Introduction to Transformers with a focus on CV

43rd VDLM 01.12.2021

Michael Pieler
ML Research Engineer
michael@contextflow.com

Overview

1. Introduction
    a. Why is a Transformer interesting?
    b. What is a Transformer on high level?
2. Building blocks
    a. Transformer block
    b. Attention layer
    c. Feedforward layer
    d. Positional encoding
3. Selected CV applications
4. Code
5. Summary & Outlook

Why is a Transformer interesting?

- Emerged in NLP:
  "The main point of the transformer was to overcome the problems of the previous state-of-the-art architecture, the RNN (usually an LSTM or a GRU)."

- "The rest of the design of the transformer is based primarily on one consideration: depth. Most choices follow from the desire to train big stacks of transformer blocks." [1]
  ➜ Performance! [2][3]

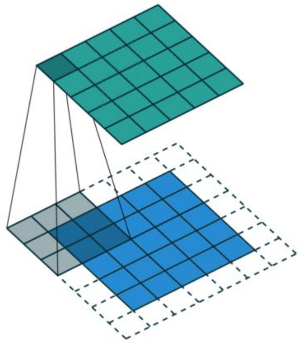
- Transfers well to other data modalities, e.g., images [4], etc.!

- No inductive / learning bias like a CNN. [3]

[1] http://peterbloem.nl/blog/transformers
[2] https://paperswithcode.com/sota/image-classification-on-imagenet,
[3] https://iaml-it.github.io/posts/2021-04-28-transformers-in-vision/
[4] "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", https://arxiv.org/abs/2010.11929

Why is a Transformer (maybe) not interesting?

- No inductive / learning bias like a CNN [3]:
  locality & spatial translation invariance



  - Needs more data than a CNN or special setups.

    ➔ Can be tricky for specific applications,
      but there are solutions to that (see examples).

[3] https://iaml-it.github.io/posts/2021-04-28-transformers-in-vision

What is a Transformer on high level?

- “Any architecture designed to process a connected set of units - such as the tokens in a sequence or the pixels in an image - where the only interaction between units is through self-attention.” [1]

- Attention models make activations depend on the pairwise similarities between activation vectors. [5]

- This contrasts with earlier neural nets that only made activations depend on the similarity between a activation vector and a weight vector, i.e., MLP. [5]

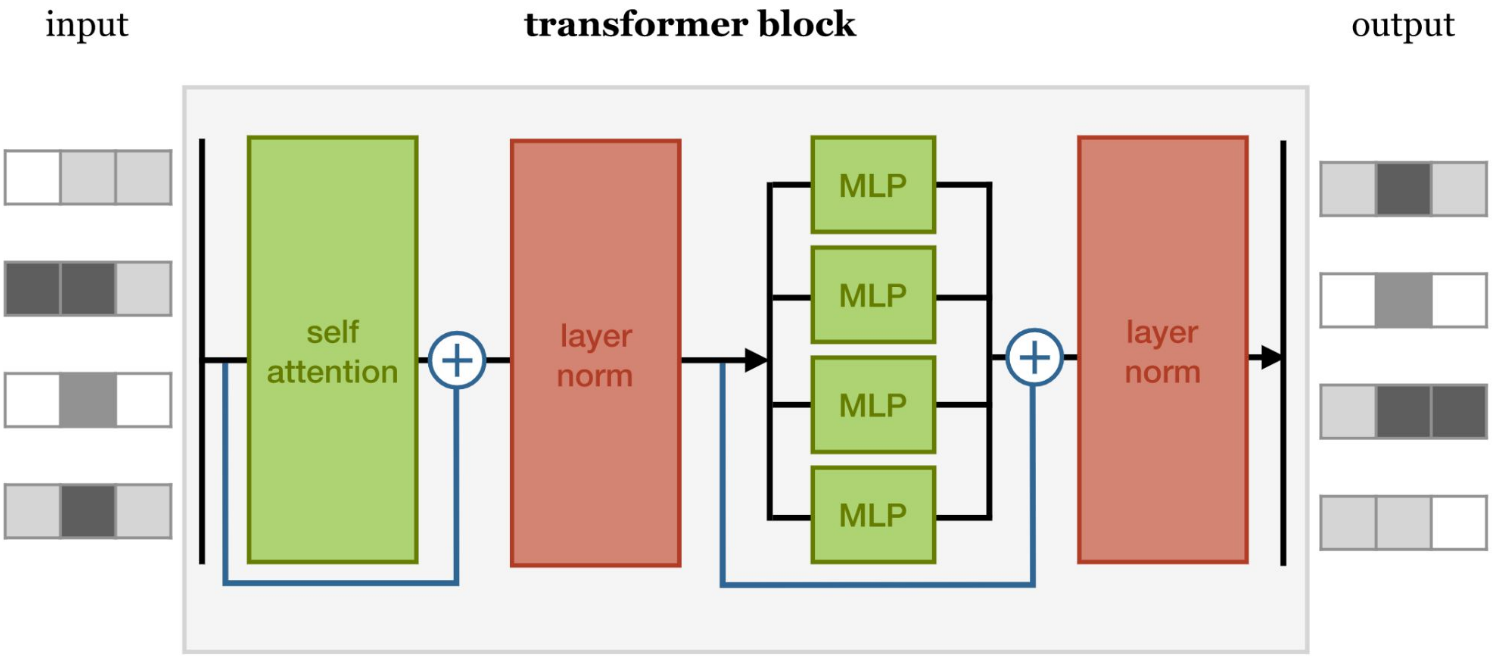
[1] http://peterbloem.nl/blog/transformers

[5] https://www.youtube.com/watch?v=eEXnJOHQ_Xw, 10m 45s

Overview

1. Introduction
    a. Why is a Transformer interesting?
    b. What is a Transformer on high level?
2. Building blocks
    a. Transformer block
    b. Attention layer
    c. Feedforward layer
    d. Positional encoding
3. Selected CV applications
4. Code
5. Summary & Outlook

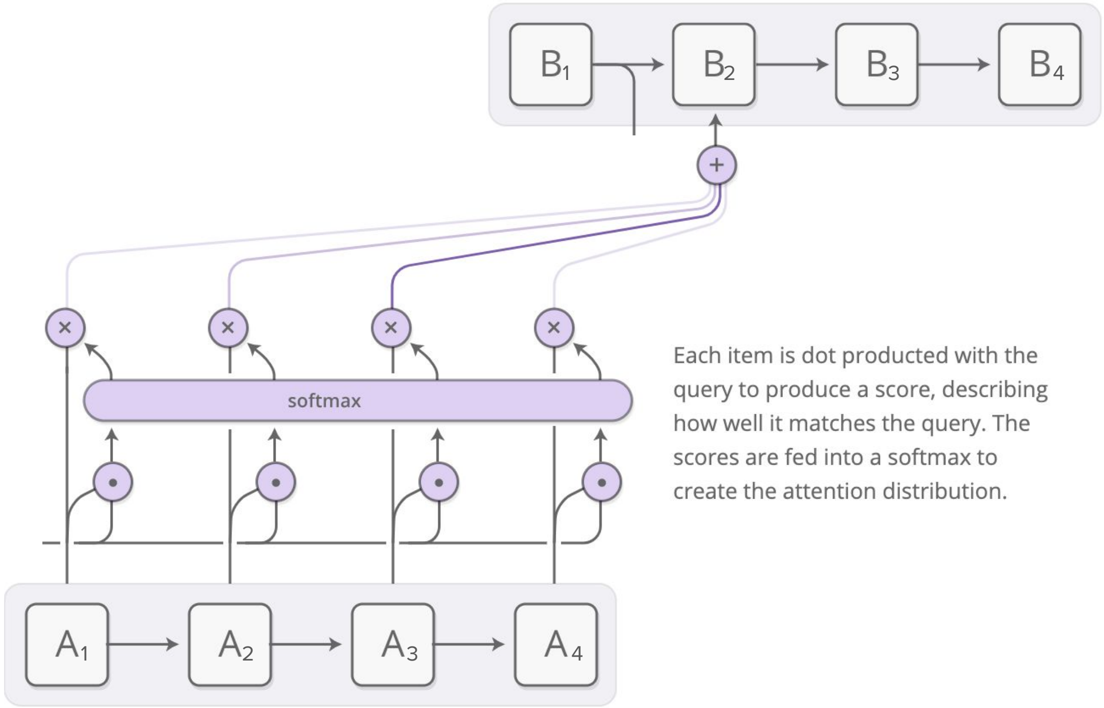
Transformer block



[1] http://peterbloem.nl/blog/transformers

Attention layer

Self-attention is a set-to-set operation:
a set of vectors goes in, and a set of vectors comes out.*



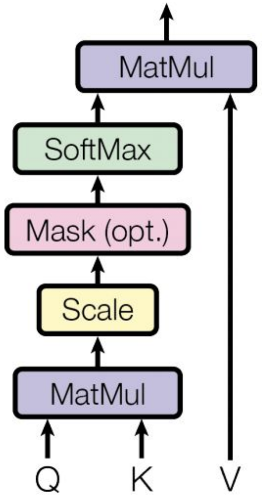
Each item is dot producted with the query to produce a score, describing how well it matches the query. The scores are fed into a softmax to create the attention distribution.

[6] https://distill.pub/2016/augmented-rnns/#attentional-interfaces ← Show this!

