EE7207: Lecture 13

Advanced Topics and Discussions

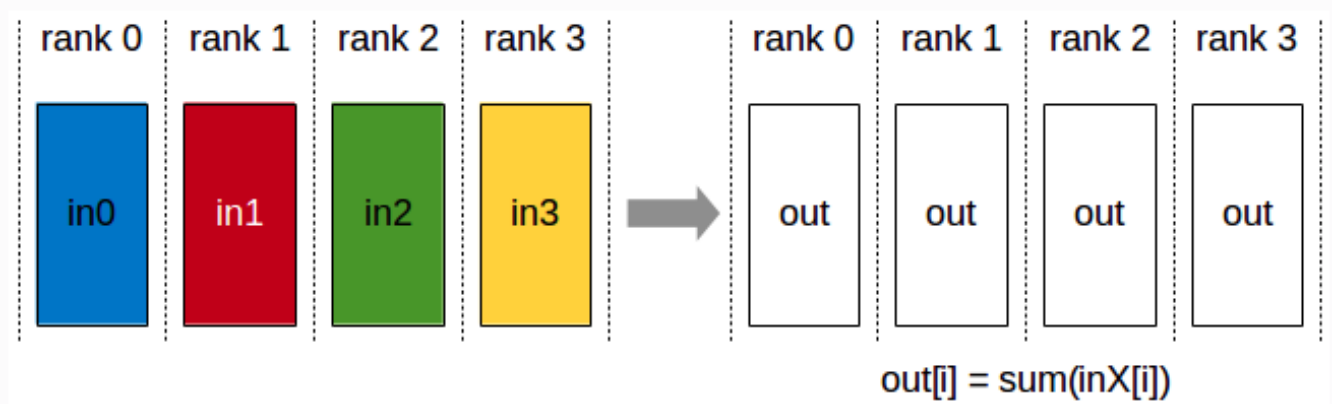
Lecturer: Zhang Handuo
handuo.zhang@shanda.com

Outline

- 1 Basics in MLSys
  - Parallelization
  - Attention Optimization Techniques
- 2 Vision-Language Model
- 3 Open-set Computer Vision

Inefficiency of Parameter Training

- Centralized Communication
  - All workers communicate with parameter servers for weights update
  - Cannot scale to large numbers of workers
- How can we decentralize communication in DNN training?
  - Scattering / Sharding + Aggregation
- Allreduce: perform element-wise reduction across multiple devices
  - The *AllReduce* operation performs reductions on data (for example, sum, min, max) across devices and stores the result in the receive buffer of every rank.



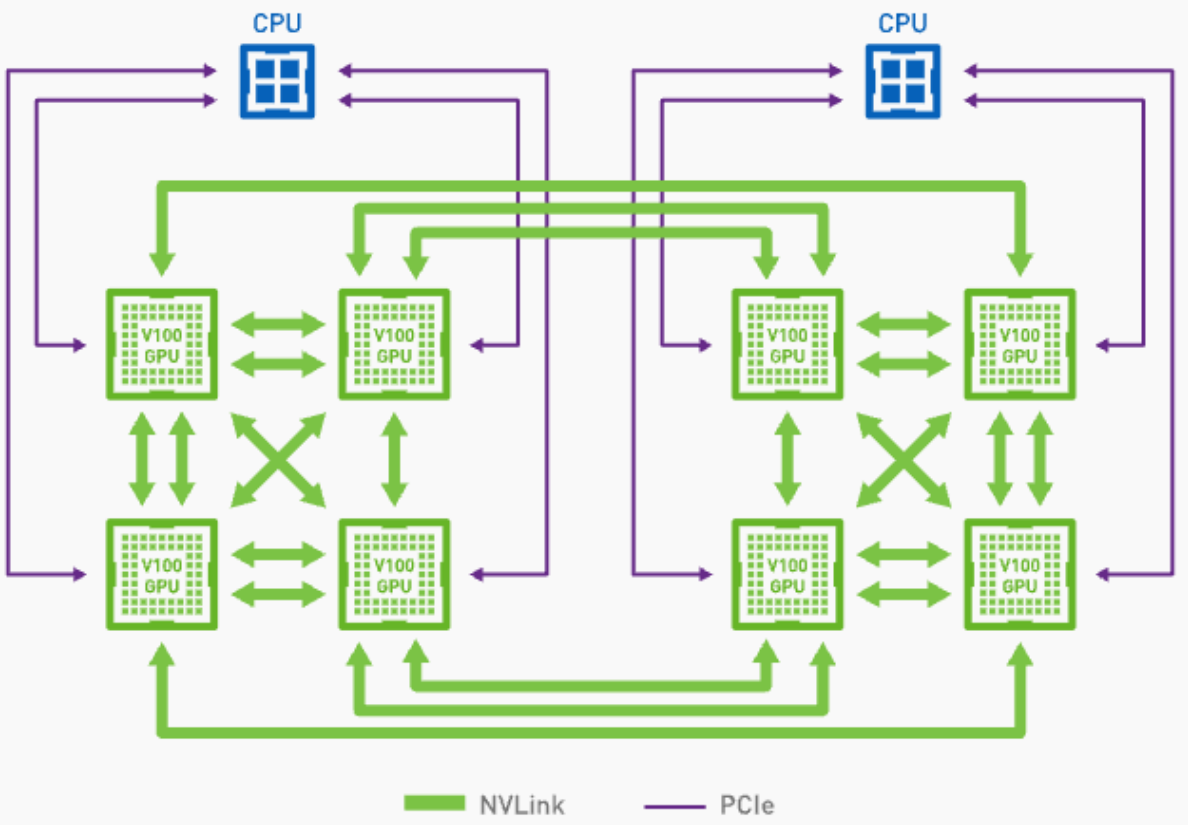
All-Reduce operation: each rank receives the reduction of input values across ranks.

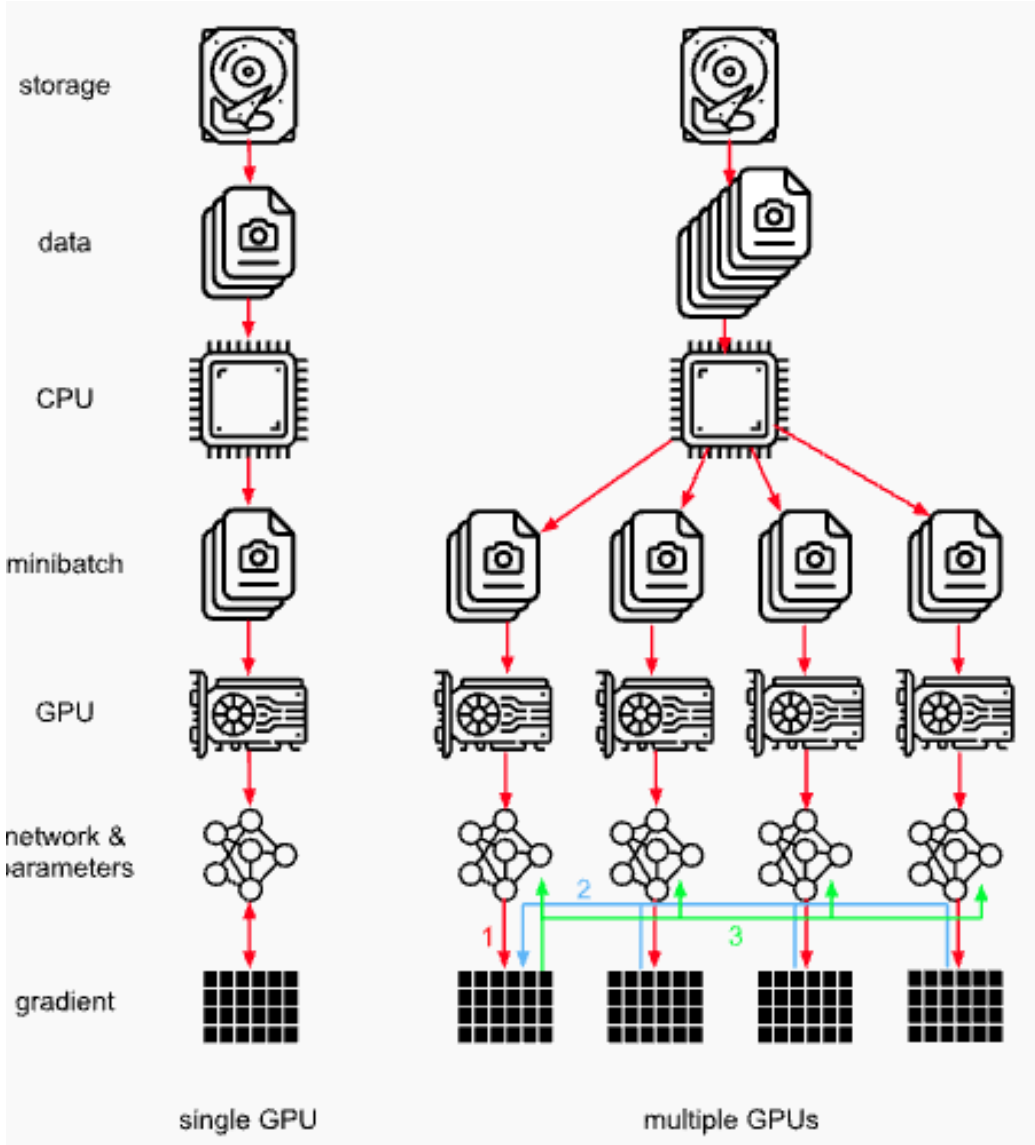
Collective Operations

Operation	Description	Result Location
Reduce	Combines data from all processes using operator (sum, min) and returns a single result	Single designated (root) process
Allreduce	Similar to Reduce, but after combining data, distributes the result to every process.	All processes
Broadcast	Distributes data from a root process to all other processes	All processes
Gather	Collect data from every process and assembles it at a root process: [1,2] + [3,4] + [5,6] -> [1,2,3,4,5,6]	Single designated (root) process
Allgather	Gathers data from all processes and distributes the complete data back to every process.	All processes
Scatter	Divides data from a root process into segments and sends each segment to different processes (opposite to gather)	Each process receives a unique segment
Reduce Scatter	Performs a reduction on data across processes and scatters parts of the reduced result to each process.	Each process receives a portion
Alltoall	Each process sends distinct data segments to every other process; every process ends up with data from all others.	All processes

1 Data Parallelism

- Ring Allreduce

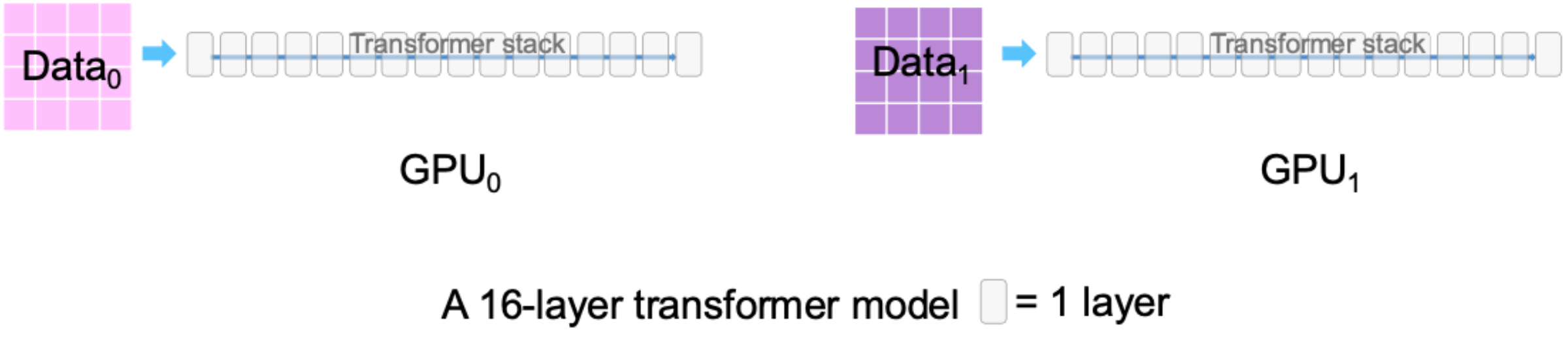




1 Data Parallelism

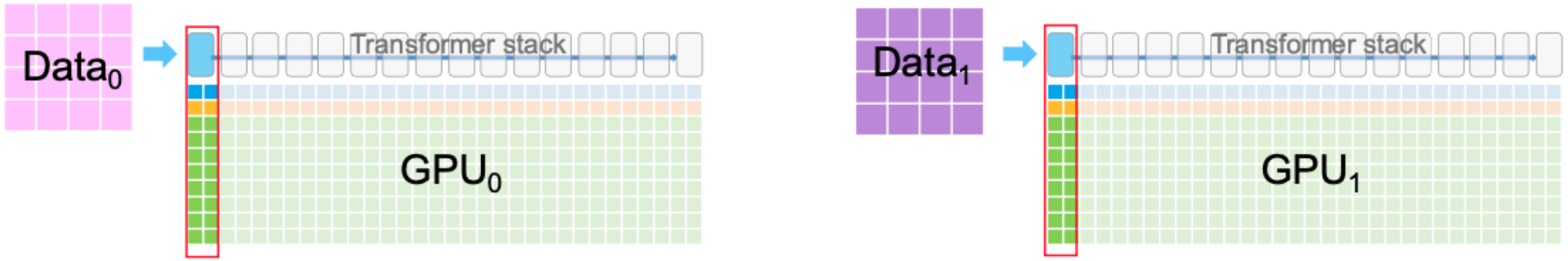
- Understanding Memory Consumption

The same setup is replicated multiple times, and each being fed a slice of the data. The processing is done in parallel and all setups are synchronized at the end of each training step



1 Data Parallelism

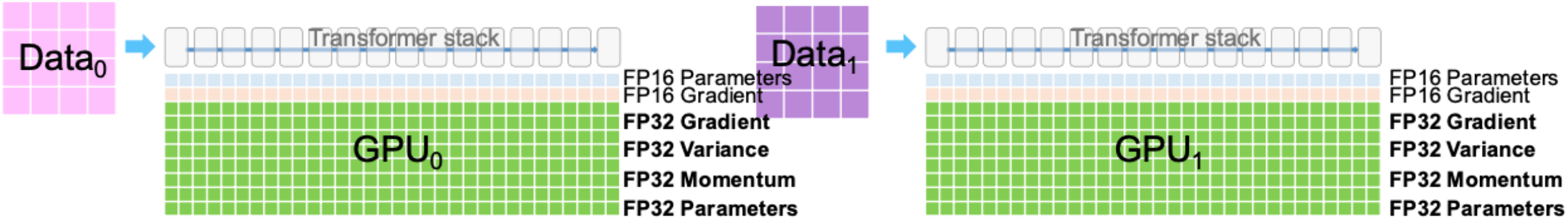
- Understanding Memory Consumption



Each cell represents GPU memory used by its corresponding transformer layer

1 Data Parallelism

- Understanding Memory Consumption

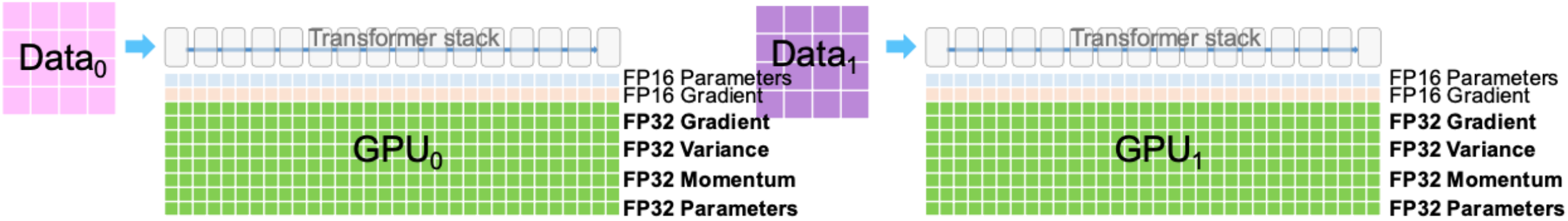


- FP16 Parameter
- FP16 Gradients
- FP32 Optimizer States
  - Gradients, Variance, Momentum, Parameters

1 Data Parallelism

- Understanding Memory Consumption

Take a 1B parameter model as example -> 20GB GPU



- FP16 Parameter: 2M bytes
- FP16 Gradients: 2M bytes
- FP32 Optimizer States: 16M bytes
  - Gradients, Variance, Momentum, Parameters

M=Number of parameters in the model

1 Data Parallelism

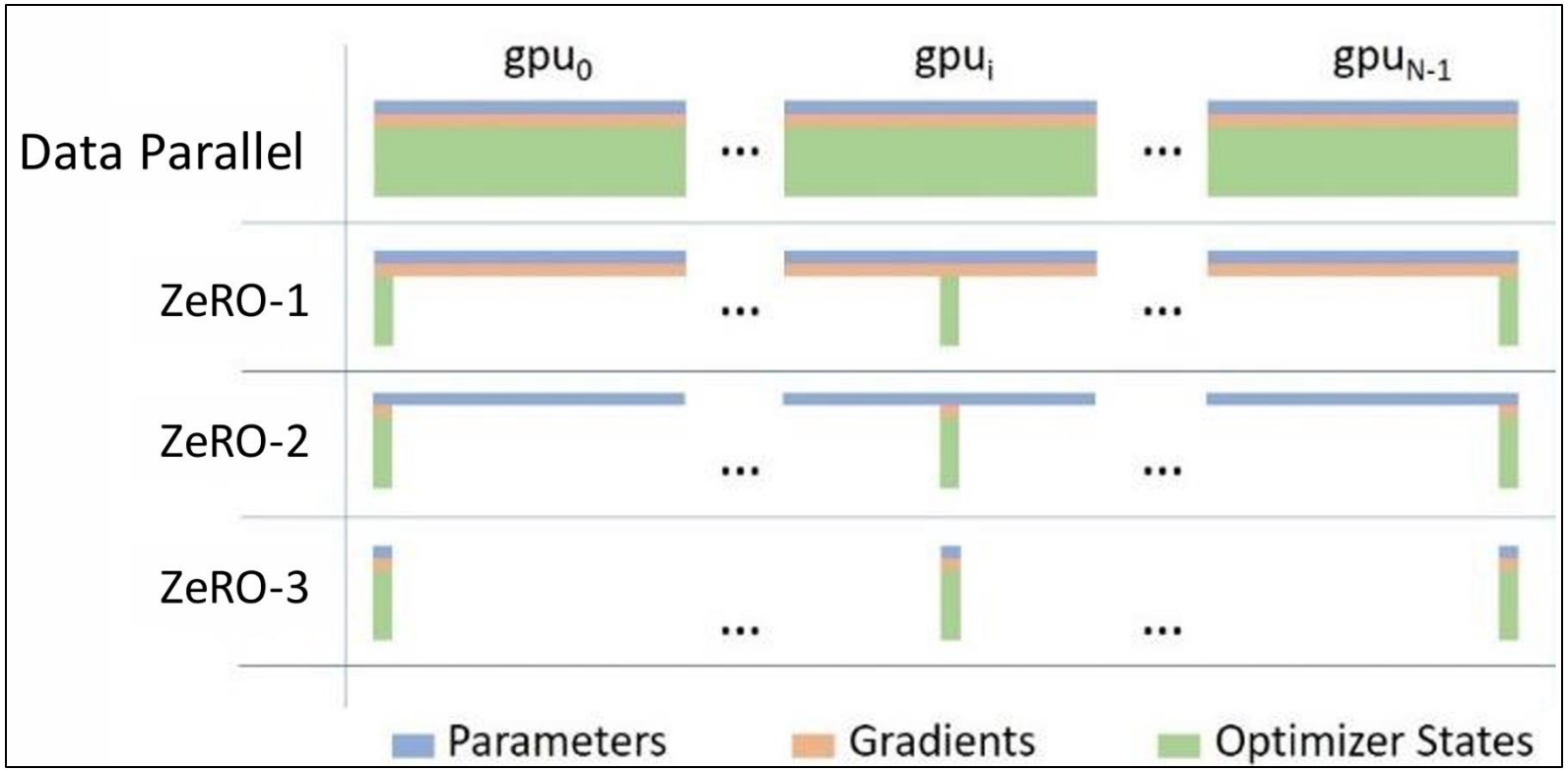
Basic Concepts

- Node (server)
  - A **node** is a single physical machine (server) in an HPC cluster.
  - Each node typically has multiple **GPUs**, CPUs, memory, etc.
  - In **multi-node** training, we use multiple nodes connected via a high-speed interconnect (like InfiniBand).
- World / Global View
  - The collection of **all processes**
  - Each process could be handling a GPU in a DDP setup
  - The **world size** is the total number of processes across all nodes (like 2 nodes * 4 GPUs = 8)
- Rank (Global Rank)
  - The **rank** is the **unique ID** of a process within the whole world, ranging from 0 to *world_size – 1*.
  - It's used to identify processes and determine their roles (e.g., rank 0 often handles logging or saving checkpoints).
- Local Rank
  - The **local_rank** is the ID of the process on a specific node.
  - If you're on node A and running 4 processes (e.g., 4 GPUs), the local ranks will be 0, 1, 2, 3.

2 DeepSpeed ZeRO

ZeRO-DP: ZeRO Powered Data Parallelism

- ZeRO removes the redundancy across data parallel process
- Stage 1: partitioning optimizer states
- Stage 2: partitioning gradients
- Stage 3: partitioning parameters



Samyam et al. ZeRO: Memory Optimizations Toward Training Trillion Parameter Models. SC.2020.

2 DeepSpeed ZeRO

- ZeRO Redundancy Optimizer Process (https://www.microsoft.com/en-us/research/blog/zero-deepspeed-new-system-optimizations-enable-training-models-with-over-100-billion-

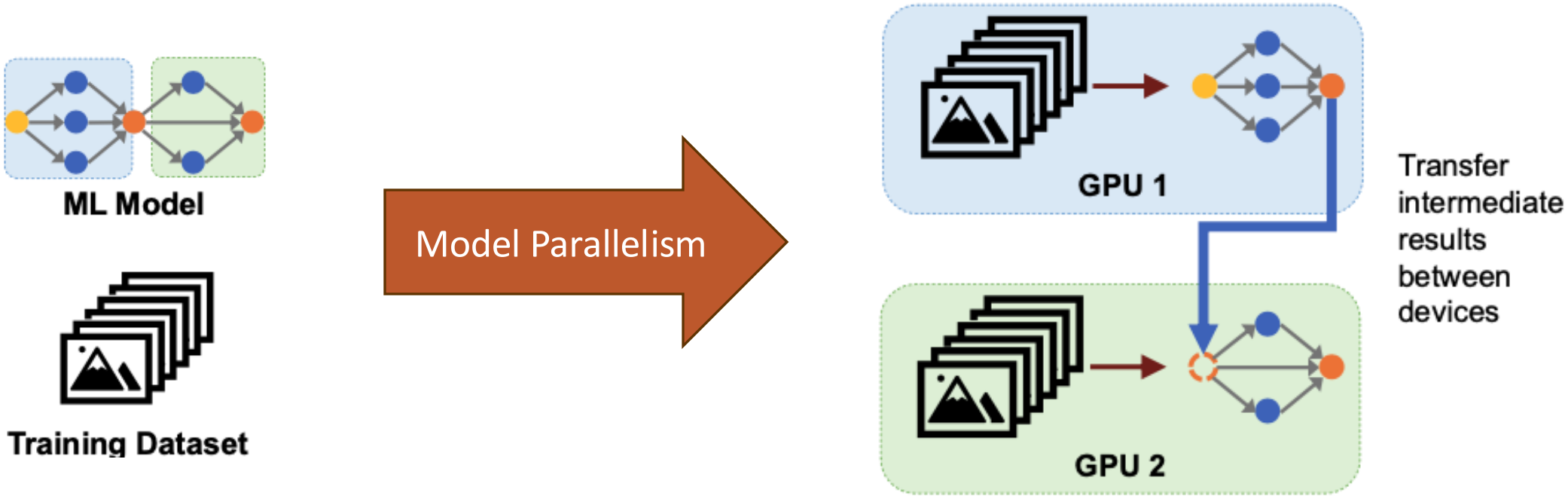
ZeRO 4-way data parallel training

Using:

- P_{os} (Optimizer state)
- P_g (Gradient)
- P_p (Parameters)

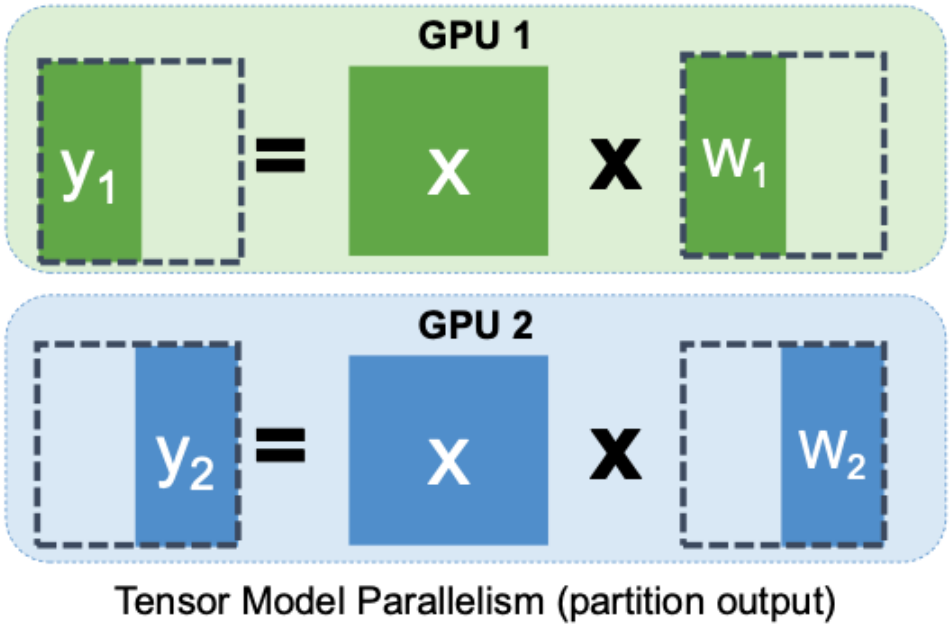
3 Model Parallelism

- Model parallelism distributes the model layers across GPUs.
  - On the forward pass, the first GPU processes a batch of data and passes it to the next group of layers on the next GPU.
  - For the backward pass, data is sent backward from the final layer to the first layer.
- Model parallelism is a useful strategy for training models that are too large to fit into the memory of a single GPU. However, GPU utilization is unbalanced because only one GPU is active at a time. Passing results between GPUs also adds communication overhead and it can be a bottleneck.

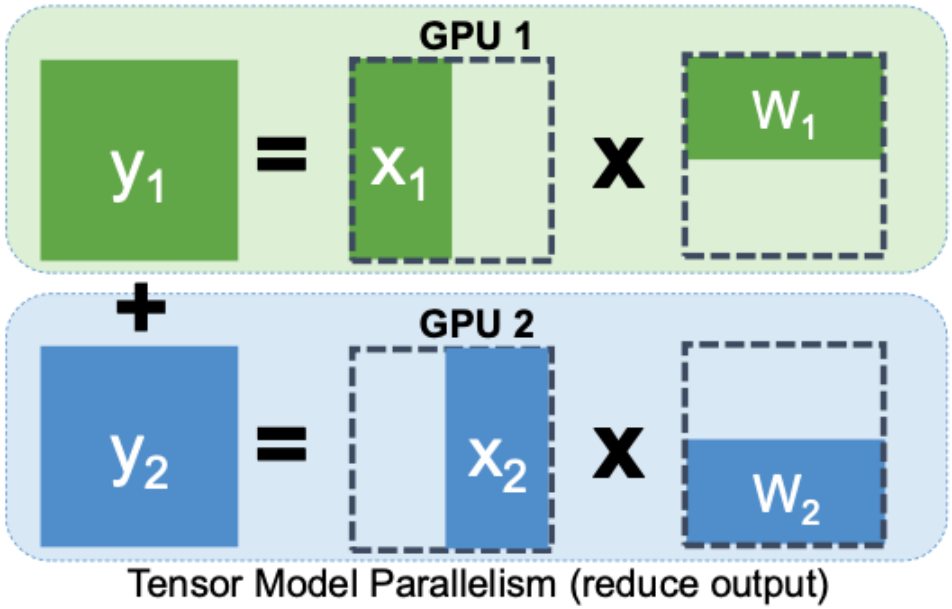


4 Tensor Parallelism

- Each tensor is split up into multiple chunks, so instead of having the whole tensor reside on a single GPU, each shard of the tensor resides on its designated GPU.
- During processing each shard gets processed separately and in parallel on different GPUs and the results are synced at the end of the step.



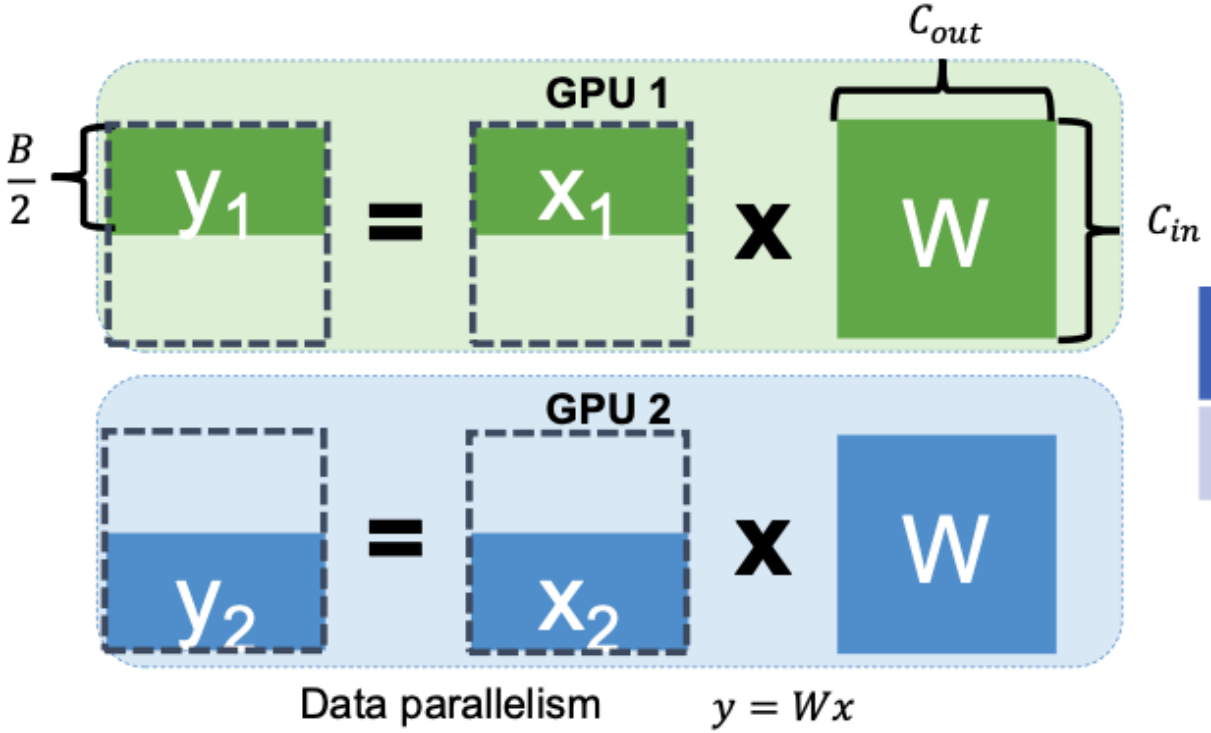
Tensor Model Parallelism (partition output)



Tensor Model Parallelism (reduce output)

$$y = y_1 + y_2$$

4 Comparing Data and Tensor Parallelism

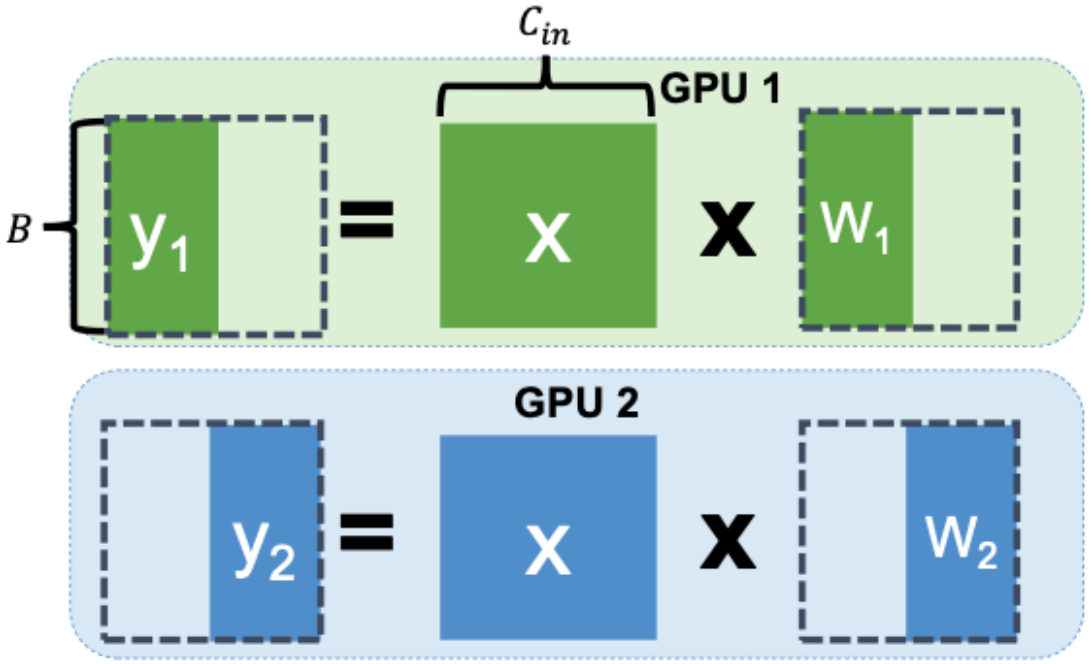


Data parallelism $y = Wx$

Forward Processing	Backward Propagation	Gradients Sync
0	0	$2 * C_{out} * C_{in}$

