Transformers: A Review, and Recent Developments in Vision

Anurag Arnab

Google Research

Transformers

Machine translation

English German Translation quality

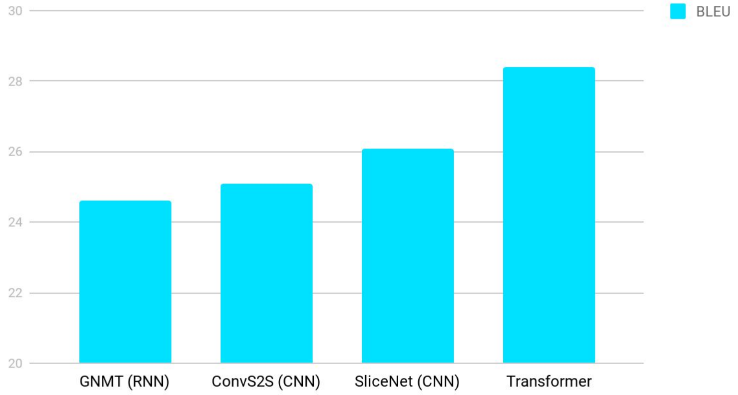
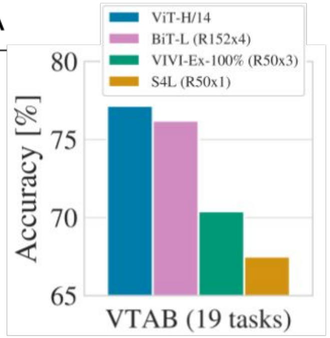


Image classification

	ViT-H	Previous SOTA
ImageNet	88.55	88.5
ImageNet-ReaL	90.72	90.55
Cifar-10	99.50	99.37
Cifar-100	94.55	93.51
Pets	97.56	96.62
Flowers	99.68	99.63



Currently the leading model across a number of domains!

Outline

- What are Transformers?
- Transformers for Computer Vision:
  - Vision Transformers (ViT)
  - Video Vision Transfomers (ViViT)

Context

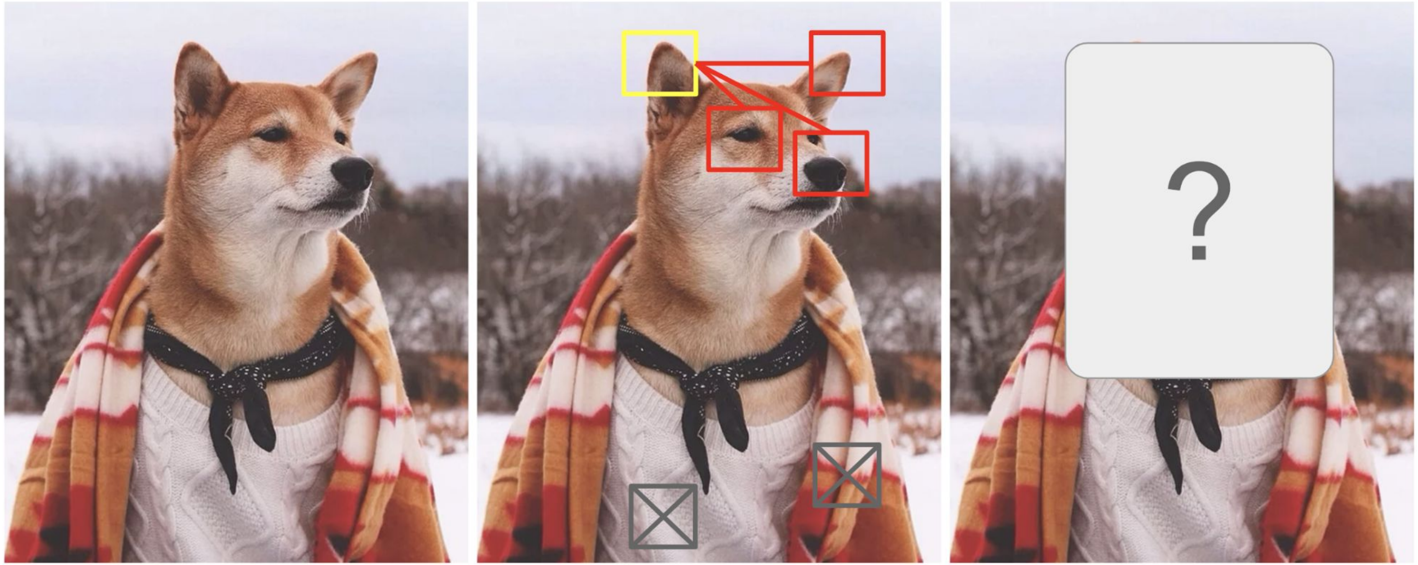
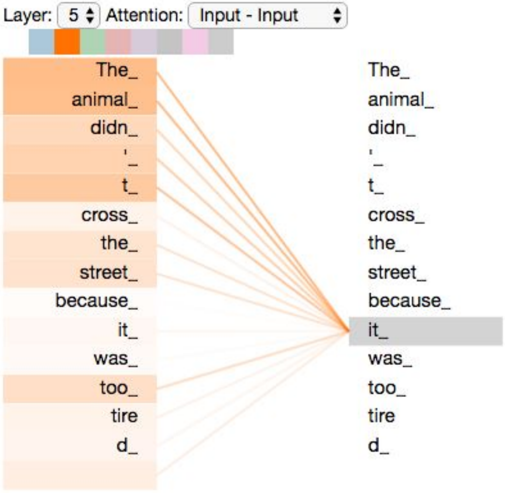


Image credit

Google Research

Context

- `“The animal didn't cross the street because it was too tired”`
- What is “it”?



Try more examples [here]()

Google Research

Context

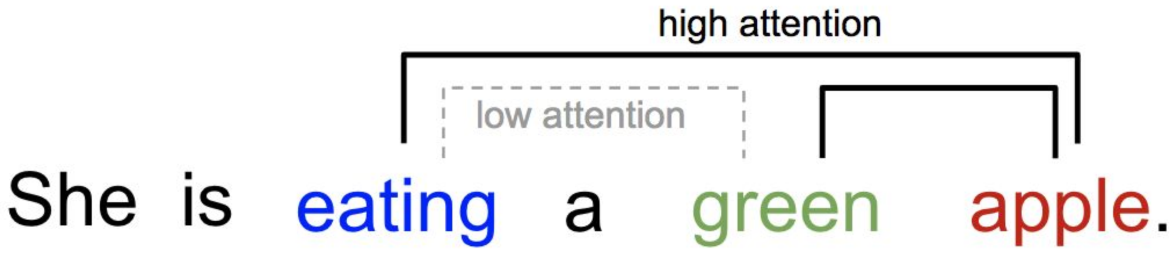
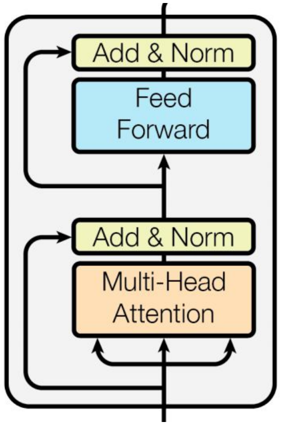


Figure credit

Google Research

Attention and Transformers

- Attention is a method of gathering relevant contextual information
- The Transformer is a neural network layer that relies on attention
- In fact, state-of-the-art models across various domains consist almost entirely of transformer layers.



What is Attention?

