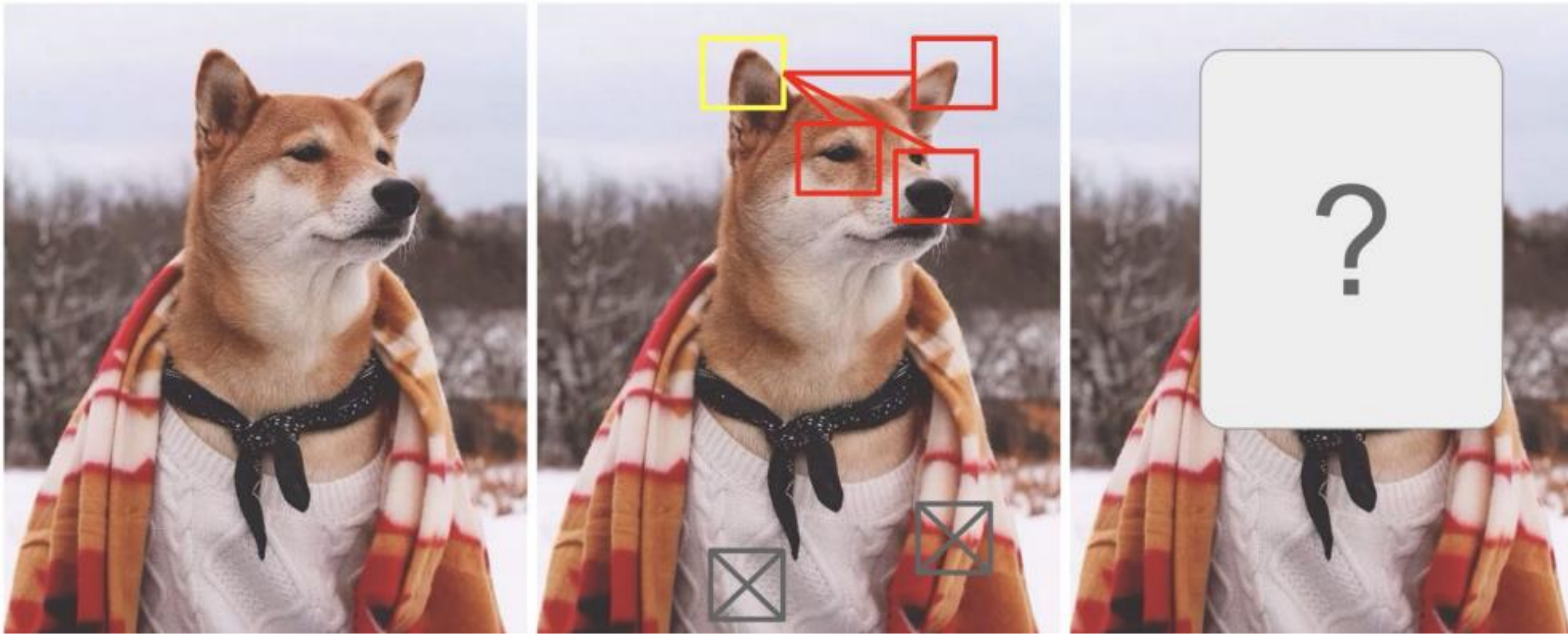
Transformers

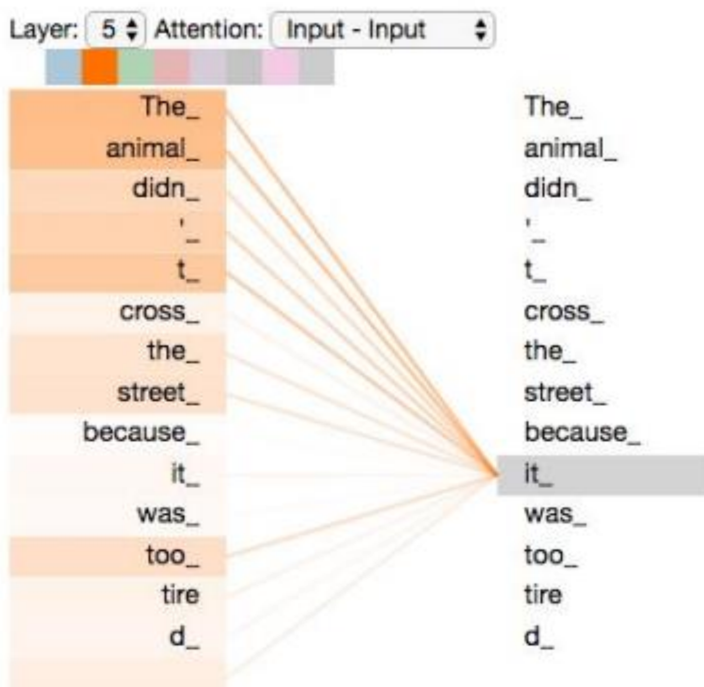
- Context



Slides: Google Research

Context

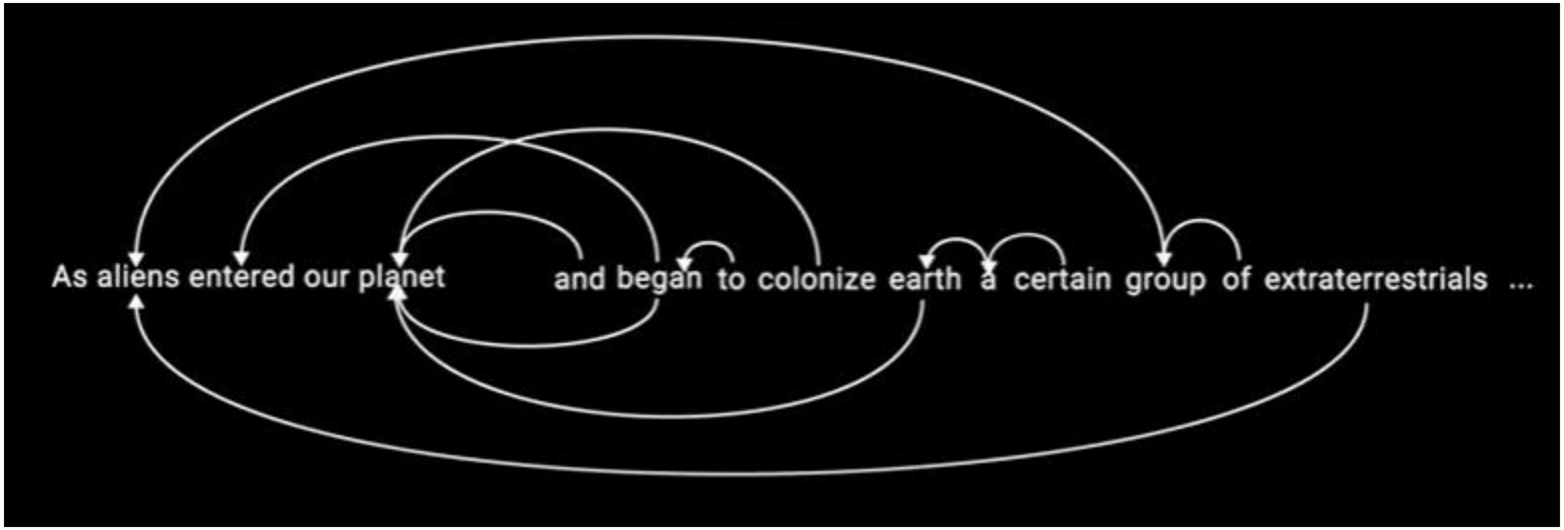
- "The animal didn't cross the street because it was too tired"
- What is "it"?



Context

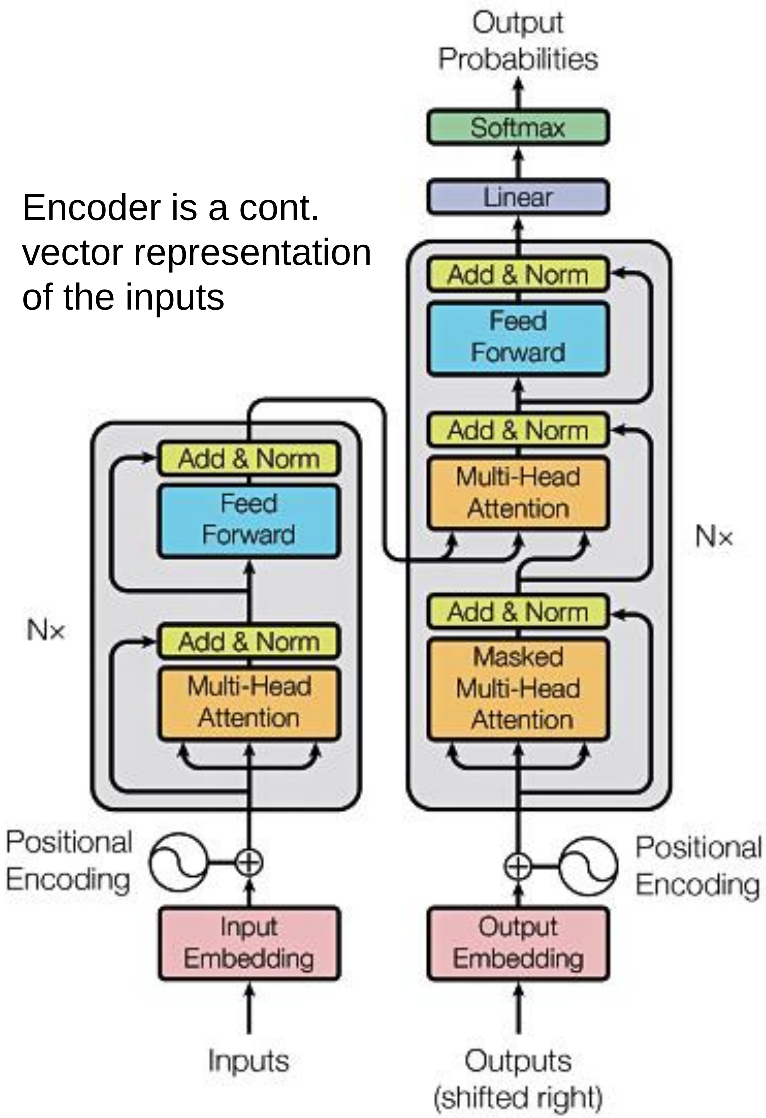
Attention

- Attention mechanism has an infinite reference window.

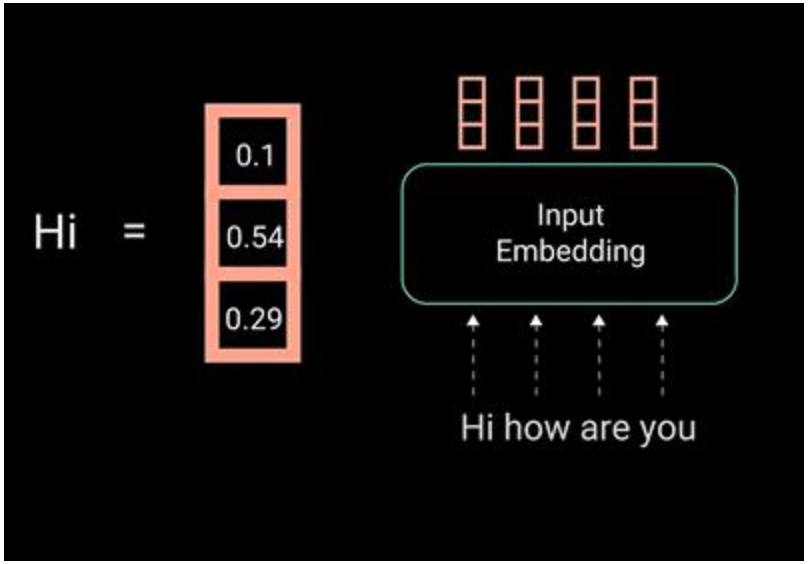


Attention is all you need

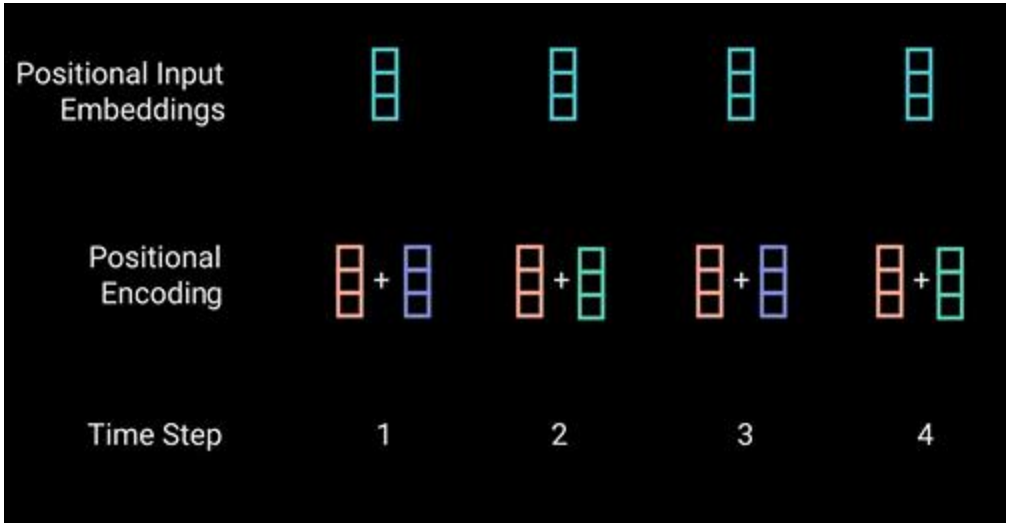
Encoder is a cont. vector representation of the inputs



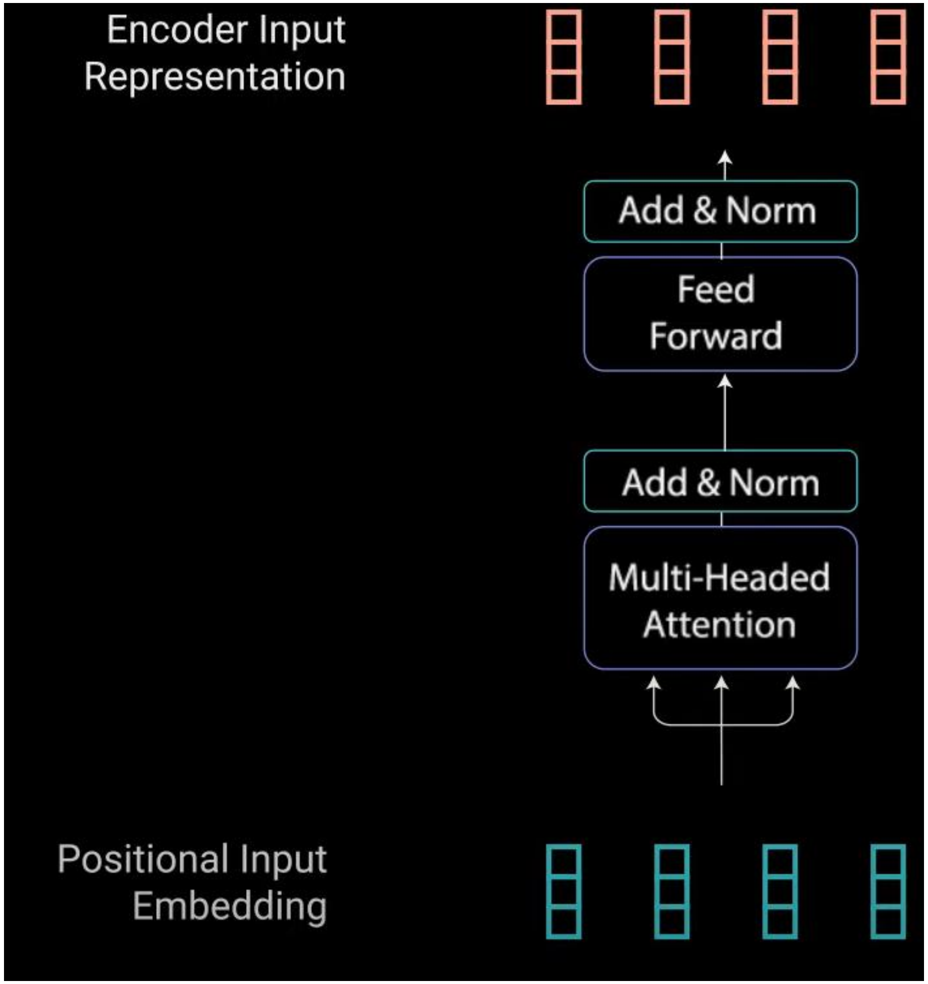
1. Input Embedding



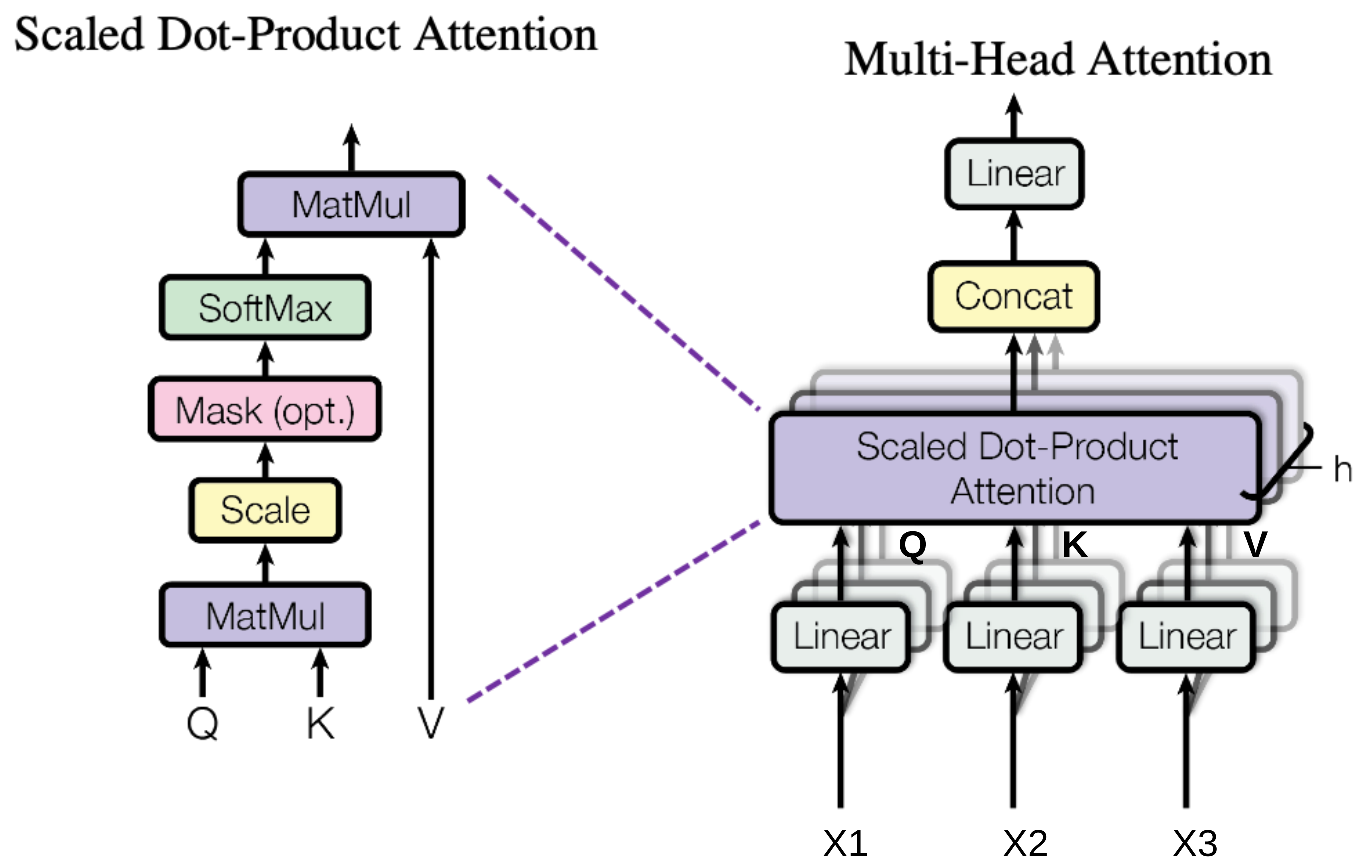
2. Positional Encoding



3-4 Encoder layer

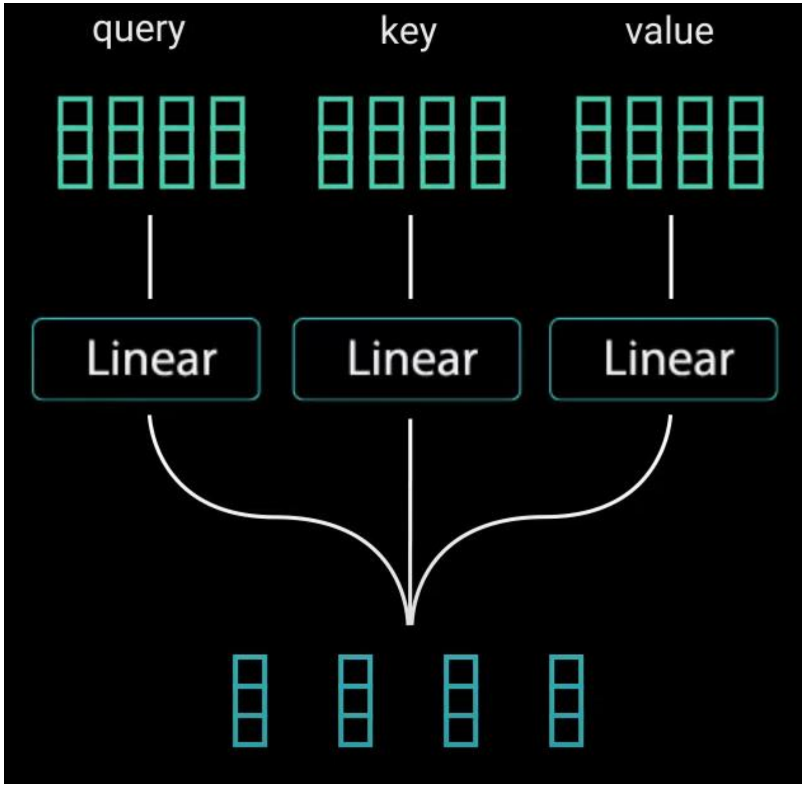


Multi-Head Attention



Self Attention

- Associates each individual word in the input to other words in the input.



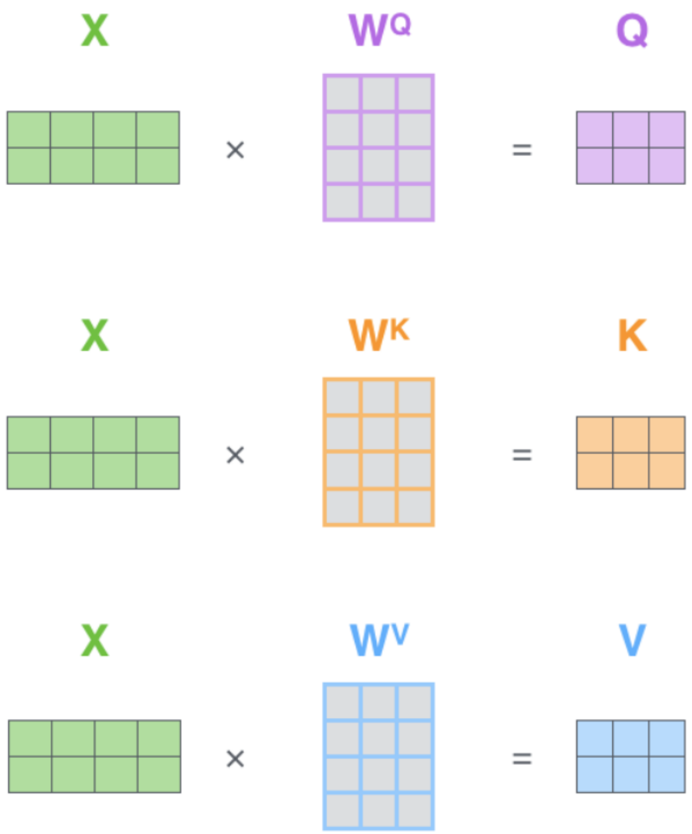
Self attention details

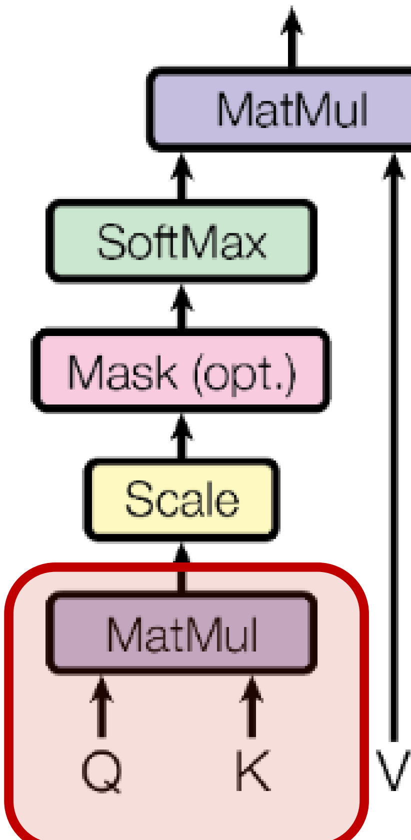
- Given an input sequence, X
- Project to Queries, Keys and Values using linear transforms.

