

LLM fine-tuning & evaluation

PRACTICAL INSIGHTS

Motivation

Motivation

We Have No Moat

And neither does OpenAI

We've done a lot of looking over our shoulders at OpenAI. Who will cross the next milestone? What will the next move be?

But the uncomfortable truth is, *we aren't positioned to win this arms race and neither is OpenAI*. While we've been squabbling, a third faction has been quietly eating our lunch.

I'm talking, of course, about open source. Plainly put, they are lapping us. Things we consider "major open problems" are solved and in people's hands today. Just to name a few:

- **LLMs on a Phone:** People are running foundation models on a Pixel 6 at 5 tokens / sec.
- **Scalable Personal AI:** You can finetune a personalized AI on your laptop in an evening.
- **Responsible Release:** This one isn't "solved" so much as "obviated". There are entire websites full of art models with no restrictions whatsoever, and text is not far behind.
- **Multimodality:** The current multimodal ScienceQA SOTA was trained in an hour.

Source: (google employee, anonymized), (2023). *We Have No Moat, And Neither Does OpenAI*. 2024, from <https://www.semianalysis.com/p/google-we-have-no-moat-and-neither>.

Motivation

We Have No Moat

And neither does OpenAI

We've done a lot of looking over our shoulders at OpenAI. Who will cross the next milestone? What will the next move be?

But the uncomfortable truth is, *we aren't positioned to win this arms race and neither is OpenAI*. While we've been squabbling, a third faction has been quietly eating our lunch.

I'm talking, of course, about open source. Plainly put, they are lapping us. Things we consider "major open problems" are solved and in people's hands today. Just to name a few:

- **LLMs on a Phone:** People are running foundation models on a Pixel 6 at 5 tokens / sec.
- **Scalable Personal AI:** You can finetune a personalized AI on your laptop in an evening.
- **Responsible Release:** This one isn't "solved" so much as "obviated". There are entire websites full of art models with no restrictions whatsoever, and text is not far behind.
- **Multimodality:** The current multimodal ScienceQA SOTA was trained in an hour.

Source: (google employee, anonyne), (2023). *We Have No Moat, And Neither Does OpenAI*. 2024, from <https://www.semianalysis.com/p/google-we-have-no-moat-and-neither>.



clem 🧐
@ClementDelangue

My prediction: in 2024, most companies will realize that smaller, cheaper, more specialized models make more sense for 99% of AI use-cases. The current market & usage is fooled by companies sponsoring the cost of training and running big models (especially with cloud incentives).



The_AI_Skeptic @The_AI_Skeptic · Oct 10, 2023

The cost of running AI is becoming an issue, even in domains where people are prepared to pay for it:

"Report: GitHub Copilot Loses an Average of \$20 Per User Per Month"

...

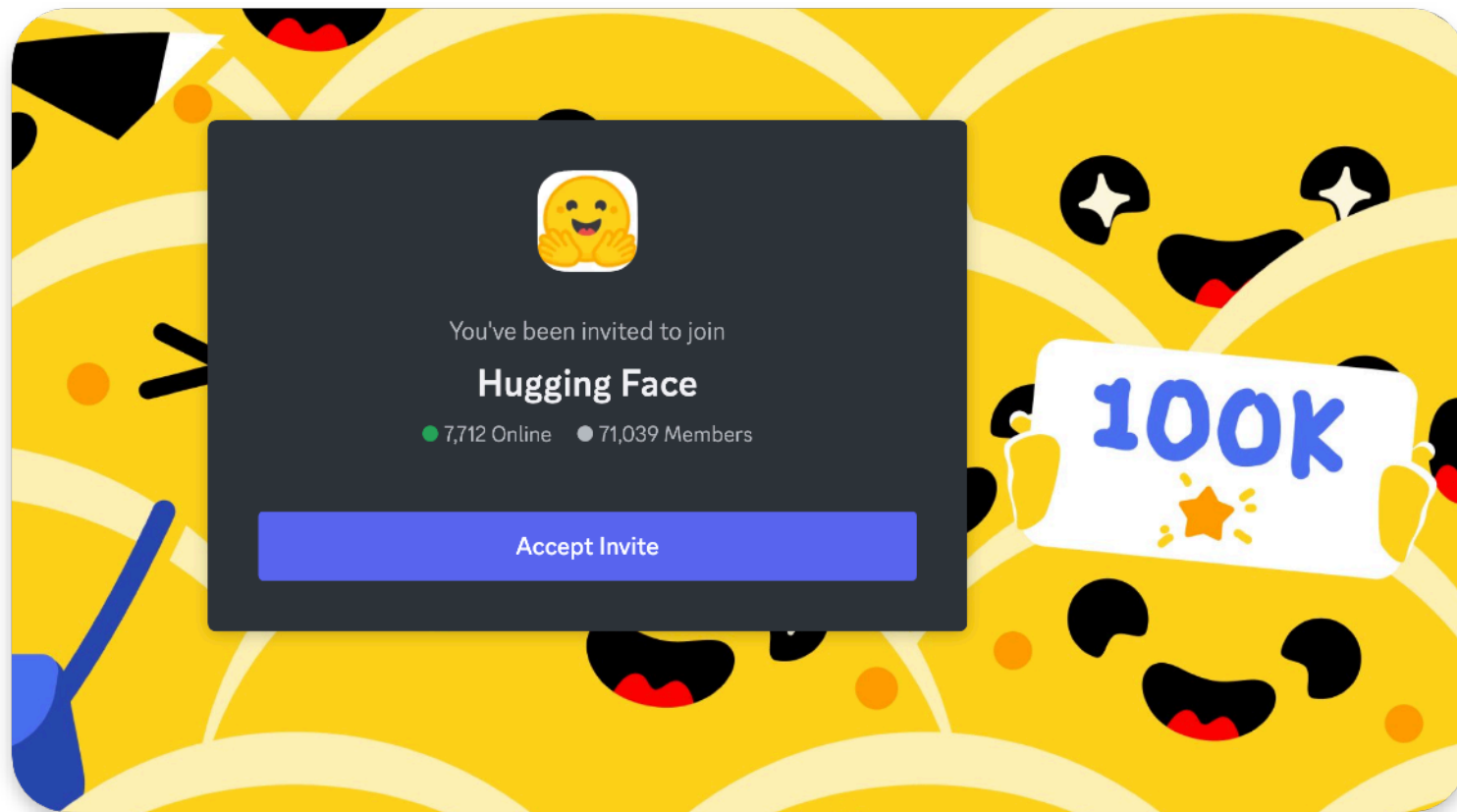
[Show more](#)

3:17 PM · Oct 10, 2023 · 657.9K Views

Source: (Clement Delangue), Hugging Face Cofounder, prediction for 2024 (2023), from x.com

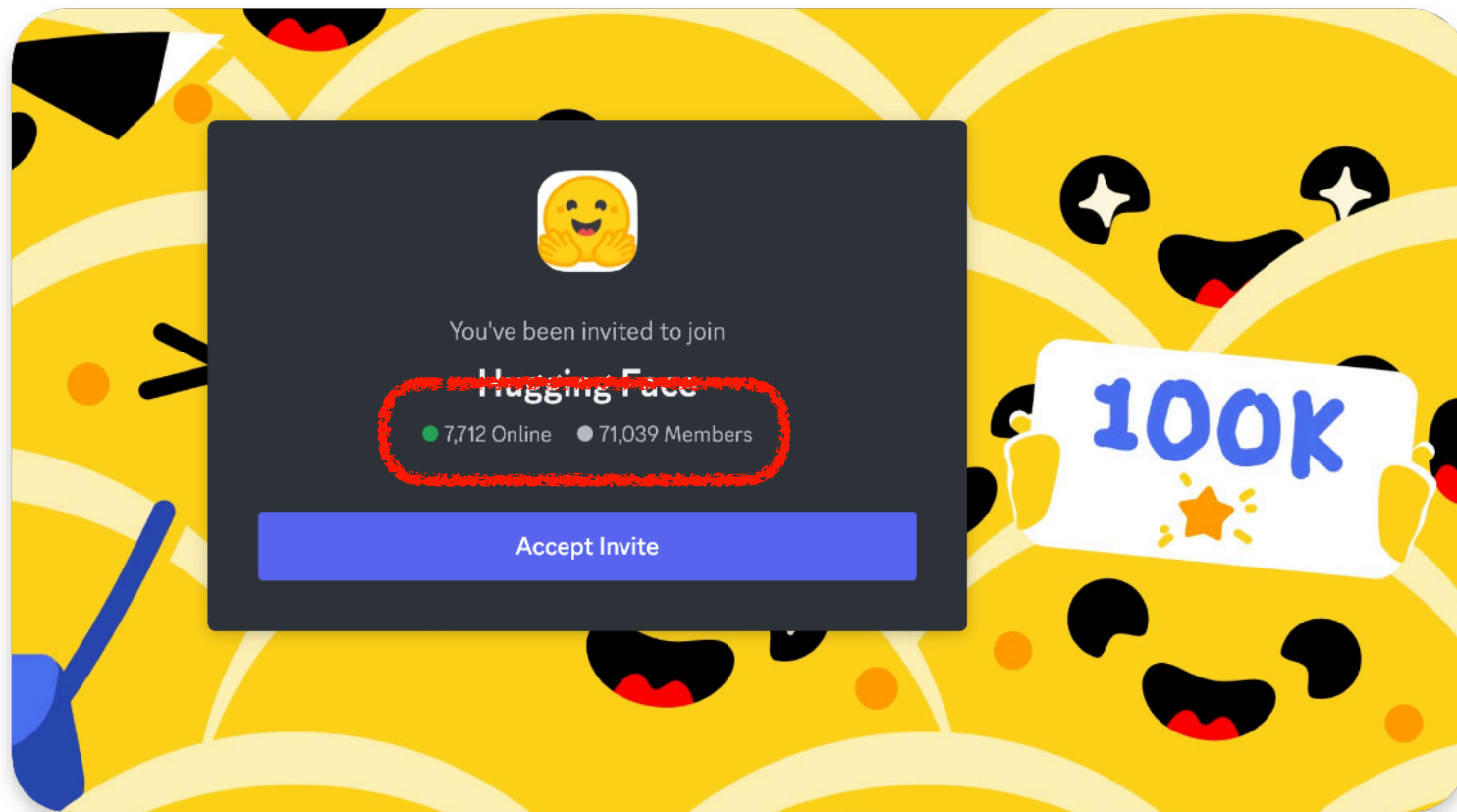
A dynamic ecosystem

A dynamic ecosystem



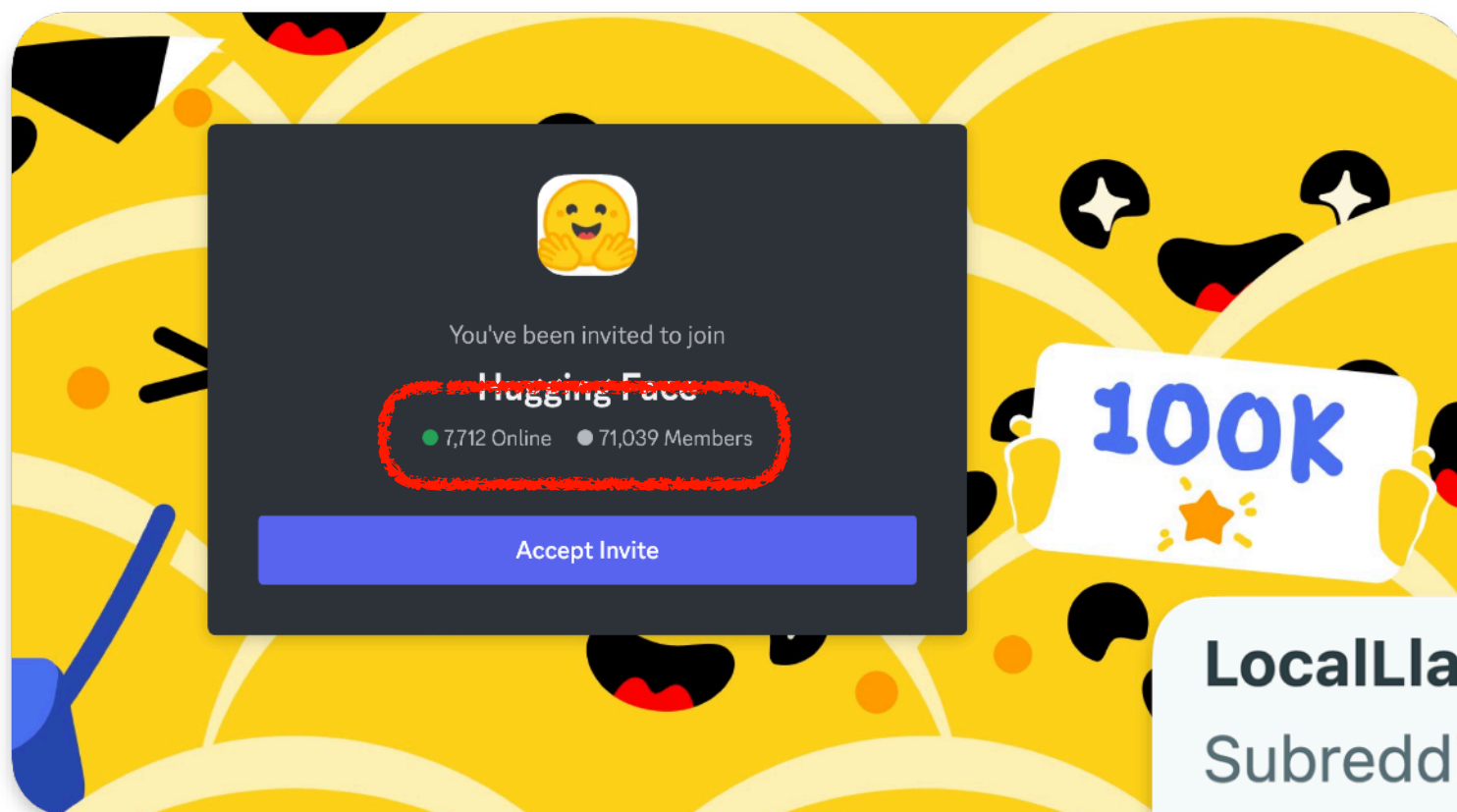
Source: Hugging Face discord server login page,
<https://discord.com>

A dynamic ecosystem



Source: Hugging Face discord server login page,
<https://discord.com>

A dynamic ecosystem



Source: Hugging Face discord server login page,
<https://discord.com>

LocalLlama

Subreddit to discuss about Llama, the large language model created by Meta AI.

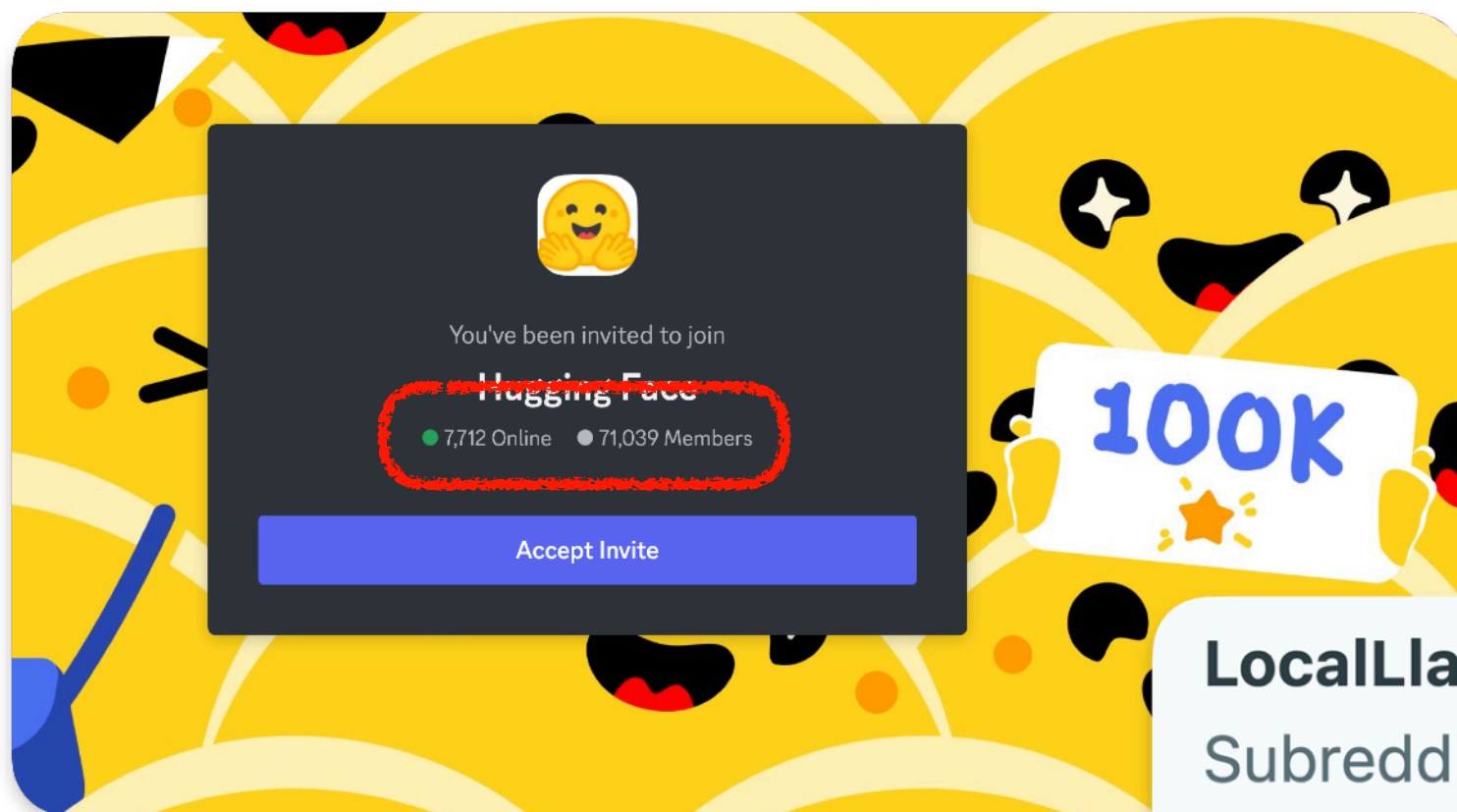
120K
Members

669
● Online

Top 2%
Rank by size ↗

Source: r/LocalLLaMA reddit,
<https://www.reddit.com/r/LocalLLaMA/>

A dynamic ecosystem



Source: Hugging Face discord server login page,
<https://discord.com>

LocalLlama

Subreddit to discuss about Llama, the large language model created by Meta AI.

120K
Members

669
● Online

Top 2%
Rank by size ↗

Source: r/LocalLLaMA reddit,
<https://www.reddit.com/r/LocalLLaMA/>

Agenda

1. Finetuning techniques

1. PEFT

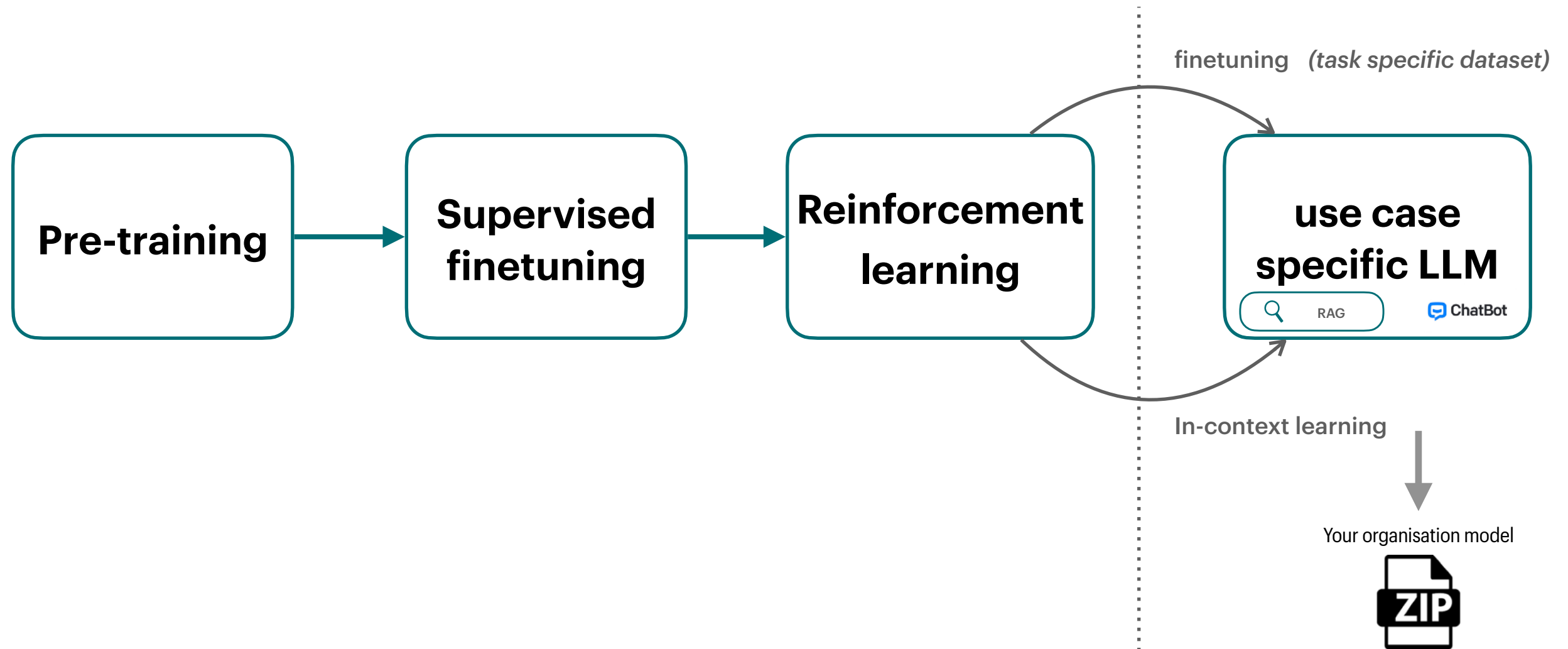
2. LoRA

2. Quantization

3. Overview of reinforcement learning

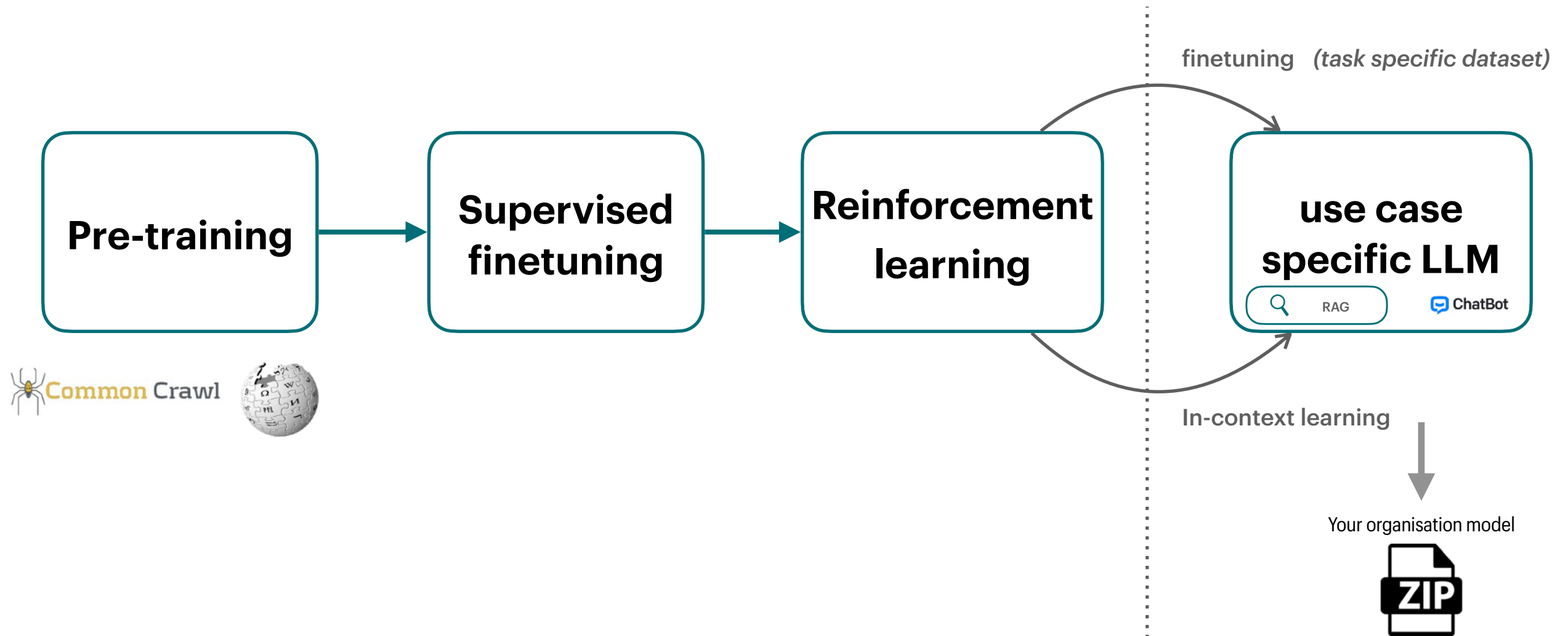
4. LLM systems eval. tips ← Very brief

LLM training pipeline



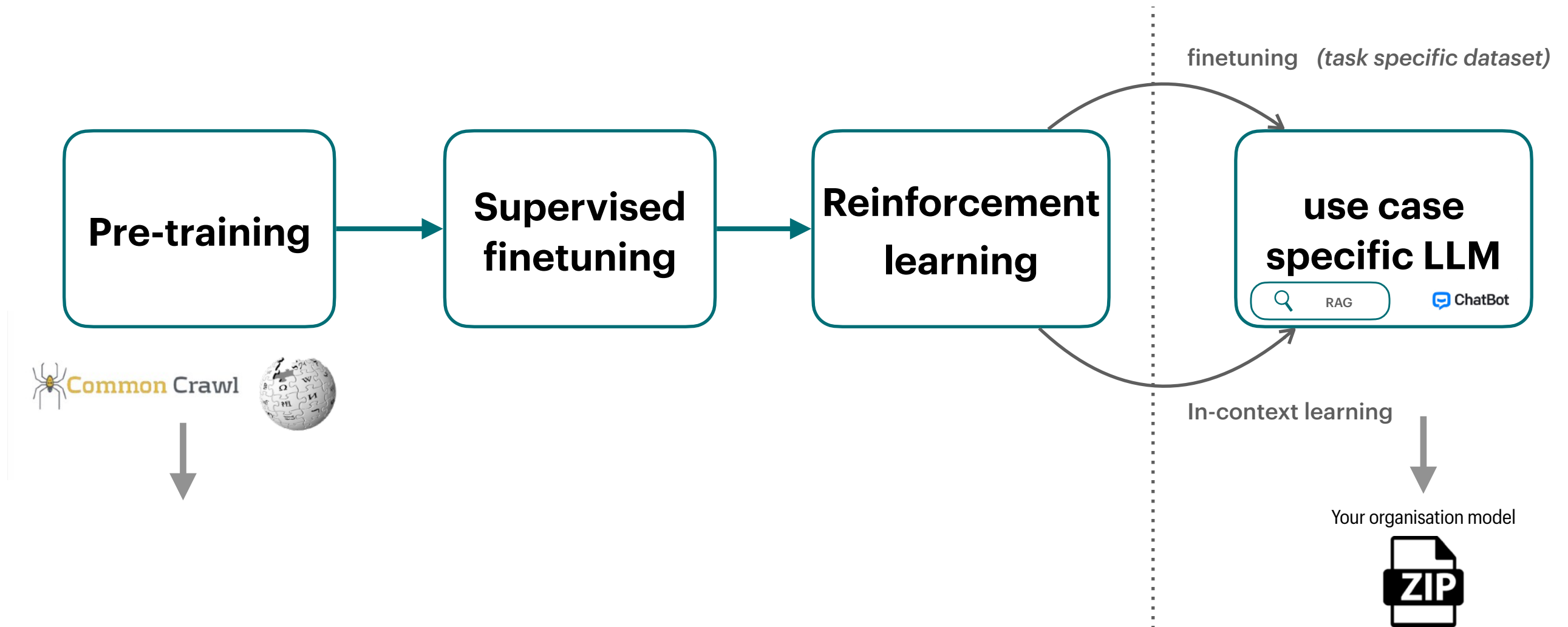
Adapted from Wolfe, C. (2023). Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More. Retrieved Feb 2024, <https://cameronwolfe.substack.com/p/easily-train-a-specialized-llm-peft>.

LLM training pipeline



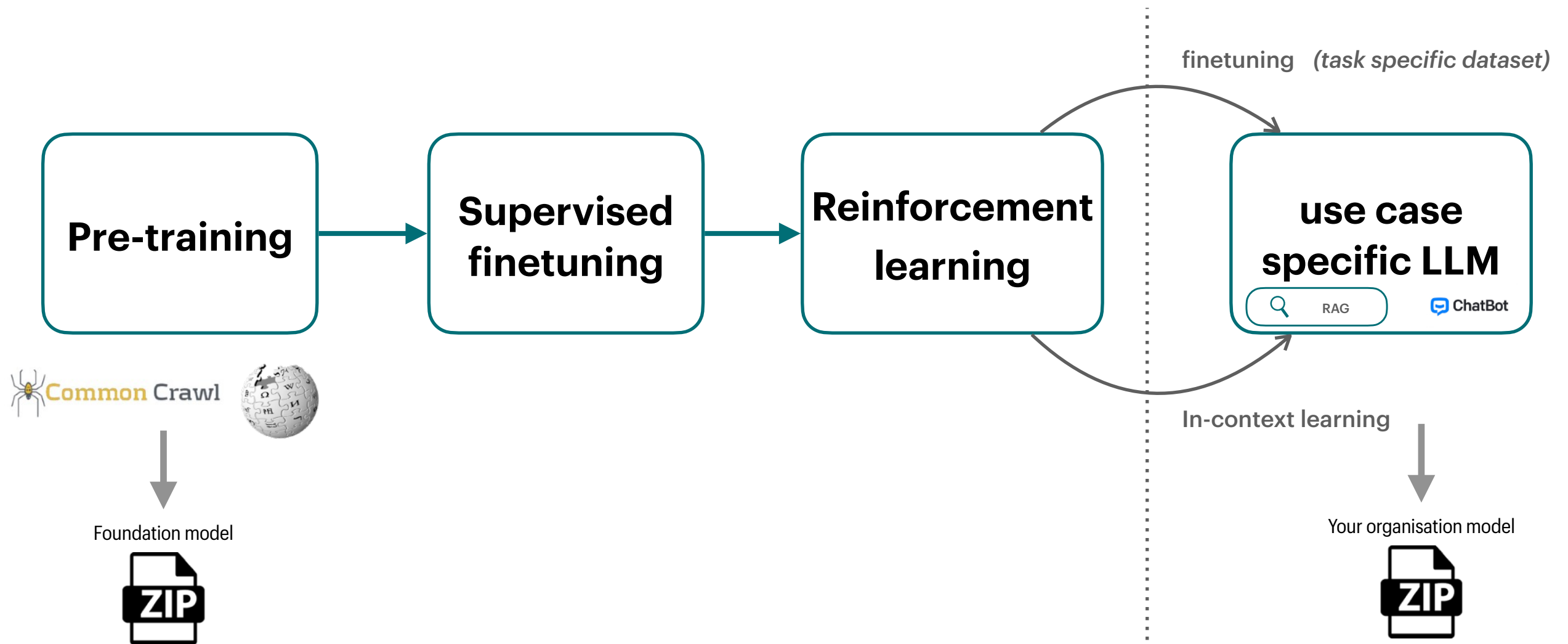
Adapted from Wolfe, C. (2023). Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More. Retrieved Feb 2024, <https://cameronwolfe.substack.com/p/easily-train-a-specialized-llm-peft>.

LLM training pipeline



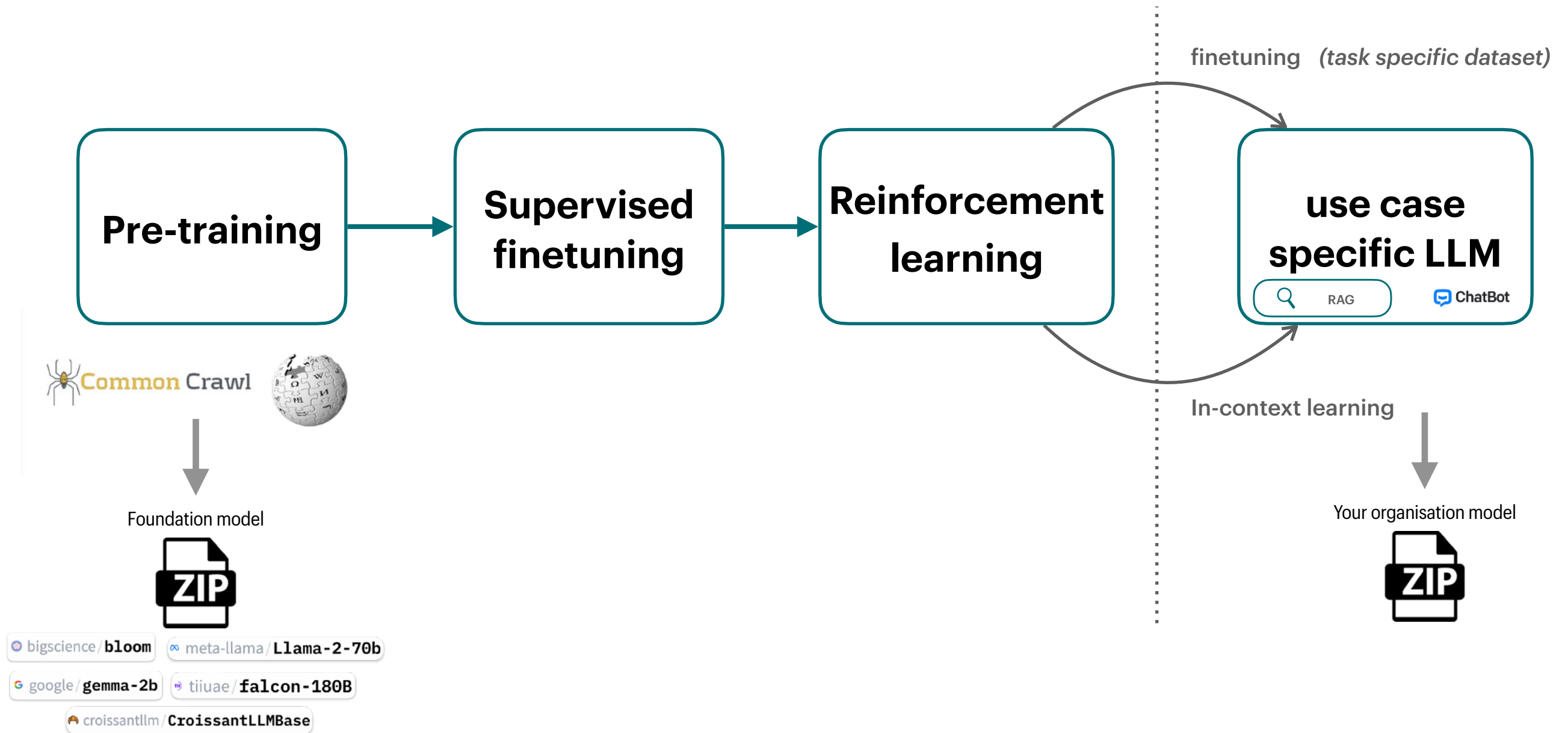
Adapted from Wolfe, C. (2023). Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More. Retrieved Feb 2024, <https://cameronwolfe.substack.com/p/easily-train-a-specialized-llm-peft>.

