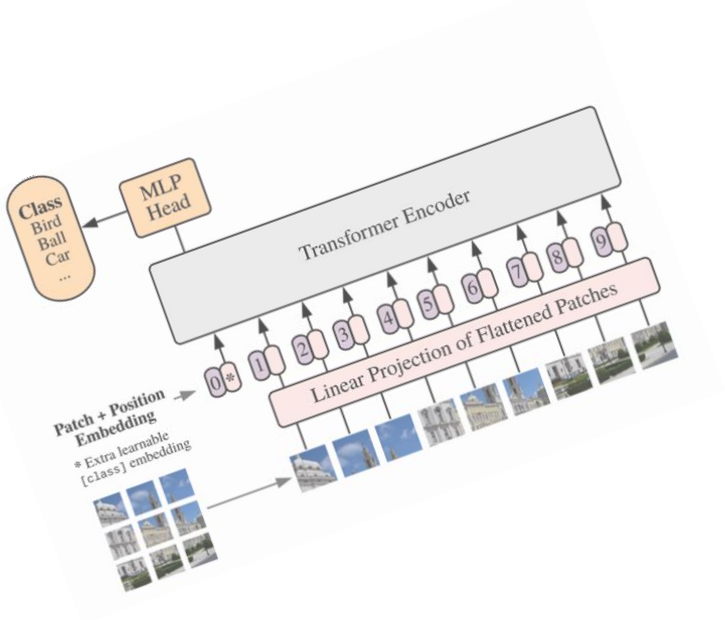
SPONSORED BY THE

Federal Ministry of Education and Research

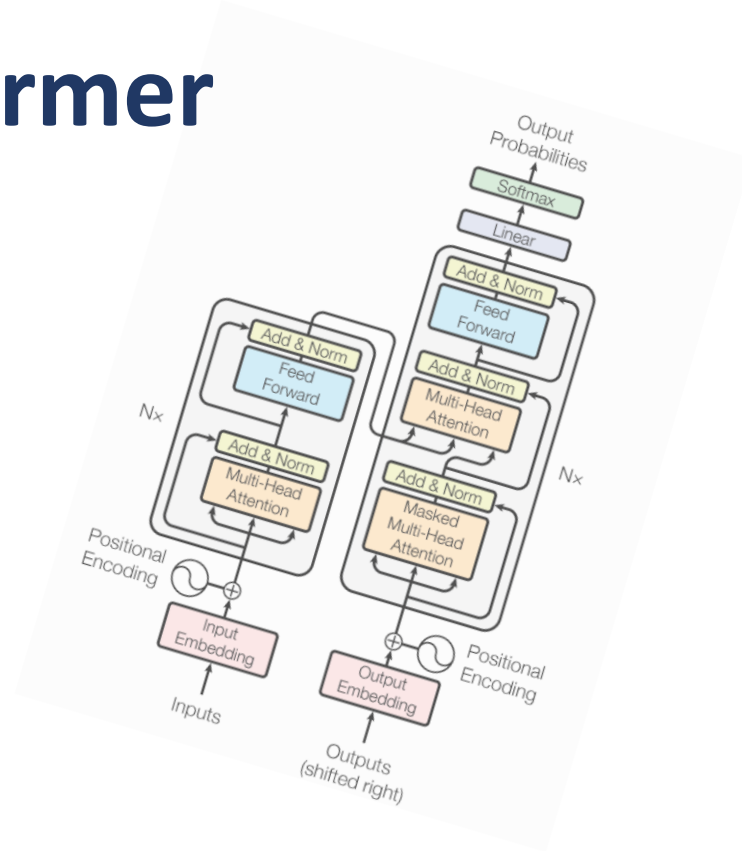
III. Physikalisches Institut A | RWTH AACHEN UNIVERSITY

Hands-On Session: Transformer



Active Training Course "Advanced Deep Learning"

