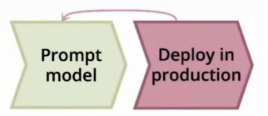
Deep Learning for Computer Vision (Discriminative way)

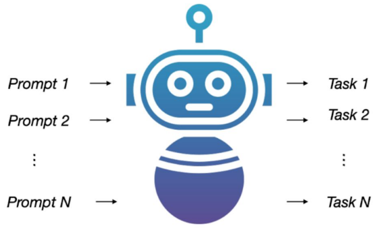
Andrii Liubonko

2024

multimodal LLMs

prompt-based ML



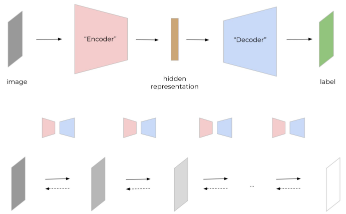


CV foundational models

CLIP, DINO, SAM, ...

Deep Learning based CV

Image → "Deep" Neural Networks → Result



"classical" CV

Image → Human Crafted Features → ML methods → Result

Descriptors: SIFT, SURF, etc.

Linear, SVM, XGBoost, ...

math

- **discriminative** model Pr(**w**|**x**)
- **generative** model Pr(**x**|**w**)
- Pr(**w**|**x**) = Pr(**x**|**w**) * Pr(**w**) / ∫[Pr(**x** | **w**) * Pr(**w**)] d**w**

multimodal LLMs

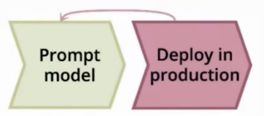
CV foundational models

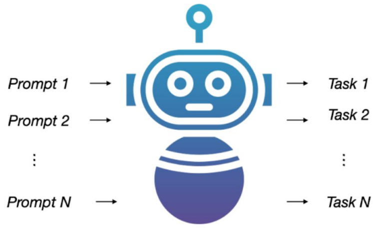
Deep Learning based CV

“classical” CV

math

prompt-based ML





CLIP, DINO, SAM, ...

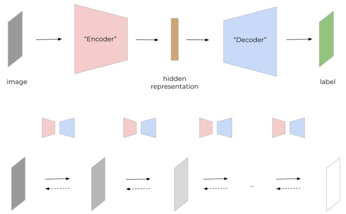


Image → “Deep” Neural Networks → Result

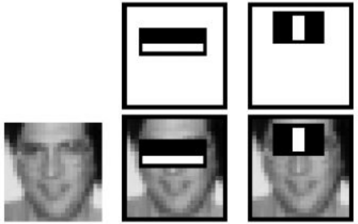
Image → Human Crafted Features → ML methods → Result

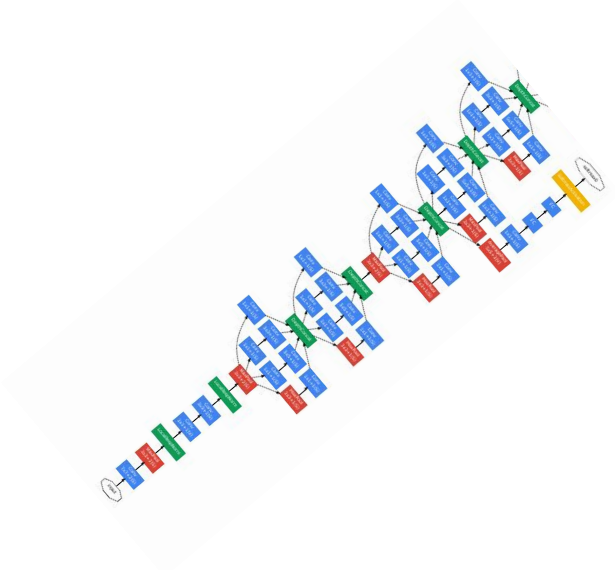
Descriptors: SIFT, SURF, etc.

Linear, SVM, XGBoost, ...

- **discriminative** model Pr(**w**|**x**)
- **generative** model Pr(**x**|**w**)
- Pr(**w**|**x**) = Pr(**x**|**w**) * Pr(**w**) / ∫[Pr(**x** | **w**) * Pr(**w**)] d**w**

Intro





Era of
Human-Crafter
Features

Era of
Deep Learning

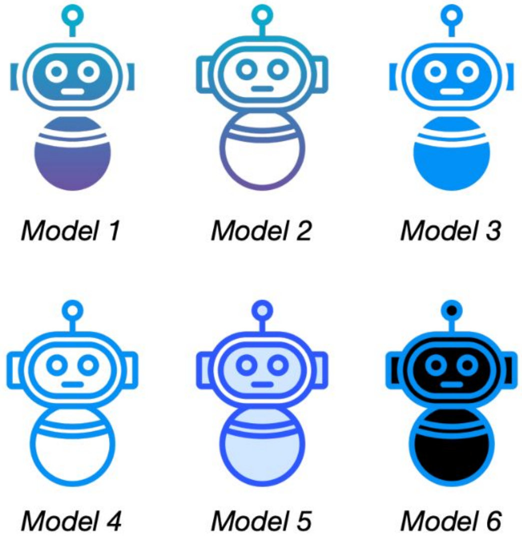
Era of
LLMs
(FoundM)

2012
AlexNet

2022
ChatGPT

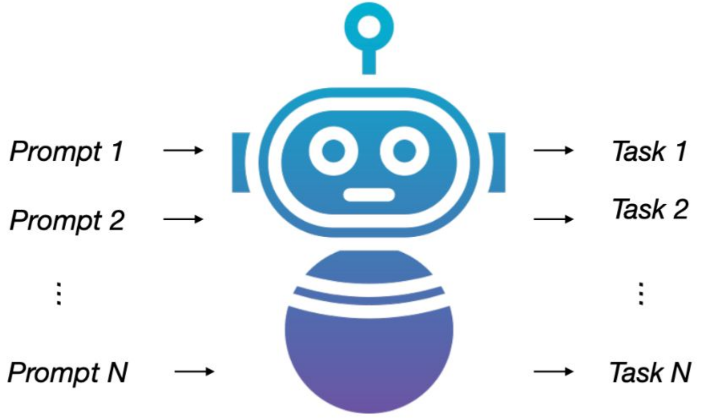
traditional ML

Old days: one model for one purpose

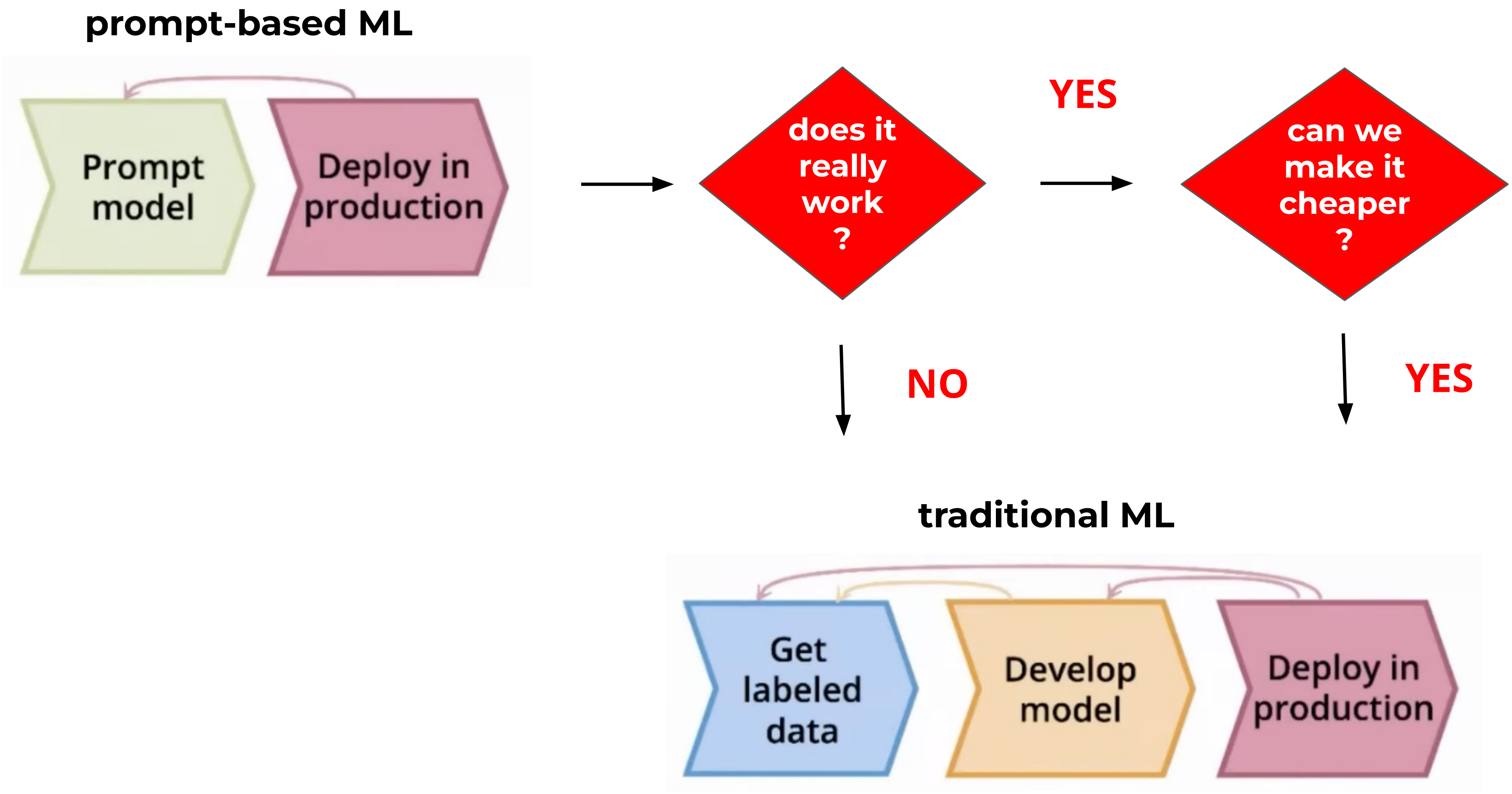


prompt-based ML

Now: one model for multiple purposes

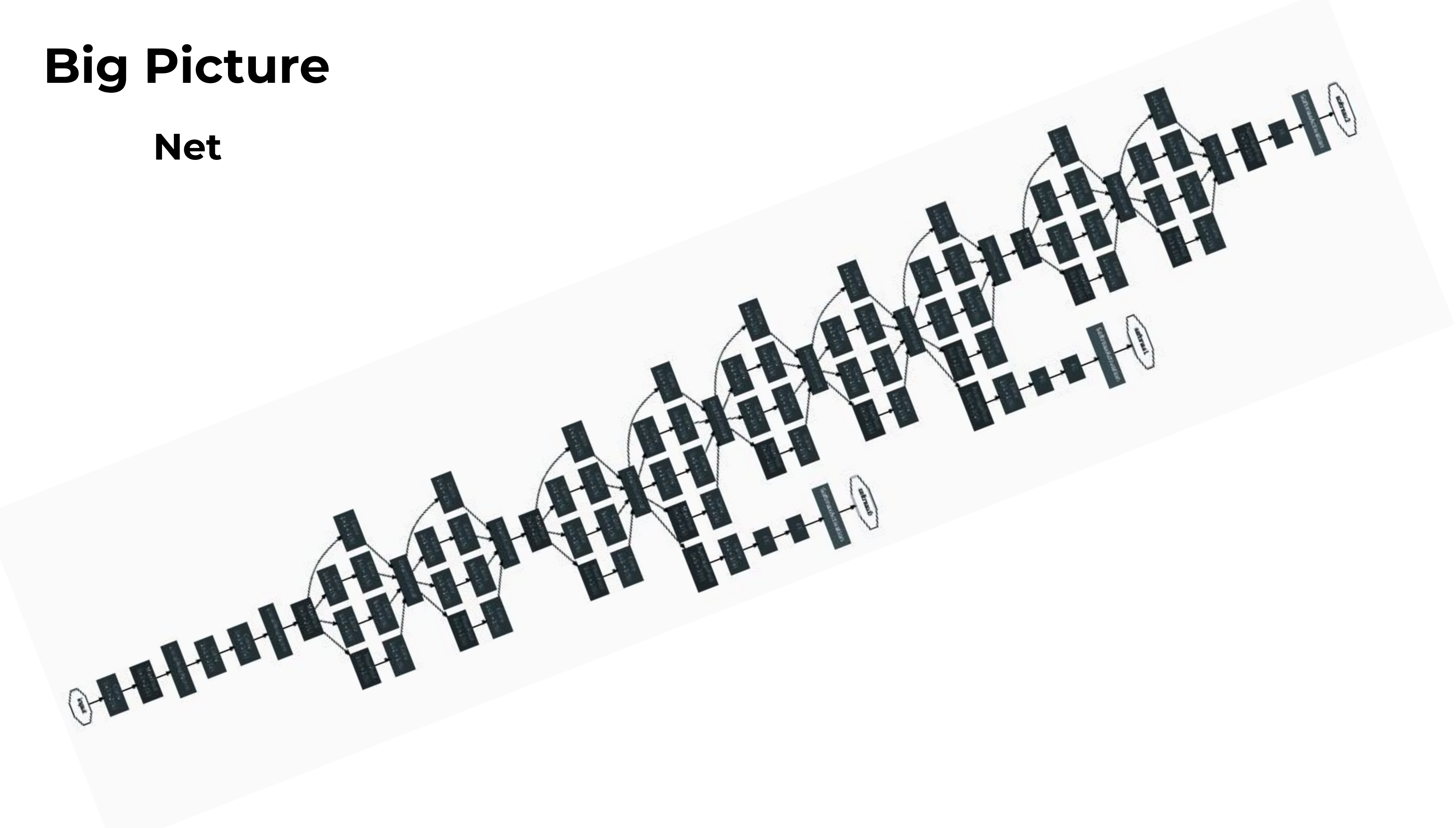


“Foundation model”


prompt-based ML
Prompt model
Deploy in production
does it really work ?
YES
can we make it cheaper ?
NO
YES
traditional ML
Get labeled data
Develop model
Deploy in production

Big Picture

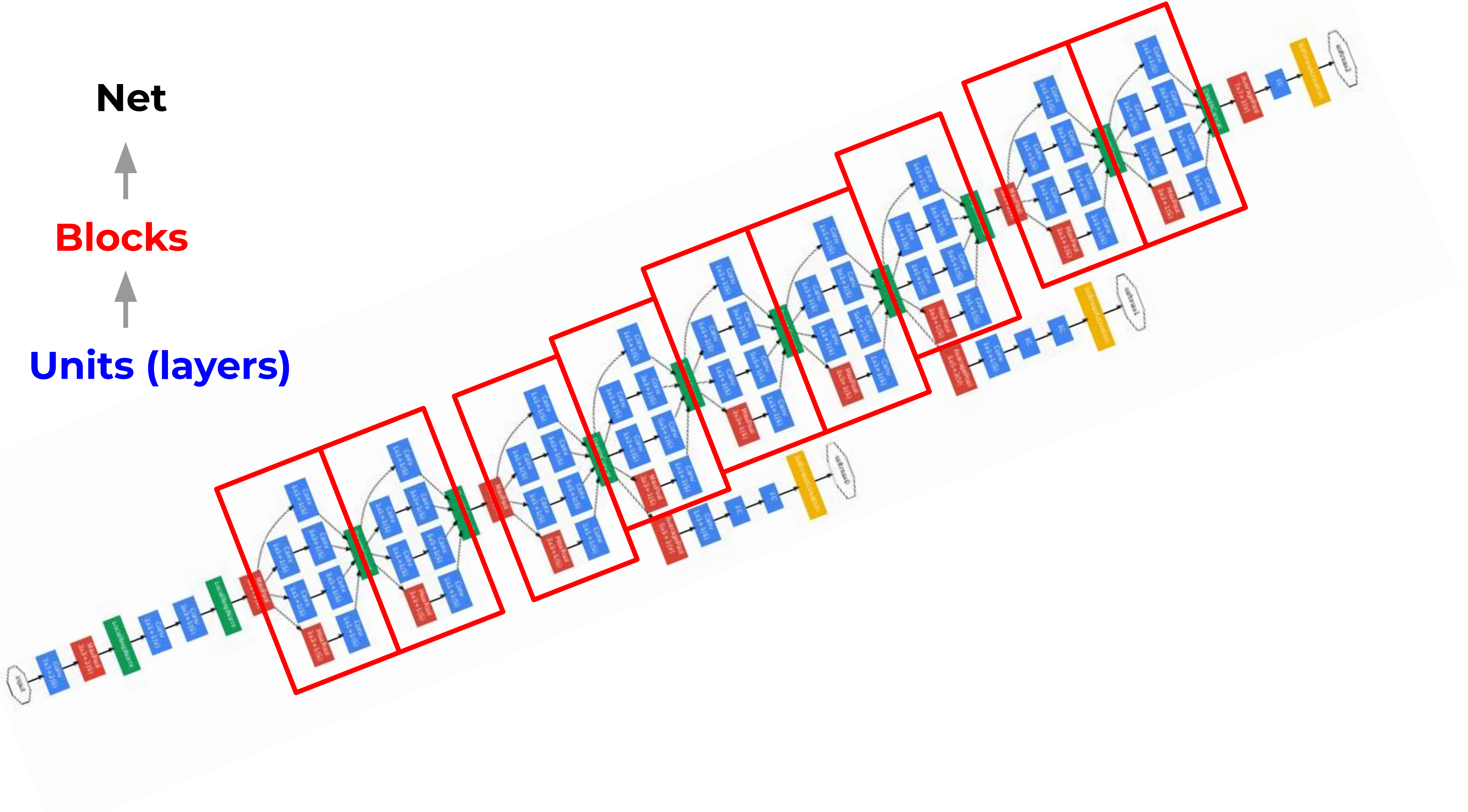
Net



Net

Blocks

Units (layers)

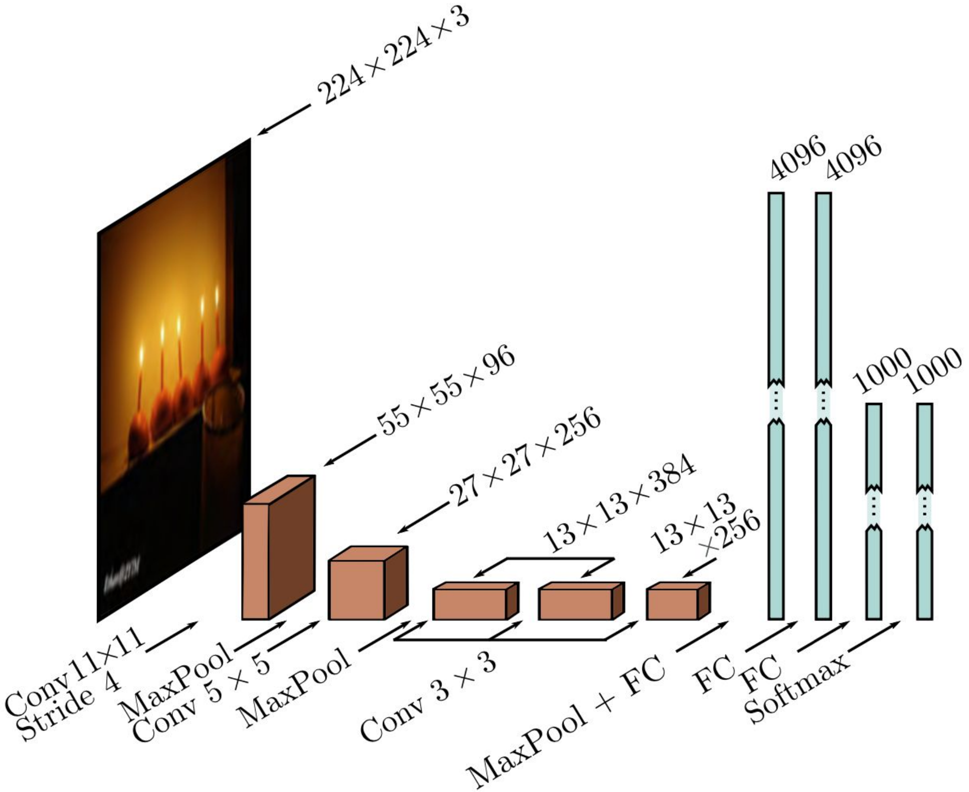


- Layers
  - Layers [Convolution] [Receptive field]
  - Layers [Dilated Convolution, Deformable Convolution]
  - Layers [Group Convolutions and its variants]
  - Layers [Upsampling, Learnable Upsampling]
  - Layers [Normalization, Batch Norm, Dropout]

