Transformers

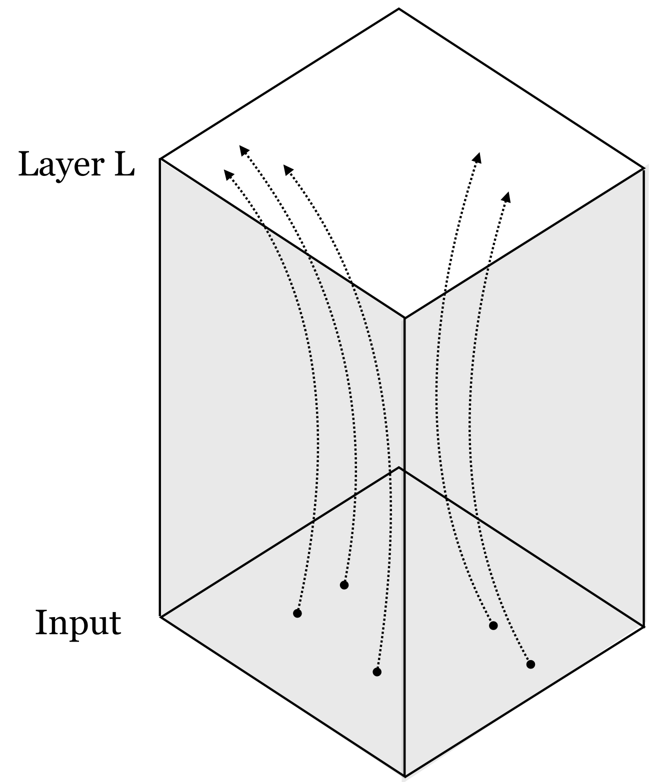
Slides adapted from Phillip Isola

Transformers

- Three key ideas
    - Tokens
    - Attention
    - Positional encoding
- Examples of architectures and applications

Deep nets are data transformers

- Deep nets transform datapoints, layer by layer
- Each layer is a different *representation* of the data
- We call these representations **embeddings**

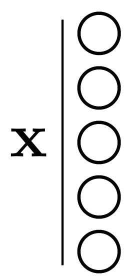


Idea #1: tokens

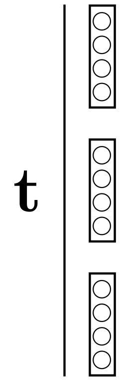
A new data structure: Tokens

- A **token** is just transformer lingo for a vector of neurons
- But the connotation is that a token is an encapsulated bundle of information; with transformers we will operate over tokens rather than over neurons

array of **neurons**



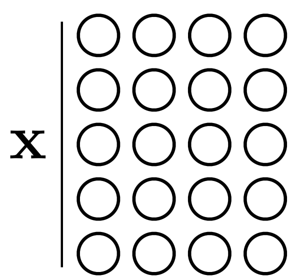
array of **tokens**



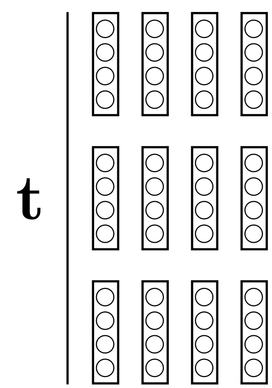
A new data structure: Tokens

- A **token** is just transformer lingo for a vector of neurons
- But the connotation is that a token is an encapsulated bundle of information; with transformers we will operate over tokens rather than over neurons

array of **neurons**



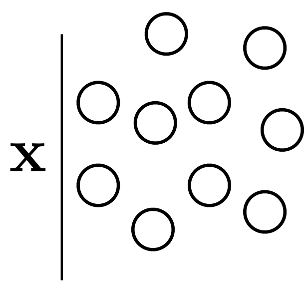
array of **tokens**



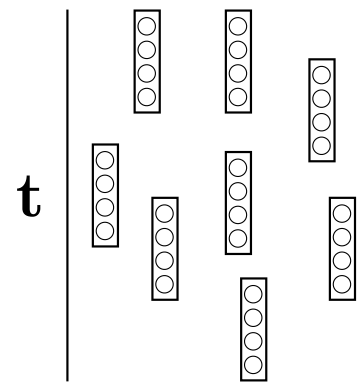
A new data structure: Tokens

- A **token** is just transformer lingo for a vector of neurons
- But the connotation is that a token is an encapsulated bundle of information; with transformers we will operate over tokens rather than over neurons

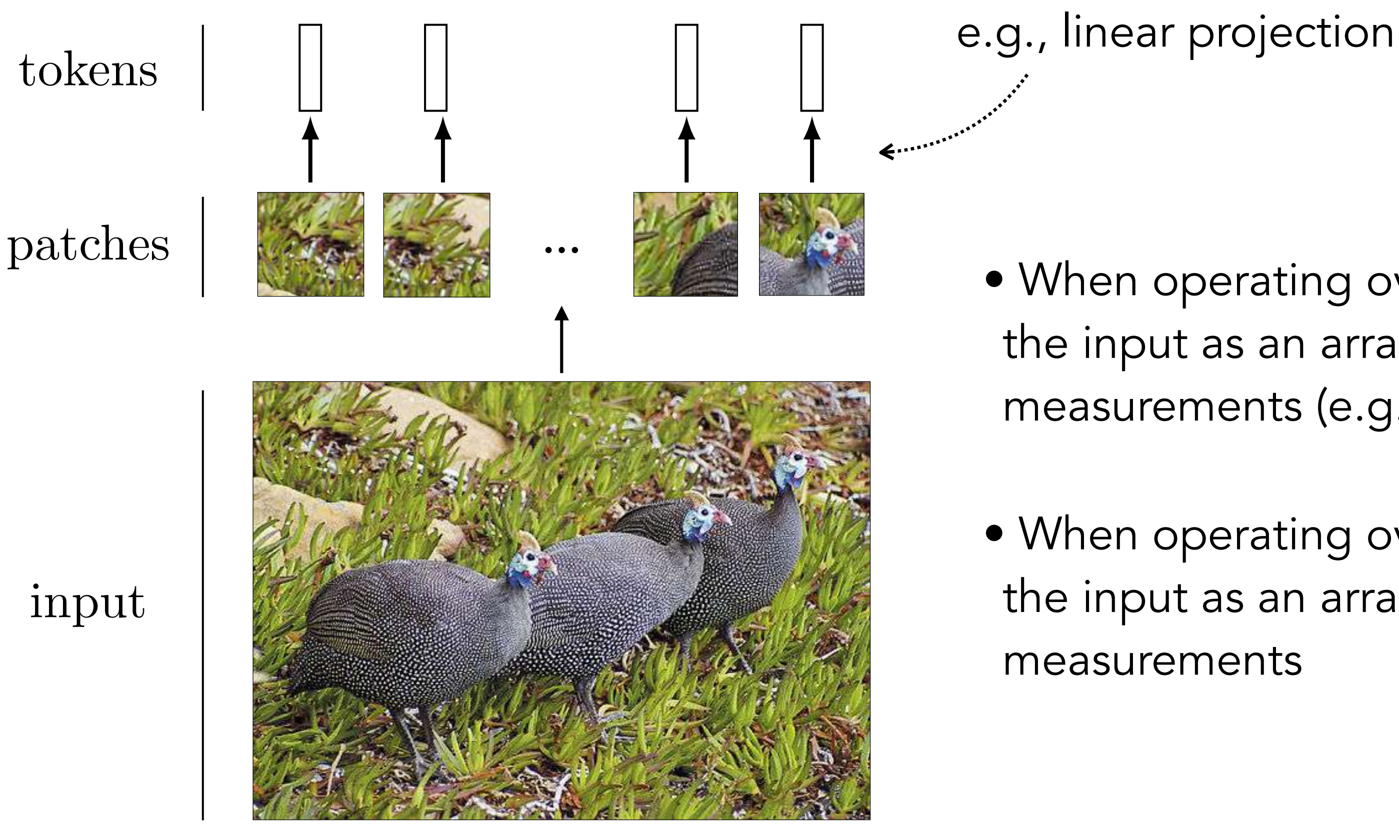
set of **neurons**



set of **tokens**



Tokenizing the input data

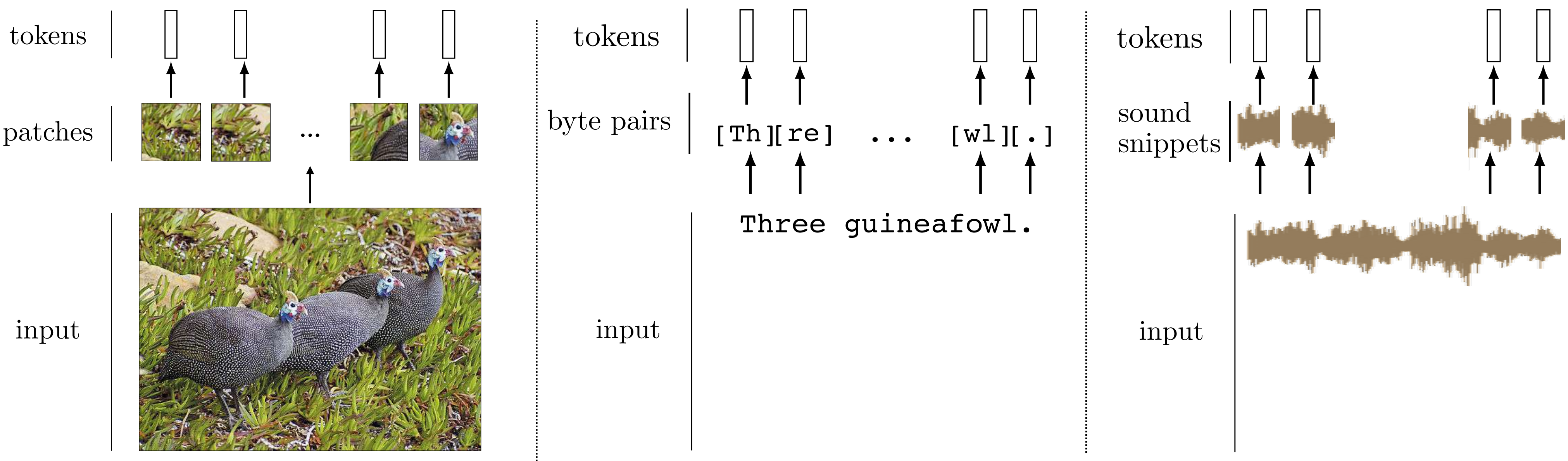


- When operating over *neurons*, we represent the input as an array of scalar-valued measurements (e.g., pixels)
- When operating over *tokens*, we represent the input as an array of vector-valued measurements

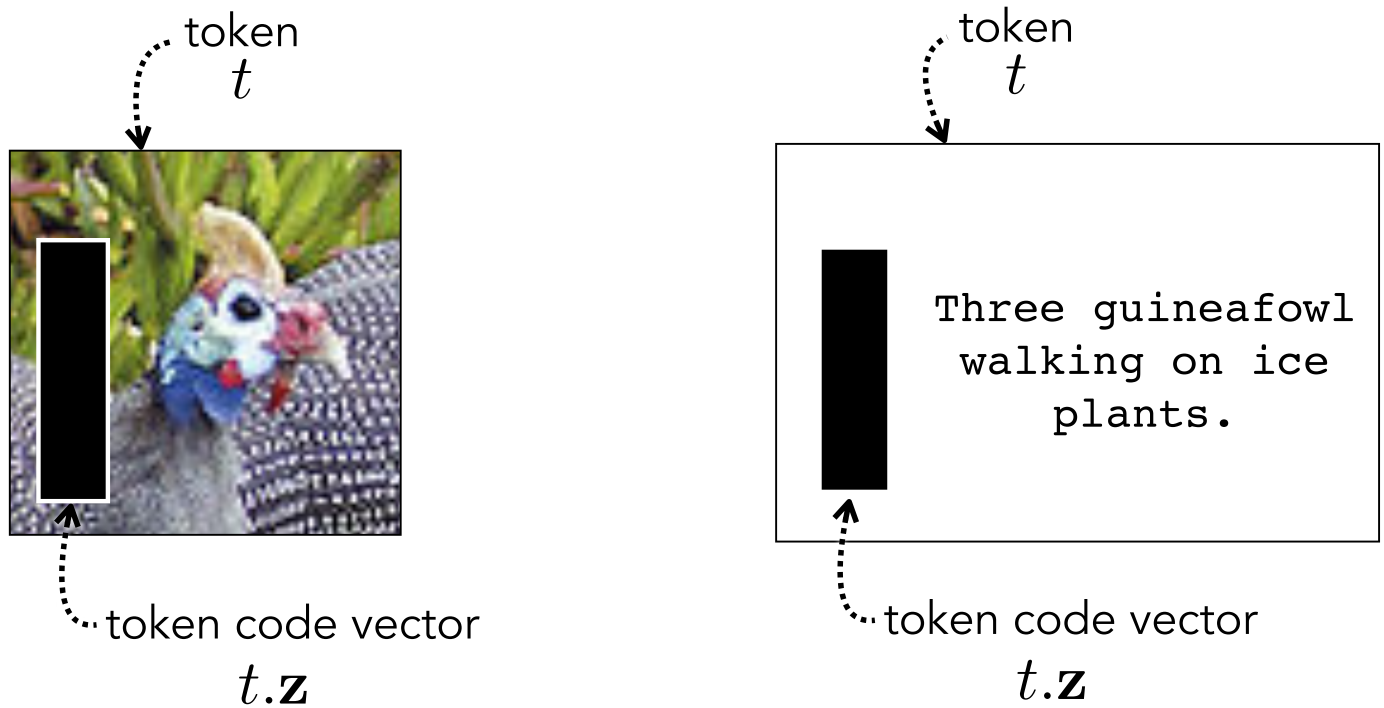
Tokenizing the input data

You can tokenize anything.

