RLHF & “Small” LLMs

Instructor: Lara J. Martin (she/they)

TA: Omkar Kulkarni (he)

https://laramartin.net/NLP-class/

Slides modified from Yejin Choi, Bill Yuchen Lin, & Valentina Pyatkin

Learning Objectives

Describe what RLHF is and how it fits into alignment

Examining…

- Methods for shrinking pre-existing models
- Methods for mimicking pre-existing models with smaller models
- Methods for faster/smaller finetuning of pre-existing models
- Methods for training new models more efficiently

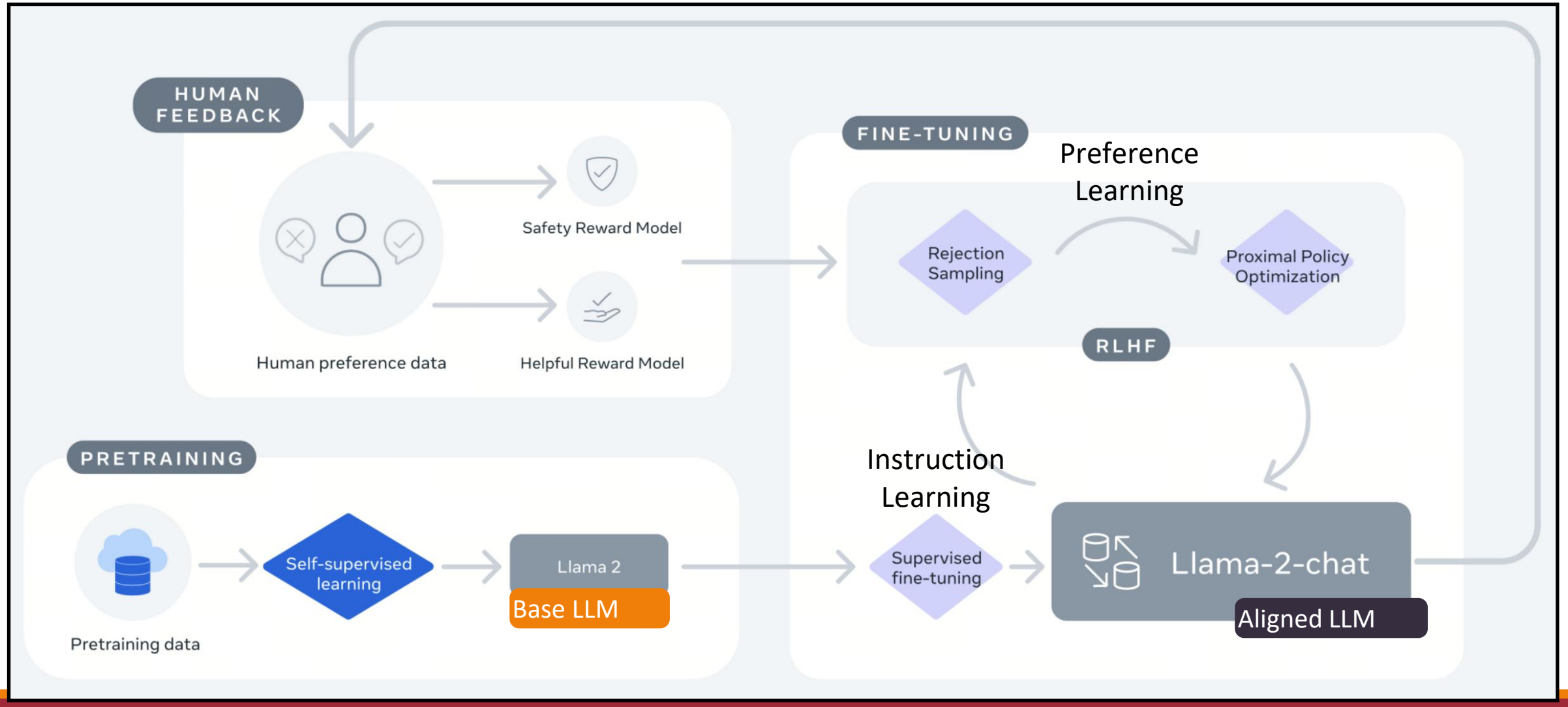
Finding where to implement these methods

Recognizing when to implement them

Review: What is Alignment?

AI concept for getting agents to match human values (value alignment), preferences, goals, etc.

Review: Llama-2's alignment



Review: Datasets for Instruction Learning

1. Synthetic Conversion
2. Human Annotation
3. Collected from ChatGPT/GPT-4
   - 3.1. Community Sharing
   - 3.2. Strategic Collecting

Review:
Examples of Evaluations for Alignment

Benchmarking Datasets

Human Annotation

GPTs as Judges

Open LLM Evaluators

Safety Evaluation

Review: Helping out Instruction Tuning

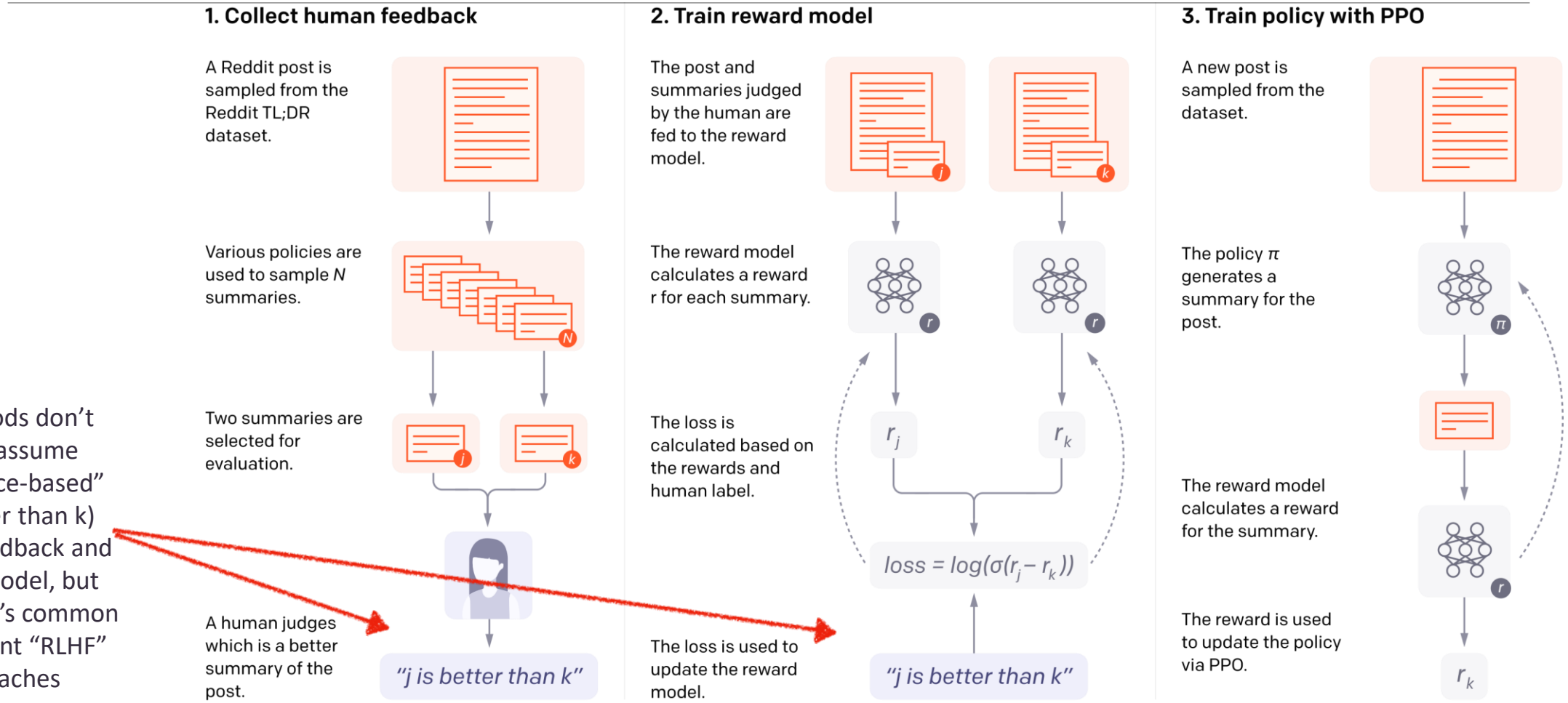
Why do we need RLHF?

What makes one output better than the other? -> hard to define

What types of LM errors should be weighted more?

LM objective != human preferences

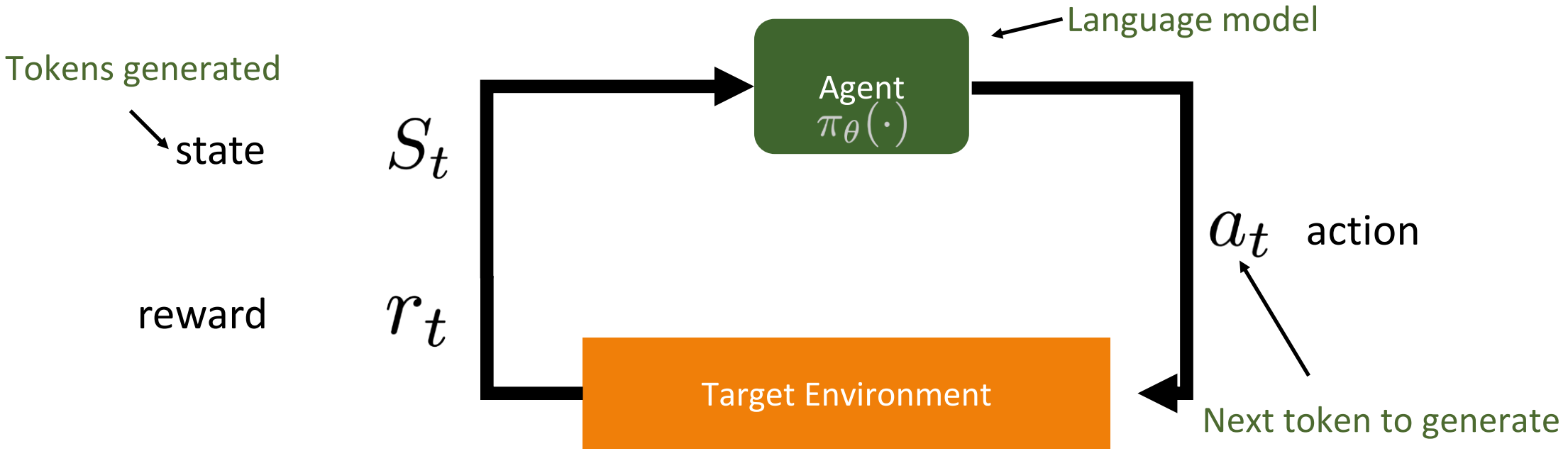
Review: "Learning to Summarize with Human Feedback"



RL methods don't always assume "preference-based" (j is better than k) human feedback and reward model, but that's what's common with current "RLHF" approaches

https://openai.com/research/learning-to-summarize-with-human-feedback

Review:
RL in the Context of Language Models…



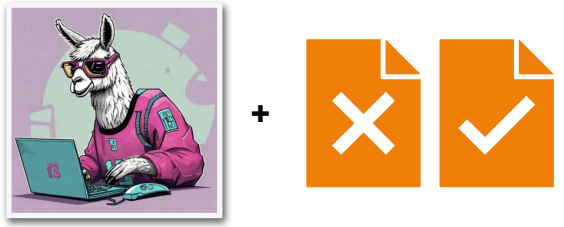
$a_t \sim \pi_\theta(S_t)$: policy

The general RLHF pipeline

1 Instruction-tuned Model

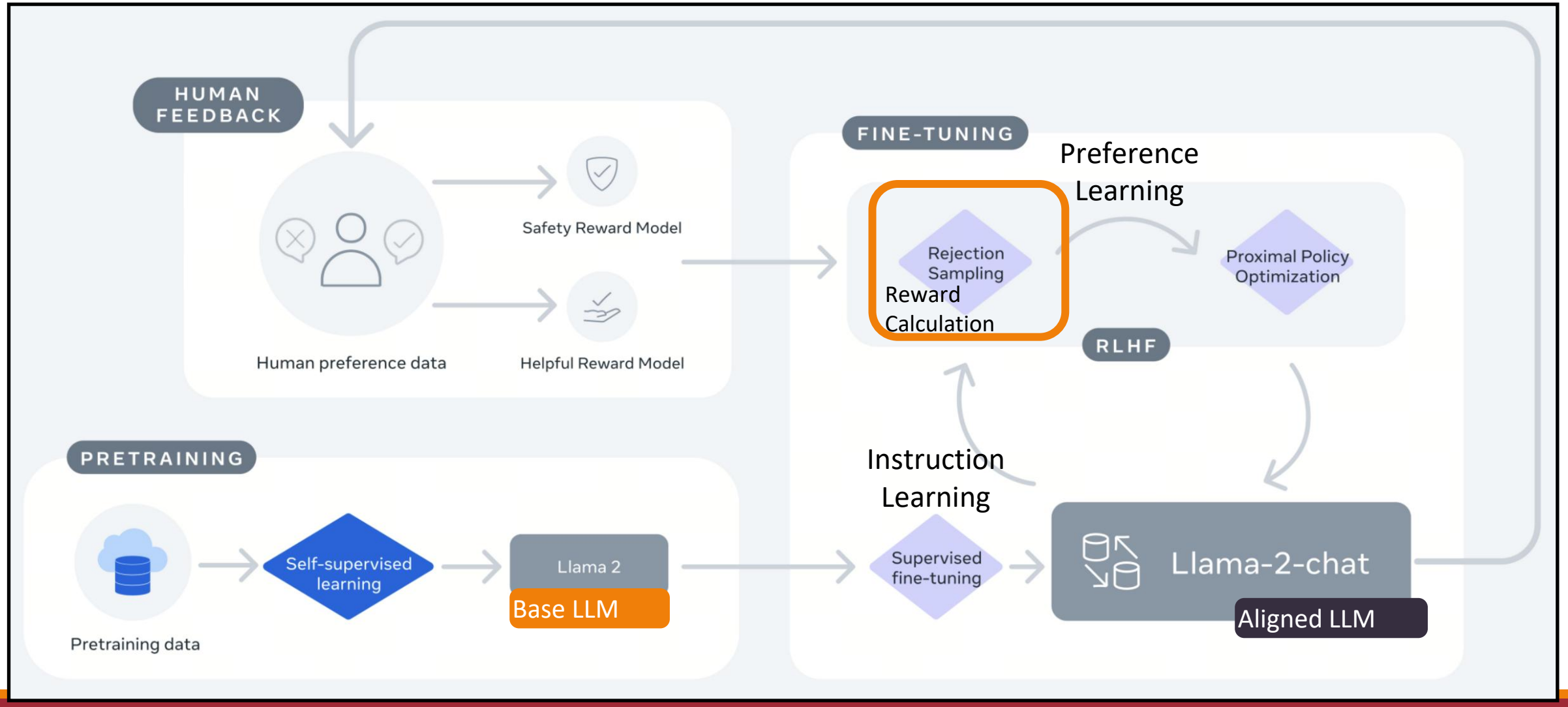
2 Collect Comparison Data

3 Train Reward Model on Comparison Data



4 Use RL to Optimize a Policy with the Reward Model

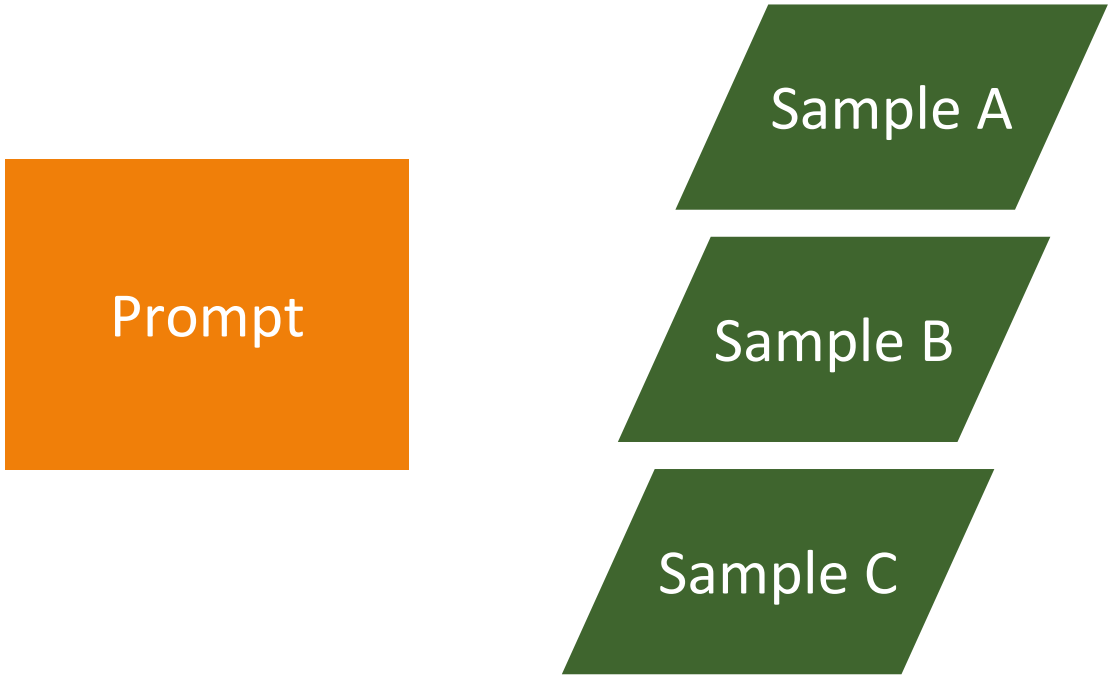
Example: Llama-2's alignment



Where does the reward come from? Start with Human Preferences

A set of sampled completions for a prompt

Triples



$$D = \{x^i, y_w^i, y_l^i\}$$

Prompt

Preferred Response

Dispreferred Response

👱‍♀️ Can you help me write a resignation letter to my current employer, while leaving on good terms and expressing gratitude for the opportunities provided?

