Pretrained models with self supervised learning: Foundation models and LLM

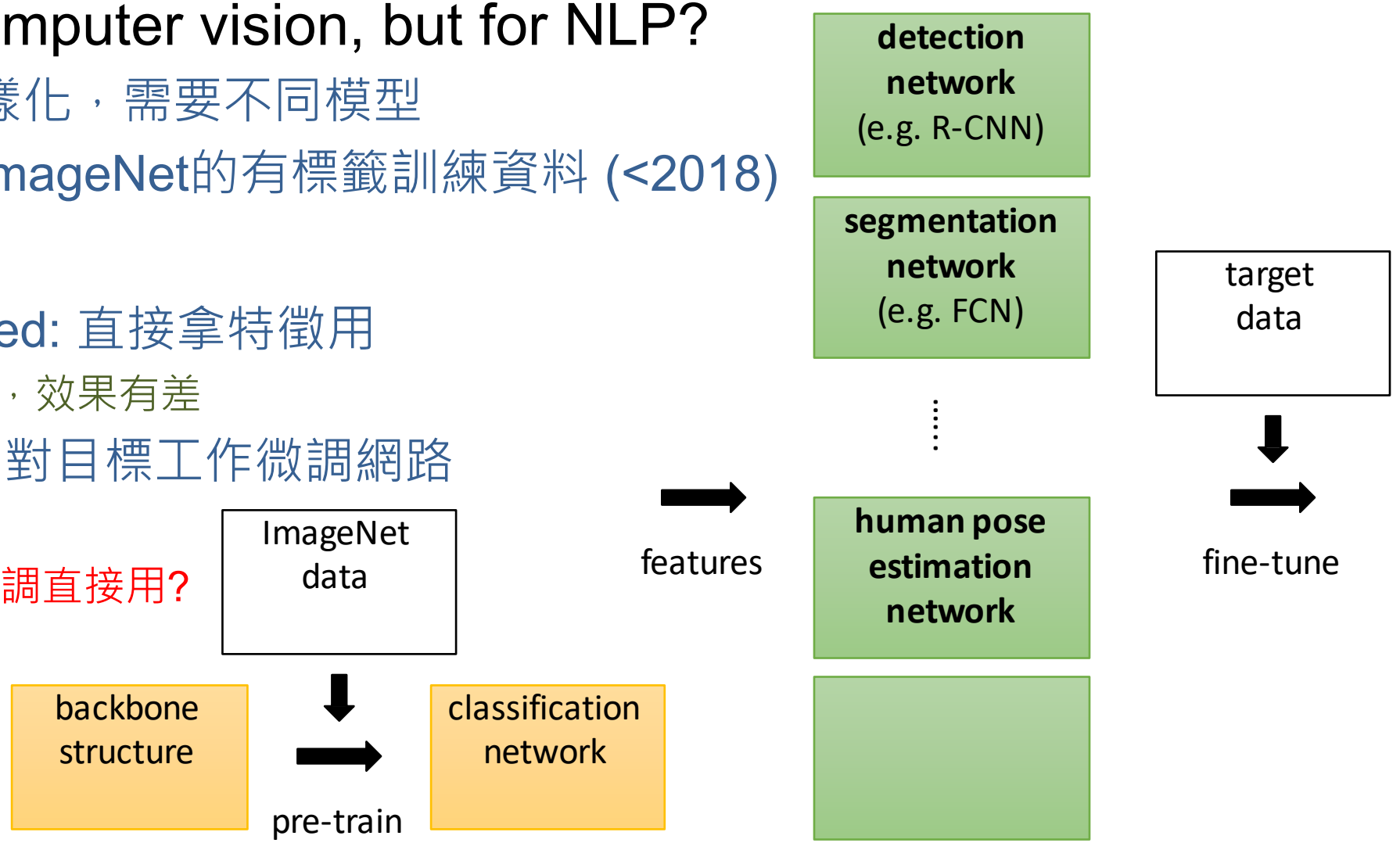
大數據，大模型
大力出奇蹟

Outline

- Large scale pretraining on NLP
  - 大力出奇蹟: from transformer to BERT and GPT
  - Llama
  - Scaling laws
- Self supervised learning
  - Contrast learning
  - Prompt based computer vision

Pretrained Model for Computer Vision

- Popular in computer vision, but for NLP?
  - NLP任務多樣化，需要不同模型
  - NLP缺少像ImageNet的有標籤訓練資料 (<2018)
- Two types
  - Feature based: 直接拿特徵用
    - 對不同task，效果有差
  - Fine-tuning: 對目標工作微調網路
    - 效果較好，
    - 可以不用微調直接用?



Transformers, mid-2017

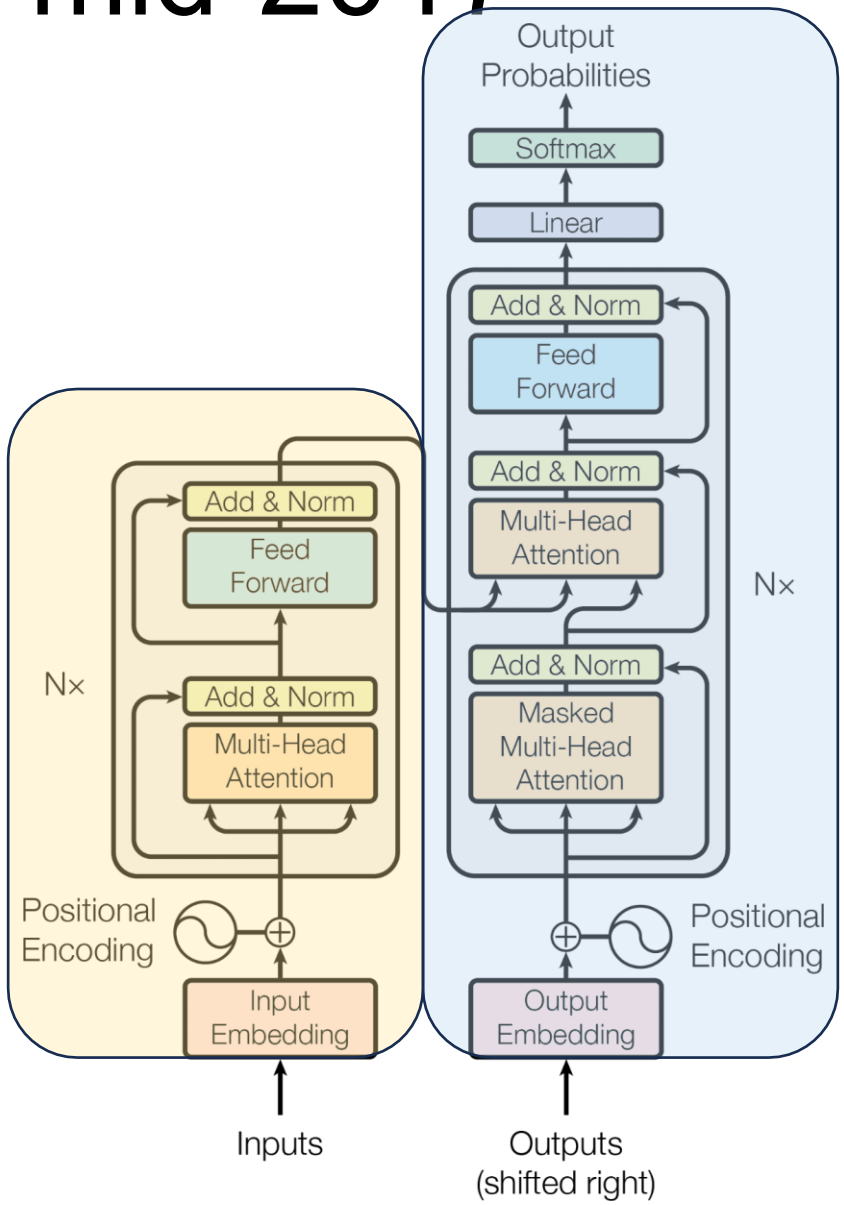
Input – input tokens
Output – hidden states

Model can see all timesteps

Does not usually output tokens, so no inherent auto-regressivity

Can also be adapted to generate tokens *by appending a module that maps hidden state dimensionality to vocab size*

Representation



Input – output tokens and hidden states*
Output – output tokens

Model can only see previous timesteps

Model is auto-regressive with previous timesteps' outputs

Can also be adapted to generate hidden states *by looking before token outputs*

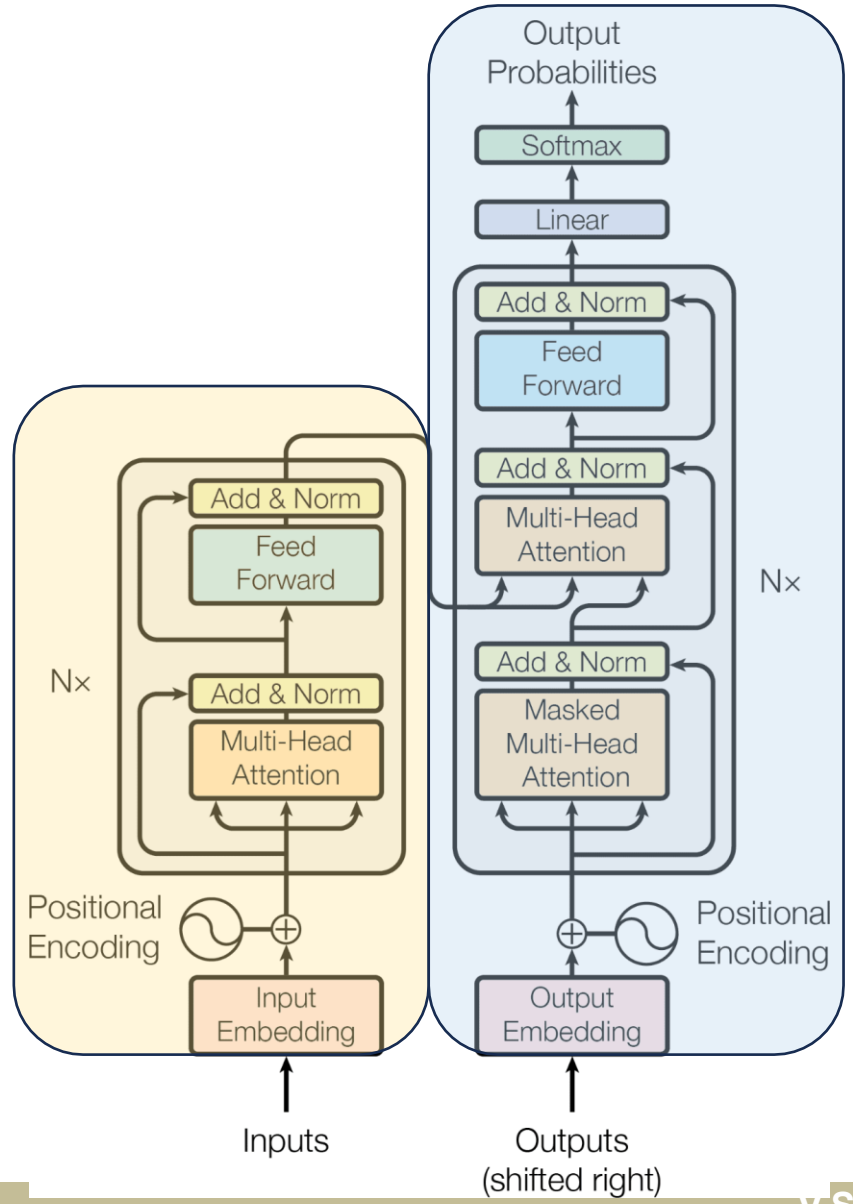
Generation

2018 – The Inception of the LLM Era

- Can we leverage large amounts of unlabeled data to pretrain an LM that understands general patterns?
  - aka. Pretrained model for all tasks

BERT
Oct 2018

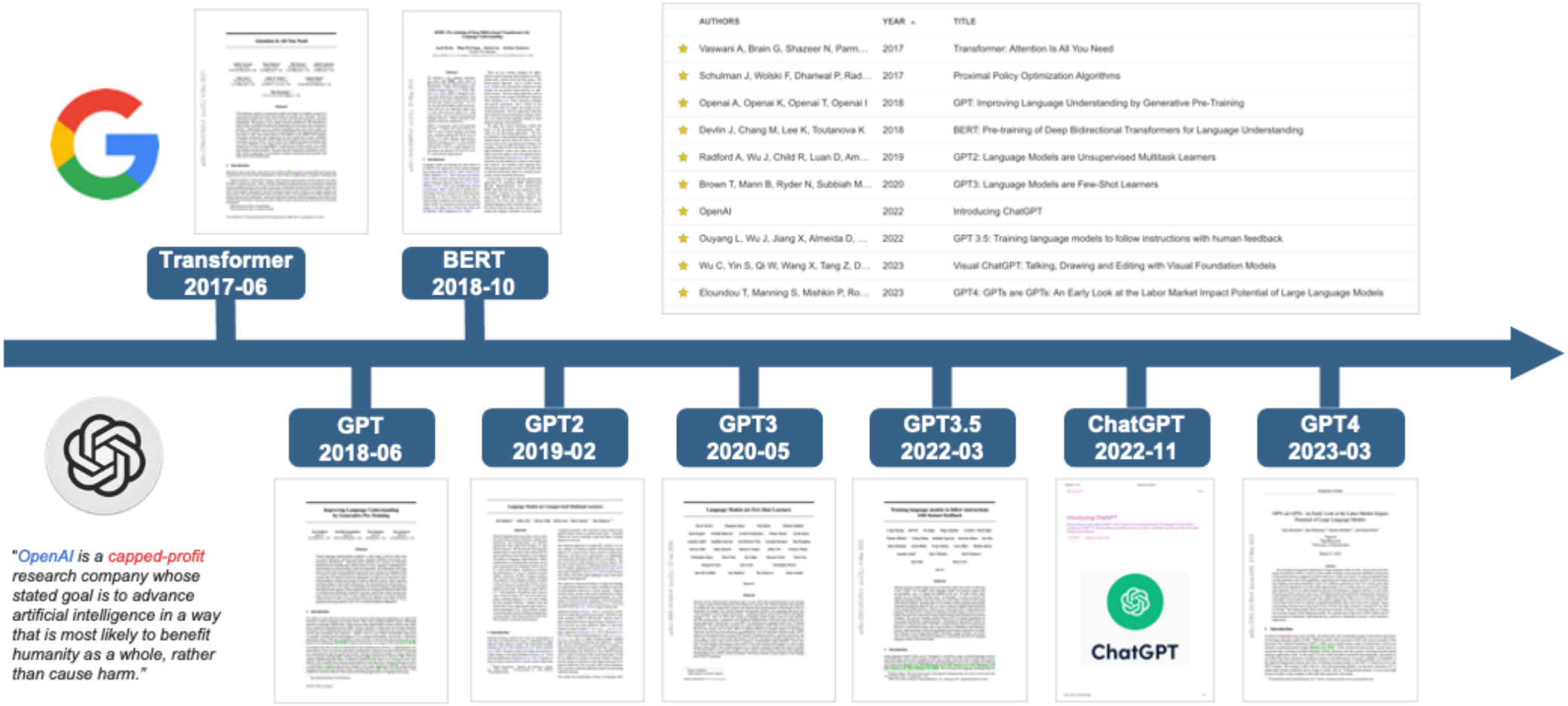
Representation



GPT
Jun 2018

Generation

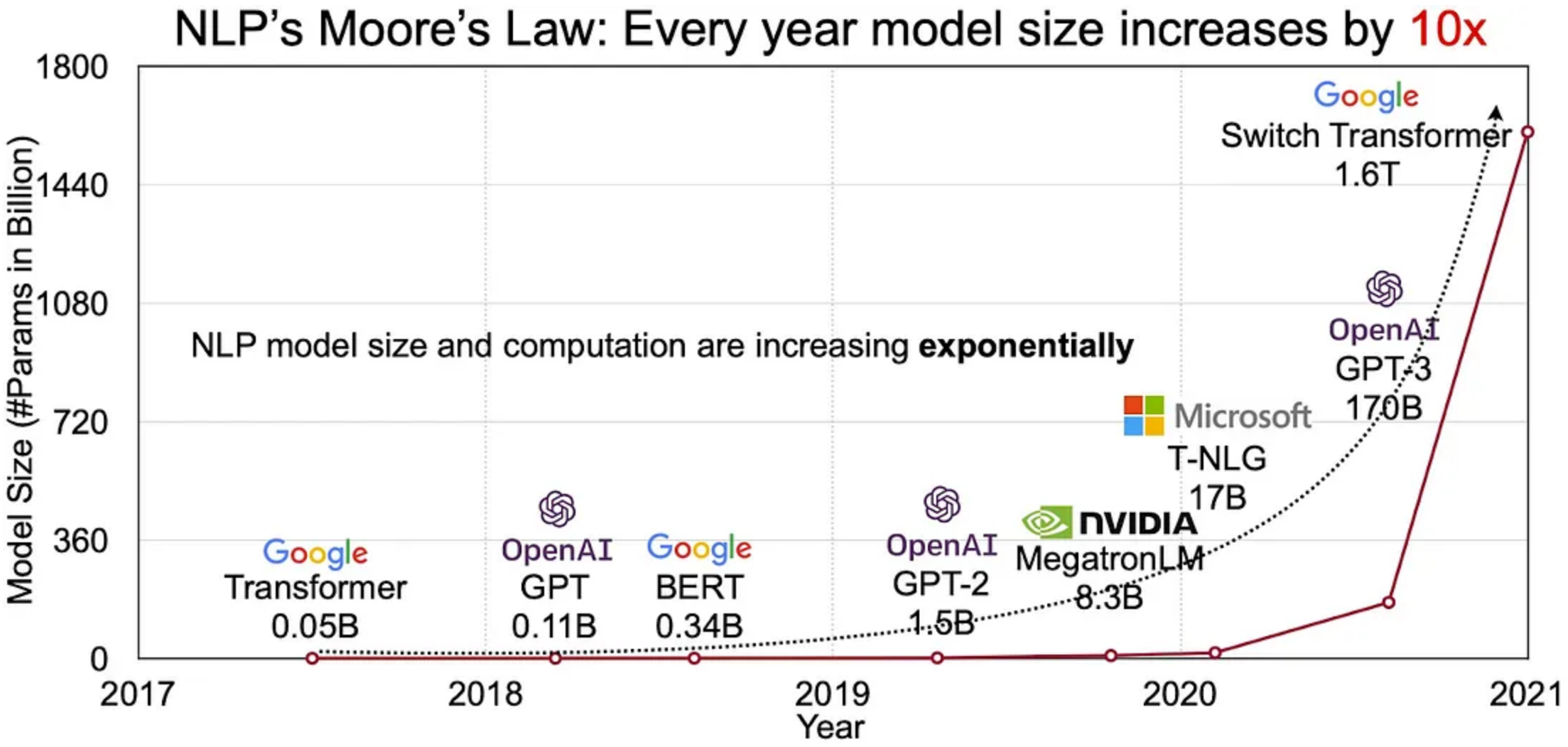
Brief Review: From Transformer to GPT (Optional: CLIP, ViT, ViLT ...)



AUTHORS	YEAR	TITLE
Vaswani A, Brain G, Shazeer N, Parm…	2017	Transformer: Attention Is All You Need
Schulman J, Wolski F, Dhariwal P, Rad…	2017	Proximal Policy Optimization Algorithms
Openai A, Openai K, Openai T, Openai I	2018	GPT: Improving Language Understanding by Generative Pre-Training
Devlin J, Chang M, Lee K, Toutanova K	2018	BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding
Radford A, Wu J, Child R, Luan D, Am…	2019	GPT2: Language Models are Unsupervised Multitask Learners
Brown T, Mann B, Ryder N, Subbiah M…	2020	GPT3: Language Models are Few-Shot Learners
OpenAI	2022	Introducing ChatGPT
Ouyang L, Wu J, Jiang X, Almeida D, …	2022	GPT 3.5: Training language models to follow instructions with human feedback
Wu C, Yin S, Qi W, Wang X, Tang Z, D…	2023	Visual ChatGPT: Talking, Drawing and Editing with Visual Foundation Models
Eloundou T, Manning S, Mishkin P, Ro…	2023	GPT4: GPTs are GPTs: An Early Look at the Labor Market Impact Potential of Large Language Models

2021.08 openAI **Scaling Laws for Neural Language Models**

2021 openAI (CLIP) Learning Transferable Visual Models From Natural Language Supervision


NLP's Moore's Law: Every year model size increases by 10x
Model Size (#Params in Billion)
1800
1440
1080
720
360
0
2017
2018
2019
2020
2021
Year
NLP model size and computation are increasing exponentially
Google
Transformer
0.05B
OpenAI
GPT
0.11B
Google
BERT
0.34B
OpenAI
GPT-2
1.5B
NVIDIA
MegatronLM
8.3B
Microsoft
T-NLG
17B
Google
Switch Transformer
1.6T
OpenAI
GPT-3
170B

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

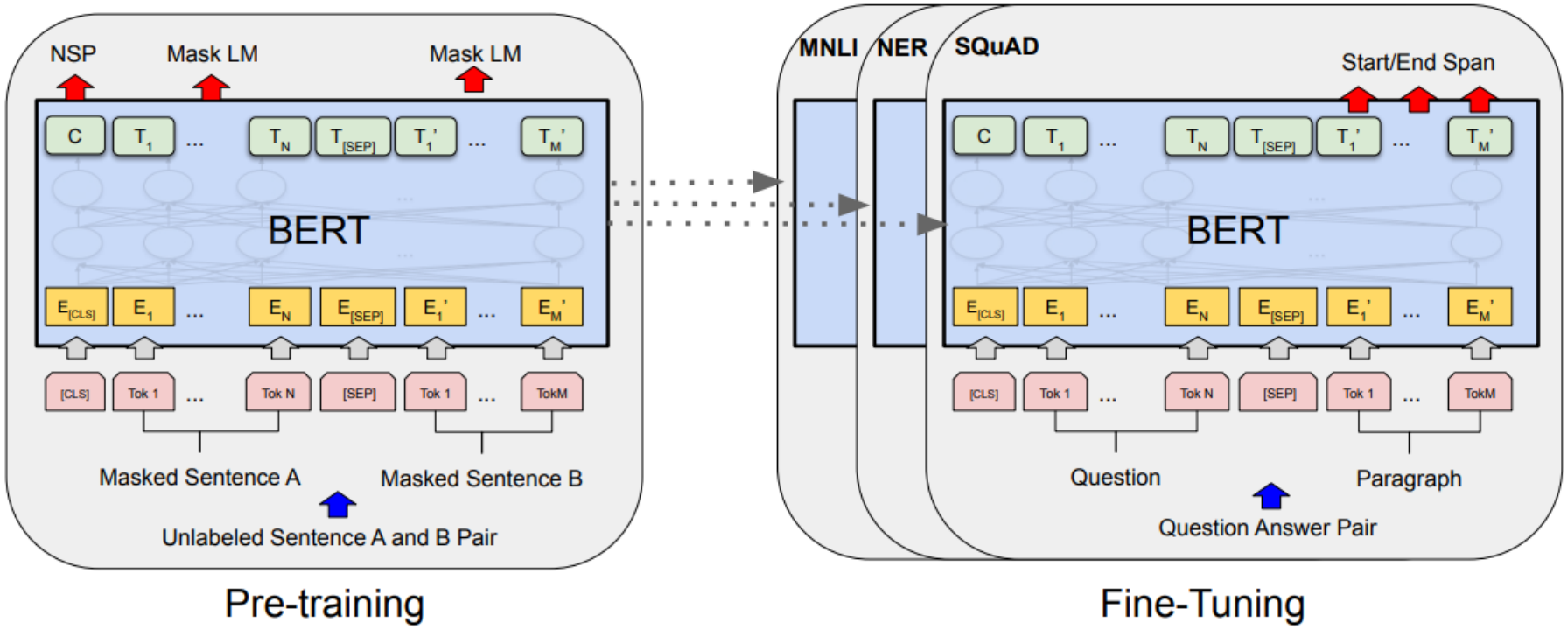
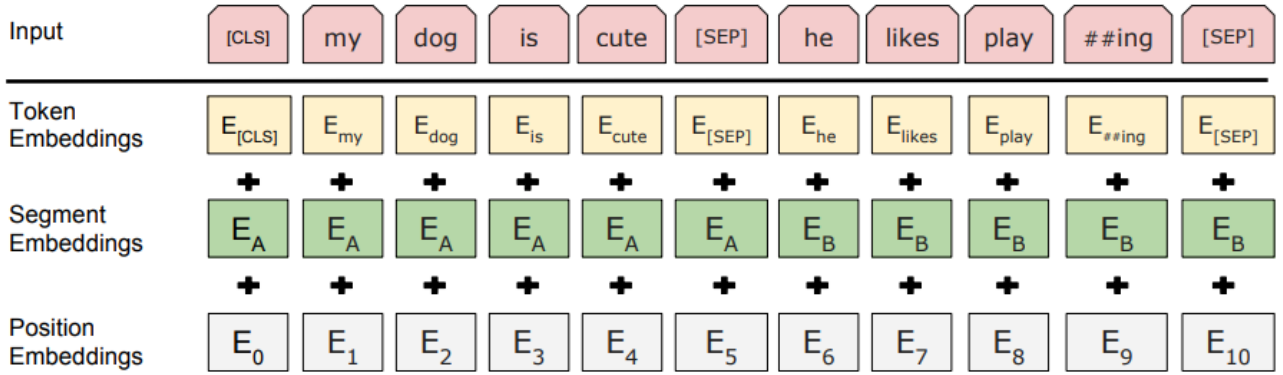


Figure 1: Overall pre-training and fine-tuning procedures for BERT. Apart from output layers, the same architectures are used in both pre-training and fine-tuning. The same pre-trained model parameters are used to initialize models for different down-stream tasks. During fine-tuning, all parameters are fine-tuned. [CLS] is a special symbol added in front of every input example, and [SEP] is a special separator token (e.g. separating questions/answers).

Bidirectional: self attention
Unidirection: LSTM

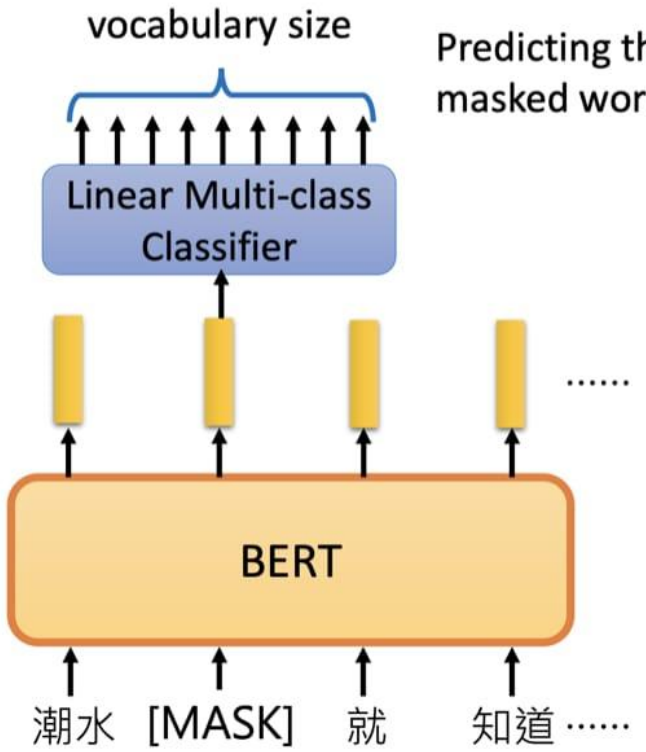
- unified architecture across different tasks
  - Pretrained on unlabeled dataset over different pre-training tasks
  - Fine-turned:
    - initialized with the pre-trained parameters, and all of the parameters are fine-tuned using labeled data from the downstream tasks
- Input (two types in one token seqnece)
  - single sentence
  - a pair of sentences, e.g. , <Question, Answer>
  - How
    - Start with a special class token [CLS], separate two sentences with [SEP] token
    - Learned embedding to indicate sentence A or B



預訓練任務 1：克漏字填空

Training of BERT

- Approach 1:
  Masked LM



預訓練任務 2：下個句子預測

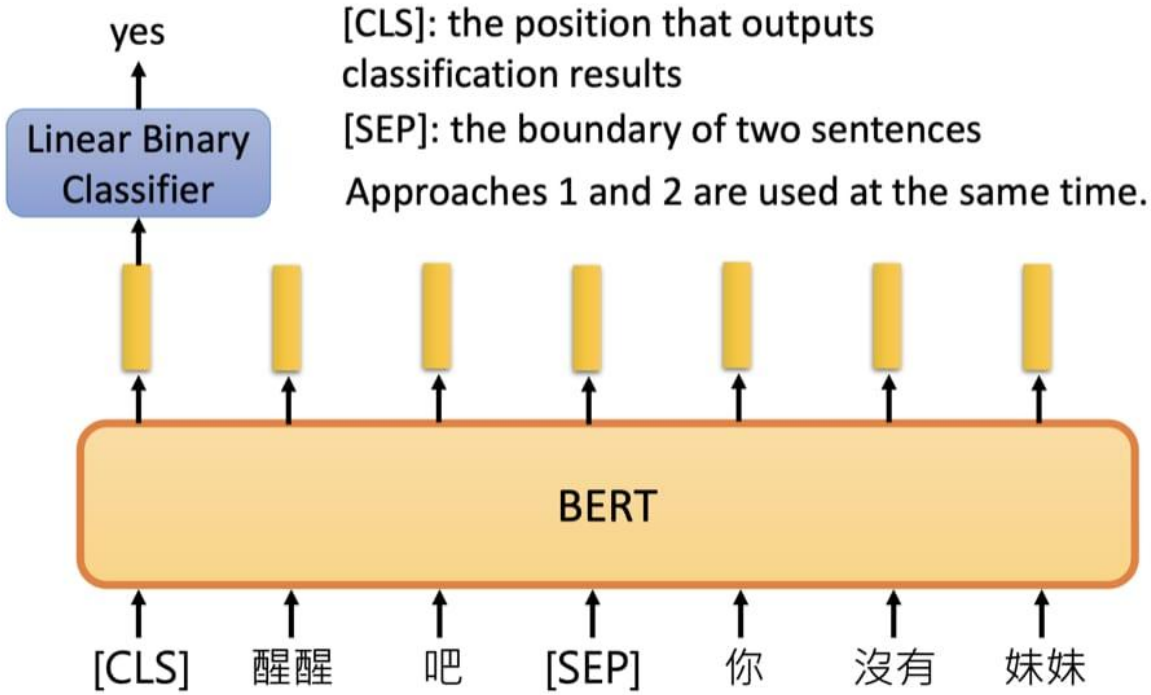
Training of BERT

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.



https://youtu.be/UYPa347-DdE?list=PLJV_el3uVTsOK_ZK5L0Iv_EQoL1JefRL4

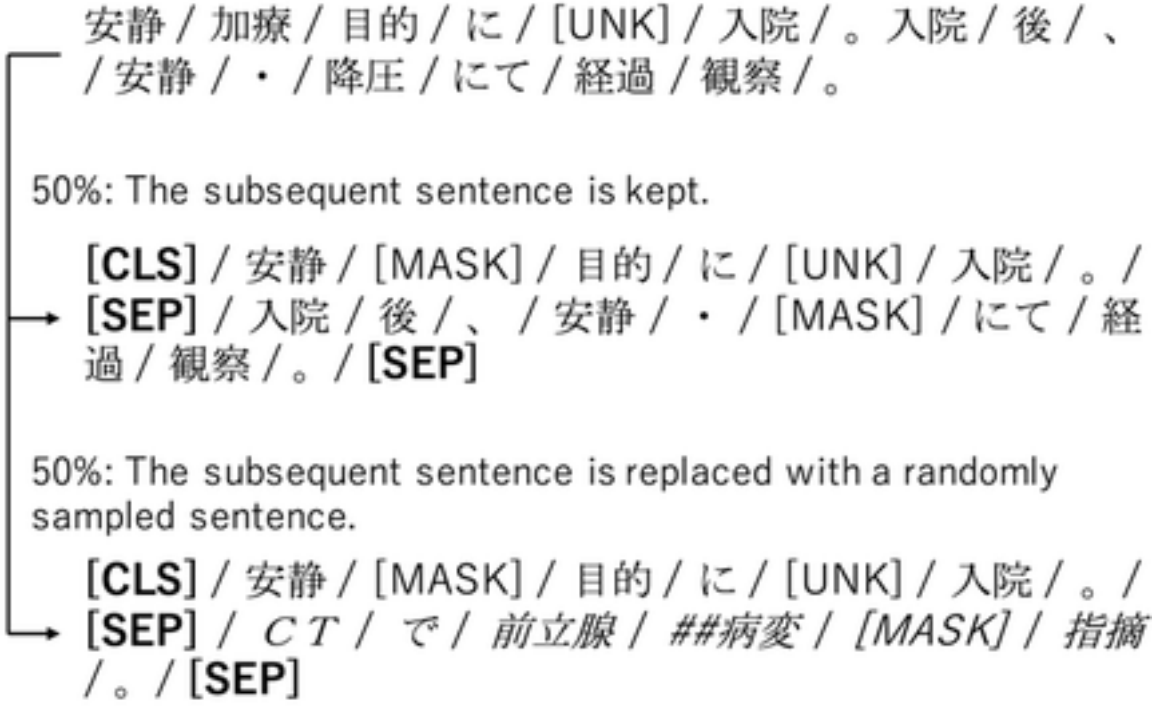
A. Masked LM

For each token, following action is conducted by a 15 percent chance.

Predict the original tokens for the masked, replaced or kept tokens.

B. Next sentence prediction

Original two sentences appears in the dataset.



Predict if the second sentence in the pair is the subsequent in the dataset.

Pretraining Task

- Task #1: Masked LM
  - mask some percentage of the input tokens at random, and then predict those masked tokens
    - Pure random replacement: create a mismatch between pre-training and fine-tuning, since the [MASK] token does not appear during fine-tuning.
- Task #2: Next Sentence Prediction (NSP)

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	**Average** -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

Table 1: GLUE Test results, scored by the evaluation server (https://gluebenchmark.com/leaderboard). The number below each task denotes the number of training examples. The "Average" column is slightly different than the official GLUE score, since we exclude the problematic WNLI set.[8] BERT and OpenAI GPT are single-model, single task. F1 scores are reported for QQP and MRPC, Spearman correlations are reported for STS-B, and accuracy scores are reported for the other tasks. We exclude entries that use BERT as one of their components.

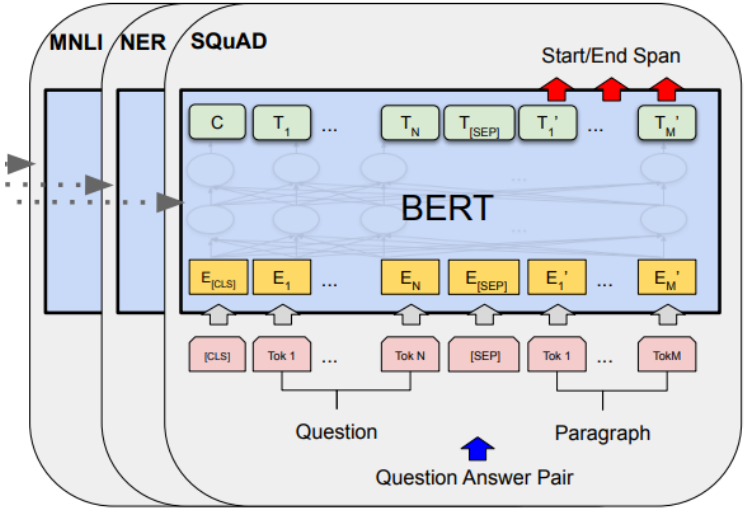
- 好處 1：
  - 無監督數據無限大。不像 ImageNet 還要找人標注數據，要訓練 LM 的話網路上所有文本都是你潛在的資料集（BERT 預訓練使用的數據集共有 33 億個字，其中包含維基百科及 BooksCorpus）
  - Easy to scale
- 好處 2：
  - 厲害的 LM 能夠學會語法結構、解讀語義甚至指代消解。透過特徵擷取或是 fine-tuning 能更有效率地訓練下游任務並提升其表現
- 好處 3：
  - 減少處理不同 NLP 任務所需的 architecture engineering 成本

BERT - Bidirectional Encoder

- **BERT Fine-Tuning:**

- Simply add a task-specific module after the last encoder layer to map it to the desired dimension.

  - *Classification Tasks:*
    - *Add a feed-forward layer on top of the encoder output for the [CLS] token*
  - *Question Answering Tasks:*
    - *Train two extra vectors to mark the beginning and end of answer from paragraph*
  - …

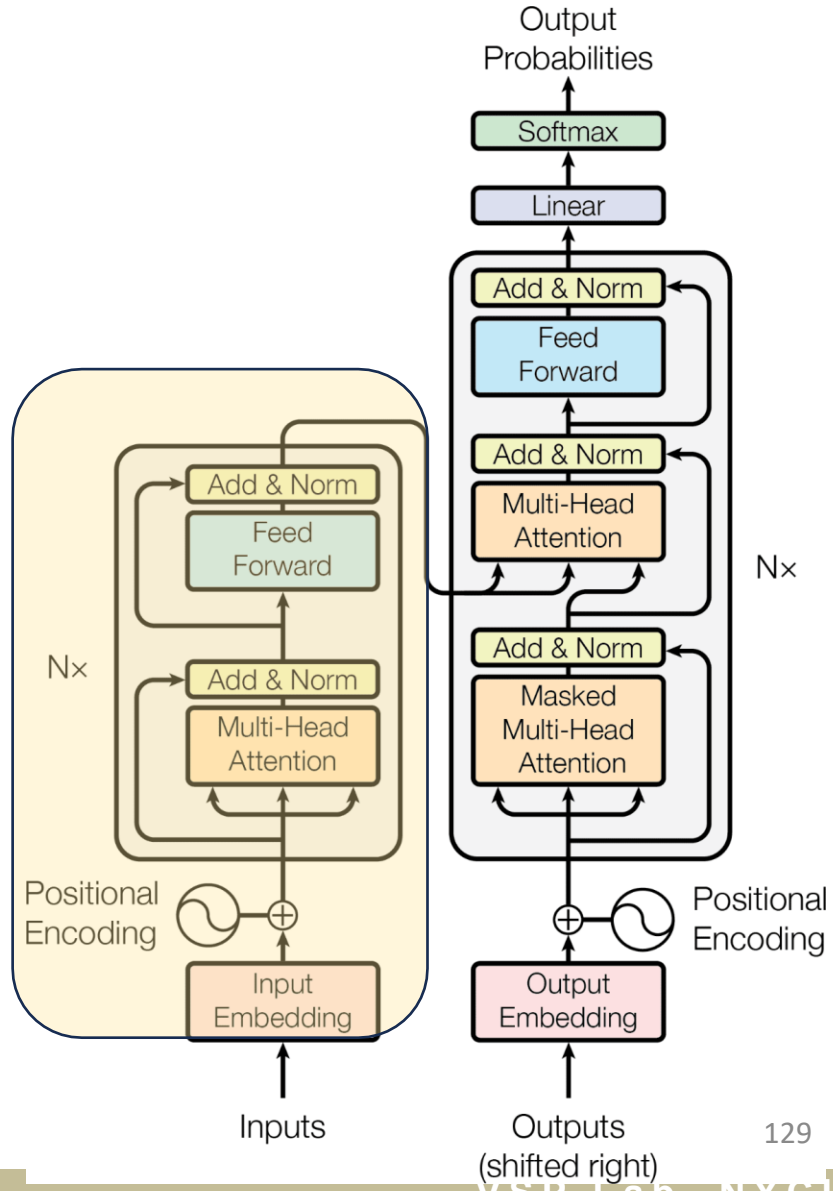


Fine-Tuning

BERT - Bidirectional Encoder

What is our takeaway from BERT?

- **Pre-training tasks can be invented flexibly…**
  - Effective representations can be derived from a flexible regime of pre-training tasks.
- **Different NLP tasks seem to be highly transferable with each other…**
  - As long as we have effective representations, that seems to form a general model which can serve as the backbone for many specialized models.
- **And scaling works!!!**
  - 340M is considered large in 2018

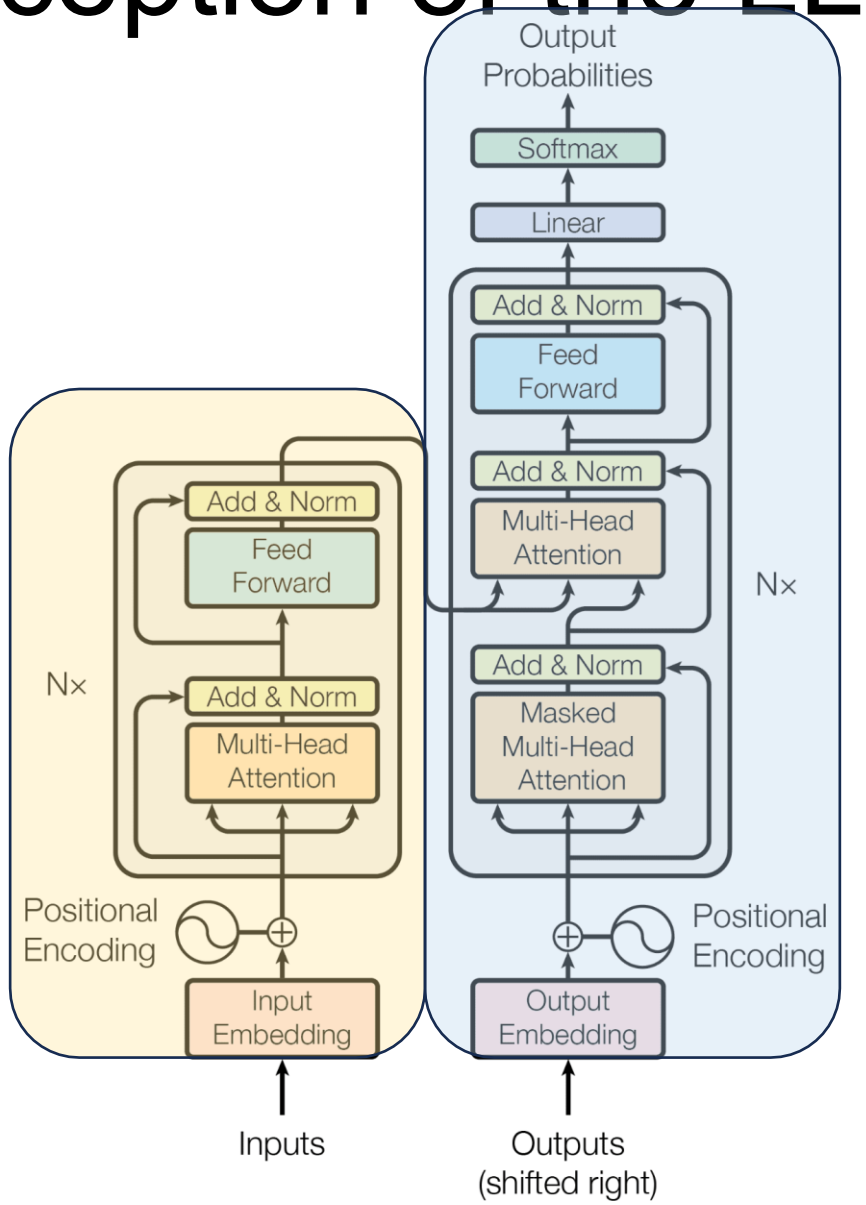


GPT1 TO GPT3

2018 – The Inception of the LLM Era

BERT
Oct 2018

Representation

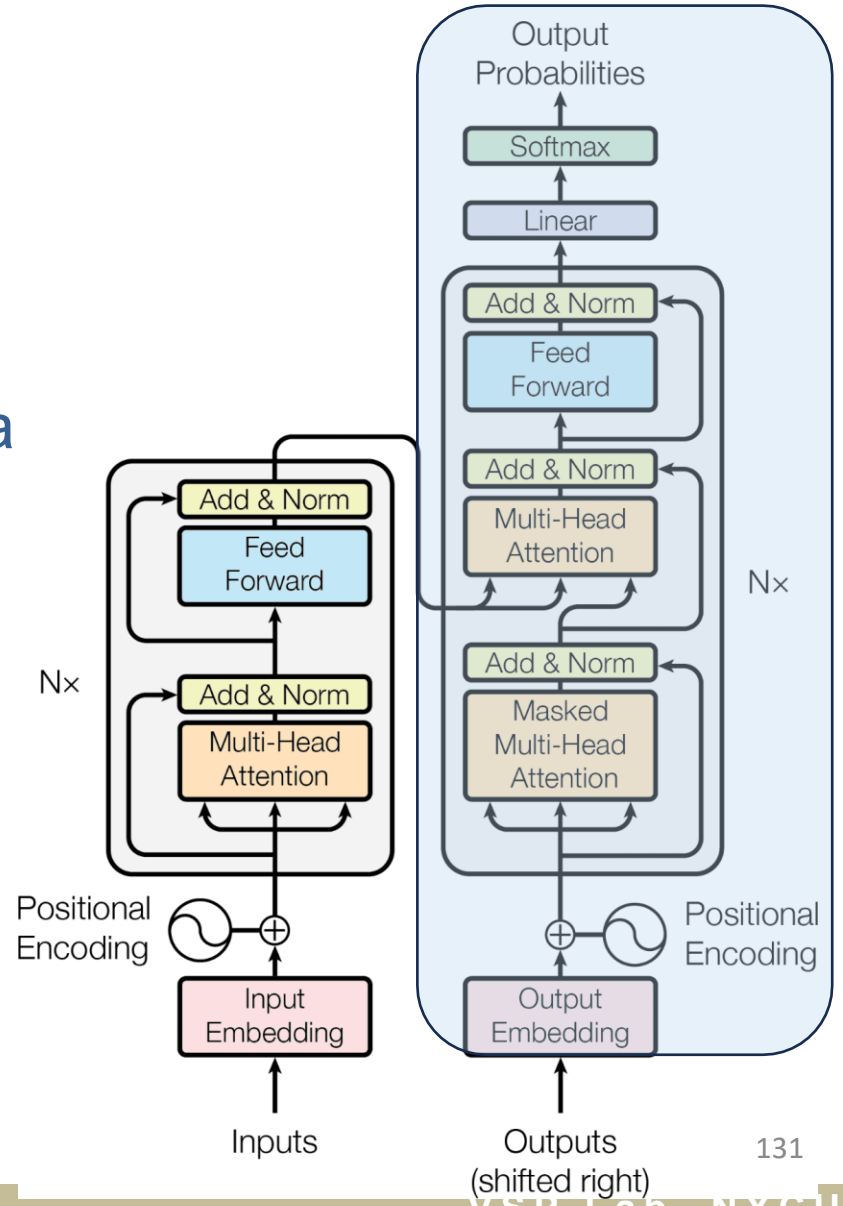


GPT
Jun 2018

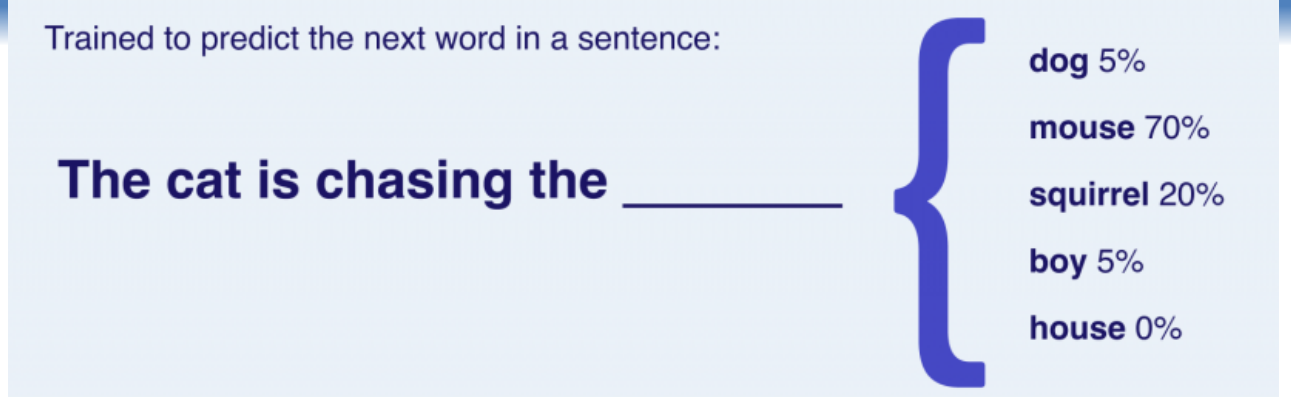
Generation

GPT –Generative Pretrained Transformer

- Similarly motivated as BERT, though differently designed
  - Can we leverage large amounts of unlabeled data to pretrain an LM that understands general patterns?



GPT-1

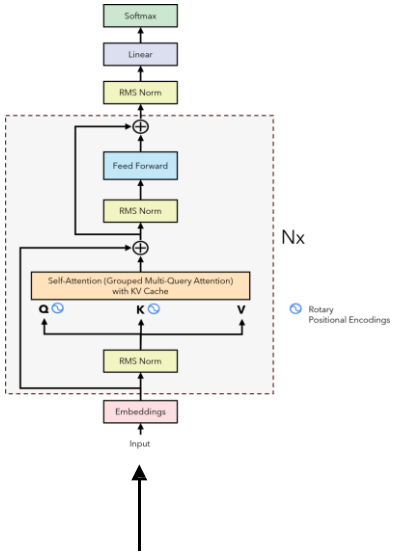


- 目標
  - 在超大的文本資料集上預訓練一個通用的模型，接著再對下游的特定任務微調的PRETRAIN-FINETUNE
- Training
  - 在大量的未標記文本數據上進行預訓練 (pretrained model)
  - 目標函數: 學習預測下一個詞的任務
    - 使用 標準的語言模型。 該目標函數旨在最大化模型輸出與輸入文本的相似度。換句話說，訓練模型以預測文本序列中的下一個詞
    - 所以使用transformer decoder (autoregressive model)

Next Token Prediction Task

Target Love that can quickly seize the gentle heart [EOS]

Training

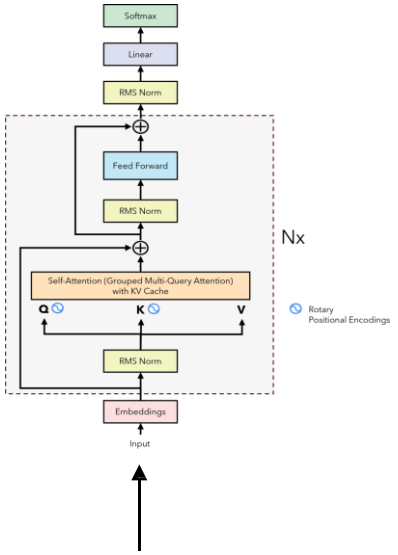


Input [SOS] Love that can quickly seize the gentle heart

Next Token Prediction Task: Inference

Output Love

Inference
T = 1

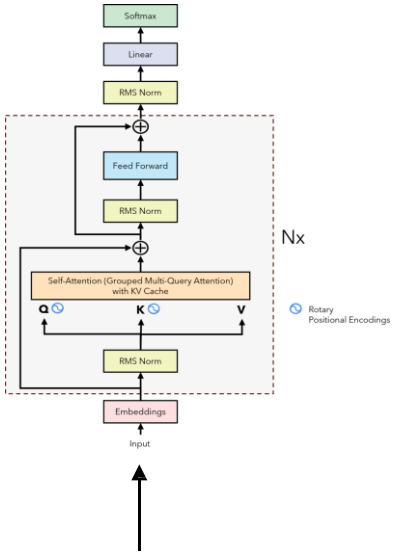


Input [SOS]

Next Token Prediction Task: Inference

Output Love that

Inference
T = 2

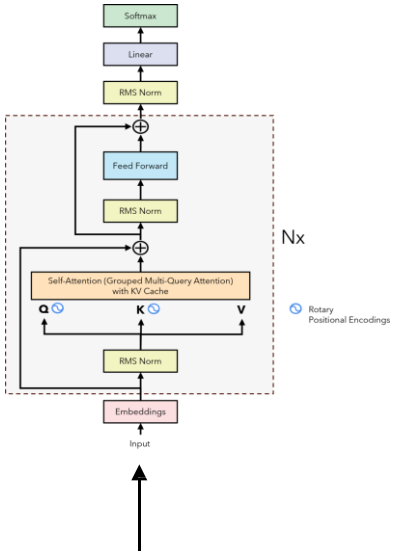


Input [SOS] Love

Next Token Prediction Task: Inference

Output Love that can

Inference
T = 3



Input [SOS] Love that

針對四種常見的 NLP 子任務構造輸入

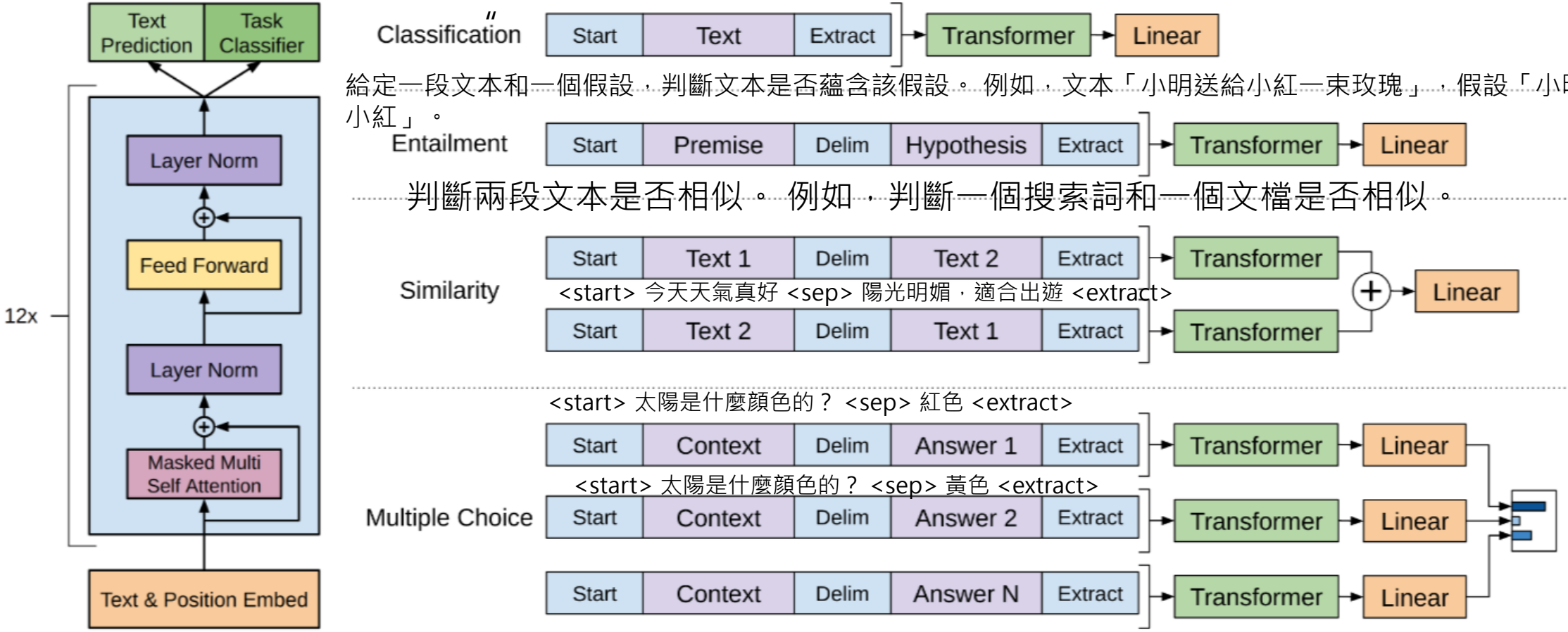


Figure 1: **(left)** Transformer architecture and training objectives used in this work. **(right)** Input transformations for fine-tuning on different tasks. We convert all structured inputs into token sequences to be processed by our pre-trained model, followed by a linear+softmax layer.

GPT-1 Models

In our experiments, we use a multi-layer *Transformer decoder* [34] for the language model, which is a variant of the transformer [62]. This model applies a multi-headed self-attention operation over the input context tokens followed by position-wise feedforward layers to produce an output distribution over target tokens:

$$
\begin{aligned}
h_0 &= UW_e + W_p \\
h_l &= \texttt{transformer\_block}(h_{l-1}) \forall i \in [1, n] \\
P(u) &= \texttt{softmax}(h_n W_e^T)
\end{aligned} \tag{2}
$$

where $U = (u_{-k}, \ldots, u_{-1})$ is the context vector of tokens, n is the number of layers, W_e is the token embedding matrix, and W_p is the position embedding matrix.

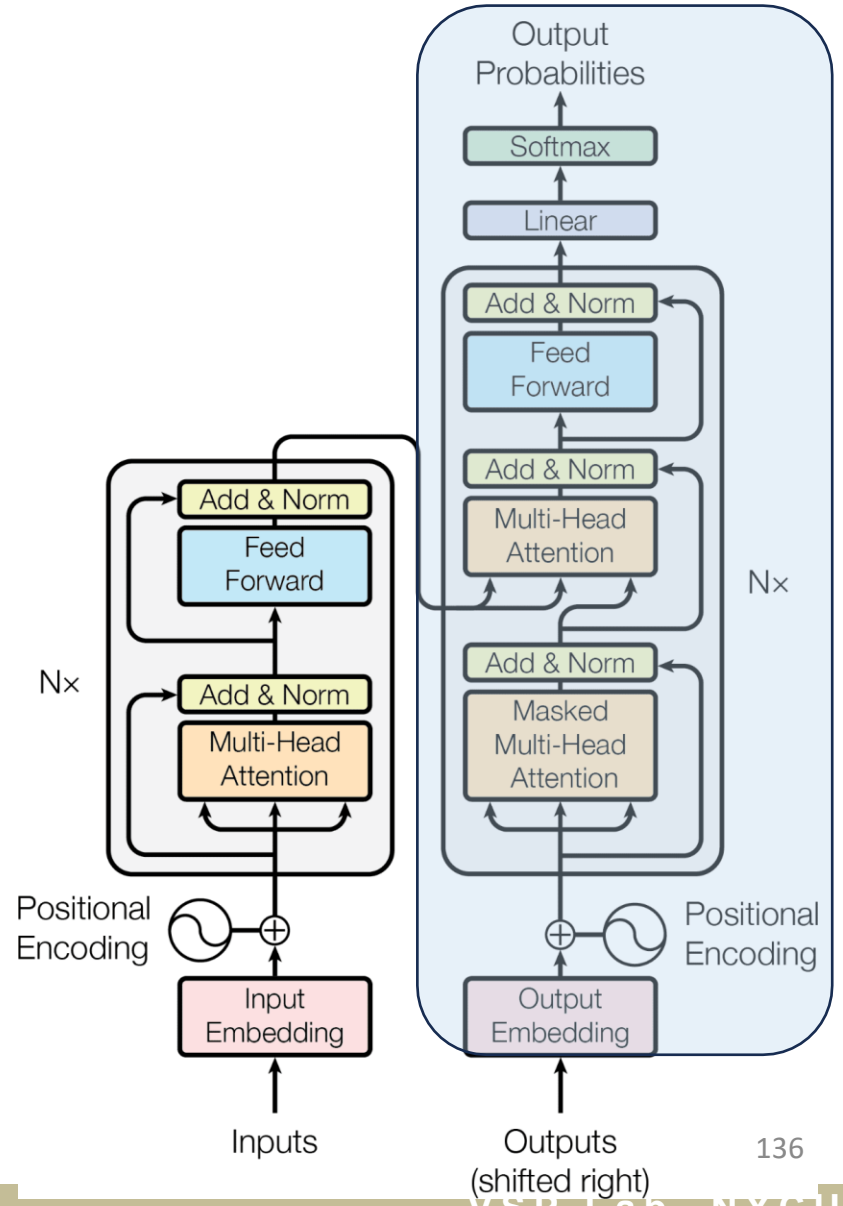
GPT – Generative Pretrained Transformer

Key takeway for GPT-1

- The Effectiveness of Self-Supervised Learning
  - Specifically, the model seems to be able to learn from generating the language itself, rather than from any specific task we might cook up.

- Language Model as a Knowledge Base
  - Specifically, a generatively pretrained model seems to have a decent zero-shot performance on a range of NLP tasks.

- And scaling works!!!



GPT-2

- vs BERT
  - BERT: encoder (bi-directional), GPT-2: decoder (autogressive)
    - 解碼器架構在處理文本序列時只能利用當前位置之前的資訊，而編碼器架構則可以利用整個序列的資訊。
    - 導致 GPT-2 模型更擅長於文本生成任務，而 BERT 模型更擅長於需要理解雙向上下文資訊的任務
  - 目標函數
    - BERT:帶mask的語言模型目標函數，即預測被遮蔽的詞 (克漏字)
    - GPT:標準的語言模型目標函數，即預測文本序列中的下一個詞 (比較難)
- GPT-2
  - Model size: 345M (BERT) => 1542M
  - Zero-Shot 學習設定，即模型不需要任何與下游任務相關的標記數據，就可以直接用於該任務
  - 模型架構進行微調
    - LAYER NORMALIZATION被放到了每一個解碼器的前端，並且在最後的一層隱層輸出做了一個LN
    - 對於模型預訓練的穩定性和微調有著重要差異

Language Models are Unsupervised Multitask Learners

Zero-Shot

- 模型無需任何與下游任務相關的標註數據，即可直接應用於該任務
  - 源於 GPT-2 更大的模型規模和更豐富的數據集，使其在預訓練階段學習到更廣泛的語言知識。
- How
  - GPT-2 在構造下游任務輸入時，捨棄了在預訓練階段未見過的特殊符號，例如起始符號、分隔符號和結束符號。
  - GPT-2 使用更接近自然語言的提示（prompt）來引導模型理解任務目標
- 效果
  - 效果仍有限 (vs BERT)，不如使用微調方法的模型
  - 原因
    - 模型的解碼器架構, prompt 設計，limited model size and dataset

GPT-3

- 100X model size, larger dataset
- 放棄了 GPT-2 的 Zero-Shot 學習設定，轉而採用 Few-Shot 學習
  - 為每個下游任務提供少量（通常是 10 到 100 個）訓練樣本
  - GPT-3 的 Few-Shot 學習設定 不涉及梯度更新或微調，模型會根據提供的樣本和任務提示，直接進行預測
  - In-Context Learning
    - 模型會將訓練樣本作為輸入的一部分，並嘗試從中學習模式和關係，而無需更新模型權重。
  - 更接近人類的學習方式，也更符合實際應用場景
- 效能
  - 顯著的性能提升，在許多情況下甚至可以媲美使用大量標註數據進行微調的算法
  - 更強的文本生成能力，可以生成逼真且連貫的長篇文本，例如新聞稿、故事和對話等

The three settings we explore for in-context learning

Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

```
Translate English to French:        ← task description
cheese =>                           ← prompt
```

One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.

```
Translate English to French:        ← task description
sea otter => loutre de mer          ← example
cheese =>                           ← prompt
```

Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

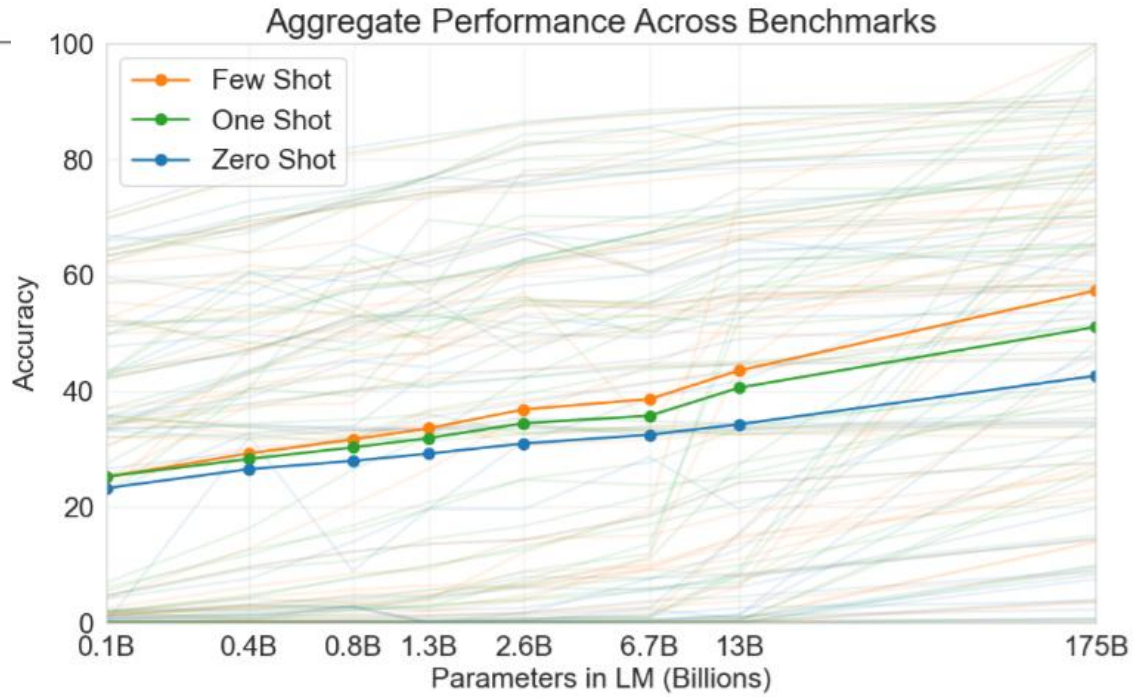
```
Translate English to French:        ← task description
sea otter => loutre de mer          ← examples
peppermint => menthe poivrée
plush girafe => girafe peluche
cheese =>                           ← prompt
```

Traditional fine-tuning (not used for GPT-3)

Fine-tuning

The model is trained via repeated gradient updates using a large corpus of example tasks.

```
sea otter => loutre de mer          ← example #1
        ↓
   gradient update
        ↓
peppermint => menthe poivrée        ← example #2
        ↓
   gradient update
        ↓
        ...
        ↓
plush giraffe => girafe peluche     ← example #N
        ↓
   gradient update
        ↓
cheese =>                           ← prompt
```



- Large batch size
  - 提升計算性能/降低通訊量/減少overfitting

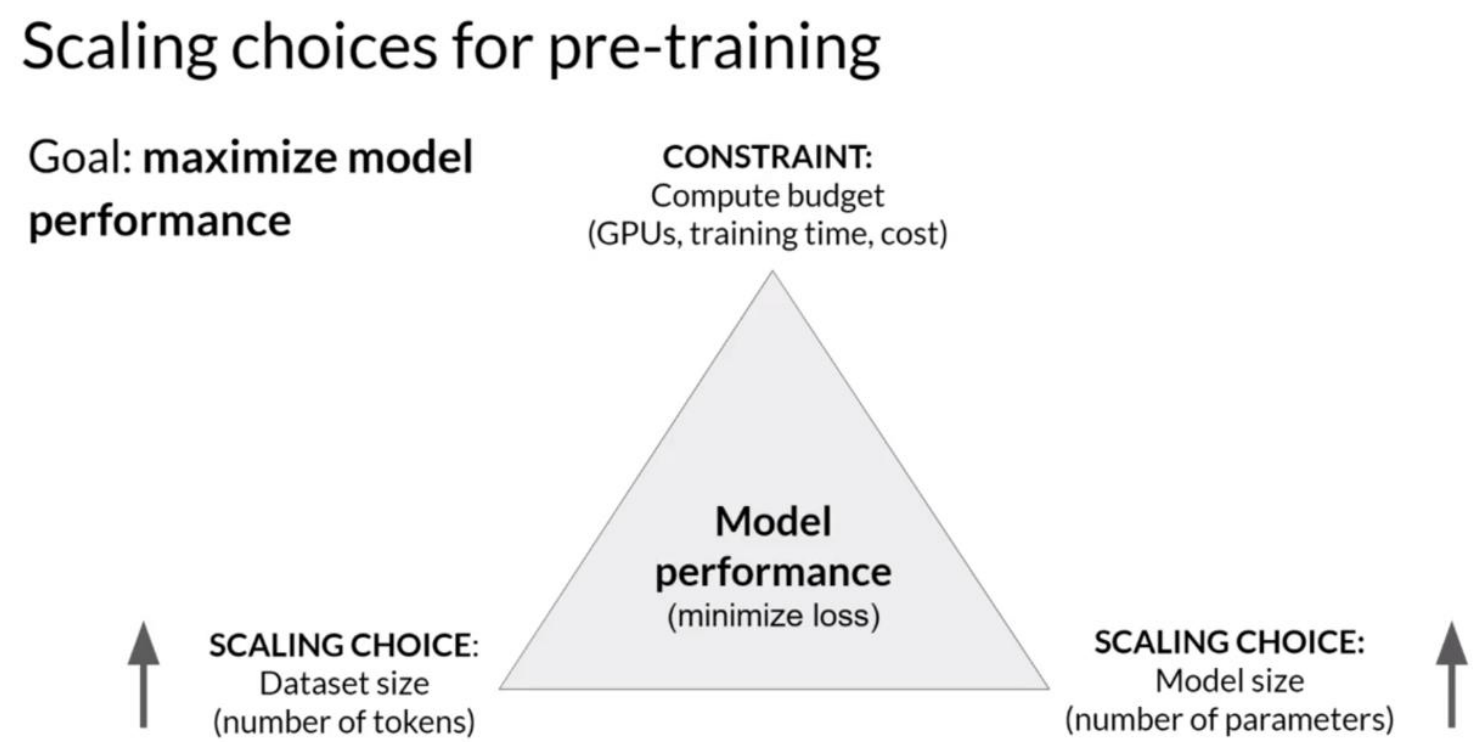
Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-4}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}
GPT-3 2.7B	2.7B	32	2560	32	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.

- 為了提高數據集的品質，GPT-3 的訓練數據經過了三個步驟的處理
  - 過濾
    - 使用 GPT-2 爬取的 Reddit 數據集作為正例，Common Crawl 數據集作為負例，訓練一個二元分類器，過濾掉 Common Crawl 中低質量的網頁
  - 去除重複
  - 加高品質數據集：
    - 加入BERT、GPT-2 和 GPT 等模型使用的數據集
- Data sampling 比例偏重高品質數據

Dataset	Quantity (tokens)	Weight in training mix	Epochs elapsed when training for 300B tokens
Common Crawl (filtered)	410 billion	60%	0.44
WebText2	19 billion	22%	2.9
Books1	12 billion	8%	1.9
Books2	55 billion	8%	0.43
Wikipedia	3 billion	3%	3.4

Table 2.2: Datasets used to train GPT-3. "Weight in training mix" refers to the fraction of examples during training



SCALING LAWS

What Drives the Performance in DL?

- More data
- Bigger model
- More compute

What Drives the Performance in DL? Data

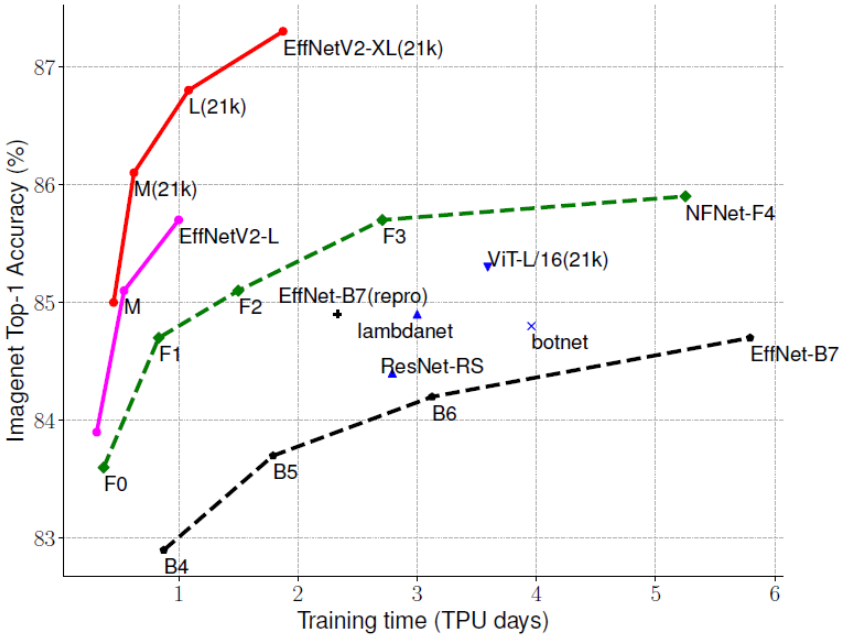
- More data
  - Neural networks can efficiently utilize large amounts of data for training

ImageNet: 1M images

Common Crawl

Common Crawl: 400+B tokens)

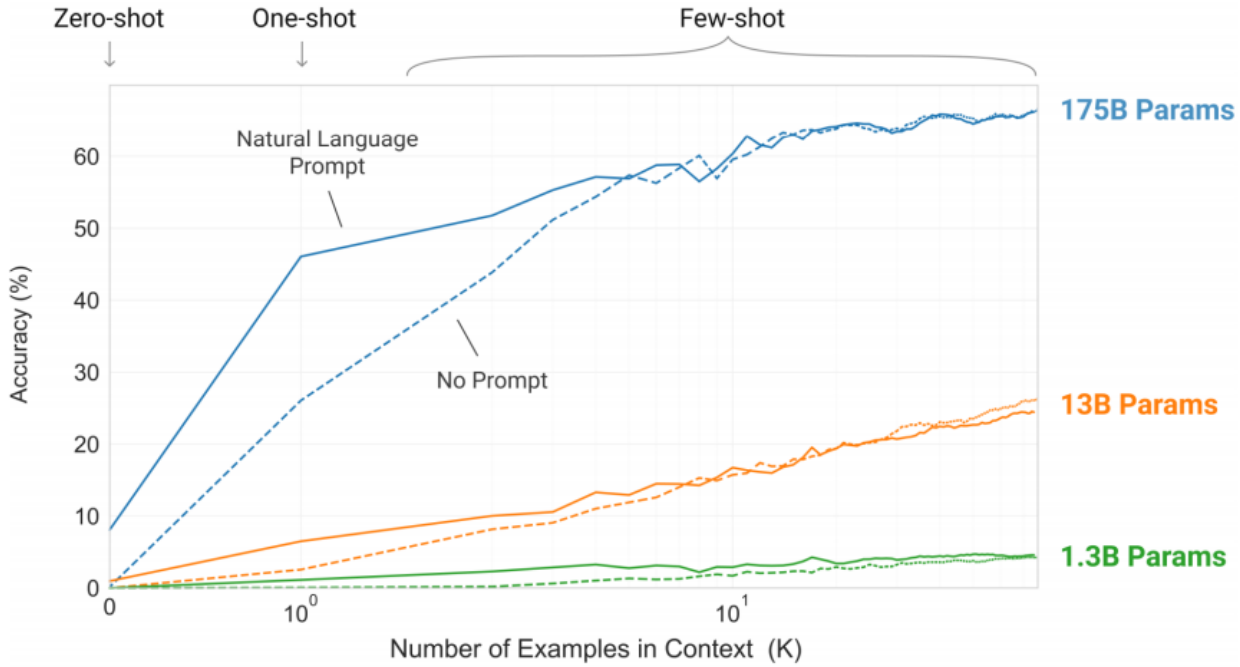
What Drives the Performance in DL? Big model



(a) Training efficiency.

	EfficientNet (2019)	ResNet-RS (2021)	DeiT/ViT (2021)	EfficientNetV2 (ours)
Top-1 Acc.	84.3%	84.0%	83.1%	83.9%
Parameters	43M	164M	86M	24M

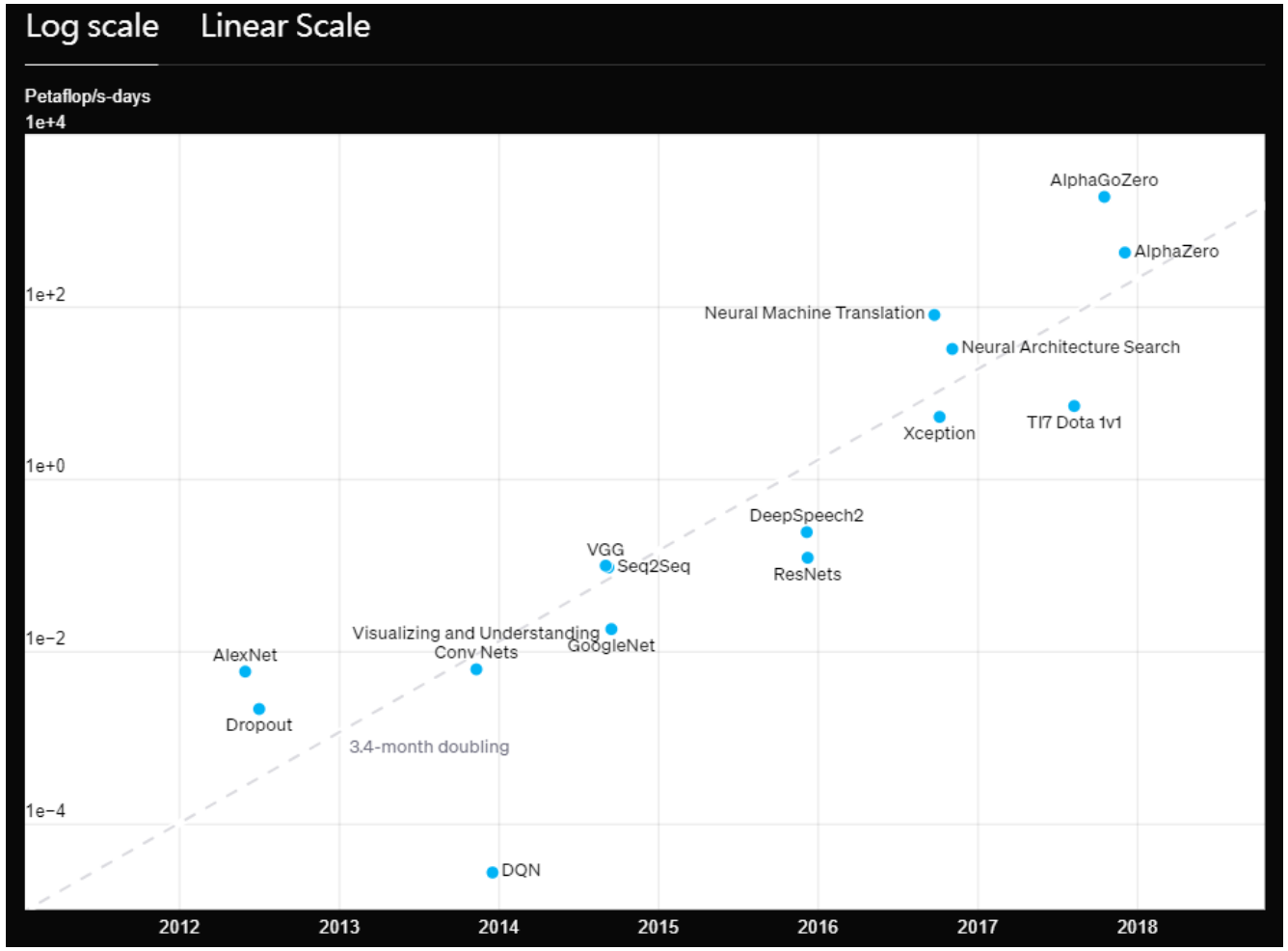
(b) Parameter efficiency.



GPT-3

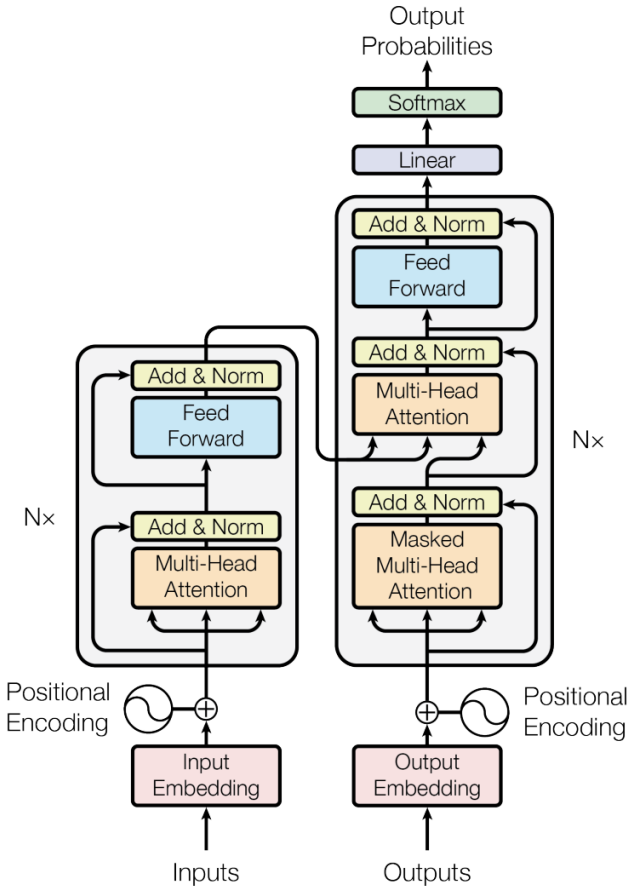
What Drives the Performance in DL? Compute

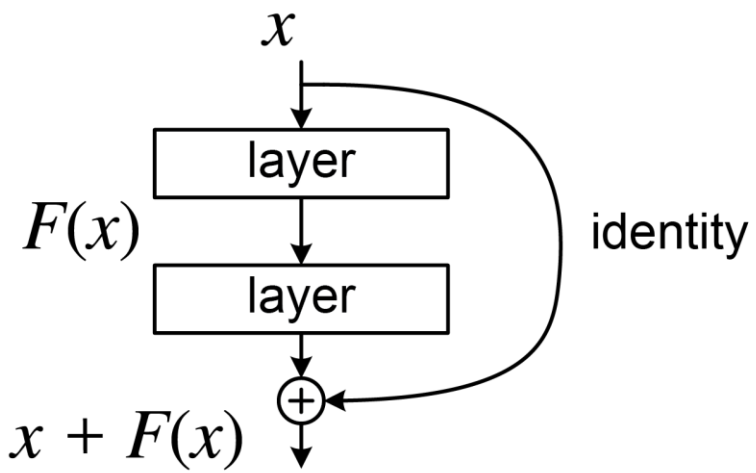
- OpenAI Blog: AI and Compute

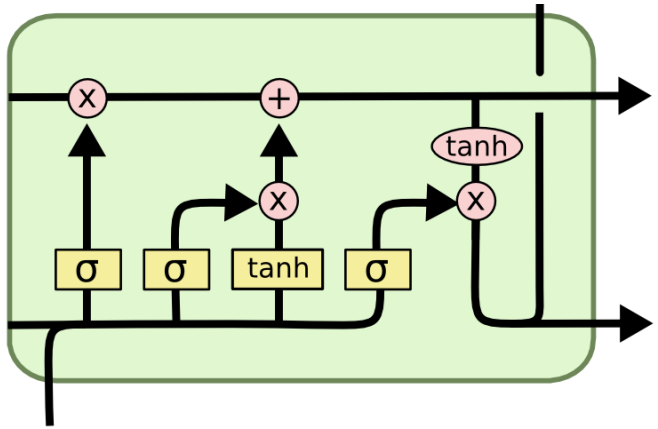


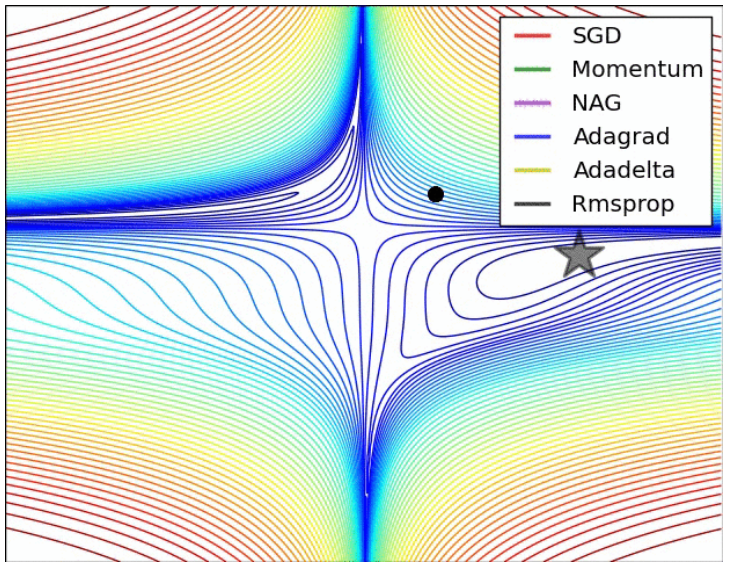
What Drives the Performance in DL? Others

- Better architecture, optimization, algorithms, or other tricks







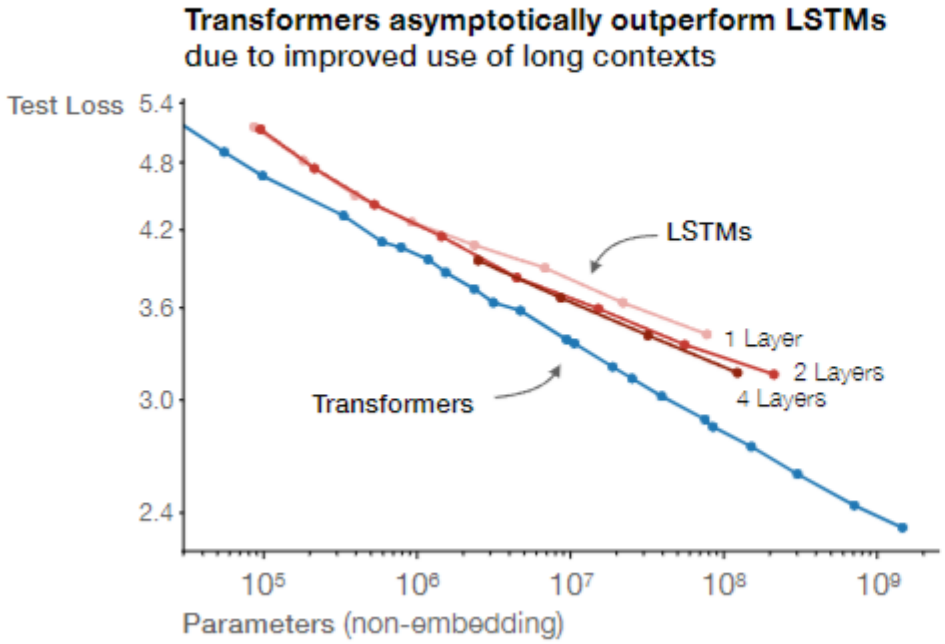


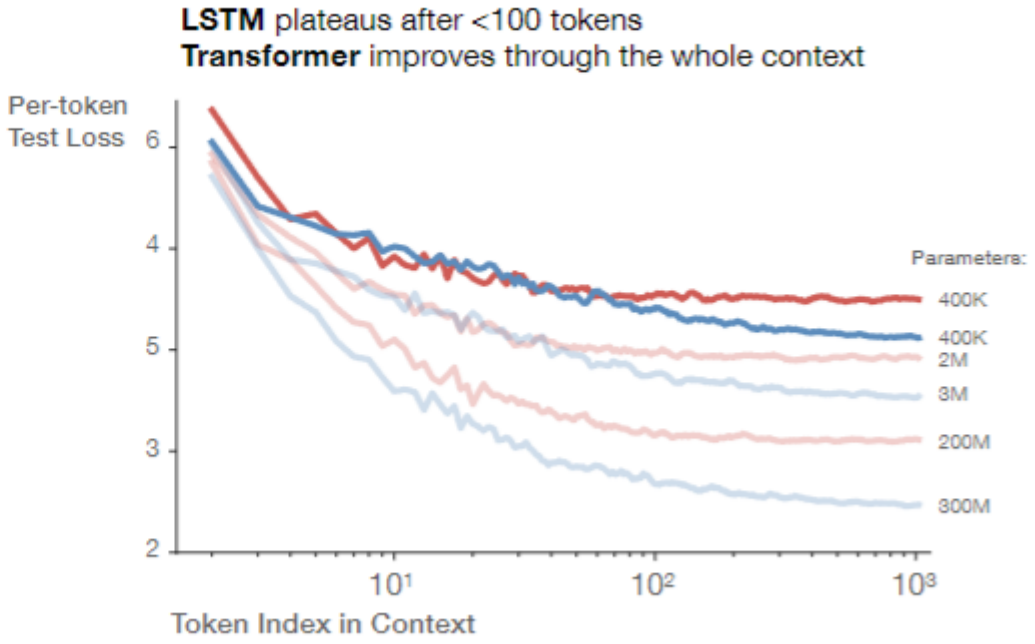
Why Care About Scaling?

- As more data is used for training, they perform better
- As neural networks become bigger, they perform better
- Advance in AI seems to be heavily driven by compute
- Understand choices that would scale with more compute
  - "..the only thing that matter in the long run is the leveraging of computing." [Rich Sutton, the Bitter Lesson]

Why care about a theory for scaling?

- Predict future performance given more data/bigger models
- Understand what problems/design choices are important
- Search and design models/ data/ optimization that scale better?



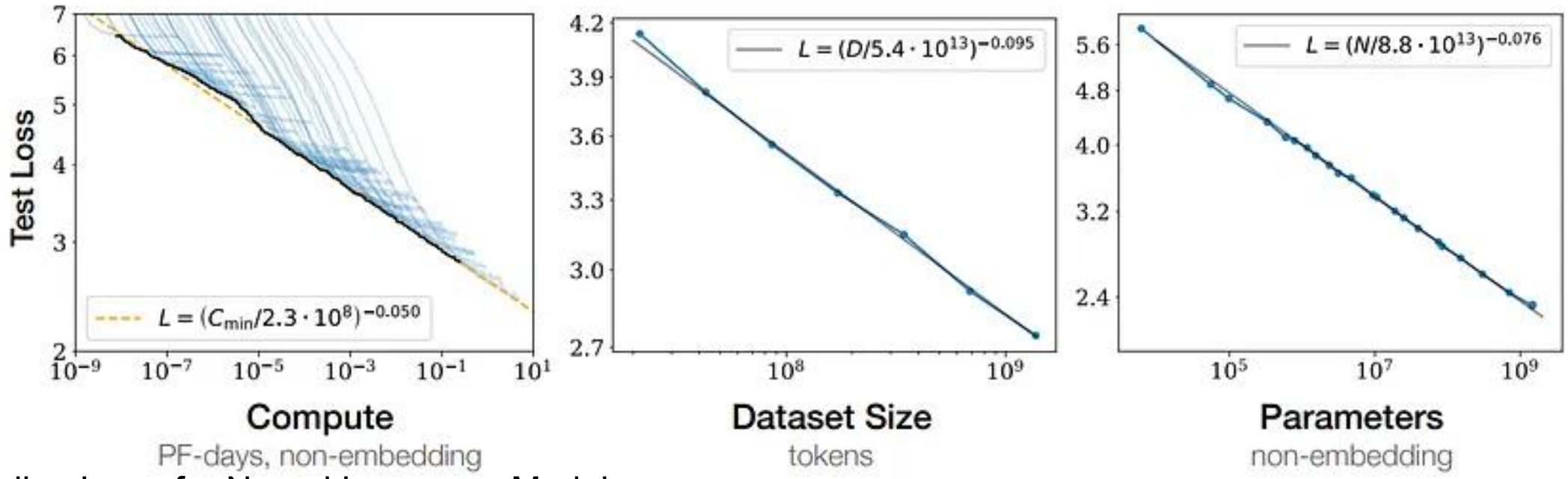


Scaling Laws for Neural Language Models

Scaling Law

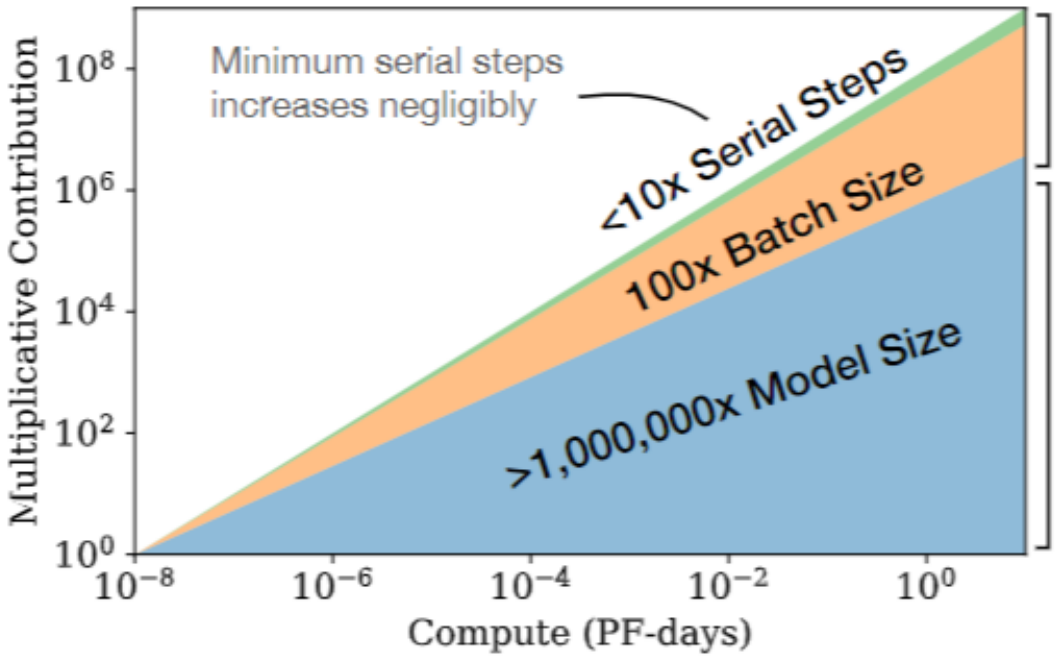
Problem dependent

- Scaling Law 定義：Loss ~= $1/N^r = N^{-r}$
  - 我們可以用模型大小、Dataset大小、用於訓練的總計算量，來預測模型最終能力。（通常以相對簡單的函數型態, ex: Linear relationship）
  - 當不受其他兩個因素的限制時，模型表現與每個單獨的因素都有power-law relationship



Scaling Laws for Neural Language Models

How to Allocate Your Computing Budget

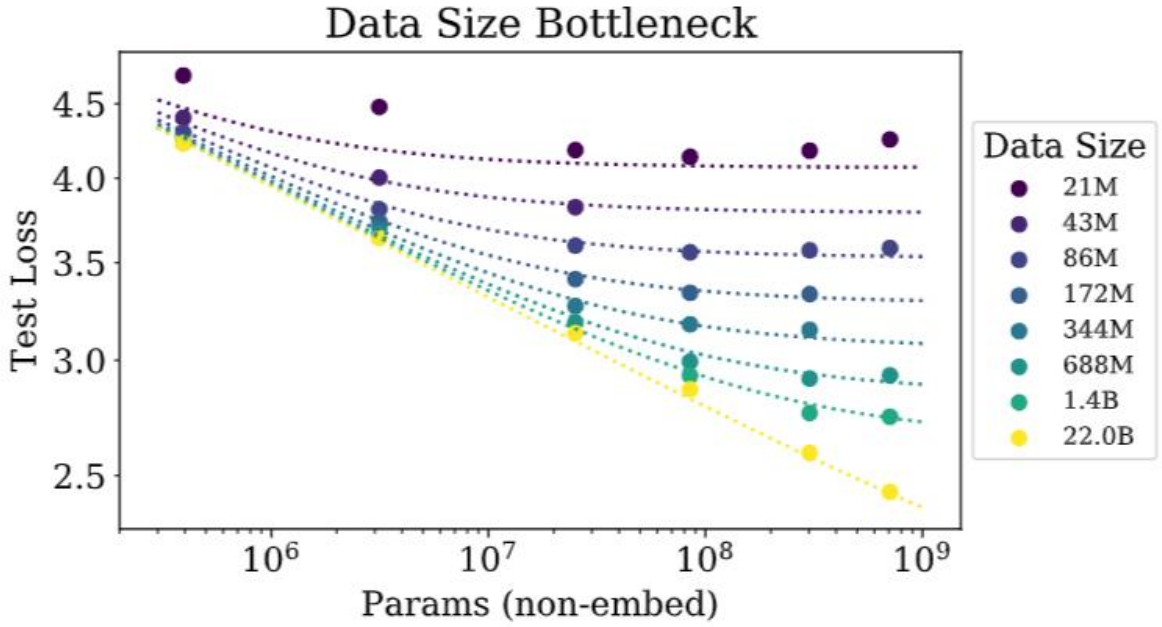


$D \sim C^{0.27}$

Figure 3 As more compute becomes available, we can choose how much to allocate towards training larger models, using larger batches, and training for more steps. We illustrate this for a billion-fold increase in compute. For optimally compute-efficient training, most of the increase should go towards increased model size. A relatively small increase in data is needed to avoid reuse. Of the increase in data, most can be used to increase parallelism through larger batch sizes, with only a very small increase in serial training time required.

計算量增加時，最高效的訓練方式是使用大模型，提高batch，並且相應的擴大少量資料。

Universality of overfitting



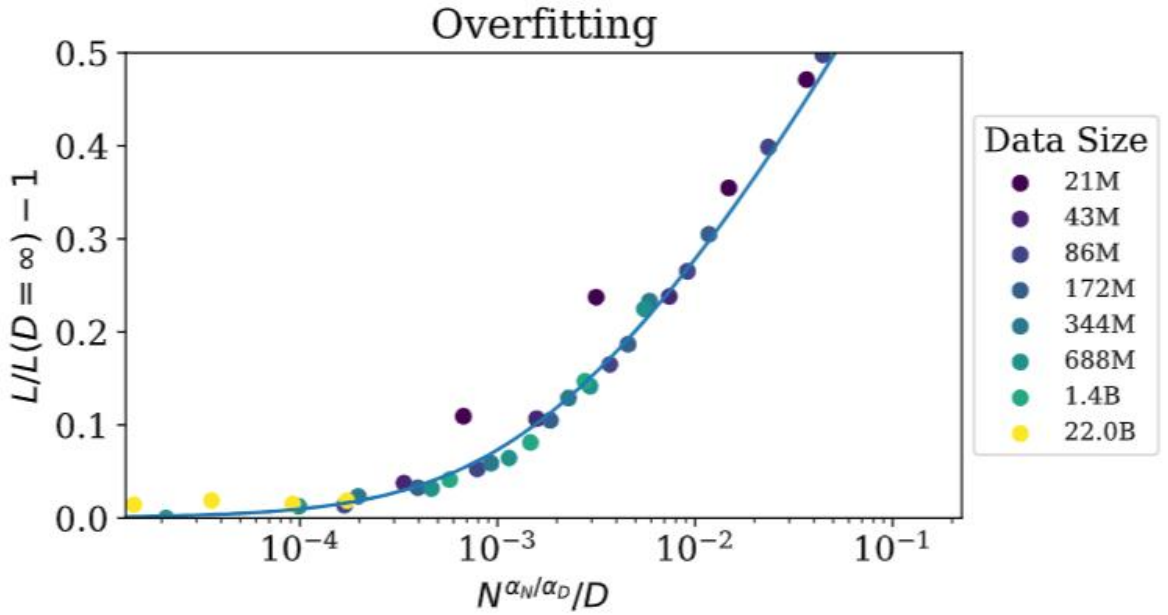
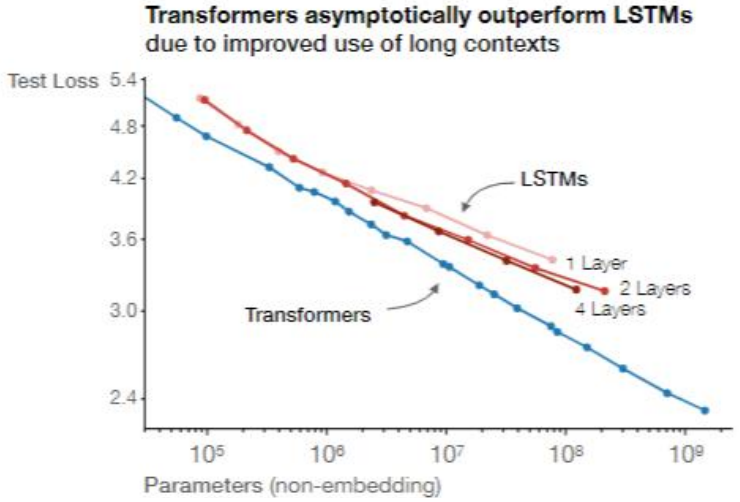


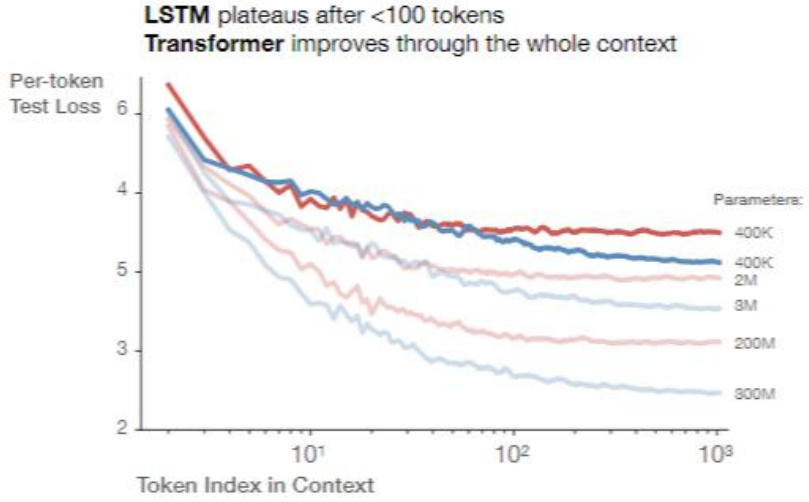
Figure 9 The early-stopped test loss $L(N, D)$ depends predictably on the dataset size D and model size N according to Equation (1.5). **Left**: For large D, performance is a straight power law in N. For a smaller fixed D, performance stops improving as N increases and the model begins to overfit. (The reverse is also true, see Figure 4.) **Right**: The extent of overfitting depends predominantly on the ratio $N^{\frac{\alpha_N}{\alpha_D}}/D$, as predicted in equation (4.3). The line is our fit to that equation.

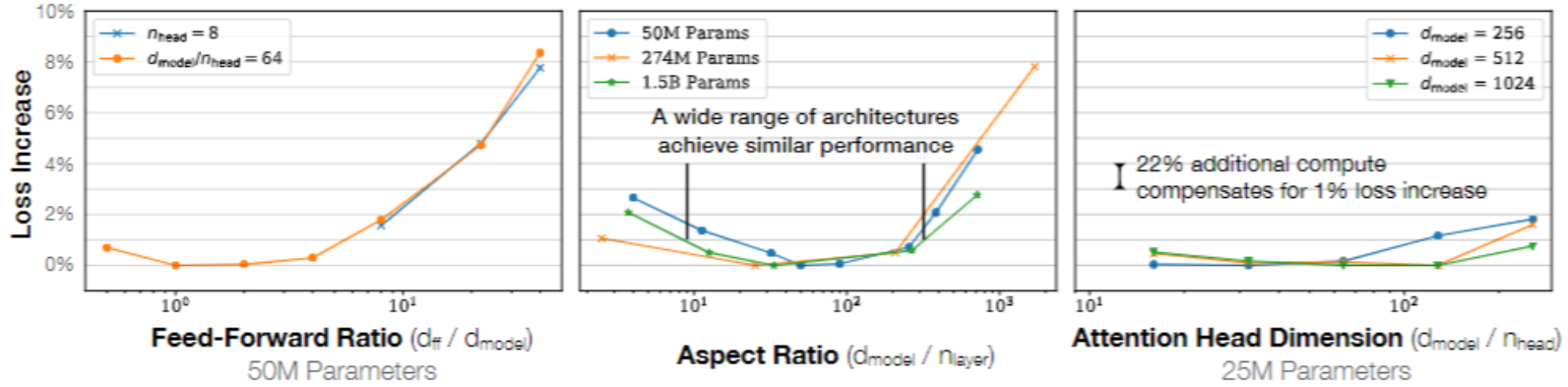
一起增大 N 和 D ， 性能就會可預測得提高。但是當其中一個被固定，另一個在增加時，性能就會下降。二者比例關係大致為 N^{0.74}/D，這意味著，每次將模型增大8倍，只需要將資料量增大6倍來避免性能下降 (過擬合)

Architecture's Less Important

- Weak relationship to model shape (depth, width, number of head)
- …except when the architecture itself creates a bottleneck







Sample Efficiency

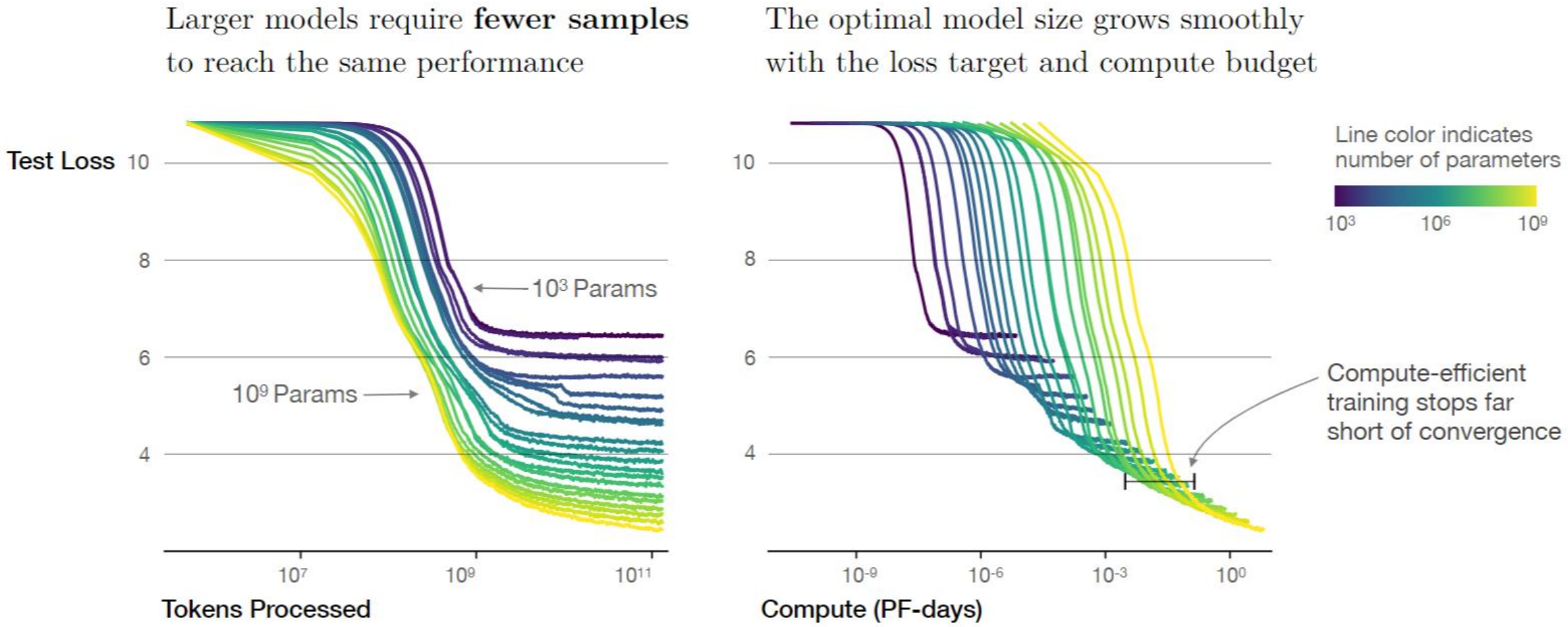


Figure 2 We show a series of language model training runs, with models ranging in size from 10^3 to 10^9 parameters (excluding embeddings).

大模型能在更少的step內，更少的數據上達到相同的性能

1.2 Summary of Scaling Laws

The test loss of a Transformer trained to autoregressively model language can be predicted using a power-law when performance is limited by only either the number of non-embedding parameters N, the dataset size D, or the optimally allocated compute budget $C_{\min}$ (see Figure 1):

1. For models with a limited number of parameters, trained to convergence on sufficiently large datasets: 模型參數受限時

$$L(N) = (N_c/N)^{\alpha_N}; \quad \alpha_N \sim 0.076, \quad N_c \sim 8.8 \times 10^{13} \text{ (non-embedding parameters)} \tag{1.1}$$

2. For large models trained with a limited dataset with early stopping:

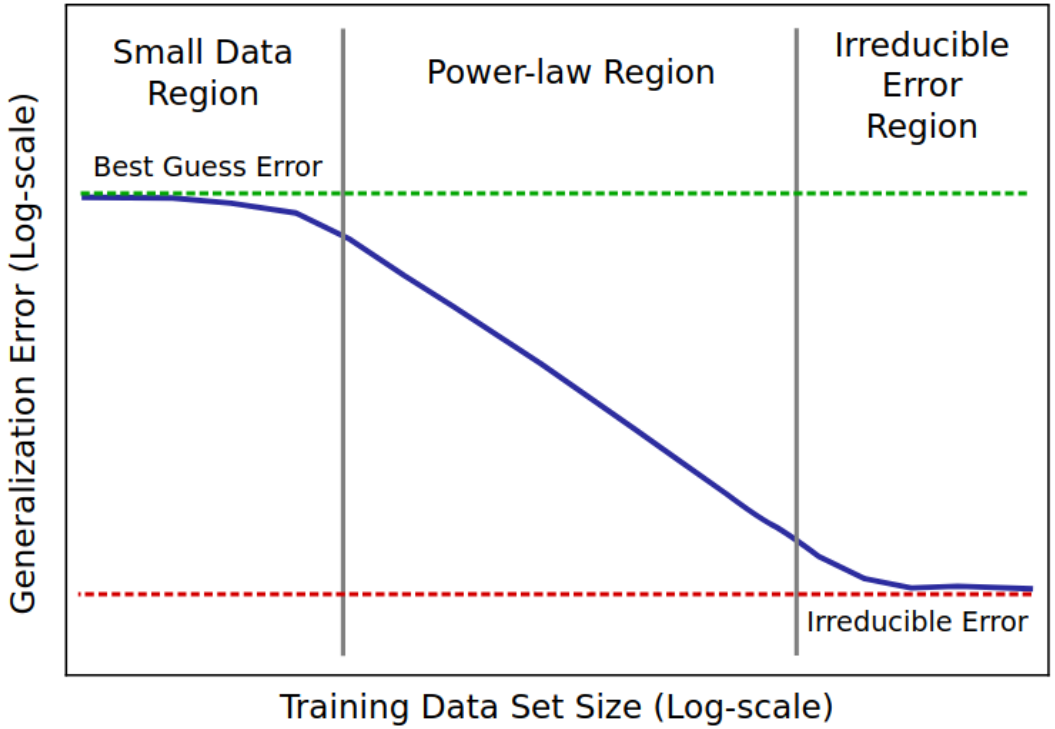
數據量受限時 $$L(D) = (D_c/D)^{\alpha_D}; \quad \alpha_D \sim 0.095, \quad D_c \sim 5.4 \times 10^{13} \text{ (tokens)} \tag{1.2}$$

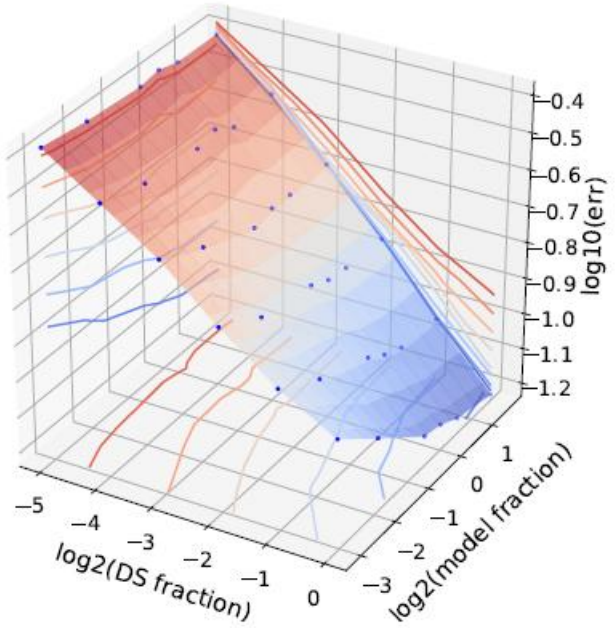
3. When training with a limited amount of compute, a sufficiently large dataset, an optimally-sized model, and a sufficiently small batch size (making optimal[3] use of compute):

計算量受限時 $$L(C_{\min}) = \left(C_c^{\min}/C_{\min}\right)^{\alpha_C^{\min}}; \quad \alpha_C^{\min} \sim 0.050, \quad C_c^{\min} \sim 3.1 \times 10^8 \text{ (PF-days)} \tag{1.3}$$

提升資料量對performance 提升最大，其次是模型參數，最後才是計算量

在各種DL領域早就提出了Scaling Law





(a) Error landscape when scaling depth (at constant baseline width).

Scaling Laws for Deep Learning

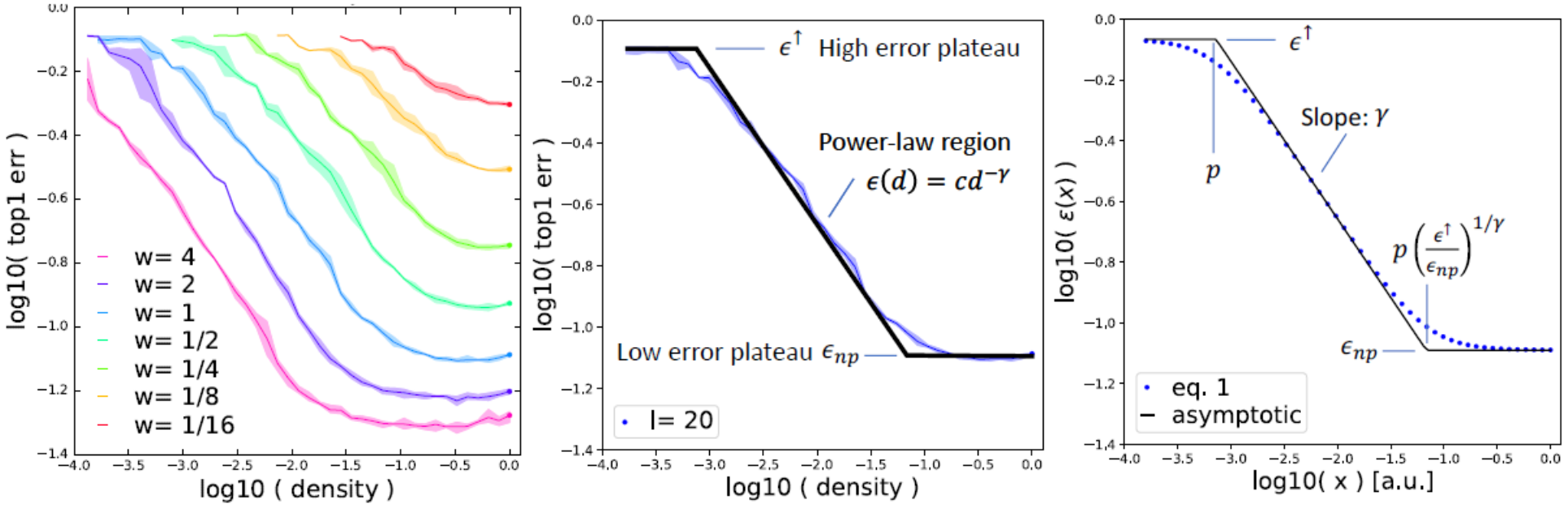


Figure 3-1: Relationship between density and error when pruning CIFAR-10 ResNets; w varies, $l = 20$, $n = N$ (left). Low-error plateau, power-law region, and high-error plateau for CIFAR-10 ResNet $l = 20$, $w = 1$, $n = N$ (center). Visualizing Equation 3.1 and the roles of free parameters (right).

ViT

Double-saturating power law

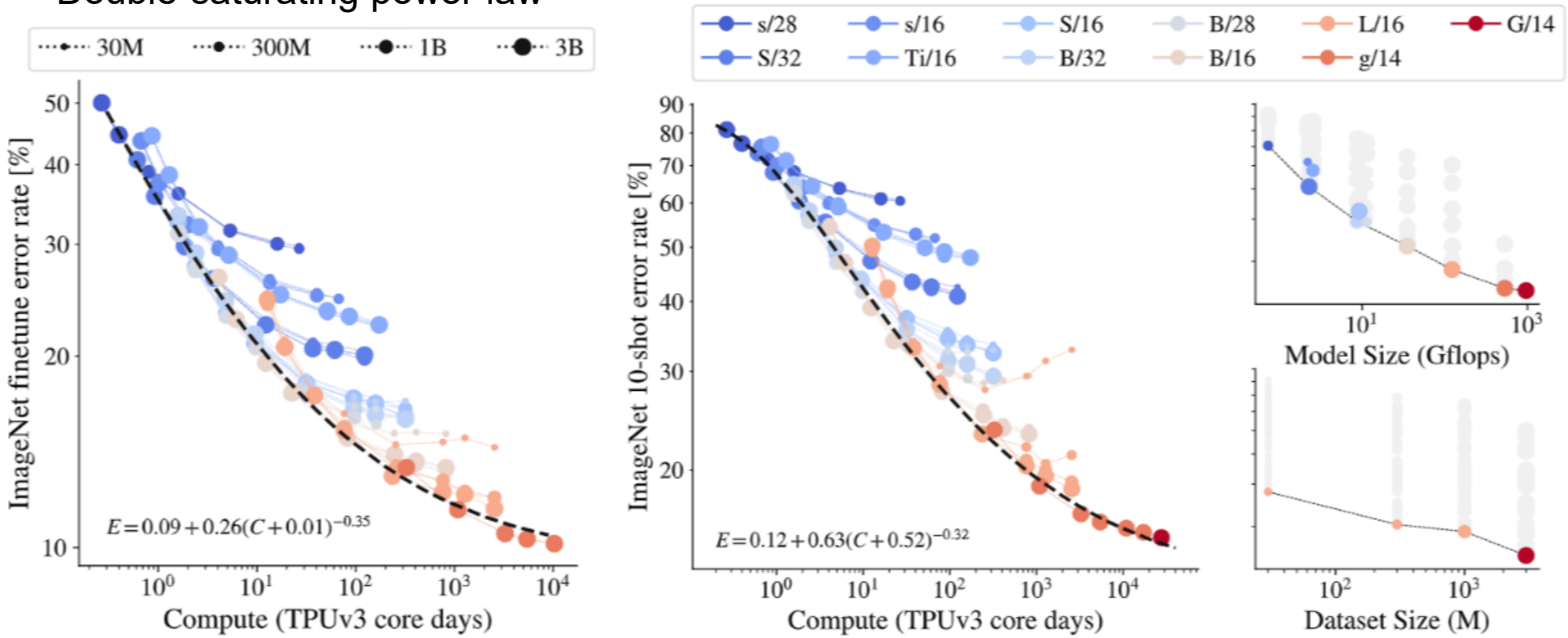
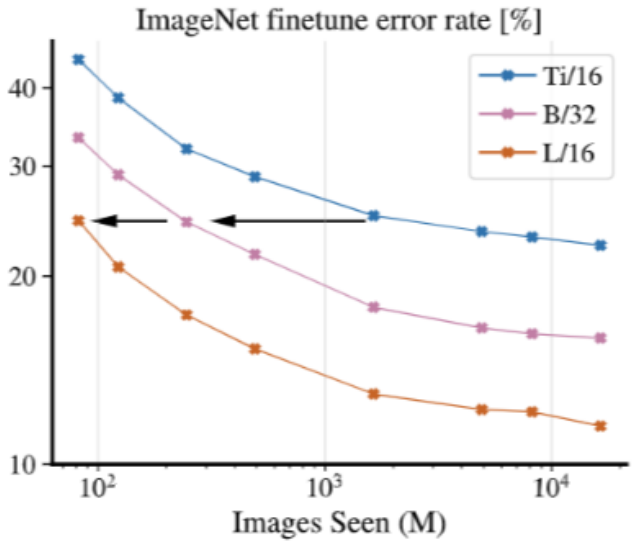
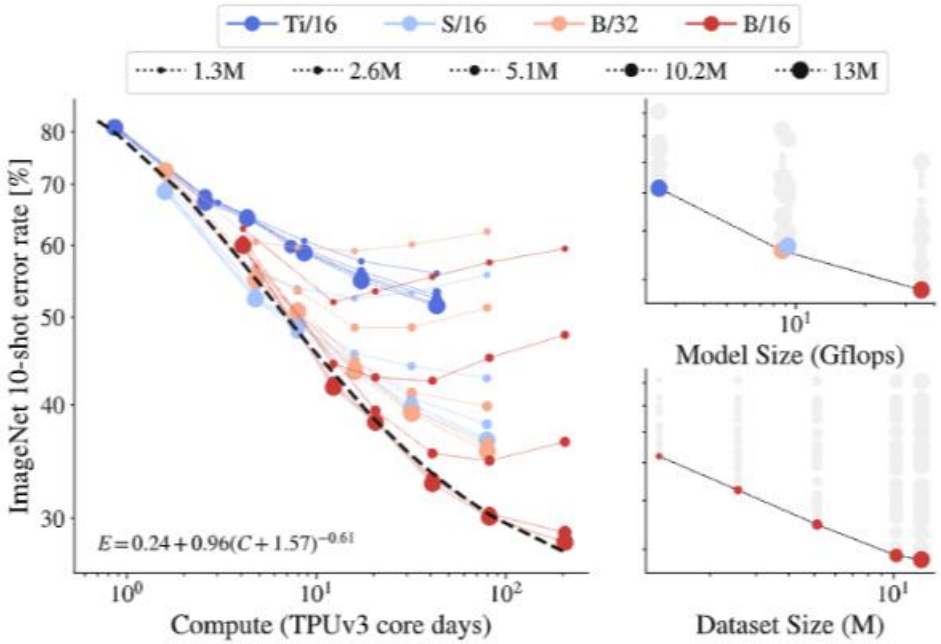


Figure 2. **Left/Center**: Representation quality, measured as ImageNet finetune and linear 10-shot error rate, as a function of total training compute. A saturating power-law approximates the Pareto frontier fairly accurately. Note that smaller models (blue shading), or models trained on fewer images (smaller markers), saturate and fall off the frontier when trained for longer. **Top right**: Representation quality when bottlenecked by model size. For each model size, a large dataset and amount of compute is used, so model capacity is the main bottleneck. Faintly-shaded markers depict sub-optimal runs of each model. **Bottom Right**: Representation quality by datasets size. For each dataset size, the model with an optimal size and amount of compute is highlighted, so dataset size is the main bottleneck.

bigger models are more sample efficient



scaling laws still apply on fewer images



要達到相同error rate, 大模型比較容易

Table 1: ViT-22B model architecture details.

Name	Width	Depth	MLP	Heads	Params [M]
ViT-G	1664	48	8192	16	1843
ViT-e	1792	56	15360	16	3926
ViT-22B	6144	48	24576	48	21743

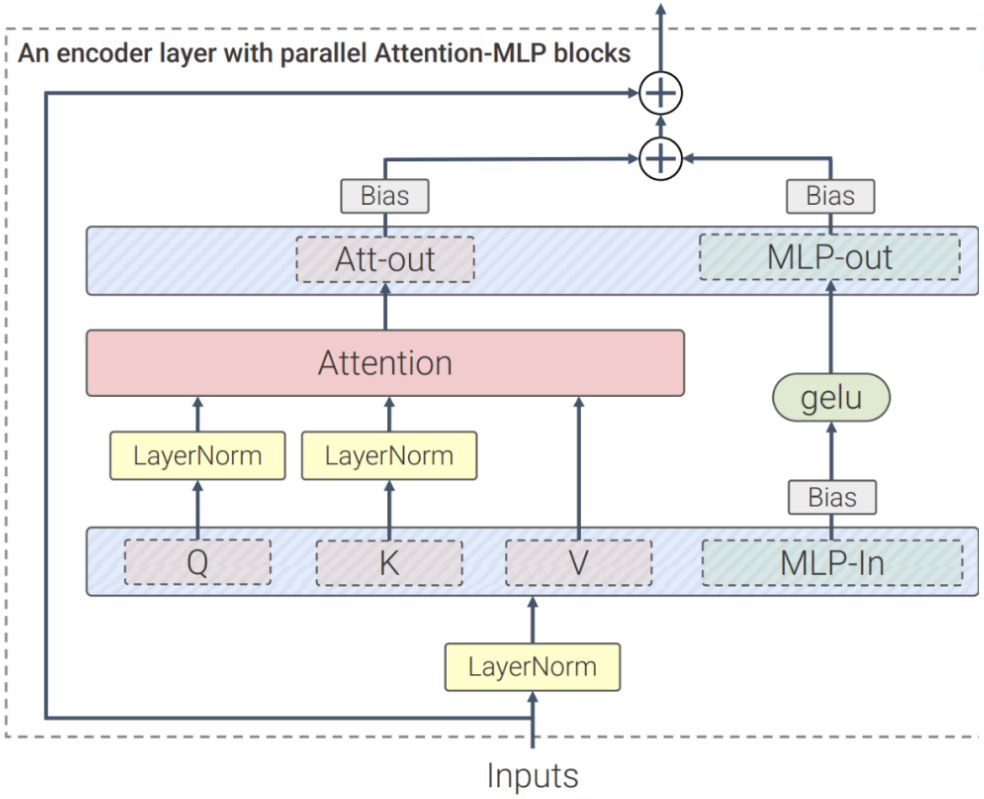


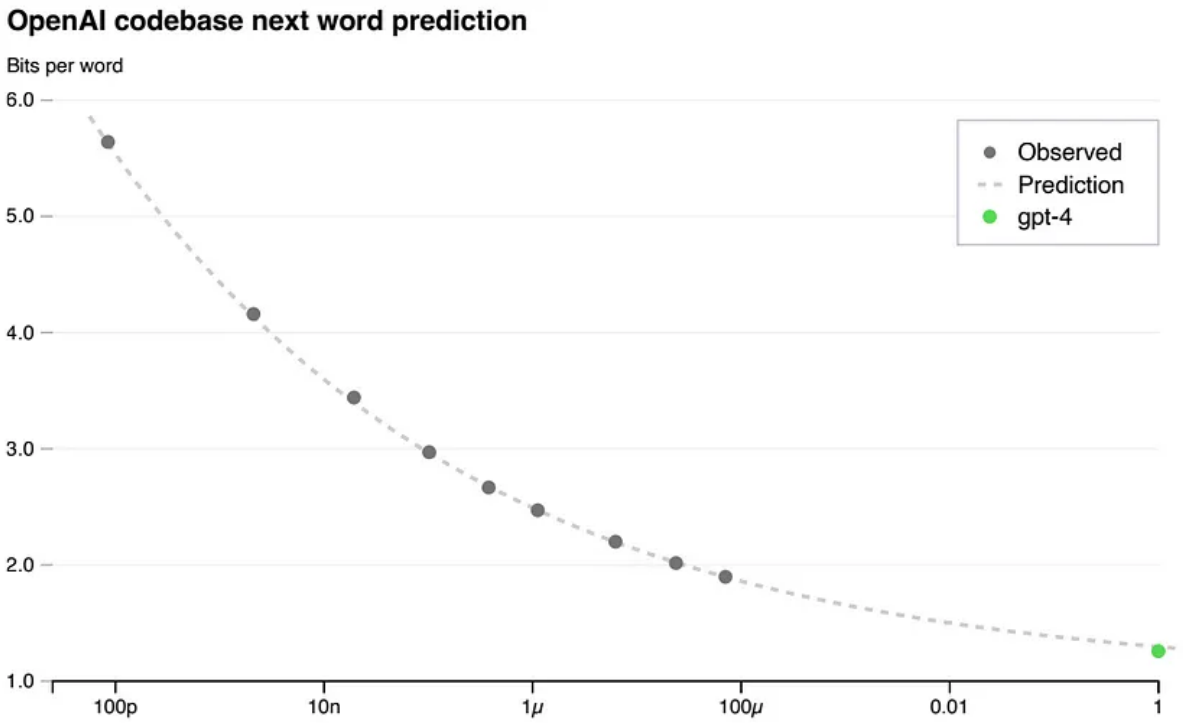
Figure 2: Parallel ViT-22B layer with QK normalization.

Table 2: Linear evaluation on ImageNet-1k (Deng et al., 2009) with varying scale. All models pre-train
large datasets. Performances of a few high-resolution fine-tuned models from are provided for refere

Model	IN	ReaL	INv2	ObjectNet	IN-R	IN-A
224px linear probe (frozen)						
B/32	80.18	86.00	69.56	46.03	75.03	31.2
B/16	84.20	88.79	75.07	56.01	82.50	52.67
ALIGN (360px)	85.5	-	-	-	-	-
L/16	86.66	90.05	78.57	63.84	89.92	67.96
g/14	88.51	90.50	81.10	68.84	92.33	77.51
G/14	88.98	90.60	81.32	69.55	91.74	78.79
e/14	89.26	90.74	82.51	71.54	**94.33**	81.56
22B	**89.51**	**90.94**	**83.15**	**74.30**	94.27	**83.80**
High-res fine-tuning						
L/16	88.5	90.4	80.4	-	-	-
FixNoisy-L2	88.5	90.9	80.8	-	-	-
ALIGN-L2	88.64	-	-	-	-	-
MaxViT-XL	89.53	-	-	-	-	-
G/14	90.45	90.81	83.33	70.53	-	-
e/14	90.9	91.1	84.3	72.0	-	-

Scaling Laws for Neural Language Models

- 搞懂Scaling Law能夠幫助我們在訓練前預測模型能力
- Having a sense of the capabilities of a model before training can improve decisions around alignment, safety, and deployment.



from OpenAI GPT4 technical report. 從此圖可以看出他們在train GPT4之前，就已經完美的預測到了GPT4的能力水平

Chinchilla Scaling Law (DeepMind)

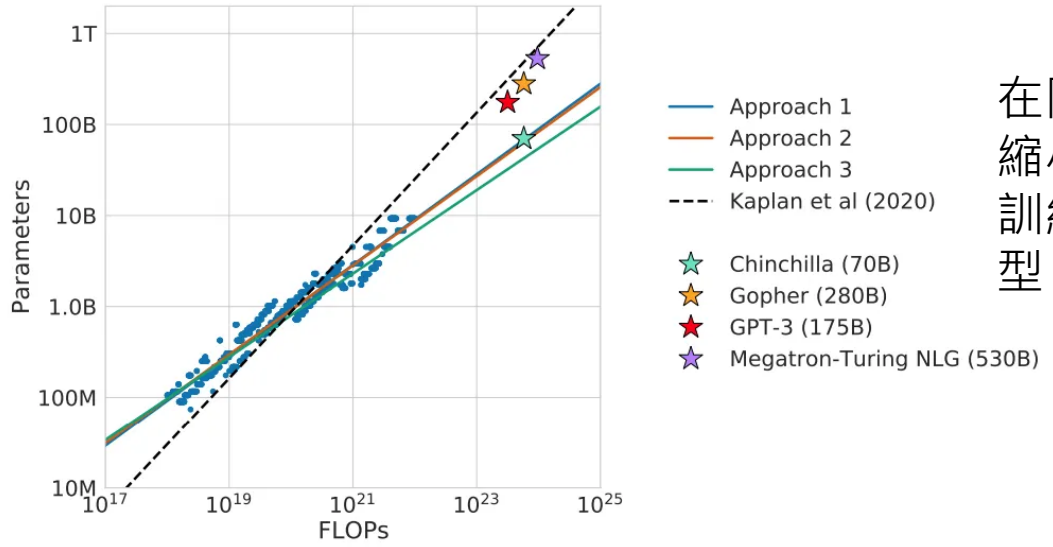
- 依據OpenAI scaling law 想法，衍伸出兩個很重要的問題
  - Return（收益）
    - 在固定的訓練計算量之下，我們所能得到的最好性能是多好？
  - Allocation（分配）
    - 我們要怎麼分配我們的模型參數量跟Dataset大小。
    - （假設計算量 = 參數量 * Dataset size，我們要大模型 * 少量data、中模型 * 中量data、還是小模型 * 大量data?）

Chinchilla Scaling Law (DeepMind)

$$L(N, D) = E + \frac{A}{N^{0.34}} + \frac{B}{D^{0.28}}, \qquad E = 1.69, A = 406.4, B = 410.7$$

- N是模型參數量、D是數據量、其他都是係數

- 數據量（Tokens數）應該要約等於模型參數量的20倍
- 並且數據量跟模型參數量要同比放大（Ex: 模型放大一倍，數據也要跟著增加一倍）



在同樣計算量下，Chinchilla把模型縮小、但是加入更多Data，馬上就訓練出比當時其他模型都更好的模型。

Other Scaling Law

- Language models scale reliably with over-training and on downstream tasks (Llama 1/2)
  - Computation bound is for ultra large model
  - 7B, 13B model can be better with over-training，用更多的資料訓練小模型
- 藉由Data Quality提升Scaling Law斜率 (Phi)
  - 藉由更好的資料，來讓模型在同樣大小參數、Dataset的情況下得到更好的效果，而最終他們推出phi-2，用2.7B model贏過大量7B, 13B模型。
- Instruction tuning的Scaling law
  - instruction tuning階段也有scaling law，因此instruction tuning的data數量還是一個簡單且重要的考量依據
- Mixture of Expert LM的Scaling Law
  - Expert count也會跟loss呈現Scaling關係，其中發現當我們有8個Experts的時候，一個模型大概可以跟兩倍大小的模型有差不多的performance。

Textbooks Are All You Need

Scaling instruction-finetuned language models.

Other Scaling Law

- Test time compute
  - 使用額外的推理時間計算自動生成改進的模型輸出
  - 測試時間計算和預訓練計算並非一對一的「可交換」關係。對於簡單的問題或推論需求較低的情況，測試時間計算可以彌補額外預訓練的不足。然而，對於具有挑戰性的問題或推論需求較高的情況，增加預訓練可能更有效。
    - 小模型多給點推理時間計算量，對簡單問題效果可以和較大模型類似，但複雜問題還是要大模型

https://arxiv.org/pdf/2408.03314

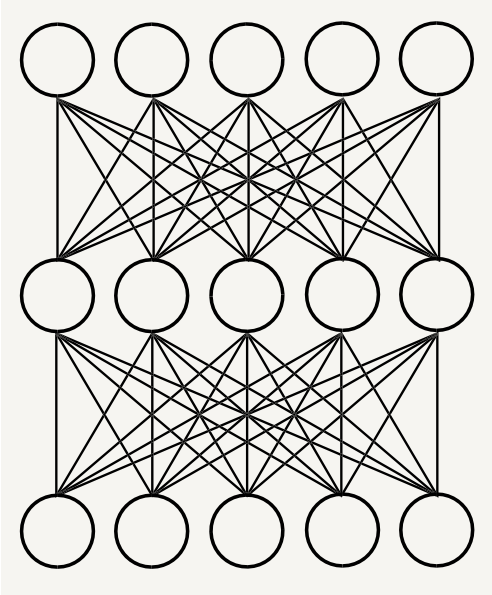
Grokking頓悟

Train a neural network to learn binary operations

★	a	b	c	d	e
a	a	d	?	c	d
b	c	d	d	a	c
c	?	e	d	b	d
d	a	?	?	b	c
e	b	b	c	?	a

$$a \circ b = c$$

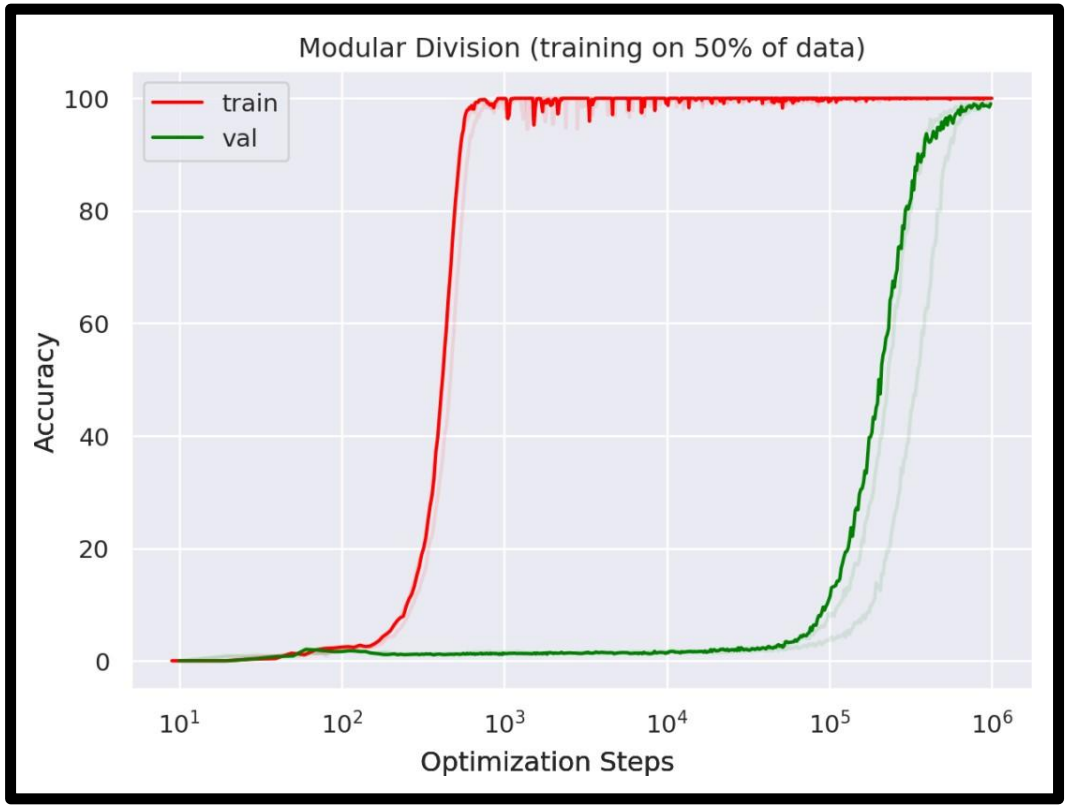
Logits for a, b, c, …



a b

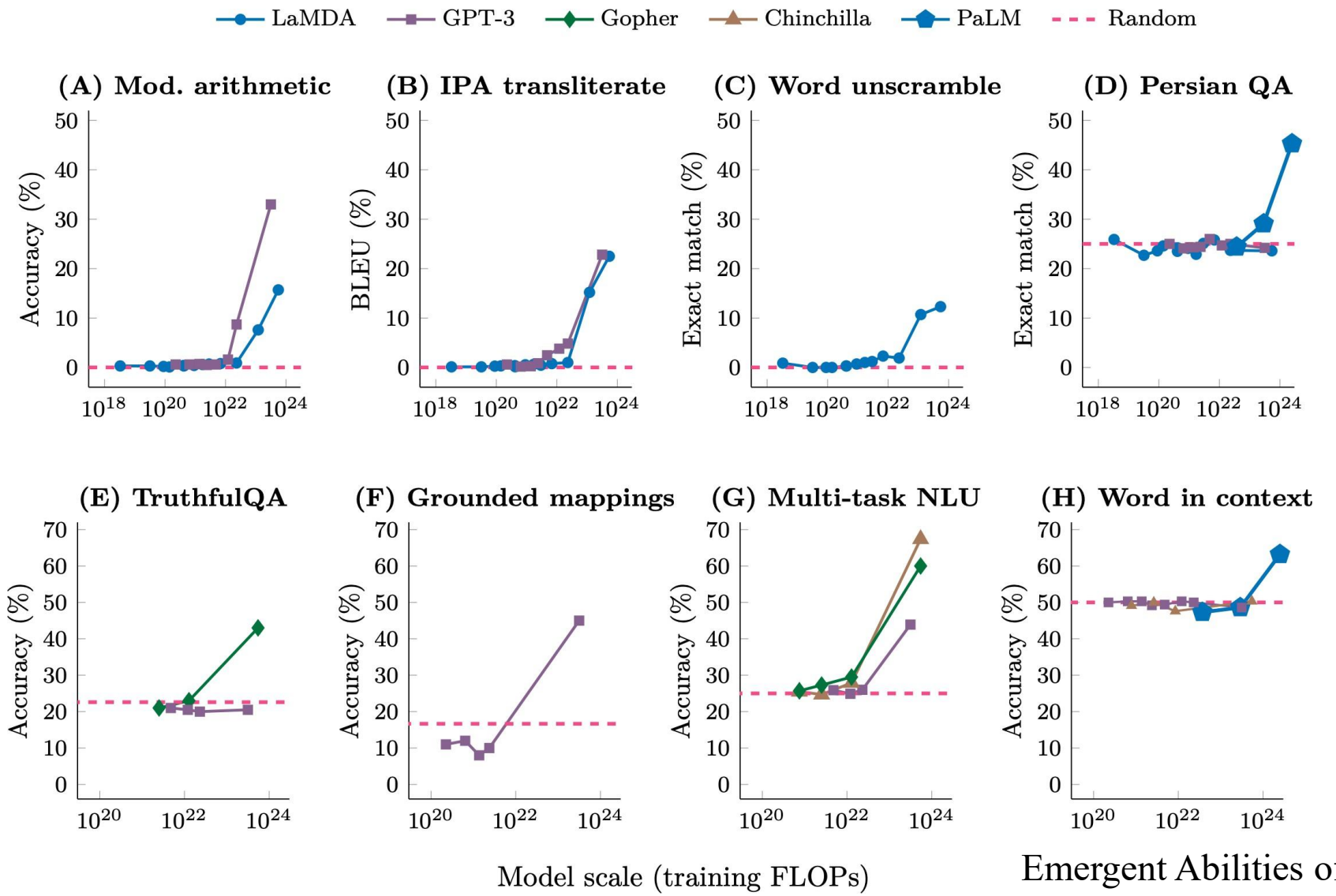
Trainable Embeddings

Phase transition behavior



https://arxiv.org/abs/2201.02177

Emergent abilities (EA)



arXiv: 2206.07682

Emergent Abilities of Large Language Models

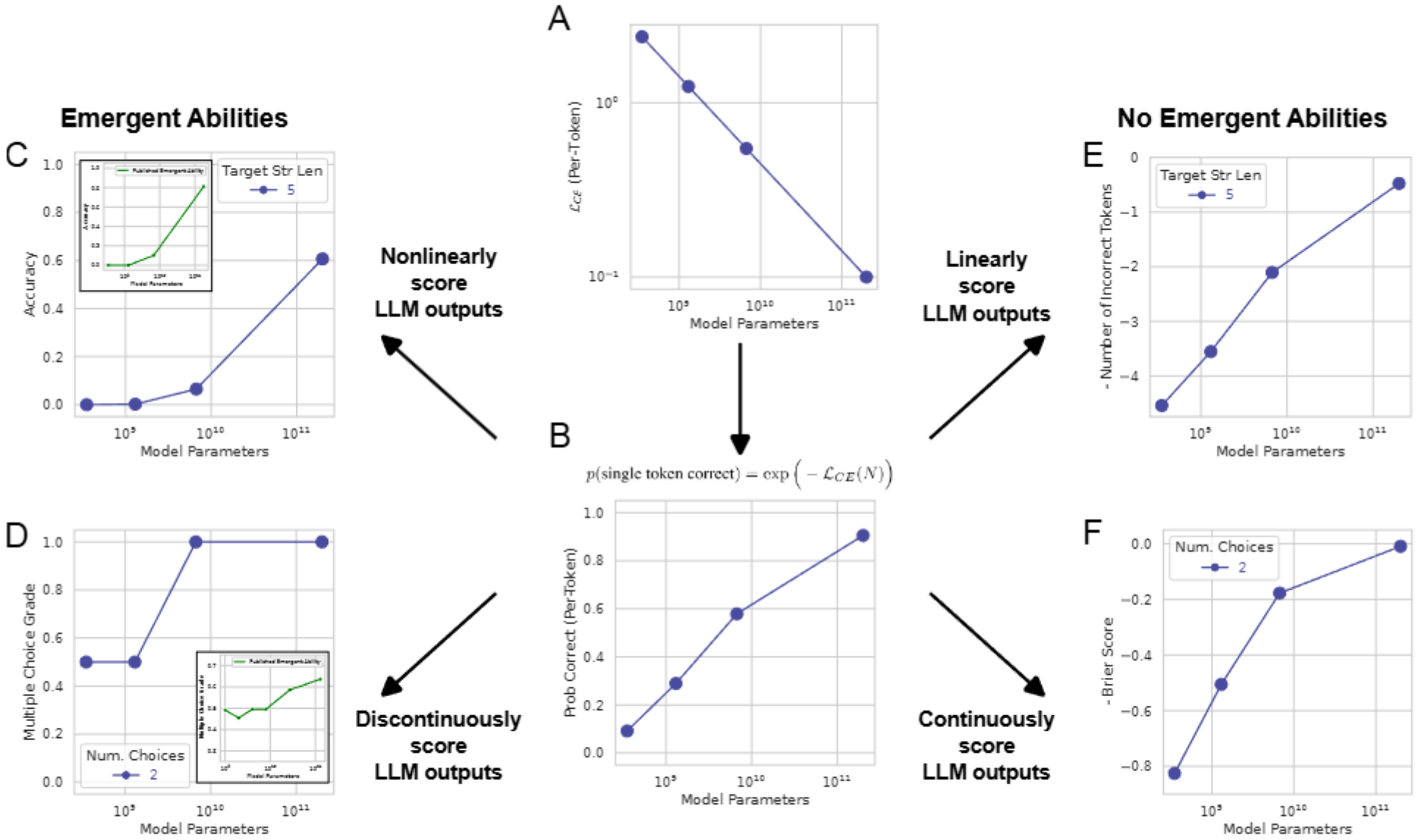


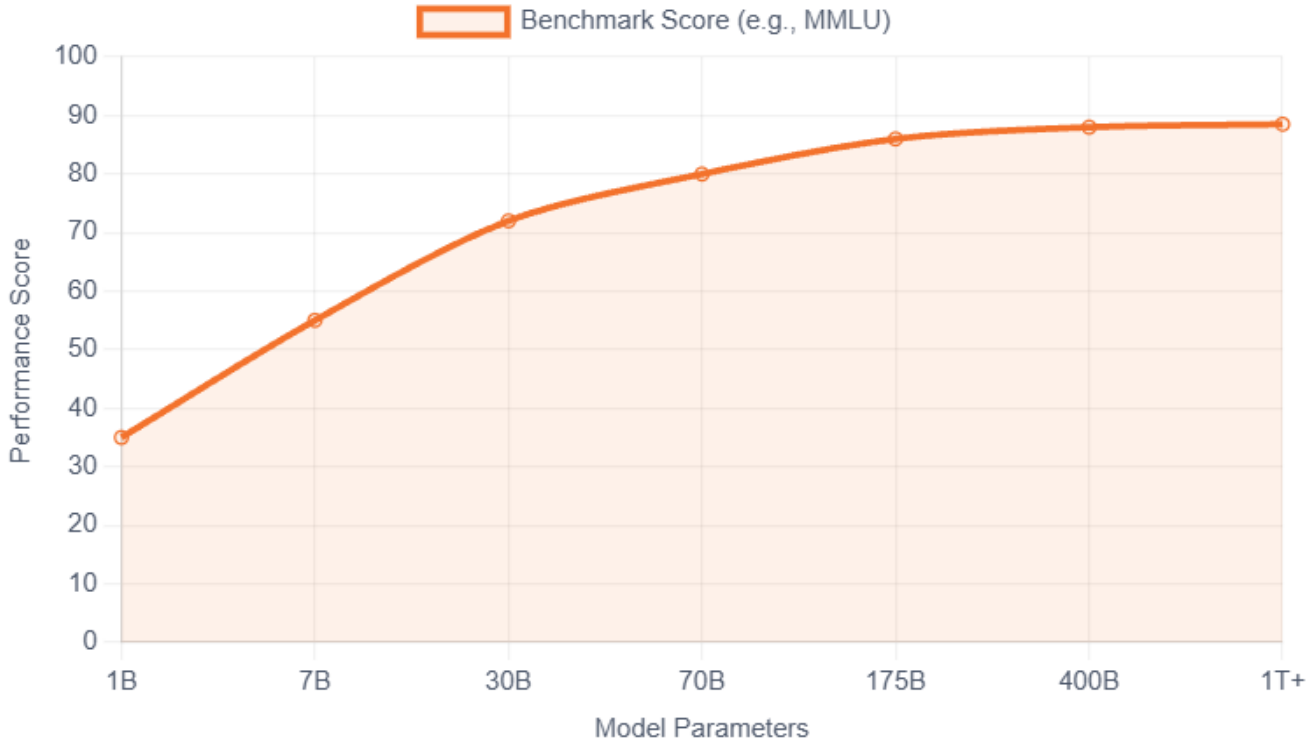
Figure 2: **Emergent abilities of large language models are created by the researcher's chosen metrics, not unpredictable changes in model behavior with scale.** (A) Suppose the per-token

https://arxiv.org/abs/2304.15004

Beyond “Bigger is Better”
2025 New Trends

The End of an Era: Hitting the Wall

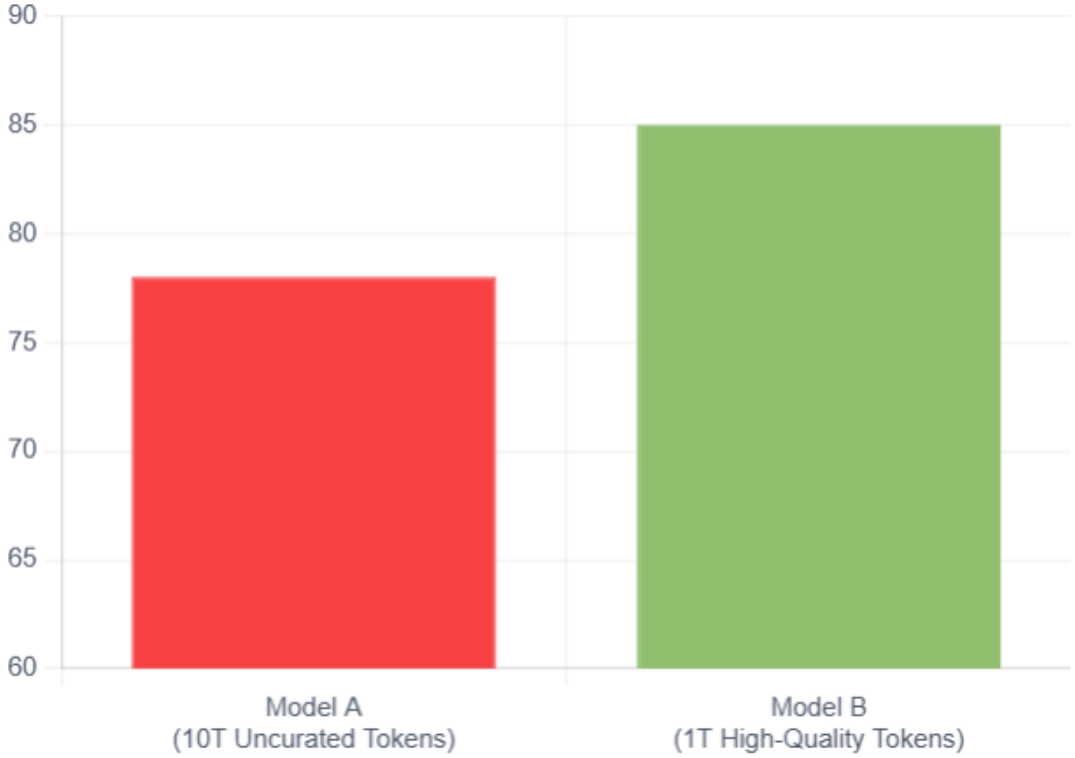
Before 2024, the "Chinchilla" scaling laws dominated: model performance scaled predictably with more compute, data, and parameters. The primary strategy was simple: build bigger models. However, recent research highlights significant diminishing returns. Doubling model size no longer guarantees a proportional leap in capability, forcing a pivot in research.



Performance on benchmarks like MMLU begins to plateau at massive parameter counts, while training costs skyrocket.

New Factor 1: Data Quality > Quantity

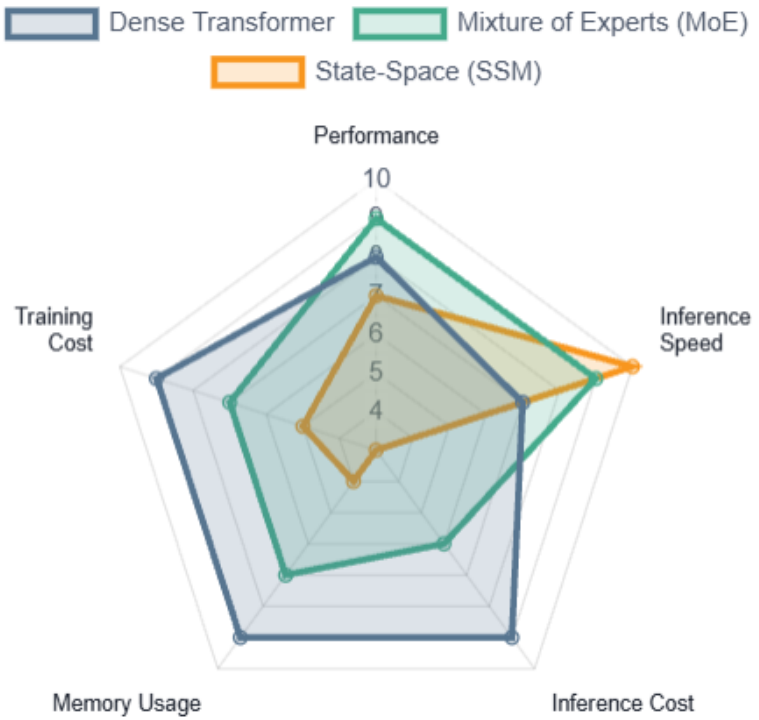
The new consensus is that data quality is a more potent scaling vector than raw data quantity. We've exhausted much of the high-quality web text. Models trained on smaller, meticulously curated, or high-quality synthetic datasets (e.g., Phi-3) are now outperforming models many times their size that were trained on unfiltered web scrapes.



A smaller model with high-quality data can surpass a larger model trained on vast, uncurated data.

New Factor 2: Architecture Matters

Scaling is no longer just about the number of parameters; it's about *how* they are used. Architectures like Mixture of Experts (MoE) and State-Space Models (SSMs) are changing the equation. MoE models decouple total parameters from active parameters, offering massive capacity with lower inference cost.



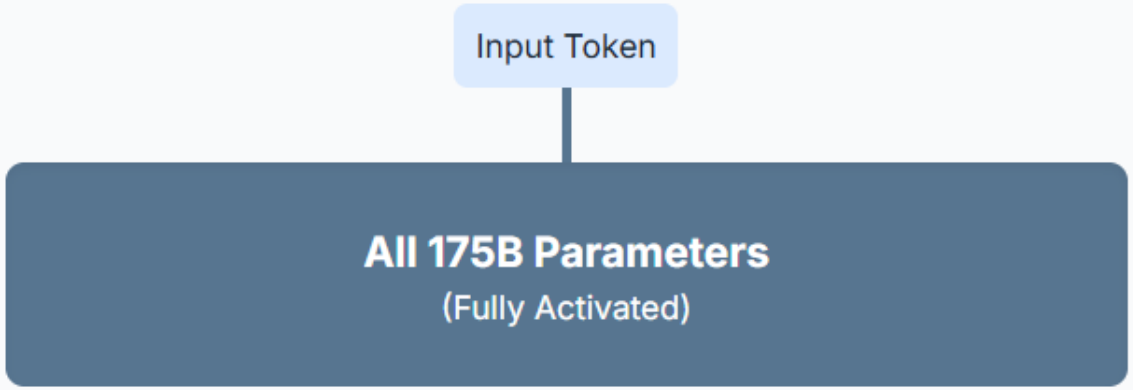
Different architectures present new trade-offs beyond simple performance, especially in inference efficiency.

Understanding the MoE Shift

A key architectural change is the move from dense models, where every token activates every parameter, to sparse Mixture of Experts (MoE) models. In an MoE, a "router" network sends each token to only a few "expert" sub-networks. This allows models to have trillions of total parameters while only using a fraction for any single computation, drastically improving training and inference efficiency.

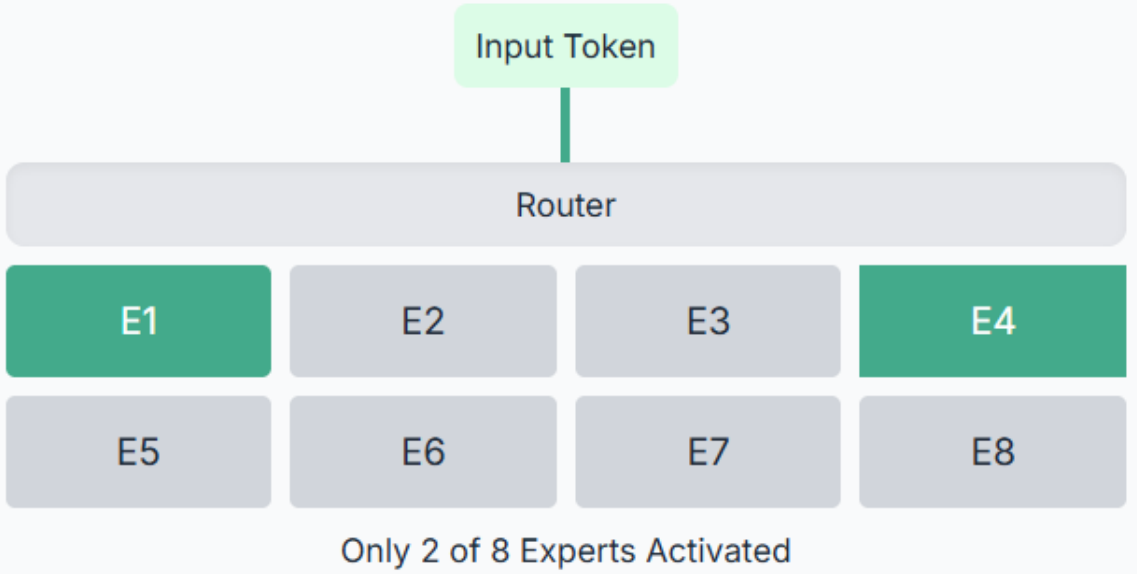
Dense Model

All parameters are activated for every input token. High cost, high consistency.



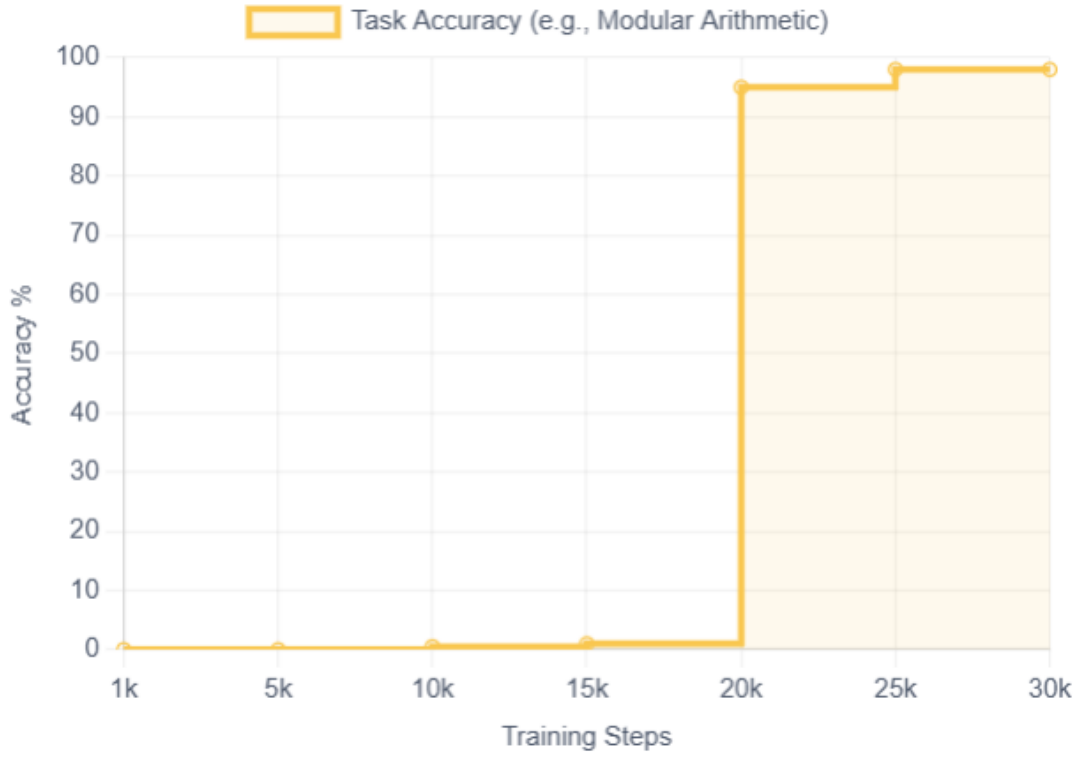
Mixture of Experts (MoE)

A router selects a few "experts" to process each token. High capacity, low cost.



New Phenomenon: Phase Transitions

Scaling is not always smooth. Research on "grokking" and phase transitions shows that capabilities can emerge suddenly and unpredictably during training. A model might show zero ability on a task for 90% of its training, then abruptly achieve near-perfect accuracy. This suggests scaling isn't just "more of the same," but about crossing critical thresholds.



Accuracy on a specific, complex task can "grok" (suddenly spike) late in training.

The New Scaling "Equation"

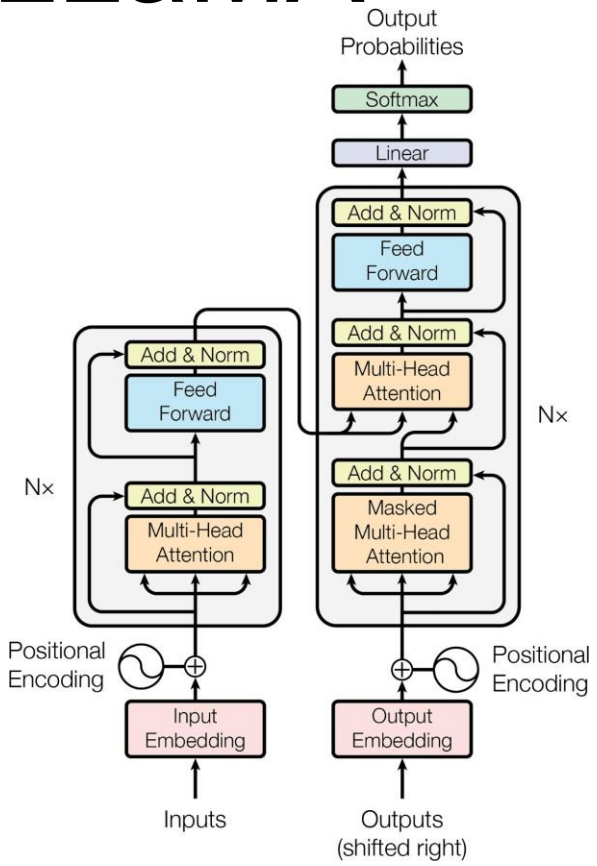
The 2024+ view of scaling is multi-dimensional. Performance is no longer just a function of compute, but a complex interplay of multiple factors.

$$\text{Performance} \approx f(\text{Data Quality}, \text{Architecture}, \text{Compute}, \text{Task Complexity}, \text{Post-Training}\ldots)$$

Future breakthroughs will come from optimizing this entire system, not just one variable.

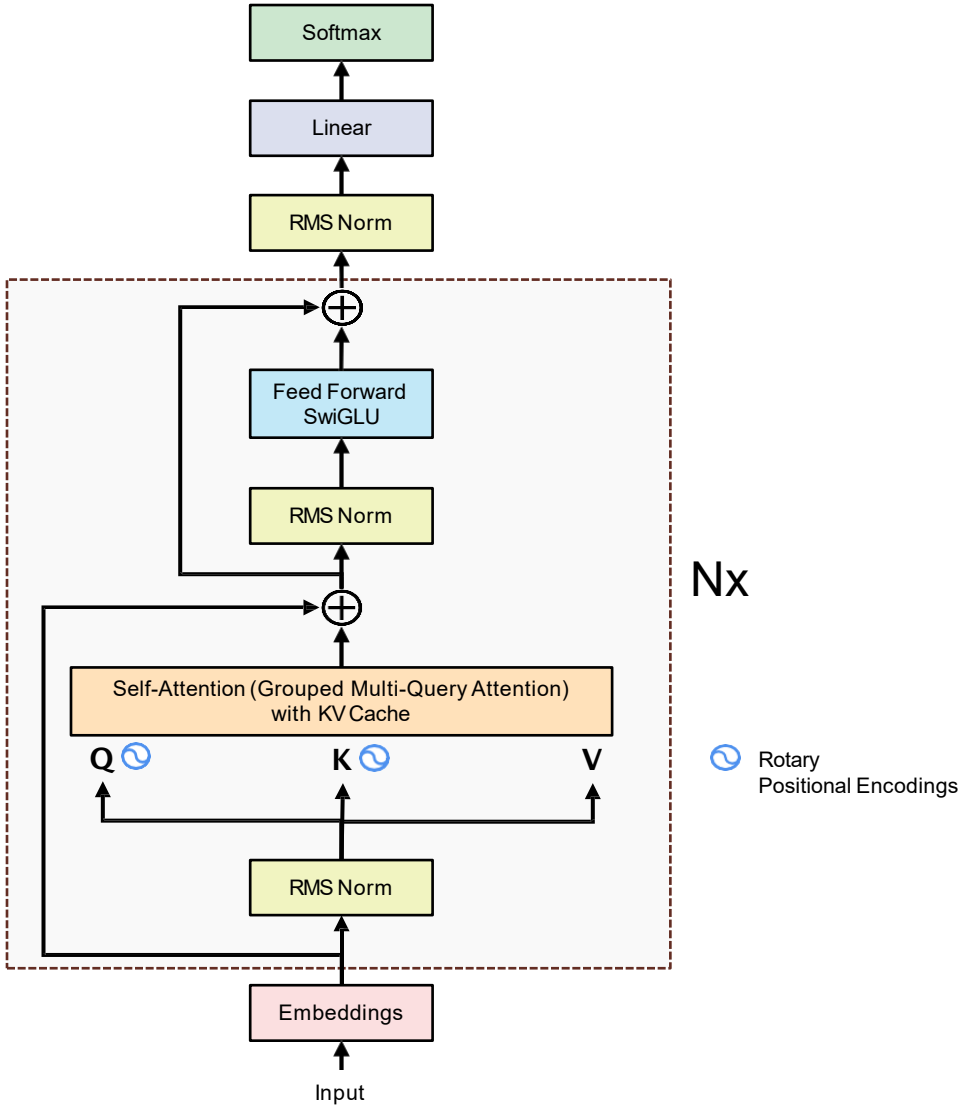
LLAMA

Transformer vs LLaMA



Transformer
("Attention is all you need")

RMSNorm (prenorm)
RoPE
Grouped MQA
FFN_SwiGLU



LLaMA

	Training Data	Params	Context Length	GQA	Tokens	LR
LLAMA 1	*See Touvron et al. (2023)*	7B	2k	✗	1.0T	3.0×10^{-4}
		13B	2k	✗	1.0T	3.0×10^{-4}
		33B	2k	✗	1.4T	1.5×10^{-4}
		65B	2k	✗	1.4T	1.5×10^{-4}
LLAMA 2	*A new mix of publicly available online data*	7B	4k	✗	2.0T	3.0×10^{-4}
		13B	4k	✗	2.0T	3.0×10^{-4}
		34B	4k	✓	2.0T	1.5×10^{-4}
		70B	4k	✓	2.0T	1.5×10^{-4}

	Training Data	Params	Context Length	GQA	Tokens
LlaMA 3	**Public Sources**	**8B**	**8K**	✓	**15T**
		70B	**8K**	✓	**15T**

模型輸入 + 輸出最多可以到多少個 Tokens

The context window (or “context length”) of a large language model (LLM) is the amount of text, in tokens, that the model can consider or “remember” at any one time. A larger context window enables an AI model to process longer inputs and incorporate a greater amount of information into each output.

https://www.icodeformybhasa.com/p/the-llama-family-of-models-model

KEY FEATURES	LLAMA 3	LLAMA 3.1	LLAMA 3.2
Parameter Sizes	8B, 70B	8B, 70B, 405B	1B, 3B (text only) 11B, 90B (vision enabled)
Token Limit	Up to 2,048 tokens	Up to 128K tokens	Varies by model, optimized for real-time
Deployment	Cloud or on-premise	Cloud or specialized hardware	Cloud, edge, or mobile environments
Use Cases	Chatbots, content generation	Decision support, complex query resolution	AR/VR, mobile apps, interactive services
Text Generation	✓	✓	✓
Advanced Reasoning	✗	✓	✓
Extended Context	✗	✓	✓
Multimodal Reasoning	✗	✗	✓
Mobile Optimization	✗	✗	✓
Real-Time Interactions	✗	✗	✓

datasciencedojo
data science for everyone

Overview of the key hyperparameters of Llama 3

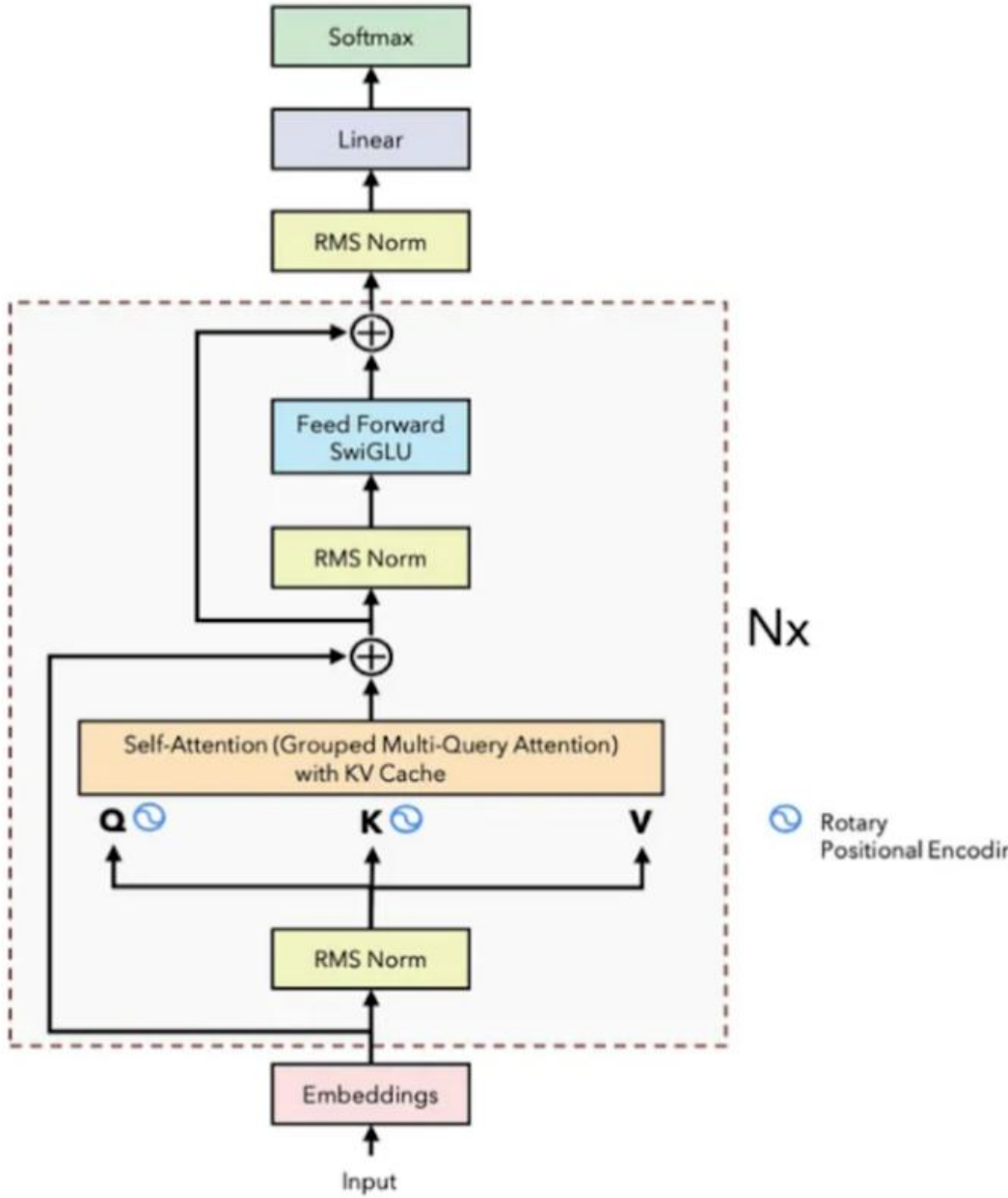
	8B	70B	405B
Layers	32	80	126
Model Dimension	4,096	8192	16,384
FFN Dimension	14,336	28,672	53,248
Attention Heads	32	64	128
Key/Value Heads	8	8	8
Peak Learning Rate	3×10^{-4}	1.5×10^{-4}	8×10^{-5}
Activation Function	SwiGLU		
Vocabulary Size	128,000		
Positional Embeddings	RoPE ($\theta = 500,000$)		

# of query heads

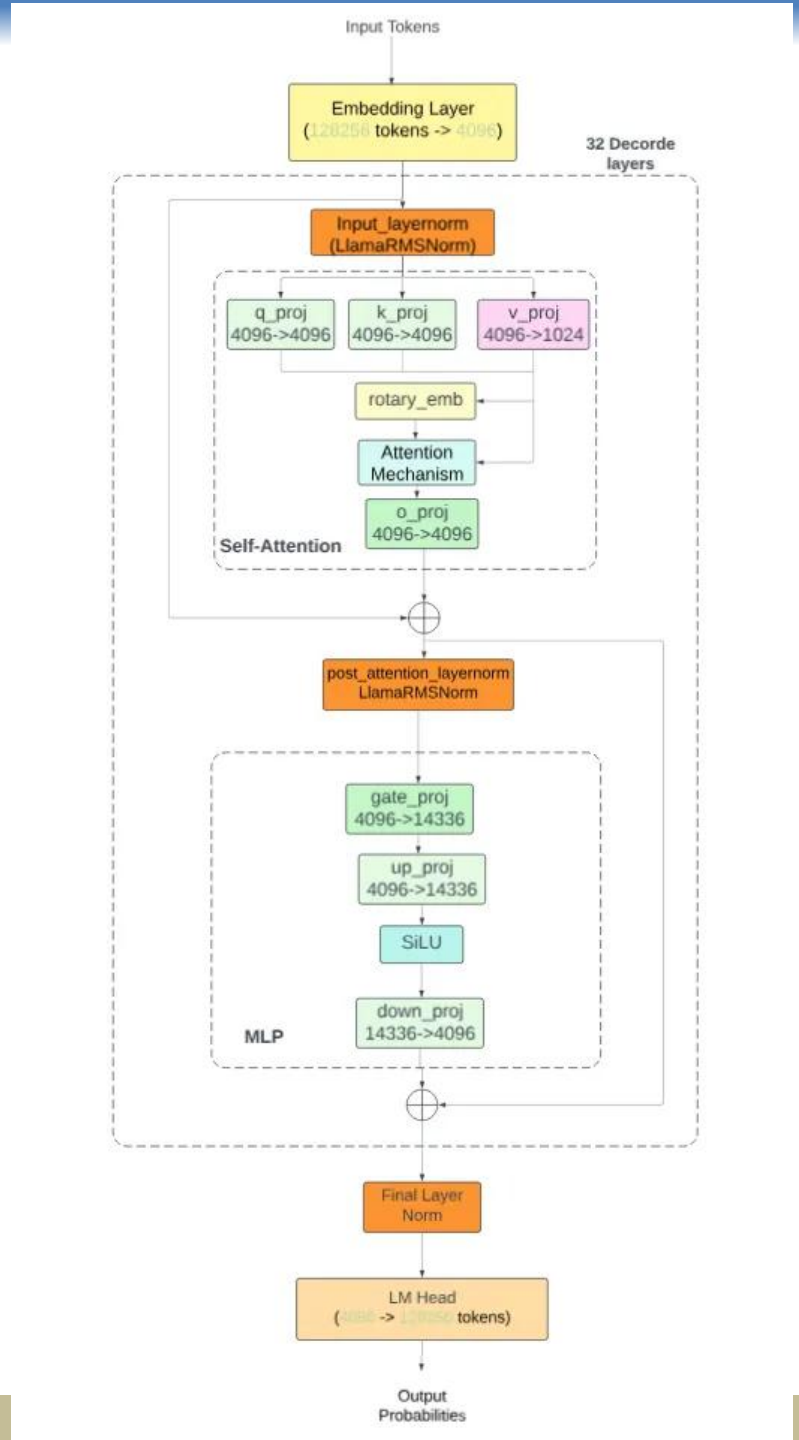
# of k/v heads

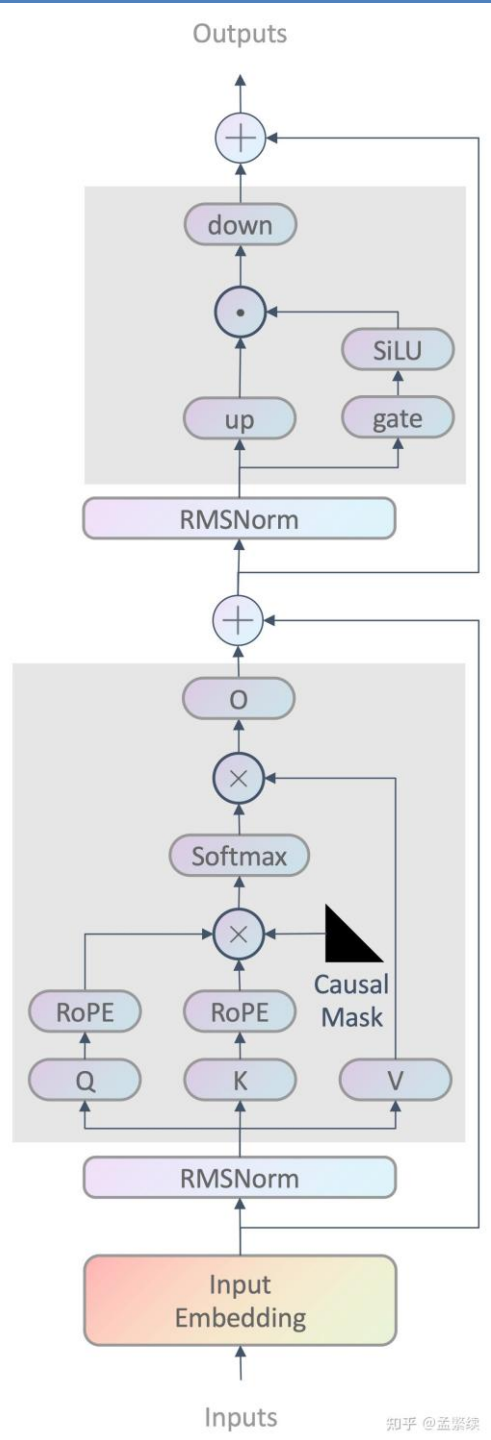
https://arxiv.org/pdf/2407.21783

Number of groups = 32 / 8 = 4



LLaMA


Input Tokens
Embedding Layer
(128256 tokens -> 4096)
32 Decorde layers
Input_layernorm (LlamaRMSNorm)
q_proj 4096->4096
k_proj 4096->4096
v_proj 4096->1024
rotary_emb
Attention Mechanism
o_proj 4096->4096
Self-Attention
post_attention_layernorm LlamaRMSNorm
gate_proj 4096->14336
up_proj 4096->14336
SiLU
down_proj 14336->4096
MLP
Final Layer Norm
LM Head
(4096 -> 128256 tokens)
Output Probabilities

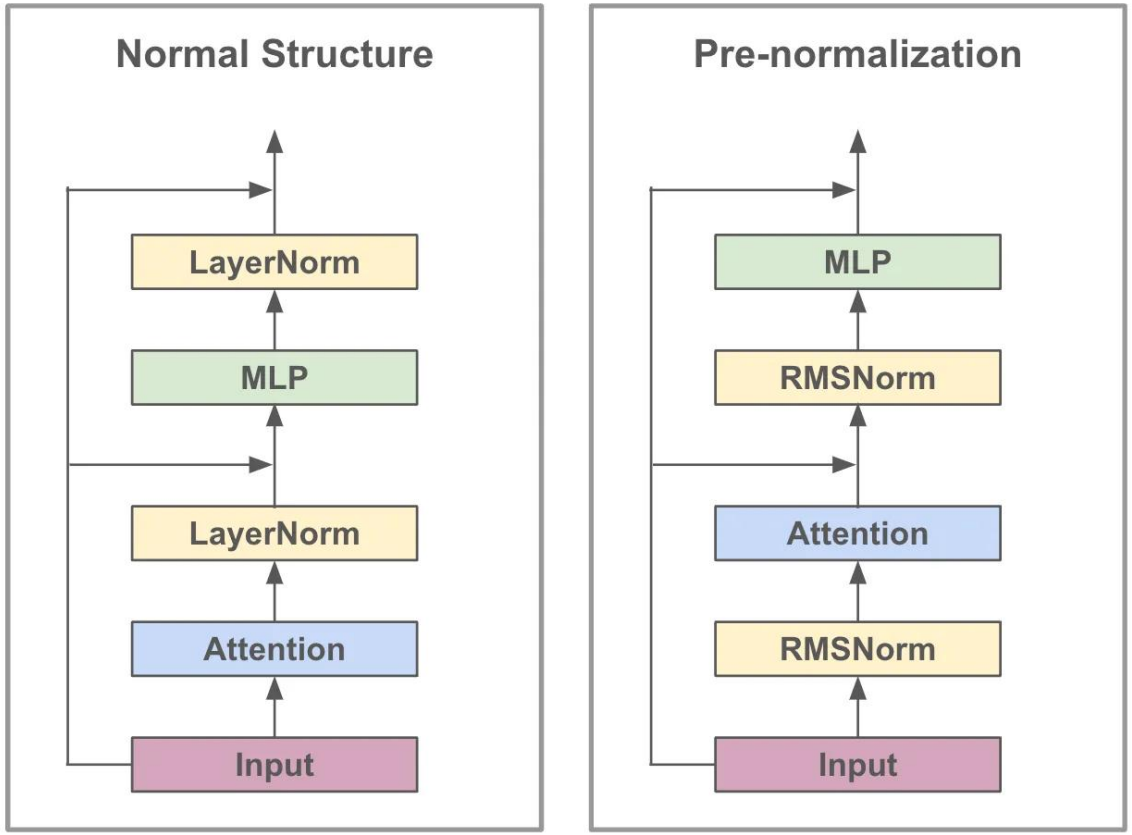

Outputs
down
SiLU
up
gate
RMSNorm
O
Softmax
Causal Mask
RoPE
RoPE
Q
K
V
RMSNorm
Input Embedding
Inputs

LLaMA

- Model modifications (vs. transformers)
  - LN => RMSNorm (pre-normalization type)
    - Better training stability
    - Lower computation and memory
      - No need for mean calculation

$$\bar{a}_i = \frac{a_i}{\text{RMS}}, \quad where \ \text{RMS} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} a_i^2}$$

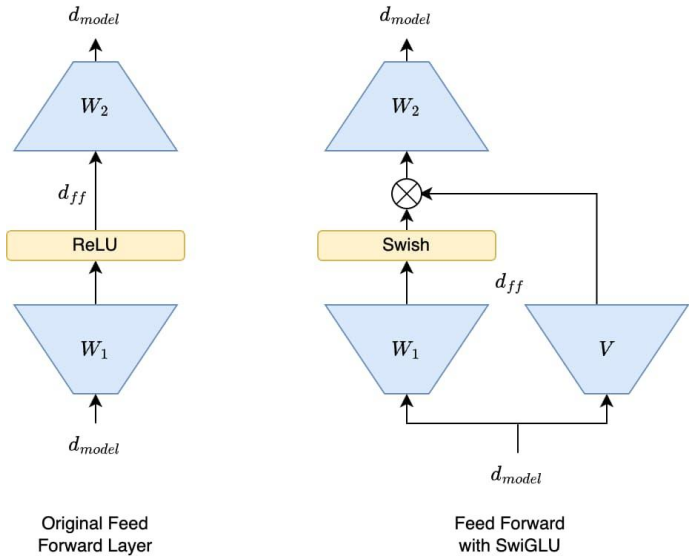
Root Mean Square Layer Normalization (RMSNorm)
可以看作LayerNorm在均值為0時的一個特例

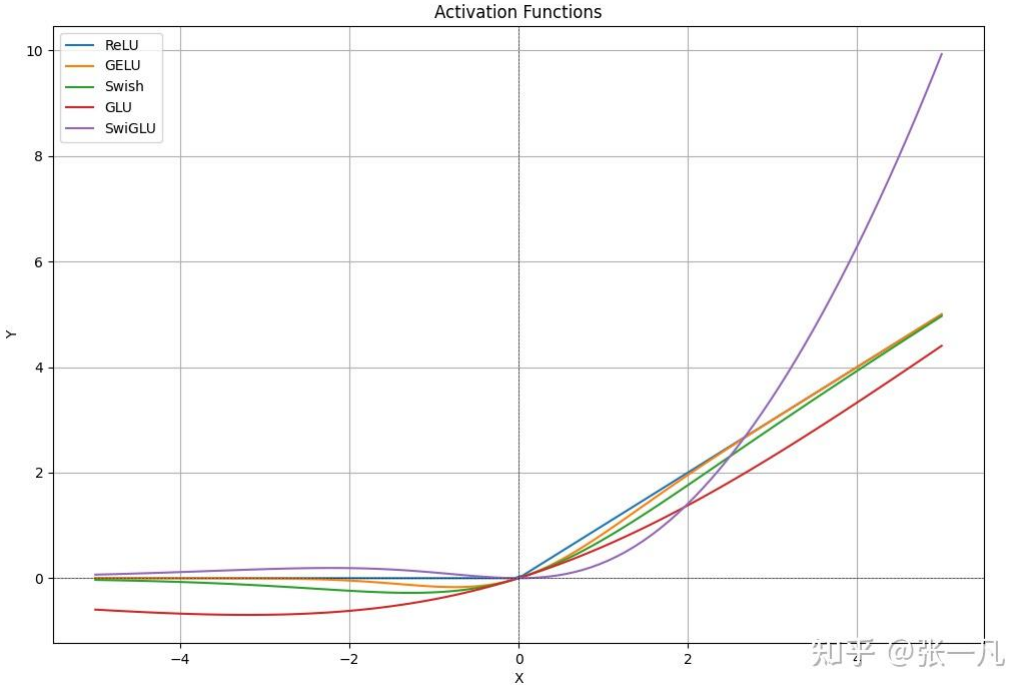


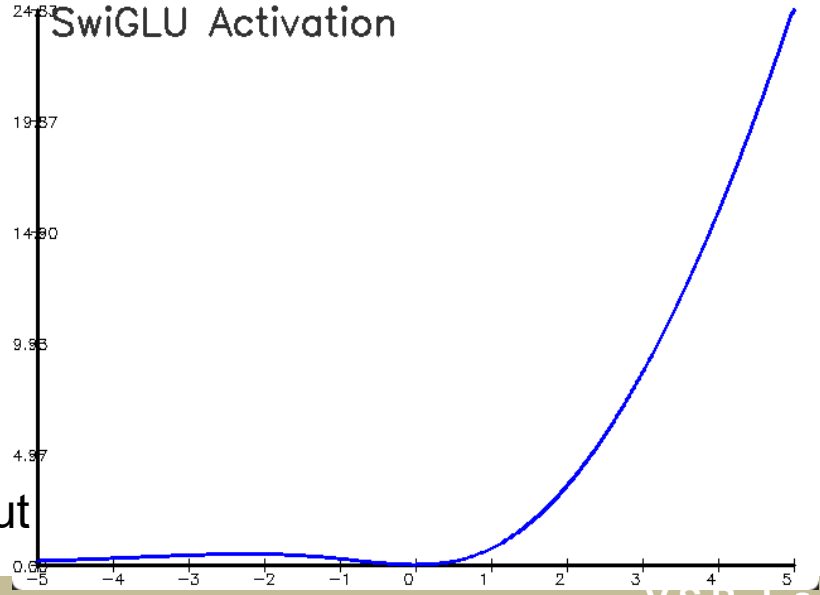
- Activation function: SwiGLU in FFN

$$\texttt{SwiGLU}(x) = \texttt{Swish}(xW) \cdot xV$$

$$\texttt{Swish}(x) = x \cdot \texttt{Sigmoid}(\beta x)$$



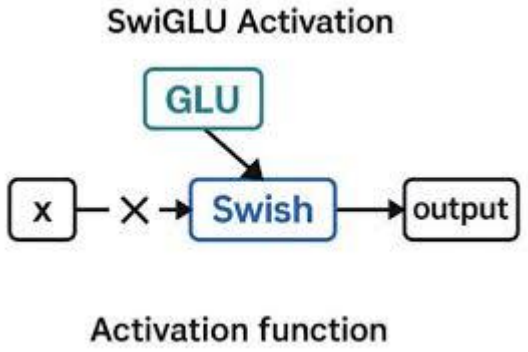




With GELU MLP: hidden size = 4d
With SwiGLU MLP: hidden size often = 2/3 * 4d ≈ 2.67d *per linear*, but there are two branches, so parameter/compute stays in the same ballpark

SwiGLU

- Activation in MHA and FFN
- SwiGLU formula
  - variant of **GLU** with **SiLU/Swish** as the gating activation
  - Classic GLU:
    - $\mathrm{GLU}(x) = (XW_1) \odot \sigma(XW_2)$
      - where σ is logistic sigmoid and ⊙ is elementwise multiply
  - **SwiGLU** replaces σ with **SiLU / Swish**:
    - $\mathrm{SwiGLU}(x) = (XW_1) \odot \mathit{SiLU}(XW_2)$
  - And SiLU (aka Swish) is:
    - $\mathrm{SiLU}(z) = z \cdot \sigma(z)$

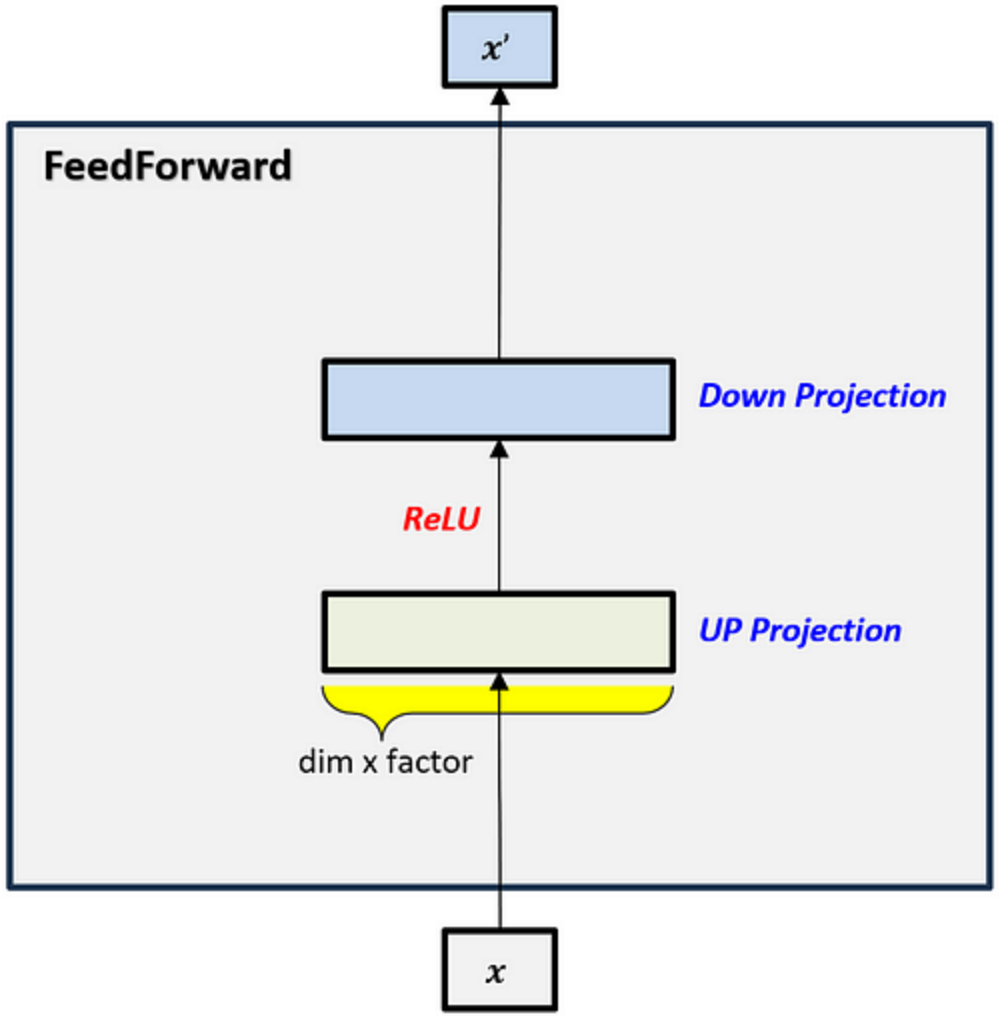



```python
a = x @ W1      # linear
b = x @ W2      # gate pre-activation
gate = b * sigmoid(b)   # SiLU(b)
y = a * gate    # elementwise product
out = y @ W3    # second linear projection
```

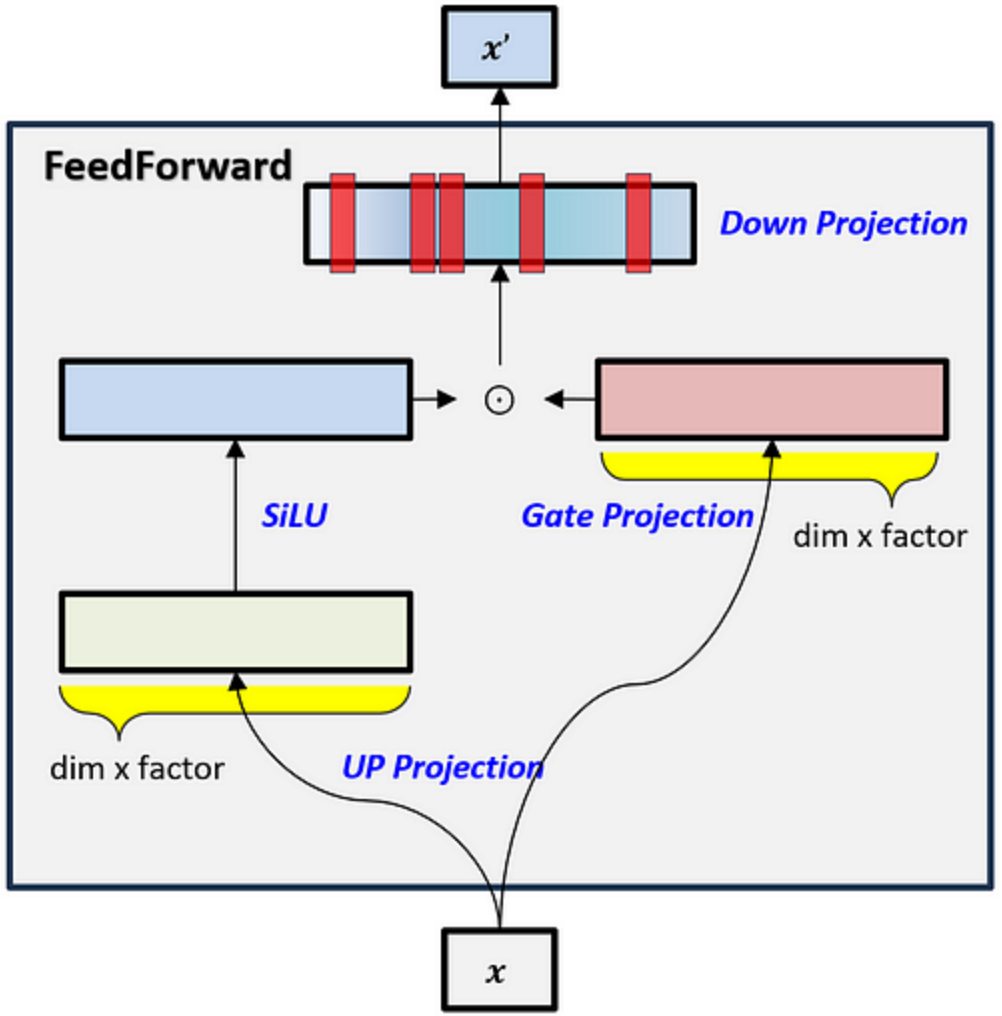
Often implementations fold W3 into the usual W_out of the MLP.

- 1、Swish對於負值的回應
  - 相對較小克服了 ReLU 某些神經元上輸出始終為零的缺點
- 2、GLU 的門控特性，
  - 這意味著它可以根據輸入的情況決定哪些資訊應該通過、哪些資訊應該被過濾。這種機制可以使網路更有效地學習到有用的表示，有助於提高模型的泛化能力。在大語言模型中，這對於處理長序列、長距離依賴的文本特別有用。
- 3、SwiGLU 中的參數
  - 可以通過訓練學習，使得模型可以根據不同任務和資料集動態調整這些參數，增強了模型的靈活性和適應性。
- 4、計算效率
  - 相比某些較複雜的啟動函數（如 GELU）更高，同時仍能保持較好的性能。

Original Transformer (2017)



Llama1 (2023)



$$down_{proj}(\text{silu}\left(up_{proj}(x)\right) * gate_{proj}(x))$$

TOKENIZATION

Tokenization（分詞）

- 文本內容處理為最小基本單元即token，用於後續的處理
- Granularity
  - Word
    - 中文句子：我喜歡看電影和讀書。
    - 分詞結果：我 | 喜歡 | 看 | 電影 | 和 | 讀書。
    - 英文句子：I enjoy watching movies and reading books.
    - 分詞結果：I | enjoy | watching | movies | and | reading | books.
    - 優點
      - 語義明確，有助於保留詞語之間的關聯性和上下文資訊
    - 缺點
      - 長尾效應和稀有詞問題，OOV（Out-of-Vocabulary），形態關係和詞綴關係
        - **長尾效應和稀有詞問題**： 詞表可能變得巨大，包含很多不常見的詞彙，增加存儲和訓練成本，稀有詞的訓練數據有限，難以獲得準確的表示。
        - **OOV（Out-of-Vocabulary）**： 詞粒度分詞模型只能使用詞表中的詞來進行處理，無法處理詞表之外的詞彙，這就是所謂的OOV問題。
        - **形態關係和詞綴關係**： 無法捕捉同一詞的不同形態，也無法有效學習詞綴在不同詞彙之間的共通性，限制了模型的語言理解能力，比如love和loves在word（詞）粒度的詞表中將會是兩個詞。

- char（字元）
  - example
    - 中文句子：我喜歡看電影和讀書。
    - 分詞結果：我 | 喜 | 歡 | 看 | 電 | 影 | 和 | 讀 | 書 | 。
    - 英文句子：I enjoy watching movies and reading books.
    - 分詞結果：I | | e | n | j | o | y | | w | a | t | c | h | i | n | g | | m | o | v | i | e | s | | a | n | d | | r | e | a | d | i | n | g | | b | o | o | k | s | .
  - 優點
    - 統一處理方式，解決OOV問題
  - 缺點
    - 語義資訊不明確，處理效率低

- subword（子詞）:
  - BERT: WordPiece
  - LLM: Byte-Pair Encoding (BPE)，Byte-level BPE（BBPE）
- Byte-Pair Encoding (BPE)
  - 逐步合併出現頻率最高的子詞對，從而構建出一個詞彙表
  - **計算初始詞表**：通過訓練語料獲得或者最初的英文中26個字母加上各種符號以及常見中文字符，這些作為初始詞表。
  - **構建頻率統計**：統計所有子詞單元對（兩個連續的子詞）在文本中的出現頻率。
  - **合併頻率最高的子詞對**：選擇出現頻率最高的子詞對，將它們合併成一個新的子詞單元，並更新詞彙表。
  - **重複合併步驟**：不斷重複步驟 2 和步驟 3，直到達到預定的詞彙表大小、合併次數，或者直到不再有有意義的合併（即，進一步合併不會顯著提高詞彙表的效益）。
  - **分詞**：使用最終得到的詞彙表對文本進行分詞。

- Suppose we have a text corpus with the following four words: “ab”, “bc”, “bcd”, and “cde”. The initial vocabulary consists of all the bytes or characters in the text corpus: {“a”, “b”, “c”, “d”, “e”}.

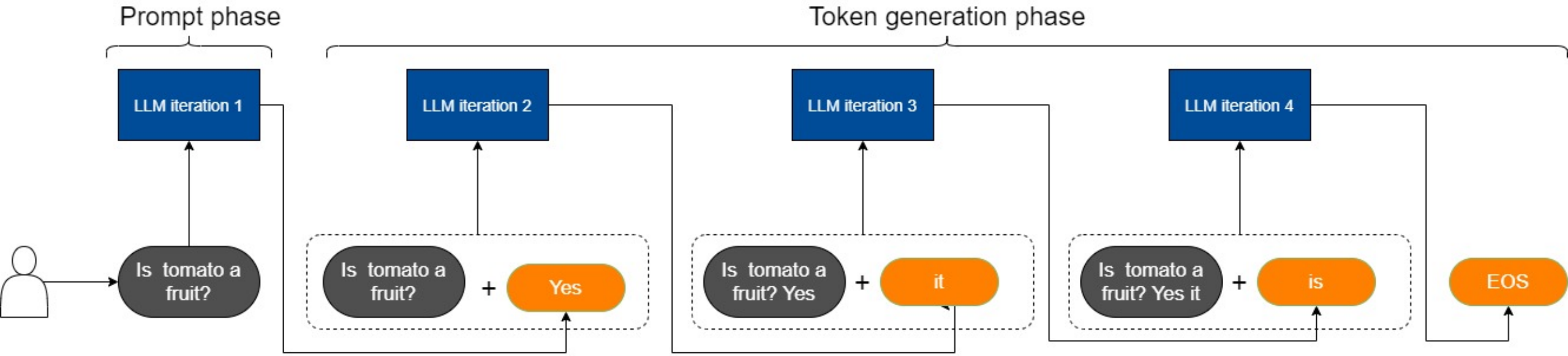
- Step 1: Initialize the vocabulary
  - Vocabulary = {"a", "b", "c", "d", "e"}
- Step 2: Calculate the frequency
  - Frequency = {"a": 1, "b": 3, "c": 3, "d": 2, "e": 1}
- Step 3a: Find the most frequent pair of two characters
  - The most frequent pair is "bc" with a frequency of 2.
- Step 3b: Merge the pair
  - Merge "bc" to create a new subword unit "bc".
- Step 3c: Update frequency counts
  - Update the frequency counts of all the bytes or characters that contain “bc”:
  - Frequency = {"a": 1, "b": 2, "c": 3, "d": 2, "e": 1, "bc": 2}
- Step 3d: Add the new subword unit to the vocabulary
  - Add “bc” to the vocabulary:
  - Vocabulary = {"a", "b", "c", "d", "e", "bc"}
- Repeat steps 3a-3d until the desired vocabulary size is reached.
- Step 4: Represent the text corpus using subword units

- The resulting vocabulary consists of the following subword units: {"a", "b", "c", "d", "e", "bc", "cd",

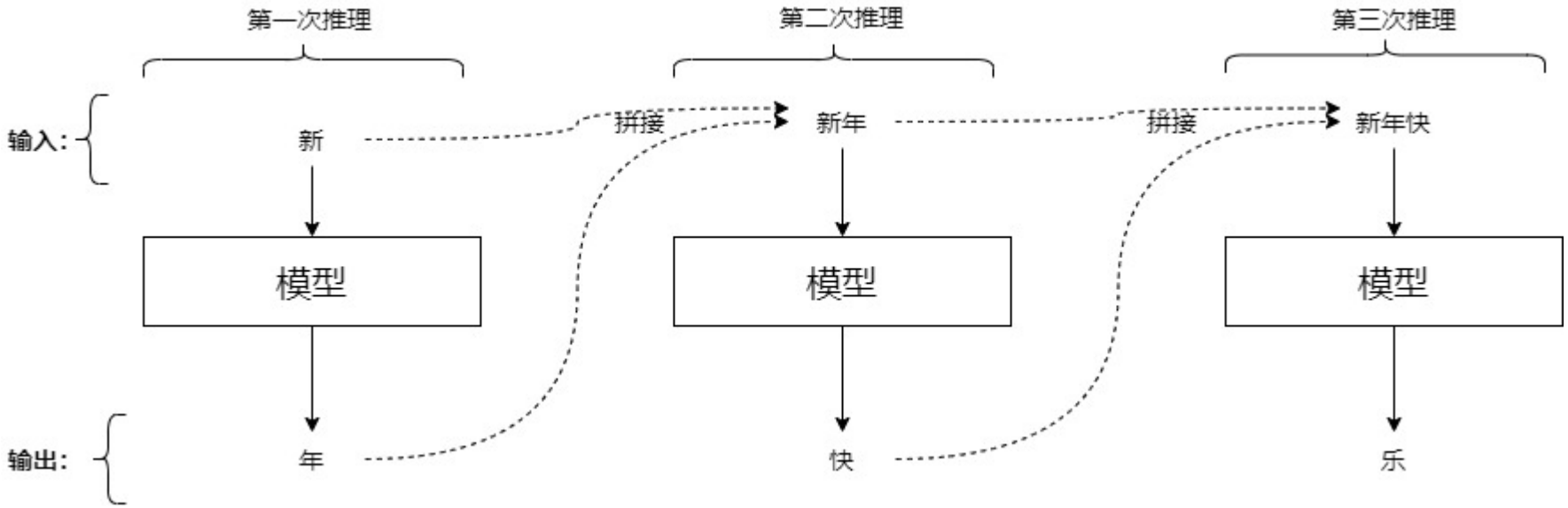
LLAMA: GQA AND KV CACHE

KV Cache

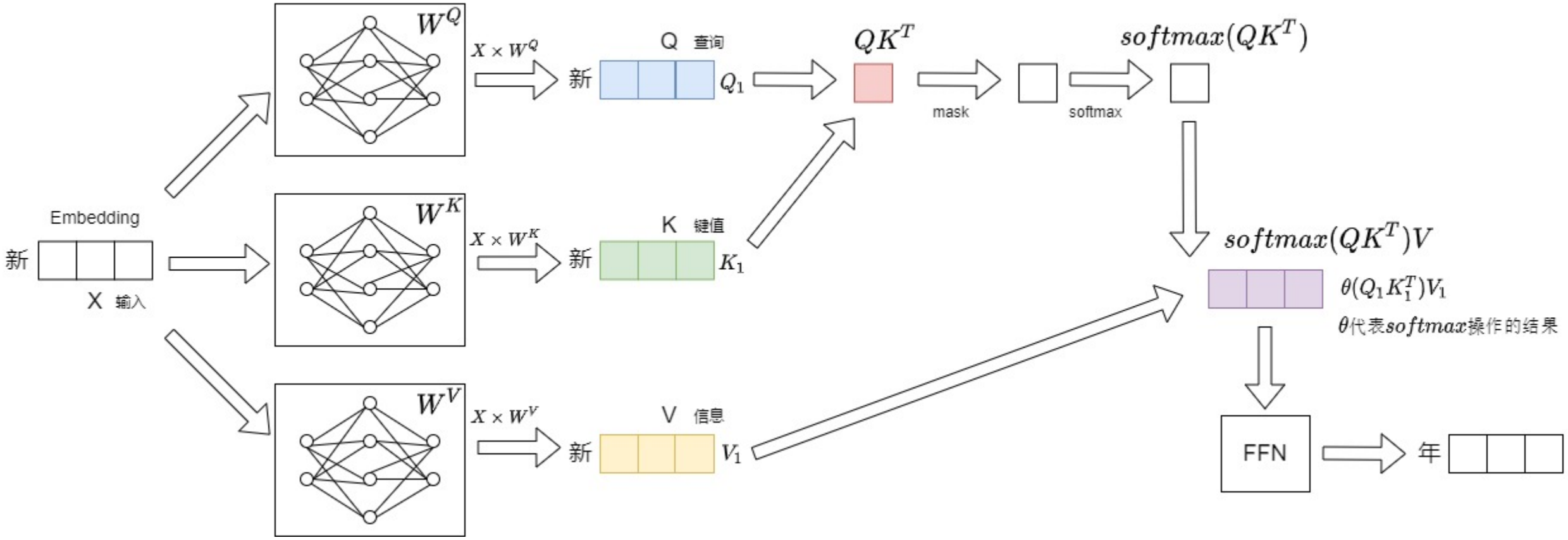
- LLM inference



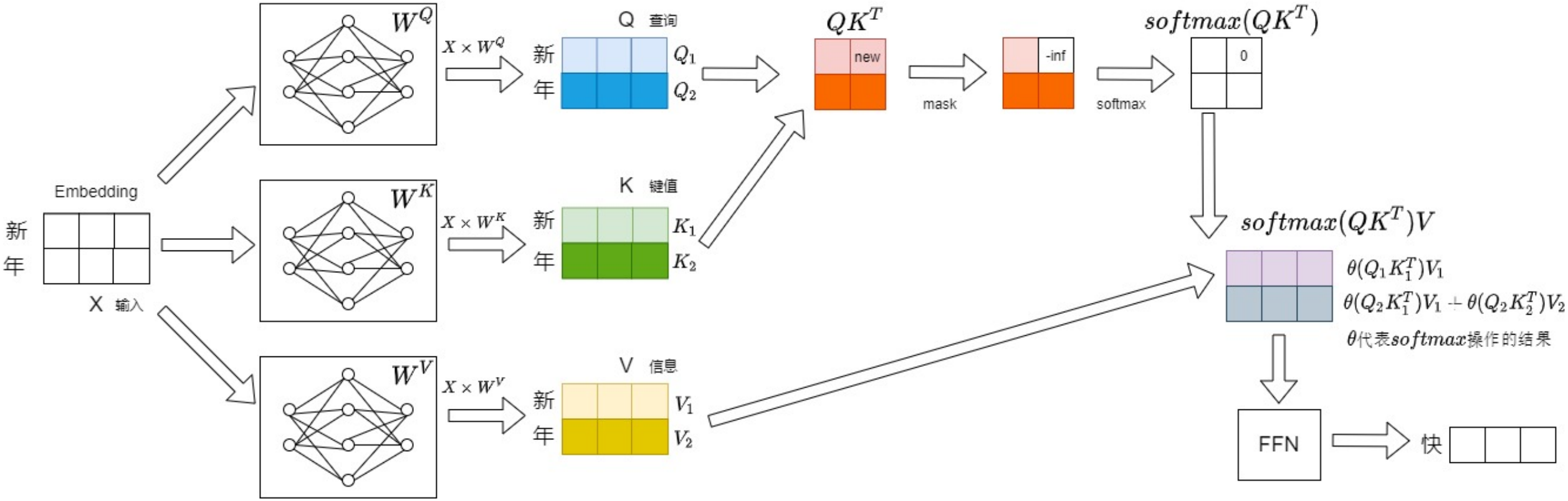
A Simple Case



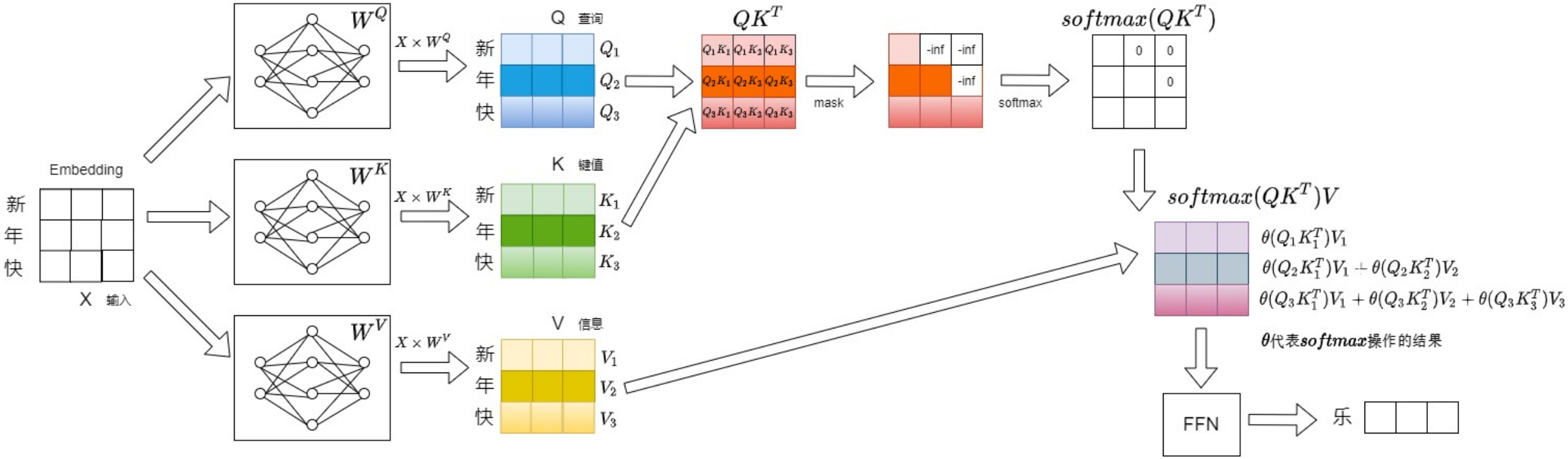
Computation of LLM Inference: 1st step



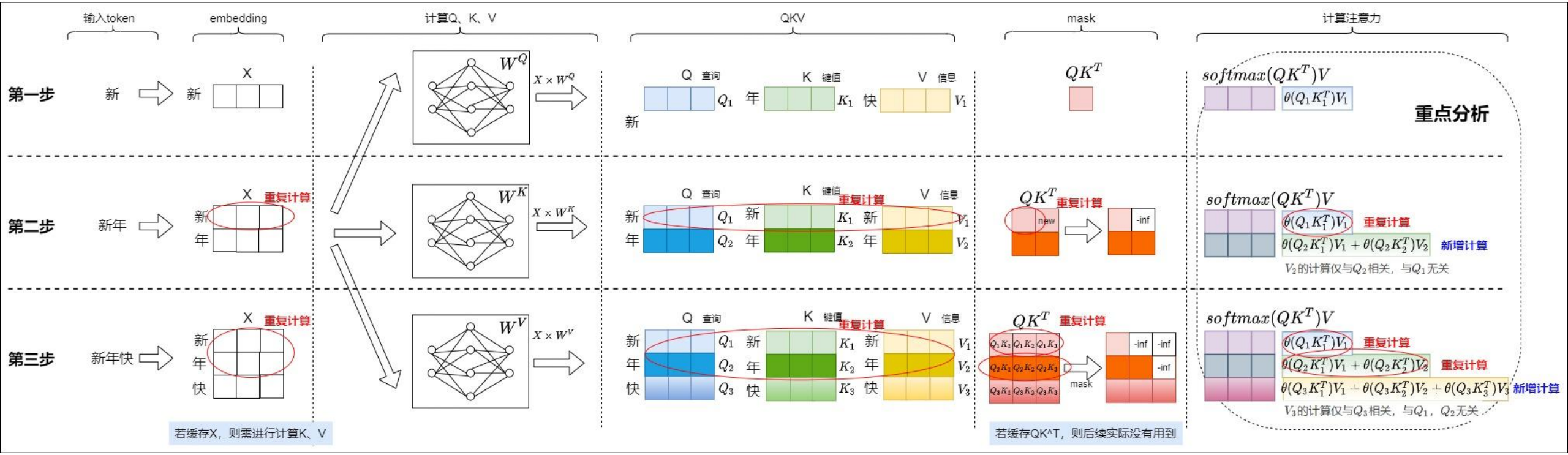
2nd step


Embedding
新
年
X 输入
W^Q
W^K
W^V
$X \times W^Q$
$X \times W^K$
$X \times W^V$
Q 查询
K 键值
V 信息
新
年
Q_1
Q_2
K_1
K_2
V_1
V_2
QK^T
new
mask
-inf
softmax
$softmax(QK^T)$
0
$softmax(QK^T)V$
$\theta(Q_1K_1^T)V_1$
$\theta(Q_2K_1^T)V_1 + \theta(Q_2K_2^T)V_2$
θ代表$softmax$操作的结果
FFN
快

3rd step

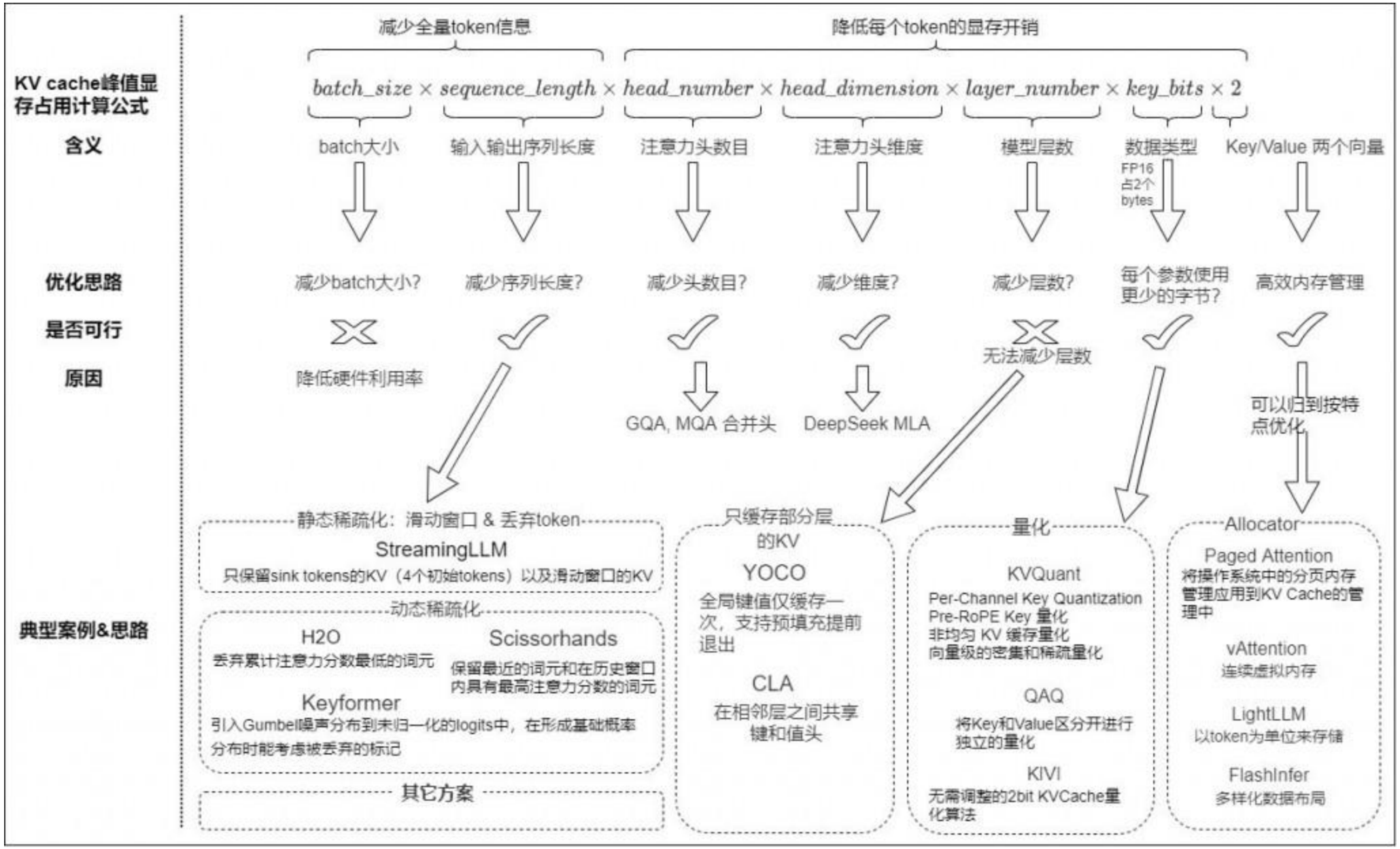

Embedding
新
年
快
X 输入
W^Q
W^K
W^V
$X \times W^Q$
$X \times W^K$
$X \times W^V$
Q 查询
K 键值
V 信息
Q_1
Q_2
Q_3
K_1
K_2
K_3
V_1
V_2
V_3
QK^T
mask
-inf
softmax
$softmax(QK^T)$
$softmax(QK^T)V$
$\theta(Q_1K_1^T)V_1$
$\theta(Q_2K_1^T)V_1 + \theta(Q_2K_2^T)V_2$
$\theta(Q_3K_1^T)V_1 + \theta(Q_3K_2^T)V_2 + \theta(Q_3K_3^T)V_3$
θ代表softmax操作的结果
FFN
乐

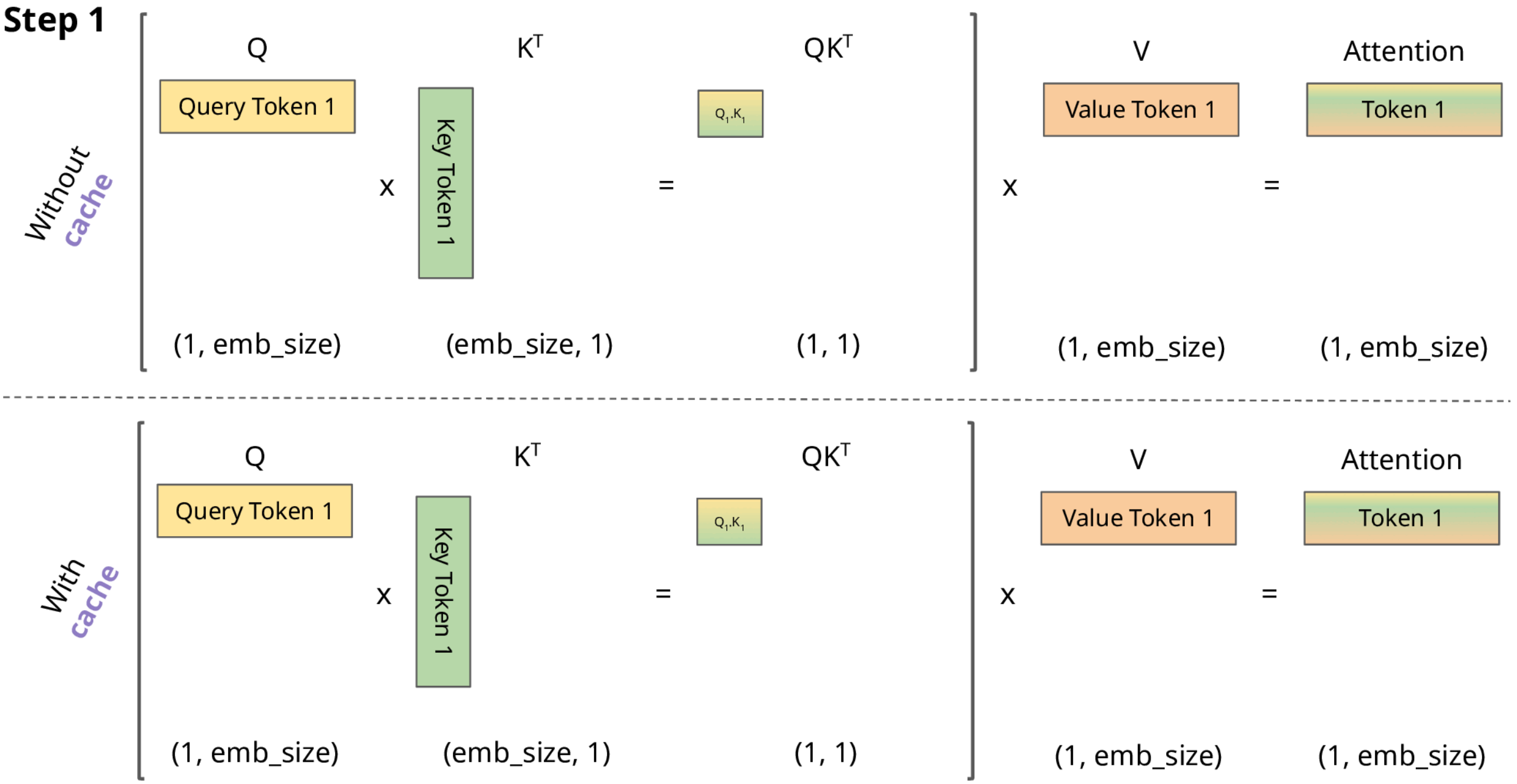
Redundancy During LLM inference

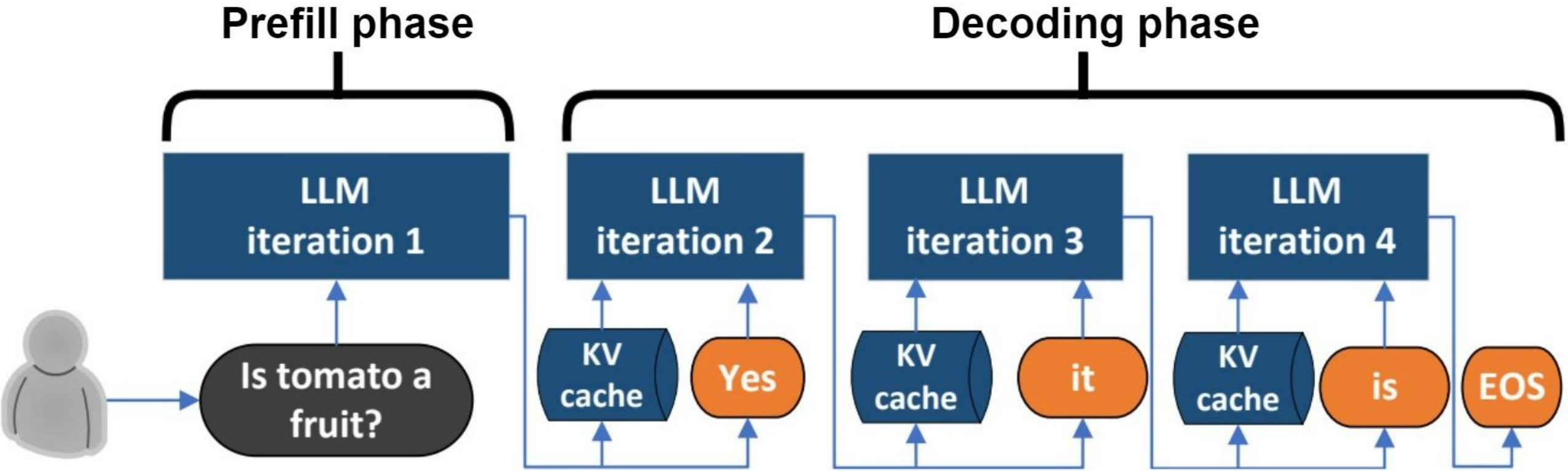


KV Cache Size

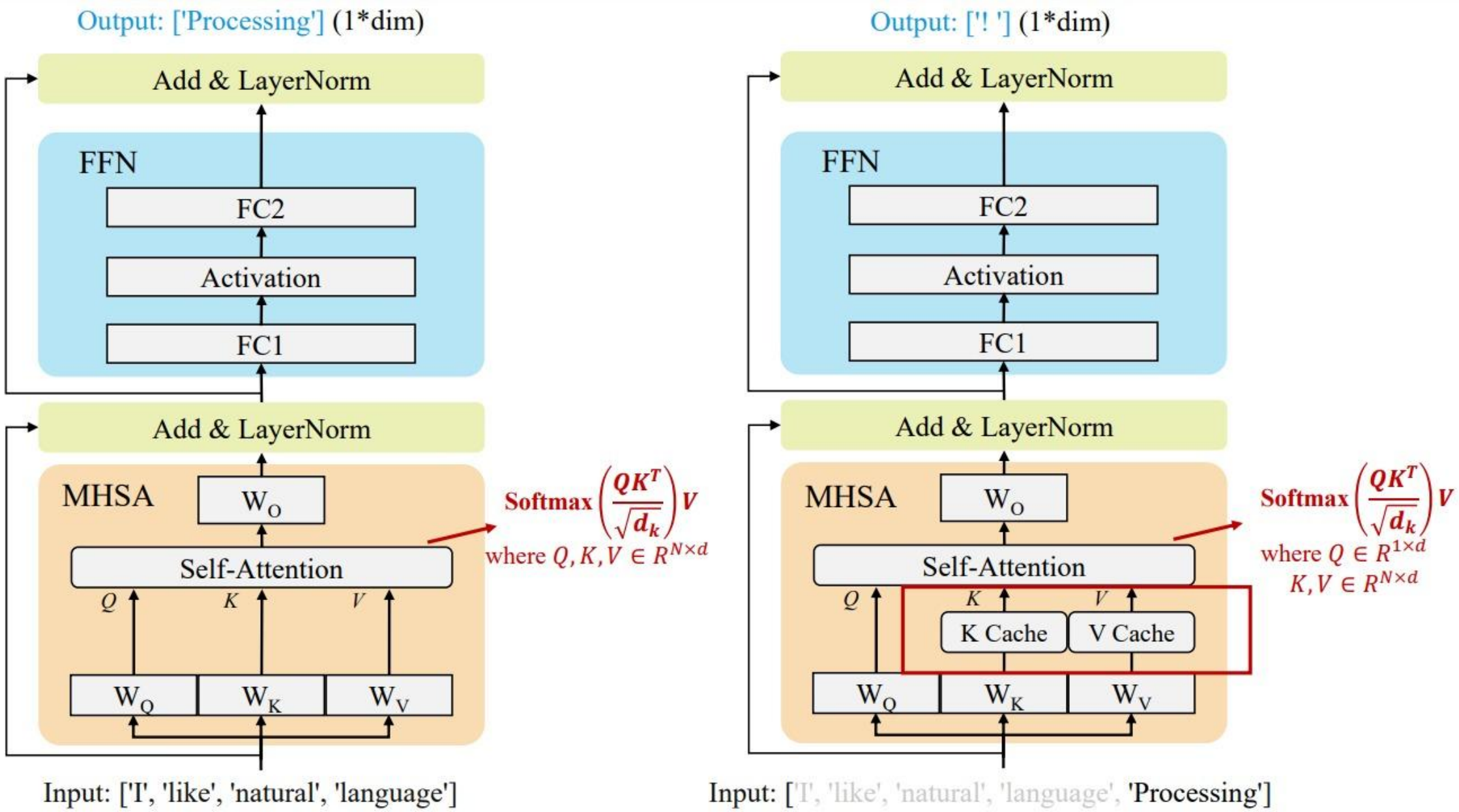
- Memory for KV Cache
  - 2×B×L×H×D×PxN
    - 2 → k, v 兩個向量
    - B: batch size
    - L: seq_len (提示 + 完成部分)
    - H: n_head
    - D: head_dim
    - P: 需要用多少位元來儲存，如果 fp16 則需要 2 byte
    - N: 模型層數
  - 可以變的就只有B, L, P，那其中 seq_len 最重要
    - 沒有特別去優化的話，你會發現 100K 就需要 22.8GB


KV cache峰值显存占用计算公式
含义
优化思路
是否可行
原因
典型案例&思路
减少全量token信息
降低每个token的显存开销
$batch\_size \times sequence\_length \times head\_number \times head\_dimension \times layer\_number \times key\_bits \times 2$
batch大小
输入输出序列长度
注意力头数目
注意力头维度
模型层数
数据类型
FP16 占2个 bytes
Key/Value 两个向量
减少batch大小?
减少序列长度?
减少头数目?
减少维度?
减少层数?
每个参数使用更少的字节?
高效内存管理
降低硬件利用率
GQA, MQA 合并头
DeepSeek MLA
无法减少层数
可以归到按特点优化
静态稀疏化：滑动窗口 & 丢弃token
StreamingLLM
只保留sink tokens的KV（4个初始tokens）以及滑动窗口的KV
动态稀疏化
H2O
丢弃累计注意力分数最低的词元
Scissorhands
保留最近的词元和在历史窗口内具有最高注意力分数的词元
Keyformer
引入Gumbel噪声分布到未归一化的logits中，在形成基础概率分布时能考虑被丢弃的标记
其它方案
只缓存部分层的KV
YOCO
全局键值仅缓存一次，支持预填充提前退出
CLA
在相邻层之间共享键和值头
量化
KVQuant
Per-Channel Key Quantization
Pre-RoPE Key 量化
非均匀 KV 缓存量化
向量级的密集和稀疏量化
QAQ
将Key和Value区分开进行独立的量化
KIVI
无需调整的2bit KVCache量化算法
Allocator
Paged Attention
将操作系统中的分页内存管理应用到KV Cache的管理中
vAttention
连续虚拟内存
LightLLM
以token为单位来存储
FlashInfer
多样化数据布局


Step 1
Without cache
Q
K^T
QK^T
V
Attention
Query Token 1
Key Token 1
Q1.K1
Value Token 1
Token 1
x
=
x
=
(1, emb_size)
(emb_size, 1)
(1, 1)
(1, emb_size)
(1, emb_size)
With cache
Q
K^T
QK^T
V
Attention
Query Token 1
Key Token 1
Q1.K1
Value Token 1
Token 1
x
=
x
=
(1, emb_size)
(emb_size, 1)
(1, 1)
(1, emb_size)
(1, emb_size)
Values that will be masked
Values that will be taken from cache
Transformers KV Caching



图片来源：论文 Splitwise: Efficient Generative LLM Inference Using Phase Splitting



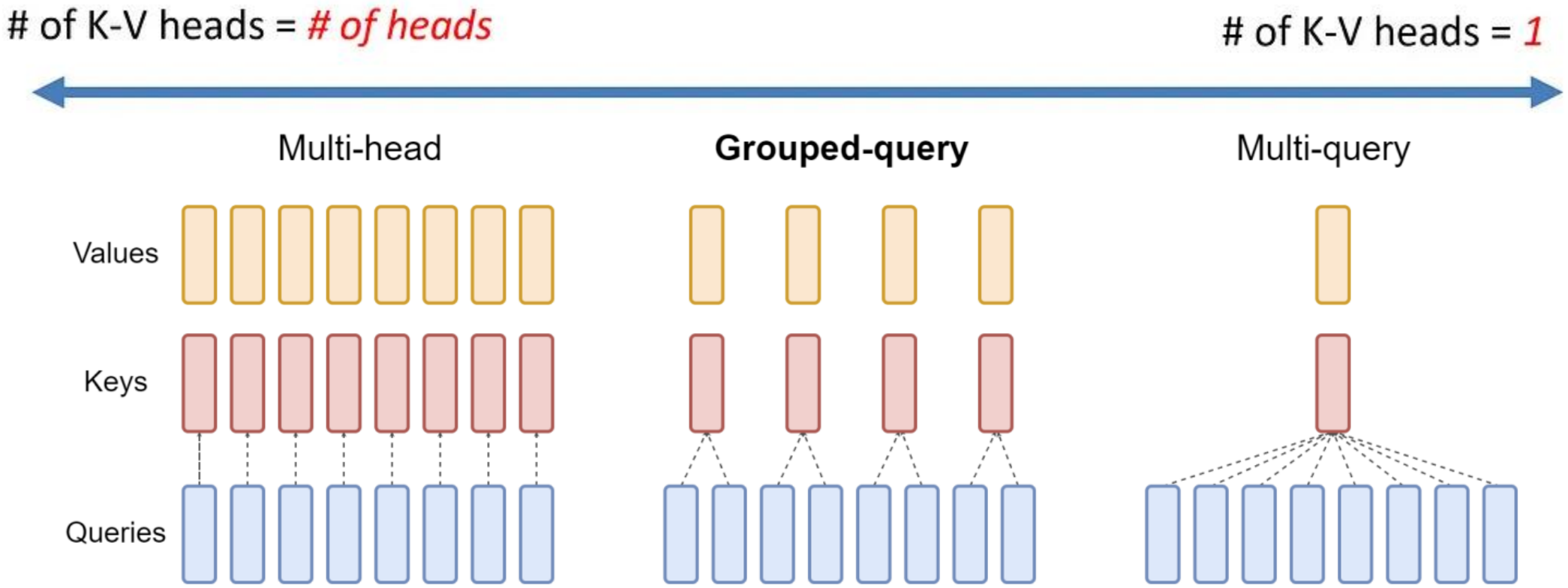
(a) The prefilling stage

(b) The decoding stage

图片来源：论文 A Survey on Efficient Inference for Large Language Models

GQA

- GQA 只在 LLama 2 中部分應用，從 LLama 3 開始應用於所有模型。



分组查询注意力机制 GQA 是在每组查询向量中共享一个键和值向量

Figure 2: Overview of grouped-query method. Multi-head attention has H query, key, and value heads. Multi-query attention shares single key and value heads across all query heads. Grouped-query attention instead shares single key and value heads for each *group* of query heads, interpolating between multi-head and multi-query attention.

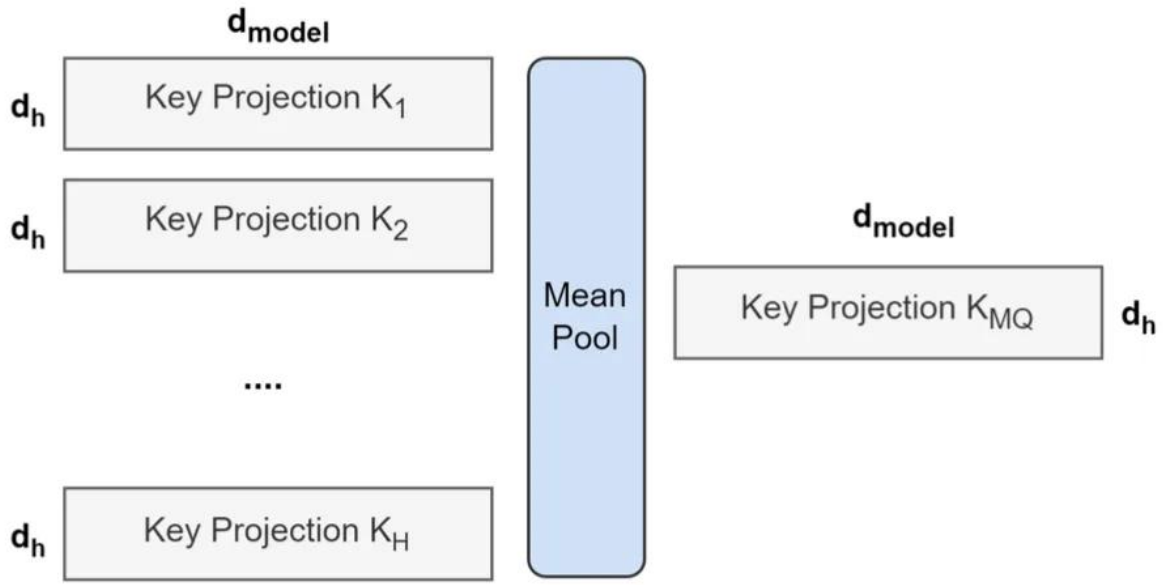
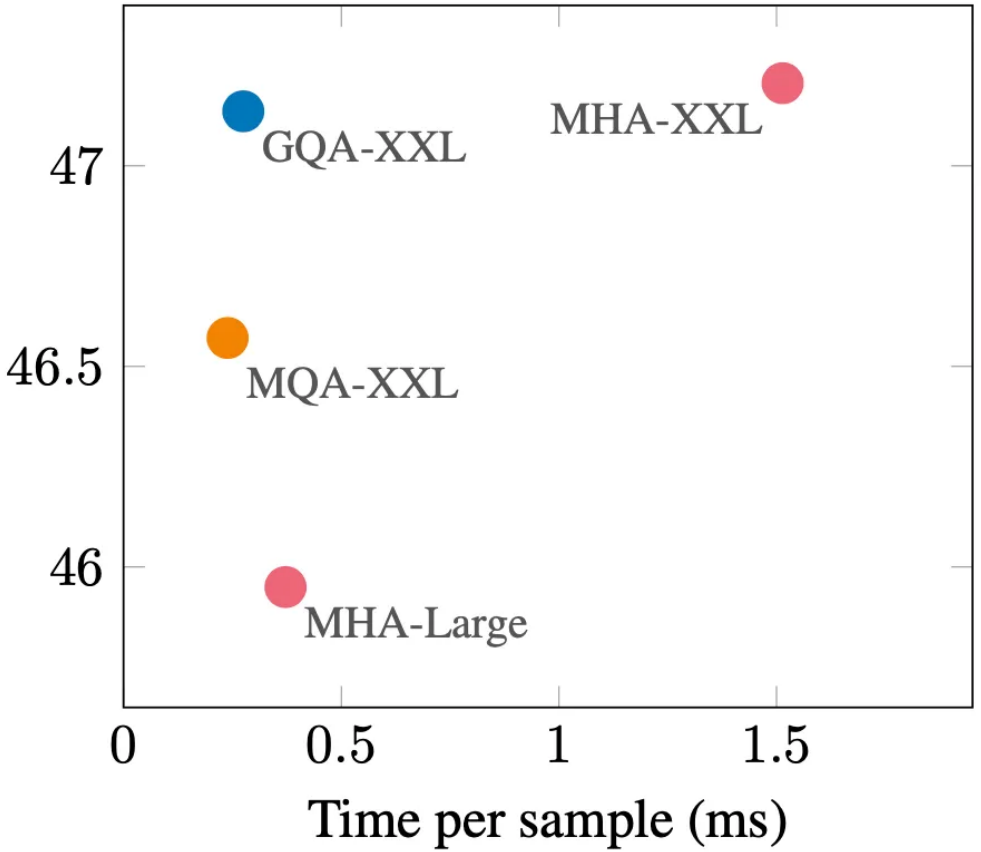


Figure 1: Overview of conversion from multi-head to multi-query attention. Key and value projection matrices from all heads are mean pooled into a single head.

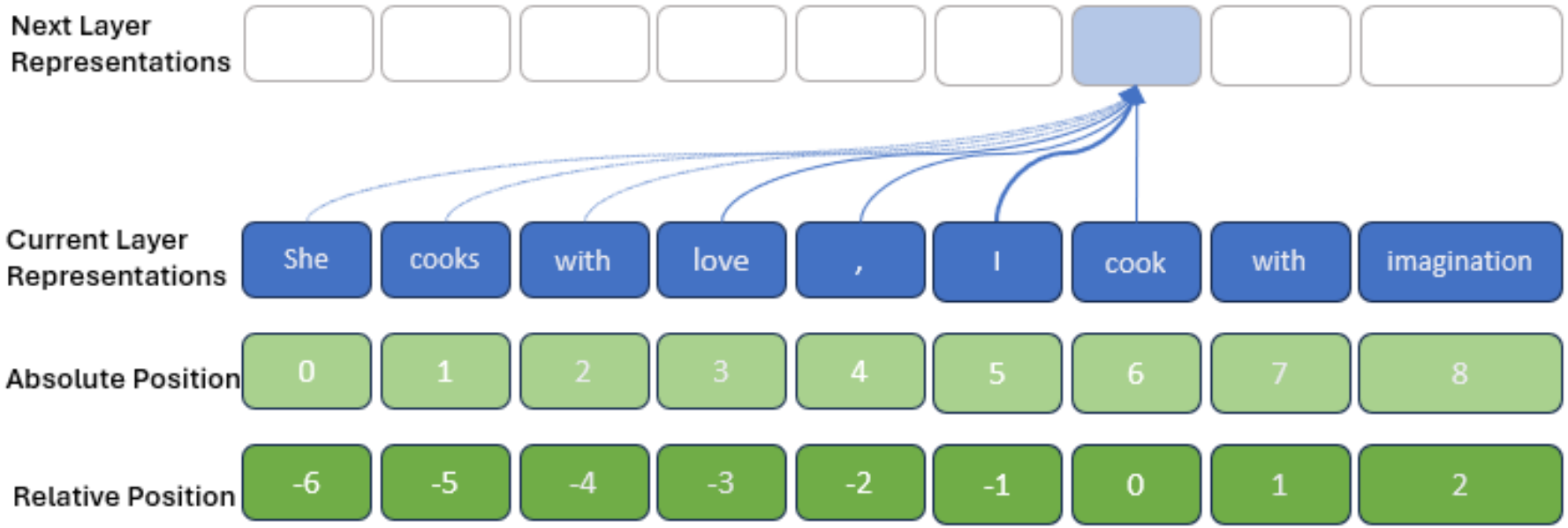


LLAMA: ROPE

(Optional) Positional Embeddings

- absolute positional embeddings
  - Transformer 模型預設無法辨識詞彙順序，會將句子視為無序的集合。 為了保留順序資訊，必須添加位置資訊，
  - 方法:
    - 從數據中學習:最大長度受限於訓練數據中最長句子的長度
    - 正弦函數
      - 使用正弦函數可以為序列中的每個可能位置構建獨特的位置嵌入,可以表示任意長度
  - 問題
    - 長度限制： 如果只學習到特定位置的向量，則無法表示超過該長度的序列
    - 位置間關係缺乏： 每個位置嵌入彼此獨立，無法表示位置之間的相對關係。 例如，位置1和2應該比位置1和500更相似。

- absolute positional encoding
  - lack of translation invariance

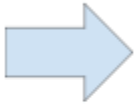


- Relative position embeddings
  - T5: use relative bias for each relative position to reduce number of learned positional parameters
    - Add extra computation for each position

$$att = softmax(\frac{1}{\sqrt{(dim)}}(QK^T + PosBias_{rel}))$$

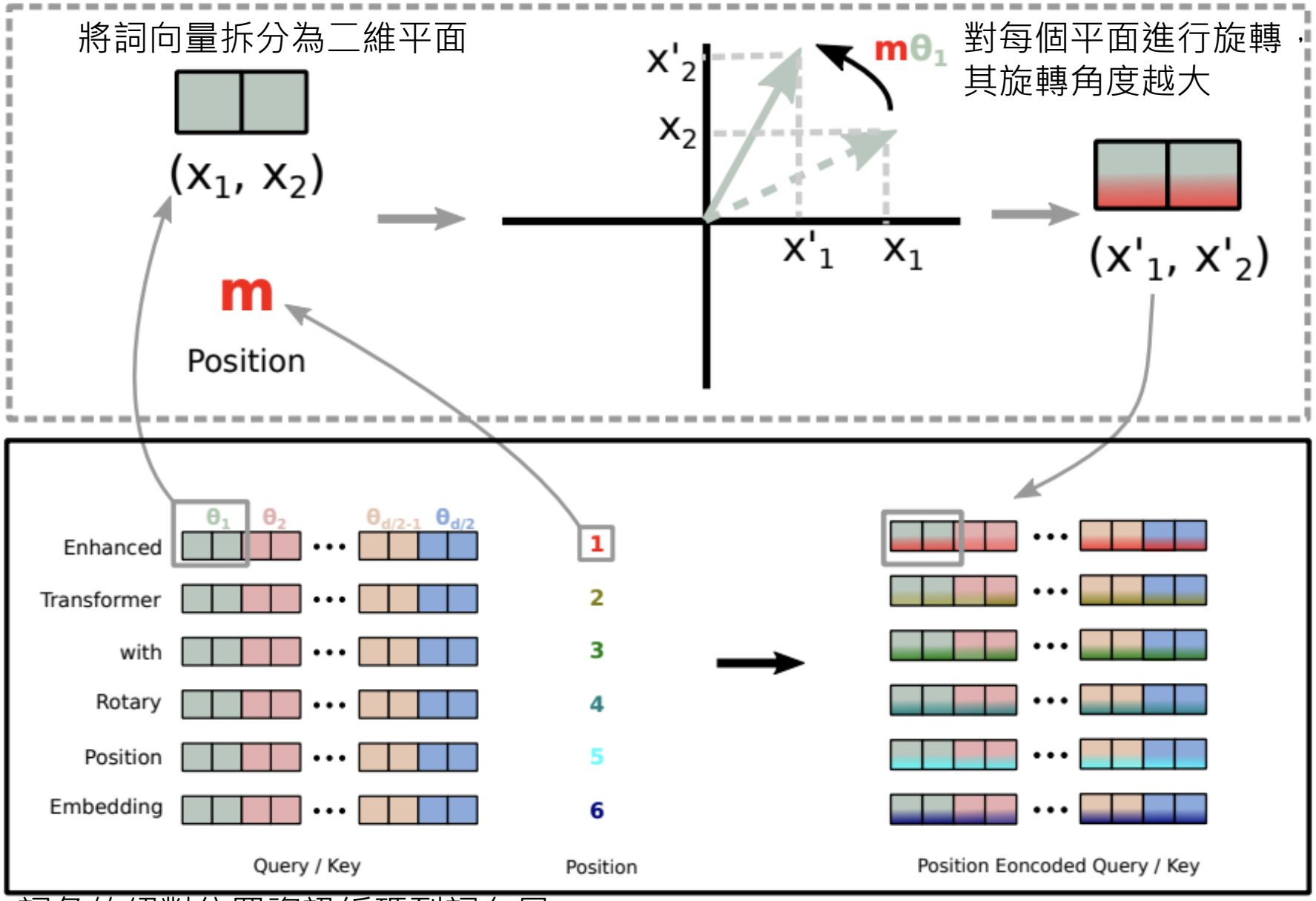
0	1	2	3	4
-1	0	1	2	3
-2	-1	0	1	2
-3	-2	-1	0	1
-4	-3	-2	-1	0

Relative Positions Pattern
in 5 token Attention matrix



-4	-3	-2	-1	0	1	2	3	4

Unique Relative Positions



詞彙的絕對位置資訊編碼到詞向量

Figure 1: Implementation of Rotary Position Embedding(RoPE).

The rotary position embeddings are only applied to the query and the keys, but not the values.

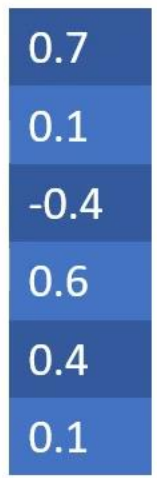
The rotary position embeddings are applied after the vector q and k have been multiplied by the W matrix in the attention mechanism, while in the vanilla transformer they're applied before.

- Positional embedding: RoPE (rotary position embedding)
  - 結合了絕對位置嵌入和相對位置嵌入的優點
  - 核心概念
    - 透過**旋轉詞向量來編碼位置**資訊。
    - 與傳統的將位置向量直接加到詞向量上的方法不同，RoPE 將詞向量視為高維空間中的點，並透過旋轉操作來表示其在序列中的位置。
  - 運作步驟：
    - 1.將詞向量拆分為二維平面：
      - 對於高維詞向量，RoPE 會將其分成多個二維平面，每個平面代表詞向量中的兩個維度。
    - 2. 對每個平面進行旋轉：
      - RoPE 會根據詞彙在句子中的位置，對每個二維平面進行旋轉。 位置越靠後的詞彙，其旋轉角度越大。
      - 旋轉的角度由一個預先定義的函數決定，該函數與詞彙的位置和維度相關。
      - 每個二維平面的旋轉角度都不同，以確保每個維度都能夠編碼位置資訊。
    - 3.合併旋轉後的向量：
      - 將所有旋轉後的二維平面合併起來，得到最終的詞向量表示。

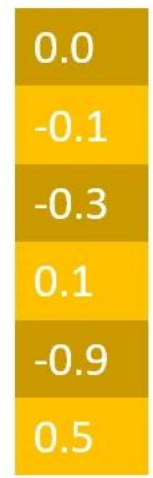
- 優點：
  - 保留絕對位置資訊：
    - RoPE 透過旋轉操作，將詞彙的絕對位置資訊編碼到詞向量中。
  - 捕捉相對位置關係：
    - 當兩個詞彙在句子中的距離保持不變時，它們的旋轉角度差也會保持不變，因此它們的點積也保持不變。 這意味著 RoPE 能夠有效地捕捉詞彙之間的相對位置關係。
  - 計算效率高：
    - RoPE 可以使用向量乘法和加法來實現，避免了耗時的矩陣乘法運算

Absolute Positional Embeddings

The dog chased the pig



position = 2



position = 2

Learned from data

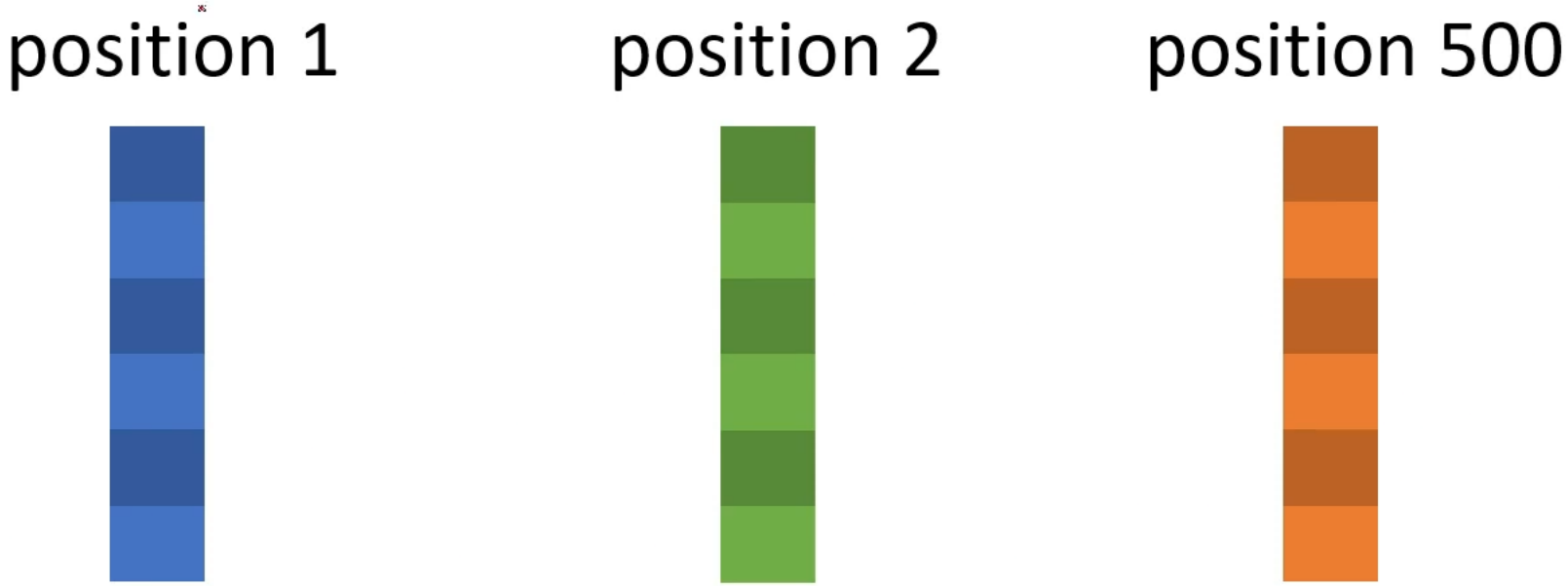
- Position vectors for 1-512
- Max length is bounded

Sinusoidal function

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

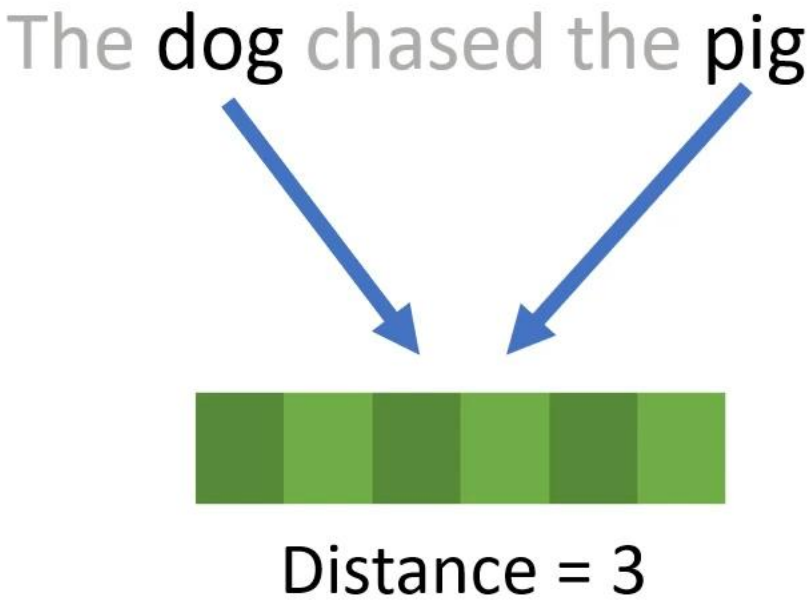
Empirically: learned and sinusoidal positional embeddings have similar performance



每個位置嵌入彼此獨立，**缺乏位置間的關聯性**。例如，位置 1 和 2 的相似度應該比位置 1 和 500 更高，但絕對位置嵌入無法體現這一點。

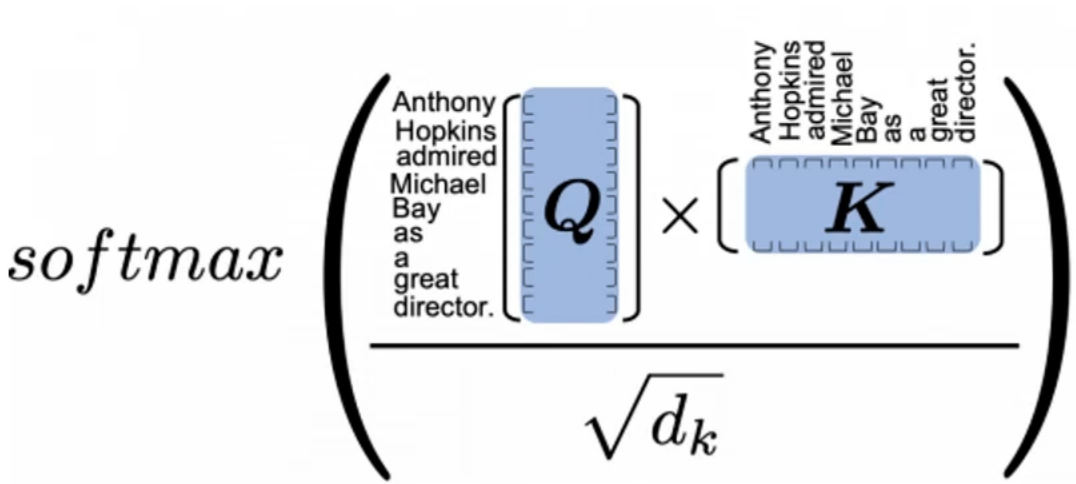
- **相對位置嵌入**：不直接表示詞彙的絕對位置，而是**學習詞彙對之間的相對距離**。
  - **優點**：
    - 能捕捉位置間的關聯性，例如距離 3 的詞彙對，無論其絕對位置為何，都具有相同的表示。
    - 可以處理任意長度的序列。
  - **缺點**：
    - **實作上較複雜**，需要修改注意力機制。
    - **計算效率較低**，尤其對於長序列而言。
    - 難以有效利用KV cache。

Relative Positional Embeddings

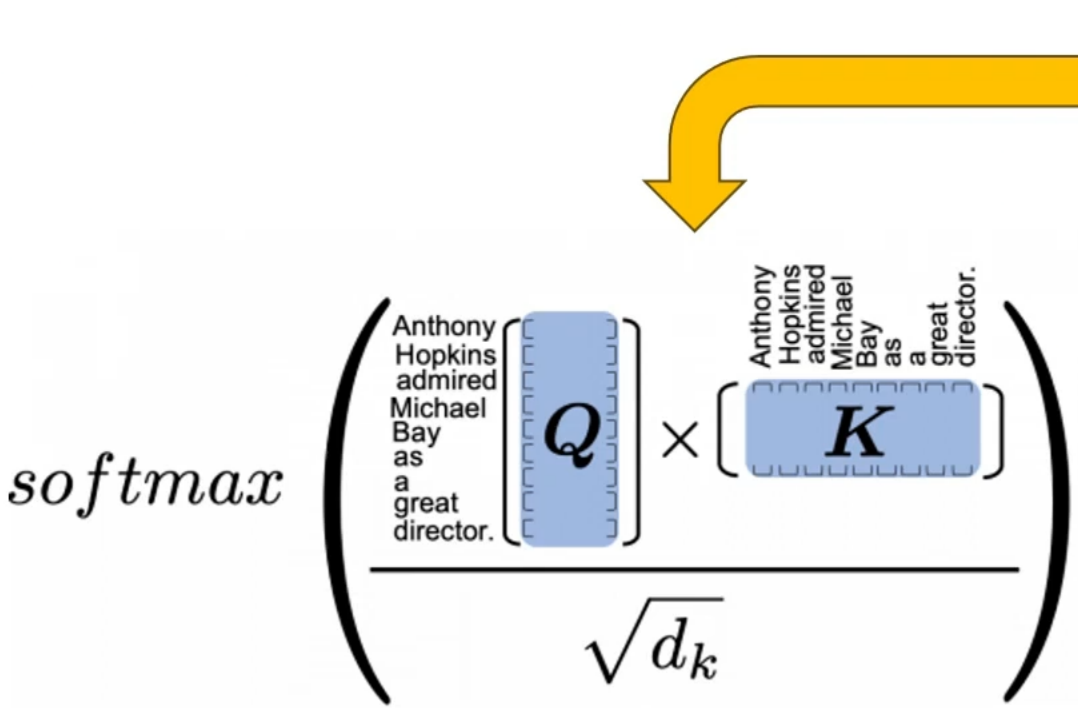


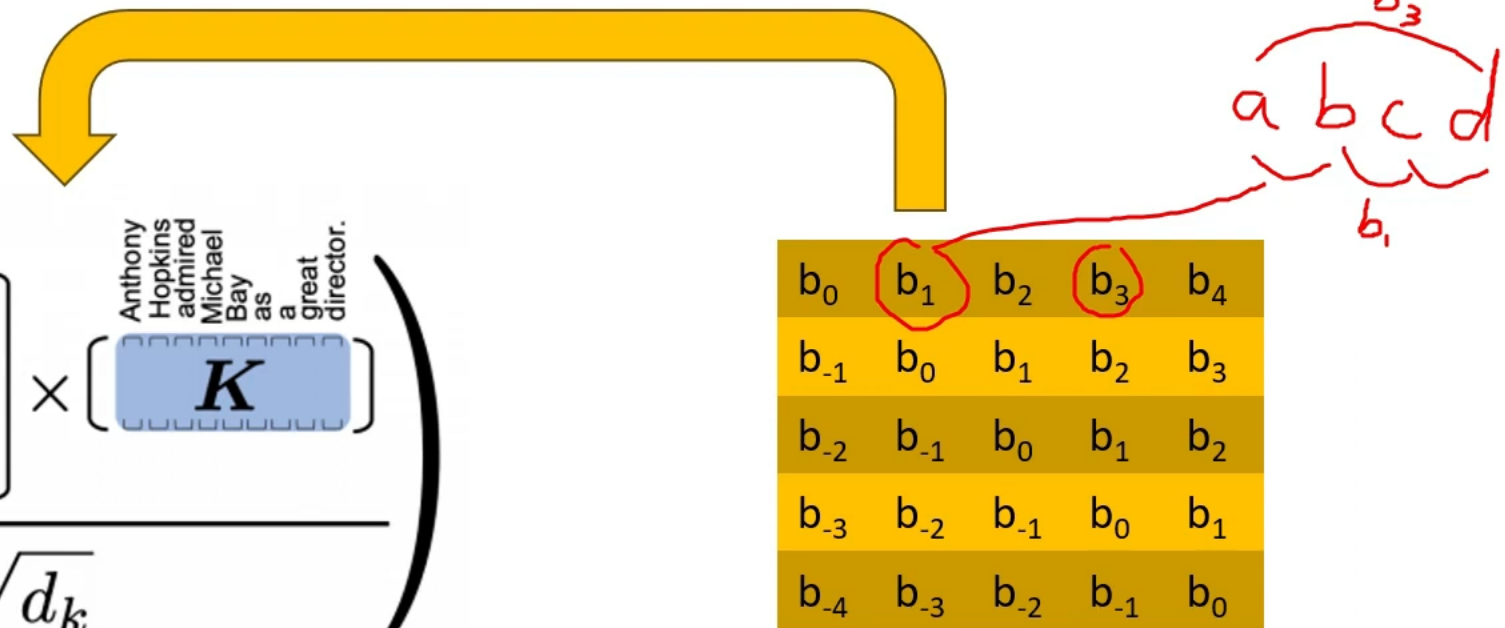
But every pair has different positional embeddings. We cannot simply add this to input vector
We need to modify the attention calculation

Relative Positional Embeddings: T5 Model

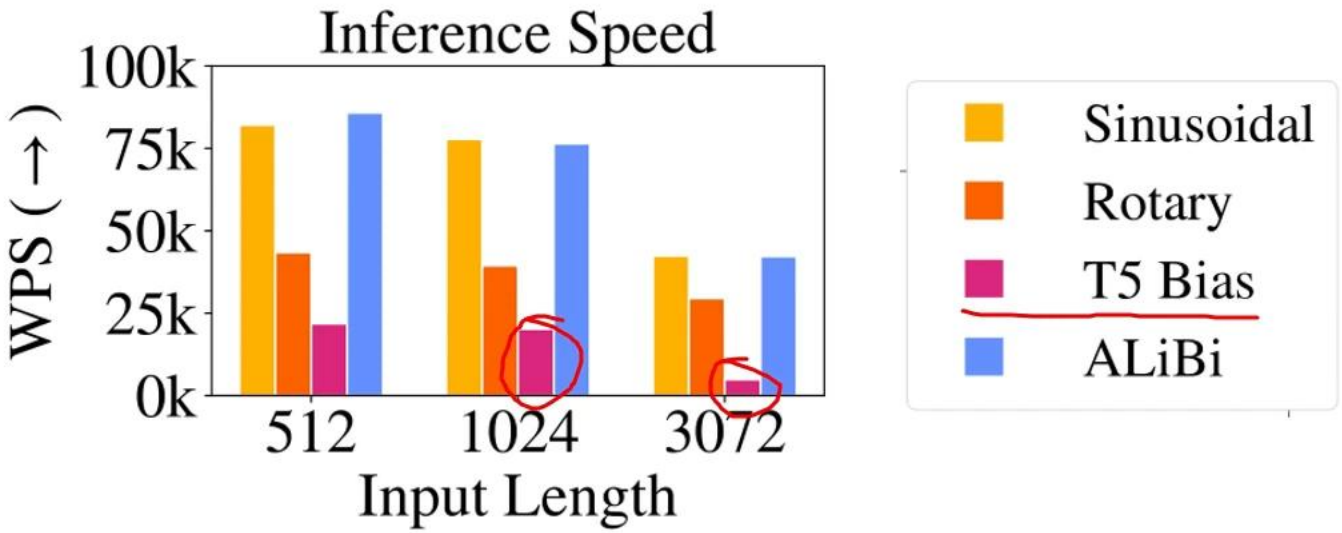


Relative Positional Embeddings: T5 Model





Relative embeddings are slow!

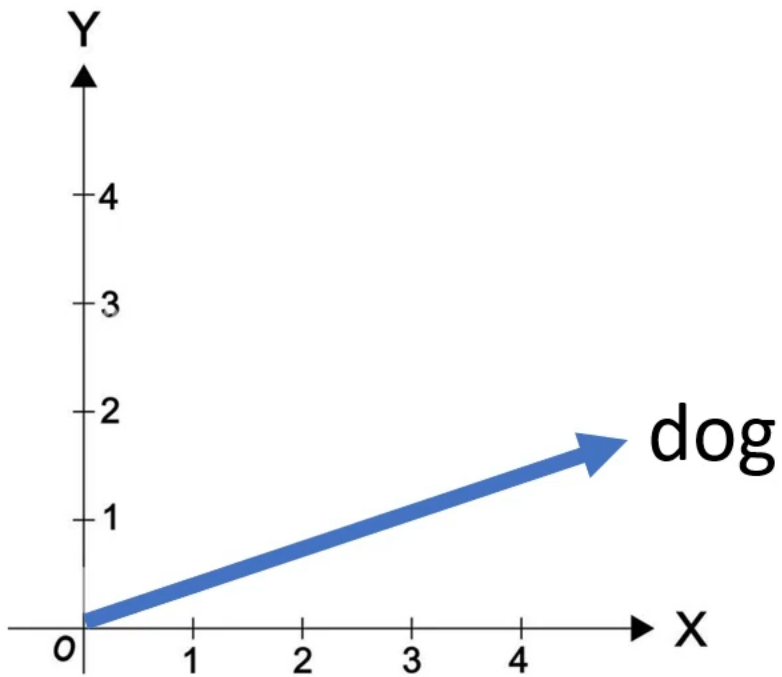


Why?

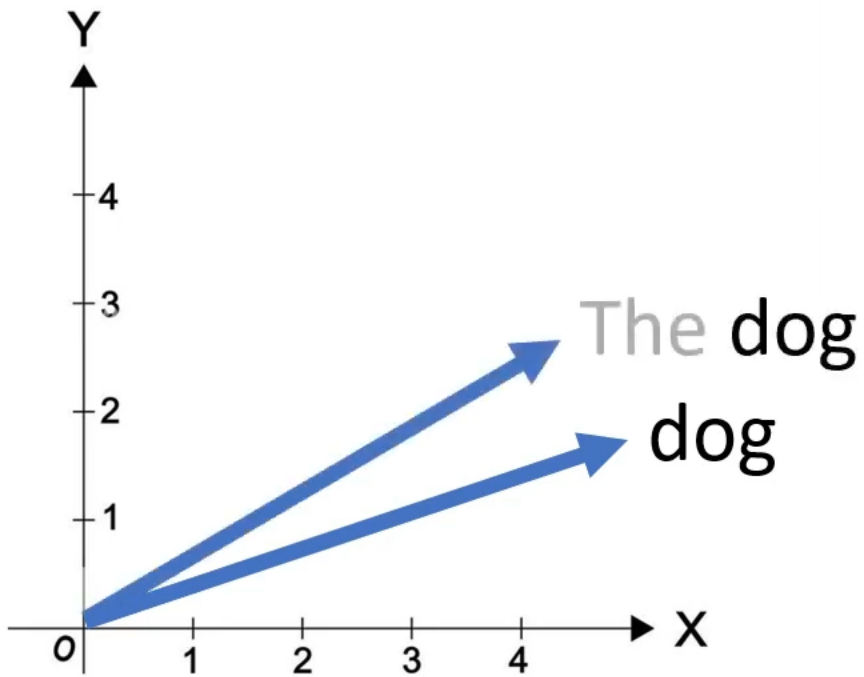
- Extra step in self-attention layer
- Changes in every step → difficult for KV cache
- Not commonly used

[1] Press, Ofir, Noah A. Smith, and Mike Lewis. "Train short, test long: Attention with linear biases enables input length extrapolation."

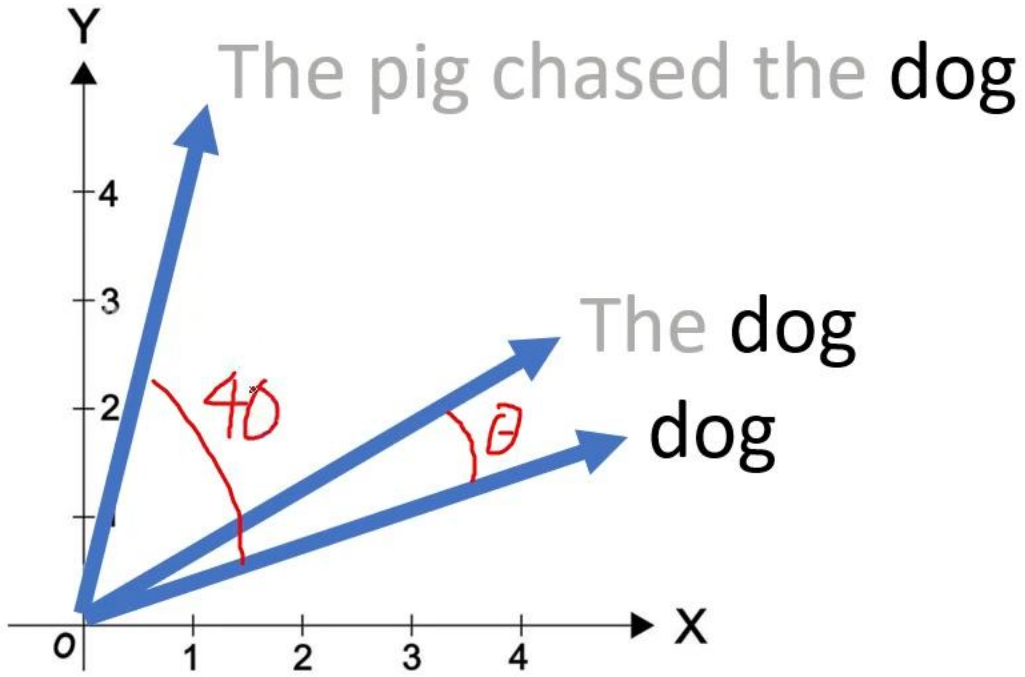
Rotary Positional Embeddings



Rotary Positional Embeddings

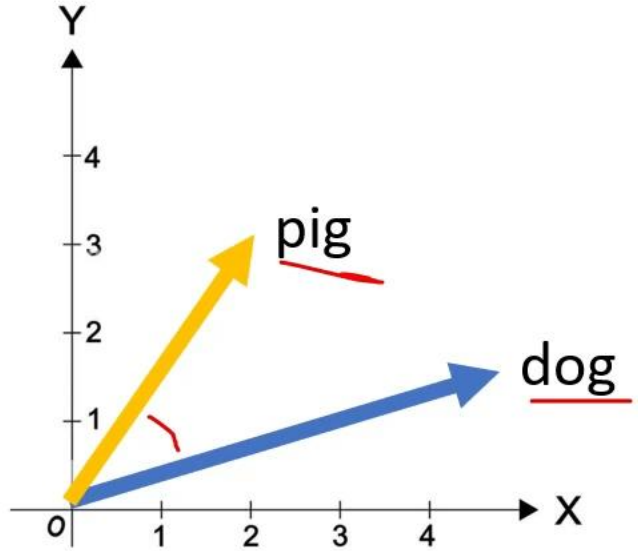


Rotary Positional Embeddings

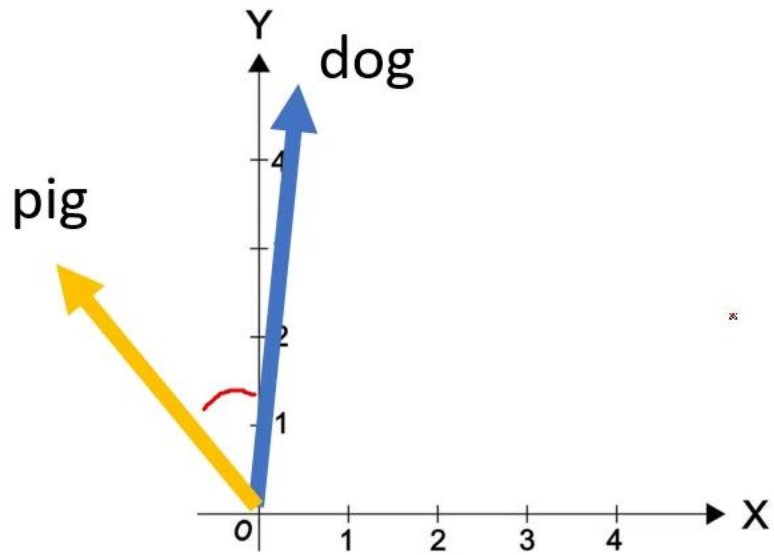


旋轉位置嵌入 (Rotary Positional Embeddings, RoPE) 是一種結合絕對位置嵌入和相對位置嵌入優點的新方法。它不是將位置向量加到詞彙向量上，而是**對詞彙向量進行旋轉**。旋轉的角度與詞彙在句子中的位置成正比。

The pig chased the dog



Once upon a time, the pig chased the dog



Implementation: Matrix Multiplication

$$f_{\{q,k\}}(\boldsymbol{x}_m, m) = \begin{pmatrix} \cos m\theta & -\sin m\theta \\ \sin m\theta & \cos m\theta \end{pmatrix} \begin{pmatrix} W^{(11)}_{\{q,k\}} & W^{(12)}_{\{q,k\}} \\ W^{(21)}_{\{q,k\}} & W^{(22)}_{\{q,k\}} \end{pmatrix} \begin{pmatrix} x^{(1)}_m \\ x^{(2)}_m \end{pmatrix}$$

Q, K

not V

$$f_q(\boldsymbol{x}_m, m) = (\boldsymbol{W}_q \boldsymbol{x}_m) e^{im\theta}$$

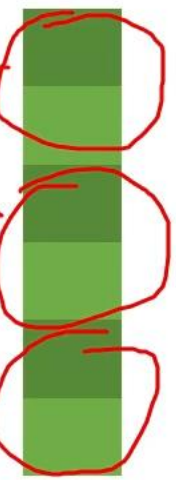
$$f_k(\boldsymbol{x}_n, n) = (\boldsymbol{W}_k \boldsymbol{x}_n) e^{in\theta}$$

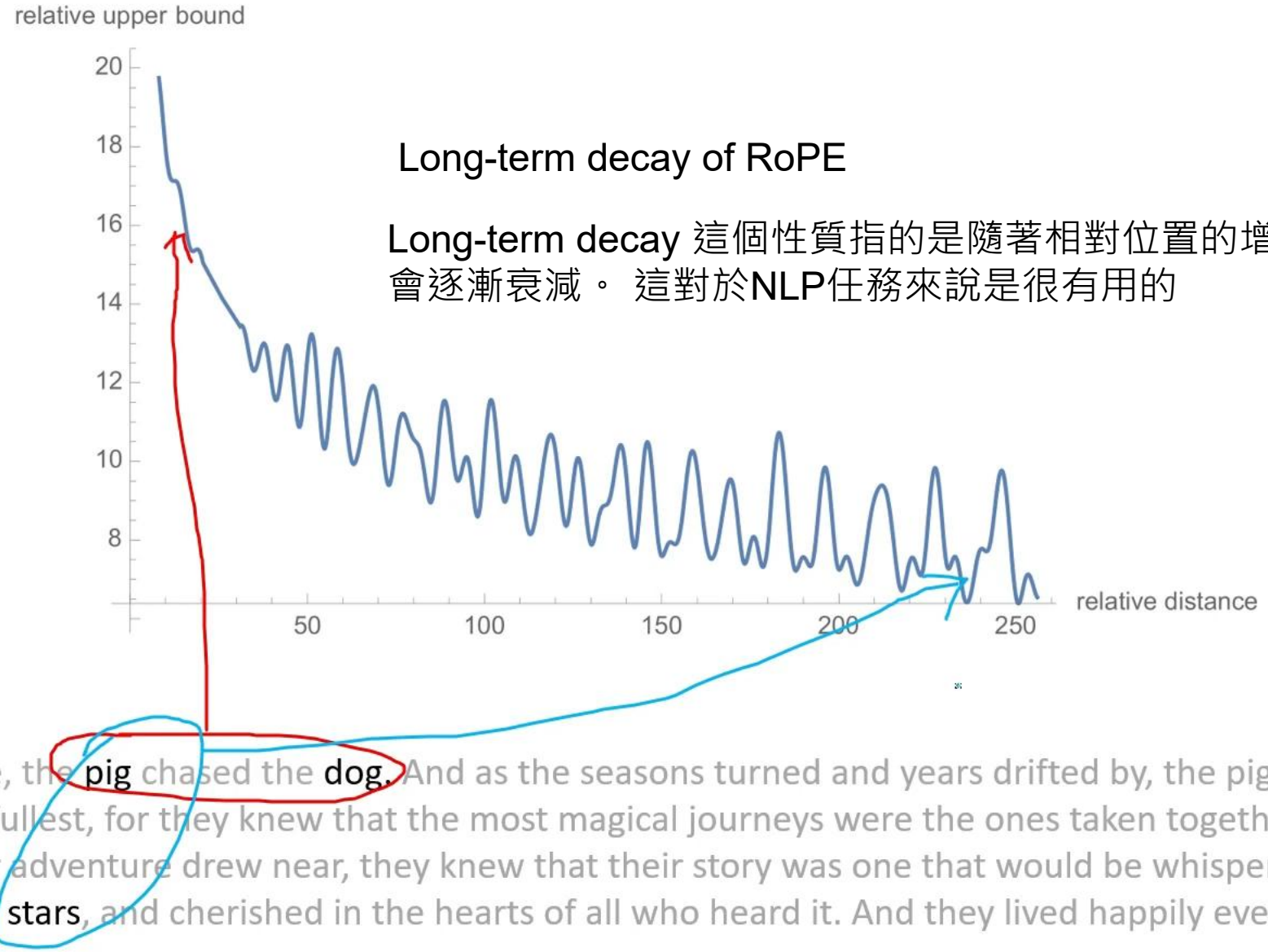
$$g(\boldsymbol{x}_m, \boldsymbol{x}_n, m-n) = \mathrm{Re}[(\boldsymbol{W}_q \boldsymbol{x}_m)(\boldsymbol{W}_k \boldsymbol{x}_n)^* e^{i(m-n)\theta}]$$

Implementation: Matrix Multiplication

$$f_{\{q,k\}}(\boldsymbol{x}_m, m) = \boldsymbol{R}^d_{\Theta,m} \boldsymbol{W}_{\{q,k\}} \boldsymbol{x}_m$$

$$\boldsymbol{R}^d_{\Theta,m} = \begin{pmatrix} \cos m\theta_1 & -\sin m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ \sin m\theta_1 & \cos m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cos m\theta_2 & -\sin m\theta_2 & \cdots & 0 & 0 \\ 0 & 0 & \sin m\theta_2 & \cos m\theta_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \cos m\theta_{d/2} & -\sin m\theta_{d/2} \\ 0 & 0 & 0 & 0 & \cdots & \sin m\theta_{d/2} & \cos m\theta_{d/2} \end{pmatrix}$$





Long-term decay of RoPE

Long-term decay 這個性質指的是隨著相對位置的增大，query和key的內積會逐漸衰減。 這對於NLP任務來說是很有用的

SELF SUPERVISED LEARNING

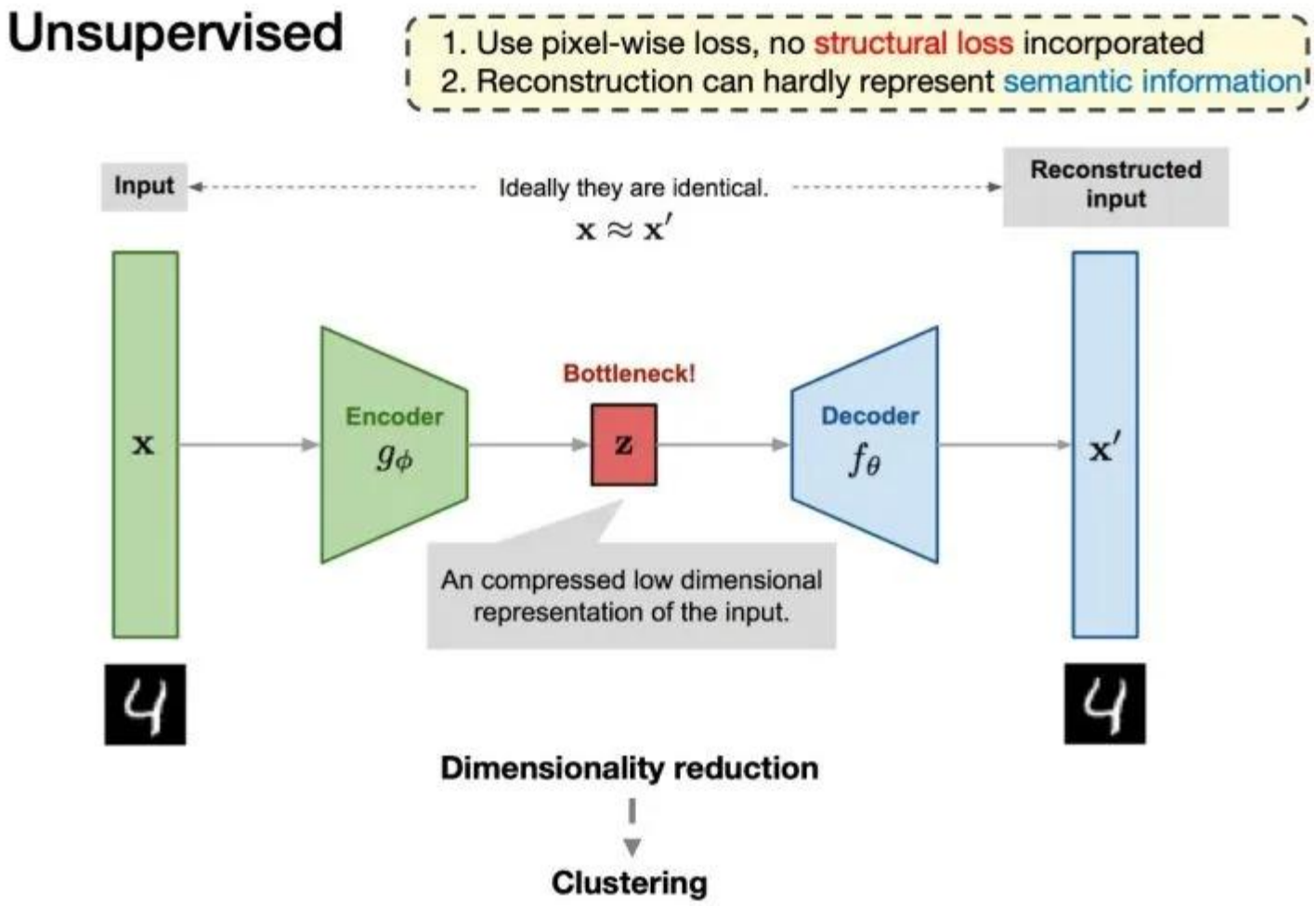
Self supervised Learning

SSL is a subset of unsupervised learning

Feature	Self-Supervised Learning	Unsupervised Learning
Data Utilization	Generates supervisory signals from unlabeled data	Operates entirely without labels
Learning Objective	Learns representations by solving specific pretext tasks	Discovers patterns and structures in the data
Feedback Mechanism	Involves feedback through pseudo-labels derived from data	Lacks feedback loops; focuses on overall data analysis
Common Techniques	Predicting missing parts, contrastive learning	Clustering, dimensionality reduction

Unsupervised Learning with AutoEncoder

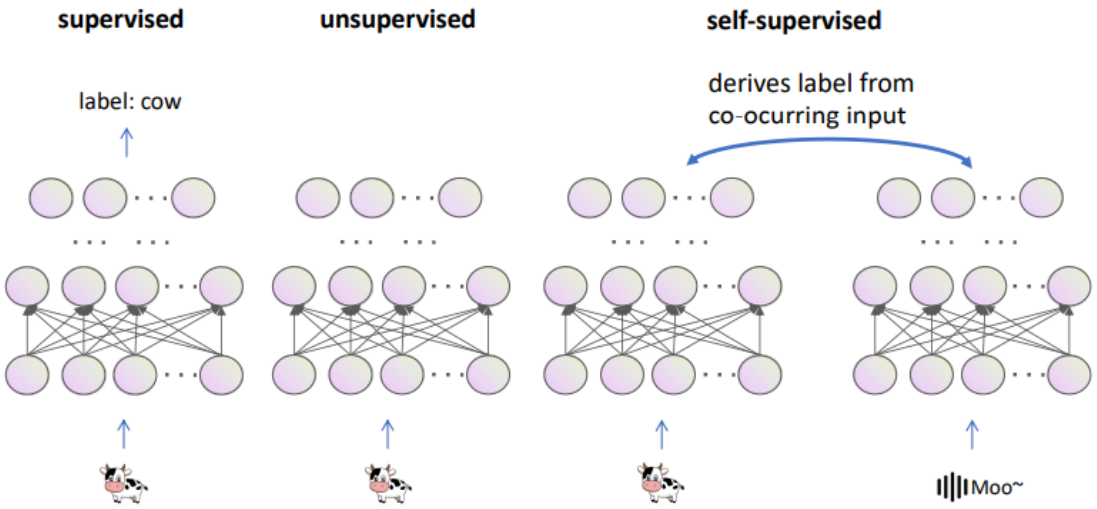
- Map input sample to latent space by encoder
- Reconstruct input by decoder



How Much Information is the Machine Given during Learning?

- **"Pure" Reinforcement Learning (cherry)**
  - The machine predicts a scalar reward given once in a while.
  - **A few bits for some samples**
- **Supervised Learning (icing)**
  - The machine predicts a category or a few numbers for each input
  - Predicting human-supplied data
  - **10→10,000 bits per sample**
- **Self-Supervised Learning (cake génoise)**
  - The machine predicts any part of its input for any observed part.
  - Predicts future frames in videos
  - **Millions of bits per sample**

Self-supervised Learning



ig. 3: The differences among supervised learning, unsuper-ised learning, and SSL. The image is reproduced from [32]. SL utilizes freely derived labels as supervision instead of nanually annotated labels.

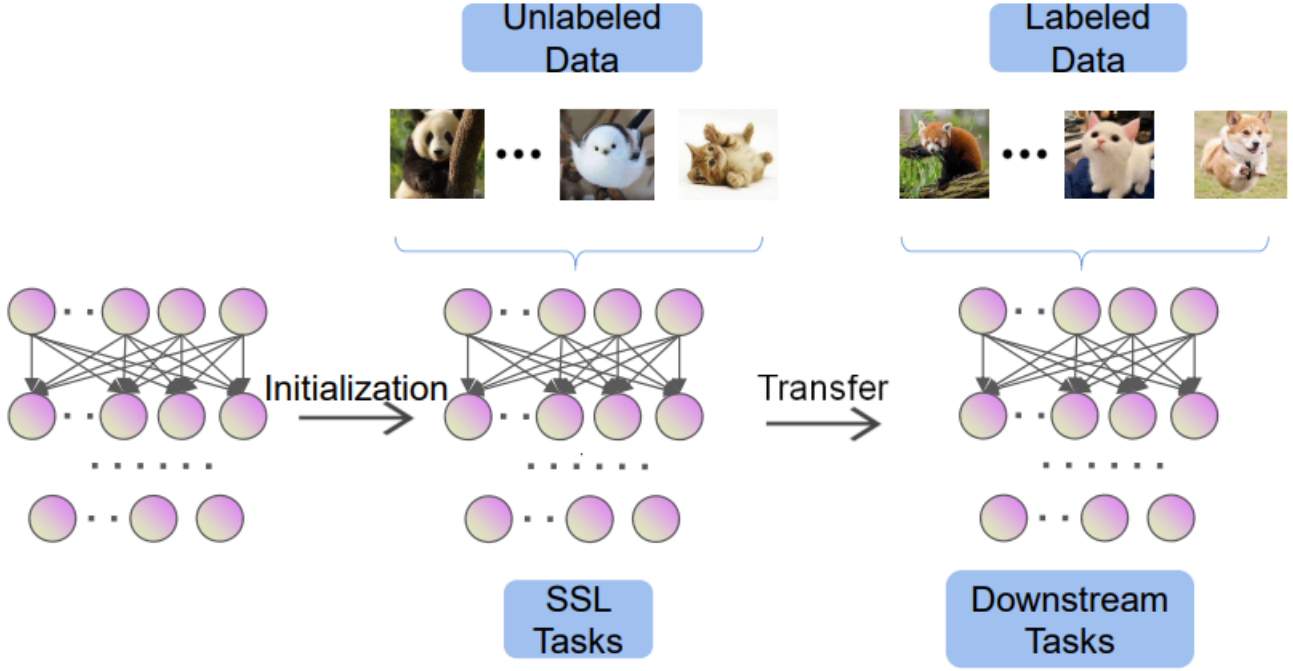
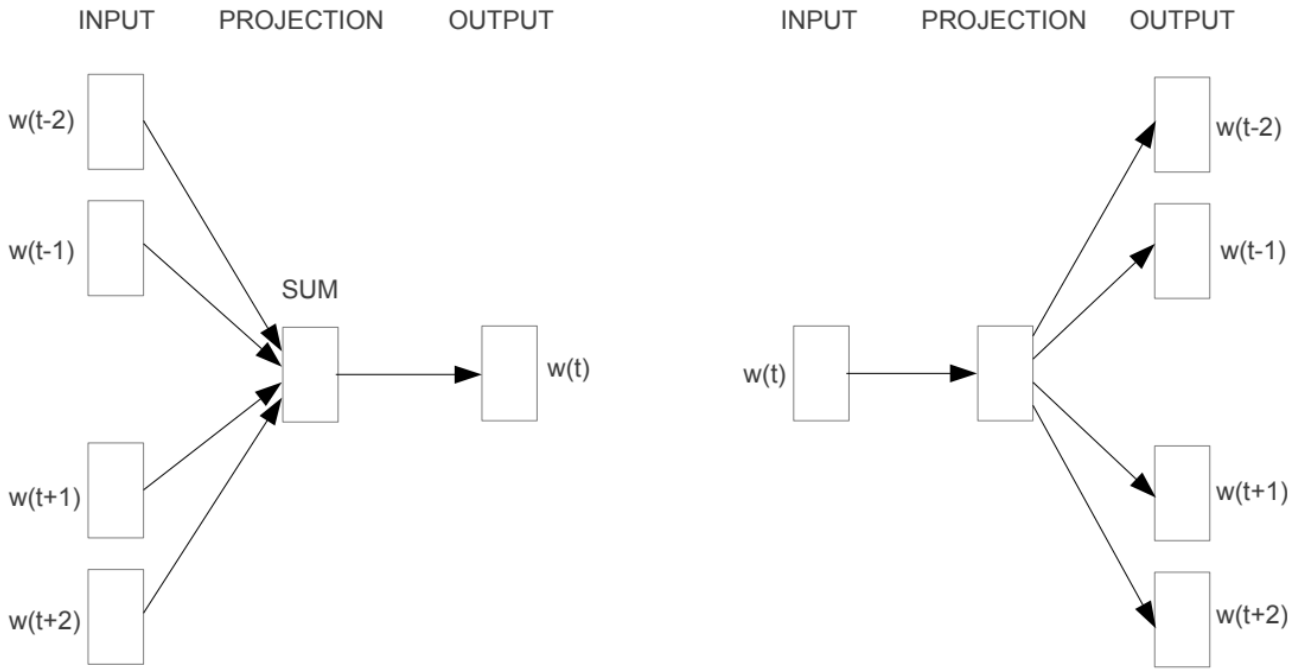


Fig. 1: The general pipeline of applying SSL methods to downstream tasks. The SSL models are first pre-trained on the unlabeled data and then fine-tuned, or directly evaluated, on the labeled data of the downstream tasks.

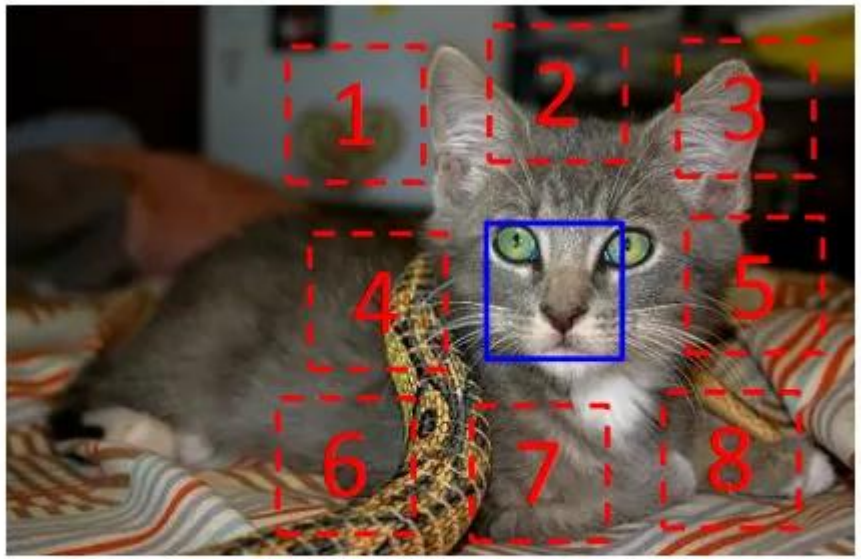
Types of Self Supervised Learning

- Context based
  - 基於資料本身的上下文
  - Word2vec 主要是利用語句的順序，例如 CBOW 通過前後的詞來預測中間的詞，而 Skip-Gram 通過中間的詞來預測前後的詞。

Fig. 4: Illustration of three common context-based methods: rotation, jigsaw, and colorization.

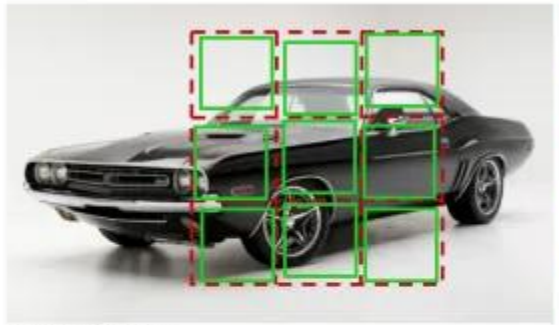


1: New model architectures. The CBOW architecture predicts the current word based on the xt, and the Skip-gram predicts surrounding words given the current word.



透過在圖像中選定兩個 patch，並利用其中一個 patch 來預測另一個 patch 的相對位置。

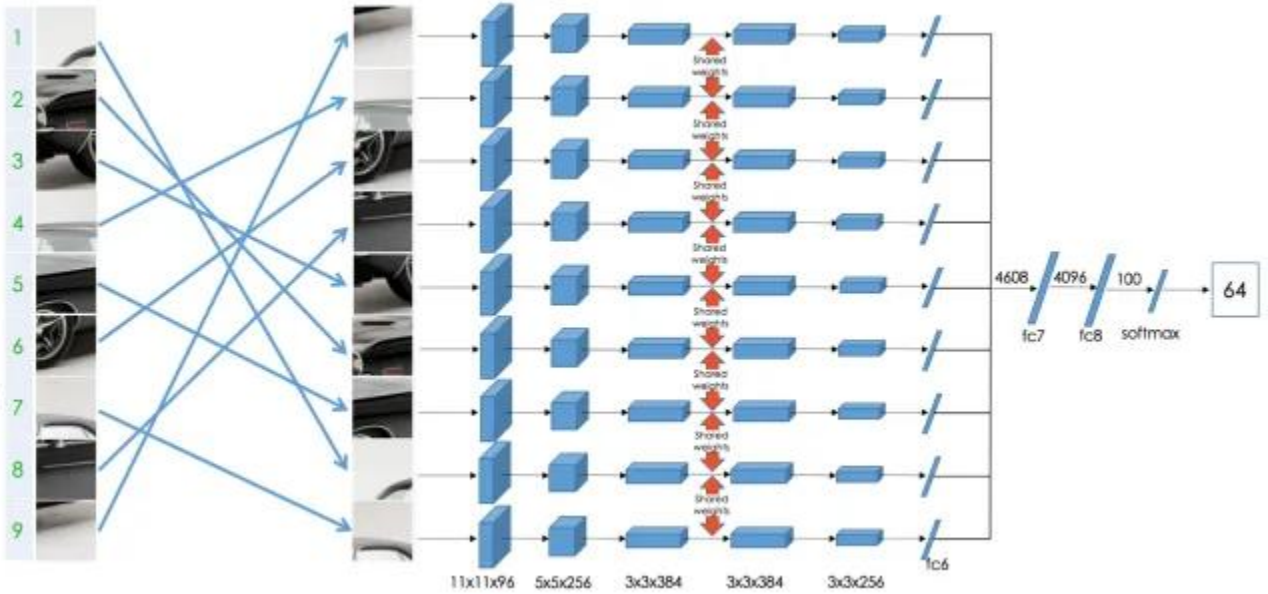
X = ( , ); Y = 3



Permutation Set

index	permutation
64	9,4,6,8,3,2,5,1,7

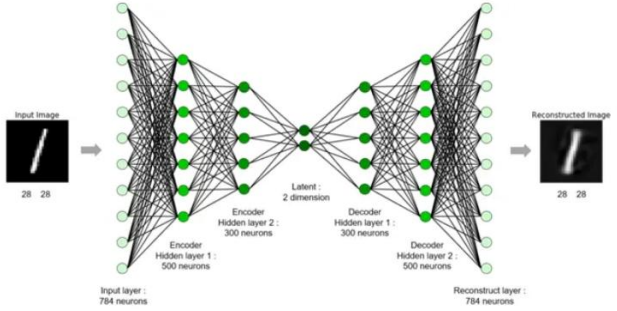
Reorder patches according to the selected permutation

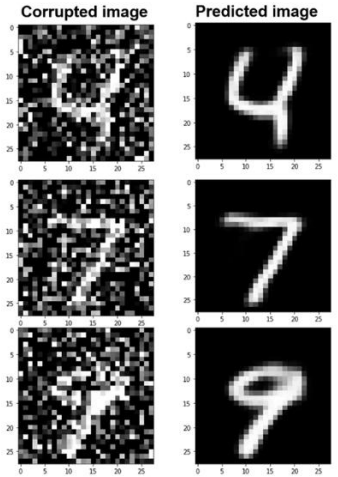


BERT AND MAE

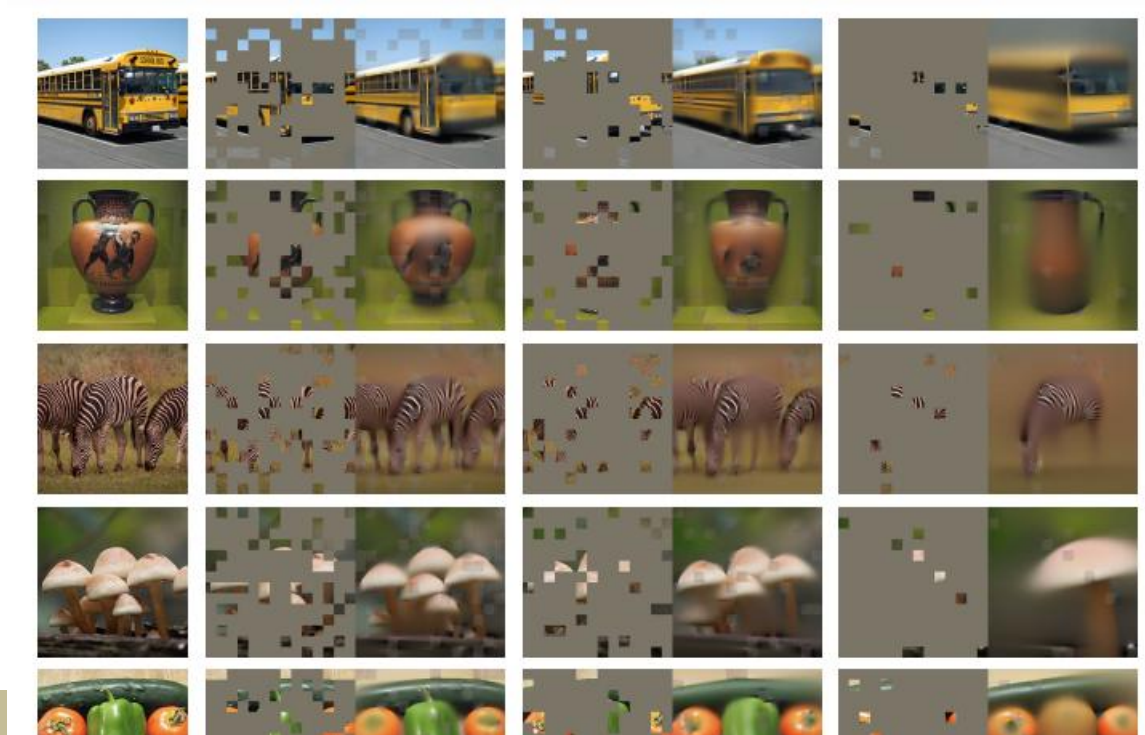
Masked Autoencoder (MAE) (CV 版BERT)

- Masked autoencoder
  - 是一種更通用的去噪自編碼器(Denoised Autoencoder)
  - 對NLP (BERT) 效果很好，但vision MAE發展為何還是落後於 NLP?
- 原因
  - CV 和 NLP 主流架構不同
    - ViT前，CV: CNN, NLP: transformer
  - 語言和圖片 (視頻) 的資訊密度不同
    - 語言:高度語意資訊
    - 圖片:空間上是高度冗餘 (可透過內插還原)
  - Decoder部分在 CV 和 NLP 中充當的角色不同
    - CV 領域，Decoder 作用是重建影像，Decoder輸出的語意(訊息)級別是低階的。
    - NLP 領域，Decoder 作用是重建單詞，Decoder輸出的是高階、富含資訊的語義級別





- Large random masked patch
  - 這種策略在很大程度上減少了冗餘，並創造了一個具有挑戰性的自監督任務，該任務需要超越低級圖像統計的整體理解



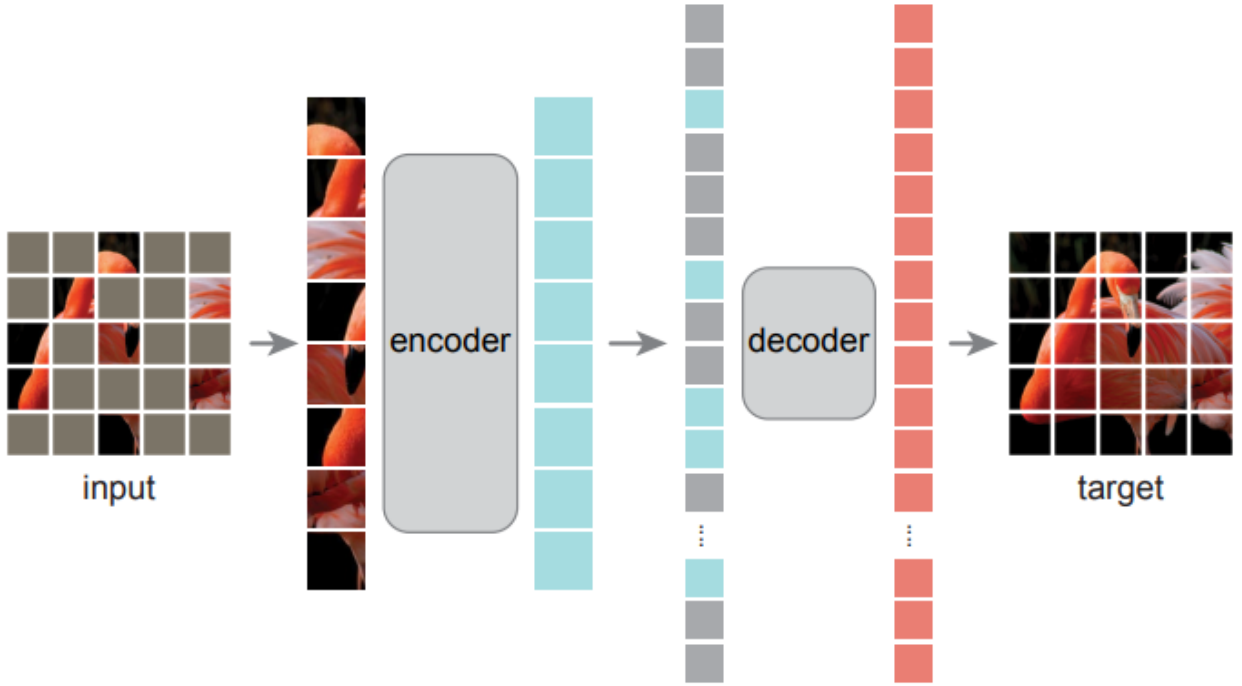


Figure 1. **Our MAE architecture**. During pre-training, a large random subset of image patches (*e.g.*, 75%) is masked out. The encoder is applied to the small subset of *visible patches*. Mask tokens are introduced *after* the encoder, and the full set of encoded patches and mask tokens is processed by a small decoder that reconstructs the original image in pixels. After pre-training, the decoder is discarded and the encoder is applied to uncorrupted images (full sets of patches) for recognition tasks.

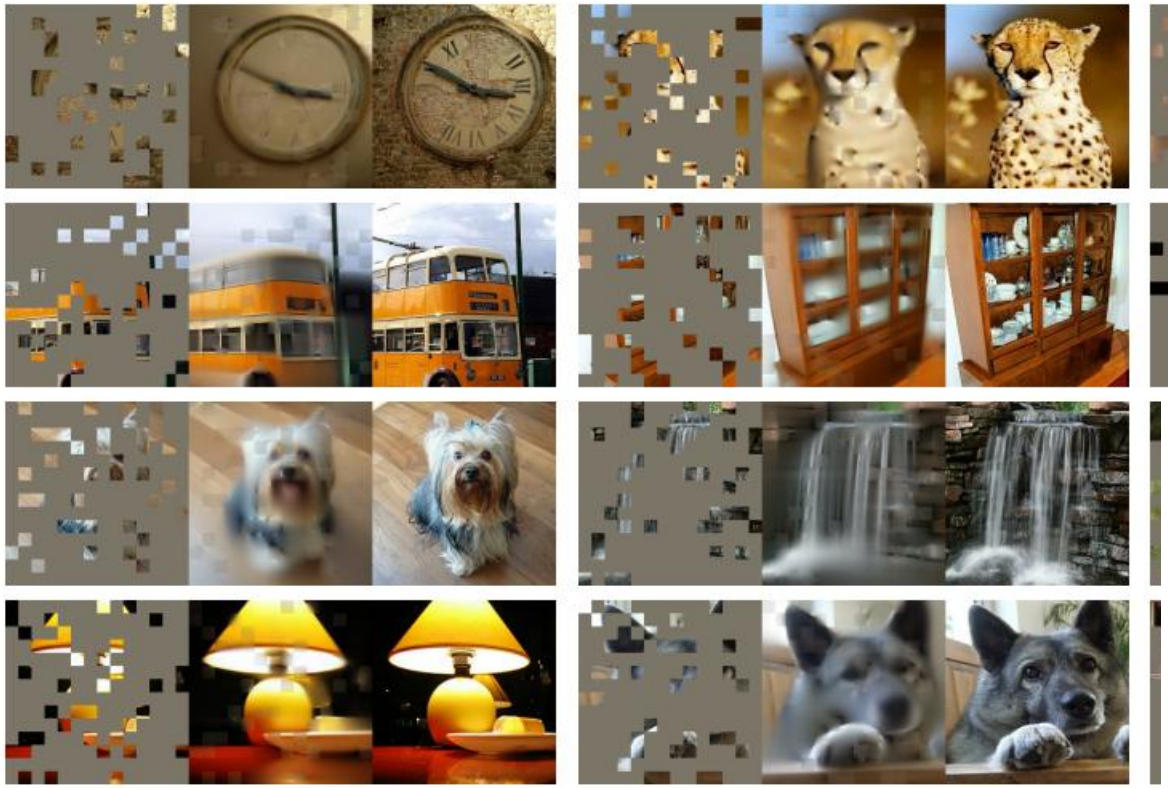


Figure 2. Example results on ImageNet *validation* images. For each t
(middle), and the ground-truth (right). The masking ratio is 80%, leavi
†*As no loss is computed on visible patches, the model output on visible patche*
patches to improve visual quality. We intentionally opt not to do this, so we can

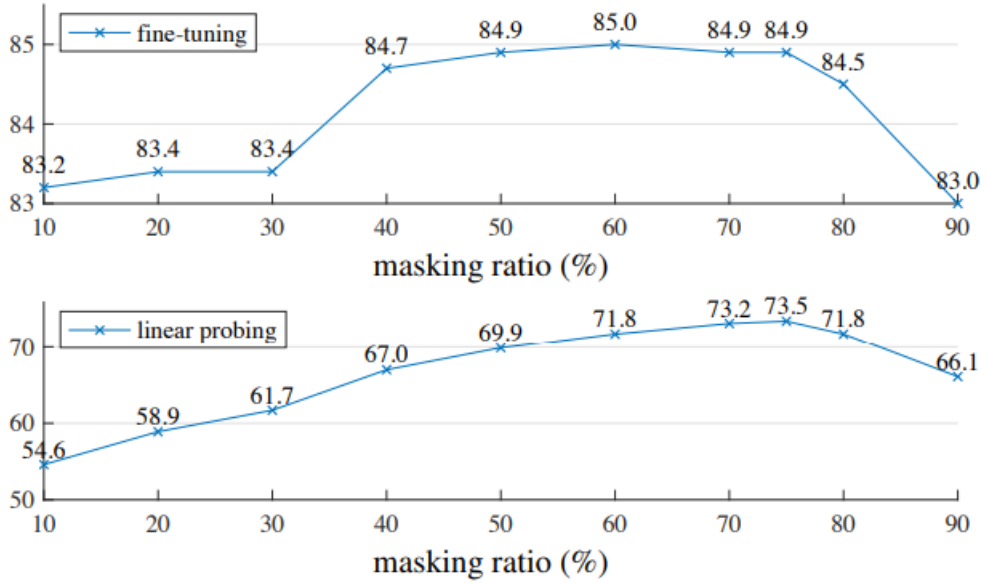


Figure 5. **Masking ratio**. A high masking ratio (75%) works well for both fine-tuning (top) and linear probing (bottom). The y-axes are ImageNet-1K validation accuracy (%) in all plots in this paper.

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

Table 3. **Comparisons with previous results on ImageNet-1K**. The pre-training data is the ImageNet-1K training set (except the tokenizer in BEiT was pre-trained on 250M DALLE data [50]). All self-supervised methods are evaluated by end-to-end fine-tuning. The ViT models are B/16, L/16, H/14 [16]. The best for each column is underlined. All results are on an image size of 224, except for ViT-H with an extra result on 448. Here our MAE reconstructs normalized pixels and is pre-trained for 1600 epochs.

		AP^{box}		AP^{mask}	
method	pre-train data	ViT-B	ViT-L	ViT-B	ViT-L
supervised	IN1K w/ labels	47.9	49.3	42.9	43.9
MoCo v3	IN1K	47.9	49.3	42.7	44.0
BEiT	IN1K+DALLE	49.8	**53.3**	44.4	47.1
MAE	IN1K	**50.3**	**53.3**	**44.9**	**47.2**

Table 4. **COCO object detection and segmentation** using a ViT Mask R-CNN baseline. All entries are based on our implementation. Self-supervised entries use IN1K data *without* labels. Mask AP follows a similar trend as box AP.

method	pre-train data	ViT-B	ViT-L
supervised	IN1K w/ labels	47.4	49.9
MoCo v3	IN1K	47.3	49.1
BEiT	IN1K+DALLE	47.1	53.3
MAE	IN1K	**48.1**	**53.6**

Table 5. **ADE20K semantic segmentation** (mIoU) using UperNet. BEiT results are reproduced using the official code. Other entries are based on our implementation. Self-supervised entries use IN1K data *without* labels.

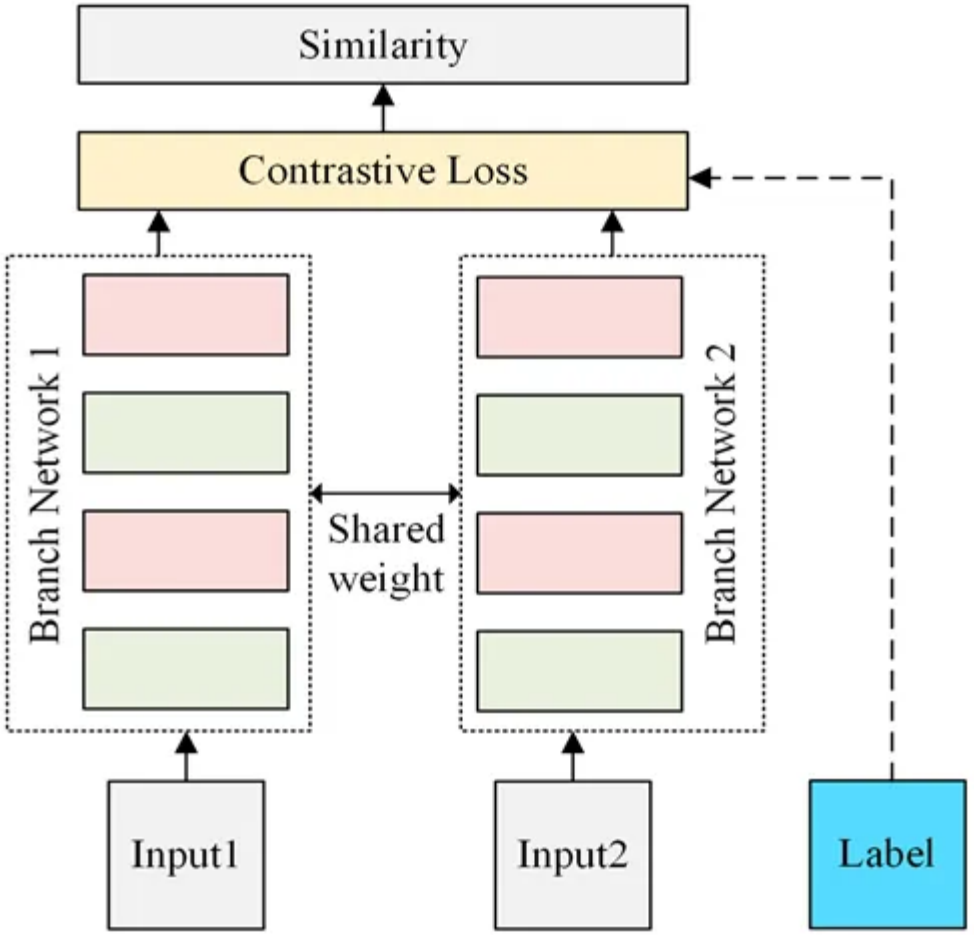
self-supervised pre-training on the ImageNet-1K (IN1K) [13] training set. Then we do supervised training to evaluate the representations with (i) end-to-end fine-tuning or (ii) linear probing

CONTRAST LEARNING

Contrastive Learning

- 概念
  - 相同類別的圖像間的相似度越高越好 (即距離盡可能地近)，
  - 不同類別的圖像相似度越低越好 (即距離盡可能地遠)，
  - 模型架構主要是使用 Siamese Network。
- 代表作有
  - FAIR 提出的 MoCov1、MoCov2、MoCov3，Google Brain 提出的 SimCLRv1、SimCLRv2

Contrastive Learning



對比學習的 5 大要素

- Data Augmentation:
  - 這是對比學習的基礎成分。提出新的數據擴增方法也很有學術價值，例如 MixUp, CutMix。
- Parallel Augmentation:
  - 指的是如何運用 noise data (擴增數據) 提取特徵。主要有兩種方式：端到端 End-to-End 與 動量編碼器 Momentum Encoder
- Architecture:
  - 指的是如何操作正負樣本來計算損失函數的架構設計。主要可分 4類：End-to-End, Memory Bank, MoCo, Clustering。
- Loss Function:
  - 損失函數的設計。最經典的可能是 InfoNCE。
- Data-Modal:
  - 不同模態的數據會衍生不同的對比學習方法。

	Low-Level Targets				High-Level Targets			Self-Distillation		Contrastive / Multi-modal Teacher	
Algorithm	ViT [5]	MAE [70]	SimMIM [101]	Maskfeat [106]	BEiT [99]	CAE [100]	PeCo [107]	data2vec [108]	SdAE [109]	MimCo [110]	BEiT v2 [111]
Target	Raw Pixel			HOG	VQ-VAE		VQ-GAN	self		MoCo v3	CLIP

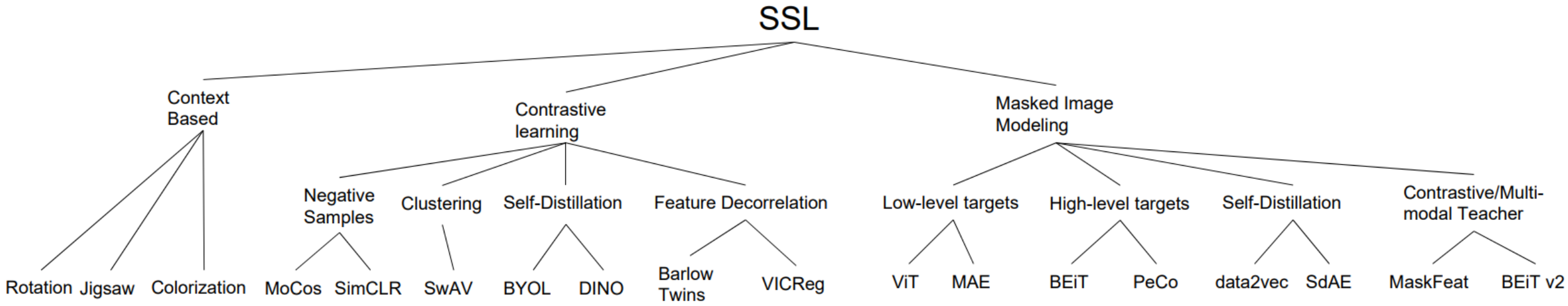


Fig. 8: Several representative pretext tasks of SSL.

TABLE 4: Experimental results of the tested algorithms for linear classification and transfer learning tasks. DB denotes the default batch size. The symbol "-" indicates the absence or unavailability of the data point in the respective paper. The subscripts A, R, and V represent AlexNet, ResNet-50, and ViT-B, respectively. The superscript "e" indicates the utilization of extra data, specifically VOC2012.

Methods	Linear Probe	Fine-Tuning	VOC_det	VOC_seg	COCO_det	COCO_seg	ADE20K_seg	DB
Random:	17.1_A [8]	-	60.2^e_R [69]	19.8_A [8]	36.7_R [50]	33.7_R [50]	-	-
R50 Sup	76.5 [68]	76.5 [68]	81.3^e [69]	74.4 [67]	40.6 [50]	36.8 [50]	-	-
ViT-B Sup	82.3 [70]	82.3 [70]	-	-	47.9 [70]	42.9 [70]	47.4 [70]	-
Context-Based:								
Jigsaw [8]	45.7_R [68]	54.7	61.4_R [42]	37.6	-	-	-	256
Colorization [38]	39.6_R [68]	40.7 [7]	46.9	35.6	-	-	-	-
Rotation [7]	38.7	50.0	54.4	39.1	-	-	-	128
CL Based on Negative Examples:								
Examplar [132]	31.5 [48]	-	-	-	-	-	-	-
Instdisc [48]	54.0	-	65.4	-	-	-	-	256
MoCo v1 [50]	60.6	-	74.9	-	40.8	36.9	-	256
SimCLR [52]	73.9_V [82]	-	81.8^e [69]	-	37.9 [69]	33.3 [69]	-	4096
MoCo v2 [51]	72.2 [69]	-	82.5^e	-	39.8 [56]	36.1 [56]	-	256
MoCo v3 [82]	76.7	83.2	-	-	47.9 [70]	42.7 [70]	47.3 [70]	4096
CL Based on Clustering:								
SwAV [68]	75.3	-	82.6^e [56]	-	41.6	37.8 [56]	-	4096
CL Based on Self-distillation:								
BYOL [67]	74.3	-	81.4^e [69]	76.3	40.4 [56]	37.0 [56]	-	4096
SimSiam [69]	71.3	-	82.4^e [69]	-	39.2	34.4	-	512
DINO [83]	78.2	83.6 [98]	-	-	46.8 [100]	41.5 [100]	44.1 [99]	1024
CL Based on Feature Decorrelation:								
Barlow Twins [55]	73.2	-	82.6^e [56]	-	39.2	34.3	-	2048
VICReg [56]	73.2	-	82.4^e	-	39.4	36.4	-	2048

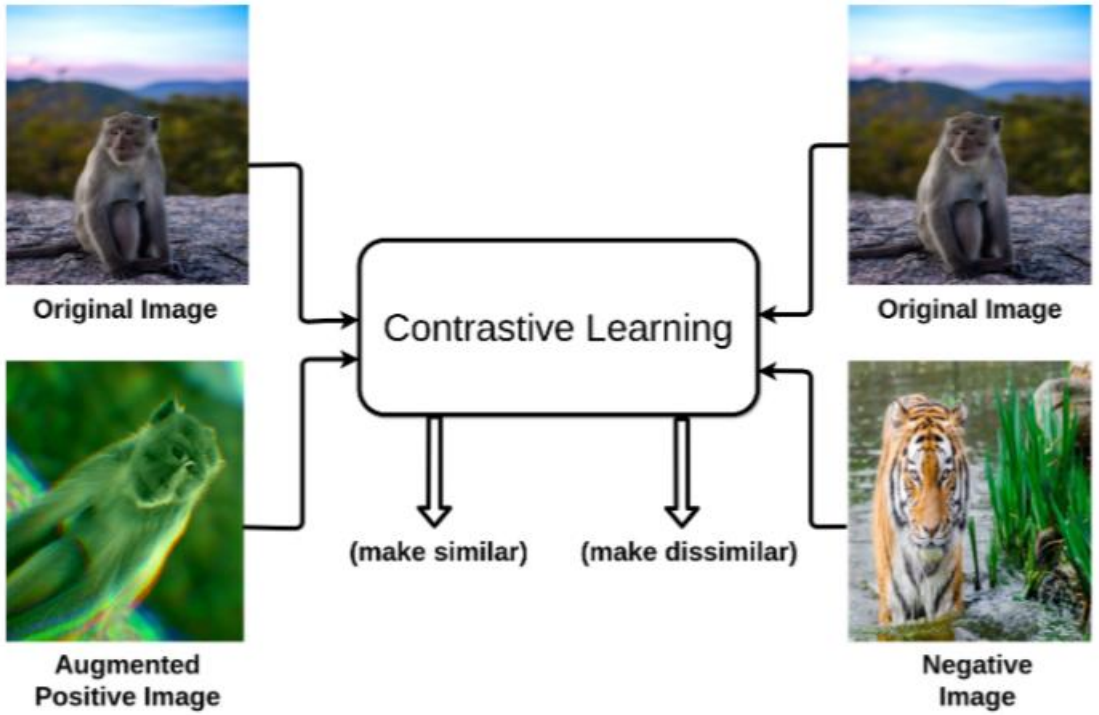
TABLE 4: Experimental results of the tested algorithms for linear classification and transfer learning tasks. DB denotes the default batch size. The symbol "-" indicates the absence or unavailability of the data point in the respective paper. The subscripts A, R, and V represent AlexNet, ResNet-50, and ViT-B, respectively. The superscript "e" indicates the utilization of extra data, specifically VOC2012.

Methods	Linear Probe	Fine-Tuning	VOC_det	VOC_seg	COCO_det	COCO_seg	ADE20K_seg	DB
Random:	17.1_A [8]	-	60.2^e_R [69]	19.8_A [8]	36.7_R [50]	33.7_R [50]	-	-
R50 Sup	76.5 [68]	76.5 [68]	81.3^e [69]	74.4 [67]	40.6 [50]	36.8 [50]	-	-
ViT-B Sup	82.3 [70]	82.3 [70]	-	-	47.9 [70]	42.9 [70]	47.4 [70]	-
Masked Image Modeling (ViT-B by default):								
Context Encoder [104]	21.0_A [7]	-	44.5_A [7]	30.0_A	-	-	-	-
BEiT v1 [99]	56.7 [111]	83.4 [98]	-	-	49.8 [70]	44.4 [70]	47.1 [70]	2000
MAE [70]	67.8	83.6	-	-	50.3	44.9	48.1	4096
SimMIM [101]	56.7	83.8	-	-	52.3_{Swin-B} [244]	-	52.8_{Swin-B} [244]	2048
PeCo [107]	-	84.5	-	-	43.9	39.8	46.7	2048
iBOT [98]	79.5	84.0	-	-	51.2	44.2	50.0	1024
MimCo [110]	-	83.9	-	-	44.9	40.7	48.91	2048
CAE [100]	70.4	83.9	-	-	50	44	50.2	2048
data2vec [108]	-	84.2	-	-	-	-	-	2048
SdAE [109]	64.9	84.1	-	-	48.9	43.0	48.6	768
BEiT v2 [111]	80.1	85.5	-	-	-	-	53.1	2048

CONTRAST LEARNING

Contrast Learning

- One type of self-supervised learning



Basic intuition behind contrastive learning paradigm: push original and augmented images closer and push original and negative images away

關鍵: 如何定義正樣本和負樣本 (想的到的組合都可以)

History

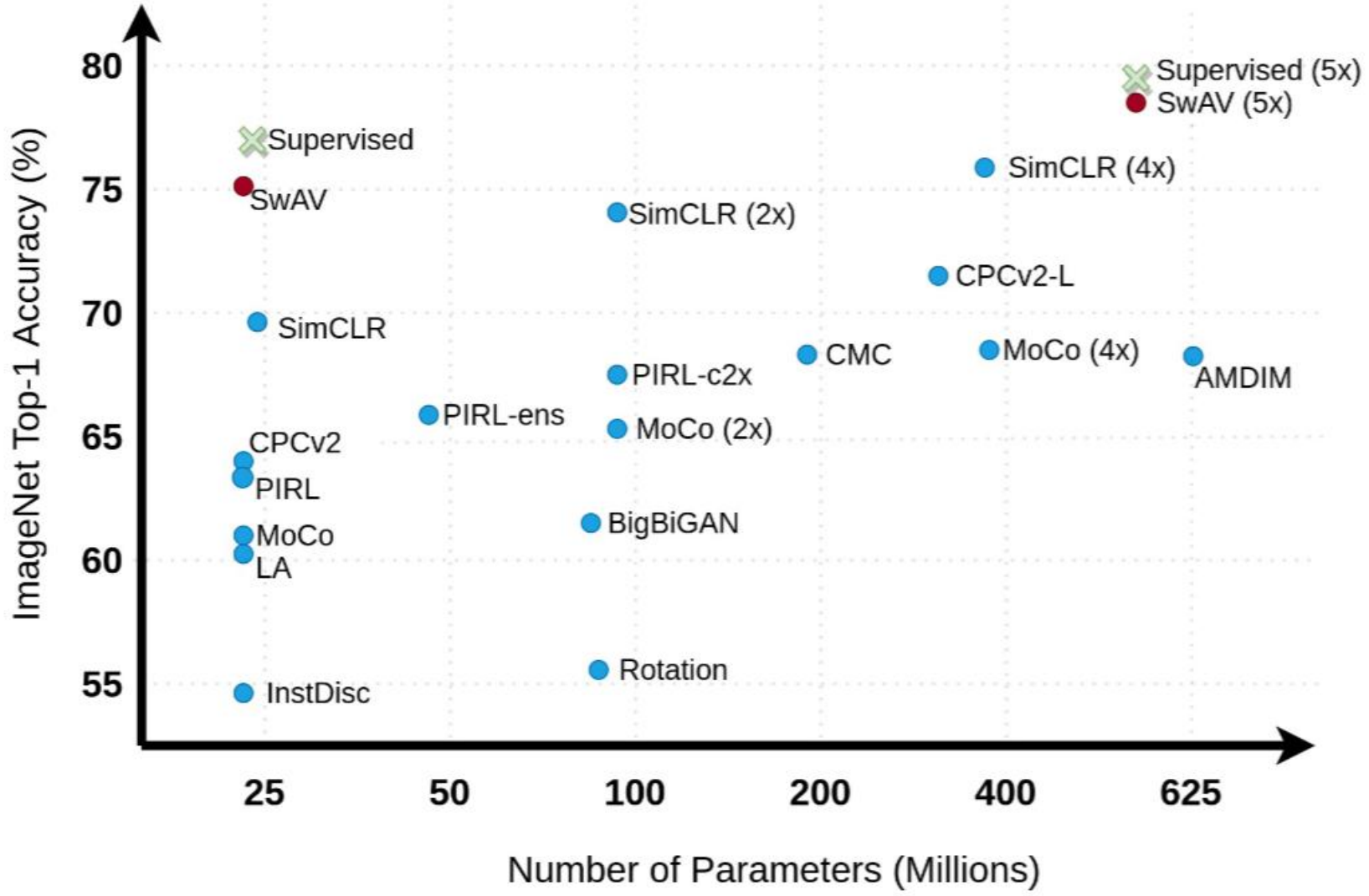
- 第一階段：百花齊放（2018 年至 2019 年中期）
- 此階段方法、模型、目標函數和代理任務尚未統一，呈現多元發展的狀態。
- 代表性工作包括：
  - InstDisc (Instance Discrimination)：
    - 奠定了對比學習的基礎，提出了「個體判別」代理任務，即將每張圖片視為獨立類別。
    - 使用 Memory Bank 儲存大量負樣本特徵，並引入動量更新機制。
  - InvaSpread：
    - 可視為 SimCLR 的前身，採用端到端學習，僅使用單一編碼器，正負樣本來自同一個 minibatch。
    - 由於缺乏 SimCLR 的數據增強、MLP Projector 和大 batch size，效果不如 SimCLR 顯著。
  - CPC (Contrastive Predictive Coding)： 使用生成式任務
    - 使用「預測」作為代理任務，利用上下文特徵預測未來時刻的特徵輸出。
    - 可應用於圖像、音頻、文本等多種數據類型。
  - CMC (Contrastive Multiview Coding)：
    - 提出「多視角」概念，將同一物體的不同視角視為正樣本，例如：同一張圖片的 RGB 圖像、深度資訊、表面法線和分割圖像。
    - 證明了對比學習的靈活性，可應用於多視角、多模態學習，為後續 CLIP 等模型的發展奠定了基礎。

- 第二階段：雙雄爭霸（2019 年中期至 2020 年中期）
- 以 MoCo 和 SimCLR 為代表，這兩個模型在 ImageNet 上取得了突破性的成果，推動了對比學習的快速發展。
- MoCo：
  - 將對比學習歸納為字典查詢問題，提出了「queue」和「動量編碼器」來構建更大、更一致的字典。
  - 繼承了 InstDisc 的許多設計，包括使用 Res 50、128 維特徵、InfoNCE loss 和數據增強方式。
  - MoCo v2 融合了 SimCLR 的 MLP Projection head 和更強的數據增強技巧，進一步提升了性能。
- SimCLR：
  - 簡化了對比學習流程，僅使用單一編碼器，正負樣本來自同一個 minibatch。
  - 提出 MLP Projector，將編碼器輸出的特徵進行非線性變換，顯著提升了性能。
  - 採用更強的數據增強、更大的 batch size 和更長的訓練時間，進一步提升了性能。 *
  - SimCLR v2 主要關注半監督學習，並通過更大的模型、更深的 MLP Projection head 和動量編碼器提升了自監督性能。
- 其他重要工作：
  - SwAV 結合了聚類和對比學習，並提出了 multi-crop 技巧。
  - CPC v2 和 InfoMin 分別對 CPC 和 CMC 進行了改進。

- 第三階段：擺脫負樣本（2020 年中期至 2021 年初）
  - 以 BYOL 和 SimSiam 為代表，探索了不使用負樣本的對比學習方法。
- BYOL：
  - 提出「自舉你的潛在表示」概念，不使用任何形式的負樣本，僅通過預測任務進行模型訓練。
  - 採用動量編碼器和 MLP Projector，並使用 MSE loss 作為目標函數。
  - 引發了關於 Batch Normalization 在 BYOL 中作用的討論，最終證明 Batch Normalization 主要作用是穩定訓練，而非提供隱式負樣本。
- SimSiam：
  - 進一步簡化了 BYOL，不使用動量編碼器和大 batch size，僅依靠 Stop Gradient 操作和對稱性 loss 避免模型坍塌。
  - 將 SimSiam 解釋為一種 EM 演算法，並提出了 k-means 聚類的解釋。
- Barlow Twins：
  - 提出新的目標函數，通過比較正樣本特徵的關聯矩陣和單位矩陣的相似性來進行模型訓練。

- 第四階段：Transformer 時代（2021 年至今）
  - Vision Transformer 的出現，為對比學習帶來了新的挑戰和機遇，研究重點轉向如何有效地訓練自監督 Vision Transformer 模型。
- MoCo v3：
  - 將 MoCo 框架擴展到 Vision Transformer，發現 ViT 在自監督訓練中存在不穩定性問題。
  - 提出凍結 patch projection layer 的技巧，有效解決了訓練不穩定問題。
- DINO：
  - 將自監督 Vision Transformer 訓練框架解釋為自蒸餾，並提出了 centering 操作來避免模型坍塌。
  - 發現自監督 Vision Transformer 可以準確地捕捉物體輪廓，展現出強大的潛力。

Contrast Learning



Unsupervised Feature Learning via Non-Parametric Instance Discrimination

- InstDisc
  - 觀察
    - 分類器輸出機率較高的類別通常是與該圖片內容相似的物體。 例如，將一張豹子的圖片輸入分類器，分類結果中排名靠前的類別可能是獵豹、雪豹等，而排名靠後的類別則與豹子無關。
  - 原因
    - 並非僅僅因為相似的圖片擁有相似的語義標籤，
    - 而是因為它們在視覺上非常相似。
    - 因此提出將每個個體（instance）都看作是一個獨立的類別，並希望學習一種能夠區分每個個體的特徵表示方法。
      - 每個圖片都被視為一個獨立的類別

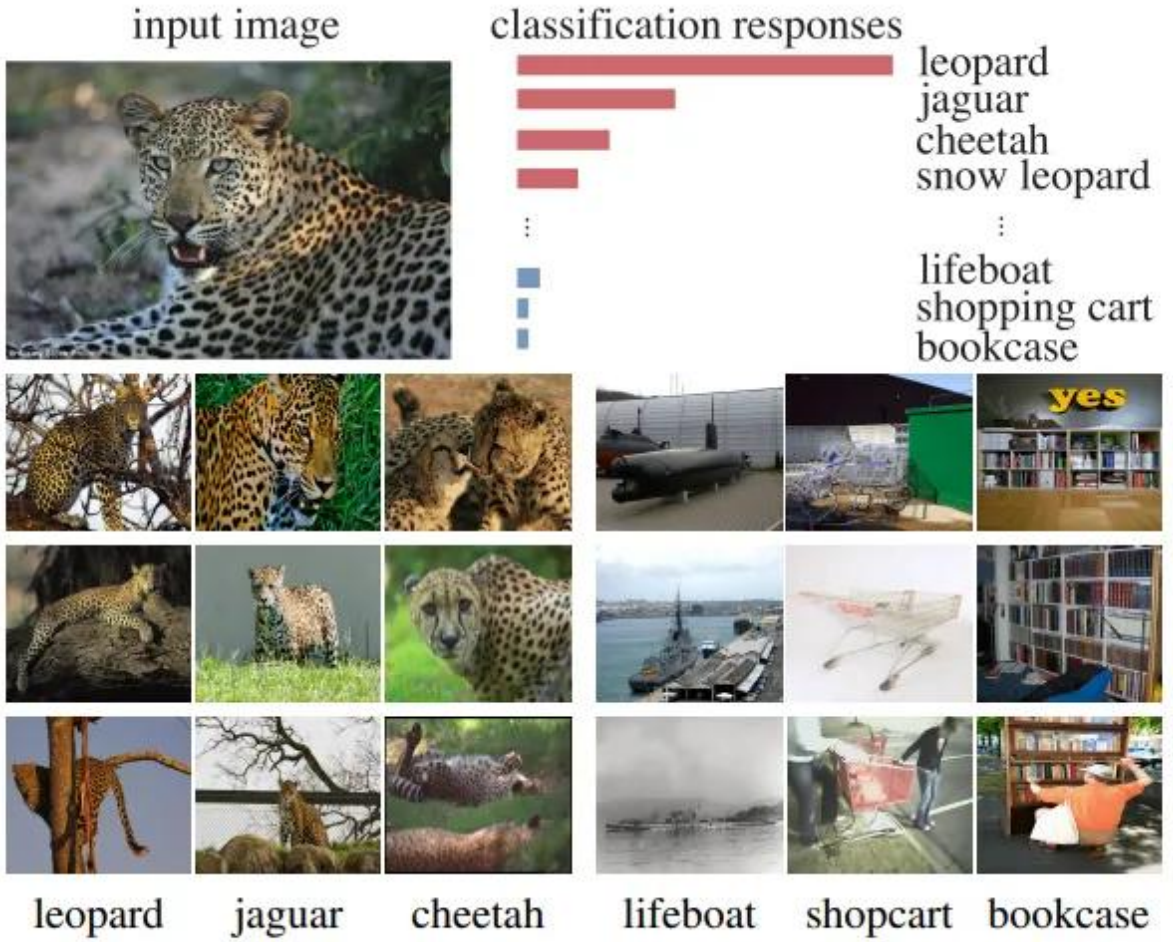
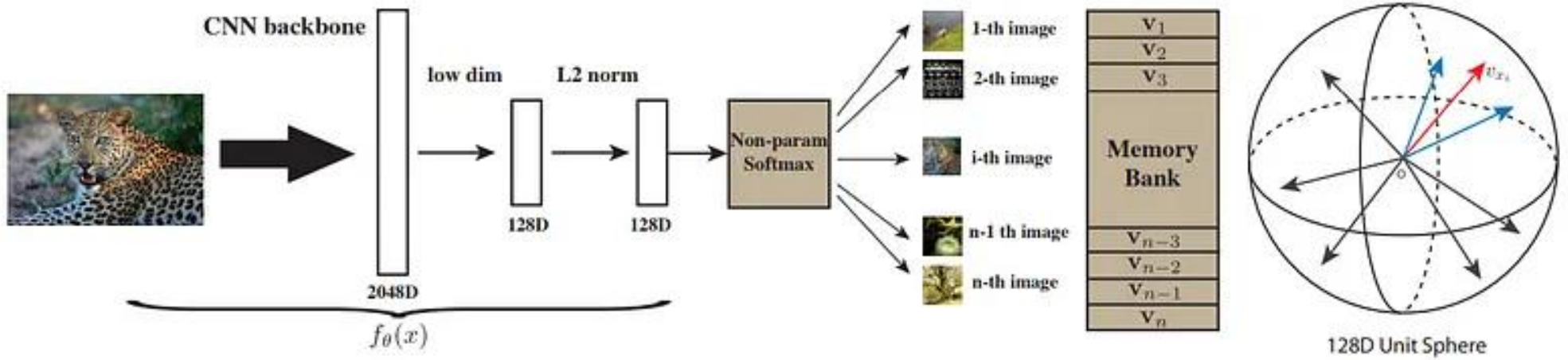


Figure 1: Supervised learning results that motivate our unsupervised approach. For an image from class *leopard*, the classes that get highest responses from a trained neural net classifier are all

Arxiv: 1805.01978

- 正樣本
  - 圖片本身和其圖片的數據增強，
- 負樣本
  - 數據集中其餘的圖片
- 使用 memory bank 存儲所有圖片的特徵向量 ，訓練過程更新
- 使用 proximal regularization 對特徵向量進行動量式更新，類似 MoCo 的動量編碼器
- NCE(Noise Contrastive Estimation) loss



BatchSize為 256，那就會有 256 張正樣本進入 CNN Encoder。而負樣本從 Memory Bank 中隨機採樣出 4096 張(固定

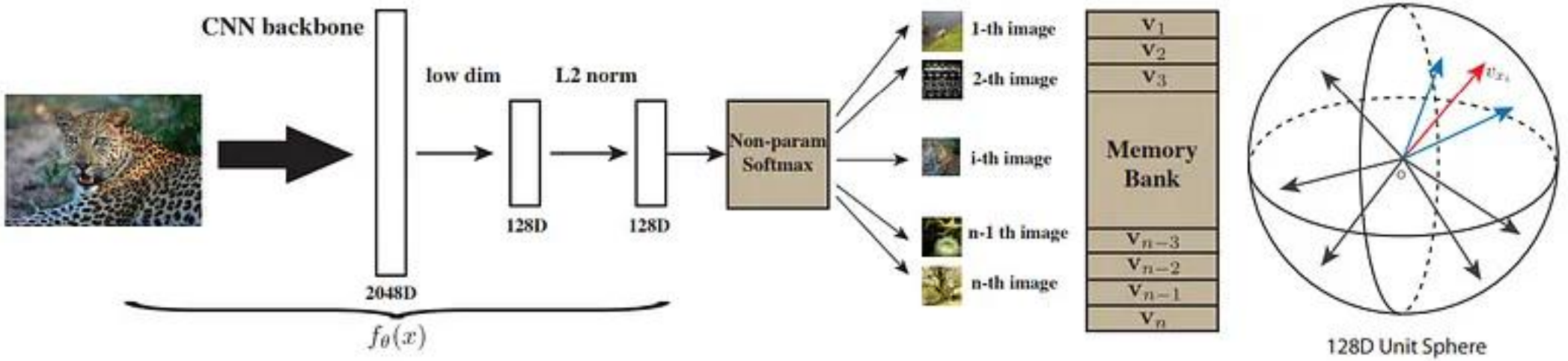


Figure 2: The pipeline of our unsupervised feature learning approach. We use a backbone CNN to encode each image as a feature vector, which is projected to a 128-dimensional space and L2 normalized. The optimal feature embedding is learned via instance-level discrimination, which tries to maximally scatter the features of training samples over the 128-dimensional unit sphere.

- BatchSize為 256，那就會有 256 張正樣本進入 CNN Encoder。而負樣本從 Memory Bank 中隨機採樣出 4096 張，

- 訓練目標
  - 訓練一個網路，對輸入影像編碼，編碼後的特徵，在特徵空間裡能夠盡可能地分開，因為對於個體判別任務來說，每張圖片都是自己的類別
  - 正樣本
    - 圖片本身和其圖片的data augmentation
  - 負樣本
    - 數據集中其餘的圖片
- 特徵向量
  - 128 dimenstion
  - 使用 memory bank 存儲所有圖片的特徵向量 ，訓練過程更新特徵向量
  - 使用 proximal regularization 對特徵向量進行動量式更新，類似 MoCo 的動量編碼器
- NCE(Noise Contrastive Estimation) loss

Unsupervised Embedding Learning via Invariant and Spreading Instance Feature

- InvaSpread (正負樣本都來自同一個mini-batch)
  - 數據集中抽取一個 mini batch (e.g. N=256)的圖像，對所有圖片進行數據增強
    - 用mini-batch 為單位去做，可以不用memory bank，用一個編碼器做到end-to-end learning
  - 正樣本為一張圖片及它數據增強的圖片 (pos: 2)
  - 負樣本為剩下的圖片及它們數據增強的圖片。 (neg: 2N-2, (256-1)*2)

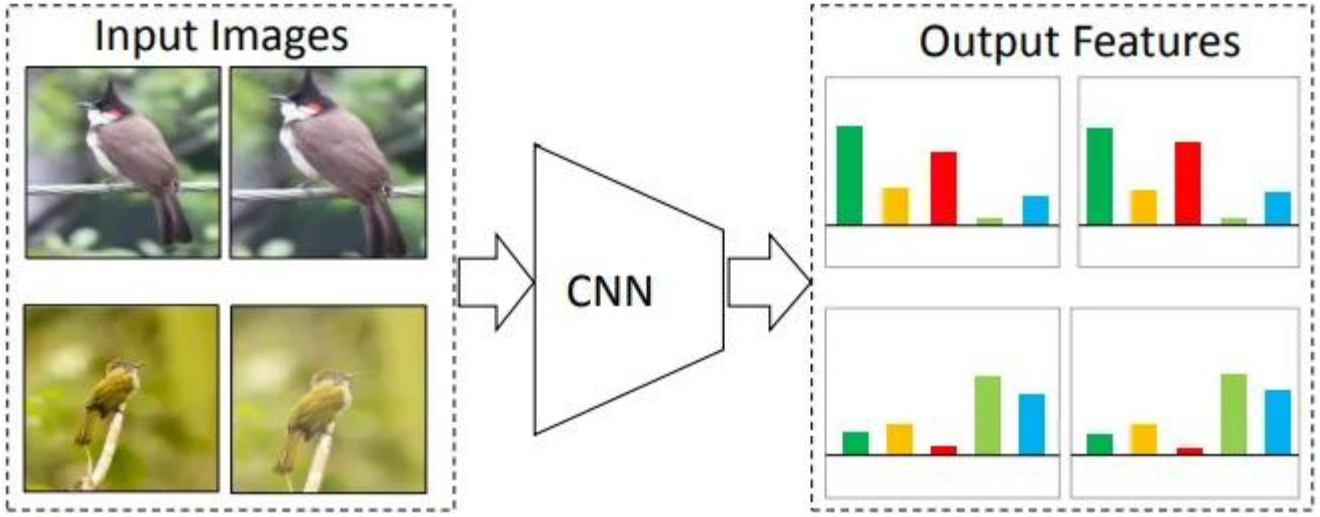


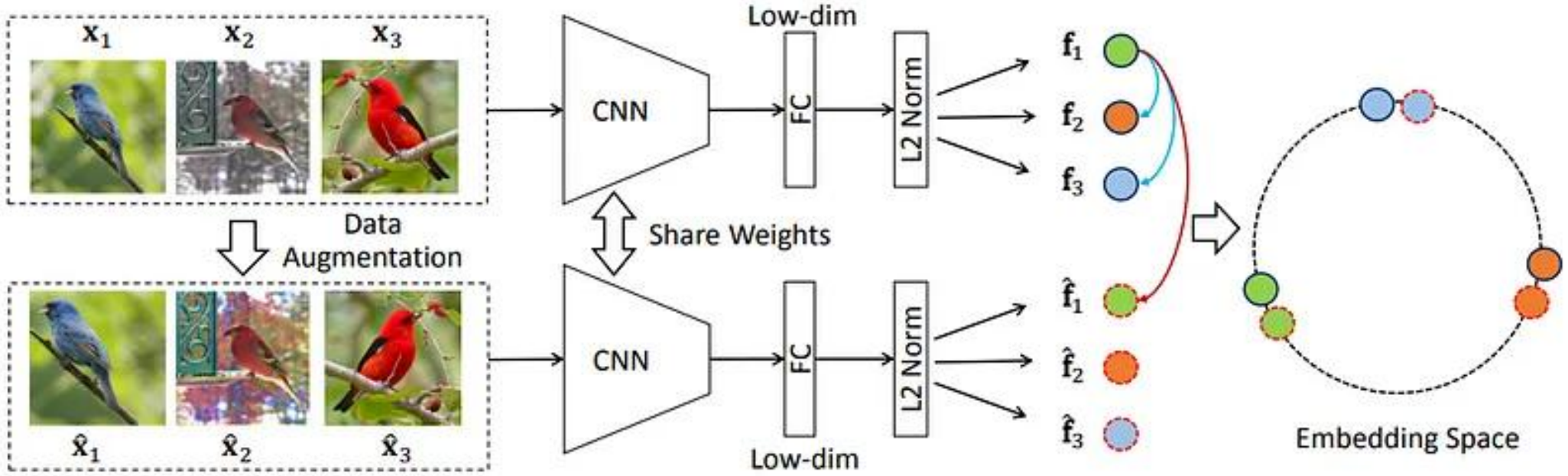
Figure 1: Illustration of our basic idea. The features of the same

一樣用instance discrimination

對於相似的圖片
相似的物體呢
它的特徵應該保持不變性

但是對於不相似的物體
或者完全不沾邊的物體呢
它的特徵應該盡可能的分散開

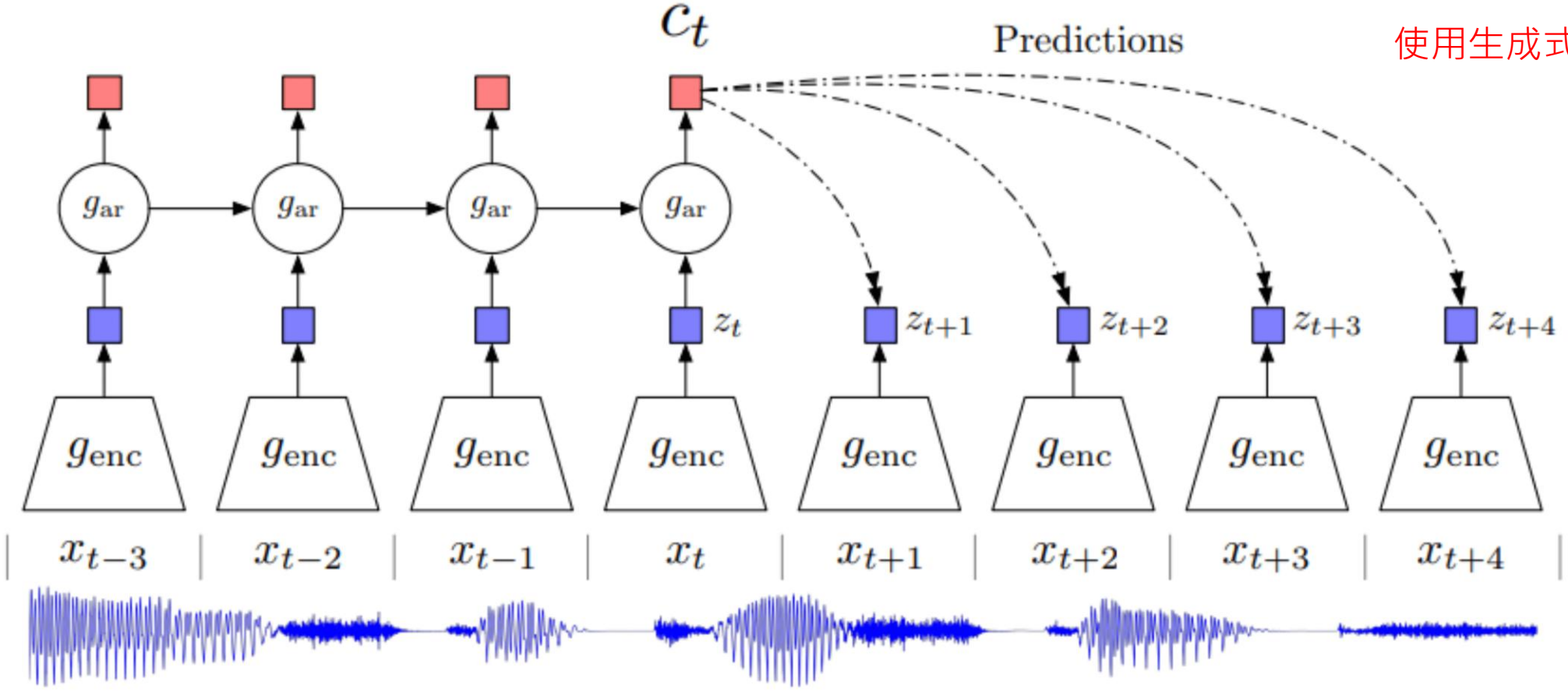
- InvaSpread
  - 沒有使用數據結構 Memory Bank 去存儲負樣本，InvaSpread 只用一個 Encoder 進行端到端的學習，可以在同一個 mini batch 中實現正負樣本的對比學習。InvaSpread 可以被理解成是 SimCLR 的前身。
- InvaSpread 跟 SimCLR 這麼像，為什麼沒有取得炸裂的成果呢
  - 是因為做對比學習的時候，負樣本最好是足夠多，字典必須足夠大
  - InvaSpread 的作者沒有 GPU，所以 Batchsize 設置為 256，意味著負樣本只有 (256*2-2) = 510 個，負樣本屬實太少囉！



CPC

正樣本： 未來時刻的輸入通過編碼器後得到的未來時刻的特徵輸出。
負樣本： 可以是任意選取的輸入通過編碼器得到的特徵輸出，它們都應該與你的預測不相似。

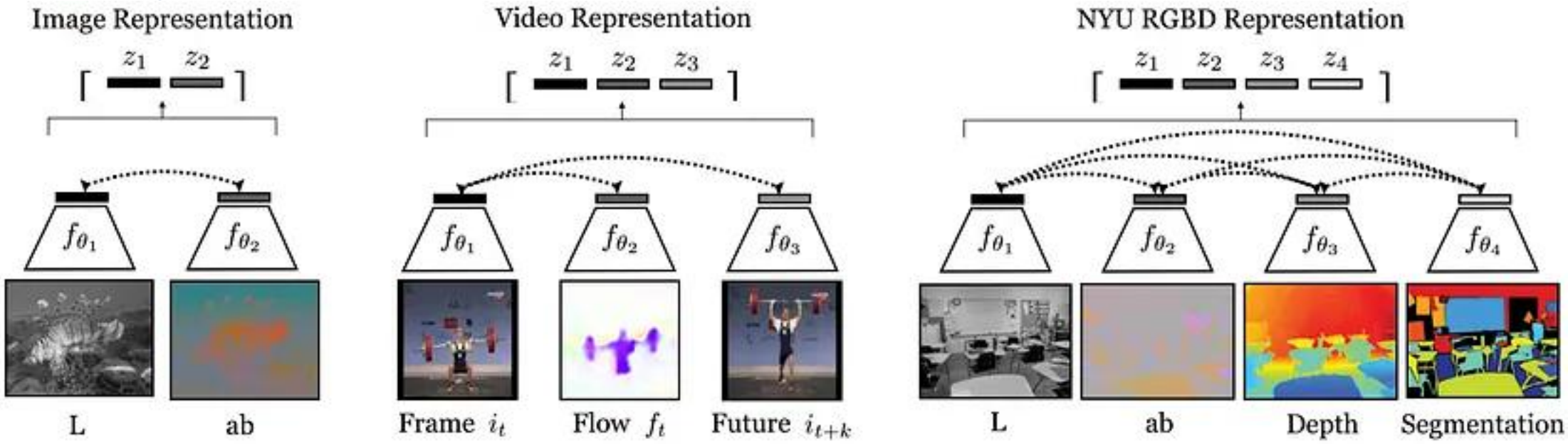
- Contrastive Predictive Coding (CPC)
  - Although the figure shows audio as input, similar setup can be used for videos, images, text etc



使用生成式任務 (預測)

CMC: Contrastive Multiview Coding

- 利用多個視角進行對比學習的方法
  - 其核心概念是，一個物體可以從多個不同的角度或模態進行觀察，而這些不同的視角都包含著關於該物體的重要信息。
  - CMC 方法旨在學習一種能夠捕捉所有視角下關鍵因素的特征表示，從而使其具有視角不變性。
  - 來自同一物體的不同視角被視為正樣本，而來自不同物體的視角被視為負樣本



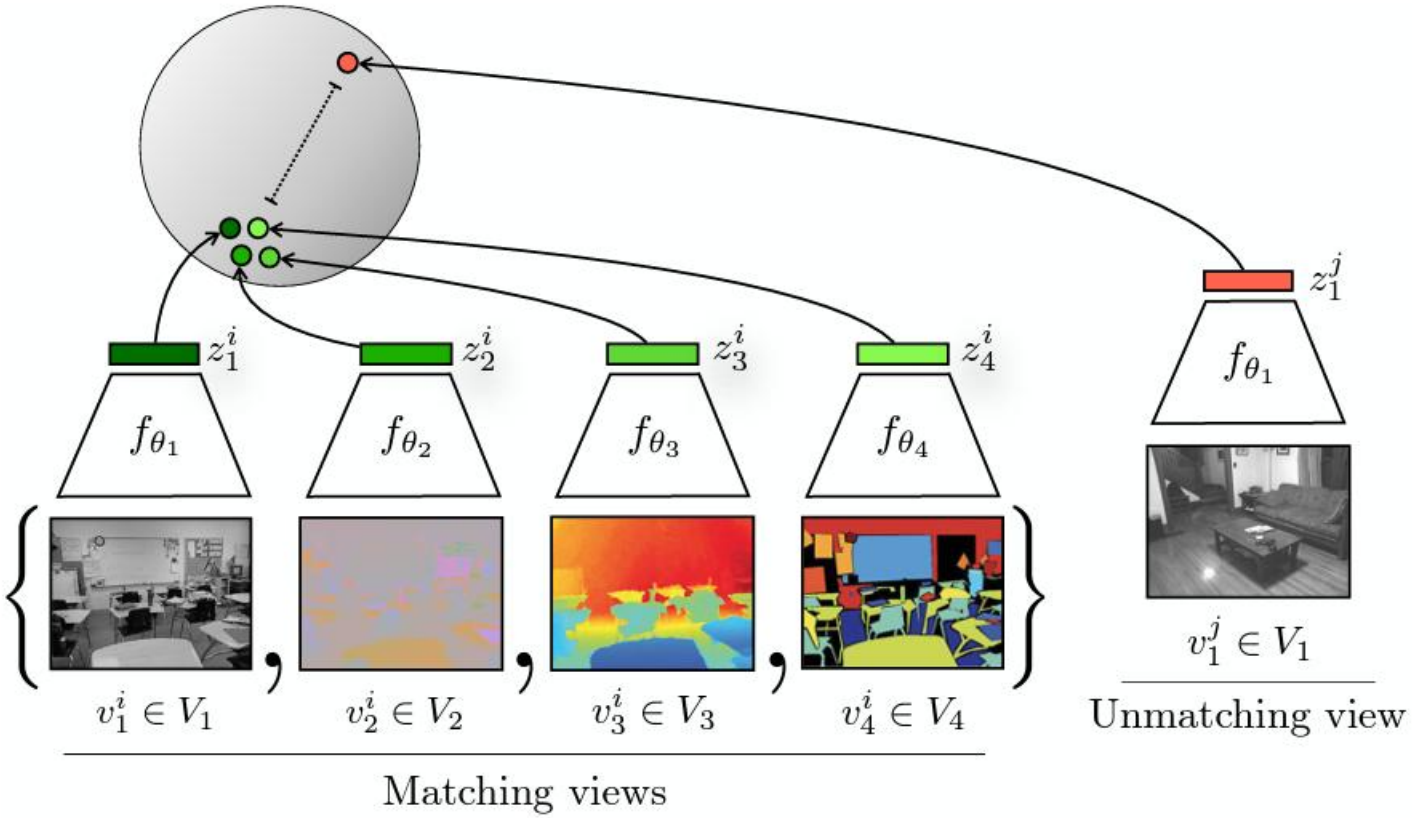
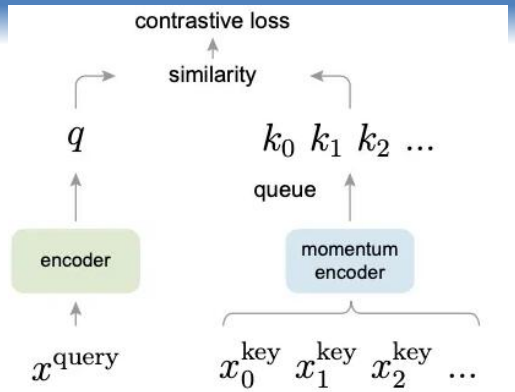


Figure 1: Given a set of sensory views, a deep representation is learnt by bringing views of the *same* scene together in embedding space, while pushing views of *different* scenes apart. Here we show and example of a 4-view dataset (NYU RGBD [53]) and its learned representation. The encodings for each view may be concatenated to form the full representation of a scene.

MOCO v1

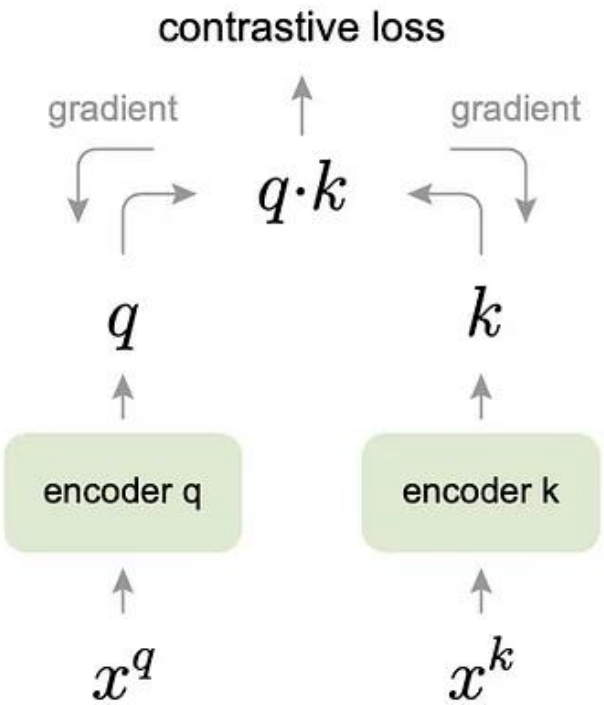


- 基於 instDisc 的架構進行改進
  - 把contrast learning轉化為字典查找
  - 構建了一個帶有queue和moving average Encoder 的 Dynamic Dictionary

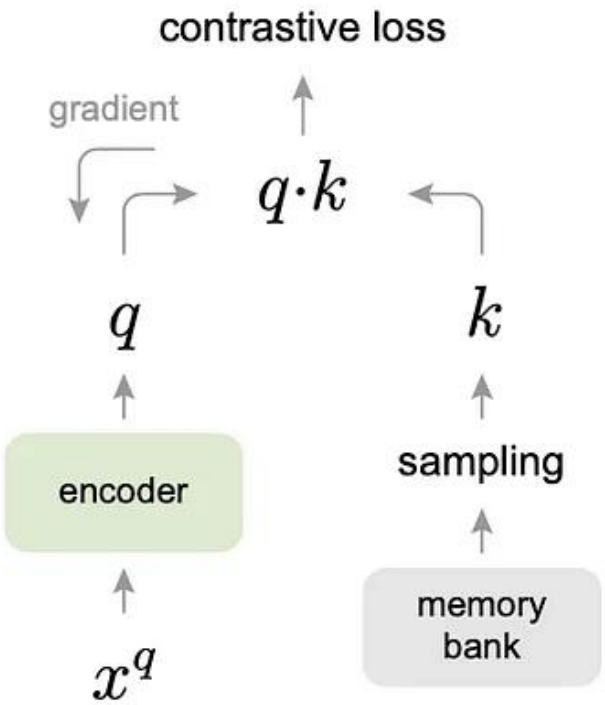
區別

  - 字典構建方法：Queue取代了InstDisc 中的Memory Bank
    - Memory Bank 的大小固定，並且更新速度較慢，這限制了對比學習的效能
    - 用queue，字典的大小可以動態調整
  - 特徵更新方式：Momentum (moving average) Encoder
    - MoCo 使用動量編碼器來更新特徵，而不是像 InstDisc 那樣直接更新 Memory Bank 中的特徵。 這種動量更新的方式可以使字典中的特徵更加一致，有利於對比學習。
    - 使用query可以使字典變大，但它也使得通過反向傳播更新key-Encoder變得更難。
    - 直接複製，從Query encoder 複製weight 到Key encoder，效果不好，
    - 假設這種失敗是由快速變化的編碼器引起的，這降低了我們希望的representation 一致性(字典不一致問題)。 提出Momentum update 來解決這個問題
    - 只有參數 θq 通過反向傳播更新

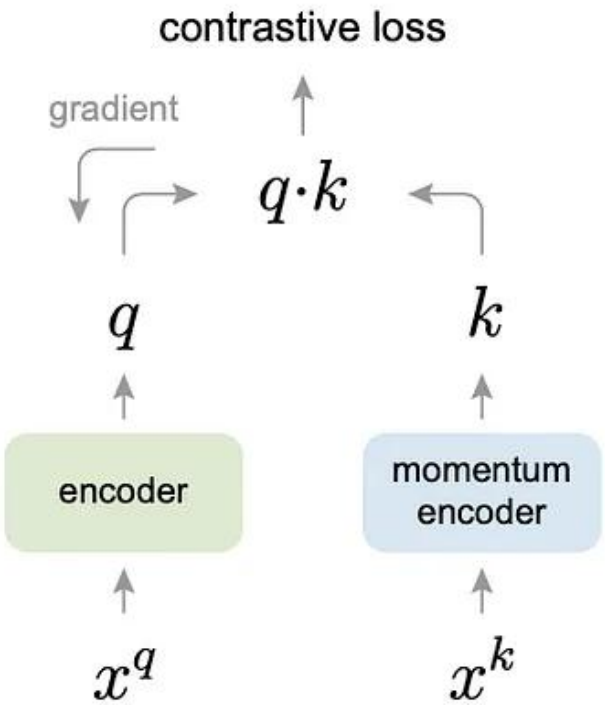
$$\theta_{\mathrm{k}} \leftarrow m\theta_{\mathrm{k}} + (1-m)\theta_{\mathrm{q}}.$$



(a) end-to-end



(b) memory bank



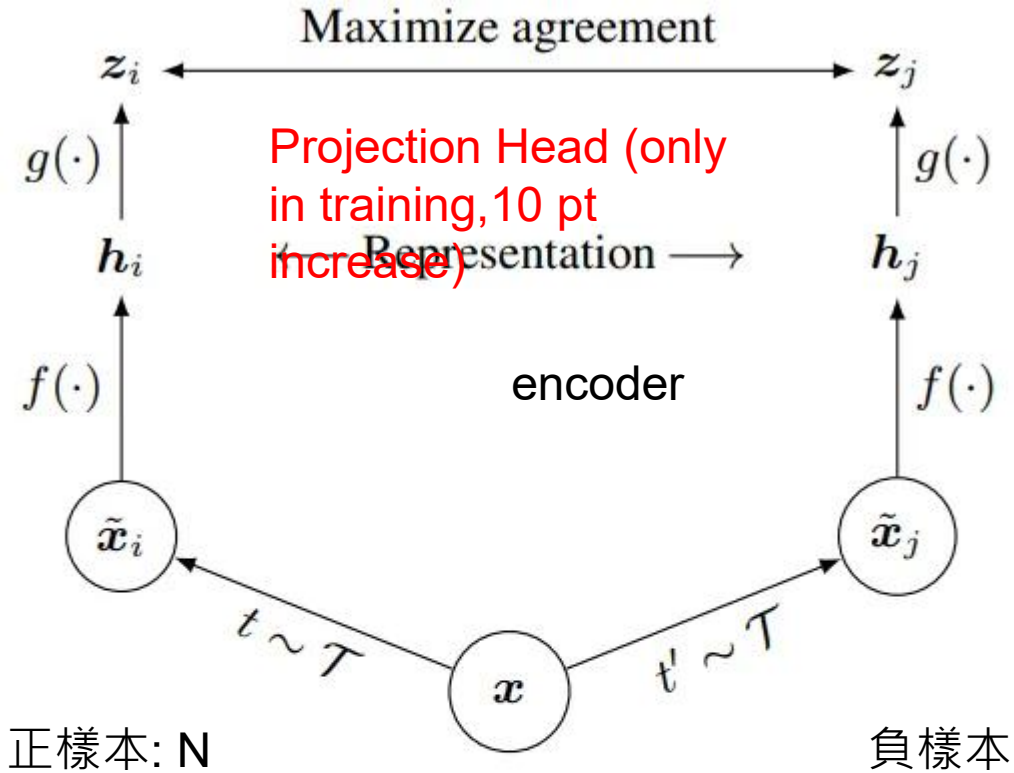
(c) MoCo

$$\theta_{\mathrm{k}} \leftarrow m\theta_{\mathrm{k}} + (1-m)\theta_{\mathrm{q}}.$$

SimCLR v1

- 訓練過程：
  - 從一個 mini-batch 中選取 N 張圖片。
  - 對每張圖片進行兩次不同的數據增強，生成 2N 個增強後的圖片。
  - 將所有 2N 個增強後的圖片輸入編碼器，生成 2N 個特徵向量。
  - 將所有 2N 個特徵向量輸入 Projection Head，生成 2N 個低維特徵向量。
  - 計算對比學習的損失函數 (例如 InfoNCE loss)，使得來自同一張圖片的增強版本的特徵向量彼此靠近，而自不同圖片的增強版本的特徵向量彼遠離。
  - 更新encoder和 Projection Head 的參數。

正負樣本都來自於同一個 mini-batch
需要使用大的 batch size 才能取得好的效果。



正樣本: N

負樣本: 2(N-1

Figure 2. A simple framework for contrastive learning of visual representations. Two separate data augmentation operators are sampled from the same family of augmentations ($t \sim \mathcal{T}$ and $t' \sim \mathcal{T}$) and applied to each data example to obtain two correlated views. A base encoder network $f(\cdot)$ and a projection head $g(\cdot)$ are trained to maximize agreement using a contrastive loss. After

SimCLR vs MOCO

- 架構差異
  - SimCLR： SimCLR 採用端到端的學習方式，只使用一個編碼器來處理所有圖片。 它不需要額外的數據結構（例如 memory bank 或隊列）來存儲負樣本，它的正負樣本都來自於同一個 mini-batch。
  - MoCo： MoCo 使用兩個獨立的編碼器：查詢編碼器和鍵編碼器。 查詢編碼器用於生成輸入圖片的特徵向量，而鍵編碼器用於生成負樣本特徵向量。 為了存儲大量的負樣本特徵向量，MoCo 使用了隊列這種數據結構。
- 訓練策略差異
  - 數據增強： SimCLR 和 MoCo 都強調了數據增強在對比學習中的重要性，但 SimCLR 使用了更強大的數據增強技術，例如組合多種數據增強方法。
  - Projection Head： SimCLR 和 MoCo v2 都引入了 Projection Head，它是一個多層感知器 (MLP)，用於將特徵向量映射到一個低維空間。 SimCLR v2 發現使用更深的 Projection Head 可以進一步提升模型的性能。
  - 動量編碼器： MoCo 使用動量編碼器來生成負樣本特徵向量，這確保了負樣本特徵向量的一致性和穩定性。SimCLR v2 也採用了動量編碼器，但發現它的提升幅度不如 MoCo 顯著。
  - Batch Size： SimCLR 需要使用很大的 batch size 才能取得好的效果，這對硬體資源的要求較高。 MoCo 可以使用較小的 batch size 進行訓練，因此對硬體資源的要求較低。
- 性能比較
  - ImageNet 線性評估： 在 ImageNet 線性評估任務上，SimCLR 和 MoCo 都取得了很好的效果，但 MoCo v2 的性能略優於 SimCLR v2。
  - 下游任務遷移： 在下游任務遷移方面，MoCo v2 的表現優於 SimCLR。

MOCO v2

- SimCLR v1 => MOCO v2
  - 大 Batchsize、 多 Epochs、更多更強的數據增強、增加一個 projection 層

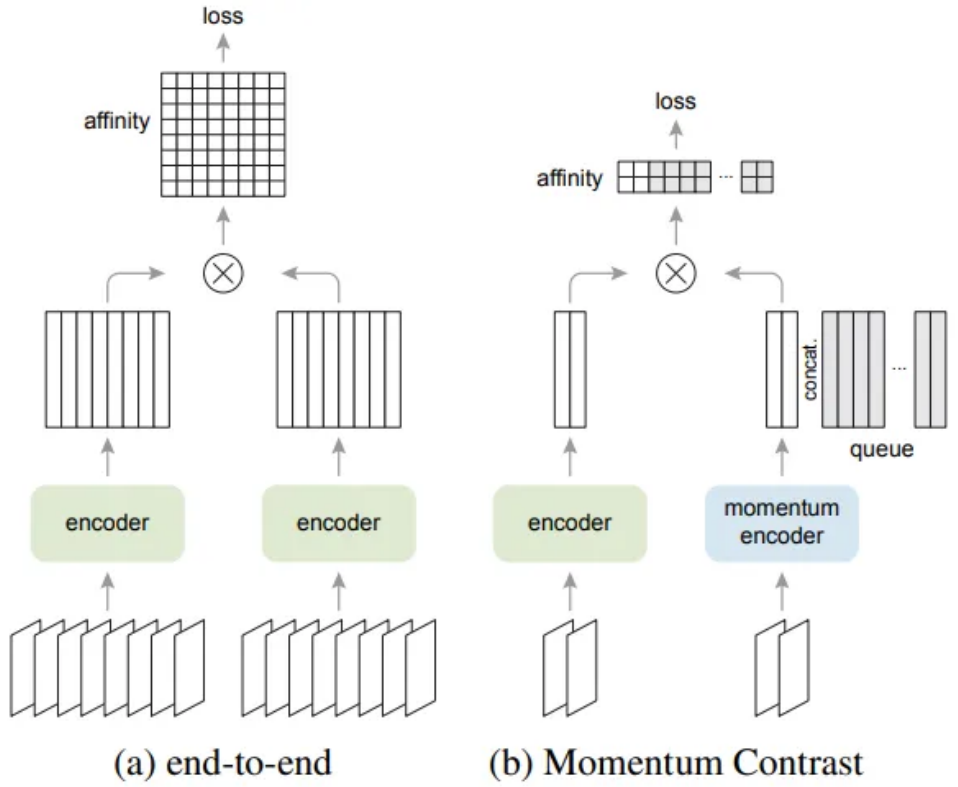


Figure 1. A **batching** perspective of two optimization mechanisms

case	unsup. pre-train MLP	aug+	cos	epochs	batch	ImageNet acc.
MoCo v1 [6]				200	256	60.6
SimCLR [2]	✓	✓	✓	200	256	61.9
SimCLR [2]	✓	✓	✓	200	8192	66.6
MoCo v2	✓	✓	✓	200	256	**67.5**
*results of **longer** unsupervised training follow:*						
SimCLR [2]	✓	✓	✓	1000	4096	69.3
MoCo v2	✓	✓	✓	800	256	**71.1**

Table 2. **MoCo *vs.* SimCLR**: ImageNet linear classifier accuracy (**ResNet-50, 1-crop 224×224**), trained on features from unsupervised pre-training. "aug+" in SimCLR includes blur and stronger color distortion. SimCLR ablations are from Fig. 9 in [2] (we thank the authors for providing the numerical results).

case	unsup. pre-train MLP	aug+	cos	epochs	ImageNet acc.	VOC detection AP_{50}	AP	AP_{75}
supervised					76.5	81.3	53.5	58.8
MoCo v1				200	60.6	81.5	55.9	62.6
(a)	✓			200	66.2	82.0	56.4	62.6
(b)		✓		200	63.4	82.2	56.8	63.2
(c)	✓	✓		200	67.3	**82.5**	57.2	63.9
(d)	✓	✓	✓	200	67.5	82.4	57.0	63.6
(e)	✓	✓	✓	**800**	**71.1**	**82.5**	**57.4**	**64.0**

Table 1. **Ablation of MoCo baselines**, evaluated by ResNet-50 for (i) ImageNet linear classification, and (ii) fine-tuning VOC object detection (mean of 5 trials). "**MLP**": with an MLP head; "**aug+**": with extra blur augmentation; "**cos**": cosine learning rate schedule.

mechanism	batch	memory / GPU	time / 200-ep.
MoCo	256	**5.0G**	**53 hrs**
end-to-end	256	7.4G	65 hrs
end-to-end	4096	93.0G†	n/a

Table 3. **Memory and time cost** in 8 V100 16G GPUs, implemented in PyTorch. †: based on our estimation.

SimCLR v2

- 換了更大的模型，取得更好的 performance
  - 讓微調的模型作為 Teacher 模型，可以使用 Student 小網絡來壓縮大網絡的訊息，將訓練好的大的網絡的知識，灌入小的網絡中

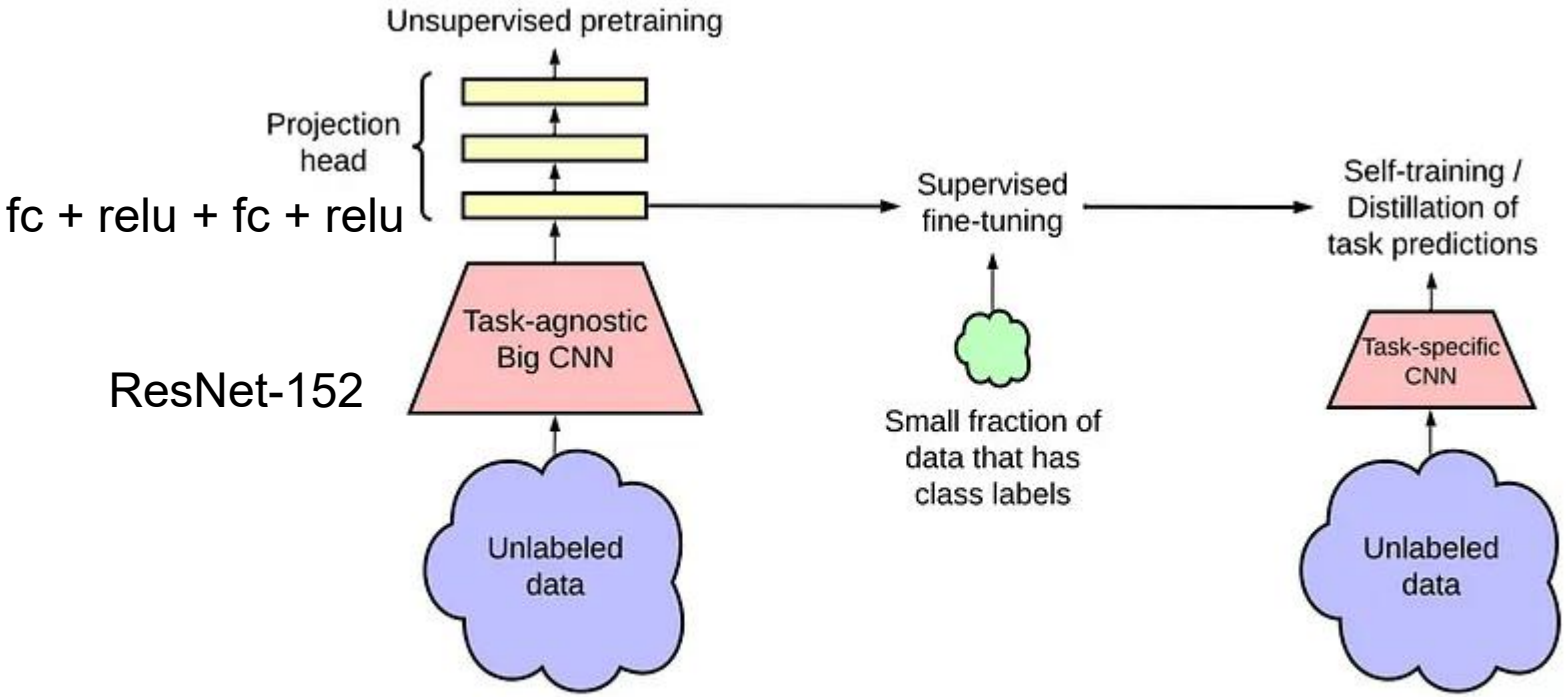


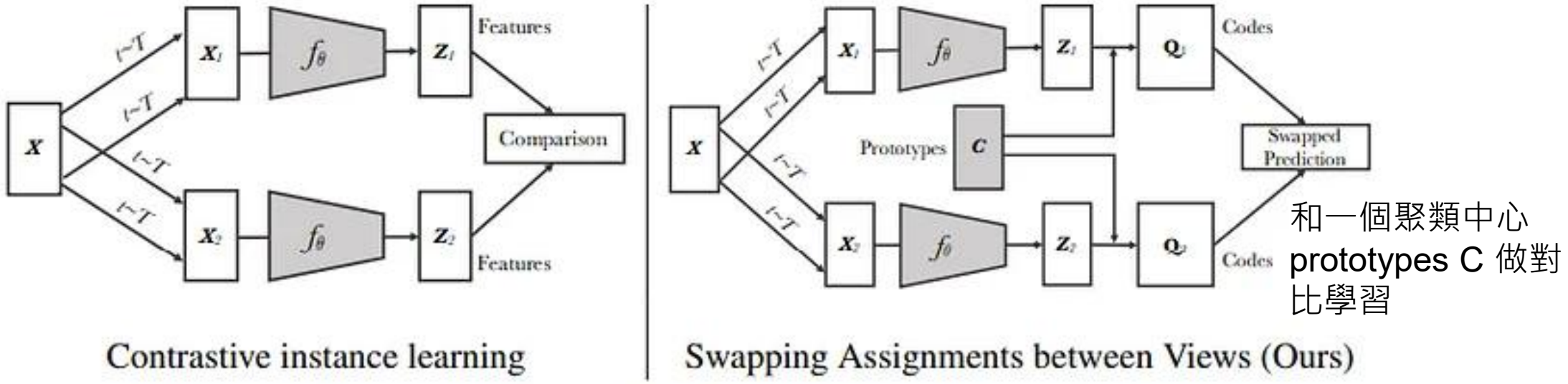
Figure 3: The proposed semi-supervised learning framework leverages unlabeled data in two ways: (1) task-agnostic use in unsupervised pretraining, and (2) task-specific use in self-training / distillation.

simCLR v2 嘗試使用 MOCO 的動量編碼器，發現效果提升的 1 %，提升的沒有很顯著，可能原因為 SimCLR v2 已經有很大的 Batchsize 了，所以不需要動量編碼器以及隊列的負樣本了

SwAV (Swap Assignment Views)

- 結合clustering和對比學習，學習具有視角不變性的強大特徵表示
  - SwAV 不直接使用特徵進行對比，利用聚類中心 (e.g.3000)來代替負樣本，語義明確有利學習，從而減少計算成本並提升訓練效率
  - 使用 Swapped Prediction 機制，利用一個視角的特徵向量和聚類中心來預測另一個視角的聚類分配向量

clustering



和一個聚類中心 prototypes C 做對比學習

Figure 1: **Contrastive instance learning (left) *vs.* SwAV (right).** In contrastive learning methods

- Multi-crop
  - 以往都在一張 256×256 的圖片上用兩個 224×224 的 crop 提取兩個正樣本，但是因為 crop 過大了，所選取的 crop 都是基於全局特徵的，很多局部特徵才是非常有價值的，
  - SwAV 選擇了兩個 160×160 的 crop 去提取全局特徵，選擇四個 96×96的 crop 提取局部特徵，在計算量沒有劇增的情況下，獲取更多的正樣本。

Method	Top-1 2x224	Top-1 2x160+4x96	Δ
Supervised	76.5	76.0	−0.5
Contrastive-instance approaches			
SimCLR	68.2	70.6	+2.4
Clustering-based approaches			
SeLa-v2	67.2	71.8	+4.6
DeepCluster-v2	70.2	74.3	+4.1
SwAV	70.1	74.1	+4.0

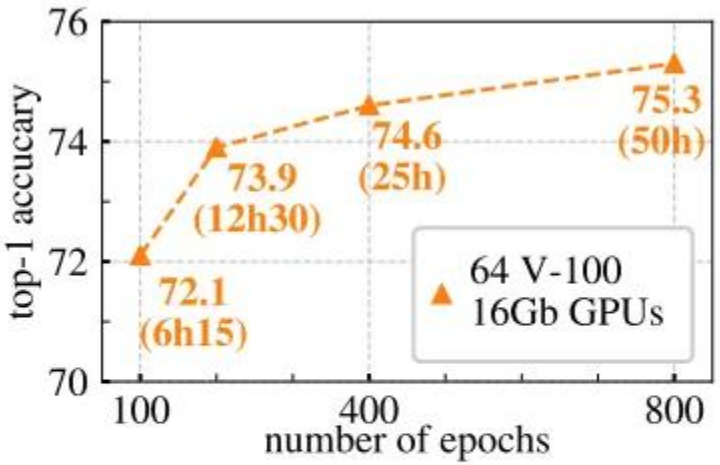


Figure 3: Top-1 accuracy on ImageNet with a linear classifier trained on top of frozen features from a ResNet-50. **(left) Comparison between clustering-based and contrastive instance methods and impact of multi-crop.** Self-supervised methods are trained for 400 epochs and supervised models for 200 epochs. **(right) Performance as a function of epochs.** We compare SwAV models trained with different number of epochs and report their running time based on our implementation.

Method	Arch.	Param.	Top1
Supervised	R50	24	76.5
Colorization [65]	R50	24	39.6
Jigsaw [46]	R50	24	45.7
NPID [58]	R50	24	54.0
BigBiGAN [15]	R50	24	56.6
LA [68]	R50	24	58.8
NPID++ [44]	R50	24	59.0
MoCo [24]	R50	24	60.6
SeLa [2]	R50	24	61.5
PIRL [44]	R50	24	63.6
CPC v2 [28]	R50	24	63.8
PCL [37]	R50	24	65.9
SimCLR [10]	R50	24	70.0
MoCov2 [11]	R50	24	71.1
SwAV	R50	24	**75.3**

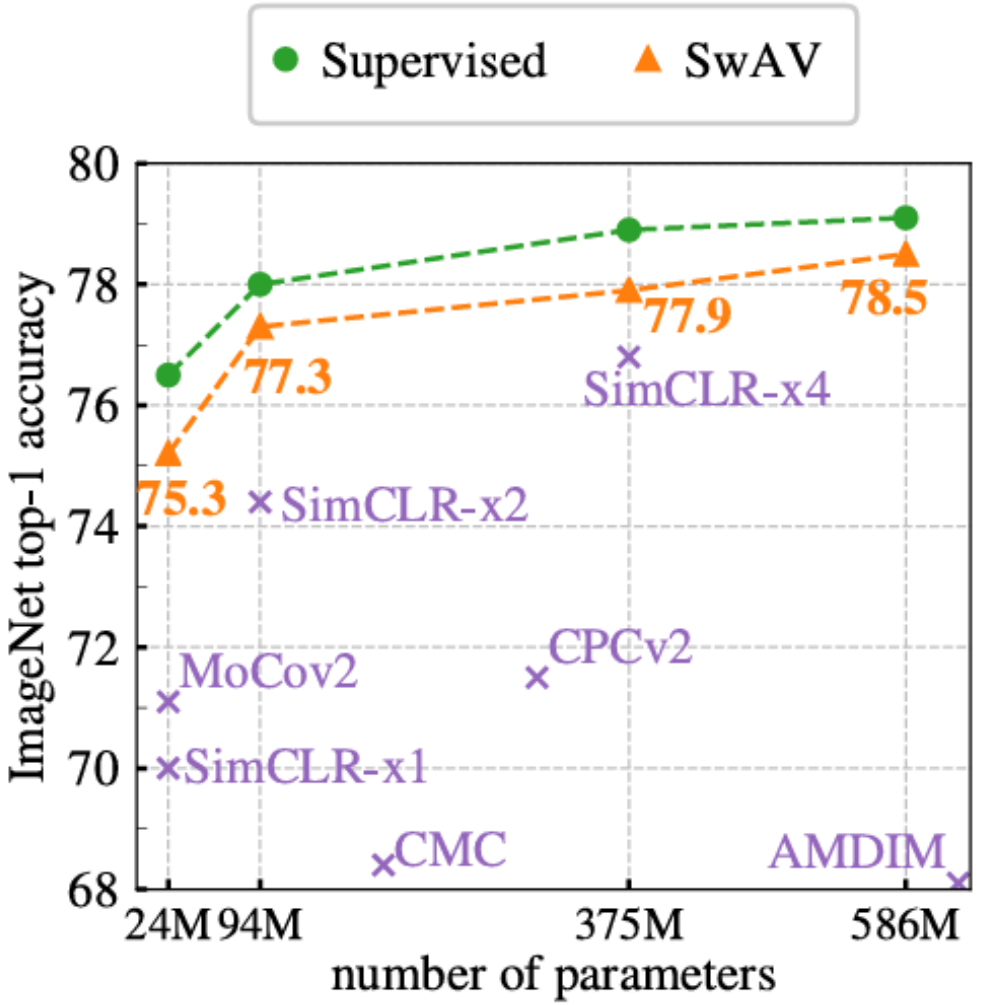
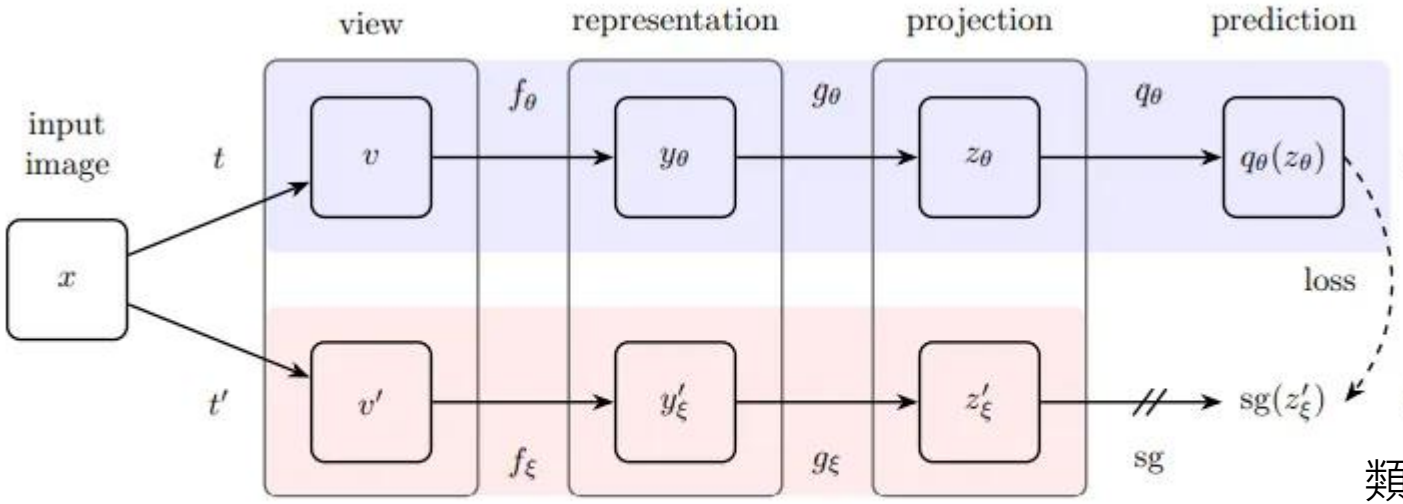


Figure 2: **Linear classification on ImageNet.** Top-1 accuracy for linear models trained on frozen features from different self-supervised methods. **(left)** Performance with a standard ResNet-50. **(right)** Performance as we multiply the width of a ResNet-50 by a factor $\times 2$, $\times 4$, and $\times 5$.

- 負樣本的必要性
  - 假設模型的輸入只有正樣本，那麼模型需要讓正樣本之間的距離盡可能地縮小，但對模型來說正樣本之間是相似的，所有正樣本之間的距離無限接近。這會導致模型學習一個捷徑解，即無論輸入是什麼，模型都返回相同的輸出
  - 此時添加負樣本可以對模型形成一個約束，讓正樣本之間的距離接近，讓負樣本之間的距離拉遠，所以添加負樣本是非常關鍵的，讓模型不會坍塌。
- SwAV
  - 已經稱得上不用負樣本了，它是和一個聚類中心 prototypes C 做對比學習
- 可以不要負樣本嗎?

BYOL: Bootstrap Your Own Latent

- 架構: 提出正樣本自己和自己學習，完全沒有負樣本或者聚類中心
  - 通過**預測**機制來學習特徵表示，即利用一個視角的特徵來預測另一個視角的特徵
  - BYOL 採用孿生網絡架構，包含兩個具有相同網絡結構但參數不同的編碼器
  - fθ 會隨著梯度更新而更新，而 fξ 則使用動量更新方式來更新
  - 整體思路是將上半部的網絡拿來預測下半部網絡的內容，讓二者盡量相似



類似query encoder

將對比學習問題轉化為預測問題
MSE loss

類似key encoder,動量編碼器更新

Bootstrap Your Own Latent A New Approach to Self-Supervised Learning

Figure 2. BYOL's architecture. BYOL minimizes a similarity loss between $q_\theta(z_\theta)$ and $\mathrm{sg}(z'_\xi)$, where θ are the trained

Simsiam

- 也不需要用負樣本，更是直接把動量編碼器給拿掉了
  - 將一張圖片 X 經過兩個數據增強變為 X1 和 X2，接著經過孿生的 Encoder f，將左半部網路得到的 Embedding 向量放入 predictor h ，拿來預測右半部網路的內容

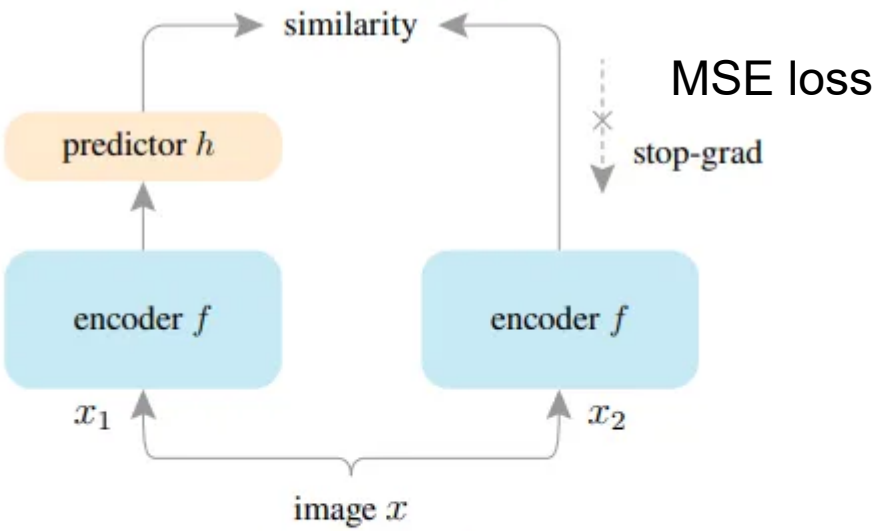
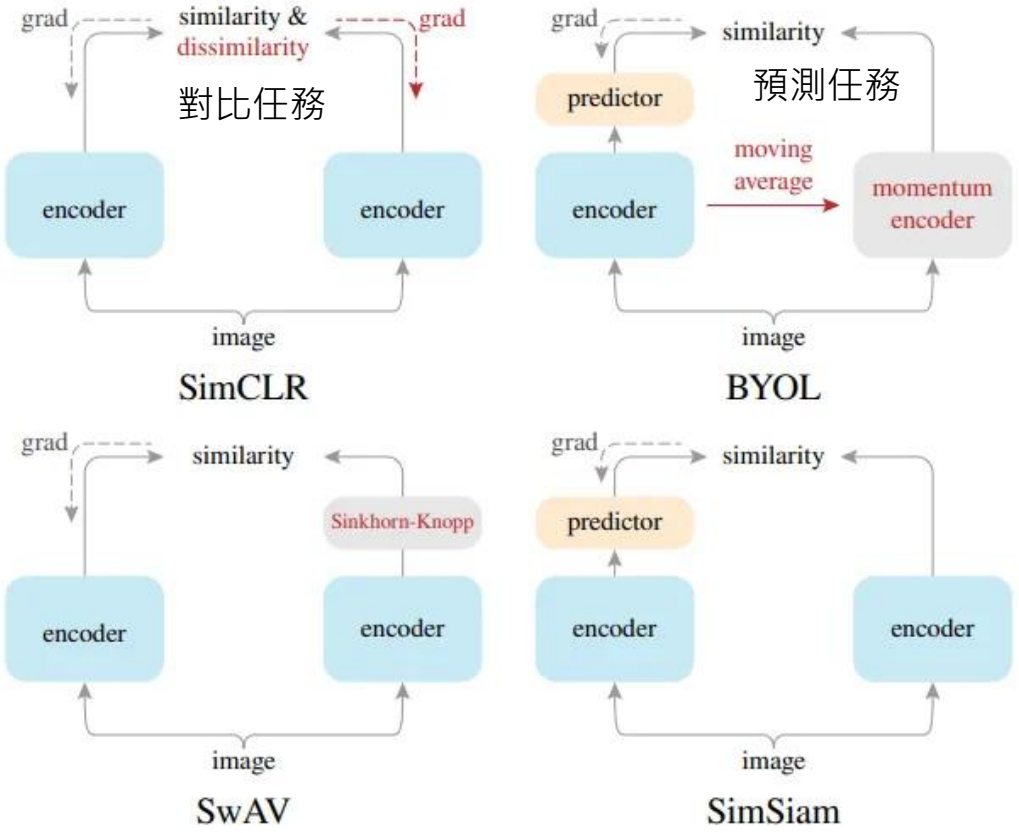


Figure 1. **SimSiam architecture**. Two augmented views of one image are processed by the same encoder network f (a backbone plus a projection MLP). Then a prediction MLP h is applied on one side, and a stop-gradient operation is applied on the other side. The model maximizes the similarity between both sides. It uses neither



method	batch size	negative pairs	momentum encoder	100 ep	200 ep	400 ep	800 ep
SimCLR (repro.+)	4096	✓		66.5	68.3	69.8	70.4
MoCo v2 (repro.+)	**256**	✓	✓	67.4	69.9	71.0	72.2
BYOL (repro.)	4096		✓	66.5	**70.6**	**73.2**	**74.3**
SwAV (repro.+)	4096			66.5	69.1	70.7	71.8
SimSiam	**256**			**68.1**	70.0	70.8	71.3

Table 4. **Comparisons on ImageNet linear classification**. All are based on **ResNet-50** pre-trained with **two 224×224 views**. Evaluation is on a single crop. All competitors are from our reproduction, and "+" denotes *improved* reproduction *vs*. original papers (see supplement).

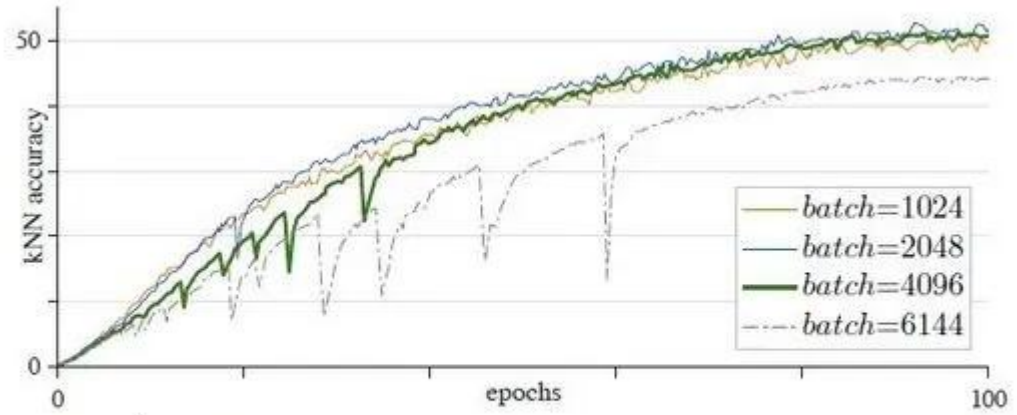
	VOC 07 detection			VOC 07+12 detection			COCO detection			COCO instance seg.		
pre-train	AP_{50}	AP	AP_{75}	AP_{50}	AP	AP_{75}	AP_{50}	AP	AP_{75}	AP^{mask}_{50}	AP^{mask}	AP^{mask}_{75}
scratch	35.9	16.8	13.0	60.2	33.8	33.1	44.0	26.4	27.8	46.9	29.3	30.8
ImageNet supervised	74.4	42.4	42.7	81.3	53.5	58.8	58.2	38.2	41.2	54.7	33.3	35.2
SimCLR (repro.+)	75.9	46.8	50.1	81.8	55.5	61.4	57.7	37.9	40.9	54.6	33.3	35.3
MoCo v2 (repro.+)	**77.1**	**48.5**	**52.5**	**82.3**	**57.0**	**63.3**	**58.8**	**39.2**	**42.5**	**55.5**	**34.3**	**36.6**
BYOL (repro.)	**77.1**	47.0	49.9	81.4	55.3	61.1	57.8	37.9	40.9	54.3	33.2	35.0
SwAV (repro.+)	75.5	46.5	49.6	81.5	55.4	61.4	57.6	37.6	40.3	54.2	33.1	35.1
SimSiam, base	75.5	47.0	50.2	**82.0**	56.4	62.8	57.5	37.9	40.9	54.2	33.2	35.2
SimSiam, optimal	**77.3**	**48.5**	**52.5**	**82.4**	**57.0**	**63.7**	**59.3**	**39.2**	**42.1**	**56.0**	**34.4**	**36.7**

Table 5. **Transfer Learning**. All unsupervised methods are based on 200-epoch pre-training in ImageNet. *VOC 07 detection*: Faster R-CNN [32] fine-tuned in VOC 2007 trainval, evaluated in VOC 2007 test; *VOC 07+12 detection*: Faster R-CNN fine-tuned in VOC 2007 trainval + 2012 train, evaluated in VOC 2007 test; *COCO detection* and *COCO instance segmentation*: Mask R-CNN [18] (1× schedule) fine-tuned in COCO 2017 train, evaluated in COCO 2017 val. All Faster/Mask R-CNN models are with the C4-backbone [13]. All VOC results are the average over 5 trials. **Bold entries** are within 0.5 below the best.

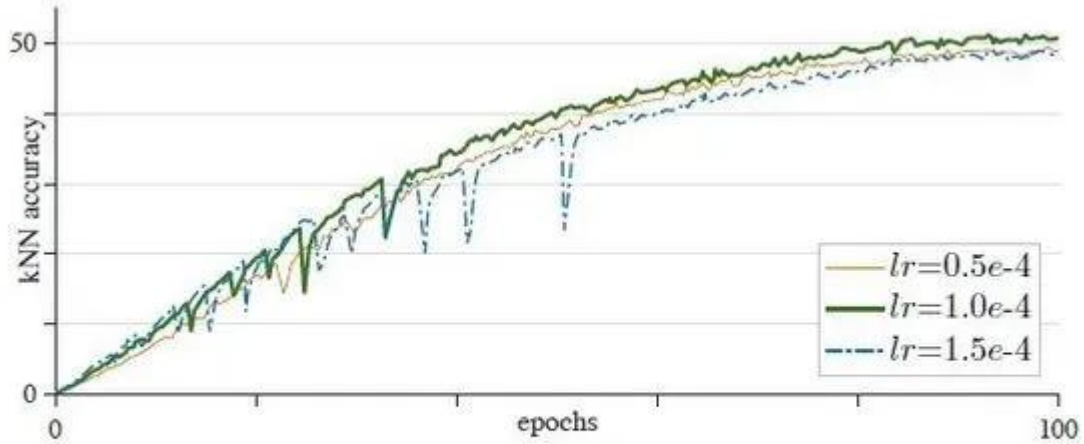
- **Stop gradient 操作的作用:**
  - 在 BYOL 和 SimSiam 中，模型會使用兩個編碼器（例如 ResNet-50）來分別處理同一張圖片的不同視角（view）。其中一個編碼器被稱為 online network 或 student network，它的參數會隨著梯度更新而更新；另一個編碼器則被稱為 target network 或 teacher network，它的參數則使用動量更新的方式進行更新，並且在**計算損失函數時不進行梯度回傳**，這就是 stop gradient 操作的應用之處。
- **Stop gradient 操作的意義:**
  1. **防止模型坍塌**: 如果沒有 stop gradient 操作，模型很容易學到一個捷徑解，也就是對所有輸入都輸出相同的特徵表示。 這是因為在沒有負樣本約束的情況下，模型只需要讓所有正樣本的特徵表示盡可能相似就可以最小化損失函數。 但是，這種情況下模型並沒有學到任何有用的資訊，因為它無法區分不同的輸入。
  2. **實現自蒸餾**: **Stop gradient 操作可以將模型的訓練過程人為地分成兩部分**，一部分是 student network 的學習過程，另一部分是 teacher network 的更新過程。 這種**交替更新的方式類似於 EM 演算法，可以說明模型更好地學習到資料中的結構資訊**。
  3. **簡化模型訓練**: Stop gradient 操作可以簡化模型的訓練過程，因為它不需要像 SimCLR 那樣使用大量的負樣本，也不需要像 MoCo 那樣使用佇列和動量編碼器。
- **Stop gradient 操作的爭議:**
  - 一些研究者認為，BYOL 中 stop gradient 操作的成功可能與 batch normalization (BN) 有關。他們認為，BN 在計算批次統計量時會洩露其他樣本的資訊，這相當於提供了一種隱式的負樣本。 然而，BYOL 的作者通過進一步的實驗反駁了這個觀點，他們證明了即使在沒有 BN 的情況下，BYOL 仍然可以正常訓練，並取得良好的效果。

MOCO v3 (MOCO v2+BYOL => ViT)

- 當 MOCO v3 的 Encoder = Vision Transformer 時，
  - 訓練會變得不穩定， 造成的結果並不是訓練無法收斂，而是 Performance 會輕微的下降



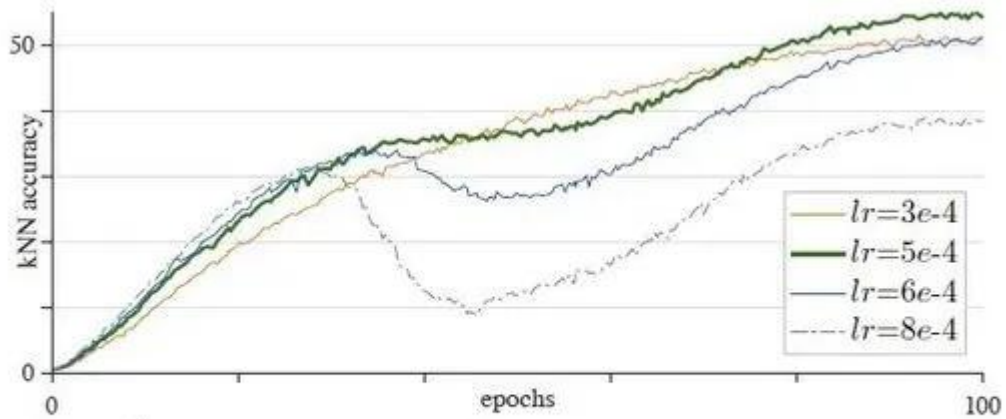
batch	1024	2048	4096	6144
linear acc.	71.5	72.6	72.2	69.7



lr, $\times 1e$-4	0.5	1.0	1.5
linear acc.	70.4	72.2	71.7

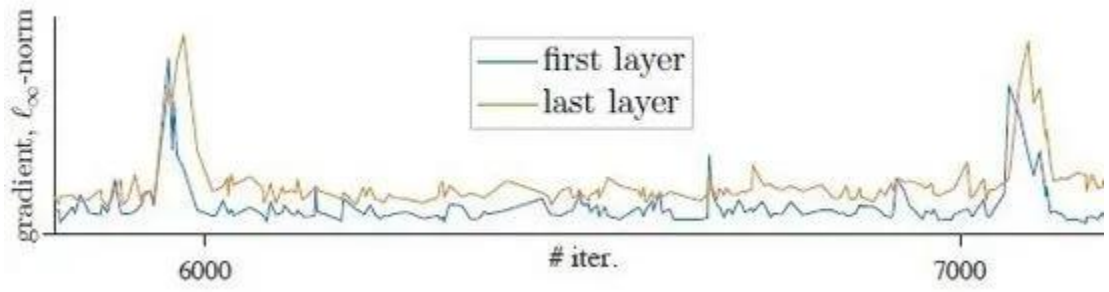
當 Batch Size=4096 時，曲線出現了 Dip 現象 。當 Batch Size=6144 時，曲線的 Dip 現象更明顯了。這種不穩定的現象導致精度下降。

當 Learning Rate = 0.5e-4 時，曲線出現了 Dip 現象 。當 Learning Rate = 1.5e-4 時，曲線的 Dip 現象更明顯了

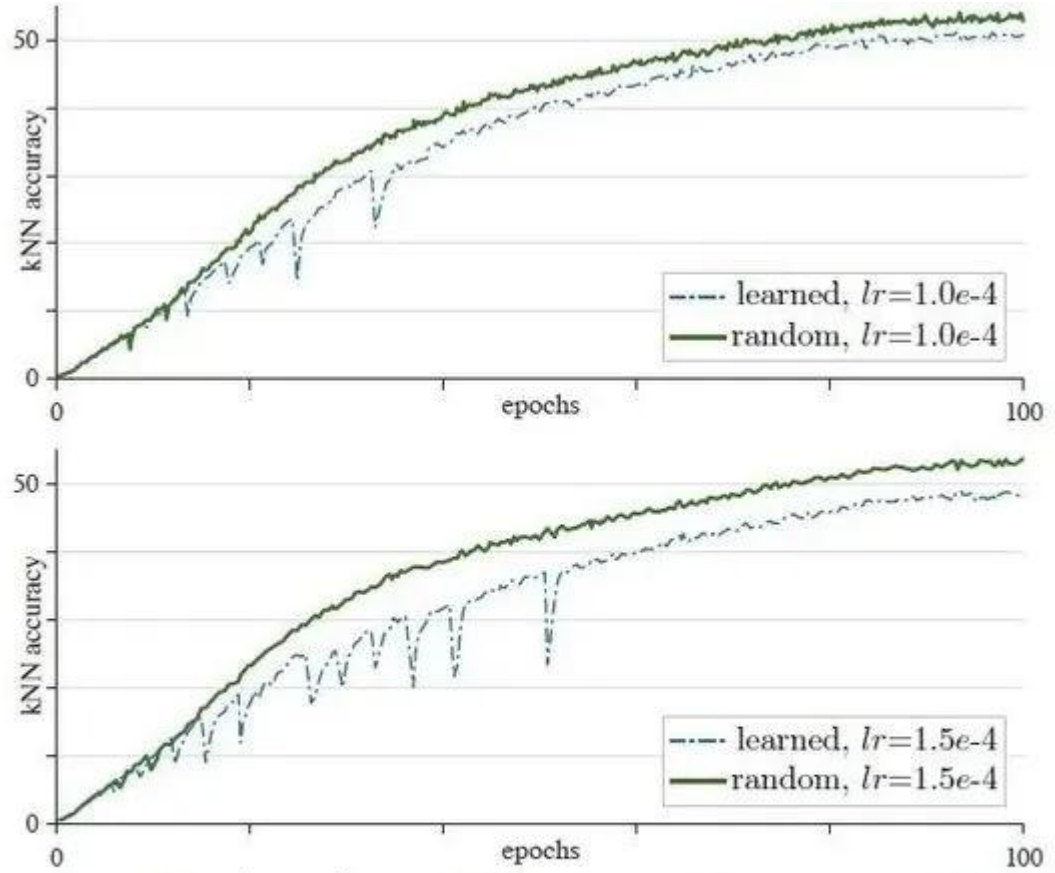


lr, $\times 1e\text{-}4$	3.0	5.0	6.0	8.0
linear acc.	71.6	72.5	70.9	66.5

當 Optimizer = LARS 時當 Learning Rate = 5e-4 時，LARS 的 Performance 可以超過 AdamW，而且曲線是平滑的，沒有 Dip 現象。



導致訓練出現不穩定的跟第 1 層梯度暴漲有關



lr, $\times 10^{-4}$	0.5	1.0	1.5
learned patch proj.	70.4	72.2	71.7
random patch proj.	**70.8**	**72.8**	**73.4**

凍結 Patch Embedding 層的參數，都能夠提升訓練的穩定性

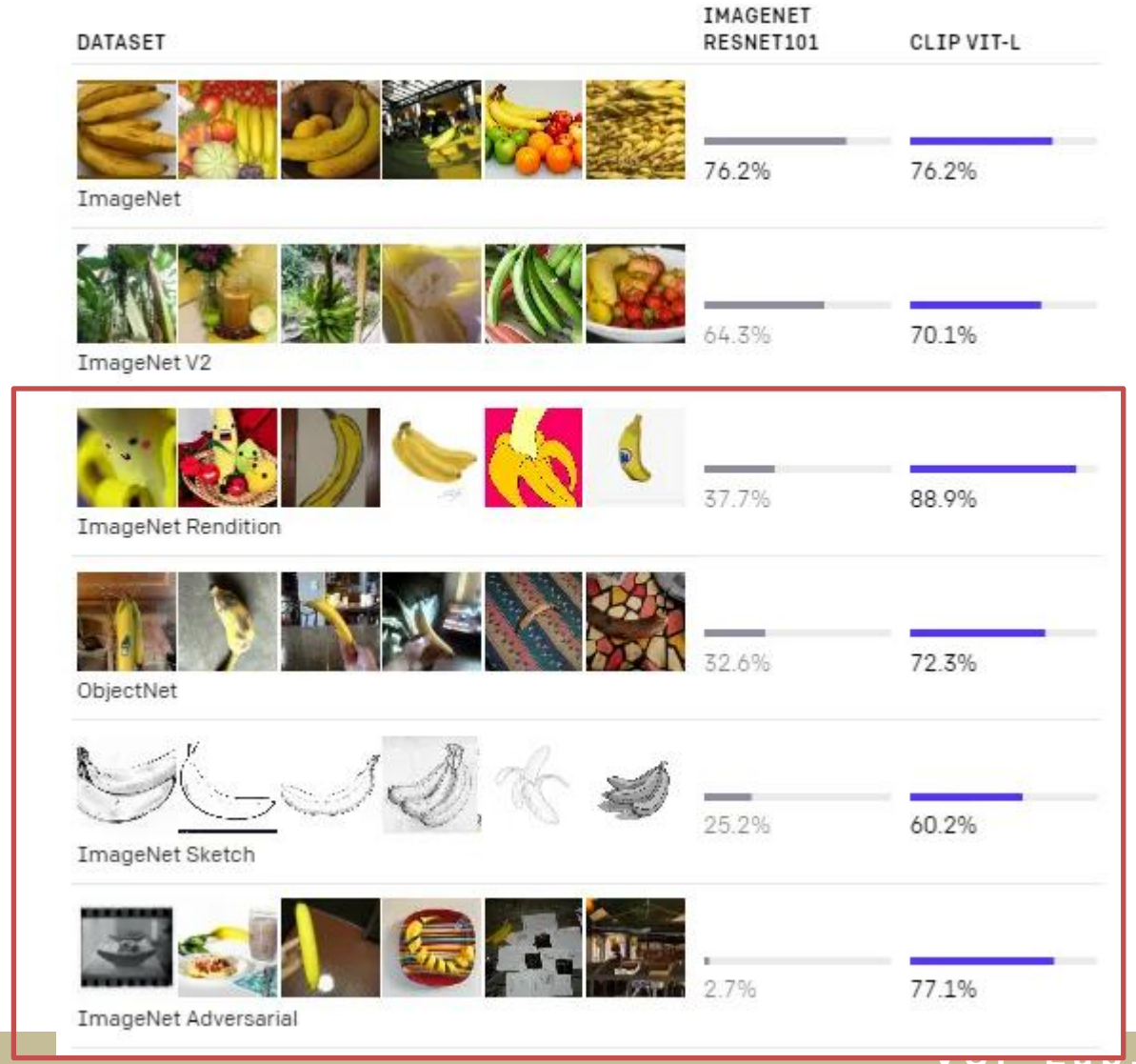
DINO (Self-distillation with no labels)

- 使用 8x8 patches 的 Vi-T Self-Supervised Learning 的 Attention Map。
  - Self-Supervised Learning，還是可以把注意力都學在這些物體上，模型非常準確的抓住每個物體的輪廓，這個效果甚至可以直接對這個物體作語義分割。

PROMPT BASED COMPUTER VISION

CLIP: Learning Transferable Visual Models From Natural Language Supervision

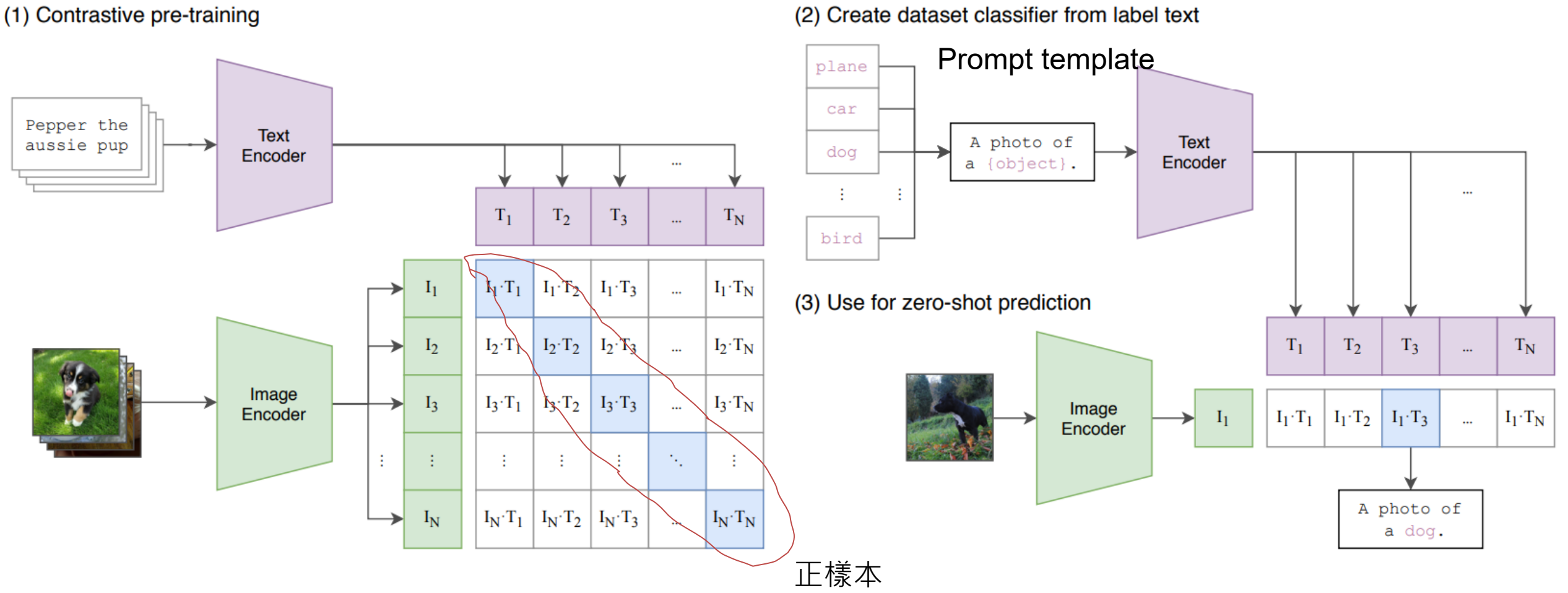
- 傳統作法問題
  - 訓練和預測通常使用一組固定的預定義物件類別，例如ImageNet。
  - 難擴展，不夠robust
- How
  - 預訓練任務是在一個包含4億（圖像，文本）對的數據集上，從頭開始預測應該學習哪個標題以獲得圖像特徵表示。
  - 預訓練後，自然語言被用來參照學習到的視覺概念（或描述新概念），實現模型對下游任務的zero-shot transfer



- (text, image) pair, WebImageText (WIT) Dataset
  - 從自然語言中學習相對於標準的眾包標註圖像分類方式更容易擴展，因為它不需要標註。
  - 相較於大多數的無監督或自監督學習方法，它還有一個重要優勢，那就是它不"僅僅"學習一種表示，而且還將該影像特徵表示與語言相連接，這使得靈活的zero-shot轉移成為可能。
- 效果
  - 在 ImageNet 資料集上，CLIP 達到了 76.2% 的準確率。
  - 與使用 ImageNet 訓練資料進行監督式訓練的 ResNet-50 模型的準確率相當，但 CLIP 在此過程中完全沒有使用任何 ImageNet 訓練集中的圖片
  - 但它的準確率仍然落後於目前 ImageNet 上表現最佳的模型，例如 Noisy Student 和 Vision Transformer。這些模型的準確率已經達到 88% 甚至 90% 以上，而 CLIP 僅為 76.2%。作者估計，如果要縮小這個差距，需要將 CLIP 的訓練規模擴大 1000 倍，這對目前的硬體條件來說是難以實現的

好處

- 擺脫固定類別標籤的限制
  - CLIP 模型則利用自然語言作為監督信號，學習圖像和文本之間的語義關聯，從而擺脫了固定類別標籤的限制，可以靈活地進行 zero-shot 遷移學習，識別任何可以用文本描述的物體類別。
- 強大的 Zero-Shot 遷移能力
  - 在 ImageNet 上達到了 76.2% 的準確率，這與使用 ImageNet 訓練資料進行監督式訓練的 ResNet-50 模型的準確率相當，但 CLIP 完全沒有使用任何 ImageNet 訓練集中的圖片。
- 更接近人類的感知方式:
  - CLIP 模型通過學習圖像和文本之間的語義關聯，使其在某些任務上的表現更接近人類的感知方式。例如，對於 CLIP 模型來說，難分類的類別對人類來說也很難。
- 更高的數據利用效率
- 更强大的泛化能力

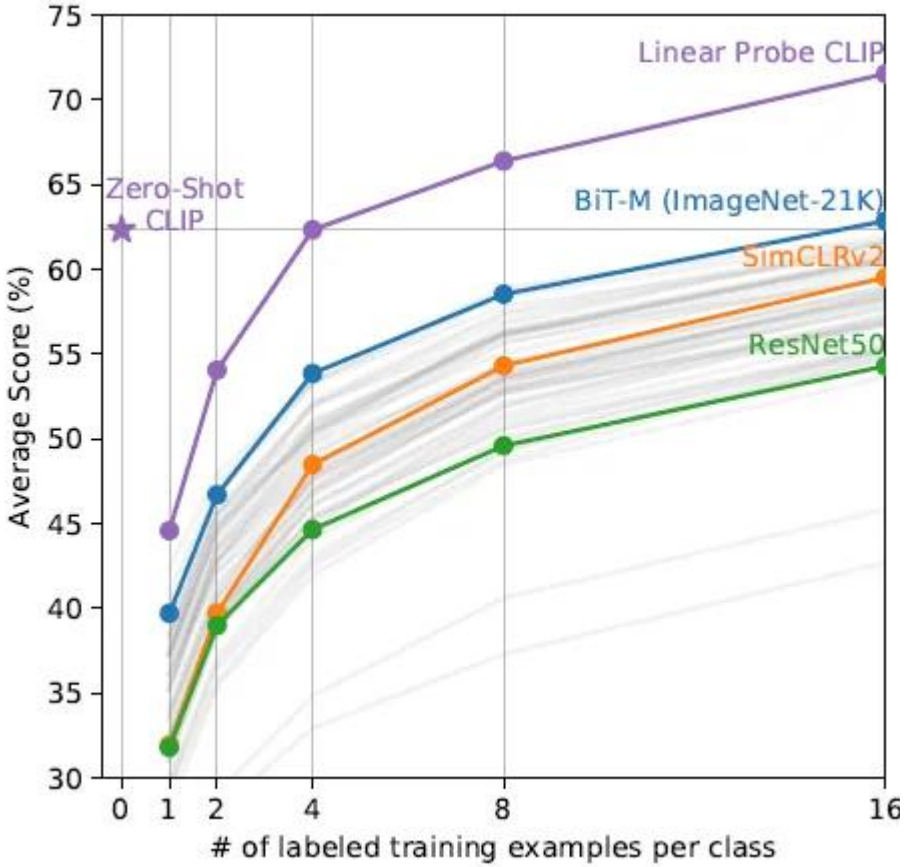


正樣本

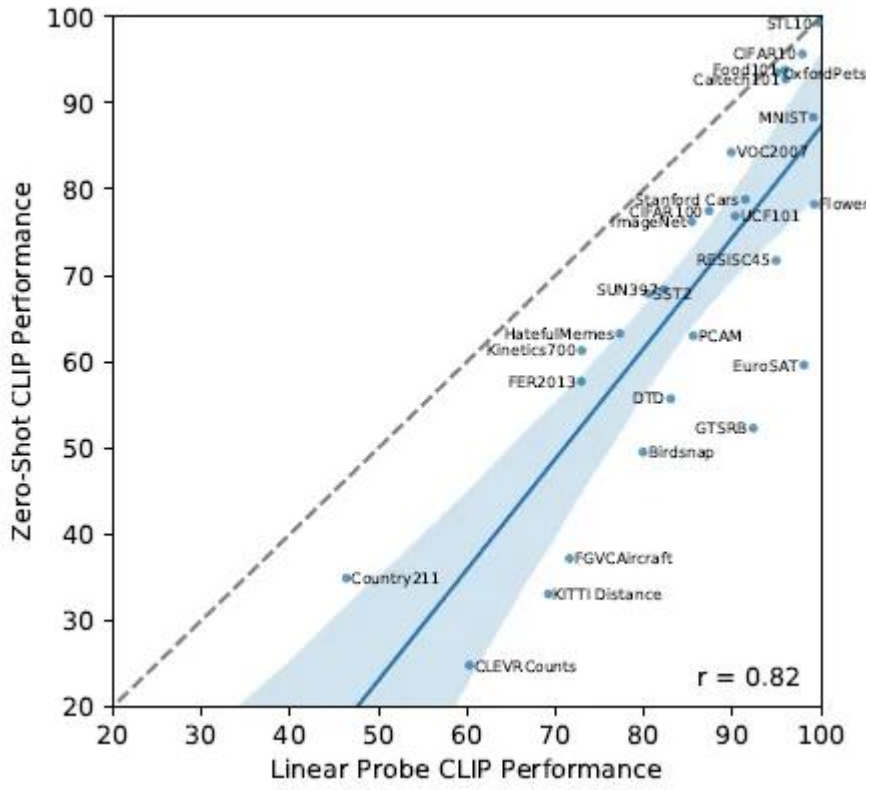
用NLP來做監督訊號

CLIP同時訓練一個圖像編碼器和一個文本編碼器，以預測一批（image, text）訓練樣本的正確配對。在測試時，學習後的文本編碼器通過嵌入目標數據集類別的名稱或描述，合成了一個zero-shot線性分類器

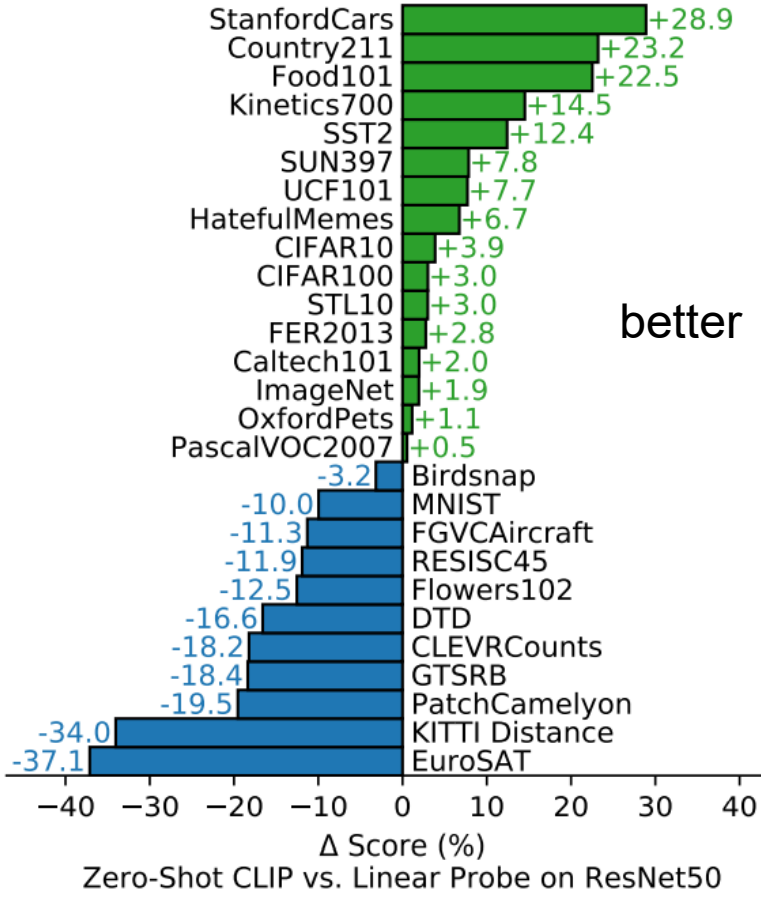
- Training Objective
  - The objective of CLIP is to align text embeddings and image embeddings for correct pairs. IT achieved this by maximizing the cosine similarity between the correct pair of image and text embeddings and minimising the cosine similarity between incorrect image and text embeddings.
  - The image and text encoder are trained from scratch without using any pre-weights.
  - The embeddings generated by the image and text encoder are linear projected to match their dimension
  - For a dataset of N (image, text Paris) there are N*N possible (image, text) pairs. Out of these N*N pairs N pairs are correct and the rest N*N-N pairs are incorrect.
  - The model is trained to maximize the cosine similarity of the image and text embeddings of the N correct pairs and minimize the cosine similarity of N*N-N pairs.
  - Cross entropy loss over similarity score is used as a loss function.
- During inference time
  - the embedding of image and text description of all possible pairs is obtained through the trained image and text encoder and a similarity score is calculated which indicates the relevance of text description concerning image



Zero-shot CLIP 優於少數樣本的線性探測。



Zero-shot性能與線性探測性能相關，但仍然大多數情況下並非最優。



Zero-shot CLIP is competitive with a fully supervised baseline

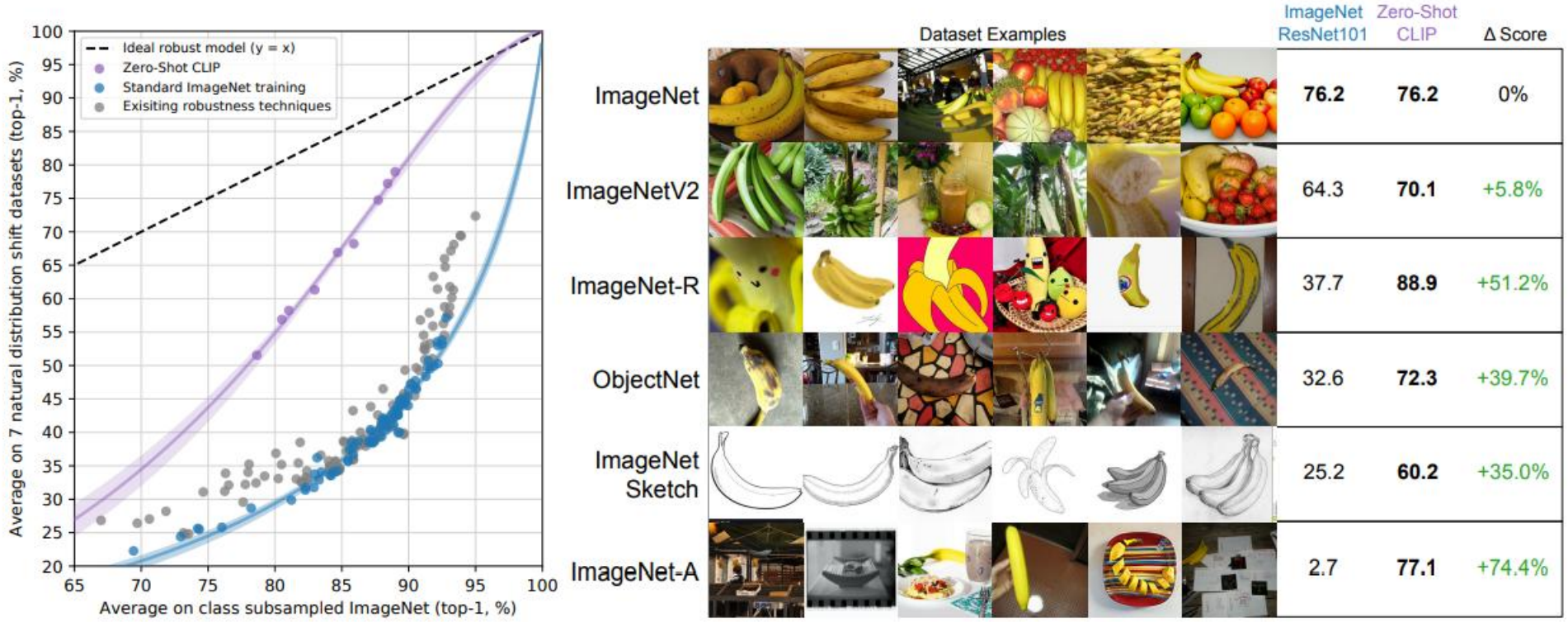


Figure 13. **Zero-shot CLIP is much more robust to distribution shift than standard ImageNet models.** (Left) An ideal robust model (dashed line) performs equally well on the ImageNet distribution and on other natural image distributions. Zero-shot CLIP models shrink this "robustness gap" by up to 75%. Linear fits on logit transformed values are shown with bootstrap estimated 95% confidence intervals. (Right) Visualizing distribution shift for bananas, a class shared across 5 of the 7 natural distribution shift datasets. The performance of the best zero-shot CLIP model, ViT-L/14@336px, is compared with a model that has the same performance on the ImageNet validation set, ResNet-101.

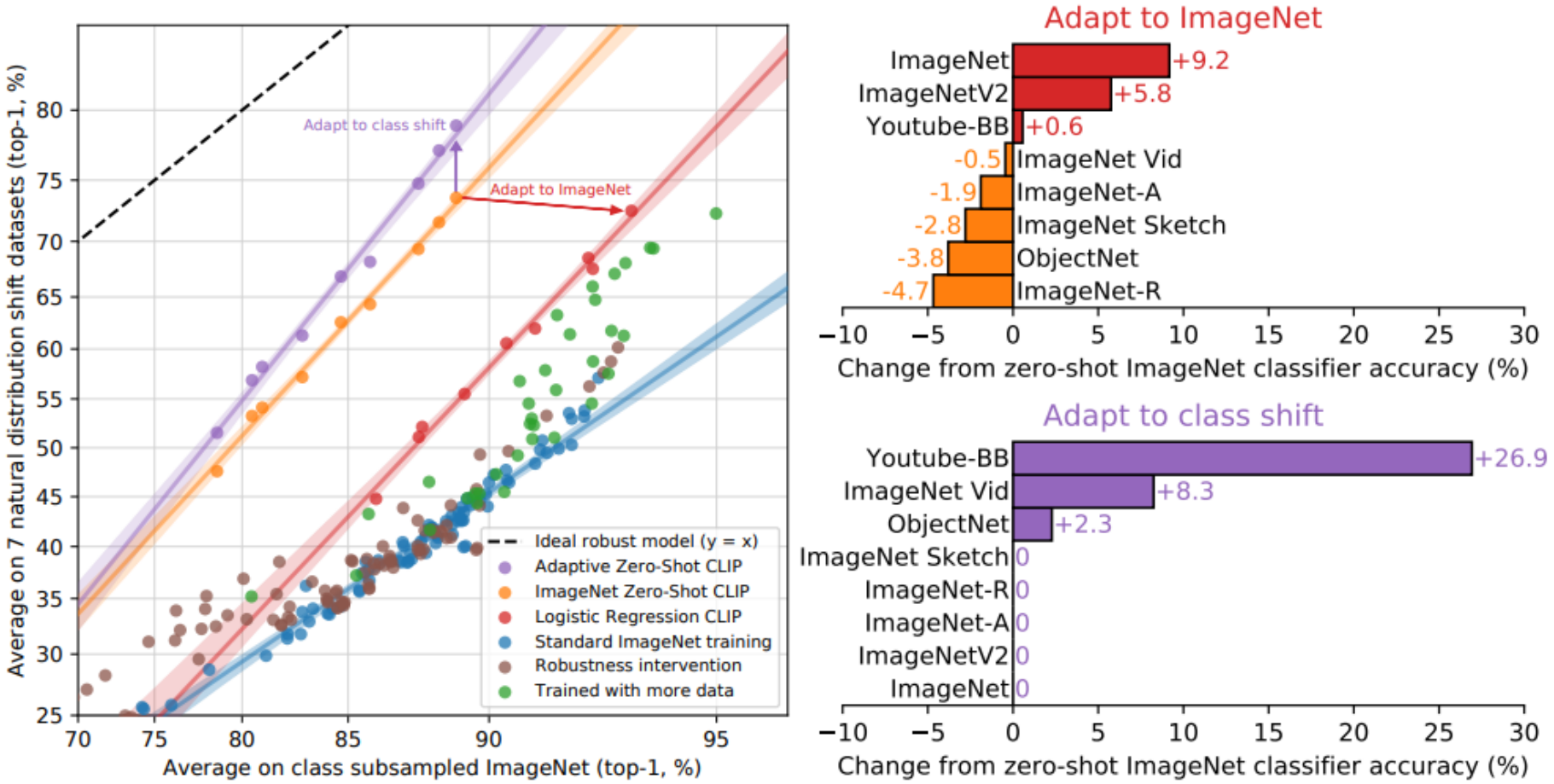


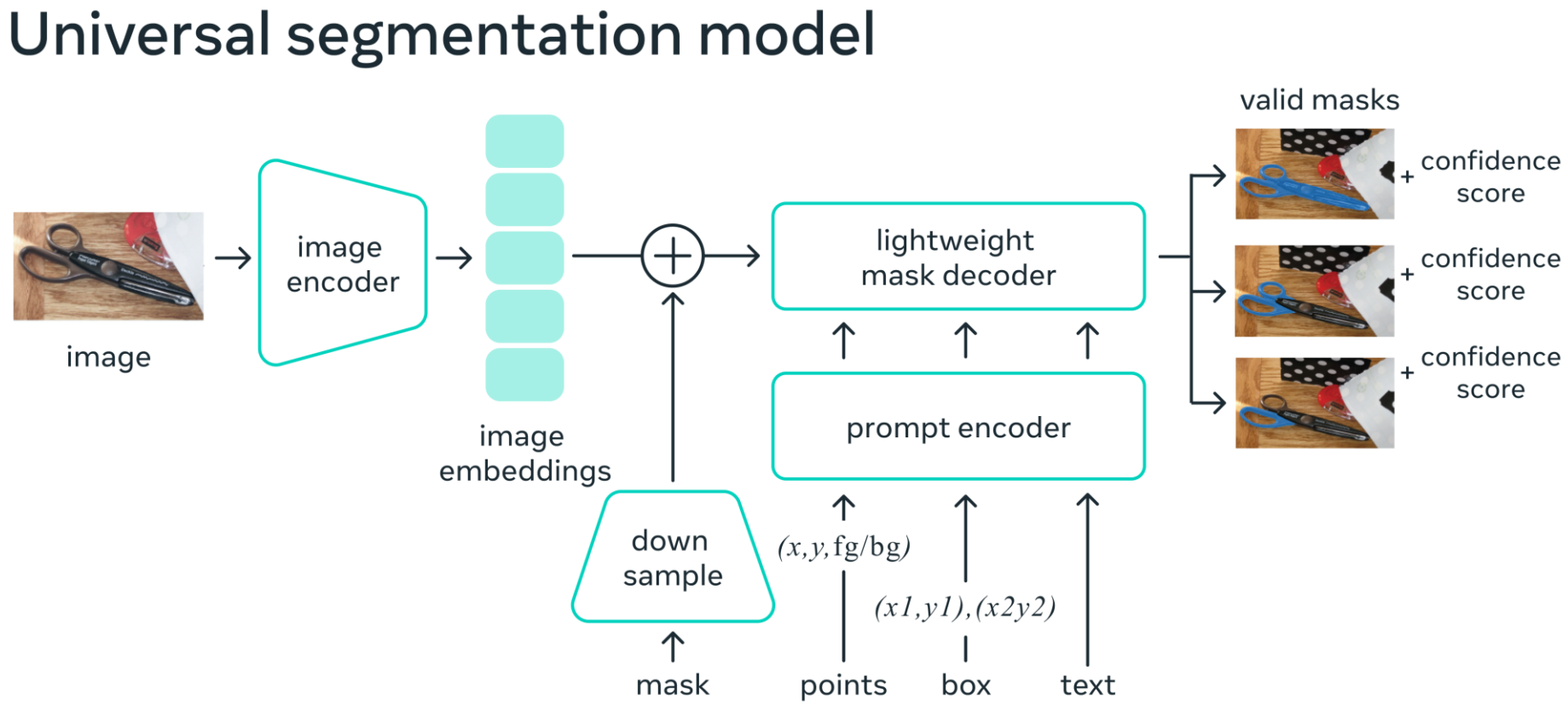
Figure 14. **While supervised adaptation to ImageNet increases ImageNet accuracy by 9.2%, it slightly reduces average robustness.** (Left) Customizing zero-shot CLIP to each dataset improves robustness compared to using a single static zero-shot ImageNet classifier and pooling predictions across similar classes as in Taori et al. (2020). CLIP models adapted to ImageNet have similar effective robustness as the best prior ImageNet models. (Right) Details of per dataset changes in accuracy for the two robustness interventions. Adapting to ImageNet increases accuracy on ImageNetV2 noticeably but trades off accuracy on several other distributions. Dataset specific zero-shot classifiers can improve accuracy by a large amount but are limited to only a few datasets that include classes which don't perfectly align with ImageNet categories.

Key Applications and Uses of CLIP in Real-World Scenarios

- Image generation model:
  - The SOTA image generation model from text like Dalle-3 and Midjourney uses the CLIP model to generate image embeddings from the input text embeddings to generate images consistent with the text.
- Image Segmentation model:
  - The popular SAM (Segment anything model) from meta uses CLIP to understand user prompts and generate segments from images based on user prompt.
- Image Captioning:
  - CLIP is used to get the best caption for the image.

- Content moderation:
  - social media sites employ CLIP to analyze images that can be harmful or do not comply with their policy and filter out them.
- Semantic Retrieval:
  - Text-to-image or image text searches are possible with CLIP embeddings.
- Image Search:
  - CLIP can be utilized to find images corresponding to a specific text query.
- Visual Question Answering (VQA):
  - CLIP has the ability to respond to queries regarding the visual elements of an image using a given text input.

Segment Anything (SAM)



[MASK] is All You Need

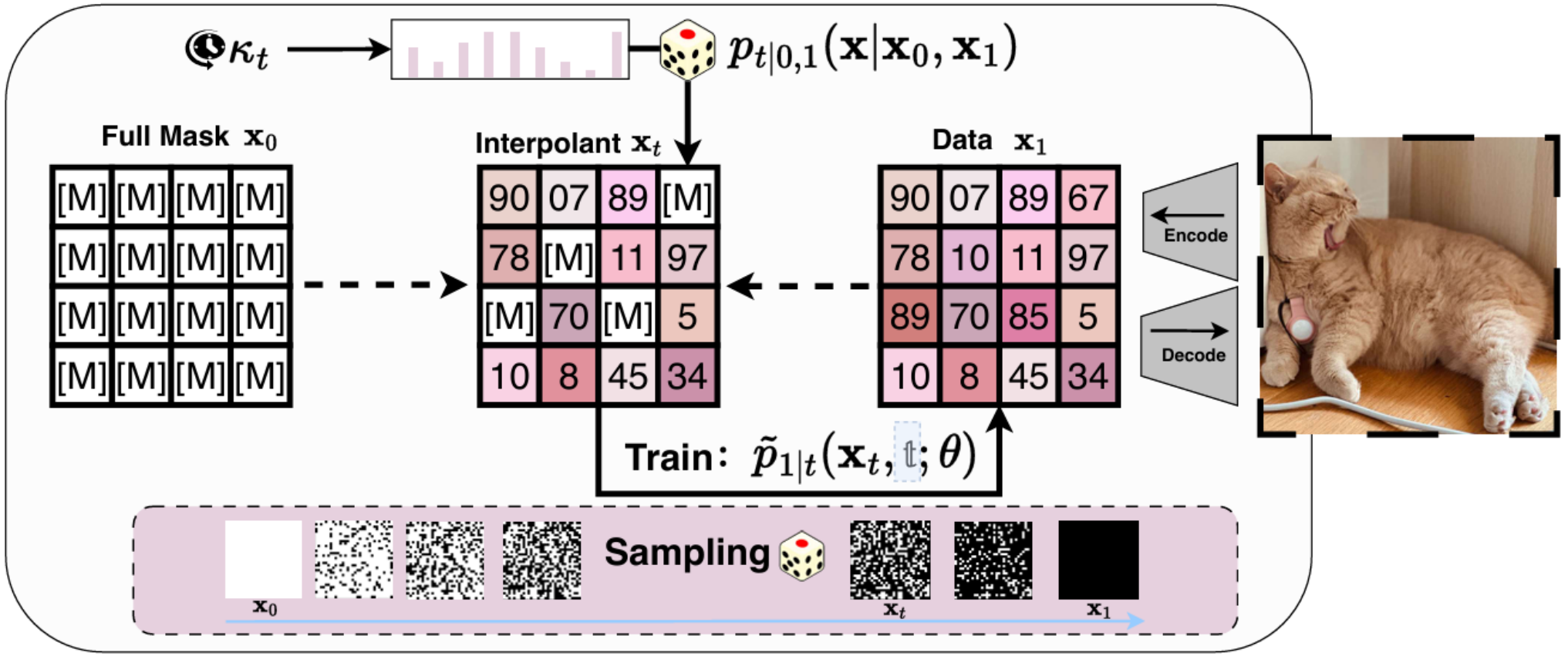


Figure 1. **Discrete Interpolants for training and sampling:** During training, we first obtain discrete interpolants x_t from x_0 and x_1 following a specific scheduler κ_t. We then train a model with the cross-entropy loss to predict the original data with $\tilde{p}_{1|t}(\mathbf{x}_t, \mathbb{t}; \theta)$, where $\mathbb{t}$ indicates that our timestep t is optional, leading to both Explicit Timestep and Implicit Timestep Models. For sampling, we begin with a fully masked x_0 and progressively unmask to reach the final fully unmasked x_1. Lastly, we decode the indices back to pixel space.

Type	Model	#Para.	FID↓	IS↑
AR & MGM Models	VQGAN [18]	1.4B	15.78	74.3
	VQGAN-re [18]	1.4B	5.20	280.3
	ViT-VQGAN [81]	1.7B	4.17	175.1
	ViT-VQGAN-re [81]	1.7B	3.04	227.4
	RQTran. [44]	3.8B	7.55	134.0
	RQTran.-re [44]	3.8B	3.80	323.7
	LlamaGen-XL [69]	775M	3.39	227.1
	MaskGIT [9]	227M	6.18	182
	Open-MAGVIT2-L [52]	804M	2.51	271.7
Continuous Diffusion	ADM [15]	554M	10.94	101.0
	CDM [32]	–	4.88	158.7
	LDM-4 [60]	400M	3.60	247.7
	DiT-XL/2 [58]	675M	2.27	278.2
Discrete	VQ-Diffusion [26]		5.32	–
DD & MGM	Explicit Timestep Model(*Our*)	546M	5.84	186.1
DD & MGM	Implicit Timestep Model(*Our*)	546M	**5.30**	183.0

Table 3. **Class-conditional generation on ImageNet** 256×256**.** MGM denotes Masked Generative Models. AR denotes Auto-Regressive Models. DD denotes Discrete Diffusion Models.

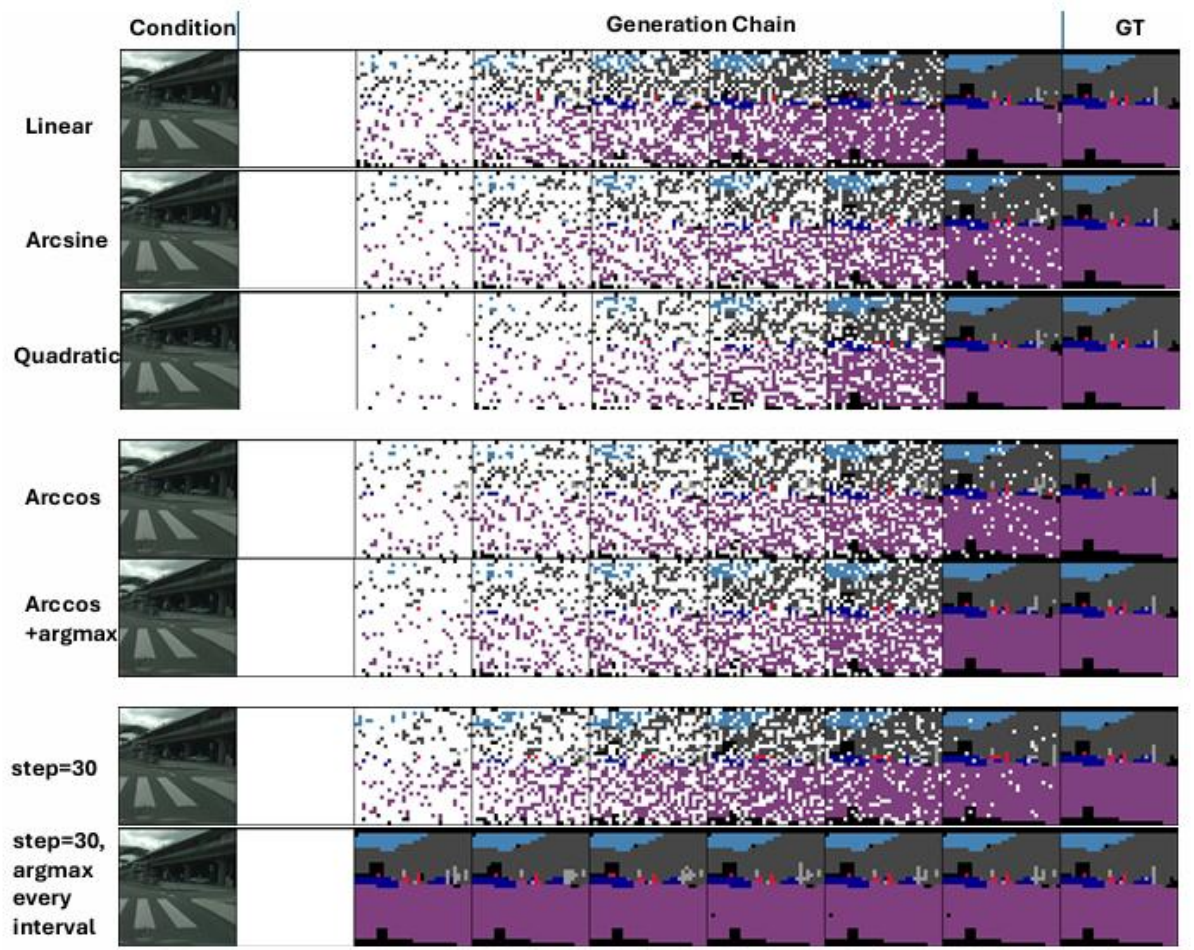


Figure 2. **Churning sampling by `argmax` can 1), alleviate the misalignment between schedulers. 2), boost sampling performance in low-NFE.** First, we visualize the progressive chain of changes when sampling with a scheduler κ_t that differs from the linear scheduler used during training. Our sampling process uses 50 steps and a CFG scale of 3. Second, we demonstrate that applying the `argmax` operation to logits can significantly reduce the occurrence of remaining `[MASK]` tokens after sampling.

Tweaking transformers

BIG LLM ARCHITECTURE COMPARISON

https://magazine.sebastianraschka.com/p/the-big-llm-architecture-comparison?fbclid=IwY2xjawLztTtleHRuA2FlbQIxMABicmlkETF6WHlDV3VNR1V1b0N5VzRBAR6RqD5hgCq5gBbyzGqNhcJJmYodbYoPpxGHSD8oChLX86qPvptQ2E_2EZC53Q_aem_RpcBqpJBlVS02Rm5L6rJEw&hide_intro_popup=true

Tweaking Transformers

- The Transformer architecture has not changed much since 2017.
- But a few changes have become common in modern LLM
  - Post-LN => Pre-RMSNorm
    - Move normalization inside residual
  - LN => RMSNorm
  - SwiGLU activation
  - Mixture of Experts (MoE)

Pre-LN vs Post-LN

- Post-LN
  - LN is outside the residual connections
  - Used in original transformer model (original BERT and GPT)
  - Work well when training is stable and not very deep
- Pre-LN
  - Move LN before self-attention and MLP, inside the residual connections
  - Applied before residual connection
  - Stability and Training Depth
  - Support deeper network due to help in mitigating the vanishing gradient problem
  - Less learning rate/initialization sensitivity
    - Allow aggressive training schedules
  - Used in GPT-3 and later

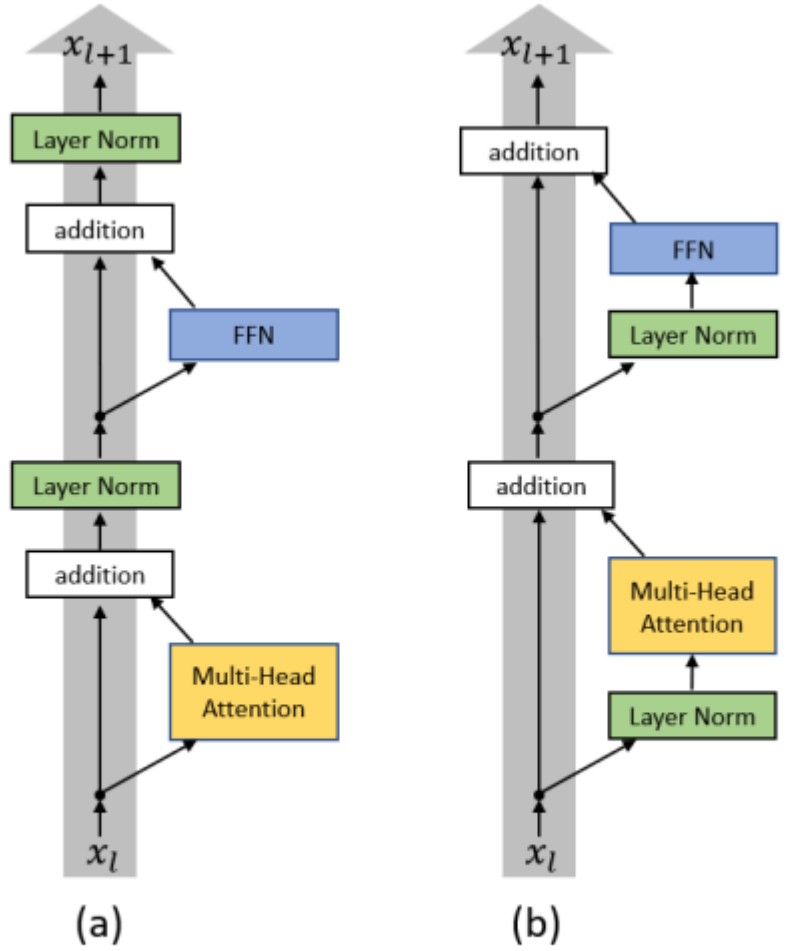


Figure 1. (a) Post-LN Transformer layer; (b) Pre-LN Transforme layer.

https://arxiv.org/abs/2002.04745

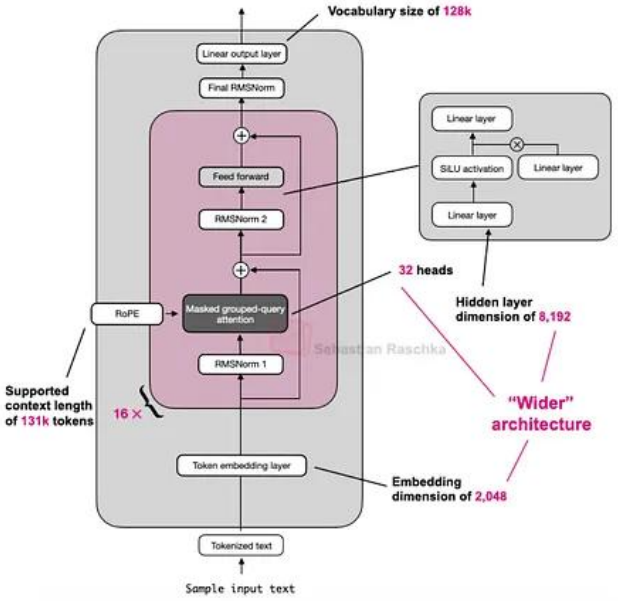
Post-LN 在深層模型中，梯度可能在殘差加法後放大

為什麼會「放大」？

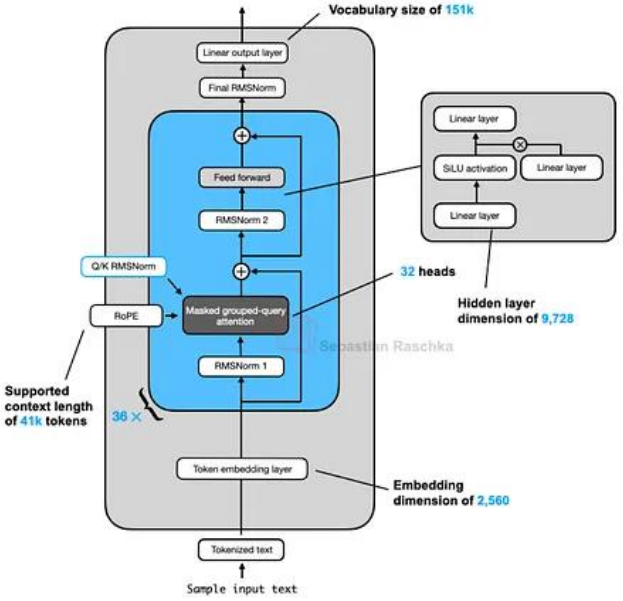
- **殘差加法的作用**：殘差連接本意是讓梯度穩定傳遞，但如果 $f(x)$的梯度貢獻很大（例如，權重初始化不當或層數深時，子層的輸出變異數增加），則在加法後 z的梯度會被放大。想像一下：如果捷徑梯度是 1，子層梯度是 1.5，總和就變成 2.5，傳到上一層時更大。
- **LN 的位置問題**：在 Post-LN 中，LN 是放在加法之後，所以它只能正規化最終輸出 o，但無法防止加法過程中的梯度放大。結果是，梯度在層間傳遞時，可能在深層累積變大（尤其是從模型頂層向下傳時）。
- **深層模型的放大效應**：在淺層模型（層數少，如 6-12 層）中，這問題不明顯，因為累積層數少。但在深層模型（如 100+ 層的 LLM）中，梯度要穿過許多層，每層的殘差加法都可能輕微放大，導致指數級增長。這就像複利效應：每層放大 1.1 倍，10 層後是 2.59 倍，100 層後可能爆炸到天文數字。

LLM

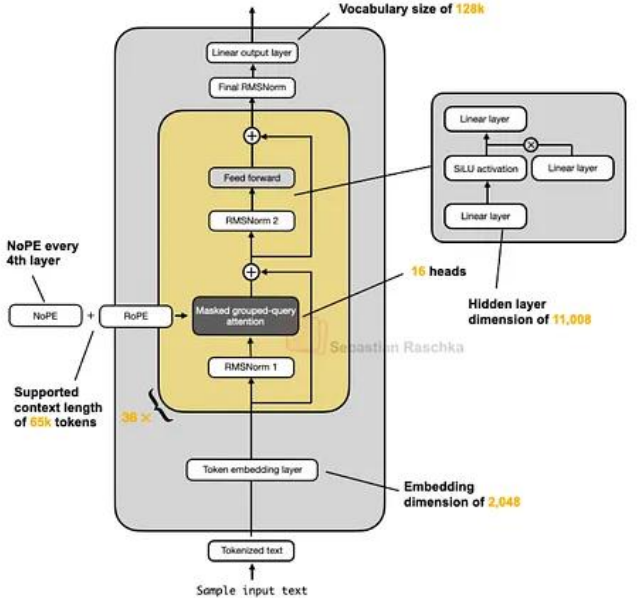
Llama 3.2 1B



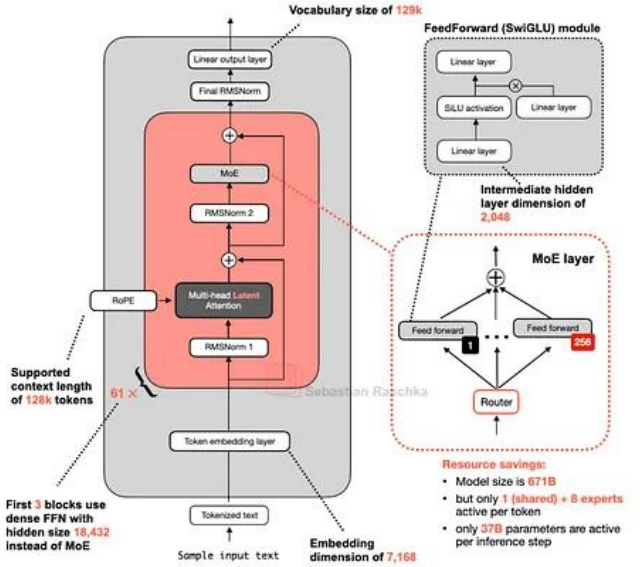
Qwen3 4B



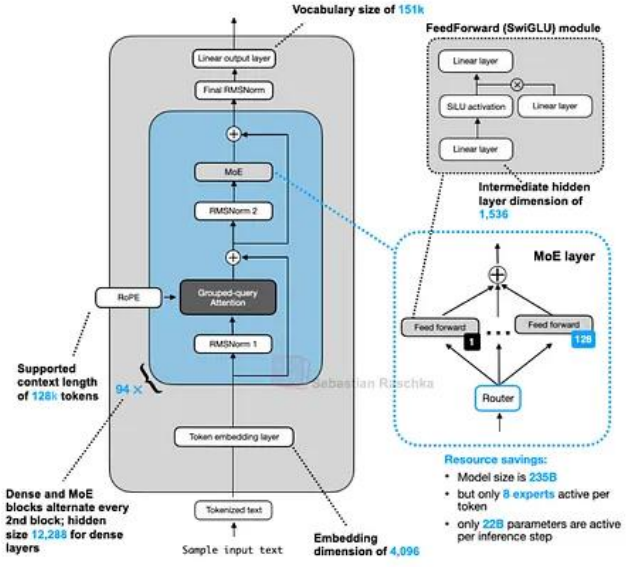
SmolLM3 3B



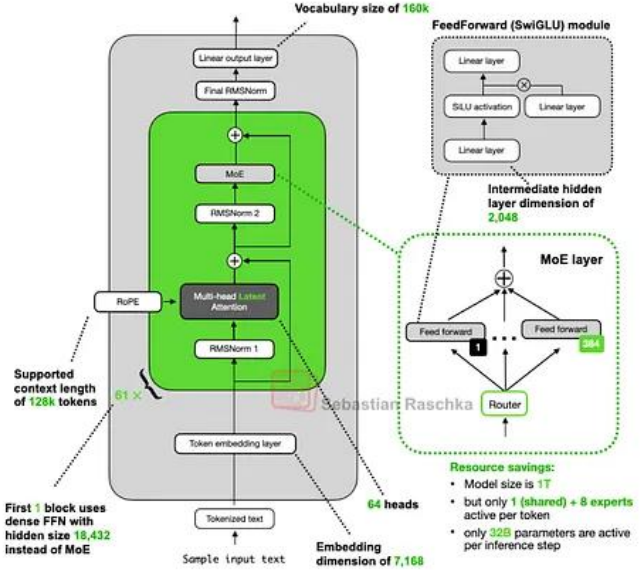
DeepSeek V3 (671B)



Qwen3 235B-A22B



Kimi K2 (1 trillion)

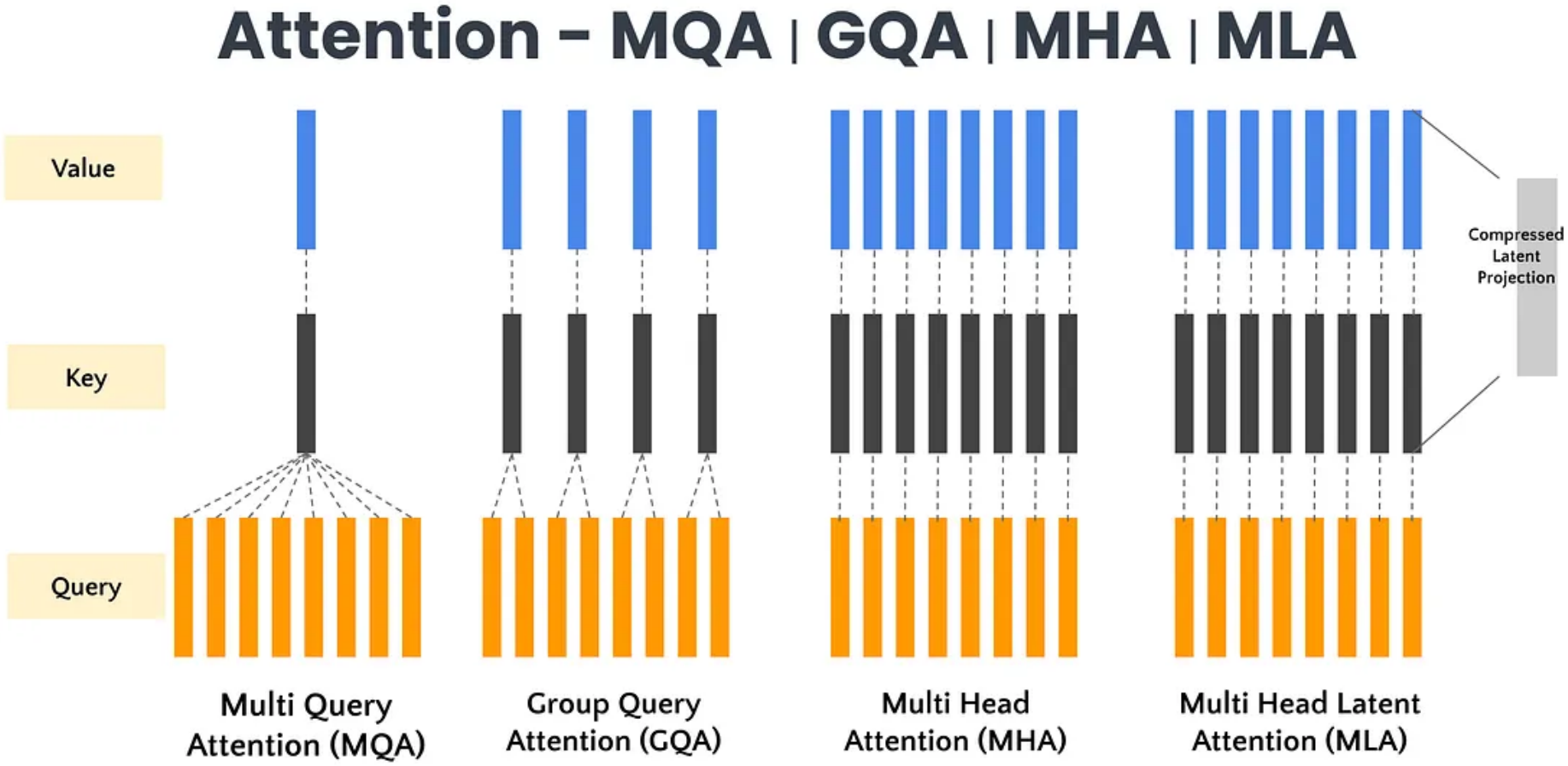


特性 / 技術	經典 Transformer / GPT-2	Llama 系列 / Qwen3	DeepSeek / Kimi	主要優勢與創新
注意力機制 (Attention)	多頭注意力 (MHA)	分組查詢注意力 (GQA)	多層注意力 (MLA)	GQA: 透過在多個查詢頭之間共享鍵和值，減少計算和記憶體使用，成為新的標準。 MLA: 進一步將鍵值張量壓縮到較低維度空間，顯著減少 KV 快取的記憶體佔用，推理時效率更高
專家混合 (MoE)	不適用	Mixtral 等模型開始採用	是 (DeepSeek-V3)	在巨大的模型中（如 DeepSeek 671B），每次只激活一小部分參數（37B），大幅提升訓練和推理效率，同時保持高性能。
正規化 (Normalization)	Post-LN (後層正規化)	Pre-RMSNorm (事前均方根正規化)	Pre-RMSNorm	Pre-RMSNorm: 改善了訓練的穩定性並成為現代 LLM 的主流選擇。
位置編碼 (Positional Encoding)	絕對位置編碼 (Absolute)	旋轉位置編碼 (RoPE)	旋轉位置編碼 (RoPE)	RoPE: 能更好地處理長序列文本，並成為了如 Llama、Gemma 等模型的核心技術。
整體架構	編碼器-解碼器或純解碼器	純解碼器 (Decoder-Only)	純解碼器 (Decoder-Only)	純解碼器架構：已成為生成式語言模型的主流，專注於根據前面的文本預測下一個詞元。

Feature / Technology	Classic Transformer / GPT-2	Llama Series / Qwen3	DeepSeek / Kimi	Key Advantages & Innovations
Attention Mechanism	Multi-Head Attention (MHA)	Grouped-Query Attention (GQA)	Multi-Layer Attention (MLA)	GQA: Reduces computation and memory by sharing keys and values across multiple query heads, becoming the new standard. MLA: Further compresses key-value tensors into a lower-dimensional space, significantly reducing KV cache memory usage for higher inference efficiency.
Mixture of Experts (MoE)	Not Applicable	Adopted in models like Mixtral	Yes (DeepSeek-V3)	In massive models (e.g., DeepSeek 671B), only a small fraction of parameters (37B) are activated per token, drastically improving training and inference efficiency while maintaining high performance.
Normalization	Post-Layer Normalization (Post-LN)	Pre-Root Mean Square Norm (Pre-RMSNorm)	Pre-Root Mean Square Norm	Pre-RMSNorm: Improves training stability and has become the mainstream choice for modern LLMs.
Positional Encoding	Absolute Positional Encoding	Rotary Position Embedding (RoPE)	Rotary Position Embedding (RoPE)	RoPE: Better handles long text sequences and has become a core technology in models like Llama and Gemma.
Overall	Encoder-Decoder or			Decoder-Only Architecture: Has become the dominant paradigm for generative language models, focusing on predicting the next token

DeepSeek V3

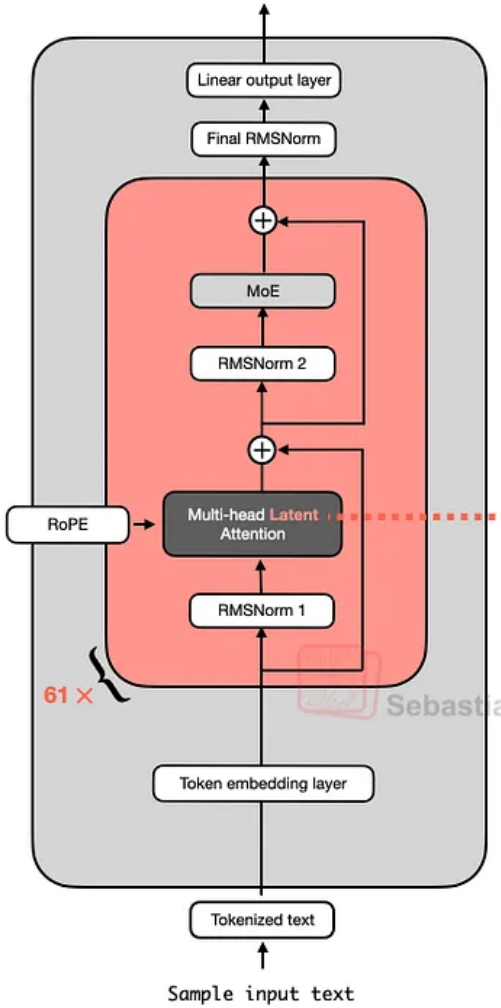
- Multi-Head Latent Attention (MLA)



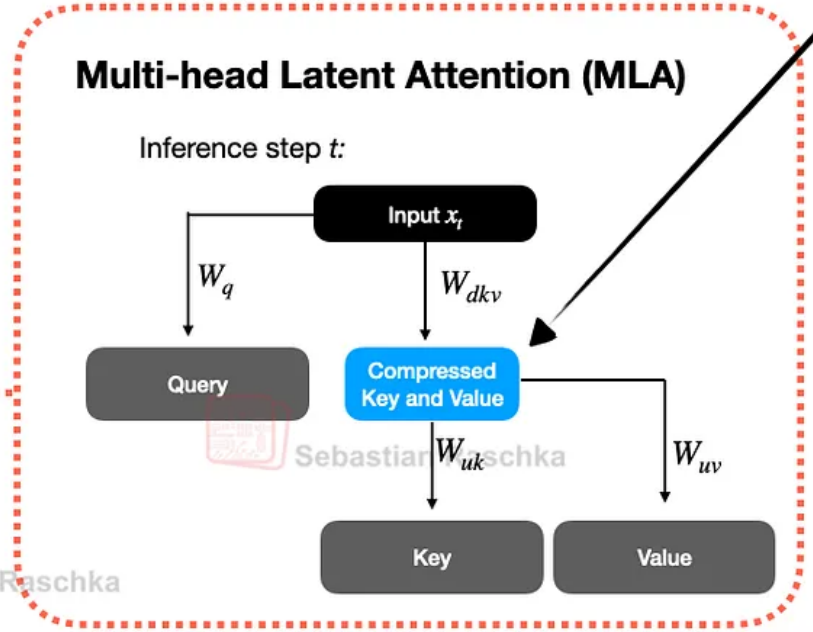
https://verticalserve.medium.com/group-query-attention-58283b337c65

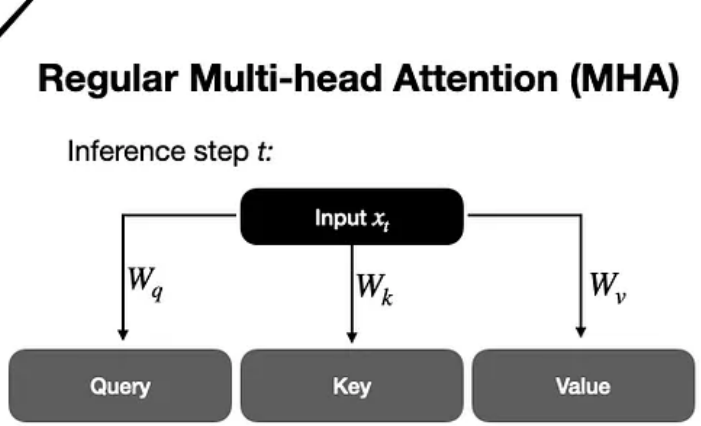
Feature / Aspect	MHA (Multi-Head Attention)	MQA (Multi-Query Attention)	GQA (Grouped Query Attention)	MLA (Multi-Latent Attention)
Core Mechanism	Splits input into multiple heads, each processing the full set of queries, keys, and values.	Uses a single, shared key-value pair for all attention heads.	Groups tokens into smaller subsets to perform attention within each group.	Compresses the sequence into latent embeddings and performs attention on them.
Computational Complexity	Quadratic (**$O(n^2)$**), computationally expensive.	Significantly reduced compared to MHA.	Reduced complexity compared to MHA.	**Linear or near-linear** ($O(n)$), enabling faster inference.
Context Representation	Captures diverse, token-level interactions across multiple heads.	Compromises on nuanced token interactions due to the shared key-value pair.	Focuses on **localized** attention within groups; may miss global dependencies.	Captures **global** patterns efficiently through latent embeddings, preserving rich context.
Key Advantage	High expressiveness and ability to capture diverse contexts.	Very low memory overhead, ideal for large models.	Efficient for tasks requiring localized attention.	Balances efficiency and expressiveness; faster inference and better generalization.
Primary Limitation	High computational and memory cost, scales poorly with sequence	Sacrifices token-level granularity and	Can fail to capture important global dependencies across	(Not specified) Overcomes the limitations of the other

DeepSeek V3/R1



Key idea: Reduces memory usage in KV cache





Attention Mechanism	KV Cache per Token (# Element)	Capability
Multi-Head Attention (MHA)	$2n_h d_h l$	Strong
Grouped-Query Attention (GQA)	$2n_g d_h l$	Moderate
Multi-Query Attention (MQA)	$2d_h l$	Weak
MLA (Ours)	$(d_c + d_h^R)l \approx \frac{9}{2} d_h l$	Stronger

Unlike what previous papers found, this paper finds that MHA performs much BETTER than GQA

Benchmark (Metric)	# Shots	Dense 7B w/ MQA	Dense 7B w/ GQA (8 Groups)	Dense 7B w/ MHA
# Params	-	7.1B	6.9B	6.9B
BBH (EM)	3-shot	33.2	35.6	**37.0**
MMLU (Acc.)	5-shot	37.9	41.2	**45.2**
C-Eval (Acc.)	5-shot	30.0	37.7	**42.9**
CMMLU (Acc.)	5-shot	34.6	38.4	**43.5**

Table 8 | Comparison among 7B dense models with MHA, GQA, and MQA, respectively. MHA demonstrates significant advantages over GQA and MQA on hard benchmarks.

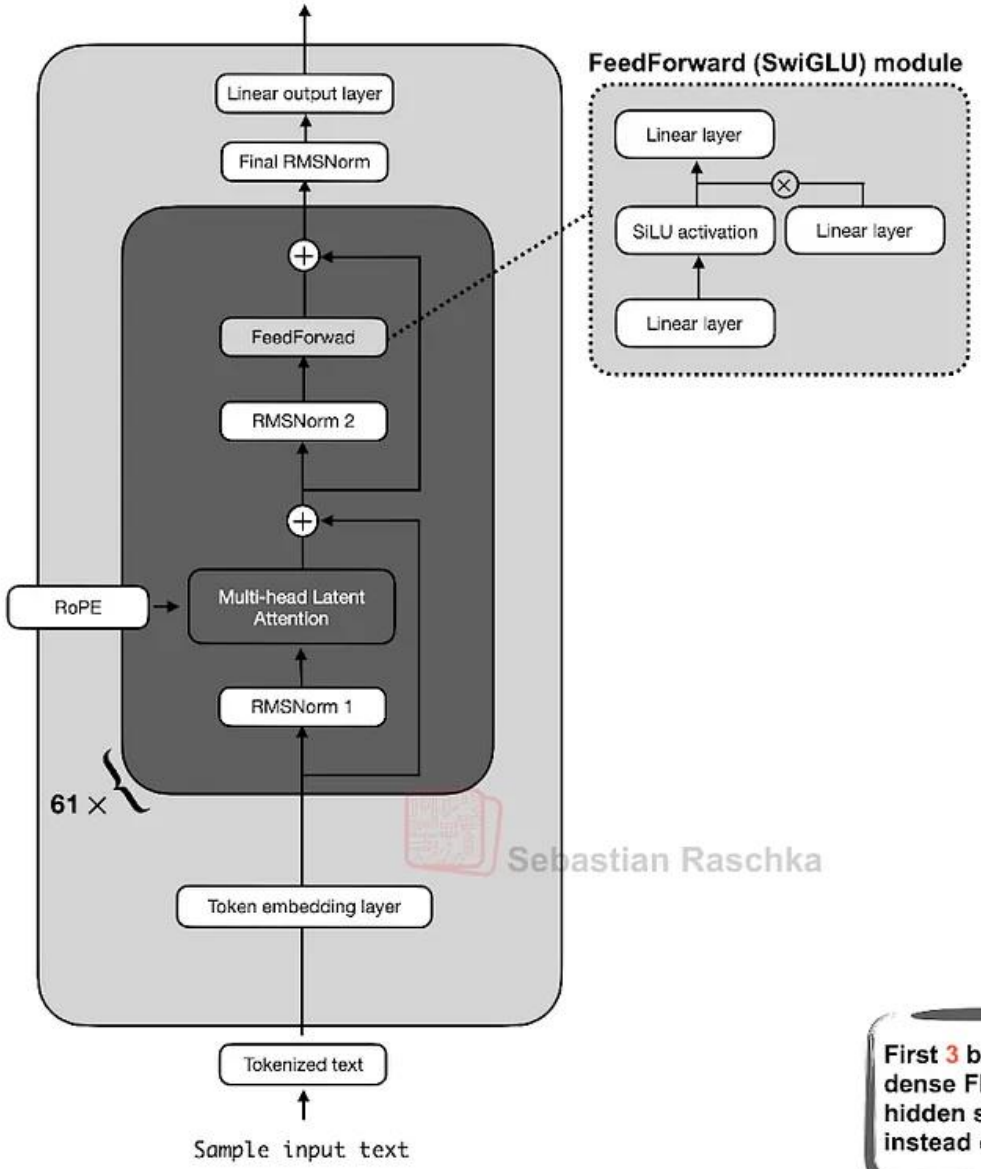
Benchmark (Metric)	# Shots	Small MoE w/ MHA	Small MoE w/ MLA	Large MoE w/ MHA	Large MoE w/ MLA
# Activated Params	-	2.5B	2.4B	25.0B	21.5B
# Total Params	-	15.8B	15.7B	250.8B	247.4B
KV Cache per Token (# Element)	-	110.6K	15.6K	860.2K	34.6K
BBH (EM)	3-shot	37.9	**39.0**	46.6	**50.7**
MMLU (Acc.)	5-shot	48.7	**50.0**	57.5	**59.0**
C-Eval (Acc.)	5-shot	**51.6**	50.9	57.9	**59.2**
CMMLU (Acc.)	5-shot	52.3	**53.4**	60.7	**62.5**

Table 9 | Comparison between MLA and MHA on hard benchmarks. DeepSeek-V2 shows better performance than MHA, but requires a significantly smaller amount of KV cache.

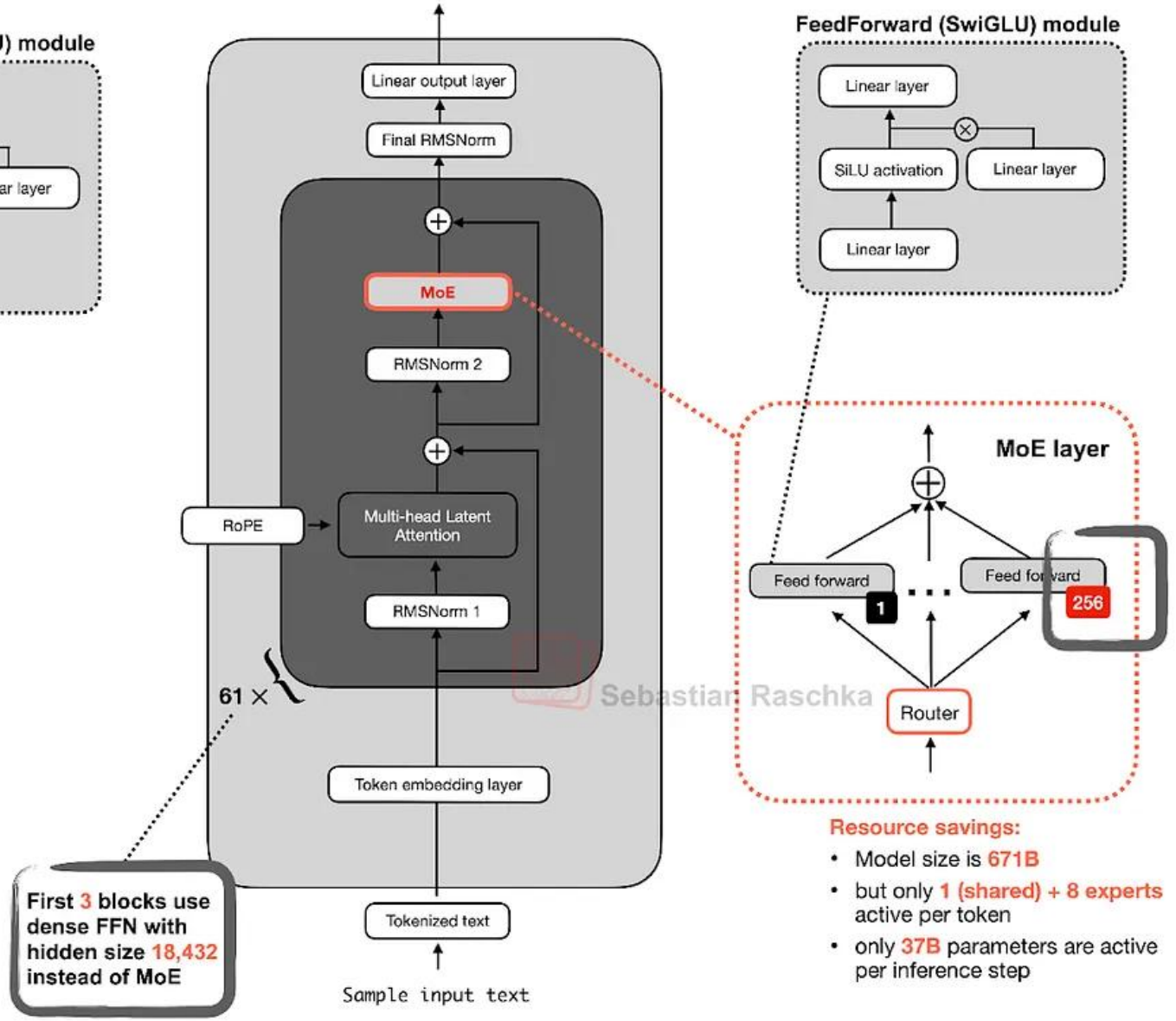
The memory requirements for MLA are much lower than for MHA

MLA performs better than MHA (here tested on Mixture-of-Experts architectures)

Architecture without MoE ("dense")



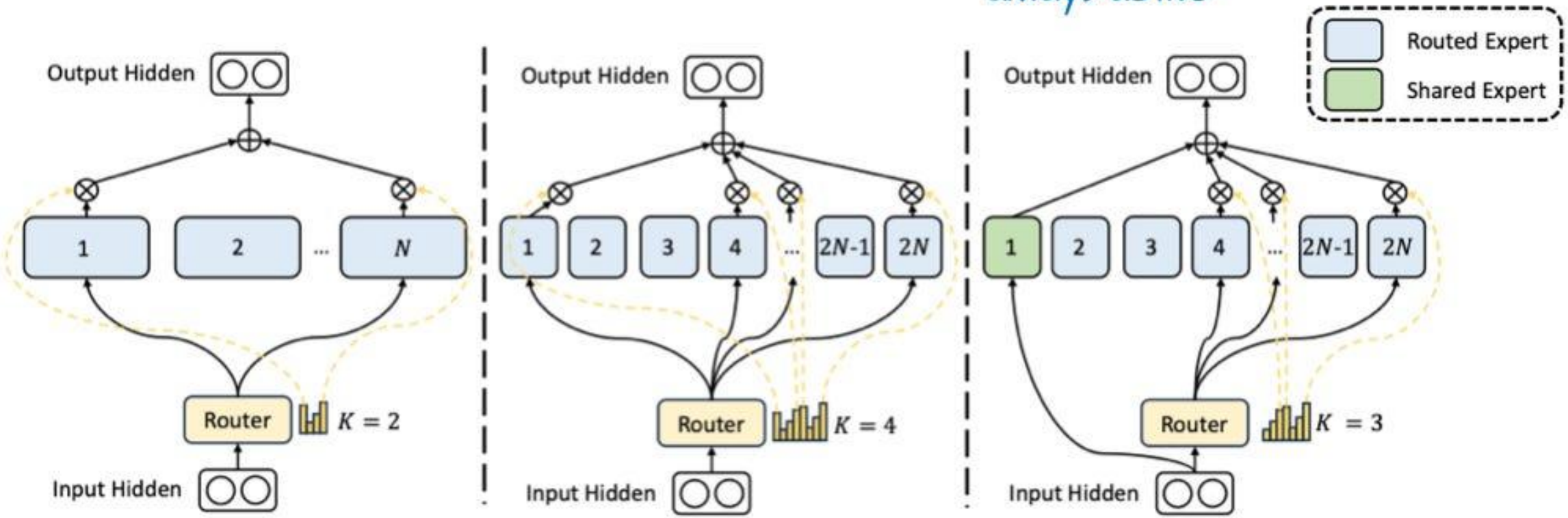
DeepSeek V3/R1 with MoE ("sparse")



Early MoE: Has bigger and fewer experts, and activates only a few experts (here: 2)

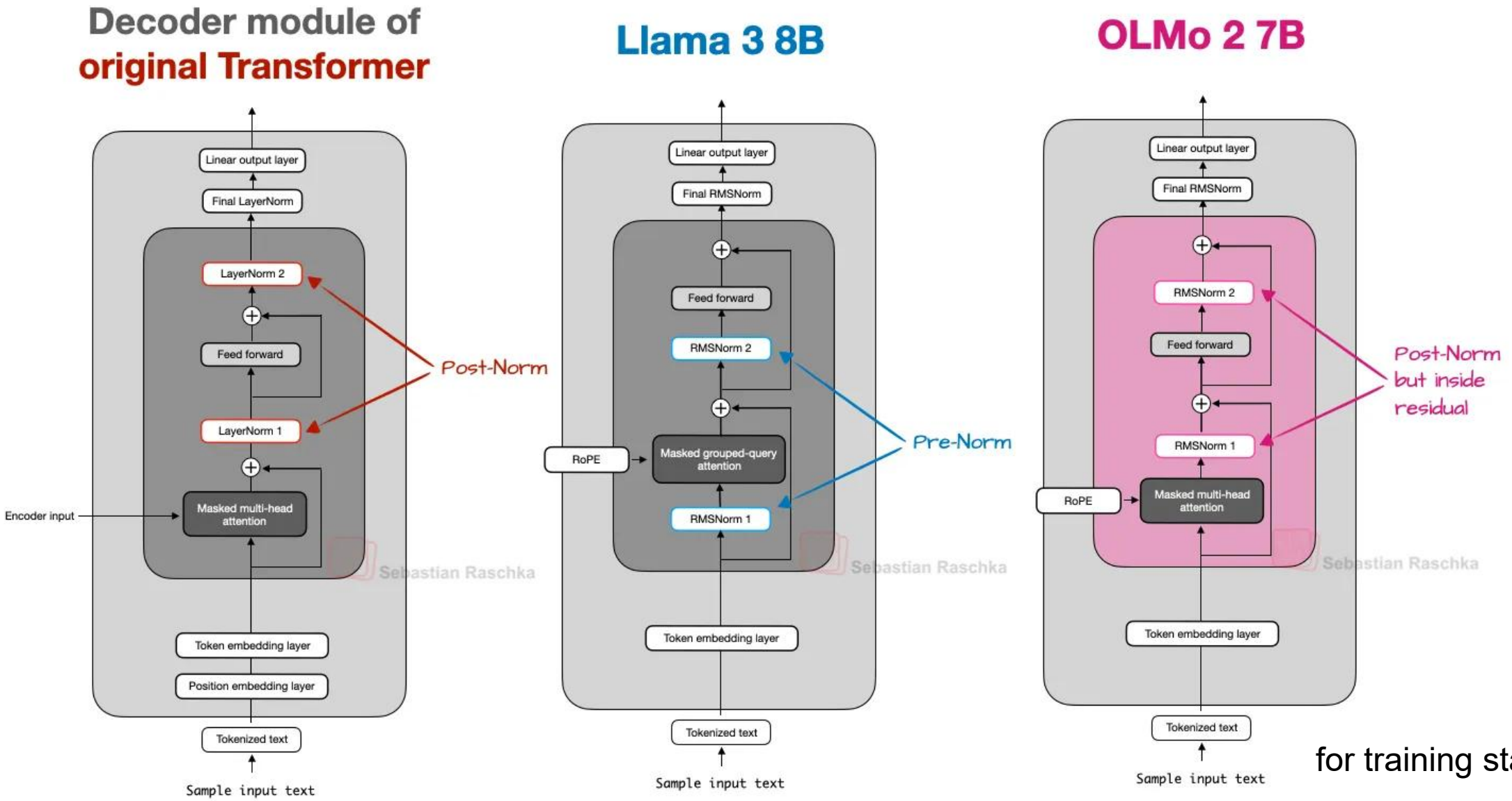
Fine-grained MoE uses more but smaller experts, and activates more experts (here: 4)

MoE with shared expert: also uses many small experts, but adds a shared expert that is always active

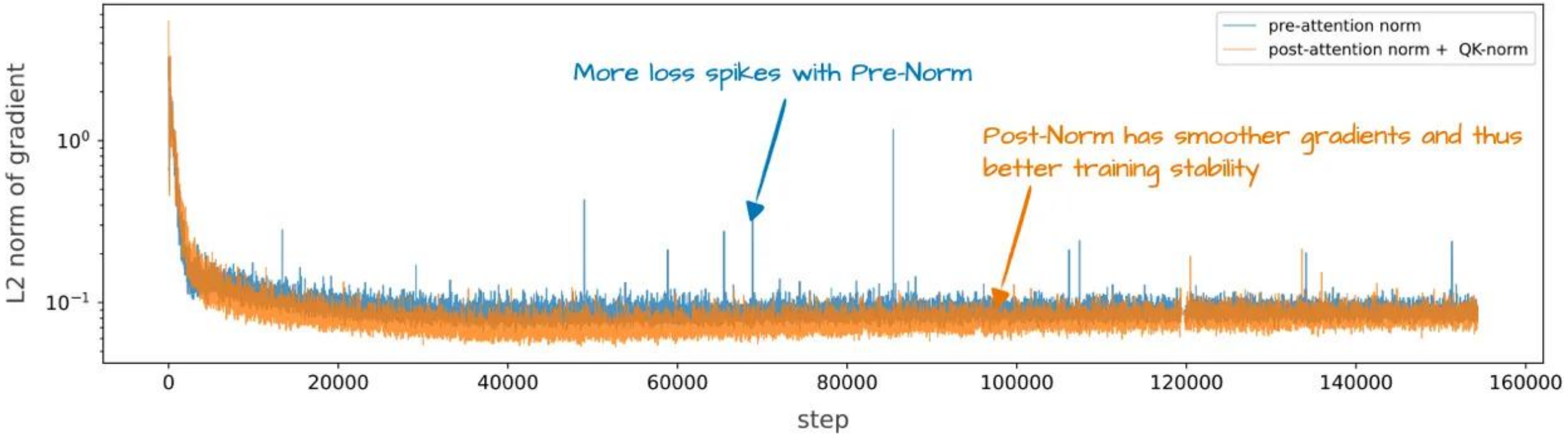


(a) Conventional Top-2 Routing → **(b) + Fine-grained Expert Segmentation** → **(c) + Shared Expert Isolation (DeepSeekMoE)**

OLMO2: Post-Norm



for training stability



A plot showing the training stability for Pre-Norm (like in GPT-2, Llama 3, and many others) versus OLMo 2's flavor of Post-Norm. This is an annotated figure from the OLMo 2 paper, https://arxiv.org/abs/2501.00656

QK-Norm in Gemma 2

- yet another RMSNorm
- placed inside the Multi-Head Attention (MHA) module and applied to the queries (q) and keys (k) before applying RoPE

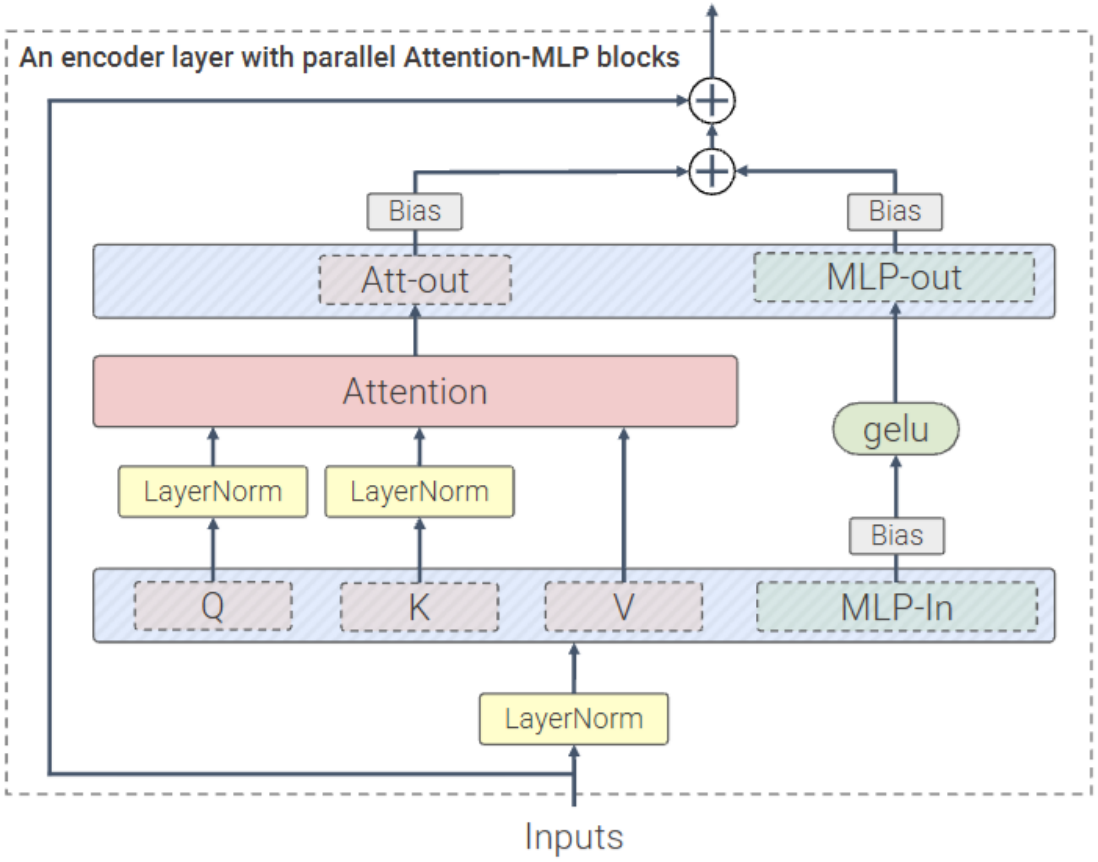
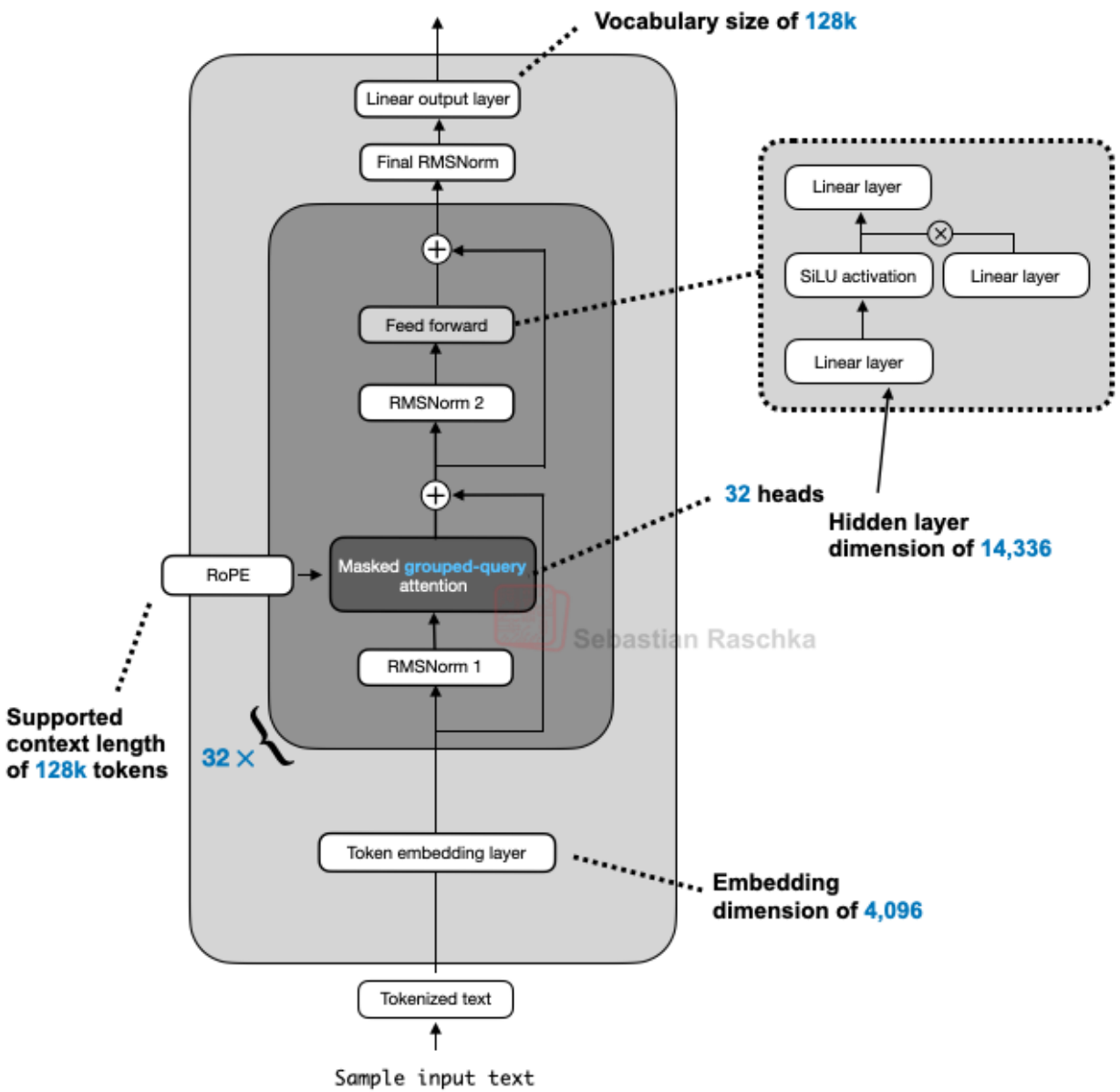


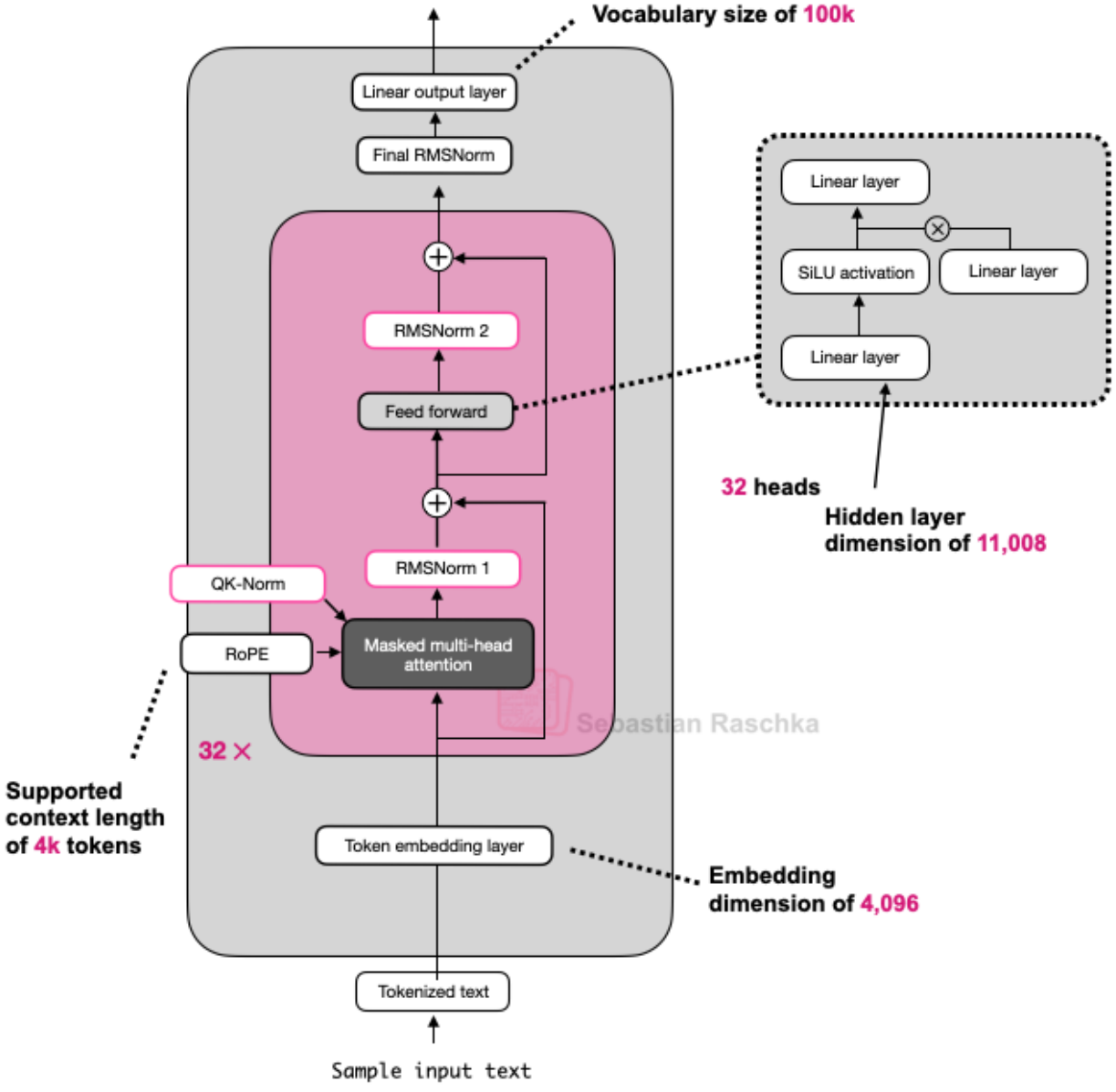
Figure 2: Parallel ViT-22B layer with QK normalization.

https://arxiv.org/abs/2302.05442

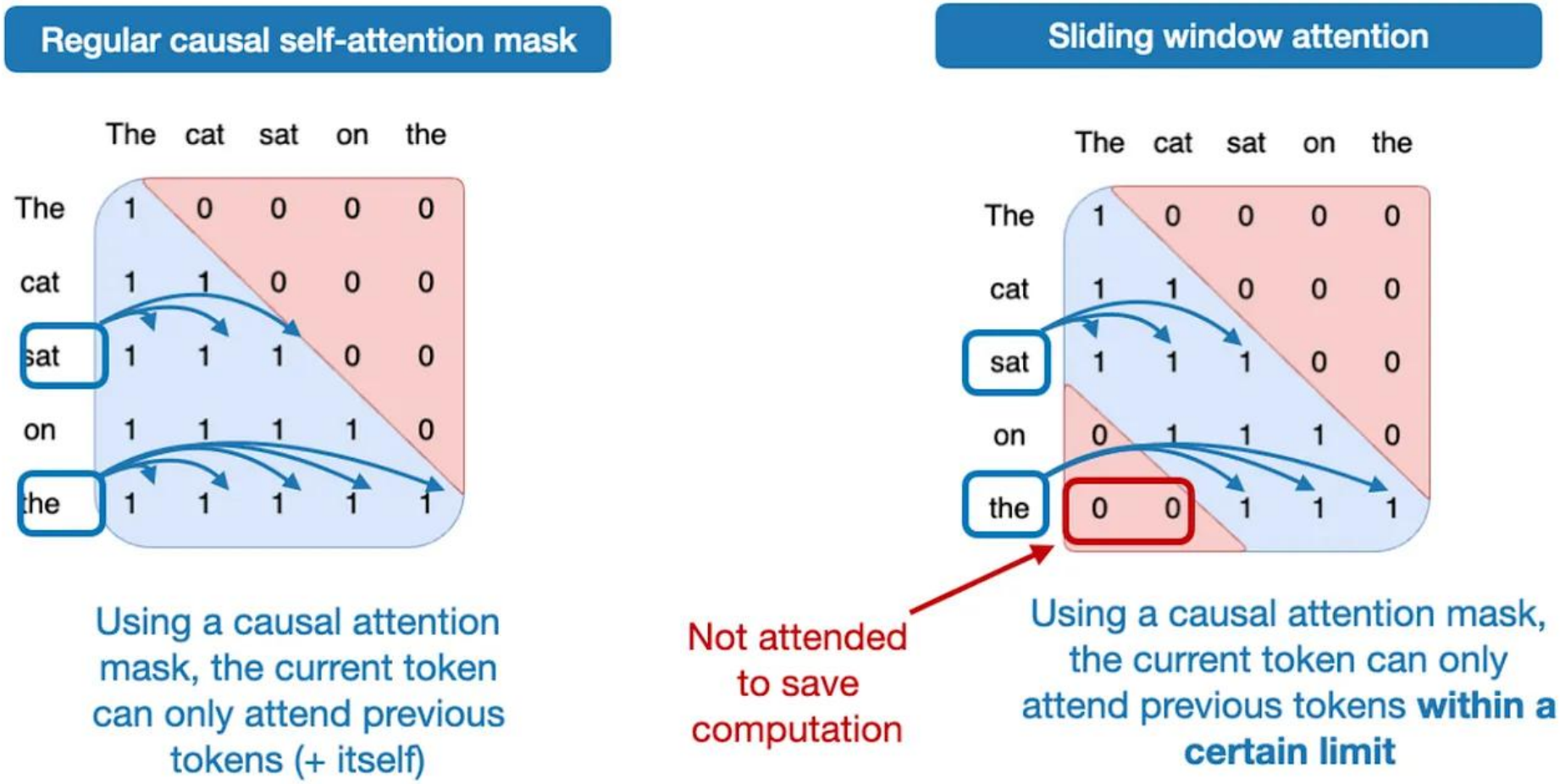
Llama 3 8B

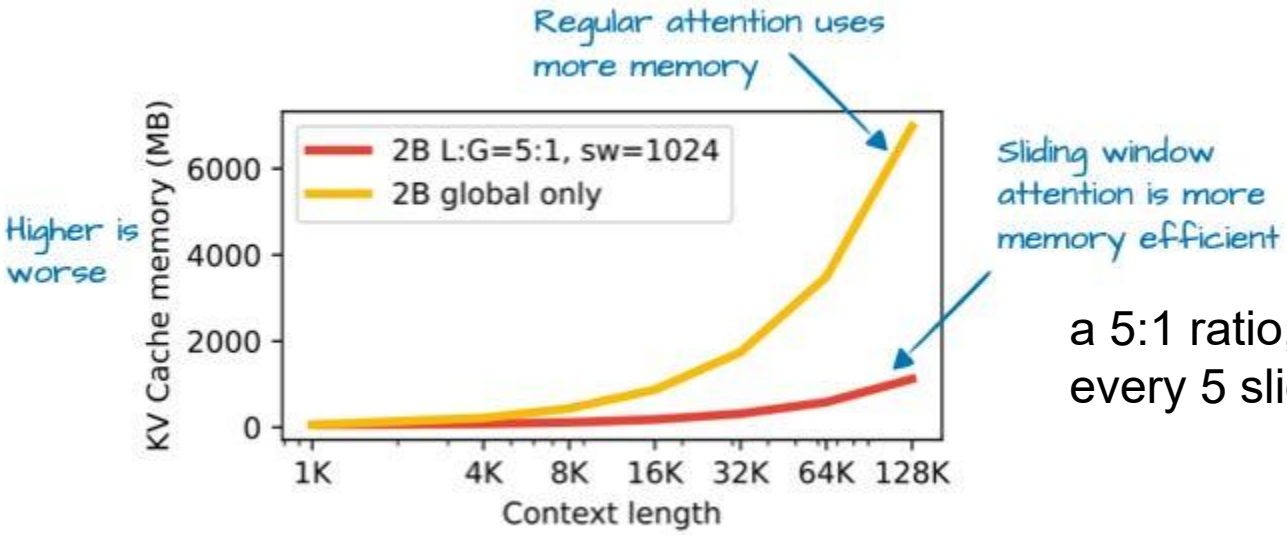


OLMo 2 7B

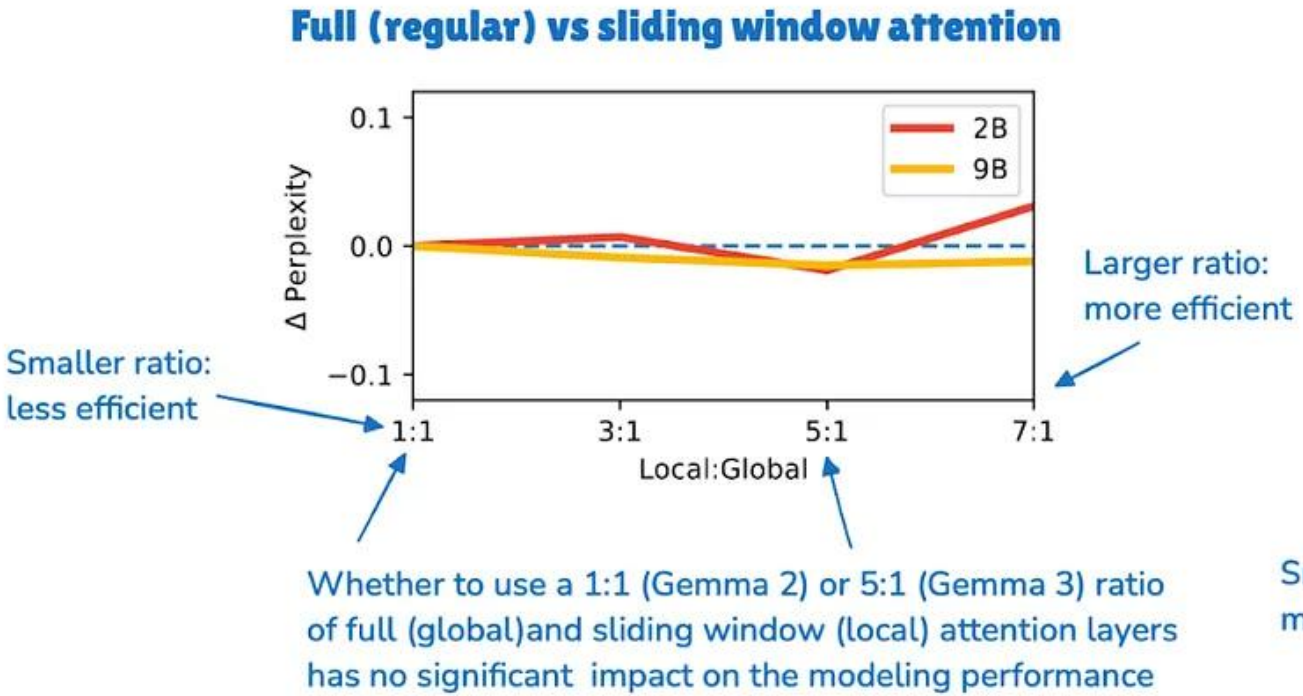


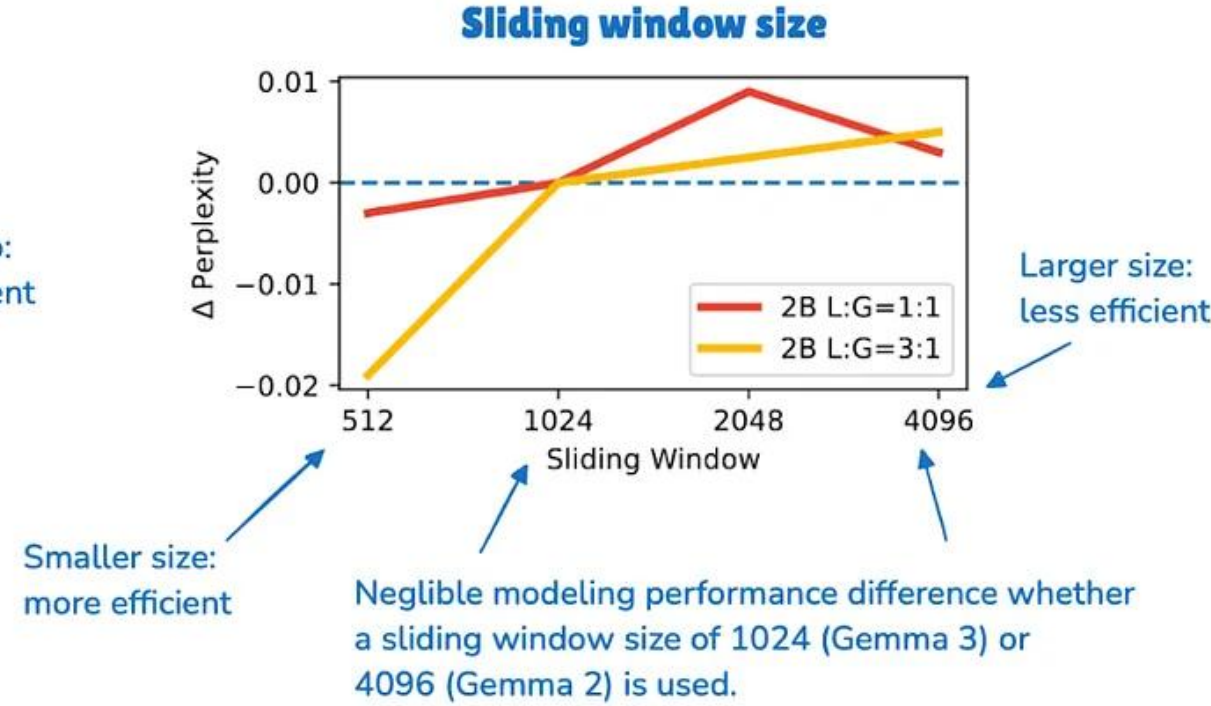
Gemma 3: Sliding Window Attention



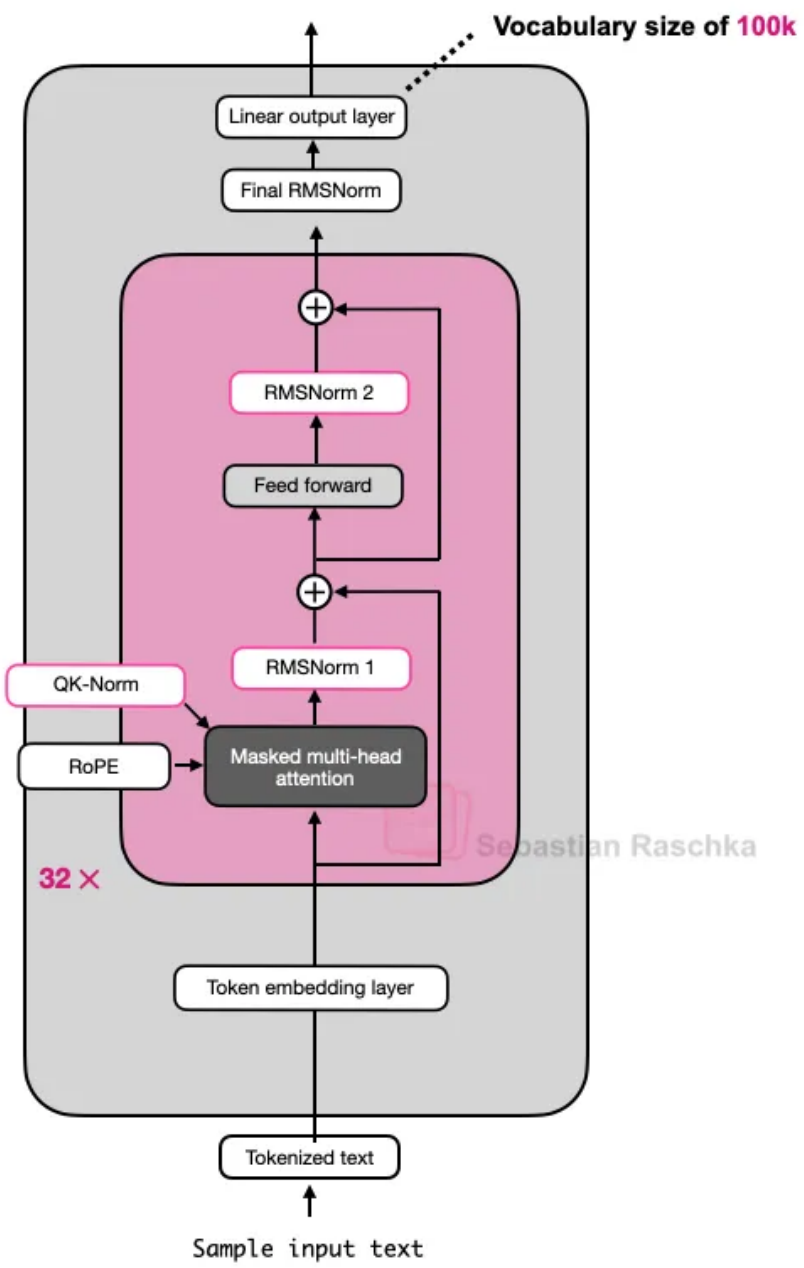


a 5:1 ratio, meaning there's only 1 full attention layer for every 5 sliding windows (local) attention layers

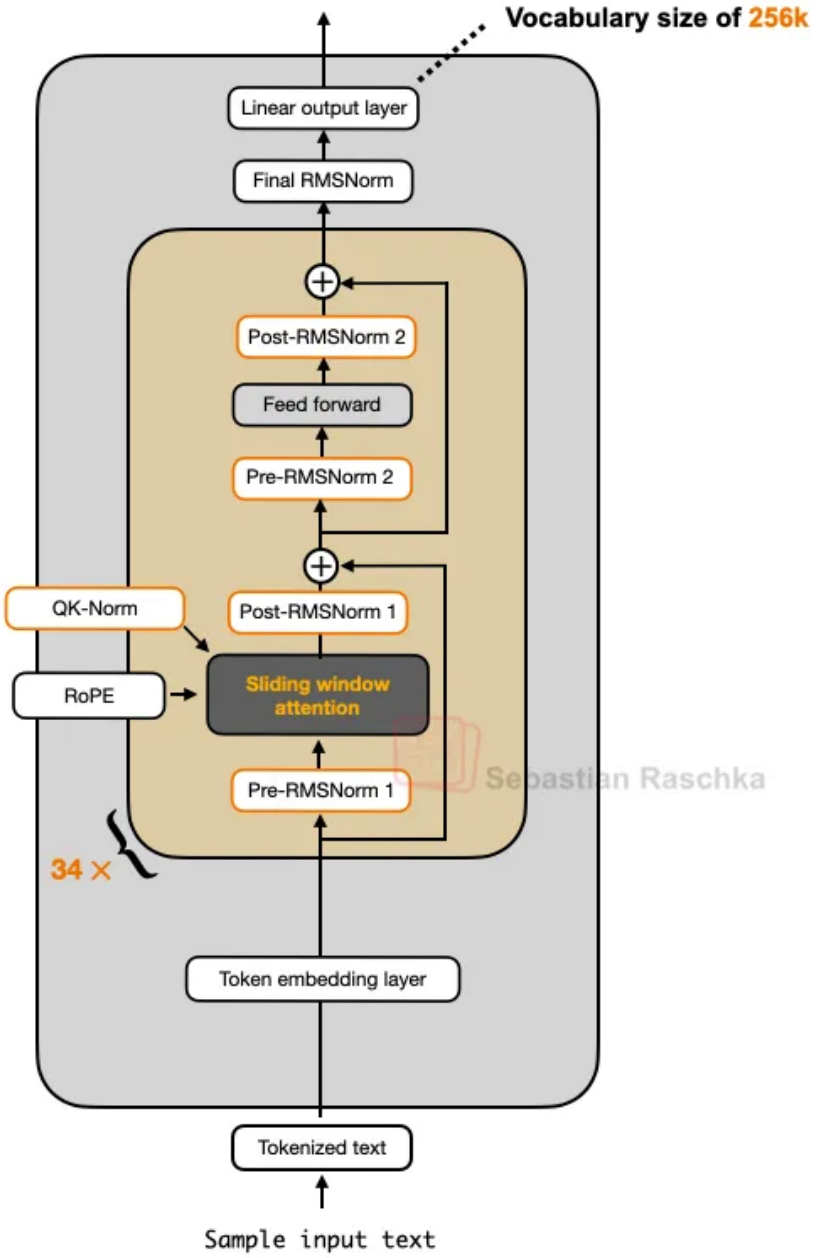




OLMo 2 7B

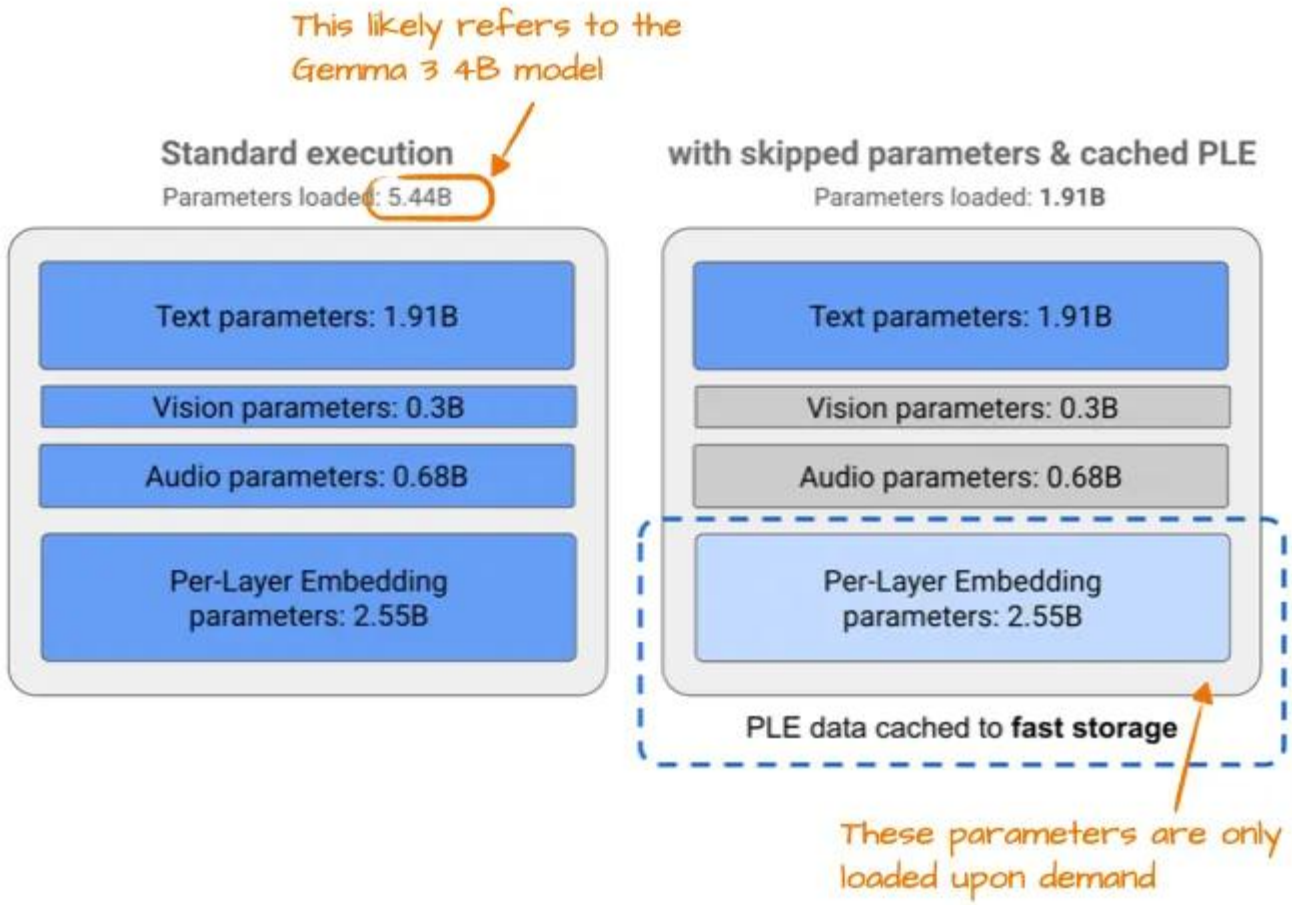


Gemma 3 4B



Gemma 3n: Per-Layer Embedding (PLE)

- Idea
  - keep only a subset of the model's parameters in GPU memory.
  - Token-layer specific embeddings, such as those for text, audio, and vision modalities, are then streamed from the CPU or SSD on demand.



MatFormer

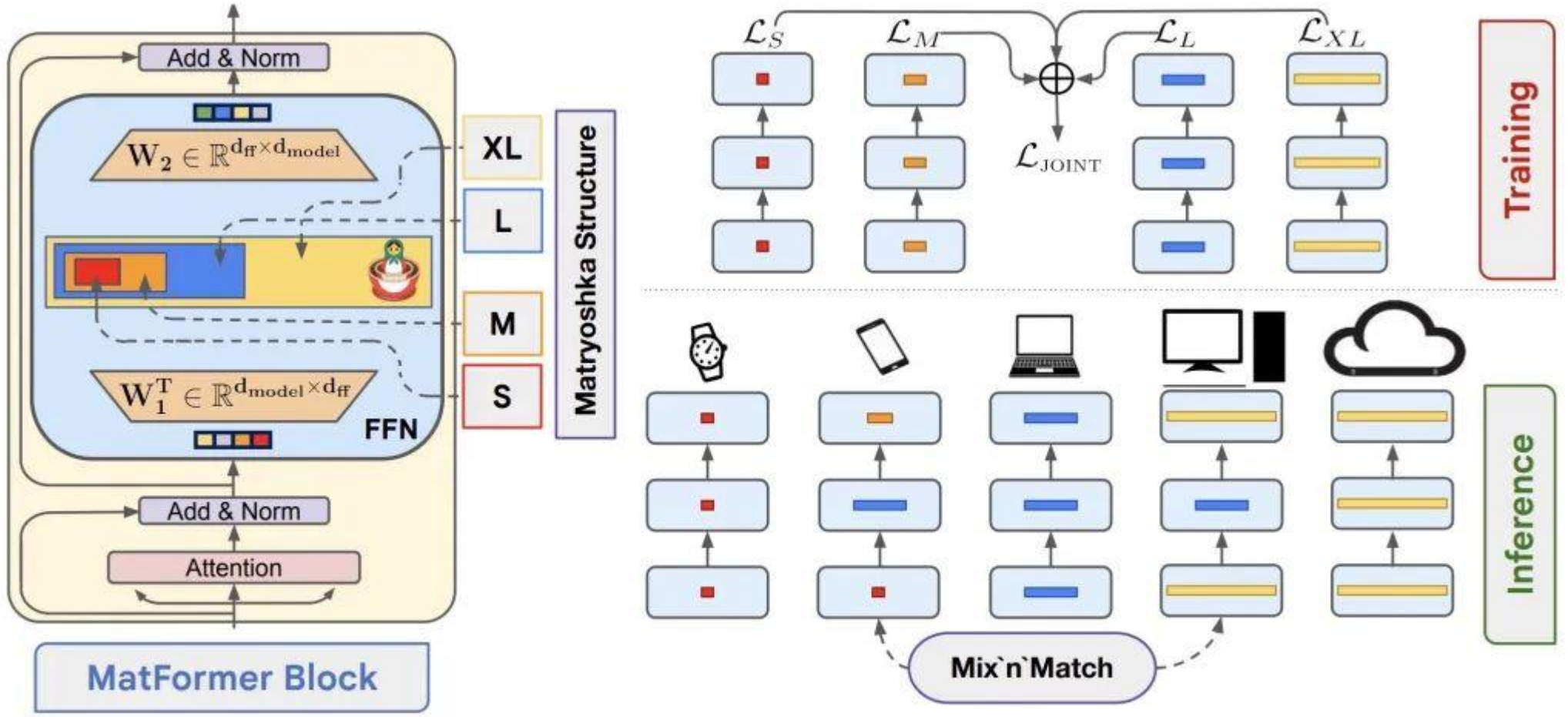
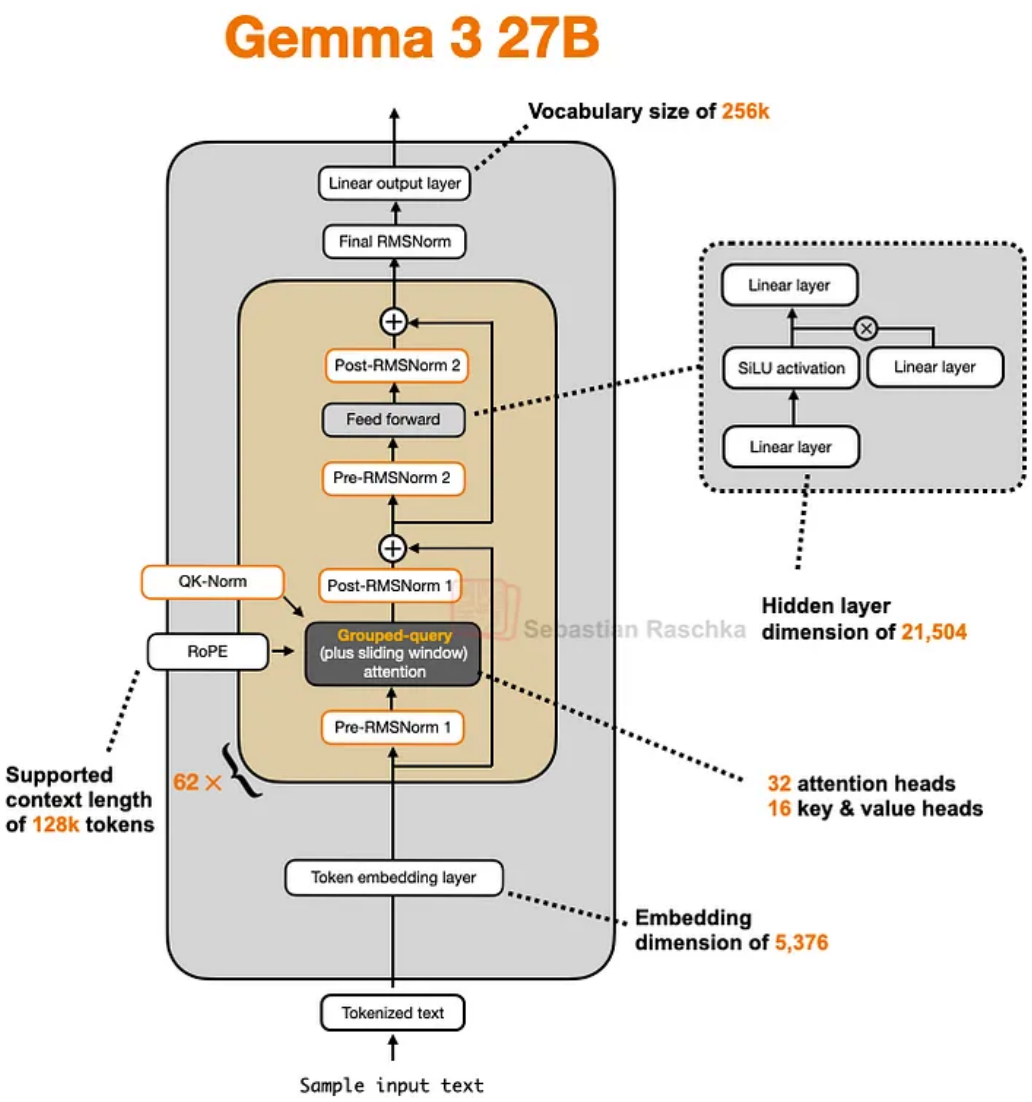
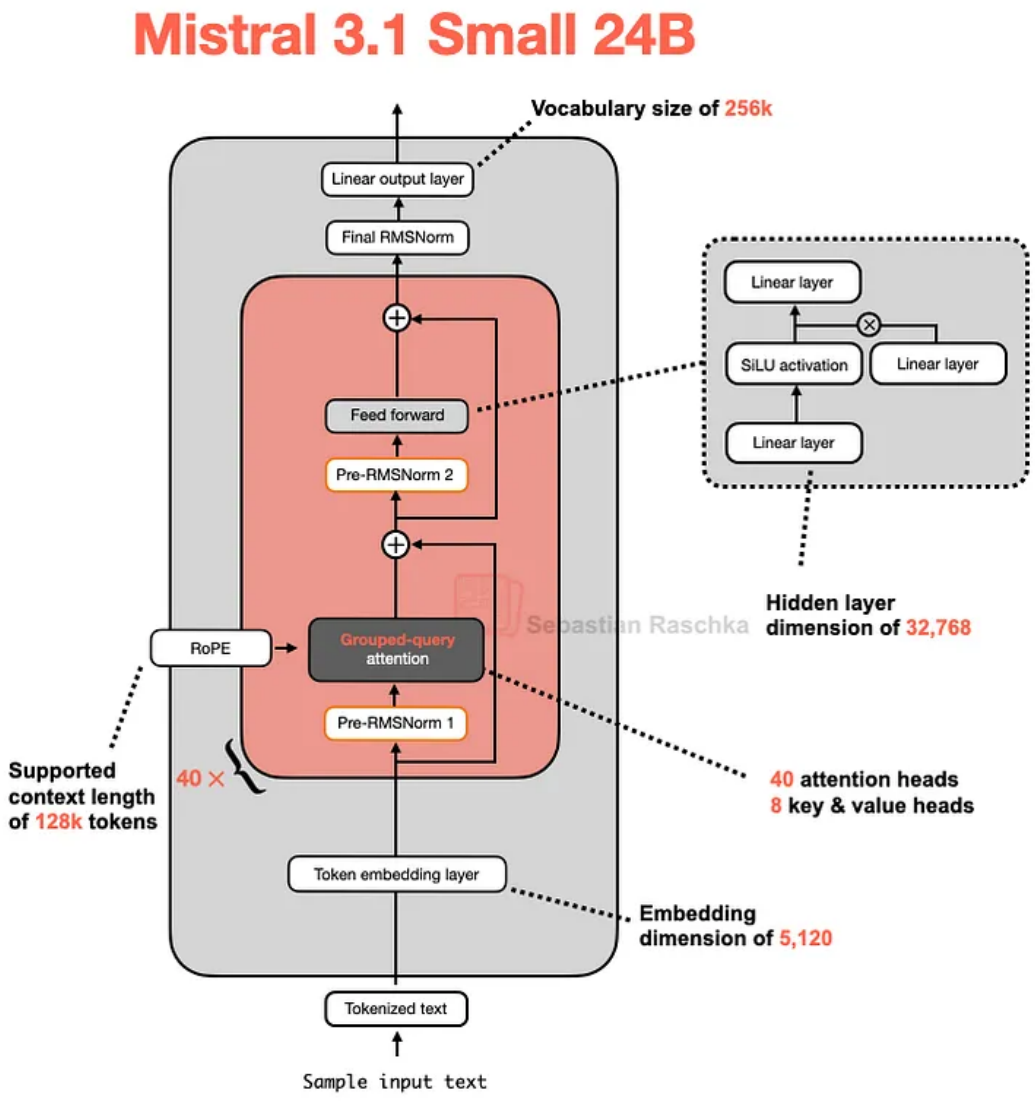


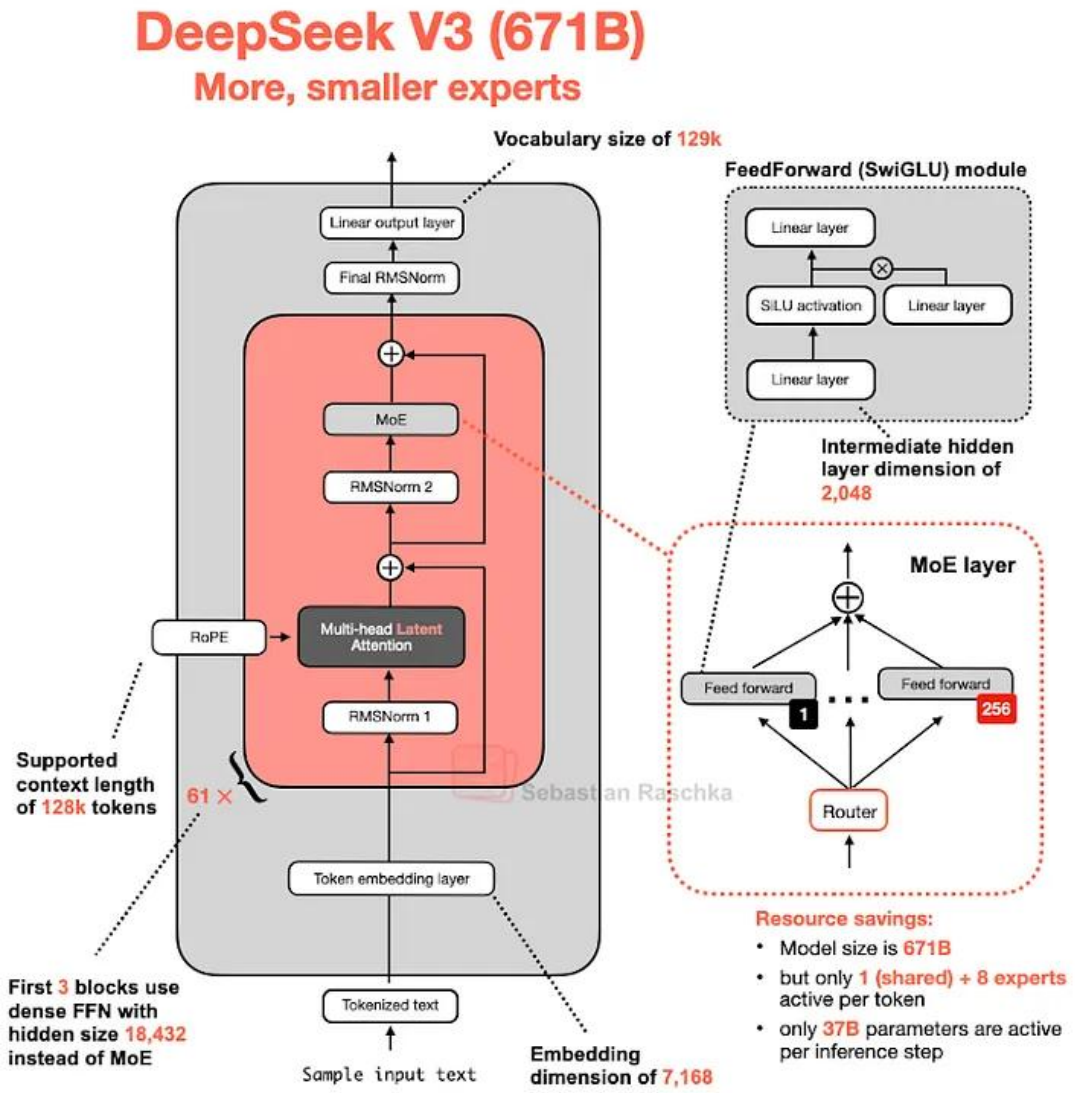
Figure 1: MatFormer introduces nested structure into the Transformer's FFN block & jointly trains all the submodels, enabling free extraction of hundreds of accurate submodels for elastic inference.

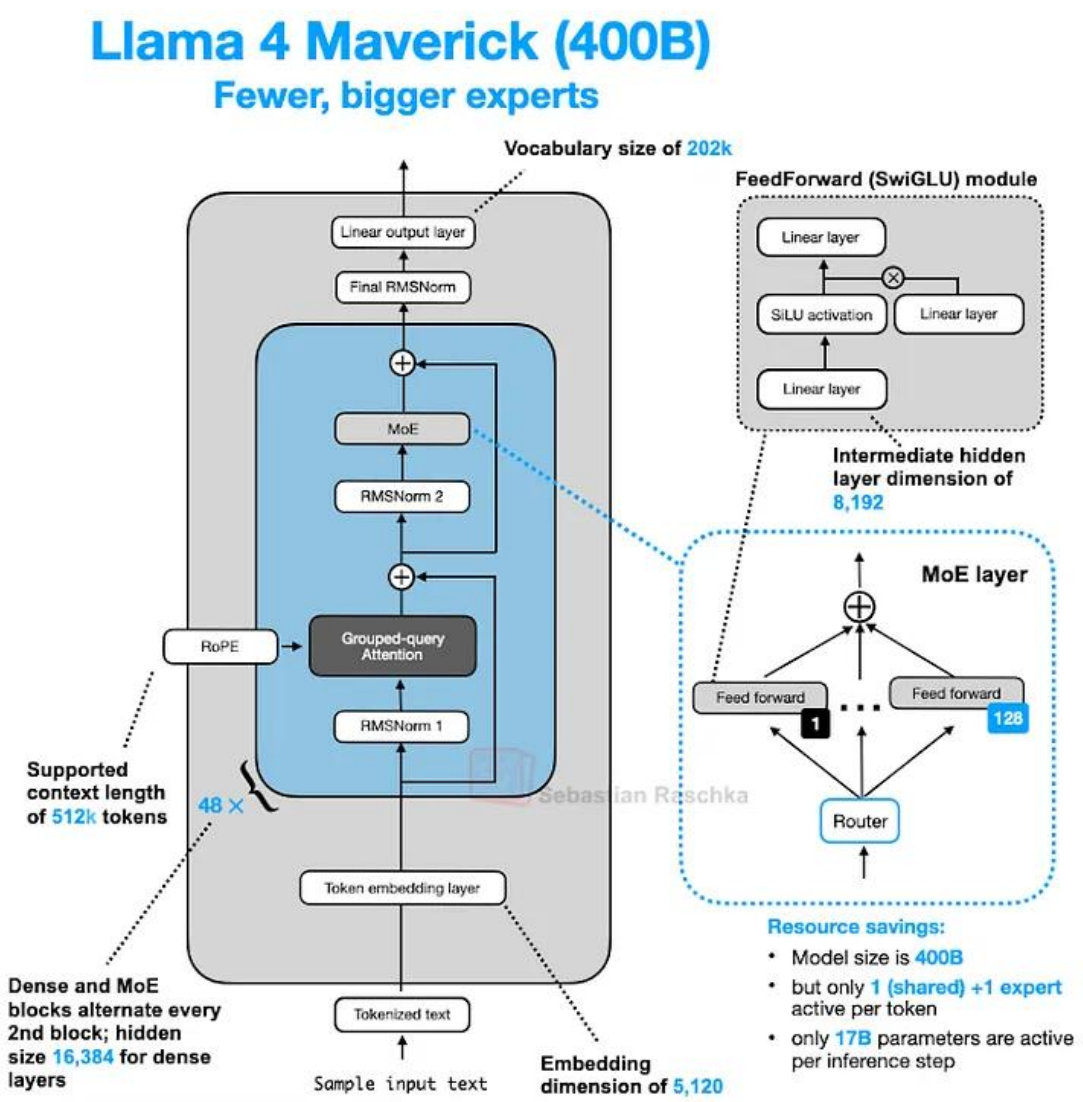
Mistral Small 3.1



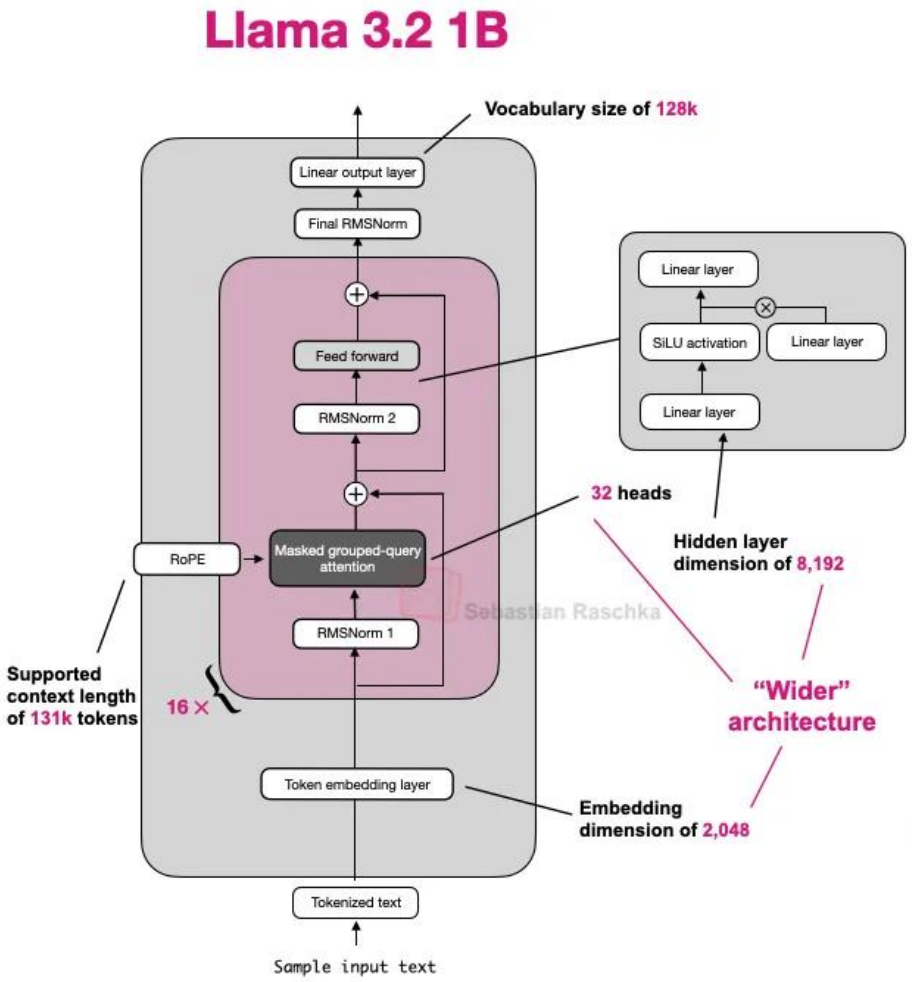


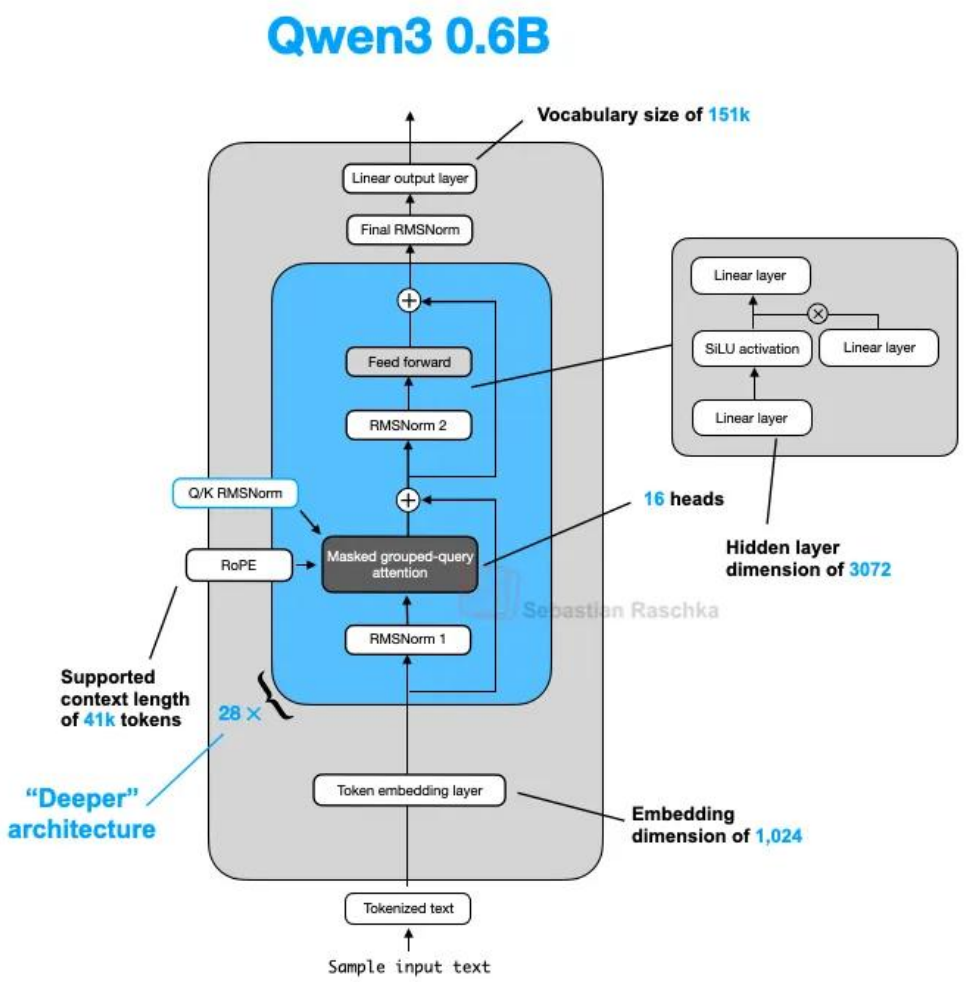
Llama4: MOE


DeepSeek V3 (671B)
More, smaller experts
Vocabulary size of 129k
Linear output layer
Final RMSNorm
MoE
RMSNorm 2
RoPE
Multi-head Latent Attention
RMSNorm 1
Token embedding layer
Tokenized text
Sample input text
61 ×
Supported context length of 128k tokens
First 3 blocks use dense FFN with hidden size 18,432 instead of MoE
Embedding dimension of 7,168
FeedForward (SwiGLU) module
Linear layer
SiLU activation
Linear layer
Linear layer
Intermediate hidden layer dimension of 2,048
MoE layer
Feed forward
1
Feed forward
256
Router
Resource savings:
• Model size is 671B
• but only 1 (shared) + 8 experts active per token
• only 37B parameters are active per inference step
Sebastian Raschka


Llama 4 Maverick (400B)
Fewer, bigger experts
Vocabulary size of 202k
Linear output layer
Final RMSNorm
MoE
RMSNorm 2
RoPE
Grouped-query Attention
RMSNorm 1
Token embedding layer
Tokenized text
Sample input text
48 ×
Supported context length of 512k tokens
Dense and MoE blocks alternate every 2nd block; hidden size 16,384 for dense layers
Embedding dimension of 5,120
FeedForward (SwiGLU) module
Linear layer
SiLU activation
Linear layer
Linear layer
Intermediate hidden layer dimension of 8,192
MoE layer
Feed forward
1
Feed forward
128
Router
Resource savings:
• Model size is 400B
• but only 1 (shared) +1 expert active per token
• only 17B parameters are active per inference step
Sebastian Raschka

Qwen3 (Dense)

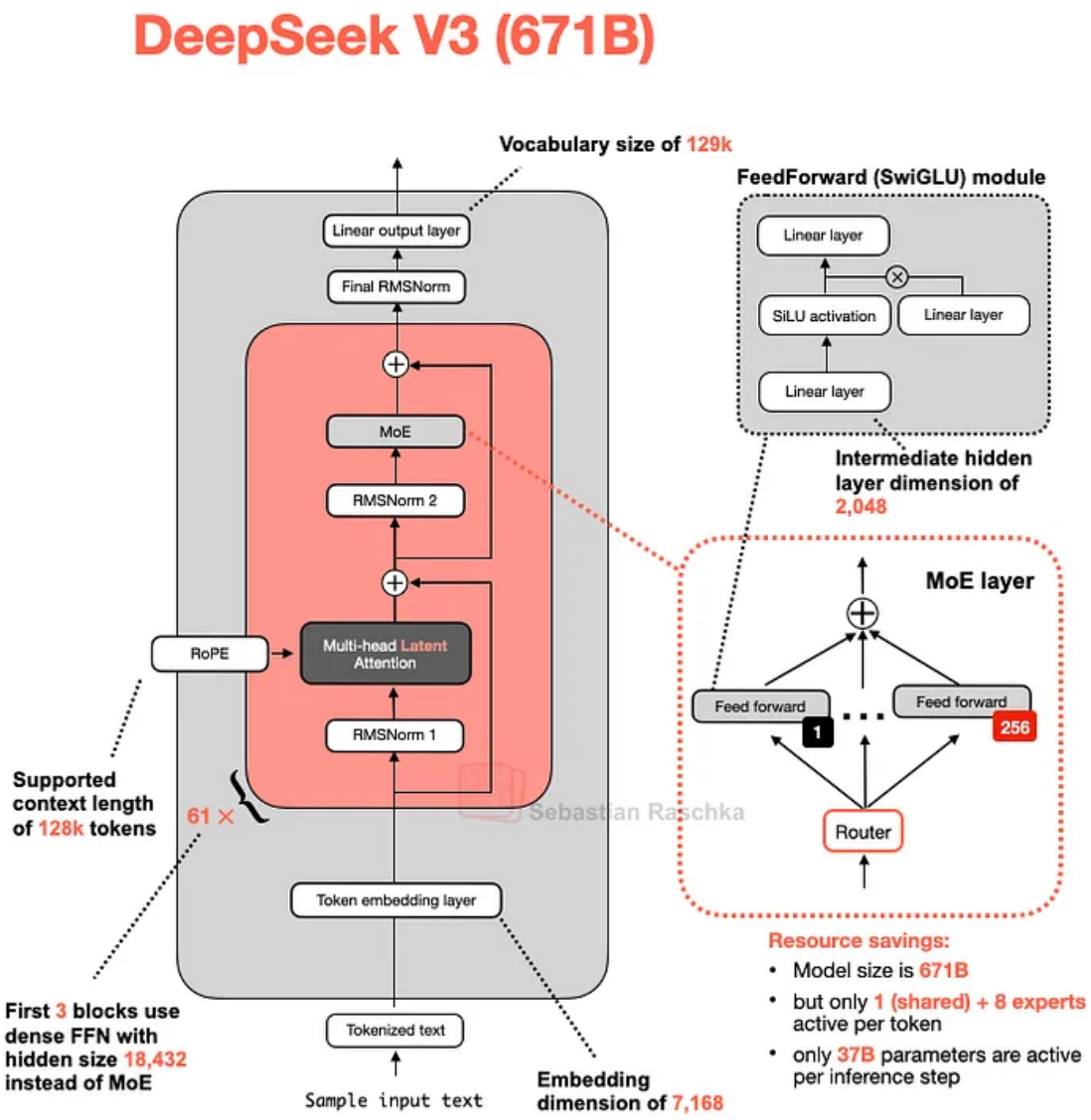


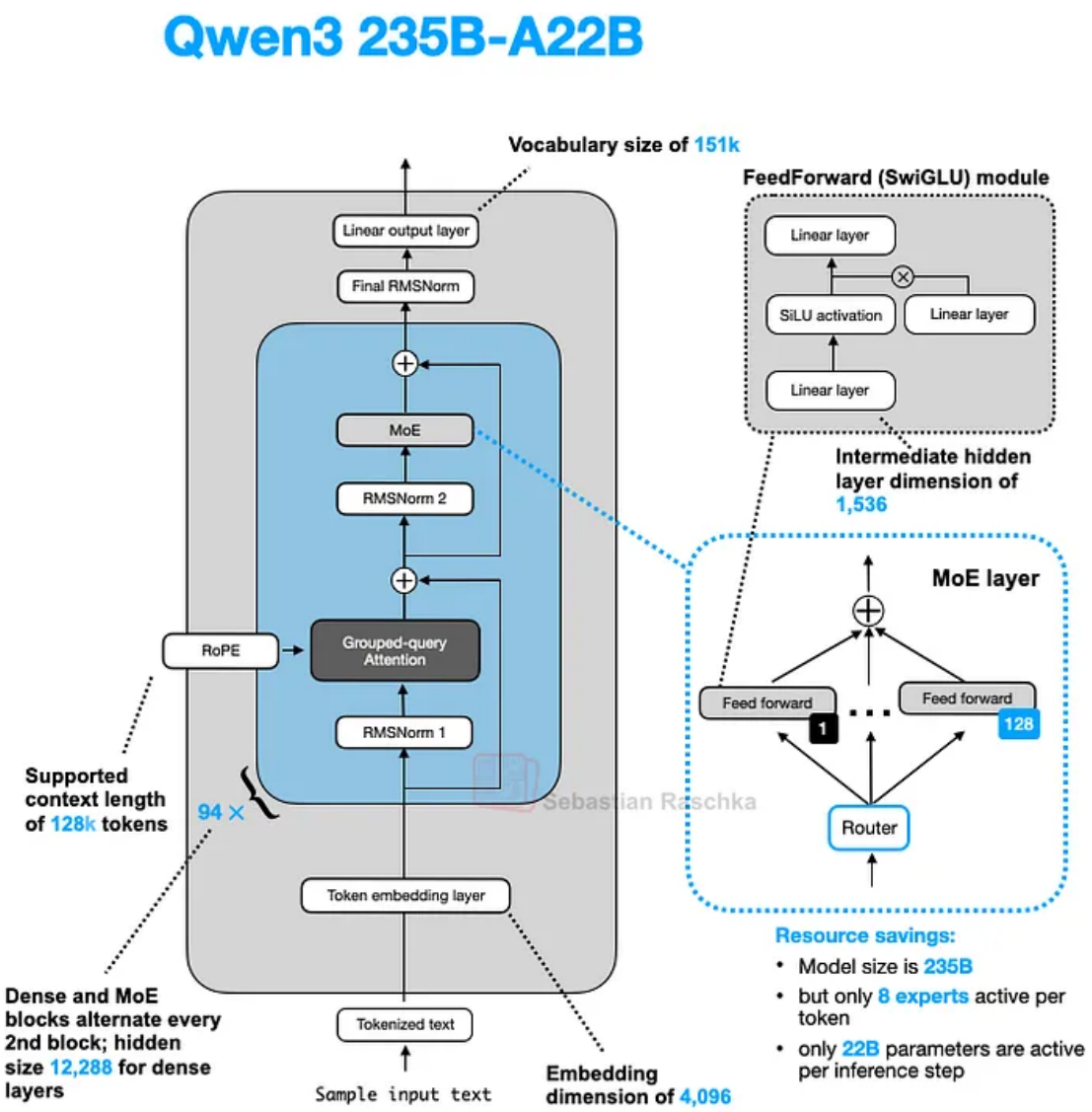


	Tokens/sec	Memory
Llama3Model 1B	42	2.91 GB
Llama3Model 1B compiled	170	3.12 GB

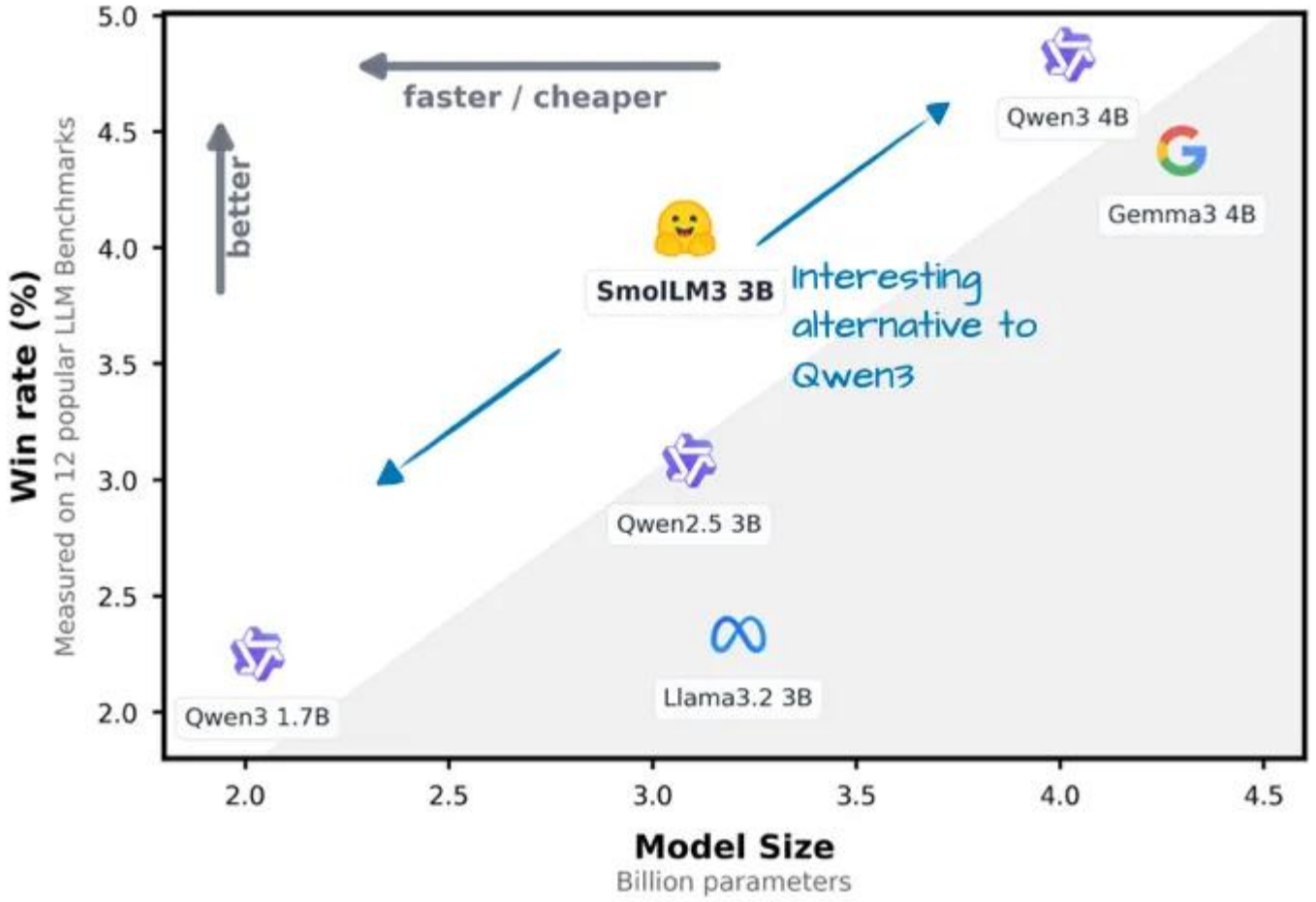
	Tokens/sec	Memory
Qwen3Model 0.6B	24	1.49 GB
Qwen3Model 0.6B compiled	101	1.99 GB

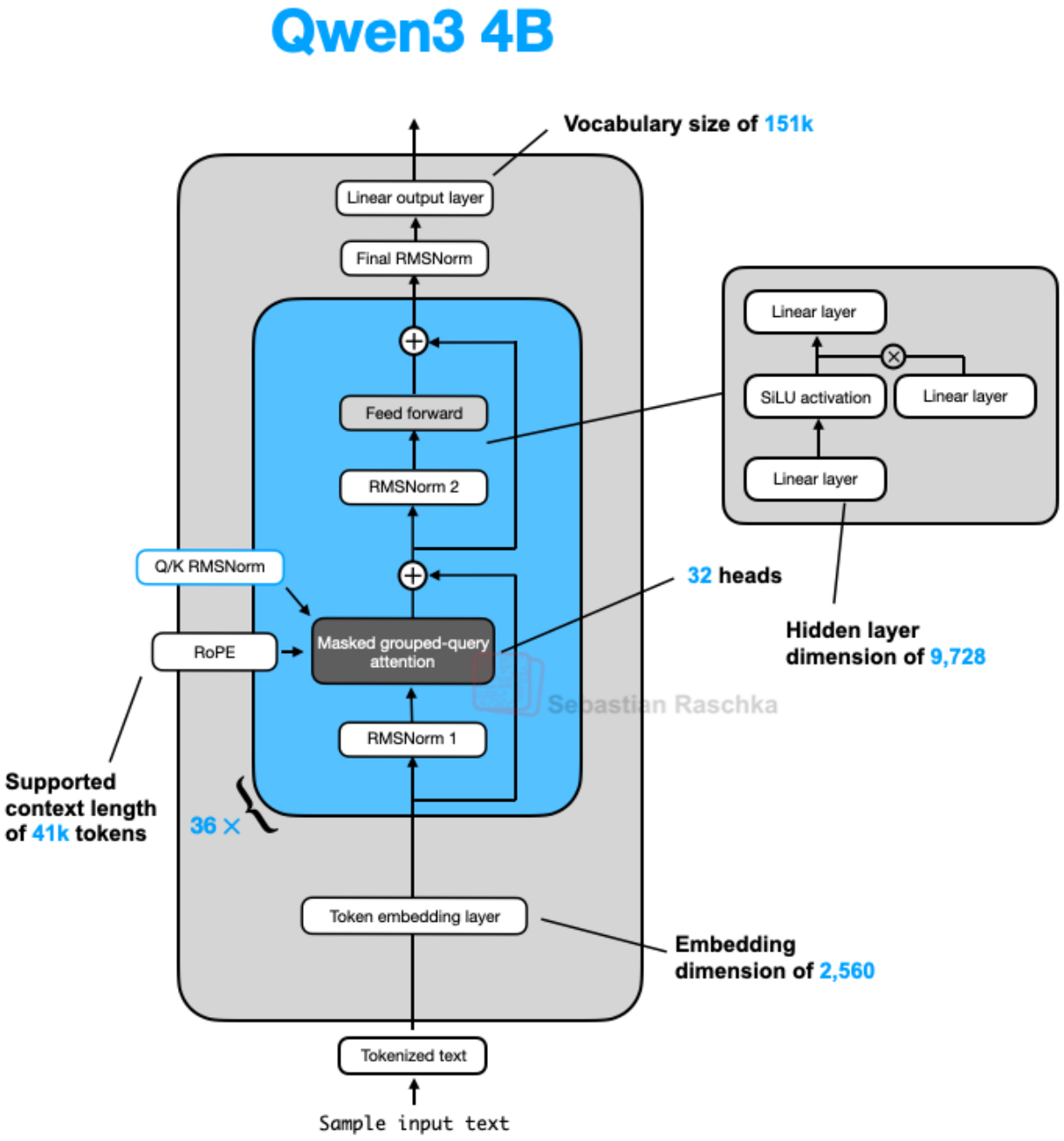
Qwen3 (MoE)

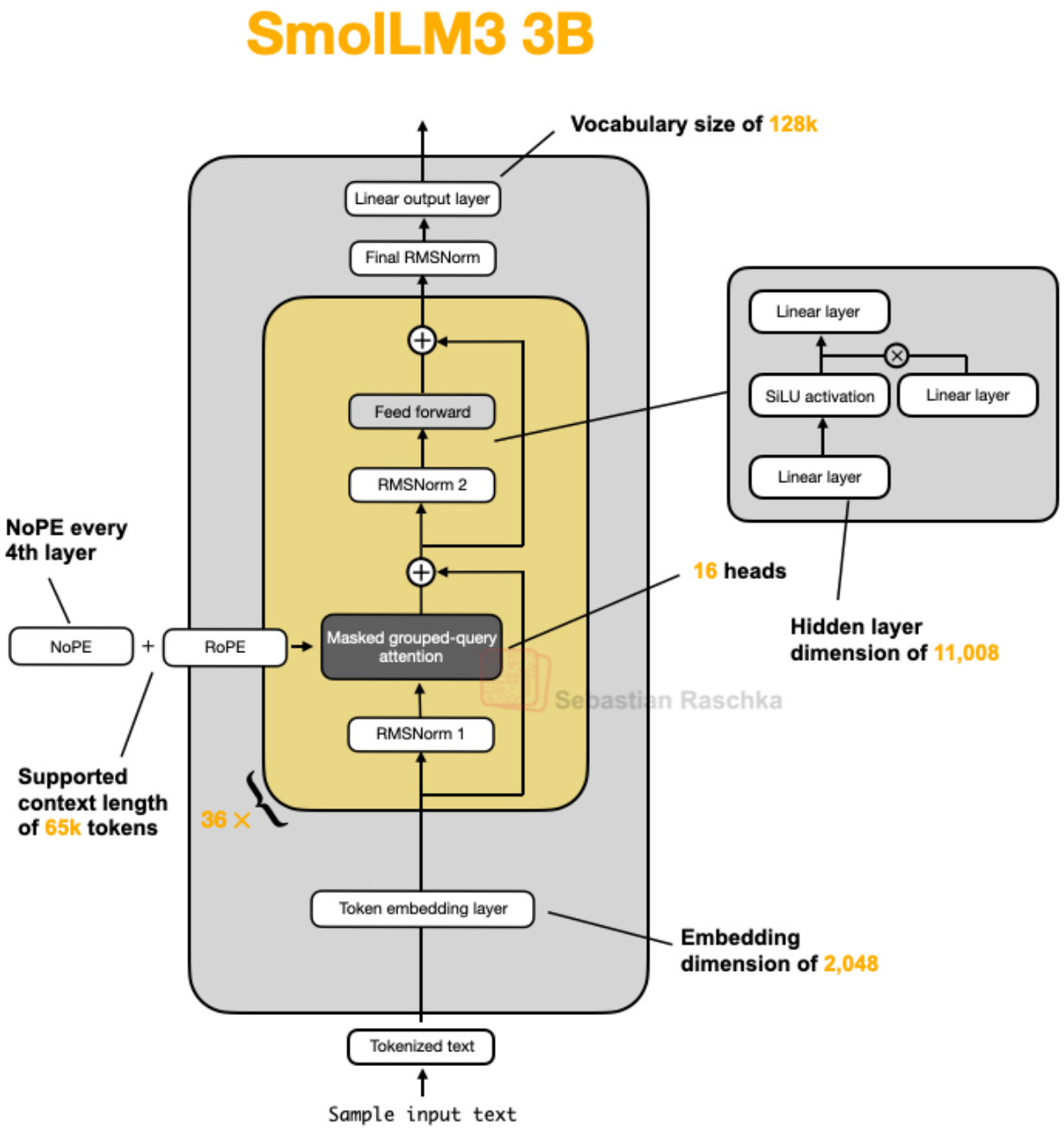

DeepSeek V3 (671B)
Vocabulary size of 129k
Linear output layer
Final RMSNorm
MoE
RMSNorm 2
RoPE
Multi-head Latent Attention
RMSNorm 1
Token embedding layer
Tokenized text
Sample input text
61 ×
Supported context length of 128k tokens
First 3 blocks use dense FFN with hidden size 18,432 instead of MoE
Embedding dimension of 7,168
FeedForward (SwiGLU) module
Linear layer
SiLU activation
Linear layer
Linear layer
Intermediate hidden layer dimension of 2,048
MoE layer
Feed forward
1
Feed forward
256
Router
Resource savings:
Model size is 671B
but only 1 (shared) + 8 experts active per token
only 37B parameters are active per inference step
Sebastian Raschka


Qwen3 235B-A22B
Vocabulary size of 151k
Linear output layer
Final RMSNorm
MoE
RMSNorm 2
RoPE
Grouped-query Attention
RMSNorm 1
Token embedding layer
Tokenized text
Sample input text
94 ×
Supported context length of 128k tokens
Dense and MoE blocks alternate every 2nd block; hidden size 12,288 for dense layers
Embedding dimension of 4,096
FeedForward (SwiGLU) module
Linear layer
SiLU activation
Linear layer
Linear layer
Intermediate hidden layer dimension of 1,536
MoE layer
Feed forward
1
Feed forward
128
Router
Resource savings:
Model size is 235B
but only 8 experts active per token
only 22B parameters are active per inference step
Sebastian Raschka

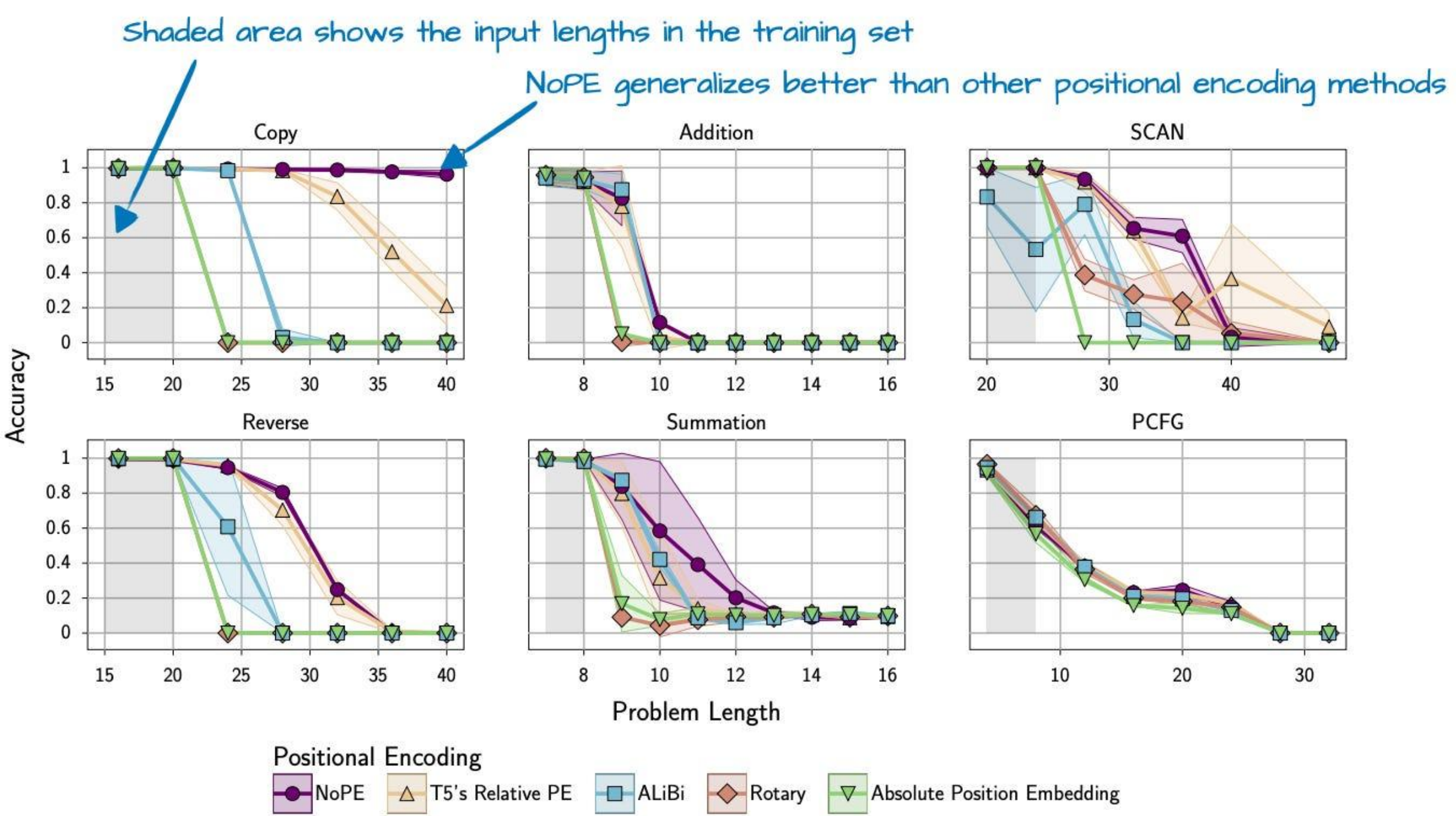
SmolLM3


faster / cheaper
better
Win rate (%)
Measured on 12 popular LLM Benchmarks
Model Size
Billion parameters
SmolLM3 3B
Interesting alternative to Qwen3
Qwen3 4B
Gemma3 4B
Qwen2.5 3B
Llama3.2 3B
Qwen3 1.7B
5.0
4.5
4.0
3.5
3.0
2.5
2.0
2.0
2.5
3.0
3.5
4.0
4.5

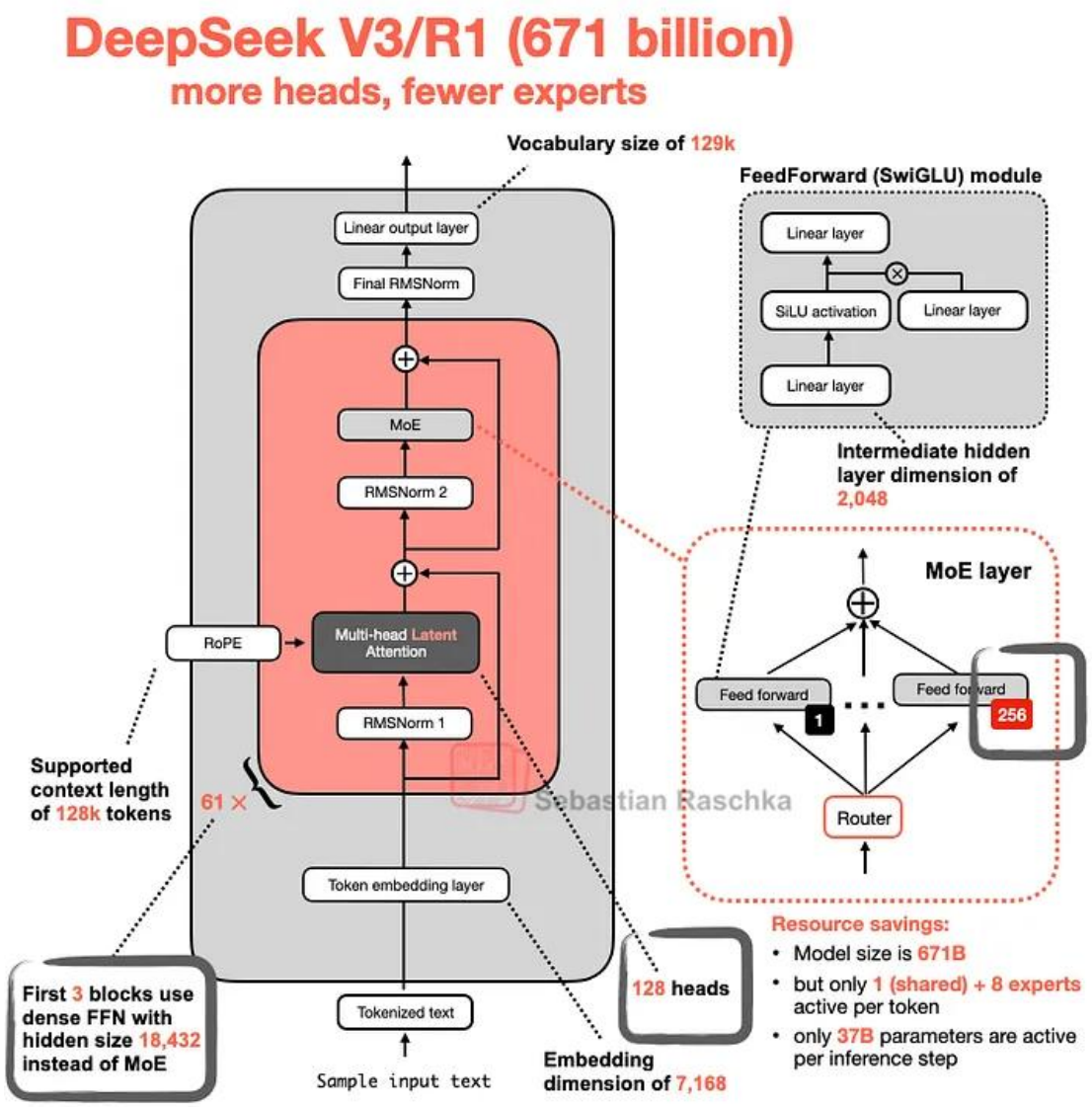


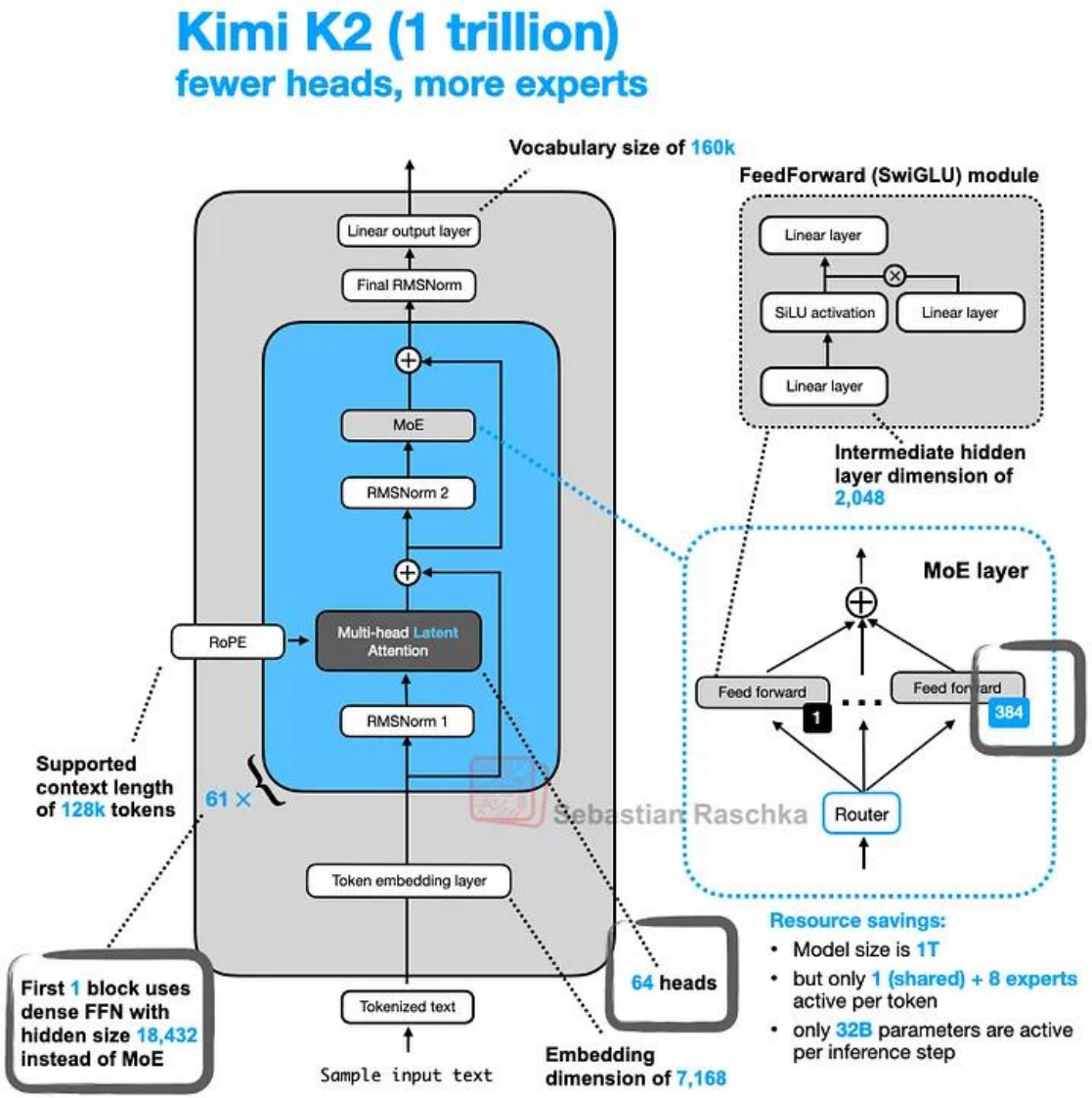


NoPE (No Positional Embedding

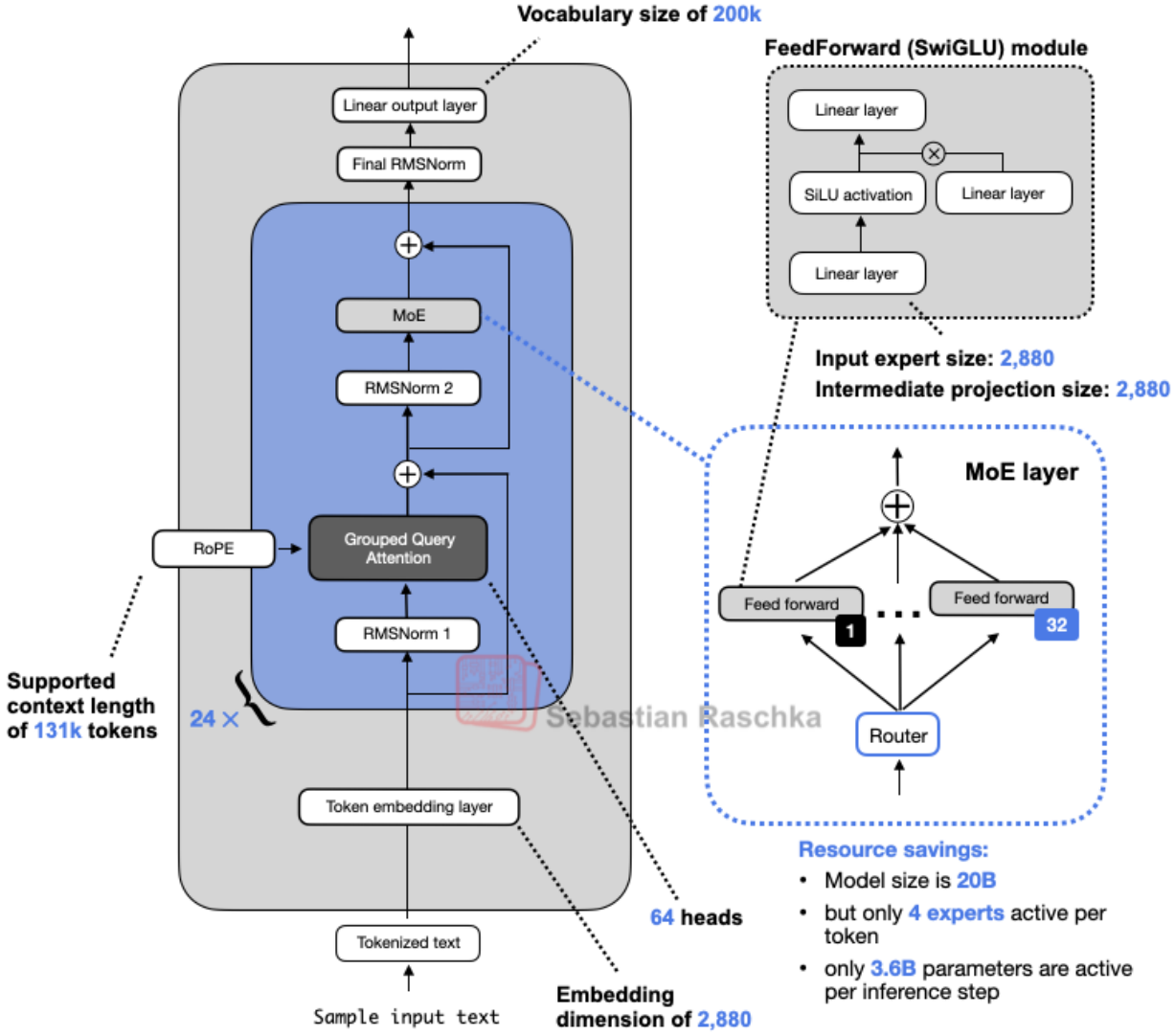

Shaded area shows the input lengths in the training set
NoPE generalizes better than other positional encoding methods
Copy
Addition
SCAN
Reverse
Summation
PCFG
Accuracy
Problem Length
Positional Encoding
NoPE
T5's Relative PE
ALiBi
Rotary
Absolute Position Embedding

Kimi


DeepSeek V3/R1 (671 billion)
more heads, fewer experts
Vocabulary size of 129k
Linear output layer
Final RMSNorm
MoE
RMSNorm 2
RMSNorm 1
RoPE
Multi-head Latent Attention
Token embedding layer
Tokenized text
Sample input text
61 ×
Supported context length of 128k tokens
First 3 blocks use dense FFN with hidden size 18,432 instead of MoE
128 heads
Embedding dimension of 7,168
FeedForward (SwiGLU) module
Linear layer
SiLU activation
Linear layer
Linear layer
Intermediate hidden layer dimension of 2,048
MoE layer
Feed forward
1
256
Router
Resource savings:
• Model size is 671B
• but only 1 (shared) + 8 experts active per token
• only 37B parameters are active per inference step
Sebastian Raschka


Kimi K2 (1 trillion)
fewer heads, more experts
Vocabulary size of 160k
Linear output layer
Final RMSNorm
MoE
RMSNorm 2
RMSNorm 1
RoPE
Multi-head Latent Attention
Token embedding layer
Tokenized text
Sample input text
61 ×
Supported context length of 128k tokens
First 1 block uses dense FFN with hidden size 18,432 instead of MoE
64 heads
Embedding dimension of 7,168
FeedForward (SwiGLU) module
Linear layer
SiLU activation
Linear layer
Linear layer
Intermediate hidden layer dimension of 2,048
MoE layer
Feed forward
1
384
Router
Resource savings:
• Model size is 1T
• but only 1 (shared) + 8 experts active per token
• only 32B parameters are active per inference step
Sebastian Raschka

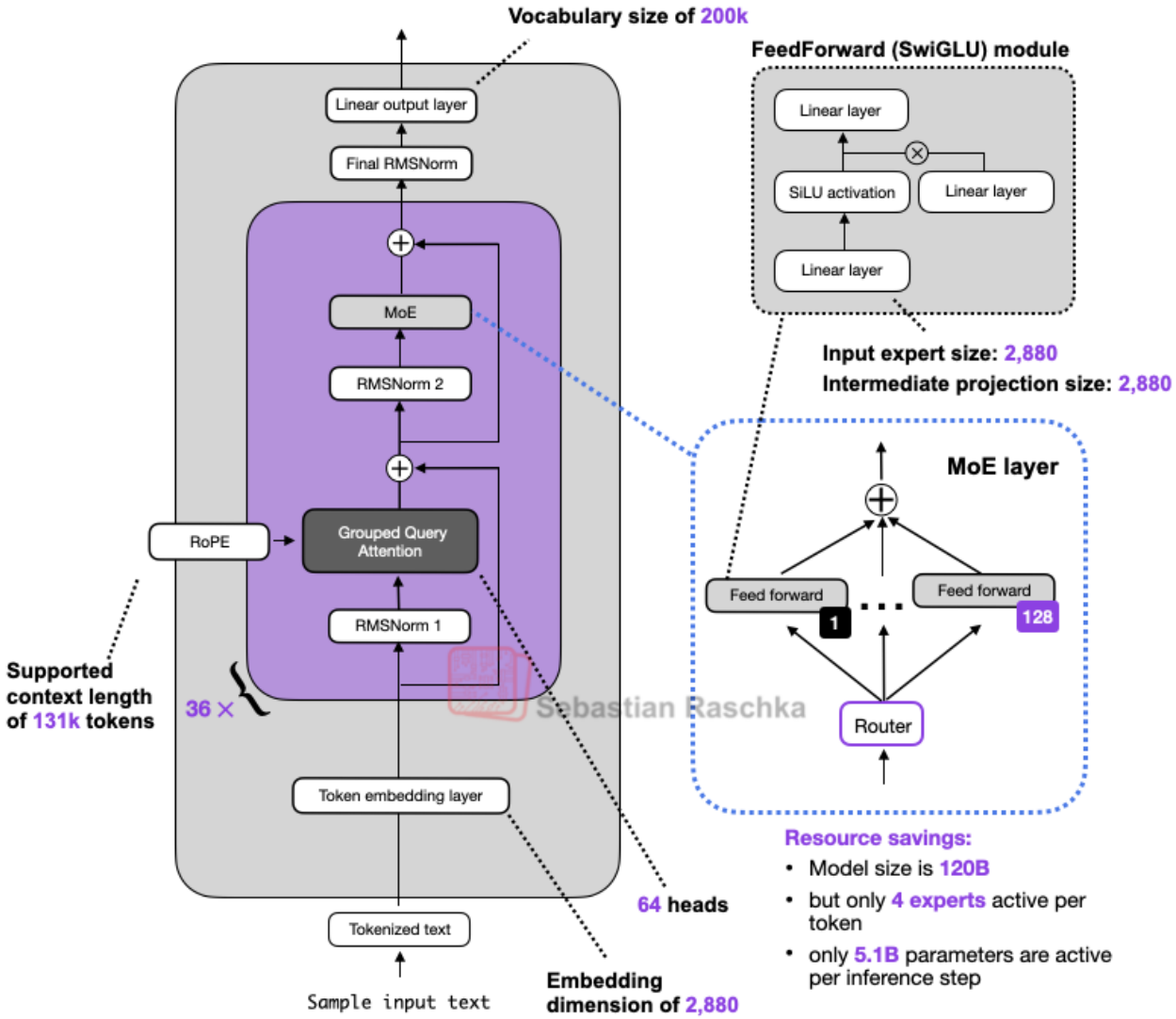
GPT-OSS 20B



Resource savings:

- Model size is **20B**
- but only **4 experts** active per token
- only **3.6B** parameters are active per inference step

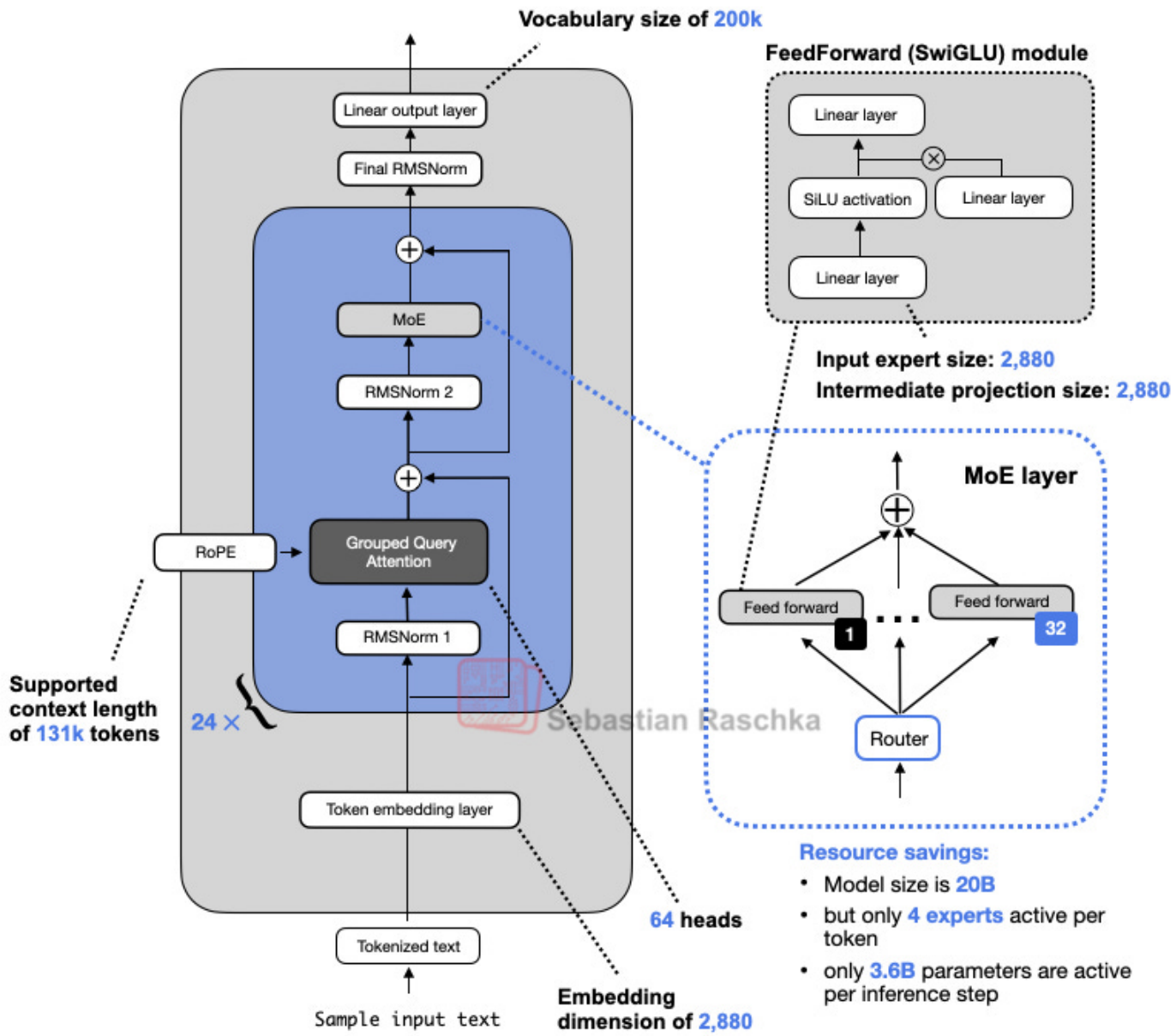
GPT-OSS 120B



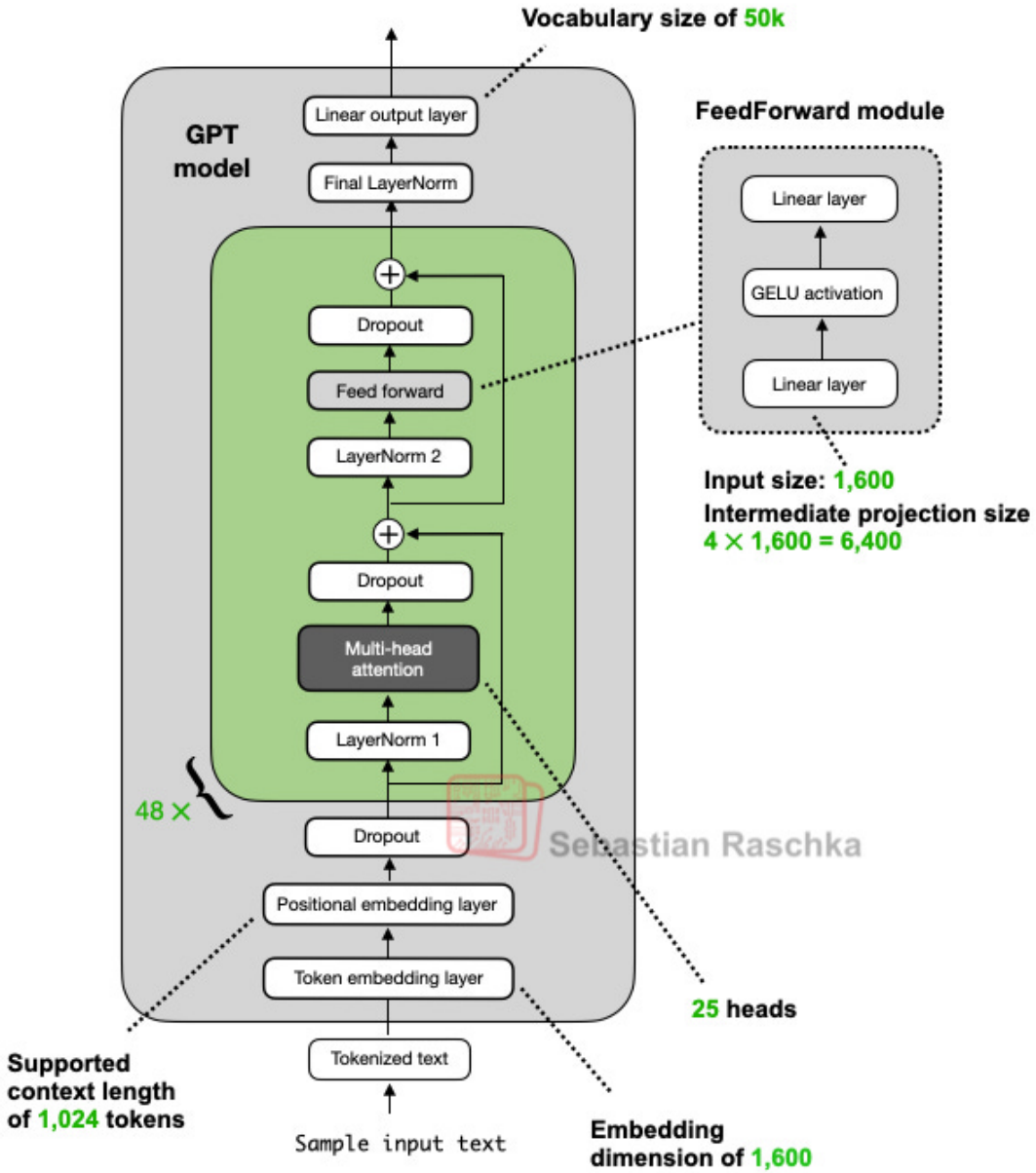
Resource savings:

- Model size is **120B**
- but only **4 experts** active per token
- only **5.1B** parameters are active per inference step

GPT-OSS 20B (2025)

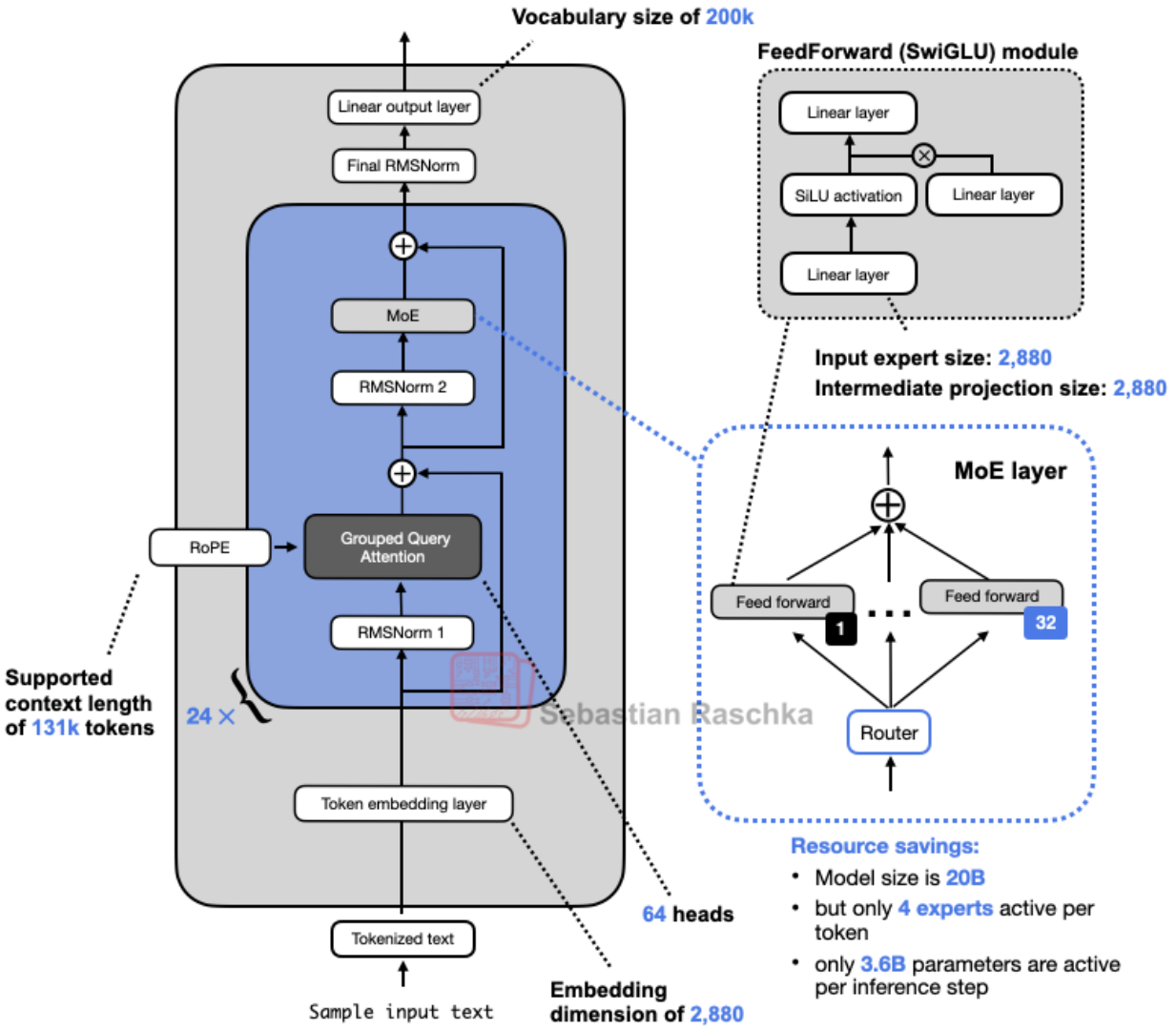


GPT-2 XL 1.5B (2019)



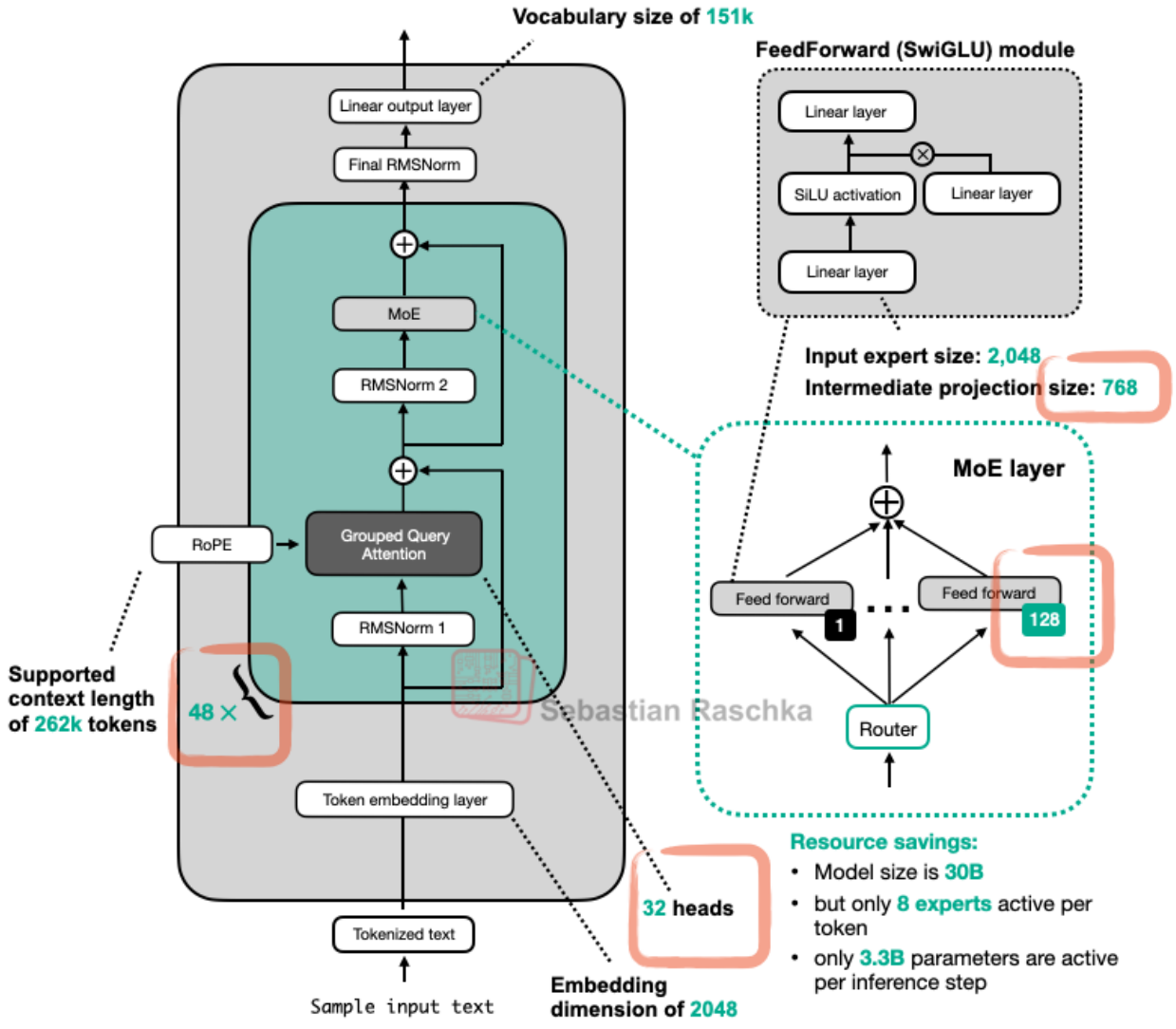
GPT-OSS 20B

Wider, fewer & bigger experts,



Qwen3 30B-A3B

Deeper, more & smaller experts



Reference

- https://arxiv.org/abs/1810.04805
  - BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding