物体との相互作用を考慮した把持画像生成 ～FLUX.2におけるLoRA実装と単一GPU学習手法の検証～

田中 基貴[*1]　堀江 雅也[*1]　井上 ゆり[*1]　金 東南[*1]　金 赫珍[*1]
高山 健志[*1]　Swanson James[*1]　楊 興超[*1]　山森 徹[*1]　武富 貴史[*1]

Abstract – **With the widespread adoption of open image generation models such as Stable Diffusion and FLUX.1, fine-tuning approaches using custom datasets have garnered significant attention. In this report, we implement and verify the architecture and training process of Low-Rank Adaptation (LoRA) for FLUX.2 [dev], a large-scale model with 32 billion parameters released in November 2025. To enable training on a single NVIDIA A100 80GB GPU, we constructed a memory optimization method combining CPU-Offload via PyTorch FSDP2 and Gradient Checkpointing. We validated our implementation through an overfitting experiment using a single image and investigated training trends using a high-quality dataset derived from HICO-DET. The results demonstrated improvements in contrast and detail during the early stages of training, whereas extended training tended to cause structural collapse in features such as hands and faces.**

Keywords : **FLUX.2, LoRA, Image-Generation, Single-GPU Training, Fine-tuning**

1　はじめに

画像生成技術は，拡張現実感や人工現実感のためのシーン生成，広告コンテンツの制作，ゲームやアニメなどのエンターテインメントコンテンツの制作など幅広い分野での利用がなされている．特に，Stable Diffusion [1] や FLUX.1 [2] をはじめとするオープンな画像生成モデルの公開により，高精細な画像生成が手軽に行えるようになった．

広告生成やエンターテインメント分野においては，公開モデルを追加学習することで独自のコンテンツを生成するアプローチが注目されており，これらの実践においては musubi-tuner [3] や DiffSynth-Studio [4] といったオープンソースのツールを利用する場合が多い．これらのツールは積極的な開発が行われているものの，高度な研究開発を行うためにはモデルアーキテクチャや学習プロセスにおける新しいアプローチをより柔軟に実装できることが重要である．

本稿では，特定の物体を把持した人物画像生成へ向けた初期の検証として，2025 年 11 月に公開された Flux.2[dev] [5] を対象に，LoRA [6] のアーキテクチャと学習プロセスを実装することにより追加学習実験を行った．また，1 基の NVIDIA A100 80GB GPU で学習するために必要なメモリコストの削減方法を導入し，HICO-DET [7] より物体把持画像を選別することによって作成したデータセットを用いて学習における傾向を調査した．

2　関連研究

2.1　FLUX.2

FLUX.2 [5] は Diffution Transformer [8] をアーキテクチャとして採用した Flow Matching [9] による大規模画像生成モデルの一種である．主に API として公開されている FLUX.2 [pro] や FLUX.2 [flex] と呼ばれる高性能モデルに加え，FLUX.2 [dev] と呼ばれる 32B のパラメータを持つモデルの学習済みパラメータと推論用ソースコードが公開されている．

FLUX.2[dev] は FLUX.1 Kontext [10] に似たテキストプロンプトと参照画像を統一したアーキテクチャにより，Text-to-Image のほか Image Editing などのタスクも統一的に解くことができる．

またテキストエンコーダとして Mistral-Small-3.2 [11]，画像エンコーダとして FLUX.2-VAE [12] を用いており，これらの重みも公開されている．

2.2　Low-Rank Adaptation

Low-Rank Adaptation(LoRA) [6] は大規模モデルの Fine-tuning において，過学習や破滅的忘却，甚大な計算コストといった問題を改善するために，学習可能なパラメータを削減するアプローチとして自然言語

[*1] CyberAgent

処理分野において提案された手法である．

LoRA は過剰なパラメータのモデルの多くは低い固有次元の部分空間に存在するという観察に基づいて，事前学習済みの重み行列 $W_0 \in R^{d\times k}$ を凍結し，その更新分 ΔW を低ランク行列の積 BA（$B \in R^{d\times r}, A \in R^{r\times k}$，ランク $r \ll \min(d,k)$）として近似する．すなわち，順伝播は $h = W_0x + BAx$ として定式化される．

画像生成分野において LoRA は，ControlNet [13] などでの適用以降，特定の画風やキャラクター，概念を追加学習するための手法として標準的に利用されるようになっている．

2.3 Zero Redundancy Optimizer

Zero Redundancy Optimizer(ZeRO) [14] は，データ並列学習を用いた大規模モデルの学習において GPU のメモリ制約に対処するために提案された手法である．

従来のデータ並列学習では各 GPU がモデル全体を複製して保持していたのに対し，ZeRO Stage3 ではオプティマイザ状態，勾配，およびモデルパラメータを GPU 間で分割して管理することでメモリ効率を改善している．一方で分割に伴い各 GPU が他の GPU からパラメータを収集するための通信オーバーヘッドが発生するため，ZeRO では通信を計算とオーバーラップさせる工夫が行われている．また，より厳しいメモリ制約に対処するために DRAM や NVMe ストレージを学習に組み込む CPU-Offload [15] も提案されている．

PyTorch FSDP (Fully Sharded Data Parallel) [16] などの分散学習フレームワークは，ZeRO の概念に基づき実装されており，限られた GPU リソースで大規模モデルを学習することが可能となる．

2.4 Gradient Checkpointing

Gradient Checkpointing [17] は，モデルの層が深くなるにつれて増大する，順伝播時の中間状態によるメモリ消費の問題に対処する手法である． 本手法では，順伝播時にすべての中間状態をメモリに保存する代わりに，特定のチェックポイントとなる層の出力のみを保存する．保存されなかった中間状態については，逆伝播時に直前のチェックポイントから順伝播を部分的に再計算することで復元し，必要な勾配を算出する．これにより計算コストは増加するものの，メモリ使用量を削減することが可能となる．

3 開発システムの概要

3.1 FLUX.2 LoRA

本研究では，FLUX.2[dev] に LoRA を適用するための追加のレイヤーを実装した．FLUX.2[dev] のアーキテクチャは主に Double Stream Block と Single Stream Block の組み合わせで構成されている．Double Stream Block では画像トークンとテキストトークンそれぞれに対応する別の重みをもち，トークンの連結に対して Attention を行うことで情報の統合が行われる．全体としては 8 層の Double Stream Block を通したのち画像トークンとテキストトークンが結合され，その後 48 層の Single Stream Block が適用される．

今回の実装では，Double Stream Block における画像トークンとテキストトークンそれぞれに対する self-attention および projection と，Single Stream Block における self-attention および projection に LoRA を適用するアーキテクチャ採用した．これは FLUX.1 における XLabs AI による LoRA のアーキテクチャ実装 [18] に近いものである．詳細は A 付録に示す．

3.2 学習におけるメモリ制約の改善

今回の実装においては，LoRA を適用した FLUX.2[dev] のモデルを 1 基の NVIDA A100 80GB で学習できるようにするため，PyTorch FSDP2 による CPU-Offload と Gradient Checkpointing を組み合わせた学習プロセスを実装した．Gradient Checkpointing は Double Stream Block と Single Stream Block のすべてのブロックの前に Checkpointing を設置した．またモデルおよび入力データの dtype は bfloat16 を基本とし，Automatic Mixed Precision(AMP) により必要に応じてより高精度な型で計算されるようにした．この設定により，1 基の NVIDA A100 80GB を用いてバッチサイズ 8，解像度 1024×1024 ピクセルの画像を用いた追加学習が可能となる．

本研究では，メモリ制約と計算速度がトレードオフとなる手法を積極的に採用し，一方でベースモデルの量子化などメモリ制約と精度がトレードオフとなるような手法は採用しないことを重視している．

4 検証

4.1 過学習実験

FLUX.2[dev] の LoRA 実装が正しく動作することを確かめるため，一枚の画像と対応するプロンプトを用いて，追加学習によりプロンプトからそのデータを再現できるようになるかを確認する過学習実験を行った．

実験では，Optimizer として AdamW を利用し，学習率 1e-4 でスケジューラを利用せず学習を行った．追加したパラメータは LoRA の rank を r とし，Down-projection 行列 W_{down} を正規分布 $\mathcal{N}(0, 1/r^2)$ で初期化した．一方，Up-projection 行列 W_{up} はゼロで初期化した．また r はすべての LoRA Layer で 16 を利用し, 画像は CenterCrop と 512×512 ピクセルへの Resize を適用した．

図 1　過学習実験の学習過程における生成画像の比較

図 2　HICO-DET 学習実験の学習過程における生成画像の比較

4.2　過学習実験 結果

過学習実験の結果を図 1 に示す．図は学習に利用した 1 枚の画像と，学習中に 200 ステップ毎にサンプリングされた出力結果を示している．

生成される画像は loss の減少とともに約 1000 ステップで学習データとよく似た画像が生成されるように収束しており，うまく学習が機能していることが分かる．

4.3　HICO-DET 学習実験

HICO-DET [7] は人間とオブジェクトのインタラクションを検出するために作成された画像データセットである．今回は検証において HICO-DET の中から人物と物体それぞれが明確に映っている画像を手作業で選別し，Gemini 2.5 pro [19] で画像に対応するプロンプトを生成する手法で約 3400 枚のオブジェクトを持った人物の高品質な画像テキストペアデータセットを作成した．またこのデータセットを用いて，FLUX.2 LoRA における学習規模が大きい場合の学習傾向を観察する検証を行った．学習は，CenterCrop を適用したのち 1024×1024 ピクセルにリサイズし，バッチサイズは 8 で学習を行った．

4.4　HICO-DET 学習実験 結果

過学習実験の結果を図 2 に示す．図では 1 ステップ，501 ステップ，2401 ステップの各モデルにおいて，学習に用いていない同一のプロンプトから生成した出力を示している．実験においては，1 ステップから 501 ステップにかけては高いコントラストや過度にぼやけた背景が自然な見た目に近づく傾向がみられたが，学習を続けた 2401 ステップでは手や顔などが構造的に破綻してしまう傾向が観察された．

5 まとめ

本研究では，FLUX.2[dev] に対して LoRA を適用するためのアーキテクチャを追加し，学習プロセスにおいて CPU-Offload と Gradient Checkpointing を適用することで単一の NVIDIA A100 80GB GPU 環境下においても実用的な学習が可能であることを実証した．検証においては，まず過学習実験によりシステムが正しく動作することを確認した．さらに HICO-DET をベースにオブジェクトを保持した人物の高品質なテキスト画像ペアデータセットを作成し，このデータセットを用いて FLUX.2[dev] の LoRA を学習した際の傾向を観察した．その結果，HICO-DE 学習実験では学習の初期段階で品質の改善が見られた一方で，長時間の学習は生成結果の破綻につながることが明らかとなった．

謝辞

本研究は株式会社サイバーエージェントにおけるデジタルヒューマンゼミの活動の一環として，実施されたものである．

A 付録

A.1 FLUX.2 LoRA アーキテクチャ詳細

実験に用いた FLUX.2 に適用した LoRA のアーキテクチャの詳細を図 3 に示す．緑のブロックで示したレイヤーが追加した LoRA レイヤーである．

参考文献

[1] Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-Resolution Image Synthesis with Latent Diffusion Models; Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 10684-10695 (2022.06)

[2] Black Forest Labs: FLUX; https://github.com/black-forest-labs/flux (2024)

[3] kohya-ss: musubi-tuner; https://github.com/kohya-ss/musubi-tuner (2025)

[4] modelscope: DiffSynth-Studio; https://github.com/modelscope/DiffSynth-Studio (2025)

[5] Black Forest Labs: FLUX.2; https://bfl.ai/models/flux-2 (2025)

[6] Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W.: LoRA: Low-Rank Adaptation of Large Language Models; International Conference on Learning Representations (ICLR), (2022.04)

[7] Chao, Y. W., Liu, Y., Liu, X., Zeng, H., Deng, J.: Learning to Detect Human-Object Interactions; Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV), 381-389 (2018.03)

[8] Peebles, W., Xie, S.: Scalable Diffusion Models with Transformers; Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 4195-4205 (2023.10)

[9] Lipman, Y., Chen, R. T. Q., Ben-Hamu, H., Nickel, M., Le, M.: Flow Matching for Generative Modeling; International Conference on Learning Representations (ICLR), (2023.05)

[10] Black Forest Labs, Batifol, S., Blattmann, A., Boesel, F., Consul, S., Diagne, C., Dockhorn, T., English, J., English, Z., Esser, P., Kulal, S., Lacey, K., Levi, Y., Li, C., Lorenz, D., Müller, J., Podell, D., Rombach, R., Saini, H., Sauer, A., Smith, L.: FLUX.1 Kontext: Flow Matching for In-Context Image Generation and Editing in Latent Space; arXiv preprint arXiv:2506.15742, (2025.06)

[11] Mistral AI: Mistral-Small-3.2-24B-Instruct-2506; https://huggingface.co/mistralai/Mistral-Small-3.2-24B-Instruct-2506, (2025.06)

[12] Black Forest Labs: FLUX.2-VAE; https://bfl.ai/research/representation-comparison (2025)

[13] Zhang, L., Rao, A., Agrawala, M.: Adding Conditional Control to Text-to-Image Diffusion Models; Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 3836-3847 (2023.10)

[14] Rajbhandari, S., Rasley, J., Ruwase, O., He, Y.: ZeRO: Memory Optimizations Toward Training Trillion Parameter Models; Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis (SC), (2020.11)

[15] Ren, J., Rajbhandari, S., Aminabadi, R. Y., Ruwase, O., Yang, S., Zhang, M., Li, D., He, Y.: ZeRO-Offload: Democratizing Billion-Scale Model Training; 2021 USENIX Annual Technical Conference (USENIX ATC 21), 551-564 (2021.07)

[16] Zhao, Y., Gu, A., Varma, R., Luo, L., Huang, C. C., Xu, M., Wright, L., Shojanazeri, H., Ott, M., Shleifer, S., Desmaison, A., Balioglu, C., Damania, P., Nguyen, B., Chauhan, G., Hao, Y., Mathews, A., Li, S.: PyTorch FSDP: Experiences on Scaling Fully Sharded Data Parallel; Proceedings of the VLDB Endowment, 16(12), 3848-3860 (2023.08)

[17] Chen, T., Xu, B., Zhang, C., Guestrin, C.: Training Deep Nets with Sublinear Memory Cost; arXiv preprint arXiv:1604.06174, (2016.04)

[18] XLabs-AI: x-flux; https://github.com/XLabs-AI/x-flux (2024)

[19] Google DeepMind: Gemini 2.5 Pro; https://deepmind.google/technologies/gemini/ (2025)

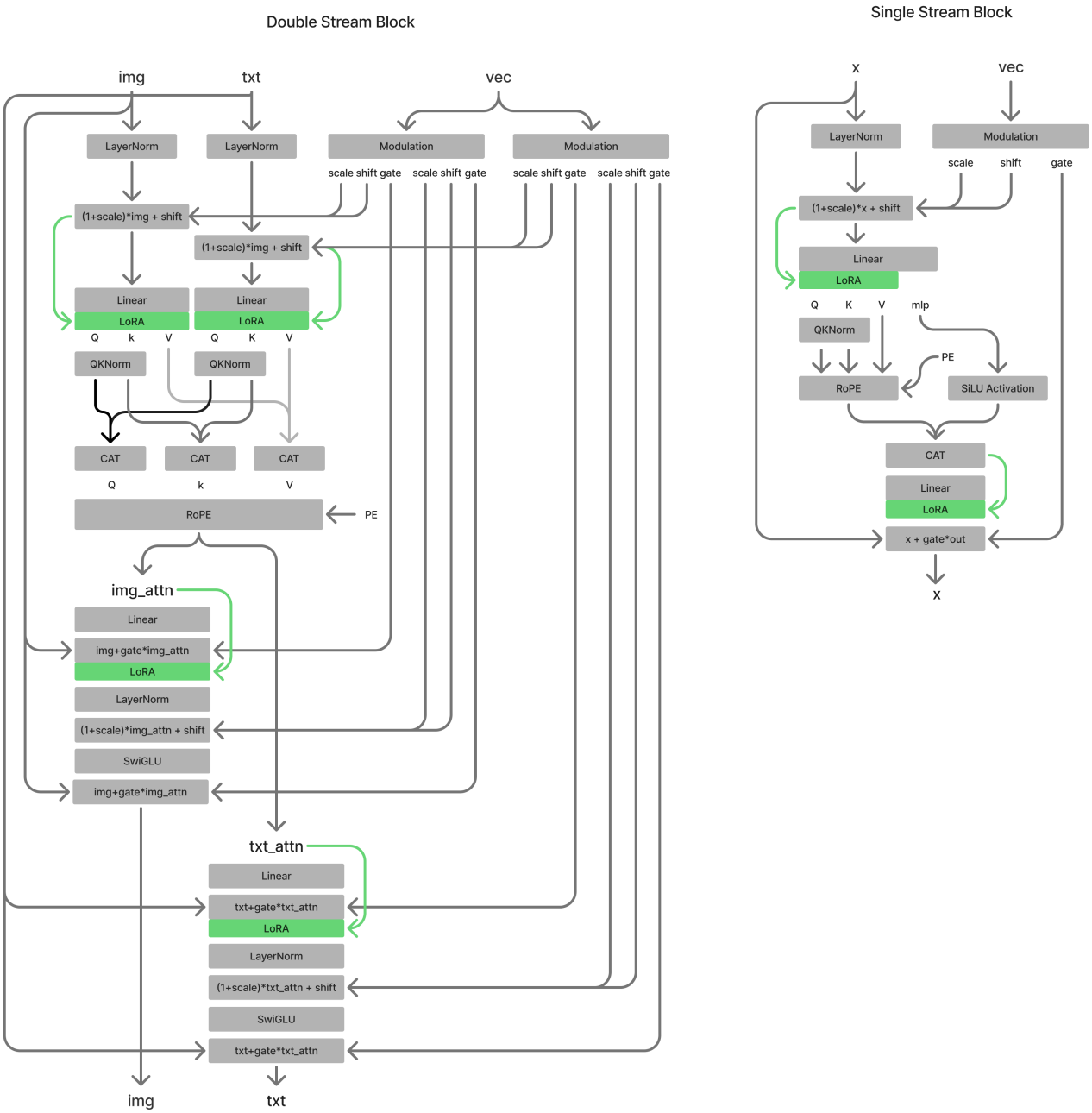

図 3　FLUX.2 LoRA アーキテクチャ