General strategy: chop the input up into chunks, project each chunk to a vector.

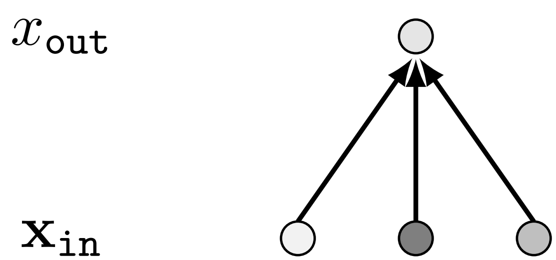


Tokenizing the input data



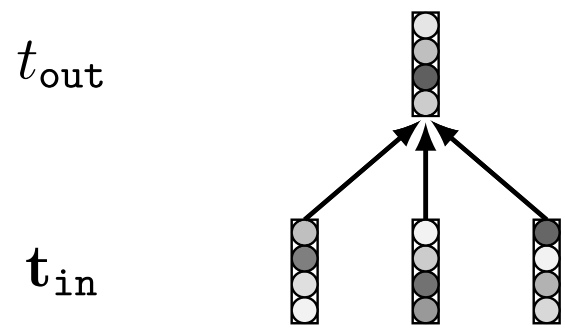
Linear combination of tokens

Linear combination of neurons



$$x_{\mathtt{out}} = \sum_{i=1}^{N} w_i x_{\mathtt{in}_i}$$

Linear combination of tokens



$$t_{\mathtt{out}}.\mathbf{z} = \sum_{i=1}^{N} w_i t_{\mathtt{in}_i}.\mathbf{z}$$

Token-wise nonlinearity

$$\mathbf{x}_{\texttt{out}} = [\texttt{relu}(x_1), \dots, \texttt{relu}(x_N)]$$

$$\mathbf{t}_{\texttt{out}} = [F_\theta(t_1.\mathbf{z}), \dots, F_\theta(t_N.\mathbf{z})]$$

F is typically an Multilayer Perceptron
(MLP)
(aka. fully-connected neural network)

Equivalent to a CNN with 1x1 kernels
run over token sequence

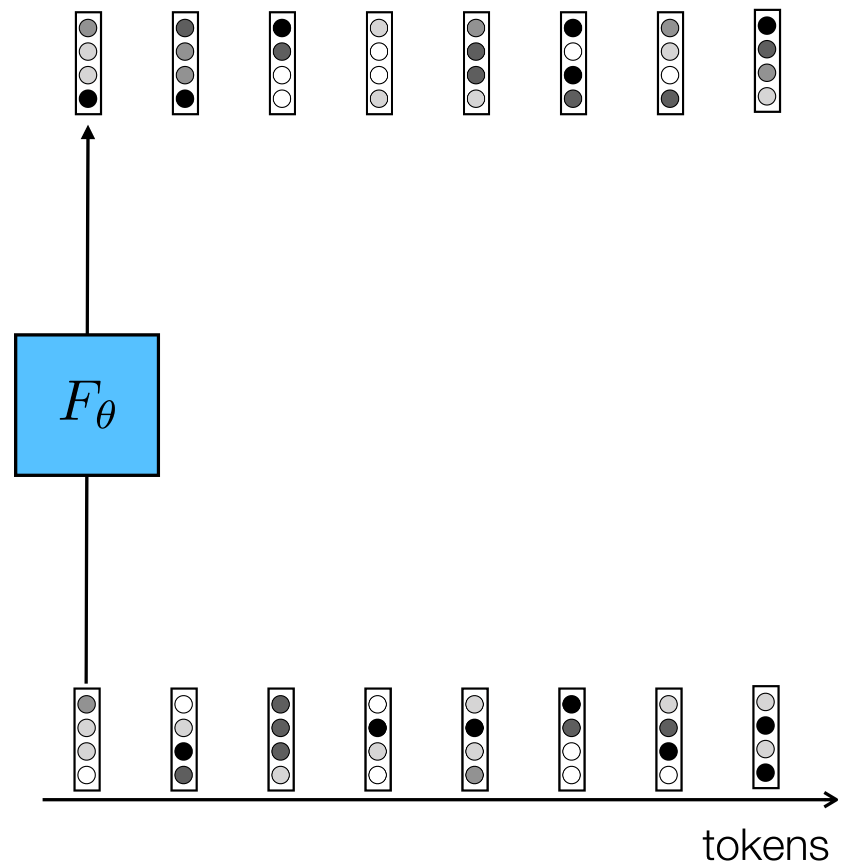
Token-wise nonlinearity

$$\mathbf{x}_{\texttt{out}} = [\texttt{relu}(x_1), \ldots, \texttt{relu}(x_N)]$$

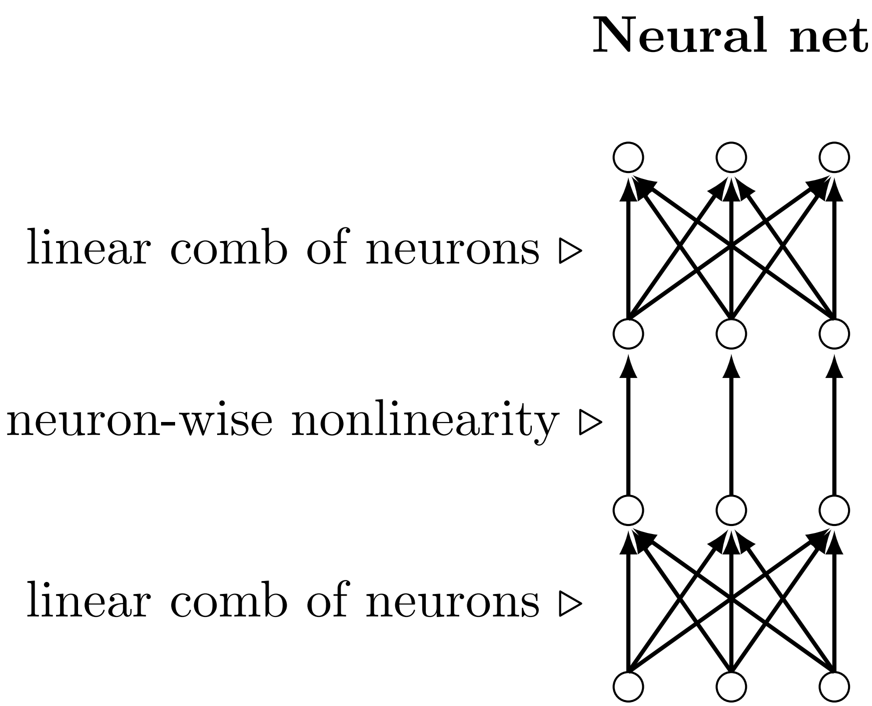
$$\mathbf{t}_{\texttt{out}} = [F_\theta(t_1.\mathbf{z}), \ldots, F_\theta(t_N.\mathbf{z})]$$

F is typically an MLP

Equivalent to a CNN with 1x1 kernels run over token sequence

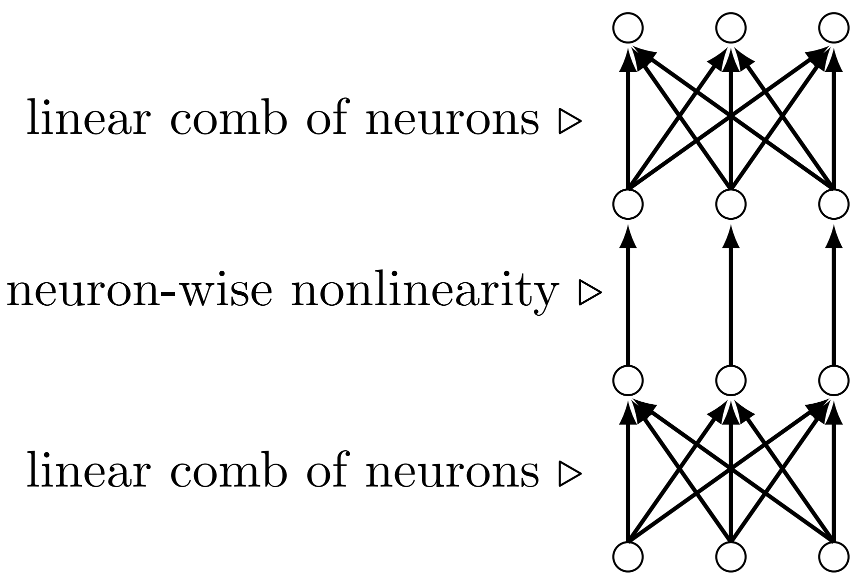


Token nets

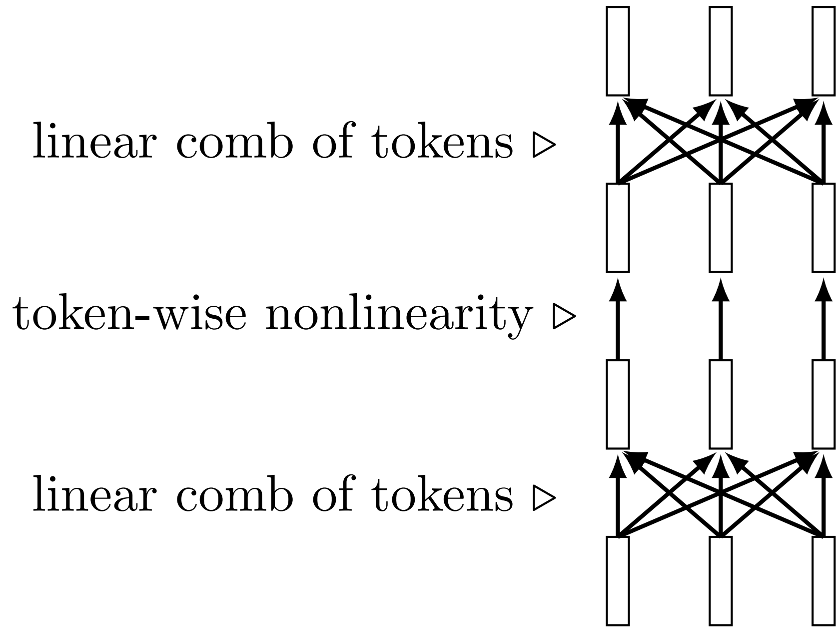

Neural net
linear comb of neurons ▷
neuron-wise nonlinearity ▷
linear comb of neurons ▷

Token nets

Neural net



Token net



Idea #2: attention

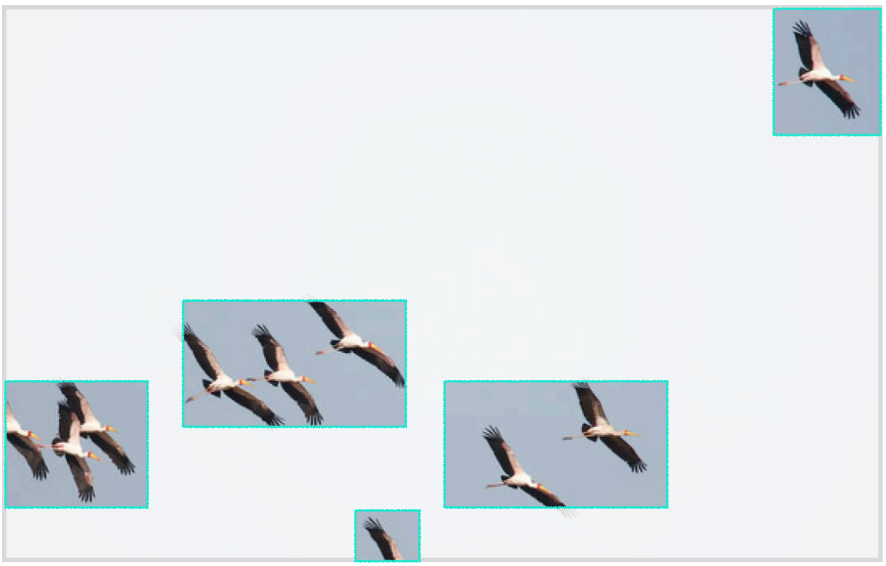
A limitation of CNNs

How many birds are in this image?