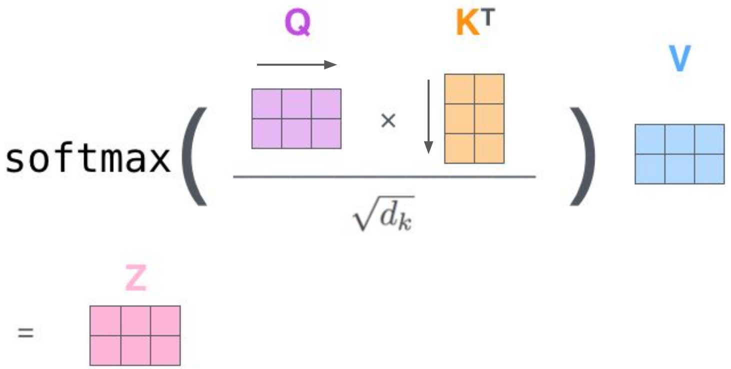
* w/o positional encodings/embeddings, or autoregressive setups we only get a set operation, see slide 15 for details

Attention layer details

Scaled Dot-Product Attention



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



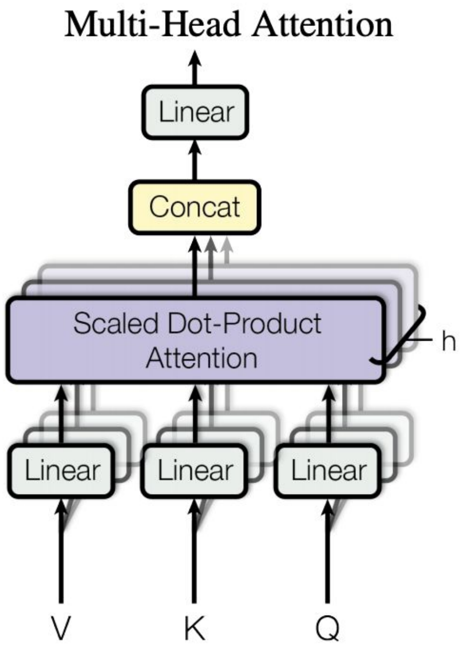
[7] “Attention Is All You Need”, https://arxiv.org/abs/1706.03762

[8] https://jalammar.github.io/illustrated-transformer/

[9] Attention step-by-step notebook: https://github.com/MicPie/pytorch/blob/master/attention.ipynb ← Try this on your own.

Attention layers rely on three tricks

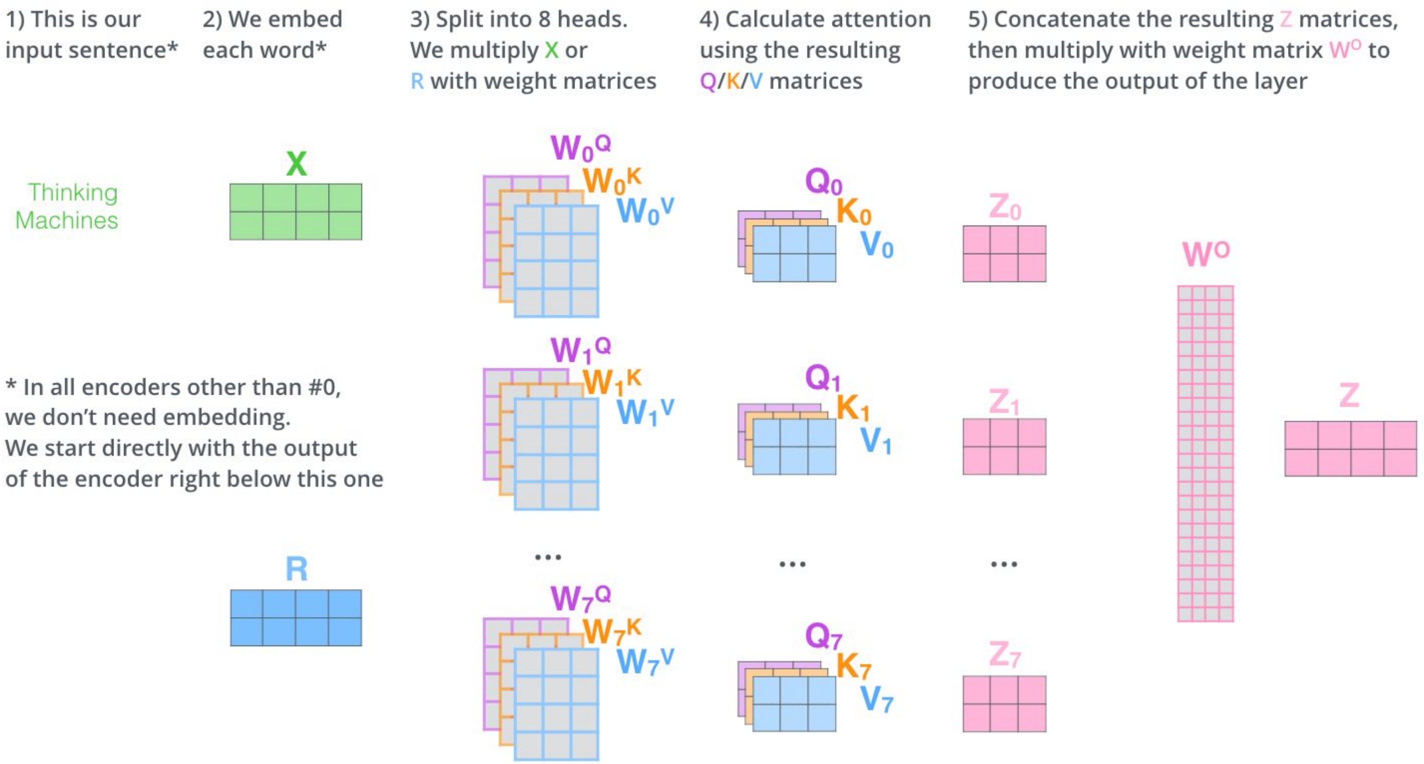
1. Query, key and value projections
2. Scaling the dot product
3. Multi-head attention



[1] http://peterbloem.nl/blog/transformers

[7] "Attention Is All You Need", https://arxiv.org/abs/1706.03762

Multi-Head Self-Attention



[8] https://jalammar.github.io/illustrated-transformer/

Multi-Head Self-Attention Mathematically 1 [10]

- Input set of N vectors: $x_i \in \mathbb{R}^D \quad \{x_1, \ldots, x_N\}$
- Represented as a matrix: $X \in \mathbb{R}^{N \times D}$
- Multihead self-attention (MSA) consists of M heads where M is chosen to divide D.
- The output of each head is a set of N vectors of dimension D/M where each vector is obtained by taking a weighted average of the input vectors with weights given by a weight matrix W, followed by a linear map:

$$W^V \in \mathbb{R}^{D \times D/M}$$

[10] “LieTransformer - Equivariant self-attention for Lie Groups”, https://arxiv.org/abs/2012.10885

Multi-Head Self-Attention Mathematically 2 [10]

- Using m to index the head (m = 1, ..., M) the output of the m-th head can be written as*:

$$W^{Q,m}, W^{K,m}, W^{V,m} \in \mathbb{R}^{D \times D/M}$$

$$XW^{Q,m}(XW^{K,m})^\top \in \mathbb{R}^{N \times N}$$

- The softmax normalisation is performed on each row of the matrix:

$$W = \mathrm{softmax}\left(XW^{Q,m}(XW^{K,m})^\top\right) \in \mathbb{R}^{N \times N}$$

$$f^m(X) = WXW^{V,m} \in \mathbb{R}^{N \times D/M}$$

[10] "LieTransformer - Equivariant self-attention for Lie Groups", https://arxiv.org/abs/2012.10885

* scaling by 1/sqrt(d) and addition of bias is not shown

Multi-Head Self-Attention Mathematically 3 [10]

$$W = \mathrm{softmax}\left(XW^{Q,m}(XW^{K,m})^\top\right) \in \mathbb{R}^{N\times N}$$

$$f^m(X) = WXW^{V,m} \in \mathbb{R}^{N\times D/M}$$

- Finally, the outputs of all heads are concatenated into a N x D matrix and then right multiplied by:

$$W^O \in \mathbb{R}^{D\times D}$$

$$\mathrm{MSA}(X) = [f^1(X), \ldots, f^M(X)]W^O \in \mathbb{R}^{N\times D}$$

[10] “LieTransformer - Equivariant self-attention for Lie Groups”, https://arxiv.org/abs/2012.10885

Feedforward layer

$$\mathrm{FFN}\left(\mathbf{H}'\right) = \texttt{Linear}(\texttt{ReLU}(\texttt{Linear}(\mathbf{H}')))$$

$$\mathrm{FFN}\left(\mathbf{H}'\right) = \texttt{ReLU}\left(\mathbf{H}'\mathbf{W}^1 + \mathbf{b}^1\right)\mathbf{W}^2 + \mathbf{b}^2$$

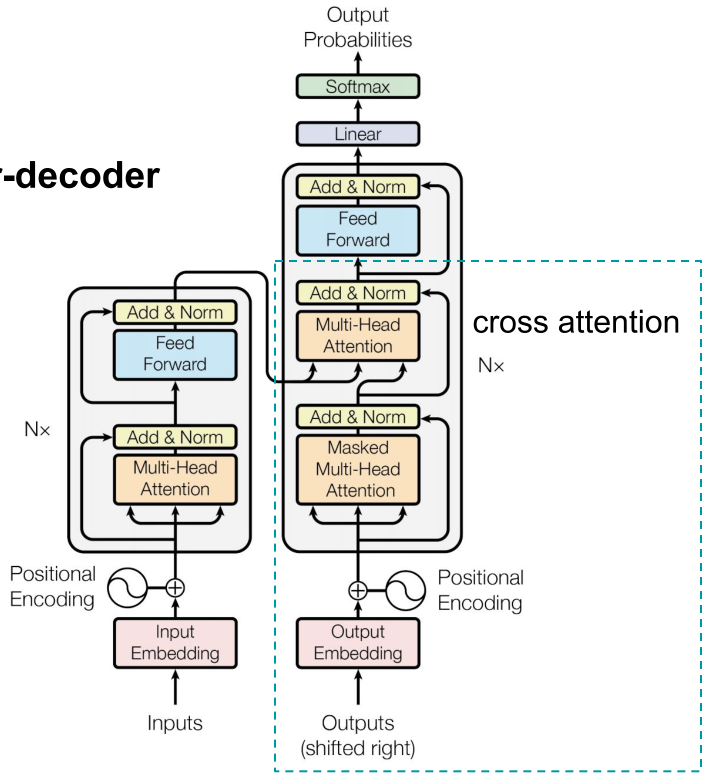
- Typically the intermediate dimension of the FF layer is set to be larger than the input/output dimensions.

