CS-503
Visual Intelligence: Machines and Minds

Roman Bachmann

27.02.2025

École
polytechnique
fédérale
de Lausanne

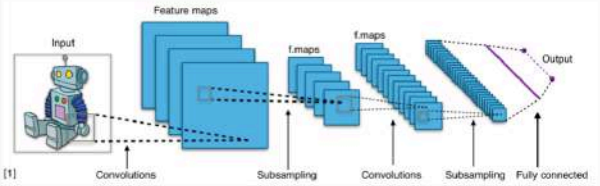
SotA lecture: Transformers

A story of unification

The **classic landscape**: One architecture per “community”

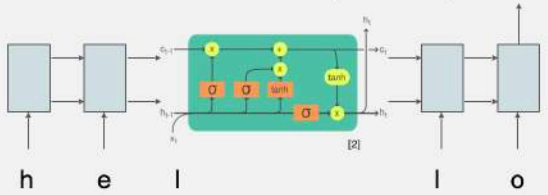
Computer Vision

Convolutional NNs (+ResNets)



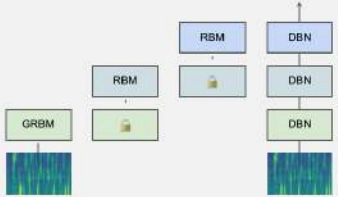
Natural Lang. Proc.

Recurrent NNs (+LSTMs)



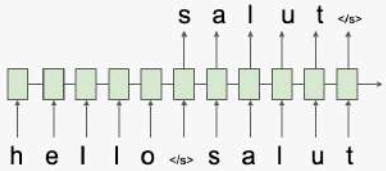
Speech

Deep Belief Nets (+non-DL)



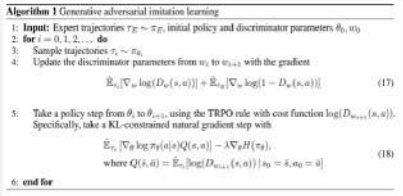
Translation

Seq2Seq



RL

BC/GAIL



Algorithm 1 Generative adversarial imitation learning

1: **Input:** Expert trajectories $\tau_E \sim \pi_E$, initial policy and discriminator parameters θ_0, w_0
2: **for** $i = 0, 1, 2, \ldots$ **do**
3: Sample trajectories $\tau_i \sim \pi_{\theta_i}$
4: Update the discriminator parameters from w_i to w_{i+1} with the gradient

$$\hat{\mathbb{E}}_{\tau_i}[\nabla_w \log(D_w(s,a))] + \hat{\mathbb{E}}_{\tau_E}[\nabla_w \log(1 - D_w(s,a))] \quad (17)$$

5: Take a policy step from θ_i to θ_{i+1}, using the TRPO rule with cost function $\log(D_{w_{i+1}}(s,a))$. Specifically, take a KL-constrained natural gradient step with

$$\hat{\mathbb{E}}_{\tau_i}[\nabla_\theta \log \pi_\theta(a|s) Q(s,a)] - \lambda \nabla_\theta H(\pi_\theta), \quad \text{where } Q(\bar{s},\bar{a}) = \hat{\mathbb{E}}_{\tau_i}[\log(D_{w_{i+1}}(s,a)) \mid s_0 = \bar{s}, a_0 = \bar{a}] \quad (18)$$

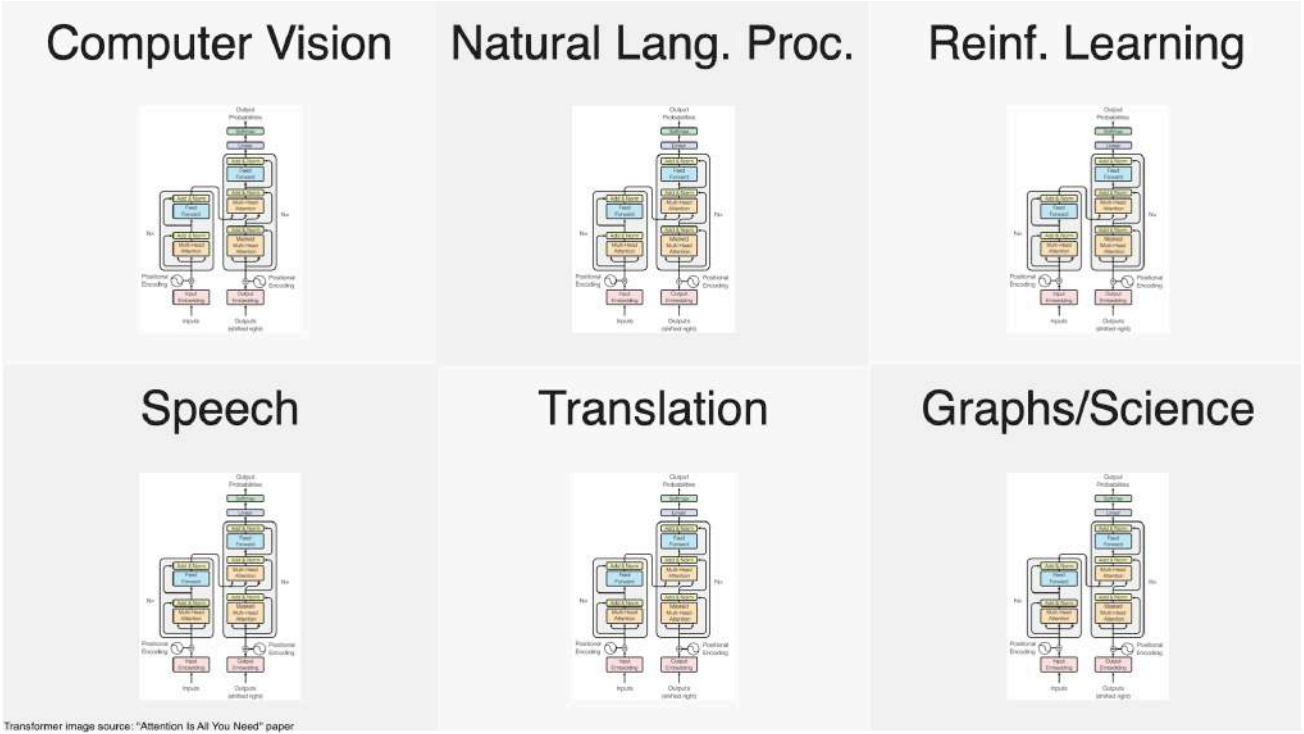
6: **end for**

[1] CNN image CC-BY-SA by Aphex34 for Wikipedia https://commons.wikimedia.org/wiki/File:Typical_cnn.png
[2] RNN image CC-BY-SA by GChe for Wikipedia https://commons.wikimedia.org/wiki/File:The_LSTM_Cell.svg

[Slide credit: Lucas Beyer]

A story of unification

The **Transformer's takeover**: One community at a time



[Slide credit: Lucas Beyer]

Transformers in vision

Leaderboards for image classification on ImageNet1K val:

Rank	Model	Top 1 Accuracy	Top 5 Accuracy	Number of params	GFLOPs	Extra Training Data	Paper
1	BASIC-L (Lion, fine-tuned)	91.1%		2440M		✓	Symbolic Discovery of Optimization Algorithms
2	CoCa (finetuned)	91.0%		2100M		✓	CoCa: Contrastive Captioners are Image-Text Foundation Models
3	Model soups (BASIC-L)	90.98%		2440M		✓	Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time
4	Model soups (ViT-G/14)	90.94%		1843M		✓	Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time
5	ViT-e	90.9%		3900M		✓	PaLI: A Jointly-Scaled Multilingual Language-Image Model
6	CoAtNet-7	90.88%		2440M	2586	✓	CoAtNet: Marrying Convolution and Attention for All Data Sizes
7	ViT-G/14 (Lion)	90.71%		1843M		✓	Symbolic Discovery of Optimization Algorithms
8	CoCa (frozen)	90.60%		2100M		✓	CoCa: Contrastive Captioners are Image-Text Foundation Models
9	CoAtNet-6	90.45%		1470M	1521	✓	CoAtNet: Marrying Convolution and Attention for All Data Sizes
10	ViT-G/14	90.45%		1843M	2859.9	✓	Scaling Vision Transformers

Papers with Code (08.03.23)

Transformers in vision

Leaderboards for semantic segmentation on ADE20K val:

Rank	Model	Validation↑ mIoU	Test Score	Params (M)	GFLOPs (512 x 512)	Extra Training Data	Paper
1	**InternImage-H** (M3I Pre-training)	62.9		1310		✓	InternImage: Exploring Large-Scale Vision Foundation Models with Deformable Convolutions
2	**M3I Pre-training** (InternImage-H)	62.9		1310		✓	Towards All-in-one Pre-training via Maximizing Multi-modal Mutual Information
3	**BEiT-3**	62.8		1900		✓	Image as a Foreign Language: BEiT Pretraining for All Vision and Vision-Language Tasks
4	**EVA**	62.3		1074		✓	EVA: Exploring the Limits of Masked Visual Representation Learning at Scale
5	**ViT-Adapter-L** (Mask2Former, BEiTv2 pretrain)	61.5		571		✓	Vision Transformer Adapter for Dense Predictions
6	**FD-SwinV2-G**	61.4				✓	Contrastive Learning Rivals Masked Image Modeling in Fine-tuning via Feature Distillation
7	**RevCol-H** (Mask2Former)	61.0				✓	Reversible Column Networks
8	**MasK DINO** (SwinL, multi-scale)	60.8		223		✓	Mask DINO: Towards A Unified Transformer-based Framework for Object Detection and Segmentation
9	**ViT-Adapter-L** (Mask2Former, BEiT pretrain)	60.5		571		✓	Vision Transformer Adapter for Dense Predictions
10	**SwinV2-G** (UperNet)	59.9				✓	Swin Transformer V2: Scaling Up Capacity and Resolution

Papers with Code (08.03.23)

Transformers in vision

Leaderboards for semantic segmentation on ADE20K val:

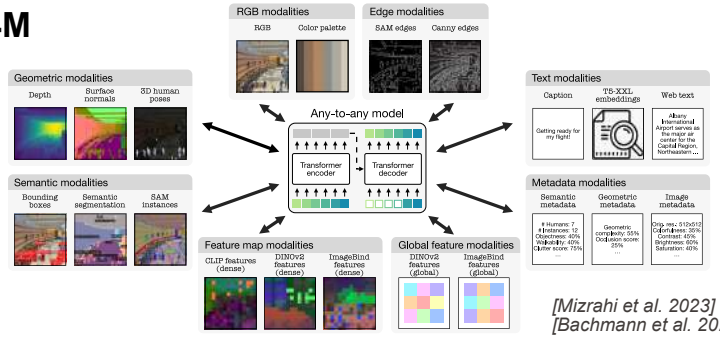
... in 2025

Papers with Code (25.02.2025)

Transformers in vision

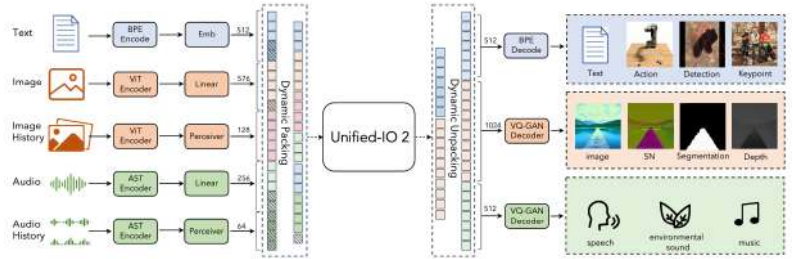
After Nov. 2022 "ChatGPT moment":
Gradual shift from task-specific models to **unified foundation models**

4M



[Mizrahi et al. 2023]
[Bachmann et al. 2024]

Unified-IO



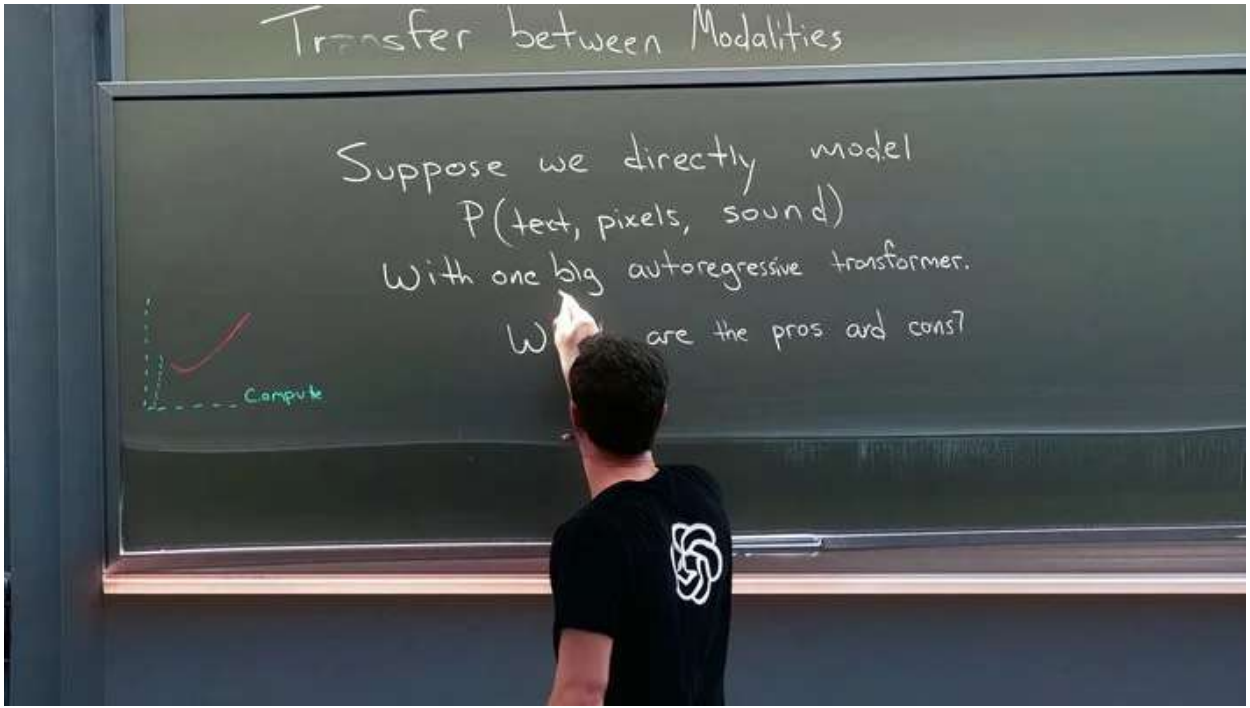
[Lu et al. 2022, 2023]

Any-to-any LLMs

Gemini
GPT-4o
Chameleon
Janus-Pro

Transformers in vision

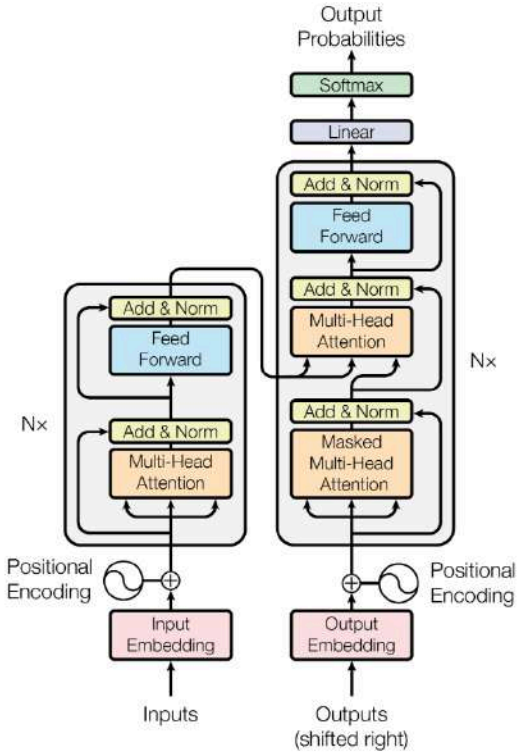
After Nov. 2022 "ChatGPT moment":
Gradual shift from task-specific models to **unified foundation models**



[GPT-4o generated image posted by Greg Brockman. 16.05.2024. https://x.com/gdb/status/1790869434174746805]

Transformers in vision

Since *[Attention Is All You Need, Vaswani et al., 2017]*, the core architecture has *largely* remained the same.



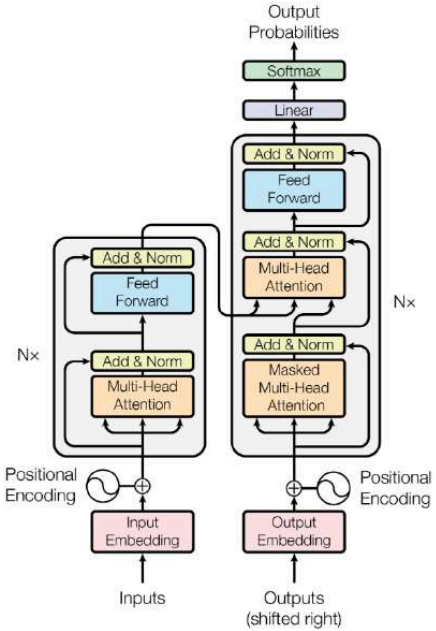
[Vaswani et al. 2017]

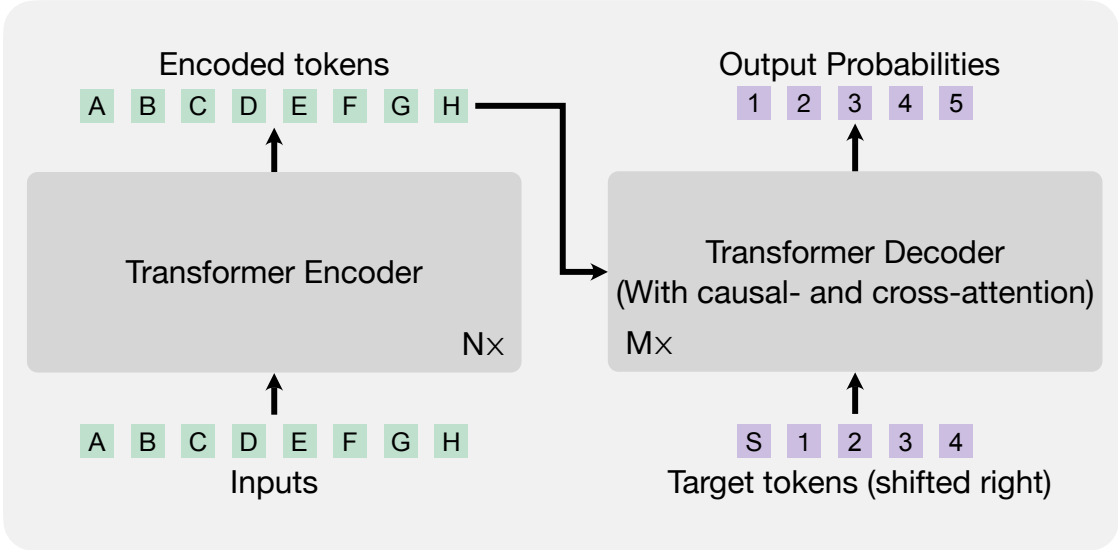
Overview

Attention Is All You Need [Vaswani et al. 2017]

Transformer I/O and overview

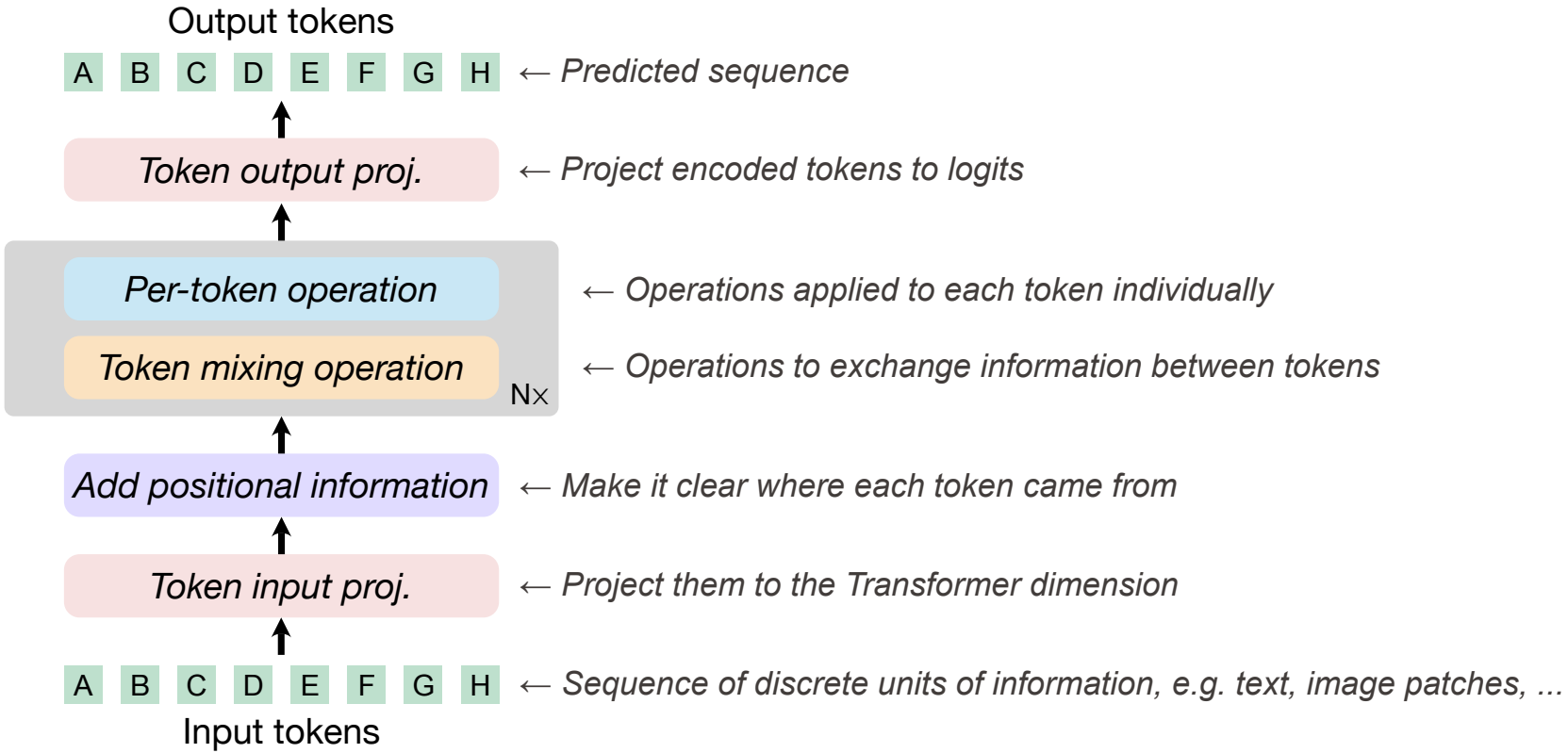
Generic sequence to sequence architecture





Common Transformer types

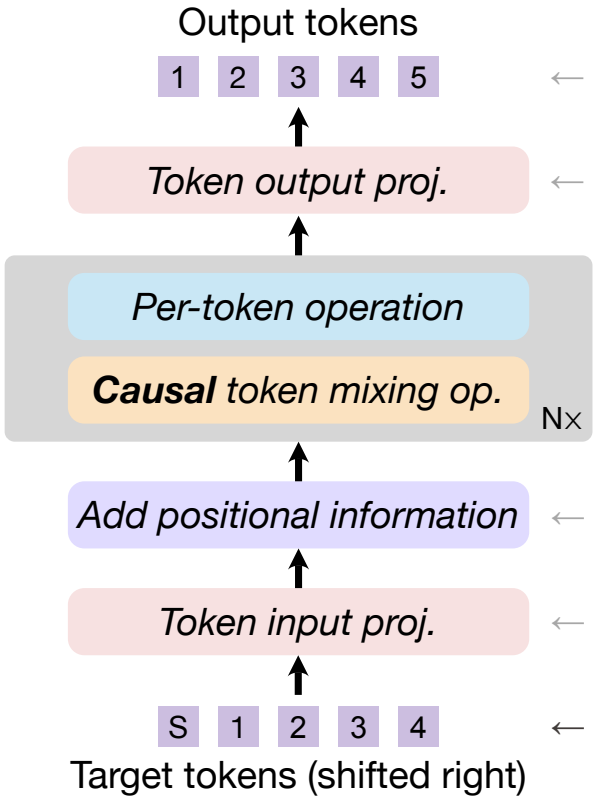
"Encoder-only" Transformer



Common Transformer types

"Decoder-only" Transformer

(Also: "GPT", "Causal Transformer", "Autoregressive Transformer", "Teacher forcing")



← *Predicted sequence*

← *Project encoded tokens to logits*

← *Operations applied to each token individually*

← *Operations to exchange information between tokens in a **causal manner***

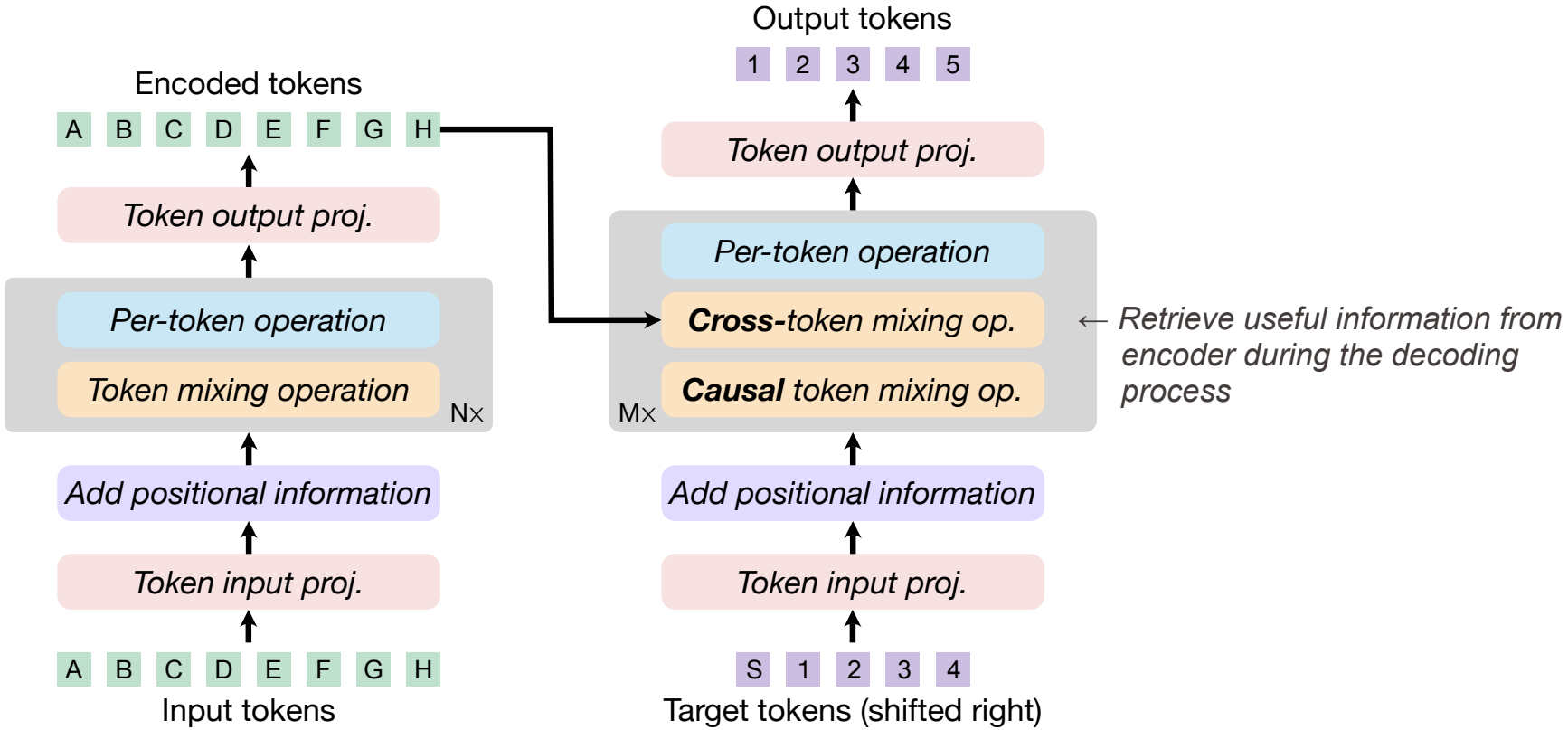
← *Make it clear where each token came from*

← *Project them to the Transformer dimension*

← *Sequence of discrete units of information, e.g. text, image patches, ... Shifted right for **next-token prediction**.*

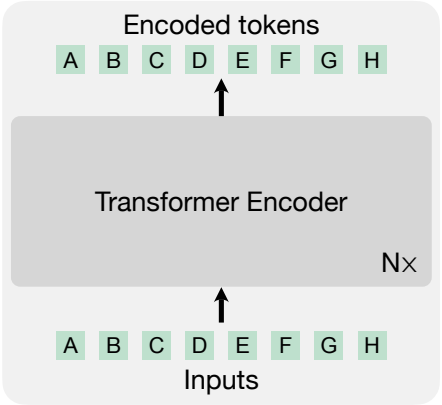
Common Transformer types

"Encoder-Decoder" Transformer



Common Transformer types

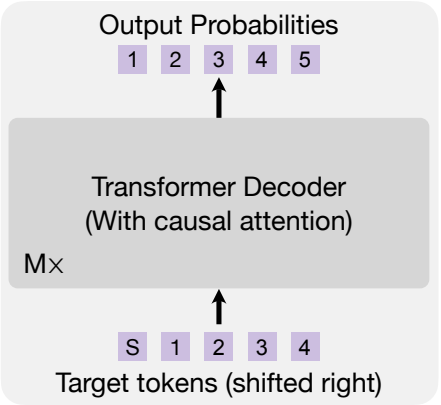
“Encoder-only”



Encoding & masking

(E.g. ViT, BERT, MaskGIT, etc...)

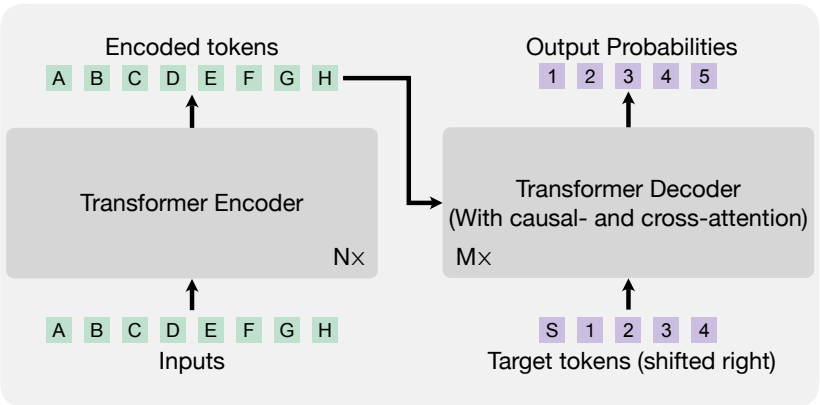
“Decoder-only”



Next-token prediction

(E.g. GPT, LLamaGen, etc...)

“Encoder-decoder”



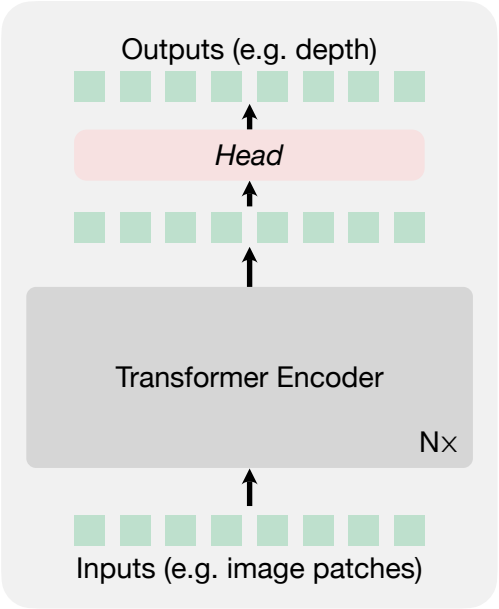
Sequence-to-sequence

(E.g. T5, 4M, Unified-IO, etc...)

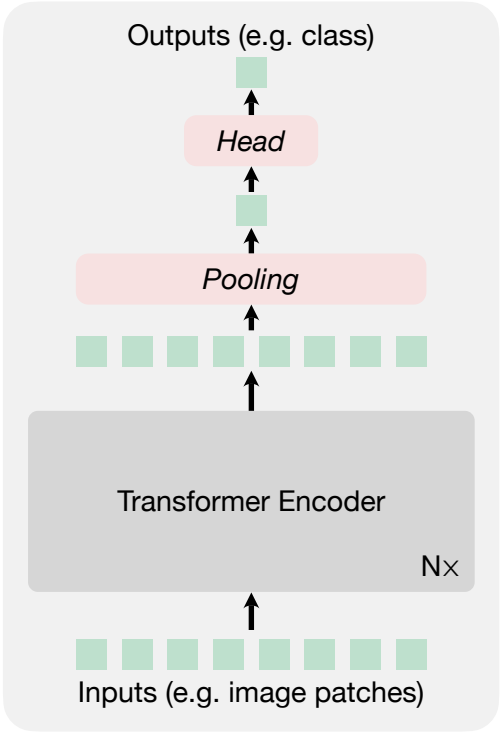
Common Transformer training schemes

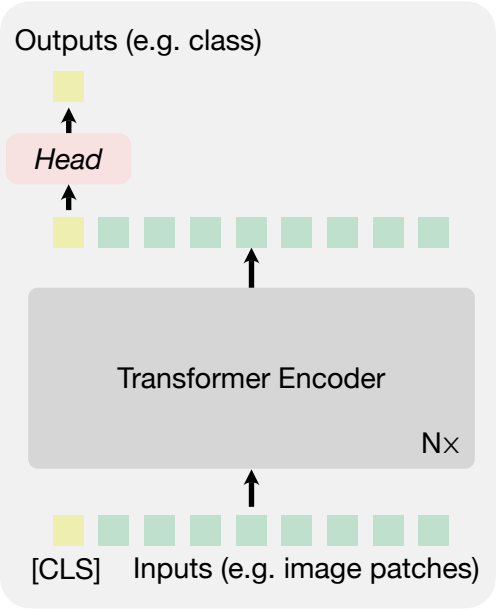
As a vision backbone...

... for dense prediction tasks



... for classification, SSL, etc.

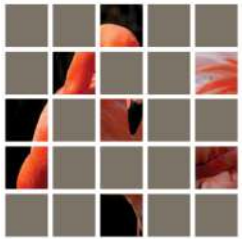


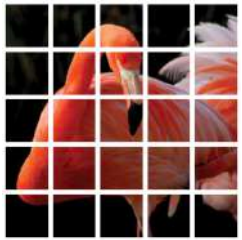


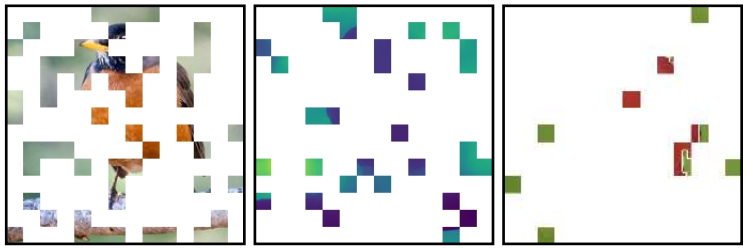
Common Transformer training schemes

Masked training

"I love drinking [MASK] tea when it's hot outside." → "iced"





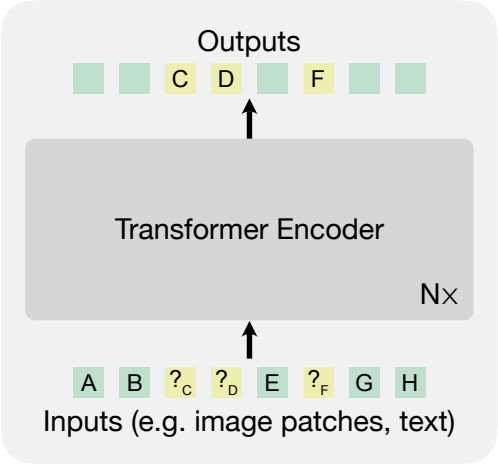


Bachmann

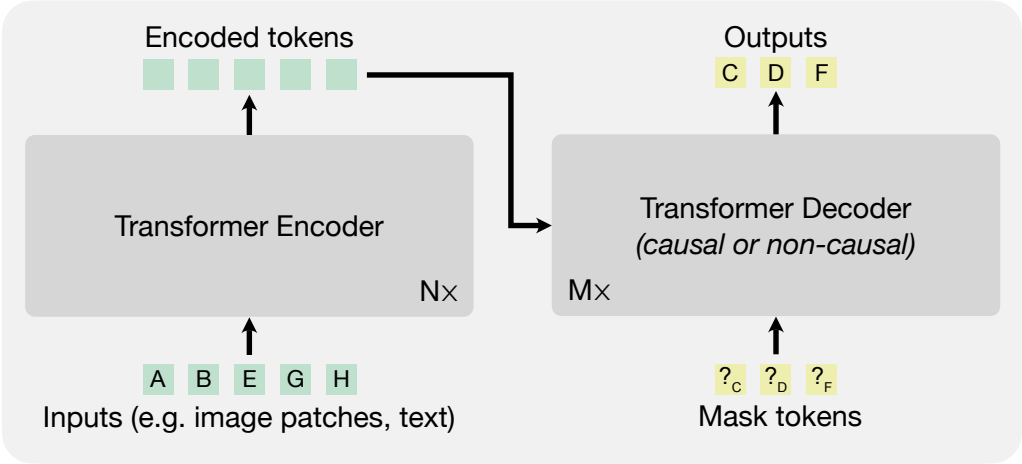
Common Transformer training schemes

Masked training

Masked modeling with "Encoder-only" Transformer



Masked modeling with "Encoder-Decoder" Transformer



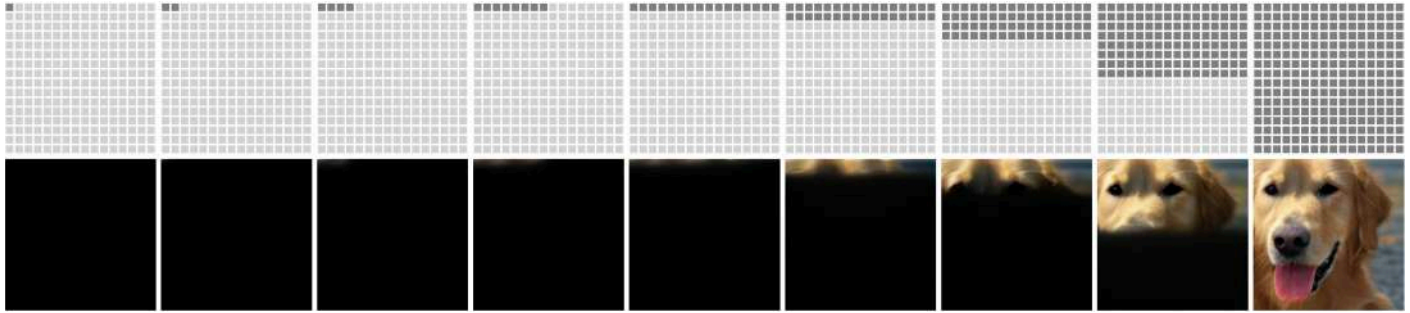
? = mask token (learnable placeholder token)

Common Transformer training schemes

Autoregressive training / next token prediction

Task	Example sentence in pre-training that would teach that task
Grammar	In my free time, I like to {code, banana}
Lexical semantics	I went to the store to buy papaya, dragon fruit, and {durian, squirrel}
World knowledge	The capital of Azerbaijan is {Baku, London}
Sentiment analysis	Movie review: I was engaged and on the edge of my seat the whole time. The movie was {good, bad}
Translation	The word for "pretty" in Spanish is {bonita, hola}
Spatial reasoning	Iroh went into the kitchen to make tea. Standing next to Iroh, Zuko pondered his destiny. Zuko left the {kitchen, store}
Math question	Arithmetic exam answer key: 3 + 8 + 4 = {15, 11}
[millions more]	Extreme multi-task learning!

[Stanford CS25: V4 I Jason Wei & Hyung Won Chung of OpenAI]



Common Transformer training schemes

Autoregressive training / next token prediction

We model $p(x_{i+1} \mid x_1, \ldots, x_i)$ with a single model.

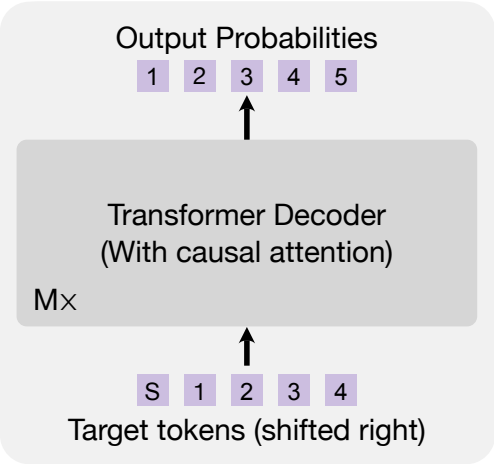
""	→	"I"
"I"	→	"love"
"I love"	→	"drinking"
"I love drinking"	→	"iced"
"I love drinking iced"	→	"tea"
"I love drinking iced tea"	→	"when"
"I love drinking iced tea when"	→	"it's"
"I love drinking iced tea when it's"	→	"hot"
"I love drinking iced tea when it's hot"	→	"outside"

Bachmann

Common Transformer training schemes

Autoregressive training / next token prediction

We model $p(x_{i+1} \mid x_1, \ldots, x_i)$ with a single model.

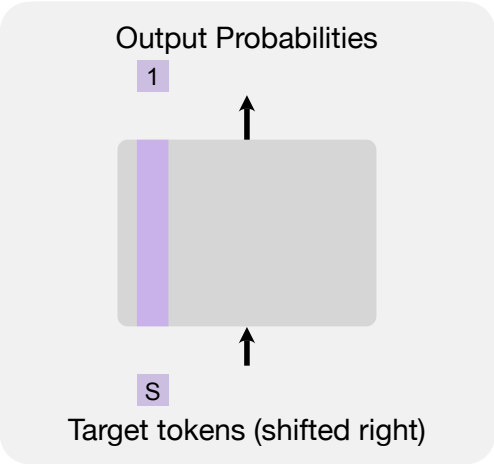


""	→	"I"
"I"	→	"love"
"I love"	→	"drinking"
"I love drinking"	→	"iced"
"I love drinking iced"	→	"tea"
"I love drinking iced tea"	→	"when"
"I love drinking iced tea when"	→	"it's"
"I love drinking iced tea when it's"	→	"hot"
"I love drinking iced tea when it's hot"	→	"outside"

Common Transformer training schemes

Autoregressive training / next token prediction

We model $p(x_{i+1} \mid x_1, \ldots, x_i)$ with a single model.

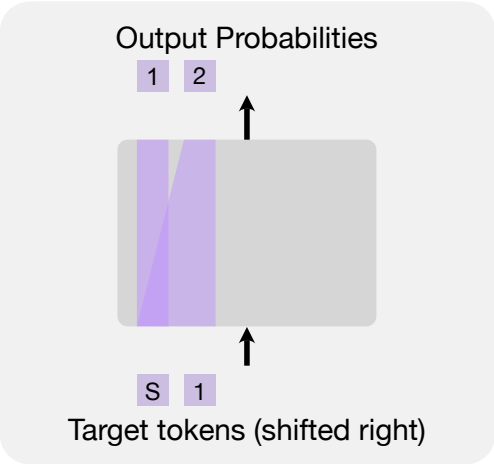


""	→	"I"
"I"	→	"love"
"I love"	→	"drinking"
"I love drinking"	→	"iced"
"I love drinking iced"	→	"tea"
"I love drinking iced tea"	→	"when"
"I love drinking iced tea when"	→	"it's"
"I love drinking iced tea when it's"	→	"hot"
"I love drinking iced tea when it's hot"	→	"outside"

Common Transformer training schemes

Autoregressive training / next token prediction

We model $p(x_{i+1} \mid x_1, \ldots, x_i)$ with a single model.

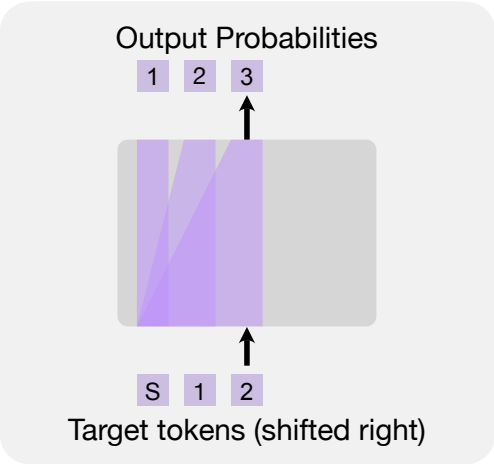


""	→	"I"
"I"	→	"love"
"I love"	→	"drinking"
"I love drinking"	→	"iced"
"I love drinking iced"	→	"tea"
"I love drinking iced tea"	→	"when"
"I love drinking iced tea when"	→	"it's"
"I love drinking iced tea when it's"	→	"hot"
"I love drinking iced tea when it's hot"	→	"outside"

Common Transformer training schemes

Autoregressive training / next token prediction

We model $p(x_{i+1} \mid x_1, \ldots, x_i)$ with a single model.

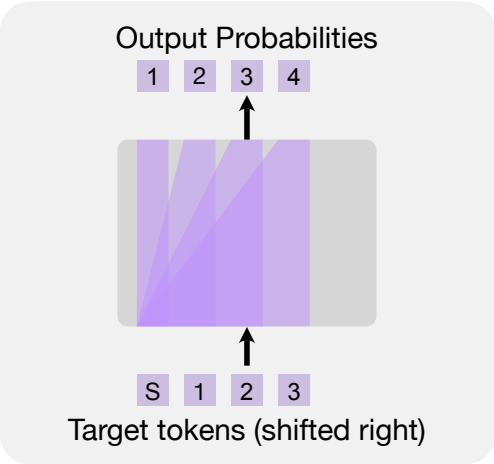


""	→	"I"
"I"	→	"love"
"I love"	→	"drinking"
"I love drinking"	→	"iced"
"I love drinking iced"	→	"tea"
"I love drinking iced tea"	→	"when"
"I love drinking iced tea when"	→	"it's"
"I love drinking iced tea when it's"	→	"hot"
"I love drinking iced tea when it's hot"	→	"outside"

Common Transformer training schemes

Autoregressive training / next token prediction

We model $p(x_{i+1} \mid x_1, \ldots, x_i)$ with a single model.

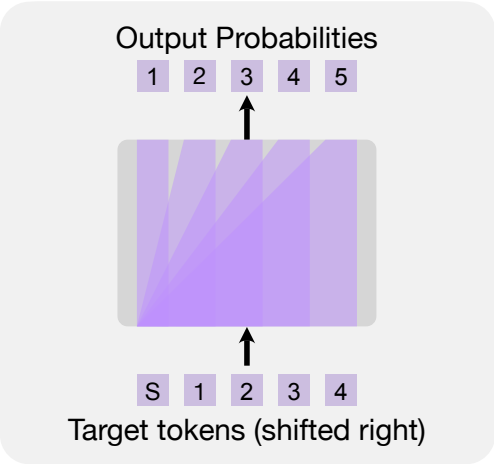


""	→	"I"
"I"	→	"love"
"I love"	→	"drinking"
"I love drinking"	→	"iced"
"I love drinking iced"	→	"tea"
"I love drinking iced tea"	→	"when"
"I love drinking iced tea when"	→	"it's"
"I love drinking iced tea when it's"	→	"hot"
"I love drinking iced tea when it's hot"	→	"outside"

Common Transformer training schemes

Autoregressive training / next token prediction

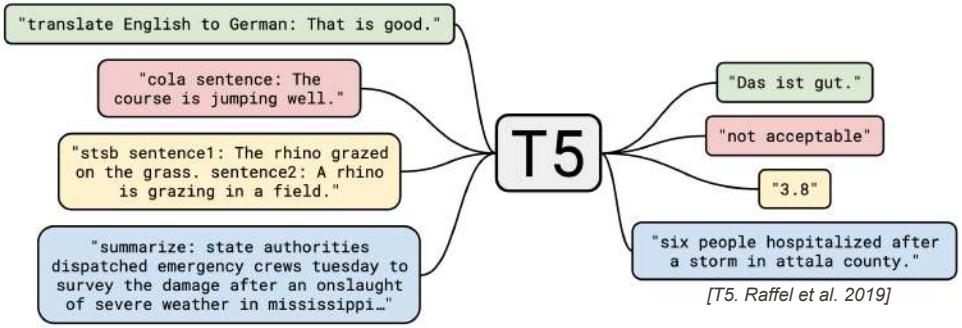
We model $p(x_{i+1} \mid x_1, \ldots, x_i)$ with a single model.



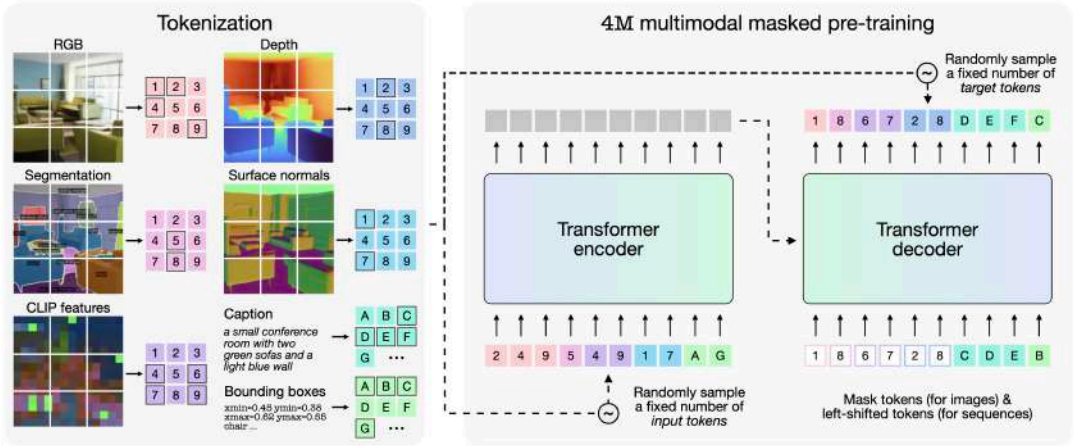
""	→	"I"
"I"	→	"love"
"I love"	→	"drinking"
"I love drinking"	→	"iced"
"I love drinking iced"	→	"tea"
"I love drinking iced tea"	→	"when"
"I love drinking iced tea when"	→	"it's"
"I love drinking iced tea when it's"	→	"hot"
"I love drinking iced tea when it's hot"	→	"outside"

Common Transformer training schemes

Sequence to sequence (seq2seq)



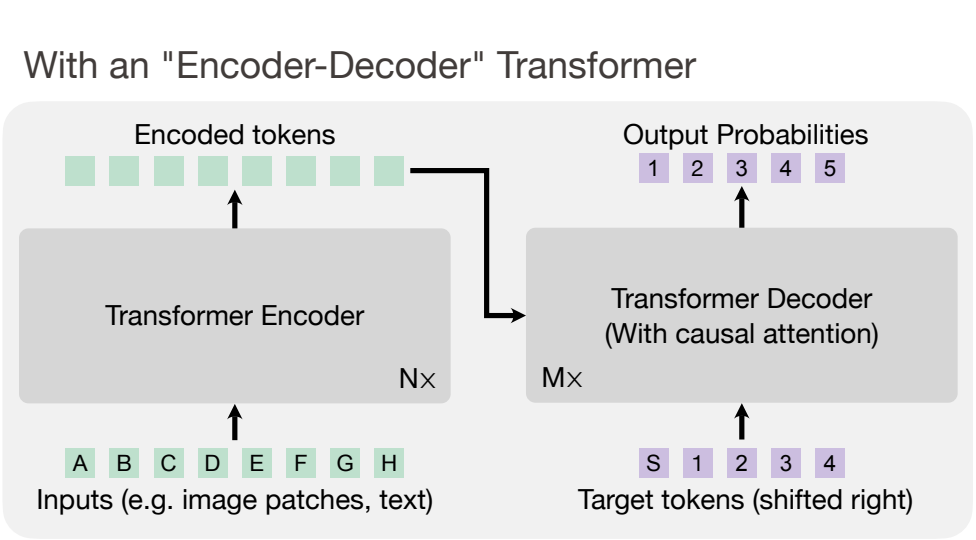
[T5. Raffel et al. 2019]

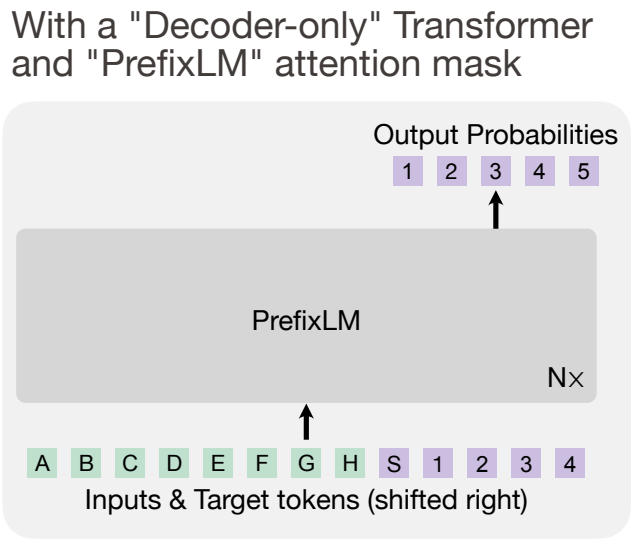


[4M. Mizrahi et al. 2023]