Is the top right bird the same species as the bottom left bird?

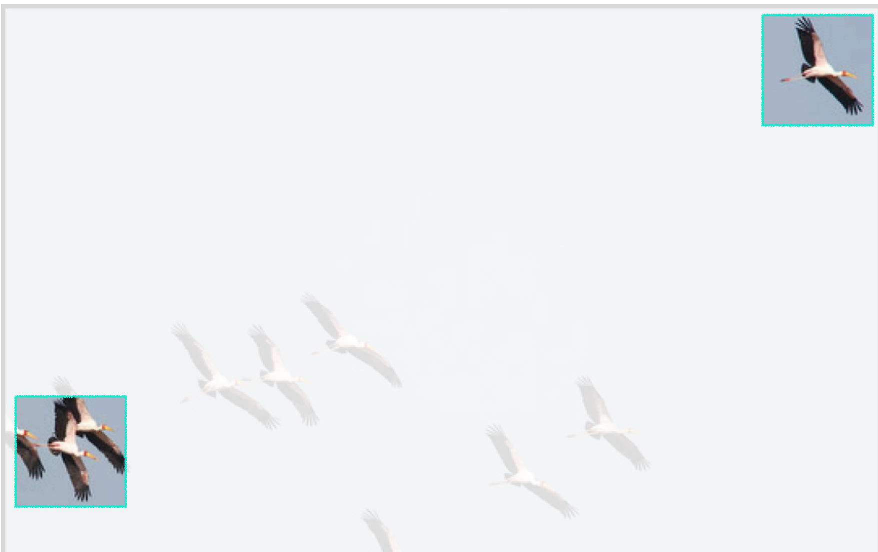
CNNs are built around the idea of locality, and are not well-suited to modeling long distance relationships

What is attention?



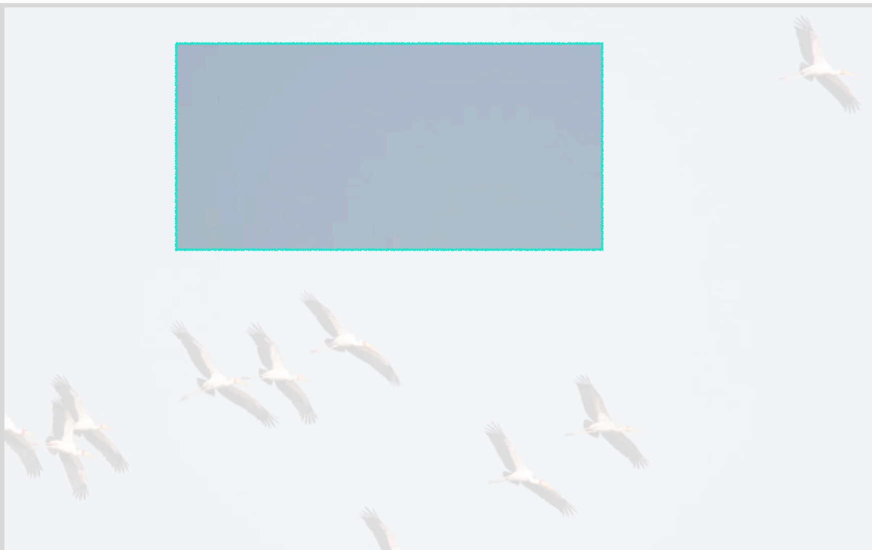
How many birds are in this image?

What is attention?



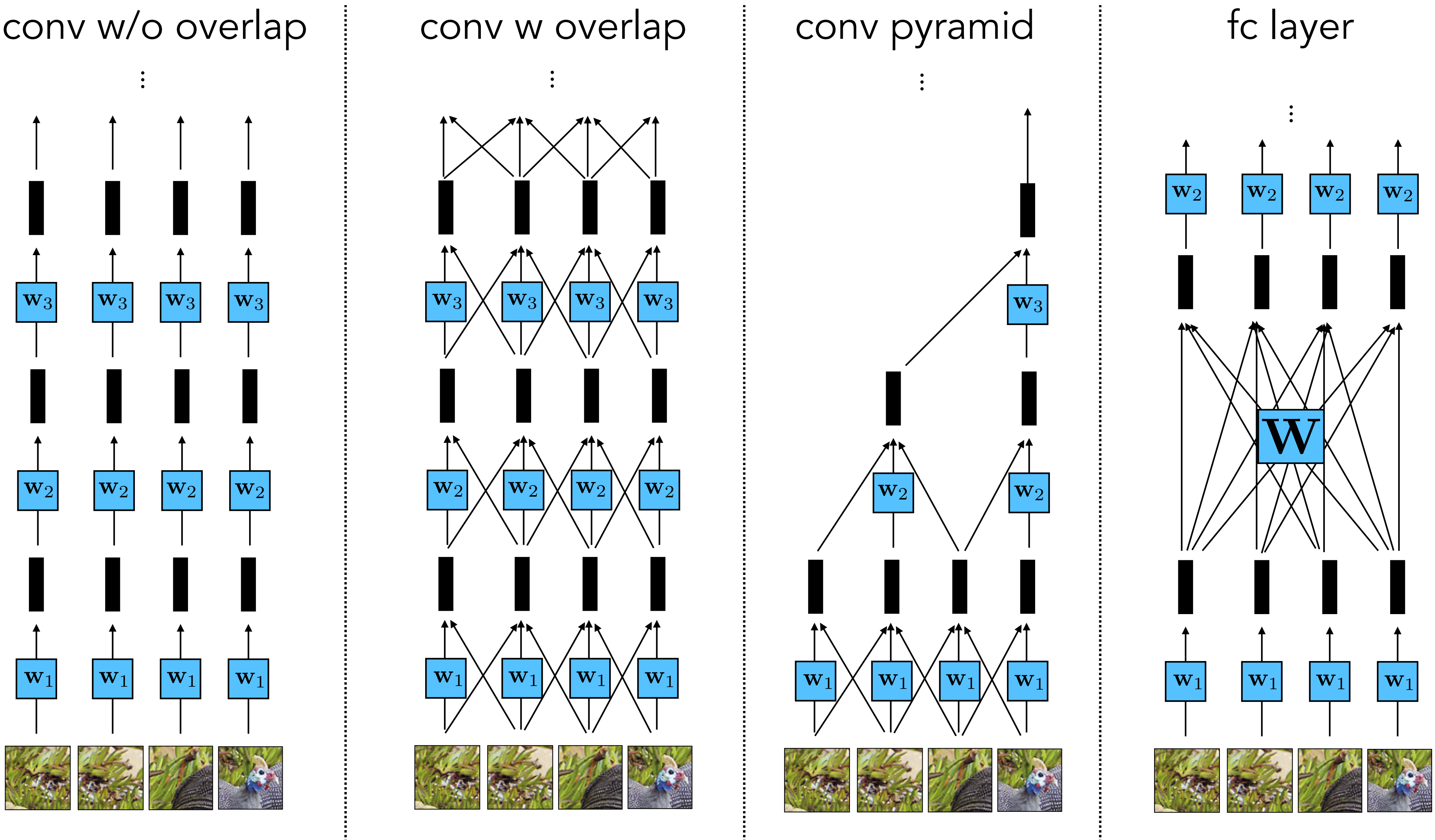
Is the top right bird the same species as the bottom left bird?

What is attention?



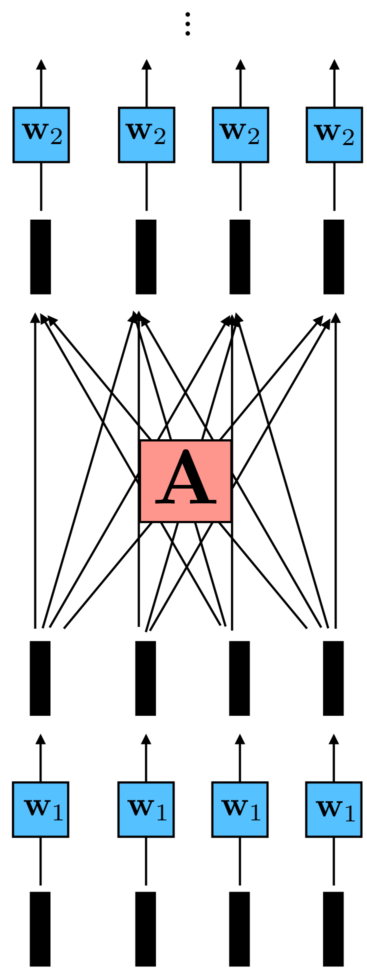
What's the color of the sky?

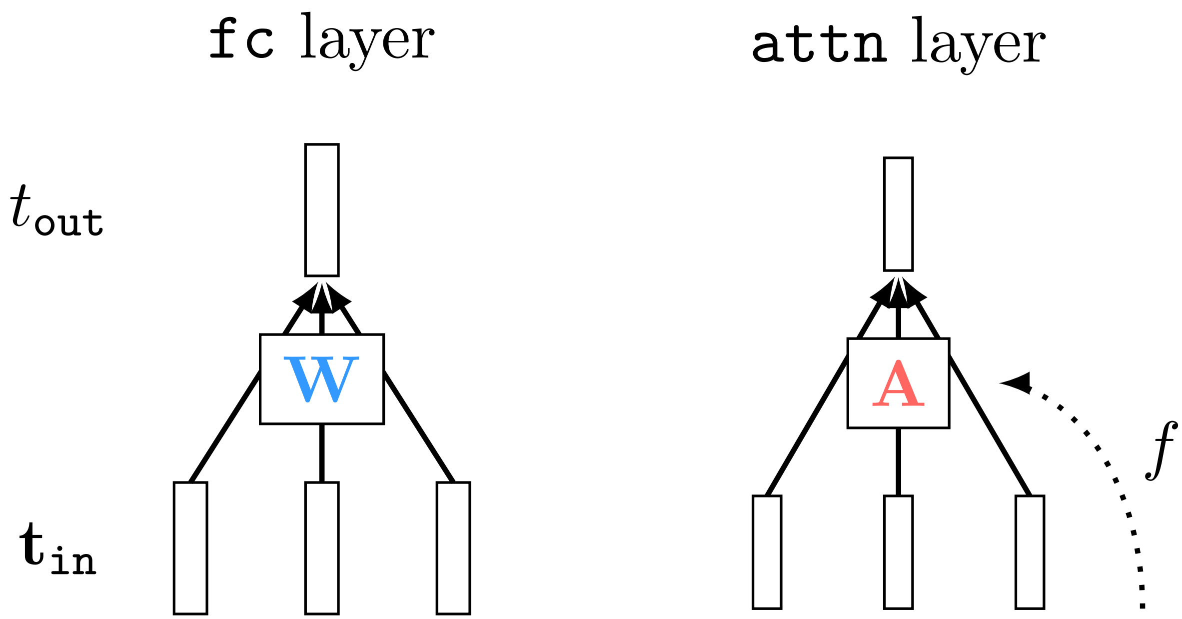
Different ways of aggregating information over space



conv w/o overlap
conv w overlap
conv pyramid
fc layer
w1
w2
w3
W

Attention layer





$$\mathbf{A} = f(\ldots) \quad \triangleleft \text{ attention}$$

$$\mathbf{Z}_{\texttt{out}} = \mathbf{A}\mathbf{Z}_{\texttt{in}}$$

$$\mathbf{Z}_{\texttt{in}} = \begin{pmatrix} t_1.\mathbf{z}^T \\ \vdots \\ t_N.\mathbf{z}^T \end{pmatrix}$$