- Blocks
  - VGG, Inception
  - ResNet*
  - MobileNet*

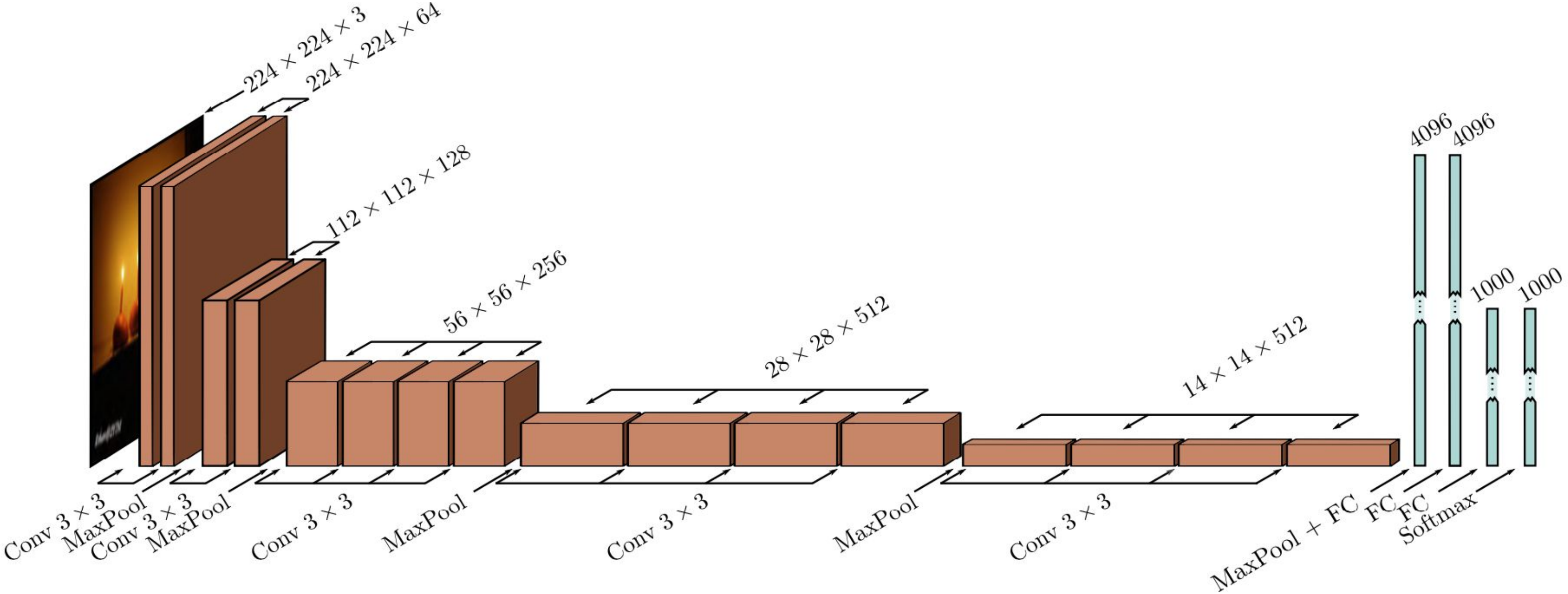
- Architectures
  AlexNet, VGG, Inception,
  ResNet, MobileNet, EfficientNet

Net [AlexNet]



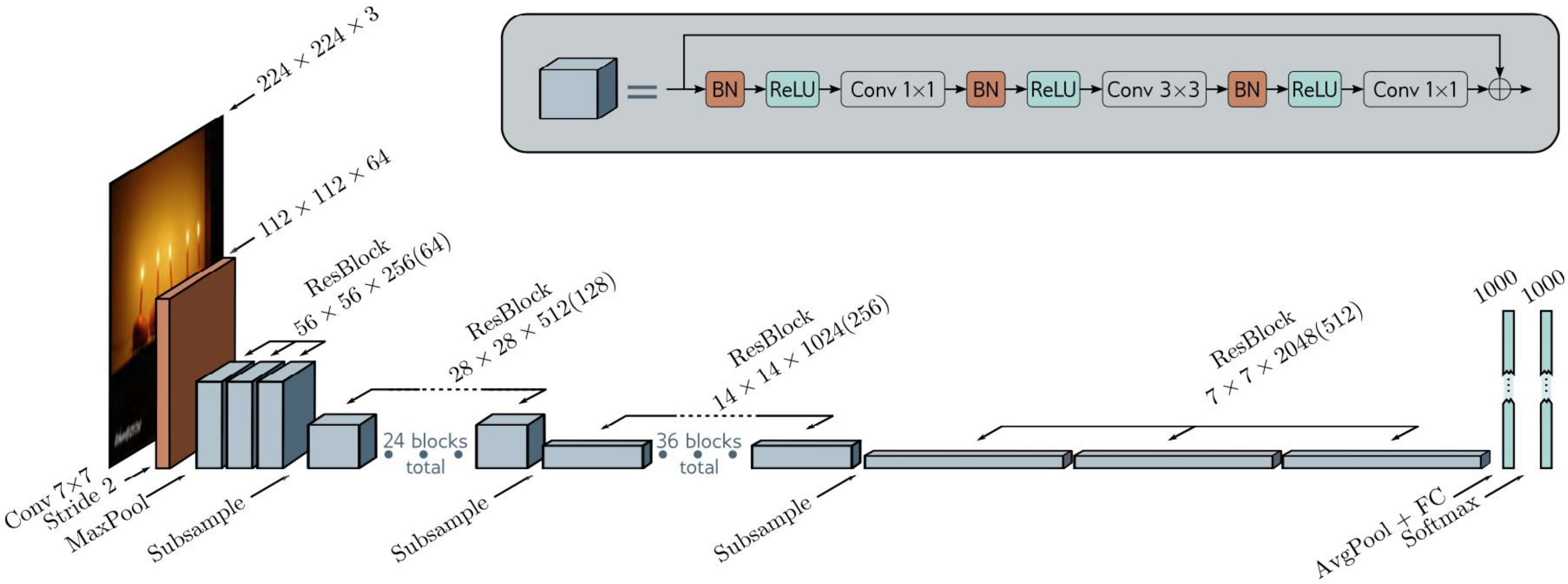
224×224×3
55×55×96
27×27×256
13×13×384
13×13
×256
4096
4096
1000
1000
Conv11×11
Stride 4
MaxPool
Conv 5 × 5
MaxPool
Conv 3 × 3
MaxPool + FC
FC
FC
Softmax

Net [VGG16]



224 × 224 × 3
224 × 224 × 64
112 × 112 × 128
56 × 56 × 256
28 × 28 × 512
14 × 14 × 512
4096
4096
1000
1000
Conv 3 × 3
MaxPool
Conv 3 × 3
MaxPool
Conv 3 × 3
MaxPool
Conv 3 × 3
MaxPool
Conv 3 × 3
MaxPool + FC
FC
FC
Softmax

Net [ResNet]



BN
ReLU
Conv 1×1
BN
ReLU
Conv 3×3
BN
ReLU
Conv 1×1
224 × 224 × 3
112 × 112 × 64
ResBlock
56 × 56 × 256(64)
ResBlock
28 × 28 × 512(128)
ResBlock
14 × 14 × 1024(256)
ResBlock
7 × 7 × 2048(512)
1000
1000
24 blocks total
36 blocks total
Conv 7×7
Stride 2
MaxPool
Subsample
Subsample
Subsample
AvgPool + FC
Softmax

Content of today's lecture

- Attention Intro
  - Brief History
  - Seq2seq Attention
  - Self-Attention
  - Transformer

- Attention in CV
  - SE
  - ViT
  - latest developments

Attention Timeline

Attention Intro

Seq2Seq
- Cho et al. (2014)
- Sutskever et e. (2014)

Align & Translate
- Bahdanau et al. (2015)
- Luong et al. (2015)

Self attention / Transformer
- Vaswahi et al (2017)

BERT
- Devlin et al (2018)

GPT-3
- Brown et al (2020)

GPT-4
- OpenAI (2023)

2014 2015 2016 2017 2018 2019 2020 2021 2022 2023

Attention in CV

Visual attention
- Xu et al. (2015)

SE networks
- Hu et al. (2018)

DETR
- Carion et al.(2020)

ViT
- Dosovitskiy et al.(2020)

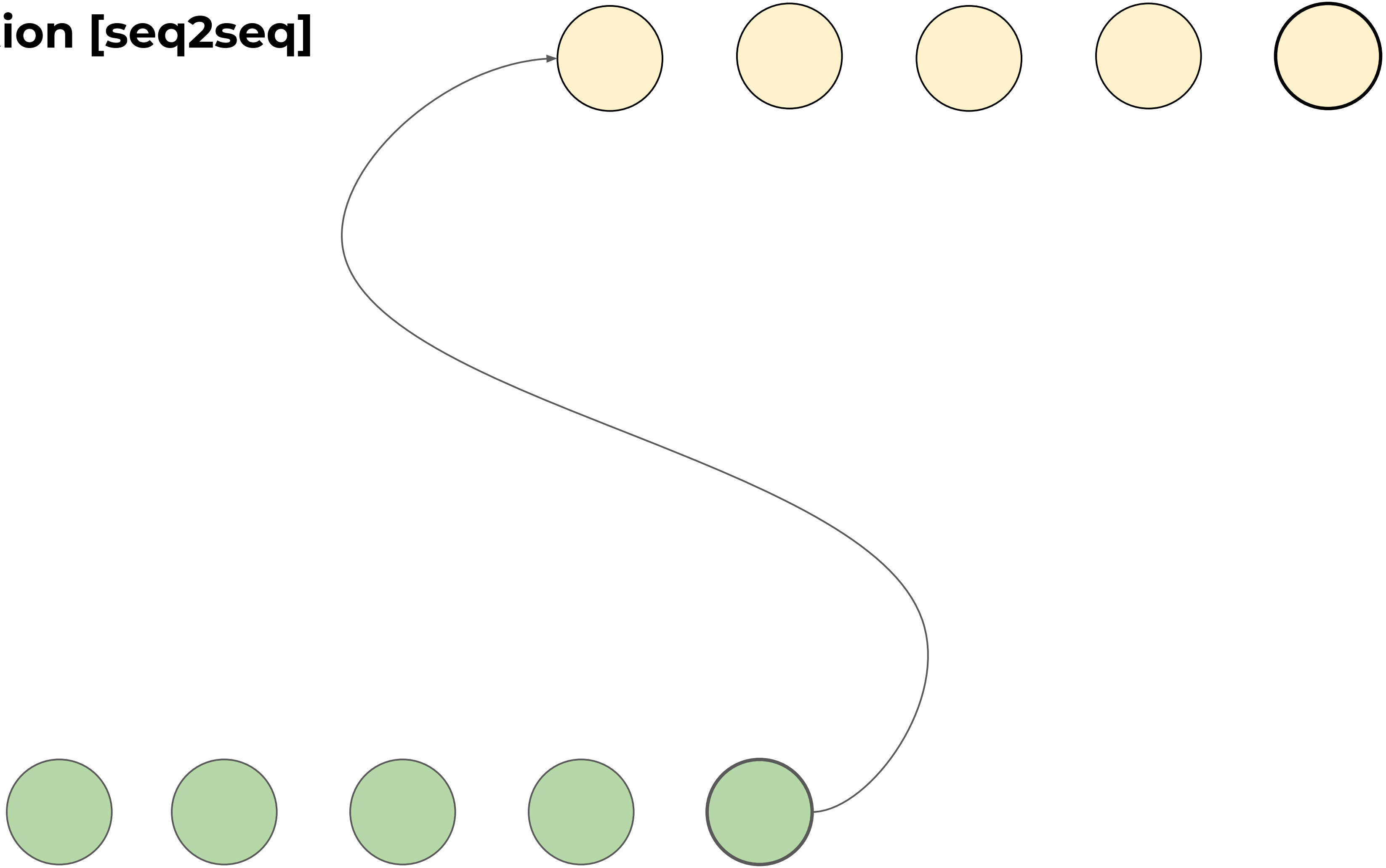
DINO
Caron et al.

MAE
He et al.

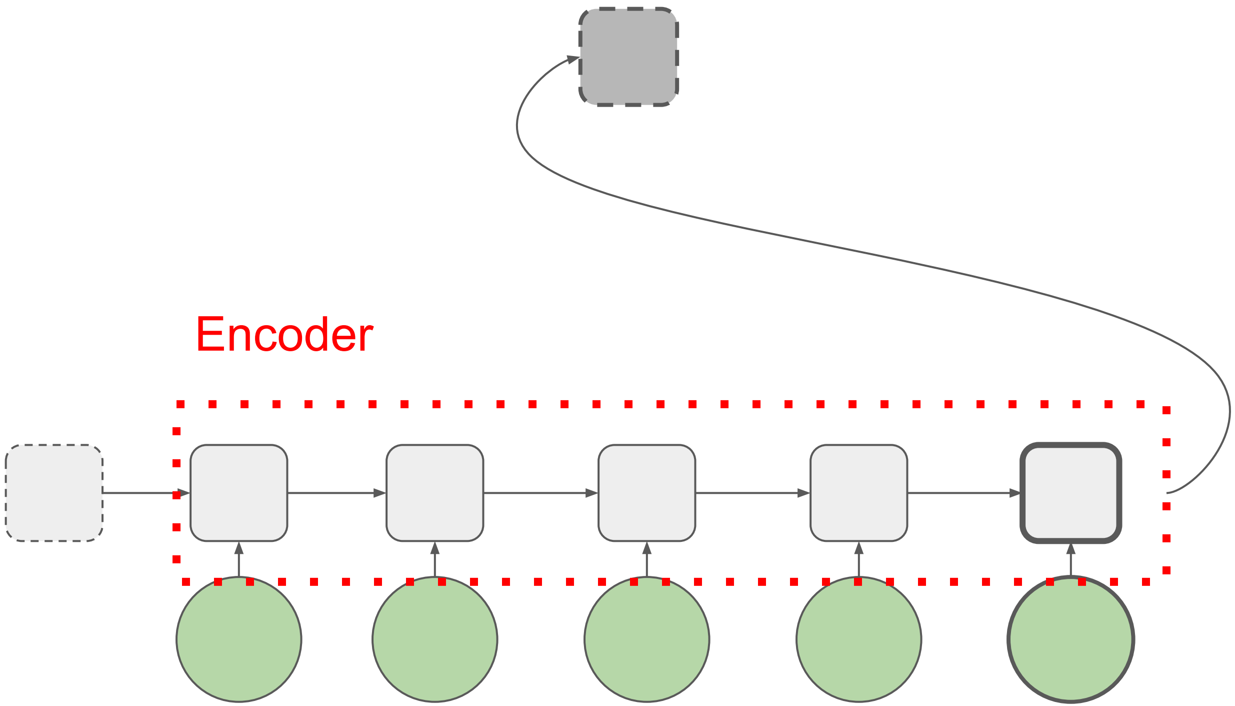
SAM
Meta

Attention [intro]

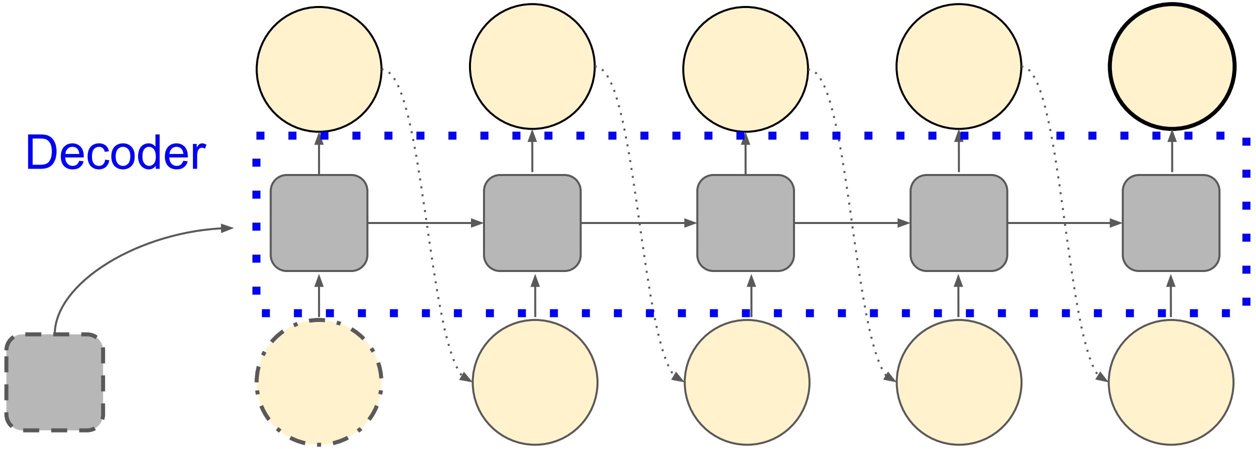
Attention [seq2seq]



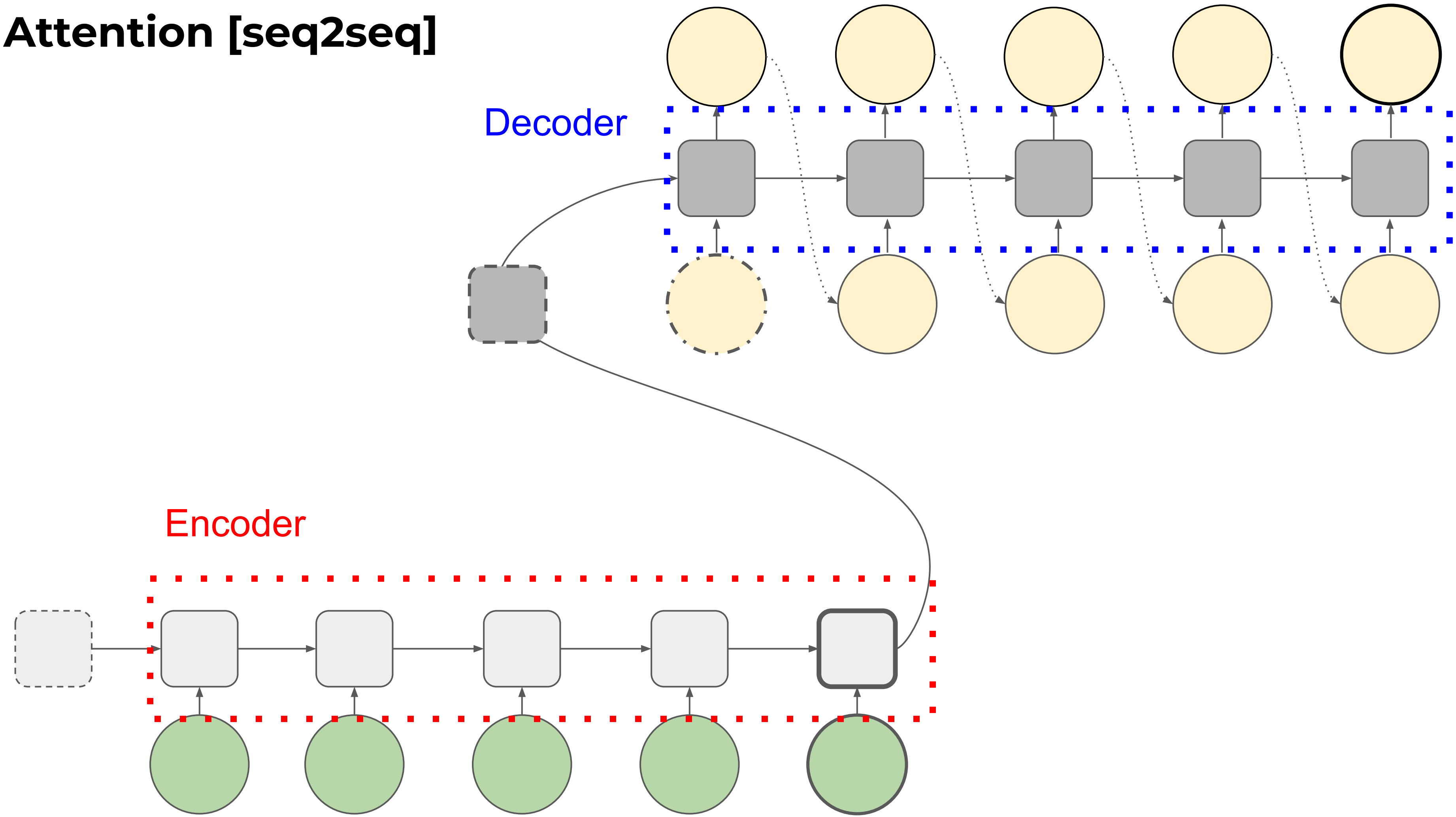
Attention [seq2seq]



