SPONSORED BY THE

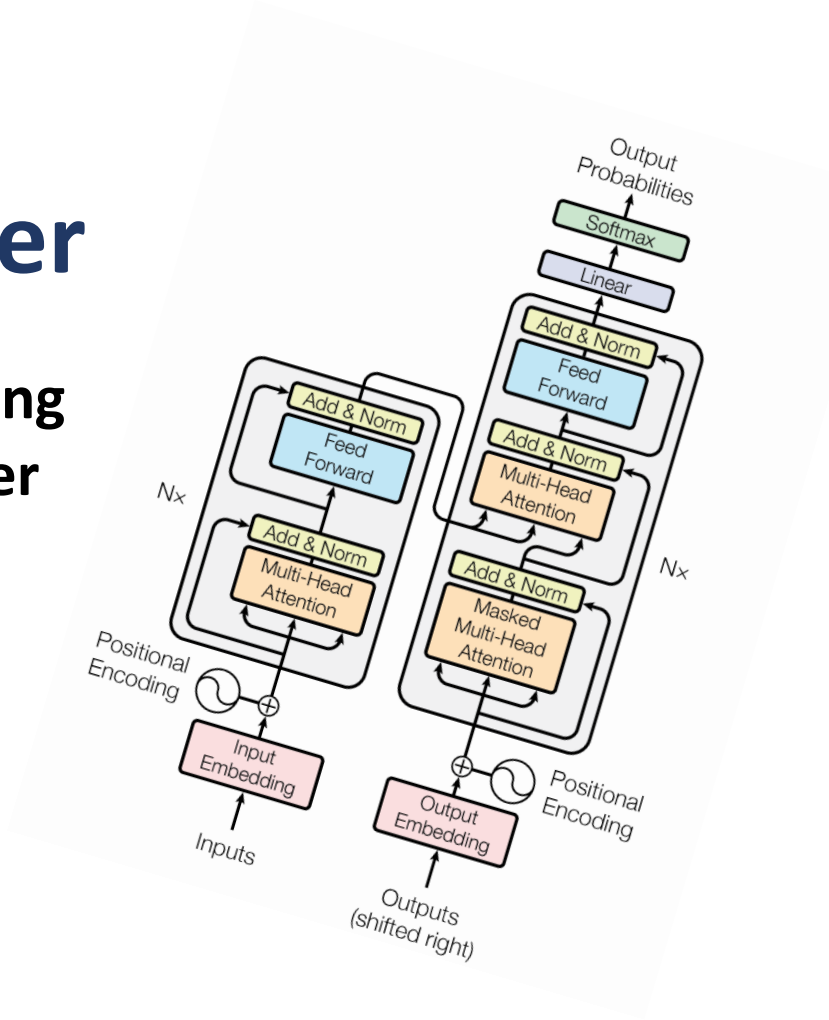
Federal Ministry of Education and Research

III. Physikalisches Institut A | RWTH AACHEN UNIVERSITY

Hackathon: Transformer

Conceptual Advances in Deep Learning for Research on Universe and Matter

ERUM DATA HUB

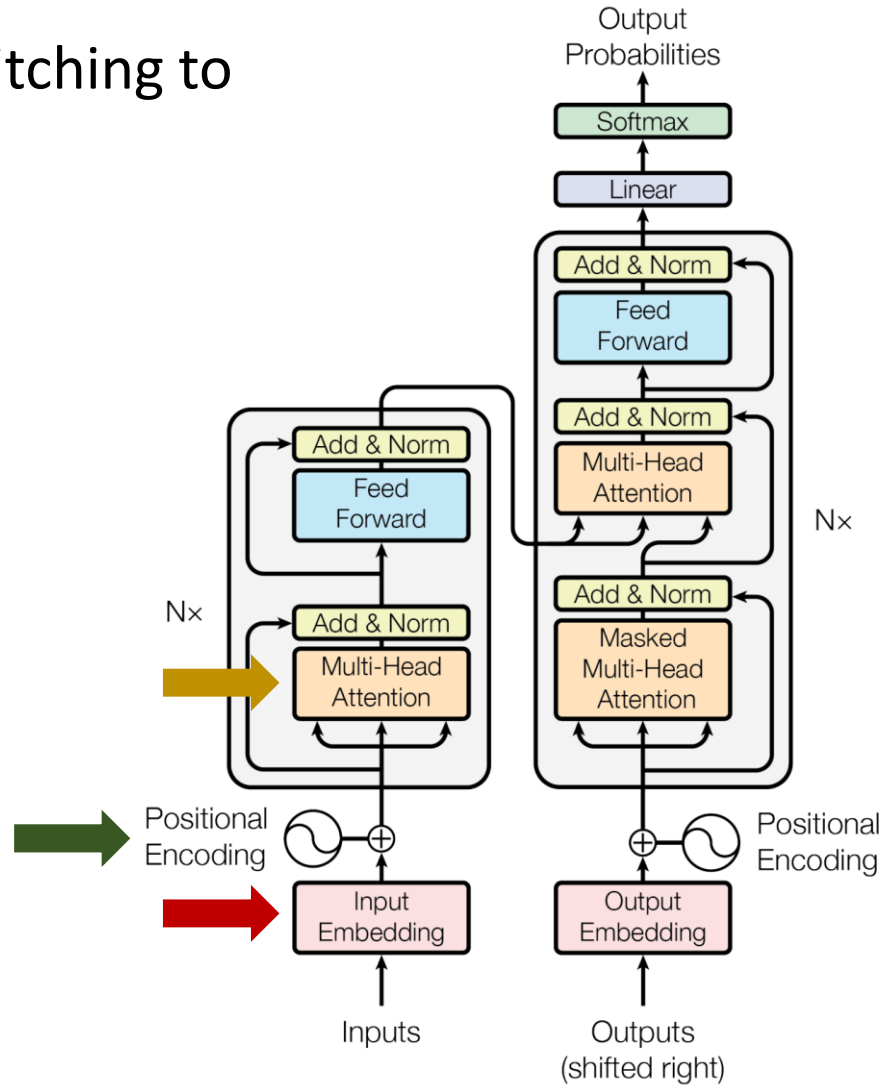


Niklas Langner
RWTH Aachen University

Transformer: What's new?

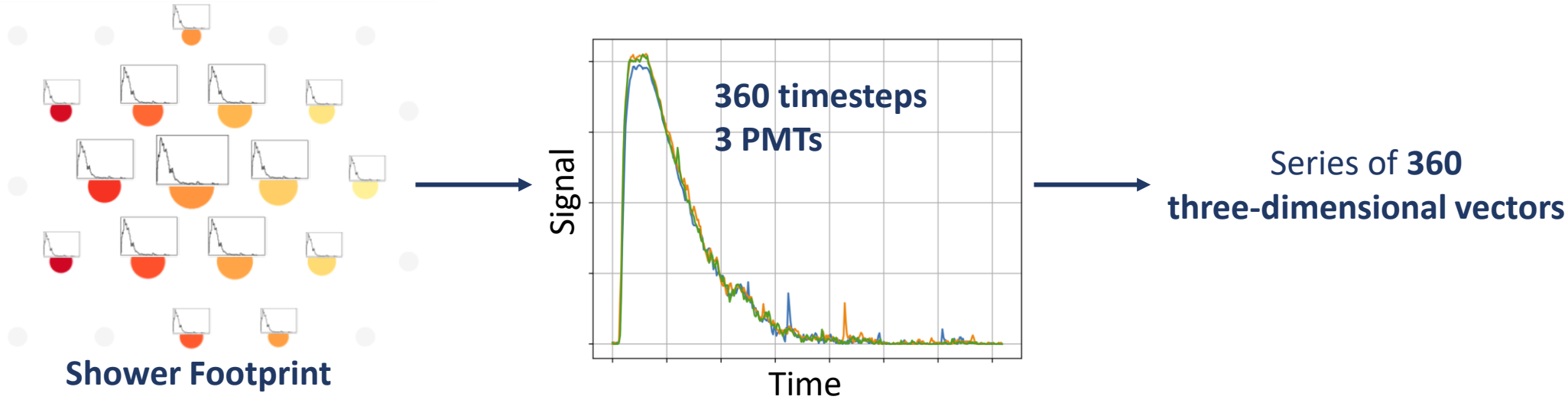
Three things to consider when switching to transformers:

1. Embedding
2. Positional Encoding
3. Multi-Head Attention



Embedding

- Input must be a **series/set of vectors**
- Measurements often already have shape of vectors
  Example:



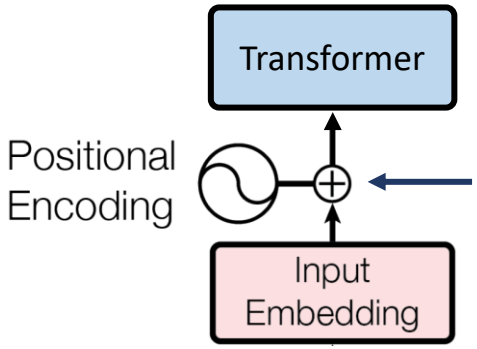
- Otherwise implement **embedding to obtain vectors**:
  - **Fixed set of inputs** (words, classes,…)**:** e.g. Keras Embedding Layer
  - **Varying inputs** (image patches, higher dimensional data): e.g. Keras Dense Layer

Many more possibilities!

Positional Encoding

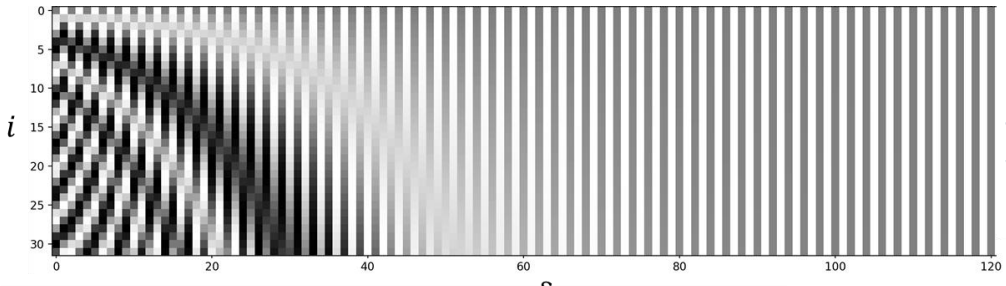
Transformer does not inherently treat input as time series → **add order information**

Principle



Add vectors (identical set for all samples) to embedding vectors

Non-Trainable (Example)



(or any other fixed set of vectors)

$$\mathrm{PE}(i,\delta) = \begin{cases} \sin\left(\frac{i}{10000^{2\delta'/d}}\right) & \text{if } \delta = 2\delta' \\ \cos\left(\frac{i}{10000^{2\delta'/d}}\right) & \text{if } \delta = 2\delta' + 1 \end{cases}$$

i: Time bin
δ: vector dimension
d: network dims.

Trainable (Examples)

Keras Embedding Layer

↓

Learnable vector for each input position

$$\tilde{\boldsymbol{x}}_i(t_{i,j}) = \phi_{\mathrm{tnh}}(\boldsymbol{\gamma}(t_{i,j})) \odot \dot{\boldsymbol{x}}_i(t_{i,j} + \varepsilon_t) + \boldsymbol{\beta}(t_{i,j}), \quad (26)$$

$$\tilde{\boldsymbol{x}}'_i(t_{i,j}) = \phi_{\mathrm{ReLU}}\left(\boldsymbol{W}^T \tilde{\boldsymbol{x}}_i(t_{i,j}) + \boldsymbol{b}\right), \quad (27)$$

$$\gamma_k(t) = \sum_{m=1}^{M} a'_{k,m} \sin\left(\frac{2\pi m}{T_{\max}} t\right) + b'_{k,m} \cos\left(\frac{2\pi m}{T_{\max}} t\right), \quad (28)$$

$$\beta_k(t) = \sum_{m=1}^{M} v'_{k,m} \sin\left(\frac{2\pi m}{T_{\max}} t\right) + w'_{k,m} \cos\left(\frac{2\pi m}{T_{\max}} t\right),$$

Can get very complicated!

Example: TimeFiLM

https://arxiv.org/pdf/2201.08482.pdf

Works well for non-periodic time series

Multi-Head Attention

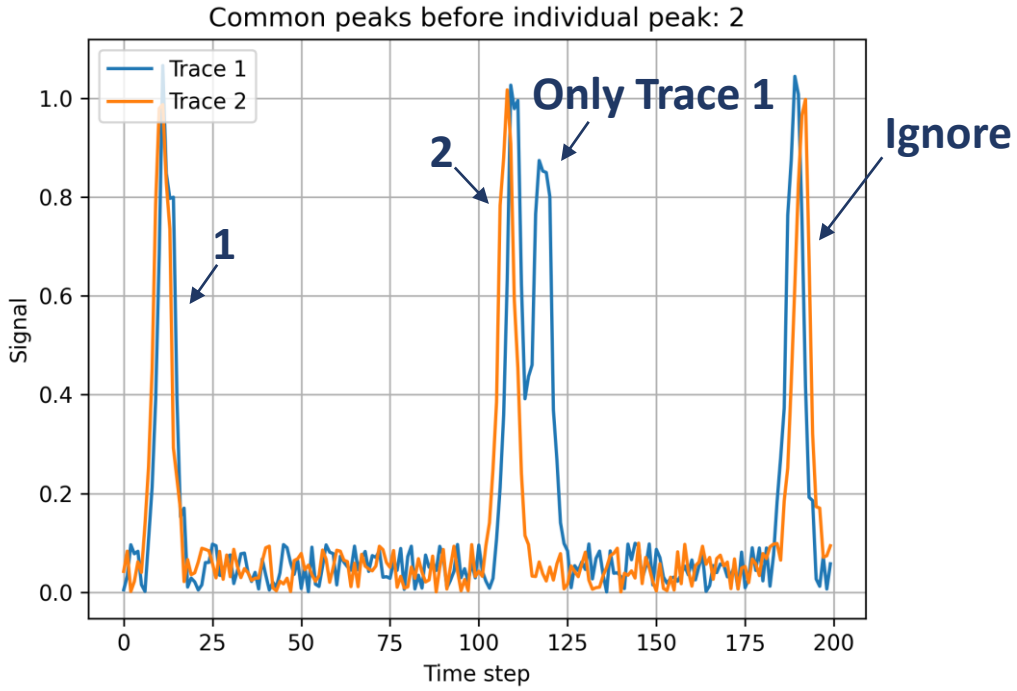
Standard implementation available in both [tensorflow/keras] and [pytorch]

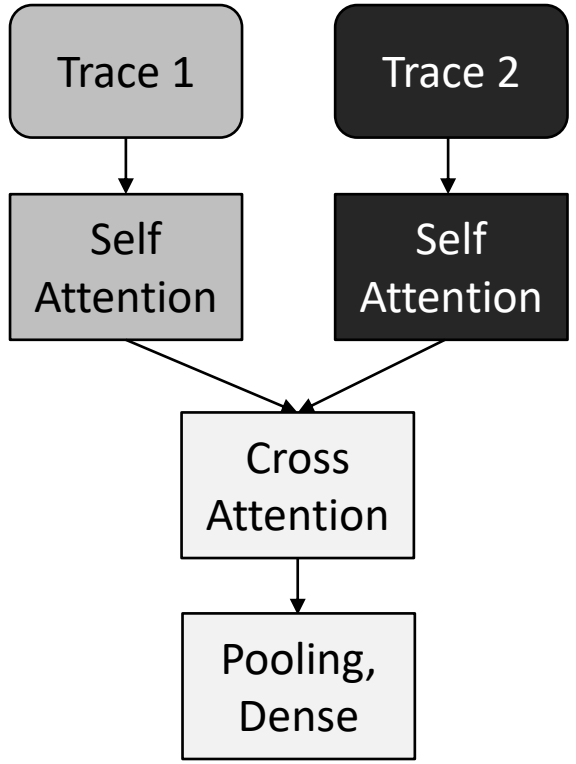
```
tf.keras.layers.MultiHeadAttention(
    num_heads,
    key_dim,
    value_dim=None,
    dropout=0.0,
    use_bias=True,
    output_shape=None,
    attention_axes=None,
    kernel_initializer='glorot_uniform',
    bias_initializer='zeros',
    kernel_regularizer=None,
    bias_regularizer=None,
    activity_regularizer=None,
    kernel_constraint=None,
    bias_constraint=None,
    **kwargs
)
```

```
CLASS  torch.nn.MultiheadAttention(
       embed_dim, num_heads,
       dropout=0.0, bias=True,
       add_bias_kv=False,
       add_zero_attn=False,
       kdim=None, vdim=None,
       batch_first=False,
       device=None, dtype=None)  [SOURCE]
```

Example: Signal Trace Analysis

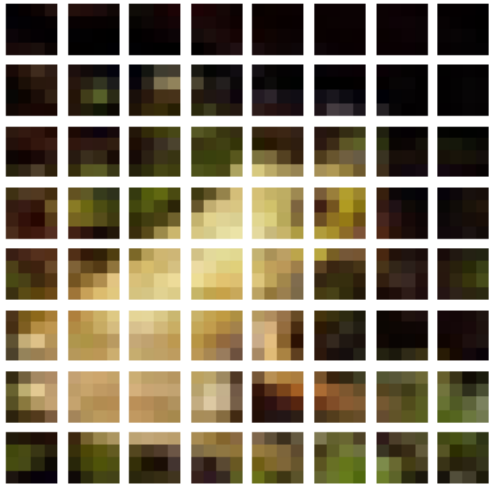
Count common peaks in two time traces before individual peak in trace 1:



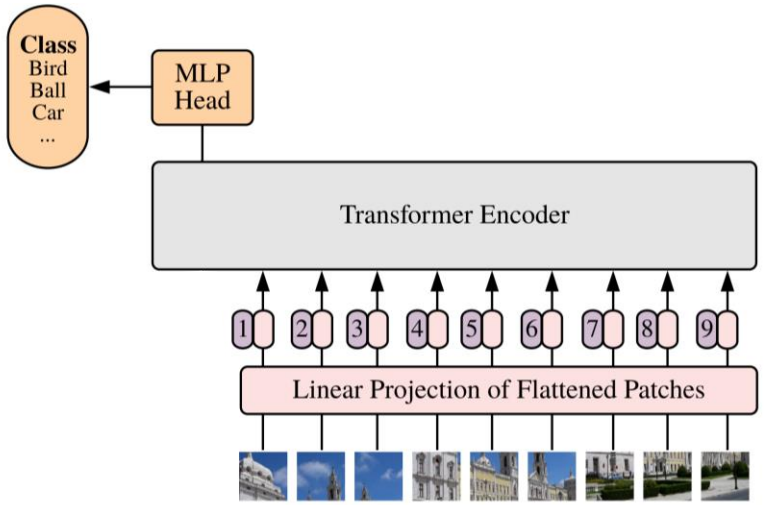


VISPA Example!

Example: Image Classification with Vision Transformer



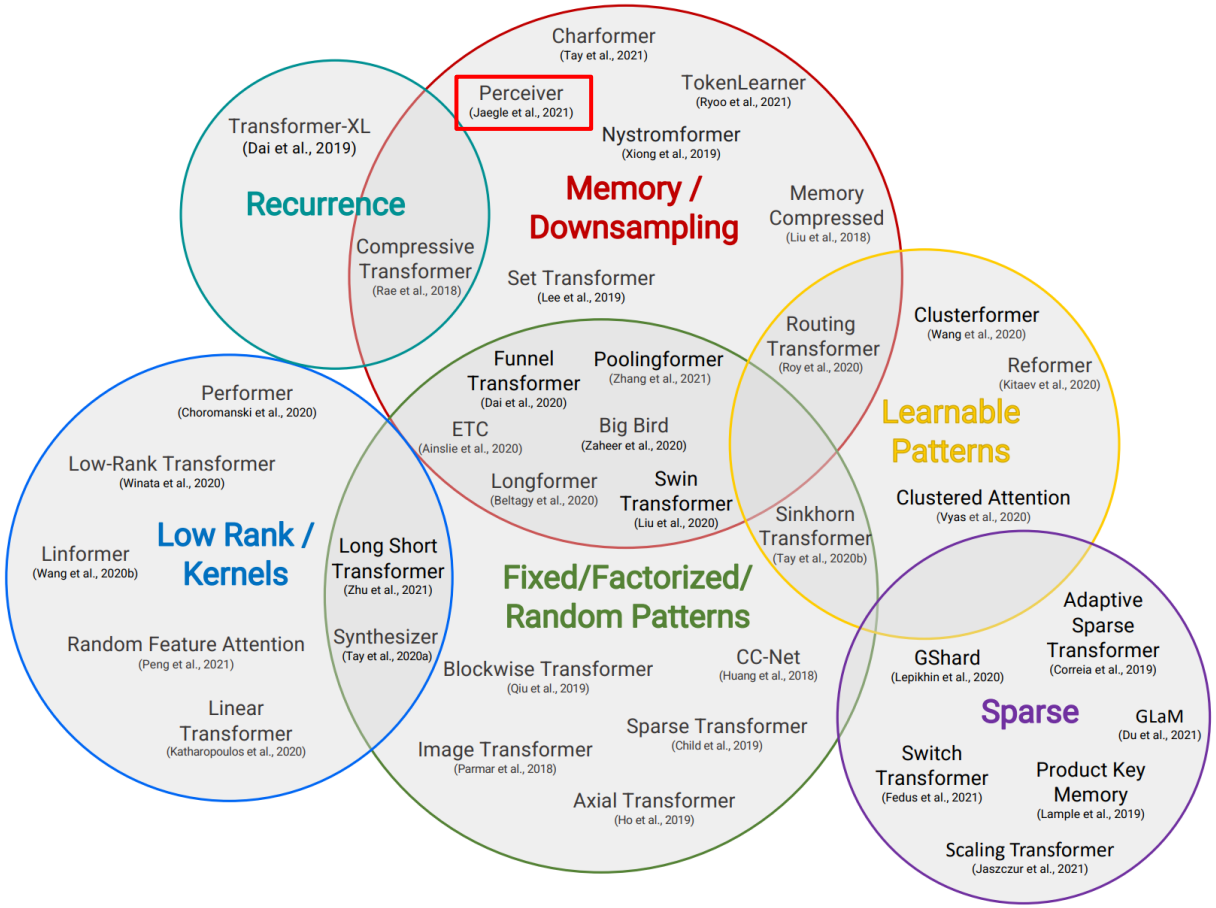
Vision Transformer



VISPA Example!

Efficient Transformer

Large inputs might lead to VRAM problems using traditional transformer → Try **efficient transformer**



https://arxiv.org/pdf/2009.06732.pdf

Many different implementations:

Model / Paper	Complexity
Memory Compressed (Liu et al., 2018)	$\mathcal{O}(N_c^2)$
Image Transformer (Parmar et al., 2018)	$\mathcal{O}(N.m)$
Set Transformer (Lee et al., 2019)	$\mathcal{O}(kN)$
Transformer-XL (Dai et al., 2019)	$\mathcal{O}(N^2)$
Sparse Transformer (Child et al., 2019)	$\mathcal{O}(N\sqrt{N})$
Reformer (Kitaev et al., 2020)	$\mathcal{O}(N \log N)$
Routing Transformer (Roy et al., 2020)	$\mathcal{O}(N\sqrt{N})$
Axial Transformer (Ho et al., 2019)	$\mathcal{O}(N\sqrt{N})$
Compressive Transformer (Rae et al., 2020)	$\mathcal{O}(N^2)$
Sinkhorn Transformer (Tay et al., 2020b)	$\mathcal{O}(B^2)$
Longformer (Beltagy et al., 2020)	$\mathcal{O}(n(k+m))$
ETC (Ainslie et al., 2020)	$\mathcal{O}(N_g^2 + NN_g)$
Synthesizer (Tay et al., 2020a)	$\mathcal{O}(N^2)$
Performer (Choromanski et al., 2020a)	$\mathcal{O}(N)$
Funnel Transformer (Dai et al., 2020)	$\mathcal{O}(N^2)$
Linformer (Wang et al., 2020c)	$\mathcal{O}(N)$
Linear Transformers (Katharopoulos et al., 2020)	$\mathcal{O}(N)$
Big Bird (Zaheer et al., 2020)	$\mathcal{O}(N)$
Random Feature Attention (Peng et al., 2021)	$\mathcal{O}(N)$
Long Short Transformers (Zhu et al., 2021)	$\mathcal{O}(kN)$
Poolingformer (Zhang et al., 2021)	$\mathcal{O}(N)$
Nyströmformer (Xiong et al., 2021b)	$\mathcal{O}(kN)$
Perceiver (Jaegle et al., 2021)	$\mathcal{O}(kN)$
Clusterformer (Wang et al., 2020b)	$\mathcal{O}(N \log N)$
Luna (Ma et al., 2021)	$\mathcal{O}(kN)$
TokenLearner (Ryoo et al., 2021)	$\mathcal{O}(k^2)$
Adaptive Sparse Transformer (Correia et al., 2019)	$\mathcal{O}(N^2)$
Product Key Memory (Lample et al., 2019)	$\mathcal{O}(N^2)$
Switch Transformer (Fedus et al., 2021)	$\mathcal{O}(N^2)$
ST-MoE (Zoph et al., 2022)	$\mathcal{O}(N^2)$
GShard (Lepikhin et al., 2020)	$\mathcal{O}(N^2)$
Scaling Transformers (Jaszczur et al., 2021)	$\mathcal{O}(N^2)$
GLaM (Du et al., 2021)	$\mathcal{O}(N^2)$

Perceiver

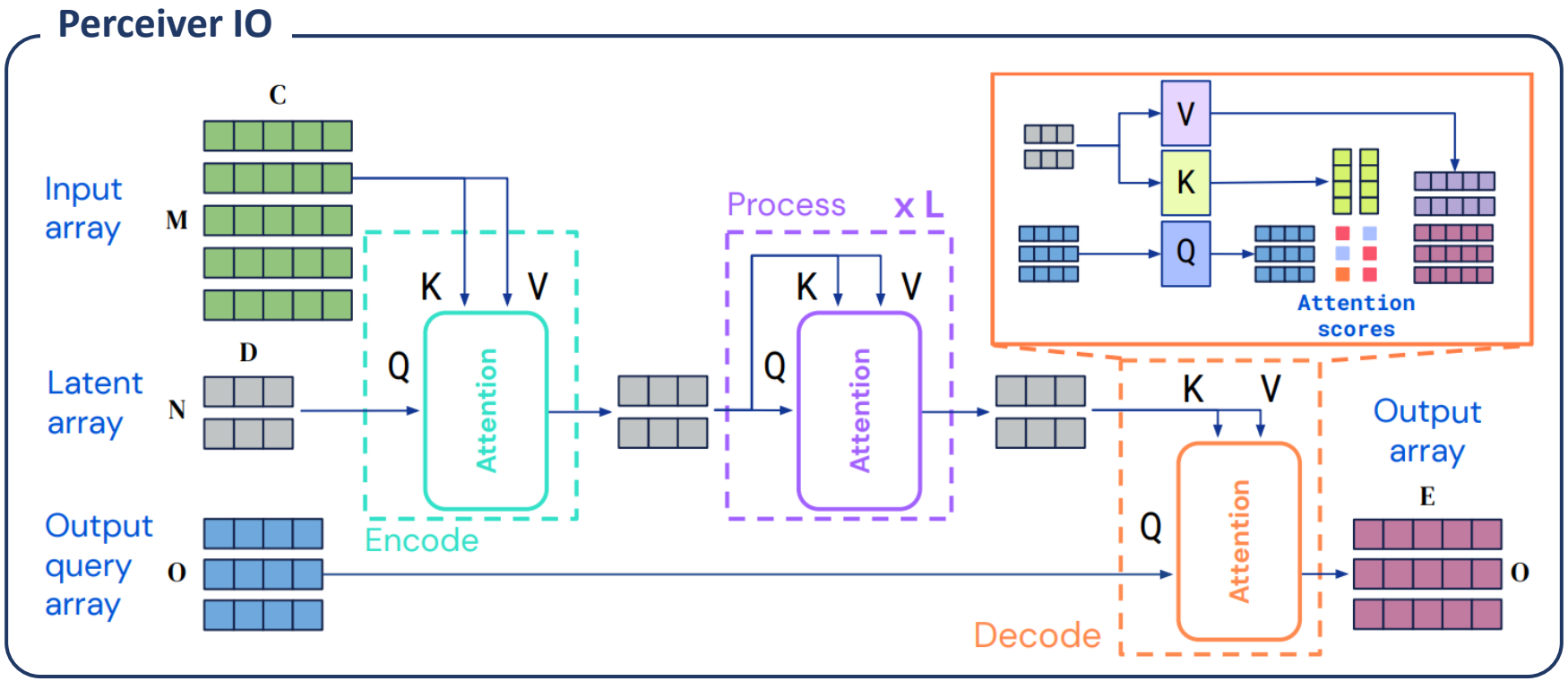
Easy to implement example of **efficient transformer**, intended as general-purpose architecture

https://arxiv.org/abs/2107.14795

Note: Number of Query inputs does not have to match number of Keys/Values:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{\overbrace{QK^T}^{N \times M}}{\sqrt{d_k}})V \Rightarrow N \times D$$

$N \times D$ $M \times D$



See https://keras.io/examples/vision/perceiver_image_classification/ for examplary code in tensorflow/keras