Transformers

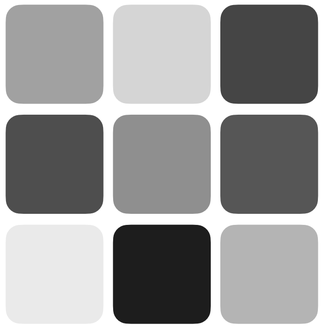
Active Training Course “Advanced Deep Learning”

Jonas Spinner
Universität Heidelberg

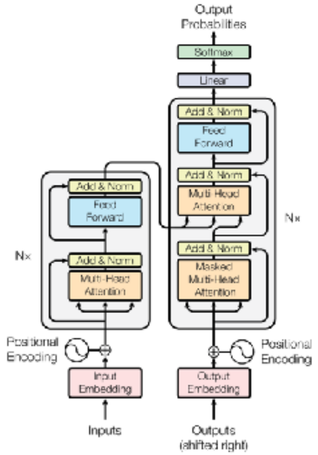
Why transformers?

POINT CLOUD DATA

FLEXIBLE & SCALABLE ARCHITECTURE



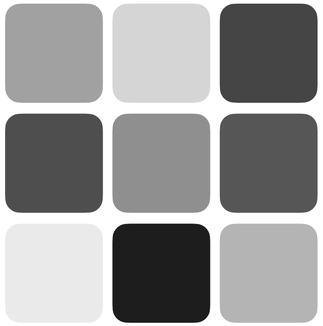
TRANSFORMERS



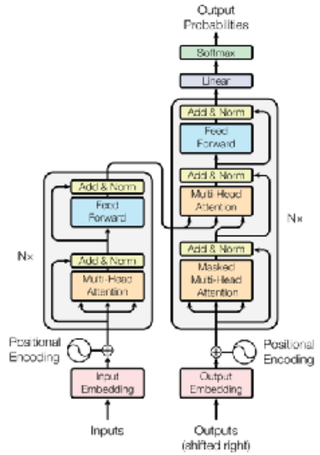
Roadmap

POINT CLOUDS

ATTENTION



TRANSFORMERS



EXAMPLES

Point clouds

Know your data

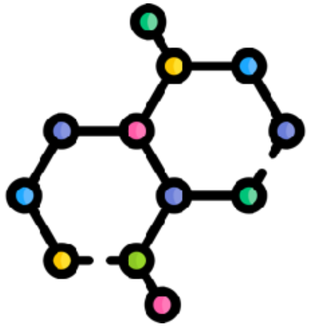
	UNSTRUCTURED	GRID	POINT CLOUD
Structure			
Network Architecture	MLP	Convolutional NN (CNN)	Graph NN (GNN)/ Transformer
Inductive Bias	-	Locality	Permutation symmetry Variable-size inputs Pairwise relations

Point clouds

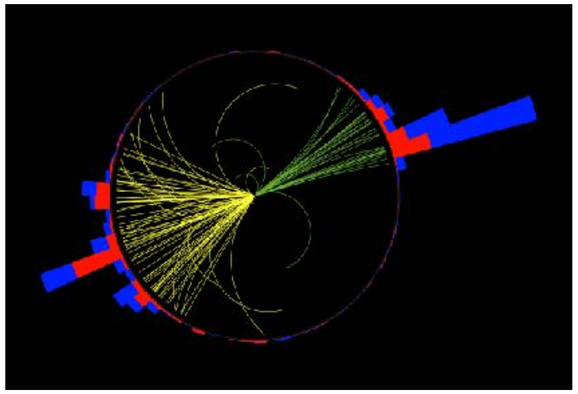
Point clouds are everywhere

Planets

Molecules



Particle Jets



People

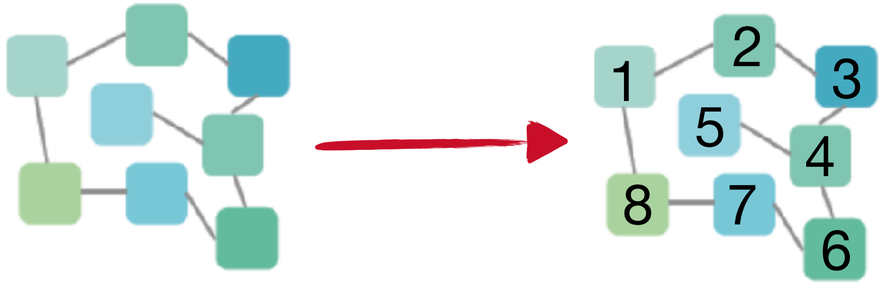
POINT CLOUD

Graph NN (GNN)/
Transformer

Permutation symmetry
Variable-size inputs
Pairwise relations

Point clouds

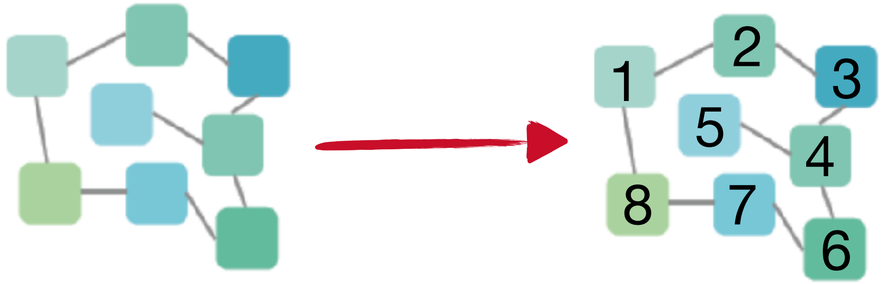
Anything can be a point cloud



- **Permutation symmetry breaking**
  = Add **type information** onto each token
  - Also called ‘positional encoding’

Point clouds

Anything can be a point cloud



- **Permutation symmetry breaking**
  = Add **type information** onto each token
  - Also called ‘positional encoding’
- **Why** would you do that?
  - Implicit bias ‘similar objects’
  - Regularisation from weight sharing in Transformers

Fixed-dimensional data
Type information: particle type

$$q\bar{q} \to e^{+}e^{-}\gamma$$

Words in a text
Type information:
Position in text

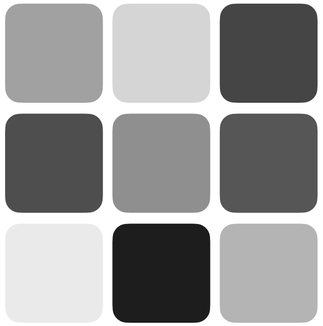
Attention is all you need .

Patches in an image
Type information:
Position in image

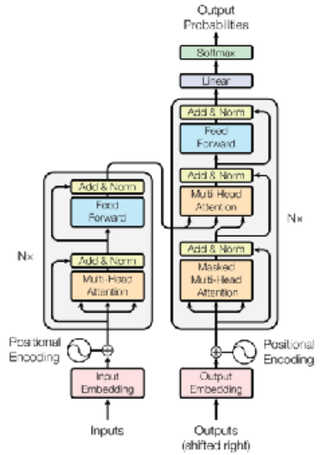
Roadmap

POINT CLOUDS

ATTENTION



TRANSFORMERS

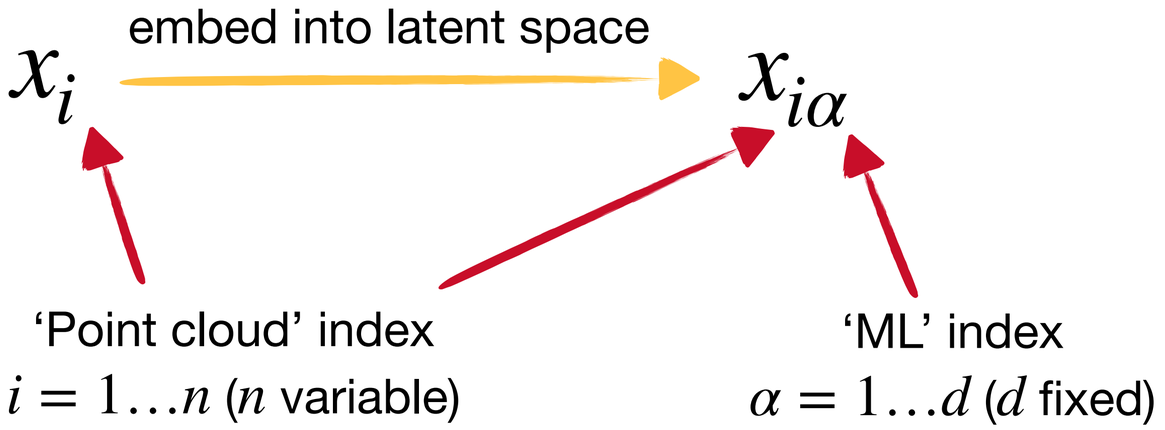


EXAMPLES

Attention

Wishlist

- Point clouds:

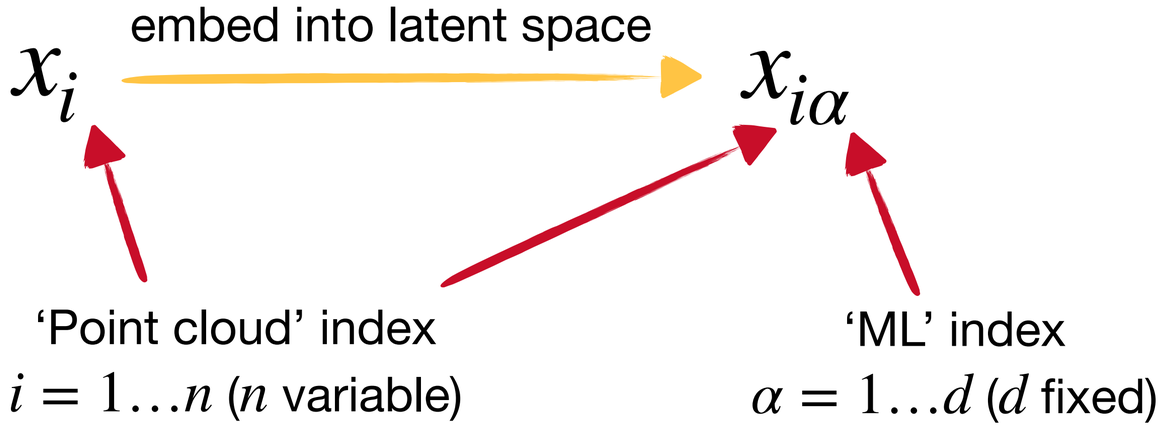


Data has shape (batchsize, #tokens, #latent dims)

Attention

Wishlist

- Point clouds: x_i → embed into latent space → $x_{i\alpha}$



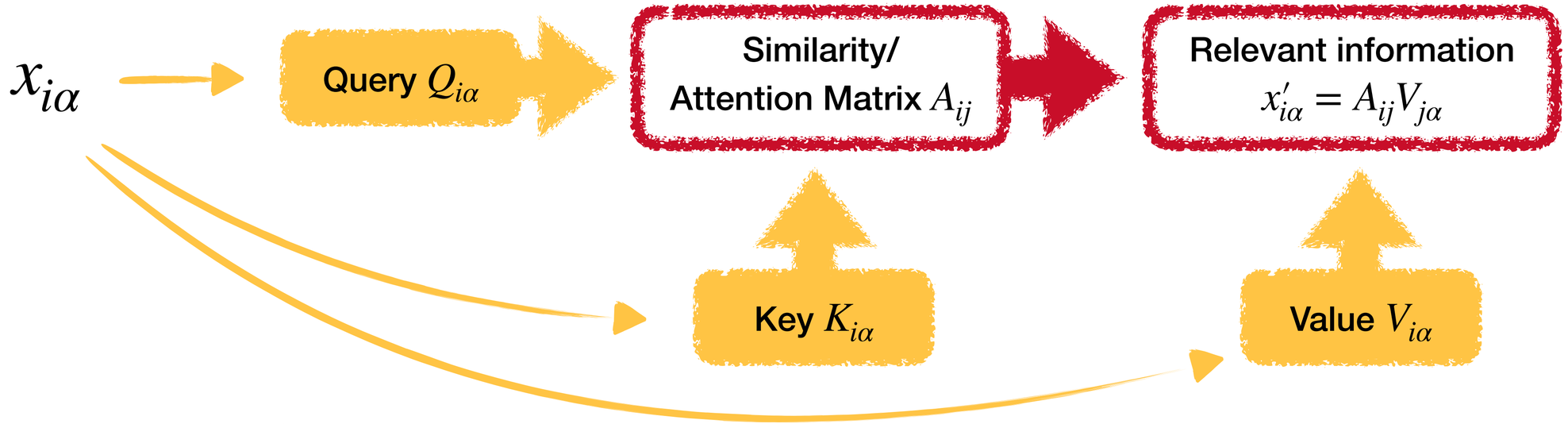
Data has shape (batchsize, #tokens, #latent dims)

- Update operation $x'_{i\alpha} = f_{i\alpha}(x_{j\beta})$
  - Exchange information along the ‘ML’ index ✓ Linear layer
  - Exchange information along the ‘Point cloud’ index while keeping permutation symmetry $f_{(\mathscr{P}i)\alpha}(x_{j\beta}) = f_{i\alpha}(x_{(\mathscr{P}j)\beta})$? Attention!

$$a_i b_i := \sum_i a_i b_i$$

Attention

Overview



Construct Q, K, V
($W^X_{\alpha\beta}$ learnable)

$$Q_{i\alpha} = W^Q_{\alpha\beta} x_{i\beta}$$
$$K_{i\alpha} = W^K_{\alpha\beta} x_{i\beta}$$
$$V_{i\alpha} = W^V_{\alpha\beta} x_{i\beta}$$

Attention matrix

$$A_{ij} = \mathrm{Softmax}_j\left(\frac{Q_{i\alpha} K_{j\alpha}}{\sqrt{d}}\right)$$

Attention

A generic update rule

- Wishlist: Update operation $x'_{i\alpha} = f_{i\alpha}(x_{j\beta})$
  - Exchange information along the ‘ML’ index ✓ Linear layer
  - Exchange information along the ‘Point cloud’ index while keeping permutation symmetry $f_{(\mathscr{P}i)\alpha}(x_{j\beta}) = f_{i\alpha}(x_{(\mathscr{P}j)\beta})$? Attention!

Attention

A generic update rule

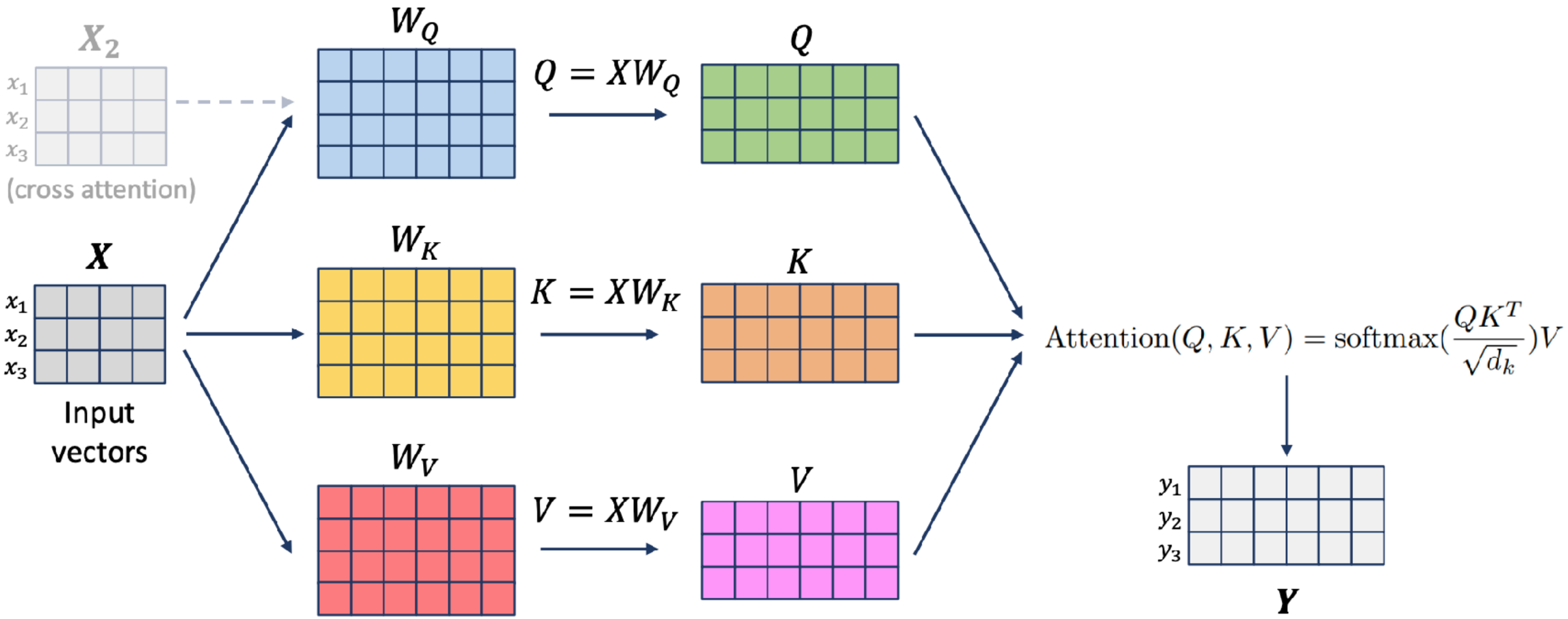
- Wishlist: Update operation $x'_{i\alpha} = f_{i\alpha}(x_{j\beta})$
  - Exchange information along the ‘ML’ index ✓ Linear layer
  - Exchange information along the ‘Point cloud’ index while keeping permutation symmetry $f_{(\mathscr{P}i)\alpha}(x_{j\beta}) = f_{i\alpha}(x_{(\mathscr{P}j)\beta})$ ✓ Attention!

$$x'_{i\alpha} = A_{ij} W^V_{\alpha\beta} x_{j\beta}$$

$$A_{ij}(x) = \mathrm{Softmax}_j\left(\frac{x_{i\alpha} W^Q_{\alpha\beta} W^K_{\beta\gamma} x_{j\gamma}}{\sqrt{d}}\right)$$

Attention

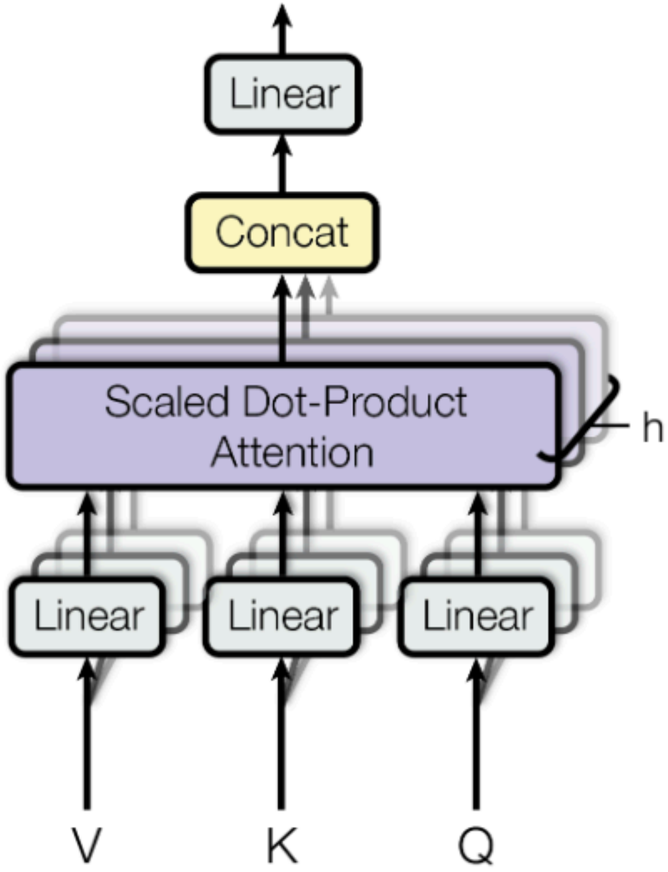
Another perspective



Attention

Multi-headed attention

1. Split latent space into h sub-latent-spaces/heads
2. Perform single-headed attention on each sub-latent-space
3. Concatenate the results for all sub-latent-spaces



- Similar to filters for different features
- Heads are independent of each other
  ⇒ Can be trained in parallel on multiple opus

Attention

Self-attention vs cross-attention

- Previously: **Self-attention**
  'update x based on itself'
- Observation: Can have different point cloud size n for $Q_{i\alpha}$ and $K_{i\alpha}, V_{i\alpha}$
- **Cross-attention**:
  'update x based on itself and a **condition** y'
  - Useful for translation tasks

Generic attention

$$x'_{i\alpha} = \mathrm{Softmax}_j\left(\frac{Q_{i\beta}K_{j\beta}}{\sqrt{d}}\right)V_{j\alpha}$$

Self-attention

$$Q_{i\alpha} = W^Q_{\alpha\beta}x_{i\beta}$$
$$K_{i\alpha} = W^K_{\alpha\beta}x_{i\beta}$$
$$V_{i\alpha} = W^V_{\alpha\beta}x_{i\beta}$$

Cross-attention

$$Q_{i\alpha} = W^Q_{\alpha\beta}x_{i\beta}$$
$$K_{i\alpha} = W^K_{\alpha\beta}y_{i\beta}$$
$$V_{i\alpha} = W^V_{\alpha\beta}y_{i\beta}$$

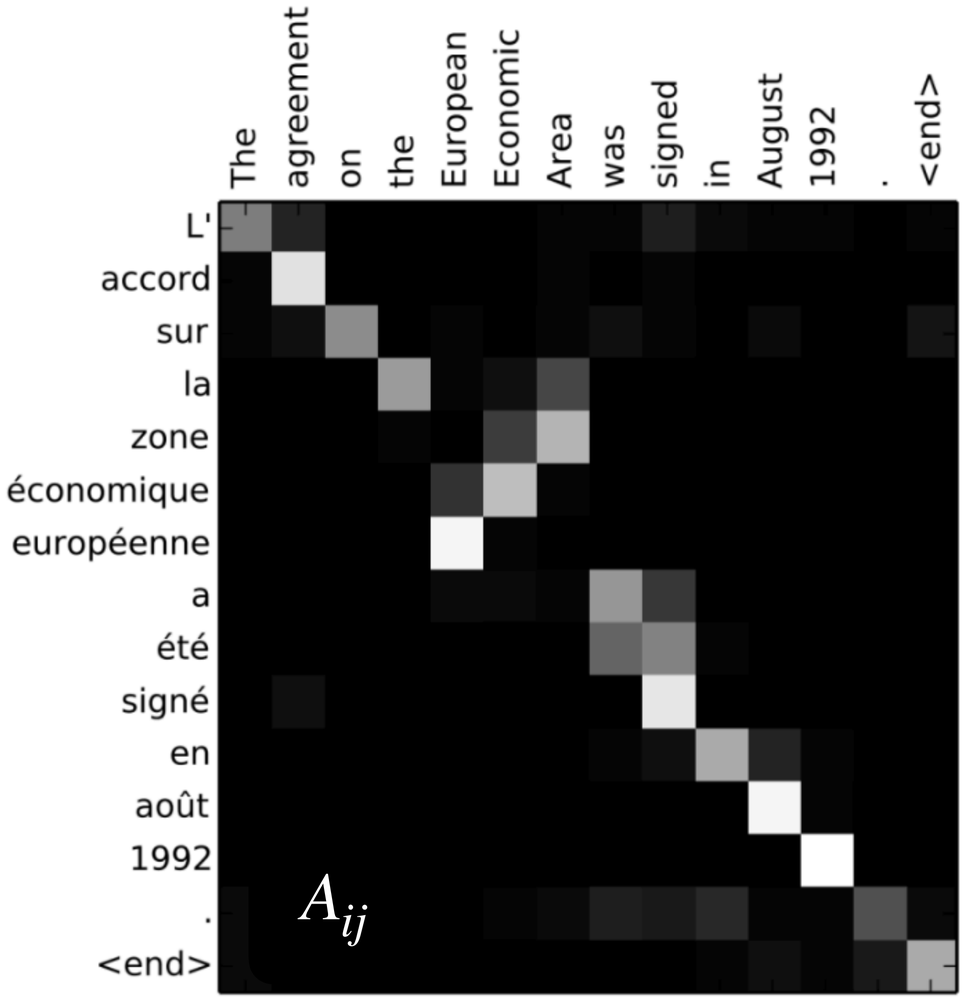
Attention

Visualising the attention matrix

Self-attention

	Hello	I	love	you
Hello	0.8	0.1	0.05	0.05
I	0.1	0.6	0.2	0.1
love	0.05	0.2	0.65	0.1
you	0.2	0.1	0.1	0.6

