# Portable Reward Tuning: Towards Reusable Fine-Tuning across Different Pretrained Models

Daiki Chijiwa*, Taku Hasegawa*, Kyosuke Nishida, Kuniko Saito, Susumu Takeuchi

* equal contribution

ICML 2025 accepted

2025年 7月 9日

# 自己紹介

## 長谷川 拓 （プレゼンター）

■ 経歴

- 大阪府立大（中退）進化計算
- 大阪府立大（修士・博士・PD）最適化アルゴリズムと機械学習
- Freiburg 大学（Invited Researcher）NNのアーキテクチャサーチ

■ 現所属：NTT人間情報研究所 視覚言語基盤モデル構築・知識転移

## 千々和 大輝

■ 経歴

- 旧・東京工業大学　学士（数学） 代数幾何学・ホッジ理論
- 東京大学　修士（数理科学） 同上

■現所属：NTT コンピュータ＆データサイエンス研究所 深層学習・言語モデルの研究

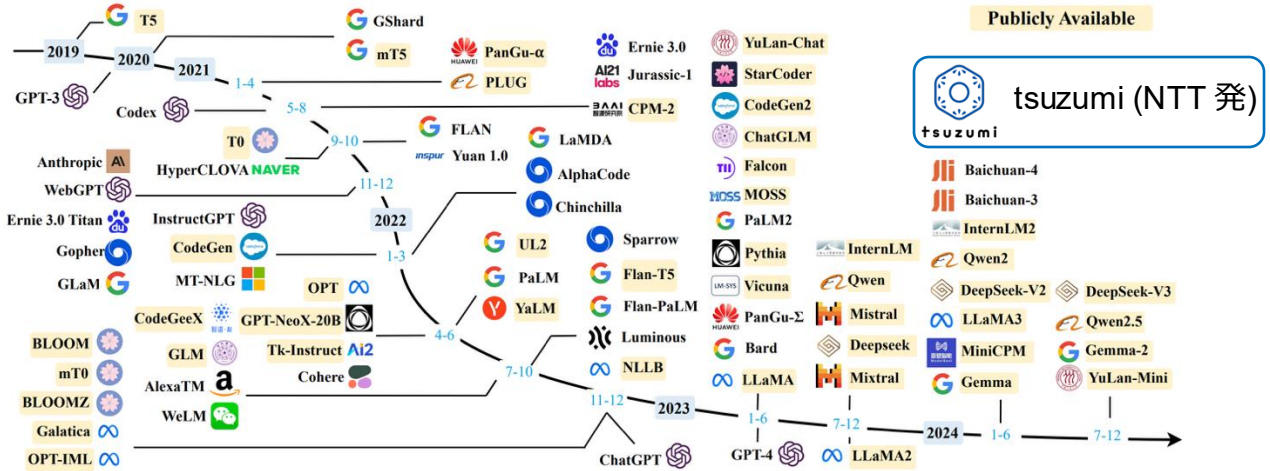
# 目次

# 背景

# 基盤モデル

基盤モデルは、言語・画像・音声など多様な領域で活用が拡大

- 特に大規模言語モデル（LLM）は近年、急速に発展
- 研究・産業の両面で不可欠な技術に

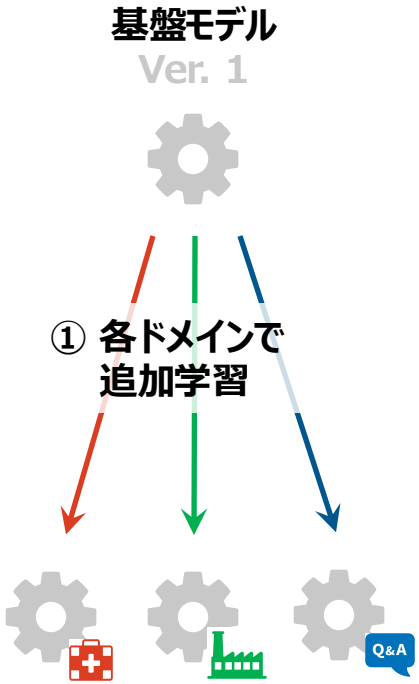


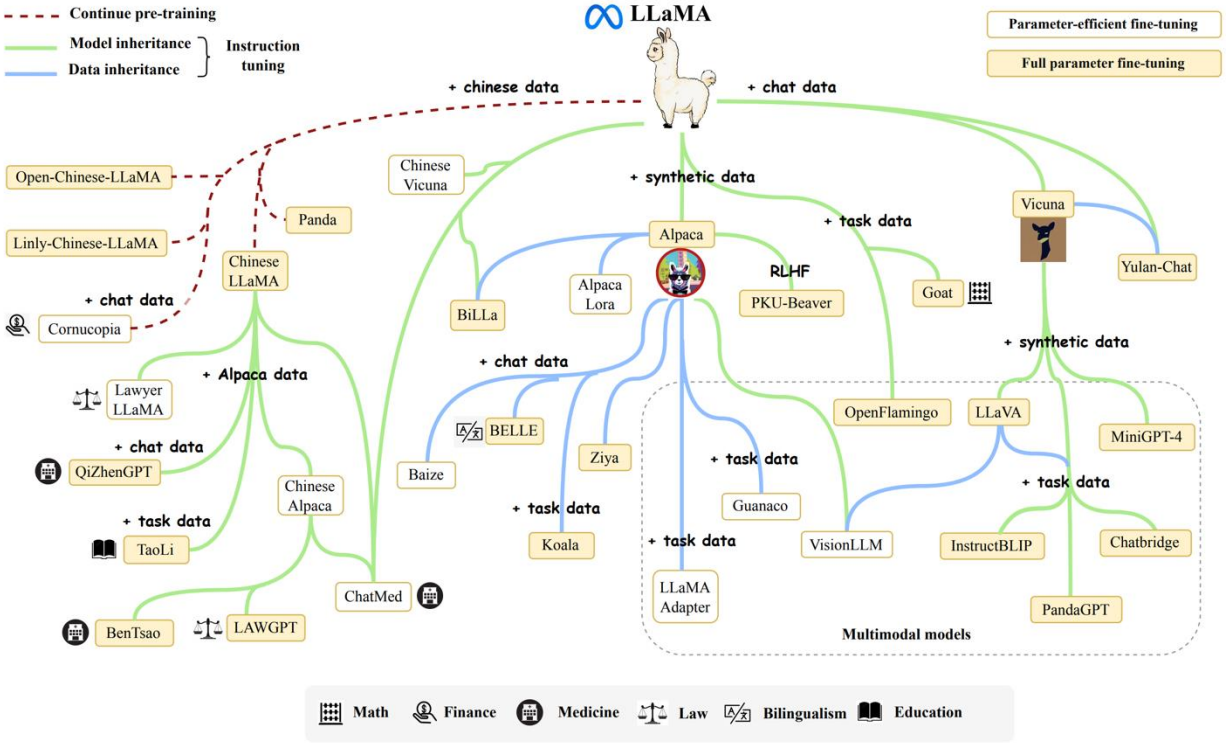



NTT版大規模言語モデル「tsuzumi」

A Survey of Large Language Models, Zhao et al., https://arxiv.org/abs/2303.18223

# 特化モデルの学習

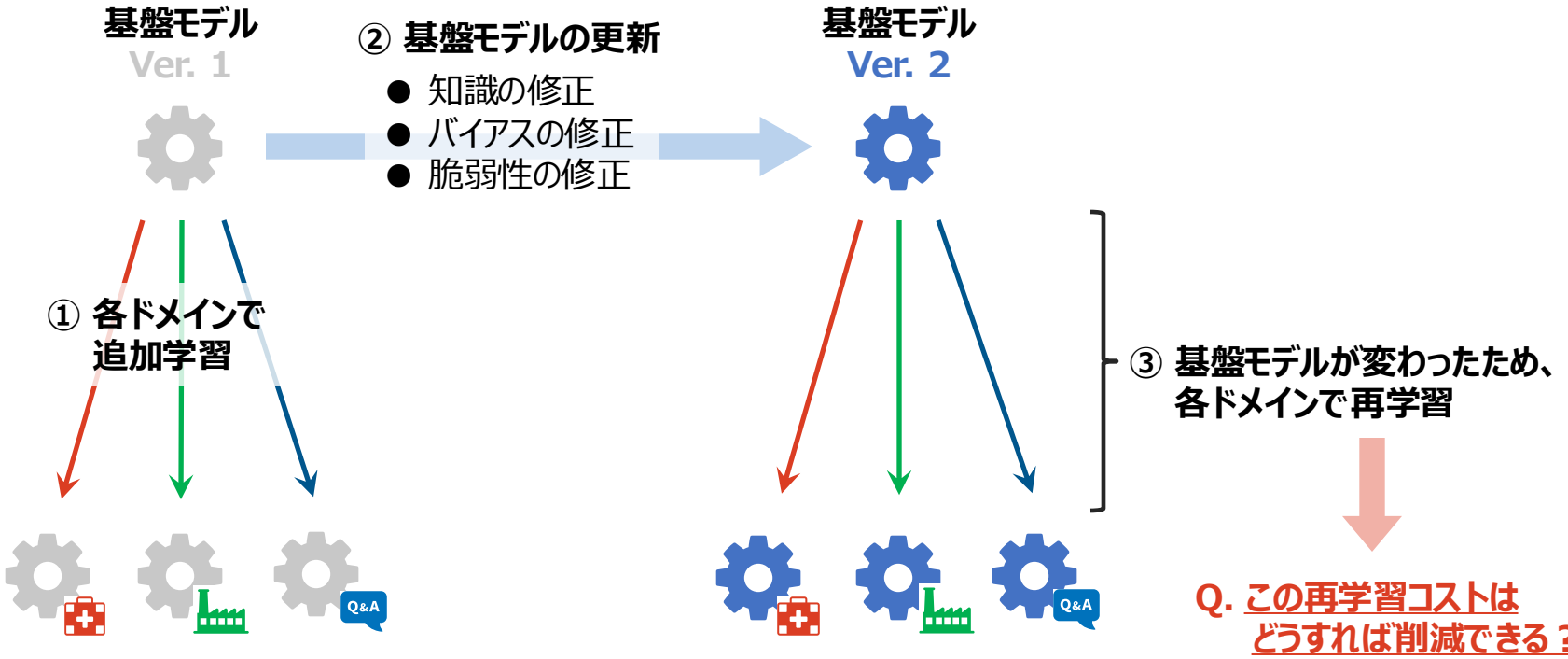





A Survey of Large Language Models, Zhao et al., https://arxiv.org/abs/2303.18223

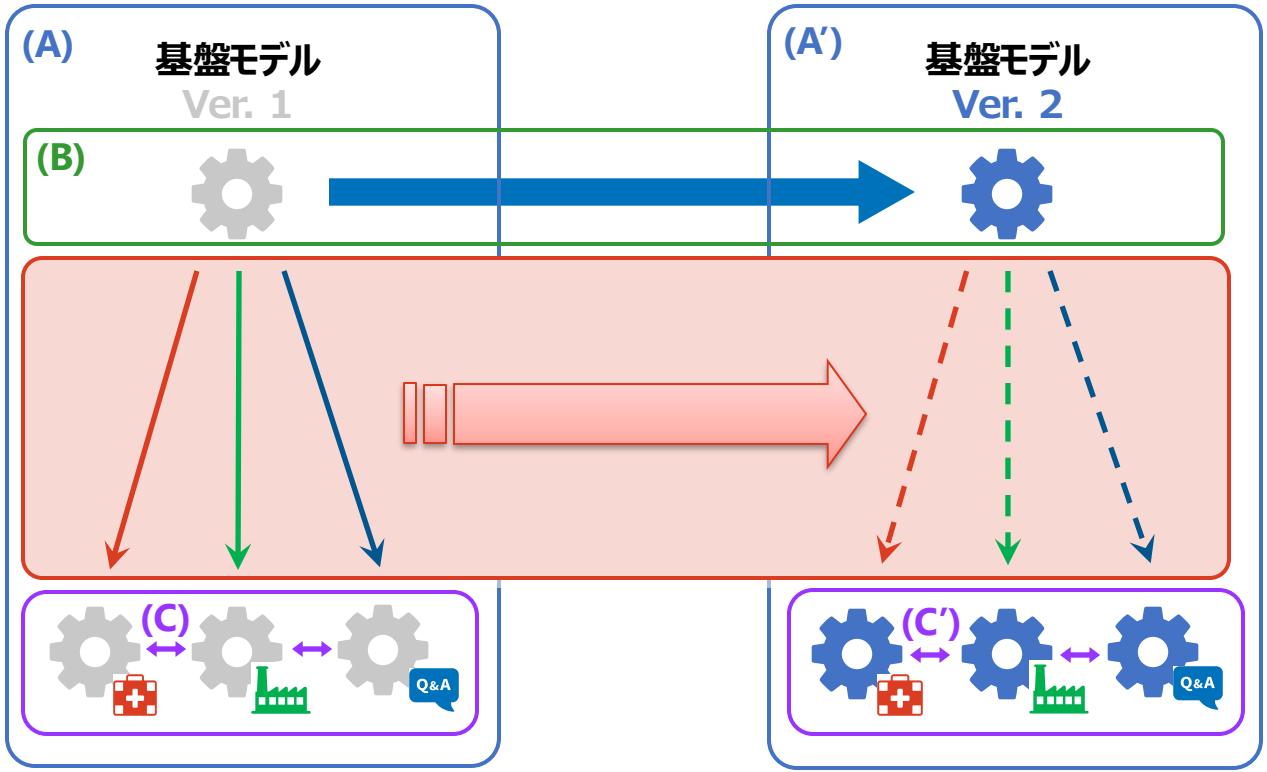
# モデルアップデート






A Survey of Large Language Models, Zhao et al., https://arxiv.org/abs/2303.18223

# モデルアップデート



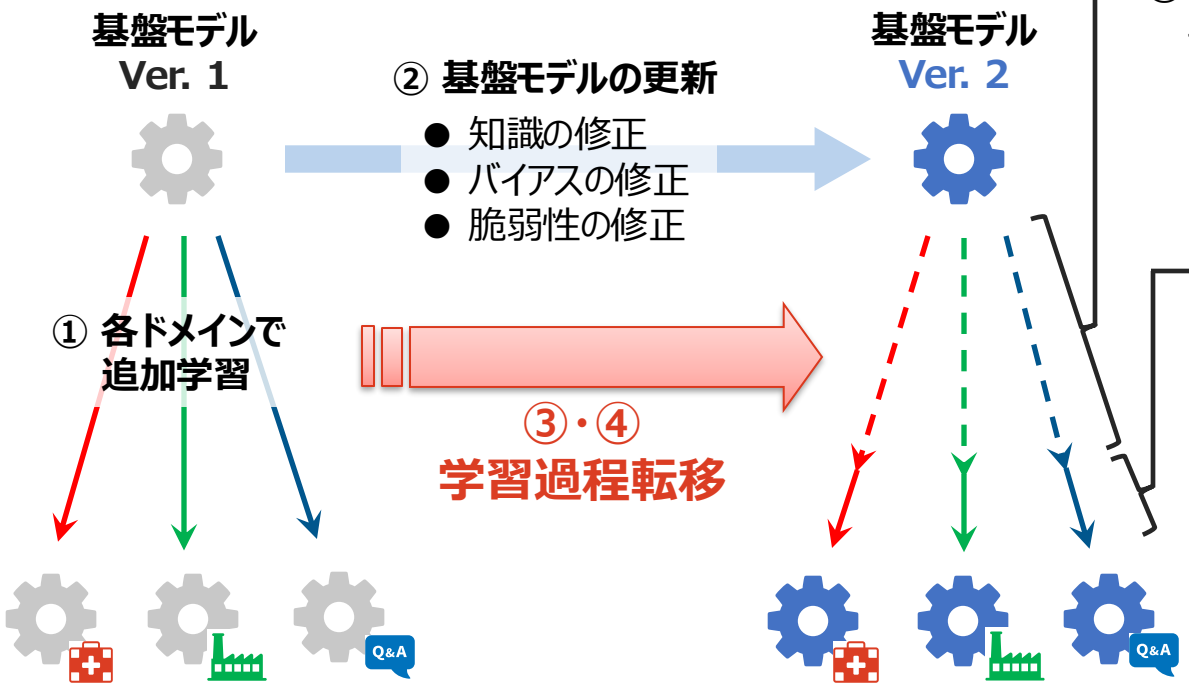

**(A) 転移学習、ドメイン適合**
→ 各ドメインの学習を効率化

**(B) 継続学習、再学習、etc.**
→ 基盤モデルの性能を改善

**(C) モデルマージ、タスク算術**
→ 同じ基盤モデルをベースとする特化モデルが複数あるときに、それらの演算により新たなモデルを作成

**学習転移**
**異なる基盤モデル間で学習結果を転移**させ、各ドメインでの再学習をコスト削減する新たな枠組み

# 学習過程転移




③ **数十秒～数分程度**で学習転移
→ 実際の学習よりも超高速

④ Vision系モデルでの追加学習
→ 従来の**3倍以上の高速化**

**課題：**

- パラメータ空間内で転移を行うため、**モデル構造の一致**を要請
- ③の転移だけでは再学習時の精度には及ばず
  少量ながら**④の追加学習は不可欠**

→ **導入コストとして無視できない障壁**



Transferring Learning Trajectories of Neural Networks, Daiki Chijiwa, ICLR2024

# 関連研究

# 報酬最大化としての Fine-Tuning

分類モデル：$\pi_\theta(y|x) \coloneqq \mathrm{softmax}(f(x;\theta))$, 入力 $x \in \mathcal{X}$, 出力 $y \in \mathcal{Y}$

例）
画像分類： $x$ は画像、 $y$ は対応するラベル、 $f(x;\theta)$ は CNN や ViT
言語生成： $x$ はトークン列 $t_1, \cdots t_k$、 $y$ は次のトークン $t_k$ 、$f(x;\theta)$ は decoder-only Transformer

- 従来の Fine-Tuning (FT) は、次のような「KL制約つき報酬最大化」を行っているとみなせる：

基盤モデル $\pi_{pt}(y|x)$

FTモデル $\pi_{ft}(y|x)$

