Computer Vision – TP15
Attention

Miguel Coimbra, Hélder Oliveira

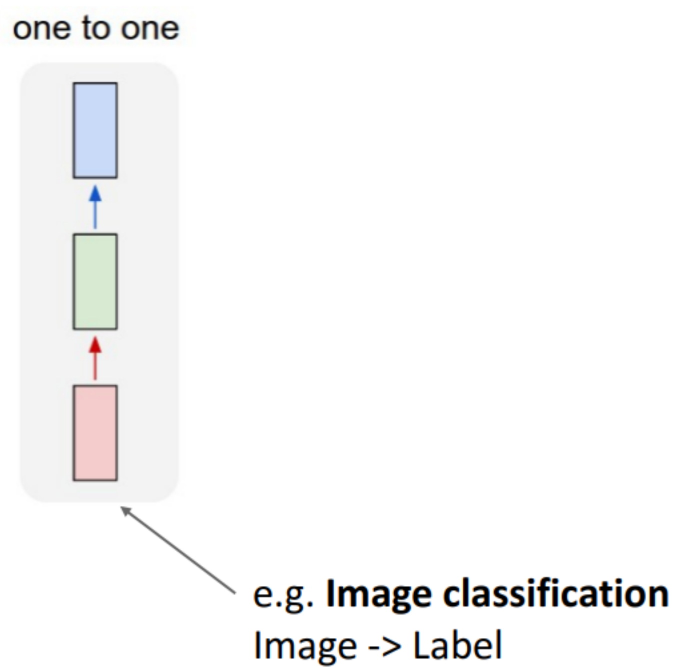
U.PORTO FC

Outline

- RNN
- Attention
- Transformers

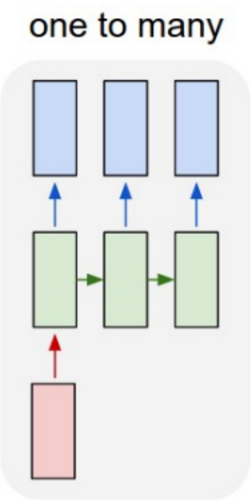
Recurrent Neural Networks

- So far: “Feedforward” Neural Networks



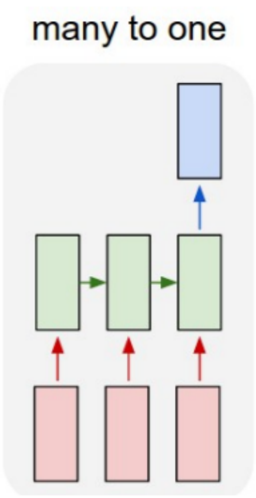
Michigan Online, Recurrent Networks, Justin Johnson

RNN: Examples



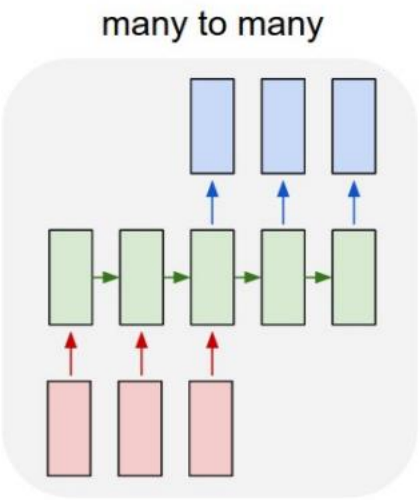
Michigan Online, Recurrent Networks, Justin Johnson

RNN: Examples



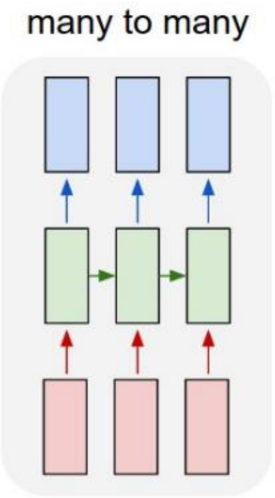
Michigan Online, Recurrent Networks, Justin Johnson

RNN: Examples



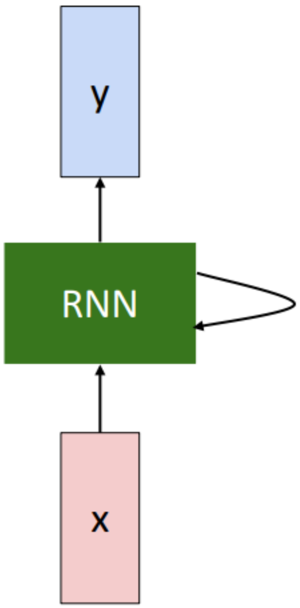
Michigan Online, Recurrent Networks, Justin Johnson

RNN: Examples



Michigan Online, Recurrent Networks, Justin Johnson

RNN: formulation



$$\boxed{h_t} = \boxed{f_W}(\boxed{h_{t-1}}, \boxed{x_t})$$

new state

some function
with parameters W

old state

input vector at
some time step

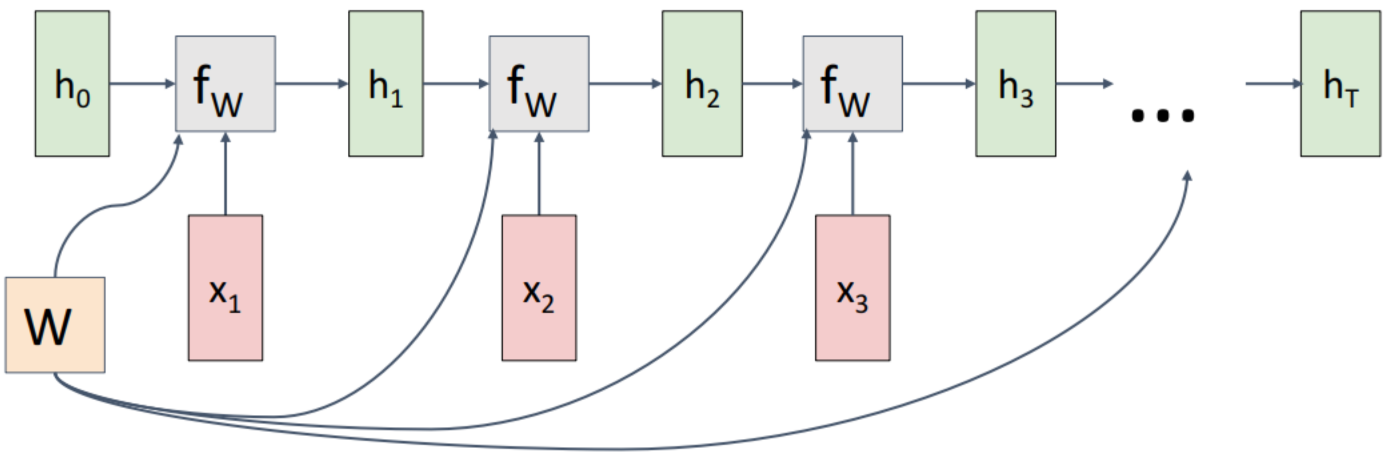
Michigan Online, Recurrent Networks, Justin Johnson

RNN: Computational Graph

- Re-use the same weight matrix at every time-step

$$\boxed{h_t} = \boxed{f_W}(\boxed{h_{t-1}}, \boxed{x_t})$$

new state
old state
input vector at some time step
some function with parameters W



Michigan Online, Recurrent Networks, Justin Johnson

RNN: Computational Graph (many to many)

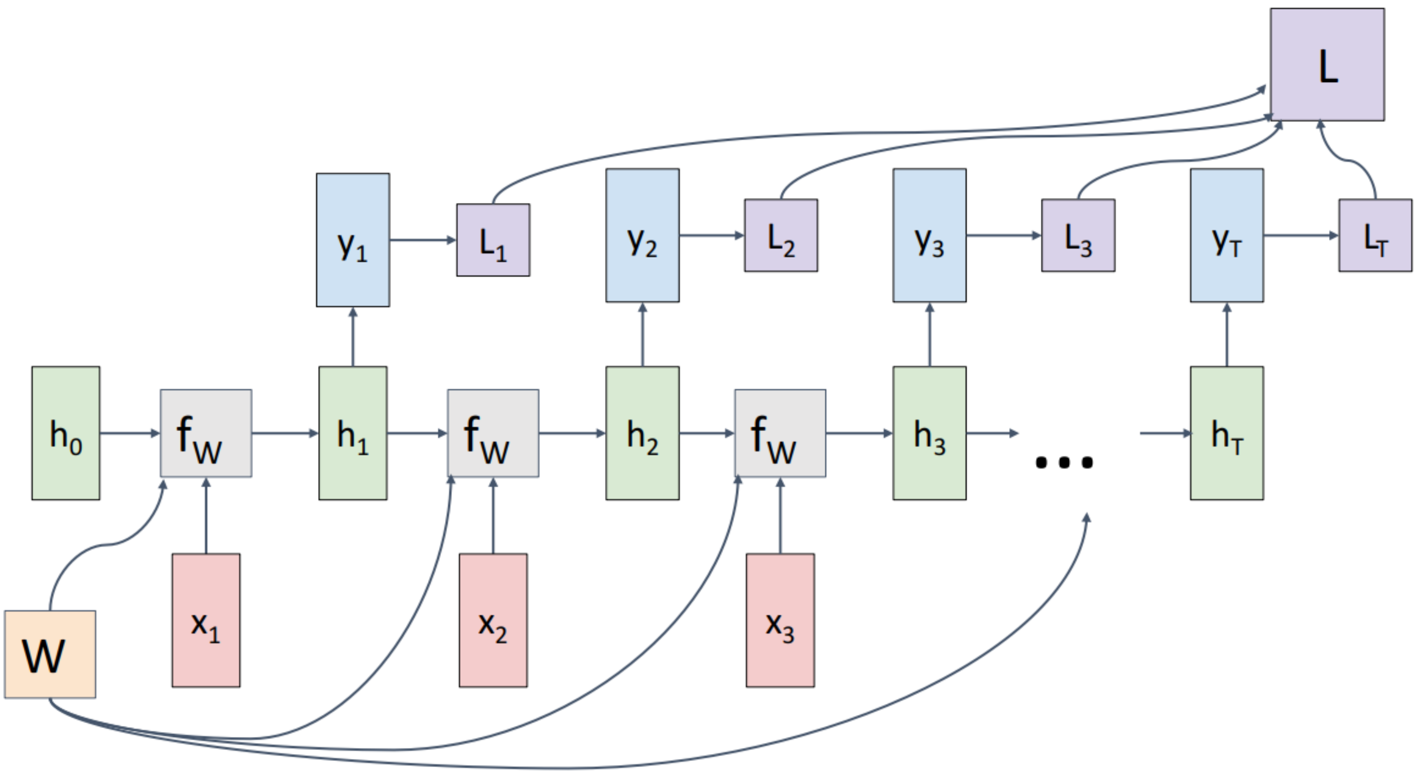
$$\boxed{h_t} = \boxed{f_W}(\boxed{h_{t-1}}, \boxed{x_t})$$

new state

old state

input vector at some time step

some function with parameters W



Michigan Online, Recurrent Networks, Justin Johnson

RNN: Computational Graph (many to one)

