Lecture 9: Attention and Transformers

Dr. José Ramón Iglesias

DSP-ASIC BUILDER GROUP
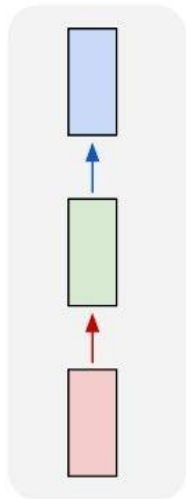
Director Semillero TRIAC

Ingenieria Electronica
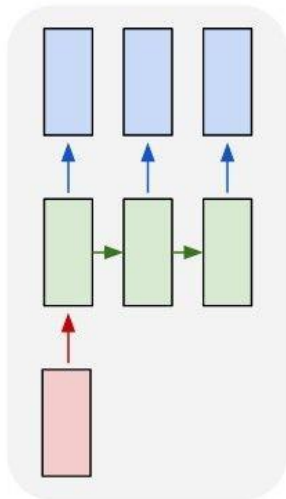
Universidad Popular del Cesar

Universidad Popular del Cesar
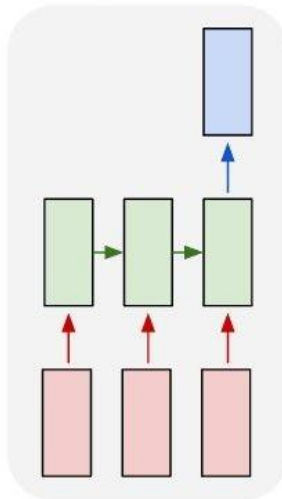
Last Time: Recurrent Neural Networks

one to one · one to many · many to one · many to many · many to many
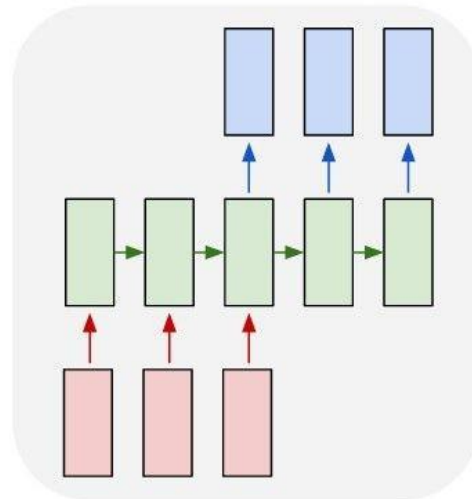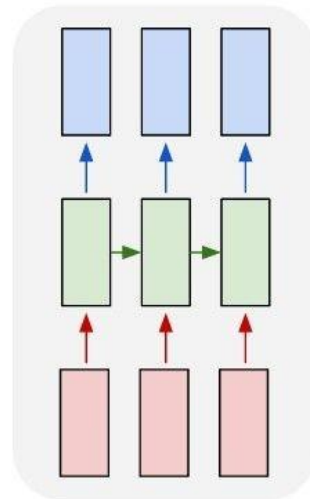
Last Time: Variable length computation graph with shared weights
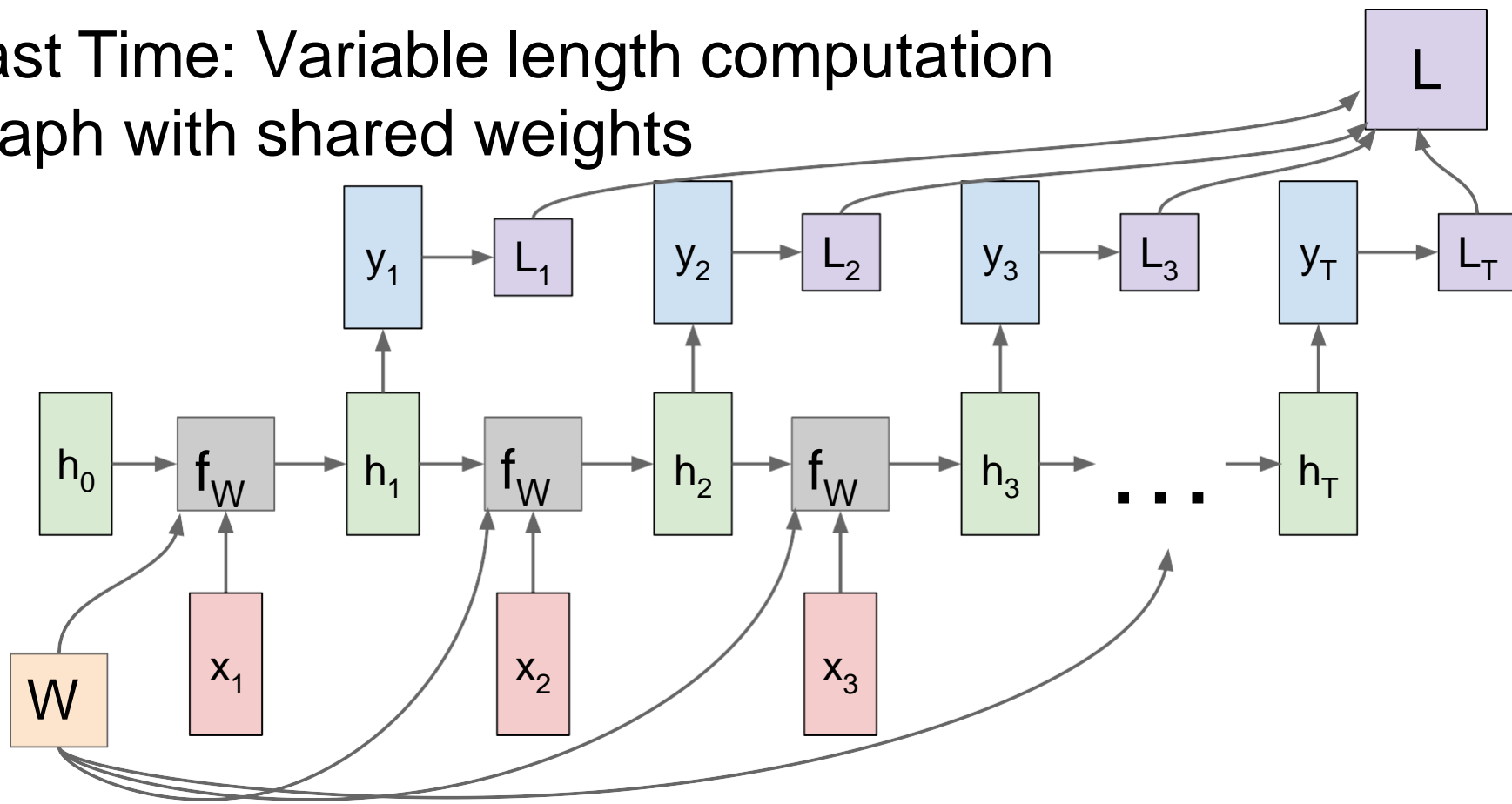
Universidad
Popular del Cesar

Sequence to Sequence with RNNs

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

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

Sutskever et al, "Sequence to sequence learning with neural networks", NeurIPS 2014

Attention and Transformers

4
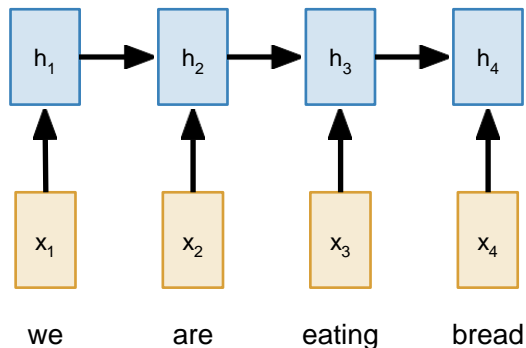
Sequence to Sequence with RNNs

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

From final hidden state predict:

Encoder: $h_t = f_W(x_t, h_{t-1})$ **Initial decoder state** s_0

Context vector c (often $c=h_T$)

Attention and Transformers

Universidad Popular del Cesar

Sequence to Sequence with RNNs

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

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

estamos

From final hidden state predict:
Encoder: $h_t = f_W(x_t, h_{t-1})$ **Initial decoder state** s_0
Context vector c (often c=h_T)

| h_1 | h_2 | h_3 | h_4 | | s_0 | | s_1 |

| x_1 | x_2 | x_3 | x_4 | | c | | y_0 |

we   are   eating   bread   [START]

Sutskever et al, "Sequence to sequence learning with neural networks", NeurIPS 2014

Attention and Transformers

Universidad
Popular del Cesar

Sequence to Sequence with RNNs

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

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

Encoder: $h_t = f_W(x_t, h_{t-1})$ From final hidden state predict:
Initial decoder state s_0
Context vector c (often $c = h_T$)

Sutskever et al, "Sequence to sequence learning with neural networks", NeurIPS 2014

Attention and Transformers

Universidad Popular del Cesar

Sequence to Sequence with RNNs

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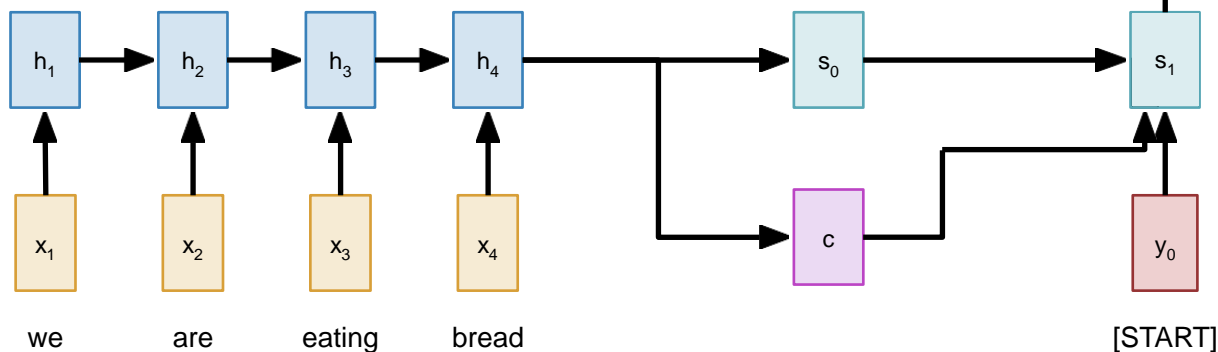**Output**: Sequence $y_1, \ldots, y_{T'}$

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

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

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

Sutskever et al, "Sequence to sequence learning with neural networks", NeurIPS 2014

Attention and Transformers

Universidad
Popular del Cesar

Sequence to Sequence with RNNs

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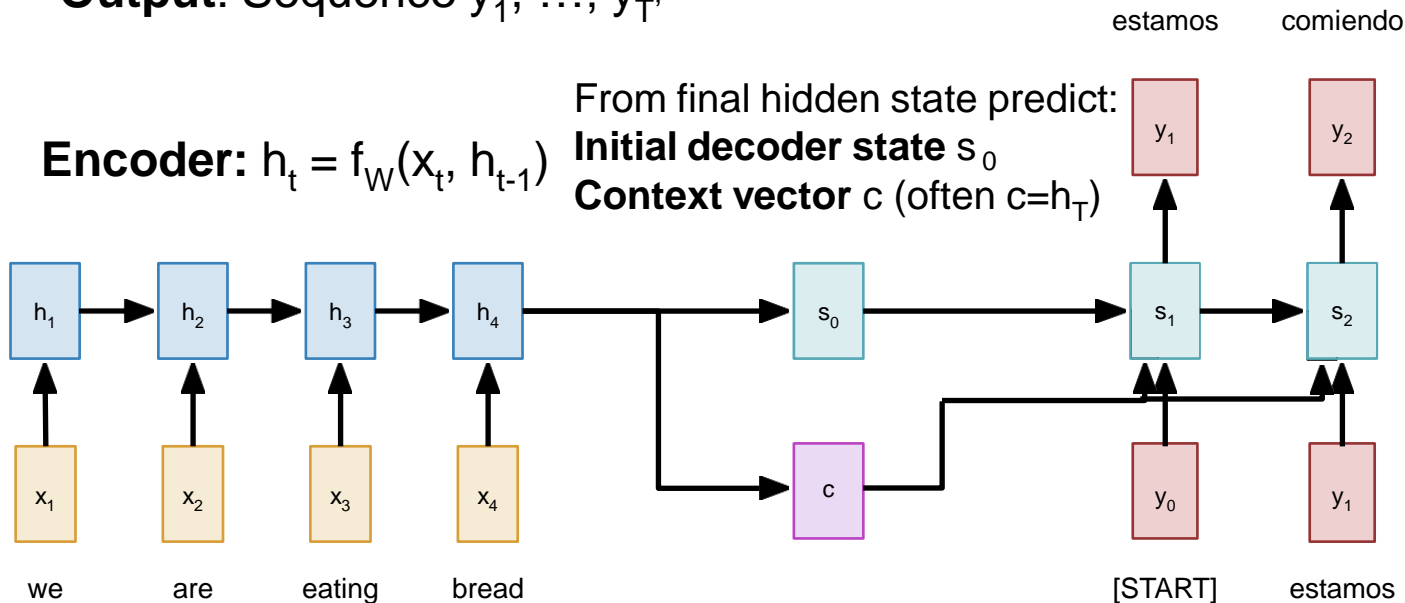**Output**: Sequence $y_1, \ldots, y_{T'}$

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

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?

Sutskever et al, "Sequence to sequence learning with neural networks", NeurIPS 2014

Attention and Transformers

10

Sequence to Sequence with RNNs

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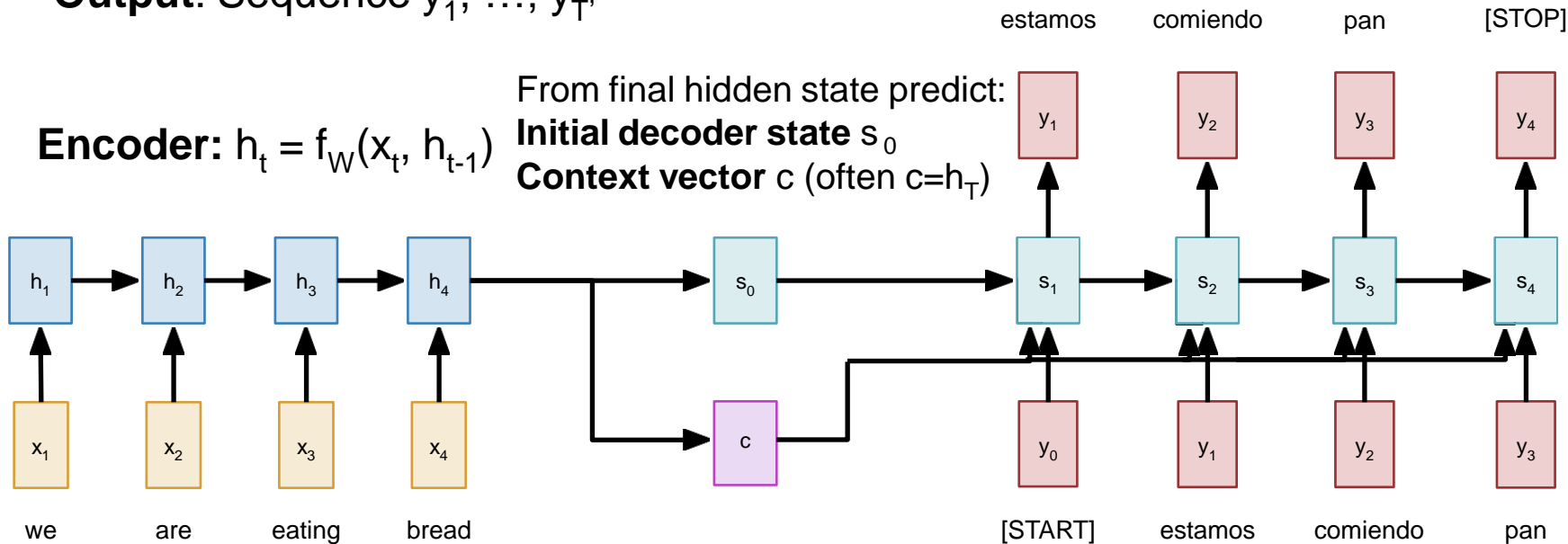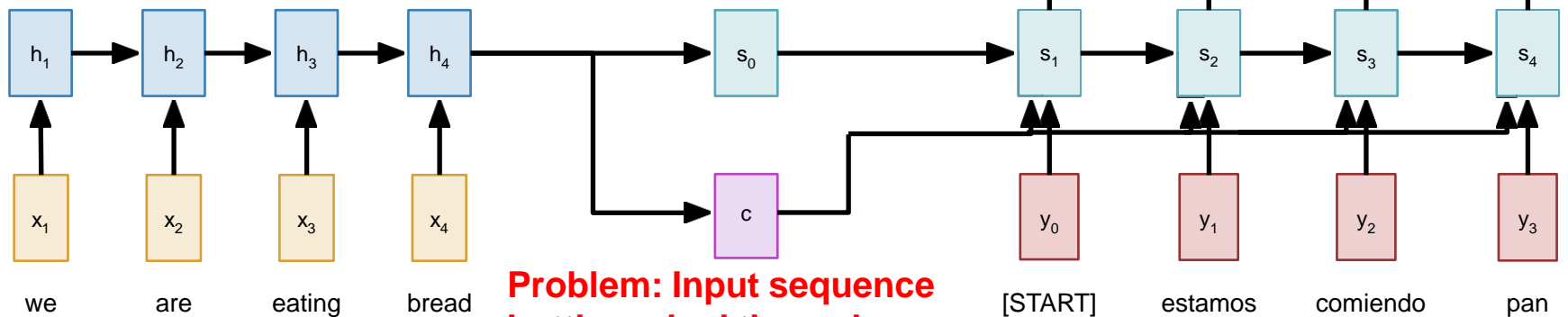**Output**: Sequence $y_1, \ldots, y_{T'}$

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

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!

Sutskever et al, "Sequence to sequence learning with neural networks", NeurIPS 2014

Attention and Transformers

10

Universidad Popular del Cesar

Sequence to Sequence with RNNs and Attention

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

Encoder: $h_t = f_W(x_t, h_{t-1})$ From final hidden state:
Initial decoder state s_0

Attention and Transformers

Universidad
Popular del Cesar

Sequence to Sequence with RNNs and Attention

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

From final hidden state:
Initial decoder state s_0

we   are   eating   bread

Bahdanau et al, "Neural machine translation by jointly learning to align and translate", ICLR 2015

Attention and Transformers

Universidad
Popular del Cesar

Sequence to Sequence with RNNs and Attention

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

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

From final hidden state: **Initial decoder state** s_0
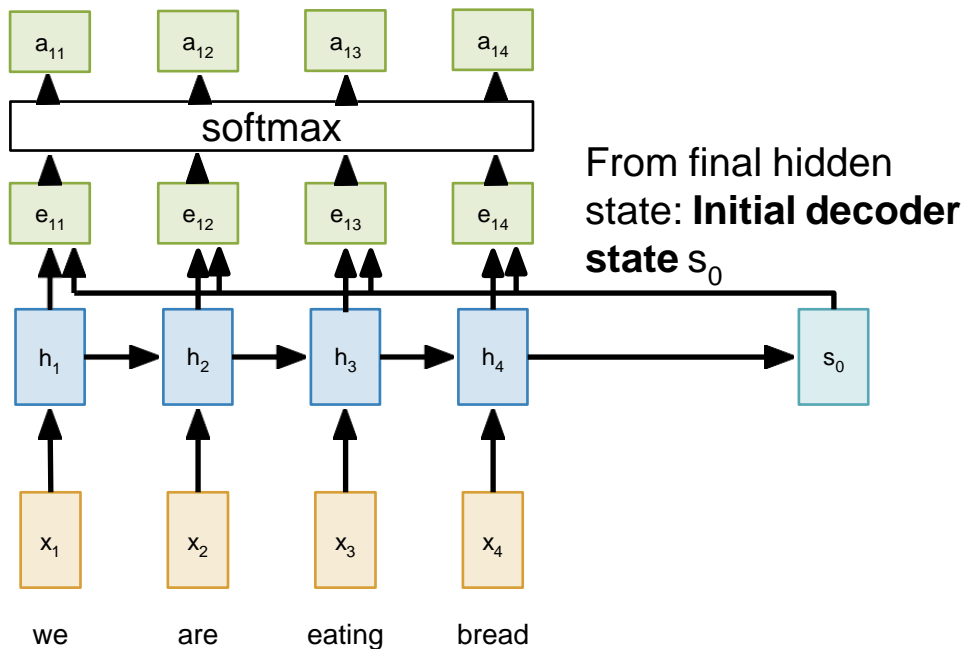
Attention and Transformers

Universidad
Popular del Cesar

Sequence to Sequence with RNNs and Attention

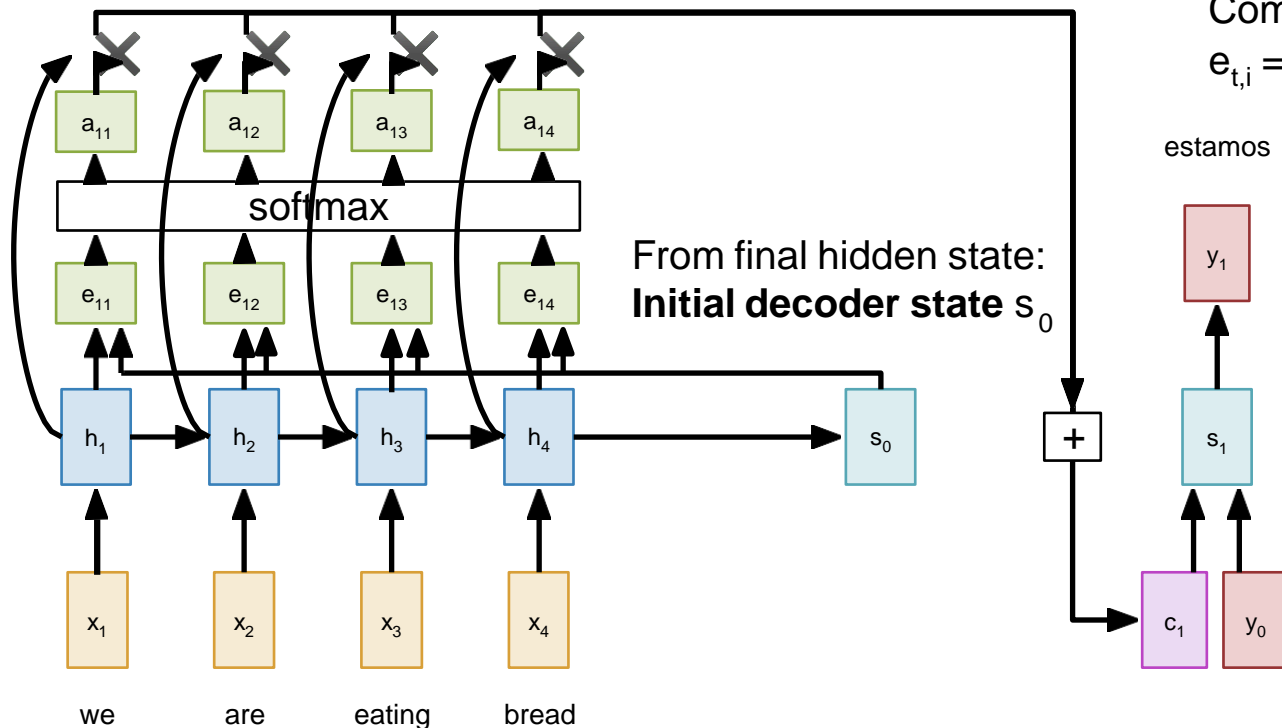Compute (scalar) **alignment scores**
$$e_{t,i} = f_{att}(s_{t-1}, h_i) \qquad (f_{att} \text{ is an MLP})$$

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

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

From final hidden state:
Initial decoder state s_0

we   are   eating   bread

estamos

[START]

Bahdanau et al, "Neural machine translation by jointly learning to align and translate", ICLR 2015

