LLM LE4 VT2026

Pre-training and Scaling

Fredrik Heintz
Dept. of Computer Science
Linköping University
fredrik.heintz@liu.se
@FredrikHeintz

Outline:

- Pre-training
- Parallel Training
- Scaling

LINKÖPING UNIVERSITY

Language modelling

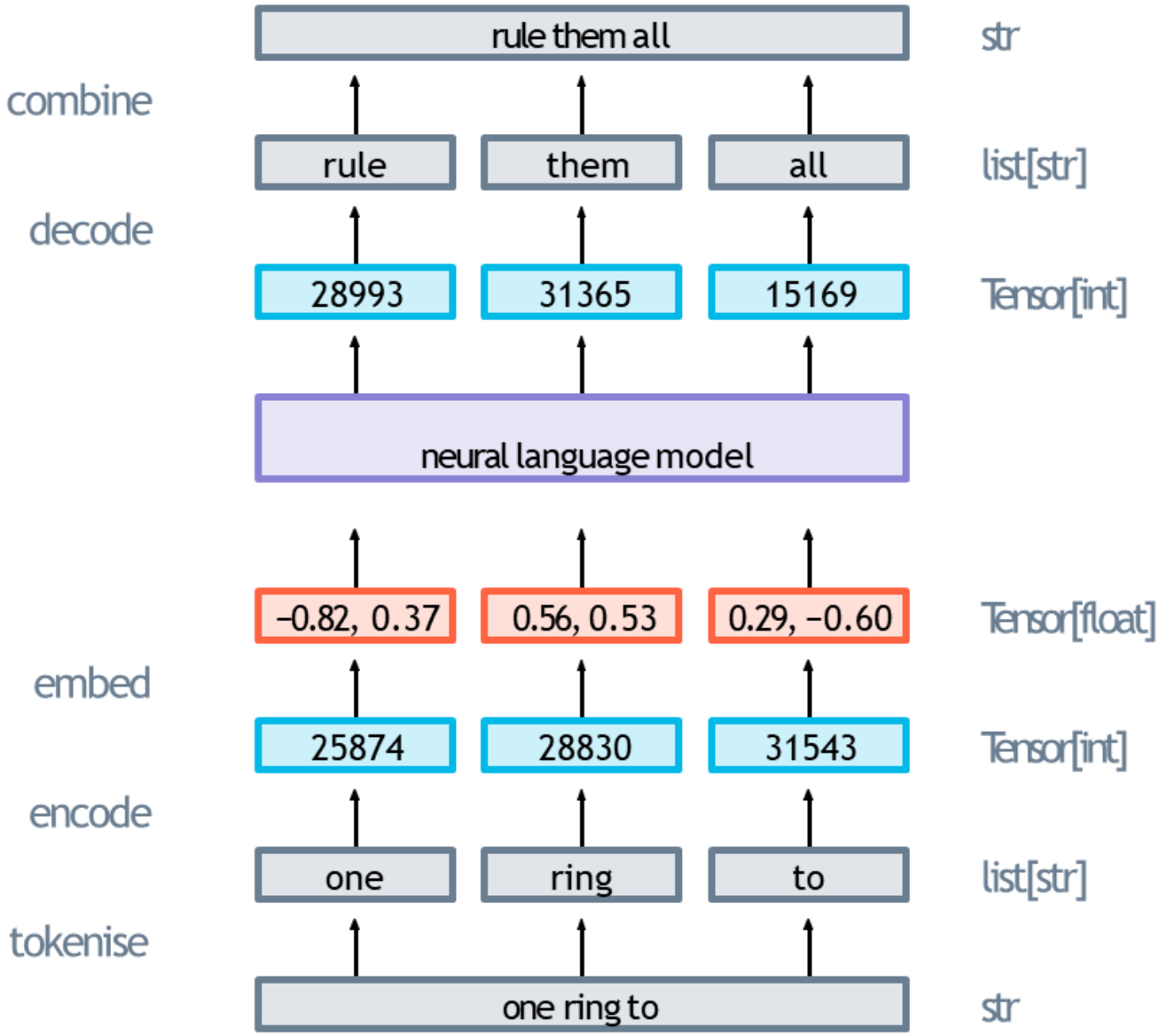
- **Language modelling** is the task of predicting which word comes next in a sequence of words.
- More formally, given a sequence of words $w_1, \ldots, w_t$ we want to know the probability of the next word, w_{t+1}:

$$P(w_{t+1} \mid w_1, \ldots, w_t)$$

- We are assuming that w_{t+1} comes from a finite vocabulary V.

language models = classifiers

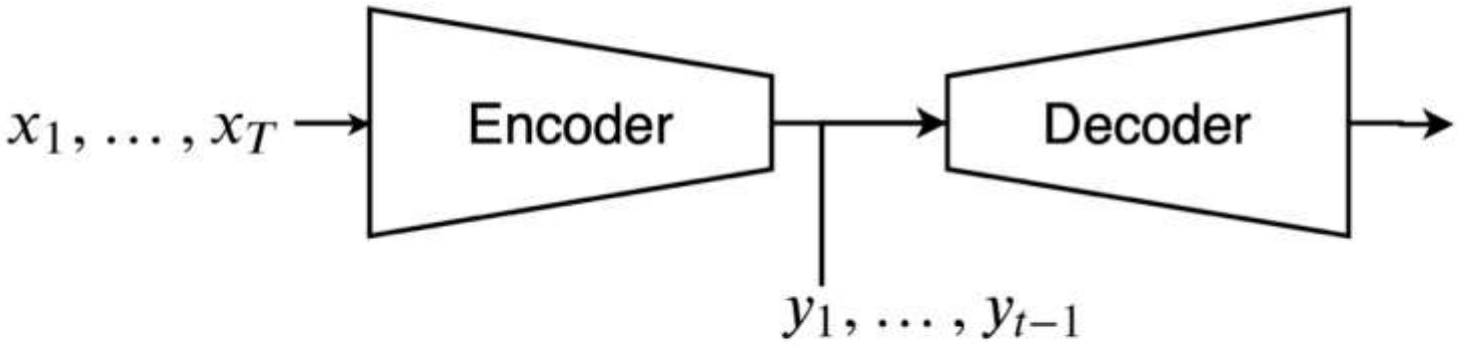
Funded by the European Union



Funded by the European Union

Neural Language Models

Input sequence: $x_1, \ldots, x_T$
Target sequence: $y_1, \ldots, y_T$



$P(Y_t = i)$

Or sometimes…

$P_\Theta(Y_t = i)$

Or sometimes…

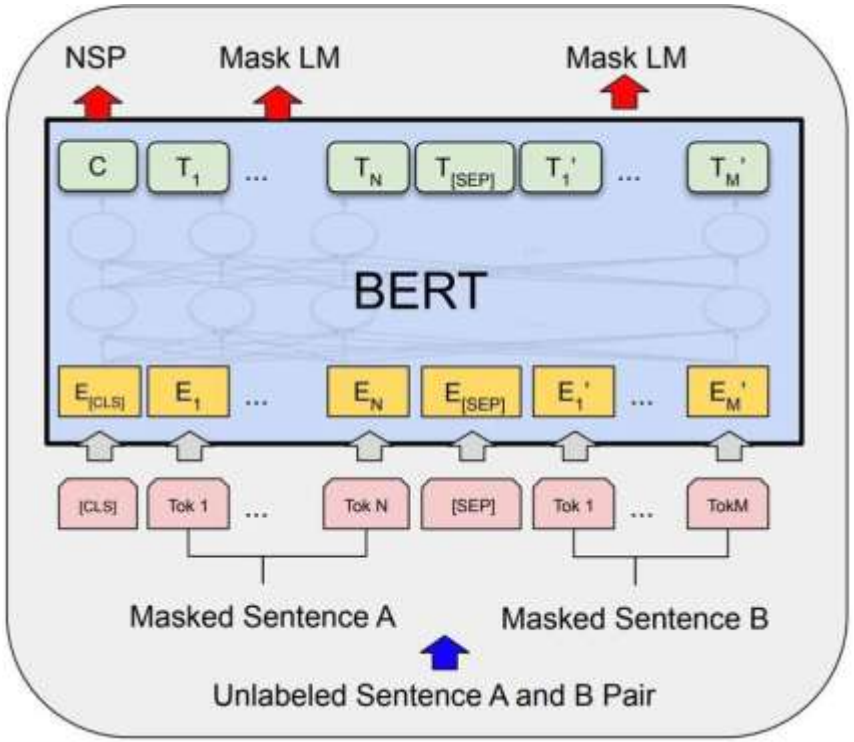
$P(Y_t = i | y_1, \ldots, y_{t-1}; x_1, \ldots, x_T; \Theta)$

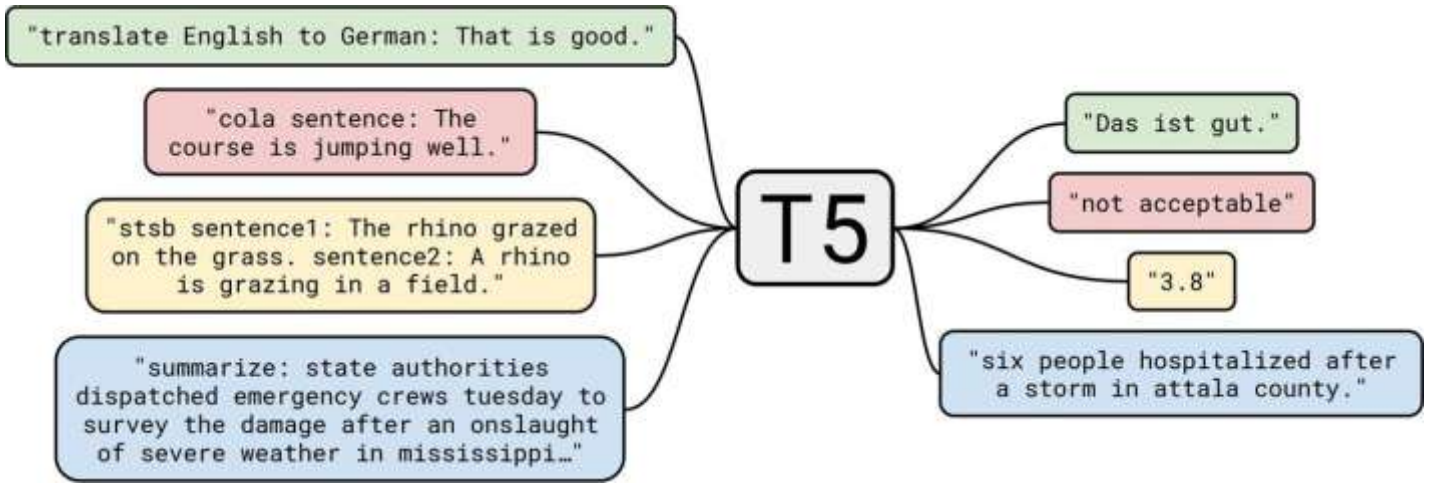
Funded by the European Union

Language models:Broad Sense

- Decoder-only models (GPT-x models)
- Encoder-only models (BERT, RoBERTa, ELECTRA)
- Encoder-decoder models (T5, BART)

The latter two usually involve a different **pre-training** objective.





Funded by the European Union

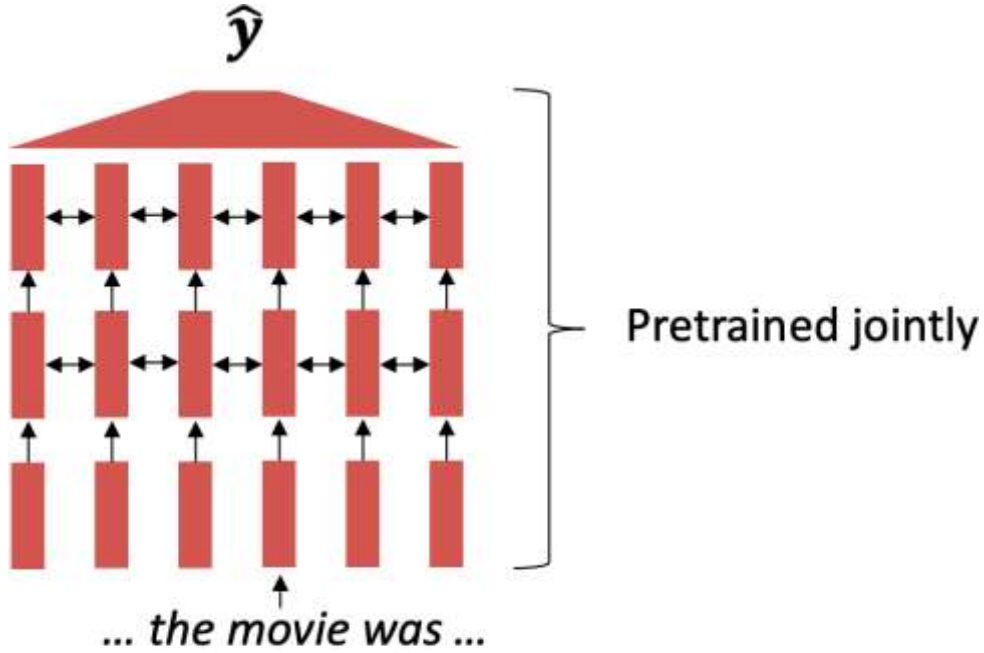
Why Pretraining?

In modern NLP:

- All (or almost all) parameters in NLP networks are initialized via **pre-training.**
- Pretraining methods hide parts of the input from the model, and then train the model to reconstruct those parts.

This has been exceptionally effective at building strong:

- **representations of language**
- **parameter initializations** for strong NLP models.
- **probability distributions** over language that we can sample from

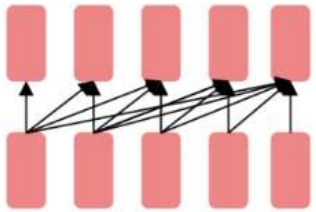


[This model has learned how to represent entire sentences through pretraining]

Funded by the European Union

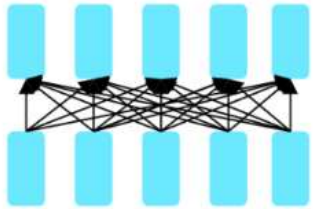
Pretraining for three types of architectures

The neural architecture influences the type of pretraining, and natural use cases.



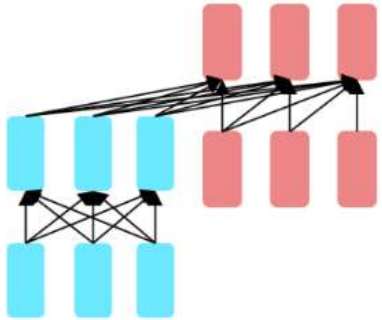
Decoders

- Language models! What we've seen so far.
- Nice to generate from; can't condition on future words
- **Examples:** GPT-2, GPT-3, LaMDA



Encoders

- Gets bidirectional context – can condition on future!
- Wait, how do we pretrain them?
- **Examples:** BERT and its many variants, e.g. RoBERTa



Encoder-Decoders

- Good parts of decoders and encoders?
- What's the best way to pretrain them?
- **Examples:** Transformer, T5, Meena

Funded by the European Union

What do we want from our learning objective.

The goals of **pre-training** (the first stage of training) are to get the language model to:

- learn the structure of natural language
- learn humans' understanding of the world (as encoded in the training data).

We want a learning objective that facilitates these goals.

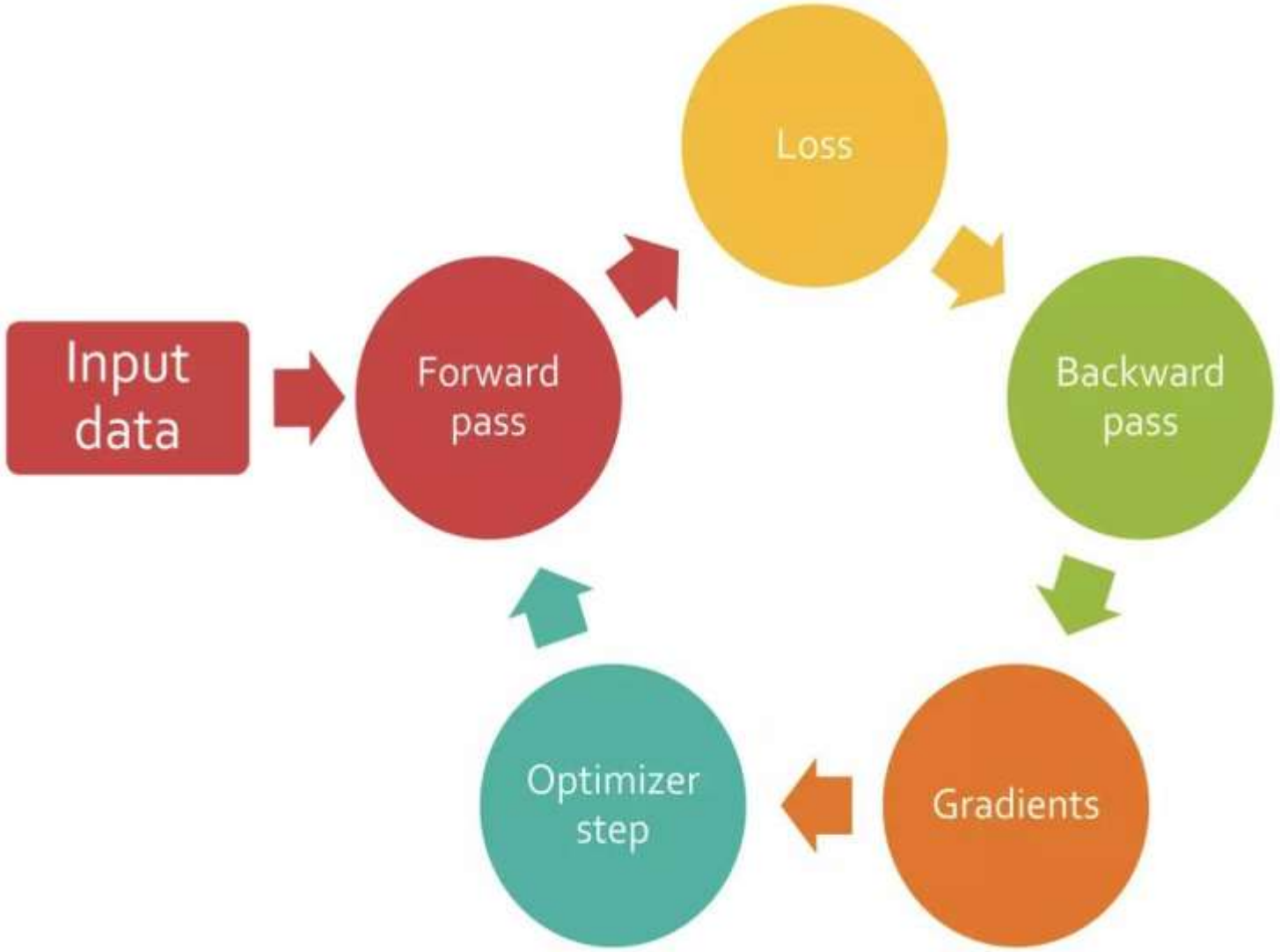
Funded by the European Union

Objectives for pre-training

- Predict a suffix given a prefix.
  - Input: I took my dog, Fido, to the
  - Target: park for his walk.
- Masked language modeling
  - Input: I took <x> to <y> his walk.
  - Target: <x> my dog, Fido, <y> the park for
- Full language modeling: predict next token given previous tokens
  - Target: I took my dog, Fido, to the park for his walk.