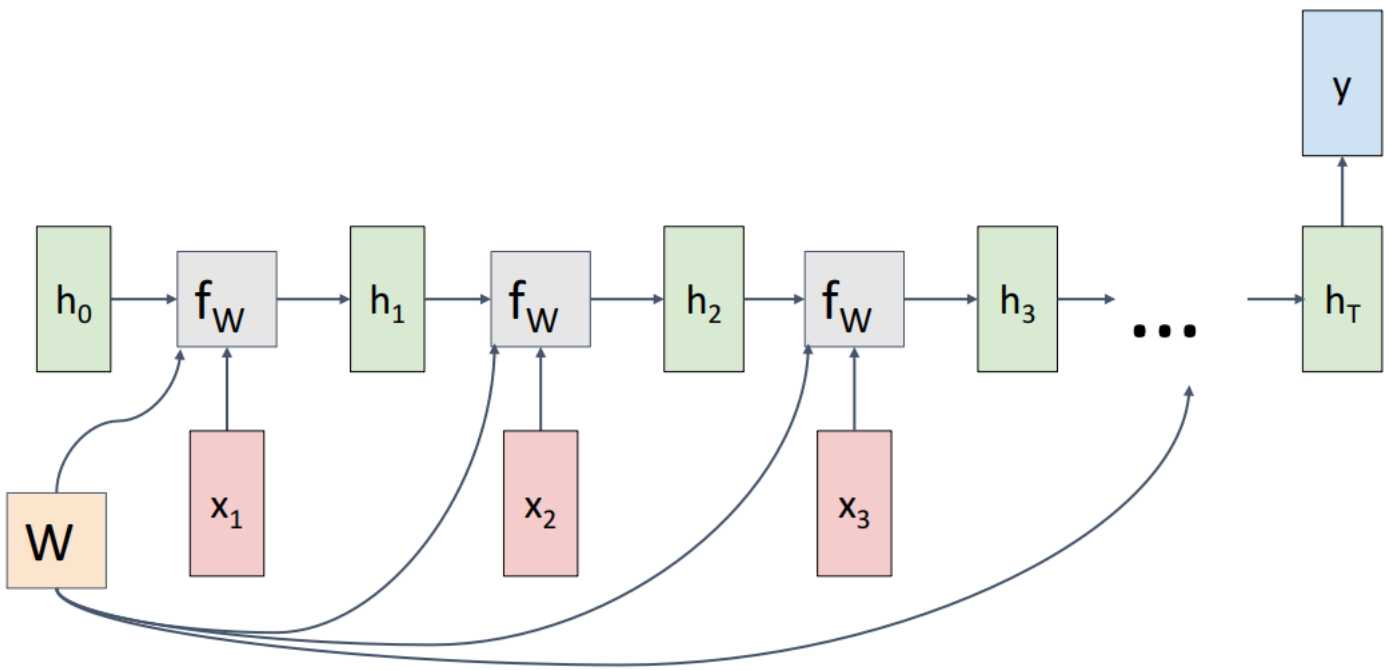
$$\boxed{h_t} = \boxed{f_W}(\boxed{h_{t-1}}, \boxed{x_t})$$

new state

old state

input vector at some time step

some function with parameters W



Michigan Online, Recurrent Networks, Justin Johnson

RNN: Computational Graph (One to many)

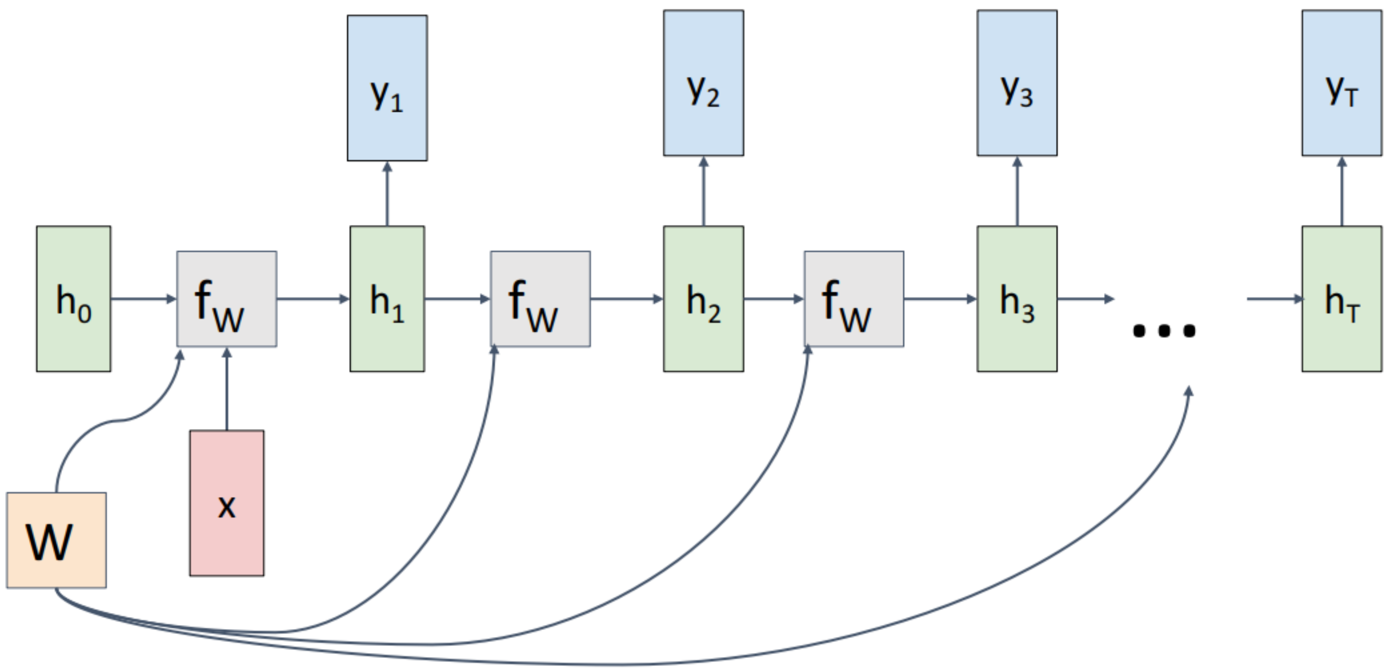
$$\boxed{h_t} = \boxed{f_W}(\boxed{h_{t-1}}, \boxed{x_t})$$

new state

old state

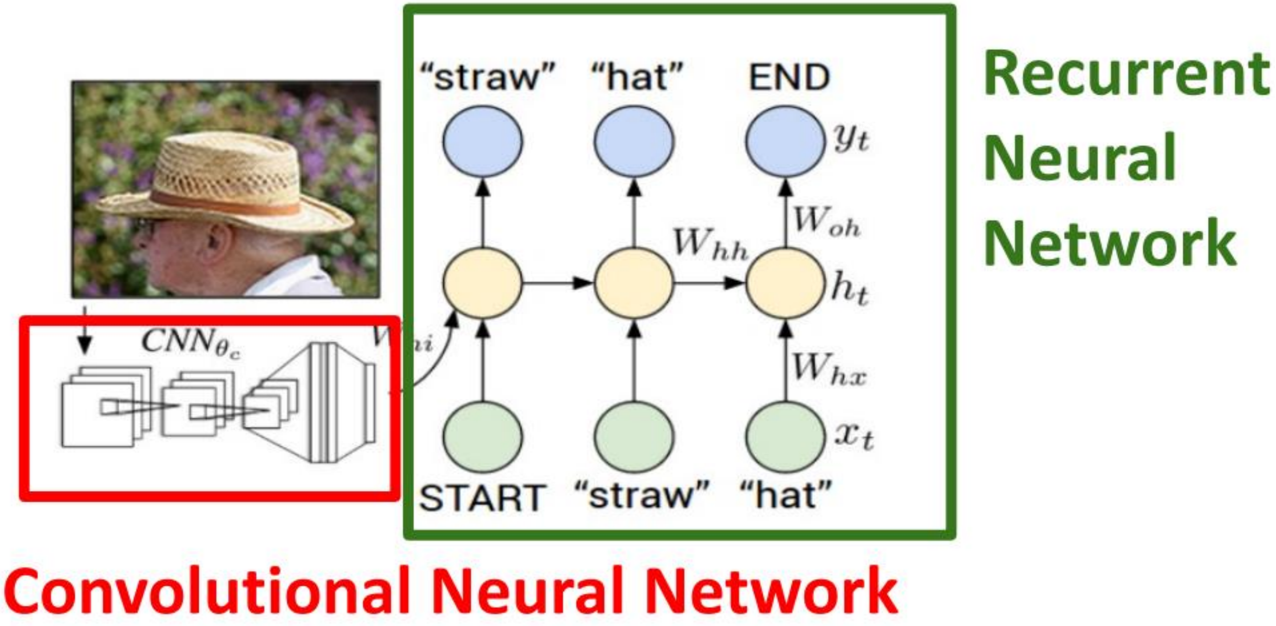
input vector at some time step

some function with parameters W



Michigan Online, Recurrent Networks, Justin Johnson

Examples: Image Captioning



Michigan Online, Recurrent Networks, Justin Johnson

Examples: Image Captioning

A cat sitting on a suitcase on the floor

A cat is sitting on a tree branch

A dog is running in the grass with a frisbee

A white teddy bear sitting in the grass

Two people walking on the beach with surfboards

A tennis player in action on the court

Two giraffes standing in a grassy field

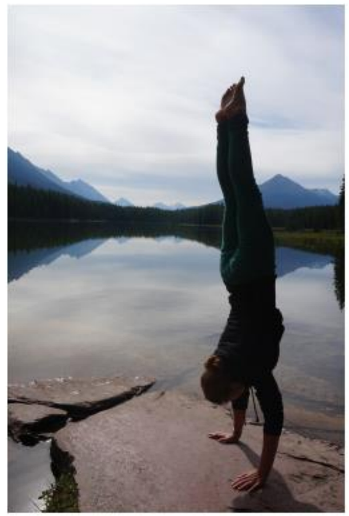
A man riding a dirt bike on a dirt track