Example: Annotation

🤖 Here are two responses from the chatbot. (Please scroll down on the content to see the entire response if it is too long)

Annotator needs to choose whether they prefer A or B.

$$D = \{x^i, y_w^i, y_l^i\}$$

Prompt; Preferred Response; Dispreferred Response

Calculating reward

Bradley-Terry Model

Reward for preferred response

Reward for dispreferred response

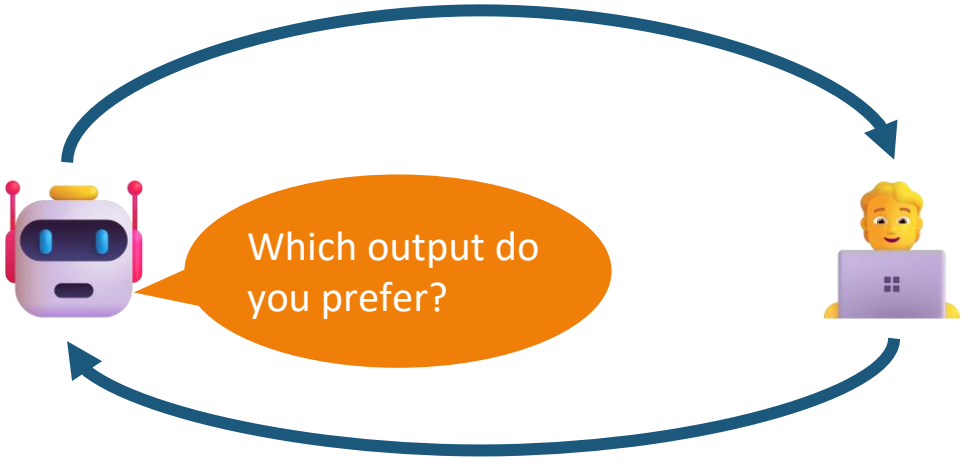
$$p(y_w > y_l|x) = \sigma(r(x, y_w) - r(x, y_l))$$

Logistic function; which is equivalent to using softmax:

$$\frac{1}{1 + e^{-x}}$$

$$p(y_w > y_l|x) = \frac{\exp(r(x, y_w))}{\exp(r(x, y_w)) + \exp(r(x, y_l))}$$

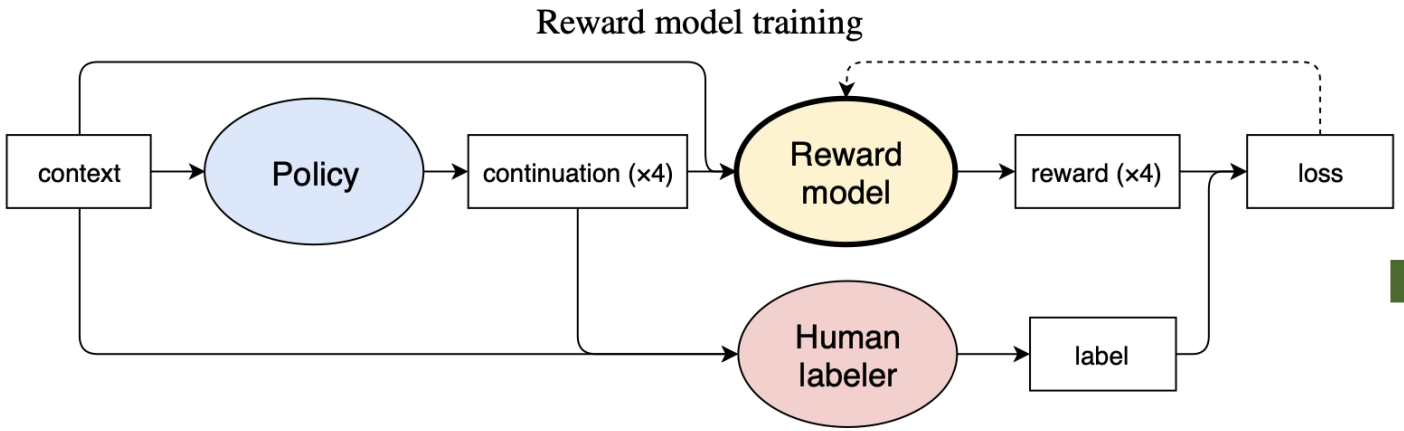
Making things cheaper…



Having a human in the loop is very expensive!

How can we more cheaply get the reward?

Reward Models



Instead: train a Reward Model (RM) on preference data to predict preferences!

Ziegler et al., 2019 "Fine-Tuning Language Models from Human Preferences"

Reward Modeling

Train on preference data.

Minimizing negative log likelihood.

Bradley-Terry Model

$$\mathcal{L}_R(\phi, D) = -\mathbb{E}_{(x,y_w,y_l)\sim D}[\log \sigma(r(x, y_w) - r(x, y_l))]$$

Train an LLM with an additional layer to minimize the neg. log likelihood

Fun Facts about Reward Models

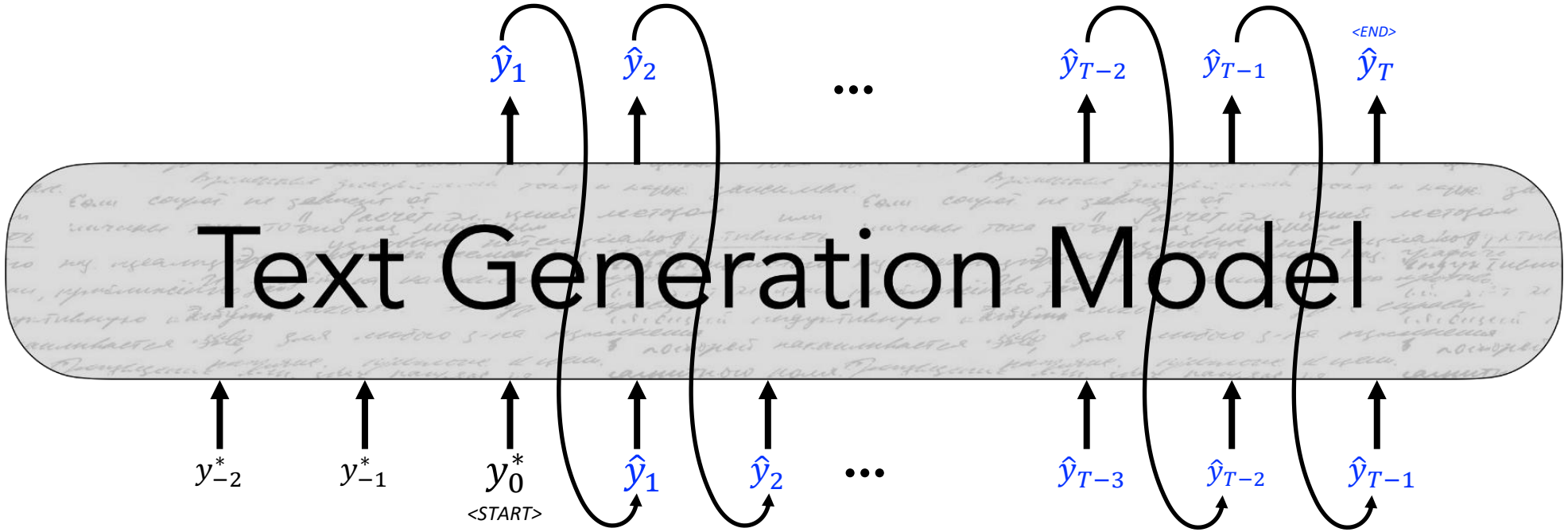
Trained for 1 epoch (to avoid overfitting)!

Evaluation often only has 65% - 75% agreement

Lambert et al., 2023

Review: REINFORCE

Sample a sequence from your model, score the sequence, and use the score to train the model.



REINFORCE

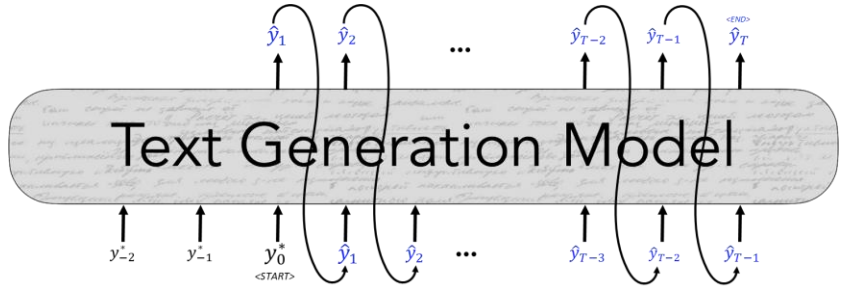
- Sample a sequence from your model, score the sequence, and use the score to train the model.

Next time, increase the probability of this sampled token in the same context.

$$L_{RL} = -\sum_{t=1}^{T} r(\hat{y}_t) \log P(\hat{y}_t | \{y^*\}; \{\hat{y}\}_{<t})$$

... but increase it more if I get a higher reward from the reward function.

- $r(\cdot)$: Your reward model
- y^* : Input sequence given to the model
- $\hat{y}$: The sequence sampled from the model given y^*



Updating the weights using REINFORCE

$$\theta_{t+1} := \theta_t + \alpha \frac{1}{m} \sum_{i=1}^{m} R(S_i) \nabla_{\theta_t} \log p_{\theta_t}(S_i)$$

Simplified Intuition: good actions are reinforced and bad actions are discouraged.

Williams, 1992

Updating the weights using REINFORCE

$$\theta_{t+1} := \theta_t + \alpha \frac{1}{m} \sum_{i=1}^{m} R(S_i) \nabla_{\theta_t} \log p_{\theta_t}(S_i)$$

If: Reward is high/positive

Then: maximize this

Simplified Intuition: good actions are reinforced and bad actions are discouraged

Williams, 1992

Updating the weights using REINFORCE

$$\theta_{t+1} := \theta_t + \alpha \frac{1}{m} \sum_{i=1}^{m} R(S_i) \nabla_{\theta_t} \log p_{\theta_t}(S_i)$$

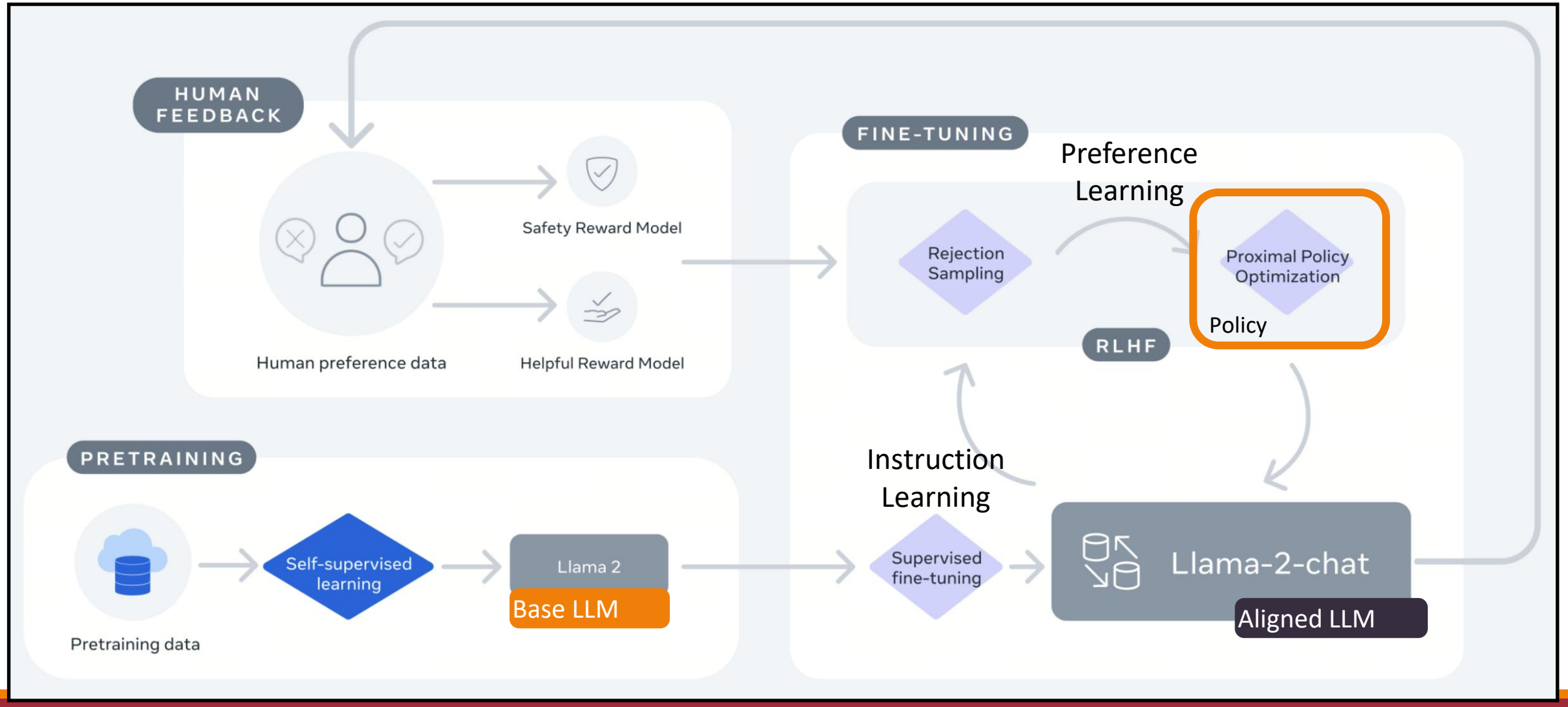
If: Reward is negative/low

Then: minimize this

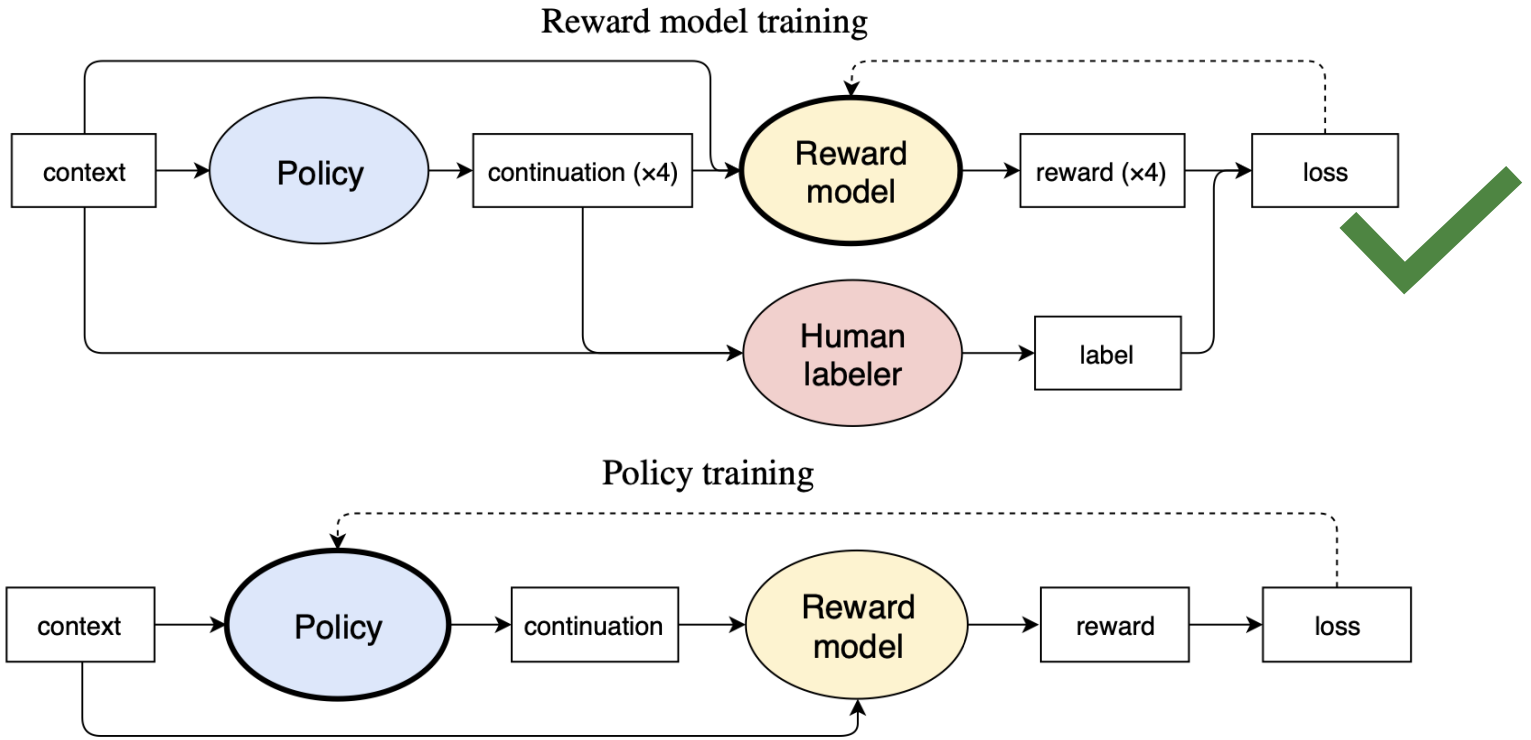
Simplified Intuition: good actions are reinforced and bad actions are discouraged

Williams, 1992

Example: Llama-2's alignment



Policy Training with Reward Model



Ziegler et al., 2019 "Fine-Tuning Language Models from Human Preferences"

Policy

We have: Reward Model

Next step: learn a **policy** to maximize the reward (minus KL regularization term) using the reward model

$$\max_{\pi_\theta} \mathbb{E}_{x\sim D, y\sim\pi_\theta(y|x)}[r_\phi(x,y)] - \beta\mathbb{D}_{KL}[\pi_\theta(y|x)||\pi_{ref}(y|x)]$$

Sampling from policy

Reward given prompt and sampled generation

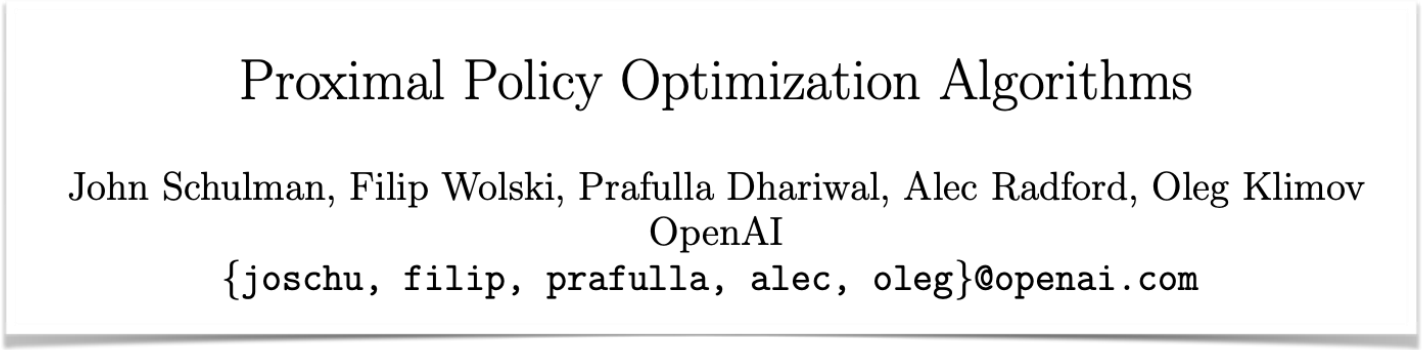
Should be high!

KL-divergence between original model's generation and the sampled generation

Should be low!

PPO

Proximal Policy Optimization



Proximal Policy Optimization Algorithms

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov
OpenAI
{joschu, filip, prafulla, alec, oleg}@openai.com

arxiv in July 2017

PPO: builds on Policy Gradient Methods

Gradient Estimator

$$\hat{g} = \hat{\mathbb{E}}_t[\nabla_\theta \log \pi_\theta(a_t|s_t)\hat{A}_t]$$

Expectation: empirical average over a finite batch of samples

Objective / Loss:

$$L^{PG}(\theta) = \hat{\mathbb{E}}_t[\log \pi_\theta(a_t|s_t)\hat{A}_t]$$

Often leads to (too) large policy updated

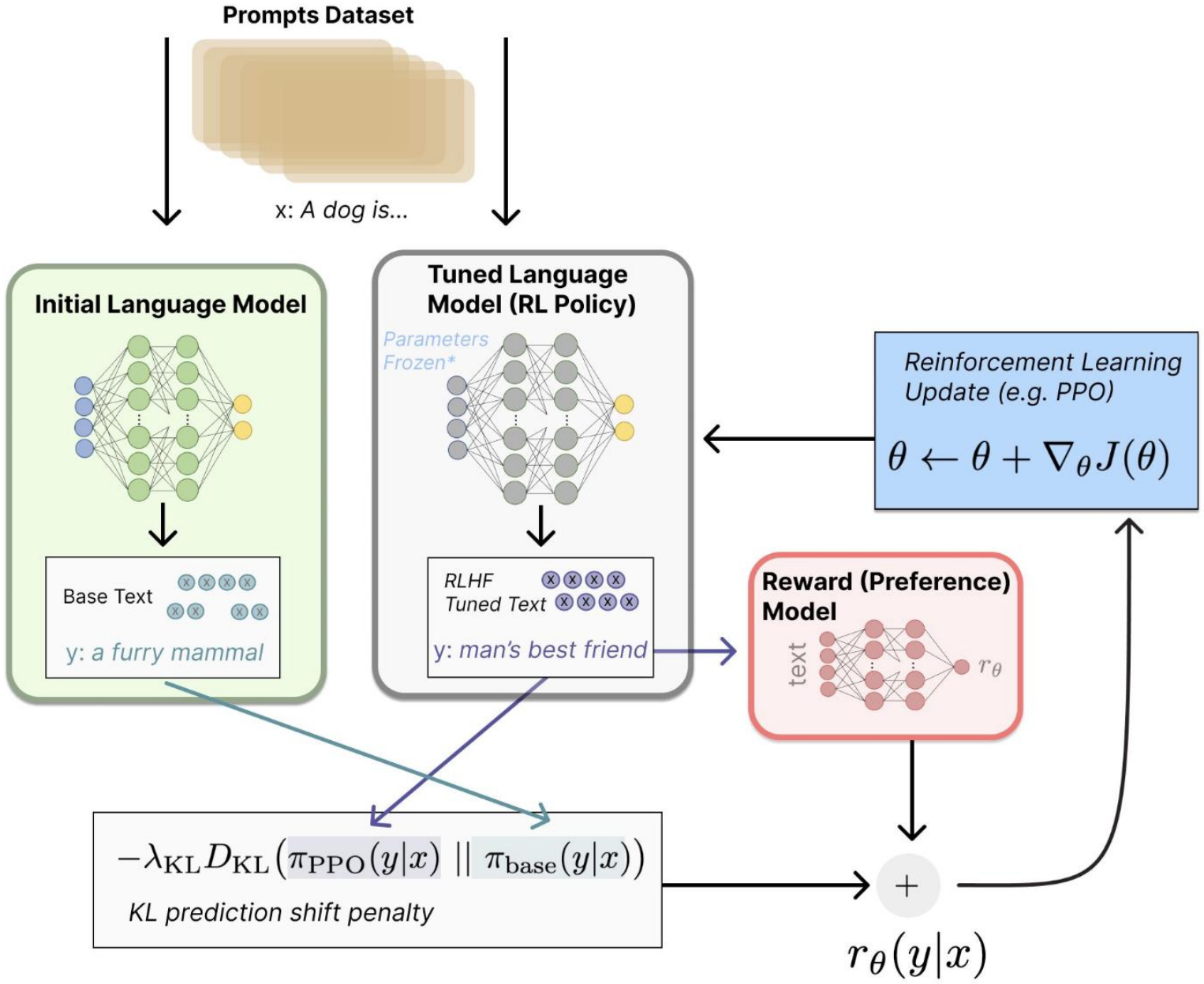
$\hat{A}_t$: estimator of the advantage function at timestep t (critic in actor-critic training)

π_θ : policy that we are trying to learn via PPO; this is initialized as a language model

$$\hat{A}_t = \hat{A}(s_t, a_t) = -V_\phi(t) + G_t = -V_\phi(t) + \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'},$$

Schulman, 2017

PPO



Lambert, 2023

Evaluating the Learned Policy

Win Rate: How often does my policy's output win against a reference model's output, given the same instruction?

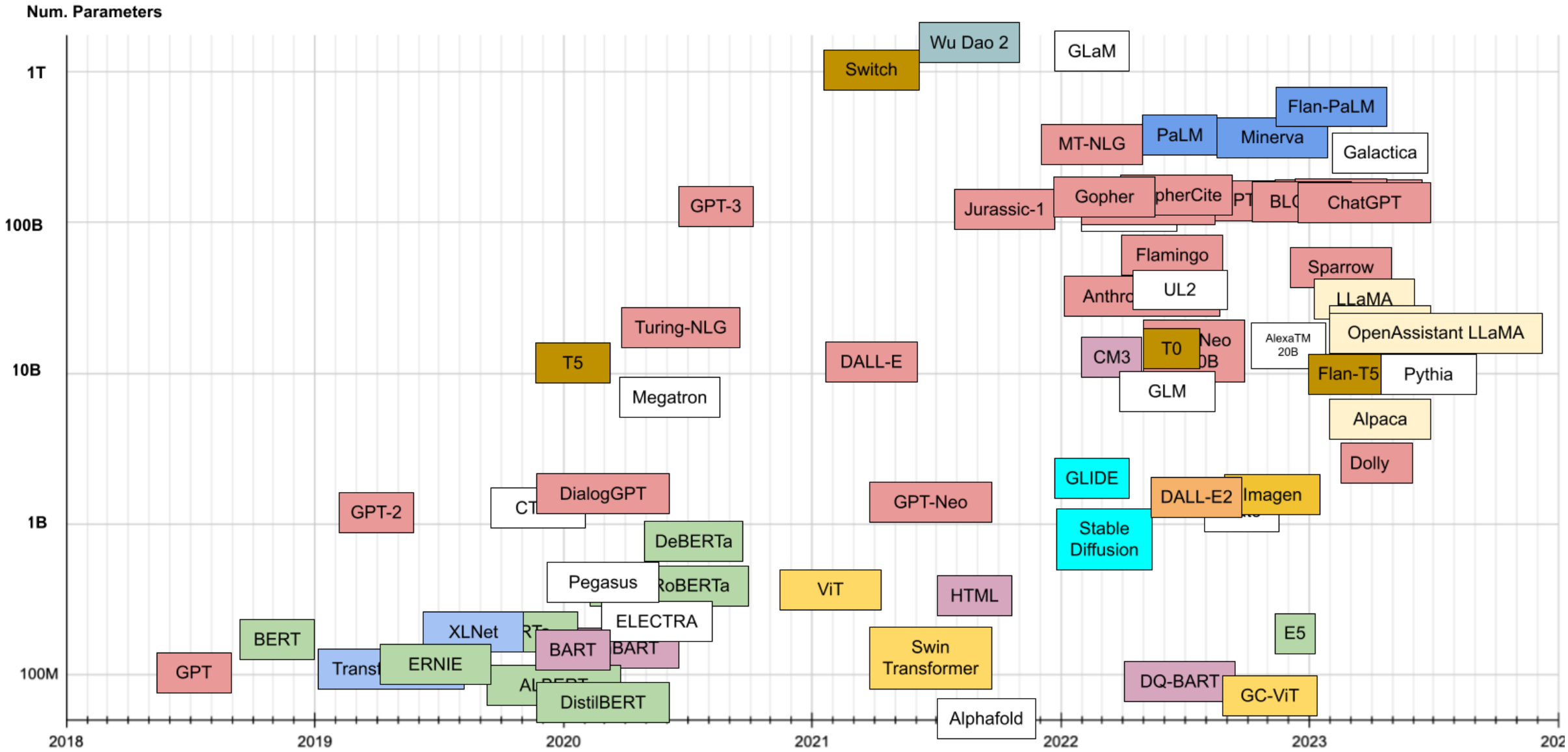
- Who compares the two outputs?
  - Humans
  - Simulated humans (and human variability!) using GPT-4 (**e.g.,** Alpacafarm eval)

Dubois et al., 2023

Small LLMs

Efficient LLMs

Methods for shrinking pre-existing models	Model Compression
Methods for mimicking pre-existing models with smaller models	Knowledge Distillation
Methods for faster/smaller finetuning of pre-existing models	PEFT
Methods for training new models more efficiently	Efficient Training e.g., Sparse Mixture of Experts



https://amatriain.net/blog/transformer-models-an-introduction-and-catalog-2d1e9039f376/

Model Compression

Pruning

Remove parts of the model

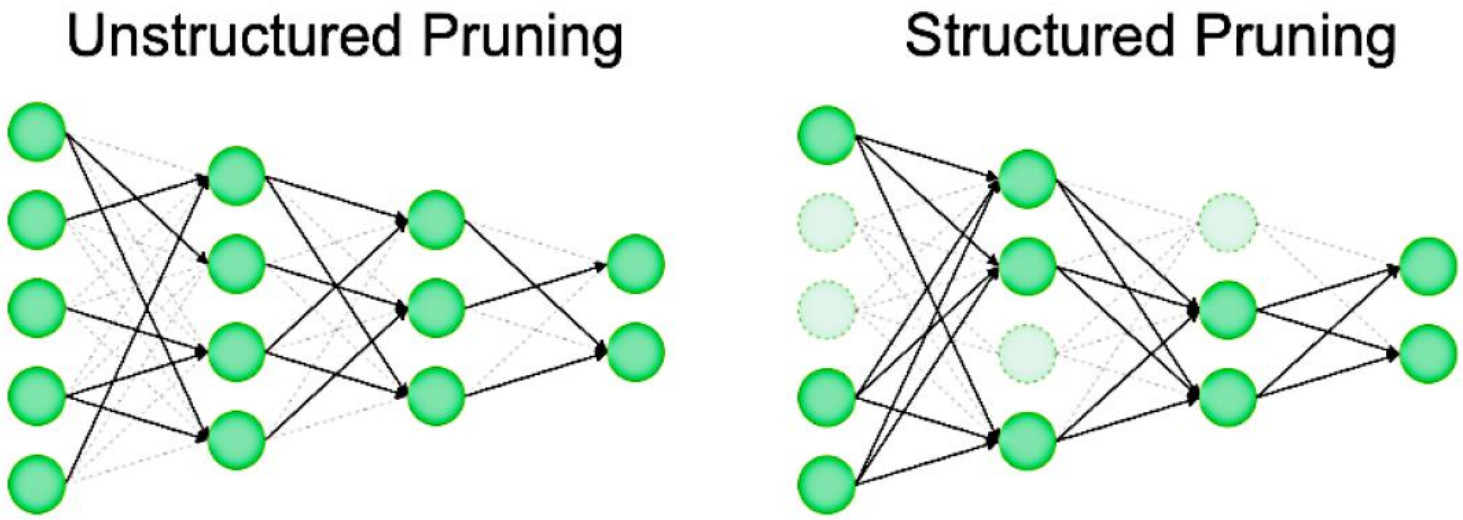
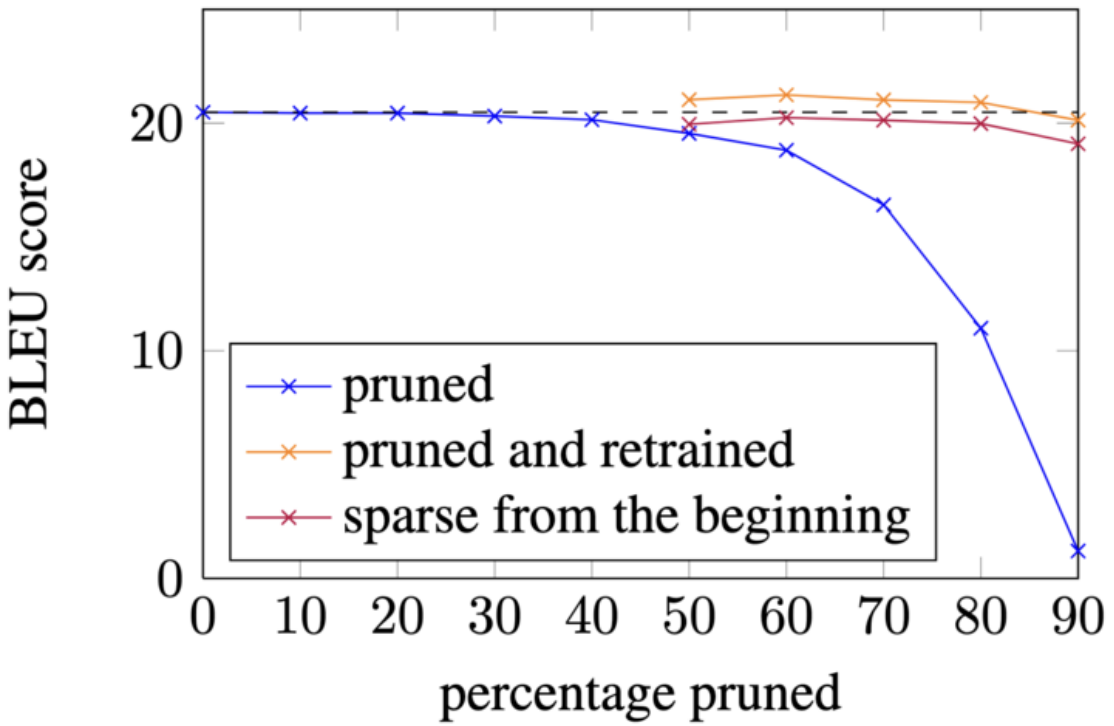