High-Level Overview

- Use machine translation as initial example, as this is what Transformers were initially developed for

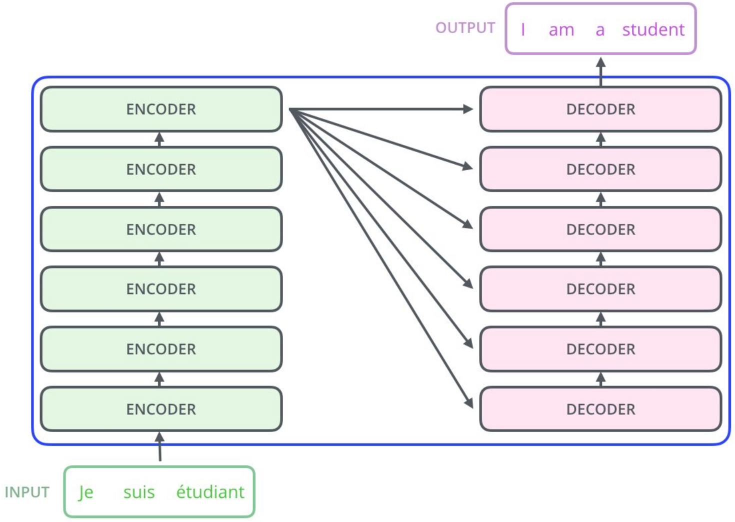
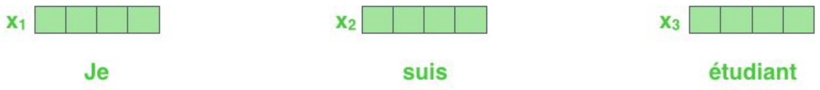


Figure credit for this, and next few slides

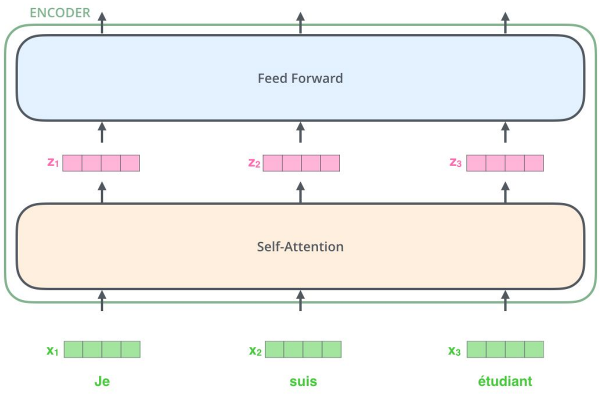
Google Research

High-Level Overview

- Embed input into tokens (fixed dimensional vector)



- Process with encoder layer



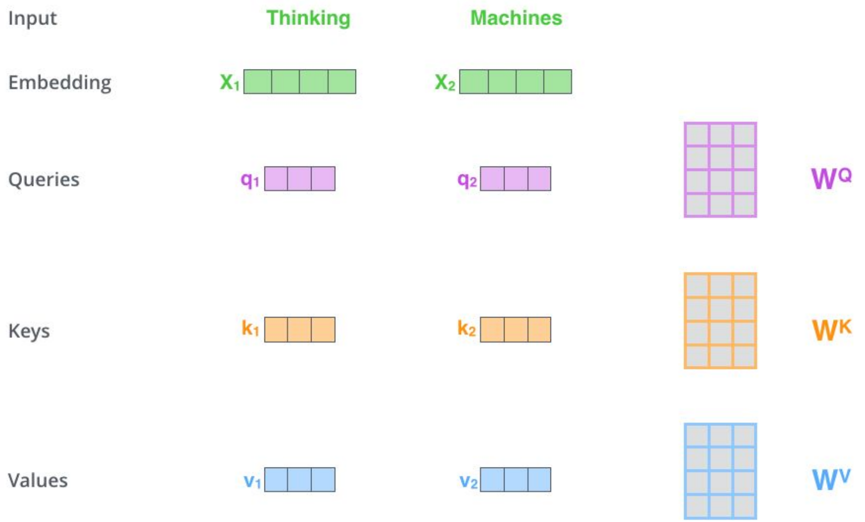
Google Research

Self-Attention in Detail

- Given an input sequence, X
- Project to Query, Key, Values using linear transforms.
- Head output = Softmax(QK^T)V

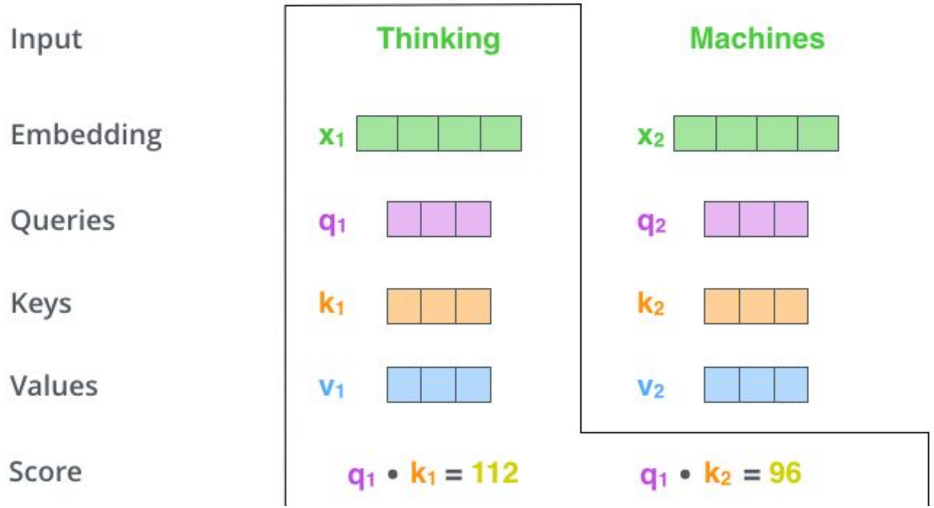
Self-Attention in Detail

- Given an input sequence, X
- Project to Queries, Keys and Values using linear transforms.
  - $Q = W^Q X$, $K = W^K X$, $V = W^V X$



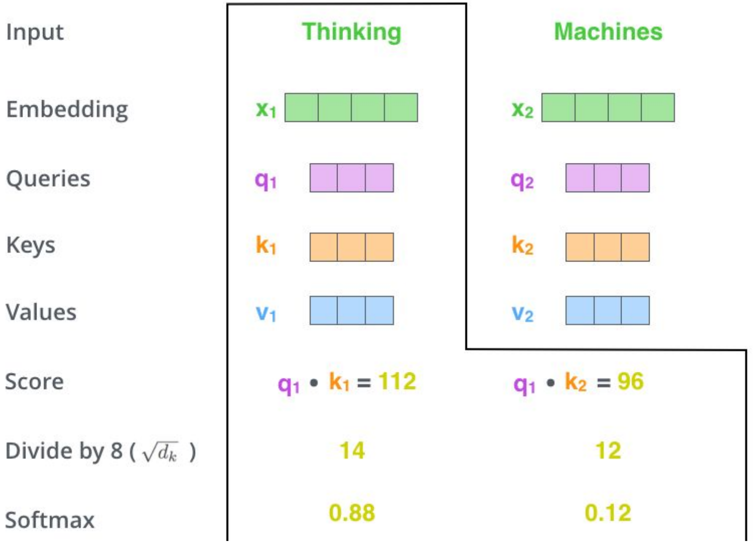
Self-Attention in Detail

- Project to Queries, Keys and Values using linear transforms.
- Calculate a score: For each query, how relevant are all the other words?
    - $\mathrm{Softmax}(QK^T)$



Self-Attention in Detail

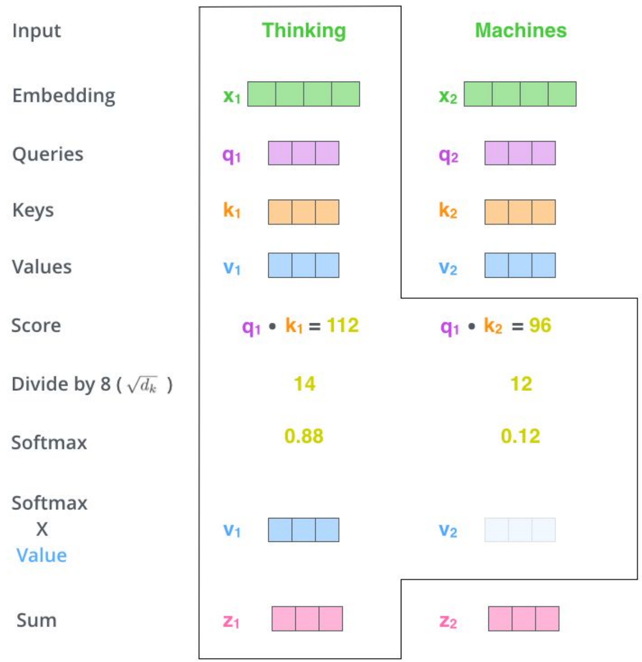
- Project to Queries, Keys and Values using linear transforms.
- Calculate a score: For each query, how relevant are all the other words?



Google Research

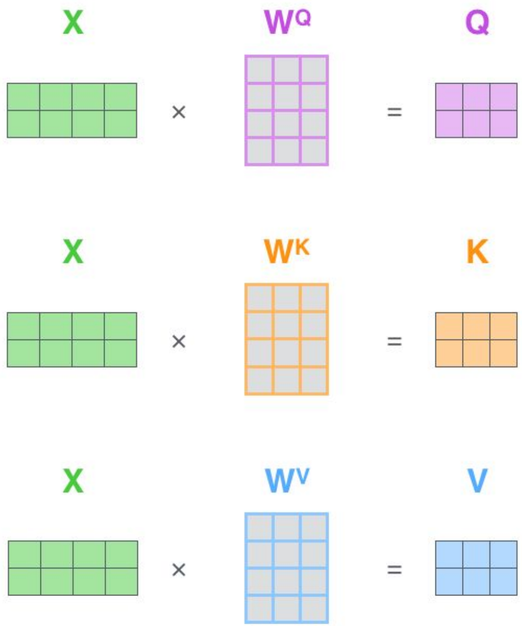
Self-Attention in Detail

- Project to Queries, Keys and Values using linear transforms.
- Calculate a score
- Representation of each query token is attention-weighted sum of values.
    - $Z = \text{Softmax}(QK^T)V$

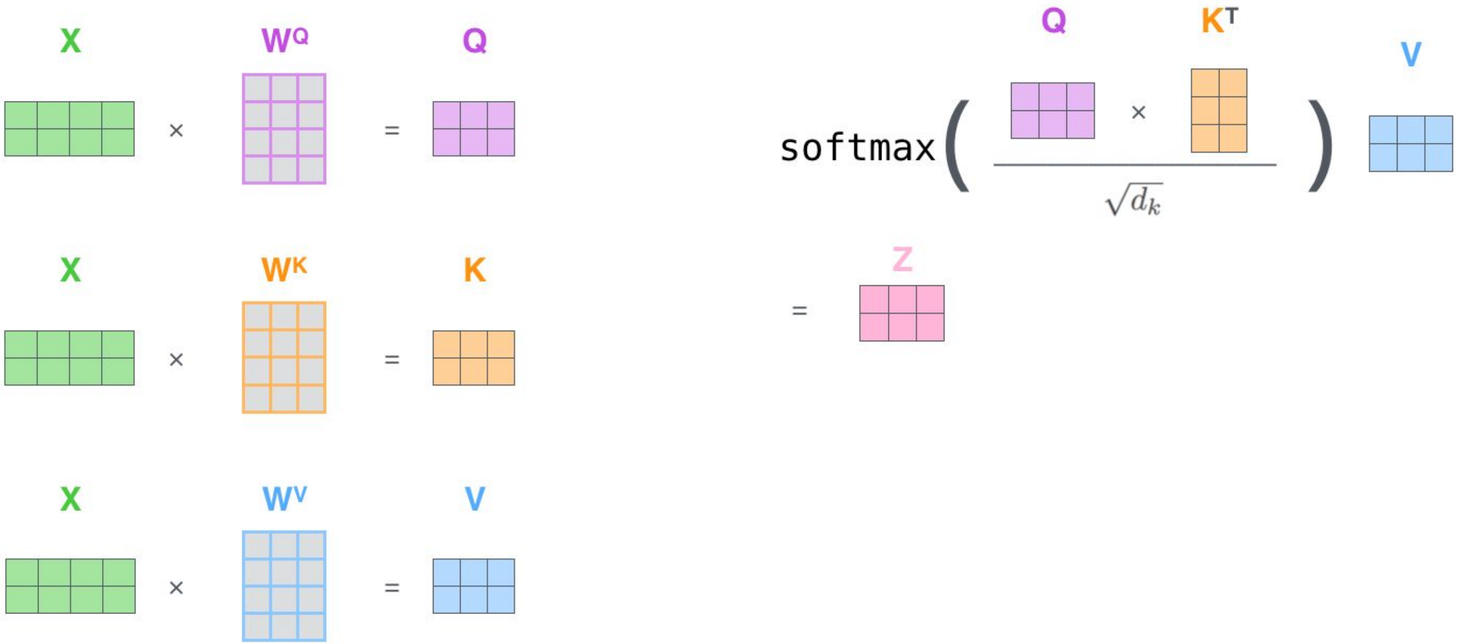


Google Research

Self-Attention in Detail: As a Matrix



Self-Attention in Detail: As a Matrix



Google Research

Self-Attention in Detail: Multiple Heads

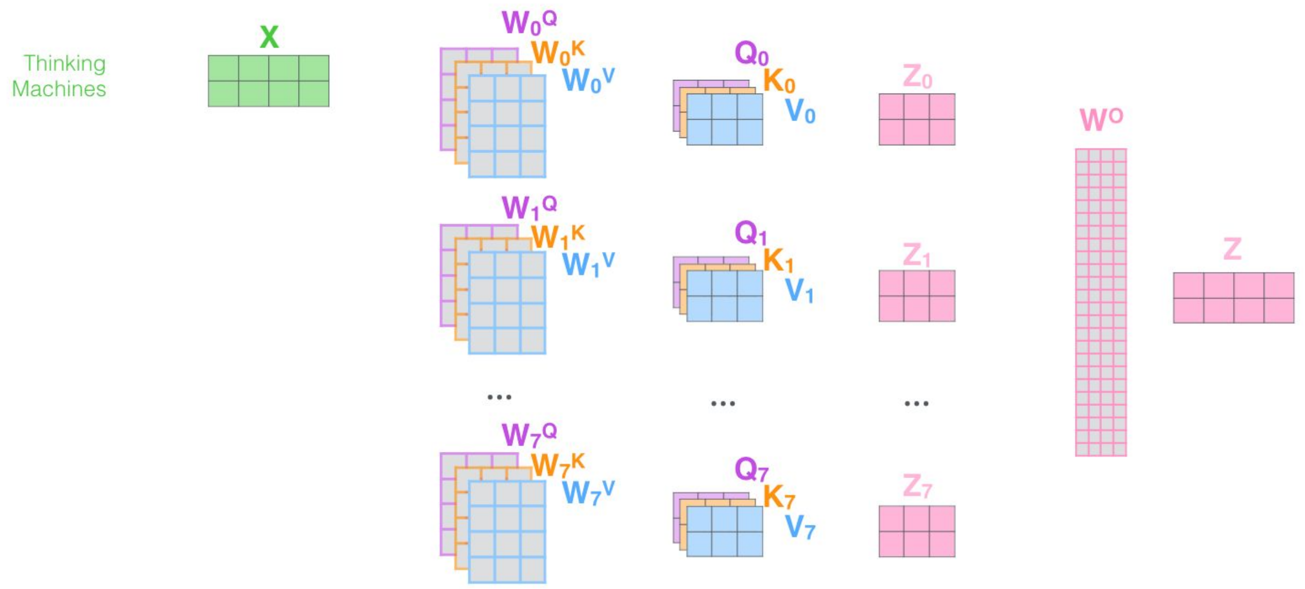
1) This is our input sentence*

2) We embed each word*

3) Split into 8 heads. We multiply X or R with weight matrices

4) Calculate attention using the resulting Q/K/V matrices

5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer



Google Research

Positional Embeddings

- Self-attention is permutation invariant!
    - Say input is [x1, x2, x3]. And output is [y1, y2, y3]
    - If input is [x1, x3, x2]. Output is ...