Attention and Transformers

Universidad
Popular del Cesar
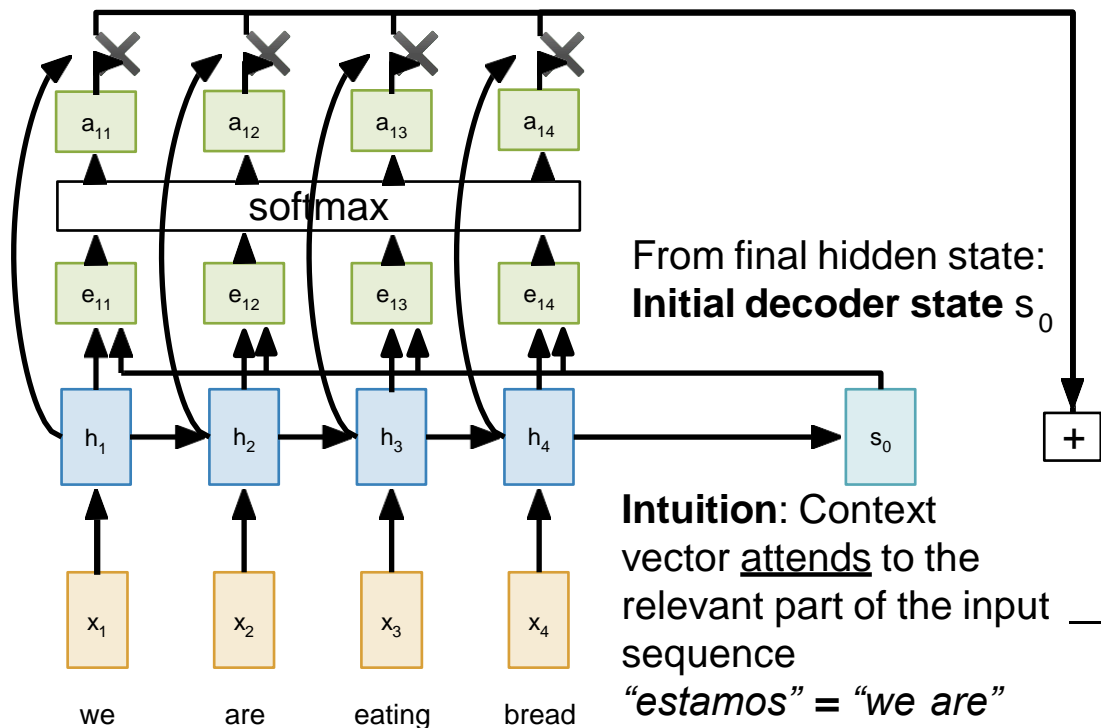
Sequence to Sequence with RNNs and Attention

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: $s_t = g_U(y_{t-1}, s_{t-1}, c_t)$

From final hidden state:
Initial decoder state s_0

Intuition: Context vector attends to the relevant part of the input sequence
"estamos" = *"we are"*
so maybe $a_{11} = a_{12} = 0.45$, $a_{13} = a_{14} = 0.05$

we   are   eating   bread

[START]

Bahdanau et al, "Neural machine translation by jointly learning to align and translate", ICLR 2015

Universidad Popular del Cesar

Sequence to Sequence with RNNs and Attention

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

estamos

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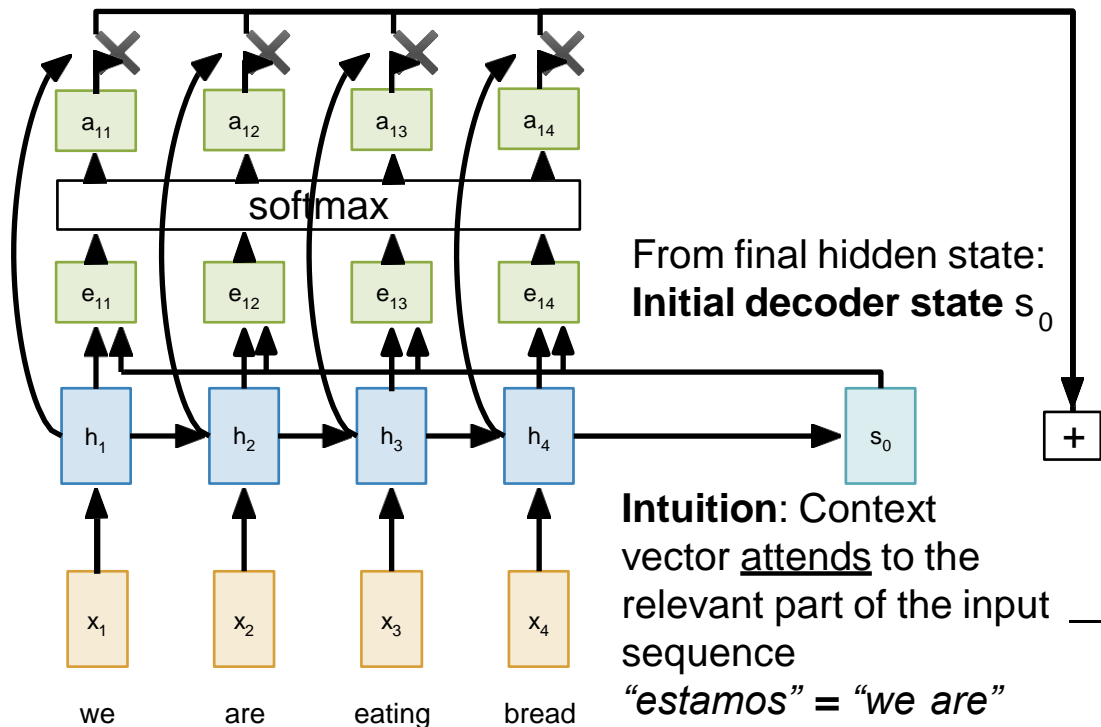$0 < a_{t,i} < 1$ $\sum_i a_{t,i} = 1$

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

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

From final hidden state:
Initial decoder state s_0

Intuition: Context vector <u>attends</u> to the relevant part of the input sequence
"estamos" = "we are"
so maybe $a_{11}=a_{12}=0.45$, $a_{13}=a_{14}=0.05$

This is all differentiable! No supervision on attention weights – backprop through everything

we   are   eating   bread

[START]

Bahdanau et al, "Neural machine translation by jointly learning to align and translate", ICLR 2015

Universidad Popular del Cesar

Sequence to Sequence with RNNs and Attention

Repeat: Use s_1 to compute new context vector c_2

Bahdanau et al, "Neural machine translation by jointly learning to align and translate", ICLR 2015

Attention and Transformers

Sequence to Sequence with RNNs and Attention

Repeat: Use s_1 to compute new context vector c_2

Use c_2 to compute s_2, y_2

Bahdanau et al, "Neural machine translation by jointly learning to align and translate", ICLR 2015

Attention and Transformers

Universidad Popular del Cesar

Sequence to Sequence with RNNs and Attention

Repeat: Use s_1 to compute new context vector c_2

a_{21} a_{22} a_{23} a_{24}

softmax

e_{21} e_{22} e_{23} e_{24}

h_1 h_2 h_3 h_4 s_0

x_1 x_2 x_3 x_4

we   are   eating   bread

estamos   comiendo

y_1 y_2

$+$

s_1 s_2

Use c_2 to compute s_2, y_2

c_1 y_0 c_2 y_1

[START]   estamos

Intuition: Context vector <u>attends</u> to the relevant part of the input sequence *"comiendo" = "eating"* so maybe $a_{21}=a_{24}=0.05$, $a_{22}=0.1$, $a_{23}=0.8$

Bahdanau et al, "Neural machine translation by jointly learning to align and translate", ICLR 2015

Attention and Transformers

19

Universidad Popular del Cesar

Sequence to Sequence with RNNs 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**
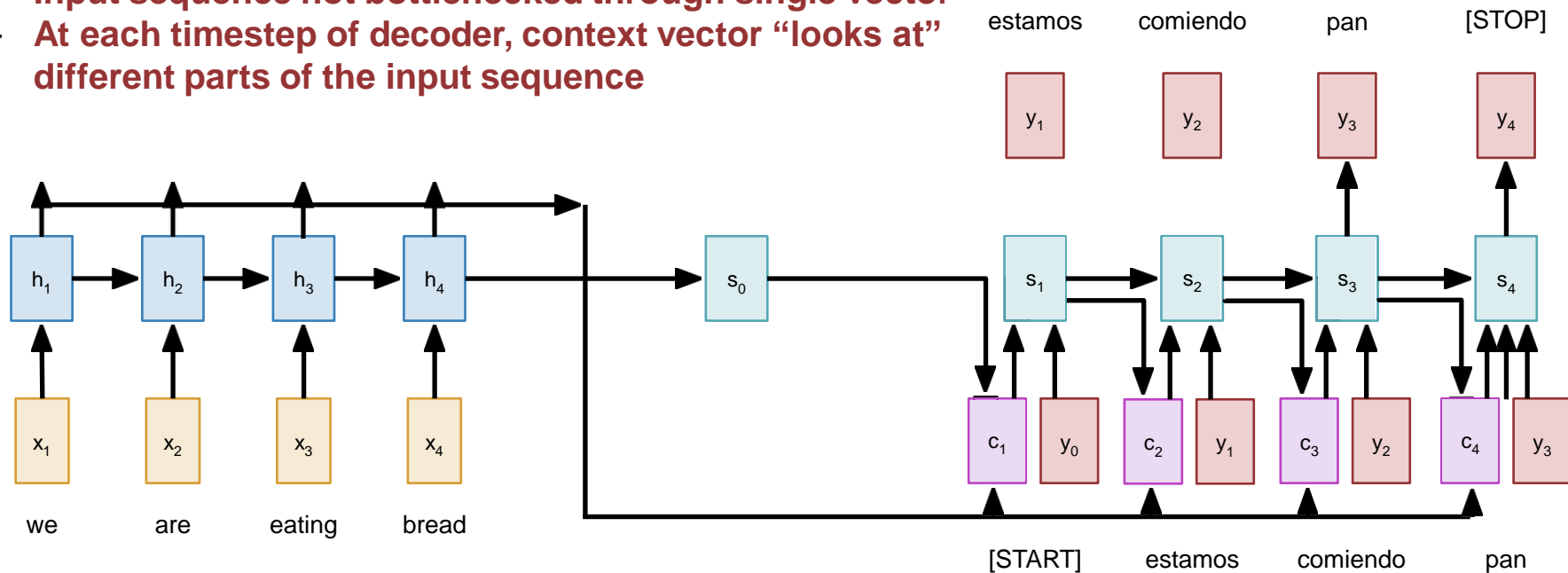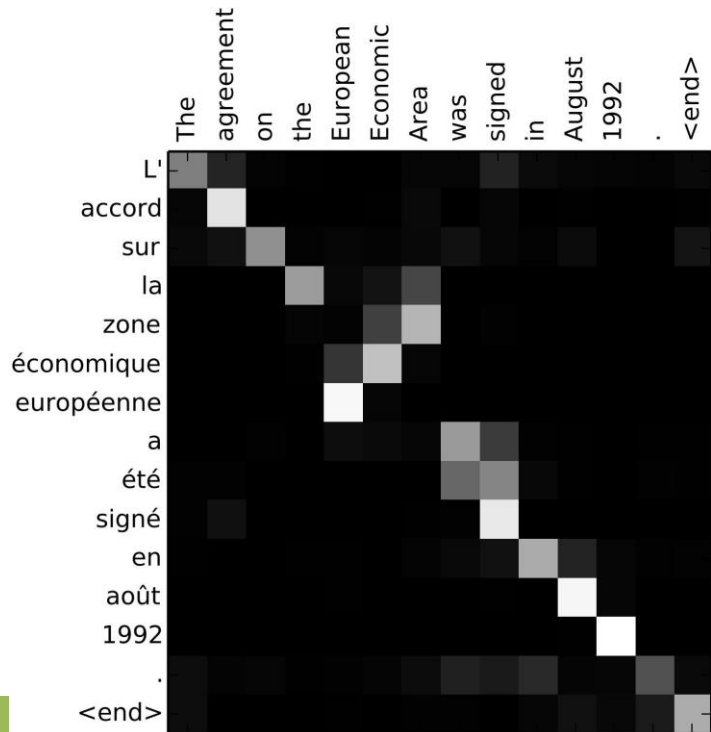
Attention and Transformers

Universidad Popular del Cesar

Sequence to Sequence with RNNs and Attention

Example: English to French translation

Input: "The agreement on the European Economic Area was signed in August 1992."

Output: "L'accord sur la zone économique européenne a été signé en août 1992."

Visualize attention weights $a_{t,i}$

Bahdanau et al, "Neural machine translation by jointly learning to align and translate", ICLR 2015

Attention and Transformers

Universidad Popular del Cesar

Sequence to Sequence with RNNs and Attention

Visualize attention weights $a_{t,i}$
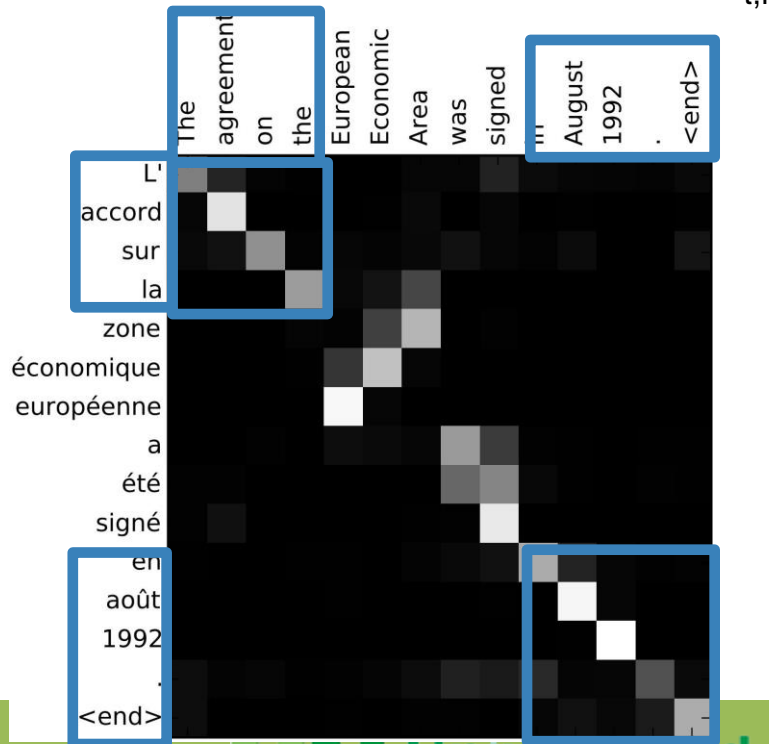
Example: English to French translation

Input: "**The agreement on the** European Economic Area was signed **in August 1992**."

Output: "**L'accord sur la** zone économique européenne a été signé **en août 1992**."

Diagonal attention means words correspond in order

Diagonal attention means words correspond in order

Bahdanau et al, "Neural machine translation by jointly learning to align and translate", ICLR 2015

Attention and Transformers

Universidad Popular del Cesar

Sequence to Sequence with RNNs and Attention

Example: English to French translation

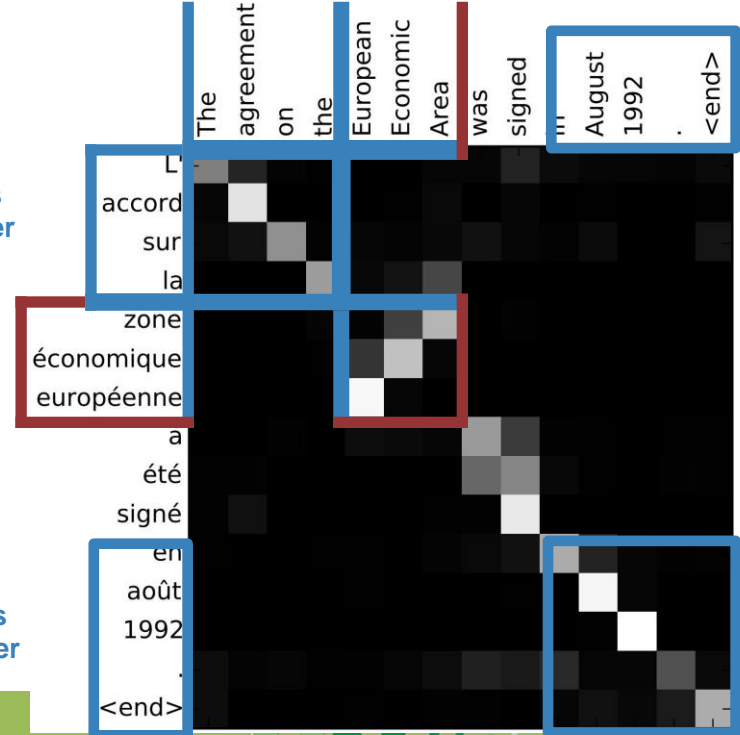**Input**: "**The agreement on the European Economic Area** was signed **in August 1992**."

Output: "**L'accord sur la zone économique européenne** a été signé **en août 1992**."

Visualize attention weights $a_{t,i}$

Diagonal attention means words correspond in order

Attention figures out different word orders

Diagonal attention means words correspond in order

Bahdanau et al, "Neural machine translation by jointly learning to align and translate", ICLR 2015

Attention and Transformers

Universidad Popular del Cesar

Sequence to Sequence with RNNs and Attention

The decoder doesn't use the fact that h_i form an ordered sequence – it just treats them as an unordered set $\{h_i\}$

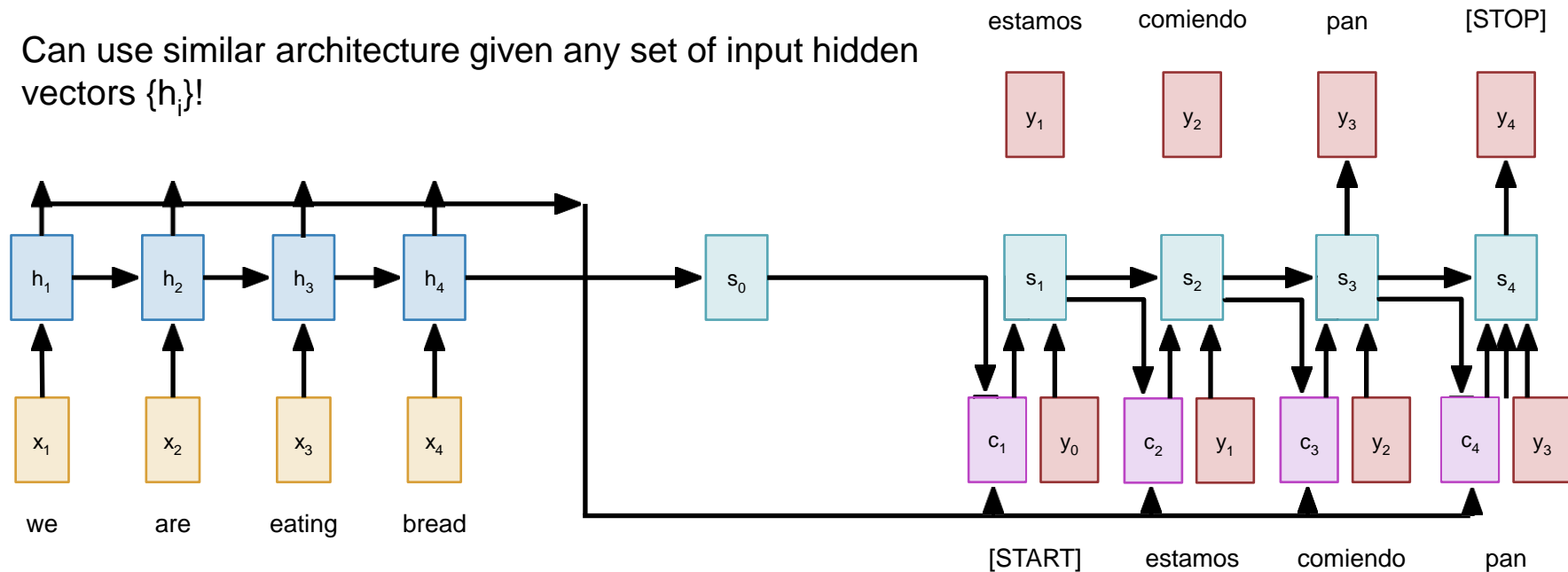Can use similar architecture given any set of input hidden vectors $\{h_i\}$!
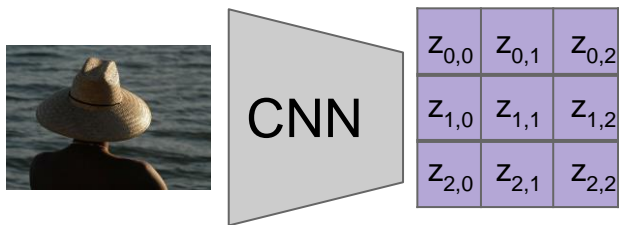
Attention and Transformers

24

Universidad Popular del Cesar

Image Captioning using spatial features

Input: Image **I**
Output: Sequence $\mathbf{y} = y_1, y_2,..., y_T$

CNN

$z_{0,0}$ $z_{0,1}$ $z_{0,2}$
$z_{1,0}$ $z_{1,1}$ $z_{1,2}$
$z_{2,0}$ $z_{2,1}$ $z_{2,2}$

Extract spatial features from a pretrained CNN

Features: H x W x D

Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

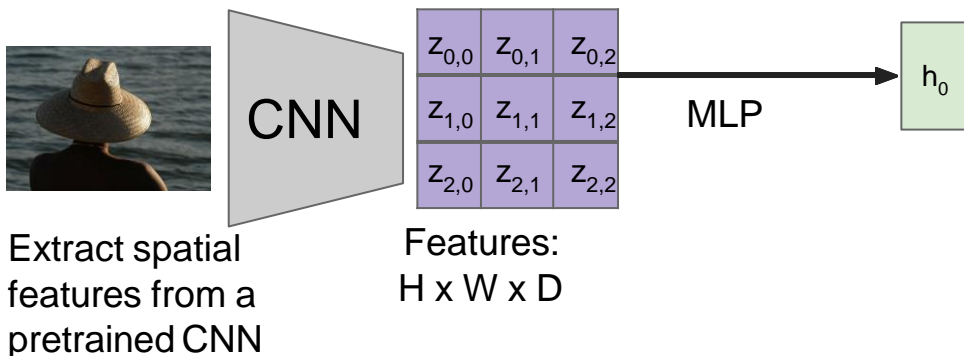Attention and Transformers

Universidad Popular del Cesar

Image Captioning using spatial features

Input: Image **I**
Output: Sequence $\mathbf{y} = y_1, y_2,..., y_T$

Encoder: $h_0 = f_\mathbf{W}(\mathbf{z})$
where **z** is spatial CNN features
$f_\mathbf{W}(.)$ is an MLP

Extract spatial
features from a
pretrained CNN

Features:
H x W x D

Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

Attention and Transformers

Image Captioning using spatial features

Input: Image **I**
Output: Sequence $\mathbf{y} = y_1, y_2,..., y_T$

Decoder: $y_t = g_V(y_{t-1}, h_{t-1}, c)$
where context vector c is often $c = h_0$

Encoder: $h_0 = f_W(\mathbf{z})$
where $\mathbf{z}$ is spatial CNN features
$f_W(.)$ is an MLP

person

y_1

CNN

$z_{0,0}$	$z_{0,1}$	$z_{0,2}$
$z_{1,0}$	$z_{1,1}$	$z_{1,2}$
$z_{2,0}$	$z_{2,1}$	$z_{2,2}$

MLP

h_0

h_1

c

y_0

Extract spatial
features from a
pretrained CNN

Features:
H x W x D

[START]

Attention and Transformers

Universidad
Popular del Cesar

Image Captioning using spatial features

Input: Image **I**
Output: Sequence $\mathbf{y} = y_1, y_2, ..., y_T$

Decoder: $y_t = g_\mathbf{V}(y_{t-1}, h_{t-1}, c)$
where context vector c is often $c = h_0$

Encoder: $h_0 = f_\mathbf{W}(\mathbf{z})$
where **z** is spatial CNN features
$f_\mathbf{W}(.)$ is an MLP

person   wearing

CNN

MLP

Extract spatial features from a pretrained CNN

Features:
H x W x D

$z_{0,0}$	$z_{0,1}$	$z_{0,2}$
$z_{1,0}$	$z_{1,1}$	$z_{1,2}$
$z_{2,0}$	$z_{2,1}$	$z_{2,2}$

h_0 h_1 h_2

y_1 y_2

c

y_0 y_1

Image Captioning using spatial features

Input: Image **I**
Output: Sequence $\mathbf{y} = y_1, y_2,..., y_T$

Decoder: $y_t = g_V(y_{t-1}, h_{t-1}, c)$
where context vector c is often $c = h_0$

