Transformers (1)

2022/03/07

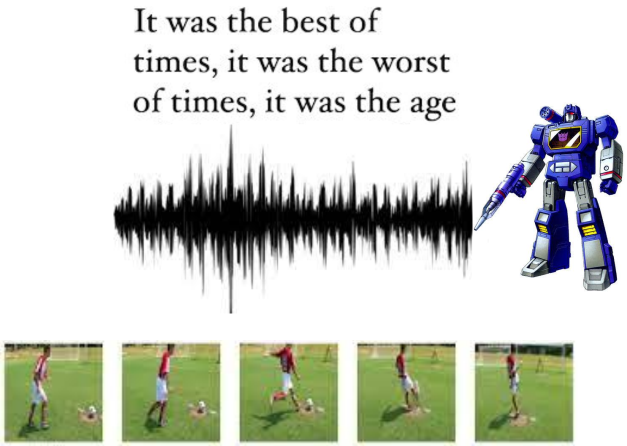
CS 182/282A Lecture 12

Today's lecture

- This week, we will learn about the **transformer** neural network architecture
    - Today: the setup and basics
    - Next time (and next week as well): transformers in action
- This will be the last neural network architecture we cover
    - There are some other interesting architectures, e.g., for graph data
    - You should be able to learn new/other architectures pretty quickly now
- If you took CS 189 last semester, the slides this week may look pretty familiar
    - In fact, they're still in last semester's format!
    - Some new slides added in, but mostly it is last semester's slide decks

Setup

features $\mathbf{x}$



- sequential data
- may be variable length

label $\mathbf{y}$

could correspond to…
- sentiment analysis,
  translation to another language,
  …
- audio transcription,
  speaker identification,
  …
- activity identification,
  video captioning,
  …

or there could be no label!
- unsupervised learning /
  generative modeling

model

- Markov / n-gram models,
  hidden Markov models
- embedding / clustering based methods
- recurrent neural networks (RNNs)
  - GRUs, LSTMs, …
- convolutions
- transformers

Why transformers?

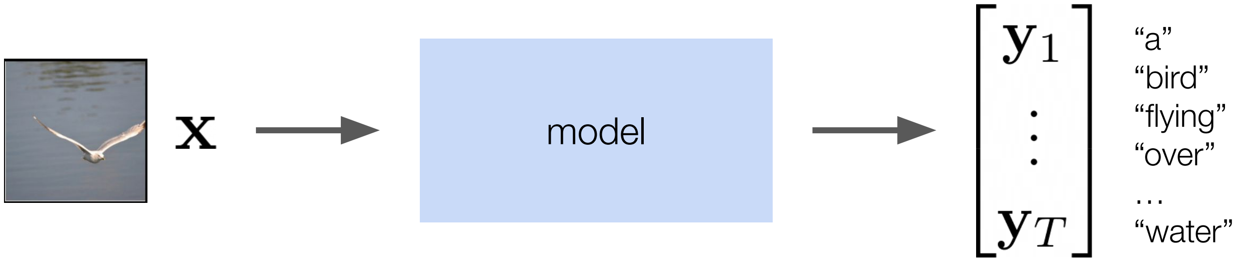
- Massively influential in the last 4-5 years
    - Outcompeting other deep architectures in a number of domains, e.g., RNNs in language modeling and convolutional networks in vision tasks
    - Other state-of-the-art models, though not transformers, also use *attention*
    - You might say they have been… transformative
- The backbone of models including BERT and GPT
    - Dubbed by Stanford as “foundation models”: https://crfm.stanford.edu/
- Surprisingly not that hard to understand
    - Once the right background knowledge is in place

Attention

Setup for attention

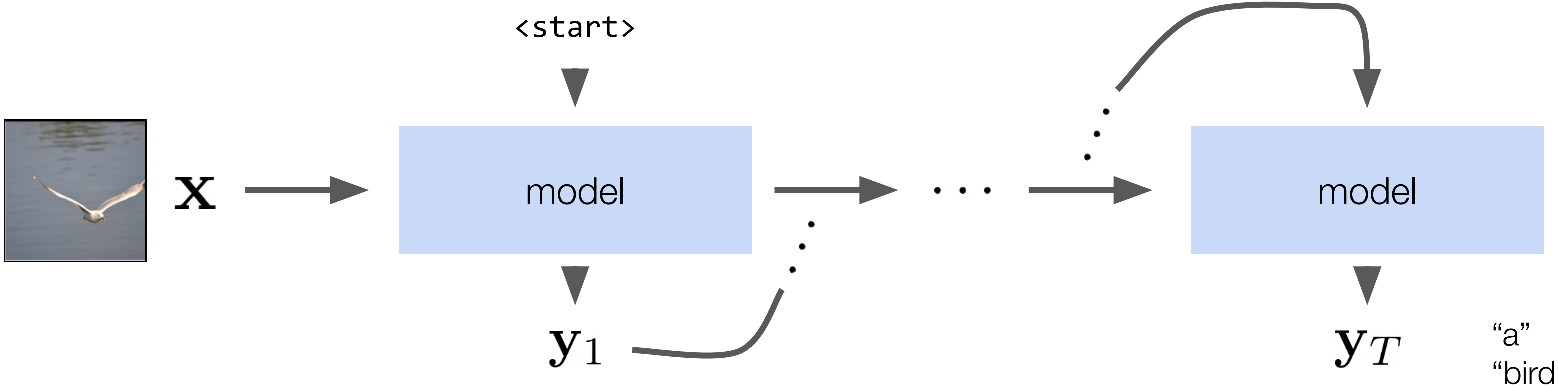
Originally formulated for tasks with sequential *outputs*