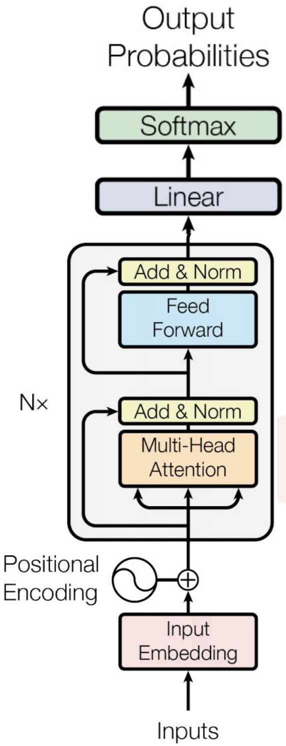
- Difference between attention and FF layer?
  ➜ Two FF layers = "attention over parameters"!

[11] "A Survey of Transformers", https://arxiv.org/abs/2106.04554

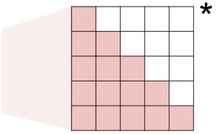
Encoder-decoder, encoder, and decoder?

Encoder-decoder



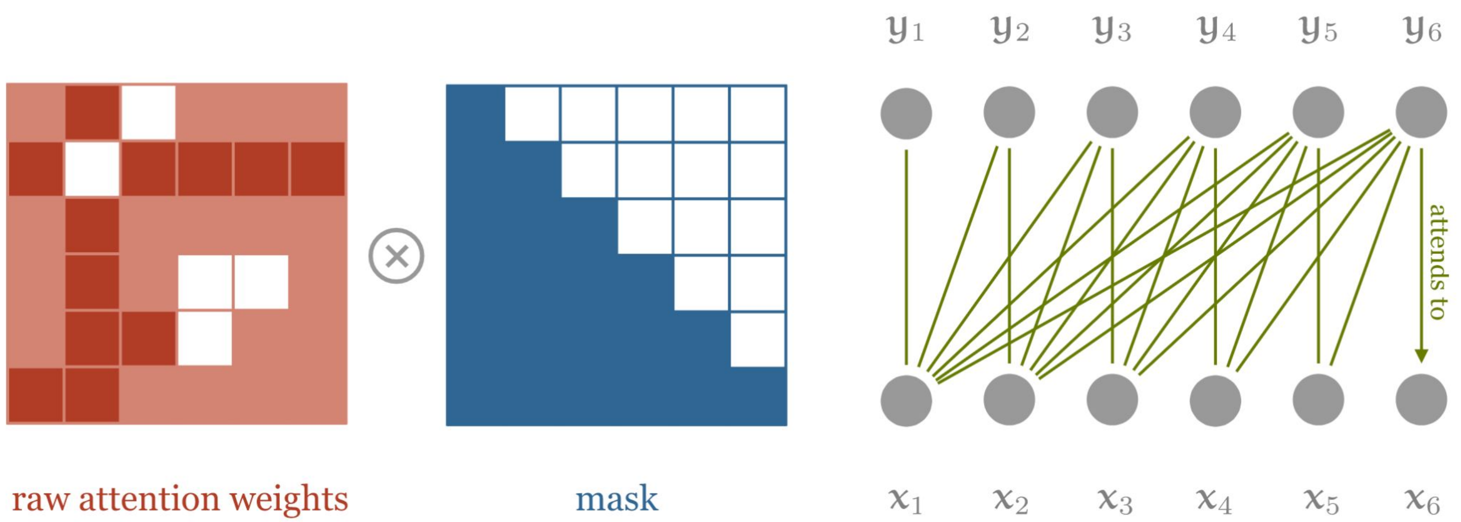


Encoder (w/o masking)
Decoder (w/ masking)*



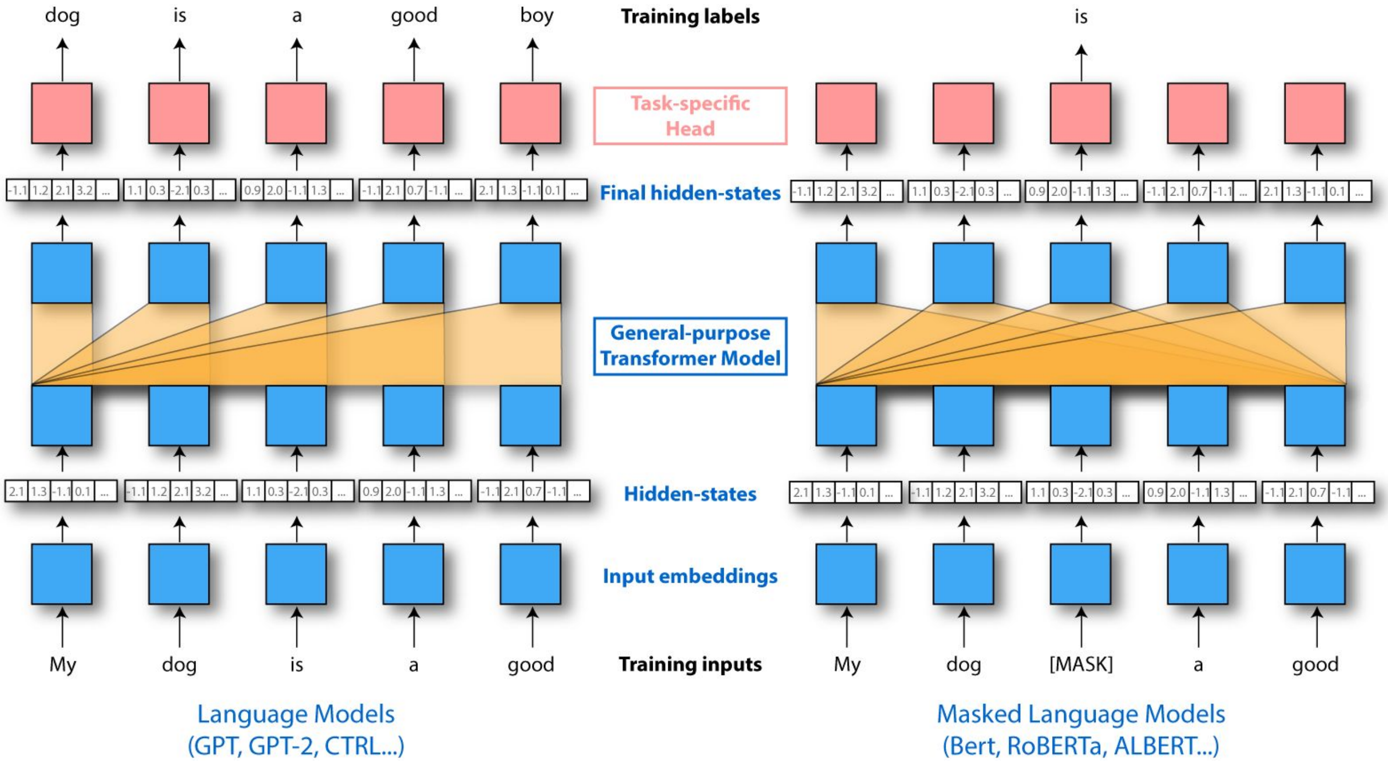
[7] "Attention Is All You Need", https://arxiv.org/abs/1706.03762

LM / decoder / autoregressive / causal masking



[1] http://peterbloem.nl/blog/transformers

Language models (LM) vs. masked lang. models (MLM)



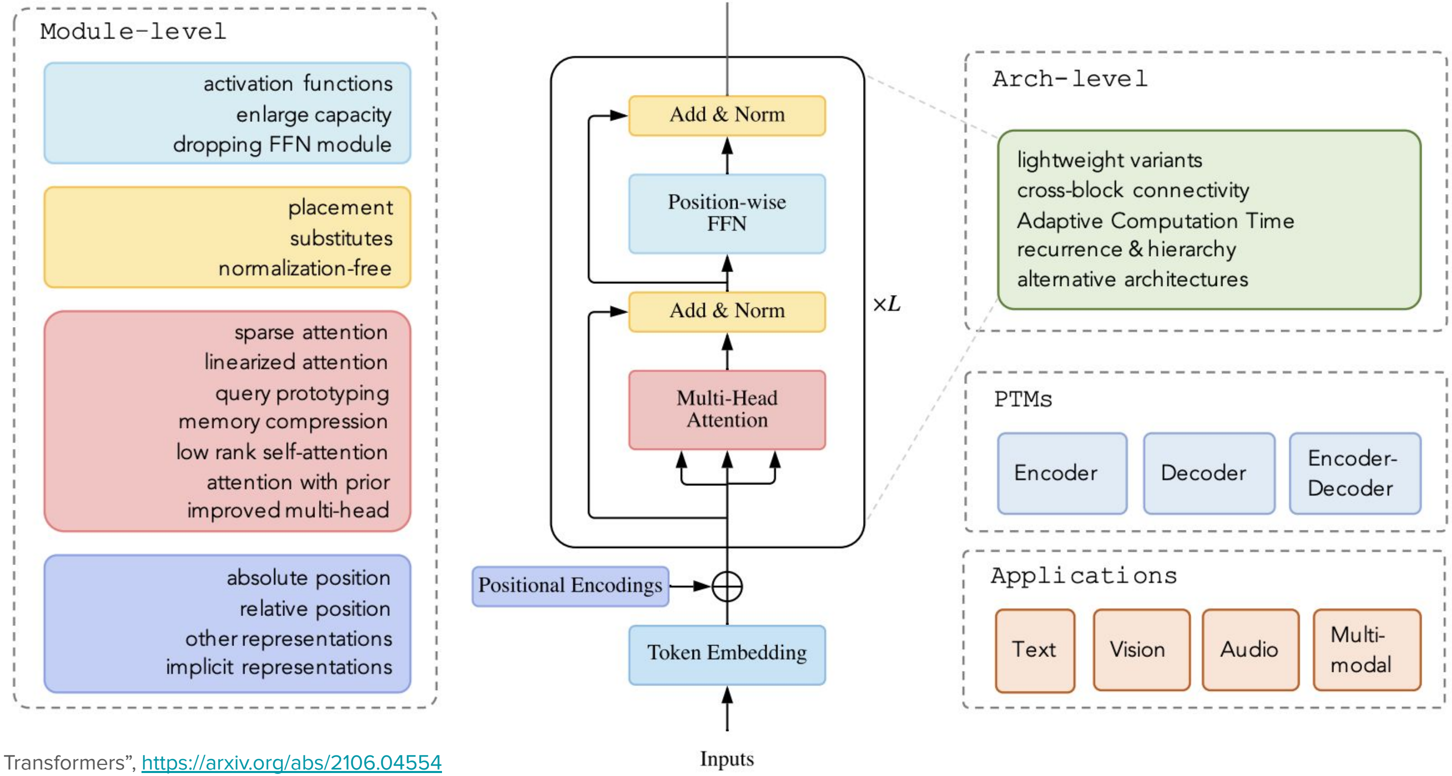
[12] https://twitter.com/thom_wolf/status/1186225108282757120?s=21

Positional encodings, embeddings & Co.