Encoder: $h_0 = f_W(\mathbf{z})$
where $\mathbf{z}$ is spatial CNN features
$f_W(.)$ is an MLP

person   wearing   hat

y_1 y_2 y_3

CNN

$z_{0,0}$	$z_{0,1}$	$z_{0,2}$
$z_{1,0}$	$z_{1,1}$	$z_{1,2}$
$z_{2,0}$	$z_{2,1}$	$z_{2,2}$

MLP

h_0 h_1 h_2 h_3

c y_0 y_1 y_2

Extract spatial
features from a
pretrained CNN

Features:
H x W x D

Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

[START]   person   wearing

Attention and Transformers

Universidad Popular del Cesar

29

Image Captioning using spatial features

Input: Image **I**
Output: Sequence $\mathbf{y}$ = y_1, y_2,..., y_T

Decoder: $y_t = g_{\mathbf{V}}(y_{t-1}, h_{t-1}, c)$
where context vector c is often c = h_0

Encoder: $h_0 = f_{\mathbf{W}}(\mathbf{z})$
where **z** is spatial CNN features
$f_{\mathbf{W}}(\textbf{.})$ is an MLP

person   wearing   hat   [END]

y_1 y_2 y_3 y_4

CNN

$z_{0,0}$ $z_{0,1}$ $z_{0,2}$
$z_{1,0}$ $z_{1,1}$ $z_{1,2}$
$z_{2,0}$ $z_{2,1}$ $z_{2,2}$

MLP

h_0 h_1 h_2 h_3 h_4

c y_0 y_1 y_2 y_3

Extract spatial
features from a
pretrained CNN

Features:
H x W x D

Image Captioning using spatial features

Problem: Input is "bottlenecked" through c

- Model needs to encode everything it
  wants to say within c

**This is a problem if we want to generate
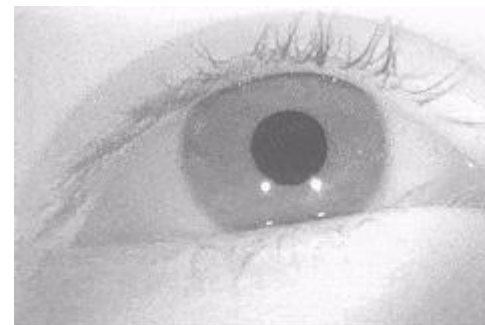really long descriptions? 100s of words long**

Extract spatial
features from a
pretrained CNN

Features:
H x W x D
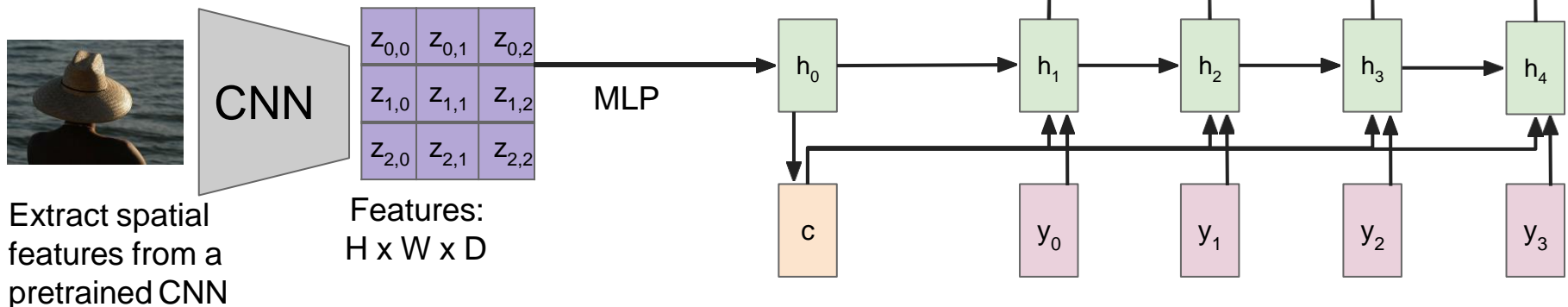
Image Captioning with RNNs and Attention

Attention idea: New context vector at every time step.

Each context vector will attend to different image regions

Attention Saccades in humans

$z_{0,0}$	$z_{0,1}$	$z_{0,2}$
$z_{1,0}$	$z_{1,1}$	$z_{1,2}$
$z_{2,0}$	$z_{2,1}$	$z_{2,2}$

CNN

h_0

Extract spatial features from a pretrained CNN

Features: H x W x D

Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

Attention and Transformers

Universidad Popular del Cesar

Image Captioning with RNNs and Attention

Compute alignments
scores (scalars):

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

$f_{att}(.)$ is an MLP

Alignment scores:
H x W

$e_{1,0,0}$	$e_{1,0,1}$	$e_{1,0,2}$
$e_{1,1,0}$	$e_{1,1,1}$	$e_{1,1,2}$
$e_{1,2,0}$	$e_{1,2,1}$	$e_{1,2,2}$

CNN

$z_{0,0}$	$z_{0,1}$	$z_{0,2}$
$z_{1,0}$	$z_{1,1}$	$z_{1,2}$
$z_{2,0}$	$z_{2,1}$	$z_{2,2}$

h_0

Extract spatial
features from a
pretrained CNN

Features:
H x W x D

Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

Attention and Transformers
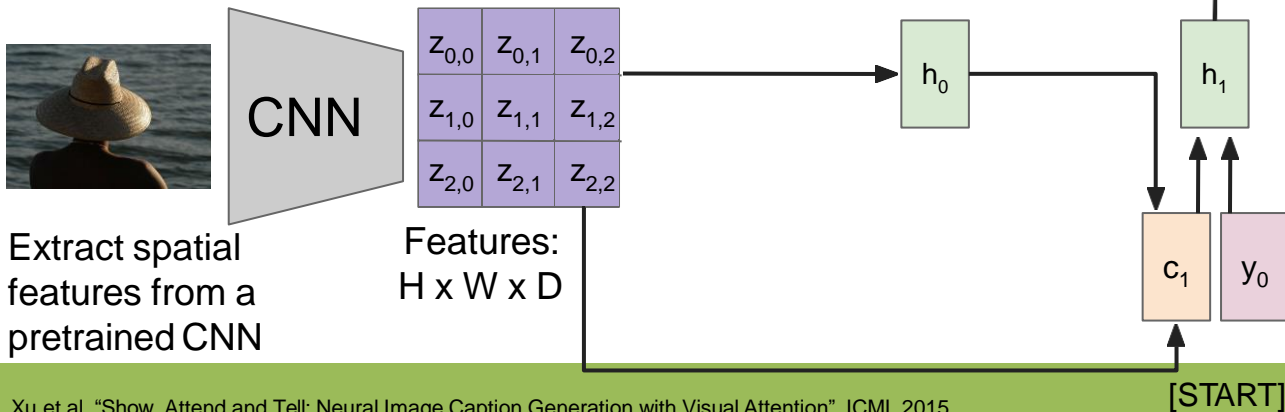
Universidad
Popular del Cesar

Image Captioning with RNNs and Attention

Compute alignments scores (scalars):

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

$f_{att}(.)$ is an MLP

Alignment scores:
H x W

$e_{1,0,0}$	$e_{1,0,1}$	$e_{1,0,2}$
$e_{1,1,0}$	$e_{1,1,1}$	$e_{1,1,2}$
$e_{1,2,0}$	$e_{1,2,1}$	$e_{1,2,2}$

Attention:
H x W

$a_{1,0,0}$	$a_{1,0,1}$	$a_{1,0,2}$
$a_{1,1,0}$	$a_{1,1,1}$	$a_{1,1,2}$
$a_{1,2,0}$	$a_{1,2,1}$	$a_{1,2,2}$

Normalize to get attention weights:

$$a_{t,:,:} = softmax(e_{t,:,:})$$

$0 < a_{t,i,j} < 1$,
attention values sum to 1

CNN

$z_{0,0}$	$z_{0,1}$	$z_{0,2}$
$z_{1,0}$	$z_{1,1}$	$z_{1,2}$
$z_{2,0}$	$z_{2,1}$	$z_{2,2}$

h_0

Extract spatial features from a pretrained CNN

Features:
H x W x D

Attention and Transformers

Universidad Popular del Cesar

Image Captioning with RNNs and Attention

Compute alignments scores (scalars):

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

$f_{att}(.)$ is an MLP

Alignment scores: H x W

$e_{1,0,0}$	$e_{1,0,1}$	$e_{1,0,2}$
$e_{1,1,0}$	$e_{1,1,1}$	$e_{1,1,2}$
$e_{1,2,0}$	$e_{1,2,1}$	$e_{1,2,2}$

Attention: H x W

$a_{1,0,0}$	$a_{1,0,1}$	$a_{1,0,2}$
$a_{1,1,0}$	$a_{1,1,1}$	$a_{1,1,2}$
$a_{1,2,0}$	$a_{1,2,1}$	$a_{1,2,2}$

Normalize to get attention weights:

$$a_{t,:,:} = softmax(e_{t,:,:})$$

$0 < a_{t,i,j} < 1$,
attention values sum to 1

Compute context vector:

$$c_t = \sum_{i,j} a_{t,i,j} \, z_{t,i,j}$$

CNN

$z_{0,0}$	$z_{0,1}$	$z_{0,2}$
$z_{1,0}$	$z_{1,1}$	$z_{1,2}$
$z_{2,0}$	$z_{2,1}$	$z_{2,2}$

h_0

Extract spatial features from a pretrained CNN

Features: H x W x D

c_1

(X)

Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

Attention and Transformers

Universidad Popular del Cesar

Image Captioning with RNNs and Attention

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

Decoder: $y_t = g_V(y_{t-1}, h_{t-1}, c_t)$
New context vector at every time step

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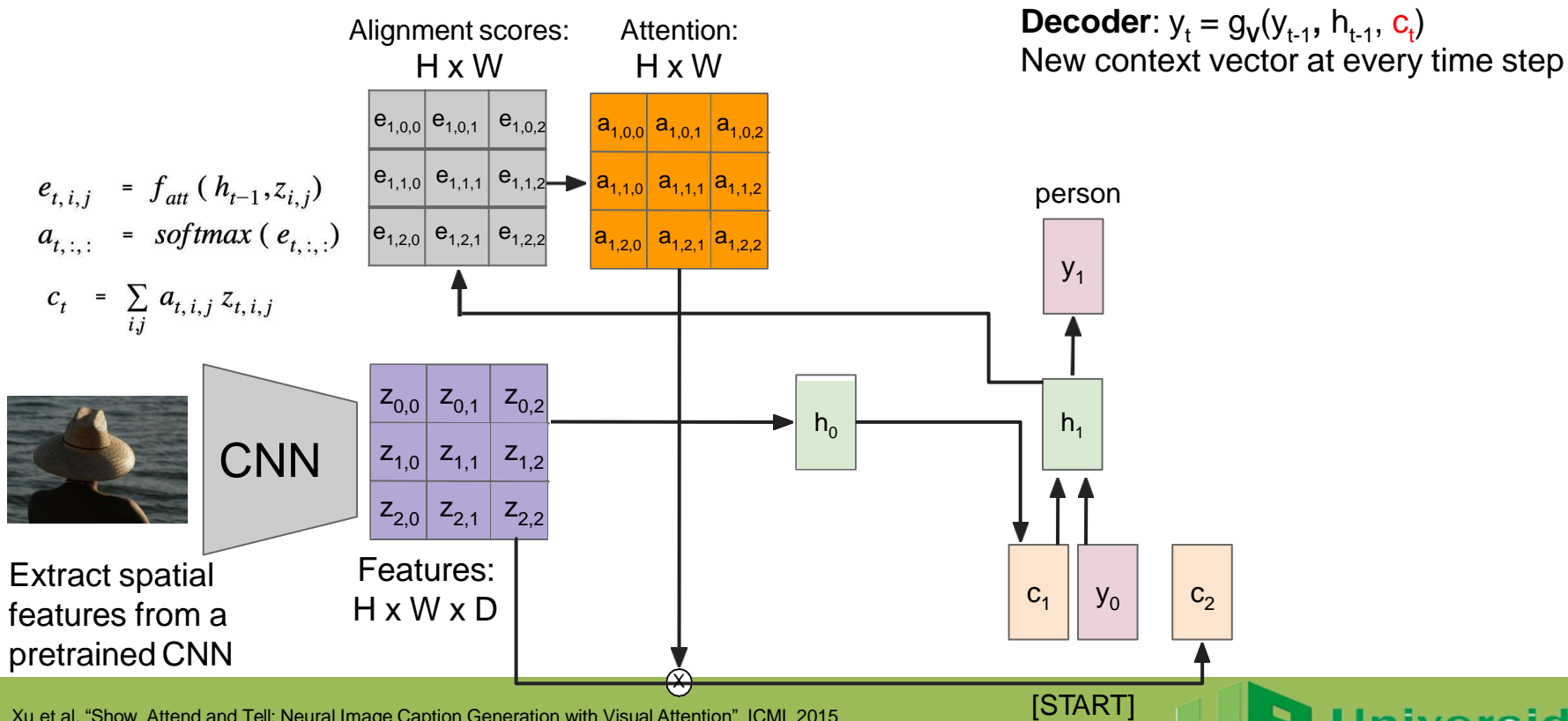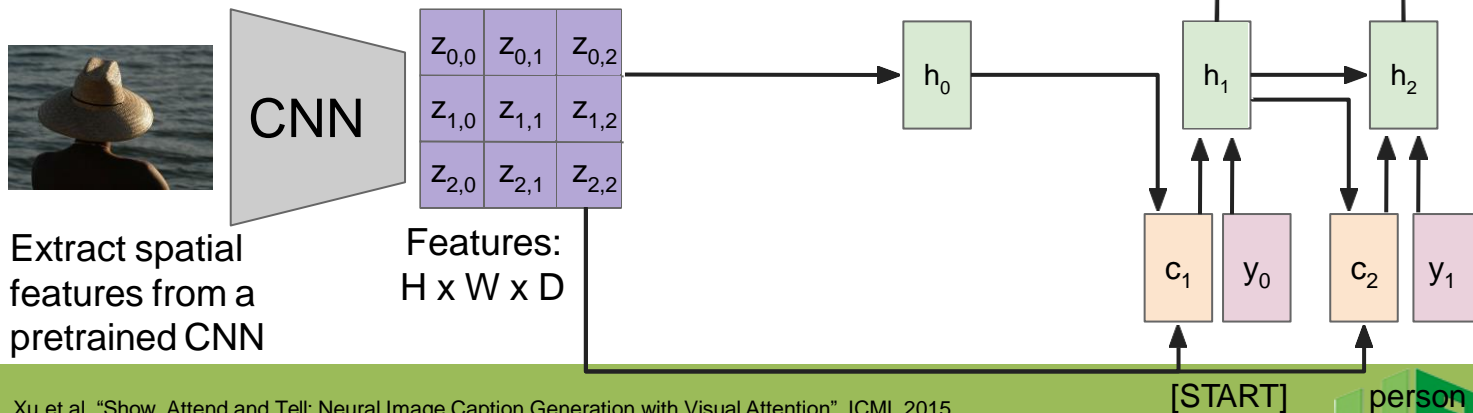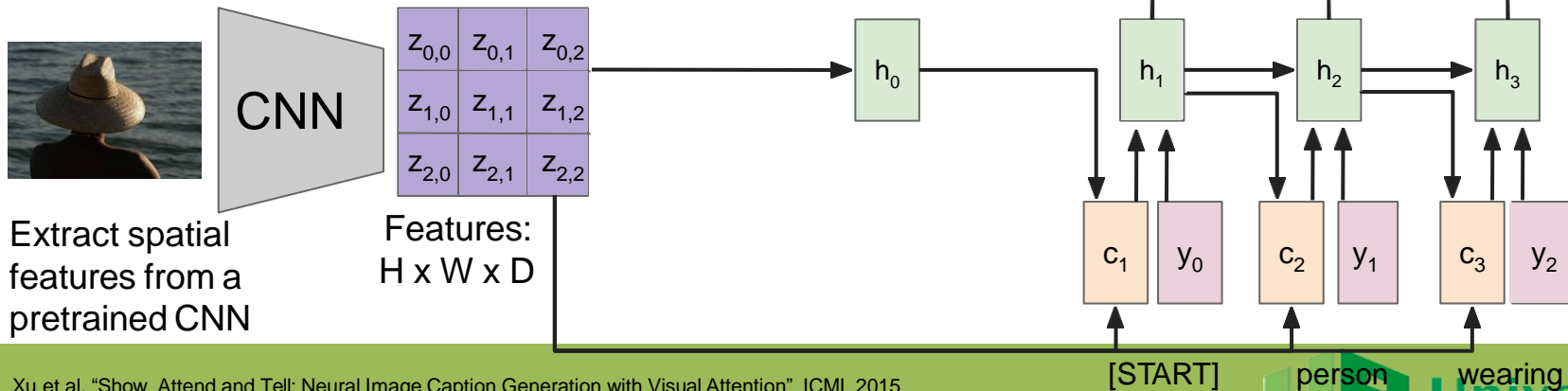$$a_{t,:,:} = softmax(e_{t,:,:})$$
$$c_t = \sum_{i,j} a_{t,i,j} z_{t,i,j}$$

CNN

Extract spatial features from a pretrained CNN

Features: H x W x D

person

y_1

h_0 h_1

c_1 y_0

[START]

Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

Attention and Transformers

Universidad Popular del Cesar

Image Captioning with RNNs and Attention

Alignment scores:
H x W

Attention:
H x W

Decoder: $y_t = g_V(y_{t-1}, h_{t-1}, c_t)$
New context vector at every time step

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

$e_{1,0,0}$	$e_{1,0,1}$	$e_{1,0,2}$
$e_{1,1,0}$	$e_{1,1,1}$	$e_{1,1,2}$
$e_{1,2,0}$	$e_{1,2,1}$	$e_{1,2,2}$

$a_{1,0,0}$	$a_{1,0,1}$	$a_{1,0,2}$
$a_{1,1,0}$	$a_{1,1,1}$	$a_{1,1,2}$
$a_{1,2,0}$	$a_{1,2,1}$	$a_{1,2,2}$

person

y_1

CNN

$z_{0,0}$	$z_{0,1}$	$z_{0,2}$
$z_{1,0}$	$z_{1,1}$	$z_{1,2}$
$z_{2,0}$	$z_{2,1}$	$z_{2,2}$

h_0

h_1

Extract spatial
features from a
pretrained CNN

Features:
H x W x D

c_1 y_0 c_2

⊗

[START]

Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

Attention and Transformers

37

Universidad
Popular del Cesar

Image Captioning with RNNs and Attention

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

Decoder: $y_t = g_v(y_{t-1}, h_{t-1}, c_t)$
New context vector at every time step

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

$$a_{t,:,:} = softmax(e_{t,:,:})$$

$$c_t = \sum_{i,j} a_{t,i,j}\, z_{t,i,j}$$

person   wearing
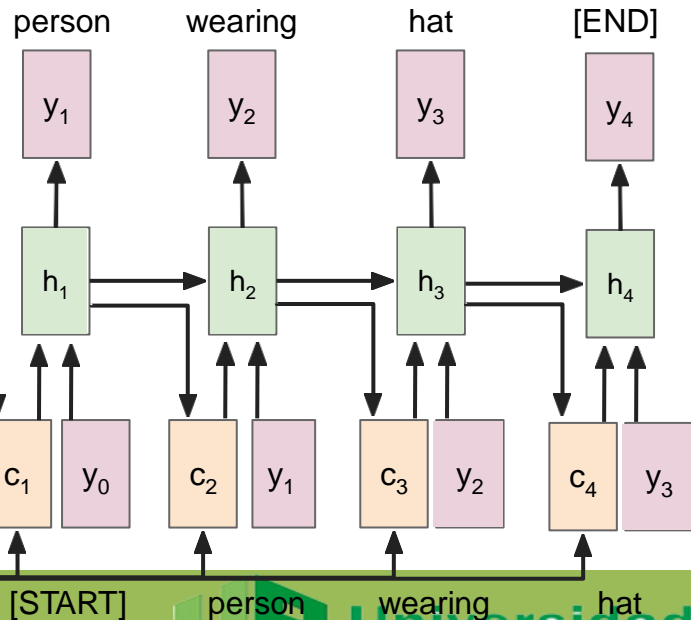
y_1 y_2

CNN

$z_{0,0}$	$z_{0,1}$	$z_{0,2}$
$z_{1,0}$	$z_{1,1}$	$z_{1,2}$
$z_{2,0}$	$z_{2,1}$	$z_{2,2}$

h_0 h_1 h_2

c_1 y_0 c_2 y_1

Extract spatial features from a pretrained CNN

Features: H x W x D

[START]   person

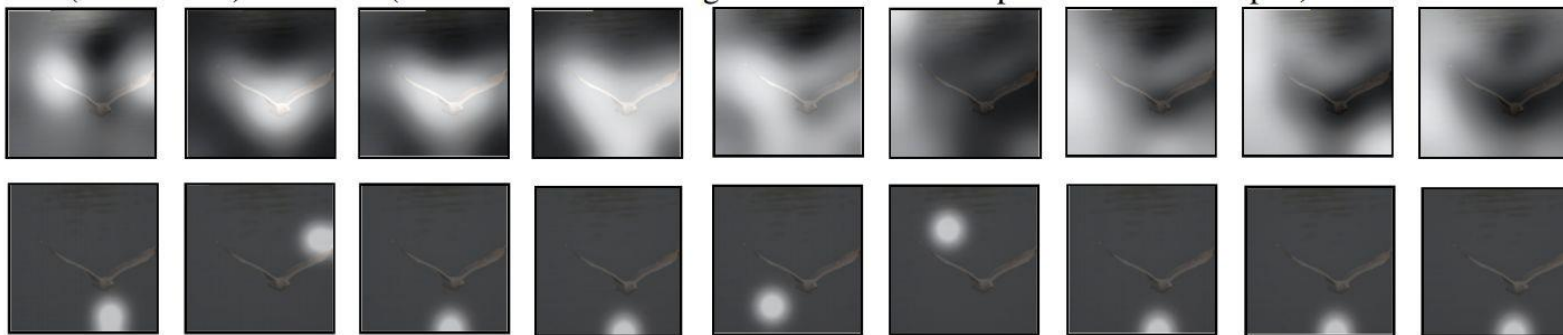Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

Attention and Transformers

Universidad Popular del Cesar

Image Captioning with RNNs and Attention

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

Decoder: $y_t = g_v(y_{t-1}, h_{t-1}, c_t)$
New context vector at every time step

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

person   wearing   hat

y_1 y_2 y_3

Extract spatial features from a pretrained CNN

Features: H x W x D

$z_{0,0}$ $z_{0,1}$ $z_{0,2}$
$z_{1,0}$ $z_{1,1}$ $z_{1,2}$
$z_{2,0}$ $z_{2,1}$ $z_{2,2}$

h_0 h_1 h_2 h_3

c_1 y_0 c_2 y_1 c_3 y_2

[START]   person   wearing

Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

Attention and Transformers

39

Universidad Popular del Cesar

Image Captioning with RNNs and Attention

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

Decoder: $y_t = g_v(y_{t-1}, h_{t-1}, c_t)$
New context vector at every time step

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

Extract spatial features from a pretrained CNN

Features:
H x W x D

Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

Attention and Transformers

41

Universidad Popular del Cesar

Image Captioning with RNNs and Attention

Alignment scores: H x W

Attention: H x W

This entire process is differentiable.
- model chooses its own attention weights. No attention supervision is required

Extract spatial features from a pretrained CNN

Features: H x W x D

Attention and Transformers

Universidad Popular del Cesar

Image Captioning with Attention

Soft attention

Hard attention
(requires
reinforcement
learning)

A   bird   flying   over   a   body   of   water   .

Attention and Transformers

Universidad
Popular del Cesar

Image Captioning with Attention

A woman is throwing a <u>frisbee</u> in a park.

A <u>dog</u> is standing on a hardwood floor.

A <u>stop</u> sign is on a road with a mountain in the background.

A little <u>girl</u> sitting on a bed with a teddy bear.

A group of <u>people</u> sitting on a boat in the water.

A giraffe standing in a forest with <u>trees</u> in the background.

