Exploring Landscapes for Better Minima along Valleys

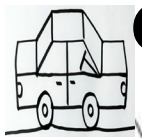
Tong Zhao, Jiacheng Li, Yuanchang Zhou, Guangming Tan, Weile Jia

Institute of Computing Technology, Chinese Academy of Sciences

November 5, 2025

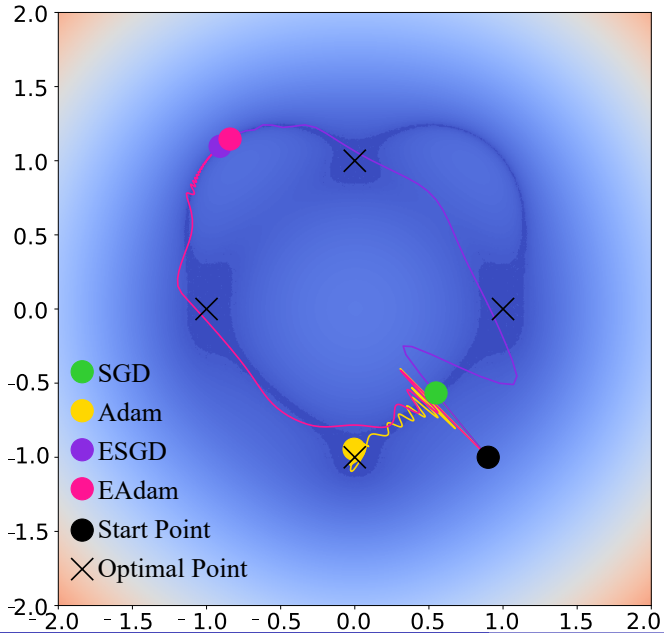
Contents

Exploitation or Exploration

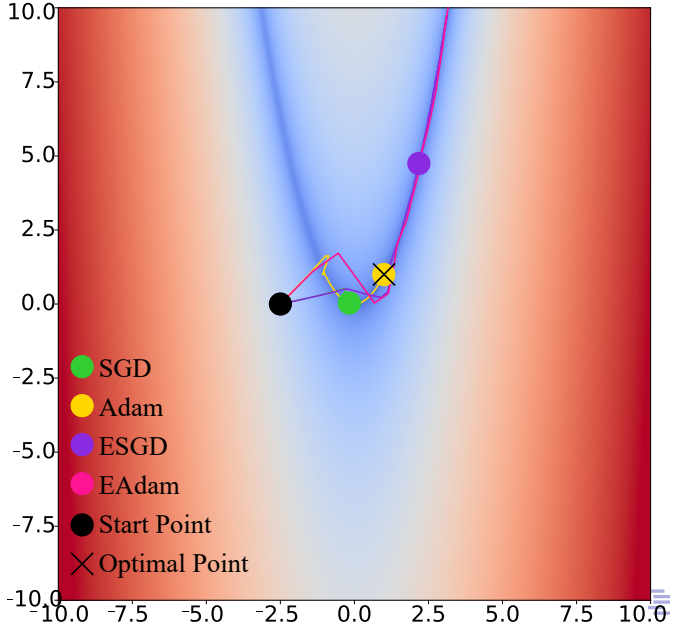


Gradient-Base Optimizer

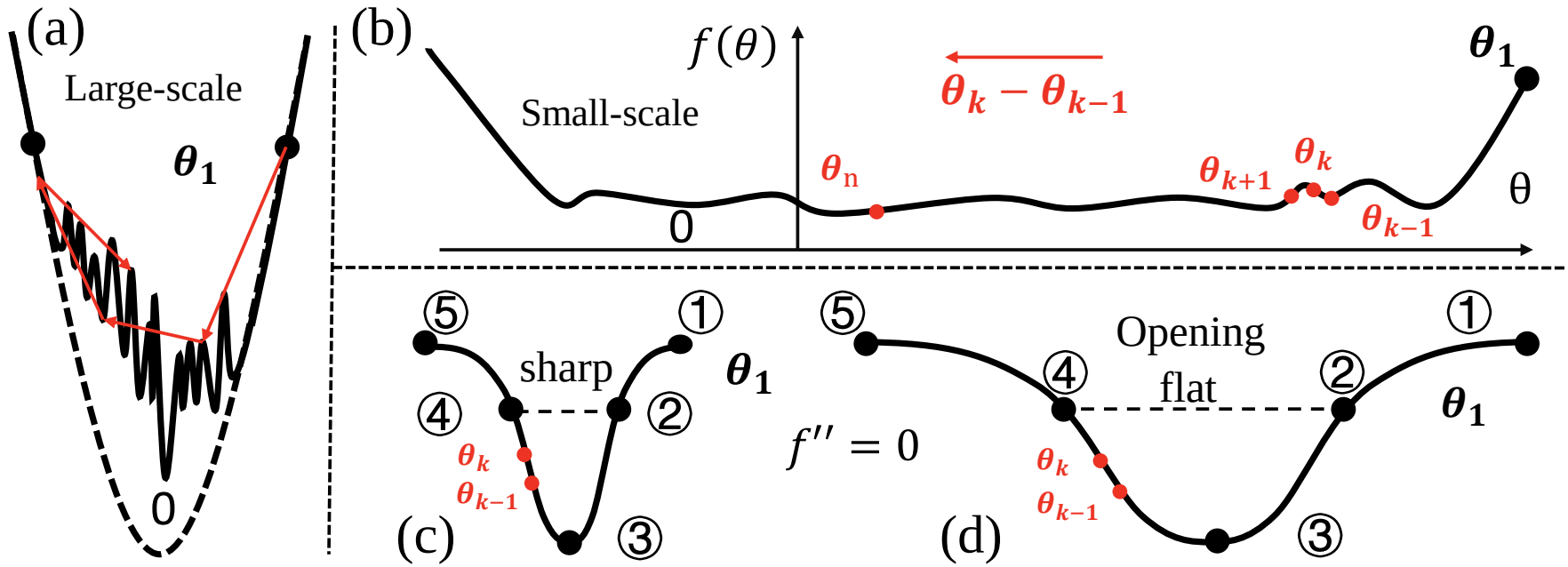
The square of the cardioid



Rosenbrock function



Large-Scale Exploitation, Small-Scale Exploration



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

$$-\overline{\mathbf{g}}_k \approx \theta_k - \theta_{k-1},$$

$$-\nabla\|\nabla f(\theta_k)\|^2 \approx \mathbf{H}_k(\theta_k - \theta_{k-1})$$

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

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

Main Idea

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

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: determining how long to explore and exploit.

Algorithm: Generic form of E-adapted optimizer

Initialize $\boldsymbol{\theta}_1$, gradient estimation operation $\phi(\cdot)$, standard origin optimzier updating operation $\psi(\cdot)$, number of iterations T, and learning rate $\eta_k > 0$ at iteration k,

for $k = 1, 2, \ldots, T$ **do**

1. Gradient estimation:

$$\mathbf{g}_k = \phi(\{\ell(\boldsymbol{\theta}_k, \boldsymbol{\zeta}_j)\}_{j \in \mathcal{Z}_k}^k),$$

where $\mathcal{Z}_k$ denotes a set of mini-batch stochastic samples used at iteration k,

2. Gradient replacement:

$$\mathbf{g}_k \leftarrow \mathbf{g}_k + \alpha \mathbf{a}_k, \text{where } \mathbf{a}_k = \beta_1 \mathbf{a}_{k-1} + (1-\beta_1)(\mathbf{g}_k - \mathbf{g}_{k-1}),$$

3. Standard parameter updating of original optimizer:

$$\boldsymbol{\theta}_t = \psi(\mathbf{g}_k, \mathbf{g}_{k-1}, \ldots, \mathbf{g}_1, \boldsymbol{\theta}_1, \eta_t).$$

end for

Algorithm: 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**

$\mathbf{g}_k = \frac{1}{|\mathcal{Z}_k|} \sum_{\boldsymbol{\zeta}_i \in \mathcal{Z}_k} \nabla \ell(\boldsymbol{\theta}_k, \boldsymbol{\zeta}_i)$;

$\mathbf{a}_k = \beta_1 \mathbf{a}_{k-1} + (1-\beta_1)(\mathbf{g}_k - \mathbf{g}_{k-1})$;

$\mathbf{m}_k = \beta_2 \mathbf{m}_{k-1} + (1-\beta_2)(\mathbf{g}_k + \alpha \mathbf{a}_k)$;

$\mathbf{v}_k = \beta_3 \mathbf{v}_{k-1} + (1-\beta_3)[\mathbf{g}_k + \alpha \mathbf{a}_k]^2$;

$\hat{\mathbf{m}}_k = \mathbf{m}_k / (1-\beta_2^k)$;

$\hat{\mathbf{v}}_k = \mathbf{v}_k / (1-\beta_3^k)$;

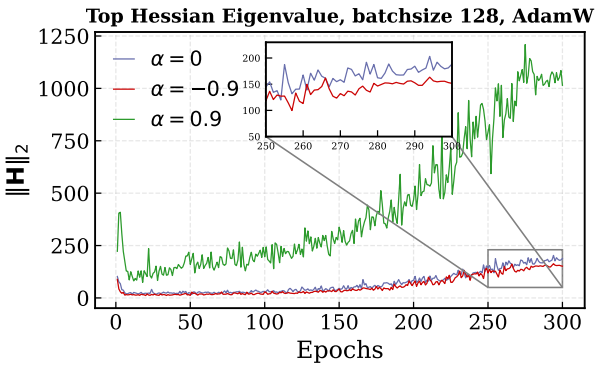
$\boldsymbol{r}_k = \hat{\mathbf{m}}_k / (\sqrt{\hat{\mathbf{v}}_k} + \varepsilon_1) + \lambda_k \boldsymbol{\theta}_k$