Michigan Online, Recurrent Networks, Justin Johnson

Examples: Image Captioning

A woman is holding a cat in her hand

A person holding a computer mouse on a desk



A woman standing on a beach holding a surfboard

A bird is perched on a tree branch

A man in a baseball uniform throwing a ball

Michigan Online, Recurrent Networks, Justin Johnson

RNN: Sequence to Sequence

$$\boxed{h_t} = \boxed{f_W}(\boxed{h_{t-1}}, \boxed{x_t})$$

new state

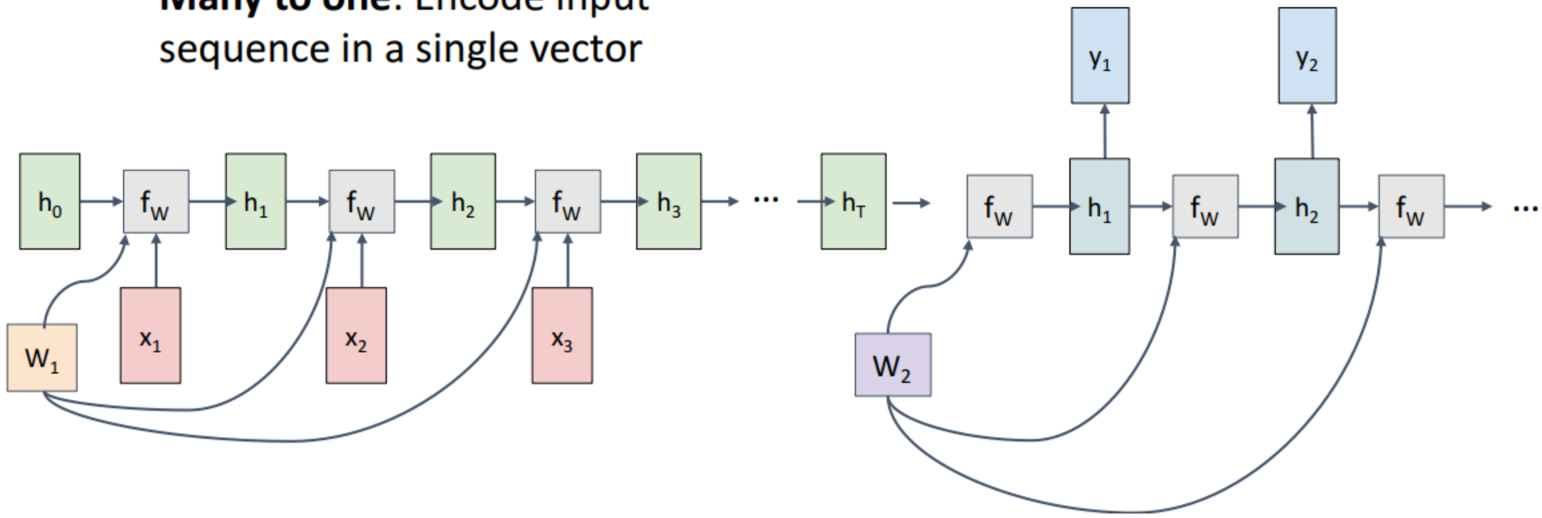
old state

input vector at some time step

some function with parameters W

One to many: Produce output sequence from single input vector

Many to one: Encode input sequence in a single vector



Michigan Online, Recurrent Networks, Justin Johnson

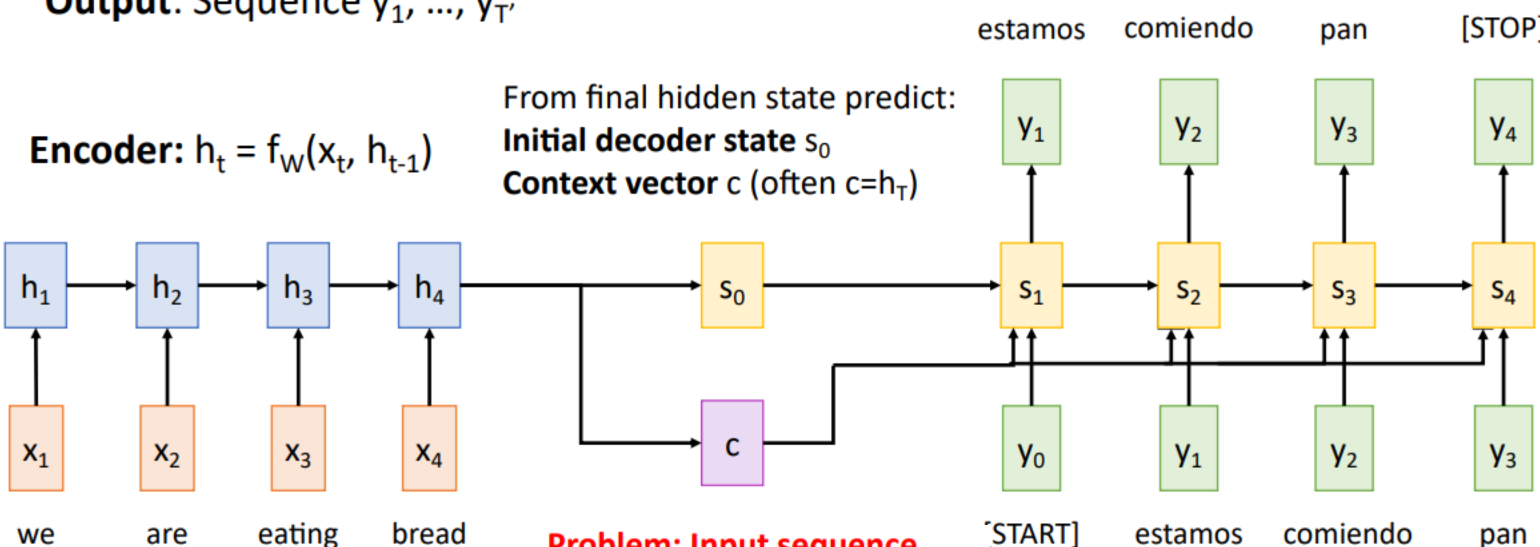
RNN: Sequence to Sequence

Input: Sequence $x_1, \ldots x_T$
Output: Sequence $y_1, \ldots, y_{T'}$

Decoder: $s_t = g_U(y_{t-1}, h_{t-1}, c)$

Encoder: $h_t = f_W(x_t, h_{t-1})$

From final hidden state predict:
Initial decoder state s_0
Context vector c (often $c=h_T$)



Problem: Input sequence bottlenecked through fixed-sized vector. What if T=1000?

Idea: use new context vector at each step of decoder!

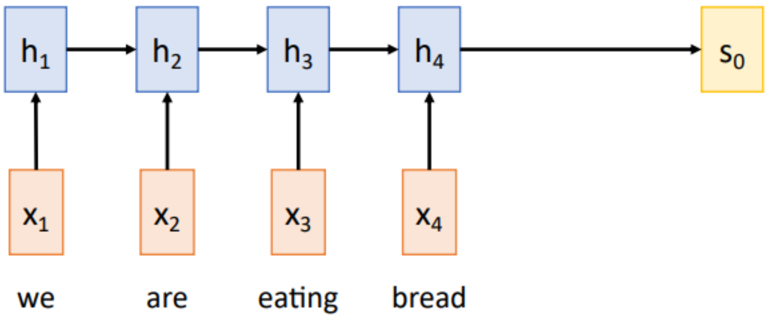
Michigan Online, Attention, Justin Johnson

RNN: Sequence to Sequence and Attention

Input: Sequence $x_1, \dots x_T$
Output: Sequence $y_1, \dots, y_{T'}$

Encoder: $h_t = f_W(x_t, h_{t-1})$

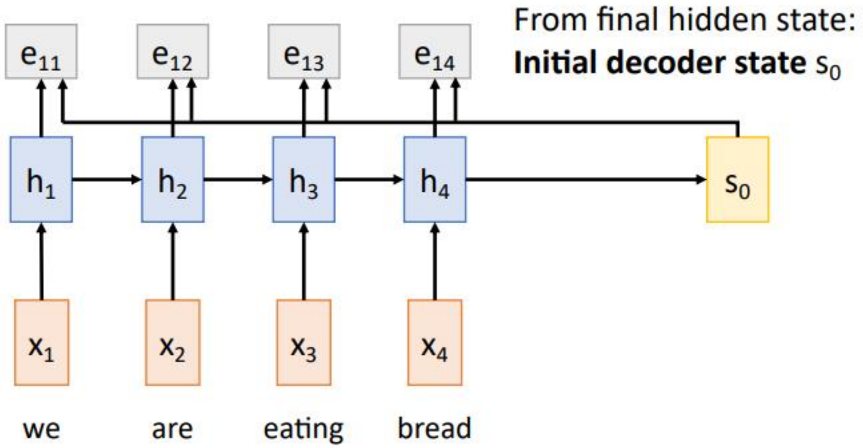
From final hidden state:
Initial decoder state s_0



Michigan Online, Attention, Justin Johnson

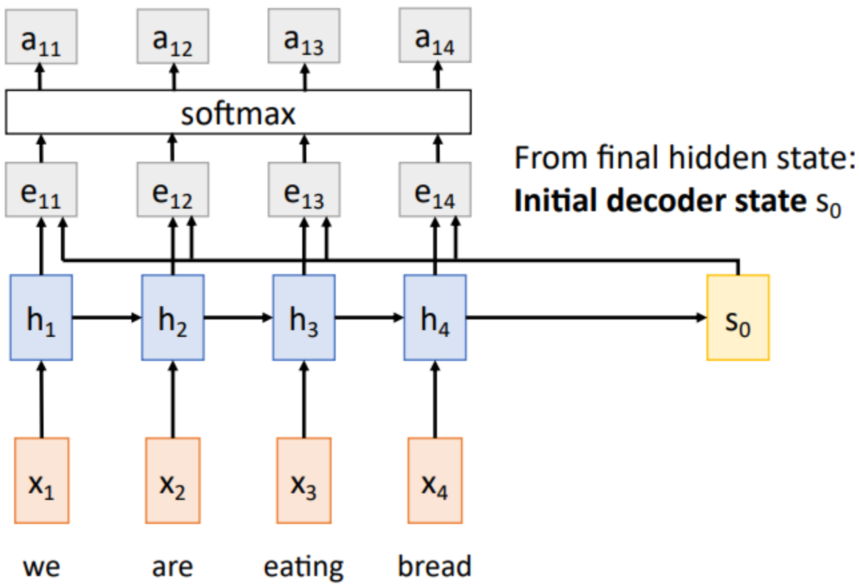
RNN: Sequence to Sequence and Attention

Compute (scalar) **alignment scores**
$e_{t,i} = f_{att}(s_{t-1}, h_i)$ (f_{att} is an MLP)



Michigan Online, Attention, Justin Johnson

RNN: Sequence to Sequence and Attention

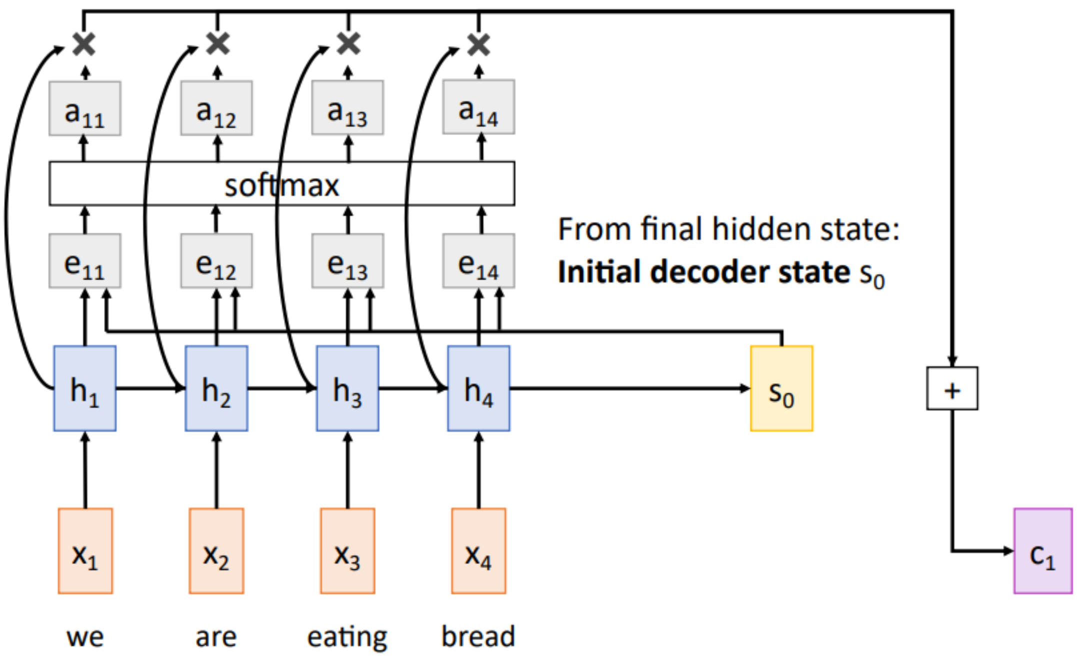


Compute (scalar) **alignment scores**
$e_{t,i} = f_{att}(s_{t-1}, h_i)$ (f_{att} is an MLP)

Normalize alignment scores
to get **attention weights**
$0 < a_{t,i} < 1$ $\sum_i a_{t,i} = 1$

Michigan Online, Attention, Justin Johnson

RNN: Sequence to Sequence and Attention



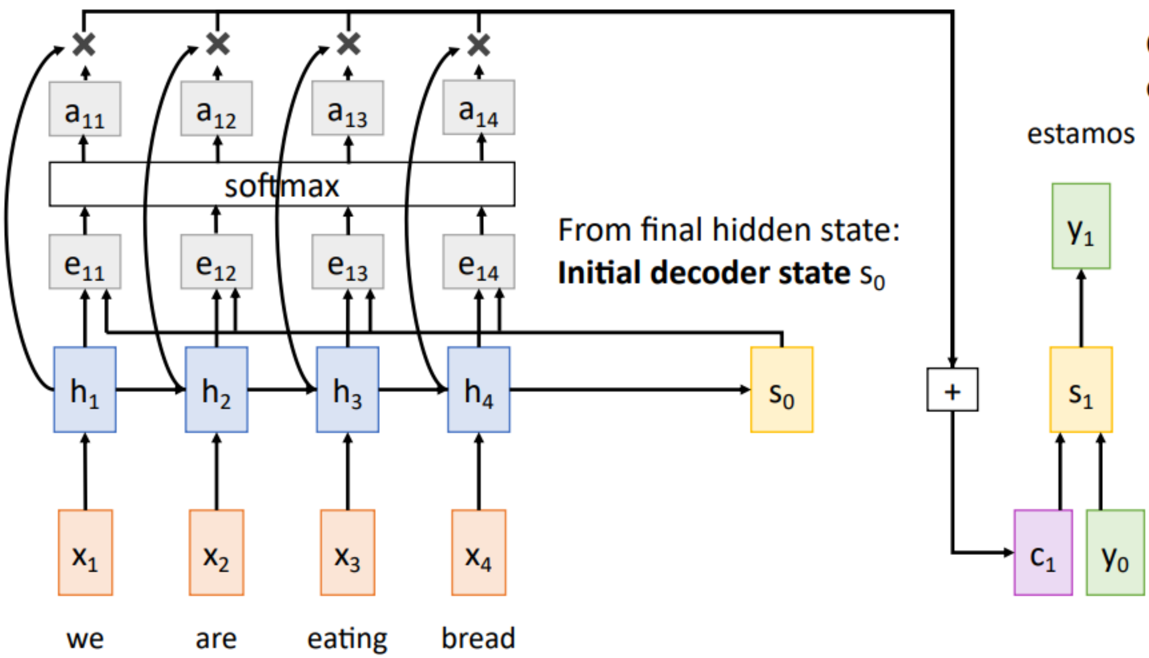
Compute (scalar) **alignment scores**
$e_{t,i} = f_{att}(s_{t-1}, h_i)$ (f_{att} is an MLP)

Normalize alignment scores to get **attention weights**
$0 < a_{t,i} < 1$ $\sum_i a_{t,i} = 1$

Compute context vector as linear combination of hidden states
$c_t = \sum_i a_{t,i} h_i$

Michigan Online, Attention, Justin Johnson

RNN: Sequence to Sequence and Attention



Compute (scalar) **alignment scores**
$e_{t,i} = f_{att}(s_{t-1}, h_i)$ (f_{att} is an MLP)

Normalize alignment scores to get **attention weights**
$0 < a_{t,i} < 1$ $\sum_i a_{t,i} = 1$

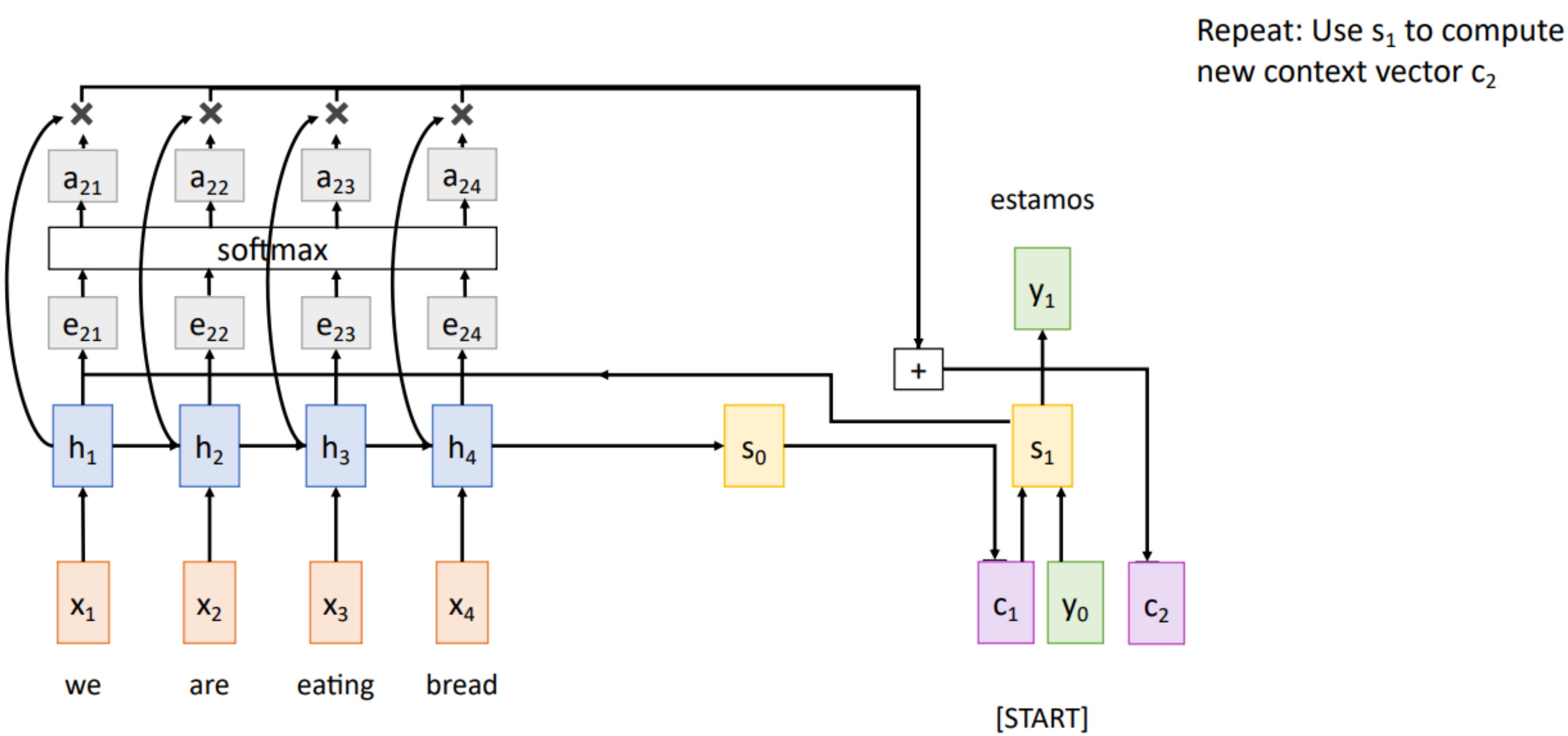
Compute context vector as linear combination of hidden states
$c_t = \sum_i a_{t,i} h_i$

Use context vector in decoder: $s_t = g_U(y_{t-1}, s_{t-1}, c_t)$

This is all differentiable! Do not supervise attention weights – backprop through everything