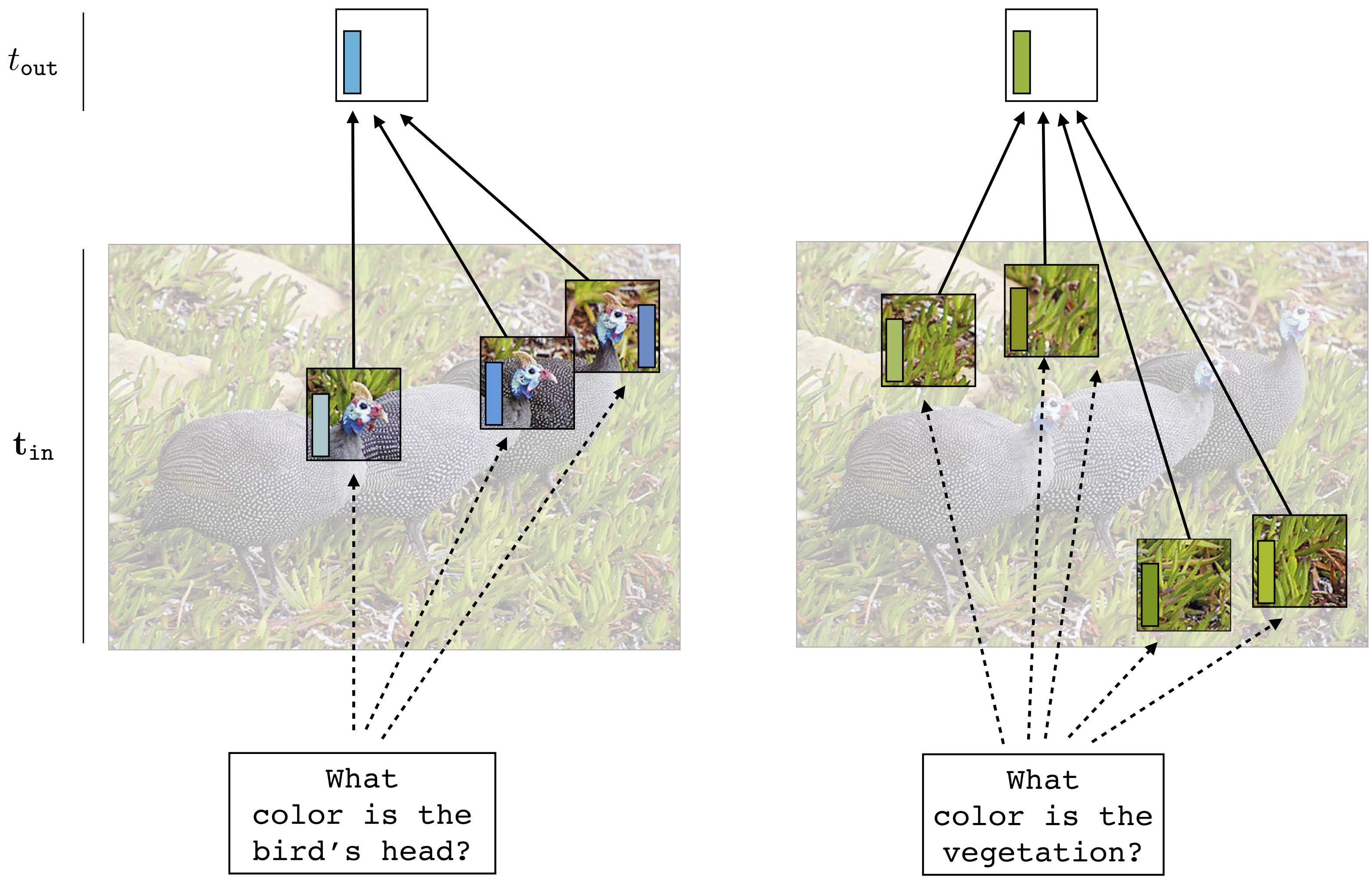
W is free *parameters*.

A is a function of some input *data*. The data tells us which tokens to attend to (assign high weight in weighted sum)

$t_{\mathtt{out}}$

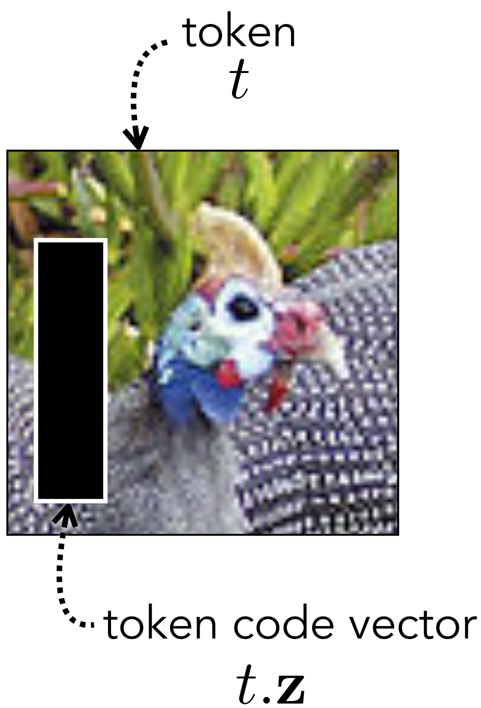
$\mathbf{t}_{\mathtt{in}}$

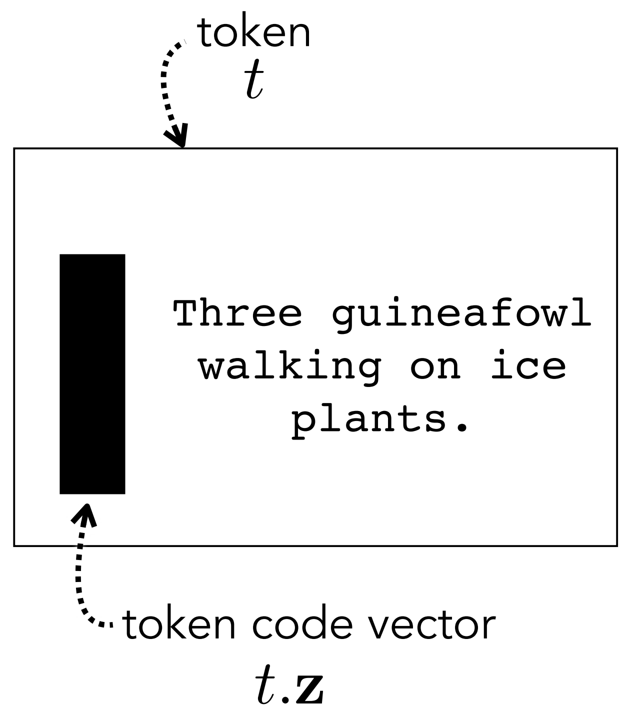
```
What
color is the
bird's head?
```



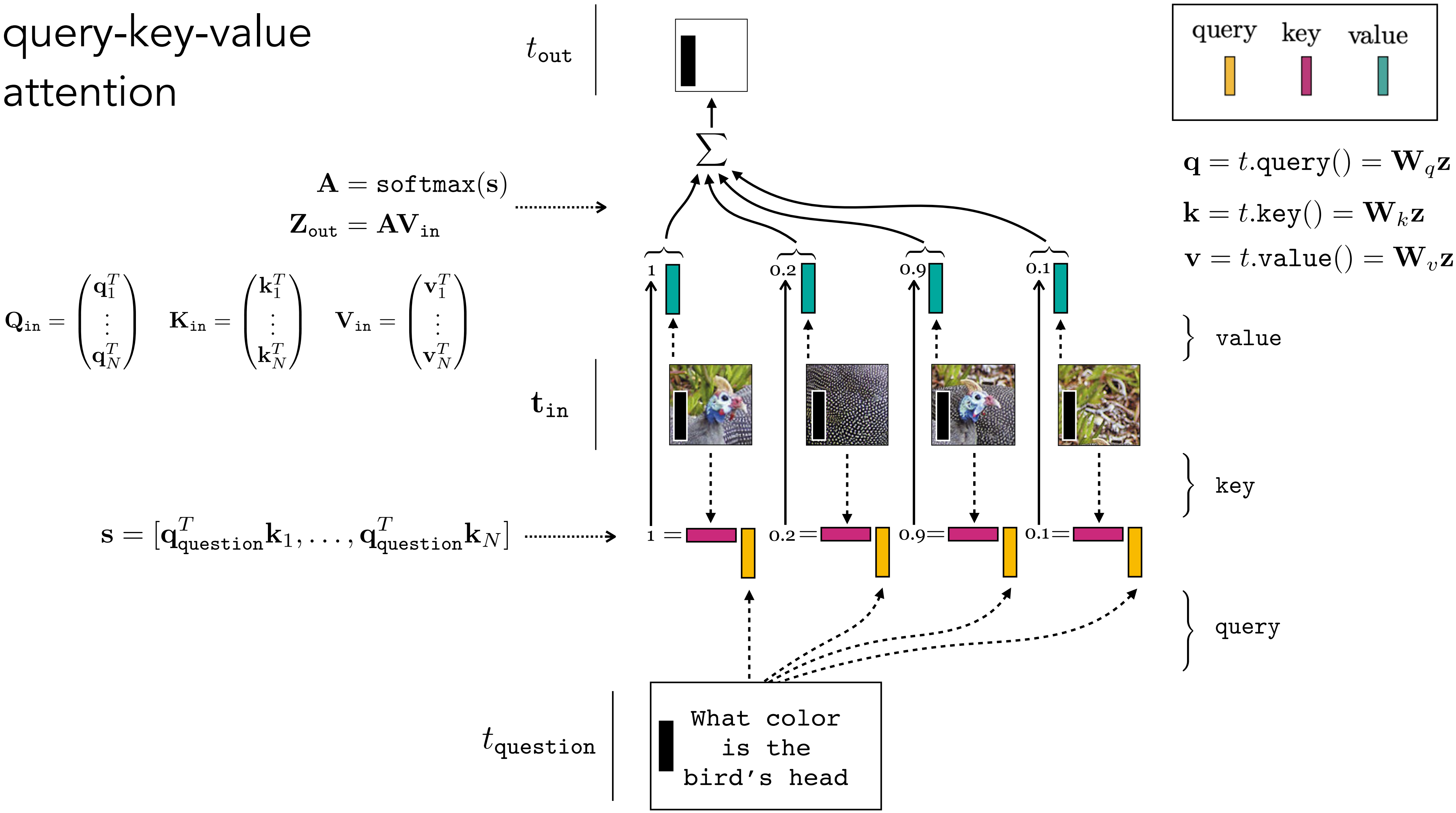
t_{out}
$\mathbf{t}_{\text{in}}$
What color is the bird's head?
What color is the vegetation?

Notation reminder

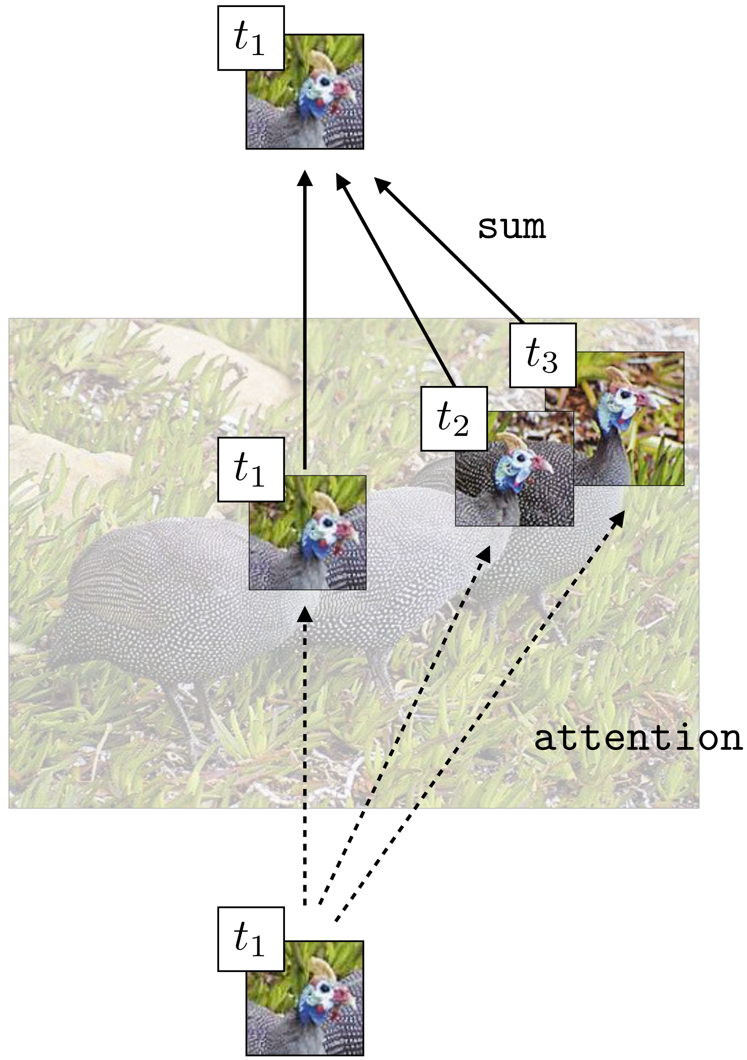




query-key-value attention



Self-attention

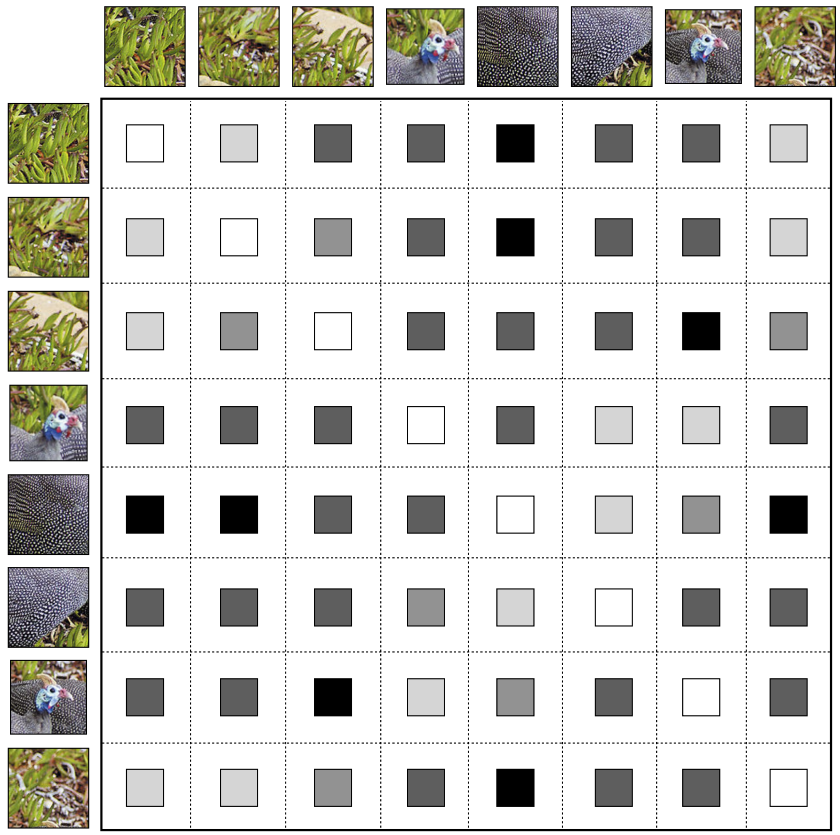


Attention maps in a trained transformer

["DINO", Caron et all. 2021]

