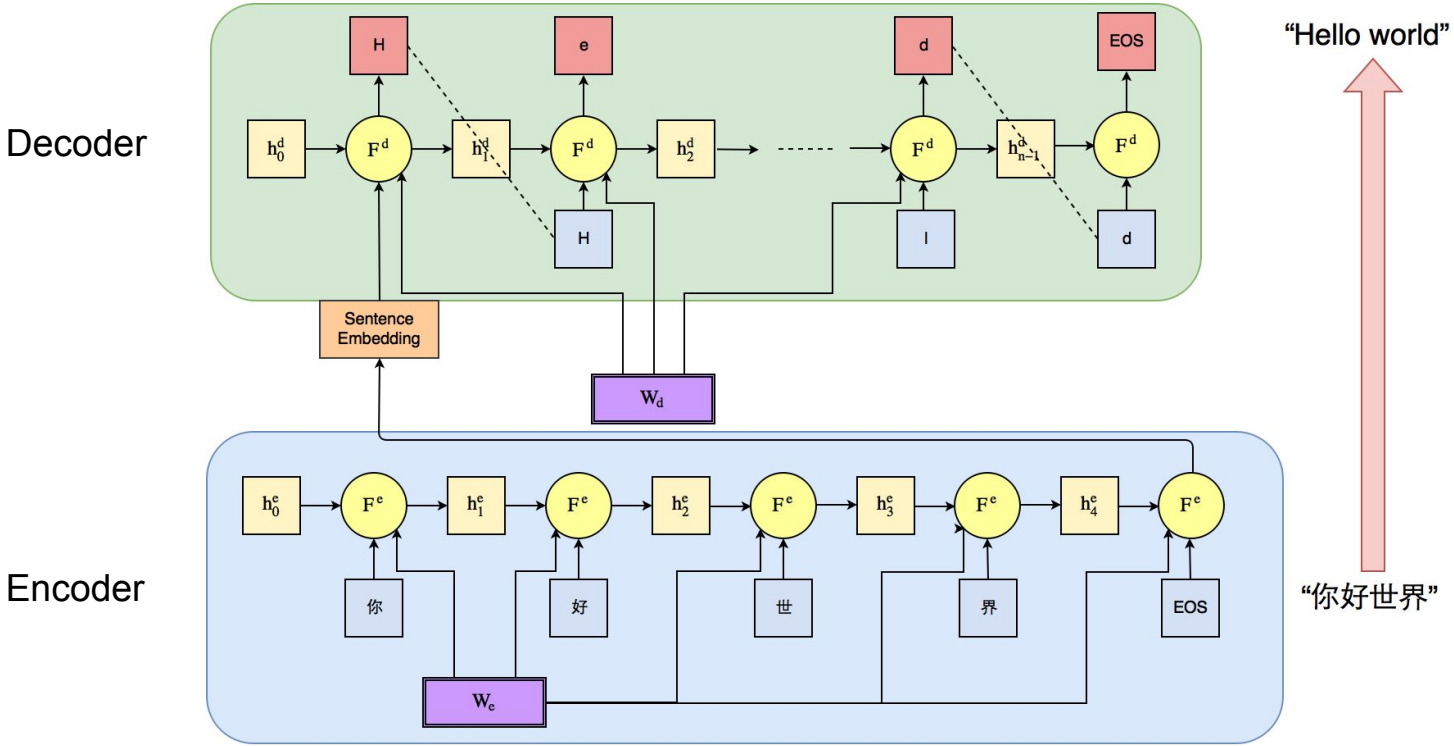
Many-to-One + One-to-Many

Neural Machine Translation



Many-to-One + One-to-Many

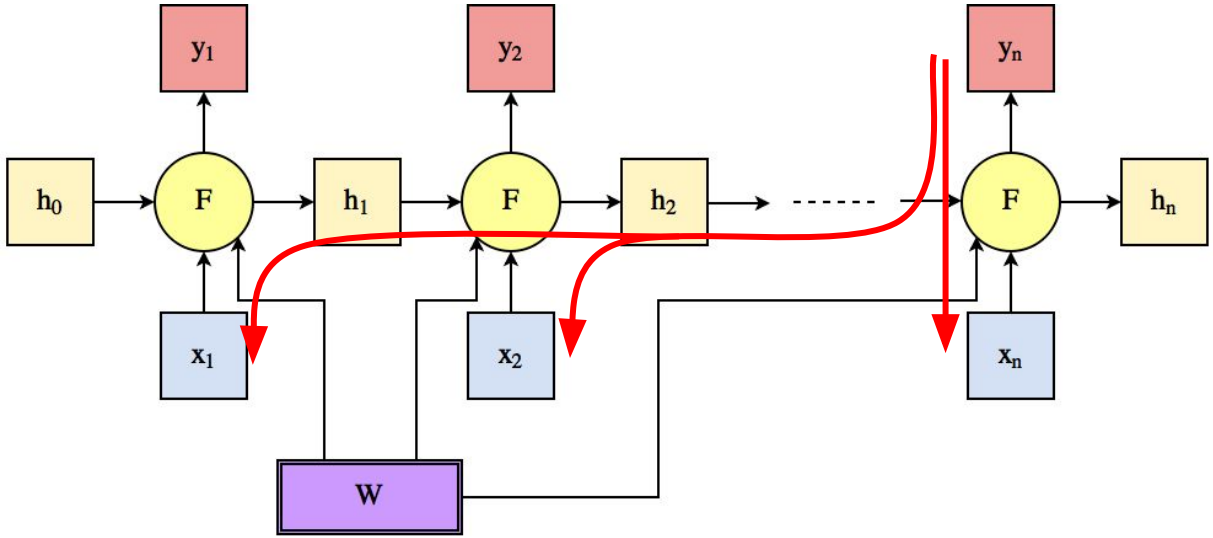
Neural Machine Translation



Vanishing/Exploding Gradient Problem

Training RNN

- “Backpropagation Through Time”
  - Truncated BPTT
- The chain rule of differentiation
  - Just Backpropagation



Vanishing/Exploding Gradient Problem

- Consider a ***linear*** recurrent net with zero inputs
- $$h_t = W_{hh} h_{t-1} = h_0 W_{hh}^t$$

Bengio, Yoshua, Patrice Simard, and Paolo Frasconi. "Learning long-term dependencies with gradient descent is difficult." IEEE transactions on neural networks 5.2 (1994): 157-166.
https://en.wikipedia.org/wiki/Power_iteration
http://www.cs.cornell.edu/~bindel/class/cs6210-f09/lec26.pdf

Vanishing/Exploding Gradient Problem

- Consider a ***linear*** recurrent net with zero inputs
- $$h_t = W_{hh} h_{t-1} = h_0 W_{hh}^t$$
-
- Singular value > 1 ⇒ Explodes
- Singular value < 1 ⇒ Vanishes

Bengio, Yoshua, Patrice Simard, and Paolo Frasconi. "Learning long-term dependencies with gradient descent is difficult." IEEE transactions on neural networks 5.2 (1994): 157-166.
https://en.wikipedia.org/wiki/Power_iteration
http://www.cs.cornell.edu/~bindel/class/cs6210-f09/lec26.pdf

Vanishing/Exploding Gradient Problem

- Consider a ***linear*** recurrent net with zero inputs
- $$h_t = W_{hh} h_{t-1} = h_0 W_{hh}^t$$
-
- “It is **sufficient** for the largest eigenvalue λ_1 of the recurrent weight matrix to be smaller than 1 for long term components to vanish (as t → ∞) and **necessary** for it to be larger than 1 for gradients to explode.”

Details are here

Bengio, Yoshua, Patrice Simard, and Paolo Frasconi. "Learning long-term dependencies with gradient descent is difficult." IEEE transactions on neural networks 5.2 (1994): 157-166.
https://en.wikipedia.org/wiki/Power_iteration
http://www.cs.cornell.edu/~bindel/class/cs6210-f09/lec26.pdf

Long short-term memory (LSTM) come to the rescue

Vanilla RNN

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

LSTM

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

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

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

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

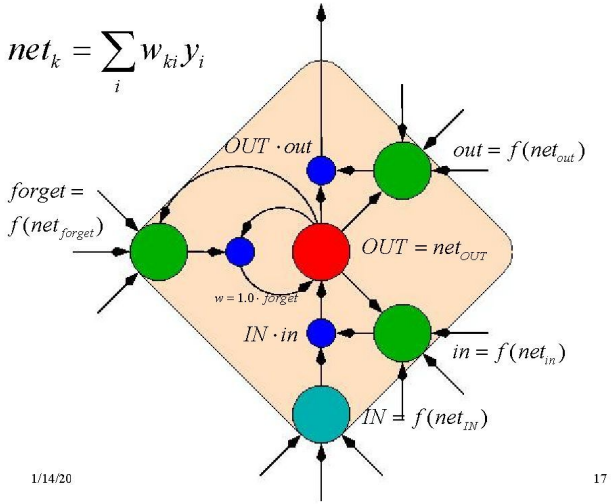
Why LSTM works

- i: input gate
- f: forget gate
- o: output gate
- g: temp variable
- c: memory cell
-
- Key observation:
  - If **f** == 1, then
    - $c_t = c_{t-1} + i \odot g$
  - Looks like a ResNet!
    - $x_{t+1} = x_t + F(x_t)$

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

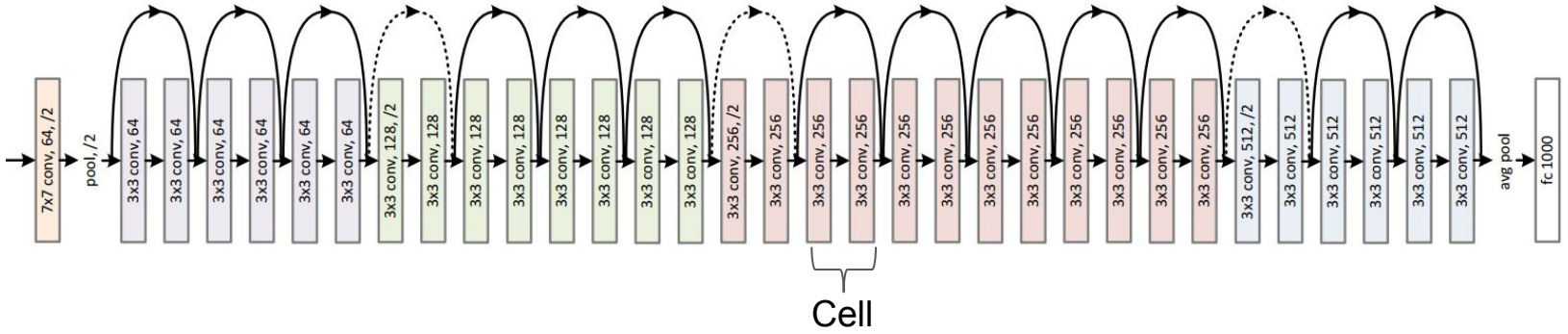
$$c_t = f \odot c_{t-1} + i \odot g$$

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



http://people.idsia.ch/~juergen/lstm/sld017.htm

LSTM vs Weight Sharing ResNet



- Difference
  - Never forgets
  - No intermediate inputs

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

vs

$$x_{t+1} = x_t + F(x_t)$$

GRU

- Similar to LSTM
- Let information flow without a separate memory cell
-
- Consider $z_t = 0$

$$\begin{pmatrix} z_t \\ r_t \end{pmatrix} = \sigma\left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

$$\tilde{h}_t = \tanh(W x_t + U(r_t \odot h_{t-1}))$$

$$h_t = (1 - z_t) h_{t-1} + z_t \tilde{h}_t$$

Chung, Junyoung, et al. "Empirical evaluation of gated recurrent neural networks on sequence modeling." arXiv preprint arXiv:1412.3555 (2014).

Search for Better RNN Architecture

1. **Initialize** a pool with {LSTM, GRU}
2. **Evaluate** new architecture with 20 hyperparameter settings
3. **Select** one at random from the pool
4. **Mutate** the selected architecture **Key step**
5. **Evaluate** new architecture with 20 hyperparameter settings
6. **Maintain** a list of 100 best architectures
7. Goto 3

MUT1:

$$\begin{aligned} z &= \text{sigm}(W_{\text{xz}}x_t + b_{\text{z}}) \\ r &= \text{sigm}(W_{\text{xr}}x_t + W_{\text{hr}}h_t + b_{\text{r}}) \\ h_{t+1} &= \tanh(W_{\text{hh}}(r \odot h_t) + \tanh(x_t) + b_{\text{h}}) \odot z \\ &+ h_t \odot (1 - z) \end{aligned}$$

MUT2:

$$\begin{aligned} z &= \text{sigm}(W_{\text{xz}}x_t + W_{\text{hz}}h_t + b_{\text{z}}) \\ r &= \text{sigm}(x_t + W_{\text{hr}}h_t + b_{\text{r}}) \\ h_{t+1} &= \tanh(W_{\text{hh}}(r \odot h_t) + W_{xh}x_t + b_{\text{h}}) \odot z \\ &+ h_t \odot (1 - z) \end{aligned}$$

MUT3:

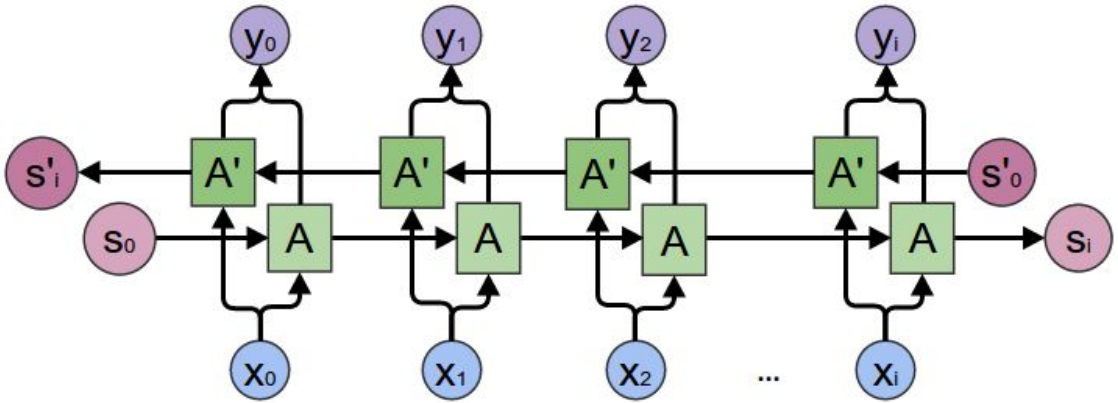
$$\begin{aligned} z &= \text{sigm}(W_{\text{xz}}x_t + W_{\text{hz}}\tanh(h_t) + b_{\text{z}}) \\ r &= \text{sigm}(W_{\text{xr}}x_t + W_{\text{hr}}h_t + b_{\text{r}}) \\ h_{t+1} &= \tanh(W_{\text{hh}}(r \odot h_t) + W_{xh}x_t + b_{\text{h}}) \odot z \\ &+ h_t \odot (1 - z) \end{aligned}$$

Jozefowicz, Rafal, Wojciech Zaremba, and Ilya Sutskever. "An empirical exploration of recurrent network architectures." Proceedings of the 32nd International Conference on Machine Learning (ICML-15). 2015.

Simple RNN Extensions

Bidirectional RNN (BDRNN)

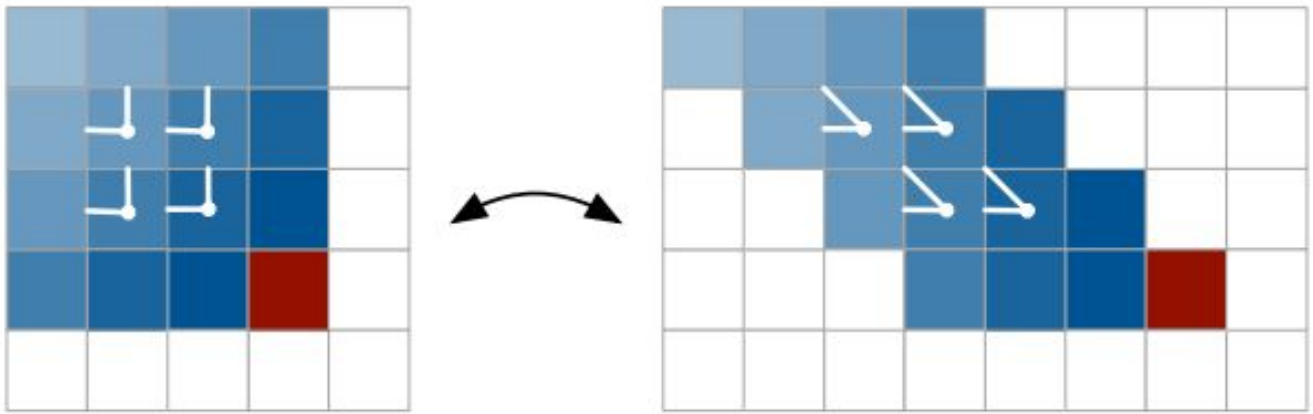
- RNN can go either way
- “Peak into the future”
- Truncated version used in speech recognition



https://github.com/huseinzol05/Generate-Music-Bidirectional-RNN

2D-RNN: Pixel-RNN

- Pixel-RNN
- Each pixel depends on its top and left neighbor



Oord, Aaron van den, Nal Kalchbrenner, and Koray Kavukcuoglu. "Pixel recurrent neural networks." arXiv preprint arXiv:1601.06759 (2016).

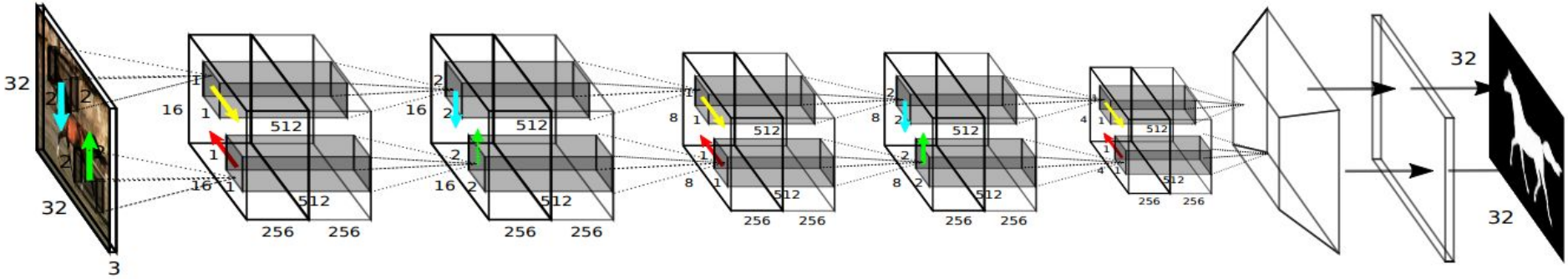
Pixel-RNN

Figure 1. Image completions sampled from a PixelRNN.

Oord, Aaron van den, Nal Kalchbrenner, and Koray Kavukcuoglu. "Pixel recurrent neural networks." arXiv preprint arXiv:1601.06759 (2016).

Pixel-RNN Application

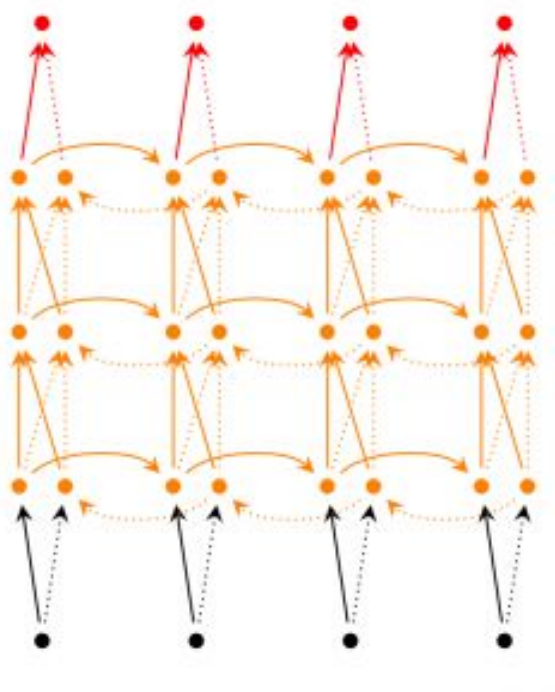
- Segmentation



Visin, Francesco, et al. "Reseg: A recurrent neural network-based model for semantic segmentation." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. 2016.

Deep RNN

- Stack more of them
  - Pros
    - More representational power
  - Cons
    - Harder to train
      - ⇒ Need residual connections along depth



RNN Basics Summary

- The evolution of RNN from Feedforward NN
- Recurrence as unrolled computation graph
- Vanishing/Exploding gradient problem
  - LSTM and variants
  - $\text{recurrence} \in \text{weight-sharing}$ and the relation to ResNet
- Extensions
  - BDRNN
  - 2DRNN
  - Deep-RNN

RNN with Attention

What is Attention?

- Differentiate entities by its importance
  - spatial attention is related to location
  - temporal attention is related to causality

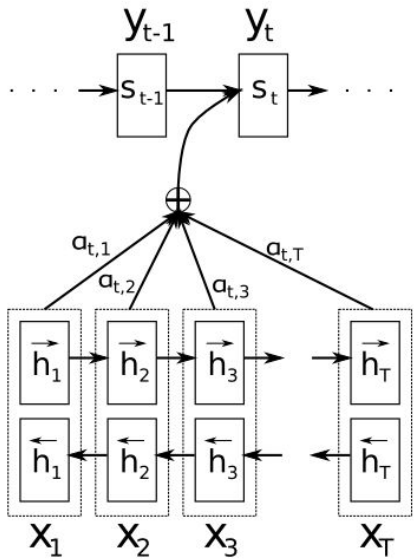
$$\sum \alpha_i x_i$$

$$0 \leq \alpha_i \leq 1$$

https://distill.pub/2016/augmented-rnns

Attention over Input Sequence

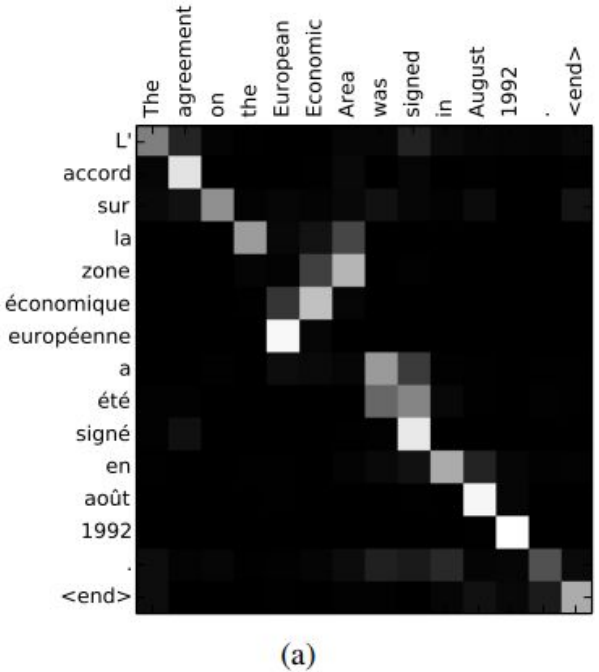
- Neural Machine Translation (NMT)



Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." arXiv preprint arXiv:1409.0473 (2014).

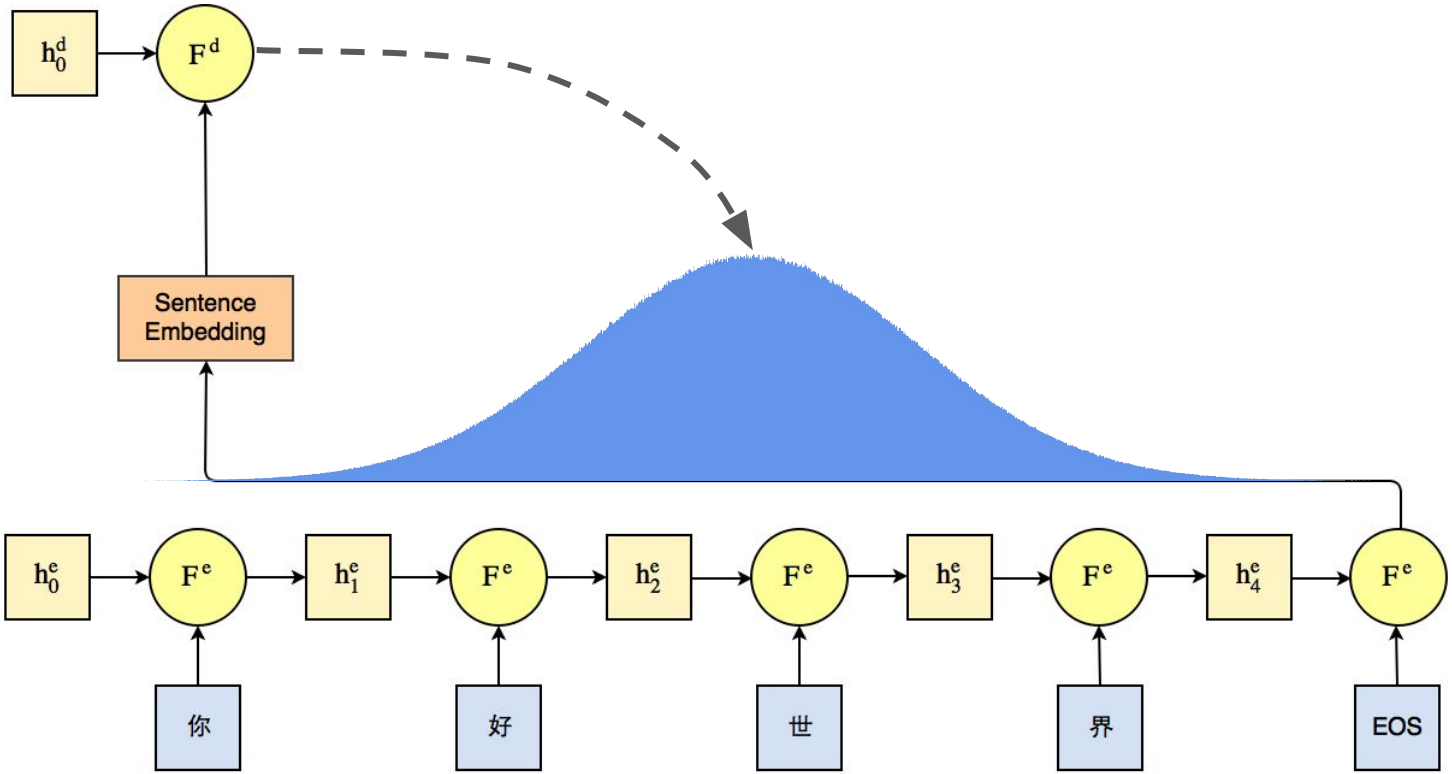
Neural Machine Translation (NMT)

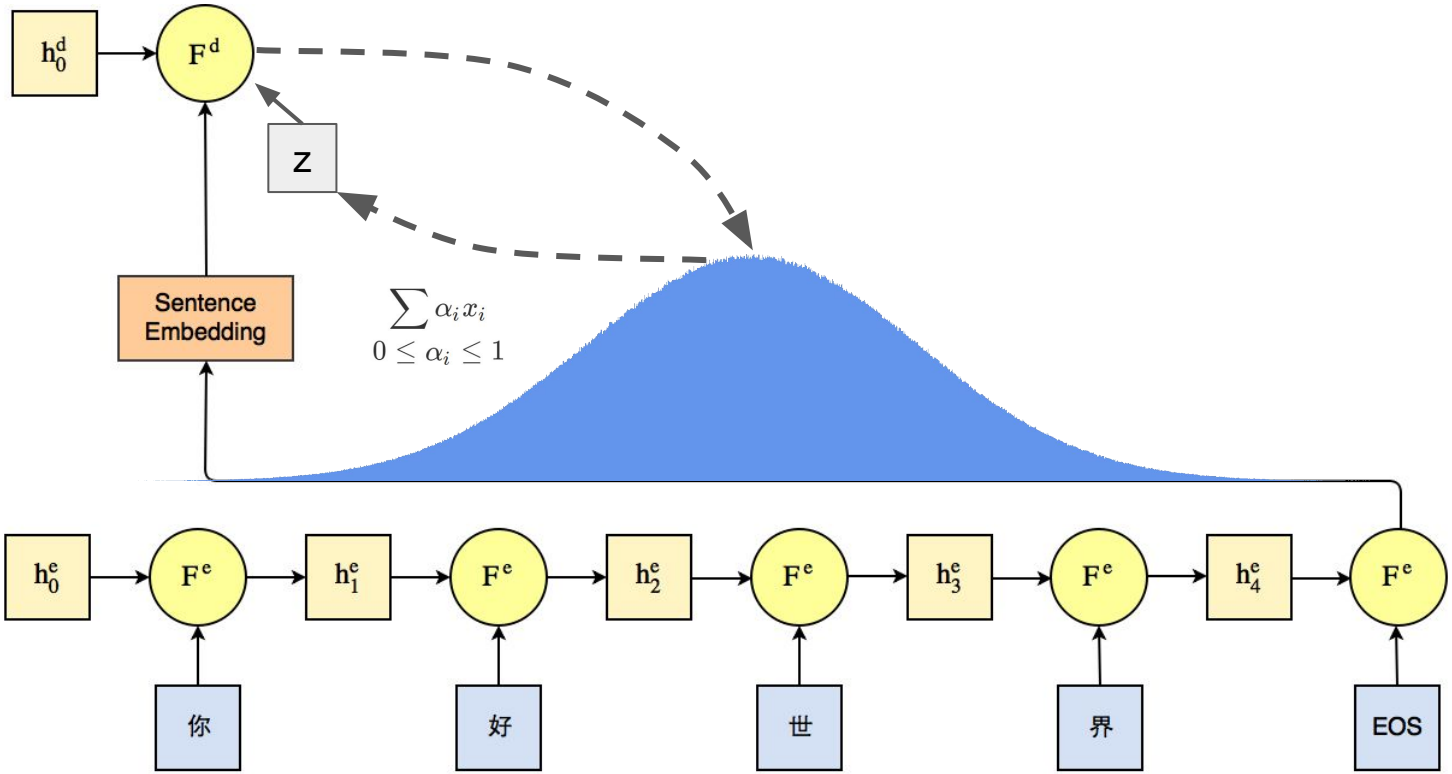
- Attention over input sequence
- There’re words in two languages that share the same meaning.
- Attention ⇒ Alignment
  - Differentiable, allowing end-to-end training

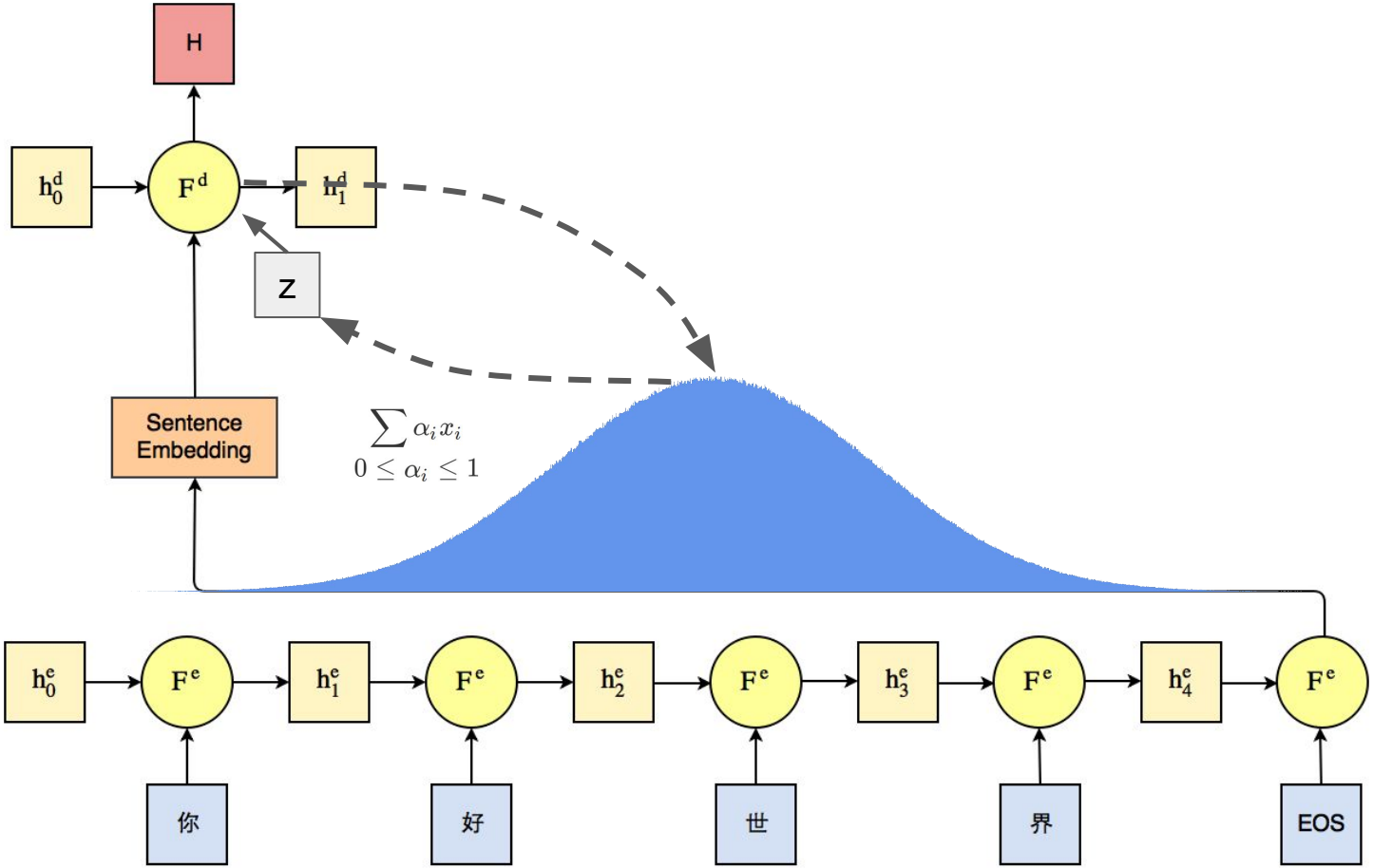


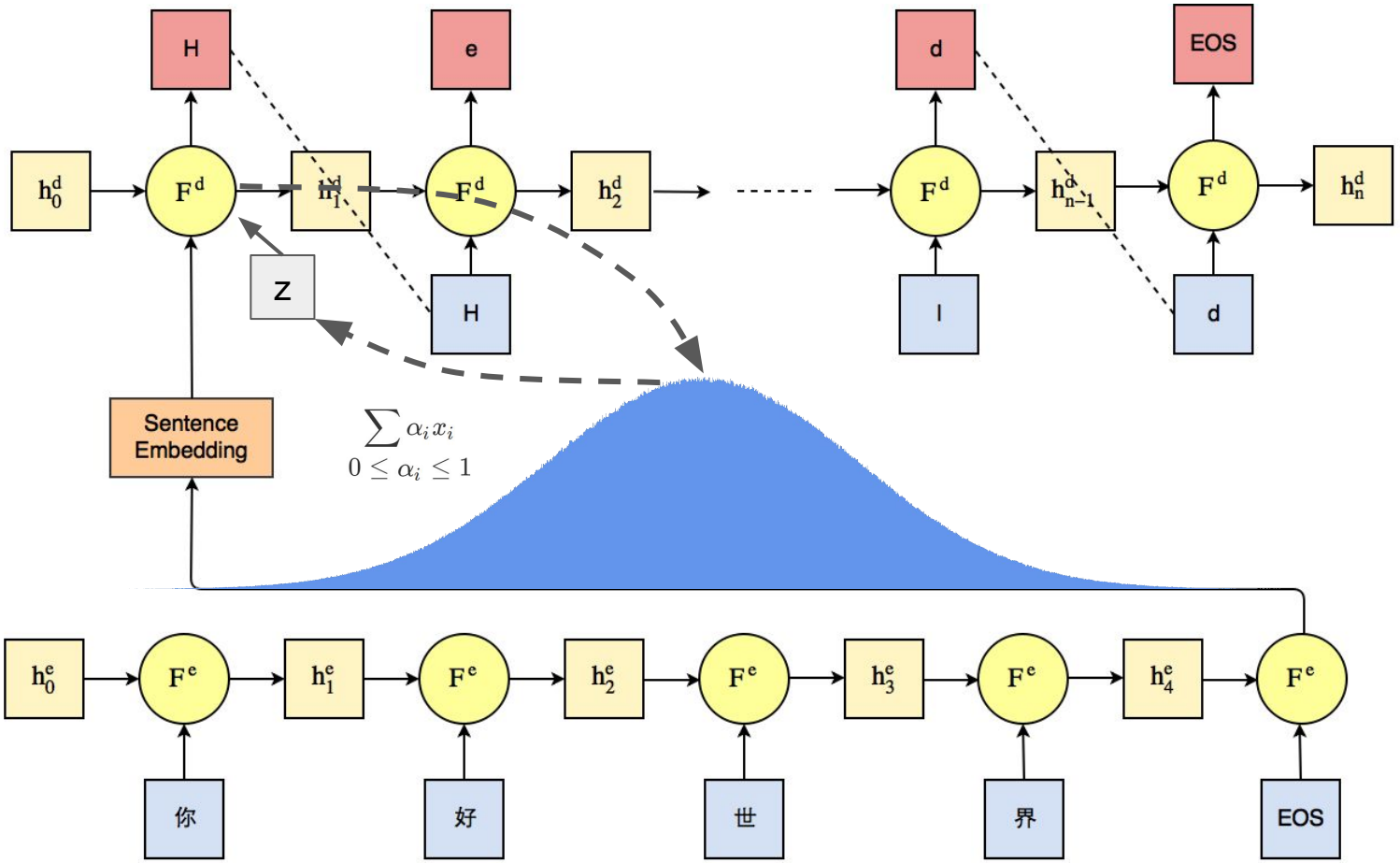
(a)

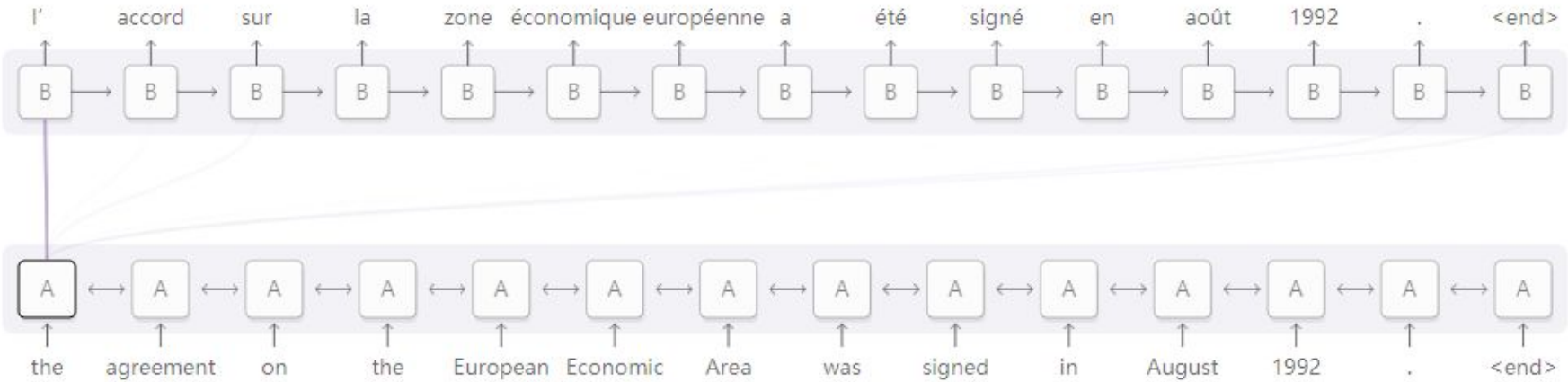
Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." arXiv preprint arXiv:1409.0473 (2014).


h_0^d
F^d
Sentence Embedding
h_0^e
F^e
h_1^e
F^e
h_2^e
F^e
h_3^e
F^e
h_4^e
F^e
你
好
世
界
EOS

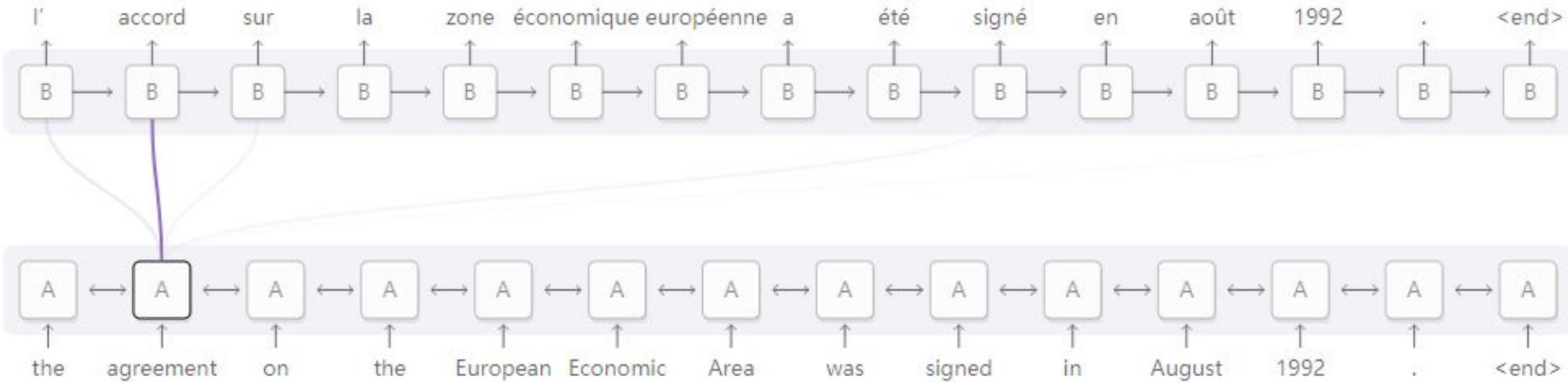

h_0^d
F^d
z
Sentence Embedding
$\sum \alpha_i x_i$
$0 \leq \alpha_i \leq 1$
h_0^e
F^e
h_1^e
h_2^e
h_3^e
h_4^e
你
好
世
界
EOS


H
h_0^d
F^d
h_1^d
Z
Sentence Embedding
$\sum \alpha_i x_i$
$0 \leq \alpha_i \leq 1$
h_0^e
F^e
h_1^e
h_2^e
h_3^e
h_4^e
你
好
世
界
EOS

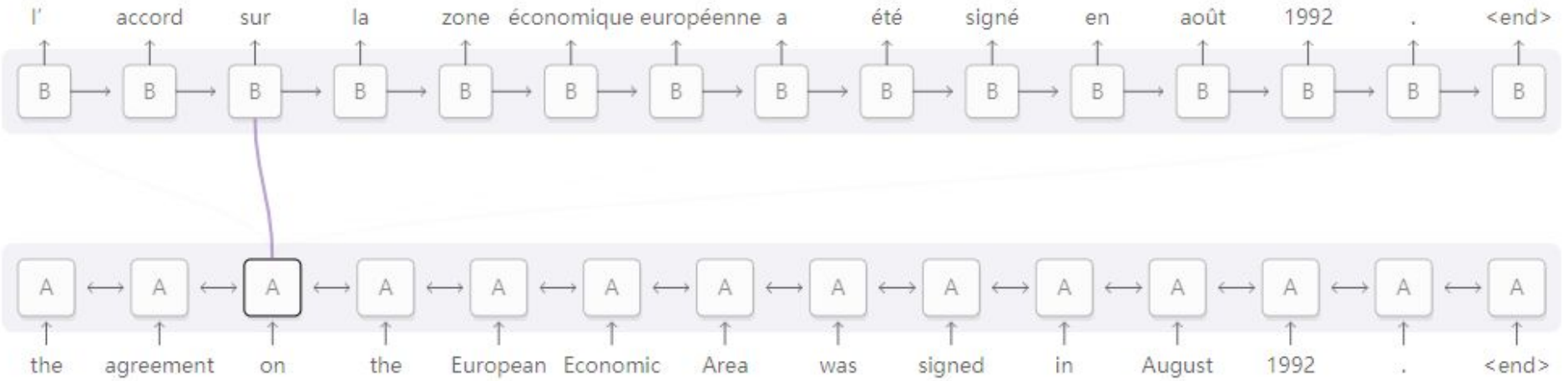

H
e
d
EOS
h_0^d
F^d
h_1^d
F^d
h_2^d
F^d
h_{n-1}^d
F^d
h_n^d
Z
H
I
d
Sentence Embedding
$\sum \alpha_i x_i$
$0 \leq \alpha_i \leq 1$
h_0^e
F^e
h_1^e
F^e
h_2^e
F^e
h_3^e
F^e
h_4^e
F^e
你
好
世
界
EOS