Positional Embeddings

- Self-attention is permutation invariant!
  - Say input is [x1, x2, x3]. And output is [y1, y2, y3]
  - If input is [x1, x3, x2]. Output is [y1, y3, y2]

Positional Embeddings

- Self-attention is permutation invariant!
    - Say input is [x1, x2, x3]. And output is [y1, y2, y3]
    - If input is [x1, x3, x2]. Output is [y1, y3, y2]
- But what if the ordering of the input vectors conveys information as well?
    - The position of a word in a sentence matters!
    - "“The man ate a fish” != “The fish ate a man”"

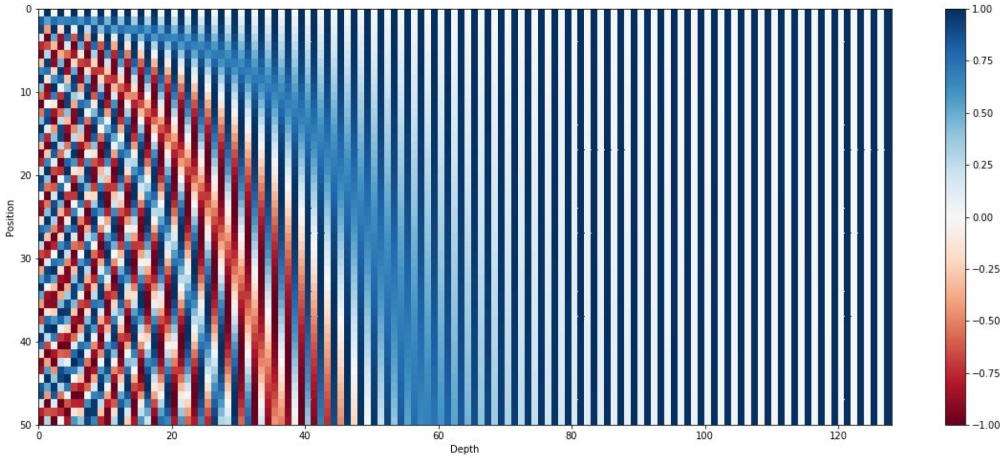
Positional Embeddings

- Self-attention is permutation invariant!
- Learned positional embedding
  - At the input, add a learned vector to each token
  - Representation of the token changes depending on its input position

Positional Embeddings

- Self-attention is permutation invariant!
- Sinusoidal positional embedding

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



Google Research

Figure credit

Putting it all together

- Transformer of [Vaswani et al. Attention is all You Need]()

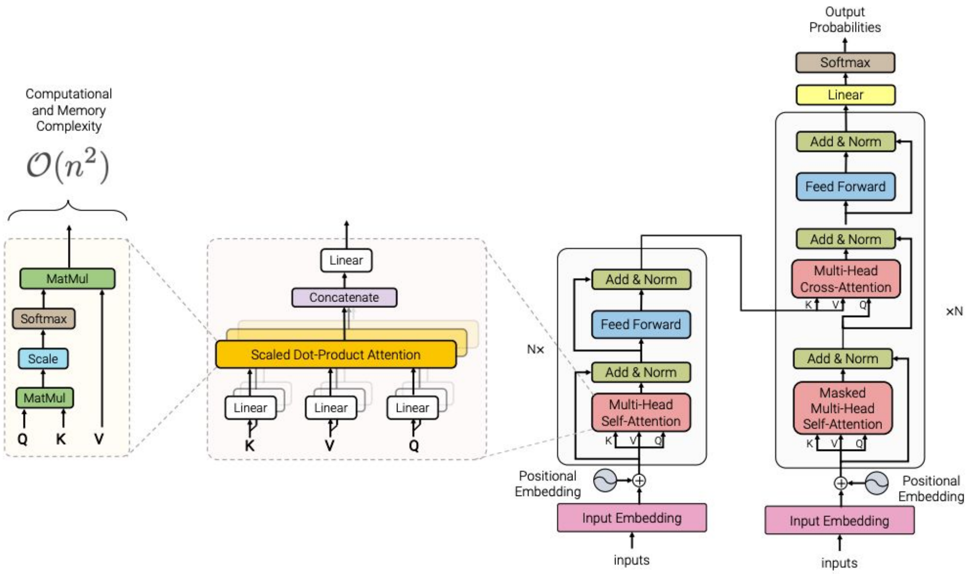


Figure 1: Architecture of the standard Transformer (Vaswani et al., 2017)

Advantages of Transformers

- Great for modelling context
    - Each token can have access to all other tokens in the sequence
- A generic architecture:
    - Operates on any inputs that can be tokenized!
- Parallelizable
- Empirically shown to perform excellently at scale

Transformers at Scale

- Keep performing better with deeper models and more data
- [Scaling laws for neural language models]

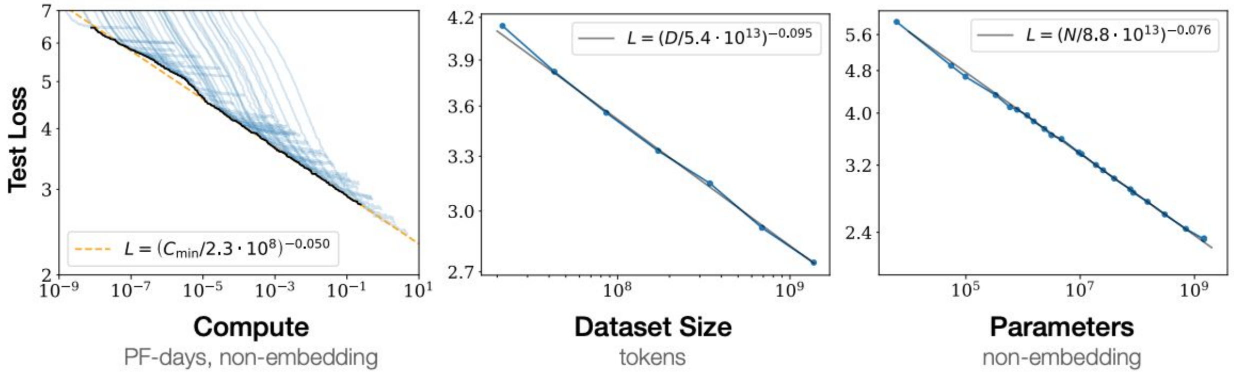


Figure 1 Language modeling performance improves smoothly as we increase the model size, datasetset size, and amount of compute[2] used for training. For optimal performance all three factors must be scaled up in tandem. Empirical performance has a power-law relationship with each individual factor when not bottlenecked by the other two.

Google Research

Weaknesses of Transformers

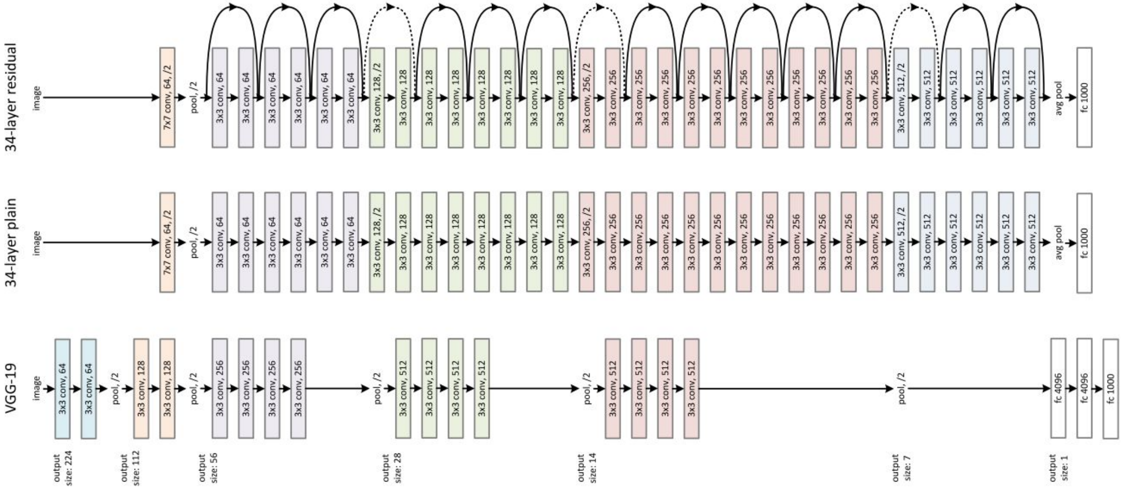
- Quadratic complexity
  - Each token attends to every other token
  - N tokens ⟶ N^2 operations
  - Prohibitive as the number of tokens increases!
- Most powerful language models are extremely expensive
- Large body of work on more efficient transformers. Good survey paper
- Transformers can overfit easily on smaller datasets

Going beyond language

- Transformers are the state-of-the-art in language processing
- Lot of data available for language
- Can transformers be adapted for other domains (ie Computer Vision)?