Funded by the European Union

Model Training Process



Funded by the European Union

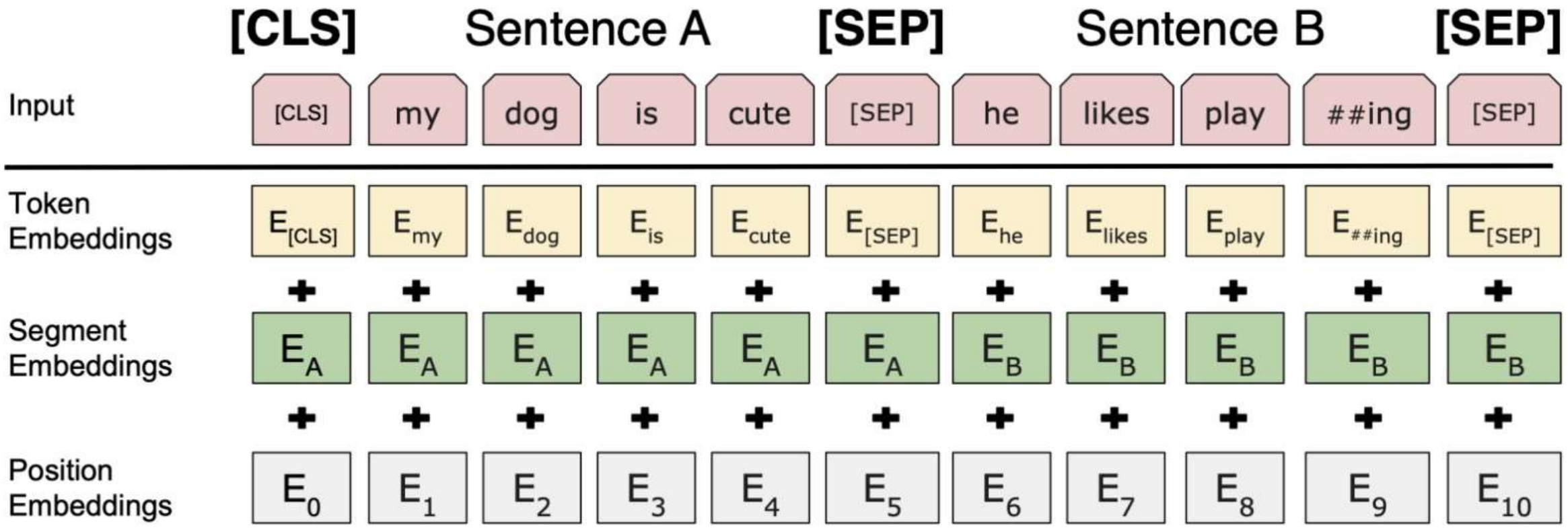
Pretraining Encoders

BERT [Devlin et al, NAACL 2019]

- **Fully bidirectional transformer encoder**
  - BERTbase: 12 layers, hidden size=768, 12 att'n heads (110M parameters)
  - BERTlarge: 24 layers, hidden size=1024, 16 att'n heads (340M parameters)
- **Input:** sum of token, positional, segment embeddings
  - **Segment embeddings** (A and B): is this token part of sentence A (before SEP) or sentence B (after SEP)?
- **[CLS]** and **[SEP]** tokens: added during pre-training
- **Pre-training tasks:**
  - Masked language modeling
  - Next sentence prediction

Funded by the European Union

BERT Input



Funded by the European Union

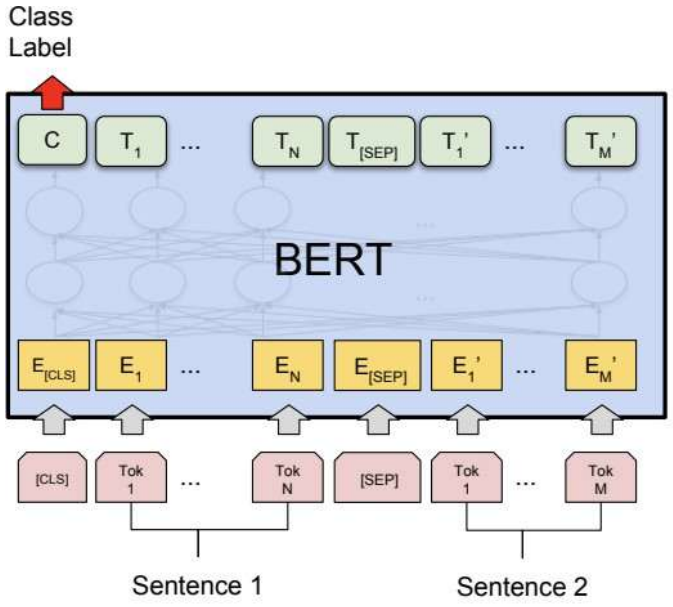
BERT Pre-training Tasks

BERT is jointly pre-trained on two tasks:

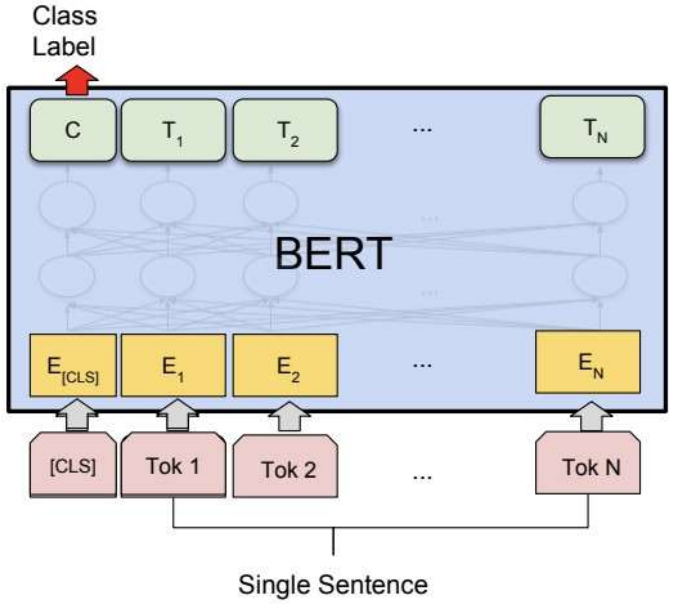
- **Next-sentence prediction:** [based on CLS token]
  - Does sentence B follow sentence A in a real document?
- **Mask language modeling:**
  - **15%** of tokens are randomly chosen as masking tokens
  - 10% of the time, a masking token remains unchanged
  - 10% of the time, a masking token is replaced by a random token
  - **80% of the time,** a masking token is replaced by **[MASK]**, and the **output layer has to predict the original token**

Funded by the European Union

Using BERT for Classification



Sentence Pair Classification

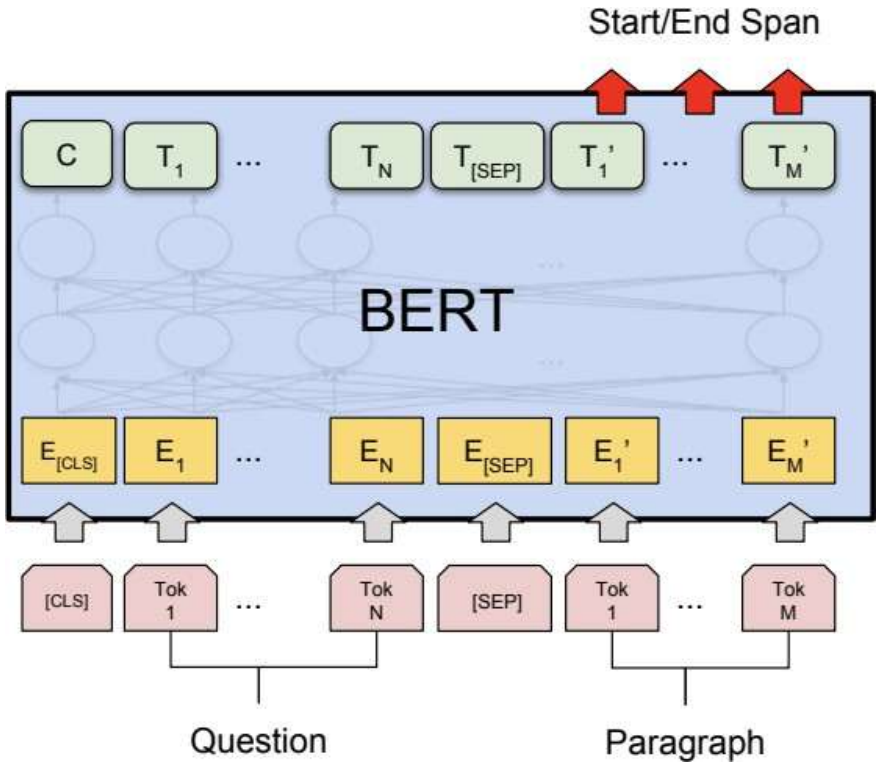


Single Sentence Classification

Add a **softmax classifier** on final layer of **[CLS]** token

Funded by the European Union

Using BERT for Question-Answering

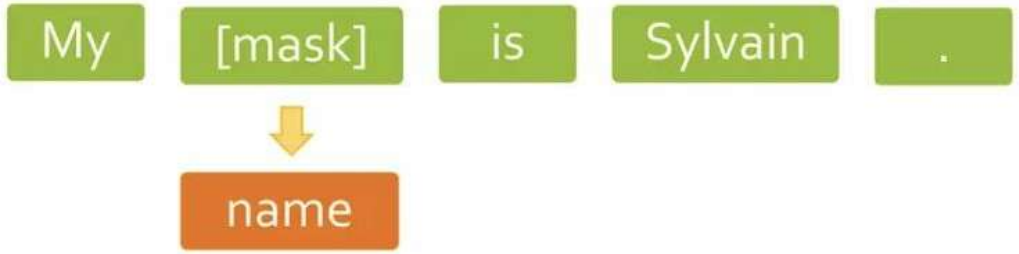


Input: [CLS] question [SEP] answer passage [SEP]
Learn to predict a START and an END token on answer tokens

Funded by the European Union

Examples of language models pretraining objectives

Guess the next word in the sentence (GPT)



Guess some masked words in the sentence (BERT)

Why not encoder-based LLMs?

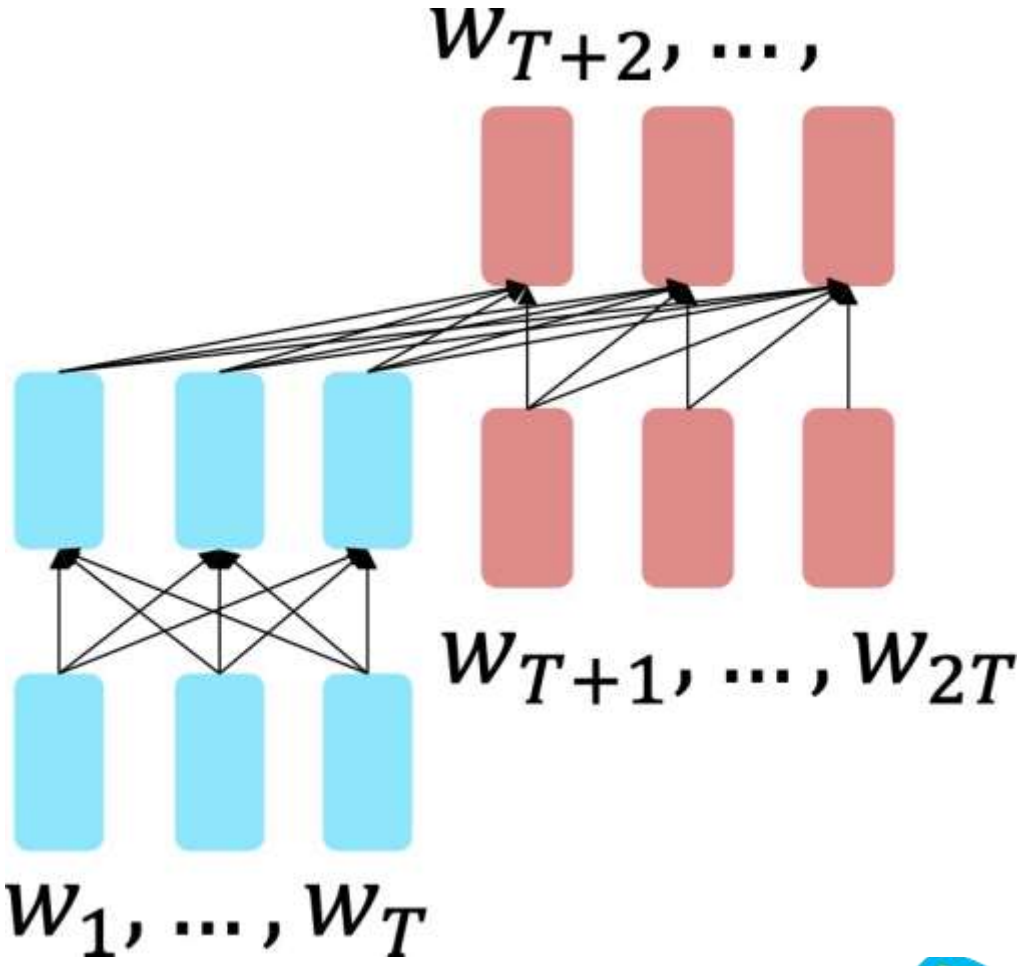
1. **It cannot generate (it discriminates)**: It can only work for classification (discrimination) tasks, it is not easy to generate something new.

1. **Its objective is not scalable:** Its self-supervised tasks (masked language model) are just too simple for LLMs, and increasing model size does not improve performance too much.

Funded by the European Union

Pretraining Encoder-Decoders

The **encoder** portion benefits from bidirectional context; the **decoder** portion is used to train the whole model through language modeling.



Raffel, Colin, et al. "Exploring the limits of transfer learning with a unified text-to-text transformer." The Journal of Machine Learning Research 21.1 (2020): 5485-5551.

Funded by the European Union

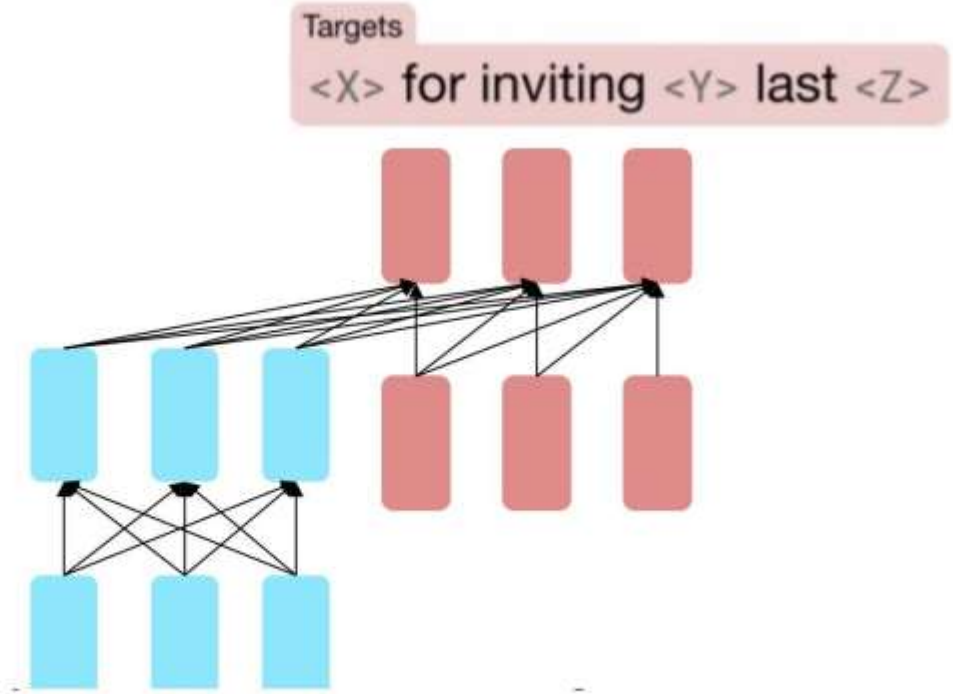
Pretraining Encoder-Decoders: Span Corruption

What [Raffel et al., 2018] found to work best was span corruption. Their model: T5.

Replace different-length spans from the input with unique placeholders; decode out the spans that were removed!

Original text

Thank you ~~for inviting~~ me to your party ~~last~~ week.



This is implemented in text preprocessing: it's still an objective that looks like **language modeling** at the decoder side.

Funded by the European Union

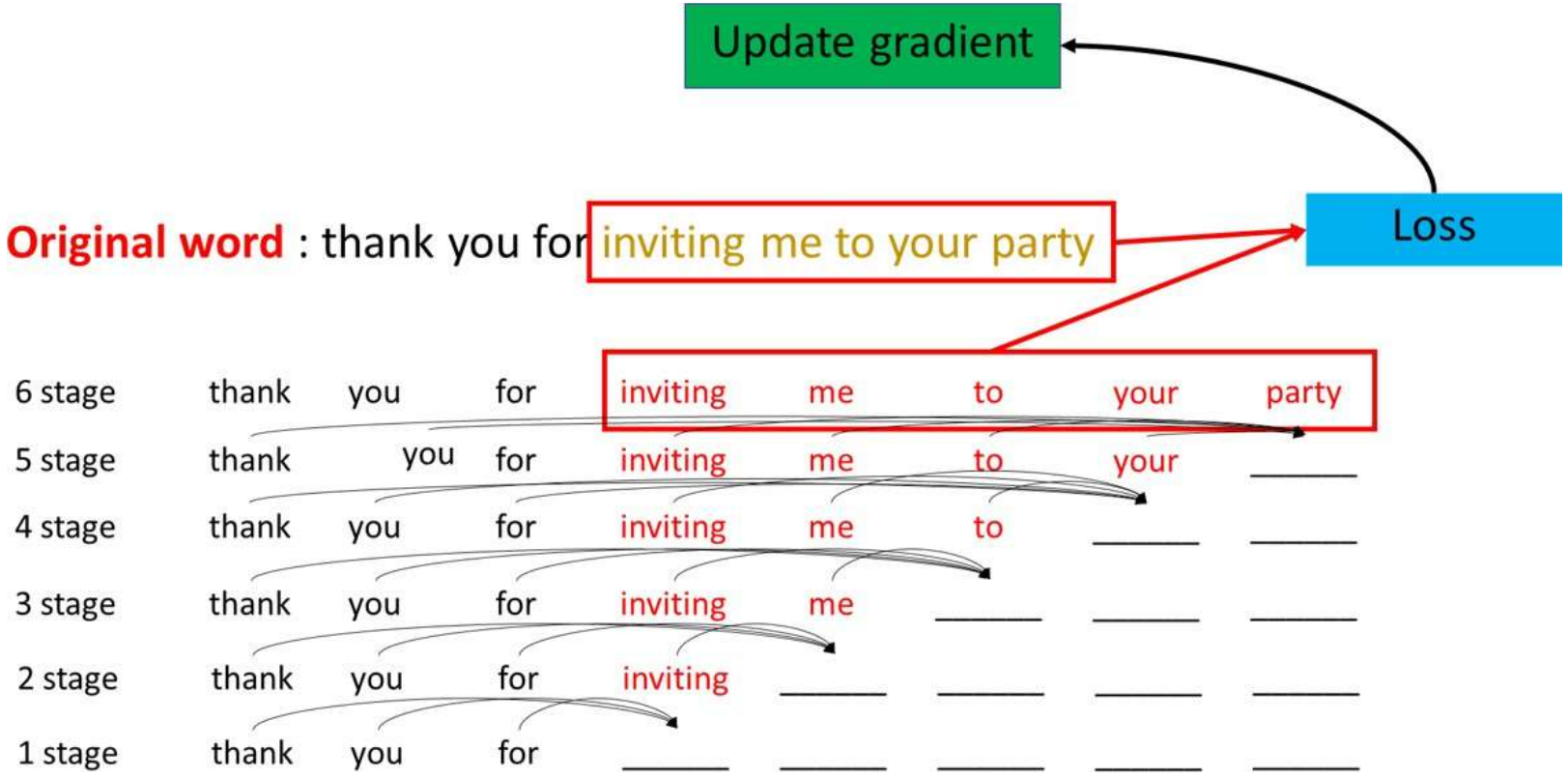
Why not Encoder-Decoder LLMs?

1. Decoder could work also as a seq-2-seq task, its protocol is much easier
2. When performing multi-turn generation, it is not easy to cache previous values.

Funded by the European Union

Pretraining Decoders

It's natural to pretrain decoders as language models and then use them as generators, finetuning their $p_\theta(w_t|w_{1:t-1})$!



Funded by the European Union

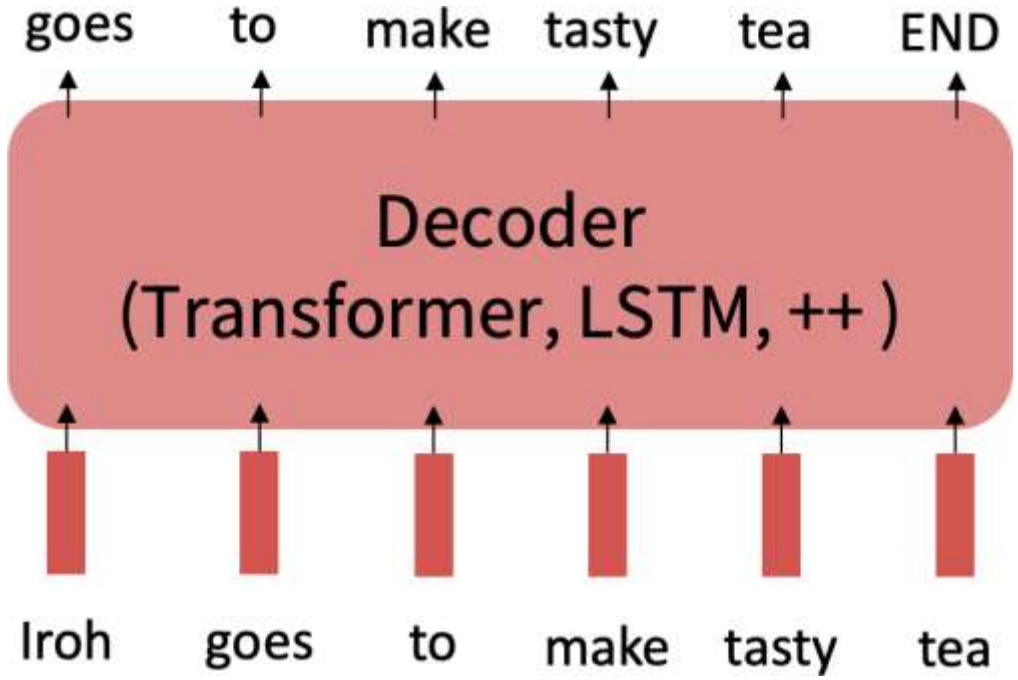
Pretraining through language modeling

Recall the language modeling task:

- Model the probability distribution over words given their past contexts.
- There's lots of data for this! (In English.)

Pretraining through language modeling:

- Train a neural network to perform language modeling on a large amount of text.
- Save the network parameters.



Funded by the European Union

Common roadmap for LLMs

- Phase 1: pre-training
  - Learn **general** world knowledge, ability, etc.
- Phase 2: Supervised finetuning
  - Tailor to **tasks** (**unlock** some abilities)
- Phase 3: RLHF
  - Tailor to **humans**
  - *Even you could teach ChatGPT to do something*

Funded by the European Union

Parallel Training

Parallel Training: Overview

As scale grows, training with one GPU is not enough

- There are many ways to improve efficiency on single-GPU training
  - Checkpointing: moving part of the operations to CPU memory
  - Quantizing different part of the optimization to reduce GPU memory cost
- Eventually more FLOPs are needed

Different setups of parallel training:

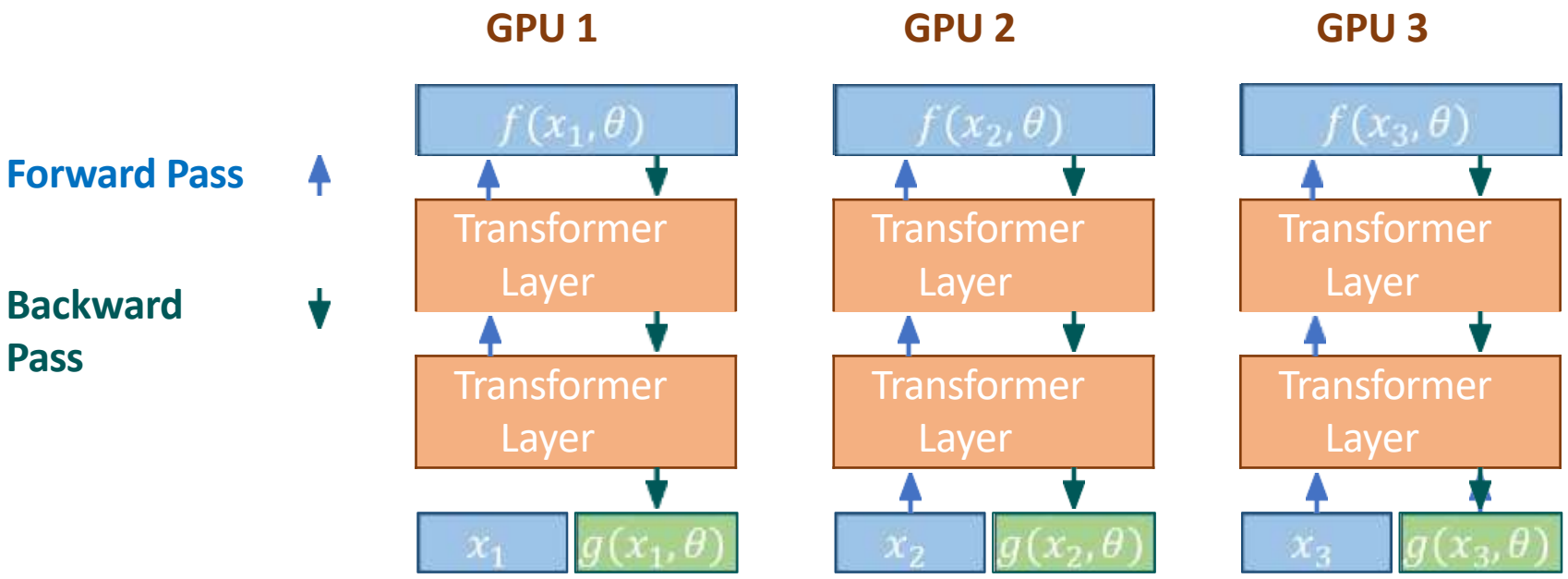
- When model training can fit into single-GPU
  →Data parallelism
- When model training cannot fit into single-GPU
  → Model parallelism: pipeline or tensor

Funded by the European Union

Parallel Training: Data Parallelism

Split training data batch into different GPUs

- Each GPU maintains its own copy of model and optimizer
- Each GPU gets a different local data batch, calculates its gradients

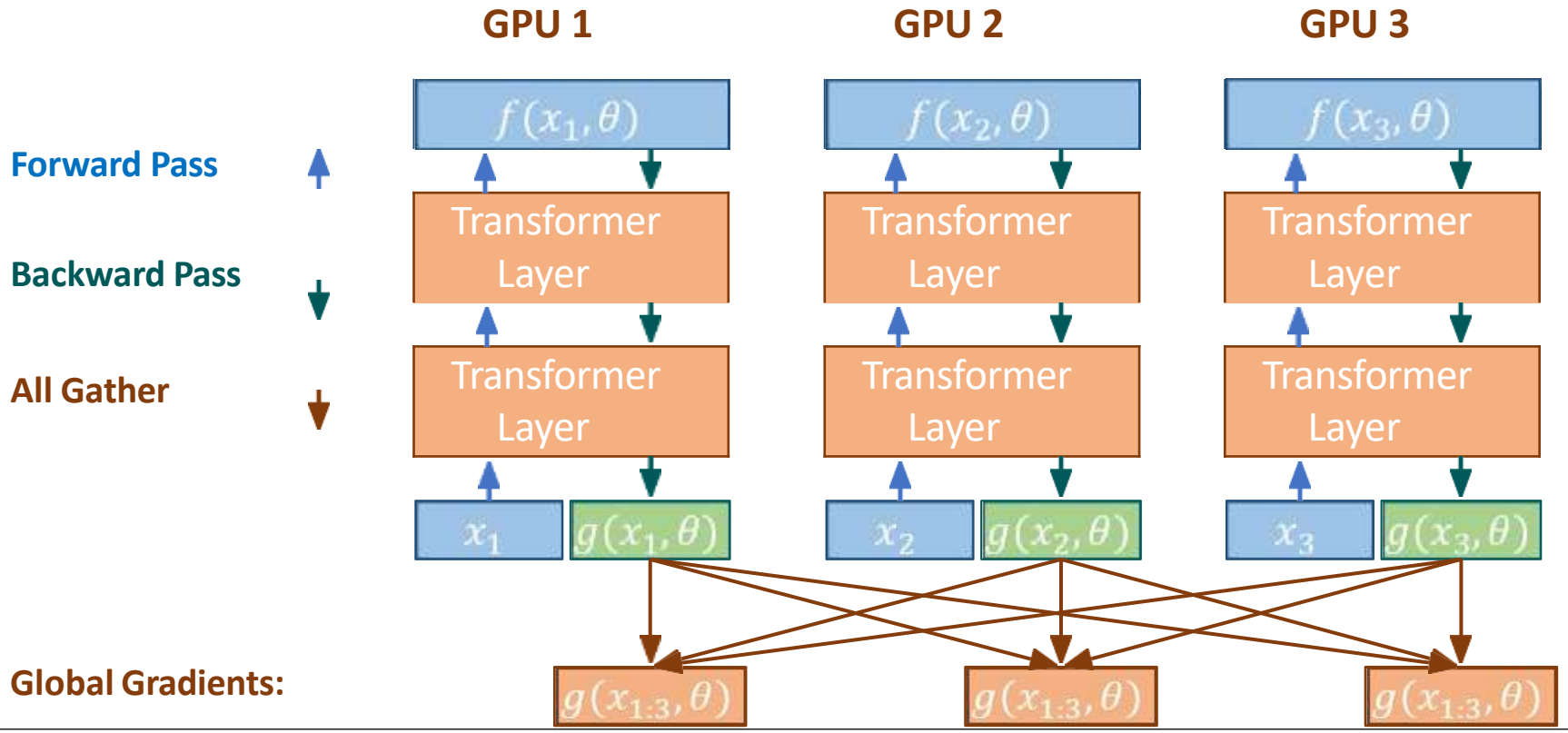


Funded by the European Union

Parallel Training: Data Parallelism

Split training data batch into different GPUs

- Each GPU maintains its own copy of model and optimizer
- Each GPU gets a different local data batch, calculates its gradients
- Gather local gradients together to each GPU for global updates

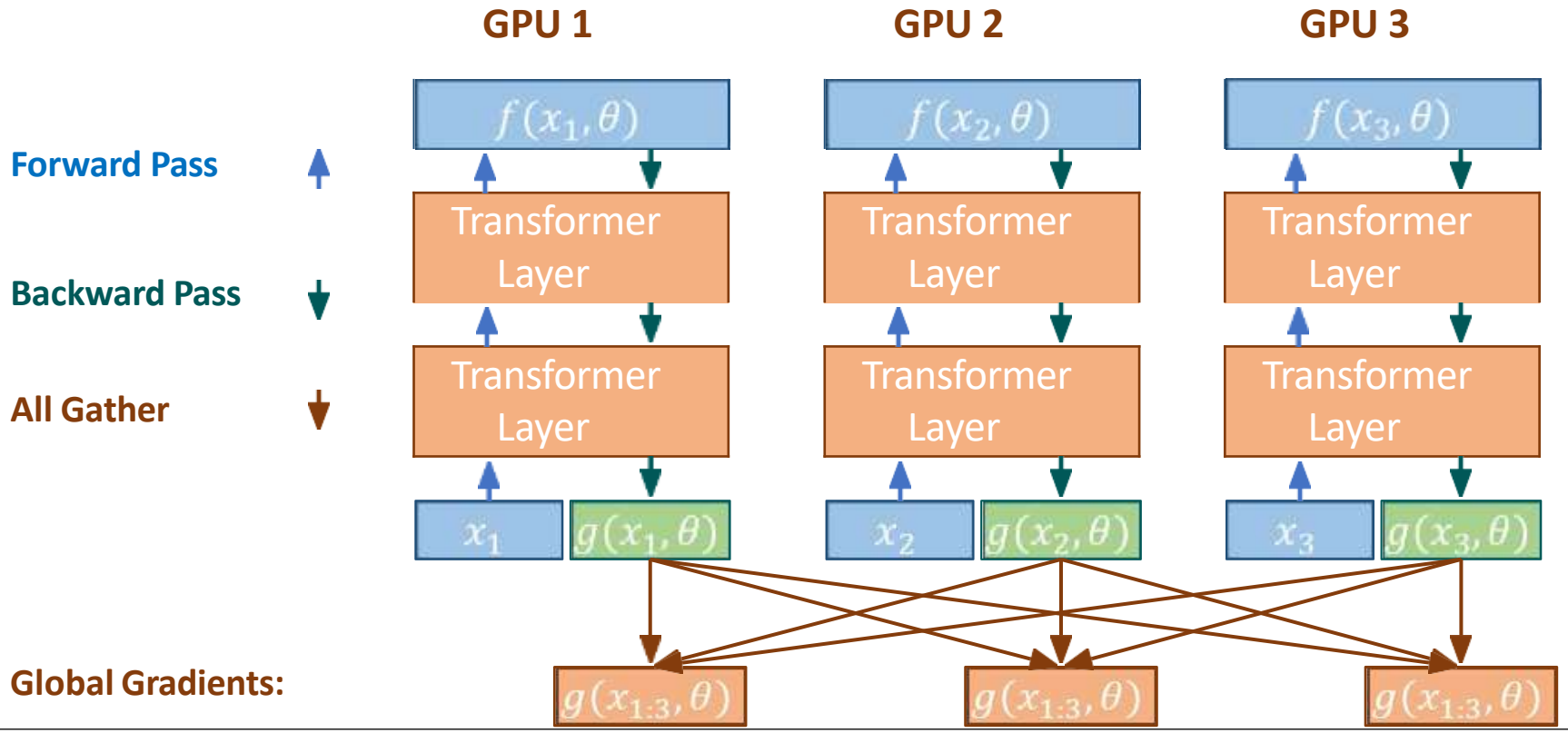


Funded by the European Union

Parallel Training: Data Parallelism

Split training data batch into different GPUs

- Each GPU maintains its own copy of model and optimizer
- Each GPU gets a different local data batch, calculates its gradients
- Gather local gradients together to each GPU for global updates



Communication:

- The full gradient tensor between every pair of GPUs, at each training batch.
- Not an issue between GPUs in the same machine or machines with infinity band
- Will need work around without fast cross-GPU connection

Funded by the European Union

Parallel Training: Model Parallelism

LLM size grew quickly and passed the limit of single GPU memory

	Cost of 10B Model
Parameter Bytes	20GB
Gradient Bytes	20GB
Optimizer State: 1st Order Momentum	20GB
Optimizer State: 2nd Order Momentum	20GB
Total Per Model Instance	80GB

Memory Consumption of Training Solely with **BF16** (Ideal case) of a model sized Ψ

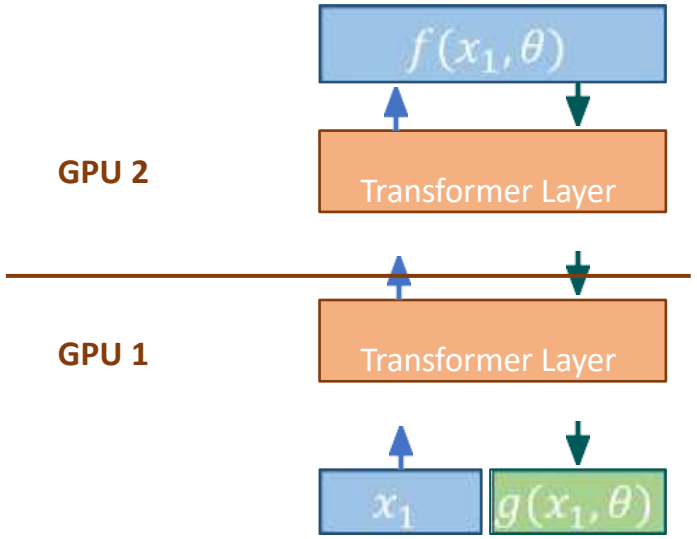
Solution: Split network parameters (thus their gradients and corresponding optimizer states) to different GPUs

Funded by the European Union

Parallel Training: Model Parallelism

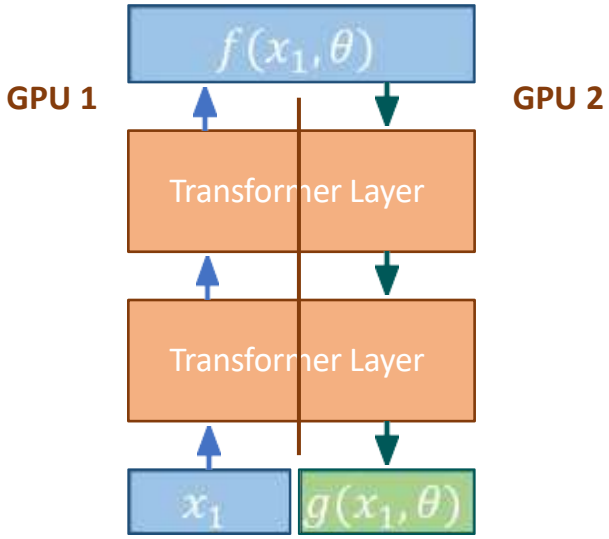
Two ways of splitting network parameters

Pipeline Parallelism



Split by Layers

Tensor Parallelism



Split Tensors

Funded by the European Union

Parallel Training: Pipeline Parallelism

Split network by layers, aligning devices by layer order to a pipeline, and pass data through devices [7]

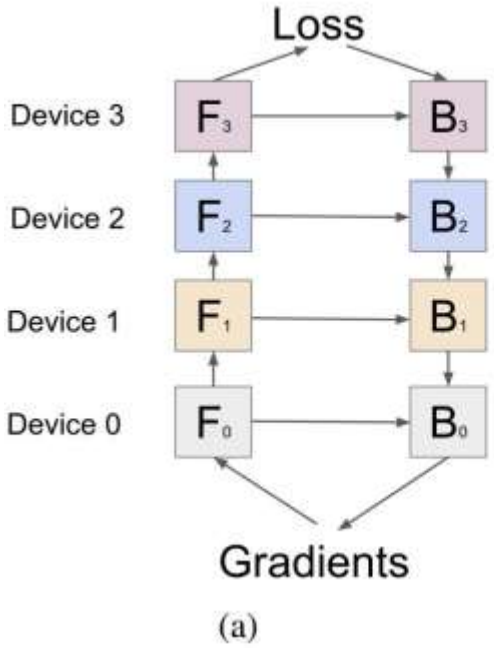
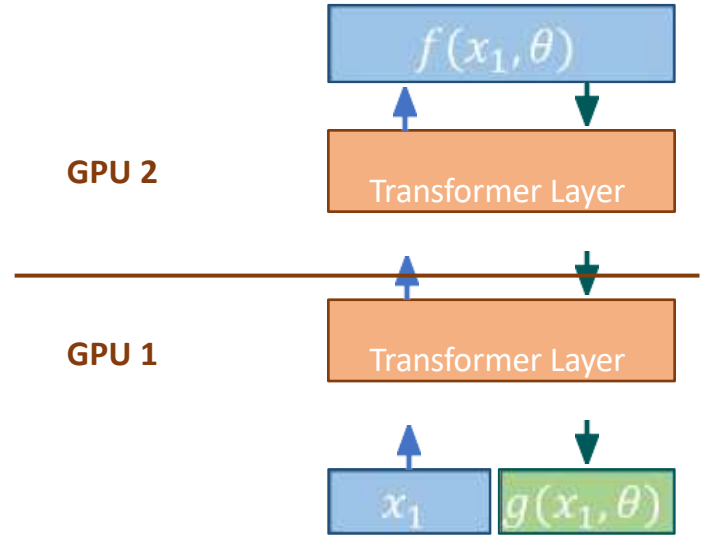


Illustration of Pipeline Parallelism [7]

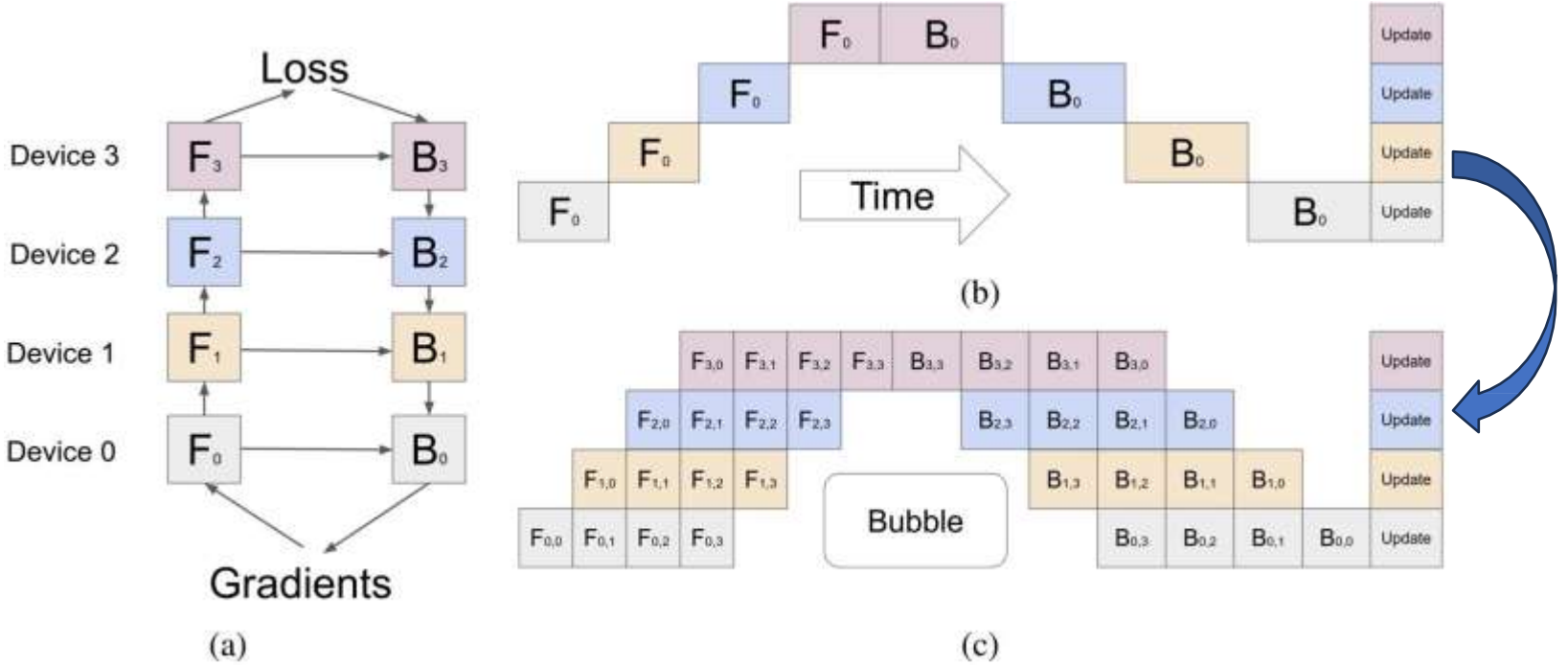
Pipeline Parallelism



Split by Layers

Parallel Training: Pipeline Parallelism

Split network by layers, aligning devices by layer order to a pipeline, and pass data through devices [7]



Split batches for more fine-grained pipelines

Illustration of Pipeline Parallelism [7]

Parallel Training: Pipeline Parallelism

Split network by layers, aligning devices by layer order to a pipeline, and pass data through devices [7]

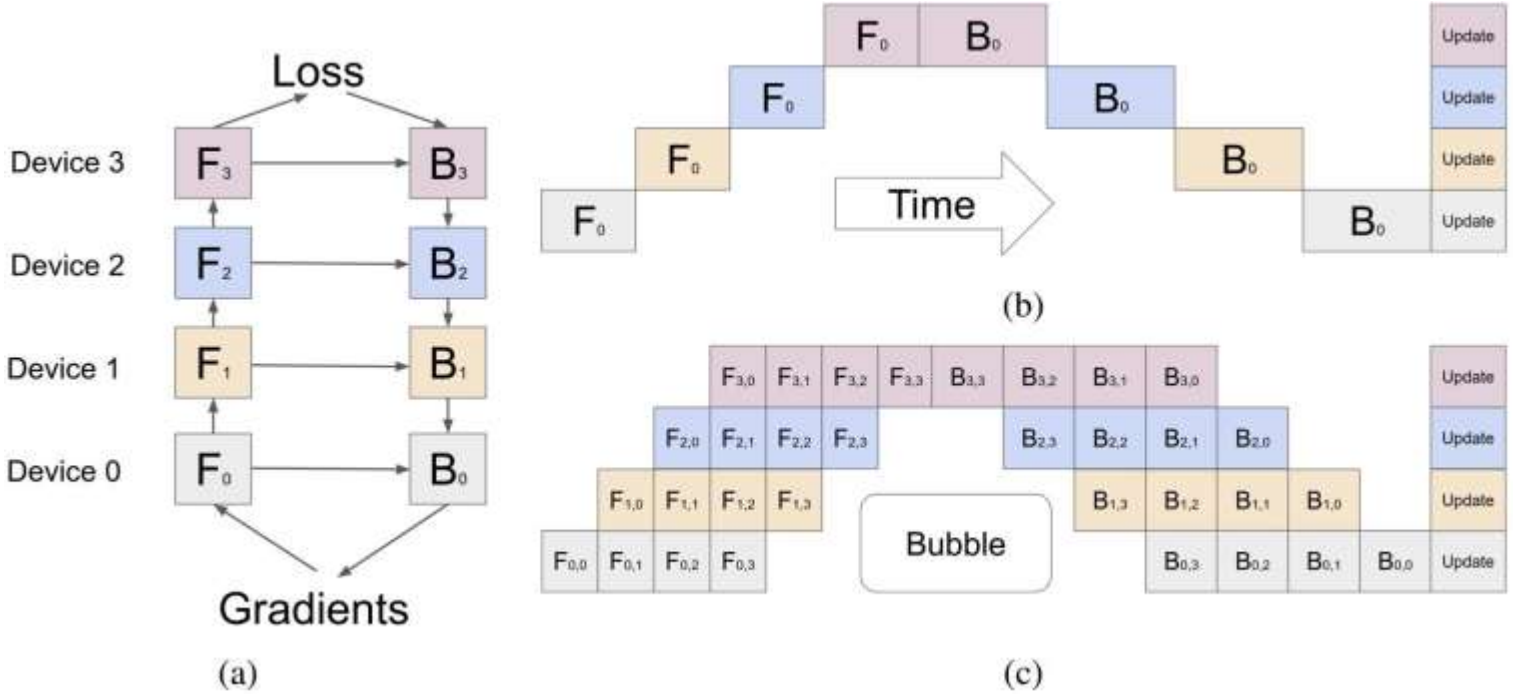


Illustration of Pipeline Parallelism [7]

Communication:

- Activations between nearby devices in forward pass
- Partial gradients between nearby devices in backward

Parallel Training: Pipeline Parallelism

Split network by layers, aligning devices by layer order to a pipeline, and pass data through devices [7]

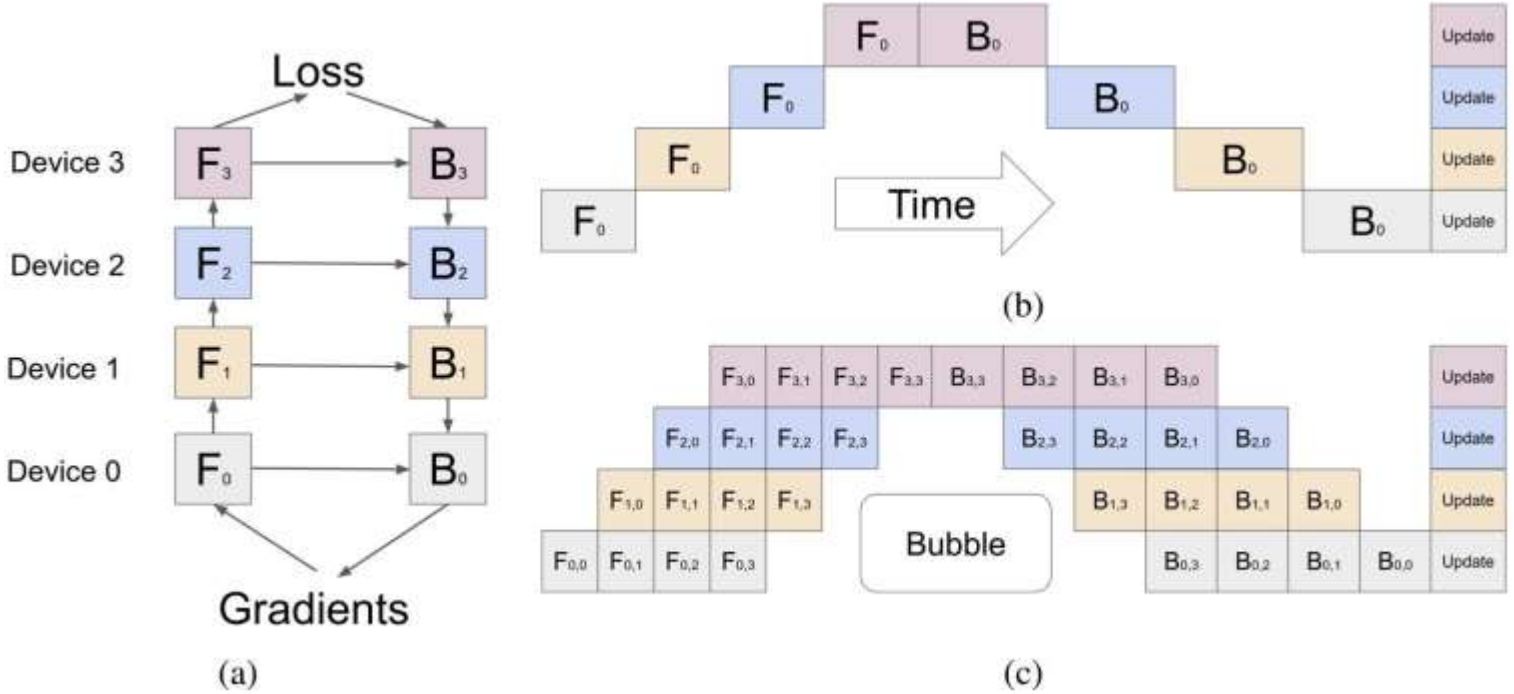


Illustration of Pipeline Parallelism [7]

Communication:

- Activations between nearby devices in forward pass
- Partial gradients between nearby devices in backward

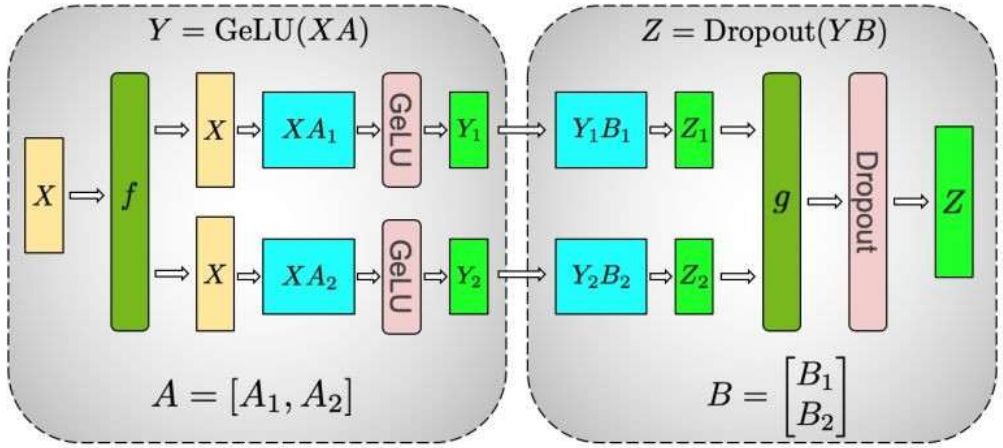
Pros: Conceptually simple and not coupled with network architectures. All networks have multiple layers.

Cons: Waste of compute in the Bubble. Bubble gets bigger with more devices and bigger batches.

Funded by the European Union

Parallel Training: Tensor Parallelism

Split the parameter tensors of network layers into different devices for parallel matrix operations

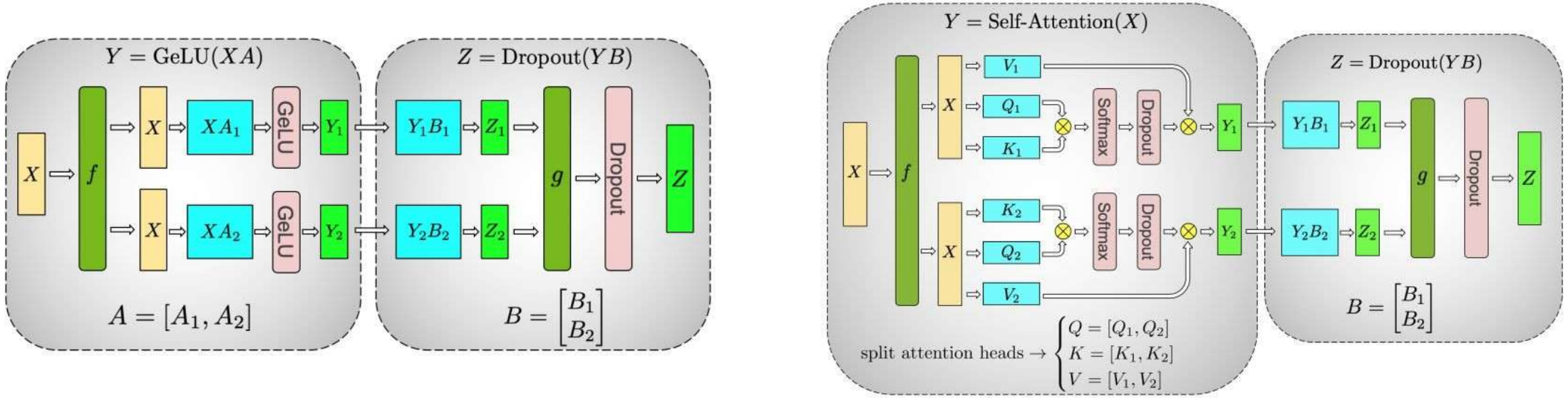


Tensor Parallelism of MLP blocks and Self-attention Blocks [8]

Funded by the European Union

Parallel Training: Tensor Parallelism

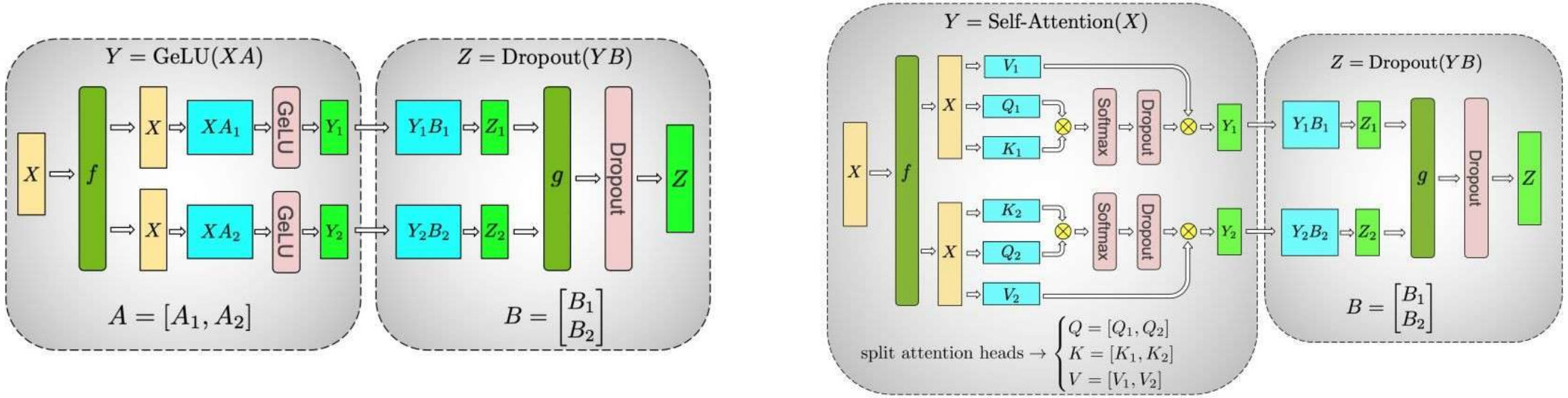
Split the parameter tensors of network layers into different devices for parallel matrix operations



Tensor Parallelism of MLP blocks and Self-attention Blocks [8]

Parallel Training: Tensor Parallelism

Split the parameter tensors of network layers into different devices for parallel matrix operations



Tensor Parallelism of MLP blocks and Self-attention Blocks [8]

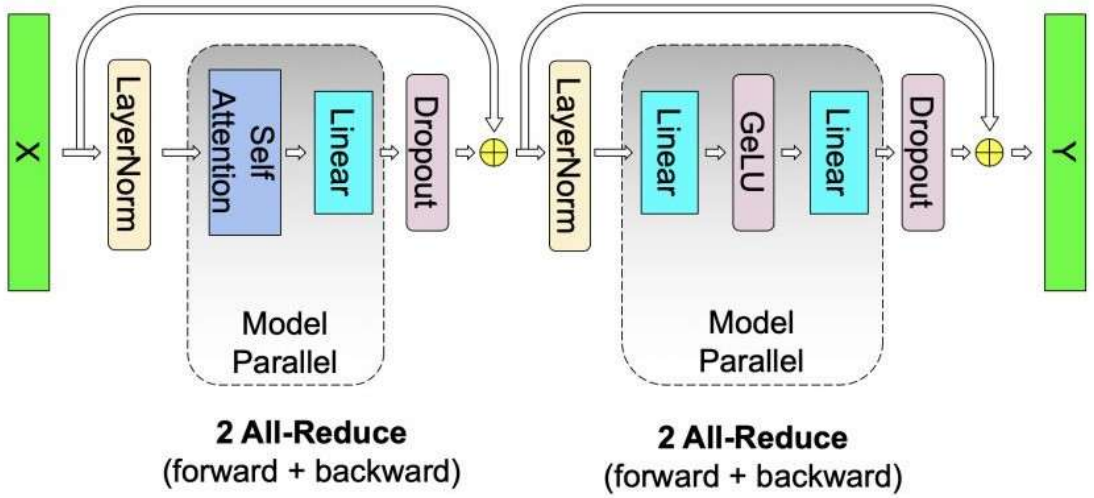
Pros: No bubble

Cons: Different blocks are better split differently, lots of customizations

Funded by the European Union

Parallel Training: Tensor Parallelism

Split the parameter tensors of network layers into different devices for parallel matrix operations



Communication of Tensor Papalism for a Transformer Layer [8]

Communication:

- All-gather of partial activations and gradients for each split tensor

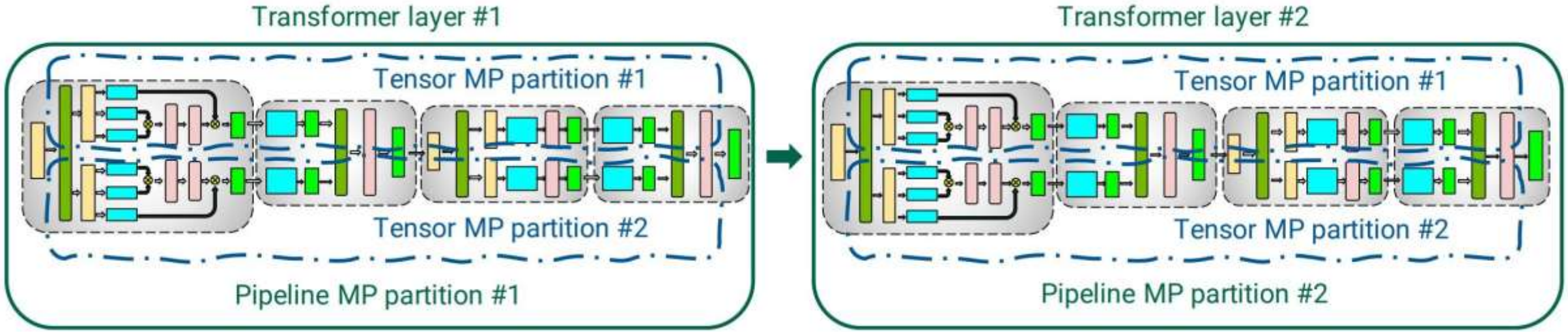
Funded by the European Union

Parallel Training: Combining Different Approaches

Often data parallelism and model parallelism are used together.

- No need not to use data parallelism

Pipeline Parallelism and Tensor Parallelism can also be used together.



Combination of Tensor Parallelism and Pipeline Parallelism [9]

Funded by the European Union

Distributed Training and Memory Optimizations - ZeRO: Train Trillion-scale models

Funded by the European Union

Let’s take a step back for training a singler layer in practice

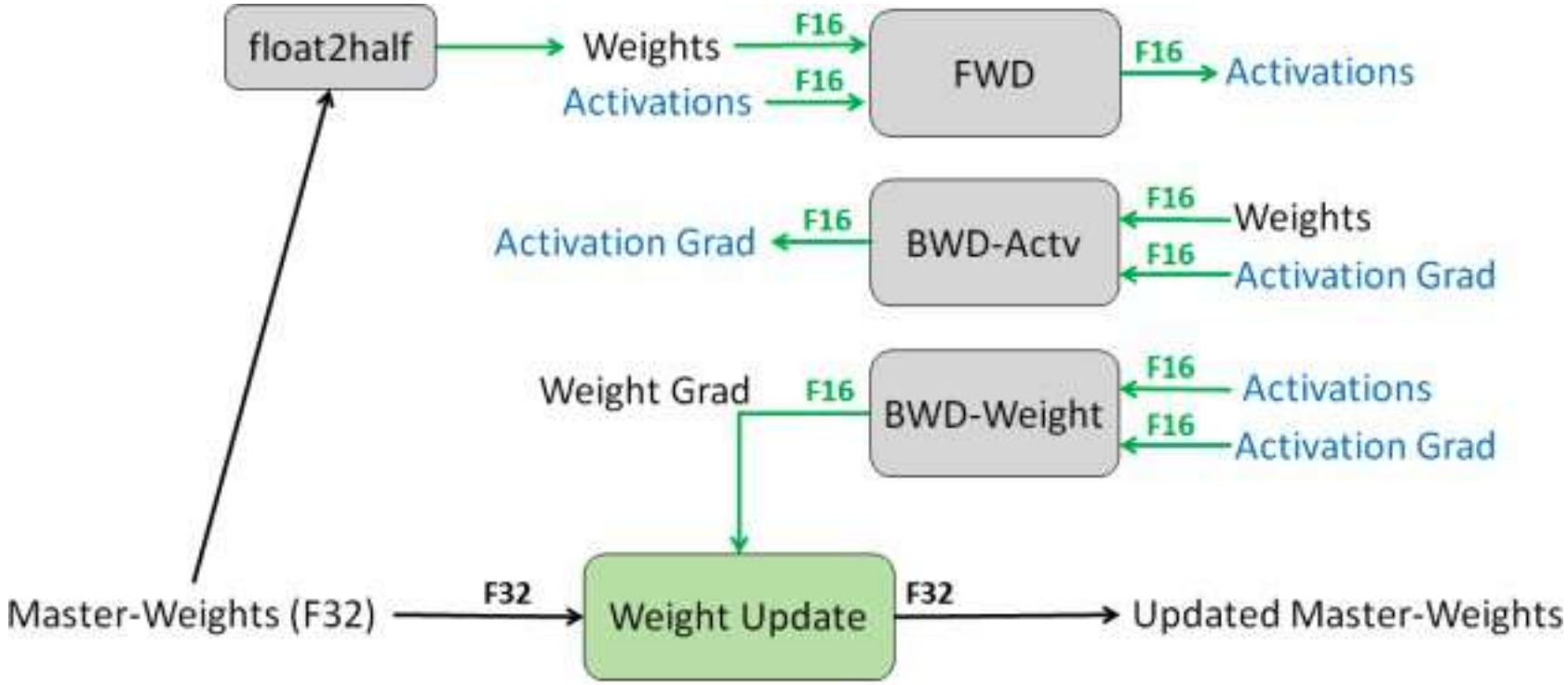


Figure 1: Mixed precision training iteration for a layer.

Funded by the European Union

Memory Consumptions for this example:

Suppose the layer (or model) is trained using Adam Optimizer.
The number of parameters are Φ.

Then in a single training iteration, we have to save (corresponding memory consumption):

- Model parameters (fp16): 2Φ
- Model gradients (fp16): 2Φ
- Adam Optimizer states - copy of Parameters, Momentum and Variance (fp32): $4\Phi + 4\Phi + 4\Phi = 12\Phi$
- Residual states, including activations, buffer, fragmentations

For a GPT-2 model, even it has only 1.5B model parameters (3GB memory is enough to hold it), training it would cost at least 24GB memory!

Funded by the European Union

VRam Estimation

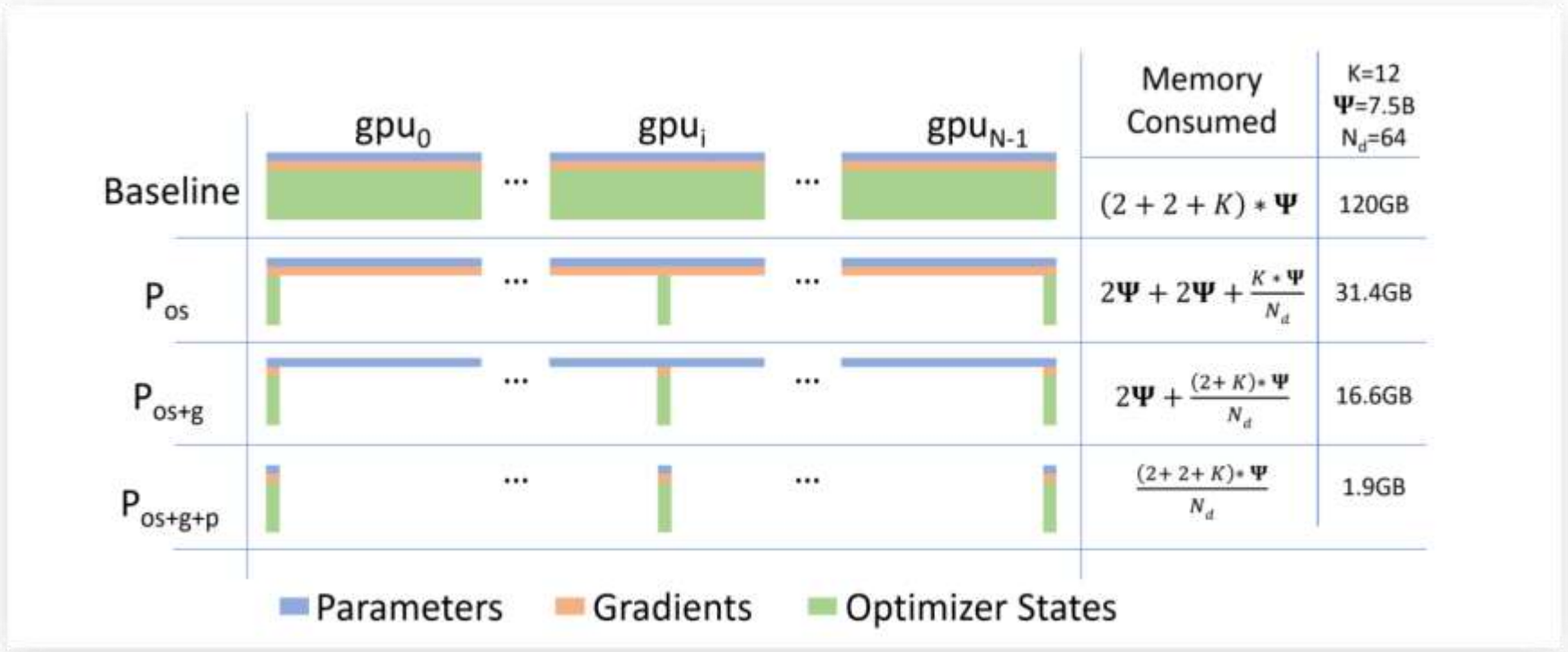
Model: HuatuoGPT-7B

1. Model
    a. Param(fp16): 7B*2=14GB
    b. Grad(fp16):7B*2=14GB
2. **Optimizer(AdamW)**
    a. Master Weights(fp32): 7B*4=28GB
    b. Adam m(fp32): 7B*4=28GB
    c. Adam v(fp32): 7B*4=28GB
3. Activation
4. Buffer&Fragmentation

Funded by the European Union

Parallel Strategy: ZeRO

1. ZeRO-DP: Shard the optimizer state
2. ZeRO-1&2: Same communication volume as DP
3. ZeRO-3: 1.5 communication volume as DP



K denotes the memory multiplier of optimizer states, and N denotes DP degree

[1] [1910.02054] ZeRO: Memory Optimizations Toward Training Trillion Parameter Models

Funded by the European Union

ZeRO: the More GPUs, the Less Memory Consumption!

DP	7.5B Model (GB)			128B Model (GB)			1T Model (GB)		
	P_{os}	P_{os+g}	P_{os+g+p}	P_{os}	P_{os+g}	P_{os+g+p}	P_{os}	P_{os+g}	P_{os+g+p}
1	120	120	120	2048	2048	2048	16000	16000	16000
4	52.5	41.3	**30**	896	704	512	7000	5500	4000
16	35.6	**21.6**	7.5	608	368	128	4750	2875	1000
64	**31.4**	16.6	1.88	536	284	**32**	4187	2218	250
256	30.4	15.4	0.47	518	263	8	4046	2054	62.5
1024	30.1	15.1	0.12	513	257	2	4011	2013	**15.6**

- We can train a 7.5B model (like Llama2) using only 4 V100-32GB GPUs
- We can even train a 128B model using 64 V100-32GB GPUs

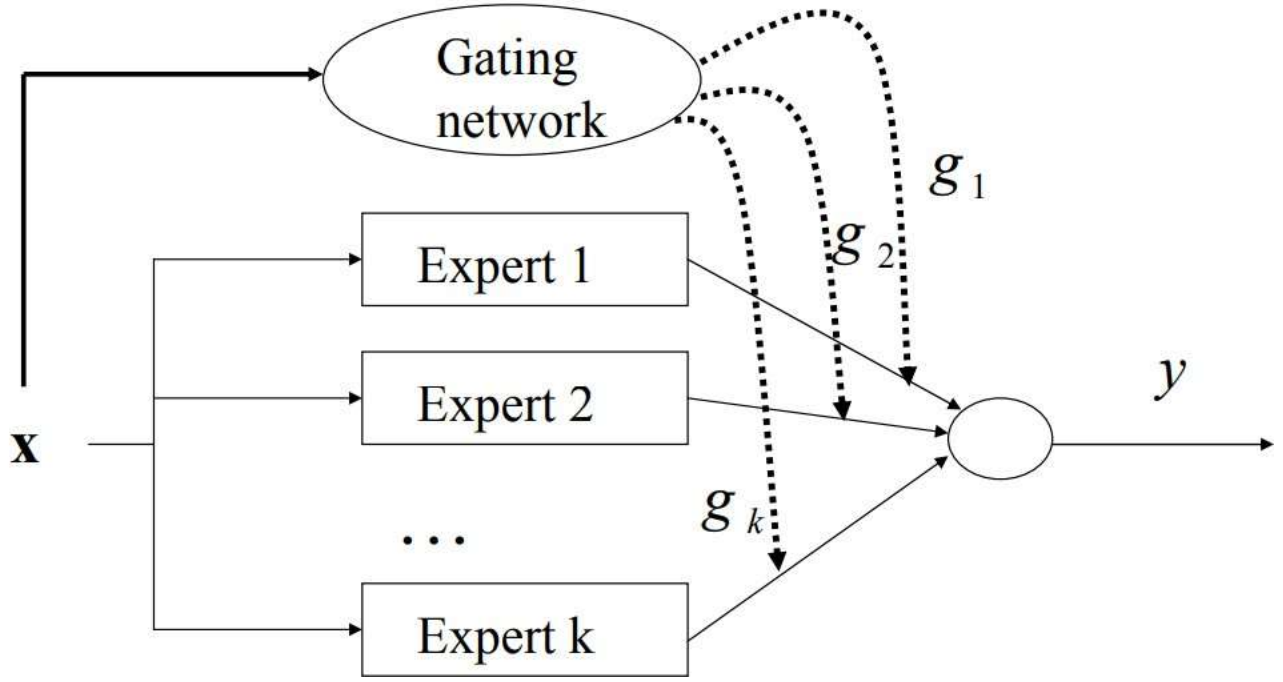
Funded by the European Union

Mixture of Experts (MoE)

Mixture of Expert

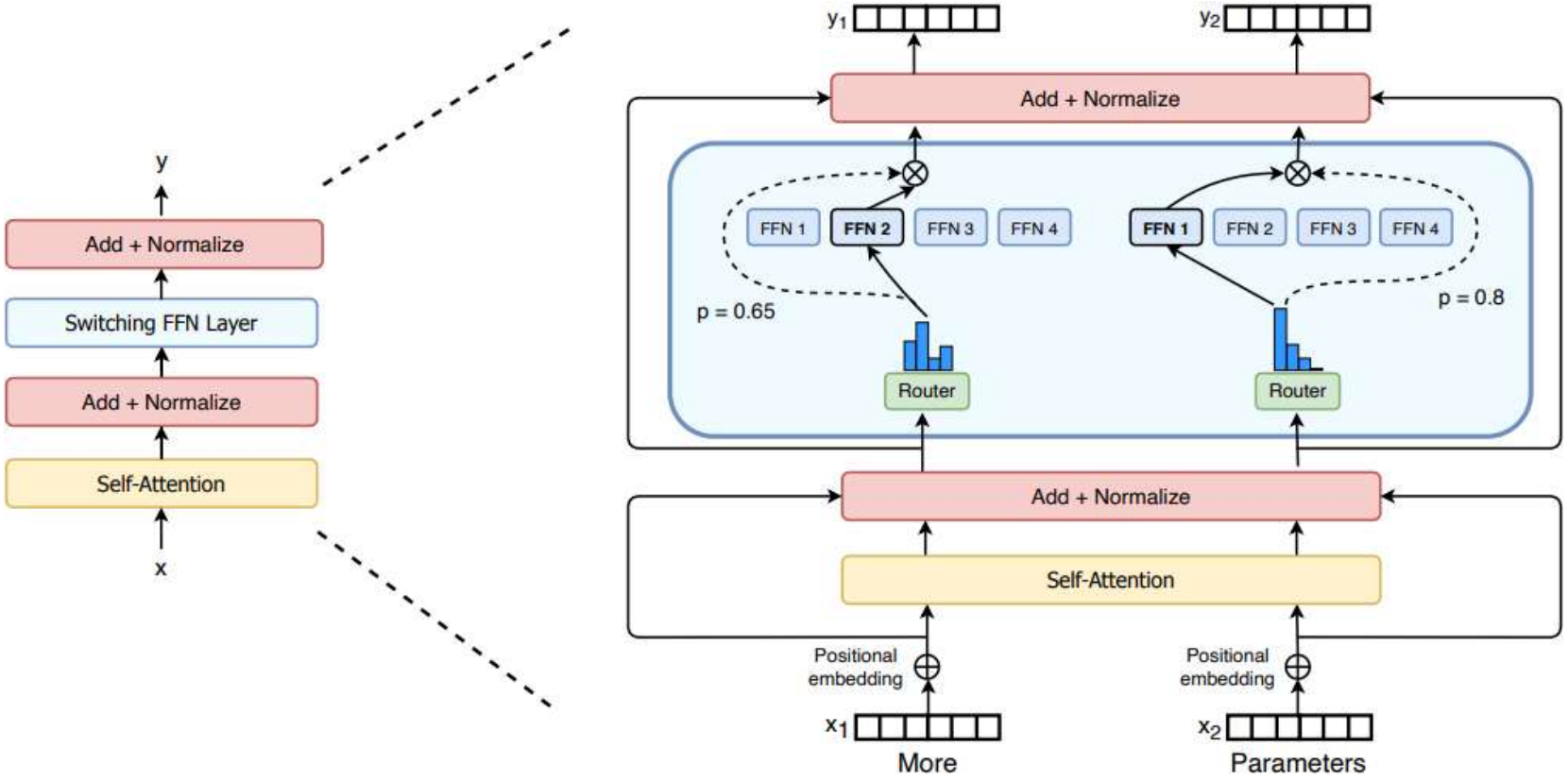
- **Gating network** : decides what expert to use

$g_1, g_2, \ldots g_k$ - gating functions



Funded by the European Union

MOE in Transformer



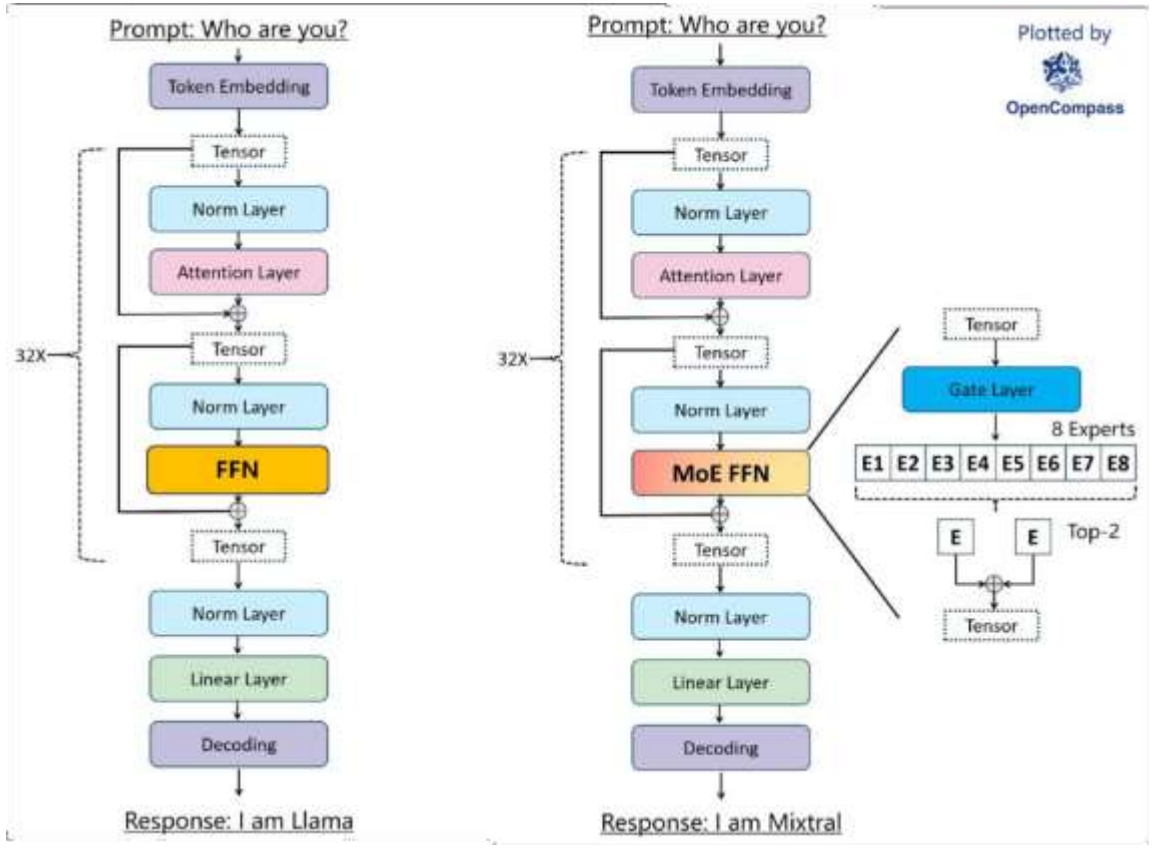
Token-specific routing

William Fedus, Barret Zoph, Noam Shazeer. Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity. https://arxiv.org/pdf/2101.03961.pdf

Funded by the European Union

Mixture of Expert

- **Model Architectures**



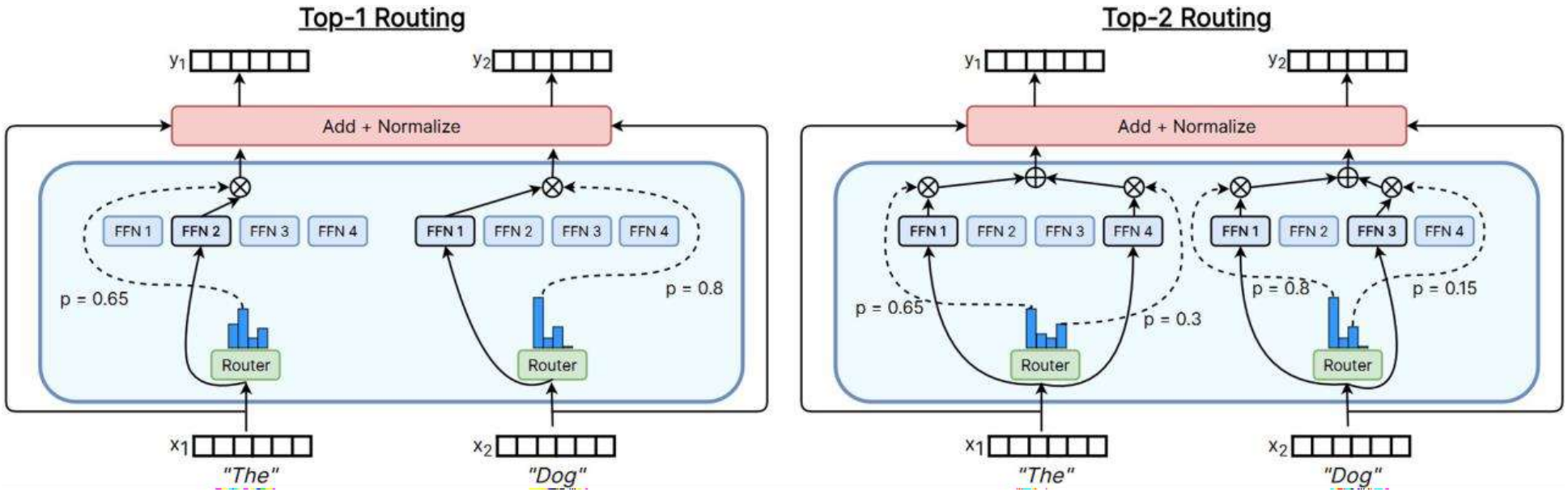
Key points:

Activate different experts parameters for each input token.

Sparse activation. Not all parameters are activated.

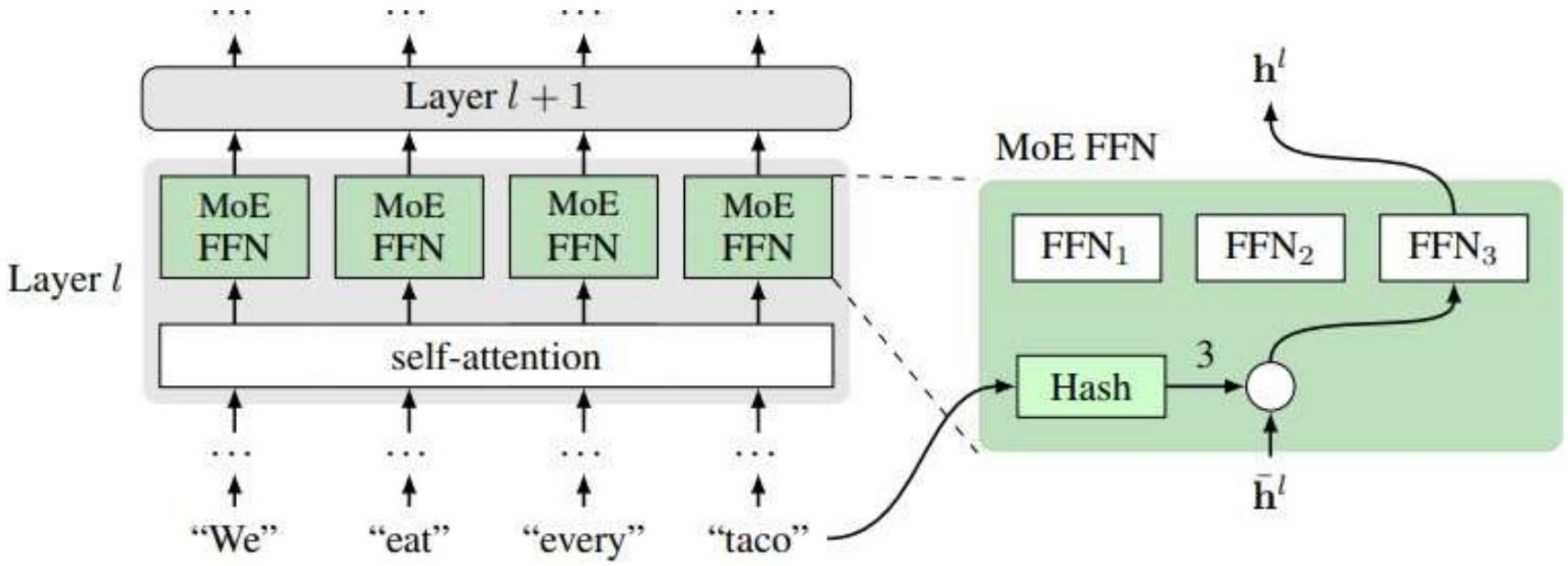
Funded by the European Union

Routing Algorithms



Funded by the European Union

An example of Hash



Stephen Roller, Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston. Hash Layers For Large Sparse Models. https://arxiv.org/pdf/2106.04426.pdf

Funded by the European Union

Random hash also works

Table 3: **Different Hash Layering Methods** on pushshift.io Reddit.

Model	Hashing Type	Valid PPL	Test PPL
Baseline Transformer	-	24.90	24.96
Hash Layer 1x64	Balanced assignment	23.16	23.23
Hash Layer 1x64	Fixed random assignment	23.22	23.27
Hash Layer 1x64	Token clustering (using Baseline Transformer)	23.90	23.99
Hash Layer 1x64	Dispersed Hash (within token clusters)	23.17	23.22
Hash Layer 1x64	Hash on position	25.07	25.14
Hash Layer 1x64	Bigrams	24.19	24.28
Hash Layer 1x64	Previous token	24.16	24.22
Hash Layer 1x64	Future token predictions (using Transformer Baseline)	25.02	25.09
Hash Layer 1x64	Future token (Oracle)	1.97	1.97
Hash Layer 5x16	Same hash per layer (balance assignment)	23.74	23.81
Hash Layer 5x16	Different Hash per layer	23.21	23.27

Stephen Roller, Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston. Hash Layers For Large Sparse Models. https://arxiv.org/pdf/2106.04426.pdf

Funded by the European Union

MoE Models

- Open-source (above the arrow).
- Private models (under the arrow).

MoE Design

What should we care when designing a MoE?

Network types	FFN, Attention
Fine-grained experts	64 experts/128 experts/…
Shared experts	Isolated experts
Activation Function	ReLU/GEGLU/SwiGLU
MoE frequency	Every two layer/Each layer/…
Training auxiliary loss	Auxiliary loss/Z-loss/…

Funded by the European Union

Fine-grained and Shared Experts

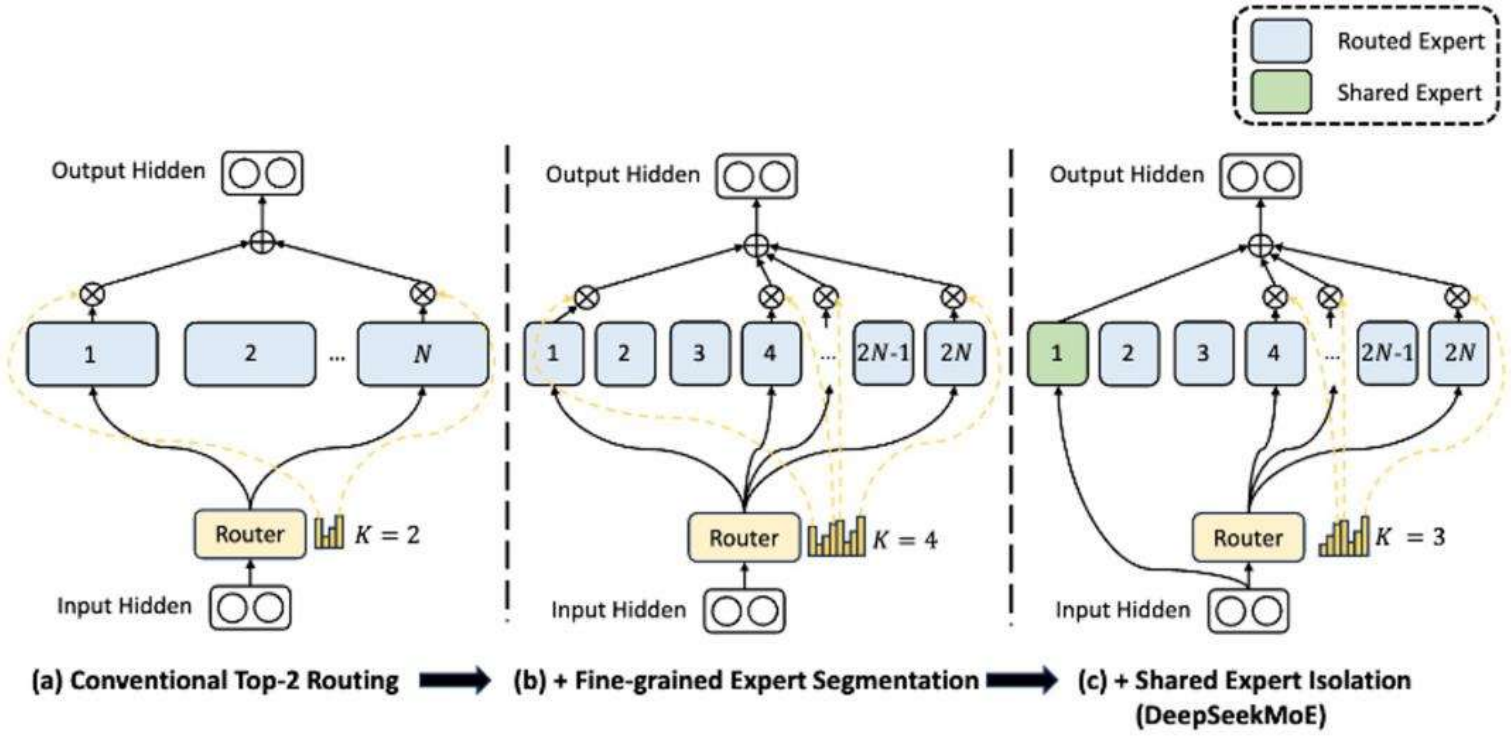


Figure 2 | Illustration of DeepSeekMoE. Subfigure (a) showcases an MoE layer with the conventional top-2 routing strategy. Subfigure (b) illustrates the fine-grained expert segmentation strategy. Subsequently, subfigure (c) demonstrates the integration of the shared expert isolation strategy, constituting the complete DeepSeekMoE architecture. It is noteworthy that across these three architectures, the number of expert parameters and computational costs remain constant.

