# COAP: Memory-Efficient Training with Correlation-Aware Gradient Projection

Jinqi Xiao[1,2] · Shen Sang[1] · Tiancheng Zhi[1] · Jing Liu[1] · Qing Yan[1] · Yuqian Zhang[2] · Linjie Luo[1] · Bo Yuan[2]

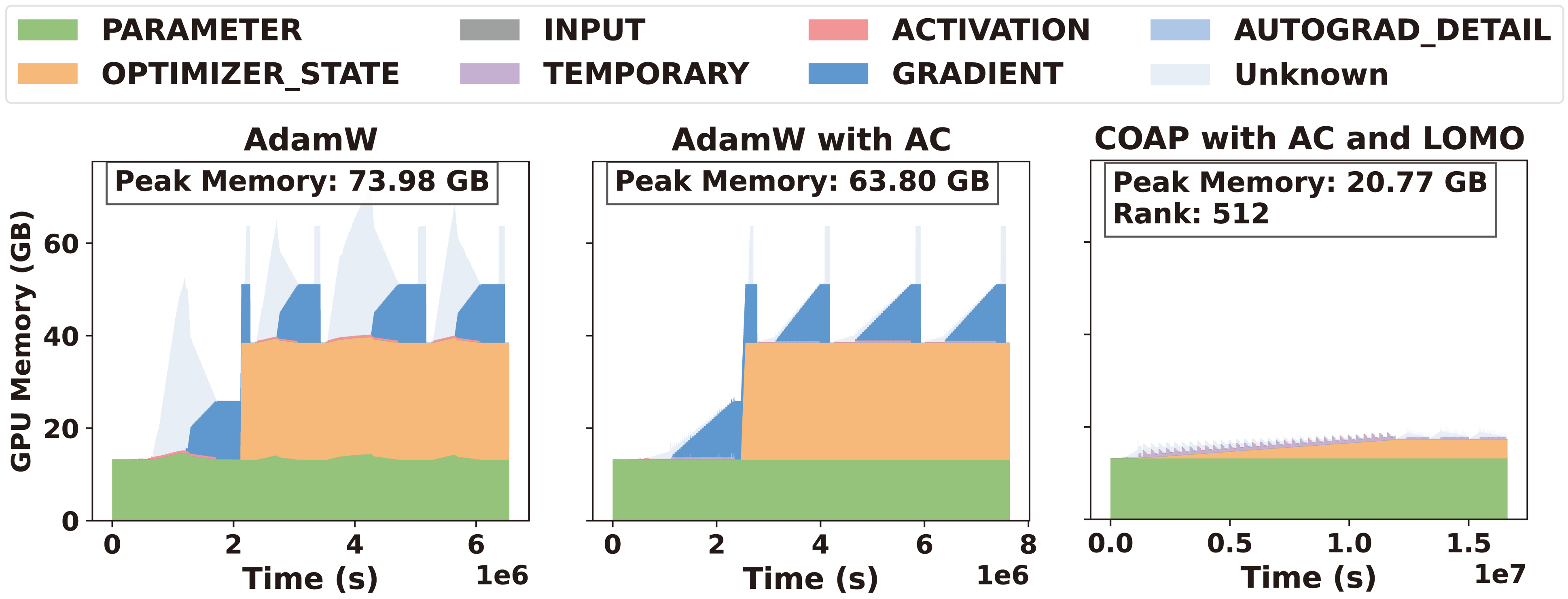
[1]ByteDance Inc. [2]Rutgers University

COAP Optimizer

CVPR Nashville JUNE 11-15, 2025

## Motivation & Overview

*Figure 1: Profiling the GPU memory usage during the training stage of LLaVA-v1.5-7B on 1xA100*



❖**Challenge**:

1. Optimizer states are a major memory bottleneck in large-scale model training.
2. Reducing memory costs often incurs high computational costs or compromises training stability & model performance.