- Is needed because otherwise the Transformer has no notion of the position or distance between the tokens, i.e., set operation. [13]
- Can be an encoding or be trained, i.e., embedding.
- Recent work shows that it is not needed for autoregressive setups, because the model is able to pick up positional information from the attention mask. [14]
- They're necessary for MLM (masked language model) and encoder-decoder models because there is no positional information contained in the attention mask.
- A lot of recent developments, e.g., rotary embeddings [15], ALiBi [16], etc.!

[13] https://towardsdatascience.com/master-positional-encoding-part-i-63c05d90a0c3, [14] "Language Modeling with Deep Transformers", https://arxiv.org/abs/1905.04226
[15] https://blog.eleuther.ai/rotary-embeddings/, [16] "Train Short, Test Long - Attention with Linear Biases Enables Input Length Extrapolation", https://ofir.io/train_short_test_long.pdf

A lot of Transformer variants



[11] "A Survey of Transformers", https://arxiv.org/abs/2106.04554

[17] https://lilianweng.github.io/lil-log/2020/04/07/the-transformer-family.html

[18] "Transformers from Scratch", https://e2eml.school/transformers.html ← Recent detailed intro!

Overview

1. Introduction
    a. Why is a Transformer interesting?
    b. What is a Transformer on high level?

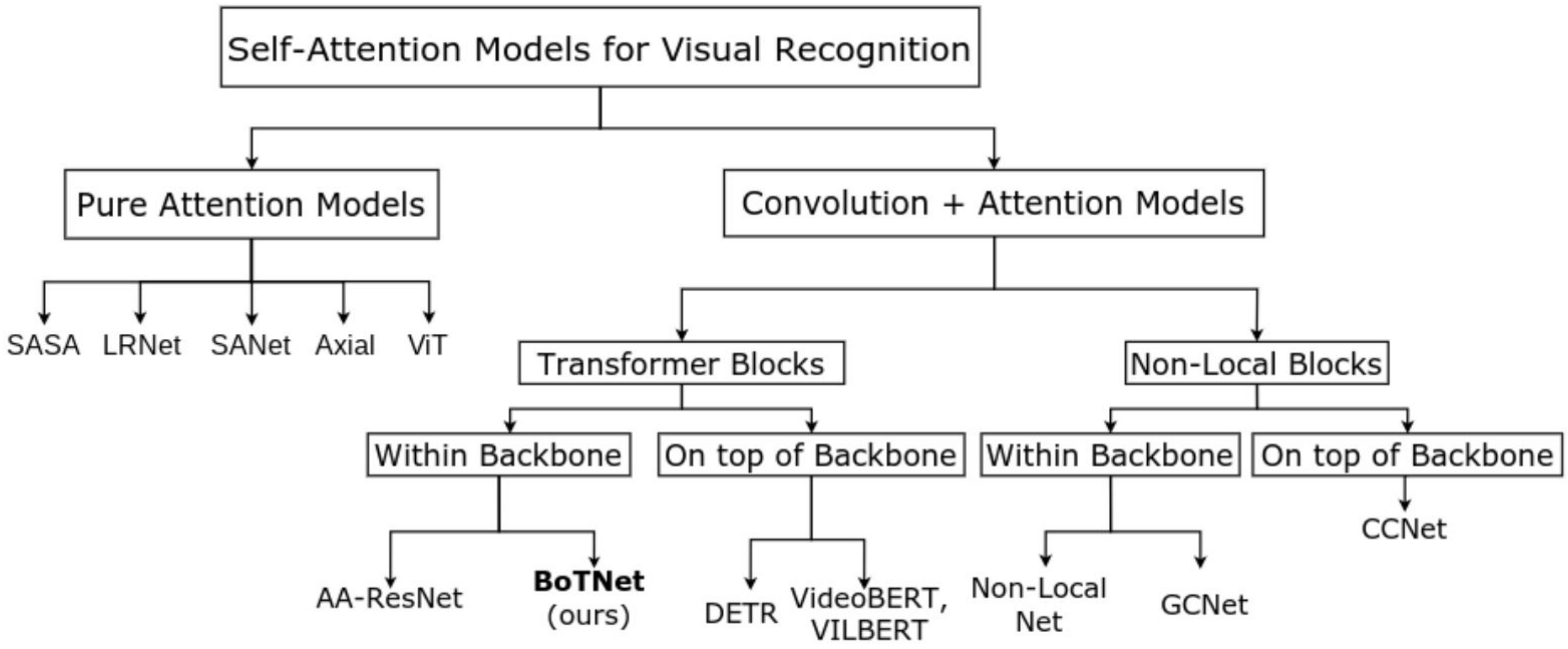
2. Building blocks
    a. Transformer block
    b. Attention layer
    c. Feedforward layer
    d. Positional encoding

3. Selected CV applications

4. Code

5. Summary & Outlook

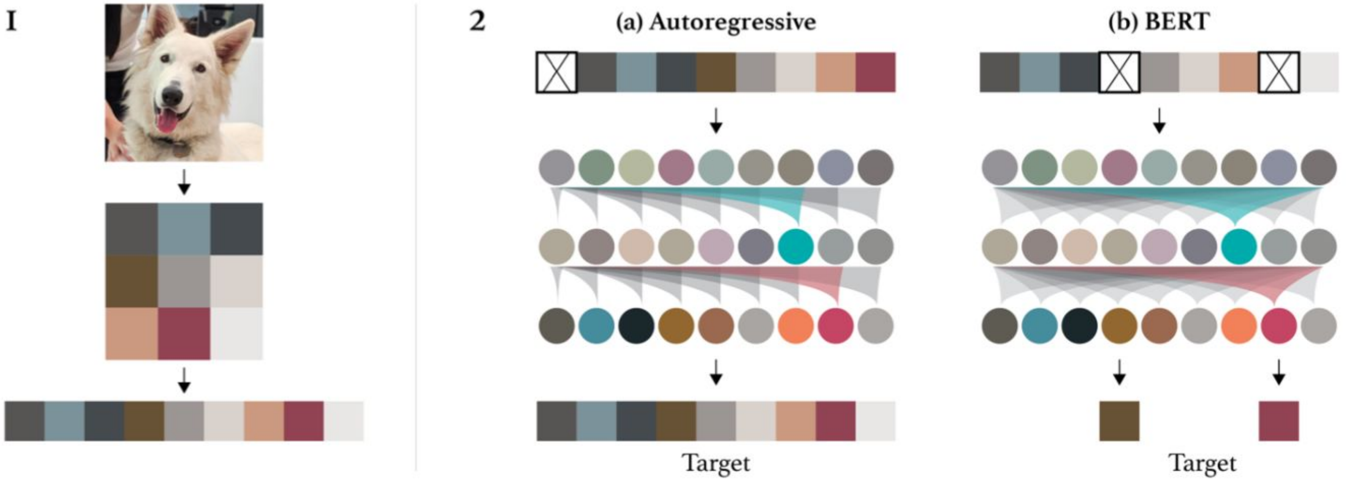
Computer vision setups



[19] "Bottleneck Transformers for Visual Recognition", https://arxiv.org/abs/2101.11605

[3] https://iaml-it.github.io/posts/2021-04-28-transformers-in-vision/ ← Show this!

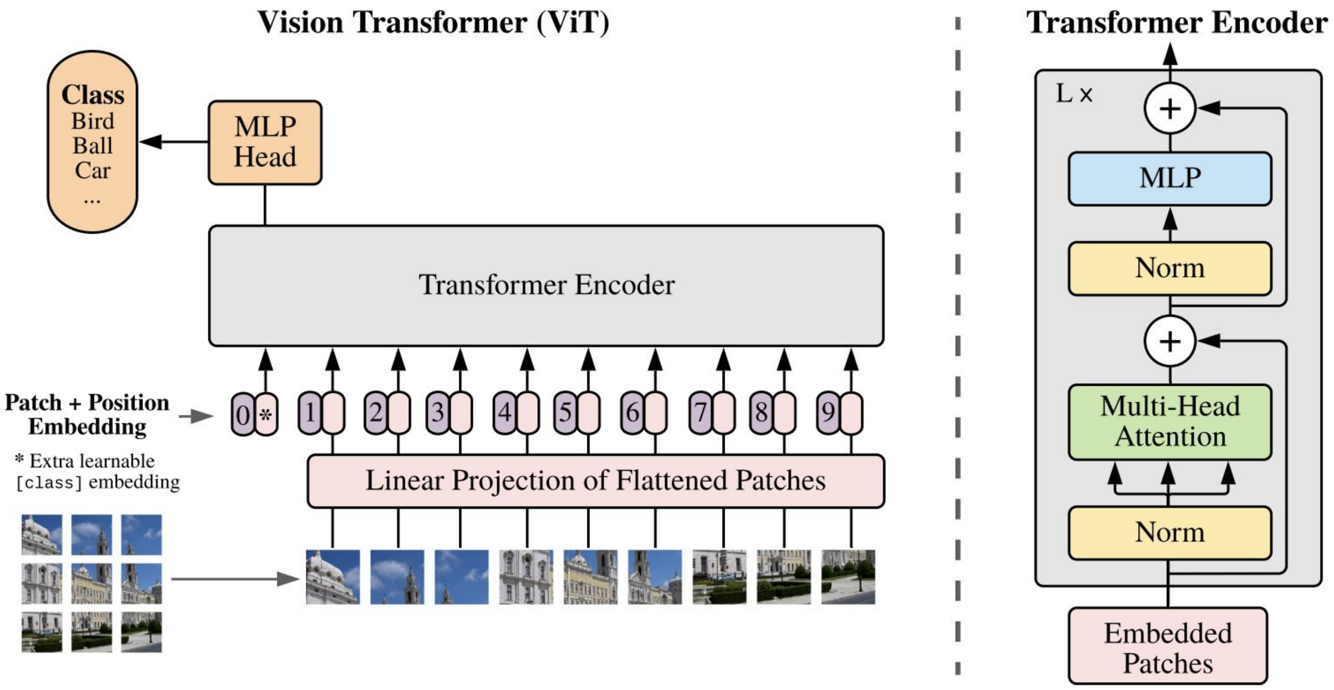
Example: Image GPT (iGPT)



- But "iGPT-L has 2 to 3 times as many parameters as similarly performing models on ImageNet and uses more compute." [19]

[20] "Generative Pretraining from Pixels", https://openai.com/blog/image-gpt/ ← Show this!