Cross-attention



Attention

More tricks

- **Learnable query**
  - Can be useful in special cases
- **Multi-query**: Use the same query for each attention head
  - Slightly reduce memory consumption and #parameters

Exercises

Generic attention

$$x'_{i\alpha} = \mathrm{Softmax}_j\left(\frac{Q_{i\beta}K_{j\beta}}{\sqrt{d}}\right)V_{j\alpha}$$

Self-attention

$$Q_{i\alpha} = W^Q_{\alpha\beta}x_{i\beta}$$
$$K_{i\alpha} = W^K_{\alpha\beta}x_{i\beta}$$
$$V_{i\alpha} = W^V_{\alpha\beta}x_{i\beta}$$

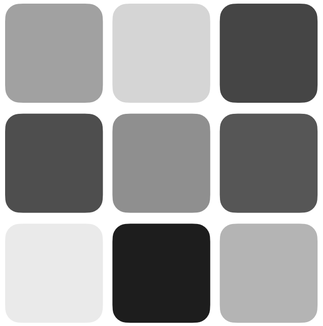
Learnable query

$$Q_{i\alpha} = W^Q_{\alpha}$$
$$K_{i\alpha} = W^K_{\alpha\beta}x_{i\beta}$$
$$V_{i\alpha} = W^V_{\alpha\beta}x_{i\beta}$$

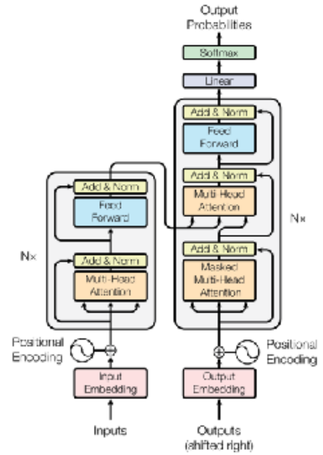
Roadmap

POINT CLOUDS

ATTENTION



TRANSFORMERS



EXAMPLES

LET’S HAVE A 5MIN BREAK

Transformers

Attention is all you need

- Transformer = Network architecture based solely on attention mechanisms
- Proposed 2017 in ‘Attention is all you need’ (already 120k citations)
- State of the art across many different tasks

1706.03762

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

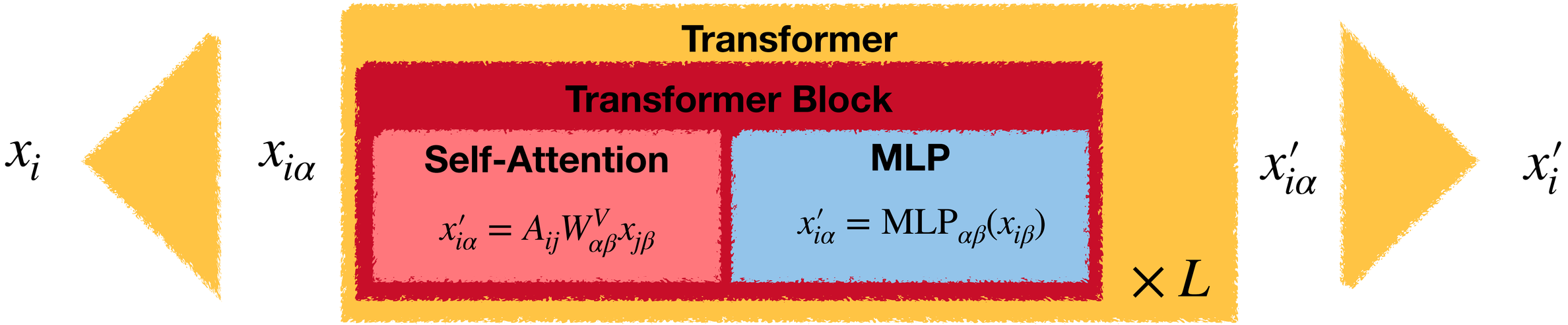
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

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

Transformers

Overview



Embed into latent space

‘Exchange information across the point cloud’

Nonlinearity + ‘Clean up latent space’

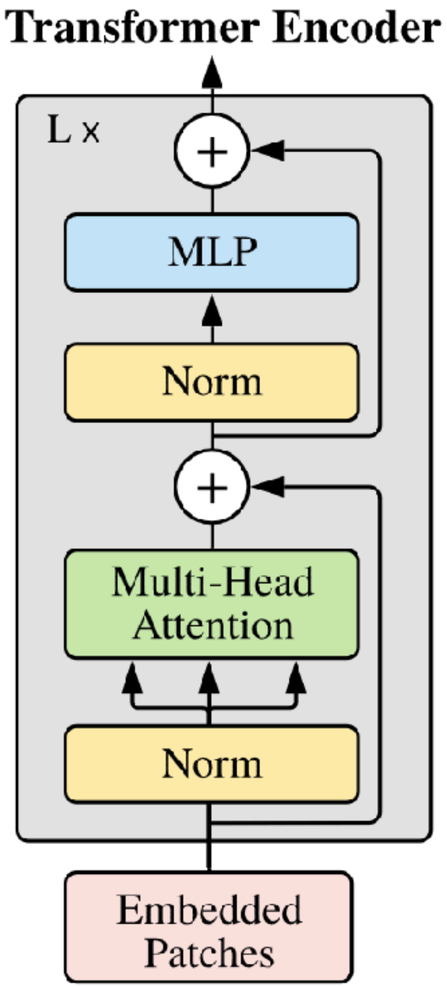
Extract from latent space

Transformers

Implementation

- **Residual connections** ⇒ Helps for deep networks
- **LayerNorm** ⇒ Numerical stability
- **Dropout** before residual connections ⇒ Regularisation
- Off-the-shelf implementation:
  `torch.nn.TransformerEncoderLayer`

