Transformers, Dynamical Systems and Optimal Transport.

Pierre Ablin
Joint work with Valérie Castin, Michael Sander and Gabriel Peyré

Neural networks

Neural networks

- Neural networks are parameterized functions from an input space (text, images, sounds) to an output space (vector space, text, …)

$$f_\theta(x)$$

Parameters

Inputs

- Parameters θ live in a vector space. The way the parameters define the transform f is called the network's **architecture**.

Neural networks as a chain of simple transforms

- To build a complicated function f_θ, we chain simpler transforms

$$
\begin{aligned}
x^0 &= x \\
x^{l+1} &= f^l_{\theta^l}(x^l) \\
f_\theta(x) &= x^L, \theta = (\theta^0, \ldots, \theta^{L-1})
\end{aligned}
$$

where each $f^l_{\theta^l}$ is a simple function

The simplest transform: Multi-Layer-Perceptron (MLP)

- A MLP is a map from $\mathbb{R}^d$ to $\mathbb{R}^p$ parameterized by $\theta = (W_1, W_2, b_1, b_2)$, where W_i are matrices and b_i are vectors. The hidden dimension is $h = \dim(b_1)$.

$$f_\theta(x) = W_2 \sigma(W_1 x + b_1) + b_2$$

- The element-wise function σ is a Rectified Linear Unit (ReLU): $\sigma(u) = \max(u, 0)$
- These functions are universal approximators [Cybenko 89]: any continuous function on a compact can be approximated by f_θ:

$$\forall f, \varepsilon > 0, \exists h, \theta \text{ such that } \|f - f_\theta\|_\infty \leq \varepsilon$$

Going deep with residual connections

- Iterating only MLPs leads to instability: training becomes harder and harder with depth.
- Residual connections is a simple way to facilitate training
- Intuition: it is easy to learn to *do nothing*: simply take $\theta^l = 0$, the layer has no effect.

$$x^{l+1} = f^l_{\theta^l}(x^l)$$

$$x^{l+1} = x^l + f^l_{\theta^l}(x^l)$$

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770-778. 2016.

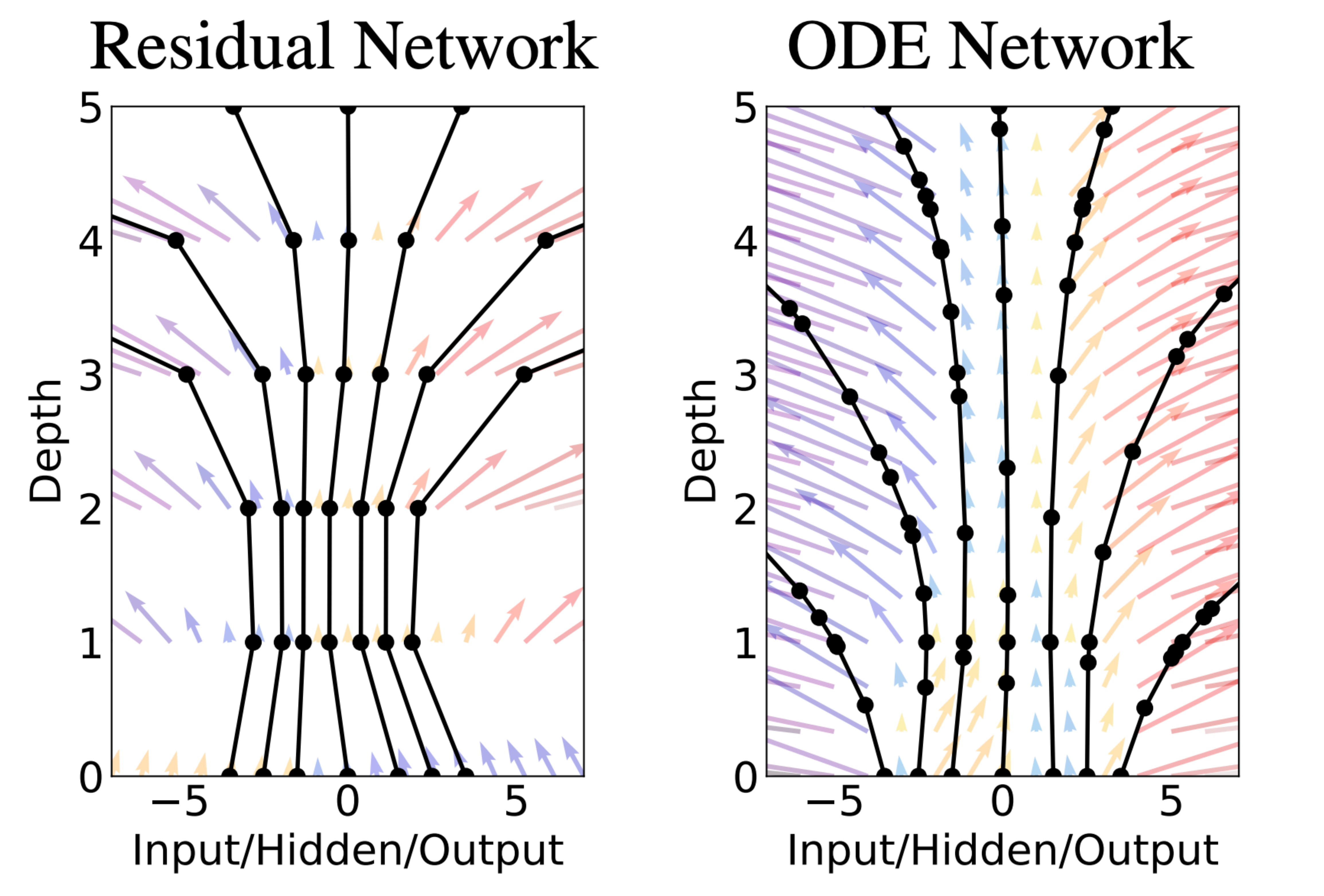
Link with dynamical systems

$$x^{l+1} = x^l + f^l_{\theta^l}(x^l)$$

- If the functions f^l are all the same, this is an Euler discretization (with step 1) of the ordinary differential equation:

$$\frac{dx}{dt} = f_{\theta(t)}(x(t))$$

- Makes a parallel between deep residual networks and dynamical systems
- Precise link between the two studied in [2, 3]

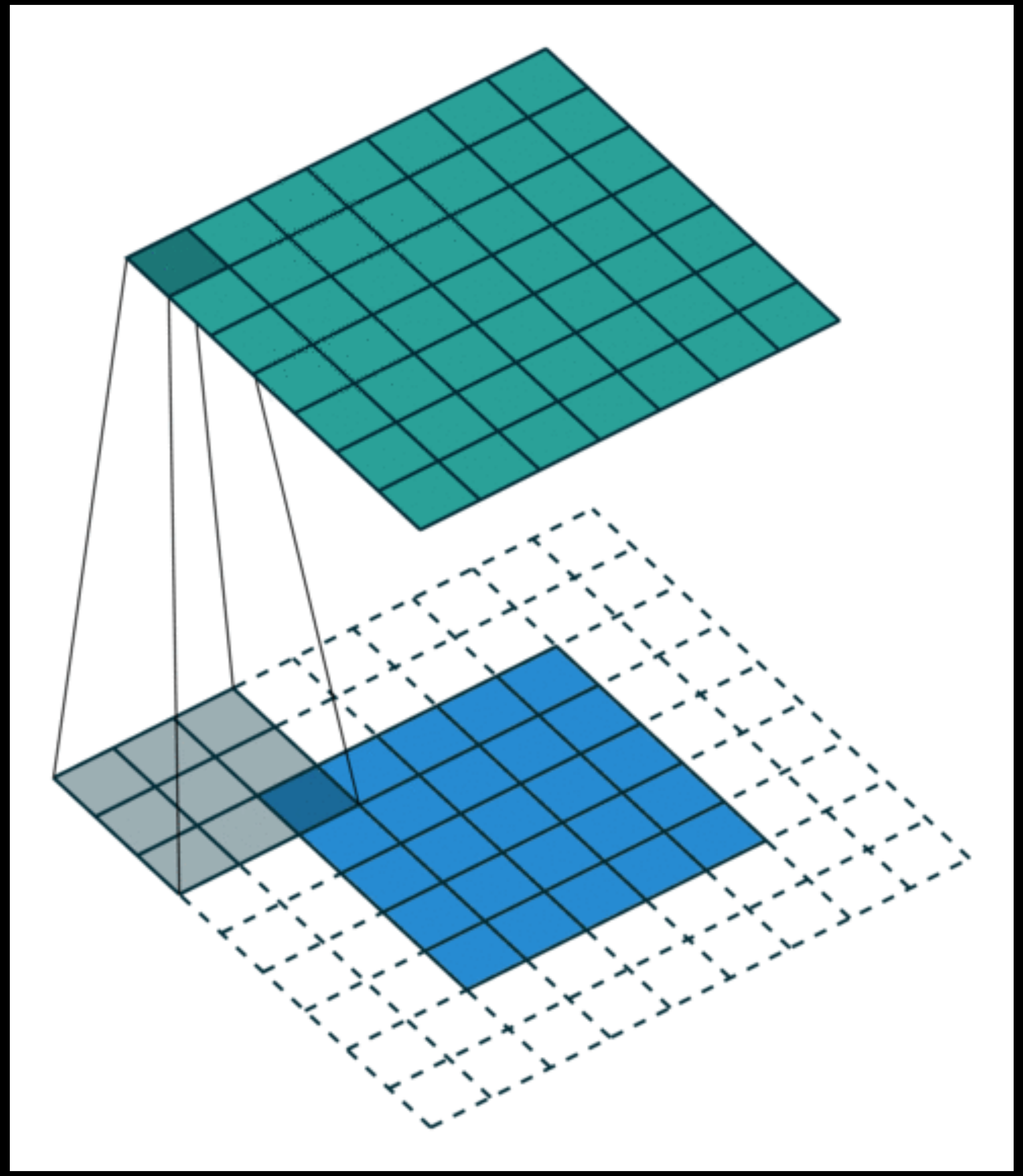


[1] Chen, Ricky TQ, Yulia Rubanova, Jesse Bettencourt, and David K. Duvenaud. "Neural ordinary differential equations." Advances in neural information processing systems 31 (2018).