Transformers and Computer Vision

- CNNs are the architecture of choice in Vision
- Transformers are the architecture of choice in NLP

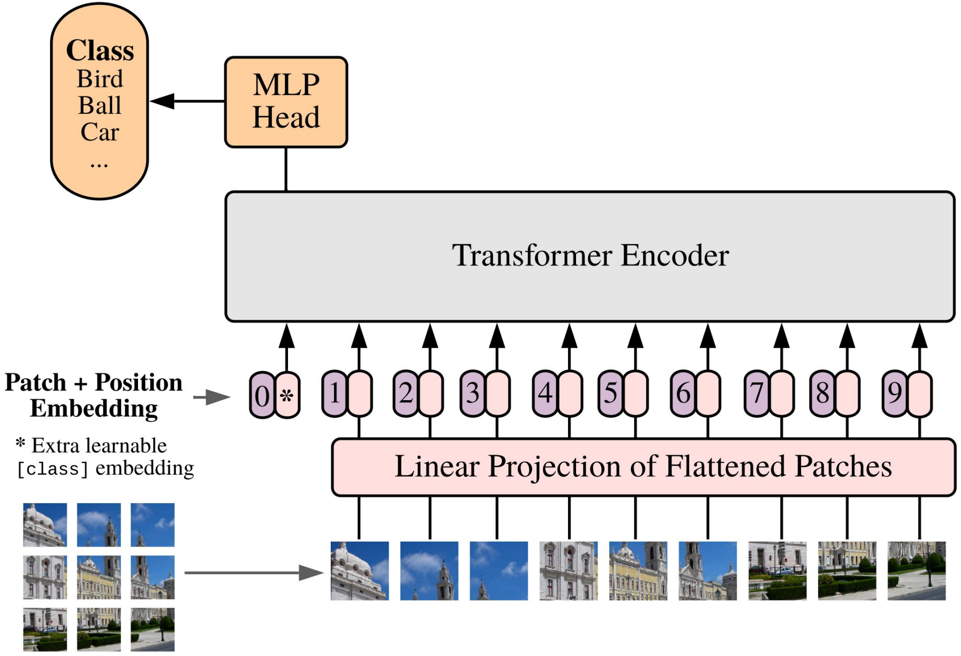


Transformers and Computer Vision

- CNNs are the architecture of choice in Vision
- Transformers are the architecture of choice in NLP
- Numerous attempts to incorporate self-attention into CNNs:
  - Wang CVPR 2018, Bello ICCV 2019, Huang ICCV 2019, Carion ECCV 2020
- Or to replace convolutions entirely with self-attention
  - Parmar ICML 2018, Ramachandran NeurIPS 2019

Vision Transformers

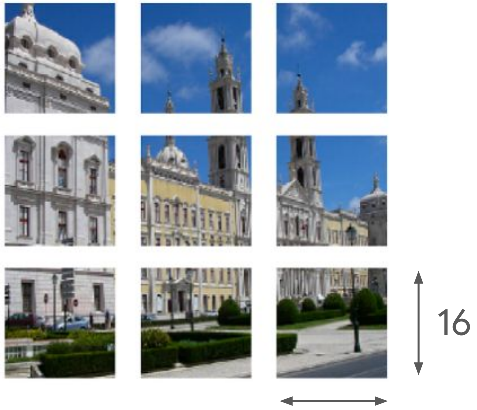
- [An Image is Worth 16x16 Words]



Google Research

Vision Transformer Models

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

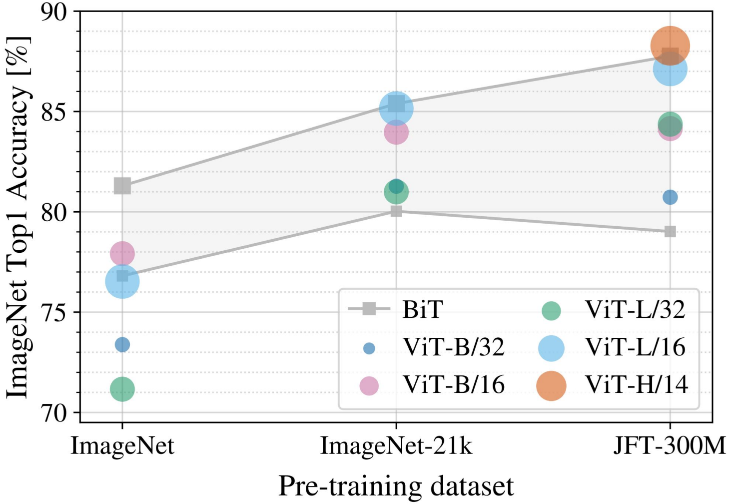


Notation e.g. ViT-L/16

Google Research

Vision Transformers are effective at scale

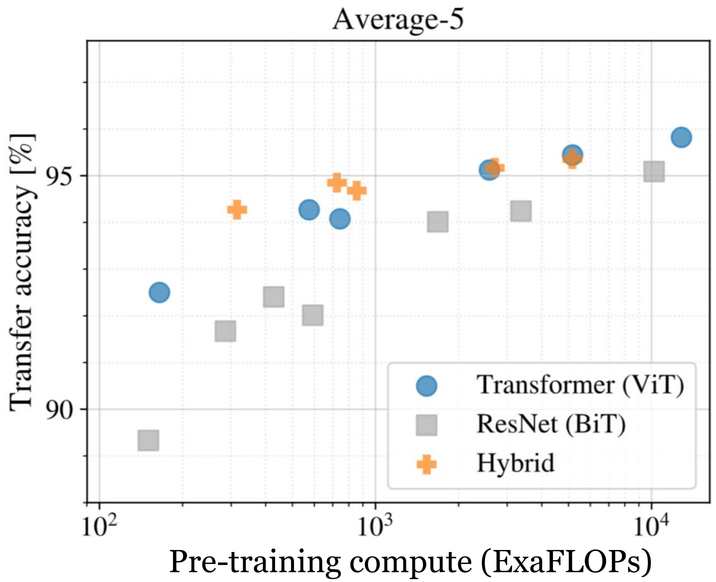
- Transformers have less inductive biases then Convolutional Networks (ie translational equivariance)
- So they need more data to train



Google Research

Vision Transformers are effective at scale

- Transformers, are however, able to take advantage of large-scale data better than CNNs can
- And are more compute-efficient too in terms of computation to reach accuracy.