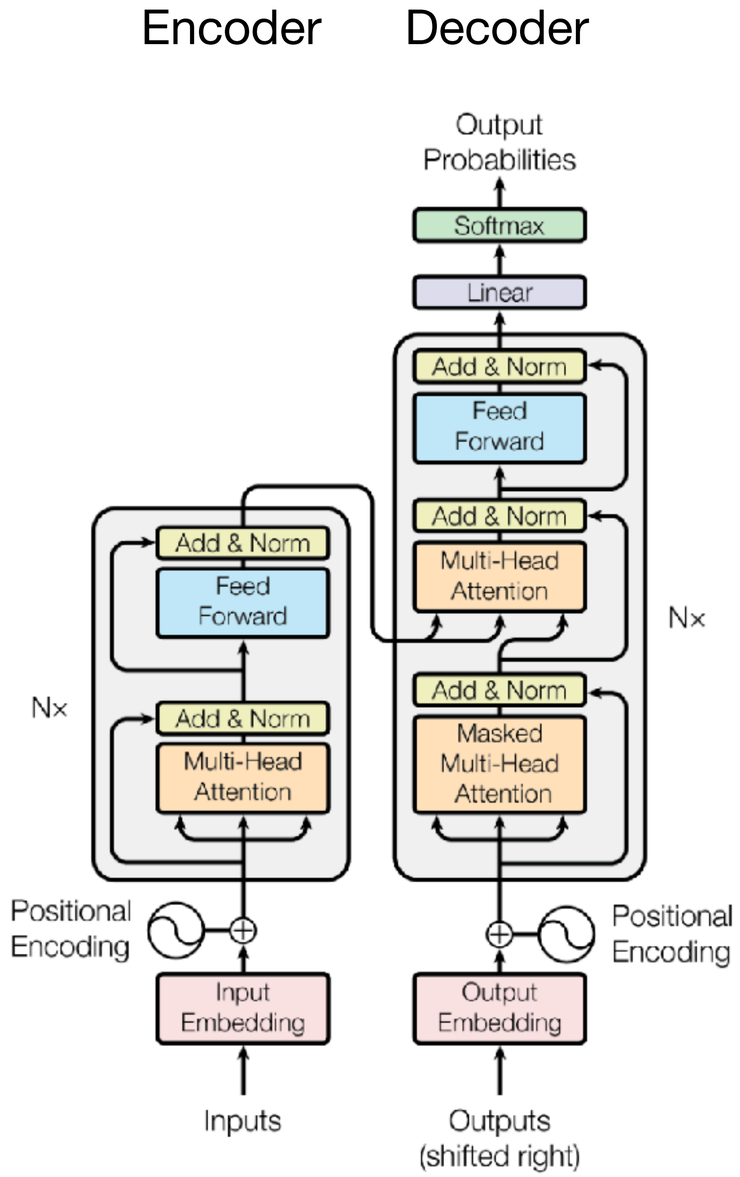
Exercises



Transformers

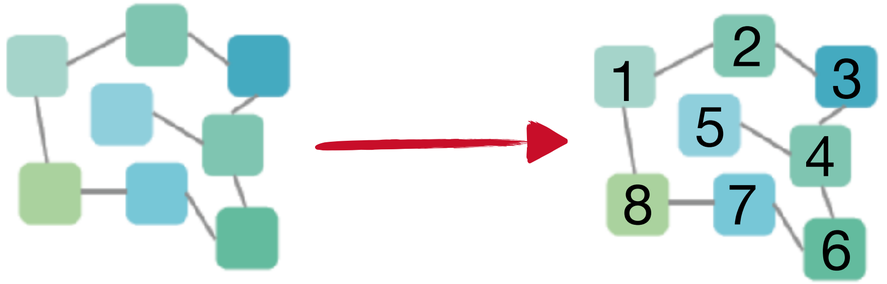
Conditional Transformer

- Previously: **TransformerEncoder** `torch.nn.TransformerEncoder`
  - Use self-attention to update prediction
- **Encoder-Decoder structure** `torch.nn.Transformer`
  - Combine Encoder (process condition) and Decoder (integrate condition into prediction)
  - Use self-attention and cross-attention



Transformers

Breaking permutation symmetry



Discrete classes
(no measure of closeness, limited sequence length)
e.g. particle type, profession of a person

Continuous index
(generalization)
e.g. time, sequence length, position in image

One-hot encoding

$0 \to (1,0,0,0,\ldots 0)$

$2 \to (0,0,1,0,\ldots 0)$

Fourier embedding

$$f(t,\alpha) = \begin{cases} \sin(\omega^{-2\alpha/d}t) & \alpha \text{ even} \\ \cos(\omega^{-2\alpha/d}t) & \alpha \text{ odd} \end{cases}$$

and $\omega \sim 10^{-4}$

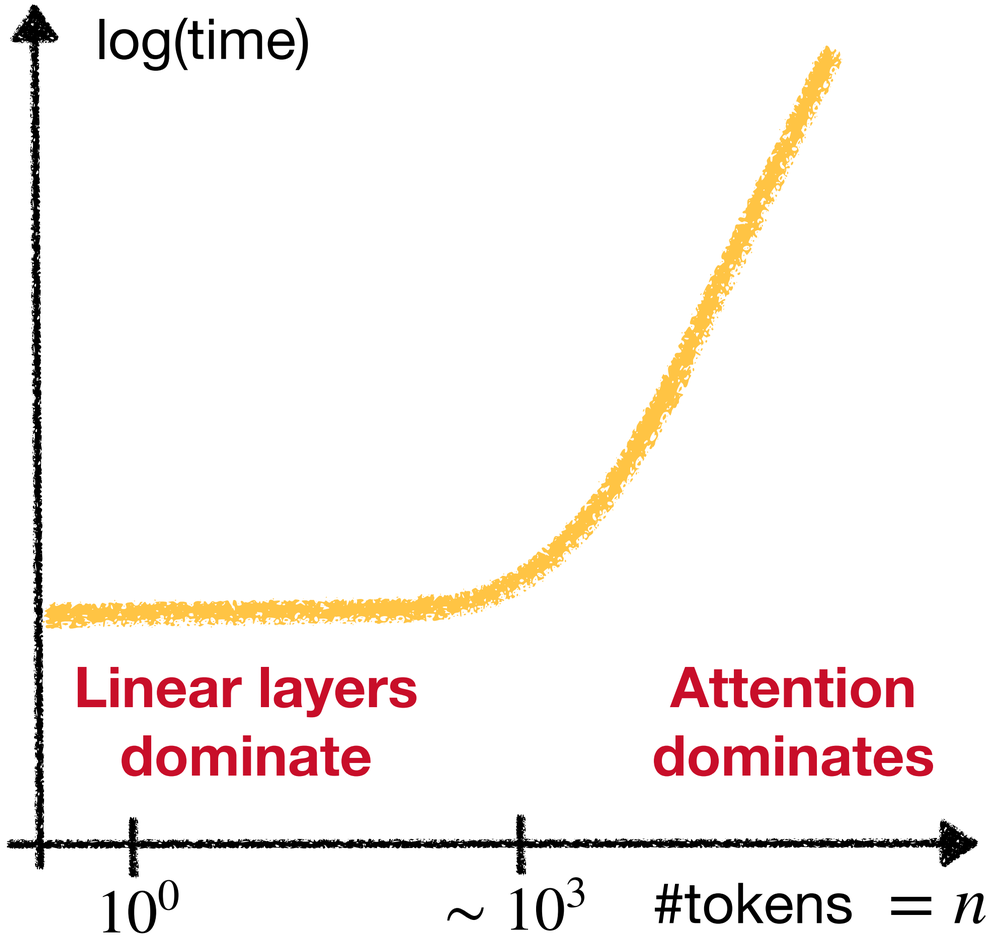
Rotary Positional Embeddings (RoPE)
2104.09864
Add Fourier embeddings onto keys in each attention step

Transformers

Scaling things up

- Attention complexity $\mathcal{O}(n^2)$
- **Too many tokens?** ⇒ Tricks to delay scaling
  - Reformer $\mathcal{O}(n \log n)$
  - Sparse Transformer $\mathcal{O}(n\sqrt{n})$
  - Perceiver $\mathcal{O}(kn)$

  Many more: 2009.06732
- **Very few tokens?** ⇒ Can add pair-tokens
  - Idea: On top of n tokens, add n^2 pair-tokens
  - Pair-tokens contain pairwise features
  - Information is **redundant**, but can help

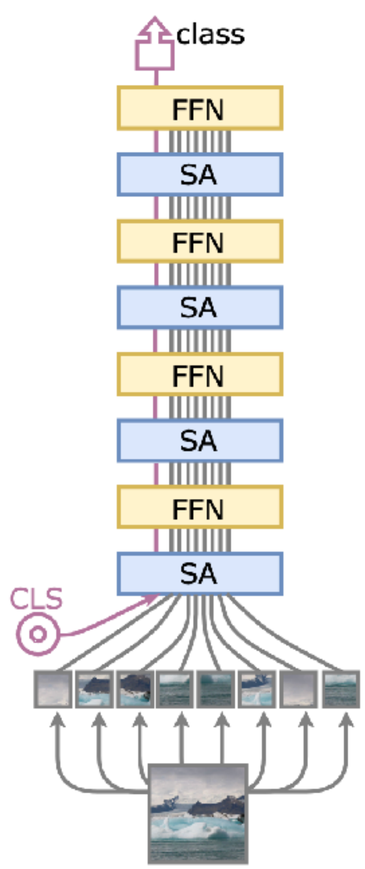


Transformers

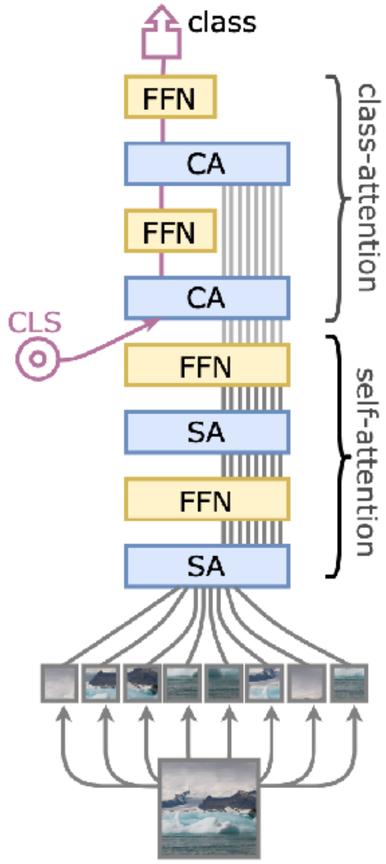
Aggregating point clouds

- Challenge: Extract scalar value x_α from point cloud $x_{i\alpha}$
  - x_α should be permutation invariant
- Simple approach: **Mean aggregation** $x_\alpha := \frac{1}{n} \sum_i x_{i\alpha}$
- Advanced: Add extra **global/class token** (CLS) to collect global information
  - Handle to add global information already in embedding
  - **CaiT**: Formally separate attention and extraction stage

2103.17239



Global token



CaiT

Transformers

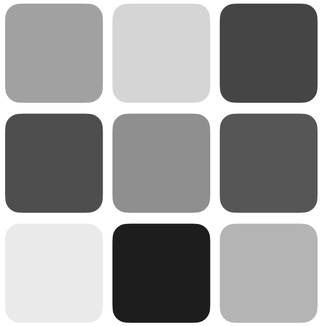
Foundation models

- Transformers are effective in constructing **abstract representations** of data
  - Scaling up #data and #parameters seems to improve performance more than modifications to the architecture (e.g. GPT)
- The **pre-training paradigm**:
  1. Train transformer on variety of carefully designed pre-training tasks (huge amount of unsupervised data)
  2. Take pre-trained model and fine-tune on specific downstream tasks
  - Setup: Generic transformer backbone + task-specific head
- Foundation models for physics? 2401.13537 2403.07066 2403.05618 2404.16091

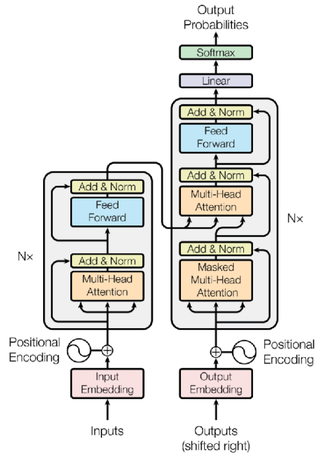
Roadmap

POINT CLOUDS

ATTENTION



TRANSFORMERS

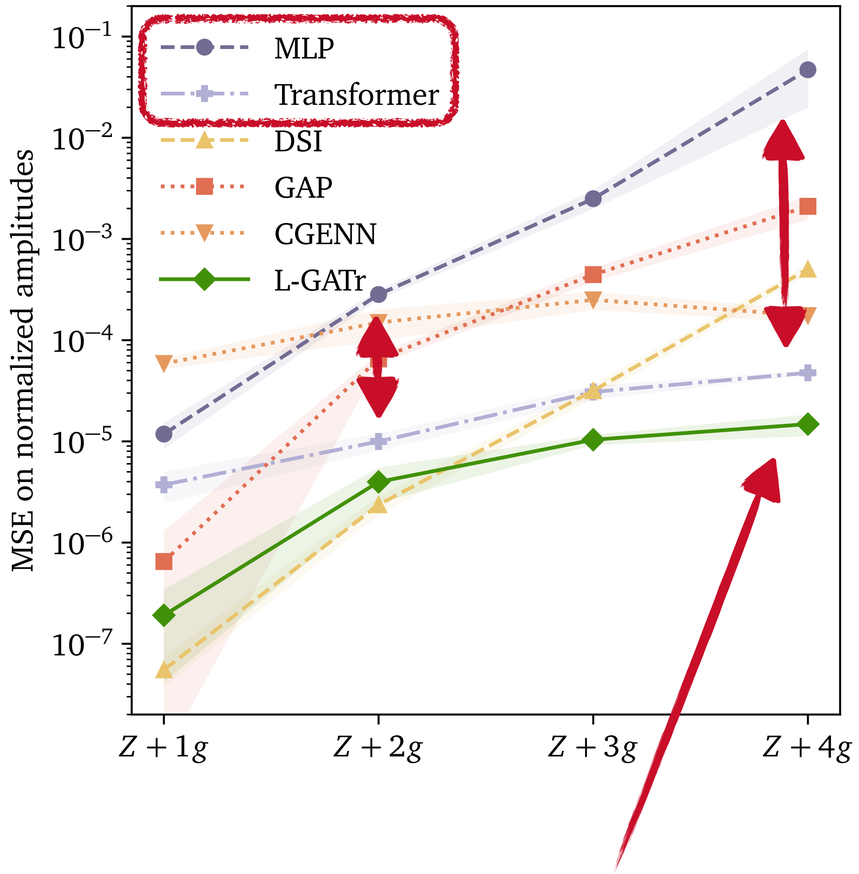


EXAMPLES

Examples

QFT Amplitude regression

- Task: Learn expensive function $f(p_1, \ldots p_n)$ with particles $p_i = (E, p_x, p_y, p_y) \in \mathbb{R}^4$
  - Fixed dimensionality - no ‘true’ point cloud problem
- Token embedding: $(p_i, \text{particle id}_i)$
- Extract information from global token
- Transformer outperforms MLP for complex processes



Transformer with built-in Lorentz symmetry

2202.03772

Examples

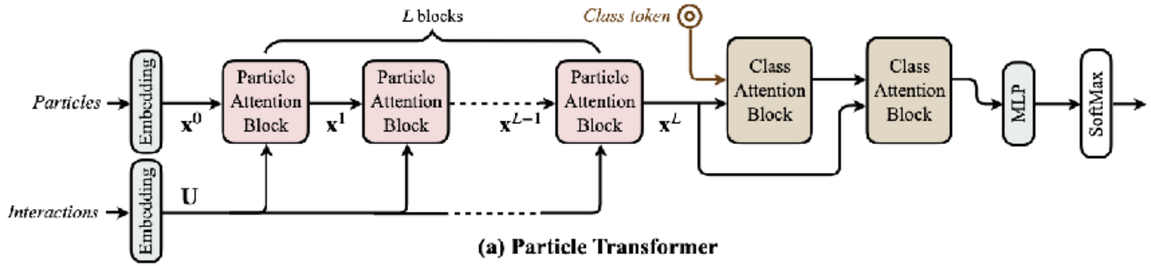
Exercises

Jet tagging: Particle Transformer (ParT)

- Task: Classification on jets $(p_1, \ldots p_n)$ with variable n
  - Variable-dimensional inputs and permutation symmetry
  - Extract classifier score with CaiT approach
- Trick: Attention bias U_{ij} based on pairwise features:

$$A_{ij} = \mathrm{Softmax}_j\left(\frac{Q_{i\alpha}K_{j\alpha}}{\sqrt{d}} + U_{ij}\right)$$

- Pretrained on 100M samples and fine-tuned on 1M samples ⇒ ParT outperforms all other approaches



