Transformers
(attention is all you need)

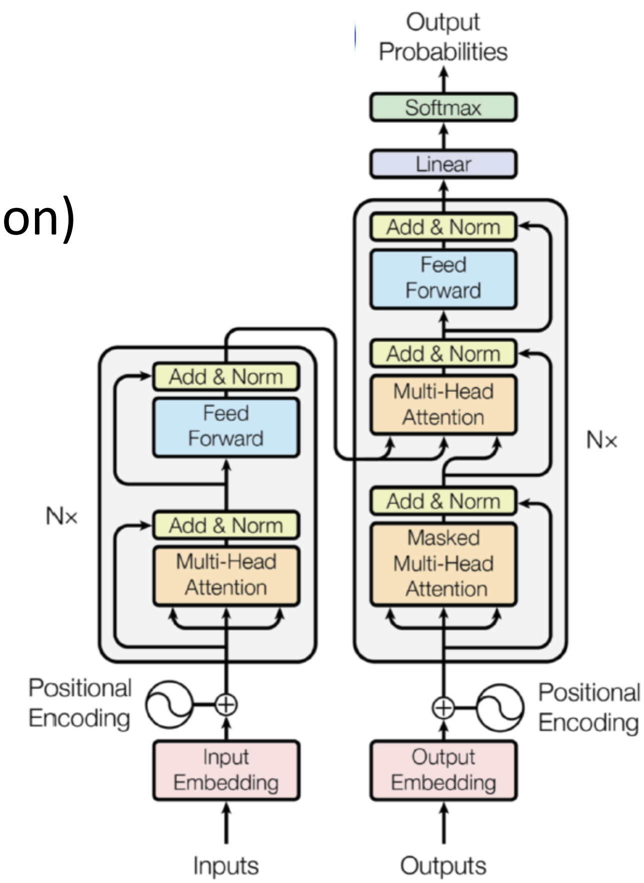
Some slides from [Full Stack Deep Learning – UC Berkeley Spring 2021]
by Sergey Karayev, Josh Tobin, Pieter Abbeel

Some material from https://peterbloem.nl/blog/transformers

Transformers

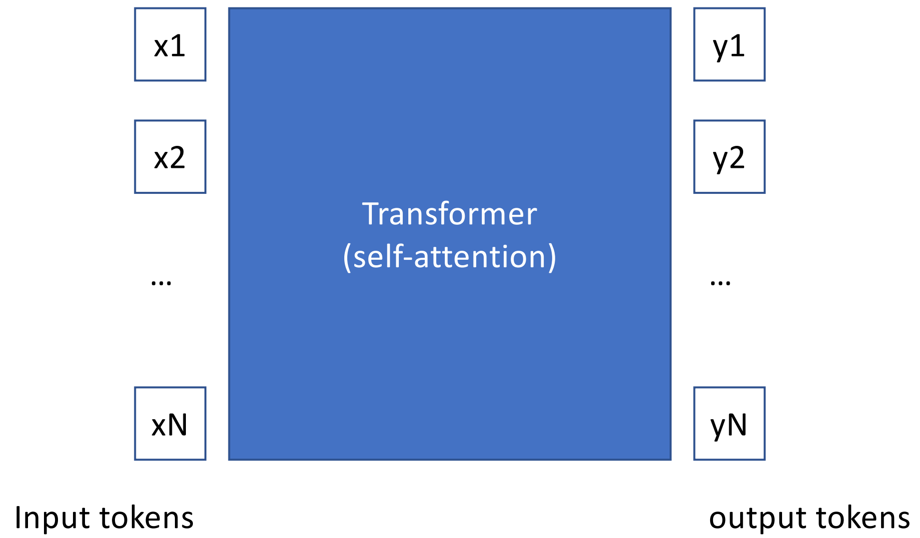
- Revolutionized NLP (developed for translation)
- Revolutionized Vision (ViT architectures)
- Revolutionized GenAI (DiT architectures)

- An “encoder-decoder” architecture



Core: sequence to sequence translation

- Equivariant to changes in input order
  - permute(input) => permute(output)



How: self-attention

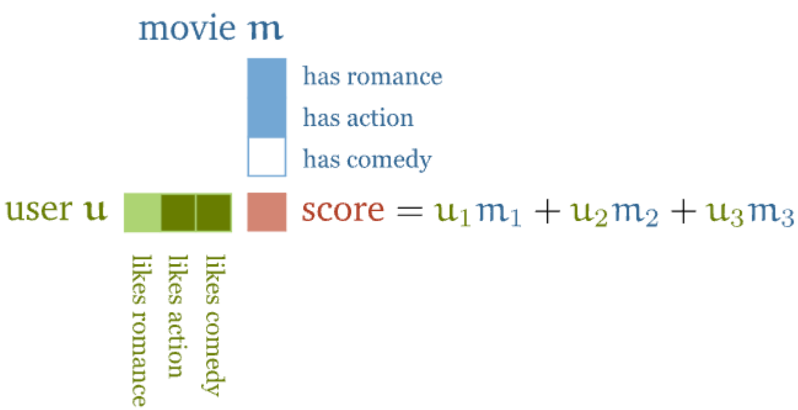
- Relate inputs to outputs
- Output is a weighted sum
- Weights relate pairs of inputs

- What are these weights?
  - CNNs had fixed weights
  - New input? Same weights
  - Transformers have dynamic weights

$$\mathbf{y}_i = \sum_j w_{ij} \mathbf{x}_j \,.$$

How: self-attention

- Weights derived from input
- Defined by dot/inner product

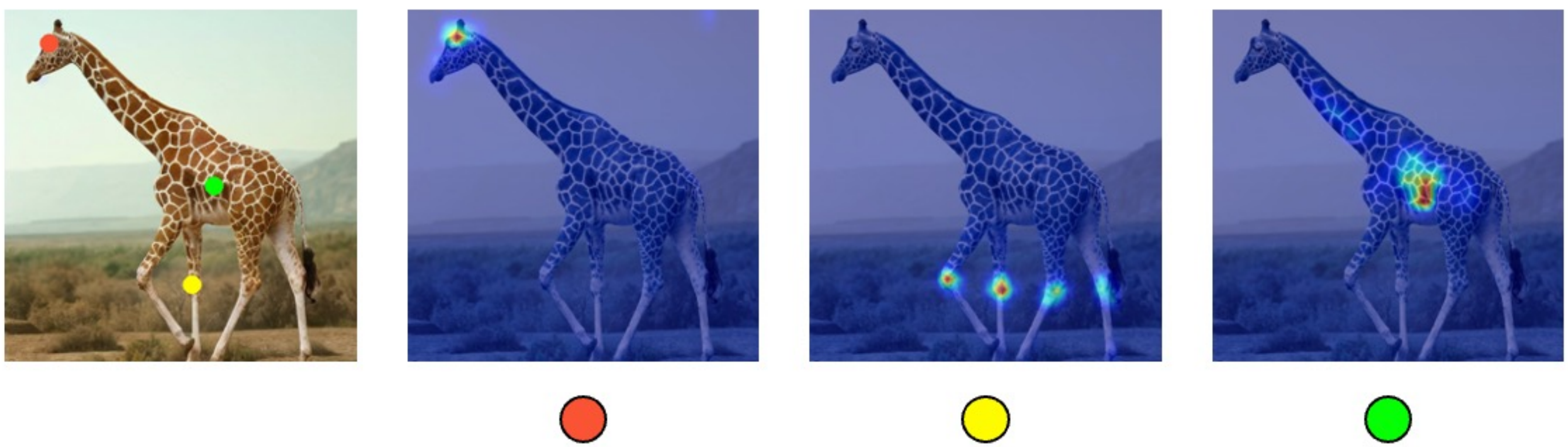


$$\mathbf{y}_i = \sum_j w_{ij} \mathbf{x}_j \, .$$

$$w'_{ij} = \mathbf{x}_i{}^{\mathsf{T}} \mathbf{x}_j \, .$$

How: self-attention (example in GenAI)

- Image generated by a diffusion transformer (DiT)
- What does self-attention look like?



How: self-attention

- Problem
  - Range of dot products? [-∞,+ ∞]
- Solution
  - Softmax
-

  Maps weights [0,1], sum to 1
- Output is a <u>convex</u> combination
- Range(outputs) ~ Range(inputs)

$$y_i = \sum_j w_{ij} x_j \ .$$

$$w'_{ij} = \mathbf{x}_i^{\mathsf{T}} \mathbf{x}_j \ .$$

$$w_{ij} = \frac{\exp w'_{ij}}{\sum_j \exp w'_{ij}} \ .$$

Implementation (pytorch)

- Do not write for loops over pairs of vertices… use matrices!

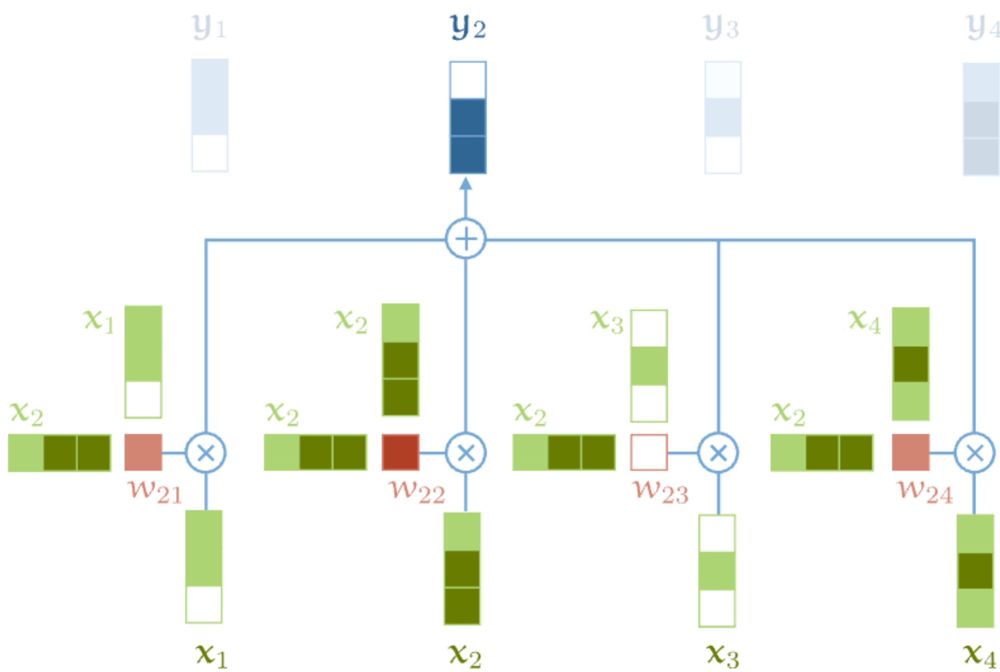
```
import torch
import torch.nn.functional as F

# assume we have some tensor x with size (b, t, k)
x = ...

raw_weights = torch.bmm(x, x.transpose(1, 2))
# - torch.bmm is a batched matrix multiplication. It
#   applies matrix multiplication over batches of
#   matrices.
weights = F.softmax(raw_weights, dim=2)

y = torch.bmm(weights, x)
```

How: self-attention (...where is learning?)



Intuition: query / key / value (database)

- A soft realization of “if query matches the key, return the value”

Query1: Anastasia

Query2: Ferguson

Key	Phone Number	Address
Rob	+1-708-706-7562	5685 Random St, City 22, State FL
Lucas	+1-453-162-3657	2110 Random St, City 41, State TX
Maria	+1-853-582-6620	9825 Random St, City 1, State TX
Ferguson	+1-491-106-4045	4148 Random St, City 7, State IL
Peter	+1-231-340-4706	6494 Random St, City 44, State IL
Jonathan	+1-935-552-9803	6910 Random St, City 10, State IL
Ming	+1-692-772-1483	6026 Random St, City 4, State IL
Johnson	+1-978-590-4234	6580 Random St, City 13, State TX
Anastasia	+1-221-654-3441	6144 Random St, City 13, State NY
Lily	+1-285-431-8770	278 Random St, City 1, State IL
Person 11	+1-463-319-6478	4075 Random St, City 28, State CA
Person 12	+1-961-975-4480	5256 Random St, City 7, State NY
Person 13	+1-337-563-4023	4303 Random St, City 4, State TX
Person 14	+1-571-765-8826	413 Random St, City 16, State CA
Person 15	+1-614-477-9465	9312 Random St, City 30, State CA

Math: query / key / values (learning)

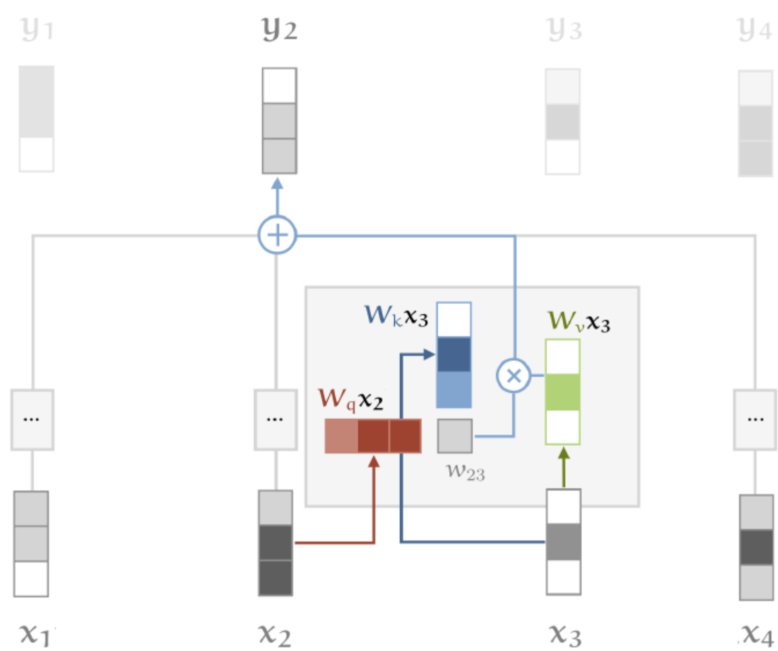
$$\mathbf{q}_i = \boxed{\boldsymbol{W}_q} \mathbf{x}_i \qquad \mathbf{k}_i = \boxed{\boldsymbol{W}_k} \mathbf{x}_i \qquad \boldsymbol{v}_i = \boxed{\boldsymbol{W}_v} \mathbf{x}_i$$