[2] Barboni, Raphaël, Gabriel Peyré, and François-Xavier Vialard. "On global convergence of ResNets: From finite to infinite width using linear parameterization." Advances in Neural Information Processing Systems 35 (2022)

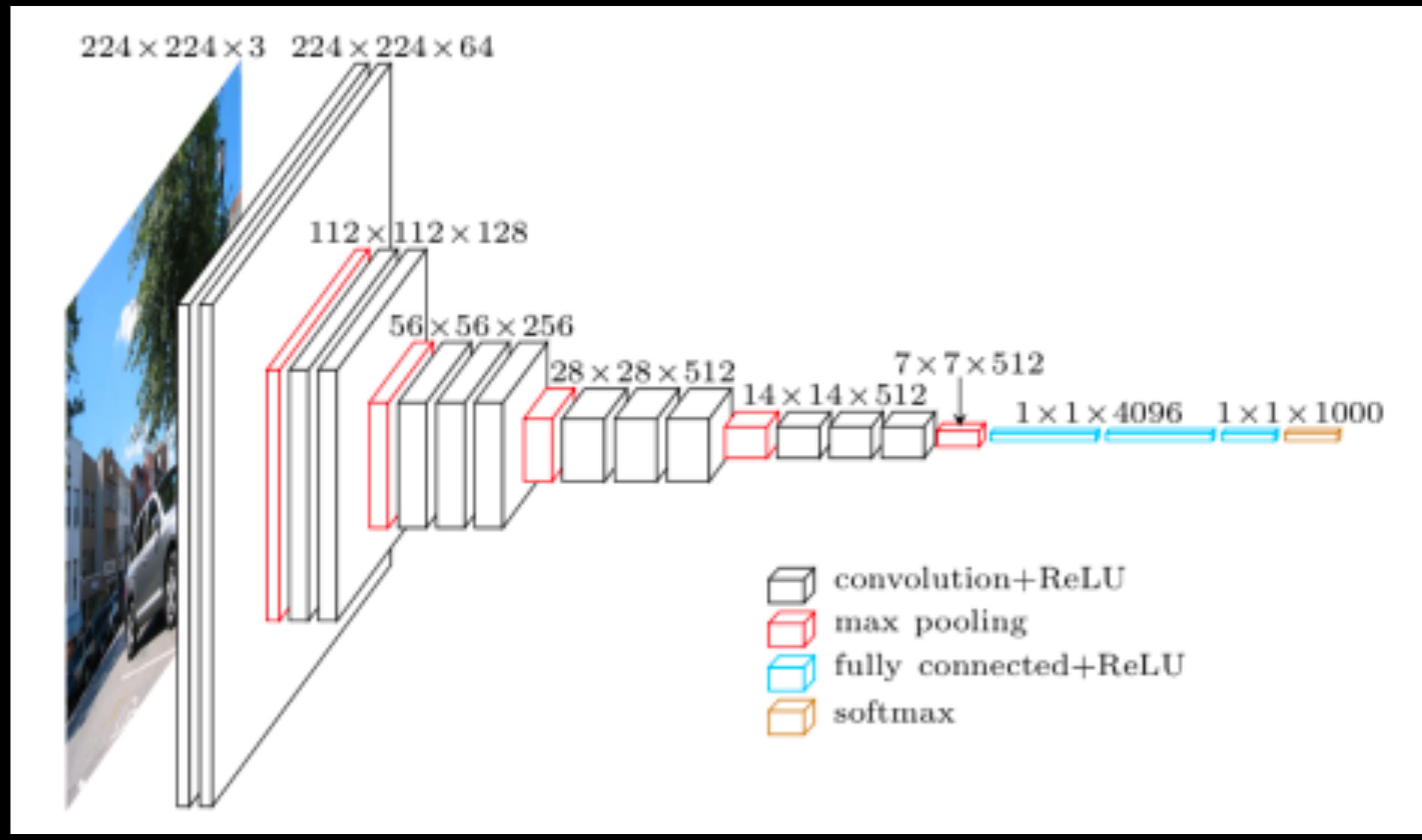
[3] Sander, Michael, Pierre Ablin, and Gabriel Peyré. "Do Residual Neural Networks discretize Neural Ordinary Differential Equations?." Advances in Neural Information Processing Systems 35 (2022)

A bestiary of transforms for each application



- In vision and audio signal processing: convolutions

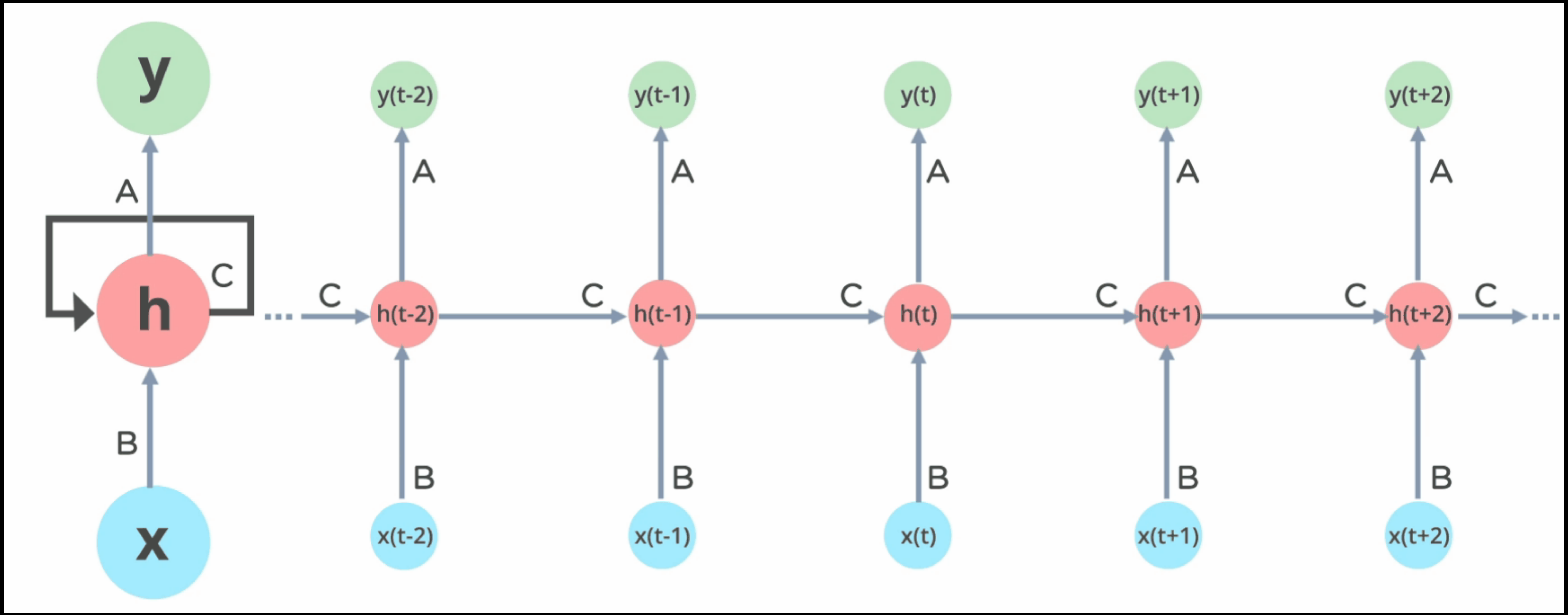
- Stacking them gives deep convolutional networks (CNNs)



Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems 25 (2012).

A bestiary of transforms for each application

- In text processing: recurrent neural networks to encode the recurrent nature of language.



Sepp Hochreiter, Jürgen Schmidhuber, "Long short-term memory", 1997

Transformers: an all-purpose architecture?
How does chat-GPT works?