Xu et al, "Show, Attend, and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

Attention and Transformers

Universidad Popular del Cesar

Image Captioning with RNNs and Attention

Alignment scores:
H x W

$e_{1,0,0}$	$e_{1,0,1}$	$e_{1,0,2}$
$e_{1,1,0}$	$e_{1,1,1}$	$e_{1,1,2}$
$e_{1,2,0}$	$e_{1,2,1}$	$e_{1,2,2}$

Attention:
H x W

$a_{1,0,0}$	$a_{1,0,1}$	$a_{1,0,2}$
$a_{1,1,0}$	$a_{1,1,1}$	$a_{1,1,2}$
$a_{1,2,0}$	$a_{1,2,1}$	$a_{1,2,2}$

This entire process is differentiable.
- model chooses its own attention weights. No attention supervision is required

CNN

$z_{0,0}$	$z_{0,1}$	$z_{0,2}$
$z_{1,0}$	$z_{1,1}$	$z_{1,2}$
$z_{2,0}$	$z_{2,1}$	$z_{2,2}$

Extract spatial features from a pretrained CNN

Features:
H x W x D

person y_1 wearing y_2 hat y_3 [END] y_4

h_0 h_1 h_2 h_3 h_4

c_1 y_0 c_2 y_1 c_3 y_2 c_4 y_3

$\times$

[START]   person   wearing   hat

Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

Universidad Popular del Cesar

Attention we just saw in image captioning

$z_{0,0}$	$z_{0,1}$	$z_{0,2}$
$z_{1,0}$	$z_{1,1}$	$z_{1,2}$
$z_{2,0}$	$z_{2,1}$	$z_{2,2}$

Features

h

Inputs:
Features: **z** (shape: H x W x D)
Query: **h** (shape: D)

Attention we just saw in image captioning

Operations:
Alignment: $e_{i,j} = f_{att}(h, z_{i,j})$
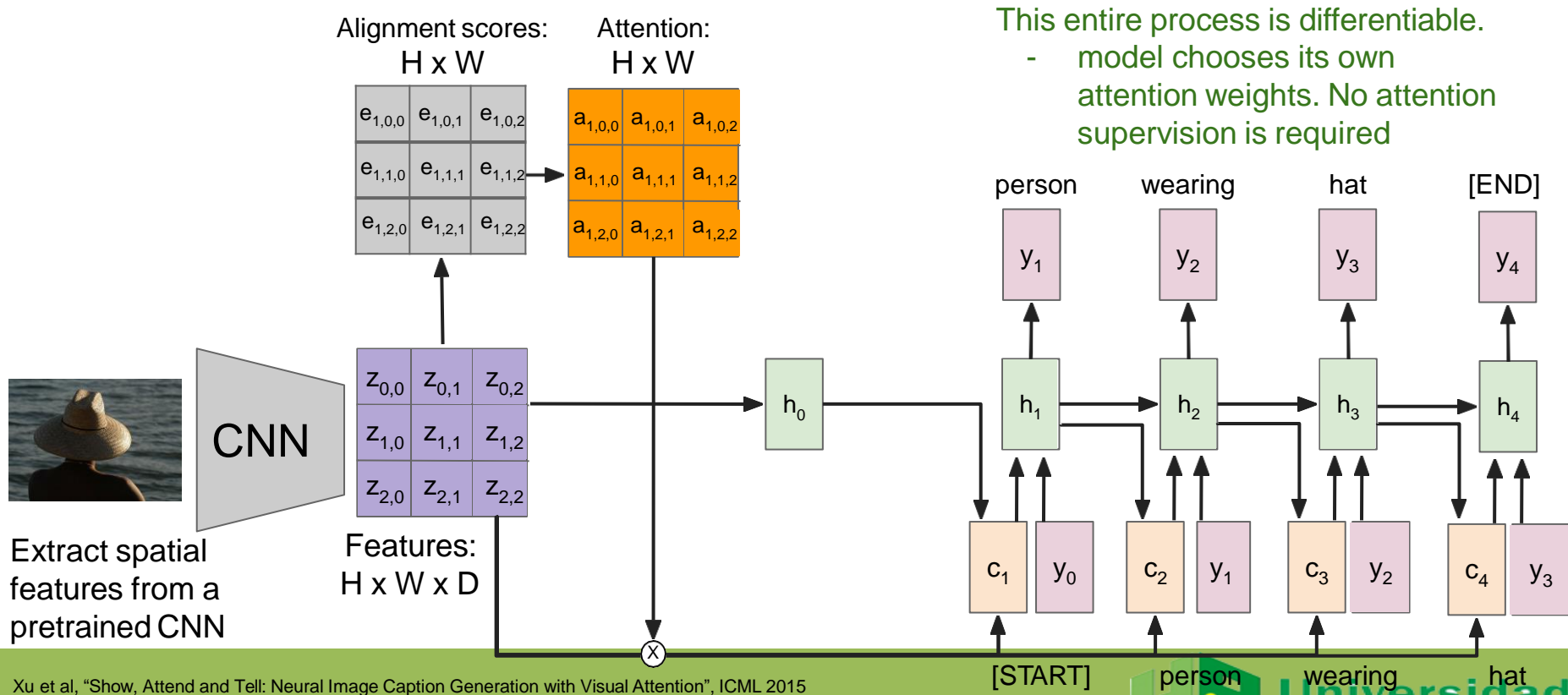
Features

$z_{0,0}$	$z_{0,1}$	$z_{0,2}$
$z_{1,0}$	$z_{1,1}$	$z_{1,2}$
$z_{2,0}$	$z_{2,1}$	$z_{2,2}$

$e_{0,0}$	$e_{0,1}$	$e_{0,2}$
$e_{1,0}$	$e_{1,1}$	$e_{1,2}$
$e_{2,0}$	$e_{2,1}$	$e_{2,2}$

Alignment

h

Inputs:
Features: **z** (shape: H x W x D)
Query: **h** (shape: D)

Attention and Transformers

Universidad
Popular del Cesar

Attention we just saw in image captioning
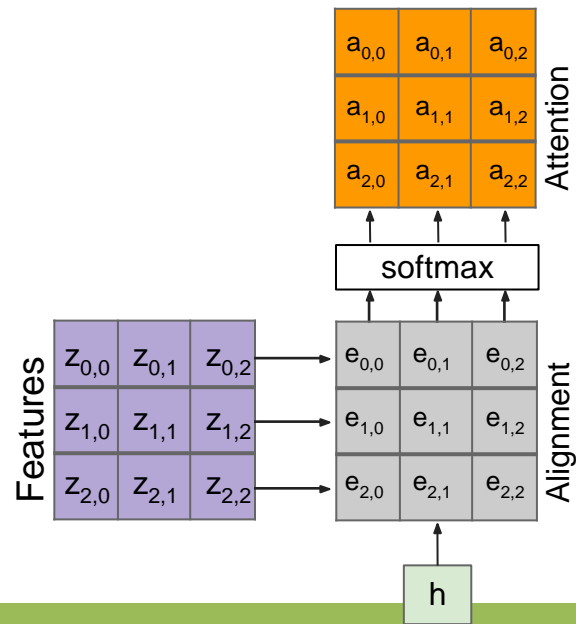
Operations:
Alignment: $e_{i,j} = f_{att}(h, z_{i,j})$
Attention: $\mathbf{a} = \text{softmax}(\mathbf{e})$

Inputs:
Features: $\mathbf{z}$ (shape: H x W x D)
Query: $\mathbf{h}$ (shape: D)

Attention and Transformers

Universidad
Popular del Cesar

Attention we just saw in image captioning

Outputs:
context vector: $\mathbf{c}$ (shape: D)

Operations:
Alignment: $e_{i,j} = f_{att}(h, z_{i,j})$
Attention: $\mathbf{a} = \text{softmax}(\mathbf{e})$
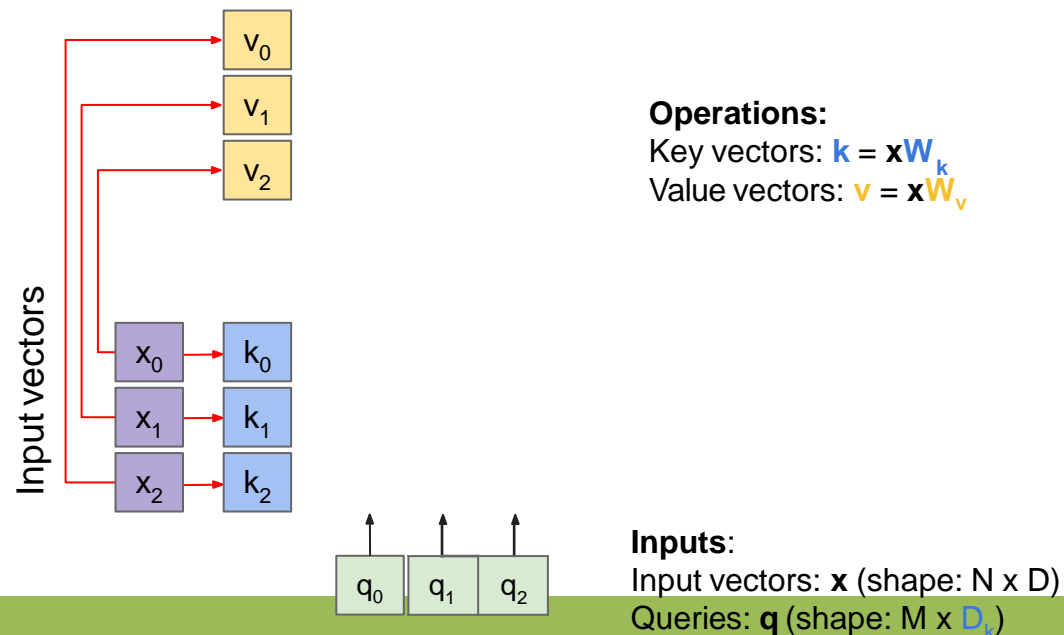Output: $\mathbf{c} = \sum_{i,j} a_{i,j} z_{i,j}$

Inputs:
Features: $\mathbf{z}$ (shape: H x W x D)
Query: $\mathbf{h}$ (shape: D)

Universidad
Popular del Cesar

General attention layer

Outputs:
context vector: c (shape: D)

Operations:
Alignment: $e_i = f_{att}(h, x_i)$
Attention: $a = \text{softmax}(e)$
Output: $c = \sum_i a_i x_i$
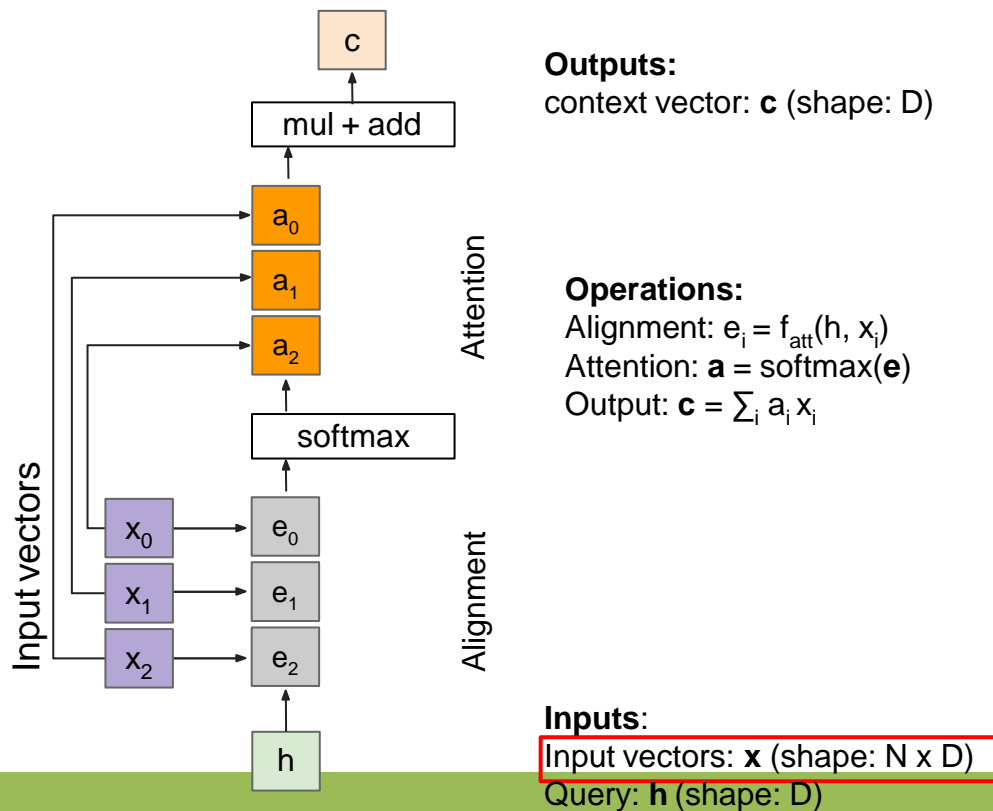
Attention operation is **permutation invariant.**
- Doesn't care about ordering of the features
- Stretch H x W = N into N vectors

Inputs:
Input vectors: x (shape: N x D)
Query: h (shape: D)

Universidad Popular del Cesar

General attention layer
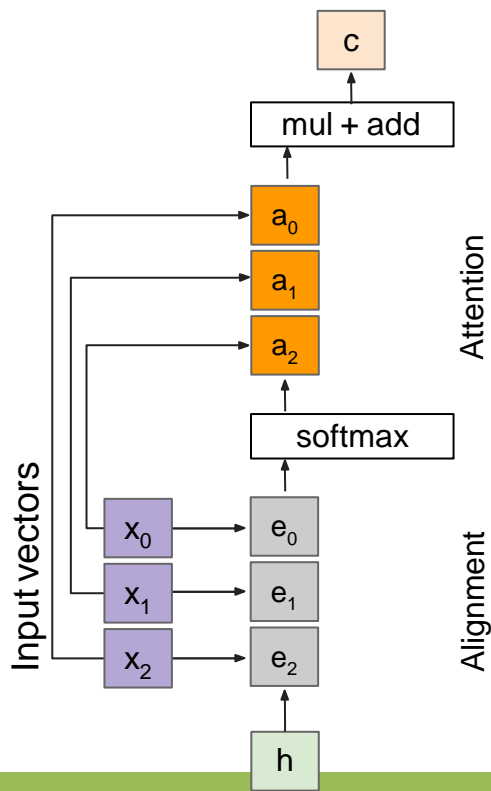
Outputs:
context vector: c (shape: D)

Operations:
Alignment: $e_i = h \cdot x_i$
Attention: $a = \text{softmax}(e)$
Output: $c = \sum_i a_i x_i$

Change $f_{att}(.)$ to a simple dot product
- only works well with key & value transformation trick (will mention in a few slides)

Inputs:
Input vectors: x (shape: N x D)
Query: h (shape: D)

Universidad
Popular del Cesar

General attention layer
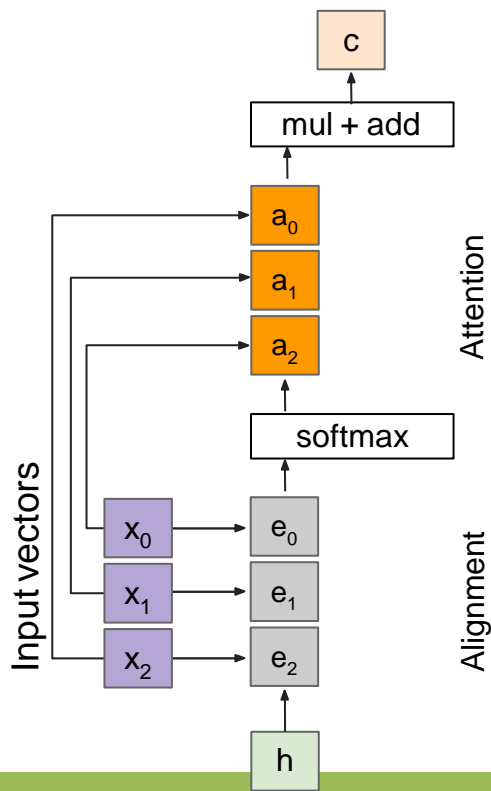
Outputs:
context vector: **c** (shape: D)

Operations:
Alignment: $e_i = h \cdot x_i / \sqrt{D}$
Attention: $\mathbf{a} = \text{softmax}(\mathbf{e})$
Output: $\mathbf{c} = \sum_i a_i x_i$

Inputs:
Input vectors: **x** (shape: N x D)
Query: **h** (shape: D)

Change $f_{att}(.)$ to a scaled simple dot product
- Larger dimensions means more terms in the dot product sum.
- So, the variance of the logits is higher. Large magnitude vectors will produce much higher logits.
- So, the post-softmax distribution has lower-entropy, assuming logits are IID.
- Ultimately, these large magnitude vectors will cause softmax to peak and assign very little weight to all others
- Divide by $\sqrt{D}$ to reduce effect of large magnitude vectors

Attention and Transformers

Universidad
Popular del Cesar

General attention layer

Outputs:
context vectors: $\mathbf{y}$ (shape: D)

Multiple query vectors
- each query creates a new output context vector

Operations:
Alignment: $e_{i,j} = q_j \cdot x_i / \sqrt{D}$
Attention: $\mathbf{a} = \text{softmax}(\mathbf{e})$
Output: $y_j = \sum_i a_{i,j} x_i$

Inputs:
Input vectors: $\mathbf{x}$ (shape: N x D)
Queries: $\mathbf{q}$ (shape: M x D)

Multiple query vectors

Universidad
Popular del Cesar

General attention layer
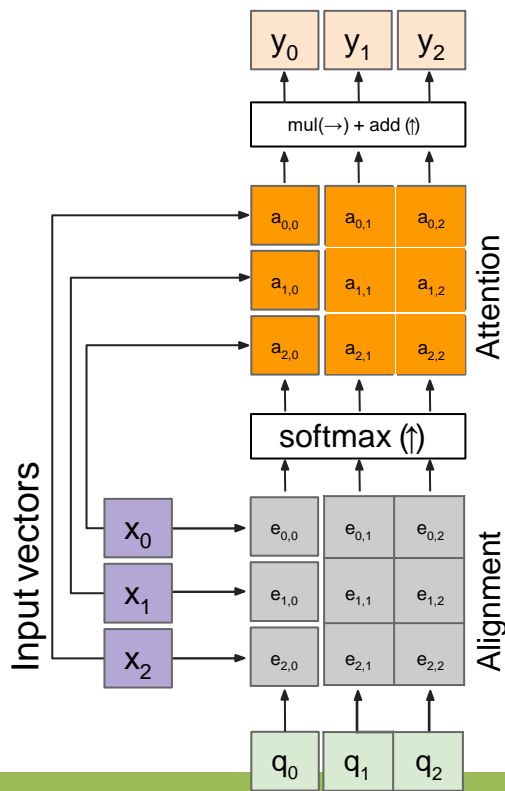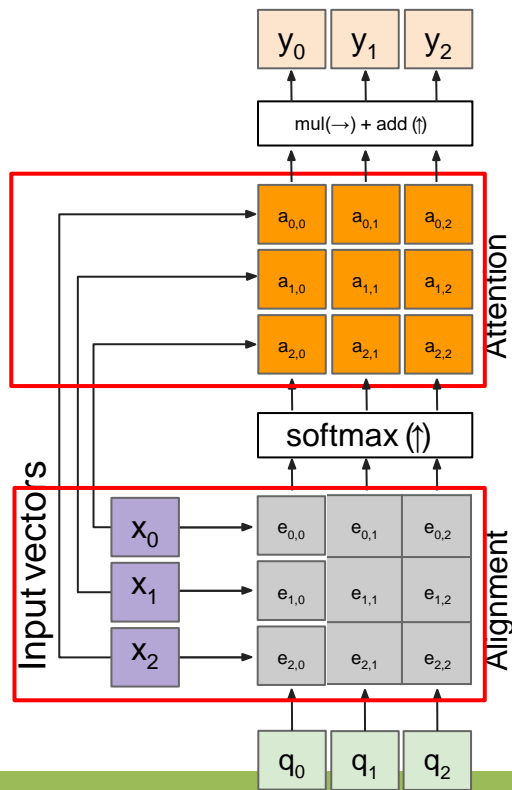
Outputs:
context vectors: **y** (shape: D)

Operations:
Alignment: $e_{i,j} = q_j \cdot x_i / \sqrt{D}$
Attention: $\mathbf{a} = \text{softmax}(\mathbf{e})$
Output: $y_j = \sum_i a_{i,j} x_i$

Inputs:
Input vectors: **x** (shape: N x D)
Queries: **q** (shape: M x D)

Notice that the input vectors are used for both the alignment as well as the attention calculations.
- We can add more expressivity to the layer by adding a different FC layer before each of the two steps.

Universidad
Popular del Cesar

General attention layer

Operations:
Key vectors: $\mathbf{k} = \mathbf{x}\mathbf{W}_k$
Value vectors: $\mathbf{v} = \mathbf{x}\mathbf{W}_v$

Notice that the input vectors are used for both the alignment as well as the attention calculations.
- We can add more expressivity to the layer by adding a different FC layer before each of the two steps.

Inputs:
Input vectors: $\mathbf{x}$ (shape: N x D)
Queries: $\mathbf{q}$ (shape: M x D_k)

Universidad Popular del Cesar

General attention layer

Outputs:
context vectors: **y** (shape: D_v)

The input and output dimensions can now change depending on the key and value FC layers

Notice that the input vectors are used for both the alignment as well as the attention calculations.
- We can add more expressivity to the layer by adding a different FC layer before each of the two steps.

Operations:
Key vectors: $\mathbf{k} = \mathbf{x}\mathbf{W_k}$
Value vectors: $\mathbf{v} = \mathbf{x}\mathbf{W_v}$
Alignment: $e_{i,j} = q_j \cdot k_i / \sqrt{D}$
Attention: $\mathbf{a} = \text{softmax}(\mathbf{e})$
Output: $y_j = \sum_i a_{i,j} v_i$

Inputs:
Input vectors: **x** (shape: N x D)
Queries: **q** (shape: M x D_k)

Universidad Popular del Cesar

General attention layer

Outputs:
context vectors: $\mathbf{y}$ (shape: D_v)

Recall that the query vector was a function of the input vectors
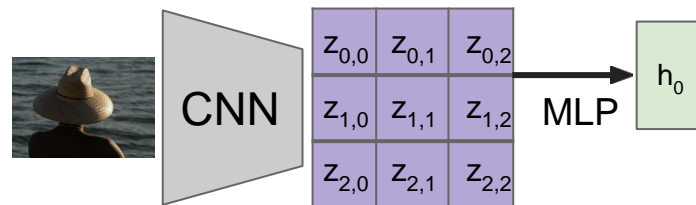
Operations:
Key vectors: $\mathbf{k} = \mathbf{x}\mathbf{W_k}$
Value vectors: $\mathbf{v} = \mathbf{x}\mathbf{W_v}$
Alignment: $e_{i,j} = q_j \cdot k_i / \sqrt{D}$
Attention: $\mathbf{a} = \text{softmax}(\mathbf{e})$
Output: $y_j = \sum_i a_{i,j} v_i$

Encoder: $h_0 = f_{\mathbf{W}}(\mathbf{z})$
where $\mathbf{z}$ is spatial CNN features
$f_{\mathbf{W}}(.)$ is an MLP

Inputs:
Input vectors: $\mathbf{x}$ (shape: N x D)
Queries: $\mathbf{q}$ (shape: M x D_k)

Self attention layer

We can calculate the query vectors from the input vectors, therefore, defining a "self-attention" layer.

Operations:
Key vectors: $\mathbf{k} = \mathbf{xW_k}$
Value vectors: $\mathbf{v} = \mathbf{xW_v}$
Query vectors: $\mathbf{q} = \mathbf{xW_q}$
Alignment: $e_{i,j} = q_j \cdot k_i / \sqrt{D}$
Attention: $\mathbf{a} = \text{softmax}(\mathbf{e})$
Output: $y_j = \sum_i a_{i,j} v_i$

Instead, query vectors are calculated using a FC layer.