Image credits: neuralmagic.com

Implementation Tutorial: https://pytorch.org/tutorials/intermediate/pruning_tutorial.html

Magnitude Pruning



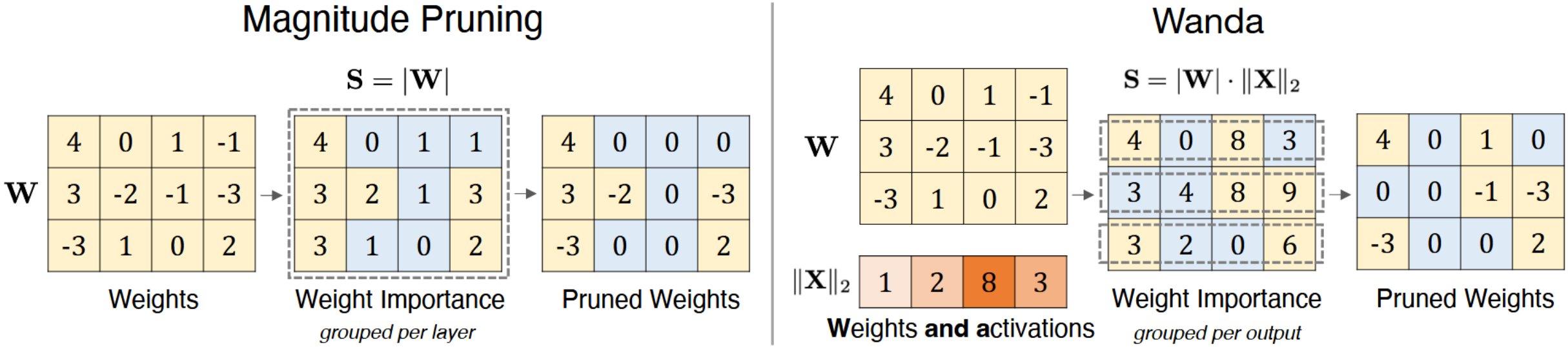
- prune weights with smallest absolute value
- prunes 40% of the weights with negligible performance loss
- by adding a retraining phase after pruning, we can prune 80% with no performance loss

Image credits: See et al. 2016

Slide by Dinesh Raghu

See, Abigail, Minh-Thang Luong, and Christopher D. Manning. 2016. "Compression of Neural Machine Translation Models via Pruning." In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*. Berlin, Germany: Association for Computational Linguistics, 291–301. doi:10.18653/v1/K16-1029.

Wanda



Sun, Mingjie, Zhuang Liu, Anna Bair, and J. Zico Kolter. 2024. "A Simple and Effective Pruning Approach for Large Language Models." In *International Conference on Learning Representations (ICLR)*, Vienna, Austria. https://iclr.cc/virtual/2024/poster/18687

Quantizing Models

Compresses weights and activations from floating point numbers to integers (e.g., 4-bit, 8-bit)

Then use the scaling factor to get the “original” value

Implementation:

https://pytorch.org/docs/stable/quantization.html

https://pypi.org/project/bitsandbytes/

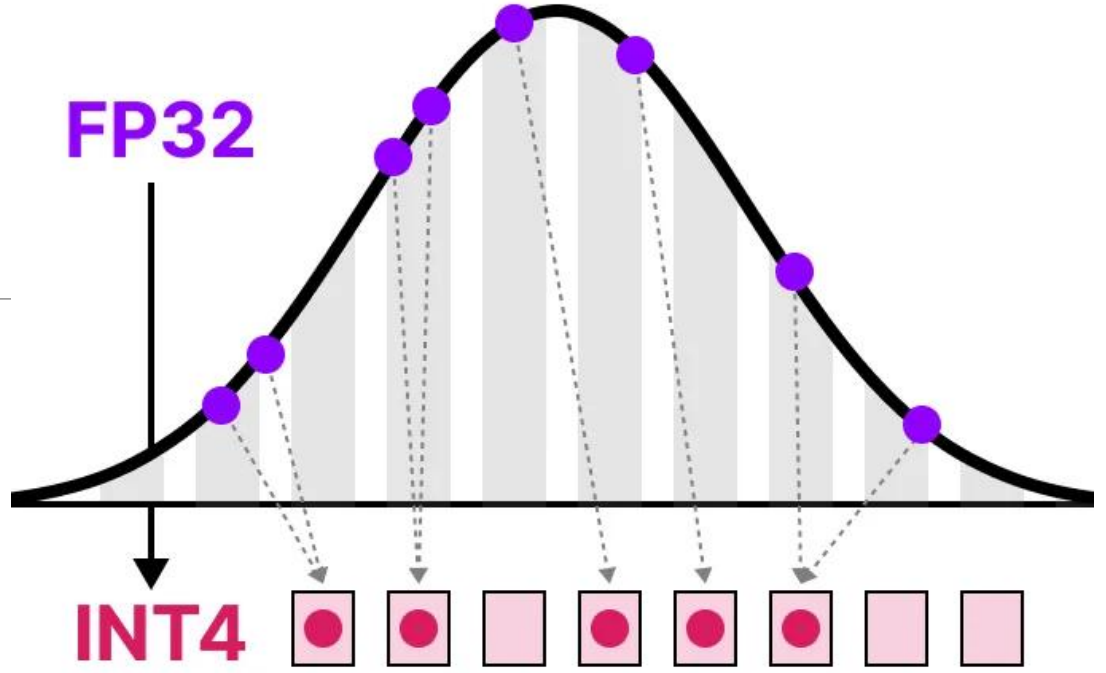
Learn more here:
https://huggingface.co/blog/hf-bitsandbytes-integration

Quantizing Models

Given a layer l with weight matrix W_l and layer input X_l, find quantized weight $\hat{W}_l$:

$$(\hat{W}_l^* = argmin_{\hat{W}_l} ||W_l X - \hat{W}_l X||_2^2)$$

https://huggingface.co/blog/merve/quantization

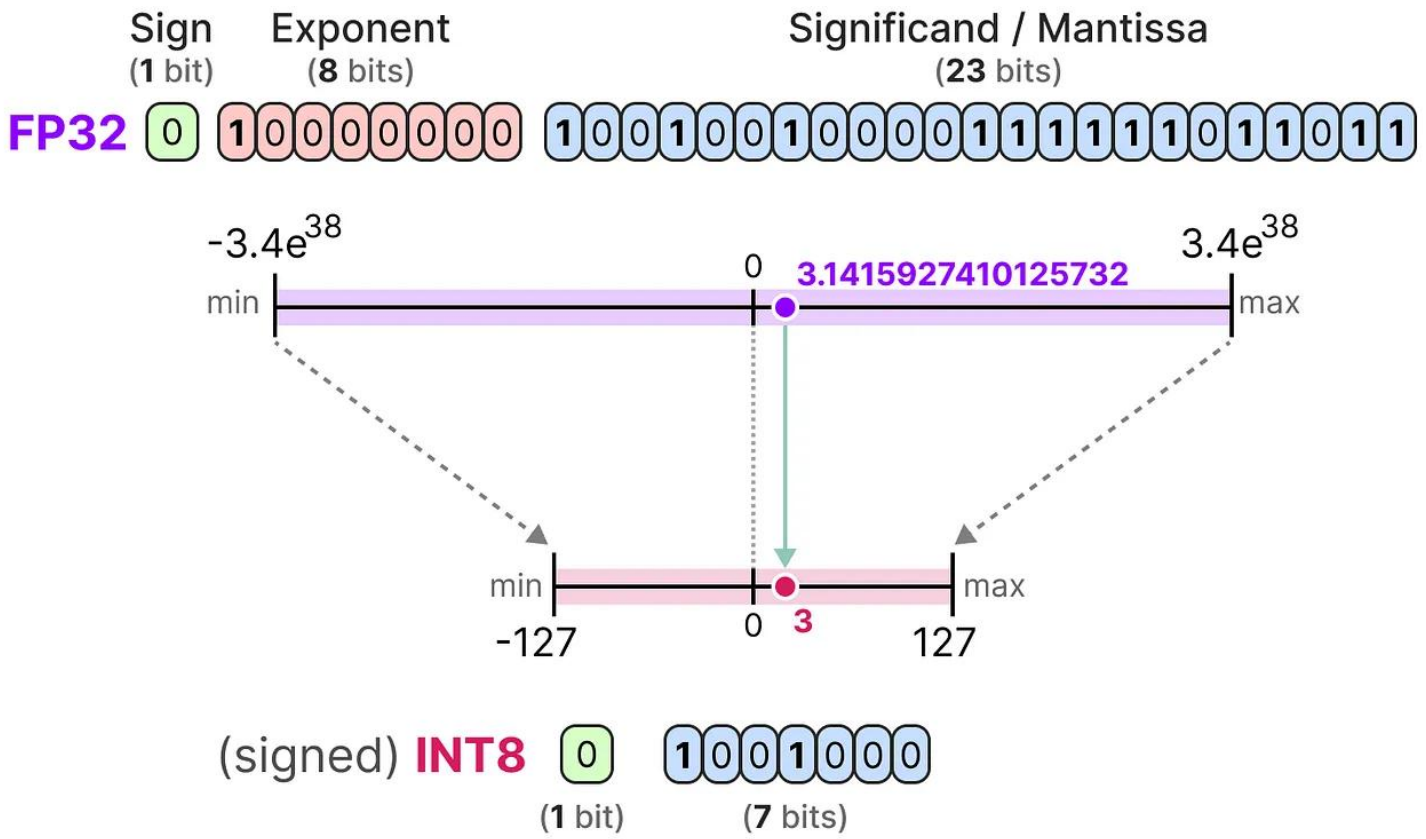


https://substack.com/home/post/p-145531349

$$\mathbf{X}^{\text{Int8}} = \text{round}\left(\frac{127}{\text{absmax}(\mathbf{X}^{\text{FP32}})}\mathbf{X}^{\text{FP32}}\right) = \text{round}(c^{\text{FP32}} \cdot \mathbf{X}^{\text{FP32}}),$$ where *c* is the *quantization constant*

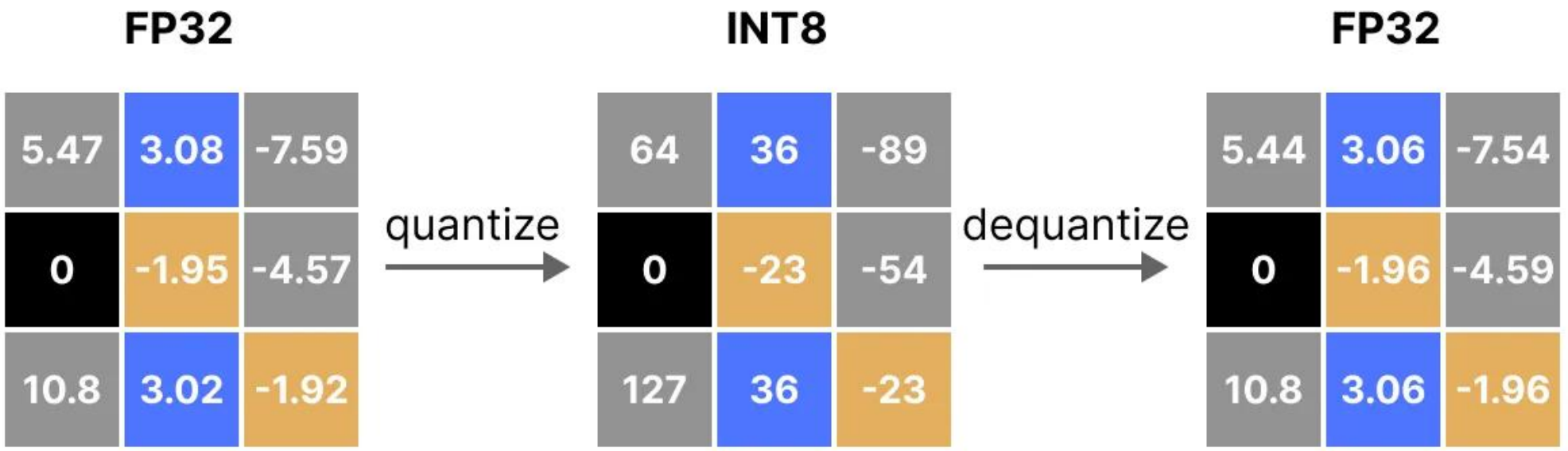
Dettmers, Tim, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. "QLoRA: Efficient Finetuning of Quantized LLMs." Advances in Neural Information Processing Systems

Mapping floating point to integer



https://substack.com/home/post/p-145531349

Quantizing → Dequantizing



https://substack.com/home/post/p-145531349

Quantizing/Pruning (Model Compression)

PROS

More efficient

Less resources

Can add more parameters (if quantizing)

Model interpretability (if pruning)

CONS

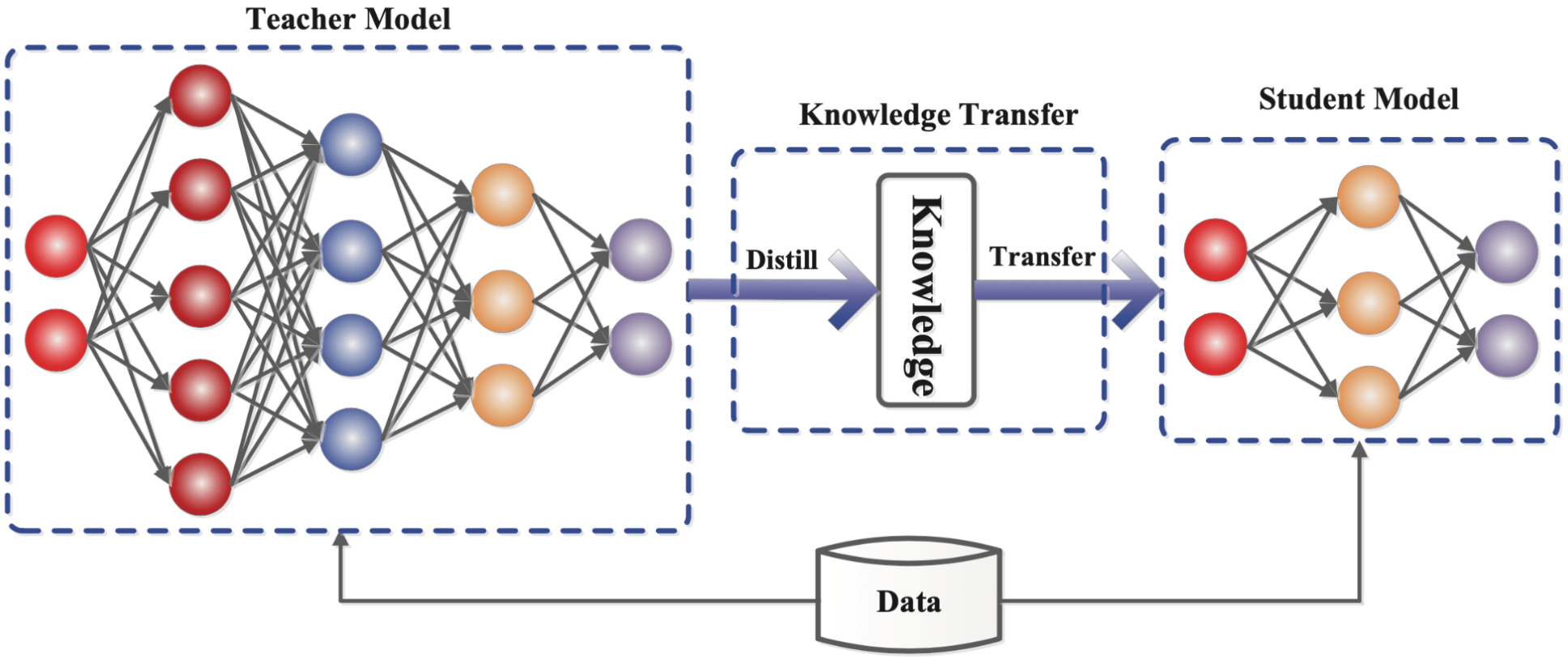
Losing information (lossy)

Possibly losing generalization

Adds a bit of time to dequanitize?

Knowledge Distillation

Knowledge Distillation

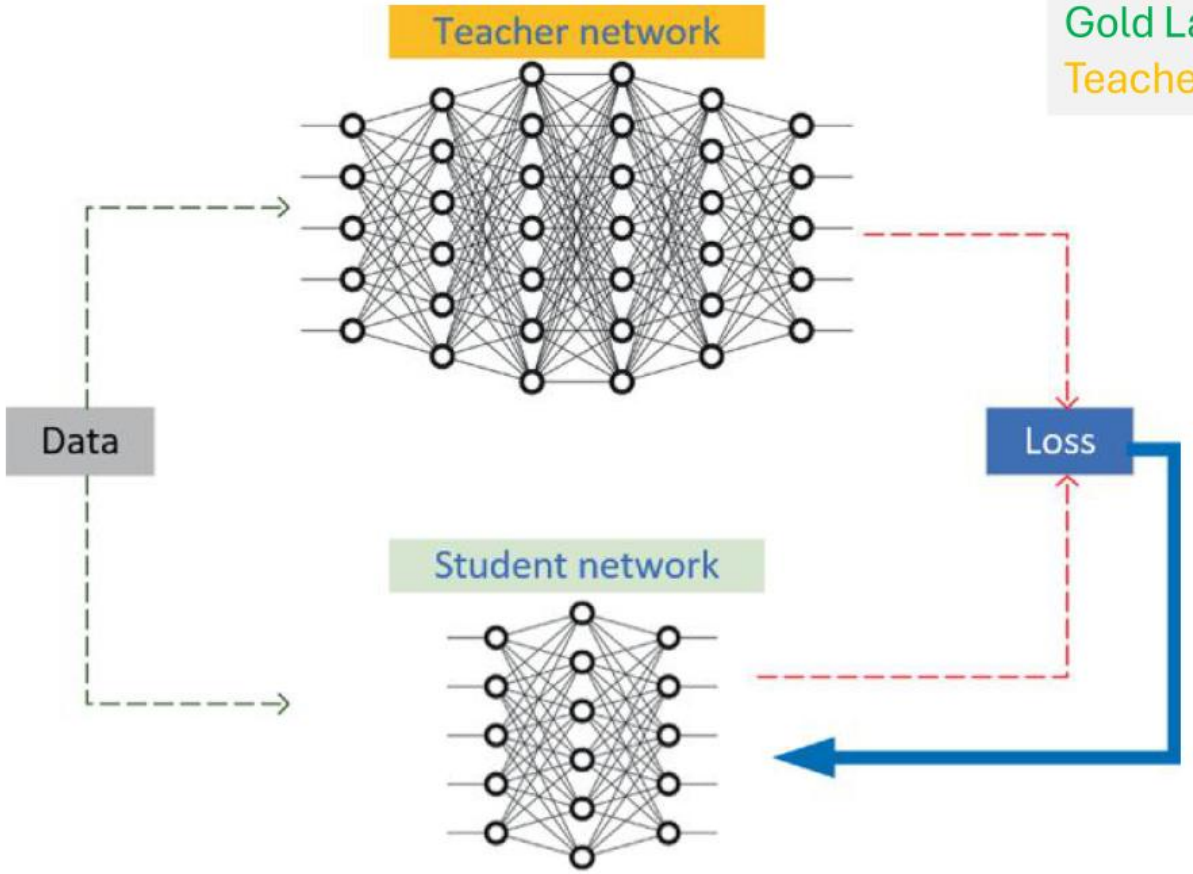


https://neptune.ai/blog/knowledge-distillation

Implementation Tutorial: https://pytorch.org/tutorials/beginner/knowledge_distillation_tutorial.html

Hinton, Geoffrey, Oriol Vinyals, and Jeff Dean. 2014. "Distilling the Knowledge in a Neural Network." In *NeurIPS 2014 Deep Learning Workshop*, doi:10.48550/arXiv.1503.02531.

Training the Student Network



Gold Label [0 0 1 0 0]
Teacher Prediction [0.10 0.20 0.50 0.15 0.05]

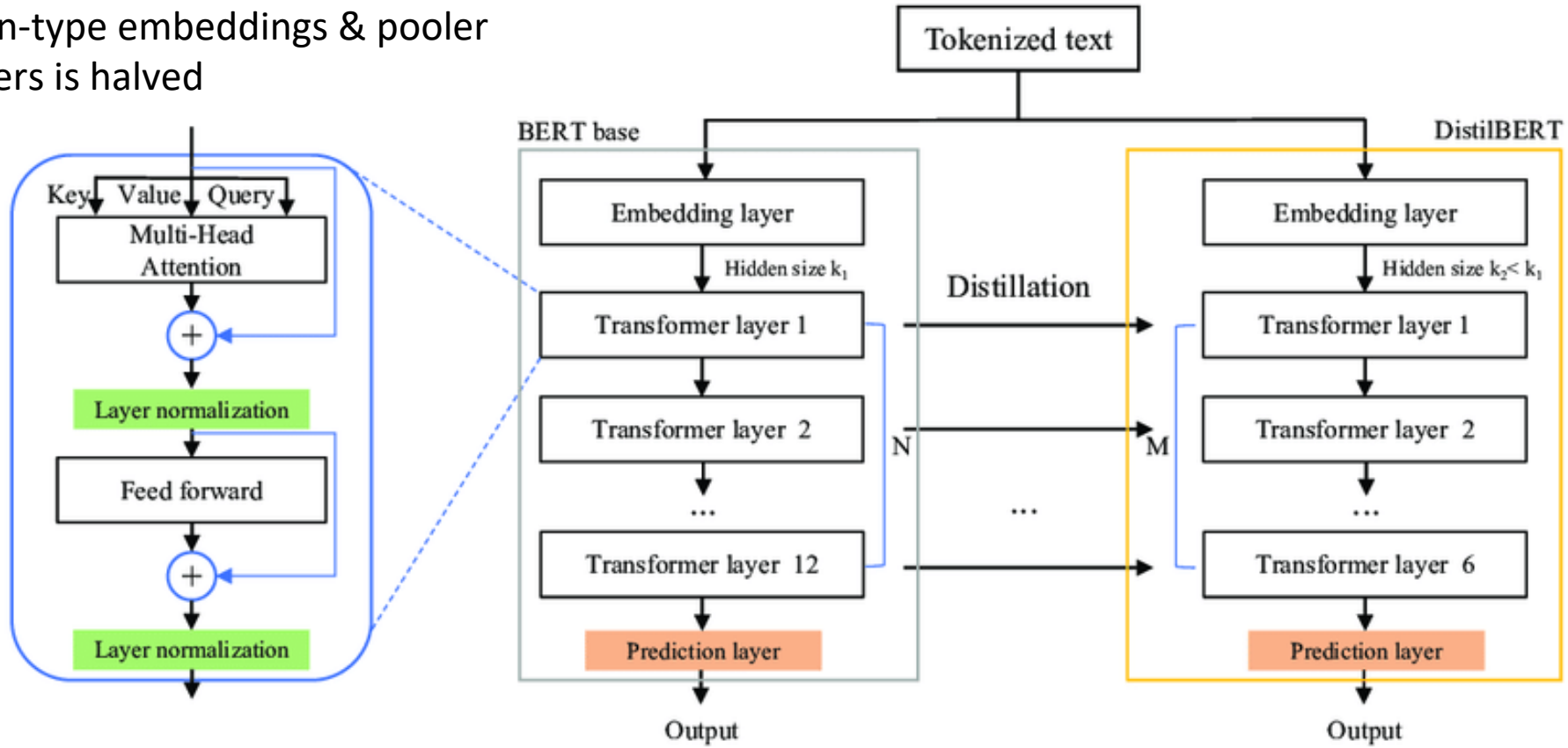
$$\mathcal{L}_{\text{NLL}}(\theta) = -\sum_{k=1}^{|\mathcal{V}|} \mathbb{1}\{y = k\} \log p(y = k \mid x; \theta)$$

$$\mathcal{L}_{\text{KD}}(\theta; \theta_T) = -\sum_{k=1}^{|\mathcal{V}|} q(y = k \mid x; \theta_T) \times \log p(y = k \mid x; \theta)$$

Slide by Dinesh Raghu

DistilBERT

Removed token-type embeddings & pooler
Number of layers is halved



https://www.researchgate.net/figure/The-DistilBERT-model-architecture-and-components_fig2_358239462

Sanh, Victor, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. "DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter." In *Workshop on Energy Efficient Machine Learning and Cognitive Computing - NeurIPS 2019*, Vancouver, Canada. doi:10.48550/arXiv.1910.01108.

Knowledge Distillation

PROS

Similar output but smaller architecture

Simple to implement

Flexibility in new model architecture

CONS

Time consuming to train both models

Time consuming to run through both models to train student

Need to decide what the new architecture is

Student picks up on teacher's biases

Student isn't as generalizable

Stealing?

PEFT

Parameter-efficient Fine-tuning (PEFT)

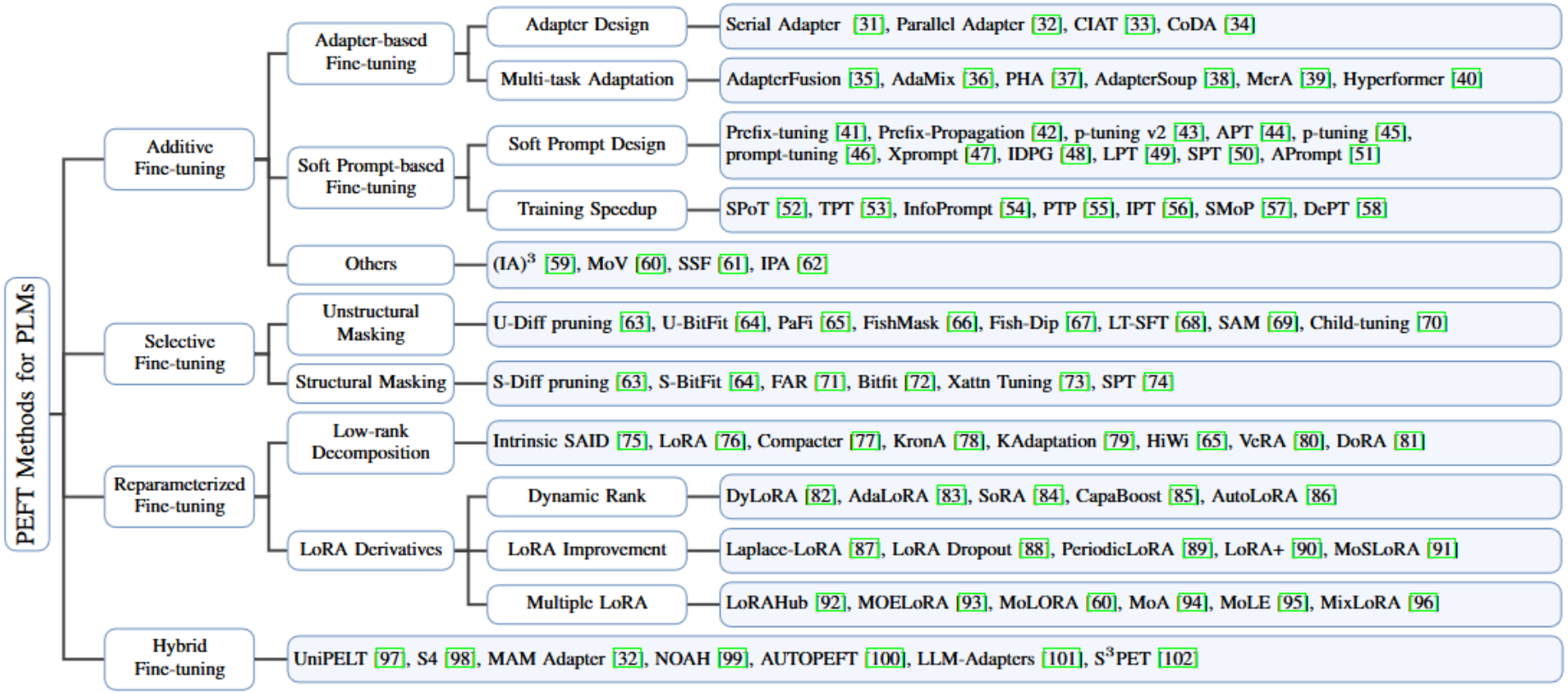


Fig. 3: Taxonomy of Parameter-Efficient Fine-Tuning Methods for Large Models.

Han, Zeyu, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. 2024. "Parameter-Efficient Fine-Tuning for Large Models: A Comprehensive Survey." doi:10.48550/arXiv.2403.14608.

LoRA (Low-Rank Adaptation)

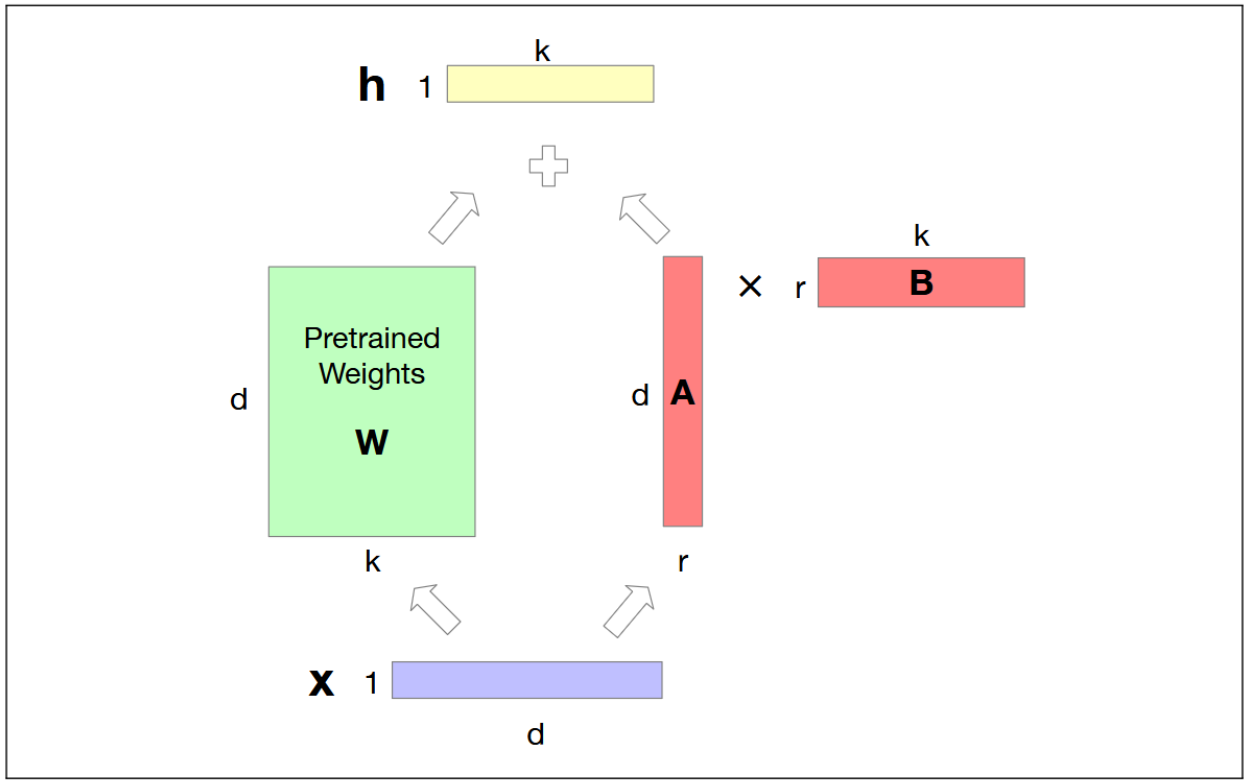


Figure 10.8 The intuition of LoRA. We freeze **W** to its pretrained values, and instead fine-tune by training a pair of matrices **A** and **B**, updating those instead of **W**, and just sum **W** and the updated **AB**.

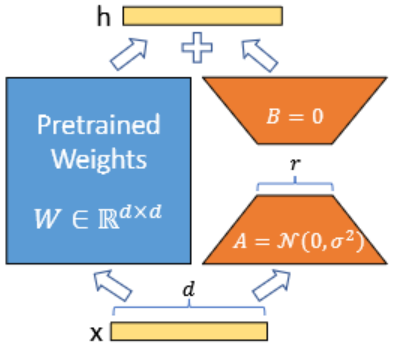
Train a model using a pretrained LLM but give the new model fewer parameters → a low-rank decomposition of the original weight matrix

$$h = xW + xAB$$

$$r << \min(d, N)$$

From SLP book Chapter 10

LoRA (Low-Rank Adaptation)



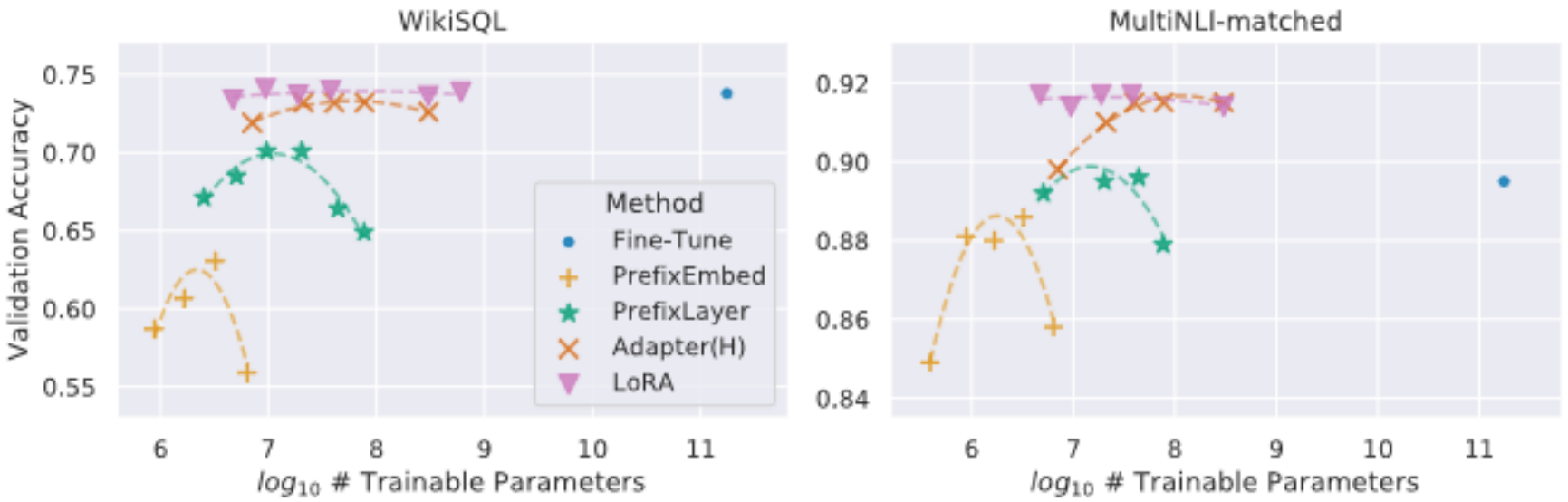
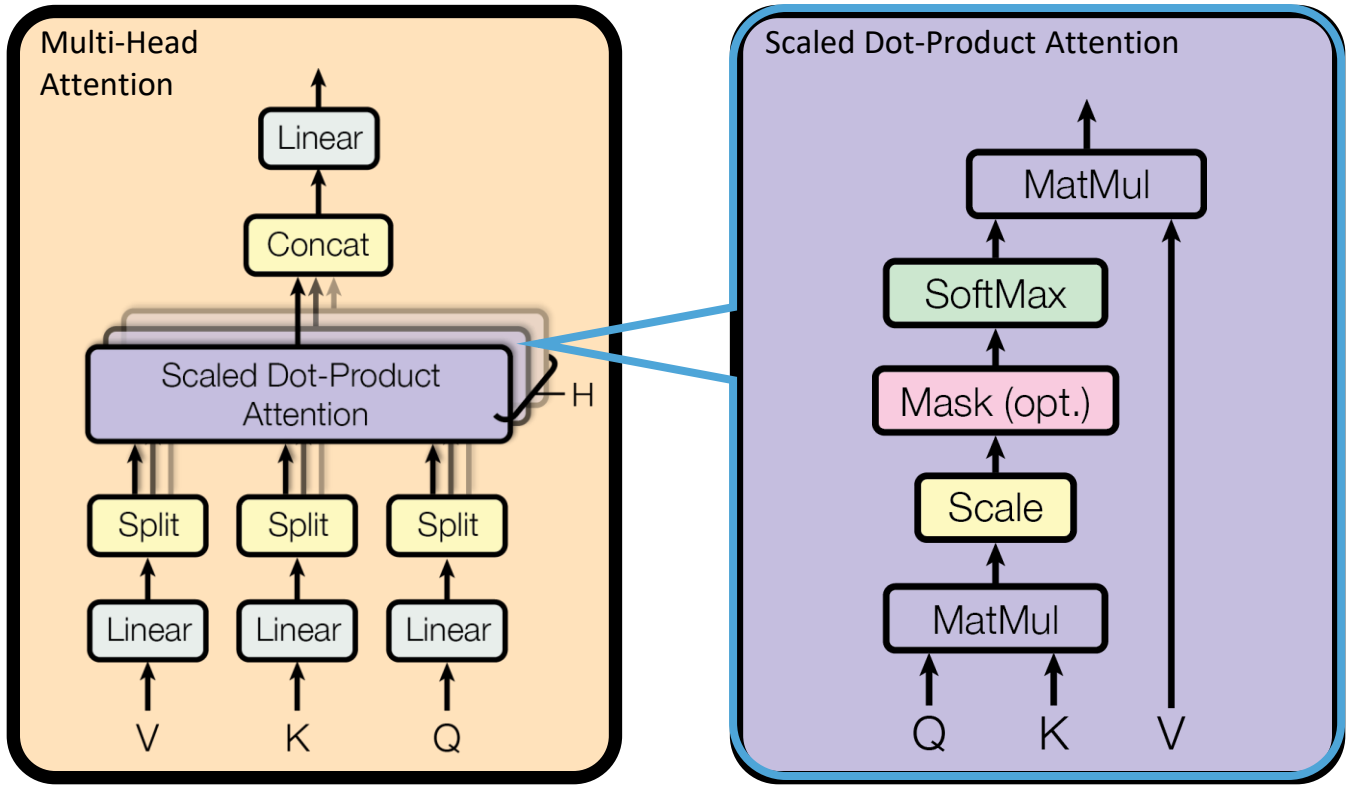


Figure 2: GPT-3 175B validation accuracy vs. number of trainable parameters of several adaptation methods on WikiSQL and MNLI-matched. LoRA exhibits better scalability and task performance. See Section I.2 for more details on the plotted data points.

Implementation:
https://github.com/microsoft/LoRA
https://huggingface.co/docs/diffusers/training/lora

Hu, Edward J., Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. "LoRA: Low-Rank Adaptation of Large Language Models." In *International Conference on Learning Representations (ICLR)*, Virtual. https://iclr.cc/virtual/2022/poster/6319

Review: Attention Mechanism



Original LoRA was just applied to the attention weights:
W_Q, W_K, W_V, and W_O

Guanaco: QLoRA

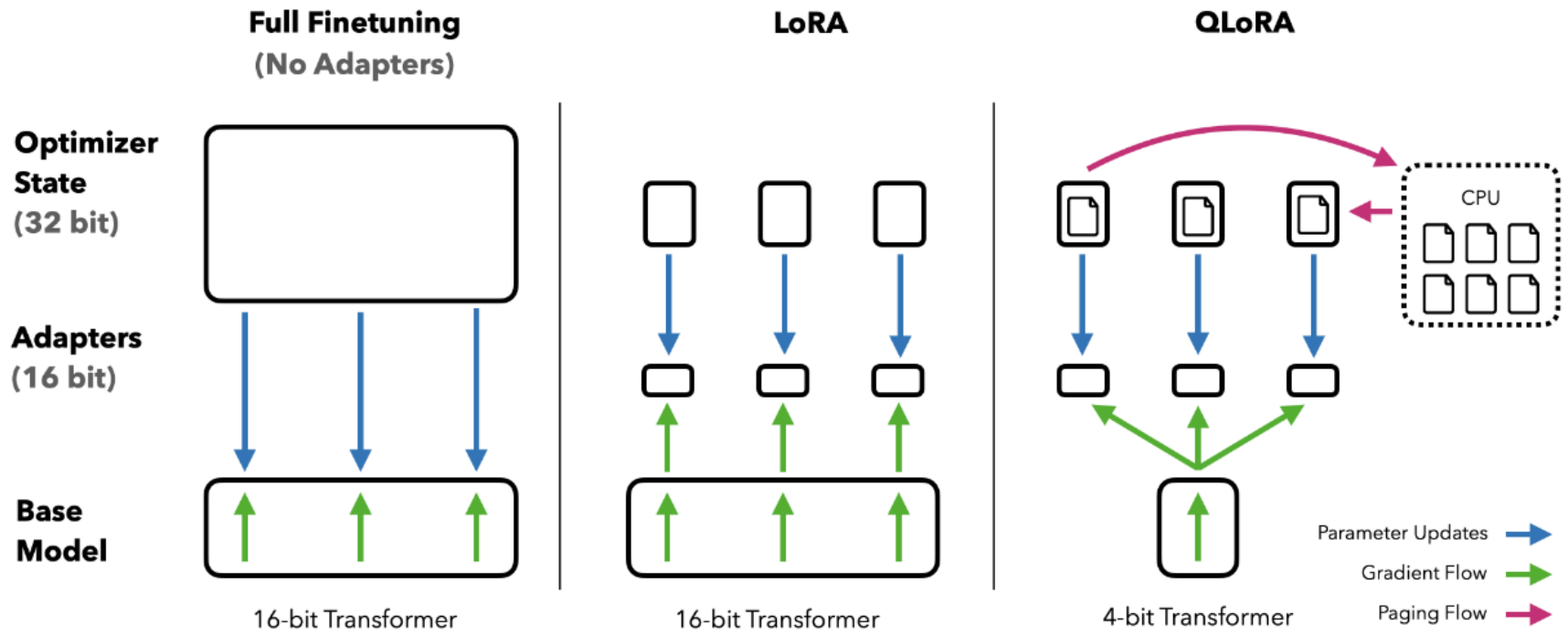


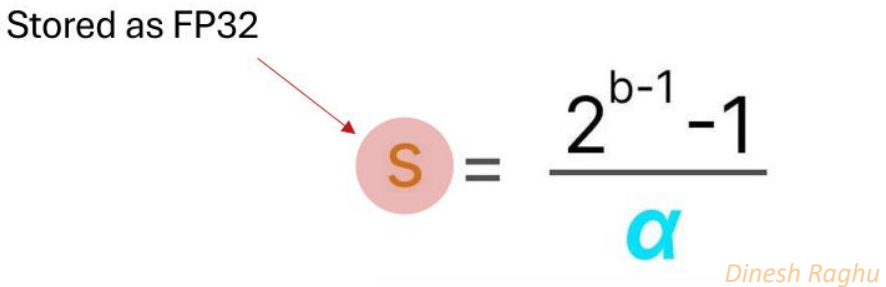
Figure 1: Different finetuning methods and their memory requirements. QLoRA improves over LoRA by quantizing the transformer model to 4-bit precision and using paged optimizers to handle memory spikes.

Dettmers, Tim, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. "QLoRA: Efficient Finetuning of Quantized LLMs." *Advances in Neural Information Processing Systems* 36: 10088–115. https://proceedings.neurips.cc/paper_files/paper/2023/hash/1feb87871436031bdc0f2beaa62a049b-Abstract-Conference.html

Guanaco: QLoRA

4-bit NormalFloat quantization

Uses *double quantization* (quantizing the quantization constants)



Dettmers, Tim, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. "QLoRA: Efficient Finetuning of Quantized LLMs." *Advances in Neural Information Processing Systems* 36: 10088–115. https://proceedings.neurips.cc/paper_files/paper/2023/hash/1feb87871436031bdc0f2beaa62a049b-Abstract-Conference.html

Guanaco: QLoRA

Table 3: Experiments comparing 16-bit BrainFloat (BF16), 8-bit Integer (Int8), 4-bit Float (FP4), and 4-bit NormalFloat (NF4) on GLUE and Super-NaturalInstructions. QLoRA replicates 16-bit LoRA and full-finetuning.

Dataset	GLUE (Acc.)	Super-NaturalInstructions (RougeL)				
Model	RoBERTa-large	T5-80M	T5-250M	T5-780M	T5-3B	T5-11B
BF16	88.6	40.1	42.1	48.0	54.3	62.0
BF16 replication	88.6	40.0	42.2	47.3	54.9	-
LoRA BF16	88.8	40.5	42.6	47.1	55.4	60.7
QLoRA Int8	88.8	40.4	42.9	45.4	56.5	60.7
QLoRA FP4	88.6	40.3	42.4	47.5	55.6	60.9
QLoRA NF4 + DQ	-	40.4	42.7	47.7	55.3	60.9

Table 8: Evaluation of biases on the CrowS dataset. A lower score indicates lower likelihood of generating biased sequences. Guanaco follows the biased pattern of the LLaMA base model.

	LLaMA-65B	GPT-3	OPT-175B	Guanaco-65B
Gender	70.6	62.6	65.7	**47.5**
Religion	79.0	73.3	68.6	**38.7**
Race/Color	57.0	64.7	68.6	**45.3**
Sexual orientation	81.0	76.2	78.6	**59.1**
Age	70.1	64.4	67.8	**36.3**
Nationality	64.2	61.6	62.9	**32.4**
Disability	66.7	76.7	76.7	**33.9**
Physical appearance	77.8	74.6	76.2	**43.1**
Socioeconomic status	71.5	73.8	76.2	**55.3**
Average	66.6	67.2	69.5	**43.5**

Dettmers, Tim, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. "QLoRA: Efficient Finetuning of Quantized LLMs." *Advances in Neural Information Processing Systems* 36: 10088–115. https://proceedings.neurips.cc/paper_files/paper/2023/hash/1feb87871436031bdc0f2beaa62a049b-Abstract-Conference.html

LoRA

PROS

Preserving a lot of the original model calculations

Does about the same performance-wise

Significantly smaller than the original

CONS

Potentially slightly worse for larger models

Efficient Training

Review: Transformer Architecture

Multi-Head Attention $= \text{MultiHeadAtt}(\mathbf{H}_i^{enc}, \mathbf{H}_i^{enc}, \mathbf{H}_i^{enc})$

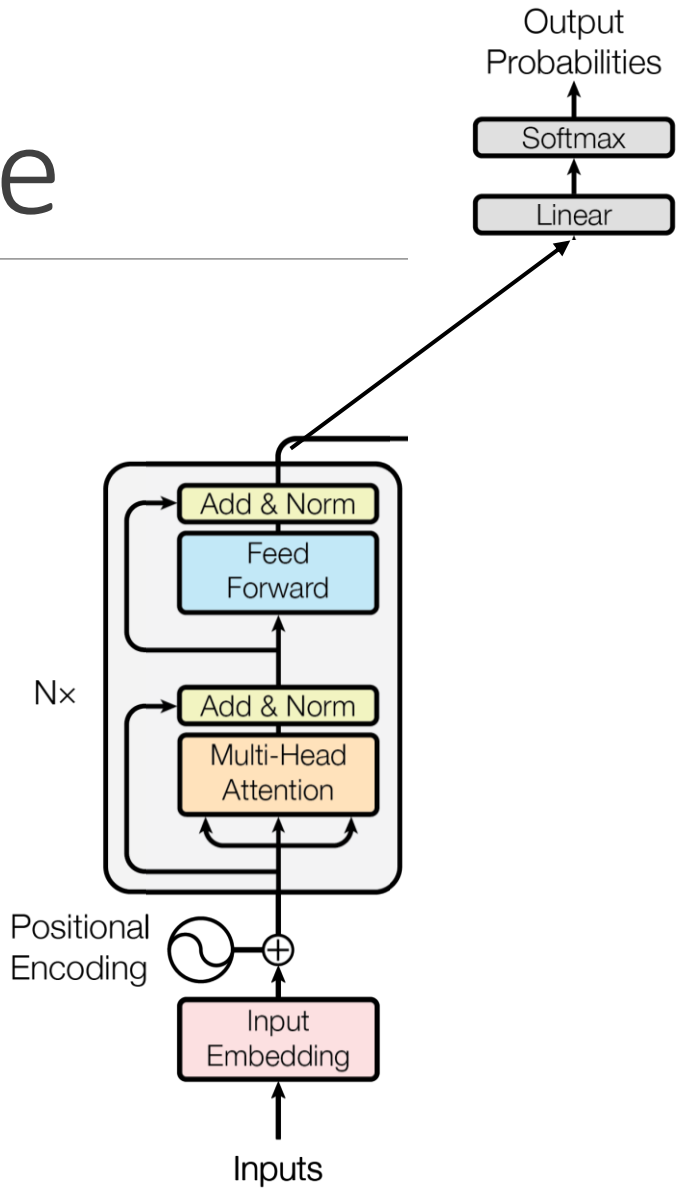
Add & Norm $= \text{LayerNorm}(\text{Multi-Head Attention} + \mathbf{H}_i^{enc})$

Feed Forward $= max(0, \text{Add \& Norm}\,\mathbf{W}_1 + b_1)\mathbf{W}_2 + b_2$

Add & Norm (2) $= \text{LayerNorm}(\text{Feed Forward} + \mathbf{H}_i^{enc})$

$\mathbf{H}_{i+1}^{enc} =$ Add & Norm (2)

A decoder-only architecture is very similar to the encoder of the original transformer architecture



Slide adapted from Daphne Ippolito

Sparse Mixture-of-Experts

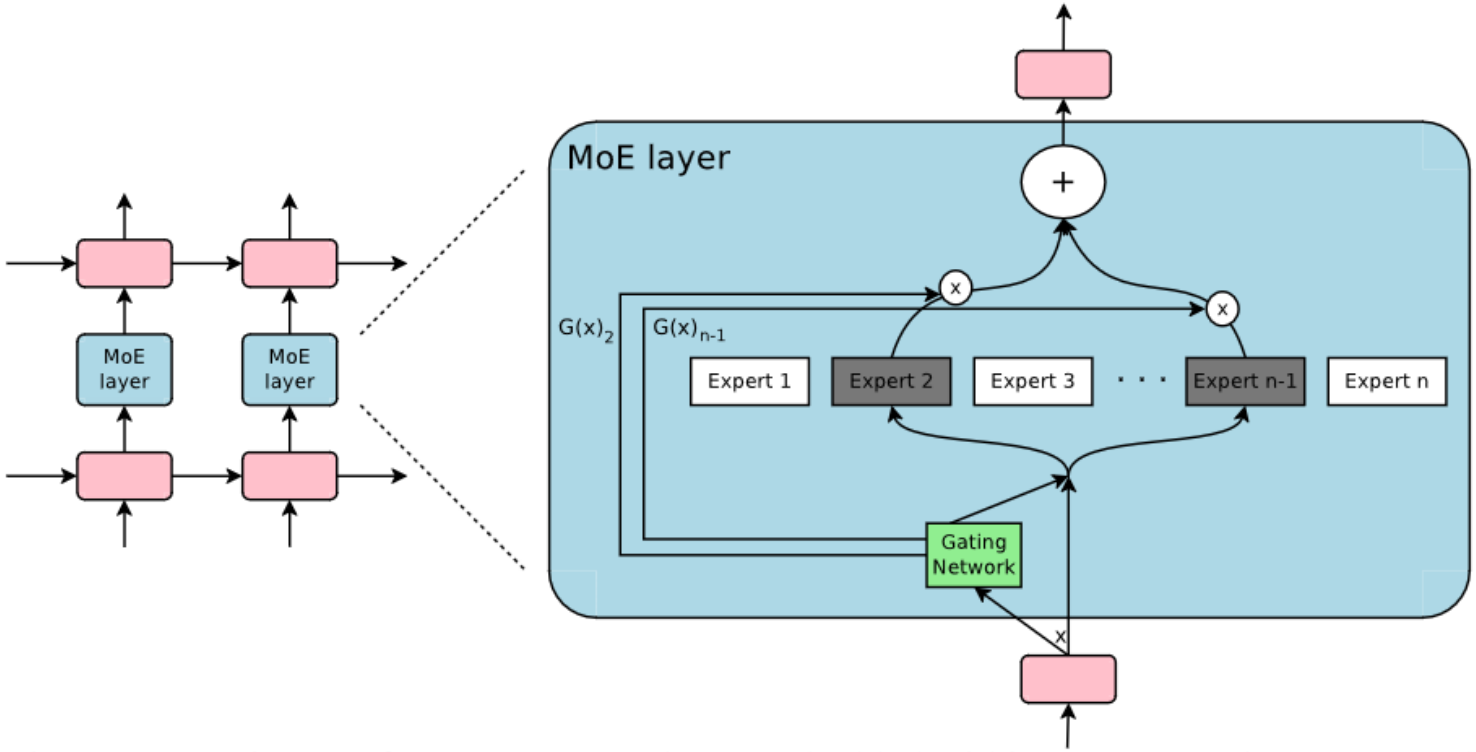


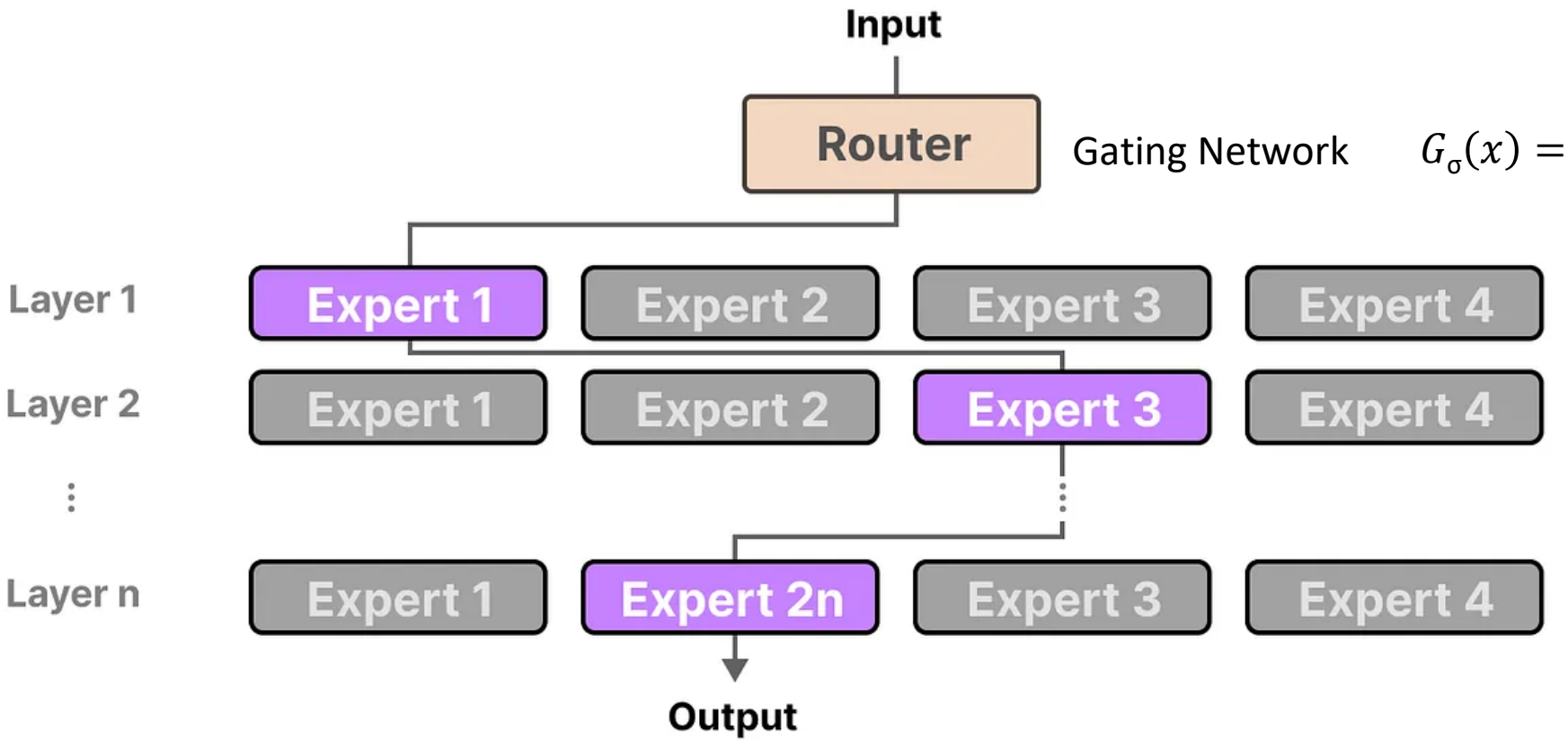
Figure 1: A Mixture of Experts (MoE) layer embedded within a recurrent language model. In this case, the sparse gating function selects two experts to perform computations. Their outputs are modulated by the outputs of the gating network.

Implementations
Megablocks: https://github.com/stanford-futuredata/megablocks
Fairseq: https://github.com/facebookresearch/fairseq/tree/main/examples/moe_lm
OpenMoE: https://github.com/XueFuzhao/OpenMoE

Shazeer, Noam, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. "Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer." In *International Conference on Learning Representations (ICLR)*, Toulon, France. https://openreview.net/forum?id=B1ckMDqlg

Mixture-of-Experts

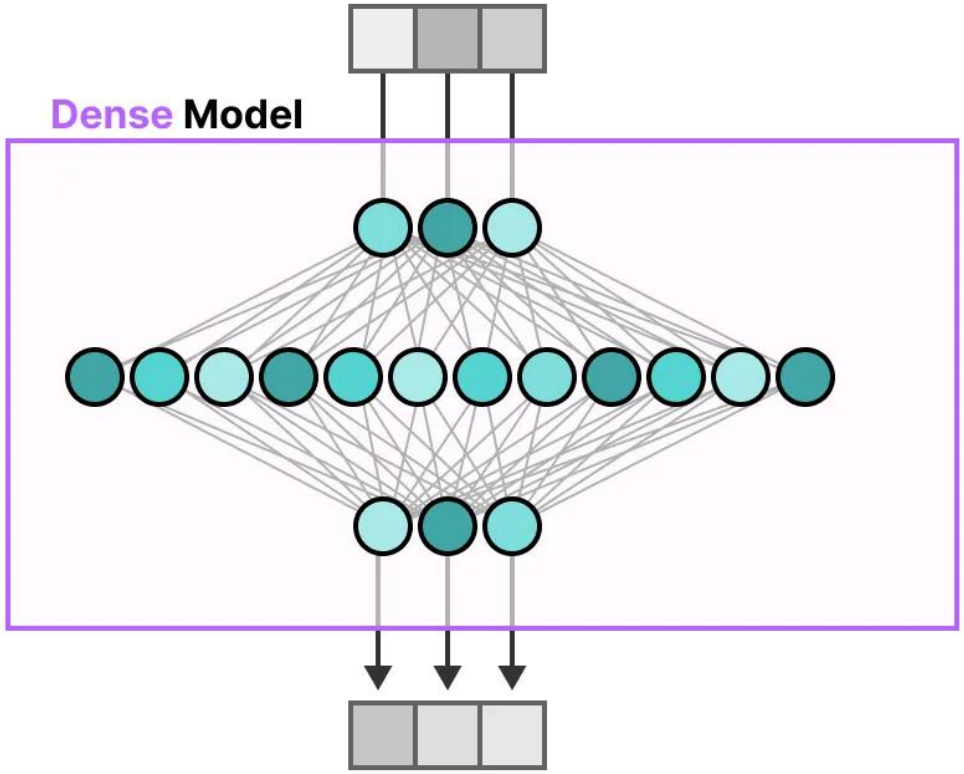
$$y = \sum_{i=1}^{n} G(x)_i E_i(x)$$

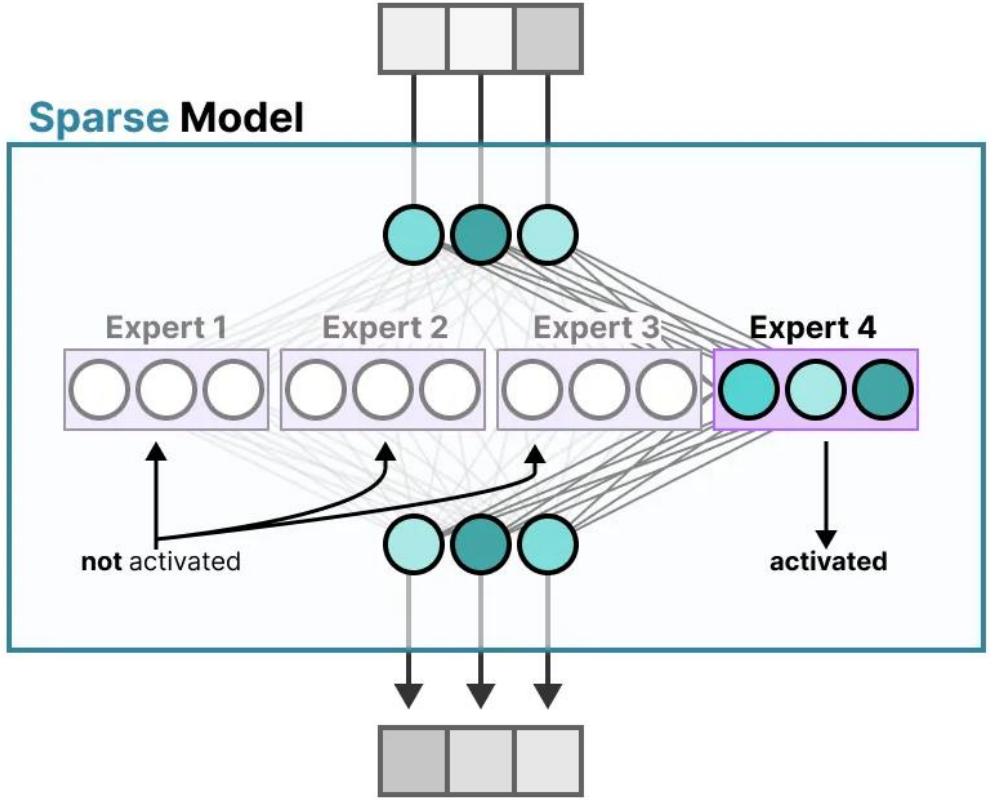


https://substack.com/home/post/p-148217245

Equations from https://huggingface.co/blog/moe

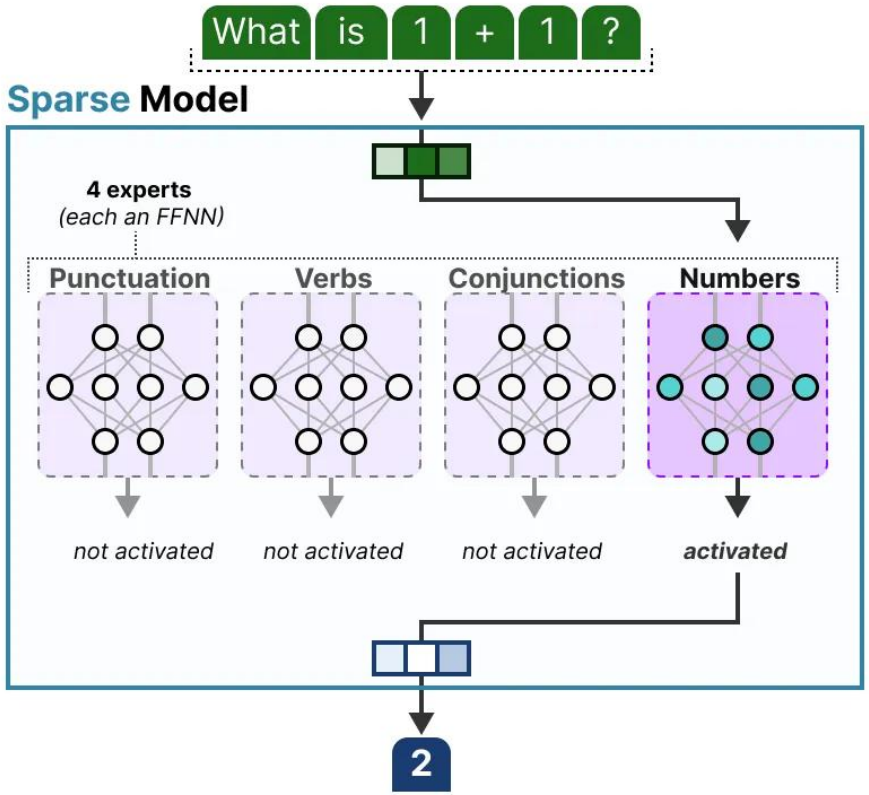
Sparse MoE



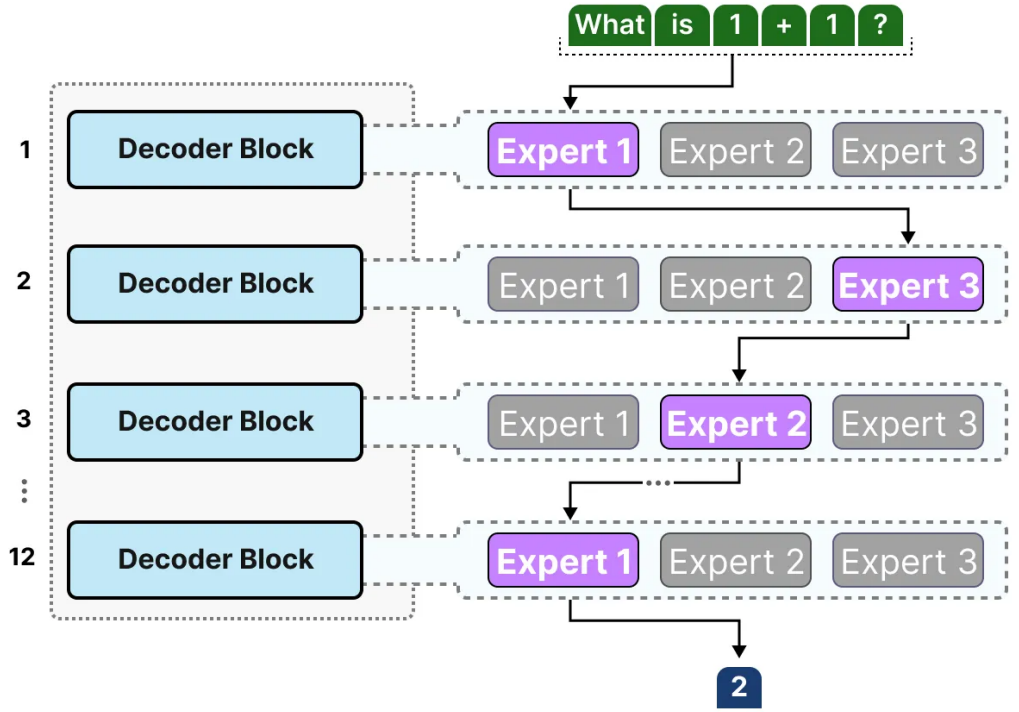


https://substack.com/home/post/p-148217245

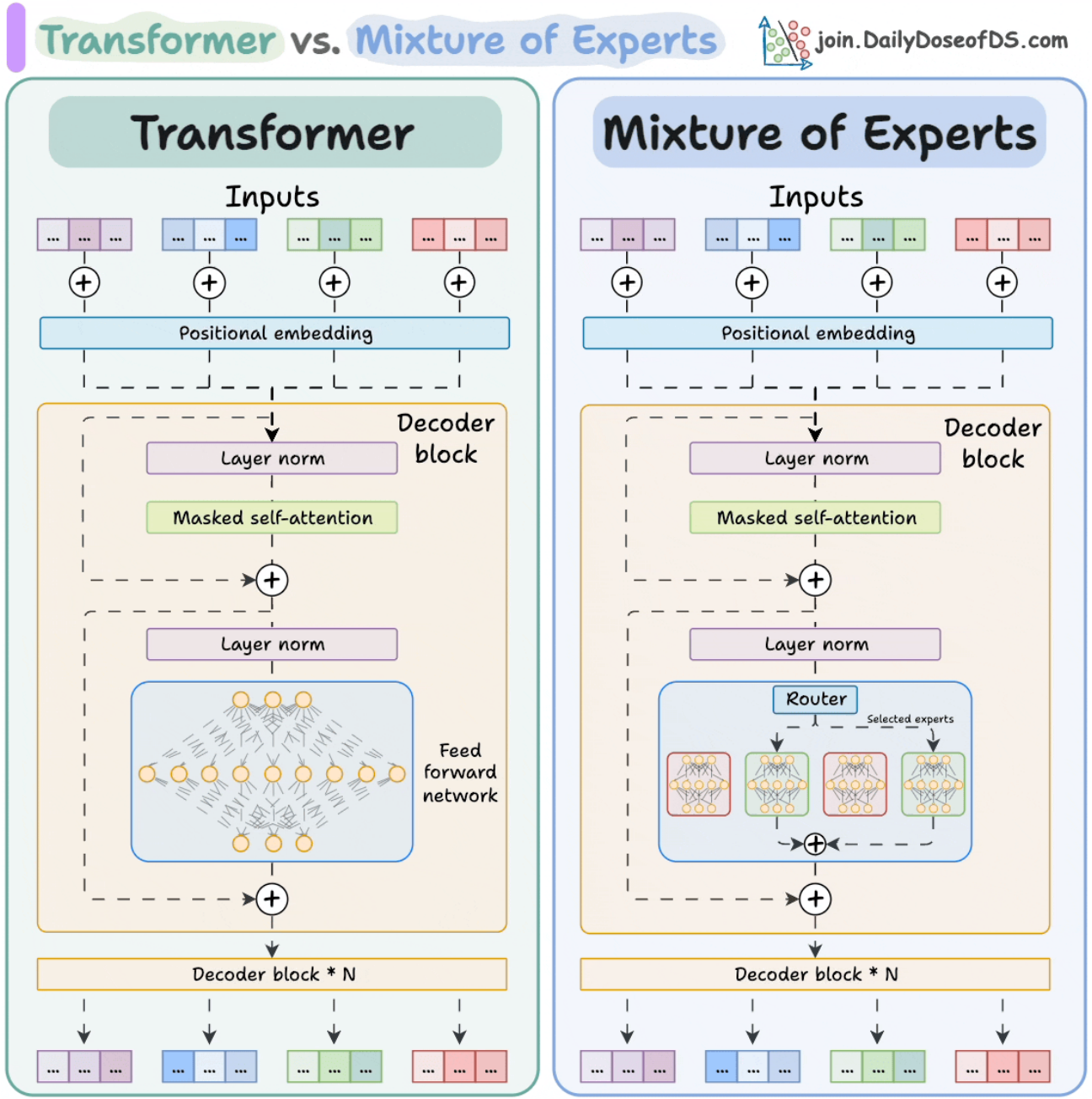
MoE Example



Most transformers have multiple decoder blocks



https://substack.com/home/post/p-148217245


Transformer vs. Mixture of Experts
join.DailyDoseofDS.com
Transformer
Inputs
Positional embedding
Decoder block
Layer norm
Masked self-attention
Layer norm
Feed forward network
Decoder block * N
Mixture of Experts
Inputs
Positional embedding
Decoder block
Layer norm
Masked self-attention
Layer norm
Router
Selected experts
Decoder block * N

8 experts considered
Top 2 experts Selected

Mixtral 8x7B

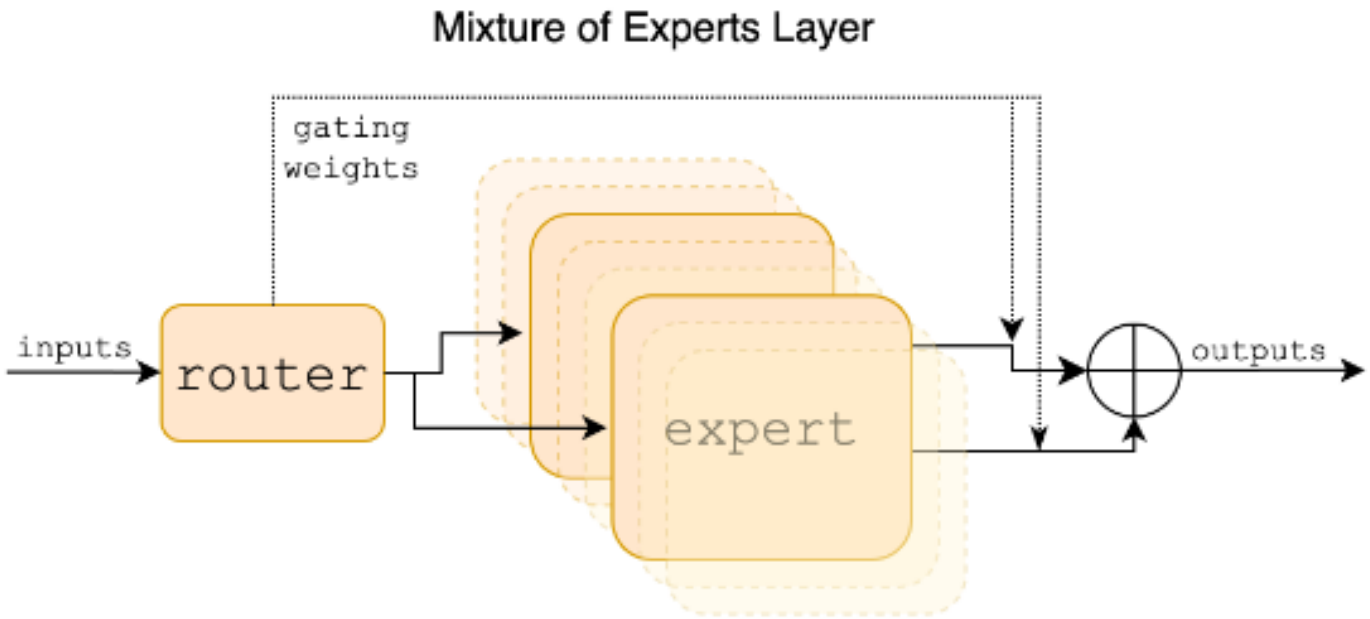


Figure 1: Mixture of Experts Layer. Each input vector is assigned to 2 of the 8 experts by a router. The layer's output is the weighted sum of the outputs of the two selected experts. In Mixtral, an expert is a standard feedforward block as in a vanilla transformer architecture.

Jiang, Albert Q., Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, et al. 2024. "Mixtral of Experts." doi:10.48550/arXiv.2401.04088.

Mixtral is 13B active parameters vs LLaMA 2's 70B

Mixtral 8x7B

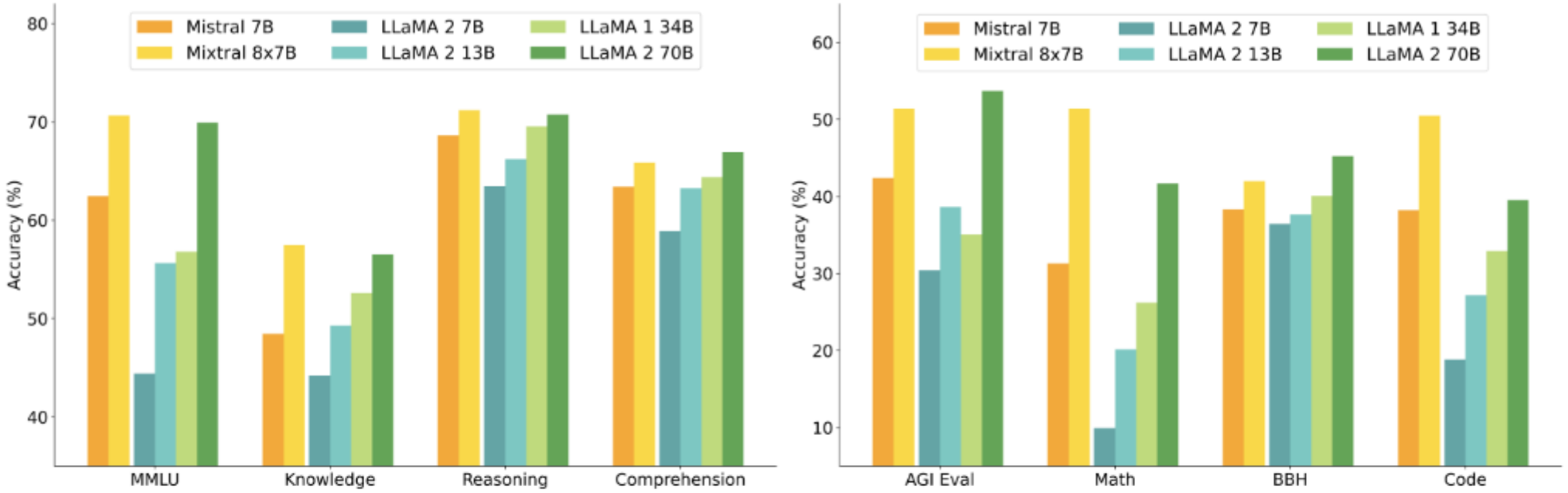
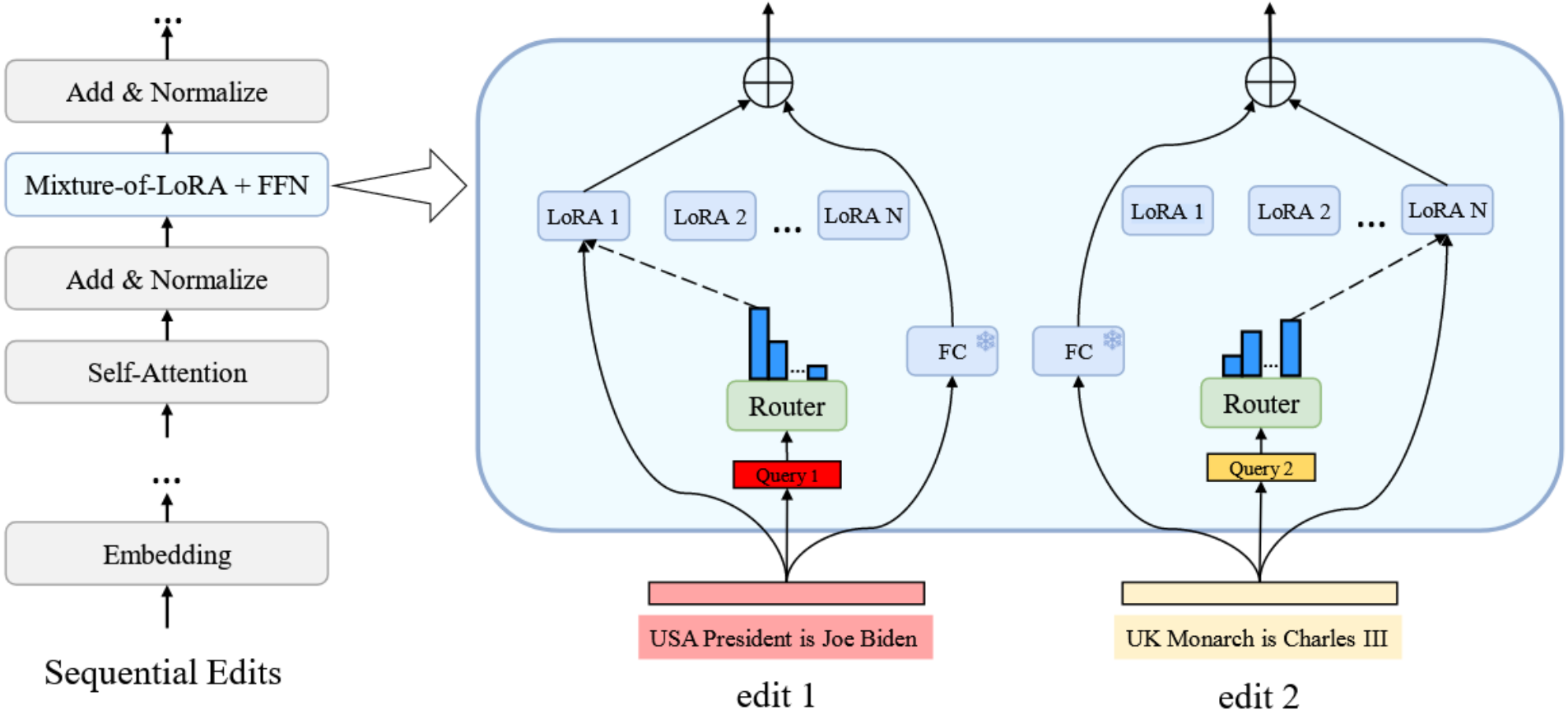


Figure 2: Performance of Mixtral and different Llama models on a wide range of benchmarks. All models were re-evaluated on all metrics with our evaluation pipeline for accurate comparison. Mixtral outperforms or matches Llama 2 70B on all benchmarks. In particular, it is vastly superior in mathematics and code generation.

Jiang, Albert Q., Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, et al. 2024. "Mixtral of Experts." doi:10.48550/arXiv.2401.04088.

ELDER: Mixture-of-LoRA



Li, Jiaang, Quan Wang, Zhongnan Wang, Yongdong Zhang, and Zhendong Mao. 2025. "ELDER: Enhancing Lifelong Model Editing with Mixture-of-LoRA." *Proceedings of the AAAI Conference on Artificial Intelligence* 39(23): 24440–48. doi:10.1609/aaai.v39i23.34622.

Mixture of Experts

PROS

Fewer parameters activated

Potentially more relevant answers

CONS

Same amount of parameters as original transformer

Potentially more errors if routed wrong

Might not work well if not clear categories to separate into experts