- E.g., translating from one language to another, captioning an image, …
- The features may or may not be sequential



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

Setup for attention



"a"
"bird"
"flying"
"over"
…
"water"

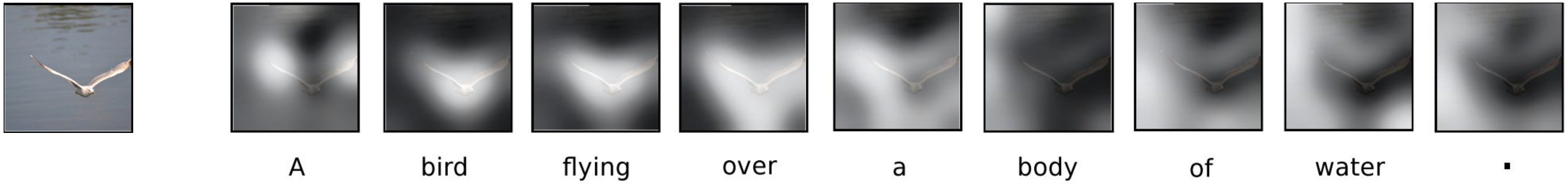
The model generates the output "one step at a time"

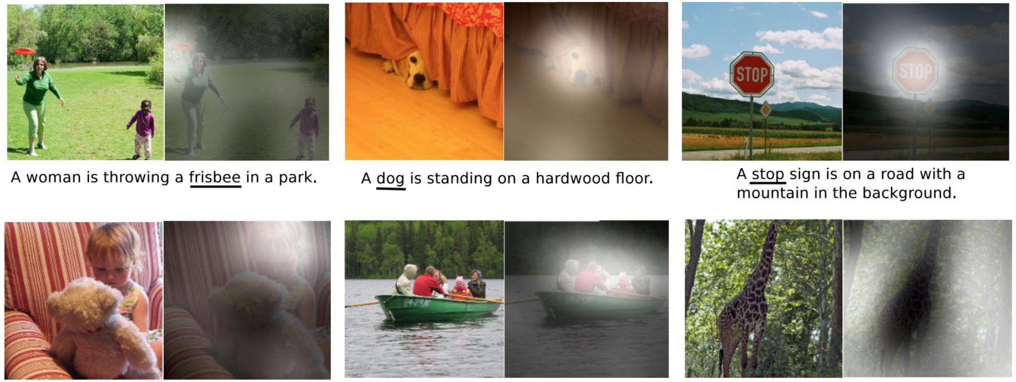
However, the model needs to know what is has generated so far

- We can do this via *autoregressive* generation from an RNN, as we learned previously

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

Motivation / intuition





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.

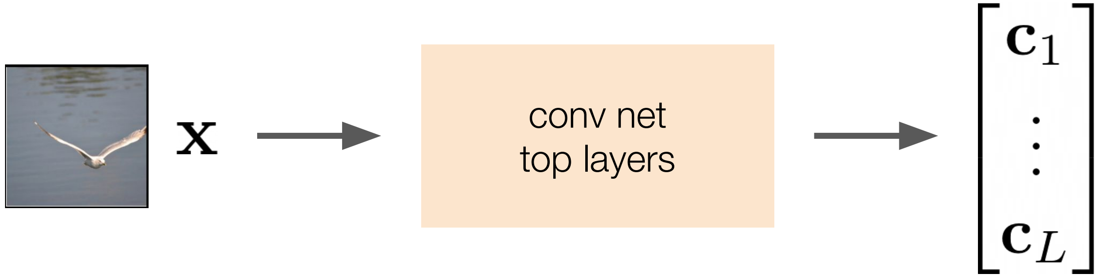
A giraffe standing in a forest with trees in the background.

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

What do we attend over?