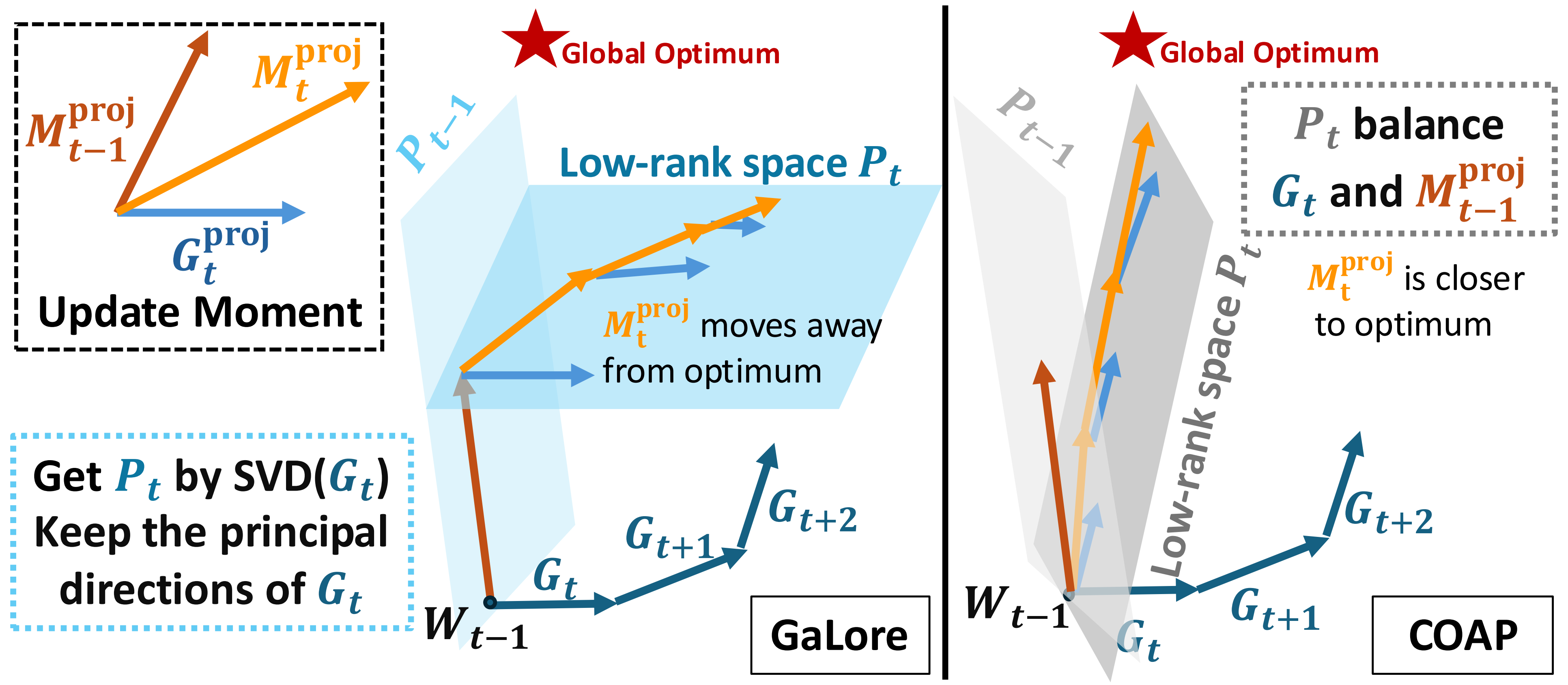
❖**Solution (COAP)**:

1. **Correlation-Aware Projection Update:** Smoothly evolves the projection matrix, leveraging prior optimization history for consistent gradient representation.
2. **Low-Cost SVD Recalibration:** Using occasional low-cost SVD to recalibrate the low-rank projection matrices, slashing computational cost (O(mn²) → O(mr²)) and promoting robust optimization.

❖**Strengths** :

1. **Significant Memory Reduction**: Up to 81% optimizer memory savings with minimal overhead (e.g., +2% training time).
2. **Maintained/Improved Performance:** Achieves comparable or superior model quality to full-rank training.
3. **Seamless Integration:** Easily integrates with optimizers (e.g., AdamW, Adafactor). Effectiveness proven across multimodal, diffusion, and large language models (LLMs).

## Proposed Method



For a weight matrix $W \in \mathbb{R}^{m\times n}$, the corresponding gradient matrix at time step $t$ can be denoted as $G_t = \nabla_W \mathcal{L}(W) \in \mathbb{R}^{m\times n}$.

Then, the general weight update process can be formulated as:
$W_{t+1} = W_t - \eta\rho_t(G_t)$ ($\eta$ is the learning rate, $\rho_t$ adjusts the gradients.)

### 1. Inter-projection Correlation-aware $P$ Update

$$\min_P \underbrace{\mathrm{MSE}(\hat{G}, G)}_{\text{reconstruction term}} \underbrace{(1 - \mathrm{CosSim}(\hat{M}, G))}_{\text{direction term}},$$

### 2. Occasional Low-cost SVD

$$Q_{\text{red}}, \_ = \mathrm{QR}_{\text{red}}(G_t P_{t-1}),$$
$$U, \Sigma, Z^\top = \mathrm{SVD}(Q_{\text{red}}^\top G_t),$$
$$P_t = Z$$

### Training with Low-rank Gradient

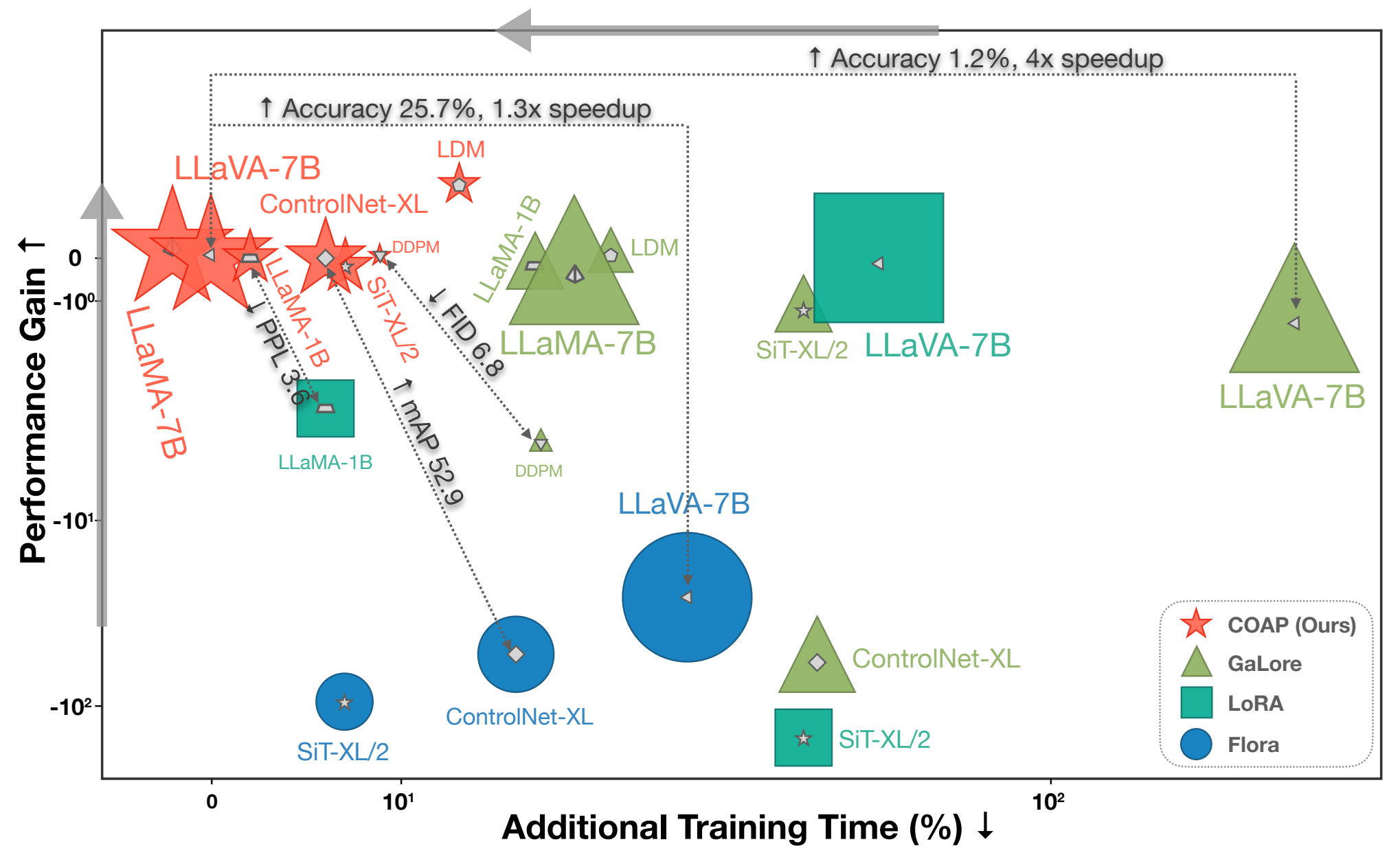
$$M_t^{\text{proj}} = \beta_1 M_{t-1}^{\text{proj}} + (1-\beta_1) G_t^{\text{proj}}$$
$$V_t^{\text{proj}} = \beta_2 V_{t-1}^{\text{proj}} + (1-\beta_2)(G_t^{\text{proj}})^2$$
$$W_{t+1} = W_t - \eta\rho_t(G_t^{\text{proj}})$$
$$\rho_t(G_t^{\text{proj}}) = \frac{M_t^{\text{proj}}/(1-\beta_1^t)}{\sqrt{V_t^{\text{proj}}/(1-\beta_2^t)} + \epsilon} P_t^\top$$
$$G_t^{\text{proj}} = G_t P_t$$

### Adam with COAP

```
Input: Weight matrix $\boldsymbol{W} \in \mathbb{R}^{m\times n}$, Learning rate $\eta$, Rank $r$,
       Betas $[\beta_1, \beta_2]$, Update interval $[\lambda, T_u]$.
Initialize: $\boldsymbol{M}_0^{\text{proj}} \in \mathbb{R}^{m\times r} \leftarrow 0$, $\boldsymbol{V}_0^{\text{proj}} \in \mathbb{R}^{m\times r} \leftarrow 0$, $t \leftarrow 0$
Randomly Initialize: $\boldsymbol{P}_0 \in \mathbb{R}^{n\times r}$
Compute: $\boldsymbol{P}_0 \leftarrow (\boldsymbol{P}_0, \boldsymbol{G}_0)$    ▷ Step 2
for $t$ in $[1, 2, \cdots]$ do
    Compute: gradient $\boldsymbol{G}_t$ of $\boldsymbol{W}_t$ in the loss function.
    if $t \mod T_u = 0$ then
        if $t \mod (\lambda \times T_u) = 0$ then
            Compute: $\boldsymbol{P}_t \leftarrow (\boldsymbol{P}_{t-1}, \boldsymbol{G}_t)$    ▷ Step 2
        else
            Update: $\boldsymbol{P}_t \leftarrow (\boldsymbol{P}_{t-1}, \boldsymbol{G}_t, \boldsymbol{M}_{t-1})$    ▷ Step 1
    else
        $\boldsymbol{P}_t \leftarrow \boldsymbol{P}_{t-1}$
    ▷Project gradient and moments into low-rank space.
    $\boldsymbol{G}_t^{\text{proj}} \leftarrow \boldsymbol{G}_t \boldsymbol{P}_t$
    $\boldsymbol{M}_t^{\text{proj}} \leftarrow \beta_1 \boldsymbol{M}_{t-1}^{\text{proj}} + (1-\beta_1)\boldsymbol{G}_t^{\text{proj}}$
    $\boldsymbol{V}_t^{\text{proj}} \leftarrow \beta_2 \boldsymbol{V}_{t-1}^{\text{proj}} + (1-\beta_2)(\boldsymbol{G}_t^{\text{proj}})^2$
    ▷Calculate the bias correction term in low-rank space.
    $\Delta\boldsymbol{W}_t^{\text{proj}} \leftarrow \frac{\boldsymbol{M}_t^{\text{proj}}/(1-\beta_1^t)}{\sqrt{\boldsymbol{V}_t^{\text{proj}}/(1-\beta_2^t)}+\epsilon}$
    ▷Restore $\Delta\boldsymbol{W}_t^{\text{proj}}$ to original space and update $\boldsymbol{W}$.
    $\boldsymbol{W}_t \leftarrow \boldsymbol{W}_{t-1} - \eta\Delta\boldsymbol{W}_t^{\text{proj}}\boldsymbol{P}_t^\top$
Return: updated $\boldsymbol{W}$
```

## Main Results

*Comparison between COAP and other low-rank-based methods.*



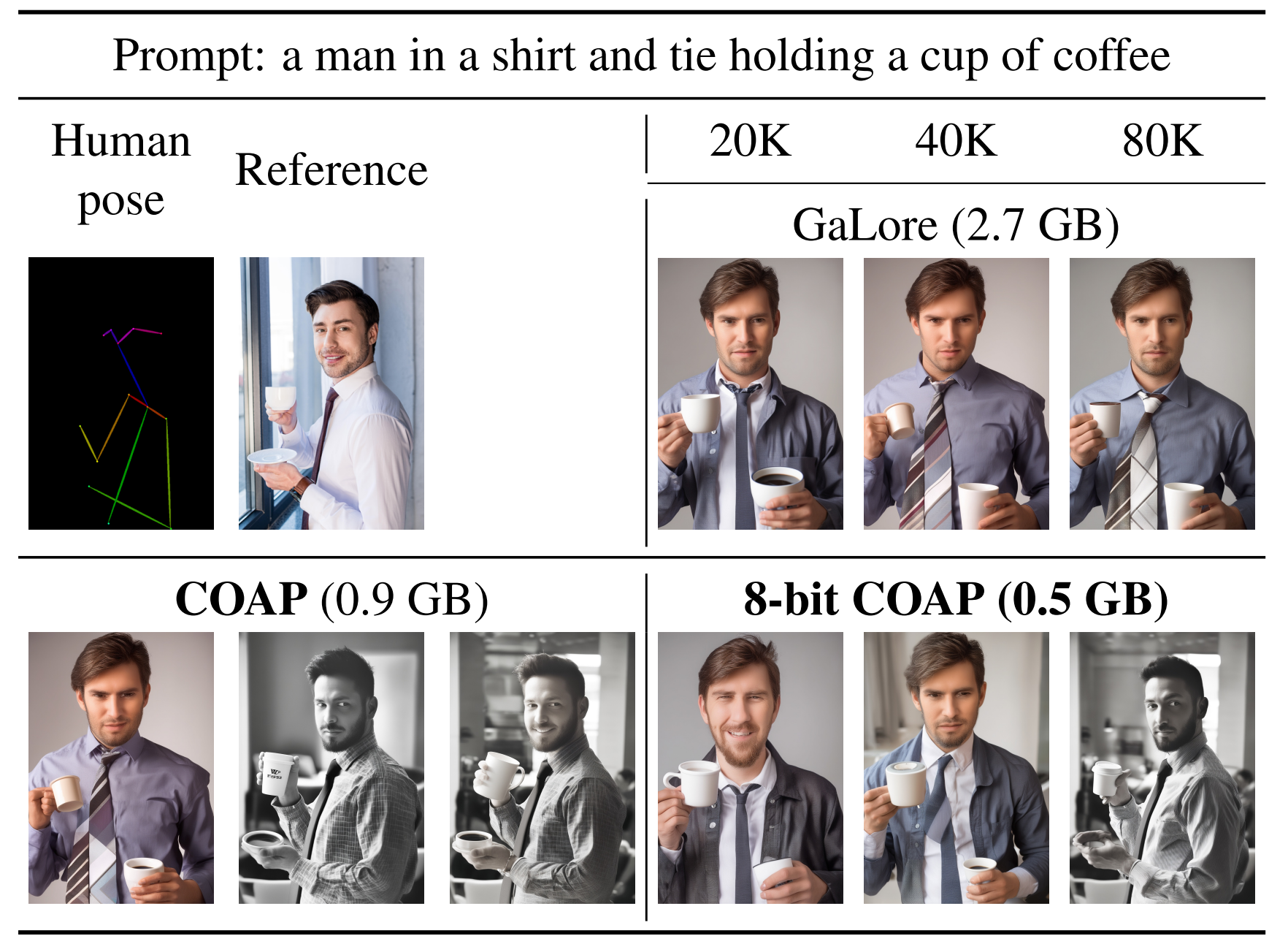
*Pre-training SiT-XL/2 with REPA on the ImageNet-1K dataset for 400K steps using 8xH100 GPUs.*

| Method | Optimizer Mem. (GB)↓ | Model Mem. (GB)↓ | Training Time↓ | FID↓ |
|---|---|---|---|---|
| AdamW | 5.1 | 2.5 | 20.8 h | 1.9 |
| GaLore | 2.6 (-49%) | 2.5 | +38% | 2.3 |
| LoRA | 3.6 (-29%) | 3.7 (+48%) | +33% | 151.9 |
| ReLoRA | 3.6 (-29%) | 3.7 (+48%) | +33% | 151.8 |
| **COAP** | **2.6 (-49%)** | 2.5 | **+14%** | **2.2** |
| Adafactor | 2.5 | 2.5 | 25.5 h | 1.9 |
| GaLore | 1.5 (-40%) | 2.5 | +33% | 3.0 |
| Flora | 1.6 (-36%) | 2.5 | +7% | 115.2 |
| **COAP** | **1.5 (-40%)** | 2.5 | **+7%** | **2.1** |

*Fine-tuning LLaVA-v1.5-7B on the ScienceQA dataset using 1xA100. ``OOM" means out-of-memory.*

| Method | Optimizer Mem. (GB)↓ | Model Mem. (GB)↓ | Training Time↓ | FID↓ |
|---|---|---|---|---|
| AdamW | 5.1 | 2.5 | 20.8 h | 1.9 |
| GaLore | 2.6 (-49%) | 2.5 | +38% | 2.3 |
| LoRA | 3.6 (-29%) | 3.7 (+48%) | +33% | 151.9 |
| ReLoRA | 3.6 (-29%) | 3.7 (+48%) | +33% | 151.8 |
| **COAP** | **2.6 (-49%)** | 2.5 | **+14%** | **2.2** |
| Adafactor | 2.5 | 2.5 | 25.5 h | 1.9 |
| GaLore | 1.5 (-40%) | 2.5 | +33% | 3.0 |
| Flora | 1.6 (-36%) | 2.5 | +7% | 115.2 |
| **COAP** | **1.5 (-40%)** | 2.5 | **+7%** | **2.1** |

*Comparison of generated images at different training steps (20K, 40K, 80K).*



*Pre-training LLaMA-1B and LLaMA-7B on the C4 dataset using 8xH100 GPUs.*

| Model | Method | Optimizer Mem. (GB)↓ | Model Mem. (GB)↓ | Training Time↓ | PPL↓ |
|---|---|---|---|---|---|
| LLaMA 1B (100K) | AdamW | 4.99 | 2.49 | 28.50 h | 15.56 |
| | GaLore | 1.94 (-61%) | 2.49 | +17% | 15.64 |
| | LoRA | 2.27 (-55%) | 3.38 (+36%) | +6% | 19.21 |
| | ReLoRA | 2.27 (-55%) | 3.38 (+36%) | +6% | 18.33 |
| | **COAP** | **1.94 (-61%)** | 2.49 | **+2%** | **15.56** |
| LLaMA 7B (80K) | 8-bit Adam | 12.55 | 12.55 | 52.01 h | 15.39 |
| | 8-bit GaLore | 5.25 (-58%) | 12.55 | +19% | 15.47 |
| | **8-bit COAP** | **5.25 (-58%)** | 12.55 | **-2%** | **15.28** |

*Training ControlNet based on SDXL for 80K steps using 8xH100 GPUs in BF16 format, conditioned on human poses.*

| Method | Rank Ratio | Optimizer Mem. (GB)↓ | mAP↑ @ training steps | | | Converged | Training Time (80K)↓ |
|---|---|---|---|---|---|---|---|
| | | | 20K | 40K | 80K | | |
| AdamW | - | 9.3 | 18.3 | 19.1 | 19.9 | ✗ | 19.3 h |
| Adafactor | - | 5.1 | 19.2 | 70.0 | 72.7 | ✓ | 22.3 h |
| Flora | 2 | 3.9 (-24%) | 18.0 | 18.9 | 19.6 | ✗ | +16% |
| GaLore | 2 | 4.7 (-8%) | 18.6 | 67.0 | 72.7 | ✓ | +39% |
| GaLore-8bit | 2 | 3.1 (-39%) | 19.6 | 20.8 | 20.9 | ✗ | +49% |
| **COAP** | 2 | 3.6 (-29%) | **66.6** | **71.6** | **73.4** | ✓ | **+4%** |
| **8-bit COAP** | 2 | **1.9 (-63%)** | 18.7 | 66.9 | 72.2 | ✓ | +17% |
| GaLore | 4 | 3.5 (-31%) | 18.9 | 19.7 | 19.5 | ✗ | +39% |
| 8-bit GaLore | 4 | 3.1 (-39%) | 18.8 | 19.7 | 19.8 | ✗ | +50% |
| **COAP** | 4 | 1.8 (-65%) | **50.9** | **70.4** | **72.1** | ✓ | **+5%** |
| **8-bit COAP** | 4 | **1.0 (-80%)** | 19.4 | 19.2 | 71.5 | ✓ | +15% |
| GaLore | 8 | 2.7 (-47%) | 18.6 | 18.2 | 19.7 | ✗ | +35% |
| 8-bit GaLore | 8 | 2.3 (-55%) | 18.6 | 18.2 | 19.7 | ✗ | +45% |
| **COAP** | 8 | 0.9 (-82%) | **25.8** | **70.2** | **72.6** | ✓ | **+6%** |
| **8-bit COAP** | 8 | **0.5 (-90%)** | 19.3 | 18.9 | 69.9 | ✓ | +13% |