

Provable Stochastic Optimization for Global Contrastive Learning: Small Batch Does Not Harm Performance

Zhuoning Yuan¹, Yuexin Wu², Zi-Hao Qiu³, Xianzhi Du², Lijun Zhang³,
Denny Zhou², Tianbao Yang¹

¹University of Iowa, ²Google Research, ³Nanjing University



ICML
International Conference
On Machine Learning



Self-Supervised Learning

- Learning representations on large amount of unlabeled data (e.g., image, text)
 - **Contrastive learning** – SimCLR (Google), CLIP (OpenAI)
- Challenges of contrastive learning, such as
 - **Large batch size** – high computation resource demand
- In this work, we explain **why** SimCLR is sensitive to batch size and **how** to tackle this challenge by proposing a new framework

SimCLR: Mini-batch Contrastive Objective

- SimCLR defines contrastive objective for each augmented pair $(\mathcal{A}(\mathbf{x}_i), \mathcal{A}'(\mathbf{x}_i))$ for an image \mathbf{x}_i in **mini-batch level** B :

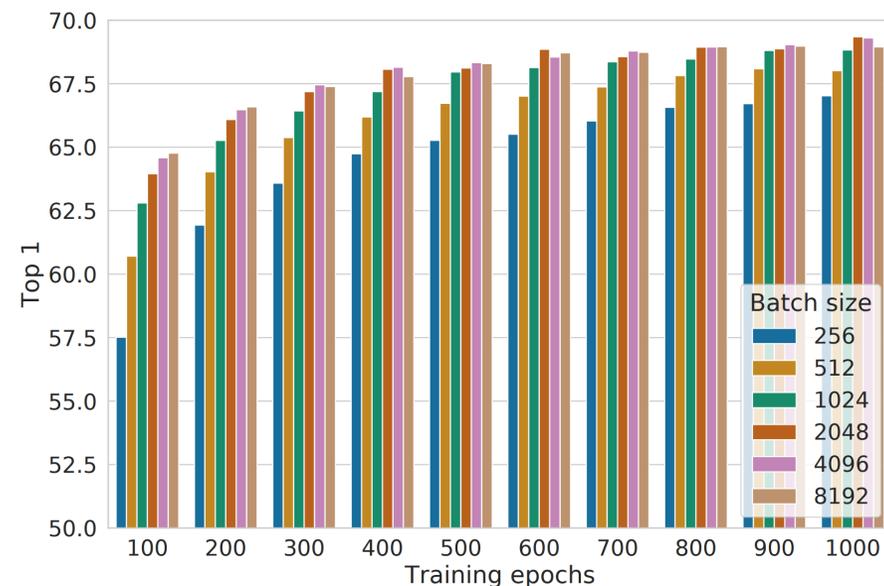
$$L_B(\mathbf{w}; \mathbf{x}_i, \mathcal{A}, \mathcal{A}') = -\ln \frac{\exp(E(\mathcal{A}(\mathbf{x}_i))^\top E(\mathcal{A}'(\mathbf{x}_i))/\tau)}{g(\mathbf{w}; \mathbf{x}_i, \mathcal{A}, \mathcal{B})},$$

Positive Pair

$$g(\mathbf{w}; \mathbf{x}_i, \mathcal{A}, \mathcal{B}_i) = \sum_{\mathbf{z}_j \in \mathcal{B}_i} \exp(E(\mathcal{A}(\mathbf{x}_i))^\top E(\mathbf{z}_j)/\tau)$$

Negative Pairs

Mini-batch



Why Large batch sizes work better?

Global Contrastive Objective

- To explain of issues of InfoNCE loss, we use **Global Contrastive Objective** for each augmented pair $(\mathcal{A}(\mathbf{x}_i), \mathcal{A}'(\mathbf{x}_i))$ of an image \mathbf{x}_i in all data S :

Mini-batch Contrastive Objective

$$L_{\mathcal{B}}(\mathbf{w}; \mathbf{x}_i, \mathcal{A}, \mathcal{A}') = -\ln \frac{\exp(E(\mathcal{A}(\mathbf{x}_i))^{\top} E(\mathcal{A}'(\mathbf{x}_i))/\tau)}{g(\mathbf{w}; \mathbf{x}_i, \mathcal{A}, \mathcal{B})},$$

$$g(\mathbf{w}; \mathbf{x}_i, \mathcal{A}, \mathcal{B}_i) = \sum_{\mathbf{z}_j \in \mathcal{B}_i} (\exp(E(\mathcal{A}(\mathbf{x}_i))^{\top} E(\mathbf{z}_j)/\tau))$$

Negative Pairs
(mini-batch)