Transformers are used everywhere

- Most widely used architectures for computer vision: Transformers
- Most widely used architectures for text processing: Transformers
- Most widely used architecture for audio processing: Transformers
- Chat-GPT built upon GPT: Generative Pretraining Transformers

How can we have the same architecture for all these different modalities?

Transformers are sequence-to-sequence mappings

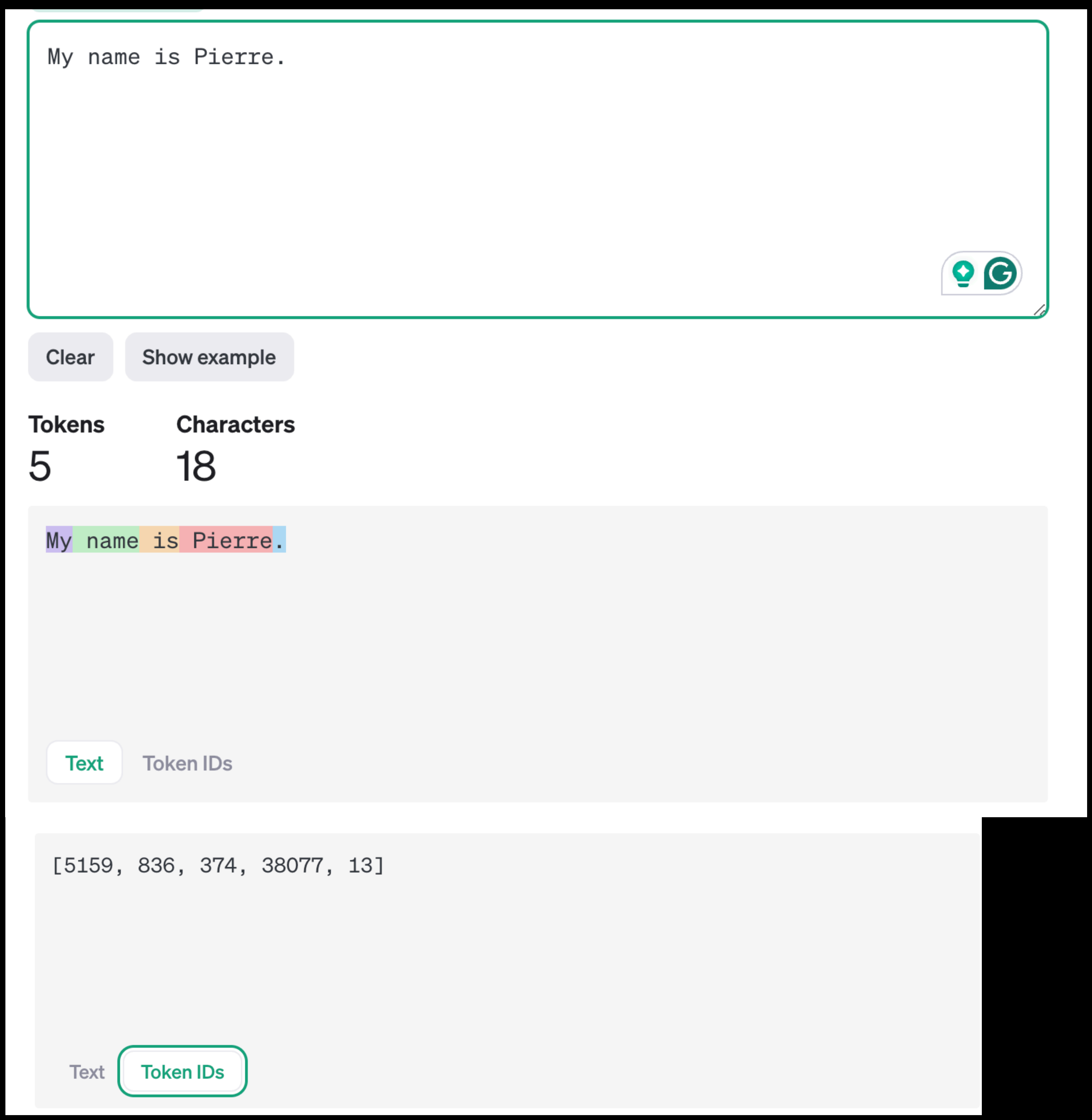
- The input to a transformer is not a single vector x but a sequence of vectors $X = (x_1, \ldots, x_n)$ where each x_i is in $\mathbb{R}^d$
- It outputs a sequence of **same length** $Y = (y_1, \ldots, y_n)$ where each y_i is in $\mathbb{R}^d$
- It processes sequences of arbitrary length: n can change from input to input.

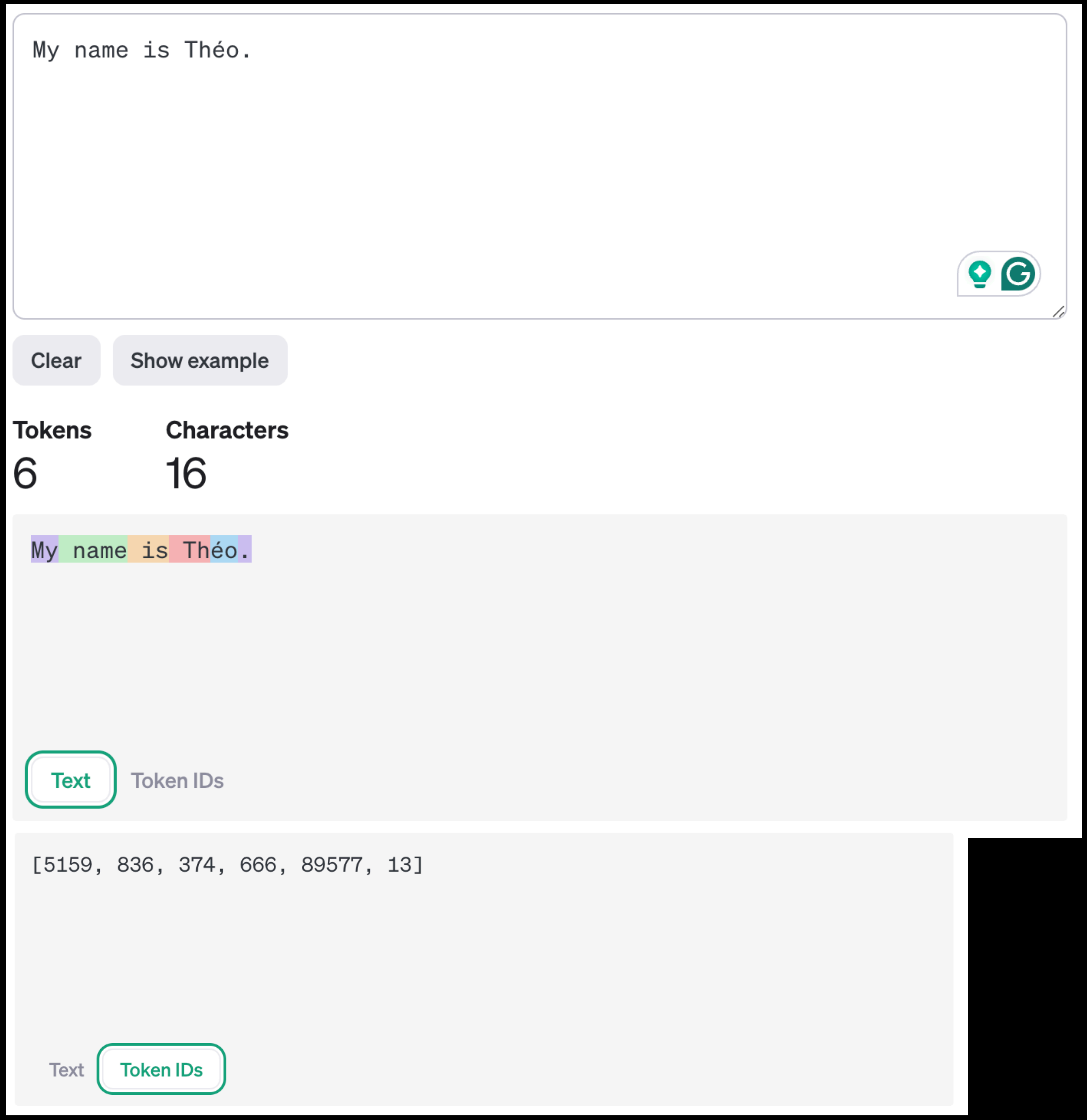
Transformers from scratch

- How to turn an input into a sequence of vectors?
- This process is called **tokenization**. It depends on the input space.

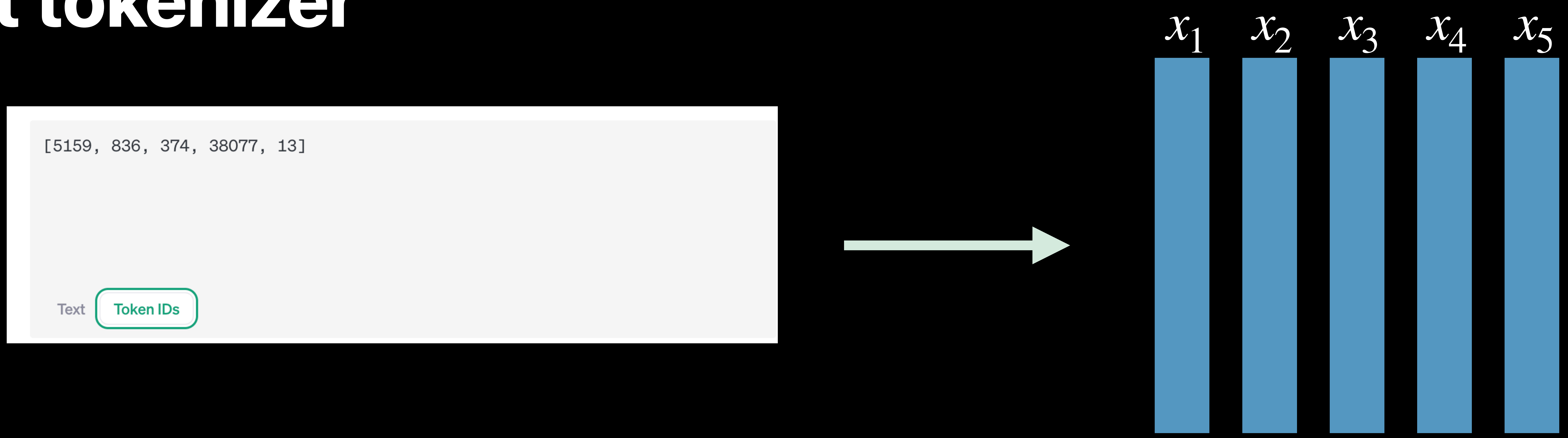
Text tokenizer

- From https://platform.openai.com/tokenizer



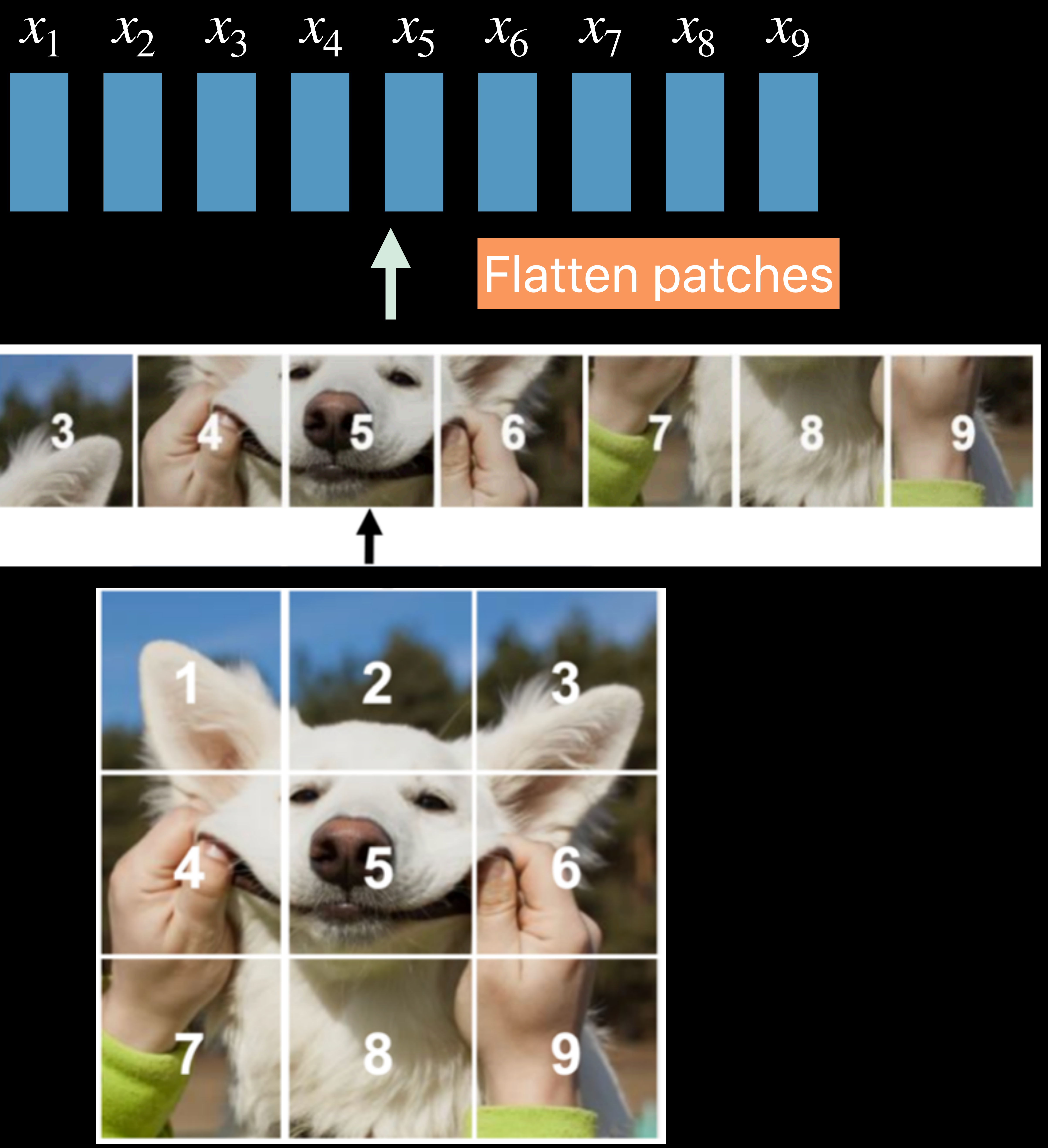


Text tokenizer



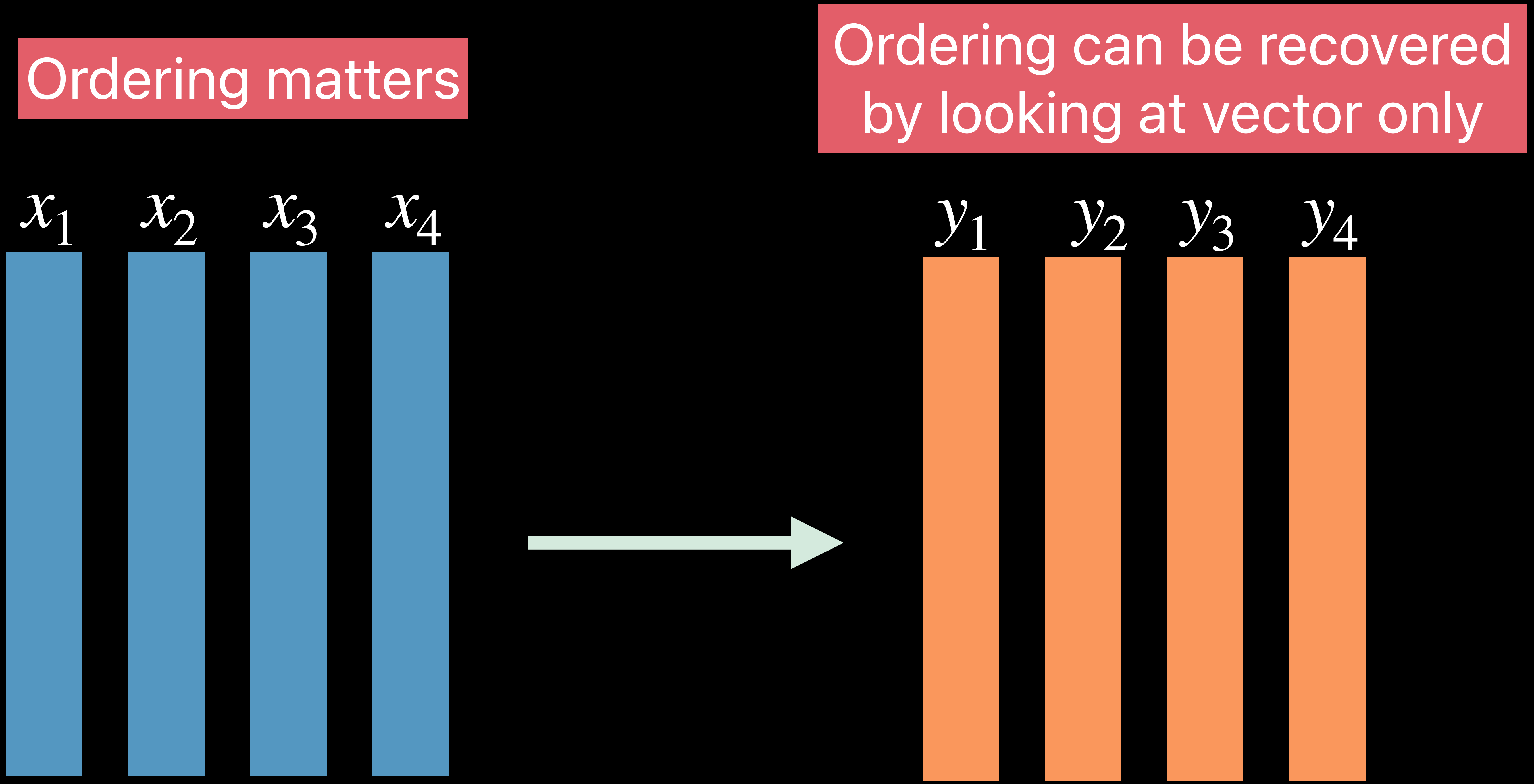
- Each token ID is then mapped to a high dimensional vector. The mapping is learned (it is part of the parameters of the transformer θ).
- There is one learned vector for each token id in the vocabulary.

Image tokenizer



Positional encoding

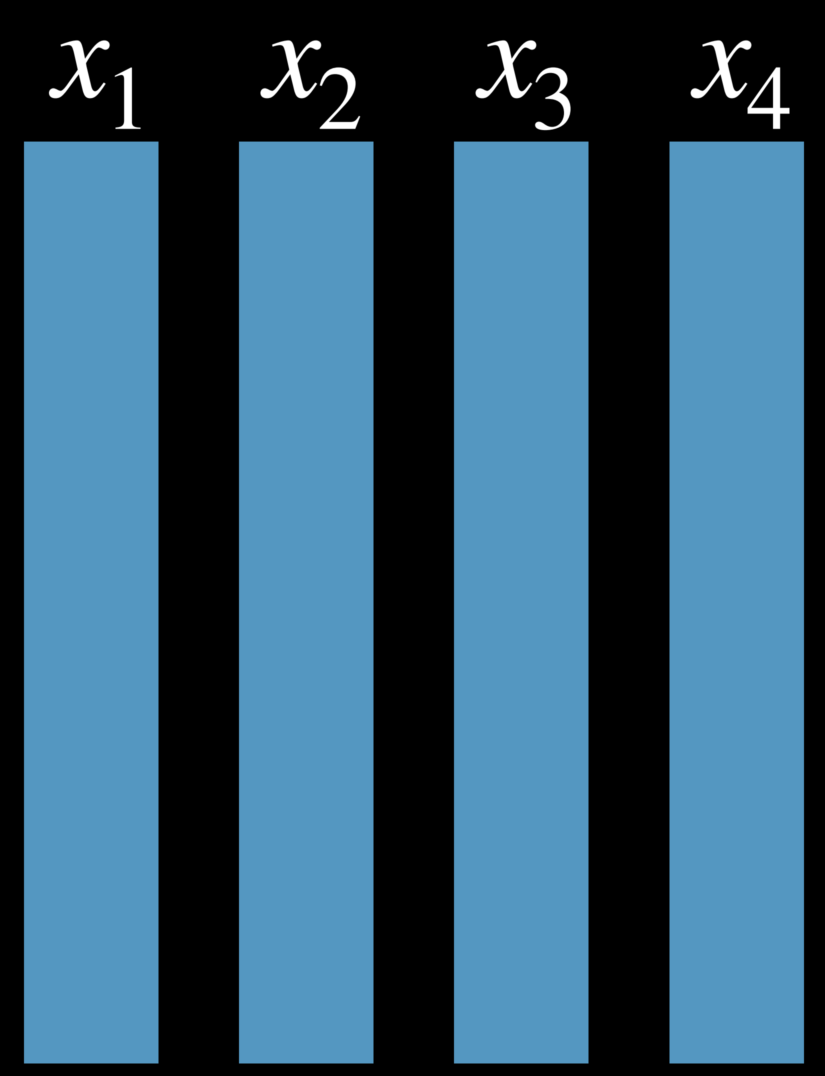
- At this point in the process, ordering of the vectors is crucial.
- **Positional encoding** encodes the position of the vectors into the vectors themselves.



- Simple solution: append a coordinate: $y_i = [x_i, i]$
- In practice, more complicated methods are used

So far...

- We have transformed an input (image or text) into a sequence of vectors for which the ordering does not matter.
- The transformer is then composed of two repeated simple operations.

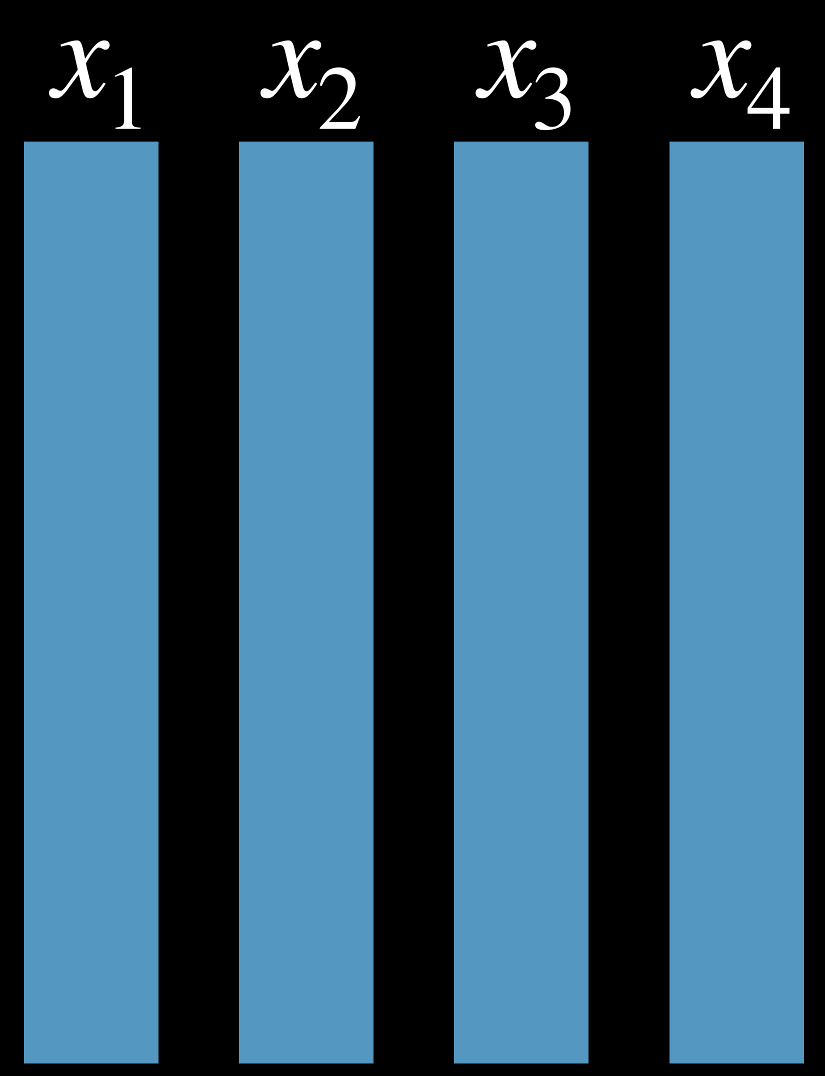


The two core building blocks: MLPs and Attention

Individual MLPs

- Each vector x_i is in $\mathbb{R}^d$. We use an MLP that acts on each vector individually:

$$f_\theta((x_1, \ldots, x_n)) = (MLP_\theta(x_1), \ldots, MLP_\theta(x_n))$$



Vector interactions with attention

- The most important function in a Transformer is Attention.

$$(y_1, \ldots, y_n) = \mathrm{Attn}((x_1, \ldots, x_n))$$

- It makes vectors interact with each other: y_i depends on all the other x_j.

- Parameterized by three matrices $\theta = (W_Q, W_K, W_V) \in \mathbb{R}^{d \times d}$

Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).

Attention

- Parameterized by three matrices $\theta = (W_Q, W_K, W_V) \in \mathbb{R}^{d \times d}$

- Compute the queries, keys and values

$$q_i = W_Q x_i, k_i = W_K x_i, \text{ and } v_i = W_V x_i$$

- The i-th output vector y_i is a convex combination of the values:

$$y_i = \sum_{j=1}^{n} w_{ij} v_j, \text{ with } w_{ij} > 0 \text{ and } \sum_j w_{ij} = 1$$

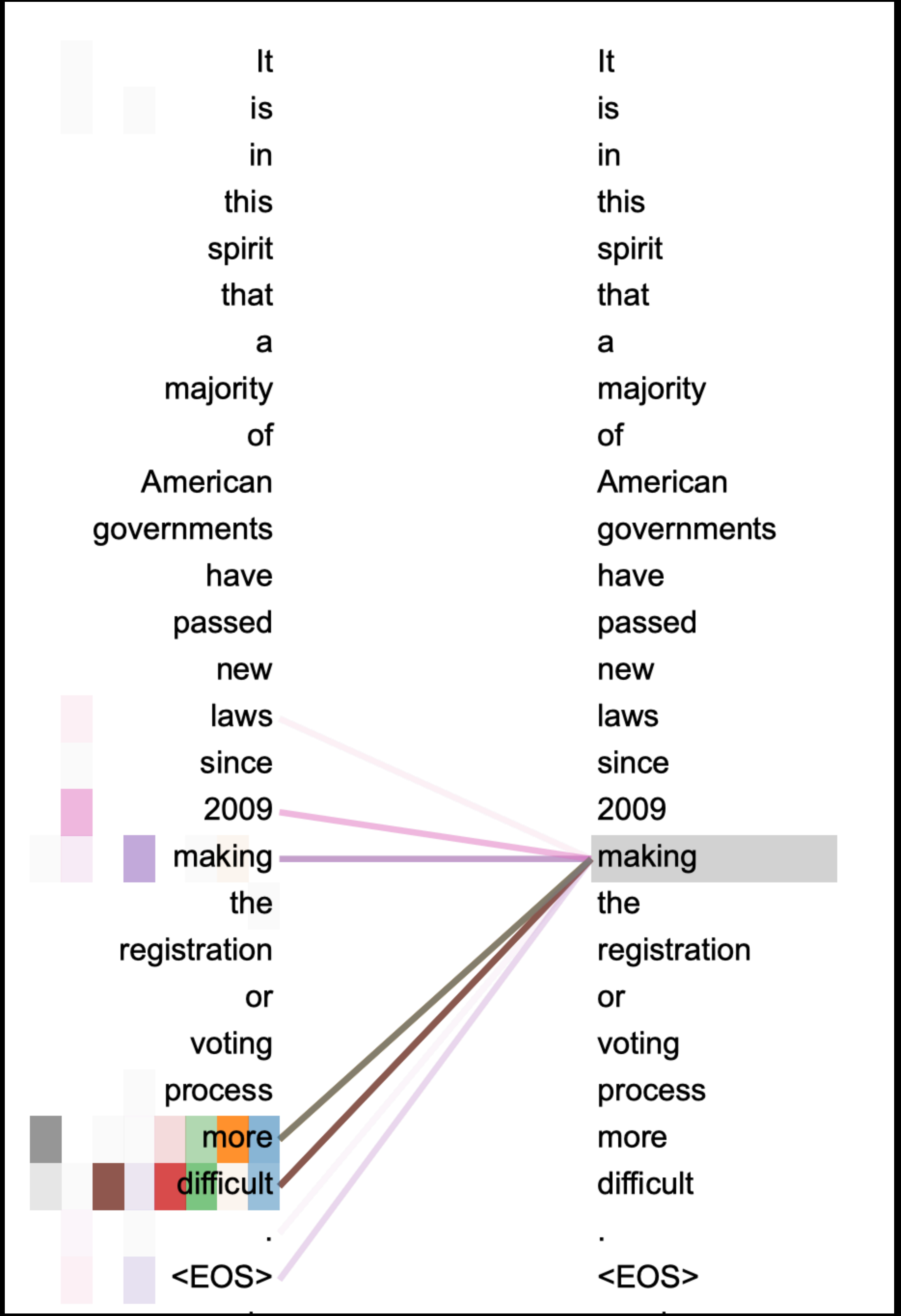
- Weights depend on the alignement between the query q_i and all the keys k_j:

$$w_{ij} = \mathrm{softmax}((\langle q_i, k_j \rangle))_j := \frac{\exp(\langle q_i, k_j \rangle)}{\sum_{l=1}^{n} \exp(\langle q_i, k_l \rangle)}$$

Attention: intuition

- The coefficient w_{ij} is large when q_i and k_j are well aligned.

- Allows to focus on important links between tokens

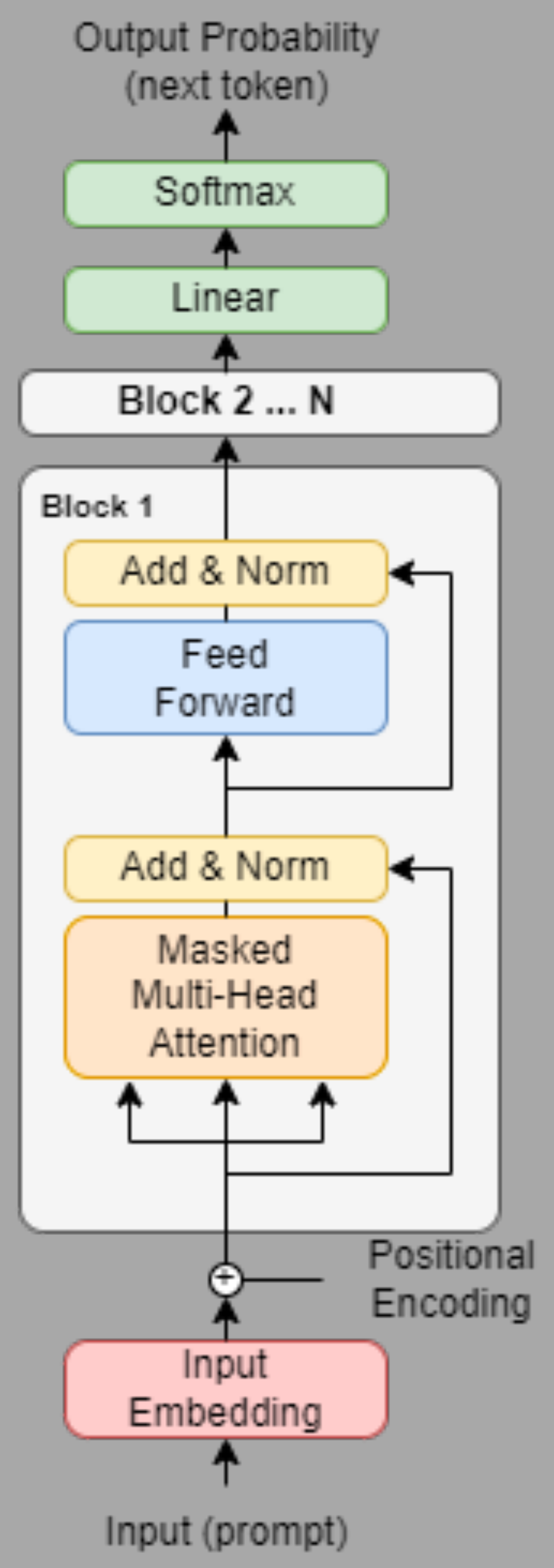


Full transformer architecture

- Stack Attention and MLPs, using residual connections

$$Z = X + \mathrm{Attn}(X)$$

$$Y = Z + \mathrm{MLP}(Z)$$



The subtle bits: normalization layers

- There are also normalization layers. Simplest one is called RMSNorm.
- Acts on vectors individually.
- Trainable parameters $\beta \in \mathbb{R}^d$

$$\mathrm{Norm}((x_1, \ldots, x_n)) = (y_1, \ldots, y_n)$$

$$y_i = \beta \odot \frac{x_i}{\|x_i\|}$$

- Projects the vectors on an ellipsis.

$$Z = X + \mathrm{Attn}(\mathrm{Norm}(X))$$

$$Y = Z + \mathrm{MLP}(\mathrm{Norm}(Z))$$

The subtle bits: multi-head attention

- Attention is not flexible enough; can only focus on one specific interaction between vectors.
- Multi-head attention: use multiple attention layers in parallel, and then aggregate them