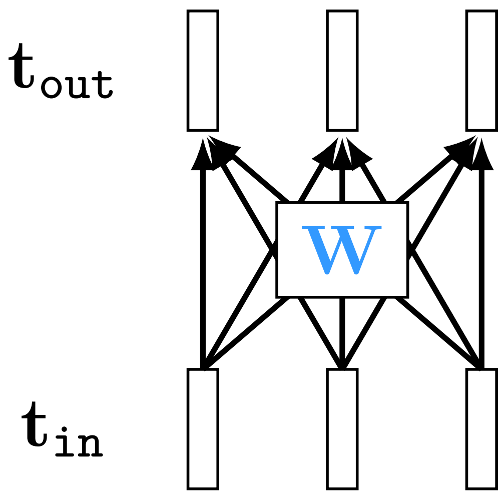
Self-attention

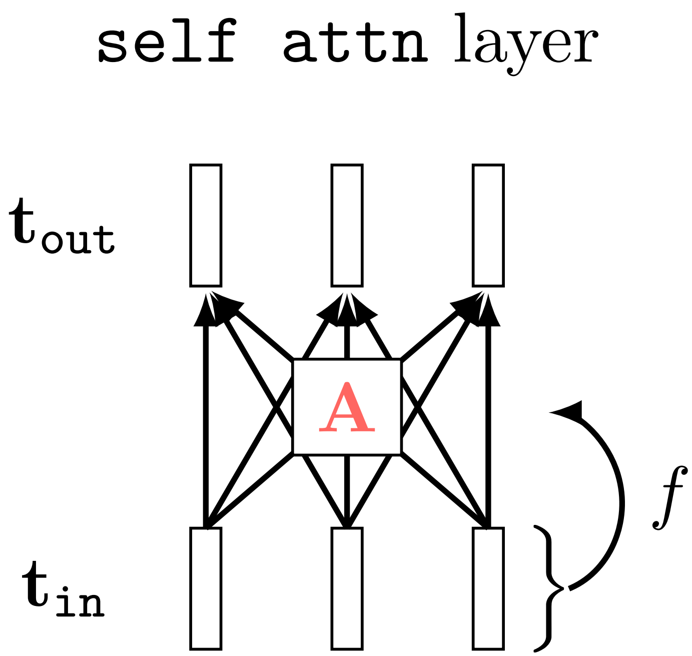
Example of attention if query() and key() are the identity function



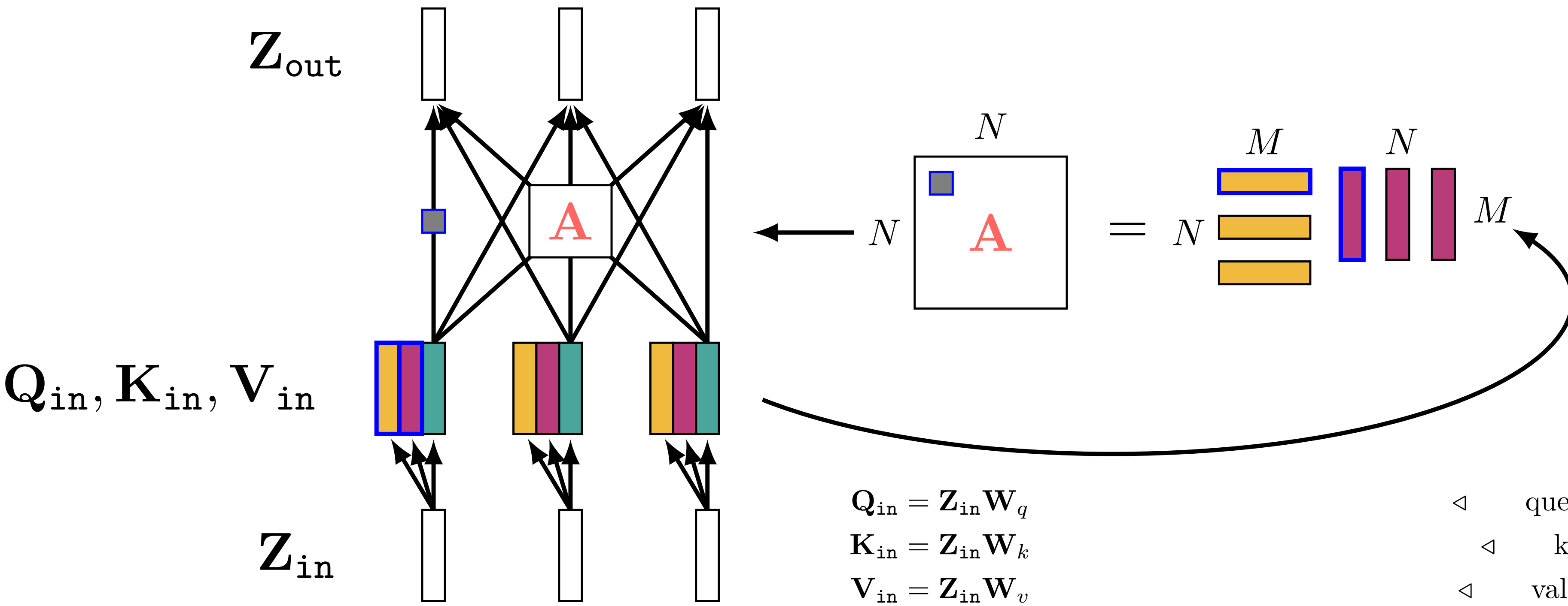
—> just a Gram matrix (similarity matrix) over tokens!
Essentially: clusters similar tokens

$\texttt{fc}$ layer




self attn layer
$\mathbf{t}_{\texttt{out}}$
A
f
$\mathbf{t}_{\texttt{in}}$

$\texttt{self attn}$ layer (expanded)



$$\mathbf{Q}_{\texttt{in}} = \mathbf{Z}_{\texttt{in}}\mathbf{W}_q \qquad \triangleleft \quad \text{query matrix}$$

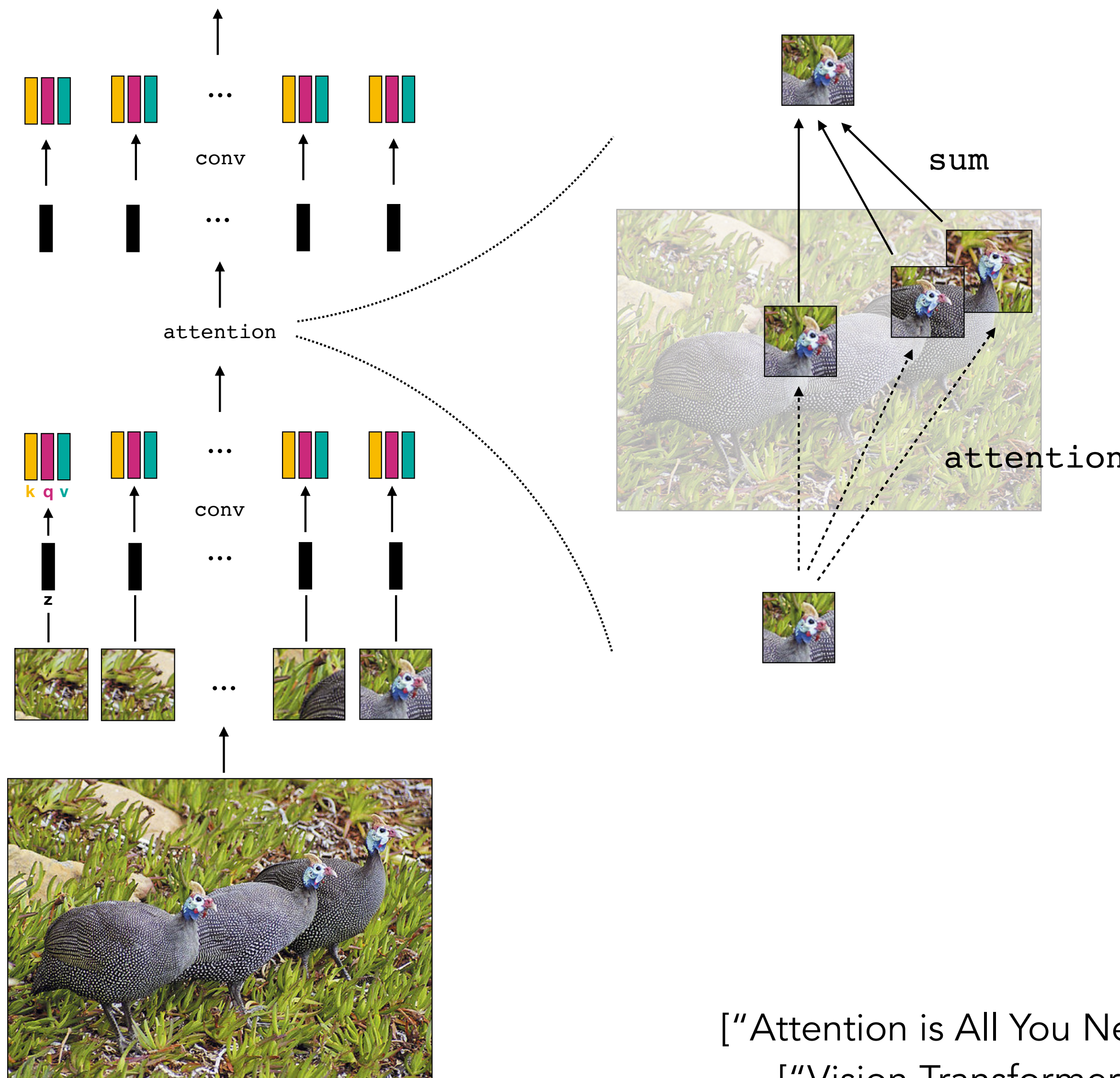
$$\mathbf{K}_{\texttt{in}} = \mathbf{Z}_{\texttt{in}}\mathbf{W}_k \qquad \triangleleft \quad \text{key matrix}$$

$$\mathbf{V}_{\texttt{in}} = \mathbf{Z}_{\texttt{in}}\mathbf{W}_v \qquad \triangleleft \quad \text{value matrix}$$

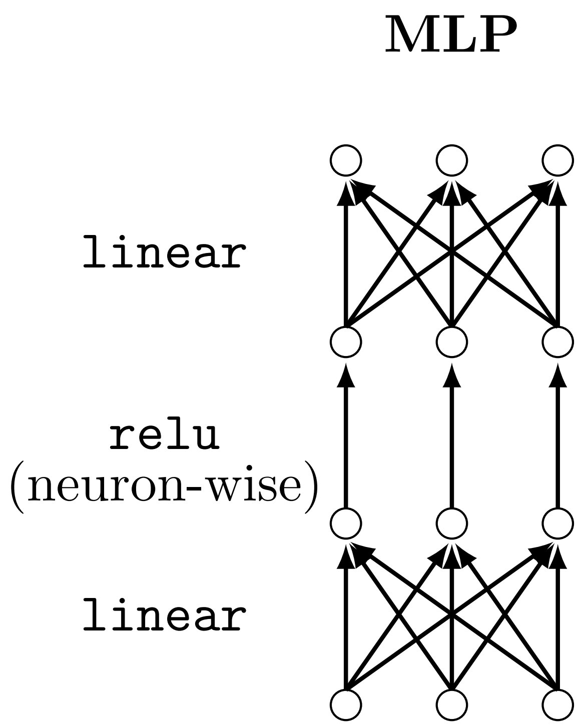
$$\mathbf{A} = f(\mathbf{t}_{\texttt{in}}) = \texttt{softmax}\left(\frac{\mathbf{Q}_{\texttt{in}}\mathbf{K}_{\texttt{in}}^T}{\sqrt{d}}\right) \qquad \triangleleft \quad \text{attention matrix}$$

