# X-risk Optimization: A New Paradigm for Deep Learning

Tianbao Yang

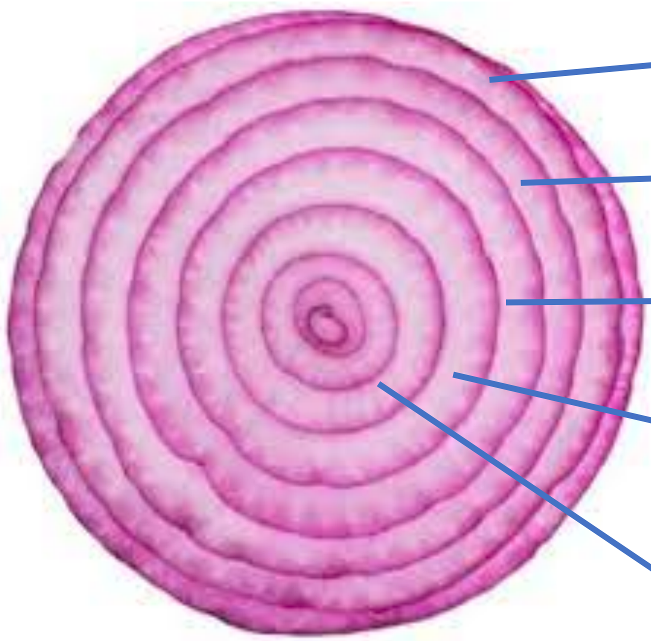
Texas A&M University

# Outline

- Overview & Background
- Three Use Cases

# My Research Focus

**AI is like an Onion**



**D**omains (e.g., medicine)

**A**pplications (e.g., drug discovery)

**R**epresentations (data, models)

**F**ormulations (objectives)

**O**ptimization (algorithms)

## Advancing Optimization to Make ML/AI Faster and Better

**Training Faster**

**Testing better**

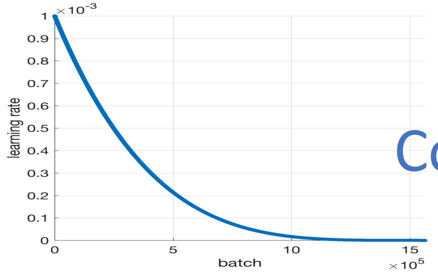
# Optimization for Machine Learning

$$\min_{\mathbf{w}} F(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} \ell(\mathbf{w}, \mathbf{z}_i)$$

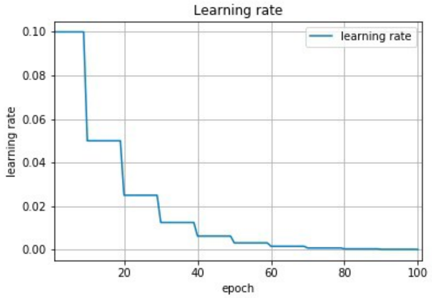
Empirical Risk Minimization (ERM)

# SGD: Stochastic Gradient Descent

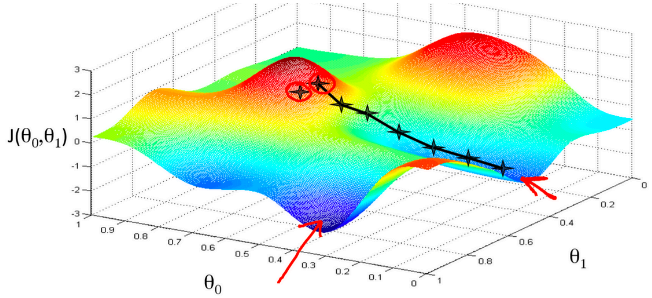
$$\mathbf{w}_{t+1} = \mathbf{w}_t - \boxed{\eta_t}\nabla \ell(\mathbf{w}_t, \mathbf{z}_t)$$




Conventional: Polynomially Decreasing




Modern: Stagewise




Modern: Adaptive

# Momentum and Adaptive Methods

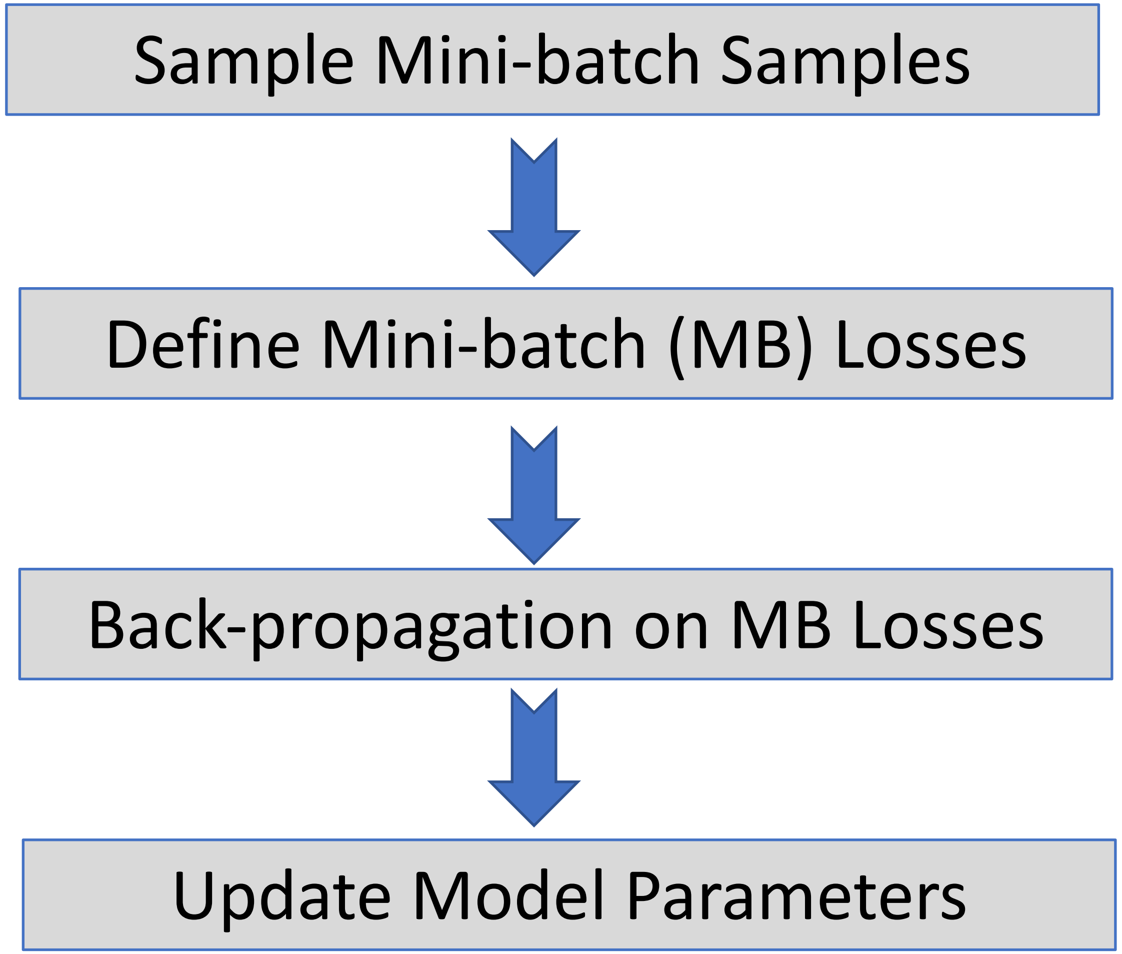
| | | |
|---|---|---|
| Imagenet classification with deep convolutional neural networks<br>A Krizhevsky, I Sutskever, GE Hinton<br>Advances in neural information processing systems 25, 1097-1105 | 99188 | 2012 |
| Stochastic Heavy-ball Method (SHB) | | |
| On the importance of initialization and momentum in deep learning<br>I Sutskever, J Martens, G Dahl, G Hinton<br>International conference on machine learning, 1139-1147 | 4069 | 2013 |
| Stochastic Nesterov's Accelerated Gradient (SNAG) | | |
| Adam: A method for stochastic optimization<br>D Kingma, J Ba<br>International Conference on Learning Representations | 92479 | 2015 |
| Adam | | |

Momentum term

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \boxed{\eta_t} \nabla \ell(\mathbf{w}_t, \mathbf{z}_t) + \boxed{\delta_t}$$

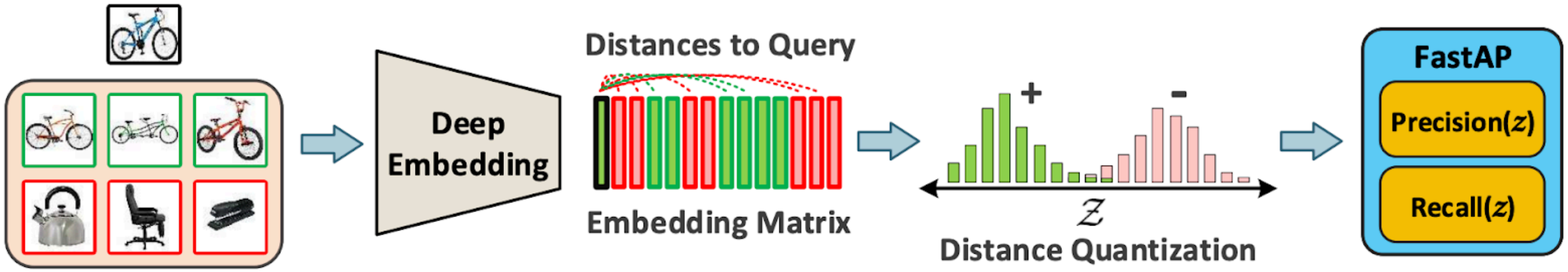
Adaptive or Stagewise

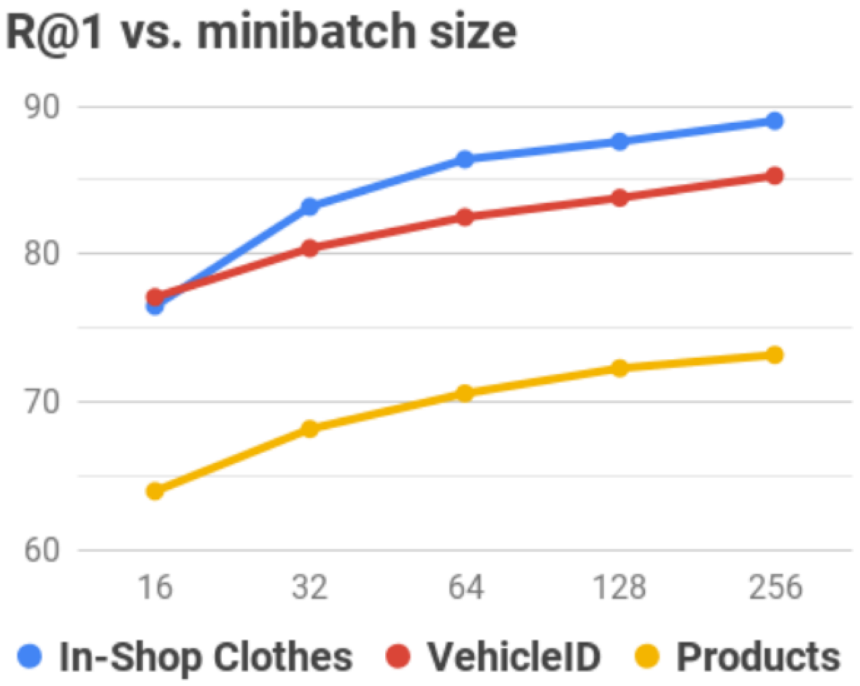
# A Standard Learning Paradigm



# Some Undesirable Consequences

Cakir et al. Deep metric learning to rank. In CVPR, 2019.