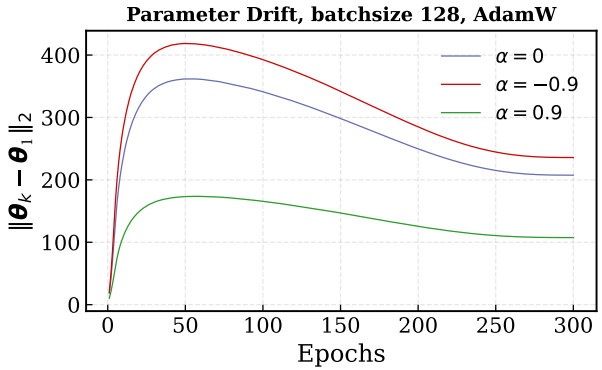
$\boldsymbol{\theta}_{k+1}^{(i)} = \boldsymbol{\theta}_k^{(i)} - \frac{\eta_k \mathbf{r}_k^{(i)} \phi(\|\boldsymbol{\theta}_k^{(i)}\|)}{(\|\mathbf{r}_k^{(i)}\| + \varepsilon_2 \phi(\|\boldsymbol{\theta}_k^{(i)}\|))}$

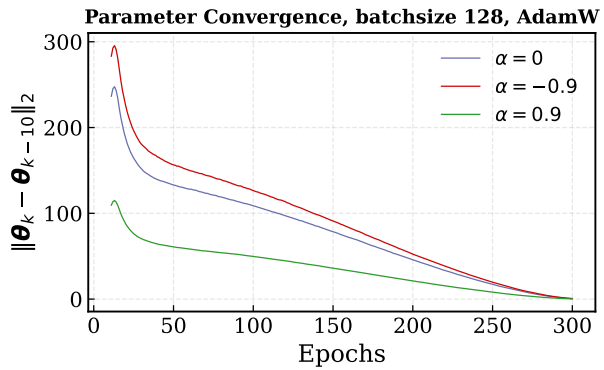
end while

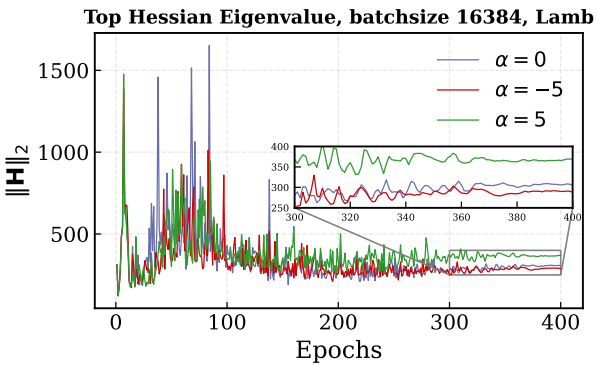
Flatness, Convergence Rate, Exploration Range

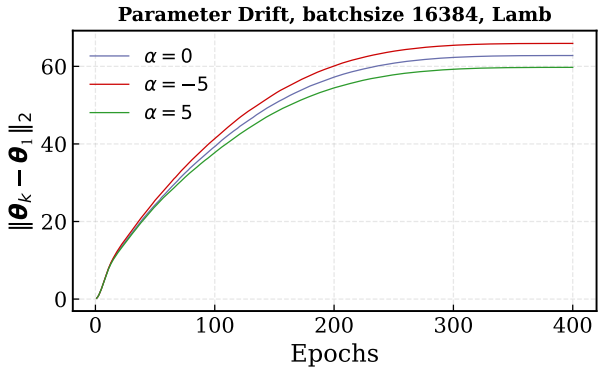
1. $\alpha < 0$ converges slowly (exploring large parameter space) but tends to flat minima.
2. $\alpha > 0$ converges fast (exploring small parameter space) but tends to sharp minima.

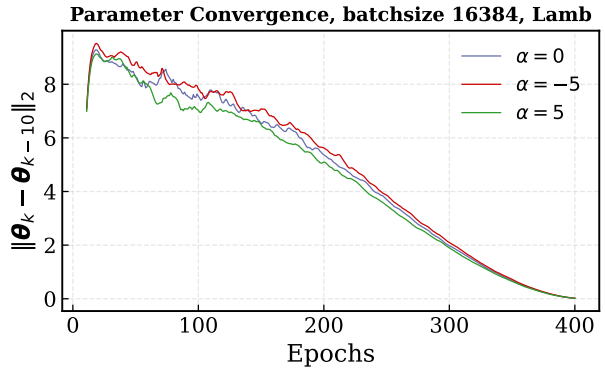












Convex and Non-Convex

Non-Convex:

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

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

Convex:

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

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

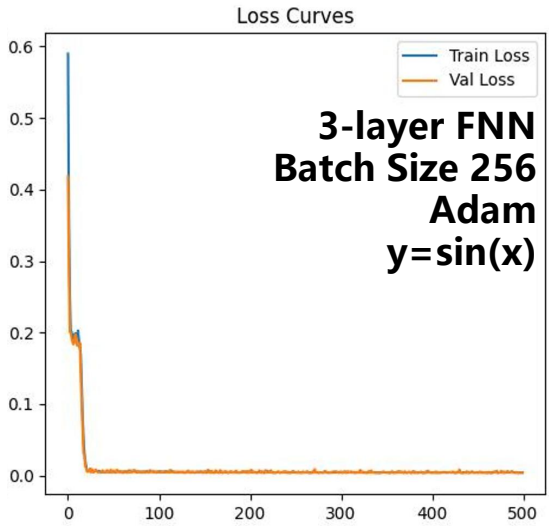
$$\frac{\sqrt{k}\left(\sqrt{\mathbf{v}_{k,i}} + \varepsilon_1\right)\left(\left\|\mathbf{r}_k^{(i)}\right\| + \varepsilon_2\phi\left(\left\|\boldsymbol{\theta}_k^{(i)}\right\|\right)\right)}{\phi\left(\left\|\boldsymbol{\theta}_k^{(i)}\right\|\right)}$$

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

Very Suitable for Large Batch Training (Exploration)

Small-Batch Training: applicable, but advantage is marginal.
Large-Batch Training: very suitable, the focus of following discussion.