$$\max_{\pi(\cdot|x)} \mathbb{E}_{y\sim\pi(y|x)}\left[\log\left(\frac{\pi_{\mathrm{ft}}(y|x)}{\pi_{\mathrm{pt}}(y|x)}\right)\right] - D_{KL}\left(\pi(\cdot\,|x) \,\|\, \pi_{\mathrm{pt}}(\cdot\,|x)\right)$$

← 「基盤モデルから離れすぎない」ことを示す正則化項

↑ 期待値の中身は
**タスク報酬を表す、暗黙的な報酬関数**
とみなせる。

解析解

$$\pi(y|x) = \frac{1}{Z(x)}\pi_{\mathrm{pt}}(y|x)\exp\left(\log\left(\frac{\pi_{\mathrm{ft}}(y|x)}{\pi_{\mathrm{pt}}(y|x)}\right)\right) = \pi_{\mathrm{ft}}(y|x)$$

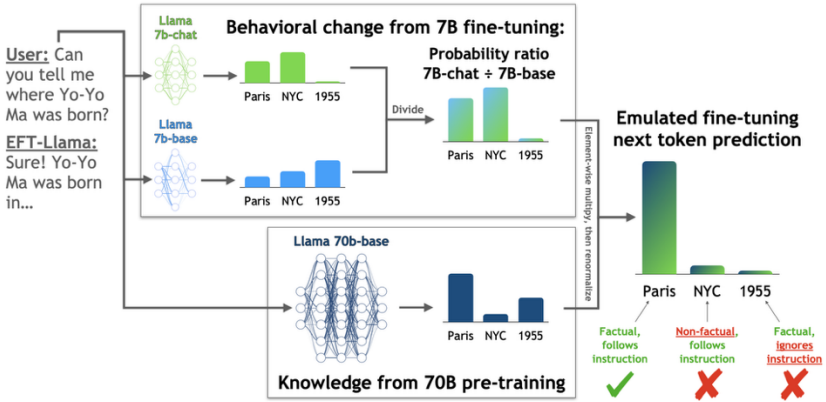
最適化問題を解くと
FT モデルが復元される

# Emulated Fine-tuning

An Emulator for Fine-Tuning Large Language Models using Small Language Models, Mitchell et al., ICLR 2024

先ほどの報酬最大化としての Fine-Tuningの定式化において、異なる事前学習 $\tilde{\pi}_{\mathrm{pt}}(\cdot\,|x)$ を与えると以下の解が得られることに着目

$$\underset{\pi(\cdot|x)}{\operatorname{argmax}}\, \mathbb{E}_{y\sim\pi(y|x)}\left[\log\left(\frac{\pi_{\mathrm{ft}}(y|x)}{\pi_{\mathrm{pt}}(y|x)}\right)\right] - D_{KL}\left(\pi(\cdot\,|x)\,\|\,\tilde{\pi}_{\mathrm{pt}}(\cdot\,|x)\right)$$
$$= \frac{1}{\tilde{Z}(x)}\,\tilde{\pi}_{\mathrm{pt}}(y|x)\,\frac{\pi_{\mathrm{ft}}(y|x)}{\pi_{\mathrm{pt}}(y|x)}$$




特徴：追加でタスク特化の学習が不要

課題：推論時に3つのモデルを同時に動かす必要

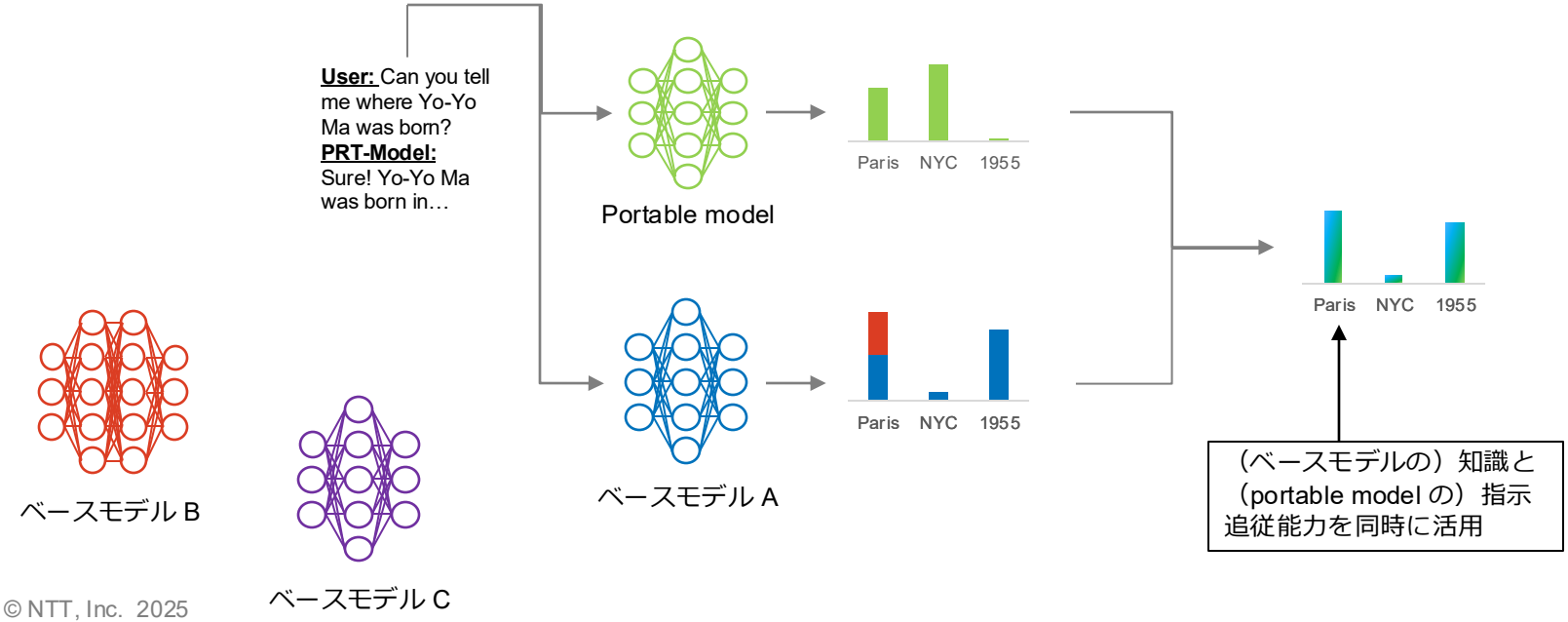
→ 推論コスト増

Emulated Fine-tuning (EFT)：
基盤モデルとFTモデルの関係性から、推論時に新しい基盤モデルに能力を付加することが可能

# 提案手法

# Portable Reward Tuning

【アイデア】使い回し可能なモデルを学習する



# PRTの学習

## Portable Reward Tuning (PRT)

基盤モデル $\pi_{pt}(y|x)$　報酬モデル $r_\theta(x,y)$

PRTモデル $\pi_\theta(x,y)$

「KL制約付き報酬最適化」問題における
報酬関数を直接モデル化＆学習：

$$\max_{\pi(\cdot|x)} \mathbb{E}_{a\sim\pi(y|x)}\left[r_\theta(x,y)\right] - \lambda D_{\mathrm{KL}}\left(\pi(\cdot|x) \,\|\, \pi_{\mathrm{pt}}(\cdot|x)\right).$$

←タスク特化の報酬関数をモデル化

その解として得られる次のモデルの学習を通して、
$\theta$ を学習：

$$\pi_\theta(y|x) = \frac{1}{Z_\theta(x)}\pi_{\mathrm{pt}}(y|x)\exp\left(\frac{1}{\lambda}r_\theta(x,y)\right),$$

ロス関数 $\mathcal{L}(\mathbf{p}, y^*) := \mathrm{CE}(\mathbf{p}, y^*) := -\log \pi_\theta(y^*|x)$ とすると

$$\begin{aligned}
&\arg\min_\theta \frac{1}{|S|}\sum_{(x,y^*)\in S}\mathcal{L}(\mathbf{p}, y^*) && (7)\\
&= \arg\max_\theta \sum_{(x,y^*)\in S}\log \pi_\theta(y^*|x)\\
&= \arg\max_\theta \sum_{(x,y^*)\in S} r_\theta(x,y^*) - V_\theta(x), && (8)\\
&\qquad \text{where } V_\theta(x) := \lambda \log Z_\theta(x)
\end{aligned}$$

この最適化問題を通して $r_\theta$ は何を学習している？

→ イェンゼンの不等式

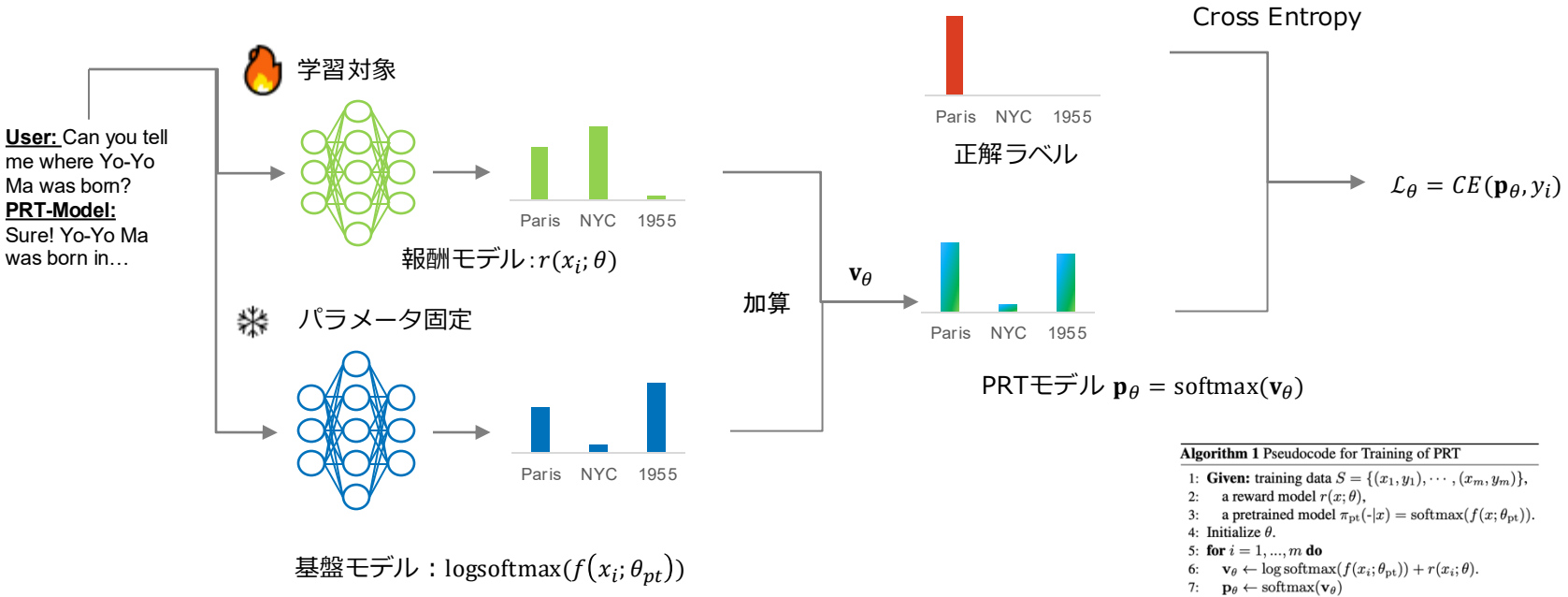
$$\begin{aligned}
&r_\theta(x,y^*) - V_\theta(x)\\
&= r_\theta(x,y^*) - \lambda \log \mathbb{E}_{y\sim\pi_{\mathrm{pt}}(y|x)}\exp(r_\theta(y|x)/\lambda)\\
&\le r_\theta(x,y^*) - \lambda \mathbb{E}_{y\sim\pi_{\mathrm{pt}}(y|x)}\log\exp(r_\theta(y|x)/\lambda)\\
&= r_\theta(x,y^*) - \mathbb{E}_{y\sim\pi_{\mathrm{pt}}(y|x)} r_\theta(y|x). && (9)
\end{aligned}$$

正解 $y^*$の報酬を高くする

事前学習モデルからサンプルされる他のラベルの平均報酬を低くする

# PRTの学習

具体的にどう実装するか？




**Algorithm 1** Pseudocode for Training of PRT

**Given:** training data $S = \{(x_1, y_1), \cdots, (x_m, y_m)\}$,
  a reward model $r(x;\theta)$,
  a pretrained model $\pi_{\mathrm{pt}}(\cdot|x) = \mathrm{softmax}(f(x;\theta_{\mathrm{pt}}))$.
Initialize $\theta$.
**for** $i = 1, \ldots, m$ **do**
  $\mathbf{v}_\theta \leftarrow \log \mathrm{softmax}(f(x_i;\theta_{\mathrm{pt}})) + r(x_i;\theta)$.
  $\mathbf{p}_\theta \leftarrow \mathrm{softmax}(\mathbf{v}_\theta)$
  $\mathcal{L}_\theta \leftarrow \mathrm{CE}(\mathbf{p}_\theta, y_i)$
  Update $\theta$ with the gradient of $\mathcal{L}_\theta$.
**end for**
**return** $\theta$.

# 表現力の等価性

【疑問】

- PRT で学習したモデルってちゃんと学習できる？
- 通常の FT で学習したモデルと比べて能力に差がある？

【命題】

FT で学習したモデルに対して、精度面で 1 対 1 対応するモデルが存在する

$$\{\pi_{\mathrm{ft}}(y|x) : \textit{fine-tuned models}\}$$
$$\rightarrow \{r(y|x) : \textit{rewards satisfying } \mathbb{E}_{y\sim\pi_{\mathrm{pt}}(y|x)}[e^{r(x,y)}] = 1\}$$
$$\textit{where } \pi_{\mathrm{ft}}(y|x) \textit{ is mapped to the implicit reward } \log(\pi_{\mathrm{ft}}(y|x)/\pi_{\mathrm{pt}}(y|x)).$$

# PRTの推論

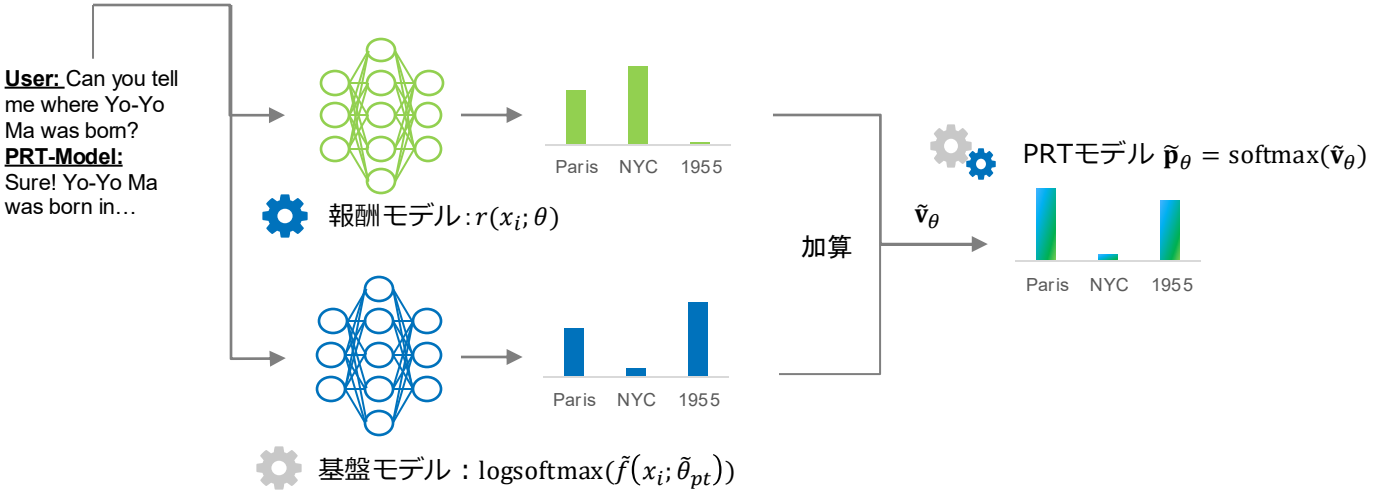
## ポータブルモデルを使った推論

$$\pi_\theta(y|x) = \frac{1}{Z_\theta(x)} \pi_{\mathrm{pt}}(y|x) \exp\left(\frac{1}{\lambda} r_\theta(x,y)\right)$$

$$\tilde{\pi}_\theta(y|x) = \frac{1}{\tilde{Z}_\theta(x)} \tilde{\pi}_{\mathrm{pt}}(y|x) \exp\left(\frac{1}{\lambda} r_\theta(x,y)\right)$$

**Algorithm 2** Pseudocode for Inference of PRT

**Given:** an input $x$, the trained reward $r(x;\theta)$,
  a pretrained model $\widetilde{\pi}_{\mathrm{pt}}(y|x) = \mathrm{softmax}(\widetilde{f}(x;\widetilde{\theta}_{\mathrm{pt}}))$,
$\widetilde{\mathbf{v}} \leftarrow \log \mathrm{softmax}(\widetilde{f}(x;\widetilde{\theta}_{\mathrm{pt}})) + r(x;\theta)$.
$\widetilde{\mathbf{p}} \leftarrow \mathrm{softmax}(\widetilde{\mathbf{v}})$
**return** $\widetilde{\mathbf{p}}$ as the output probability conditioned by $x$.



# 基盤モデルの入れ替え

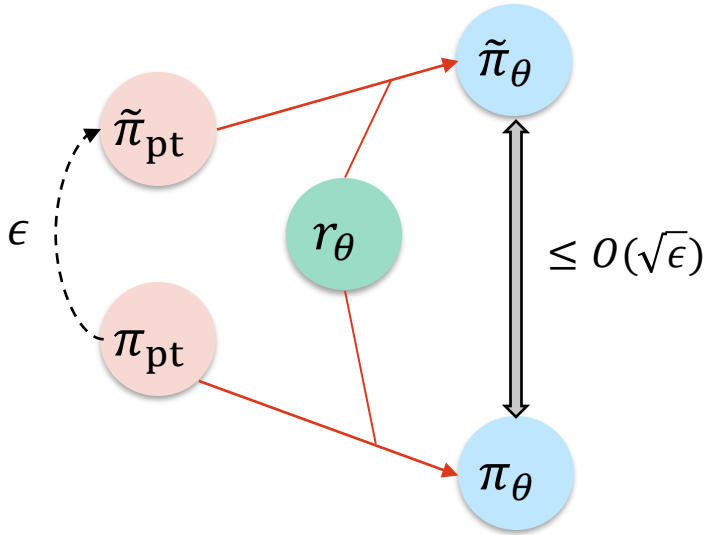
【疑問】

- 推論時に基盤モデルを入れ替えて大丈夫？
- どんな基盤モデルでも入れ替えできるの？

【命題】

**Proposition 3.2.** *Suppose that* $\widetilde{\pi}_{\mathrm{pt}}(y|x)$ *is close to* $\pi_{\mathrm{pt}}(y|x)$*, i.e.,* $D_{\mathrm{KL}}\left(\pi_{\mathrm{pt}}(\text{-}|x) \,\|\, \widetilde{\pi}_{\mathrm{pt}}(\text{-}|x)\right) \leq \varepsilon$*. Additionally, we assume that the maximum and mean value ratio of the exponential reward, i.e.,* $\max_y \exp r_\theta(x,y) / \mathbb{E}_y \exp r_\theta(x,y)$*, is bounded by some constant* $C$*. Then, the PRT models* $\pi_\theta(y|x)$ *and* $\widetilde{\pi}_\theta(y|x)$ *are also close as distributions:*

$$D_{\mathrm{KL}}\left(\pi_\theta(y|x) \,\|\, \widetilde{\pi}_\theta(y|x)\right) \leq O(\sqrt{\varepsilon}).$$



# 実験

# 実験設定：画像分類

## 実験の流れ

【訓練フェーズ】

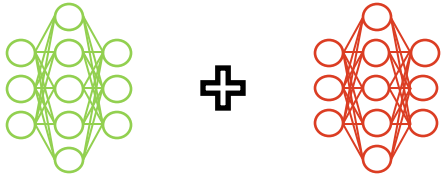
報酬モデルのパラメータ初期値 = 基盤モデルのパラメータ

基盤モデル:
ViT-B-16 train on
LAION-400M

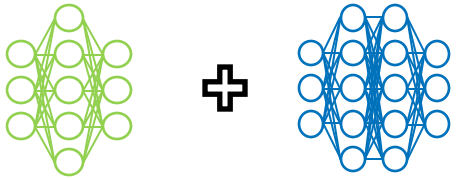

【推論フェーズ】

学習済み報酬モデル　　Target model

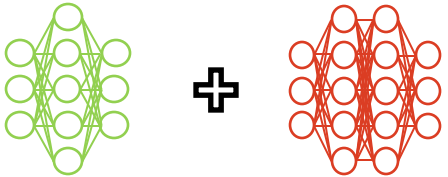
パターン A) 学習時と同じアーキテクチャ、異なるデータで学習したモデル




パターン B) 学習時と同じデータ、異なるアーキテクチャで学習したモデル



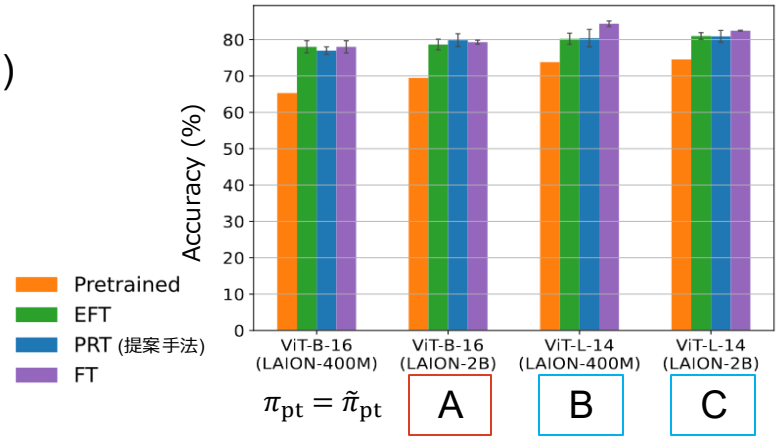

パターン C) 学習時と異なるアーキテクチャ、データで学習したモデル



# 画像分類タスクでPRTは効果があるか？

- 学習時の基盤モデル $\pi_{\mathrm{pt}}$：ViT-B16 (train on LAION-400M)
- 推論時の基盤モデル $\tilde{\pi}_{\mathrm{pt}}$ ：
  - A) 学習時と同じアーキテクチャ、異なるデータで学習したモデル
  - B) 学習時と同じデータ、異なるアーキテクチャで学習したモデル
  - C) 学習時と異なるアーキテクチャ、データで学習したモデル

CUB データセット：鳥の画像分類




【結論】画像分類タスクで PRT は SFT に近しい性能を達成した

- 基盤モデルを入れ替えず（ $\pi_{\mathrm{pt}} = \tilde{\pi}_{\mathrm{pt}}$ ）に推論した場合は SFT、EFTと同等の性能
- A) の学習データが異なる基盤モデルへの入れ替えの場合も SFT、EFTと同等の性能
- B) 、C) のアーキテクチャが異なる基盤モデルの入れ替えの場合は SFT に僅かに及ばずだが、基盤モデル単体と比較してタスク性能は向上

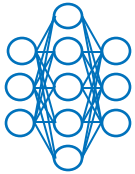
# 実験設定：言語生成

## 実験の流れ

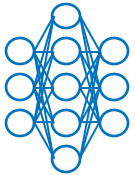
【訓練フェーズ】

報酬モデルのパラメータ初期値：基盤モデルのパラメータ

基盤モデル
Llama3.2-1B
or
Qwen2-0.5B
or
Qwen2.5-0.5B



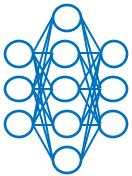
学習対象



パラメータ固定

Instruction Dataset で 学習

報酬モデル：$r(x_i;\theta)$



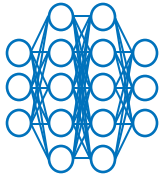
基盤モデル：$\pi_{\mathrm{pt}}$

【推論フェーズ】

学習済み報酬モデル　　Target model

パターン A) 学習時の基盤モデルと同じバージョン、大きいモデル

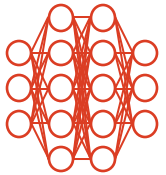
例）
Llama3.2-1B
or
Qwen2.5-0.5B



例）
Llama3.2-3B
or
Qwen2.5-1.5B
Qwen2.5-3B
⋮

パターン B) 学習時の基盤モデルより古いバージョン、大きいモデル

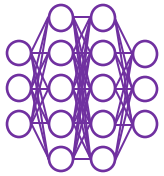
例）
Llama3.2-1B



例）
Llama3-8B
Llama3.1-8B

パターン C) 学習時の基盤モデルより新しいバージョン

例）
Qwen2-0.5B



例）
Qwen2.5-0.5B
Llama2.5-1.5BB

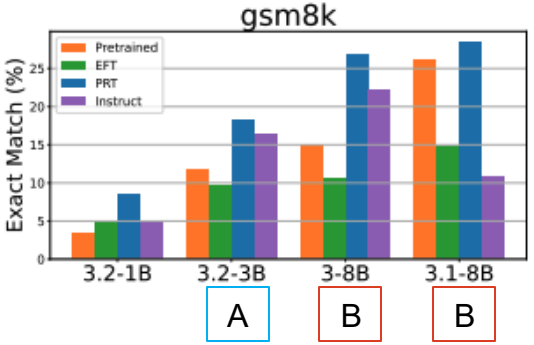
# 言語生成タスクで PRT は効果的か？

- 学習時の基盤モデル $\pi_{\mathrm{pt}}$ ：Llama3.2-1B、Qwen2.5-0.5B
- 推論時の基盤モデル $\tilde{\pi}_{\mathrm{pt}}$ ：
  - A) 学習時の基盤モデルと同じバージョン、大きいモデル
  - B) 学習時の基盤モデルより古いバージョン、大きいモデル