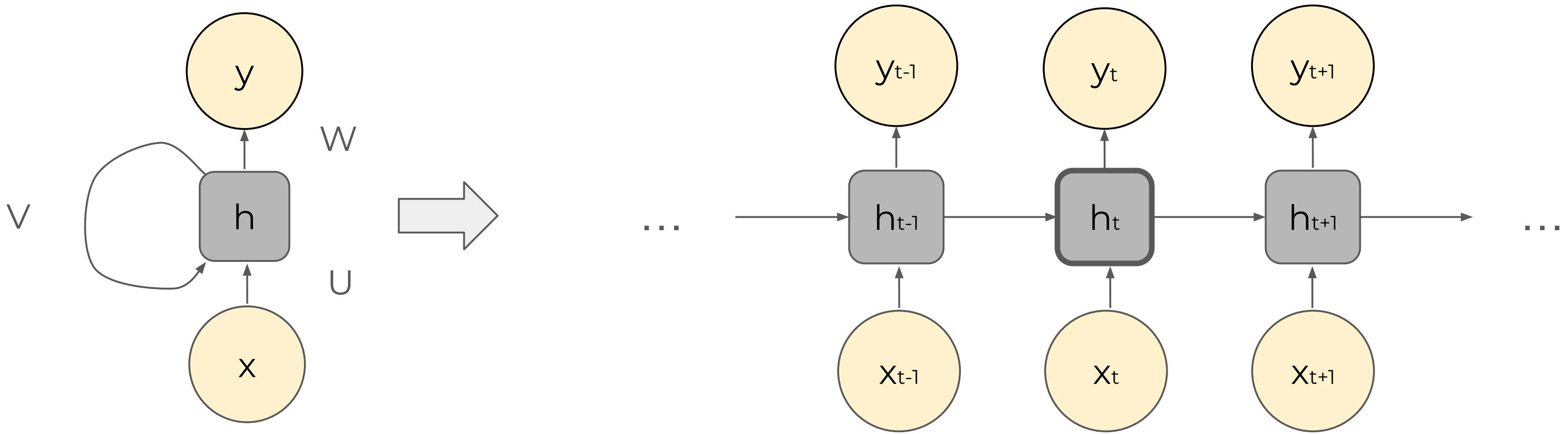
Attention [seq2seq]



Attention [seq2seq]

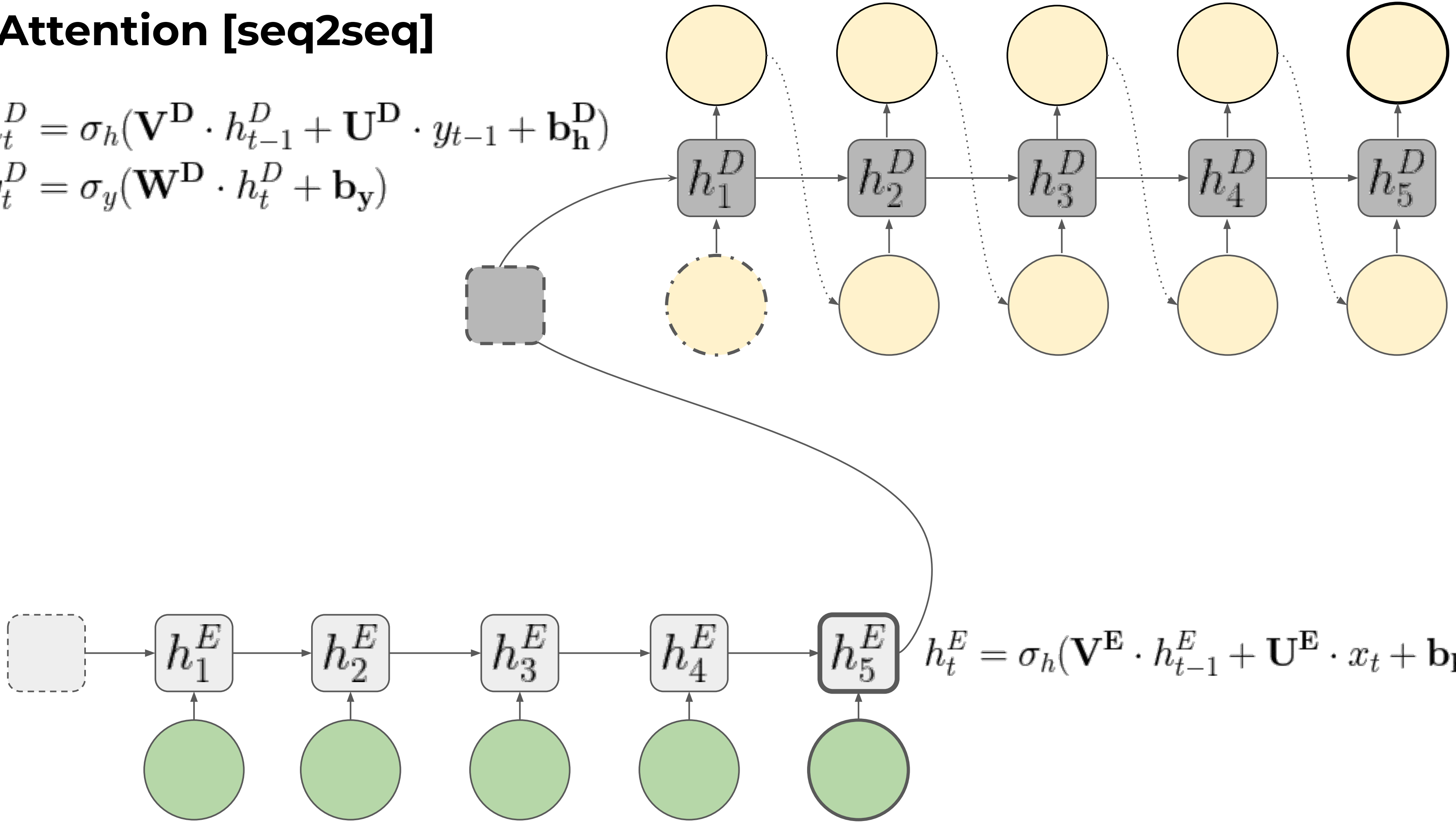


Attention [seq2seq]



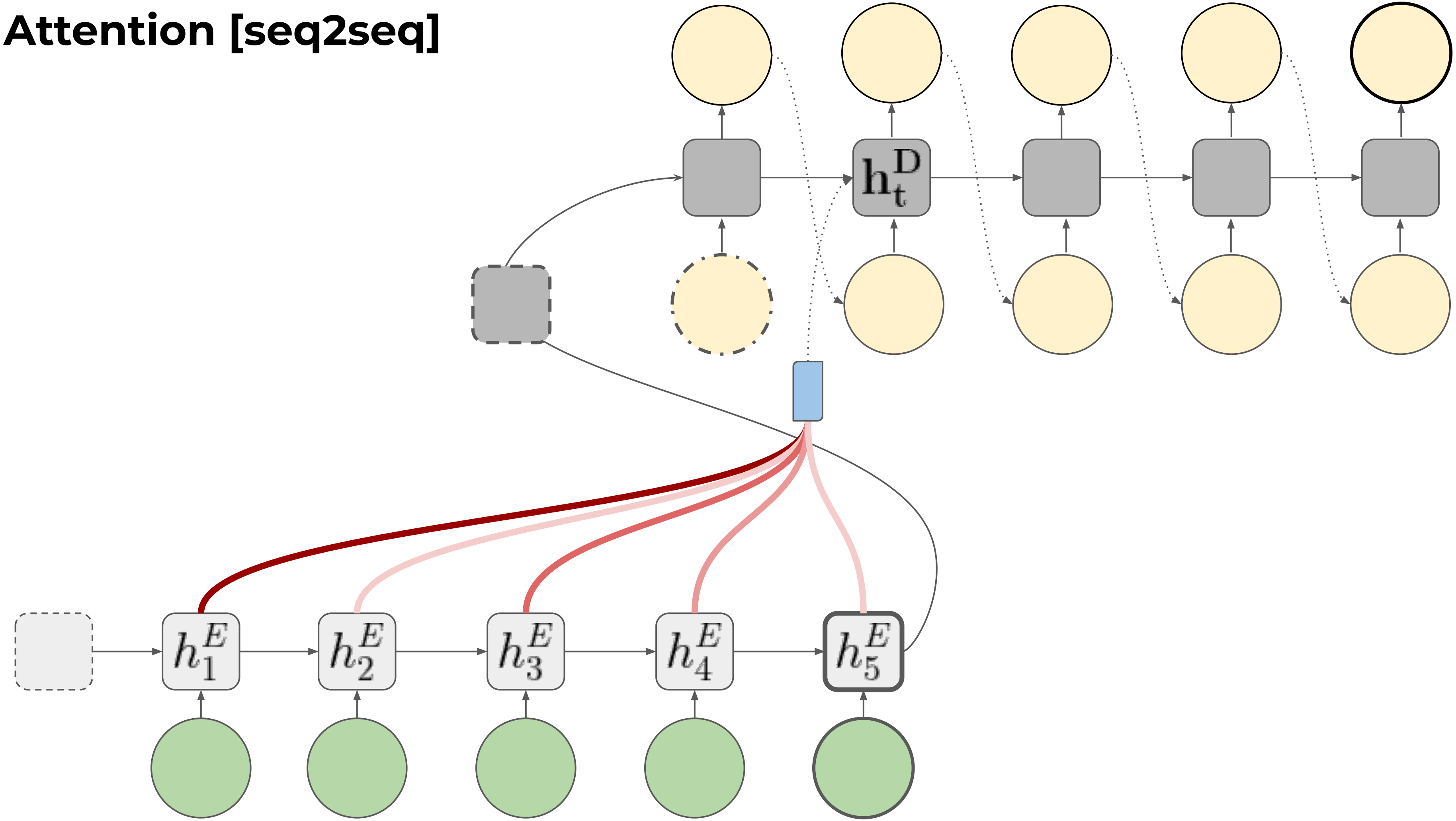
Attention [seq2seq]

$$h_t^D = \sigma_h(\mathbf{V^D} \cdot h_{t-1}^D + \mathbf{U^D} \cdot y_{t-1} + \mathbf{b_h^D})$$
$$y_t^D = \sigma_y(\mathbf{W^D} \cdot h_t^D + \mathbf{b_y})$$



$$h_t^E = \sigma_h(\mathbf{V^E} \cdot h_{t-1}^E + \mathbf{U^E} \cdot x_t + \mathbf{b_h})$$

Attention [seq2seq]



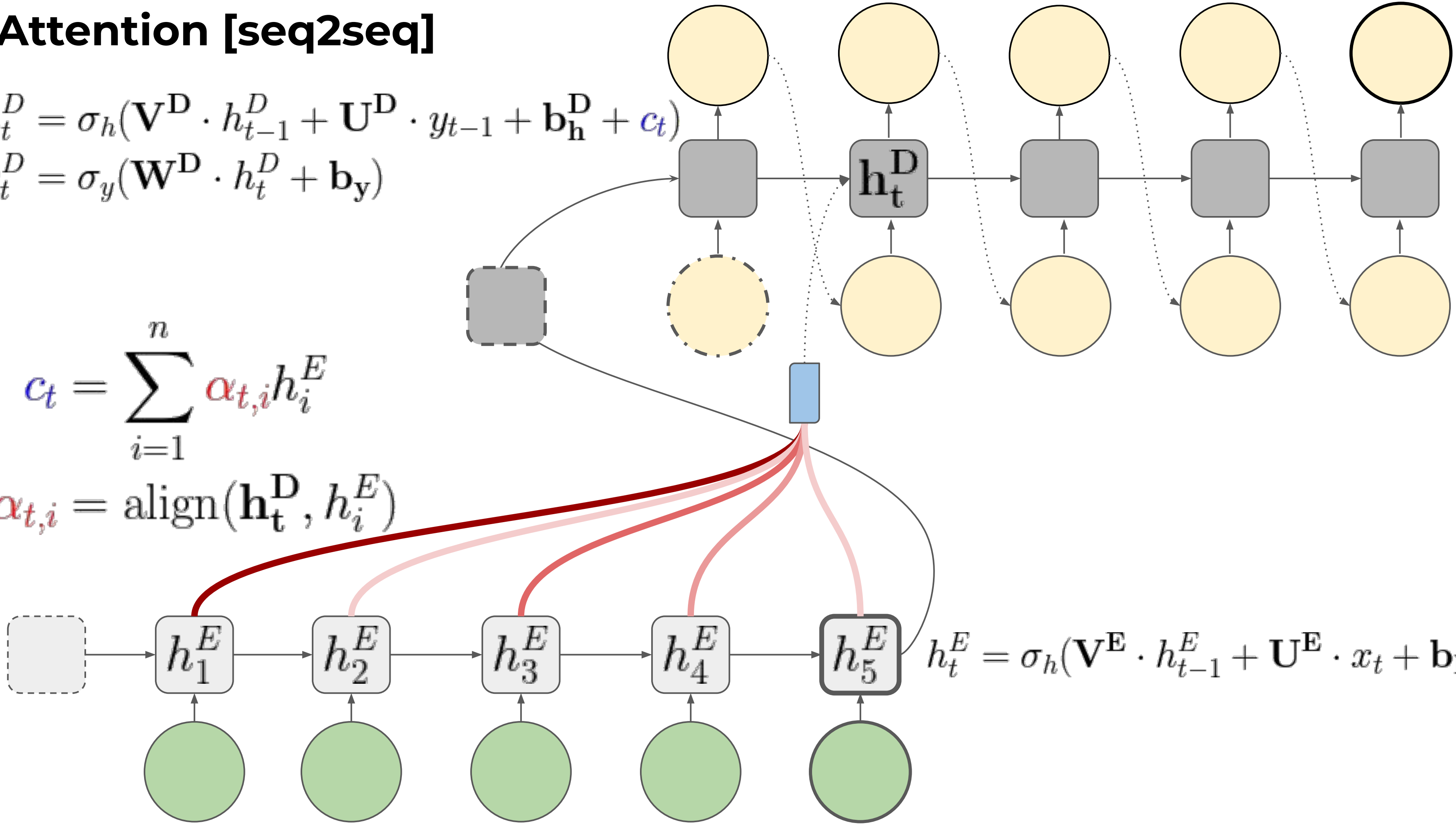
Attention [seq2seq]

$$h_t^D = \sigma_h(\mathbf{V^D} \cdot h_{t-1}^D + \mathbf{U^D} \cdot y_{t-1} + \mathbf{b_h^D} + c_t)$$

$$y_t^D = \sigma_y(\mathbf{W^D} \cdot h_t^D + \mathbf{b_y})$$

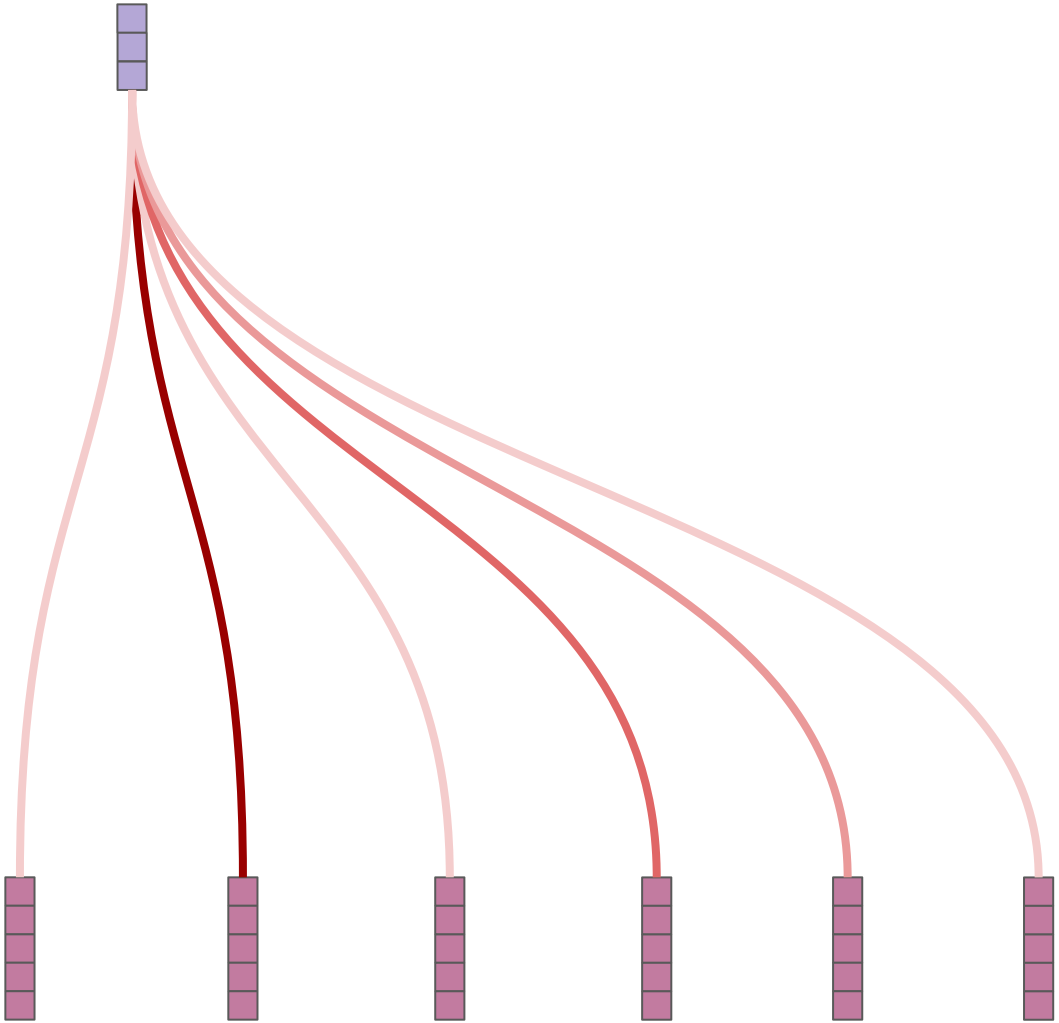
$$c_t = \sum_{i=1}^{n} \alpha_{t,i} h_i^E$$

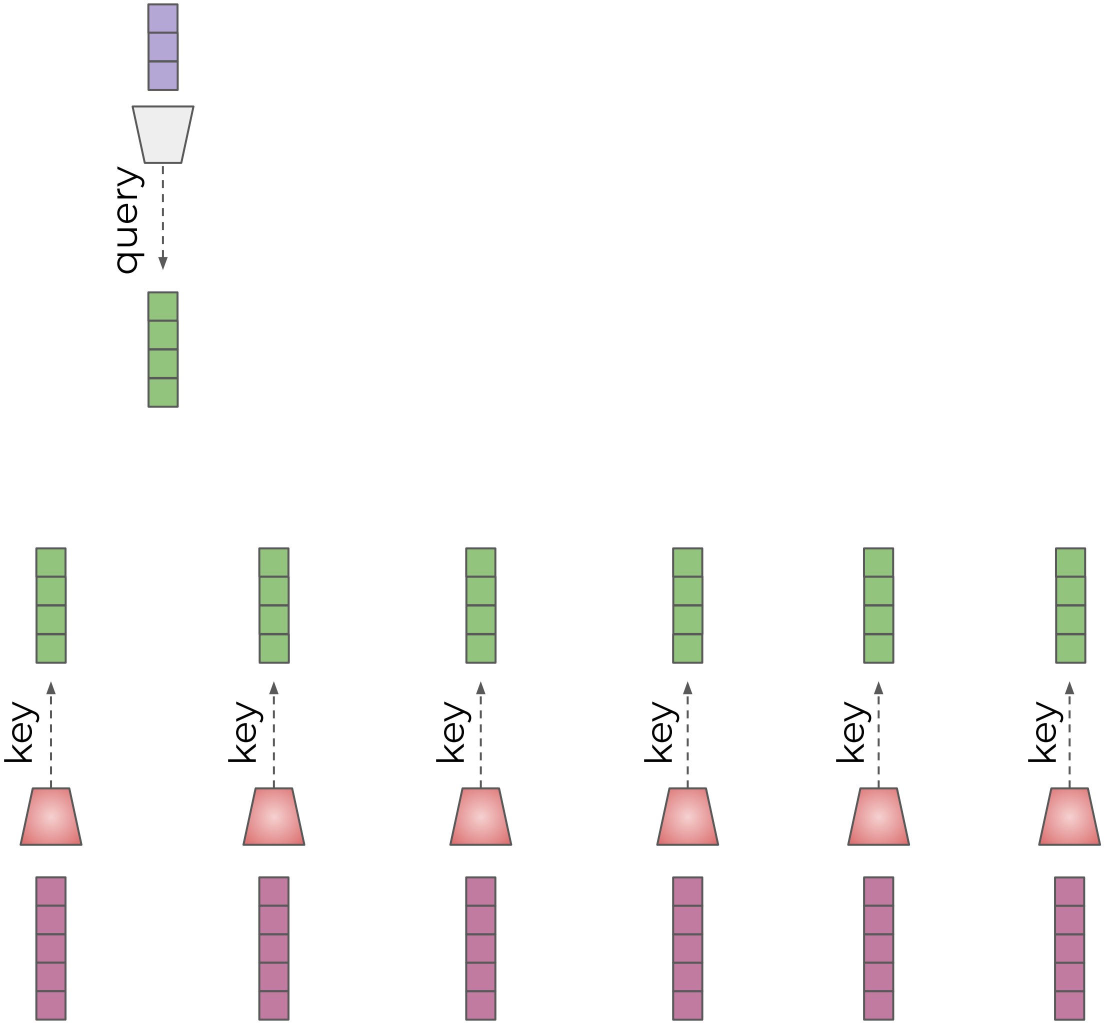
$$\alpha_{t,i} = \text{align}(\mathbf{h_t^D}, h_i^E)$$



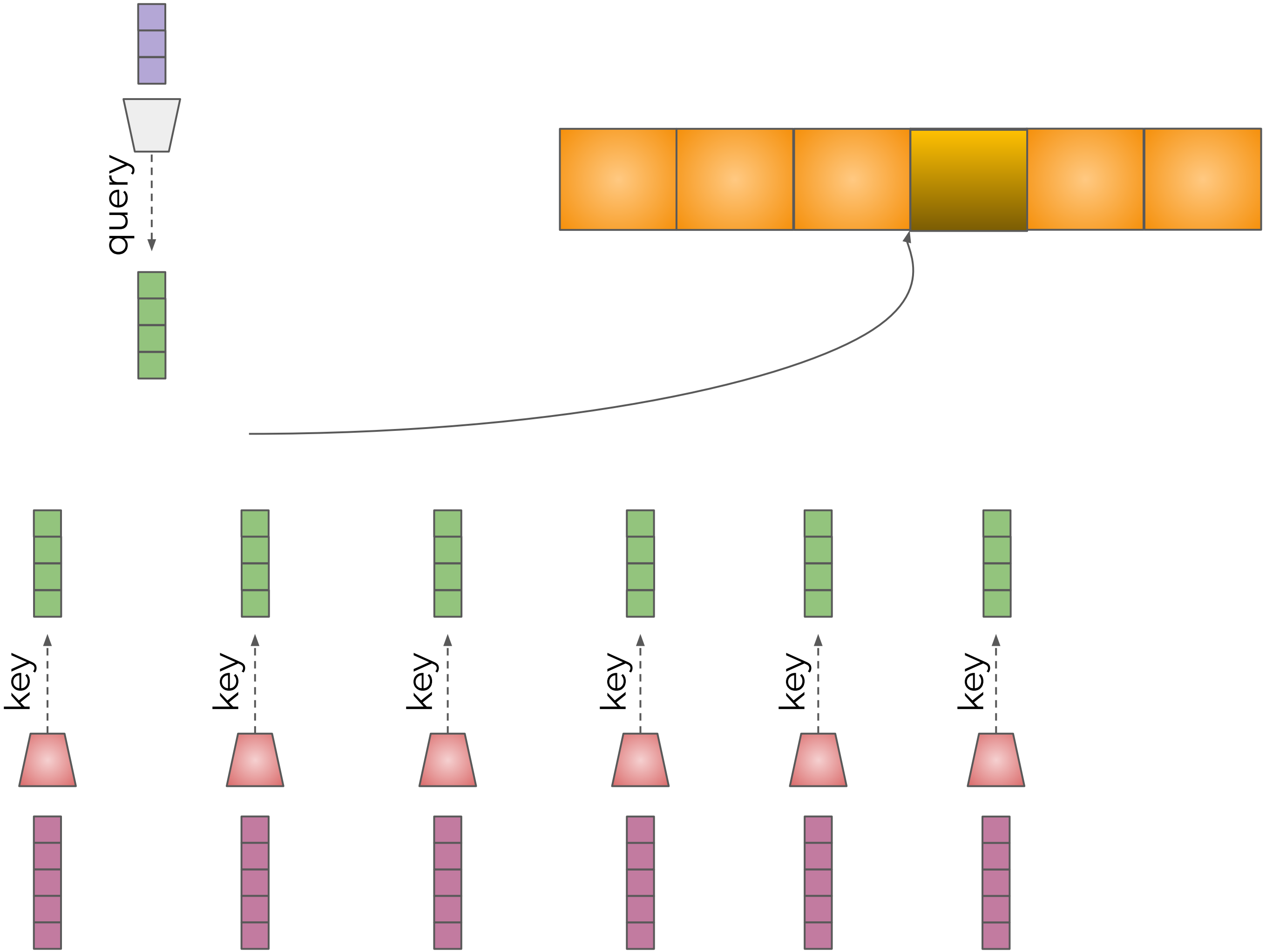
$$h_t^E = \sigma_h(\mathbf{V^E} \cdot h_{t-1}^E + \mathbf{U^E} \cdot x_t + \mathbf{b_h})$$

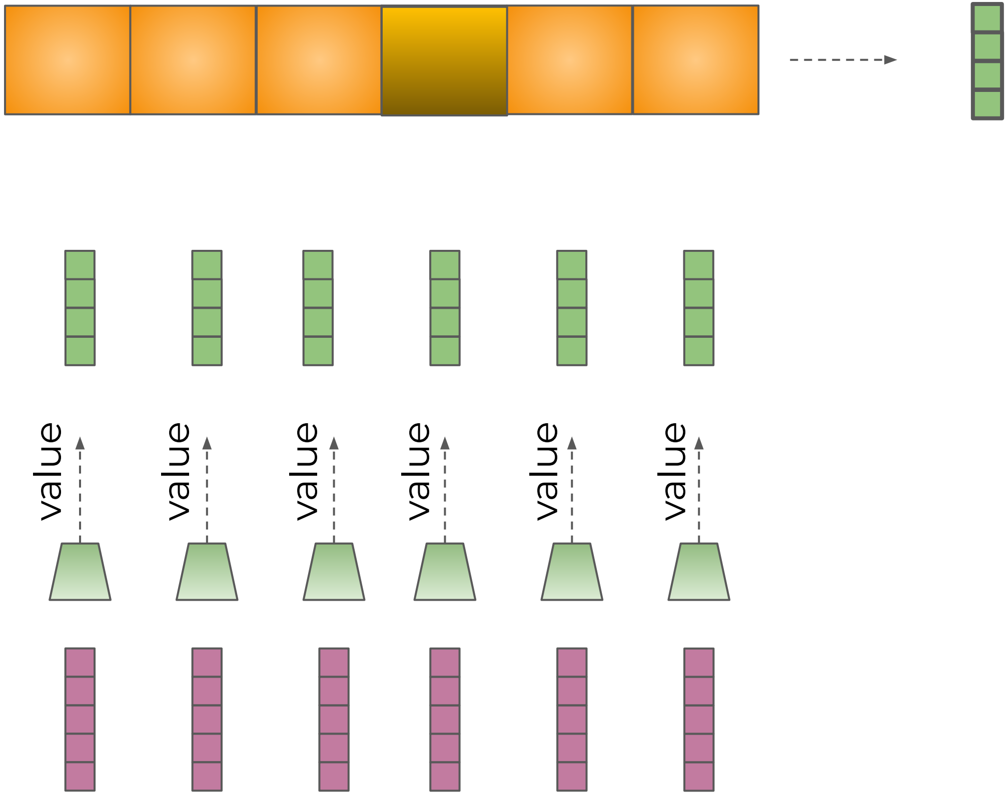
query-key-value abstraction



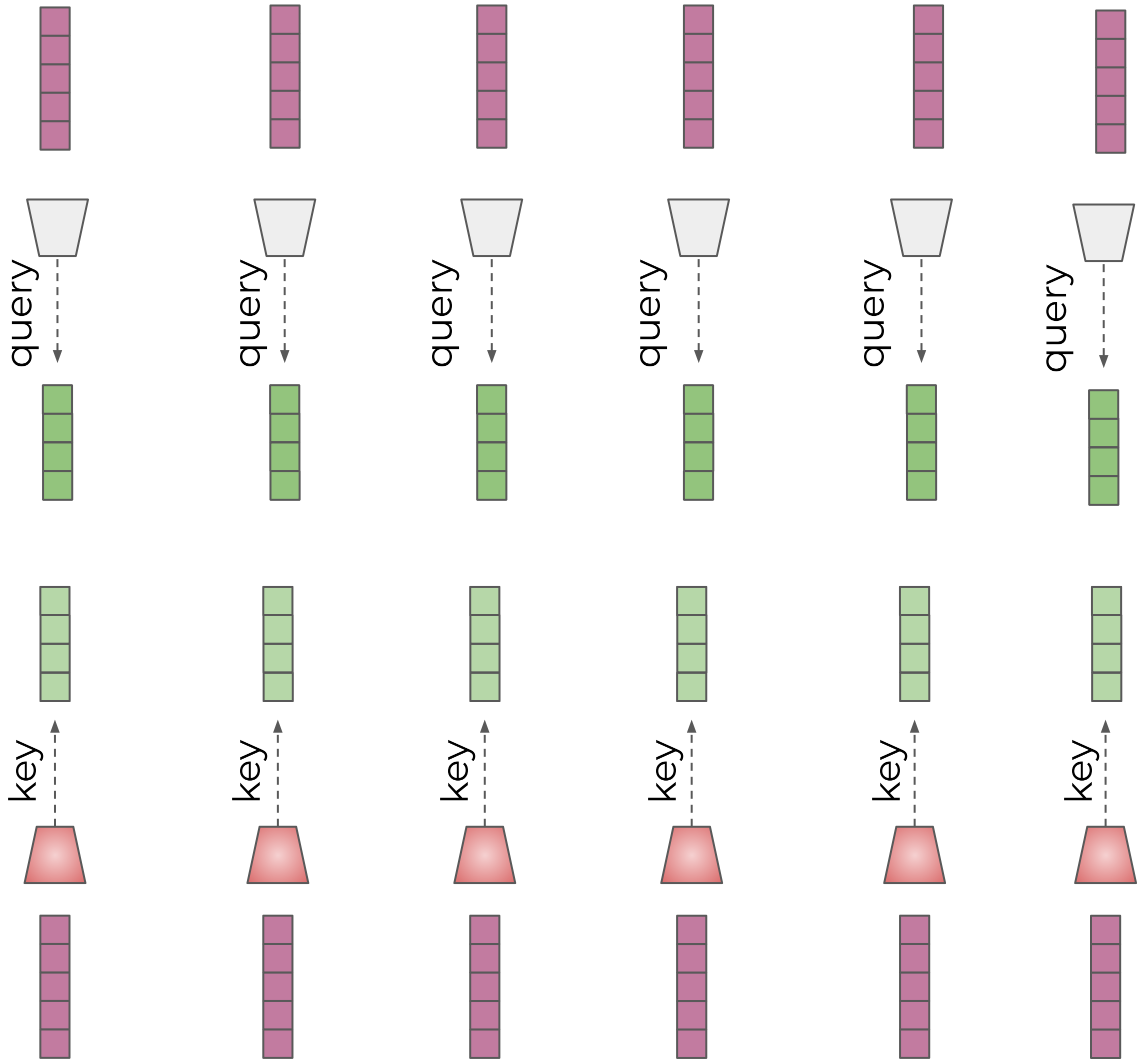


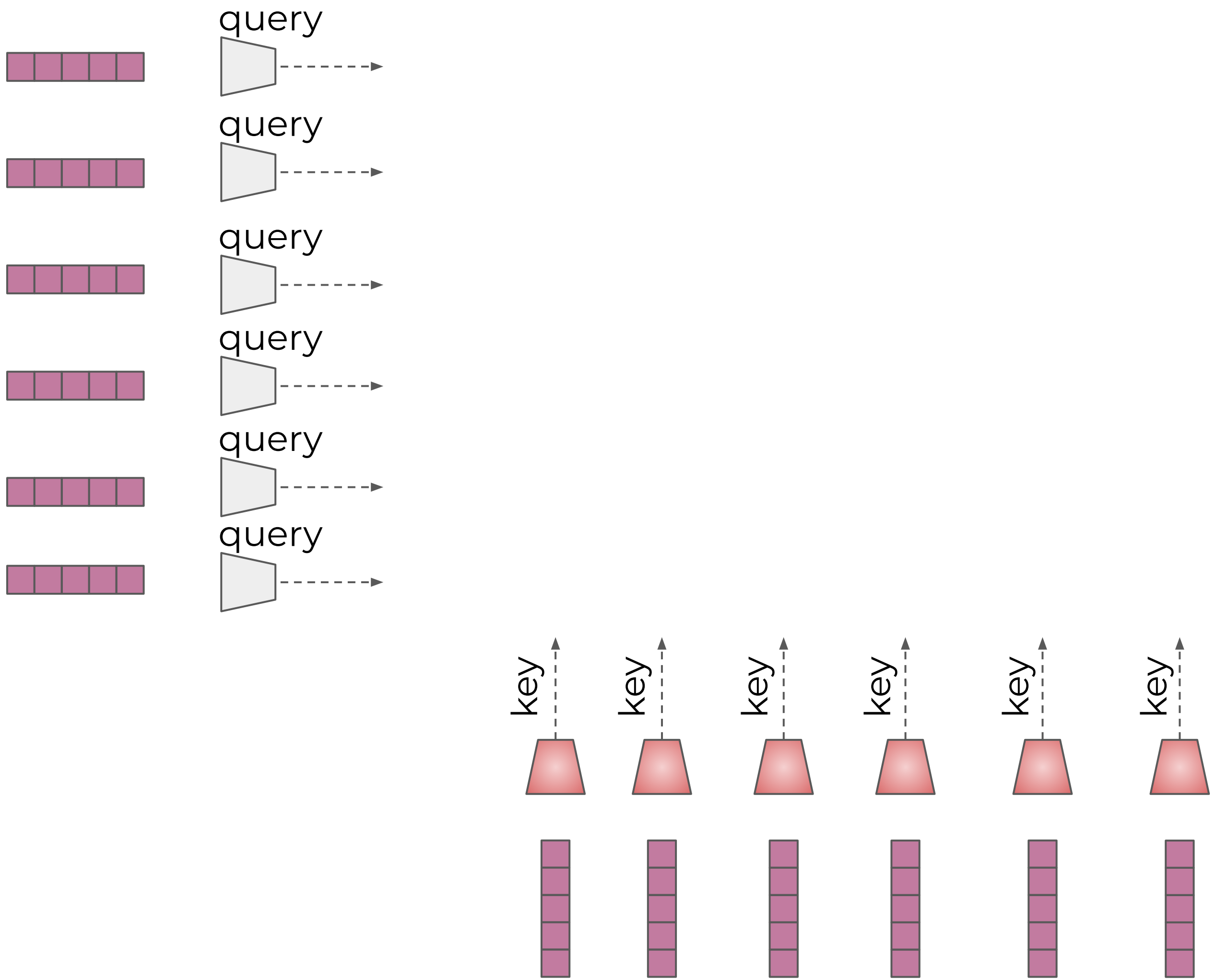
query
key
key
key
key
key
key

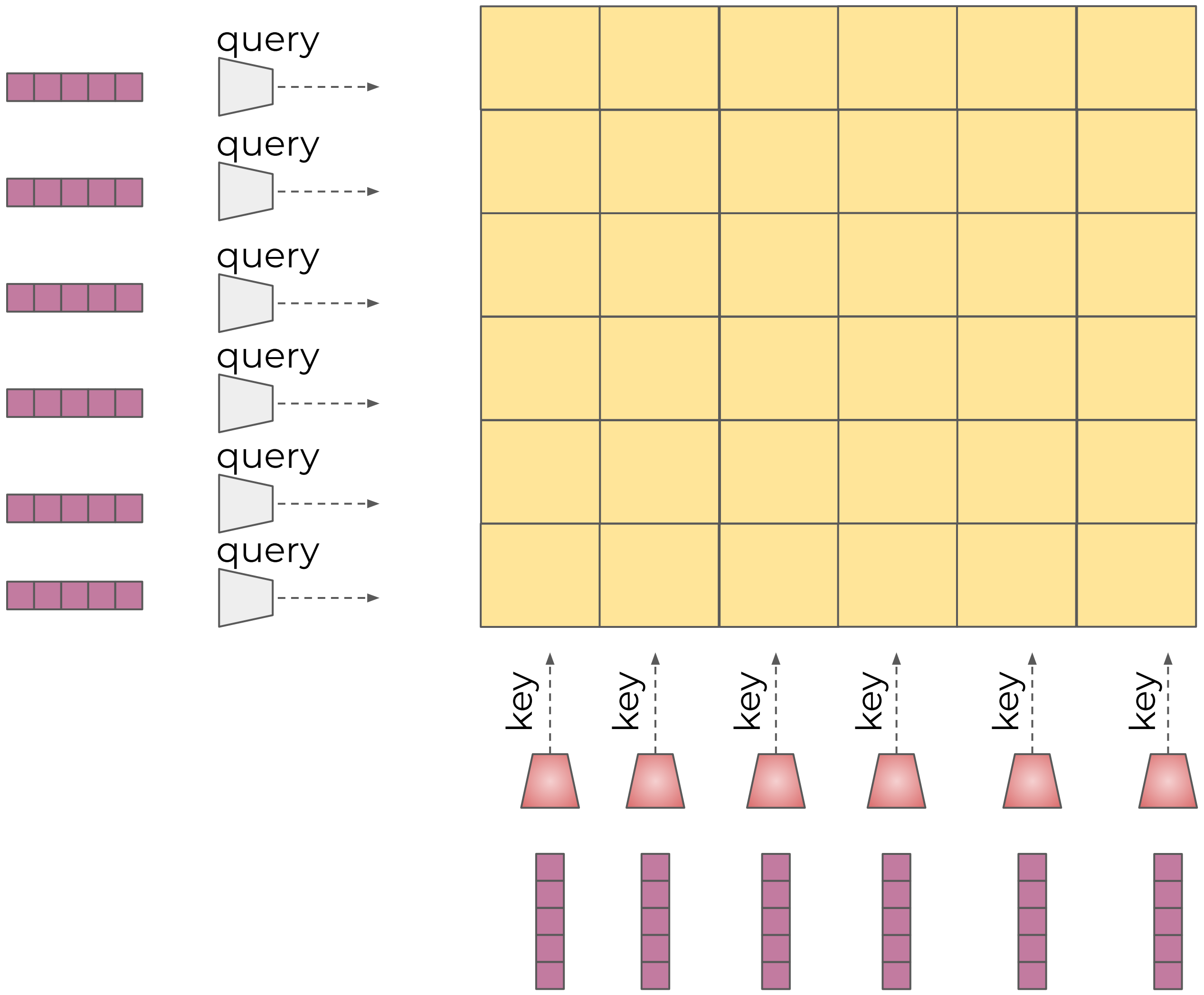

query
key
key
key
key
key
key

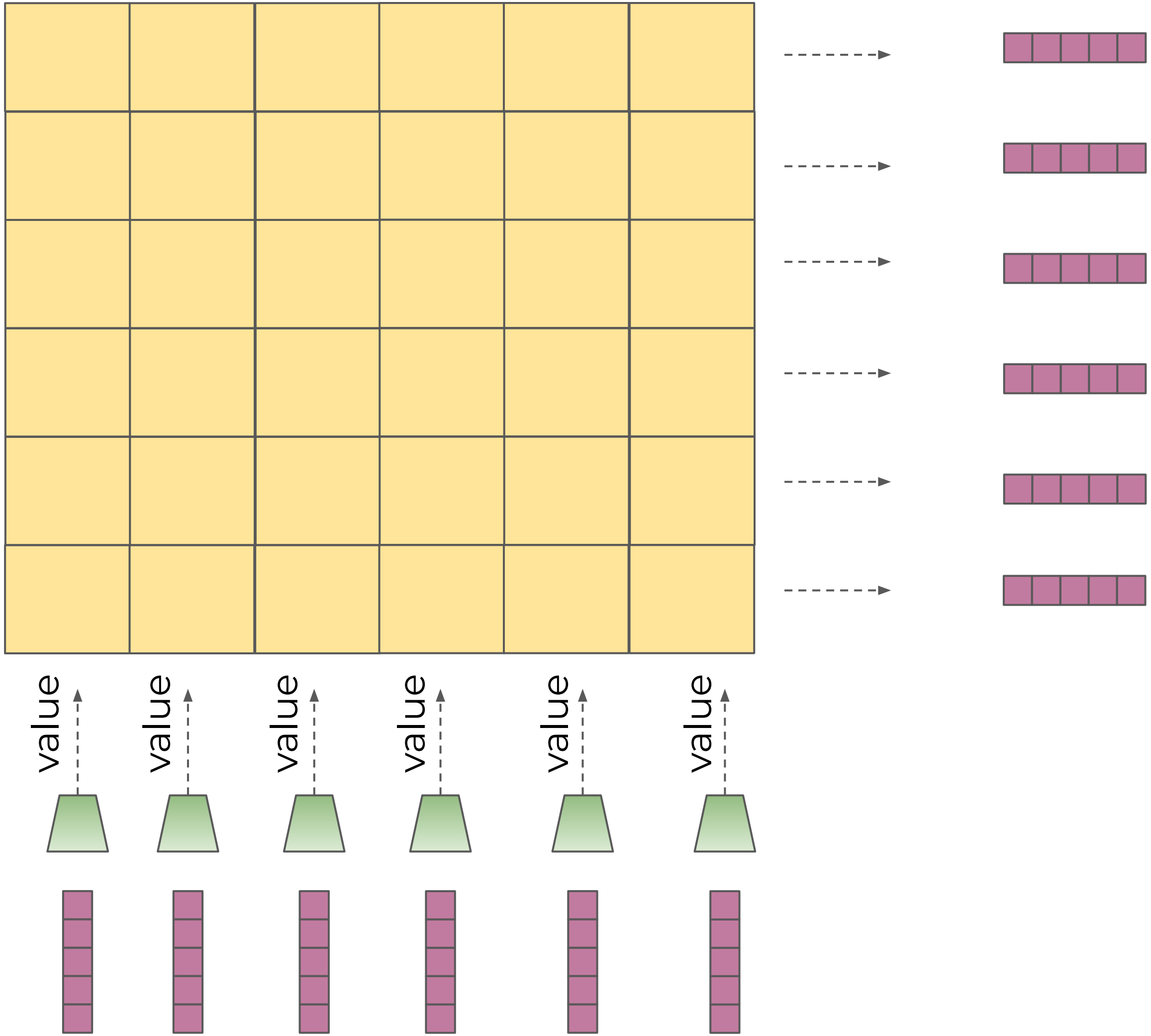

value
value
value
value
value
value

Self-attention

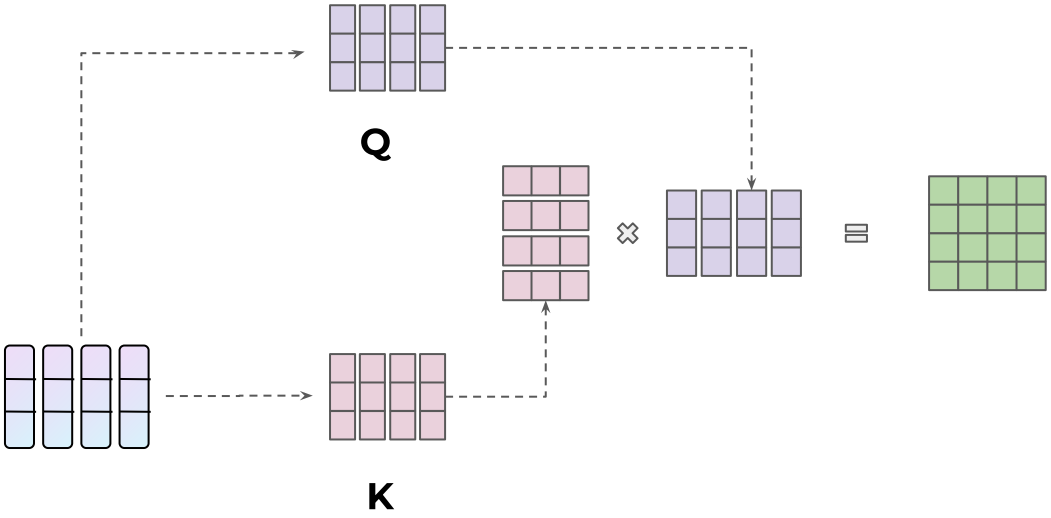

query
query
query
query
query
query
key
key
key
key
key
key


query
query
query
query
query
query
key
key
key
key
key
key

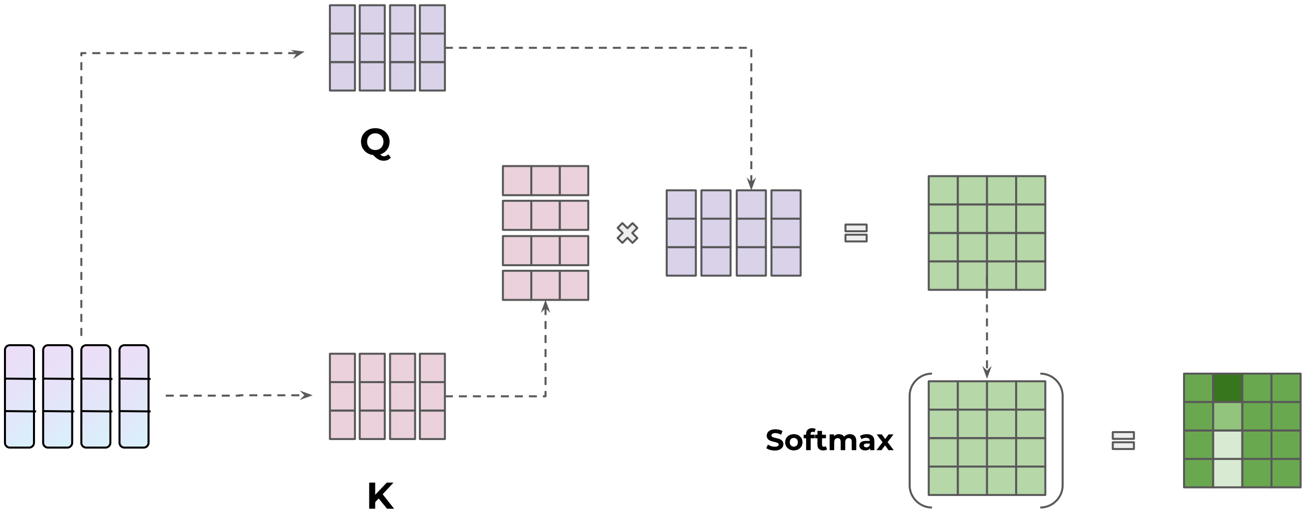

query
query
query
query
query
query
key
key
key
key
key
key


value
value
value
value
value
value

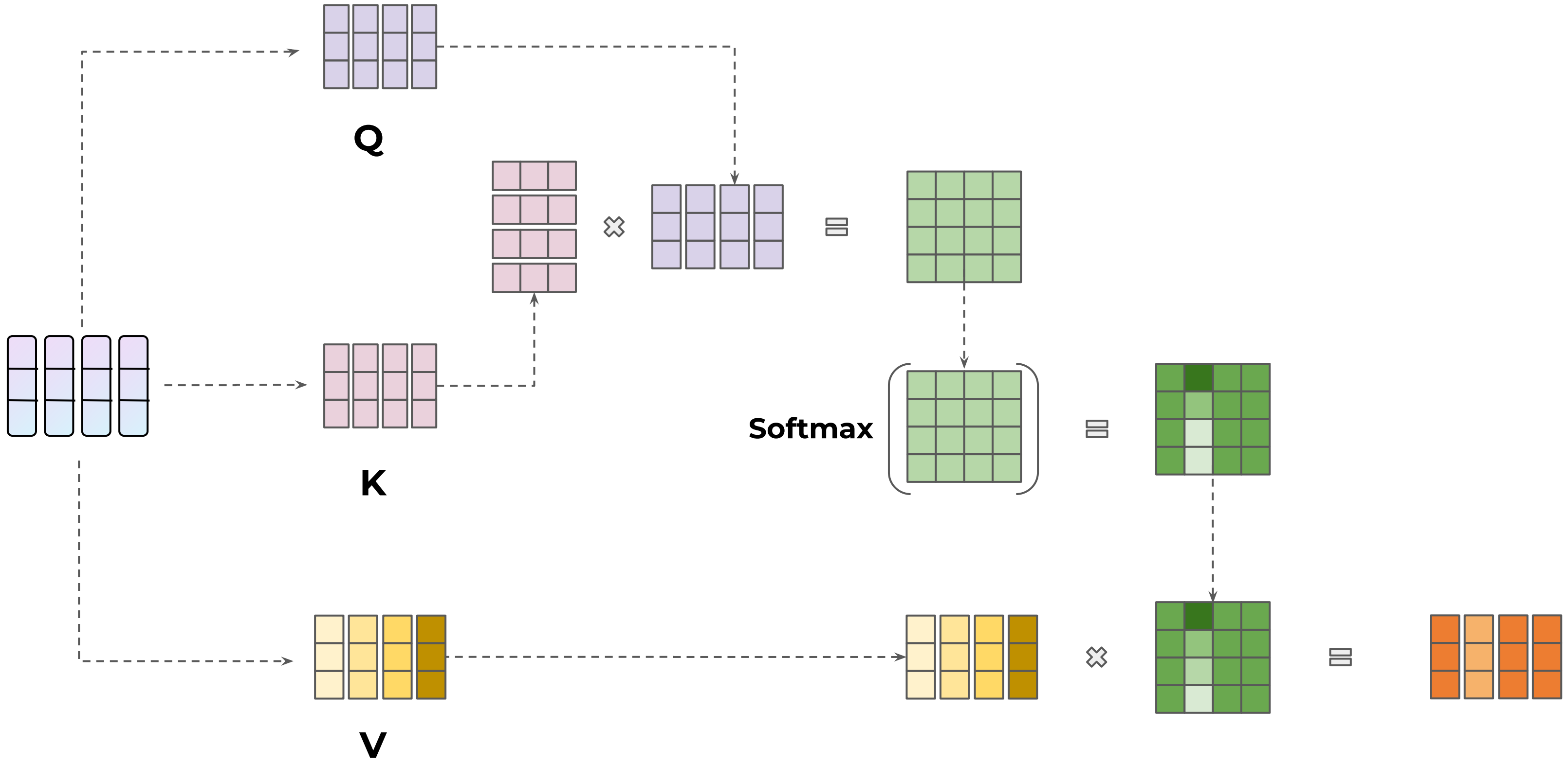
Self-Attention



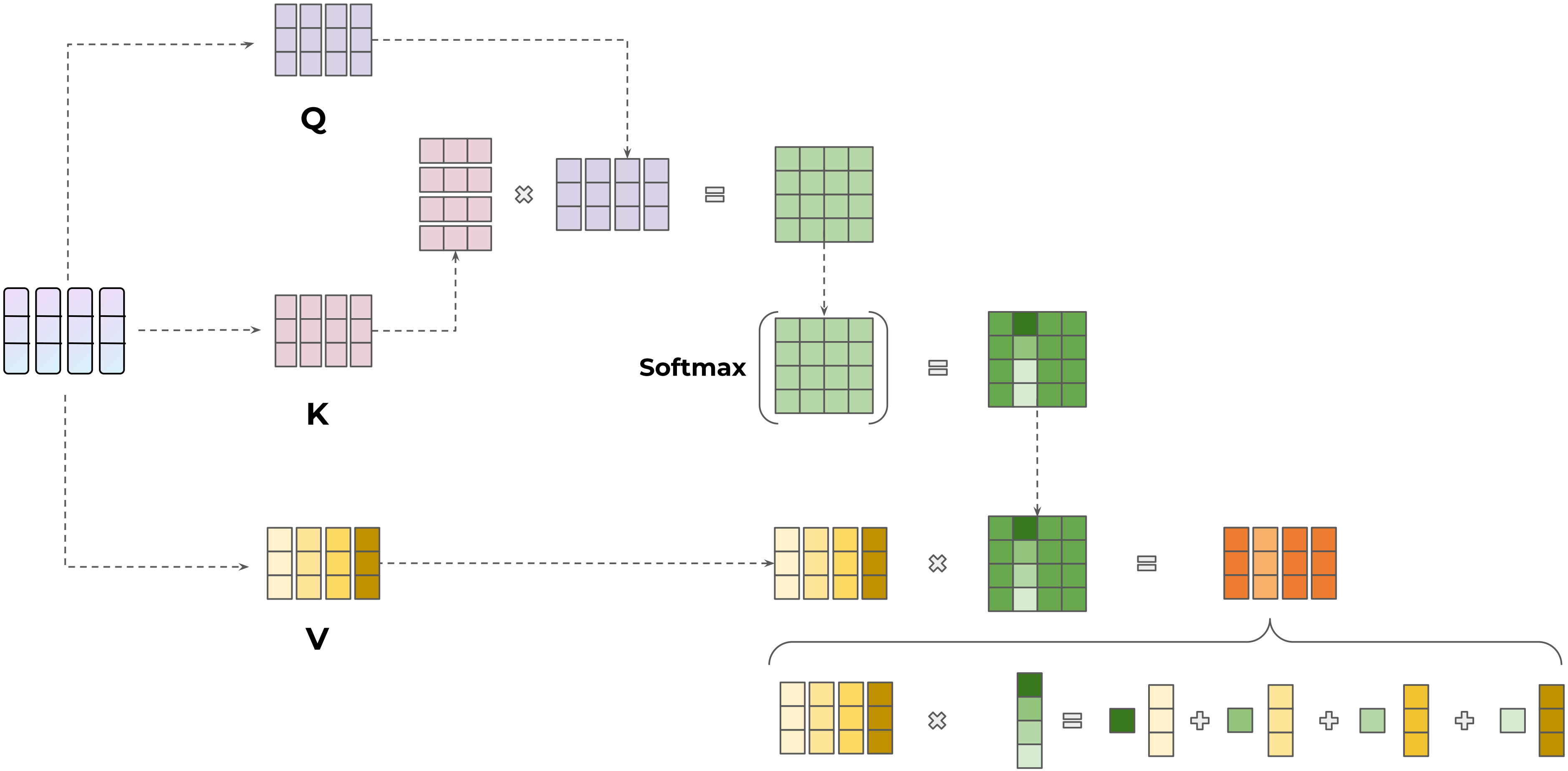
Self-Attention



Self-Attention



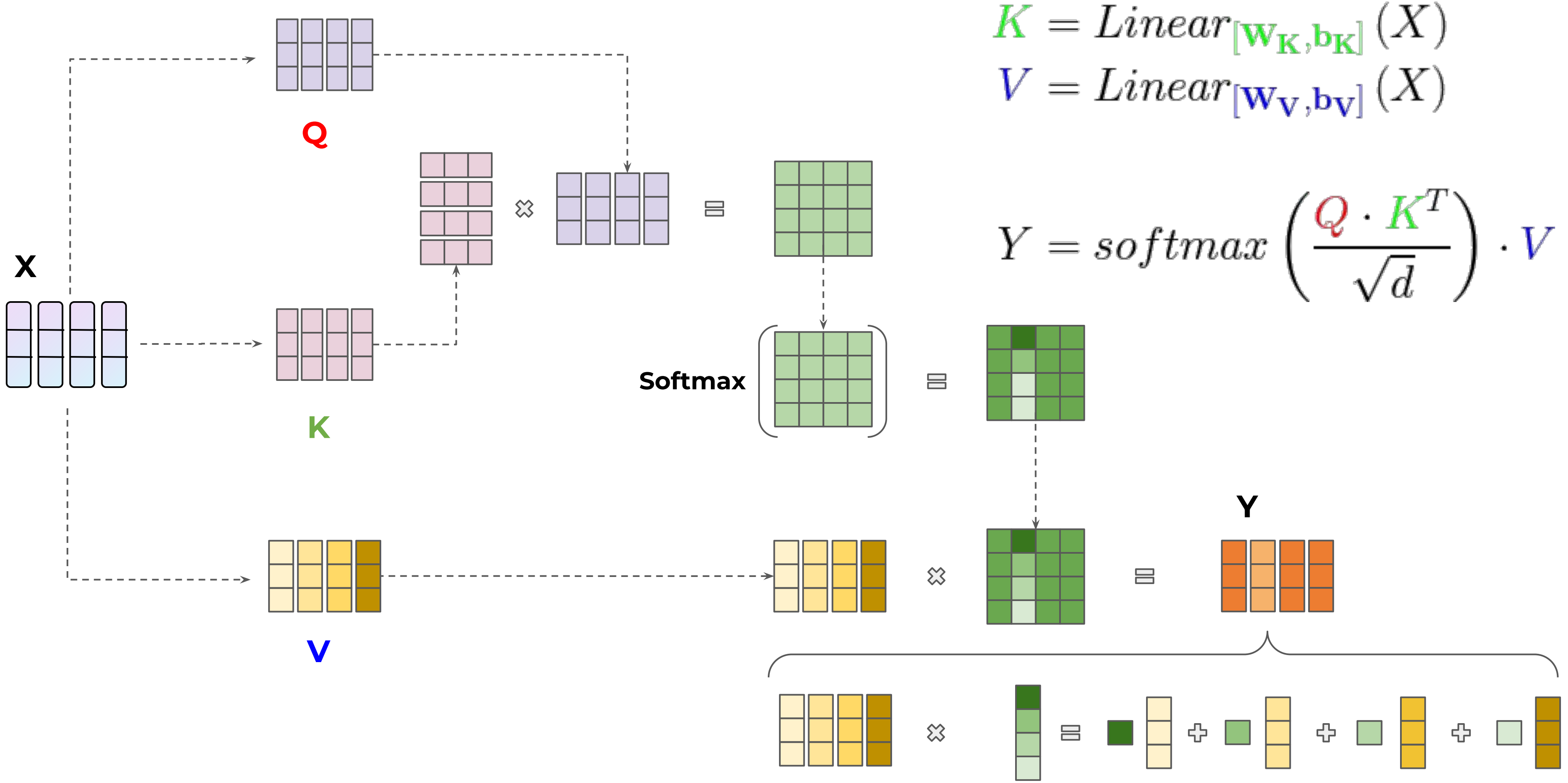
Self-Attention

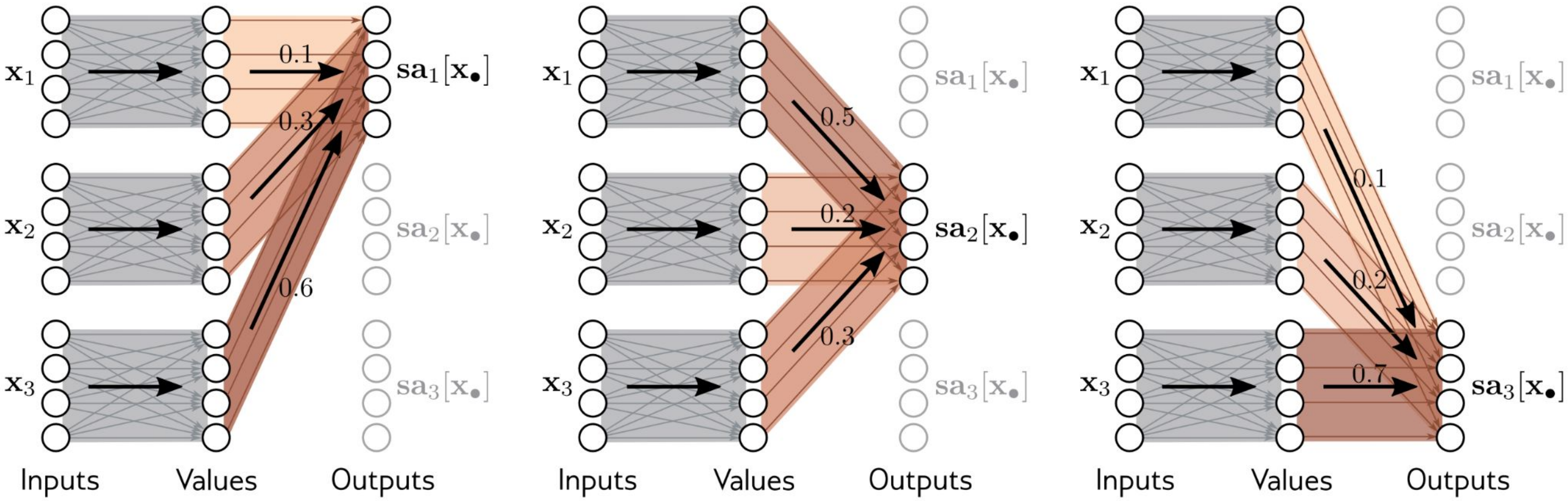


Self-Attention

$$Q = Linear_{[\mathbf{W_Q},\mathbf{b_Q}]}(X)$$
$$K = Linear_{[\mathbf{W_K},\mathbf{b_K}]}(X)$$
$$V = Linear_{[\mathbf{W_V},\mathbf{b_V}]}(X)$$

$$Y = softmax\left(\frac{Q \cdot K^T}{\sqrt{d}}\right) \cdot V$$

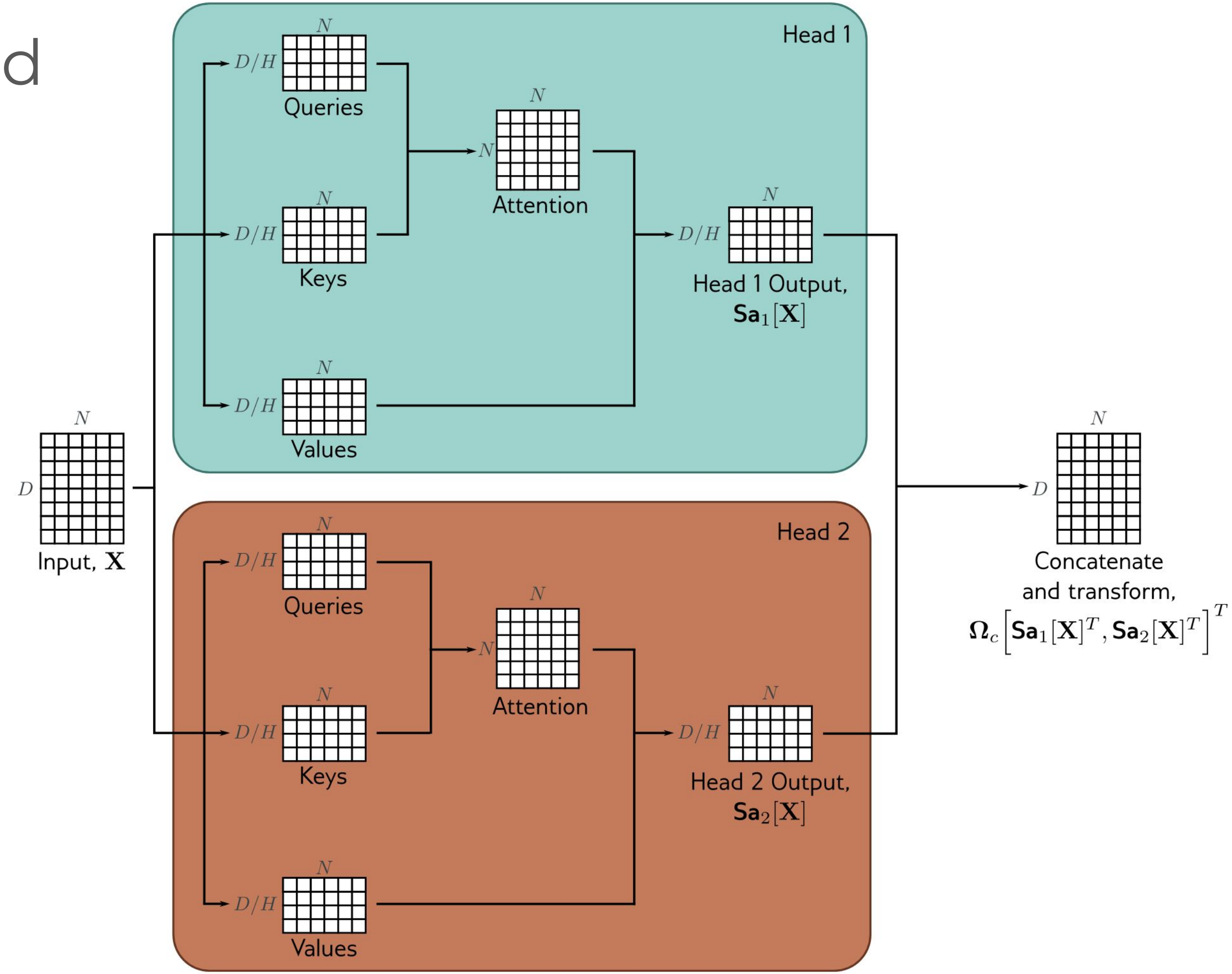




* image from Understanding Deep Learning, book by Simon J.D. Prince, 2023

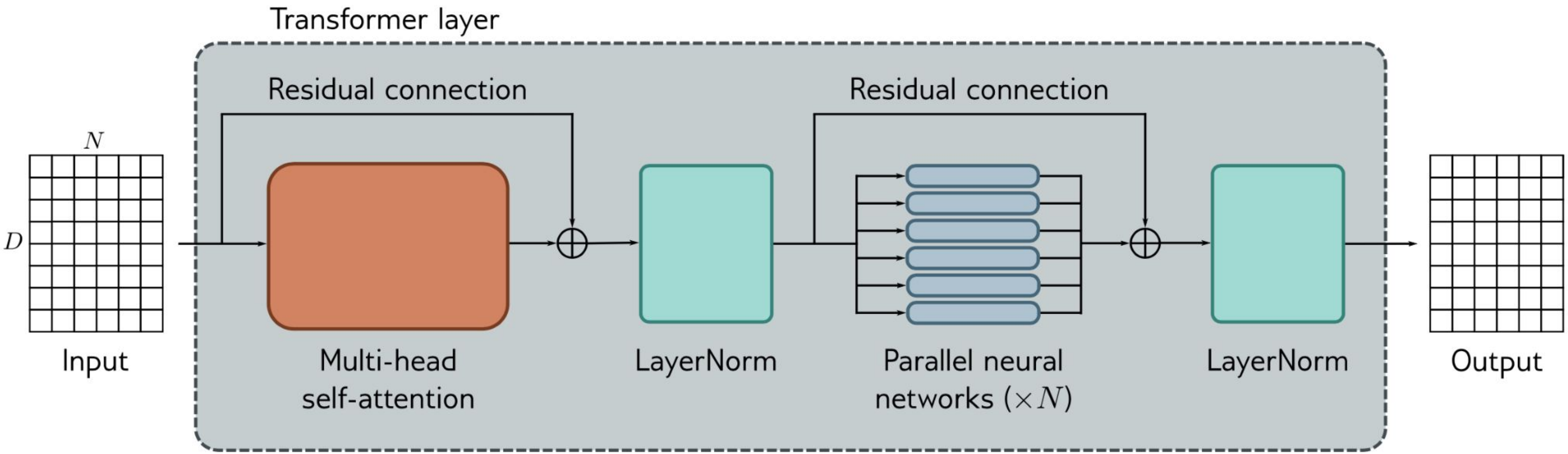
Transformer layer

Multihead



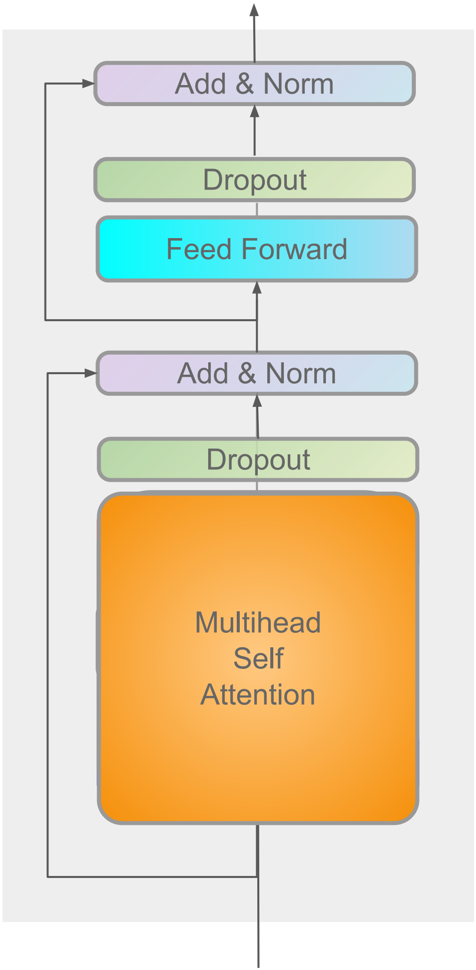
Head 1
N
D/H
Queries
Keys
Values
Attention
Head 1 Output,
$\mathbf{Sa}_1[\mathbf{X}]$
D
Input, $\mathbf{X}$
Head 2
Queries
Keys
Values
Attention
Head 2 Output,
$\mathbf{Sa}_2[\mathbf{X}]$
Concatenate
and transform,
$\boldsymbol{\Omega}_c\left[\mathbf{Sa}_1[\mathbf{X}]^T, \mathbf{Sa}_2[\mathbf{X}]^T\right]^T$

Transformer layer

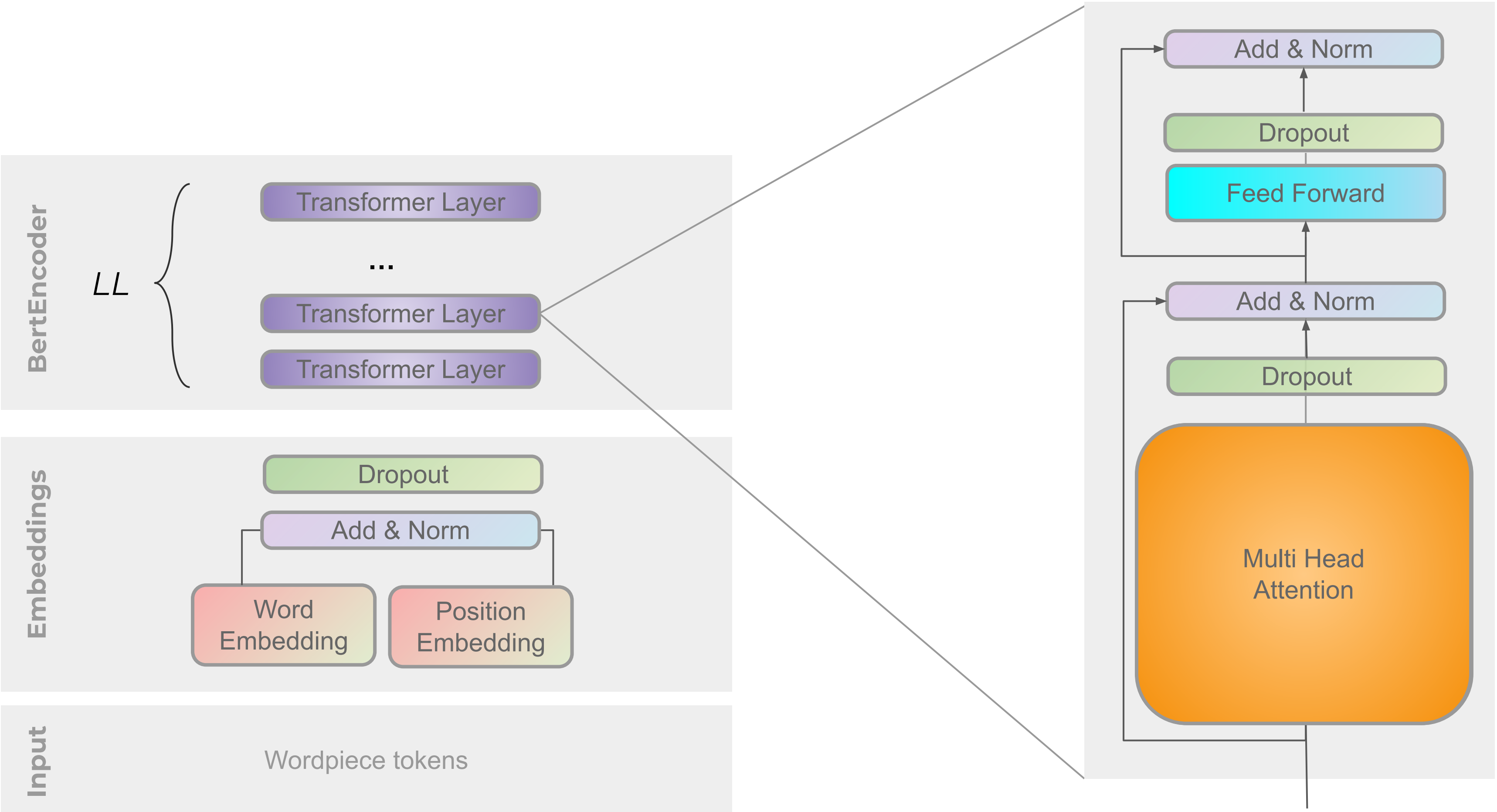


* image from Understanding Deep Learning, book by Simon J.D. Prince, 2023

Transformer layer

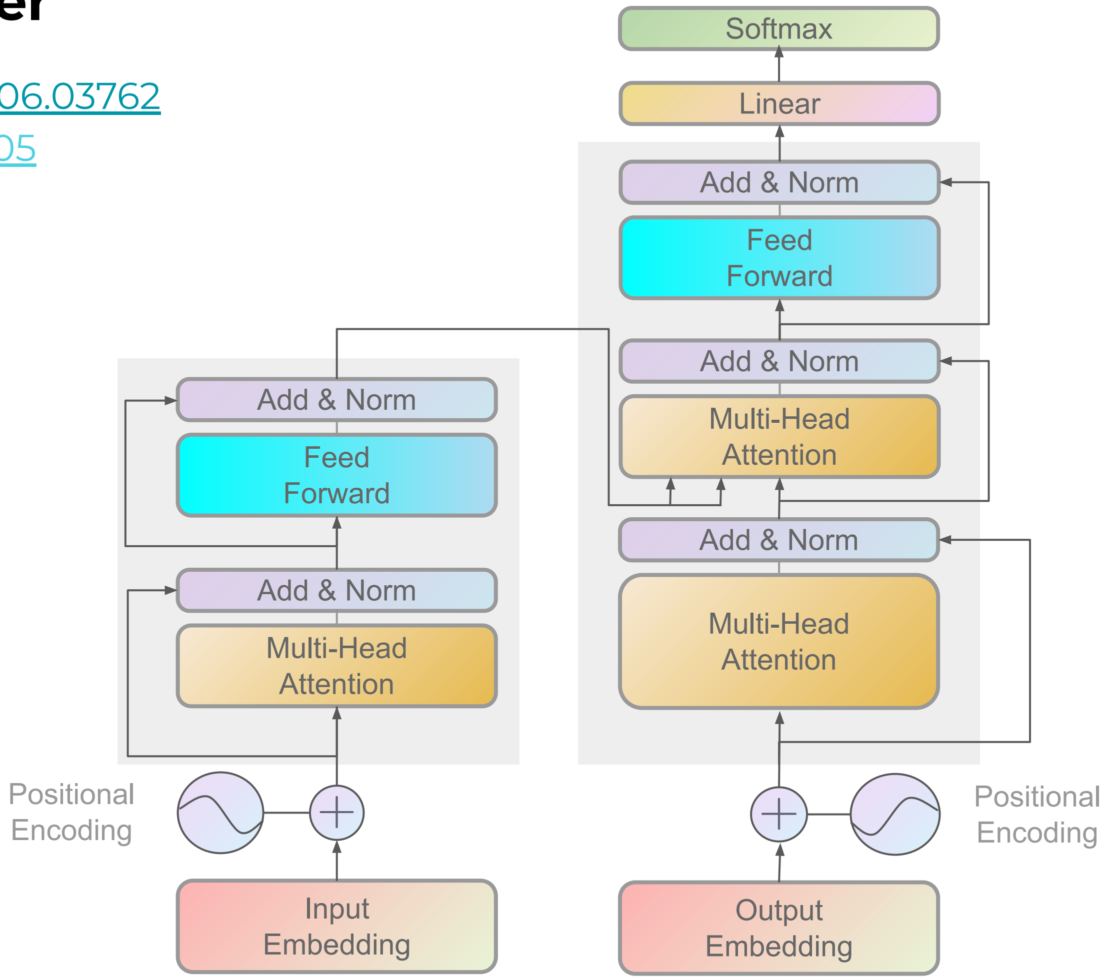


Transformer [Encoder]

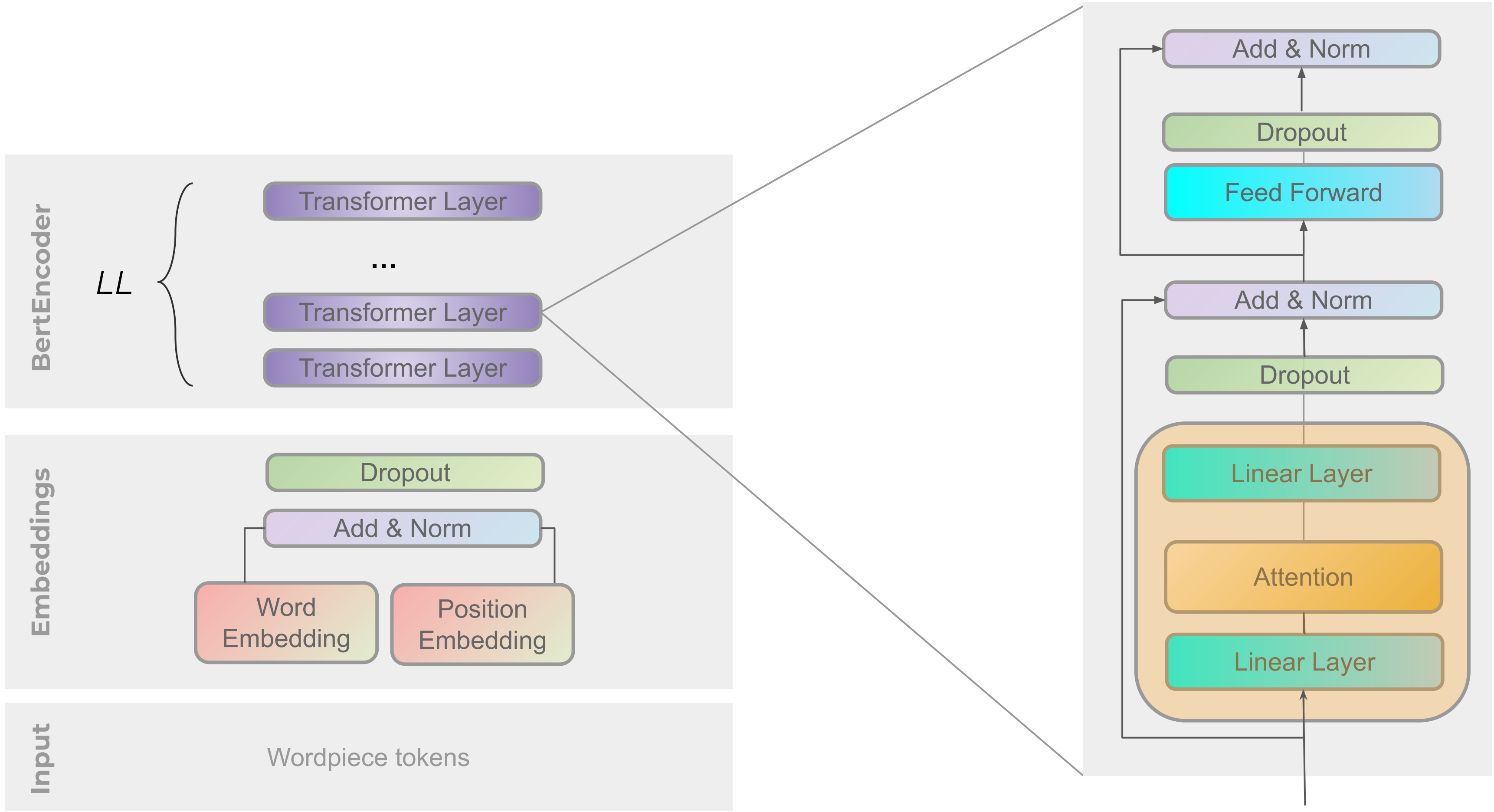


Transformer

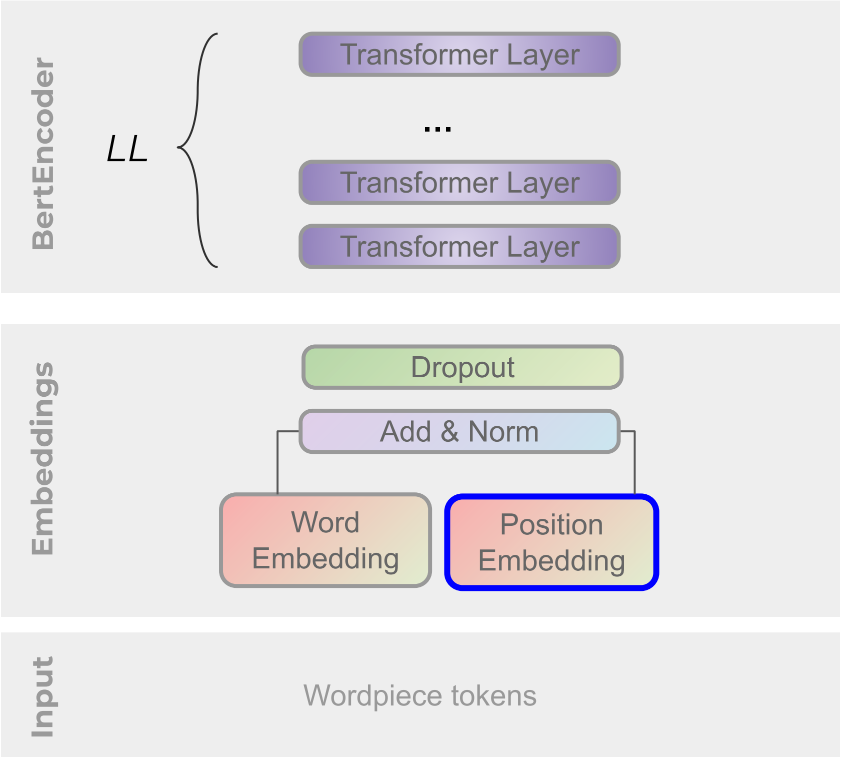
Transformer: [1706.03762]
BERT: [1810.04805]

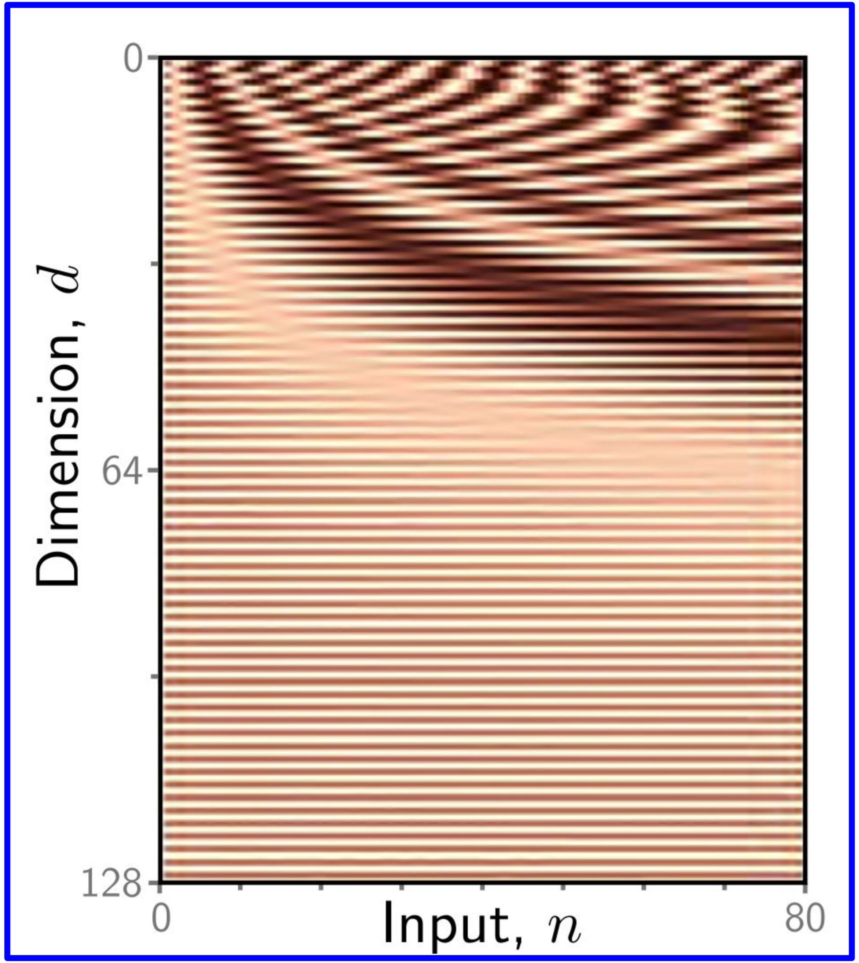


Transformer [Encoder]



Transformer [Encoder]





Memory & Compute

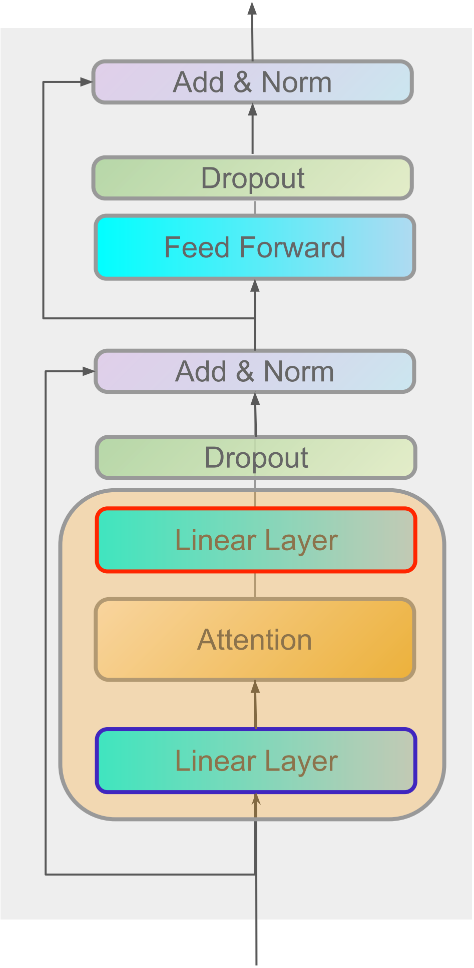
Transformer [Multi head attention]

$$FC(X) = Linear_{[\mathbf{W_{out}}, \mathbf{b_{out}}]}(\sigma(Linear_{[\mathbf{W_{in}}, \mathbf{b_{in}}]}(X)))$$

$$MultiHead(\mathbf{X}, \mathbf{X}) = Linear_{[\mathbf{W^O}, \mathbf{b^O}]}\left(\underset{i \in [A]}{concat}\left[\mathbf{H^i}\right]\right)$$

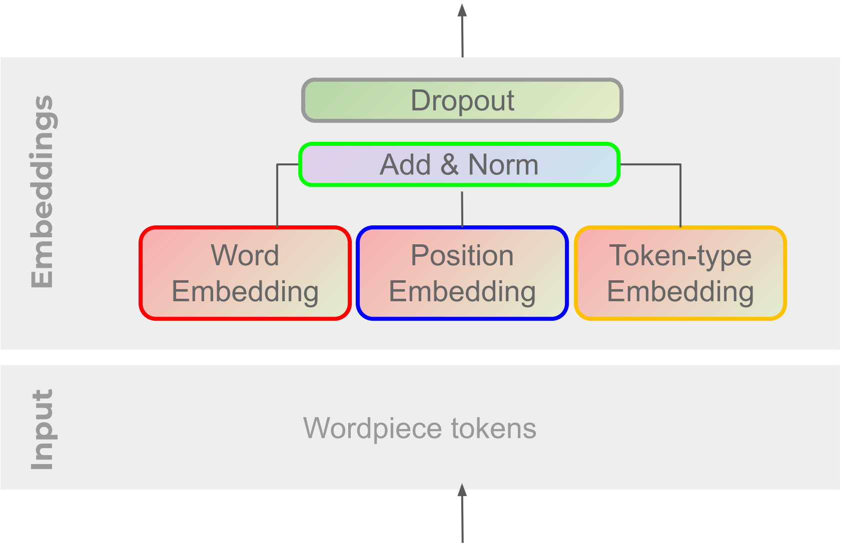
$$Attention(Q, K, V) = softmax\left(\frac{Q \cdot K^T}{\sqrt{d}}\right) \cdot V$$

$$\begin{aligned}\mathbf{H^i} = Attention(&Linear_{[\mathbf{W_Q^{(i)}}, \mathbf{b_Q^{(i)}}]}(X), \\ &Linear_{[\mathbf{W_K^{(i)}}, \mathbf{b_K^{(i)}}]}(X), \\ &Linear_{[\mathbf{W_V^{(i)}}, \mathbf{b_V^{(i)}}]}(X))\end{aligned}$$



Transformer [Embeddings]

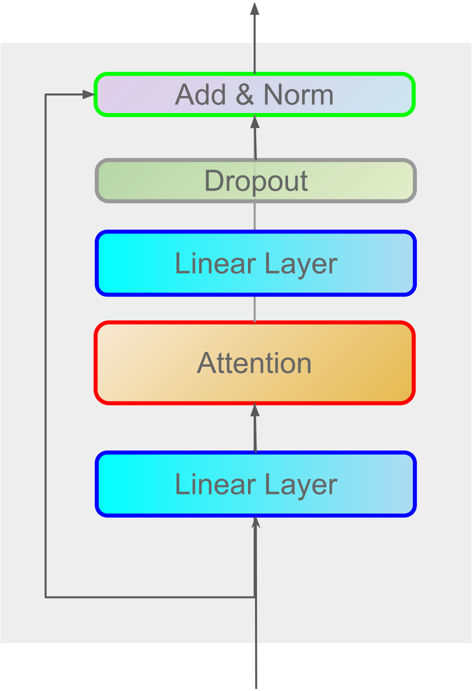
Memory	Computations
*2 * [H]*	*B * L* H*
[V x H]	-
[M x H]	-
[T x H]	-



B - bach_size
H - hidden *[bert-base: 768]*
V - vocabulary size *[bert-base: 50265]*
M - max token length *[bert-base: 514]*
T - number of types *[bert-base: 1]*

Transformer [MHA]

Memory	Computations
*2 * [H]*	*B * L* H*
*[A * S + 1 x H]*	*B * L * (A * S) * H*
-	*B * A * L * L * S * 2*
*3 * [H + 1 x A * S]*	*B * L * H * (A * S) * 3*

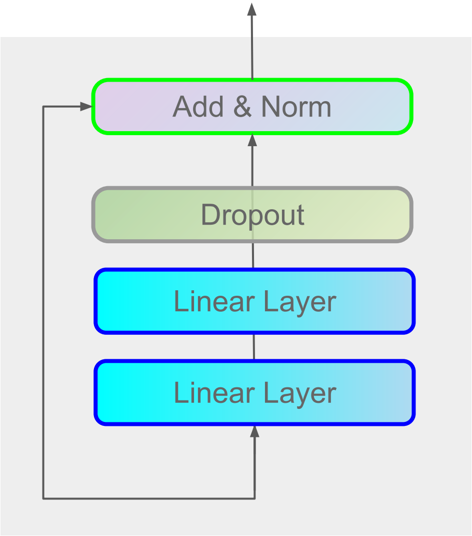


B - bach_size
L - batch_length

H - hidden *[bert-base: 768]*
A - num_heads *[bert-base: 12]*
S - head_size (= H/A) *[bert-base: 64]*
I - intermediate FC size *[bert-base: 3072]*

Transformer [FC]

Memory	Computations
2 * [H]	B * L* H
[H + 1 x I] + [I + 1 x H]	2 * B * L * H * I



B - bach_size
L - batch_length

H - hidden [bert-base: 768]
A - num_heads [bert-base: 12]
S - head_size (= H/A) [bert-base: 64]
I - intermediate FC size [bert-base: 3072]

Transformer [summary]

		#params [M]	mocs [B]
LL	FC-LayerNorm	2 * H	B * L * H
	FC-Out	(1 + H) * I	B * L * H * I
	FC-In	(1 + I) * H	B * L * H * I
	Atten-LayerNorm	2 * H	B * L * H
	Atten-LinearOutput	(1 + H) * H	B * L * H * H
	Atten-AttenScoreValue	0	B * A * L * L * S
	Atten-AttenScore	0	B * A * L * L * S
	Atten-LinearInput	(1 + H) * H * 3	B * L * H * H * 3
	Embedding-LayerNorm	2 * H	B * L * H
	Embedding-Word	V * H	0
	Embedding-Position	M * H	0
	Embedding-Token-Type	T * H	0

Transformer [summary]

		#params [M]	mocs [B]
	FC-LayerNorm	2 * H	B * L * H
	FC-Out	(1 + H) * I	B * L * H * I
	FC-In	(1 + I) * H	B * L * H * I
LL	Atten-LayerNorm	2 * H	B * L * H
	Atten-LinearOutput	(1 + H) * H	B * L * H * H
	Atten-AttenScoreValue	0	B * A * **L * L** * S
	Atten-AttenScore	0	B * A * **L * L** * S
	Atten-LinearInput	(1 + H) * H * 3	B * L * H * H * 3
	Embedding-LayerNorm	2 * H	B * L * H
	Embedding-Word	V * H	0
	Embedding-Position	M * H	0
	Embedding-Token-Type	T * H	0

Transformer [Memory & Runtime]

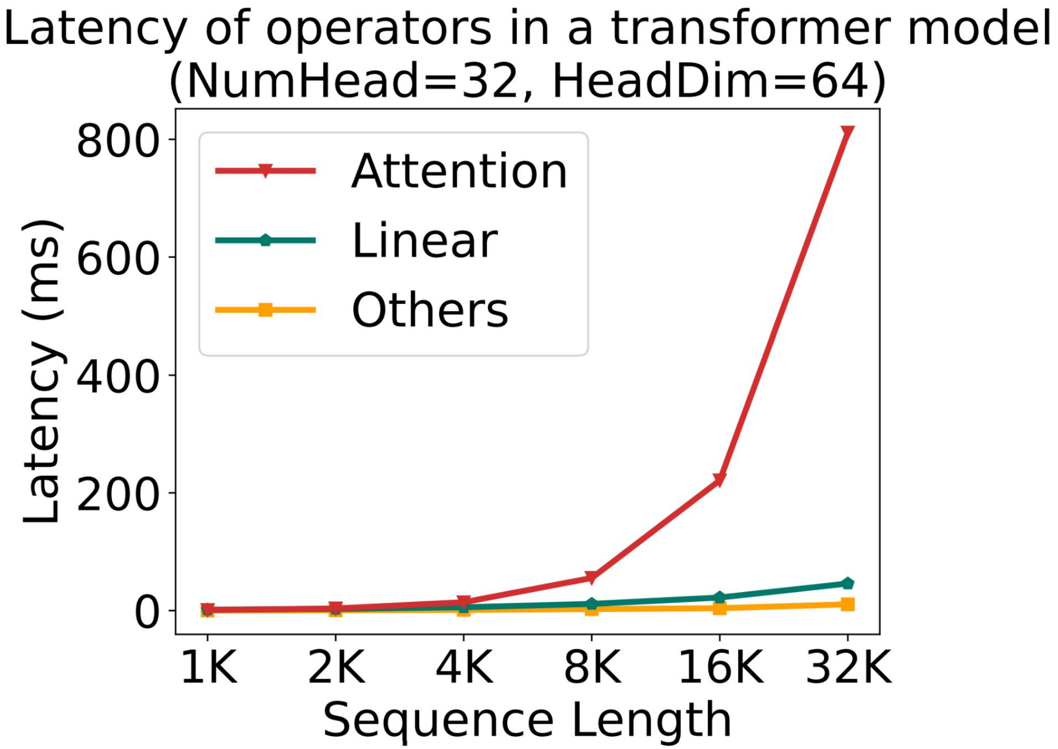


image source: [2410.02367]

Transformer [Flash Attention]

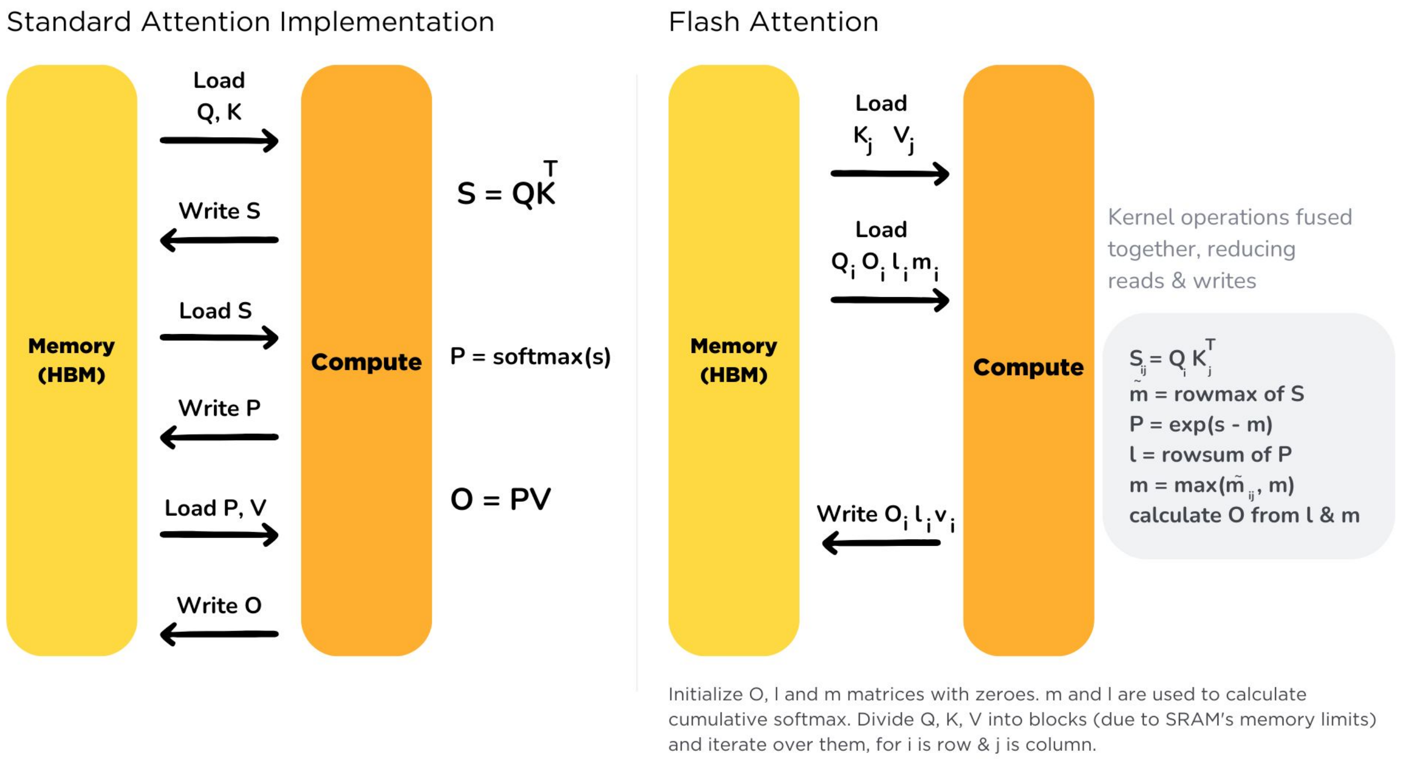


image source: [Flash Attention]

Transformer [Flash Attention]

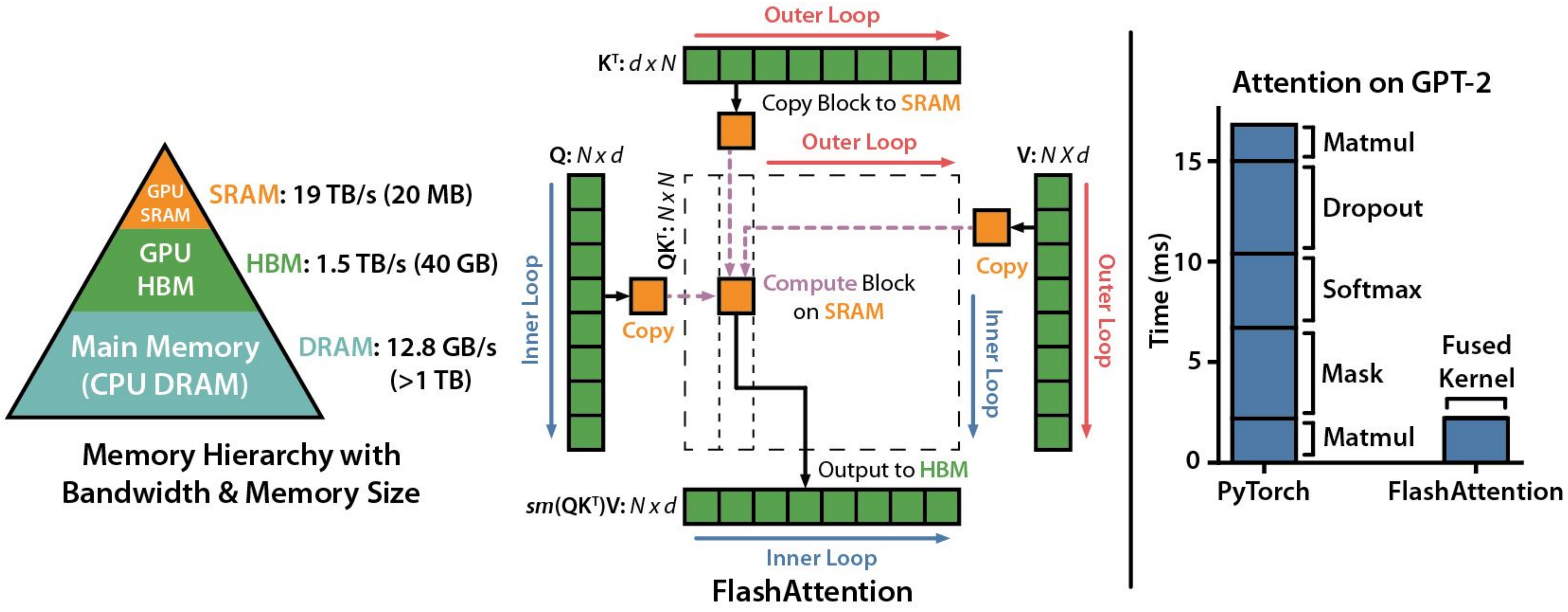
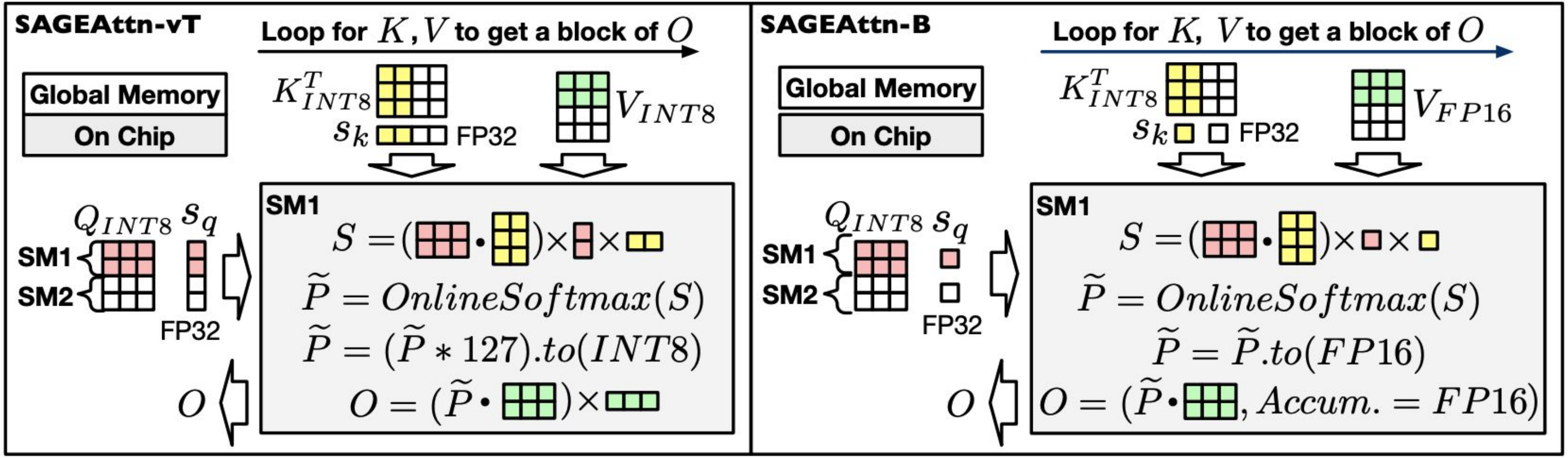


image source: [GitHub - Dao-AILab/flash-attention: Fast and memory-efficient exact attention]

Transformer [Sage Attention]



(a) SageAttention (per-token quantize Q,K; INT8 V) (b) SageAttention (per-block quantize Q,K; FP16 V)

image source: 2410.02367

Transformer [another view]

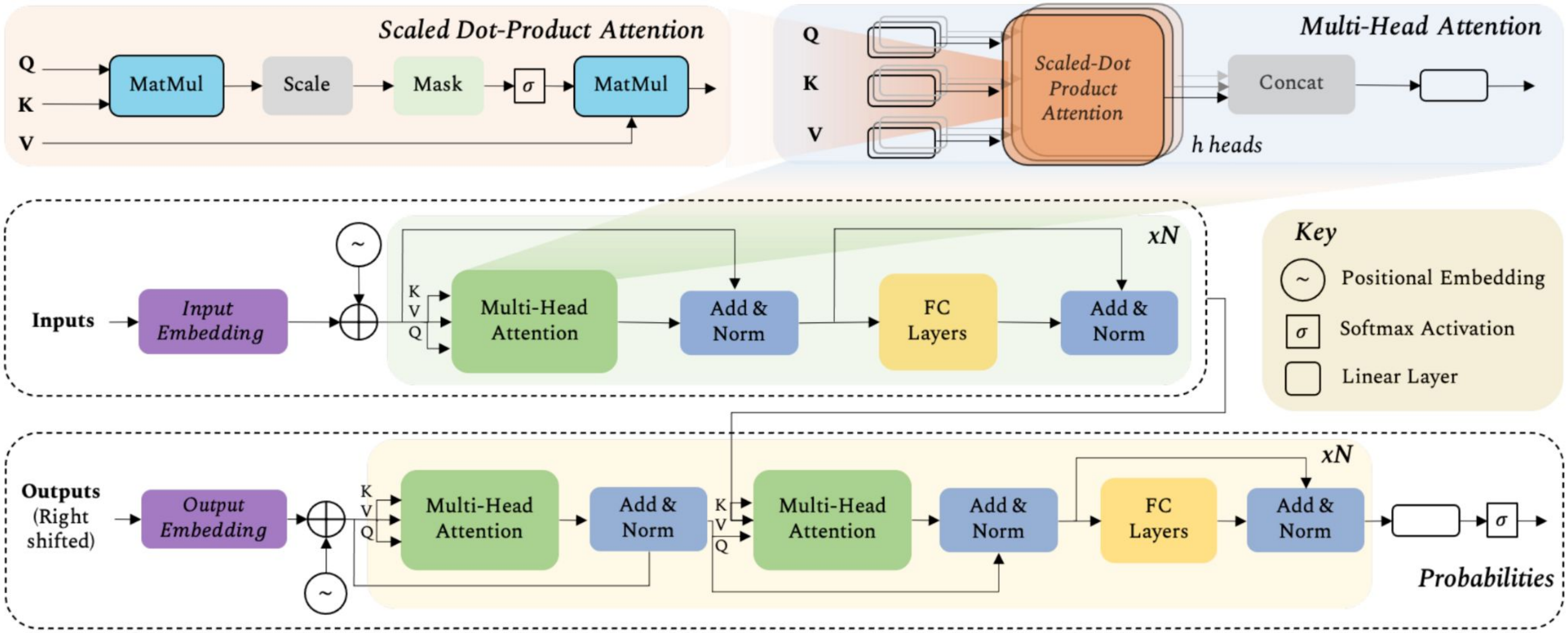
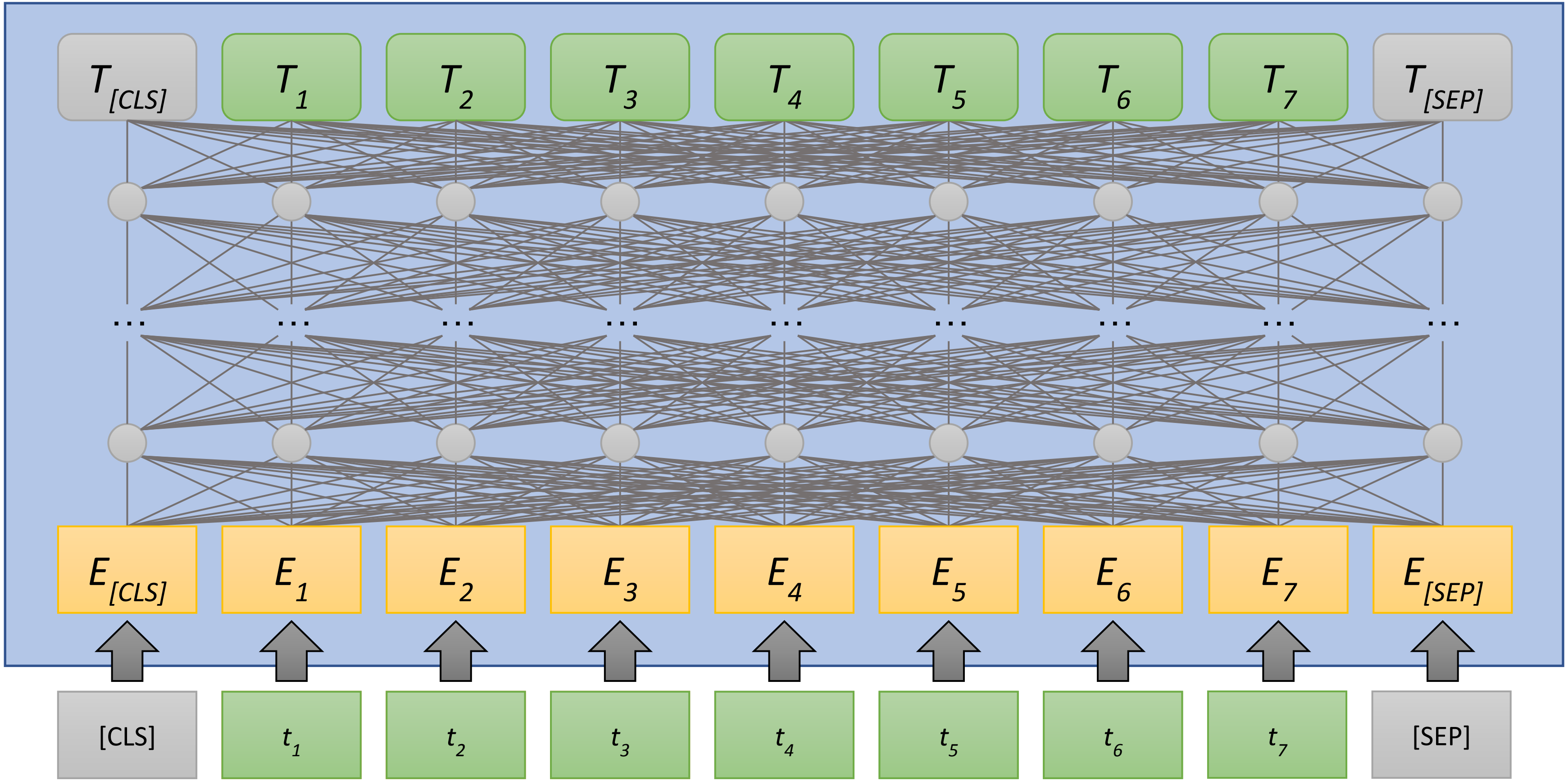
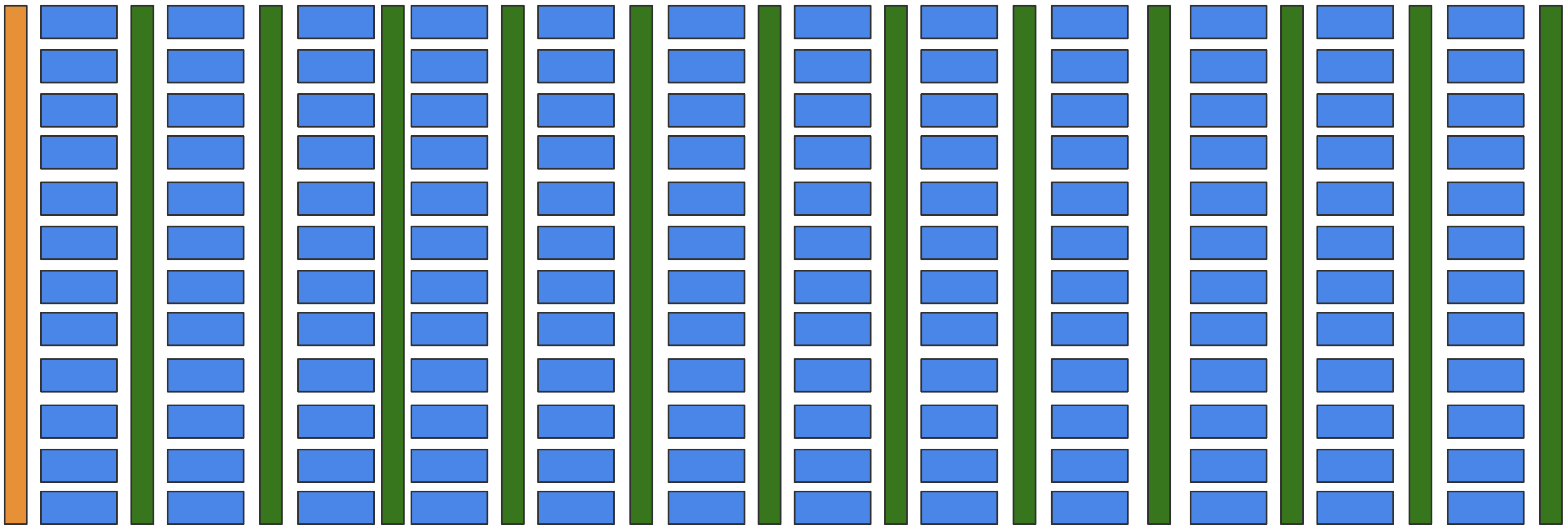


image source: [2101.01169]

Transformer [another view]



Transformer [another view]



L = 12 (bert-base), 12 x 12 = 144 heads