Example: Vision Transformer (ViT)



[4] "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", https://arxiv.org/abs/2010.11929
[21] https://ai.googleblog.com/2020/12/transformers-for-image-recognition-at.html

Example: ViT model sizes

Model	Layers	Hidden size D	MLP size	Heads	Params
ViT-Base	12	768	3072	12	86M
ViT-Large	24	1024	4096	16	307M
ViT-Huge	32	1280	5120	16	632M

Model	**#params**	**Throughput (img/sec/core)**	**Patch Resolution**	**Sequence Length**	**Hidden Size**	**#heads**	**#layers**
ViT-S/32	23M	6888	32×32	49	384	6	12
ViT-S/16	22M	2043	16×16	196	384	6	12
ViT-S/14	22M	1234	14×14	256	384	6	12
ViT-S/8	22M	333	8×8	784	384	6	12
ViT-B/32	88M	2805	32×32	49	768	12	12
ViT-B/16	87M	863	16×16	196	768	12	12

[4] "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", https://arxiv.org/abs/2010.11929

[3] https://iaml-it.github.io/posts/2021-04-28-transformers-in-vision/ ← Show this!

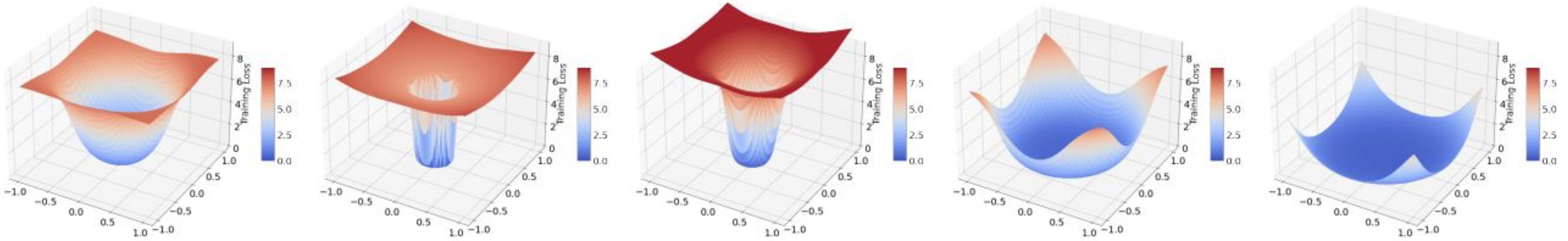
[22] "When Vision Transformers Outperform ResNets without Pretraining or Strong Data Augmentations", https://arxiv.org/abs/2106.01548

Example: ViT can outperform ResNets without pretraining & strong data augmentations

Model	#params	Throughput (img/sec/core)	ImageNet	Real	V2
			ResNet		
ResNet-50-SAM	25M	2161	76.7 (+0.7)	83.1 (+0.7)	64.6 (+1.0)
ResNet-101-SAM	44M	1334	78.6 (+0.8)	84.8 (+0.9)	66.7 (+1.4)
ResNet-152-SAM	60M	935	79.3 (+0.8)	84.9 (+0.7)	67.3 (+1.0)
ResNet-50x2-SAM	98M	891	79.6 (+1.5)	85.3 (+1.6)	67.5 (+1.7)
ResNet-101x2-SAM	173M	519	80.9 (+2.4)	86.4 (+2.4)	69.1 (+2.8)
ResNet-152x2-SAM	236M	356	81.1 (+1.8)	86.4 (+1.9)	69.6 (+2.3)
			Vision Transformer		
ViT-S/32-SAM	23M	6888	70.5 (+2.1)	77.5 (+2.3)	56.9 (+2.6)
ViT-S/16-SAM	22M	2043	78.1 (+3.7)	84.1 (+3.7)	65.6 (+3.9)
ViT-S/14-SAM	22M	1234	78.8 (+4.0)	84.8 (+4.5)	67.2 (+5.2)
ViT-S/8-SAM	22M	333	81.3 (+5.3)	86.7 (+5.5)	70.4 (+6.2)
ViT-B/32-SAM	88M	2805	73.6 (+4.1)	80.3 (+5.1)	60.0 (+4.7)
ViT-B/16-SAM	87M	863	79.9 (+5.3)	85.2 (+5.4)	67.5 (+6.2)
			MLP-Mixer		
Mixer-S/32-SAM	19M	11401	66.7 (+2.8)	73.8 (+3.5)	52.4 (+2.9)
Mixer-S/16-SAM	18M	4005	72.9 (+4.1)	79.8 (+4.7)	58.9 (+4.1)
Mixer-S/8-SAM	20M	1498	75.9 (+5.7)	82.5 (+6.3)	62.3 (+6.2)
Mixer-B/32-SAM	60M	4209	72.4 (+9.9)	79.0 (+10.9)	58.0 (+10.4)
Mixer-B/16-SAM	59M	1390	77.4 (+11.0)	83.5 (+11.4)	63.9 (+13.1)
Mixer-B/8-SAM	64M	466	79.0 (+10.4)	84.4 (+10.1)	65.5 (+11.6)

[22] "When Vision Transformers Outperform ResNets without Pretraining or Strong Data Augmentations", https://arxiv.org/abs/2106.01548

Example: ViT can outperform ResNets with SAM

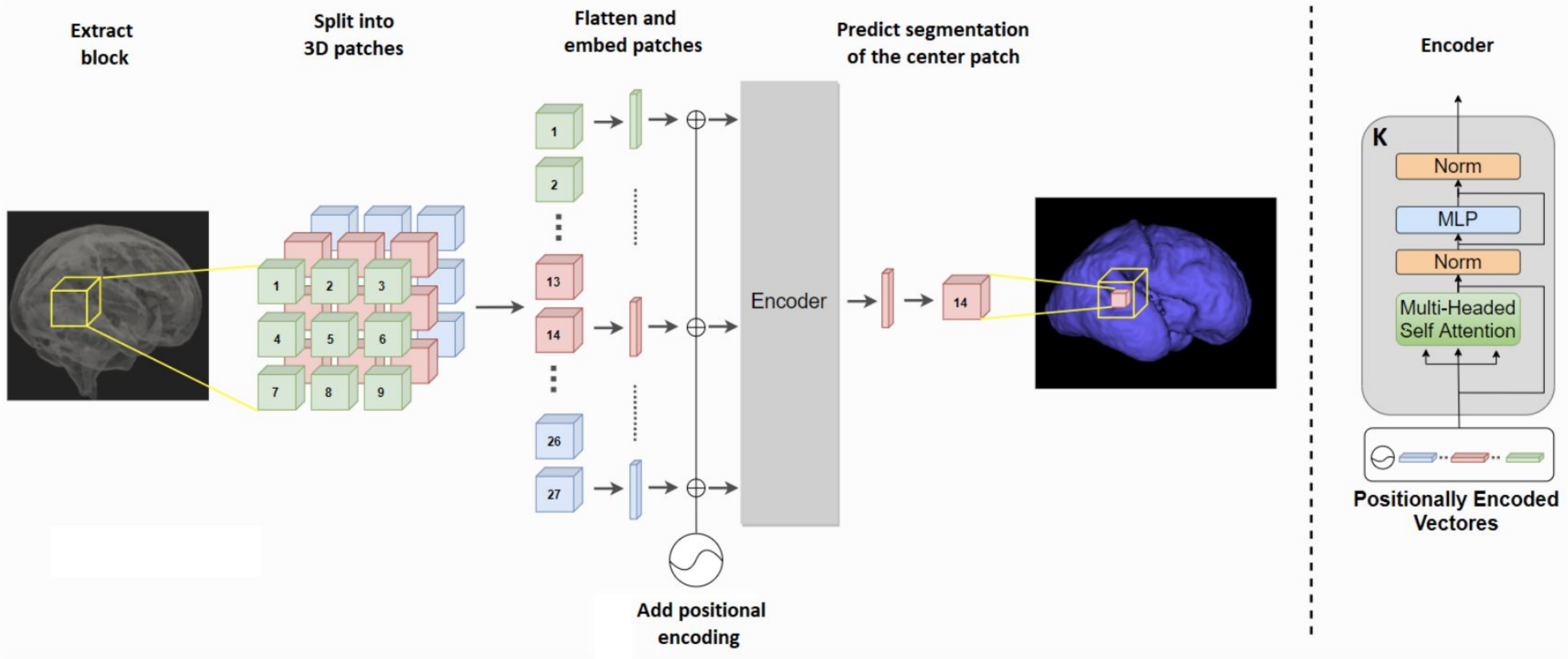


(a) ResNet (b) ViT (c) Mixer (d) ViT-SAM (e) Mixer-SAM

Figure 1: Cross-entropy loss landscapes of ResNet-152, ViT-B/16, and Mixer-B/16. ViT and MLP-Mixer converge to sharper regions than ResNet when trained on ImageNet with the basic Inception-style preprocessing. SAM, a sharpness-aware optimizer, significantly smooths the landscapes.