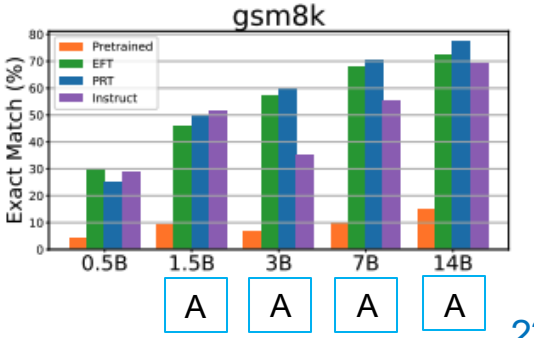
【結論】言語生成タスクで PRT は Instruction tuning モデルに近しい性能を達成した

- 基盤モデル単体と比較してPRT のタスク性能の方が高い
  → PRTの学習結果を活用できている
- PRTでは、サイズがより大きな基盤モデルの方がでより良いスコアを達成　→　基盤モデルの能力を活用できた
- バージョンが古いモデルでも有効性が確認できた

GSM8kデータセット: 算数タスク




Llama3 シリーズ




Qwen2.5 シリーズ

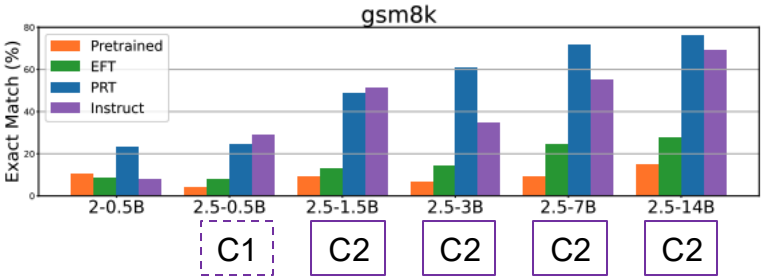
# 基盤モデルのアップデートに対応可能か？

- 学習時の基盤モデル $\pi_{\mathrm{pt}}$ ：Qwen2-0.5B
- 推論時の基盤モデル $\tilde{\pi}_{\mathrm{pt}}$ ：
  - C1) 学習時の基盤モデルより新しいバージョン、サイズは同じ
  - C2) 学習時の基盤モデルより新しいバージョン、大きいモデル

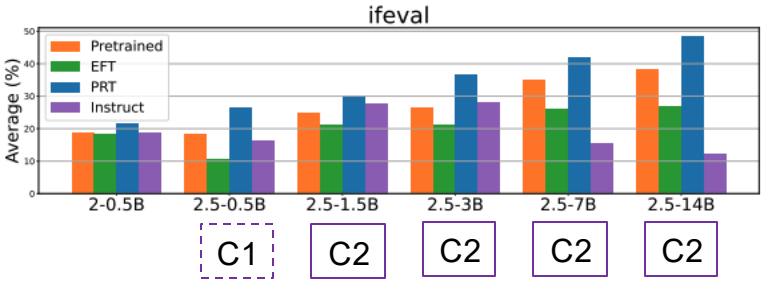
【結論】基盤モデルのアップデートのシナリオにおいても、PRT は有効性を示した

- 基盤モデル単体と比較して、 PRT のタスク性能の方が高い
- 指示追従タスクでも、基盤モデル単体と比較してタスク性能は向上

GSM8kデータセット：算数タスク




ifeval データセット：指示追従タスク



# 出力はどのようになっているか？

【調査方法】

“R=27. James has” に続く

- 基盤モデル $\tilde{\pi}_{\mathrm{pt}}$ ← PRT による分布修正前
- PRTモデル $\tilde{\pi}_{\theta}$ ← PRT による分布修正後

のトークン予測確率を調べる

【モデル】
- 学習時の基盤モデル $\pi_{\mathrm{pt}}$ : Llama3.2-1B
- 推論時の基盤モデル $\tilde{\pi}_{\mathrm{pt}}$ : Llama3-8B

【結論】

ベースの基盤モデルは 33 が最も高い確率
PRTモデル $\pi_{\theta}(x, y)$ は 6 が最も高い確率
→ PRT モデルがstep by step で考えるように分布を修正している

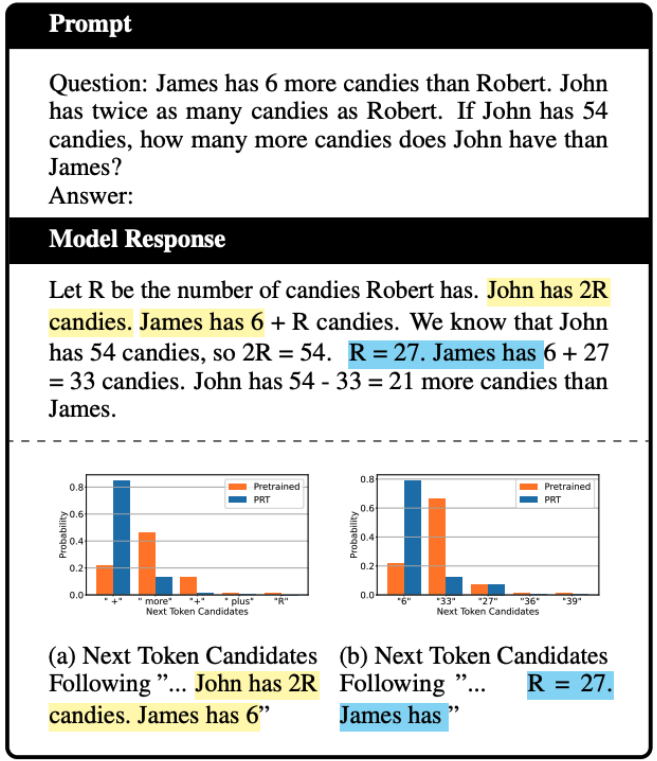



Figure 5: Changes in next-token probabilities by PRT.

# 提案手法の計算コストは？

【学習時の計算コスト】

基盤モデルは凍結：学習対象のパラメータ数はFTと同じ

- GPUメモリ使用量：FTと比べPRTは約 1.02倍
- 処理速度：FTと比べPRTは約 1.2～1.3倍

  → 学習時のオーバーヘッドはそこまで大きくない

【推論時の計算コスト】

推論時に計算するモデルの数：FT: 1つ, EFT: 3つ, PRT: 2つ

視覚モデルで

- FTと比較するとGPUメモリは約1.1 ~ 1.5倍増加、推論時間は約1.5 ~ 2倍増加した。
- EFTと比較するとGPUメモリは 7~18% 程度削減、推論時間は24 ~32% 程度削減した

言語モデルでも

- EFTと比較すると推論時間は 7% ~30%程度削減に成功した

学習時のコスト（視覚モデル）

| Models | | FT | PRT |
|---|---|---|---|
| ResNet-50 | Peak GPU memory | 12.84 GB | 13.08 GB |
| | Average time per batch | $38.26_{\pm 3.62}$ ms | $46.36_{\pm 3.34}$ ms |
| ViT-B-16 | Peak GPU memory | 20.17 GB | 20.58 GB |
| | Average time per batch | $116.10_{\pm 0.43}$ ms | $151.13_{\pm 0.48}$ ms |

推論時のコスト（視覚モデル）

| Source Models | Target Models | | Target FT (Oracle) | EFT | PRT |
|---|---|---|---|---|---|
| ResNet-50 | ViT-B-16 | Peak GPU memory | 1.46 GB | 2.42 GB | 2.18 GB |
| | | Average time per batch | $3.52 \pm 0.17$ ms | $10.27 \pm 0.22$ ms | $7.00 \pm 0.12$ ms |
| ResNet-50 | ViT-L-14 | Peak GPU memory | 2.96 GB | 3.44 GB | 3.20 GB |
| | | Average time per batch | $5.85 \pm 0.11$ ms | $12.52 \pm 0.21$ ms | $9.43 \pm 0.19$ ms |
| ViT-B-16 (LAION-400M) | ViT-B-16 (LAION-2B) | Peak GPU memory | 1.46 GB | 2.29 GB | 1.88 GB |
| | | Average time per batch | $3.52 \pm 0.17$ ms | $9.18 \pm 0.12$ ms | $6.33 \pm 0.07$ ms |
| ViT-B-16 (LAION-2B) | ViT-L-14 (LAION-2B) | Peak GPU memory | 2.96 GB | 3.79 GB | 3.38 GB |
| | | Average time per batch | $5.85 \pm 0.11$ ms | $11.73 \pm 0.15$ ms | $8.89 \pm 0.11$ ms |

推論時のコスト（言語モデル）

| Source Models / Target Models | Average Time per Token (ms) | | |
|---|---|---|---|
| | **Llama2-7B** **Llama2-13B** | **Llama3.2-1B** | **Llama3.2-3B** **Llama3-8B** |
| Pretrained | $26.0_{\pm 0.2}$ | $17.1_{\pm 1.1}$ | |
| EFT | $39.8_{\pm 0.1}$ (× 0.65) | $24.4_{\pm 0.2}$(×0.7) | $30.4_{\pm 0.0}$(×0.56) |
| PRT | $27.8_{\pm 0.5}$ (× 0.93) | $22.7_{\pm 1.8}$(×0.75) | $23.1_{\pm 1.0}$(×0.74) |

# 発展～汎化性能とエントロピー正則化～

お気持ち：様々な基盤モデルで使い回し可能になるよう汎化性能を上げたい！

## PAC-Bayes的視点からの解析

- $\mathcal{P}$：事前分布（pretrained modelの分布）
- $\mathcal{Q}_\theta$：報酬モデルに基づくPRTモデルの事後分布

【命題】
データ分布 $D$ に従ってサンプル $S = \{x_1, \dots, x_m\} \sim D_m$ を取得したとき、任意の $\delta \in (0,1)$ に対して、少なくとも $1-\delta$ の確率で成立：

$$\mathbb{E}_{\pi\sim\mathcal{Q}_\theta}\mathbb{E}_{x\sim\mathcal{D}}\left[l(x,\pi)\right] \le \mathbb{E}_{\pi\sim\mathcal{Q}_\theta}\left[\frac{1}{m}\sum_i^m l(x_i,\pi)\right] + \sqrt{\frac{D_{\mathrm{KL}}(\mathcal{Q}_\theta \,\|\, \mathcal{P}) + \log(\frac{1}{\delta}) + \frac{5}{2}\log m + 8}{2m-1}},$$

$D_{KL}(\mathcal{Q}_\theta \,\|\, \mathcal{P})$の項に着目
→ PRTモデル$\tilde{\pi}_\theta(y|x)$ が事前学習モデル $\tilde{\pi}_{\mathrm{pt}}(y|x)$ に近いほど汎化性能が高い

【提案手法の拡張】
報酬分布のエントロピーを正則化項として導入
- メリット：汎化性能をさらに高めることが可能
- デメリット：学習の最適性には悪影響を与える可能性

$$\widetilde{\mathcal{L}}(\theta) := \mathcal{L}(\theta) - \alpha\frac{1}{|S|}\sum_{(x,y)\in S} H(\rho_\theta(\text{-}|x)),$$

- $L(\theta)$：元の損失（クロスエントロピー）
- $H(\rho_\theta(\cdot \mid x)) := -\sum_{y\in Y}\rho_\theta(y \mid x)\log\rho_\theta(y \mid x)$：報酬分布のエントロピー
- $\alpha \in R_{\ge 0}$：正則化の強さを制御するハイパーパラメータ

# Entropy正則化の効果はあるか？

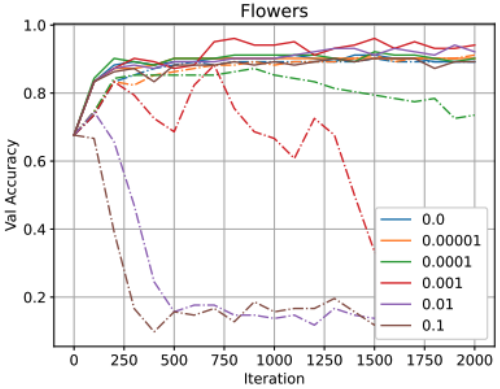
【結論】

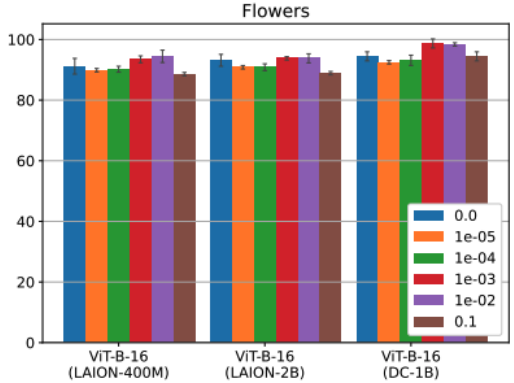
- 正則化係数が大きすぎると学習中に reward model 自体の精度が低くなる
- 一方で、PRTの最終的な性能がそれに伴って悪化するわけではない

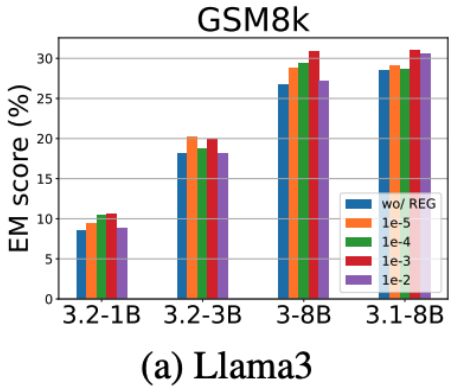
  → 報酬の精度に関わらずPRT の性能が保たれる・改善される＝ただのアンサンブルではない

【直感的解釈】

- 正則化により、報酬モデルの出力分布が一様分布に近づく（＝クセが少なくなる）
  → 元のモデルの持ち味を活かしつつ、必要な分だけ補正をかけることができるようになる



# おわりに

# おわりに

研究のスタートはモデルマージだった

- 特定のタスクに対する能力を獲得済みのモデル同士を組み合わせて、様々なタスクに使えるモデルを作りたい

  → 学習過程転移を使って実現できないか？

課題：アーキテクチャを揃える必要あり

  → 異なる基盤モデルだとパーミュテーションも考慮する必要あり

結論！！出力分布のマージの方が柔軟性が高く、やりやすい！