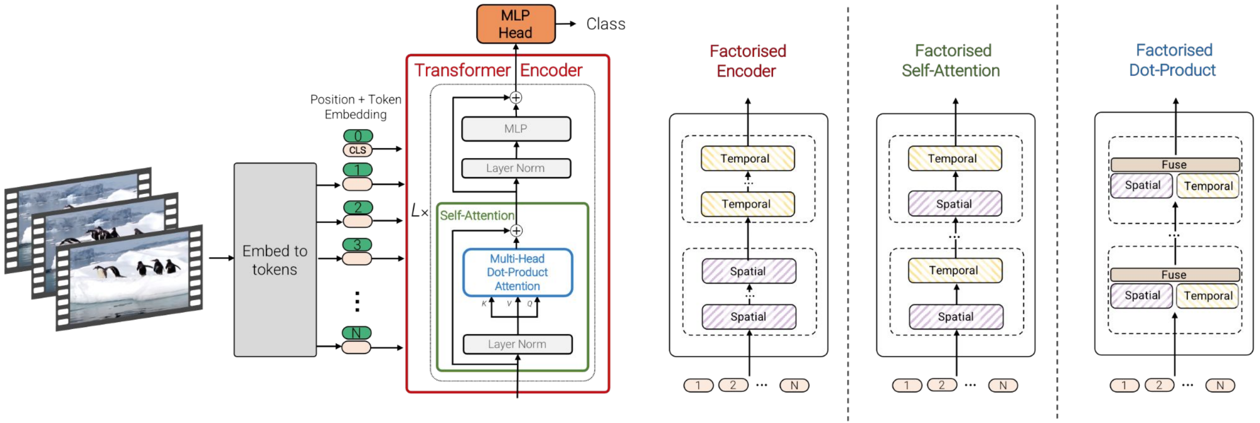
Google Research

Vision Transformers

- A paradigm shift in Computer Vision
  - Do we still need convolutional networks?
- A number of follow-ups:
  - More efficient
  - Other tasks like segmentation and detection
  - [Pyramid ViT], [Swin Transformer], [mViT]

Vision Transformers for Video

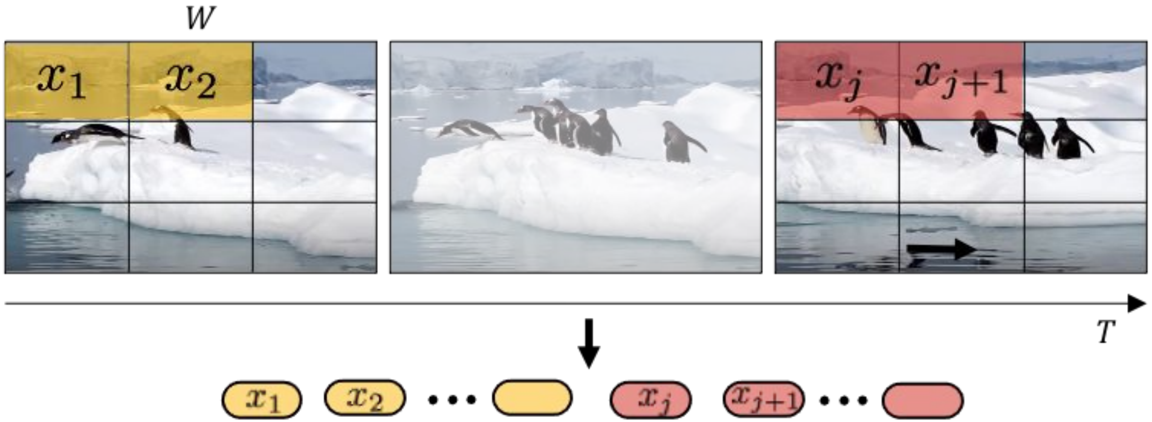
- [ViViT: A Video Vision Transformer]
- To handle large number of tokens, explore more efficient factorised attention variants.
- Regularisation to train on comparatively small video datasets.



Google Research

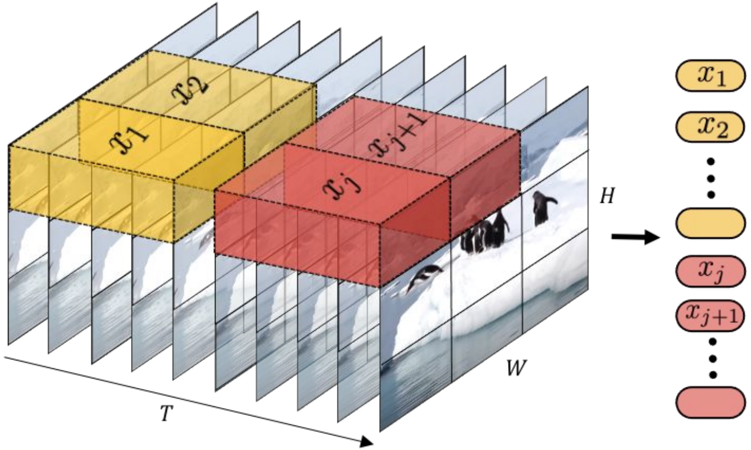
Input Encoding 1: Uniform Frame Sampling

- Sample frames, extract 2D patches and linearly project (as in ViT)
- Effectively consider a video as a “big image”



Input Encoding 2: Tubelet embedding

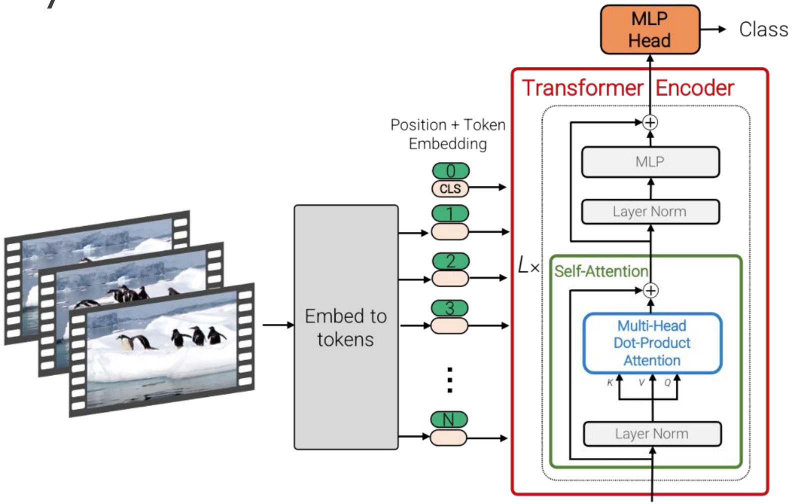
- Extract 3D tubelets to encode spatio-temporal “tubes” into tokens
- Temporal information included from the initial tokenisation stage.
- Works better when initialised appropriately.



Google Research

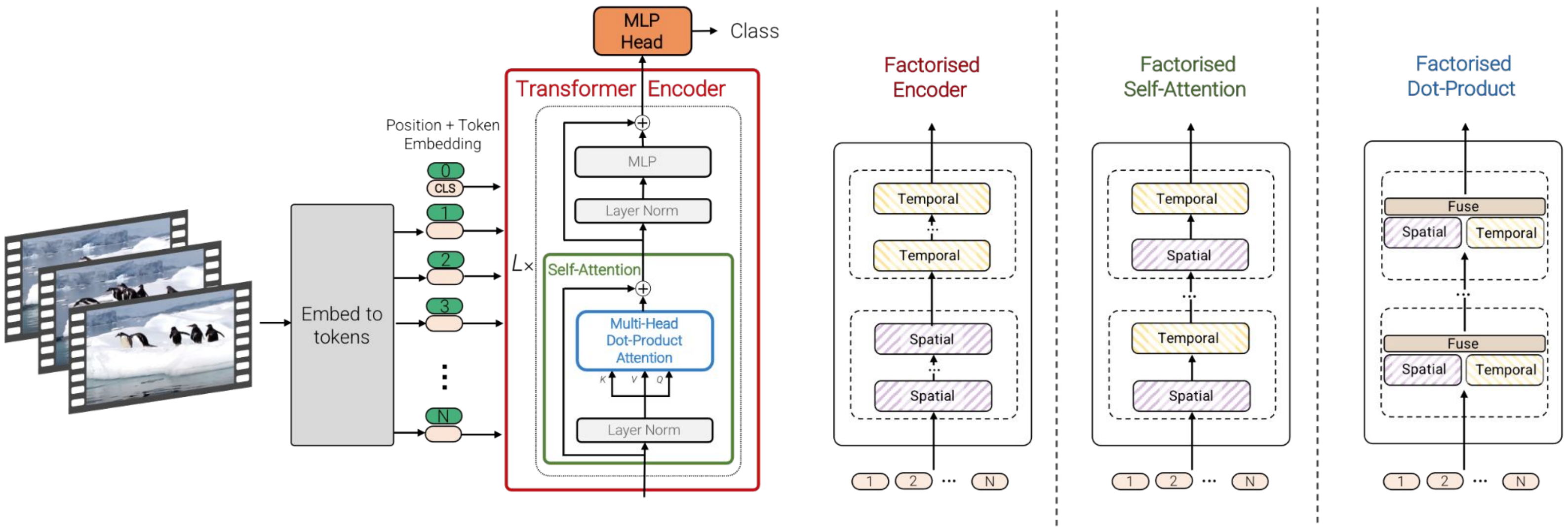
ViViT: Joint Spatio-Temporal Attention

- Simply forward many spatio-temporal tokens through multiple transformer layers.
- Requires a lot of computation, and high-capacity means it can overfit easily on smaller datasets.