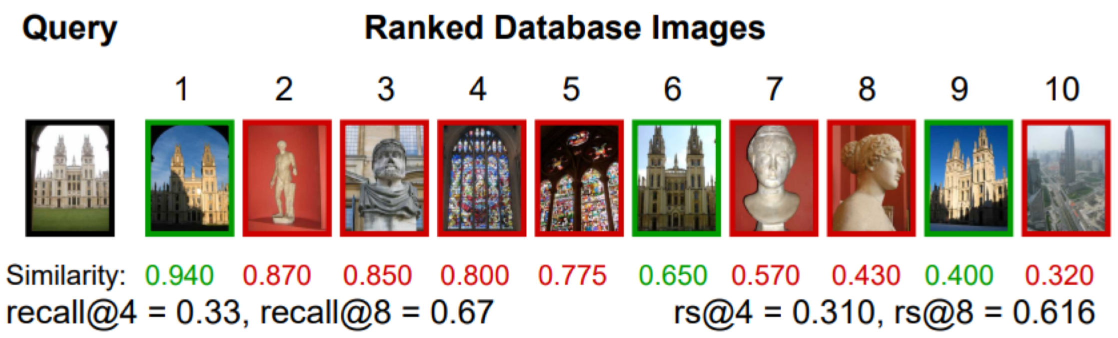


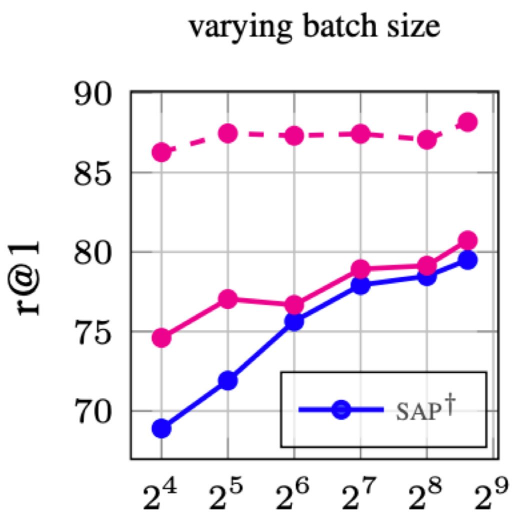




"As provided in Figure 4a, R@1 monotonically improves with larger batch size on all three datasets. This observation resonates with the fact that large batches reduce the variance of the stochastic gradients, which has been shown to be beneficial [32]. On the other hand, from the learn-"

# Some Undesirable Consequences

Patel et al. Recall@k Surrogate Loss with Large Batches and Similarity Mixup. In CVPR, 2022.



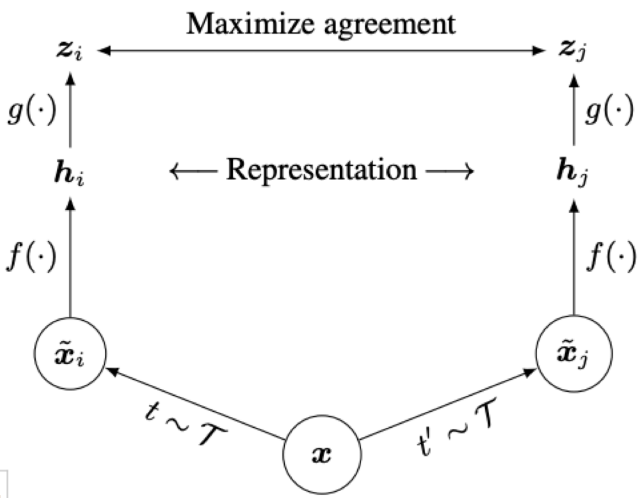

“

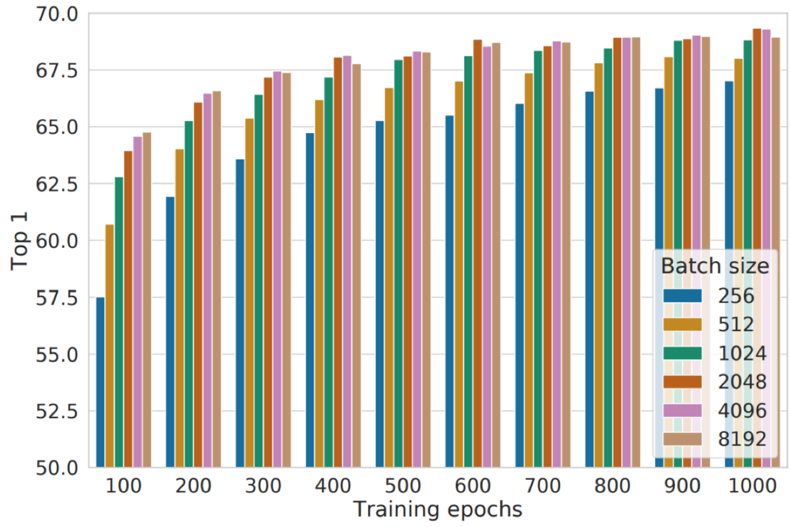
**Batch size.** The effect of the varying batch size is shown in Figure 4 (right). It demonstrates that large batch size leads to better results. A significant performance boost is

”

# Some Undesirable Consequences

Chen et al. A Simple Framework for Contrastive Learning of Visual Representations. In ICML, 2020.




“ **5.2. Contrastive learning benefits (more) from larger batch sizes and longer training**

Figure 9 shows the impact of batch size when models are trained for different numbers of epochs. We find that, when the number of training epochs is small (e.g. 100 epochs), larger batch sizes have a significant advantage over the smaller ones. With more training steps/epochs, the gaps ”

# Conventionally Small Batch is Fine

$$\min_{\mathbf{w}} F(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} \ell(\mathbf{w}, \mathbf{z}_i)$$

"

The stochastic gradient descent (SGD) method and its variants are algorithms of choice for many Deep Learning tasks. These methods operate in a small-batch regime wherein a fraction of the training data, say 32–512 data points, is sampled to compute an approximation to the gradient. It has been observed in practice that when using a larger batch there is a degradation in the quality of the model, as

"

Keskar et al. ON LARGE-BATCH TRAINING FOR DEEP LEARNING: GENERALIZATION GAP AND SHARP MINIMA. ICLR 2017.

# A Standard Learning Paradigm

Sample Mini-batch Samples

Define Mini-batch (MB) Losses

Back-propagation on MB Losses

Update Model Parameters

Q: What is Wrong about this Learning Paradigm?

A: ERM is **NOT** enough

# Beyond ERM: Deep X-risk Optimization

# X-risk

**Definition**

A family of **Compositional** measures in which the loss function of each data point is defined in a way that **Contrasts** the data point with a **Large number of items**.

$$F(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} f_i(g(\mathbf{w}, \mathbf{z}_i, \mathcal{S}_i))$$

A Large Set

# Challenges of Optimizing X-risk

$$F(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} f_i(g(\mathbf{w}, \mathbf{z}_i, \mathcal{S}_i))$$

**Full Gradient**
for each data

$$\nabla f_i(g(\mathbf{w}, \mathbf{z}_i, \mathcal{S}_i))\nabla g(\mathbf{w}, \mathbf{z}_i, \mathcal{S}_i)$$

$$\mathbb{E} \neq$$

**Mini-batch Gradient**

$$\boxed{\nabla f_i(g(\mathbf{w}, \mathbf{z}_i, \mathcal{B}_i))\nabla g(\mathbf{w}, \mathbf{z}_i, \mathcal{B}_i)}$$

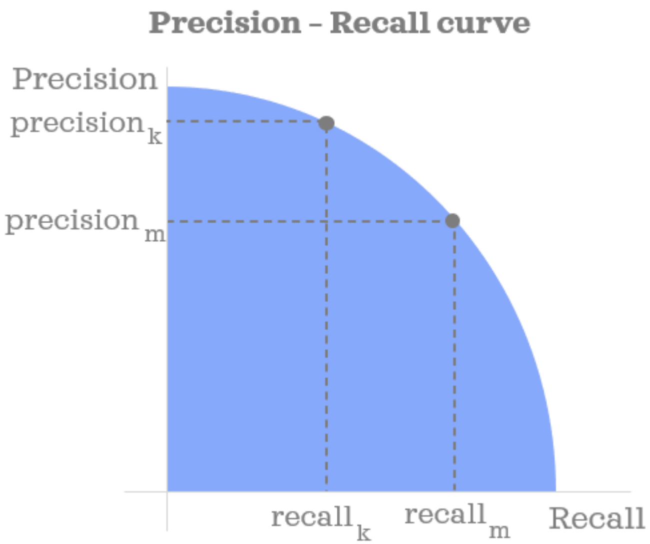
Biased

Mini-batch

# Outline

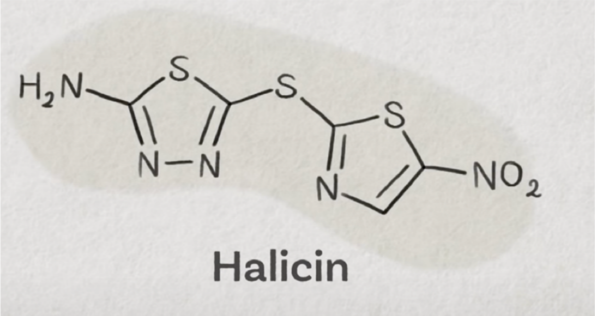
- Three Use Cases
    - AUPRC/AP Maximization
    - Top-K NDCG Maximization
    - Self-supervised Learning

# Deep AUPRC/AP Maximization



MIT AICures Challenge

**Fighting Secondary Effects of Covid**




Stokes et al. 2020. Cell.

