Exploring Landscapes for Better Minima along Valleys

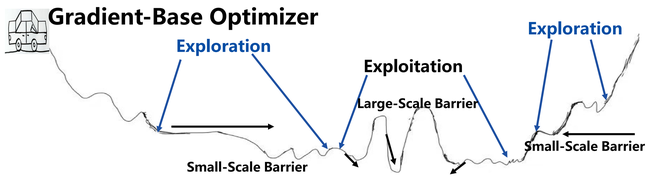
Tong Zhao [1,2] Jiacheng Li [1,2] Yuanchang Zhou [1,2] Guangming Tan [1,2] Weile Jia [1,2]
[1]Institute of Computing Technology, Chinese Academy of Sciences [2]University of Chinese Academy of Sciences

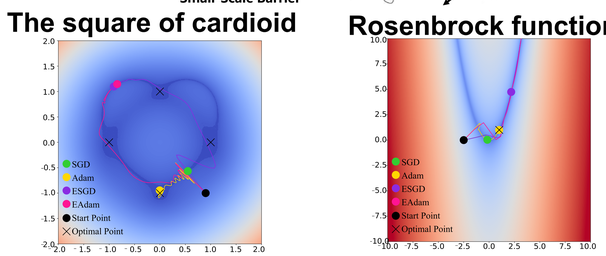
ABSTRACT

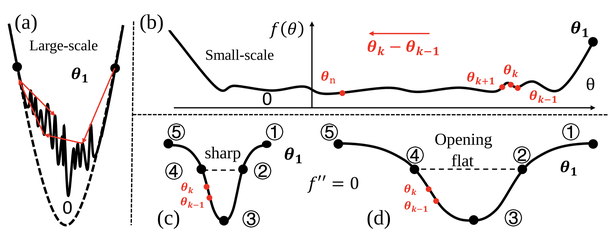
- Optimizers stop searching for possible better minima once they reach a minimum. Given the complex geometric properties of the loss landscape, it is difficult to guarantee that such a point is the lowest or provides the best generalization.
- We proposed "E" that enables a gradient-based optimizer to continue exploring potentially better local minima. This feature is well-suited for addressing the challenges of large-batch training, especially LLM pretraining task.
- In large-batching training task, the adapted Lamb (ALTO) outperforms other optimizers consistently. Also, the adaptor is applicable to small-batch training but the improvement is marginal.

Paper Code

Method







$-\nabla\|\nabla f(\boldsymbol{\theta}_k)\|^2 = -2\mathbf{H}_k\mathbf{g}_k,$

$-\mathbf{g}_k \approx \boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1},$

$-\nabla\|\nabla f(\boldsymbol{\theta}_k)\|^2 \approx \mathbf{H}_k(\boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1})$

$\approx \mathbf{g}_k - \mathbf{g}_{k-1}$ Find Accelerator!!!

Direction/Sign	①②	②③	③④	④⑤
$\boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1}$	-	-	-	-
$-\mathbf{g}_k$	-	-	+	+
$-\nabla\|\nabla f(\boldsymbol{\theta}_k)\|^2$	+	-	+	-
$\mathbf{g}_k - \mathbf{g}_{k-1}$	+	-	-	+
$\langle\boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1}, \mathbf{g}_k - \mathbf{g}_{k-1}\rangle$	-	+	+	-

$$\mathbf{g}_k \longleftarrow \underbrace{\mathbf{g}_k}_{\text{Large-Scale Exploitation}} + \underbrace{\alpha}_{\text{Scaling Factor}} \underbrace{\mathbf{a}_k}_{\text{Small-Scale Exploration}}$$

Scaling Factor: ABS for the difference of large-scale and small-scall, negative for exploration, and positive for exploitation.

$$\underbrace{\mathbf{a}_k}_{\text{EMA}of(\mathbf{g}_k-\mathbf{g}_{k-1})} = \underbrace{\beta_1}_{\text{Persistence Factor}} \mathbf{a}_{k-1} + (1-\beta_1)(\mathbf{g}_k - \mathbf{g}_{k-1}),$$

Persistence Factor: how long to explore and exploit.

Algorithm 1: ALTO

Input: initialize $\boldsymbol{\theta}_1$, learning rate η_k, EMA factors $(\beta_1, \beta_2, \beta_3) \in [0,1)^3$, stable parameter $\varepsilon_1, \varepsilon_2 > 0$, weight decay $\lambda_k > 0$, acceleration factor $|\alpha| < 1/(1-\beta_1)$, $\mathbf{a}_0 = \mathbf{0}$, $\mathbf{m}_0 = \mathbf{0}$, $\mathbf{v}_0 = 0$, and $\mathbf{g}_0 = \mathbf{0}$.

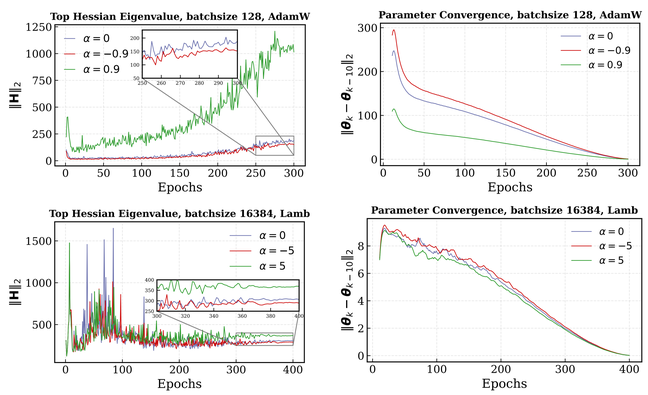
Output: $\{\boldsymbol{\theta}_k\}_{k=1}^T$.

while $k < T$ **do**
- $\mathbf{g}_k = \frac{1}{|\mathcal{Z}_k|}\Sigma_{\boldsymbol{\zeta}_i \in \mathcal{Z}_k} \nabla\ell(\boldsymbol{\theta}_k, \boldsymbol{\zeta}_i);$
- $\mathbf{a}_k = \beta_1\mathbf{a}_{k-1} + (1-\beta_1)(\mathbf{g}_k - \mathbf{g}_{k-1});$
- $\mathbf{m}_k = \beta_2\mathbf{m}_{k-1} + (1-\beta_2)(\mathbf{g}_k + \alpha\mathbf{a}_k);$
- $\mathbf{v}_k = \beta_3\mathbf{v}_{k-1} + (1-\beta_3)[\mathbf{g}_k + \alpha\mathbf{a}_k]^2;$
- $\hat{\mathbf{m}}_k = \mathbf{m}_k/(1-\beta_2^k); \hat{\mathbf{v}}_k = \mathbf{v}_k/(1-\beta_3^k);$
- $\boldsymbol{r}_k = \hat{\mathbf{m}}_k/(\sqrt{\hat{\mathbf{v}}_k} + \varepsilon_1) + \lambda_k\boldsymbol{\theta}_k$
- $\boldsymbol{\theta}_{k+1}^{(i)} = \boldsymbol{\theta}_k^{(i)} - \frac{\eta_k \mathbf{r}_k^{(i)}\phi(\|\boldsymbol{\theta}_k^{(i)}\|)}{(\|\mathbf{r}_k^{(i)}\| + \varepsilon_2\phi(\|\boldsymbol{\theta}_k^{(i)}\|))}$

end while

Analysis

1. $\alpha < 0$ converges slowly but tends to flat minima.
2. $\alpha > 0$ converges fast but tends to sharp minima.



Convergence Analysis Non-Convex:

$$\frac{1}{T+1}\sum_{k=0}^{T}\mathbb{E}\left(\|\nabla f(\boldsymbol{\theta}_k)\|^2\right) \le \mathcal{O}\left(\frac{1}{\mathbf{T}}\right),$$

if common assumptions holds and $\|\boldsymbol{\theta}_k\| \le D$.

Convergence Analysis Convex:

$$R(T) := \sum_{k=1}^{T}(\ell_k(\boldsymbol{\theta}_k) - \ell_k(\boldsymbol{\theta}^*)) \le \mathcal{O}(\sqrt{\mathbf{T}}),$$

if common assumptions holds and $\forall i \in [d]$,

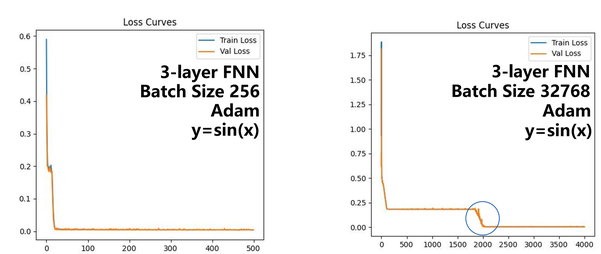
$$\frac{\sqrt{k}\left(\sqrt{\mathbf{v}_{k,i}} + \varepsilon_1\right)\left(\left\|\mathbf{r}_k^{(i)}\right\| + \varepsilon_2\phi\left(\left\|\boldsymbol{\theta}_k^{(i)}\right\|\right)\right)}{\phi\left(\left\|\boldsymbol{\theta}_k^{(i)}\right\|\right)}$$

increases monotonically with respect to k (similar condition also exists in Adam).

Experiments

Small Batch: applicable, but advantage is marginal.
Large Batch: very suitable, the focus.

1. **Why large-batch training?** Accelerate training.
2. **What challenges?** #iteration is limited, but learning rate is bounded.
3. **How to solve?** Effectively and softly enlarge learning rate by the exploration attribute.



