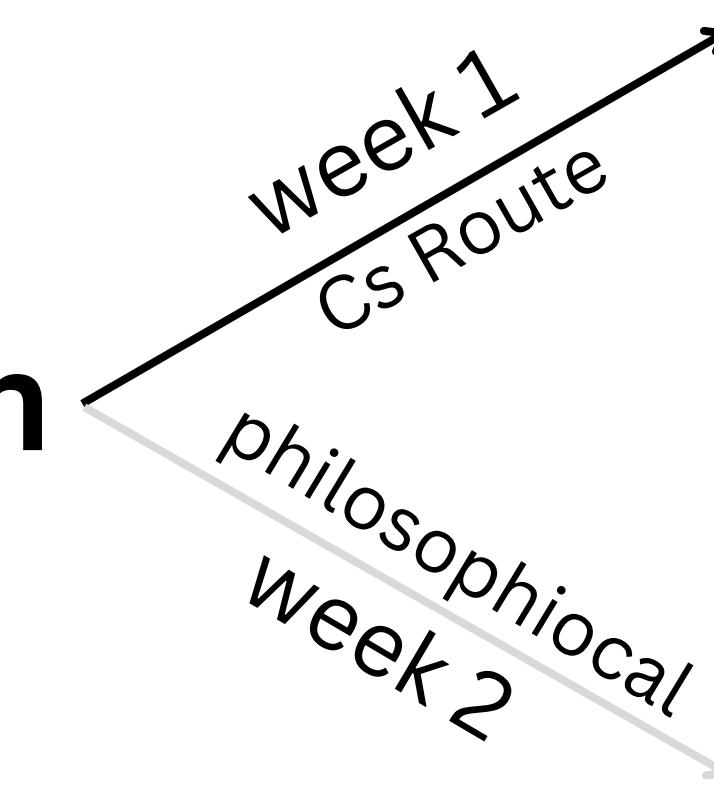


# The Llama 3.1 Herd of Models

Paper Reading session



**Basics and Building  
Blocks of llama**

Bench marks, Training  
recipes and Impact on  
LLM space

## **This session Hopefully answers these question from our POV**

- llama is another Decoder only Model (ever wondered why companies use decoder only models but they exist encoder and decoder based models ?)
- What did they do new to talk about them except for the fact of being open source
- What are interesting finds on this Models except data collecting, data mining (more on collection of data on next session) ?
- What are the guidelines of training a LLMs in this era efficiently and parallelly

**Bit into transformers decoder only (skip if people are familiar)**

i) Let's assume Encoder is parameterized with  $\Theta$  & Decoder is with  $\Phi$

So the parameters from the Encoder( $\Theta$ ) & Decoder( $\Phi$ ) interact only when cross-att.

So let  $x_{encIn}$  be Encoder Input (let it be image, Text or any)

$$x_{enc} = \Theta(x_{enc\text{Input}})$$

& on decoder Input  $x_{decIn}$

$$x_{dec} = \Phi_{\text{MMHA}}(x_{decIn})$$

$$x_{decout} = \Phi_{\text{CMHA}}(x_{enc}, x_{dec})$$

↳ Loss is evaluated here

A Solution for this problem any?

comes Decoder model just add them as tokens

Eg:

<S> words.. <Img> Img tokens </Img>

and so on

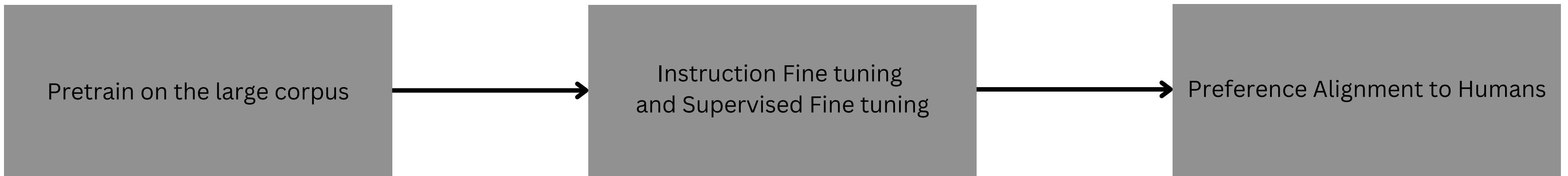
so we can get  $y = p(x_{\text{words}}, x_{\text{Img tok}}, \text{other modalities})$

# Lets answer the first question of this session ?

- Why decoder only models used in Multimodal (leave llama) in general ?
- why not encoder decoder models ? doesnt make sense right ? or does it arguable convos be encoder decoder models such as T5

explanation in Whiteboard

# General Pipe Line of LLM



# Model architecture details on the paper

	<b>8B</b>	<b>70B</b>	<b>405B</b>
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	$3 \times 10^{-4}$	$1.5 \times 10^{-4}$	$8 \times 10^{-5}$
Activation Function		SwiGLU	
Vocabulary Size		128,000	
Positional Embeddings		RoPE ( $\theta = 500,000$ )	

1.?

2.?

32  
8

$3 \times 10^{-4}$

$1.5 \times 10^{-4}$

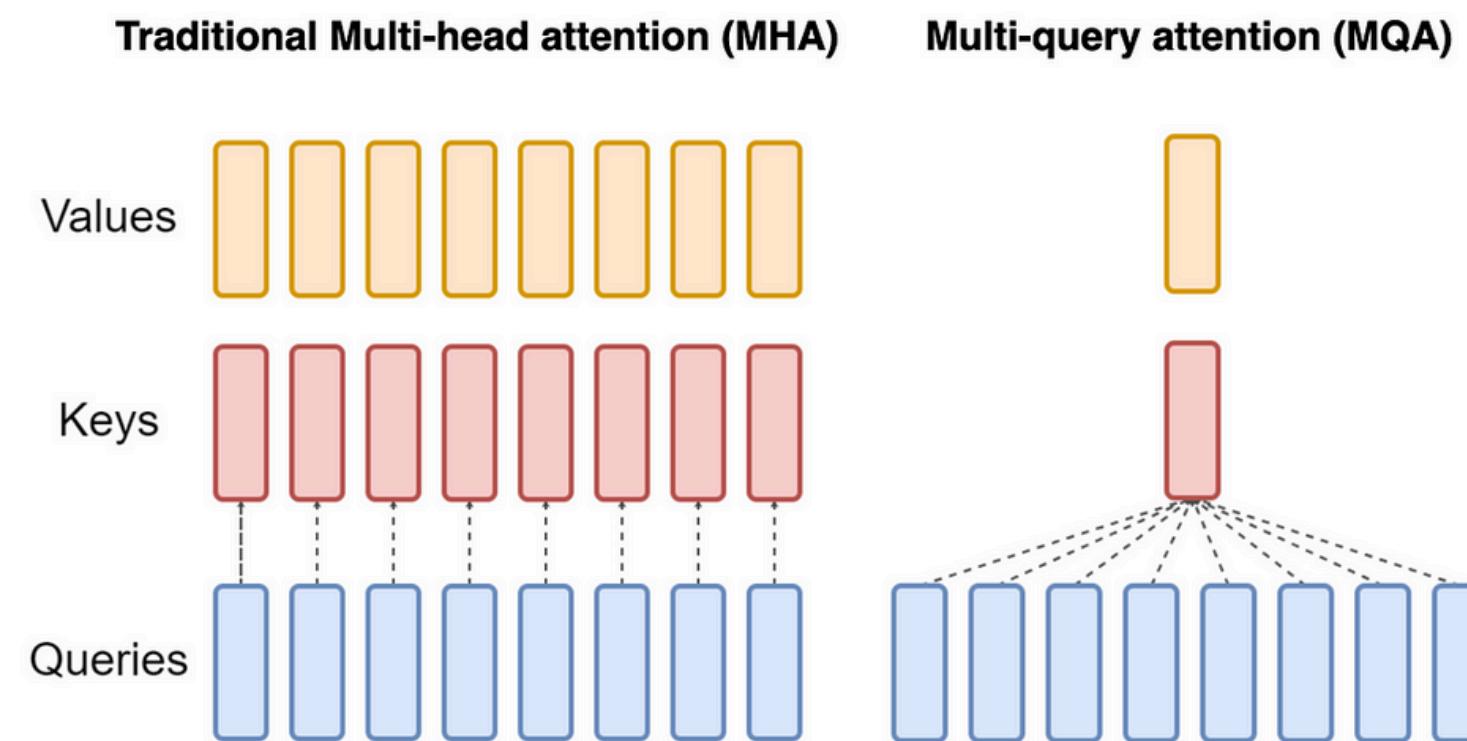
SwiGLU

128,000

RoPE ( $\theta = 500,000$ )

# 1) Grouped Query attention

- Grouped query attention is used in llama where we separate the head dimension from Batch, num\_heads,seq,Dim into Batch,num\_heads//num\_key\_value, num\_key\_value,dim
- Perform attention operation as it is on the newly 5 dimensional data



# Pesudo code of GQA with causal Mask

```
from einops import rearrange,reduce,repeat,einsum
def gqa(query,key,value,group,num_head,dropout=0.):
    scale=query.size(-1)**0.5
    query=query/scale
    b,hq,t,d=query.shape
    b,hk,s,d=key.shape
    num_head_groups= num_head//group

    query=rearrange(query,"b (h g) n d -> b g h n d",g=num_head_groups)
    attn=einsum(query, key, "b g h n d,b h s d -> b g h n s")
    mask=torch.ones((b,t,s),device=query.device,dtype=torch.bool).tril_()
    if mask is not None:
        if mask.ndim == 2:
            mask = rearrange(mask, "b s -> b () () () s")
        elif mask.ndim == 3:
            mask = rearrange(mask, "b n s -> b () () n s")
        attn.masked_fill_(~mask,torch.finfo(attn.dtype).min)
    attn=F.softmax(attn,dim=-1)
    attn=F.dropout(attn,p=dropout)
    attn=einsum(attn,value,"b g h t s, b h s d -> b g h t d")
    attn=rearrange(attn,"b g h t d -> b (g h) t d")
    return attn
```

# Absolute position embedding to relative

In Standard Transformer

$$Q = w_Q(x + PE)$$

$$K = w_K(x + PE)$$

$$Q^T K = \cancel{Q^T Q} (x^T w_Q^T + PE_Q^T w_Q^T) (w_K x_k + w_K PE_k)$$

$$= x^T w_Q^T w_K x_k + x^T w_Q^T w_K P_E_k + \cancel{w_Q^T w_K x_k}$$

$$+ \cancel{P_E_Q^T w_Q^T w_K P_E_k} \rightarrow \text{Relative}$$

$\sqrt{T}$

$\sqrt{T}, \sqrt{V}$  make it learnable that's how Inception on Relative Position Embedding

## 2) Relative position encoding Starting

- Relative position embedding where we would add position vectors on each key and query value this converts the equation into 2set of different attention equation which are needed to be added.
- K is a hyper parameter where it clips the distance matrix between query and key/ value
- more on that in the next slides

# Pesudo Code

```
class Relative_position_embedding(nn.Module):

    def __init__(self,head_dim,k,device):
        super().__init__()
        self.head_dim=head_dim
        self.k=k

        self.device=device
        self.position=nn.EmbeddingLayer(self.k*2+1,self.head_dim)

    def forward(self,q_len,k_len):
        vec_q=torch.arange(q_len)
        vec_k=torch.arange(k_len)
        distance_mat= vec_k[None,:,:] - vec_q[:,None]
        clipped_distance=torch.clamp(distance_mat,-self.k,self.k)
        final_mat= clipped_distance + self.k
        final_mat = torch.LongTensor(final_mat).to(self.device)
        embeddings=self.position[final_mat].to(self.device)

    return embeddings
```

# Pesudo code on attention

```
"""Relational Positional embedding"""
if self.relative_pos:
    len_q=query.shape[2]
    len_k=key.shape[2]
    len_v=value.shape[2]

    relative_positionk=self.relative_position_k(len_q,len_k)
    relative_positionv=self.relative_position_v(len_q,len_v)

if self.relative_pos:
    r_q=rearrange(tensor=query,pattern='B N T D -> T (B N) D')

    attn2=(r_q@relative_positionk.transpose(1,2)).transpose(0,1)

    attn2=rearrange(tensor=attn2,pattern="(B N) Q K -> B N Q K",N=self.num_heads)

    mask=torch.ones((len_q,len_k),device=query.device,dtype=torch.bool).tril_() #casual Mask

    attn2=attn2.masked_fill(mask[:len_q,:len_k]==0,float('-inf')) #if causal

    scale=(self.dim//self.num_heads)**-0.5
    attn2=attn2*scale
    attn2=F.softmax(attn2,dim=-1)
    attn2=rearrange(tensor=attn2,pattern="B N Q K -> Q (B N) K")
    attn2=(attn2@relative_positionv).transpose(0,1)
    attn2=rearrange(tensor=attn2,pattern="(B N) Q D -> B N Q D",N=self.num_heads)

    attn2=F.dropout(attn2,p=self.attention_dropout)

attn = attn1 + attn2
```

# RoPE (Rotational Position Embedding)

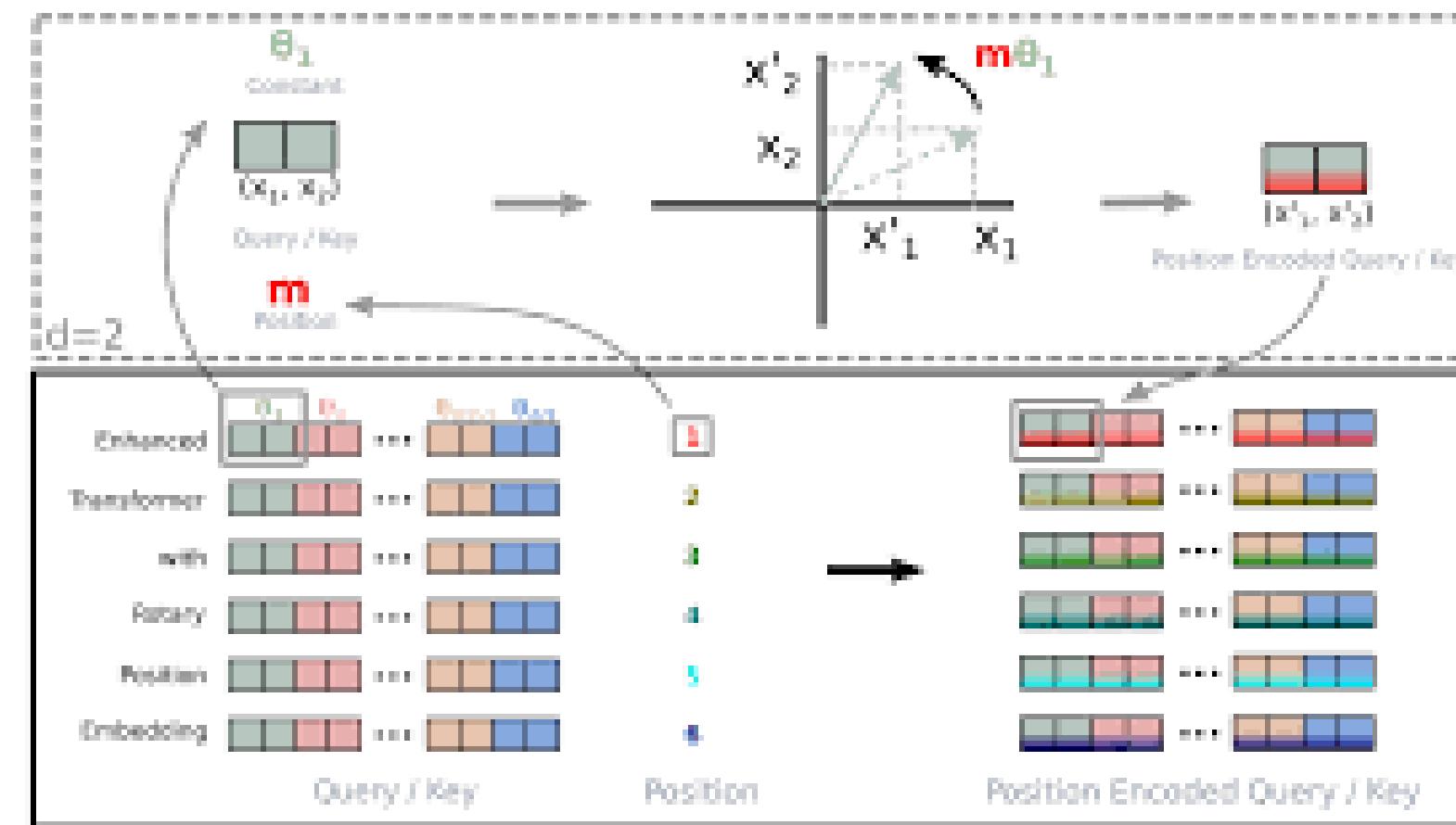


Figure 1: Implementation of Rotary Position Embedding(RoPE).

$$\begin{aligned}
f_q(x_m, m) &= (\mathbf{W}_q x_m) e^{im\theta} \\
f_k(x_n, n) &= (\mathbf{W}_k x_n) e^{in\theta} \\
g(x_m, x_n, m - n) &= \operatorname{Re}[(\mathbf{W}_q x_m)(\mathbf{W}_k x_n)^* e^{i(m-n)\theta}]
\end{aligned} \tag{12}$$

where  $\operatorname{Re}[\cdot]$  is the real part of a complex number and  $(\mathbf{W}_k x_n)^*$  represents the conjugate complex number of  $(\mathbf{W}_k x_n)$ .  $\theta \in \mathbb{R}$  is a preset non-zero constant. We can further write  $f_{\{q,k\}}$  in a multiplication matrix:

$$f_{\{q,k\}}(x_m, m) = \begin{pmatrix} \cos m\theta & -\sin m\theta \\ \sin m\theta & \cos m\theta \end{pmatrix} \begin{pmatrix} W_{\{q,k\}}^{(11)} & W_{\{q,k\}}^{(12)} \\ W_{\{q,k\}}^{(21)} & W_{\{q,k\}}^{(22)} \end{pmatrix} \begin{pmatrix} x_m^{(1)} \\ x_m^{(2)} \end{pmatrix} \tag{13}$$

### 3.2.2 General form

In order to generalize our results in 2D to any  $x_i \in \mathbb{R}^d$  where  $d$  is even, we divide the  $d$ -dimension space into  $d/2$  sub-spaces and combine them in the merit of the linearity of the inner product, turning  $f_{\{q,k\}}$  into:

$$f_{\{q,k\}}(x_m, m) = \mathbf{R}_{\Theta,m}^d \mathbf{W}_{\{q,k\}} x_m \quad (14)$$

where

$$\mathbf{R}_{\Theta,m}^d = \begin{pmatrix} \cos m\theta_1 & -\sin m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ \sin m\theta_1 & \cos m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cos m\theta_2 & -\sin m\theta_2 & \cdots & 0 & 0 \\ 0 & 0 & \sin m\theta_2 & \cos m\theta_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \cos m\theta_{d/2} & -\sin m\theta_{d/2} \\ 0 & 0 & 0 & 0 & \cdots & \sin m\theta_{d/2} & \cos m\theta_{d/2} \end{pmatrix} \quad (15)$$

is the rotary matrix with pre-defined parameters  $\Theta = \{\theta_i = 10000^{-2(i-1)/d}, i \in [1, 2, \dots, d/2]\}$ . A graphic illustration of RoPE is shown in Figure (1). Applying our RoPE to self-attention in Equation (2), we obtain:

$$q_m^\top k_n = (\mathbf{R}_{\Theta,m}^d \mathbf{W}_q x_m)^\top (\mathbf{R}_{\Theta,n}^d \mathbf{W}_k x_n) = x^\top \mathbf{W}_q \mathbf{R}_{\Theta,n-m}^d \mathbf{W}_k x_n \quad (16)$$

### 3.4.2 Computational efficient realization of rotary matrix multiplication

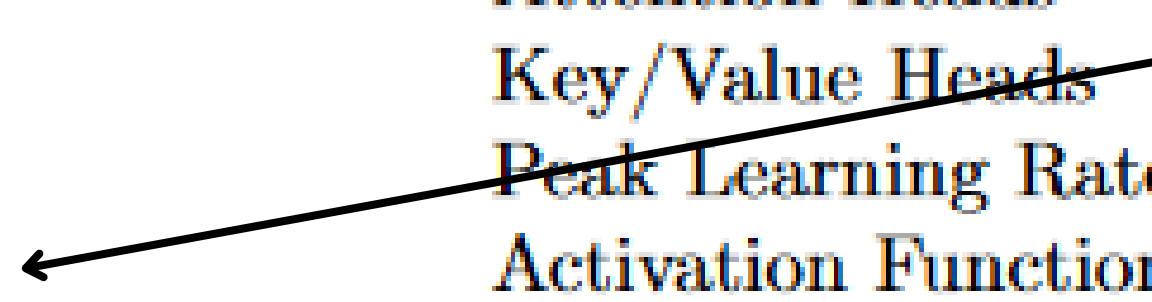
Taking the advantage of the sparsity of  $R_{\Theta,m}^d$  in Equation (15), a more computational efficient realization of a multiplication of  $R_{\Theta}^d$  and  $x \in \mathbb{R}^d$  is:

$$R_{\Theta,m}^d x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \vdots \\ x_{d-1} \\ x_d \end{pmatrix} \otimes \begin{pmatrix} \cos m\theta_1 \\ \cos m\theta_1 \\ \cos m\theta_2 \\ \cos m\theta_2 \\ \vdots \\ \cos m\theta_{d/2} \\ \cos m\theta_{d/2} \end{pmatrix} + \begin{pmatrix} -x_2 \\ x_1 \\ -x_4 \\ x_3 \\ \vdots \\ -x_d \\ x_{d-1} \end{pmatrix} \otimes \begin{pmatrix} \sin m\theta_1 \\ \sin m\theta_1 \\ \sin m\theta_2 \\ \sin m\theta_2 \\ \vdots \\ \sin m\theta_{d/2} \\ \sin m\theta_{d/2} \end{pmatrix} \quad (34)$$

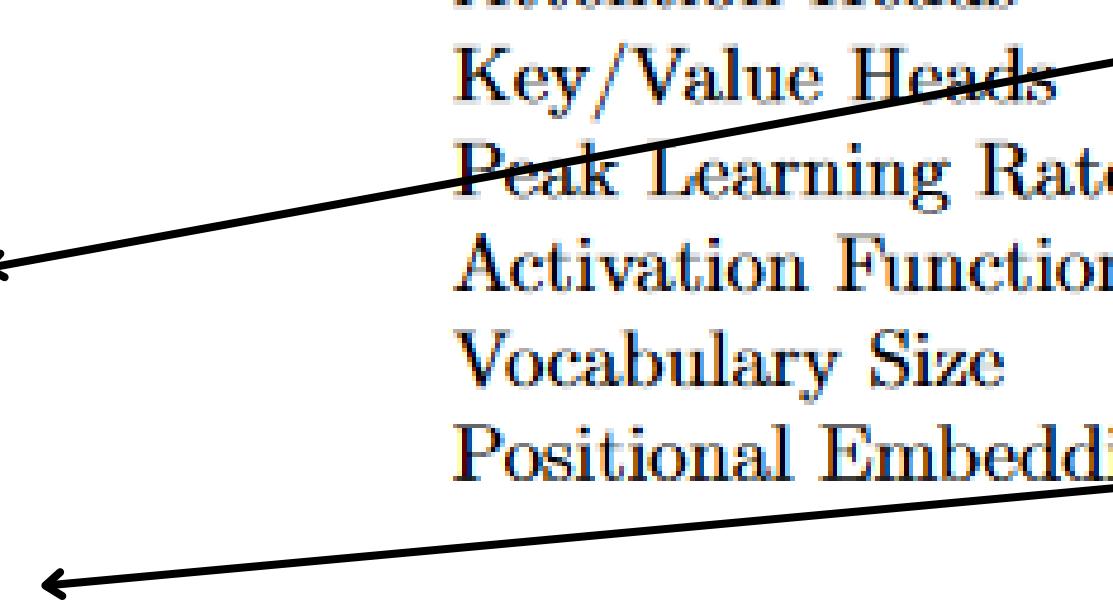
# Model architecture details on the paper

	<b>8B</b>	<b>70B</b>	<b>405B</b>
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	$3 \times 10^{-4}$	$1.5 \times 10^{-4}$	$8 \times 10^{-5}$
Activation Function		SwiGLU	
Vocabulary Size		128,000	
Positional Embeddings		RoPE ( $\theta = 500,000$ )	

!

