# LoRA（Low-Rank Adaptation）技術解説

©kokurenの自習室

# 課題: 大規模モデルの巨大化

- 近年のAIモデル: パラメータ数が爆発的に増加

- GPT-3は1750億、他の基盤モデルも数十億～何百億パラメータに達する。
- これらのモデルは強力だが、特定のタスクに合わせて微調整（ファインチューニング）するには課題がある。

# 課題: フルファインチューニングの壁

- 従来のファインチューニング手法の限界
- フルファインチューニング: 全パラメータを再学習するため、最高の性能が期待できる。
- しかし、課題も多い:
  - 計算コスト: 膨大なGPUメモリと計算時間が必要。
  - ストレージコスト: タスクごとに巨大なモデル（数十〜数百GB）を保存する必要があり非効率。
  - 現実性: 多くの研究者や開発チームにとって、リソース的に実施困難。

# パラメータ効率の良いファインチューニング (PEFT)

- PEFT: より少ないコストでファインチューニング
- PEFT (Parameter-Efficient Fine-Tuning): 大規模モデルのファインチューニングを、より少ない計算資源・パラメータで行う手法群。
- 目的: フルファインチューニングに近い性能を、低コストで実現する。
- 従来手法の例: アダプタ層の挿入、プレフィックスチューニングなど。
- 従来手法の課題: 推論時の遅延増加、最適化の難しさなどがあった。

# LoRAとは？ - 概要

- LoRA (Low-Rank Adaptation): 新しいPEFT手法

- Microsoftの研究者ら (Hu et al., 2021) によって提案されたPEFT手法。
- 核心アイデア:
  - 元の巨大なモデルの重み ($W_0$) は 凍結 (変更しない)。
  - 代わりに、各層に 「低ランク行列」 で構成される小さなモジュールを追加し、このモジュールだけを学習 する。
- これにより、モデル全体を作り直すことなく、タスクに適応させる。

# LoRAの驚くべき効率性

- 学習パラメータ数の劇的削減:
  - GPT-3 (1750億パラメータ) の場合、LoRAでは学習対象が 約1/10,000 (約1800万パラメータ) にまで減少。
- GPUメモリ消費量の削減:
  - フルファインチューニング比で 約1/3 に圧縮。
- 高性能の維持:
  - 削減されたパラメータ数にも関わらず、モデルの品質はフルファインチューニングと同等か、場合によってはそれ以上。
- 結論: LoRAは、極めて少ない追加コストで大規模モデルを効率的に適応させる有望な技術。

# LoRAの理論的根拠: 低ランク仮説

- なぜLoRAは機能するのか？ 低ランク仮説
- 基盤となる考え: 大規模言語モデルなどを特定のタスクに適応させる際の「重みの変化（ΔW)」は、本質的に低い次元（低いランク）の構造を持つ という仮説。
- 意味:
  - 大規模モデルはパラメータに冗長性を含む。
  - タスク適応に必要な「真のパラメータ変化」は、元のパラメータ空間よりもずっと小さい部分空間で表現できる。
- LoRAはこの性質を利用し、学習対象を低ランク空間に限定することで効率化を図る。

電脳大使工業

# LoRAの仕組み (1): 低ランク行列分解

- LoRAのコア技術: 低ランク行列による差分近似
- 元の重み行列 $W_0$ (サイズ d × k) は凍結。
- 学習したい変化分 ΔW を直接学習する代わりに、 ΔW を2つの低ランク行列 B と A の積で近似する。
- $\Delta W \approx B A$
  - B: サイズ d × r
  - A: サイズ r × k
  - r: ランク (Rank)。非常に小さい整数 ($r \ll \min(d, k)$)。
- 学習対象: $W_0$ は固定し、小さな行列 A と B のみ学習。
- パラメータ数: 元の d × k 個から r × (d+k) 個に大幅削減。

# LoRAの仕組み (2): 前向き計算

- LoRA適用時の計算フロー
- 通常のモデル: 出力 $h = W_0 x$ ($x$ は入力)
- LoRA適用モデル: 出力 $h = W_0 x + (B A) x$
  - 元の計算 ($W_0 x$) に、LoRAモジュールによる補正項 ($(B A) x$) が加算される。
- 概念図の説明:
- 元の重み ($W_0$): 事前学習済み、凍結。
- LoRA重み ($A$, $B$): 学習対象。
- 入力 $x$ は両方の経路を通り、結果が足し合わされる。

# LoRAの仕組み (3): 初期化とスケーリング

- 学習の安定化と調整: 初期化とスケーリング係数 $\alpha$
- 初期化:
  - A 行列: 小さな乱数 (例: ガウス分布) で初期化。
  - B 行列: **ゼロ** で初期化。
  - 効果: 学習開始時、BA=0 となり、出力は元のモデルと同じ。これにより学習初期の挙動が安定する。
- スケーリング係数 $\alpha$:
  - LoRAの出力を調整するハイパーパラメータ。
  - 式: $h = W_0 x + (\alpha / r) * (B A) x$
  - $\alpha$ と r の比 ($\alpha / r$) で、LoRAによる補正の影響度を調整。
  - 論文では $\alpha = r$ (例: $\alpha = 16$, $r = 16$) を使用し、元の重みとLoRAのバランスを取るケースが多い。

# LoRAの利点: 学習効率

- 利点①: 劇的な学習コストの削減
- 学習パラメータ数削減: 桁違いに少ない (例: GPT-3で1/10000)。
- メモリ削減: 勾配やオプティマイザ状態の保存に必要なメモリ量が大幅に減少 (例: GPT-3で1/3)。
- 結果:
  - より少ないGPUリソースでの学習が可能に。
  - 同じリソースで、より大きなバッチサイズの使用や学習時間の短縮が可能。

# LoRAの利点: 推論効率

- 利点②: 推論時のオーバーヘッドがほぼゼロ
- 学習後、LoRAモジュール (A, B) を元の重み $W_0$ に マージ（統合） することが可能。
- $W\_merged = W_0 + B\ A$
- マージ後のモデルは、追加の計算レイヤーなしに、通常のモデルと同じように推論できる。
- 結果: 推論速度（レイテンシ）への悪影響は基本的にない。
- これは、推論時に追加層の計算が必要なアダプタ方式に対する大きな利点。

# LoRAの利点: 高い性能

- 利点③: 効率性と性能の両立
- LoRAは、パラメータ数を大幅に削減しながらも、モデルの性能を高く維持できる。
- Huらの実験では、RoBERTa, DeBERTa, GPT-2, GPT-3など様々なモデル・タスクにおいて、フルファインチューニングと同等、あるいはそれを上回る精度を達成。
- 示唆: 低ランク適応というアプローチが、タスクに必要な情報を効率的に捉えられていることを示している。

# PEFT手法の概要と比較対象

- 様々なPEFTアプローチ
- LoRA以外にも、パラメータ効率の良いファインチューニング手法が存在する。

- 比較対象:
  - フルファインチューニング: ベースライン。全パラメータを学習。
  - アダプタ層 (Adapter Tuning): モデルの層間に小さな「アダプタ」モジュールを挿入し、そこだけ学習。
  - プレフィックスチューニング (Prefix-Tuning): 入力シーケンスの前に学習可能な「プレフィックス」ベクトルを追加。
  - LoRA: 重み行列の「変化分」を低ランク行列で近似。

# PEFT手法比較: アダプタ層 vs LoRA

- アダプタ層 (Adapter Tuning) との比較
- アダプタ層:
  - Transformerブロック内にボトルネック構造の小規模全結合層を追加。
  - 追加層のみ学習。パラメータ効率は良い。
  - 欠点: 推論時に追加層の計算が必要となり、わずかに **遅延が増加** する。
- LoRA:
  - 重み行列自体に作用。追加の「層」ではない。
  - 推論時にマージ可能で、遅延増加なし。
  - 性能面でもアダプタを上回るケースが多いとされる。

# PEFT手法比較: プレフィックスチューニング vs LoRA

- プレフィックスチューニング (Prefix-Tuning) との比較
- プレフィックスチューニング:
  - 入力の先頭に学習可能なベクトル列（ソフトプロンプト）を追加。
  - モデルの重みは完全に凍結。学習パラメータは非常に少ない。
  - 課題: モデルの挙動を間接的に制御するため、最適化が難しい場合があり、性能が不安定になることも。シーケンス長が実質的に伸びる。
- LoRA:
  - モデルの重み更新を直接的に（ただし低ランク空間で）行う。
  - 一般的に、プレフィックスチューニングより安定して高い性能を示しやすい。

# LoRAの利点: モジュール性と共有

- LoRAがもたらす運用上のメリット: 軽量性とモジュール性
- 軽量な差分ファイル:
  - 学習されたLoRAモジュール (A, B) は、数十MB程度の非常に小さなファイルサイズ。
  - 例: Stable Diffusion (数GB) に対するLoRAは10MB～100MB程度。
- タスクスイッチングの容易さ:
  - 1つのベースモデルに対し、異なるタスク用のLoRAファイルを動的に読み込み、切り替えることが可能。
  - 巨大なモデル全体を再ロードする必要がない。
- 配布・管理の効率化: ベースモデルは1つ共有し、タスク固有のLoRAファイルだけを配布・管理すれば良い。

# LoRAの応用範囲: 言語モデルから画像生成へ

- LoRAの適用: LLMから画像生成モデルへ
- LoRAは元々LLM向けに開発されたが、その有効性から画像生成モデル、特にStable Diffusion (SD)のような潜在拡散モデルで広く使われている。
- 目的: 特定の画風、キャラクター、オブジェクトなどを効率的に追加学習させる。

# Stable DiffusionとLoRAの適用箇所

- Stable DiffusionにおけるLoRAの適用ポイント
- Stable Diffusionの構成要素: U-Net (画像生成コア), テキストエンコーダ (CLIPなど), VAE。
- キーコンポーネント: クロスアテンション層
  - U-Net内に存在し、テキスト情報 (プロンプト) と画像特徴を結びつける重要な役割。
  - 例: 「猫」という単語と画像内の「猫」領域を関連付ける。

- LoRAの主な適用箇所: このクロスアテンション層内の線形変換 (WQ, WK, WV 行列) にLoRAモジュールを適用する。

# 画像生成におけるLoRAのメリット(1)

- メリット①: 学習コストの大幅削減

- SDモデル全体（数億パラメータ）を再学習する代わりに、LoRAモジュール（数百万パラメータ、元の1%未満の場合も）のみを学習。

- 結果:
  - 個人ユーザーのGPU環境でも、SDのような大規模モデルのファインチューニングが現実的に可能に。
  - 学習時間の大幅な短縮。

電脳大使工業

# 画像生成におけるLoRAのメリット(2)

- メリット②: 高品質 & 少量データでの学習

- 高品質な生成: LoRAで微調整しても、元のSDモデルに近い、あるいはそれ以上の画質を維持できることが多い。

- 少量データでの適応:
  - 驚くほど少ない枚数 (例:5〜10枚程度) の画像からでも、特定のキャラクターやスタイルを学習可能。
  - DreamBooth (特定対象の学習技術) との組み合わせも効果的。
  - データ準備のハードルが大幅に下がる。

電脳大使工業

# 画像生成におけるLoRAのメリット(3)

- メリット③: 推論速度と運用の容易さ

- 推論速度: LoRAモジュールによる追加計算量はわずかなため、画像生成速度への影響はほとんどない。

- 運用の容易さ:
  - ベースのSDモデルをメモリにロードしておけば、LoRAファイル(スタイルやキャラ)を切り替えるだけで生成結果を変更可能。
  - モデル全体の再起動が不要で、インタラクティブな利用やサービス提供に有利。

# LoRA + 画像生成 の活用例

- LoRAによる画像生成の広がり
- 研究: 高解像度化、欠損補完など特定タスクへのSDモデル特化。
- ビジネス: ブランドイメージに合わせた画像生成、製品ビジュアル作成、パーソナライズド画像サービス。
- クリエイター: 自身の画風をLoRA化、作品制作の効率化・多様化。
- コミュニティ: Hugging Face Diffusersでのサポート、Civitaiでの無数のLoRA共有 (キャラ、スタイルなど)。
- LoRAは、プロンプトだけでは難しい独自スタイルの実現を可能にする。

# LoRAの限界: フルチューニングとの比較

- 課題①: 万能ではない？ フルチューニングとのトレードオフ
- LoRAは多くの場合フルFTに匹敵するが、タスクによっては差が出ることも。
- 特に苦手な可能性: 事前学習データから大きく離れた、全く新しい知識の獲得が要求されるタスク。
  - 例: モデルが知らない専門分野のプログラミングコード生成などでは、フルFTに劣る可能性が報告されている。
- トレードオフ:
  - LoRA:新知識獲得量は限定的かもしれないが、元のモデルの知識（既存能力）を壊しにくい（忘却が少ない）。
  - フルFT:新知識獲得に強いが、元の知識を忘却しやすい。

# LoRAの課題: ハイパーパラメータ選択

- 課題②: 最適な設定は？ ランクと α

- ランク r の選択:
  - 性能とパラメータ数のトレードオフ。
  - r が小さすぎると表現力不足、大きすぎると効率低下。
  - 最適な r の選択は経験的に行われることが多い。
  - 研究: タスクや層に応じて r を動的に調整する手法 (例: AdaLoRA)。

- スケーリング係数 α の選択:
  - α/r の比率が重要だが、最適な値や理論的根拠はまだ研究途上。
  - 学習安定化のための正規化手法なども提案されている (例: $\alpha/\sqrt{r}$)。

# LoRAの進化と派生技術

- LoRAの進化: より賢く、より効率的に
- LoRA自体も改良が進んでいる。
- DoRA (Weight-Decomposed LoRA):
  - 重み更新を「大きさ」と「方向」に分離して学習。
  - LoRAよりも少ないパラメータ (低いランク) で同等以上の性能を発揮する可能性。ランク選択への感度も低いとされる。
- QLoRA (Quantized LoRA):
  - ベースモデルを低ビット (例: 4bit) に量子化し、メモリ使用量を劇的に削減。
  - その量子化モデルに対してLoRAでファインチューニング。
  - 効果: 例えば65Bモデルを単一GPU (48GB) で学習可能に。性能は16bitフル精度FTに匹敵。

# LoRAの課題: 解釈性と安全性

- 課題③: 中身はどうなっている？ 解釈性と安全性

- モデル解釈性:
  - LoRAモジュールが具体的にモデルのどの部分に、どのように影響を与えているのか (例: 注意機構、特徴表現) の理解はまだ十分ではない。
  - 内部挙動の可視化・説明は、信頼性向上やデバッグに重要。

- 安全性・ロバスト性:
  - LoRAによるモデル改変が、予期せぬ挙動や脆弱性を生まないか。
  - 悪意のある入力に対する耐性など、安全性評価の枠組みも必要。

# LoRAの今後の展望

- LoRAの未来: さらなる発展と普及
- 自動化: 最適ランク r や適用層の自動選択技術の発展。
- ツール・エコシステム: Hugging Face PEFTライブラリなど、LoRA利用を支援する環境のさらなる充実。
- 標準技術へ: 大規模モデルのカスタマイズにおける、デファクトスタンダードとしての地位を確立していく可能性。

- 結論: LoRAとその派生技術は、計算資源の制約を乗り越え、AIモデルの応用を加速させるための不可欠な技術であり続けるだろう。

# まとめ: LoRAのインパクト

- LoRA: 大規模モデルを、もっと身近に、もっと便利に

- LoRAは、大規模モデルのファインチューニングにおける **コストと性能のトレードオフを劇的に改善** した革新的技術。
- キーポイント:
  - 圧倒的なパラメータ効率:学習コストとストレージを大幅削減。
  - 高い性能:フルファインチューニングに匹敵。
  - 推論効率:追加の遅延なし。
  - モジュール性:軽量な差分ファイルによる容易な管理・切り替え。
  - 広範な応用:LLMから画像生成まで。
  - 活発な研究開発:QLoRA, DoRAなど進化が継続。