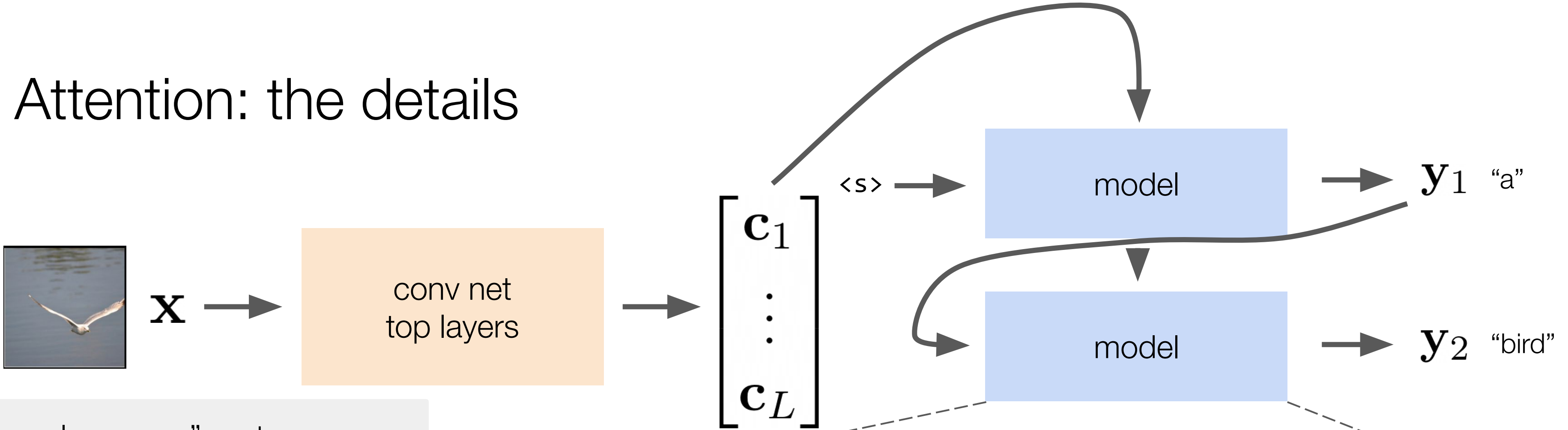
It depends on the task. Some examples:

- If the features are words, we can attend over them directly
- What if the features are pixels in an image?



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

Attention: the details



“key-value-query” system:

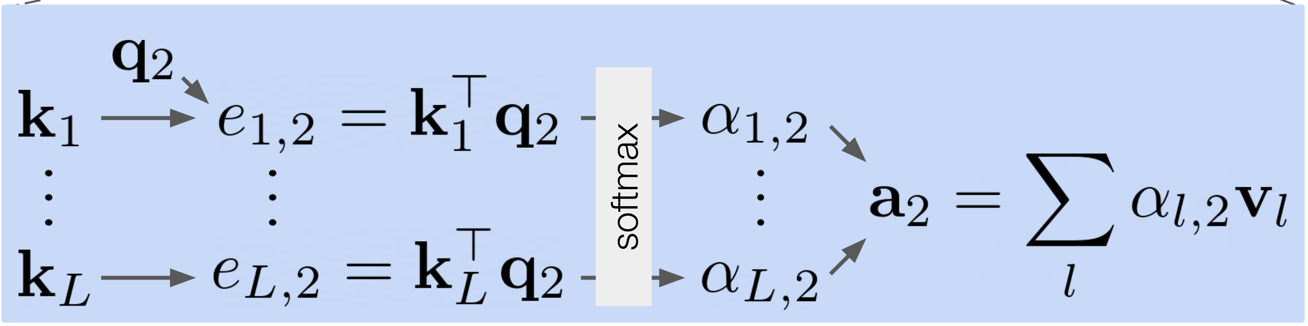
$$\mathbf{q}_t = q\left(\text{model info from prev step}\right)$$

$$\mathbf{k}_l = k(\mathbf{c}_l)$$

$$\mathbf{v}_l = v(\mathbf{c}_l)$$

These functions are learned and can be, e.g., simple linear layers

(just the attention part, output generation not shown)



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

Self-attention: the building block of transformers

The goal of self-attention is to handle sequential features as the input

Think of it as a neural network layer that allows for processing the whole sequence

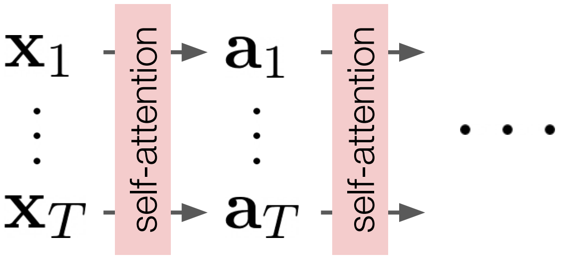
"key-value-query" system:

$$\mathbf{q}_t = q(\mathbf{x}_t)$$
$$\mathbf{k}_t = k(\mathbf{x}_t)$$
$$\mathbf{v}_t = v(\mathbf{x}_t)$$

These functions are learned and can be, e.g., simple linear layers

$$\mathbf{k}_1 \xrightarrow{\mathbf{q}_2} e_{1,2} = \mathbf{k}_1^\top \mathbf{q}_2 \xrightarrow{\text{softmax}} \alpha_{1,2}$$
$$\vdots$$
$$\mathbf{k}_T \longrightarrow e_{T,2} = \mathbf{k}_T^\top \mathbf{q}_2 \xrightarrow{\text{softmax}} \alpha_{T,2}$$