LLM training pipeline



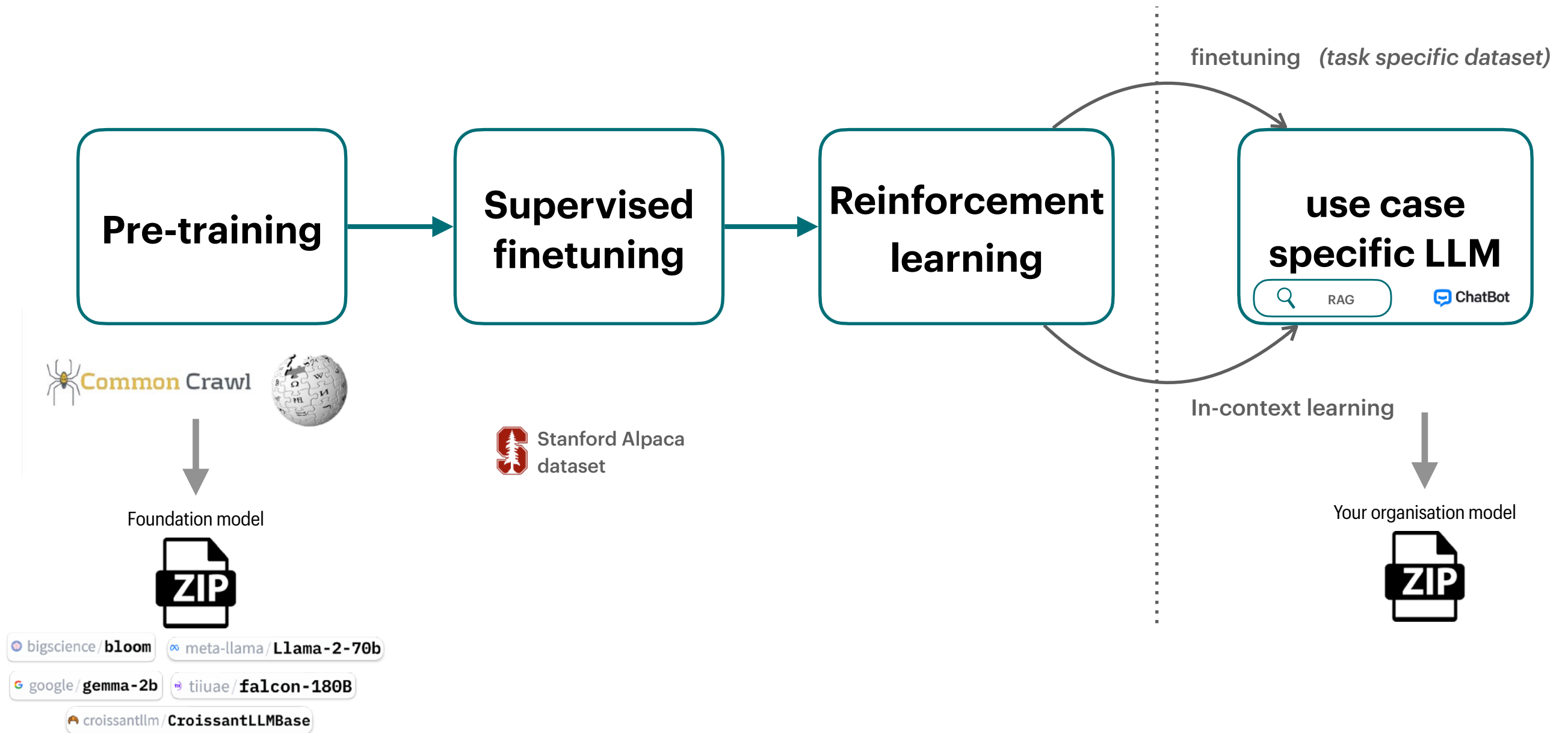
Adapted from Wolfe, C. (2023). Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More. Retrieved Feb 2024, <https://cameronwolfe.substack.com/p/easily-train-a-specialized-llm-peft>.

LLM training pipeline



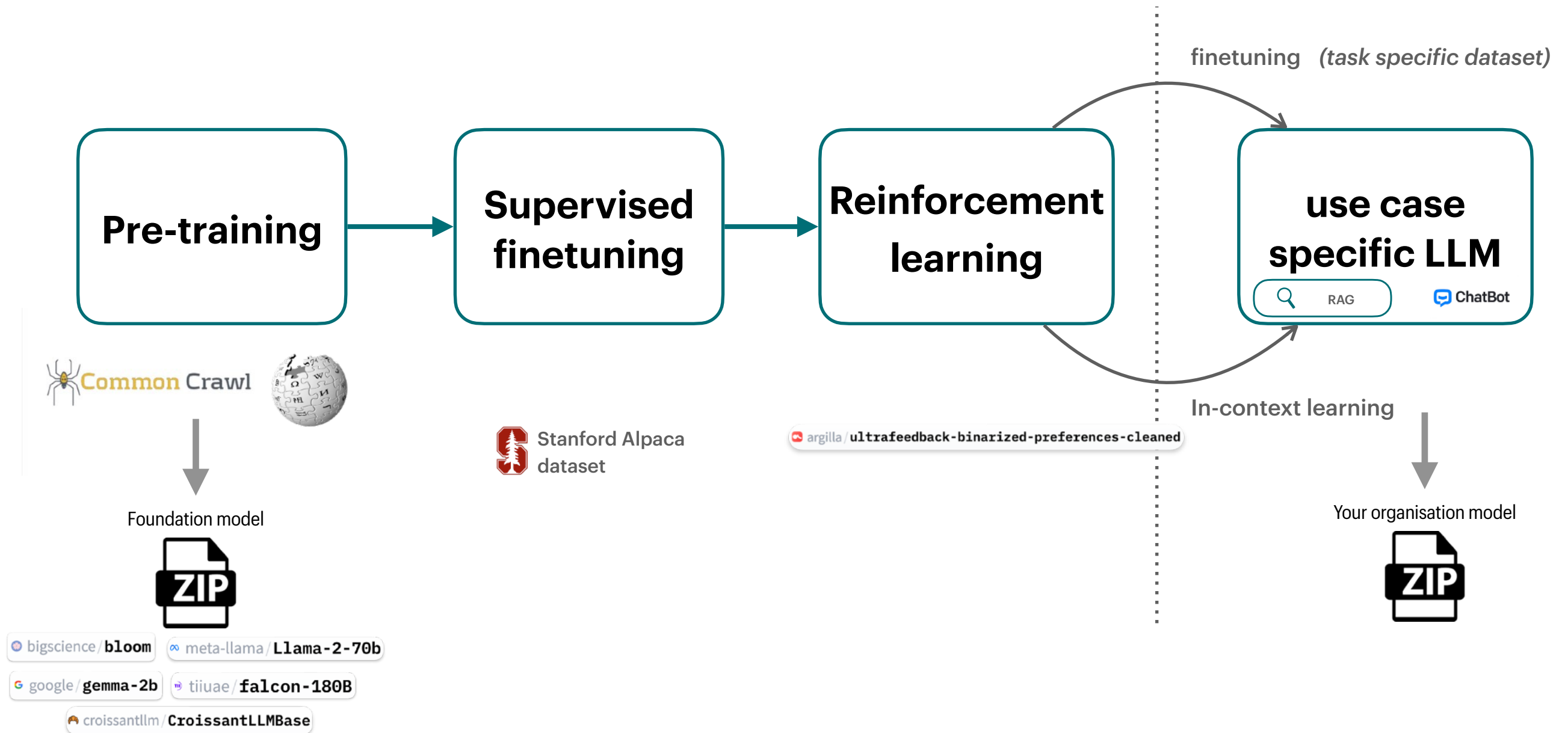
Adapted from Wolfe, C. (2023). Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More. Retrieved Feb 2024, <https://cameronwolfe.substack.com/p/easily-train-a-specialized-llm-peft>.

LLM training pipeline



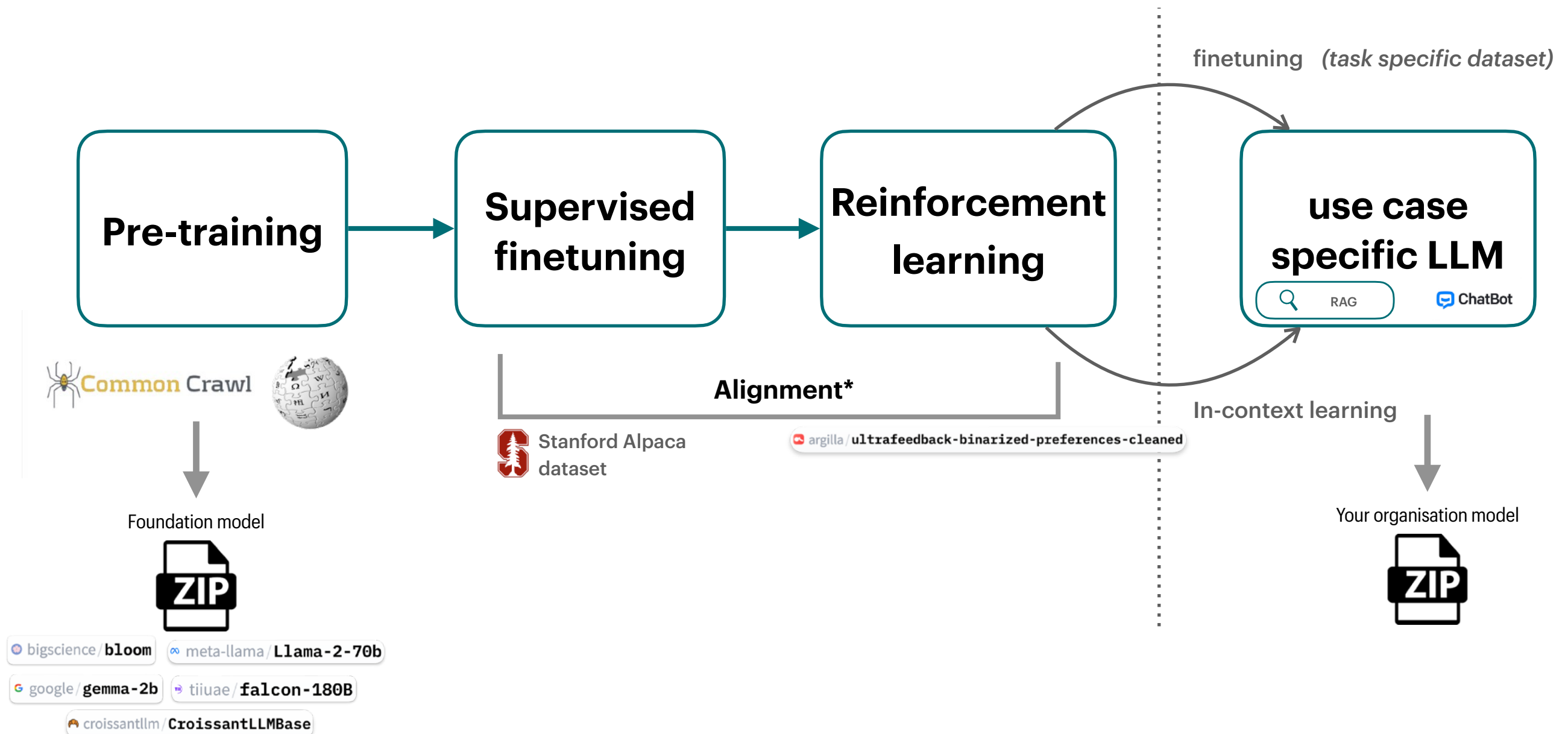
Adapted from Wolfe, C. (2023). Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More. Retrieved Feb 2024, <https://cameronwolfe.substack.com/p/easily-train-a-specialized-llm-peft>.

LLM training pipeline



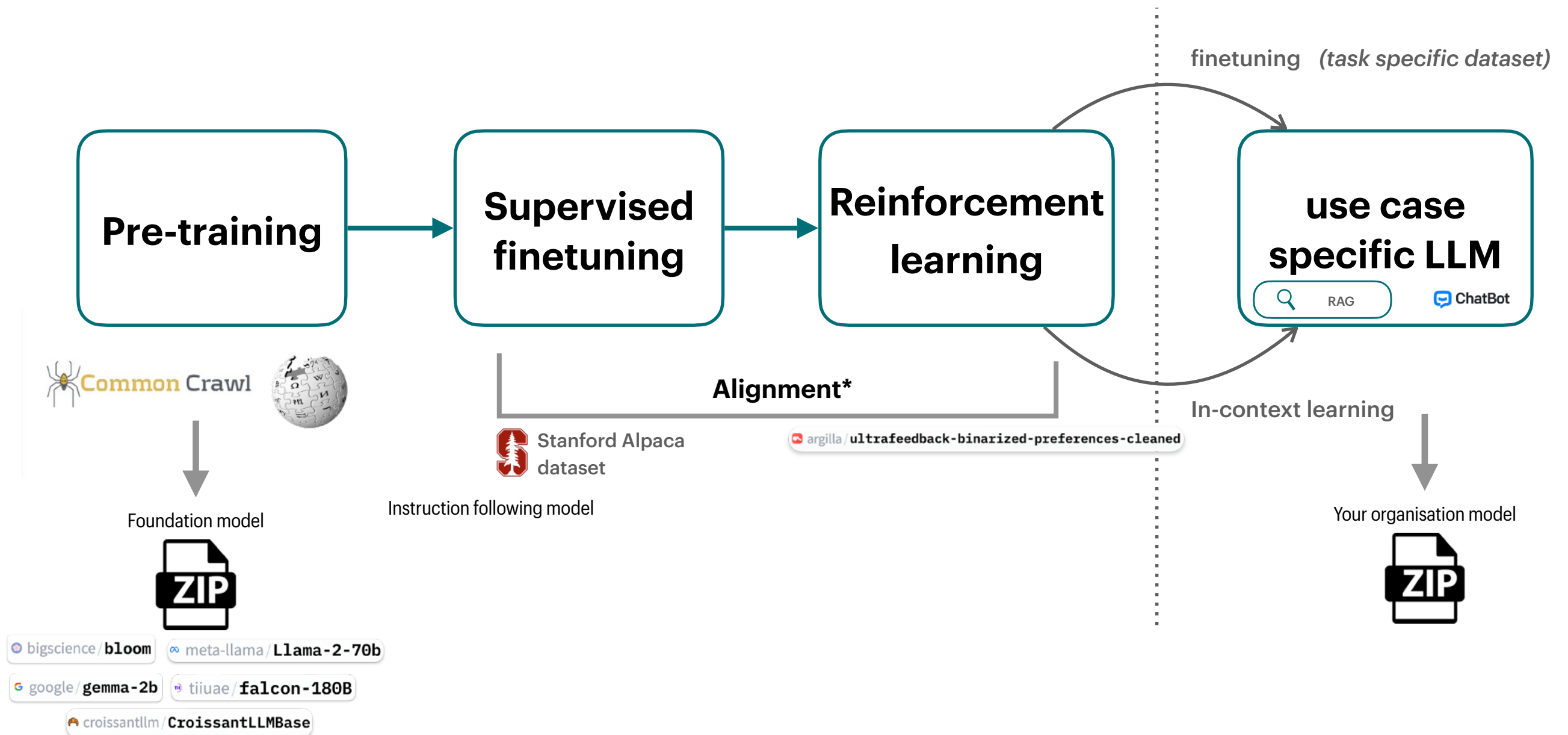
Adapted from Wolfe, C. (2023). Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More. Retrieved Feb 2024, <https://cameronwolfe.substack.com/p/easily-train-a-specialized-llm-peft>.

LLM training pipeline



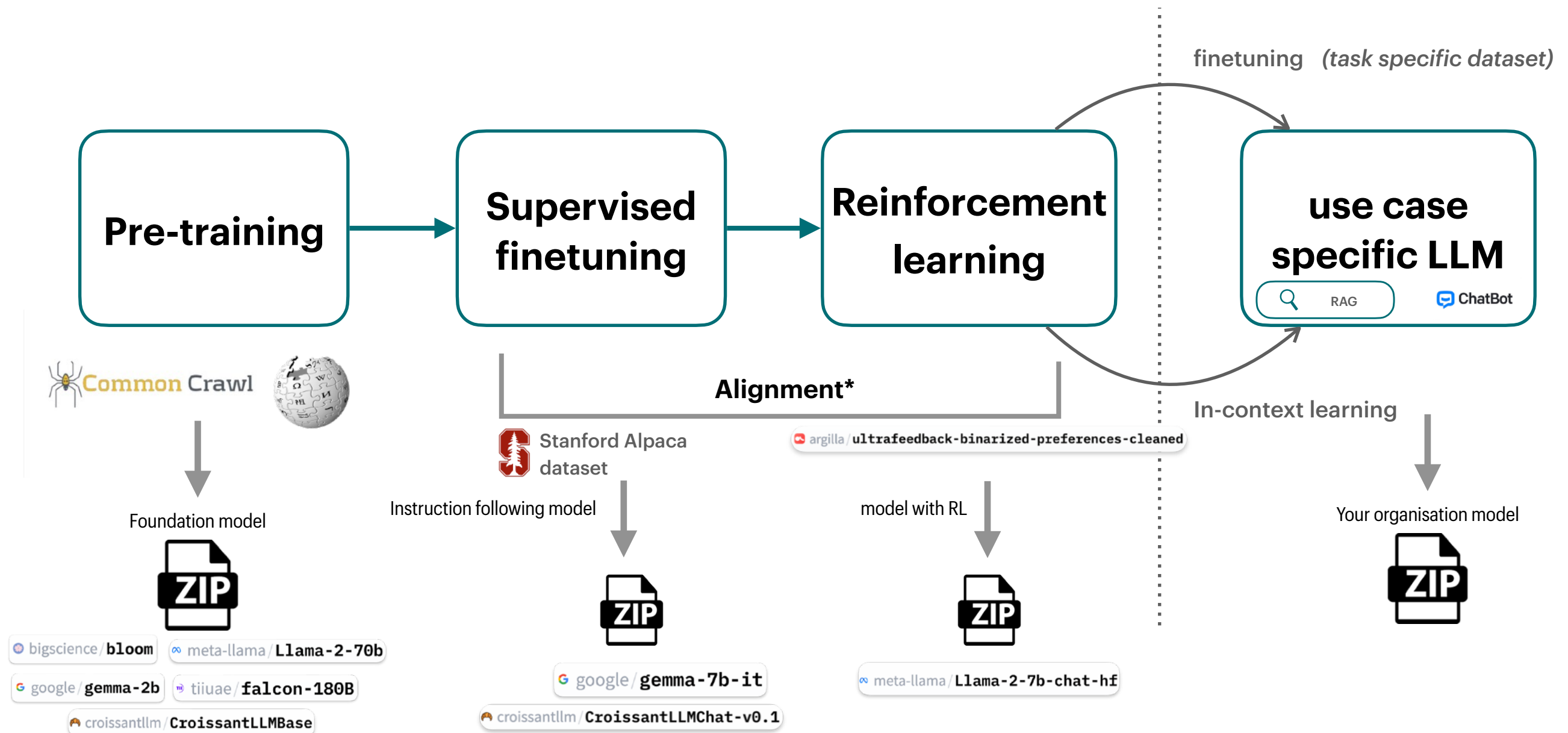
Adapted from Wolfe, C. (2023). Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More. Retrieved Feb 2024, <https://cameronwolfe.substack.com/p/easily-train-a-specialized-llm-peft>.

LLM training pipeline



Adapted from Wolfe, C. (2023). Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More. Retrieved Feb 2024, <https://cameronwolfe.substack.com/p/easily-train-a-specialized-llm-peft>.

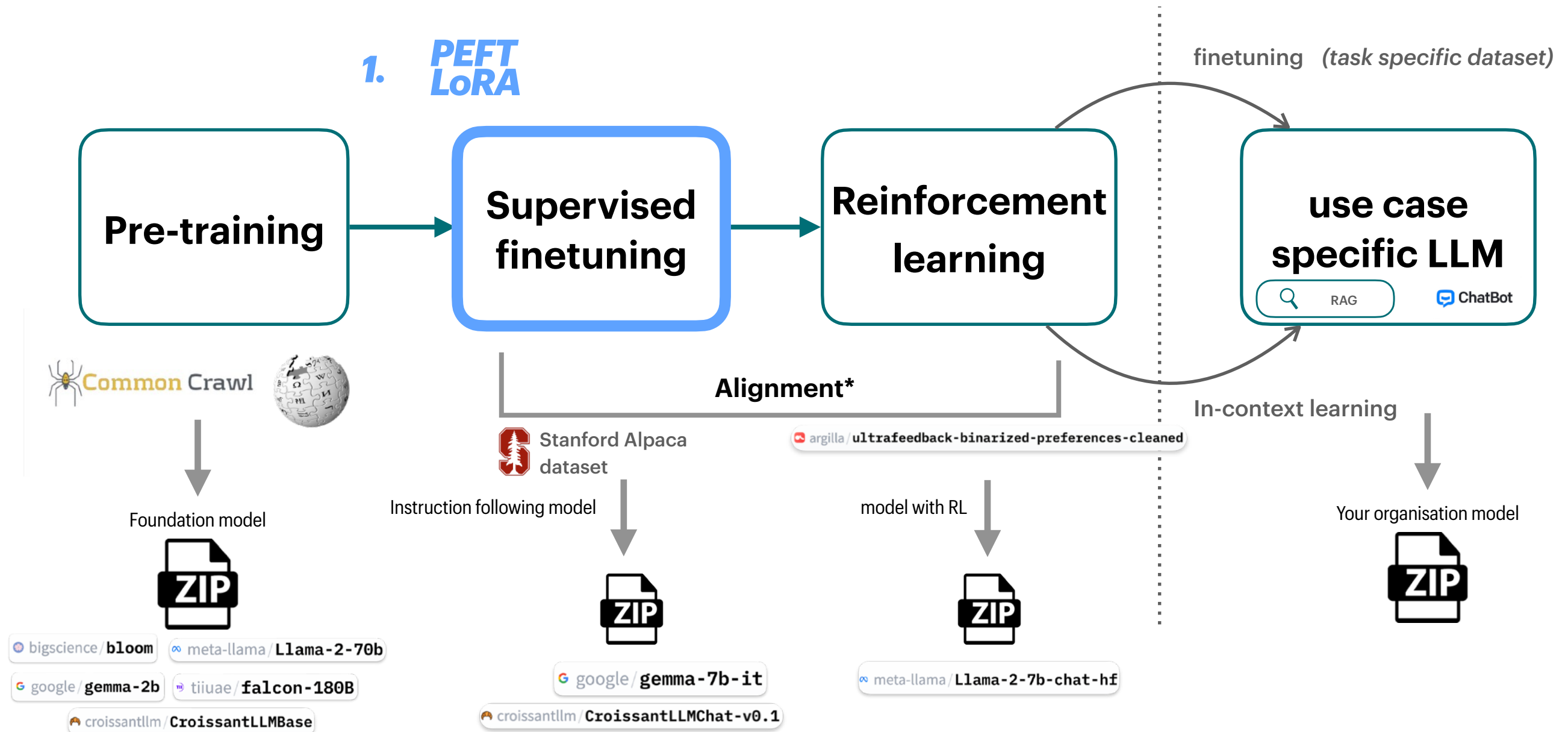
LLM training pipeline



Adapted from Wolfe, C. (2023). Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More. Retrieved Feb 2024, <https://cameronwolfe.substack.com/p/easily-train-a-specialized-llm-peft>.

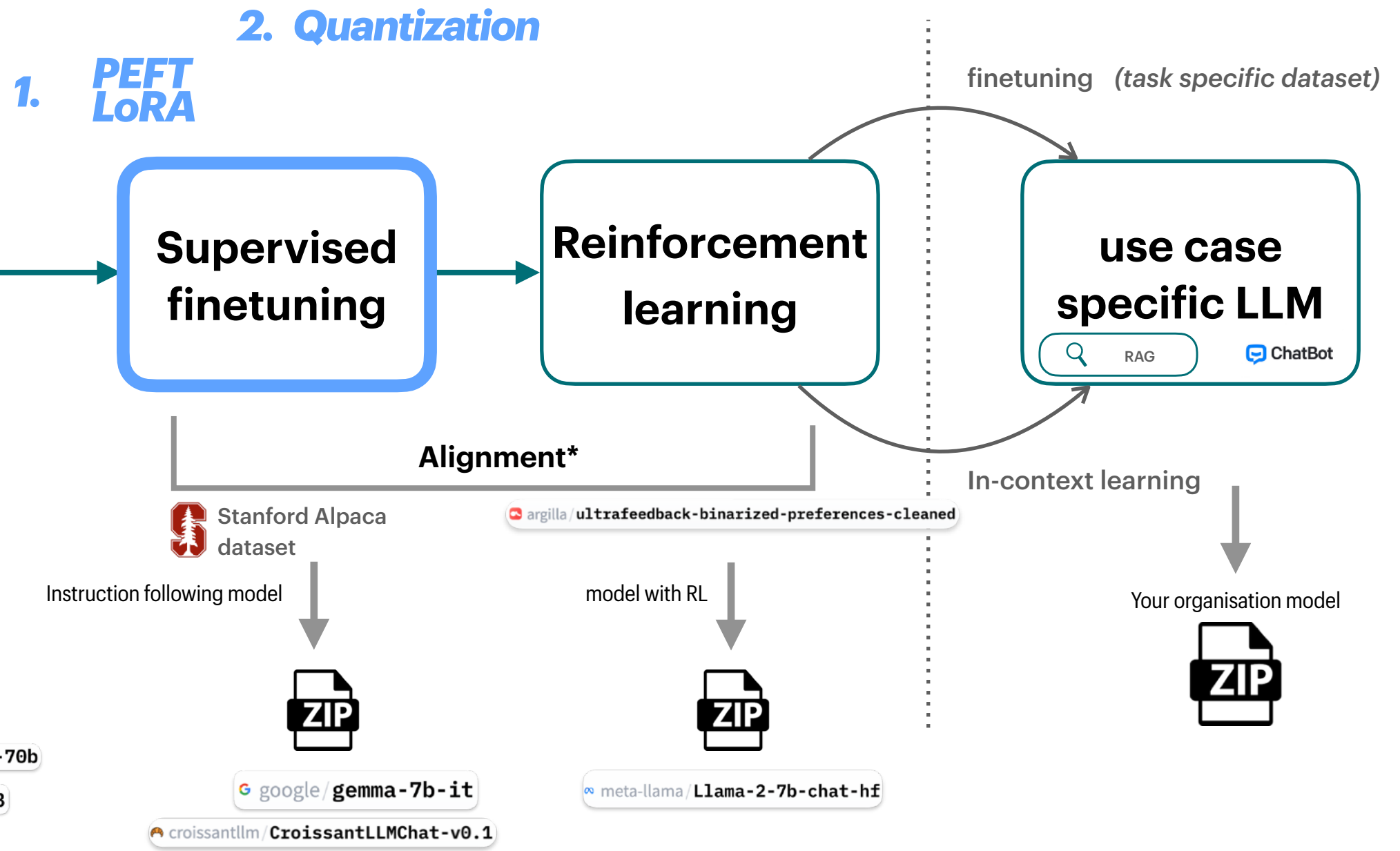
LLM training pipeline

1. PEFT
LoRA



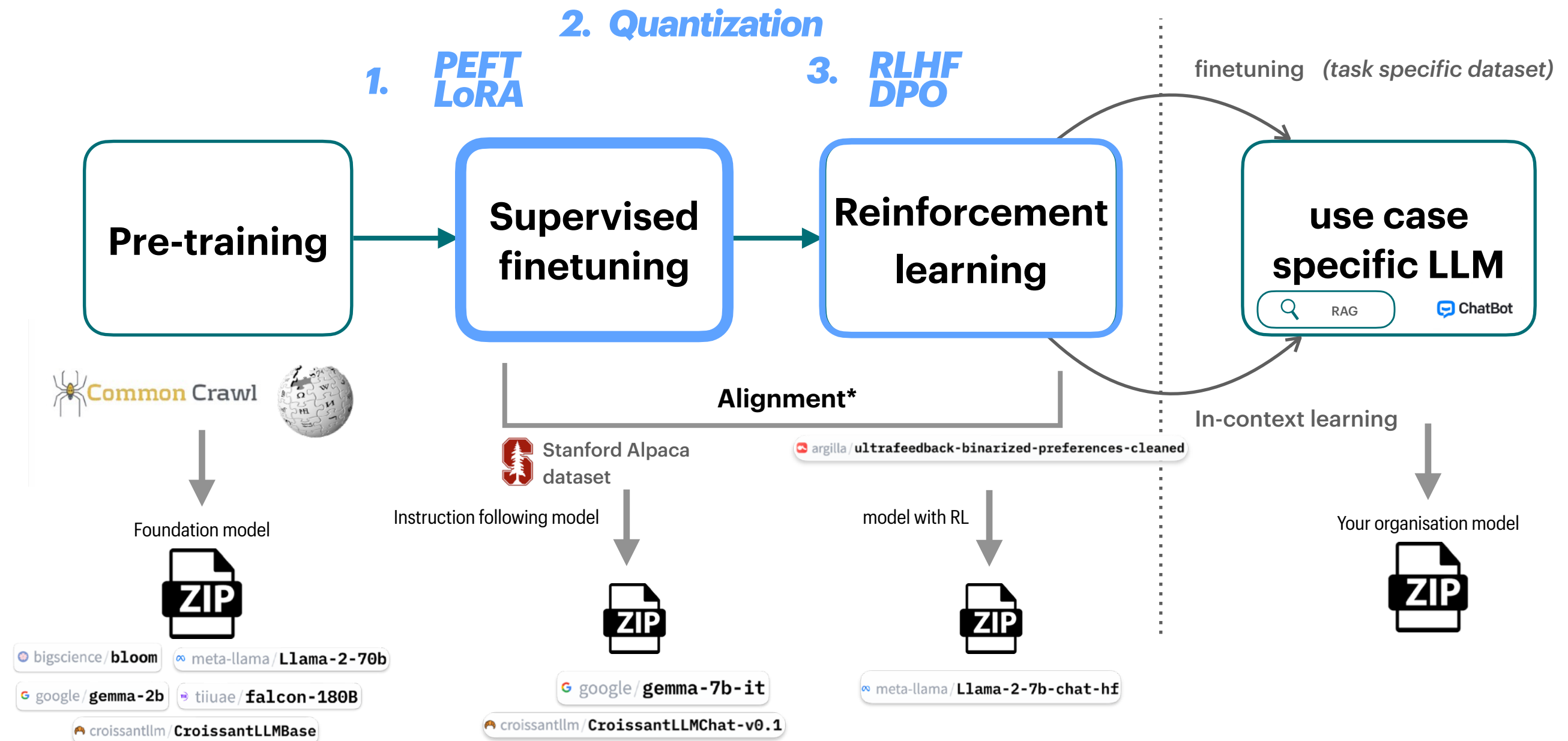
Adapted from Wolfe, C. (2023). Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More. Retrieved Feb 2024, <https://cameronwolfe.substack.com/p/easily-train-a-specialized-llm-peft>.

LLM training pipeline



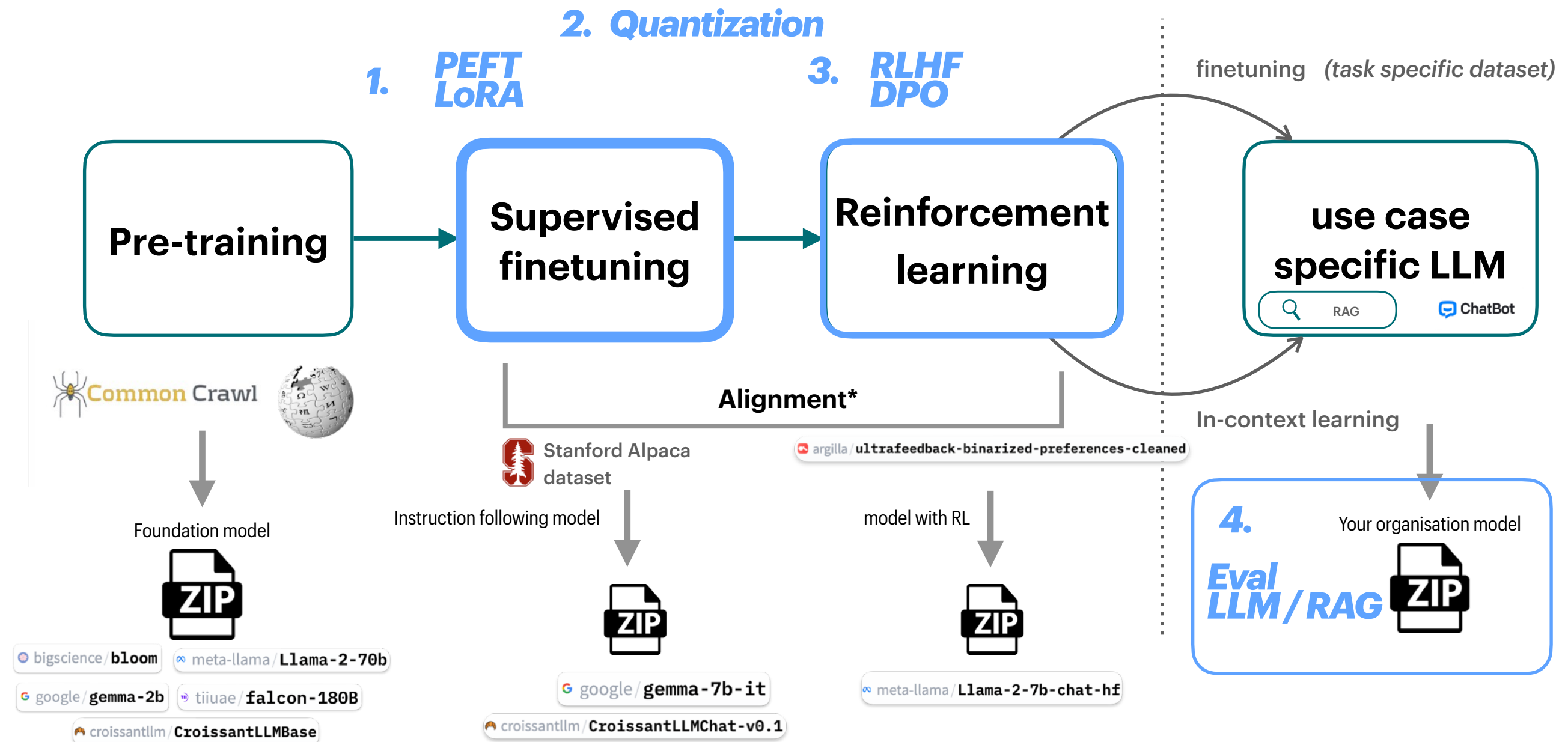
Adapted from Wolfe, C. (2023). Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More. Retrieved Feb 2024, <https://cameronwolfe.substack.com/p/easily-train-a-specialized-llm-peft>.

LLM training pipeline



Adapted from Wolfe, C. (2023). Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More. Retrieved Feb 2024, <https://cameronwolfe.substack.com/p/easily-train-a-specialized-llm-peft>.

LLM training pipeline



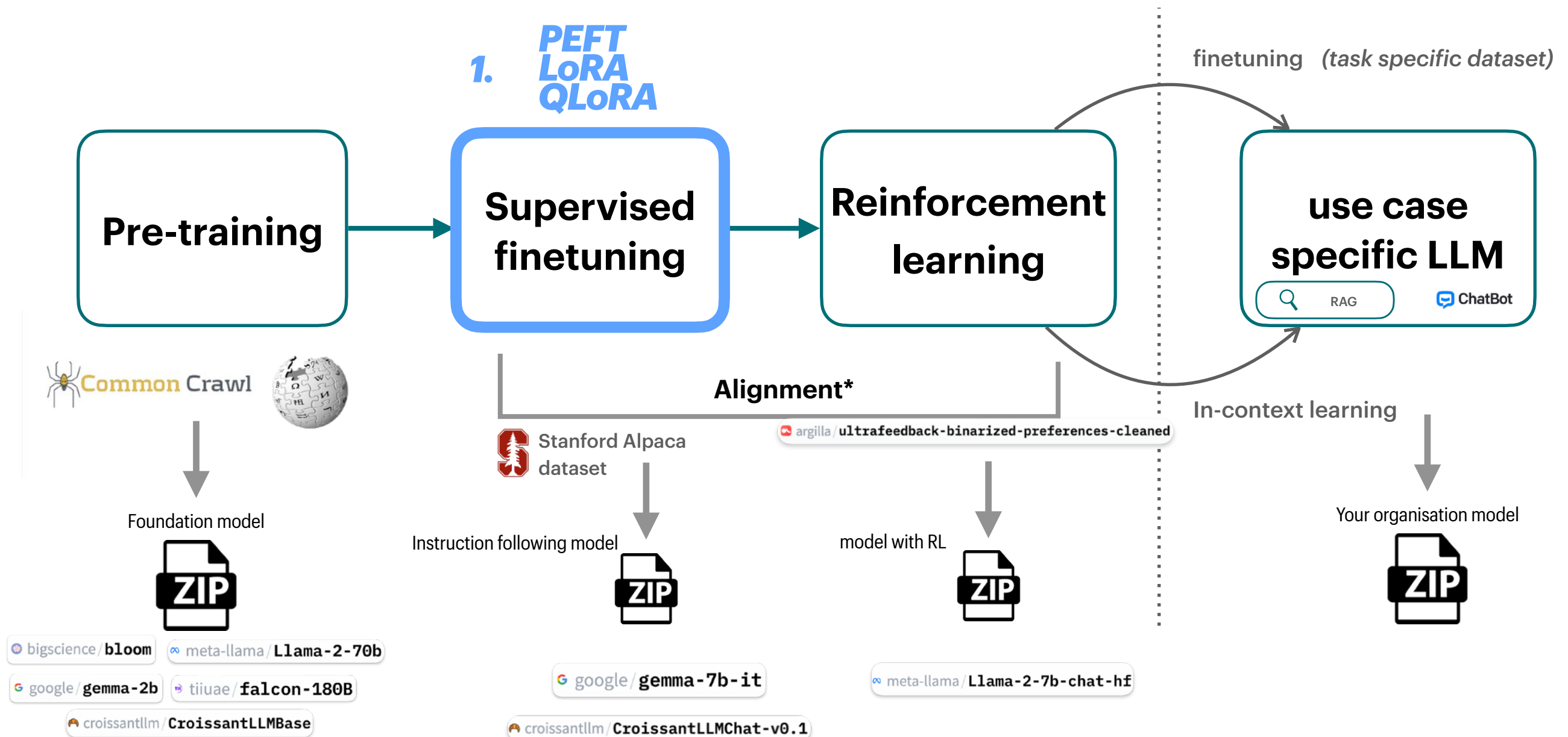
Adapted from Wolfe, C. (2023). Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More. Retrieved Feb 2024, <https://cameronwolfe.substack.com/p/easily-train-a-specialized-llm-peft>.

Pt 1.

Finetuning techniques

Parameter efficient finetuning (PEFT), Low-Rank Adaptation (LoRA)

LLM training pipeline



Adapted from Wolfe, C. (2023). Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More. Retrieved Feb 2024, <https://cameronwolfe.substack.com/p/easily-train-a-specialized-llm-peft>.

LLM Full finetuning is prohibitively expensive

- Compute and memory intensive (update all params)
- Risk of forgetting previously learned information (catastrophic forgetting)
- Extensive set of hyper parameters to explore
- Need to retrain & store a model for each new task (complex model management)

LLM Full finetuning is prohibitively expensive

- Compute and memory intensive (update all params)
- Risk of forgetting previously learned information (catastrophic forgetting)
- Extensive set of hyper parameters to explore
- Need to retrain & store a model for each new task (complex model management)

...so how to fine tune efficiently ?

LLM Full finetuning is prohibitively expensive

- Compute and memory intensive (update all params)
- Risk of forgetting previously learned information (catastrophic forgetting)
- Extensive set of hyper parameters to explore
- Need to retrain & store a model for each new task (complex model management)

...so how to fine tune efficiently ?

LLM Full finetuning is prohibitively expensive

- Compute and memory intensive (update all params)
- Risk of forgetting previously learned information (catastrophic forgetting)
- Extensive set of hyper parameters to explore
- Need to retrain & store a model for each new task (complex model management)

...so how to fine tune efficiently ?

Looking for finetuning techniques that are:

LLM Full finetuning is prohibitively expensive

- Compute and memory intensive (update all params)
- Risk of forgetting previously learned information (catastrophic forgetting)
- Extensive set of hyper parameters to explore
- Need to retrain & store a model for each new task (complex model management)

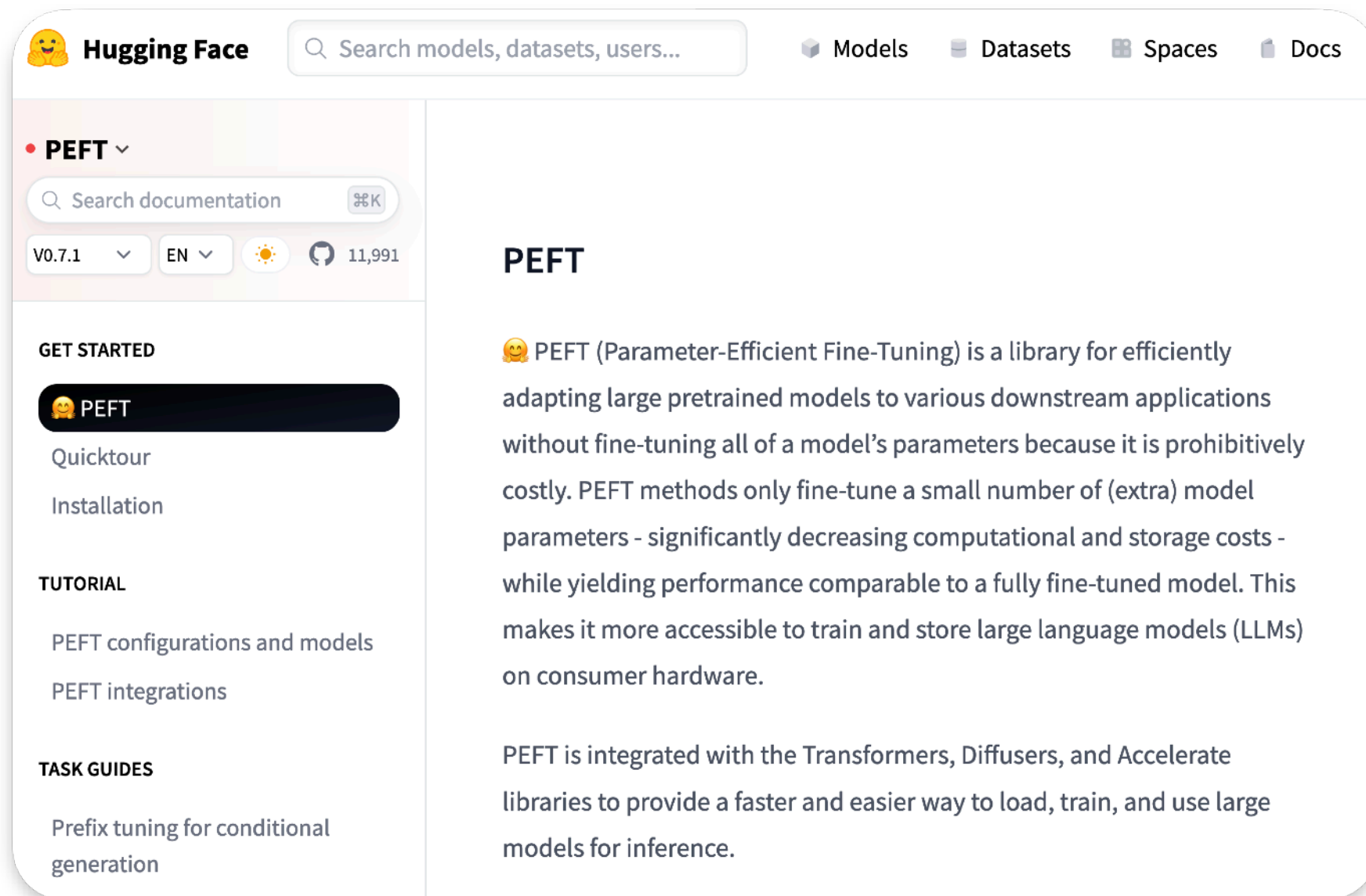
...so how to fine tune efficiently ?

Looking for finetuning techniques that are:

- Fast to train**
- Run on affordable hardware**
- Easy to deploy**

Parameter efficient finetuning

Only adapting a 'small' numbers of parameters - PEFT



The screenshot shows the Hugging Face website interface for the PEFT (Parameter-Efficient Fine-Tuning) documentation. The top navigation bar includes the Hugging Face logo, a search bar, and links for Models, Datasets, Spaces, and Docs. The left sidebar is titled 'PEFT' and contains a search bar, version information (V0.7.1), language (EN), and a count of 11,991 items. Below this, there are sections for 'GET STARTED' (with a highlighted 'PEFT' button), 'TUTORIAL' (with links for 'PEFT configurations and models' and 'PEFT integrations'), and 'TASK GUIDES' (with a link for 'Prefix tuning for conditional generation'). The main content area is titled 'PEFT' and contains a paragraph describing the library's purpose and a section for integration with other libraries.

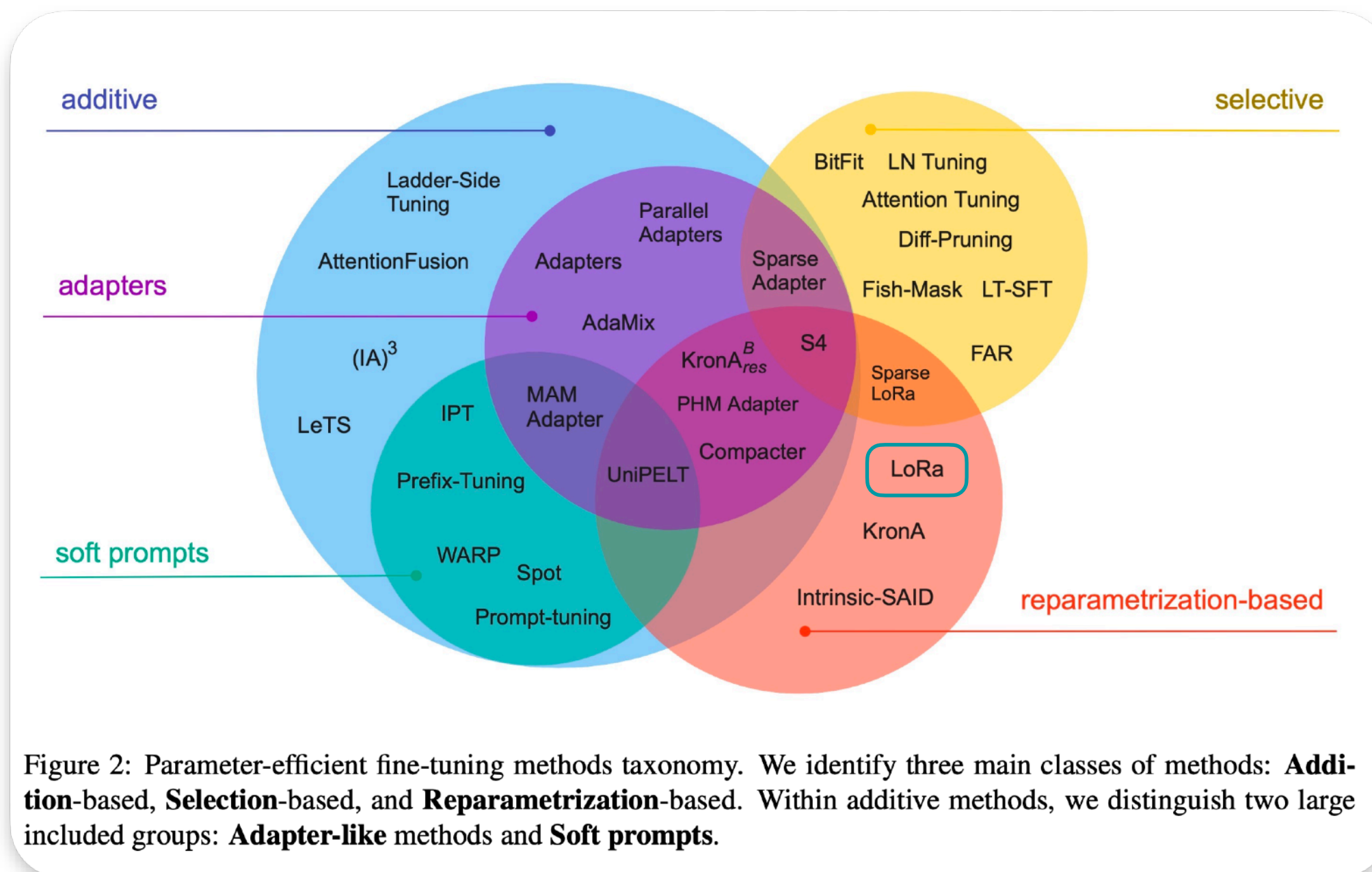
PEFT

🤗 PEFT (Parameter-Efficient Fine-Tuning) is a library for efficiently adapting large pretrained models to various downstream applications without fine-tuning all of a model's parameters because it is prohibitively costly. PEFT methods only fine-tune a small number of (extra) model parameters - significantly decreasing computational and storage costs - while yielding performance comparable to a fully fine-tuned model. This makes it more accessible to train and store large language models (LLMs) on consumer hardware.

PEFT is integrated with the Transformers, Diffusers, and Accelerate libraries to provide a faster and easier way to load, train, and use large models for inference.

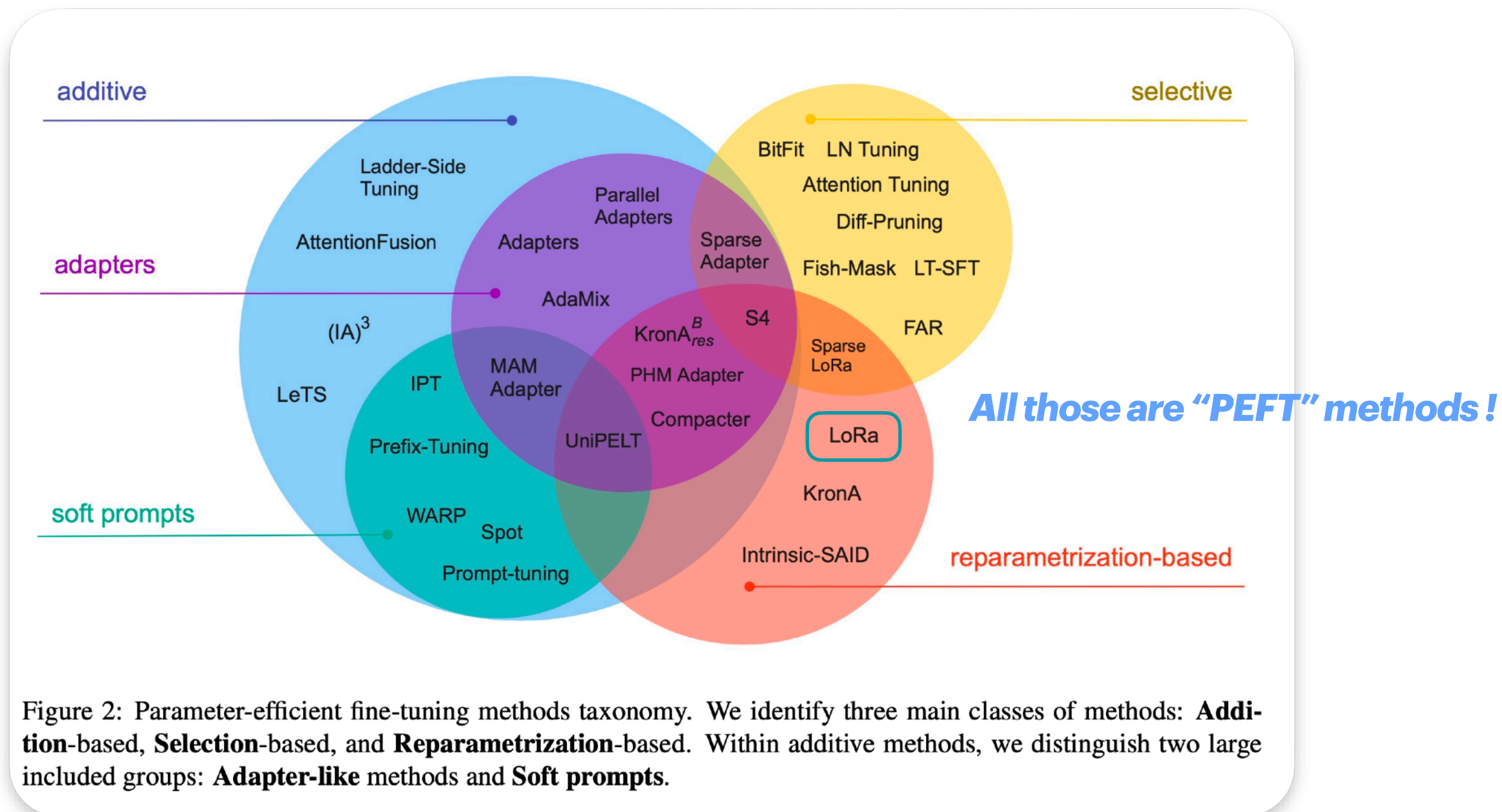
Source: Hugging Face PEFT module, <https://huggingface.co/docs/peft/en/index>

PEFT is an active field of research



Source: Vladislav, L., Vijeta, D., & Anna, R. (2023). Scaling Down to Scale Up: A Guide to Parameter-Efficient Fine-Tuning.

PEFT is an active field of research



Source: Vladislav, L., Vijeta, D., & Anna, R. (2023). Scaling Down to Scale Up: A Guide to Parameter-Efficient Fine-Tuning.

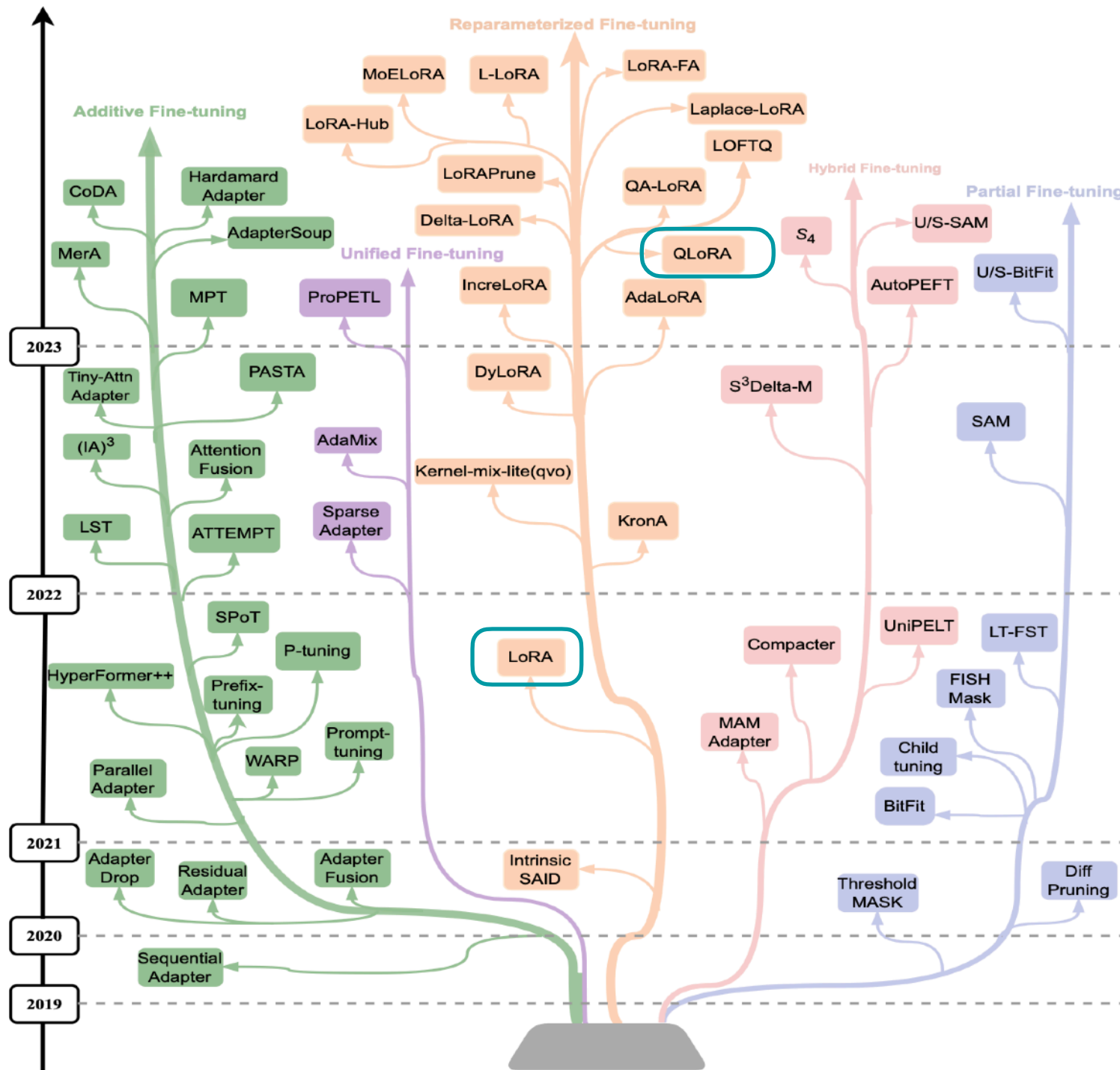


Fig. 1: The evolutionary development of PEFT methods in recent years. Models on the same branch have some common features. The vertical position of the models shows the timeline of their release dates. Notably, the year of the paper's initial publication is shown as the reference. For instance, if a paper is published in ACL 2022 but listed on arXiv in 2021, the year 2021 will be considered as the reference date.

Source: Lingling, X., Haoran, X., Si-Zhao, J. Q., Xiaohui, T., & Fu, L. W. (2023). Parameter-Efficient Fine-Tuning Methods for Pretrained Language Models: A Critical Review and Assessment.

Focus on Low-Rank Adaptation (LoRA)

Why LoRA?

- Can be finetuned on commodity hardware
- Small(er) numbers of parameters to train
- Well documented and several implementations
- Lightweight & easy to switch at runtime

Focus on Low-Rank Adaptation (LoRA)

Why LoRA?

- Can be finetuned on commodity hardware
- Small(er) numbers of parameters to train
- Well documented and several implementations
- Lightweight & easy to switch at runtime

Looking for finetuning techniques that are:

Focus on Low-Rank Adaptation (LoRA)

Why LoRA?

- Can be finetuned on commodity hardware
- Small(er) numbers of parameters to train
- Well documented and several implementations
- Lightweight & easy to switch at runtime

Looking for finetuning techniques that are:

- Fast to train
- Run on affordable hardware
- Easy to deploy

Intuition on LoRA

Introducing “*intrinsic dimension*” - High performance with a few parameters in a sub-dimension of initial parameter space

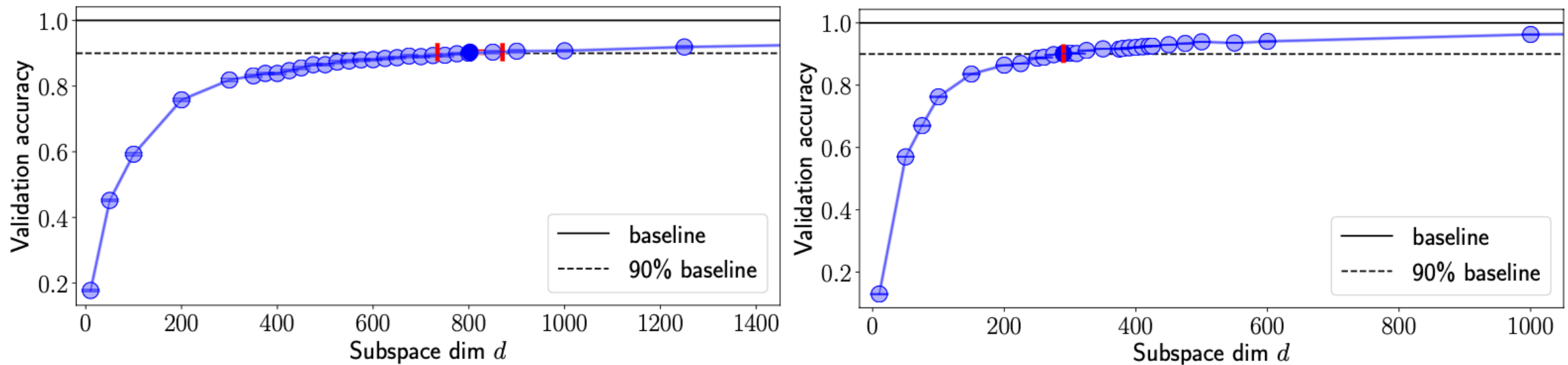
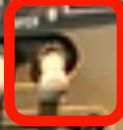
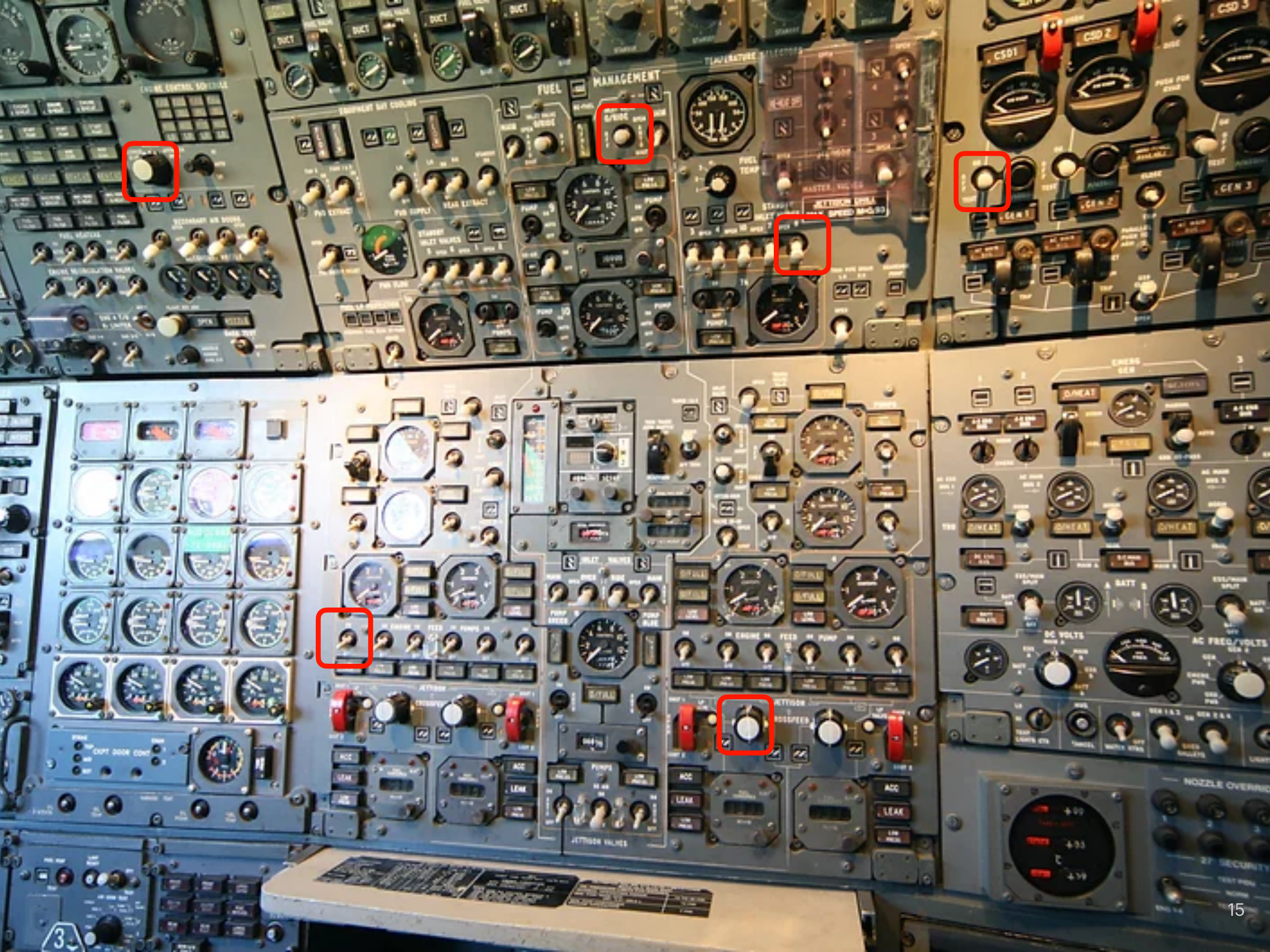


Figure 2: Performance (validation accuracy) vs. subspace dimension d for two networks trained on MNIST: **(left)** a 784–200–200–10 fully-connected (FC) network ($D = 199,210$) and **(right)** a convolutional network, LeNet ($D = 44,426$). The solid line shows performance of a well-trained direct (FC or conv) model, and the dashed line shows the 90% threshold we use to define d_{int90} . The standard derivation of validation accuracy and measured d_{int90} are visualized as the blue vertical and red horizontal error bars. We oversample the region around the threshold to estimate the dimension of crossing more exactly. We use one-run measurements for d_{int90} of 750 and 290, respectively.

Source: Chunyuan, L., Heerad, F., Rosanne, L., & Jason, Y. (2018). *Measuring the Intrinsic Dimension of Objective Landscapes*.





INTRINSIC DIMENSIONALITY EXPLAINS THE EFFECTIVENESS OF LANGUAGE MODEL FINE-TUNING

Armen Aghajanyan, Luke Zettlemoyer, Sonal Gupta

Facebook

{armenag, lsz, sonalgupta}@fb.com

ABSTRACT

Although pretrained language models can be fine-tuned to produce state-of-the-art results for a very wide range of language understanding tasks, the dynamics of this process are not well understood, especially in the low data regime. Why can we use relatively vanilla gradient descent algorithms (e.g., without strong regularization) to tune a model with hundreds of millions of parameters on datasets with only hundreds or thousands of labeled examples? In this paper, we argue that analyzing fine-tuning through the lens of intrinsic dimension provides us with empirical and theoretical intuitions to explain this remarkable phenomenon. We empirically show that common pre-trained models have a very low intrinsic dimension; in other words, there exists a low dimension reparameterization that is as effective for fine-tuning as the full parameter space. For example, by optimizing only 200 trainable parameters randomly projected back into the full space, we can tune a RoBERTa model to achieve 90% of the full parameter performance levels on MRPC. Furthermore, we empirically show that pre-training implicitly minimizes intrinsic dimension and, perhaps surprisingly, larger models tend to have lower intrinsic dimension after a fixed number of pre-training updates, at least in part explaining their extreme effectiveness. Lastly, we connect intrinsic dimensionality with low dimensional task representations and compression based generalization bounds to provide intrinsic-dimension-based generalization bounds that are independent of the full parameter count.

Source: Armen, A., Luke, Z., & Sonal, G. (2020). Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning.

Why finetuning works well on large language model?

Intrinsic dimension

5 INTRINSIC DIMENSION, PRE-TRAINING, AND GENERALIZATION GAP

One interpretation of the intrinsic parameter vector is that it encodes the task at hand with respect to the original pre-trained representations. Therefore, we can interpret d as the minimal description length of the task within the framework dictated by the pre-trained representations (Hinton & Zemel, 1993). Under this interpretation of intrinsic dimensionality, we hypothesize that pre-training is implicitly lowering the intrinsic dimensionality of the average NLP task, and therefore compress the minimal description length of those same tasks.

Source: Armen, A., Luke, Z., & Sonal, G. (2020). Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning.

Why finetuning works well on large language model?

Intrinsic dimension

5 INTRINSIC DIMENSION, PRE-TRAINING, AND GENERALIZATION GAP

One interpretation of the intrinsic parameter vector is that it encodes the task at hand with respect to the original pre-trained representations. Therefore, we can interpret d as the minimal description length of the task within the framework dictated by the pre-trained representations (Hinton & Zemel, 1993). Under this interpretation of intrinsic dimensionality, we hypothesize that pre-training is implicitly lowering the intrinsic dimensionality of the average NLP task, and therefore compress the minimal description length of those same tasks.

For example, in the last section, we represented MRPC with roughly 200 parameters, which translates to needing less than a kilobyte of data to encode a complex natural language task within the framework provided by RoBERTa.

We hypothesize that the better the pre-trained models are, the fewer bits (description length) are needed to represent the average NLP task, as we will demonstrate empirically in the next section.

Source: Armen, A., Luke, Z., & Sonal, G. (2020). Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning.

The surprising power of few parameters

Large pretraining develop a compact & efficient representation of language

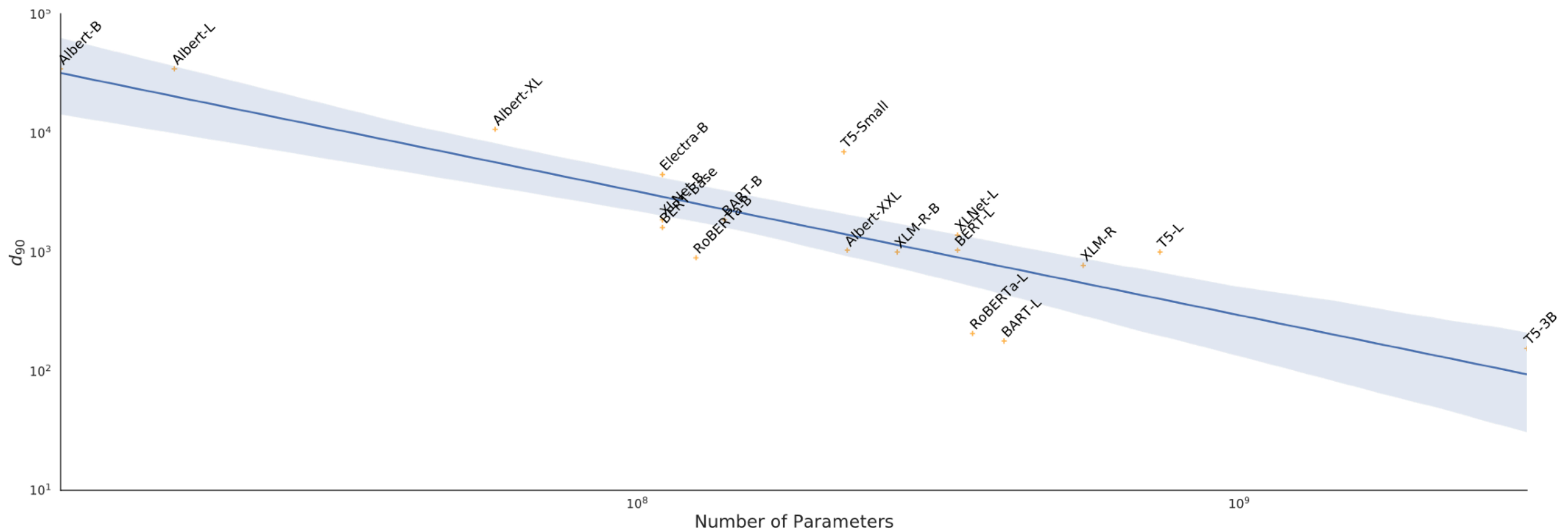


Figure 3: We calculate the intrinsic dimension for a large set of pre-trained models using the SAID method on the MRPC dataset.

Source: Armen, A., Luke, Z., & Sonal, G. (2020). Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning.

LoRA: LOW-RANK ADAPTATION OF LARGE LANGUAGE MODELS

Edward Hu* Yelong Shen* Phillip Wallis Zeyuan Allen-Zhu
Yuanzhi Li Shean Wang Lu Wang Weizhu Chen

Microsoft Corporation

{edwardhu, yeshe, phwallis, zeyuana,
yuanzhil, swang, luw, wzchen}@microsoft.com
yuanzhil@andrew.cmu.edu

(Version 2)

ABSTRACT

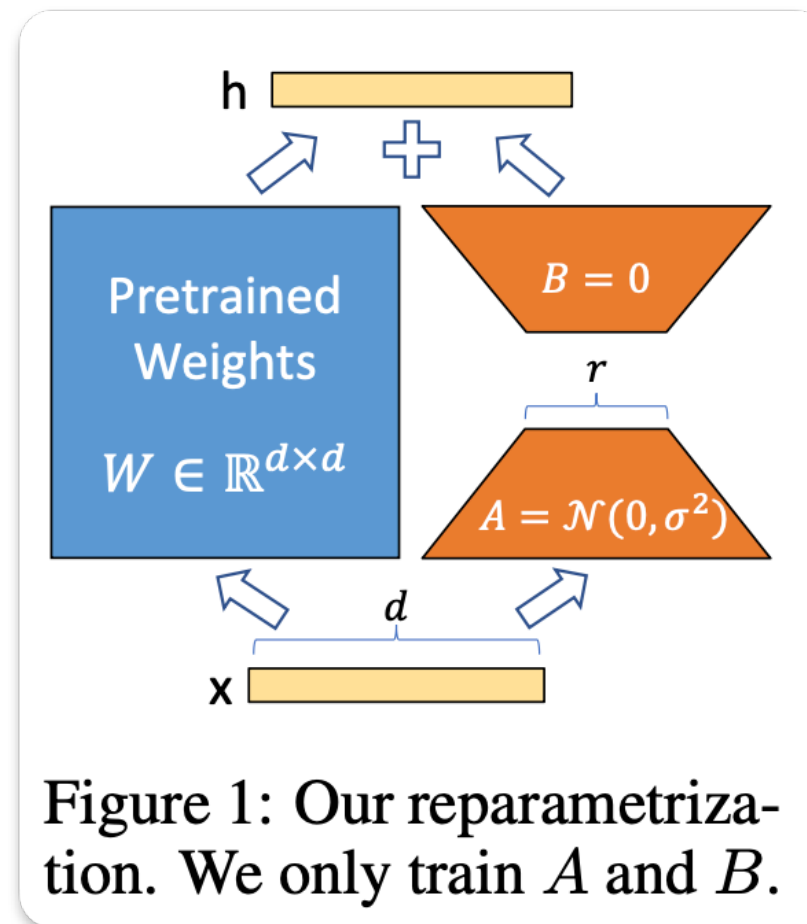
An important paradigm of natural language processing consists of large-scale pre-training on general domain data and adaptation to particular tasks or domains. As we pre-train larger models, full fine-tuning, which retrains all model parameters, becomes less feasible. Using GPT-3 175B as an example – deploying independent instances of fine-tuned models, each with 175B parameters, is prohibitively expensive. We propose **Low-Rank Adaptation**, or LoRA, which freezes the pre-trained model weights and injects **trainable rank decomposition matrices into each layer of the Transformer architecture**, greatly reducing the number of trainable parameters for downstream tasks. Compared to GPT-3 175B fine-tuned with Adam, LoRA can reduce the number of trainable parameters by **10,000 times and the GPU memory requirement by 3 times**. LoRA performs on-par or better than fine-tuning in model quality on RoBERTa, DeBERTa, GPT-2, and GPT-3, despite having fewer trainable parameters, a higher training throughput, and, unlike adapters, **no additional inference latency**. We also provide an empirical investigation into rank-deficiency in language model adaptation, which sheds light on the efficacy of LoRA. We release a package that facilitates the integration of LoRA with PyTorch models and provide our implementations and model checkpoints for RoBERTa, DeBERTa, and GPT-2 at <https://github.com/microsoft/LoRA>.

Source: Edward J. Hu, Y. S., Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen. (2021). LoRA: Low-Rank Adaptation of Large Language Models.

We take inspiration from Li et al. (2018a); Aghajanyan et al. (2020) which show that the learned over-parametrized models in fact reside on a low intrinsic dimension. We hypothesize that the change in weights during model adaptation also has a low “intrinsic rank”, leading to our proposed **Low-Rank Adaptation (LoRA)** approach. LoRA allows us to train some dense layers in a neural

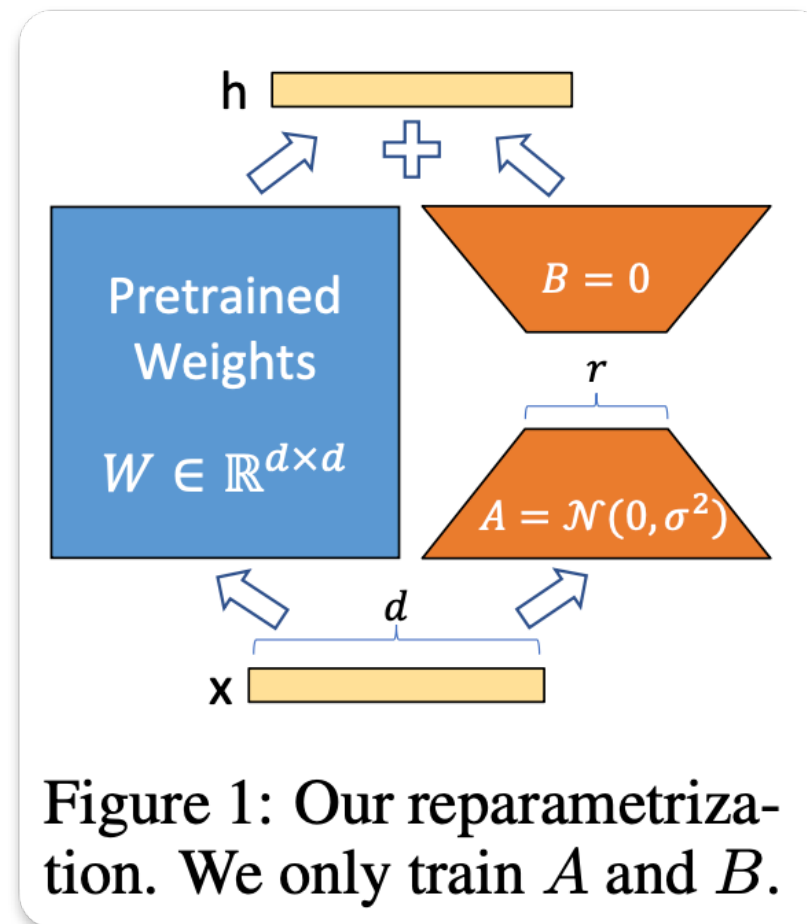
Source: Edward J. Hu, Y. S., Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen. (2021). LoRA: Low-Rank Adaptation of Large Language Models.

We take inspiration from Li et al. (2018a); Aghajanyan et al. (2020) which show that the learned over-parametrized models in fact reside on a low intrinsic dimension. We hypothesize that the change in weights during model adaptation also has a low “intrinsic rank”, leading to our proposed **Low-Rank Adaptation (LoRA)** approach. LoRA allows us to train some dense layers in a neural



Source: Edward J. Hu, Y. S., Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen. (2021). LoRA: Low-Rank Adaptation of Large Language Models.

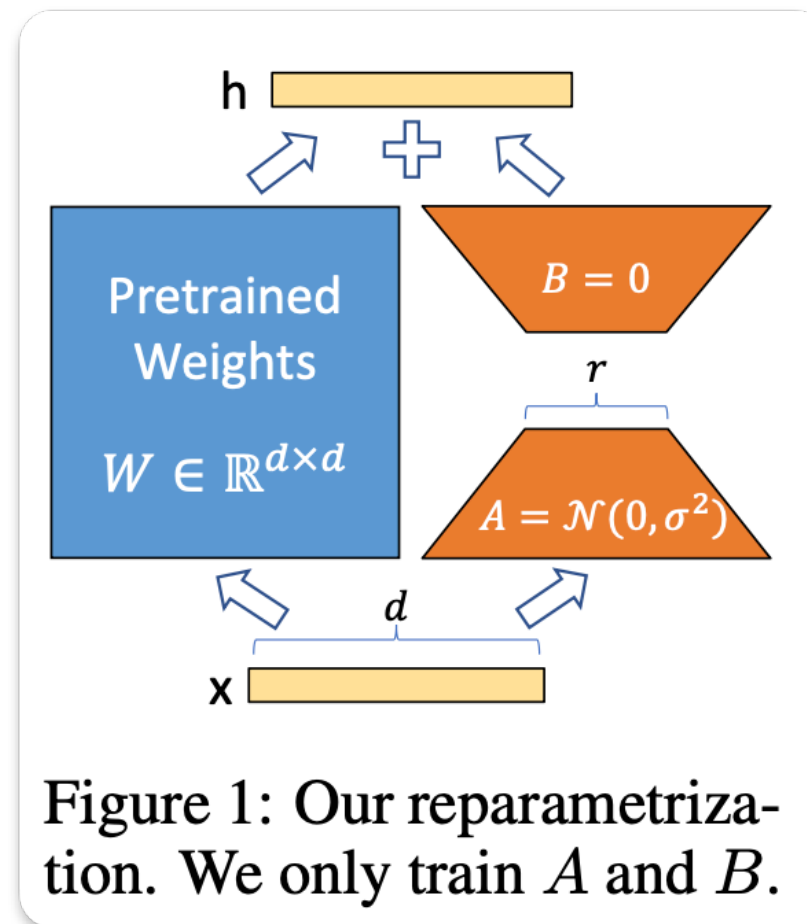
We take inspiration from Li et al. (2018a); Aghajanyan et al. (2020) which show that the learned over-parametrized models in fact reside on a low intrinsic dimension. We hypothesize that the change in weights during model adaptation also has a low “intrinsic rank”, leading to our proposed **Low-Rank Adaptation (LoRA)** approach. LoRA allows us to train some dense layers in a neural



What does this mean ?

Source: Edward J. Hu, Y. S., Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen. (2021). LoRA: Low-Rank Adaptation of Large Language Models.

We take inspiration from Li et al. (2018a); Aghajanyan et al. (2020) which show that the learned over-parametrized models in fact reside on a low intrinsic dimension. We hypothesize that the change in weights during model adaptation also has a low “intrinsic rank”, leading to our proposed **Low-Rank Adaptation (LoRA)** approach. LoRA allows us to train some dense layers in a neural

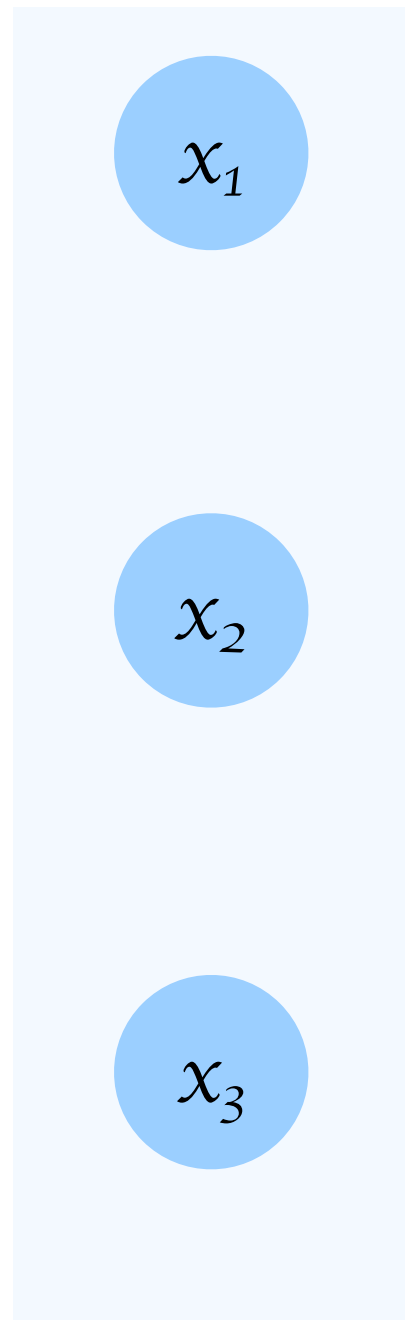


What does this mean ?

Source: Edward J. Hu, Y. S., Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen. (2021). LoRA: Low-Rank Adaptation of Large Language Models.

Refresher on weights in a neural net.

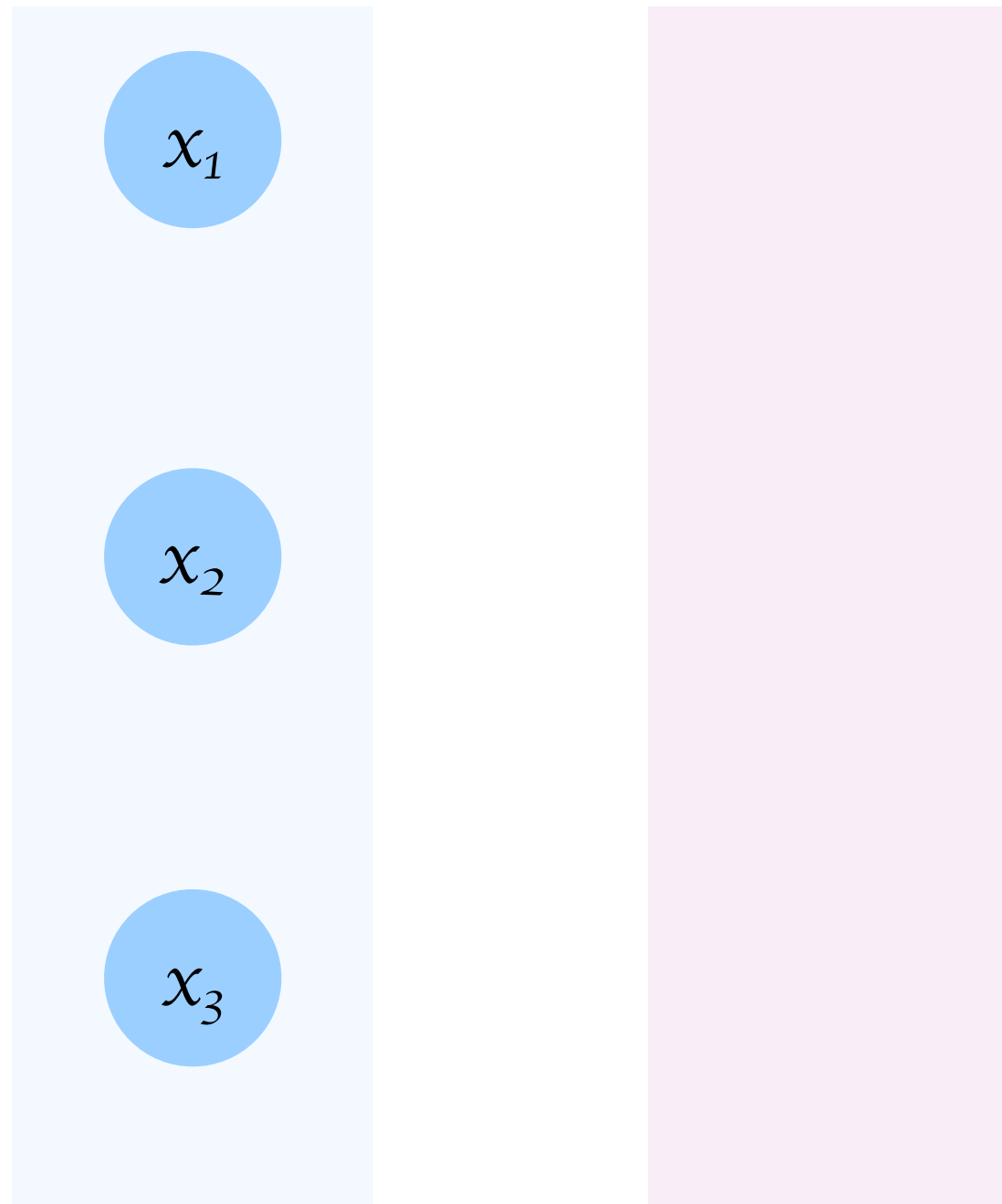
Refresher on weights in a neural net.



X

input

Refresher on weights in a neural net.



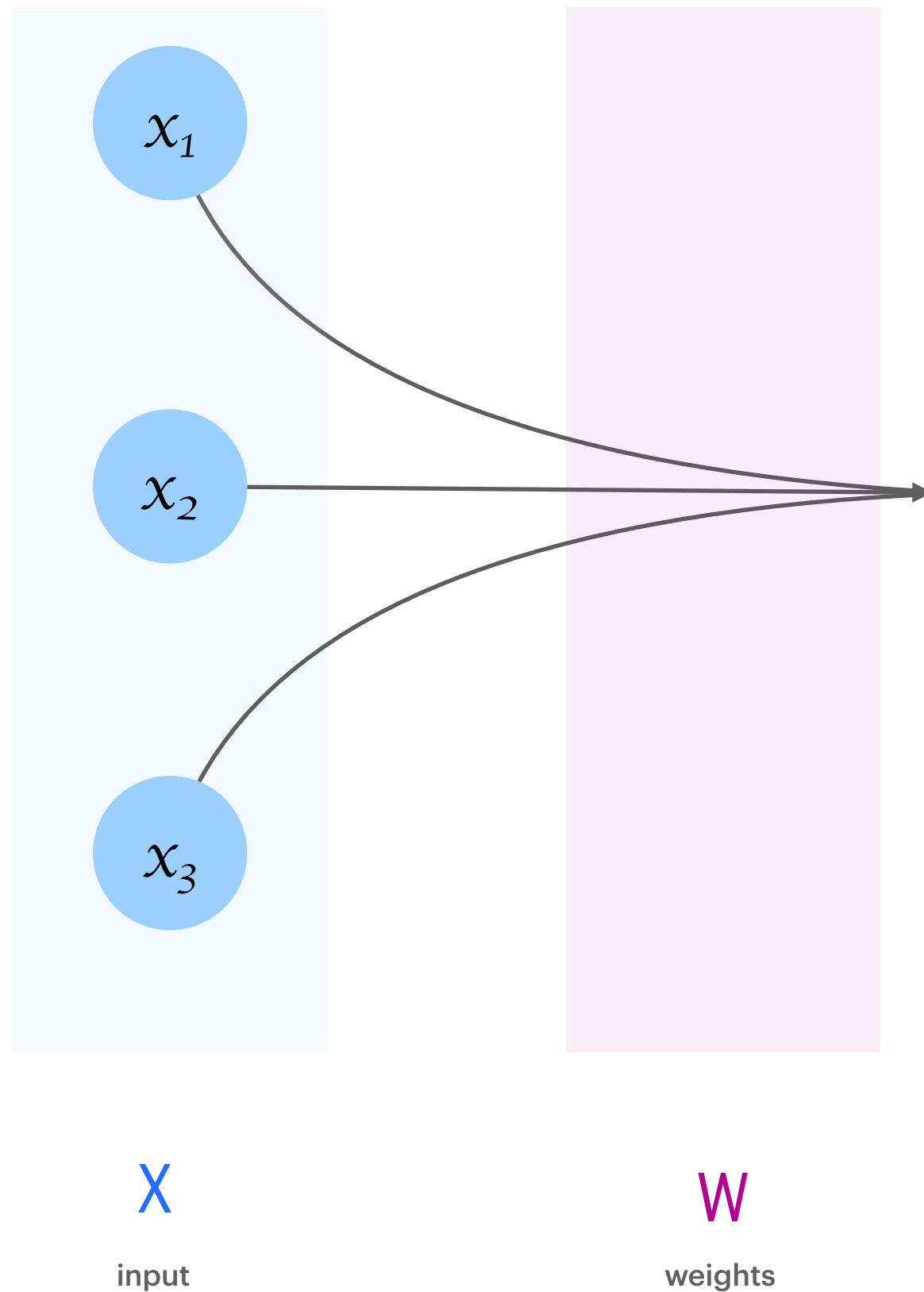
X

input

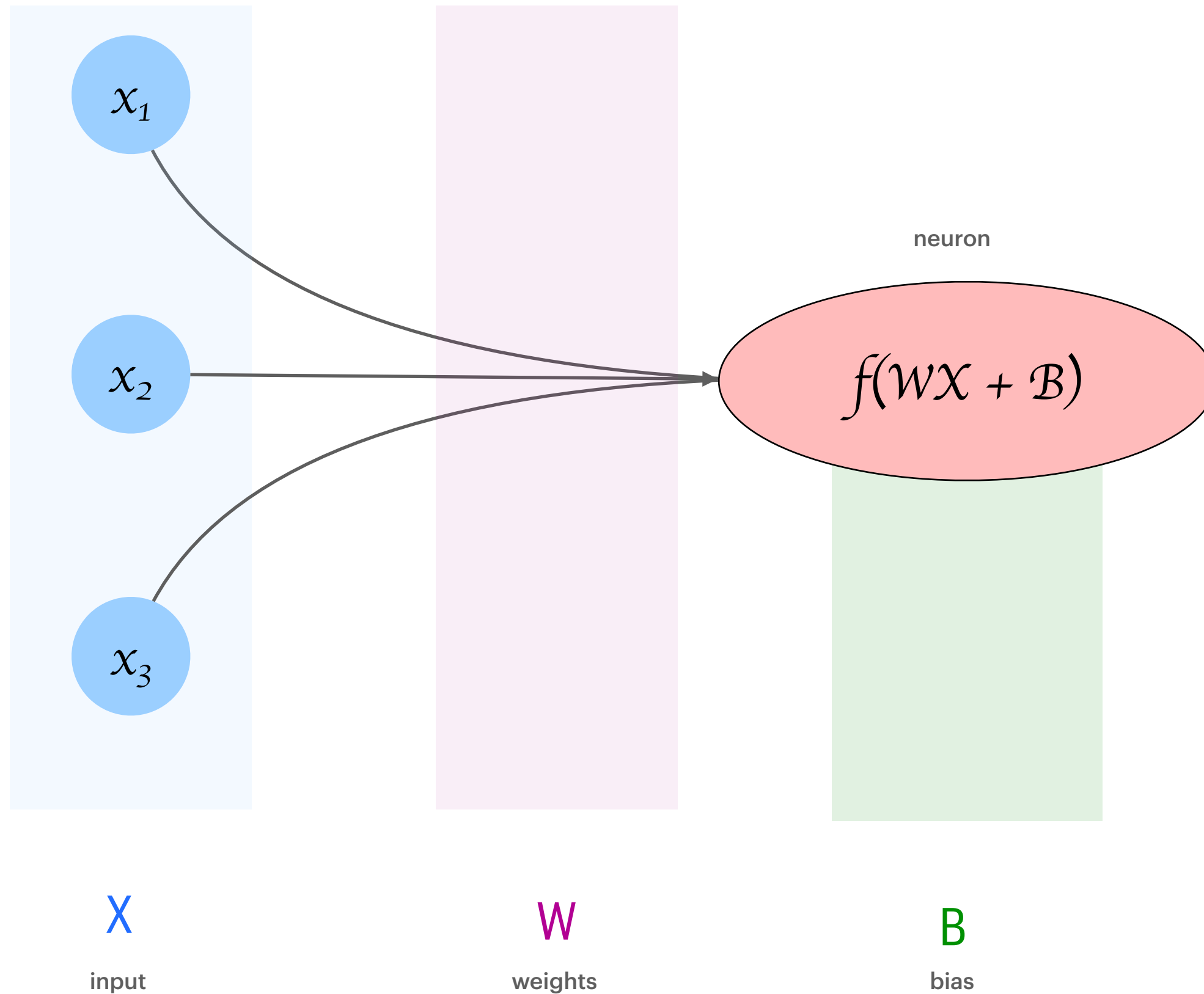
W

weights

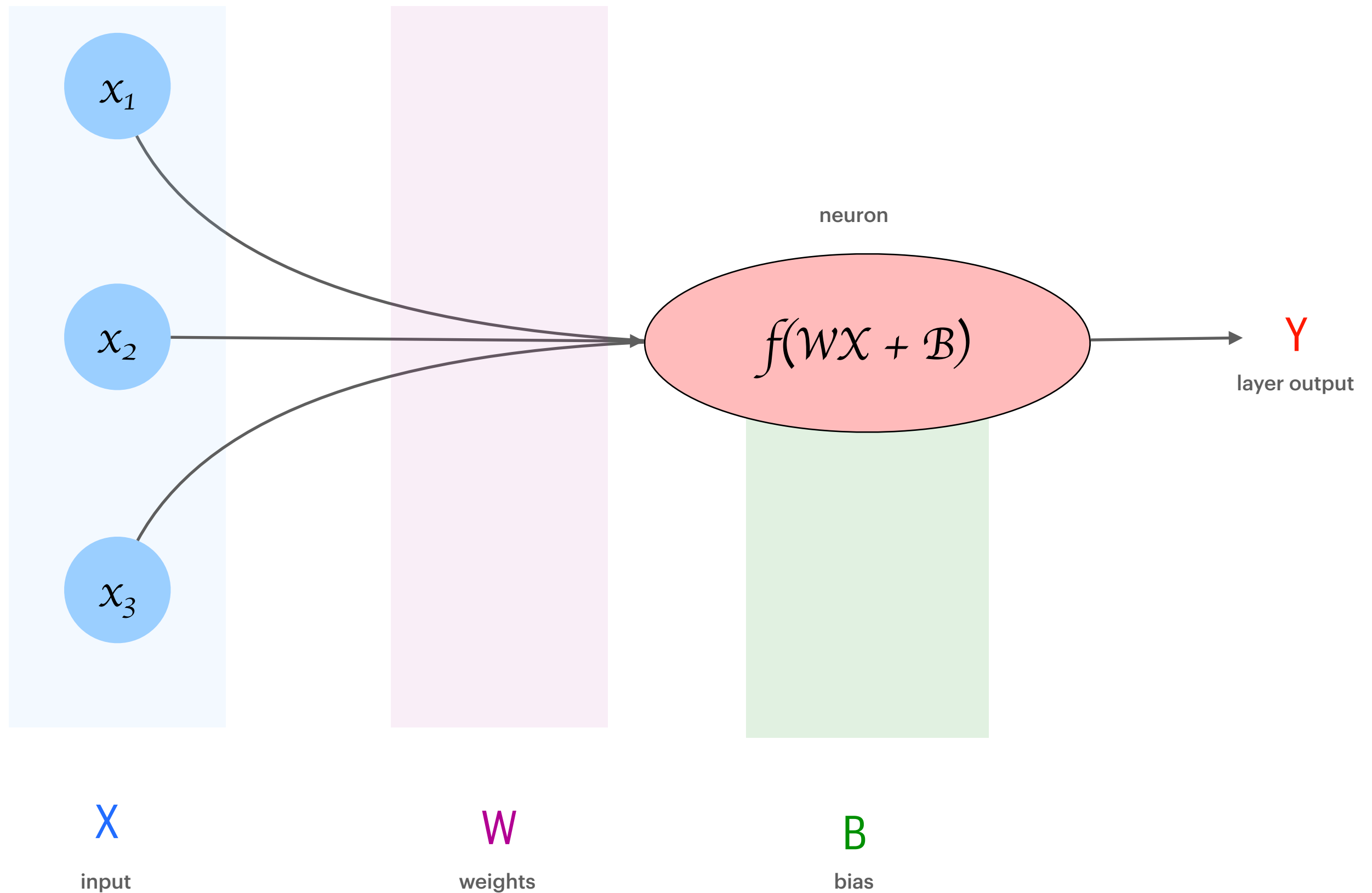
Refresher on weights in a neural net.



Refresher on weights in a neural net.



Refresher on weights in a neural net.



Refresher on weights in a neural net.

Linear combination of inputs with weights and biases

Example with $d = 3$

$$\begin{aligned}y_1 &= W_{11} * X_1 + W_{12} * X_2 + W_{13} * X_3 + b_1 \\y_2 &= W_{21} * X_1 + W_{22} * X_2 + W_{23} * X_3 + b_2 \\y_3 &= W_{31} * X_1 + W_{32} * X_2 + W_{33} * X_3 + b_3\end{aligned}$$

Refresher on weights in a neural net.

Linear combination of inputs with weights and biases
Example with $d = 3$

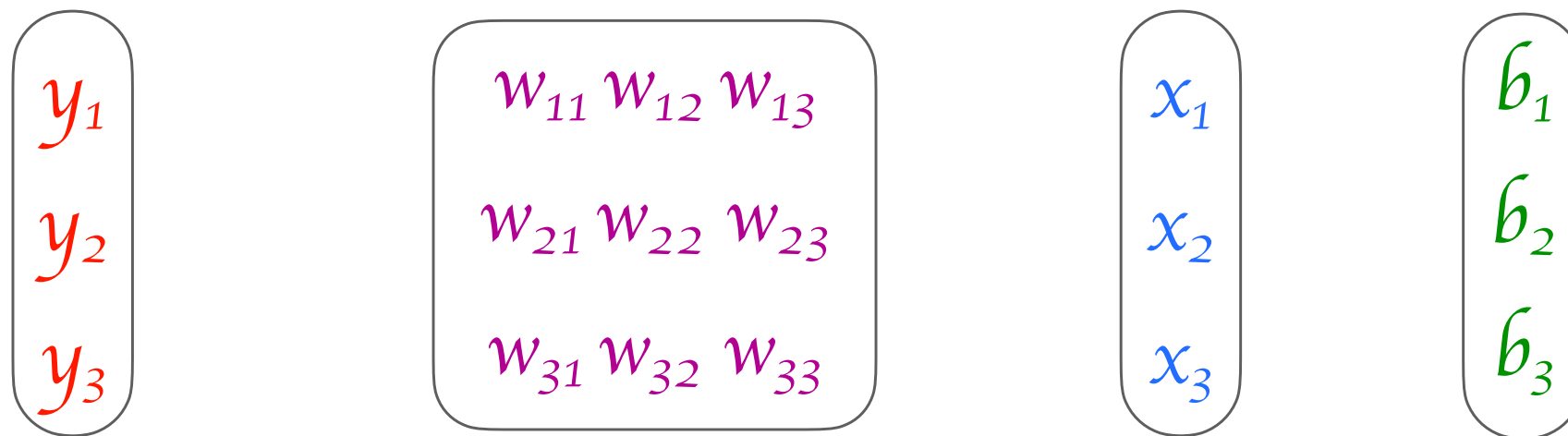
$$\begin{aligned}y_1 &= W_{11} * X_1 + W_{12} * X_2 + W_{13} * X_3 + b_1 \\y_2 &= W_{21} * X_1 + W_{22} * X_2 + W_{23} * X_3 + b_2 \\y_3 &= W_{31} * X_1 + W_{32} * X_2 + W_{33} * X_3 + b_3\end{aligned}$$

$$Y \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = W \begin{pmatrix} W_{11} & W_{12} & W_{13} \\ W_{21} & W_{22} & W_{23} \\ W_{31} & W_{32} & W_{33} \end{pmatrix} X \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + B \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

Refresher on weights in a neural net.

Linear combination of inputs with weights and biases
Example with $d = 3$

$$\begin{aligned}y_1 &= W_{11} * X_1 + W_{12} * X_2 + W_{13} * X_3 + b_1 \\y_2 &= W_{21} * X_1 + W_{22} * X_2 + W_{23} * X_3 + b_2 \\y_3 &= W_{31} * X_1 + W_{32} * X_2 + W_{33} * X_3 + b_3\end{aligned}$$



$$Y = W * X + B$$

Refresher on the rank of a matrix

In linear algebra, the rank of a matrix W is the dimension of the vector space generated by its columns (resp. rows). This corresponds to the maximal number of linearly independent columns (resp. rows) of W .

The rank is commonly denoted by $\text{rank}(W)$ or $\text{rk}(W)$; sometimes the parentheses are not written, as in $\text{rank } W$.

Source: Wikipedia, Rank of a matrix, [https://en.wikipedia.org/wiki/Rank_\(linear_algebra\)](https://en.wikipedia.org/wiki/Rank_(linear_algebra))

Refresher on the rank of a matrix

In linear algebra, the rank of a matrix W is the dimension of the vector space generated by its columns (resp. rows). This corresponds to the maximal number of linearly independent columns (resp. rows) of W .

The rank is commonly denoted by $\text{rank}(W)$ or $\text{rk}(W)$; sometimes the parentheses are not written, as in $\text{rank } W$.

Source: Wikipedia, Rank of a matrix, [https://en.wikipedia.org/wiki/Rank_\(linear_algebra\)](https://en.wikipedia.org/wiki/Rank_(linear_algebra))

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$W1$

$$\begin{pmatrix} 3 & -2 & 1 \\ 2 & -2 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$

$W2$

Refresher on the rank of a matrix

In linear algebra, the rank of a matrix W is the dimension of the vector space generated by its columns (resp. rows). This corresponds to the maximal number of linearly independent columns (resp. rows) of W .

The rank is commonly denoted by $\text{rank}(W)$ or $\text{rk}(W)$; sometimes the parentheses are not written, as in $\text{rank } W$.

Source: Wikipedia, Rank of a matrix, [https://en.wikipedia.org/wiki/Rank_\(linear_algebra\)](https://en.wikipedia.org/wiki/Rank_(linear_algebra))

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\text{rank } W1 = 3$$

$$\begin{pmatrix} 3 & -2 & 1 \\ 2 & -2 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$

$$\text{rank } W2 = 2$$

Refresher on the rank of a matrix

In linear algebra, the rank of a matrix W is the dimension of the vector space generated by its columns (resp. rows). This corresponds to the maximal number of linearly independent columns (resp. rows) of W .

The rank is commonly denoted by $\text{rank}(W)$ or $\text{rk}(W)$; sometimes the parentheses are not written, as in $\text{rank } W$.

Source: Wikipedia, Rank of a matrix, [https://en.wikipedia.org/wiki/Rank_\(linear_algebra\)](https://en.wikipedia.org/wiki/Rank_(linear_algebra))

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\text{rank } W1 = 3$$

$$\begin{matrix} C_1 & C_2 & C_3 \\ \begin{pmatrix} 3 & -2 & 1 \\ 2 & -2 & 0 \\ 0 & 1 & 1 \end{pmatrix} \end{matrix}$$

$$\text{rank } W2 = 2$$

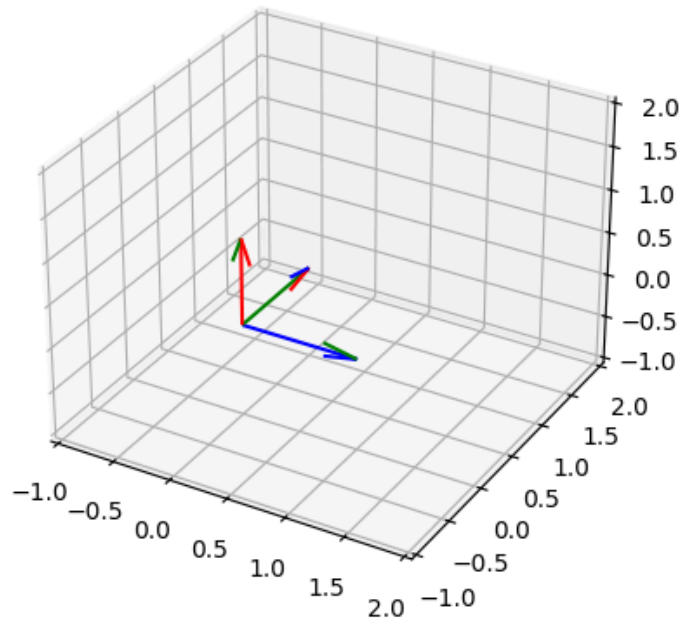
$$C_3 = C_1 + C_2$$

Refresher on the rank of a matrix

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

rank $W1 = 3$

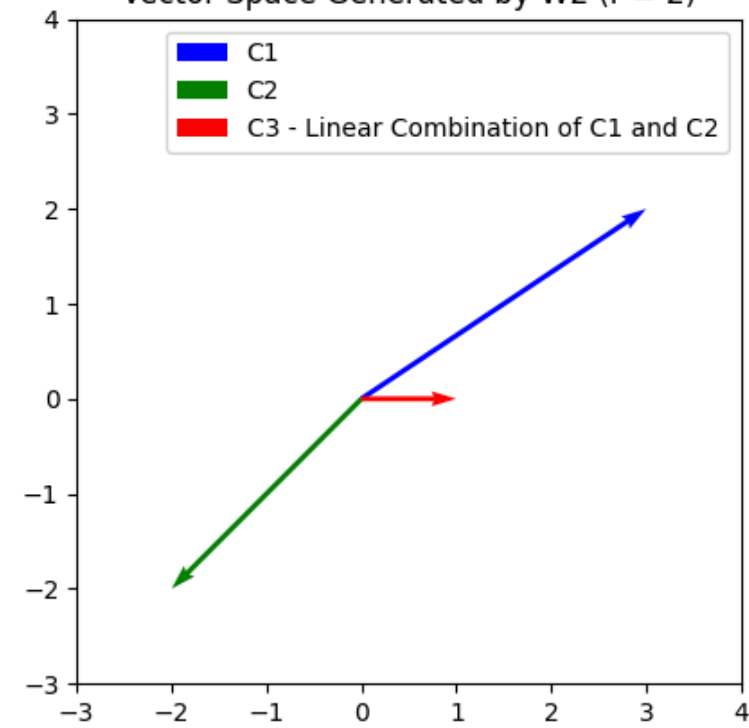
Vector Space Generated by $W1$ ($r = 3$)



$$\begin{pmatrix} 3 & -2 & 1 \\ 2 & -2 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$

rank $W2 = 2$ $C_3 = C_1 + C_2$

Vector Space Generated by $W2$ ($r = 2$)

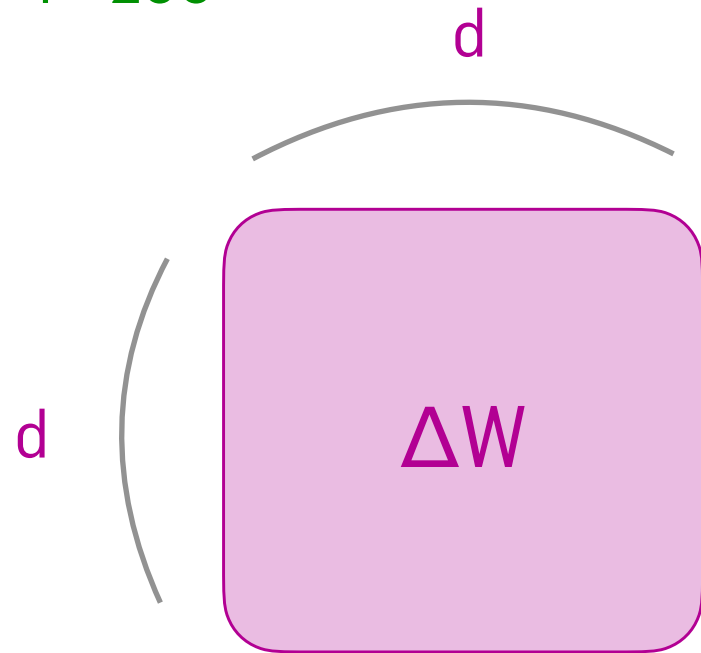


Source code available on GitHub Gist, matrix rank, <https://gist.github.com/fpaupier/1582e7e4f649feccd3aa6c18023cb2507>

You can represent low rank matrix with fewer parameters

$d = 1\,000$

$r = 200$

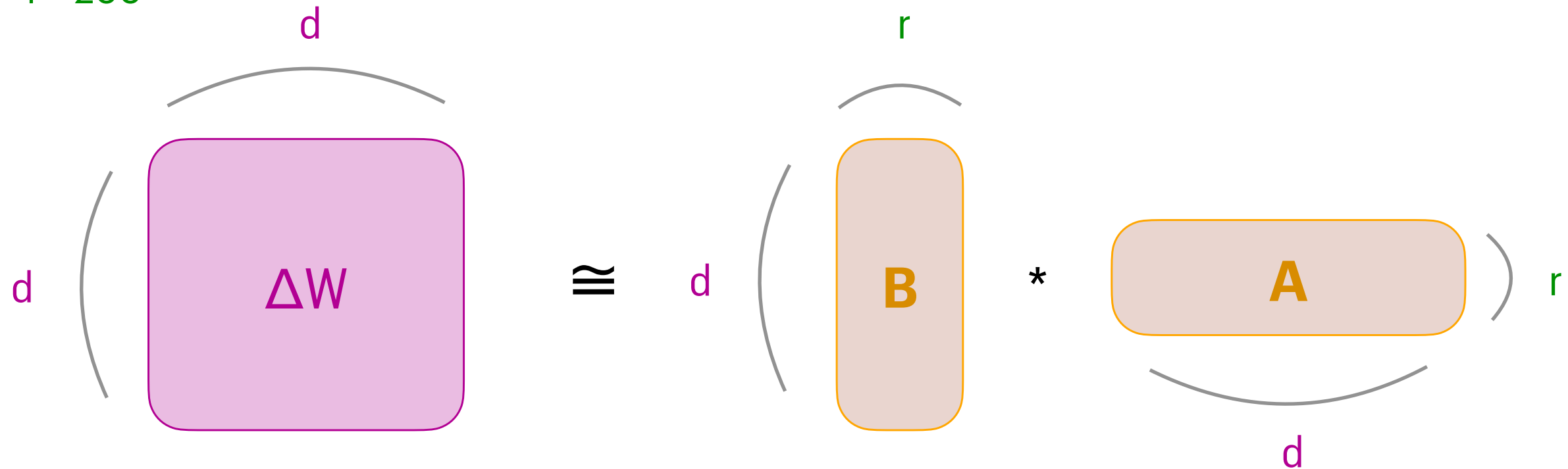


num params $\Delta W =$
 $d * d =$
1 000 000

You can represent low rank matrix with fewer parameters

$d = 1\,000$

$r = 200$



num params $\Delta W =$
 $d * d =$
1 000 000

num params matrix
decomposition =
 $d * r * 2 =$
400 000

The intuition for LoRA

Decompose the weight matrix in 2: 1/ a pretrained part, and 2/ the newly low rank matrix

$$Y = W * X \quad \longrightarrow \quad Y = (W_0 + \Delta W) * X$$

pretrained weights frozen trainable parameters

The intuition for LoRA

Decompose the weight matrix in 2: 1/ a pretrained part, and 2/ the newly low rank matrix

$$Y = W * X \quad \longrightarrow \quad Y = (W_0 + \Delta W) * X$$

pretrained weights frozen trainable parameters

Decompose low rank matrix into the product of two matrix $B \in \mathbb{R}^{d \times r}$ and $A \in \mathbb{R}^{r \times d}$

The intuition for LoRA

Decompose the weight matrix in 2: 1/ a pretrained part, and 2/ the newly low rank matrix

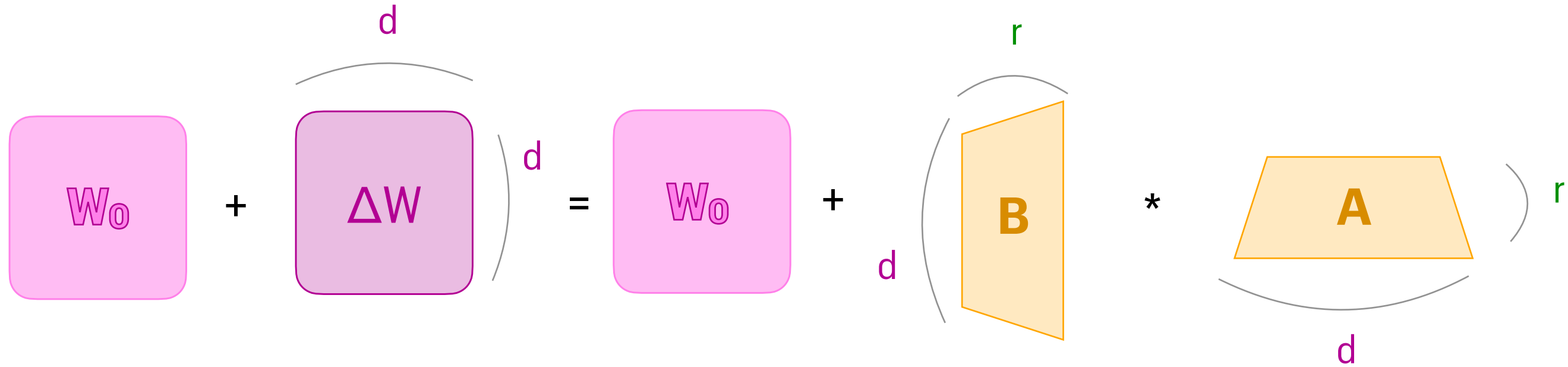
$$Y = W * X \quad \longrightarrow \quad Y = (W_0 + \Delta W) * X$$

pretrained weights frozen trainable parameters

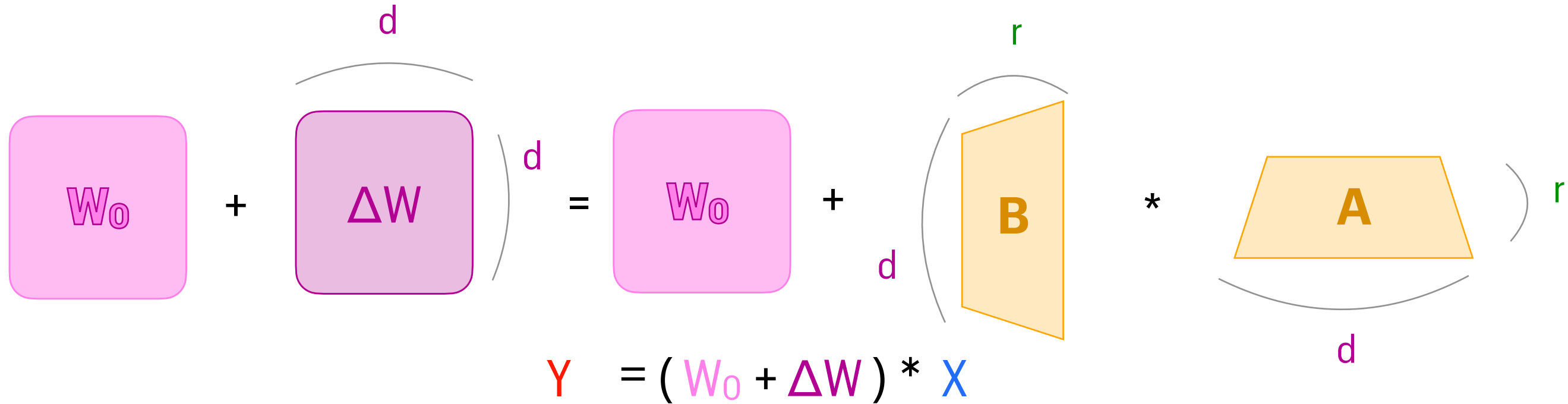
Decompose low rank matrix into the product of two matrix $B \in \mathbb{R}^{d \times r}$ and $A \in \mathbb{R}^{r \times d}$

$$\Delta W = B * A$$

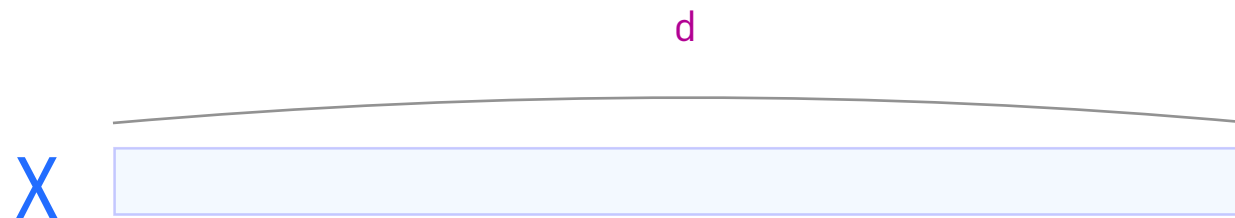
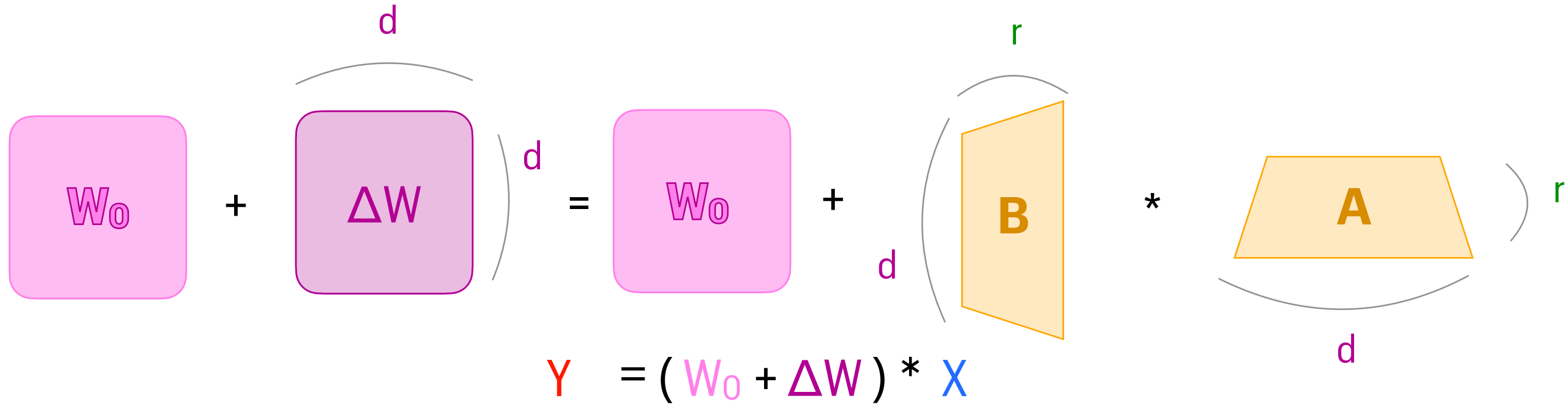
Only updating parameters for matrices A and B



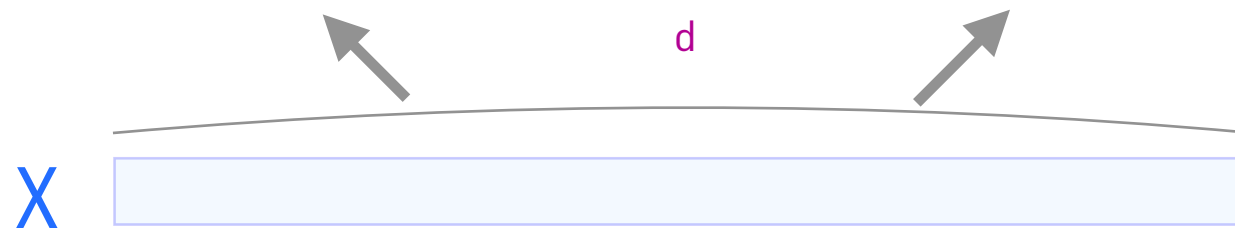
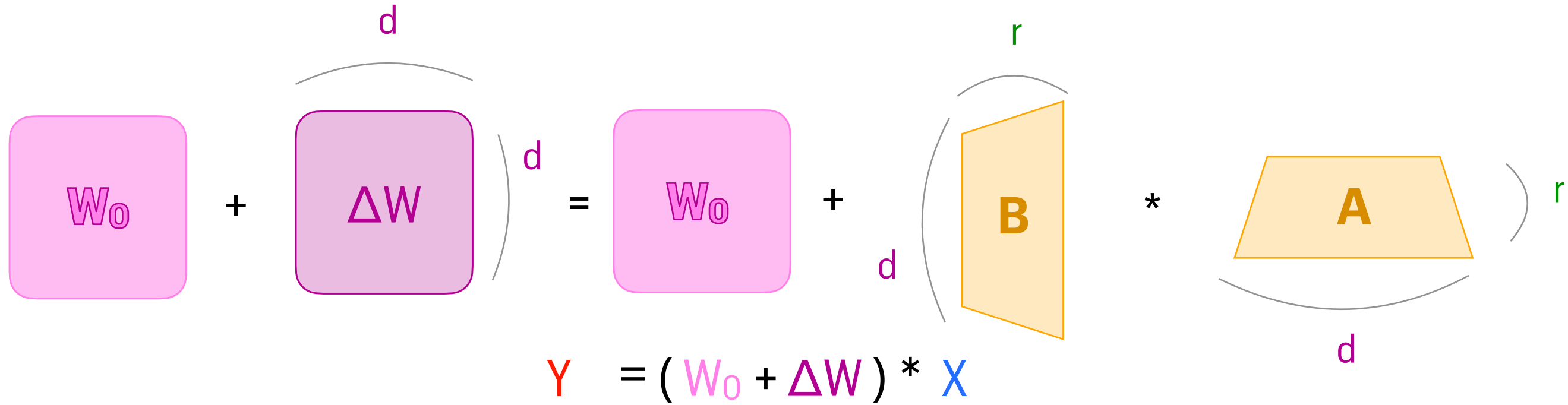
Only updating parameters for matrices A and B



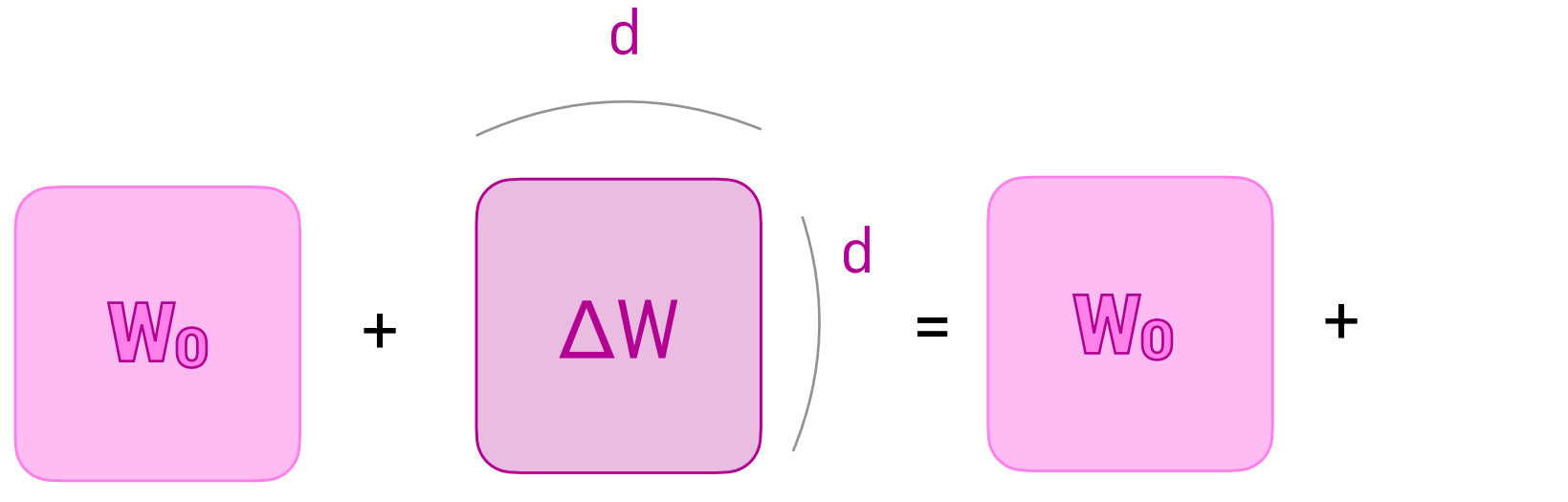
Only updating parameters for matrices A and B

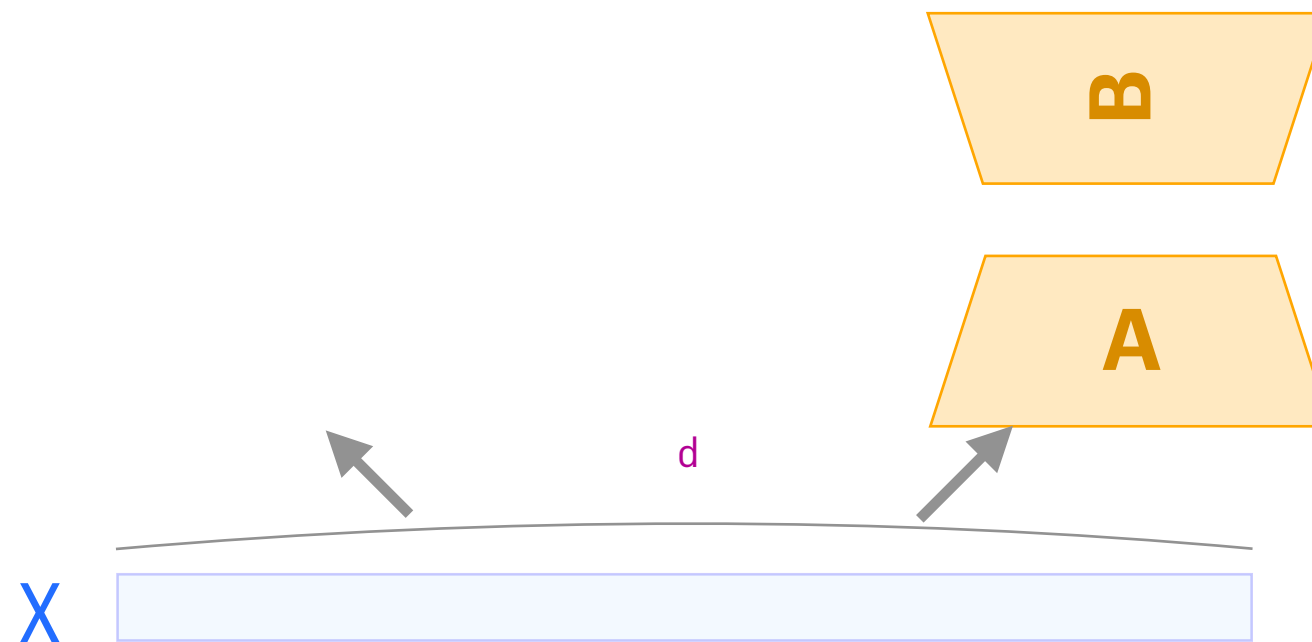


Only updating parameters for matrices A and B



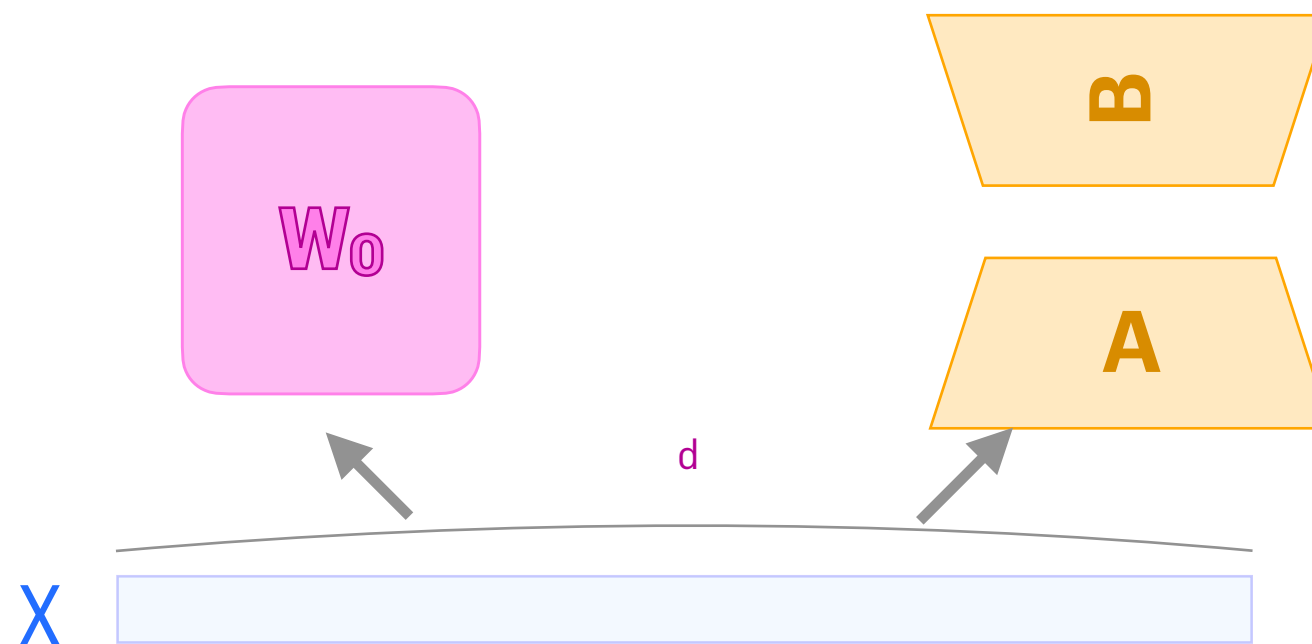
Only updating parameters for matrices A and B


$$W_0 + \overset{d}{\Delta W} = W_0 + \Delta W$$
$$Y = (W_0 + \Delta W) * X$$



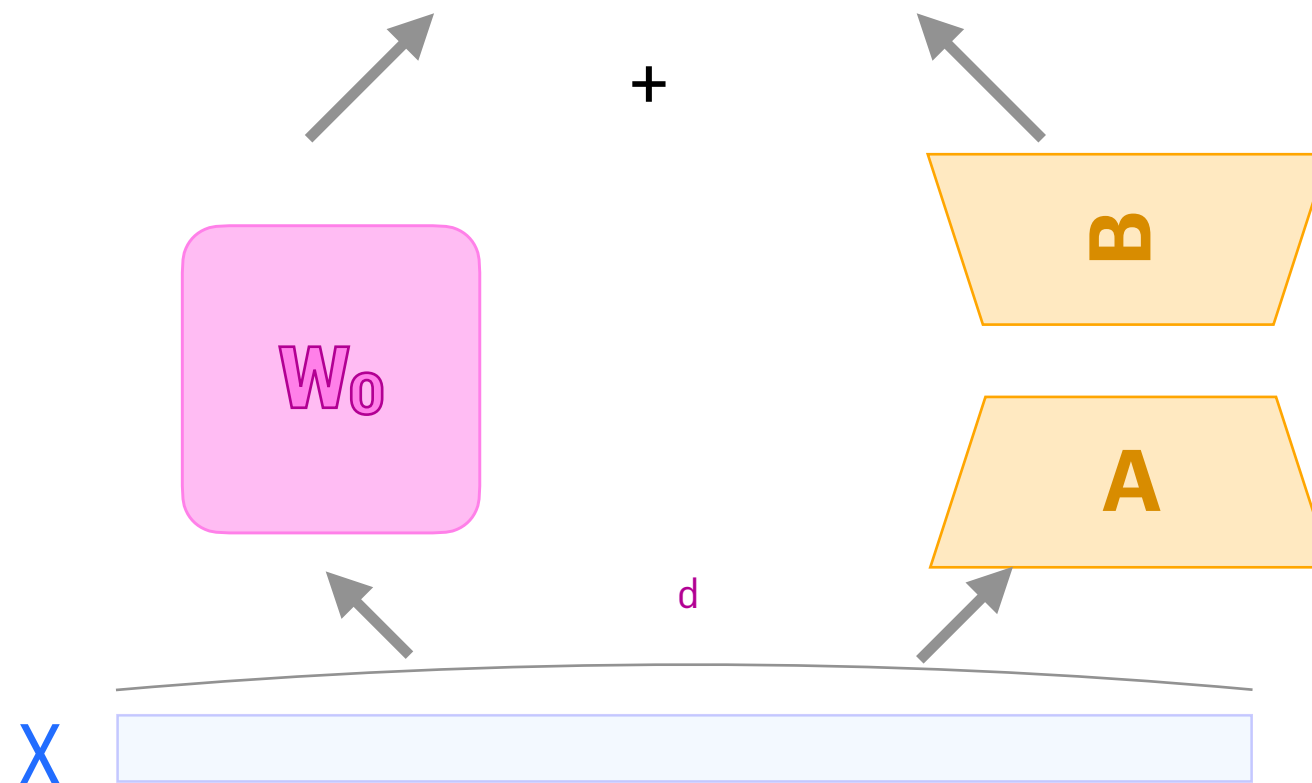
Only updating parameters for matrices A and B

$$+ \begin{matrix} \text{d} \\ \text{---} \\ \boxed{\Delta W} \\ \text{---} \\ \text{d} \end{matrix} = \boxed{W_0} +$$
$$Y = (W_0 + \Delta W) * X$$



Only updating parameters for matrices A and B

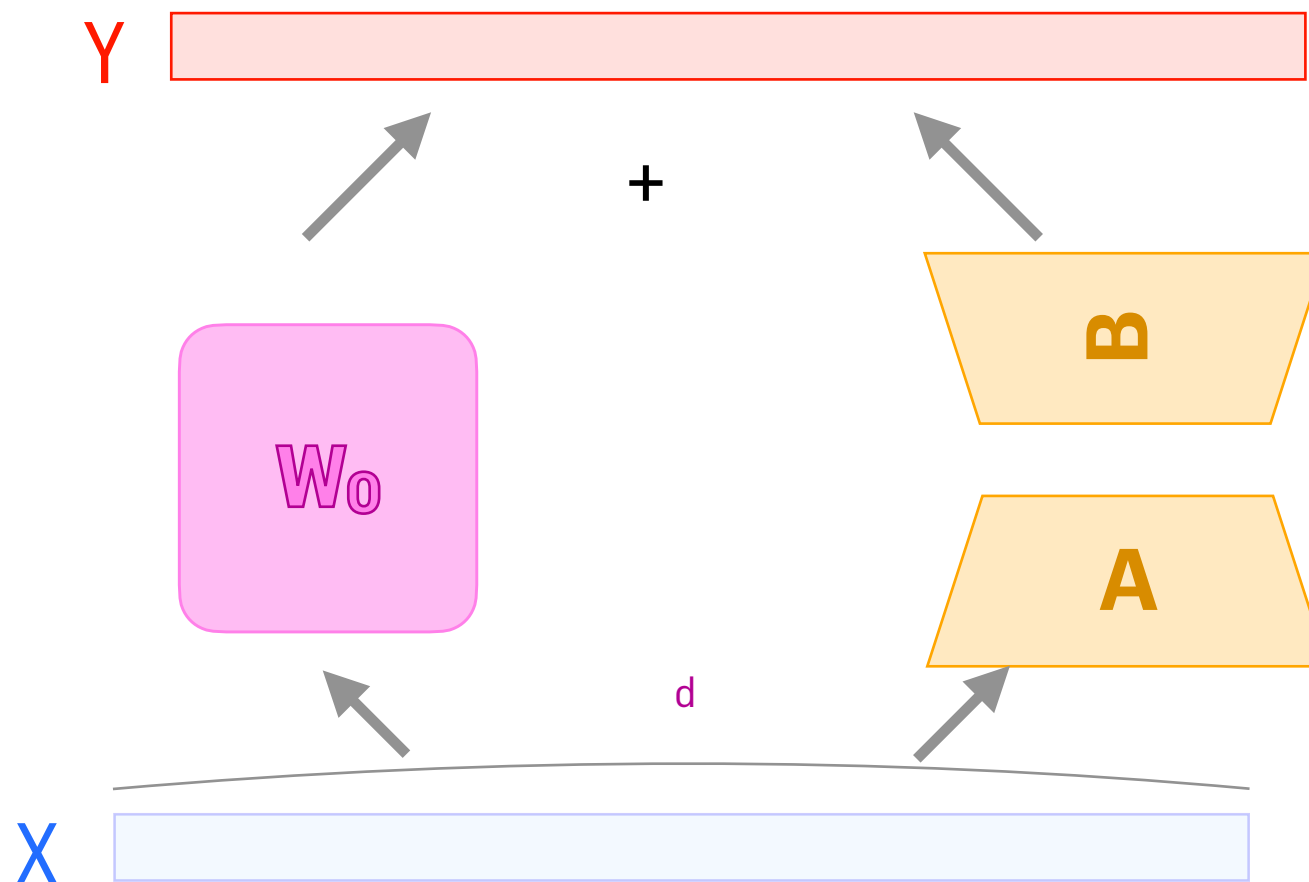
$$+ \begin{matrix} \text{d} \\ \text{---} \\ \boxed{\Delta W} \\ \text{---} \\ \text{d} \end{matrix} = \boxed{W_0} +$$
$$Y = (W_0 + \Delta W) * X$$

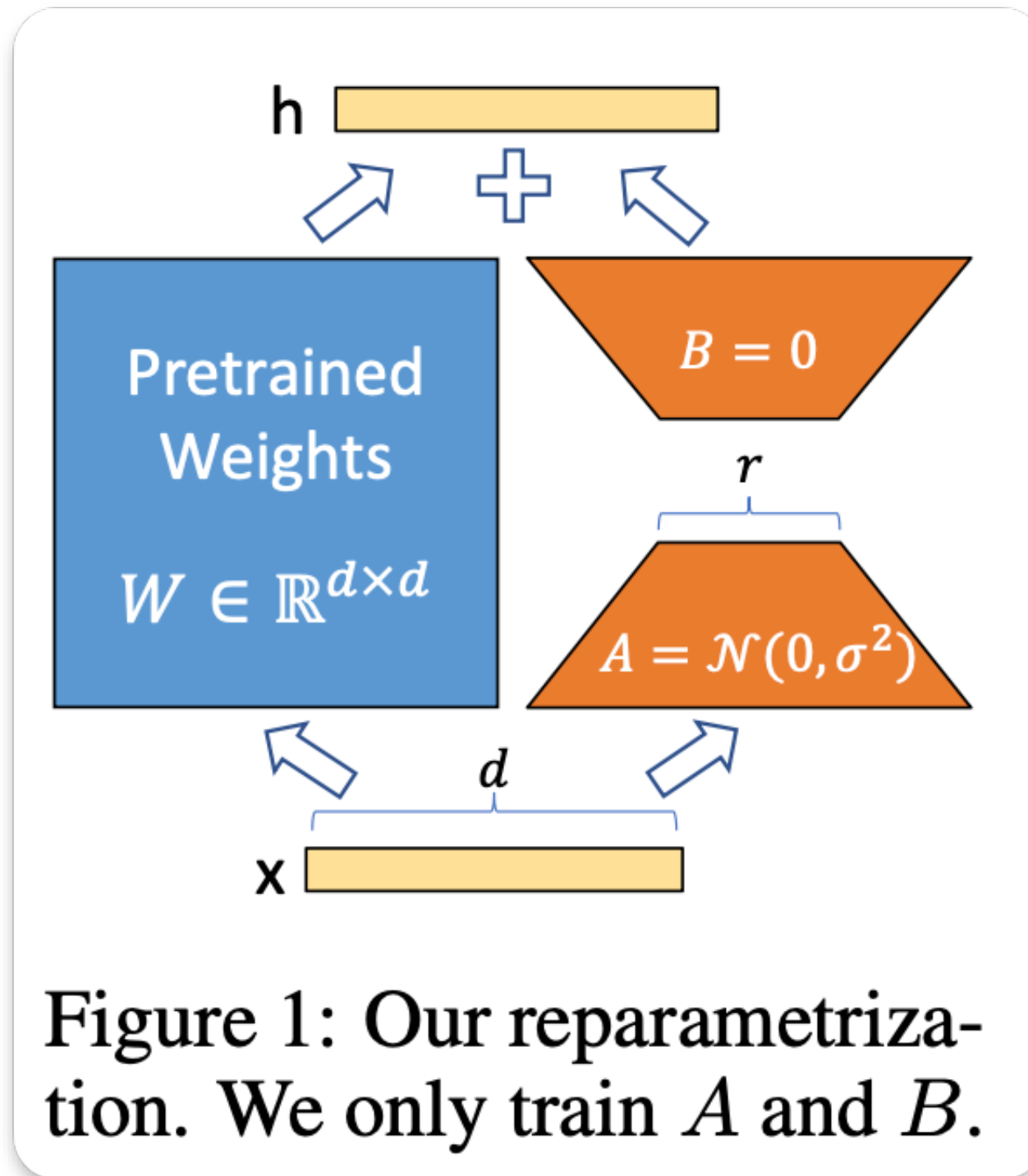


Only updating parameters for matrices A and B

$$+ \begin{matrix} \text{d} \\ \text{---} \\ \square \Delta W \end{matrix} = \begin{matrix} \square W_0 \end{matrix} +$$

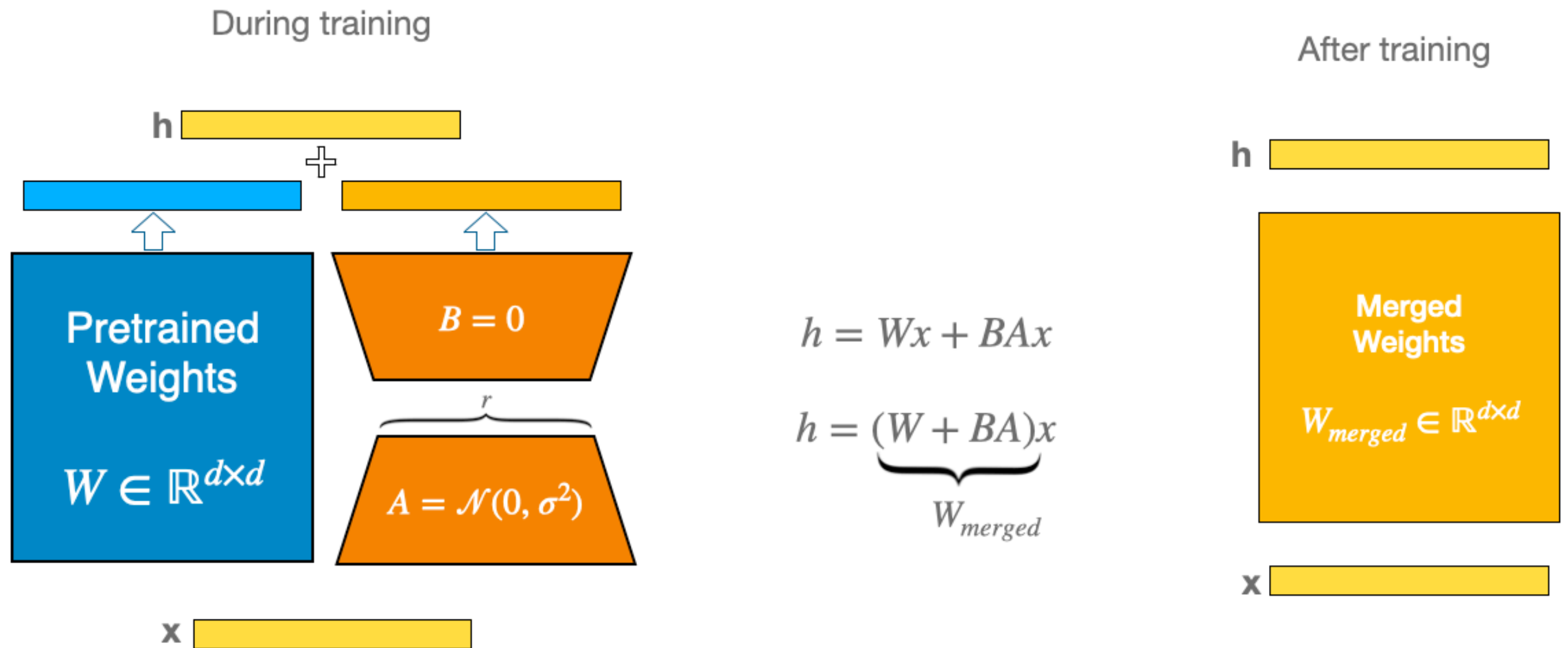
$$Y = (W_0 + \Delta W) * X$$





Source: Edward J. Hu, Y. S., Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen. (2021). LoRA: Low-Rank Adaptation of Large Language Models.

You can merge LoRA weights after training



Source: Edward J. Hu, Y. S., Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen. (2021). LoRA: Low-Rank Adaptation of Large Language Models.

Where LoRA can be applied in a transformers architecture

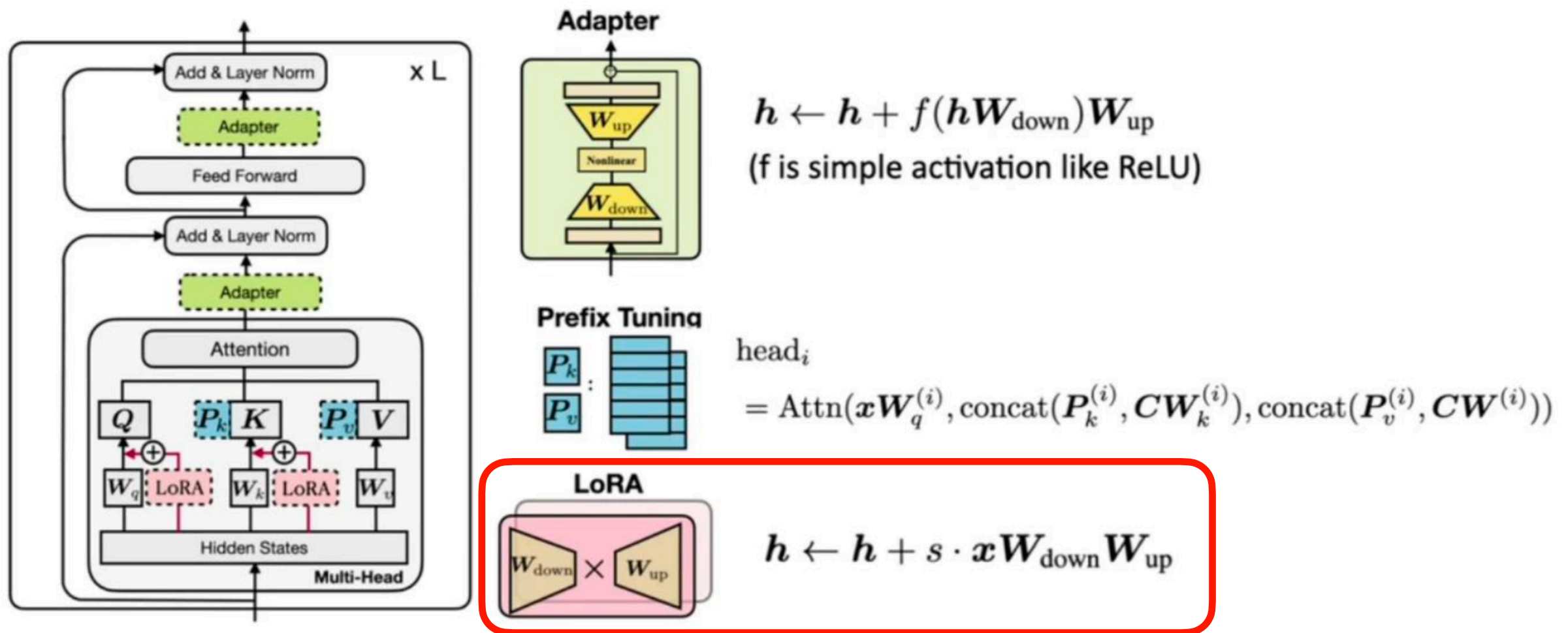


Figure 1 Three representative Parameter Efficient Tuning models: Adapter [11], Prefix Tuning [12], and LoRA [13].

Source: Kim, E., Hwang, M., & Lee, M. (2022). Recent Research Trend in Efficient NLP of Transformer.



LoRA - Lessons from hundreds of experiments

- **Multiple epochs over the finetuning dataset degrades performance**
- **Finetuned model has better performance when LoRA is applied across all weight matrices in the transformer (rather than just applying to the Q and V matrices)**
- **The rank r is a hyperparameter to finetune**
- **Using $\alpha = 2 * r$ rank is a good rule of thumb, but not infallible**
- **7B models can be finetuned within a few hours on a single GPU possessing 14 GB of VRAM.**

Sources:

- PhD, Sebastian R. (2023). Practical Tips for Finetuning LLMs Using LoRA (Low-Rank Adaptation). Retrieved Feb 2024, <https://magazine.sebastianraschka.com/p/practical-tips-for-finetuning-llms>.
- Wolfe, C. (2023). Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More. Retrieved Jan 2024, <https://cameronwolfe.substack.com/p/easily-train-a-specialized-llm-peft>.

PEFT - Useful libs

Name	Scope	Url
 Unloth	Finetuning with several approaches LoRa, QLoRA, RLHF (DPO)	https://github.com/unslothai/unsloth
 Hugging Face PEFT	Implementation of most PEFT approaches, regularly updated	https://huggingface.co/docs/peft
bitsandbytes	Python wrapper around CUDA with utils for quantization	https://github.com/TimDettmers/bitsandbytes

Dataset format for applying LoRA in supervised finetuning

Instruction dataset

Datasets: HuggingFaceH4 / **instruction-dataset** like 28 Dataset

Split (1)
test · 327 rows ▼

🔍 Search this dataset

prompt string · lengths	completion string · lengths
 35→276 76.1%	 312→623 17.4%
Arianna has 12 chocolates more than Danny. Danny has 6 chocolates more than Robbie. Arianna has twice as many...	Denote the number of chocolates each person has by the letter of their first name. We know that $A = D + 12$ $D = R + 6$ $A = 2 * ...$
Write a plot summary for a comedic novel involving Elon Musk and sea travel.	Elon Musk hires a team of experts to build the ultimate yacht, but when the yacht is completed, he realizes that he has no idea how to sail it. With the help of a quirky crew and a fearless captain, the playboy embarks on a wild and hilarious adventure across the open seas, where the crew have to keep Elon alive despite his inability to do anything himself. All the while, Elon takes credit for their hard work.
Create a 3 turn conversation between a customer and a grocery store clerk - that is, 3 per person. Then tell me what they...	Clerk: How are you doing today? Customer: Great! I'm buying groceries to cook my wife a homemade steak and lobster dinner...
Write a poem about the sun and moon.	The sun and the moon, the guards from the sky One works the day, other watches the night As the rays of sunshine hide...
Does Searle believe that AI can think? Explain your using reasons based off this text. TEXT Searle's thought experimen...	No, Searle does not believe that AI can think. Here is a step by step explanation of his reasoning: First, think of an AI...

Source: HuggingFace H4 dataset, <https://huggingface.co/datasets/HuggingFaceH4/instruction-dataset>

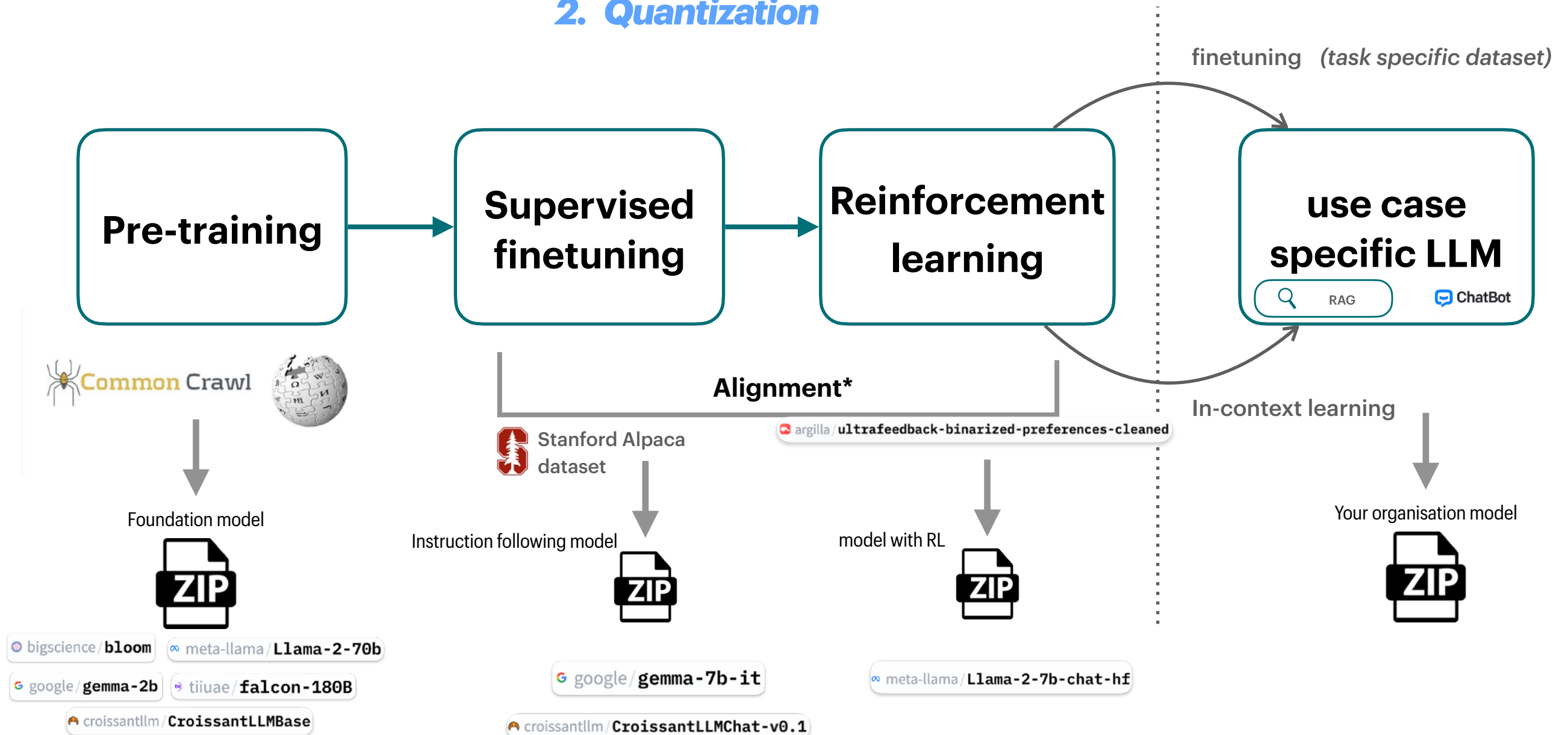
Pt 2.

Quantization

Compressing information

LLM training pipeline

2. Quantization



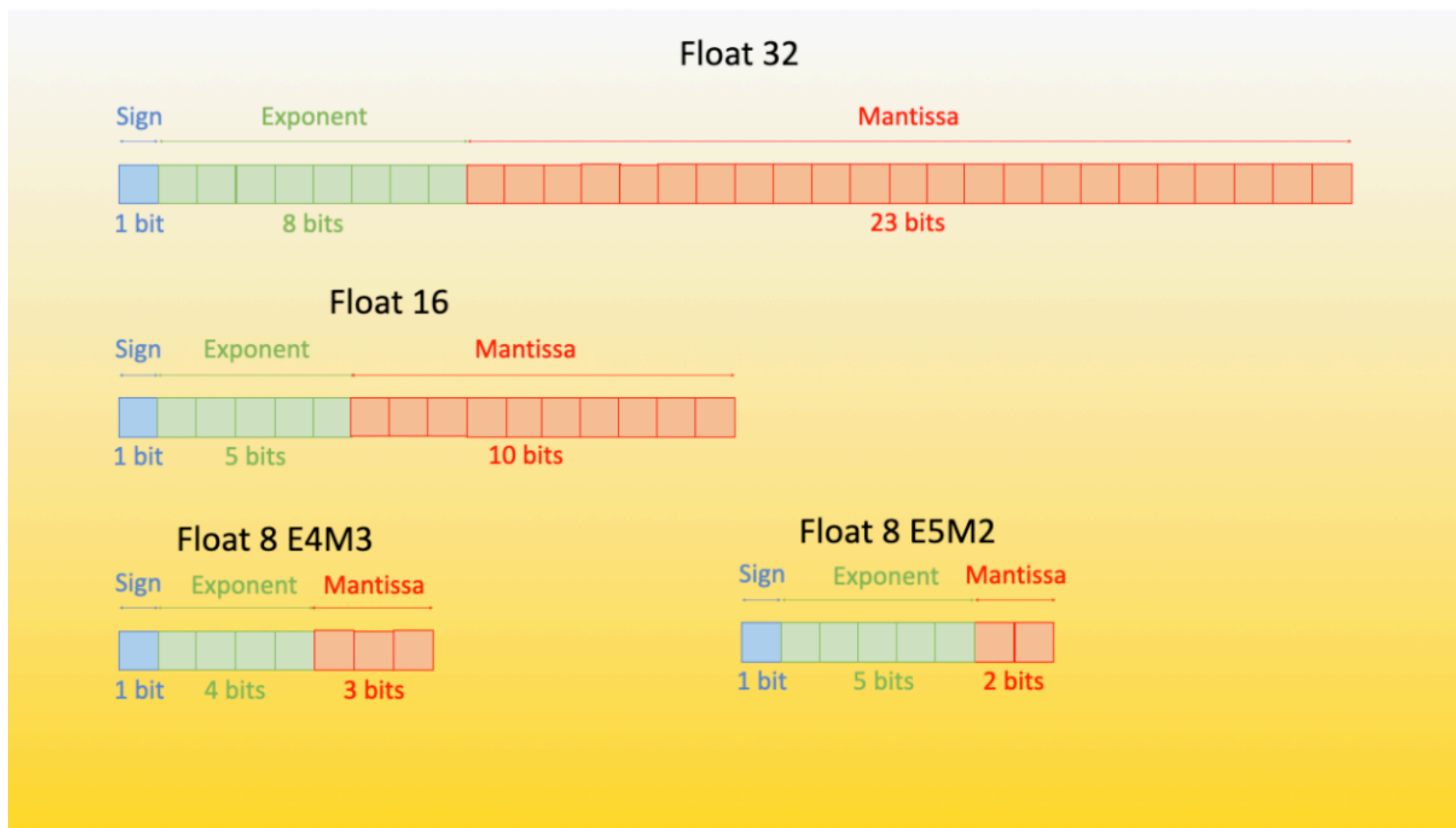
Adapted from Wolfe, C. (2023). Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More. Retrieved Feb 2024, <https://cameronwolfe.substack.com/p/easily-train-a-specialized-llm-peft>.

Quantization - motivations

- **Run training / inference on consumer hardware**
- **Achieve the best performance / memory consumption tradeoff**

NB: Not limited to NLP also plenty of use cases in CV

Quantization is about how you represent parameters in your model






Source: Overview of Floating Point 8 (FP8) format. Source: Original content from [sgugger](https://huggingface.co/blog/4bit-transformers-bitsandbytes), <https://huggingface.co/blog/4bit-transformers-bitsandbytes>

Quantization - two main approaches

POST-TRAINING QUANTIZATION

PTQ





-  Takes a pre-trained network and converts it to a fixed-point network without access to the training pipeline
-  Data-free or small calibration set needed
-  Lower accuracy at lower bit-widths

Example implementation

QLoRA

QUANTIZATION AWARE TRAINING

QAT

-  Requires access to training pipeline and labelled data
-  Longer training times
-  Hyper-parameter tuning
-  Achieves higher accuracy

Example implementation

QALoRA

Source: tinyML Talks: A Practical Guide to Neural Network Quantization, Qualcomm AI Research, Sep. 28 2021, <https://youtu.be/KASuxB3XoYQ>

Quantization - two main approaches

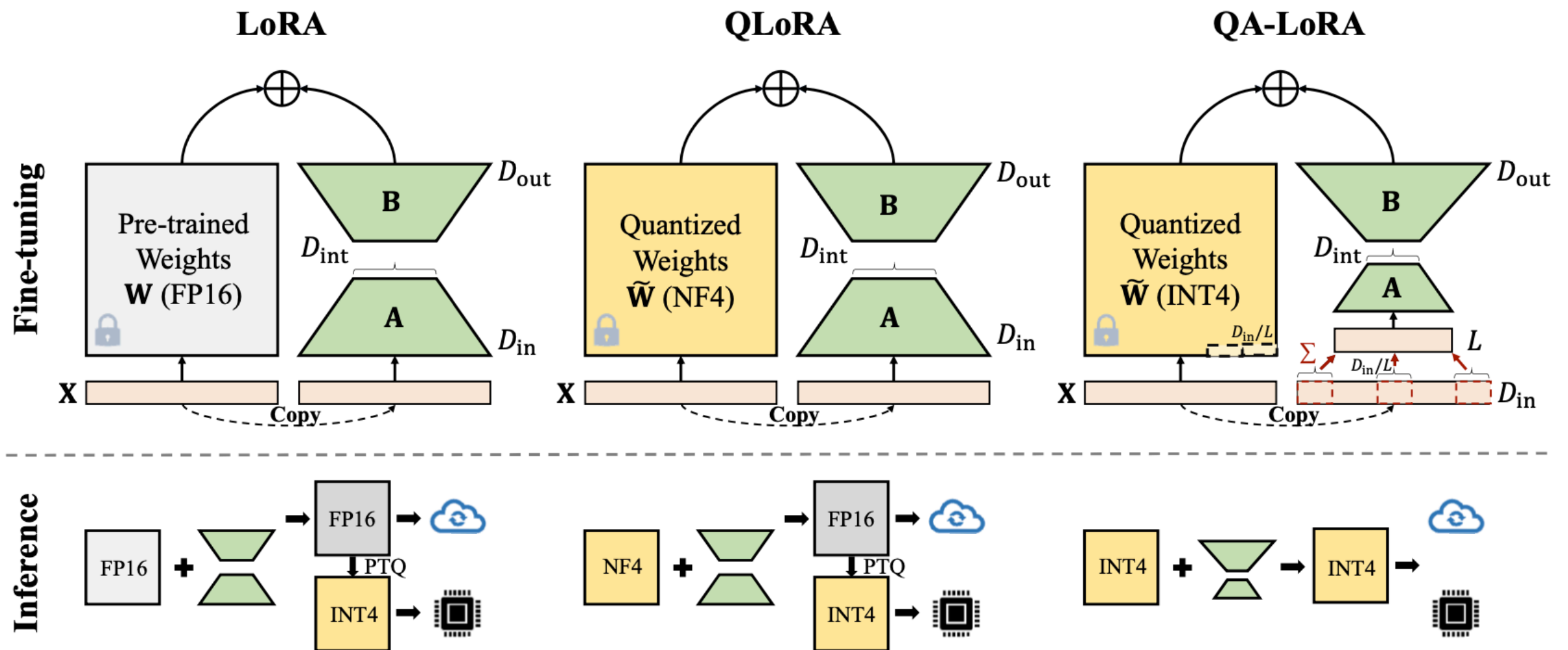
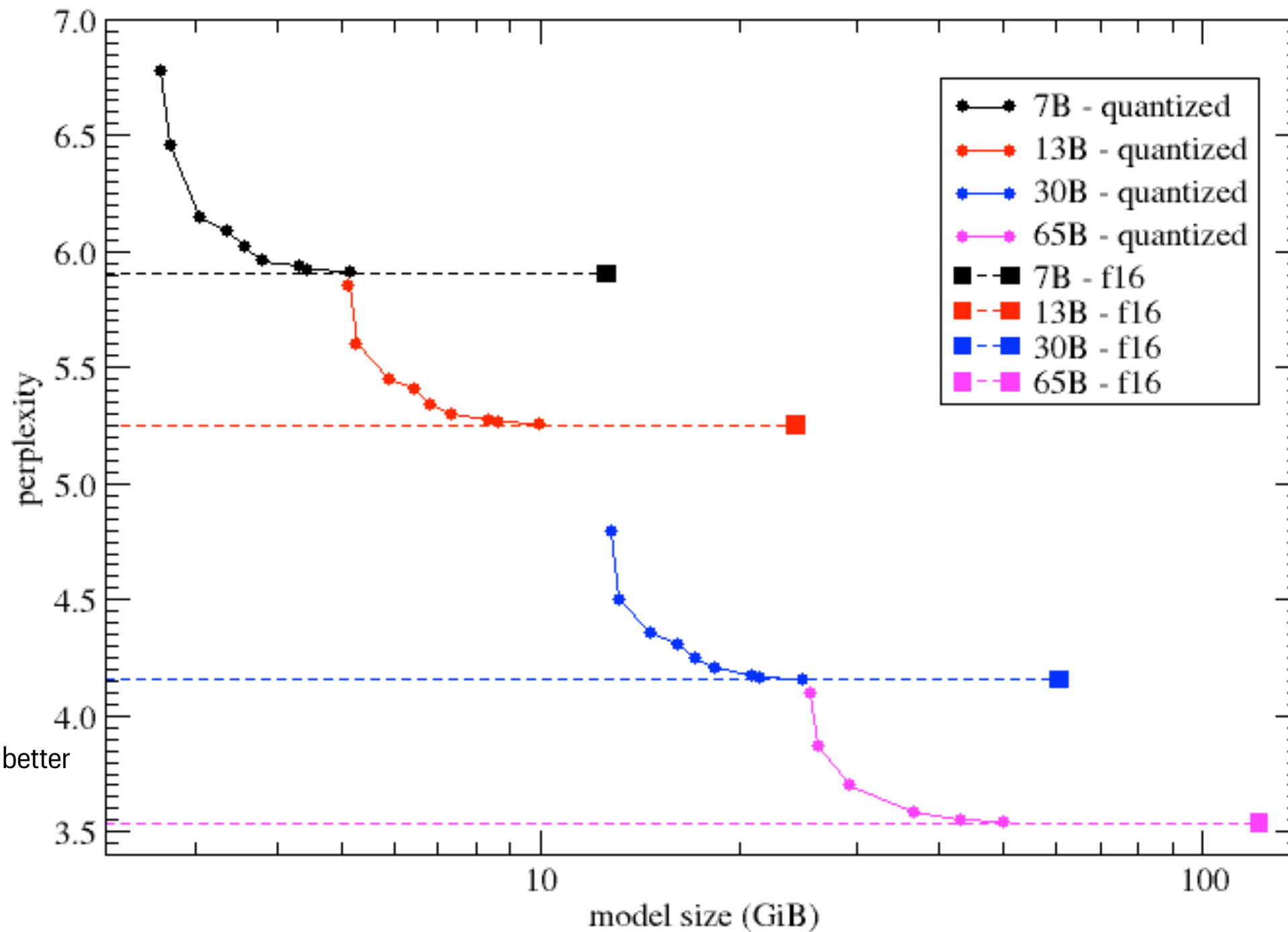


Figure 2: An illustration of the goal of QA-LoRA. Compared to prior adaptation methods, LoRA and QLoRA, our approach is computationally efficient in both the fine-tuning and inference stages. More importantly, it does not suffer an accuracy loss because post-training quantization is not required. We display INT4 quantization in the figure, but QA-LoRA is generalized to INT3 and INT2.

Source: Yuhui, X., Lingxi, X., Xiaotao, G., Xin, C., Heng, C., Hengheng, Z. et al. (2023). QA-LoRA: Quantization-Aware Low-Rank Adaptation of Large Language Models.

Quantization impact on performance



lower perplexity is better



Source: Source: ikawrakow, k-quants, <https://github.com/ggerganov/llama.cpp/pull/1684>

Understanding GGUF quantization naming conventions

Naming convention of “q” + the number of bits used to store the weights (precision) + a particular variant.

Explanation of quantisation methods

▼ Click to see details

The new methods available are:

- GGML_TYPE_Q2_K - "type-1" 2-bit quantization in super-blocks containing blocks, each block having 16 weight. Block scales and mins are quantized with 6 bits. This ends up effectively using 2.5625 bits per weight (bpw)
- GGML_TYPE_Q3_K - "type-0" 3-bit quantization in super-blocks containing blocks, each block having 16 weights. Scales are quantized with 6 bits. This ends up using 3.4375 bpw.
- GGML_TYPE_Q4_K - "type-1" 4-bit quantization in super-blocks containing blocks, each block having 32 weights. Scales and mins are quantized with 6 bits. This ends up using 4.5 bpw.
- GGML_TYPE_Q5_K - "type-1" 5-bit quantization. Same super-block structure as GGML_TYPE_Q4_K resulting in 5.5 bpw
- GGML_TYPE_Q6_K - "type-0" 6-bit quantization. Super-blocks with 16 blocks, each block having 16 weights. Scales are quantized with 8 bits. This ends up using 6.5625 bpw

Name	Quant method	Bits	Size	Max RAM required	Use case
codellama-7b.Q2_K.gguf	Q2_K	2	2.83 GB	5.33 GB	smallest, significant quality loss - not recommended for most purposes
codellama-7b.Q3_K_S.gguf	Q3_K_S	3	2.95 GB	5.45 GB	very small, high quality loss
codellama-7b.Q3_K_M.gguf	Q3_K_M	3	3.30 GB	5.80 GB	very small, high quality loss
codellama-7b.Q3_K_L.gguf	Q3_K_L	3	3.60 GB	6.10 GB	small, substantial quality loss
codellama-7b.Q4_0.gguf	Q4_0	4	3.83 GB	6.33 GB	legacy; small, very high quality loss - prefer using Q3_K_M

Sources:

- The Bloke, CodeLlama 7B - GGUF, <https://huggingface.co/TheBloke/CodeLlama-7B-GGUF>
- Labonne, M. (2023). Quantize Llama models with GGUF and llama.cpp. Retrieved Jan 22 2024, https://mlabonne.github.io/blog/posts/Quantize_Llama_2_models_using_ggml.html

Lessons from trying several quantization config

- **K_M models are generally better than the K_S**
- **Q2_K or Q3_* have bad performances and should only be used for test/dev**
- **As a rule of thumb, Q5_K_M model offer the best performance/memory trade-off**

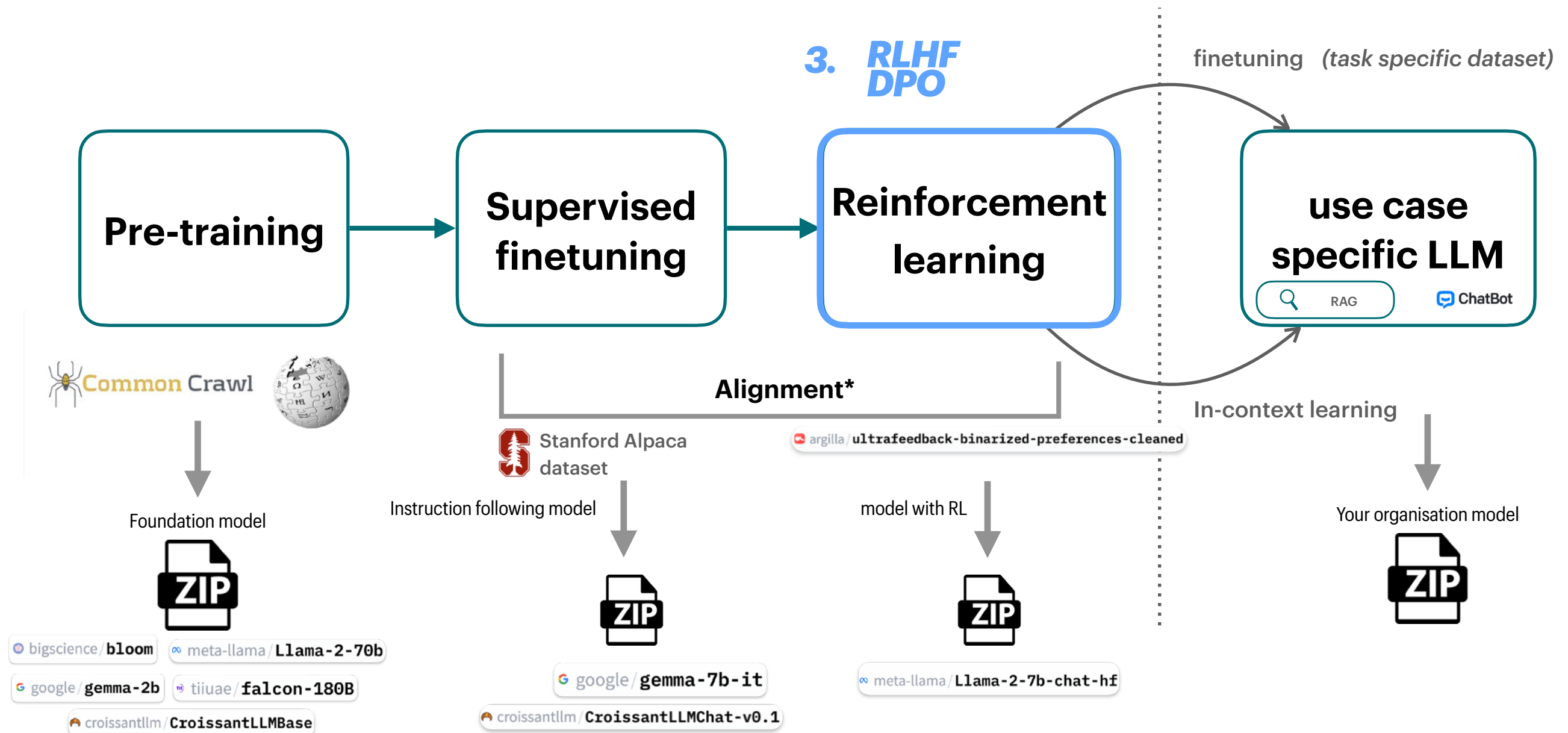
Source: Labonne, M. (2023). Quantize Llama models with GGUF and llama.cpp. Retrieved Jan 22 2024, https://mlabonne.github.io/blog/posts/Quantize_Llama_2_models_using_ggml.html

Pt 3.

RLHF

Reinforcement learning, Direct Preference Optimisation


LLM training pipeline




Adapted from Wolfe, C. (2023). Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More. Retrieved Feb 2024, <https://cameronwolfe.substack.com/p/easily-train-a-specialized-llm-peft>.

Dataset required: preference dataset

Prompt	Chosen	Rejected
Generate an approximately fifteen-word sentence that describes all this data: Midsummer House eatType restaurant; Midsummer House food Chinese; Midsummer House priceRange moderate; Midsummer House customer rating 3 out of 5; Midsummer House near All Bar One	Midsummer House is a moderately priced Chinese restaurant with a 3/5 customer rating, located near All Bar One.	Sure! Here's a sentence that describes all the data you provided: "Midsummer House is a moderately priced Chinese restaurant with a customer rating of 3 out of 5, located near All Bar One, offering a variety of delicious dishes."
...

 [argilla/distilabel-intel-orca-dpo-pairs](#)

 [argilla/ultrafeedback-binarized-preferences-cleaned](#)

Fine-Tuning Language Models from Human Preferences

Daniel M. Ziegler* Nisan Stiennon* Jeffrey Wu Tom B. Brown
Alec Radford Dario Amodei Paul Christiano Geoffrey Irving

OpenAI

{dmz,nisan,jeffwu,tom,alec,damodei,paul,irving}@openai.com

Abstract

Reward learning enables the application of reinforcement learning (RL) to tasks where reward is defined by human judgment, building a model of reward by asking humans questions. Most work on reward learning has used simulated environments, but complex information about values is often expressed in natural language, and we believe reward learning for language is a key to making RL practical and safe for real-world tasks. In this paper, we build on advances in generative pretraining of language models to apply reward learning to four natural language tasks: continuing text with positive sentiment or physically descriptive language, and summarization tasks on the TL;DR and CNN/Daily Mail datasets. For stylistic continuation we achieve good results with only 5,000 comparisons evaluated by humans. For summarization, models trained with 60,000 comparisons copy whole sentences from the input but skip irrelevant preamble; this leads to reasonable ROUGE scores and very good performance according to our human labelers, but may be exploiting the fact that labelers rely on simple heuristics.

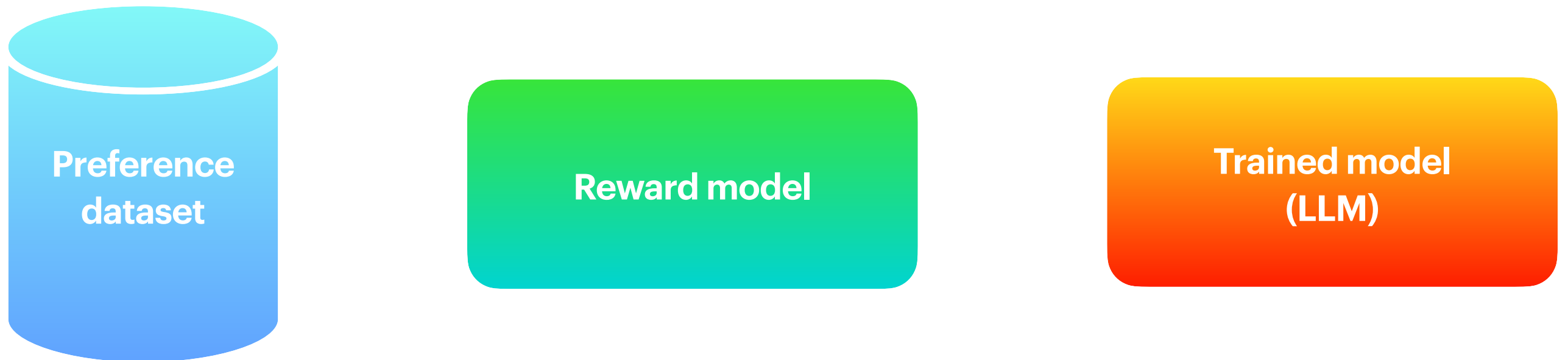
plex goals to AI agents are likely to both involve and require natural language, which is a rich medium for expressing value-laden concepts. Natural language is particularly important when an agent must communicate back to a human to help provide a more accurate supervisory signal (Irving et al., 2018; Christiano et al., 2018; Leike et al., 2018).

Natural language processing has seen substantial recent advances. One successful method has been to pretrain a large generative language model on a corpus of unsupervised data, then fine-tune the model for supervised NLP tasks (Dai and Le, 2015; Peters et al., 2018; Radford et al., 2018; Khandelwal et al., 2019). This method often substantially outperforms training on the supervised datasets from scratch, and a single pretrained language model often can be fine-tuned for state of the art performance on many different supervised datasets (Howard and Ruder, 2018). In some cases, fine-tuning is not required: Radford et al. (2019) find that generatively trained models show reasonable performance on NLP tasks with no additional training (zero-shot).

There is a long literature applying reinforcement learning to natural language tasks. Much of this work uses algorithmically defined reward functions such as BLEU for translation (Ranzato et al., 2015; Wu et al., 2016), ROUGE for summarization (Ranzato et al., 2015; Paulus et al., 2017; Wu and Hu, 2018; Gao et al., 2019b), music theory-based rewards

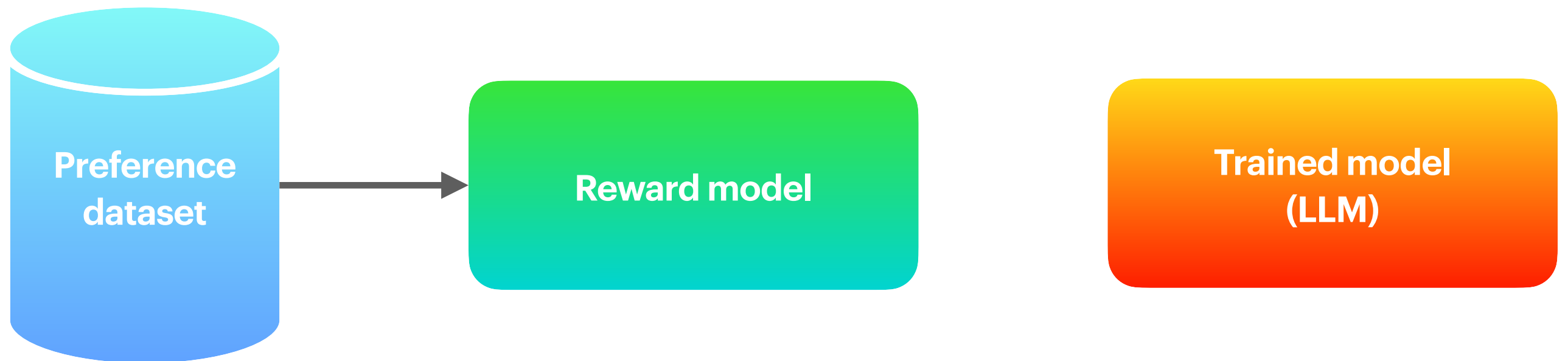
Source: Daniel, M. Z., Nisan, S., Jeffrey, W., Tom, B. B., Alec, R., Dario, A. et al. (2020). *Fine-Tuning Language Models from Human Preferences*.

RL is a multistep training process : PPO example



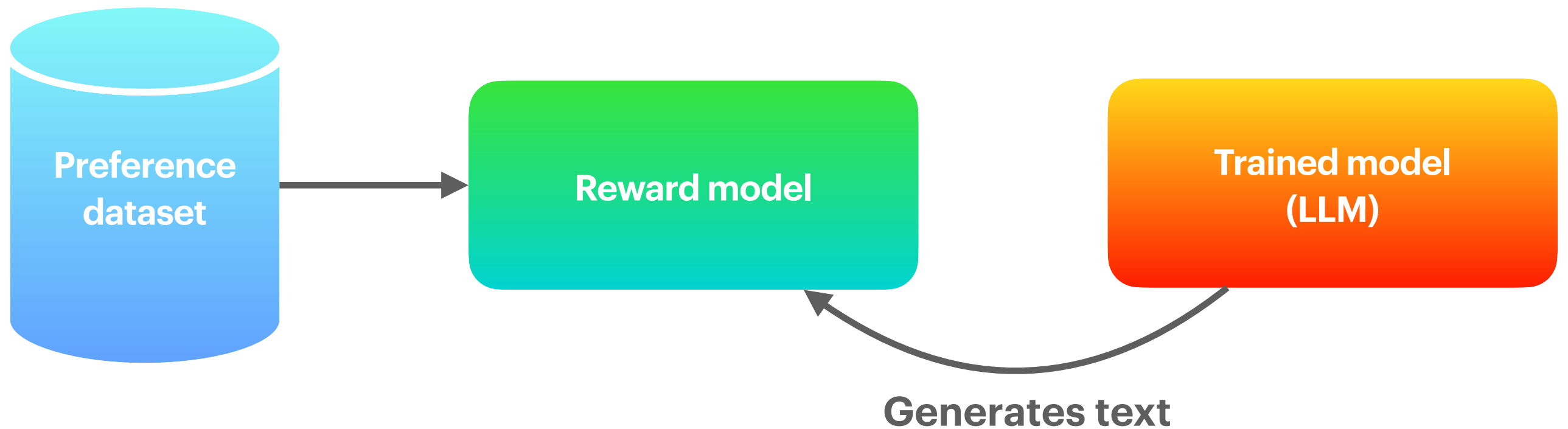
Source: Labonne, M. (2023). ML Blog - Fine-tune Mistral-7b with Direct Preference Optimization. Retrieved Jan 22 2024, 2024, from https://mlabonne.github.io/blog/posts/Fine_tune_Mistral_7b_with_DPO.html.

RL is a multistep training process : PPO example



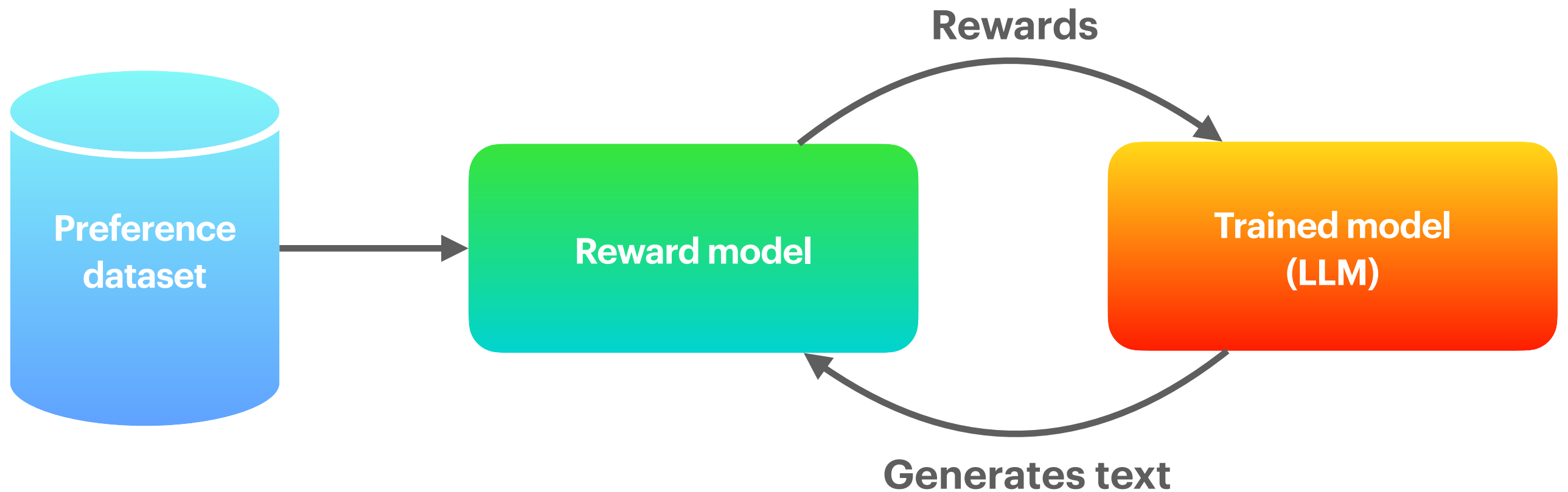
Source: Labonne, M. (2023). ML Blog - Fine-tune Mistral-7b with Direct Preference Optimization. Retrieved Jan 22 2024, 2024, from https://mlabonne.github.io/blog/posts/Fine_tune_Mistral_7b_with_DPO.html.

RL is a multistep training process : PPO example



Source: Labonne, M. (2023). ML Blog - Fine-tune Mistral-7b with Direct Preference Optimization. Retrieved Jan 22 2024, 2024, from https://mlabonne.github.io/blog/posts/Fine_tune_Mistral_7b_with_DPO.html.

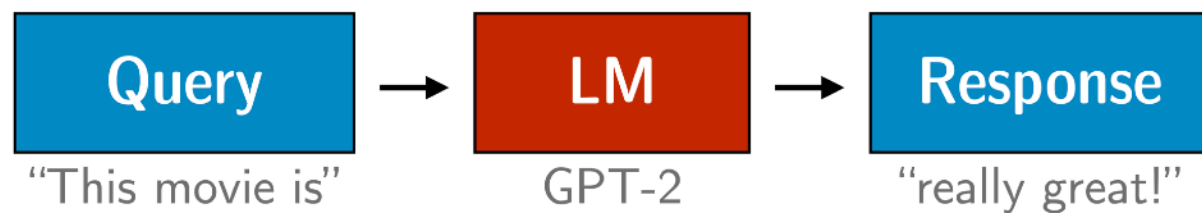
RL is a multistep training process : PPO example



Source: Labonne, M. (2023). ML Blog - Fine-tune Mistral-7b with Direct Preference Optimization. Retrieved Jan 22 2024, 2024, from https://mlabonne.github.io/blog/posts/Fine_tune_Mistral_7b_with_DPO.html.

RL is a multistep training process : PPO example

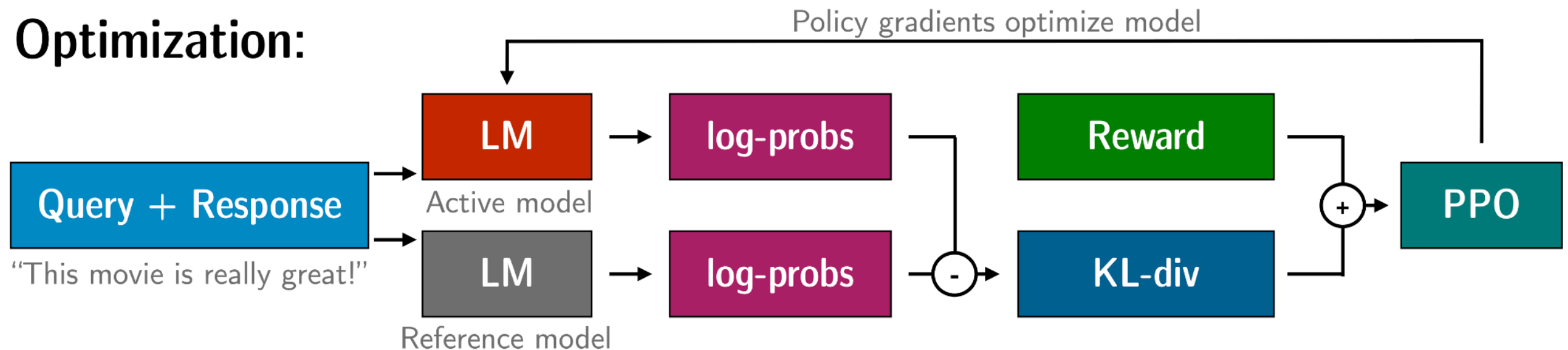
Rollout:



Evaluation:



Optimization:



Source: Hugging Face TRL library documentation, <https://github.com/huggingface/trl>

Direct Preference Optimization: Your Language Model is Secretly a Reward Model

Rafael Rafailov*[†]

Archit Sharma*[†]

Eric Mitchell*[†]

Stefano Ermon^{†‡}

Christopher D. Manning[†]

Chelsea Finn[†]

[†]Stanford University [‡]CZ Biohub
{rafailov,architsh,eric.mitchell}@cs.stanford.edu

Abstract

While large-scale unsupervised language models (LMs) learn broad world knowledge and some reasoning skills, achieving precise control of their behavior is difficult due to the completely unsupervised nature of their training. Existing methods for gaining such steerability collect human labels of the relative quality of model generations and fine-tune the unsupervised LM to align with these preferences, often with reinforcement learning from human feedback (RLHF). However, RLHF is a complex and often unstable procedure, first fitting a reward model that reflects the human preferences, and then fine-tuning the large unsupervised LM using reinforcement learning to maximize this estimated reward without drifting too far from the original model. In this paper we introduce a new parameterization of the reward model in RLHF that enables extraction of the corresponding optimal policy in closed form, allowing us to solve the standard RLHF problem with only a simple classification loss. The resulting algorithm, which we call *Direct Preference Optimization* (DPO), is stable, performant, and computationally lightweight, eliminating the need for sampling from the LM during fine-tuning or performing significant hyperparameter tuning. Our experiments show that DPO can fine-tune LMs to align with human preferences as well as or better than existing methods. Notably, fine-tuning with DPO exceeds PPO-based RLHF in ability to control sentiment of generations, and matches or improves response quality in summarization and single-turn dialogue while being substantially simpler to implement and train.

Source: Rafael Rafailov, A. S., Eric Mitchell, Stefano Ermon, Christopher D. Manning, Chelsea Finn. (2023). Direct Preference Optimization: Your Language Model is Secretly a Reward Model. Retrieved 12 Jan 2024, <https://arxiv.org/abs/2305.18290v2>.

DPO optimizes for human preferences while avoiding reinforcement learning

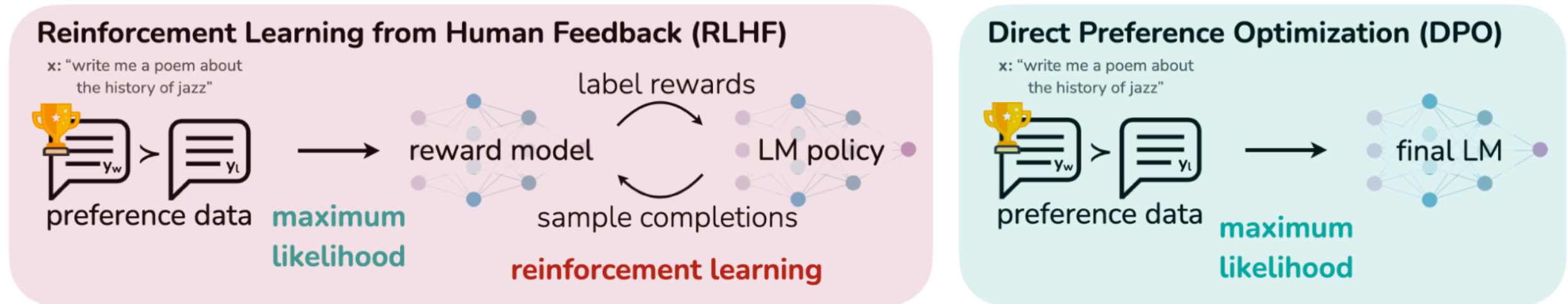
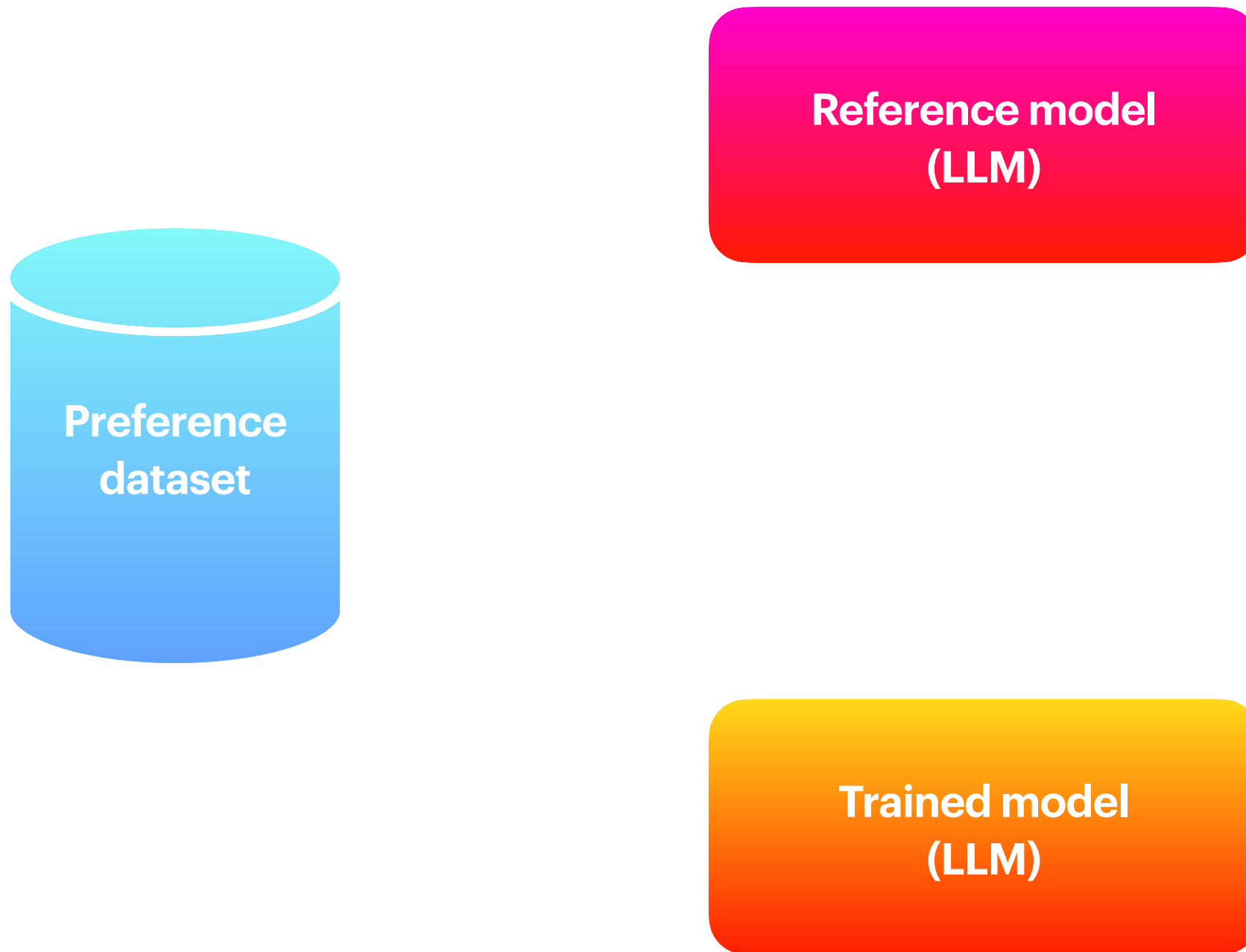


Figure 1: **DPO optimizes for human preferences while avoiding reinforcement learning.** Existing methods for fine-tuning language models with human feedback first fit a reward model to a dataset of prompts and human preferences over pairs of responses, and then use RL to find a policy that maximizes the learned reward. In contrast, DPO directly optimizes for the policy best satisfying the preferences with a simple classification objective, fitting an *implicit* reward model whose corresponding optimal policy can be extracted in closed form.

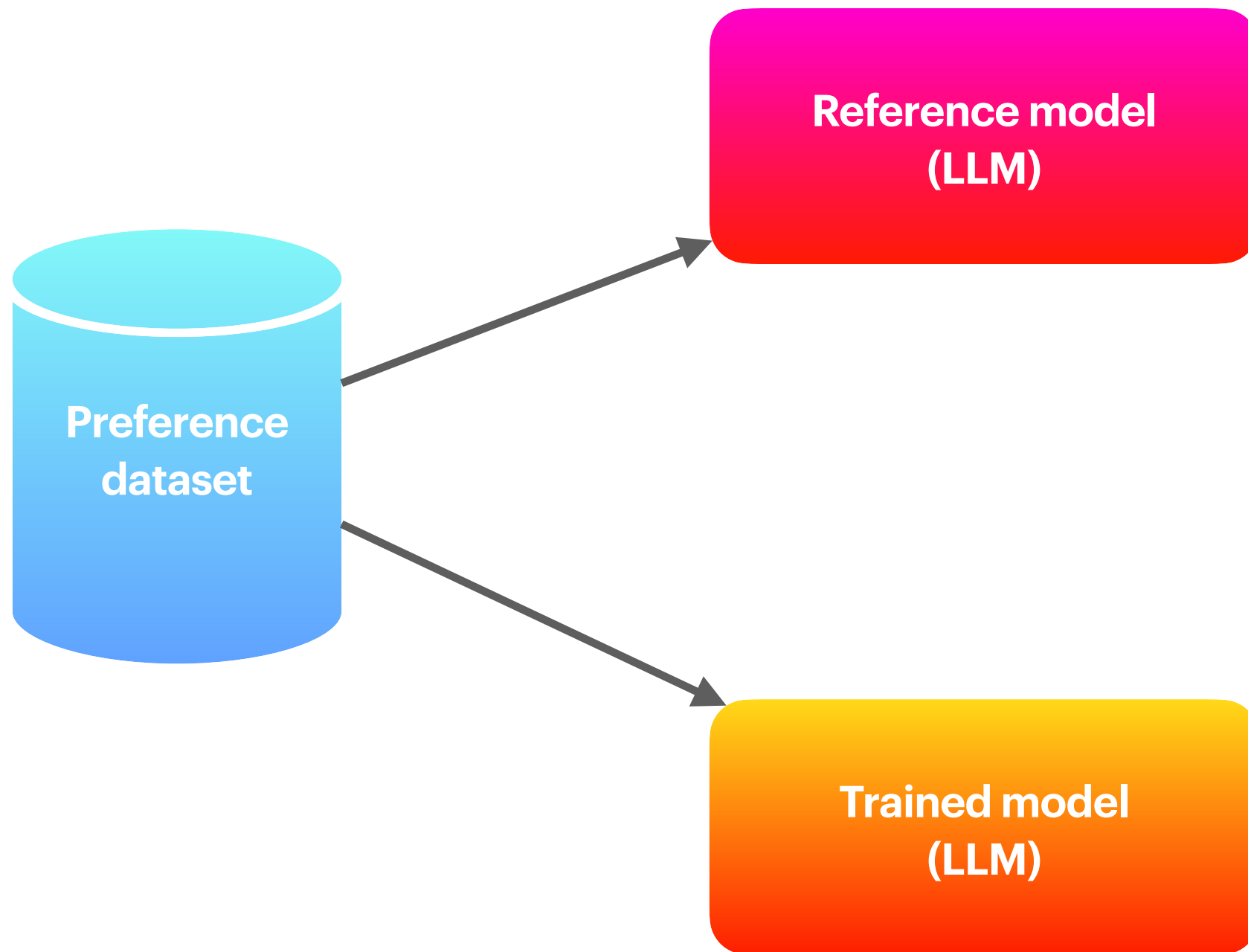
Source: Rafael Rafailov, A. S., Eric Mitchell, Stefano Ermon, Christopher D. Manning, Chelsea Finn. (2023). Direct Preference Optimization: Your Language Model is Secretly a Reward Model. Retrieved 12 Jan 2024, <https://arxiv.org/abs/2305.18290v2>.

Direct preference optimisation makes it easier to train



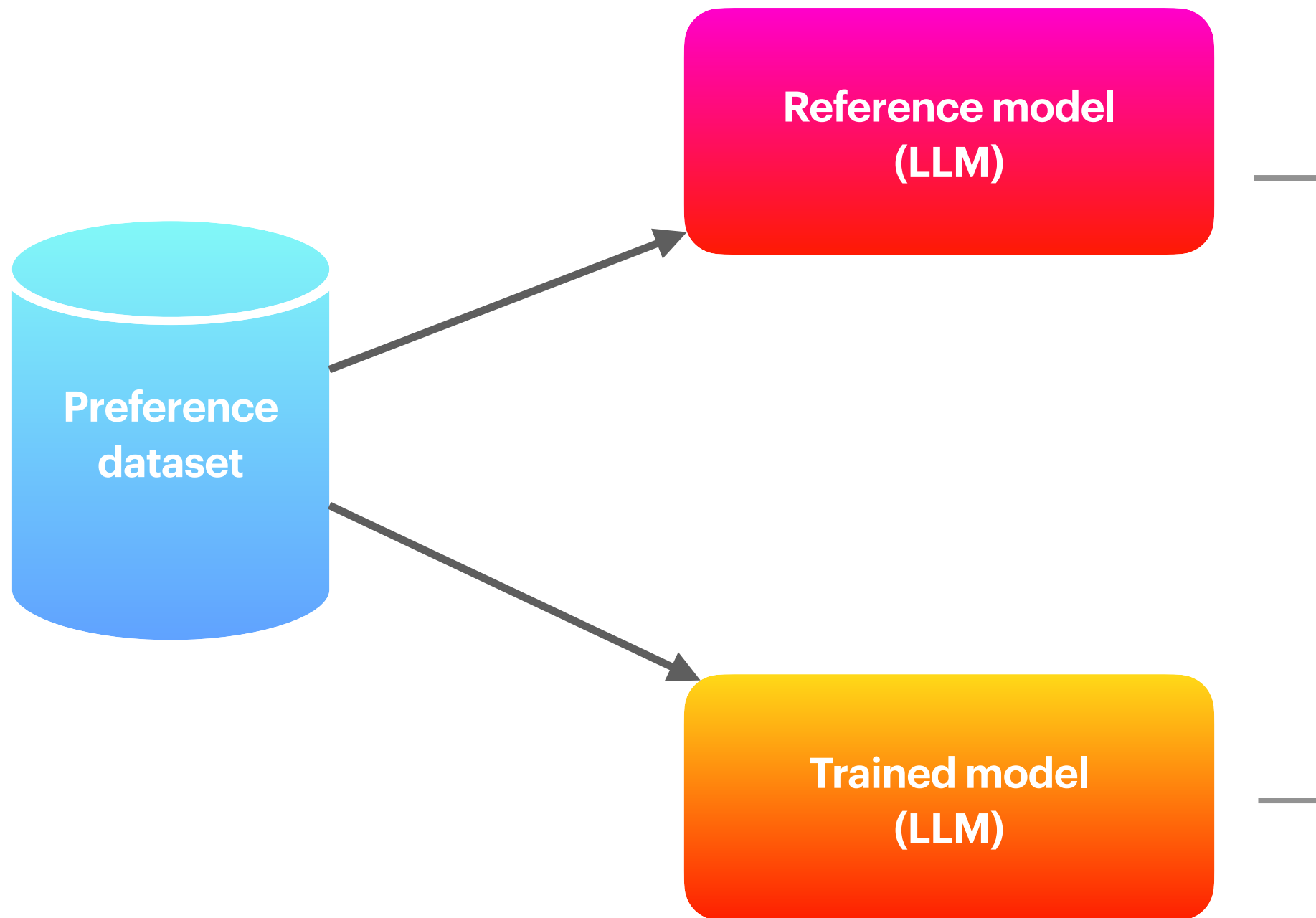
Source: Labonne, M. (2023). ML Blog - Fine-tune Mistral-7b with Direct Preference Optimization. Retrieved Jan 22 2024, 2024, from https://mlabonne.github.io/blog/posts/Fine_tune_Mistral_7b_with_DPO.html.

Direct preference optimisation makes it easier to train



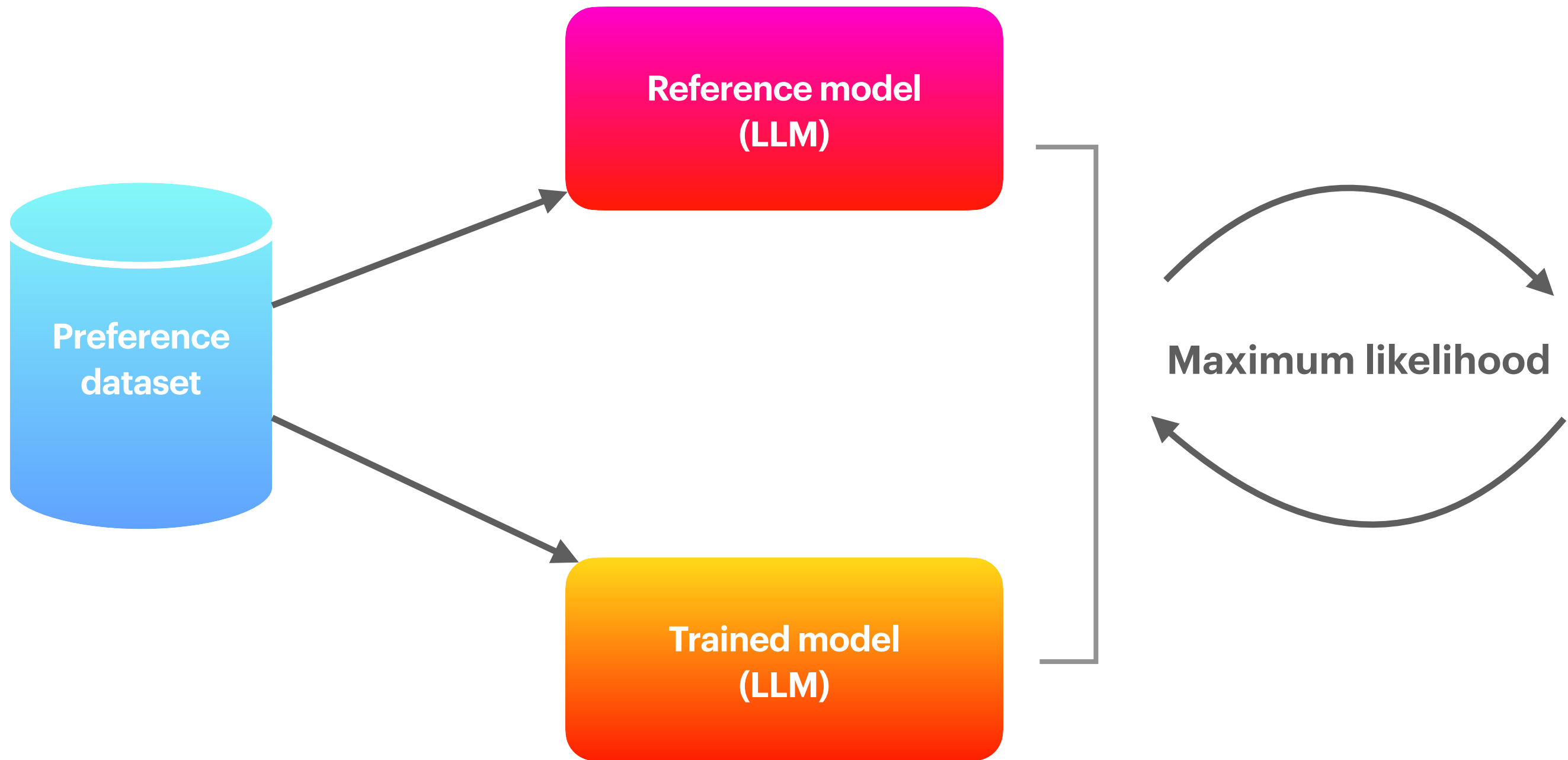
Source: Labonne, M. (2023). ML Blog - Fine-tune Mistral-7b with Direct Preference Optimization. Retrieved Jan 22 2024, 2024, from https://mlabonne.github.io/blog/posts/Fine_tune_Mistral_7b_with_DPO.html.

Direct preference optimisation makes it easier to train






Source: Labonne, M. (2023). ML Blog - Fine-tune Mistral-7b with Direct Preference Optimization. Retrieved Jan 22 2024, 2024, from https://mlabonne.github.io/blog/posts/Fine_tune_Mistral_7b_with_DPO.html.

Direct preference optimisation makes it easier to train



Source: Labonne, M. (2023). ML Blog - Fine-tune Mistral-7b with Direct Preference Optimization. Retrieved Jan 22 2024, 2024, from https://mlabonne.github.io/blog/posts/Fine_tune_Mistral_7b_with_DPO.html.

RL - Useful libs

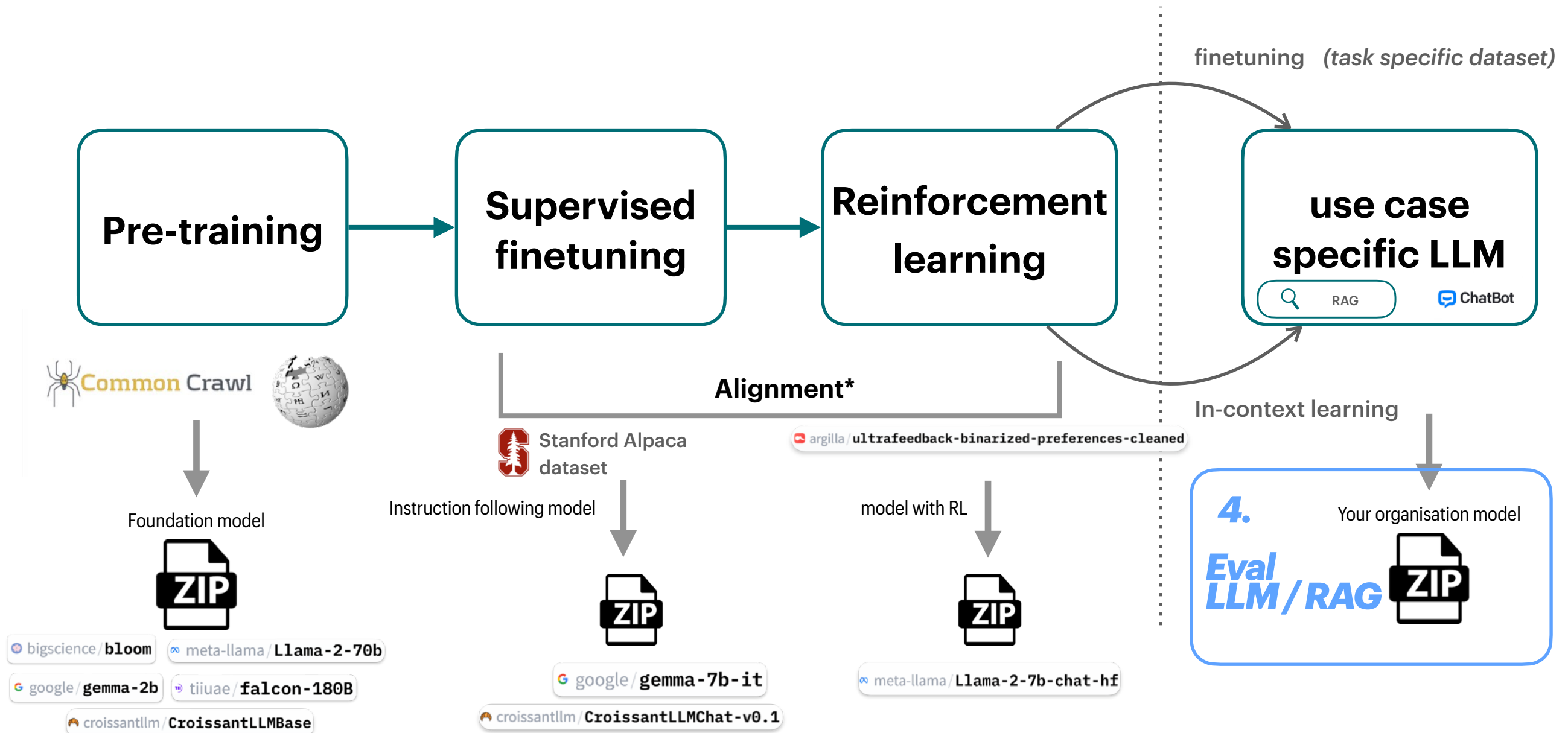
Name	Scope	Url
 Unloth	Finetuning with several approaches LoRa, QLoRA, RLHF (DPO)	https://github.com/unslorhai/unloth
 Hugging Face TRL	Implementation of most RL approaches, regularly updated	https://huggingface.co/docs/trl
Hugging Face DPO Trainer 	HF DPO implementation	https://huggingface.co/docs/trl/dpo_trainer

Pt 4.

LLM systems evaluation

Tips for relevant evaluation

LLM training pipeline

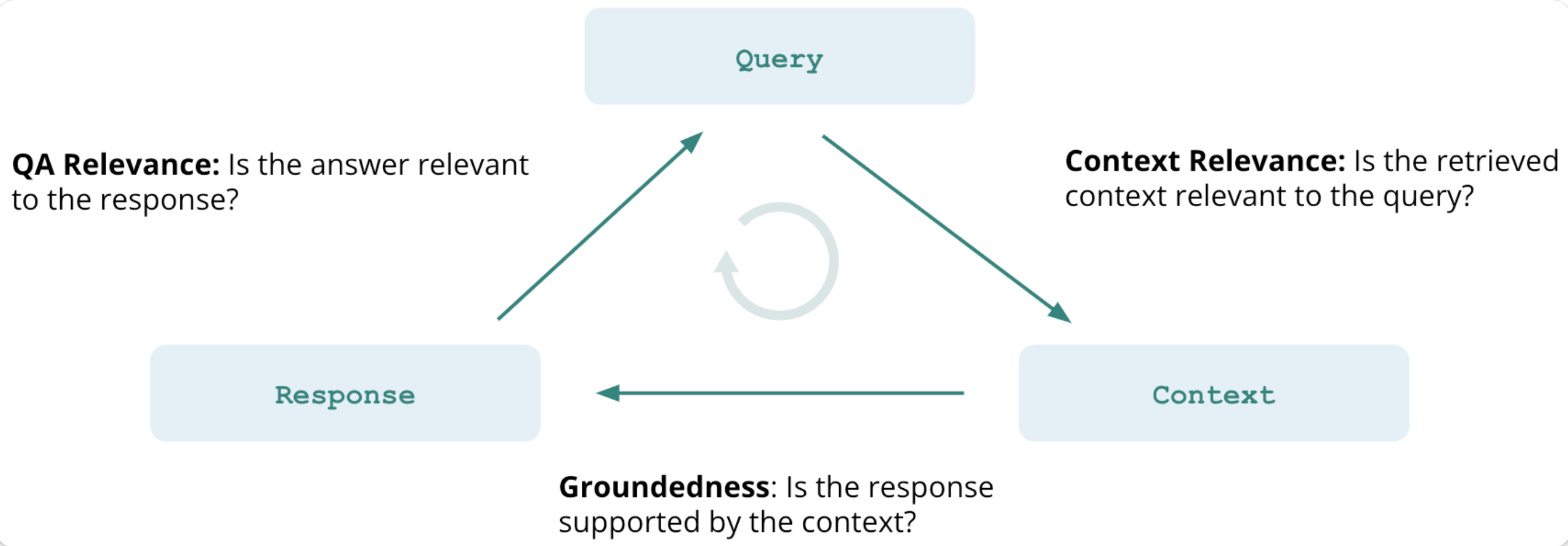


Adapted from Wolfe, C. (2023). Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More. Retrieved Feb 2024, <https://cameronwolfe.substack.com/p/easily-train-a-specialized-llm-peft>.

Evaluation principles

- **SOTA models in NLP change every month (week) → model value decreases**
- **Use-case specific curated dataset are the moat → eval dataset value increases**
- **Craft eval metrics aligned with your business objectives → academic benchmark metrics rarely reflect users perceived value**

Error analysis - learning from bad generations - example of RAG



Note: highlights the importance of capturing good and bad predictions in your product to perform error analysis (ML Ops tooling) prompt/answer/good/bad, if multiple answers capture chosen answer/rejected answer for RLHF.

Recommendation: Ng, Andrew (2017). Carrying out error analysis. Retrieved Feb 2024, <https://cs230.stanford.edu/files/C3M2.pdf>

Source: Trulens - RAG evaluation dimensions, <https://docs.pinecone.io/docs/trulens>

LLM fine-tuning & evaluation

PRACTICAL INSIGHTS

Quiz w/ prize



kahoot time!

Appendix

LLM training pipeline

Model Details

Note: Use of this model is governed by the Meta license. In order to download the model weights and tokenizer, please visit the [website](#) and accept our License before requesting access here.

Meta developed and publicly released the Llama 2 family of large language models (LLMs), a **collection of pretrained and fine-tuned generative text models** ranging in scale from 7 billion to 70 billion parameters. Our fine-tuned LLMs, called Llama-2-Chat, are optimized for dialogue use cases. Llama-2-Chat models outperform open-source chat models on most benchmarks we tested, and in our human evaluations for helpfulness and safety, are on par with some popular closed-source models like ChatGPT and PaLM.

Source: Hugging face, meta-llama org. Llama-2-7b-chat-hf model card, <https://huggingface.co/meta-llama/Llama-2-7b-chat-hf>

LLM training pipeline

Model Details

Note: Use of this model is governed by the Meta license. In order to download the model weights and tokenizer, please visit the [website](#) or [requesting access here](#).

Meta developed and publicly released the Llama 2 (LLMs), a collection of pretrained and fine-tuned models that scale from 7 billion to 70 billion parameters. Our Llama 2 Chat, are optimized for dialogue use cases. Llama 2 source chat models on most benchmarks we test for helpfulness and safety, are on par with some top-performing ChatGPT and PaLM.

Model Developers Meta

Variations Llama 2 comes in a range of parameter sizes — 7B, 13B, and 70B — as well as pretrained and fine-tuned variations.

Input Models input text only.

Output Models generate text only.

Model Architecture Llama 2 is an auto-regressive language model that uses an optimized transformer architecture. The tuned versions use supervised fine-tuning (SFT) and reinforcement learning with human feedback (RLHF) to align to human preferences for helpfulness and safety.

Source: Hugging face, meta-llama org. Llama-2-7b-chat-hf model card, <https://huggingface.co/meta-llama/Llama-2-7b-chat-hf>

FoMo Architecture

Insertion of Fine-tuning Parameters

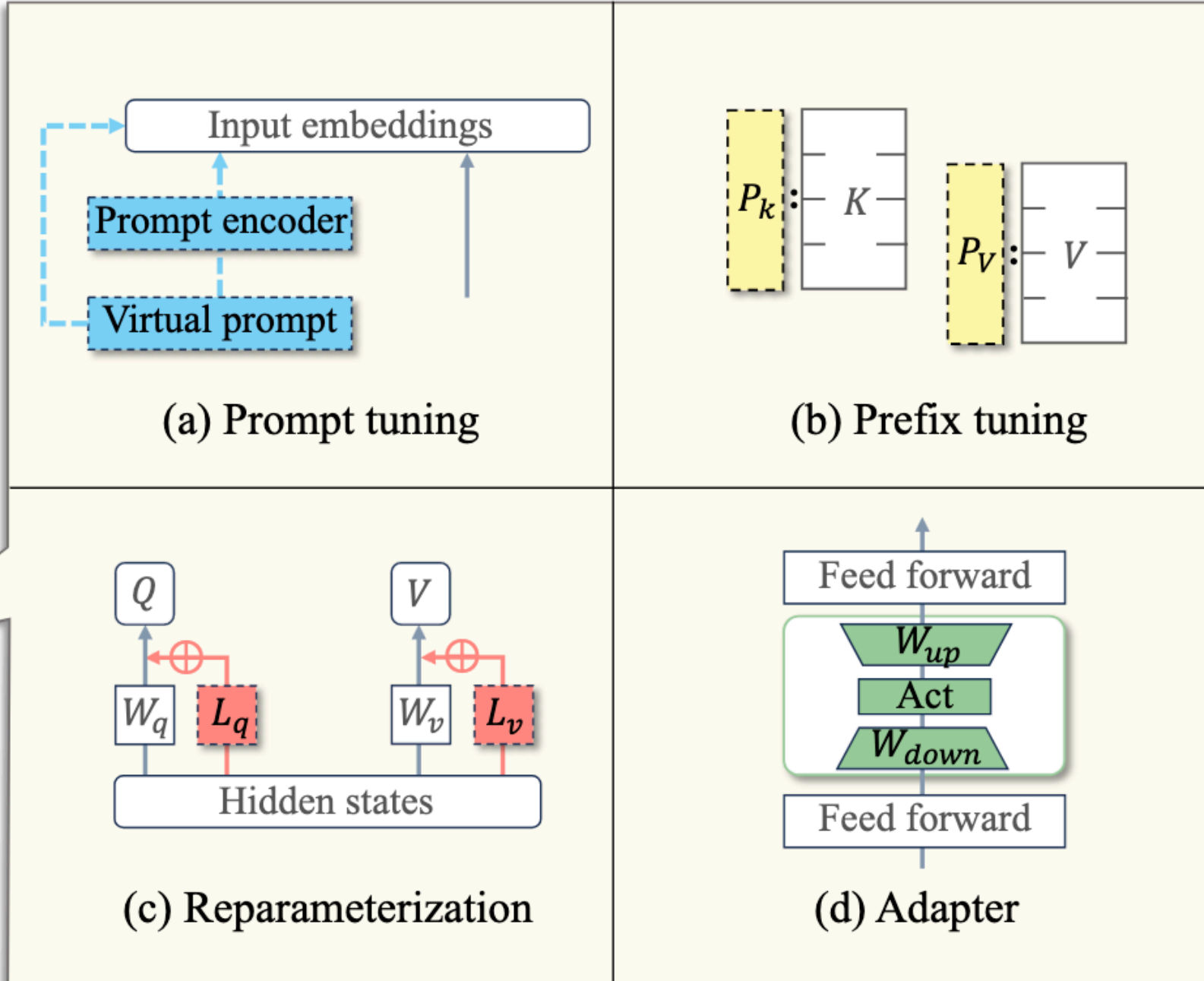
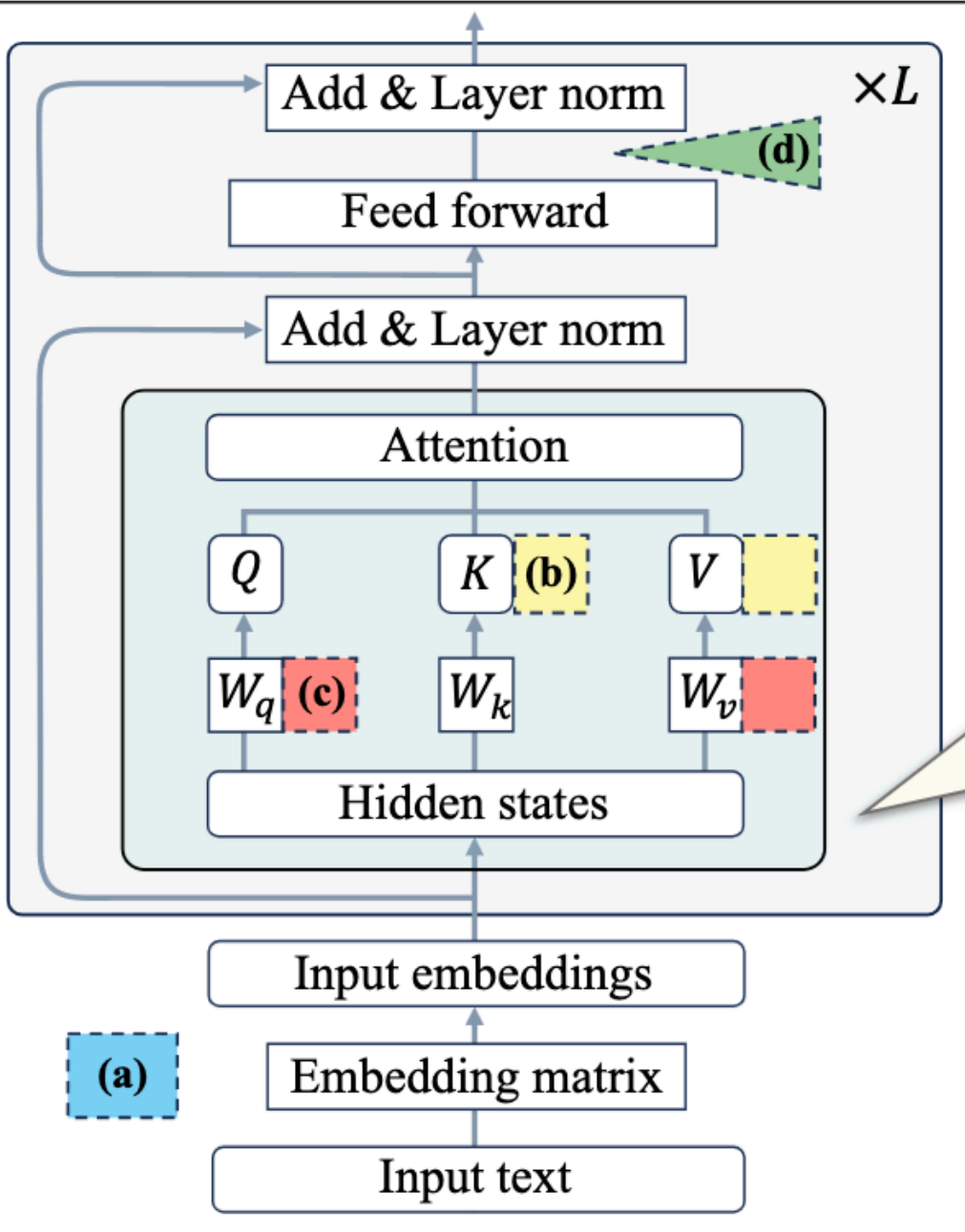
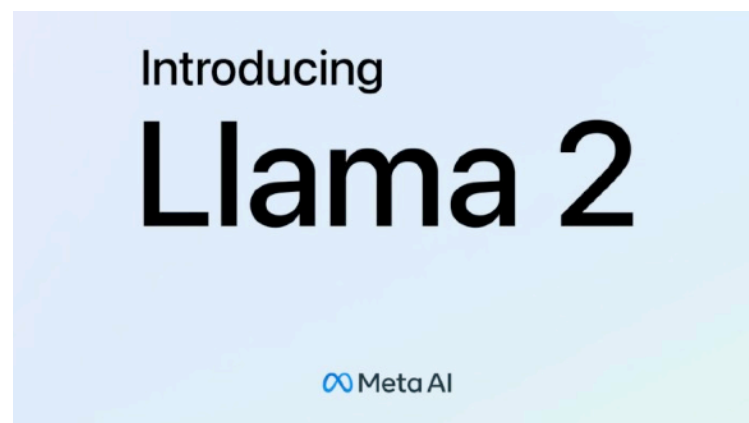


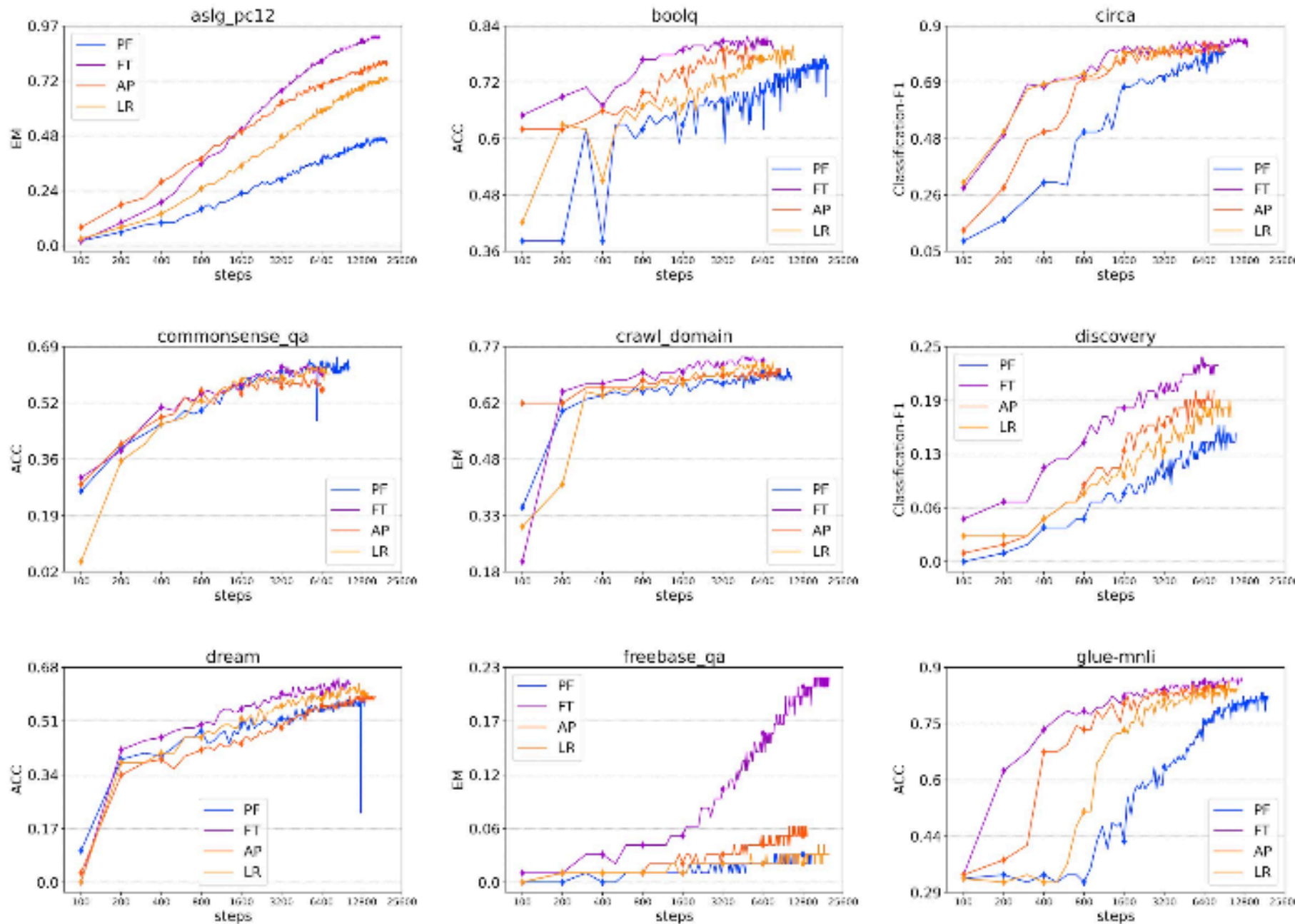
Fig. 1. State-of-the-Art PEFT techniques.

Source: Device-Edge Cooperative Fine-Tuning of Foundation Models as a 6G Service, <https://arxiv.org/ftp/arxiv/papers/2310/2310.18602.pdf>

Ever evolving landscape



If you have the resources, full fine-tuning tends to work the best.



PF: prefix tuning
FT: full fine-tuning
AP: adapter
LR: LoRA

This survey overall found:
Full fine-tuning >
LoRA >
Adapters >
Prefix Tuning >
Prompt Tuning
In terms of performance.

Plots for many more tasks can be found in the paper.

Source: University, Carnegie Mellon (2023). Parameter Efficient Tuning - 11-667: LARGE LANGUAGE MODELS: METHODS AND APPLICATIONS. 2023, from https://www.andrew.cmu.edu/course/11-667/lectures/W4L2_PETM.pptx.pdf