Common Transformer training schemes

Sequence to sequence (seq2seq)

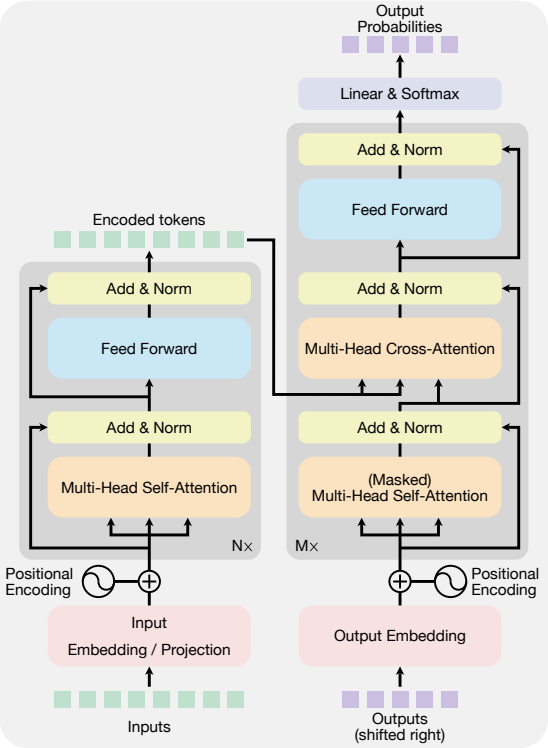




Attention is all you need

Attention Is All You Need [Vaswani et al. 2017]

Transformer I/O and overview

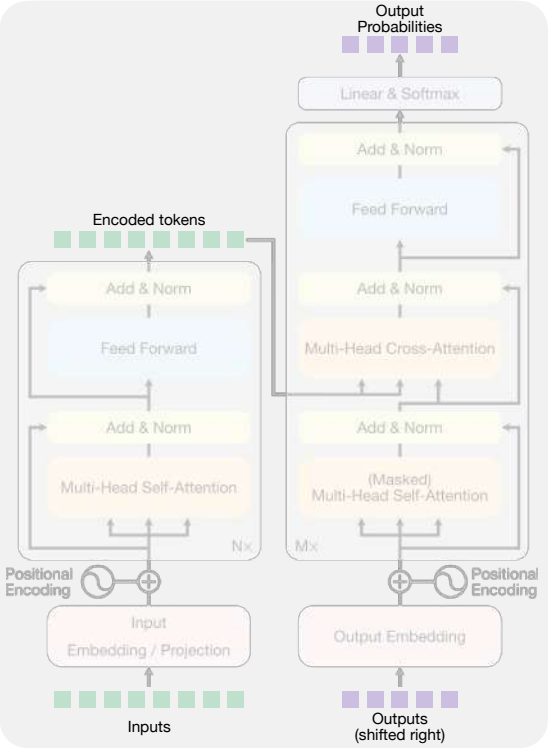


Generic sequence to sequence architecture

[Slides inspiration: Lucas Beyer]

Attention Is All You Need [Vaswani et al. 2017]

Transformer I/O and overview



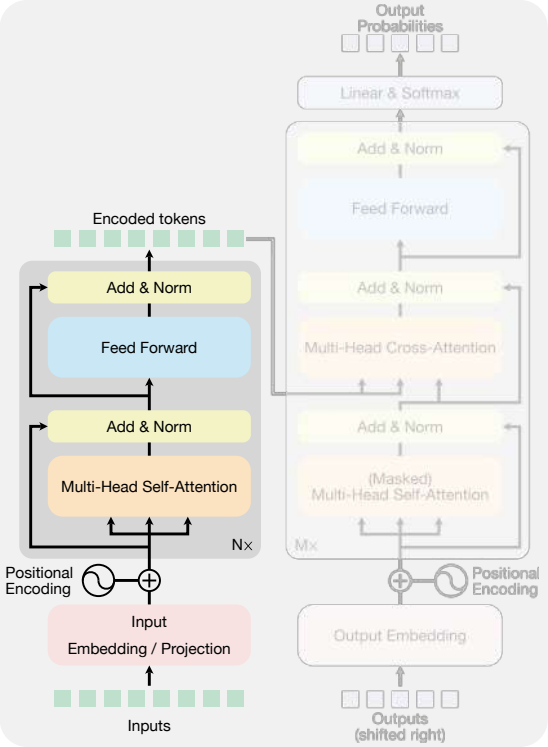
Generic sequence to sequence architecture:

- **Input**: sequence of N_{in} tokens
- **Target**: sequence of N_{out} tokens

[Slides inspiration: Lucas Beyer]

Attention Is All You Need [Vaswani et al. 2017]

Transformer I/O and overview



Generic sequence to sequence architecture:

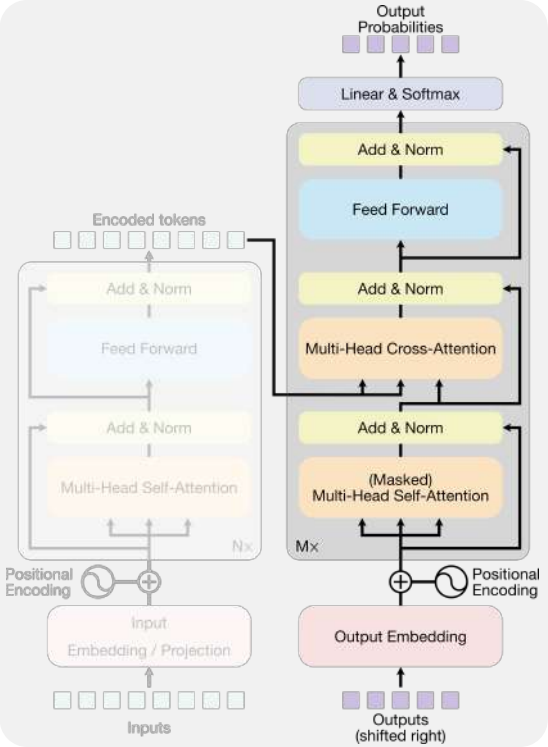
- **Input**: sequence of N_{in} tokens
- **Target**: sequence of N_{out} tokens

Split into Encoder and Decoder:

- **Encoder**: Processes input tokens with alternating self-attention and feed forward layers

Attention Is All You Need [Vaswani et al. 2017]

Transformer I/O and overview



Generic sequence to sequence architecture:

- **Input**: sequence of N_{in} tokens
- **Target**: sequence of N_{out} tokens

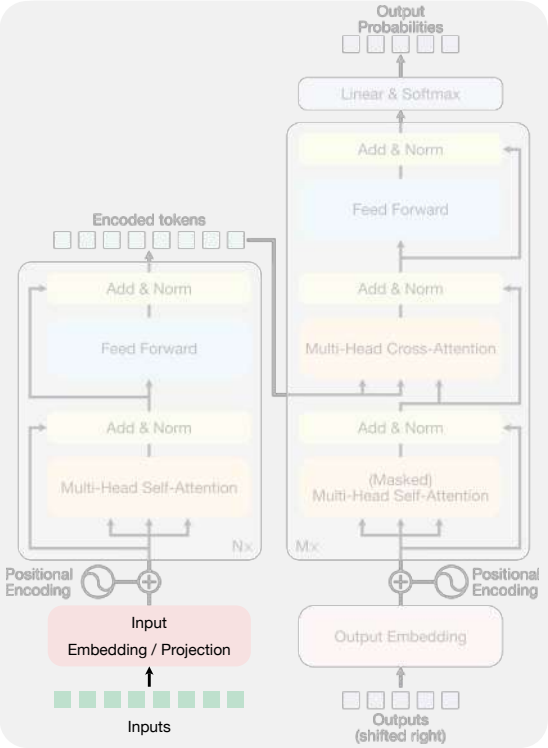
Split into Encoder and Decoder:

- **Encoder**: Processes input tokens with alternating self-attention and feed forward layers
- **Decoder**: Generates outputs, conditioned on encoded tokens (via cross-attention)

NB: We'll also discuss "encoder-only" and "decoder-only" architectures.

Attention Is All You Need [Vaswani et al. 2017]

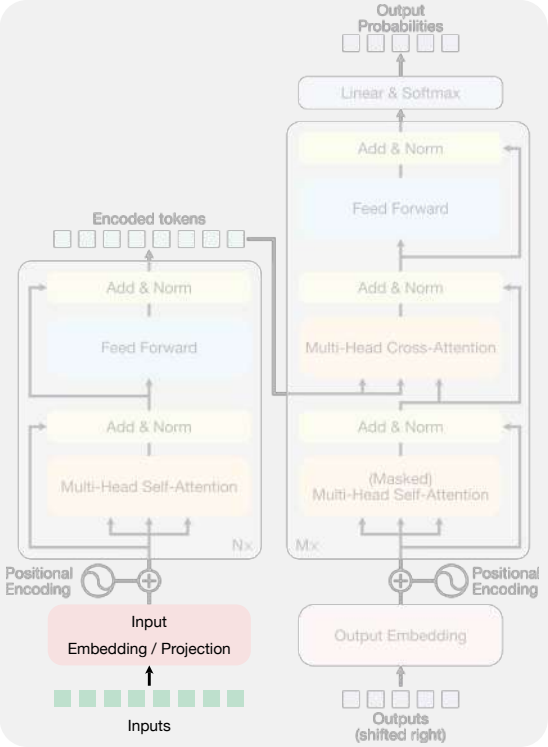
Tokenization and embeddings



- Transformers operate on sequences / sets of tokens
- Each token is a vector of dimension d_{model}
- How to bring data into this format?

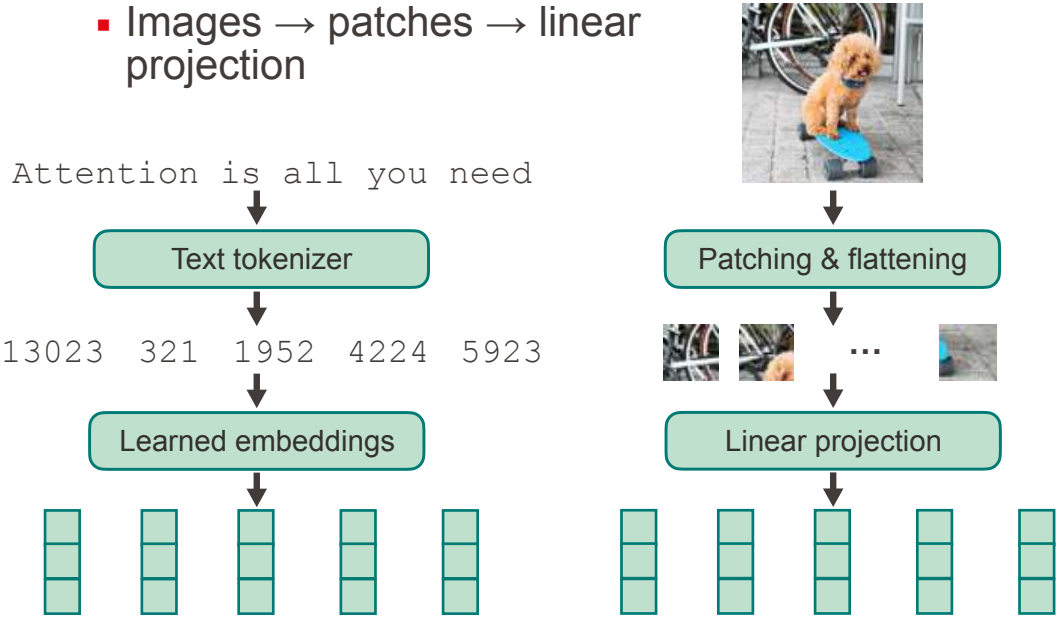
Attention Is All You Need [Vaswani et al. 2017]

Tokenization and embeddings



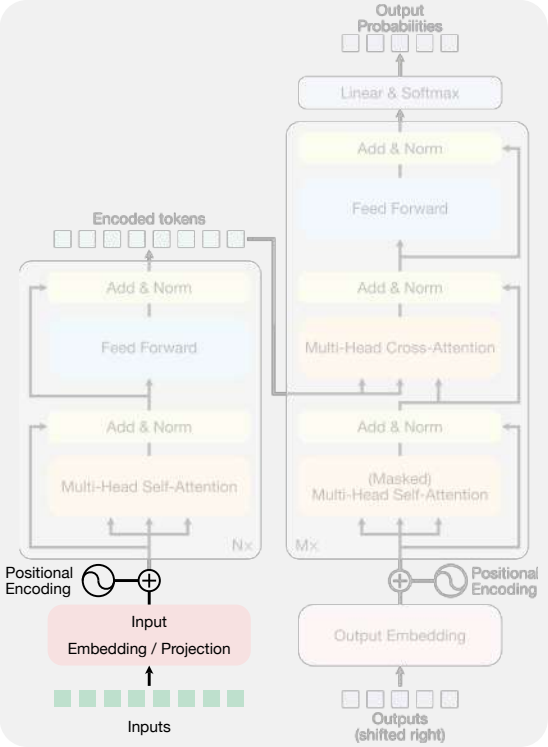
Tokenization depends on modalities used:

- Text → characters / words / sub words → learned embeddings
- Images → patches → linear projection



Attention Is All You Need [Vaswani et al. 2017]

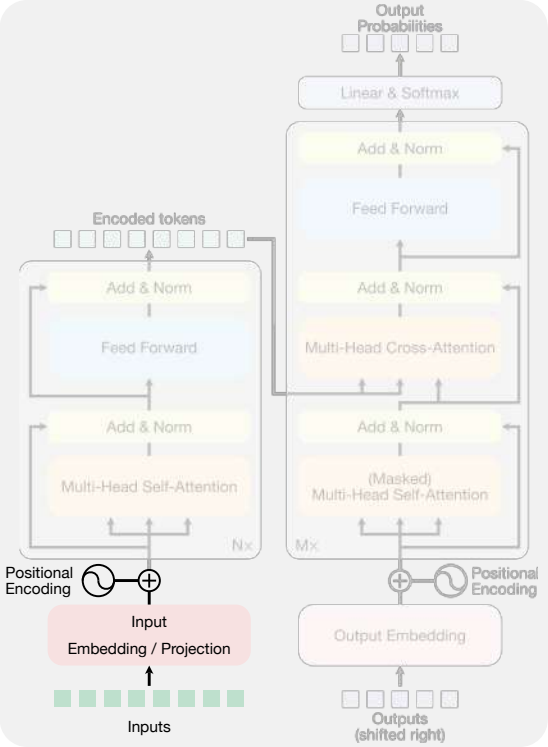
Positional embeddings



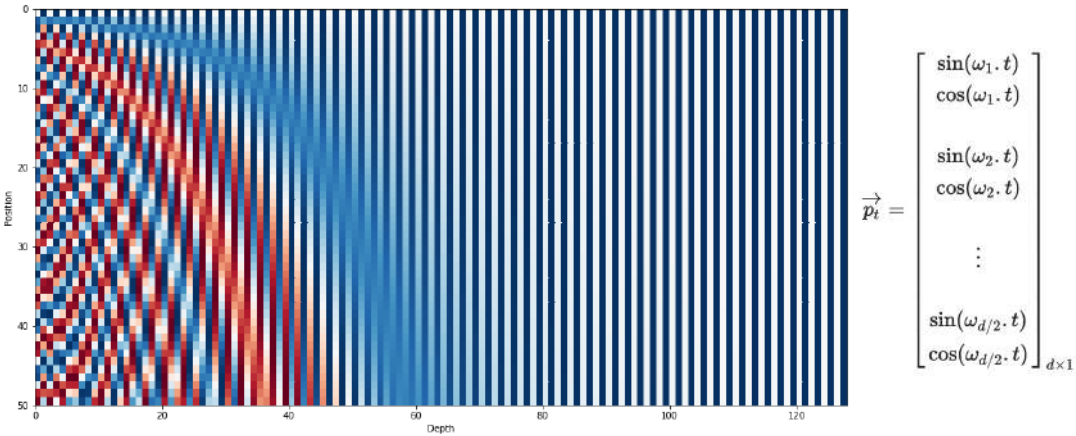
- Attention is permutation invariant, but often our data is not!
- To each token, add unique “positional” encoding vectors
- Learned or hand-engineered (e.g. sinusoidal)
- Format depends on the data:
  - 1D for sequences
  - 2D embeddings for images
  - 3D embeddings for videos
  - Modality-specific embeddings to distinguish sources

Attention Is All You Need [Vaswani et al. 2017]

Positional embeddings



Example: 128-dimensional embedding for maximum sequence length 50. Each row represents the embedding vector p_t.

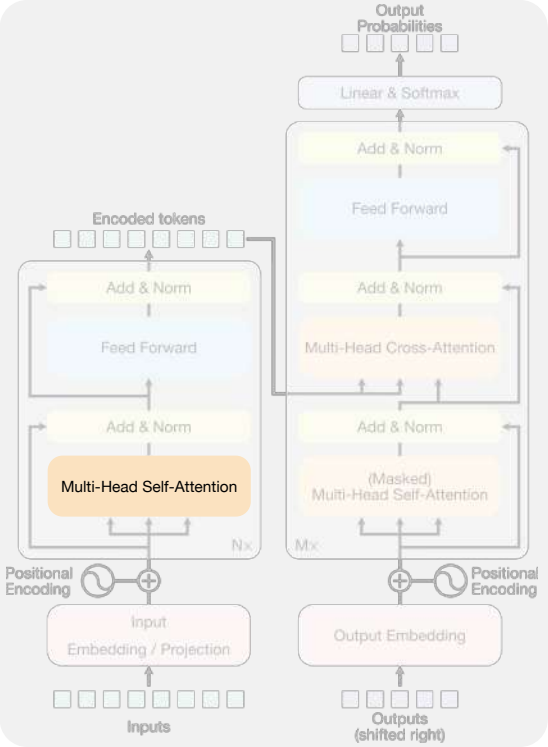


$$\vec{p_t} = \begin{bmatrix} \sin(\omega_1 . t) \\ \cos(\omega_1 . t) \\ \sin(\omega_2 . t) \\ \cos(\omega_2 . t) \\ \vdots \\ \sin(\omega_{d/2} . t) \\ \cos(\omega_{d/2} . t) \end{bmatrix}_{d \times 1}$$

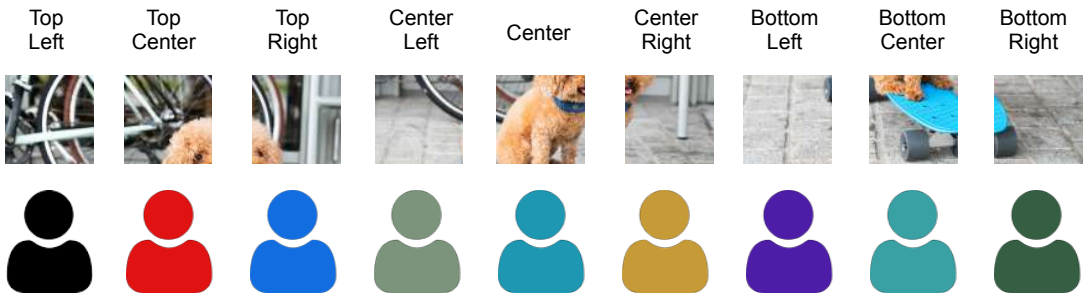
[Transformer Architecture: The Positional Encoding. Kazemnejad, 2019]

Attention Is All You Need [Vaswani et al. 2017]

Attention motivation



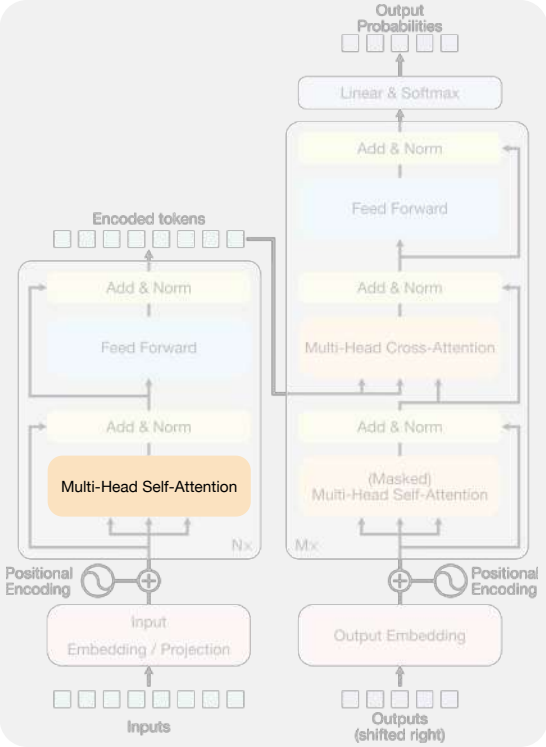
Goal:
Figure out what's in the image



[Thanks @David Mizrahi for the slides!]

Attention Is All You Need [Vaswani et al. 2017]

Attention motivation



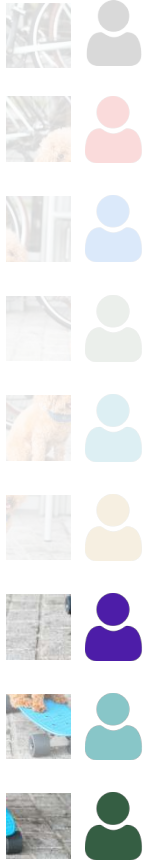
Q: Is there some grey at the bottom?

No clue.
There's some black and white here (but I'm at the top).

Kind of.
There's some blue and some grey.

Yes!
There's grey everywhere.

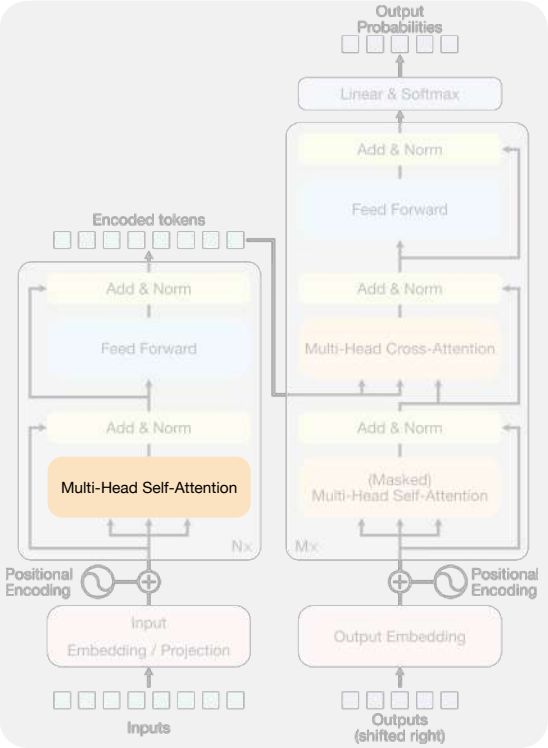
...



[Thanks @David Mizrahi for the slides!]

Attention Is All You Need [Vaswani et al. 2017]

Attention motivation



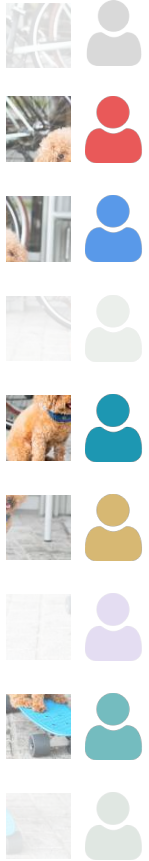
Q: Do you see some fur anywhere?

Yes!
There's fur all over this patch.

Not at all.
I mostly see pavement.

Kind of.
I see a bit of fur on my bottom left.

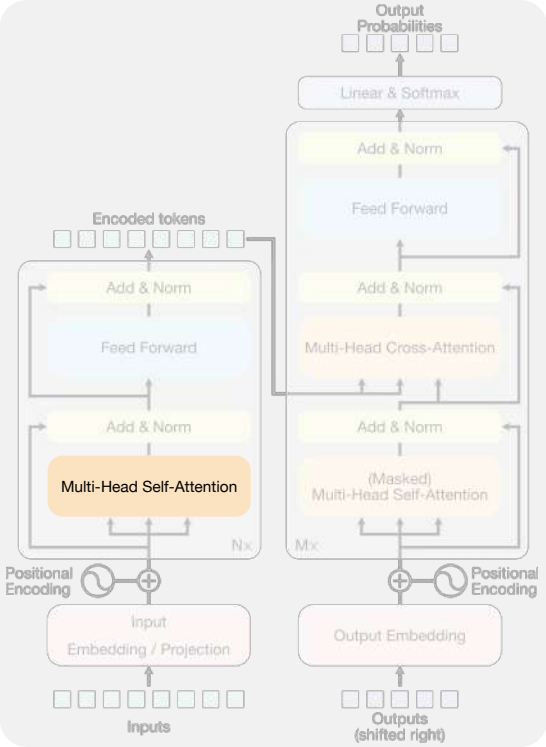
...



[Thanks @David Mizrahi for the slides!]

Attention Is All You Need [Vaswani et al. 2017]

Attention motivation



Q: Do you know what's on my left?

No clue.
I see some eyes and ears (but I'm not on your left).

I do!
I see a skateboard and some fluffy legs.

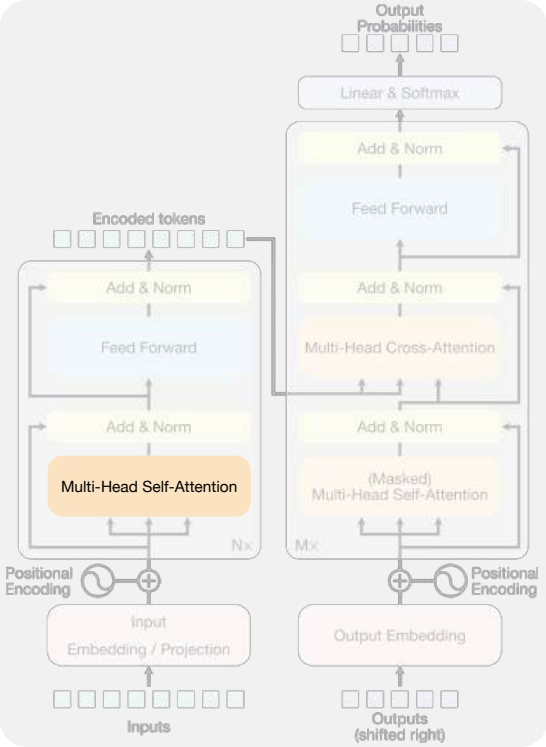
I somewhat do.
I see a lot of pavement.

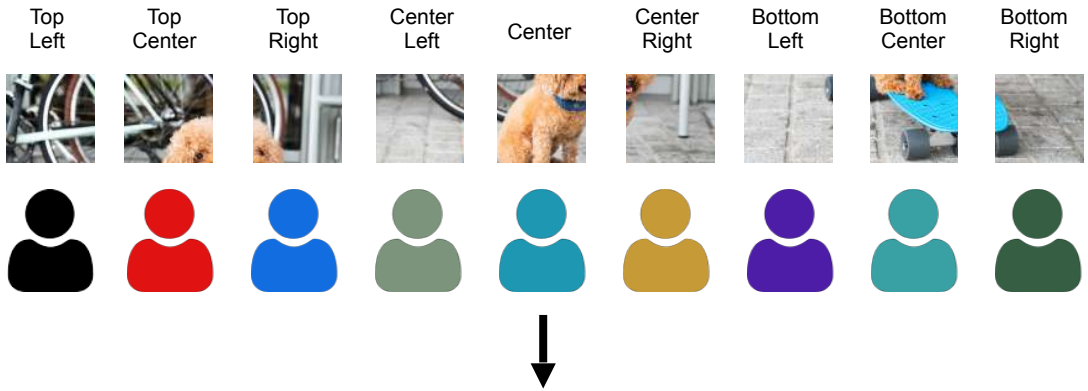
...

[Thanks @David Mizrahi for the slides!]

Attention Is All You Need [Vaswani et al. 2017]

Attention motivation



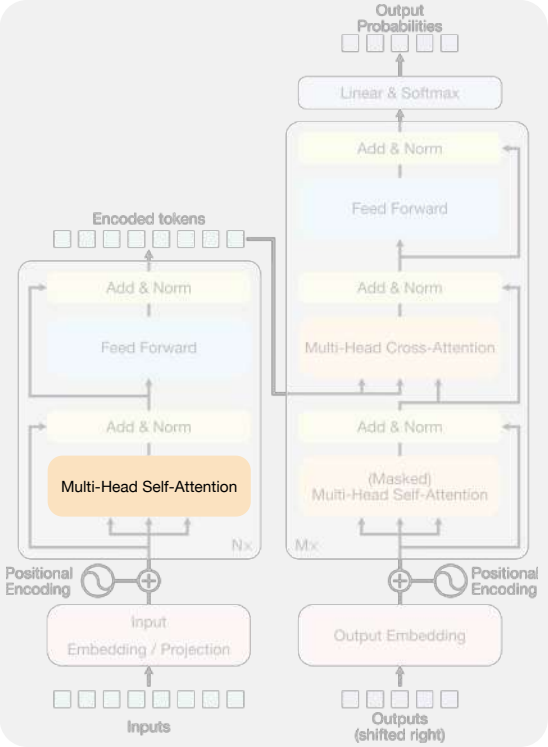


A small dog riding a skateboard in front of a building.

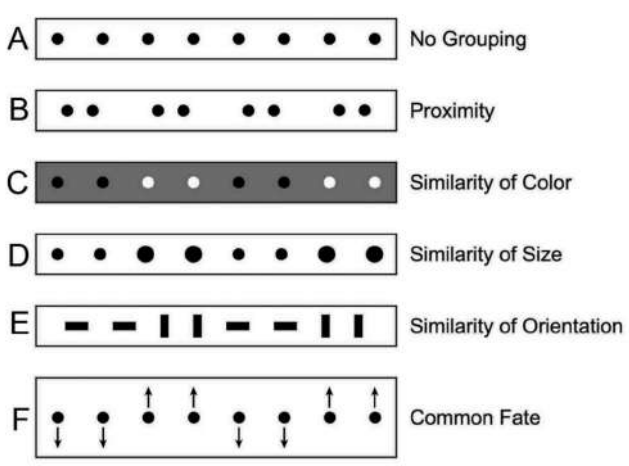
[Thanks @David Mizrahi for the slides!]

Attention Is All You Need [Vaswani et al. 2017]

Attention motivation



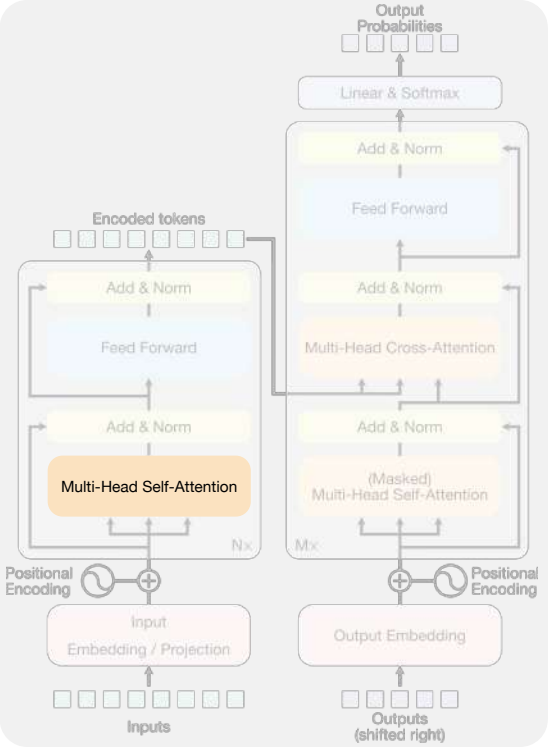
Organize / group visual stimuli through selective attention



[Credit: Brunswick & Kamiya, Jitendra Malik]

Attention Is All You Need [Vaswani et al. 2017]

Different forms of attention



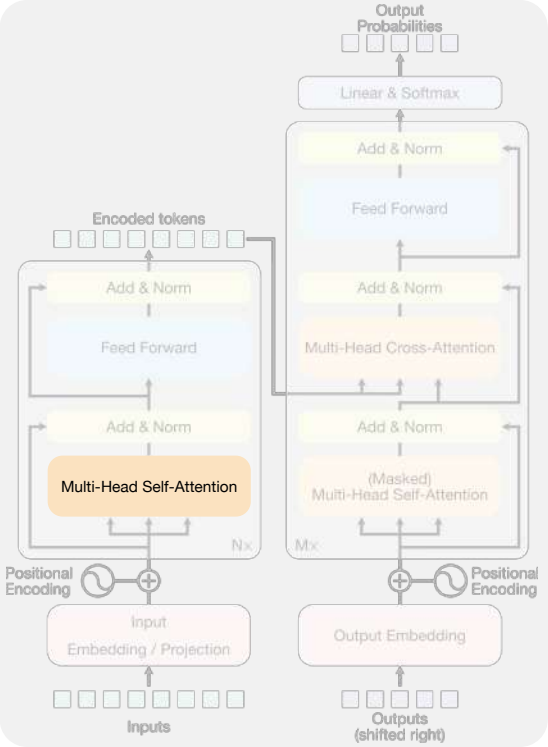
- Many types of alignment score functions to compute attention
- We'll focus on scaled dot-product attention

Name	Alignment score function	Citation
Content-base attention	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \text{cosine}[\boldsymbol{s}_t, \boldsymbol{h}_i]$	Graves2014
Additive(*)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\boldsymbol{s}_t; \boldsymbol{h}_i])$	Bahdanau2015
Location-Base	$\alpha_{t,i} = \text{softmax}(\mathbf{W}_a \boldsymbol{s}_t)$ Note: This simplifies the softmax alignment to only depend on the target position.	Luong2015
General	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \mathbf{W}_a \boldsymbol{h}_i$ where $\mathbf{W}_a$ is a trainable weight matrix in the attention layer.	Luong2015
Dot-Product	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \boldsymbol{h}_i$	Luong2015
Scaled Dot-Product(^)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \frac{\boldsymbol{s}_t^\top \boldsymbol{h}_i}{\sqrt{n}}$ Note: very similar to the dot-product attention except for a scaling factor; where n is the dimension of the source hidden state.	Vaswani2017

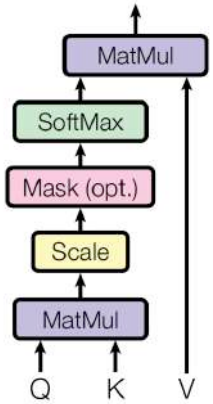
[Credit: "Attention? Attention!" - Lilian Weng]

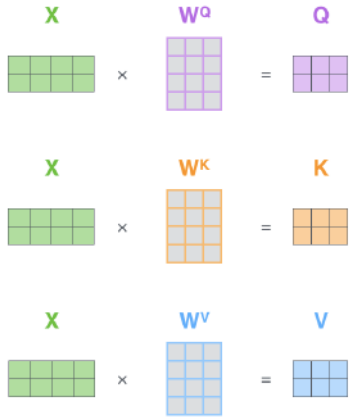
Attention Is All You Need [Vaswani et al. 2017]

Scaled dot-product attention



Attention = *"averaging of **values** associated to **keys** matching a **query**"*

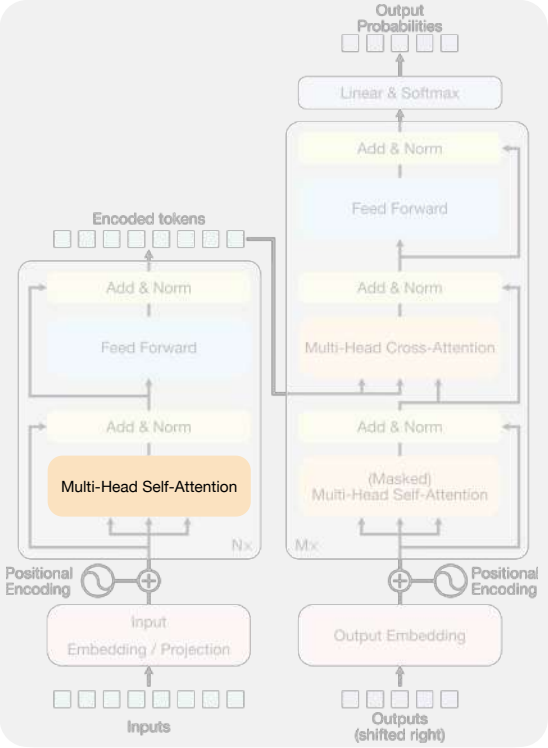




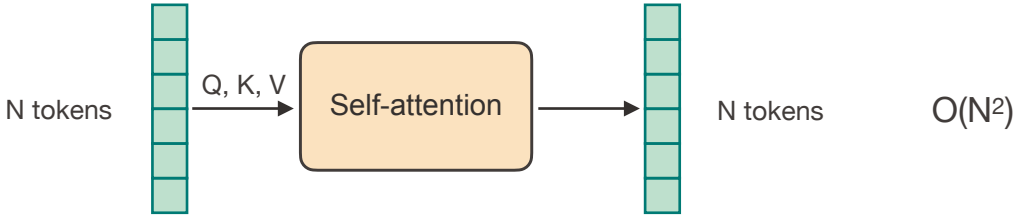
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Attention Is All You Need [Vaswani et al. 2017]

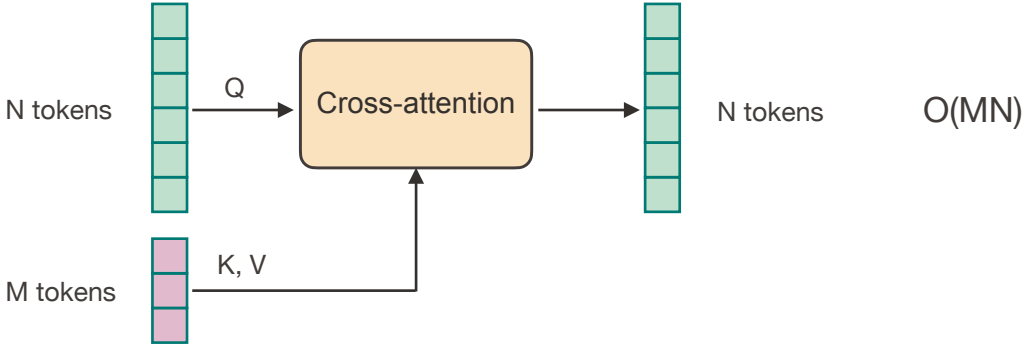
Scaled dot-product attention



- **Self-attention**: Attention from each token to all other tokens

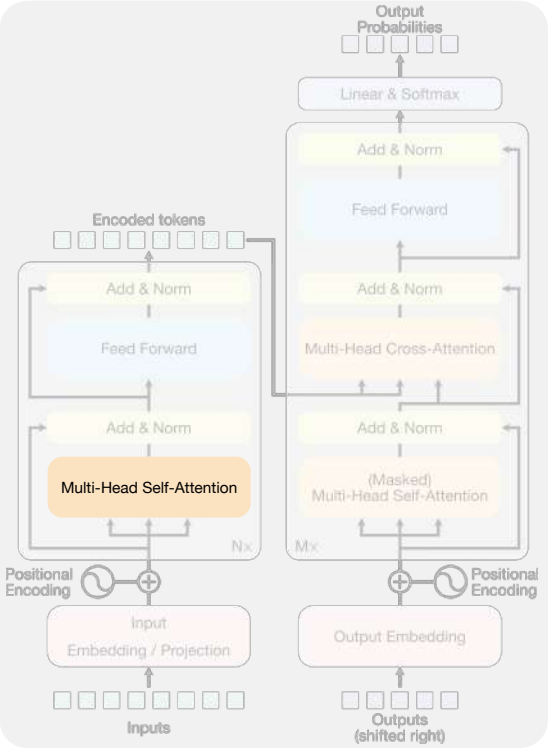


- **Cross-attention**: Attention from one set of tokens to another set of tokens



Attention Is All You Need [Vaswani et al. 2017]

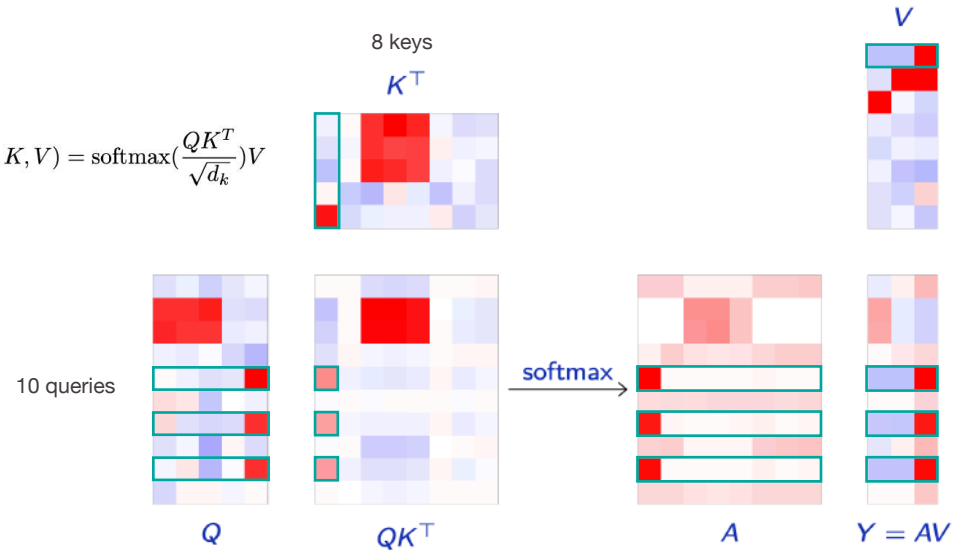
Scaled dot-product attention



Attention = *"averaging of **values** associated to **keys** matching a **query**"*

Example: Cross-attention from 10 queries to 8 keys

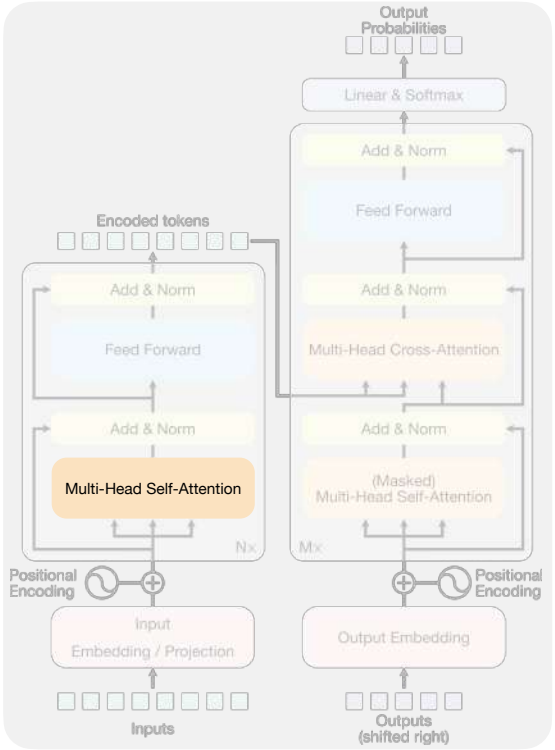
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



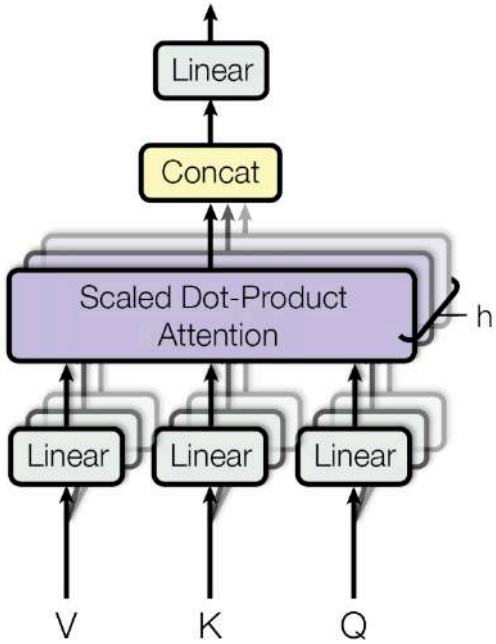
[Credit: Deep Learning Course of UNIGE/EPFL, François Fleuret]

Attention Is All You Need [Vaswani et al. 2017]

Multi-headed Self-Attention



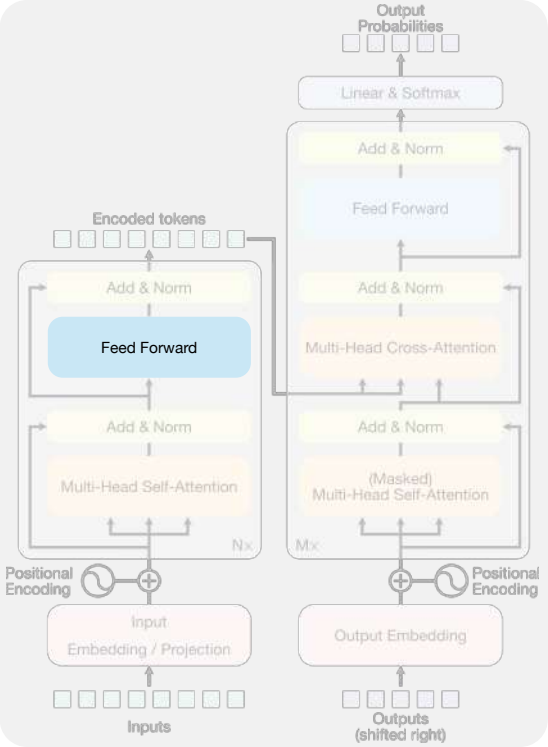
*"**multi-head attention** allows the model to jointly attend to information from different representation **subspaces** at different positions. With a single attention head, averaging inhibits this."*



[Credit: Attention Is All You Need, Vaswani et al. 2017]

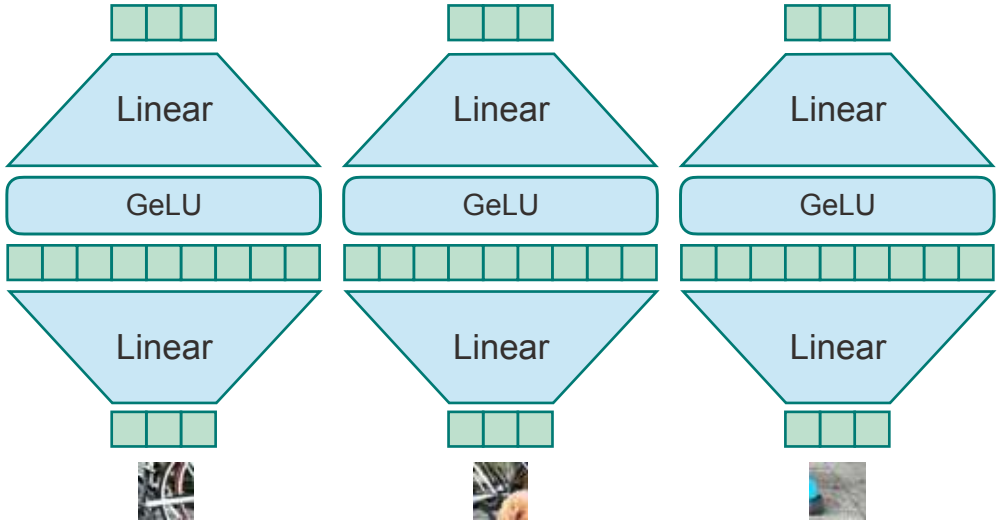
Attention Is All You Need [Vaswani et al. 2017]

Feed-forward layers



An MLP applied to each token individually ("point-wise"):

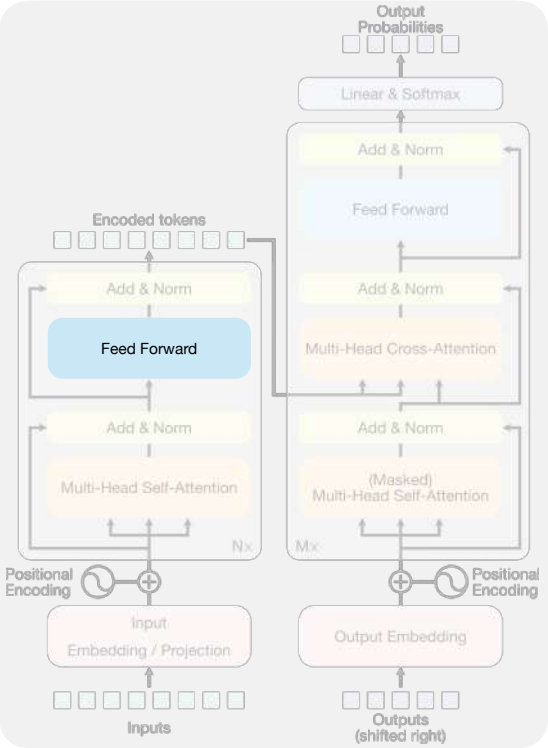
$$\mathrm{FF}(x) = W_2 \mathrm{GeLU}(W_1 x + b_1) + b_2$$



Contains the bulk of parameters!

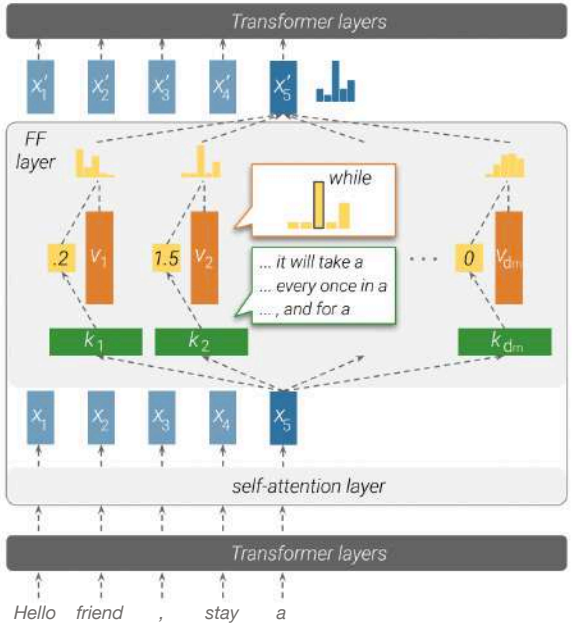
Attention Is All You Need [Vaswani et al. 2017]

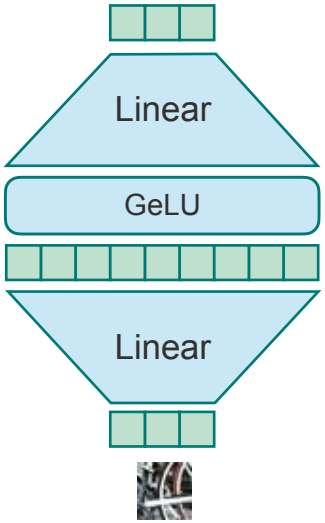
Feed-forward layers



Can be interpreted as key-value memory:

$$\mathrm{FF}(x) = f(x \cdot K^\top)V \quad \text{(omitting bias)}$$

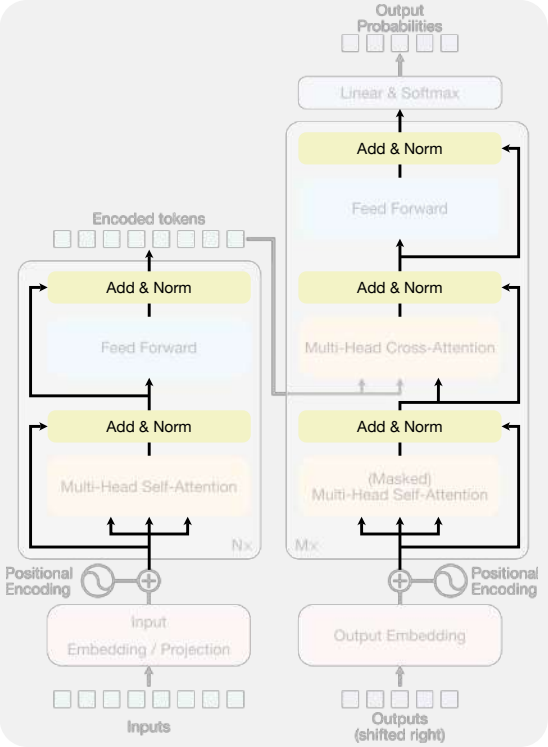




[Transformer Feed-Forward Layers Are Key-Value Memories. Geva et al. 2020]

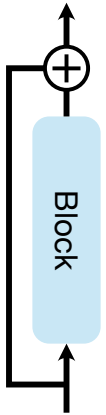
Attention Is All You Need [Vaswani et al. 2017]

Skip connections & normalization



Skip connections:

- $z_i = \text{Module}(x_i) + x_i$
- Following ResNets, dramatically improves trainability

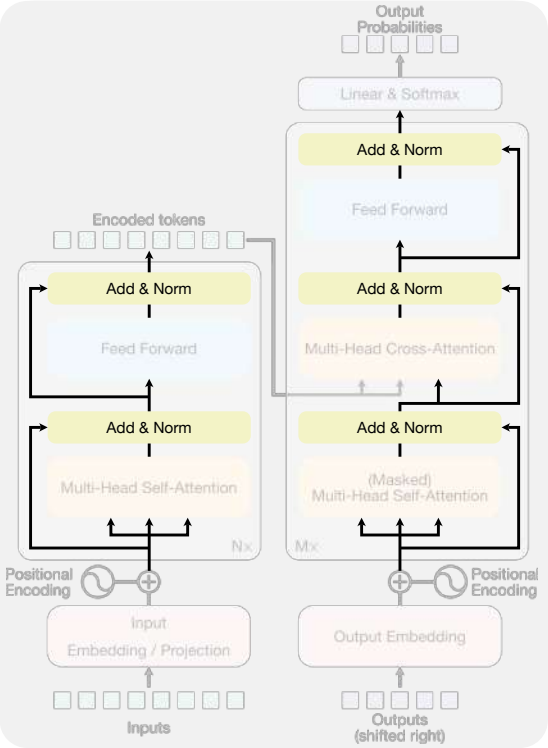


LayerNorm:

- Dramatically improves trainability
- Post-norm (original): $z_i = \text{LN}(\text{Module}(x_i) + x_i)$
- Pre-norm (modern): $z_i = \text{Module}(\text{LN}(x_i)) + x_i$
- Sandwich norm: Pre-norm + post-norm

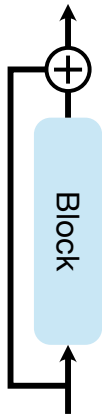
Attention Is All You Need [Vaswani et al. 2017]

Residual stream perspective

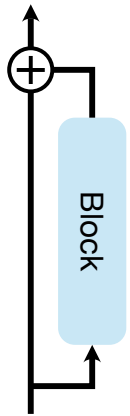


Skip connections:

- $z_i = \text{Module}(x_i) + x_i$
- Following ResNets, dramatically improves trainability



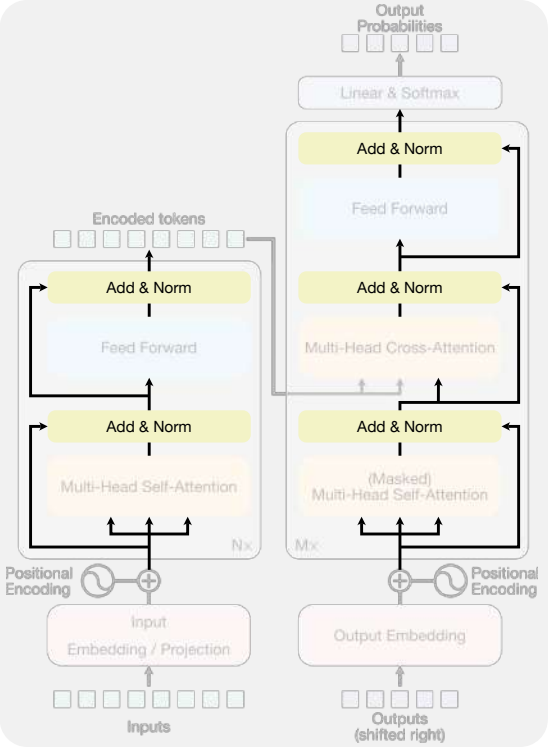
"Skip connection"



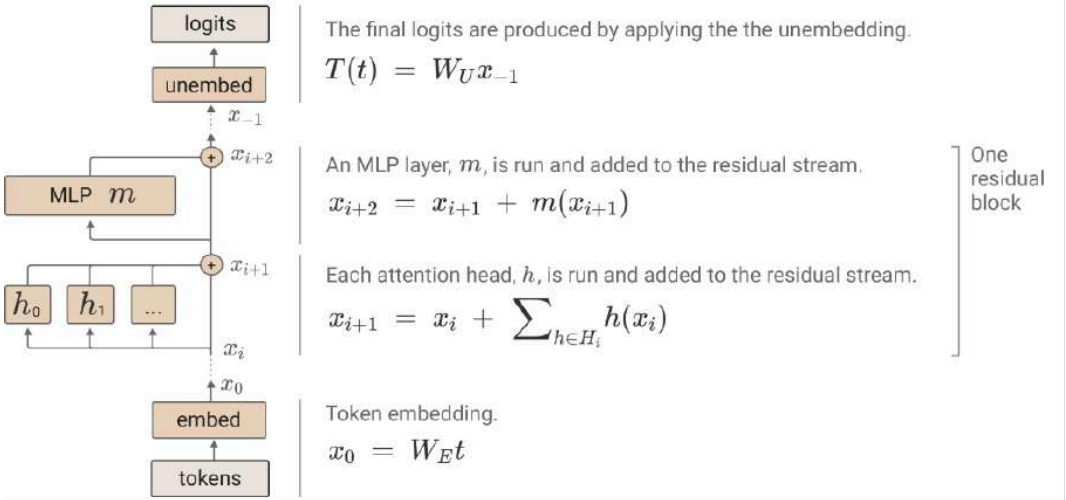
"Residual block"

Attention Is All You Need [Vaswani et al. 2017]

Residual stream perspective



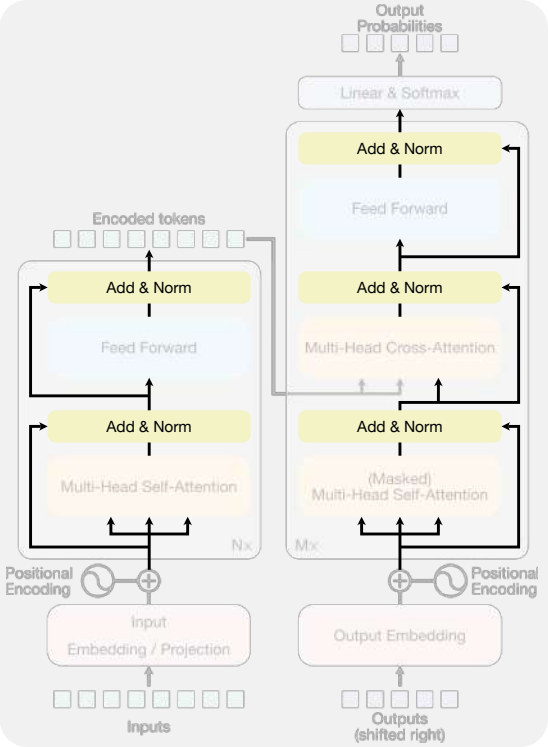
All components of a transformer communicate with each other by reading and writing to different subspaces of the residual stream



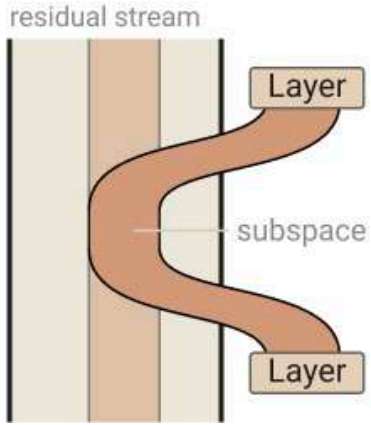
[A Mathematical Framework for Transformer Circuits. Elhage et al. 2021]

Attention Is All You Need [Vaswani et al. 2017]

Residual stream perspective



Residual stream is high dimensional and can be divided into different subspaces

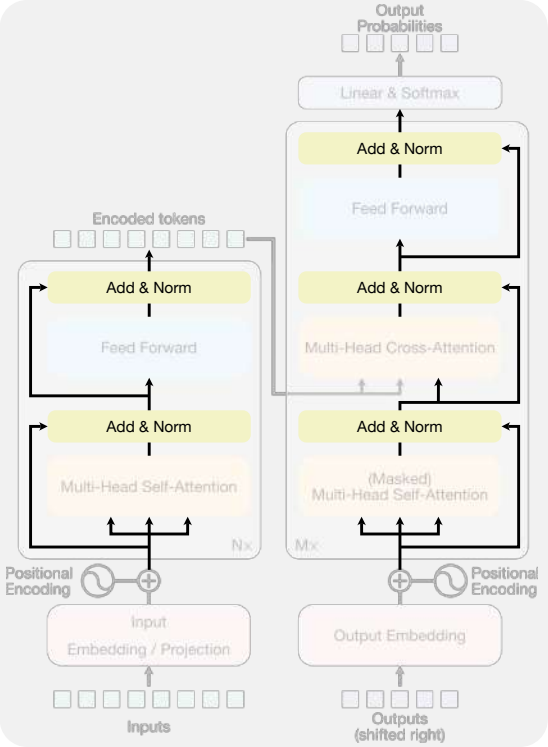


Layers can interact by writing to and reading from the same or overlapping subspaces. If they write to and read from disjoint subspaces, they won't interact. Typically the spaces only partially overlap.

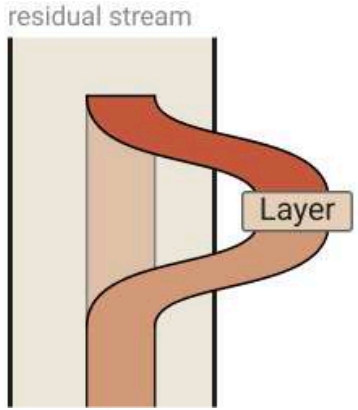
[A Mathematical Framework for Transformer Circuits. Elhage et al. 2021]

Attention Is All You Need [Vaswani et al. 2017]

Residual stream perspective



Memory management: Residual stream subspaces can be cleared

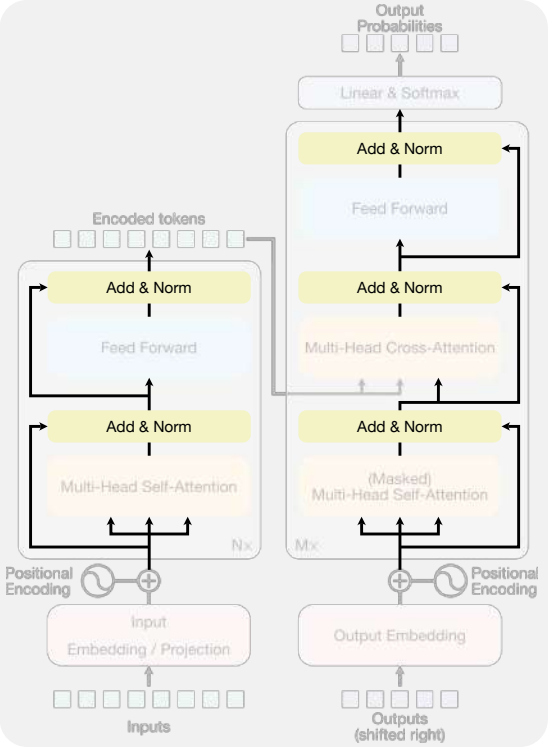


Layers can delete information from the residual stream by reading in a subspace and then writing the negative verison.

[A Mathematical Framework for Transformer Circuits. Elhage et al. 2021]

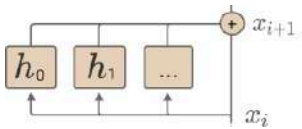
Attention Is All You Need [Vaswani et al. 2017]

Residual stream perspective



Attention heads can be understood as independent operations, each outputting a result which is added into the residual stream

$$W_O^H \begin{bmatrix} r^{h_1} \\ r^{h_2} \\ \dots \end{bmatrix} = \left[W_O^{h_1}, W_O^{h_2}, \dots \right] \cdot \begin{bmatrix} r^{h_1} \\ r^{h_2} \\ \dots \end{bmatrix} = \sum_i W_O^{h_i} r^{h_i}$$



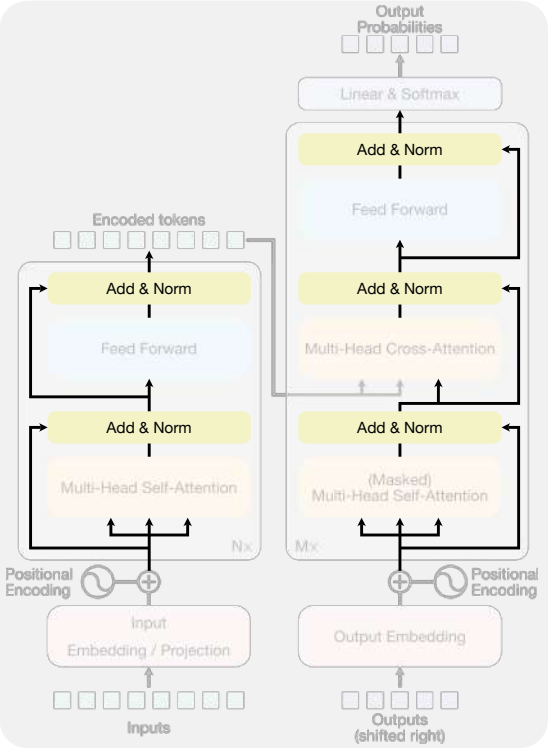
Each attention head, h, is run and added to the residual stream.

$$x_{i+1} = x_i + \sum_{h \in H_i} h(x_i)$$

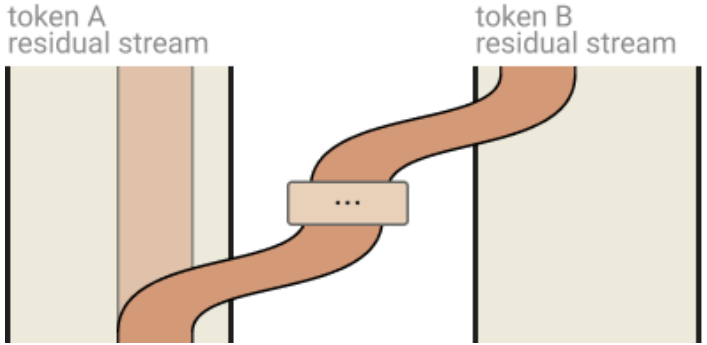
[A Mathematical Framework for Transformer Circuits. Elhage et al. 2021]

Attention Is All You Need [Vaswani et al. 2017]

Residual stream perspective



Attention heads as information movement

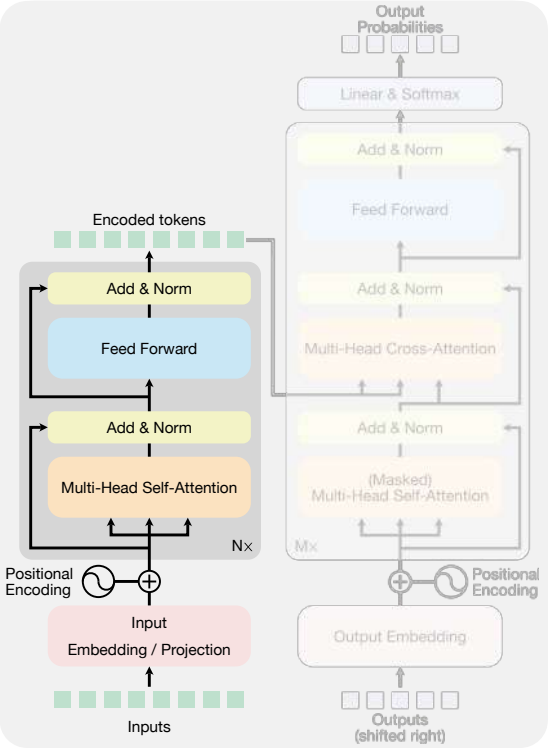


Attention heads copy information from the residual stream of one token to the residual stream of another. They typically write to a different subspace than they read from.

[A Mathematical Framework for Transformer Circuits. Elhage et al. 2021]

Attention Is All You Need [Vaswani et al. 2017]

Encoder

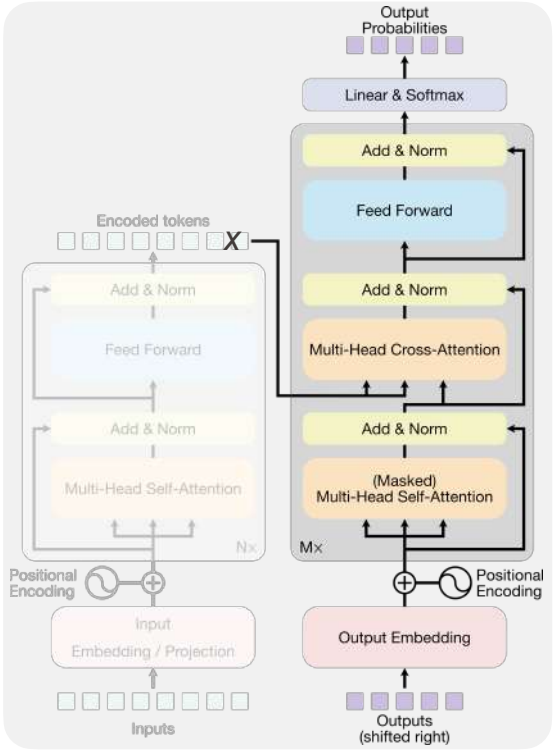


- Input / output dimension of each block is identical → stack N blocks
- For encoder-decoder transformers, e.g. 6 (“base”), 12 (“large”), etc.
- Encoder output is a processed version of the input (but not the output yet!)

Attention Is All You Need [Vaswani et al. 2017]

Decoder

$p(z_3 \mid z_2, z_1, x)$



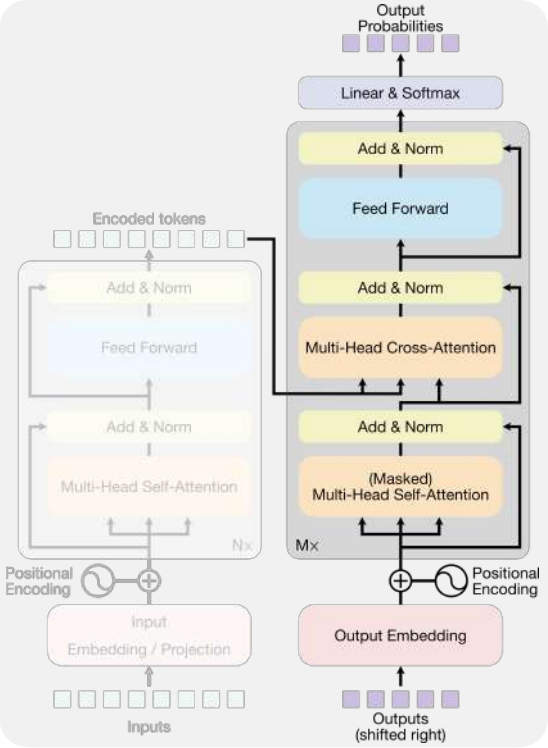
z_1, z_2

- Generates target sequences conditioned on encoded tokens
- I.e. we want to model:
  $p(z \mid x) = p(z_1, z_2, \ldots, z_L \mid x)$
  $= p(z_1 \mid x)\, p(z_2 \mid z_1, x)\, p(z_3 \mid z_2, z_1, x) \ldots$
- E.g. for generating $p(z_3 \mid z_2, z_1, x)$:
  - x = encoded tokens
  - z_1 and z_2 are already predicted. Loop back into decoder input

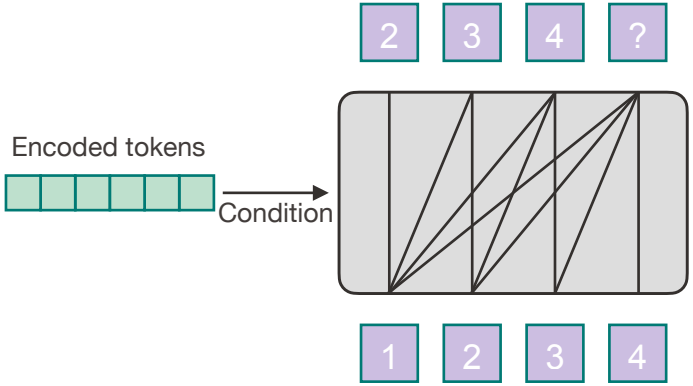
Attention Is All You Need [Vaswani et al. 2017]

Bachmann

Decoder

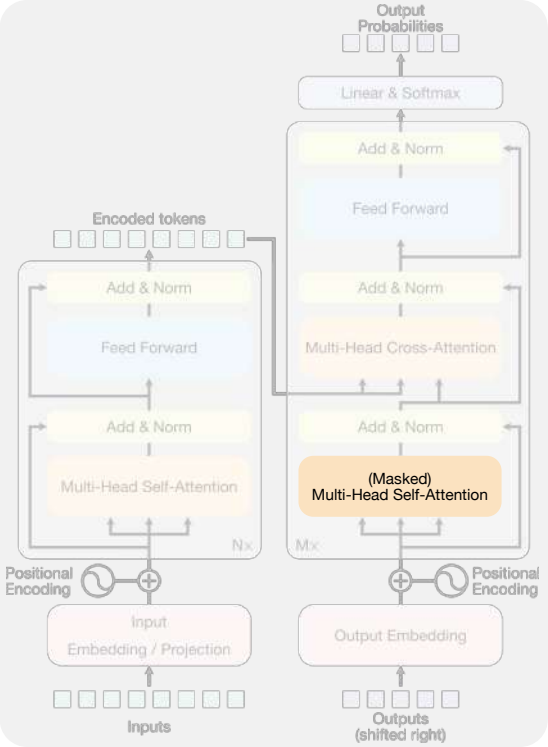


Inference: Decode auto-regressively (i.e. one token at a time)

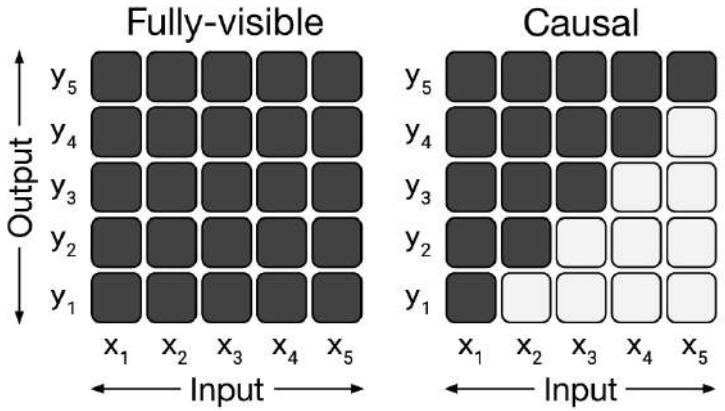


Attention Is All You Need [Vaswani et al. 2017]

Masked self-attention

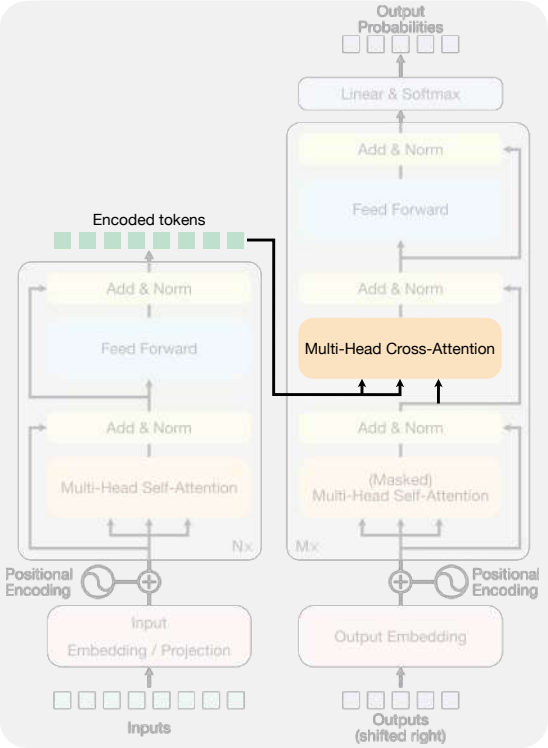


- Like standard self-attention, but apply a mask on the attention matrix
- E.g. causal self-attention: Prevent tokens to look at “future” tokens



Attention Is All You Need [Vaswani et al. 2017]

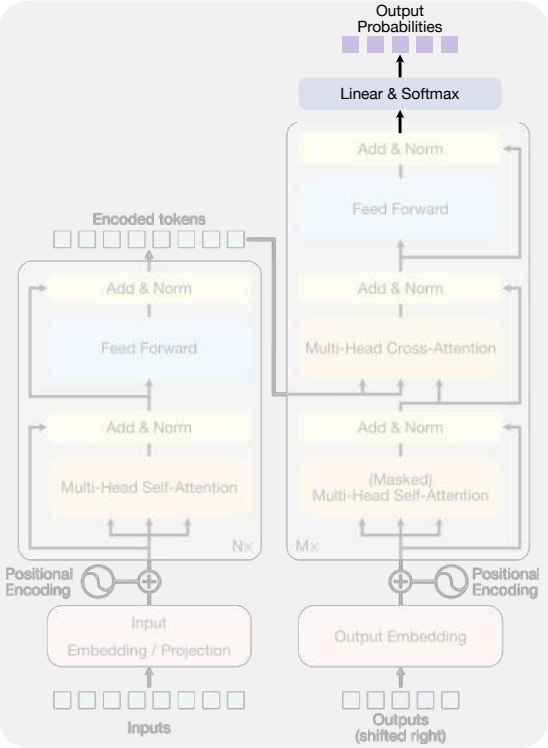
Cross-attention



- Decoder reads relevant information from encoded tokens
- Attention from Decoder tokens to Encoder tokens
  - Queries from Decoder
  - Keys and Values from Encoder
- Cross-attention allows information retrieval from arbitrary sets of tokens (i.e. number of encoder tokens can be different than number of decoder tokens)

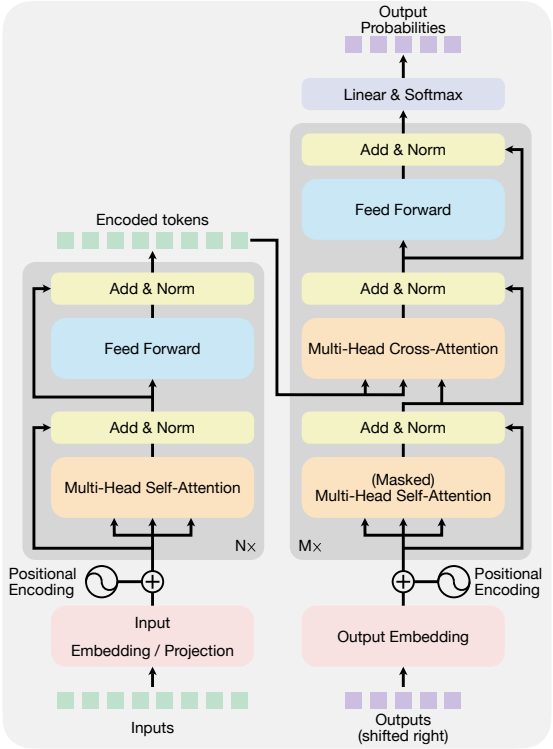
Attention Is All You Need [Vaswani et al. 2017]

Output layer



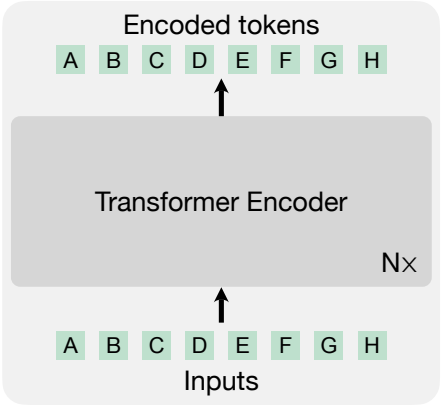
- Predict a probability distribution over the vocabulary
- Linear & Softmax

Attention Is All You Need [Vaswani et al. 2017]



Common transformer types

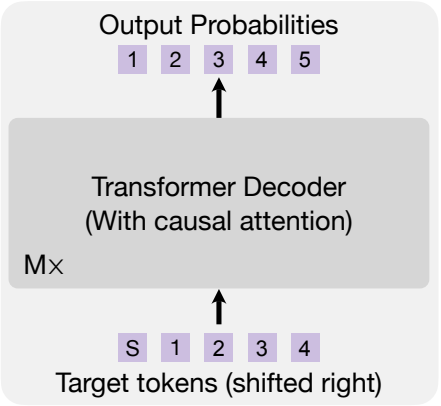
“Encoder-only”



Encoding & masking

E.g. ViT, BERT,
MaskGIT, etc...

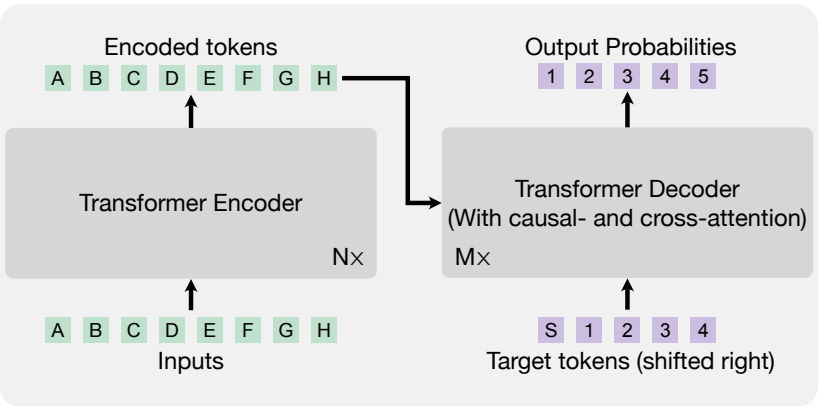
“Decoder-only”



Next-token prediction

E.g. GPT,
LLamaGen, etc...

“Encoder-decoder”



Sequence-to-sequence

E.g. T5, 4M,
Unified-IO, etc...

Modern architecture modifications

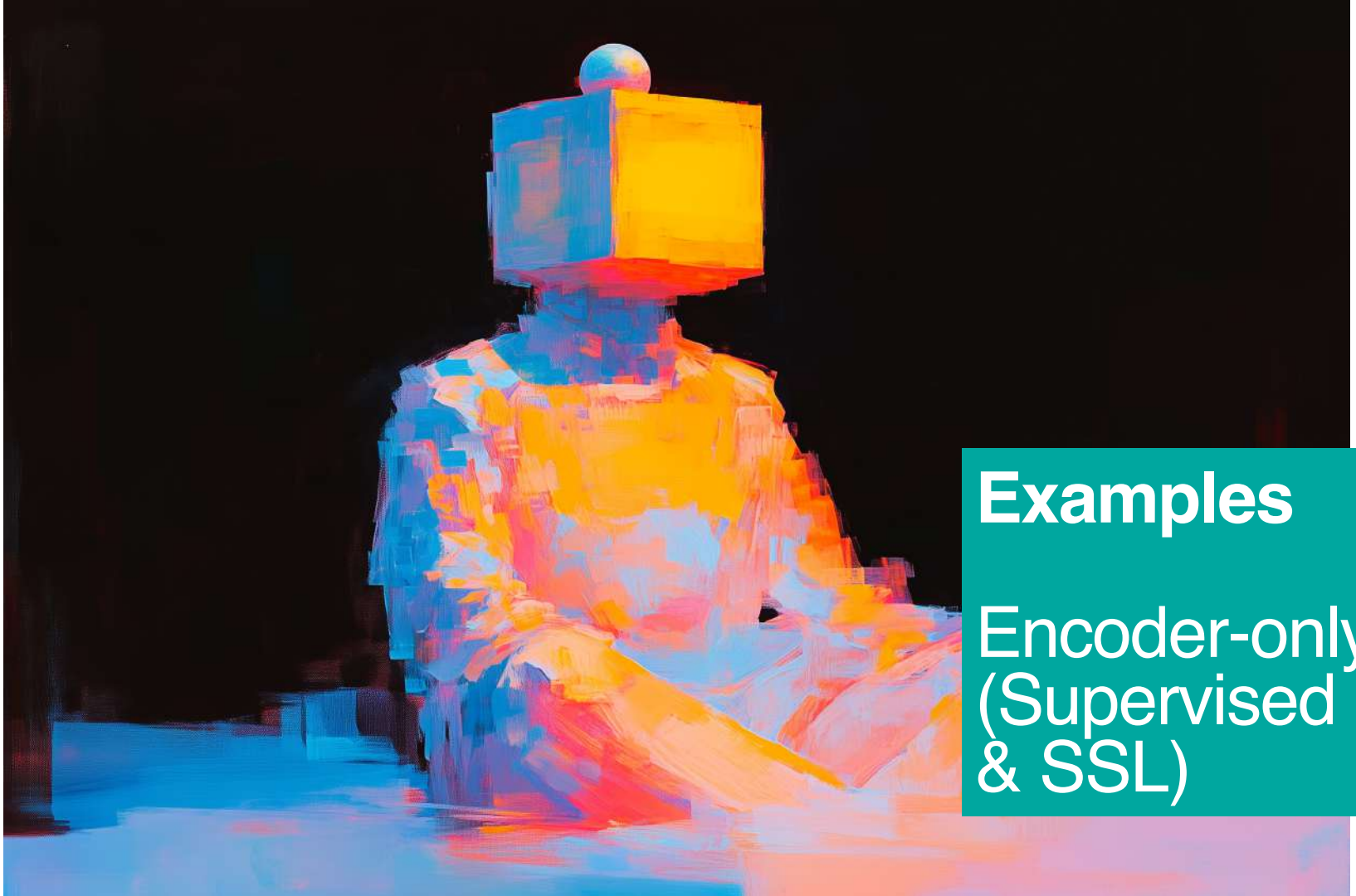
Some pointers

- **Normalization**:
  - Pre-norm instead of post-norm
  - RMSNorm instead of LayerNorm
- **Positional embeddings**: RoPE instead of absolute
- **FFN/Activation**: SwiGLU instead of MLP+GLU
- **Attention**:
  - Grouped Query Attention for more efficient KV-caches
  - QK-Normalization for stability

Efficiency improvements

Some pointers

- **KV-caching**: Avoid redundant recalculation by caching keys and values
- **Speculative decoding**: Use a smaller model for fast autoregressive inference and verify with larger model
- **FlashAttention / RingAttention**: Reduce attention memory overhead from $O(N^2)$ to $O(N)$, while time complexity is still $O(N^2)$. Much faster than naive implementation thanks to GPU optimizations.
- **Alternative architectures**: E.g. state-space models, linear attention, ...



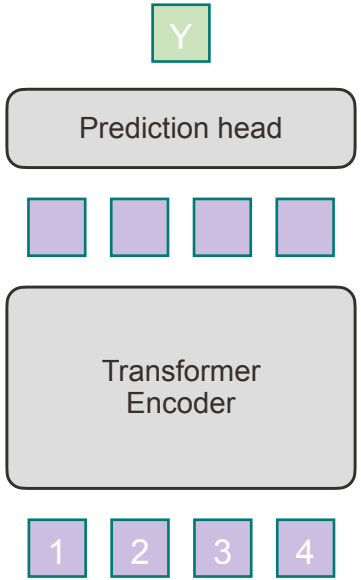
Examples

Encoder-only (Supervised & SSL)

Encoder-only (supervised training)

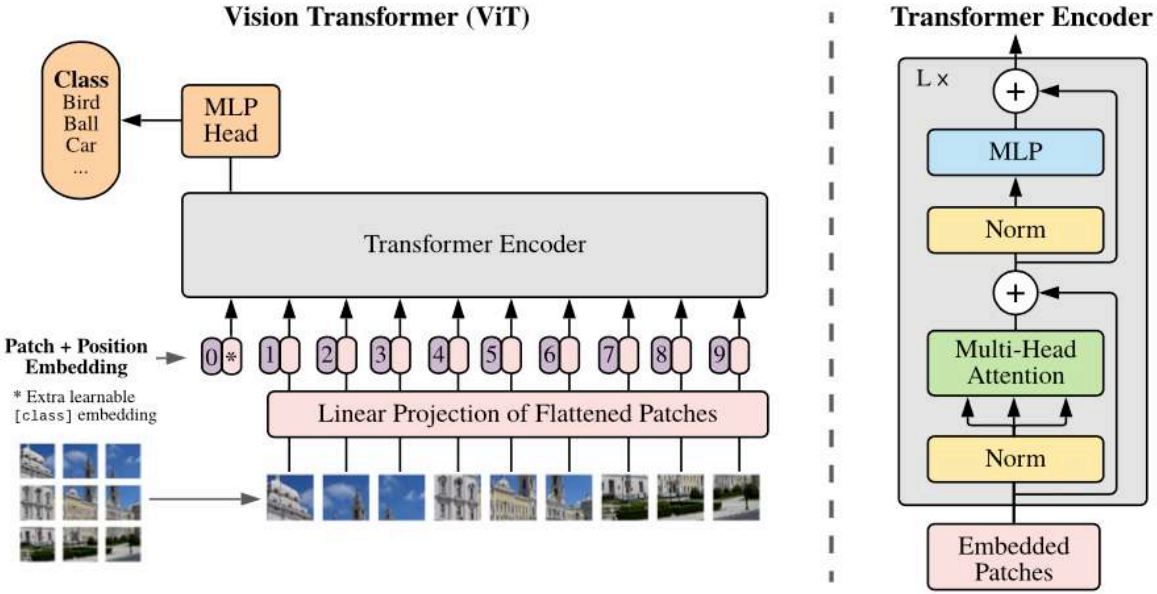
ViT, DPT, self-supervised methods, …

- Treat (almost) like a classical ConvNet backbone
- Attach some prediction head (e.g. linear layer or convolutional layers)



Vision Transformers (ViT)

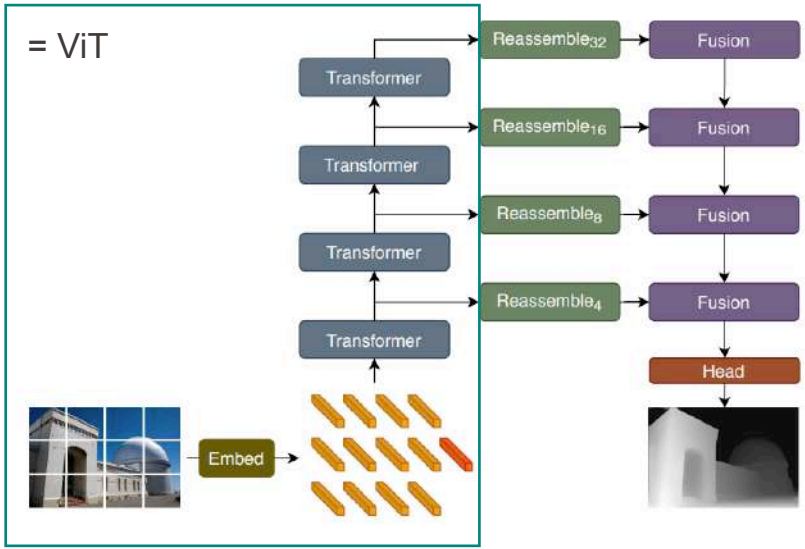
- Flattened images are too large for transformers
- **Idea**: Model image as sequence of patches and use vanilla transformer
- Add extra learnable class token for classification head



Credit: An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, Dosovitskiy et al. 2020

Dense Prediction Transformers (DPT)

- ViTs are classification architectures
- What about dense tasks (e.g. depth estimation, semantic segmentation)?
- Global receptive field and full resolution at every layer!



Credit: Vision Transformers for Dense Prediction, Ranftl et al. 2021

Dense Prediction Transformers (DPT)

- Monocular depth estimation (MiDaS) visuals:

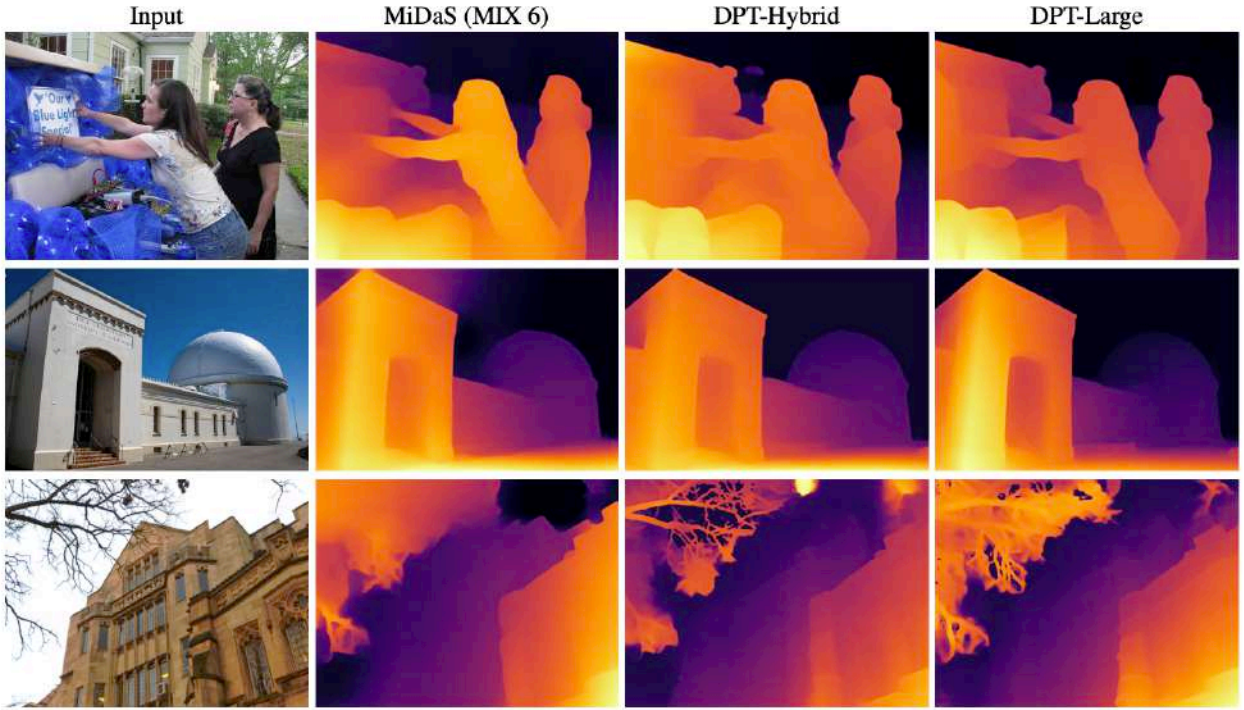


Figure 2. Sample results for monocular depth estimation. Compared to the fully-convolutional network used by MiDaS, DPT shows better global coherence (e.g., sky, second row) and finer-grained details (e.g., tree branches, last row).

Credit: Vision Transformers for Dense Prediction, Ranftl et al. 2021

Dense Prediction Transformers (DPT)

- Semantic segmentation (ADE20K) visuals:

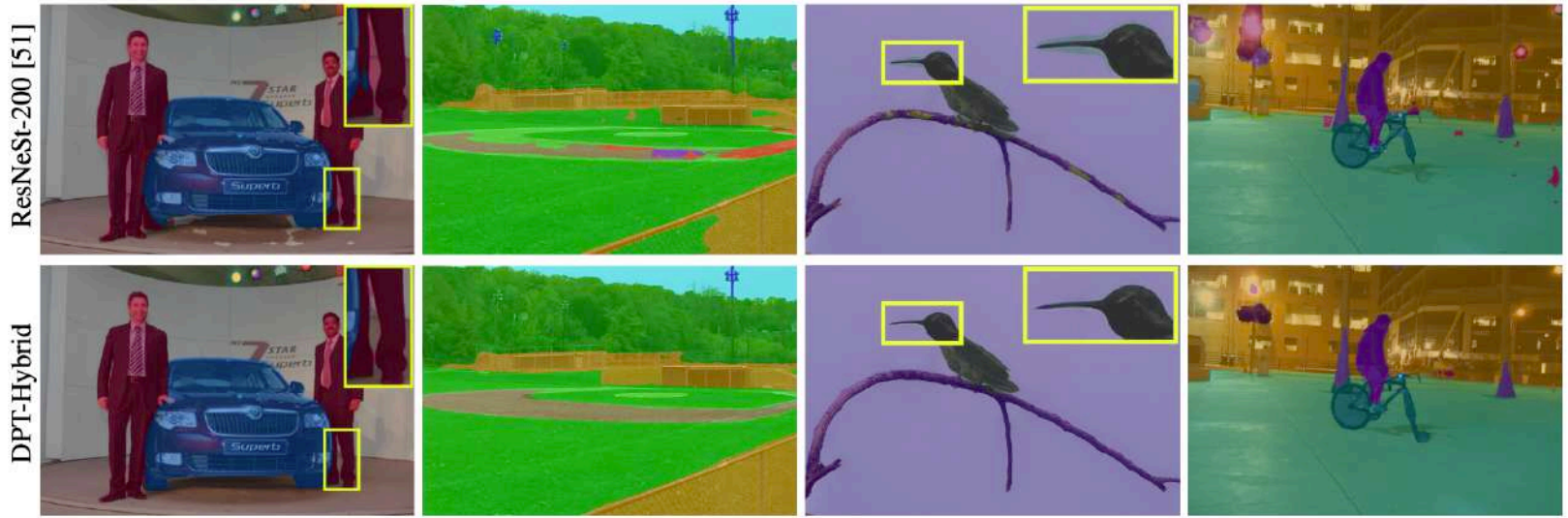
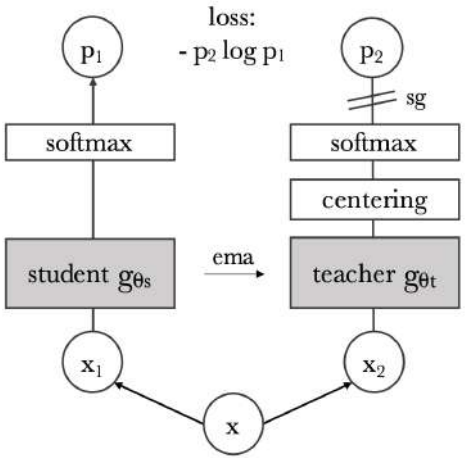


Figure 3. Sample results for semantic segmentation on ADE20K (first and second column) and Pascal Context (third and fourth column). Predictions are frequently better aligned to object edges and less cluttered.

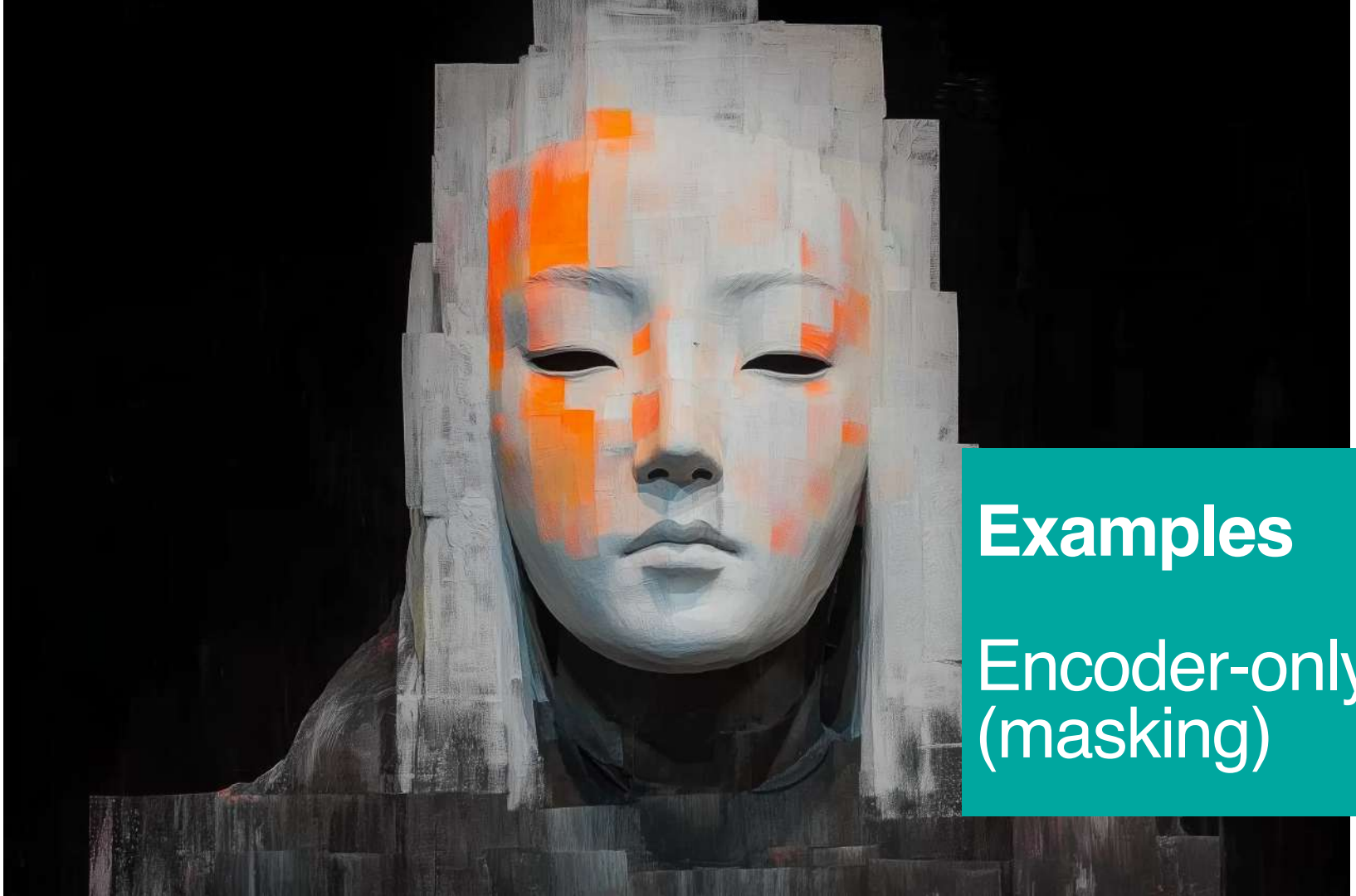
Credit: Vision Transformers for Dense Prediction, Ranftl et al. 2021

Self-supervised learning. E.g. DINO

- Train ViT in self-supervised manner (self-distillation)
- Attention maps of different heads semantically interesting



Credit: Emerging Properties in Self-Supervised Vision Transformers, Caron et al. 2021



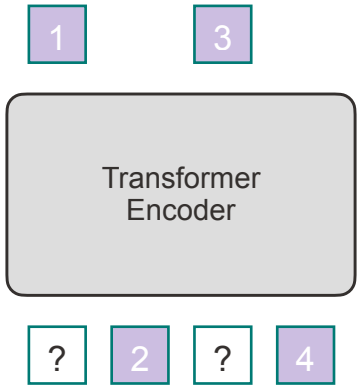
Examples

Encoder-only (masking)

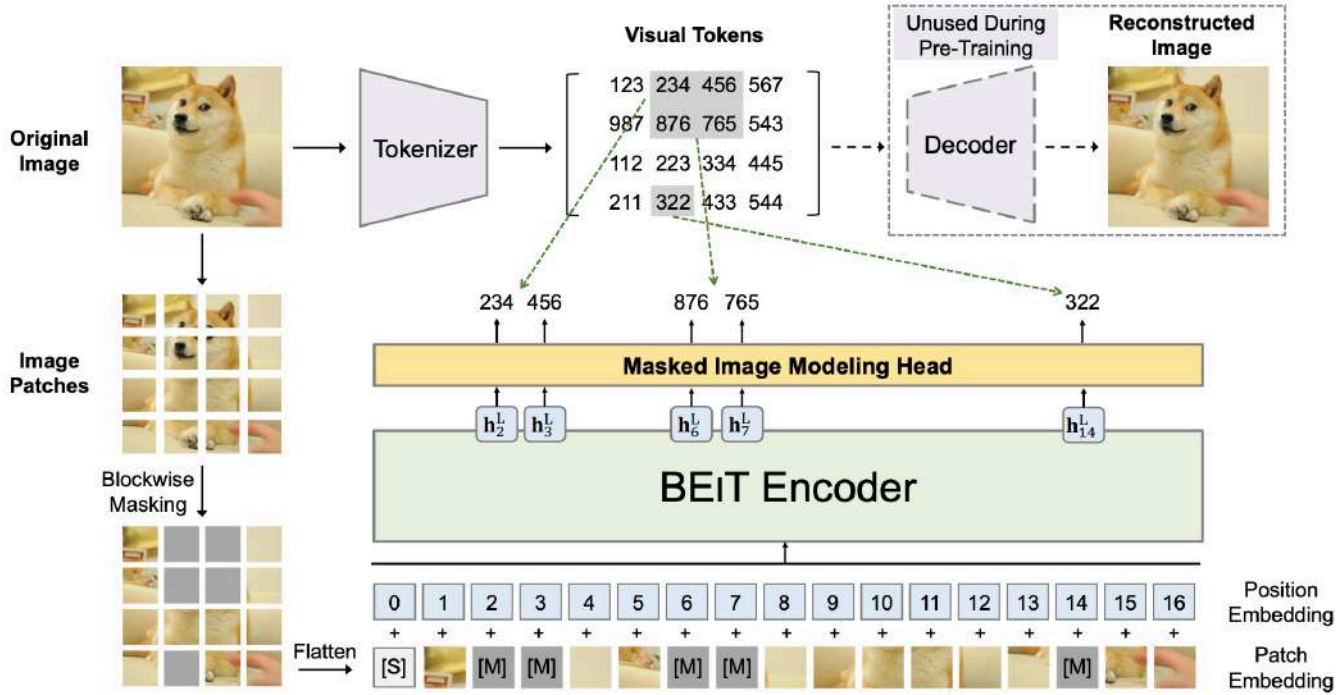
Encoder-only (masking)

BERT, BEiT, MaskGIT, …

- Inspired by success of masked pre-training in NLP (BERT)
- **Idea**:
  - 1) Predict masked-out patches / tokens
  - 2) Use as backbone and fine-tune to downstream task



BEiT: BERT Pre-Training of Image Transformers



Credit: BEIT: BERT Pre-Training of Image Transformers, Bao et al. 2021

BEiT: BERT Pre-Training of Image Transformers

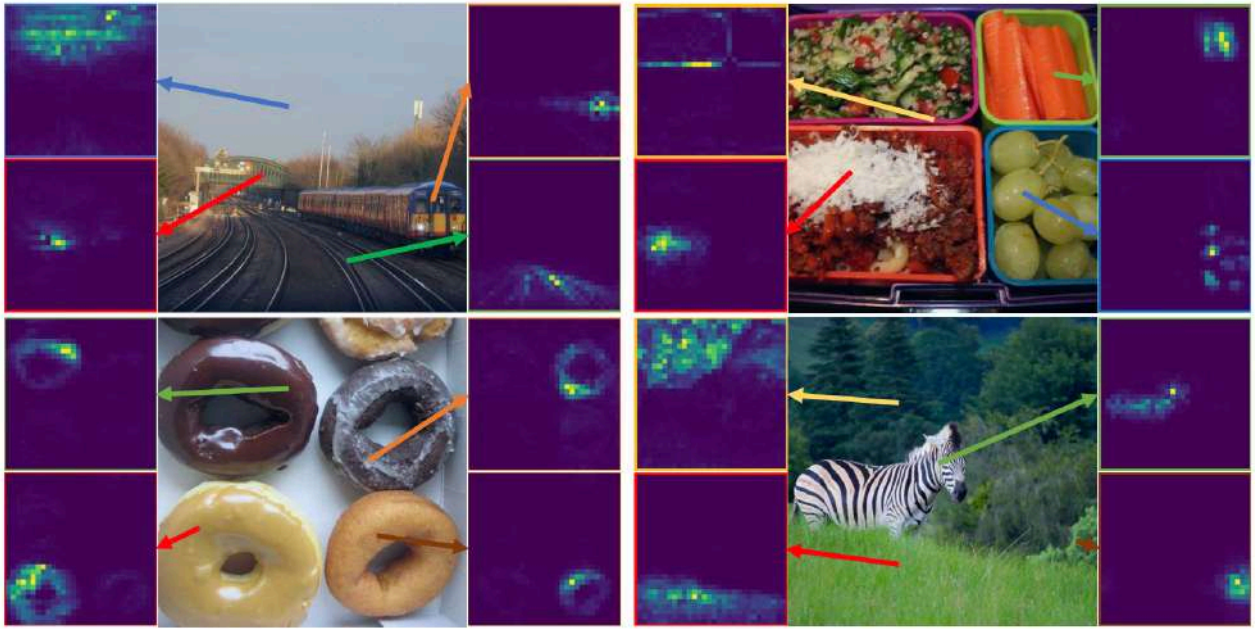
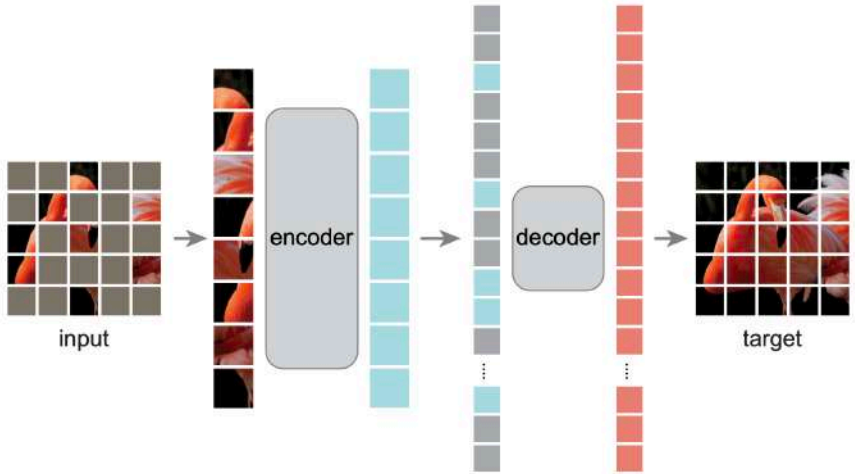


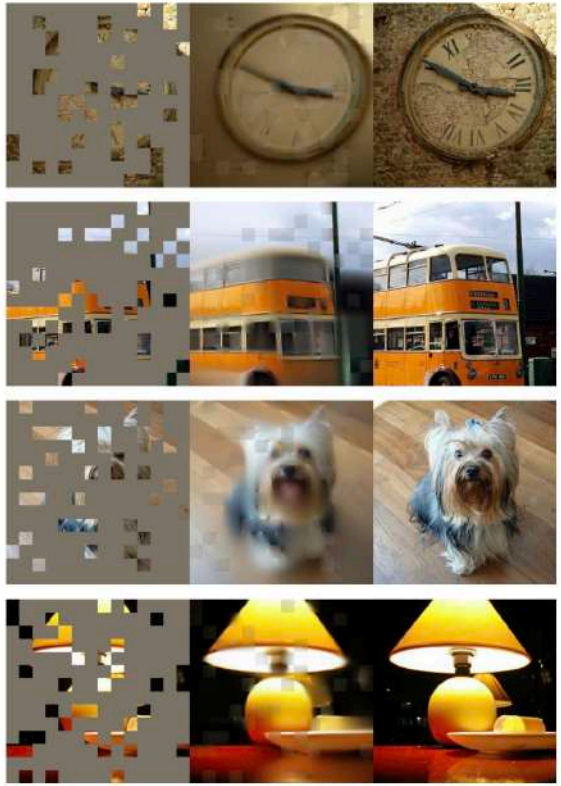
Figure 3: Self-attention map for different reference points. The self-attention mechanism in BEIT is able to separate objects, although self-supervised pre-training does not use manual annotations.

Credit: BEIT: BERT Pre-Training of Image Transformers, Bao et al. 2021

Masked Autoencoders

- **Idea**: Only encode visible tokens. Read out missing tokens with shallow decoder.



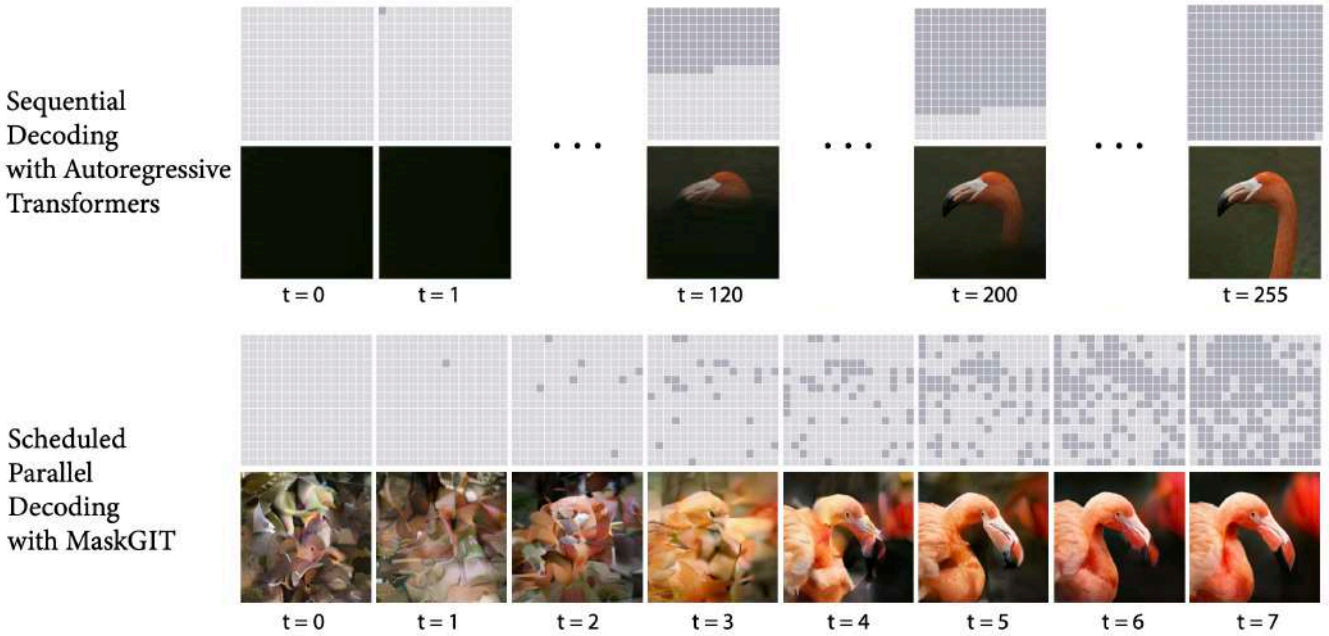


Credit: Masked Autoencoders Are Scalable Vision Learners, He et al. 2021

Discrete diffusion / Parallel decoding

MaskGIT, Muse, MAGE, …

- Token-based masked image models are fast generative models



Credit: MaskGIT: Masked Generative Image Transformer, Chang et al. 2022

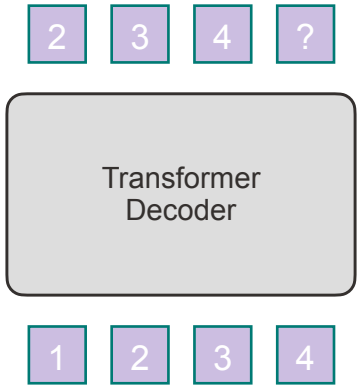
Examples

Decoder-only (next token prediction)

Decoder-only (next token prediction)

GPT, DALL·E 1, LLamaGen, AIM, …

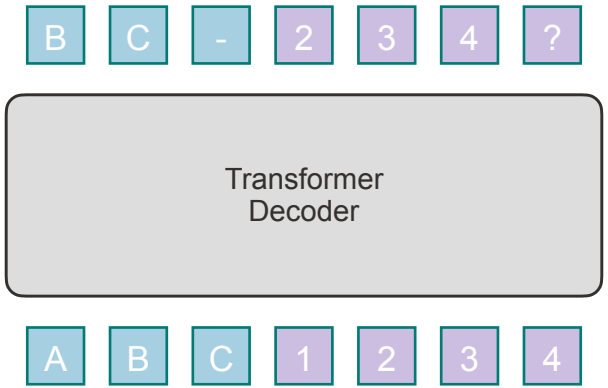
- Train for next token prediction (w/ causal attention)
- Autoregressive decoding



Decoder-only (next token prediction)

GPT, DALL·E 1, LLamaGen, AIM, …

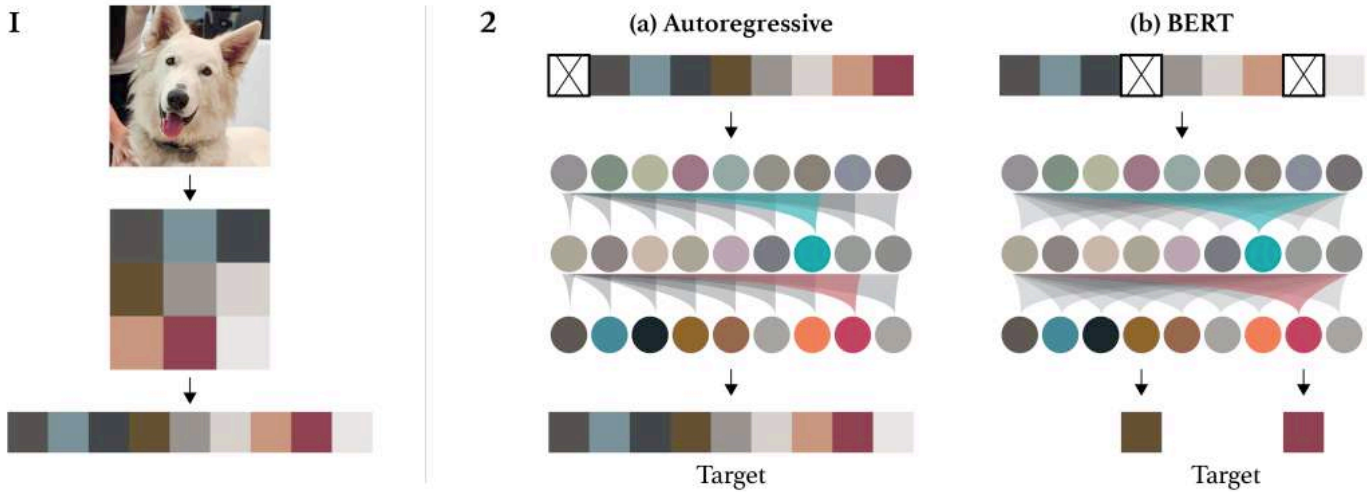
- Train for next token prediction (w/ causal attention)
- Autoregressive decoding



- Condition by prepending tokens (context). Also called PrefixLM.

Image GPT

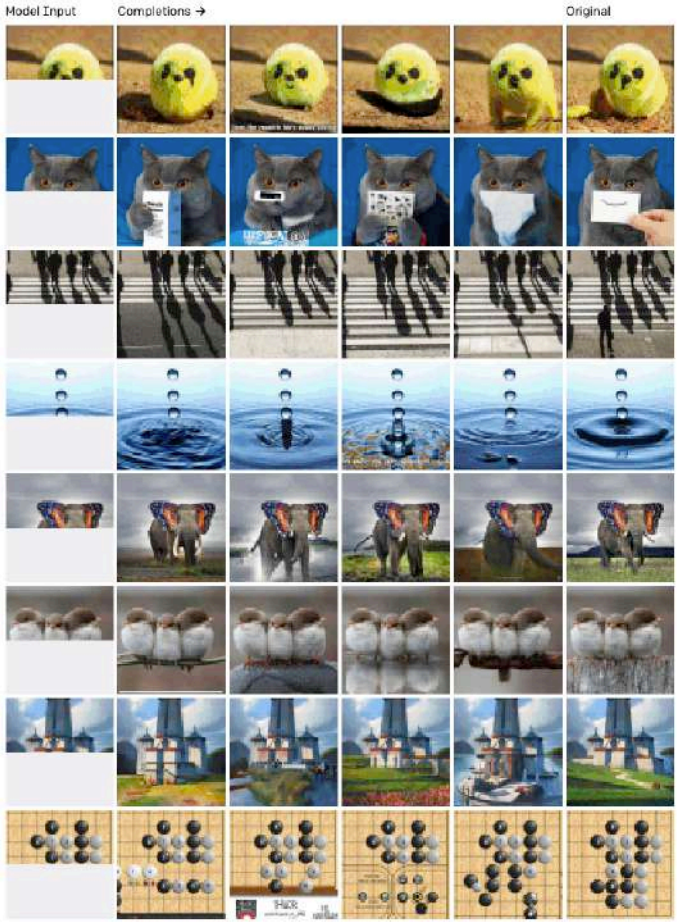
- First turn pictures into small set of tokens
- Then train a GPT on them



Credit: Generative Pretraining from Pixels, Chen et al. 2020

Image GPT

- Samples and completions

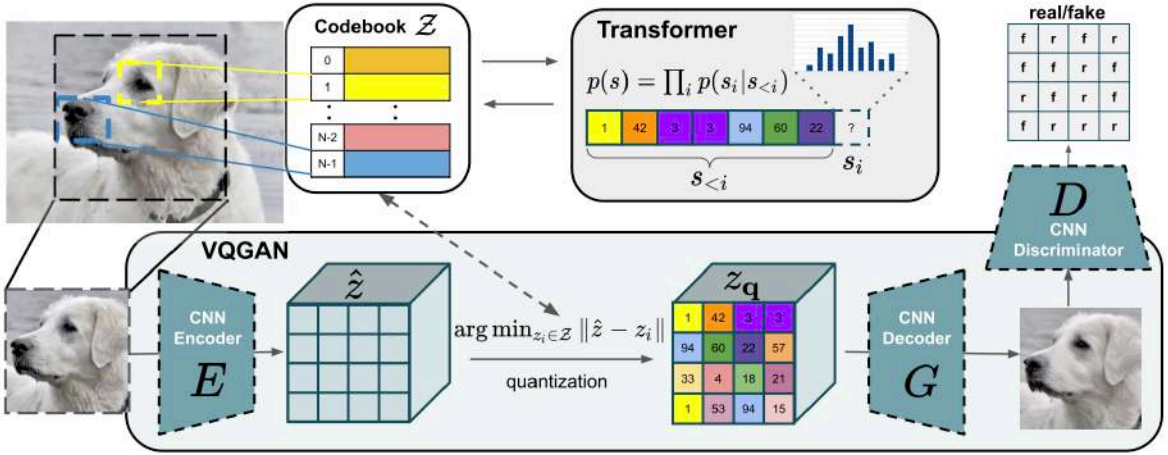


Credit: Generative Pretraining from Pixels, Chen et al. 2020

Taming Transformers for High-Resolution Image Synthesis

Transformers seem great at modelling sequences, but quadratic complexity is problematic.

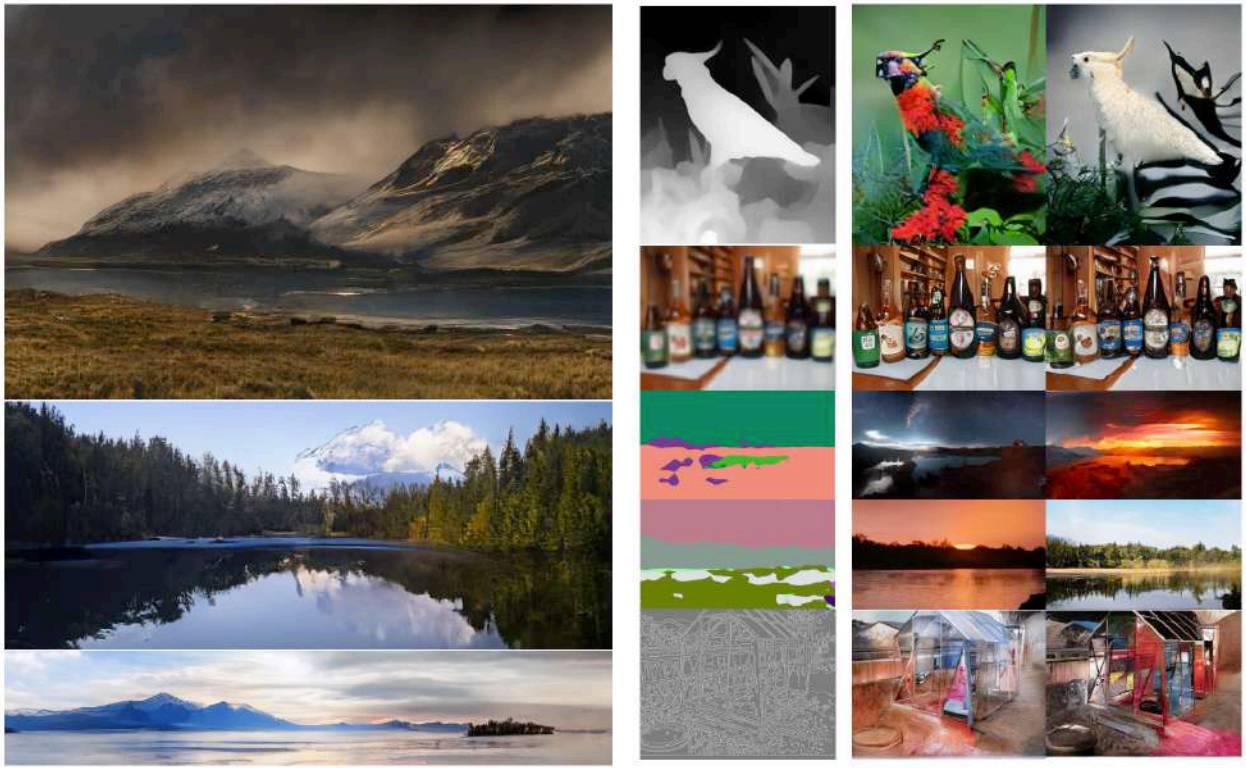
- **Idea**: Reduce sequence length by modelling latent space of a discrete VAE



Credit: Taming Transformers for High-Resolution Image Synthesis, Esser et al. 2020

Figure 3. Sliding attention window.

Taming Transformers for High-Resolution Image Synthesis



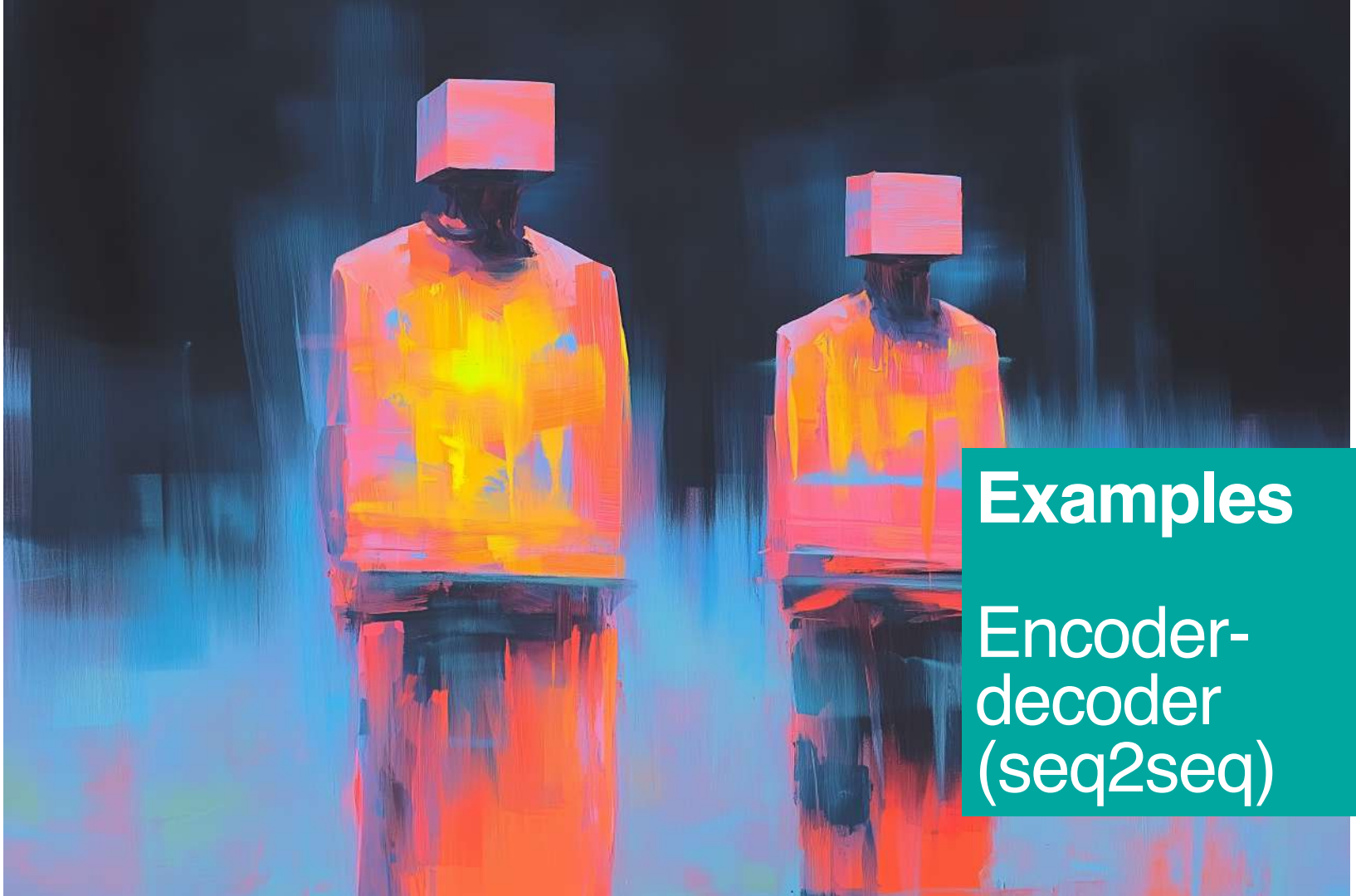
Credit: Taming Transformers for High-Resolution Image Synthesis, Esser et al. 2020

Image generation with vanilla autoregressive models - LLamaGen

Autoregressive language modeling has come a long way and is highly efficient.

- **Idea**: Simply train Llama architecture on next-image-token prediction

Credit: Autoregressive Model Beats Diffusion: Llama for Scalable Image Generation, Sun et al. 2024



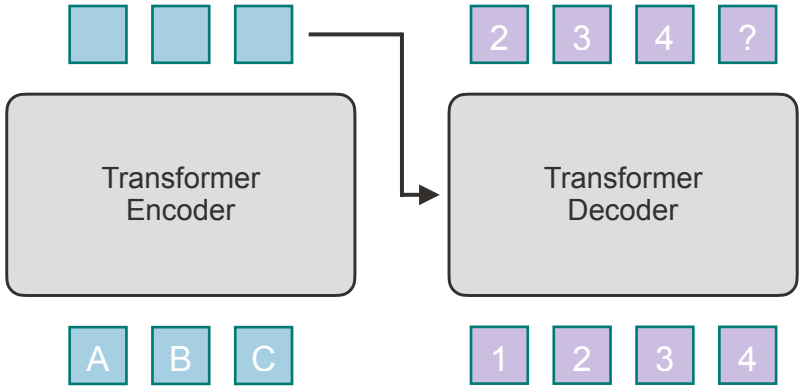
Examples

Encoder-decoder (seq2seq)

Encoder-Decoder (sequence-to-sequence)

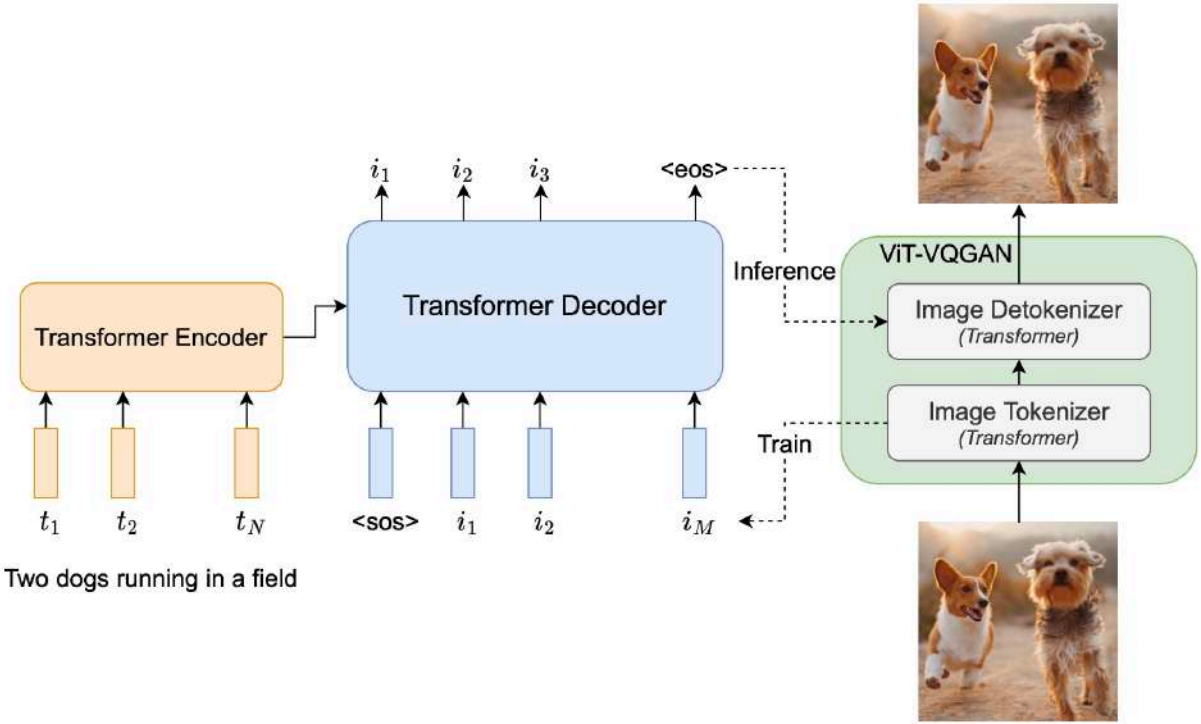
Parti, DETR, T5, 4M, Unified-IO, …

- Sequence-to-sequence (e.g. Parti)
- Sequence-to-set (e.g. DETR)
- Hybrid sequence modeling + masking (e.g. T5, 4M, Unified-IO)



Parti

Scaling autoregressive image modeling



Credit: Scaling Autoregressive Models for Content-Rich Text-to-Image Generation, Yu et al. 2022

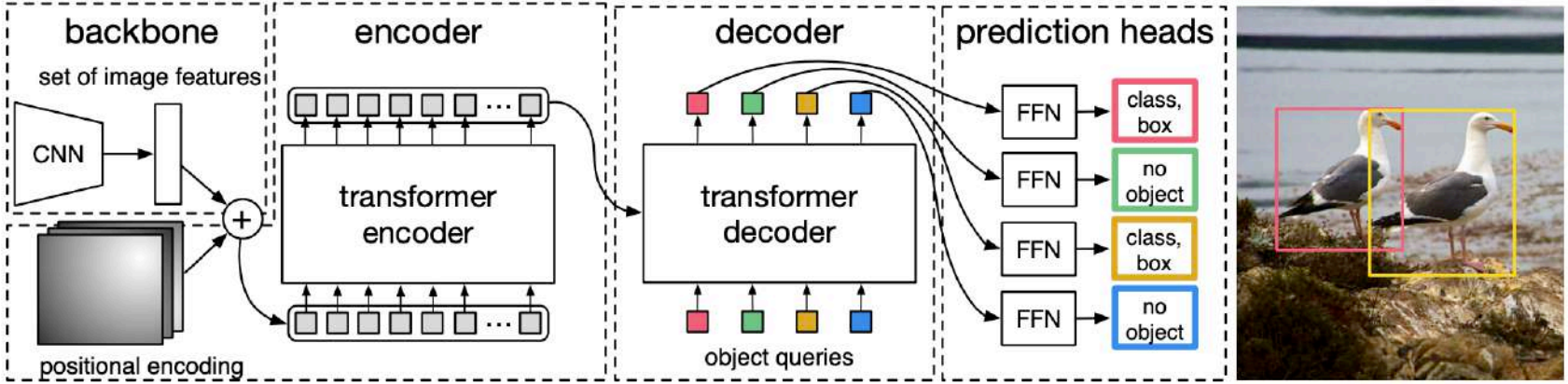
Parti

Scaling autoregressive image modeling

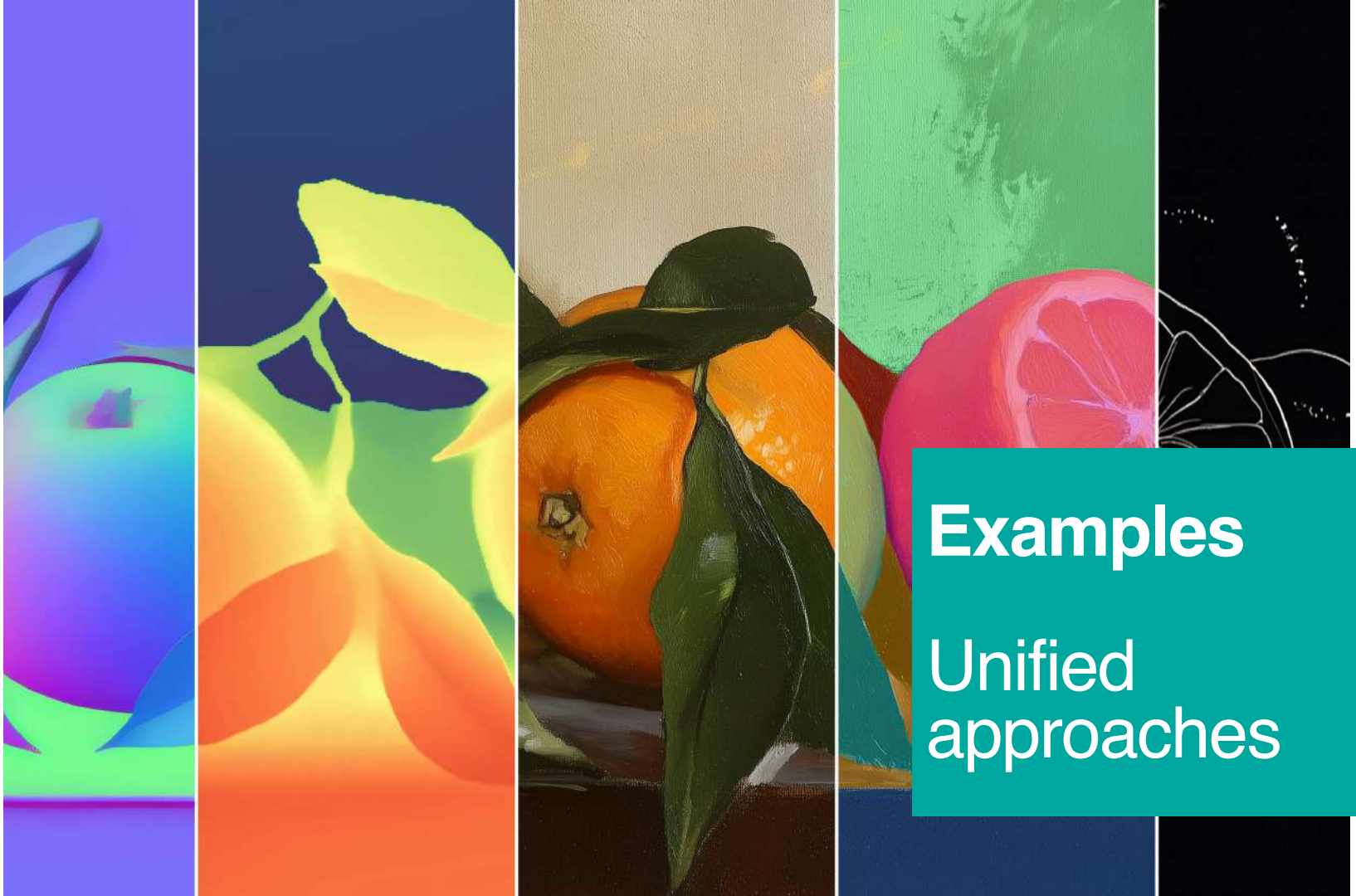
Credit: Scaling Autoregressive Models for Content-Rich Text-to-Image Generation, Yu et al. 2022

DETR: DEtection TRansformer

- Object detection & Panoptic segmentation (instance + semantic + detection) using transformers
- Approach as a set prediction problem with set-based loss
- Decoder directly outputs set in parallel. Queries can “communicate” with each other to avoid overlaps.



Credit: End-to-End Object Detection with Transformers, Carion et al. 2020

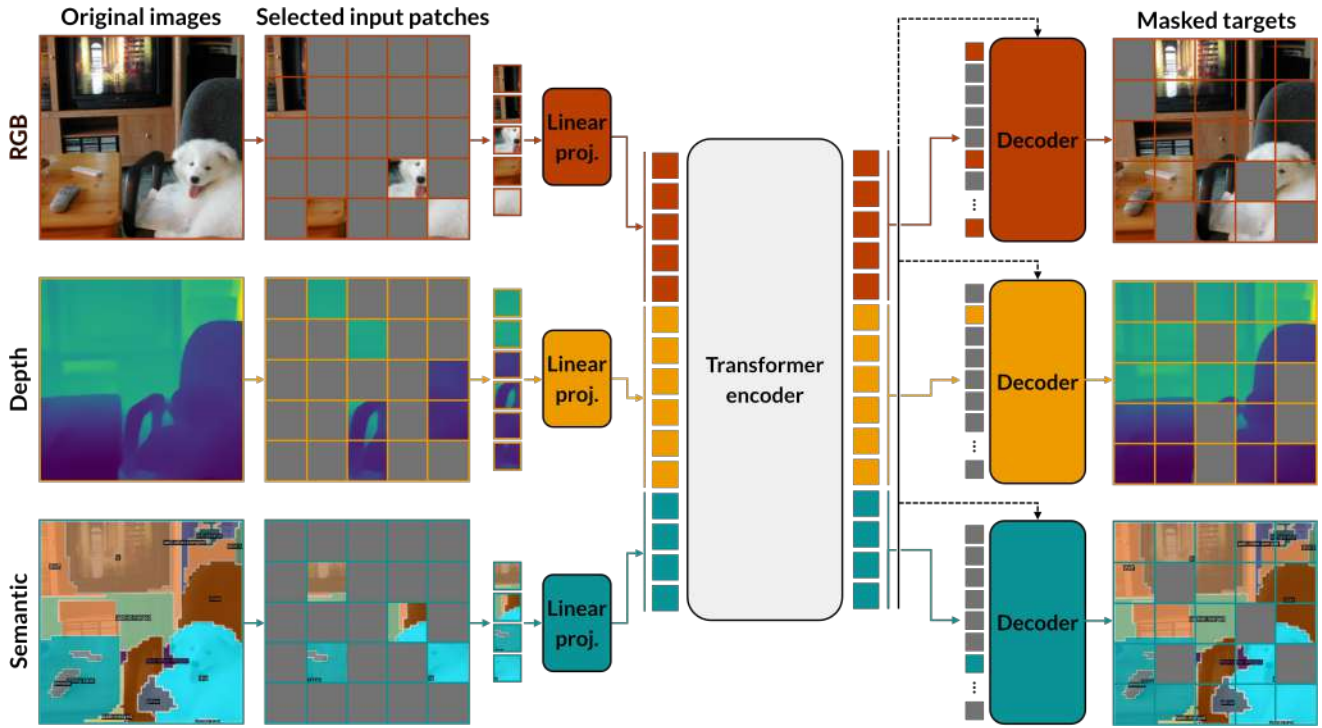


Examples

Unified approaches

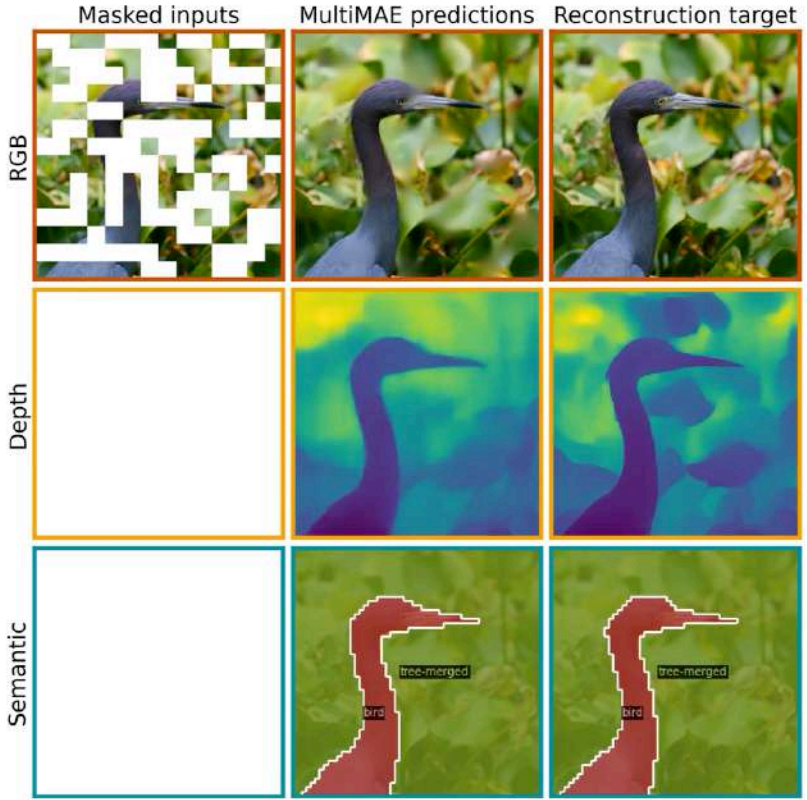
MultiMAE

- **Idea**: Predict any patch of any modality from any other



MultiMAE: Multi-modal Multi-task Masked Autoencoders, Bachmann et al. 2022

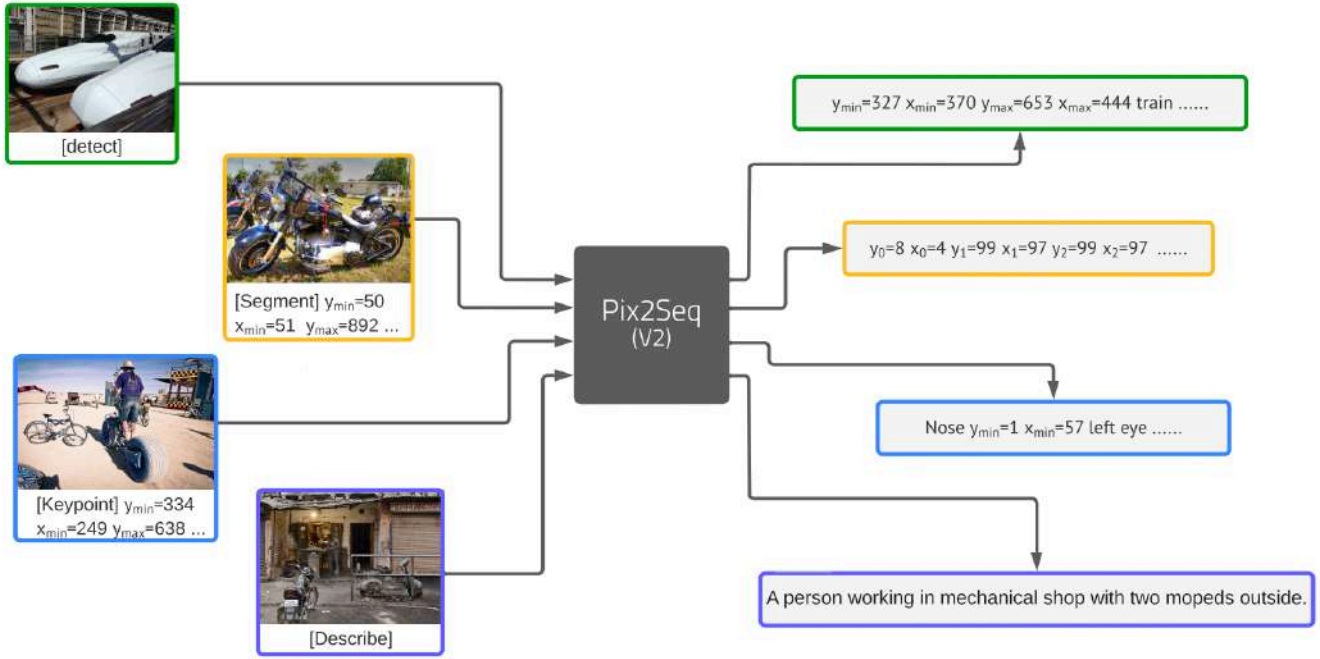
MultiMAE



MultiMAE: Multi-modal Multi-task Masked Autoencoders, Bachmann et al. 2022

Pix2Seq (V2)

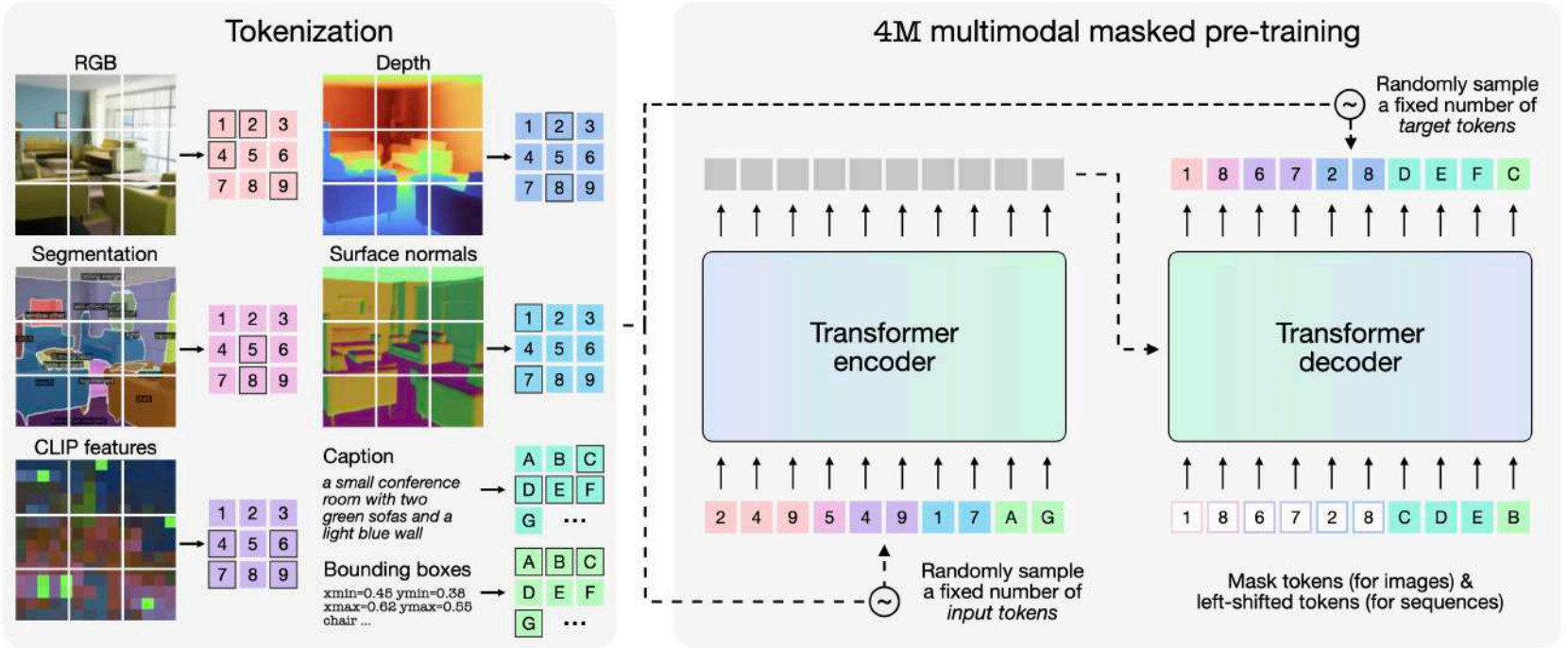
- **Idea**: Tokenize detection / segmentation / keypoints / etc as “text”



Credit: A Unified Sequence Interface for Vision Tasks, Chen et al. 2022

4M

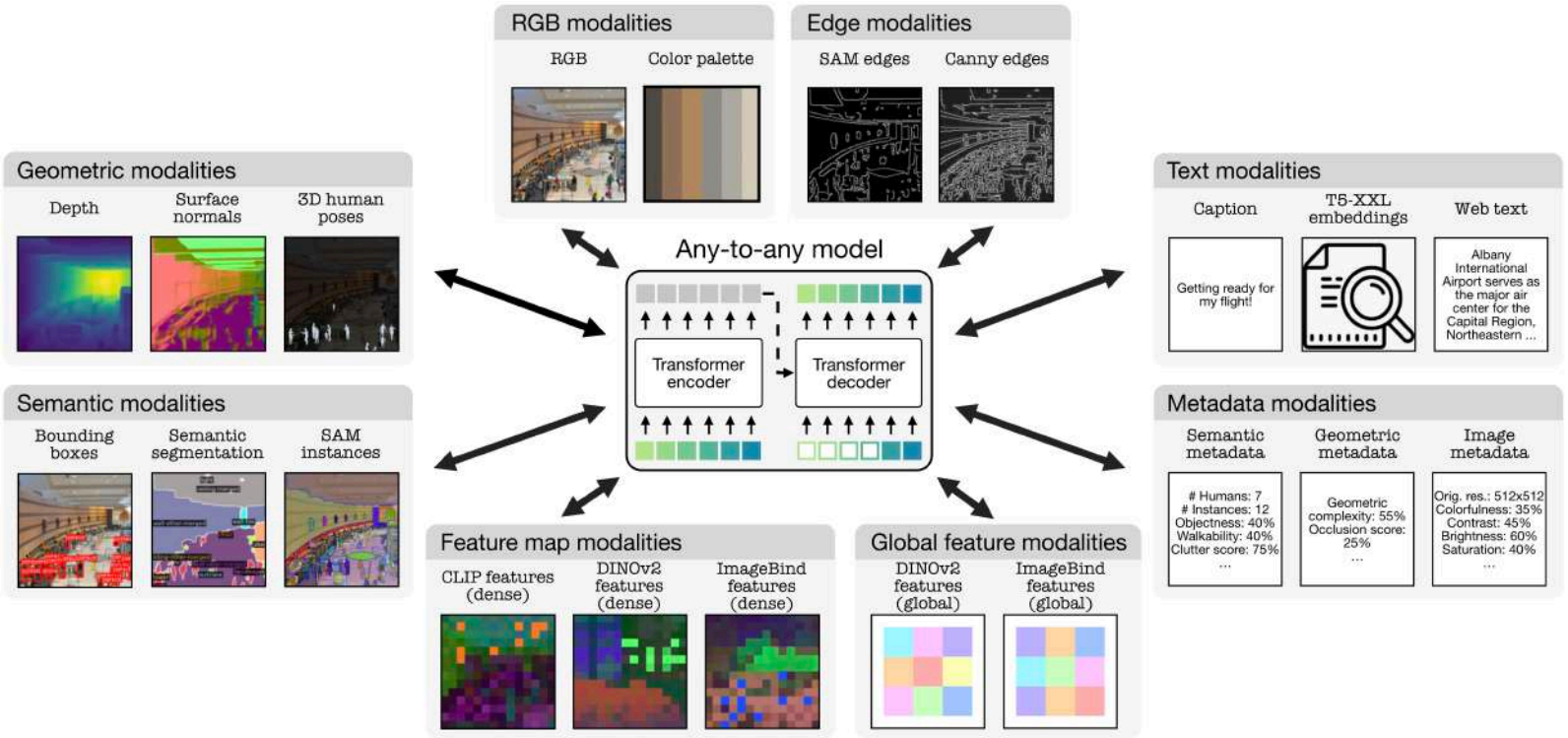
- **Idea**: Any-to-any prediction in unified representation space for many tasks through tokenization



Credit: 4M: Massively Multimodal Masked Modeling, Mizrahi et al. 2023

4M

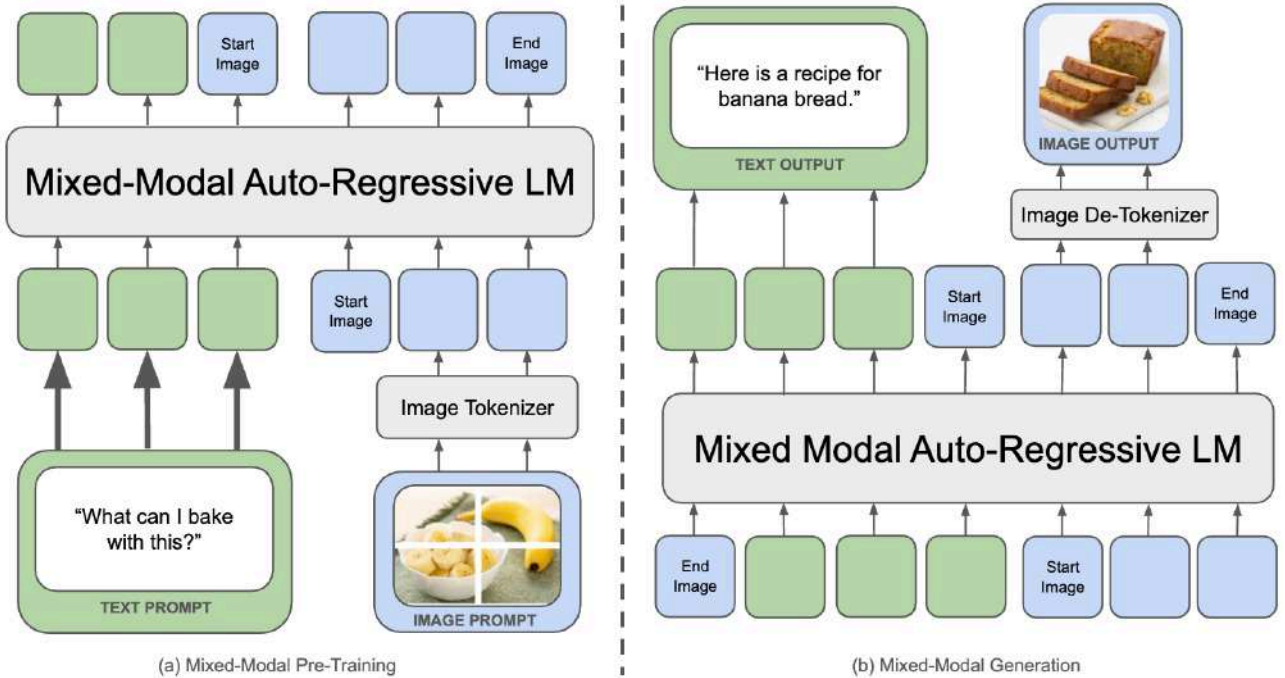
- **Idea**: Any-to-any prediction in unified representation space for many tasks through tokenization



Credit: 4M-21: An Any-to-Any Vision Model for Tens of Tasks and Modalities, Bachmann et al. 2024

Any-to-any vision-language models

- **Idea**: AR model predicting both text and image tokens



Credit: Chameleon: Mixed-Modal Early-Fusion Foundation Models, Chameleon Team. 2024

Enjoy the Course!

Amir Zamir (amir.zamir@epfl.ch)

Rishubh Singh (rishubh.singh@epfl.ch head TA)
Zhitong Gao (zhitong.gao@epfl.ch)
Roman Bachmann (roman.bachmann@epfl.ch)

https://vilab.epfl.ch/