No input query vectors anymore

Input vectors
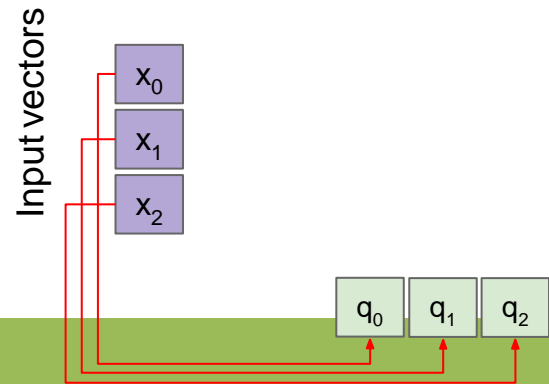
x_0

x_1

x_2

q_0 q_1 q_2

Inputs:
Input vectors: $\mathbf{x}$ (shape: N x D)
~~Queries: $\mathbf{q}$ (shape: M x D_k)~~

Attention and Transformers

Universidad
Popular del Cesar

Self attention layer

Outputs:
context vectors: $\mathbf{y}$ (shape: D_v)

Operations:
Key vectors: $\mathbf{k} = \mathbf{x}\mathbf{W_k}$
Value vectors: $\mathbf{v} = \mathbf{x}\mathbf{W_v}$
Query vectors: $\mathbf{q} = \mathbf{x}\mathbf{W_q}$
Alignment: $e_{i,j} = q_j \cdot k_i / \sqrt{D}$
Attention: $\mathbf{a} = \text{softmax}(\mathbf{e})$
Output: $y_j = \sum_i a_{i,j} v_i$

Inputs:
Input vectors: $\mathbf{x}$ (shape: N x D)

Attention and Transformers

Universidad
Popular del Cesar

Self attention layer - attends over sets of inputs

Outputs:
context vectors: $\mathbf{y}$ (shape: D_v)

Operations:
Key vectors: $\mathbf{k} = \mathbf{x}\mathbf{W_k}$
Value vectors: $\mathbf{v} = \mathbf{x}\mathbf{W_v}$
Query vectors: $\mathbf{q} = \mathbf{x}\mathbf{W_q}$
Alignment: $e_{i,j} = q_j \cdot k_i / \sqrt{D}$
Attention: $\mathbf{a} = \text{softmax}(\mathbf{e})$
Output: $y_j = \sum_i a_{i,j} v_i$

Inputs:
Input vectors: $\mathbf{x}$ (shape: N x D)

Attention and Transformers

Universidad Popular del Cesar
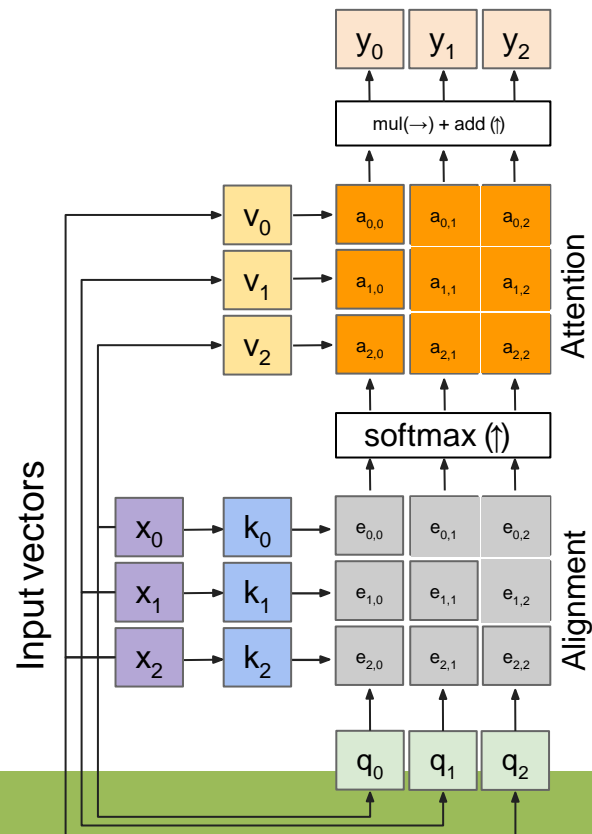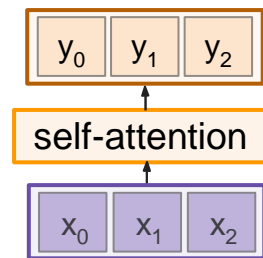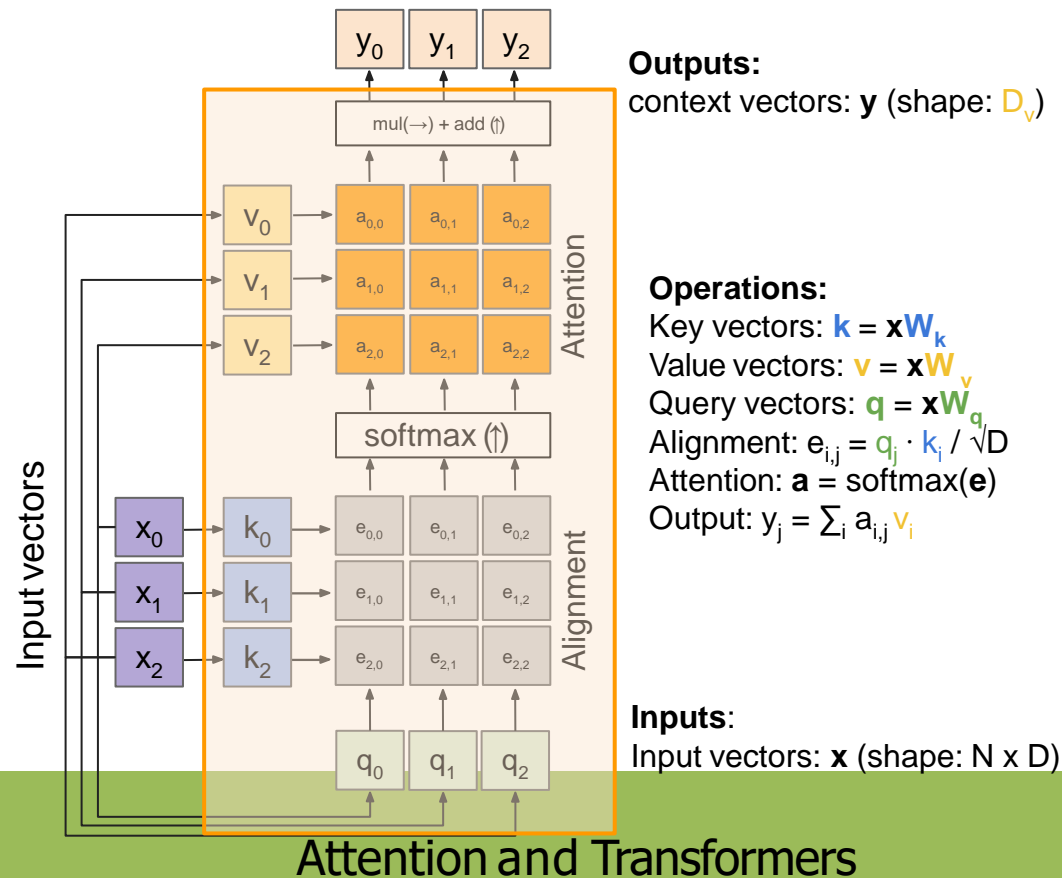
Self attention layer - attends over sets of inputs
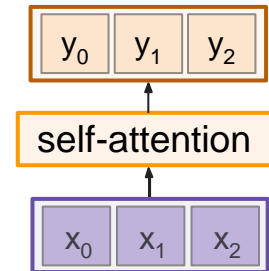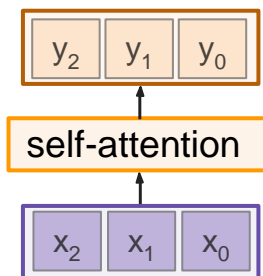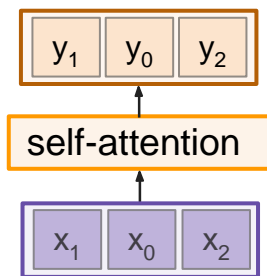
Permutation equivariant

Self-attention layer doesn't care about the orders of the inputs!

Problem: How can we encode ordered sequences like language or spatially ordered image features?

Positional encoding
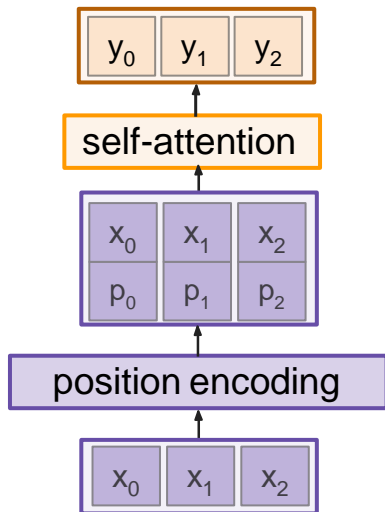
Concatenate/add special positional encoding p_j to each input vector x_j

We use a function pos: N $\rightarrow$ R^d
to process the position j of the vector into a d-dimensional vector
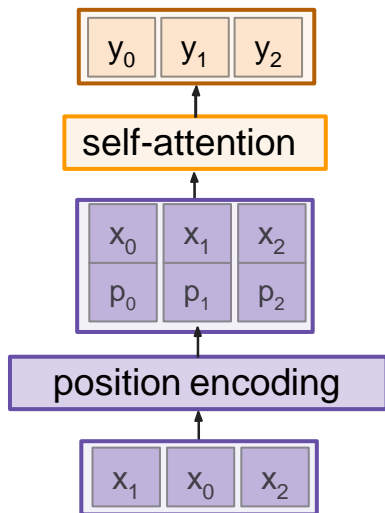
Desiderata of pos(.) :
1. It should output a **unique** encoding for each time-step (word's position in a sentence)
2. **Distance** between any two time-steps should be consistent across sentences with different lengths.
3. Our model should generalize to **longer** sentences without any efforts. Its values should be bounded.
4. It must be **deterministic**.

So, p_j = pos(j)

Universidad
Popular del Cesar

Positional encoding
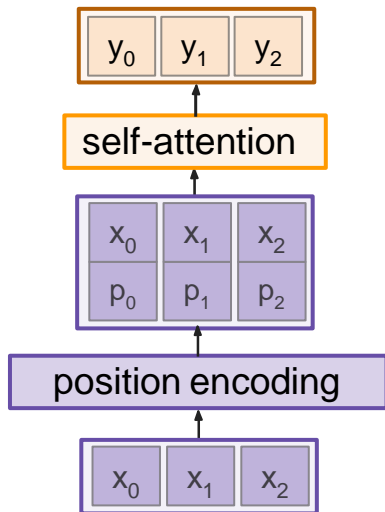
Concatenate special positional encoding p_j to each input vector x_j

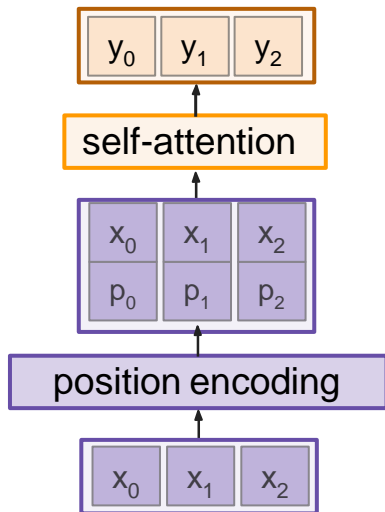We use a function $pos: N \rightarrow R^d$ to process the position j of the vector into a d-dimensional vector

Options for $pos(.)$

1. Learn a lookup table:
    - Learn parameters to use for $pos(t)$ for t ε[0, T)
    - Lookup table contains T x d parameters.

Desiderata of $pos(.)$:
1. It should output a **unique** encoding for each time-step (word's position in a sentence)
2. **Distance** between any two time-steps should be consistent across sentences with different lengths.
3. Our model should generalize to **longer** sentences without any efforts. Its values should be bounded.
4. It must be **deterministic**.

So, $p_j = pos(j)$

Universidad
Popular del Cesar

Positional encoding

y_0 y_1 y_2

self-attention

x_0 x_1 x_2
p_0 p_1 p_2

position encoding

x_1 x_0 x_2

Concatenate special positional encoding p_j to each input vector x_j

We use a function $pos: N \rightarrow R^d$ to process the position j of the vector into a d-dimensional vector

So, $p_j = pos(j)$

Options for $pos(.)$

1. Learn a lookup table:
   - Learn parameters to use for $pos(t)$ for t ϵ [0, T]
   - Lookup table contains T x d parameters.

2. Design a fixed function with the desiderata

$$p(t) = \begin{bmatrix} \sin(\omega_1 . t) \\ \cos(\omega_1 . t) \\ \sin(\omega_2 . t) \\ \cos(\omega_2 . t) \\ \vdots \\ \sin(\omega_{d/2} . t) \\ \cos(\omega_{d/2} . t) \end{bmatrix}_d$$

where $\omega_k = \dfrac{1}{10000^{2k/d}}$

Vaswani at al. "Attention is all you need" NeurIPS 2017

Attention and Transformers

63

Universidad Popular del Cesar

Positional encoding

Concatenate special positional encoding p_j to each input vector x_j

We use a function pos: N $\rightarrow$ R^d to process the position j of the vector into a d-dimensional vector

So, $p_j = pos(j)$

Options for $pos(.)$

1. Learn a lookup table:
   - Learn parameters to use for $pos(t)$ for t ϵ[0, T)
   - Lookup table contains T x d parameters.

2. Design a fixed function with the desiderata

Intuition:

$$p(t) = \begin{bmatrix} \sin(\omega_1 . t) \\ \cos(\omega_1 . t) \\ \sin(\omega_2 . t) \\ \cos(\omega_2 . t) \\ \vdots \\ \sin(\omega_{d/2} . t) \\ \cos(\omega_{d/2} . t) \end{bmatrix}_d$$

0 :	0 0 0 0	8 :	1 0 0 0
1 :	0 0 0 1	9 :	1 0 0 1
2 :	0 0 1 0	10 :	1 0 1 0
3 :	0 0 1 1	11 :	1 0 1 1
4 :	0 1 0 0	12 :	1 1 0 0
5 :	0 1 0 1	13 :	1 1 0 1
6 :	0 1 1 0	14 :	1 1 1 0
7 :	0 1 1 1	15 :	1 1 1 1

where $\omega_k = \dfrac{1}{10000^{2k/d}}$

Masked self-attention layer
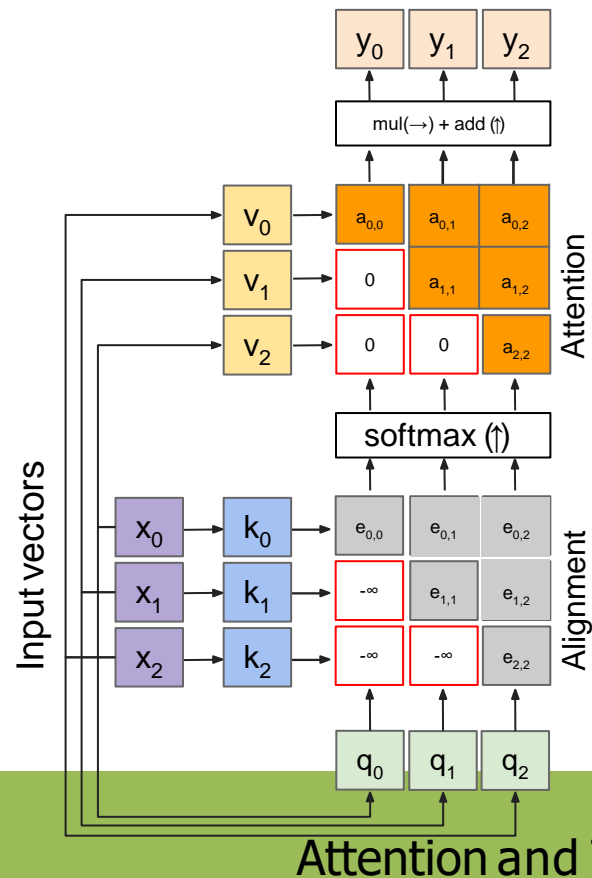
Outputs:
context vectors: $\mathbf{y}$ (shape: D_v)

Operations:
Key vectors: $\mathbf{k} = \mathbf{x}\mathbf{W_k}$
Value vectors: $\mathbf{v} = \mathbf{x}\mathbf{W_v}$
Query vectors: $\mathbf{q} = \mathbf{x}\mathbf{W_q}$
Alignment: $e_{i,j} = q_j \cdot k_i / \sqrt{D}$
Attention: $\mathbf{a} = \text{softmax}(\mathbf{e})$
Output: $y_j = \sum_i a_{i,j} v_i$

Inputs:
Input vectors: $\mathbf{x}$ (shape: N x D)

- Prevent vectors from looking at future vectors.
- Manually set alignment scores to -infinity

Attention and Transformers

Multi-head self-attention layer

- Multiple self-attention heads in parallel
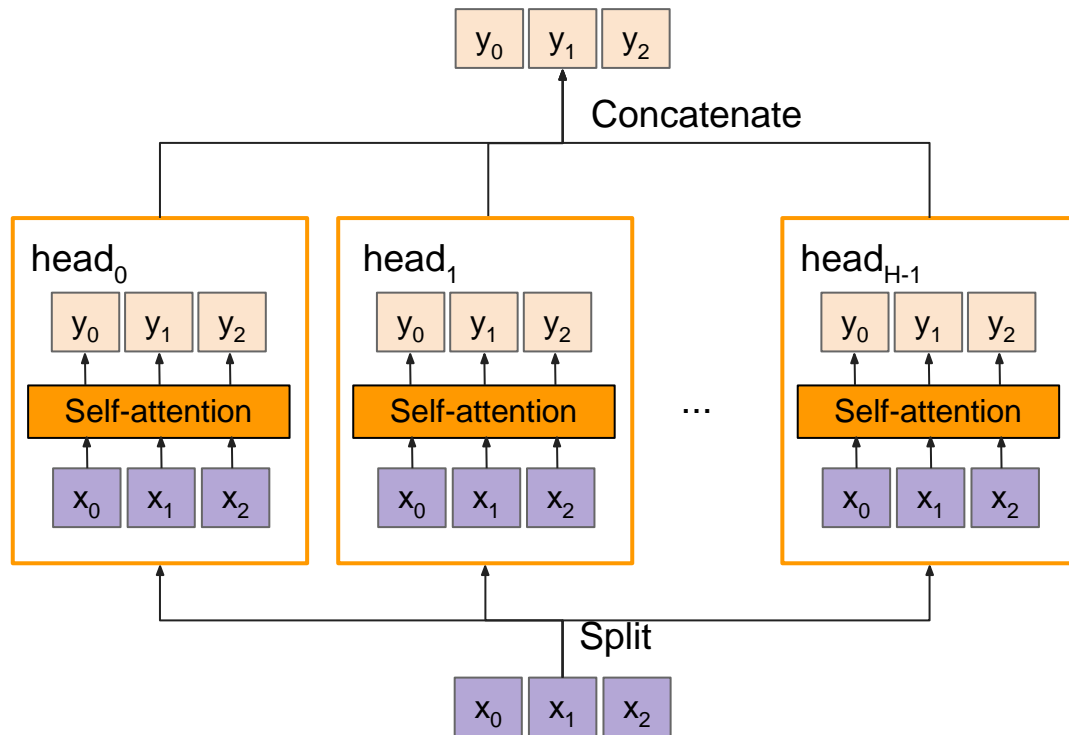
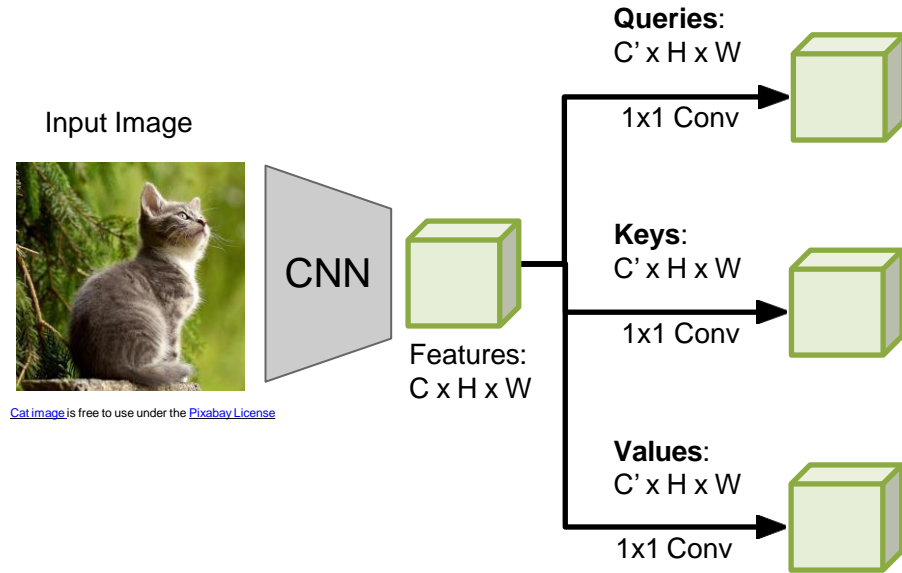# General attention versus self-attention

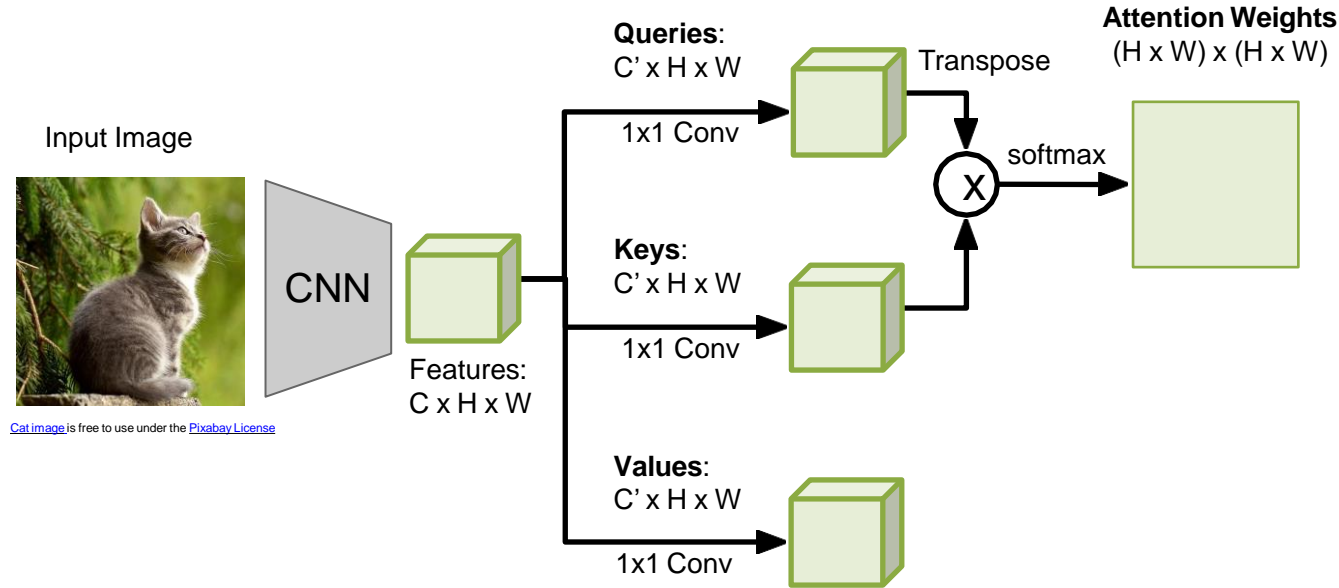# Example: CNN with Self-Attention

Input Image

CNN

Features:
C x H x W

Zhang et al, "Self-Attention Generative Adversarial Networks", ICML 2018

Attention and Transformers

Universidad
Popular del Cesar

Example: CNN with Self-Attention

Input Image

CNN

Features:
C x H x W

Queries:
C' x H x W

1x1 Conv

Keys:
C' x H x W

1x1 Conv

Values:
C' x H x W

1x1 Conv

Attention and Transformers

Universidad Popular del Cesar

Example: CNN with Self-Attention

Input Image

CNN

Features: C x H x W

Cat image is free to use under the Pixabay License

Queries: C' x H x W

1x1 Conv

Transpose

Keys: C' x H x W

1x1 Conv

Values: C' x H x W

1x1 Conv

softmax

X

Attention Weights (H x W) x (H x W)

Example: CNN with Self-Attention

Queries:
C' x H x W

1x1 Conv

Transpose

Attention Weights
(H x W) x (H x W)

Keys:
C' x H x W

1x1 Conv

Values:
C' x H x W

1x1 Conv

Input Image

CNN

Features:
C x H x W

softmax

X

X

C' x H x W

Attention and Transformers

Slide credit: Justin Johnson

Universidad
Popular del Cesar

Example: CNN with Self-Attention

Example: CNN with Self-Attention

Input Image

Cat image is free to use under the Pixabay License

CNN

Features:
C x H x W

Queries:
C' x H x W

1x1 Conv

Transpose

Keys:
C' x H x W

1x1 Conv

softmax

Attention Weights
(H x W) x (H x W)

Values:
C' x H x W

1x1 Conv

X

C' x H x W

1x1 Conv

C x H x W

+

Residual Connection

Self-Attention Module

Zhang et al, "Self-Attention Generative Adversarial Networks", ICML 2018

Attention and Transformers

73

Universidad
Popular del Cesar

Comparing RNNs to Transformer

RNNs
(+) LSTMs work reasonably well for long sequences.
(-) Expects an ordered sequences of inputs
(-) Sequential computation: subsequent hidden states can only be computed after the previous ones are done.