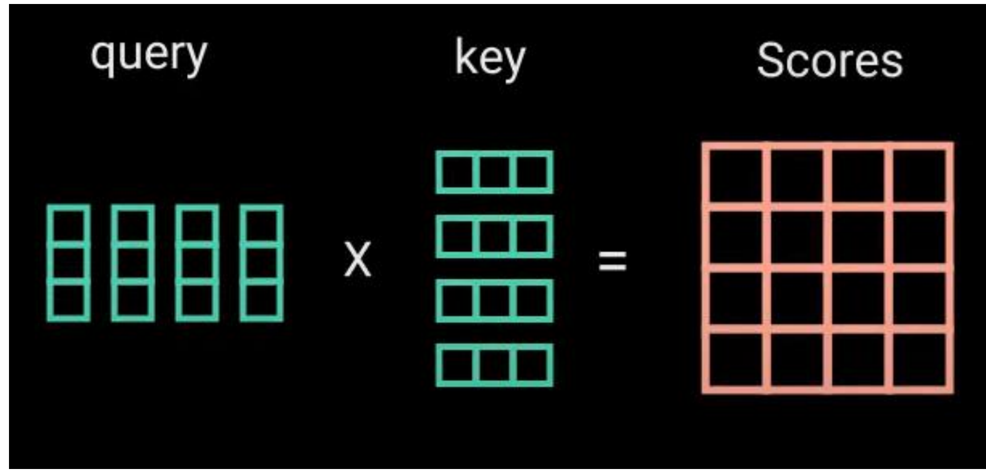
$Q = W^Q X$

$K = W^K X$

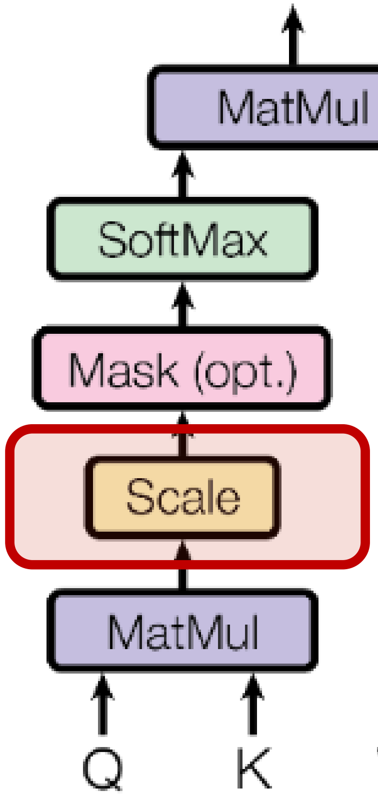
$V = W^V X$

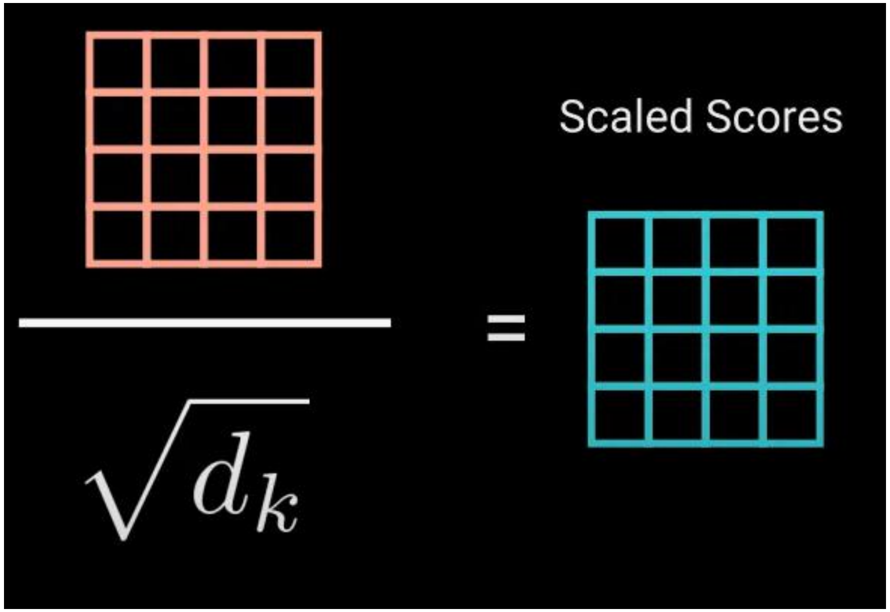


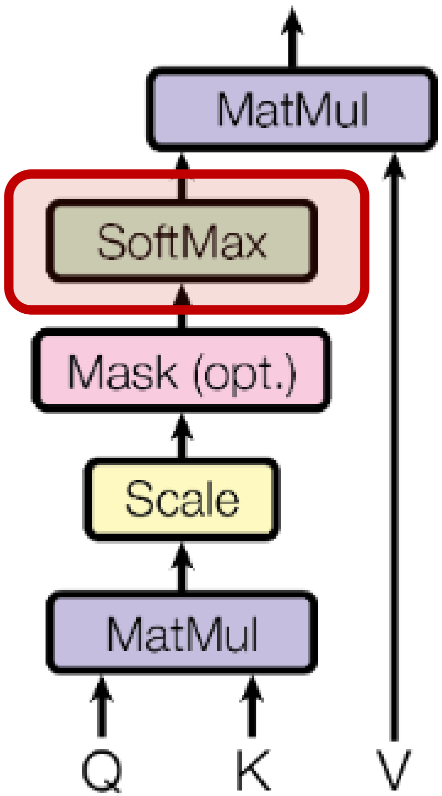


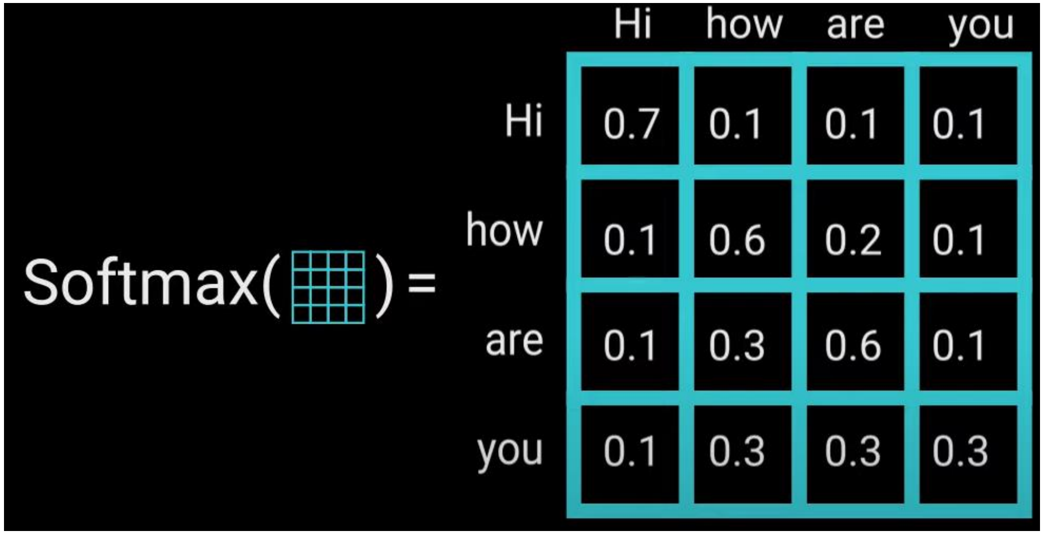


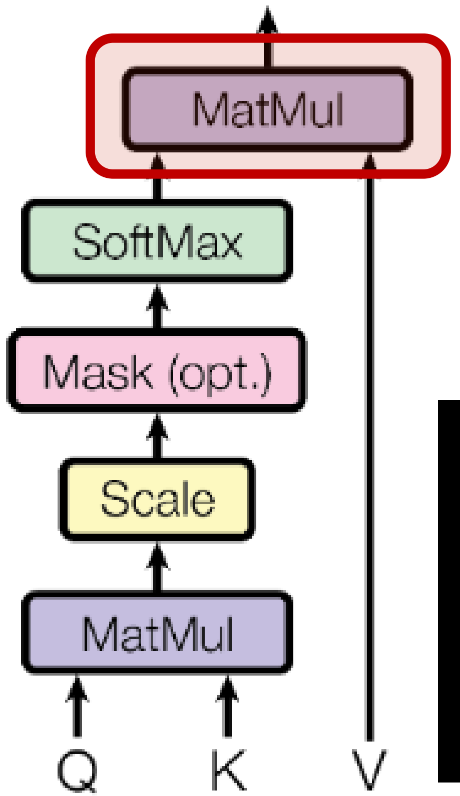
	Hi	how	are	you
Hi	98	27	10	12
how	27	89	31	67
are	10	31	91	54
you	12	67	54	92

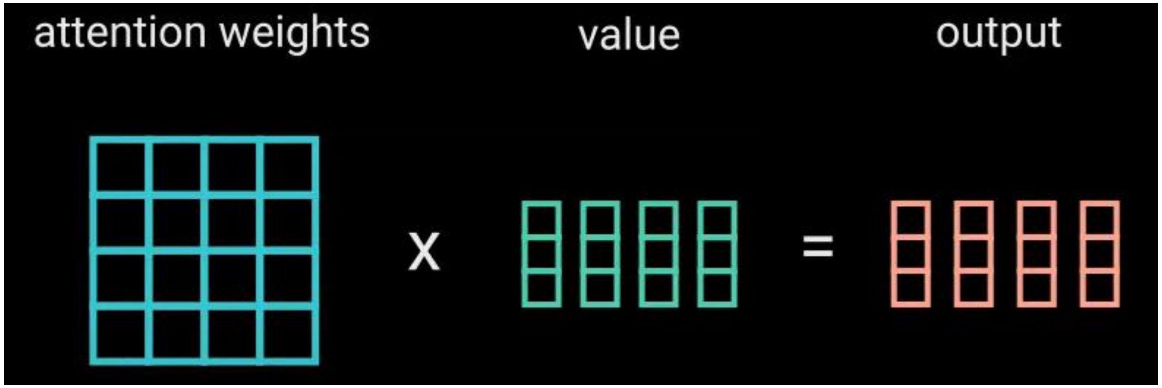

MatMul
SoftMax
Mask (opt.)
Scale
MatMul
Q
K
V

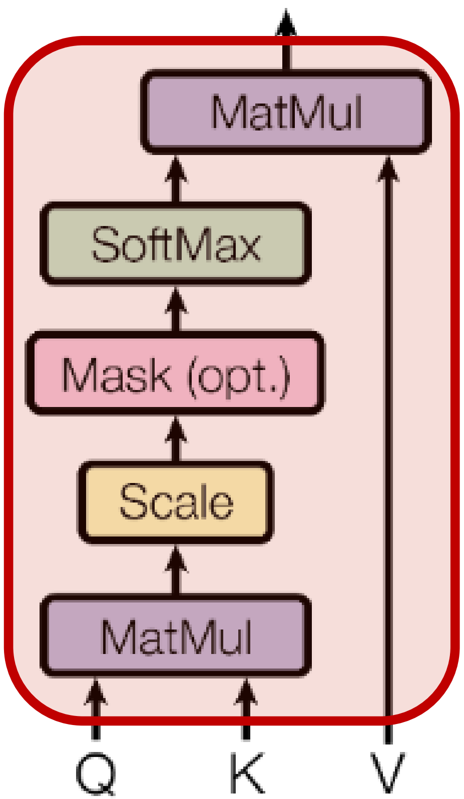

Scaled Scores
$\sqrt{d_k}$
=


MatMul
SoftMax
Mask (opt.)
Scale
MatMul
Q
K
V

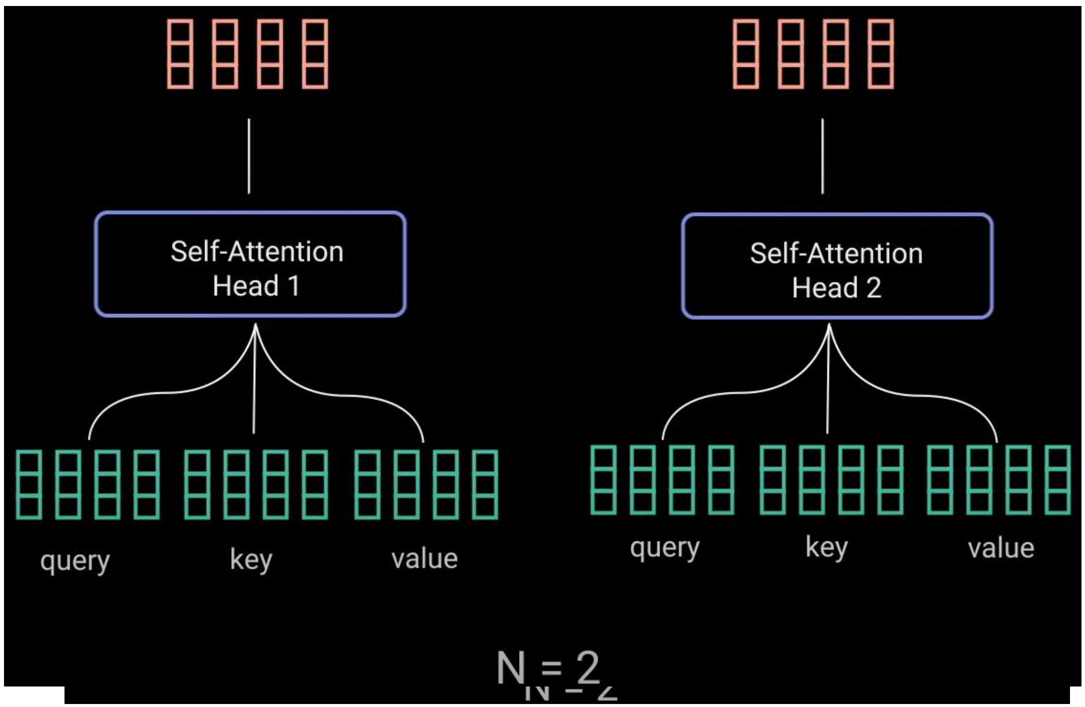

Softmax( ) =
Hi how are you
Hi 0.7 0.1 0.1 0.1
how 0.1 0.6 0.2 0.1
are 0.1 0.3 0.6 0.1
you 0.1 0.3 0.3 0.3