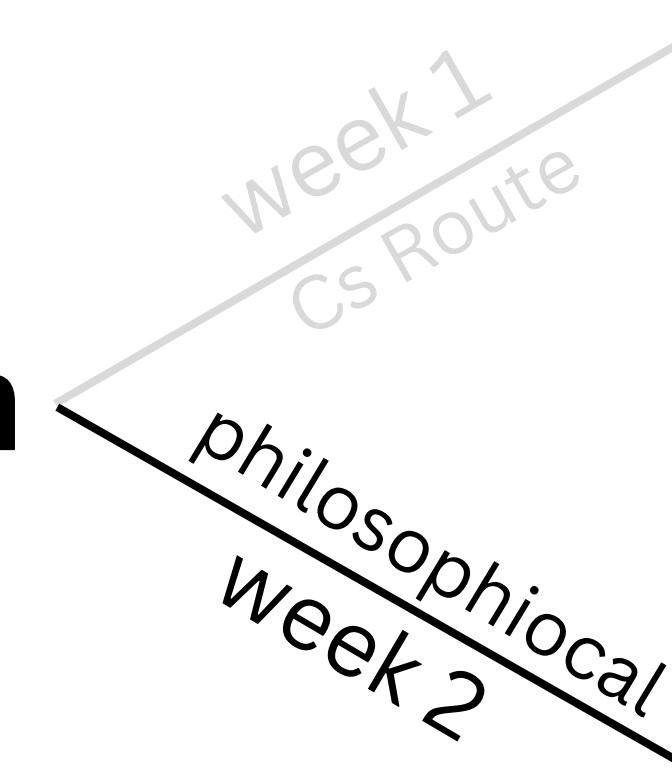


!



# The Llama 3.1 Herd of Models

Paper Reading session

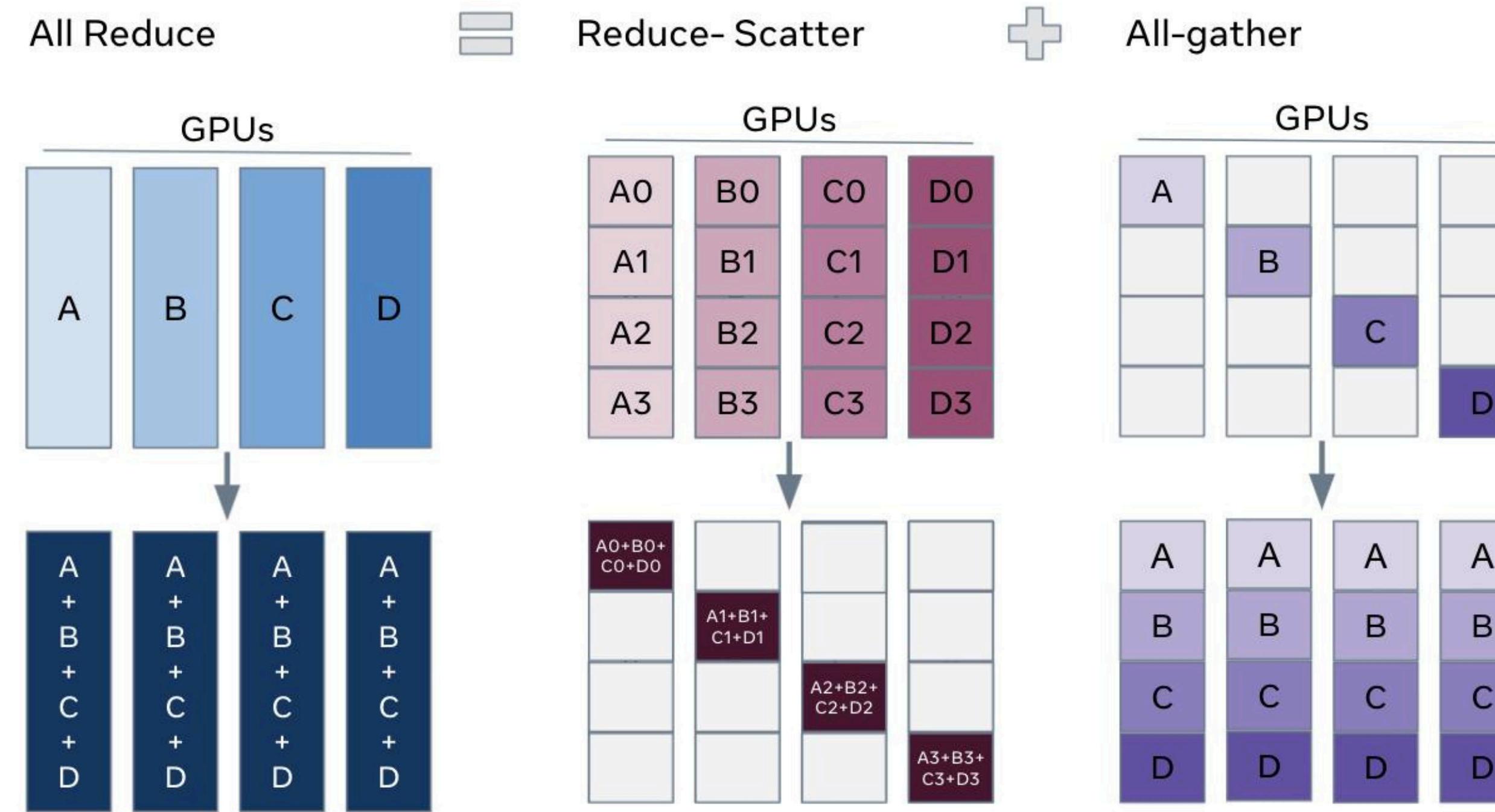


Basics and Building  
Blocks of llama

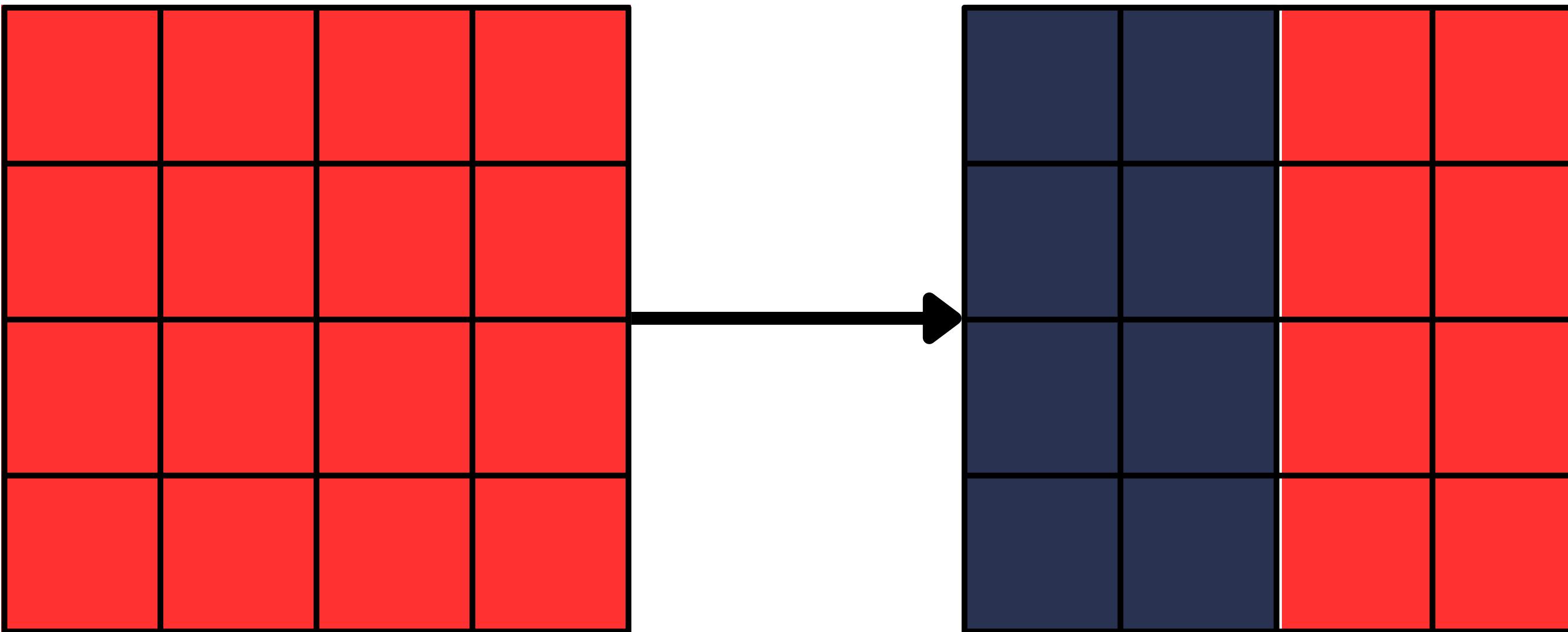
Bench marks, Training  
recipes and Impact on  
LLM space

# **Parallelism (continued)**

# operations on distributed environment

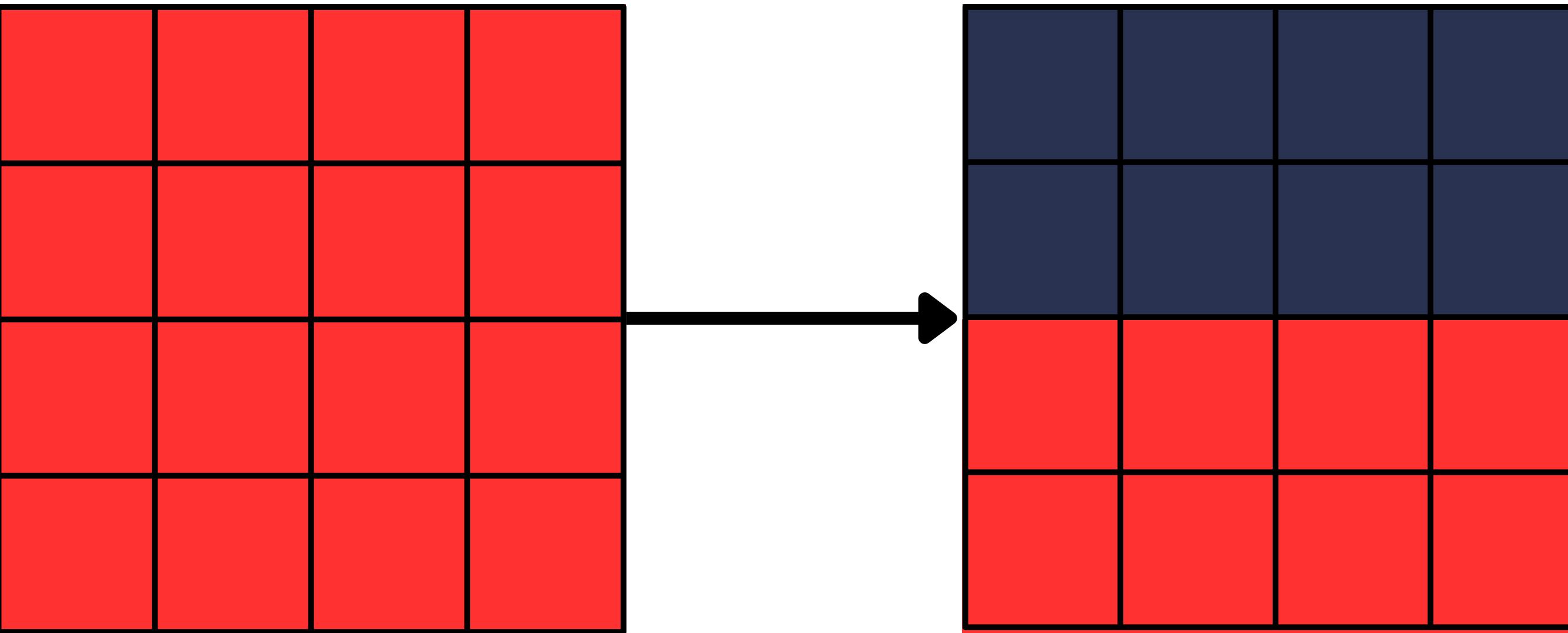


# Sharding on Column



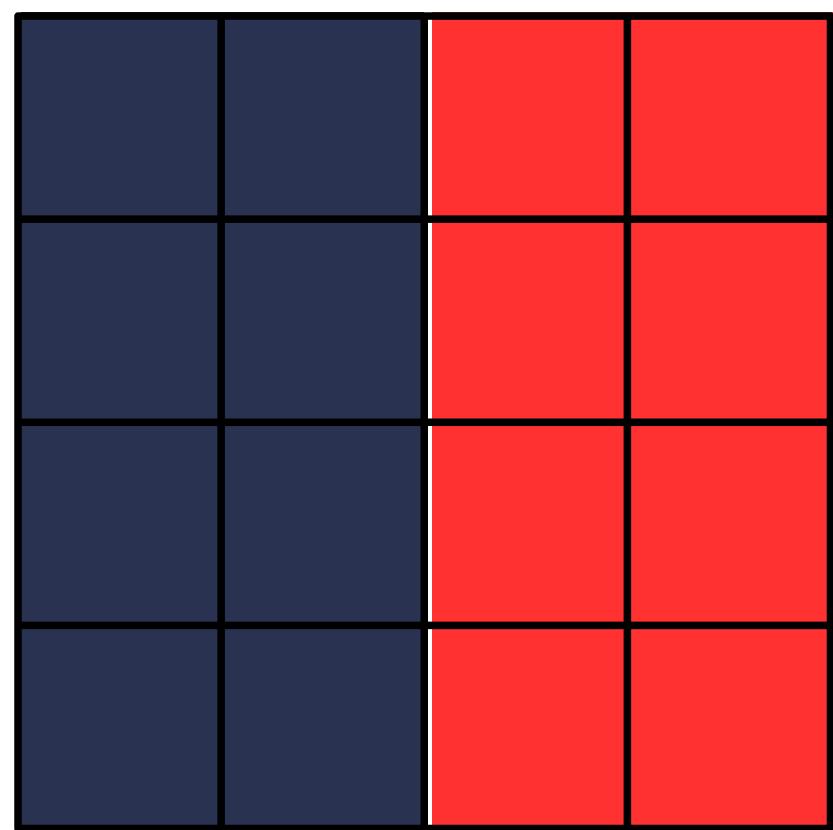
$X \longrightarrow [X_1, X_2]$

# Sharding on Row

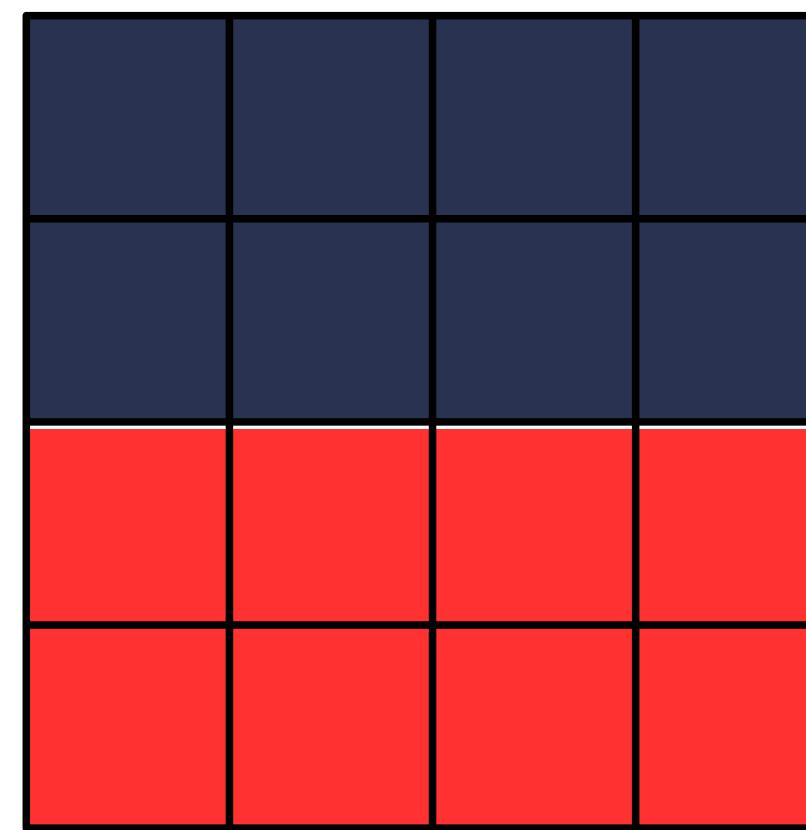


$x \longrightarrow [x_1, x_2]$

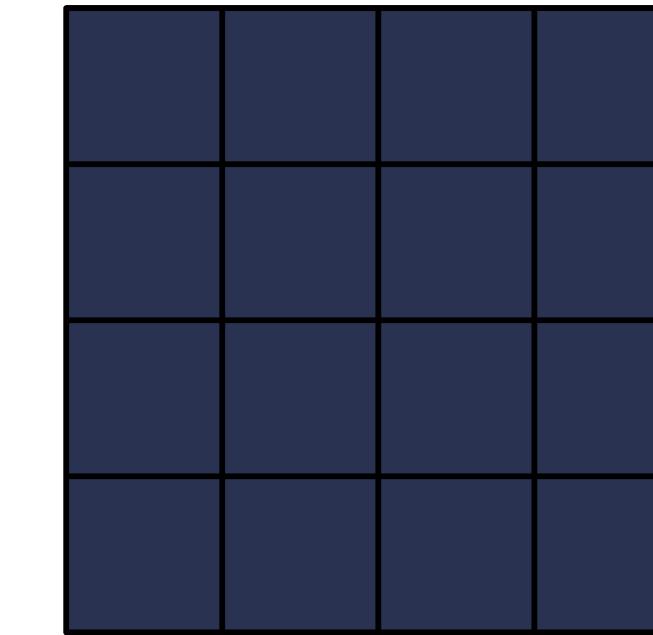
For an Operation  $Z=XA$  we would shard as