Transformer:
(+) Good at long sequences. Each attention calculation looks at all inputs.
(+) Can operate over unordered sets or ordered sequences with positional encodings.
(+) Parallel computation: All alignment and attention scores for all inputs can be done in parallel.
(-) Requires a lot of memory: N x M alignment and attention scalers need to be calculated and stored for a single self-attention head. (but GPUs are getting bigger and better)

Attention and Transformers

Universidad
Popular del Cesar

Attention Is All You Need

Ashish Vaswani[*]
Google Brain
avaswani@google.com

Noam Shazeer[*]
Google Brain
noam@google.com

Niki Parmar[*]
Google Research
nikip@google.com

Jakob Uszkoreit[*]
Google Research
usz@google.com

Llion Jones[*]
Google Research
llion@google.com

Aidan N. Gomez[*] [†]
University of Toronto
aidan@cs.toronto.edu

Łukasz Kaiser[*]
Google Brain
lukaszkaiser@google.com

Illia Polosukhin[*] [‡]
illia.polosukhin@gmail.com

"ImageNet Moment for Natural Language Processing"

Pretraining:
Download a lot of text from the internet

Train a giant Transformer model for language modeling

Finetuning:
Fine-tune the Transformer on your own NLP task

Universidad
Popular del Cesar

On the Opportunities and Risks of Foundation Models

Rishi Bommasani*   Drew A. Hudson   Ehsan Adeli   Russ Altman   Simran Arora
Sydney von Arx   Michael S. Bernstein   Jeannette Bohg   Antoine Bosselut   Emma Brunskill
Erik Brynjolfsson   Shyamal Buch   Dallas Card   Rodrigo Castellon   Niladri Chatterji
Annie Chen   Kathleen Creel   Jared Quincy Davis   Dorottya Demszky   Chris Donahue
Moussa Doumbouya   Esin Durmus   Stefano Ermon   John Etchemendy   Kawin Ethayarajh
Li Fei-Fei   Chelsea Finn   Trevor Gale   Lauren Gillespie   Karan Goel   Noah Goodman
Shelby Grossman   Neel Guha   Tatsunori Hashimoto   Peter Henderson   John Hewitt
Daniel E. Ho   Jenny Hong   Kyle Hsu   Jing Huang   Thomas Icard   Saahil Jain
Dan Jurafsky   Pratyusha Kalluri   Siddharth Karamcheti   Geoff Keeling   Fereshte Khani
Omar Khattab   Pang Wei Koh   Mark Krass   Ranjay Krishna   Rohith Kuditipudi
Ananya Kumar   Faisal Ladhak   Mina Lee   Tony Lee   Jure Leskovec   Isabelle Levent
Xiang Lisa Li   Xuechen Li   Tengyu Ma   Ali Malik   Christopher D. Manning
Suvir Mirchandani   Eric Mitchell   Zanele Munyikwa   Suraj Nair   Avanika Narayan
Deepak Narayanan   Ben Newman   Allen Nie   Juan Carlos Niebles   Hamed Nilforoshan
Julian Nyarko   Giray Ogut   Laurel Orr   Isabel Papadimitriou   Joon Sung Park   Chris Piech
Eva Portelance   Christopher Potts   Aditi Raghunathan   Rob Reich   Hongyu Ren
Frieda Rong   Yusuf Roohani   Camilo Ruiz   Jack Ryan   Christopher Ré   Dorsa Sadigh
Shiori Sagawa   Keshav Santhanam   Andy Shih   Krishnan Srinivasan   Alex Tamkin
Rohan Taori   Armin W. Thomas   Florian Tramèr   Rose E. Wang   William Wang   Bohan Wu
Jiajun Wu   Yuhuai Wu   Sang Michael Xie   Michihiro Yasunaga   Jiaxuan You   Matei Zaharia
Michael Zhang   Tianyi Zhang   Xikun Zhang   Yuhui Zhang   Lucia Zheng   Kaitlyn Zhou
Percy Liang*[1]

Center for Research on Foundation Models (CRFM)
Stanford Institute for Human-Centered Artificial Intelligence (HAI)
Stanford University

Attention and Transformers

Image Captioning using Transformers

Input: Image **I**
Output: Sequence $\mathbf{y} = y_1, y_2, ..., y_T$
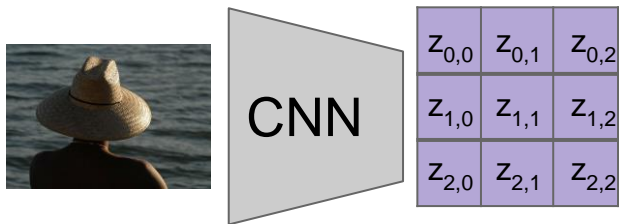
Extract spatial features from a pretrained CNN

Features: H x W x D

Image Captioning using Transformers

Input: Image **I**
Output: Sequence $\mathbf{y} = y_1, y_2,..., y_T$

Encoder: $\mathbf{c} = T_\mathbf{W}(\mathbf{z})$
where $\mathbf{z}$ is spatial CNN features
$T_\mathbf{W}(.)$ is the transformer encoder

Extract spatial features from a pretrained CNN

CNN

Features: H x W x D

$z_{0,0}$ $z_{0,1}$ $z_{0,2}$
$z_{1,0}$ $z_{1,1}$ $z_{1,2}$
$z_{2,0}$ $z_{2,1}$ $z_{2,2}$

$c_{0,0}$ $c_{0,1}$ $c_{0,2}$ … $c_{2,2}$

Transformer encoder

$z_{0,0}$ $z_{0,1}$ $z_{0,2}$ … $z_{2,2}$

Universidad Popular del Cesar
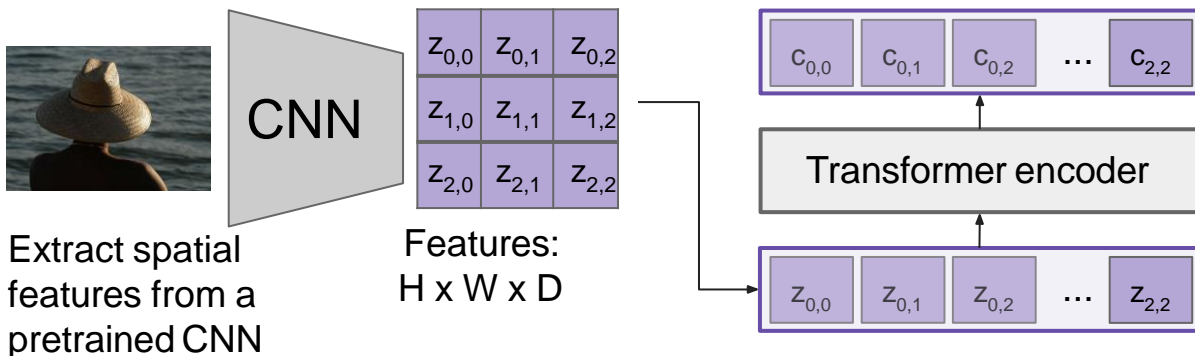
Image Captioning using Transformers

Input: Image $\mathbf{I}$
Output: Sequence $\mathbf{y} = y_1, y_2, ..., y_T$

Decoder: $y_t = T_\mathbf{D}(\mathbf{y}_{0:t-1}, \mathbf{c})$
where $T_\mathbf{D}(.)$ is the transformer decoder

Encoder: $\mathbf{c} = T_\mathbf{W}(\mathbf{z})$
where $\mathbf{z}$ is spatial CNN features
$T_\mathbf{W}(.)$ is the transformer encoder

person   wearing   hat   [END]

y_1 y_2 y_3 y_4

CNN

$z_{0,0}$	$z_{0,1}$	$z_{0,2}$
$z_{1,0}$	$z_{1,1}$	$z_{1,2}$
$z_{2,0}$	$z_{2,1}$	$z_{2,2}$

$c_{0,0}$ $c_{0,1}$ $c_{0,2}$ … $c_{2,2}$

Transformer decoder

Transformer encoder

$z_{0,0}$ $z_{0,1}$ $z_{0,2}$ … $z_{2,2}$

y_0 y_1 y_2 y_3

Extract spatial features from a pretrained CNN

Features: H x W x D

[START]   person   wearing   hat

Universidad
Popular del Cesar
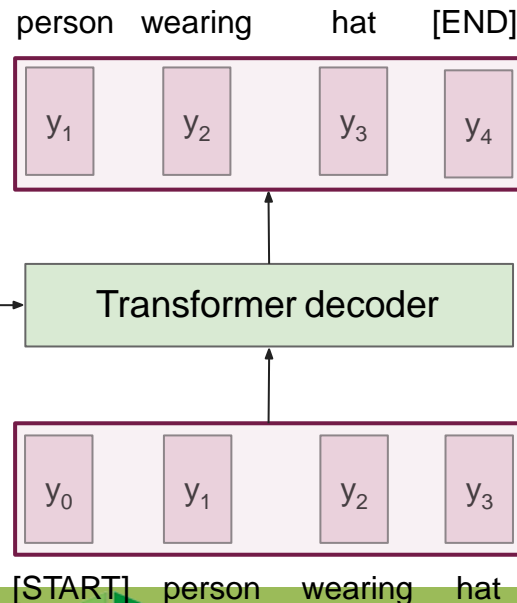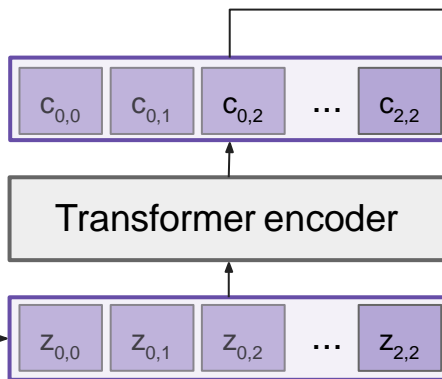
The Transformer encoder block
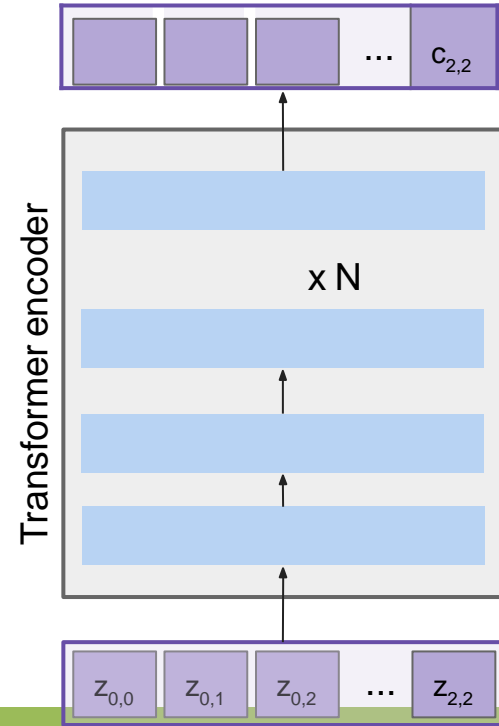
Made up of N encoder blocks.

In vaswani et al. $N = 6$, $D_q = 512$

Attention and Transformers

Vaswani et al. "Attention is all you need" NeurIPS 2017

Universidad
Popular del Cesar

The Transformer encoder block

Transformer encoder

$c_{0,0}$ $c_{0,1}$ $c_{0,2}$ … $c_{2,2}$

x N

$z_{0,0}$ $z_{0,1}$ $z_{0,2}$ … $z_{2,2}$

x_0 x_1 x_2 x_2
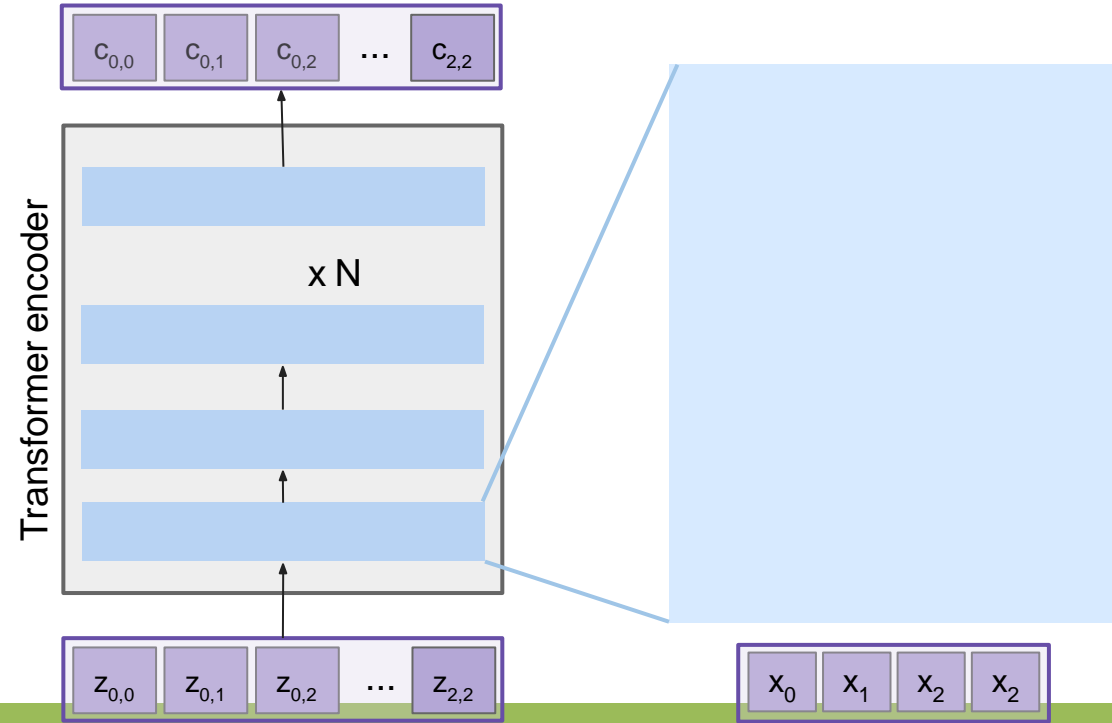
Let's dive into one encoder block
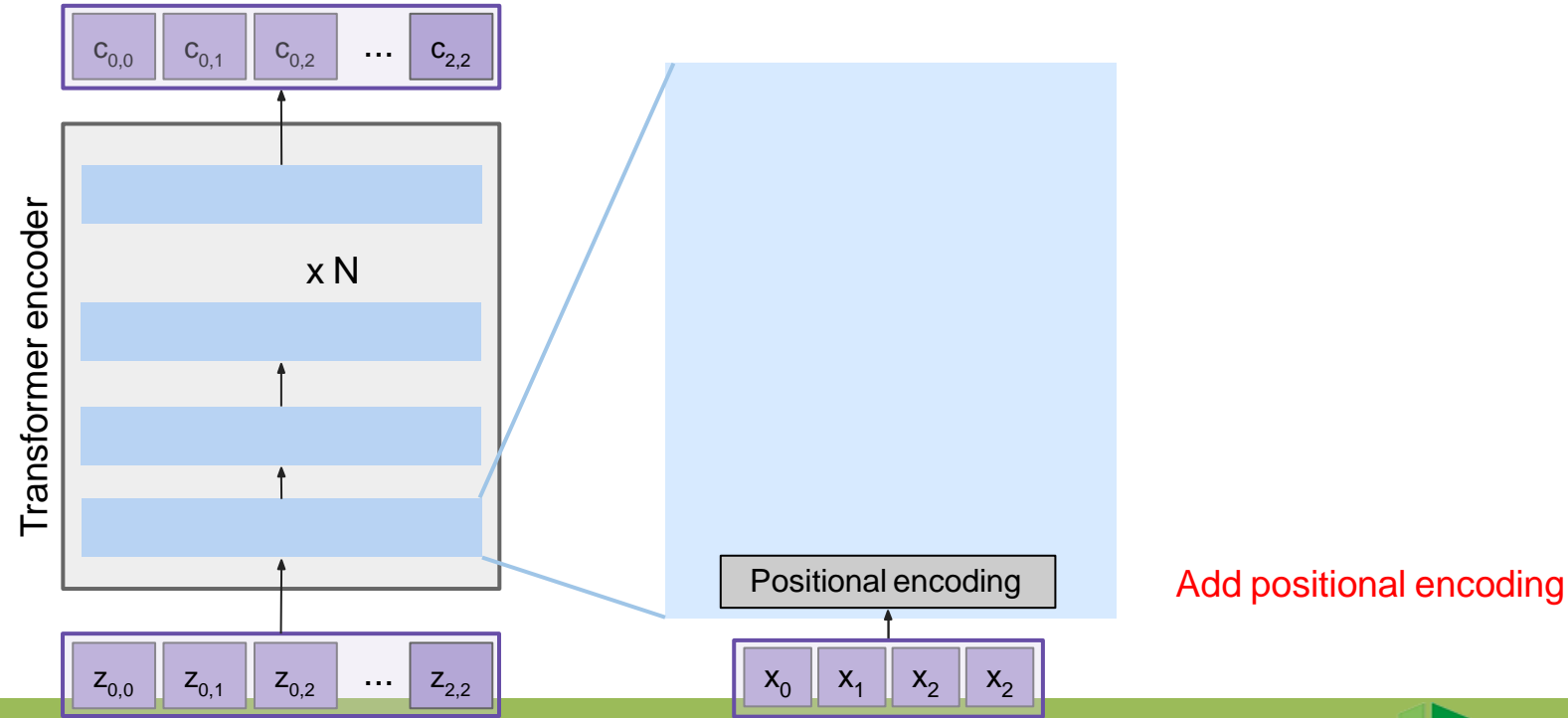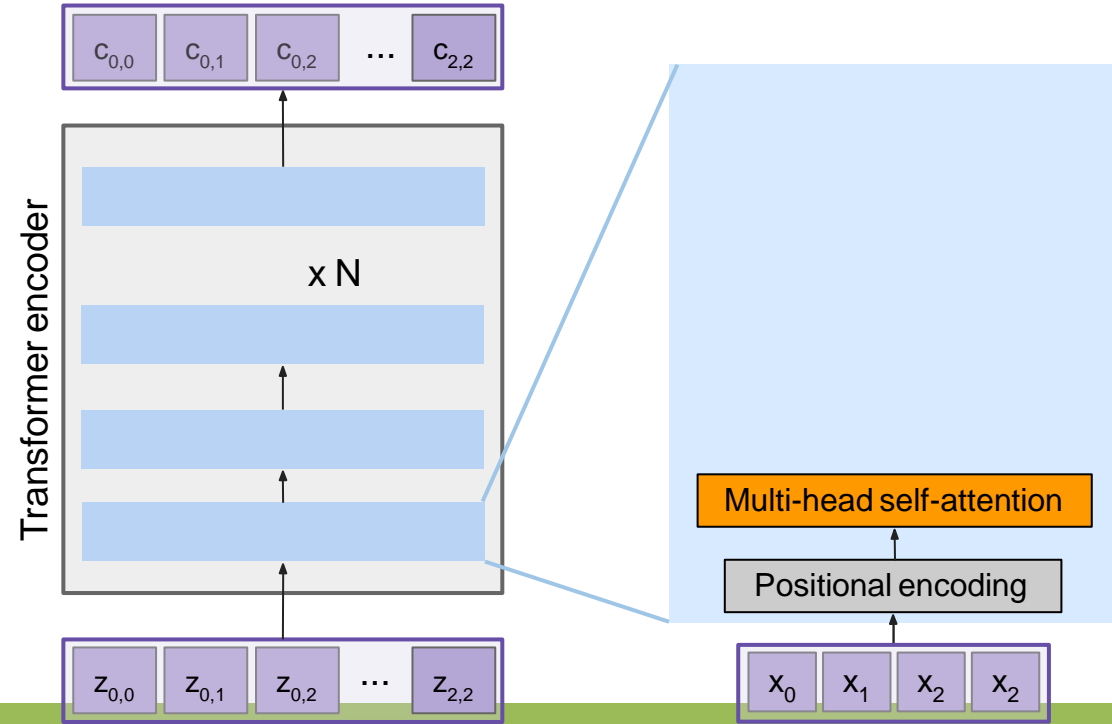
Universidad
Popular del Cesar

The Transformer encoder block

C_{0,0} C_{0,1} C_{0,2} ... C_{2,2}

Transformer encoder

x N

Positional encoding

Add positional encoding

z_{0,0} z_{0,1} z_{0,2} ... z_{2,2}

x_0 x_1 x_2 x_2

Universidad
Popular del Cesar

The Transformer encoder block

Transformer encoder

$c_{0,0}$ $c_{0,1}$ $c_{0,2}$ … $c_{2,2}$

x N

$z_{0,0}$ $z_{0,1}$ $z_{0,2}$ … $z_{2,2}$

Multi-head self-attention

Positional encoding

x_0 x_1 x_2 x_2

Attention attends over all the vectors

Add positional encoding

Attention and Transformers

Universidad Popular del Cesar

The Transformer encoder block

Residual connection

Attention attends over all the vectors

Add positional encoding

Universidad
Popular del Cesar
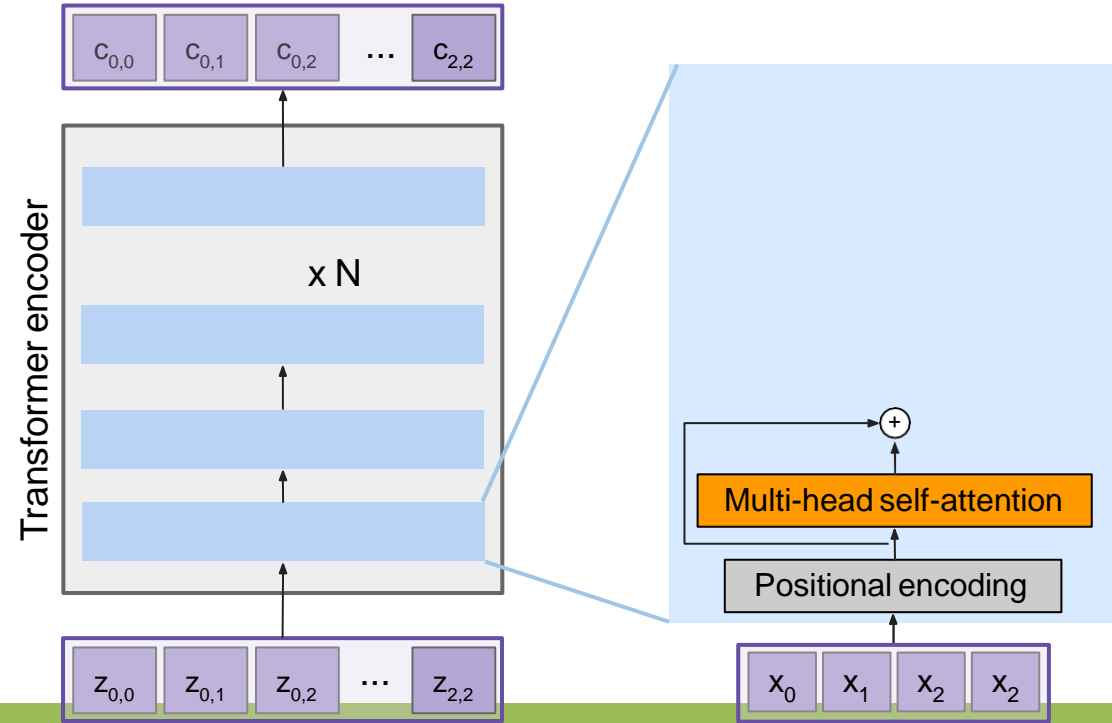
The Transformer encoder block

LayerNorm over each vector individually

Residual connection

Attention attends over all the vectors

Add positional encoding

Attention and Transformers

Universidad
Popular del Cesar
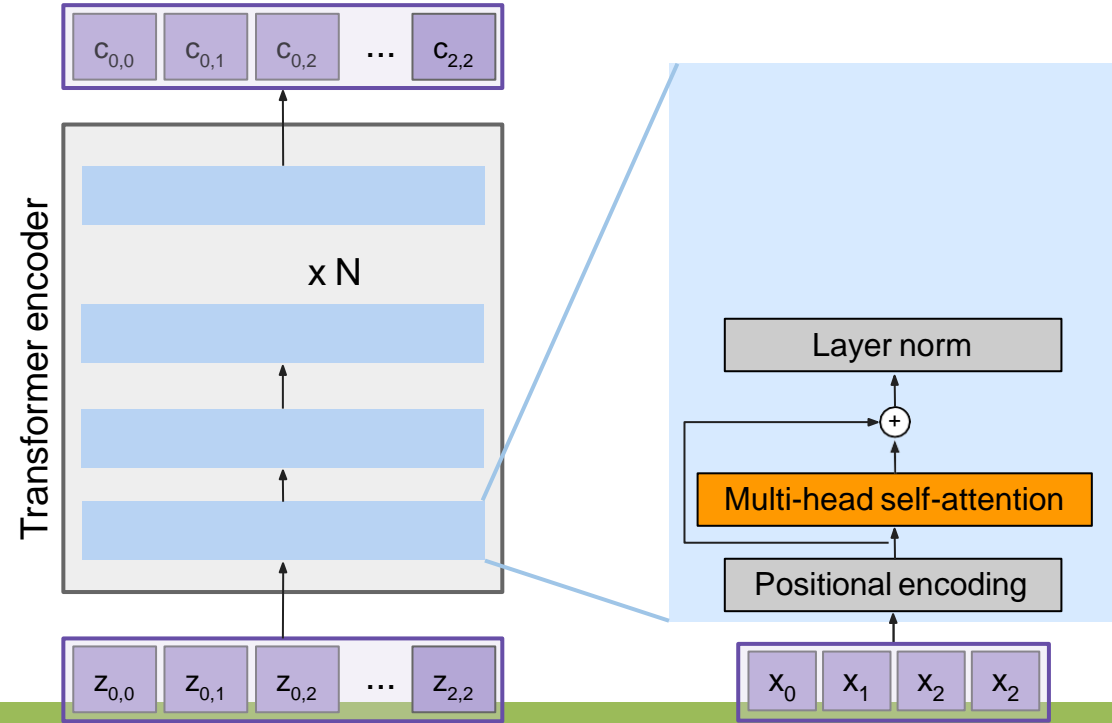
The Transformer encoder block

MLP over each vector individually

LayerNorm over each vector individually

Residual connection

Attention attends over all the vectors

Add positional encoding

Attention and Transformers

Universidad
Popular del Cesar
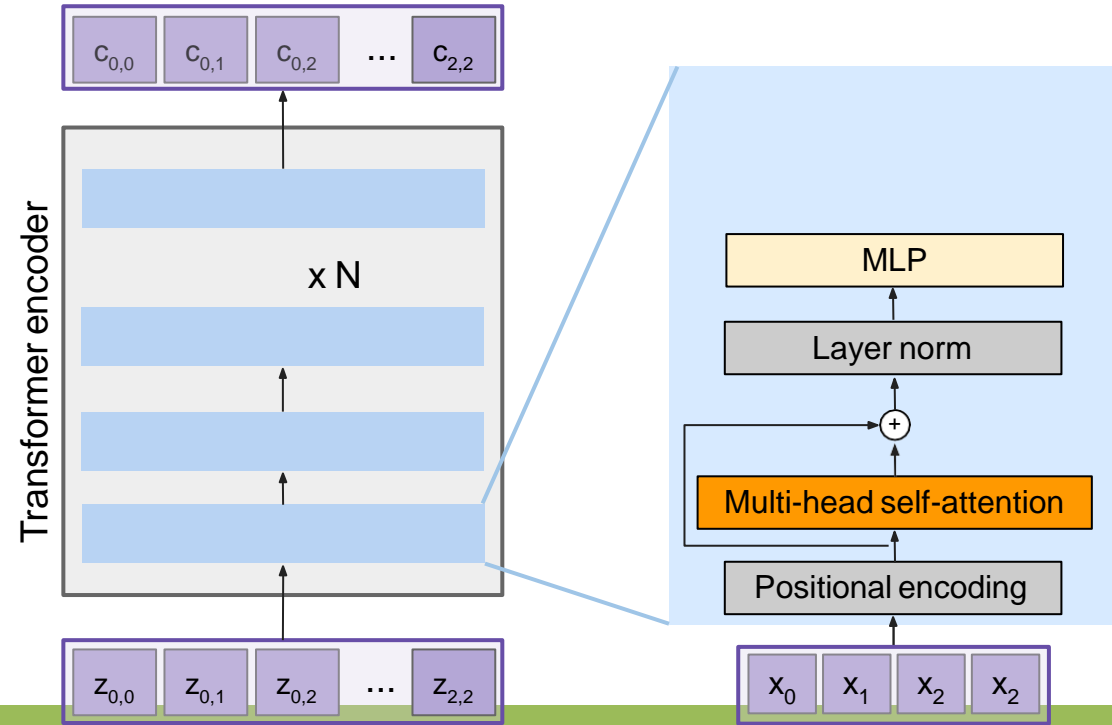
The Transformer encoder block
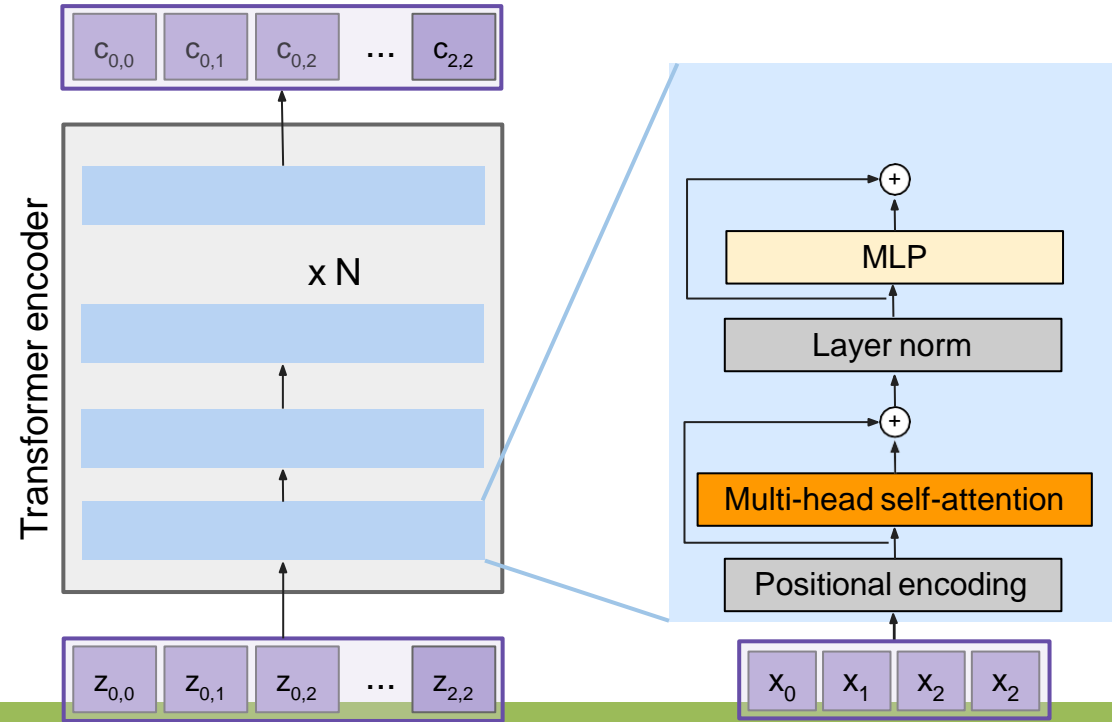
Residual connection

MLP over each vector individually

LayerNorm over each vector individually

Residual connection

Attention attends over all the vectors

Add positional encoding

Attention and Transformers

Vaswani et al. "Attention is all you need", NeurIPS 2017

Universidad Popular del Cesar

The Transformer encoder block
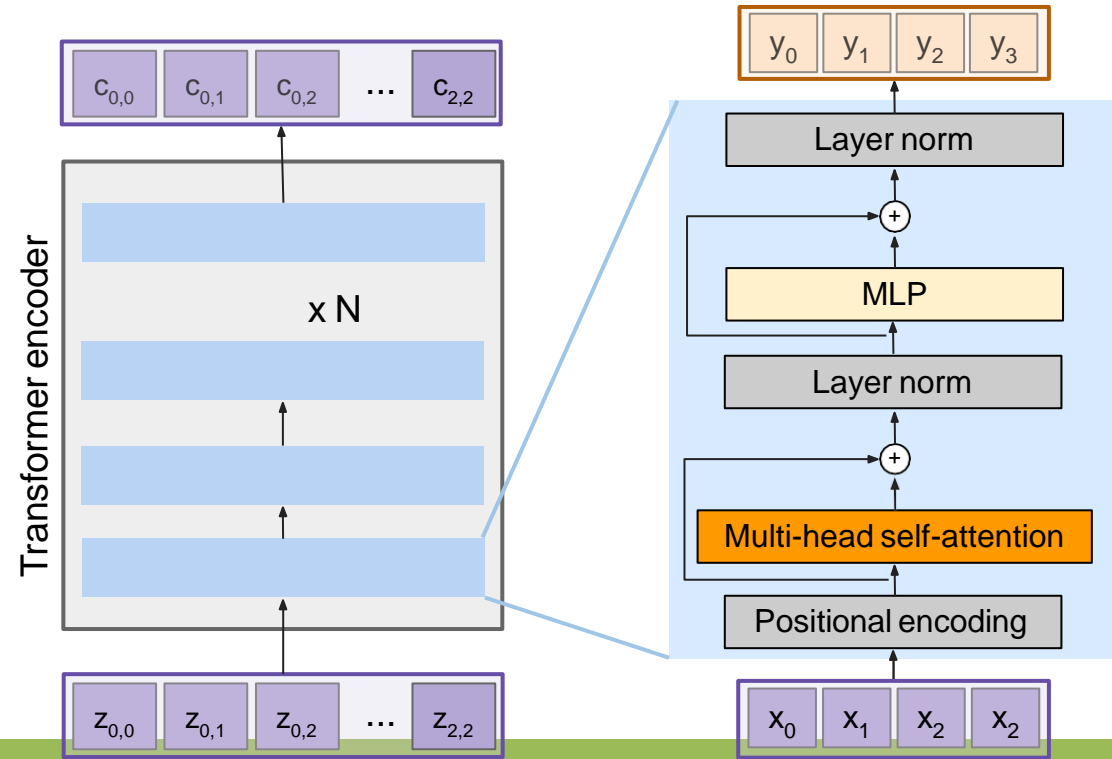
Transformer Encoder Block:

Inputs: Set of vectors **x**
Outputs: Set of vectors **y**

Self-attention is the only interaction between vectors.

Layer norm and MLP operate independently per vector.

Highly scalable, highly parallelizable, but high memory usage.

Attention and Transformers

Vaswani et al, "Attention is all you need", NeurIPS 2017

Universidad
Popular del Cesar

The Transformer decoder block
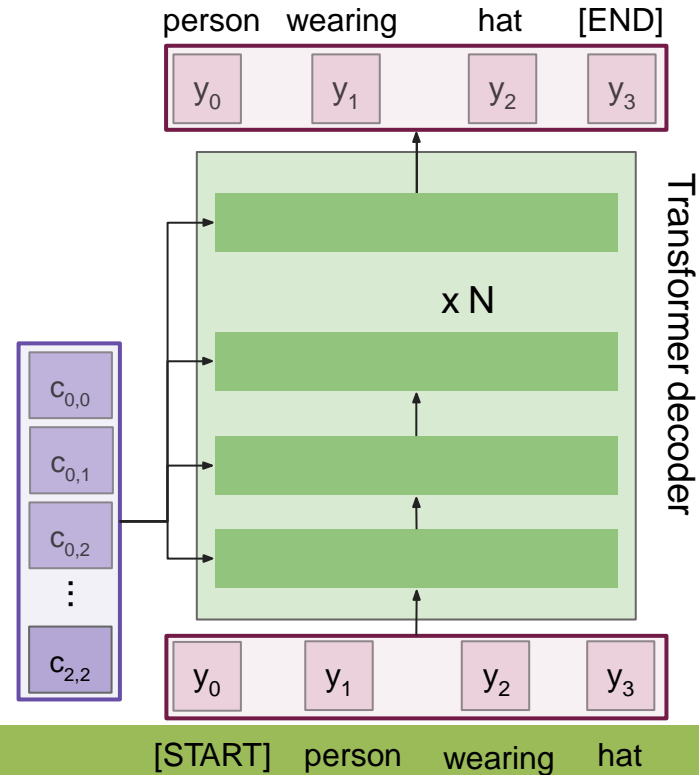
Made up of N decoder blocks.

In vaswani et al. $N = 6$, $D_q = 512$

Vaswani et al. "Attention is all you need" NeurIPS 2017

The Transformer decoder block

person wearing hat [END]

y_0 y_1 y_2 y_3

x N

Transformer decoder

$c_{0,0}$
$c_{0,1}$
$c_{0,2}$
⋮
$c_{2,2}$

y_0 y_1 y_2 y_3

[START] person wearing hat

y_0 y_1 y_2 y_3

FC

$c_{0,0}$
$c_{0,1}$
$c_{0,2}$
⋮
$c_{2,2}$

x_0 x_1 x_2 x_3
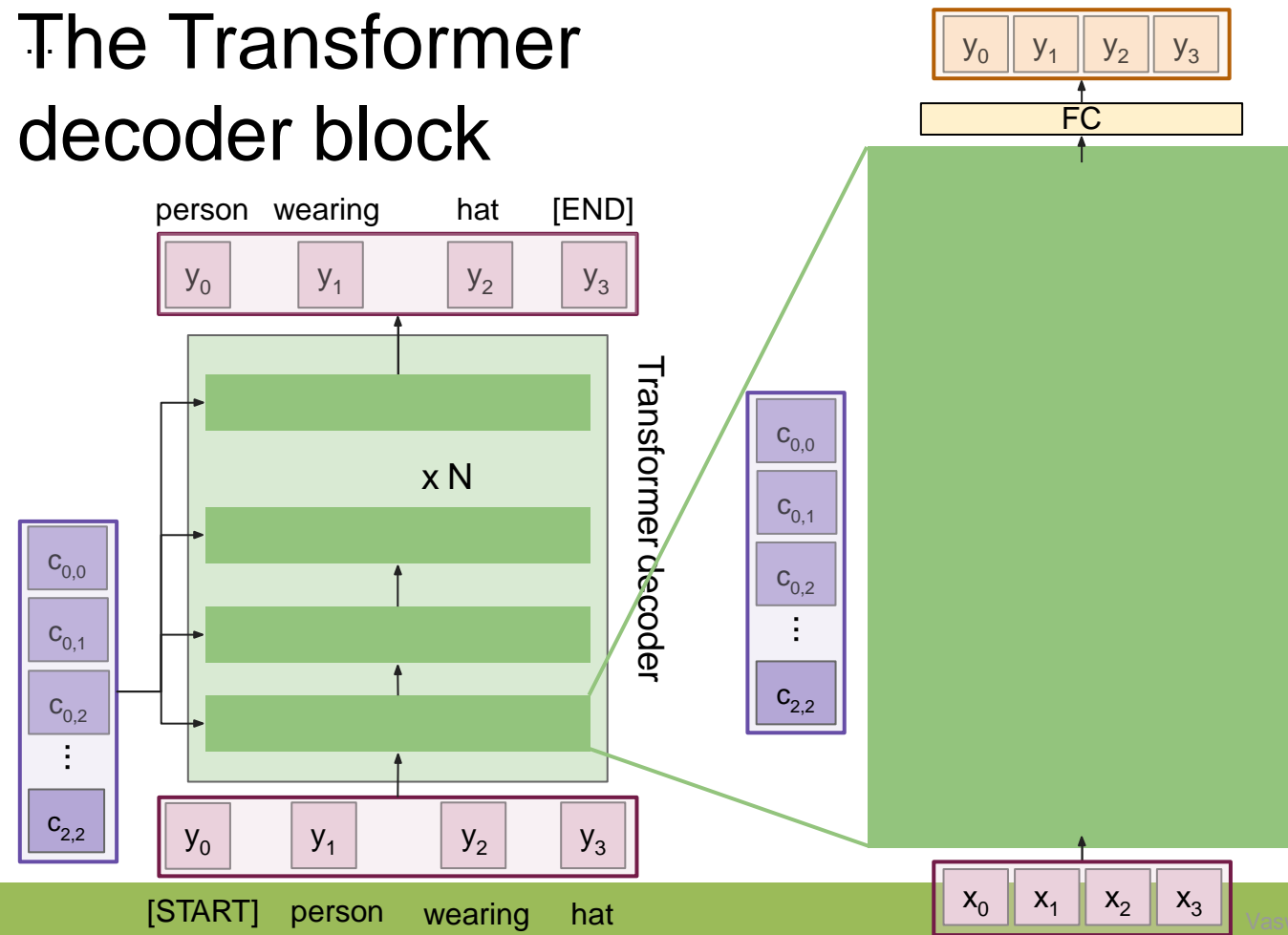
Let's dive into the transformer decoder block

Vaswani et al, "Attention is all you need", NeurIPS 2017

Universidad
Popular del Cesar

The Transformer Decoder block
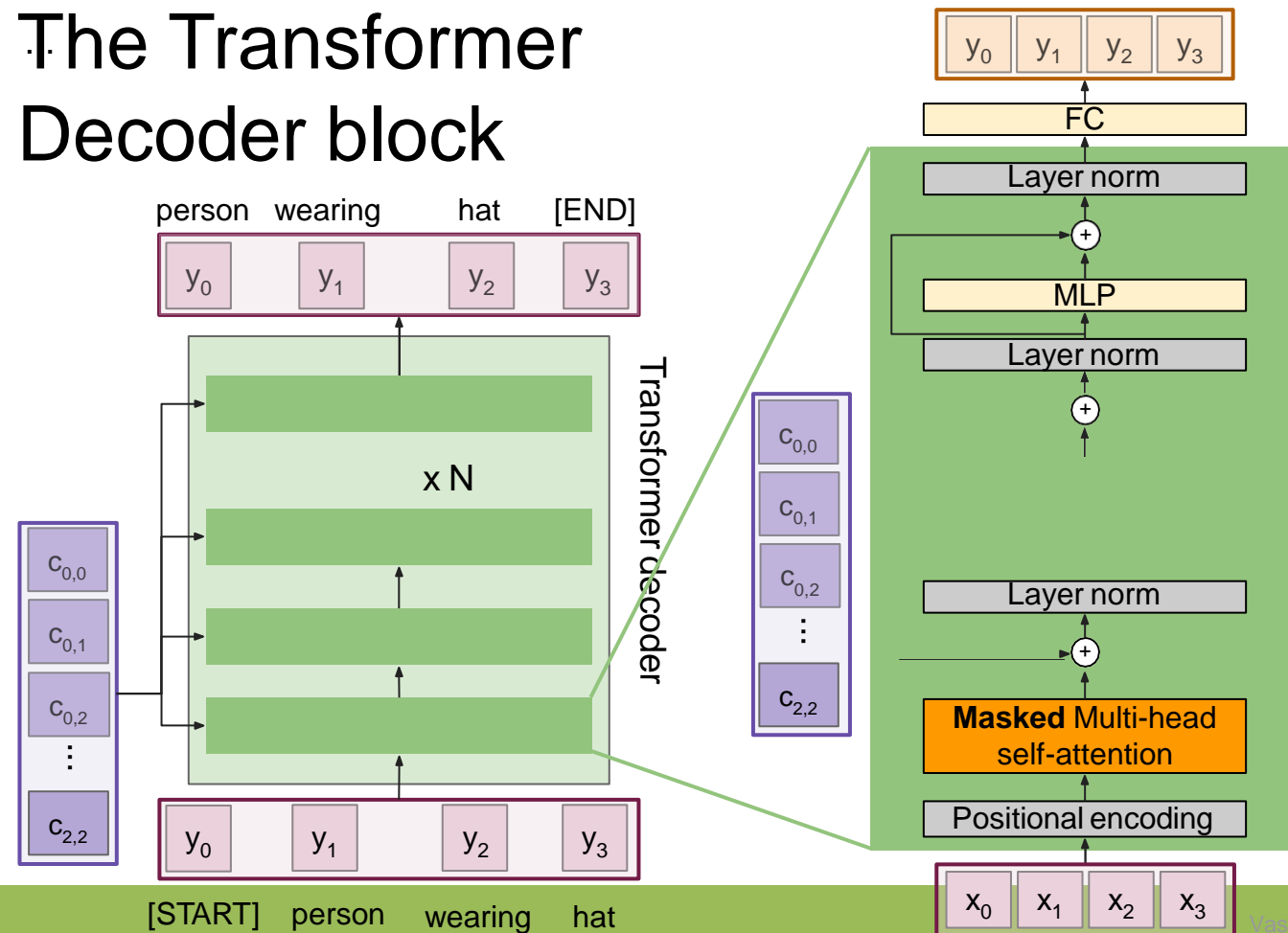
Most of the network is the same the transformer encoder.

Universidad Popular del Cesar

The Transformer Decoder block
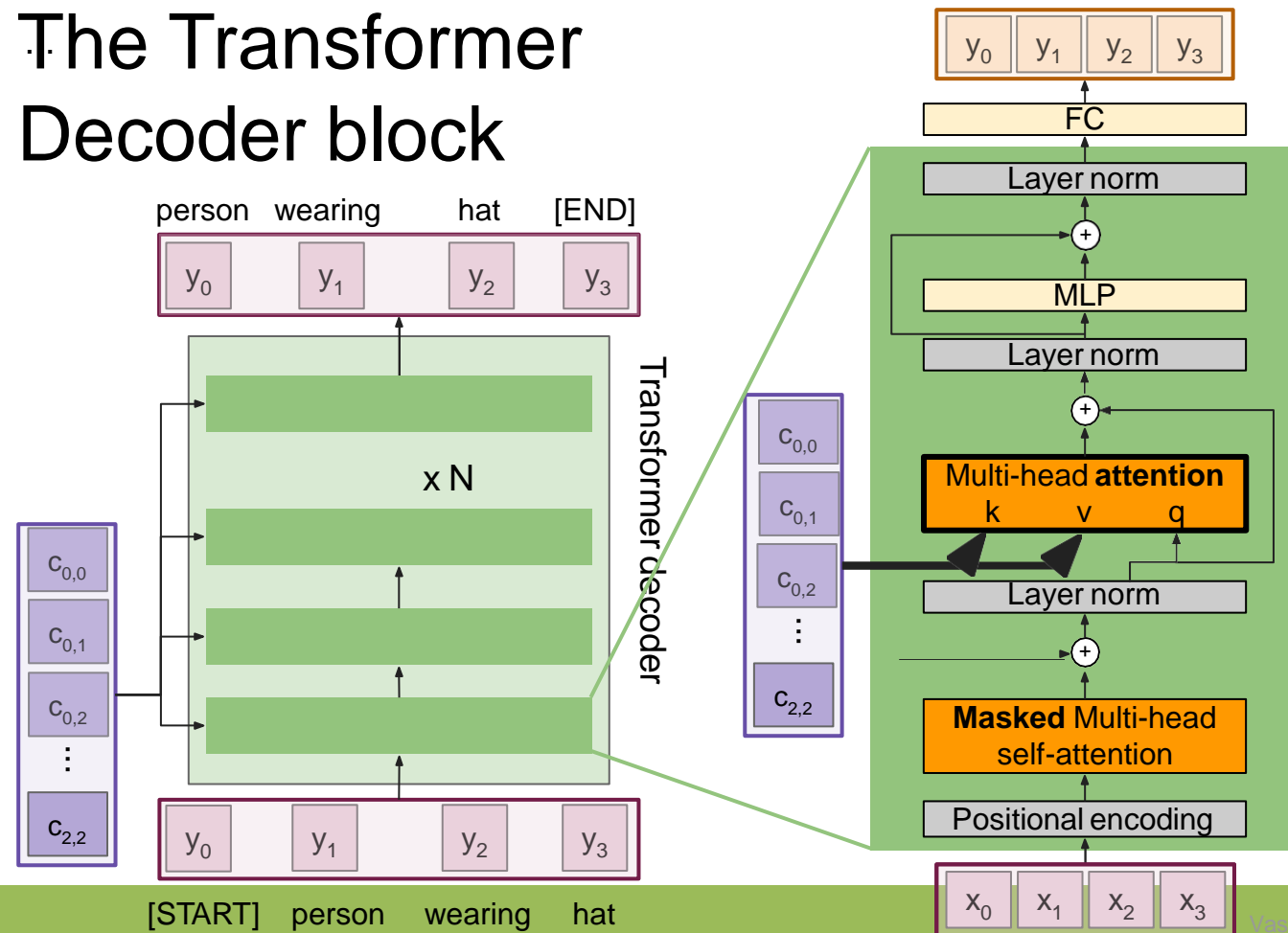
Multi-head attention block attends over the transformer encoder outputs.