$$w'_{ij} = \mathbf{q}_i{}^{\mathsf{T}} \mathbf{k}_j$$

$$w_{ij} = \text{softmax}(w'_{ij})$$

$$\mathbf{y}_i = \sum_j w_{ij} \boldsymbol{v}_j \, .$$

Math: visual interpretation

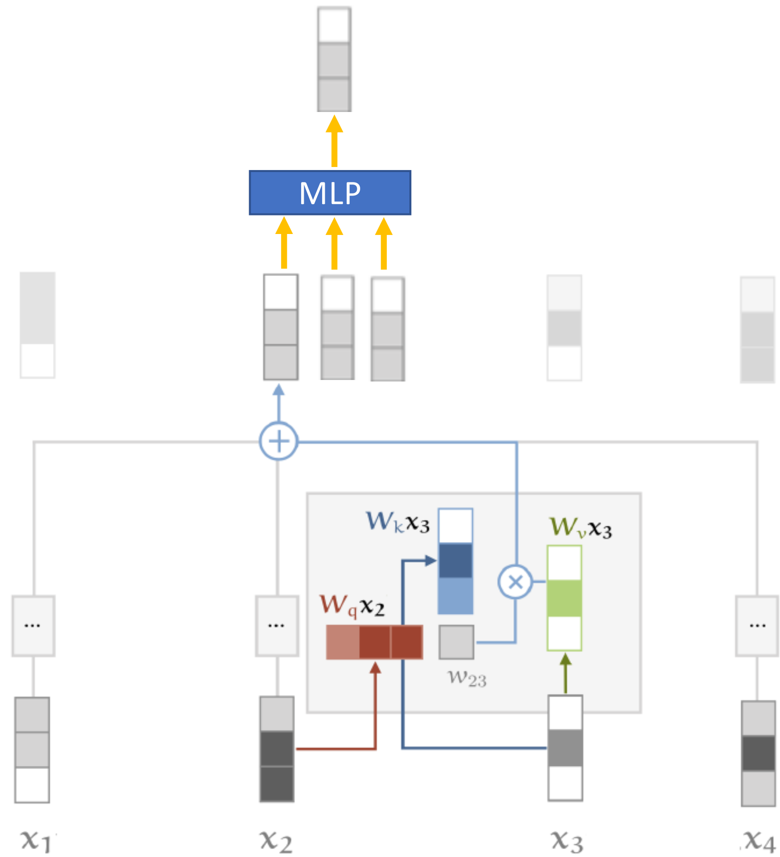


Matrix form: query / key / values

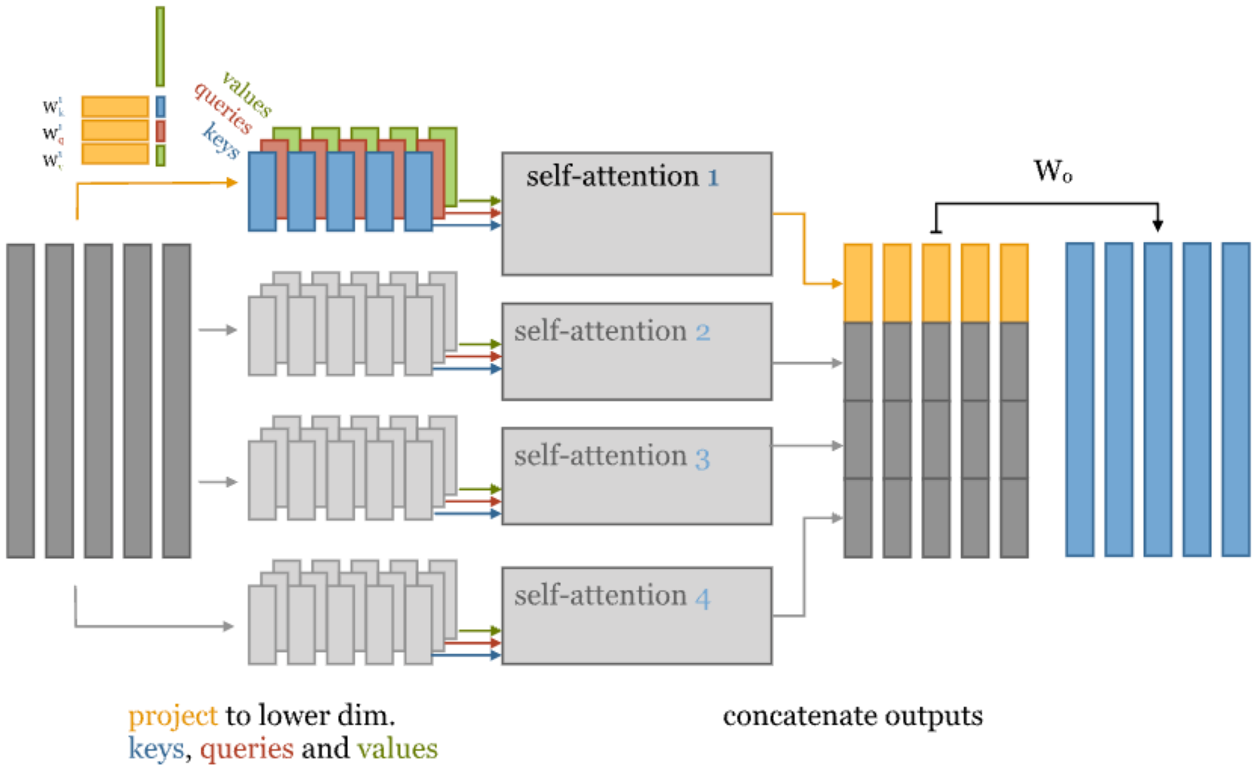
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Multi-head attention

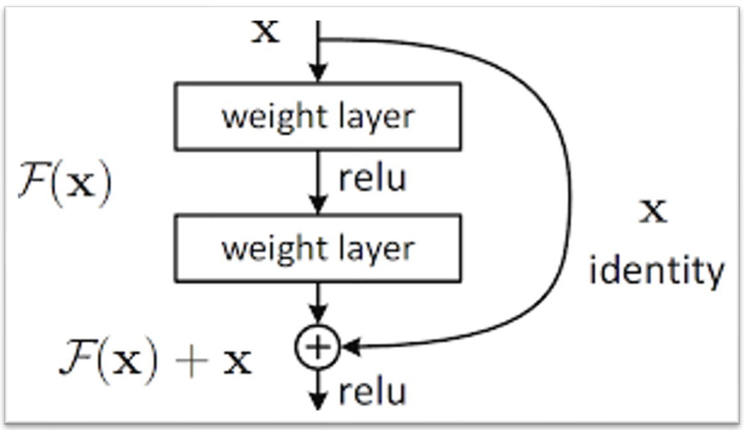
- Multiple ways to relate concepts
- Just execute the same architecture with different sets of W
- Analogous to “filter banks” in CNN
- “Squash” the output back original dimensionality with an MLP
- Why? Deeper architecture



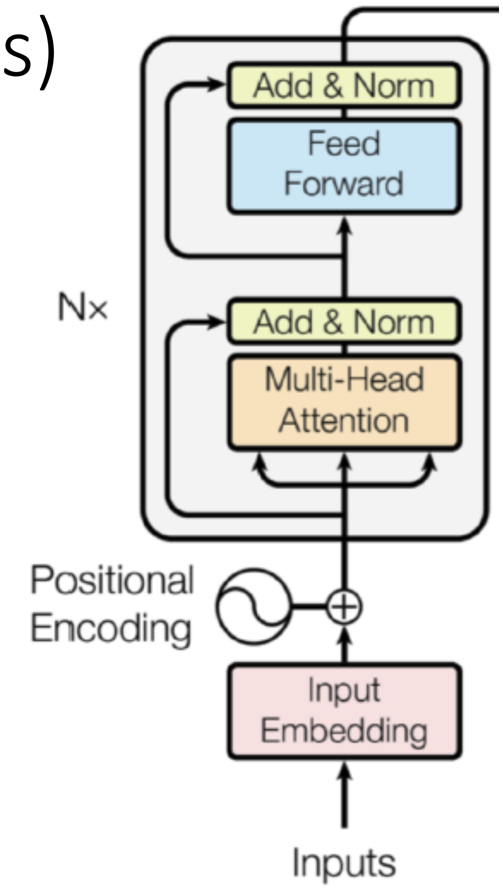
Multi-head attention (different visualization)



Skip connections (deeper nets)