Communication Cost of Data Parallelism

4 Comparing Data and Tensor Parallelism

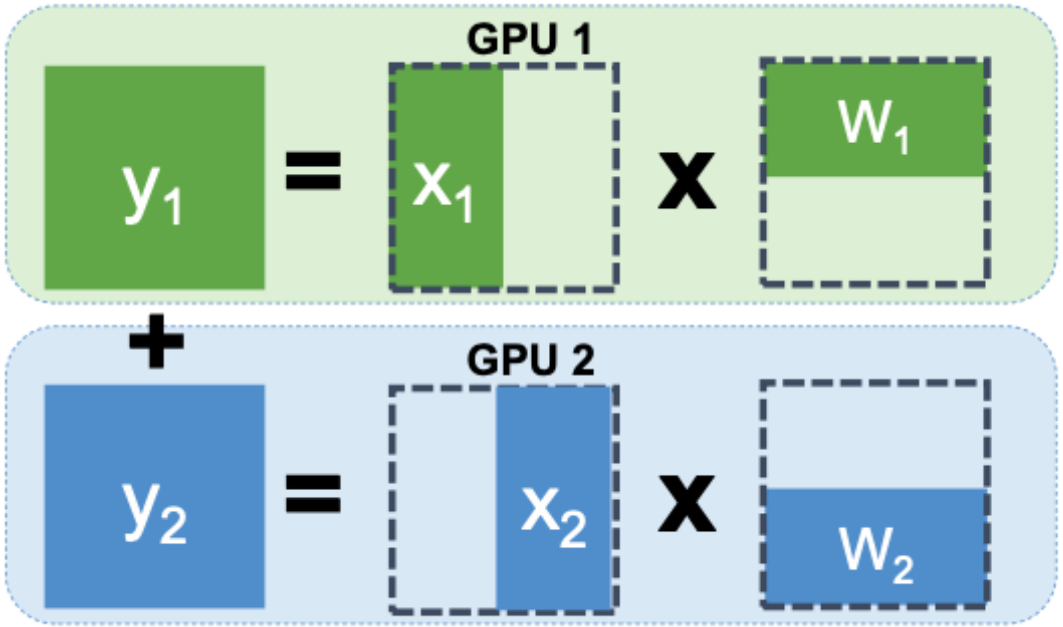


Tensor Model Parallelism (partition output)

Forward Processing	Backward Propagation	Gradients Sync
$B * C_{in}$	$B * C_{in}$	0

Communication Cost of Tensor Model Parallelism

4 Comparing Data and Tensor Parallelism



Tensor Model Parallelism (Reduce output)

$$y = y_1 + y_2$$

Forward Processing	Backward Propagation	Gradients Sync
$2 * B * C_{out}$	0	0

Communication Cost of Tensor Model Parallelism

4 Comparing Data and Tensor Parallelism

- Data parallelism: $C_{out} * C_{in}$
- Tensor model parallelism (partition output): $B * C_{in}$
- Tensor model parallelism (reduce output): $B * C_{out}$

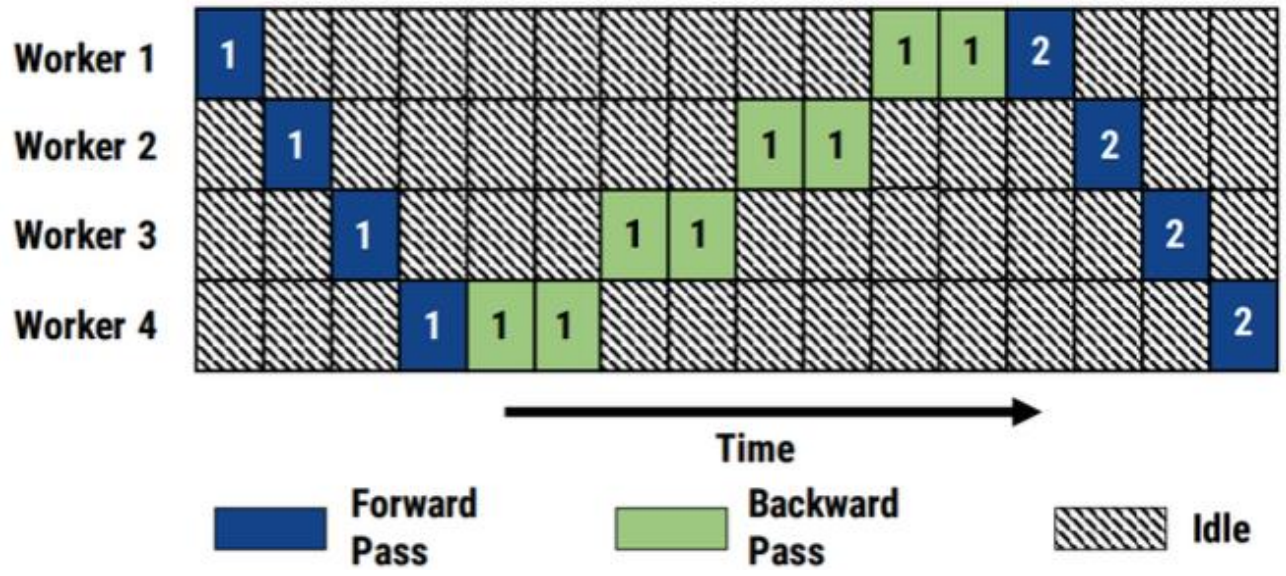
➢ The best strategy depends on the model and underlying machine

➢ Issue with Model Parallelism

- Under-utilization of compute resources
- Low overall throughput due to resource utilization

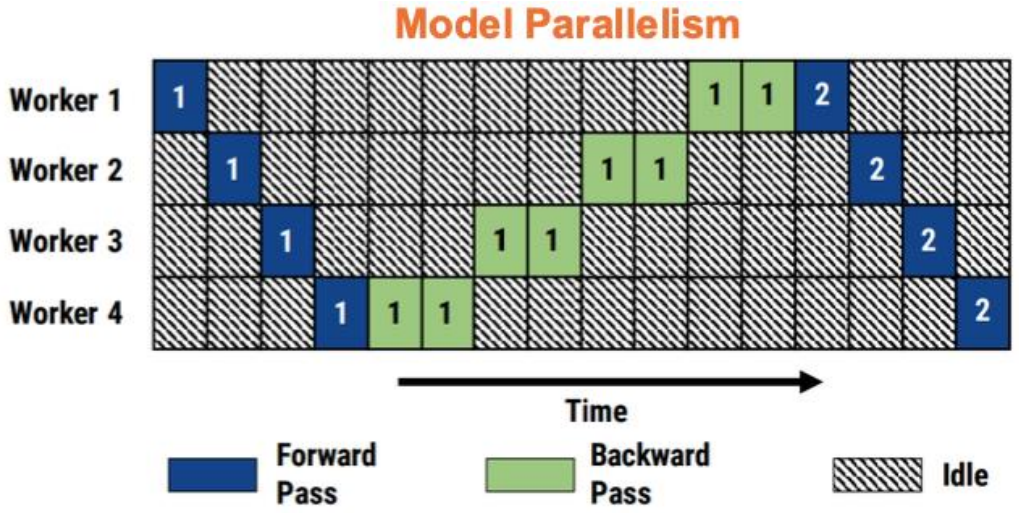
➢ Issue with Data Parallelism

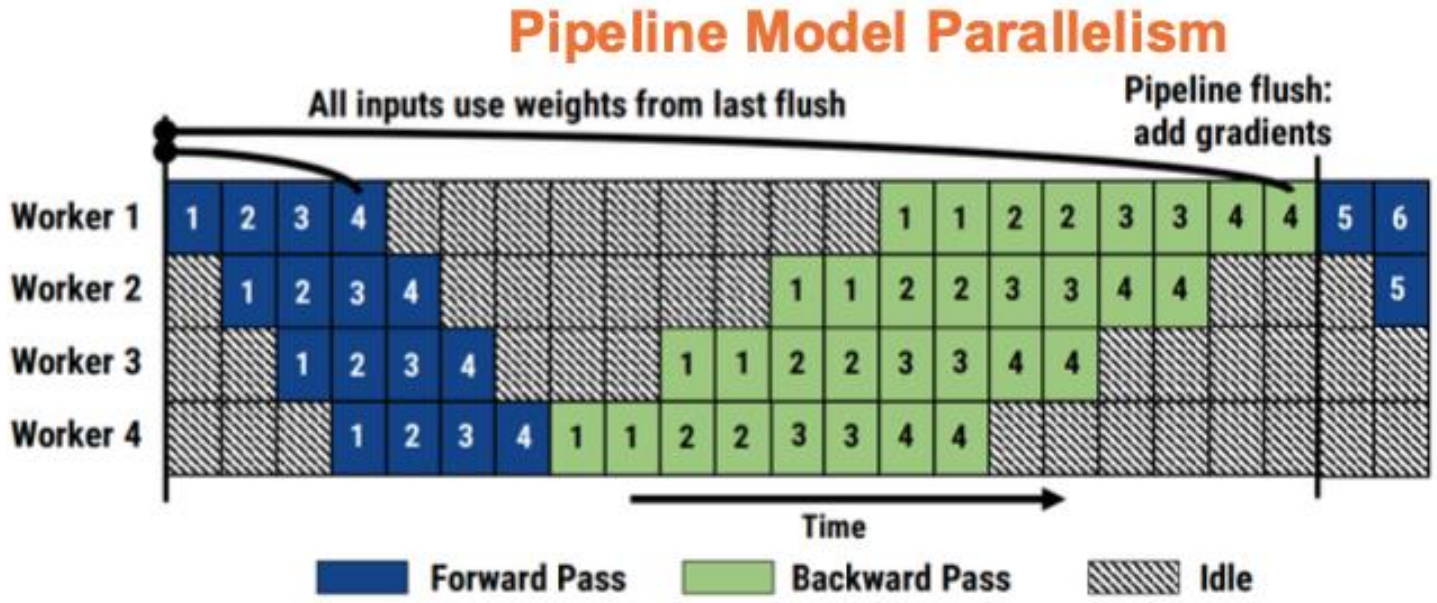
- Each GPU saves a replica of the entire model
- Cannot train large models that exceed GPU device memory



5 Pipeline Parallelism

- Mini-batch: the number of samples processed in each iteration
- Divide a mini-batch into multiple micro-batches
- Pipeline the forward and backward computations across micro-batches

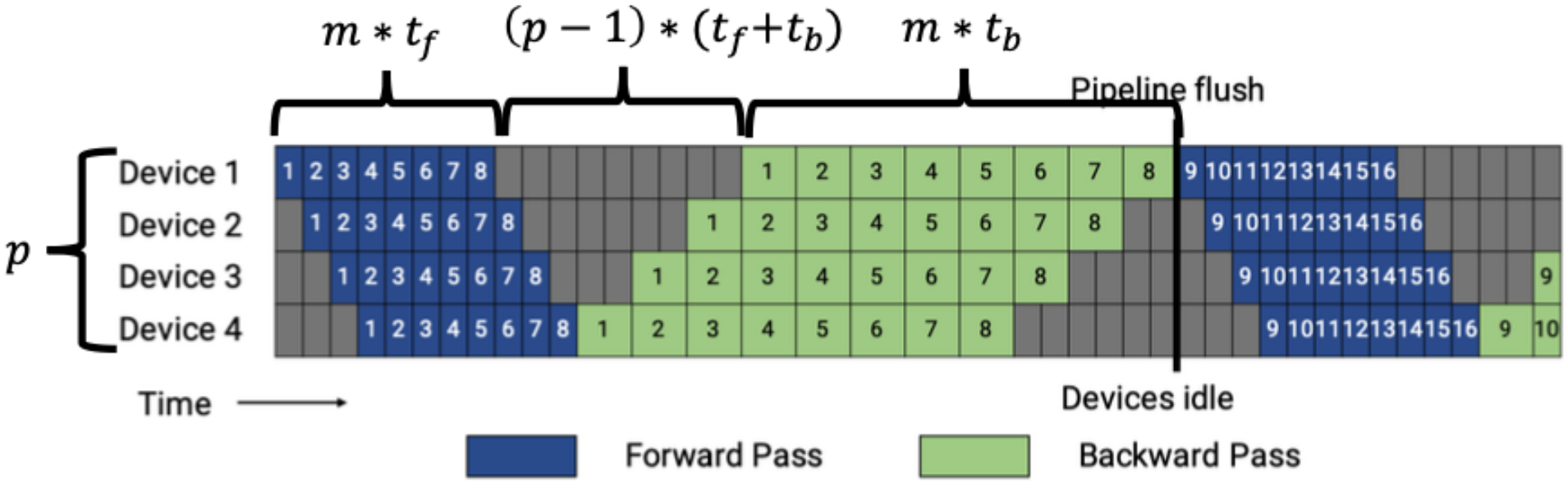




GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism. NIPS 2019.

5 Pipeline Parallelism

- m: Micro-batches in a mini-batch
- p: Number of pipeline stages
- All stages take t_f / t_b to process a forward (backward) micro-batch



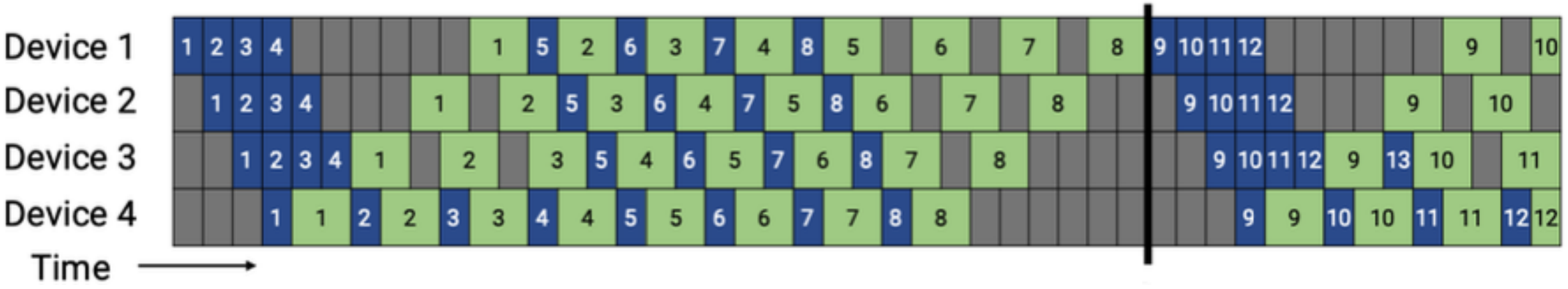
$$\textbf{BubbleFraction} = \frac{(p-1) * (t_f + t_b)}{m * t_f + m * t_b} = \frac{p-1}{m}$$

GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism. Google, NIPS 2019.

5 Pipeline Parallelism (1F1B)

- One-forward-one-backward in the steady state
- Limit the number of in-flight micro-batches to the pipeline depth *p*
- Reduce memory footprint of pipeline parallelism
- Does not reduce pipeline bubble

Can we reduce pipeline bubble?



PipeDream: Generalized Pipeline Parallelism for DNN Training. MSR. SOSP 2019.

5 Pipeline Parallelism (interleaved 1F1B)

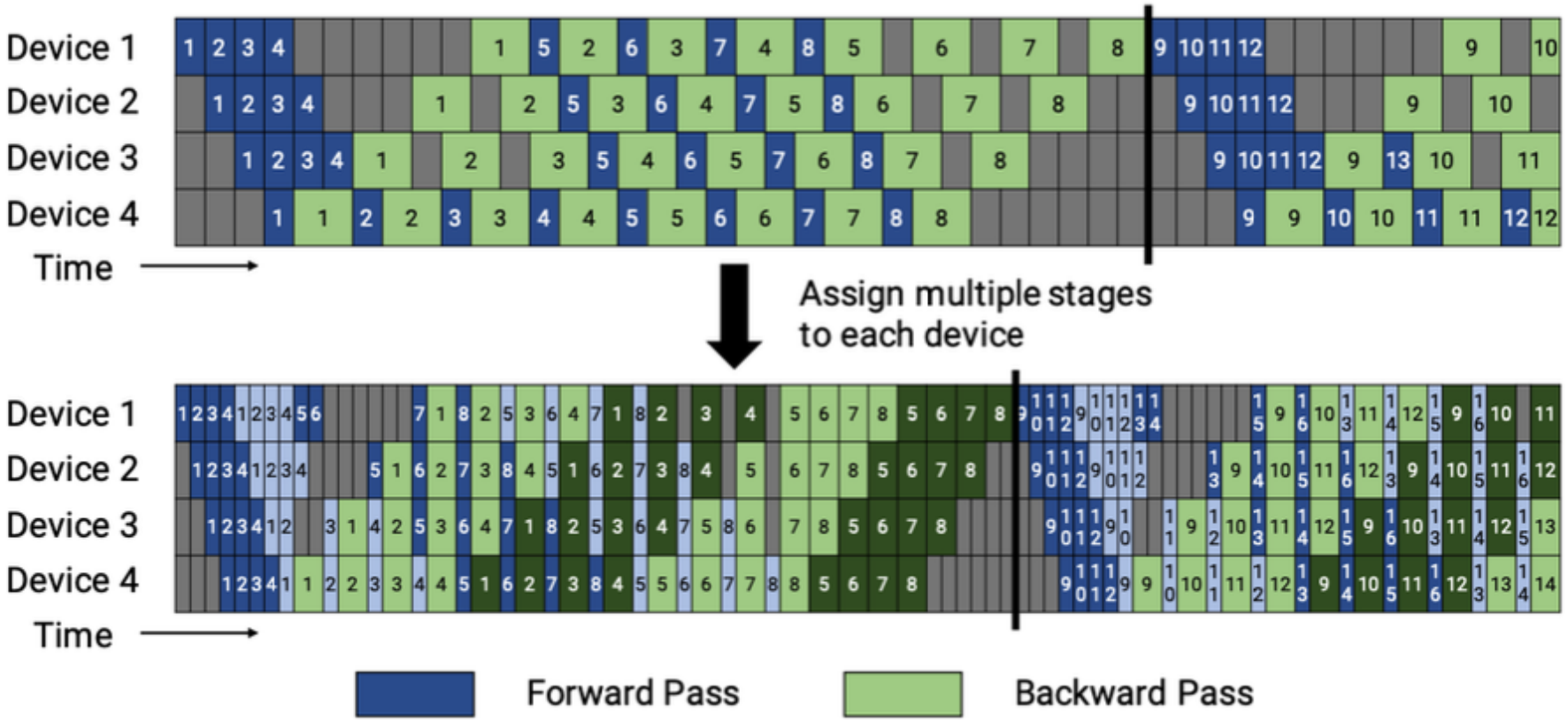
- Further divide each stage into v sub-stages
- The forward (backward) time of each sub-stage is $\frac{t_f}{v}(\frac{t_b}{v})$

Pipeline parallelism with 1F1B Schedule

$$BubbleFraction = \frac{p-1}{m}$$

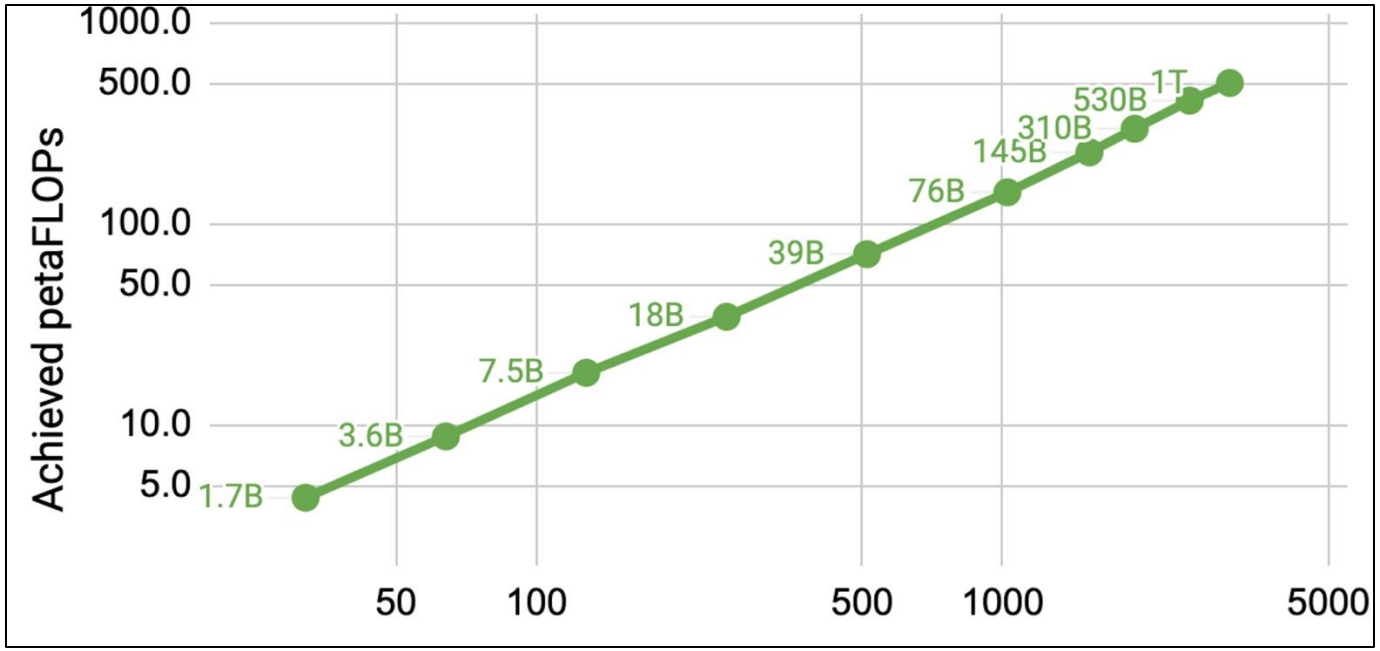
Pipeline parallelism with interleaved 1F1B Schedule

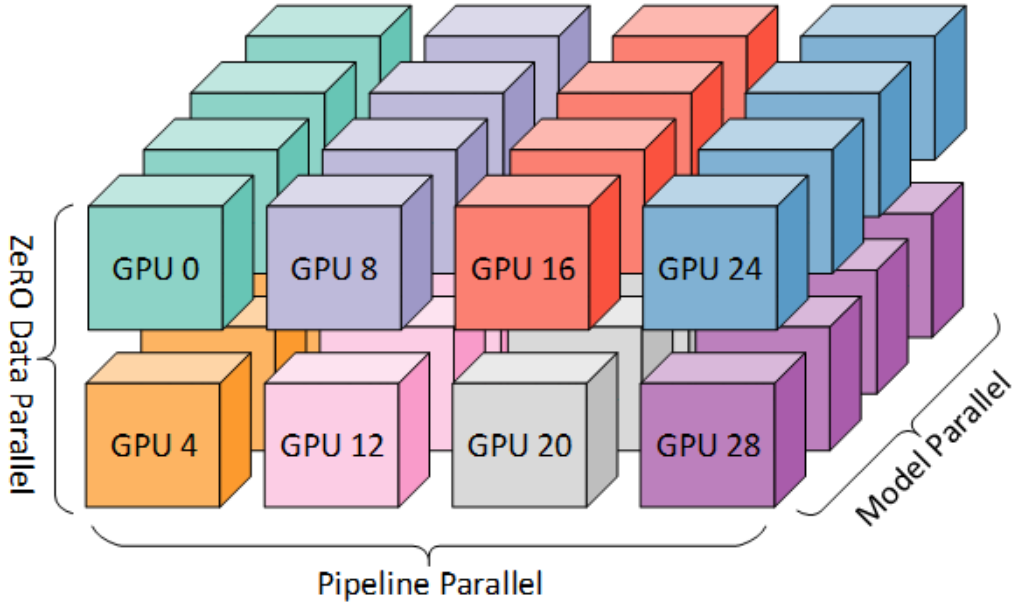
$$BubbleFraction = \frac{1}{v} * \frac{p-1}{m}$$



3D Parallelism

- Data, model, and pipeline parallelism each perform a specific role in improving memory and compute efficiency.
- Each dimension in 3D parallelism is carefully mapped onto the workers to achieve maximum compute efficiency by exploiting bandwidth amplification during communication.



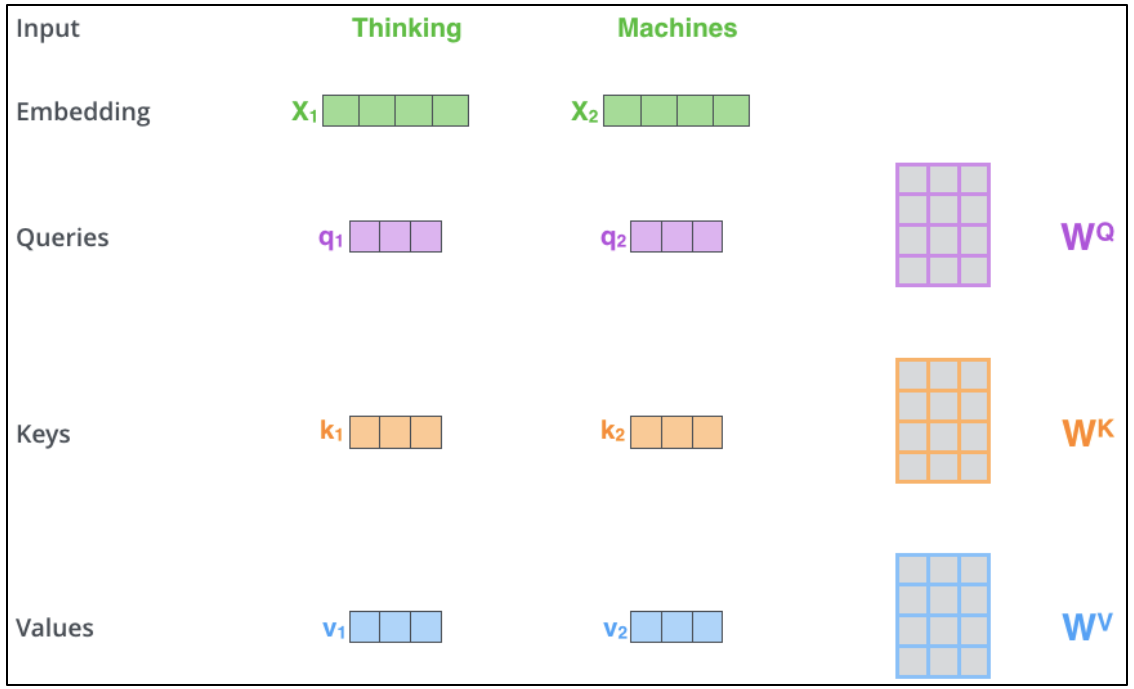


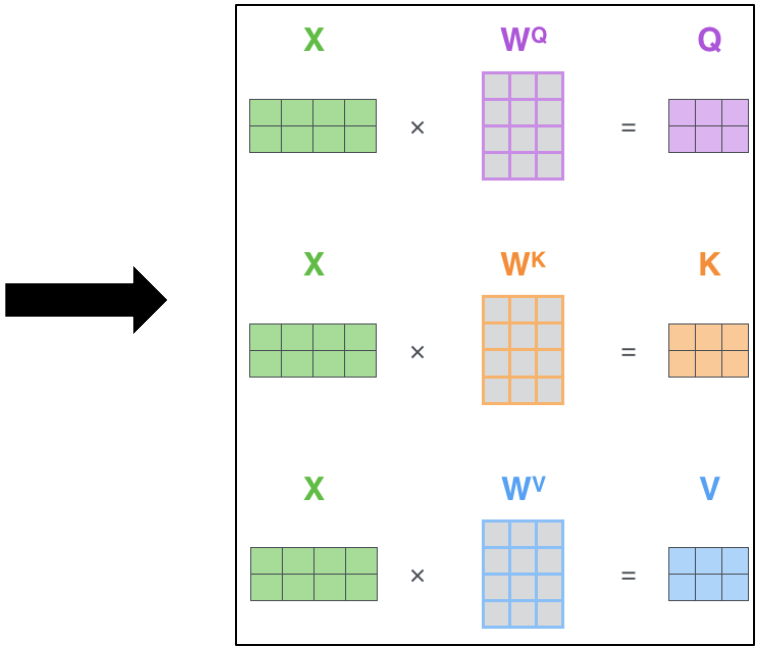
Topology aware 3D mapping

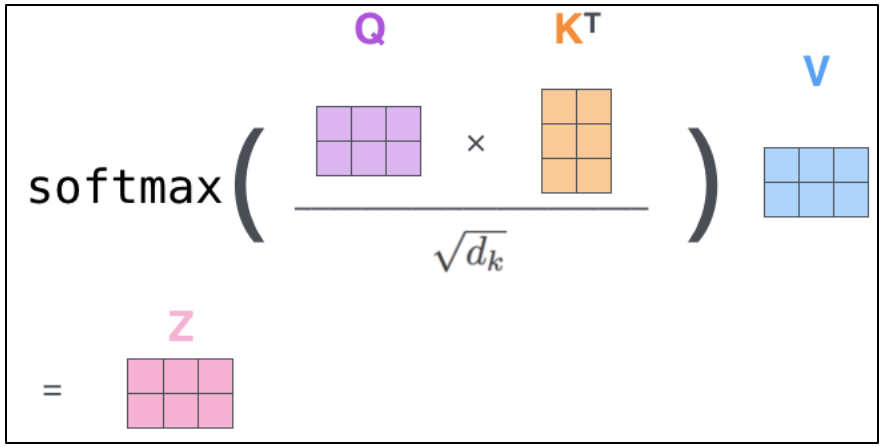
https://github.com/deepspeedai/Megatron-DeepSpeed

https://github.com/huggingface/accelerate

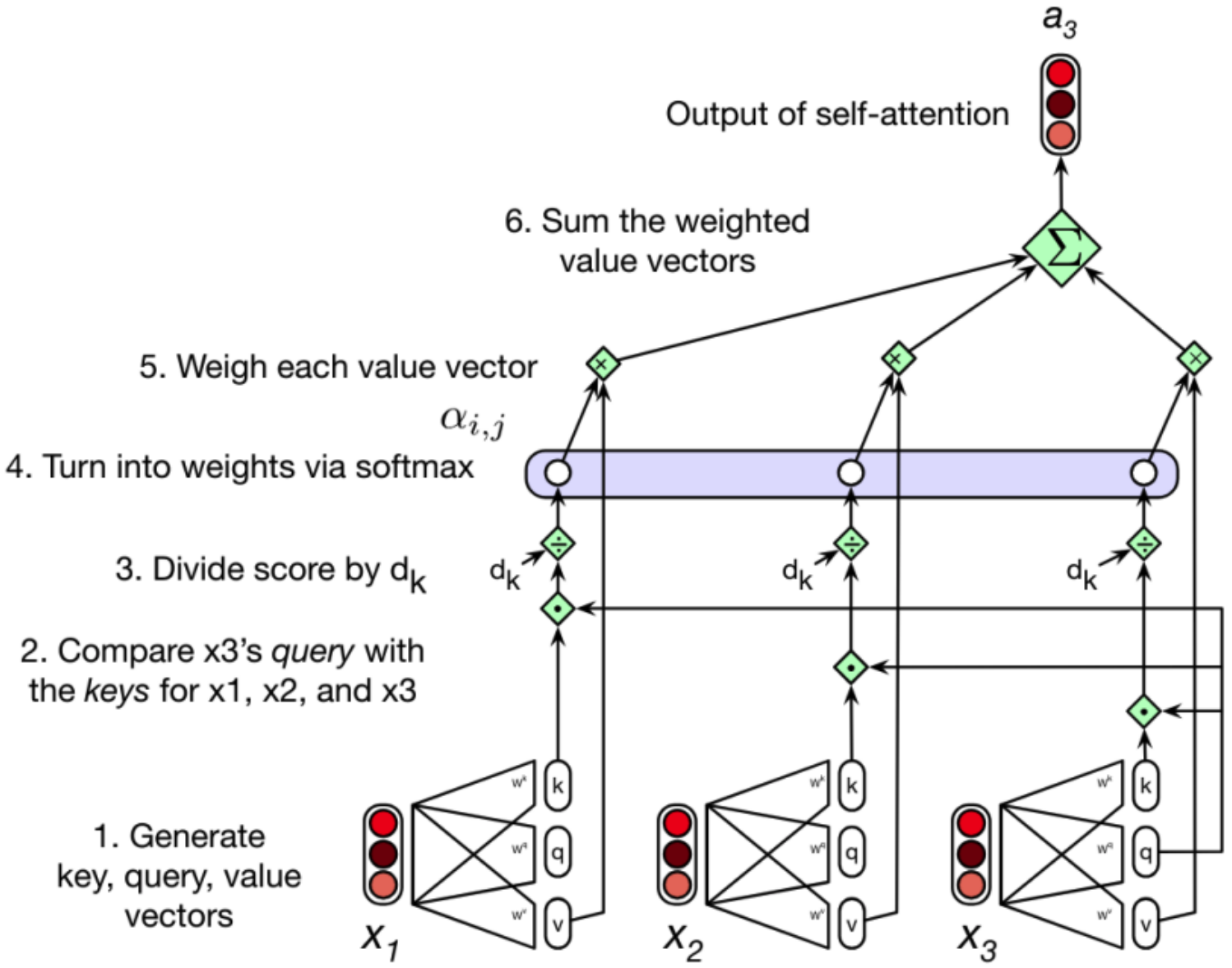
Recap: Attention with Q,K and V







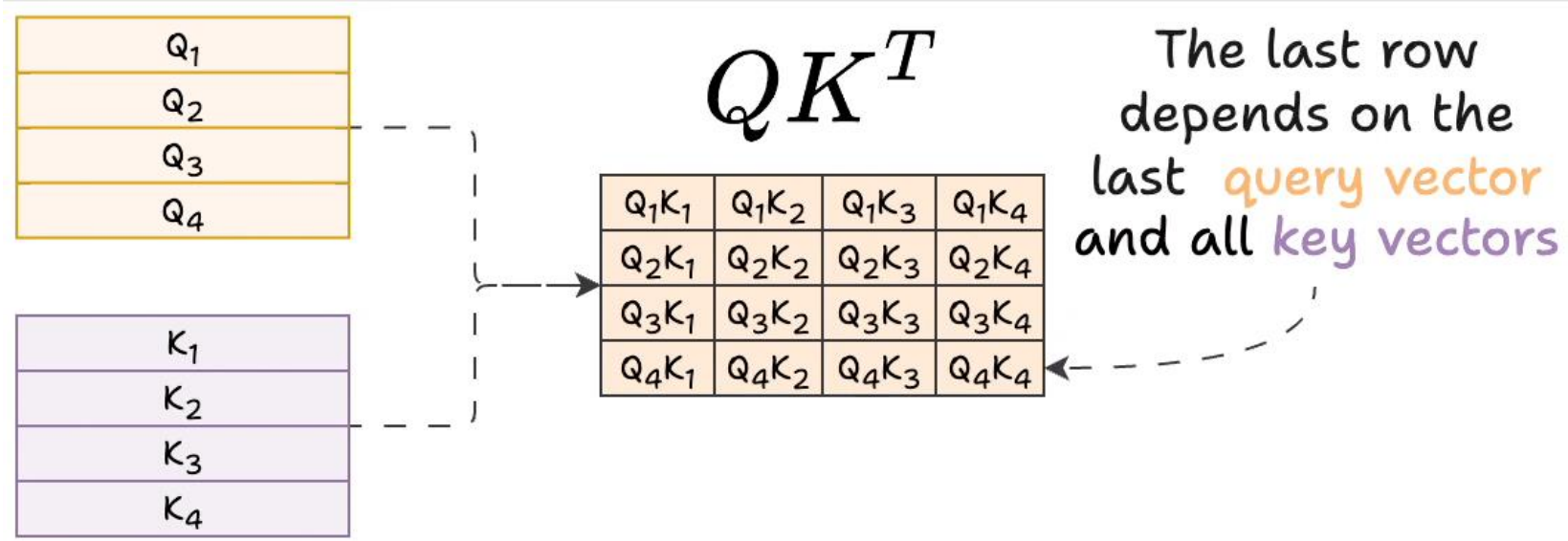
Recap: Attention with Q,K and V



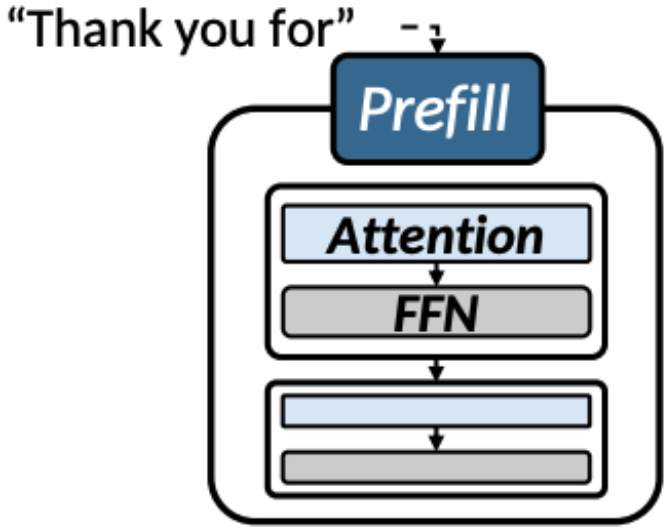
Since each output y_i is computed independently, the entire process can be parallelized by taking advantage of matrix multiplication

6 KV Cache

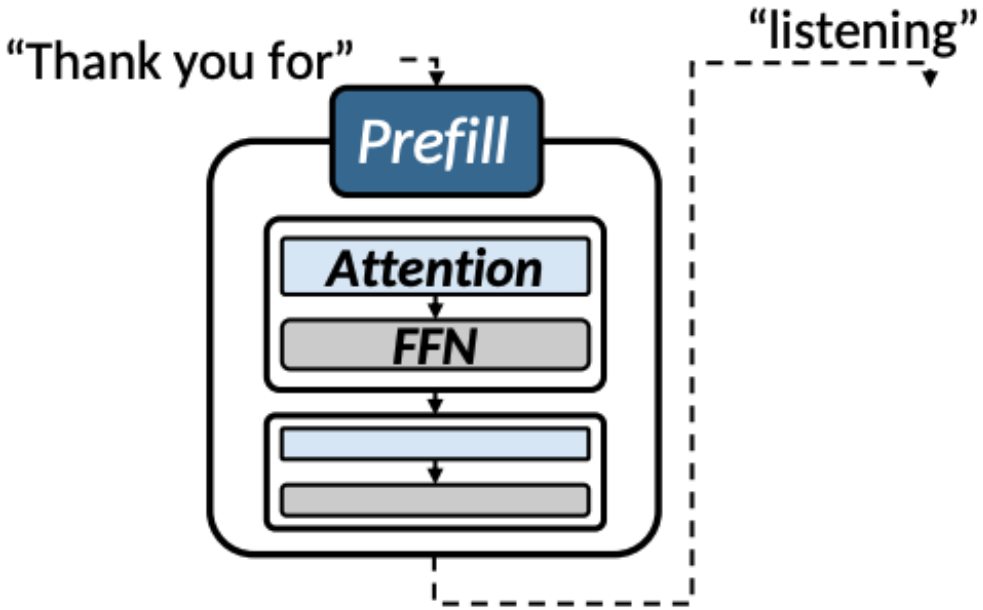
- KV Cache is applied only in the inference stage (test-time)
- KV Cache only exists in decoder
- It is used for accelerating the speed of matrix multiplication
- KV Cache will increate memory usage



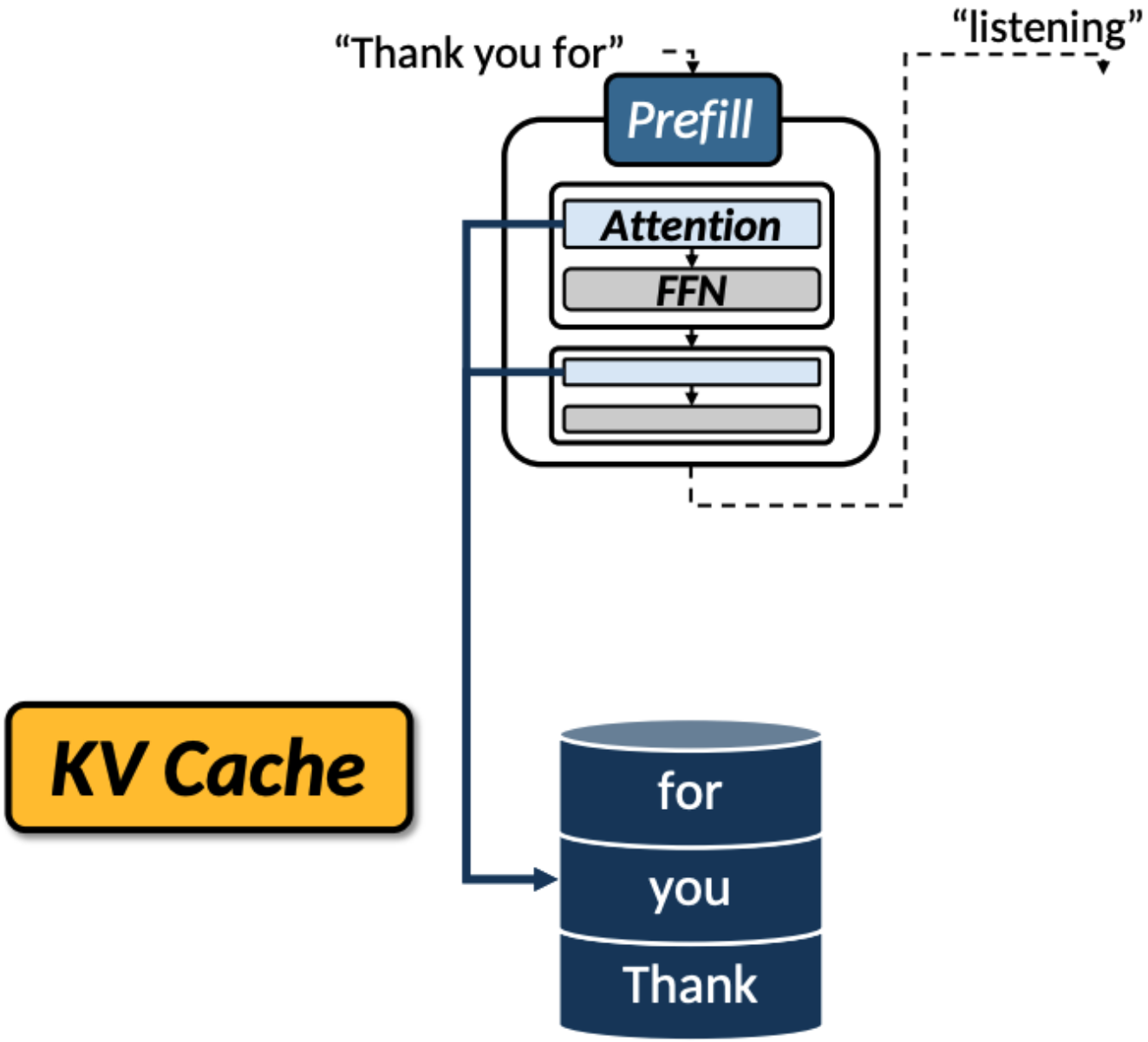
6 KV Cache



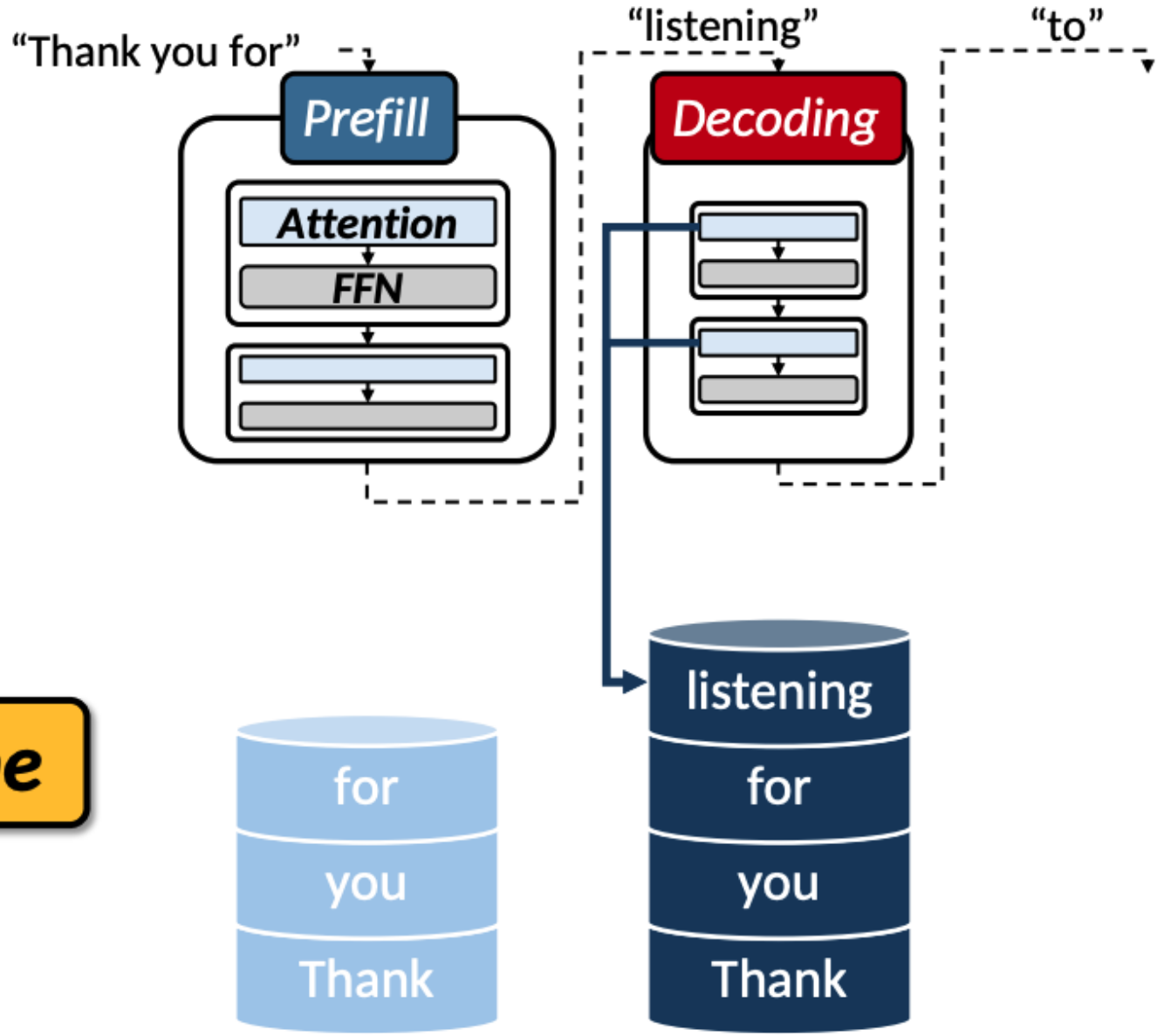
6 KV Cache



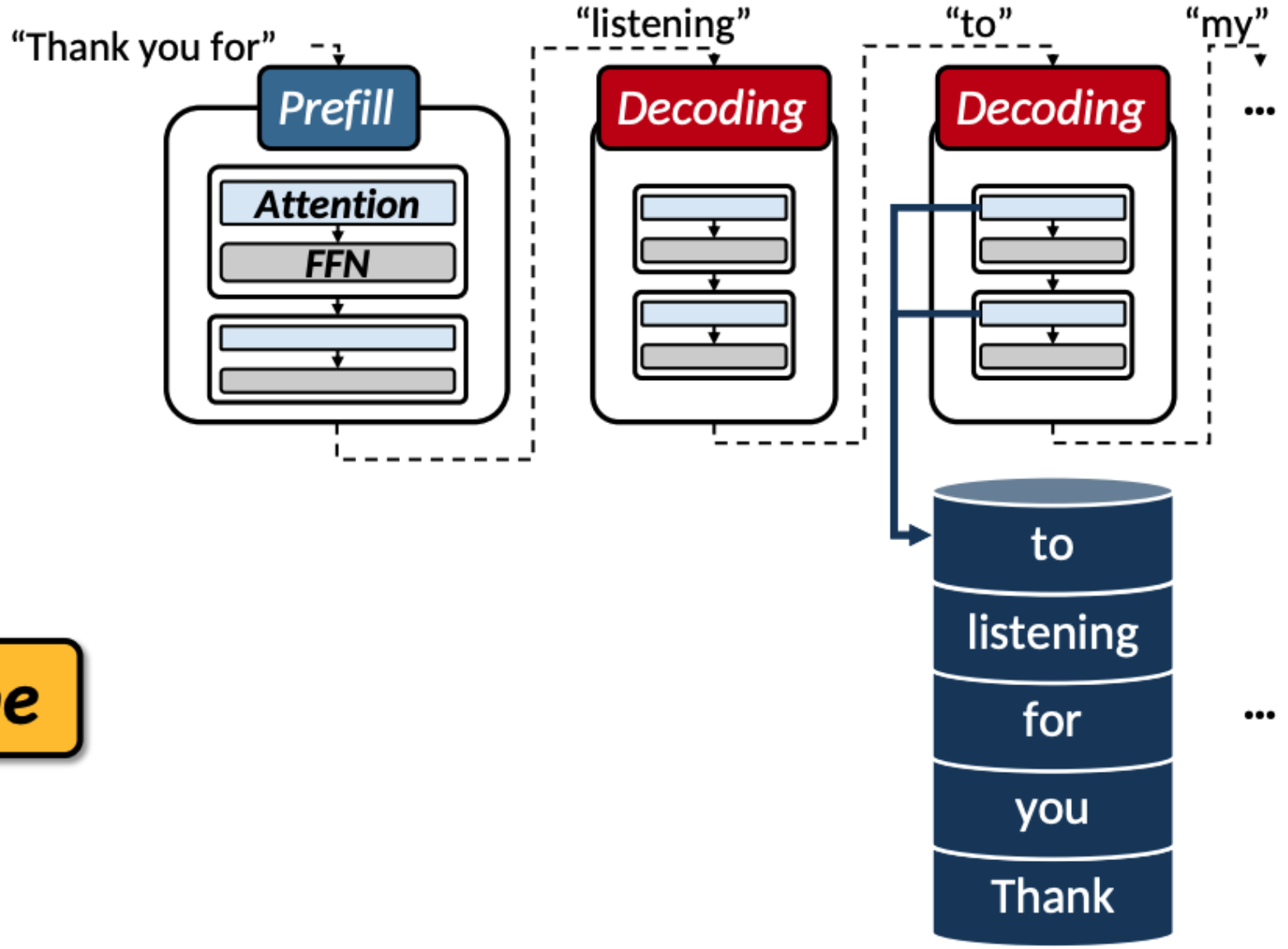
6 KV Cache


"Thank you for"
"listening"
Prefill
Attention
FFN
KV Cache
for
you
Thank

6 KV Cache


"Thank you for"
Prefill
Attention
FFN
"listening"
Decoding
"to"
KV Cache
for
you
Thank
listening
for
you
Thank

6 KV Cache



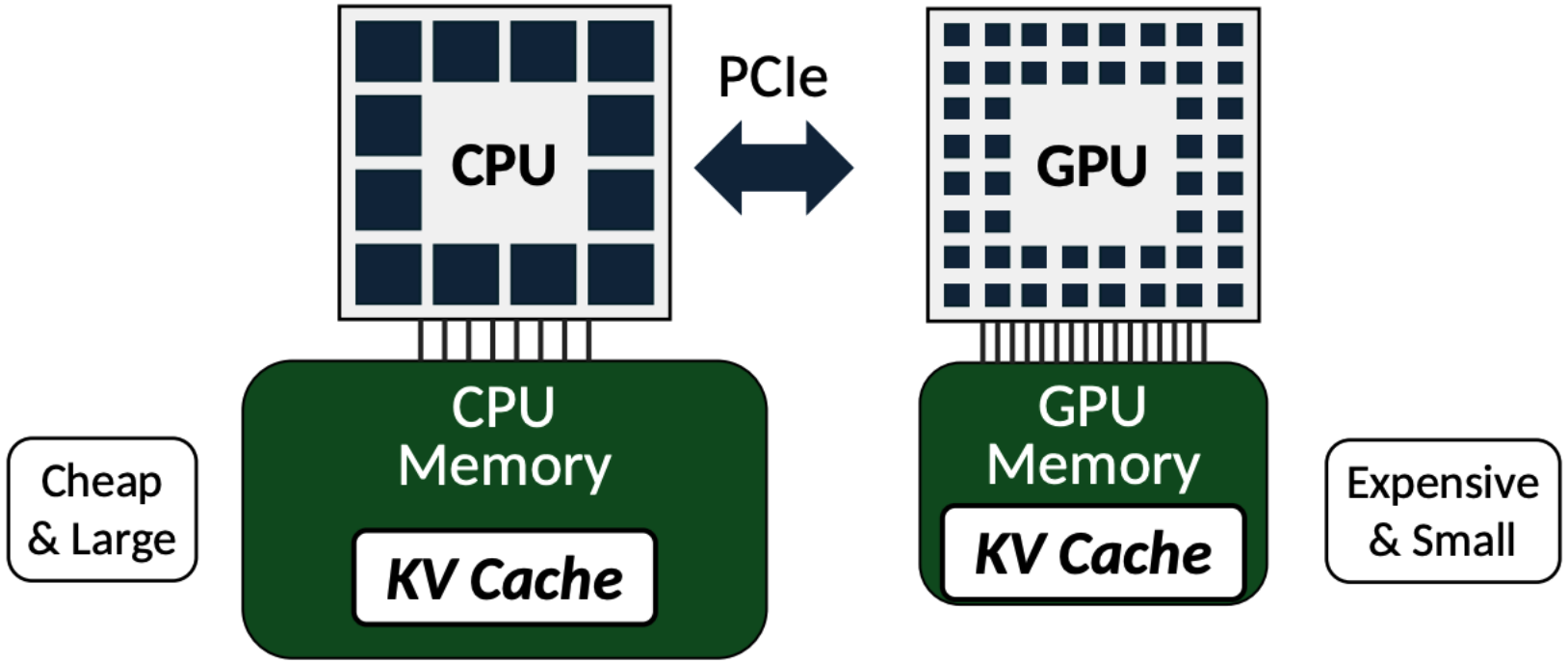
KV Cache

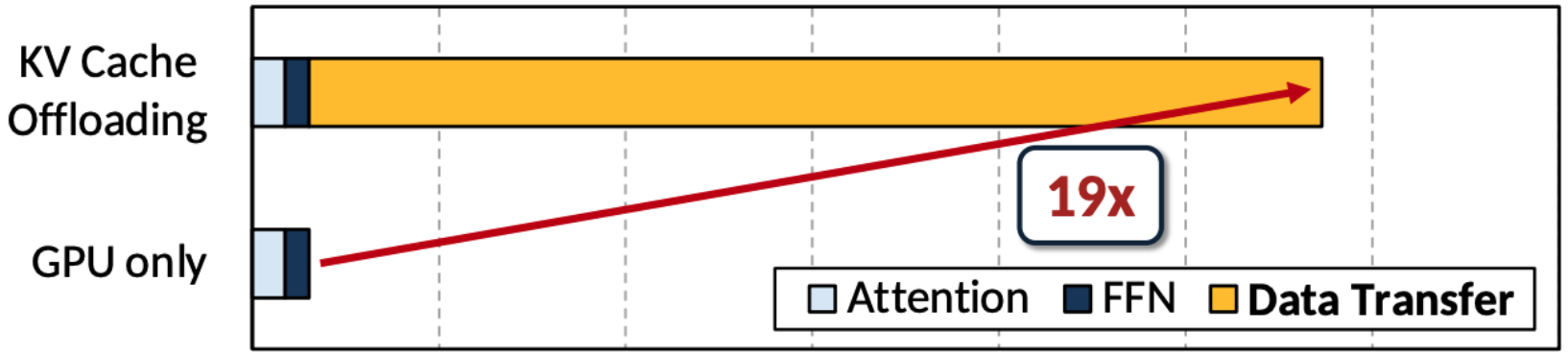
6 KV Cache

How big can the KV cache grow?

- For each token, we need to store 2 vector tensors of size d_{head} for each attention head of each attention layer.
  - 4 bytes in FP32, 2bytes in half-precision (BF16, FP16), 1byte for 8-bit INT8, FP8, etc.
- Let b the batch size, t the total sequence length, n_{layers} number of decoder blocks/attention layers, n_{heads} number of attention heads per attention layer, d_{head} hidden dimension of attention layer, p_a the precision
- The memory consumption (in bytes) of KV cache of a MHA model:
  - $2 \cdot b \cdot t \cdot n_{layers} \cdot n_{heads} \cdot d_{head} \cdot p_a$
  - Grows linearly with the batch size and the total sequence length (prompt + completion)
  - Since total sequence length is unknown ahead of time, KV cache memory reuqirements therefore unknown making memory management.

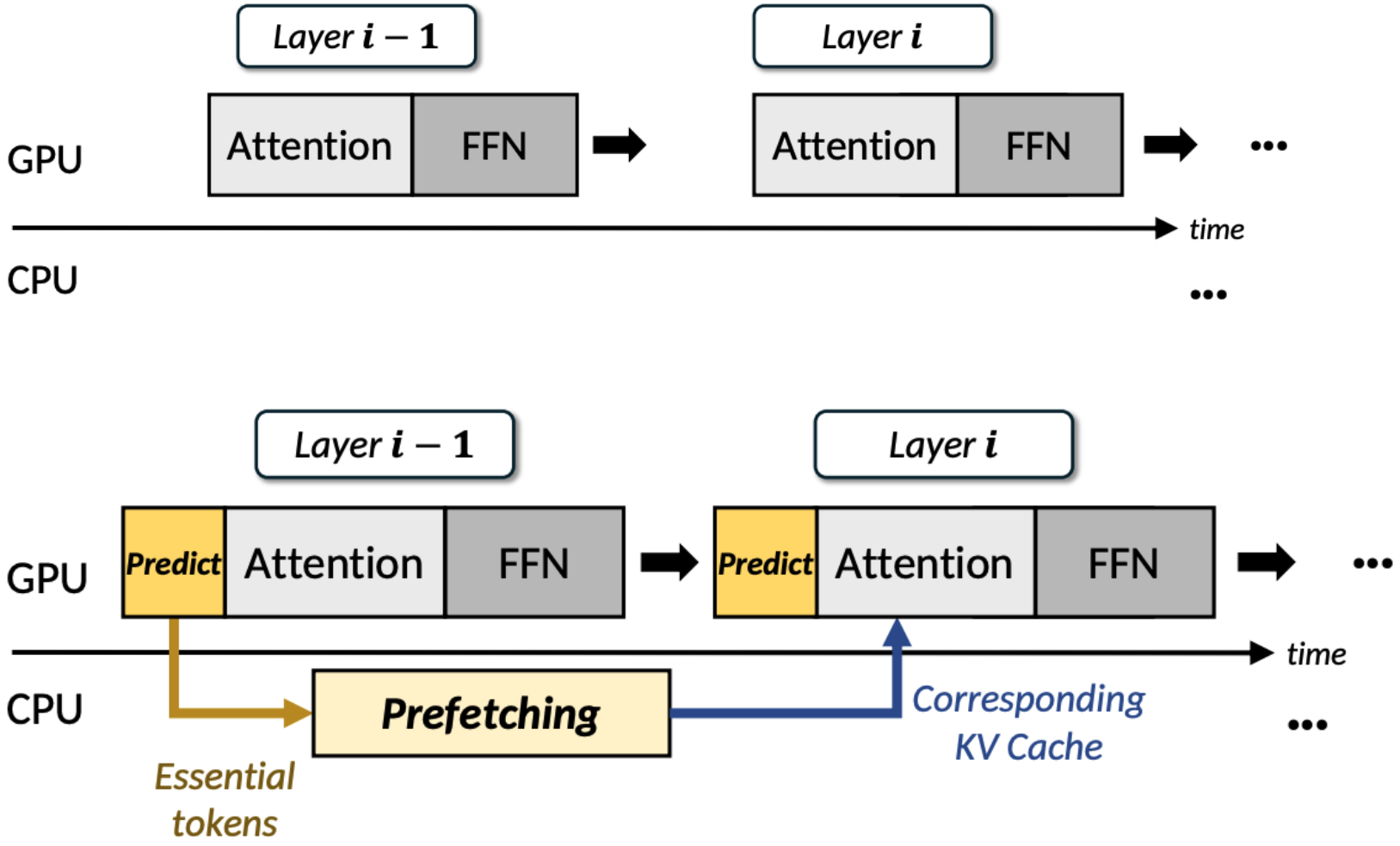
7 KV Cache Offloading



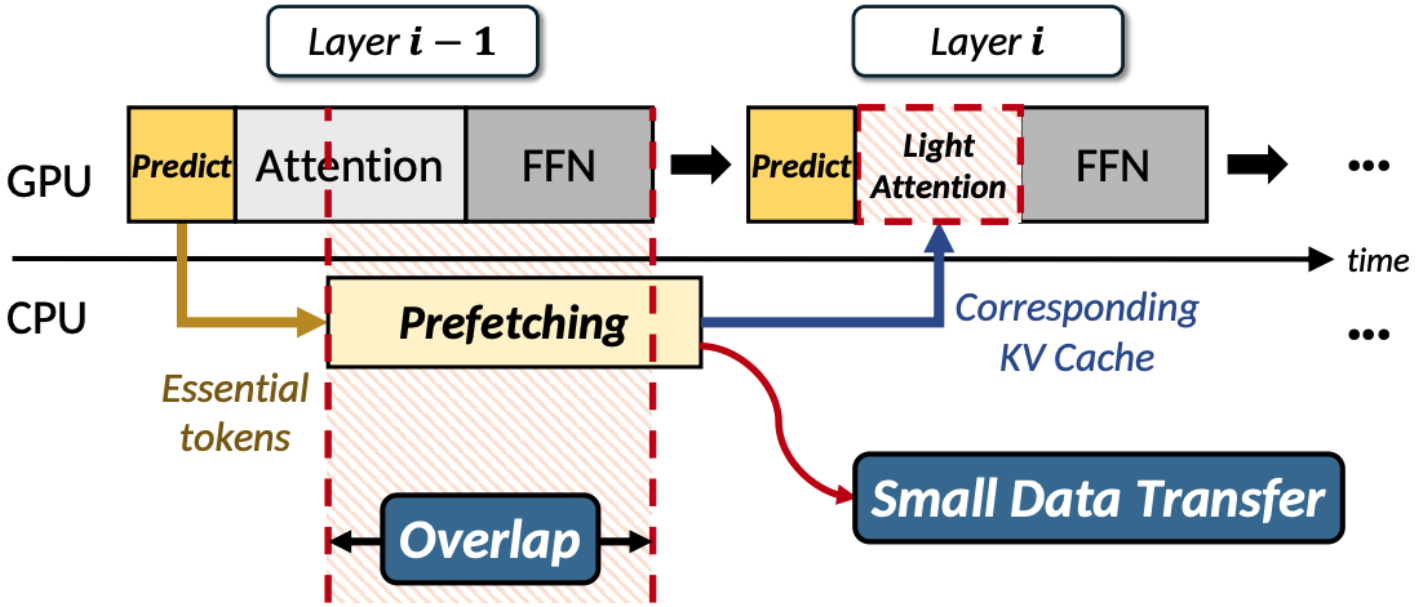


- Transfer Less
- Transfer Early

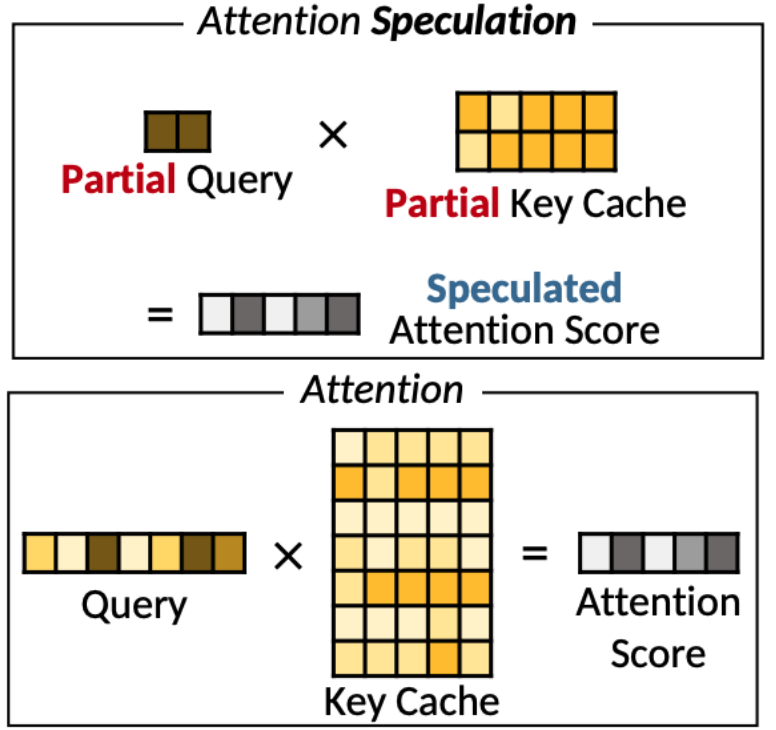
8 Speculative KV Prefetching



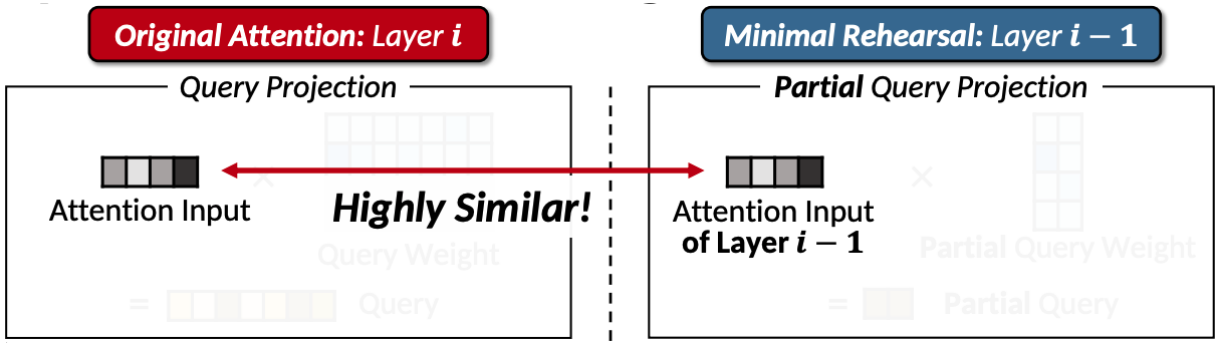
8 Speculative KV Prefetching



- How to predict essential tokens?



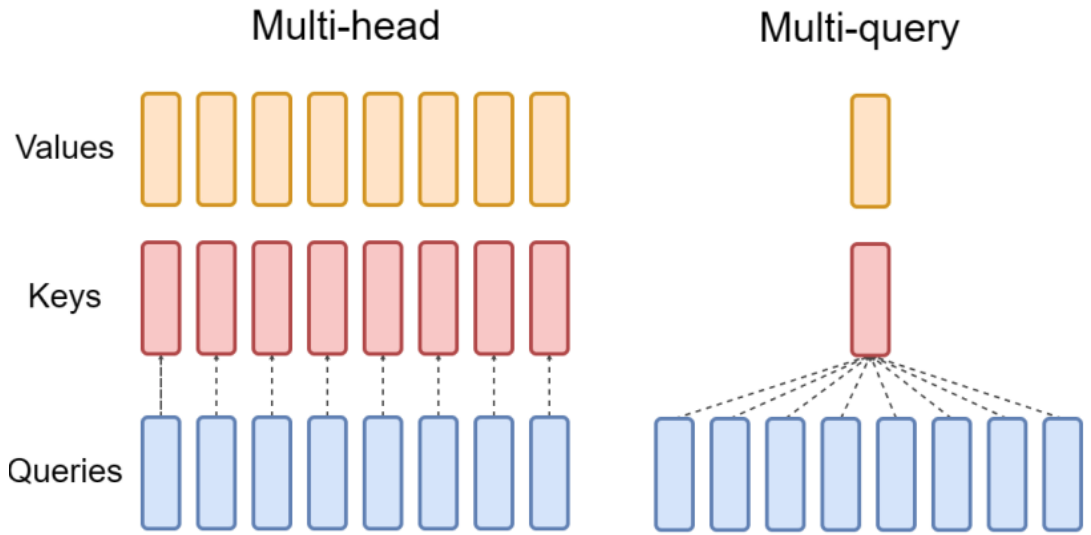
- Token attention sparsity (a few tokens dominate attention).
- Attention patterns across layers are similar.
- LLMs often show temporal locality in attention — they often look at nearby or similarly important tokens.



9 Attention Variants - MQA

Multi-Query Attention

- MQA is a more computationally efficient attention mechanism that simplifies MHA to reduce memory usage and intermediate calculations.
- Instead of training a unique key head and value head for each attention head, MQA uses a *single* key head and *single* value head at each layer.
- Therefore, key vectors and value vectors are calculated only once
- This single set of key and value vectors is then shared across all *h* attention heads.
- MQA allows a 10–100 times smaller key-value pair storage (or *KV cache*) and 12 times faster decoder inference.

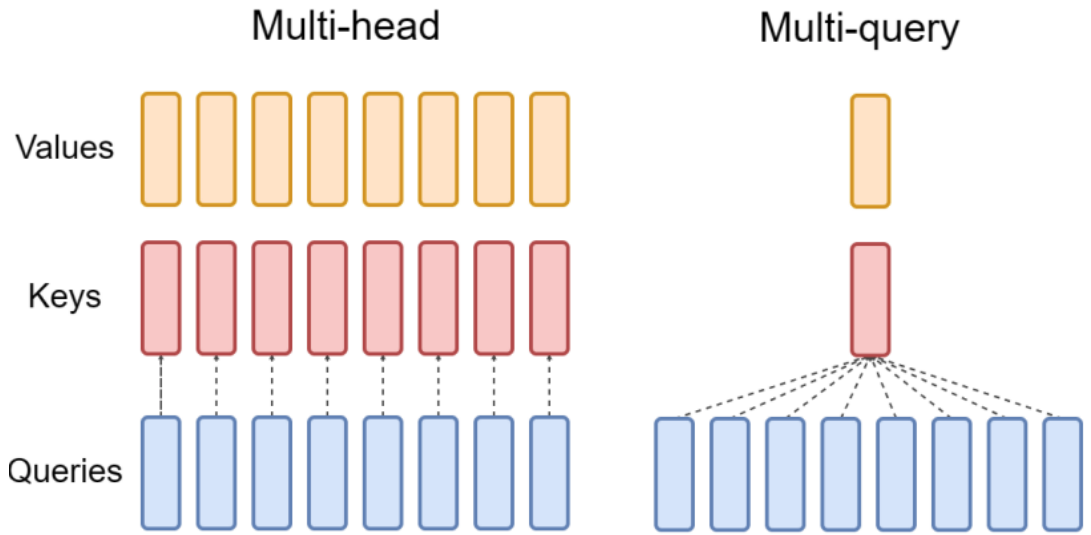


Noam Shazeer. Fast Transformer Decoding: One Write-Head is All You Need. 2019

9 Attention Variants - MQA

Disadvantages

- **Performance degradation:** Reducing number of unique, trainable model parameters reduces the model's capacity for knowledge and nuance.
- **Must be trained from scratch:** A model trained with standard MHA cannot be adapted to MQA.
- **Redundancies in tensor parallelism:** K and V values must be present on each node of the GPU cluster performing these operations, which means that in practice they must be replicated for each node. This is not an optimal use of compute resources despite still being more efficient than standard MHA.

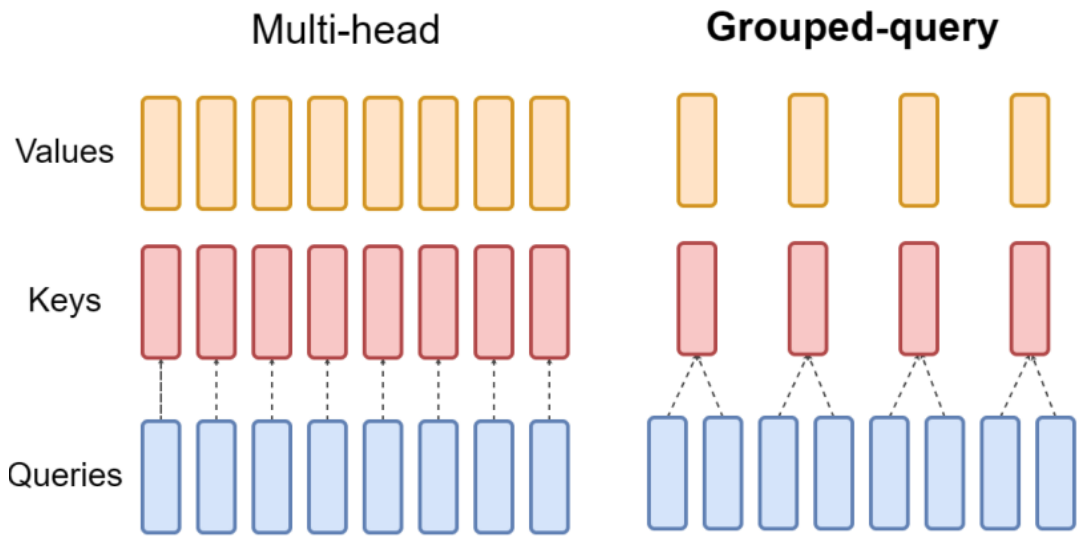


Noam Shazeer. Fast Transformer Decoding: One Write-Head is All You Need. 2019

9 Attention Variants - GQA

Group-Query Attention

- GQA is a more general, flexible formulation of multi-query attention that partitions query heads into multiple *groups* that each share a set of keys and values, rather than sharing one set of keys and values across all query heads.
- **Effective compromise:** GQA offers an ideal tradeoff between decoder inference speed and performance accuracy, in that it's nearly as accurate as MHA while being nearly as fast as MQA.
- Meta first adopted GQA for Llama2 in July 2023 and Llama 3 in 2024. So did Mistral AI adopt it in Mistral 7B.



The project matrices for K and V heads are mean-pooled into single projection matrices.

Noam Shazeer. Fast Transformer Decoding: One Write-Head is All You Need. 2019

9 Attention Variants - GQA

- Performance

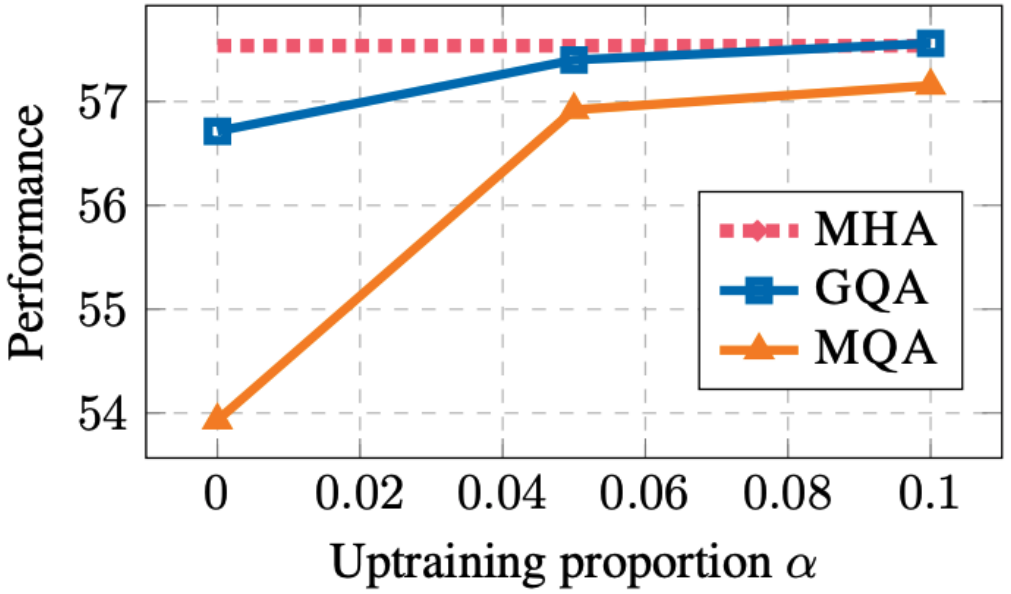
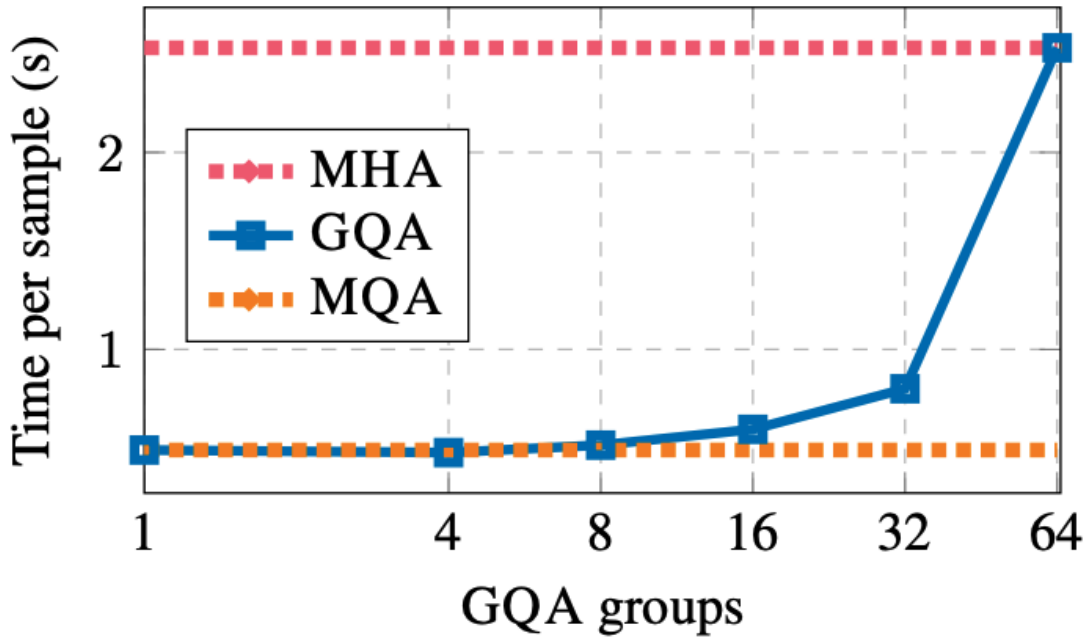
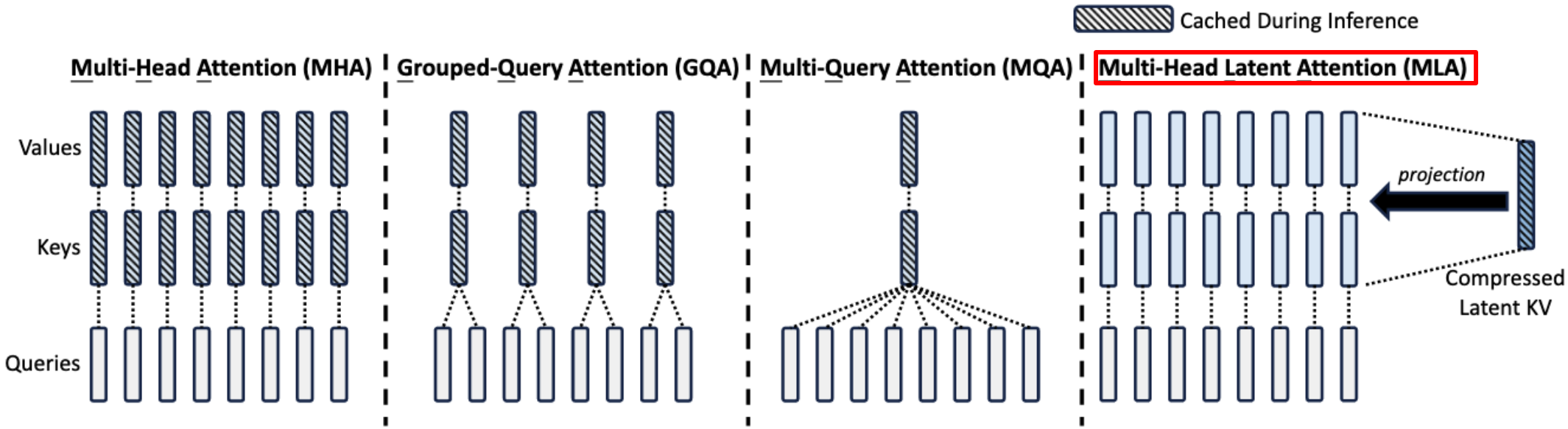


Figure 5: Performance as a function of uptraining proportion for T5 XXL models with MQA and GQA-8.



9 Attention Variants - MLA



Low-Rank Key-Value Joint Compression

- Its core idea is to compress full key/value (KV) pairs into a low-dimensional latent space before reconstructing them during the attention process.

$$\mathbf{c}_t^{KV} = W^{DKV}\mathbf{h}_t,$$
$$\mathbf{k}_t^{C} = W^{UK}\mathbf{c}_t^{KV},$$
$$\mathbf{v}_t^{C} = W^{UV}\mathbf{c}_t^{KV},$$

where $\mathbf{c}_t^{KV} \in \mathbb{R}^{d_c}$ is the compressed latent vector for keys and values; $d_c (\ll d_h n_h)$ denotes the KV compression dimension; $W^{DKV} \in \mathbb{R}^{d_c \times d}$ is the down-projection matrix; and $W^{UK}, W^{UV} \in \mathbb{R}^{d_h n_h \times d_c}$ are the up-projection matrices for keys and values, respectively.

DeepSeek-V2: A Strong, Economical, and Efficient Mixture-of-Experts Language Model. 2024.

10 Linear Attention

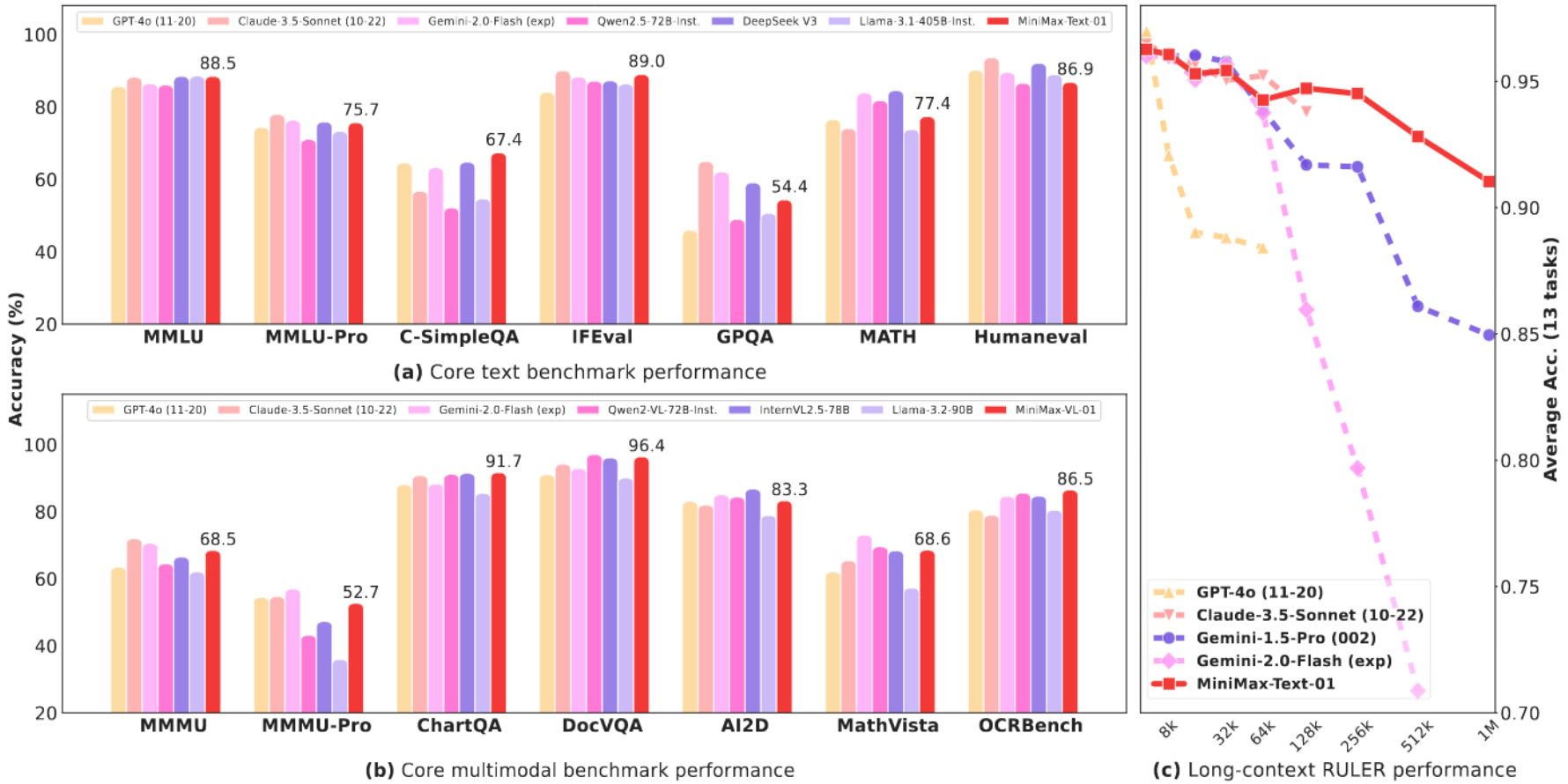


Figure 1 | **Benchmark performance.** **(a)** MiniMax-Text-01 on core text benchmarks. **(b)** MiniMax-VL-01 on core multimodal benchmarks. **(c)** MiniMax-Text-01 on the long-context RULER (Hsieh et al., 2024) benchmark. The performance of leading commercial and open-source models is presented for

[1] RULER: What's the Real Context Size of Your Long-Context Language Models? Aug 2024

Hybrid linear and SoftMax attention can achieve GPT-4o level perf?

MiniMax-01 (2025) used:

- Hybrid attention: 7 linear attention layers + 1 softmax attention layer
- Lightning attention2 (Qin et al. 2024b): simple linear attention with data-independent decay

Issues with Vanilla Attention

$$
\begin{aligned}
Q &= xW_Q, \\
K &= xW_K, \\
V &= xW_V,
\end{aligned} \tag{2}
$$

$$A_l(x) = V' = \text{softmax}\left(\frac{QK^T}{\sqrt{D}}\right)V.$$

- Training: quadratic time complexity
  - Expensive for long sequence modeling (e.g., video)
- Inference: linear memory complexity
  - Requires storing KV cache for each token (n_input+n_output)
  - High memory burden

SoftMax Related Information

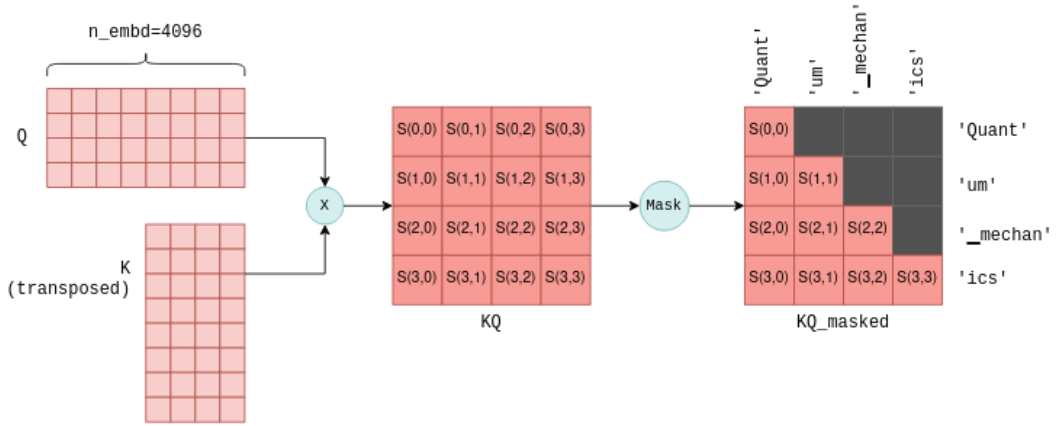
- Time complexity of Attention is $\mathcal{O}(n^2)$
- Without SoftMax it's just a matrix multiplication $\mathrm{QK^TV}$
  - Based on rate of union, we could right-multiply first to get a $d \times d$ matrix
  - Then using the left product Q, the complexity is just $\mathcal{O}(nd^2)$ since $d \ll n$. The fixed dimension of d can be interpreted as having a linear complexity for n.
- This is our final expectation： Attention with $\mathcal{O}(n)$ time complexity
- Why SoftMax is necessary：
- **Non-negative normalization:** converts any input into a probability value between 0 and 1.
- **Importance：**
  - SoftMax function can magnify the gap between larger input values and narrow the gap between smaller values.
  - This process highlights the differences in importance of the different inputs, allowing the model to better focus on the important information.

Causal Modeling

Softmax attention:

$$\text{Parallel training}: \quad \mathbf{O} = \text{softmax}(\mathbf{QK}^\top \odot \mathbf{M})\mathbf{V} \quad \in \mathbb{R}^{L\times d}$$

$$\text{Iterative inference}: \quad \mathbf{o_t} = \sum_{j=1}^{t} \frac{\exp(\mathbf{q}_t^\top \mathbf{k}_j)}{\sum_{l=1}^{t} \exp(\mathbf{q}_t^\top \mathbf{k}_l)} \mathbf{v}_j \quad \in \mathbb{R}^{d}$$



Linear attention (Katharopoulos et al. 2020):

$$V_i' = \frac{\phi(Q_i)^T \sum_{j=1}^{i} \phi(K_j) V_j^T}{\phi(Q_i)^T \sum_{j=1}^{i} \phi(K_j)}. \qquad \phi(x) = \text{elu}(x) + 1,$$

[1] Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention. PLMR, 2020. Angelos Katharopoulos.

TransNormer

- Kernel-based linear attention suffer from unbounded gradients
  - Normalization of the denominator term can lead to unstable values
- NormAttention

method	Relative Standard Deviation
$1 + elu$ (Katharopoulos et al., 2020)	0.58
Performer(Choromanski et al., 2020)	0.47
Vanilla(Vaswani et al., 2017)	0.25
NORMATTENTION	0.20

Our proposed solution is simple yet effective. Given a linear attention, the attention without scaling can be formulated as:

$$\mathbf{O} = \mathbf{Q}(\mathbf{K}^\top \mathbf{V}). \tag{13}$$

We empirically find that we can apply an arbitrary normalization on this attention to bound it, which leads to our NORMATTENTION as:

$$\mathbf{O}_{\text{norm}} = \text{XNorm}(\mathbf{Q}(\mathbf{K}^\top \mathbf{V})), \tag{14}$$

where the XNorm can be Layernorm(Ba et al., 2016) or RMSNorm (Zhang and Sennrich, 2019) and *etc*. We use the RMSNorm in our experiments as it is slightly faster than other options.

[1] The devil in linear transformer. 2022, Zhen Qin.

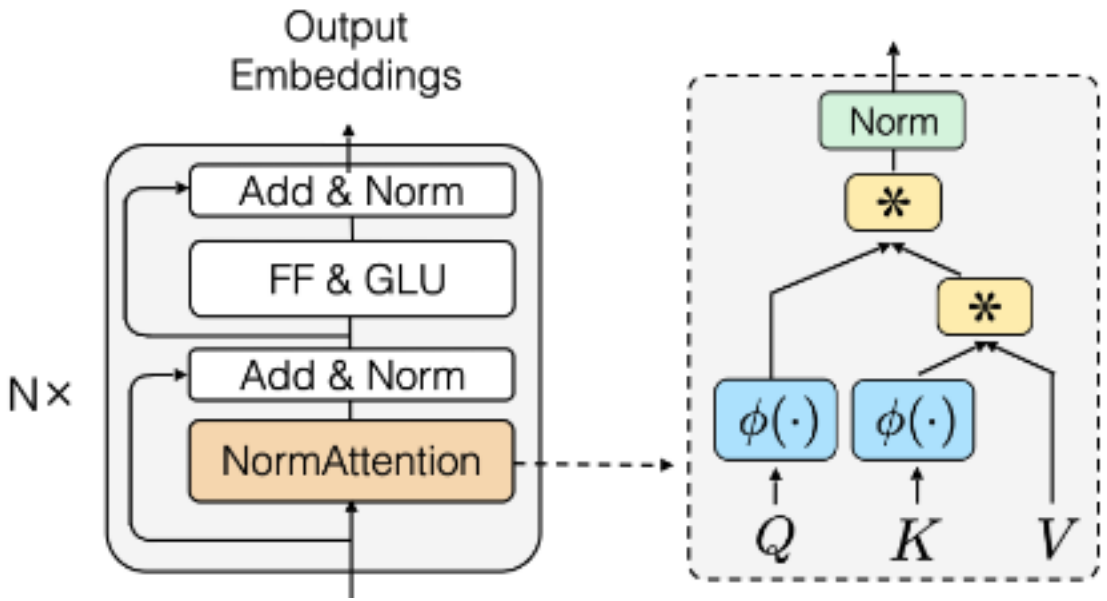
Causal Modeling

We empirically find that we can apply an arbitrary normalization on this attention to bound it, which leads to our NORMATTENTION as:

$$\mathbf{O}_{\text{norm}} = \text{XNorm}(\mathbf{Q}(\mathbf{K}^\top \mathbf{V})), \tag{14}$$

$$\text{Parallel training}: \quad \mathbf{O} = (\mathbf{Q}\mathbf{K}^\top \odot \mathbf{M})\mathbf{V} \quad \in \mathbb{R}^{L\times d}$$

$$\text{Iterative inference}: \quad \mathbf{o_t} = \sum_{j=1}^{t} (\mathbf{q}_t^\top \mathbf{k}_j)\mathbf{v}_j \quad \in \mathbb{R}^{d}$$



Linear Attention = Linear RNN + Matrix-valued Hidden States

$$\mathbf{o}_t = \sum_{j=1}^{t} (\mathbf{q}_t^\top \mathbf{k}_j)\mathbf{v}_j$$

$$= \sum_{j=1}^{t} \mathbf{v}_j(\mathbf{k}_j^\top \mathbf{q}_t) \quad \mathbf{k}_j^\top \mathbf{q}_t = \mathbf{q}_t^\top \mathbf{k}_j \in \mathbb{R}$$

$$= \underbrace{(\sum_{j=1}^{t} \mathbf{v}_j \mathbf{k}_j^\top)}_{\mathbf{S}_t \in \mathbb{R}^{d \times d}} \mathbf{q}_t \qquad \text{By associativity}$$

The output cumulative sum of the timestamp attention weights

Linear Attention = Linear RNN + Matrix-valued Hidden States

Let $\mathbf{S}_t = \sum_{j=1}^{t} \mathbf{v}_j \mathbf{k}_j^\top \in \mathbb{R}^{d \times d}$ be the matrix-valued hidden state, then:

$$\mathbf{S}_t = \mathbf{S}_{t-1} + \mathbf{v}_t \mathbf{k}_t^\top \quad \in \mathbb{R}^{d \times d}$$

$$\mathbf{o}_t = \mathbf{S}_t \mathbf{q}_t \quad \in \mathbb{R}^{d}$$

- Linear attention implements elementwise linear recurrence.
- Linear attention has a matrix-valued hidden state, where state size($d \times d$)is much bigger than the state size d of traditional RNN/LSTM
  - Big memory capacity brings big space complexity
  - Requires intermediate storage of $n \times d \times d$ matrix where $d^2 \gg n$

Challenges of Linear Attention for Parallelism

$$\mathbf{O} = (\mathbf{Q}\mathbf{K}^\top \odot \mathbf{M})\mathbf{V} \in \mathbb{R}^{L\times d}$$

Still quadratic in sequence length

➢ Convert from Parallel (Matrix) Form to Recurrent Form

$$\mathbf{S}_t = \mathbf{S}_{t-1} + \mathbf{v}_t\mathbf{k}_t^\top \quad \in \mathbb{R}^{d\times d}$$

$$\mathbf{o}_t = \mathbf{S}_t\mathbf{q}_t \quad \in \mathbb{R}^{d}$$

- Low space complexity: as longa as the cumulative state of the current position is maintained, there is no need to store all KVs of the entire sequence.
- Serial computing limits parallelization.
- Poor GPU utilization
  - Lack of matrix multiplication operations, cumsum operations cause extra computation.
  - Outer product, matrix-vector multiplication, and element-wise sum.

Hardware-efficient Training with Chunkwise Parallel Form

- Sequence of length L divided into L/C chunks of size C
- Compute only the last hidden state of each chunk
- Compute the output from two parts
  - Historical context: using recurrent form
  - Local context: using parallel form – Chunk 内部
- $C = 1$ is recurrent form, $C = L$ is parallel form

Notation:

$$\mathbf{S}_{[i]} := \mathbf{S}_{iC} \in \mathbb{R}^{d \times d} \qquad \text{(Chunk-level hidden state)}$$

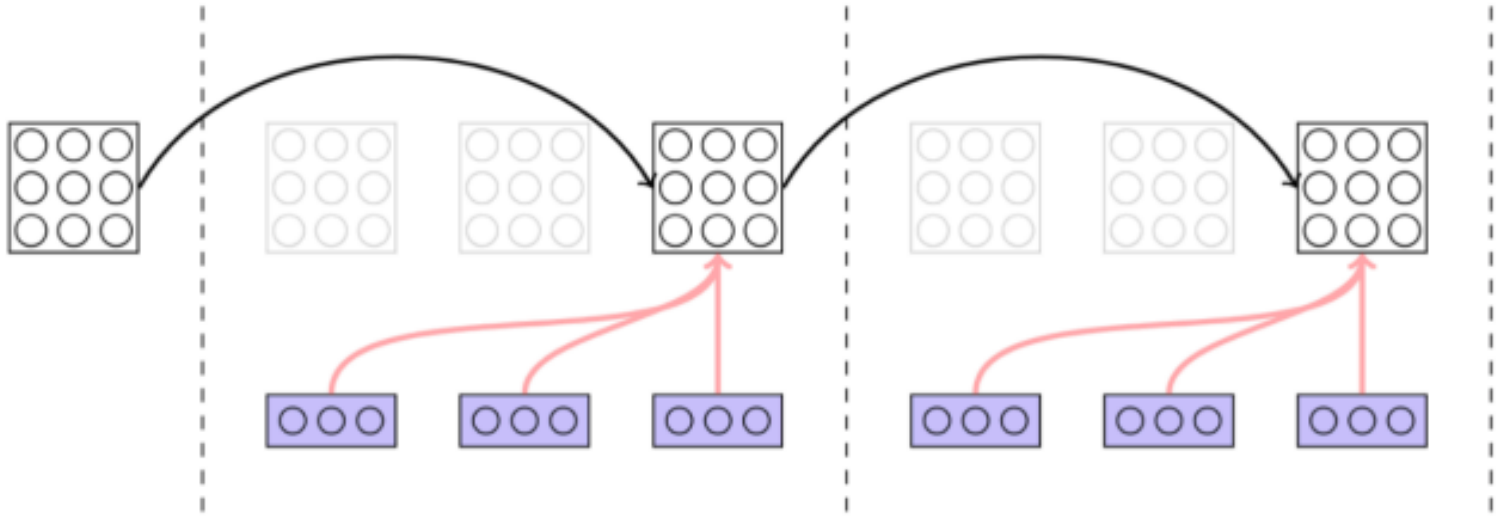
$$\square_{[i]} = \square_{iC+1:(i+1)C} \in \mathbb{R}^{C \times d} \qquad \text{(Matrix block for chunk } i\text{)}$$

$$\text{for } \square \in \{\mathbf{Q}, \mathbf{K}, \mathbf{V}, \mathbf{O}\}$$

Chunkwise Parallel Form: Hidden State Update

Sequential Chunk-Level State Passing:

$$\boldsymbol{S}_{[i+1]} = \boldsymbol{S}_{[i]} + \boldsymbol{V}_{[i]}^{\top}\boldsymbol{K}_{[i]}$$

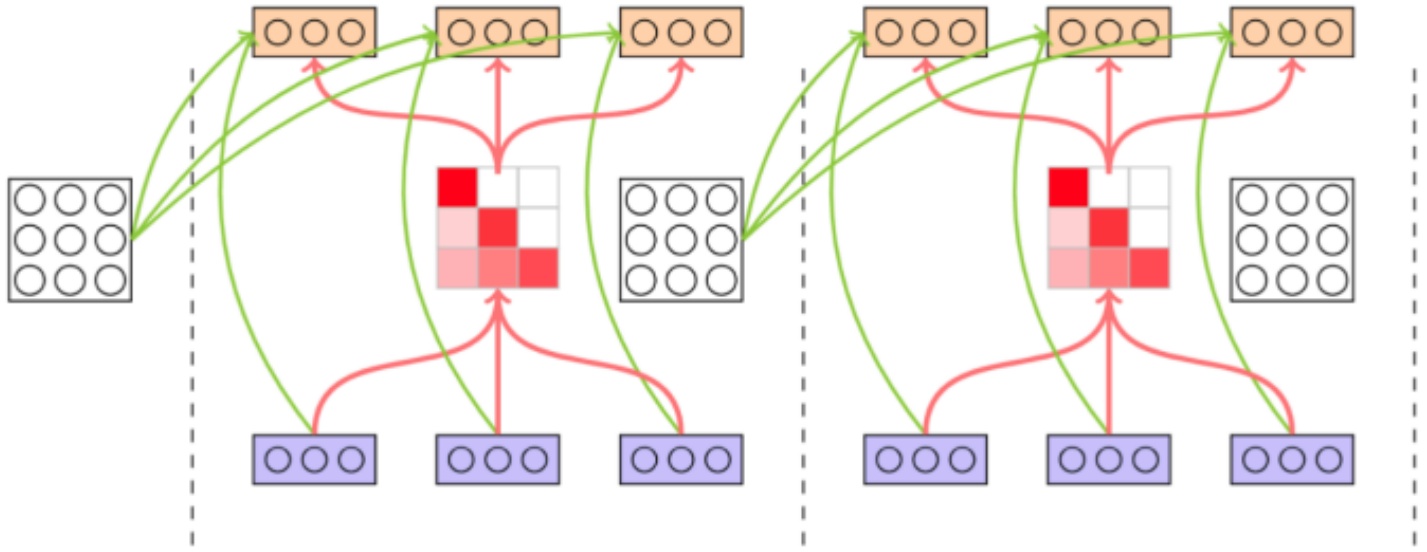


$$\mathbf{S}_{[t+1]} = \underbrace{\mathbf{S}_{[t]}}_{\mathbb{R}^{d\times d}} + \underbrace{\mathbf{V}_{[t]}^{\top}}_{\mathbb{R}^{d\times C}}\underbrace{\mathbf{K}_{[t]}}_{\mathbb{R}^{C\times d}} \quad \in \mathbb{R}^{d\times d} \qquad \text{(Matrix Form)}$$

Chunkwise Parallel Form: Parallel Output Computation

Parallel Output Computation:

$$\mathbf{O}_{[i]} = \mathbf{Q}_{[i]}\mathbf{S}_{[i]}^{\top} + \left(\mathbf{Q}_{[i]}\mathbf{K}_{[i]}^{\top} \odot \mathbf{M}\right)\mathbf{V}_{[i]}$$



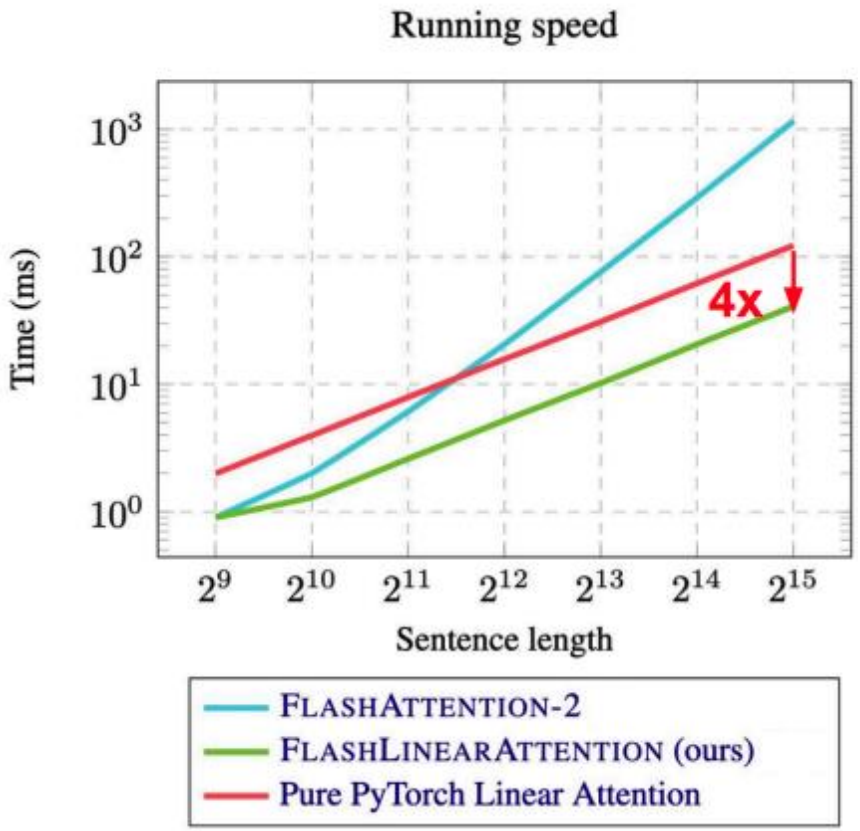
$$\mathbf{O}_{[t]} = \underbrace{\mathbf{Q}_{[t]}}_{\mathbb{R}^{C\times d}}\underbrace{\mathbf{S}_{[t]}^{\top}}_{\mathbb{R}^{d\times d}} + \underbrace{\left(\mathbf{Q}_{[t]}\mathbf{K}_{[t]}^{\top} \odot \mathbf{M}\right)}_{\mathbb{R}^{C\times C}}\underbrace{\mathbf{V}_{[t]}}_{\mathbb{R}^{C\times d}} \in \mathbb{R}^{C\times d} \qquad \text{(Matrix Form)}$$

Chunkwise Parallel Form

- Total complexity: $\mathcal{O}(Ld^2 + LdC)$
- C is set to $\{64, 128, 256\}$ in practice
- The de facto standard for train0 modern linear attention models

- The first generation of linear attention focuses on hardware efficient, parallelised approaches

Flash Linear Attention



Flash linear attention library provides hardware-efficient implementation of various linear attention models:

- RetNet
- GLA
- HGRN2
- RWKV6/RWKV7
- GSA
- Mamba2
- DeltaNet
- Gated DeltaNet
- ...

https://github.com/fla-org/flash-linear-attention

Linear Attention Performance

$$\mathbf{S}_t = \mathbf{S}_{t-1} + \mathbf{v}_t \mathbf{k}_t^\top \quad \in \mathbb{R}^{d \times d}$$

$$\mathbf{o}_t = \mathbf{S}_t \mathbf{q}_t \quad \in \mathbb{R}^{d}$$

- **Instability:** the hidden state value could explode due to cumulative sum without decay
- **Poor performance:** vanilla linear attention models significantly underperform Transformers in language modeling perplexity (Inattentive)
  - Essentially the low rank linear approximation of SoftMax attention

- The second generation of linear attention focuses on ways to improve performance.

Linear Attention with Decay

$$\mathbf{S}_t = \gamma \mathbf{S}_{t-1} + \mathbf{v}_t \mathbf{k}_t^\top \qquad \in \mathbb{R}^{d \times d}$$

$$\mathbf{o}_t = \mathbf{S}_t \mathbf{q}_t \qquad \in \mathbb{R}^{d}$$

- γ is an exponential decay factor $0 < \gamma < 1$
- **Works well in practice:** RetNet (Sun et al. 2023), Lightning Attention (Qin et al. 2024b)
- **Lacking selectivity:** a potential issue.
- Decay harms long-dependency memory which decreases retrieval performance
  - Lightning attention attempts to reduce the decay value as the number of layers decreases from low to high, with the top layer entirely vanilla linear attention
  - Bottom layer vanilla SoftMax attention, top layer linear attention

Linear Attention with Decay

$$\mathbf{S}_t = \gamma_t \mathbf{S}_{t-1} + \mathbf{v}_t \mathbf{k}_t^\top \qquad \in \mathbb{R}^{d \times d}$$

$$\mathbf{o}_t = \mathbf{S}_t \mathbf{q}_t \qquad \in \mathbb{R}^{d}$$

- γ_t is data-dependent, dynamic decay term
- **Selectivity:** Dynamically control the forgetting or memorization
  - Essentially the forgetting mechanism in LSTM
- **Examples:** Mamba2, mLSTM

Linear Attention with Decay

$$\mathbf{S}_t = \mathbf{G}_t \odot \mathbf{S}_{t-1} + \mathbf{v}_t \mathbf{k}_t^\top \qquad \in \mathbb{R}^{d \times d}$$

$$\mathbf{o}_t = \mathbf{S}_t \mathbf{q}_t \qquad \in \mathbb{R}^{d}$$

- $\boldsymbol{G}_t \in \mathbb{R}^{d \times d}$ is a fine-grained data-dependent gate matrix
- $\boldsymbol{G}_t = \beta_t \alpha_t^{\mathrm{T}}$ enables hardware-efficient training with matrix-multiply form, where $\beta_t \alpha_t \in \mathbb{R}^d$
- A simpler choice: $\beta_t = \mathbf{1}$
  - **Examples:** GLA (Yang et al, 2023), RWKV6, MetaLA (Chou et al, 2024), HGRN2 (Qin et al, 2024a), GSA (Zhang et al. 2024)

Linear Attention: a Fast Weight Programming Perspective

The hidden state matrix $\mathbf{S}_t$ is a fast weight matrix that is updated at each timestep:

$$\mathbf{S}_t = \mathbf{S}_{t-1} + \mathbf{v}_t \mathbf{k}_t^\top$$

The fast weight matrix is used to map inputs $\mathbf{q}_t$ into outputs $\mathbf{o}_t$:

$$\mathbf{o}_t = \mathbf{S}_t \mathbf{q}_t$$

$$Q = xW_Q,$$
$$K = xW_K,$$
$$V = xW_V,$$

Linear projections are "slow weight" matrices (independent of t)

- The 3rd generation of linear attention mainly studies the choice of fast weight matrix update rule from the optimisation point of view

The choice of update rule

- Hebbian update rule: $\mathbf{S}_t = \mathbf{S}_{t-1} + \mathbf{v}_t \mathbf{k}_t^\top$
- Delta rule: $\mathbf{S}_t = \mathbf{S}_{t-1} - \beta_t \left(\mathbf{S}_{t-1}\mathbf{k}_t - \mathbf{v}_t\right) \mathbf{k}_t^\top$

Both Hebbian and delta update rules are regarded as optimizing online learning objectives via single step of SGD

The objective predicts the target value $\mathbf{v}_t$ by transforming the key $\mathbf{k}_t$ with $\mathbf{S}$.

$$\mathcal{L}_t(\mathbf{S}) = -\langle \mathbf{S}\mathbf{k}_t, \mathbf{v}_t \rangle$$

Performing a single step of SGD:

$$\begin{aligned} \mathbf{S}_t &= \mathbf{S}_{t-1} - \beta_t \nabla \mathcal{L}_t(\mathbf{S}_{t-1}) \\ &= \mathbf{S}_{t-1} + \beta_t \mathbf{v}_t \mathbf{k}_t^\top \end{aligned}$$

- Enforce SK and v to be in the same direction
- Encourage the learning of a very large norm
- Which might cause gradient explosion

The choice of update rule

The objective predicts the target value $\mathbf{v}_t$ by transforming the key $\mathbf{k}_t$ with $\mathbf{S}$.

$$\mathcal{L}_t(\mathbf{S}) = -\langle \mathbf{S}\mathbf{k}_t, \mathbf{v}_t \rangle$$

Performing a single step of SGD:

$$\begin{aligned} \mathbf{S}_t &= \mathbf{S}_{t-1} - \beta_t \nabla \mathcal{L}_t(\mathbf{S}_{t-1}) \\ &= \mathbf{S}_{t-1} + \beta_t \mathbf{v}_t \mathbf{k}_t^\top \end{aligned}$$

- Learning rate $\beta_t = 1$ recovers vanilla linear attention.
- Mamba2's update rule $\mathbf{S}_t = \alpha_t \mathbf{S}_{t-1} + \mathbf{v}_t \mathbf{k}_t^\top$ can be interpreted as online SGD with weight decay α_t.

The choice of update rule

$$\mathcal{L}_t(\mathbf{S}) = -\langle \mathbf{S}\mathbf{k}_t, \mathbf{v}_t \rangle$$

Online regression loss is better for predicting $\mathbf{v}_t$ from $\mathbf{k}_t$ and $\mathbf{S}_{t-1}$.

$$\mathcal{L}_t(\mathbf{S}) = \frac{1}{2}\|\mathbf{S}\mathbf{k}_t - \mathbf{v}_t\|^2$$

Performing a single step of SGD:

$$\begin{aligned}\mathbf{S}_t &= \mathbf{S}_{t-1} - \beta_t \nabla \mathcal{L}_t(\mathbf{S}_{t-1}) \\ &= \mathbf{S}_{t-1} - \beta_t \left(\mathbf{S}_{t-1}\mathbf{k}_t - \mathbf{v}_t\right)\mathbf{k}_t^\top \\ &= \mathbf{S}_{t-1}\left(\mathbf{I} - \beta_t \mathbf{k}_t \mathbf{k}_t^\top\right) + \beta_t \mathbf{v}_t \mathbf{k}_t^\top\end{aligned}$$

When $\beta_t \in (0,1)$, DeltaNet update rule [1-2]

from the core principle of updating weights based on the "delta" (difference) between the prediction $\mathbf{S}_{t-1}\boldsymbol{k}_t$ and the target $\boldsymbol{v}_t$, which has been shown better memory capacity [28, 85, 56, 101]. This process can also be regarded as optimizing an online regression loss using a single step of SGD,[2]

$$\mathcal{L}_t(\mathbf{S}) = \frac{1}{2}\|\mathbf{S}\boldsymbol{k}_t - \boldsymbol{v}_t\|^2, \quad \mathbf{S}_t = \mathbf{S}_{t-1} - \beta_t \nabla_{\mathbf{S}_{t-1}} \mathcal{L}_t(\mathbf{S}_{t-1}) = \mathbf{S}_{t-1} - \beta_t(\mathbf{S}_{t-1}\boldsymbol{k}_t - \boldsymbol{v}_t)\boldsymbol{k}_t^\top,$$

which has been discussed in several recent works [110, 58, 125, 10]. In contrast, linear transformers

[1]It is possible in theory to use parallel scan [13] to parallelize the recurrent form, which would enable the computations to be performed in $O(\log L)$ steps and $O(Ld^2)$ FLOPs. However, this approach requires materializing the 2D hidden state for each time step, which would incur significant memory I/O cost unless the state size is small enough such that materialization can happen in faster memory (i.e., as in Mamba [31]).

[2]This perspective was discussed as early as Widrow et al. [121], which later became known as the Least Mean Squares (LMS) algorithm and has found widespread applications in signal processing [97].

[1] Linear Transformers Are Secretly Fast Weight Programmer, 2021
[2] Parallelizing Linear Transformers with the Delta Rule over Sequence Length. 2024

DeltaNet optimizes a regression loss via SGD

Performing a single step of SGD:

$$
\begin{aligned}
\mathbf{S}_t &= \mathbf{S}_{t-1} - \beta_t \nabla \mathcal{L}_t(\mathbf{S}_{t-1}) \\
&= \mathbf{S}_{t-1} - \beta_t \left(\mathbf{S}_{t-1}\mathbf{k}_t - \mathbf{v}_t\right)\mathbf{k}_t^\top \\
&= \mathbf{S}_{t-1}\left(\mathbf{I} - \beta_t \mathbf{k}_t \mathbf{k}_t^\top\right) + \beta_t \mathbf{v}_t \mathbf{k}_t^\top
\end{aligned}
$$

Nondiagonal transition matrix

- Structured matrix could exploit high-efficient parallel chunk-wise training
- More expressive than Mamba2, beyond TC^0 complexity(Mamba2, Transformer)

Mamba2's update rule $\mathbf{S}_t = \alpha_t \mathbf{S}_{t-1} + \mathbf{v}_t \mathbf{k}_t^\top$

DeltaNet Issues

Despite strong performance on synthetic benchmarks like MQAR and MAD, DeltaNet underperforms on real-world language modeling tasks compared to models like Mamba2

Model	Wiki. ppl ↓	LMB. ppl ↓	LMB. acc ↑	PIQA acc ↑	Hella. acc_n ↑	Wino. acc ↑	ARC-e acc ↑	ARC-c acc_n ↑	SIQA acc ↑	BoolQ acc ↑	Avg.
Mamba	17.92	15.06	43.98	71.32	52.91	52.95	69.52	35.40	37.76	61.13	53.12
Mamba2	16.56	12.56	45.66	71.87	55.67	55.24	72.47	37.88	40.20	60.13	54.89
DeltaNet	17.71	16.88	42.46	70.72	50.93	53.35	68.47	35.66	40.22	55.29	52.14

Table: Performance comparison on language modeling and zero-shot common-sense reasoning for 1.3B parameter models that are trained for 100B tokens.

Decay is crucial for forgetting irrelevant information

Gated DeltaNet

Gated DeltaNet combines the delta update rule in DeltaNet and the gated update rule in Mamba2:

$$\mathbf{S}_t = \mathbf{S}_{t-1}\left(\alpha_t(\mathbf{I} - \beta_t \mathbf{k}_t \mathbf{k}_t^\top)\right) + \beta_t \mathbf{v}_t \mathbf{k}_t^\top$$

- $\alpha_t \in (0, 1)$ is parameterized the same as Mamba2.
- When $\alpha_t = 1$, Gated DeltaNet is equivalent to DeltaNet.
- When $\alpha_t = 0$, Gated DeltaNet clears the entire memory.
- Gated DeltaNet can be interpreted as optimizing the online regression loss with weight decay.

Gated DeltaNet In-context Retrieval on Synthetic Data

	S-NIAH-1 (pass-key retrieval)				S-NIAH-2 (number in haystack)				S-NIAH-3 (word in haystack)		
Model	1K	2K	4K	8K	1K	2K	4K	8K	1K	2K	4K
DeltaNet	97.4	96.8	**99.0**	**98.8**	98.4	45.6	18.6	14.4	85.2	47.0	22.4
Mamba2	**99.2**	**98.8**	65.4	30.4	99.4	98.8	56.2	17.0	64.4	47.6	4.6
Gated DeltaNet	98.4	88.4	91.4	91.8	**100.0**	**99.8**	**92.2**	**29.6**	**86.6**	**84.2**	**27.6**

Table 3: Performance comparison on S-NIAH benchmark suite.

In-context retrieval on synthetic data Table 3 shows the results on Single Needle-In-A-Haystack (S-NIAH) benchmark suite from RULER (Hsieh et al., 2024). In the simplest S-NIAH-1 setting with synthetic inputs, DeltaNet achieves near-perfect performance across all sequence lengths, benefiting from its delta update rule which is specifically advantageous for in-context recall (§3.1). In contrast, Gated DeltaNet shows slightly lower retrieval accuracy since its gating mechanism discards information, compromising perfect memory retention, while Mamba2's performance degrades significantly beyond 2K sequences.

However, retrieval from memory depends on not only retention but also the ability to "forget": given fixed state size, lack of memory clearance leads to memory collision when the state becomes saturated - multiple pieces of information become superimposed, making them indistinguishable. This becomes evident in NIAH-2 and NIAH-3 where needles are grounded in real-world text data:

Go Beyond Online Linear Regression Objective

- The optimization objective assumes linear relationships in historical data dependencies
- Actual GenAI tasks involve complex, nonlinear dependences, which is hard for linear regression loss to capture the rich patterns

TTT (Sun et al. 2024) extends this to a nonlinear regression loss:

$$\mathcal{L}_t(\mathbf{S}) = \frac{1}{2}\|f_{\mathbf{S}}(\mathbf{k}_t) - \mathbf{v}_t\|^2$$

where $f_{\mathbf{S}}$ is a nonlinear transformation parameterized by $\mathbf{S}$.

- TTT-linear: $f_{\mathbf{S}}(x) = \mathrm{LN}(\mathbf{S}x) + x$ where LN is layer normalization
- TTT-MLP: $f_{\mathbf{S}}(x) = \mathrm{LN}(\mathrm{MLP}_{\mathbf{S}}(x)) + x$ where $\mathbf{S}$ is MLP weight matrix

Go Beyond Online Linear Regression Objective

- **Workaround:** Use mini-batch updates by accumulating gradients over *B* tokens before updating S (i.e., hybrid intra-chunk linear + inter-chunk nonlinear)
- **Titans** (Behrouz, Zhong, and Mirrokni 2024) improves TTT by incorporating momentum and weight decay into the mini-batch SGD update

RWKV 7

RWKV-7 as a meta-in-context learner (Nov 26, 2024)

Here I will explain the principle of RWKV-7 "Goose", as a meta-in-context learner.

Giving two sequences of vectors $\{k_t\}$ and $\{v_t\}$, RWKV-7 will test-time-train an internal model $v \approx kS^\top$ via in-context gradient descent w.r.t the L2 loss $\mathcal{L} = \frac{1}{2}\| v - kS^\top \|^2$.

The gradient is:

$$\frac{\partial \mathcal{L}}{\partial S} = Sk^\top k - v^\top k$$

The gradient descent formula (with weight decay w_t and learning rate η_t) is:

$$S_t = S_{t-1}\,\mathrm{diag}(w_t) - (S_{t-1}k_t^\top k_t - v_t^\top k_t)\,\mathrm{diag}(\eta_t)$$

which equals:

$$S_t = S_{t-1}\left(\mathrm{diag}(w_t) - k_t^\top k_t\,\mathrm{diag}(\eta_t)\right) + v_t^\top k_t\,\mathrm{diag}(\eta_t)$$

In RWKV-7 I use the generalized formula:

$$S_t = S_{t-1}(\mathrm{diag}(w_t) + \mathbf{a}_t^\top \mathbf{b}_t) + \mathbf{v}_t^\top \mathbf{k}_t$$

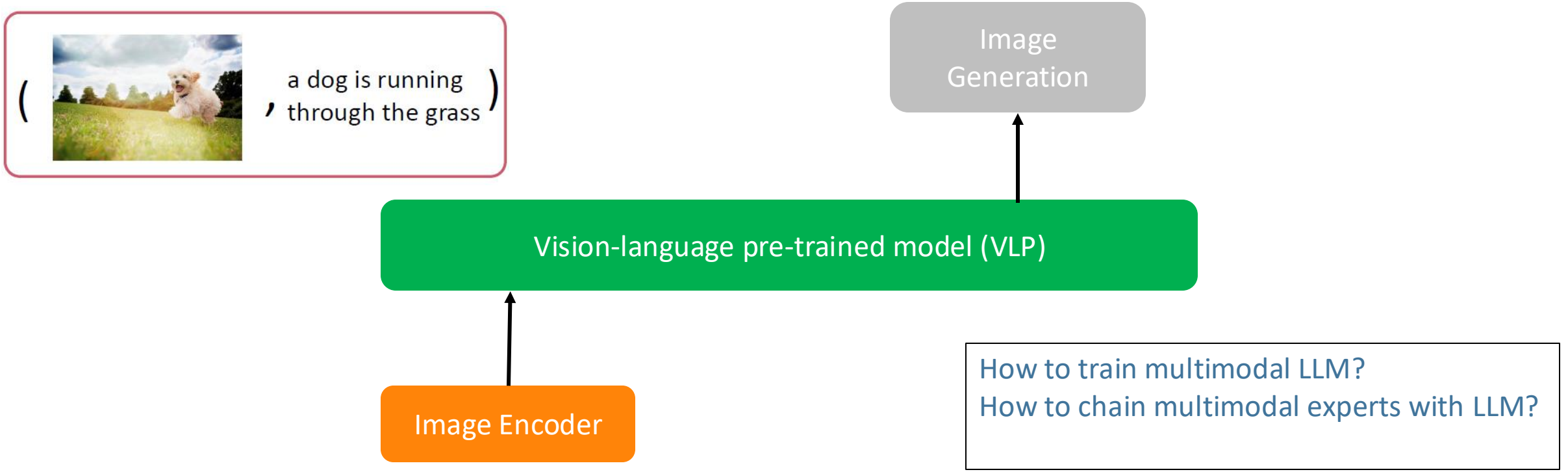
where a reasonable choice of initial values is $\mathbf{a} = -k$, $\mathbf{b} = k \cdot \eta$, $\mathbf{v} = v$, $\mathbf{k} = k \cdot \eta$.

RWKV-7 uses $\{k_t, v_t\}$ to test-time-train an internal model and uses $\{r_t\}$ as input for this model. It overcomes the TC^0 limitation of QKV-softmax-attention transformers (and RWKV-6, Mamba, Mamba-2, xLSTM, GLA, ...), while still being efficiently trainable on GPUs.

RWKV-7 “Goose” with Expressive Dynamic State Evolution. 2024.

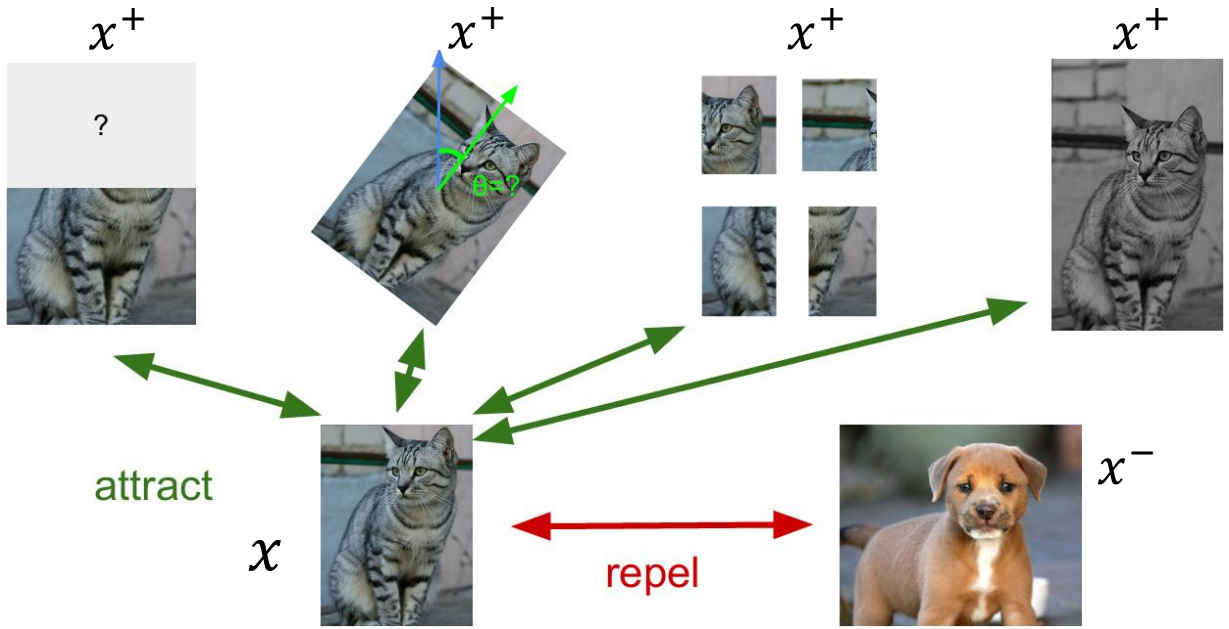
Large Multimodal Models: Image-to-Text Models

- Model Architectures
  - Pre-trained Image encoder and language models
  - Trainable modules to connect to two modalities



2.2 Contrastive Learning of Visual Representations

- Recall Self-supervised Learning
  - We call it CPC (sequence-level contrastive learning)



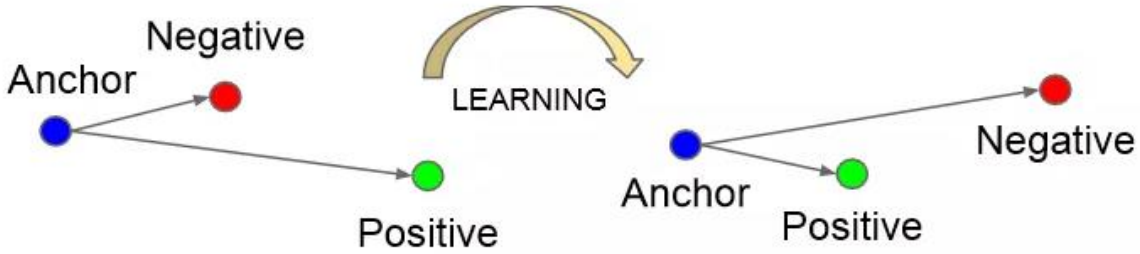
x reference
x^+ positive
x^- negative

What we want:

$$\text{score}(f(x), f(x^+)) >> \text{score}(f(x), f(x^-))$$

x: reference sample; x^+ positive sample; x^- negative sample

Given a chosen score function, we aim to learn an **encoder function** f that yields high score for positive pairs (x, x^+) and low scores for negative pairs (x, x^-).



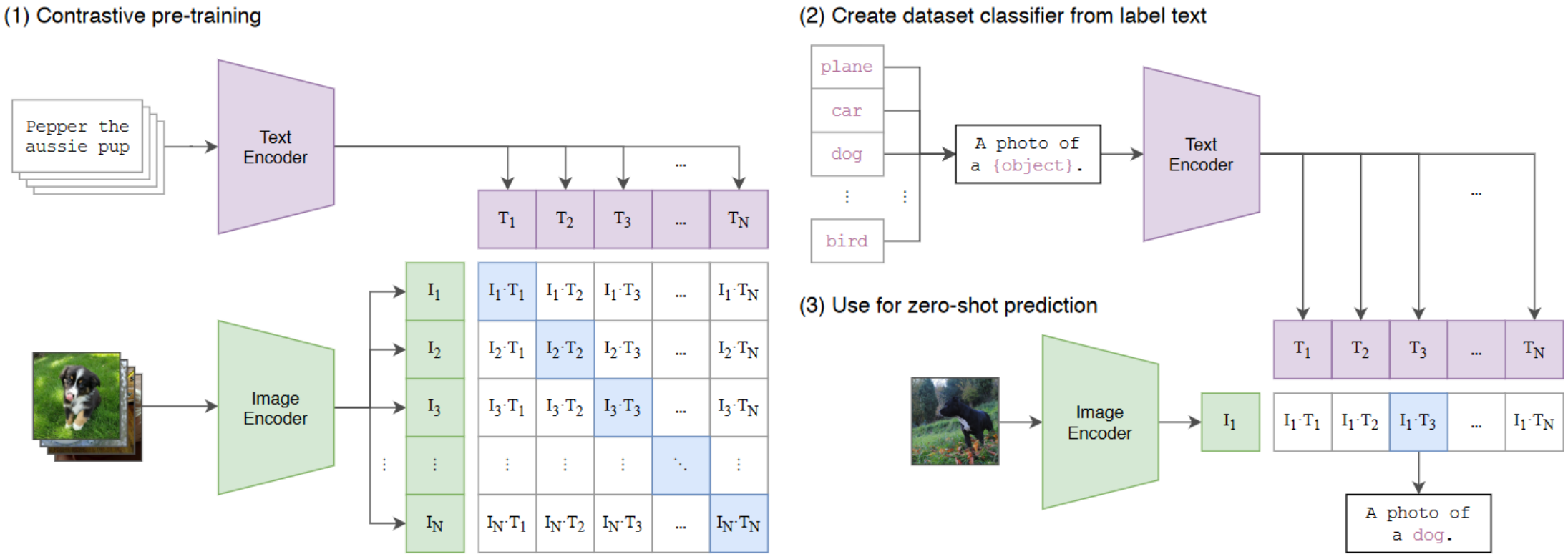
$$L = -\mathbb{E}_X \left[\log \frac{\exp(s(f(x), f(x^+))}{\exp(s(f(x), f(x^+)) + \sum_{j=1}^{N-1} \exp(s(f(x), f(x_j^-))} \right]$$

InfoNCE loss [1] : Cross entropy loss for a N-way softmax classifier

[1] Oord, Aaron van den, Yazhe Li, and Oriol Vinyals. "Representation learning with contrastive predictive coding." arXiv preprint arXiv:1807.03748 (2018)..

2.1 Contrastive language-image Pre-training (CLIP)

- Learning image representations from web-scale noisy text supervision by OpenAI[1]
  - Training: simple *contrastive* learning, and the beauty lies in large-scale pre-training
  - Downstream: *zero-shot* image classification and image-text retrieval
    - Image classification can be reformatted as a retrieval task via considering the semantics behind label names



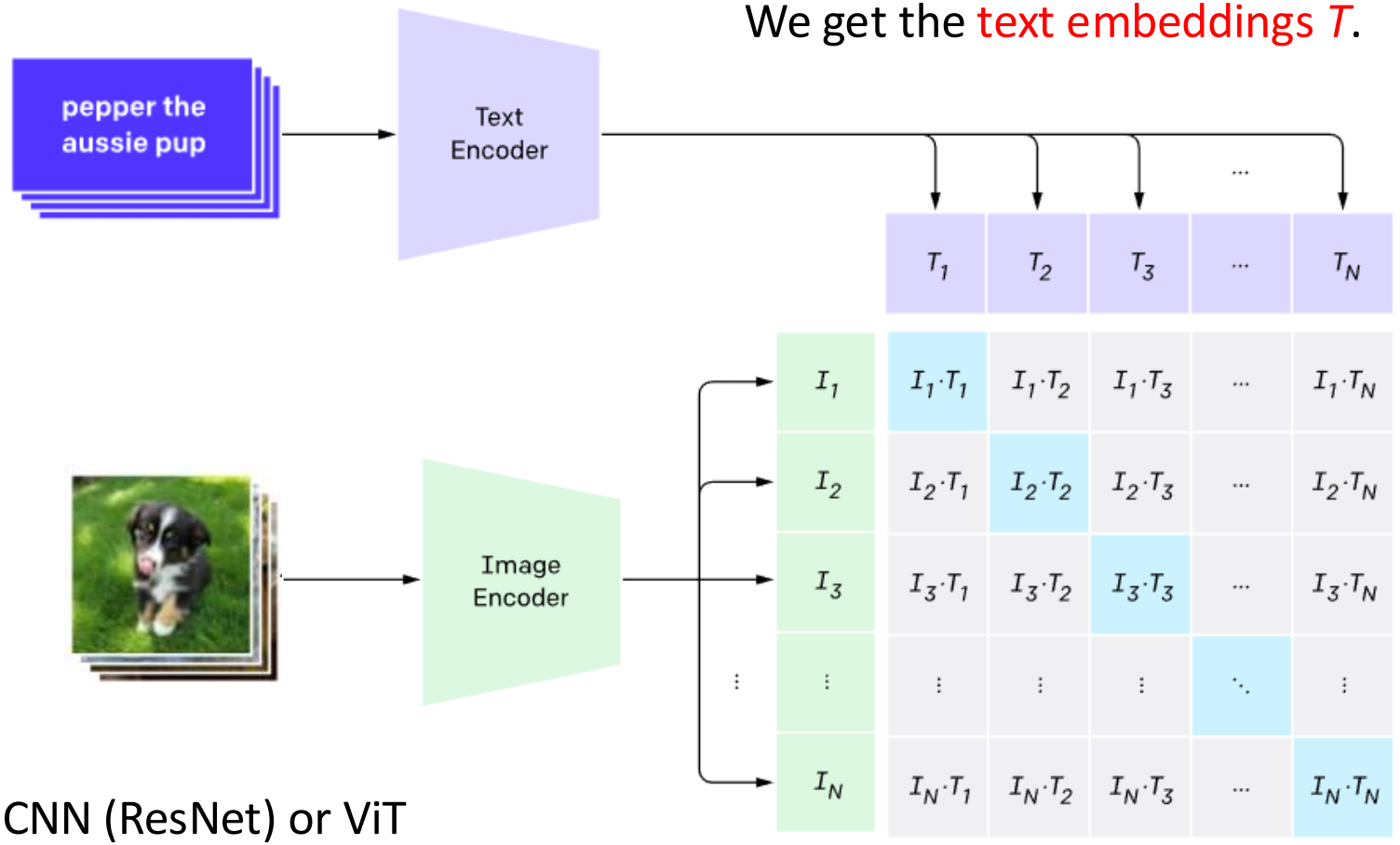
[1] CLIP: Connecting text and images (openai.com)

[2] Radford, Alec, et al. "Learning transferable visual models from natural language supervision." International conference on machine learning. PMLR, 2021.

2.1 Contrastive language-image Pre-training (CLIP)

Text encoder is a transformer language model, e.g. GPT
We get the text embeddings *T*.

Image-text pairs collected from the Internet

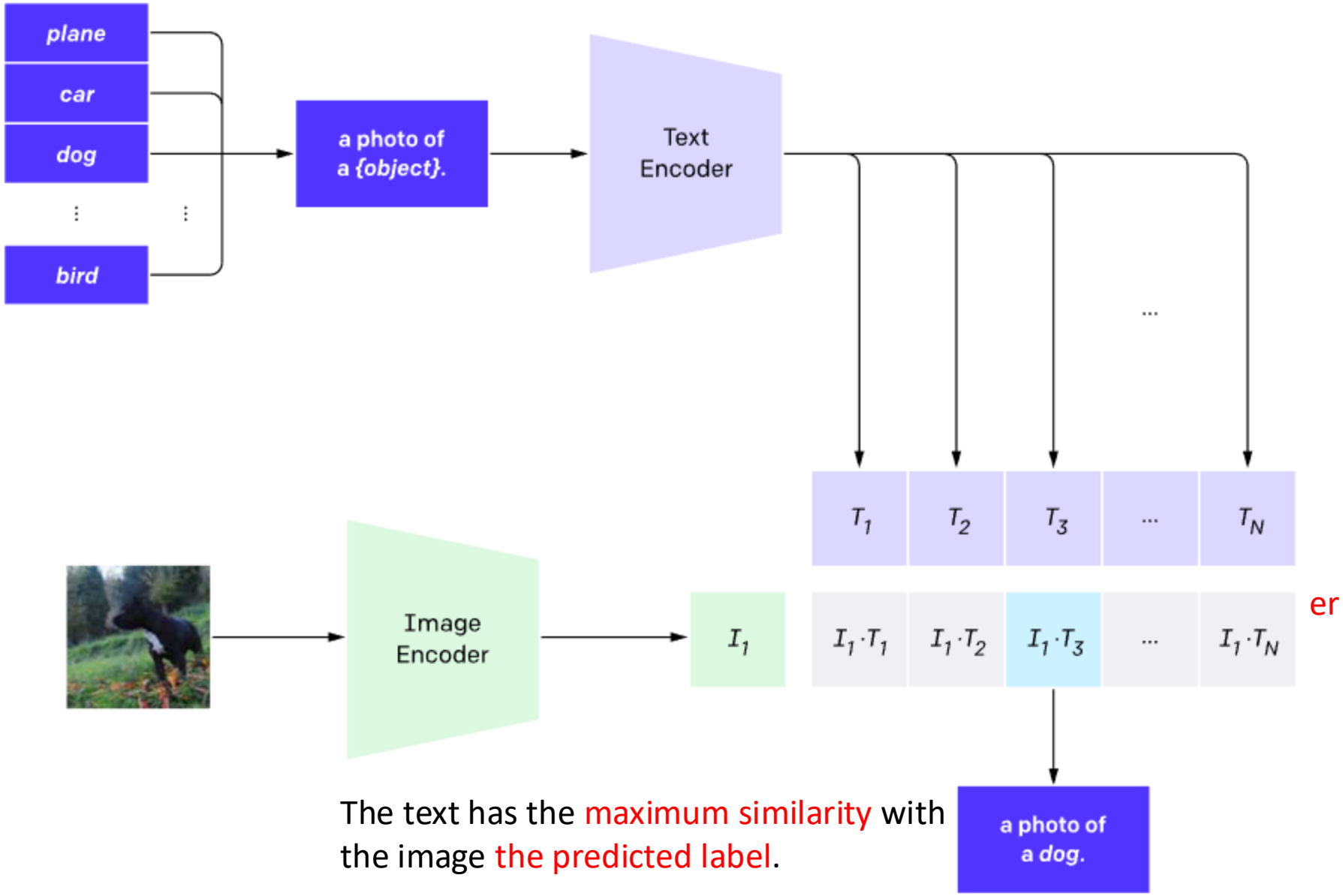


Contrastive matrix,
Maximize the similarity in the diagonal.
Minimize the similarity in other places.

Image encoder is a CNN (ResNet) or ViT
We get the image embeddings *I*.

[1] CLIP: Connecting text and images (openai.com)
[2] Radford, Alec, et al. "Learning transferable visual models from natural language supervision." International conference on machine learning. PMLR, 2021.

2.1 Contrastive language-image Pre-training (CLIP)



Get the a
that we v

er

The text has the maximum similarity with the image the predicted label.

https://openai.com/blog/clip/

2.1 Contrastive language-image Pre-training (CLIP)

- CLIP is an Open-Vocabulary Classifier

Food101

guacamole (90.1%) Ranked 1 out of 101 labels

✓ a photo of **guacamole**, a type of food.

× a photo of **ceviche**, a type of food.

× a photo of **edamame**, a type of food.

× a photo of **tuna tartare**, a type of food.

× a photo of **hummus**, a type of food.

SUN397

television studio (90.2%) Ranked 1 out of 397 labels

✓ a photo of a **television studio**.

× a photo of a **podium indoor**.

× a photo of a **conference room**.

× a photo of a **lecture room**.

× a photo of a **control room**.

Youtube-BB

airplane, person (89.0%) Ranked 1 out of 23 labels

✓ a photo of a **airplane**.

× a photo of a **bird**.

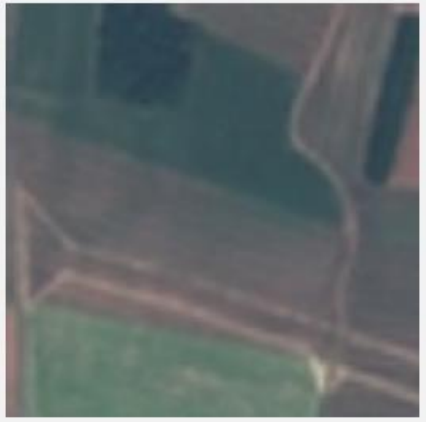
× a photo of a **bear**.

× a photo of a **giraffe**.

× a photo of a **car**.

EuroSAT

annual crop land (46.5%) Ranked 4 out of 10 labels



× a centered satellite photo of **permanent crop land**.

× a centered satellite photo of **pasture land**.

× a centered satellite photo of **highway or road**.

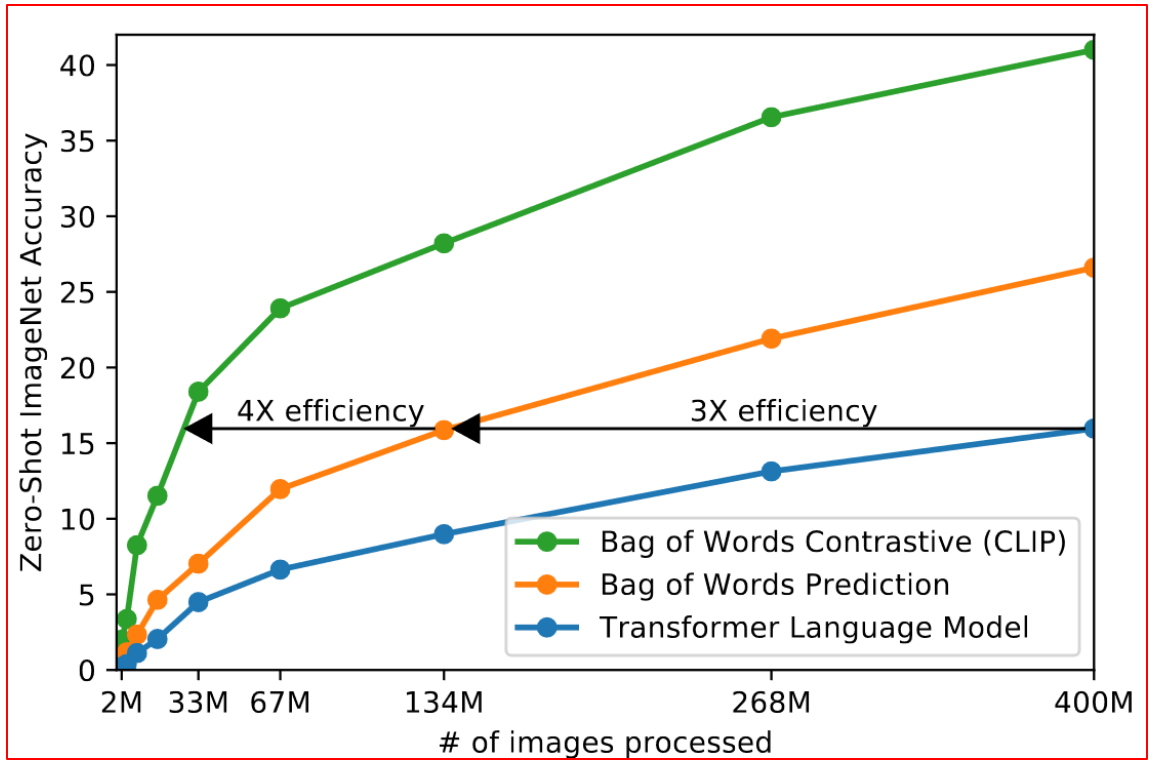
✓ a centered satellite photo of **annual crop land**.

× a centered satellite photo of **brushland or shrubland**.

2.3 Contrastive language-image Pre-training (CLIP)

The idea is simple, and can be dated back to a long time ago

- In the large-scale pre-training era: CLIP[1] and ALIGN [2]
- *Data scale* matters: Models are frequently trained with billions of image-text pairs
- *Batch size* matters: 32k by default; Model size matters



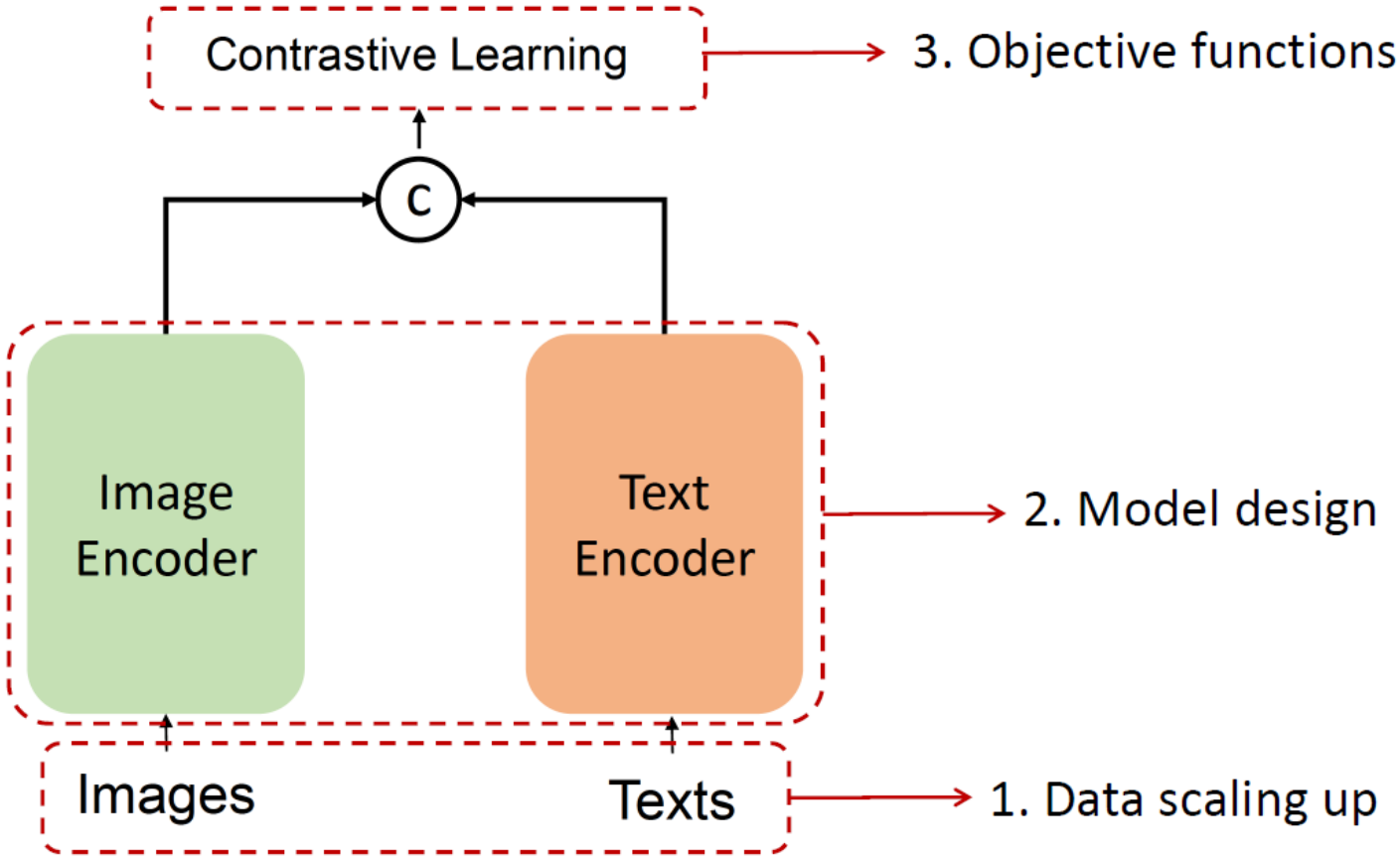
CLIP is much more efficient at zero-shot transfer than other image caption baselines (back in 2021).

- The authors' initial approach was to jointly train an image CNN and text transformer from scratch to predict the caption of an image. However, they encountered difficulties efficiently scaling this method.
- In left figure they show that a 63 million parameter transformer language model, which already uses twice the compute of its ResNet-50 image encoder, learns to recognize ImageNet classes three times slower than a much simpler baseline that predicts a bag-of-words encoding of the same text.
- Recent work in contrastive representation learning for images has found that contrastive objectives can learn better representations than their equivalent predictive objective
- That's the motivation CLIP is an easier proxy task of predicting only which text *as a whole* is paired with which image and not the exact words of that text.

[1] Learning transferable visual models from natural language supervision, ICML 2021
[2] Scaling up visual and vision-language representation learning with noisy text supervision, ICML 2021

2.2 How to improve CLIP

- There are tons of follow-up work and applications on improving CLIP



2.2 How to improve CLIP

- (1) Data Scaling Up

Reproducible scaling laws for CLIP training

- Open large-scale LAION-2B dataset (previously 400m pairs)
- Pre-training OpenCLIP across various scales [1]

	Data	Arch.	ImageNet	VTAB+	COCO
CLIP [55]	WIT-400M	L/14	75.5	55.8	61.1
Ours	LAION-2B	L/14	75.2	54.6	71.1
Ours	LAION-2B	H/14	78.0	56.4	73.4

DataComp: Since scale matters, we need to further scale it up [2]

- In search of the next generation image-text datasets
- Instead of fixing the dataset, and designing different algorithms, the authors proposed to fix the CLIP training method, but select the datasets instead
- Smaller dataset (1B) outperforms LAION-2B for training

Dataset	Dataset size	# samples seen	Architecture	Train compute (MACs)	ImageNet accuracy
OpenAI's WIT [108]	0.4B	13B	ViT-L/14	1.1×10^{21}	75.5
LAION-400M [125, 28]	0.4B	13B	ViT-L/14	1.1×10^{21}	72.8
LAION-2B [126, 28]	2.3B	13B	ViT-L/14	1.1×10^{21}	73.1
LAION-2B [126, 28]	2.3B	34B	ViT-H/14	6.5×10^{21}	78.0
LAION-2B [126, 28]	2.3B	34B	ViT-g/14	9.9×10^{21}	78.5
DataComp-1B (ours)	1.4B	13B	ViT-L/14	1.1×10^{21}	**79.2**

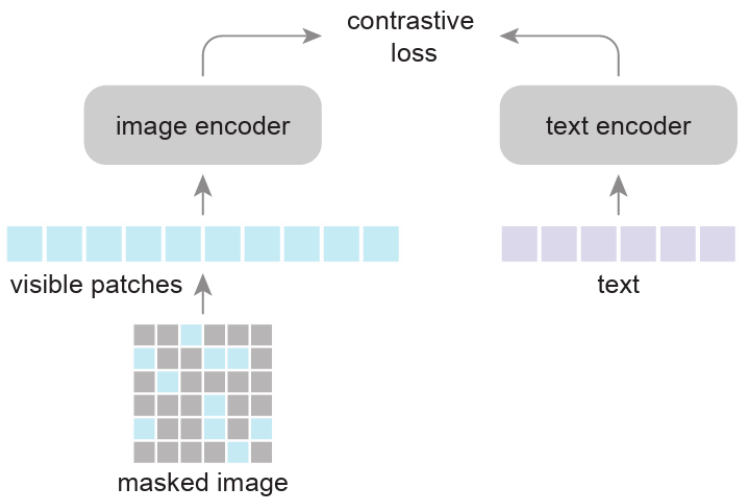
[1] Reproducible scaling laws for contrastive language-image learning, CVPR 2023
[2] Datacomp: In search of the next generation of multimodal datasets, 2023

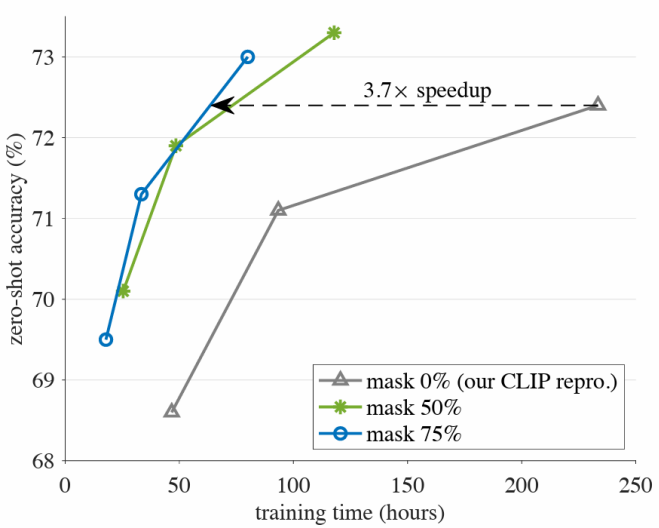
2.5 How to improve CLIP

- (2) Model design from image side

FLIP Scaling CLIP training via masking

- Training: Still use CLIP loss, without incorporating the MIM loss
- Trick: Randomly masking out image patches with a high masking ratio, and only encoding the visible patches
- Results: turns out this does not hurt performance, but improves training efficiency
  - Training is done in 256 TPU-v3 cores, with LAION-400M for 6.4, 12.8, or 32 epochs





[1] Scaling language-image pre-training via masking, CVPR 2023

2.2 How to improve CLIP

- (3) Model design from language side

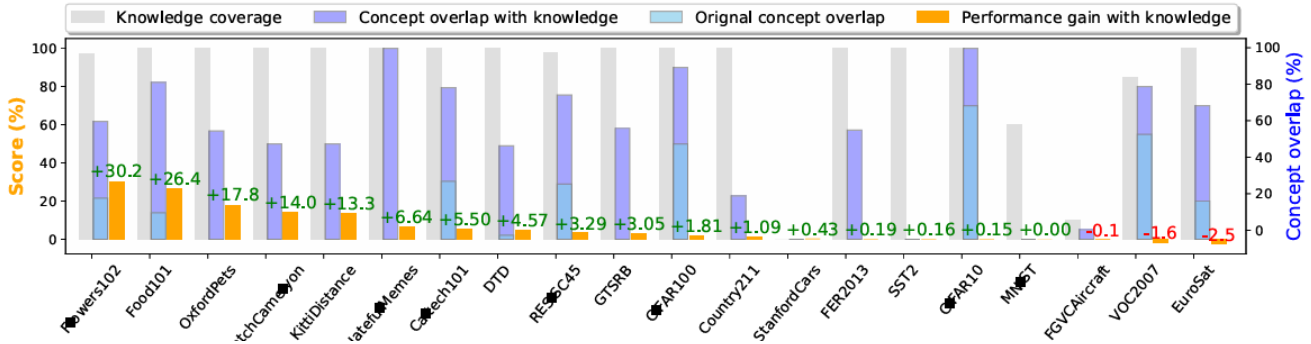
K-Lite External Knowledge

- The Wiki definition of entities (or, the so-called knowledge) can be naturally used together with the original alt-text for contrastive pre-training

Figure 1: Motivating examples: knowledge explains the content of the rare dish concepts.

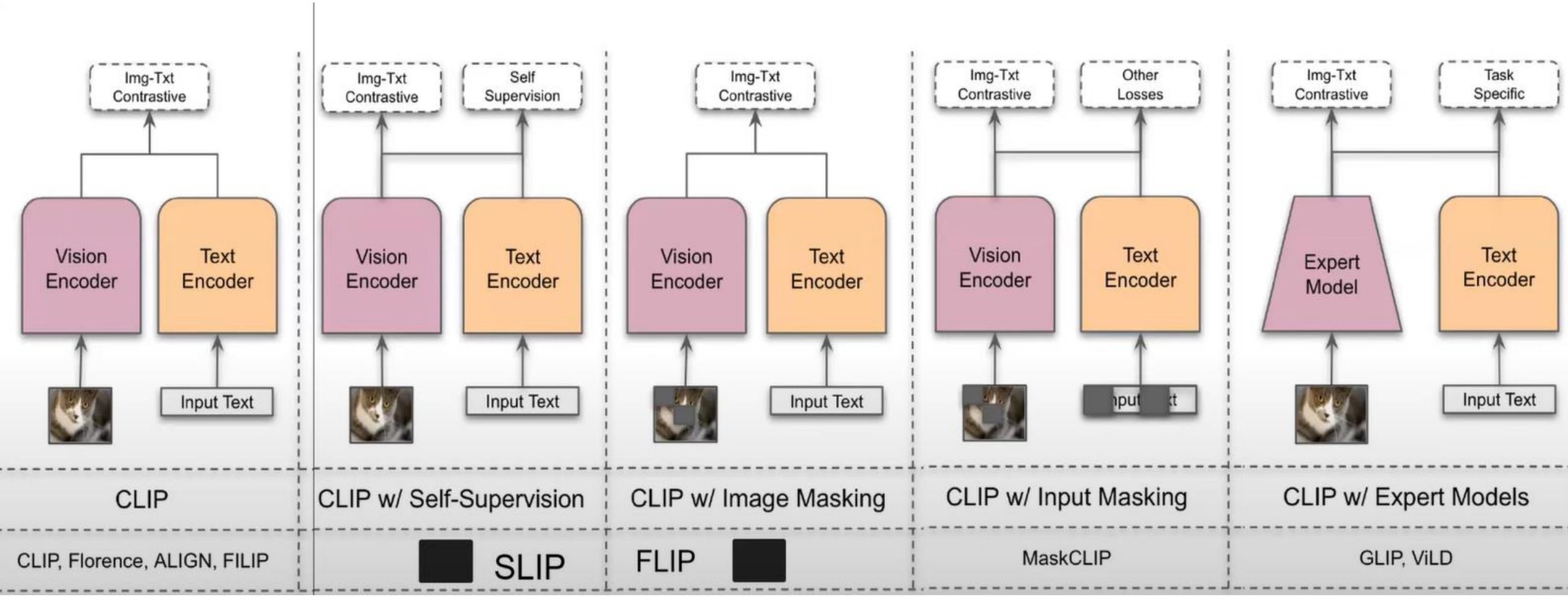
Dataset	Training Data # Samples	Method	COCO Retrieval I2T R@1	I2T R@5	T2I R@1	T2I R@5
ImageNet-21K	13M (full)	UniCL	2.66	7.46	0.98	3.54
	13M (full)	K-LITE	**4.04**	**12.20**	**1.91**	**6.48**
YFCC-14M + ImageNet-21K	14M (half)	UniCL	21.80	45.38	13.33	32.14
	14M (half)	K-LITE	**22.44**	**47.28**	**14.38**	**33.77**
GCC-15M + ImageNet-21K	15M (half)	UniCL	31.88	57.76	21.41	44.69
	15M (half)	K-LITE	**32.68**	**58.88**	**22.08**	**45.41**

Table 7: Overall comparisons of our knowledge-augmented models on zero-shot COCO Retrieval Evaluation. Each model is pre-trained with 32 epochs following UniCL [95].



[1] K-lite: Learning transferable visual models with external knowledge, NeurIPS 2022

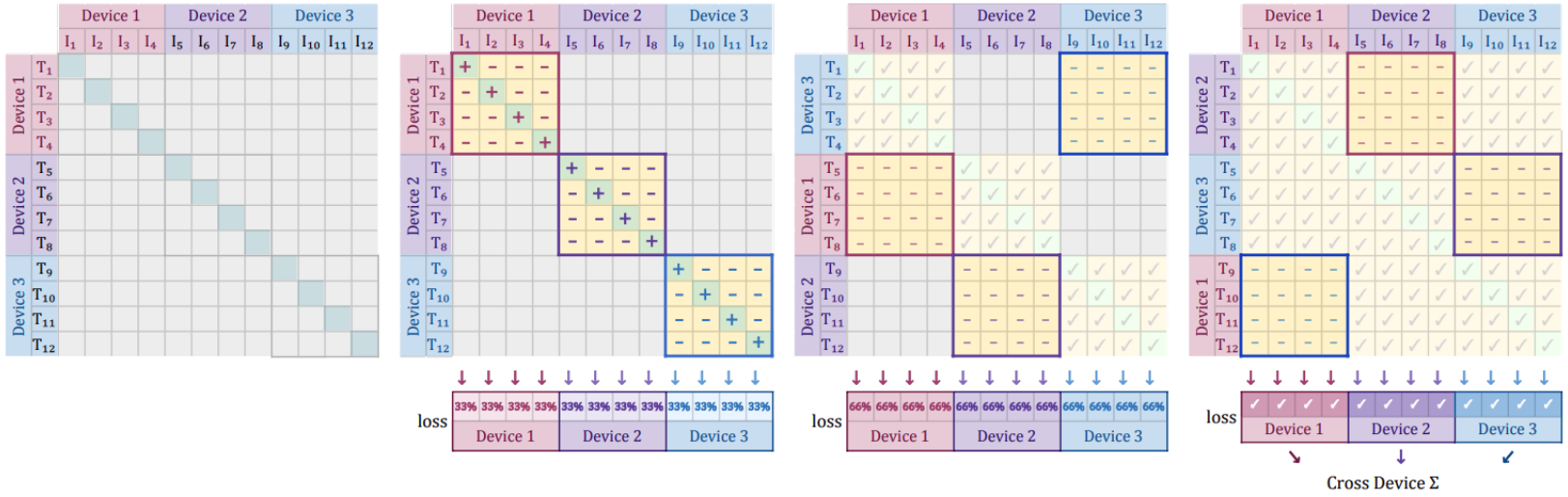
2.2 How to improve CLIP



2.2 How to improve CLIP

SigLIP[1]

- SigLIP proposes to replace the loss function used in CLIP by a simple pairwise sigmoid loss. This results in better performance in terms of zero-shot classification accuracy on ImageNet.
- The main difference is the training loss, which does not require a global view of all the pairwise similarities of images and texts within a batch. One needs to apply the sigmoid activation function to the logits, rather than the softmax.



(a) Initially each device holds 4 image and 4 text representations. Each device needs to see the representations from other devices to calculate the full loss.

(b) They each compute the component of the loss (highlighted) for their representations, which includes the positives.

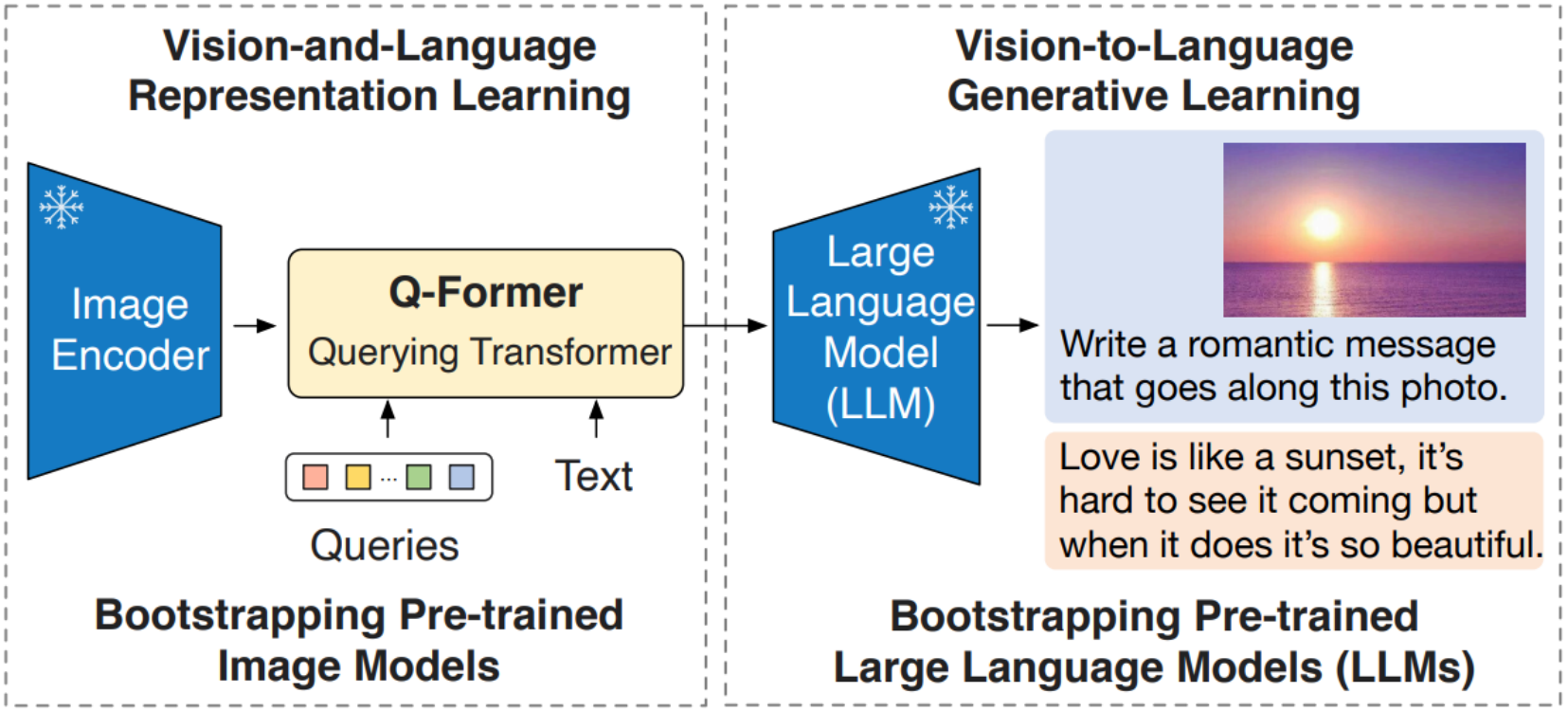
(c) Texts are swapped across the devices, so device 1 now has $I_{1:4}$ and $T_{5:8}$ etc. The new loss is computed and accumulated with the previous.

(d) This repeats till every image & text pair have interacted, e.g. device 1 has the loss of $I_{1:4}$ and $T_{1:12}$. A final cross-device sum brings everything together.

[1] X Zhai. Sigmoid Loss for Language Image Pre-Training. CVPR 2023.

2.3 BLIP-2

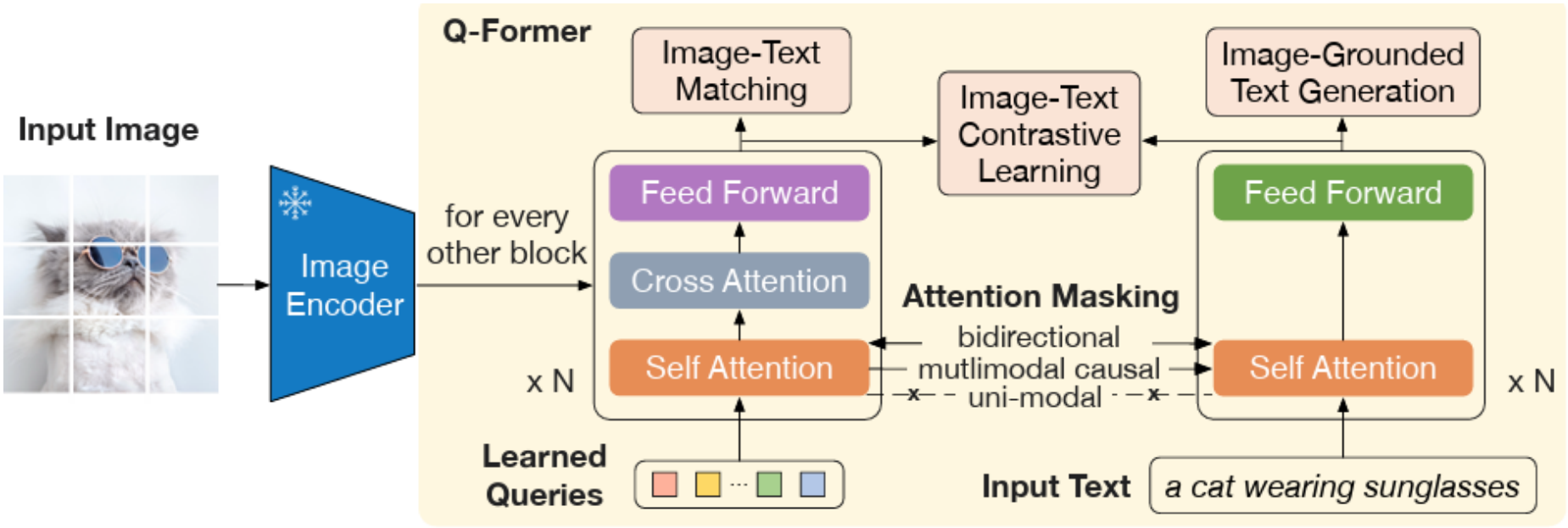
- BLIP-2[1] utilizes both a frozen image encoder and a frozen LLM
  - Less trainable parameters, also computationally less expensive
  - Outperforms similar models
- Introduces Q-Former to bridge modality gap
- Pre-trained in 2 Stages



[1] BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models, arXiv, 2023

2.3 BLIP-2

- Stage I: BLIP-2 model architecture of Q-Former



- Q-Former has 2 transformer submodules
- Queries can interact with each other through self-attention layers, and with the text
- Queries can interact with frozen image encoder at the cross-attention layer
- 32 queries with dimension of 768

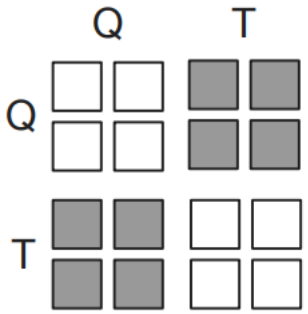
- Q-Former initialized with $BERT_{base}$ weights; cross attention layer initialized randomly
- Q-former jointly optimizes three objectives which enforce the queries (a set of learnable embeddings) to extract visual representation most relevant to the text.

[1] BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models, arXiv, 2023

2.3 BLIP-2 Stage I

- The self-attention masking strategy for each objective to control query-text interaction.
- Objective 1 – CLIP like structure

Q: query token positions; **T**: text token positions.
■ masked □ unmasked



Uni-modal
Self-Attention Mask

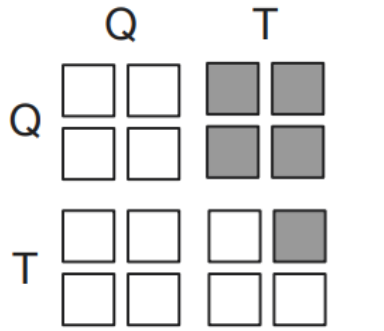
Image-Text
Contrastive Learning

- Employ Unimodal self attention – image and texts are not allowed to see each other
- Intention is to maximize text and image mutual information
- Compute pairwise similarity and take the highest

2.3 BLIP-2 Stage I

- Objective 2 – image-grounded text generation

Q: query token positions; **T**: text token positions.
■ masked □ unmasked



Multi-modal Causal Self-Attention Mask

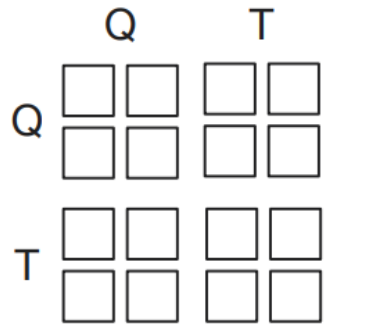
Image-Grounded Text Generation

- Employ Multimodal causal self attention
- Queries can interact with each other
- Text can interact with all the queries and the text prior to it

2.3 BLIP-2 Stage I

- Objective 3: match image-text

Q: query token positions; **T**: text token positions.
■ masked □ unmasked



Bi-directional
Self-Attention Mask

Image-Text
Matching

- Employ bi-directional self attention
- All text and queries can attend to each other
- Binary classification task – pass the pair to a binary classifier and average it

2.3 BLIP-2 Stage II Generative Learning

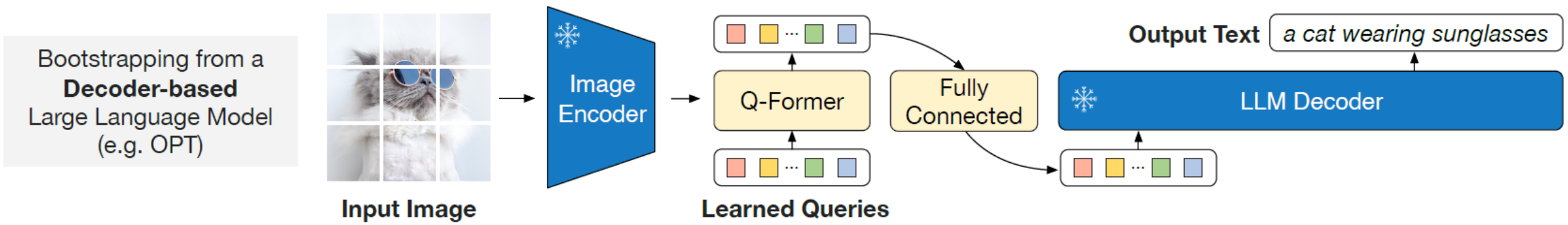
- Stage II: Generative Learning

- For generative pre-training, connect Q-Former to frozen LLM

- Use Fully Connected Layer to project output query embedding to the input embeddings of the LLM

- Q-Former feeds relevant information to LLM to mitigate forgetting problem

- **Limitations:** Does not observe an improved VQA performance with in-context VQA examples.

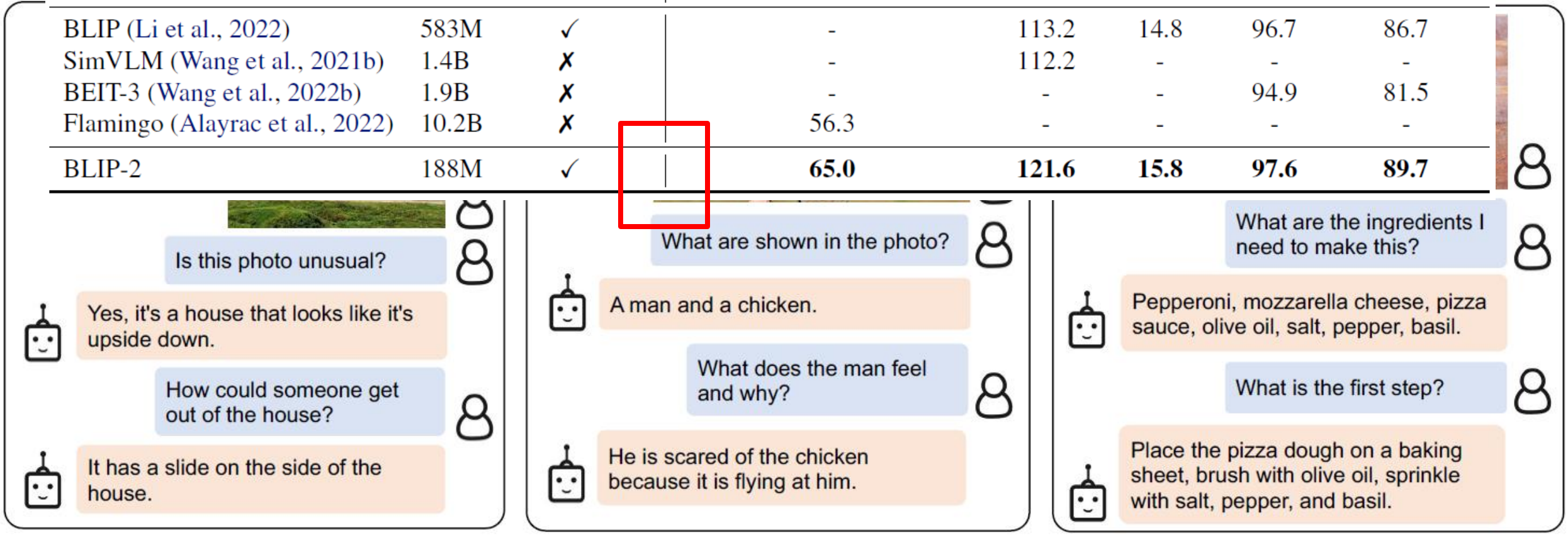


2.3 BLIP-2 Stage II Generative Learning

- Pre-training data

- Total 129M images → 10 captions per image → Top 2 captions per image

Models	#Trainable Params	Open-sourced?	Visual Question Answering VQAv2 (test-dev) VQA acc.	Image Captioning NoCaps (val) CIDEr	Image Captioning NoCaps (val) SPICE	Image-Text Retrieval Flickr (test) TR@1	Image-Text Retrieval Flickr (test) IR@1
BLIP (Li et al., 2022)	583M	✓	-	113.2	14.8	96.7	86.7
SimVLM (Wang et al., 2021b)	1.4B	✗	-	112.2	-	-	-
BEIT-3 (Wang et al., 2022b)	1.9B	✗	-	-	-	94.9	81.5
Flamingo (Alayrac et al., 2022)	10.2B	✗	56.3	-	-	-	-
BLIP-2	188M	✓	**65.0**	**121.6**	**15.8**	**97.6**	**89.7**



2.4 Large-Vision-language Models

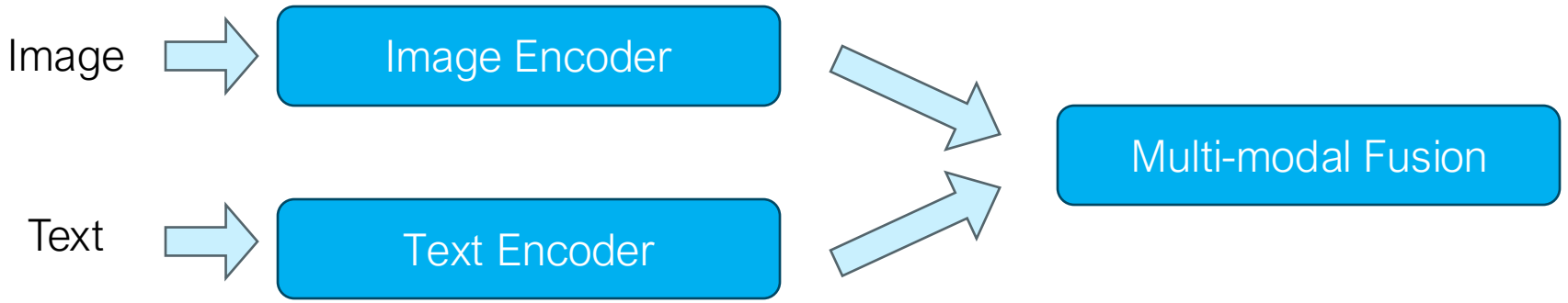
Model	Model Size				#Pre-training image-text data	Pre-training tasks
	Img Enc	Txt Enc	Fusion	Total		
CLIP ViT-L/14	302M	123M	0	425M	400M	ITC
ALIGN	480M	340M	0	820M	1.8B	ITC
Florence	637M	256M	0	893M	900M	ITC
SimVLM-huge	300M	39M	600M	939M	1.8B	PrefixLM
METER-huge	637M	125M	220M	982M	20M*	MLM+ITM
LEMON	147M	39M	636M	822M	200M	MLM
Flamingo	200M	70B	10B	80.2B	2.1B+27M video-text	LM
GIT	637M	40M	70M	747M	800M	LM
VLMo++	--	--	--	565M	1B	MLM+ITM+ITC
CoCa	1B	477M	623M	2.1B	4.8B (before filtering)	ITC+LM

Pre-training tasks:

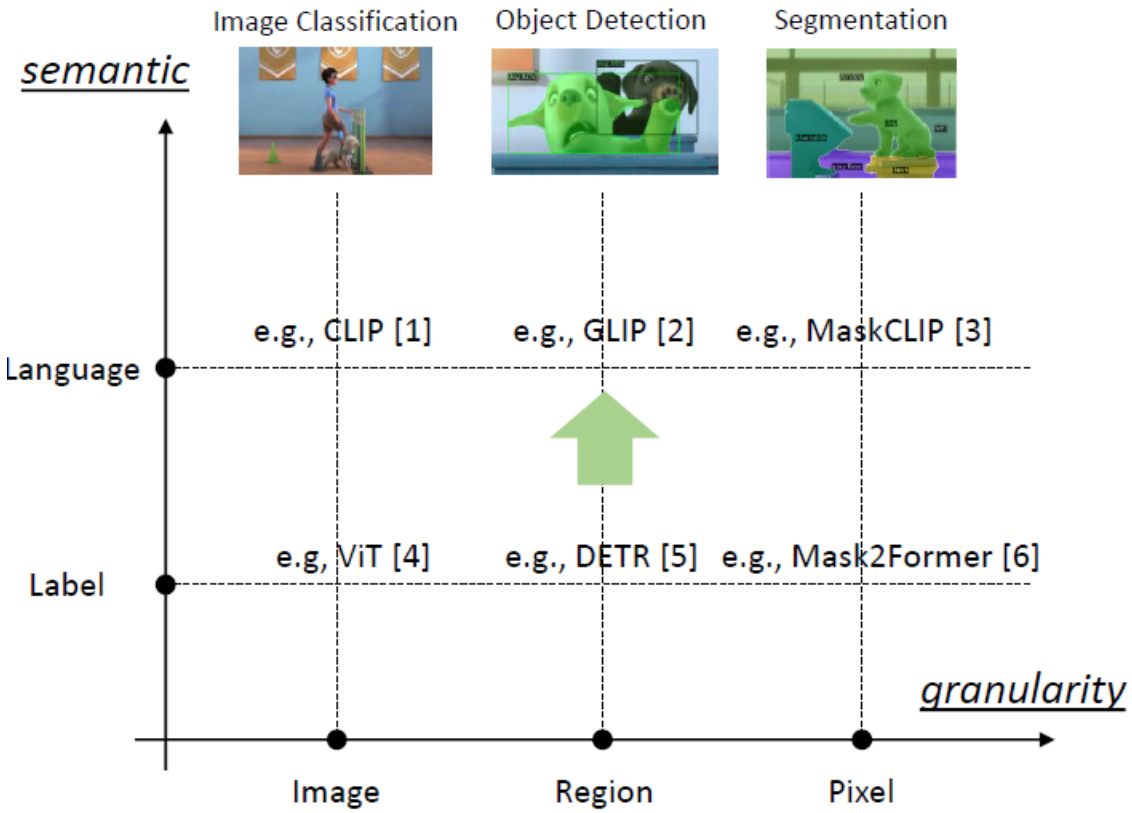
- ITC: Image-text Contrastive loss
- ITM: Image-text Matching loss
- MLM: Masked Language modelling loss
- LM: Language modelling
- PrefixLM: The prefix language modelling loss randomly samples a contiguous span of *k* tokens from the given tokenized text.

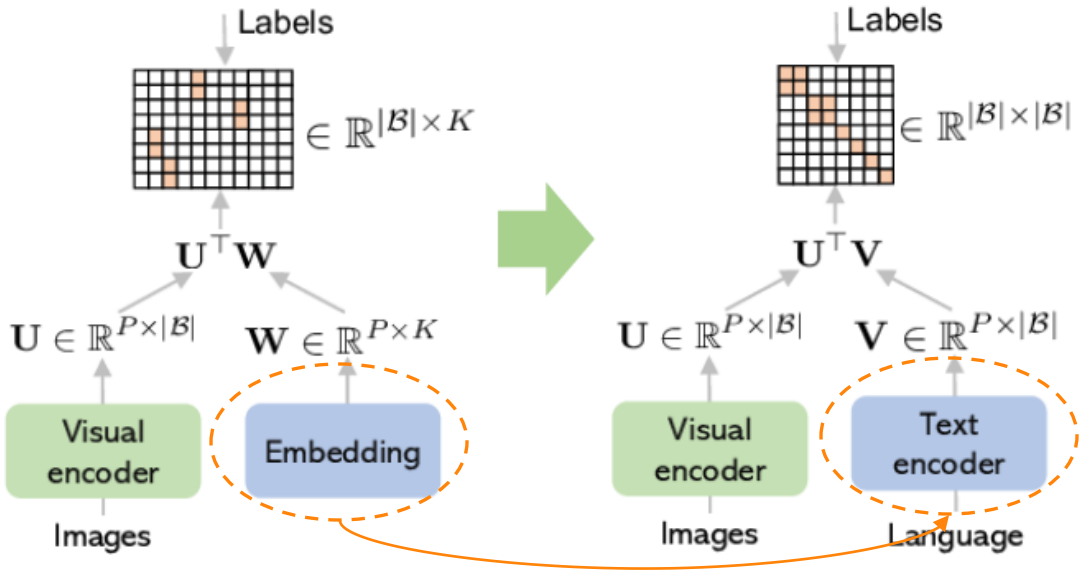
2.4 Large-Vision-language Models

- Image encoder, text encoder, multi-modal fusion



2.5 Bridge Vision with Language





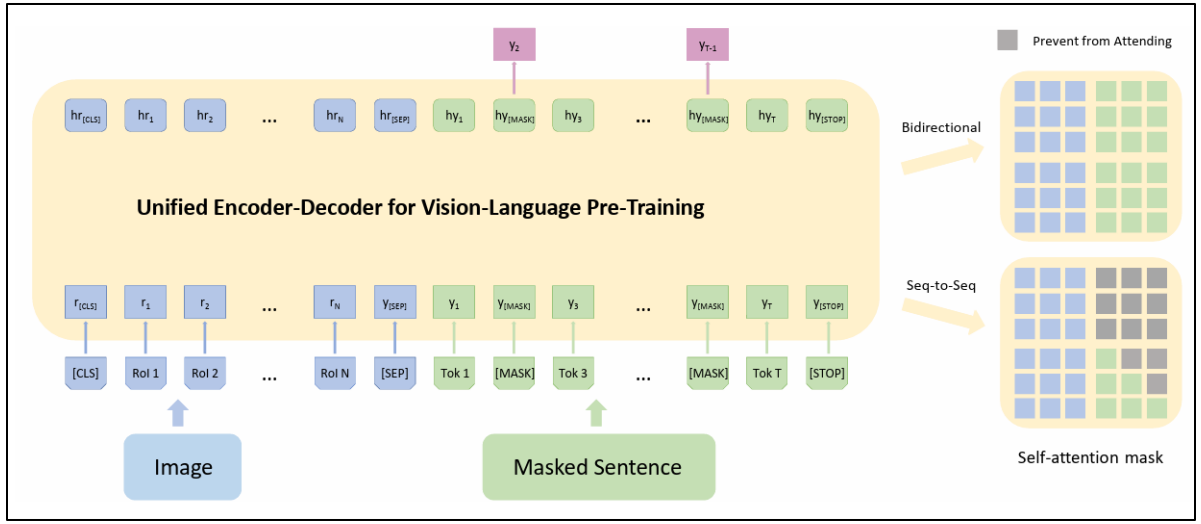
Replace labels with concept names, and use text encoder to encode all concepts as they are language tokens

[1]Radford et al. "Learning transferable visual models from natural language supervision." ICML, PMLR, 2021
[2]Li et al. "Grounded language-image pre-training. "CVPR, 2022
[3]Zhou et al. "Extract Free Dense Labels from CLIP."ECCV, 2022
[4]Dosovitskiy et al. "An image is worth 16x16 words: Transformers for image recognition at scale."*ICLR,2021*
[5]Carion et al. "End-to-end object detection with transformers." *ECCV, 2020*
[6]Cheng et al. "Masked-attention mask transformer for universal image segmentation." *CVPR. 2022*

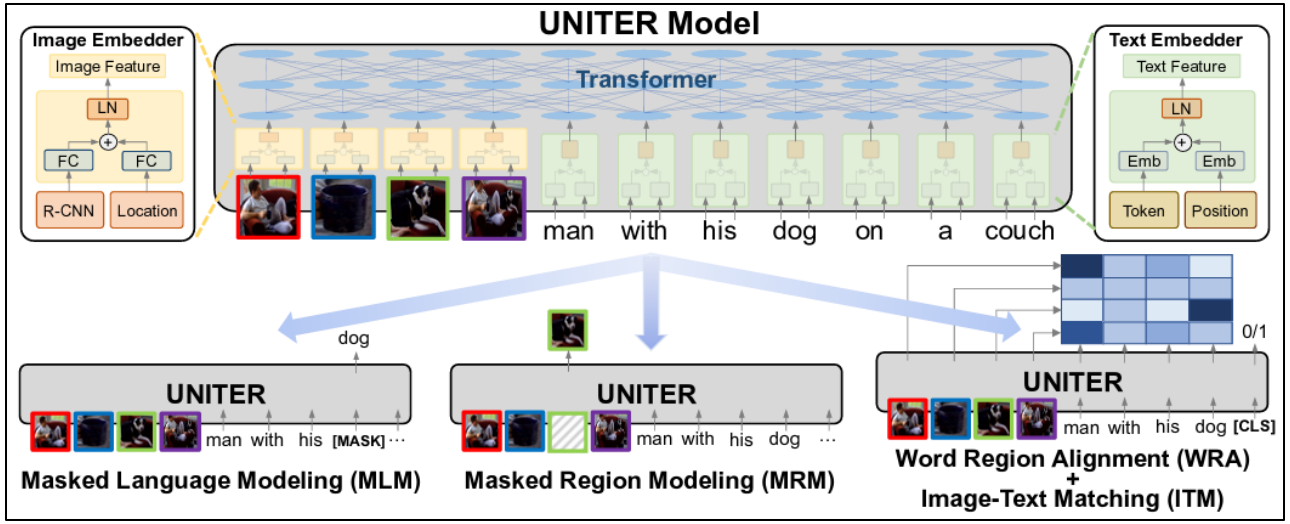
2.4 Large-Vision-language Models

- (1) Multi-modal Fusion – Transformer Encoder

- Input concatenation, self-attention, modality-unaware



Model architecture for pre-training.
The self-attention mask controls what input context the prediction conditions on. They implemented two self-attention masks depending on whether the objective is bidirectional or seq2seq.

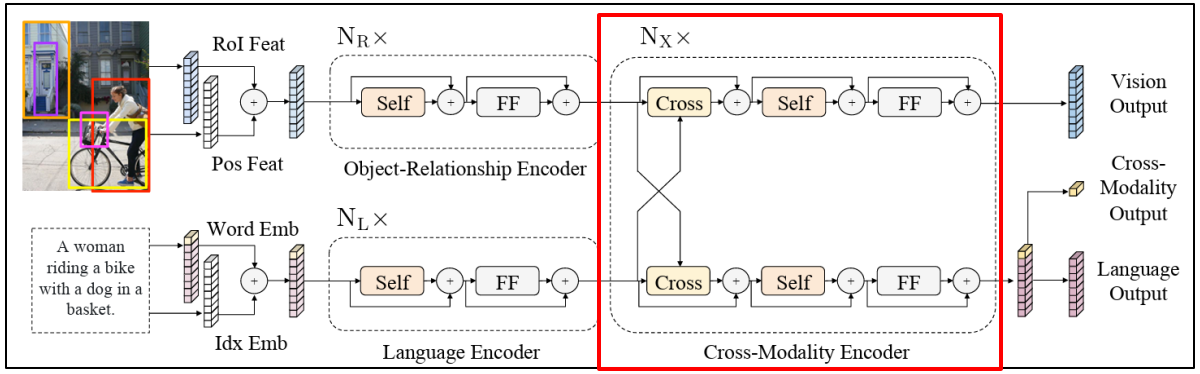


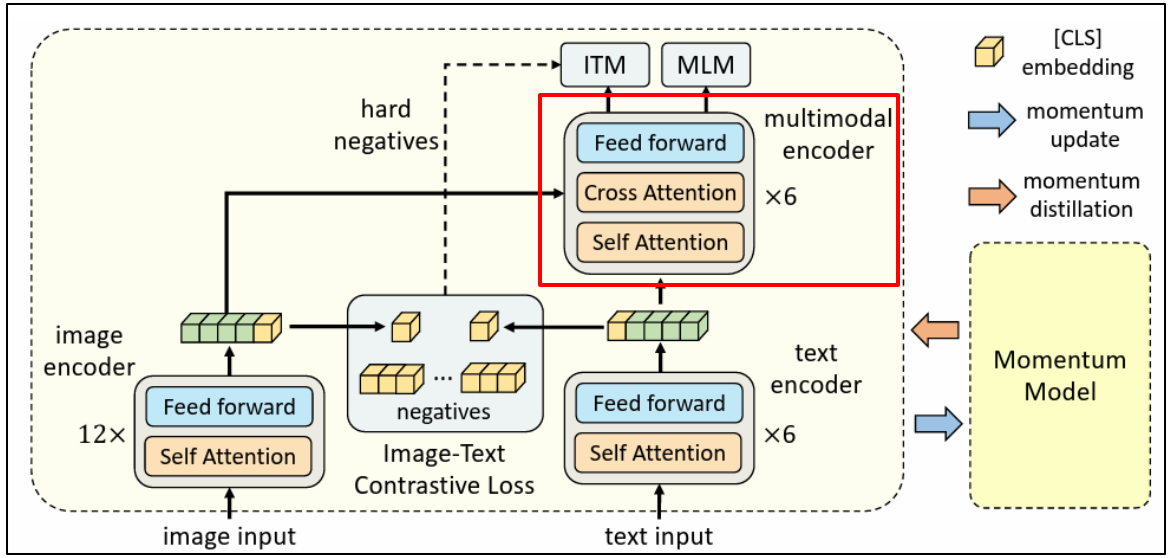
Overview of the proposed UNITER model, consisting of an Image Embedder, a Text Embedder and a multi-layer Transformer, learned through four pre-training tasks
(i) Masked Language Modelling (MLM) conditioned on image;
(ii) Masked Region Modelling (MRM) conditioned on text;
(iii) Image-Text Matching (ITM);
(iv) Word-Region Alignment (WRA)

[1] *Unified Vision-Language Pre-Training for Image Captioning and VQA*, 2019
[2] *UNITER: UNiversalImage-TExtRepresentation Learning*, 2019

2.4 Large-Vision-language Models

- (1) Multi-modal Fusion – Transformer Encoder
  - Self-attention, cross-attention, modality-aware





The LXMERT model for learning vision-and-language cross-modality representations. 'Self' and 'Cross' are abbreviations for self-attention sub-layers and cross-attention sub-layers, respectively. 'FF' denotes a feed-forward sub-layer

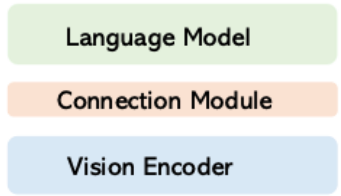
It consists of an image encoder, a text encoder, and a multimodal encoder. We propose an image-text contrastive loss to align the unimodal representations of an image-text pair before fusion.

An image-text matching loss (using in-batch hard negatives mined through contrastive similarity) and a masked-language-modelling loss are applied to learn multimodal interactions between image and text. In order to improve learning with noisy data, we generate pseudo-targets using the momentum model (a moving-average version of the base model) as additional supervision during training

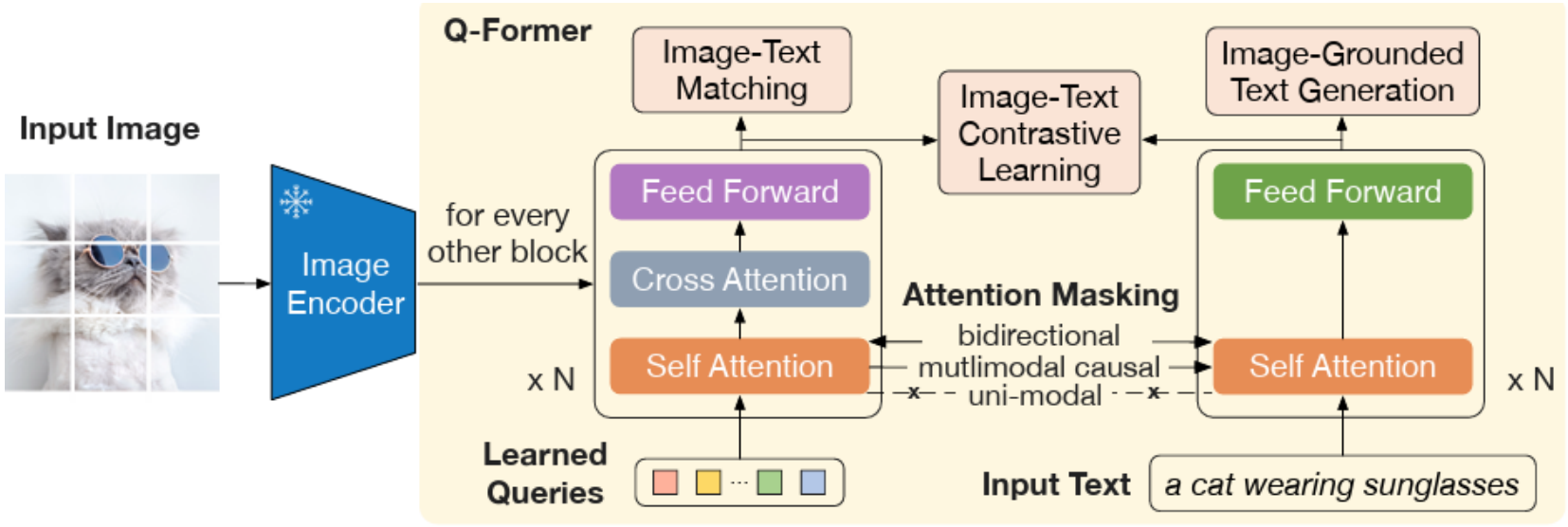
[1] *LXMERT: Learning Cross-Modality Encoder Representations from Transformers*, 2019
[2] *Align before Fuse: Vision and Language Representation Learning with Momentum Distillation*, 2021

2.4 Large-Vision-language Models

- (2) Multi-modal Fusion – Transformer Decoder
  - Self-attention, cross-attention, modality-aware
    - Freeze encoder[1-3]
    - Randomly initialize cross-attention modules[2]



Pre-trained: 70B Chinchilla
Perceiver Resampler Gated Cross-attention + Dense
Pre-trained: Nonrmalizer-Free ResNet (NFNet)



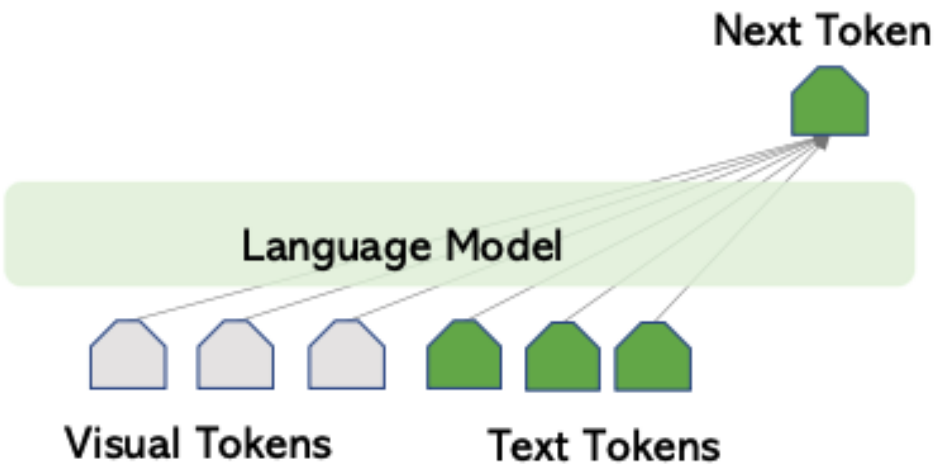
BLIP-2[3] model architecture of Q-Former and first-stage vision-language representation learning objectives. We jointly optimize three objectives which enforce the queries (a set of learnable embeddings) to extract visual representation most relevant to the text.

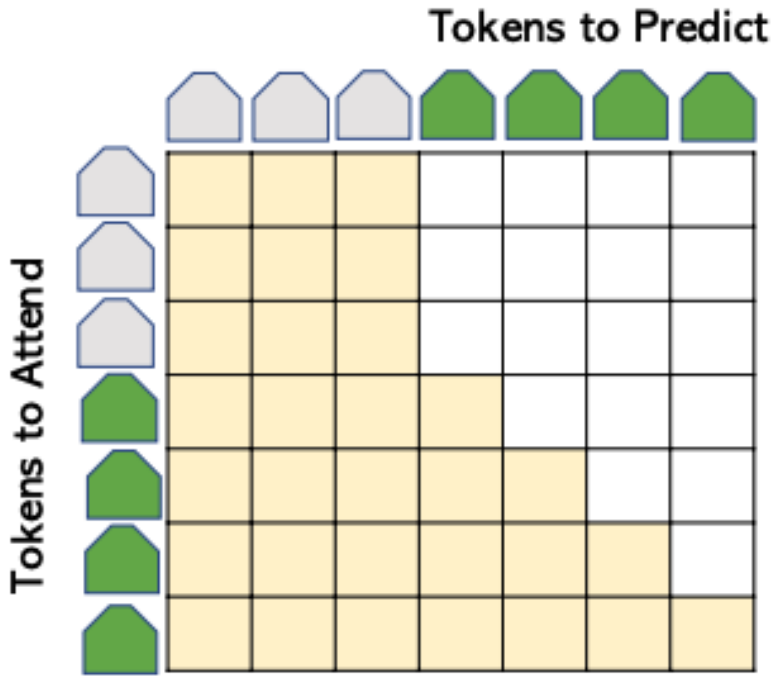
[1] *Flamingo: a Visual Language Model for Few-Shot Learning*, 2022
[2] CoCa: Contrastive Captioners are Image Text Foundation Models , 2022
[3] BLIP-2: *Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models*, arXiv, 2023

2.4 Large-Vision-language Models

Training Objectives

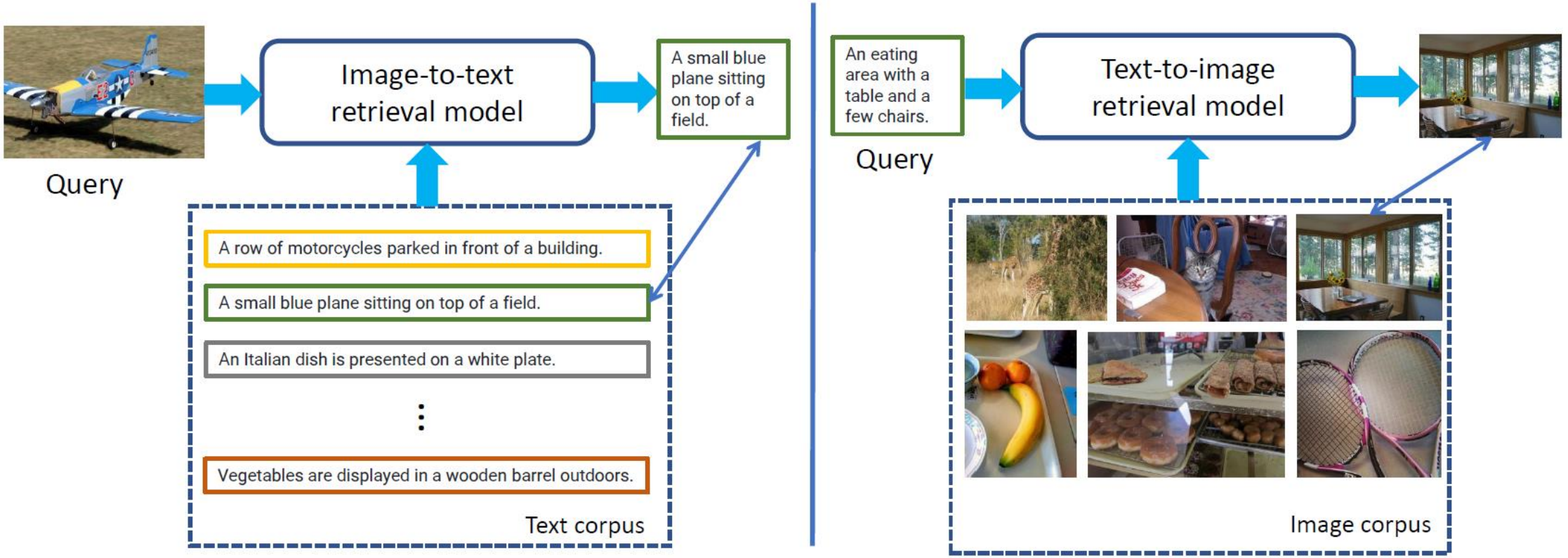
- Cross-Attended Image-to-Text Generation
- Autoregressive loss on language output





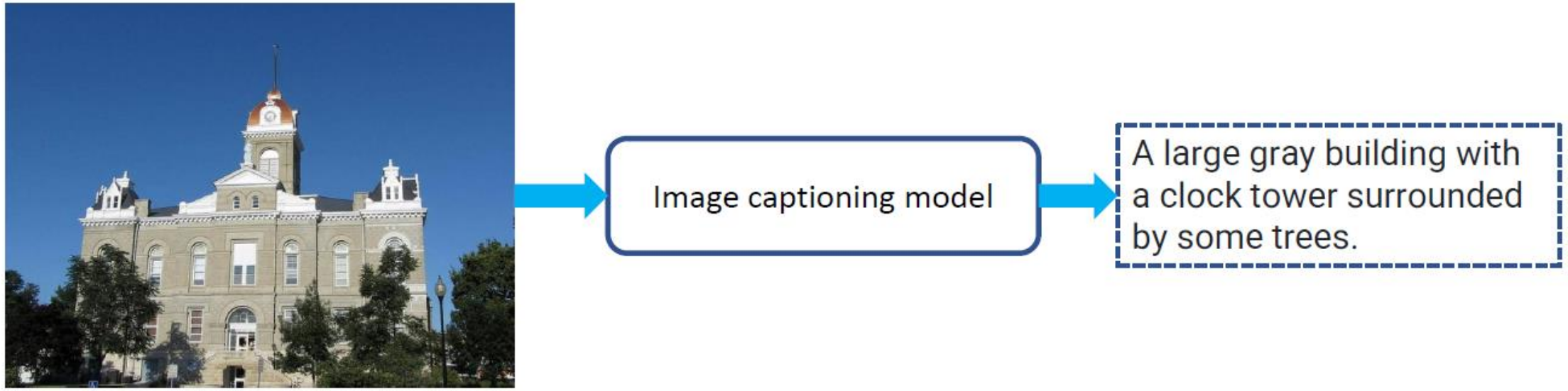
2.6 VLM Applications

- Multi-modal Retrieval



2.6 VLM Applications

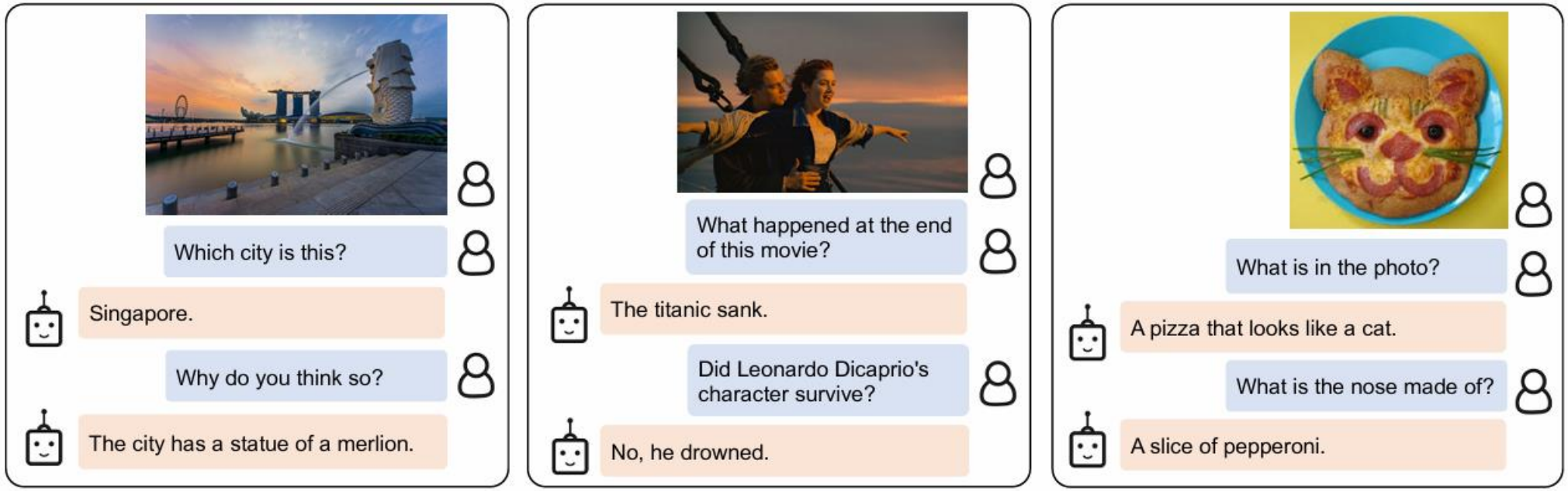
- Image Captioning



Vinyals, Oriol, et al. "Show and tell: A neural image caption generator." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

2.6 VLM Applications

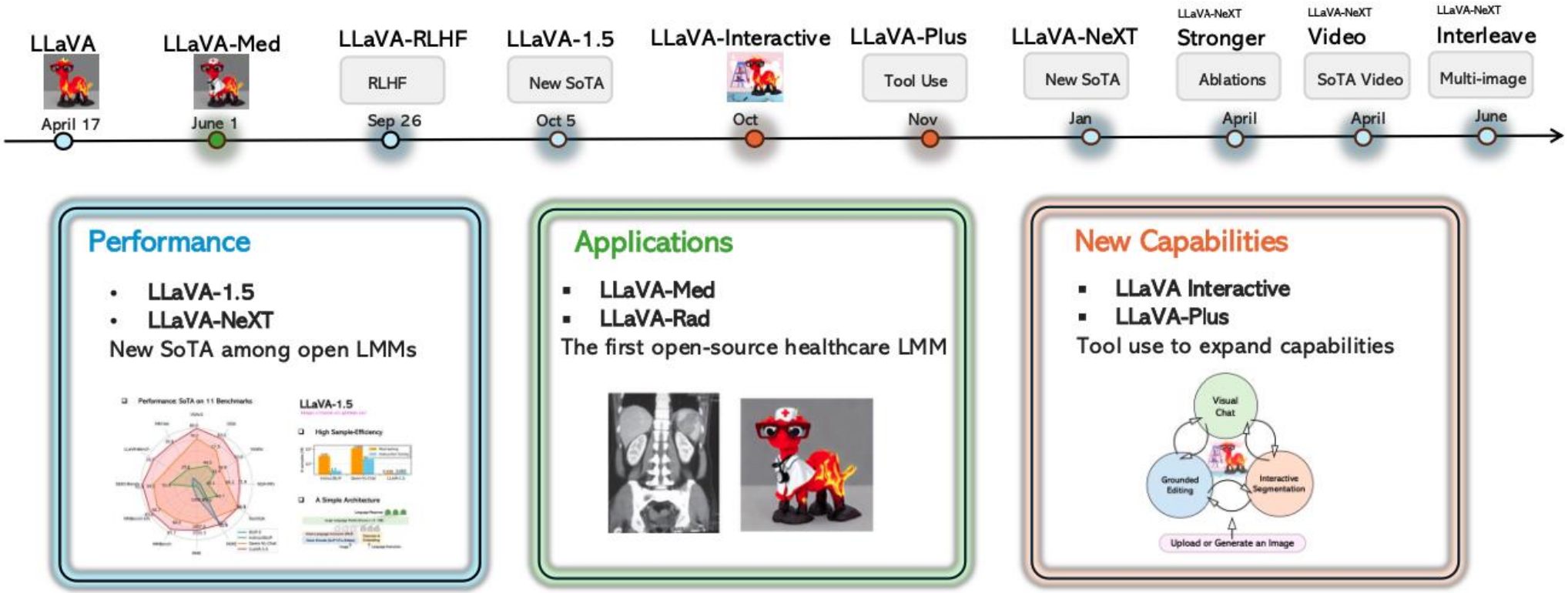
- Visual Question Answering



- Making the V in VQA Matter: Elevating the Role of Image Understanding in Visual Question Answering, 2016
- BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models, arXiv, 2023

2.7 LLaVa Series VL Models

- Milestone Open-source Project for Various Vision & Language Tasks



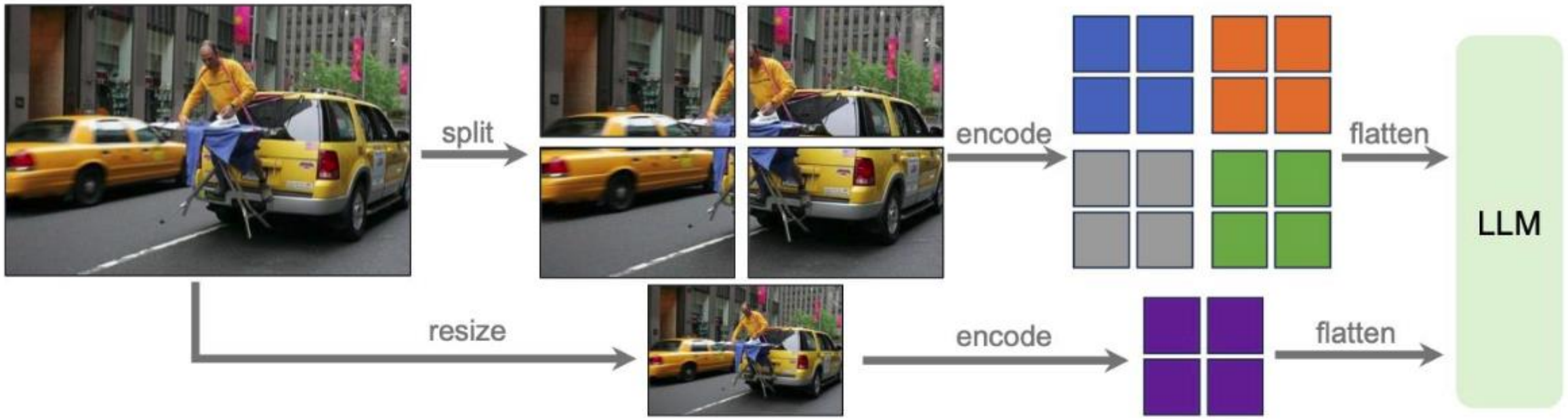
https://llava-vl.github.io

Bo Li, Llava-OneVision: Easy Visual Task Transfer. TMLR, 2025.

2.7 LLaVa Series VL Models

- LlaVA-NeXT: Improved reasoning, OCR, and world knowledge

(1) AnyRes: Dynamic High Resolution



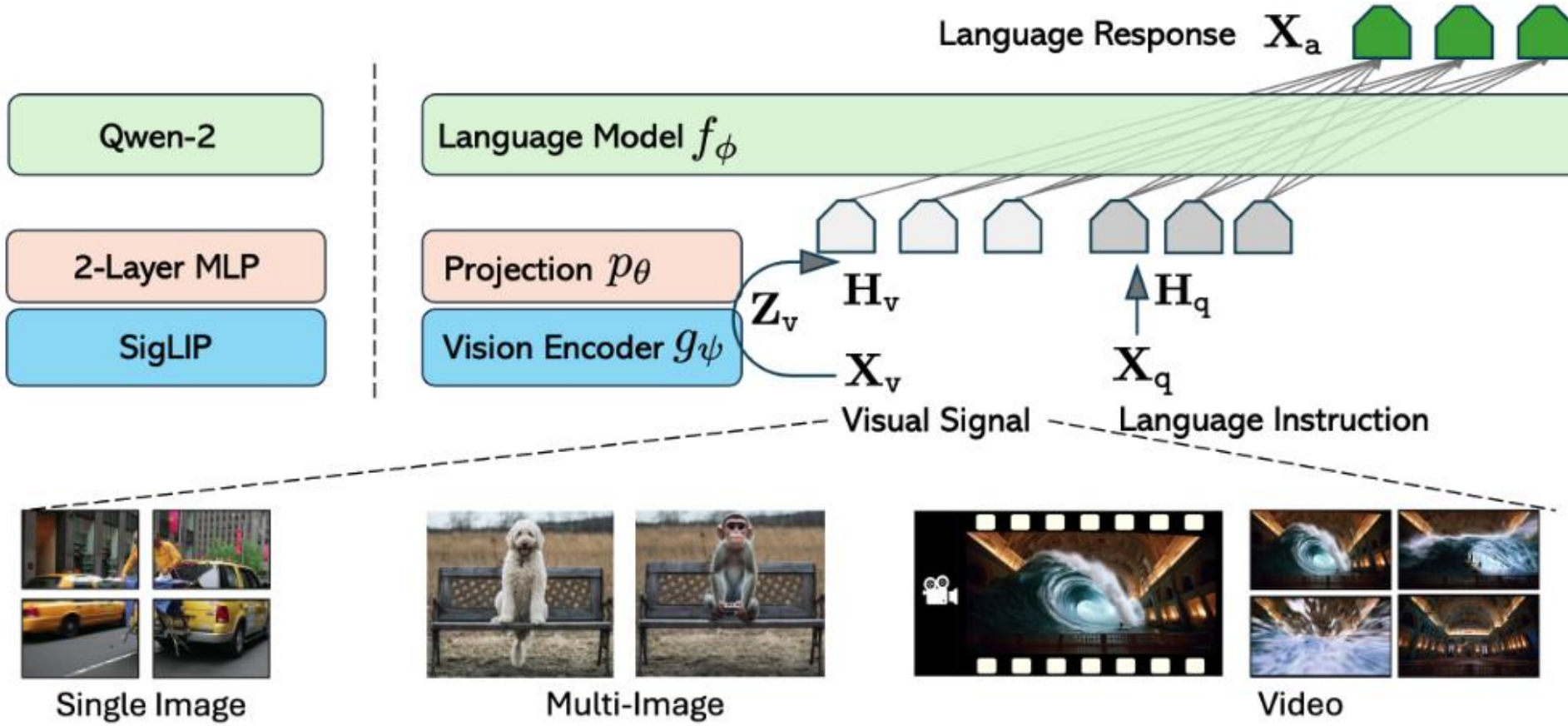
(2) Data Mixture

(3) Scaling LLM backbone with Mistral-7B, Vicuna-13B, etc.

https://llava-vl.github.io/blog/2024-01-30-llava-next/

2.7 LLaVa Series VL Models

- LLaVA-OneVision -- Easy Visual Task Transfer

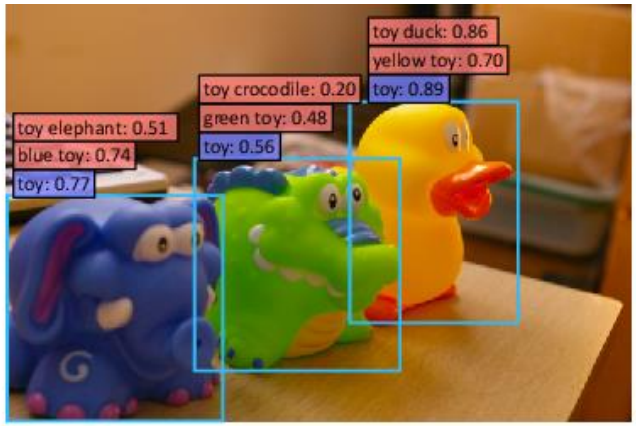


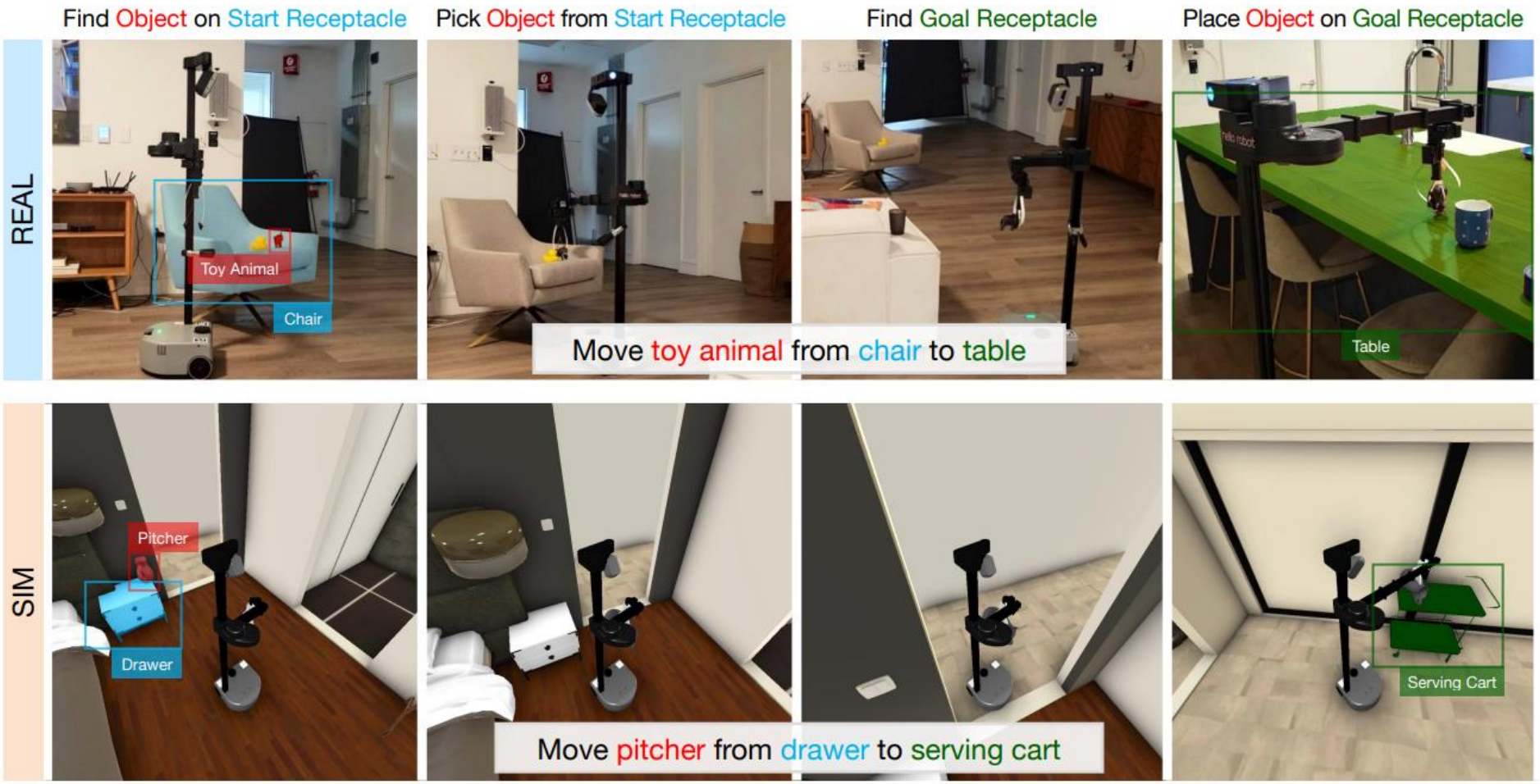
LLaVA-OneVision Network Architecture.

https://llava-vl.github.io/blog/2024-08-05-llava-onevision/

3.1 Open Vocabulary Detection

Detection techniques designed to handle and learn from inputs that include an extremely large, potentially infinite set of discrete elements such as words, symbols, or other data points.



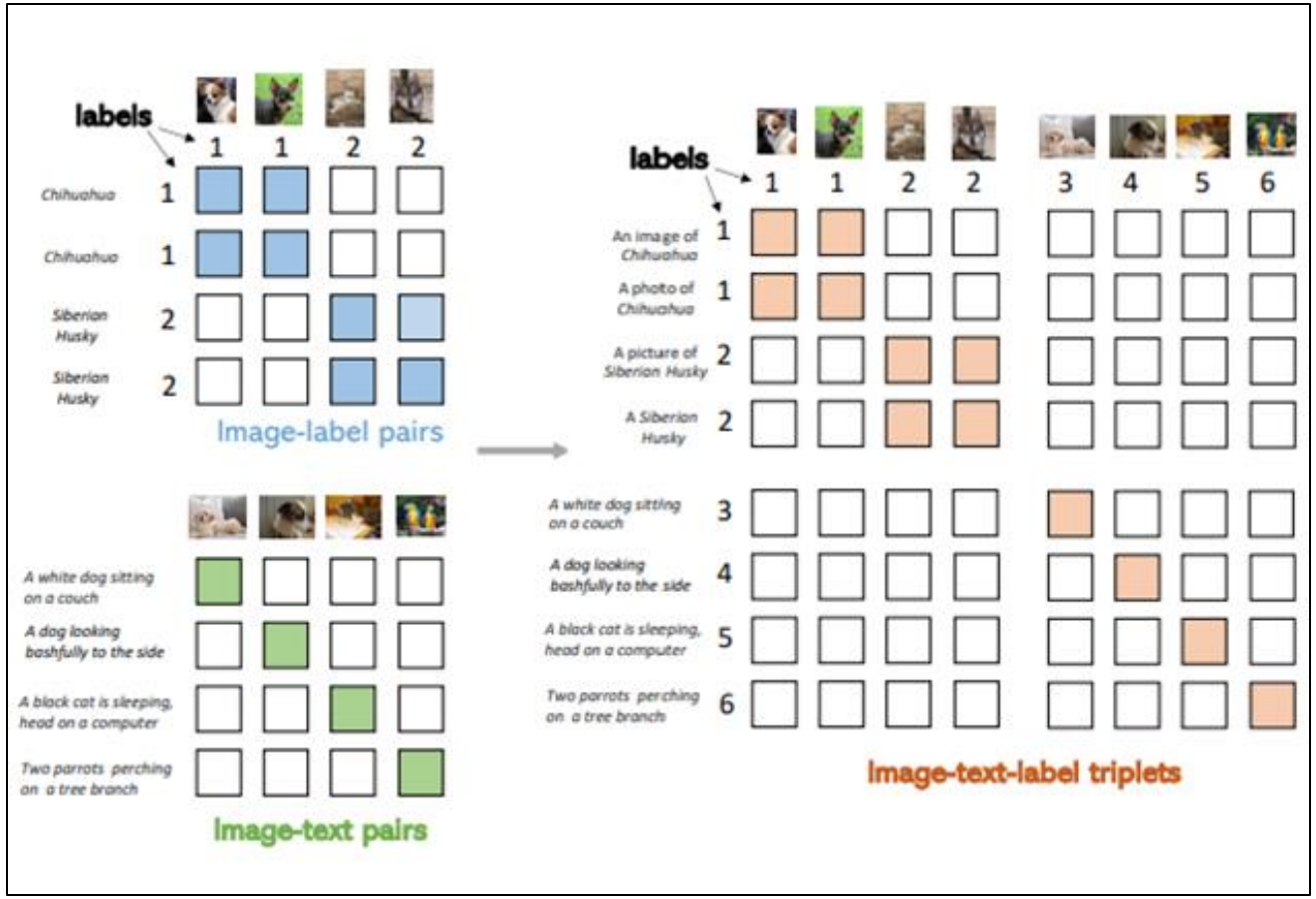


3.1 Open Vocabulary Detection

Closed-set Classification → **Open-world Recognition**

AlexNet, ResNet, ViT

CLIP, ALIGN [1], Florence [2]



Florence [2]

[1] Jia et al. "Scaling up visual and vision-language representation learning with noisy text supervision."ICML 2021.
[2] Yuan et al. "Florence: A new foundation model for computer vision."arXiv 2021.

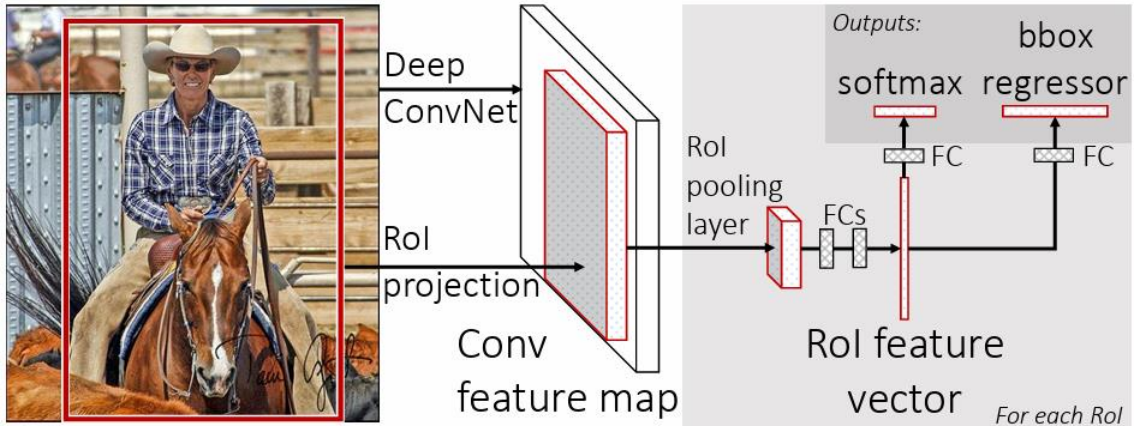
3.1 Open Vocabulary Detection

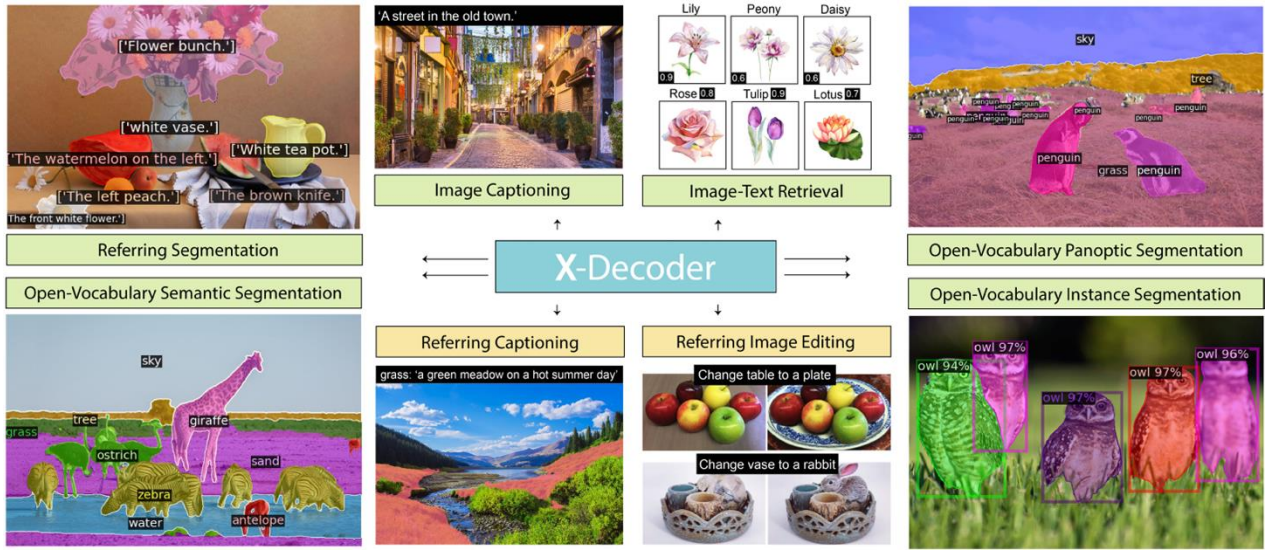
Specialist Models

Detection, Segmentation, VQA [1]

Generalist Models

Pixel2Seqv2 [2], UniTAB[3], OFA[4] , Unified-IO[5] , X-Decoder[6]





[1] Antol et al. "VQA: Visual question answering." *ICCV* 2015.
[2] Chen et al. "A unified sequence interface for vision tasks." *NeurIPS 2022*.
[3] Yang et al. "Unitab: Unifying text and box outputs for grounded vision-language modeling." *ECCV 2022*.
[4] Wang et al. "Ofa: Unifying architectures, tasks, and modalities through a simple sequence-to-sequence learning framework." *ICML* 2022.
[5] Lu et al. "Unified-io: A unified model for vision, language, and multi-modal tasks." *ICLR* 2022.
[6] Zou et al. "Generalized decoding for pixel, image, and language." *CVPR* 2023.

3.2 Open-Vocabulary Semantic Segmentation

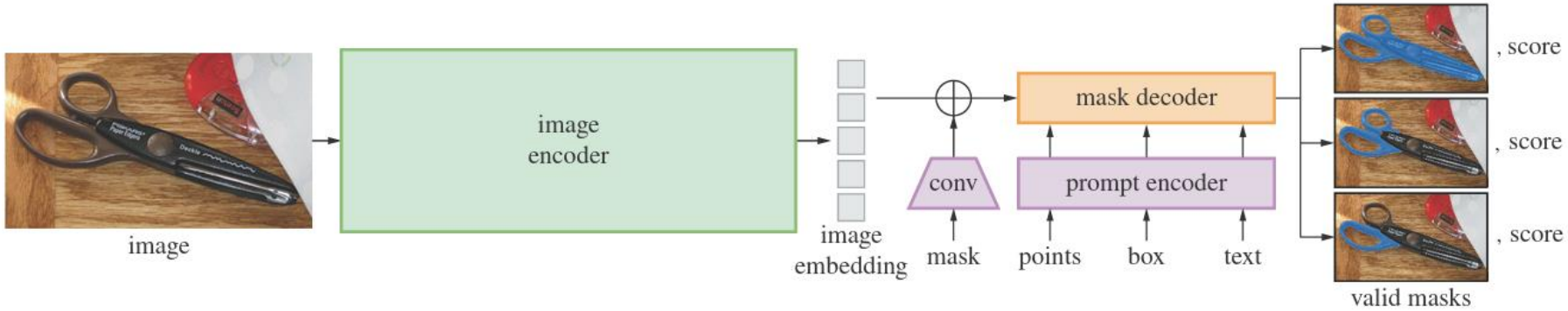
Where is Saturn V, blossom?

Where is Oculus, Ukulele?

Where is Golden gate, yacht?

3.3 Interactive Interface for Vision

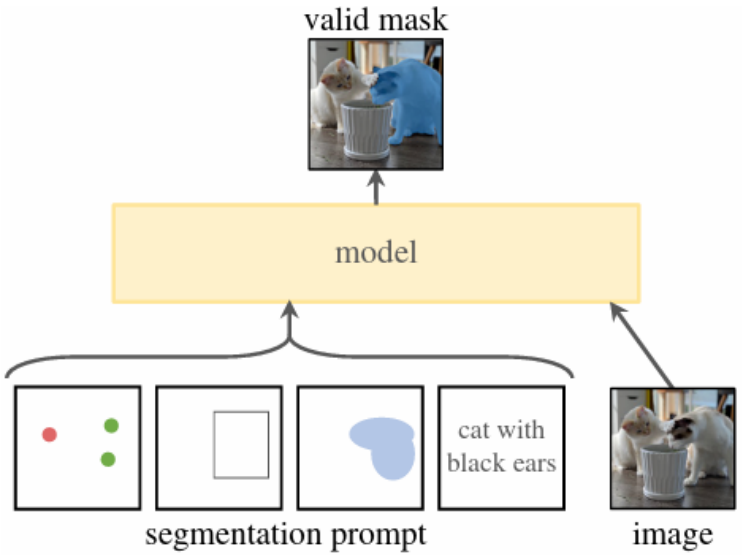
- SAM: Segment Anything [1]
  - SAM demonstrates remarkable zero-shot localization abilities



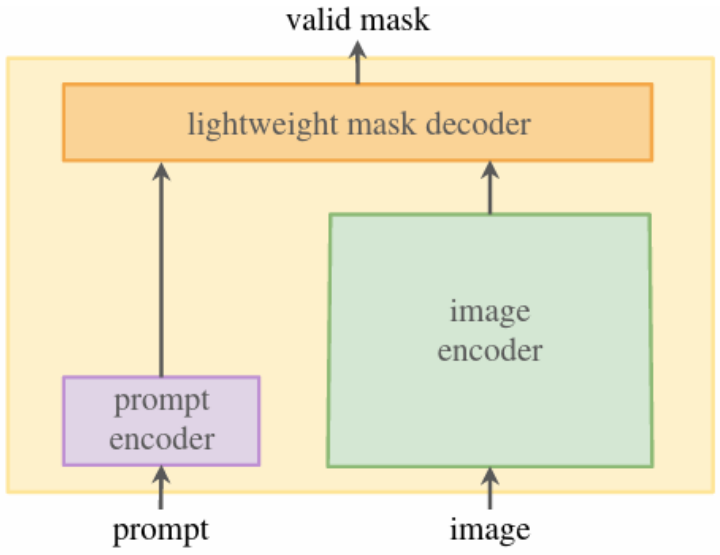
[1] Kirillov, Alexander, et al., Segment anything, Meta AI, in Arxiv, 2023

3.3 Interactive Interface for Vision

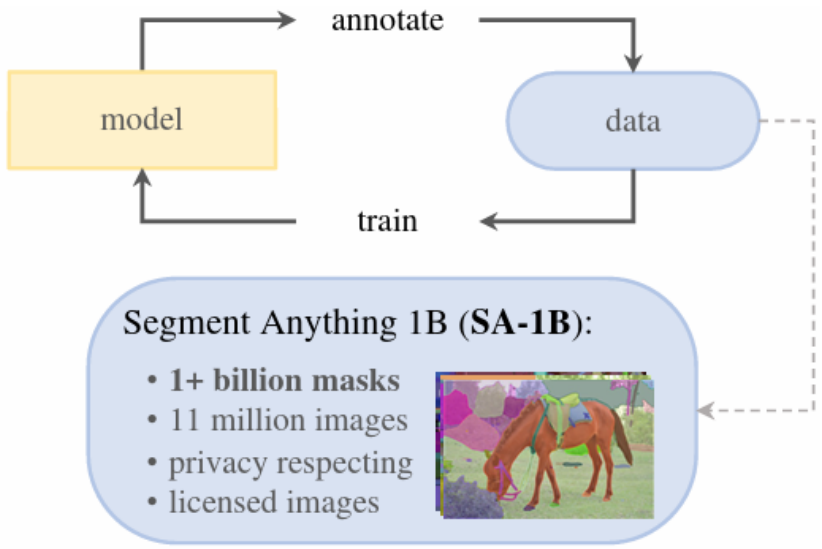
- SAM: Segment Anything [1]
  - Task: SAM supports segmentation based on input prompt (point, box, mask, text)
  - Model: A simple design satisfies all three constraints
  - Dataset: Data scaling-up of 1 billion masks via data engine
  - Assisted-manual (4.3M); Semi-automatic (10.2M); Fully automatic (1.1B)



(a) **Task**: promptable segmentation



(b) **Model**: Segment Anything Model (**SAM**)

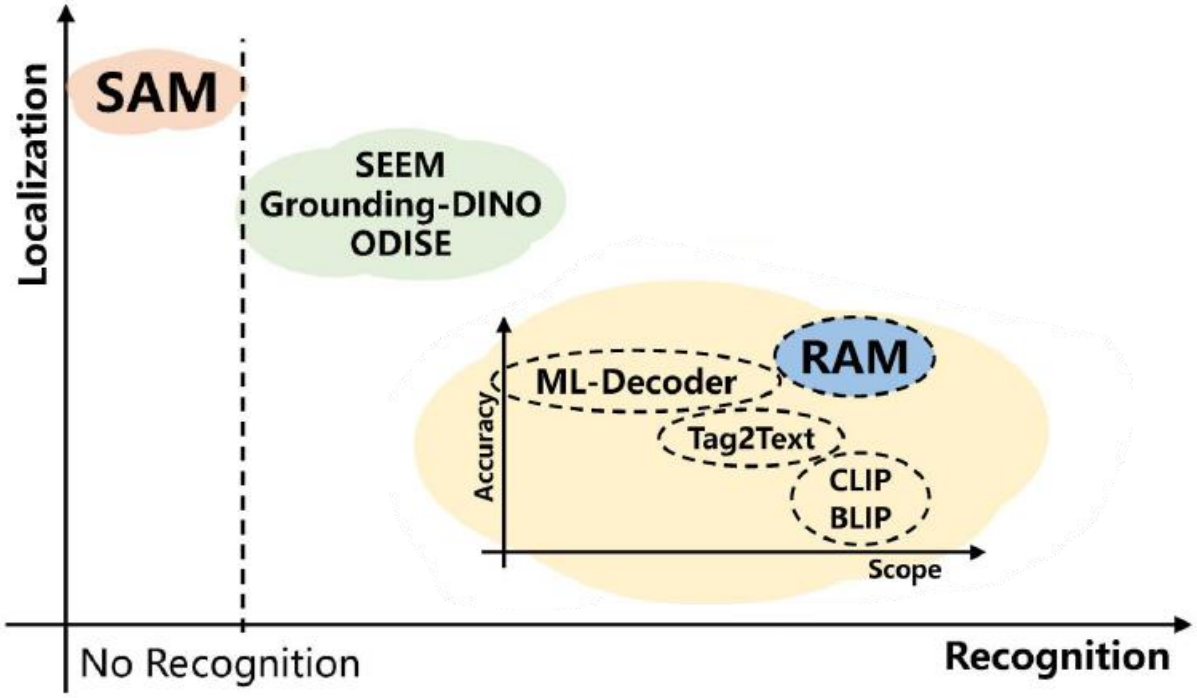


(c) **Data**: data engine (top) & dataset (bottom)

[1] Kirillov, Alexander, et al., Segment anything, Meta AI, in Arxiv, 2023

3.3 Interactive Interface for Vision

- SAM: Segment Anything [1]
  - Task: SAM supports segmentation based on input prompt (point, box, mask, text)
  - Model: A simple design satisfies all three constraints
  - Dataset: Data scaling-up of 1 billion masks via data engine
  - Assisted-manual (4.3M); Semi-automatic (10.2M); Fully automatic (1.1B)
- Vision-Language Model
  - Dataset: large scale, learning from text supervision
  - CLIP: Alignment Model
  - BLIP: Alignment + Generation Model
  - High scope, but low precision
- ML-Decoder[4]
  - Dataset: manually annotated tags with limited scale
  - High precision, but low scope



[1] Kirillov, Alexander, et al., Segment anything, Meta AI, in Arxiv, 2023
[2] Liu, Shilong, et al., Grounding dino: Marrying dino with grounded pre-training for open-set object detection, in Arxiv, 2023
[3] Xu, Jiarui, et al. Open-vocabulary panoptic segmentation with text-to-image diffusion models, in CVPR, 2023
[4] Ridnik, Tal, et al., Ml-decoder: Scalable and versatile classification head, in WACV, 2023

3.3 Interactive Interface for Vision

- SAM variants[1]: Grounded-SAM, Inpainting

Text prompt: Beach

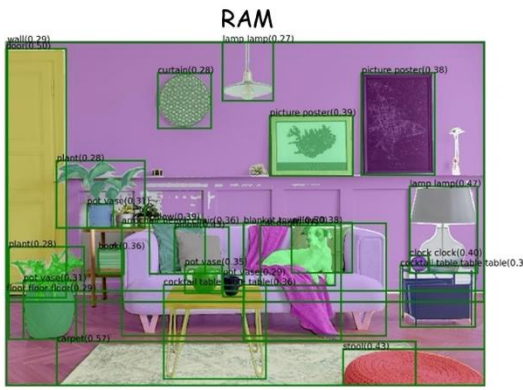
Grounded-SAM Output

Stable-Diffusion Inpainting
A Sofa, quality, detailed

- RAM: Recognize Anything[2]
  - Integration of image tagging and image captioning
  - Fine-grained alignment compared to CLIP
  - Integrates tags into text generation as guiding elements

living room, couch, plant
Missing: dog, lamp,

living room, dog, sit, blanket, rug, sofa, couch, plant, modern, ...
Missing: lamp, door



living room, couch, dog, sit, lamp, blanket, picture frame, modern, door, stool, book ...

[1] https://github.com/IDEA-Research/Grounded-Segment-Anything
[2] Zhang, Youcai, et al., Recognize Anything: A Strong Image Tagging Model, in Arxiv, 2023

Thank you!

Dr. Zhang Handuo
handuo.zhang@shanda.com