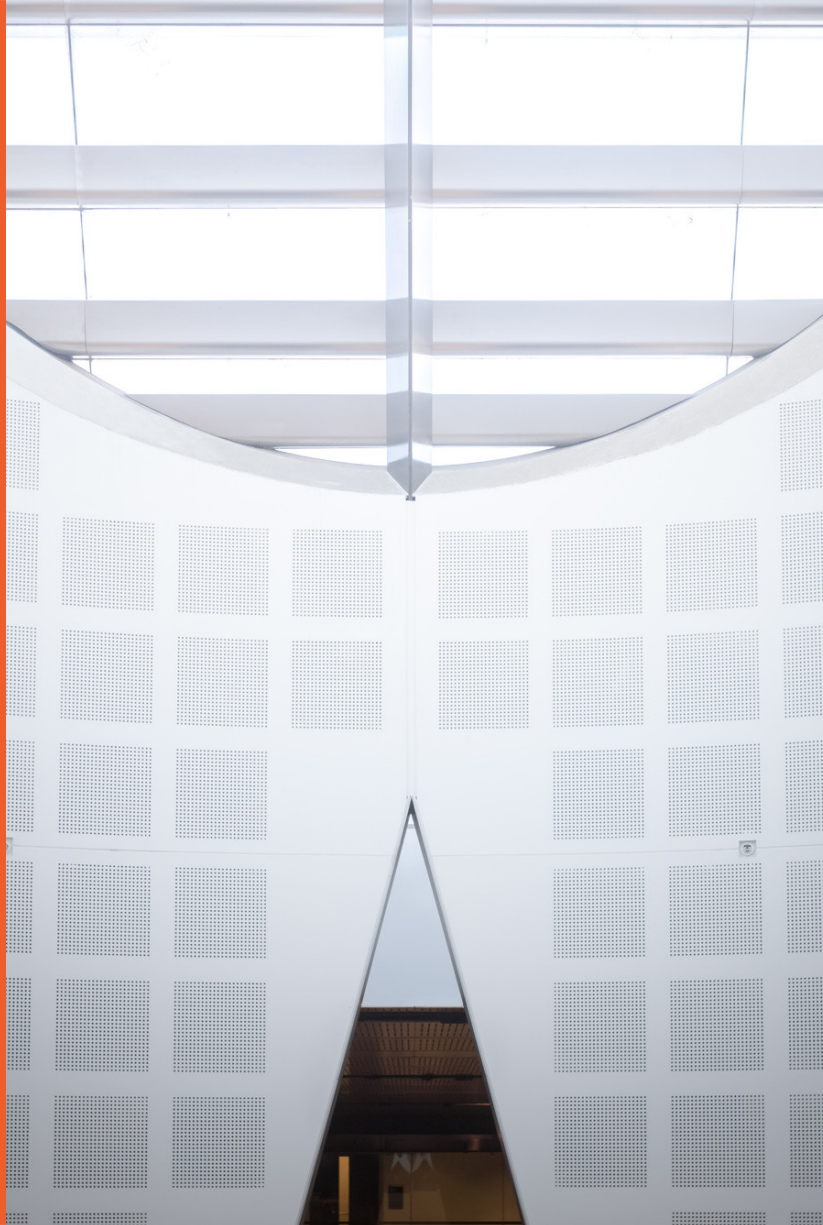
Transformer Neural Networks

Dr Chang Xu

School of Computer Science

THE UNIVERSITY OF SYDNEY



AI's Big Breakthroughs in 2020

GPT-3, the 3rd version of *Generative Pre-Trained* ***Transformer*** model released by OpenAI.

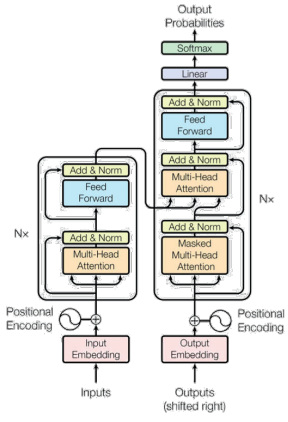
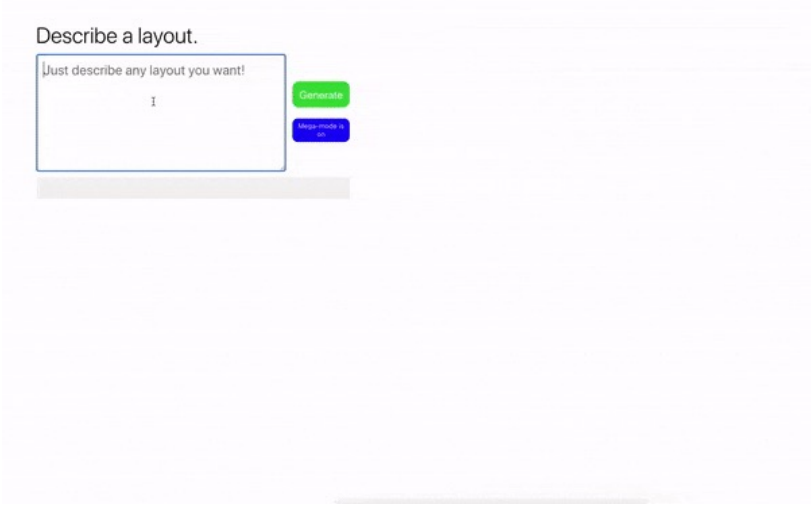
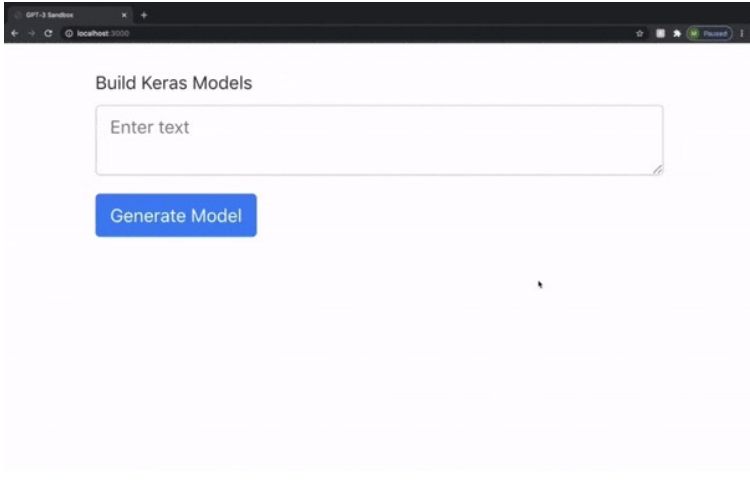


Figure 1: The Transformer - model architecture.

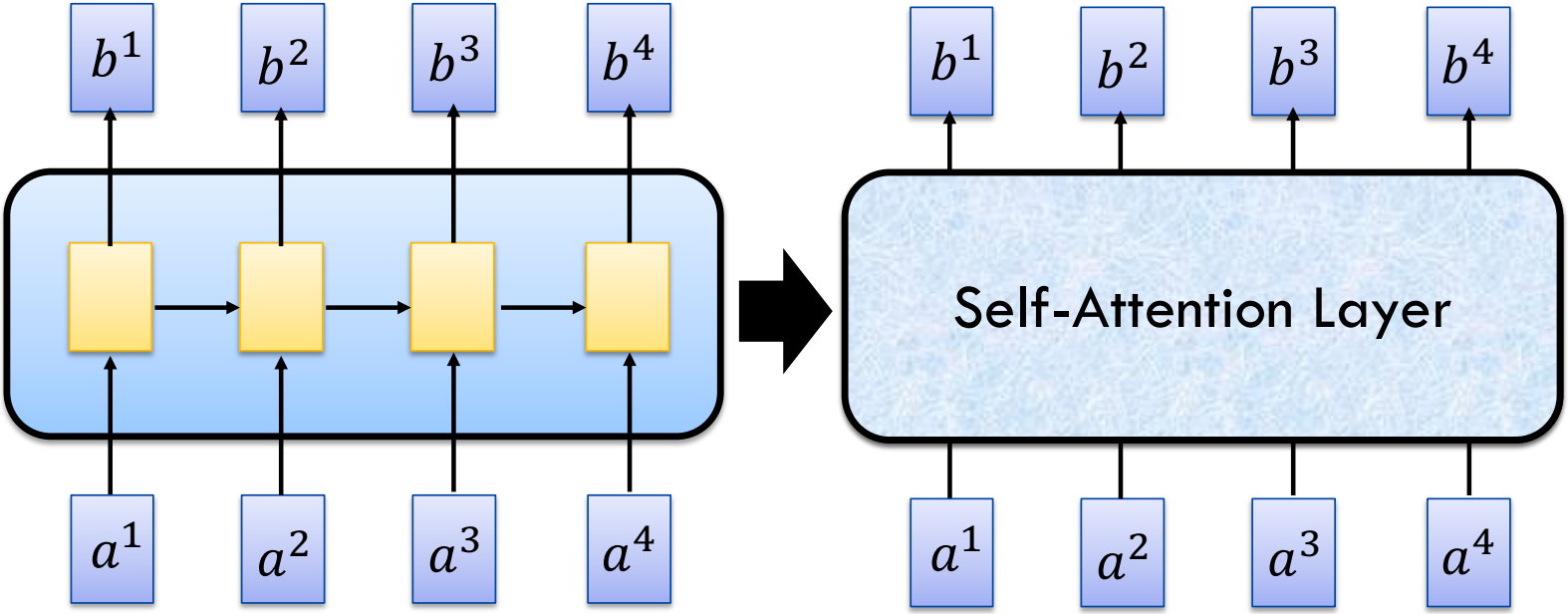




Develop Web apps. Enter a sentence describing Google home page layout, and here you see GPT-3 generating the code for it!

Building ML Models. Keras code written by GPT-3 while input was simple plain text of what ML model do we want to write code for, and boom, it generated the model.

Self-Attention

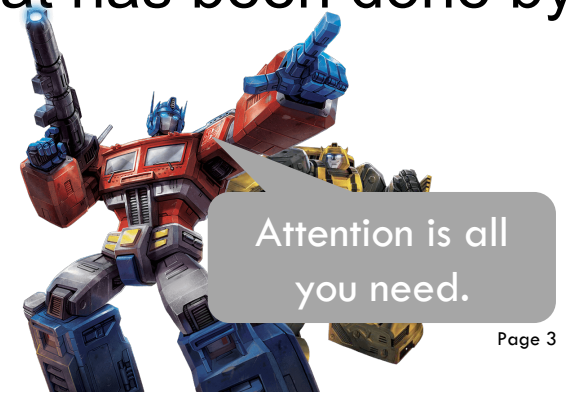


You can try to replace any thing that has been done by RNN with self-attention.

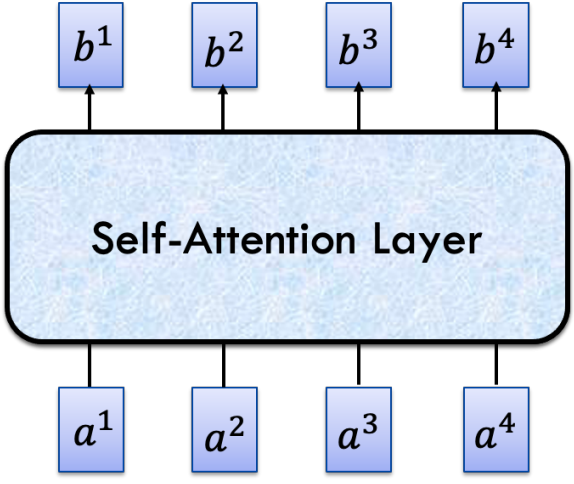
Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

6 Dec 2017



Self-Attention



q: **query** (to match others)

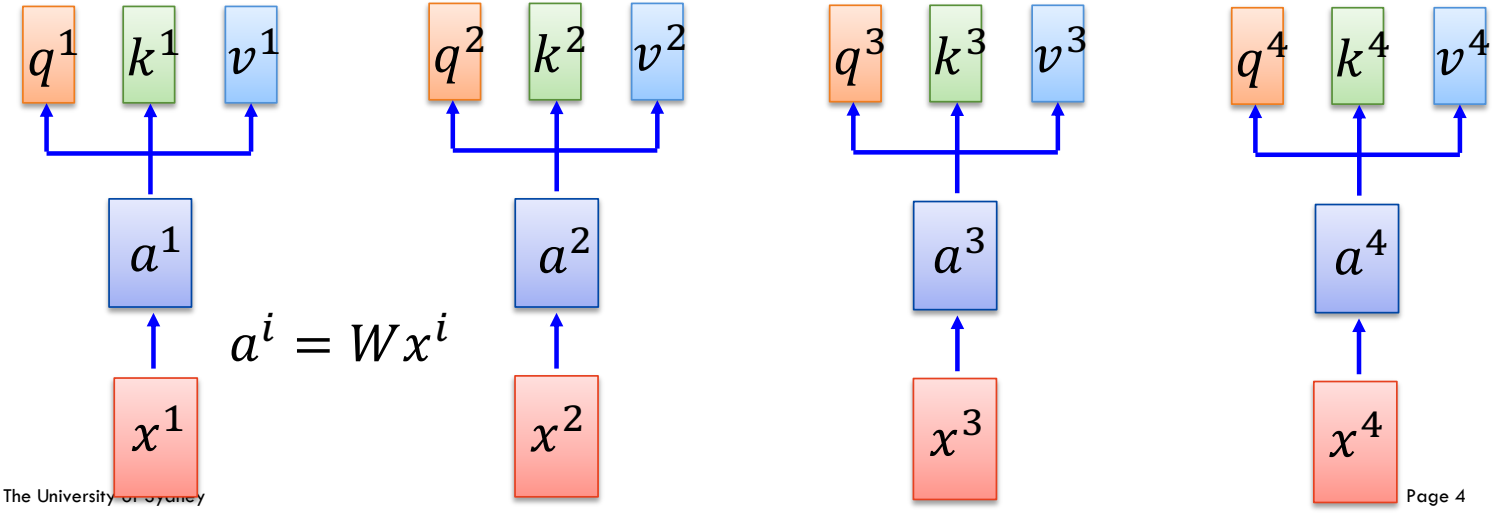
$$q^i = W^q a^i$$

k: **key** (to be matched)

$$k^i = W^k a^i$$

v: **value** (information to be extracted)

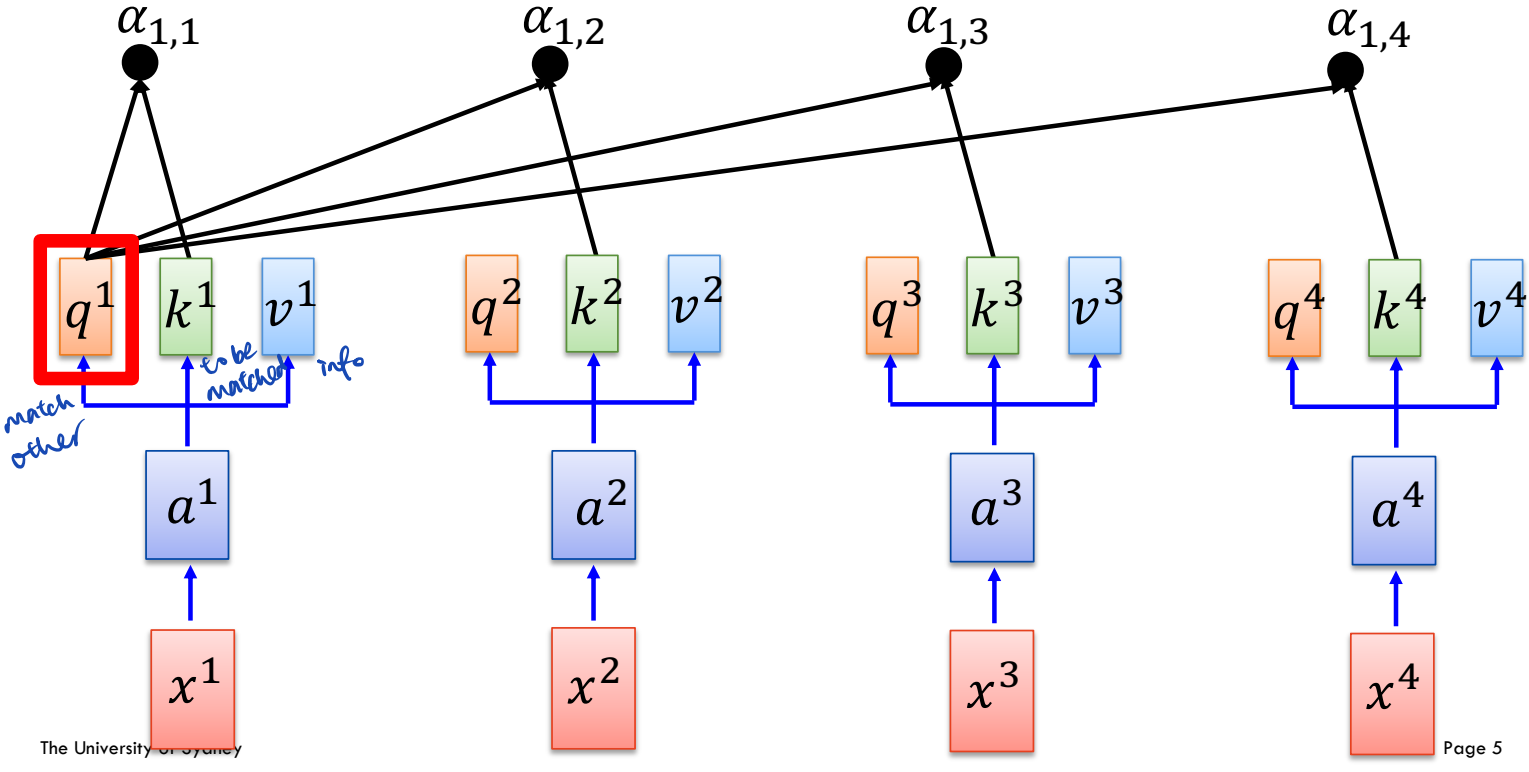
$$v^i = W^v a^i$$



Self-Attention

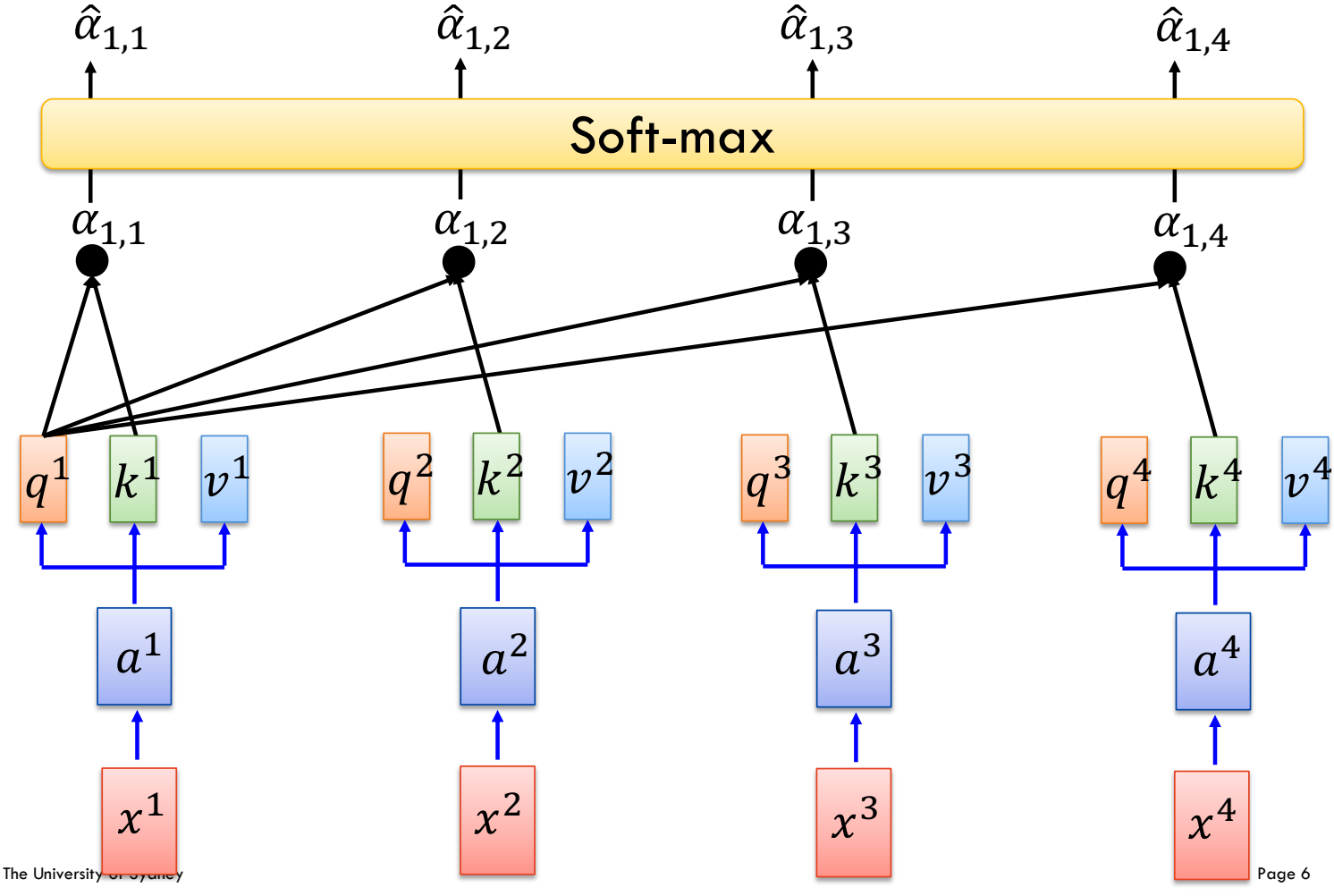
d is the dim of q and k

Scaled Dot-Product Attention: $\alpha_{1,i} = \underbrace{q^1 \cdot k^i}_{\text{dot product}} / \sqrt{d}$



Self-Attention

$$\hat{\alpha}_{1,i} = exp(\alpha_{1,i}) / \sum_j exp(\alpha_{1,j})$$

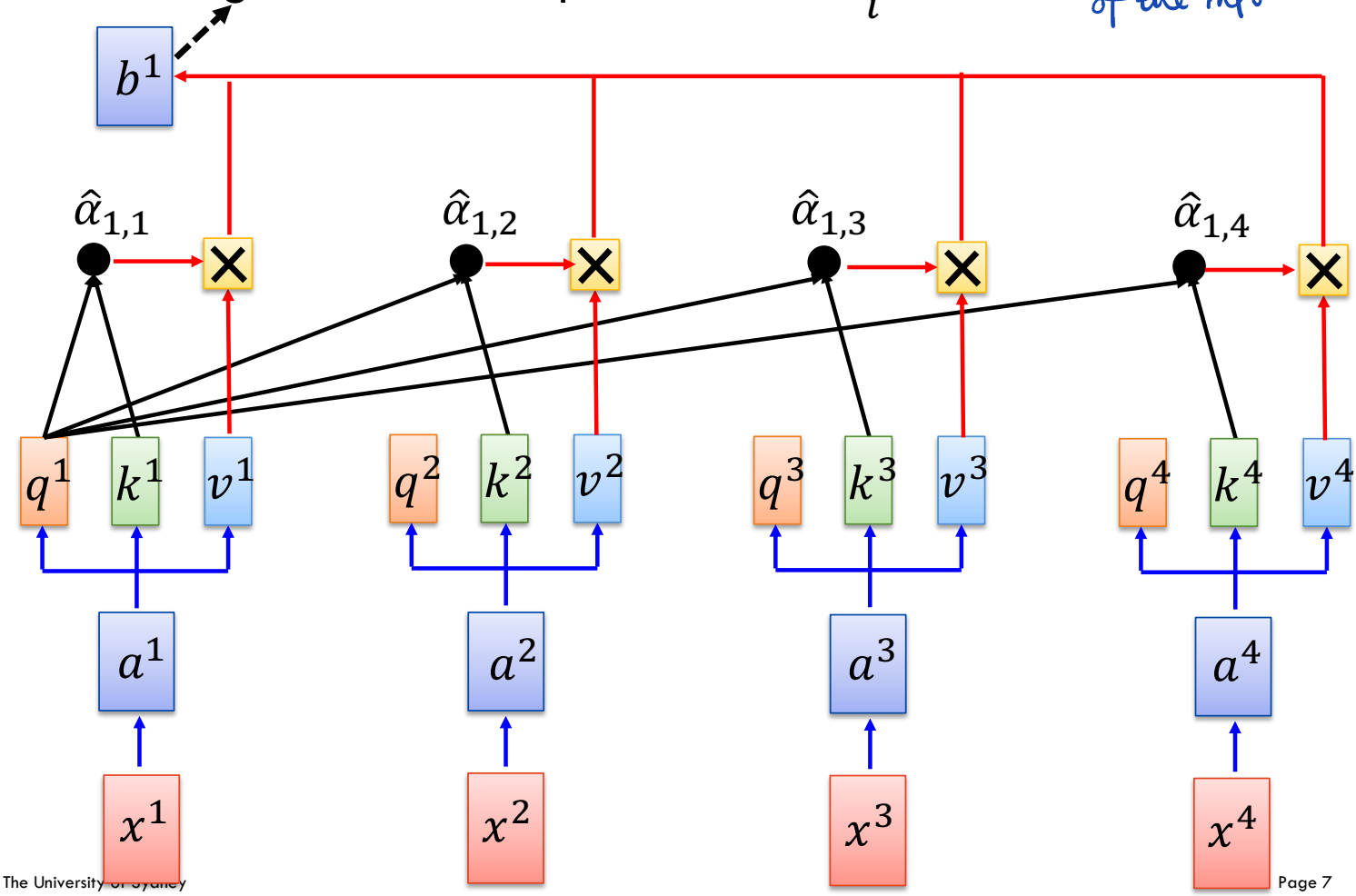


Self-Attention

Considering the whole sequence

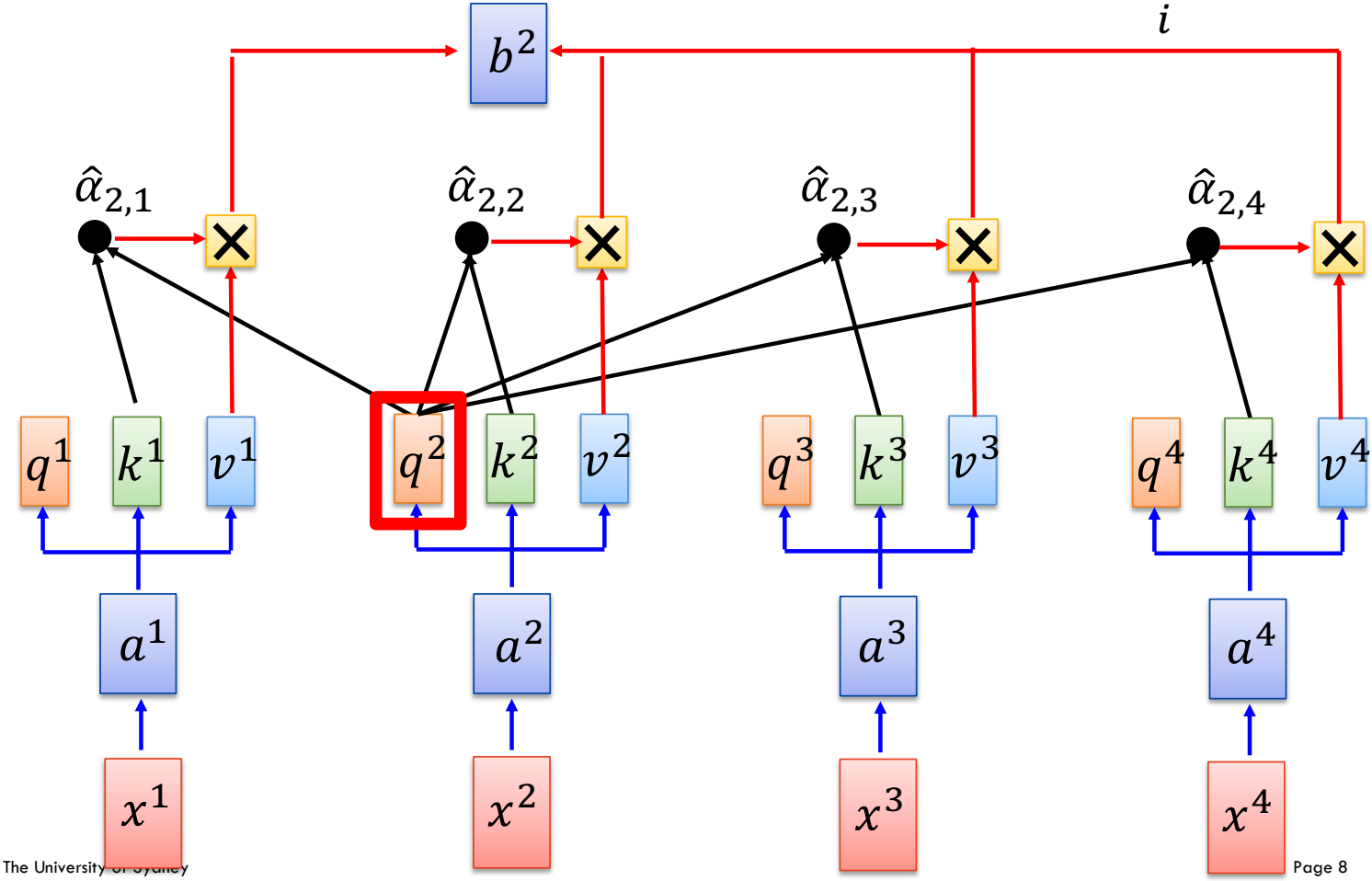
$$b^1 = \sum_i \hat{\alpha}_{1,i} v^i$$

weighted sum of the info



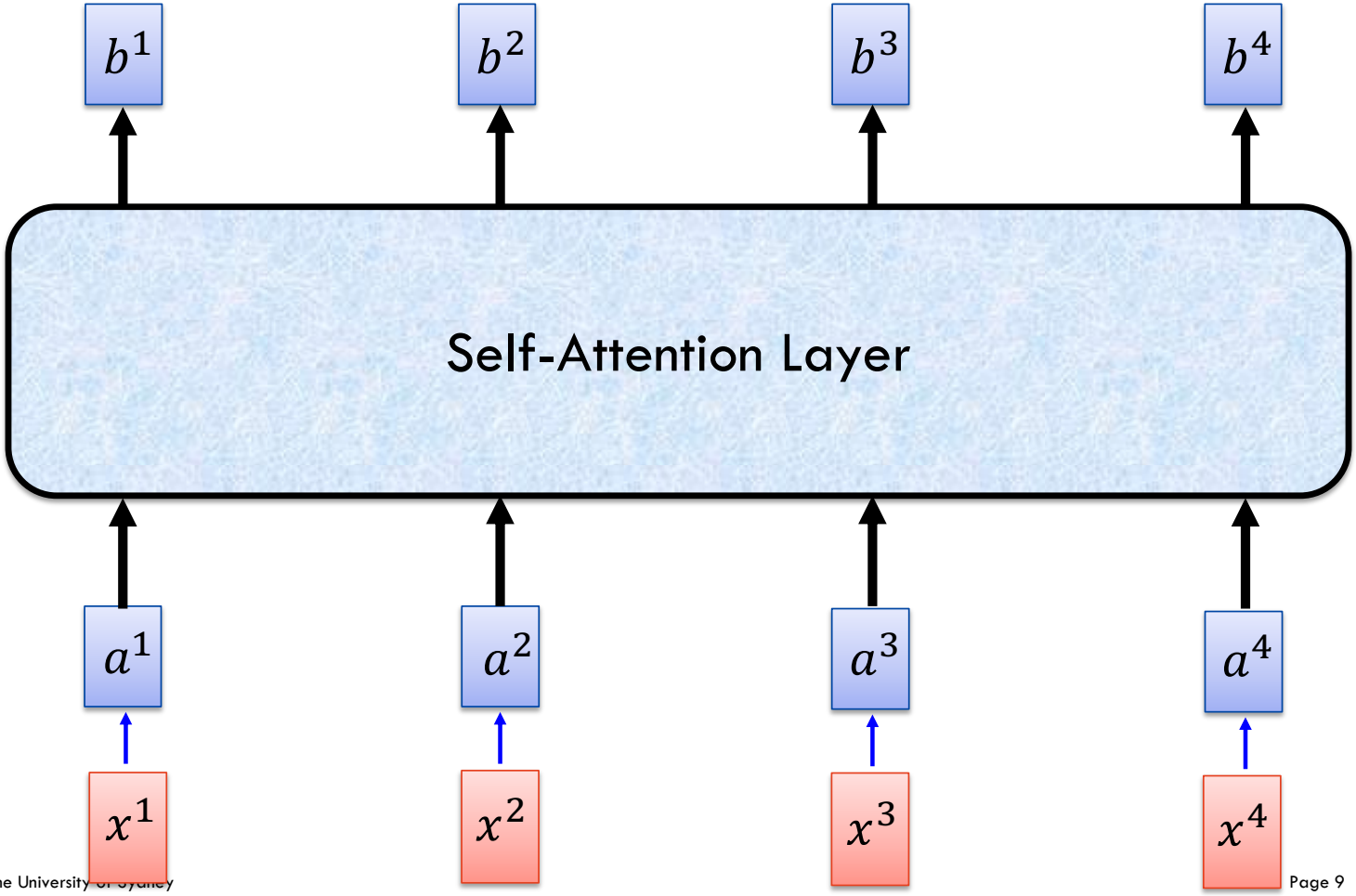
Self-Attention

$$b^2 = \sum_i \hat{\alpha}_{2,i} v^i$$



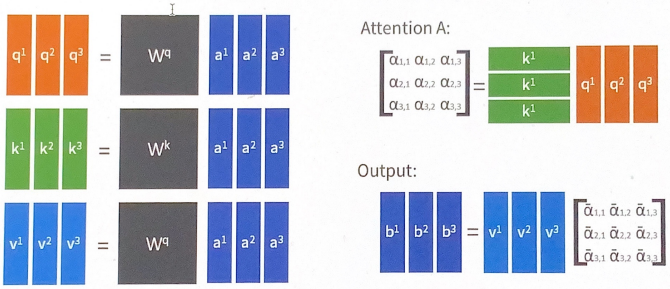
Self-Attention

b^1, b^2, b^3, b^4 can be parallelly computed.



and concludes a output b which contains relations between input x and all other words.

Matricize



All calculations happen in Self-Attention Layer is Matrix calculations

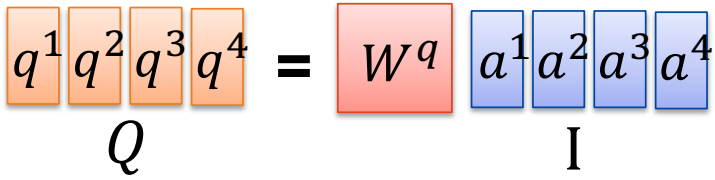
As we mentioned above, every calculations within Self-Attention Layer is Matrix

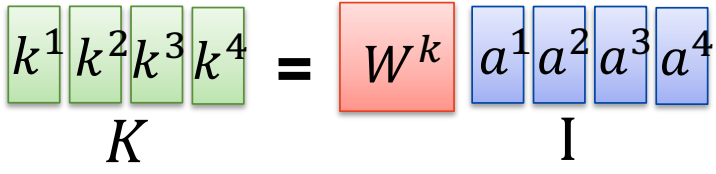
Self-Attention

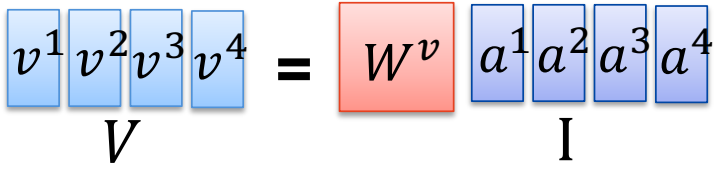
$$q^i = W^q a^i$$

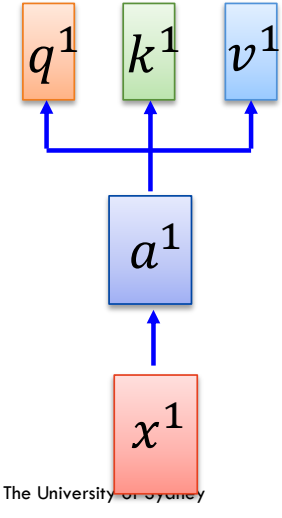
$$k^i = W^k a^i$$

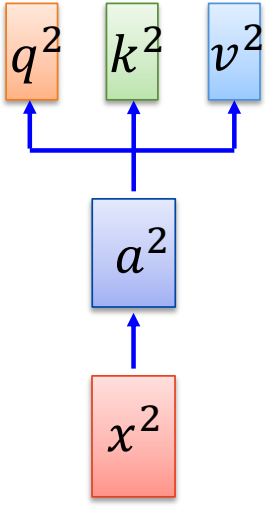
$$v^i = W^v a^i$$

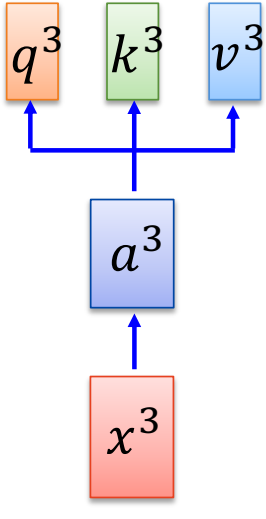


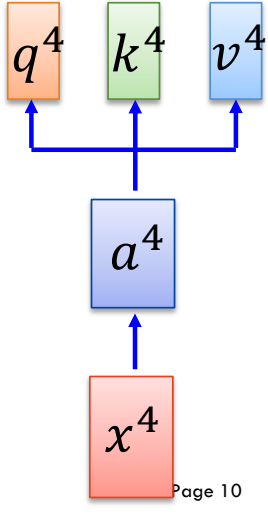




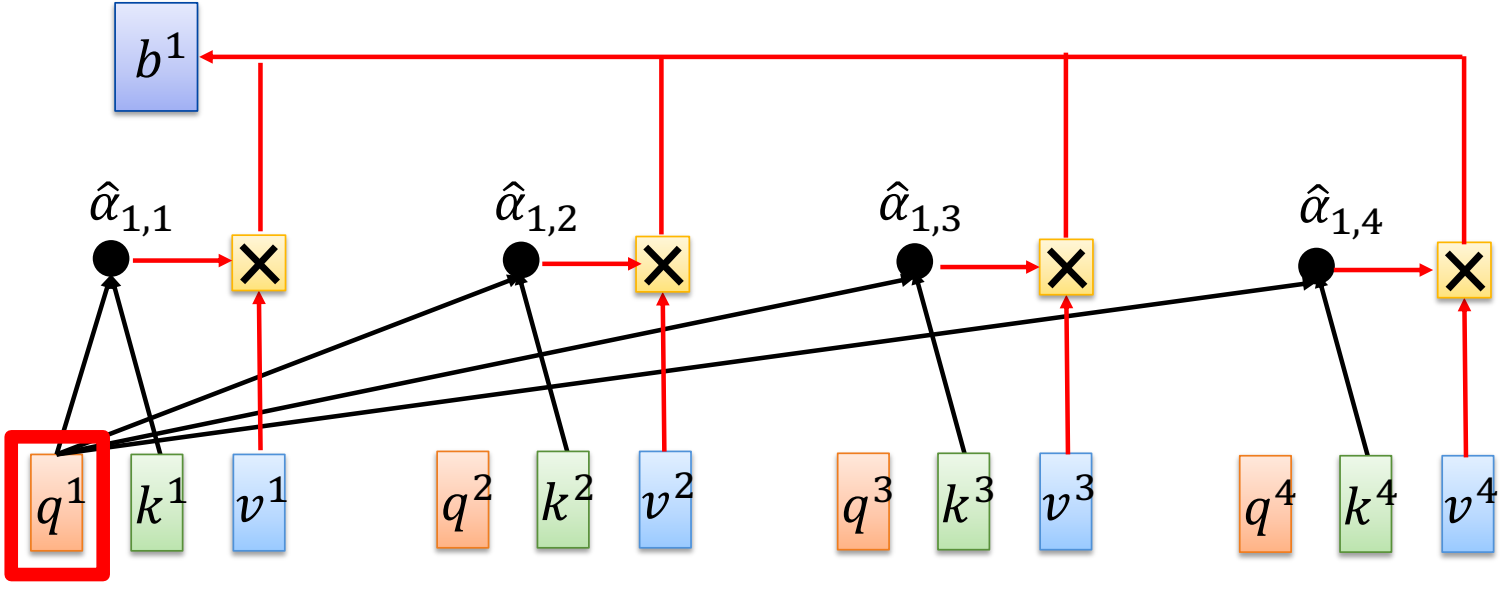








Self-Attention



$\alpha_{1,1} = k^1 q^1$ $\alpha_{1,2} = k^2 q^1$

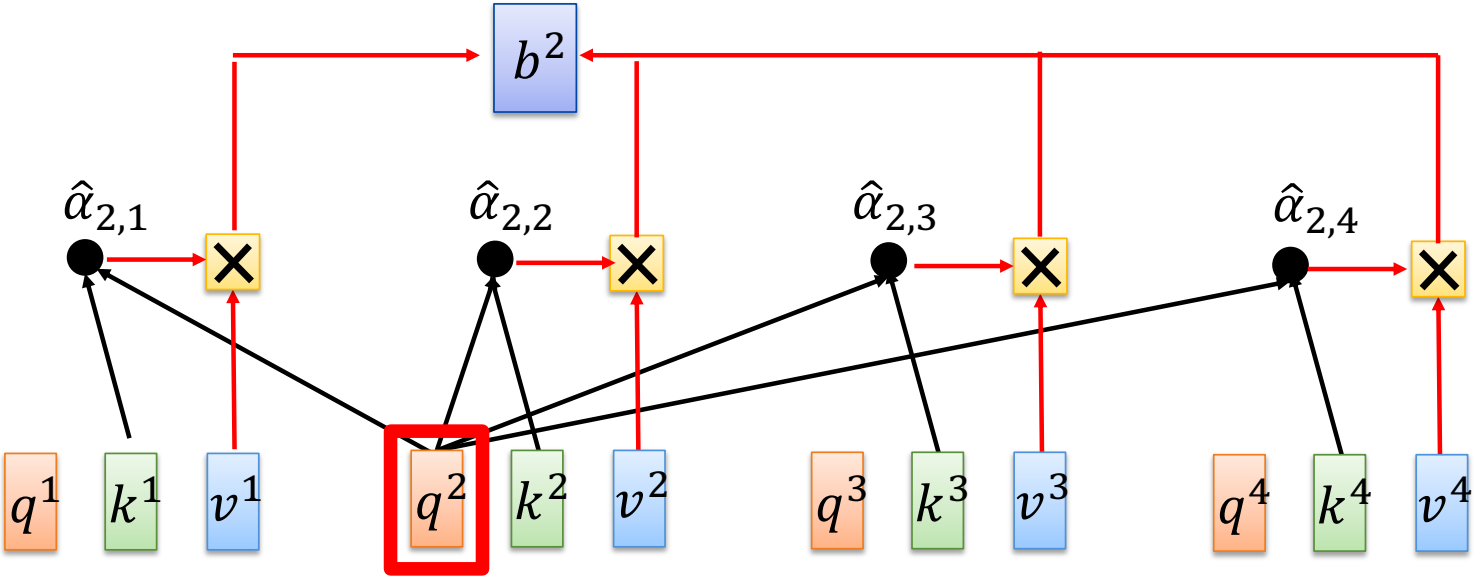
$\alpha_{1,3} = k^3 q^1$ $\alpha_{1,4} = k^4 q^1$

$$\begin{bmatrix}\alpha_{1,1}\\ \alpha_{1,2}\\ \alpha_{1,3}\\ \alpha_{1,4}\end{bmatrix} = \begin{bmatrix}k^1\\ k^2\\ k^3\\ k^4\end{bmatrix} q^1$$

(ignore $\sqrt{d}$ for simplicity)

Self-Attention

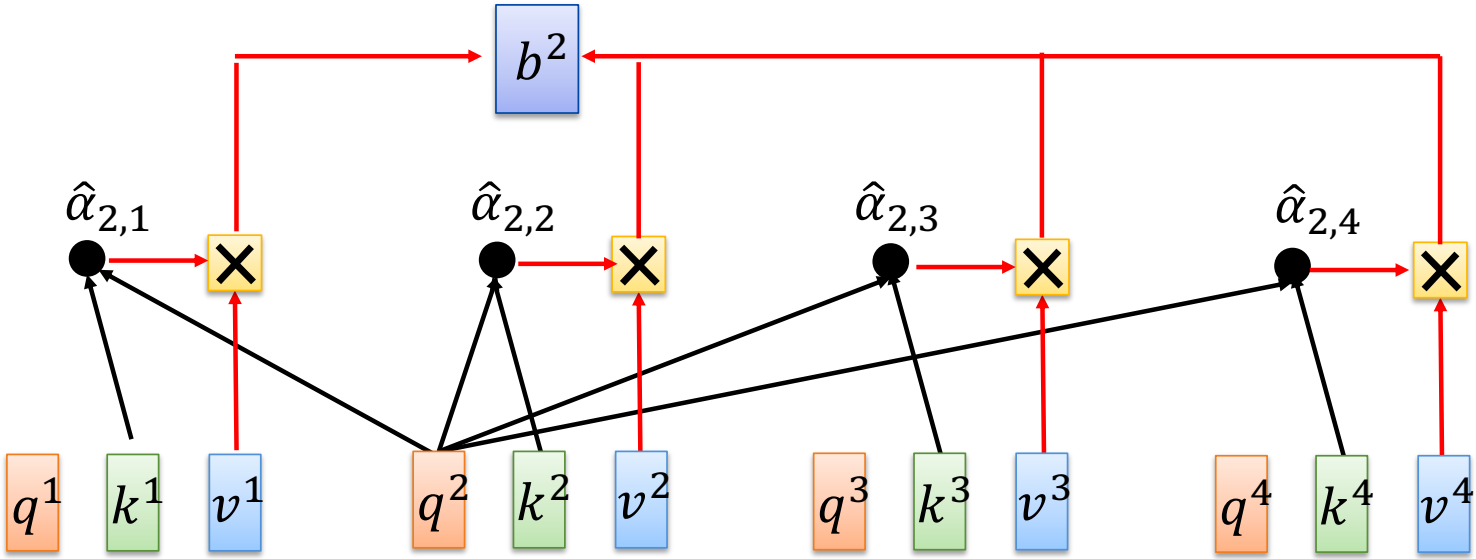
$$b^2 = \sum_i \hat{\alpha}_{2,i} v^i$$



$$\begin{bmatrix} \hat{\alpha}_{1,1} & \hat{\alpha}_{2,1} & \hat{\alpha}_{3,1} & \hat{\alpha}_{4,1} \\ \hat{\alpha}_{1,2} & \hat{\alpha}_{2,2} & \hat{\alpha}_{3,2} & \hat{\alpha}_{4,2} \\ \hat{\alpha}_{1,3} & \hat{\alpha}_{2,3} & \hat{\alpha}_{3,3} & \hat{\alpha}_{4,3} \\ \hat{\alpha}_{1,4} & \hat{\alpha}_{2,4} & \hat{\alpha}_{3,4} & \hat{\alpha}_{4,4} \end{bmatrix} \hat{A} \quad \text{softmax} \quad \begin{bmatrix} \alpha_{1,1} & \alpha_{2,1} & \alpha_{3,1} & \alpha_{4,1} \\ \alpha_{1,2} & \alpha_{2,2} & \alpha_{3,2} & \alpha_{4,2} \\ \alpha_{1,3} & \alpha_{2,3} & \alpha_{3,3} & \alpha_{4,3} \\ \alpha_{1,4} & \alpha_{2,4} & \alpha_{3,4} & \alpha_{4,4} \end{bmatrix} A = \begin{bmatrix} k^1 \\ k^2 \\ k^3 \\ k^4 \end{bmatrix} K^T \begin{bmatrix} q^1 & q^2 & q^3 & q^4 \end{bmatrix} Q$$

Self-Attention

$$b^2 = \sum_i \hat{\alpha}_{2,i} v^i$$



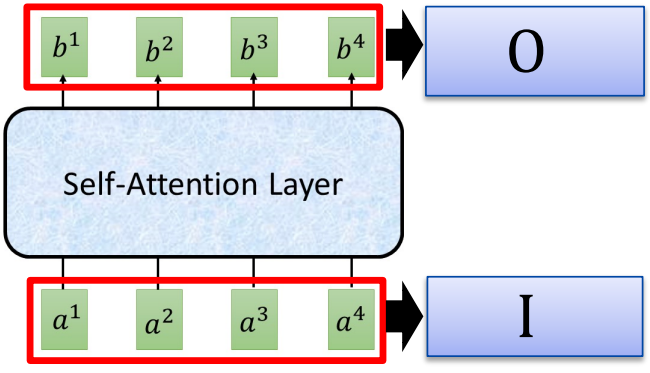
$$\begin{matrix} b^1 & b^2 & b^3 & b^4 \end{matrix} = \begin{matrix} v^1 & v^2 & v^3 & v^4 \end{matrix}$$

$$O = V$$

$$\begin{matrix} \hat{\alpha}_{1,1} & \hat{\alpha}_{2,1} & \hat{\alpha}_{3,1} & \hat{\alpha}_{4,1} \\ \hat{\alpha}_{1,2} & \hat{\alpha}_{2,2} & \hat{\alpha}_{3,2} & \hat{\alpha}_{4,2} \\ \hat{\alpha}_{1,3} & \hat{\alpha}_{2,3} & \hat{\alpha}_{3,3} & \hat{\alpha}_{4,3} \\ \hat{\alpha}_{1,4} & \hat{\alpha}_{2,4} & \hat{\alpha}_{3,4} & \hat{\alpha}_{4,4} \end{matrix}$$

$$\hat{A}$$

Self-Attention



O

I

$$Q = W^q I$$

$$K = W^k I$$

$$V = W^v I$$

$$A = K^T Q$$

$\hat{A}$

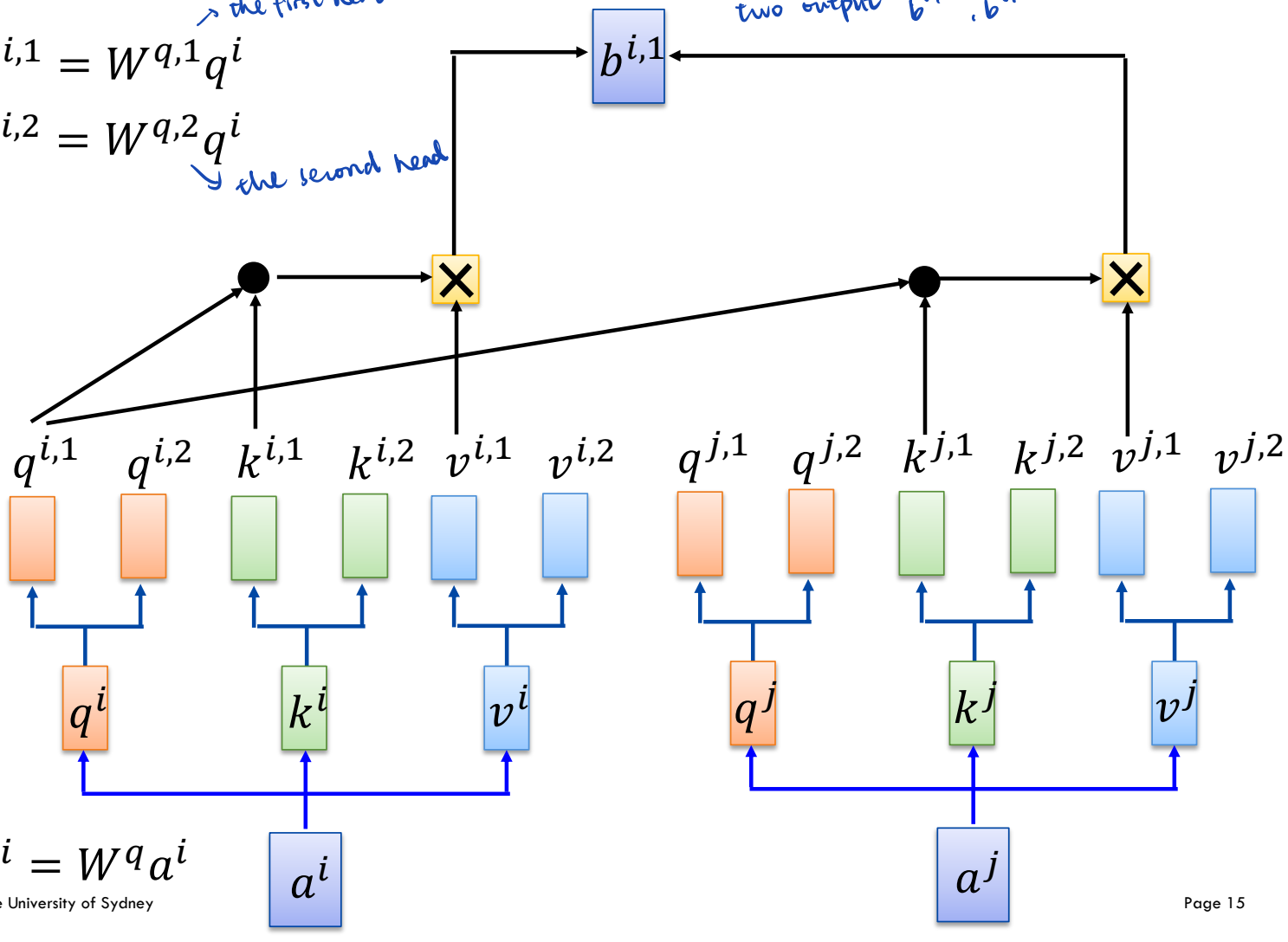
$$O = V \hat{A}$$

Multi-head Self-Attention (2 heads as example)

$q^{i,1} = W^{q,1} q^i$ → the first head

$q^{i,2} = W^{q,2} q^i$ → the second head

two output $b^{i,1}$, $b^{i,2}$.



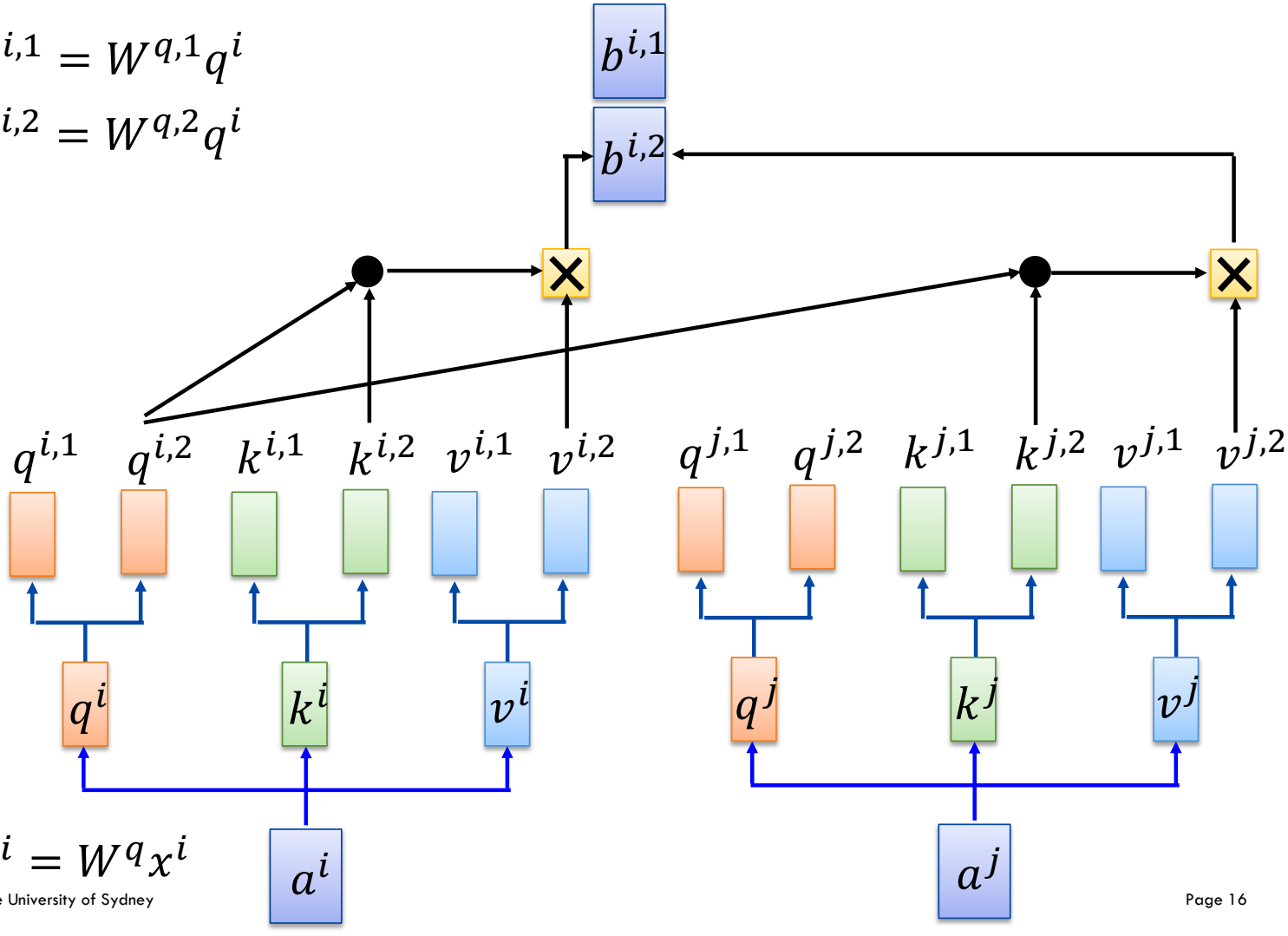
$q^i = W^q a^i$

Multi-head Self-Attention

(2 heads as example)

$$q^{i,1} = W^{q,1} q^i$$

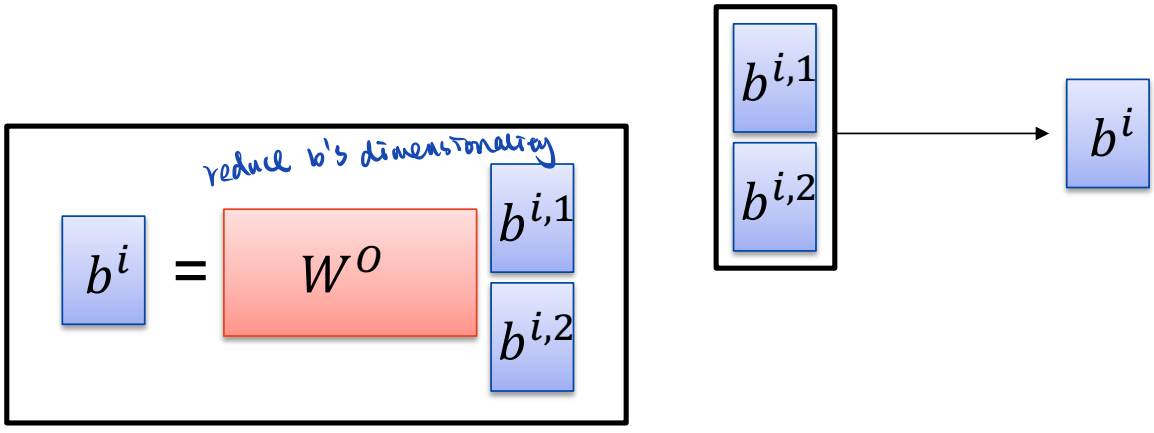
$$q^{i,2} = W^{q,2} q^i$$

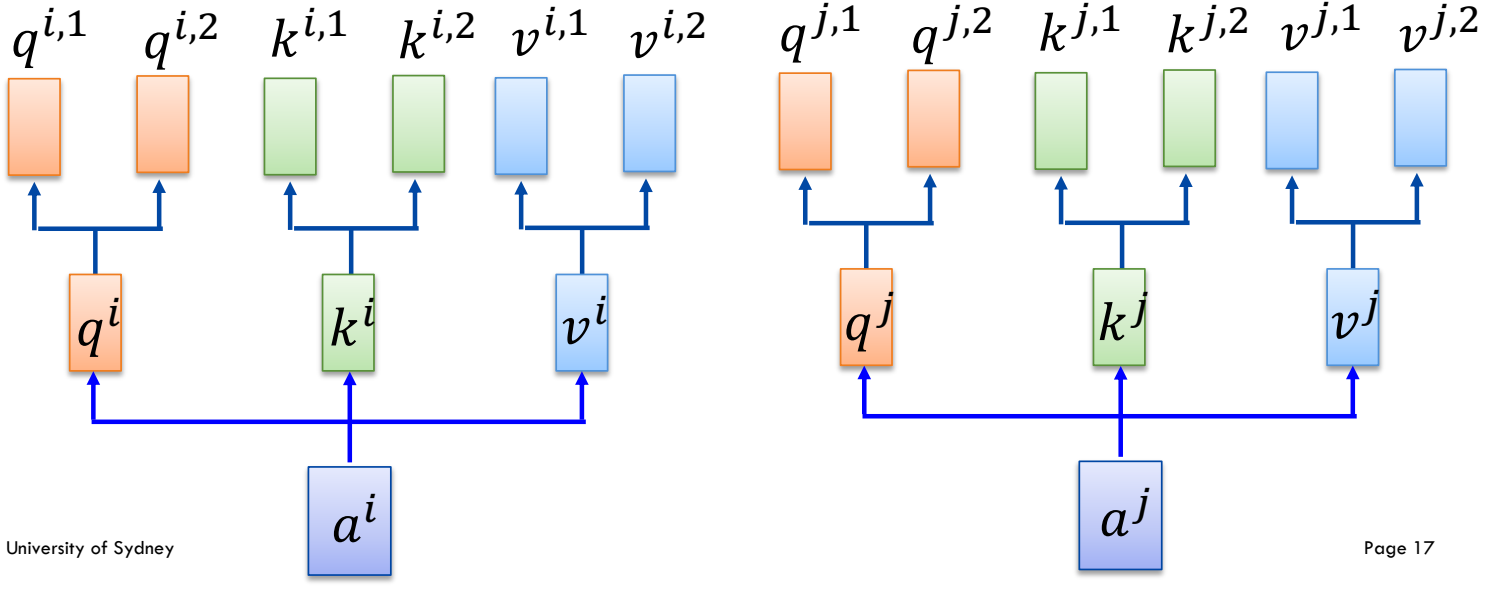


$$q^i = W^q x^i$$

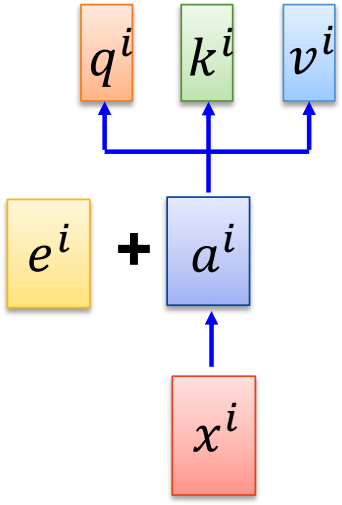
Multi-head Self-Attention

(2 heads as example)

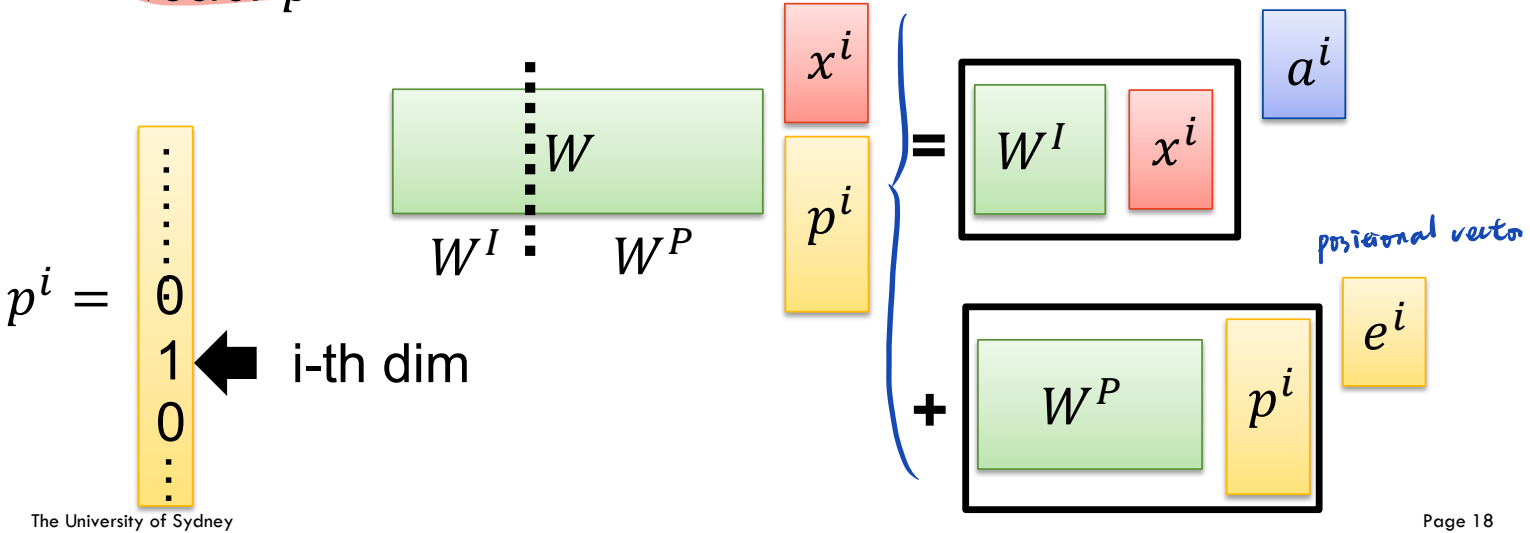


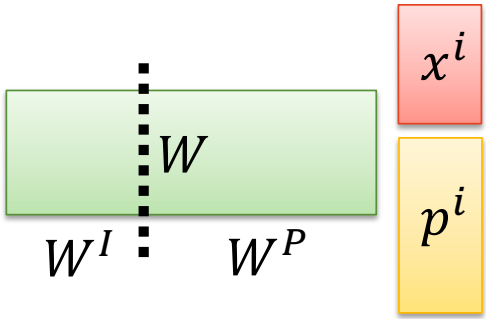


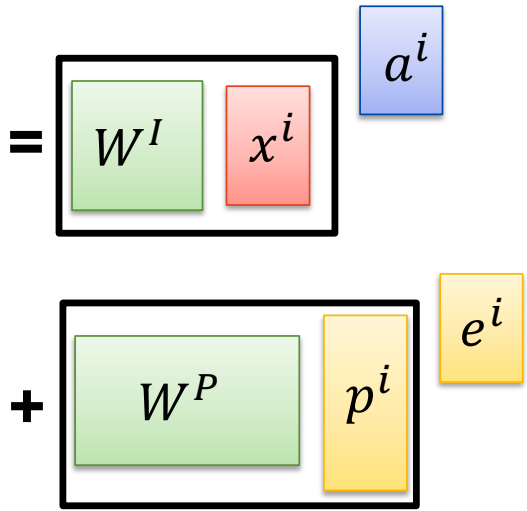
Positional Encoding

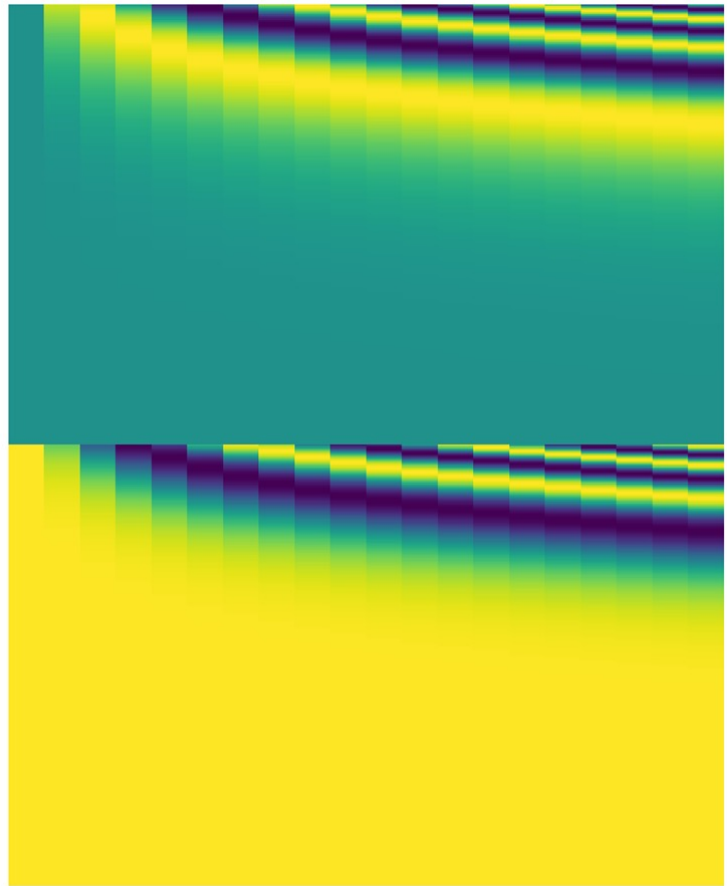


- No position information in self-attention.
- Original paper: each position has a unique positional vector e^i (not learned from data)
- In other words: each x^i appends a one-hot vector p^i









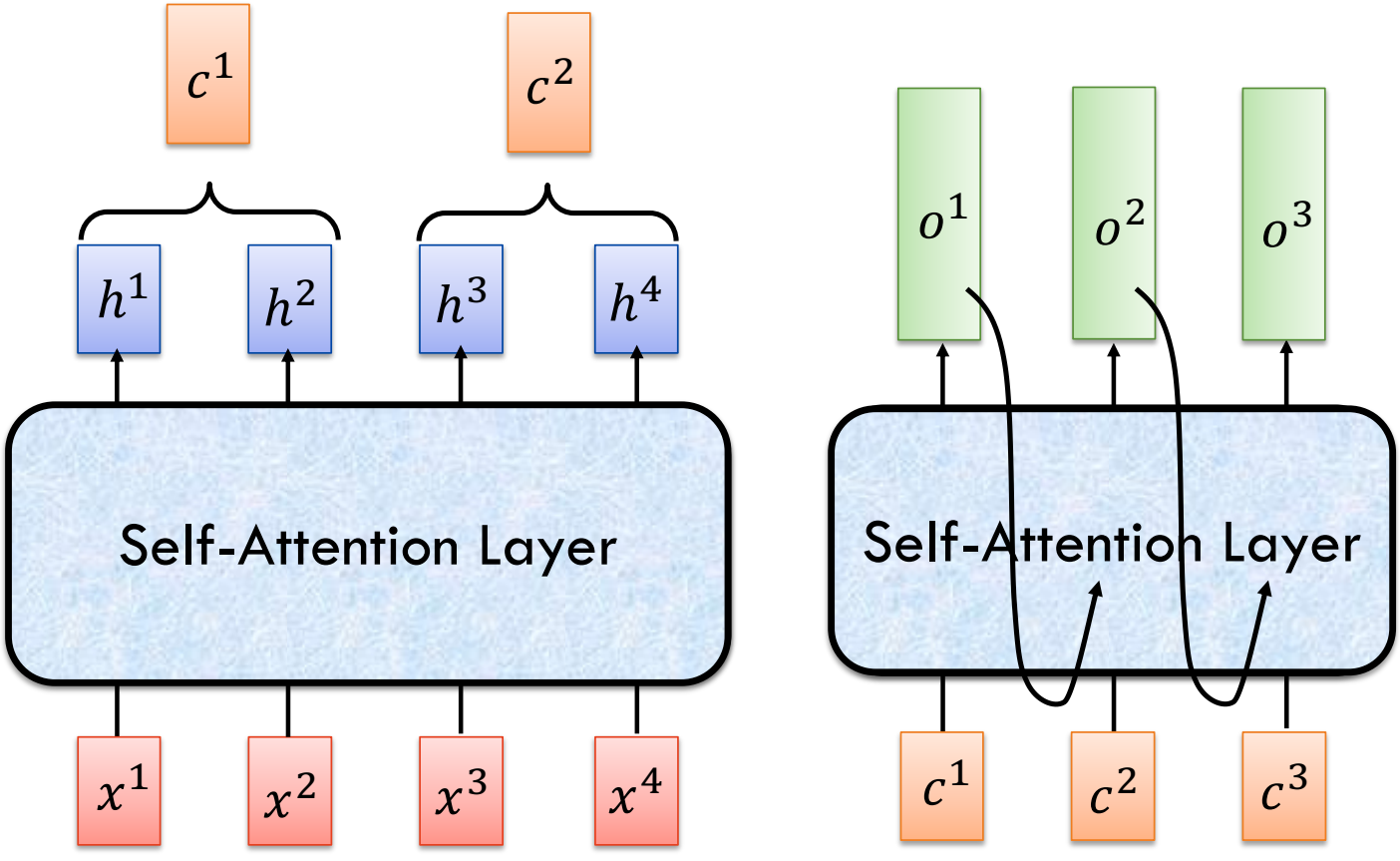
$$e^i(i, 2j) = \sin(i/10000^{2j/d})$$

$$e^i(i, 2j+1) = \cos(i/10000^{2j/d})$$

j is the dimension.

source of image: http://jalammar.github.io/illustrated-transforme

Seq2seq with Attention

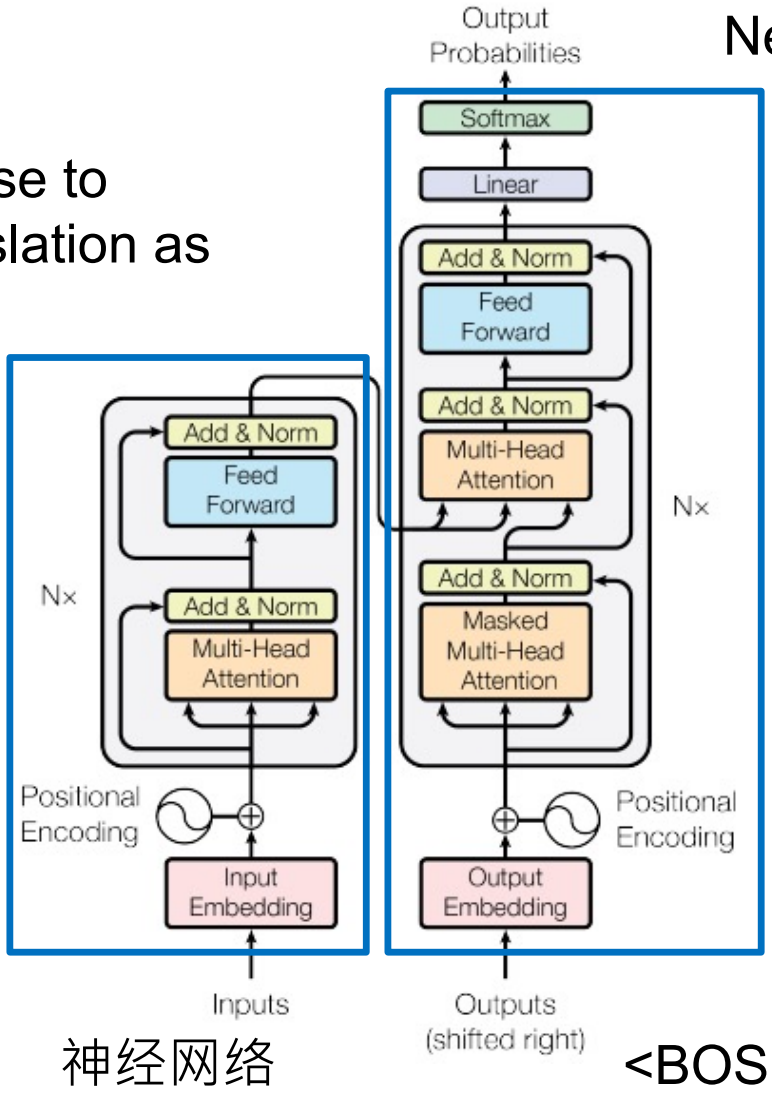


Encoder **Decoder**

Transformer

Neural Network

Using Chinese to English translation as example



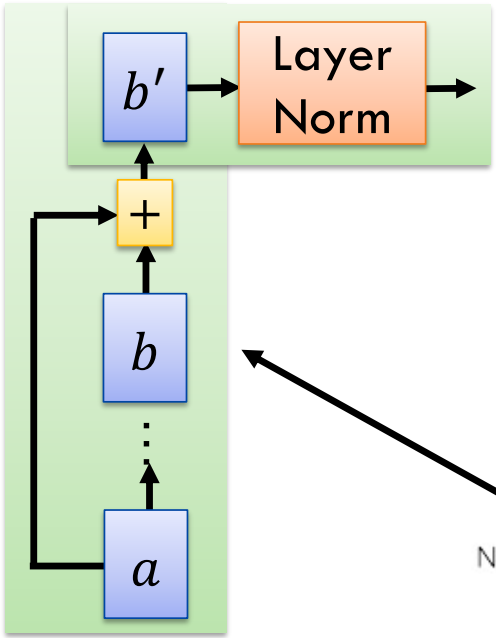
Encoder

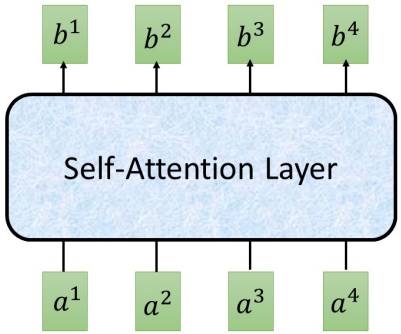
Decoder

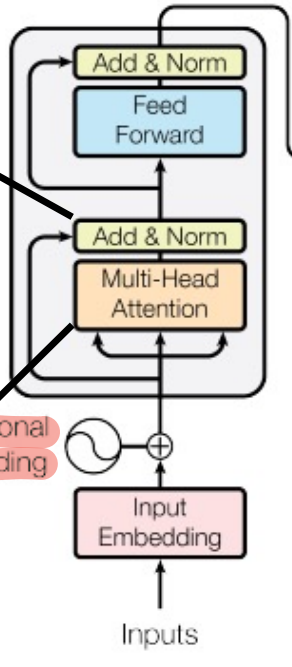
神经网络

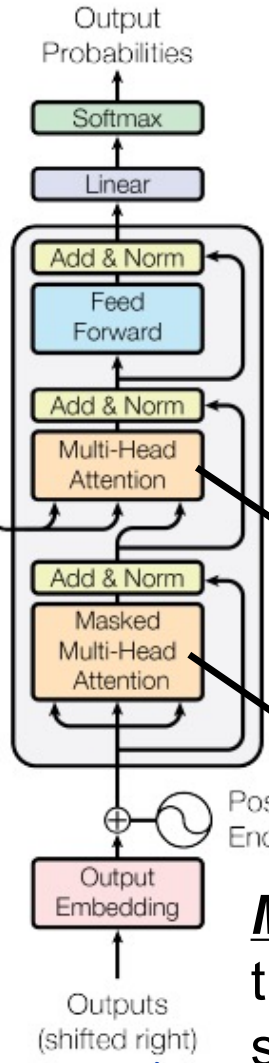
<BOS>

Transformer



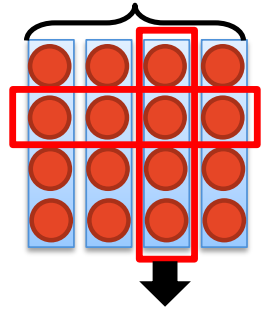






column : instance

Batch Size



$\mu = 0, \sigma = 1$ Batch

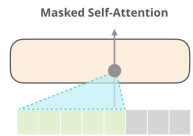
$\mu = 0, \sigma = 1$ Layer

layer norm only relevant to single instance

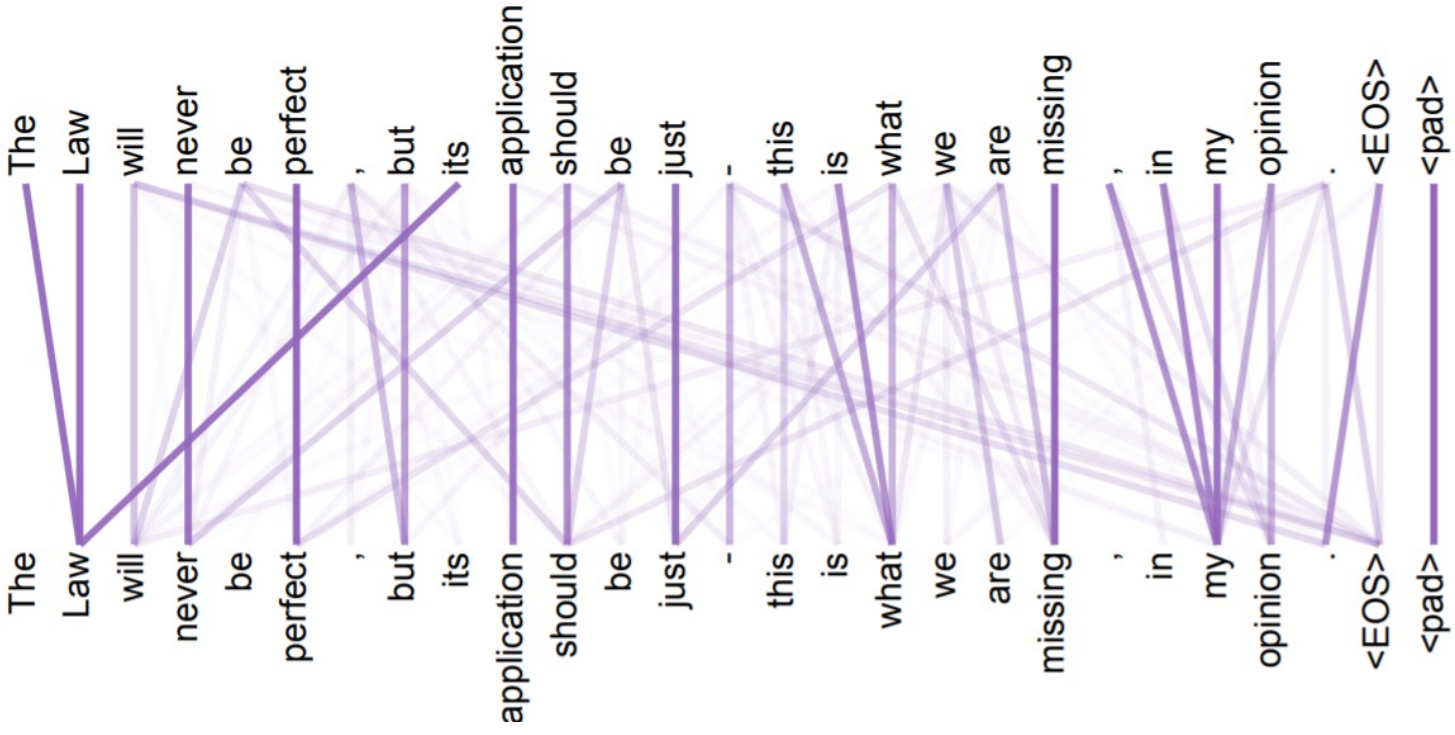
attend on the input sequence

Masked: attend on the generated sequence

Generation step: generate token one by one

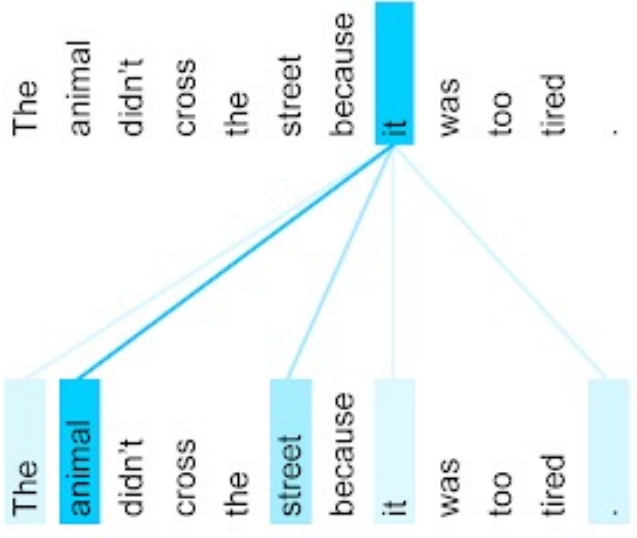


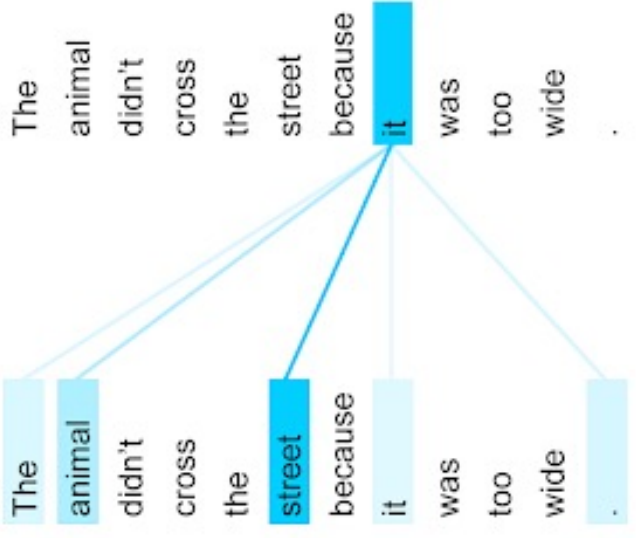
Attention Visualization



https://arxiv.org/abs/1706.03762

Attention Visualization

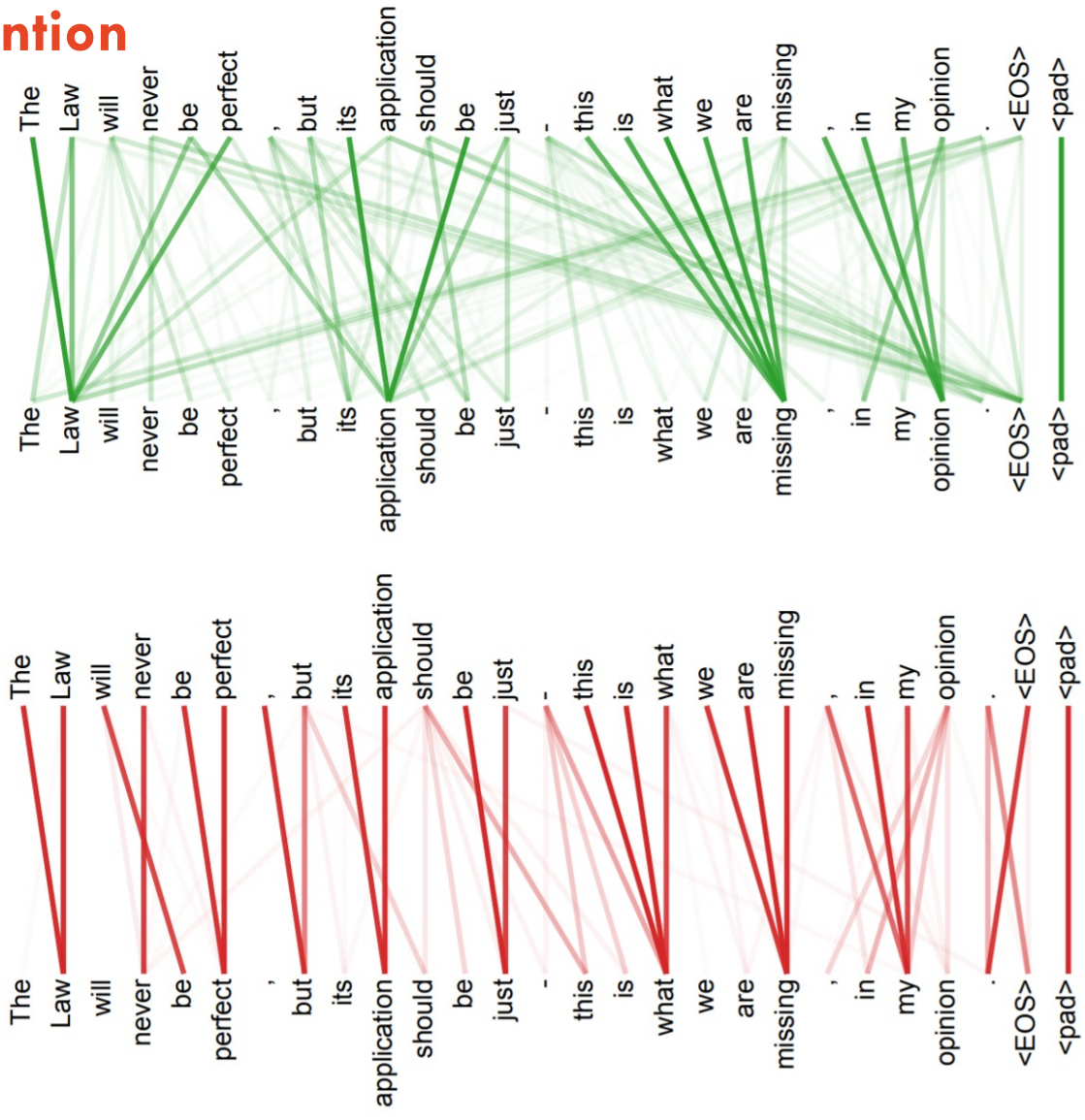




The encoder self-attention distribution for the word “it” from the 5th to the 6th layer of a Transformer trained on English to French translation (one of eight attention heads).

https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html

Multi-head Attention



BERT

A **transformer** uses Encoder stack to model input, and uses Decoder stack to model output (using input information from encoder side).

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**.

1-of-N Encoding

apple = [1 0 0 0 0]

bag = [0 1 0 0 0]

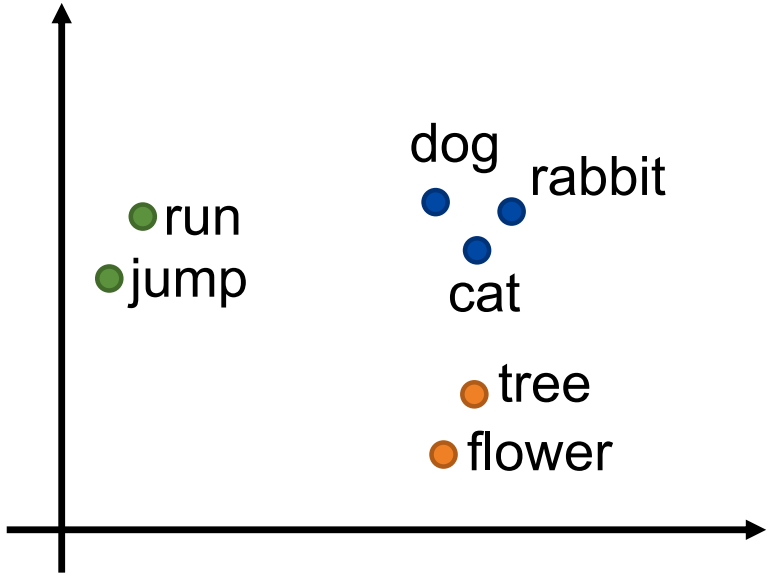
cat = [0 0 1 0 0]

dog = [0 0 0 1 0]

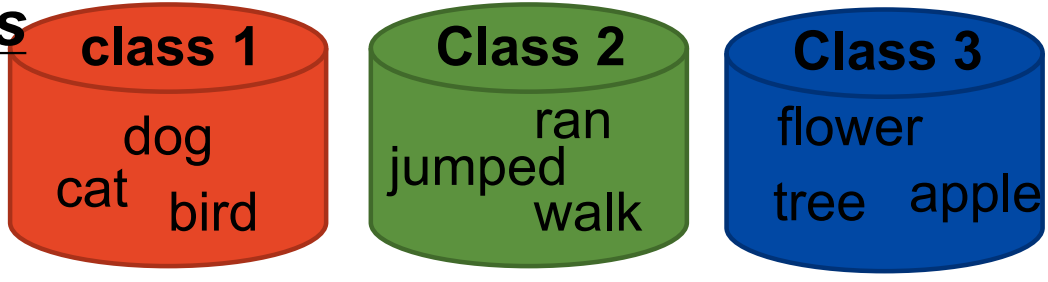
elephant = [0 0 0 0 1]

drawback:
any pair has the same distance
no info about relationship

Word Embedding



Word Class



A word can have multiple senses.

Have you paid that money to the **bank** yet ?
It is safest to deposit your money in the **bank** .

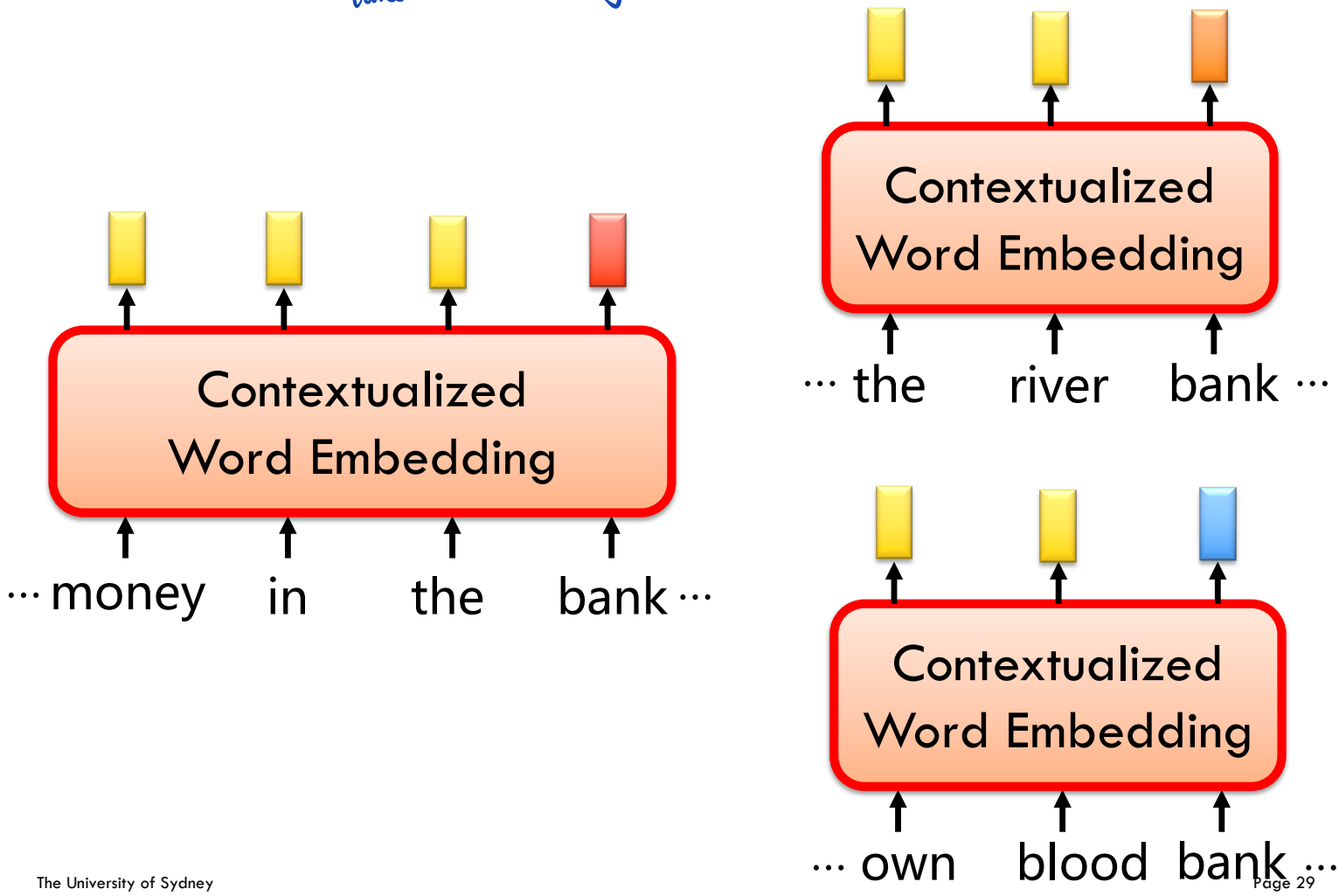
The victim was found lying dead on the river **bank** .
They stood on the river bank to fish.

The hospital has its own blood **bank**.

The third sense or not?

Contextualized Word Embedding

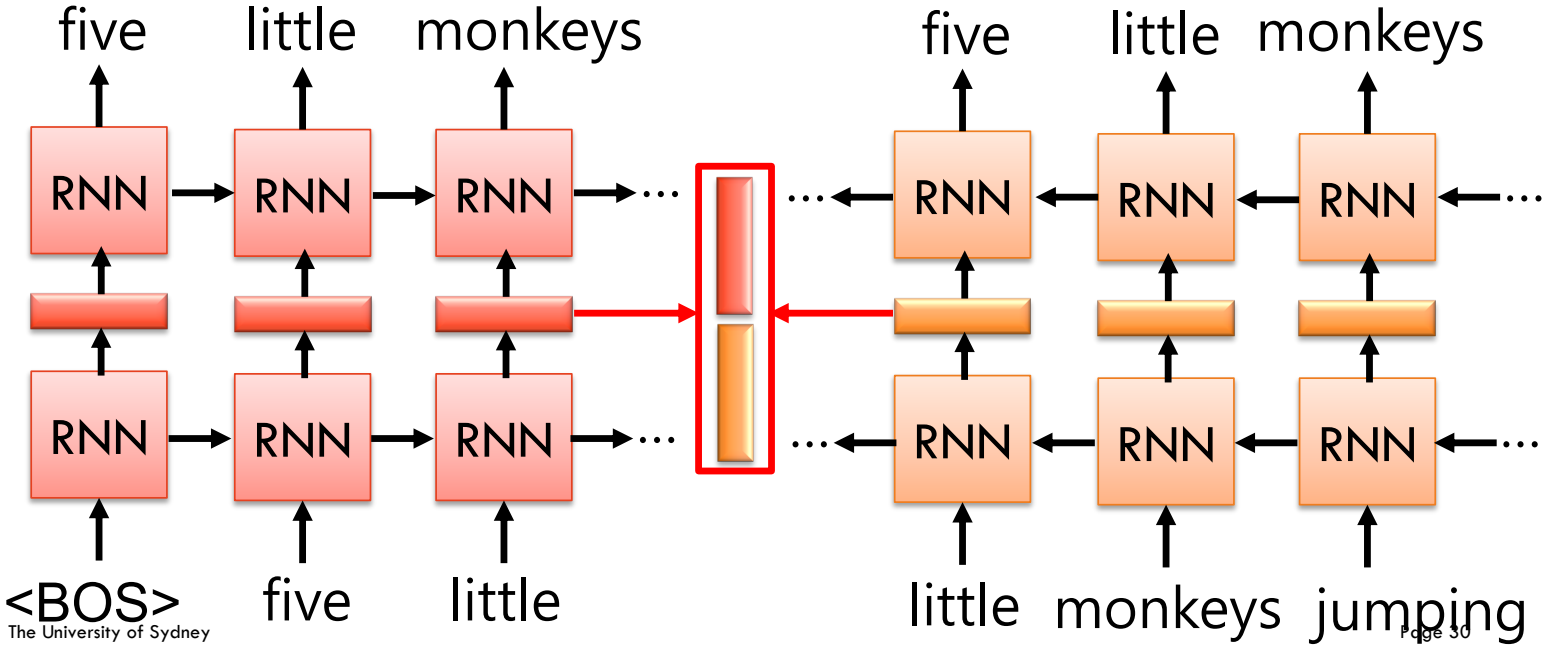
tune the embedding for different scenario



Embeddings from Language Model (ELMO)

- RNN-based language models (trained from lots of sentences)

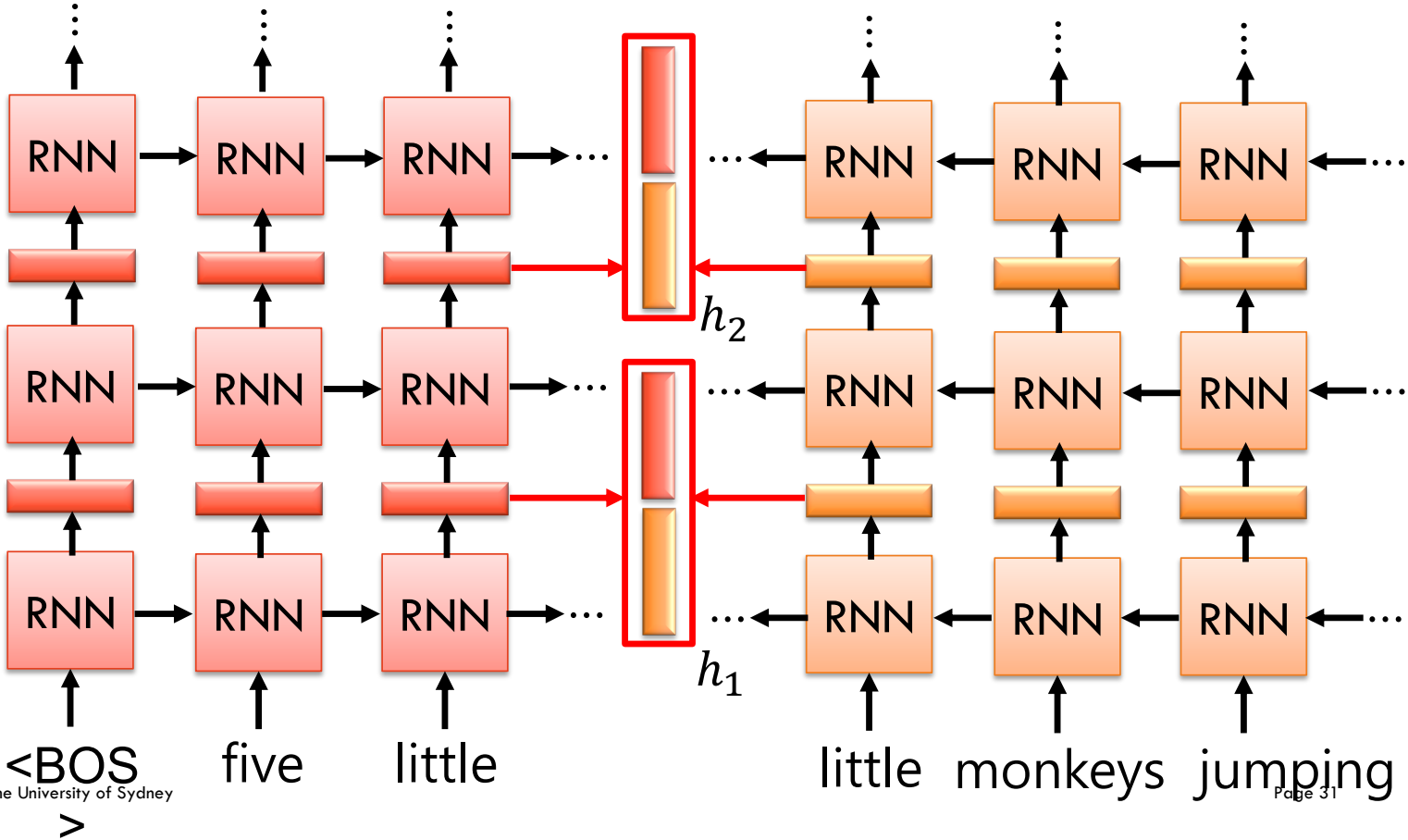
e.g. given “five little monkeys jumping on the bed”



ELMO

Each layer in deep LSTM can generate a latent representation.

Which one should we use???

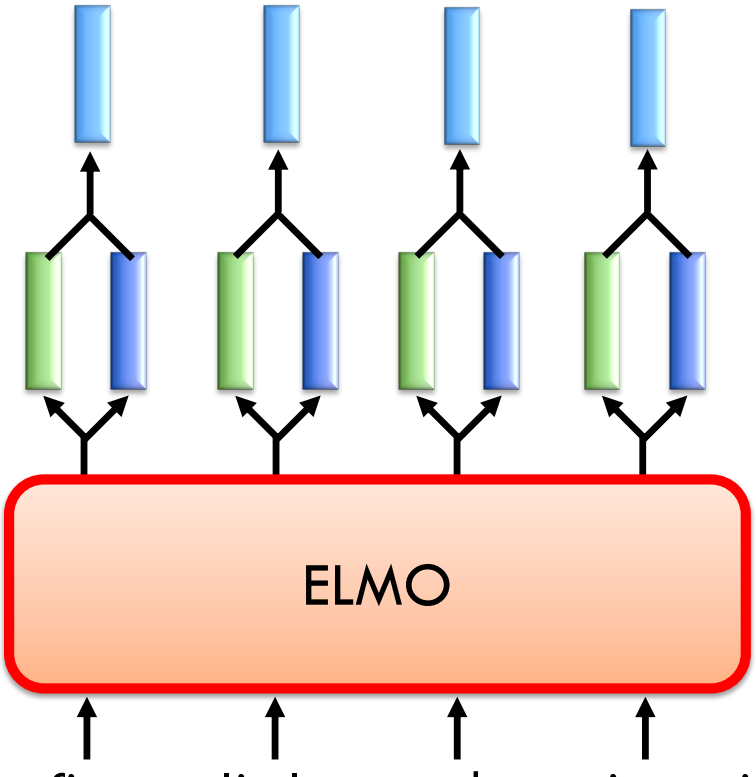


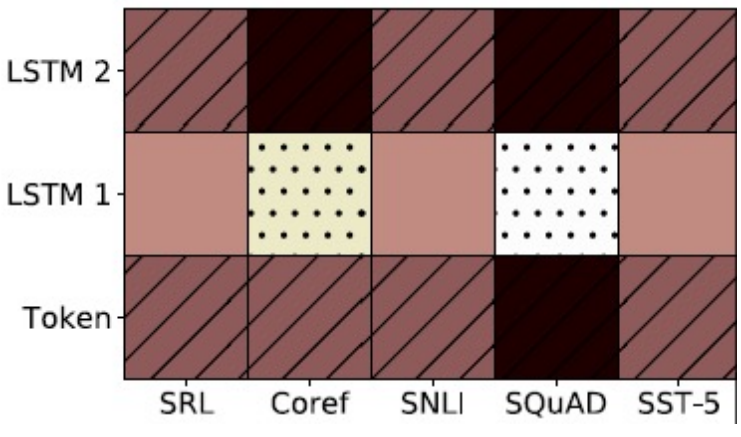
ELMO

fine tune the representation based on tasks

$= \alpha_1 + \alpha_2$

Learned with the down stream tasks

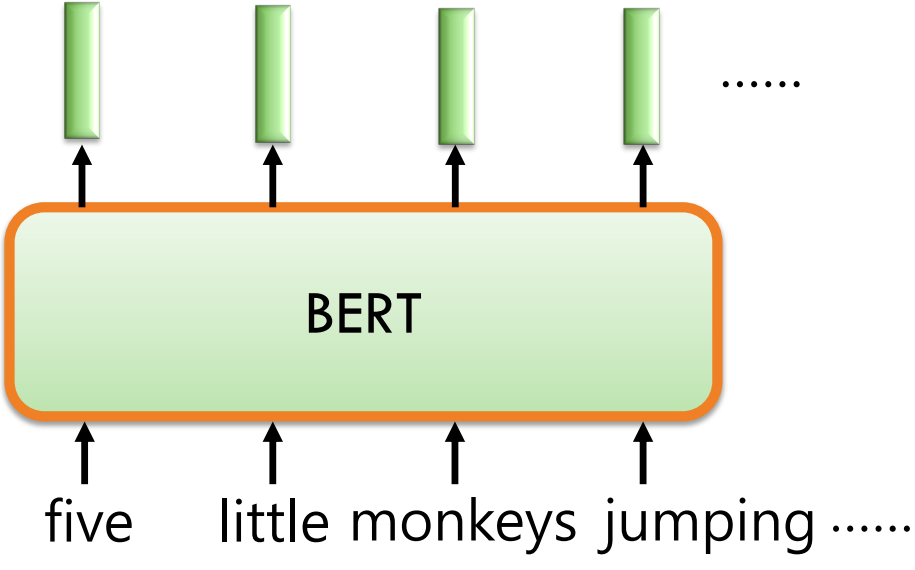


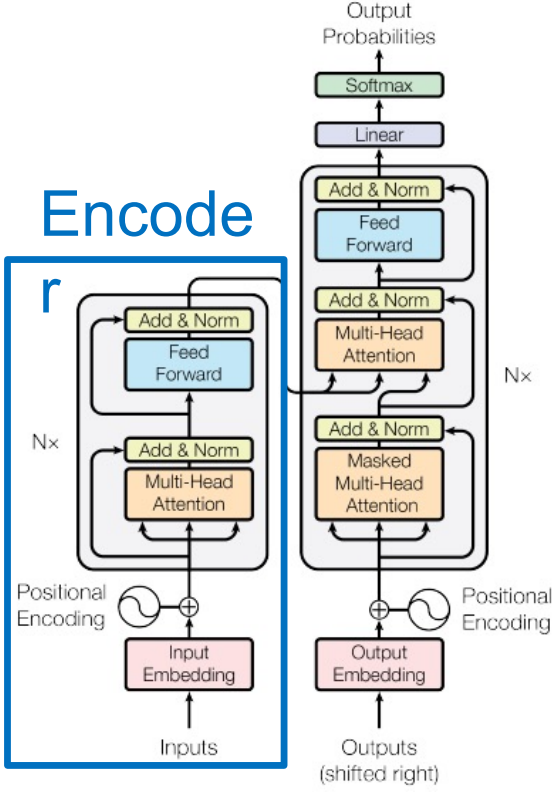


Bidirectional Encoder Representations from Transformers (BERT)

- BERT = Encoder of Transformer

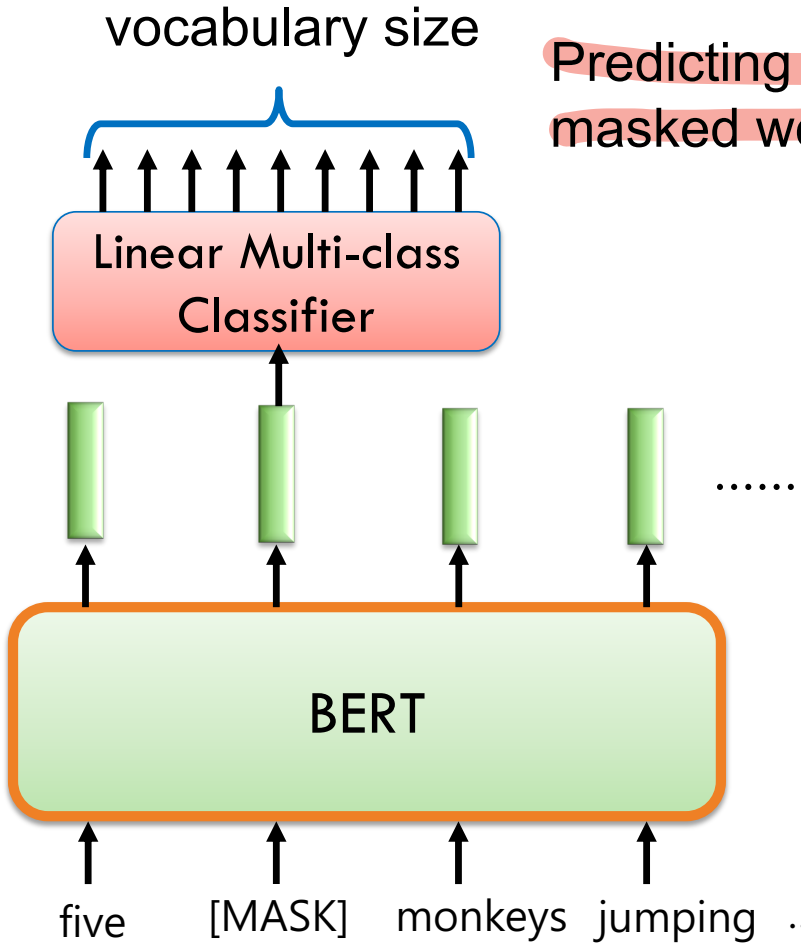
Learned from a large amount of text without annotation





Training of BERT

- Approach 1: Masked LM

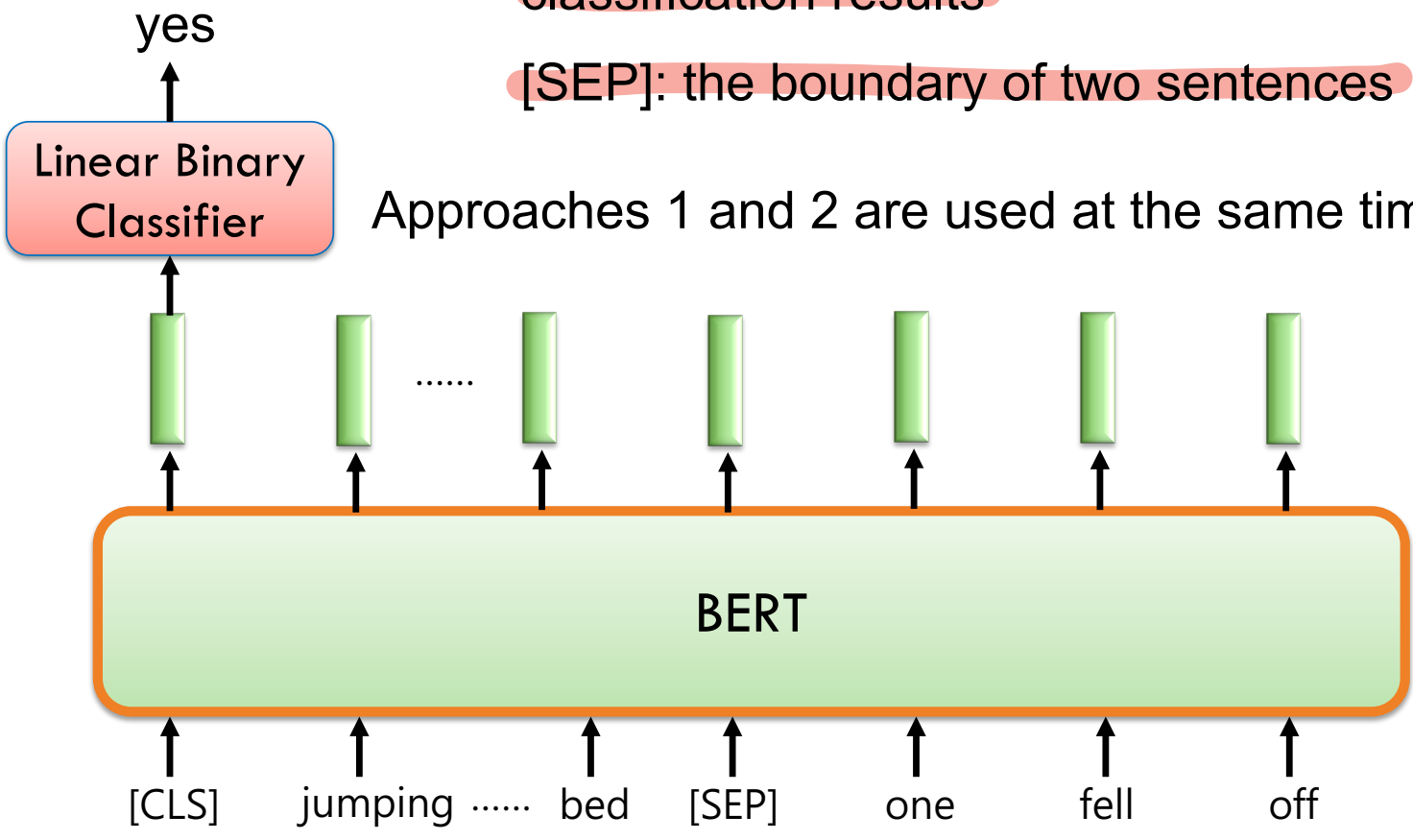


Training of BERT

Approach 2: Next Sentence Prediction

[CLS]: the position that outputs classification results

[SEP]: the boundary of two sentences

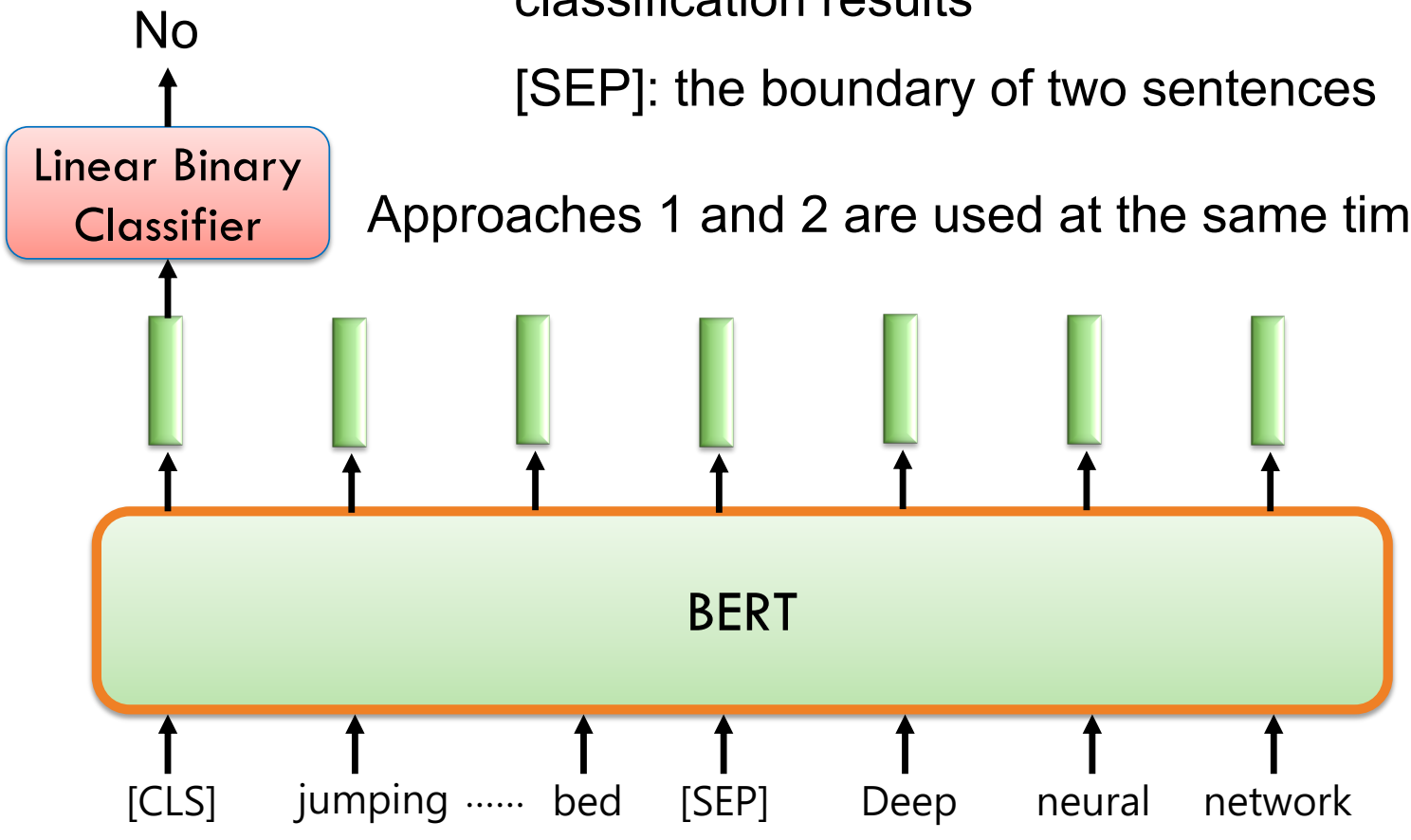


Approaches 1 and 2 are used at the same time.

Approach 2: Next Sentence Prediction

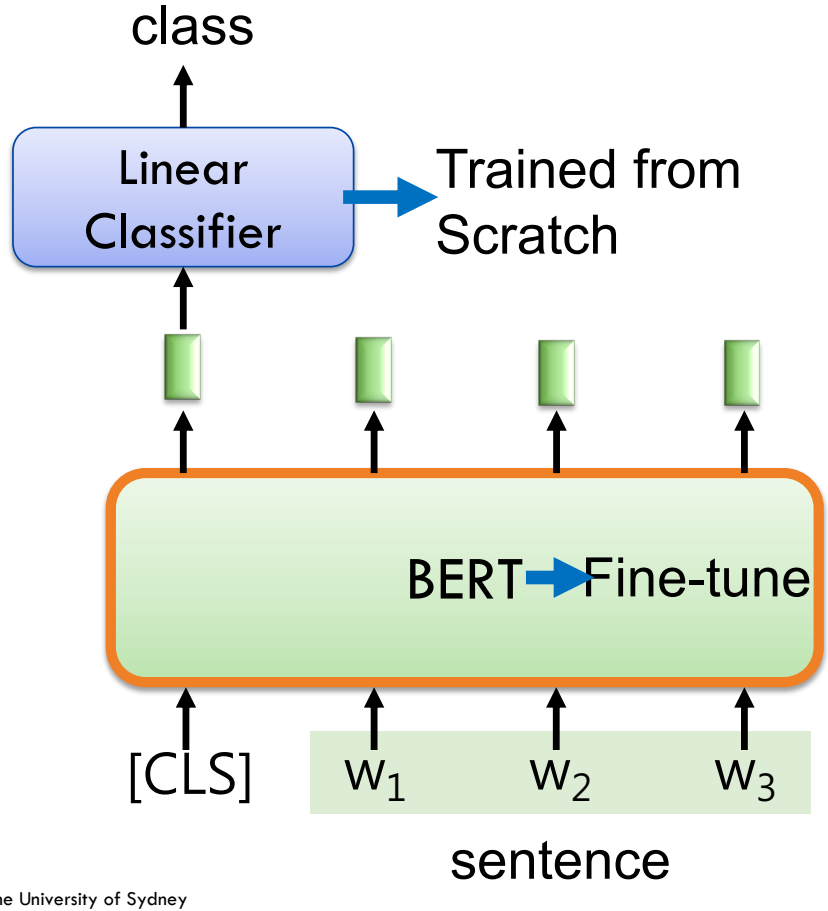
[CLS]: the position that outputs classification results

[SEP]: the boundary of two sentences



Approaches 1 and 2 are used at the same time.

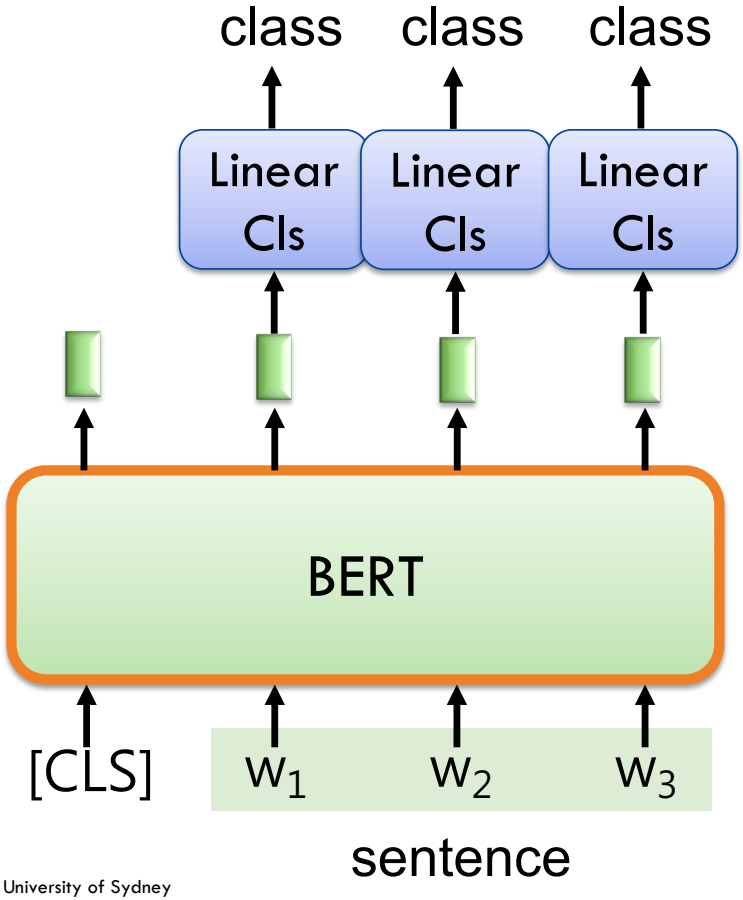
How to use BERT – Case 1



Input: single sentence, output: class

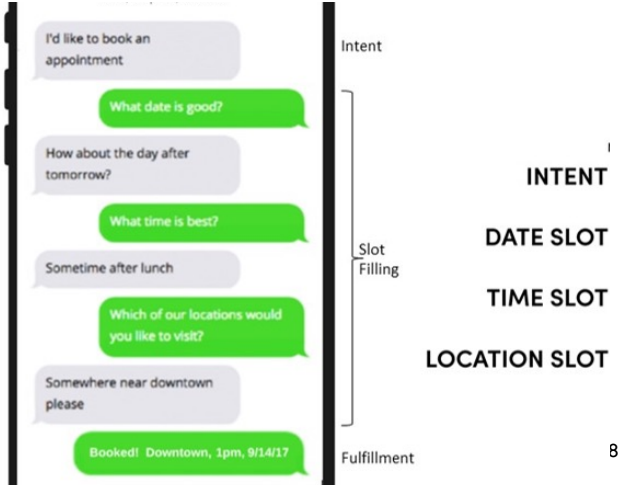
Example:
Sentiment analysis
Document Classification

How to use BERT – Case 2



Input: single sentence, output: class of each word

Example: Slot filling

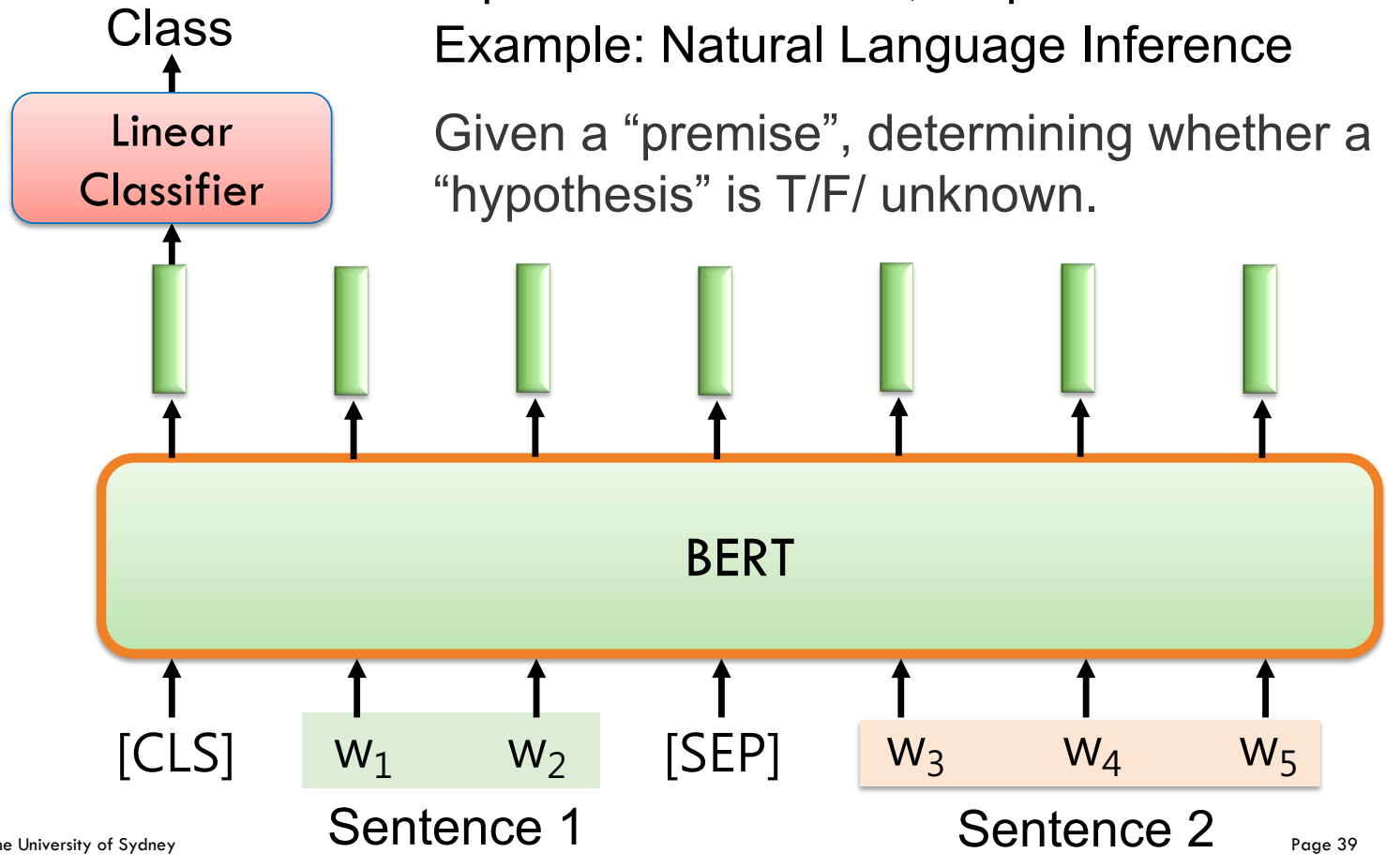


How to use BERT – Case 3

Input: two sentences, output: class

Example: Natural Language Inference

Given a "premise", determining whether a "hypothesis" is T/F/ unknown.

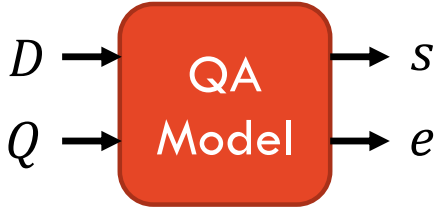


How to use BERT – Case 4

- Extraction-based Question Answering (QA) (E.g. SQuAD)

Document: $D = \{d_1, d_2, \cdots, d_N\}$

Query: $Q = \{q_1, q_2, \cdots, q_N\}$



output: two integers (s, e)

Answer: $A = \{q_s, \cdots, q_e\}$

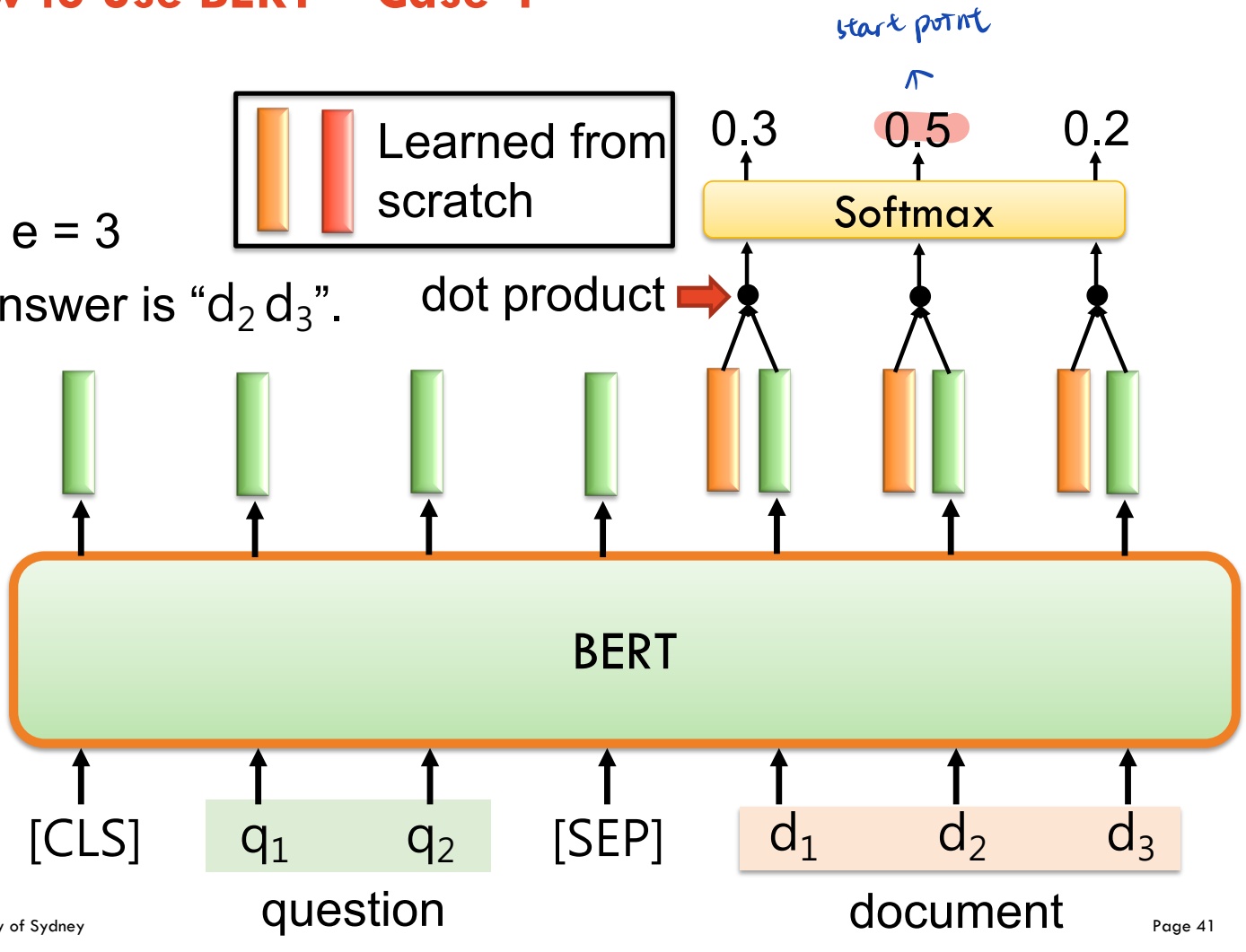
In meteorology, precipitation is any product of the condensation of 17 spheric water vapor that falls under gravity. The main forms of precipitation include drizzle, rain, sleet, snow, graupel and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals within a cloud. Short, intense periods of rain 77 atte 79 cations are called "showers".

What causes precipitation to fall?
gravity $s = 17, e = 17$

What is another main form of precipitation besides drizzle, rain, snow, sleet and hail?
graupel

Where do water droplets collide with ice crystals to form precipitation?
within a cloud $s = 77, e = 79$

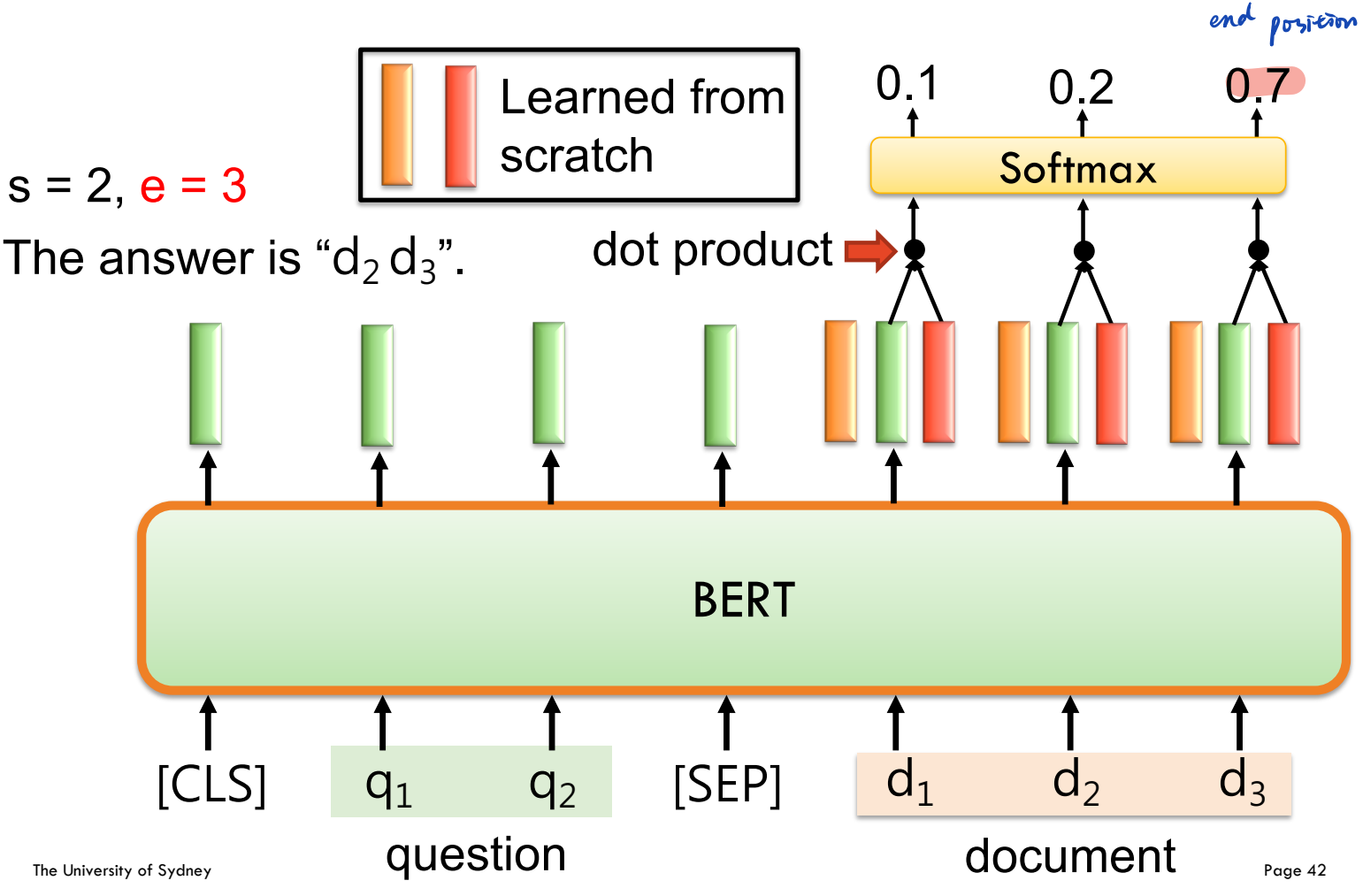
How to use BERT – Case 4



s = 2, e = 3

The answer is "$d_2\, d_3$".

How to use BERT – Case 4



What does BERT learn?

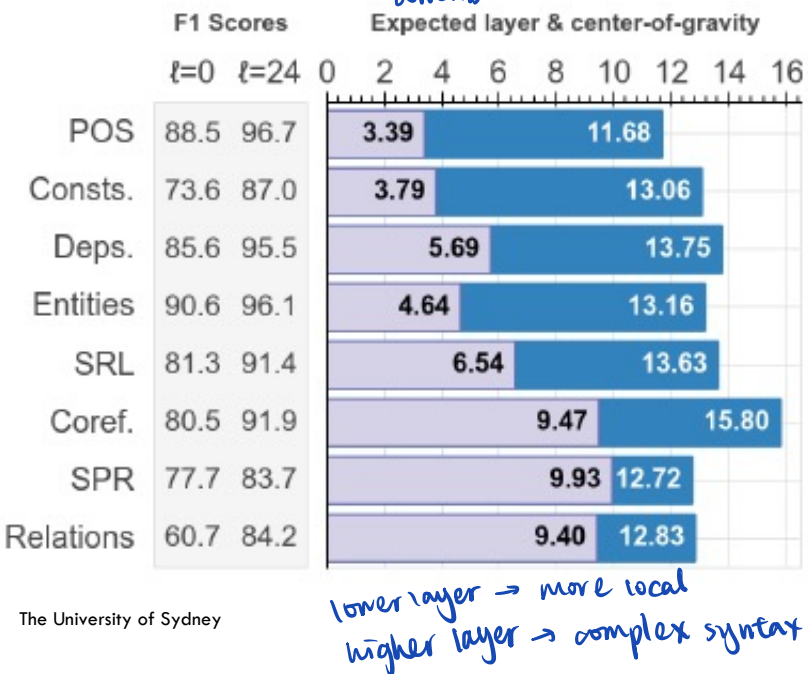
lower layers of a language model encode more local syntax while higher layers capture more complex semantics,

different layers → different feature extraction

https://arxiv.org/abs/1905.05950

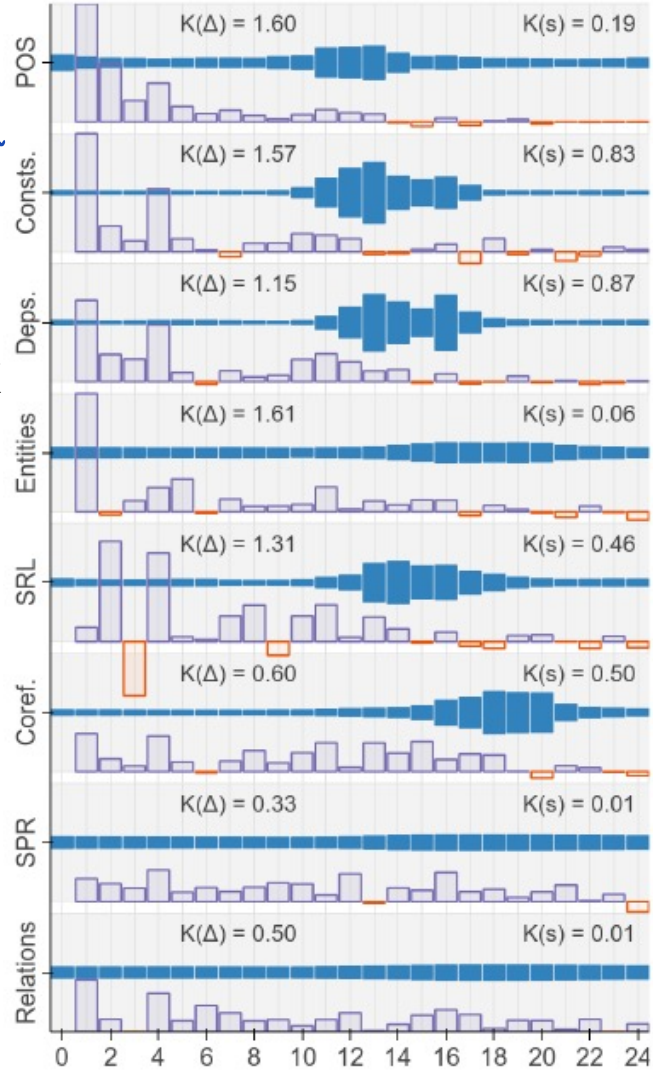
https://openreview.net/pdf?id=SJzSgnRcKX

information of which layer contribute most to the task



lower layer → more local

higher layer → complex syntax

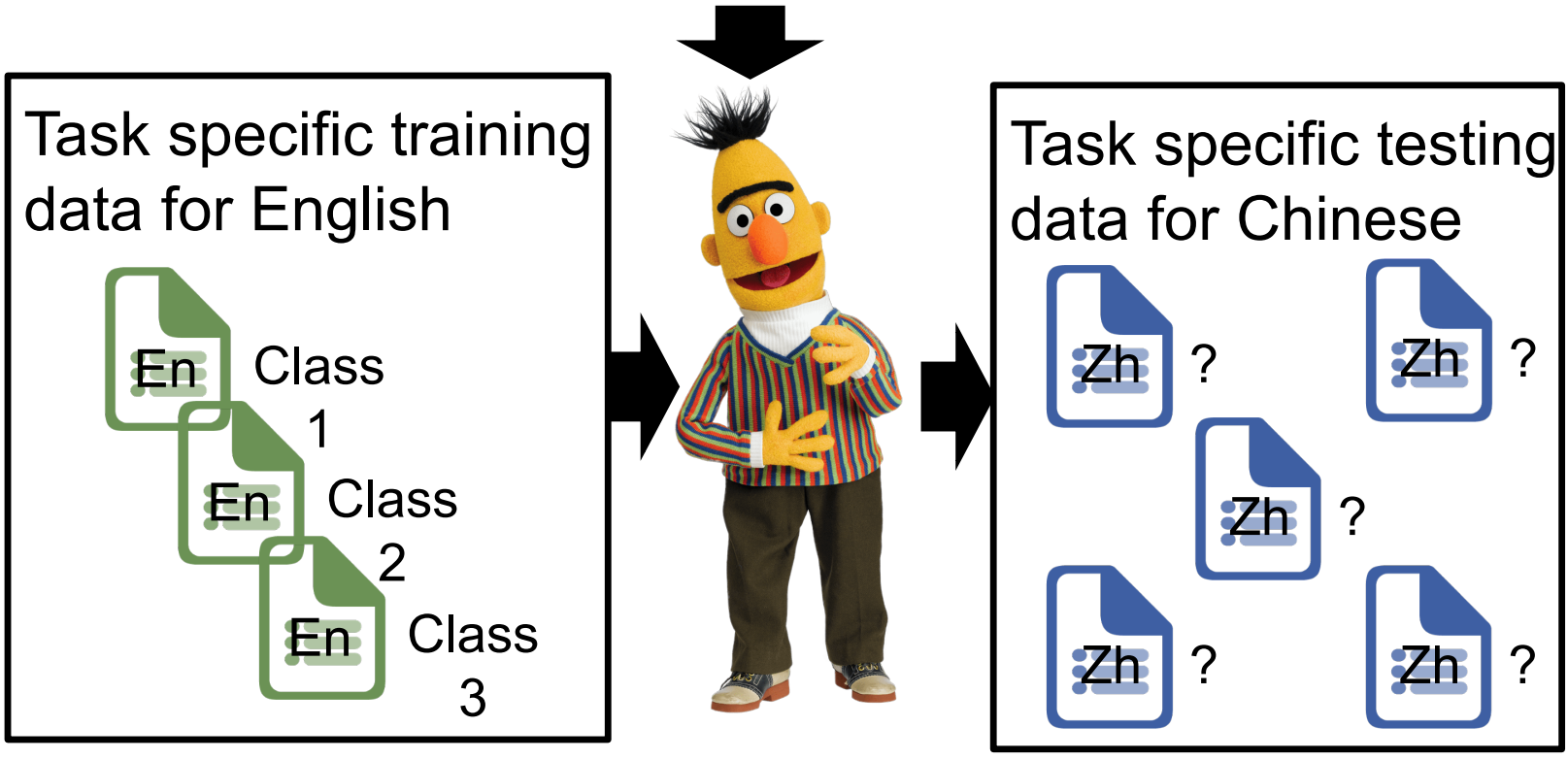


https://arxiv.org/abs/1904.09077

Multilingual BERT

transferable to downstream task

Trained on 104 languages



GPT

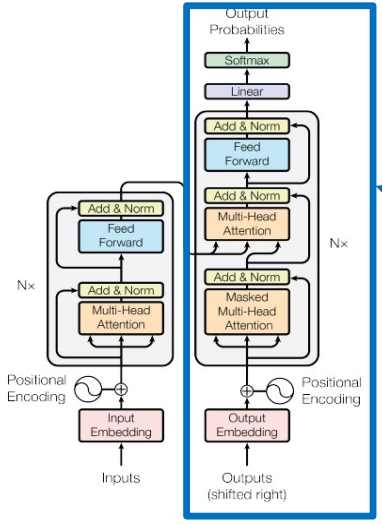
Decoder part

A **transformer** uses Encoder stack to model input, and uses Decoder stack to model output (using input information from encoder side).

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**.

But 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**.

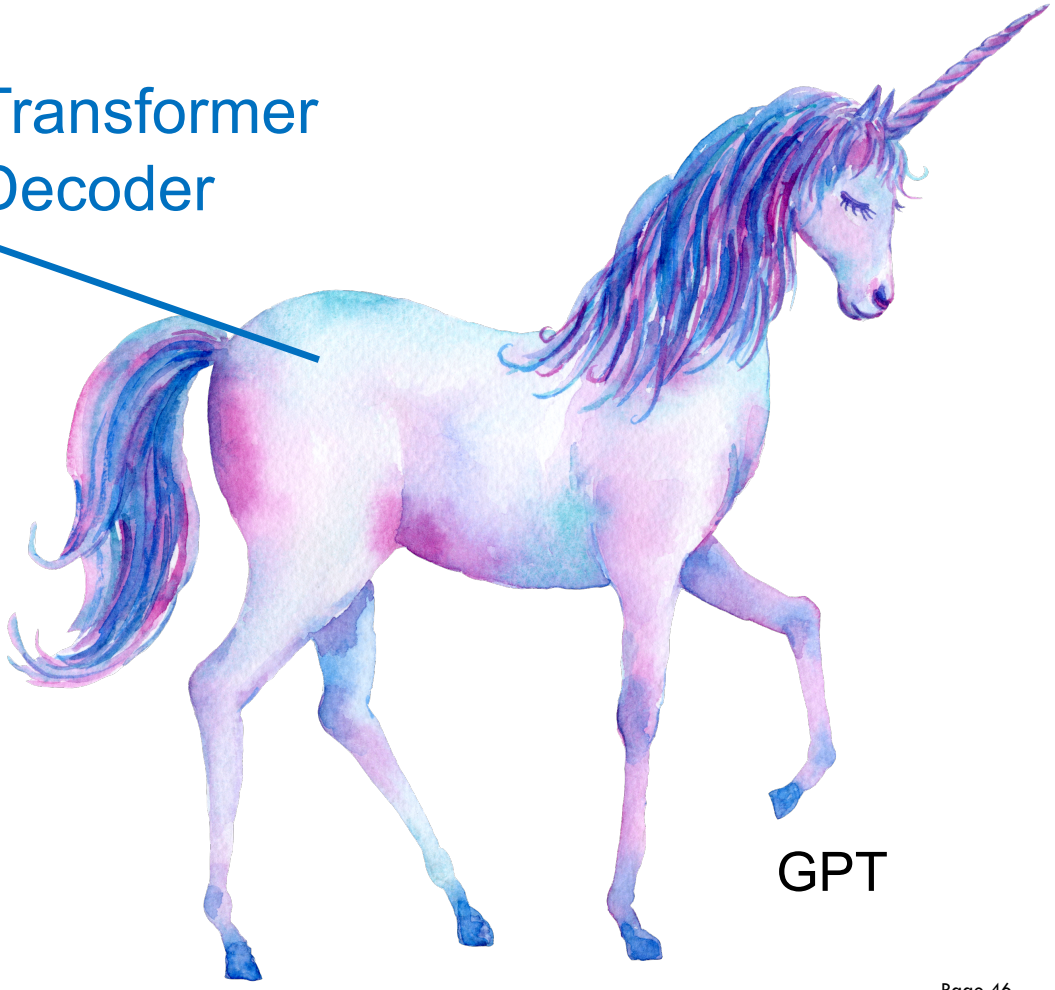
Generative Pre-Training (GPT)



Transformer Decoder

BERT (340M)

ELMO (94M)



GPT

Source of image: https://huaban.com/pins/1714071707/

The now-famous unicorn example

SYSTEM PROMPT (HUMAN-WRITTEN)

In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

MODEL COMPLETION (MACHINE-WRITTEN, 10 TRIES)

The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science.

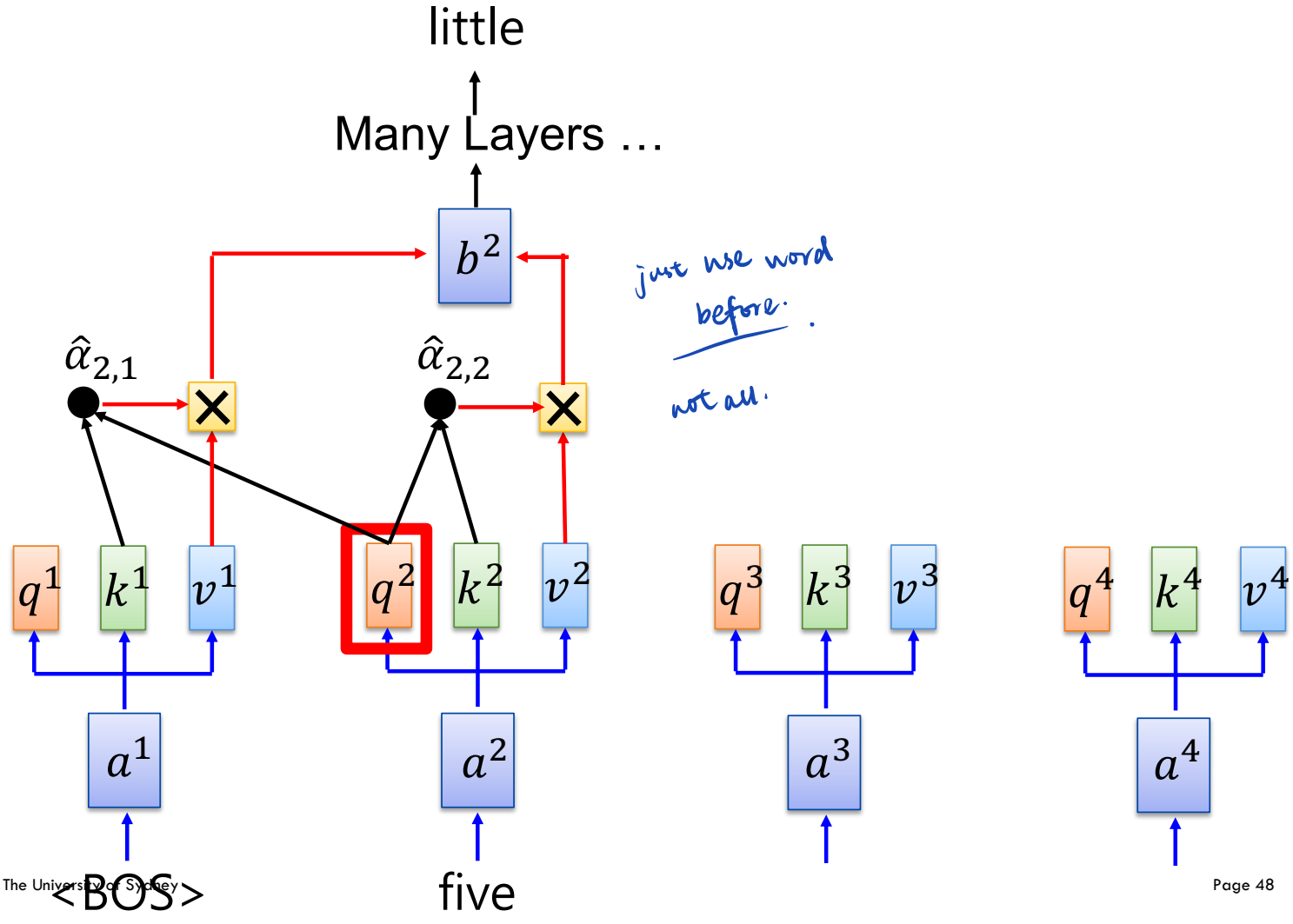
Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.

Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.

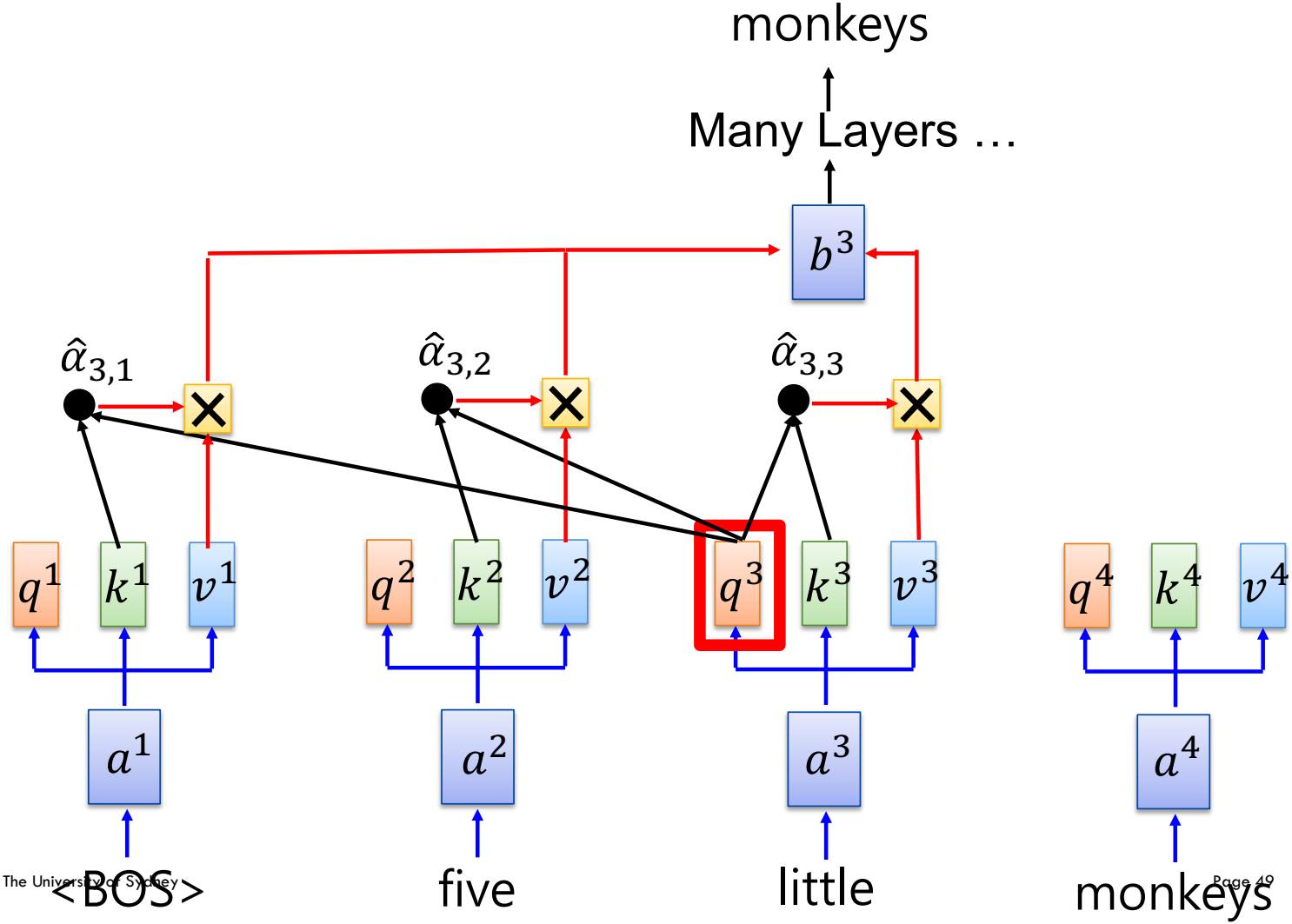
Pérez and the others then ventured further into the valley. "By the time we reached the top of one peak, the water looked blue, with some crystals on top," said Pérez.

Pérez and his friends were astonished to see the unicorn herd. These creatures could be seen from the air without having to move too much to see them – they were so close they could touch their horns.

Generative Pre-Training (GPT)



Generative Pre-Training (GPT)



Improving Language Understanding by Generative Pre-training (GPT-1):

Supervised models have two major limitations:
They need large amount of annotated data for learning a particular task which is often not easily available.
They fail to generalize for tasks other than what they have been trained for.

This paper proposed **learning a generative language model using unlabeled data** and then **fine-tuning the model** by providing examples of specific downstream tasks like classification, sentiment analysis, textual entailment etc.

Improving Language Understanding by Generative Pre-training (GPT-1):

Unsupervised Language Modelling (Pre-training): For unsupervised learning, standard language model objective was used.

maximise prob to generate next token

$$L_1(T) = \sum_i log\, P(t_i | t_{i-k}, \ldots, t_{i-1}; \theta)$$

where T was the set of tokens in unsupervised data {t_1,…,t_n}, k was size of context window, θ were the parameters of neural network trained using stochastic gradient descent.

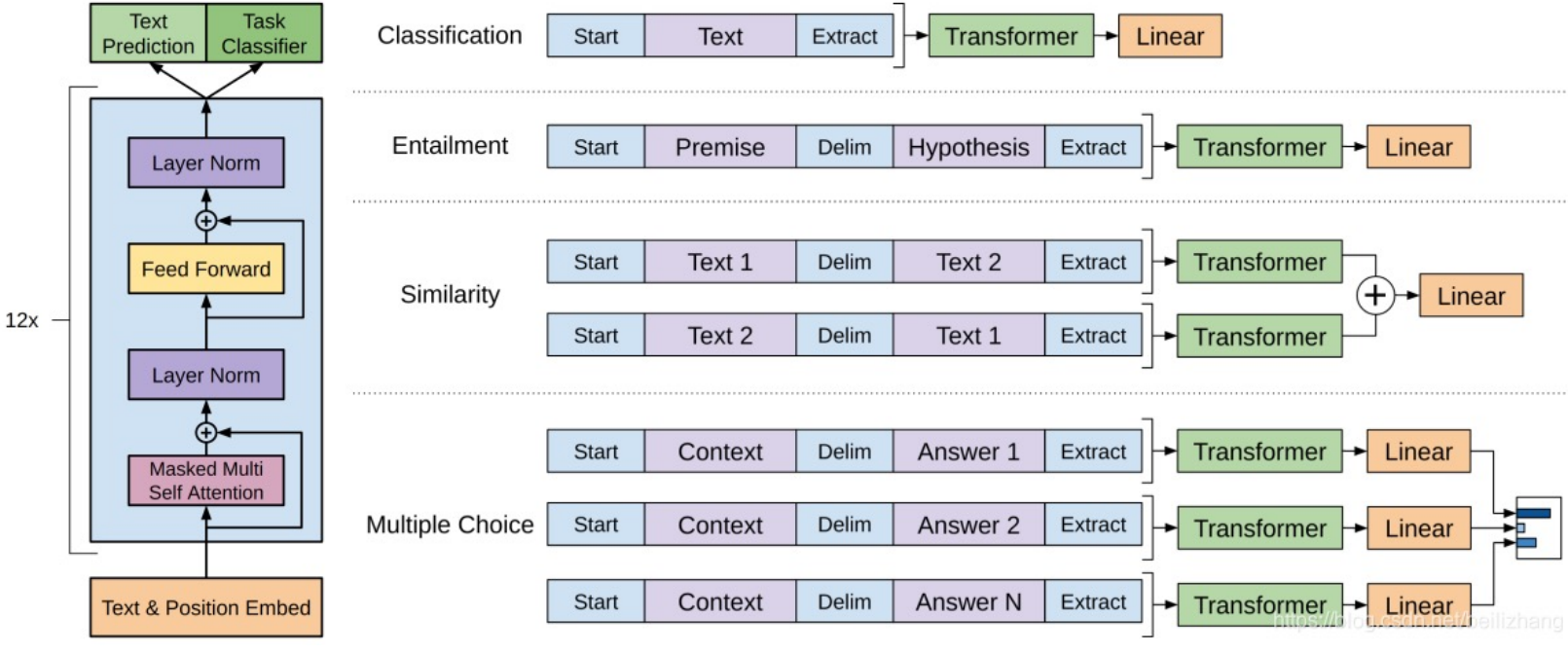
Supervised Fine-Tuning: This part aimed at maximising the likelihood of observing label y, given features or tokens x_1,…,x_n.

$$L_2(C) = \sum_{x,y} log\, P(y | x_1, \ldots, x_n)$$

where C was the labeled dataset made up of training examples.

Improving Language Understanding by Generative Pre-training (GPT-1):

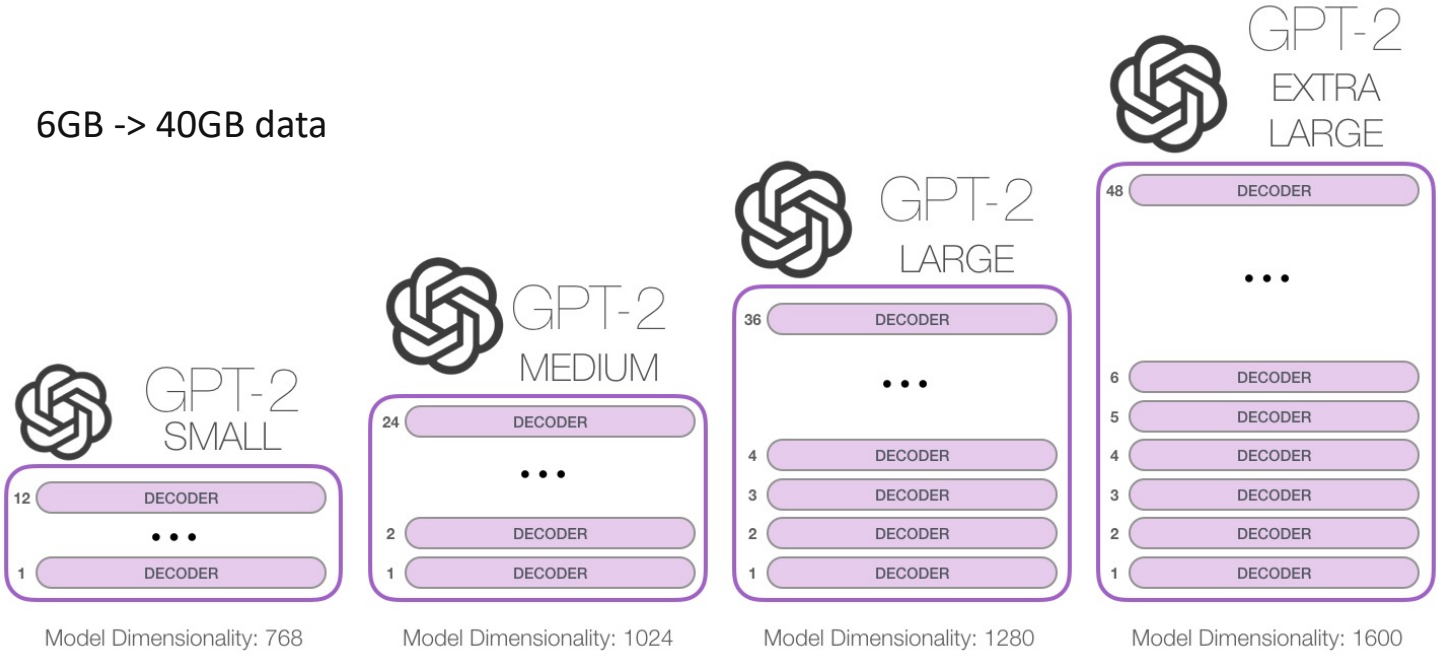
Task Specific Input Transformations: In order to make minimal changes to the architecture of the model during fine tuning, inputs to the specific downstream tasks were transformed into ordered sequences.



Language Models are unsupervised multitask learners (GPT-2)

The developments in GPT-2 model were mostly in terms of using **a larger dataset** and **adding more parameters** to the model to learn even stronger language model.

6GB -> 40GB data

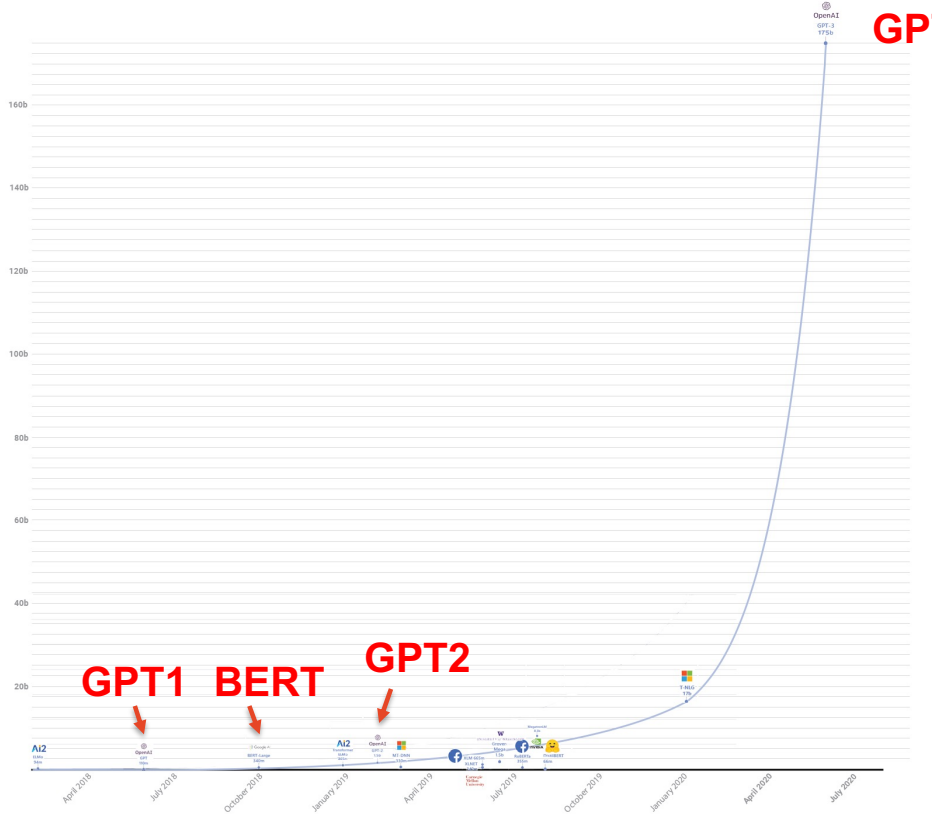


#para 117M 345M 762M 1542M

Language models are few shot learners (GPT-3):

Same architecture as GPT and GPT-2, vanilla Transformer with small upgrades accumulated over time.

#parameters



45TB data
285,000 CPUs
10,000 GPUs
$12,000,000

GPT-3

Figure 1.1: Language model meta-learning. During unsupervised pre-training, a language model develops a broad set of skills and pattern recognition abilities. It then uses these abilities at inference time to rapidly adapt to or recognize the desired task. We use the term "in-context learning" to describe the inner loop of this process, which occurs within the forward-pass upon each sequence. The sequences in this diagram are not intended to be representative of the data a model would see during pre-training, but are intended to show that there are sometimes repeated sub-tasks embedded within a single sequence.

GPT-3

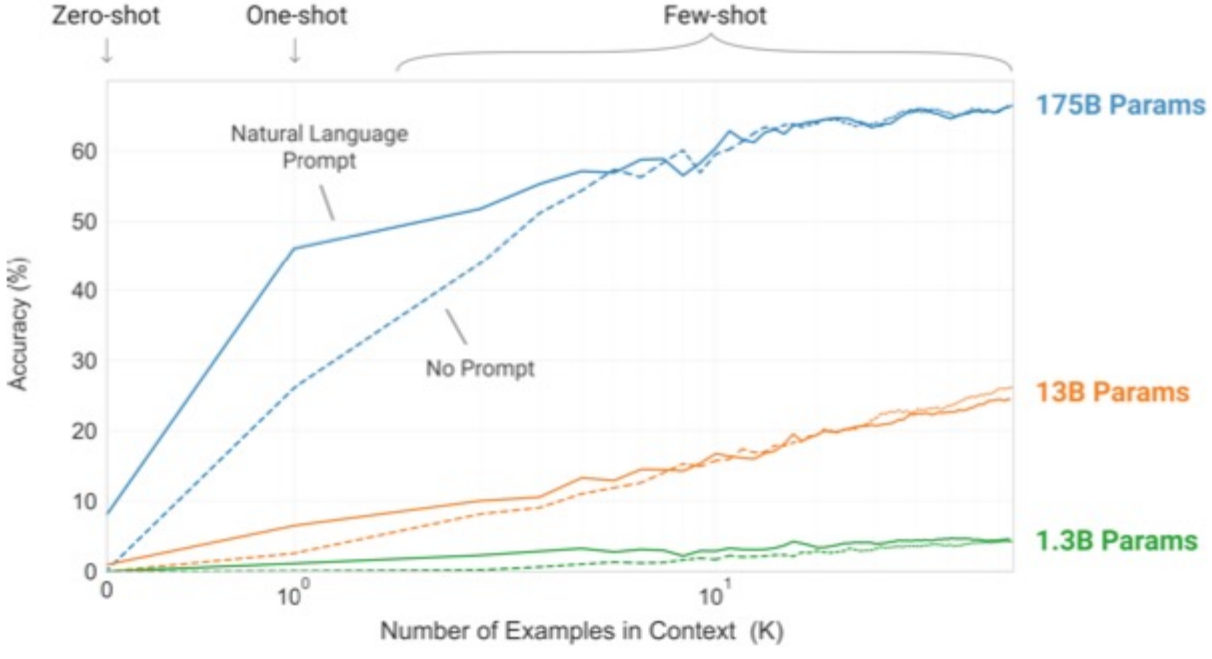


Figure 1.2: Larger models make increasingly efficient use of in-context information. We show in-context learning performance on a simple task requiring the model to remove random symbols from a word, both with and without a natural language task description (see Sec. 3.9.2). The steeper "in-context learning curves" for large models demonstrate improved ability to learn a task from contextual information. We see qualitatively similar behavior across a wide range of tasks.

GPT-3

Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-4}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-4}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	1.0×10^{-4}
GPT-3 175B or "GPT-3"	175.0B	96	12288	96	128	3.2M	0.6×10^{-4}

Table 2.1: Sizes, architectures, and learning hyper-parameters (batch size in tokens and learning rate) of the models which we trained. All models were trained for a total of 300 billion tokens.

Thank you !

Vision Transformer

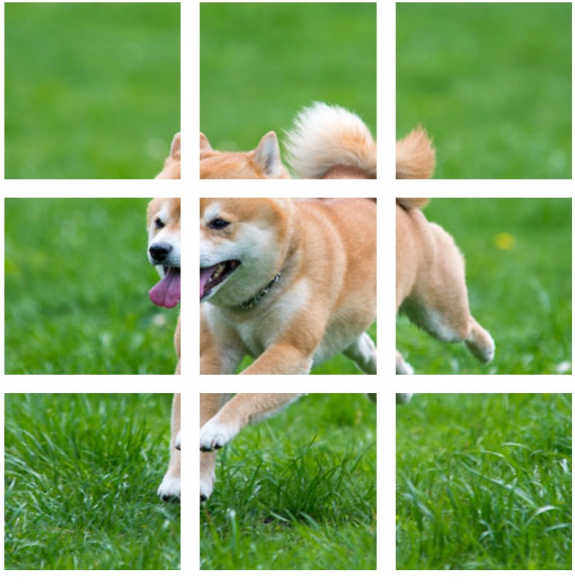
Image Classification

- CNNs, e.g., ResNet, were the best solutions to image classification.
- Vision Transformer (ViT) [1] beats CNNs (by a small margin), if the dataset for pretraining is sufficiently large (at least 100 million images).
- ViT is based on Transformer (for NLP) [2].

Reference

1. Dosovitskiy et al. An image is worth 16×16 words: transformers for image recognition at scale. In *ICLR*, 2021.
2. Vaswani et al. Attention Is All You Need. In *NIPS*, 2017.

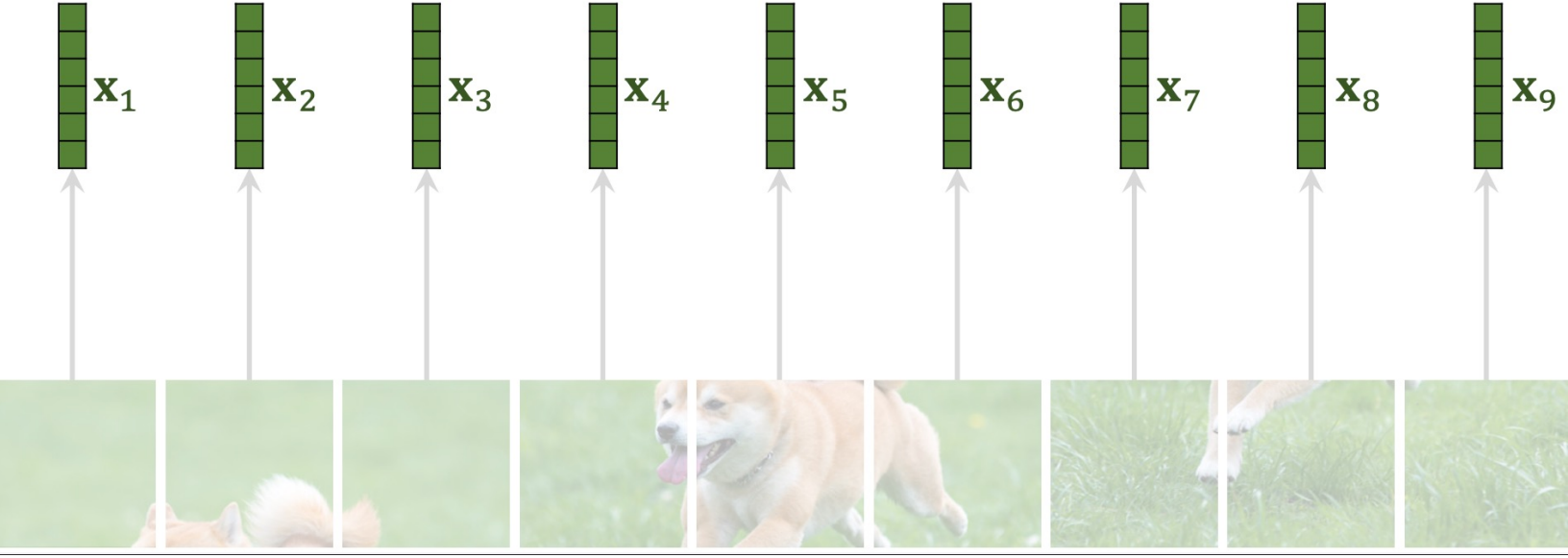
Split Image into Patches

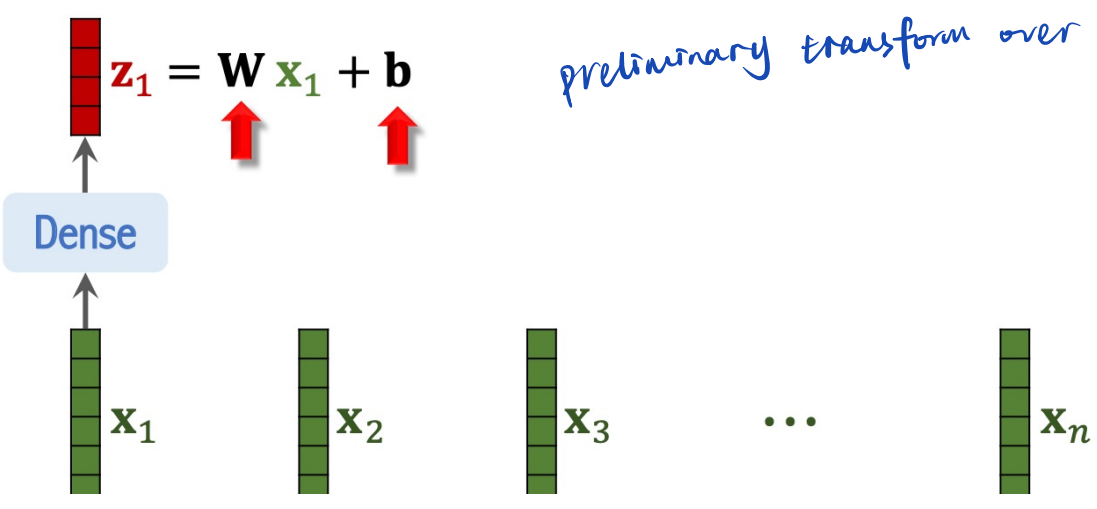


- Here, the patches do not overlap.
- The patches can overlap.
- User specifies:
  - patch size, e.g., 16×16;
  - stride, e.g., 16×16.

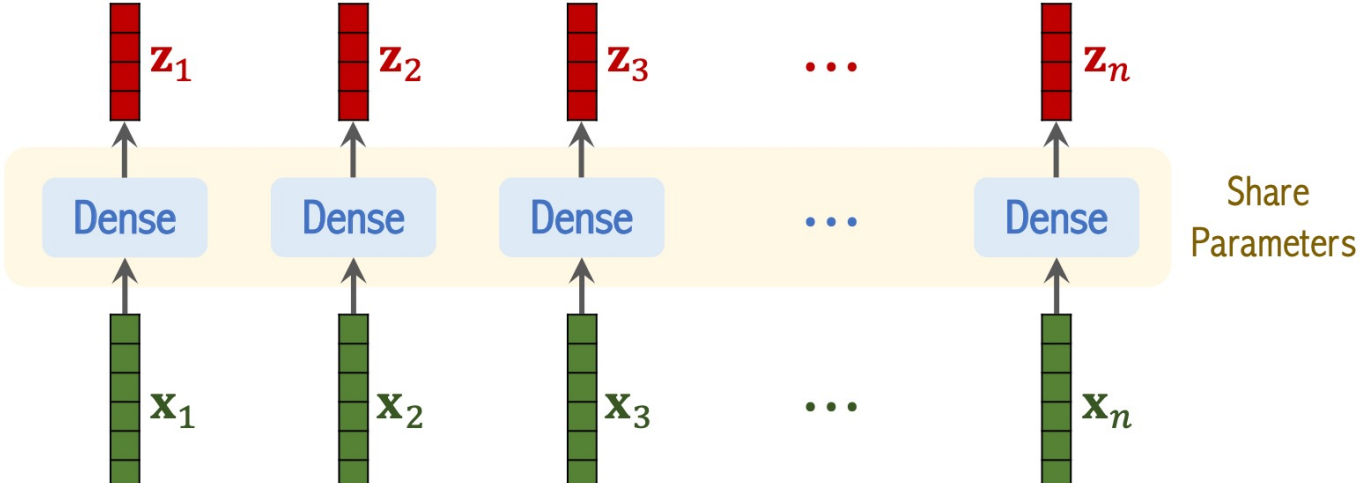
Vectorization

If the patches are $d_1 \times d_2 \times d_3$ tensors, then the vectors are $d_1 d_2 d_3 \times 1$.

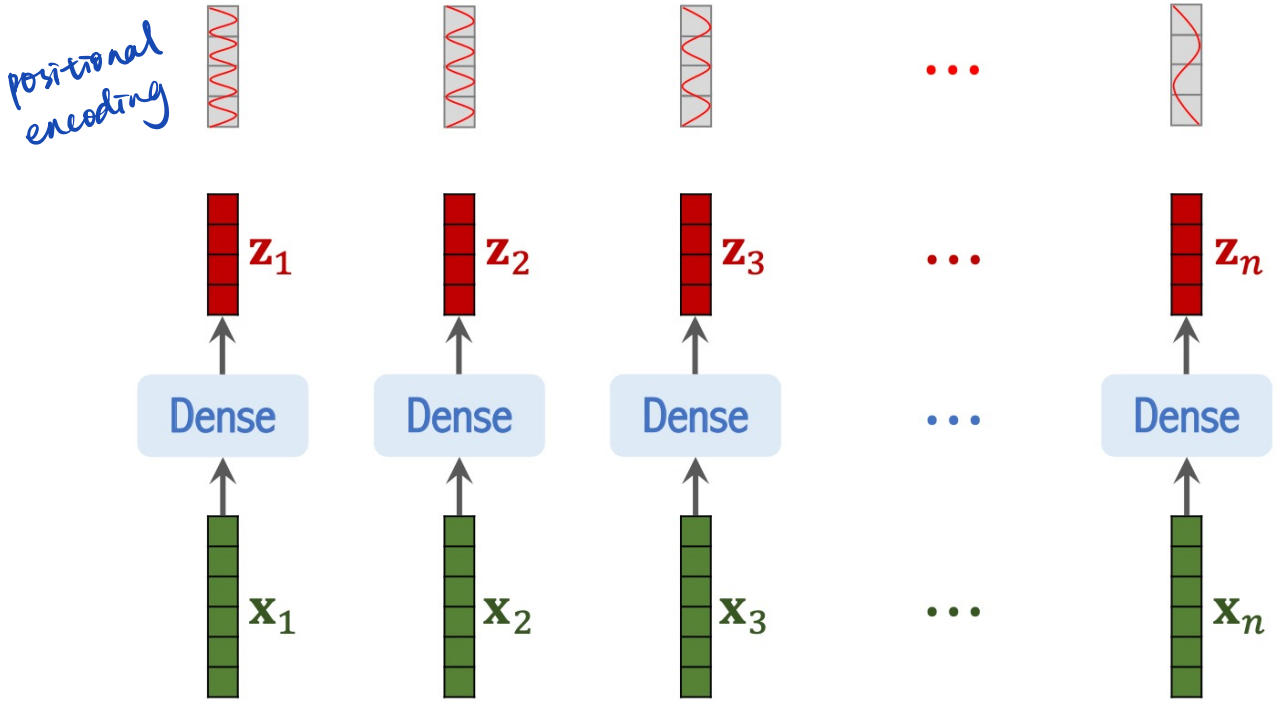




preliminary transform over pixel


$\mathbf{z}_1$
$\mathbf{z}_2$
$\mathbf{z}_3$
$\mathbf{z}_n$
Dense
Dense
Dense
Dense
Share Parameters
$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_3$
$\mathbf{x}_n$

Add positional encoding vectors to $\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_n$.



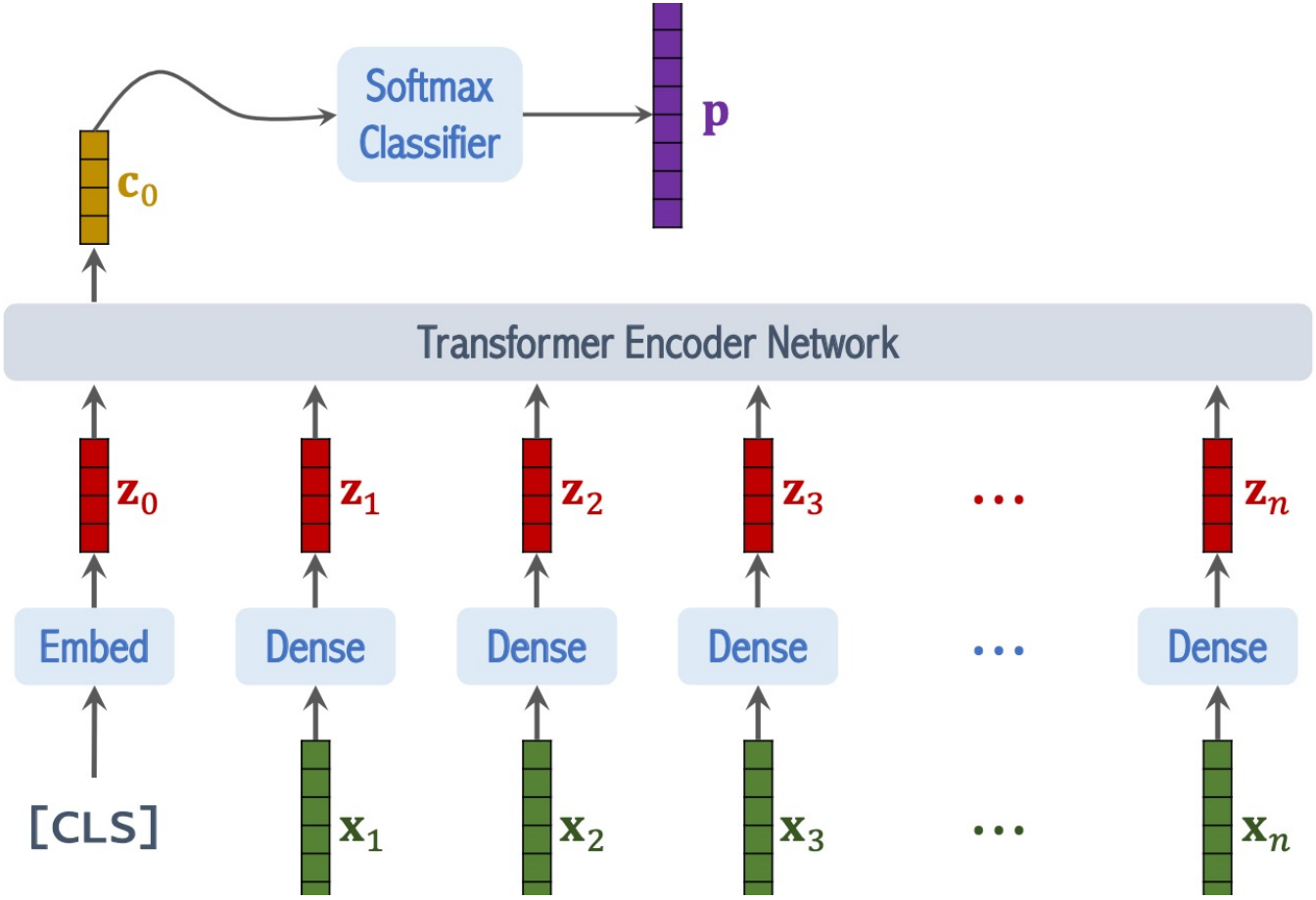

Softmax Classifier
p
c_0
Transformer Encoder Network
z_0
z_1
z_2
z_3
z_n
Embed
Dense
Dense
Dense
Dense
[CLS]
x_1
x_2
x_3
x_n

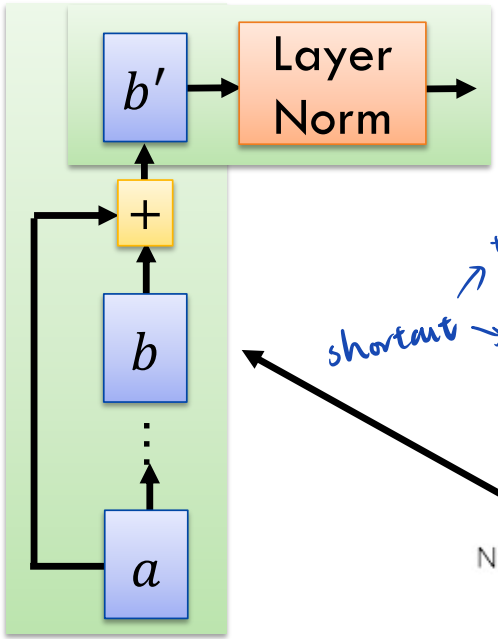
Image Classification Accuracies

- Pretrain the model on Dataset A, fine-tune the model on Dataset B, and evaluate the model on Dataset B.

- Pretrained on ImageNet (small), ViT is slightly worse than ResNet.
- Pretrained on ImageNet-21K (medium), ViT is comparable to ResNet.
- Pretrained on JFT (large), ViT is slightly better than ResNet.

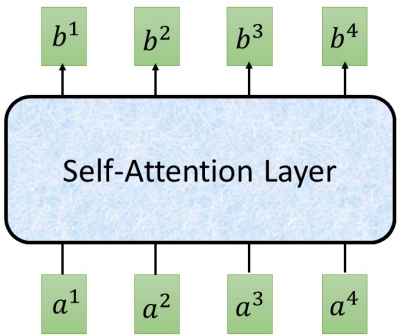
	# of Images	# of Classes
ImageNet (Small)	1.3 Million	1 Thousand
ImageNet-21K (Medium)	14 Million	21 Thousand
JFT (Big)	300 Million	18 Thousand

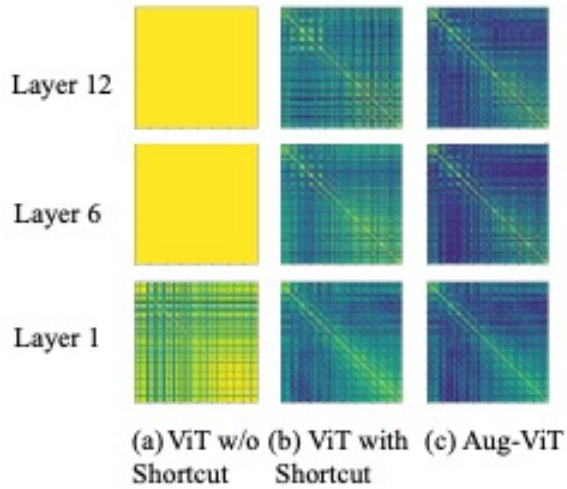
Augmented Shortcuts for Vision Transformers [NeurIPS 2021]



typical : identical mapping

shortcut → others = more shortcuts balanced/stacked





deeper nn, difference between feature lose

CMT: Convolutional Neural Networks Meet Vision Transformers [CVPR 2022]

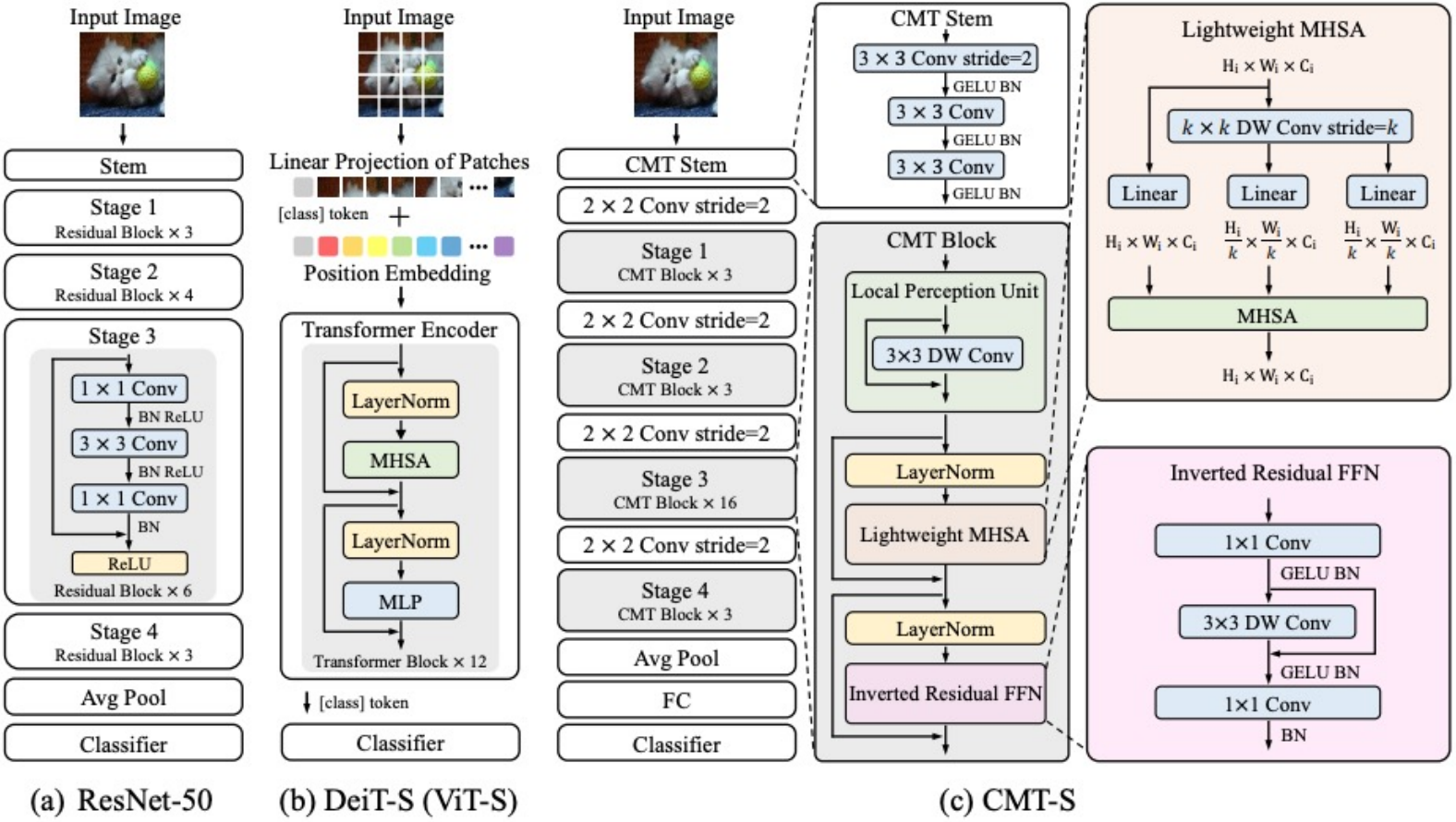


Figure 2: **Example CMT architecture for ImageNet.** (a) ResNet-50 model [15] as a reference. (b) DeiT-S [51] (ViT-S [10]) architecture. MHSA denotes the multi-head self-attention module. MLP is the multi-layer perceptron. (c) Our proposed CMT-S, described in Sec. 3. Table 1 shows more details and other variants.