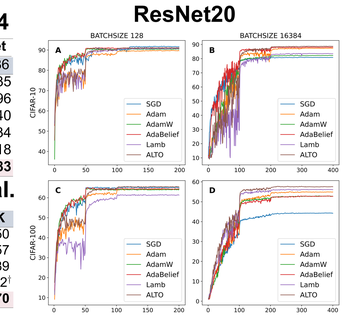
ResNet20 ResNet34

Dataset	CIFAR10		CIFAR100		ImageNet	
Batch Size	128	16384	128	16384	256	4086
SGD	**91.85**	80.86	64.93	44.20	**70.64**	49.35
Adam	89.88	87.34	64.35	54.91	65.06	54.96
AdamW	90.54	82.29	64.62	52.95	69.64	68.40
Lamb	90.89	83.56	61.29	56.06	69.17	70.34
AdaBelief	91.12	88.03	64.44	52.94	70.12	70.18
ALTO	91.24	**88.83**	**65.74**	**57.78**	69.95	**70.83**

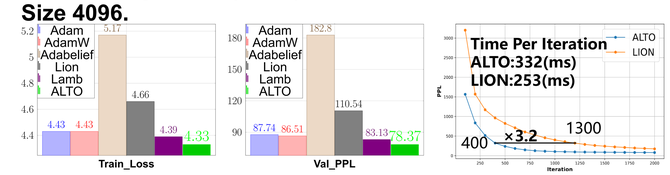
ImageNet, ResNet50.† orginal.

Batch Size	1K	2K	4K	8K	16K	32K
Adam	73.08	73.08	73.32	73.11	73.09	72.50
AdamW	75.65	74.93	74.65	74.40	74.10	73.57
Adabelief	73.32	73.48	73.41	73.14	73.00	72.89
Lamb	77.06†	77.11†	76.92†	76.89†	76.66†	76.42†
ALTO	**77.22**	**77.25**	**77.35**	**77.10**	**76.87**	**76.70**

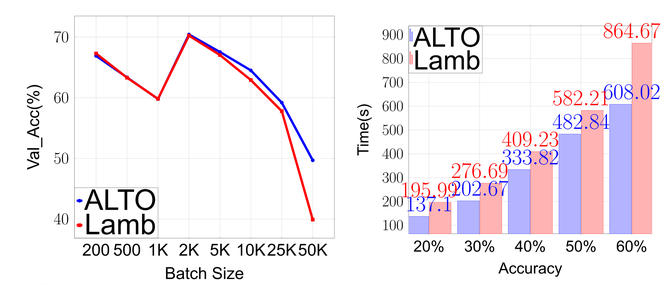
ResNet20



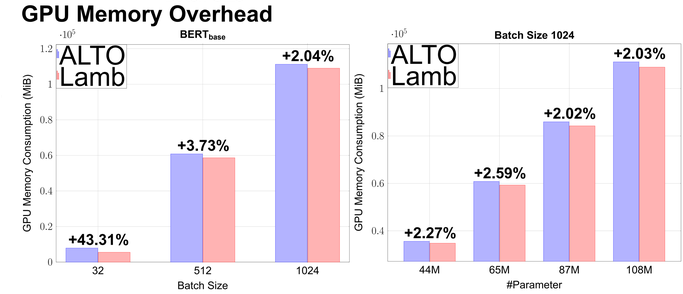
Pre-Training, GPT2 (345M), GPT2-Output-Dataset, Batch Size 4096.



Batch Size Scaling and Wall-Clock Time to Certain Accuracy, VGG16, Batch Size 16384, CIFAR100

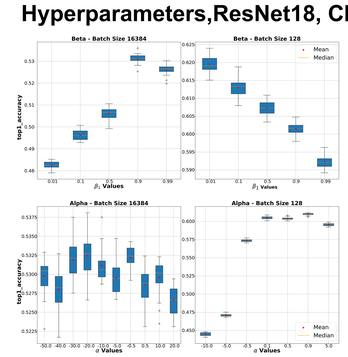


GPU Memory Overhead



$\frac{\text{Adapted Opt}}{\text{Original Opt}} \sim \mathcal{O}(\frac{1}{\text{Batch Size}})$ $\frac{\text{Adapted Opt}}{\text{Original Opt}} \sim \mathcal{O}(1)$

Hyperparameters,ResNet18, CIFAR100



The larger the batch size, the smaller the α and the greater the β_1,

$$\alpha \in \left(-\frac{1}{1-\beta_1}, \frac{1}{1-\beta_1}\right)$$

$$\beta_1 \in [0, 1)$$

Usually, use the default value:
Batch Size<1K:

$$\alpha = 0.5, \beta_1 = 0.01$$

Batch Size≥1K:

$$\alpha = -5, \beta_1 = 0.99$$