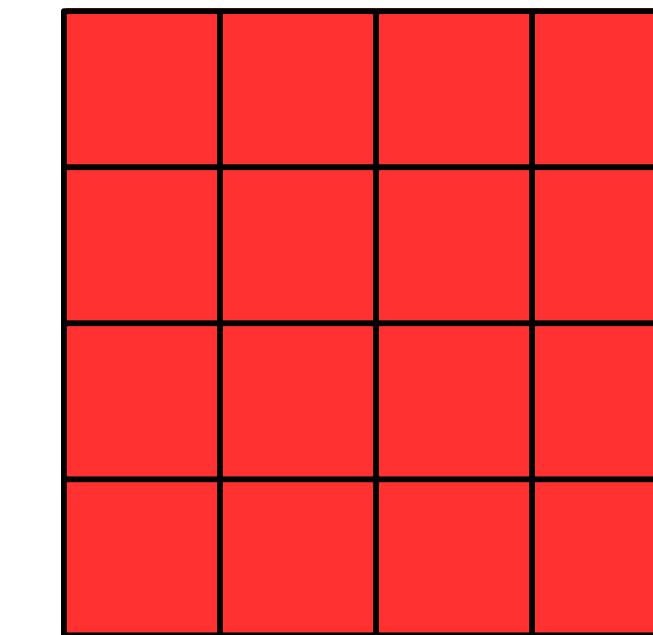
$\times$



=



+



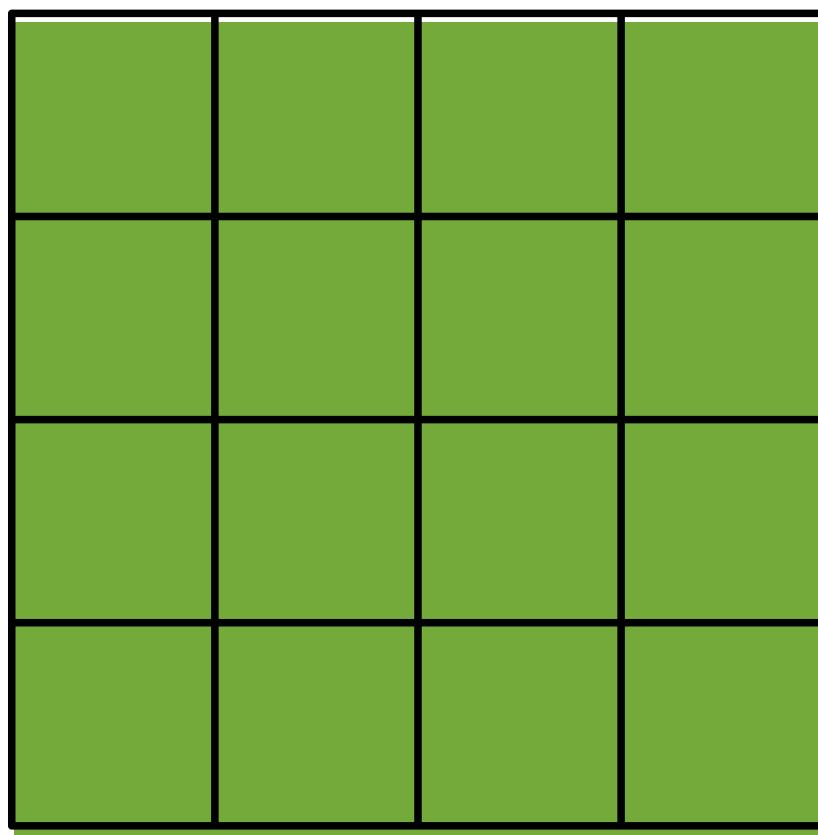
$[X_1, X_2]$

$[A_1, A_2]$

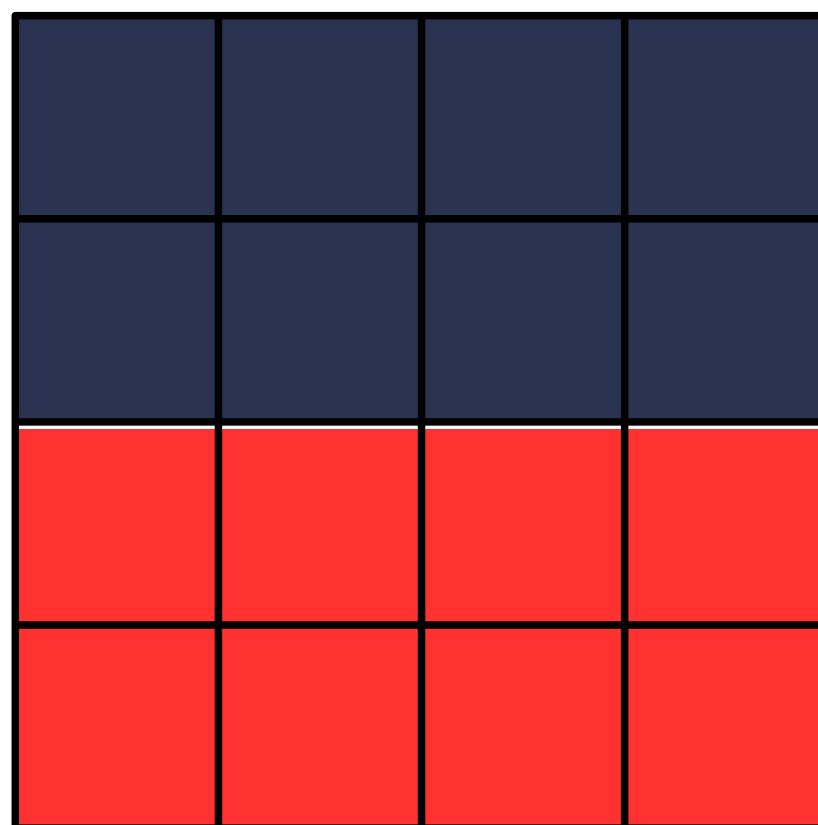
$X_1 A_1$

$X_2 A_2$

For an Operation  $Z=XA$  we can also shard as



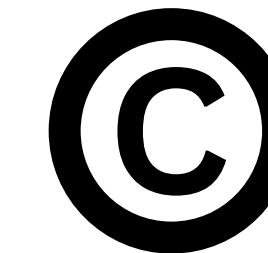
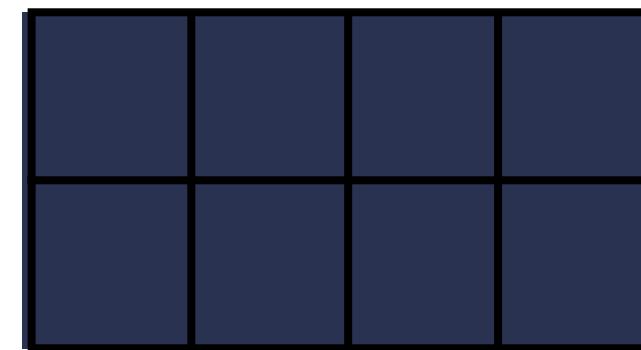
**X**



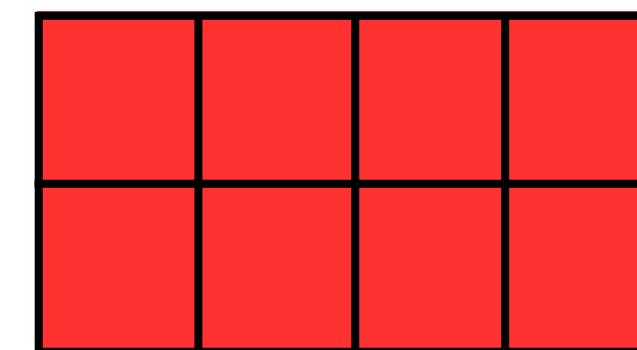
**X1**

**[A1,A2]**

=

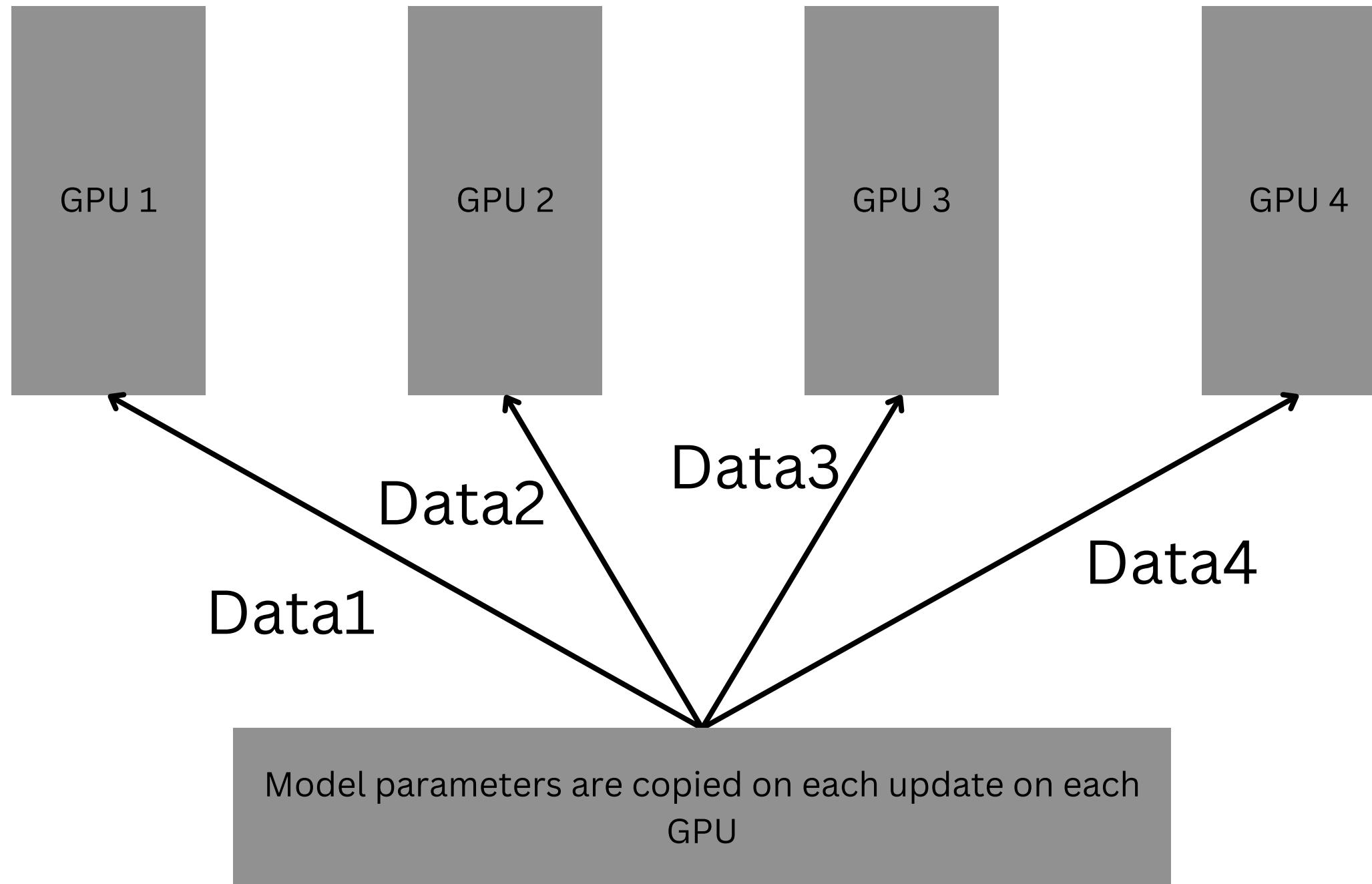


Concatenate on 1st axis of A matrix

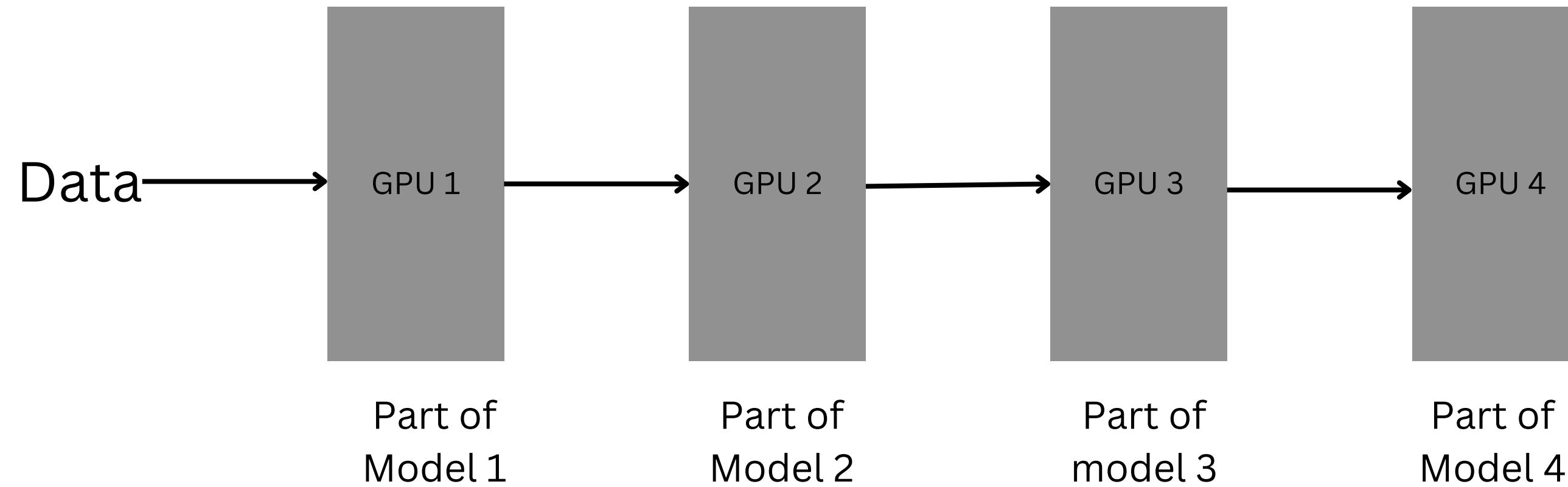


**X2A2**

# DP - Data parallel

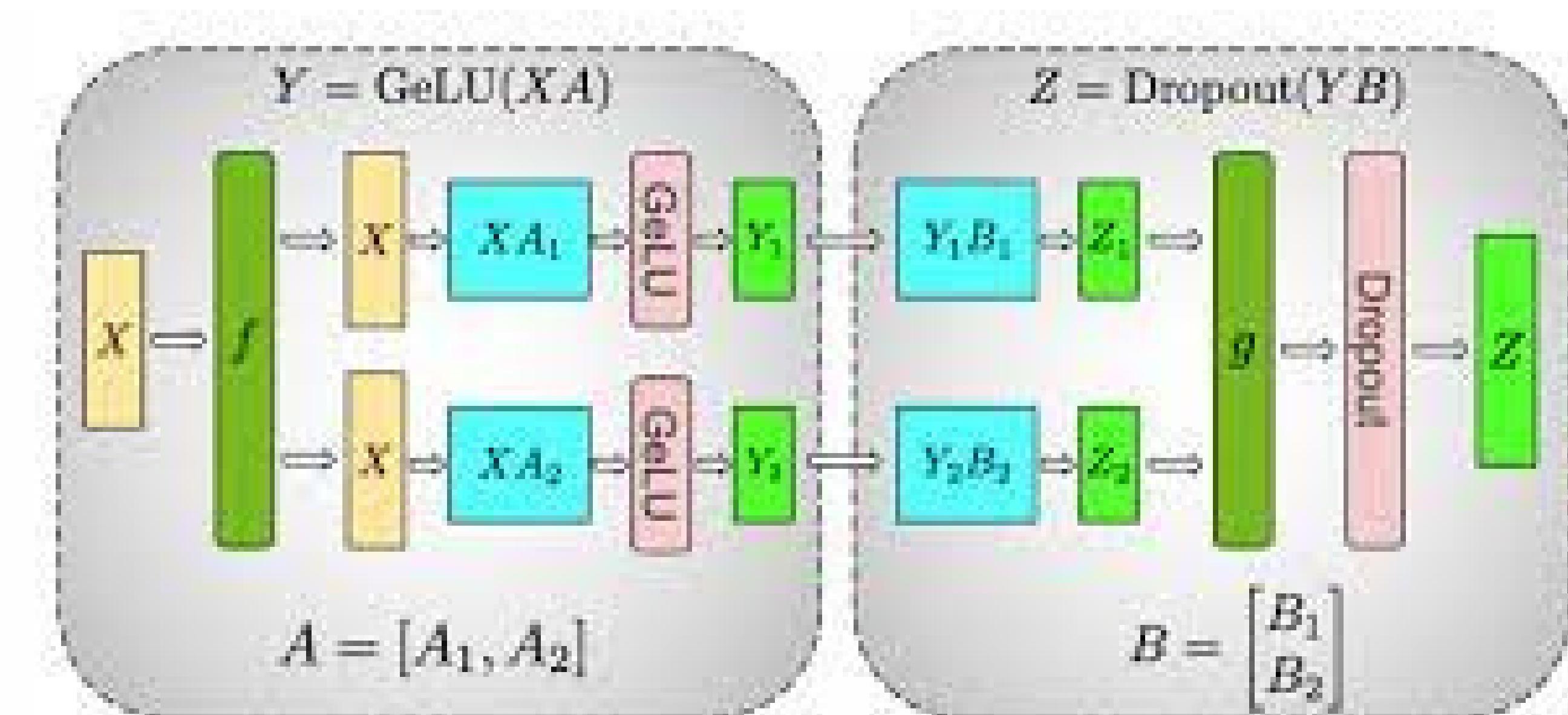


# FSDP - Fully Sharded Data parallel

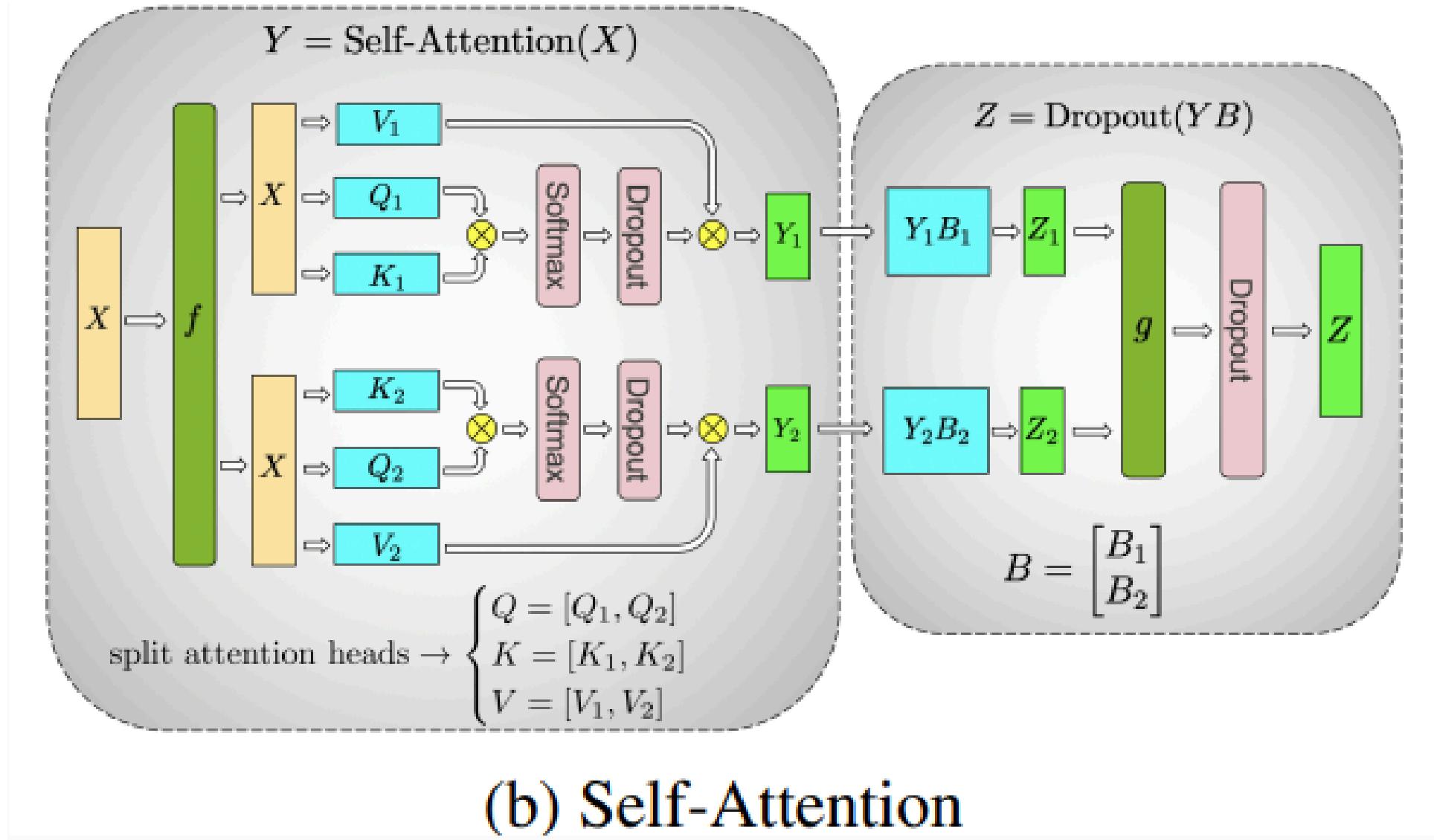


Here Models are split into parts  
and each part of the Model are  
placed in separate GPUs