# Evaluation Metric: **AUPRC**

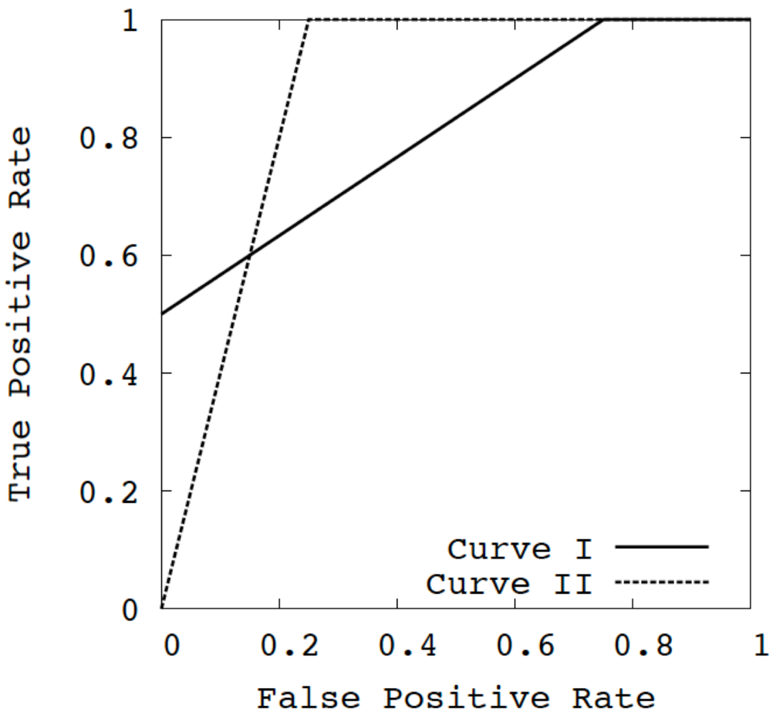
(a) Test PRC-AUC

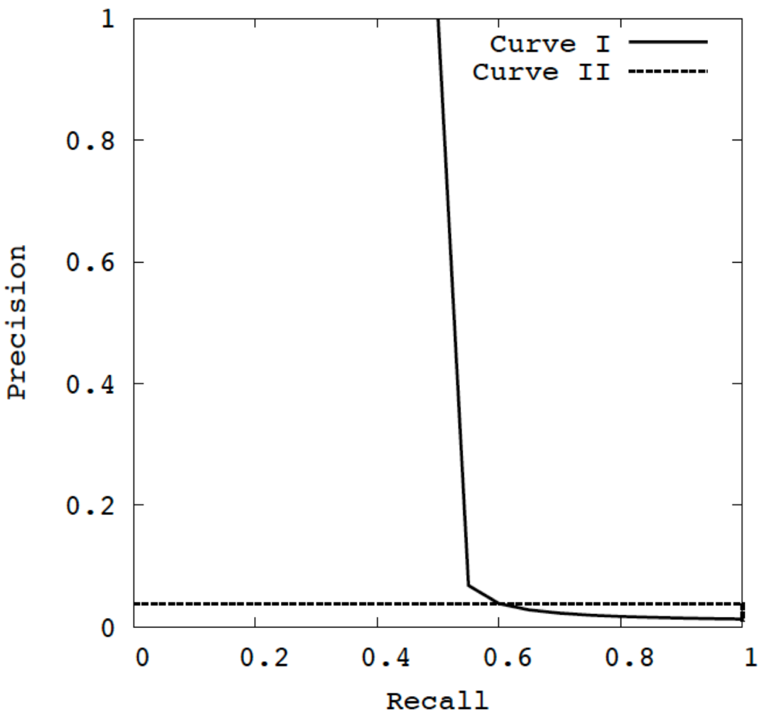
| Rank | Model | Author | Submissions | Test PRC-AUC |
|---|---|---|---|---|
| 1 | MolecularG | AIDrug@PA | 7 | 0.725 |
| 2 | - | AGL Team | 20 | 0.702 |
| 3 | MoleculeKit | DIVE@TAMU | 7 | 0.677 |
| 4 | GB | BI | 6 | 0.67 |
| 5 | Chemprop ++ | AICures@MIT | 4 | 0.662 |
| 6 | - | Mingjun Liu | 3 | 0.657 |
| 7 | Pre-trained OGB-GIN (ensemble) | Weihua Hu@Stanford | 2 | 0.651 |
| 8 | RF + fingerprint | Cyrus Maher@Vir Bio | 1 | 0.649 |
| 9 | Graph Self-supervised Learning | SJTU_NRC_Mila | 3 | 0.622 |
| 10 | - | Congjie He | 10 | 0.611 |

(b) Test ROC-AUC

| Rank | Model | Author | Submissions | Test ROC-AUC |
|---|---|---|---|---|
| 1 | MoleculeKit | DIVE@TAMU | 7 | 0.928 |
| 2 | Chemprop ++ | AICures@MIT | 4 | 0.877 |
| 3 | - | Gianluca Bontempi | 7 | 0.848 |
| 4 | - | Apoorv Umang | 1 | 0.84 |
| 5 | Pre-trained OGB-GIN (ensemble) | Weihua Hu@Stanford | 2 | 0.837 |
| 6 | - | Kexin Huang | 1 | 0.824 |
| 7 | Chemprop | Rajat Gupta | 7 | 0.818 |
| 8 | MLP | IITM | 7 | 0.807 |
| 9 | Graph Self-supervised Learning | SJTU_NRC_Mila | 3 | 0.8 |
| 10 | - | Congjie He | 10 | 0.8 |

# Why AUROC Max. is NOT Enough?




**Challenge:** Highly Imbalanced Data

# Non-Parametric Estimator: Average Precision

$$\mathrm{AP}(h) = \frac{1}{n_+} \sum_{\mathbf{x}_i \in \mathcal{S}_+} \mathrm{Precision}(h(\mathbf{x}_i))$$

Positive Examples

$$\mathrm{Precision}(h(\mathbf{x}_i)) = \frac{\sum_{\mathbf{x}_j \in \mathcal{S}_+} \mathbb{I}(h(\mathbf{x}_j) \geq h(\mathbf{x}_i))}{\sum_{\mathbf{x}_j \in \mathcal{S}} \mathbb{I}(h(\mathbf{x}_j) \geq h(\mathbf{x}_i))}$$

All Examples

# Deep AUPRC Maximization

**Limitations** of Literature on AUPRC Maximization

(1) Not applicable to deep learning (e.g., SVM-AP, Yue et al.)

(2) No Convergence, require large batch (e.g., FastAP, Cakir et al.)

**Our Contributions:**

**(1)** **N**ew Formulation based on Compositional Opt.

**(2)** **F**irst Algorithms with Convergence Theory

**(3)** **P**ractical Algorithms and **I**mproved Theory

(NeurIPS'21, AISTATS'22, ICML'22)

# Our Formulation

(NeurIPS 2021)

Precision
$$\frac{\boxed{\sum_{\mathbf{x}_j \in \mathcal{S}_+} \ell(h_{\mathbf{w}}(\mathbf{x}_j) - h_{\mathbf{w}}(\mathbf{x}_i))}}{\boxed{\sum_{\mathbf{x}_j \in \mathcal{S}} \ell(h_{\mathbf{w}}(\mathbf{x}_j) - h_{\mathbf{w}}(\mathbf{x}_i))}} \quad \begin{array}{l} \rightarrow [g_i(\mathbf{w})]_1 \\ \rightarrow [g_i(\mathbf{w})]_2 \end{array}$$

**Limitations of Existing Methods**

- Not Convergent (e.g., SGD/Adam)
- Not-scalable (e.g., NASA, Ghadimi et al.)
- Require Large batch size (e.g., BSGD, Hu et al.)

$$f(g) = -\frac{[g]_1}{[g]_2}$$

$$\min_{\mathbf{w}} F(\mathbf{w}) = \frac{1}{n_+} \sum_{\mathbf{x}_i \in \mathcal{S}_+} \boxed{f(g_i(\mathbf{w}))}$$



**Finite-sum Coupled Compositional Optimization**

# Key Idea of SOAP

Full Gradient $\nabla f(g_i(\mathbf{w}_t))$ at t^th iteration

Naïve Mini-batch $\nabla f(\hat{g}_i(\mathbf{w}_t))$ — **Unbiased** ✗

Vs.

Variance-reduced $\nabla f(u_i^t)$ — **Biased but variance-reduced** ✓

$$u_i^t = (1-\beta)u_i^{t-1} + \beta \hat{g}_i(\mathbf{w}_t) \qquad \mathbf{x}_i \in \mathcal{B}_+$$

**Sampled Positive**

# Theories

Goal $\|\nabla F(\mathbf{w})\| \leq \epsilon$

---

NeurIPS'21

First Algorithm with Convergence Guarantee

SGD-style Update $O\left(\frac{1}{\epsilon^5}\right)$

---

ICML'22, AISTATS'22

Improved Convergence

Momentum or Adam-style Update $O\left(\frac{1}{\epsilon^4}\right)$

3.5% Positive 2 ~3% Improvement

| Dataset | Method | GINE | MPNN | ML-MPNN |
|---|---|---|---|---|
| HIV | CE | 0.2774 (± 0.0101) | 0.3197 (± 0.0050) | 0.2988 (± 0.0076) |
| | CB-CE | 0.3082 (± 0.0101) | 0.3056 (± 0.0018) | 0.3291 (± 0.0189) |
| | Focal | 0.3236 (± 0.0078) | 0.3136 (± 0.0197) | 0.3279 (± 0.0173) |
| | LDAM | 0.2904 (± 0.0008) | 0.2994 (± 0.0128) | 0.3044 (± 0.0116) |
| | AUC-M | 0.2998 (± 0.0010) | 0.2786 (± 0.0456) | 0.3305 (± 0.0165) |
| | SmothAP | 0.2686 (± 0.0007) | 0.3276 (± 0.0063) | 0.3235 (± 0.0092) |
| | FastAP | 0.0169 (± 0.0031) | 0.0826 (± 0.0112) | 0.0202 (± 0.0002) |
| | MinMax | 0.2874 (± 0.0073) | 0.3119 (± 0.0075) | 0.3098 (± 0.0167) |
| | SOAP | **0.3485 (± 0.0083)** | **0.3401 (± 0.0045)** | **0.3547 (± 0.0077)** |
| MUV | CE | 0.0017 (±0.0001) | 0.0021 (±0.0002) | 0.0025 (±0.0004) |
| | CB-CE | 0.0055 (±0.0011) | 0.0483 (±0.0083) | 0.0121 (±0.0016) |
| | Focal | 0.0041 (±0.0007) | 0.0281 (±0.0141) | 0.0122 (±0.0001) |
| | LDAM | 0.0044 (±0.0022) | 0.0118 (±0.0098) | 0.0059 (±0.0021) |
| | AUC-M | 0.0026 (±0.0001) | 0.0040 (±0.0012) | 0.0028 (±0.0012) |
| | SmoothAP | 0.0073 (±0.0012) | 0.0068 (±0.0038) | 0.0029 (±0.0005) |
| | FastAP | 0.0016 (±0.0000) | 0.0023 (±0.0021) | 0.0022 (±0.0012) |
| | MinMax | 0.0028 (±0.0008) | 0.0027 (±0.0005) | 0.0043 (±0.0015) |
| | SOAP | **0.0493 (±0.0261)** | **0.3352** (±0.0008) | **0.0236** (±0.0038) |

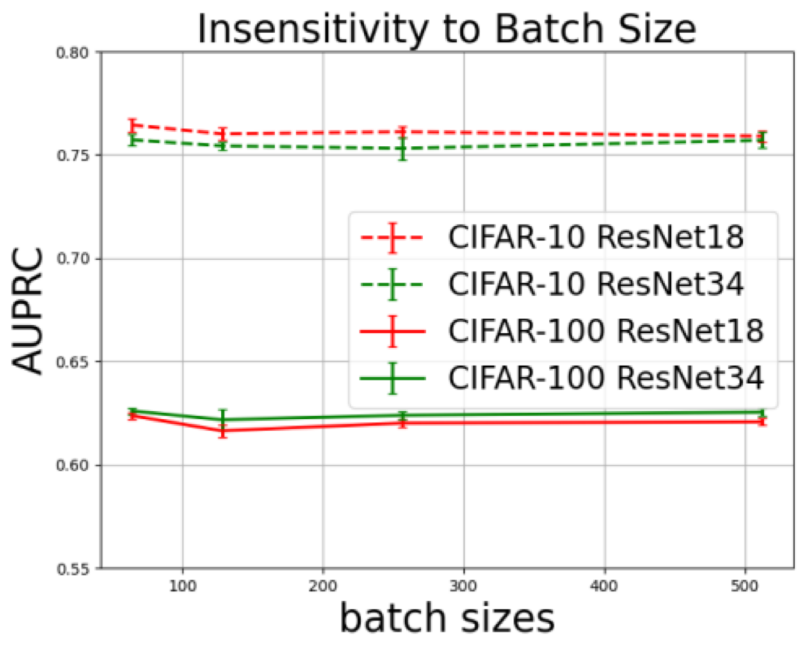
0.2% Positive 33% Improvement

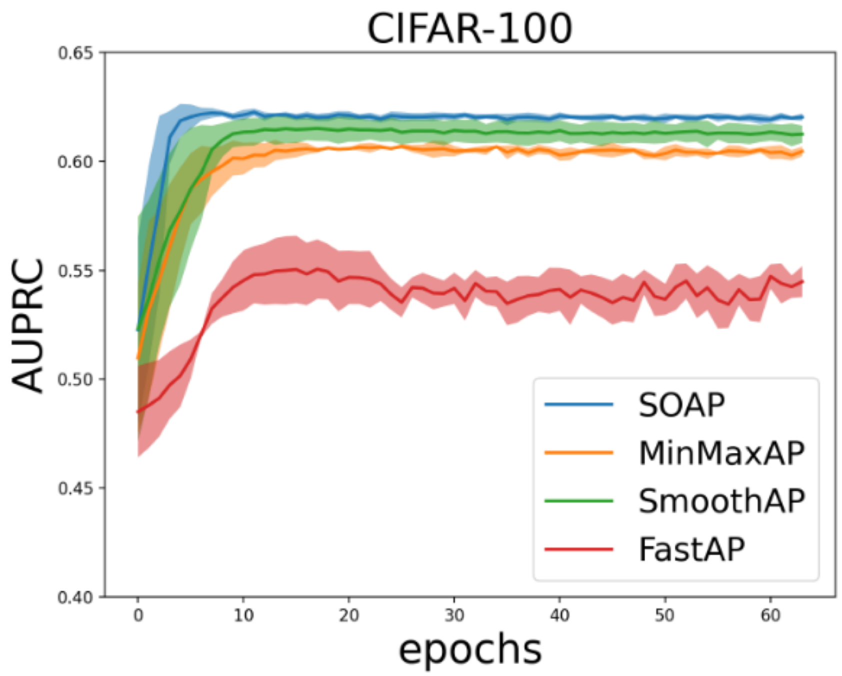
| Data | MIT AICURES | |
|---|---|---|
| Networks | GINE | MPNN |
| CE | 0.5037 (± 0.0718) | 0.6282 (± 0.0634) |
| CB-CE | 0.5655 (± 0.0453) | 0.6308 (± 0.0263) |
| Focal | 0.5143 (± 0.1062) | 0.5875 (± 0.0774) |
| LDAM | 0.5236 (± 0.0551) | 0.6489 (± 0.0556) |
| AUC-M | 0.5149 (± 0.0748) | 0.5542 (± 0.0474) |
| SmothAP | 0.2899 (± 0.0220) | 0.4081 (± 0.0352) |
| FastAP | 0.4777 (± 0.0896) | 0.4518 (± 0.1495) |
| MinMax | 0.5292 (± 0.0330) | 0.5774 (± 0.0468) |
| SOAP | **0.6639 (± 0.0515)** | **0.6547 (± 0.0616)** |

2.2% Positive 3% Improvement



**Molecular Properties Prediction** **Graph Neural Networks**


Insensitivity to Batch Size
0.80
0.75
0.70
0.65
0.60
0.55
AUPRC
100
200
300
400
500
batch sizes
CIFAR-10 ResNet18
CIFAR-10 ResNet34
CIFAR-100 ResNet18
CIFAR-100 ResNet34

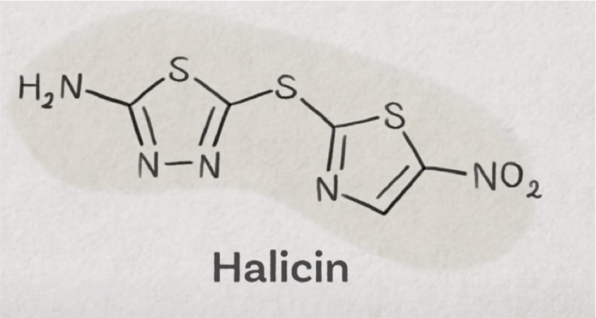

CIFAR-100
0.65
0.60
0.55
0.50
0.45
0.40
AUPRC
0
10
20
30
40
50
60
epochs
SOAP
MinMaxAP
SmoothAP
FastAP

# MIT AICures Challenge

**Evaluation Metric: AUPRC**

**1st Place**

**Fighting Secondary Effects of Covid**




Stokes et al. 2020. Cell.

Collaborating with Prof. Shuiwang Ji's group at TAMU

| Rank | Model | Author | Submissions | 10-fold CV ROC-AUC | 10-fold CV PRC-AUC | Test ROC-AUC | Test PRC-AUC |
|---|---|---|---|---|---|---|---|
| 1 | | DIVE@TAMU | 11 | | | 0.957 | 0.729 |
| 2 | MolecularG | AIDrug@PA | 9 | | | 0.7 | 0.725 |
| 3 | | AGL Team | 20 | | | 0.675 | 0.702 |
| 4 | | phucdoitoan@Fujitsu | 14 | 0.898 +/- 0.113 | 0.508 +/- 0.253 | 0.867 | 0.694 |
| 5 | GB | BI | 6 | | | 0.698 | 0.67 |
| 6 | Chemprop ++ | AICures@MIT | 4 | | | 0.877 | 0.662 |
| 7 | | Mingjun Liu | 3 | | | 0.72 | 0.657 |
| 8 | Pre-trained OGB-GIN (ensemble) | Weihua Hu@Stanford | 2 | 0.905 +/- 0.133 | 0.494 +/- 0.333 | 0.837 | 0.651 |
| 9 | RF + fingerprint | Cyrus Maher@Vir Bio | 1 | 0.896 +/- 0.074 | 0.481 +/- 0.338 | 0.799 | 0.649 |
| 10 | Graph Self-supervised Learning | SJTU_NRC_Mila | 3 | 0.825 +/- 0.210 | 0.530 +/- 0.342 | 0.800 | 0.622 |

# Comparison with w/o DAM

| Rank | Model | Author | Submissions | 10-fold CV ROC-AUC | 10-fold CV PRC-AUC | Test ROC-AUC | Test PRC-AUC |
|---|---|---|---|---|---|---|---|
| 1 | | DIVE@TAMU | 11 | | | 0.957 | 0.729 |

AUROC

| | | | | |
|---|---|---|---|---|
| 1 | MoleculeKit | DIVE@TAMU | 7 | 0.928 |

w/o DAM

AUPRC

| | | | | |
|---|---|---|---|---|
| 3 | MoleculeKit | DIVE@TAMU | 7 | 0.677 |

**5%** Improvement in **AUPRC,** **3%** Improvement in **AUROC**

# Deep top-K NDCG Maximization

Search Engines

Recommender Systems

Social Media

**M**ost **R**elevant Items on the **T**op

# NDCG

Relevance Score

$$\mathrm{NDCG}_q = \frac{1}{Z_q} \sum_{i=1}^{n} \frac{2^{y_i} - 1}{\log_2(1 + \mathrm{r}(i))}$$

Ideal DCG

Ranking position

**Challenge I**

$$r(i) = \sum_{\mathbf{x}_j \in \mathcal{S}_q} \mathbb{I}(h_{\mathbf{w}}(\mathbf{x}_j; q) \geq h_{\mathbf{w}}(\mathbf{x}_i; q))$$

**Millions of Movies on Netflix**

# NDCG Surrogate is X-risk

$$\mathrm{NDCG}_q = \frac{1}{Z_q}\sum_{i=1}^{n} \boxed{\frac{2^{y_i}-1}{\log_2(1+\mathrm{r}(i))}}$$

$$f(g(\mathbf{w};\mathbf{x}_i,\mathcal{S}_q))$$

$$\boxed{g(\mathbf{w};\mathbf{x}_i,\mathcal{S}_q) = \sum_{\mathbf{x}_j\in\mathcal{S}_q} \ell(h_{\mathbf{w}}(\mathbf{x}_j;q) - h_{\mathbf{w}}(\mathbf{x}_i;q))}$$

# Top-K NDCG

$$\frac{1}{Z_q^K}\sum_{i=1}^{n}\boxed{\mathbb{I}(i\text{-th item in top-K positions})}\frac{2^{y_i}-1}{\log_2(1+r(i))}$$

Top-K selector

$$f(g)$$

**Challenges**

- Finding top-K items require O(nlog n)
- Top-K selector is non-differentiable

# Deep top-K NDCG Maximization

**Limitations** of Literature on Top-K NDCG Maximization

(1) Small Data or No Convergence (e.g., ApproxNDCG, Qin et al.)

(2) Not Applicable to Deep Learning (e.g., SVM-NDCG, Chakrabarti et al.)

**Our Contributions:** (ICML'22)

**(1) N**ew Formulation based on Bilevel Optimization

**(2) F**irst Algorithms with Convergence Theory

**(3) P**ractical Algorithms

# Transforming Top-K Selector

(ICML 2022)

Prediction score

The **(K+1)-th** largest score

$$\mathbb{I}(h_{\mathbf{w}}(\mathbf{x}_i;q) > \lambda_q(\mathbf{w}))$$

$$\lambda_q(\mathbf{w}) = \arg\min_{\lambda} \frac{K+\varepsilon}{n}\lambda + \frac{1}{n}\sum_{i=1}^{n}(h_{\mathbf{w}}(\mathbf{x}_i;q) - \lambda)_+$$

# New Formulation

(ICML 2022)

Multi-block Bilevel Optimization

$$\min \quad \frac{1}{\mathcal{S}} \sum_{(q,\mathbf{x}_i^q)\in\mathcal{S}} \boxed{\sigma(h_{\mathbf{w}}(\mathbf{x}_i^q;q) - \lambda_q(\mathbf{w}))} f(g_{q,i}(\mathbf{w}))$$

$$s.t. \quad \lambda_q(\mathbf{w}) = \arg\min_{\lambda} L_q(\lambda, \mathbf{w}, \mathcal{S}_q), \forall q \in \mathcal{Q}$$

$$f(g_i(\mathbf{w}))$$

# Challenges

(ICML 2022)

$$\nabla\sigma(h_{\mathbf{w}}(\mathbf{x}_i^q;q)-\underbrace{\lambda_q(\mathbf{w})}_{\text{Depends on } \mathcal{S}_q})(\nabla h_{\mathbf{w}}(\mathbf{x}_i^q;q)-\underbrace{\nabla\lambda_q(\mathbf{w})}_{\text{Implicit Gradient}})$$

# Tackle Challenges (K-SONG)

(ICML 2022)

SGD on Lower Level

$\lambda_q(\mathbf{w}_t)$

**Depends on $\mathcal{S}_q$**

$$\lambda_q^{t+1} = \lambda_q^t - \eta_0 \nabla_\lambda L_q(\lambda_q^t, \mathbf{w}_t, \mathcal{B}_2^t)$$

**Only for Sampled Query**

$\nabla \lambda_q(\mathbf{w}_t)$

**Implicit Gradient**

- Smoothing Lower Level
- Estimating Hessian Inverse
- Using Mini-batches

# Theories

Goal $$\|\nabla F(\mathbf{w})\| \leq \epsilon$$

ICML'22 $$O\left(\frac{1}{\epsilon^4}\right)$$

Learning to rank

Table 2: The test NDCG on two Learning to Rank datasets. We report the average NDCG@$k$ ($k \in [10, 30, 60]$) and standard deviation (within brackets) over 5 runs with different random seeds.

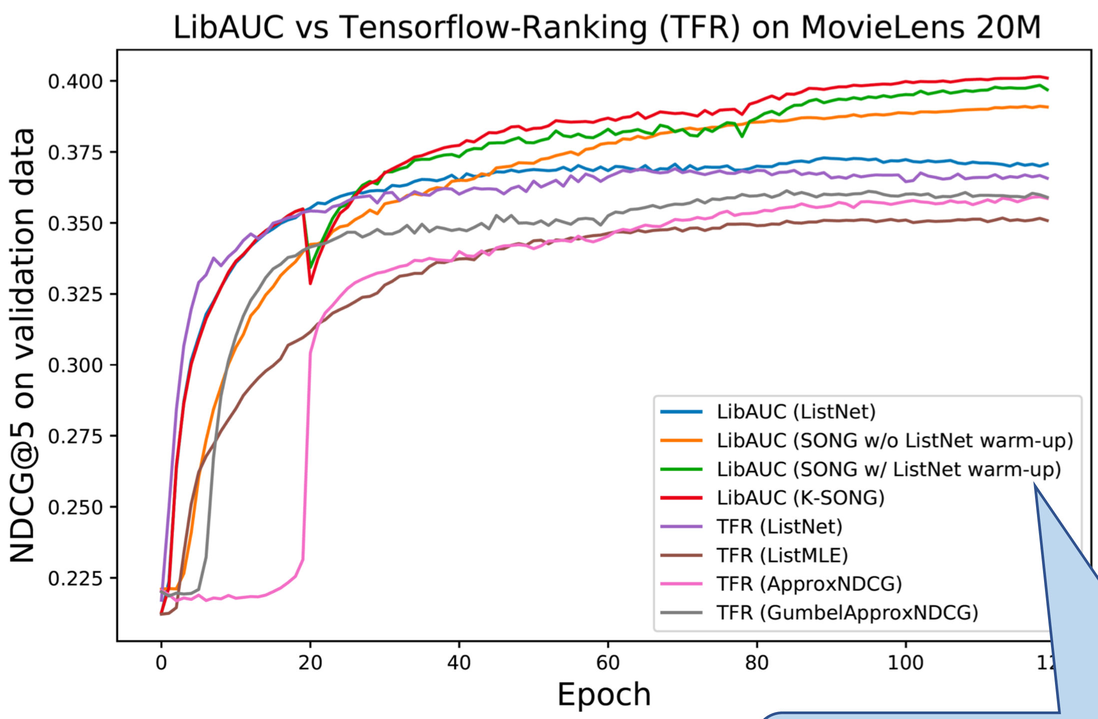
| Method | MSLR WEB30K | | | Yahoo! LTR Dataset | | |
|---|---|---|---|---|---|---|
| | NDCG@10 | NDCG@30 | NDCG@60 | NDCG@10 | NDCG@30 | NDCG@60 |
| RankNet | 0.5227±0.0012 | 0.5837±0.0006 | 0.6481±0.0007 | 0.7668±0.0007 | 0.8319±0.0008 | 0.8491±0.0008 |
| ListNet | 0.5337±0.0022 | 0.5910±0.0019 | 0.6535±0.0014 | 0.7805±0.0010 | 0.8441±0.0006 | 0.8613±0.0005 |
| ListMLE | 0.5210±0.0017 | 0.5800±0.0015 | 0.6450±0.0012 | 0.7796±0.0007 | 0.8436±0.0006 | 0.8606±0.0006 |
| LambdaRank | 0.5324±0.0037 | 0.5885±0.0032 | 0.6529±0.0026 | 0.7794±0.0009 | 0.8442±0.0008 | 0.8619±0.0007 |
| ApproxNDCG | 0.5339±0.0008 | 0.5906±0.0005 | 0.6530±0.0003 | 0.7688±0.0004 | 0.8367±0.0004 | 0.8556±0.0004 |
| NeuralNDCG | 0.5329±0.0027 | 0.5881±0.0013 | 0.6510±0.0012 | 0.7812±0.0002 | 0.8443±0.0002 | 0.8622±0.0003 |
| SONG | 0.5382±0.0007 | 0.5953±0.0006 | **0.6573**±0.0005 | 0.7842±0.0004 | **0.8477**±0.0003 | **0.8644**±0.0003 |
| K-SONG | **0.5397**±0.0009 | **0.5955**±0.0004 | 0.6571±0.0003 | **0.7859**±0.0003 | 0.8464±0.0002 | 0.8642±0.0003 |

Movie Recommendation

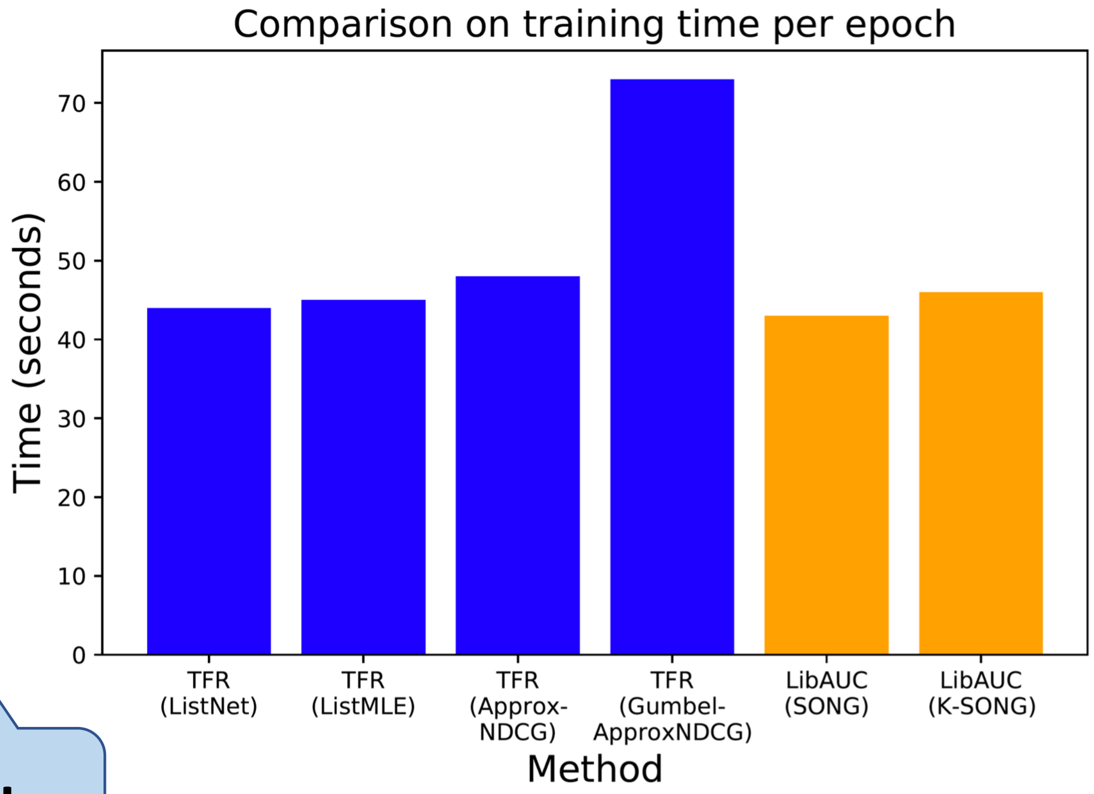
Table 4: The test NDCG on two movie recommendation datasets. We report the average NDCG@$k$ ($k \in [10, 20, 50]$) and standard deviation (within brackets) over 5 runs with different random seeds.

| Method | MovieLens20M | | | Netflix Prize Dataset | | |
|---|---|---|---|---|---|---|
| | NDCG@10 | NDCG@20 | NDCG@50 | NDCG@10 | NDCG@20 | NDCG@50 |
| RankNet | 0.0109±0.0011 | 0.0190±0.0010 | 0.0450±0.0016 | 0.0090±0.0007 | 0.0146±0.0008 | 0.0261±0.0010 |
| ListNet | 0.0182±0.0004 | 0.0305±0.0002 | 0.0587±0.0004 | 0.0115±0.0018 | 0.0191±0.0013 | 0.0347±0.0014 |
| ListMLE | 0.0117±0.0005 | 0.0210±0.0011 | 0.0493±0.0010 | 0.0081±0.0005 | 0.0134±0.0009 | 0.0253±0.0005 |
| LambdaRank | 0.0178±0.0010 | 0.0310±0.0008 | 0.0595±0.0006 | 0.0103±0.0003 | 0.0175±0.0003 | 0.0332±0.0004 |
| ApproxNDCG | 0.0202±0.0004 | 0.0338±0.0004 | 0.0629±0.0004 | 0.0121±0.0015 | 0.0198±0.0005 | 0.0360±0.0006 |
| NeuralNDCG | 0.0194±0.0013 | 0.0322±0.0011 | 0.0609±0.0012 | 0.0113±0.0011 | 0.0186±0.0008 | 0.0342±0.0007 |
| SONG | 0.0232±0.0003 | 0.0369±0.0004 | 0.0646±0.0003 | 0.0141±0.0004 | 0.0222±0.0005 | **0.0384**±0.0003 |
| K-SONG | **0.0248**±0.0003 | **0.0381**±0.0003 | **0.0662**±0.0004 | **0.0154**±0.0003 | **0.0234**±0.0006 | 0.0377±0.0005 |

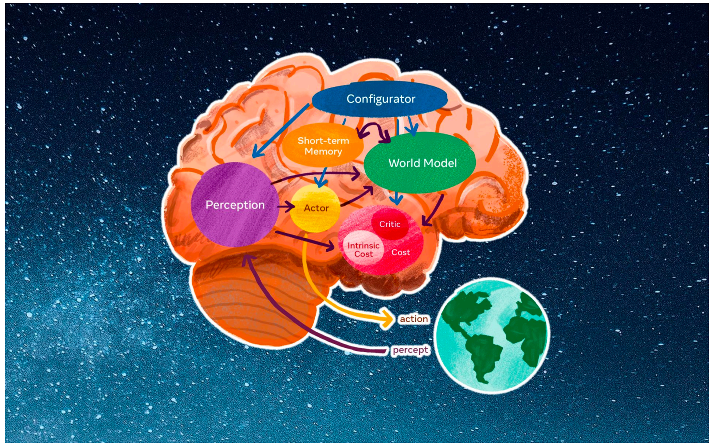
# Movielens: 20 Millions User-Movie Pairs



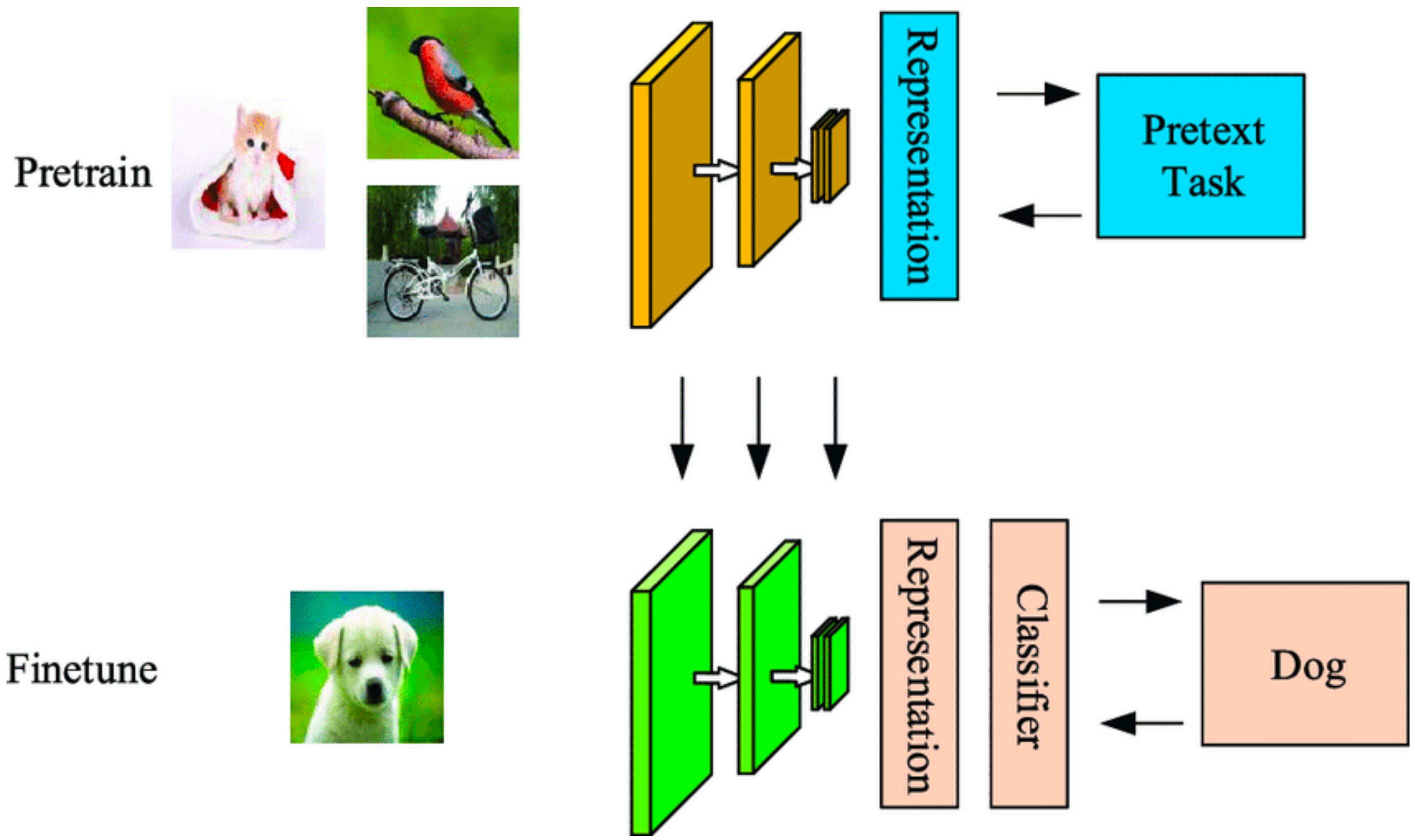

**ListNet as X-risk**



# Self-supervised Learning



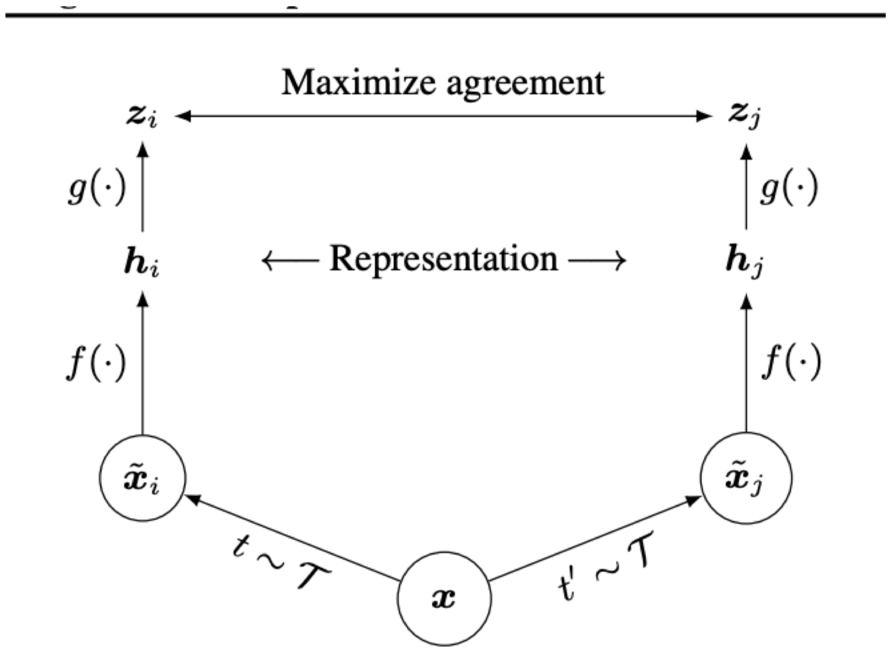
# Self-supervised learning



# SimCLR: Simple Contrastive Learning

A Simple Framework for Contrastive Learning of Visual ... - arXiv

by T Chen · 2020 · Cited by 3849 — Abstract: This paper presents **SimCLR**: a simple framework for contrastive learning of visual representations. We simplify recently proposed ...



# Mini-batch Contrastive Loss

Encoder Network

Data Augmentation

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

Mini-Batch Data

# Issue of SimCLR

Huge Difference between large batch and small batch

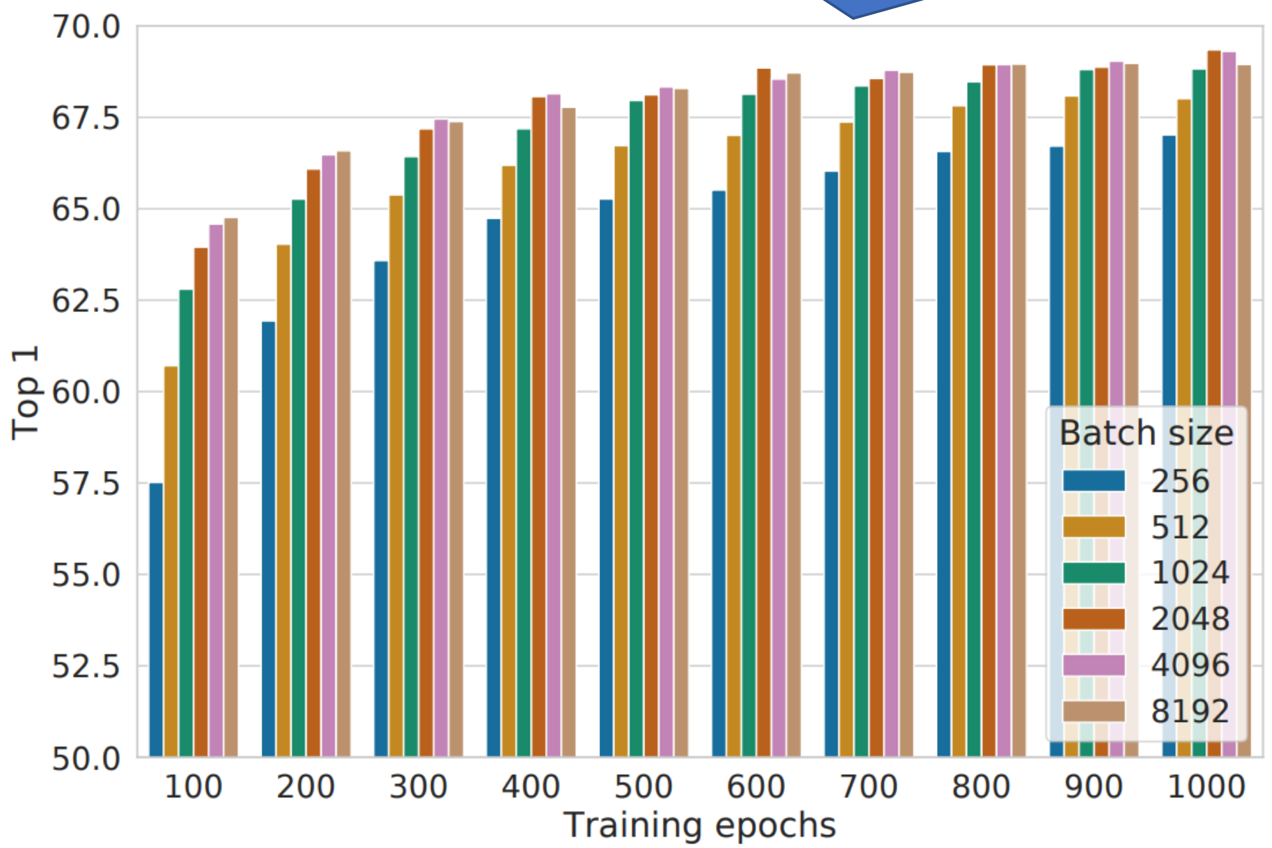



*Figure 9.* Linear evaluation models (ResNet-50) trained with different batch size and epochs. Each bar is a single run from scratch.[10]

Chen et al. 2020

**Our Contributions:**

(1) Explanation of Large Batch of SimCLR

(2) New Method SogCLR without Large Batch Size

# How do we understand the issue of SimCLR?

## Global Contrastive Loss is the Key

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

All Images Except x_i

## Global Contrastive Objective is X-risk

$$F(\mathbf{w}) = \mathbb{E}_{\mathbf{x}_i\sim\mathcal{D},\mathcal{A},\mathcal{A}'\sim\mathcal{P}}(E(\mathcal{A}(\mathbf{x}_i))^\top E(\mathcal{A}'(\mathbf{x}_i))) + \frac{\tau}{n}\sum_{\mathbf{x}_i\in\mathcal{D}}\mathbb{E}_{\mathcal{A}}\boxed{\ln\left(\frac{1}{|\mathcal{S}_i|}g(\mathbf{w};\mathbf{x}_i,\mathcal{A},\mathcal{S}_i)\right)},$$

$$\boxed{f(g(\mathbf{w};\mathbf{x}_i,\mathcal{A},\mathcal{S}_i))}$$

# SimCLR Suffers from Small Batch Size

$$\frac{1}{n}\sum_{\mathbf{x}_i\in\mathcal{D}} \mathbb{E}_{\mathcal{A}} f(g(\mathbf{w};\mathbf{x}_i,\mathcal{A},\mathcal{S}_i))$$

$$\nabla f(\boxed{g(\mathbf{w};\mathbf{x}_i,\mathcal{A},\mathcal{S}_i)})\boxed{\nabla g(\mathbf{w};\mathbf{x}_i,\mathcal{A},\mathcal{S}_i)}$$

SimCLR uses the Standard learning Paradigm

$$\nabla f(g(\mathbf{w};\mathbf{x}_i,\mathcal{A},\mathcal{B}_i))\nabla g(\mathbf{w};\mathbf{x}_i,\mathcal{A},\mathcal{B}_i)$$

$$\Rightarrow \mathbb{E}[\|\nabla F(\mathbf{w})\|] \leq O\left(\frac{1}{\sqrt{B}}\right)$$

Mini-batch

# Better way to Optimize GCL: SogCLR

**Estimating inner g**

$$\nabla f(\boxed{g(\mathbf{w};\mathbf{x}_i,\mathcal{A},\mathcal{S}_i)})\nabla g(\mathbf{w};\mathbf{x}_i,\mathcal{A},\mathcal{S}_i)$$

Maintain and update $u(\mathbf{x}_i,\mathcal{A})$ ? **Too Much Memory**  $u(\mathbf{x}_i)$

# SogCLR

Update *u*

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

Mini-batch

Compute Gradient Estimator

$$\mathbf{m}_t = -\frac{1}{B}\sum_{\mathbf{x}_i\in\mathcal{B}} \nabla(E(\mathcal{A}(\mathbf{x}_i))^\top E(\mathcal{A}'(\mathbf{x}_i))) + \boxed{\nabla f(u_{i,t-1})}\frac{1}{2|\mathcal{B}_i|}(\nabla g(\mathbf{w}_t;\mathbf{x}_i,\mathcal{A},\mathcal{B}_i) + \nabla g(\mathbf{w}_t;\mathbf{x}_i,\mathcal{A}',\mathcal{B}_i)).$$

This is the Key

Update *w*

$$\mathbf{v}_t = (1-\beta)\mathbf{v}_{t-1} + \beta\mathbf{m}_t$$
$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta\mathbf{v}_t \text{ (or use Adam-style update)}$$

# Theory of SogCLR

Theorem 1

Quantify difference of different augmented copies

$$\mathbb{E}[\|\nabla F(\mathbf{w}_{t'})\|^2] \leq O\left(\frac{1}{\sqrt{BT}} + \frac{\sqrt{n}}{B\sqrt{T}} + \epsilon^2\right)$$

Theorem 2

$$L_2(\mathbf{w};\mathbf{x}_i,\mathcal{A},\mathcal{A}') = -\ln\frac{\exp(E(\mathcal{A}(\mathbf{x}_i))^\top E(\mathcal{A}'(\mathbf{x}_i))/\tau)}{\mathbb{E}_{\mathcal{A}}g(\mathbf{w};\mathbf{x}_i,\mathcal{A},\mathcal{S}_i)}.$$

$$\mathbb{E}[\|\nabla F_{v2}(\mathbf{w}_{t'})\|^2] \leq O\left(\frac{1}{\sqrt{BT}} + \frac{\sqrt{n}}{B\sqrt{T}}\right). \xrightarrow{T\to\infty} 0$$

# Experiments

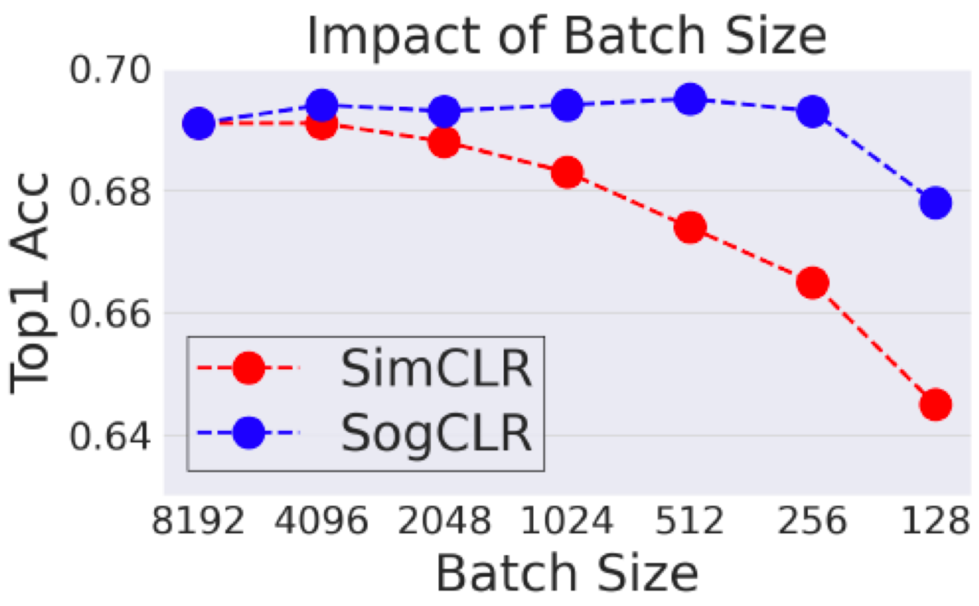



Table 6: Comparison of small-batch training approaches.

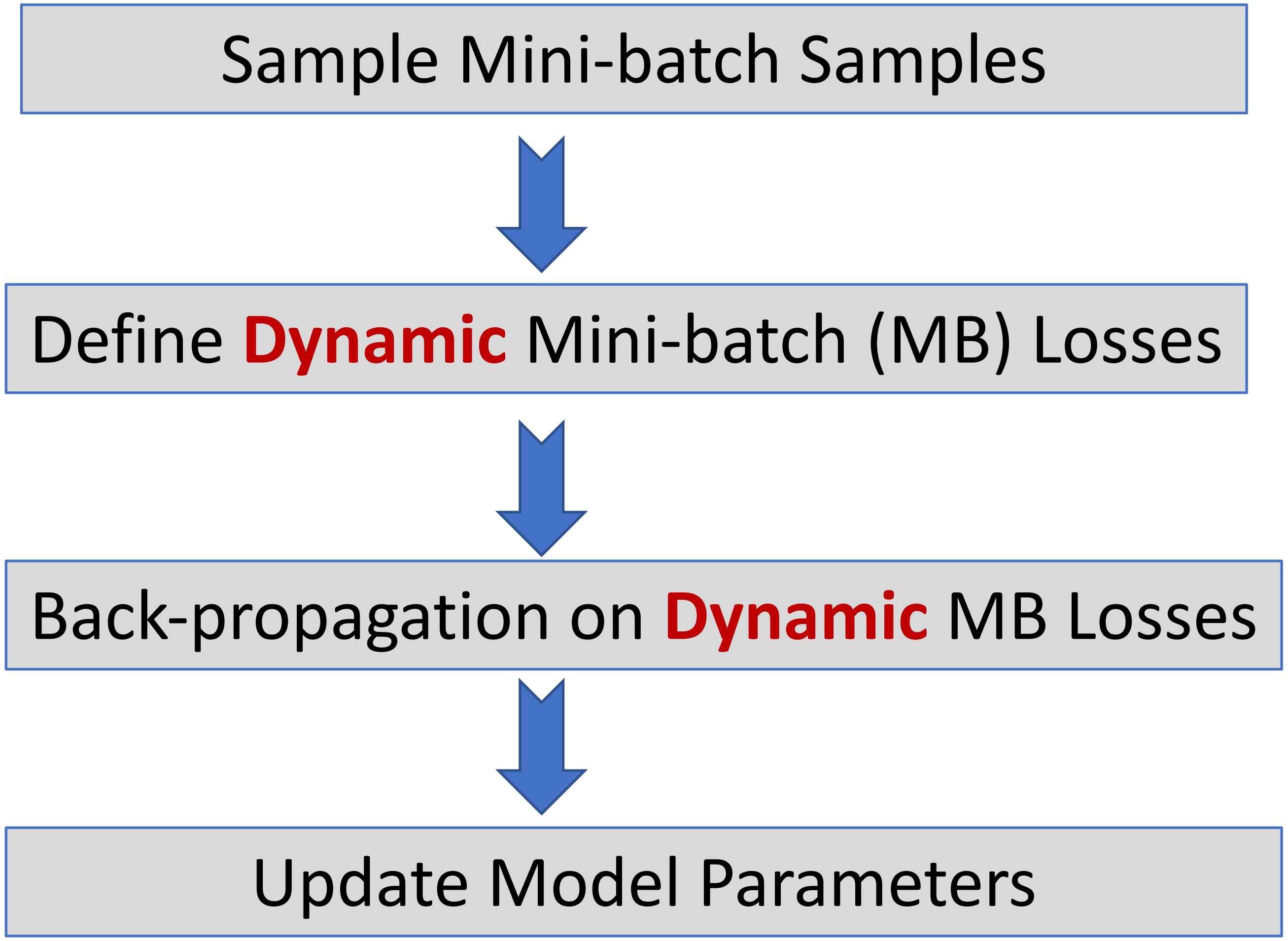
| Method | Batch Size\Epochs | 100 | 200 | 400 | 800 |
|---|---|---|---|---|---|
| SimCLR | 256 | 69.7 | 73.6 | 76.1 | 77.4 |
| FlatNCE | 256 | 71.5 | 75.5 | 76.7 | 77.8 |
| SiMo | 256 | 71.5 | 75.0 | 76.8 | 78.2 |
| SogCLR | 256 | **71.9** | **76.3** | **78.7** | **79.4** |

Table 1: Comparison of different InfoNCE-loss based contrastive learning methods and their top-1 linear evaluation accuracy by using 800 epochs, a batch size of 256, and ResNet-50 on ImageNet-1K. Momentum encoder is introduced by MoCo [20]. We expect the performance of SogCLR can be further improved by incorporating other techniques, e.g., InfoMin augmentation.

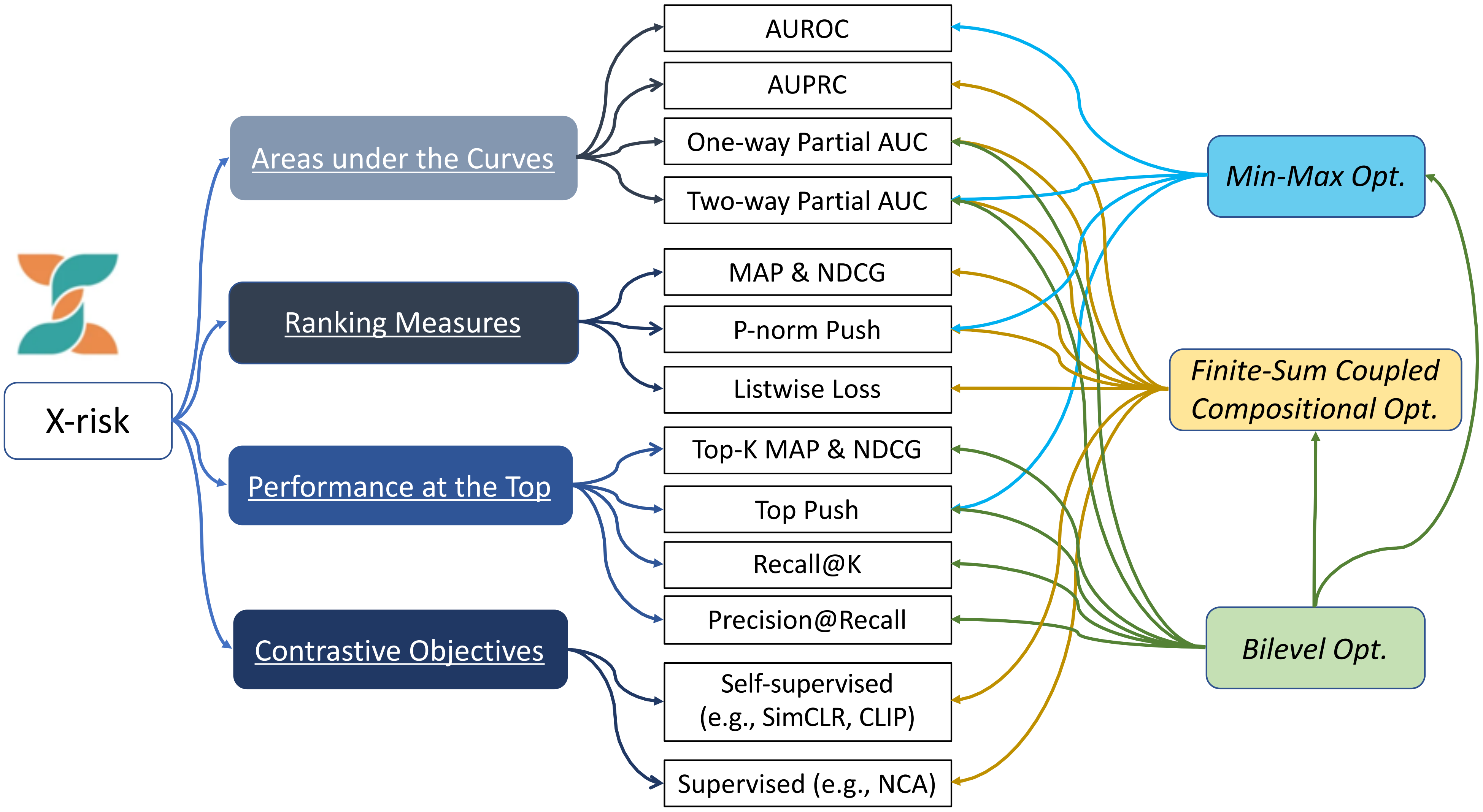
| Method | Batch Size | Memory Bank | Momentum Encoder | Other Tricks | Convergence | Top1 Acc. |
|---|---|---|---|---|---|---|
| SimCLR [4] | Large-batch | No | No | Strong Aug. | No | 66.5 |
| NNCLR [15] | Large-batch | No | No | Nearest Neighbors | No | 68.7 |
| SiMo [44] | Small-batch | No | Yes | Margin Trick | No | 72.1 |
| MoCov2 [6] | Small-batch | Yes | Yes | Strong Aug. | No | 71.1 |
| InfoMin [36] | Small-batch | Yes | Yes | InfoMin Aug. | No | 73.0 |
| SogCLR (Ours) | Small-batch | No | No | GC Optimization | Yes | 72.5 |

# Summary: X-risk as a New Learning Paradigm

- **A**ny Batch Size
- **B**road Applications
- **C**onvergence Guarantee
- **E**asy Implementation



# More X-risks



X-risk
Areas under the Curves
Ranking Measures
Performance at the Top
Contrastive Objectives
AUROC
AUPRC
One-way Partial AUC
Two-way Partial AUC
MAP & NDCG
P-norm Push
Listwise Loss
Top-K MAP & NDCG
Top Push
Recall@K
Precision@Recall
Self-supervised
(e.g., SimCLR, CLIP)
Supervised (e.g., NCA)
Min-Max Opt.
Finite-Sum Coupled Compositional Opt.
Bilevel Opt.

# libauc.org

LibAUC

LibAUC Installation Examples Research Talks Team Github

## A DEEP LEARNING LIBRARY FOR X-RISK OPTIMIZATION

An open-source library that translates theories to real-world applications

Latest News Install

[2022-06] 7 papers about optimization for ML/AI accepted to ICML 2022!

## 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 any 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.

**Hands-on Tutorials**

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

# Impact of LibAUC Library

## QUICK FACTS

The achievements we made so far.

**3+**

Challenges winning solution (e.g., Stanford CheXpert, MIT AICures, OGB Graph Property Prediction).

**4+**

Collaborations and Deployments at multiple industrial units, e.g., Google, Uber, Tencent, etc.

**17+**

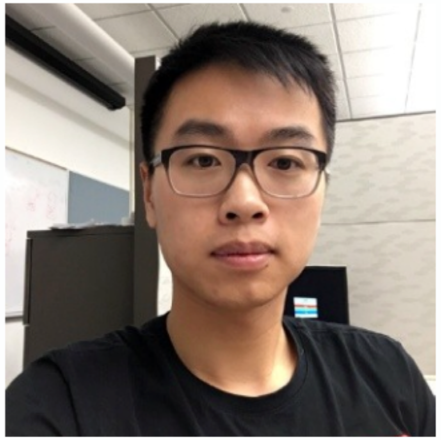
Scientific publications on top-tier AI Conferences (such as ICML, NeurIPS, ICLR).

**13000+**

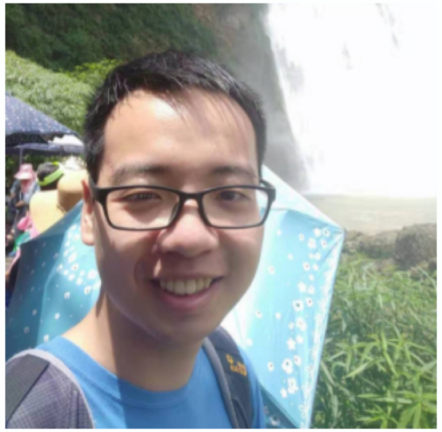
Downloaded by more than 13K+ times from over 11 countries.

# Acknowledgements: Students

**Main Development**



**Zhuoning Yuan**
PhD Student
University of Iowa



**Zi-Hao Qiu**
PhD Student
Nanjing University

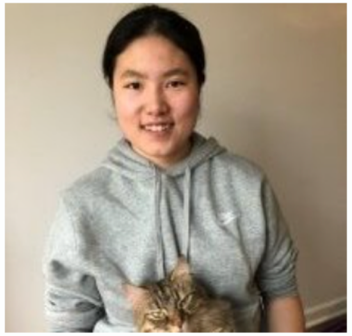
**Dixian Zhu**
PhD Student
University of Iowa

**Gang Li**
PhD Student
University of Iowa

# Acknowledgements: Students

**Other Contributors**

**Zhishuai Guo**
PhD Student
University of Iowa

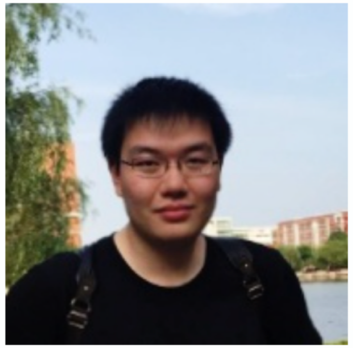


**Quanqi Hu**
PhD Student
University of Iowa

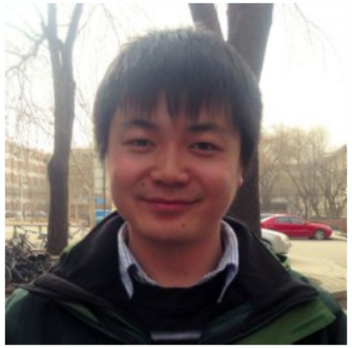

**Bokun Wang**
PhD Student
University of Iowa

**Qi Qi**
PhD Student
University of Iowa

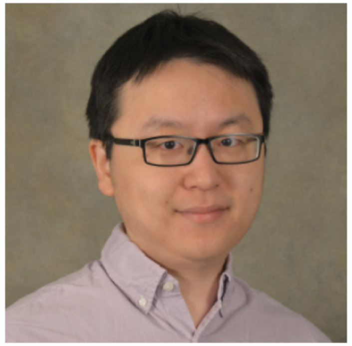
**Yongjian Zhong**
PhD Student
University of Iowa



**Mingrui Liu**
Assistant Professor
George Mason University



**Yan Yan**
Assistant Professor
Washington State University



**Yi Xu**
Associate Professor
Dalian University of Technology

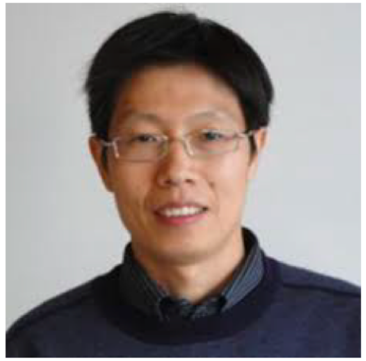
# Acknowledgements: Collaborators

Milan Sonka
(UIowa)

Nitesh Chawla
(ND)

Hassan Rafique
(UIndy)

Qihang Lin
(UIowa)



Yiming Ying
(UAlbany)

Shuiwang Ji
(TAMU)

# Acknowledgements

Big Data, Career, III, RI, Engineering, Smart Health, Fair AI


Q
A