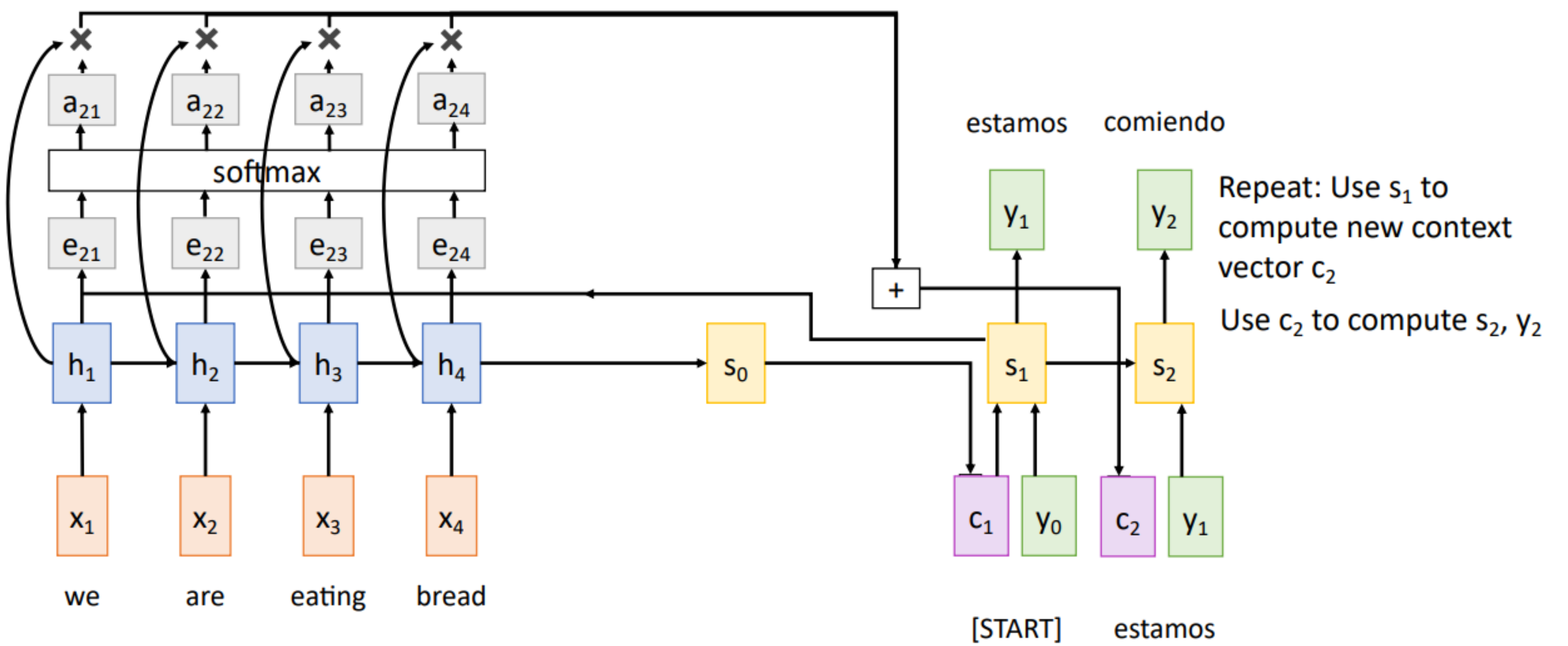
Michigan Online, Attention, Justin Johnson

RNN: Sequence to Sequence and Attention



Michigan Online, Attention, Justin Johnson

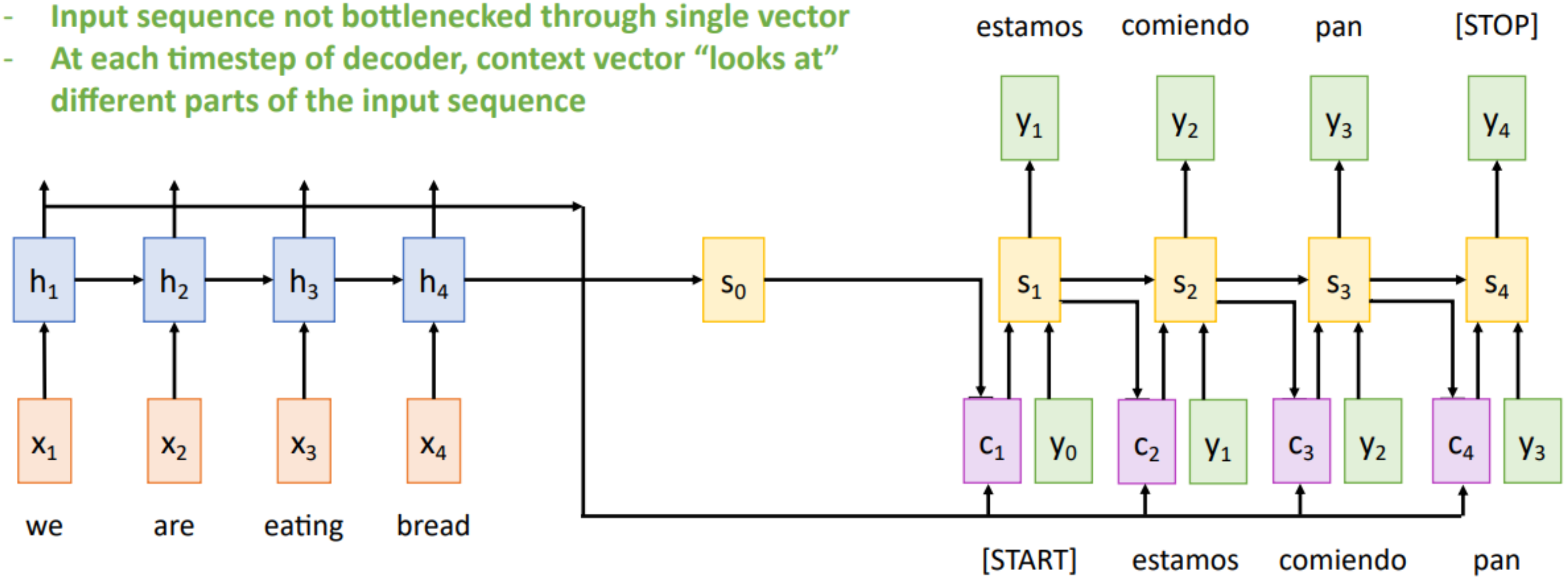
RNN: Sequence to Sequence and Attention



Michigan Online, Attention, Justin Johnson

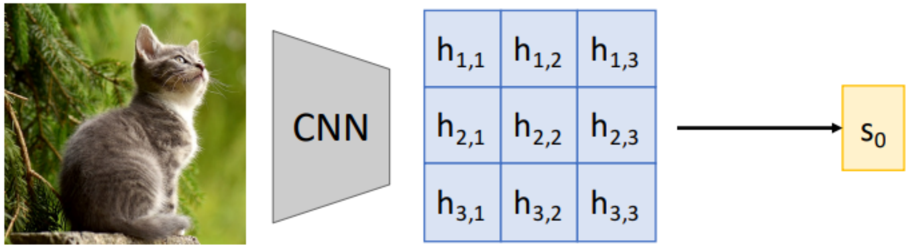
RNN: Sequence to Sequence and Attention

Use a different context vector in each timestep of decoder
- Input sequence not bottlenecked through single vector
- At each timestep of decoder, context vector "looks at" different parts of the input sequence