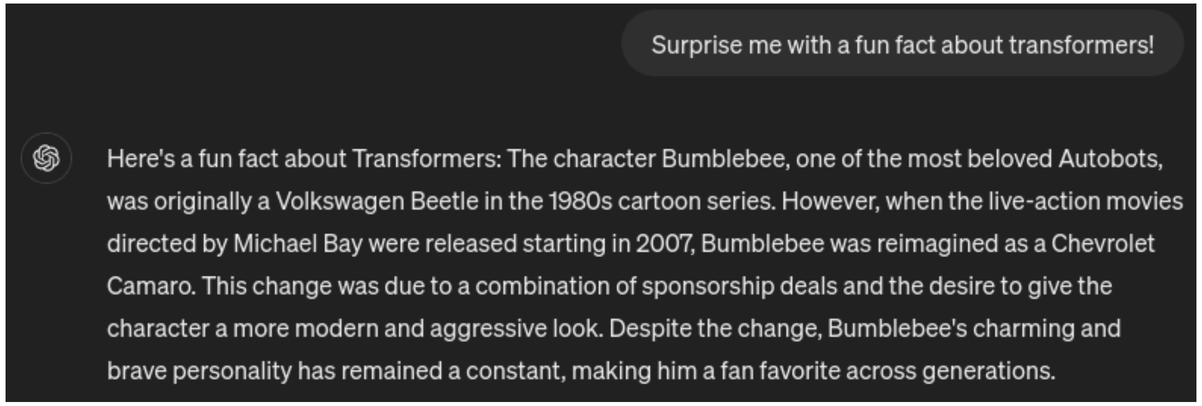
(a) Particle Transformer

	Accuracy	AUC	$\mathrm{Rej}_{50\%}$	$\mathrm{Rej}_{30\%}$
P-CNN	0.930	0.9803	201 ± 4	759 ± 24
PFN	—	0.9819	247 ± 3	888 ± 17
ParticleNet	0.940	0.9858	397 ± 7	1615 ± 93
JEDI-net (w/ $\sum O$)	0.930	0.9807	—	774.6
PCT	0.940	0.9855	392 ± 7	1533 ± 101
LGN	0.929	0.964	—	435 ± 95
rPCN	—	0.9845	364 ± 9	1642 ± 93
LorentzNet	0.942	0.9868	498 ± 18	2195 ± 173
ParT	0.940	0.9858	413 ± 16	1602 ± 81
ParticleNet-f.t.	0.942	0.9866	487 ± 9	1771 ± 80
ParT-f.t.	**0.944**	**0.9877**	$\mathbf{691 \pm 15}$	$\mathbf{2766 \pm 130}$

Examples

Text generation: Generative Pre-trained Transformer (GPT)

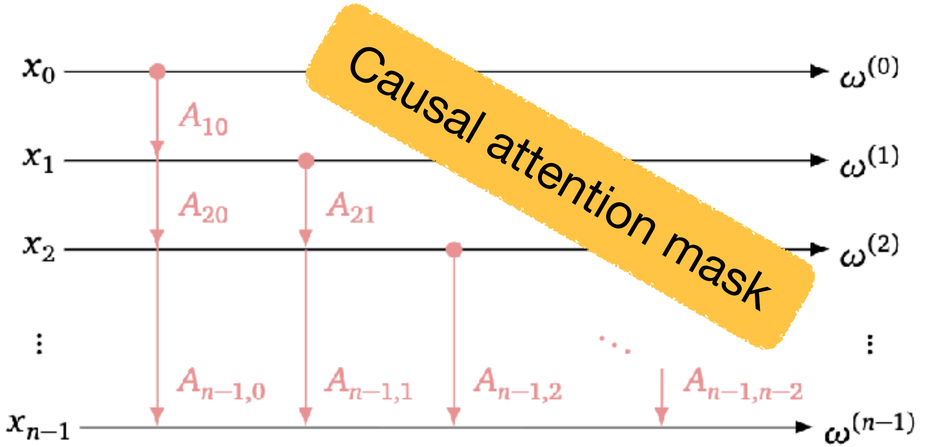
- Task: Generate text autoregressively
  - Encode text using ‘byte-pair encoding’
  - Causal attention mask $A_{ij} = 0$ for $j > i$



- Transformer autoregressively predicts probability $\omega^{(i)}$ for each possible token to come next

$$
\begin{aligned}
& p(x_1, x_2, \ldots x_n) \\
&= p(x_1)p(x_2 \mid x_1) \cdots p(x_n \mid x_{n-1}) \\
&= p(x_1 \mid \omega^{(0)})p(x_2 \mid \omega^{(1)}) \cdots p(x_n \mid \omega^{(n-1)})
\end{aligned}
$$

GitHub: nanogpt



- Pre-train Transformer on the internet (Wikipedia +arxiv +stackexchange), then fine-tune on high-quality dataset

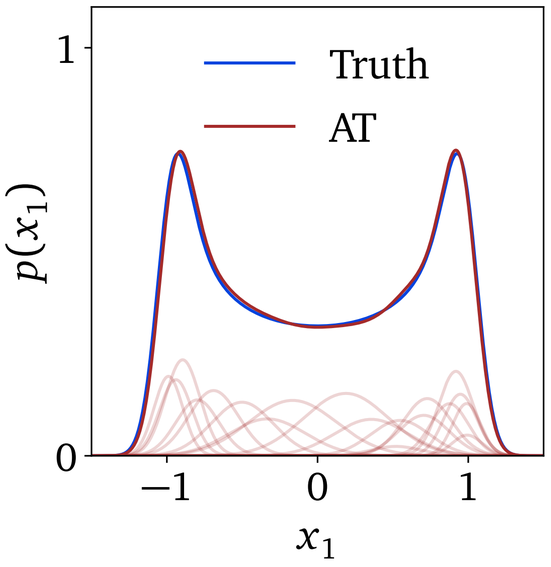
Examples

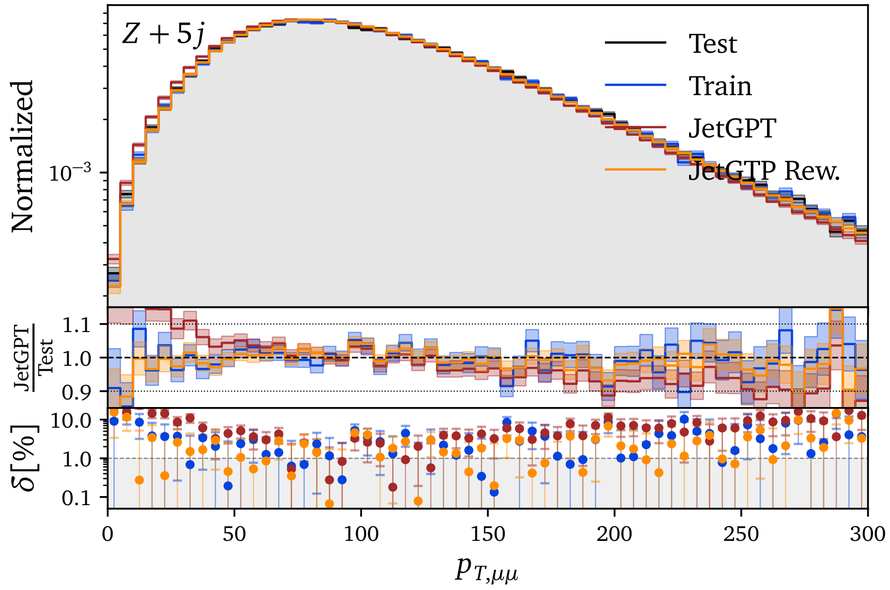
LHC Event generation: JetGPT

- Task: Generate events $(p_1, \ldots p_n)$ with fixed n with same distribution as a given set of events
  - Same embedding as for amplitude regression
- Autoregressive approach: Generate particles one by one

$$p(x_{i+1} \,|\, \omega^{(i)}) = \sum_{j=1} w_j^{(i)} \mathcal{N}(x_{i+1} \,|\, \mu_j^{(i)}, \sigma_j^{(i)})$$