$$\mathbf{a}_2 = \sum_t \alpha_{t,2} \mathbf{v}_t$$



Detail: scaled dot product attention

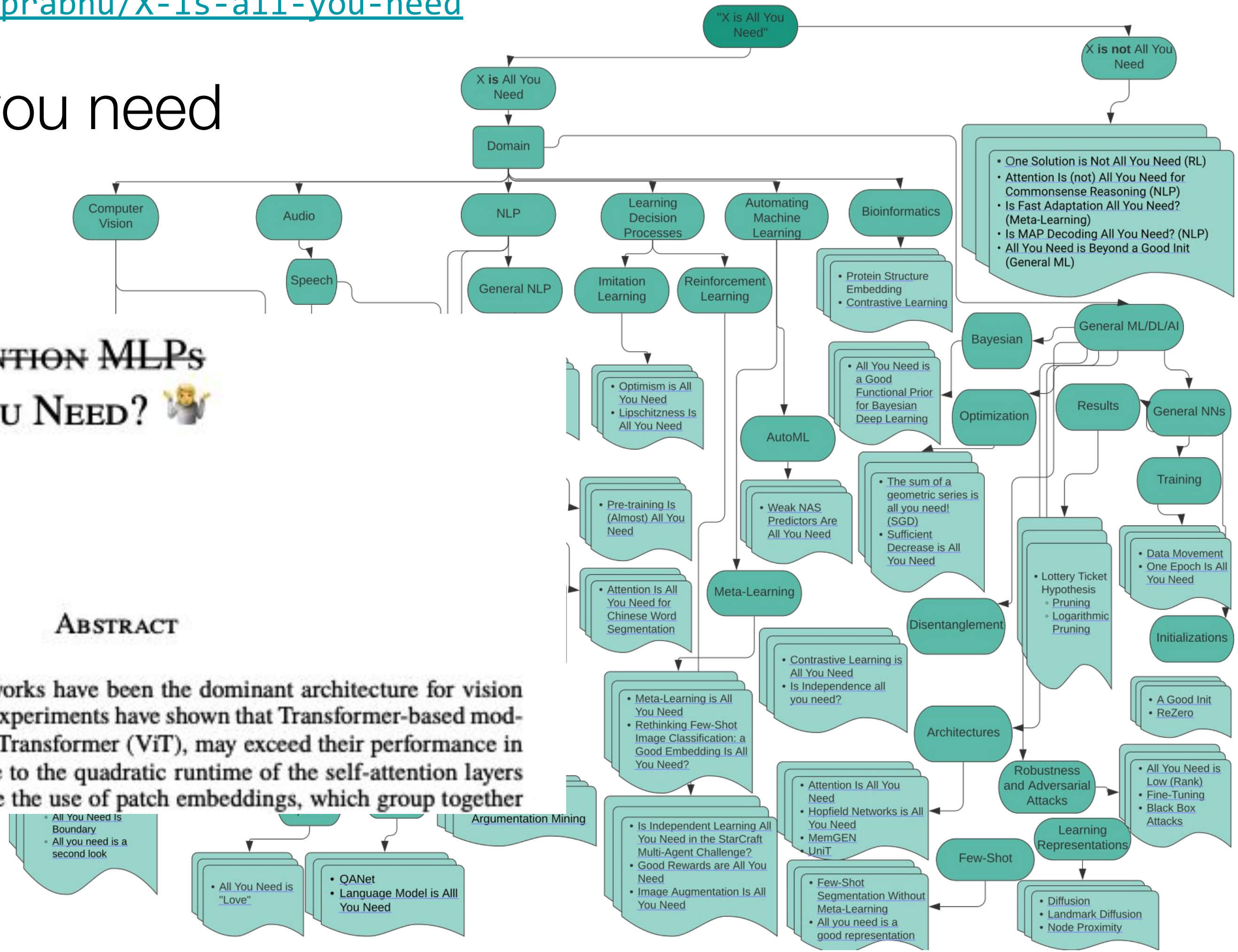
Divide each $e_{t,2}$ by $\sqrt{d}$

Transformers

Vaswani et al, “Attention is All You Need”. NIPS 2017.