1. **Why large-batch training?**
   Data parallelism. Using more GPUs to 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.



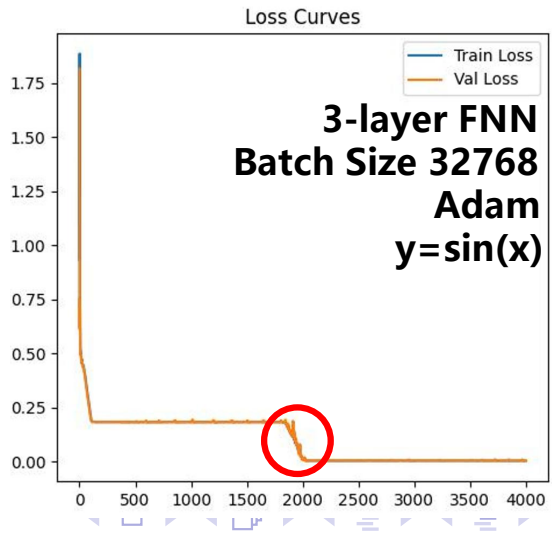


Image Classifaction

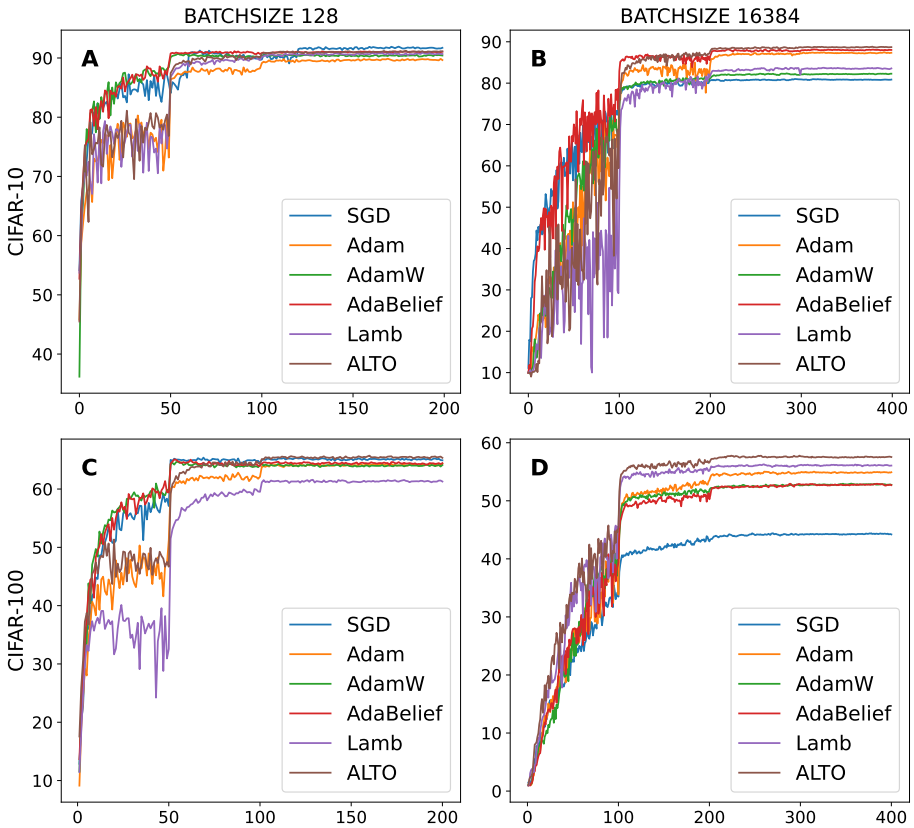
ResNet20 ResNet34

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

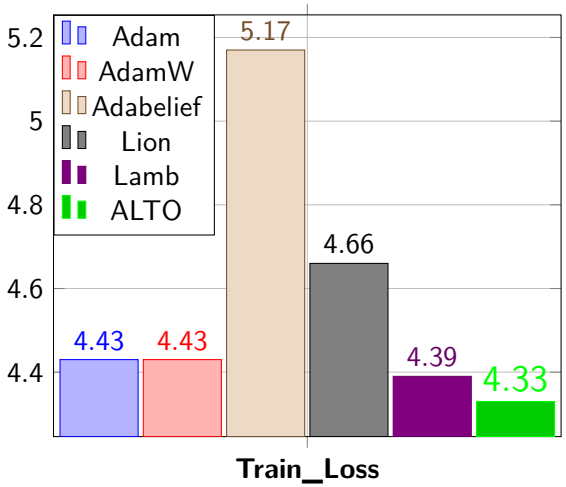


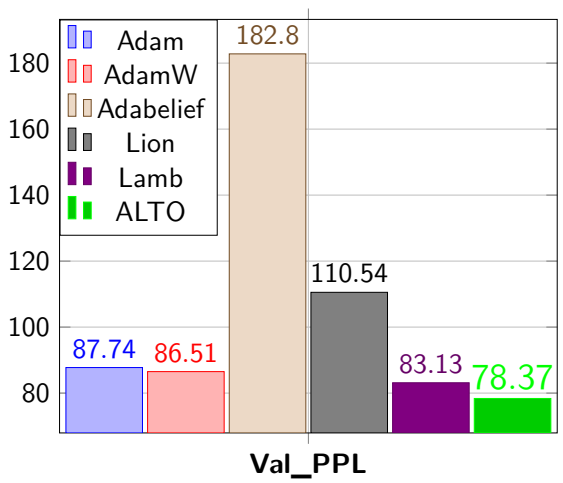
Natural Language Processing

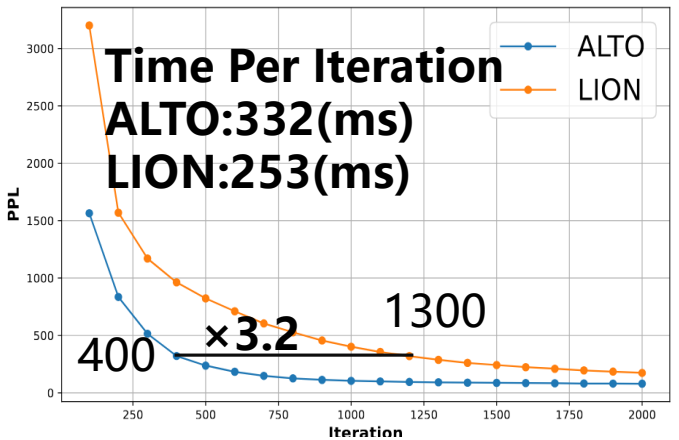
Fine-Tuning, $\text{BERT}_{\text{base}}$.

BtSz	Optimizer	CoNLL2003 (∼14K)			IMDB (25K)				MRPC (3668)				Avg
		F1	Prec.	Recall	Acc.	F1	Prec.	Recall	Acc.	F1	Prec.	Recall	F1 and Acc
32	Adam	86.97	85.70	87.15	91.77	91.52	**94.43**	88.78	84.46	88.68	85.99	91.54	88.68
	AdamW	89.32	88.41	90.25	92.42	92.43	92.24	92.62	84.63	88.70	**86.75**	90.75	89.50
	AdaBelief	89.98	89.01	90.98	92.93	92.85	93.01	91.86	83.36	87.87	85.24	90.67	89.39
	Lamb	89.86	88.82	90.93	93.13	93.09	93.56	**92.63**	84.40	88.59	86.22	91.10	89.81
	ALTO	**90.29**	**89.53**	**91.06**	**93.24**	**93.19**	93.82	92.57	**84.86**	**89.01**	86.07	**92.15**	**90.11**
1024	Adam	89.50	88.09	90.96	92.97	92.91	93.79	92.04	79.76	85.05	83.58	86.57	88.03
	AdamW	89.46	88.17	90.78	92.98	92.91	93.92	91.92	80.86	86.32	82.24	90.84	88.50
	AdaBelief	90.38	89.66	91.12	92.82	92.69	**94.46**	90.98	79.53	84.61	**84.58**	84.65	88.00
	Lamb	90.05	89.37	90.75	92.07	92.30	89.74	**95.01**	80.98	86.04	84.03	88.14	88.28
	ALTO	**90.59**	**89.95**	**91.24**	**93.10**	**93.17**	92.19	94.17	**81.39**	**86.77**	82.26	**91.80**	**89.00**