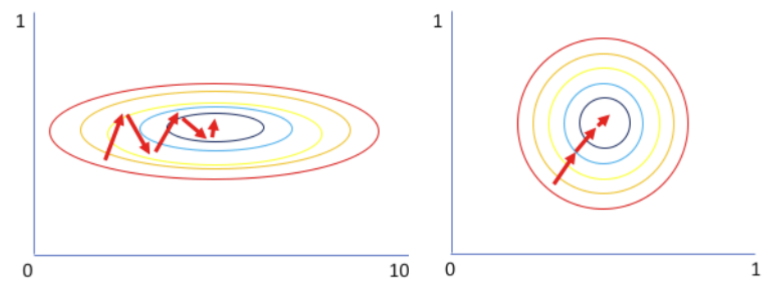


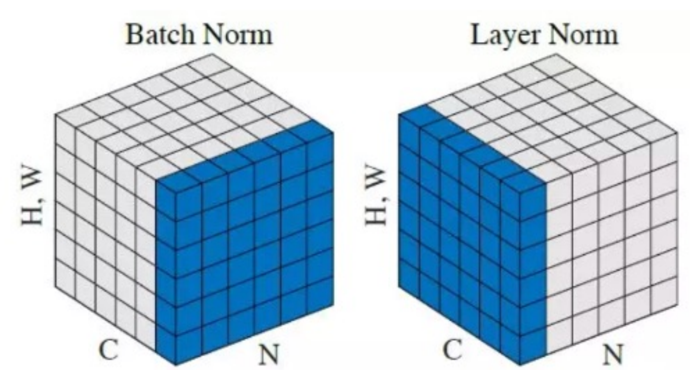
similar to the ResNet module



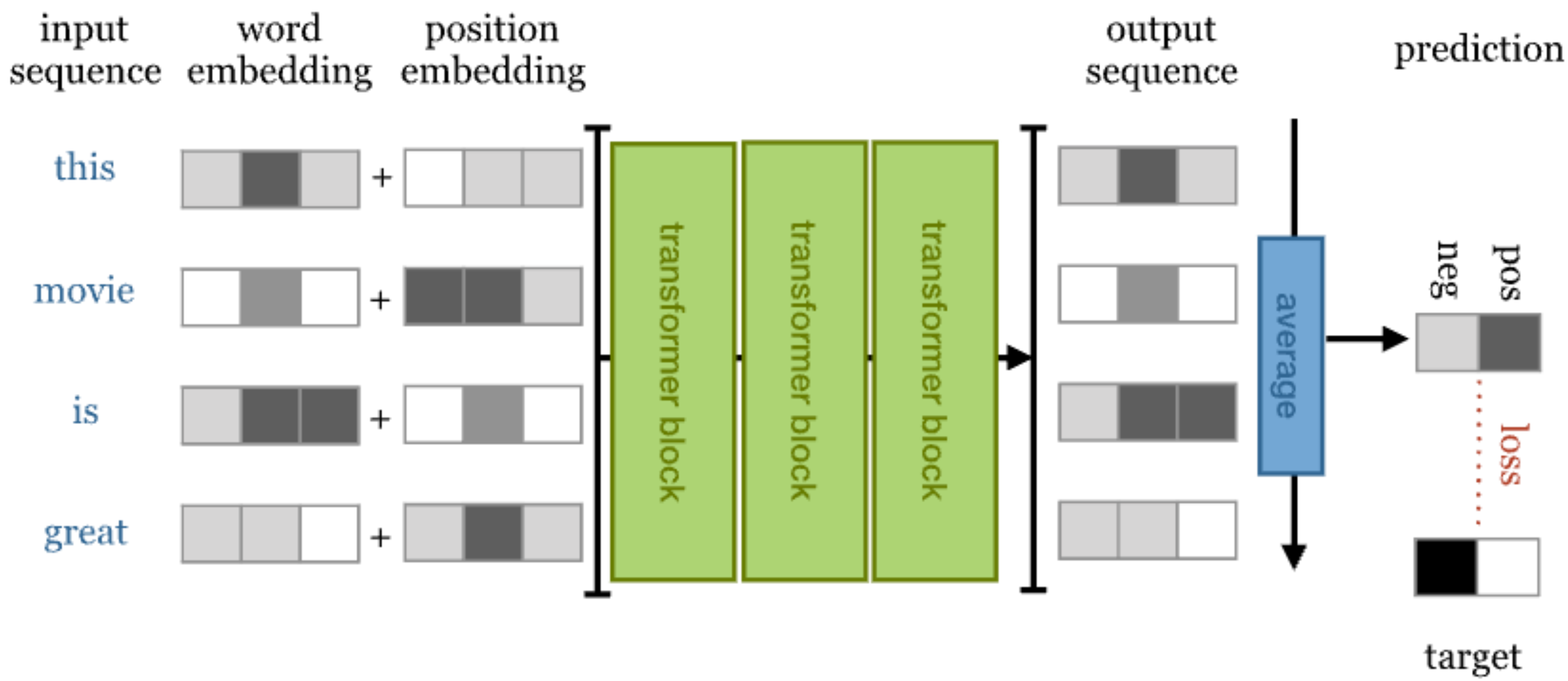
Layer normalization

- Neural nets work best when inputs have uniform mean/std in each dimension
- As data flows, mean/std may violate this property
- Layer norm ”resets” statistics between layers