https://github.com/vinayprabhu/X-is-all-you-need

Aside: X is all you need



~~Convolutions~~ ~~Attention~~ ~~MLPs~~

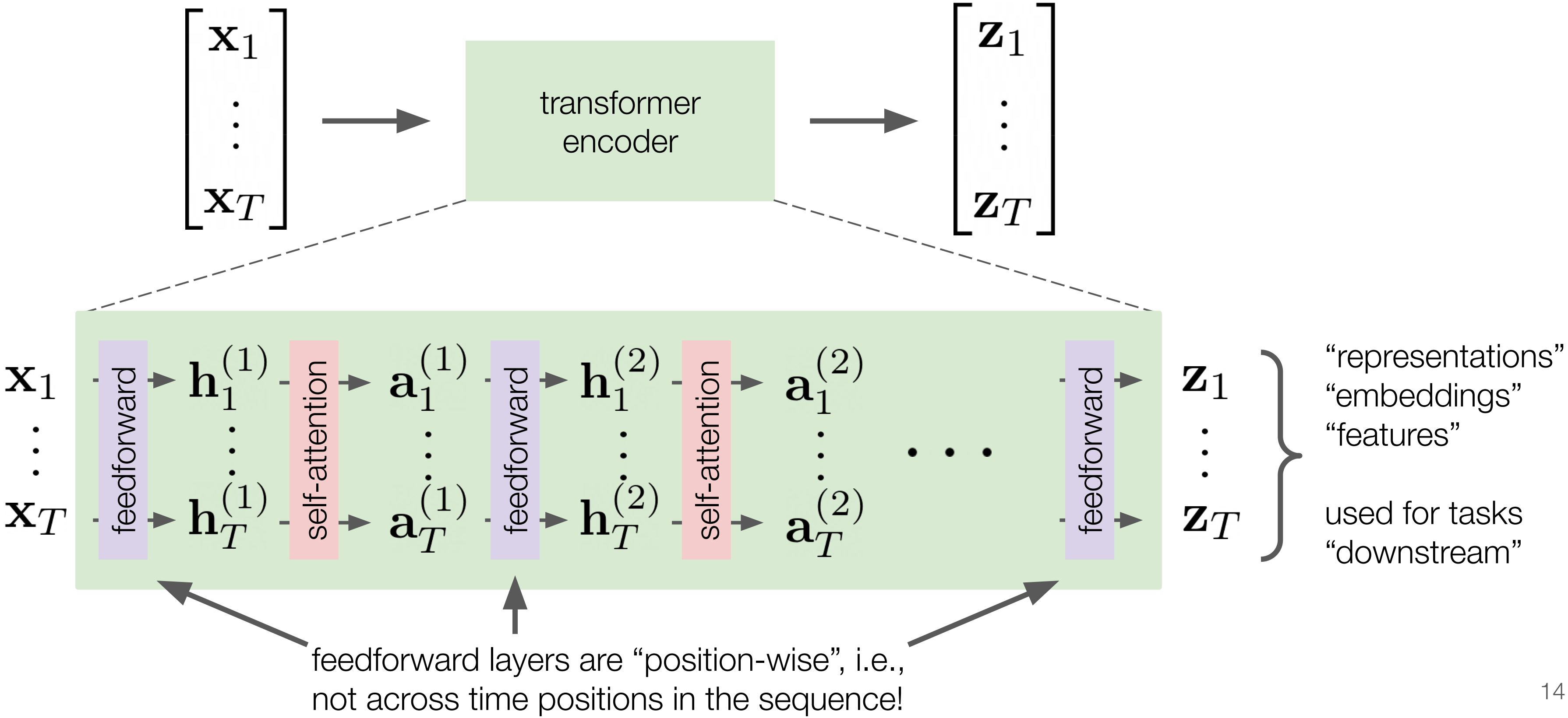
Patches Are All You Need? 🤷

Anonymous authors
Paper under double-blind review

Abstract

Although convolutional networks have been the dominant architecture for vision tasks for many years, recent experiments have shown that Transformer-based models, most notably the Vision Transformer (ViT), may exceed their performance in some settings. However, due to the quadratic runtime of the self-attention layers in Transformers, ViTs require the use of patch embeddings, which group together

Transformers for “encoding” (representation learning)



https://nlp.seas.harvard.edu/2018/04/03/attention.html

Important detail: positional encoding

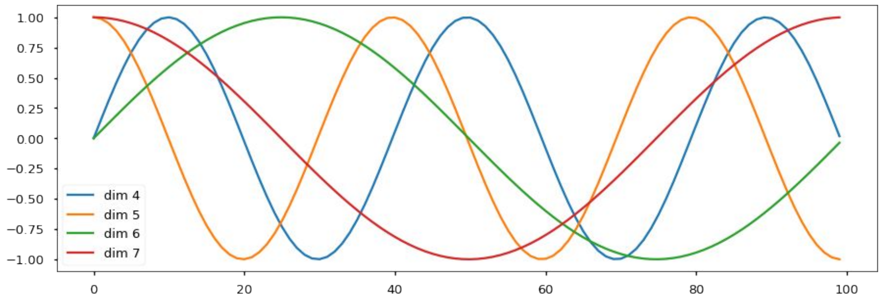
Typically after the first feedforward layer, a *positional encoding* is added to each $\mathbf{h}_t^{(1)}$

Without this, the model cannot distinguish between different permutations of the same input sequence

dimensionality of $\mathbf{h}_t^{(1)}$

Don't stare at this too hard, but a common choice is to add:

$$p_t = \begin{bmatrix} \sin(t/10000^{2*1/d}) \\ \cos(t/10000^{2*1/d}) \\ \sin(t/10000^{2*2/d}) \\ \cos(t/10000^{2*2/d}) \\ \dots \\ \sin(t/10000^{2*\frac{d}{2}/d}) \\ \cos(t/10000^{2*\frac{d}{2}/d}) \end{bmatrix}$$



Important detail: positional encoding

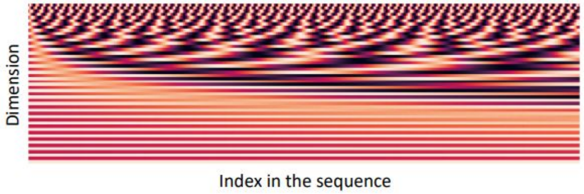
This choice of positional encoding looks pretty strange, are there alternatives?

What about just concatenating the time step after the first feedforward layer?