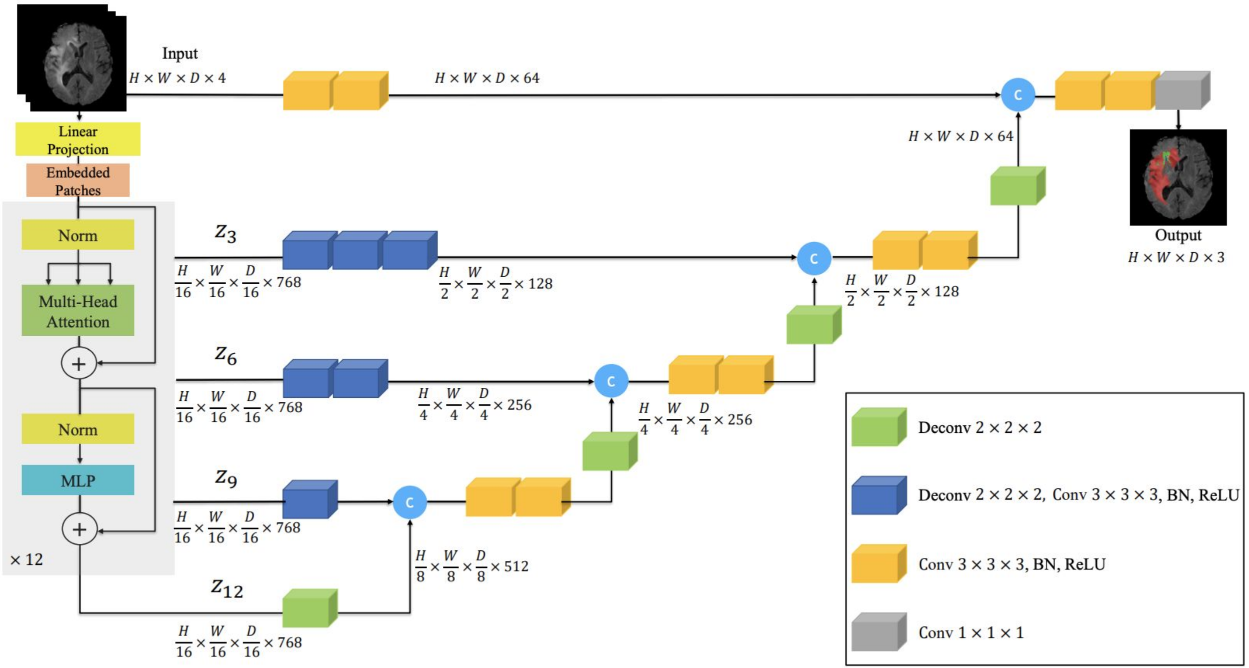
[22] “When Vision Transformers Outperform ResNets without Pretraining or Strong Data Augmentations”, https://arxiv.org/abs/2106.01548

Example: “ViT for segmentation”



[23] “Convolution-Free Medical Image Segmentation using Transformers”, https://arxiv.org/abs/2102.13645

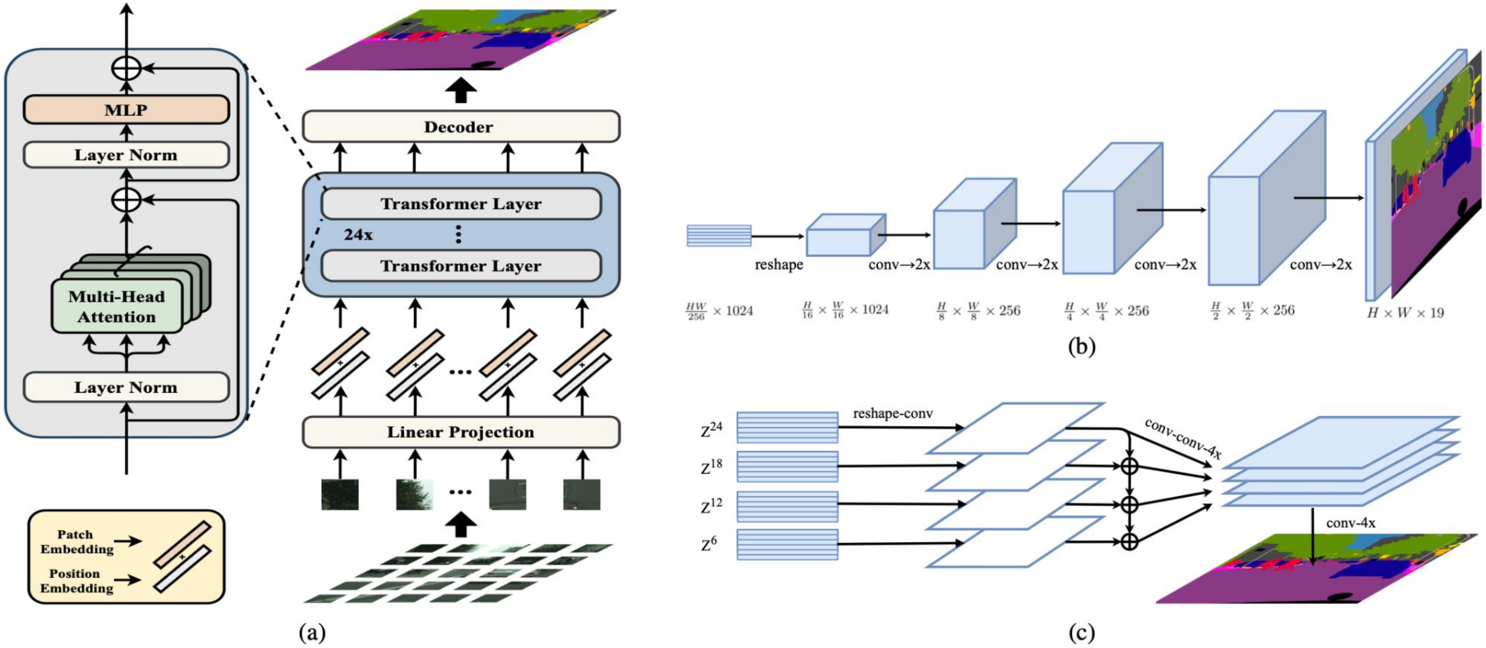
Example: UNEt TRansformers (UNETR)



[24] https://docs.monai.io/en/latest/whatsnew_0_6.html#unetr-transformers-for-medical-image-segmentation

[25] “UNETR: Transformers for 3D Medical Image Segmentation”, https://arxiv.org/abs/2103.10504

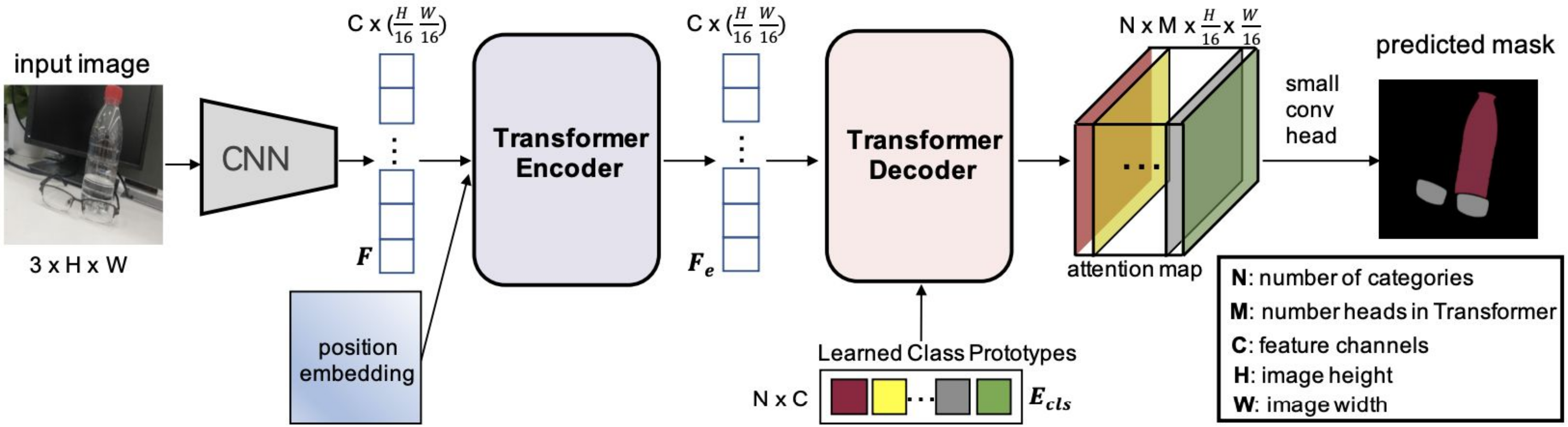
Example: SEgmentation TRansformer (SETR)



(a) (b) (c)

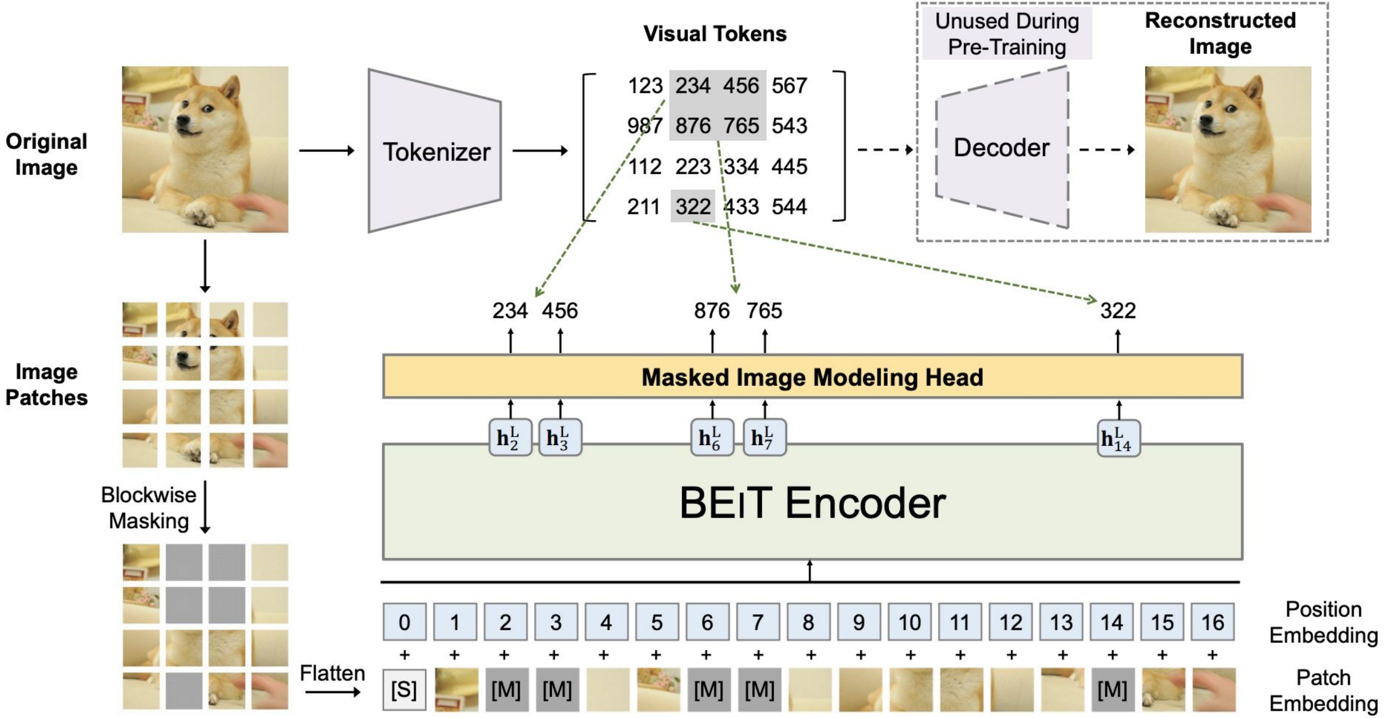
[26] “Rethinking Semantic Segmentation from a Sequence-to-Sequence Perspective with Transformers”, https://arxiv.org/abs/2012.15840