Michigan Online, Attention, Justin Johnson

Image Captioning with RNNs and Attention

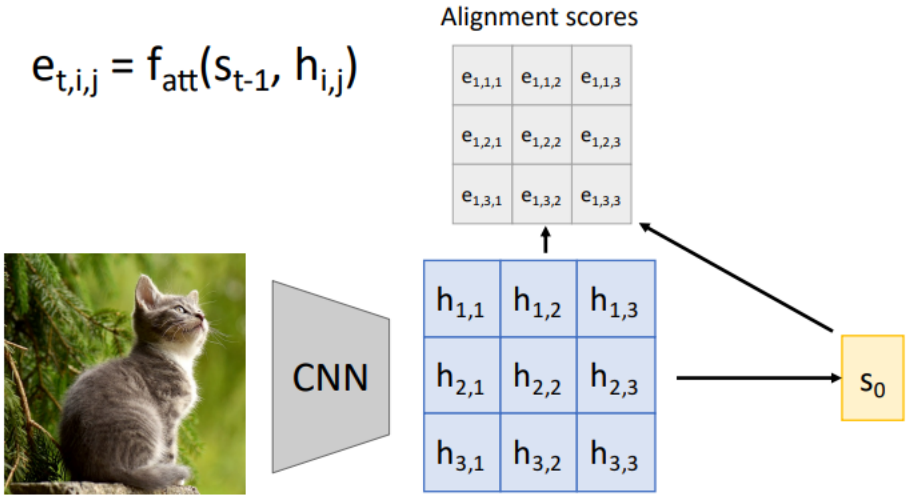


Use a CNN to compute a grid of features for an image

Michigan Online, Attention, Justin Johnson

Image Captioning with RNNs and Attention

$e_{t,i,j} = f_{att}(s_{t-1}, h_{i,j})$

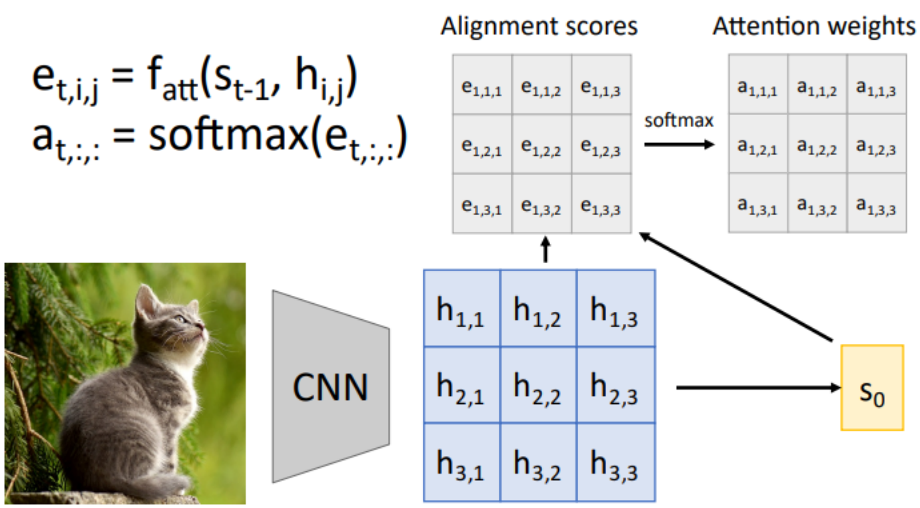


Use a CNN to compute a
grid of features for an image

Michigan Online, Attention, Justin Johnson

Image Captioning with RNNs and Attention

$e_{t,i,j} = f_{att}(s_{t-1}, h_{i,j})$
$a_{t,:,:} = \text{softmax}(e_{t,:,:})$

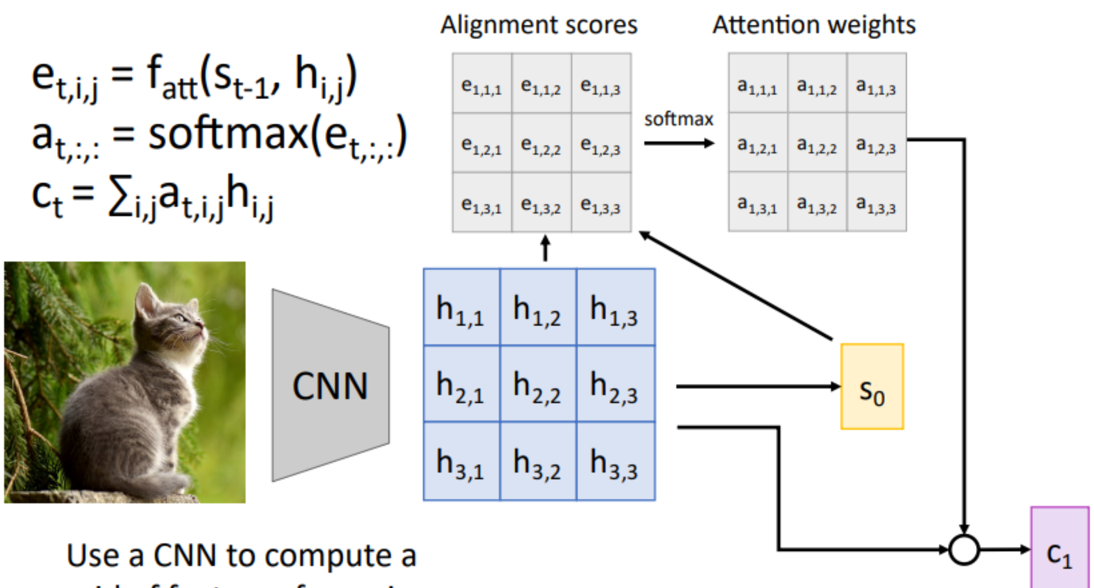


Use a CNN to compute a
grid of features for an image

Michigan Online, Attention, Justin Johnson

Image Captioning with RNNs and Attention

$e_{t,i,j} = f_{att}(s_{t-1}, h_{i,j})$
$a_{t,:,:} = \text{softmax}(e_{t,:,:})$
$c_t = \sum_{i,j} a_{t,i,j} h_{i,j}$

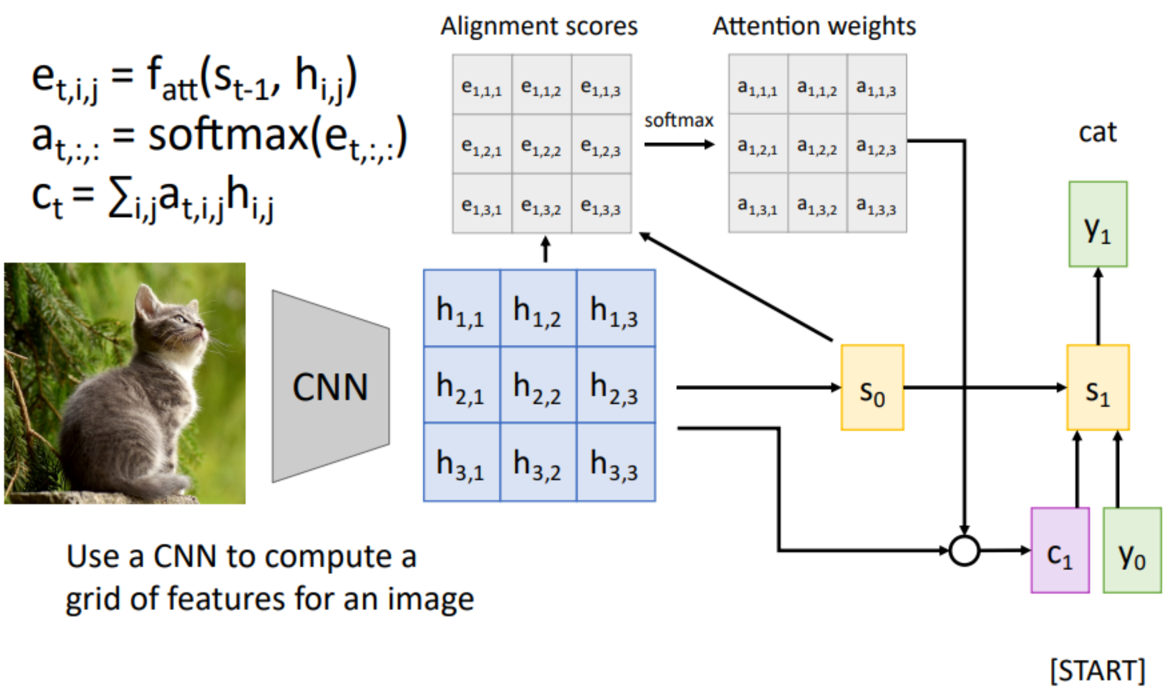


Use a CNN to compute a grid of features for an image

Michigan Online, Attention, Justin Johnson

Image Captioning with RNNs and Attention

$e_{t,i,j} = f_{att}(s_{t-1}, h_{i,j})$
$a_{t,:,:} = \text{softmax}(e_{t,:,:})$
$c_t = \sum_{i,j} a_{t,i,j} h_{i,j}$

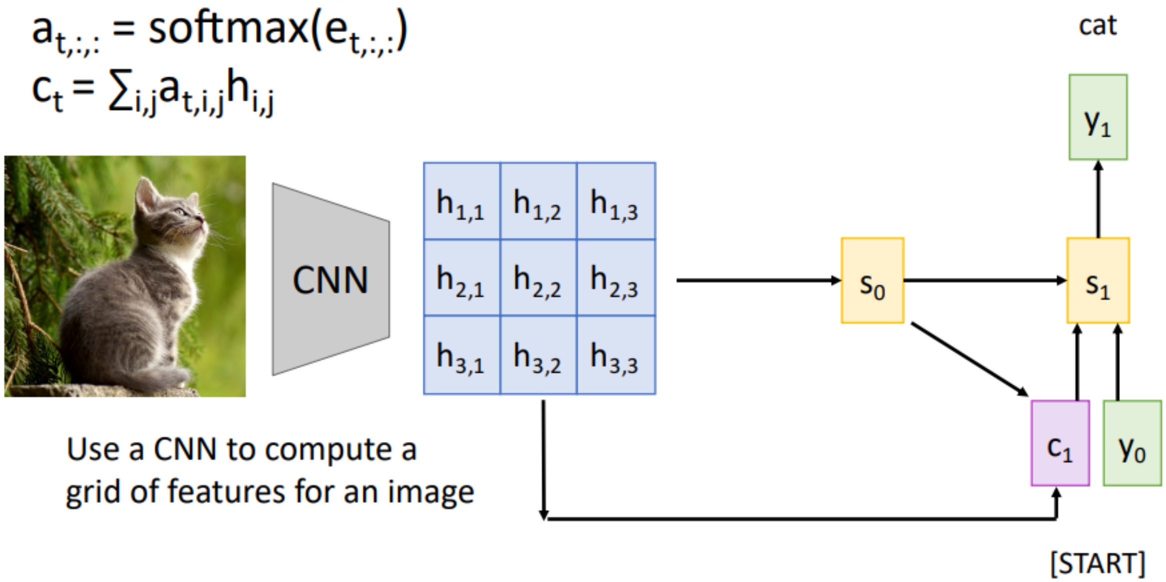


Use a CNN to compute a grid of features for an image

Michigan Online, Attention, Justin Johnson

Image Captioning with RNNs and Attention

$e_{t,i,j} = f_{att}(s_{t-1}, h_{i,j})$
$a_{t,:,:} = \text{softmax}(e_{t,:,:})$
$c_t = \sum_{i,j} a_{t,i,j} h_{i,j}$