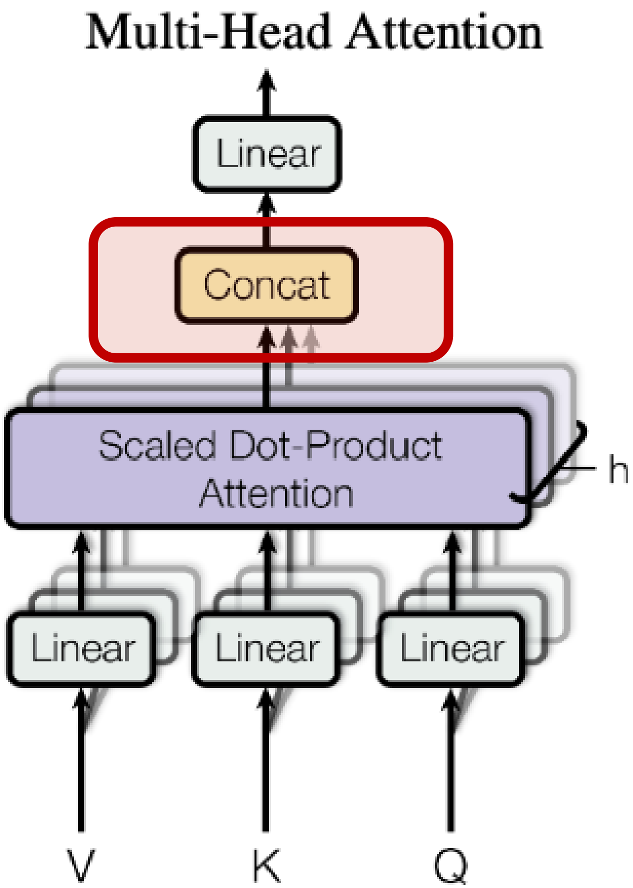

MatMul
SoftMax
Mask (opt.)
Scale
MatMul
Q
K
V

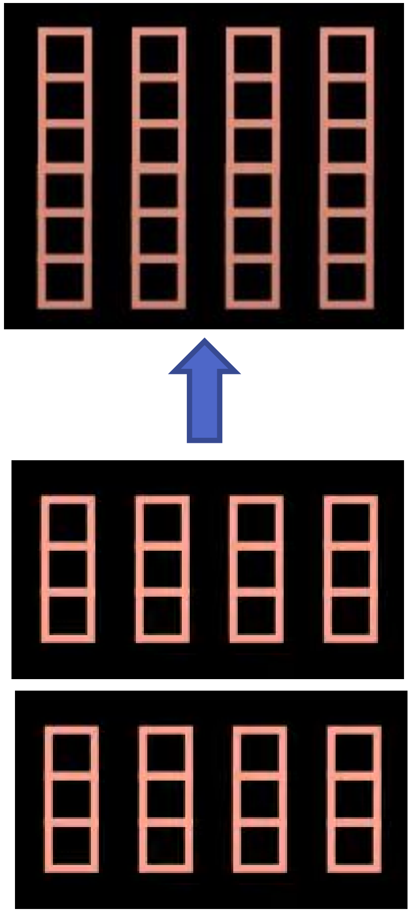

attention weights
value
output
x
=

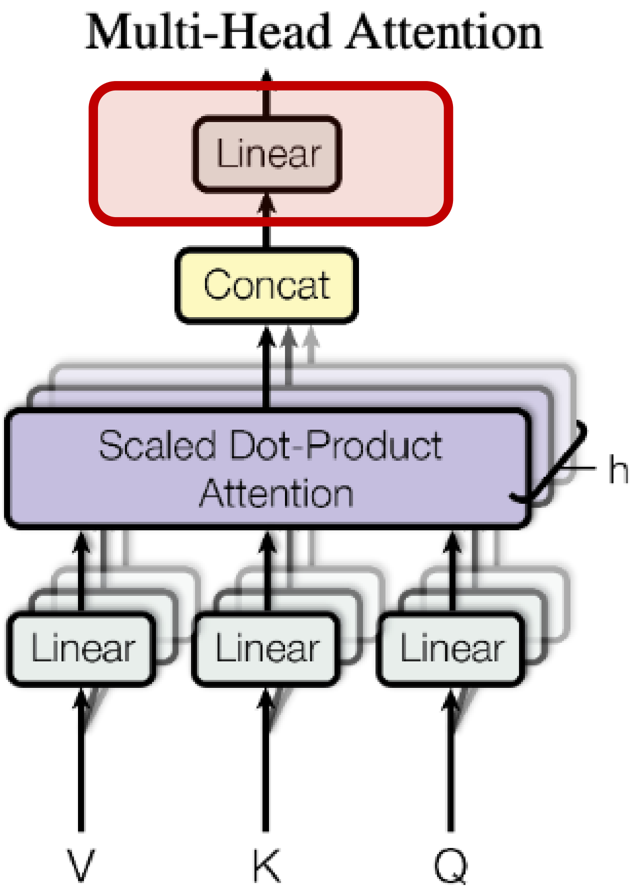

MatMul
SoftMax
Mask (opt.)
Scale
MatMul
Q
K
V

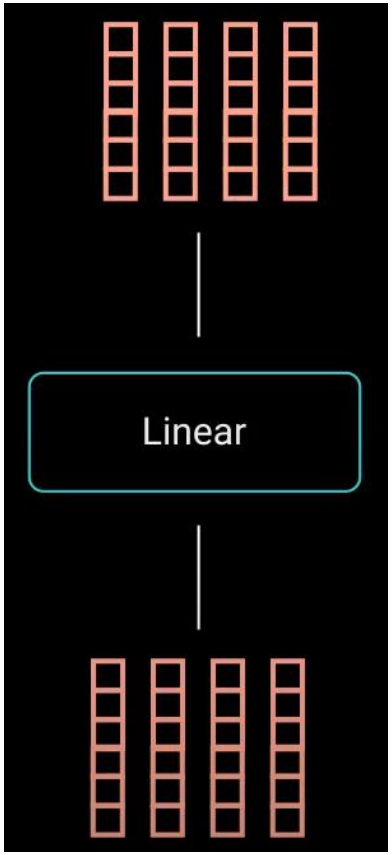
N=2


Self-Attention Head 1
Self-Attention Head 2
query
key
value
query
key
value
N = 2

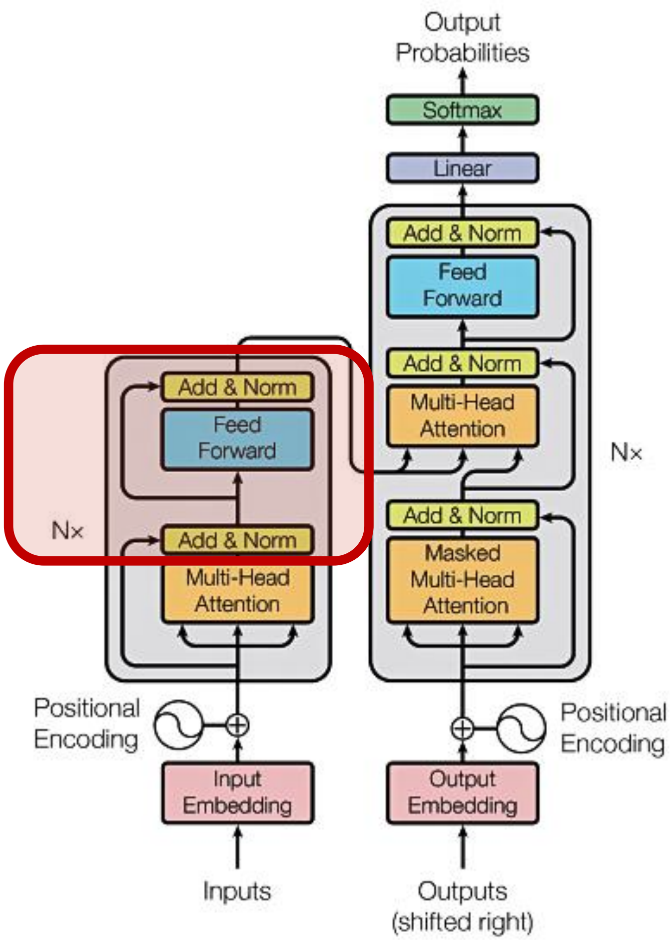

Multi-Head Attention
Linear
Concat
Scaled Dot-Product
Attention
h
Linear
Linear
Linear
V
K
Q

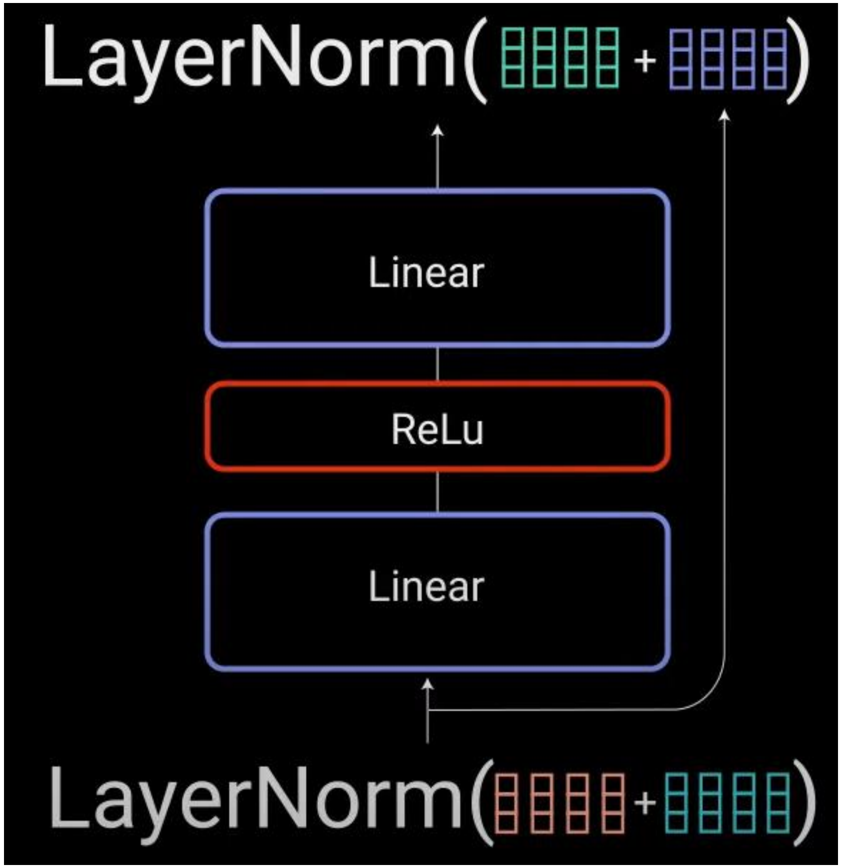



Multi-Head Attention
Linear
Concat
Scaled Dot-Product Attention
h
Linear
Linear
Linear
V
K
Q

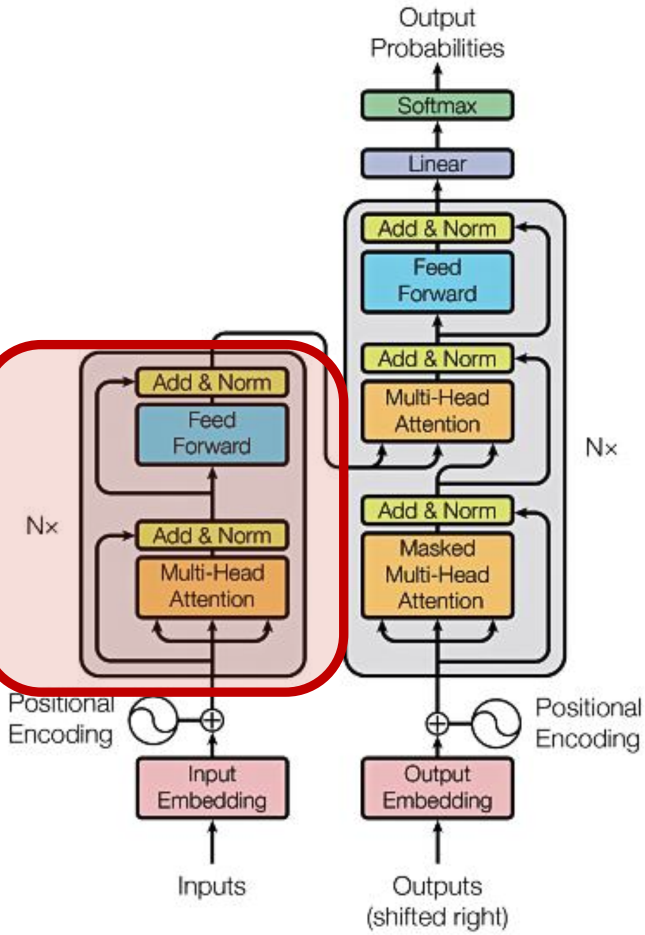

Linear

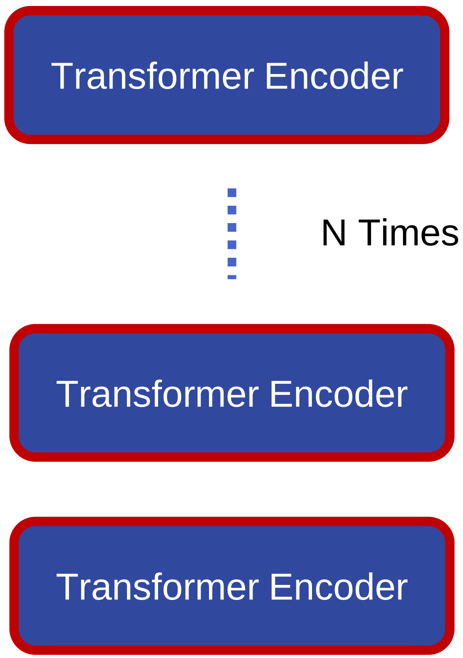
Residual Connection, Layer Normalization and Pointwise FW NW