$$\omega^{(i)} = \{w_j^{(i)}, \mu_j^{(i)}, \sigma_j^{(i)}\}$$





Examples

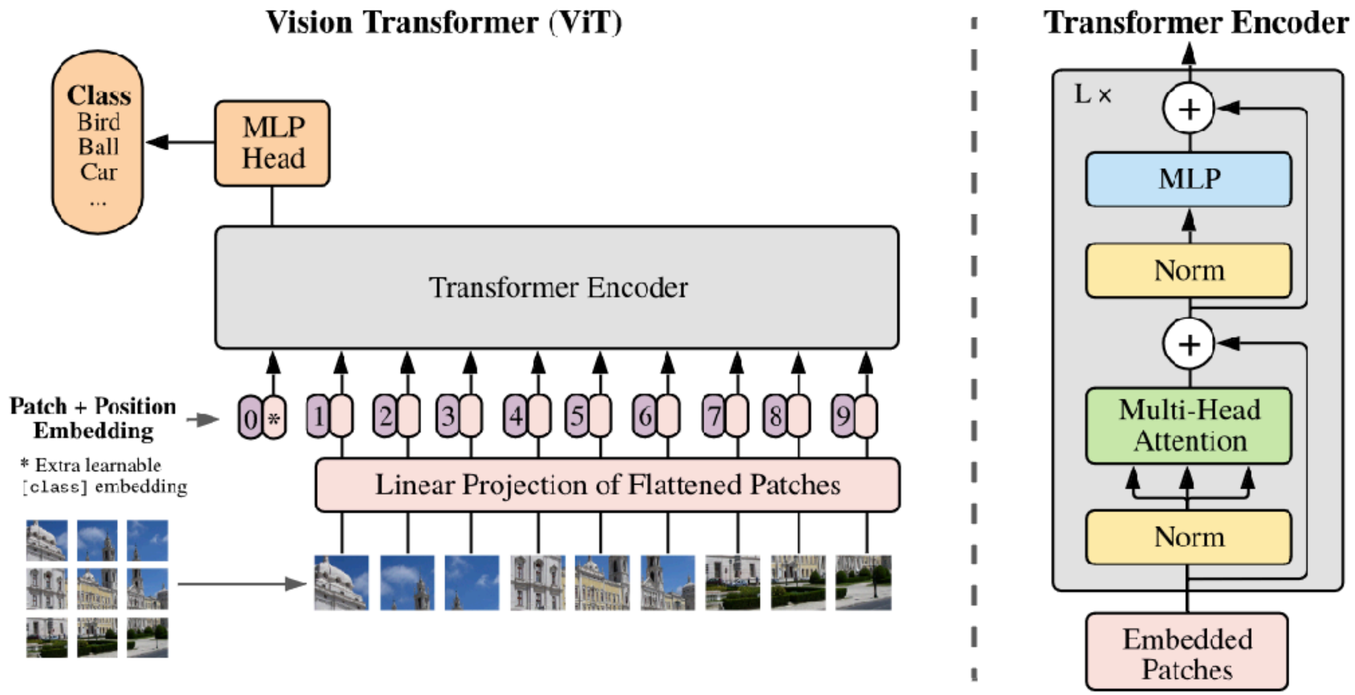
Image classification: Vision Transformer (ViT)

Original ViT: 2010.11929

GitHub ViT

ViT for CaloSim: 2405.09629

- Classical approach to image processing: **Convolutional NNs**
- ViT idea
  1. Slice image into a set of patches
  2. Push patches through transformer
- ViT outperforms highly optimised CNNs, and starts to be applied in physics

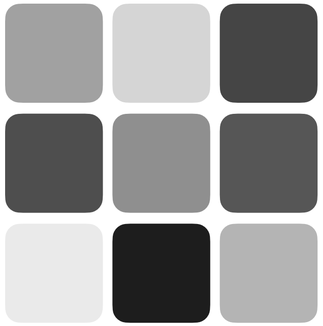


Summary

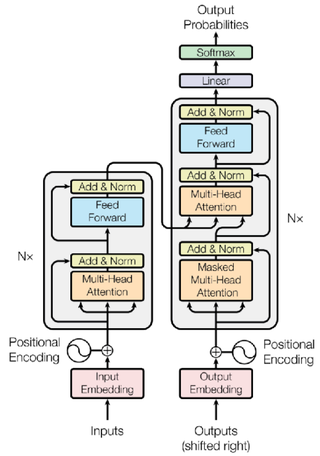
- Point clouds are everywhere!
- Transformers are efficient, flexible and scalable for handling point clouds

POINT CLOUDS

ATTENTION



TRANSFORMERS



EXAMPLES

Bonus

Why scaled dot-product attention?

Attention matrix

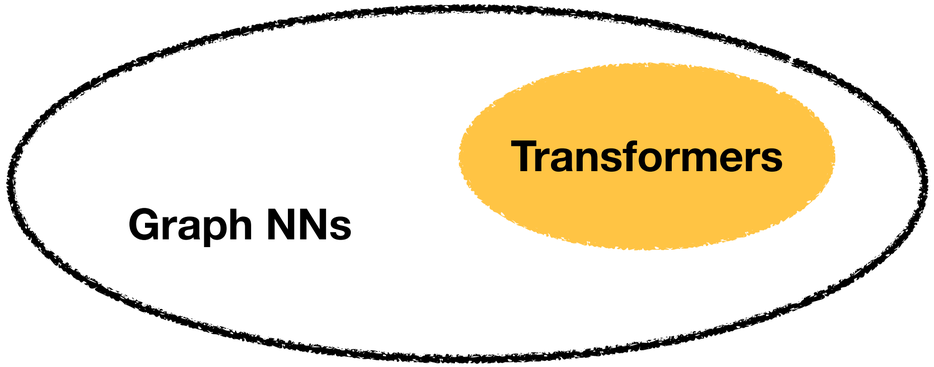
$$A_{ij} = \mathrm{Softmax}_j\left(\frac{Q_{i\alpha}K_{j\alpha}}{\sqrt{d}}\right)$$

- $Q_{i\alpha}K_{j\alpha}$: Extend the simplest choice $x_{i\alpha}x_{j\alpha}$ by learnable weights
- $\frac{1}{\sqrt{d}}$: Standardize $Q_{i\alpha}K_{j\alpha}$ assuming individual terms in the sum are $\sim \mathcal{N}(0,1)$
- $\mathrm{Softmax}_j$: Normalize results for numerical stability

$$\mathrm{Softmax}_j(x_j) = \frac{e^{x_j}}{\sum_k e^{x_k}}$$

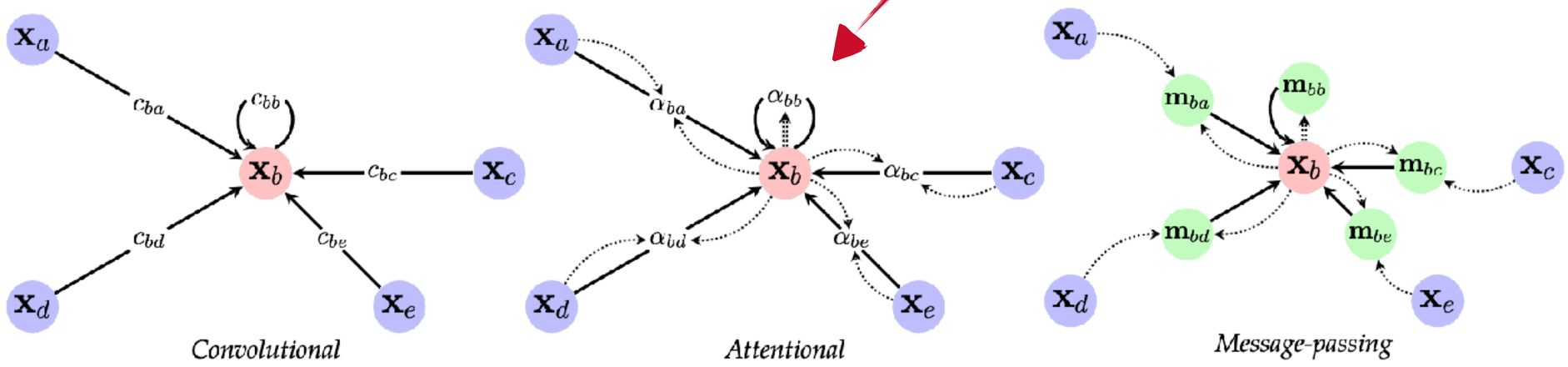
Bonus

Transformers are Graph NNs



- Graph NN (GNN): Neural network operating on graphs
  - Graph = Set of nodes and edges
- Three flavours of GNN layers:

Transformers live here



Standard reference for GNNs: 2104.13478