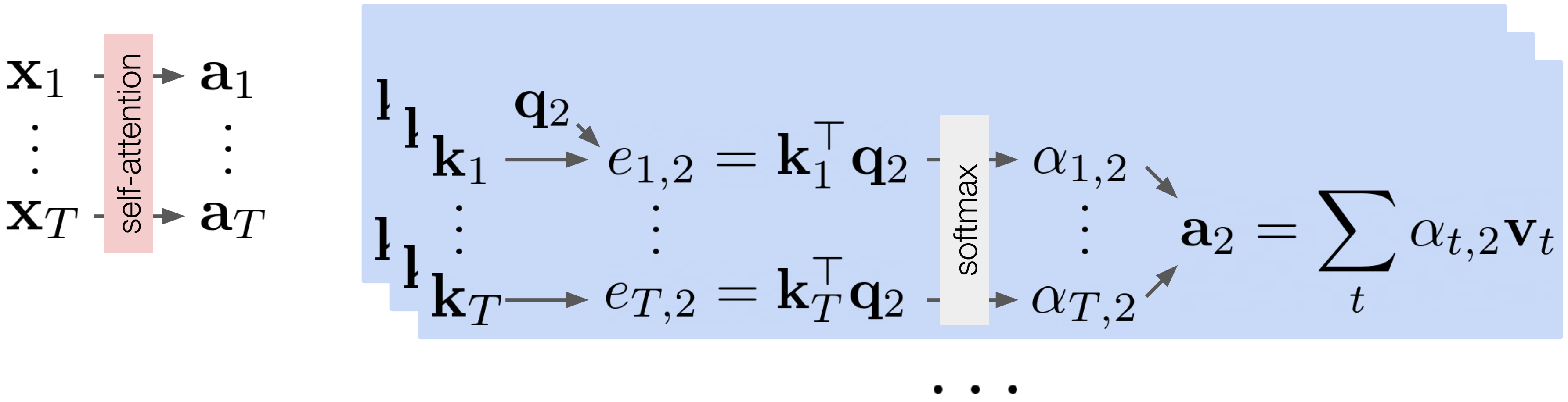
- This appears worse because we care more about *relative* positioning

What about learning the positional encodings?

- This is used sometimes and is potentially better due to greater expressivity
- There are also downsides, e.g., we can't generalize to longer sequences



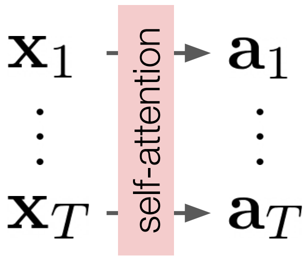
Important detail: multi-head attention

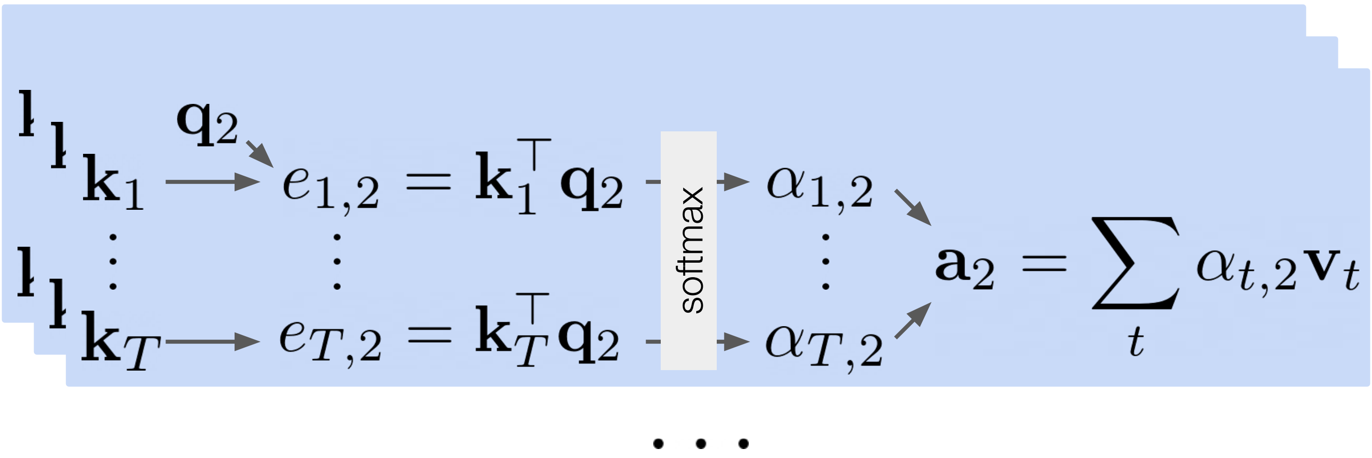


All of the key, query, and value functions (which are often just linear layers) have their own learned parameters

The final $\mathbf{a}_2$, and every other $\mathbf{a}_t$, is obtained by concatenating the outputs from every "head" and potentially feeding this through another linear layer

Important detail: multi-head attention





Typically, the key, query, and value dimensionalities are scaled down proportionally to the number of heads

E.g., if using dimensionality 512 for one head, scale down to dimensionality 64 for 8 heads https://nlp.seas.harvard.edu/2018/04/03/attention.html