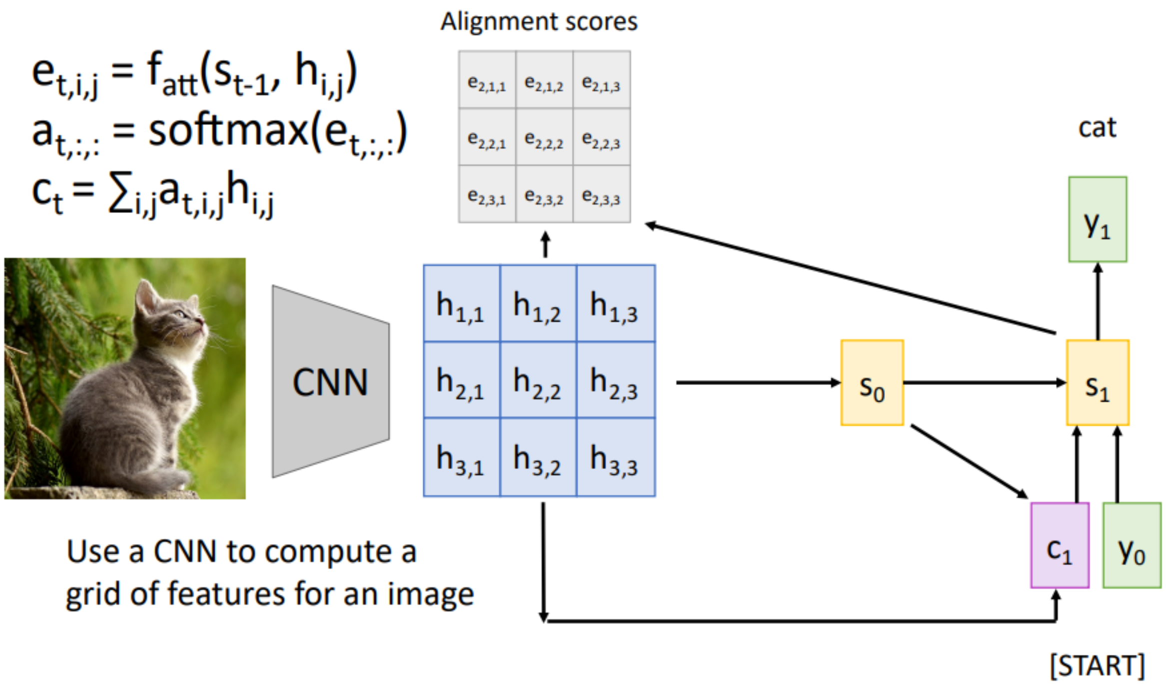


Use a CNN to compute a grid of features for an image

Michigan Online, Attention, Justin Johnson

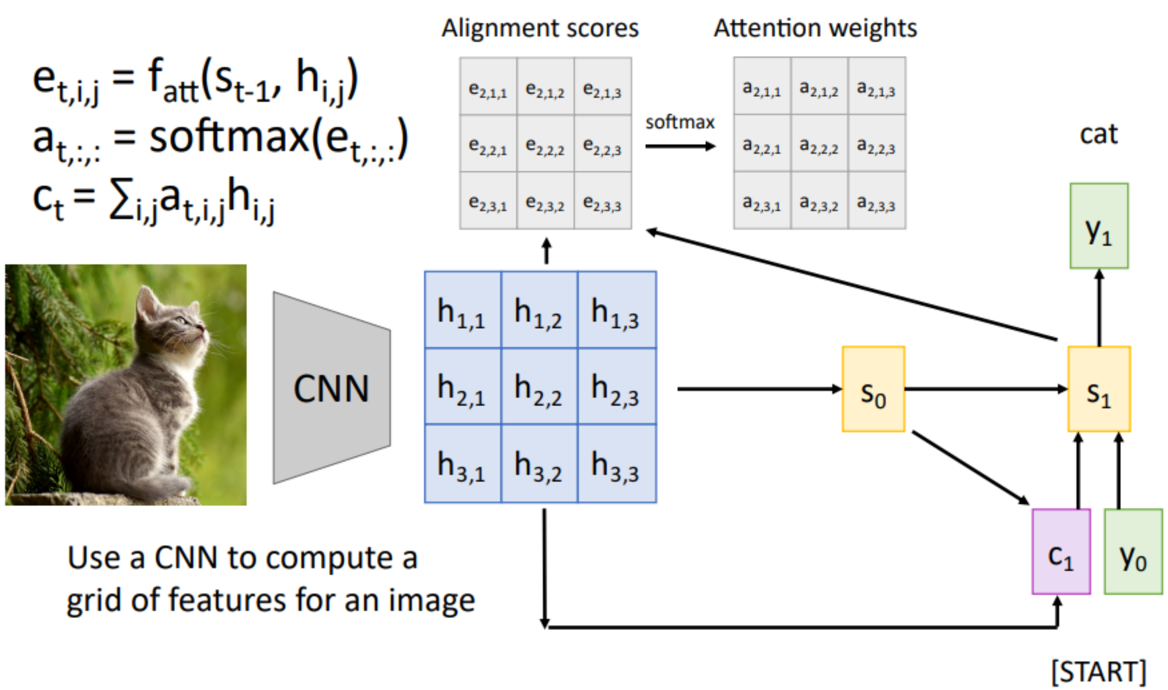
Image Captioning with RNNs and Attention

$$e_{t,i,j} = f_{att}(s_{t-1}, h_{i,j})$$
$$a_{t,:,:} = \mathrm{softmax}(e_{t,:,:})$$
$$c_t = \sum_{i,j} a_{t,i,j} h_{i,j}$$



Michigan Online, Attention, Justin Johnson

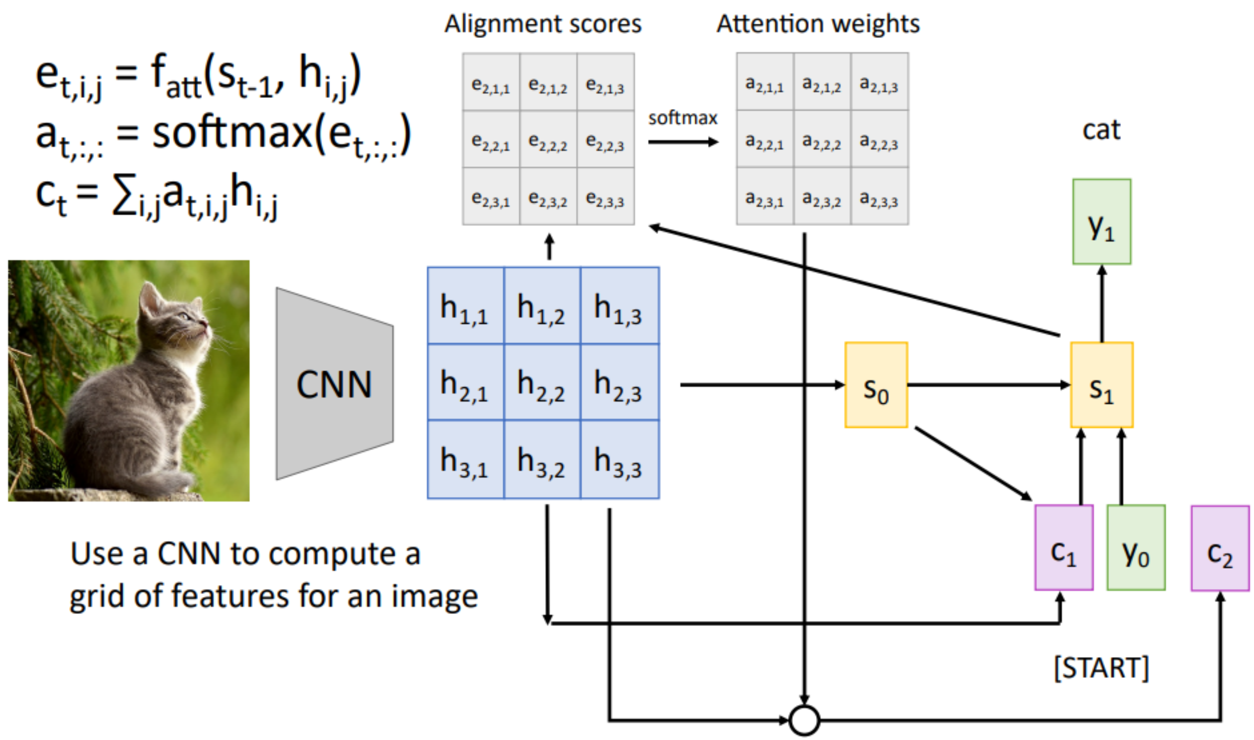
Image Captioning with RNNs and Attention



Michigan Online, Attention, Justin Johnson

Image Captioning with RNNs and Attention

$e_{t,i,j} = f_{att}(s_{t-1}, h_{i,j})$
$a_{t,:,:} = \text{softmax}(e_{t,:,:})$
$c_t = \sum_{i,j} a_{t,i,j} h_{i,j}$

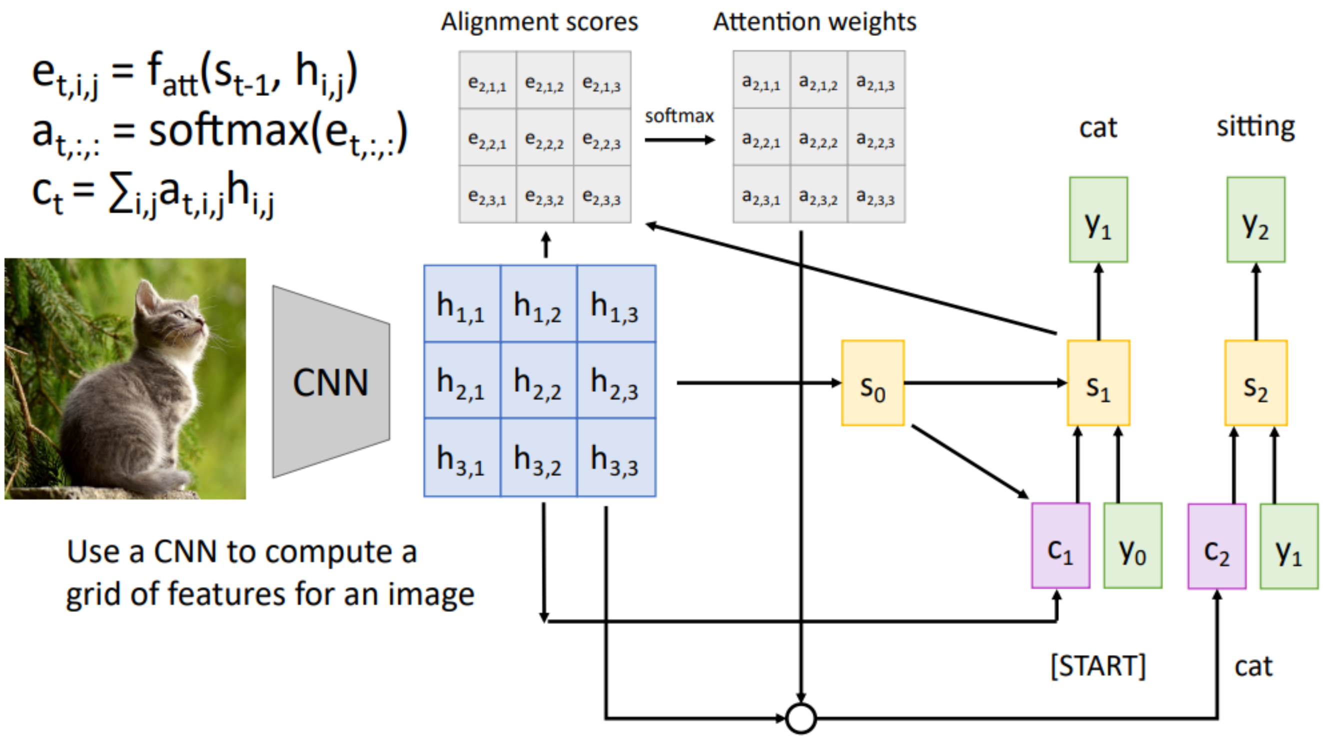


Use a CNN to compute a grid of features for an image

Michigan Online, Attention, Justin Johnson

Image Captioning with RNNs and Attention

$e_{t,i,j} = f_{att}(s_{t-1}, h_{i,j})$
$a_{t,:,:} = \text{softmax}(e_{t,:,:})$
$c_t = \sum_{i,j} a_{t,i,j} h_{i,j}$



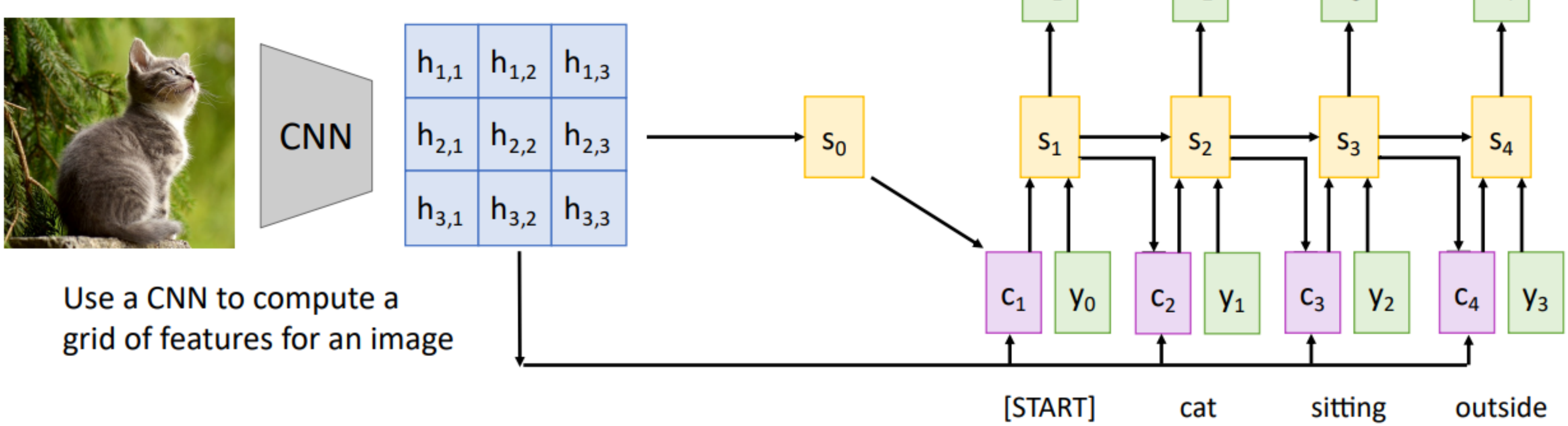
Use a CNN to compute a grid of features for an image