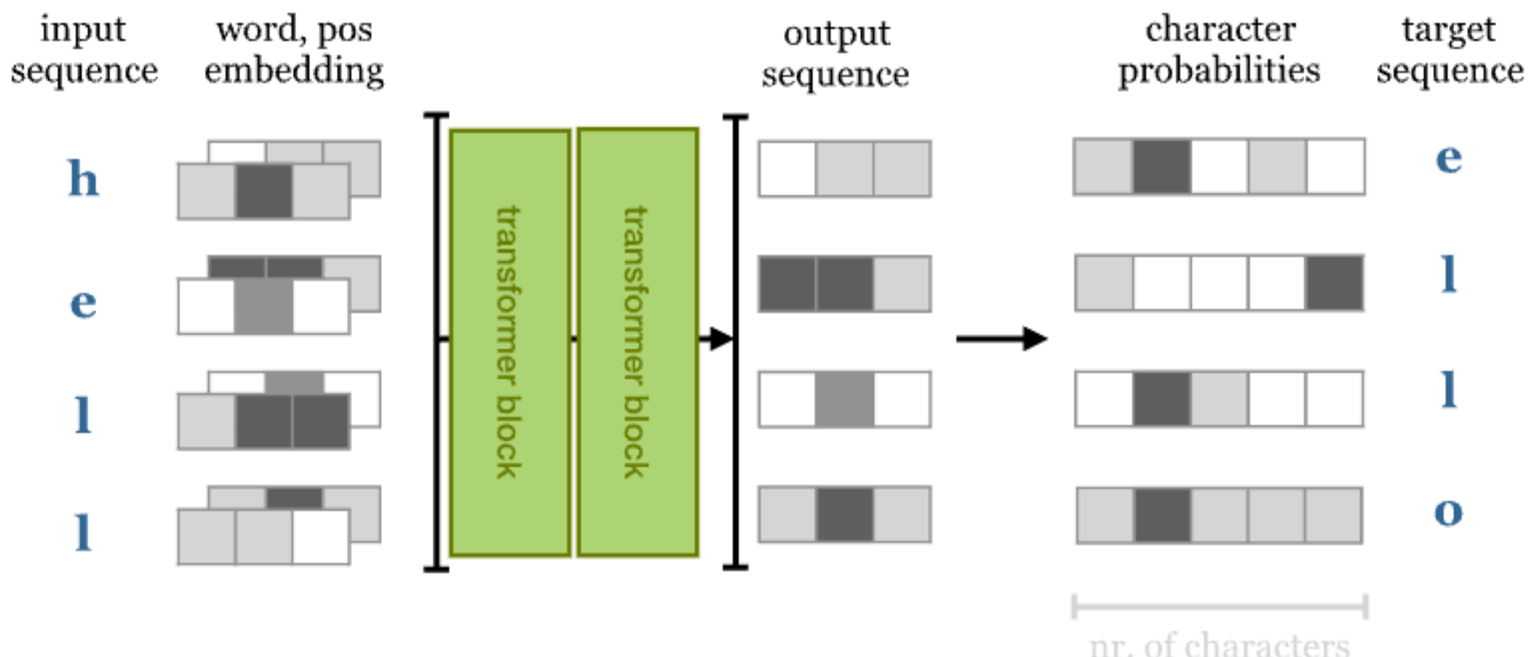




Example: classification transformer

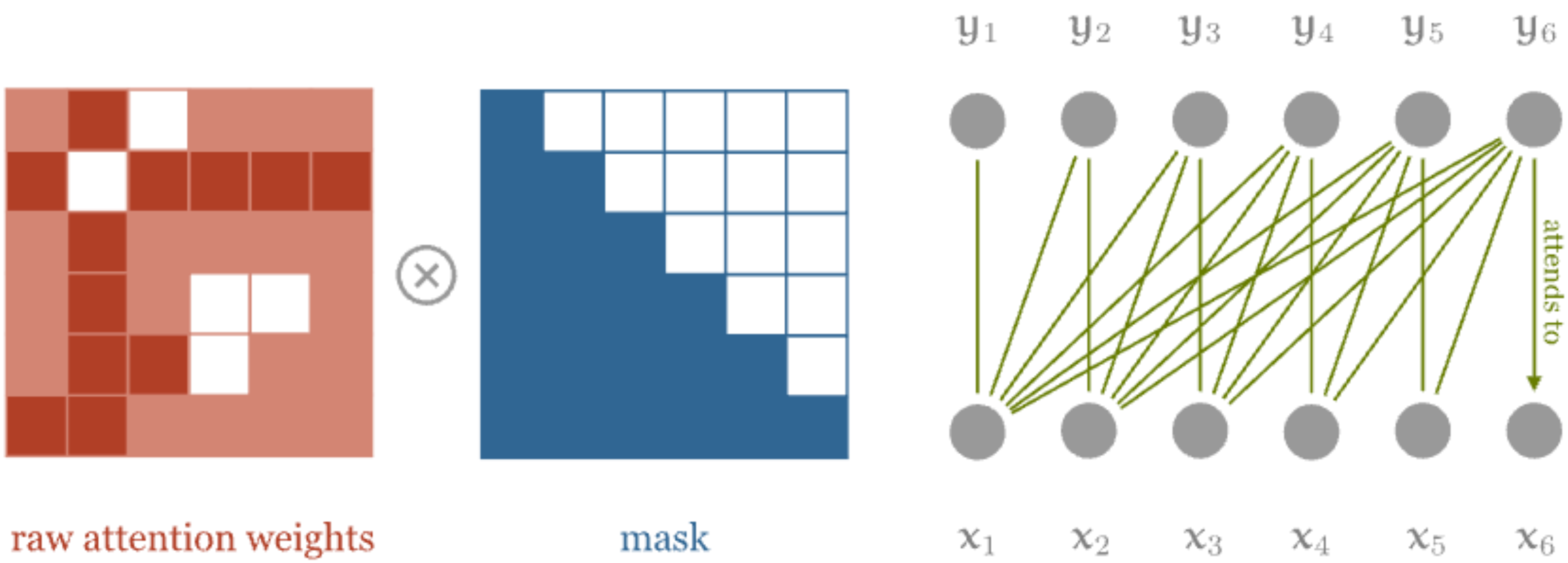


Example: auto-regressive prediction



Auto-regressive transformers… how?

- Prevents cheating (copying input token to the output)



Transformer revolutions (NLP)

towards
data science

Sign in Get started

Follow 593K Followers · Editors' Picks Features Deep Dives Grow Contribute

BERT Explained: State of the art language model for NLP

Rani Horev Nov 10, 2018 · 7 min read

BERT (Bidirectional Encoder Representations from Transformers) is a recent paper published by researchers at Google AI Language. It has caused a stir in the Machine Learning community by presenting state-of-the-art results in a wide variety of NLP tasks, including Question Answering (SQuAD v1.1), Natural Language Inference (MNLI), and others.

BERT's key technical innovation is applying the bidirectional training of Transformer, a popular attention model, to language modelling. This is in contrast to previous efforts which looked at a text sequence either from left to right or combined left-to-right and right-to-left training. The paper's results show that a language model which is bidirectionally trained can have a deeper sense of language context and flow than single-direction

neptuneblog What is Neptune? Search all articles

How to Use Neptune ML Experiment Tracking ML Model Management

Blog » Natural Language Processing » 10 Things You Need to Know About BERT and the Transformer Architecture That Are Reshaping the AI Landscape

10 Things You Need to Know About BERT and the Transformer Architecture That Are Reshaping the AI Landscape

25 mins read | Author Cathal Horan | Updated July 20th, 2021

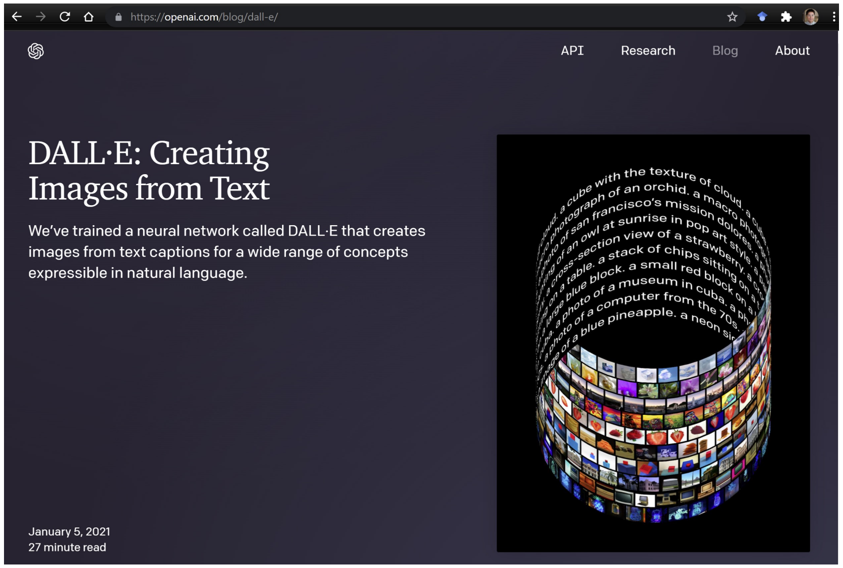
Few areas of AI are more exciting than NLP right now. In recent years language models (LM), which can perform human-like linguistic tasks, have evolved to perform better than anyone could have expected.

In fact, they're performing so well that people are wondering whether they're reaching a level of general intelligence, or the evaluation metrics we use to test them just can't keep up. When technology like this comes along, whether it is electricity, the railway, the internet or the iPhone, one thing is clear – you can't ignore it. It will end up impacting every part of the modern world.

It's important to learn about technologies like this, because then you can use them to your advantage. So, let's learn!

We will cover ten things to show you where this technology came from, how it was developed, h
it works, and what to expect from it in the near future. The ten things are:

Transformer revolutions (Vision GenAI)