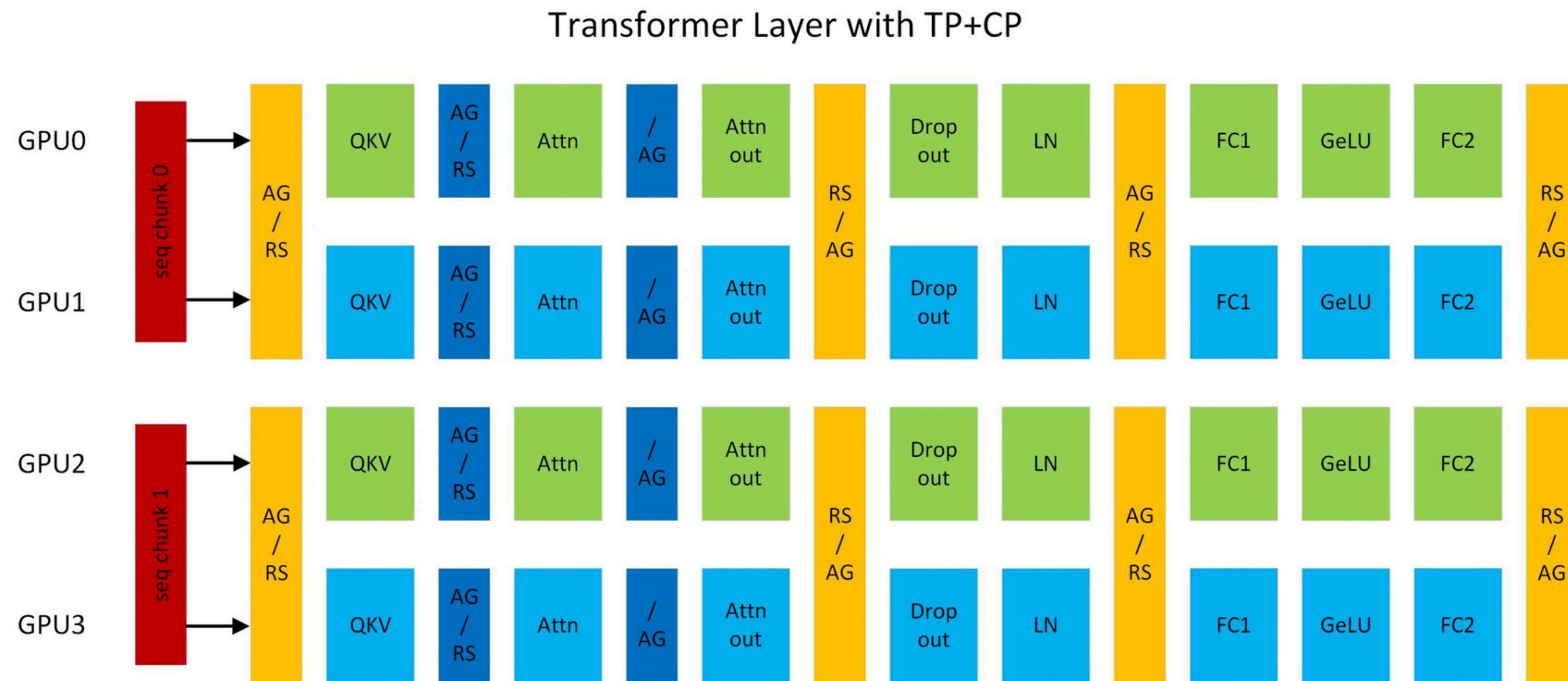
# Tensor Parallel



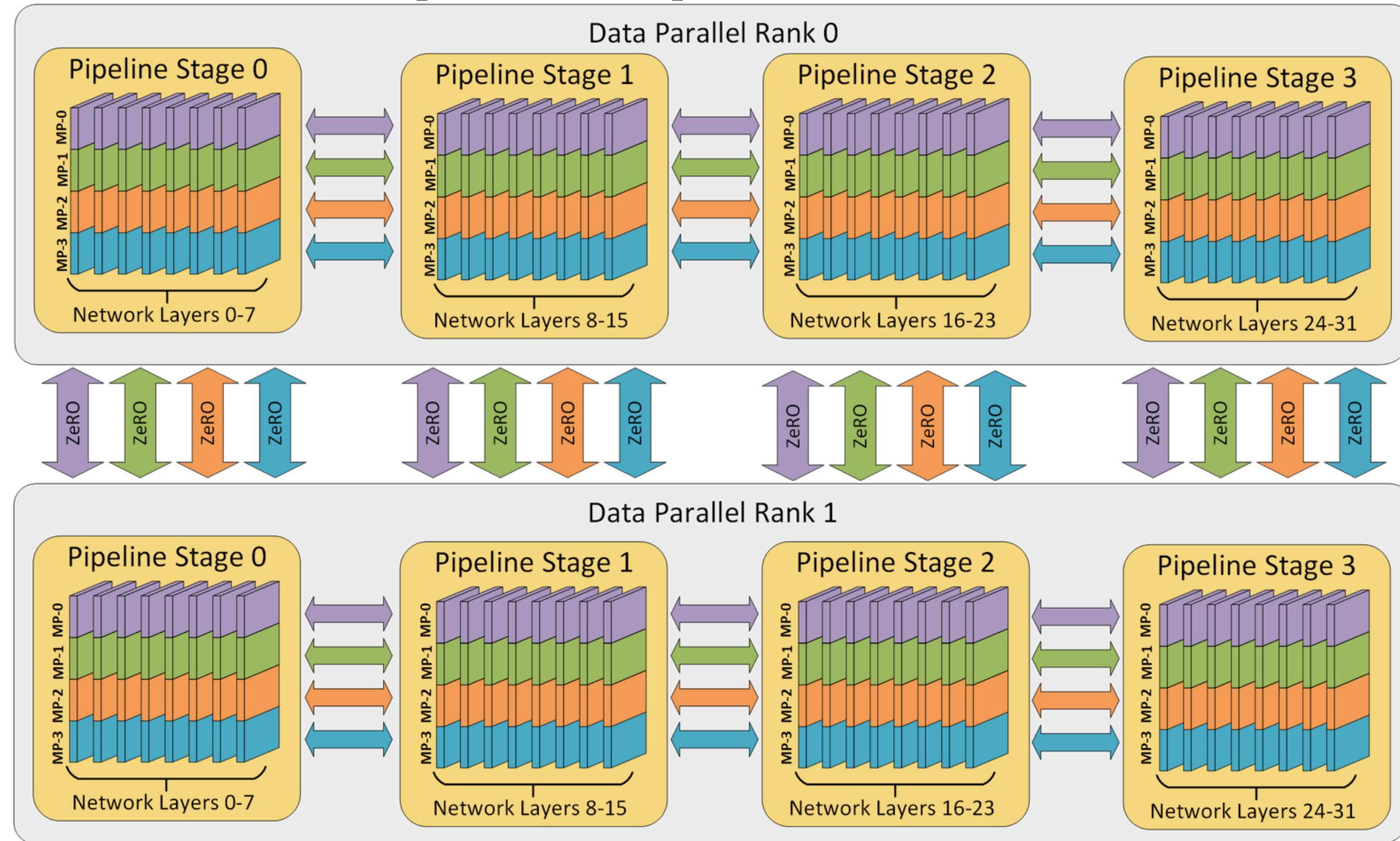
(a) MLP



# Context parallelism



# Pipeline parallelism





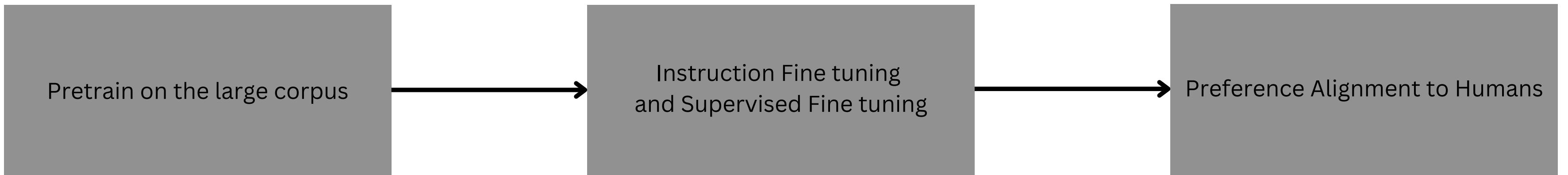
Now on Meta's Lamma paper

	N $\in$ [0, M]																																										
PP rank0	0	1	2	3	4	0	1	2	3	4	5	6	0	7	1	8	2	9	3	5	4	6	0	7	1	8	2	9	3	4	5	6	7	8	9	5	6	7	8	9			
PP rank1	0	1	2	3	4	0	1	2	3	4	0	5	1	6	2	7	3	8	4	9	0	5	1	6	2	7	3	8	4	9	5	6	7	8	9	5	6	7	8	9			
PP rank2	0	1	2	3	4	0	1	2	3	4	0	3	1	4	2	5	3	6	4	7	0	8	1	9	2	5	3	6	4	7	5	8	6	9	7	8	9	5	6	7	8	9	
PP rank3	0	1	2	3	4	0	0	1	2	3	3	4	4	5	0	6	1	7	2	8	3	9	4	5	5	6	6	7	7	8	6	9	9	5	6	7	8	9					

Time       Stage 0-3 forward       Stage 4-7 forward       Stage 0-3 backward       Stage 4-7 backward

**Back to LLMs Training**

# General Pipe Line of LLM



# Typical way of Aligning the Model to human alignment

- First we will pre-train the LLM and then we would do either IFT or SFT on the particular down stream task(Continual pre-training )
- To make it align with the human's preference we will train the reward model and then assign the most preferred outcome (for humans) with reward and KL divergence
- Rinse and repeat

# **Small Intro on KL divergence**

KL Divergence is a Metric to Evaluate Similarity  
Between distribution  $P(x) \& Q(x)$

If  $D_{KL}(P(x) || Q(x)) = 0$  then  $P(x) = Q(x)$

$D_{KL}(P(x) || Q(x)) \neq D_{KL}(Q(x) || P(x))$

$$D_{KL}(P(x) || Q(x)) = \sum P(x) \log \left( \frac{P(x)}{Q(x)} \right) \text{ - discrete}$$

or

$$= \int P(x) \log \left( \frac{P(x)}{Q(x)} \right) dx$$

$\Sigma$

Continuous

Now am just using Continuous because of generalization

$$= \int P(x) \log \left( \frac{P(x)}{Q(x)} \right) dx \Rightarrow \int P(x) \log(P(x)) dx - ①$$

$$- \int P(x) \log(Q(x)) dx - ③$$

②

### Step 1

**Collect demonstration data, and train a supervised policy.**

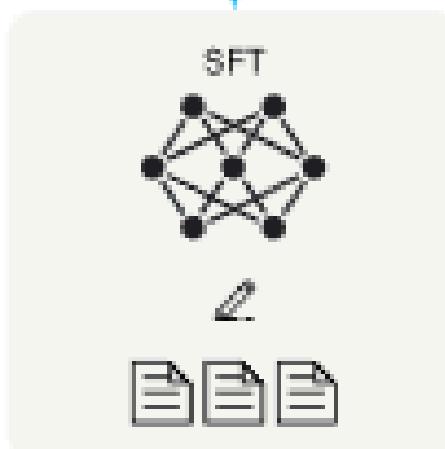
A prompt is sampled from our prompt dataset.



A labeler demonstrates the desired output behavior.



This data is used to fine-tune GPT-3 with supervised learning.



### Step 2

**Collect comparison data, and train a reward model.**

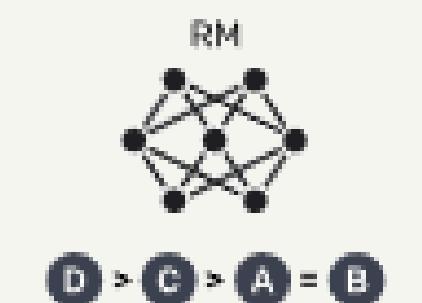
A prompt and several model outputs are sampled.



A labeler ranks the outputs from best to worst.



This data is used to train our reward model.



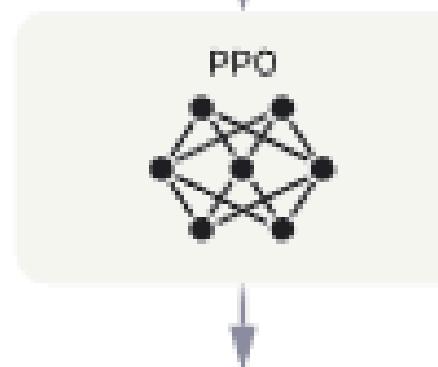
### Step 3

**Optimize a policy against the reward model using reinforcement learning.**

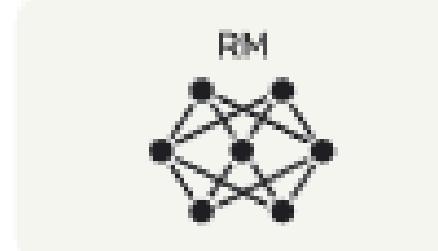
A new prompt is sampled from the dataset.



The policy generates an output.



The reward model calculates a reward for the output.



The reward is used to update the policy using PPO.



$$\text{objective}(\phi) = E_{(x,y) \sim D_{\pi_\phi^{\text{RL}}}} \left[ r_\theta(x, y) - \beta \log \left( \pi_\phi^{\text{RL}}(y \mid x) / \pi^{\text{SFT}}(y \mid x) \right) \right] + \\ \gamma E_{x \sim D_{\text{pretrain}}} \left[ \log(\pi_\phi^{\text{RL}}(x)) \right]$$

$$\max_{\pi} \mathbb{E}_{x \sim \mathcal{D}, y \sim \pi} [r(x, y)] - \beta \mathbb{D}_{\text{KL}} [\pi(y|x) || \pi_{\text{ref}}(y|x)] \quad (11)$$

under any reward function  $r(x, y)$ , reference model  $\pi_{\text{ref}}$  and a general non-parametric policy class. We now have:

$$\begin{aligned} & \max_{\pi} \mathbb{E}_{x \sim \mathcal{D}, y \sim \pi} [r(x, y)] - \beta \mathbb{D}_{\text{KL}} [\pi(y|x) || \pi_{\text{ref}}(y|x)] \\ &= \max_{\pi} \mathbb{E}_{x \sim \mathcal{D}} \mathbb{E}_{y \sim \pi(y|x)} \left[ r(x, y) - \beta \log \frac{\pi(y|x)}{\pi_{\text{ref}}(y|x)} \right] \\ &= \min_{\pi} \mathbb{E}_{x \sim \mathcal{D}} \mathbb{E}_{y \sim \pi(y|x)} \left[ \log \frac{\pi(y|x)}{\pi_{\text{ref}}(y|x)} - \frac{1}{\beta} r(x, y) \right] \\ &= \min_{\pi} \mathbb{E}_{x \sim \mathcal{D}} \mathbb{E}_{y \sim \pi(y|x)} \left[ \log \frac{\pi(y|x)}{\frac{1}{Z(x)} \pi_{\text{ref}}(y|x) \exp\left(\frac{1}{\beta} r(x, y)\right)} - \log Z(x) \right] \quad (12) \end{aligned}$$

where we have partition function:

$$Z(x) = \sum_y \pi_{\text{ref}}(y|x) \exp\left(\frac{1}{\beta} r(x, y)\right).$$

which is a valid probability distribution as  $\pi^*(y|x) \geq 0$  for all  $y$  and  $\sum_y \pi^*(y|x) = 1$ . Since  $Z(x)$  is not a function of  $y$ , we can then re-organize the final objective in Eq 12 as:

$$\min_{\pi} \mathbb{E}_{x \sim \mathcal{D}} \left[ \mathbb{E}_{y \sim \pi(y|x)} \left[ \log \frac{\pi(y|x)}{\pi^*(y|x)} \right] - \log Z(x) \right] = \quad (13)$$

$$\min_{\pi} \mathbb{E}_{x \sim \mathcal{D}} [\text{D}_{\text{KL}}(\pi(y|x) \parallel \pi^*(y|x)) - \log Z(x)] \quad (14)$$

Now, since  $Z(x)$  does not depend on  $\pi$ , the minimum is achieved by the policy that minimizes the first KL term. Gibbs' inequality tells us that the KL-divergence is minimized at 0 if and only if the two distributions are identical. Hence we have the optimal solution:

$$\pi(y|x) = \pi^*(y|x) = \frac{1}{Z(x)} \pi_{\text{ref}}(y|x) \exp \left( \frac{1}{\beta} r(x, y) \right) \quad (15)$$

for all  $x \in \mathcal{D}$ . This completes the derivation.

It is straightforward to derive the DPO objective under the Bradley-Terry preference model as have

$$p^*(y_1 \succ y_2 | x) = \frac{\exp(r^*(x, y_1))}{\exp(r^*(x, y_1)) + \exp(r^*(x, y_2))} \quad (1)$$

In Section 4 we showed that we can express the (unavailable) ground-truth reward through corresponding optimal policy:

$$r^*(x, y) = \beta \log \frac{\pi^*(y|x)}{\pi_{\text{ref}}(y|x)} + \beta \log Z(x) \quad (2)$$

Substituting Eq. 17 into Eq. 16 we obtain:

$$\begin{aligned} p^*(y_1 \succ y_2 | x) &= \frac{\exp\left(\beta \log \frac{\pi^*(y_1|x)}{\pi_{\text{ref}}(y_1|x)} + \beta \log Z(x)\right)}{\exp\left(\beta \log \frac{\pi^*(y_1|x)}{\pi_{\text{ref}}(y_1|x)} + \beta \log Z(x)\right) + \exp\left(\beta \log \frac{\pi^*(y_2|x)}{\pi_{\text{ref}}(y_2|x)} + \beta \log Z(x)\right)} \\ &= \frac{1}{1 + \exp\left(\beta \log \frac{\pi^*(y_2|x)}{\pi_{\text{ref}}(y_2|x)} - \beta \log \frac{\pi^*(y_1|x)}{\pi_{\text{ref}}(y_1|x)}\right)} \\ &= \sigma\left(\beta \log \frac{\pi^*(y_1|x)}{\pi_{\text{ref}}(y_1|x)} - \beta \log \frac{\pi^*(y_2|x)}{\pi_{\text{ref}}(y_2|x)}\right). \end{aligned}$$

The last line is the per-instance loss in Equation 7.

... now we can see the gradient of the loss w.r.t.  $\theta$ .

$$\nabla_{\theta} \mathcal{L}_{\text{DPO}}(\pi_{\theta}; \pi_{\text{ref}}) = -\nabla_{\theta} \mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}} \left[ \log \sigma \left( \beta \log \frac{\pi_{\theta}(y_l|x)}{\pi_{\text{ref}}(y_l|x)} - \beta \log \frac{\pi_{\theta}(y_w|x)}{\pi_{\text{ref}}(y_w|x)} \right) \right] \quad (21)$$

We can rewrite the RHS of Equation 21 as

$$\nabla_{\theta} \mathcal{L}_{\text{DPO}}(\pi_{\theta}; \pi_{\text{ref}}) = -\mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}} \left[ \frac{\sigma'(u)}{\sigma(u)} \nabla_{\theta}(u) \right], \quad (22)$$

where  $u = \beta \log \frac{\pi_{\theta}(y_l|x)}{\pi_{\text{ref}}(y_l|x)} - \beta \log \frac{\pi_{\theta}(y_w|x)}{\pi_{\text{ref}}(y_w|x)}$ .

Using the properties of sigmoid function  $\sigma'(x) = \sigma(x)(1 - \sigma(x))$  and  $\sigma(-x) = 1 - \sigma(x)$ , we obtain the final gradient

$$\begin{aligned} \nabla_{\theta} \mathcal{L}_{\text{DPO}}(\pi_{\theta}; \pi_{\text{ref}}) &= \\ &- \mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}} \left[ \beta \sigma \left( \beta \log \frac{\pi_{\theta}(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \frac{\pi_{\theta}(y_l|x)}{\pi_{\text{ref}}(y_l|x)} \right) \left[ \nabla_{\theta} \log \pi(y_w | x) - \nabla_{\theta} \log \pi(y_l | x) \right] \right], \end{aligned}$$

After using the reward substitution of  $\hat{r}_{\theta}(x, y) = \beta \log \frac{\pi_{\theta}(y|x)}{\pi_{\text{ref}}(y|x)}$  we obtain the final form of the gradient from Section 4.

# DPO vs RL (PPO, REINFORCE, ...)

DPO and PPO are very different optimizers.

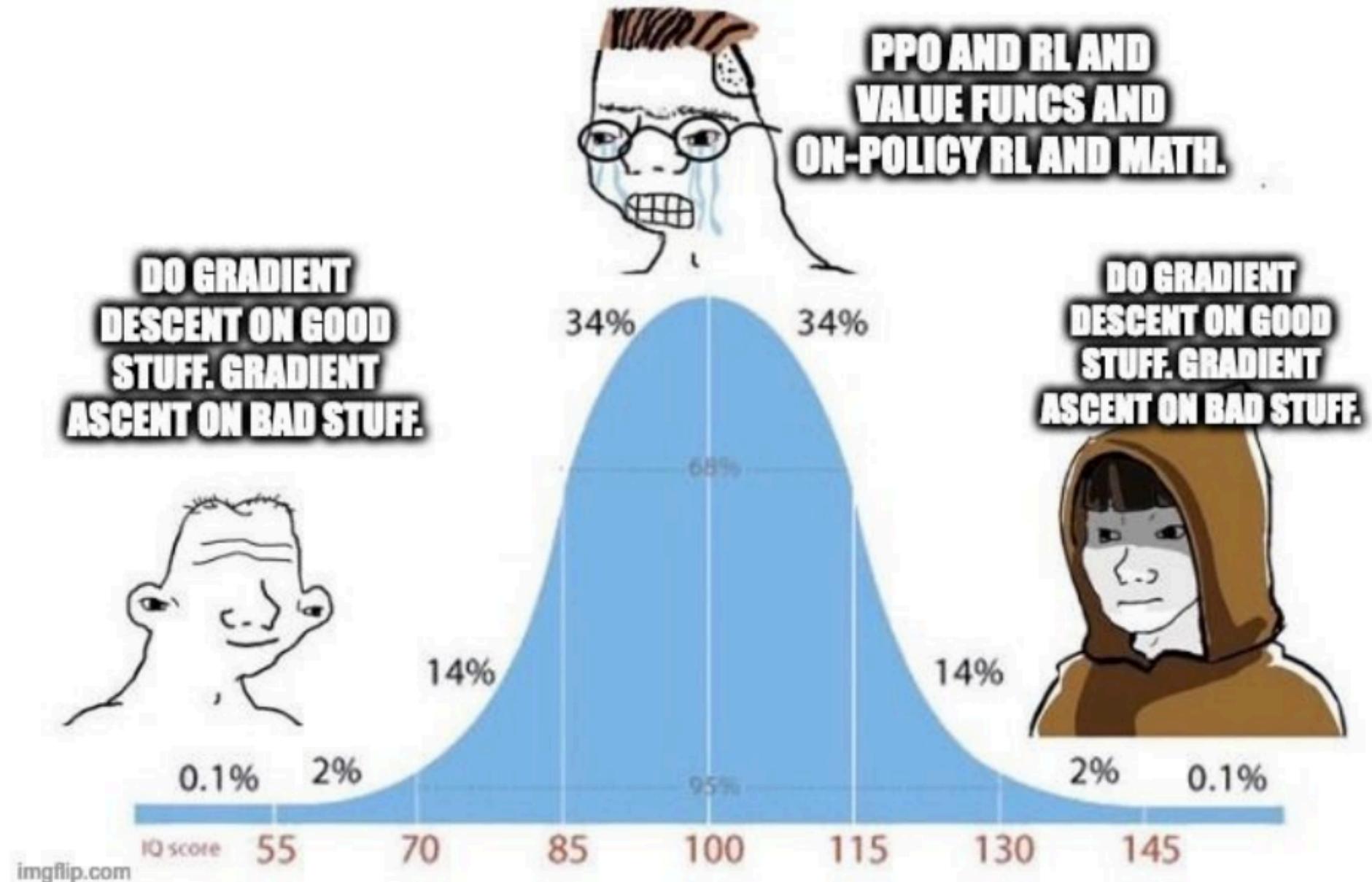
It is learning directly from preferences vs. using RL update rules.

It is also not really online vs offline RL, but that is more muddled.

More discussion:

[https://twitter.com/srush\\_nlp/status/1729896568956895370](https://twitter.com/srush_nlp/status/1729896568956895370),  
<https://www.interconnects.ai/p/the-dpo-debate>,  
<https://www.youtube.com/watch?v=YJMCSVLRUNs>

## LEARNING FROM HUMAN FEEDBACK



Credit Tom Goldstein  
<https://twitter.com/tomgoldsteincs>  
Aligning open language models | Lambert: 66

# **PipeLine of DPO**

**DPO outline.** The general DPO pipeline is as follows: 1) Sample completions  $y_1, y_2 \sim \pi_{\text{ref}}(\cdot | x)$  for every prompt  $x$ , label with human preferences to construct the offline dataset of preferences  $\mathcal{D} = \{x^{(i)}, y_w^{(i)}, y_l^{(i)}\}_{i=1}^N$  and 2) optimize the language model  $\pi_\theta$  to minimize  $\mathcal{L}_{\text{DPO}}$  for the given  $\pi_{\text{ref}}$  and  $\mathcal{D}$  and desired  $\beta$ . In practice, one would like to reuse preference datasets publicly available, rather than generating samples and gathering human preferences. Since the preference datasets are sampled using  $\pi^{\text{SFT}}$ , we initialize  $\pi_{\text{ref}} = \pi^{\text{SFT}}$  whenever available. However, when  $\pi^{\text{SFT}}$  is not available, we initialize  $\pi_{\text{ref}}$  by maximizing likelihood of preferred completions  $(x, y_w)$ , that is,  $\pi_{\text{ref}} = \arg \max_{\pi} \mathbb{E}_{x, y_w \sim \mathcal{D}} [\log \pi(y_w | x)]$ . This procedure helps mitigate the distribution shift between the true reference distribution which is unavailable, and  $\pi_{\text{ref}}$  used by DPO. Further details related to the implementation and hyperparameters can be found in Appendix B.

# Pesudo code of DPO

```
import torch.nn.functional as F

def dpo_loss(pi_logps, ref_logps, yw_idxs, yl_idxs, beta):
    """
        pi_logps: policy logprobs, shape (B,)
        ref_logps: reference model logprobs, shape (B,)
        yw_idxs: preferred completion indices in [0, B-1], shape (T,)
        yl_idxs: dispreferred completion indices in [0, B-1], shape (T,)
        beta: temperature controlling strength of KL penalty

        Each pair of (yw_idxs[i], yl_idxs[i]) represents the
        indices of a single preference pair.
    """
    pi_yw_logps, pi_yl_logps = pi_logps[yw_idxs], pi_logps[yl_idxs]
    ref_yw_logps, ref_yl_logps = ref_logps[yw_idxs], ref_logps[yl_idxs]

    pi_logratios = pi_yw_logps - pi_yl_logps
    ref_logratios = ref_yw_logps - ref_yl_logps

    losses = -F.logsigmoid(beta * (pi_logratios - ref_logratios))
    rewards = beta * (pi_logps - ref_logps).detach()

    return losses, rewards
```

# DPO Summary:

DPO uses Preference Dataset to Align itself to the human preferences with out any explicit reward Model

# Now on to dataset collection and Training

## 3.1.2 Determining the Data Mix

To obtain a high-quality language model, it is essential to carefully determine the proportion of different data sources in the pre-training data mix. Our main tools in determining this data mix are knowledge classification and scaling law experiments.

**Knowledge classification.** We develop a classifier to categorize the types of information contained in our web data to more effectively determine a data mix. We use this classifier to downsample data categories that are over-represented on the web, for example, arts and entertainment.

**Scaling laws for data mix.** To determine the best data mix, we perform scaling law experiments in which we train several small models on a data mix and use that to predict the performance of a large model on that mix (see Section 3.2.1). We repeat this process multiple times for different data mixes to select a new data mix candidate. Subsequently, we train a larger model on this candidate data mix and evaluate the performance of that model on several key benchmarks.

**Data mix summary.** Our final data mix contains roughly 50% of tokens corresponding to general knowledge, 25% of mathematical and reasoning tokens, 17% code tokens, and 8% multilingual tokens.

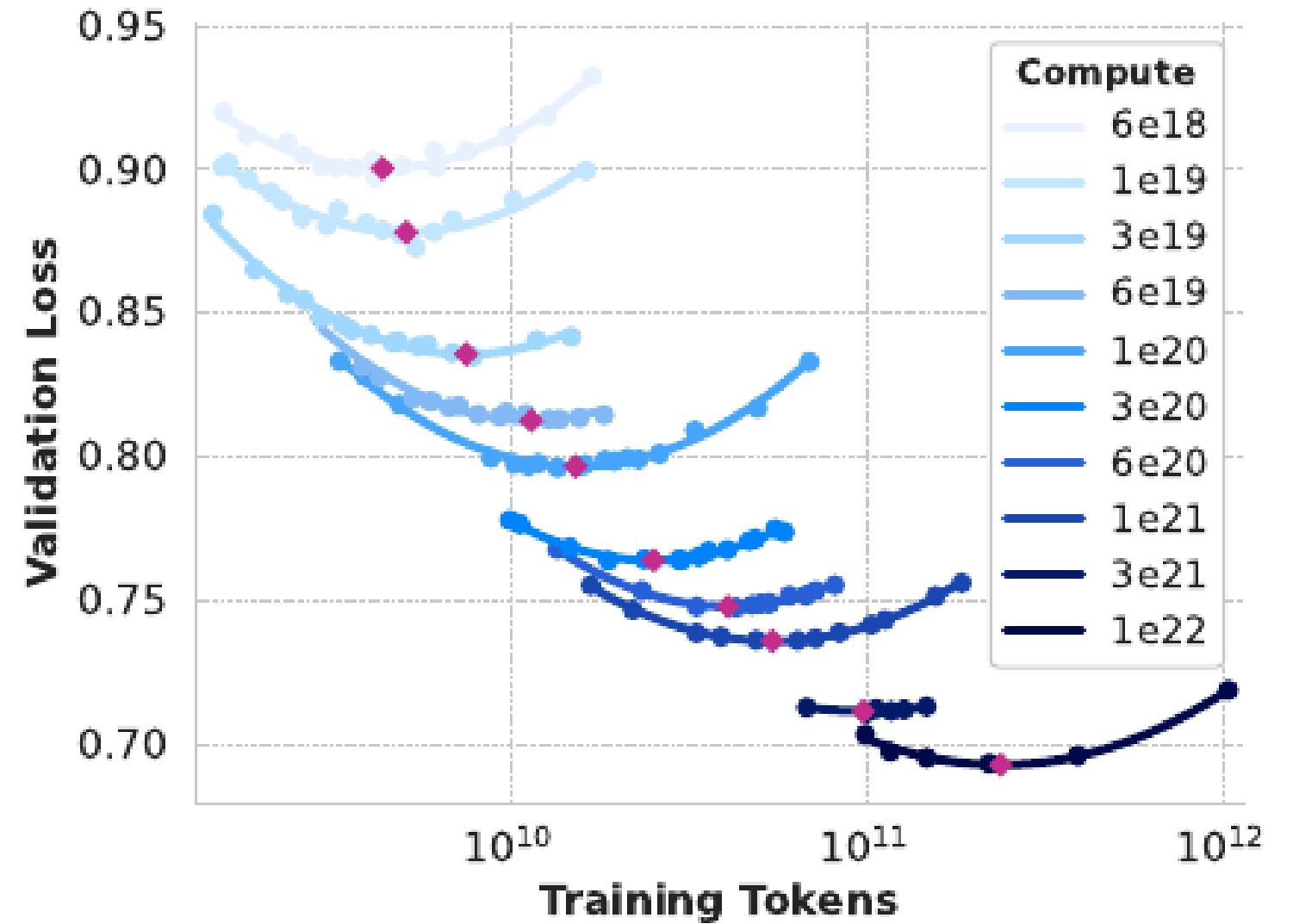
We develop scaling laws (Hoffmann et al., 2022; Kaplan et al., 2020) to determine the optimal model size for our flagship model given our pre-training compute budget. In addition to determining the optimal model size, a major challenge is to forecast the flagship model’s performance on downstream benchmark tasks, due to a couple of issues: (1) Existing scaling laws typically predict only next-token prediction loss rather than specific benchmark performance. (2) Scaling laws can be noisy and unreliable because they are developed based on pre-training runs conducted with small compute budgets (Wei et al., 2022b).

To address these challenges, we implement a two-stage methodology to develop scaling laws that accurately predict downstream benchmark performance:

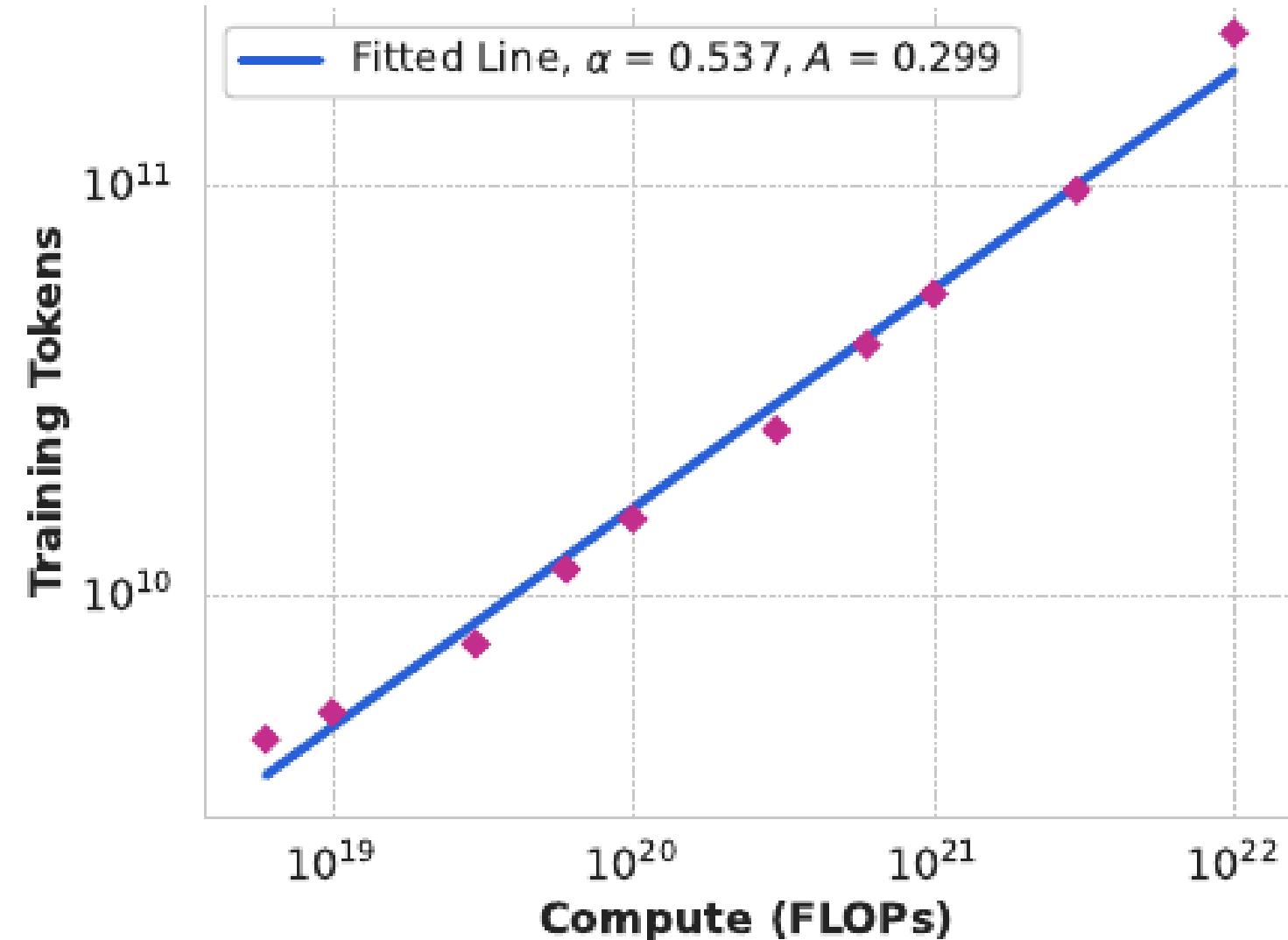
1. We first establish a correlation between the compute-optimal model’s negative log-likelihood on downstream tasks and the training FLOPs.
2. Next, we correlate the negative log-likelihood on downstream tasks with task accuracy, utilizing both the scaling law models and older models trained with higher compute FLOPs. In this step, we specifically leverage the Llama 2 family of models.

This approach enables us to predict downstream task performance given a specific number of training FLOPs for compute-optimal models. We use a similar method to select our pre-training data mix (see Section 3.4).

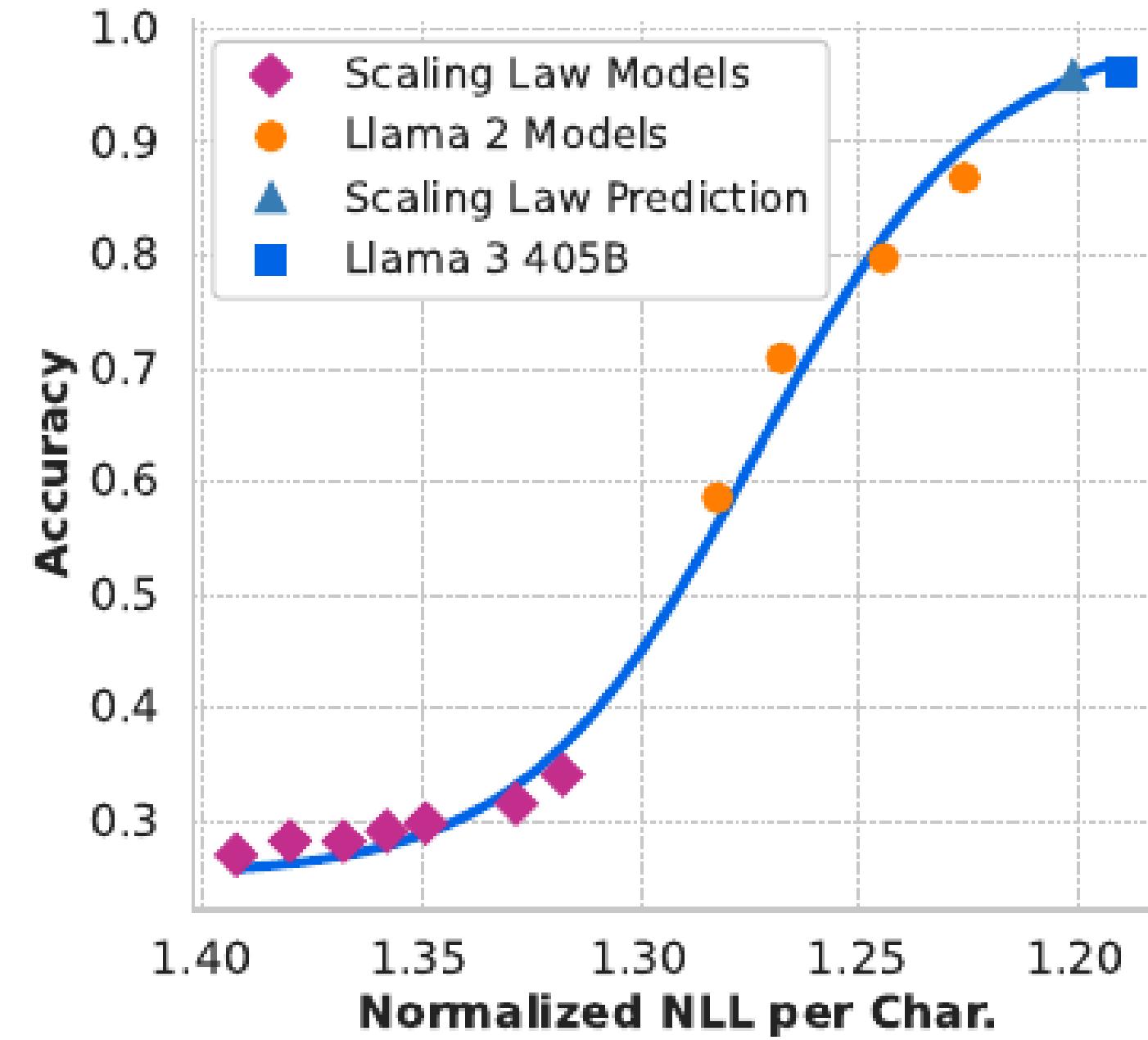
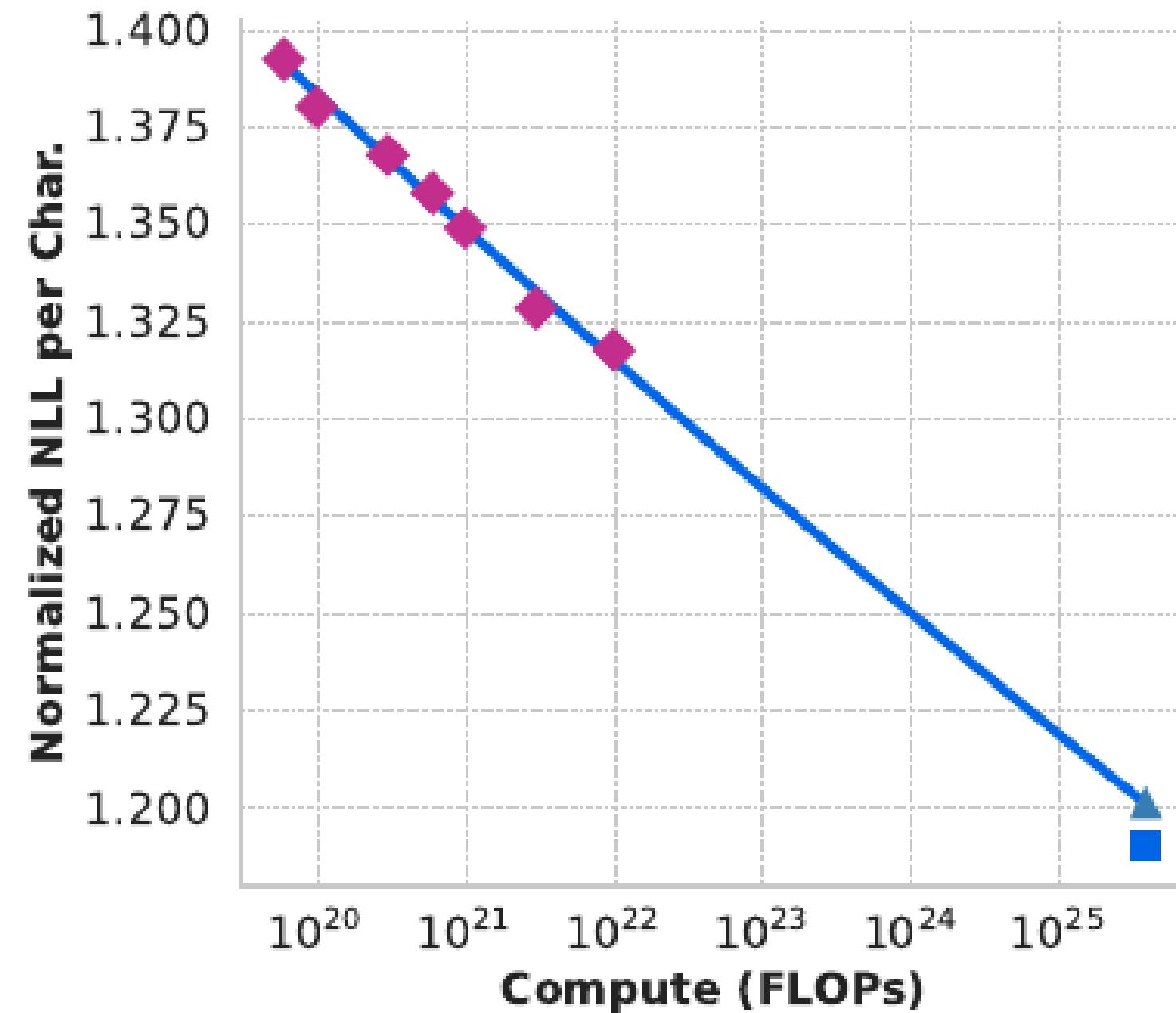
**Scaling law experiments.** Concretely, we construct our scaling laws by pre-training models using compute budgets between  $6 \times 10^{18}$  FLOPs and  $10^{22}$  FLOPs. At each compute budget, we pre-train models ranging in size between 40M and 16B parameters, using a subset of model sizes at each compute budget. In these training runs, we use a cosine learning rate schedule with a linear warmup for 2,000 training steps. The peak learning rate is set between  $2 \times 10^{-4}$  and  $4 \times 10^{-4}$  depending on the size of the model. We set the cosine decay to 0.1 of the peak value. The weight decay at each step is set to 0.1 times the learning rate at that step. We use a fixed batch size for each compute scale, ranging between 250K and 4M.

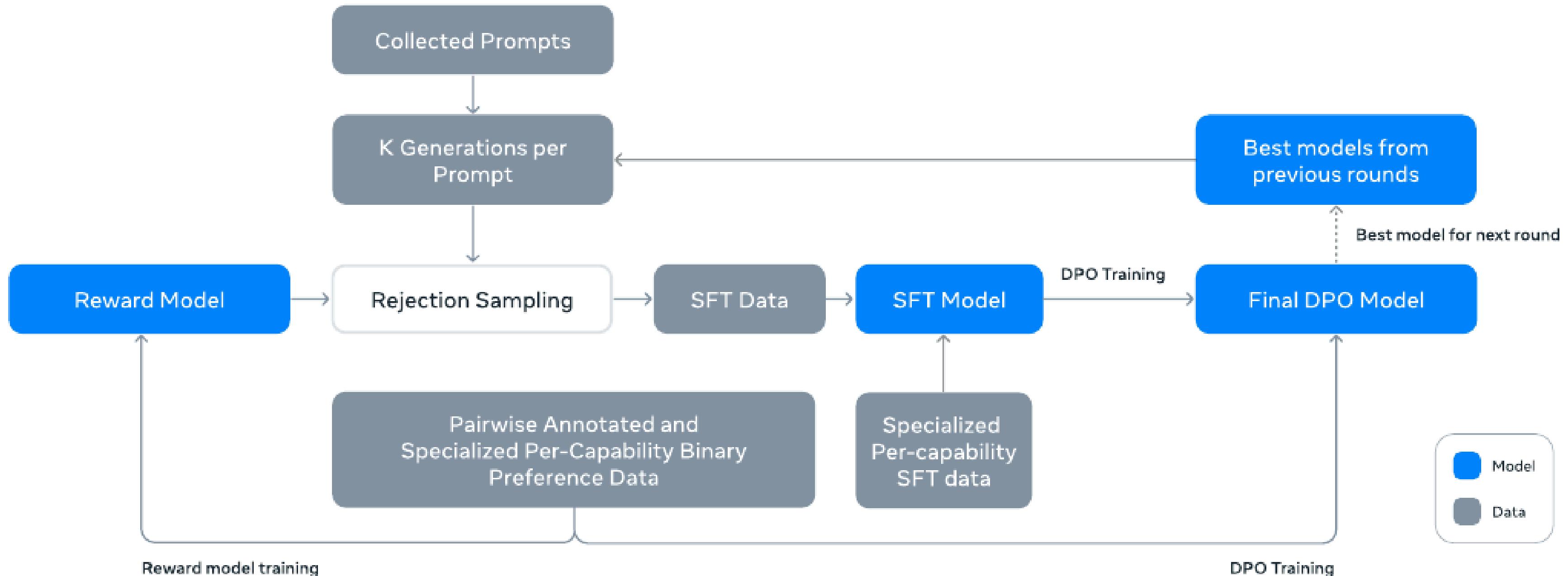


**Figure 2 Scaling law IsoFLOPs curves** between  $6 \times 10^{18}$  and  $10^{22}$  FLOPs. The loss is the negative log-likelihood on a held-out validation set. We approximate measurements at each compute scale using a second degree polynomial.

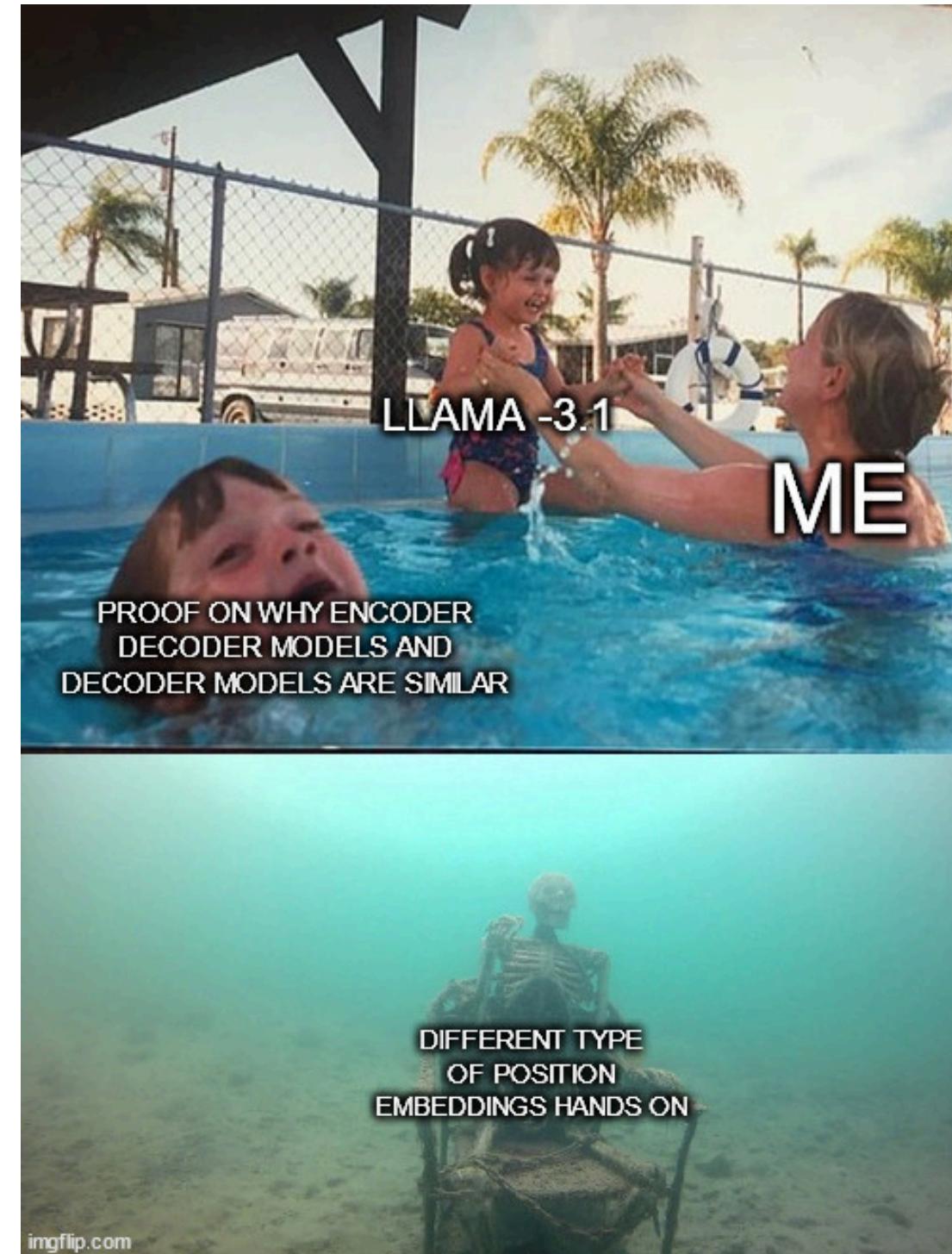


**Figure 3 Number of training tokens in identified compute-optimal models as a function of pre-training compute budget.** We include the fitted scaling-law prediction as well. The compute-optimal models correspond to the parabola minimums in Figure 2.





**So we barely scratched the surface on training let me  
see you in next week - I guess :)**



## **From these session Hopefully answers these question from our POV**

- llama is another Decoder only Model (ever wondered why companies use decoder only models but they exist encoder and decoder based models ?)
- What did they do new to talk about them except for the fact of being open source
- What are interesting finds on this Models except data collecting, data mining (more on collection of data on next session) ?
- What are the guidelines of training a LLMs in this era efficiently and parallelly

# The Llama 3.1 Herd of Models



# **Bit Into scaling Laws**

# Question With given tokens and compute how large can we scale the models (in parameter) and each inverses too ?

Use cases of this

- I can have some kind of theoretical Limit on what to do with amount of compute I have and tokens I have

$$N_{opt}(C) = G \left( \frac{C}{6} \right)^a, \quad D_{opt}(C) = G^{-1} \left( \frac{C}{6} \right)^b, \quad \text{where} \quad G = \left( \frac{\alpha A}{\beta B} \right)^{\frac{1}{\alpha+\beta}}, \quad a = \frac{\beta}{\alpha + \beta}, \quad \text{and} \quad b = \frac{\alpha}{\alpha + \beta}. \quad (4)$$

Approach	Coeff. $a$ where $N_{opt} \propto C^a$	Coeff. $b$ where $D_{opt} \propto C^b$
1. Minimum over training curves	0.50 (0.488, 0.502)	0.50 (0.501, 0.512)
2. IsoFLOP profiles	0.49 (0.462, 0.534)	0.51 (0.483, 0.529)
3. Parametric modelling of the loss	0.46 (0.454, 0.455)	0.54 (0.542, 0.543)
Kaplan et al. (2020)	0.73	0.27

Parameters	FLOPs	FLOPs (in <i>Gopher</i> unit)	Tokens
400 Million	1.92e+19	1/29,968	8.0 Billion
1 Billion	1.21e+20	1/4,761	20.2 Billion
10 Billion	1.23e+22	1/46	205.1 Billion
67 Billion	5.76e+23	1	1.5 Trillion
175 Billion	3.85e+24	6.7	3.7 Trillion
280 Billion	9.90e+24	17.2	5.9 Trillion
520 Billion	3.43e+25	59.5	11.0 Trillion
1 Trillion	1.27e+26	221.3	21.2 Trillion
10 Trillion	1.30e+28	22515.9	216.2 Trillion

# Now on to dataset collection and Training

## 3.1.2 Determining the Data Mix

To obtain a high-quality language model, it is essential to carefully determine the proportion of different data sources in the pre-training data mix. Our main tools in determining this data mix are knowledge classification and scaling law experiments.

**Knowledge classification.** We develop a classifier to categorize the types of information contained in our web data to more effectively determine a data mix. We use this classifier to downsample data categories that are over-represented on the web, for example, arts and entertainment.

**Scaling laws for data mix.** To determine the best data mix, we perform scaling law experiments in which we train several small models on a data mix and use that to predict the performance of a large model on that mix (see Section 3.2.1). We repeat this process multiple times for different data mixes to select a new data mix candidate. Subsequently, we train a larger model on this candidate data mix and evaluate the performance of that model on several key benchmarks.

**Data mix summary.** Our final data mix contains roughly 50% of tokens corresponding to general knowledge, 25% of mathematical and reasoning tokens, 17% code tokens, and 8% multilingual tokens.

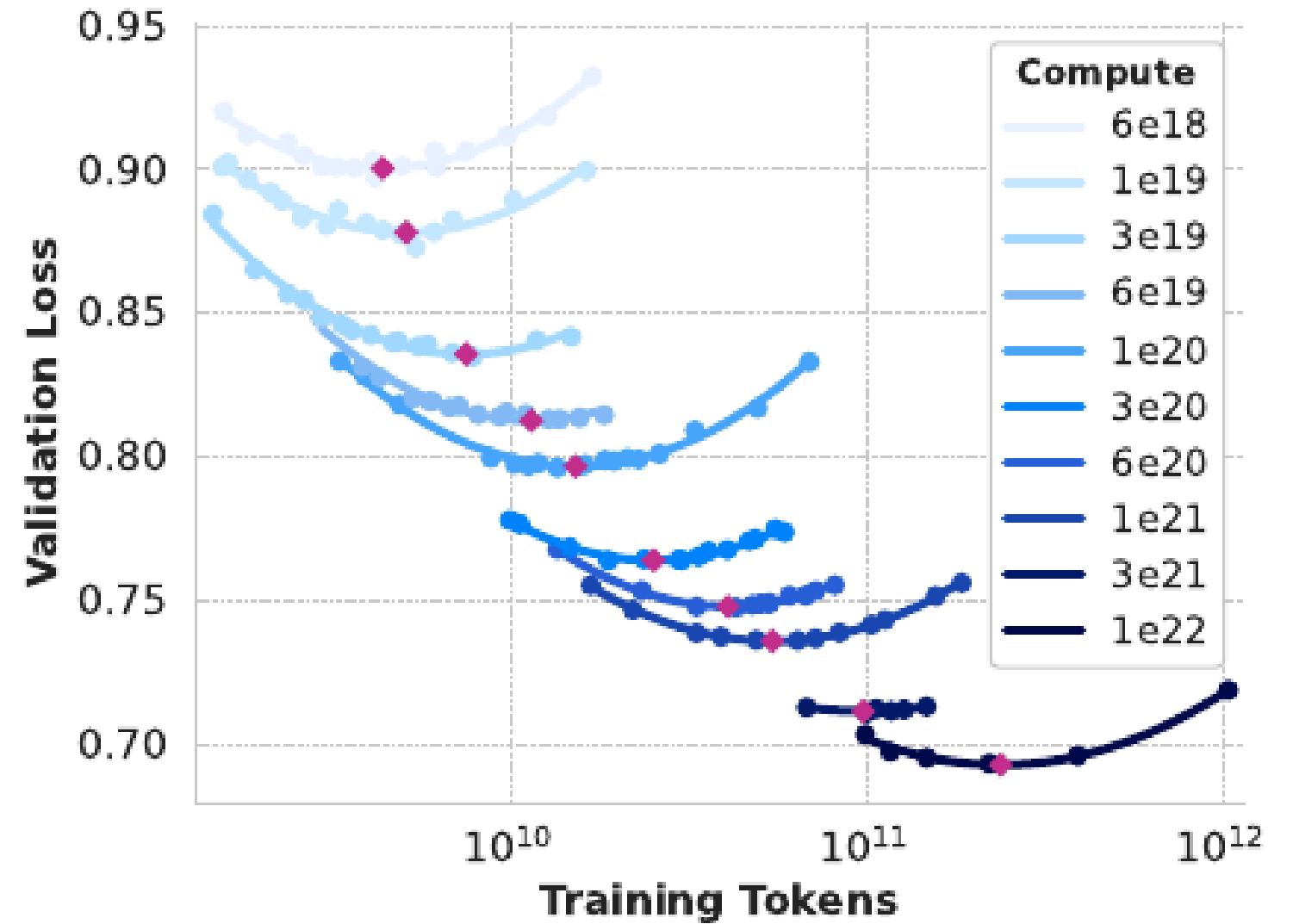
We develop scaling laws (Hoffmann et al., 2022; Kaplan et al., 2020) to determine the optimal model size for our flagship model given our pre-training compute budget. In addition to determining the optimal model size, a major challenge is to forecast the flagship model’s performance on downstream benchmark tasks, due to a couple of issues: (1) Existing scaling laws typically predict only next-token prediction loss rather than specific benchmark performance. (2) Scaling laws can be noisy and unreliable because they are developed based on pre-training runs conducted with small compute budgets (Wei et al., 2022b).

To address these challenges, we implement a two-stage methodology to develop scaling laws that accurately predict downstream benchmark performance:

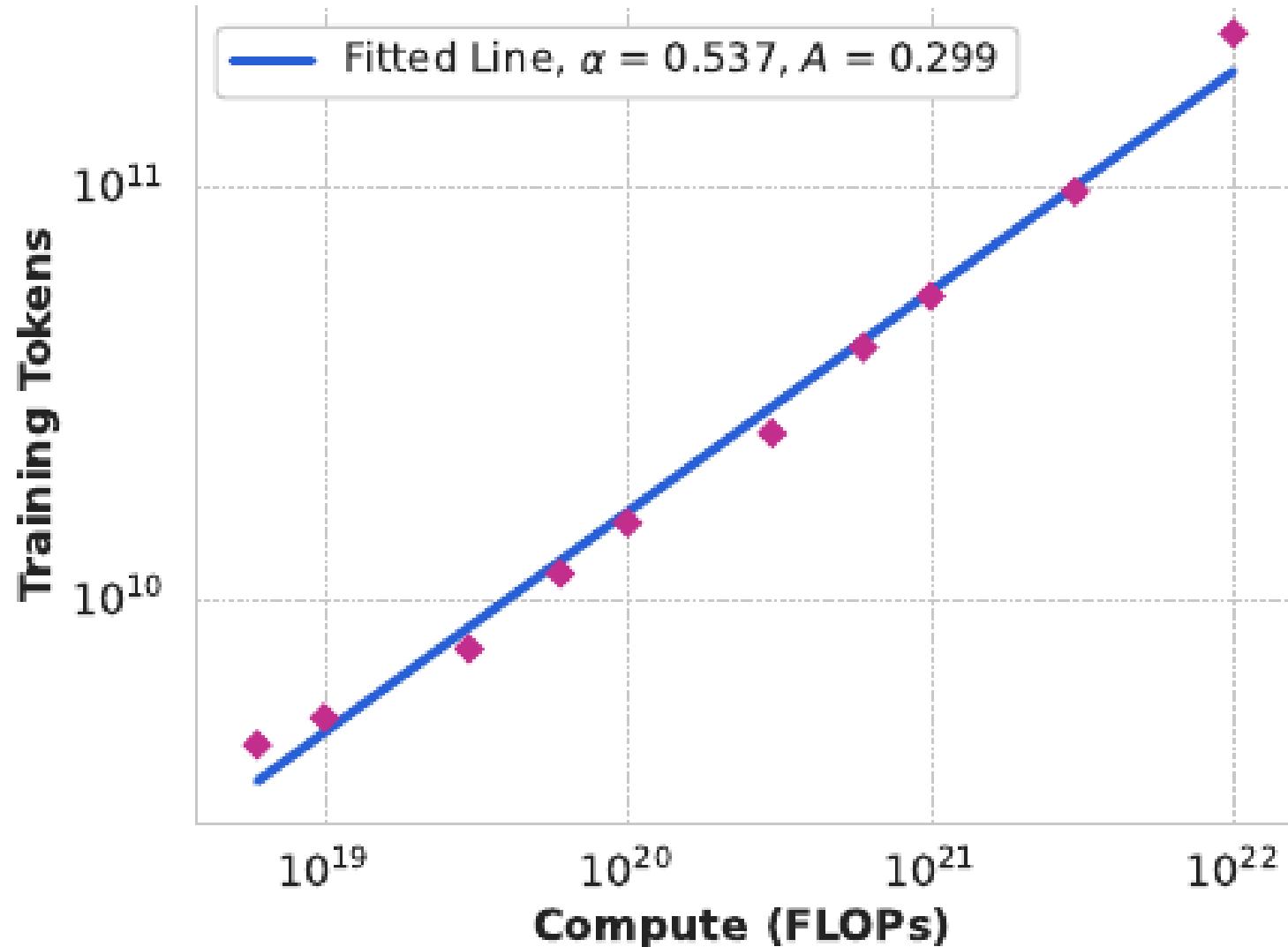
1. We first establish a correlation between the compute-optimal model’s negative log-likelihood on downstream tasks and the training FLOPs.
2. Next, we correlate the negative log-likelihood on downstream tasks with task accuracy, utilizing both the scaling law models and older models trained with higher compute FLOPs. In this step, we specifically leverage the Llama 2 family of models.

This approach enables us to predict downstream task performance given a specific number of training FLOPs for compute-optimal models. We use a similar method to select our pre-training data mix (see Section 3.4).

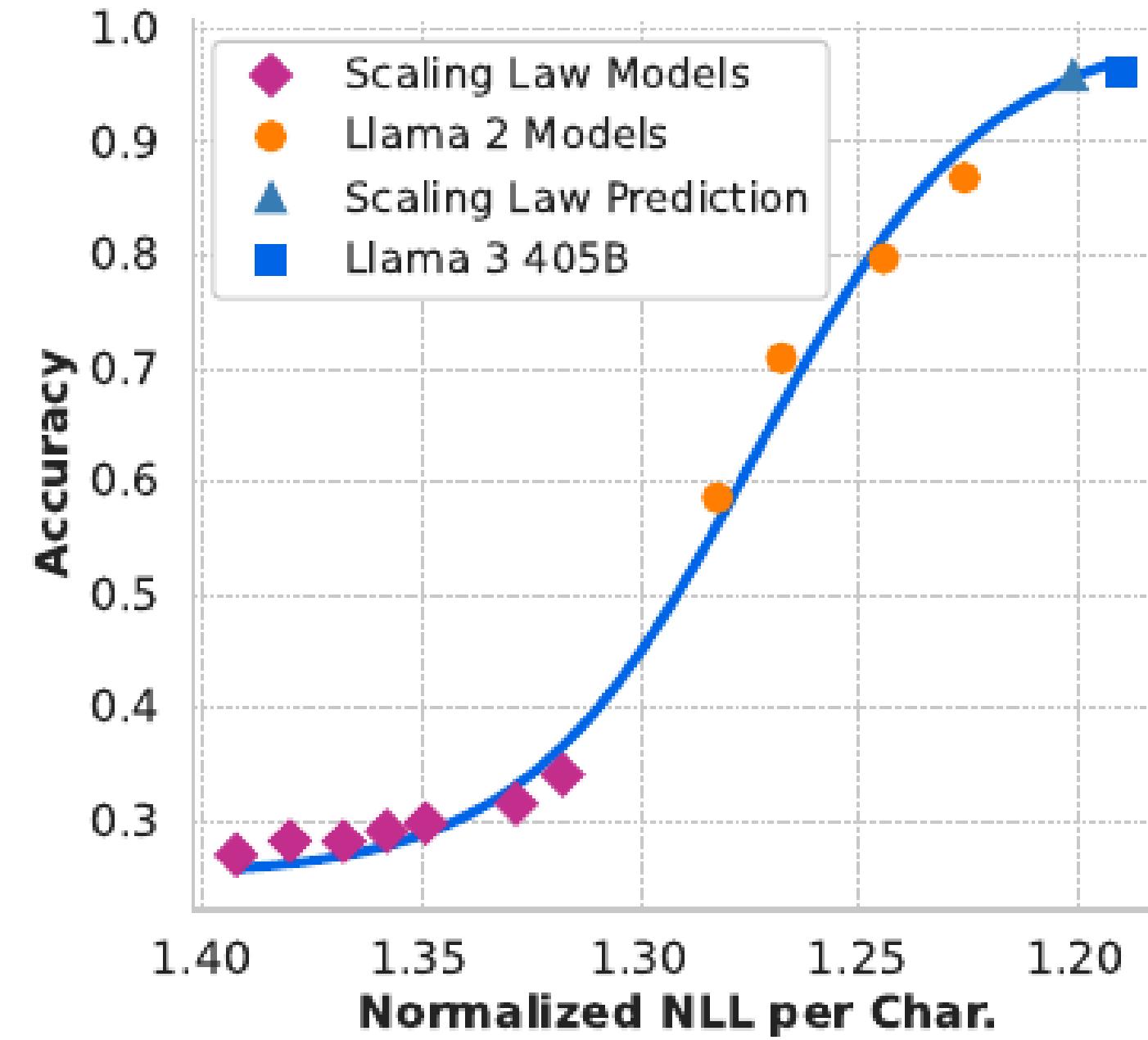
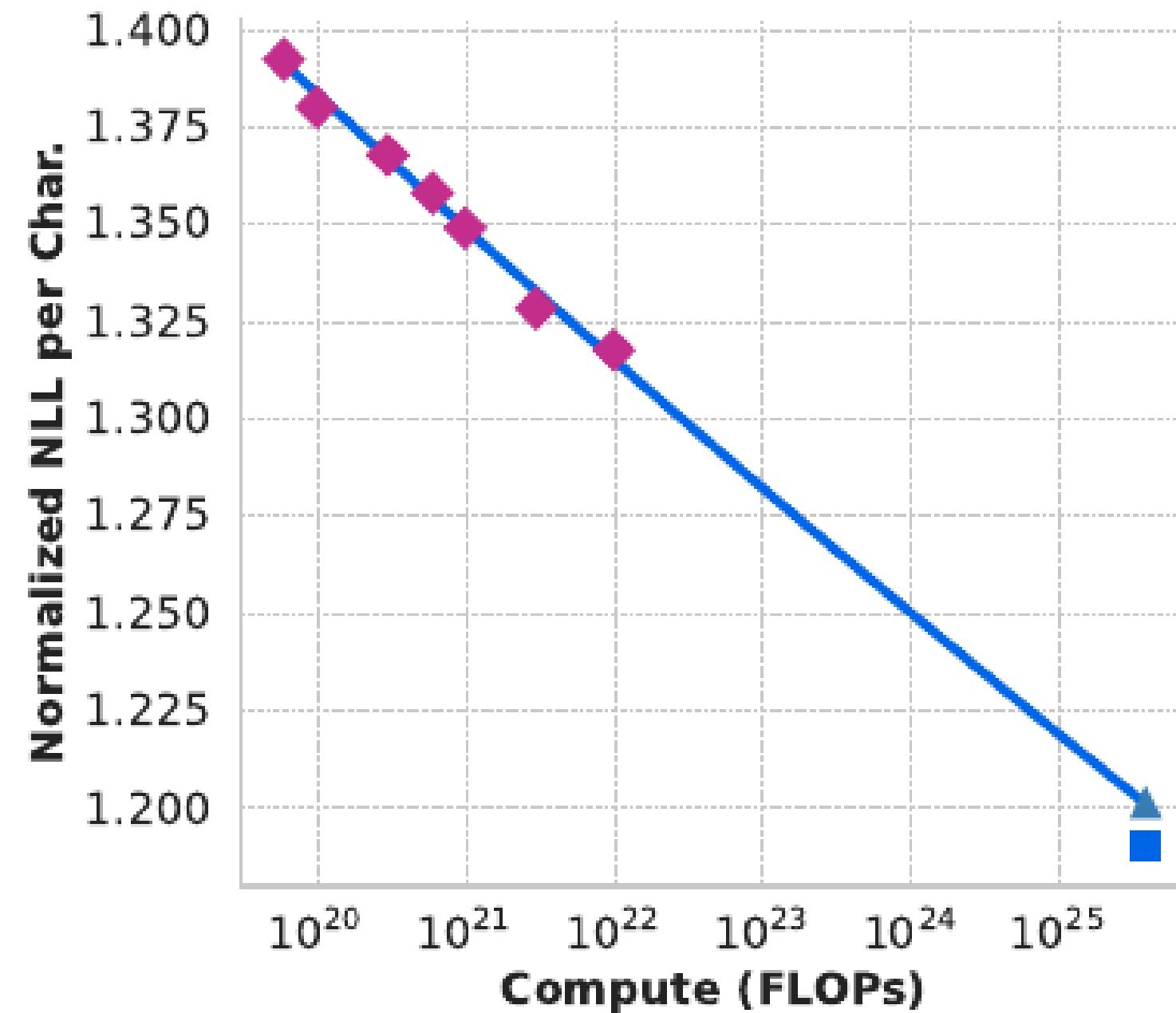
**Scaling law experiments.** Concretely, we construct our scaling laws by pre-training models using compute budgets between  $6 \times 10^{18}$  FLOPs and  $10^{22}$  FLOPs. At each compute budget, we pre-train models ranging in size between 40M and 16B parameters, using a subset of model sizes at each compute budget. In these training runs, we use a cosine learning rate schedule with a linear warmup for 2,000 training steps. The peak learning rate is set between  $2 \times 10^{-4}$  and  $4 \times 10^{-4}$  depending on the size of the model. We set the cosine decay to 0.1 of the peak value. The weight decay at each step is set to 0.1 times the learning rate at that step. We use a fixed batch size for each compute scale, ranging between 250K and 4M.



**Figure 2 Scaling law IsoFLOPs curves** between  $6 \times 10^{18}$  and  $10^{22}$  FLOPs. The loss is the negative log-likelihood on a held-out validation set. We approximate measurements at each compute scale using a second degree polynomial.



**Figure 3 Number of training tokens in identified compute-optimal models as a function of pre-training compute budget.** We include the fitted scaling-law prediction as well. The compute-optimal models correspond to the parabola minimums in Figure 2.



# Pre-Training Data

- PII (Personally Identifiable Information ) and Safety filtering
- Text extraction and cleaning
- De-Duplication
  - URL level
  - Document level (Minhash)
  - Line level
- Heuristic Filtering

# **Heuristic filtering**

- Using N gram coverage Ratio for Line level de duplication
- Dirty word counting
- KL divergence to filter our document containing excess amount of outlier words

## **Multilingual data**

- Uses Fasttext based language identification 176 language
- Same as above document-level, line-level de duplication

## **Final Data mix consist of**

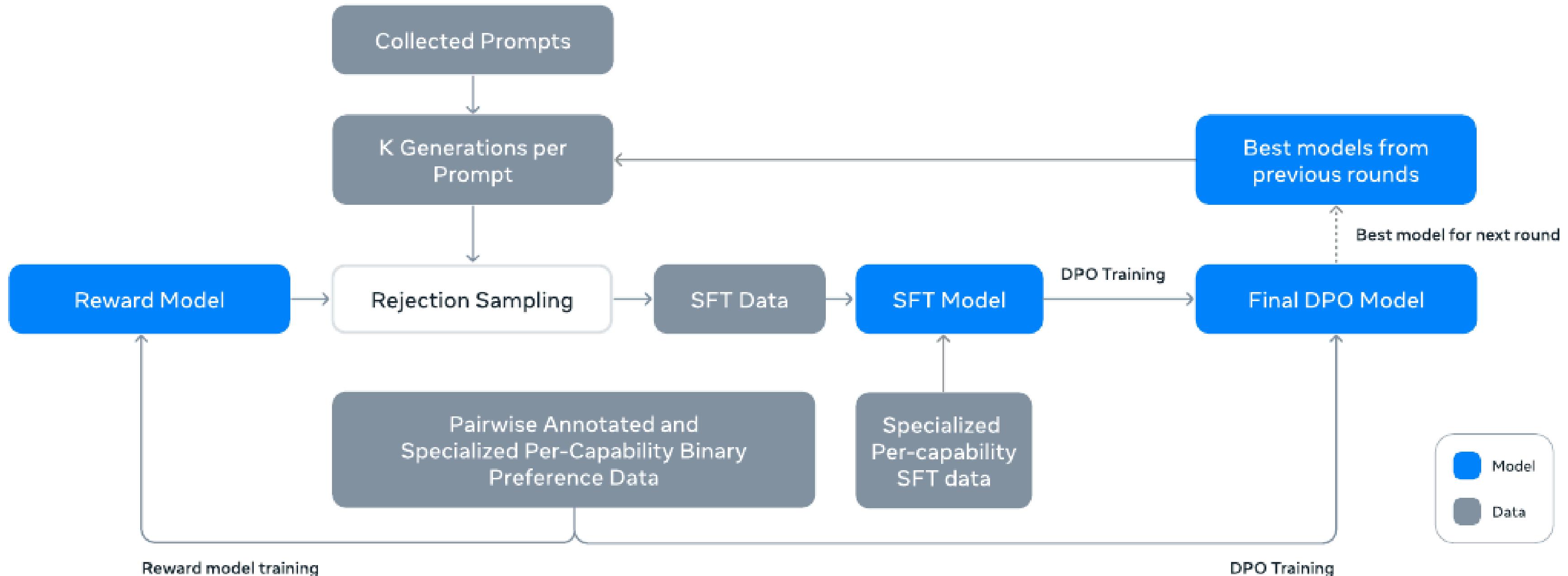
- 50% of tokens corresponding to General Knowledge
- 25% of mathematical and reasoning tokens
- 17% code tokens
- 8% multilingual tokens

### 3.1.3 Annealing Data

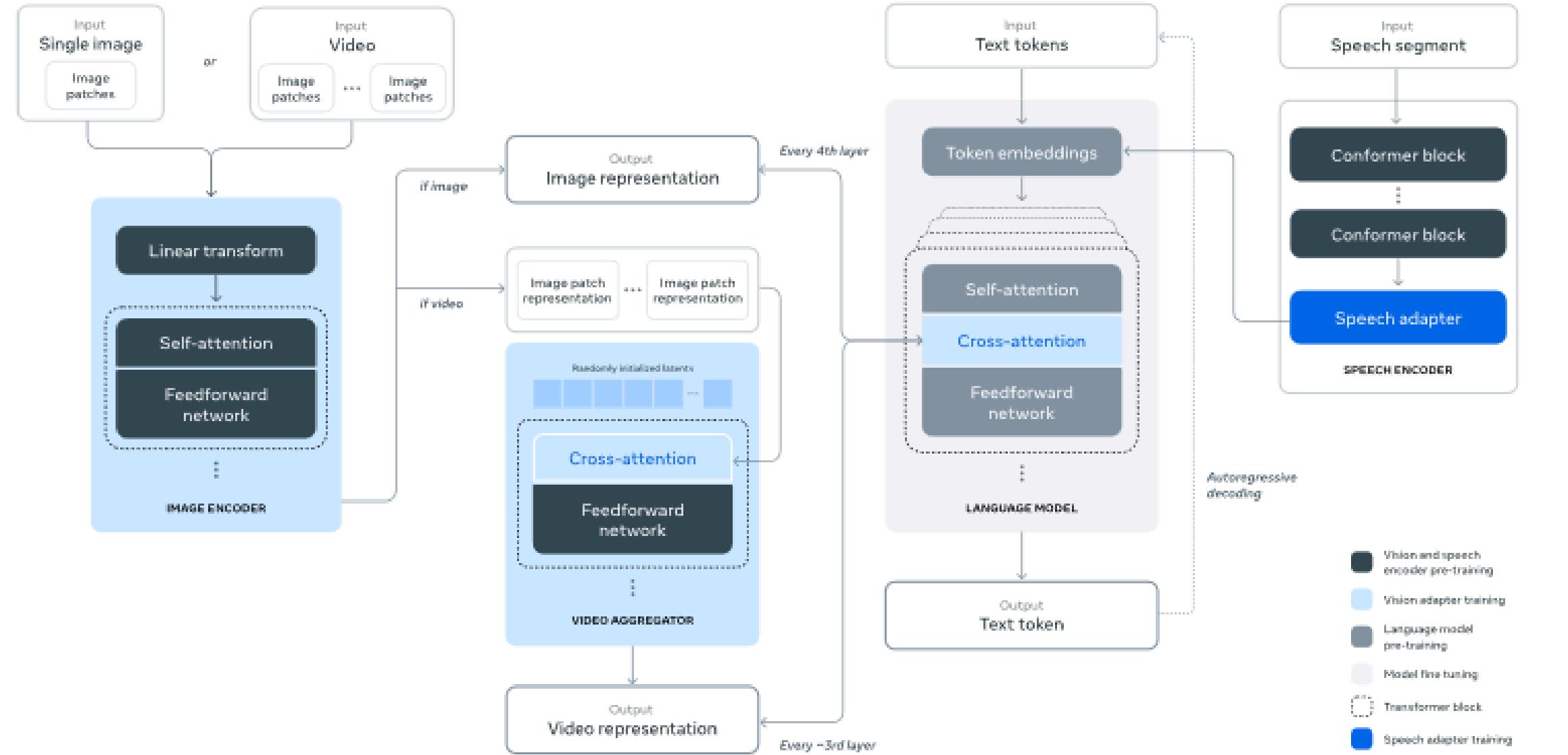
Empirically, we find that annealing (see Section 3.4.3) on small amounts of high-quality code and mathematical data can boost the performance of pre-trained models on key benchmarks. Akin to Li et al. (2024b), we perform annealing with a data mix that upsamples high-quality data in select domains. We do not include any training sets from commonly used benchmarks in our annealing data. This enables us to assess the true few-shot learning capabilities and out-of-domain generalization of Llama 3.

Following OpenAI (2023a), we evaluate the efficacy of annealing on the GSM8k (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021b) training sets in annealing. We find that annealing improved the performance of a pre-trained Llama 3 8B model on the GSM8k and MATH validation sets by 24.0% and 6.4%, respectively. However, the improvements on the 405B model are negligible, suggesting that our flagship model has strong in-context learning and reasoning capabilities and does not require specific in-domain training samples to obtain strong performance.

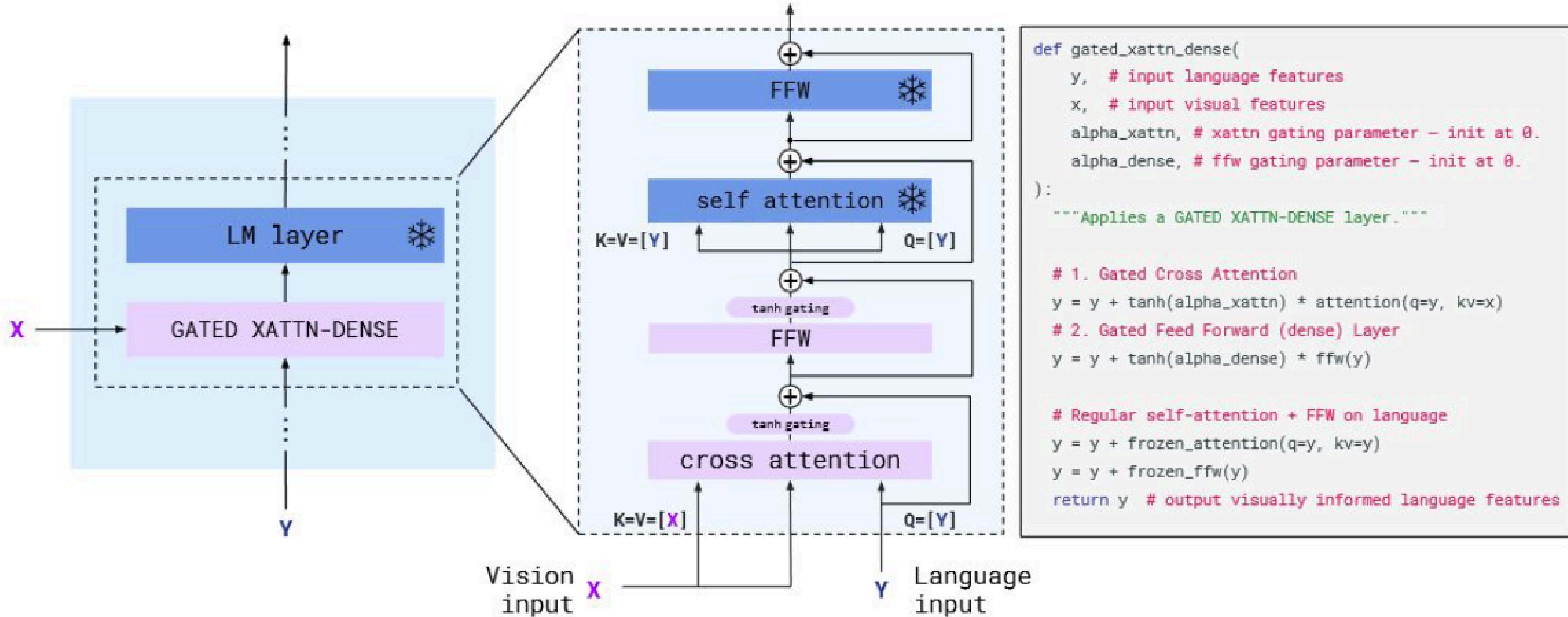
**Using annealing to assess data quality.** Similar to Blakeney et al. (2024), we find that annealing enables us to judge the value of small domain-specific datasets. We measure the value of such datasets by annealing the learning rate of a 50% trained Llama 3 8B model linearly to 0 on 40B tokens. In those experiments, we assign 30% weight to the new dataset and the remaining 70% weight to the default data mix. Using annealing to evaluate new data sources is more efficient than performing scaling law experiments for every small dataset.



# **Multi Modality**



# Not exactly similar to Flamingo



<b>Dataset</b>	<b>% of comparisons</b>	<b>Avg. # turns per dialog</b>	<b>Avg. # tokens per example</b>	<b>Avg. # tokens in prompt</b>	<b>Avg. # tokens in response</b>
General English	81.99%	4.1	1,000.4	36.4	271.2
Coding	6.93%	3.2	1,621.0	113.8	462.9
Multilingual	5.19%	1.8	1,299.4	77.1	420.9
Reasoning and tools	5.89%	1.6	707.7	46.6	129.9
Total	100%	3.8	1,041.6	44.5	284.0

**What happened After this ?**

https://developer.nvidia.com/blog/how-to-prune-and-distill-llama-3-1-8b-to-an-nvidia-llama-3-1-minitron-4b-model/

NVIDIA DEVELOPER Home Blog Forums Docs Downloads Training

Join

Technical Blog Search blog Filter Subscribe >

Generative AI

English ▾

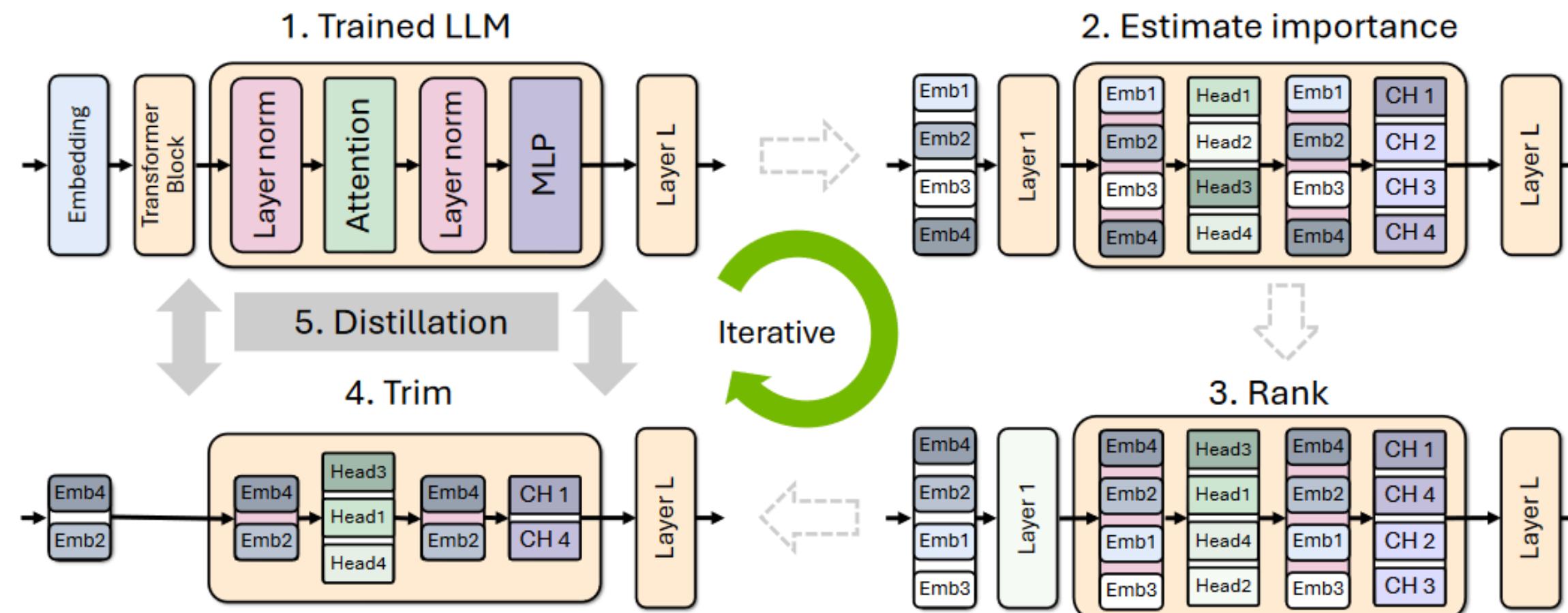
# How to Prune and Distill Llama-3.1 8B to an NVIDIA Llama-3.1-Minitron 4B Model

Aug 14, 2024

+32 Like Discuss (5)

By Sharath Sreenivas, Vinh Nguyen, Saurav Muralidharan, Marcin Chochowski and Raviraj Joshi

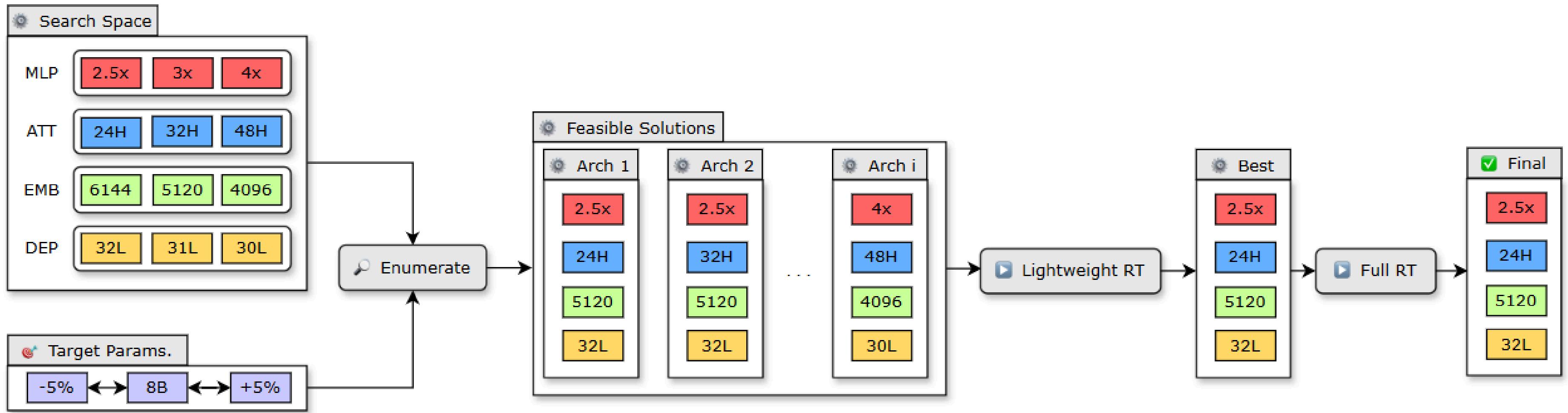


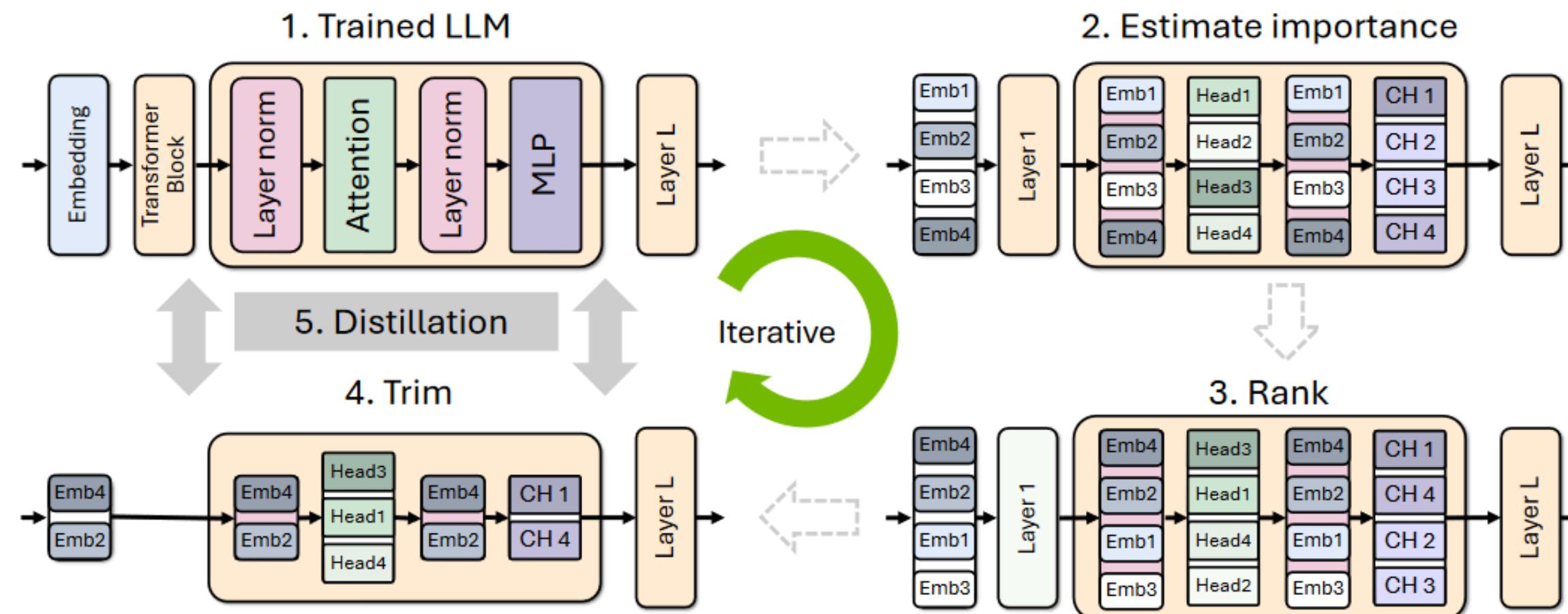


# Estimation of Importance

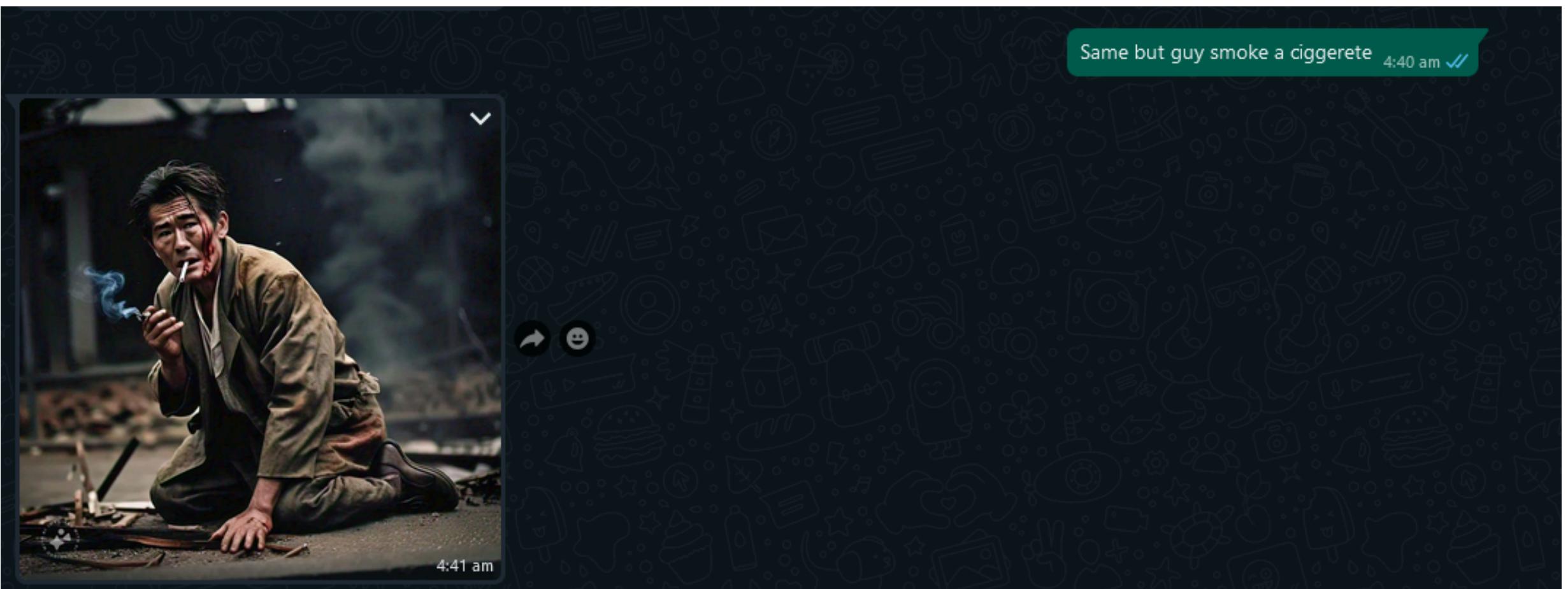
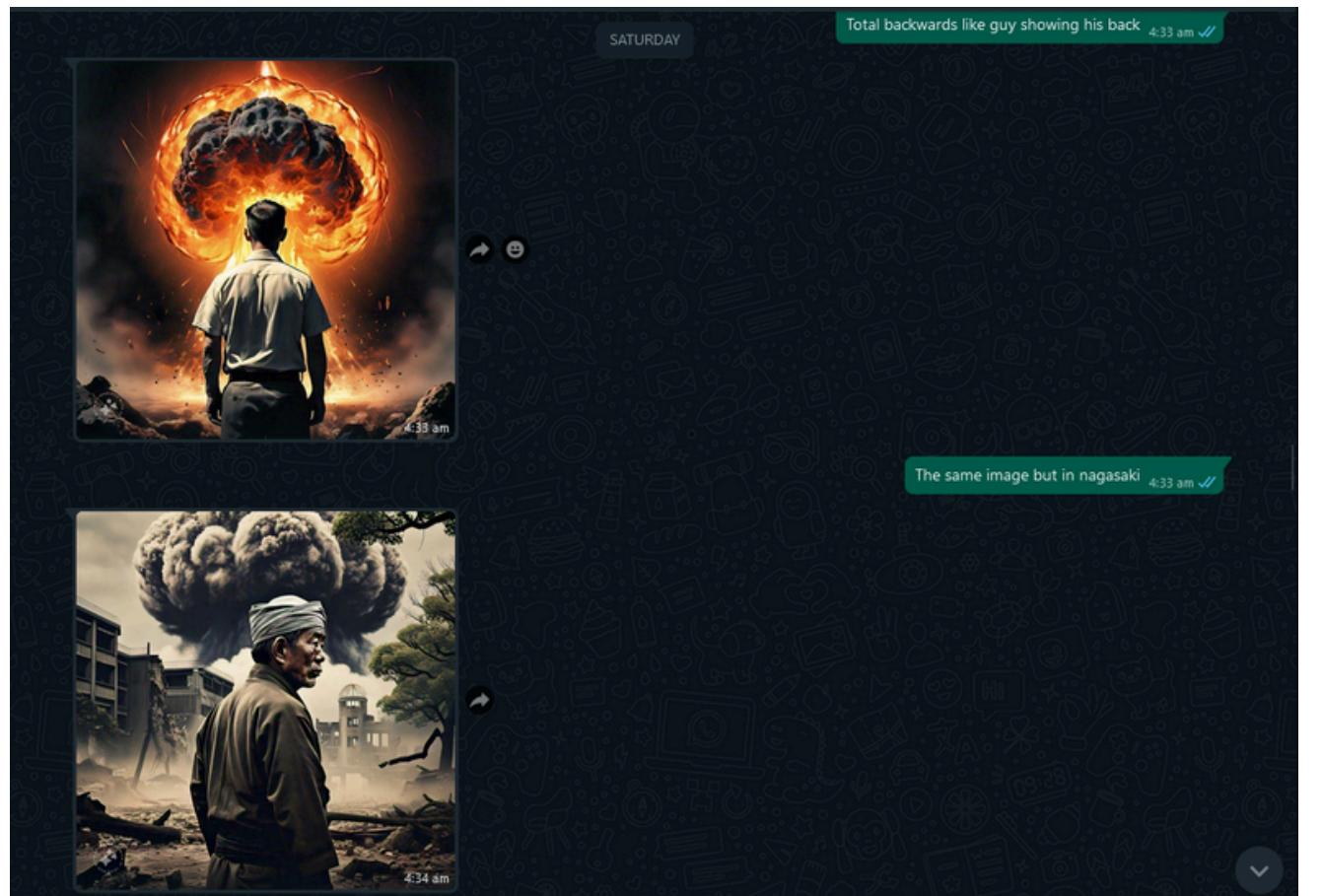
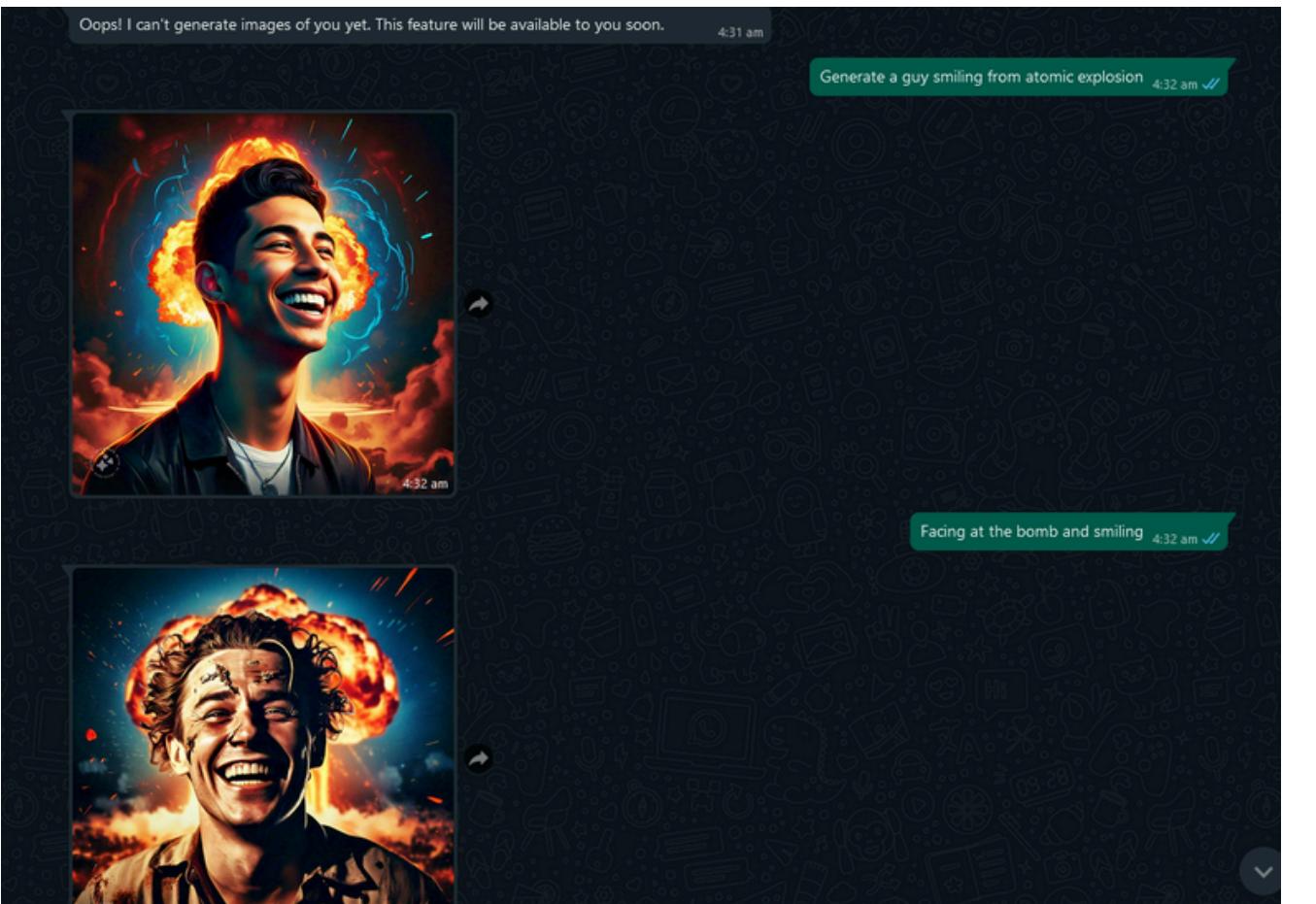
- estimate the important with the gradients graph of the network and prune the network
- But computing the gradient of the network and pruning the network for a sample size  $S$  is costly is there is any way to do this cheaply ?
- We could do this estimation of importance by taking
  - Mean between Data vs weights
  - Variance between Data vs weights
  - L2 Norm Data vs weights

# How Pruned models are Constructed





**Some Generation from the  
llama 3.1**



tell me about ¥\$ 7:05 pm ✓

¥\$ is a symbol that combines the Japanese yen (¥) and the US dollar (\$) signs. However, it's not an official currency symbol. Here are a few possible interpretations:

1. Exchange rates: ¥\$ might be used to represent the exchange rate between the Japanese yen and the US dollar.
2. Currency trading: It could be a notation used in currency trading or forex markets to denote the yen-dollar pair.
3. Creative expression: ¥\$ might be used in art, design, or marketing to create a visually striking effect, combining two recognizable currency symbols.

If you have a specific context or usage in mind, I'd be happy to help you understand it better!

7:05 pm

Does this have any band? 7:06 pm ✓

¥\$ doesn't seem to be a widely recognized symbol for a specific band. However, it's possible that a band or artist might use ¥\$ as a logo, moniker, or branding element. If you're thinking of a particular band, please provide more context or details, and I can try to help you identify them.

That being said, there are bands with names like "Yen," "Dollar Signs," or "Currency," but I couldn't find any well-known bands specifically using the ¥\$ symbol as their name or logo. If you have more information or clarification, I'd be happy to try and assist you further!

7:06 pm

What about ye and tydolla sign group ? 7:06 pm ✓

TY Dolla Sign is an American singer, songwriter, and record producer, and "Ye" could be a reference to Kanye West, who has also been known to go by the mononym "Ye".

TY Dolla Sign has collaborated with Kanye West on several occasions, and they have a good creative relationship. However, I couldn't find any evidence of a specific group or project called "¥\$ Ye" or "TY Dolla Sign ¥\$" that combines the two artists.

If you're thinking of a particular song, album, or project that involves both TY Dolla Sign and Kanye West, I'd be happy to try and help you identify it!

7:06 pm

What are those albums ? 7:06 pm ✓

YESTERDAY

7:05 pm

Does this have any band? 7:06 pm ✓

¥\$ doesn't seem to be a widely recognized symbol for a specific band. However, it's possible that a band or artist might use ¥\$ as a logo, moniker, or branding element. If you're thinking of a particular band, please provide more context or details, and I can try to help you identify them.

That being said, there are bands with names like "¥en," "Dollar Signs," or "Currency," but I couldn't find any well-known bands specifically using the ¥\$ symbol as their name or logo. If you have more information or clarification, I'd be happy to try and assist you further!

7:06 pm

What about ye and tydolla sign group ? 7:06 pm ✓

TY Dolla Sign is an American singer, songwriter, and record producer, and "Ye" could be a reference to Kanye West, who has also been known to go by the mononym "Ye".

TY Dolla Sign has collaborated with Kanye West on several occasions, and they have a good creative relationship. However, I couldn't find any evidence of a specific group or project called "¥\$ Ye" or "TY Dolla Sign ¥\$" that combines the two artists.

If you're thinking of a particular song, album, or project that involves both TY Dolla Sign and Kanye West, I'd be happy to try and help you identify it!

7:06 pm

What are those albums ? 7:06 pm ✓

TY Dolla Sign and Kanye West have collaborated on several projects. Here are a few notable ones:

1. "Ego Death" (2015) - A single by TY Dolla Sign, featuring Kanye West, FKA twigs, and Skrillex.
2. "Real Friends" (2016) - A song by Kanye West, featuring TY Dolla Sign and Kendrick Lamar, from West's album "The Life of Pablo".
3. "Fade" (2016) - A song by Kanye West, featuring TY Dolla Sign and Post Malone, from West's album "The Life of Pablo".
4. "Ye" (2018) - Kanye West's album, which features TY Dolla Sign as a co-writer and background vocalist on several tracks.
5. "Jesus Is King" (2019) - Kanye West's album, which features TY Dolla Sign as a co-writer and vocalist on the song "Everything We Need".

TY Dolla Sign has also been involved in other projects with Kanye West, such as the "Cruel Winter"

track EP that features the two artists blending their styles, with TY Dolla Sign's soulful vocals and Kanye West's innovative production.

The "Vultures" EP includes tracks like "Vultures", "New Body", "By Yourself", "No Sense", and "War". The project received positive reviews from fans and critics, highlighting the chemistry and creativity between TY Dolla Sign and Kanye West.

Is there a specific aspect of the "Vultures" project you'd like to know more about?

7:07 pm

Kanyewest and tydolla sign in a photo with playboy carti in donda era

7:08 pm ✓

Oops! I can't generate that image. Can I help you imagine something else?

7:08 pm ✓



Kanye west and ty dolla sign in a image

7:08 pm ✓

Oops! I can't generate that image. Can I help you imagine something else?

7:08 pm

Why

7:08 pm ✓

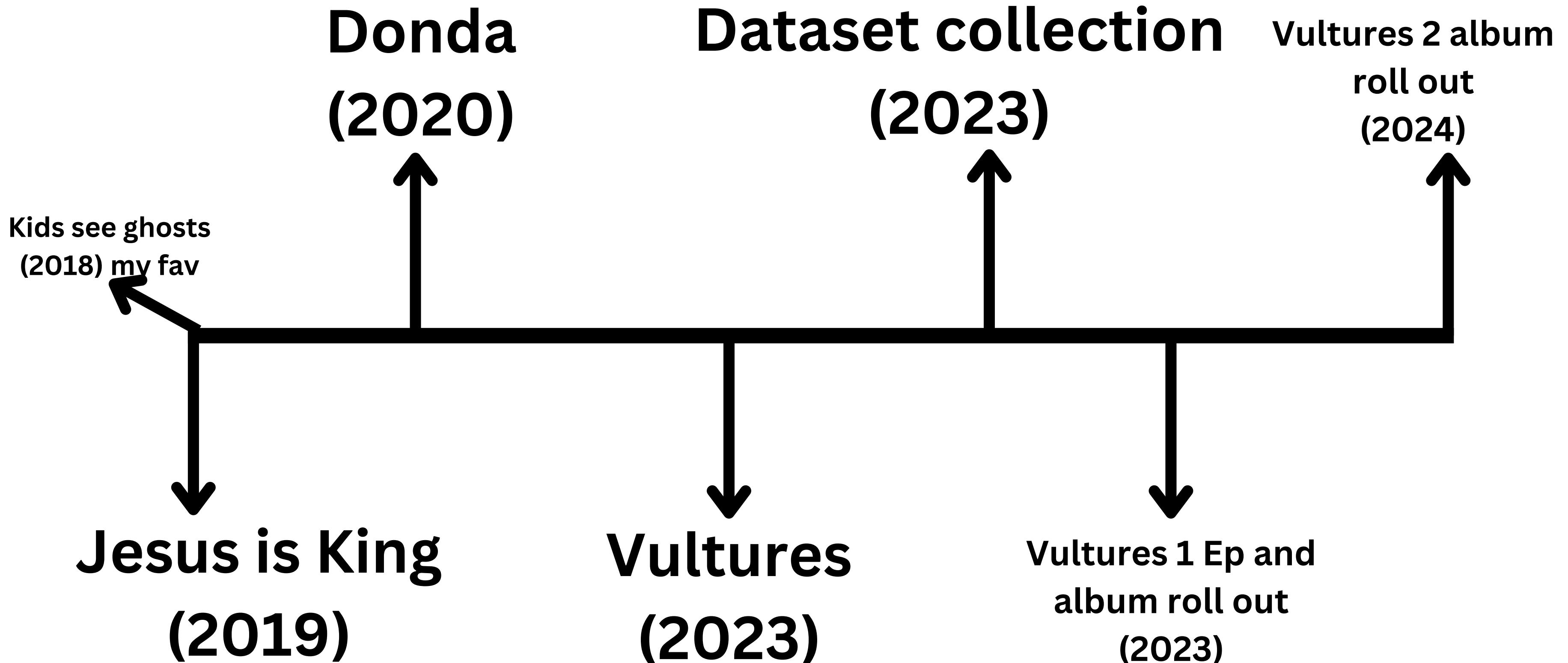
¡Hola! The reason behind the name "Vultures" is not explicitly stated by TY Dolla Sign or Kanye West. However, based on the lyrics and themes presented in the EP, here are some possible interpretations:

- Critique of societal vultures: The title might refer to people who prey on others' struggles or success, much like vultures feed on carrion.
- Protection from negativity: The artists may be symbolically warding off "vultures" that represent negative influences or toxic relationships.
- Resilience and survival: Vultures are known for their ability to thrive in harsh environments, which could represent the artists' capacity to overcome adversity.
- Commentary on the music industry: "Vultures" might represent the cutthroat nature of the industry, where artists are often seen as prey.

Keep in mind that these are just interpretations, and the true meaning behind the title "Vultures" remains open to individual understanding.

7:08 pm

# Estimation of dataset collection time from this response (from my side)



## **From these session Hopefully answers these question from our POV**

- llama is another Decoder only Model (ever wondered why companies use decoder only models but they exist encoder and decoder based models ?)
- What did they do new to talk about them except for the fact of being open source
- What are interesting finds on this Models except data collecting, data mining (more on collection of data on next session) ?
- What are the guidelines of training a LLMs in this era efficiently and parallelly

## Upcoming Hands on Lecture

- Nano GPT focused on inferences (KV caching, quantization ) etc
- IFT from scratch on LLMs and how to do it
- DPO from scratch from preference data

**I also stream btw where I code for these sessions**

KL Divergence is a Metric to Evaluate Similarity  
Between distribution  $P(x) \& Q(x)$

If  $D_{KL}(P(x) || Q(x)) = 0$  then  $P(x) = Q(x)$

$D_{KL}(P(x) || Q(x)) \neq D_{KL}(Q(x) || P(x))$

$$D_{KL}(P(x) || Q(x)) = \sum P(x) \log \left( \frac{P(x)}{Q(x)} \right) \text{- discrete}$$

or

$$= \int P(x) \log \left( \frac{P(x)}{Q(x)} \right) dx$$

$\Sigma$

continuous

Now am just using Continuous because of generalization

$$= \int P(x) \log \left( \frac{P(x)}{Q(x)} \right) dx \Rightarrow \int P(x) \log(P(x)) dx - ①$$

$$- \int P(x) \log(Q(x)) dx - ③$$

②

Equation ③ is ① - ②

① → 0 then Eq ③ is Cross Entropy loss

$$③ \text{ is } -\sum p(x) \log q(x)$$

So in Decoder Models / Encoder-Decoder models we  
are just fine with CE Loss

**Thank you Any doubts ?**

**Feel free to ping me on discord @maddy**

**or**

**Linkedin - Madhava Prasath**