Michigan Online, Attention, Justin Johnson

Image Captioning with RNNs and Attention

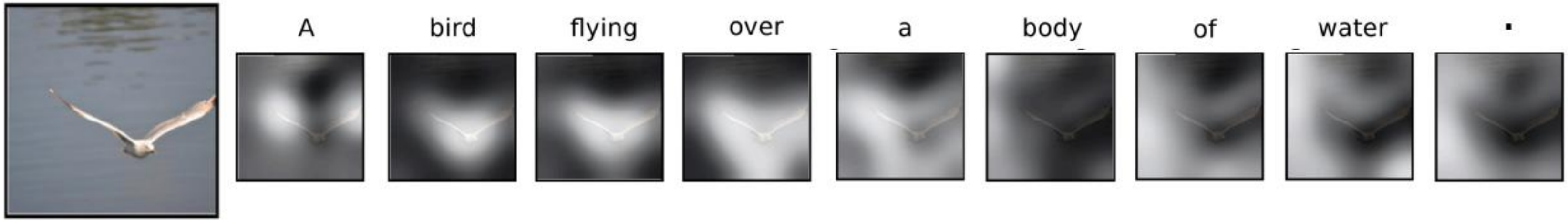
$e_{t,i,j} = f_{att}(s_{t-1}, h_{i,j})$
$a_{t,:,:} = \text{softmax}(e_{t,:,:})$
$c_t = \sum_{i,j} a_{t,i,j} h_{i,j}$

Each timestep of decoder uses a different context vector that looks at different parts of the input image



Michigan Online, Attention, Justin Johnson

Image Captioning with RNNs and Attention

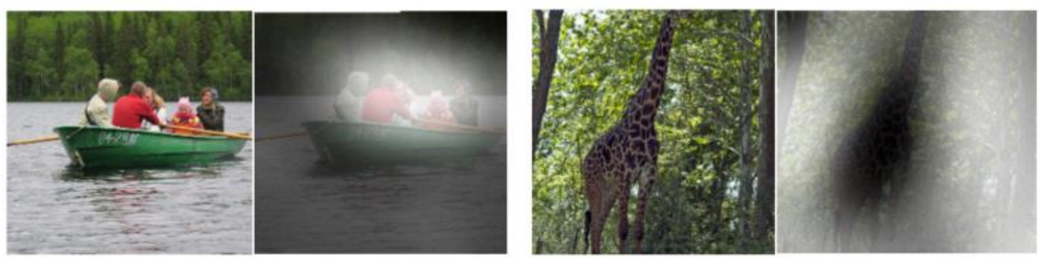


Michigan Online, Attention, Justin Johnson

Image Captioning with RNNs and Attention

A dog is standing on a hardwood floor.

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

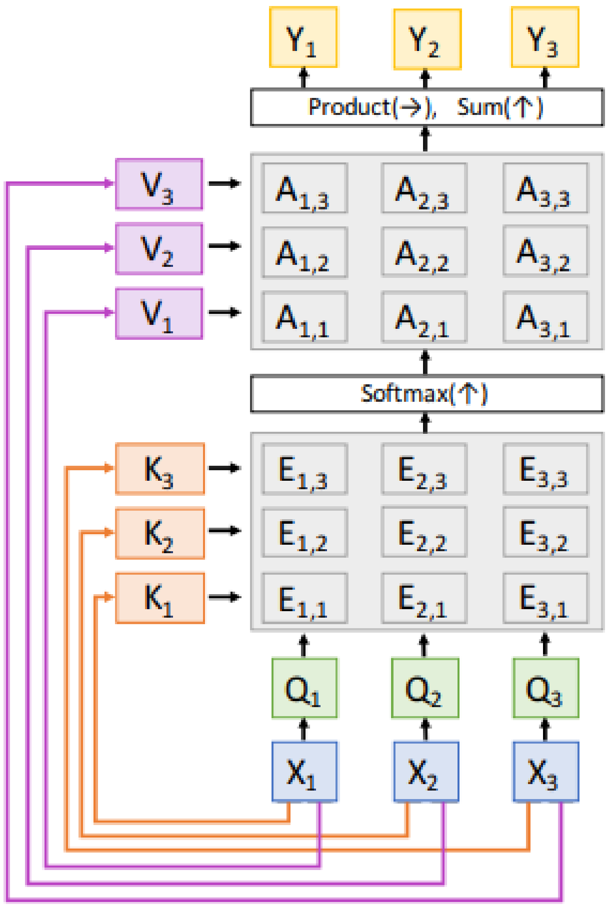


A group of people sitting on a boat in the water.

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

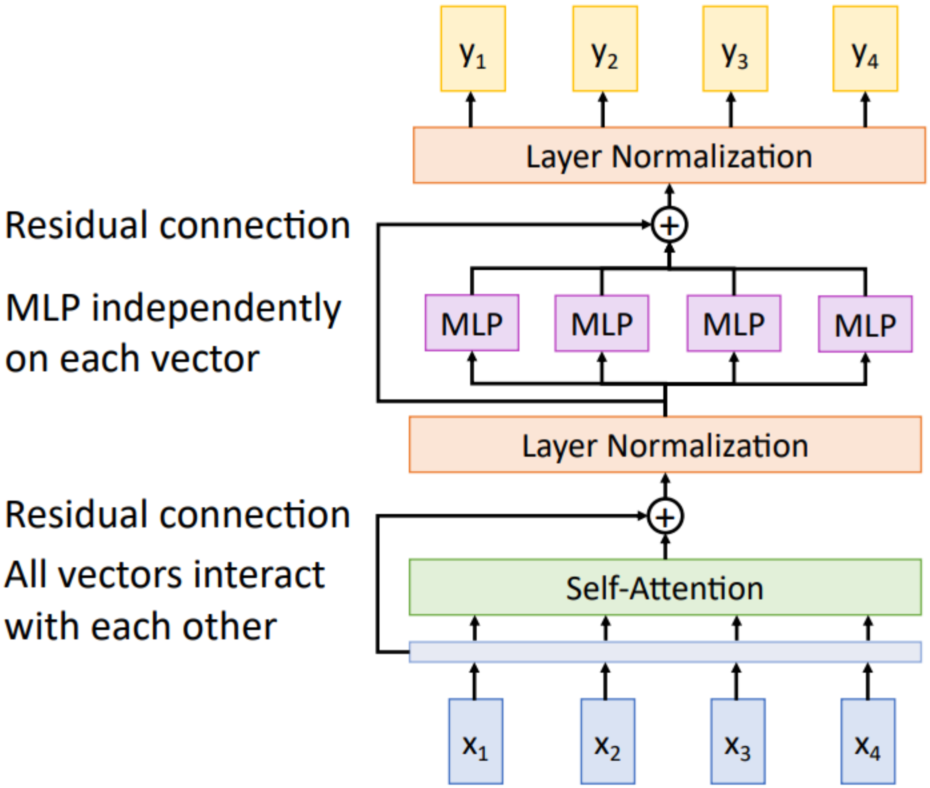
Michigan Online, Attention, Justin Johnson

Self-Attention



Michigan Online, Attention, Justin Johnson

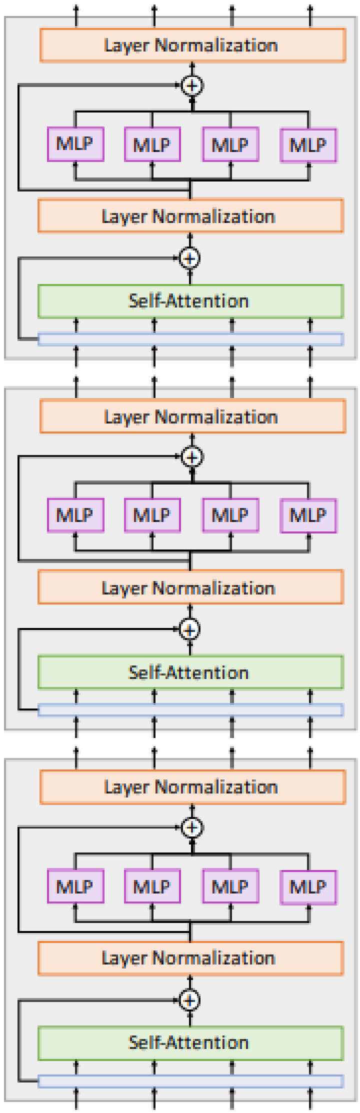
The Transformer



Michigan Online, Attention, Justin Johnson

The Transformer

A Transformer is a sequence of transformer blocks



Michigan Online, Attention, Justin Johnson

Scaling up Transformers

Model	Layers	Width	Heads	Params	Data	Training
Transformer-Base	12	512	8	65M		8x P100 (12 hours)
Transformer-Large	12	1024	16	213M		8x P100 (3.5 days)
BERT-Base	12	768	12	110M	13 GB	
BERT-Large	24	1024	16	340M	13 GB	
XLNet-Large	24	1024	16	~340M	126 GB	512x TPU-v3 (2.5 days)
RoBERTa	24	1024	16	355M	160 GB	1024x V100 GPU (1 day)
GPT-2	12	768	?	117M	40 GB	
GPT-2	24	1024	?	345M	40 GB	
GPT-2	36	1280	?	762M	40 GB	
GPT-2	48	1600	?	1.5B	40 GB	
Megatron-LM	40	1536	16	1.2B	174 GB	64x V100 GPU
Megatron-LM	54	1920	20	2.5B	174 GB	128x V100 GPU
Megatron-LM	64	2304	24	4.2B	174 GB	256x V100 GPU (10 days)
Megatron-LM	72	3072	32	8.3B	174 GB	512x V100 GPU (9 days)

Michigan Online, Attention, Justin Johnson

Some Links

- Recurrent Networks (Michigan Online)
  - https://www.youtube.com/watch?v=dUzLD91Sj-o
- Attention (Michigan Online)
  - https://www.youtube.com/watch?v=YAgjfMR9R_M