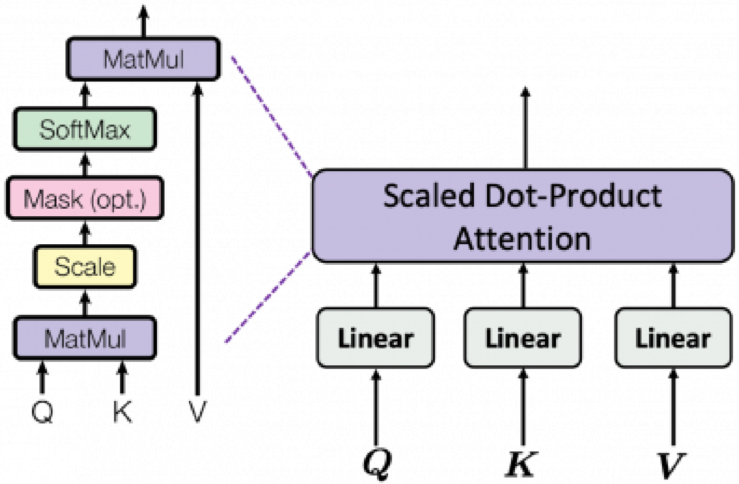


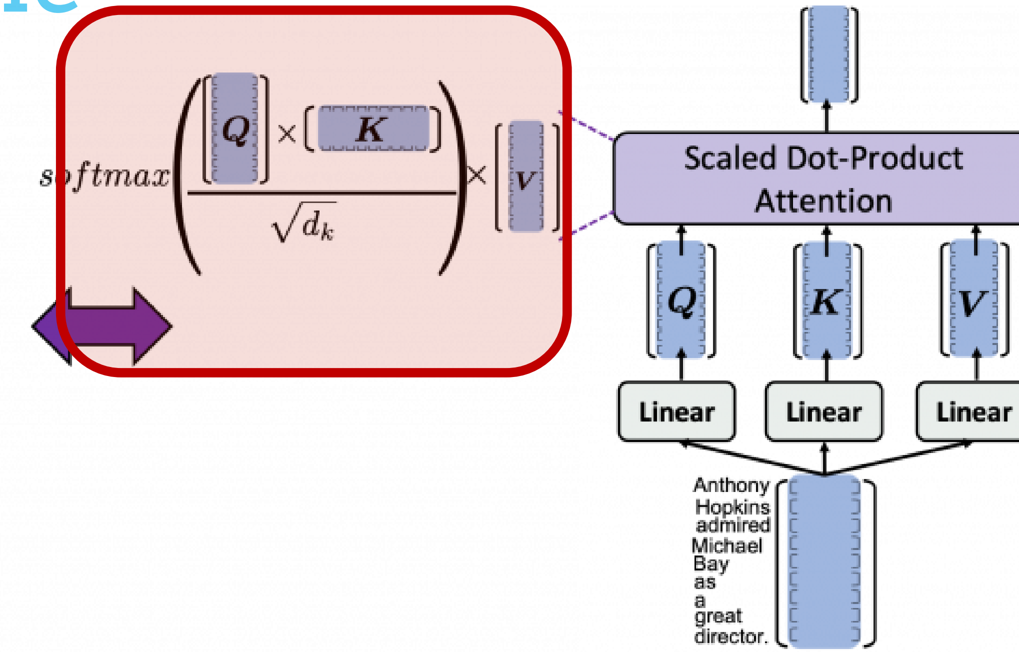
Transformer Encoder

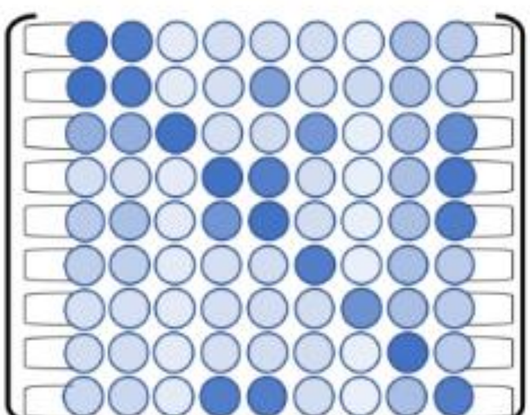




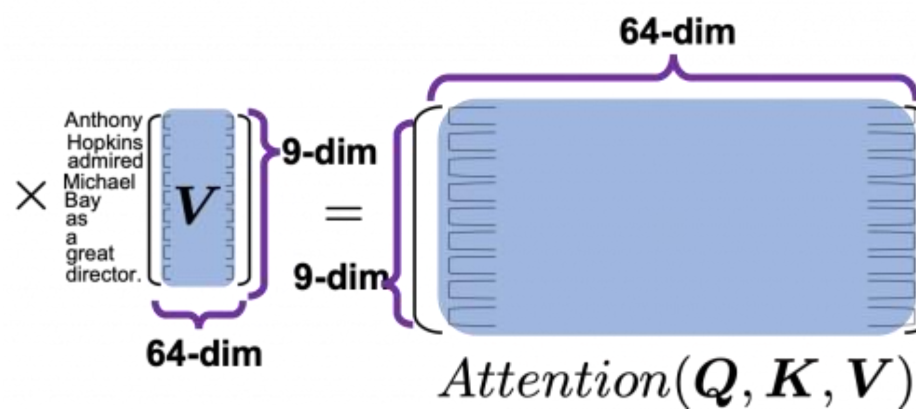
Encoder Example







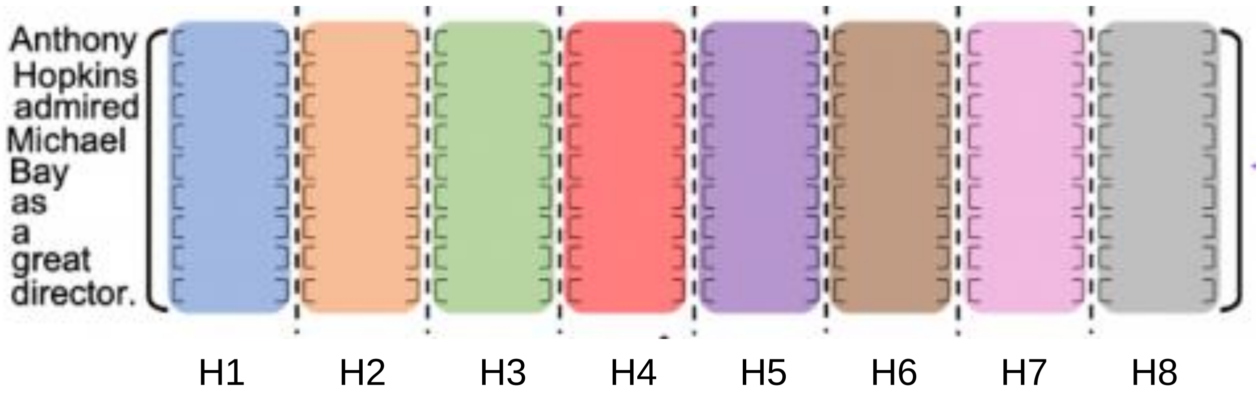
Importantly, the sum of each row is 1.



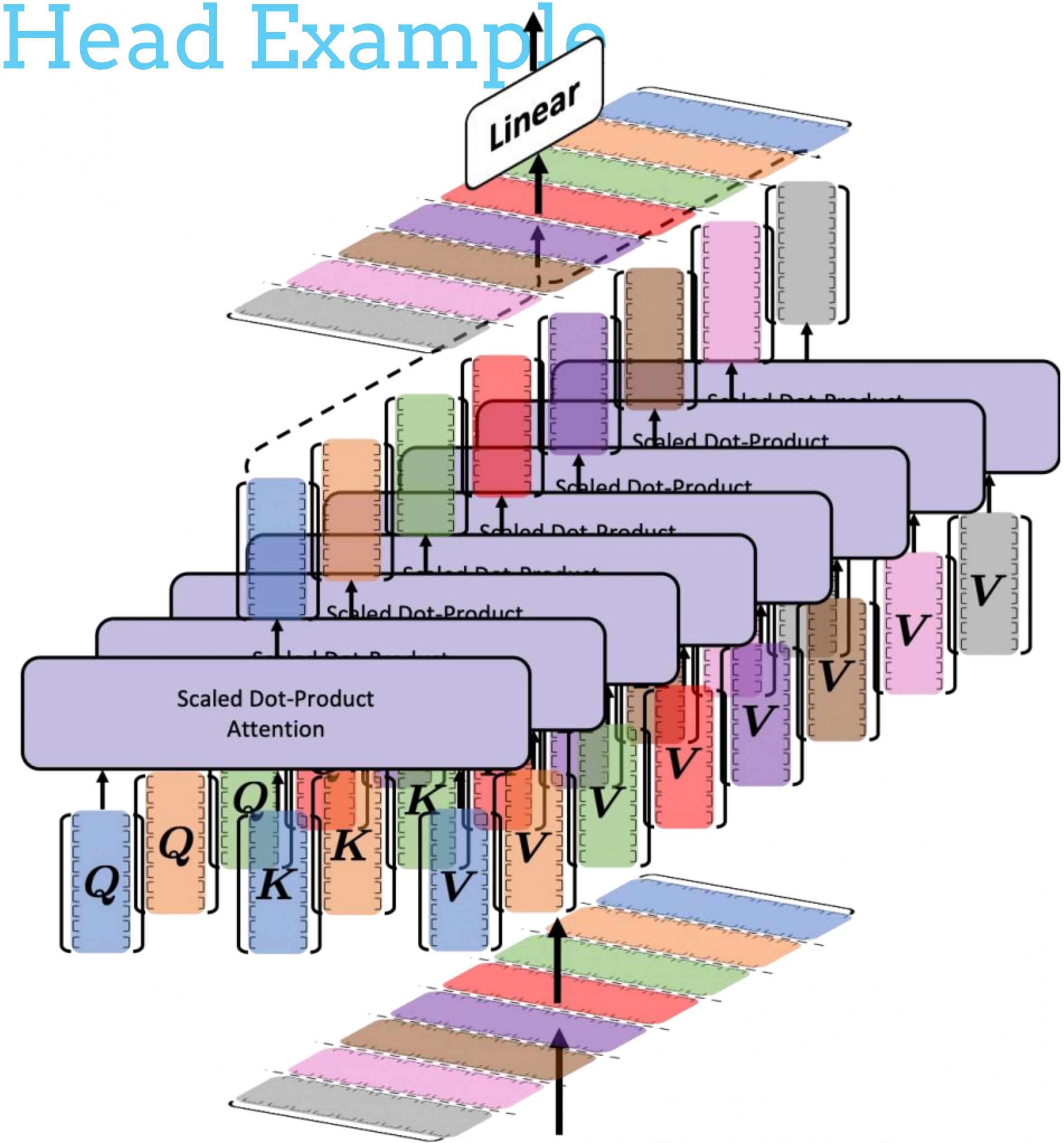
Credit: Y Tamura

Multi Head Example

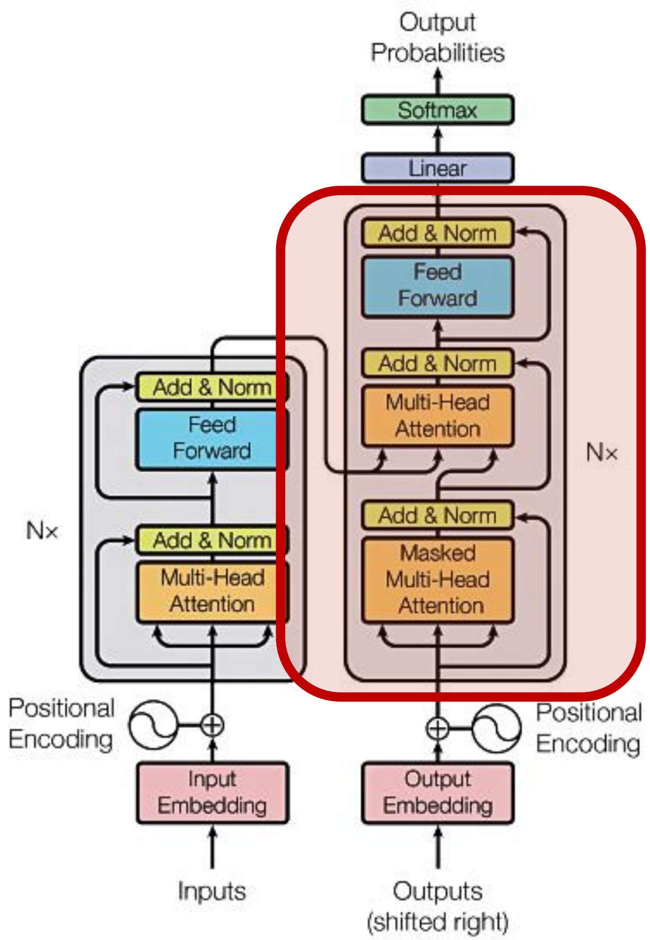
- No of tokens=9
- Token encoded dimension =512
- Split each token into 8 equal parts of 64 dimension