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



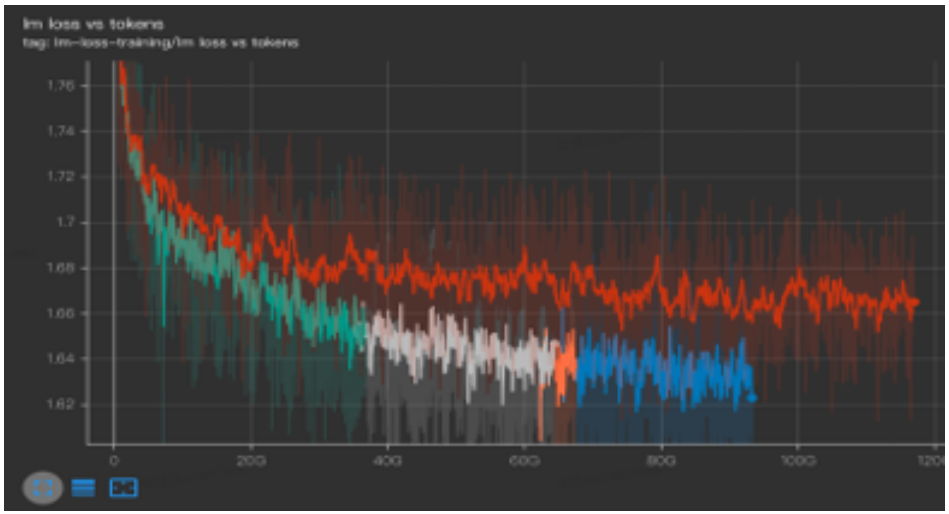




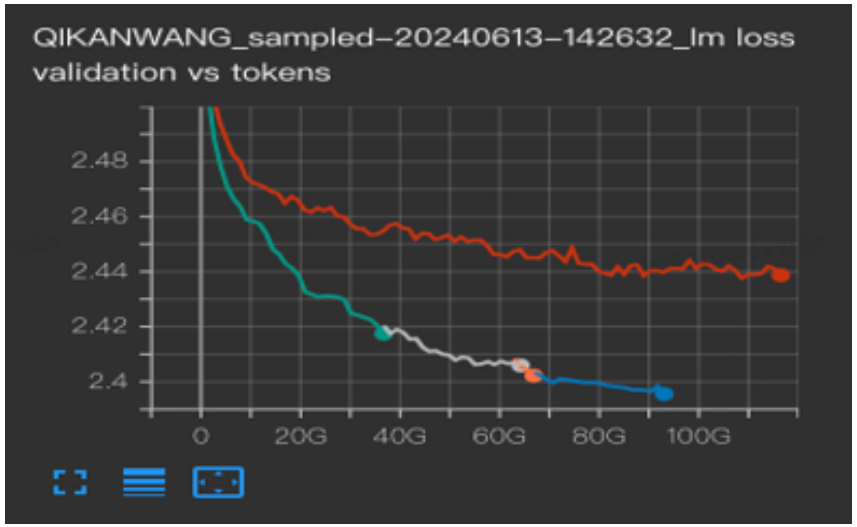
Natural Language Processing

Pre-Training, 85B-MoE-LLM,
Batch Size=1600*8192,
ALTO *v.s.* AdamW,
384 A100, Meituan Co., Ltd.