Global Contrastive Objective

$$L(\mathbf{w}; \mathbf{x}_i, \mathcal{A}, \mathcal{A}') = -\ln \frac{\exp(E(\mathcal{A}(\mathbf{x}_i))^{\top} E(\mathcal{A}'(\mathbf{x}_i))/\tau)}{\epsilon' + g(\mathbf{w}; \mathbf{x}_i, \mathcal{A}, \mathcal{S}_i)},$$

$$g(\mathbf{w}; \mathbf{x}_i, \mathcal{A}, \mathcal{S}_i) = \sum_{\mathbf{z} \in \mathcal{S}_i} (\exp(E(\mathcal{A}(\mathbf{x}_i))^{\top} E(\mathbf{z})/\tau)),$$

Negative Pairs
(All data)

Analysis of Optimization Error for SimCLR

Theorem 1. Assume \mathbf{F} is smooth, \mathbf{g} is smooth and Lipschitz continuous, SimCLR ensures that $\mathbb{E} \left[\|\nabla F(\mathbf{w}_{t'})\|^2 \right] \leq O \left(\frac{1}{\eta T} + \eta + \frac{1}{B} \right)$ for a random $t' \in \{1, \dots, T\}$.

- **Remarks:**

- SimCLR suffers a large optimization error depending on the batch size in the order of $O(1/\sqrt{B})$ for the gradient norm term,
- This also explains why SimCLR achieves good performance using large batch size B

SogCLR

- The key idea is to maintain a moving average on $g(\mathbf{w}; \mathbf{x}_i, \mathcal{A}, S_i)$ for each original image \mathbf{x}_i

$$\min_{\mathbf{w}} F(\mathbf{w}) = \mathbb{E}_{\mathbf{x}_i \sim \mathcal{D}, \mathcal{A}, \mathcal{A}' \sim \mathcal{P}} [\tau L(\mathbf{w}; \mathbf{x}_i, \mathcal{A}, \mathcal{A}')]$$

Mini-batch Gradient



$$(1) \tau \nabla L(\mathbf{w}; \mathbf{x}_i, \mathcal{A}, \mathcal{A}') = -\nabla (E(\mathcal{A}(\mathbf{x}_i))^{\top} E(\mathcal{A}'(\mathbf{x}_i)))$$

$$+ \frac{\tau}{\epsilon' + g(\mathbf{w}; \mathbf{x}_i, \mathcal{A}, S_i)} \nabla g(\mathbf{w}; \mathbf{x}_i, \mathcal{A}, S_i).$$

Non-linear function of $g(\cdot)$!

Mini-batch Gradient



$$(2) \mathbf{u}_{i,t} = (1 - \gamma) \mathbf{u}_{i,t-1} + \gamma \frac{1}{2|\mathcal{B}_i|} (g(\mathbf{w}_t; \mathbf{x}_i, \mathcal{A}, \mathcal{B}_i) + g(\mathbf{w}_t; \mathbf{x}_i, \mathcal{A}', \mathcal{B}_i))$$

maintain a scalar \mathbf{u}_i for each image to track $g(\cdot)$ by using a mini-batch \mathcal{B}_i

Using mini-batch doesn't work !

Analysis of Optimization Error for SogCLR

Theorem 2&3 Assume $\mathbb{E}_{\mathcal{A}, \mathcal{A}', z} \left| E(\mathcal{A}(\mathbf{x}_i))^\top E(\mathbf{z}) - E(\mathcal{A}'(\mathbf{x}_i))^\top E(\mathbf{z}) \right|^2 \leq \epsilon^2$, after T iterations, SogCLR ensures $\mathbb{E}[\|\nabla F_{v1}(\mathbf{w}_{t'})\|^2] \leq O\left(\frac{1}{\sqrt{BT}} + \frac{\sqrt{n}}{B\sqrt{T}} + \epsilon^2\right)$ and $\mathbb{E}[\|\nabla F_{v2}(\mathbf{w}_{t'})\|^2] \leq O\left(\frac{1}{\sqrt{BT}} + \frac{\sqrt{n}}{B\sqrt{T}}\right)$ for a random $t' \in \{1, \dots, T\}$.

- **Remarks:**

- V1: When ϵ is small enough, the optimization error of SogCLRv1 is negligible
- V2: SogCLRv2 can converges to a stationary solution, i.e., zero optimization error

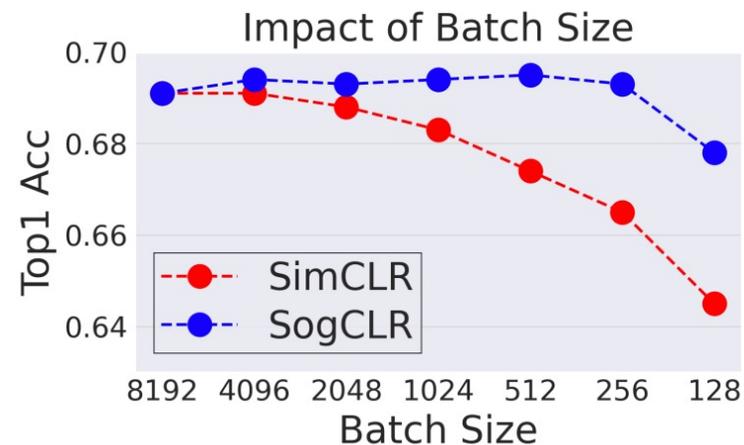
Experiments

- Experiments of image pretraining task on ImageNet-1K

Table 3. Linear evaluation (top-1 accuracy) under different batch sizes and training epoch on ResNet-50 and ImageNet-1K.