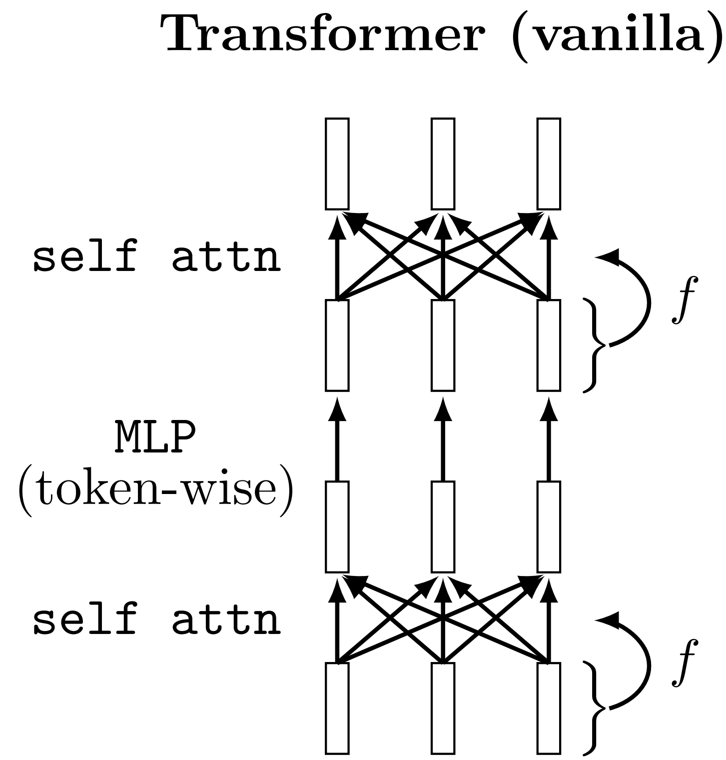
$$\mathbf{Z}_{\texttt{out}} = \mathbf{A}\mathbf{V}_{\texttt{in}}$$

Transformer (simplified)



["Attention is All You Need", Vaswani et al. 2017]
["Vision Transformer", Dosovitskiy et al. 2020]


MLP
linear
relu
(neuron-wise)
linear


Transformer (vanilla)
self attn
f
MLP
(token-wise)
self attn
f

Multihead self-attention (MSH)

Rather than having just one way of attending, why not have k?

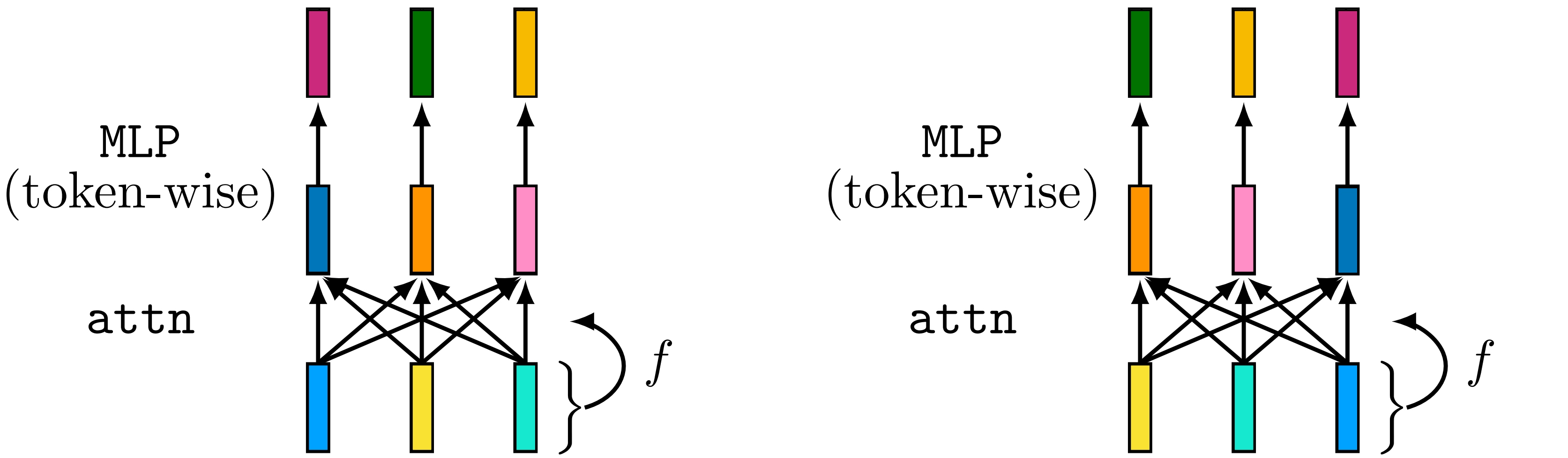
Each gets its own parameterized query(), key(), value() functions.

Run them all in parallel, then (weighted) sum the output token code vectors

$$\mathbf{Z} = \begin{pmatrix} \texttt{attn}_1(\mathbf{t}_{\texttt{in}}).\mathbf{z}^T \\ \vdots \\ \texttt{attn}_k(\mathbf{t}_{\texttt{in}}).\mathbf{z}^T \end{pmatrix}$$

$$\mathbf{t}_{\texttt{out}}.\mathbf{z} = \mathbf{W}\mathbf{Z} \qquad \triangleleft \quad \mathbf{W} \in \mathbb{R}^{M_2 \times k M_1}$$

Permutation equivariance



$$\texttt{attn}(\texttt{permute}(\mathbf{t}_{\texttt{in}})) = \texttt{permute}(\texttt{attn}(\mathbf{t}_{\texttt{in}}))$$

$$\texttt{tokenMLP}(\texttt{permute}(\mathbf{t}_{\texttt{in}})) = \texttt{tokenMLP}(\texttt{attn}(\mathbf{t}_{\texttt{in}}))$$

$$\downarrow$$

$$\texttt{transformer}(\texttt{permute}(\mathbf{t}_{\texttt{in}})) = \texttt{permute}(\texttt{transformer}(\mathbf{t}_{\texttt{in}}))$$

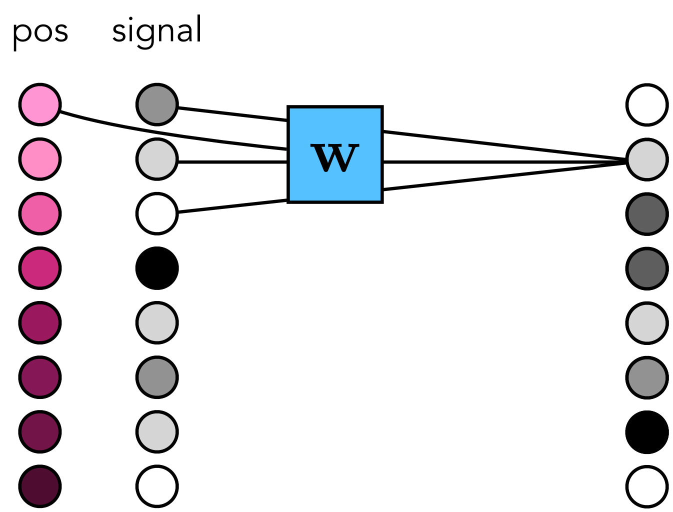
Set2Set

Idea #3: positional encoding

What if you *don't* want to be shift invariant?

1. Use an architecture that is not shift invariant (e.g., MLP)

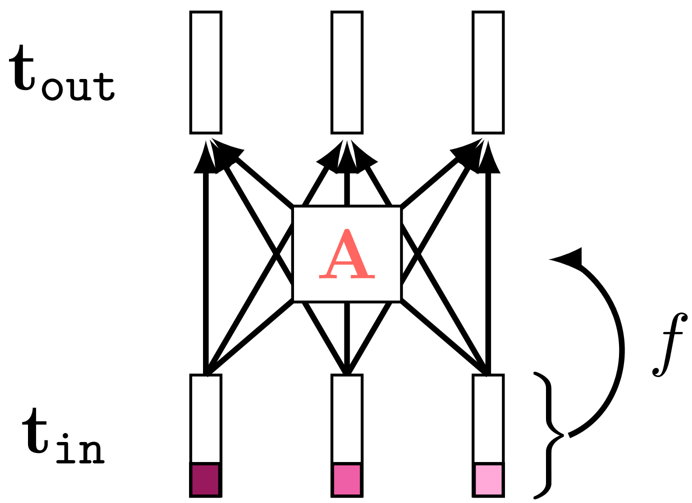
2. Add location information to the *input* to the convolutional filters — this is called **positional encoding**



What if you *don't* want to be permutation invariant?

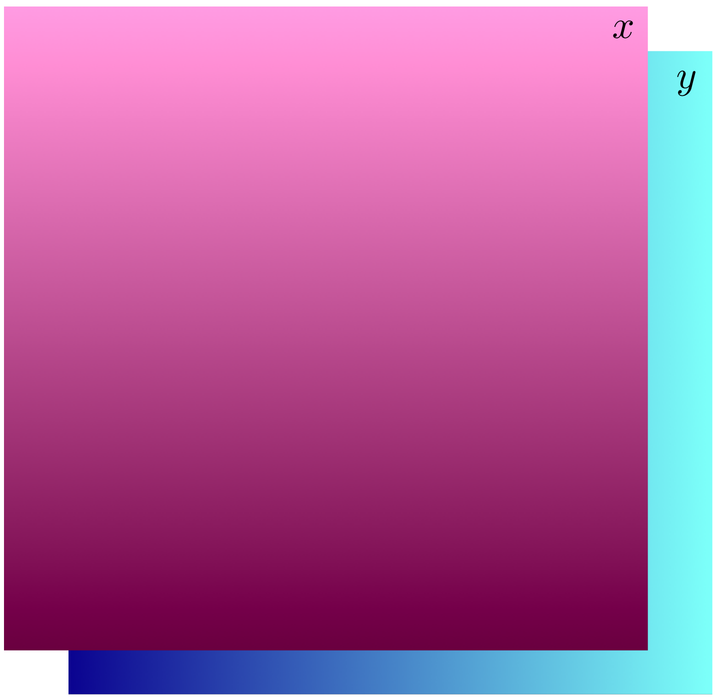
1. Use an architecture that is not permutation invariant (e.g., MLP)

2. Add location information to the token code vectors — this is called **positional encoding**



Neural Fields

Coordinates



$$\Phi : \mathbb{R}^2 \rightarrow \mathbb{R}$$

Field

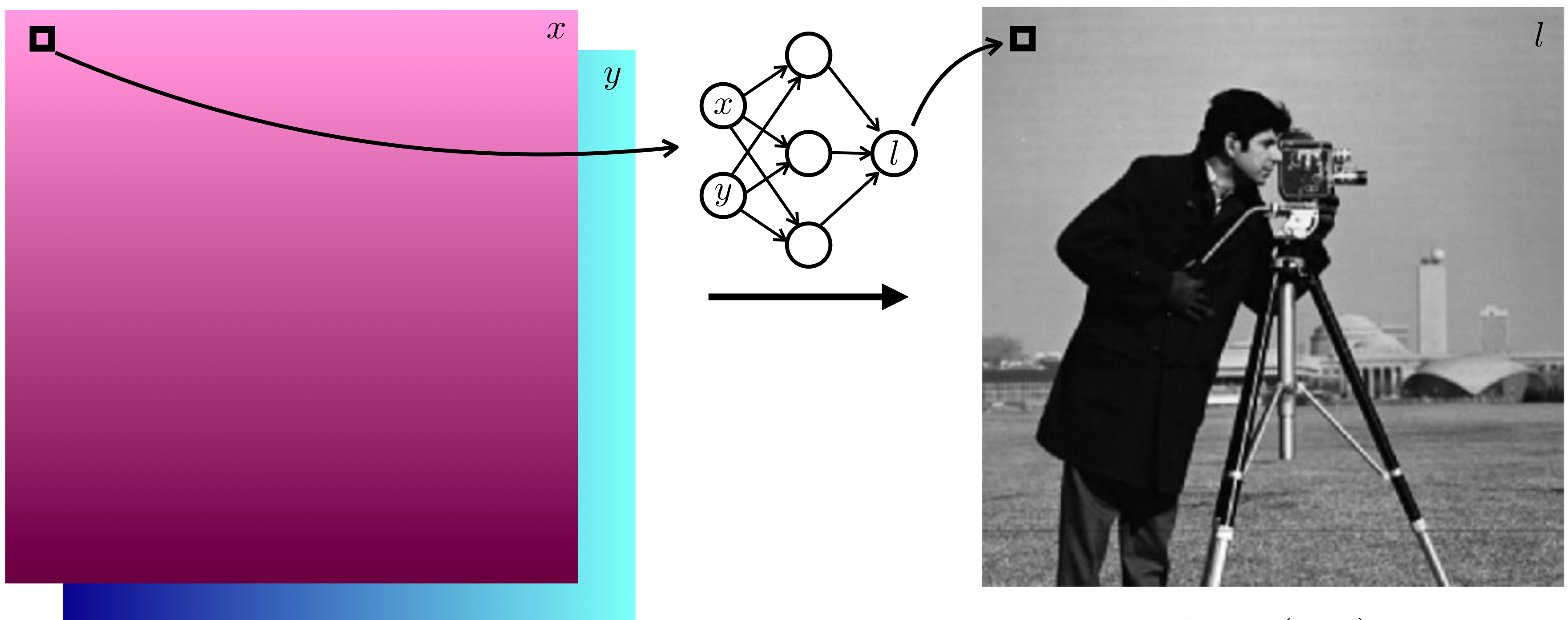
$$l = \Phi(x, y)$$

Neural Fields — SIREN

Conv net applied *per-pixel* to map from a coordinate grid to a color

Coordinates

Field



$$l = \Phi(x, y)$$

Can take continuous coordinates as input!

