Drag-and-Drop LLMs による LoRA アダプター生成器の構築と日本語タスクへの適応

源田祥子 佐藤諒 中島大 伊藤真也 中村聡史
株式会社リコー
{shoko.genda, ryo.sato4, dai.nakashima,
shinya.itoh, satoshi.ns.nakamura}@jp.ricoh.com

概要

Drag-and-Drop LLMs（DnD）[1]は学習を行わず，プロンプトのみで LoRA [2]アダプターを生成し，タスク適応する手法である．本研究では，DnD を日本語タスクに適用し，その有効性を検証した．複数の日本語データセットを用いて LoRA アダプターを収集し，汎用性能を備えたアダプター生成器を構築した後，数学等の 4 つのタスクに対して適用した．JapaneseMT-Bench[3]を用いて汎化性能を維持しつつ各タスクの性能を比較した．その結果，DnD は目的タスクと類似タスクの性能も改善できる LoRA アダプターを生成できる可能性が示唆された．以上より，DnD は今後の LLM 開発において，低コストで柔軟にタスク適応を実現できることが期待できる．

1 はじめに

大規模言語モデル（LLM: Large Language Model）に特定のタスクやドメインに適応させるための学習手法が数多く提案されている．代表的な手法の一つとして LoRA が挙げられる．LoRA は，事前学習済みモデルの重みを固定したまま，一部の低ランク行列のみを学習することで効率的に適応する手法である．しかし，LoRA を含むパラメータ効率の高い学習手法であっても，学習データの準備や計算資源の確保といった点において依然として高いコストが発生する．このような課題に対し，スパースモデル等を介してアダプターを生成しタスク適用する手法が提案されている[1,4,5]. これらの手法は，モデル全体の再学習を行うことなく，軽量な処理を通じて振る舞いを制御する点に特徴がある．

別のアプローチとして学習を伴わず，LLM の振る舞いを柔軟に制御する方法として DnD が挙げられる．DnD は，ファインチューニングを行わずにプロンプト入力のみで LoRA アダプターを生成し，タスク適応を実現する手法である．プロンプトエンジニアリングがテキストレベルでモデルの挙動を調整するのに対し，DnD は LoRA アダプターを生成することでモデルを重みレベルで直接制御できる．DnD では，あらかじめプロンプトと対応する LoRA アダプターを用いてアダプター生成器を用意し，目的タスクのプロンプトを入力するだけで，即座にタスク適応した LoRA アダプターを取得できる．このように DnD は従来の LoRA ファインチューニングと比較して，計算資源を大幅に削減しつつ，プロンプト入力のみでモデルの重みを制御する手法である. しかし，既存の DnD に関する報告は主に英語タスクを対象としており，日本語タスクにおいては十分な調査が行われていないように見受けられる．さらに，従来の DnD は特定タスクごとに個別の生成器を構築する先行例が多く，汎用的アダプター生成器の有効性についてもまだ十分に検証されていない.

そこで本研究では，汎用的なデータを用いてアダプター生成器を学習し，特定タスクのプロンプトを入力として与えることで，汎化性能を維持したまま特定タスクに特化した LoRA アダプターが生成可能であるかを検証することを目的とする. 具体的には，複数の日本語データセットを用いて LoRA アダプター生成器を学習し，従来の LoRA ファインチューニングと比較することで，DnD がどの程度計算資源やコストを削減しつつ，汎化性能を維持しながら特定タスクの性能向上に寄与できるかを評価する．これにより，一度汎用的なアダプター生成器を構築しておけば，追加の学習を行うことなく，さまざまな日本語タスクに迅速な適応が可能となる．この特性から，DnD は今後の LLM 開発におけるタスク適応の効率を飛躍的に向上させる手法として期待できる．

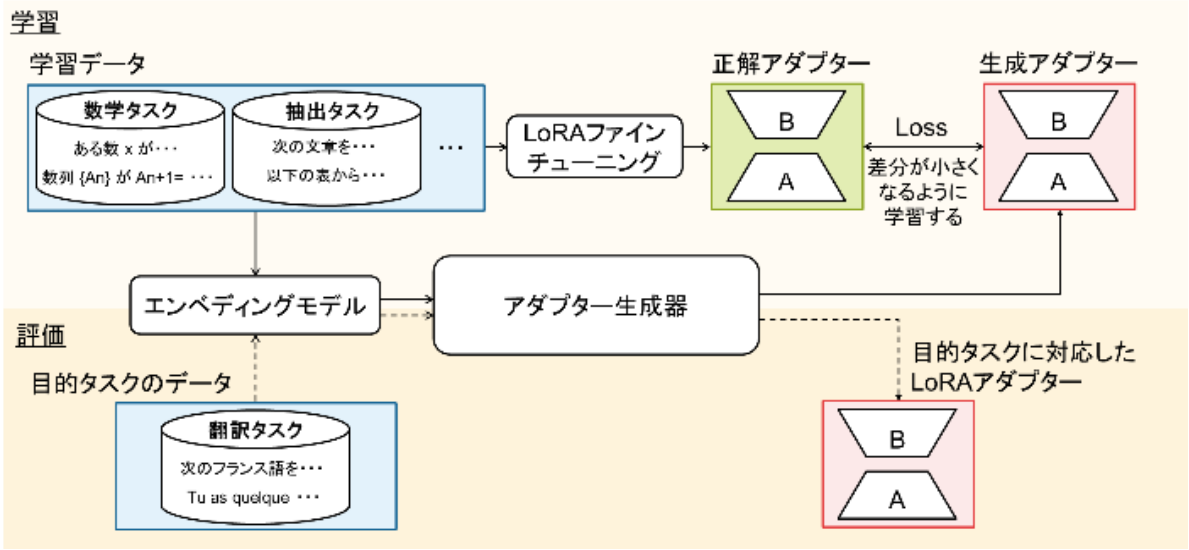

図 1　DnD の概要

2 方法

本研究では，まず複数の学習データを用いて LLM に LoRA ファインチューニングを行い，学習データに対応した LoRA アダプターを収集した．アダプター生成器の学習では，学習データを埋め込みモデルでベクトル化し，LoRA アダプターの重みに変換する生成器(アダプター生成器)を構築した．この生成器は生成するアダプターと事前に収集したアダプター(正解アダプター)との差分が小さくなるように学習される．評価試行では，目的タスクのプロンプトを用意し，エンベディングモデルと生成器に入力することで，学習過程を省いてタスクに適応した LoRA アダプターを入手することができる(図 1).

2.1 データセット

アダプター生成器の学習時におけるデータを示す(表 1). これらのデータを用いて正解アダプターを作り，汎用的なアダプター生成器を作成した．

表 1　アダプター生成器作成データセット

データ名	レコード数
GENIAC-Team-Ozaki/Hachi-Alpaca_newans[6]	27.8 k
GENIAC-Team-Ozaki/chatbot-arena-ja-karakuri-lm-8x7b-chat-v0.1-awq[7]	12.5 k
Aratako/SFT-Dataset-For-Self-Taught-Evaluators-iter1[8]	15.6 k
独自データ	15.1 k

2.2 LoRA ファインチューニングによる正解アダプターの収集

正解アダプターは，tokyotech-llm/Llama-3.1-Swallow-8B-v0.5[9]に対して SFTTrainer[10]の LoRA ファインチューニングを実施することで収集した（A.1）．汎用的なアダプター生成器の構築を目的とし，高い表現力を有するアダプターが必要であることから，本研究では LoRA ファインチューニングにおける rank を先行研究[1]と比較して大きな値に設定した. 高 rank 設定に伴う学習の不安定化を抑制するため, rsLoRA[11]に基づくスケーリングを導入し，rank および alpha をそれぞれ 256，512 とした．学習過程における学習曲線を観察し，Loss が低下し，収束したと判断される 450 step 以降に得られたアダプターを正解アダプターとして採用した．

2.3 アダプター生成器

アダプター生成器はプロンプトから埋め込み表現を入力として受け取り，LoRA アダプターの重みを直接出力するニューラルネットワークである．

2.3.1 アダプター生成器への入力

アダプター生成器の入力は，複数のプロンプトをまとめたプロンプトバッチである. 各プロンプトは，埋め込みモデルによってベクトルへ変換される．埋め込みモデルは sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2[12]を使用した．入力は以下の 4 次元テンソルとして表現される．

$$[B, N, L, C]$$

B: バッチサイズ
N: 1 バッチに含まれるプロンプト数
L: 系列長
C: 埋め込みモデルの隠れ次元

2.3.2 アダプター生成器の構造

アダプター生成器は先行研究と同一の hyper-convolutional decoder を用いた．本モデルは複数の hyper-convolutional layer を積み重ねた構成を持つ．hyper-convolutional layer は，2 つの ConvW，2 つの ConvH，および 1 つの ConvL から構成される．各 hyper-convolutional layer は，(C, L) 次元に作用する width convolution（ConvW），(L, N) 次元に作用する height convolution（ConvH），および (N, L) 次元に作用する layer-wise convolution（ConvL）である．ConvW および ConvH は 2 経路で適用され，それぞれ ConvW→ConvH，ConvH→ConvW の順で処理される．各畳み込みでは，対象とする次元が特徴次元となるように入力テンソルを転置し，残りの次元をチャネル次元として扱う各層の出力は，2 経路の出力と学習可能なバイアス項を平均した後，ConvL により統合される．なお，各層の入力には正規化を施し，一部の畳み込み後に非線形変換を適用する．ここで，c^lはl層目の出力特徴量，ここで，c^0はプロンプト埋め込み，bはバイアス項である．

$$c_l^W = Conv_H^1\left(Conv_W^1(c^{l-1})\right)$$
$$c_l^H = Conv_W^2\left(Conv_H^2(c^{l-1})\right)$$
$$c^l = Conv_L\left(\frac{c_l^W + c_l^H + b}{3}\right)$$

アダプター生成器は入力行列[N, L, C]が段階的に縮約・変形され，最終的に LoRA 重みに対応する形状へ変換される(表 2).

表 2　アダプター生成器の設定

パラメータ	値
model_config	[(8, 8192, 384), (16, 2048, 384), (64, 1024, 384), (256, 512, 384), (512, 512, 512), (2048, 256, 512), (5120, 256, 512),]
kernel_size	5

2.4 アダプター生成器の学習

学習時では，プロンプト埋め込みを入力としてアダプター生成器に与え，対応する LoRA アダプター重みを出力させた．正解アダプターの数は 2.2 で収集した 79 個の LoRA アダプターとした．

損失関数には，生成された LoRA アダプターと正解アダプター重みとの間の平均二乗誤差（Mean Squared Error: MSE）を用いた．この損失は，全ての重みに対して計算され，生成器が正解アダプターの重み分布を再現するように最適化された．

表 3　アダプター生成器学習時の設定

パラメータ	値
batch_size	8
total_steps	10000
learning_rate	3.0e-5
num_adapter	79

2.5 評価プロンプト

本研究では JMT-Bench に含まれているタスクであるコーディング, 数学, 抽出, STEM に対して DnD を適応した(表 4).

表 4　評価プロンプト

タスク	データセット
コーディング	kogi-jwu/jhumaneval[13]
抽出	sbintuitions/JSQuAD[14]
数学	elyza/JaMARD[15]
STEM	LLaMAX/BenchMAX_Science[16]

3 結果・考察

3.1 学習コスト

本研究におけるメモリ使用量やアダプター生成までに要する時間を比較した(表 5)．アダプターの生成は NVIDIA DGX H100（80GB）GPU 1 基を用いて行った．DnD ではタスク適応に必要な時間が改善されたことが確認できた．

表 5　GPU メモリ使用量

		メモリ使用量	稼働時間
LoRA		約 50 GiB	11 h 04 m
DnD	数学	約 38 GiB	0.90 sec
	コーディング		1.00 sec
	STEM		0.30 sec
	抽出		0.25 sec

3.2 ベンチマークスコア

2.2 で LoRA ファインチューニングを行ったモデルと比較し, 目的タスクであるコーディング, 数学, 抽出, STEM のプロンプトに適応した LoRA アダプターの有効性を検証するため, JMT-Bench を用いて評価を行った(表 7). 本評価では, GPT-4o を用いた LLM as a Judge により 10 点満点で採点した. さらに, LoRA ファインチューニングモデルのスコアを基準とし, DnD によってどの程度の性能を引き出せたかを相対値として算出した. 表では列と行で同一のタスクに対応するセルを灰色で示し, LoRA ファインチューニングモデルに対して性能向上が確認された箇所には太字とした.

平均スコアを見ると, いずれのタスクにおいても DnD は LoRA ファインチューニングモデルと同程度の性能を示すことが確認された. さらに, タスク別の詳細な結果に着目すると, DnD の効果はタスクごとに異なる傾向を示している.

数学タスクでは, 自タスクで明確な性能向上が見られたほか, 推論および執筆タスクにおいて一定の改善が確認された. コーディングタスクでは, 自タスクに加えて数学および執筆タスクにおいても性能向上が確認された. これらのタスクはいずれも構造的な出力生成や論理的推論を必要とする点で共通しており, プロンプトによるアダプター生成器への条件付けにより, 関連するタスク能力を選択的に強化した可能性が示唆された. また, STEM においても, 自タスクに加えてコーディング, 人文, 執筆といったタスクにおいて性能向上が確認された. これらは幅広い知識の統合や文脈に応じた表現生成を必要とする点で共通しており, 汎用的な知識活用能力が DnD により強化された可能性が示唆された.

以上の結果から, DnD は条件として与えたタスクと親和性の高い能力を選択的に強化できることが示唆された. 汎用的な能力を保持したアダプターを基盤としつつ, タスクに応じて特定の能力を柔軟に強調するという, 日本語タスクにおける効率的なタスク適応手法としての有用性を示すものである.

表 7　JMT-Bench の内訳

	LoRA	DnD			
		数学	コーディング	STEM	抽出
平均	6.19	**6.18**	**6.23**	5.94	6.02
		(1.00)	**(1.01)**	(0.96)	(0.97)
数学	4.90	**5.45**	**5.30**	3.95	**5.35**
		(1.11)	**(1.08)**	(0.81)	**(1.09)**
コーディング	4.40	**5.15**	**5.35**	**5.31**	4.20
		(1.17)	**(1.22)**	**(1.21)**	(0.95)
STEM	6.80	6.30	6.60	**6.80**	**6.90**
		(0.93)	(0.97)	**(1.00)**	**(1.01)**
抽出	7.30	6.35	6.70	5.10	7.05
		(0.87)	(0.92)	(0.70)	(0.97)
推論	4.25	**4.35**	3.90	3.70	3.20
		(1.02)	(0.92)	(0.87)	(0.75)
人文	8.25	**8.30**	8.15	**8.45**	**8.45**
		(1.01)	(0.99)	**(1.02)**	**(1.02)**
ロールプレイ	6.85	6.35	6.35	6.55	5.85
		(0.93)	(0.93)	(0.96)	(0.85)
執筆	6.75	**7.20**	**7.45**	**7.65**	**7.15**
		(1.07)	**(1.10)**	**(1.13)**	**(1.06)**

4 おわりに

本研究では, 効率的なタスク適応手法として DnD に着目し, 汎用的なデータを用いて学習したアダプター生成器を構築した上で, 追加学習を行うことなく, 複数の日本語タスクに適応した LoRA アダプターを生成できるかを検証した. その結果, ベンチマークの平均スコアにおいては従来の LoRA ファインチューニングと同程度の性能を維持しつつ, 目的タスクを含む類似したタスクにおいて, DnD により生成された LoRA アダプターが同等あるいはそれを上回る性能を示すことが確認された. したがって, DnD は条件として与えたタスクと親和性の高い能力を選択的に強化できることが示唆された.

今後は, プロンプト設計の工夫やアダプター生成器の構造および学習手法の改良を通じて, DnD の汎化性能およびタスク適応能力をさらに高めることを目指す. これにより, 生成されるアダプターの汎化性能の向上や, より広範な日本語タスクに対する安定した学習不要なタスク適応の実現が期待される.

参考文献

[1] Liang, Zhiyuan, et al. "Drag-and-Drop LLMs: Zero-Shot Prompt-to-Weights." *arXiv preprint arXiv:2506.16406* (2025)

[2] Hu, Edward J., et al. "Lora: Low-rank adaptation of large language models." *ICLR* 1.2 (2022): 3.

[3] https://github.com/Stability-AI/FastChat

[4] Charakorn, Rujikorn, et al. "Text-to-LoRA: Instant Transformer Adaption." *arXiv preprint arXiv:2506.06105* (2025).

[5] Ortiz-Barajas, Jesus-German, Helena Gomez-Adorno, and Thamar Solorio. "Hyperloader: Integrating hypernetwork-based lora and adapter layers into multi-task transformers for sequence labelling." *arXiv preprint arXiv:2407.01411* (2024).

[6] https://huggingface.co/datasets/GENIAC-Team-Ozaki/Hachi-Alpaca_newans

[7] https://huggingface.co/datasets/GENIAC-Team-Ozaki/chatbot-arena-ja-karakuri-lm-8x7b-chat-v0.1-awq

[8] https://huggingface.co/datasets/Aratako/SFT-Dataset-For-Self-Taught-Evaluators-iter1

[9] https://huggingface.co/tokyotech-llm/Llama-3.1-Swallow-8B-v0.5

[10] https://huggingface.co/docs/trl/sft_trainer

[11] Kalajdzievski, Damjan. "A rank stabilization scaling factor for fine-tuning with lora." *arXiv preprint arXiv:2312.03732* (2023).

[12] https://huggingface.co/sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2

[13] https://huggingface.co/datasets/kogi-jwu/jhumaneval

[14] https://huggingface.co/datasets/sbintuitions/JSQuAD

[15] https://huggingface.co/datasets/elyza/JaMARD

[16] https://huggingface.co/datasets/LLaMAX/BenchMAX_Science

A.1 LoRA ファインチューニング時の設定

2.2 では，以下の設定に従って LoRA ファインチューニングを実施した．

パラメータ	値
max_seq_length	8192
lr_scheduler_type	cosine
learning_rate	5.0e-6
rank	256
target_modules	["q_proj", "k_proj", "v_proj", "o_proj", "gate_proj", "up_proj", "down_proj"]
lora_alpha	512
lora_dropout	0.05