Method	BatchSize\Epoch	100	200	400	800
SimCLR	128	62.6	64.0	64.1	64.5
SogCLR	128	64.9	66.2	67.4	67.9
SimCLR	256	62.8	64.3	65.7	66.5
SogCLR	256	65.2	67.1	68.7	69.4
SimCLR	512	63.8	65.6	66.7	67.4
SogCLR	512	65.0	67.2	68.8	69.6

Linear evaluation with small batch sizes for different epochs.



Linear evaluation by varying batch sizes from 128 to 8192 on ImageNet-1K.

Experiments

- Comparison of different InfoNCE-based contrastive learning methods

Method	Batch Size	Memory Bank	Momentum Encoder	Other Tricks	Convergence	Top1 Acc.
SimCLR (Chen et al., 2020a)	Large-batch	No	No	Strong Aug.	No	66.5
NNCLR (Dwibedi et al., 2021)	Large-batch	No	No	Nearest Neighbors	No	68.7
SiMo (Zhu et al., 2020a)	Small-batch	No	Yes	Margin Trick	No	72.1
MoCov2 (Chen et al., 2020c)	Small-batch	Yes	Yes	Strong Aug.	No	71.1
InfoMin (Tian et al., 2020)	Small-batch	Yes	Yes	InfoMin Aug.	No	73.0
SogCLR (Ours)	Small-batch	No	No	GC Optimization	Yes	72.5

Notes: Linear evaluation accuracy by using 800 epochs, a batch size of 256, and ResNet-50 on ImageNet-1K. Momentum Bank/Encoder is introduced by MoCov1 (He et al. 2019).

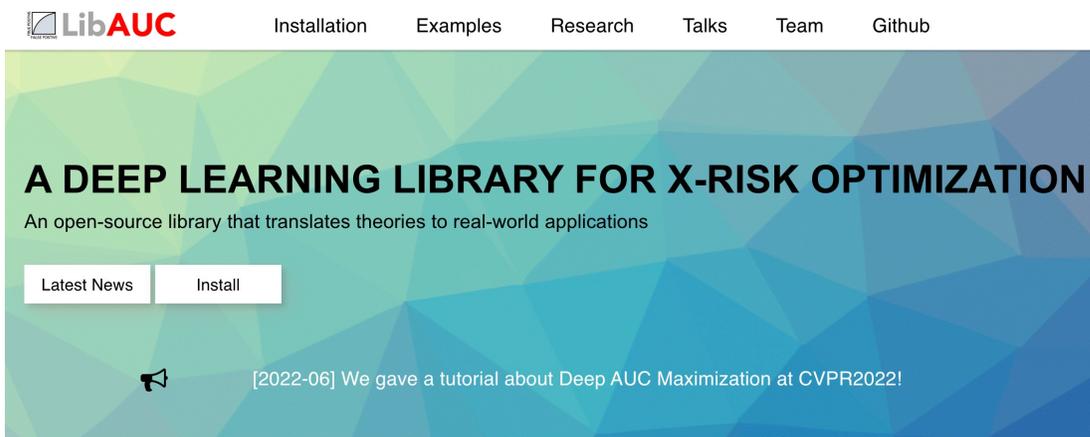
Implementations

- Code is integrated with **LibAUC**: <https://github.com/Optimization-AI/SogCLR>



LibAUC

<https://libauc.org>



The screenshot shows the LibAUC website homepage. At the top, there is a navigation bar with links for Installation, Examples, Research, Talks, Team, and Github. The main header features the LibAUC logo and the title "A DEEP LEARNING LIBRARY FOR X-RISK OPTIMIZATION". Below the title is a subtitle: "An open-source library that translates theories to real-world applications". There are two buttons: "Latest News" and "Install". A notification bell icon is present, with a message: "[2022-06] We gave a tutorial about Deep AUC Maximization at CVPR2022!".

KEY FEATURES & CAPABILITIES

Easy Installation

Easy to install and insert LibAUC code into existing training pipeline with Deep Learning frameworks like PyTorch.



Broad Applications

Users can learn different neural network structures (e.g., linear, MLP, CNN, GNN, transformer, etc) that support their data types.



Efficient Algorithms

Stochastic algorithms with provable theoretical convergence that support learning with millions of data points without a large batch size.



Hands-on Tutorials

Hands-on tutorials are provided for optimizing a variety of measures and objectives belonging to the family of X-risks.



Thank You!