Continuous version of a convnet!

["SIREN", Sitzmann, Martel et al. 2020]

Some fancy architectures and applications

AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE

Alexey Dosovitskiy[*,†], **Lucas Beyer**[*], **Alexander Kolesnikov**[*], **Dirk Weissenborn**[*], **Xiaohua Zhai**[*], **Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, Neil Houlsby**[*,†]

[*]equal technical contribution, [†]equal advising

Google Research, Brain Team

{adosovitskiy, neilhoulsby}@google.com

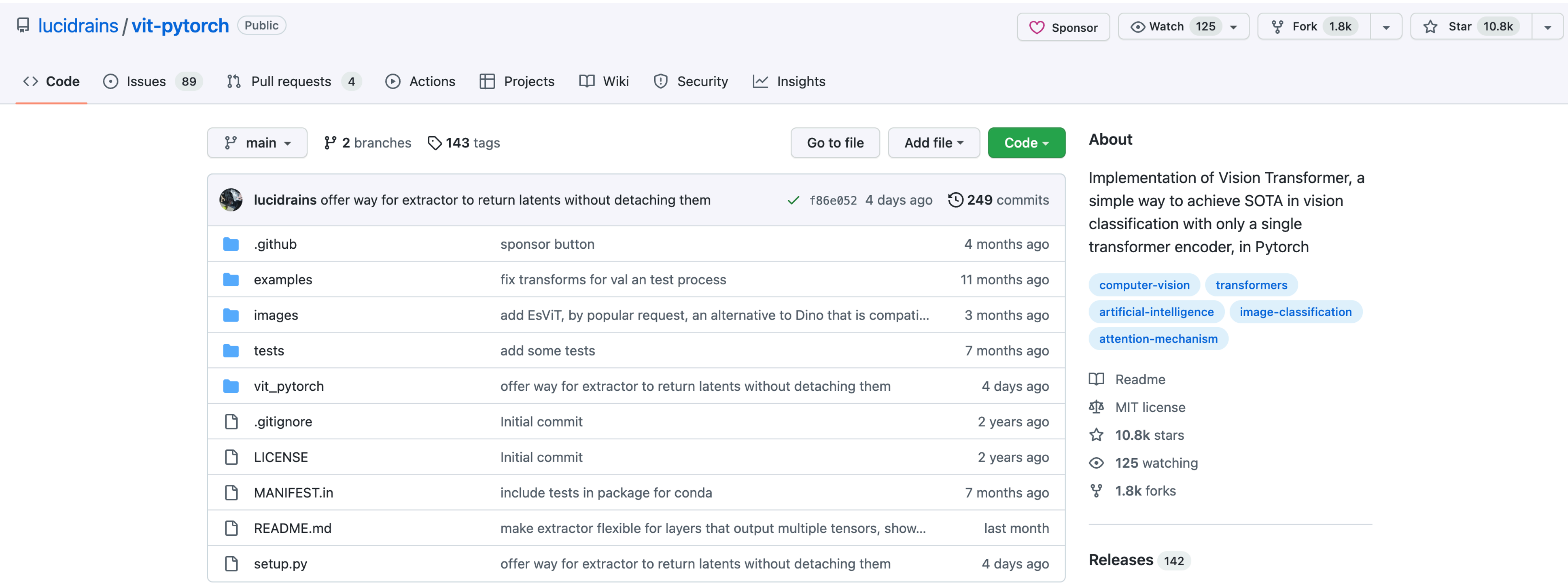
ABSTRACT

While the Transformer architecture has become the de-facto standard for natural language processing tasks, its applications to computer vision remain limited. In vision, attention is either applied in conjunction with convolutional networks, or used to replace certain components of convolutional networks while keeping their overall structure in place. We show that this reliance on CNNs is not necessary and a pure transformer applied directly to sequences of image patches can perform very well on image classification tasks. When pre-trained on large amounts of data and transferred to multiple mid-sized or small image recognition benchmarks (ImageNet, CIFAR-100, VTAB, etc.), Vision Transformer (ViT) attains excellent results compared to state-of-the-art convolutional networks while requiring substantially fewer computational resources to train.[1]

1 INTRODUCTION

Self-attention-based architectures, in particular Transformers (Vaswani et al., 2017), have become the model of choice in natural language processing (NLP). The dominant approach is to pre-train on a large text corpus and then fine-tune on a smaller task-specific dataset (Devlin et al., 2019). Thanks to Transformers' computational efficiency and scalability, it has become possible to train models of unprecedented size, with over 100B parameters (Brown et al., 2020; Lepikhin et al., 2020). With the models and datasets growing, there is still no sign of saturating performance.

https://arxiv.org/abs/2010.11929



https://github.com/lucidrains/vit-pytorch

Attention Is All You Need

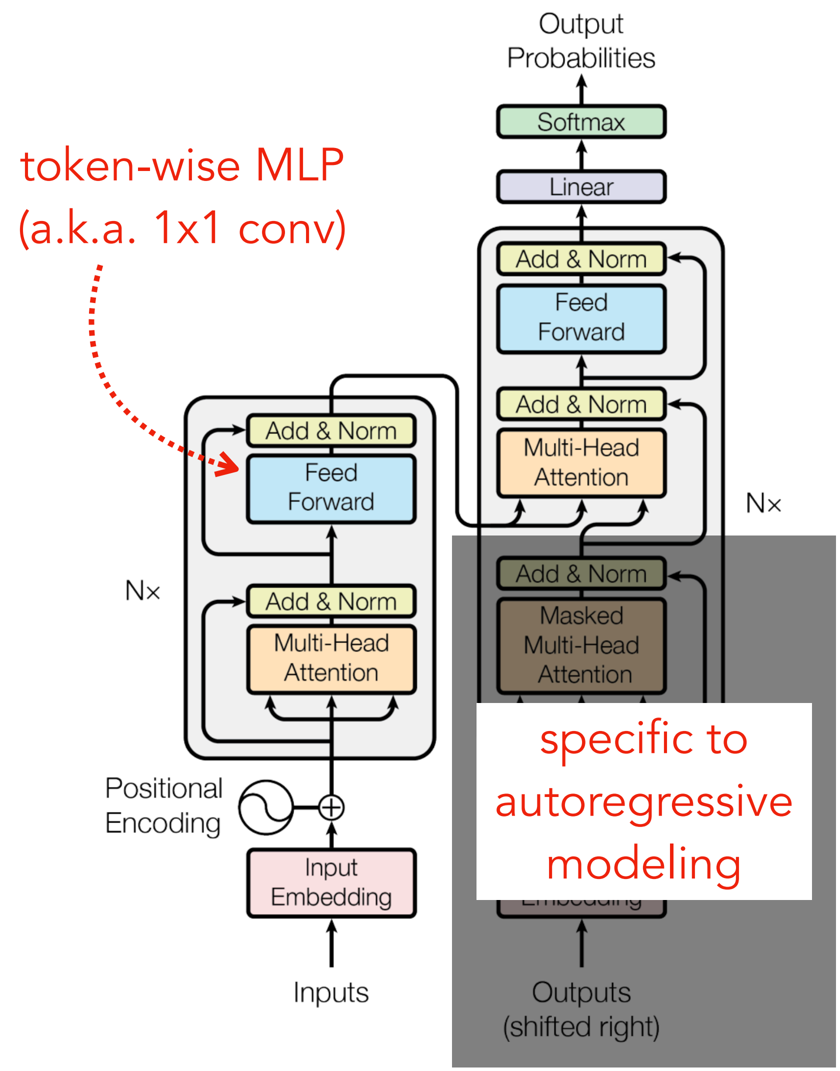
Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com

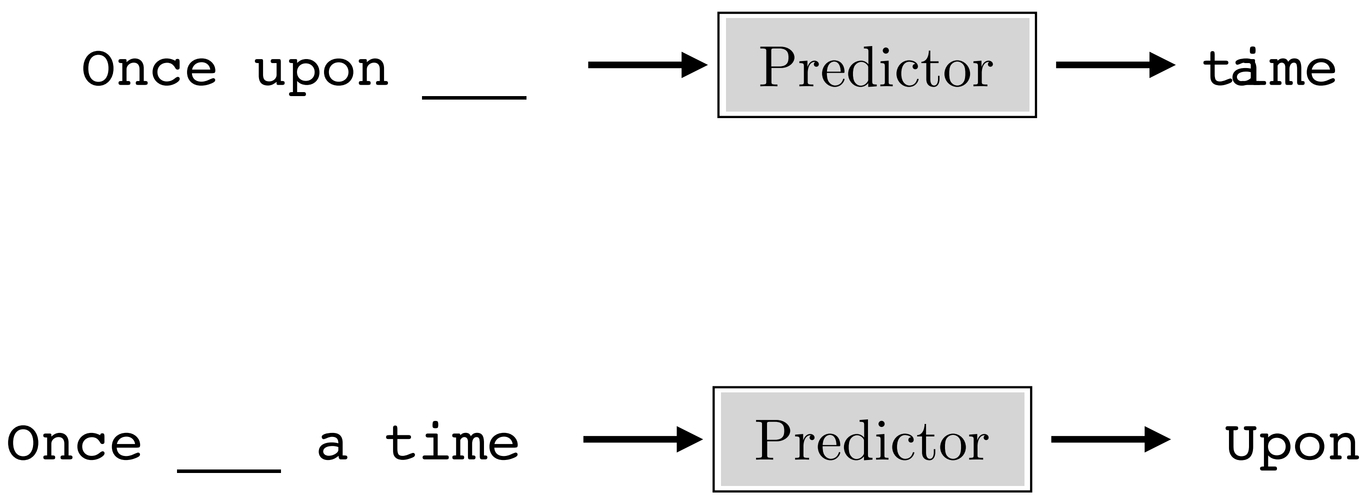
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

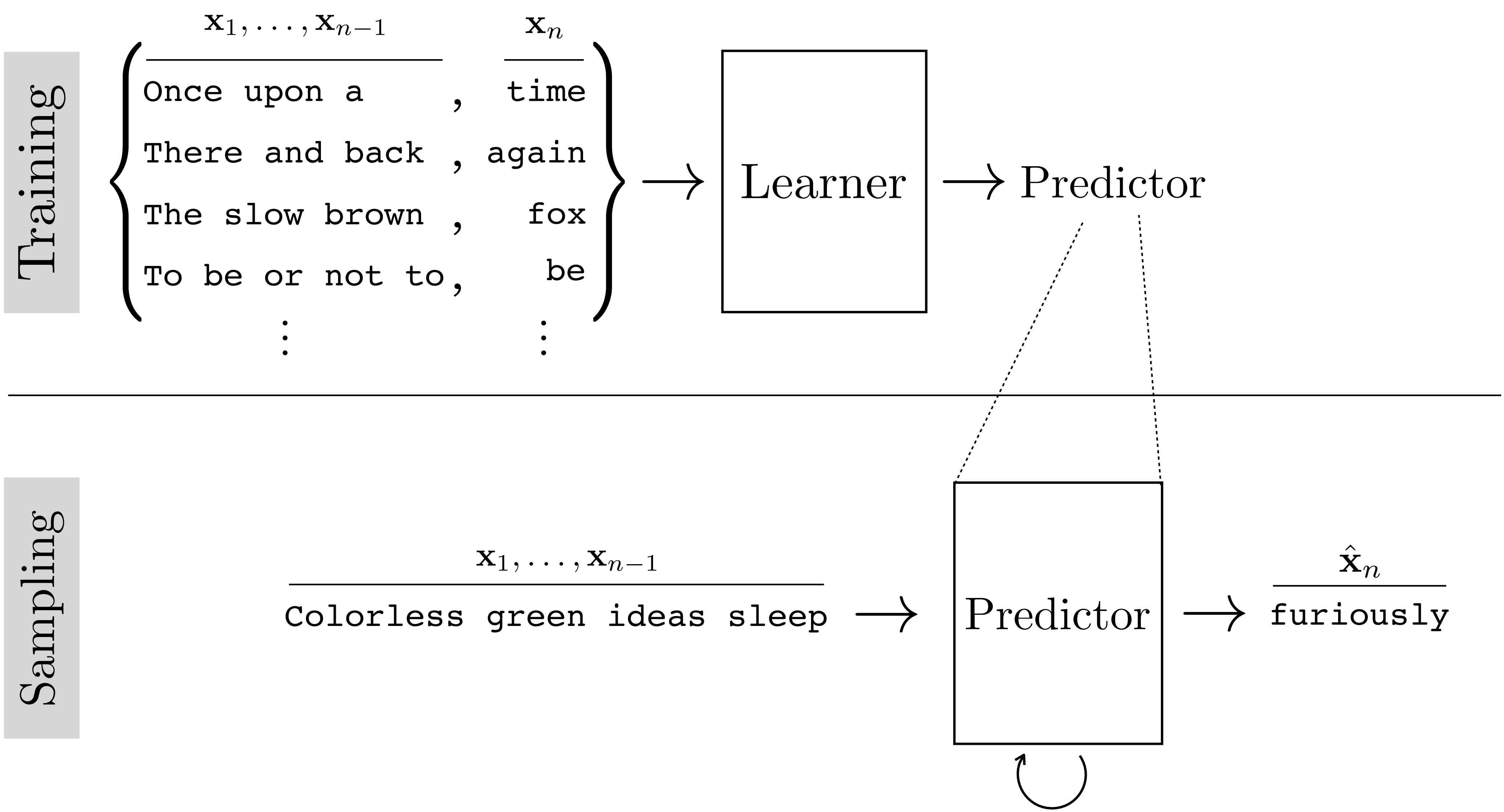
Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.



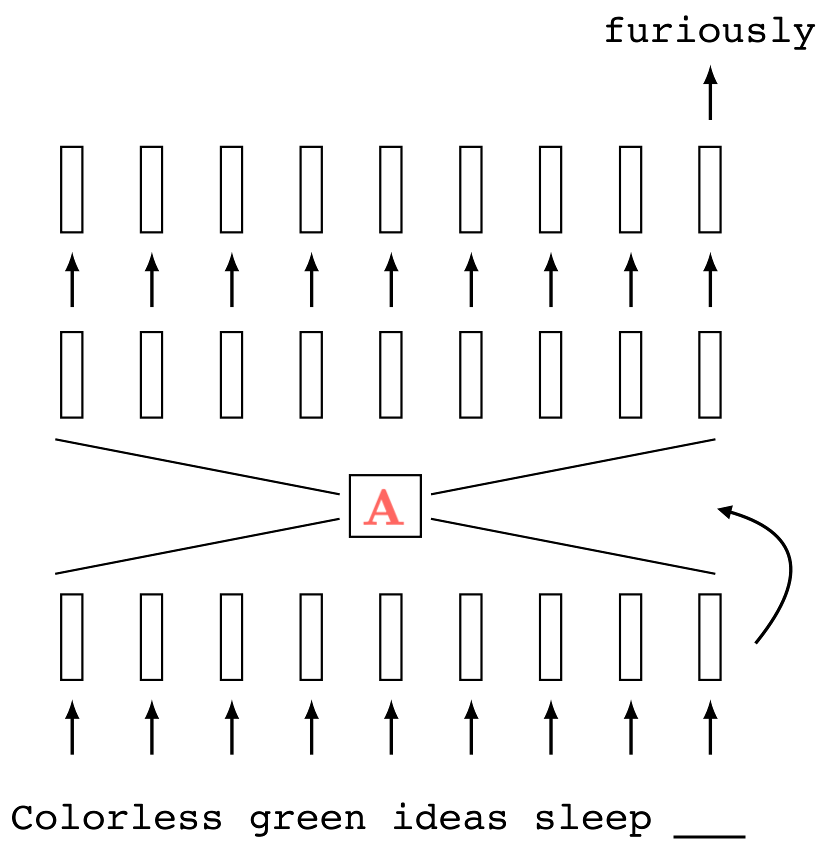
Autoregressive models



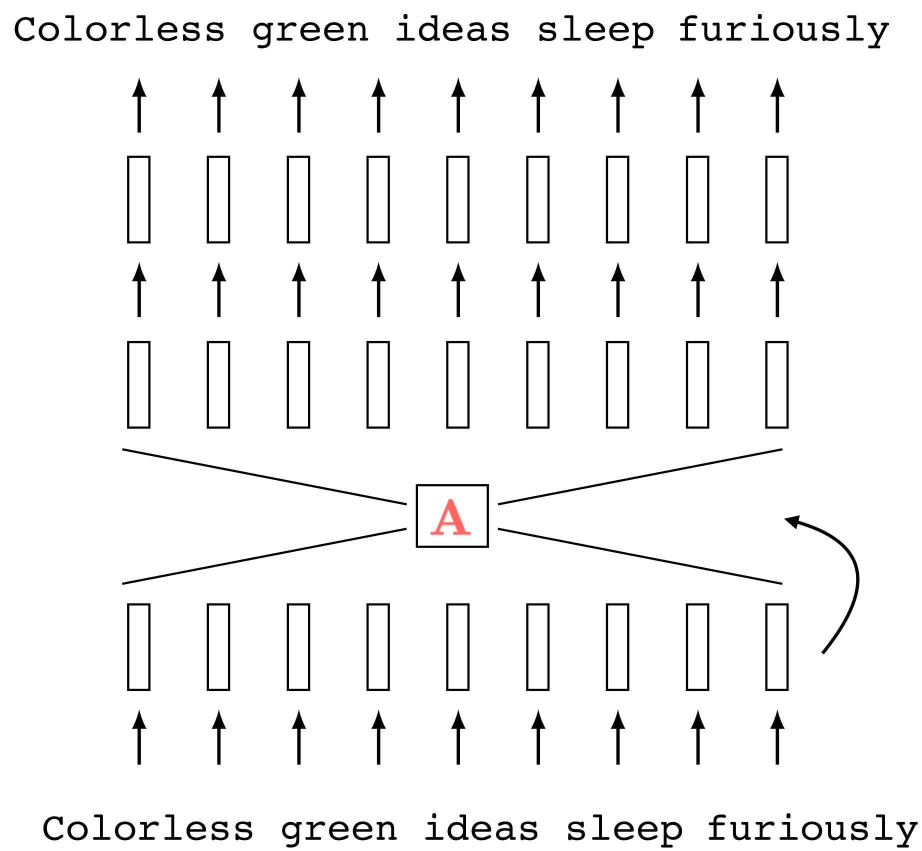


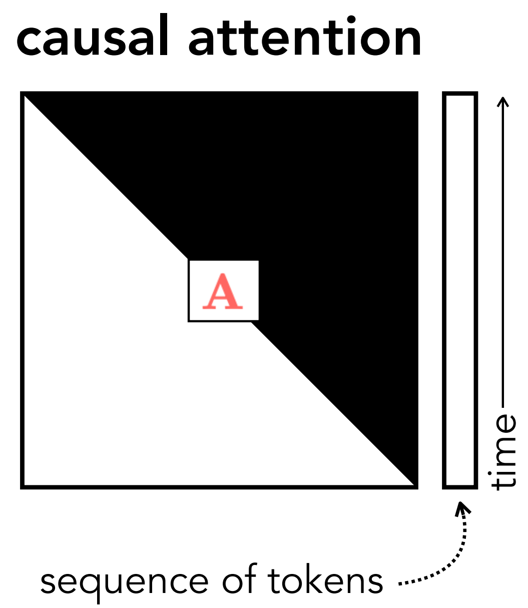
Training
$\mathbf{x}_1, \ldots, \mathbf{x}_{n-1}$
$\mathbf{x}_n$
Once upon a , time
There and back , again
The slow brown , fox
To be or not to , be
Learner
Predictor
Sampling
$\mathbf{x}_1, \ldots, \mathbf{x}_{n-1}$
Colorless green ideas sleep
Predictor
$\hat{\mathbf{x}}_n$
furiously

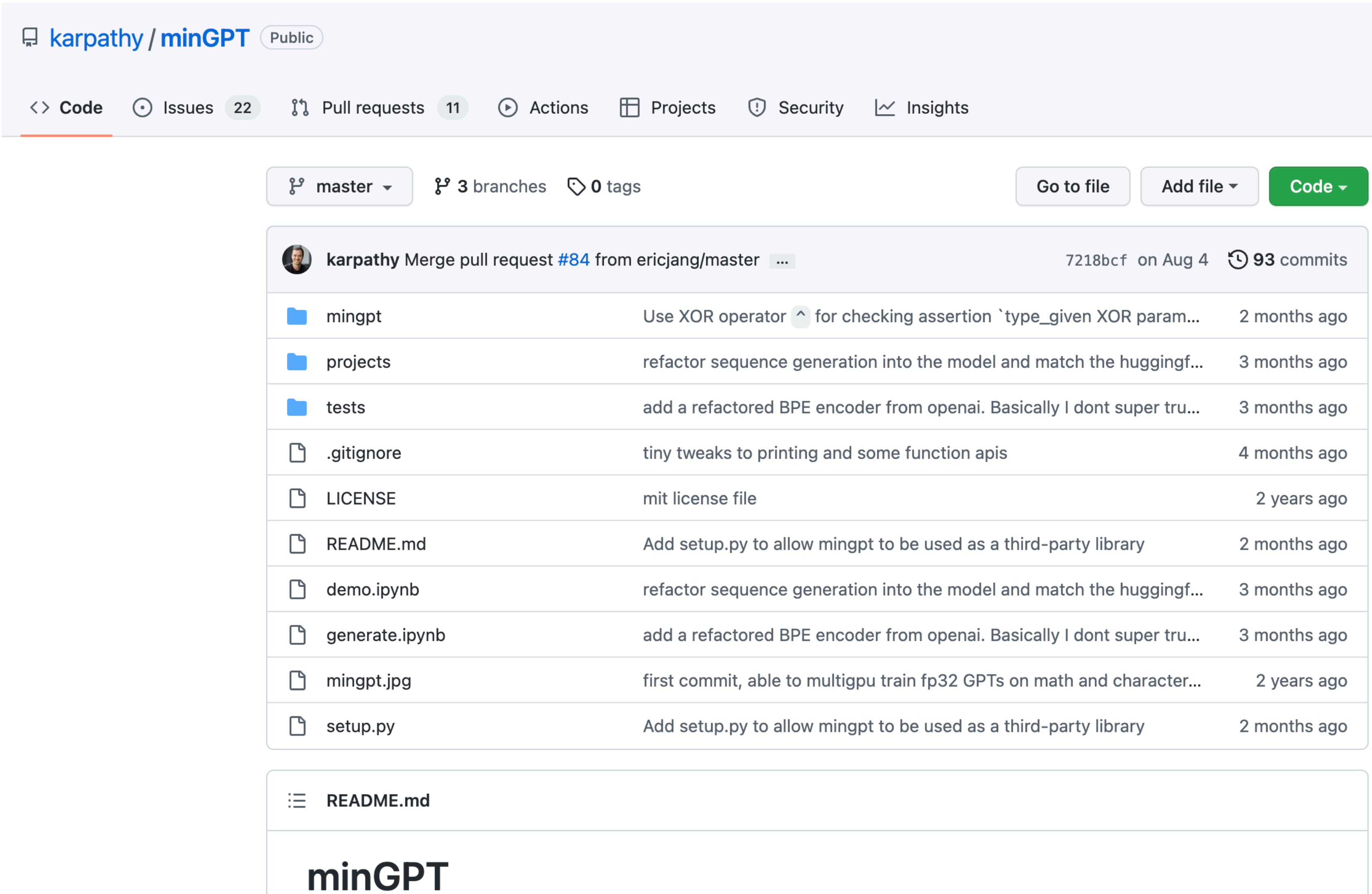
GPT (and many other related models)



GPT training (and many other related models)






karpathy / minGPT Public
Code
Issues 22
Pull requests 11
Actions
Projects
Security
Insights
master
3 branches
0 tags
Go to file
Add file
Code
karpathy Merge pull request #84 from ericjang/master ...
7218bcf on Aug 4
93 commits
mingpt
Use XOR operator ^ for checking assertion `type_given XOR param...
2 months ago
projects
refactor sequence generation into the model and match the huggingf...
3 months ago
tests
add a refactored BPE encoder from openai. Basically I dont super tru...
3 months ago
.gitignore
tiny tweaks to printing and some function apis
4 months ago
LICENSE
mit license file
2 years ago
README.md
Add setup.py to allow mingpt to be used as a third-party library
2 months ago
demo.ipynb
refactor sequence generation into the model and match the huggingf...
3 months ago
generate.ipynb
add a refactored BPE encoder from openai. Basically I dont super tru...
3 months ago
mingpt.jpg
first commit, able to multigpu train fp32 GPTs on math and character...
2 years ago
setup.py
Add setup.py to allow mingpt to be used as a third-party library
2 months ago
README.md
minGPT

Transformers

- Three key ideas
  - Tokens
  - Attention
  - Positional encoding
- Examples of architectures and applications