Training

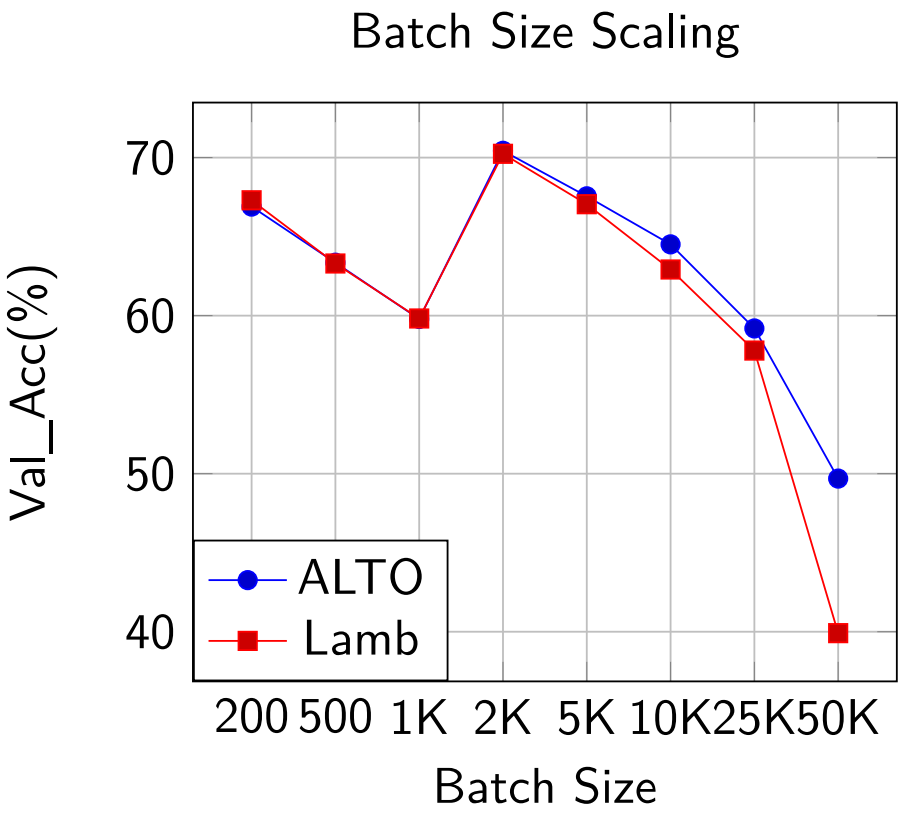


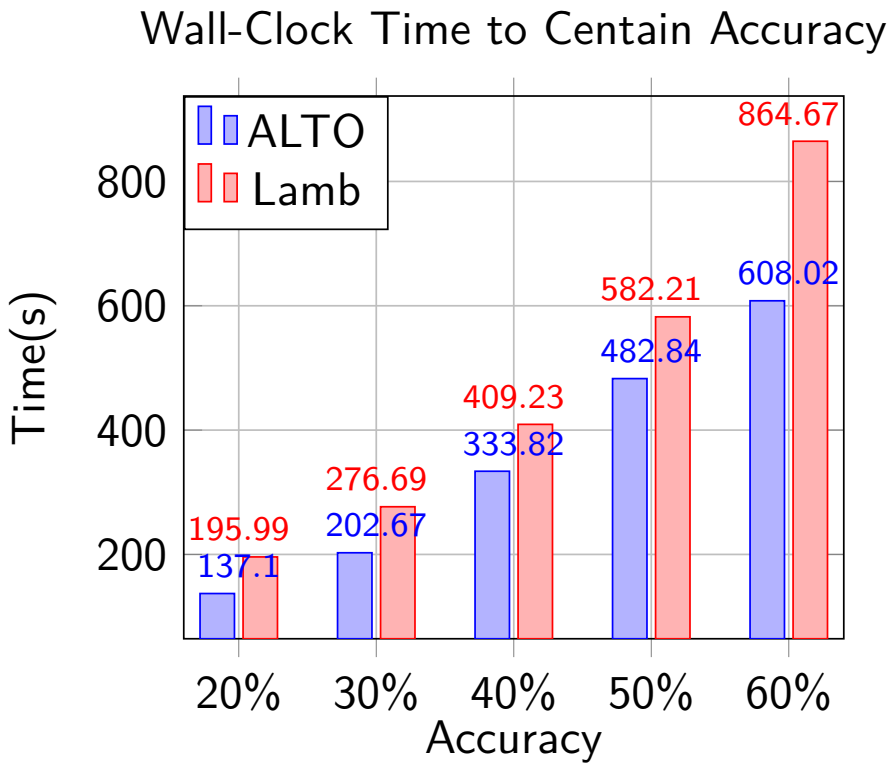
Validation



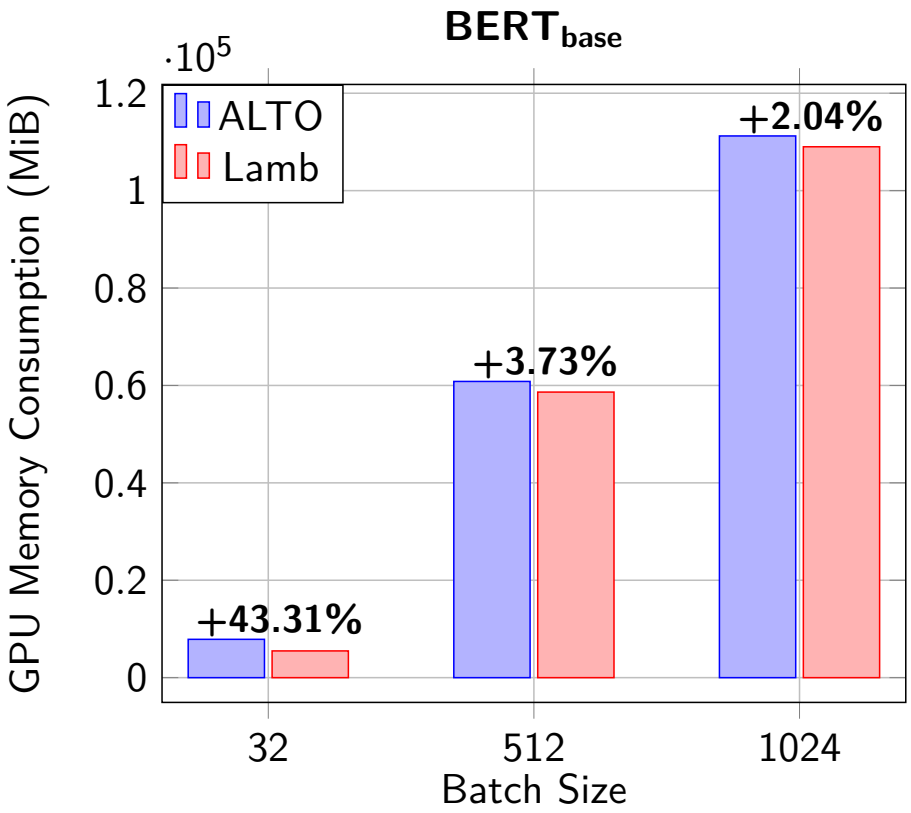
Batch Size Scaling and Wall-Clock Time

VGG16, Batch Size 16384, CIFAR100

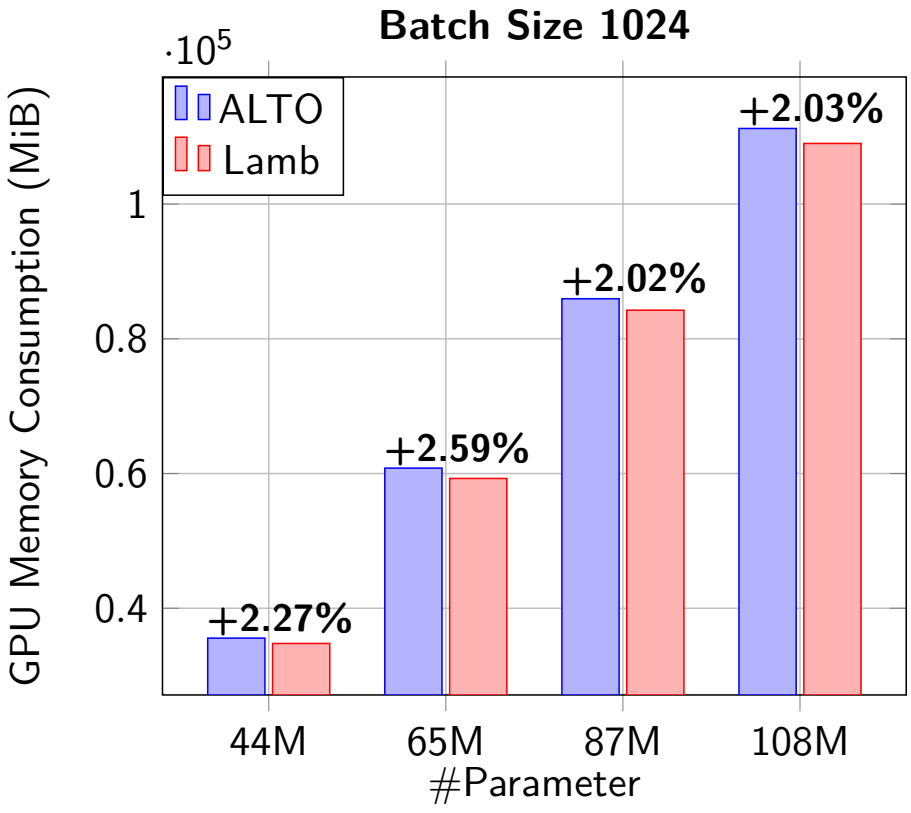




GPU Memory Overhead



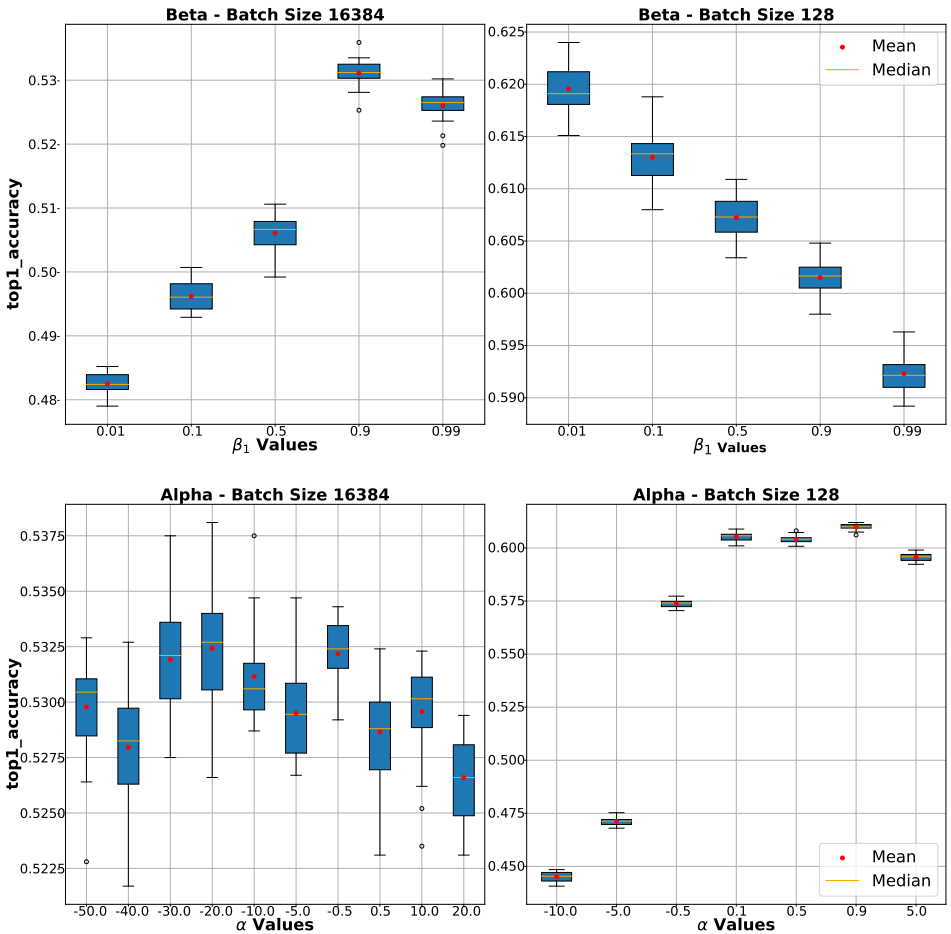
$$\frac{\text{Adapted Opt}}{\text{Original Opt}} \sim \mathcal{O}(\frac{1}{\text{Batch Size}})$$



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

Hyperparameter

ResNet18, CIFAR100



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

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

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

- Usually, use the default value:

Batch Size$<$1K:

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

Batch Size$\geq$1K:

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

Conclusion

We proposed a gradient-based optimizer adaptor E, which can

- endow optimizers with the exploration or exploitation capability.
- be applied to zero-, first-, and second-order optimizers as well as matrix-based optimizers.
- be applicable for small-batch training, but its advantage is marginal and very suitable for large-batch training, especially for LLM pretraining task (ALTO outperforms other optimizers consistently).

Paper:

Code: