








ナイス視点！

それぞれの工夫が「**GPU 専用なのか、CPU でも意味があるのか**」という観点で整理すると、**リソースに応じた使い分けが明確**になります。以下に一覧と補足を示します。







 **GPU 使用時に特に効果的（または必須）、CPU ではほぼ効果なし・無意味**

手法	GPU 専用に近い	CPU でも意味あるか？
LoRA	 高速な GPU 推論前提	 CPU でも動くが、恩恵は薄い（学習は遅い）
Gradient checkpointing	 GPU メモリ削減が目的	 CPU では計算コストが増え逆効果に近い
BF16 / FP16 mixed precision	 GPU でこそ意味あり（特に Ampere 以上）	 CPU では基本非対応 or 遅くなる

 補足

- LoRA は CPU での学習も可能だけど、**高速に微調整**できるというメリットが薄れる。
- Mixed Precision (BF16/FP16) は、**CPU では整数演算との相性も悪く、精度や速度面でメリットがほぼなし**。
- Gradient checkpointing は、**GPU のメモリ節約のために再計算する技術**で、CPU では計算負荷が重くなって逆に遅くなる。

 **CPU でもある程度使えるが、GPU でこそ最大の恩恵あり**

手法	GPU 向け	CPU 向けでも OK？
Batching optimizations		 効果あり
Data packing		 （ただし主に LLM 系列）
Sequence length limit		 （長さが増えれば遅くなるのは CPU も同様）

 補足

- バッチ処理やデータパッキングは CPU でも動作効率に貢献。特に**小型モデルで大量のサンプル処理**をしたいときには有効。
- シーケンス長の制限も、CPU で長文処理すると当然遅くなるので、**制限**

する意味はある（ただし学習じゃなく推論時に効く）。

← 総まとめ（CPU vs GPU）

手法	GPU 時有効	CPU 時有効	コメント
LoRA	✓	⚠ 微調整は可だが遅い	
Gradient checkpointing	✓	✗ 再計算が重すぎる	
BF16 / FP16	✓（Ampere 以上）	✗ 基本非対応 or 遅い	
Batching optimizations	✓	✓ 効果あり	
Sequence length limit	✓	✓ 推論高速化に有効	
Data packing	✓	✓（トークン系列限定）	

もし CPU 環境でやる場合は、

- 「**Batching 最適化**」「**Sequence 制限**」「**LoRA 推論のみ**」は比較的恩恵があり、
- 「**Gradient checkpointing**」「**BF16/FP16**」は基本避けたほうが無難です。

CPU でも賢くやりたい方向けに、PyTorch や HuggingFace Transformers の**低リソース最適化 Tips**もあるので、必要ならそっちも紹介できますよ！