Funded by the European Union

MoE Experts Design

Reference	Models	Expert Count (Activ./Total)	d_{model}	d_{ffn}	d_{expert}	#L	#H	d_{head}	Placement Frequency	Activation Function	Share Expert Count
GShard [86] (2020)	600B	2/2048	1024	8192	d_{ffn}	36	16	128	1/2	ReLU	0
	200B	2/2048	1024	8192	d_{ffn}	12	16	128	1/2	ReLU	0
	150B	2/512	1024	8192	d_{ffn}	36	16	128	1/2	ReLU	0
	37B	2/128	1024	8192	d_{ffn}	36	16	128	1/2	ReLU	0
Switch [49] (2021)	7B	1/128	768	2048	d_{ffn}	12	12	64	1/2	GEGLU	0
	26B	1/128	1024	2816	d_{ffn}	24	16	64	1/2	GEGLU	0
	395B	1/64	4096	10240	d_{ffn}	24	64	64	1/2	GEGLU	0
	1571B	1/2048	2080	6144	d_{ffn}	15	32	64	1	ReLU	0
GLaM [44] (2021)	0.1B/1.9B	2/64	768	3072	d_{ffn}	12	12	64	1/2	GEGLU	0
	1.7B/27B	2/64	2048	8192	d_{ffn}	24	16	128	1/2	GEGLU	0
	8B/143B	2/64	4096	16384	d_{ffn}	32	32	128	1/2	GEGLU	0
	64B/1.2T	2/64	8192	32768	d_{ffn}	64	128	128	1/2	GEGLU	0
DeepSpeed-MoE [121] (2022)	350M/13B	2/128	1024	$4d_{model}$	d_{ffn}	24	16	64	1/2	GeLU	0
	1.3B/52B	2/128	2048	$4d_{model}$	d_{ffn}	24	16	128	1/2	GeLU	0
	PR-350M/4B	2/32-2/64	1024	$4d_{model}$	d_{ffn}	24	16	64	1/2, 10L-32E, 2L-64E	GeLU	1
	PR-1.3B/31B	2/64-2/128	2048	$4d_{model}$	d_{ffn}	24	16	128	1/2, 10L-64E, 2L-128E	GeLU	1
ST-MoE [197] (2022)	0.8B/4.1B	2/32	1024	2816	d_{ffn}	27	16	64	1/4, add extra FFN	GEGLU	0
	32B/269B	2/64	5120	20480	d_{ffn}	27	64	128	1/4, add extra FFN	GEGLU	0
Mixtral [74] (2023)	13B/47B	2/8	4096	14336	d_{ffn}	32	32	128	1	SwiGLU	0
	39B/141B	2/8	6144	16384	d_{ffn}	56	48	128	1	SwiGLU	0
LLAMA-MoE [149] (2023)	3.0B/6.7B	2/16	4096	11008	688	32	32	128	1	SwiGLU	0
	3.5B/6.7B	4/16	4096	11008	688	32	32	128	1	SwiGLU	0
	3.5B/6.7B	2/8	4096	11008	1376	32	32	128	1	SwiGLU	0
DeepSeekMoE [30] (2024)	0.24B/1.89B	8/64	1280	-	$\frac{1}{4}d_{ffn}$	9	10	128	1	SwiGLU	1
	2.8B/16.4B	8/66	2048	10944	1408	28	16	128	1, except 1st layer	SwiGLU	2
	22B/145B	16/132	4096	-	$\frac{1}{8}d_{ffn}$	62	32	128	1, except 1st layer	SwiGLU	4
OpenMoE [172] (2024)	339M/650M	2/16	768	3072	d_{ffn}	12	12	64	1/4	SwiGLU	1
	2.6B/8.7B	2/32	2048	8192	d_{ffn}	24	24	128	1/6	SwiGLU	1
	6.8B/34B	2/32	3072	12288	d_{ffn}	32	24	128	1/4	SwiGLU	1
Qwen1.5-MoE [151] (2024)	2.7B/14.3B	8/64	2048	5632	1408	24	16	128	1	SwiGLU	4
DBRX [34] (2024)	36B/132B	4/16	6144	10752	d_{ffn}	40	48	128	1	SwiGLU	0
Jamba [94] (2024)	12B/52B	2/16	4096	14336	d_{ffn}	32	32	128	1/2, 1:7 Attention:Mamba	SwiGLU	0
Skywork-MoE [154] (2024)	22B/146B	2/16	4608	12288	d_{ffn}	52	36	128	1	SwiGLU	0
Yuan 2.0-M32 [166] (2024)	3.7B/40B	2/32	2048	8192	d_{ffn}	24	16	256	1	SwiGLU	0

- Most recent models place MoE each layer.
- Some of recent models apply Shared experts.

Funded by the European Union

Auxiliary Loss

Training with different auxiliary loss:

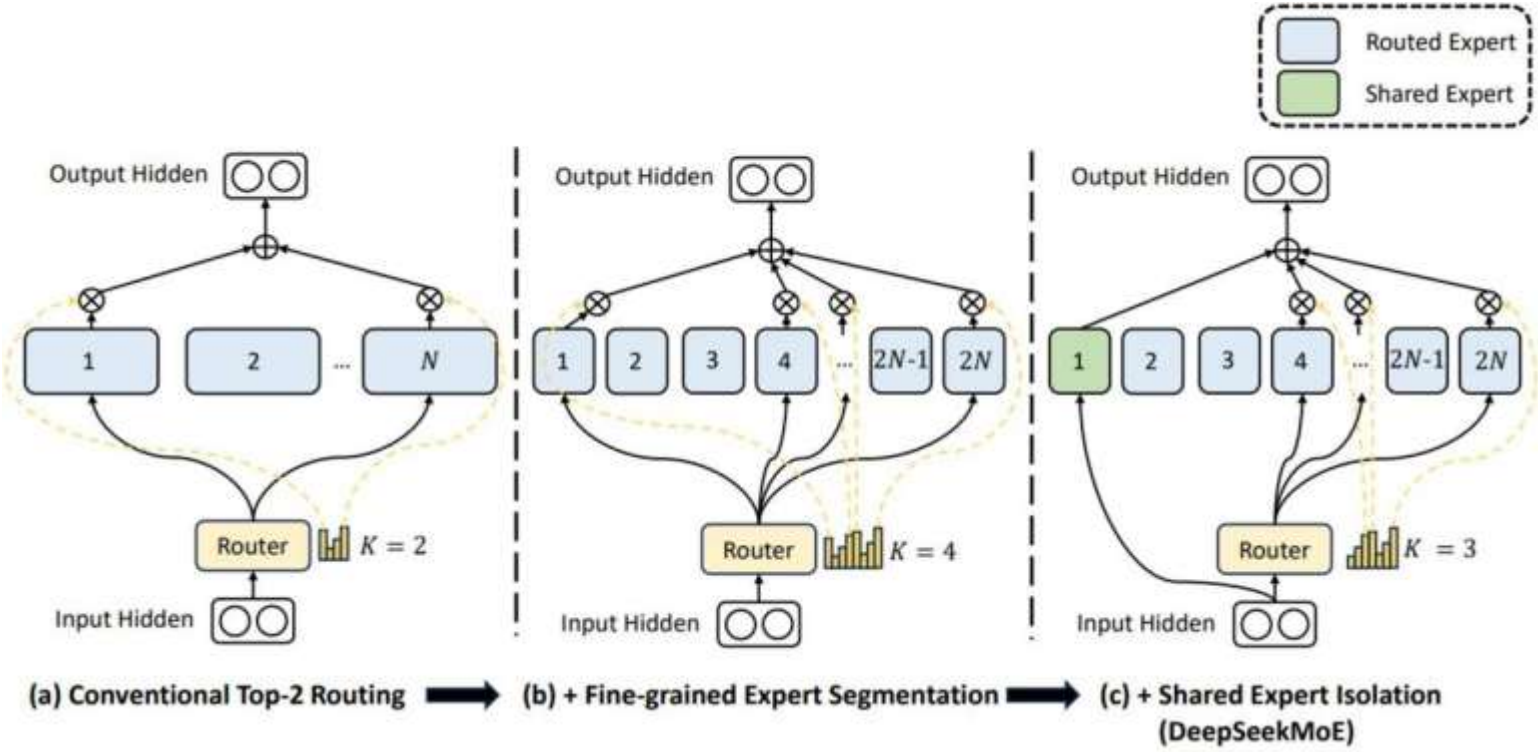
Reference	Auxiliary Loss	Coefficient
Shazeer et al.[135], V-MoE[128]	$L_{importance} + L_{load}$	$w_{importance} = 0.1, w_{load} = 0.1$
GShard[86], **Switch-T**[49], GLaM[44], Mixtral-8x7B[74], DBRX[34], Jamba[94], DeepSeekMoE[30], DeepSeek-V2[36], Skywork-MoE[154]	L_{aux}	$w_{aux} = 0.01$
ST-MoE[197], OpenMoE[172], MoA[182], JetMoE [139]	$L_{aux} + L_z$	$w_{aux} = 0.01, w_z = 0.001$
Mod-Squad[21], Moduleformer[140], DS-MoE[117]	L_{MI}	$w_{MI} = 0.001$

- Importance loss: encourages all experts to have equal importance
- Load loss: ensure balanced loads
- Auxiliary loss: mitigating load balance losses
- Z-loss: improving training stability by penalizing large logits
- MI-loss: mutual information (MI) between experts and tasks to build task-expert alignment

Funded by the European Union

Training MoE - Deepseek (example)

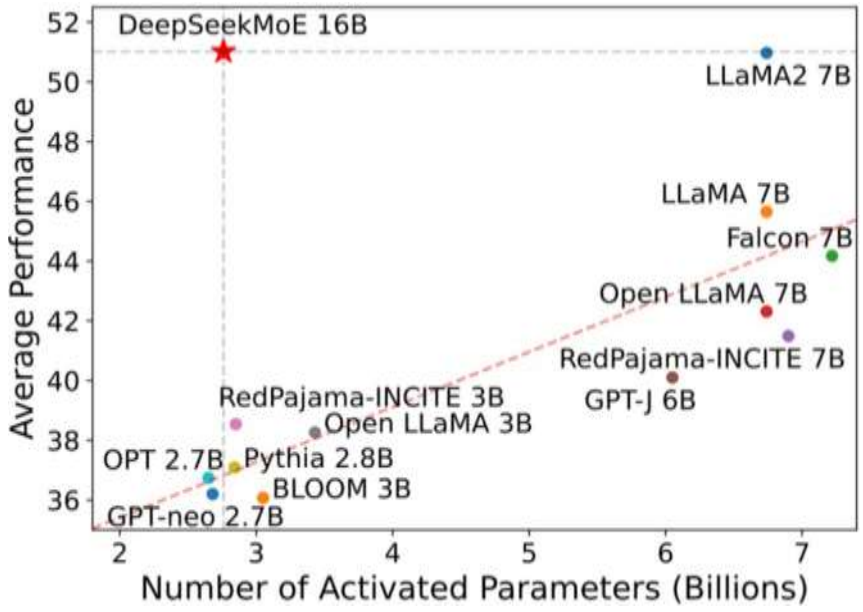
Deepseek-MoE 16B, total 16.4B parameters, 2.8B activated parameters.
Each MoE layer consists of 2 shared experts and 64 routed experts (select 6 experts).



- Most recent models place MoE each layer.
- Some of recent models apply Shared experts.

Funded by the European Union

Training MoE - Deepseek (example)



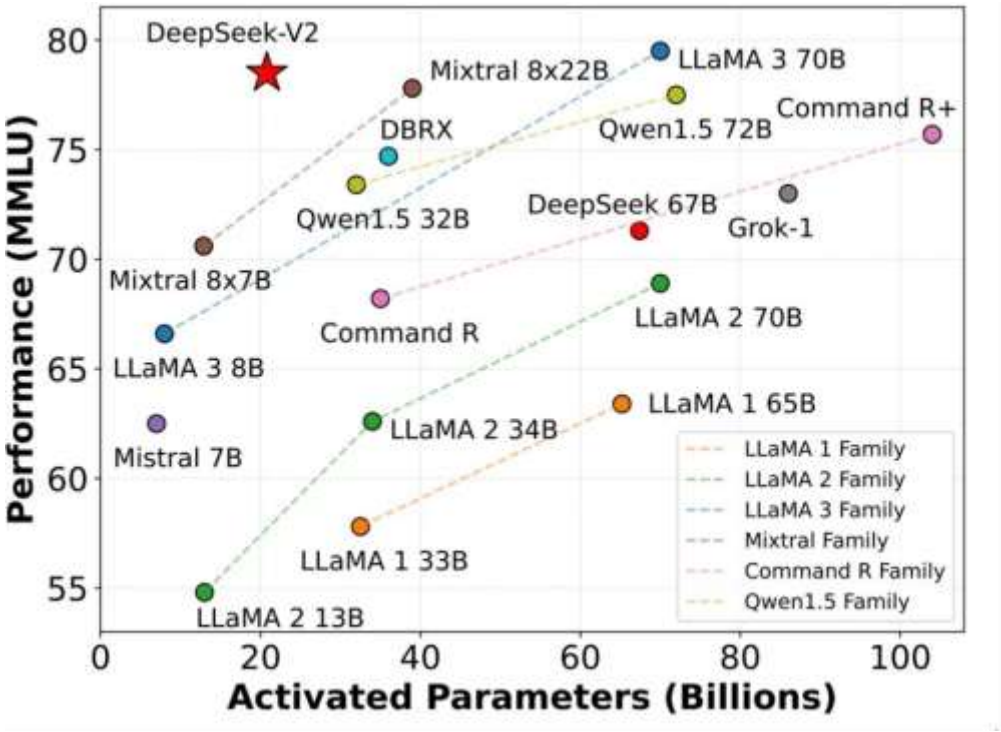
Metric	# Shot	DeepSeek 7B (Dense)	DeepSeekMoE 16B
# Total Params	N/A	6.9B	16.4B
# Activated Params	N/A	6.9B	2.8B
FLOPs per 4K Tokens	N/A	183.5T	74.4T
# Training Tokens	N/A	2T	2T
Pile (BPB)	N/A	0.75	**0.74**
HellaSwag (Acc.)	0-shot	75.4	**77.1**
PIQA (Acc.)	0-shot	79.2	**80.2**
ARC-easy (Acc.)	0-shot	**67.9**	**68.1**
ARC-challenge (Acc.)	0-shot	48.1	**49.8**
RACE-middle (Acc.)	5-shot	**63.2**	61.9
RACE-high (Acc.)	5-shot	**46.5**	**46.4**
DROP (EM)	1-shot	**34.9**	32.9
GSM8K (EM)	8-shot	17.4	**18.8**
MATH (EM)	4-shot	3.3	**4.3**
HumanEval (Pass@1)	0-shot	26.2	**26.8**
MBPP (Pass@1)	3-shot	**39.0**	**39.2**
TriviaQA (EM)	5-shot	59.7	**64.8**
NaturalQuestions (EM)	5-shot	22.2	**25.5**
MMLU (Acc.)	5-shot	**48.2**	45.0
WinoGrande (Acc.)	0-shot	**70.5**	**70.2**
CLUEWSC (EM)	5-shot	**73.1**	72.1
CEval (Acc.)	5-shot	**45.0**	40.6
CMMLU (Acc.)	5-shot	**47.2**	42.5
CHID (Acc.)	0-shot	**89.3**	**89.4**

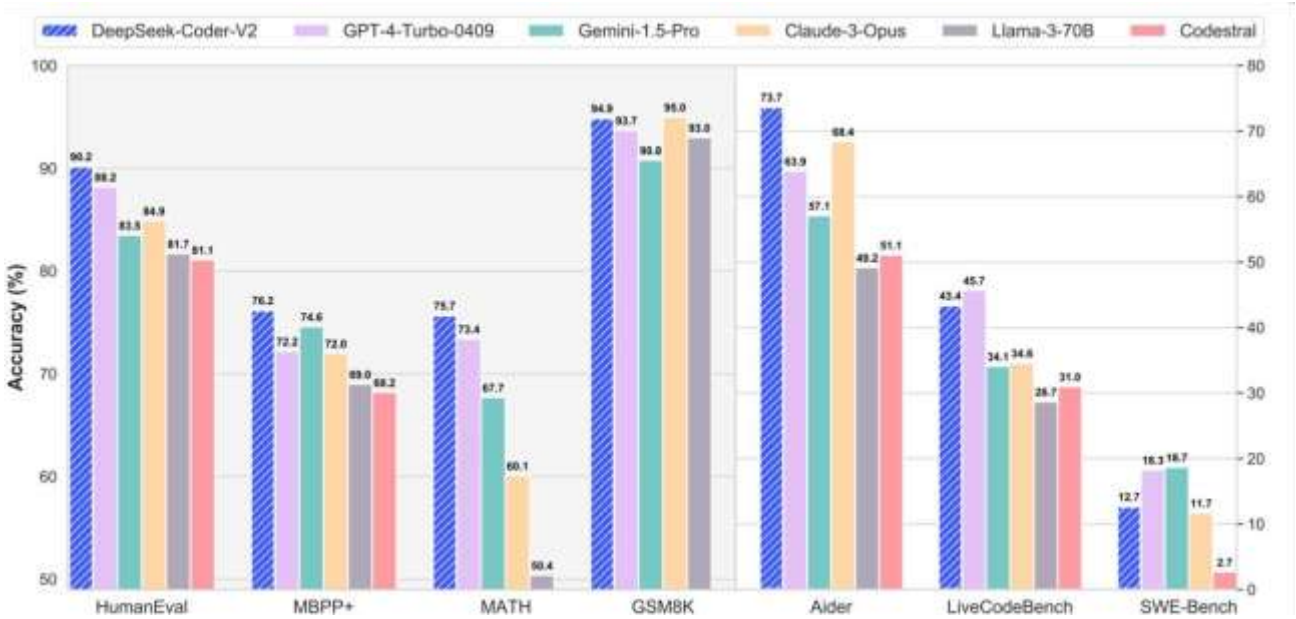
Funded by the European Union

Training MoE - Deepseek (example)

Deepseek-V2
236B total parameters, 21B are activated.
2 shared experts and 160 routed experts (6 select).

Deepseek-Coder-V2
Continue pretraining from an intermediate checkpoint of Deepseek-V2 (4.2T) and further train 6T. Total 10.2T tokens.





Sparse Upcycling - Qwen-MoE

Qwen1.5-MoE-A2.7B (Mar, 2024)
Upcycled from Qwen-1.8B, 14.3B parameters in total and 2.7B activated parameters.

- Fine-grained experts (total 64 experts)
- use shared (4 experts) and routing experts (60 experts, choose 4)

Model	MMLU	GSM8K	HumanEval	Multilingual	MT-Bench
Mistral-7B	64.1	47.5	27.4	40.0	7.60
Gemma-7B	64.6	50.9	32.3	-	-
Qwen1.5-7B	61.0	62.5	36.0	45.2	7.60
DeepSeekMoE 16B	45.0	18.8	26.8	-	6.93
Qwen1.5-MoE-A2.7B	62.5	61.5	34.2	40.8	7.17

Funded by the European Union

Scaling

Scaling laws in language modelling

- **Scaling laws** describe how model performance improves as we increase key factors such as model size and training data size.
- Empirical results suggest that performance improvements obey a power law: performance increases, but at a diminishing rate.
  - cf. Heap's law
- Scaling laws can help developers answer many practically relevant questions about resource allocation.

Funded by the European Union

Scaling Factors

- Many factors in configuring a scaled up pretraining run for Transformer Decoder + Autoregressive LM
  - Model size (parameter counts)
  - Pretraining dataset size
  - Pretraining compute (FLOPs or TPU/GPU hours)
  - Network shape (Parameters allocations)
  - Effective batch size
  - Learning rate & learning rate schedular
  - Context length

Funded by the European Union

Scaling Factors

Many factors in configuring a scaled up pretraining run for Transformer Decoder + Autoregressive LM

- Model size (parameter counts)
- Pretraining dataset size
- Pretraining compute (FLOPs or TPU/GPU hours)
- Network shape (Parameters allocations)
- Effective batch size
- Learning rate & learning rate schedular
- Context length

Main factors to study

Hyper-parameters with rule of thumb

1. Batch size determined by GPU memory
2. Try biggest LR before blowing up

Balance complexity and downstream needs

Funded by the European Union

Scaling Law Study: Setup

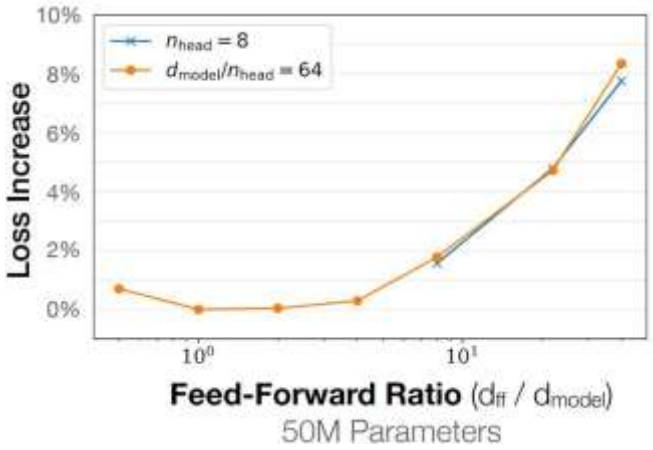
Empirically study the relationship between various factors to language model performances [4]

- Model: GPT-style, auto-regressive loss, maximum 1.5 billion non-embedding parameters
- Pretraining data: WebText2, harvest from Reddit out links, at max 23 billion tokens
- Metric: language modeling loss on testing data

Funded by the European Union

Scaling Law Study: Observations

Network shape (allocation of parameters at different parts) does not matter as much



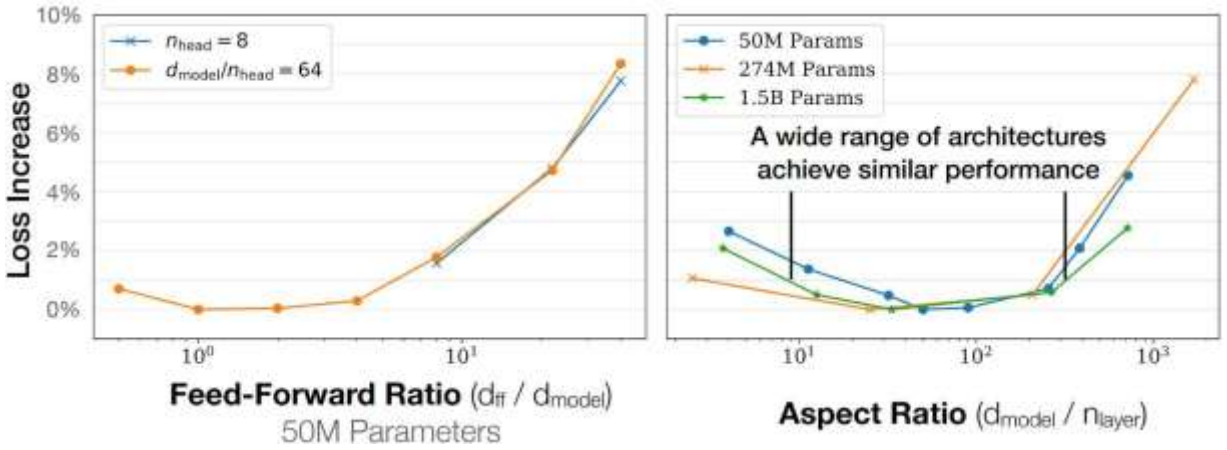
Language model loss changes with different network shape configurations [4].