ERUM DATA HUB

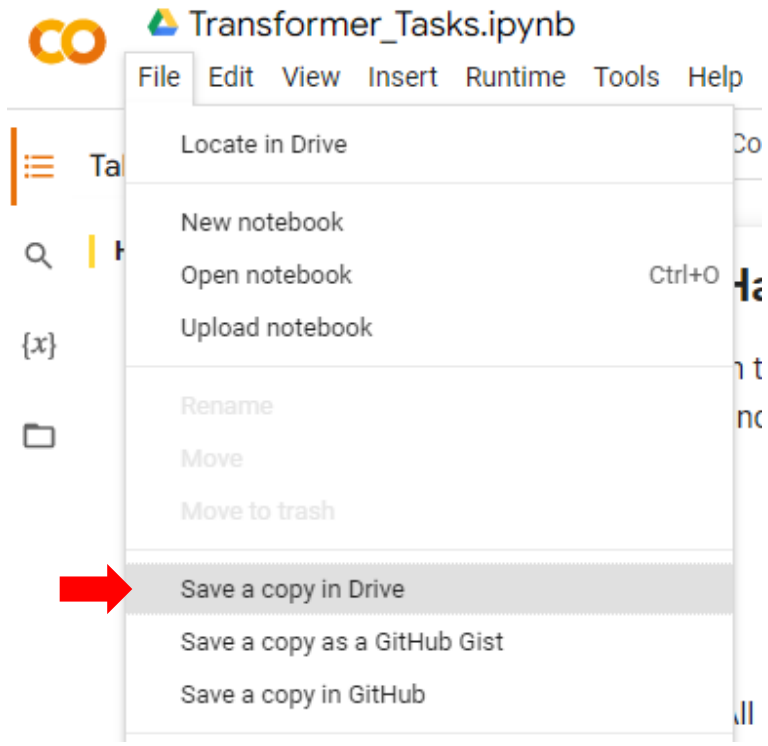


Niklas Langner, Nathan Prouvost

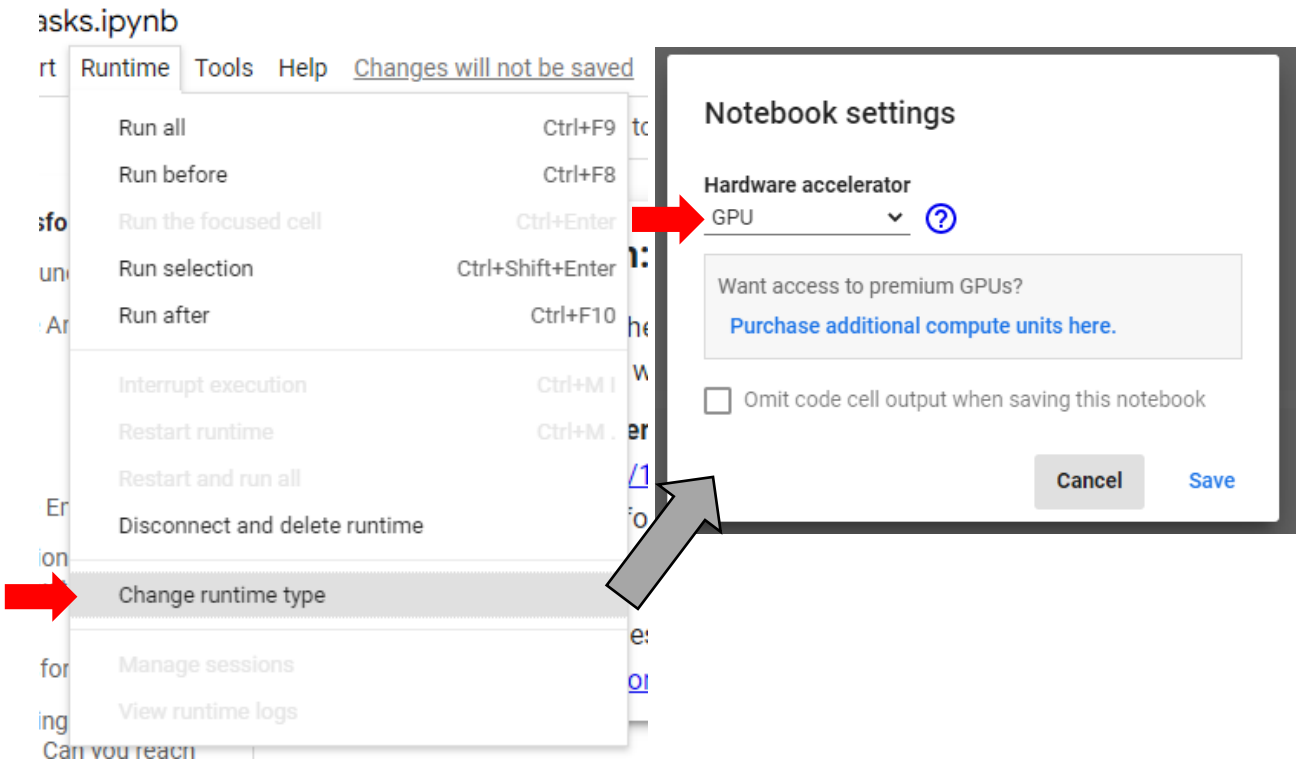
Google Colab Notebook

The exercises can be found in the Colab notebook:
https://colab.research.google.com/drive/1zVtxObz6W_cIHkitCWFhxIscOqcW6tyW?usp=sharing

Download and use locally or
copy to Drive (requires Google account):

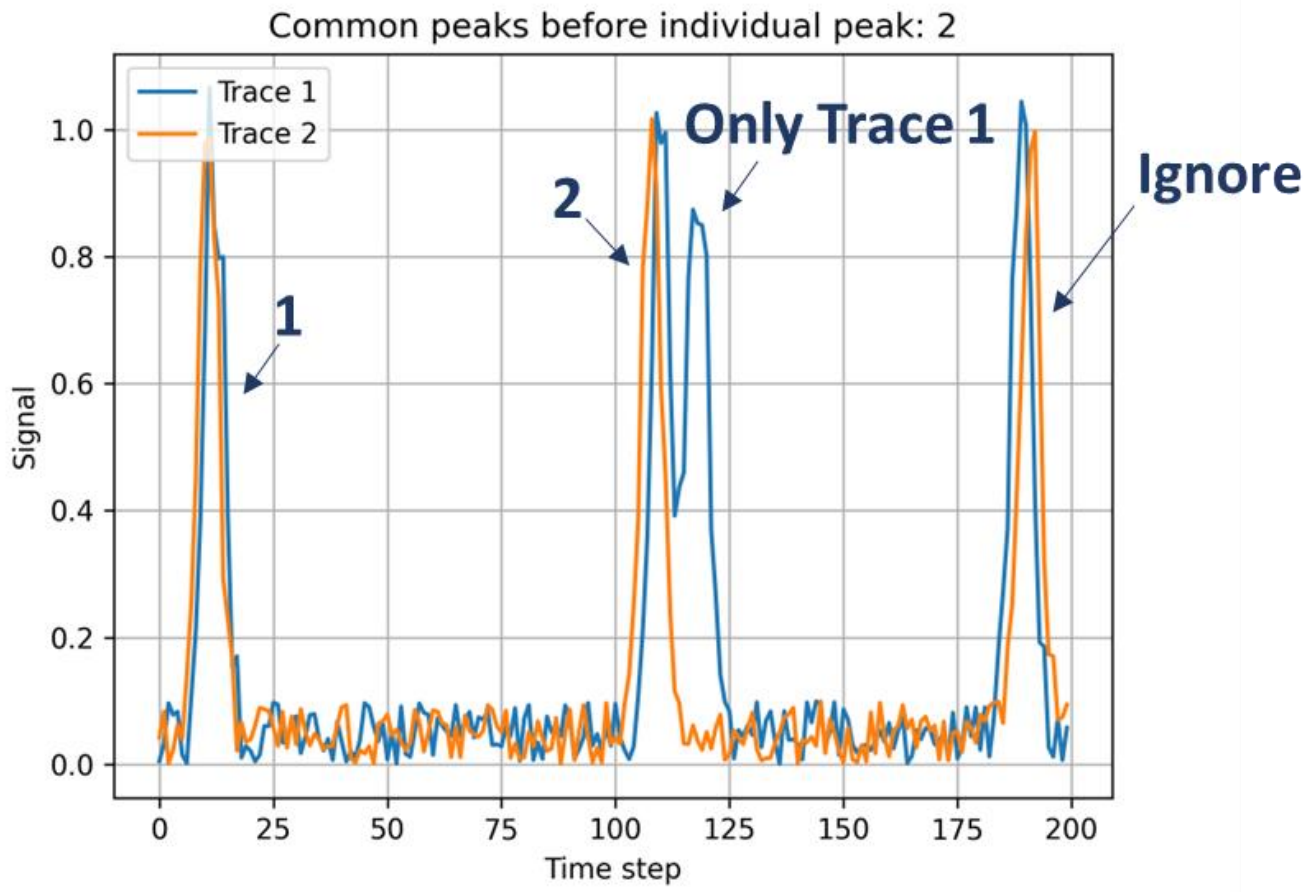


Make sure it runs on GPU (if available)

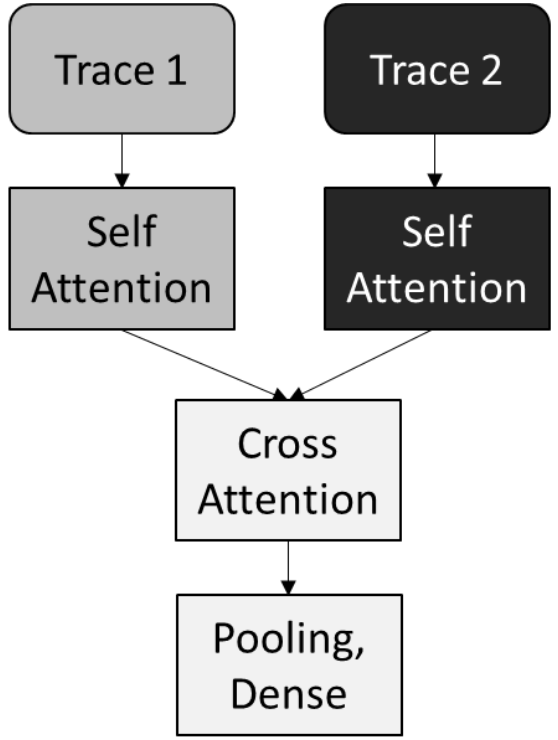


Exercise 1: Signal Trace Analysis

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



Transformer Ansatz:

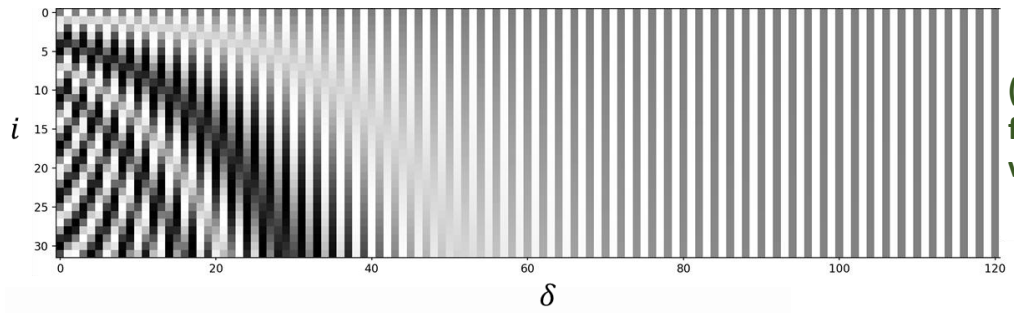


Exercise 1: Signal Trace Analysis

Implement necessary components of Transformer:

1. Embedding
2. Positional Encoding
3. Transformer Block

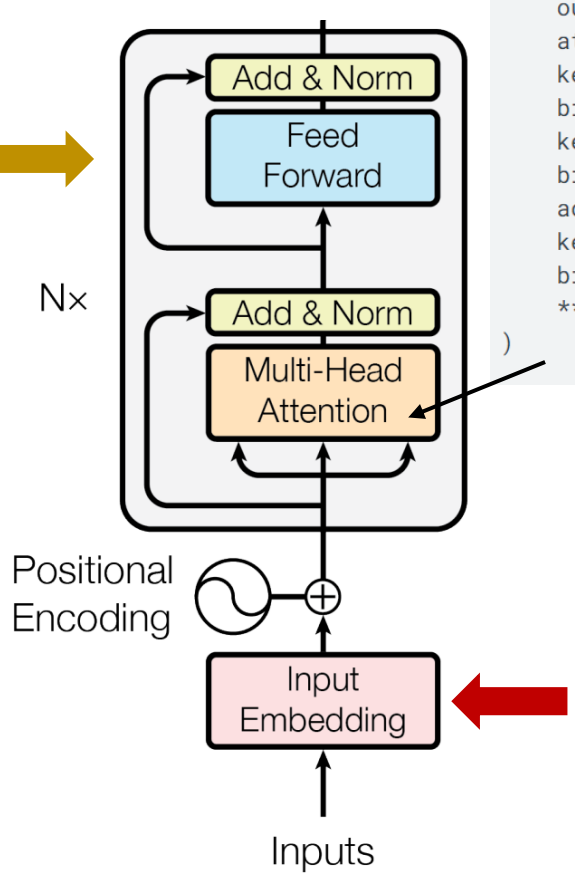
Non-Trainable (Example)



(or any other fixed set of vectors)

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

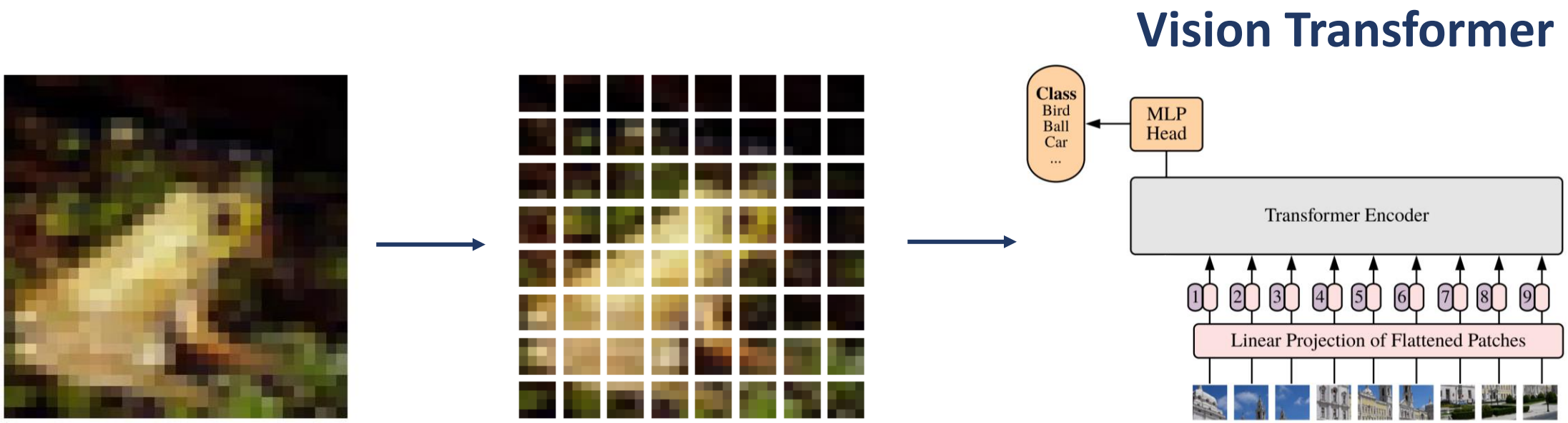
i: Time bin
δ: vector dimension
d: network dims.




```
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
)
```

Call arguments
query
value
key

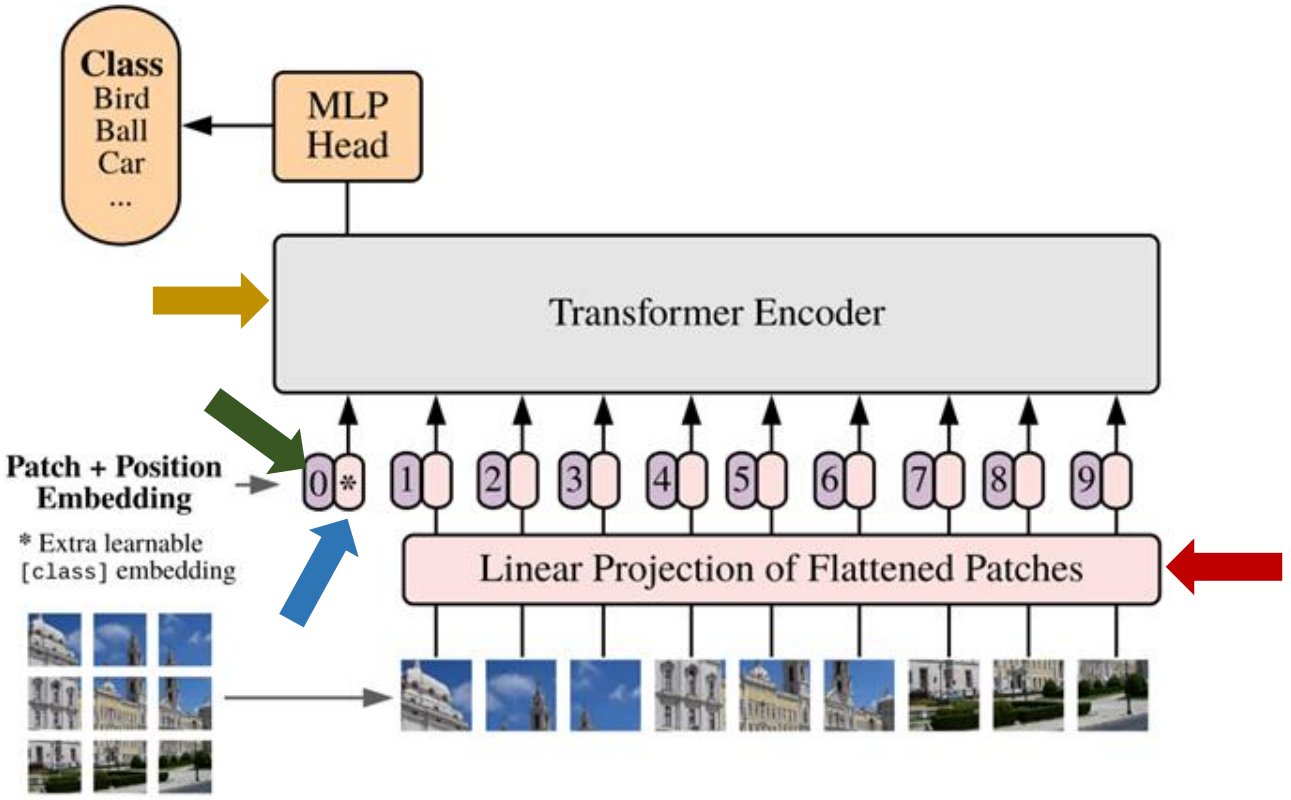
Exercise 2: Image Classification with Vision Transformer



Exercise 2: Image Classification with Vision Transformer

Implement necessary components of Vision Transformer:

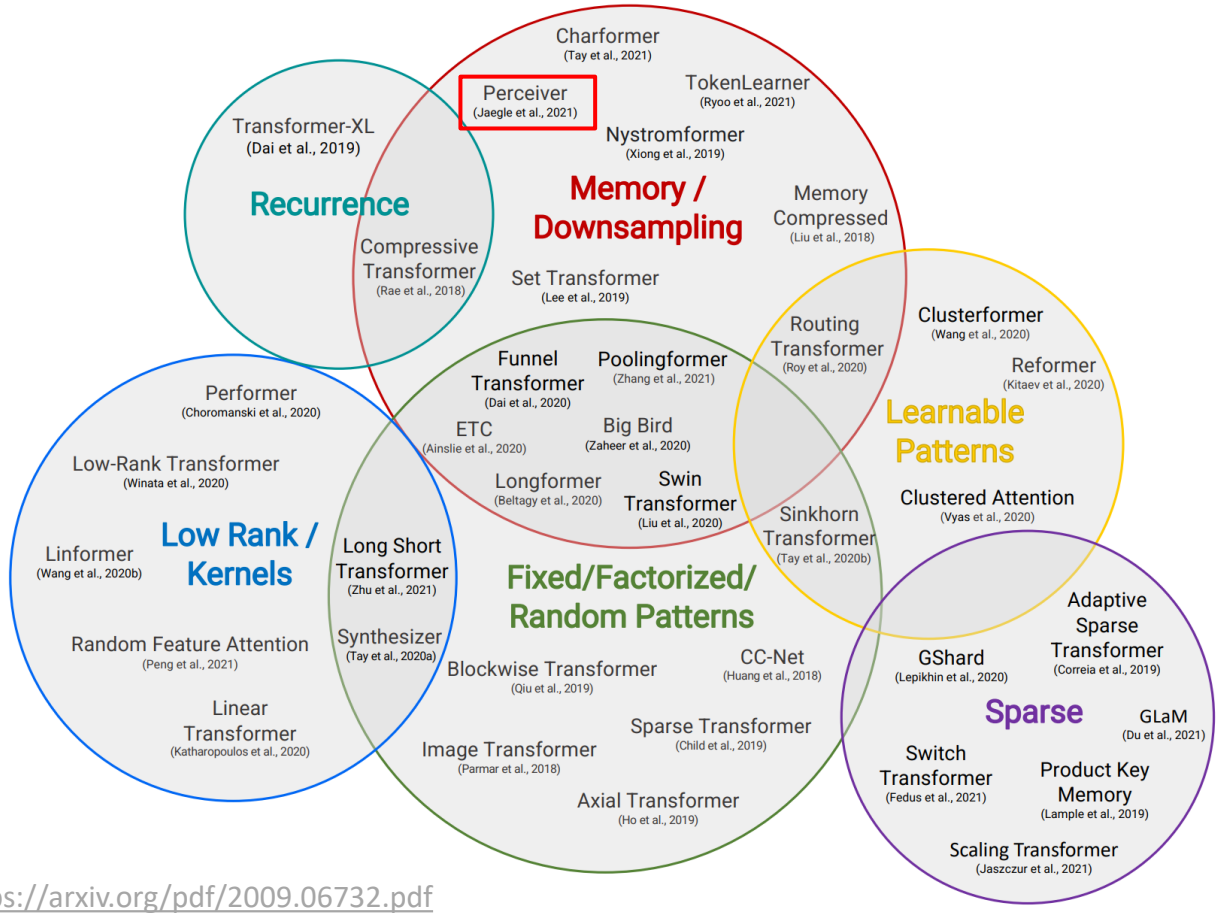
1. Patch Projection
2. Positional Encoding
3. Architecture
4. (Bonus: Class Token)



Exercise 3: Perceiver

Increasing trace length in Exercise 1 leads 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)	$\mathcal{O}(N^2)$

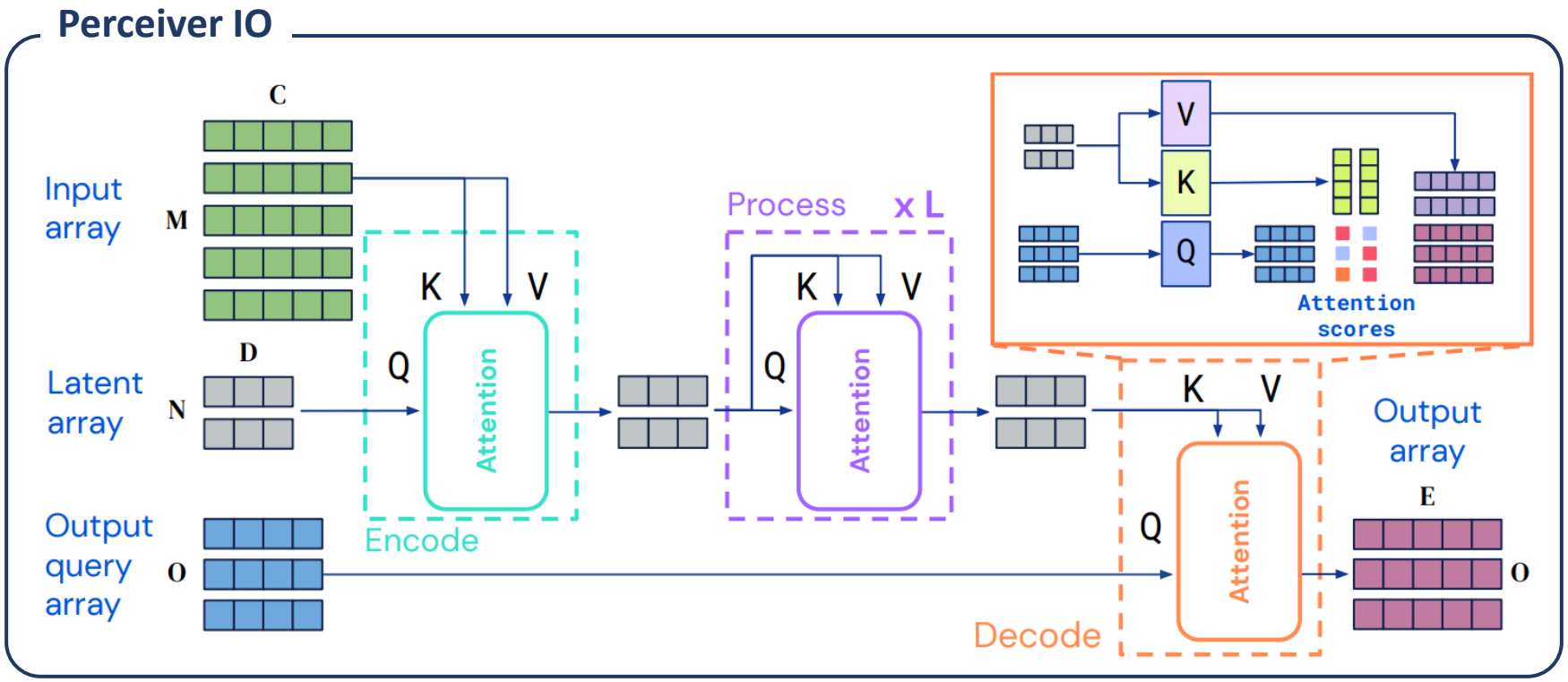
Exercise 3: Perceiver

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

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

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

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



Task: Find suited dimensionality of latent array