For image captioning, this is how we inject image features into the decoder.

Vaswani et al, "Attention is all you need", NeurIPS 2017

Universidad
Popular del Cesar

The Transformer Decoder block
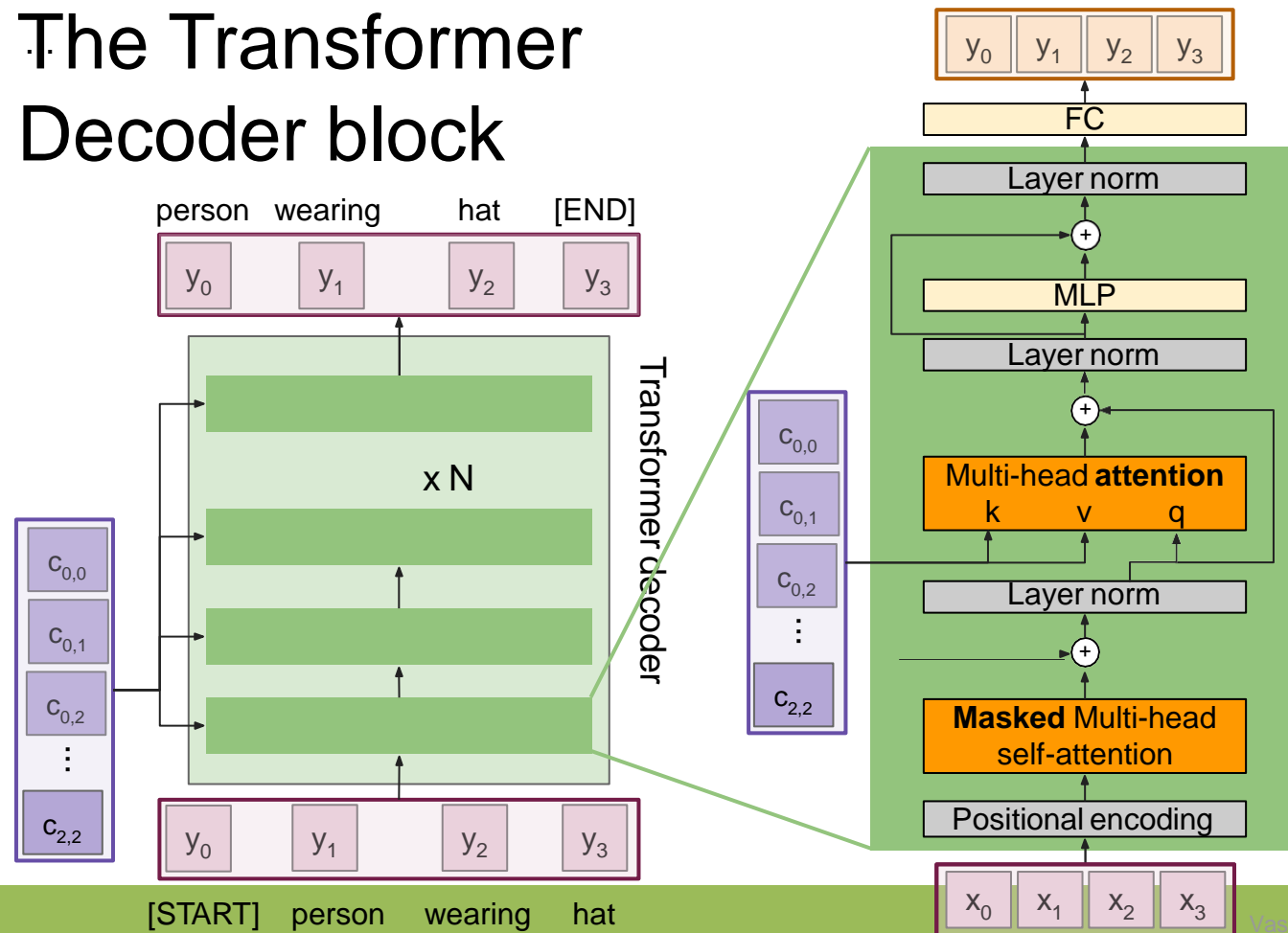
Transformer Decoder Block:

Inputs: Set of vectors **x** and Set of context vectors **c.**
Outputs: Set of vectors **y.**

Masked Self-attention only interacts with past inputs.

Multi-head attention block is NOT self-attention. It attends over encoder outputs.

Highly scalable, highly parallelizable, but high memory usage.

Vaswani at al., "Attention is all you need", NeurIPS 2017

Universidad **Popular del Cesar**

Image Captioning using transformers

- **No recurrence at all**

Extract spatial features from a pretrained CNN

Features: H x W x D
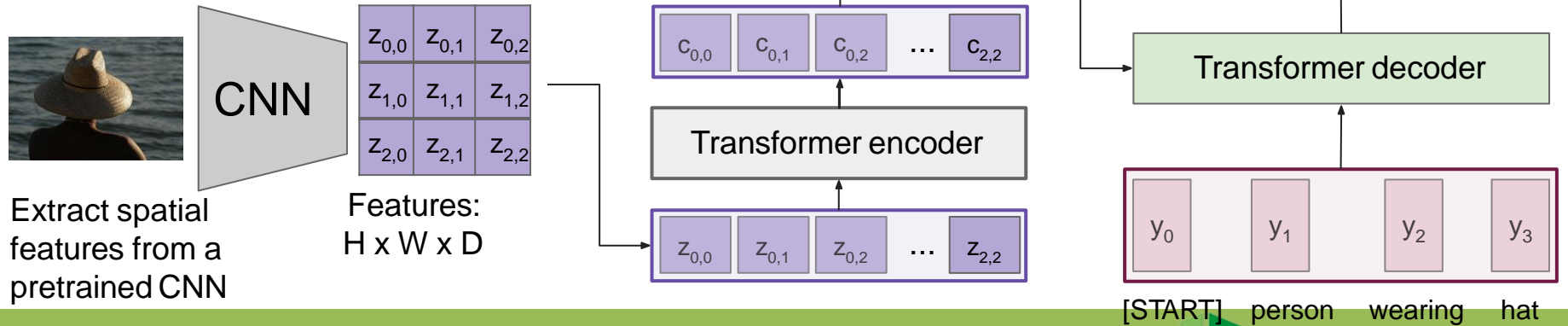
Image Captioning using transformers

- **Perhaps we don't need convolutions at all?**

Extract spatial features from a pretrained CNN

Features: H x W x D

Universidad Popular del Cesar
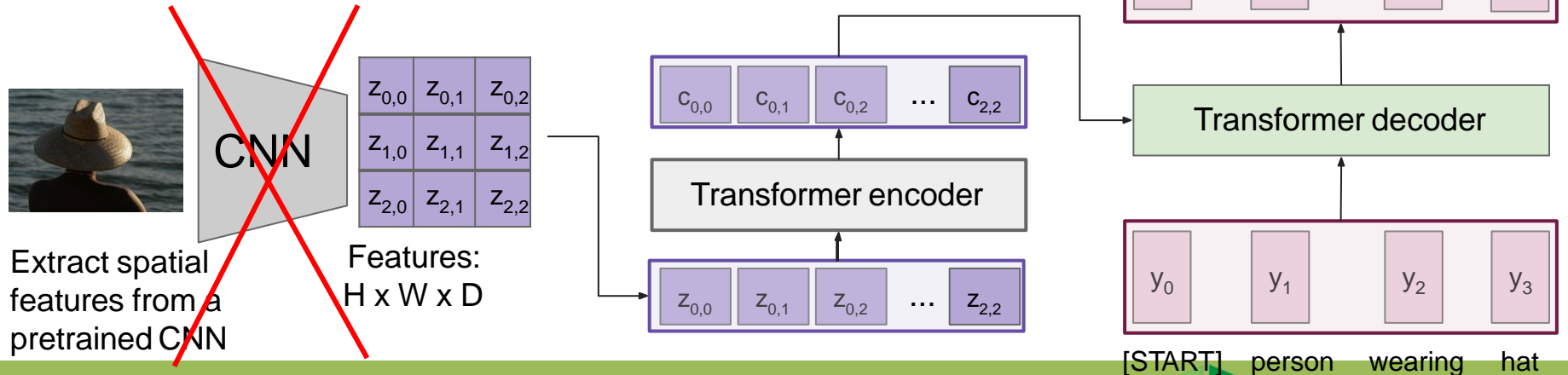
Image Captioning using ONLY transformers

- **Transformers from pixels to language**

person   wearing   hat   [END]

y_1 y_2 y_3 y_4

$c_{0,0}$ $c_{0,1}$ $c_{0,2}$ … $c_{2,2}$

Transformer encoder

Transformer decoder

y_0 y_1 y_2 y_3

[START]   person   wearing   hat
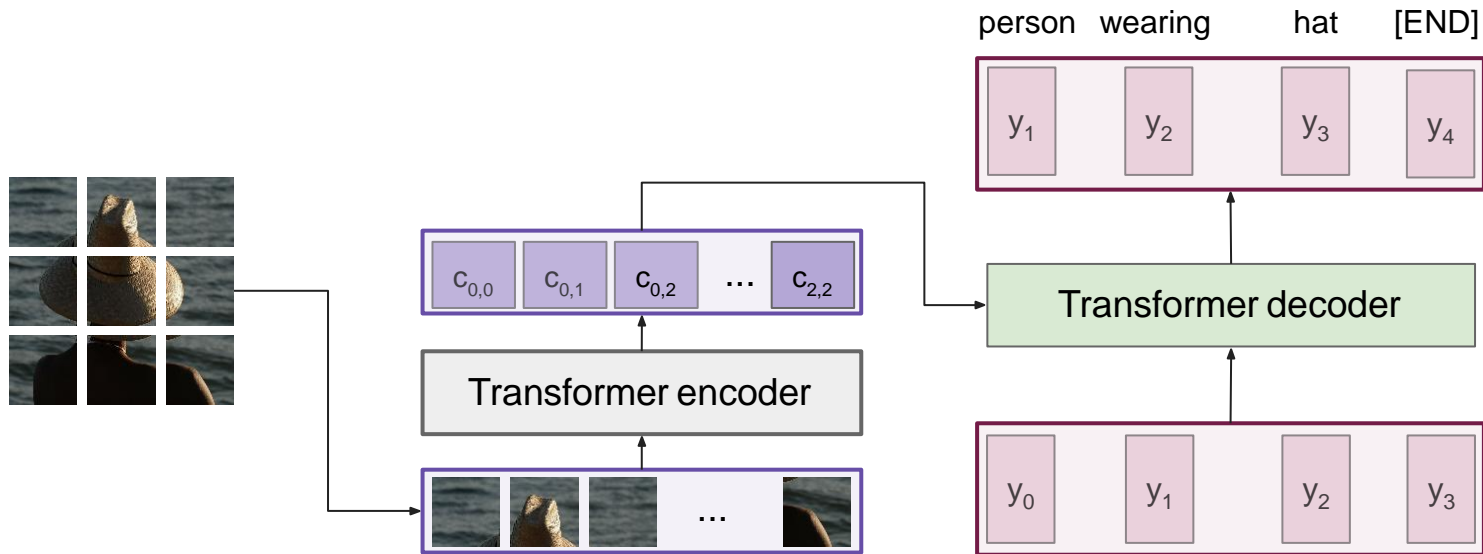
Dosovitskiy et al, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", ArXiv 2020
Colab link to an implementation of vision transformers

Attention and Transformers

Universidad
Popular del Cesar

Vision Transformers vs. ResNets
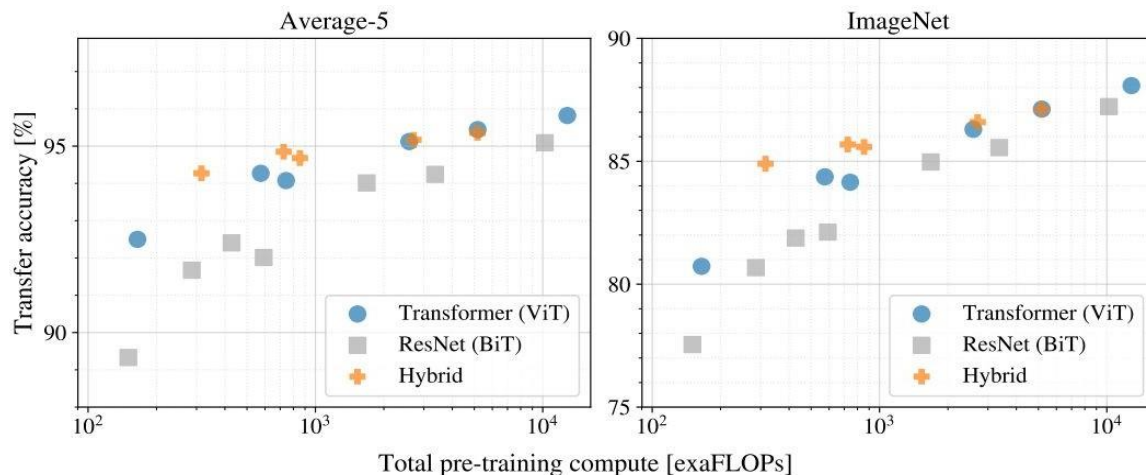
Figure 5: Performance versus cost for different architectures: Vision Transformers, ResNets, and hybrids. Vision Transformers generally outperform ResNets with the same computational budget. Hybrids improve upon pure Transformers for smaller model sizes, but the gap vanishes for larger models.
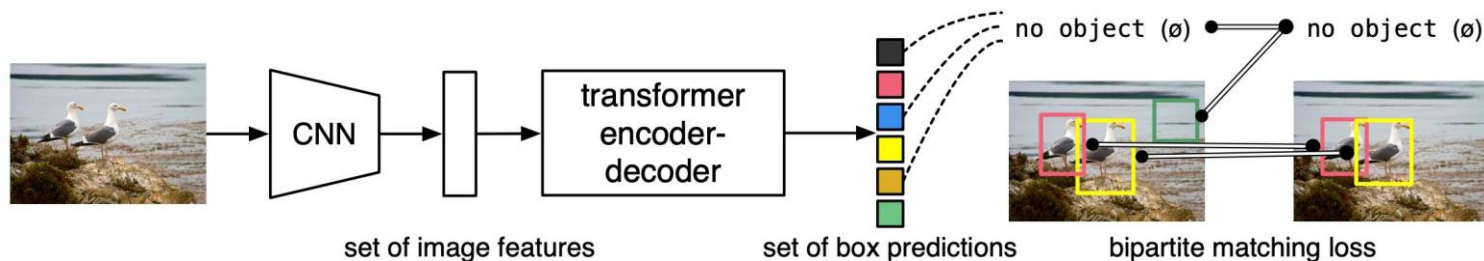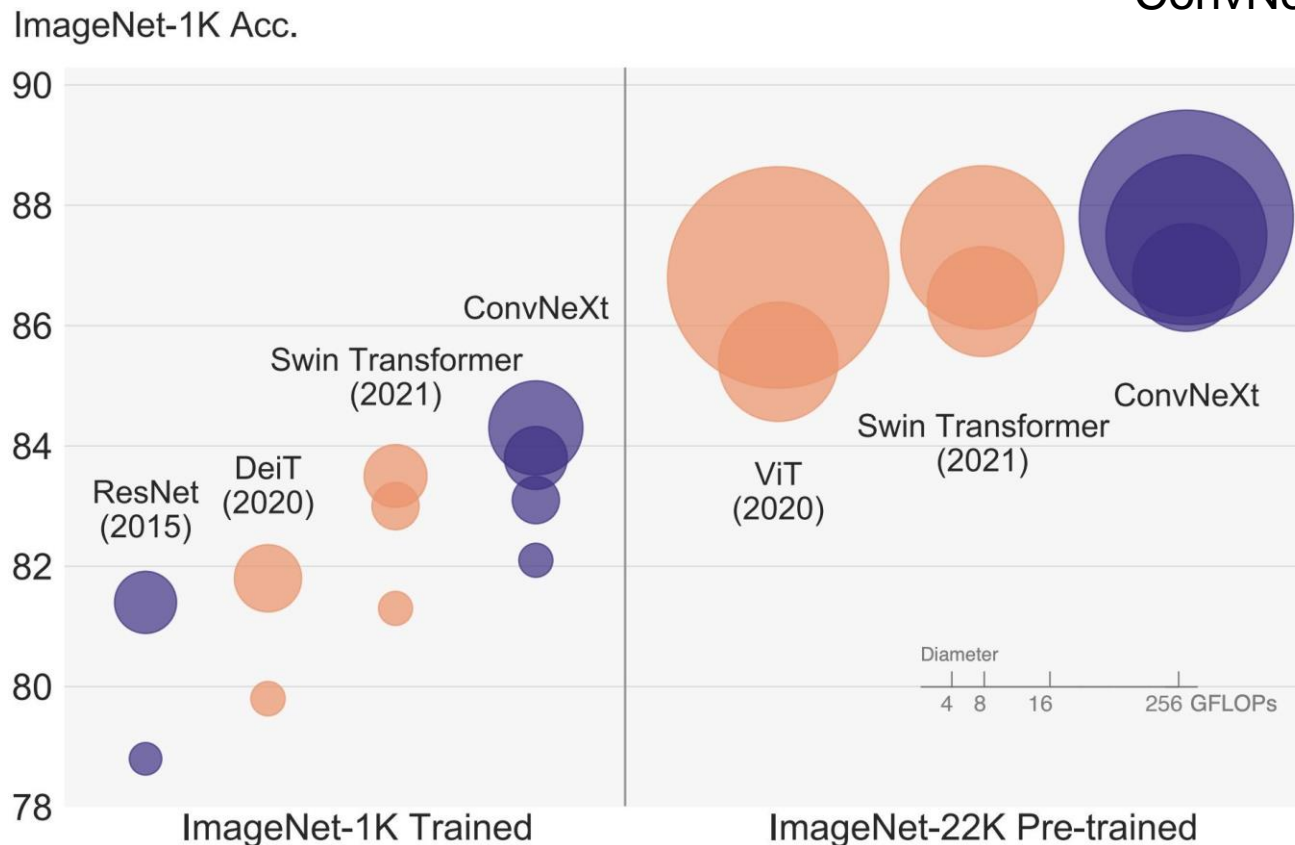
Attention and Transformers

Vision Transformers
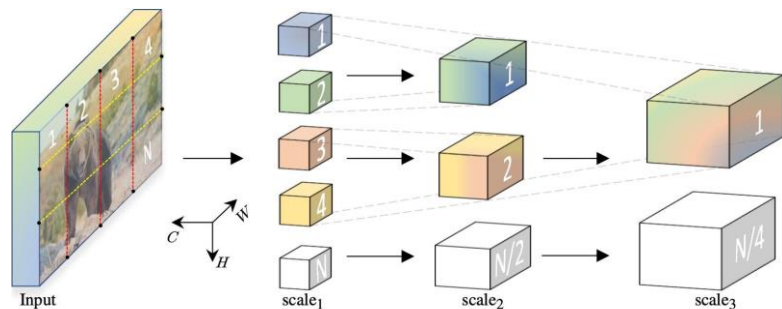
Fan et al, "Multiscale Vision Transformers", ICCV 2021
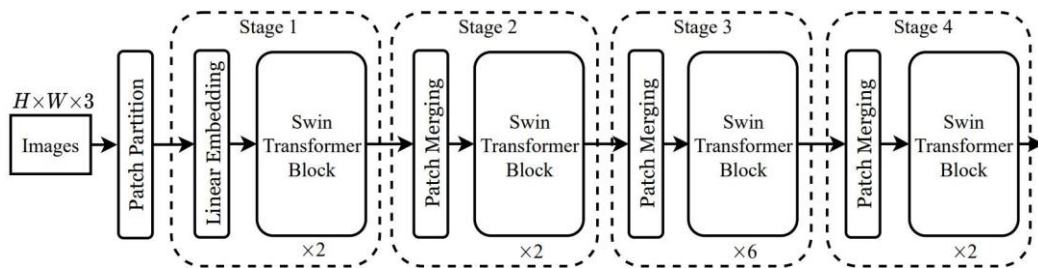
Liu et al, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows", CVPR 2021
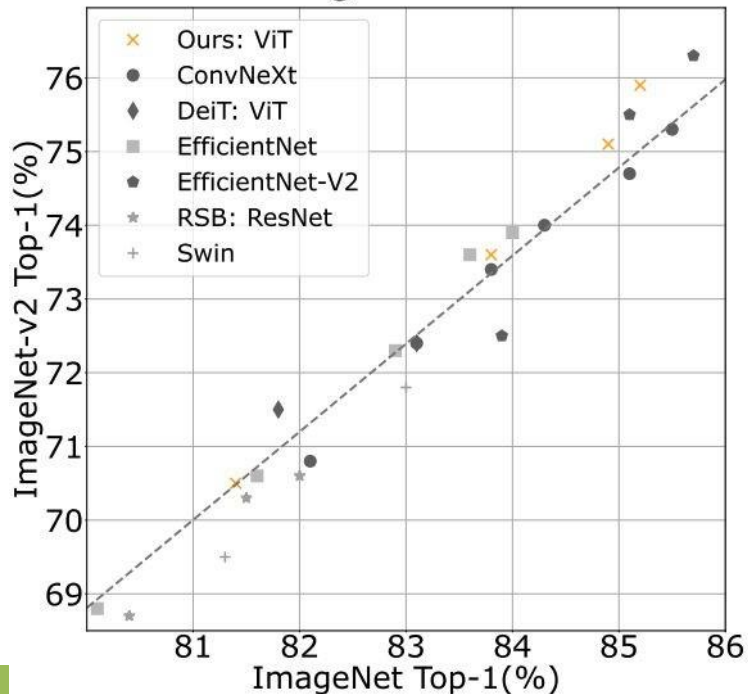

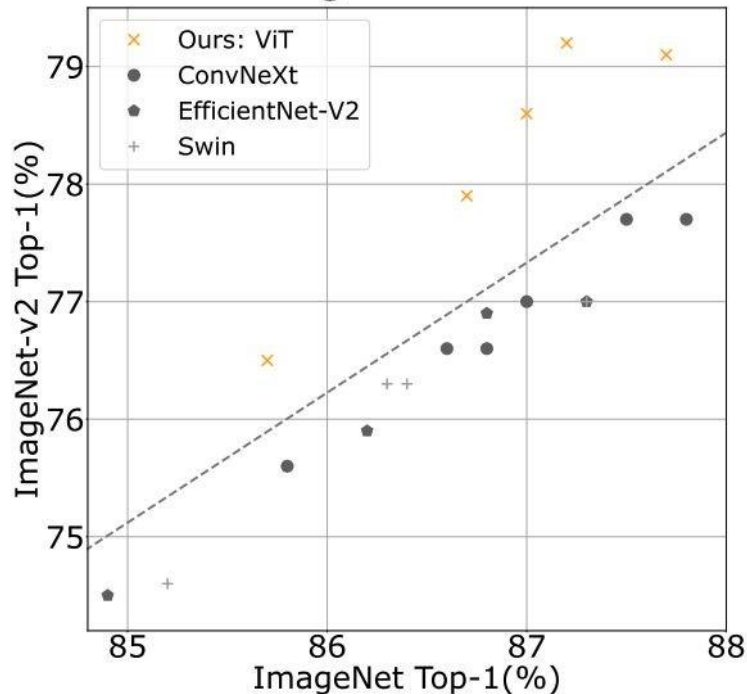Carion et al, "End-to-End Object Detection with Transformers", ECCV 2020

Universidad **Popular del Cesar**

Attention and Transformers

DeiT III: Revenge of the ViT

Hugo Touvron[*,†]   Matthieu Cord[†]   Hervé Jégou[*]

ImageNet-1k

ImageNet-21k

Summary

- Adding **attention** to RNNs allows them to "attend" to different parts of the input at every time step
- The **general attention layer** is a new type of layer that can be used to design new neural network architectures
- **Transformers** are a type of layer that uses **self-attention** and layer norm.
  - It is highly **scalable** and highly **parallelizable**
  - **Faster** training, **larger** models, **better** performance across vision and language tasks
  - They are quickly replacing RNNs, LSTMs, and may(?) even replace convolutions.

Universidad
Popular del Cesar

Next time: Video Understanding