TEXT PROMPT

an illustration of a baby daikon radish in a tutu walking a dog

AI-GENERATED IMAGES

Edit prompt or view more images↓

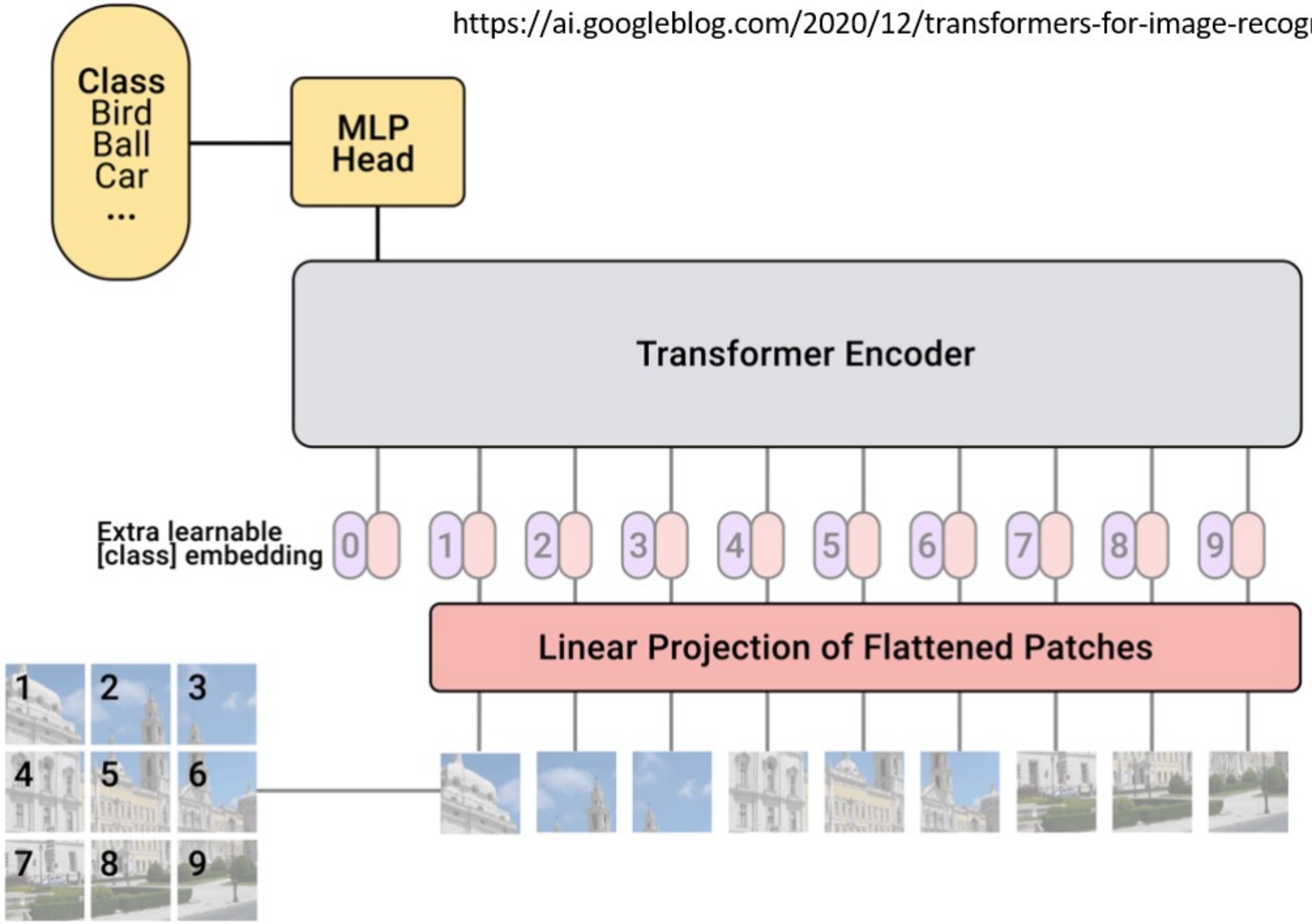
TEXT PROMPT

an armchair in the shape of an avocado. . . .

AI-GENERATED IMAGES

Edit prompt or view more images↓

https://ai.googleblog.com/2020/12/transformers-for-image-recognition-at.html https://ai.googleblog.com/2020/12/transformers-for-image-recognition-at.html



Computer Science > Computer Vision and Pattern Recognition

[Submitted on 26 May 2020 (v1), last revised 28 May 2020 (this version, v3)]

End-to-End Object Detection with Transformers

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, Sergey Zagoruyko

We present a new method that views object detection as a direct set prediction problem. Our approach streamlines the detection pipeline, effectively removing the need for many hand-designed components like a non-maximum suppression procedure or anchor generation that explicitly encode our prior knowledge about the task. The main ingredients of the new framework, called DEtection TRansformer or DETR, are a set-based global loss that forces unique predictions via bipartite matching, and a transformer encoder-decoder architecture. Given a fixed small set of learned object queries, DETR reasons about the relations of the objects and the global image context to directly output the final set of predictions in parallel. The new model is conceptually simple and does not require a specialized library, unlike many other modern detectors. DETR demonstrates accuracy and run-time performance on par with the well-established and highly-optimized Faster RCNN baseline on the challenging COCO object detection dataset. Moreover, DETR can be easily generalized to produce panoptic segmentation in a unified manner. We show that it significantly outperforms competitive baselines. Training code and pretrained models are available at this https URL.

Subjects: **Computer Vision and Pattern Recognition (cs.CV)**
Cite as: arXiv:2005.12872 **[cs.CV]**
(or arXiv:2005.12872v3 **[cs.CV]** for this version)

Submission history

From: Nicolas Carion [view email]
[v1] Tue, 26 May 2020 17:06:38 UTC (6,968 KB)
[v2] Wed, 27 May 2020 13:57:30 UTC (6,968 KB)
[v3] Thu, 28 May 2020 17:37:23 UTC (6,968 KB)

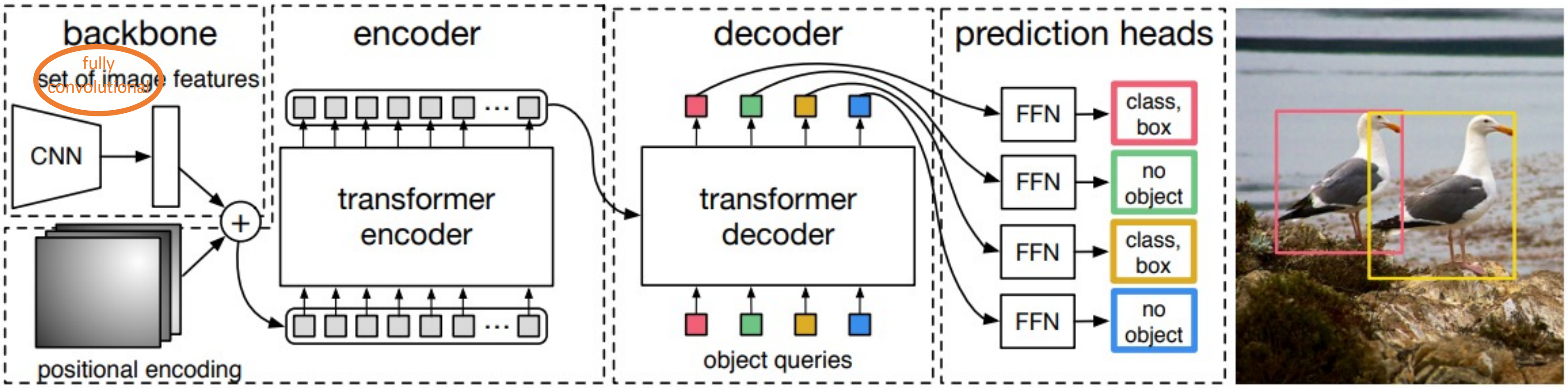
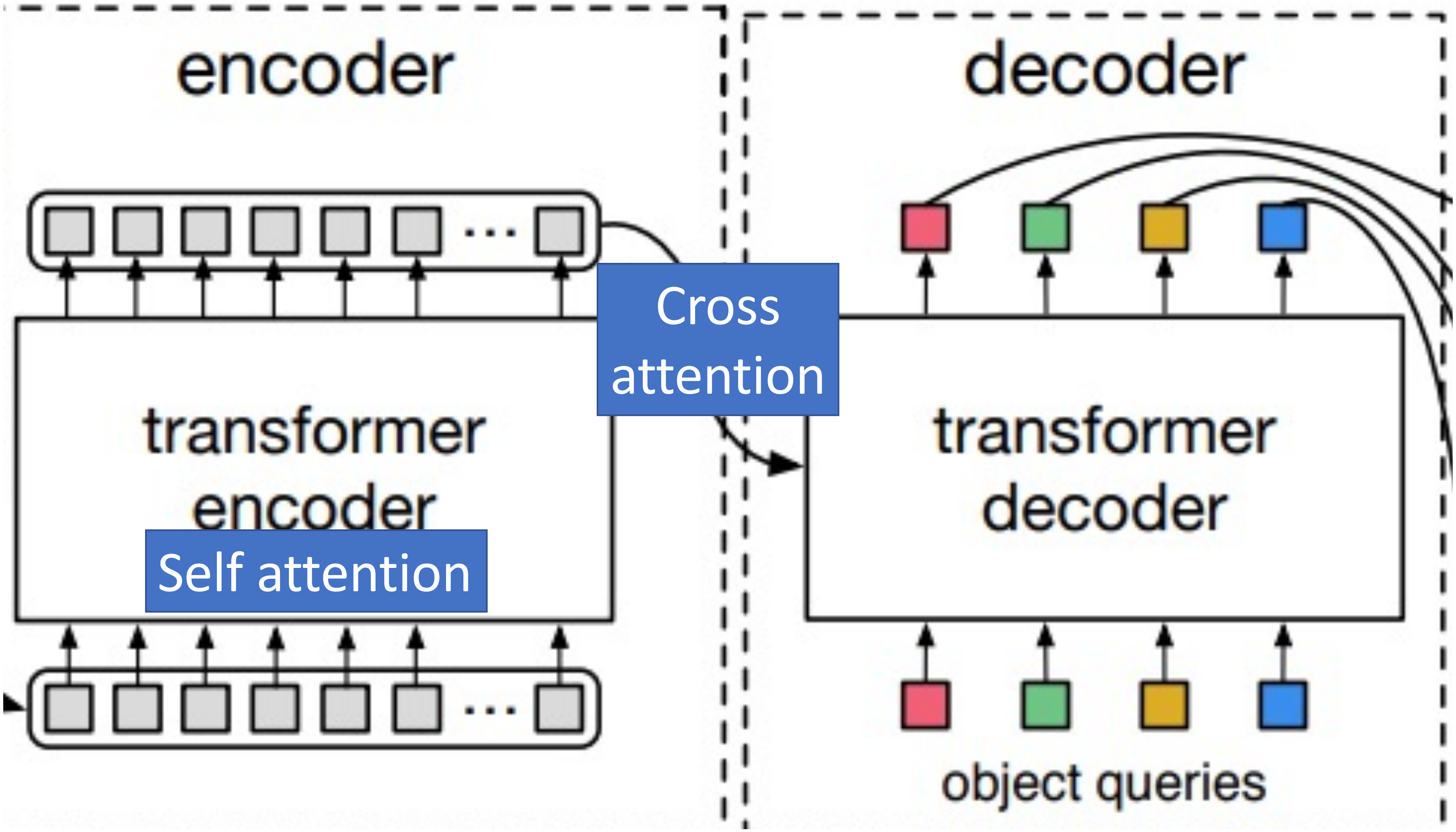
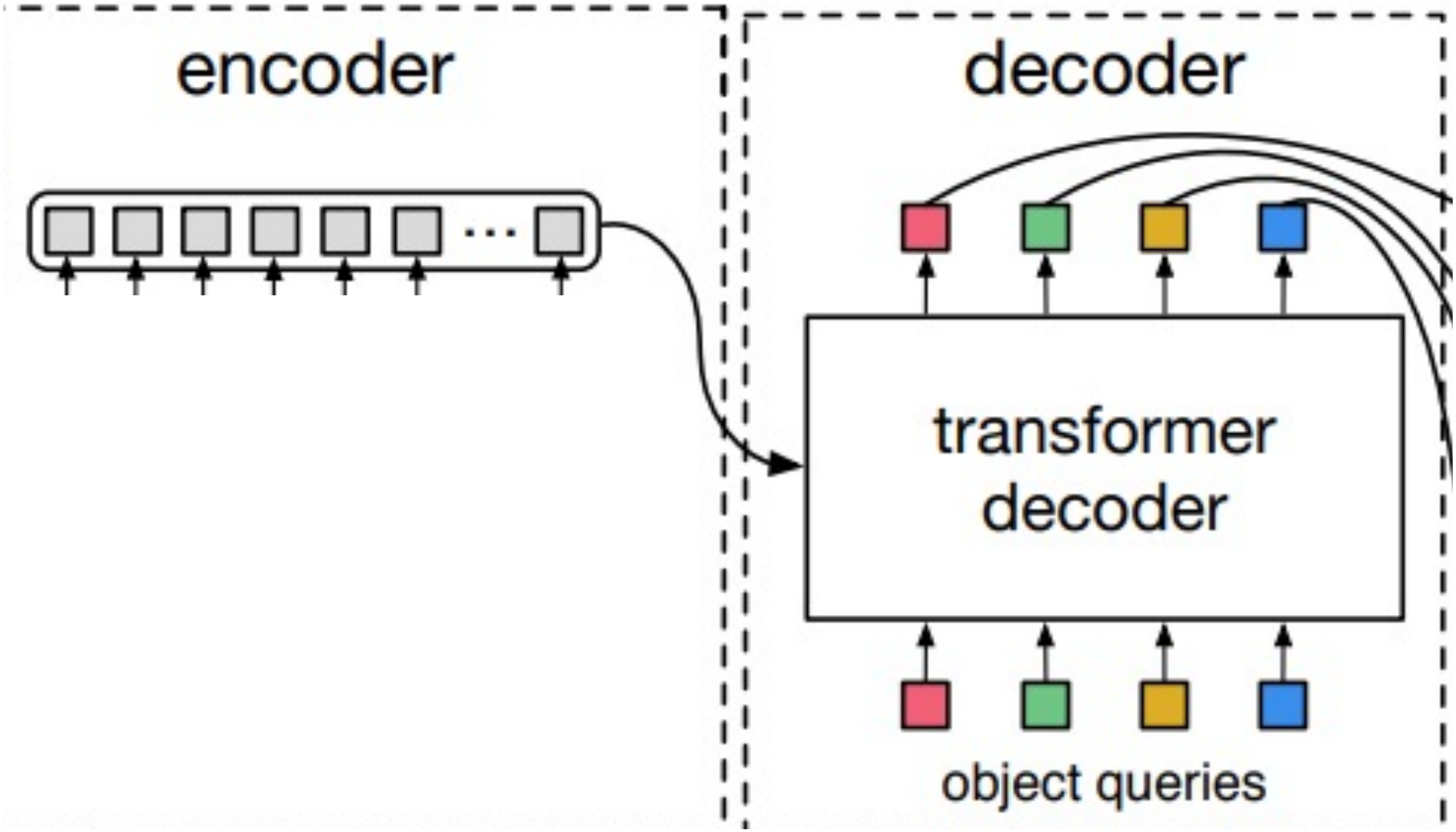
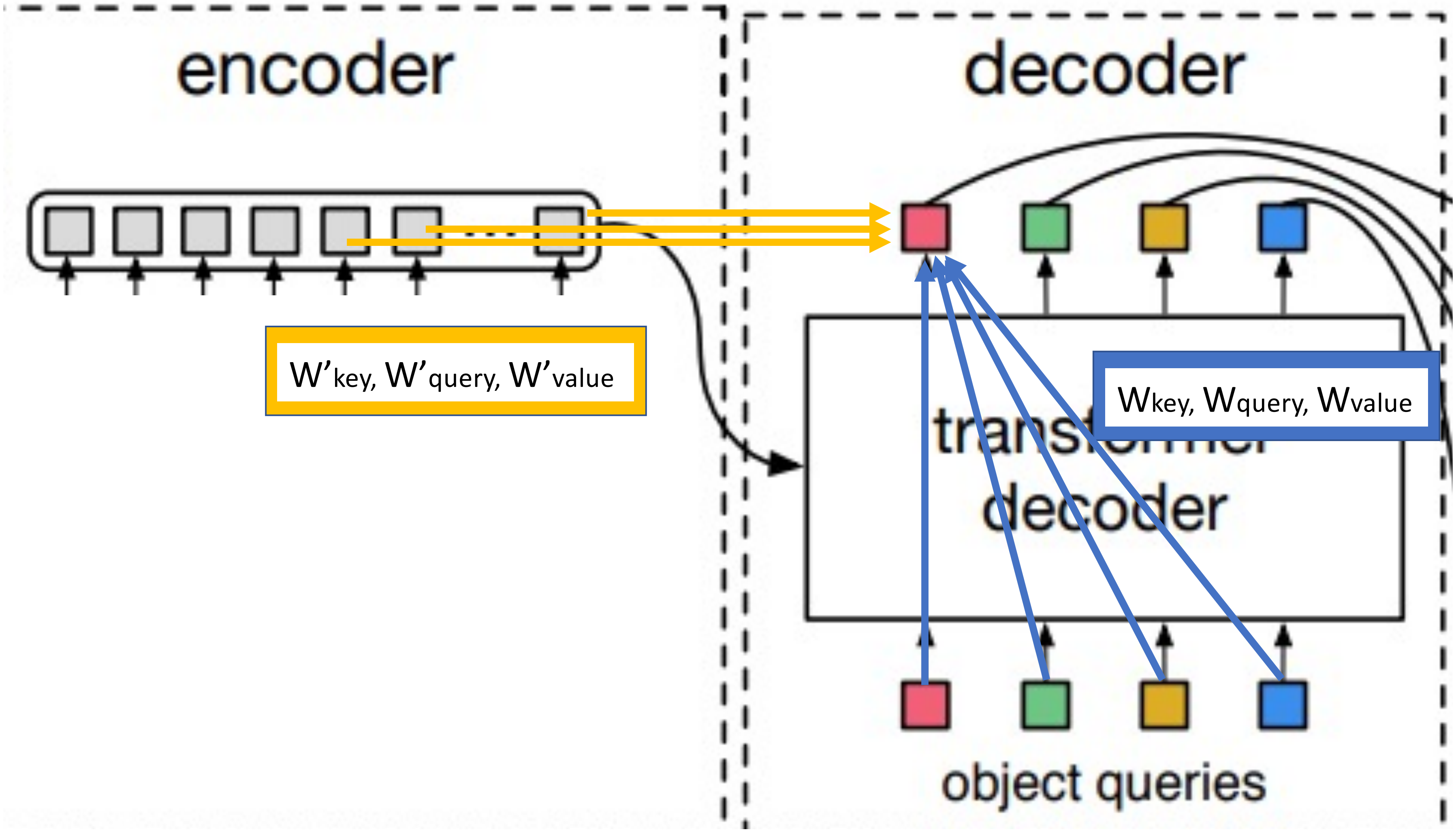