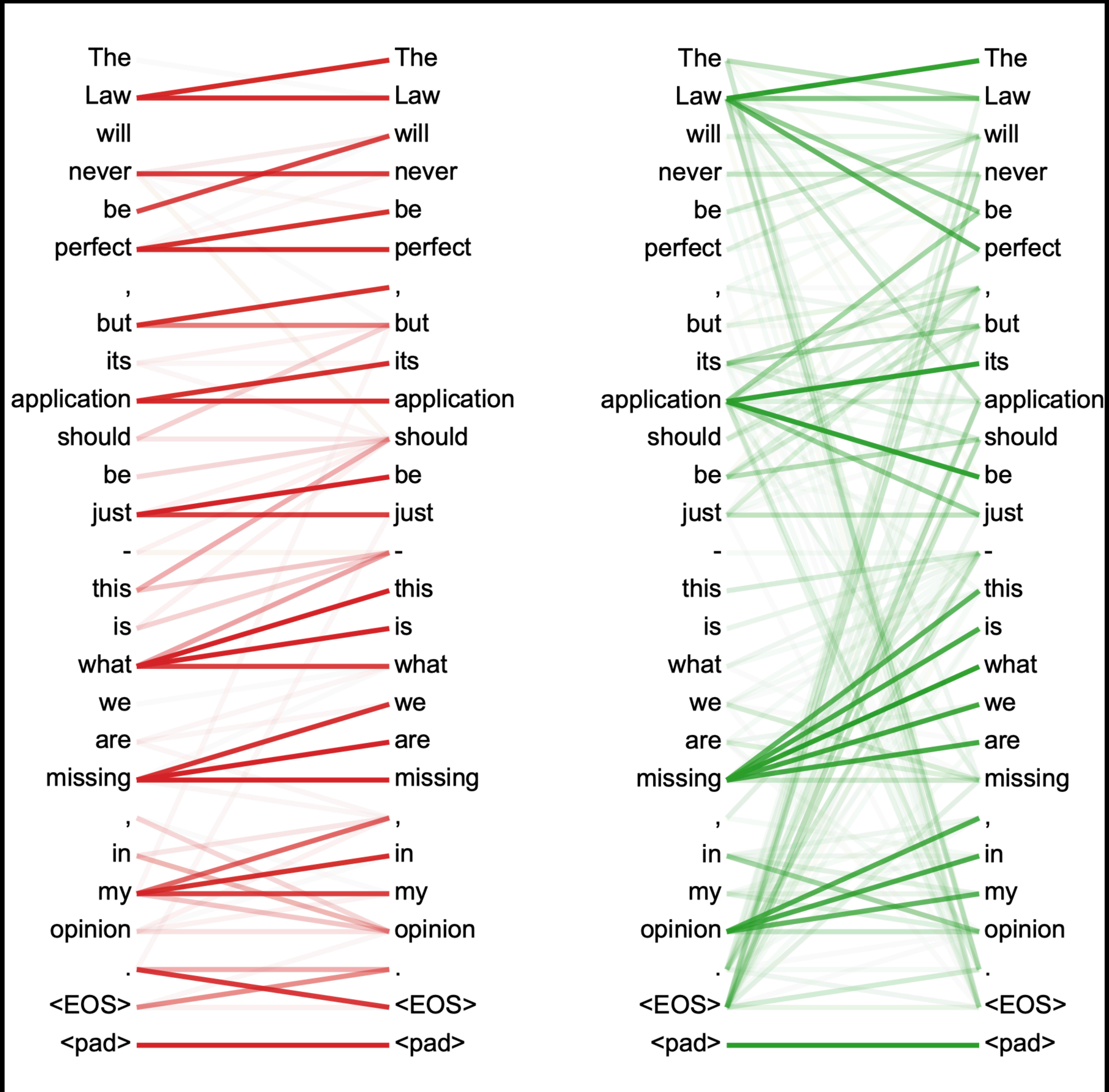
$$\mathrm{MultiAttn}(X) = \sum_{l=1}^{h} \mathrm{Attn}^{l}(X)$$

$$Z = X + \mathrm{MultiAttn}(\mathrm{Norm}(X))$$

$$Y = Z + \mathrm{MLP}(\mathrm{Norm}(Z))$$

The subtle bits: multi-head attention

Head 1



Head 2

Attention: a measure-to-measure map

Attention

- Parameterized by three matrices $\theta = (W_Q, W_K, W_V) \in \mathbb{R}^{d \times d}$
- Compute the queries, keys and values

$$q_i = W_Q x_i, k_i = W_K x_i, \text{ and } v_i = W_V x_i$$

- The i-th output vector y_i is a convex combination of the values:

$$y_i = \sum_{j=1}^{n} w_{ij} v_j, \text{ with } w_{ij} > 0 \text{ and } \sum_j w_{ij} = 1$$

- Weights depend on the alignement between the query q_i and all the keys k_j:

$$w_{ij} = \text{softmax}((\langle q_i, k_j \rangle))_j := \frac{\exp(\langle q_i, k_j \rangle)}{\sum_{l=1}^{n} \exp(\langle q_i, k_l \rangle)}$$

Key insight: Attention is equivariant w.r.t. permutations

$$\mathrm{Attn}(x_1, \ldots, x_n) = (y_1, \ldots, y_n)$$

- For σ a permutation from $\{1,\ldots,n\}$ to $\{1,\ldots,n\}$:

$$\mathrm{Attn}(x_{\sigma(1)}, \ldots, x_{\sigma(n)}) = (y_{\sigma(1)}, \ldots, y_{\sigma(n)})$$

- Same for the MLP and Normalization
- The whole transformer architecture, apart from the initial positional encoding, *is permutation-equivariant!*

Extending attention to measures

$$\mathrm{Attn}(x_1, \ldots, x_n) = (y_1, \ldots, y_n)$$

$$y_i = \frac{\sum_{j=1}^n \exp(x_i^T W_Q^T W_K^T x_j) W_V x_j}{\sum_{j=1}^n \exp(x_i^T W_Q^T W_K^T x_j)}$$

- Define the measure map for $\mu \in \mathscr{P}(\mathbb{R}^d)$: $\Gamma_\mu(x) = \dfrac{\int \exp(x^T W_Q^T W_K^T z) W_V z d\mu(z)}{\int \exp(x^T W_Q^T W_K^T z) d\mu(z)}$

We have $y_i = \Gamma_\mu(x_i)$ with $\mu = \dfrac{1}{n}\sum_{i=1}^{n} \delta_{x_i}$

Extension:

$$\mathrm{Attn}(\mu) = (\Gamma_\mu)_\# \mu$$

Sander, Michael E., Pierre Ablin, Mathieu Blondel, and Gabriel Peyré. "Sinkformers: Transformers with doubly stochastic attention." In International Conference on Artificial Intelligence and Statistics, pp. 3515-3530. PMLR, 2022.

Attention + Neural ODE = Continuity equation

Residual attention

$$(y_1, \ldots, y_n) = (x_1, \ldots, x_n) + \mathrm{Attn}(x_1, \ldots, x_n)$$

$$\Gamma_\mu(x) = \frac{\int \exp(x^T W_Q^T W_K^T z) W_V z \, d\mu(z)}{\int \exp(x^T W_Q^T W_K^T z) \, d\mu(z)}$$

- Euler discretization of the ODE $\dot{x}_i = \Gamma_\mu(x_i)$
- Equivalent to the continuity equation: $\partial_t \mu + \mathrm{div}(\mu \Gamma_\mu) = 0$
- Can then be used to study theoretical properties of transformers
- Normalization in the attention makes existence of solution non trivial / interesting

Sander, Michael E., Pierre Ablin, Mathieu Blondel, and Gabriel Peyré. "Sinkformers: Transformers with doubly stochastic attention." In International Conference on Artificial Intelligence and Statistics, pp. 3515-3530. PMLR, 2022.

Geshkovski, Borjan, Cyril Letrouit, Yury Polyanskiy, and Philippe Rigollet. "The emergence of clusters in self-attention dynamics." Advances in Neural Information Processing Systems 36 (2024).

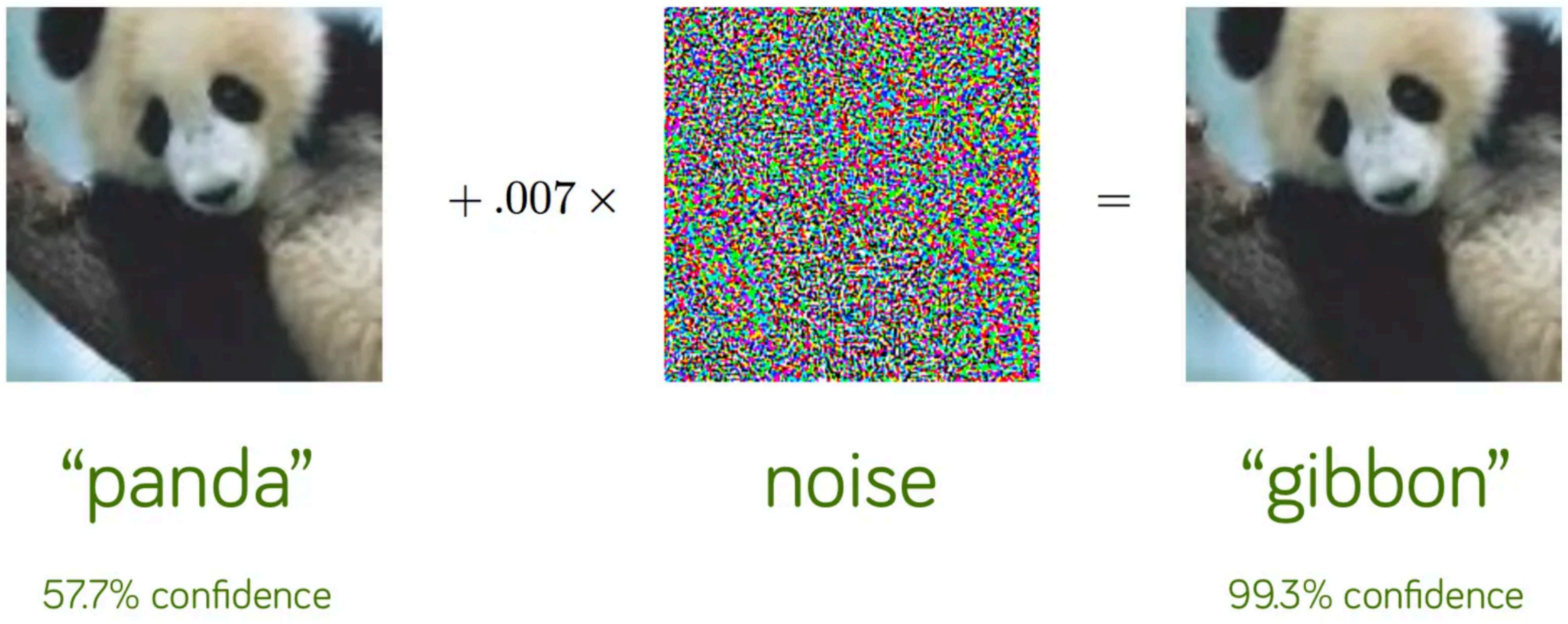
Lipschitz constant of attention

Why do we care about Lipschitz constants?

Robustness to adversarial attacks

- An adversarial attack is a small perturbation to the input of a network that leads to large change in the output:

$$\delta \text{ such that } \|f_\theta(x+\delta) - f(x)\| \gg \delta, \text{ with } \|\delta\| \ll 1$$



Robustness to adversarial attacks (contd.)

- If a network is Lipschitz, we know that by definition

$$\|f_\theta(x+\delta) - f(x)\| \leq L_f \|\delta\|$$

Hence, the network is hard to attack.

Lipschitzness certifies robustness

- In most cases, we only care about perturbation of x in the "data manifold": care only about local Lipschitz constant on the manifold.
- Hard to compute, hard to control.

Cisse, Moustapha, et al. "Parseval networks: Improving robustness to adversarial examples." *International conference on machine learning*. PMLR, 2017.

Building invertible neural networks

- **Theorem:** If $f : \mathbb{R}^p \to \mathbb{R}^p$ is L-Lipschitz with $L < 1$ then $x \mapsto x + f(x)$ is invertible, i.e. for any y the equation $y = x + f(x)$ has one and only one solution.
- To invert the map, simply iterate $x_{n+1} = y - f(x_n)$
- Hence, if we having <1 Lipschitz building blocks, we can design residual networks that are invertible by design.
- **Invertible networks usecases:**

1. Generative modeling (normalizing flows)
2. Memory-efficient backprop (no need to store activations)

Behrmann, Jens, et al. "Invertible residual networks." International conference on machine learning. PMLR, 2019.

Global Lipschitz constant of attention ?

- We need to find an input sequence $(x_1, \ldots, x_n)$ and small displacements $(d_1, \ldots, d_n)$ such that $\mathrm{Att}(x_1 + d_1, \ldots, x_n + d_n)$ is as far from $\mathrm{Att}(x_1, \ldots x_n)$ as possible.
- Simplify things: assume $W_Q = W_K = W_V = I$, and dim = 1

In this simple case: $\mathrm{Attn}(x_1, \ldots, x_n)_i = \dfrac{\sum_{j=1}^{n} \exp(x_i x_j) x_j}{\sum_{j=1}^{n} \exp(x_i x_j)}$

Problem: product interaction. The function $\phi(x, y) = \sigma(xy)$ is not Lipschitz for any non-constant function σ.

What if inputs are bounded?

Estimate $L(n, R) = \sup_{x_1, \ldots, x_n \in B(R)} \|\mathrm{Jac}(\mathrm{Attn})(x_1, \ldots, x_n))\|_2$ with $B(R)$ ball of radius R

Castin, Valérie, Pierre Ablin, and Gabriel Peyré. "Understanding the Regularity of Self-Attention with Optimal Transport." arXiv preprint arXiv:2312.14820 (2023).

Large n limit

Estimate $L(n, R) = \sup_{x_1,\ldots,x_n \in B(R)} \|\text{Jac}(\text{Attn})(x_1, \ldots, x_n))\|_2$

with $B(R)$ ball of radius R

- Catastrophic scaling in the limit $n \to \infty$: $L(n, R) \simeq_{n \to +\infty} R^2 \exp(R^2)$
- Intuition: take in 1d $\mu = \exp(-R^2)\delta_R + (1 - \exp(-R^2))\delta_{-R}$
- With a fixed number of vectors n, need $n \simeq \exp(R^2)$ to achieve this bound

Generic bound

Estimate $L(n, R) = \sup_{x_1, \ldots, x_n \in B(R)} \|\text{Jac}(\text{Attn})(x_1, \ldots, x_n))\|_2$

with $B(R)$ ball of radius R

- Generic bound : $L(n, R) \leq \sqrt{n} R^2$
- Tight when $n \simeq \exp(R^2)$

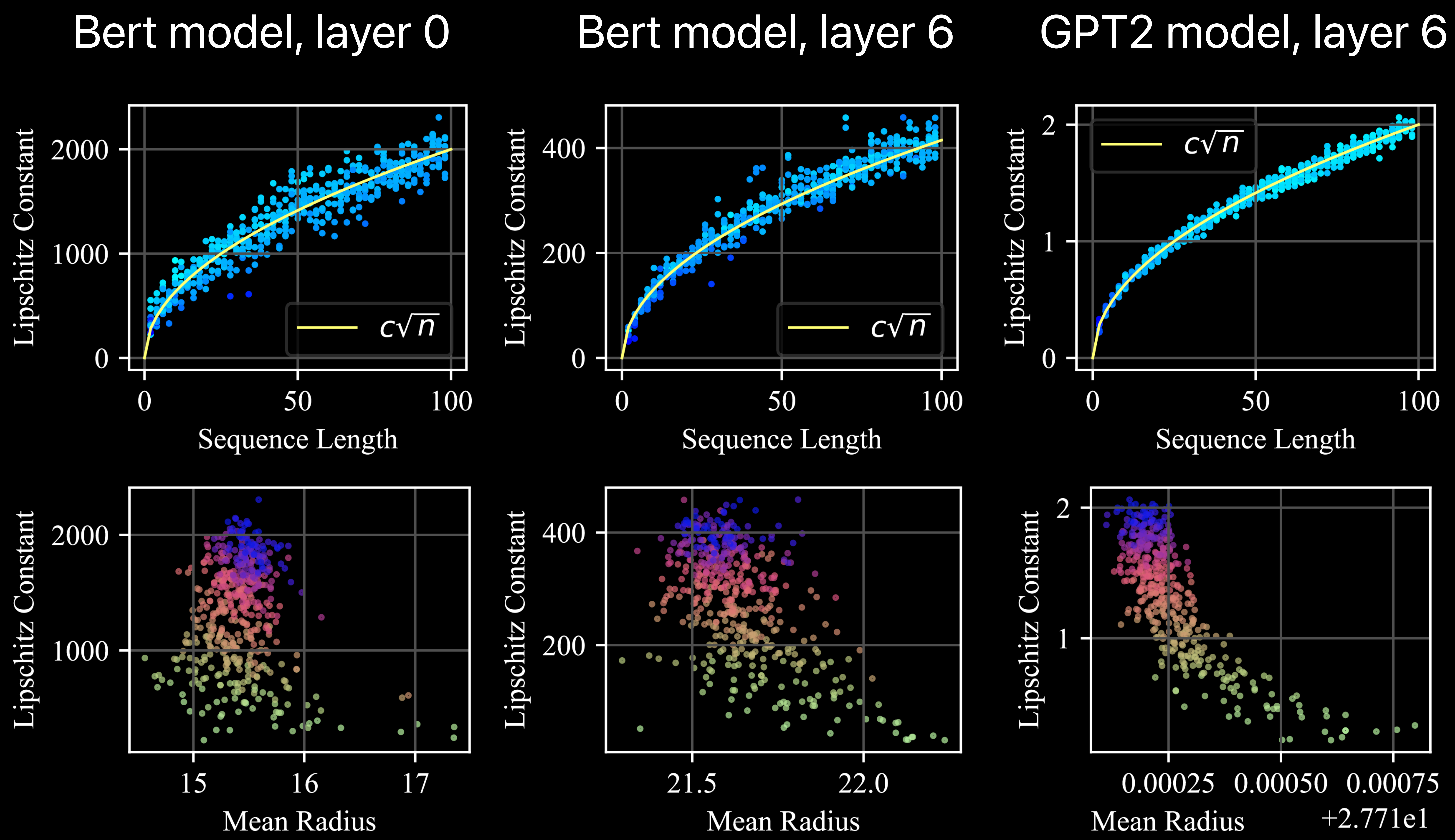
Large R limit

- Large radius regime: $\lim_{R \to +\infty} \|\mathrm{Jac}(\mathrm{Attn})(Rx_1, \ldots, Rx_n))\|_2 \leq \sqrt{n}$
- This is observed in practice!

Experiment

- Take sentences from Alice in Wonderland, and look at local Lipschitz constant when going through a trained transformer. Vary the sequence length.
- Local Lipschitz constant estimated with power method

Experiment



Bert model, layer 0
Bert model, layer 6
GPT2 model, layer 6
Lipschitz Constant
Sequence Length
$c\sqrt{n}$
0
1000
2000
50
100
200
400
1
2
Mean Radius
15
16
17
21.5
22.0
0.00025
0.00050
0.00075
+2.771e1

Conclusion

- Transformers are an all-purpose architecture used everywhere
- It takes as input sequences of vectors
- Apart from the initial positional encoding, it is permutation-equivariant, thus can be seen as acting on measures
- The corresponding continuity equation is interesting and non-standard
- The study of the regularity of the transformer leads to different surprising regimes

Thanks !