Google Research

ViViT: Space/Time Factorisations



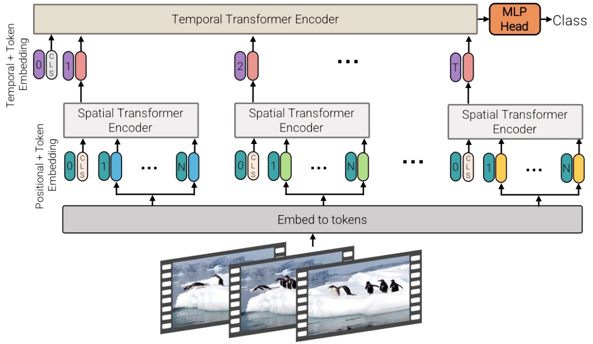
Alternative ways of mixing the temporal and spatial information

Reduces complexity from $O((w * h)^2 + t^2)$ instead of $O((w*h*t)^2)$

ViViT Factorisations

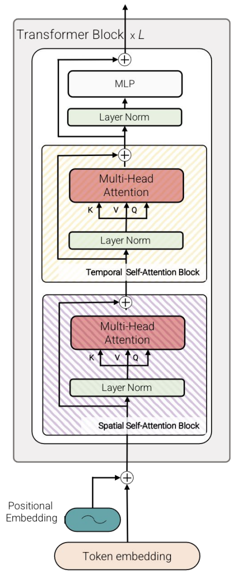
Factorised encoder

- “Late fusion” of spatial and temporal information



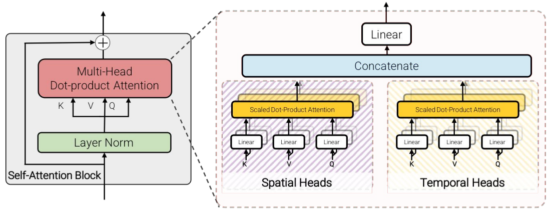
Factorised self-attention

- Perform self-attention separately over space and time



Factorised dot-product

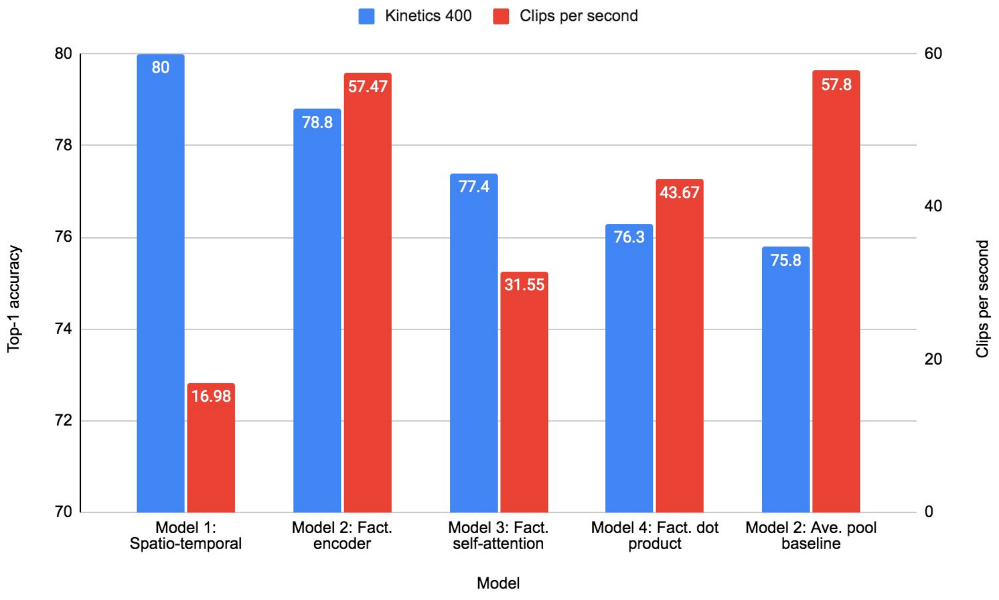
- Attention heads separated over space and time dimensions.



Google Research

Model Variants

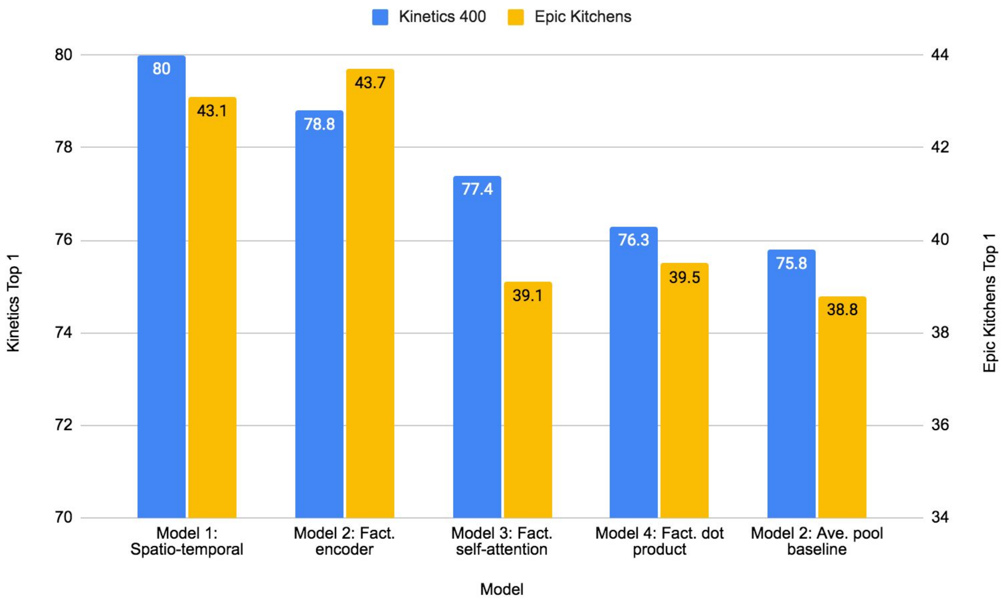
- Tokens fixed across models
- Unfactorised model works best on larger datasets (ie Kinetics), but slowest.



Google Research

Model Variants

- Factorised encoder works best on smaller datasets (ie Epic Kitchens) as it overfits less.



Google Research

State-of-the-art Results on 5 Datasets

(a) Kinetics 400

Method	Top 1	Top 5	Views
bIVNet [16]	73.5	91.2	–
STM [30]	73.7	91.6	–
TEA [39]	76.1	92.5	10 × 3
TSM-ResNeXt-101 [40]	76.3	–	–
I3D NL [72]	77.7	93.3	10 × 3
CorrNet-101 [67]	79.2	–	10 × 3
ip-CSN-152 [63]	79.2	93.8	10 × 3
LGD-3D R101 [48]	79.4	94.4	–
SlowFast R101-NL [18]	79.8	93.9	10 × 3
X3D-XXL [17]	80.4	94.6	10 × 3
TimeSformer-L [2]	80.7	94.7	1 × 3
ViViT-L/16x2	80.6	94.7	4 × 3
ViViT-L/16x2 320	**81.3**	**94.7**	4 × 3
Methods with large-scale pretraining			
ip-CSN-152 [63] (IG [41])	82.5	95.3	10 × 3
ViViT-L/16x2 (JFT)	82.8	95.5	4 × 3
ViViT-L/16x2 320 (JFT)	83.5	95.5	4 × 3
ViViT-H/16x2 (JFT)	**84.8**	**95.8**	4 × 3