- As long as the network shape is in a general sweet range, it does not impact performance much

Funded by the European Union

Scaling Law Study: Observations

Network shape (allocation of parameters at different parts) does not matter as much



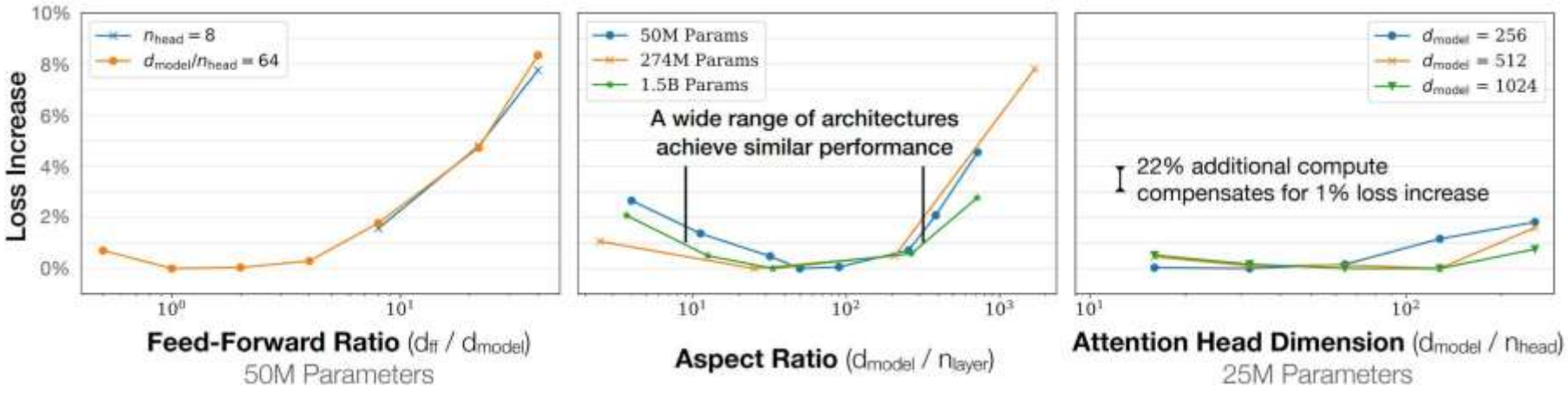
Language model loss changes with different network shape configurations [4].

- As long as the network shape is in a general sweet range, it does not impact performance much

Funded by the European Union

Scaling Law Study: Observations

Network shape (allocation of parameters at different parts) does not matter as much



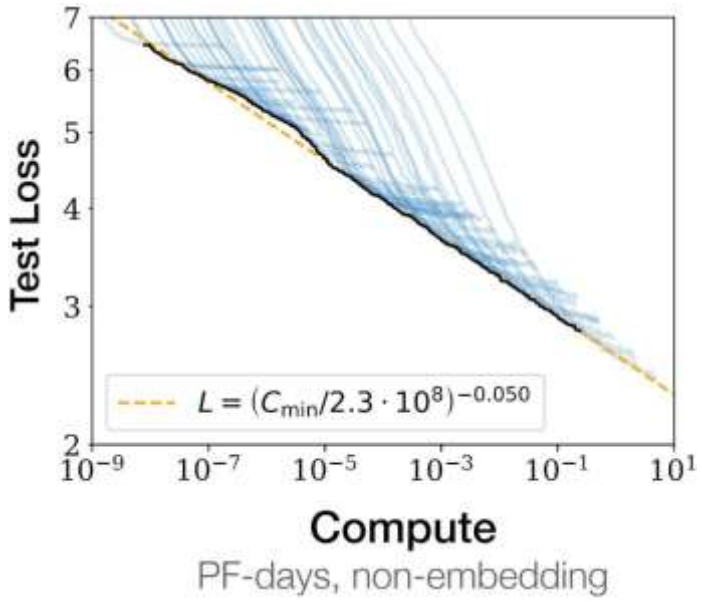
Language model loss changes with different network shape configurations [4].

- As long as the network shape is in a general sweet range, it does not impact performance much

Funded by the European Union

Scaling Law Study: Observations

A clear mapping from compute, data size, and parameter counts to testing loss

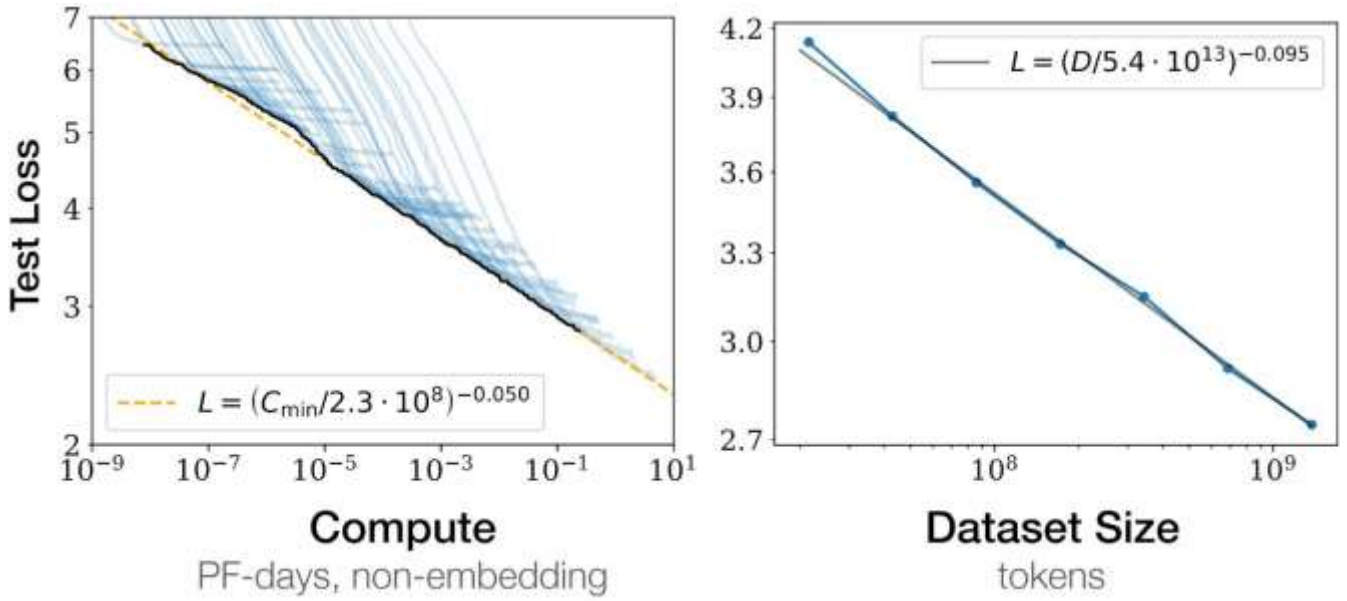


Mapping from compute (Peta-Flops days), data size, and model parameters to language modeling loss on testing data [4].

Funded by the European Union

Scaling Law Study: Observations

A clear mapping from compute, data size, and parameter counts to testing loss

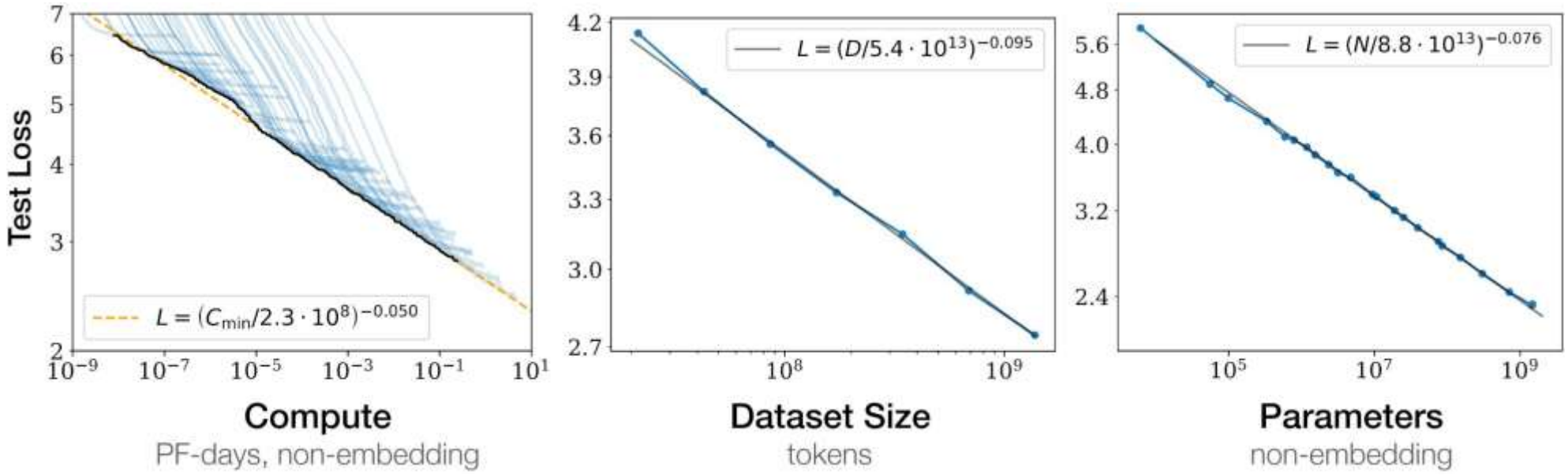


Mapping from compute (Peta-Flops days), data size, and model parameters to language modeling loss on testing data [4].

Funded by the European Union

Scaling Law Study: Observations

A clear mapping from compute, data size, and parameter counts to testing loss

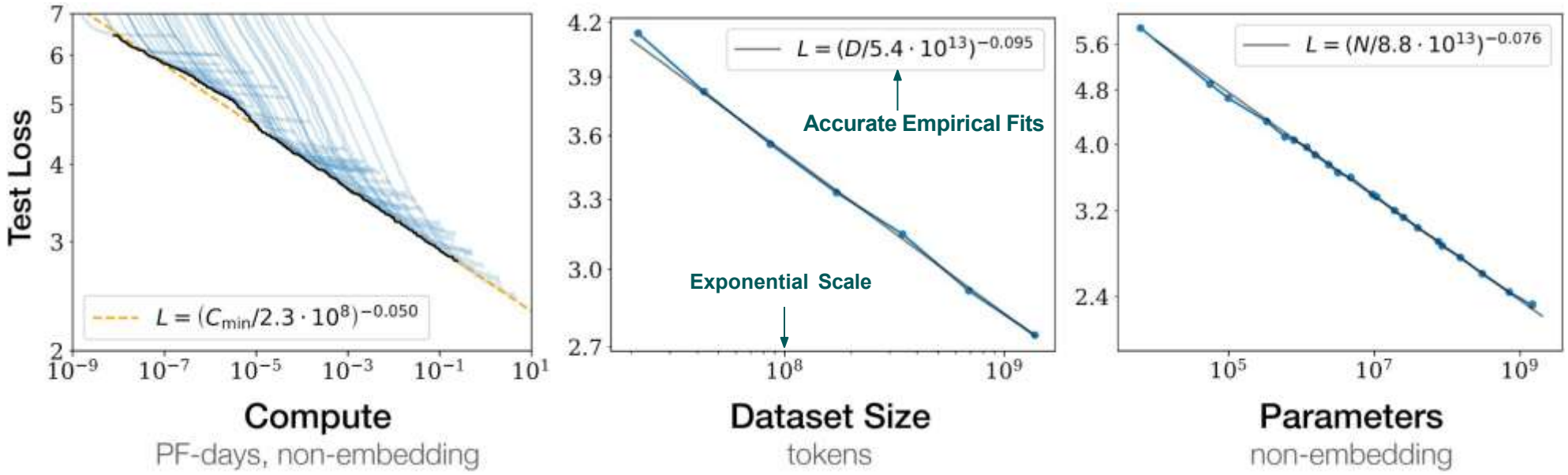


Mapping from compute (Peta-Flops days), data size, and model parameters to language modeling loss on testing data [4].

Funded by the European Union

Scaling Law Study: Observations

A clear mapping from compute, data size, and parameter counts to testing loss



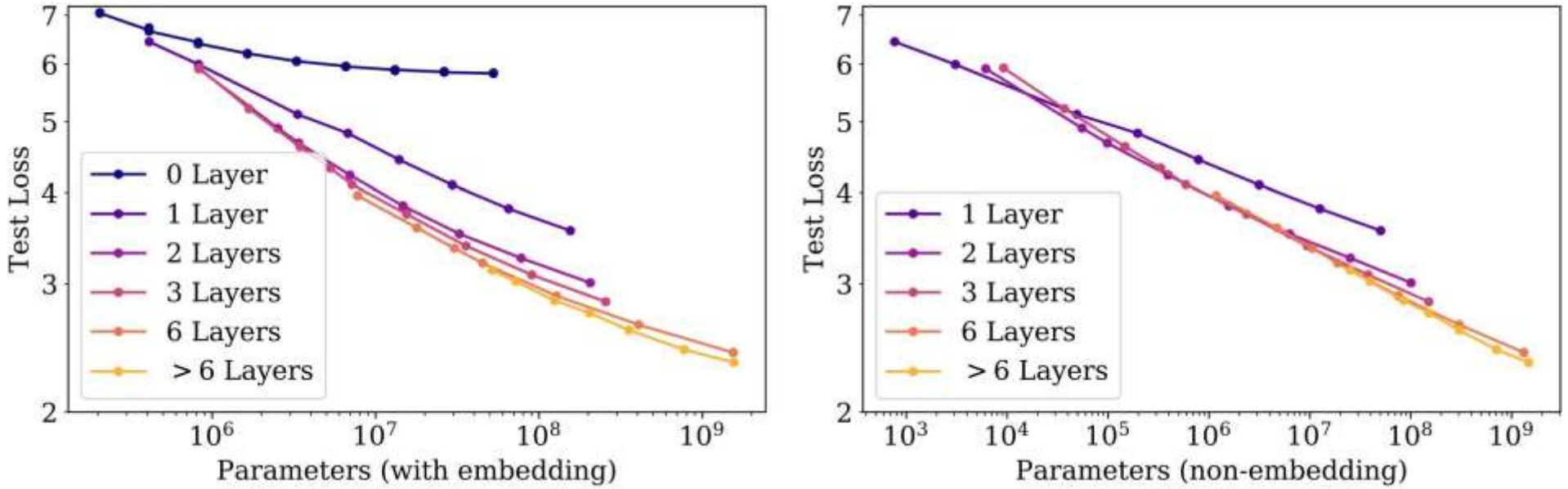
Mapping from compute (Peta-Flops days), data size, and model parameters to language modeling loss on testing data [4].

- Linear increasement of language modeling accuracy requires exponential scaling
- Three factors need to scale jointly to reach target model performance improvements

Funded by the European Union

Scaling Law Study: Observations

Network parameters matter more than embedding parameters

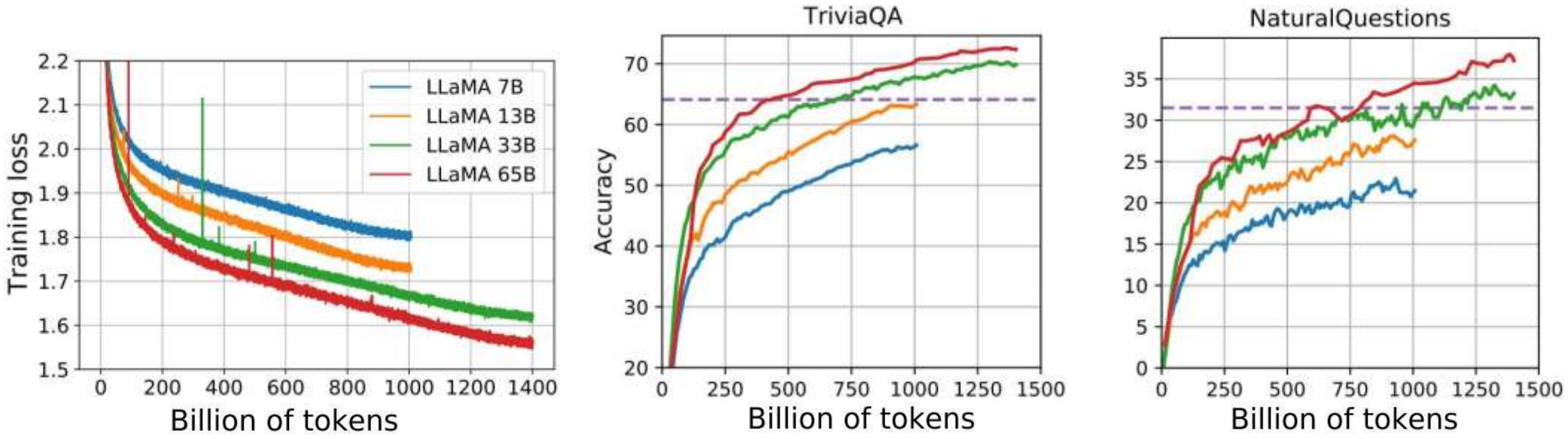


Scaling law with network parameter counts include (left) and exclude (right) embeddings [4].

Funded by the European Union

Scaling Law Study: Observations

Language modeling loss correlates well with downstream performances



Pretraining loss and downstream zero-shot accuracy during LLaMA pretraining steps [6]

Funded by the European Union

What Configurations to Scale Up

Goal: Given a computing budget and a candidate language model, select the optimal scaling up configurations

- E.g., One million H100 hours, pretrain the best LLaMA style LLM
- Configurations to choose: Model size (# of parameters) and pretraining data size (# of tokens)

A common question when scaling up

- Computing budget is the biggest constraint
- No room for exploration at target scale
- Only one scaled up pretraining run allowed, both budget-wise and time-wise.

Funded by the European Union

What Configurations to Scale Up

Goal: Given a computing budget and a candidate language model, select the optimal scaling up configurations

- E.g., One million H100 hours, pretrain the best LLaMA style LLM
- Configurations to choose: Model size (# of parameters) and pretraining data size (# of tokens)

A common question when scaling up

- Computing budget is the biggest constraint
- No room for exploration at target scale
- Only one scaled up pretraining run allowed, both budget-wise and time-wise.

Solution: Scaling law

- Use many experiments at small scale to establish the scaling law
- Use scaling Law to predict best configuration at target compute

Funded by the European Union

Scaling Configuration: Empirical Scaling Law

Empirical Approach #1: Fix model size and varying pretraining tokens [7]

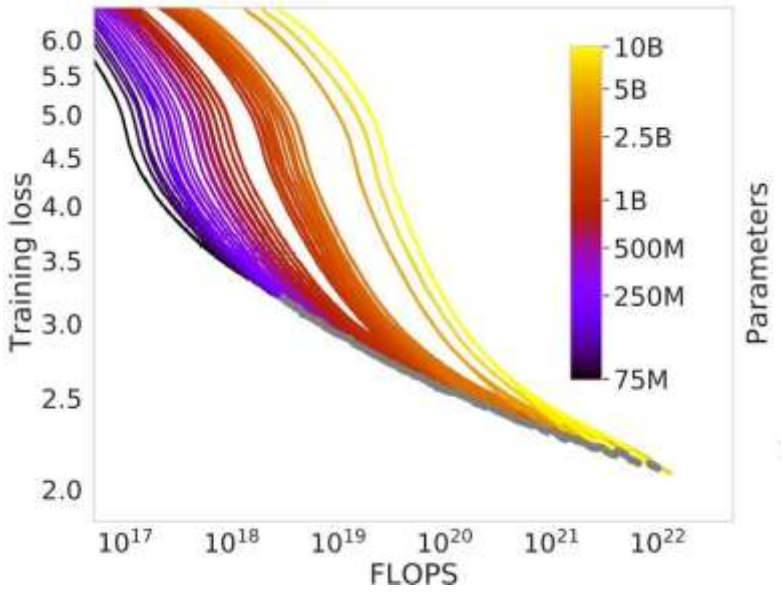
1. Pretrain different sized models to near converge and track loss
2. Record best (model size, data size) at each FLOP.
3. Estimate the scaling law

Funded by the European Union

Scaling Configuration: Empirical Scaling Law

Empirical Approach #1: Fix model size and varying pretraining tokens [7]

1. Pretrain different sized models to near converge and track loss
2. Record best (model size, data size) at each FLOP.
3. Estimate the scaling law



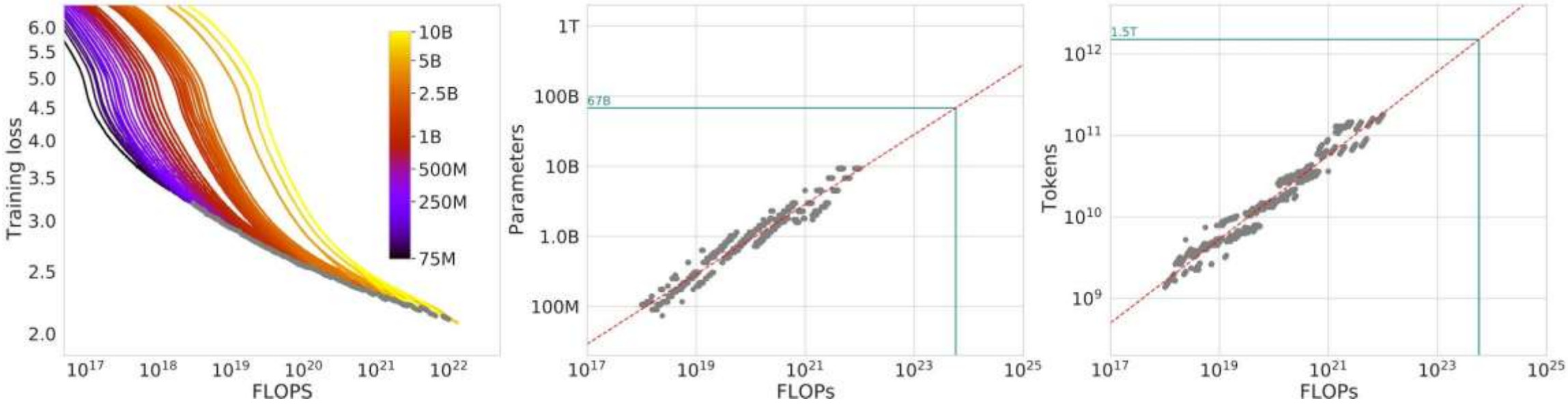
Pretraining loss of varying model (left), and the identified optimal parameters (mid) and tokens (right) at different FLOPS [6]

Funded by the European Union

Scaling Configuration: Empirical Scaling Law

Empirical Approach #1: Fix model size and varying pretraining tokens [7]

1. Pretrain different sized models to near converge and track loss
2. Record best (model size, data size) at each FLOP.
3. Estimate the scaling law