Example: Segmenting transparent objects



[27] "Segmenting Transparent Object in the Wild with Transformer", https://arxiv.org/abs/2101.08461

Example: BERT pre-train. of image Transformers (BEiT)

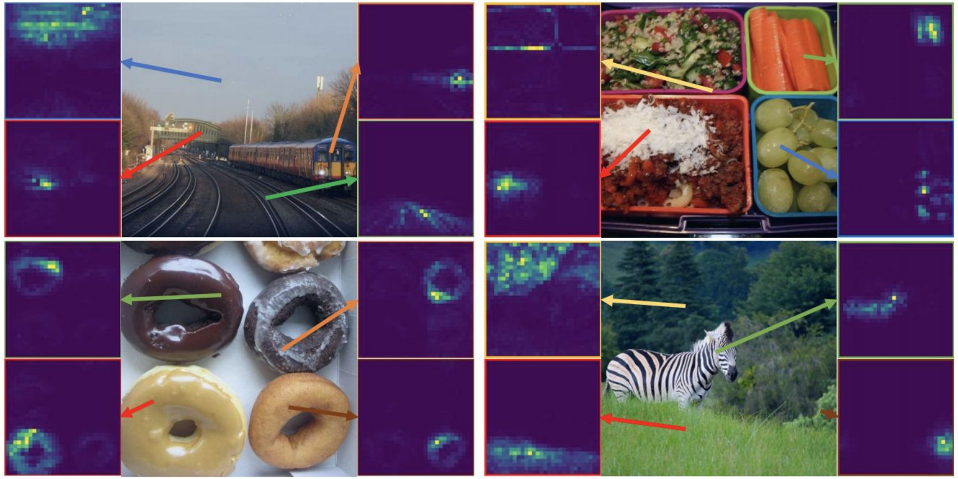


[28] “BEiT: BERT Pre-Training of Image Transformers”, https://arxiv.org/abs/2106.08254

BEiT: All-in-one computer vision models soon?

- BEiT can be (pre)trained:
  - in a self-supervised way
  - with image labels
  - with segmentation labels

“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.” [27]

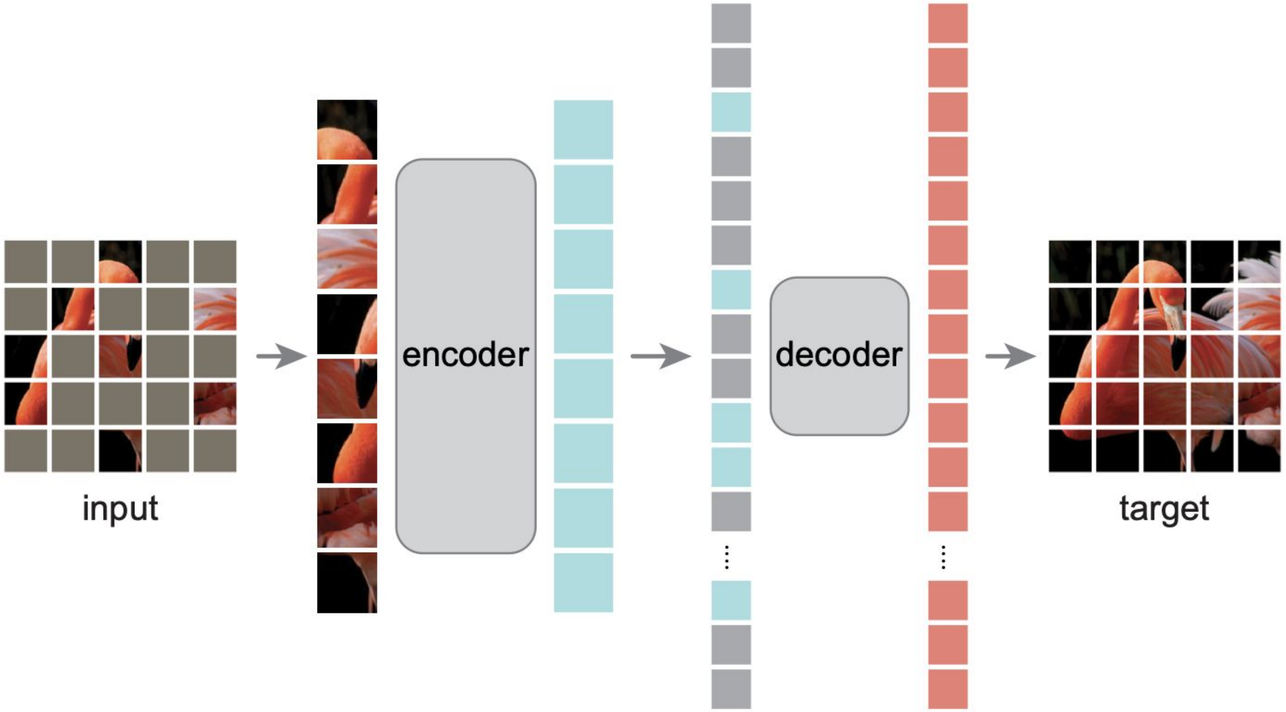


Results of semantic segmentation on ADE20K [21]:

Models	mIoU
Supervised Pre-Training on ImageNet	45.3
DINO (Caron et al., 2021)	44.1
BEIT (ours)	45.6
BEIT + Intermediate Fine-Tuning (ours)	**47.7**

[28] “BEiT: BERT Pre-Training of Image Transformers”, https://arxiv.org/abs/2106.08254

Masked Autoencoders (MAE)

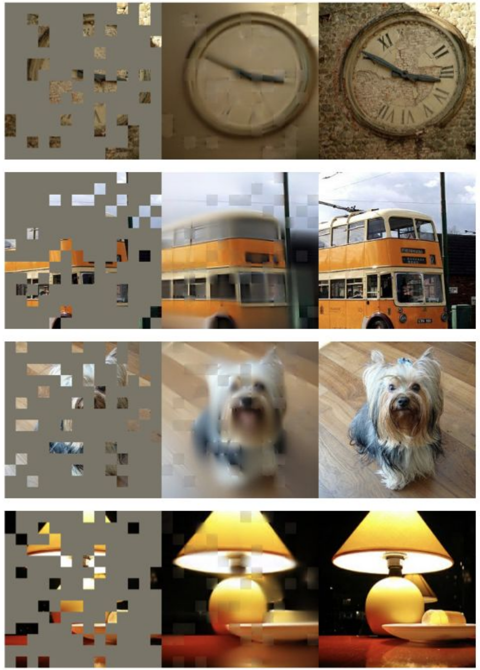


[29] "Masked Autoencoders Are Scalable Vision Learners", https://arxiv.org/abs/2111.06377

MAE

encoder	dec. depth	ft acc	hours	speedup
ViT-L, w/ `[M]`	8	84.2	42.4	-
ViT-L	8	84.9	15.4	2.8×
ViT-L	1	84.8	11.6	**3.7**×
ViT-H, w/ `[M]`	8	-	119.6†	-
ViT-H	8	85.8	34.5	3.5×
ViT-H	1	85.9	29.3	**4.1**×

method	pre-train data	ViT-B	ViT-L	ViT-H	ViT-H_{448}
scratch, our impl.	-	82.3	82.6	83.1	-
DINO [5]	IN1K	82.8	-	-	-
MoCo v3 [9]	IN1K	83.2	84.1	-	-
BEiT [2]	IN1K+DALLE	83.2	85.2	-	-
MAE	IN1K	<u>83.6</u>	<u>85.9</u>	<u>86.9</u>	**87.8**



[29] "Masked Autoencoders Are Scalable Vision Learners", https://arxiv.org/abs/2111.06377

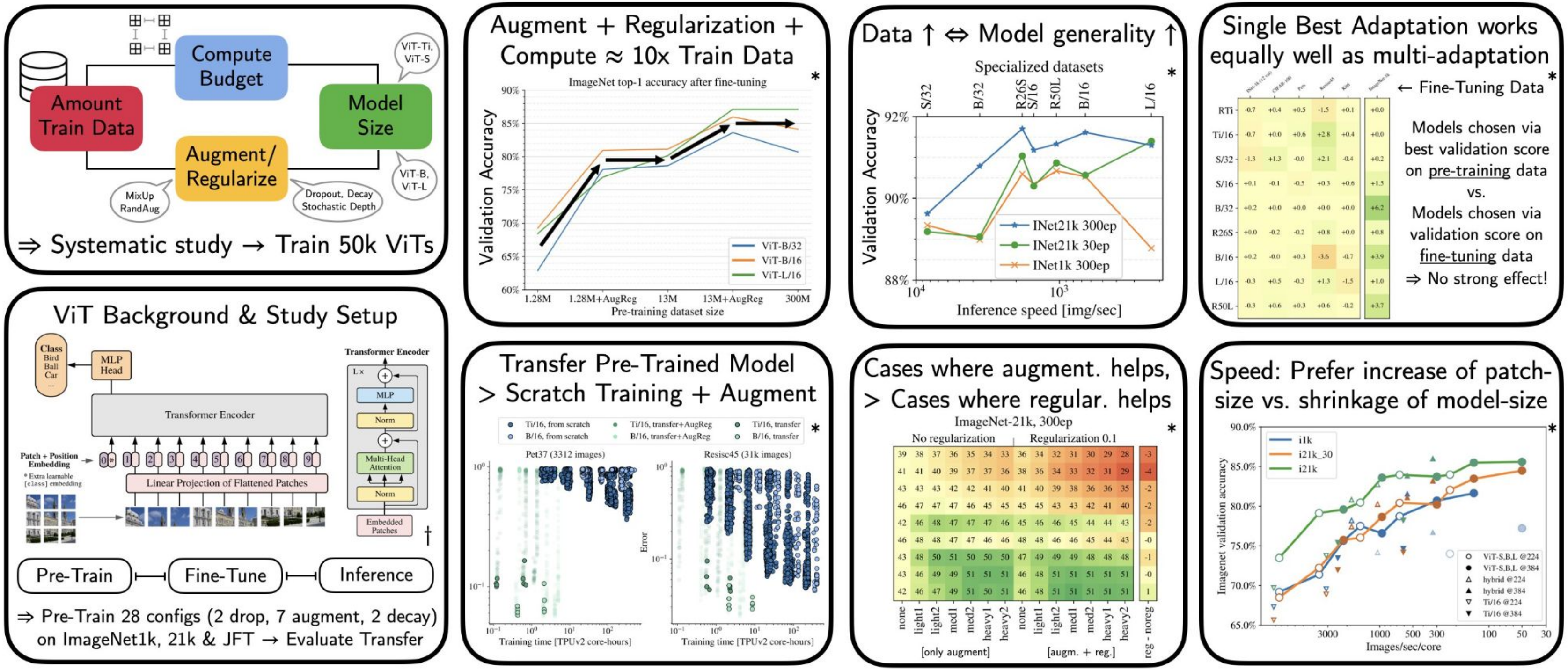
[3] https://iaml-it.github.io/posts/2021-04-28-transformers-in-vision/ ← Show this!

A lot of different ViT setups

GitHub repo with a lot of different ViT setups:

[30] Vision Transformer PyTorch, https://github.com/lucidrains/vit-pytorch

How to train your ViT?



Large-scale ViT study ⇒ AugReg+Compute → Model performance comparable to training on 10x data. Insights into transfer, regularizers, etc.

[31] https://twitter.com/roberttlange/status/1429490398720839683?s=21

[32] "How to train your ViT? Data, Augmentation, and Regularization in Vision Transformers, https://arxiv.org/abs/2106.10270

Overview

1. Introduction
    a. Why is a Transformer interesting?
    b. What is a Transformer on high level?
2. Building blocks
    a. Transformer block
    b. Attention layer
    c. Feedforward layer
    d. Positional encoding
3. Selected CV applications
4. Code
5. Summary & Outlook

How to get started with Transformer & ViT code

[33] The Annotated Transformer introduction

https://nlp.seas.harvard.edu/2018/04/03/attention.html

[30] Vision Transformer code

https://github.com/lucidrains/vit-pytorch

[34] Transformer modifications code

https://github.com/lucidrains/x-transformers

Overview

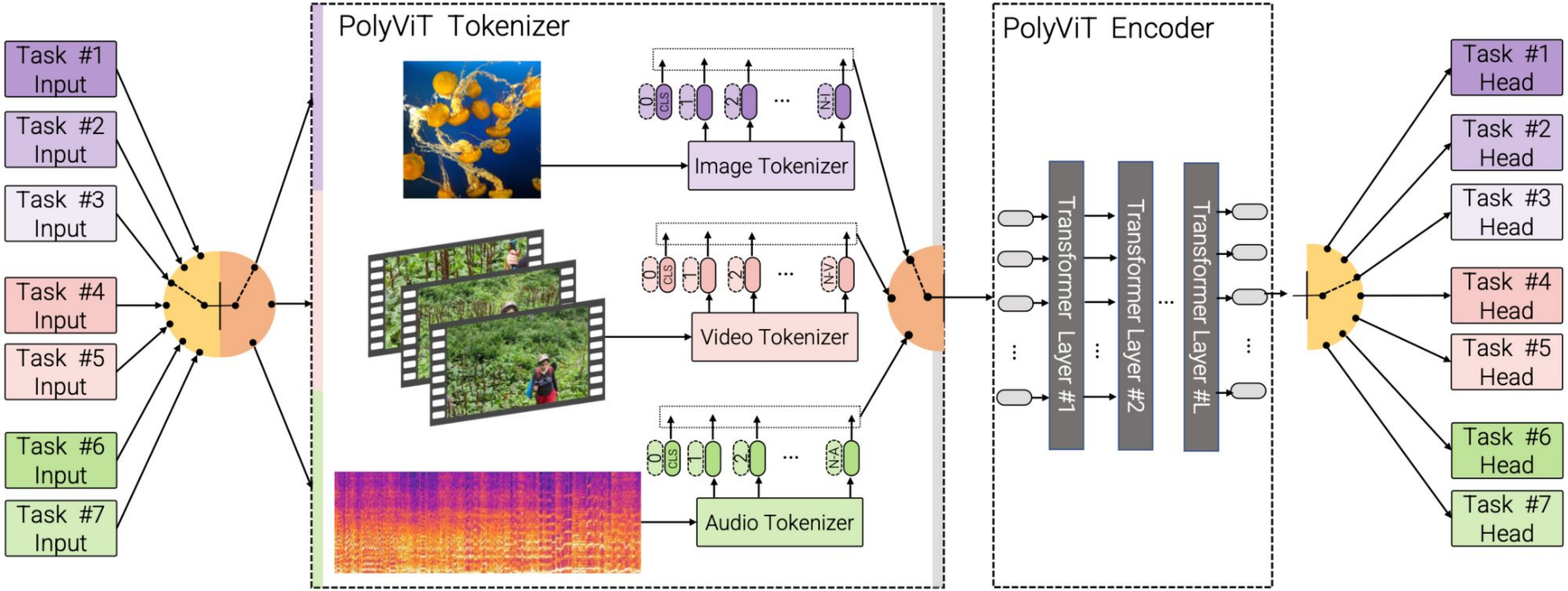
1. Introduction
    a. Why is a Transformer interesting?
    b. What is a Transformer on high level?
2. Building blocks
    a. Transformer block
    b. Attention layer
    c. Feedforward layer
    d. Positional encoding
3. Selected CV applications
4. Code
5. Summary & Outlook

Summary

- Transformers work differently than conventional CNNs but can be used for a wide range of interesting applications.
- Still no clear best practices like with CNNs?
- A lot of recent developments (hard to keep up)!

Outlook

- Multi-modality could offer even more interesting applications?



[35] “PolyViT: Co-training Vision Transformers on Images, Videos and Audio”, https://arxiv.org/abs/2111.12993

Sources 1/3

[1] http://peterbloem.nl/blog/transformers

[2] https://paperswithcode.com/sota/image-classification-on-imagenet,

[3] https://iaml-it.github.io/posts/2021-04-28-transformers-in-vision/

[4] “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, https://arxiv.org/abs/2010.11929

[5] https://www.youtube.com/watch?v=eEXnJOHQ_Xw, 10m 45s

[6] https://distill.pub/2016/augmented-rnns/#attentional-interfaces

[7] “Attention Is All You Need”, https://arxiv.org/abs/1706.03762

[8] https://jalammar.github.io/illustrated-transformer/

[9] Attention step-by-step notebook: https://github.com/MicPie/pytorch/blob/master/attention.ipynb

[10] “LieTransformer - Equivariant self-attention for Lie Groups”, https://arxiv.org/abs/2012.10885

[11] “A Survey of Transformers”, https://arxiv.org/abs/2106.04554

[12] https://twitter.com/thom_wolf/status/1186225108282757120?s=21

Sources 2/3

[13] https://towardsdatascience.com/master-positional-encoding-part-i-63c05d90a0c3

[14] "Language Modeling with Deep Transformers", https://arxiv.org/abs/1905.04226

[15] https://blog.eleuther.ai/rotary-embeddings/

[16] "Train Short, Test Long: Attention with Linear Biases Enables Input Length Extrapolation", https://ofir.io/train_short_test_long.pdf

[17] https://lilianweng.github.io/lil-log/2020/04/07/the-transformer-family.html

[18] "Bottleneck Transformers for Visual Recognition", https://arxiv.org/abs/2101.11605

[19] "Transformers from Scratch", https://e2eml.school/transformers.html

[20] "Generative Pretraining from Pixels", https://openai.com/blog/image-gpt/

[21] https://ai.googleblog.com/2020/12/transformers-for-image-recognition-at.html

[22] "When Vision Transformers Outperform ResNets without Pretraining or Strong Data Augmentations", https://arxiv.org/abs/2106.01548

[23] "Convolution-Free Medical Image Segmentation using Transformers", https://arxiv.org/abs/2102.13645

Sources 3/3

Thank you for your attention! ;-)

[24] https://docs.monai.io/en/latest/whatsnew_0_6.html#unetr-transformers-for-medical-image-segmentation

[25] “UNETR: Transformers for 3D Medical Image Segmentation”, https://arxiv.org/abs/2103.10504

[26] “Rethinking Semantic Segmentation from a Sequence-to-Sequence Perspective with Transformers”, https://arxiv.org/abs/2012.15840

[27] "Segmenting Transparent Object in the Wild with Transformer", https://arxiv.org/abs/2101.08461

[28] “BEiT: BERT Pre-Training of Image Transformers”, https://arxiv.org/abs/2106.08254

[29] “Masked Autoencoders Are Scalable Vision Learners”, https://arxiv.org/abs/2111.06377

[30] Vision Transformer code, https://github.com/lucidrains/vit-pytorch

[31] https://twitter.com/roberttlange/status/1429490398720839683?s=21

[32] “How to train your ViT? Data, Augmentation, and Regularization in Vision Transformers, https://arxiv.org/abs/2106.10270

[33] The Annotated Transformer, https://nlp.seas.harvard.edu/2018/04/03/attention.html

[34] Transformer modifications code, https://github.com/lucidrains/x-transformers

[35] “PolyViT: Co-training Vision Transformers on Images, Videos and Audio”, https://arxiv.org/abs/2111.12993