(b) Kinetics 600

Method	Top 1	Top 5	Views
AttentionNAS [73]	79.8	94.4	–
LGD-3D R101 [48]	81.5	95.6	–
SlowFast R101-NL [18]	81.8	95.1	10 × 3
X3D-XL [17]	81.9	95.5	10 × 3
TimeSformer-HR [2]	82.4	**96.0**	–
ViViT-L/16x2	82.5	95.6	4 × 3
ViViT-L/16x2 320	**83.0**	95.7	4 × 3
ViViT-L/16x2 (JFT)	84.3	96.2	4 × 3
ViViT-H/16x2 (JFT)	**85.8**	**96.5**	4 × 3

(c) Moments in Time

	Top 1	Top 5
TSN [69]	25.3	50.1
TRN [83]	28.3	53.4
I3D [6]	29.5	56.1
bIVNet [16]	31.4	59.3
AssembleNet-101 [51]	34.3	62.7
ViViT-L/16x2	**38.0**	**64.9**

(d) Epic Kitchens 100 Top 1 accuracy

Method	Action	Verb	Noun
TSN [69]	33.2	60.2	46.0
TRN [83]	35.3	65.9	45.4
TBN [33]	36.7	66.0	47.2
TSM [40]	38.3	**67.9**	49.0
SlowFast [18]	38.5	65.6	50.0
ViViT-L/16x2 Fact. encoder	**44.0**	66.4	**56.8**

(e) Something-Something v2

Method	Top 1	Top 5
TRN [83]	48.8	77.6
SlowFast [17, 77]	61.7	–
TimeSformer-HR [2]	62.5	–
TSM [40]	63.4	88.5
STM [30]	64.2	89.8
TEA [39]	65.1	–
bIVNet [16]	65.2	**90.3**
ViViT-L/16x2 Fact. encoder	**65.4**	89.8

Google Research

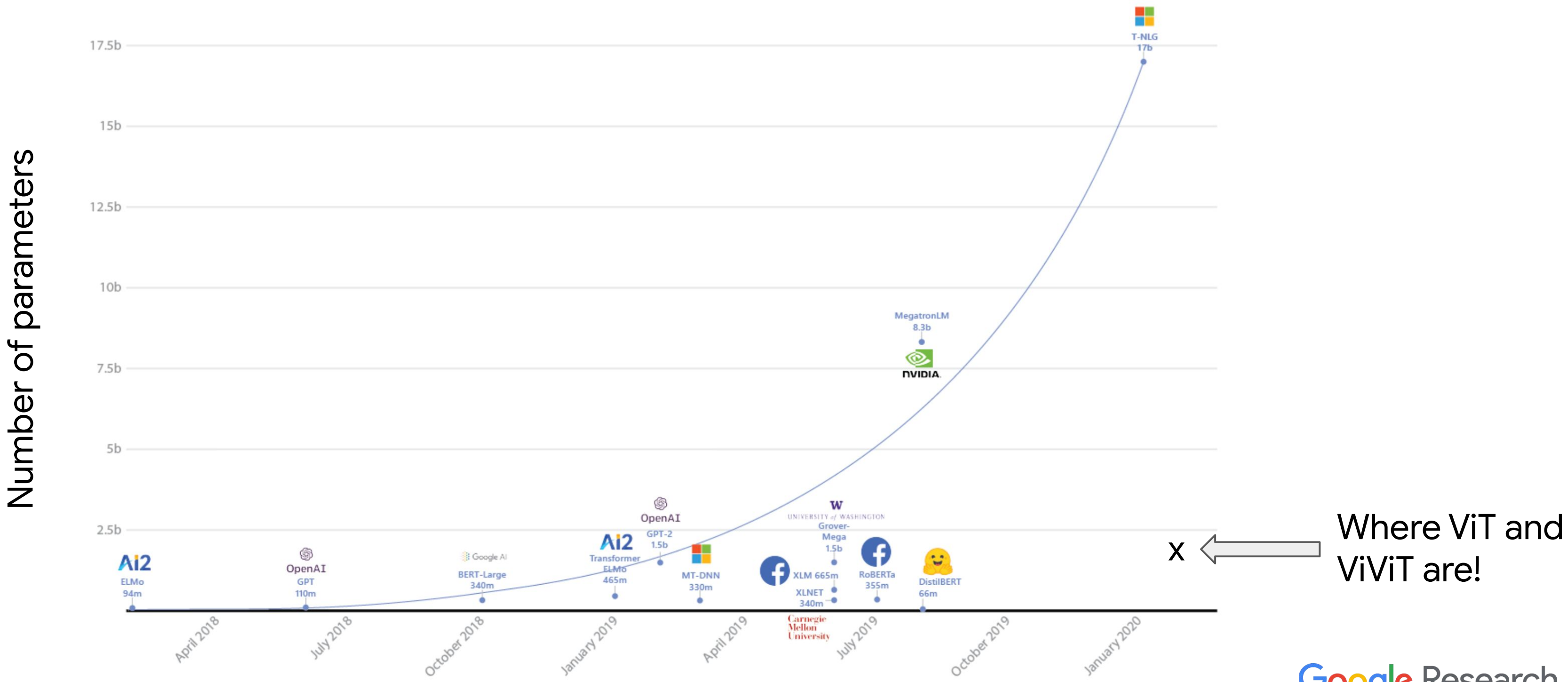
Regularisation

- Video datasets are not as large as ImageNet / ImageNet21k / JFT
  - Original ViT paper didn't get good performance on ImageNet.
- Strategies
  - Use pretrained image models from ImageNet-21K or JFT
  - For smaller datasets, we use further regularisation methods, inspired by DeIT.

	Top-1 accuracy
Random crop, flip, colour jitter	38.4
+ Kinetics 400 initialisation	39.6
+ Stochastic depth [28]	40.2
+ Random augment [10]	41.1
+ Label smoothing [58]	43.1
+ Mixup [79]	43.7

5.3% gain on Epic Kitchens

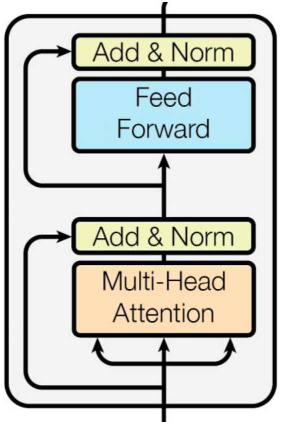
Context: NLP vs Vision

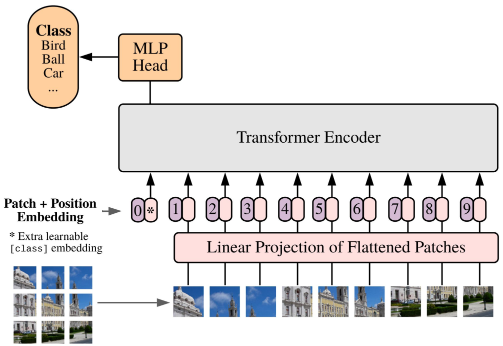


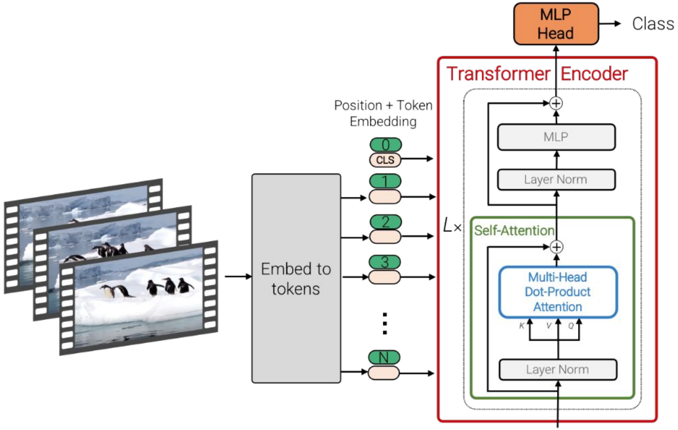
Google Research

Questions?

- anurag.arnab@gmail.com







Google Research