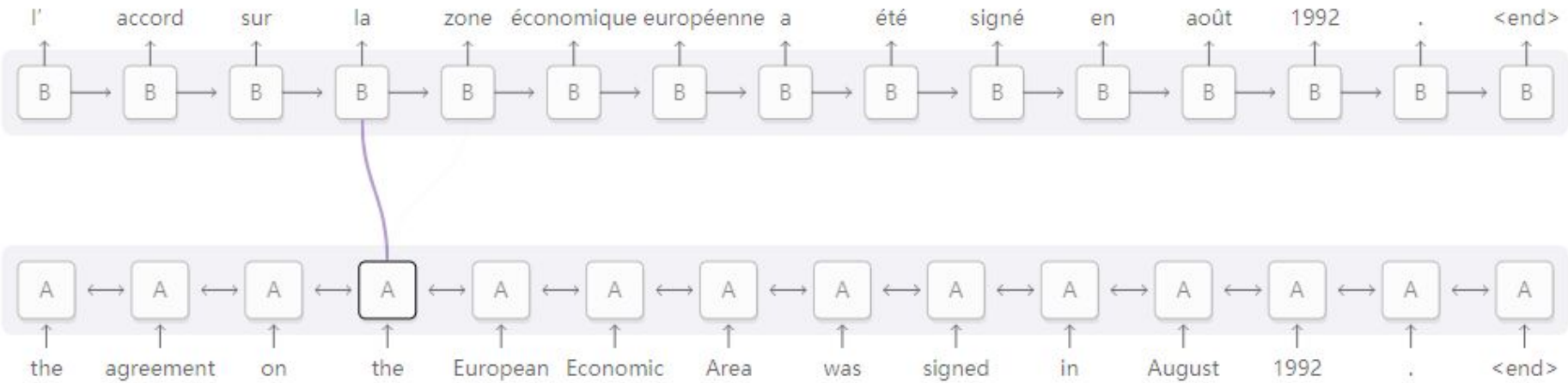
https://distill.pub/2016/augmented-rnns



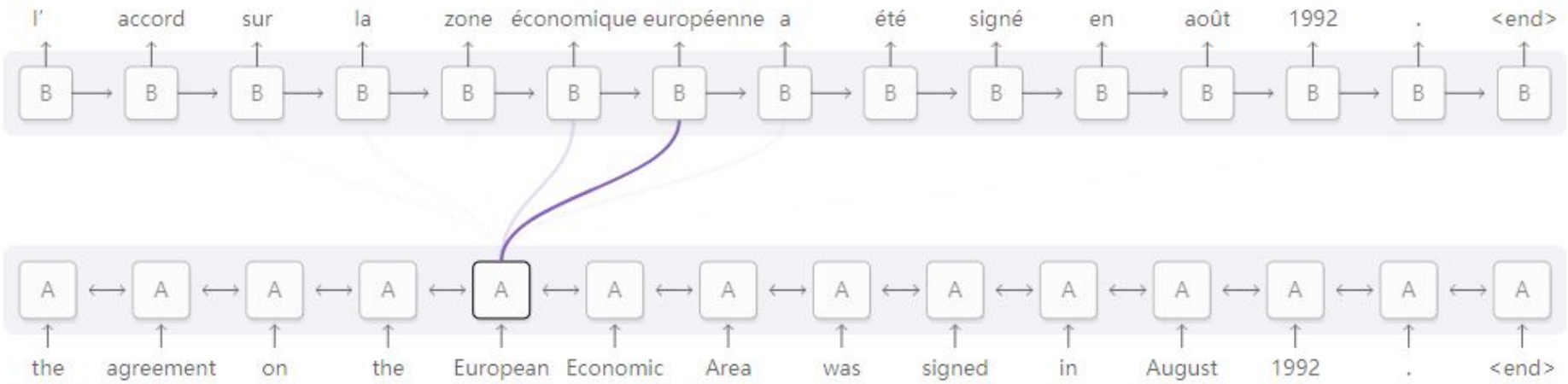
https://distill.pub/2016/augmented-rnns



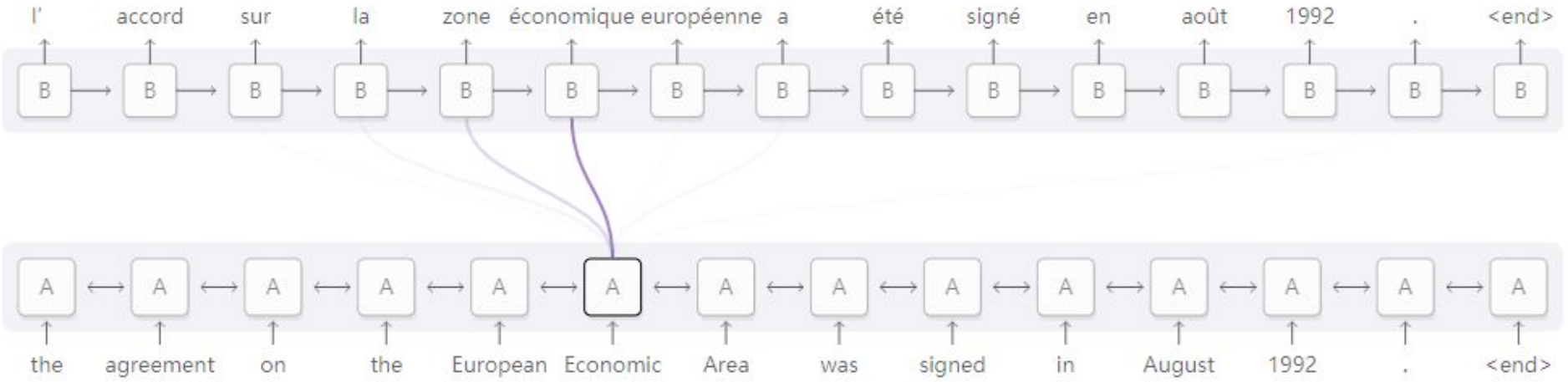
https://distill.pub/2016/augmented-rnns



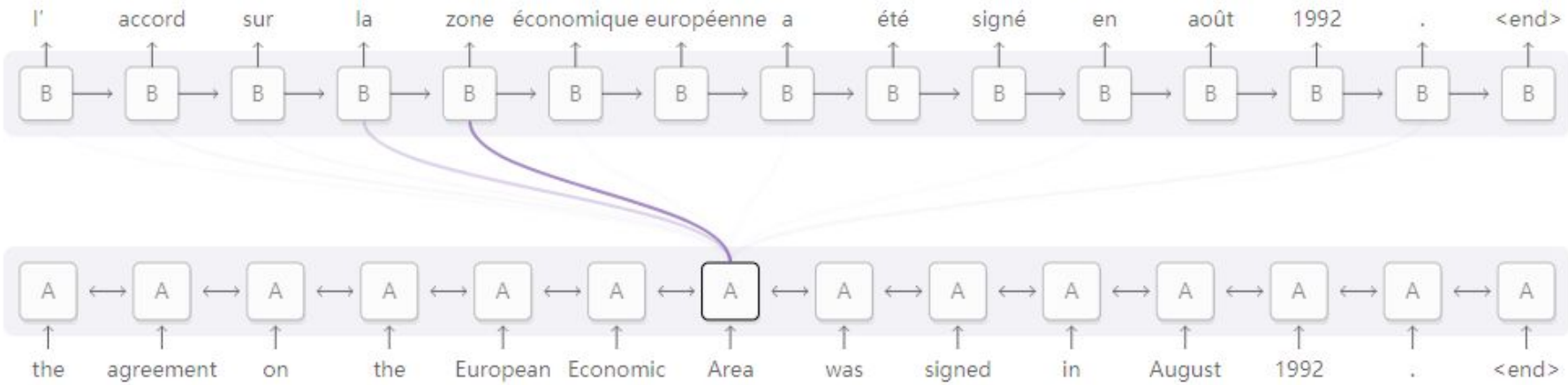
https://distill.pub/2016/augmented-rnns



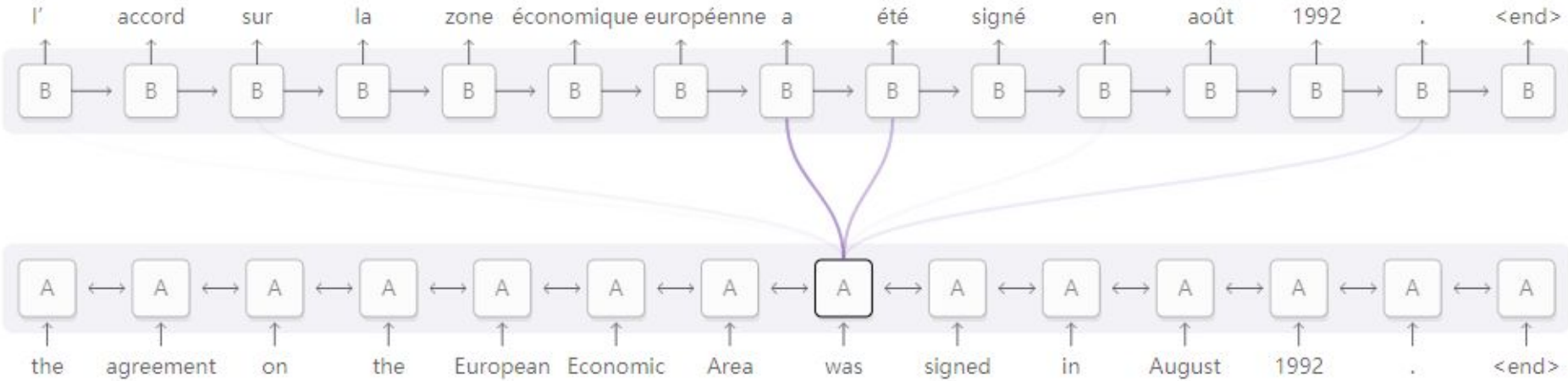
https://distill.pub/2016/augmented-rnns



https://distill.pub/2016/augmented-rnns

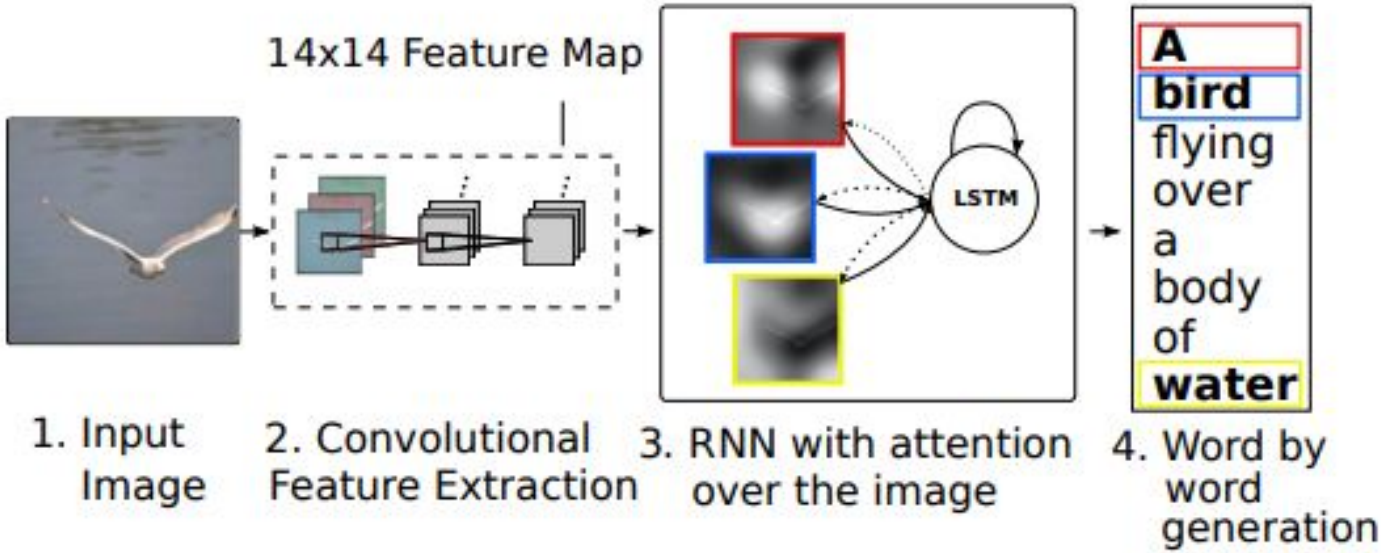


https://distill.pub/2016/augmented-rnns



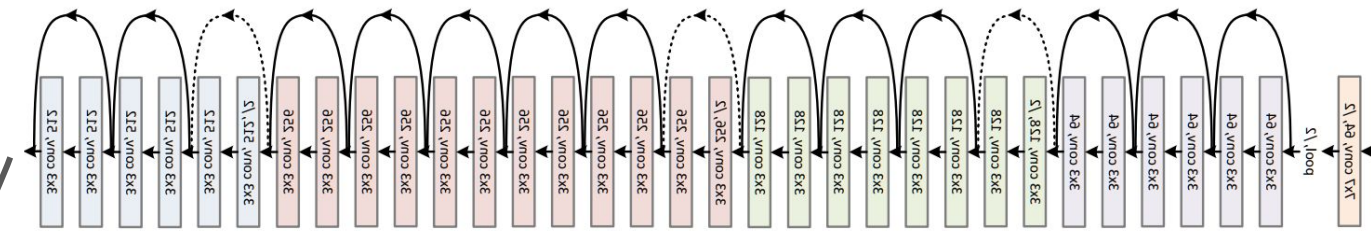
https://distill.pub/2016/augmented-rnns

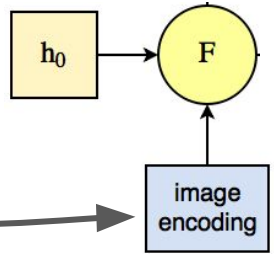
Image Attention: Image Captioning

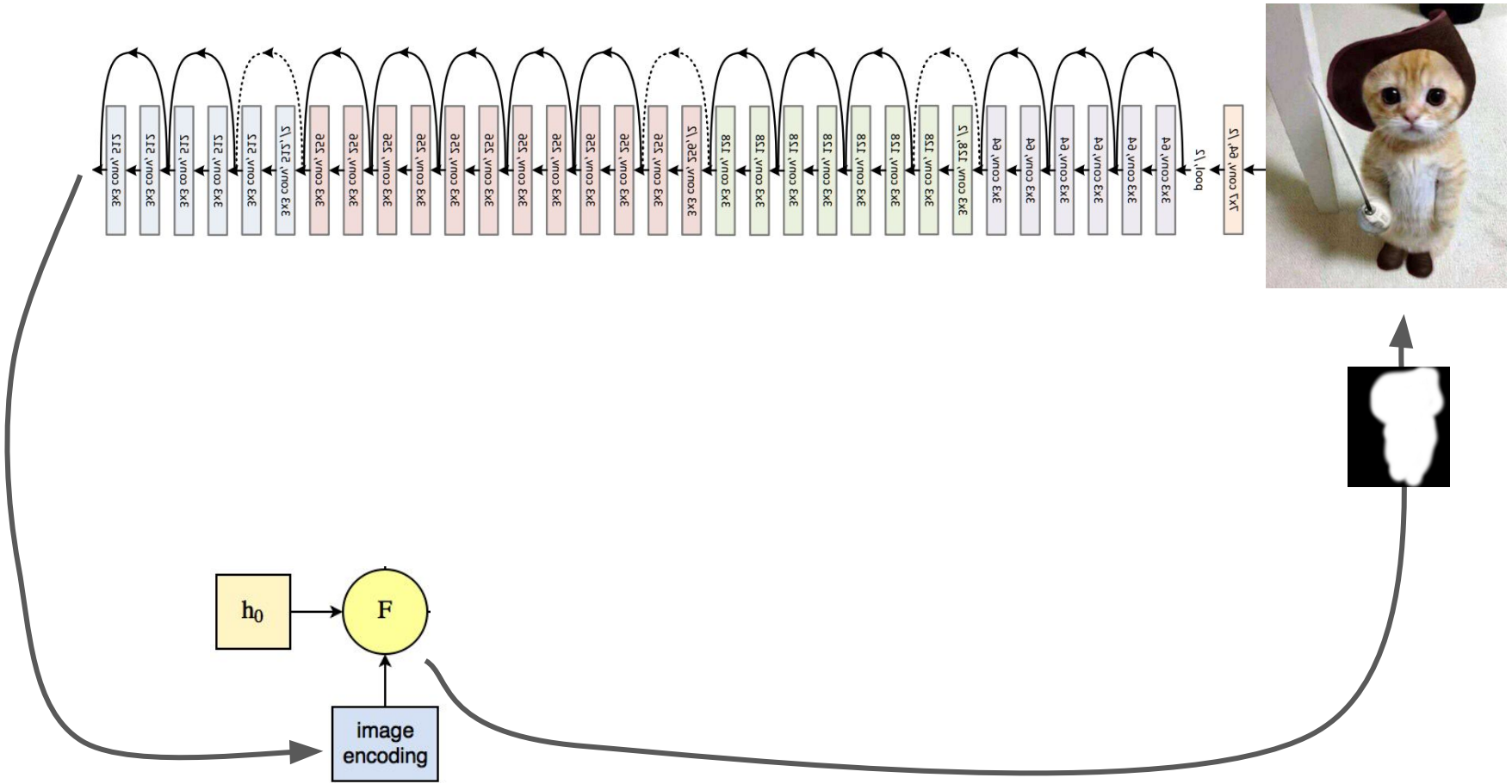


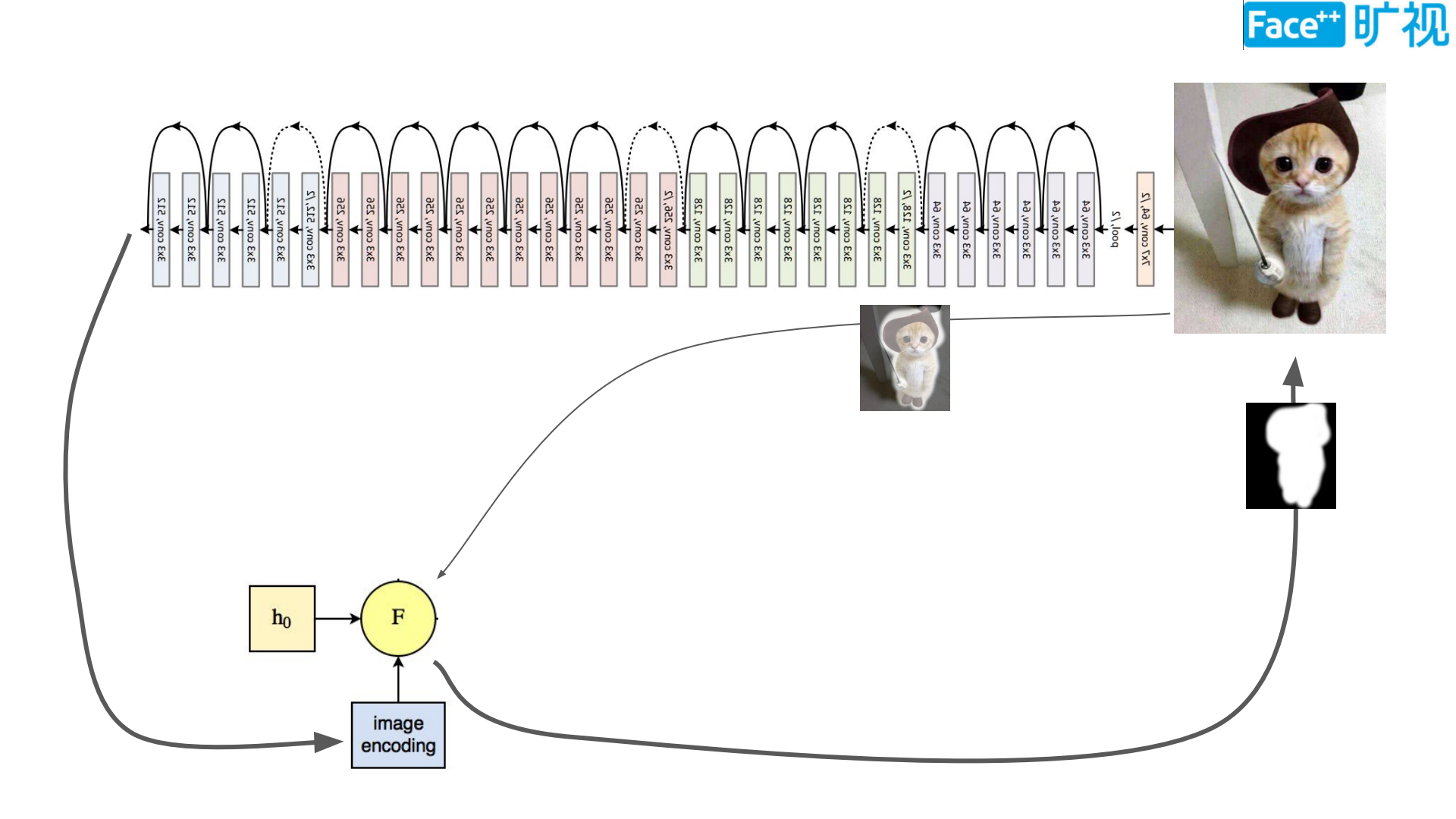
Xu, Kelvin, et al. "Show, attend and tell: Neural image caption generation with visual attention." International Conference on Machine Learning. 2015.

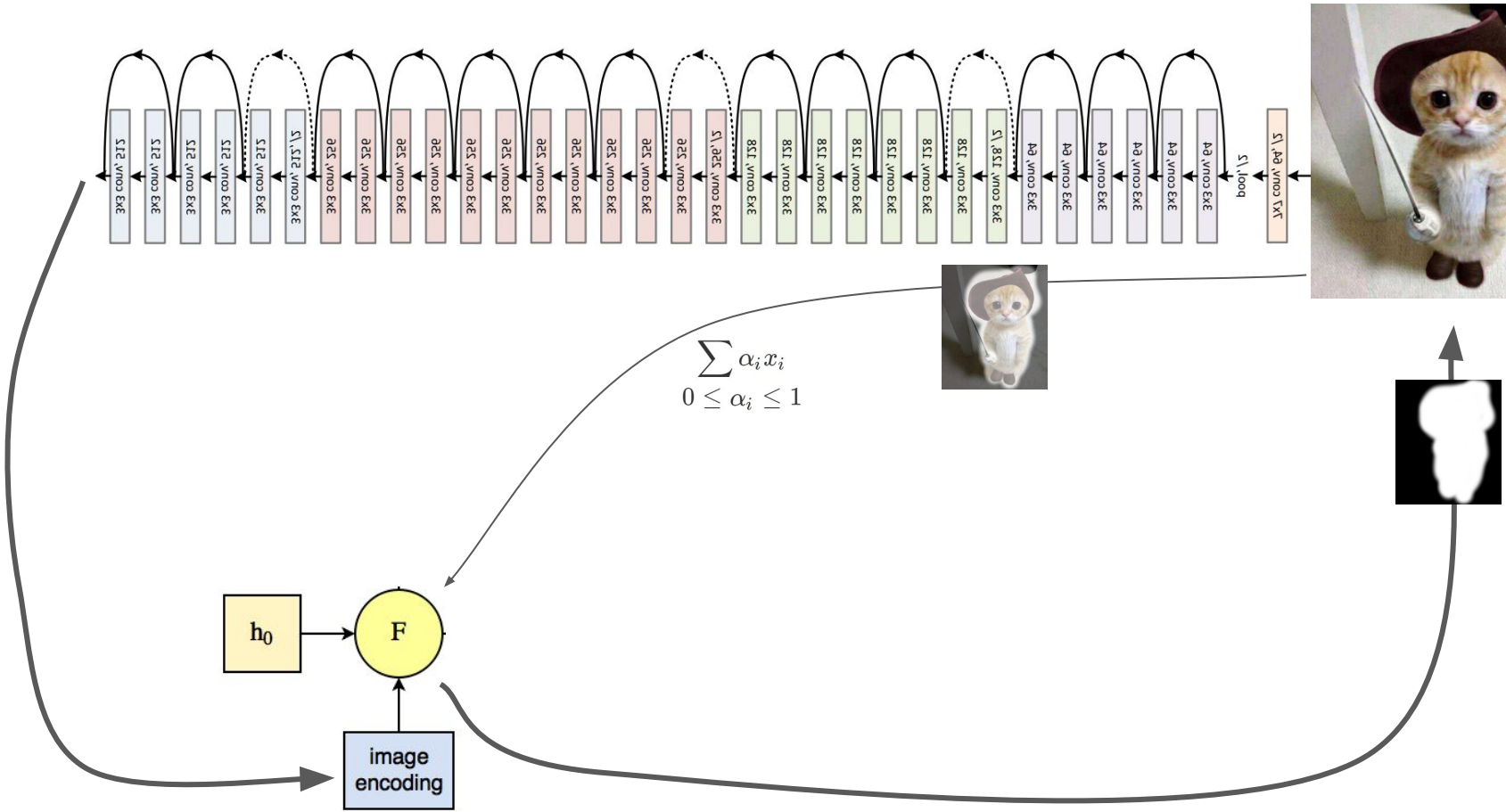
3x3 conv, 64
3x3 conv, 64
3x3 conv, 128
3x3 conv, 128
3x3 conv, 256
3x3 conv, 256
3x3 conv, 512
3x3 conv, 512
3x3 conv, 1024
3x3 conv, 1024
3x3 conv, 2048
3x3 conv, 2048
bool

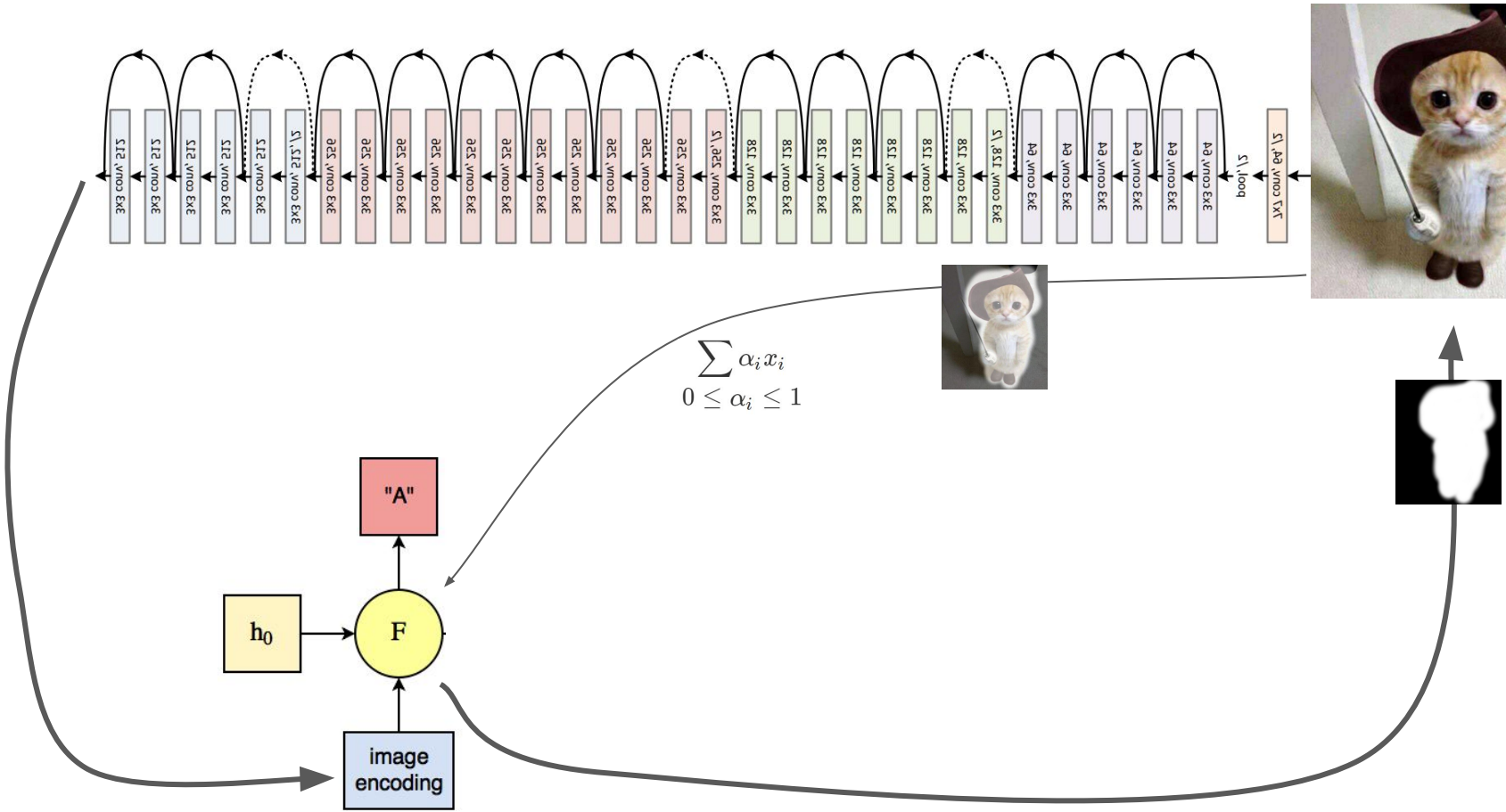


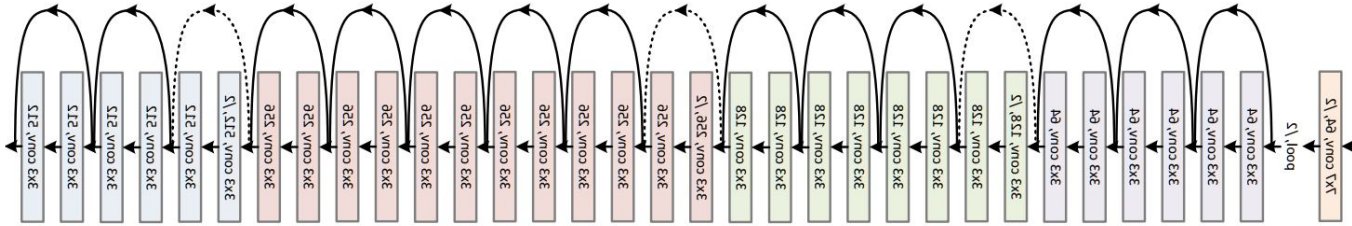

h0
F
image
encoding


h0
F
image encoding


h0
F
image
encoding


$\sum \alpha_i x_i$
$0 \leq \alpha_i \leq 1$
h_0
F
image encoding


3x3 conv, 512
3x3 conv, 256
3x3 conv, 128
3x3 conv, 64
3x3 conv, 3
bool, 1
$\sum \alpha_i x_i$
$0 \leq \alpha_i \leq 1$
"A"
h_0
F
image encoding



$$\sum \alpha_i x_i$$
$$0 \le \alpha_i \le 1$$

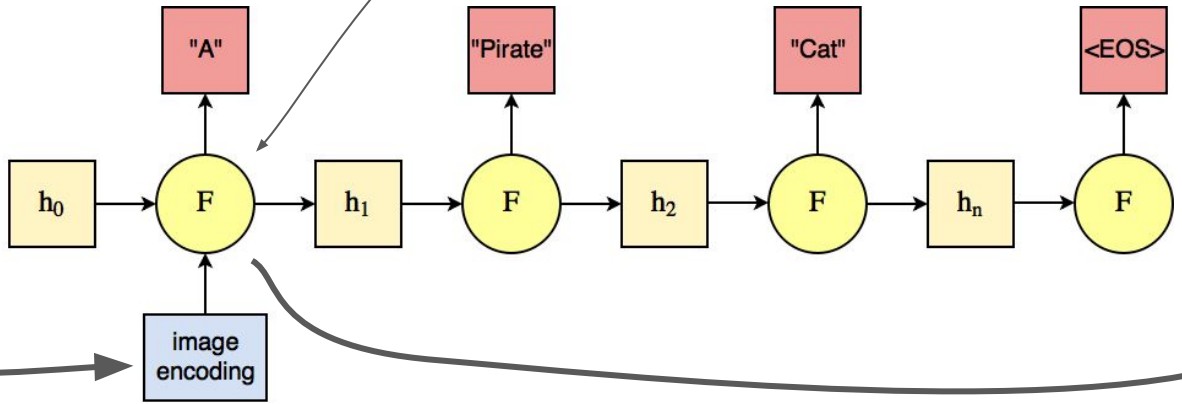


Image Attention: Image Captioning

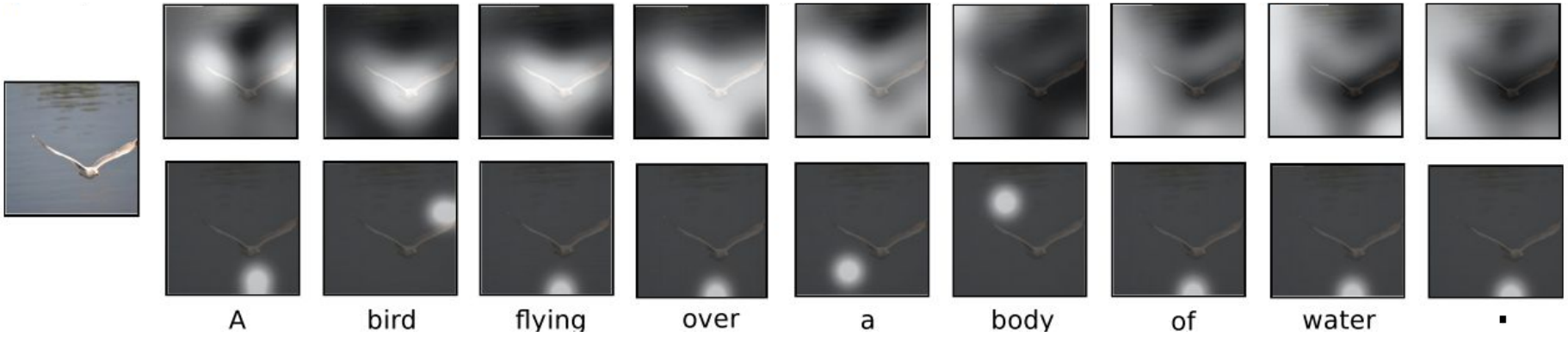


Image Attention: Image Captioning

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

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

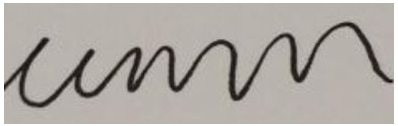
A little girl sitting on a bed with a teddy bear.

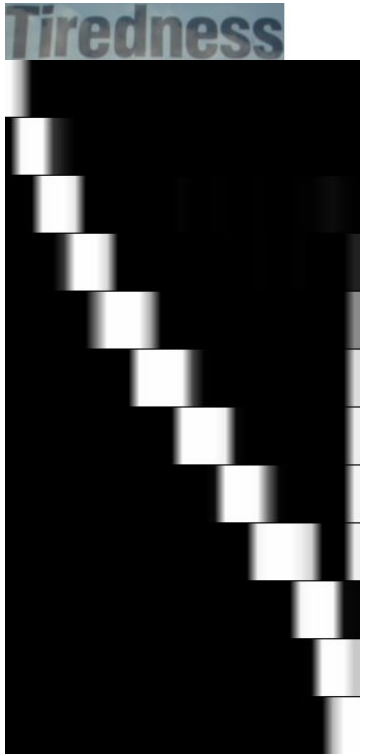
A group of people sitting on a boat in the water.

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

Text Recognition

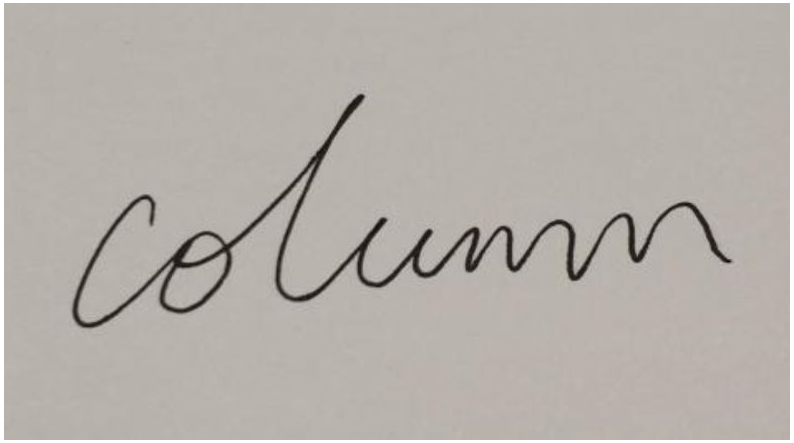
- Implicit language model

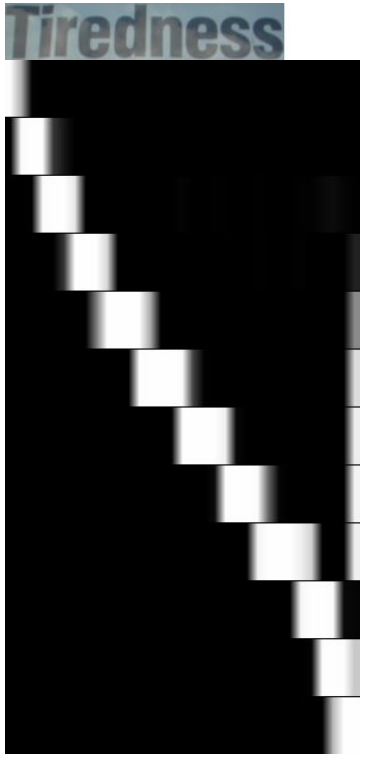




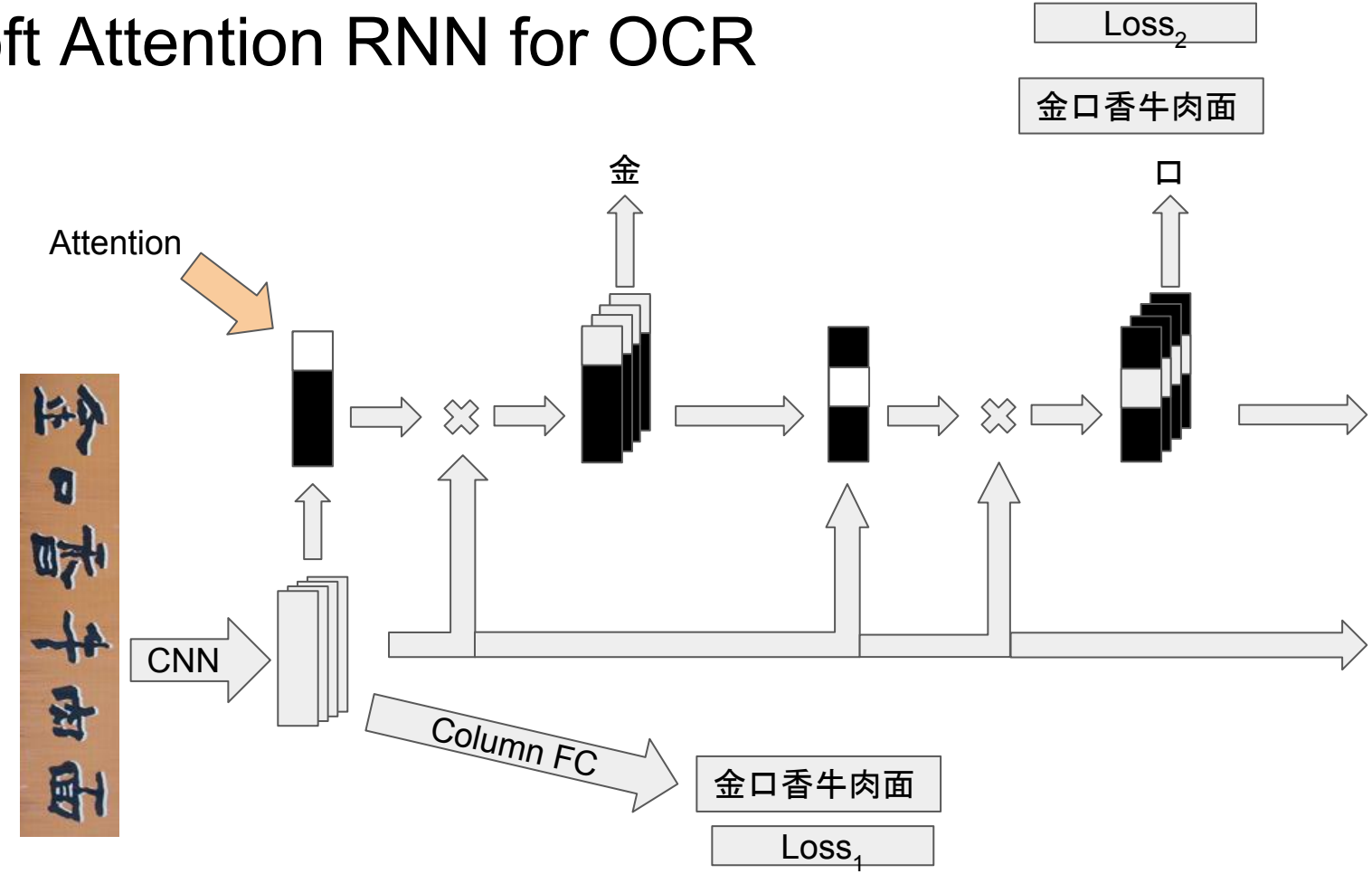
Text Recognition

- Implicit language model





Soft Attention RNN for OCR



RNN with External Memory

Copy a sequence

Input

Output

Copy a sequence

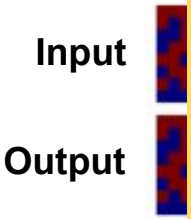
Input

Output

Solution in Python

```python
# input data
input_list = [0, 2, 4, 4, 1, 5, 2]

# initialize memory
model_memory = [0] * len(input_list)

# store everything read
loc_write = 0
for value in input_list:
    model_memory[loc_write] = value
    loc_write += 1

# write everything stored
loc_read = 0
while loc_read < loc_write:
    print(model_memory[loc_read])
    loc_read += 1

```

Copy a sequence



Can neural network learn this program purely from data?

```
1  # input data
2  input_list = [0, 2, 4, 4, 1, 5, 2]
3
                                    mory
                                   [0] * len(input_list)

                                   ing read

                                   ut_list:
                                   y[loc_write] = value
                                   = 1

                                   ing stored

15 while loc_read < loc_write:
16     print(model_memory[loc_read])
17     loc_read += 1
18
```

Traditional Machine Learning

- √ Elementary Operations
- √* Logic flow control
  - Decision tree
- × External Memory
  - As opposed to internal memory (hidden states)

Graves, Alex, Greg Wayne, and Ivo Danihelka. "Neural turing machines." arXiv preprint arXiv:1410.5401 (2014).

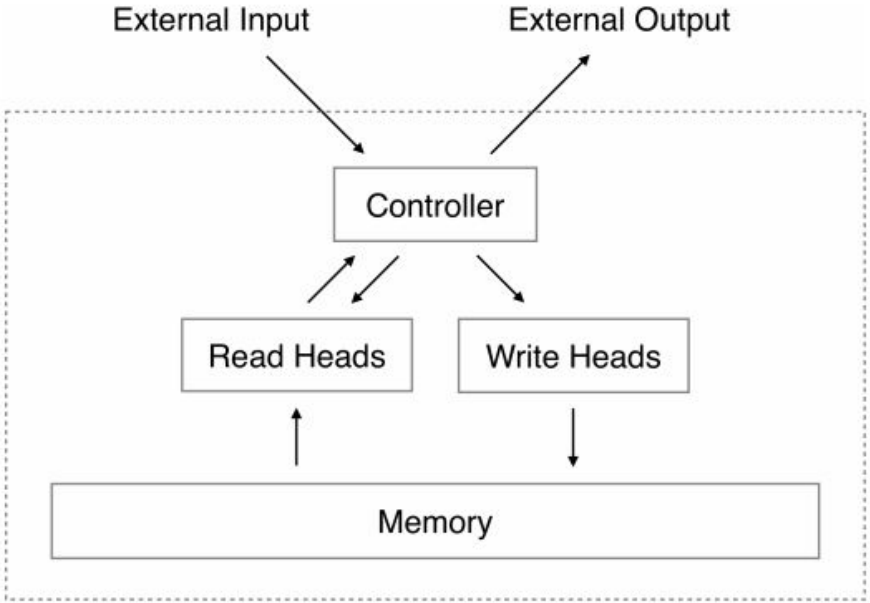
Traditional Machine Learning

- √ Elementary Operations
- √* Logic flow control
- × External Memory

Graves, Alex, Greg Wayne, and Ivo Danihelka. "Neural turing machines." arXiv preprint arXiv:1410.5401 (2014).

Neural Turing Machines (NTM)

- NTM is a neural networks with a working memory
- It reads and write multiple times at each step
- Fully differentiable and can be trained end-to-end

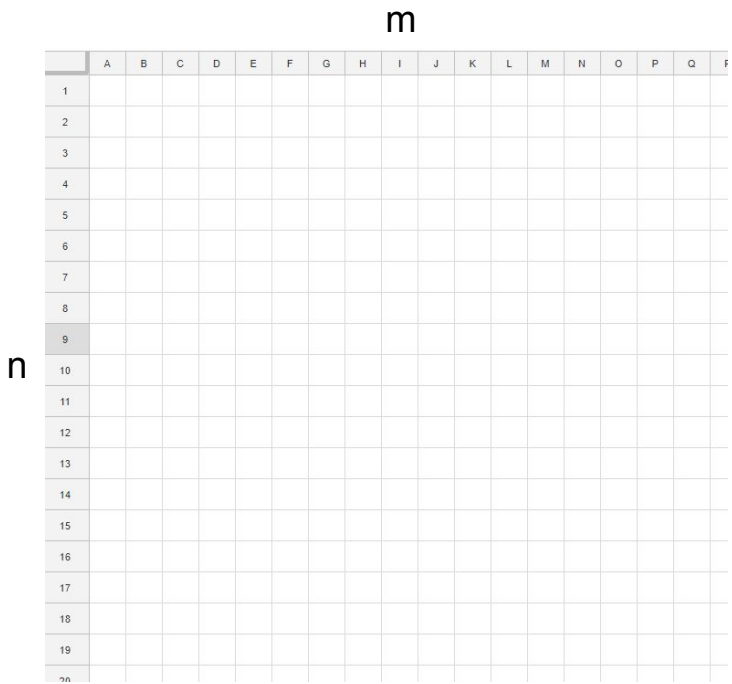


An NTM “Cell”

Graves, Alex, Greg Wayne, and Ivo Danihelka. "Neural turing machines." arXiv preprint arXiv:1410.5401 (2014).

Neural Turing Machines (NTM)

- Memory
  - $\text{An } n \times m \text{ matrix } \mathbf{M}_t \text{ at time } t$



http://llcao.net/cu-deeplearning15/presentation/NeuralTuringMachines.pdf

Neural Turing Machines (NTM)

- Read

$$\sum_i w_t(i) = 1, \;\; 0 \leq w_t(i) \leq 1, \forall i$$

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

- Hard indexing ⇒ Soft Indexing
  - A distribution of index
  - “Attention”

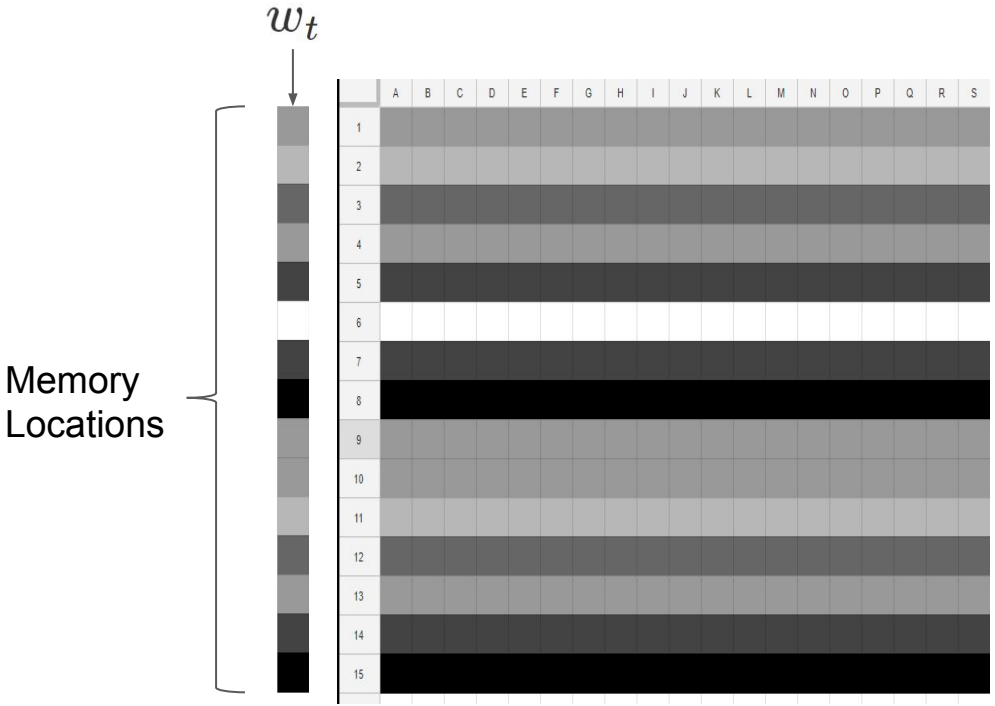
Neural Turing Machines (NTM)

- Read

$$\sum_i w_t(i) = 1, \;\; 0 \leq w_t(i) \leq 1, \forall i$$

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

- Hard indexing ⇒ Soft Indexing
  - A distribution of index
  - “Attention”



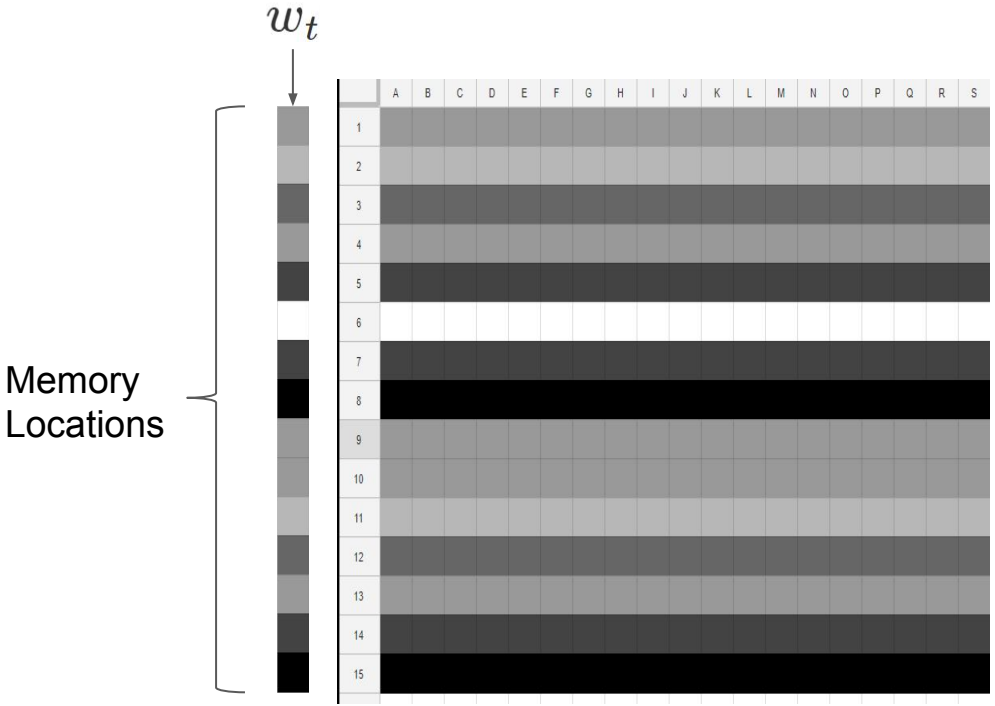
Neural Turing Machines (NTM)

- Read

$$\sum_i w_t(i) = 1, \quad 0 \le w_t(i) \le 1, \forall i$$

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

- Hard indexing ⇒ Soft Indexing
  - A distribution of index
  - “Attention”



Neural Turing Machines (NTM)

- Write
  - Write = erase + add

$$\tilde{\mathbf{M}}_t(i) \longleftarrow \mathbf{M}_{t-1}(i)\left[\mathbf{1} - w_t(i)\mathbf{e}_t\right], \quad \longleftarrow \text{erase}$$

$$\mathbf{M}_t(i) \longleftarrow \tilde{\mathbf{M}}_t(i) + w_t(i)\,\mathbf{a}_t. \quad \longleftarrow \text{add}$$

Neural Turing Machines (NTM)

- Write
  - Write = erase + add

$$\tilde{\mathbf{M}}_t(i) \longleftarrow \mathbf{M}_{t-1}(i)\left[\mathbf{1} - w_t(i)\mathbf{e}_t\right], \quad \longleftarrow \text{erase}$$

$$\mathbf{M}_t(i) \longleftarrow \tilde{\mathbf{M}}_t(i) + w_t(i)\mathbf{a}_t. \quad \longleftarrow \text{add}$$

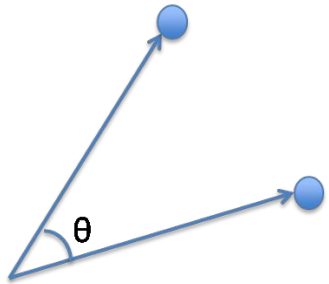
Neural Turing Machines (NTM)

- Addressing

Neural Turing Machines (NTM)

- Addressing
- 1. Focusing by Content

$$w_t^c(i) \longleftarrow \frac{\exp\Big(\beta_t K\big[\mathbf{k}_t, \mathbf{M}_t(i)\big]\Big)}{\sum_j \exp\Big(\beta_t K\big[\mathbf{k}_t, \mathbf{M}_t(j)\big]\Big)}.$$



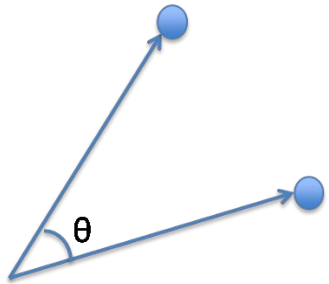
- Cosine Similarity

$$K\big[\mathbf{u}, \mathbf{v}\big] = \frac{\mathbf{u} \cdot \mathbf{v}}{||\mathbf{u}|| \cdot ||\mathbf{v}||}.$$

Neural Turing Machines (NTM)

- Addressing
- 1. Focusing by Content

$$w_t^c(i) \leftarrow \frac{\exp\left(\beta_t K\left[\mathbf{k}_t, \mathbf{M}_t(i)\right]\right)}{\sum_j \exp\left(\beta_t K\left[\mathbf{k}_t, \mathbf{M}_t(j)\right]\right)}.$$



- Cosine Similarity

$$K\left[\mathbf{u}, \mathbf{v}\right] = \frac{\mathbf{u} \cdot \mathbf{v}}{||\mathbf{u}|| \cdot ||\mathbf{v}||}.$$

Neural Turing Machines (NTM)

- 1. Focusing by Content
- 2. Interpolate with previous step

$$\mathbf{w}_t^g \longleftarrow g_t \mathbf{w}_t^c + (1 - g_t)\mathbf{w}_{t-1}.$$

Neural Turing Machines (NTM)

- 1. Focusing by Content
- 2. Interpolate with previous step

$$\mathbf{w}_t^g \longleftarrow g_t \mathbf{w}_t^c + (1 - g_t)\mathbf{w}_{t-1}.$$

Neural Turing Machines (NTM)

- 1. Focusing by Content
- 2. Interpolate with previous step
- 3. Convolutional Shift

$$\tilde{w}_t(i) \longleftarrow \sum_{j=0}^{N-1} w_t^g(j)\, s_t(i-j)$$

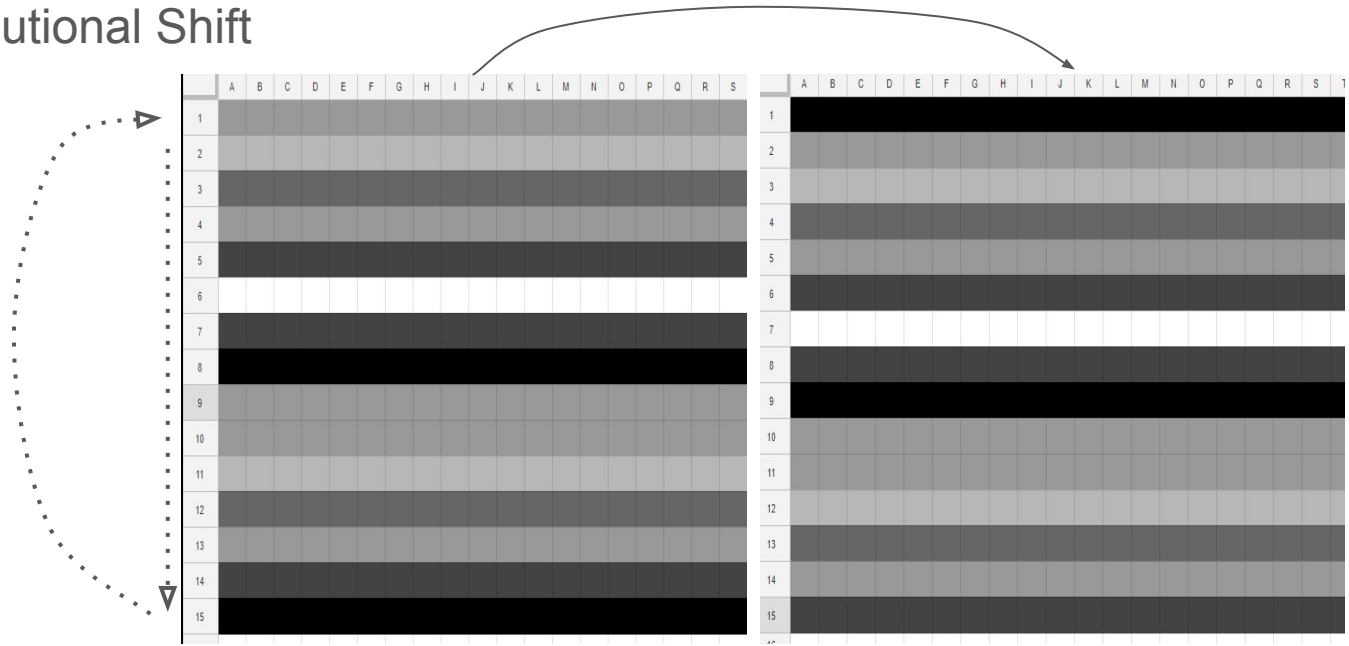
Neural Turing Machines (NTM)

- 1. Focusing by Content
- 2. Interpolate with previous step
- 3. Convolutional Shift

$$\tilde{w}_t(i) \longleftarrow \sum_{j=0}^{N-1} w_t^g(j)\, s_t(i-j)$$

Neural Turing Machines (NTM)

- 1. Focusing by Content
- 2. Interpolate with previous step
- 3. Convolutional Shift



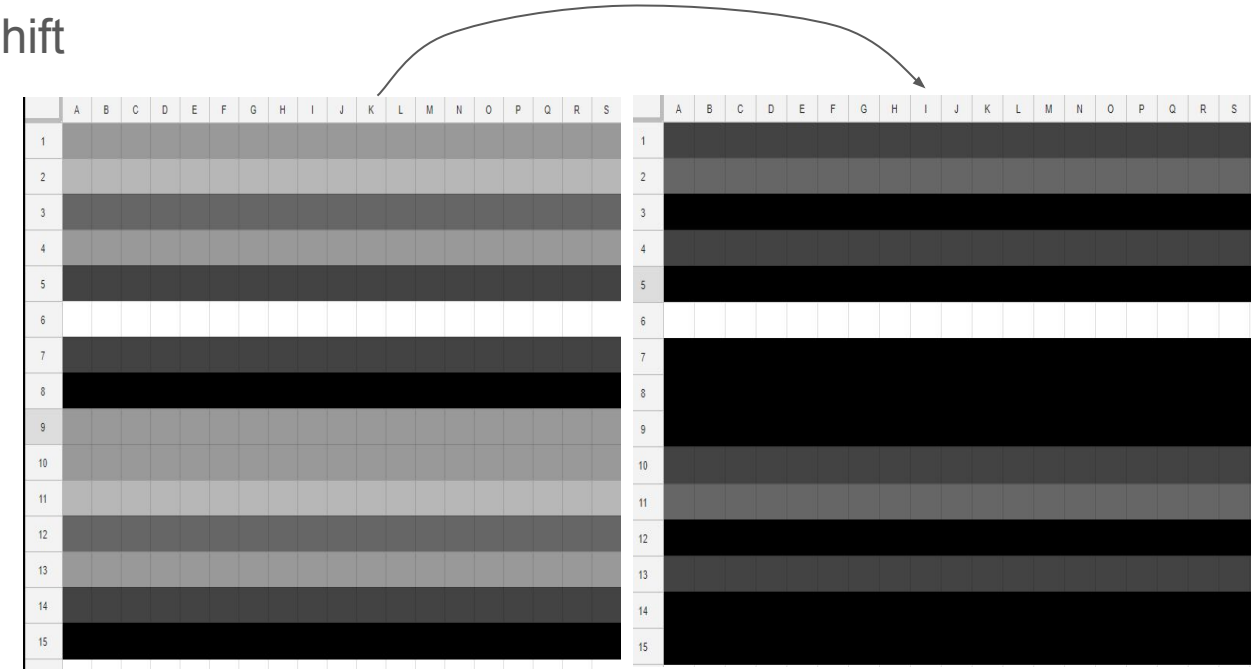
Neural Turing Machines (NTM)

- 1. Focusing by Content
- 2. Interpolate with previous step
- 3. Convolutional Shift
- 4. Shapening

$$w_t(i) \leftarrow \frac{\tilde{w}_t(i)^{\gamma_t}}{\sum_j \tilde{w}_t(j)^{\gamma_t}}$$

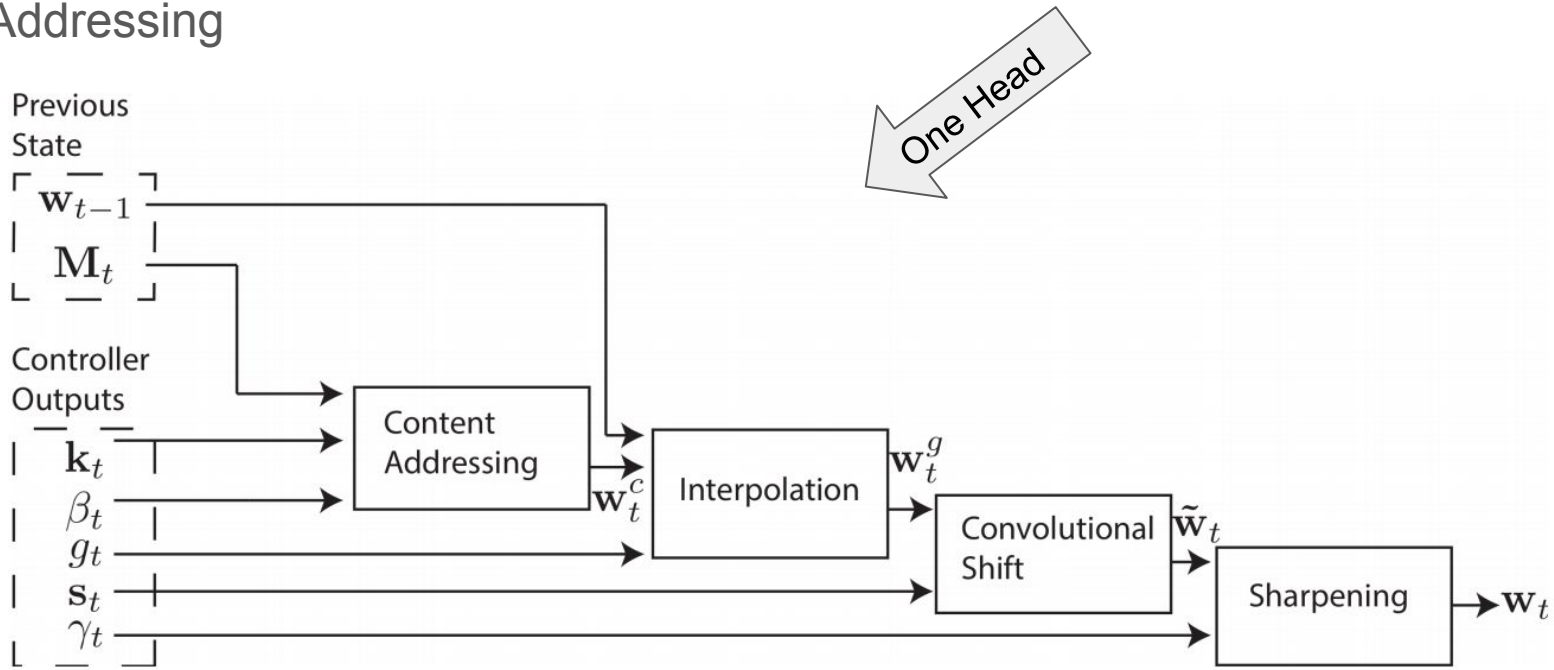
Neural Turing Machines (NTM)

- 1. Focusing by Content
- 2. Interpolate with previous step
- 3. Convolutional Shift
- 4. Shapening



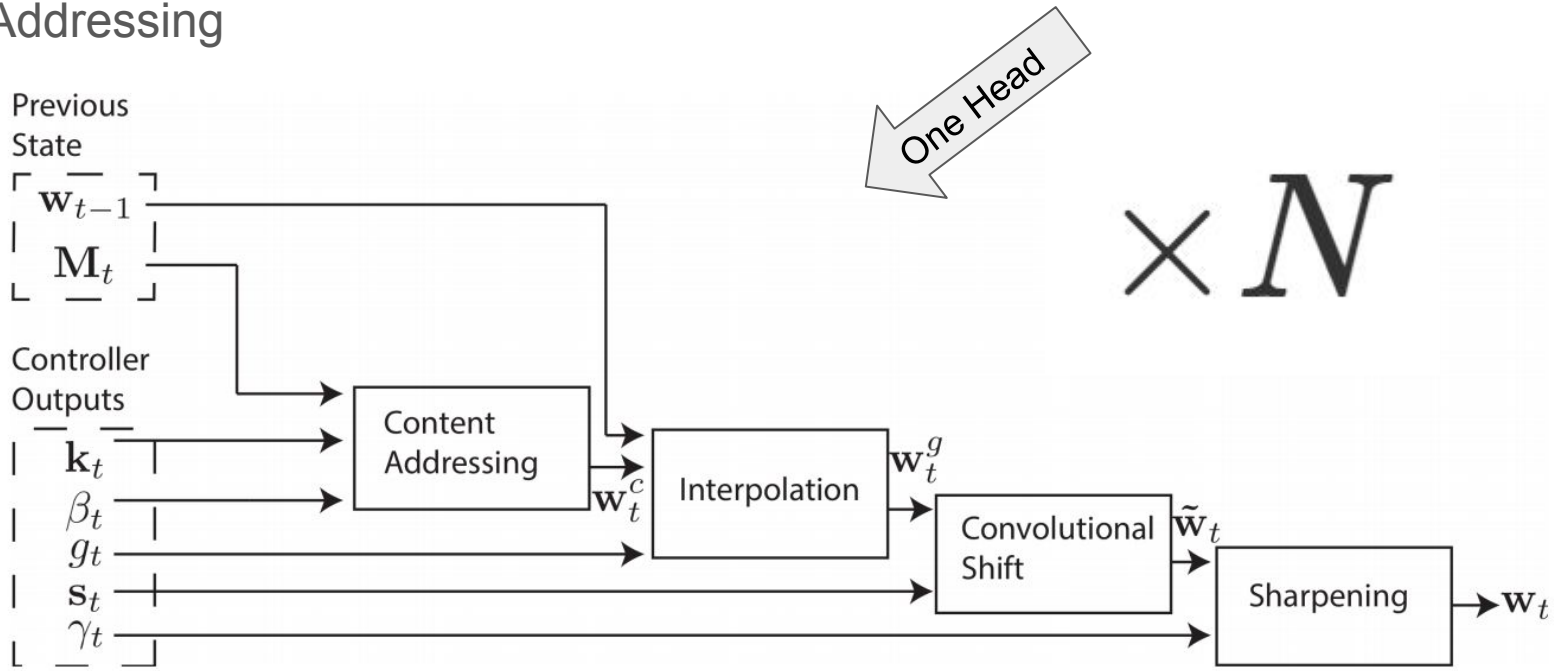
Neural Turing Machines (NTM)

- Addressing



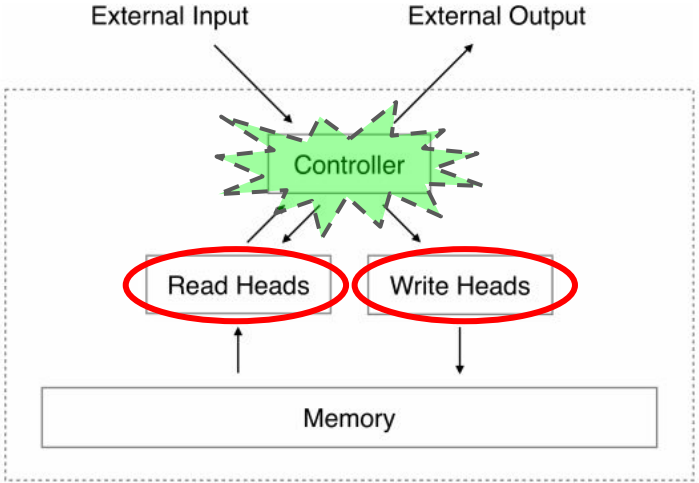
Neural Turing Machines (NTM)

- Addressing

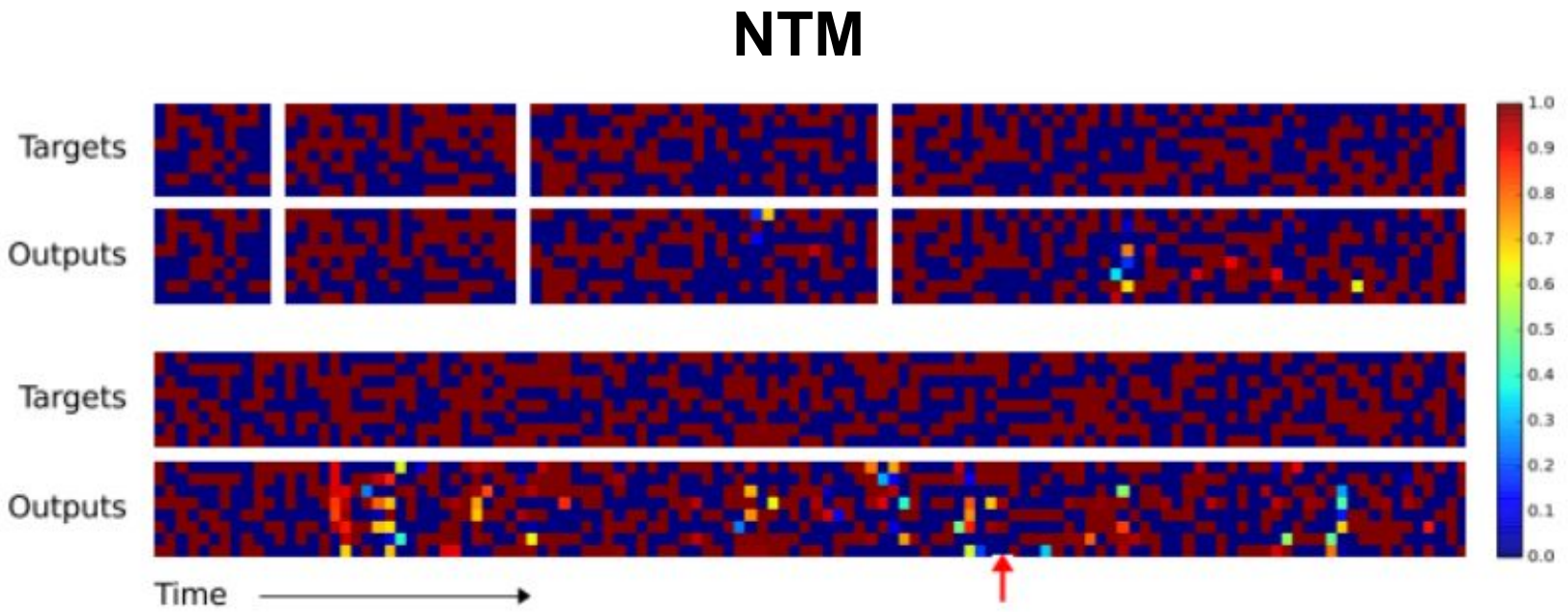


Neural Turing Machines (NTM)

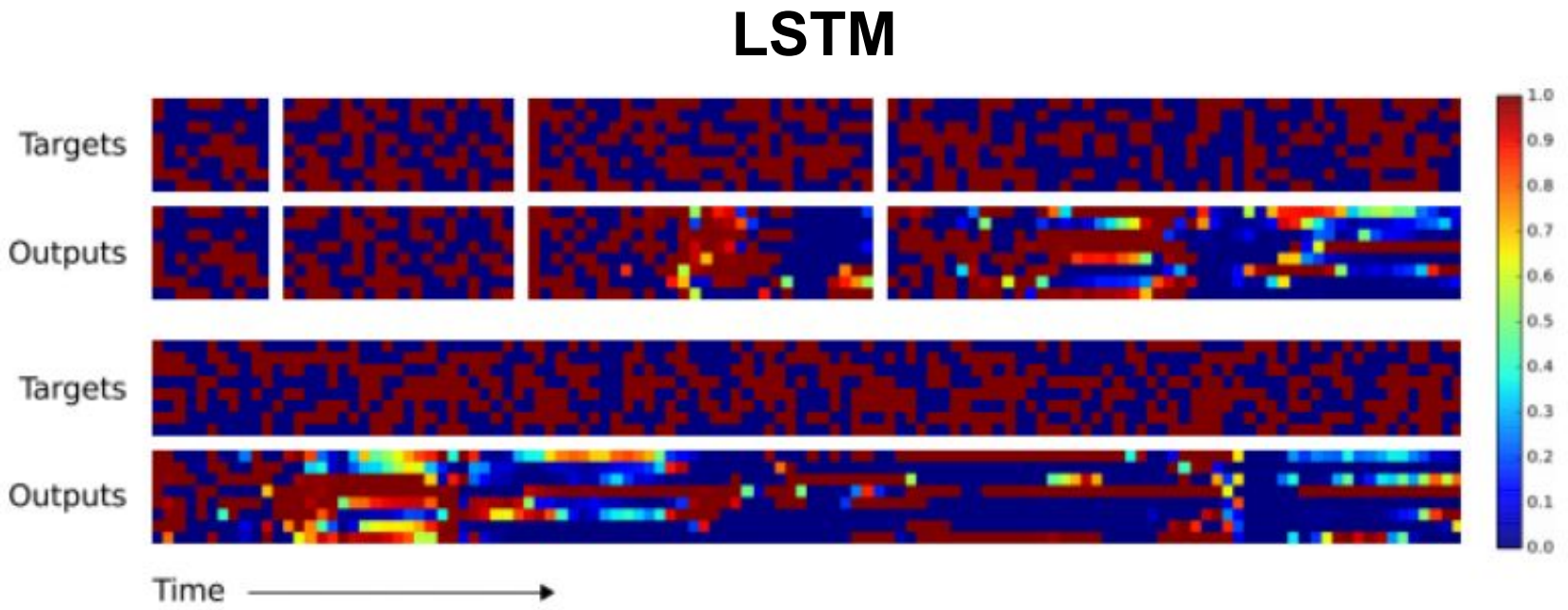
- Controller
  - Feedforward
  - LSTM
- Take input
- Predict all **red-circled** variables $\times N$
- Even if a feedforward controller is used, NTM is an RNN



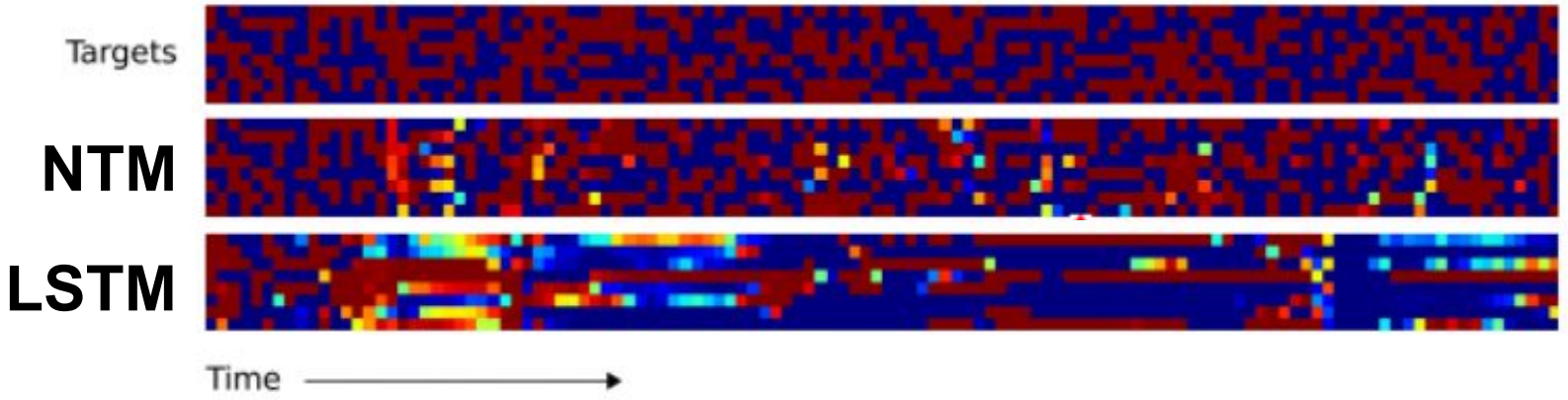
NTM: Copy Task



NTM: Copy Task



NTM: Copy Task Comparison

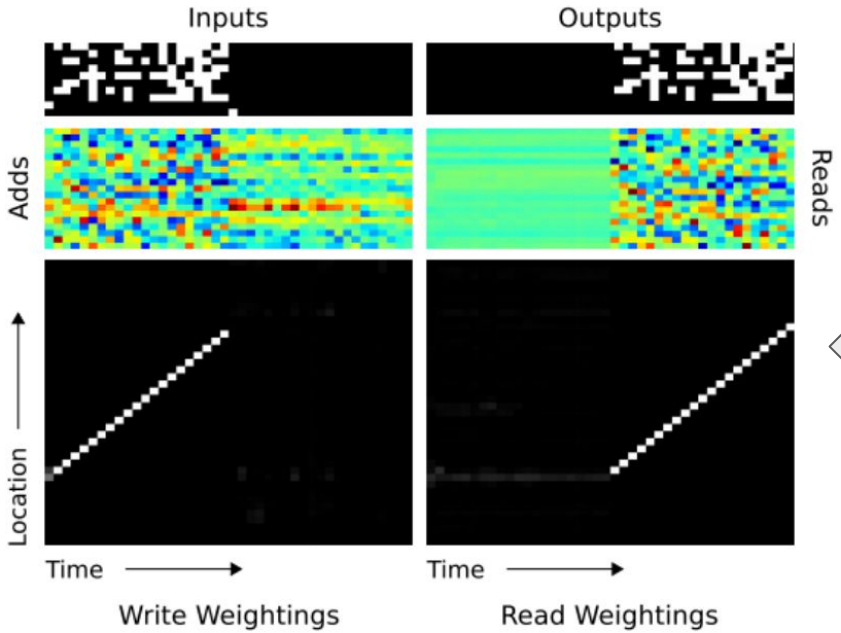


Neural Turing Machines (NTM)

- Copy Task
- Memory heads

```python
# input data
input_list = [0, 2, 4, 4, 1, 5, 2]

# model starts from here
model_memory = [0] * len(input_list)

# store everything read
loc_write = 0
for value in input_list:
    model_memory[loc_write] = value
    loc_write += 1

# write everything stored
loc_read = 0
while loc_read < loc_write:
    print(model_memory[loc_read])
    loc_read += 1
```

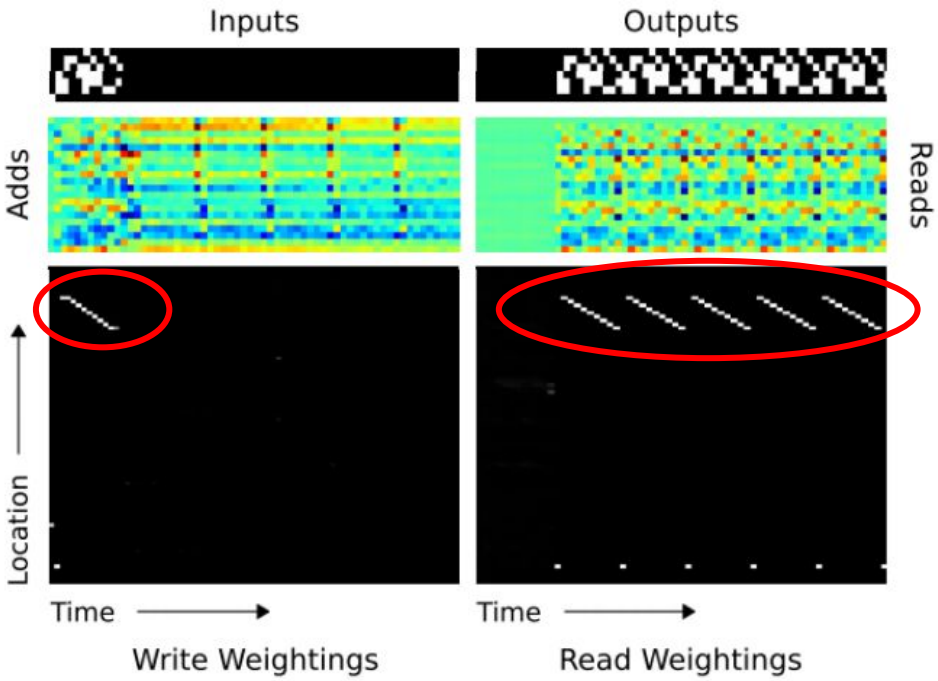
loc_read



loc_write

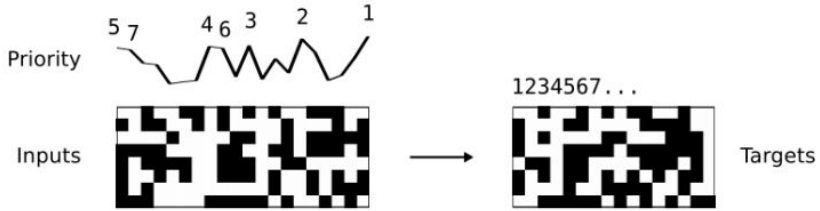
Neural Turing Machines (NTM)

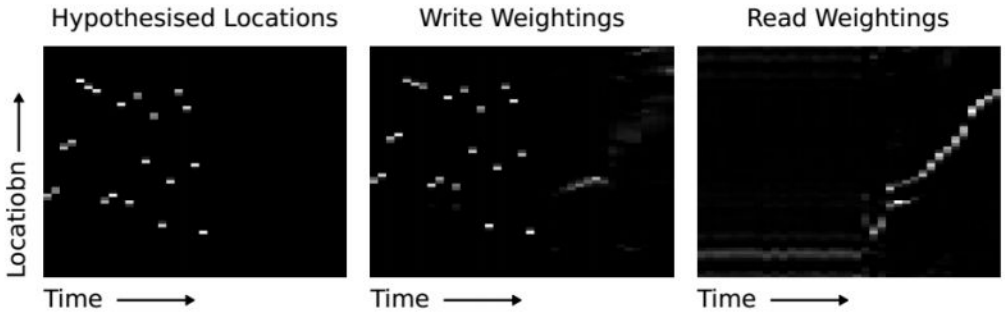
- Repeated Copy Task
- Memory heads
- White cells are positions of memory heads



Neural Turing Machines (NTM)

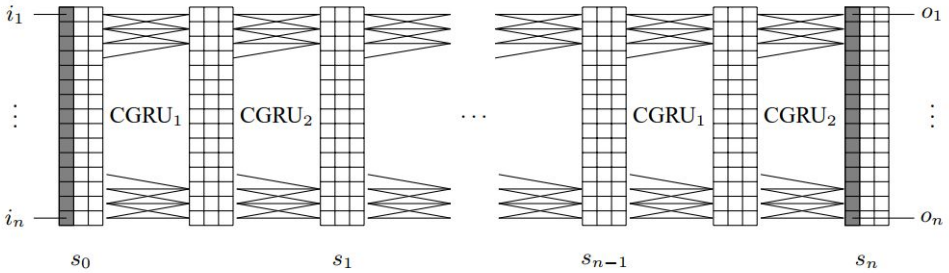
- Priority Sort





Misc

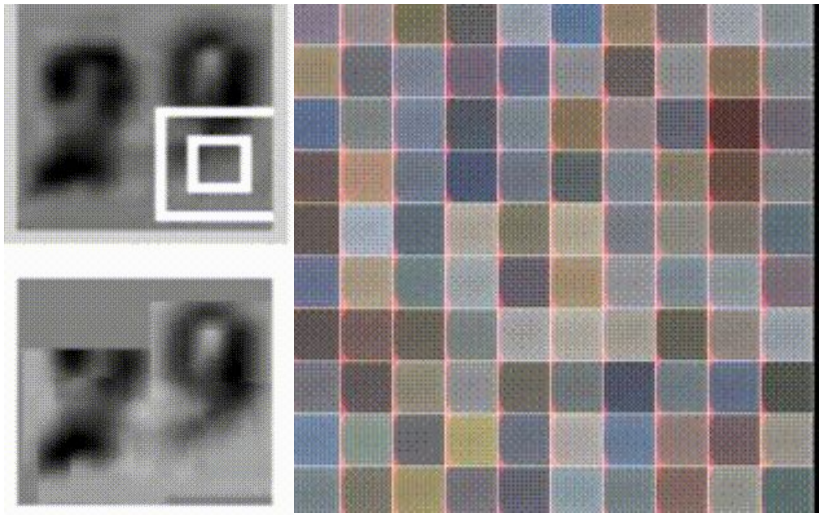
- More networks with memories
  - Memory networks
  - Differentiable Neural Computer (DNC)
- Adaptive Computing Time
- Using different weights for each step
  - HyperNetworks
- Neural GPU Learns Algorithms



More Applications

RNN without a sequence input

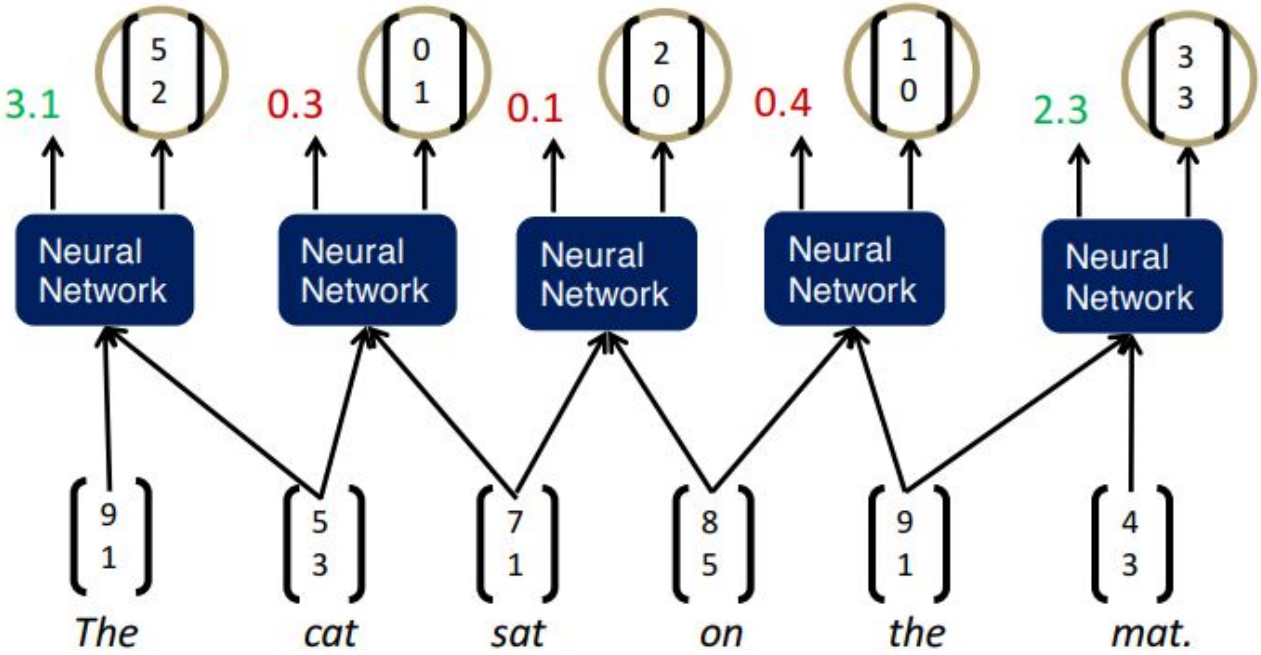
- Left
  - learns to read out house numbers from left to right
- Right
  - a recurrent network generates images of digits by learning to sequentially add color to a canvas

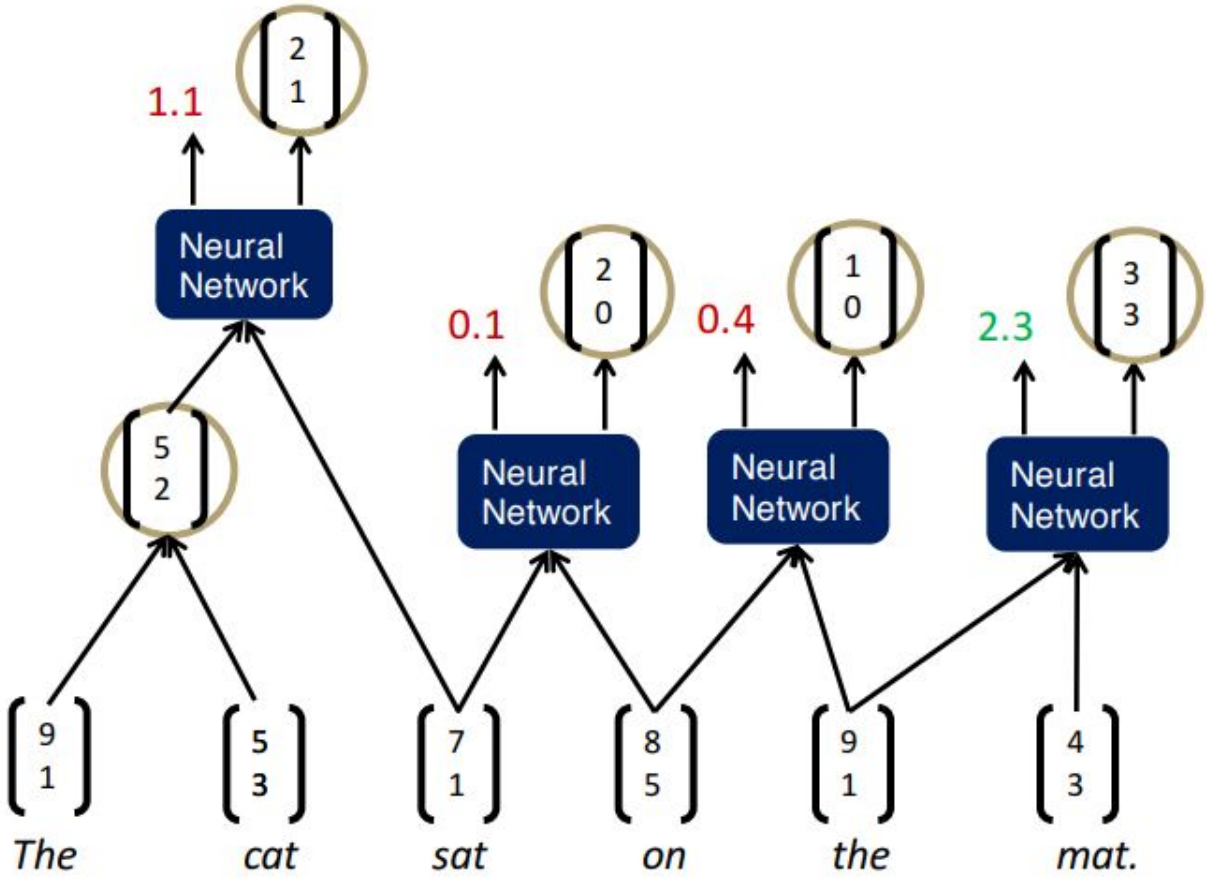


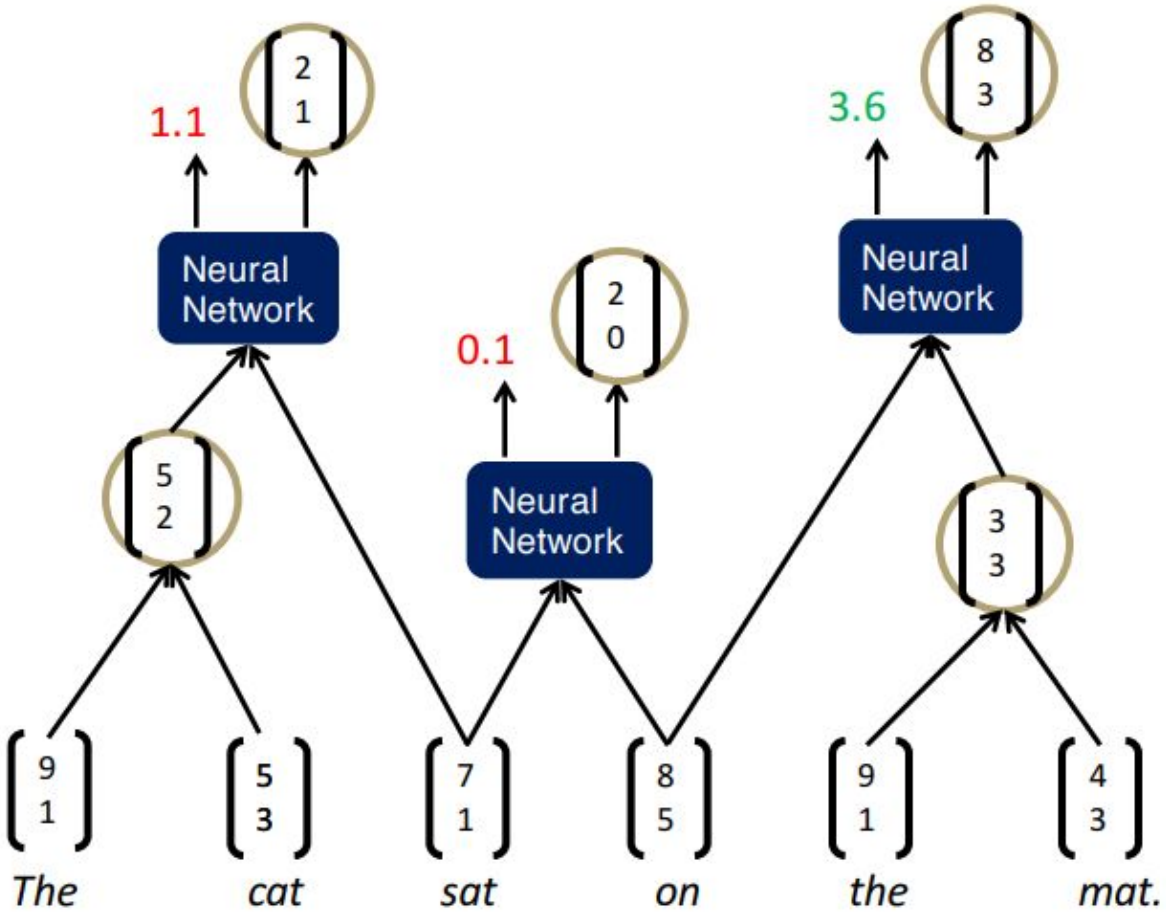
Ba, Jimmy, Volodymyr Mnih, and Koray Kavukcuoglu. "Multiple object recognition with visual attention." arXiv preprint arXiv:1412.7755 (2014).
Gregor, Karol, et al. "DRAW: A recurrent neural network for image generation." arXiv preprint arXiv:1502.04623 (2015).

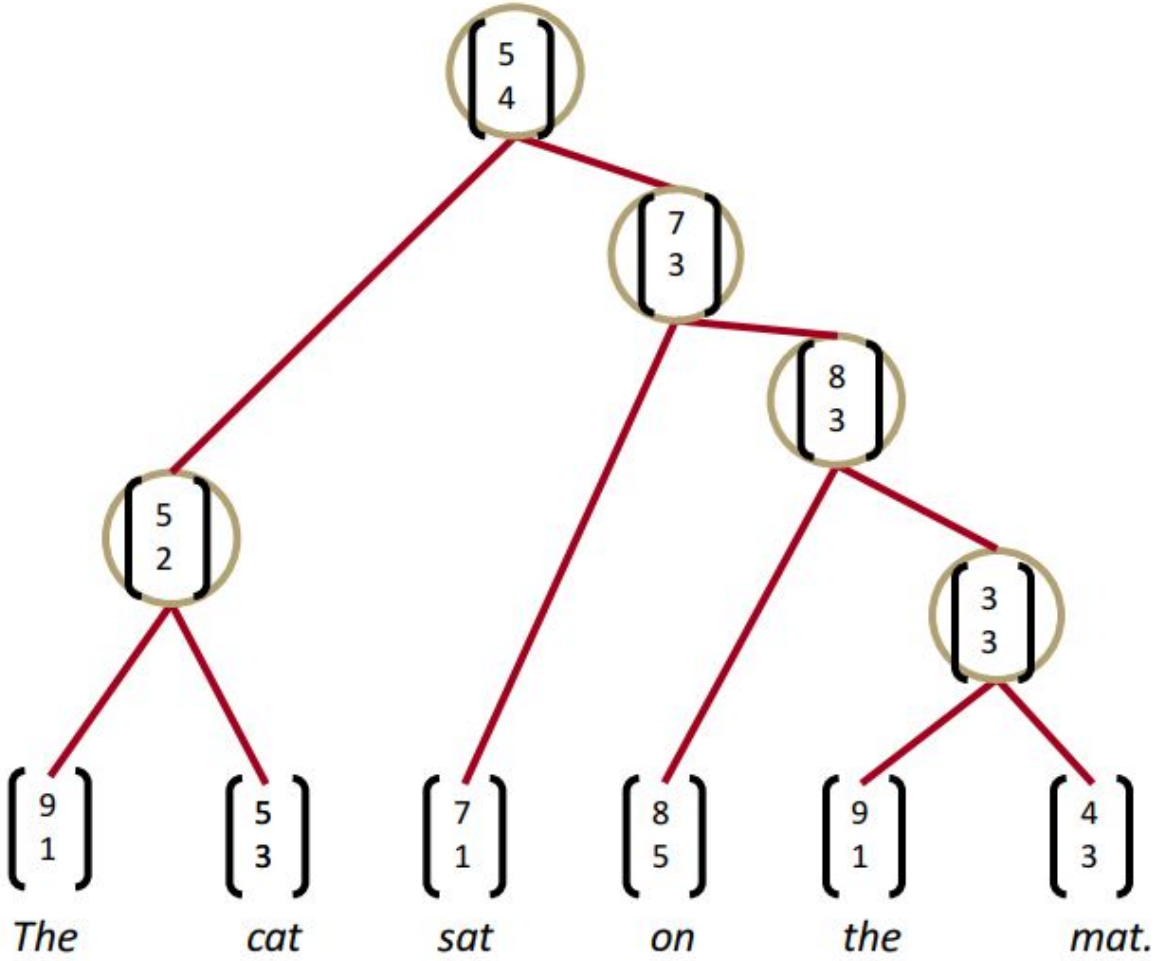
Generalizing Recurrence

- What is recurrence
  - A computation unit with shared parameter occurs at multiple places in the computation graph
    - Convolution will do too
  - … with additional states passing among them
    - That’s recurrence
- “Recursive”


3.1
0.3
0.1
0.4
2.3
Neural Network
Neural Network
Neural Network
Neural Network
Neural Network
The
cat
sat
on
the
mat.

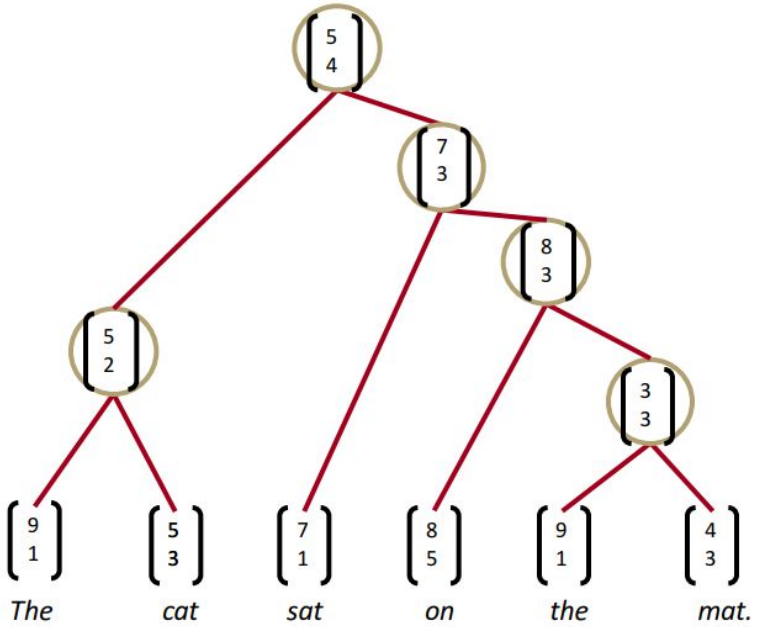

1.1
2
1
Neural Network
5
2
0.1
2
0
Neural Network
0.4
1
0
Neural Network
2.3
3
3
Neural Network
9
1
5
3
7
1
8
5
9
1
4
3
The
cat
sat
on
the
mat.


1.1
2
1
Neural Network
5
2
9
1
5
3
The
cat
0.1
2
0
Neural Network
7
1
8
5
sat
on
3.6
8
3
Neural Network
3
3
9
1
4
3
the
mat.


5
4
7
3
8
3
5
2
3
3
9
1
5
3
7
1
8
5
9
1
4
3
The
cat
sat
on
the
mat.

Recursive Neural Network

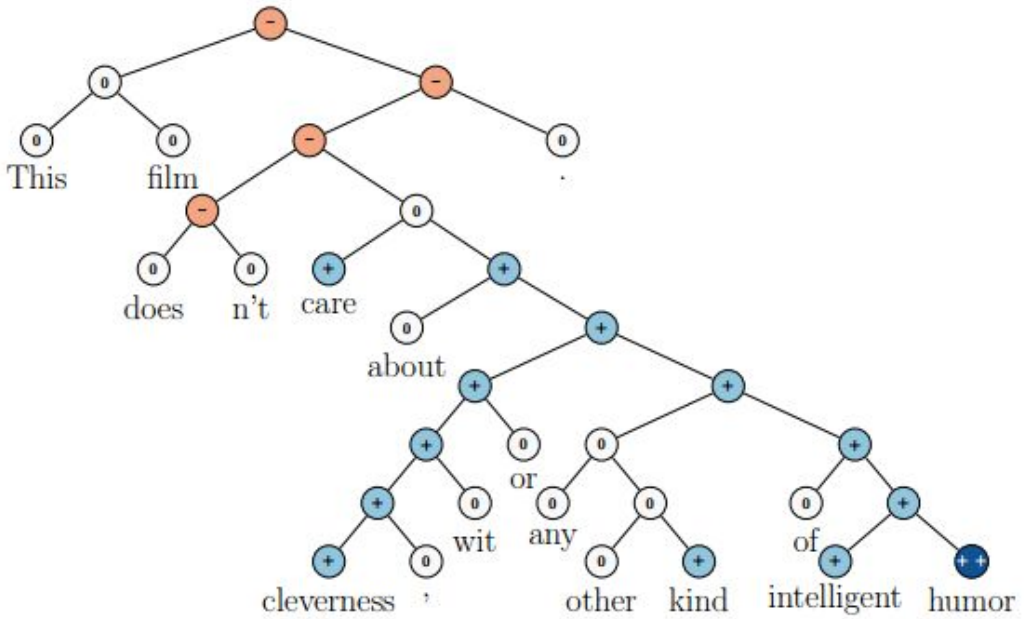
- Apply when there's tree structure in data
  - For natural language use The Standford Parser to build the syntax tree given a sentence



http://cs224d.stanford.edu/lectures/CS224d-Lecture10.pdf
https://nlp.stanford.edu/software/lex-parser.shtml

Recursive Neural Network

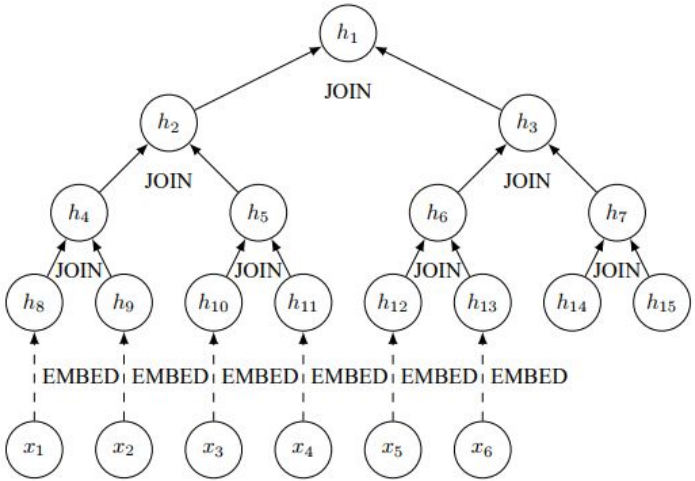
- Bottom-up aggregation of information
  - Sentiment Analysis

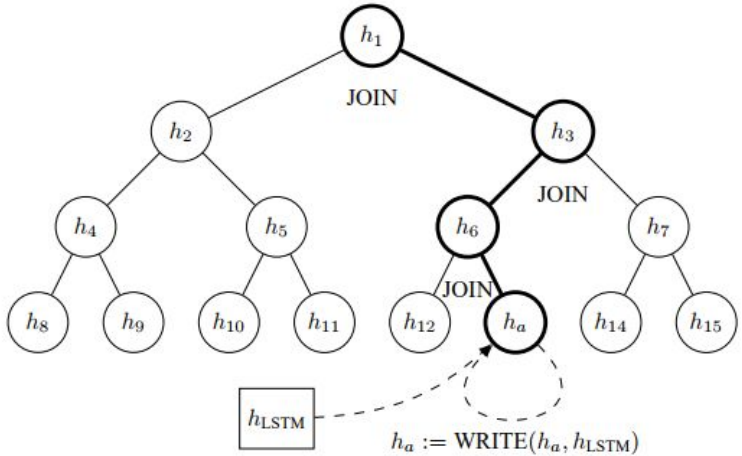


Socher, Richard, et al. "Recursive deep models for semantic compositionality over a sentiment treebank." Proceedings of the 2013 conference on empirical methods in natural language processing. 2013.

Recursive Neural Network

- As a lookup table

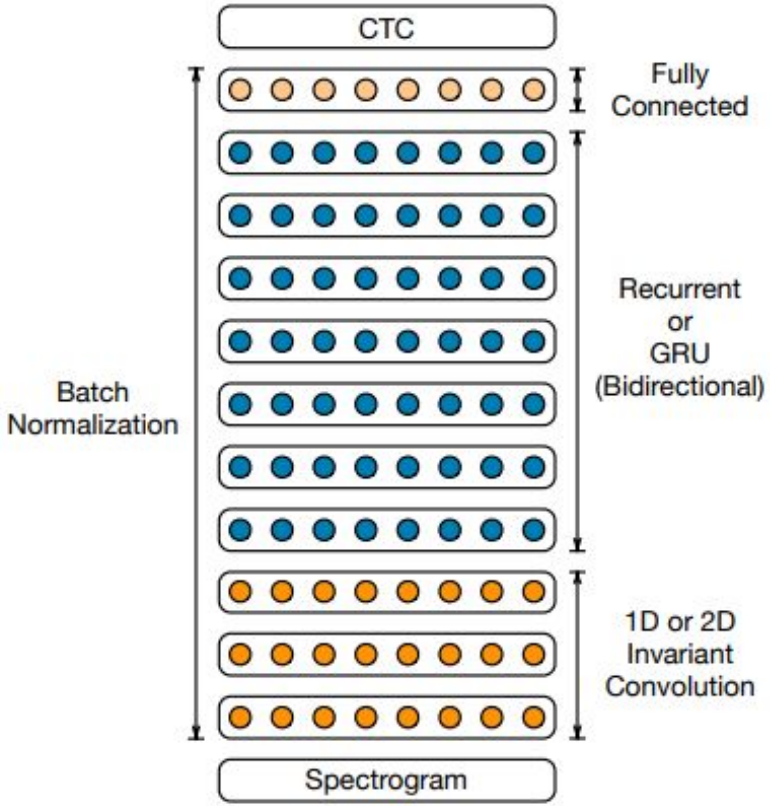




Andrychowicz, Marcin, and Karol Kurach. "Learning efficient algorithms with hierarchical attentive memory." arXiv preprint arXiv:1602.03218 (2016).

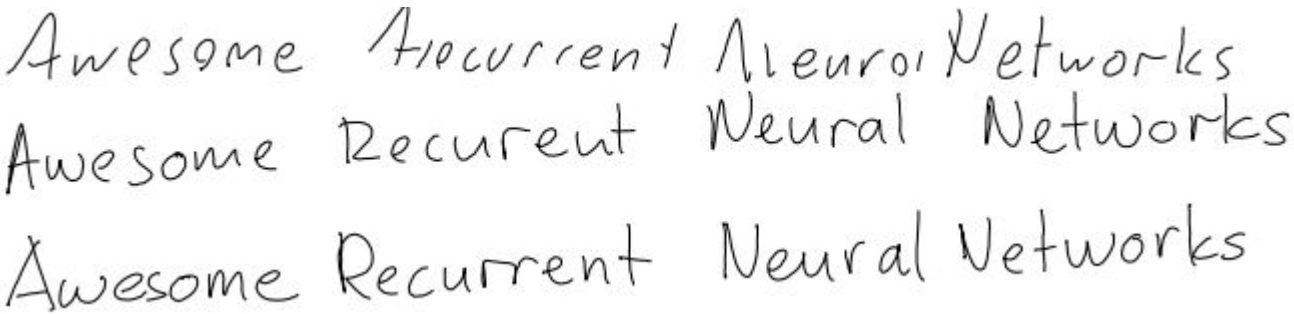
Speech Recognition

- Deep Speech 2
  - Baidu



Amodei, Dario, et al. "Deep speech 2: End-to-end speech recognition in english and mandarin." International Conference on Machine Learning. 2016.

Generating Sequence

- Language modeling
  - Input: “A”
  - Output: “A quick brown fox jumps over the lazy dog.”
- Handwriting stroke generation
  - 

https://www.cs.toronto.edu/~graves/handwriting.html

Question Answering

1. **Mary** moved to the **bathroom**
2. **John** went to the **hallway**
3. **Where** is **Mary**?
4. Answer: **bathroom**

Weston, Jason, Sumit Chopra, and Antoine Bordes. "Memory networks." arXiv preprint arXiv:1410.3916 (2014).
Sukhbaatar, Sainbayar, Jason Weston, and Rob Fergus. "End-to-end memory networks." Advances in neural information processing systems. 2015.
Andreas, Jacob, et al. "Learning to compose neural networks for question answering." arXiv preprint arXiv:1601.01705 (2016).
http://cs.umd.edu/~miyyer/data/deepqa.pdf
https://research.fb.com/downloads/babi/

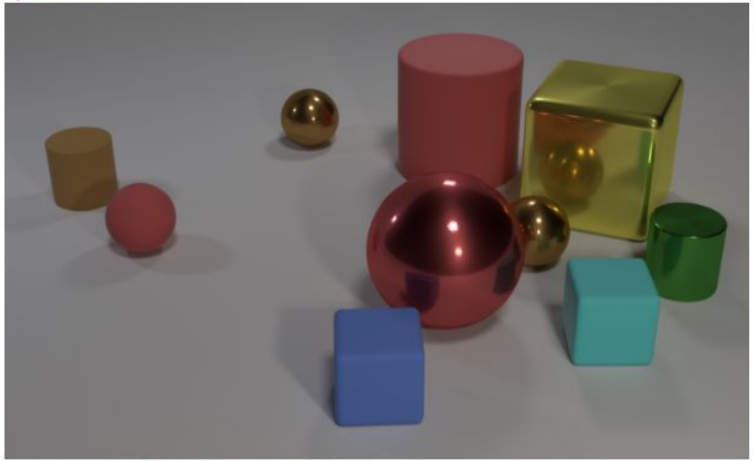
Visual Question Answering

Antol, Stanislaw, et al. "Vqa: Visual question answering." Proceedings of the IEEE International Conference on Computer Vision. 2015.

Visual Question Answering

- Reason the **relations** among Objects in image
-
- “**What size** is the **cylinder** that is **left of** the **brown metal** thing that is **left of** the **big sphere**”
-
- Dataset
  - CLEVR

Questions in CLEVR test various aspects of visual reasoning including **attribute identification**, **counting**, **comparison**, **spatial relationships**, and **logical operations**.

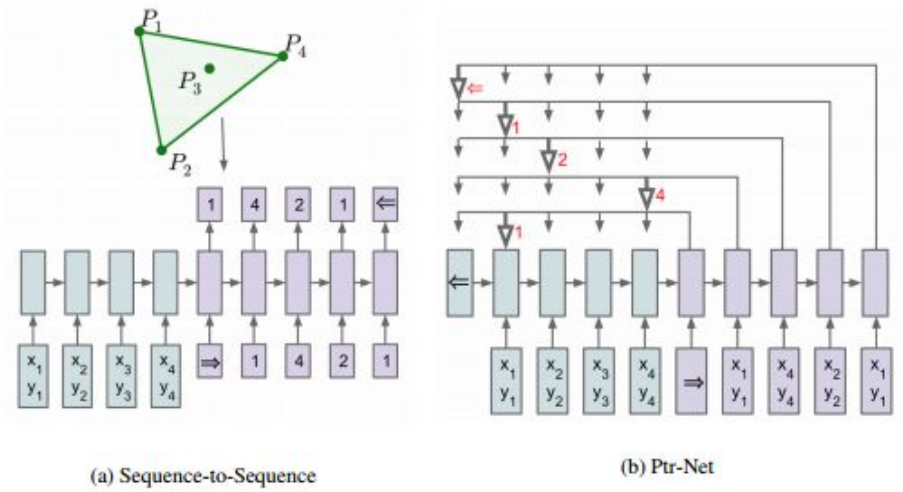


Q: Are there an **equal number** of **large things** and **metal spheres**?
Q: **What size** is the **cylinder** **that is left of** the **brown metal** thing **that is left of** the **big sphere**?
Q: There is a **sphere** with the **same size as** the **metal cube**; is it **made of the same material as** the **small red sphere**?
Q: **How many** objects are **either small cylinders** or **red** things?

https://distill.pub/2016/augmented-rnns/
http://cs.stanford.edu/people/jcjohns/clevr/

Combinatorial Problems

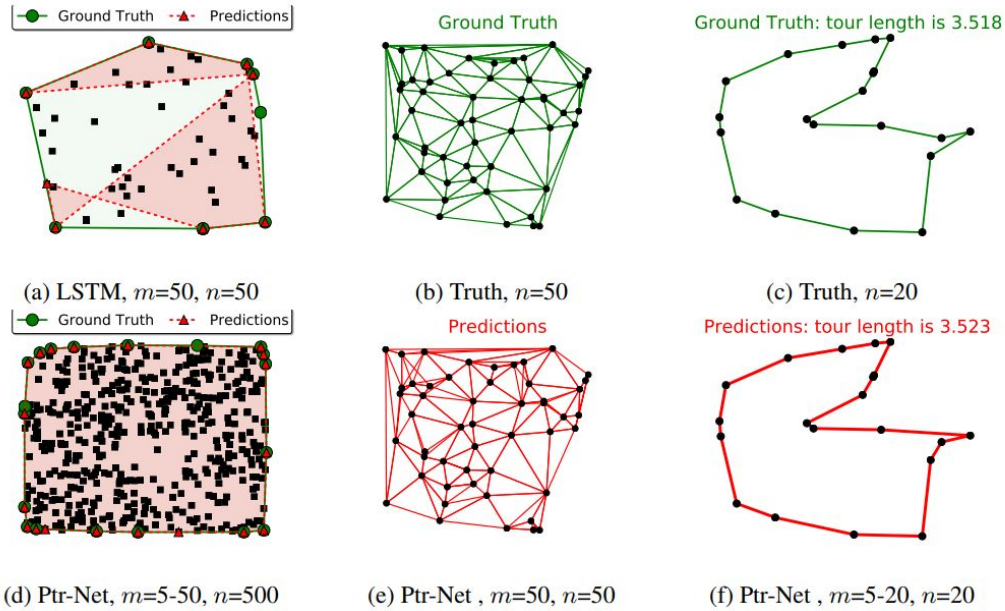
- Pointer Networks
  - Convex Hull
  - TSP
  - Delaunay triangulation
- Cross-entropy loss on Soft-attention
- Application in Vision
  - Object Tracking



MLA Vinyals, Oriol, Meire Fortunato, and Navdeep Jaitly. "Pointer networks." Advances in Neural Information Processing Systems. 2015.

Combinatorial Problems

- Pointer Networks
  - Convex Hull
  - TSP
  - Delaunay triangulation
- Cross-entropy loss on Sof
- Application in Vision
  - Object Tracking



(a) LSTM, m=50, n=50 (b) Truth, n=50 (c) Truth, n=20

(d) Ptr-Net, m=5-50, n=500 (e) Ptr-Net , m=50, n=50 (f) Ptr-Net , m=5-20, n=20

MLA Vinyals, Oriol, Meire Fortunato, and Navdeep Jaitly. "Pointer networks." Advances in Neural Information Processing Systems. 2015.

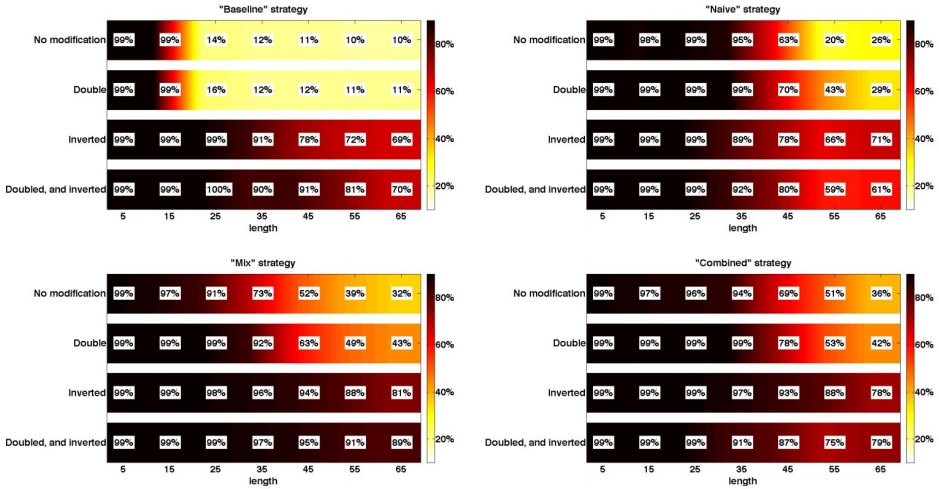
Learning to execute

- Executing program

Input:

```
f=483654
for x in range(9):f-=913681
a=f
for x in range(12):a-=926785
print((124798 if a>326533 else 576599)).
```

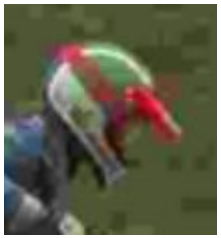
Target:	576599.
"Baseline" prediction:	176599.
"Naive" prediction:	576599.
"Mix" prediction:	576599.
"Combined" prediction:	576599.

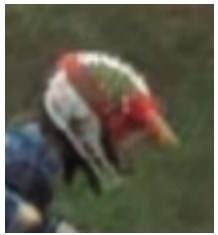


Zaremba, Wojciech, and Ilya Sutskever. "Learning to execute." arXiv preprint arXiv:1410.4615 (2014).

Compress Image

- Compete with JPEG





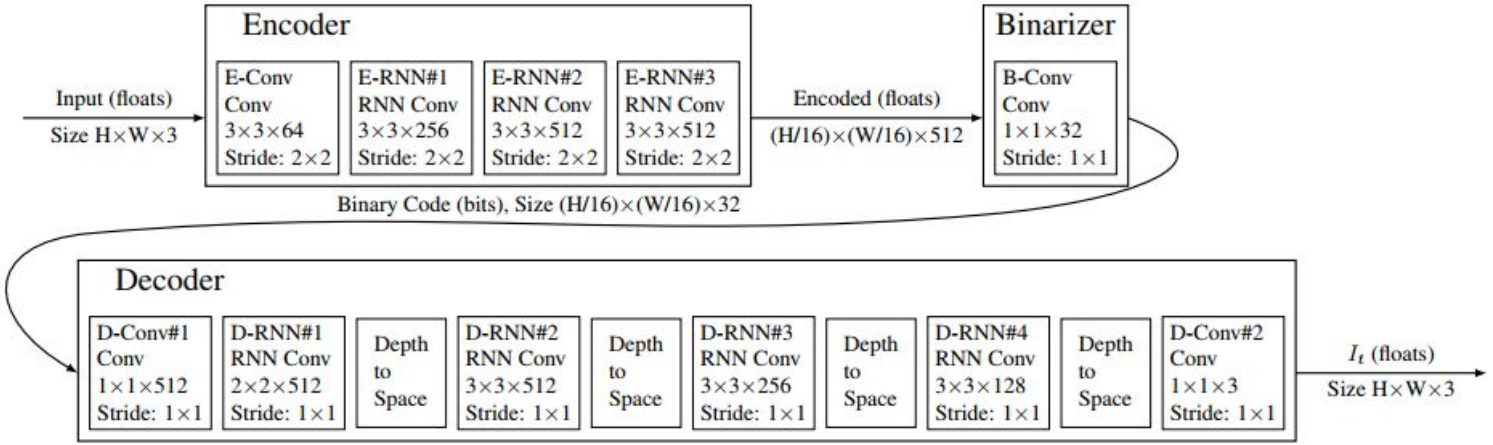
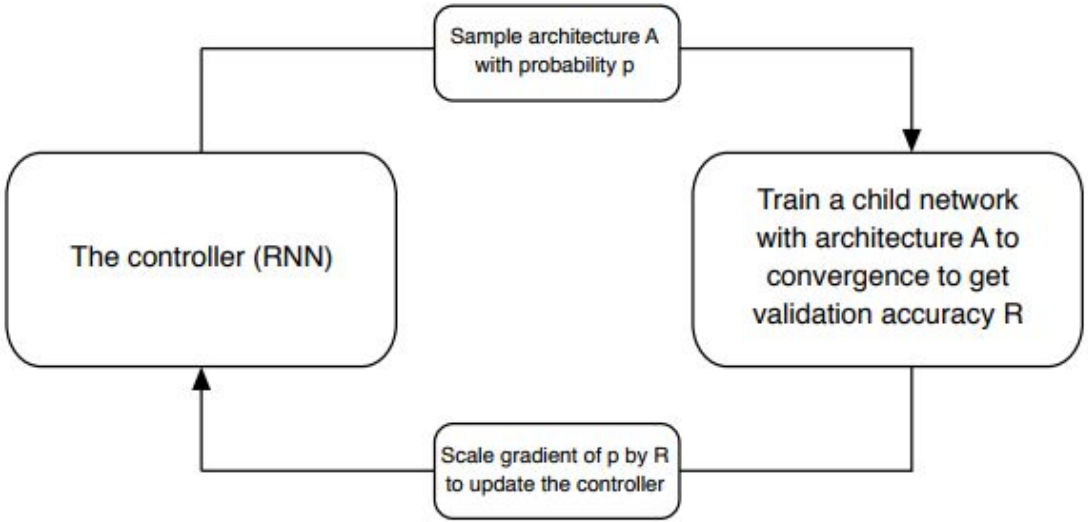


Figure 1. A single iteration of our shared RNN architecture.

Toderici, George, et al. "Full resolution image compression with recurrent neural networks." arXiv preprint arXiv:1608.05148 (2016).

Model Architecture Search

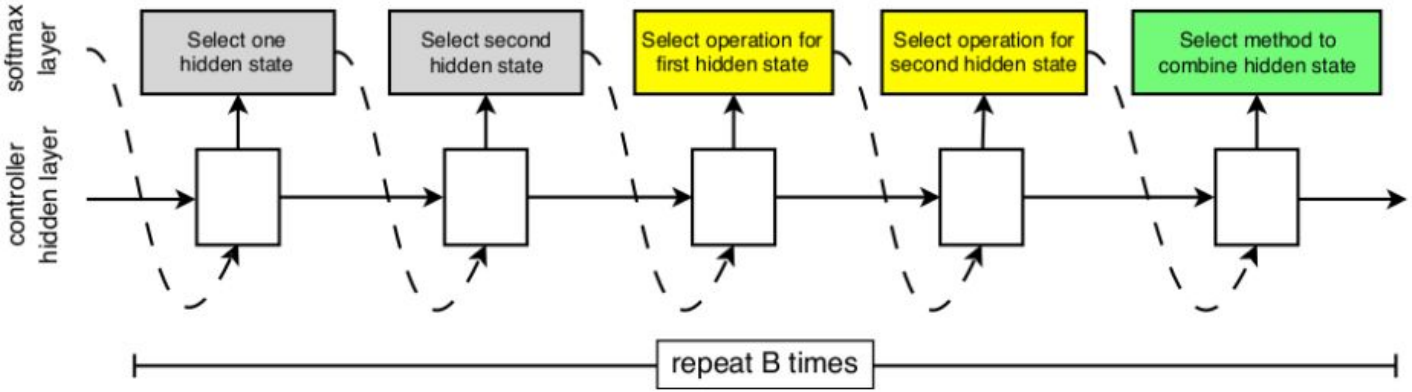
- Use an RNN to produce model architectures
  - Learned using Reinforcement Learning



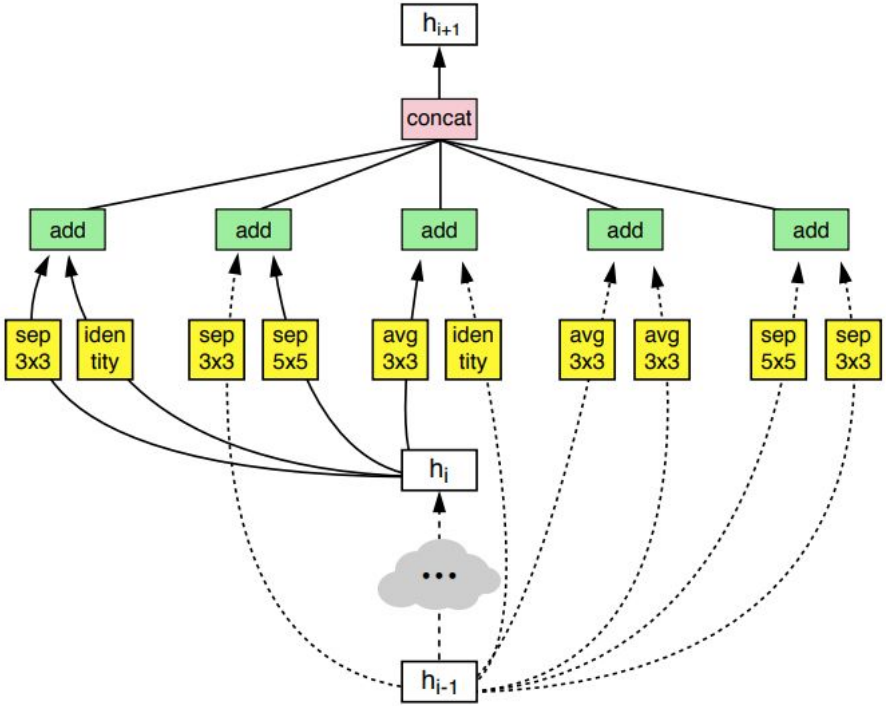
Zoph, Barret, et al. "Learning transferable architectures for scalable image recognition." arXiv preprint arXiv:1707.07012 (2017).

Model Architecture Search

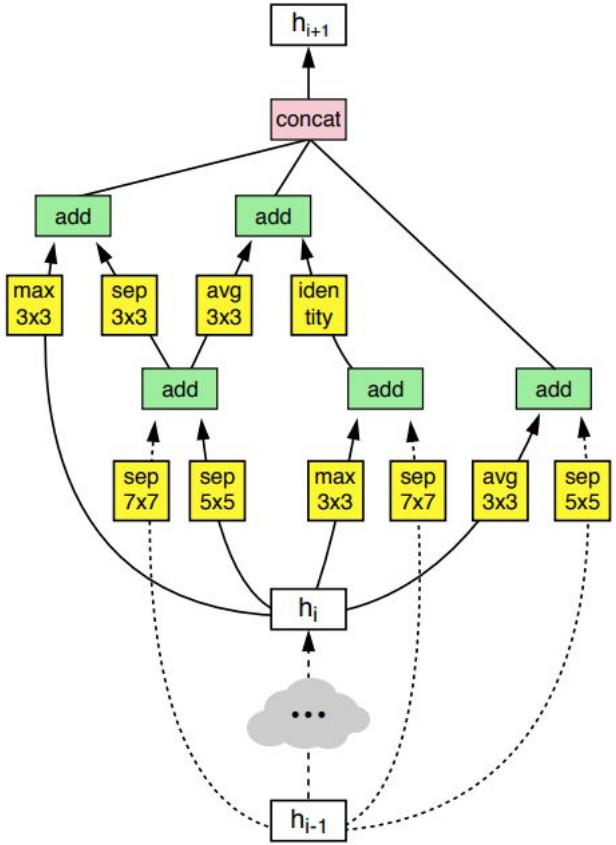
- Use an RNN to produce model architectures
  - Learned using Reinforcement Learning



Zoph, Barret, et al. "Learning transferable architectures for scalable image recognition." arXiv preprint arXiv:1707.07012 (2017).

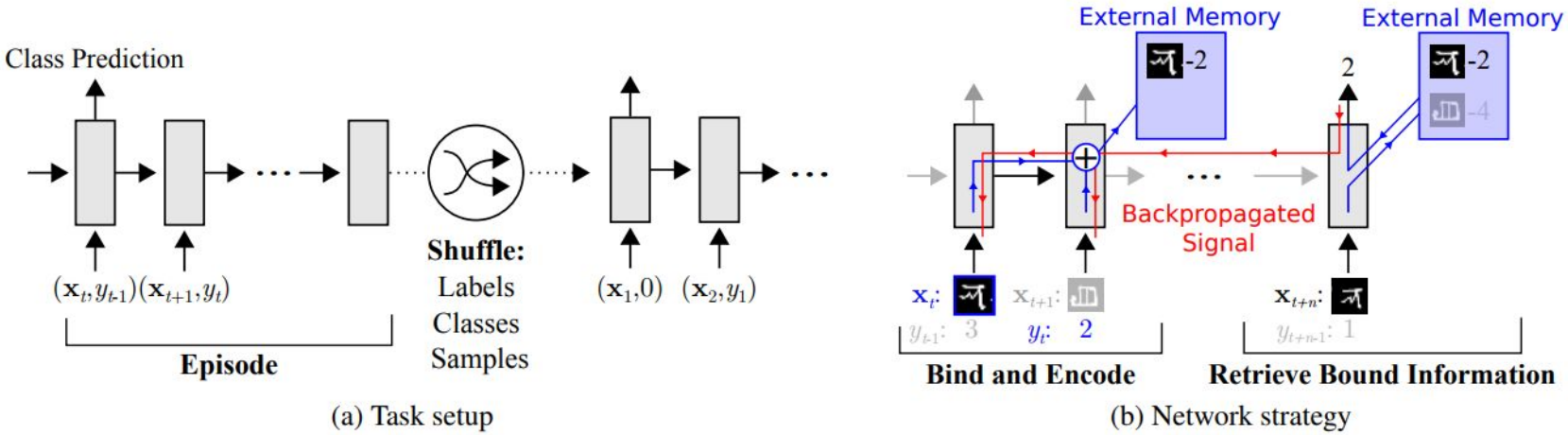


Normal Cell



Reduction Cell

Meta-Learning



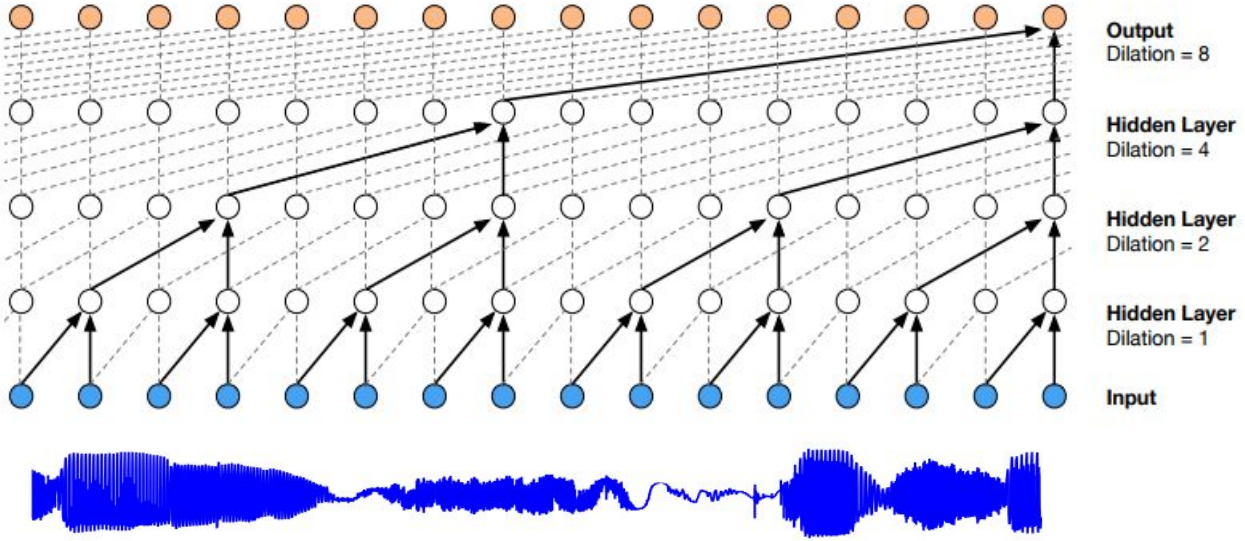
Santoro, Adam, et al. "Meta-learning with memory-augmented neural networks." International conference on machine learning. 2016.

RNN: The Good, Bad and Ugly

- Good
  - Turing Complete, strong modeling ability
- Bad
  - Dependencies between temporal connections make computation slow
    - CNNs are resurging now to predict sequence
    - WaveNet
    - Attention is all you need
      - Actually IS a kind of RNN
- Ugly
  - Generally hard to train
  - REALLY Long-term memory ??
  - The above two fights

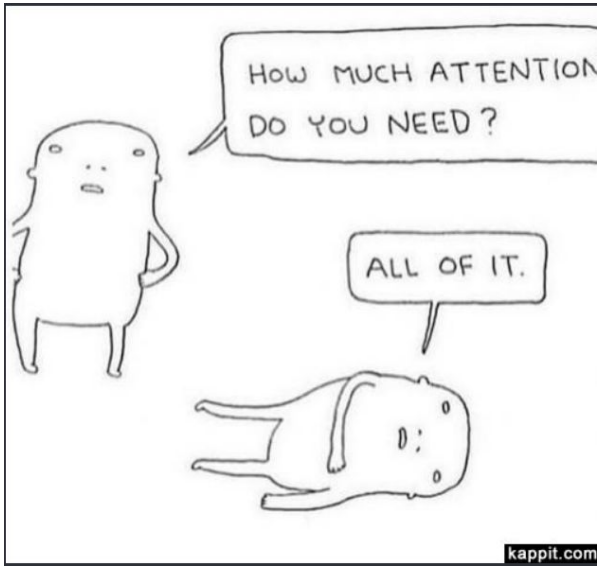
RNN’s Rival: WaveNet

- Causal Dilated Convolution



Oord, Aaron van den, et al. "Wavenet: A generative model for raw audio." arXiv preprint arXiv:1609.03499 (2016).

RNN's Rival: Attention is All You Need (Transformer)

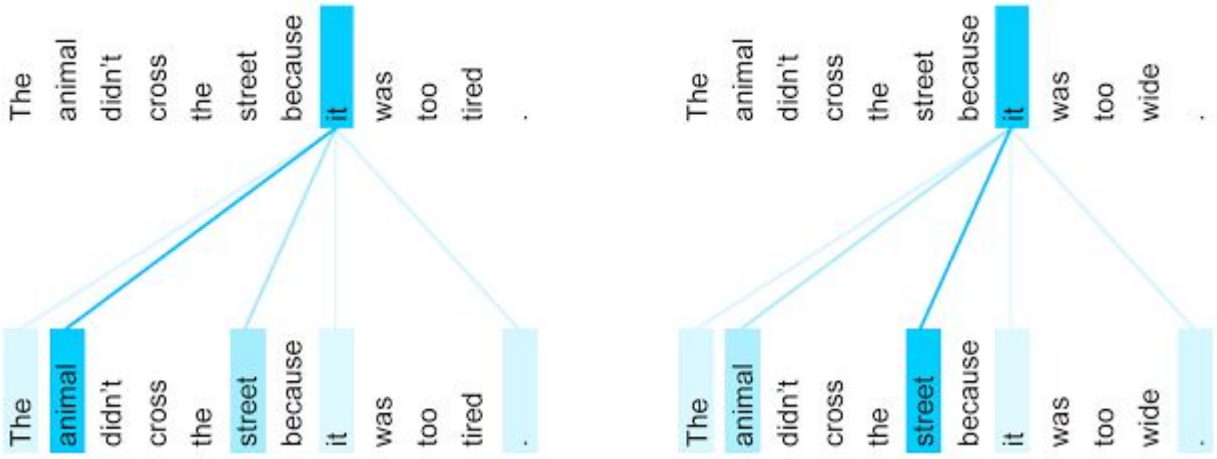


Get rid of sequential computation

Vaswani, Ashish, et al. "Attention Is All You Need." arXiv preprint arXiv:1706.03762 (2017).
https://research.googleblog.com/2017/08/transformer-novel-neural-network.html
https://courses.cs.ut.ee/MTAT.03.292/2017_fall/uploads/Main/Attention%20is%20All%20you%20need.pdf

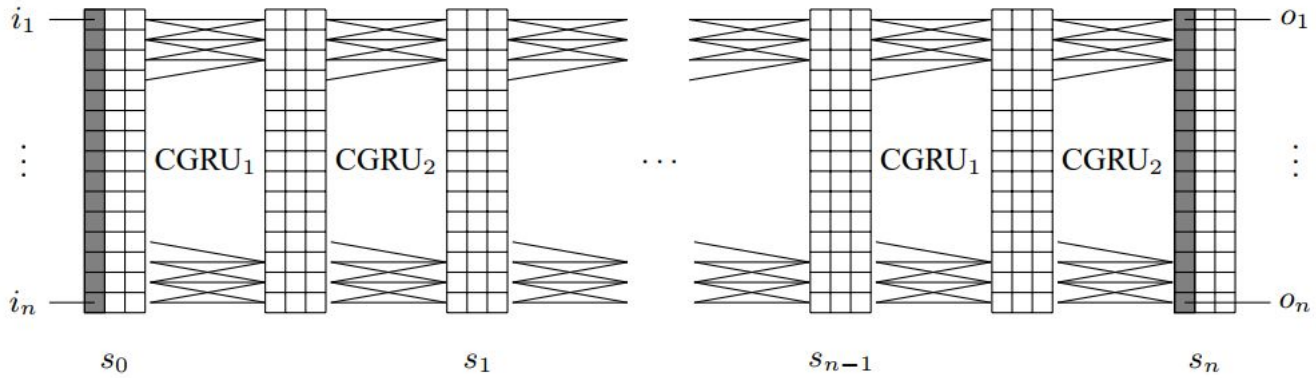
Attention is All You Need

- The encoder self-attention distribution for the word “it” from the 5th to the 6th layer of a Transformer trained on English to French translation (one of eight attention heads)



Attention is All You Need

- But … the decoder part is actually an RNN ??
  - Kinds of like neural GPU



Make RNN Great Again!

Summary

- RNN’s are great!

Summary

- RNN’s are great!

- RNN’s omnipotent!

Summary

- Turing complete 
- … So you cannot solve halting problem 
- But besides that, the only limit is your imagination.


THANKS FOR
LISTENING
ANY QUESTIONS?
NO?
GREAT!
KeepCalmAndPosters.com