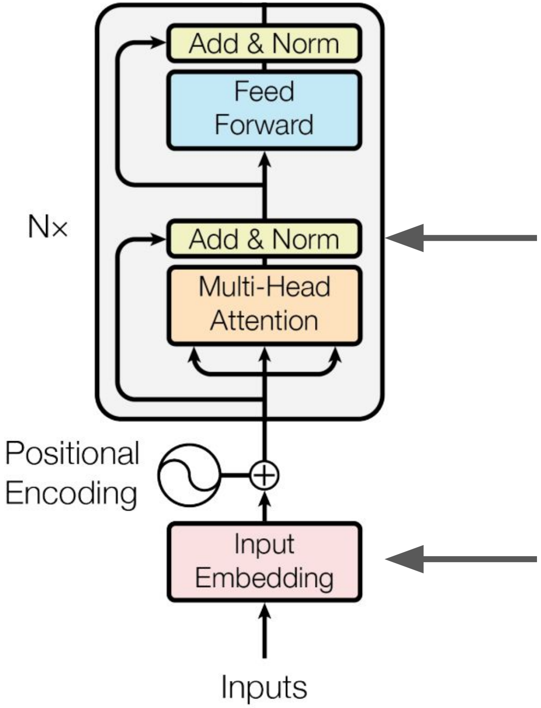
Other details

Transformers use layer normalization, dropout, and skip connections

```
class SublayerConnection(nn.Module):
    """
    A residual connection followed by a layer norm.
    Note for code simplicity the norm is first as opposed to last.
    """
    def __init__(self, size, dropout):
        super(SublayerConnection, self).__init__()
        self.norm = LayerNorm(size)
        self.dropout = nn.Dropout(dropout)

    def forward(self, x, sublayer):
        "Apply residual connection to any sublayer with the same size."
        return x + self.dropout(sublayer(self.norm(x)))
```

The transformer encoder – full picture



LN, dropout, skip connection

lookup table / linear layer

Vaswani et al, “Attention is All You Need”. NIPS 2017.

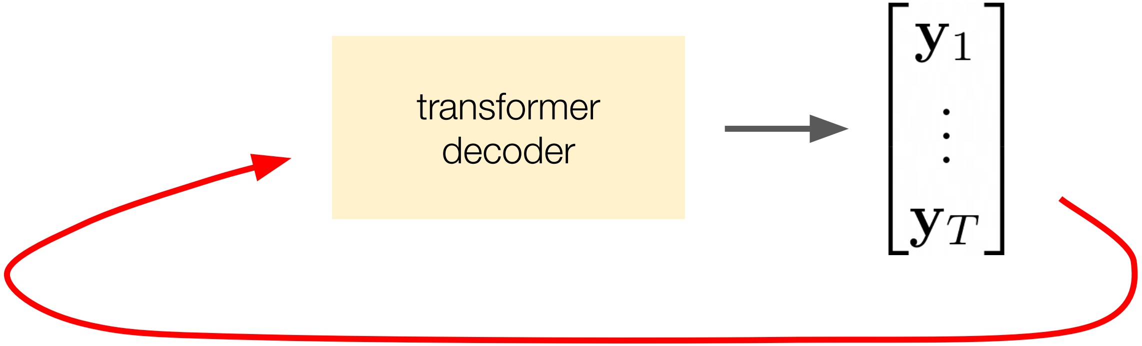
Transformers for “decoding” (generation)

Input: In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

Output: The scientist named the population, after their distinctive horn, Ovid’s Unicorn. These four-horned, silver-white unicorns were previously unknown to science.

Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.

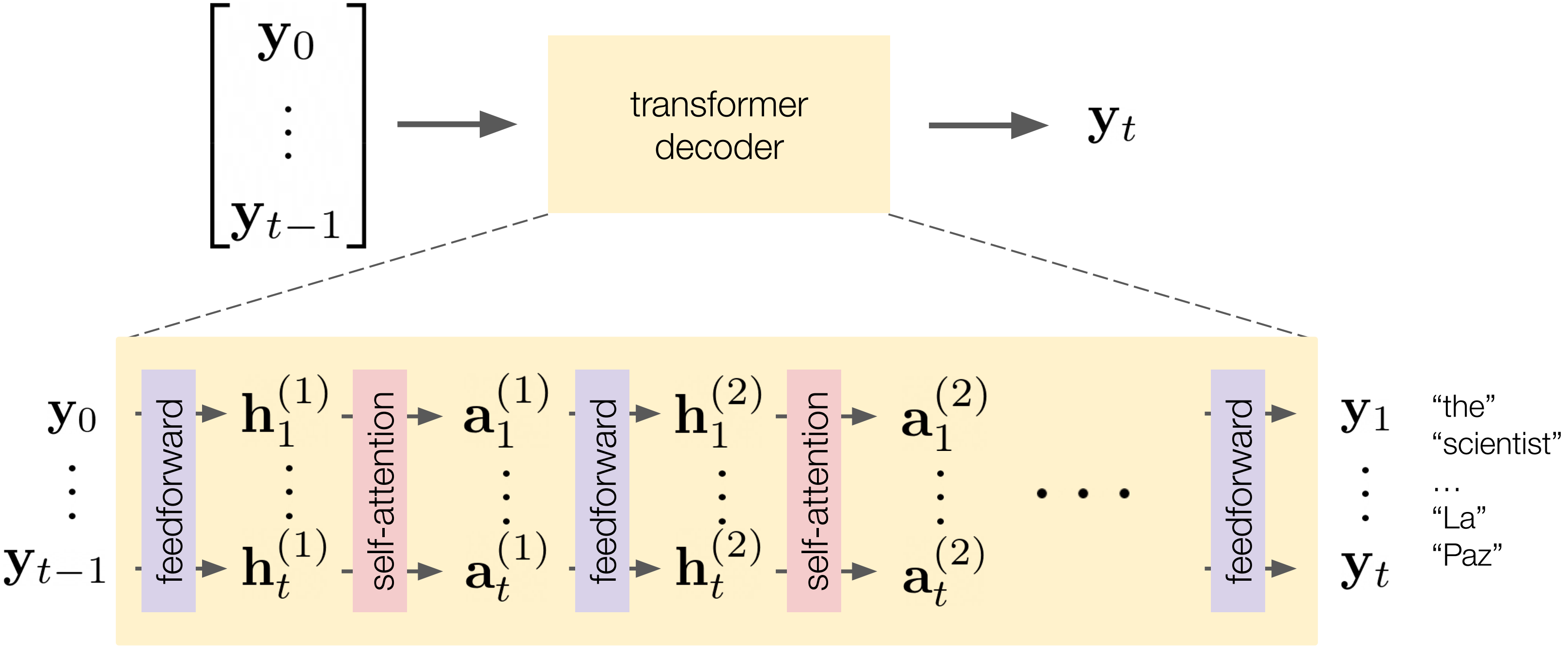
Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.



The decoder’s generated output (so far) is fed back as input into the decoder

For training, the ground truth is fed into the decoder

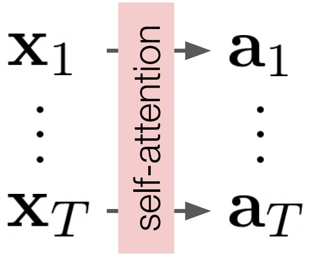
Transformers for "decoding" (generation)

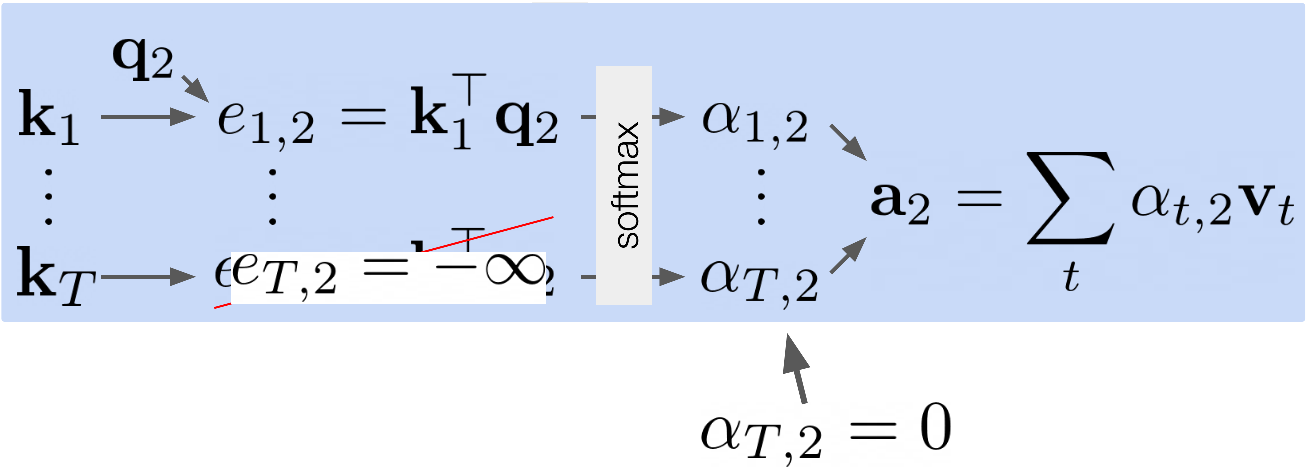


Important detail: masked attention

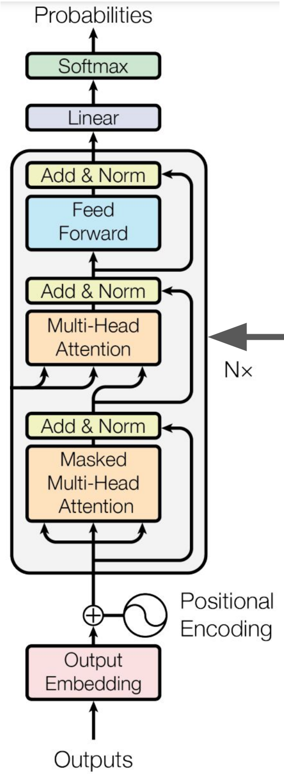
During training, entire sequences are passed into the model, and the model must be prevented from “looking at the future” when learning to generate

One way to do this is to “mask” keys and values corresponding to future time steps





The transformer decoder – full picture



cross attention (next week)

Vaswani et al, “Attention is All You Need”. NIPS 2017.

Additional resources

- The original transformer paper
    - https://arxiv.org/abs/1706.03762

- The annotated transformer (paper snippets + PyTorch code)
    - https://nlp.seas.harvard.edu/2018/04/03/attention.html

- Prof. Sergey Levine's lecture videos
    - https://www.youtube.com/watch?v=VDnEnlYzHOU&list=PL_iWQOsE6TfVmKkQHucjPAoRtIJYt8a5A