Multi Head Example

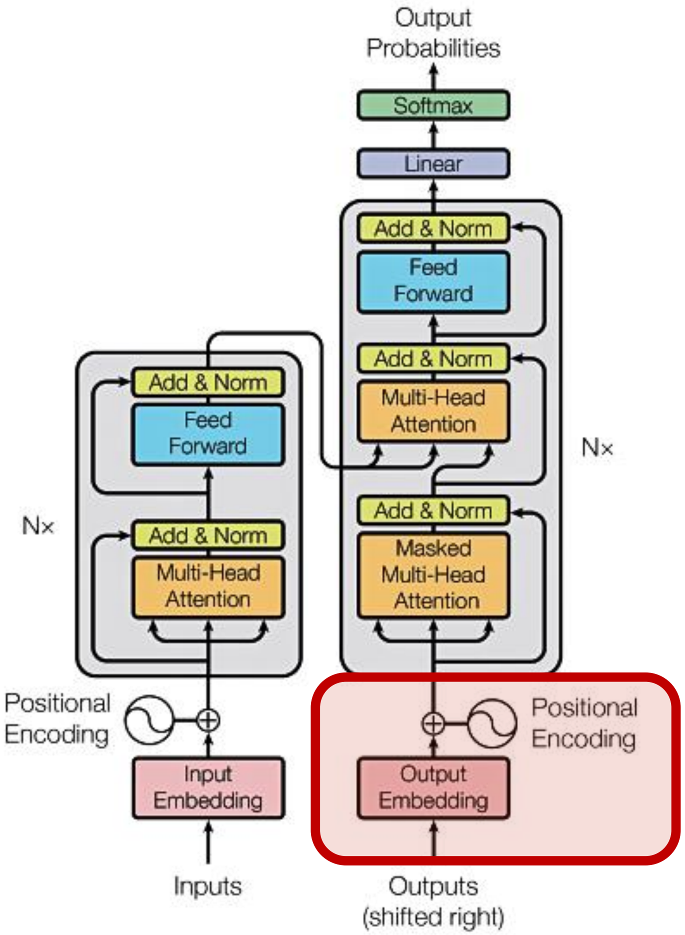


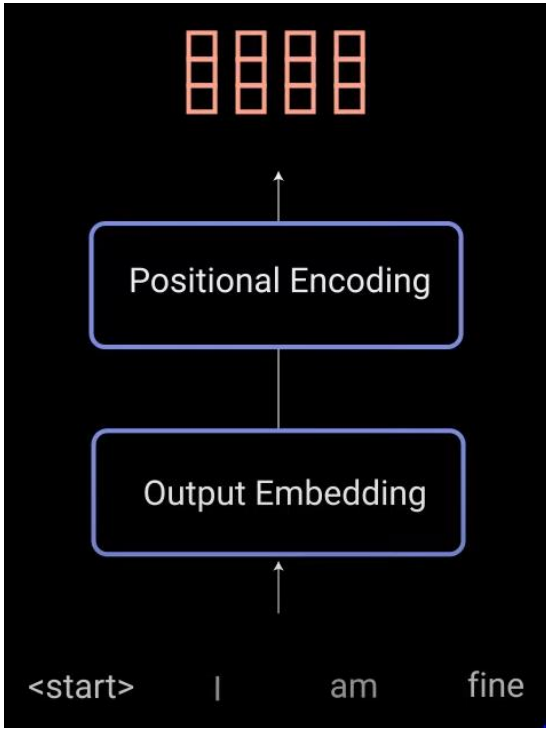
Transformer Decoder



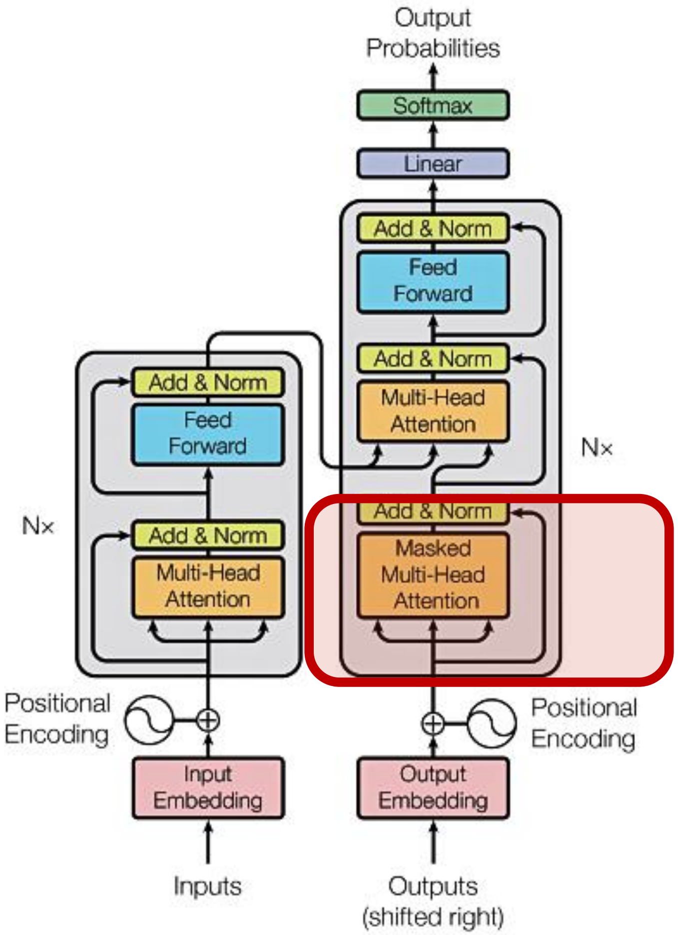
- Decoder is to generate text
- It is auto regressive – Takes previous outputs as inputs
- Stops generation when <end> token is generated.

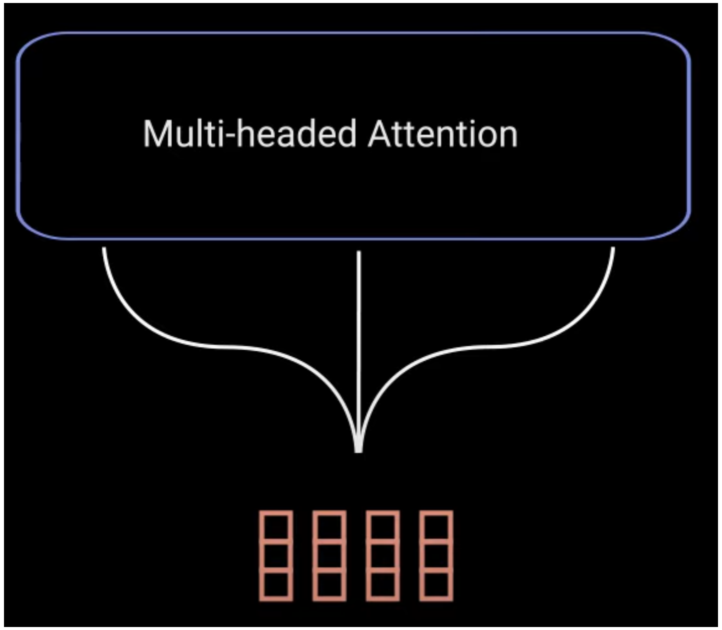
Output Embedding and positional Encoding





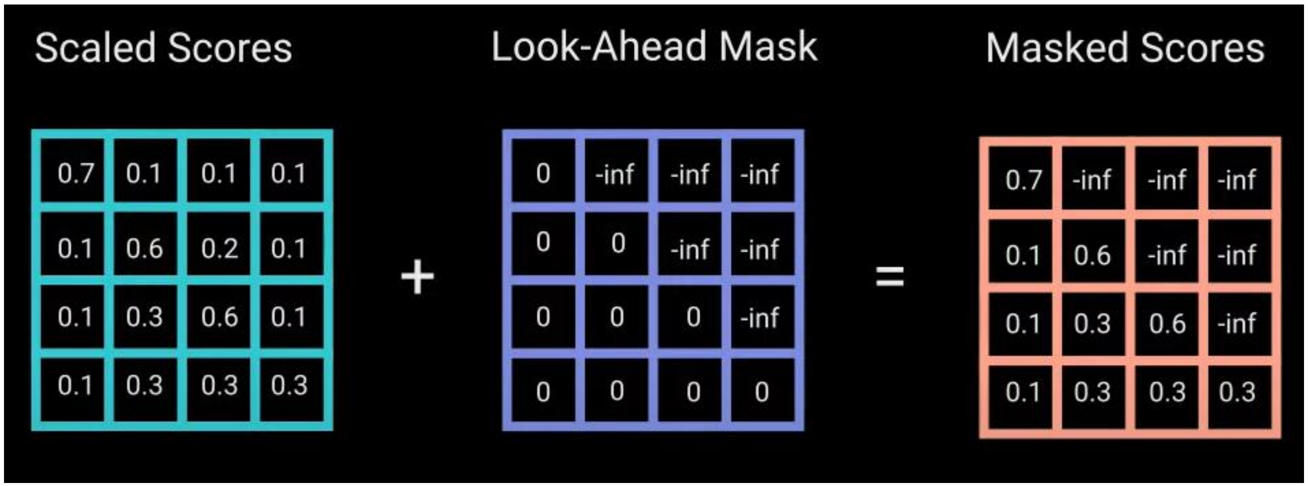
Decoder MH Attention1

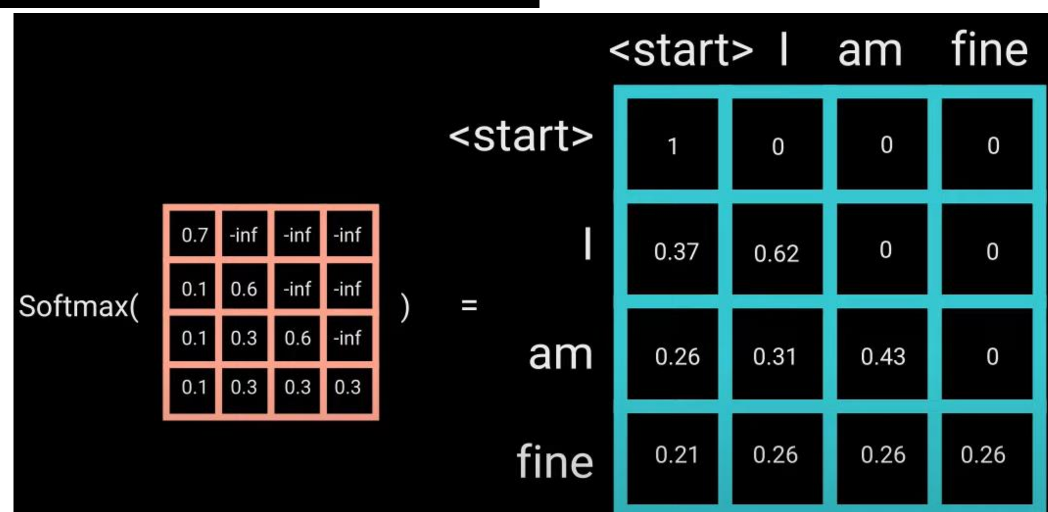




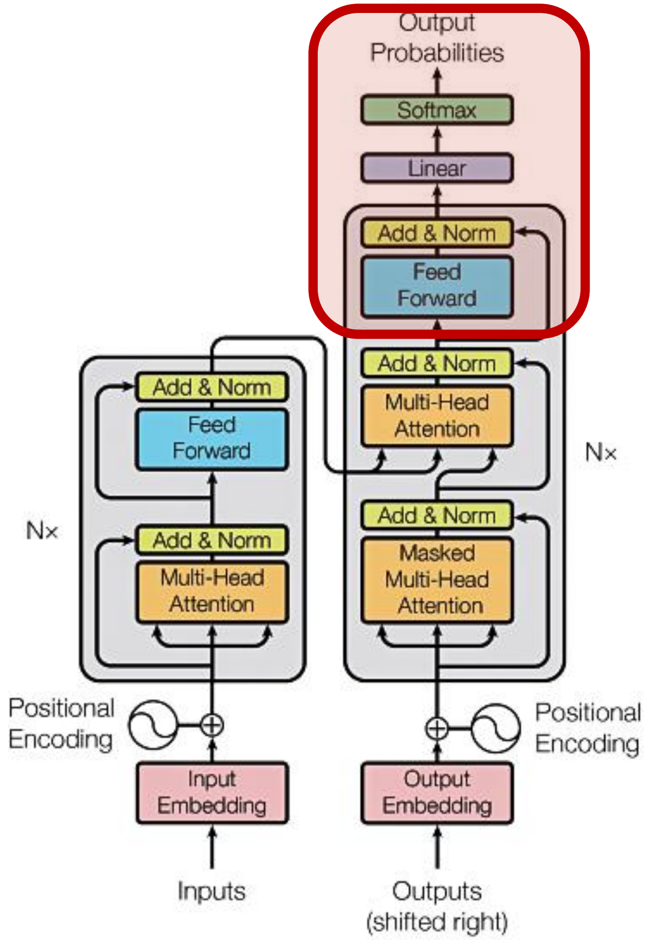
Look ahead Mask

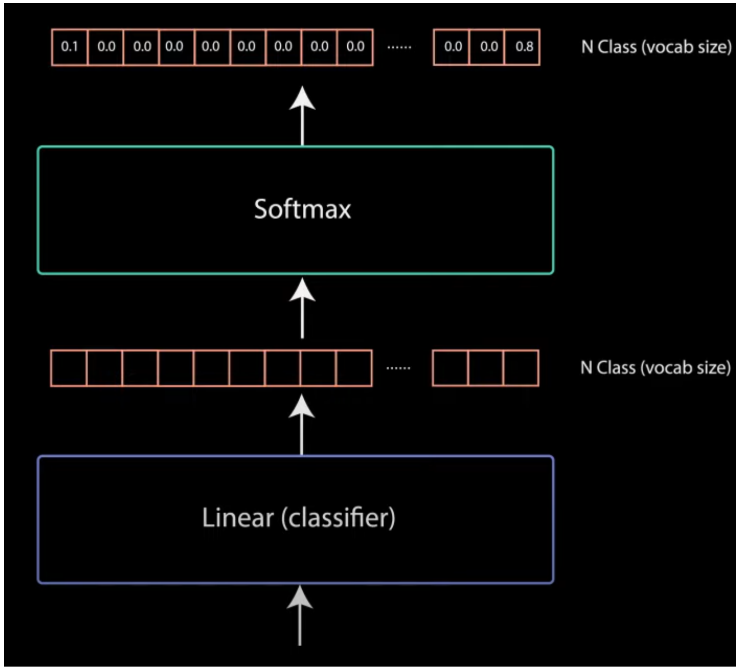
- Prevents decoder from looking at future tokens



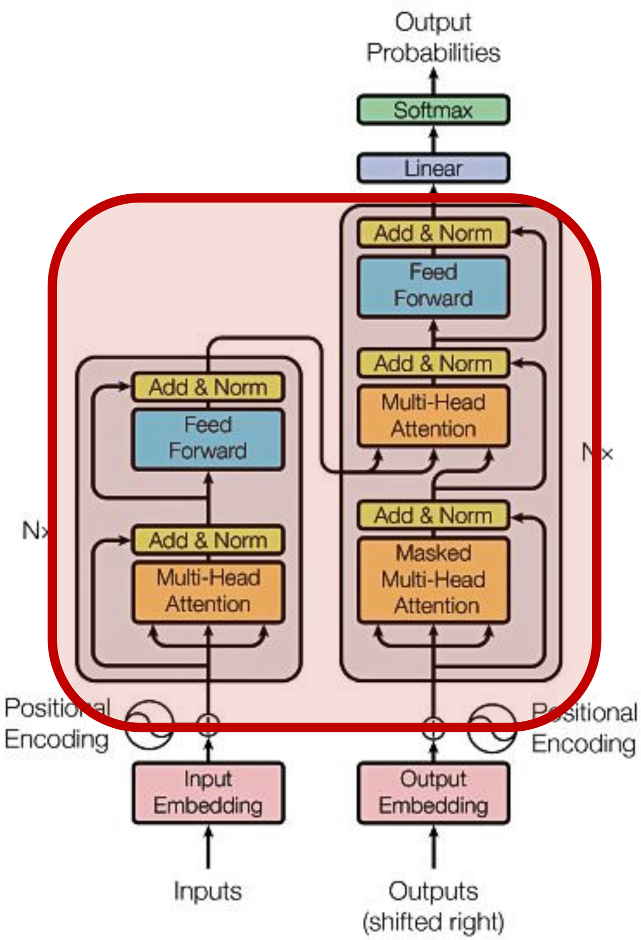


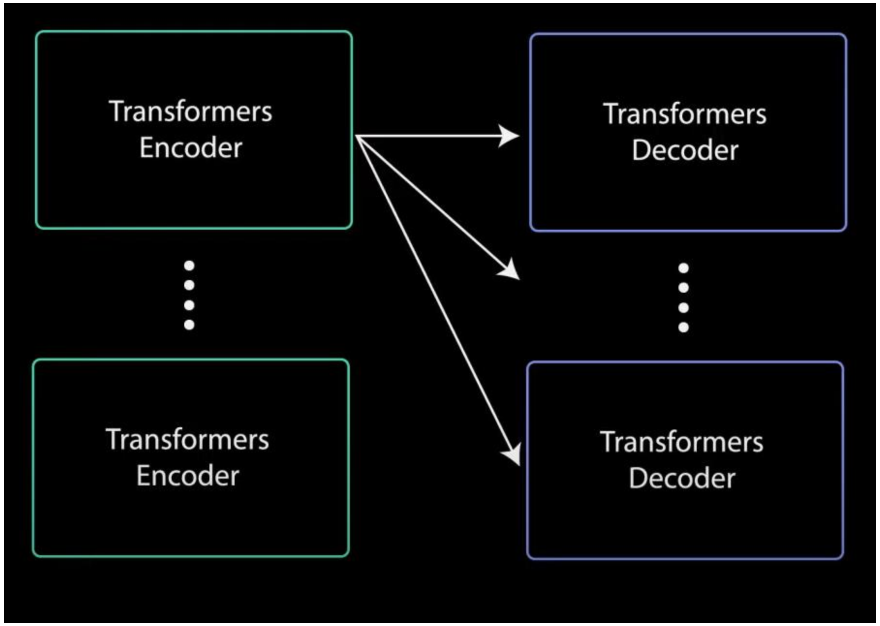
Linear Classifier





Transformer – Encoder & Decoder





Transformers, GPT-2, and BERT

1. A transformer uses an **encoder stack** to model input, and uses **decoder stack** to model output (using input information from encoder side)
2. If we do not have input, we just want to model the “next word”, we can get rid of the encoder side of a transformer and output “next word” one by one. This gives us **GPT**
3. If we are only interested in training a language model for the input for some other tasks, then we do not need the decoder of the transformer, that gives us **BERT**

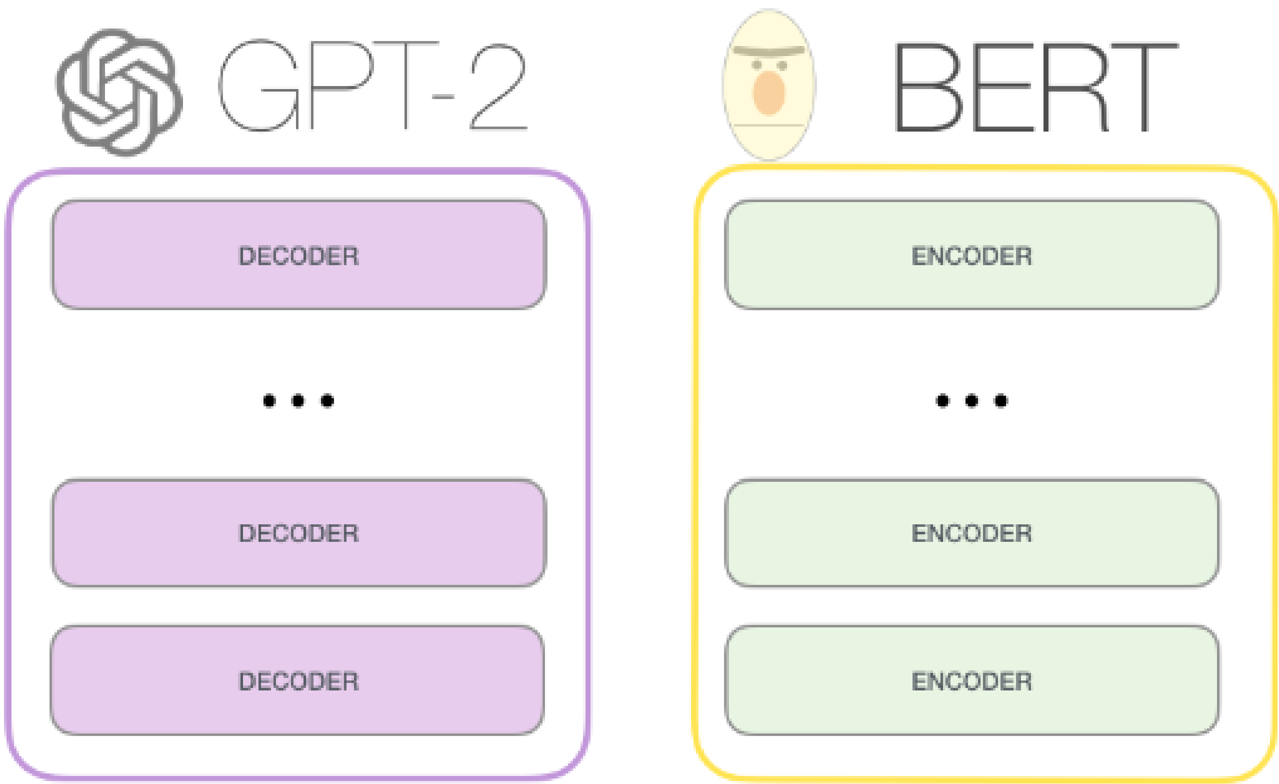
Training a Transformer

- Transformers typically use semi-supervised learning with
  - Unsupervised pretraining over a very large dataset of general text
  - Followed by supervised **fine-tuning** over a focused data set of inputs and outputs for a particular task
- Tasks for pretraining and fine-tuning commonly include:
  - language modeling
  - next-sentence prediction (aka completion)
  - question answering
  - reading comprehension
  - sentiment analysis
  - paraphrasing

Pre-trained Models

- Since training a model requires huge datasets of text and significant computation, researchers often use common pretrained models
- Examples (~ December 2021) include
  - Google's BERT model
  - Huggingface's various Transformer models
  - OpenAI's and GPT-3 models

GPT-2 & BERT

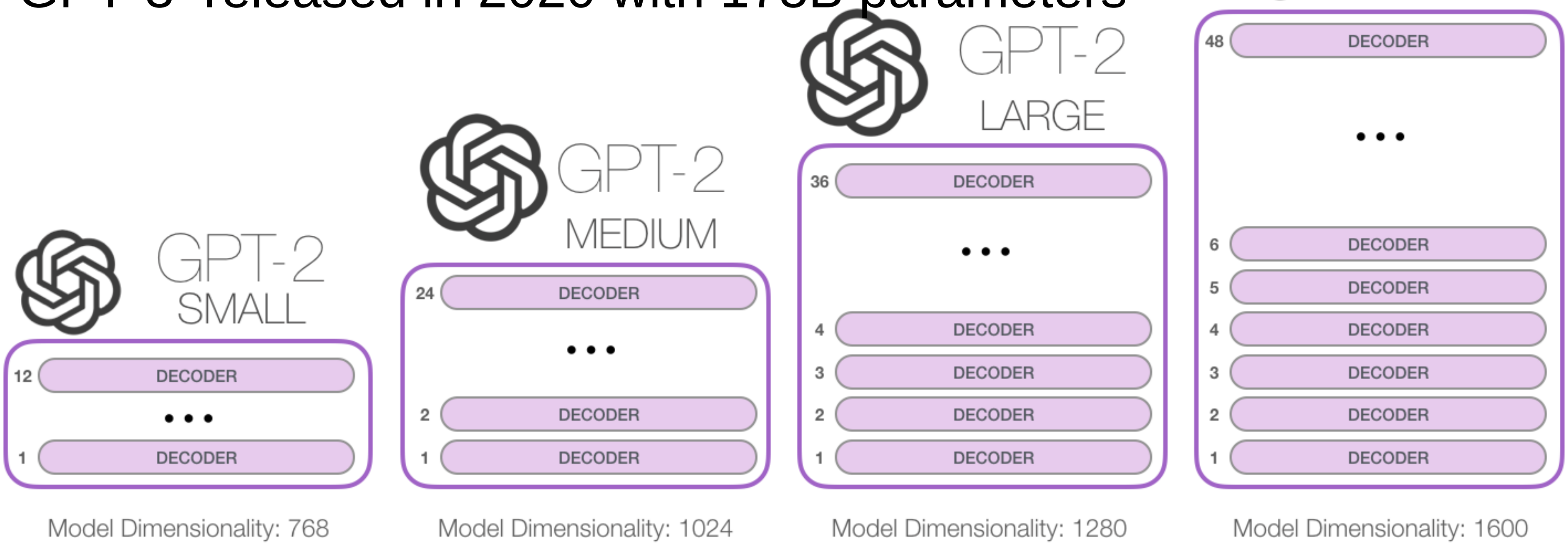

GPT-2
BERT
DECODER
DECODER
DECODER
ENCODER
ENCODER
ENCODER

GPT - Varients

GPT released June 2018

GPT-2 released Nov. 2019 with 1.5B parameters

GPT-3 released in 2020 with 175B parameters



117M parameters | 345M | 762M | 1542M

ViT – Vision Transformers

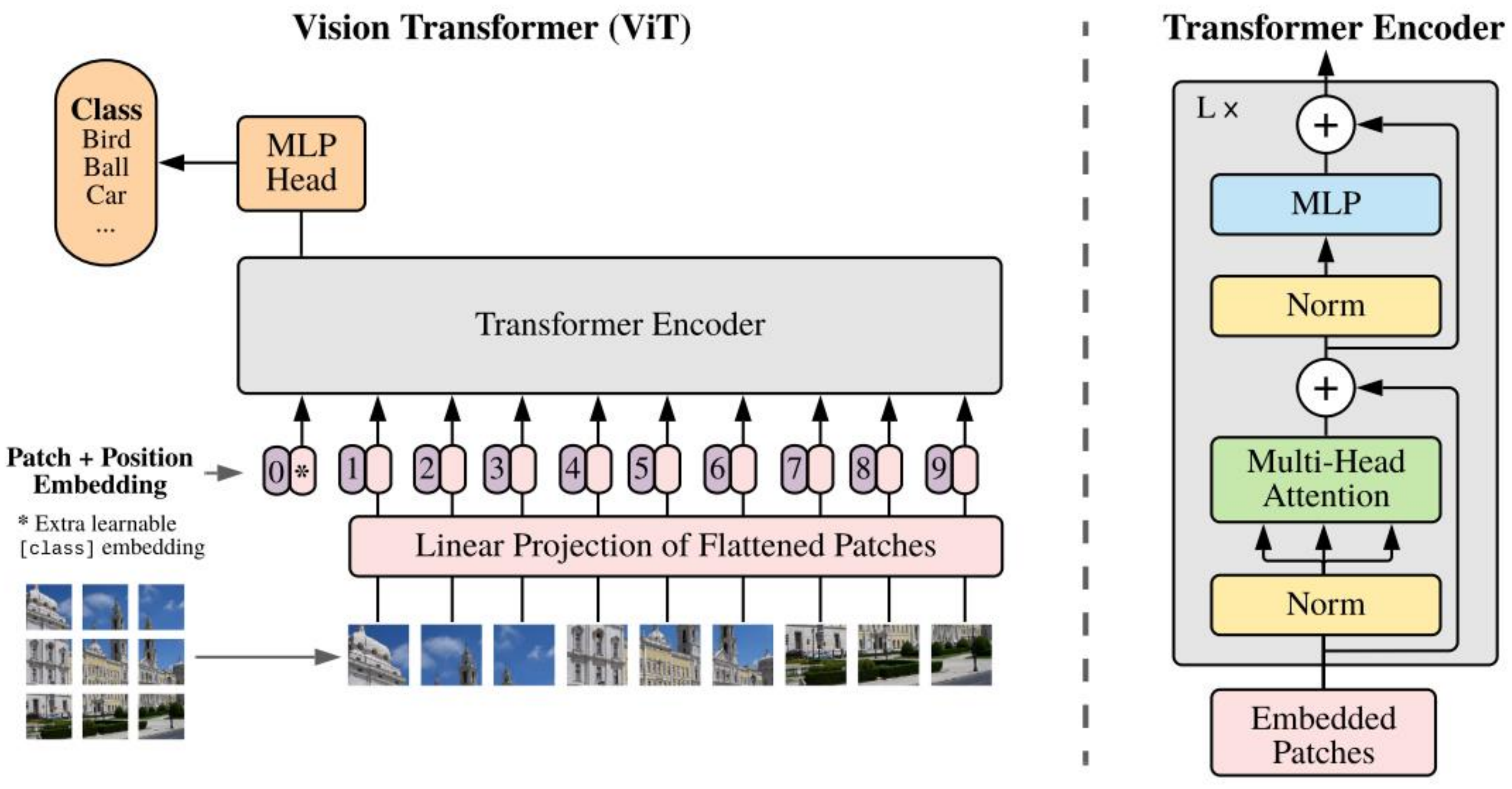
•Divide an input image into 196 (14x14) small images of size (16x16)

•Treat it as embedding in NLP

•Use it as an input for traditional transformer encoder (like in BERT)

•Use 12 transformer layers (Norm, Multi-head attention, etc.)

•Take the last output, use it as input for Dense Layer with 1000 classes

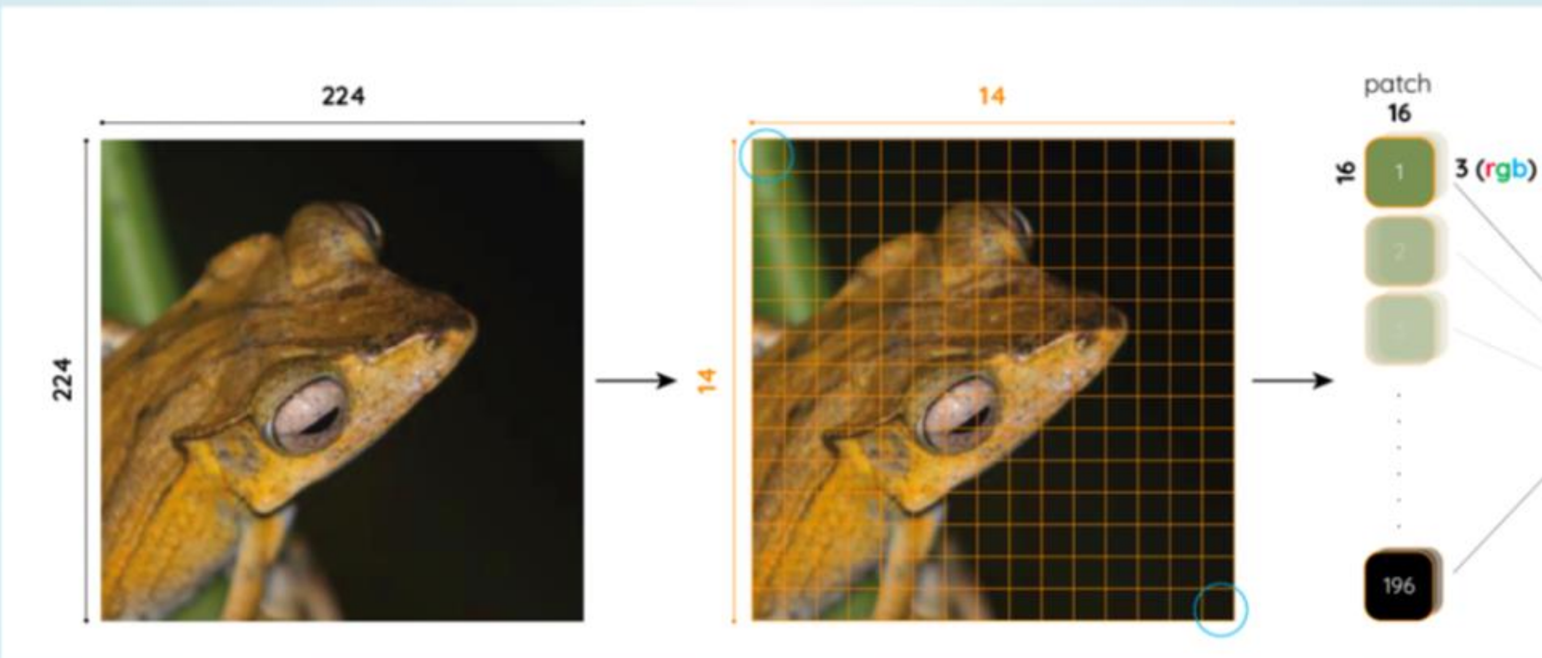

Vision Transformer (ViT)
Class
Bird
Ball
Car
...
MLP Head
Transformer Encoder
Patch + Position Embedding
* Extra learnable [class] embedding
0 *
1
2
3
4
5
6
7
8
9
Linear Projection of Flattened Patches
Transformer Encoder
L ×
MLP
Norm
Multi-Head Attention
Norm
Embedded Patches

Patch Embedding

Image of size $(3, 224, 224)$

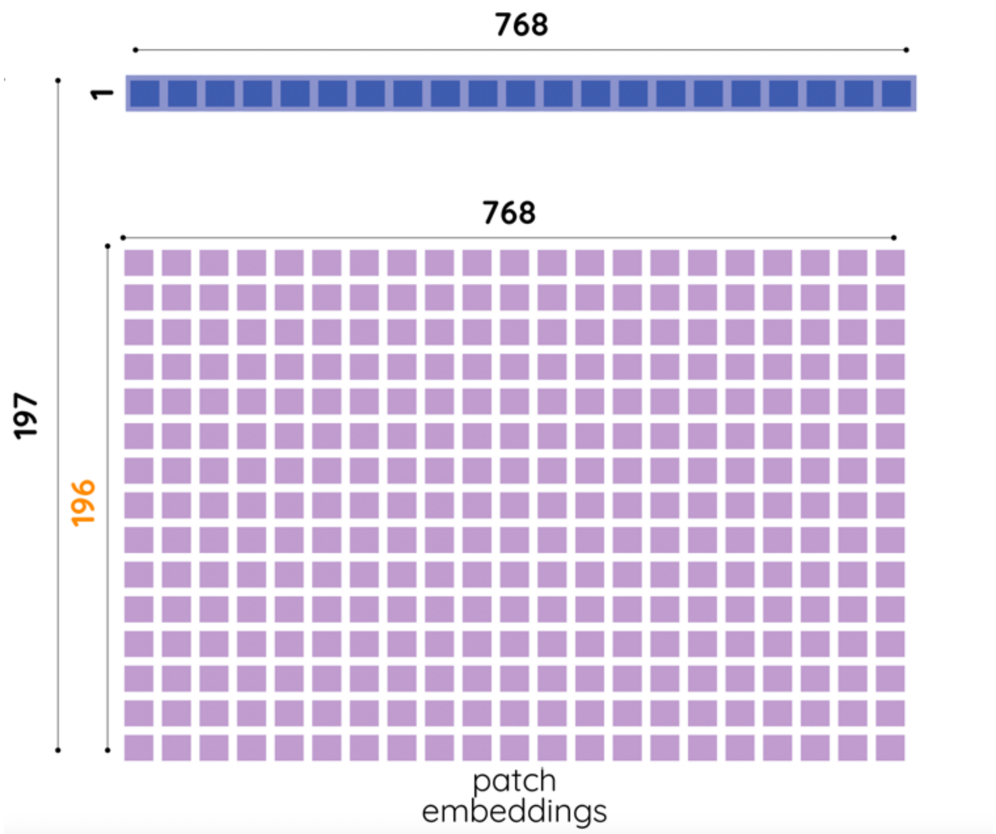
Divided into 196 (14×14) patches of size $3 \times 16 \times 16$

D16*14 = 224 (original image size)



Each patch is converted into a vector of size 3×16×16=768

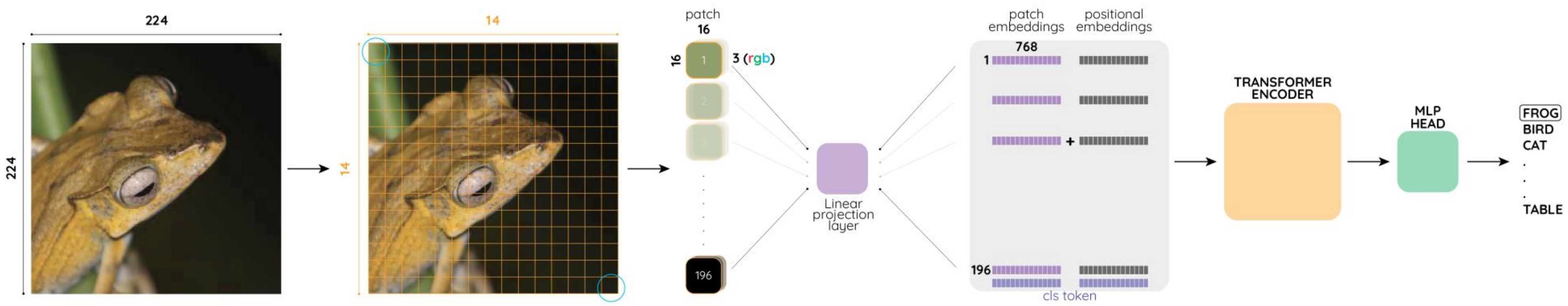
[CLS] Token



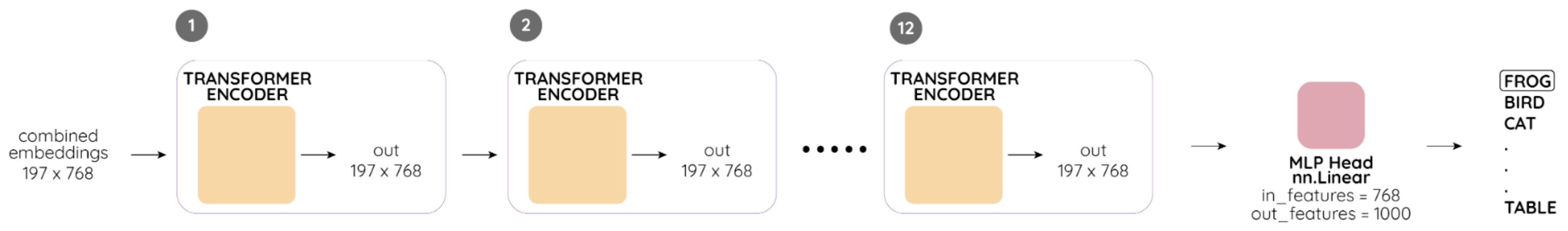
[CLS] token is a vector of size (1,768)(1,768)

The final patch matrix has size (197,768)(197,768), 196 from patches and 1 [CLS] token

ViT - Summary


224
224
14
14
patch
16
16
3 (rgb)
1
196
Linear
projection
layer
patch
embeddings
positional
embeddings
768
1
+
196
cls token
TRANSFORMER
ENCODER
MLP
HEAD
FROG
BIRD
CAT
TABLE

ViT Layers



combined embeddings 197 x 768
1
TRANSFORMER ENCODER
out 197 x 768
2
TRANSFORMER ENCODER
out 197 x 768
12
TRANSFORMER ENCODER
out 197 x 768
MLP Head
nn.Linear
in_features = 768
out_features = 1000
FROG
BIRD
CAT
TABLE

How Good is ViT?

- Worse than Resnet when trained just on ImageNet
- Performance improved when pre-trained on big (and I mean it) dataset
- Pretrained outperforms much bigger CNNs

ViT – in Numbers

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

Table 1: Details of Vision Transformer model variants.

	Ours-JFT (ViT-H/14)	Ours-JFT (ViT-L/16)	Ours-I21K (ViT-L/16)	BiT-L (ResNet152x4)	Noisy Student (EfficientNet-L2)
ImageNet	**88.55** $\pm$ 0.04	87.76 $\pm$ 0.03	85.30 $\pm$ 0.02	87.54 $\pm$ 0.02	88.4/88.5*
ImageNet ReaL	**90.72** $\pm$ 0.05	90.54 $\pm$ 0.03	88.62 $\pm$ 0.05	90.54	90.55
CIFAR-10	**99.50** $\pm$ 0.06	99.42 $\pm$ 0.03	99.15 $\pm$ 0.03	99.37 $\pm$ 0.06	–
CIFAR-100	**94.55** $\pm$ 0.04	93.90 $\pm$ 0.05	93.25 $\pm$ 0.05	93.51 $\pm$ 0.08	–
Oxford-IIIT Pets	**97.56** $\pm$ 0.03	97.32 $\pm$ 0.11	94.67 $\pm$ 0.15	96.62 $\pm$ 0.23	–
Oxford Flowers-102	99.68 $\pm$ 0.02	**99.74** $\pm$ 0.00	99.61 $\pm$ 0.02	99.63 $\pm$ 0.03	–
VTAB (19 tasks)	**77.63** $\pm$ 0.23	76.28 $\pm$ 0.46	72.72 $\pm$ 0.21	76.29 $\pm$ 1.70	–
TPUv3-core-days	2.5k	0.68k	0.23k	9.9k	12.3k

Critics

- Better results - only with more data
- The cost of training from scratch is ridiculously high (30k$)
- Is it really that different from Convolutions?