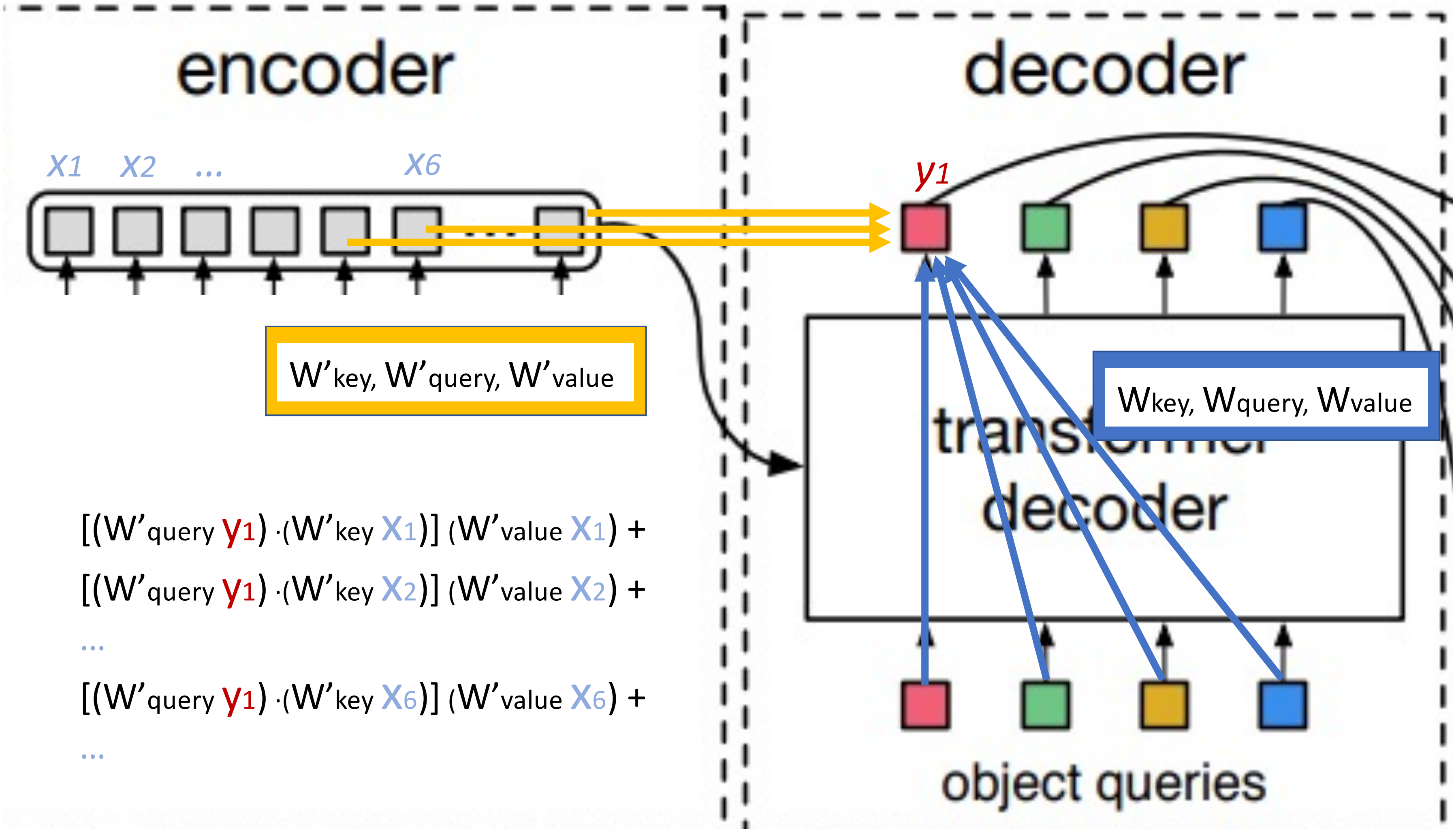
Fig. 2: DETR uses a conventional CNN backbone to learn a 2D representation of an input image. The model flattens it and supplements it with a positional encoding before passing it into a transformer encoder. A transformer decoder then takes as input a small fixed number of learned positional embeddings, which we call *object queries*, and additionally attends to the encoder output. We pass each output embedding of the decoder to a shared feed forward network (FFN) that predicts either a detection (class and bounding box) or a "`no object`" class.


encoder
decoder
Cross attention
transformer encoder
Self attention
transformer decoder
object queries


encoder
decoder
transformer
decoder
object queries



encoder
decoder
W’key, W’query, W’value
Wkey, Wquery, Wvalue
transformer decoder
object queries


encoder
decoder
x_1 x_2 … x_6
y_1
W'_{key}, W'_{query}, W'_{value}
transformer decoder
W_{key}, W_{query}, W_{value}
object queries
$[(W'_{query}\, y_1) \cdot (W'_{key}\, x_1)]\, (W'_{value}\, x_1)$ +
$[(W'_{query}\, y_1) \cdot (W'_{key}\, x_2)]\, (W'_{value}\, x_2)$ +
…
$[(W'_{query}\, y_1) \cdot (W'_{key}\, x_6)]\, (W'_{value}\, x_6)$ +
…

Summary of Transformer Architecture

- More powerful than CNNs
  - Weights are input-conditional (vs. fixed)
  - Long-range interactions (better than atreous convolutions)
  - Less memorization of boundary/padding w/ large kernels
- Flexible like GNNs
  - Arbitrary input/output dims
  - Long-range interactions
- Train faster than RNNs
  - No need to wait for auto-regressive chain to be computed to train
- State-of-the-art in language and vision
  - Drawback: $O(N^2)$ attention… lots of work here (e.g. FlashAttention, Performers, …)