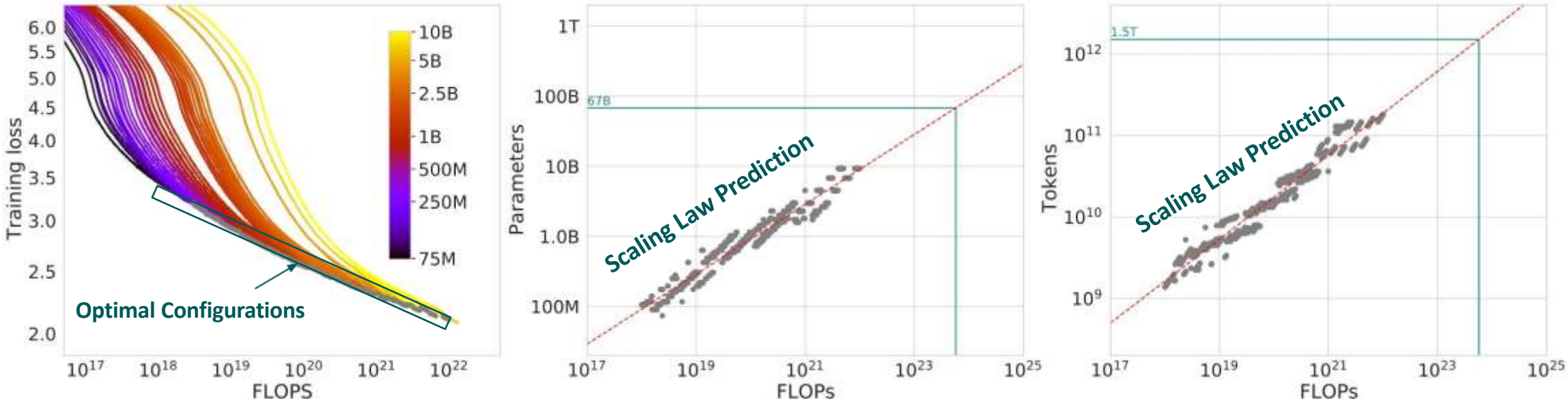
Pretraining loss of varying model (left), and the identified optimal parameters (mid) and tokens (right) at different FLOPS [6]

Funded by the European Union

Scaling Configuration: Empirical Scaling Law

Empirical Approach #1: Fix model size and varying pretraining tokens [7]

1. Pretrain different sized models to near converge and track loss
2. Record best (model size, data size) at each FLOP.
3. Estimate the scaling law



Pretraining loss of varying model (left), and the identified optimal parameters (mid) and tokens (right) at different FLOPS [6]

Funded by the European Union

Scaling Configuration: Empirical Scaling Law

Empirical Approach #2: Fix total FLOPs, pretrain different sized models [7]

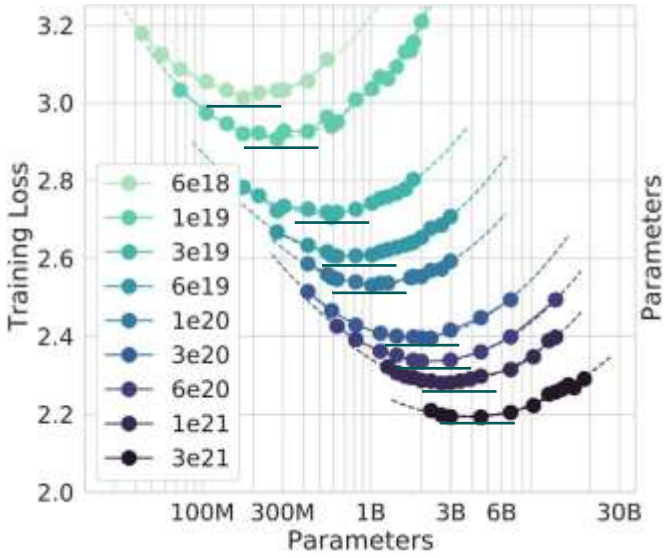
1. Pretrain to the # of tokens using total FLOPs and track final loss
2. Track best configurations and vary the total FLOPs and rerun #1
3. Estimate the scaling law

Funded by the European Union

Scaling Configuration: Empirical Scaling Law

Empirical Approach #2: Fix total FLOPs, pretrain different sized models [7]

1. Pretrain to the # of tokens using total FLOPs and track final loss
2. Track best configurations and vary the total FLOPs and rerun #1
3. Estimate the scaling law



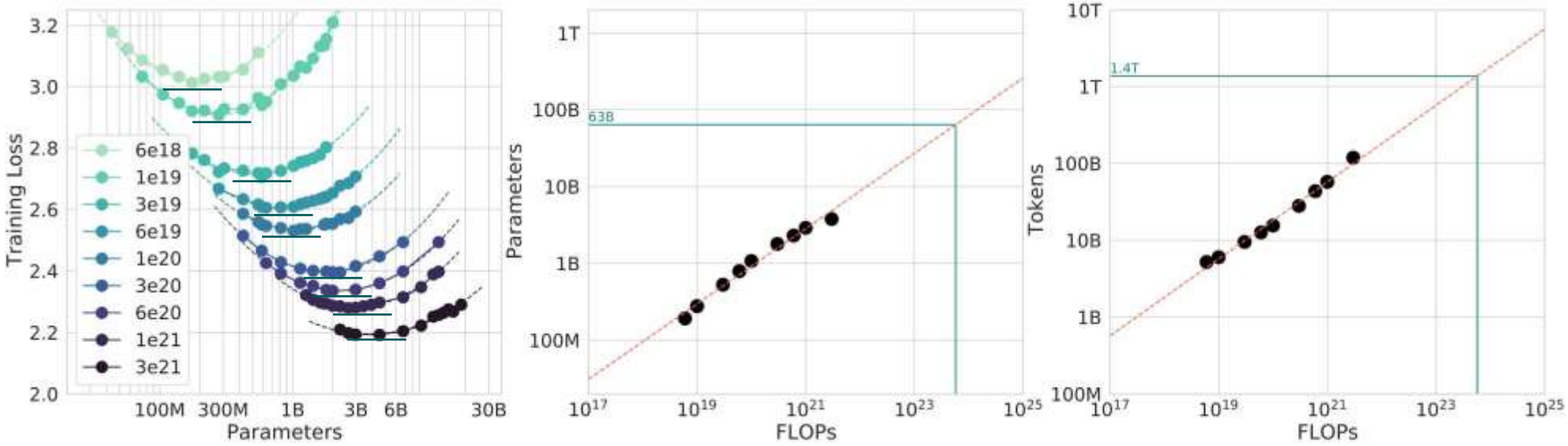
Pretraining loss of varying model sizes at varying FLOPs (left), and the identified optimal parameters (mid) and tokens (right) [6]

Funded by the European Union

Scaling Configuration: Empirical Scaling Law

Empirical Approach #2: Fix total FLOPs, pretrain different sized models [7]

1. Pretrain to the # of tokens using total FLOPs and track final loss
2. Track best configurations and vary the total FLOPs and rerun #1
3. Estimate the scaling law



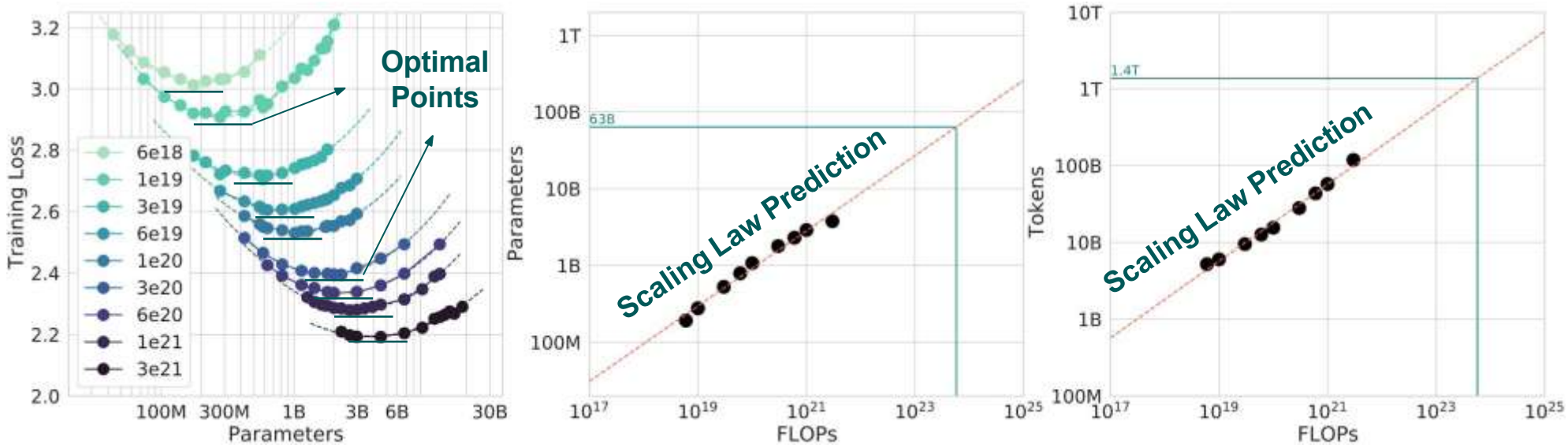
Pretraining loss of varying model sizes at varying FLOPs (left), and the identified optimal parameters (mid) and tokens (right) [6]

Funded by the European Union

Scaling Configuration: Empirical Scaling Law

Empirical Approach #2: Fix total FLOPs, pretrain different sized models [7]

1. Pretrain to the # of tokens using total FLOPs and track final loss
2. Track best configurations and vary the total FLOPs and rerun #1
3. Estimate the scaling law

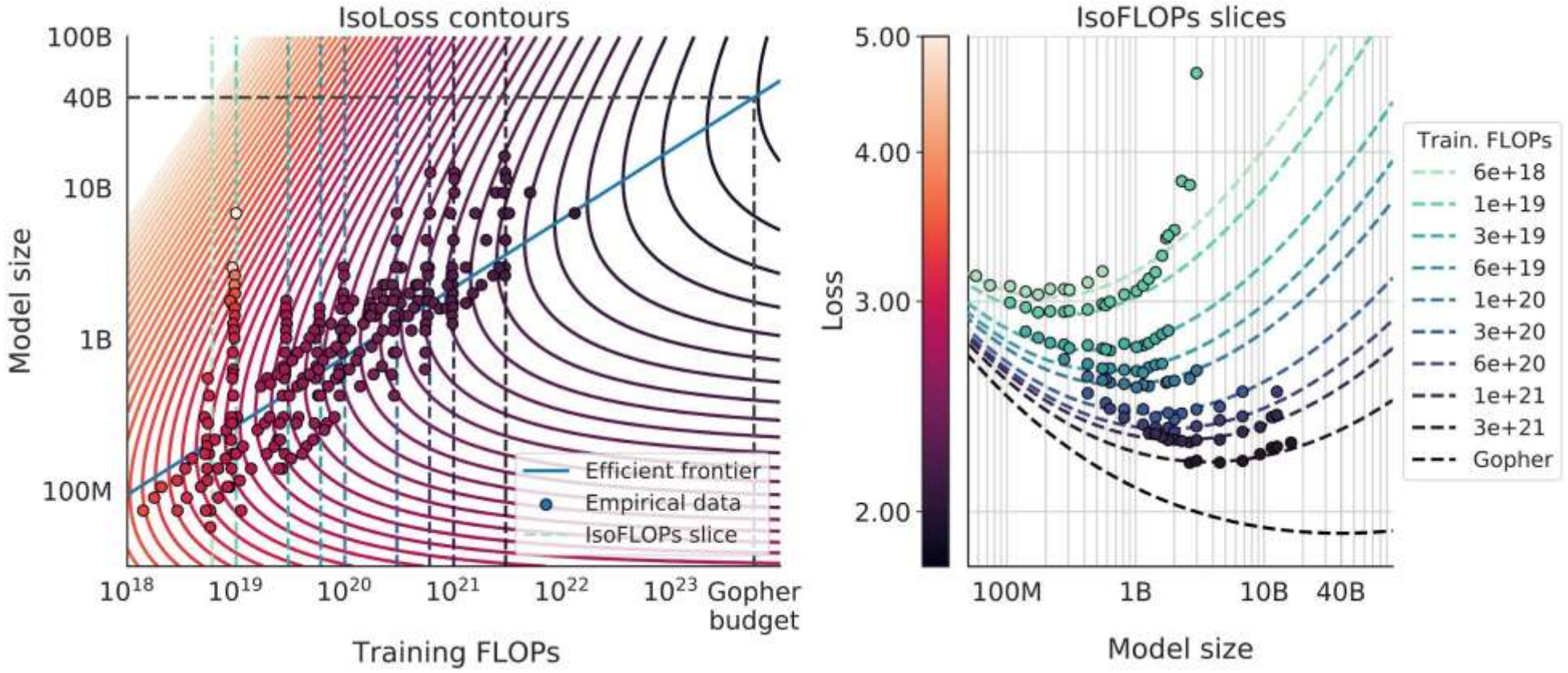


Pretraining loss of varying model sizes at varying FLOPs (left), and the identified optimal parameters (mid) and tokens (right) [6]

Funded by the European Union

Scaling Configuration: Empirical Scaling Law

Empirical Approach #3: Using data points collected from previous two approaches and fix a parametric functions



Fitted parametric function of (model size, FLOPs)→Loss using data from approach one (left) and two (right) [6]

Funded by the European Union

Scaling Configuration: Estimated Optimal Configurations

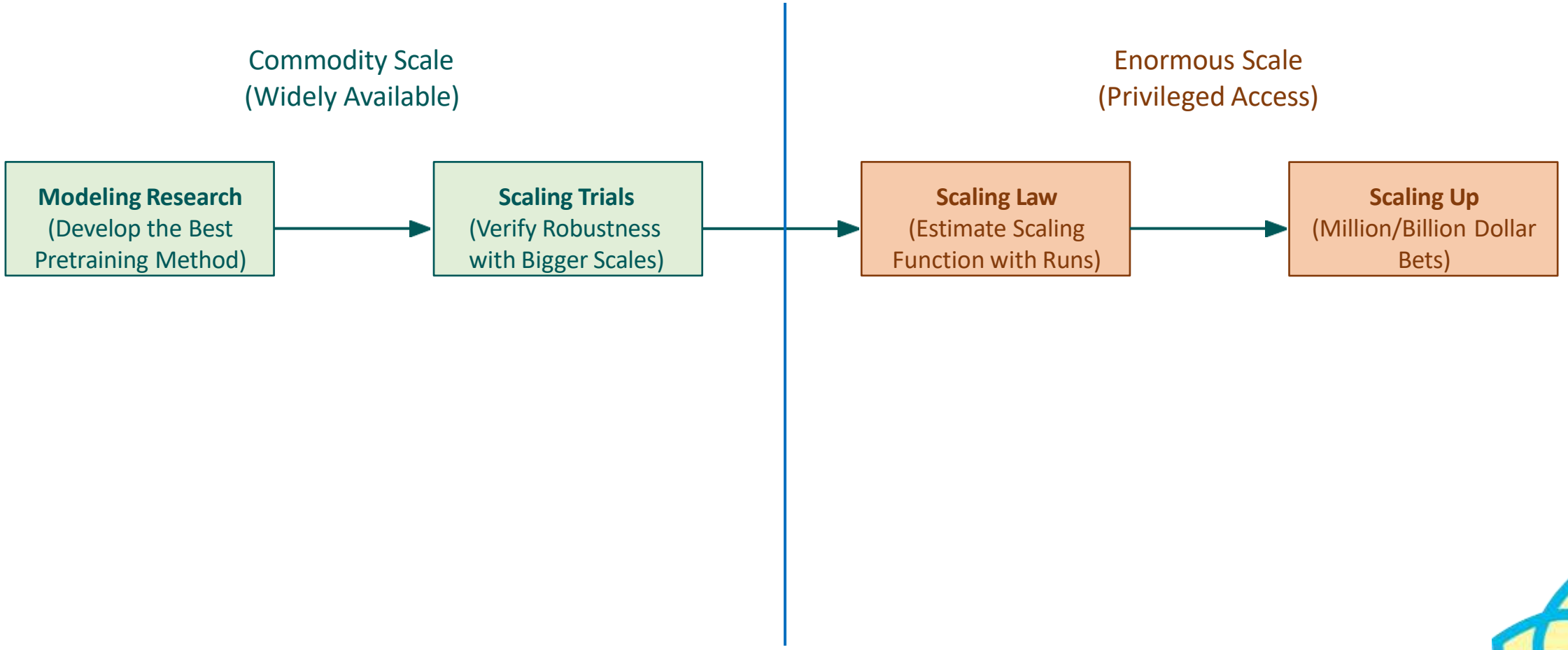
Parameters	FLOPs	FLOPs (in *Gopher* unit)	Tokens
400 Million	1.92e+19	1/29,968	8.0 Billion
1 Billion	1.21e+20	1/4,761	20.2 Billion
10 Billion	1.23e+22	1/46	205.1 Billion
67 Billion	5.76e+23	1	1.5 Trillion
175 Billion	3.85e+24	6.7	3.7 Trillion
280 Billion	9.90e+24	17.2	5.9 Trillion
520 Billion	3.43e+25	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

Examples of estimated scaling configurations at different model sizes [3].

Funded by the European Union

Scaling Up Pipeline

The current development pipeline of scaling up LLM pretraining, e.g., used by GPT-4, PaLM-2, and many more



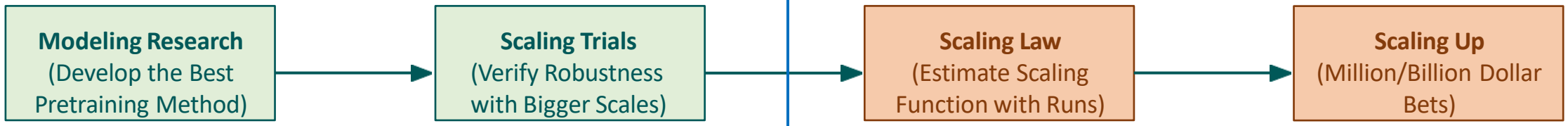
Funded by the European Union

Scaling Up Pipeline

The current development pipeline of scaling up LLM pretraining, e.g., used by GPT-4, PaLM-2, and many more

Commodity Scale
(Widely Available)

Enormous Scale
(Privileged Access)



Research

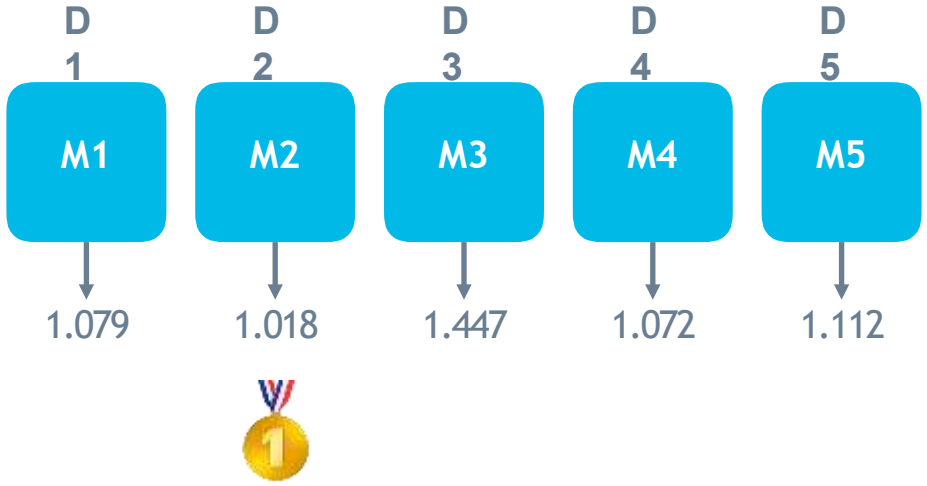
- Fast and many iterations
- Build new knowledge & understanding
- Can/Should take risks
- No guaranteed outcome
- Talent-heavy

Production?

- One scaling up run
- Build a better product: LLM
- Better not screw up
- Clear and short-term deliverables
- Resource-heavy

Funded by the European Union

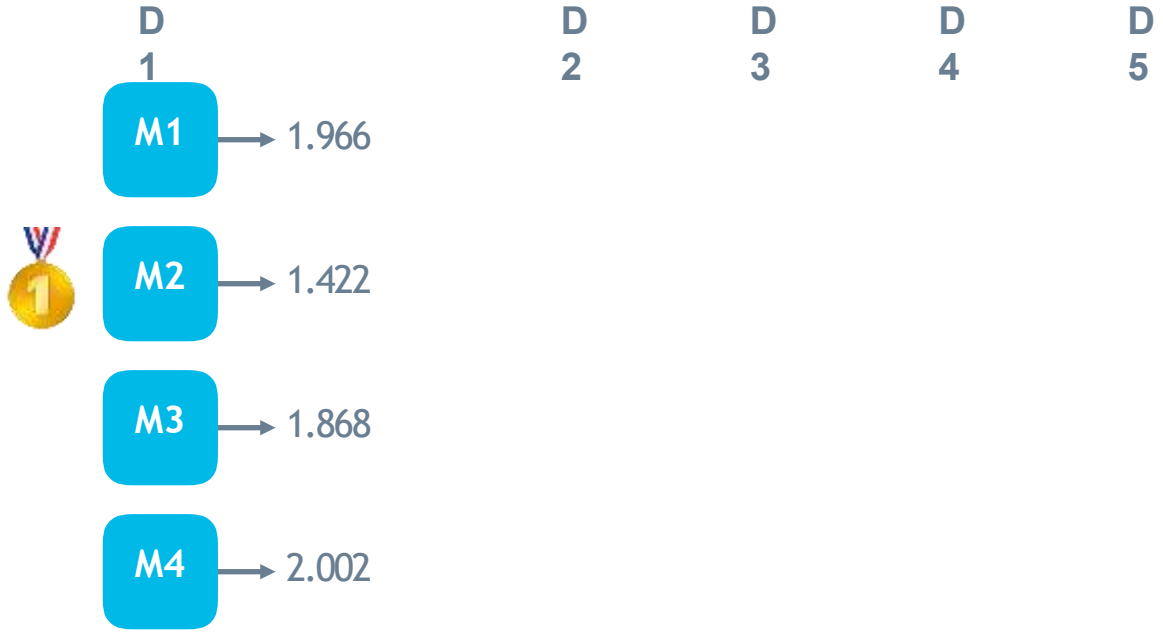
Putting scaling laws into practice



old paradigm: train a few models, select the best one

Funded by the European Union

Putting scaling laws into practice



new paradigm: train many small models, up-scale the best one

Funded by the European Union

Scaling Law: Summary

- Why Scaling Up
  - Predictable benefits in nearly all scenarios
- Which Language Model to Scale Up
  - Benefits of decoder models
- What Factors Matter in Scaling
  - Strong mapping from compute, model size, and pretraining data size to language model performances
- What Configurations to Scale Up
  - Establish scaling law with small scale explorations, scaling up based on scaling law predictions
- Capabilities Emerged from Scaling Up
  - Lots of unknowns and challenges!

Funded by the European Union

- LAIM LE4 VT2026:

Pre-training
Parallel Training
Scaling

www.ida.liu.se/~freheo8/llm

LINKÖPING UNIVERSITY