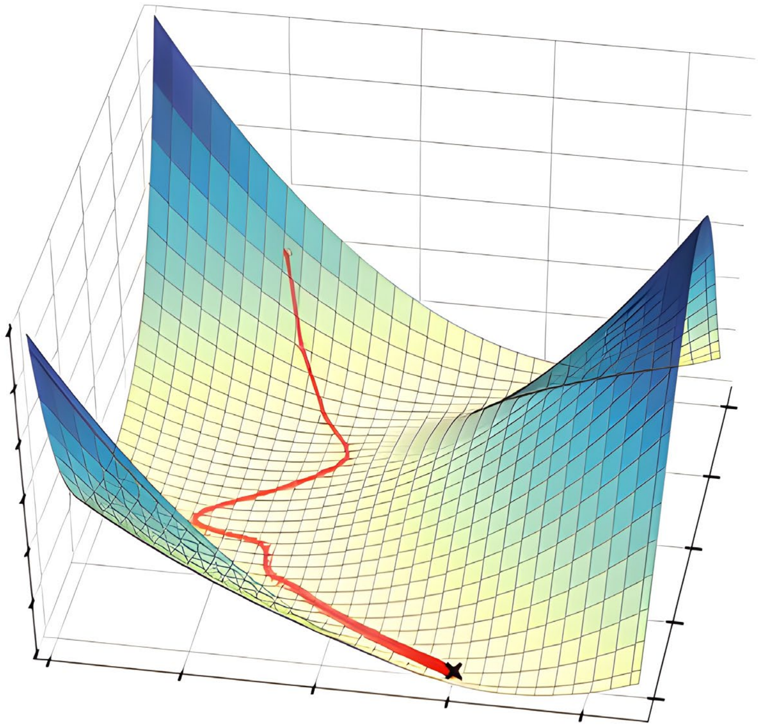
Bridging Discrete and Backpropagation: Straight-Through and Beyond

Liyuan Liu, Chengyu Dong, Xiaodong Liu, Bin Yu, Jianfeng Gao

Microsoft Research

Deep Learning and Gradient Descent



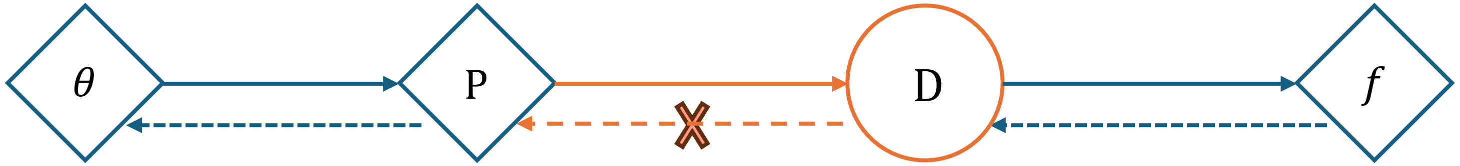
Back-propagation and Chain's Rule

For differentiable functions, back-propagation allows the gradient to be computed efficiently.

$$\frac{\partial f}{\partial \theta} = \frac{\partial f}{\partial D} \cdot \frac{\partial D}{\partial P} \cdot \frac{\partial P}{\partial \theta}$$

Back-propagation and Discrete Variables

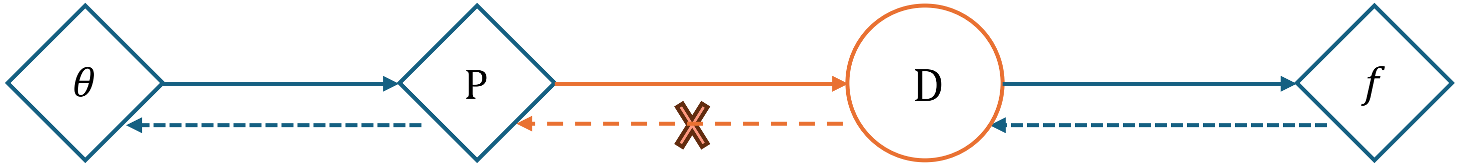
For applications involving discrete variables, back-propagation can not be directly applied as before.



$$\frac{\partial f}{\partial \theta} = \frac{\partial f}{\partial D} \cdot \cancel{\frac{\partial D}{\partial P}} \cdot \frac{\partial P}{\partial \theta}$$

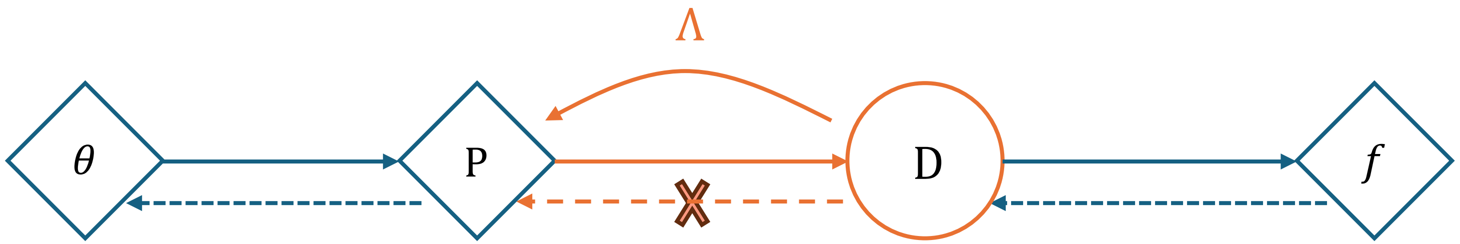
Back-propagation and Discrete Variables

This challenge impacts various applications, including Mixture-of-Experts, Differentiable Neural Architecture Search, Discrete Variational Autoencoder.



$$\frac{\partial f}{\partial \theta} = \frac{\partial f}{\partial D} \cdot \cancel{\frac{\partial D}{\partial P}} \cdot \frac{\partial P}{\partial \theta}$$

Bridge Back-propagation and Discrete

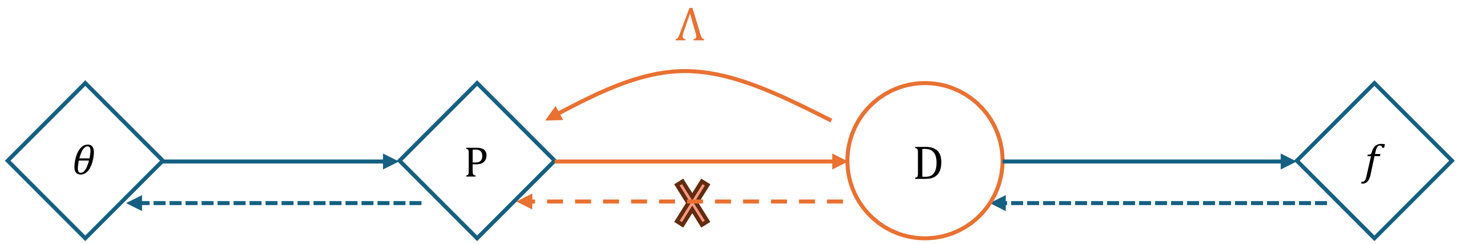


$$\frac{\partial f}{\partial \theta} = \frac{\partial f}{\partial D} \cdot \cancel{\frac{\partial D}{\partial P}} \cdot \frac{\partial P}{\partial \theta}$$

$$\frac{\partial f}{\partial \theta} \approx \frac{\partial f}{\partial D} \cdot \Lambda \cdot \frac{\partial P}{\partial \theta}$$

Bridge Back-propagation and Discrete

We propose ReinMax to bridge discrete and back-propagation. It achieves second-order accuracy with little computation overheads.



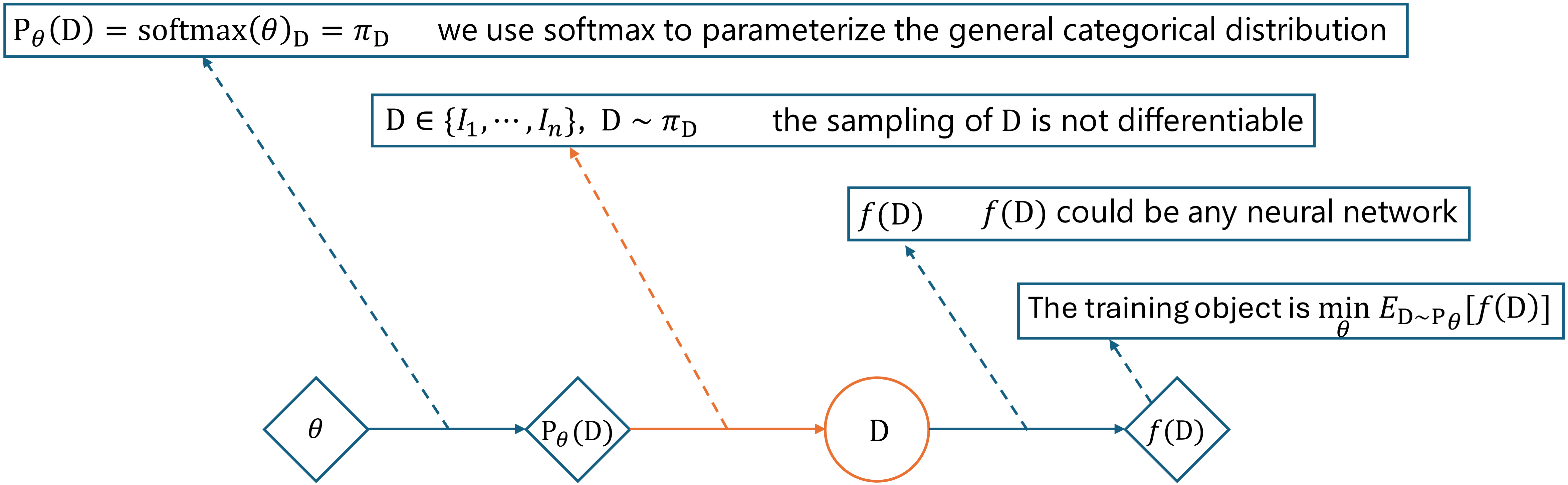
$$\frac{\partial f}{\partial \theta} = \frac{\partial f}{\partial D} \cdot \cancel{\frac{\partial D}{\partial P}} \cdot \frac{\partial P}{\partial \theta}$$

$$\frac{\partial f}{\partial \theta} \approx \frac{\partial f}{\partial D} \cdot \Lambda \cdot \frac{\partial P}{\partial \theta}$$

Outline

- Background
- Gradient Approximation: a Numerical ODE Perspective
  - How Straight-Through Works?
  - How to Make it Better?
- Discussions and Experiments

Simplified Scenario as Problem Setting



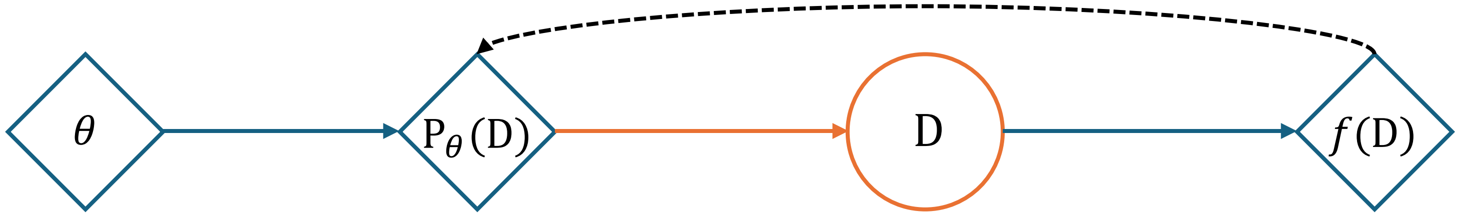
The Gradient

$$\min_{\theta} E_{\mathrm{D}\sim \mathrm{P}_\theta}[f(\mathrm{D})] = \min_{\theta} \sum_{\mathrm{D}} f(\mathrm{D}) \cdot P_\theta(\mathrm{D})$$

$$\nabla = \sum_{\mathrm{D}} f(\mathrm{D}) \cdot \frac{\partial P_\theta(\mathrm{D})}{\partial \theta} = \boxed{E_{\mathrm{D}\sim \mathrm{P}_\theta}\left[\frac{f(\mathrm{D})}{P_\theta(\mathrm{D})} \frac{\partial P_\theta(\mathrm{D})}{\partial \theta}\right]}$$

This is known as the REINFORCE algorithm

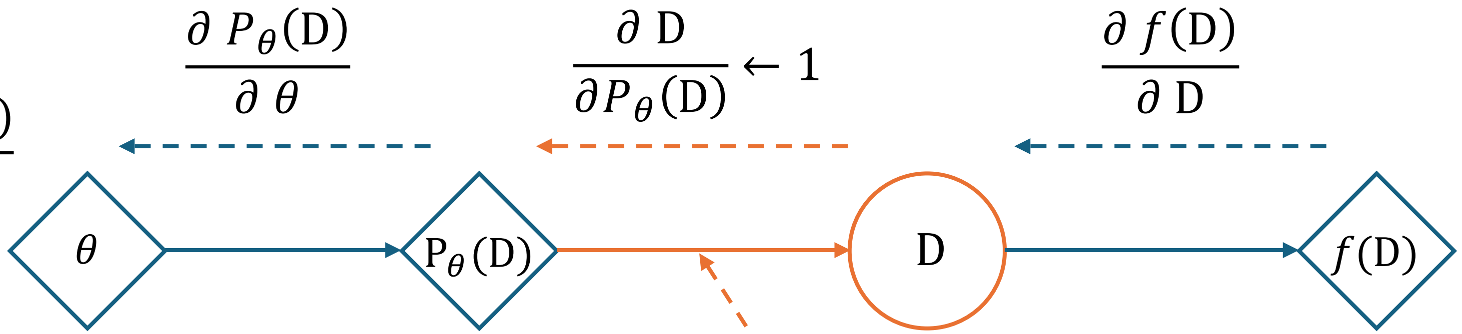
Although REINFORCE provides unbiased gradient estimations, in practice, it is usually hard to apply REINFORCE, as it suffers from a large variance



Straight-Through Gradient Approximation

In practice, a commonly used technique is called straight-through. It treats non-differentiable function (e.g., the sampling of D) as if it is an identity function in gradient computation.

$$\widehat{\nabla}_{\mathrm{ST}} = \frac{\partial f(D)}{\partial D} \cdot \frac{\partial P_\theta(D)}{\partial \theta}$$



$$\mathrm{D} \leftarrow \mathrm{P}_\theta(\mathrm{D}) - \mathrm{P}_\theta(\mathrm{D}).detach() + \mathrm{D}$$

Straight-Through Gumbel-Softmax

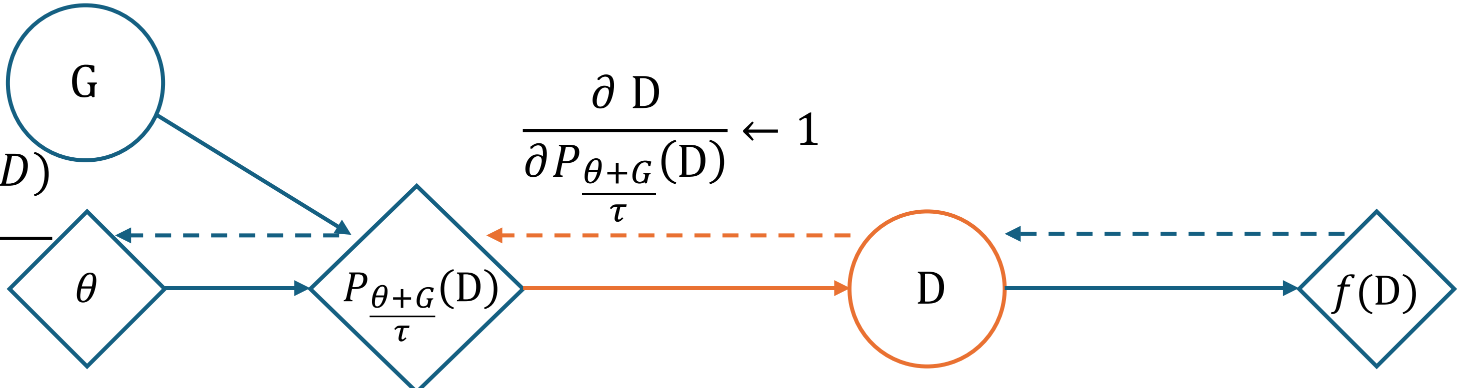
- A more popular family of Straight-Through estimators is Straight-Through Gumbel-Softmax (STGS)

- Gumbel-Softmax Trick—the sampling of D ($\mathrm{D} \sim \mathrm{P}_\theta$) can be reparameterized as zero-temperature limit:

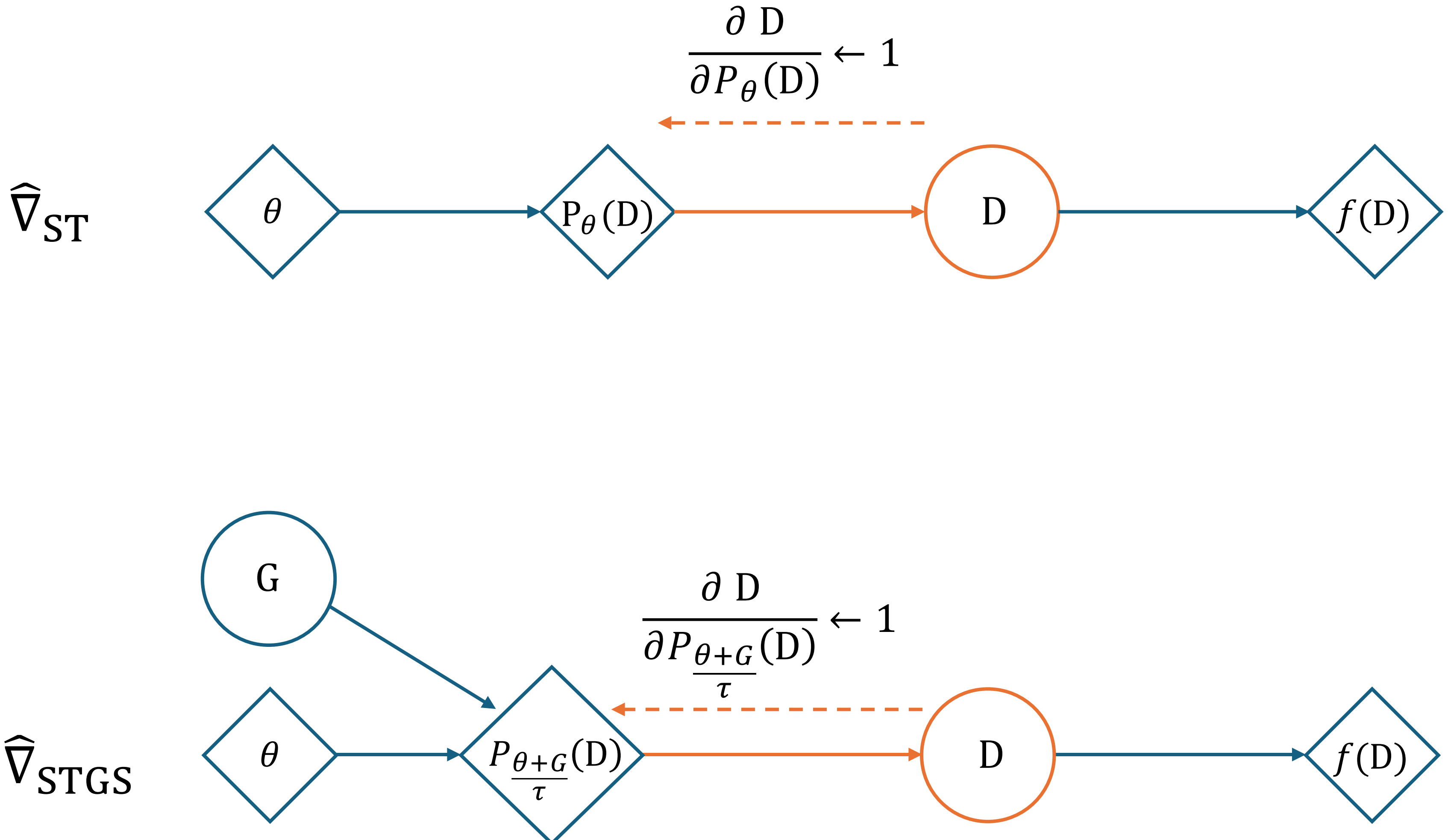
$$\mathrm{D} = \lim_{\tau \to 0} \mathrm{softmax}_\tau(\theta + G) \quad \text{where } G_i \text{ are i.i.d. and } G_i \sim \mathrm{Gumbel}(0, 1)$$

- STGS treats the zero-temperature limit as identity function when compute gradients.

$$\hat{\nabla}_{\mathrm{STGS}} = \frac{\partial f(D)}{\partial D} \cdot \frac{\partial P_{\frac{\theta+G}{\tau}}(D)}{\partial \theta}$$



How ST Works?


$\frac{\partial \mathrm{D}}{\partial P_\theta(\mathrm{D})} \leftarrow 1$
$\widehat{\nabla}_{\mathrm{ST}}$
θ
$\mathrm{P}_\theta(\mathrm{D})$
D
$f(\mathrm{D})$
G
$\frac{\partial \mathrm{D}}{\partial P_{\frac{\theta+G}{\tau}}(\mathrm{D})} \leftarrow 1$
$\widehat{\nabla}_{\mathrm{STGS}}$
θ
$P_{\frac{\theta+G}{\tau}}(\mathrm{D})$
D
$f(\mathrm{D})$

Outline

- Background
- Gradient Approximation: a Numerical ODE Perspective
  - How Straight-Through Works?
  - How to Make it Better?
- Discussions and Experiments

How Straight-Through Works?

Theorem 1: $\mathrm{E}[\widehat{\nabla}_{\mathrm{ST}}]$ is a first-order approximation of ∇, where $\widehat{\nabla}_{\mathrm{ST}} = \frac{\partial f(D)}{\partial D} \cdot \frac{\partial P_\theta(D)}{\partial \theta}$

$$\nabla = \sum_{\mathrm{D}} f(\mathrm{D}) \cdot \frac{\partial P_\theta(\mathrm{D})}{\partial \theta}$$

Subtract $\mathrm{E}[f(\mathrm{D})]$ as the baseline

$$\nabla = \sum_i \sum_{\mathrm{j}} P_\theta(I_j) \cdot (f(I_i) - f(I_j)) \cdot \frac{\partial P_\theta(I_i)}{\partial \theta}$$

Approximate $f(I_i) - f(I_j)$ as $\frac{\partial f(I_j)}{\partial I_j}(I_i - I_j)$

$$\mathrm{E}[\widehat{\nabla}_{\mathrm{ST}}] = \mathrm{E}[\frac{\partial f(D)}{\partial D} \cdot \frac{\partial P_\theta(D)}{\partial \theta}]$$

Discussions on Theorem 1

- In Tokui & Sato (2017), the authors positioned $\mathrm{E}[\widehat{\nabla}_{\mathrm{ST}}]$ as a first-order approximation, but their analyses are exclusively rooted in the properties of Bernoulli variables:
  - Consider $\mathrm{D} \in \{I_1, I_2\}$, we have: $\nabla = \left(\mathrm{f}(\mathrm{I}_1) - \mathrm{f}(\mathrm{I}_2)\right) \cdot \frac{\partial P_\theta(\mathrm{I}_2)}{\partial \theta} = \left(\mathrm{f}(\mathrm{I}_2) - \mathrm{f}(\mathrm{I}_1)\right) \cdot \frac{\partial P_\theta(\mathrm{I}_1)}{\partial \theta}$

- The analyses in Gregor et al. (2014) and Pervez et al. (2020) are applicable to multinomial variables, but resort to adding a term, i.e., $E[\frac{1}{n \cdot \pi_D} \widehat{\nabla}_{\mathrm{ST}}]$ is positioned as a first-order approximation instead.
  - We believe $\frac{1}{n \cdot \pi_D} \widehat{\nabla}_{\mathrm{ST}}$ induces unwanted instability (please check our paper for more details).

- Theorem 1 is the first that formally established that $\mathrm{D} \leftarrow \mathrm{P}_\theta(\mathrm{D}) - \mathrm{P}_\theta(\mathrm{D}).detach() + \mathrm{D}$ works as a first-order approximation in the multinomial case.

How Straight-Through Works?

Theorem 1: $\mathrm{E}[\widehat{\nabla}_{\mathrm{ST}}]$ is a first-order approximation of ∇, where $\widehat{\nabla}_{\mathrm{ST}} = \frac{\partial f(D)}{\partial D} \cdot \frac{\partial P_\theta(D)}{\partial \theta}$

$$\nabla = \sum_{\mathrm{D}} f(\mathrm{D}) \cdot \frac{\partial P_\theta(\mathrm{D})}{\partial \theta}$$

Subtract $\mathrm{E}[f(\mathrm{D})]$ as the baseline

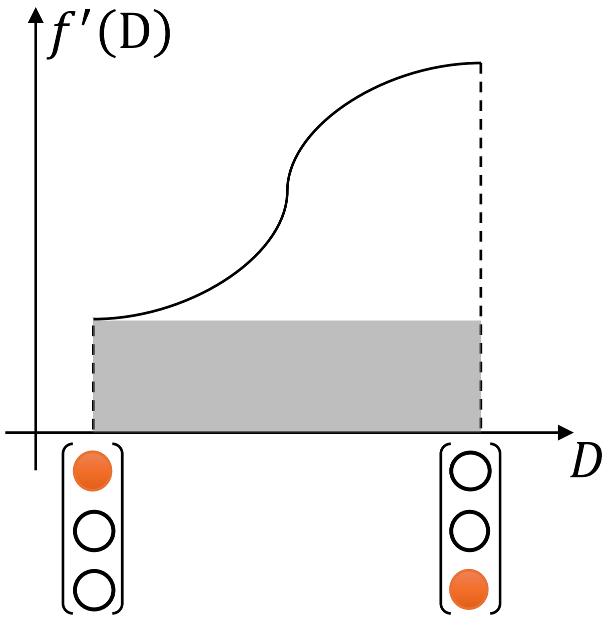
$$\nabla = \sum_i \sum_j P_\theta(I_j) \cdot (f(I_i) - f(I_j)) \cdot \frac{\partial P_\theta(I_i)}{\partial \theta}$$

Approximate $f(I_i) - f(I_j)$ as $\frac{\partial f(I_j)}{\partial I_j}(I_i - I_j)$

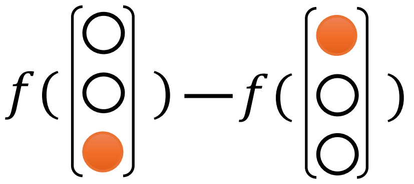
$$\mathrm{E}[\widehat{\nabla}_{\mathrm{ST}}] = \mathrm{E}[\frac{\partial f(D)}{\partial D} \cdot \frac{\partial P_\theta(D)}{\partial \theta}]$$

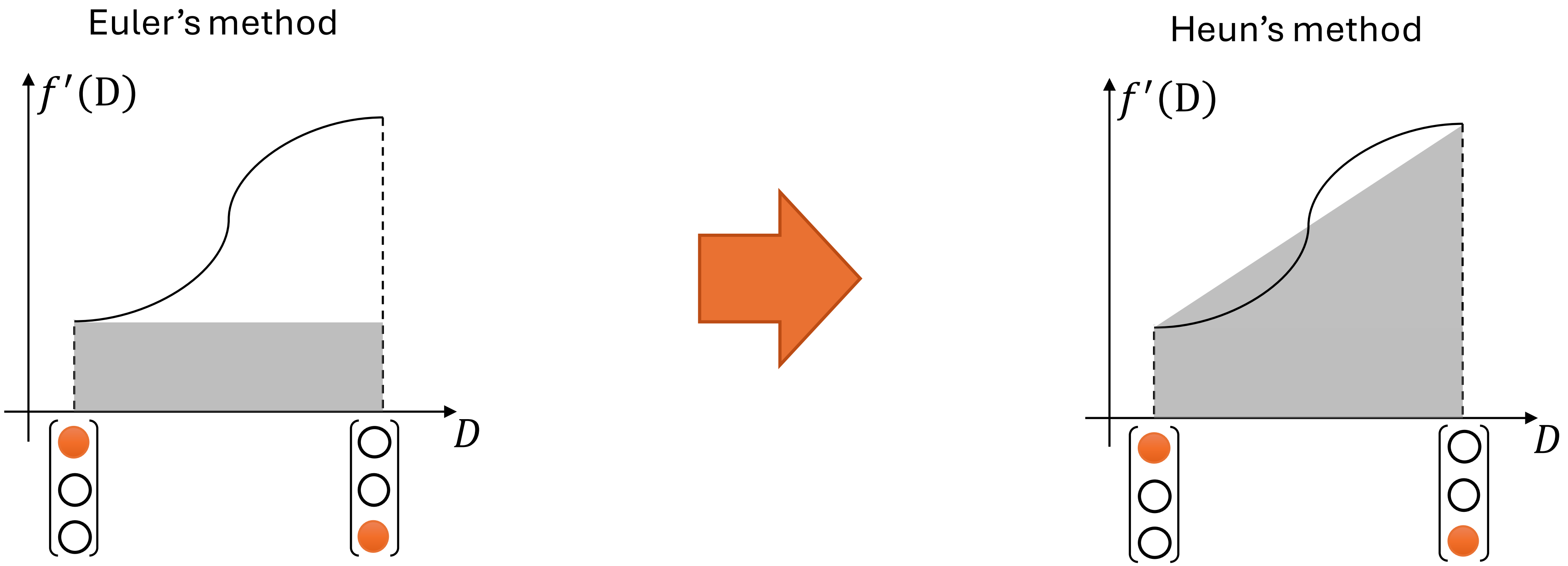
How Straight-Through Works?

- We show that Straight-Through works as $f\left(\begin{bmatrix}\circ\\\circ\\\bullet\end{bmatrix}\right)-f\left(\begin{bmatrix}\bullet\\\circ\\\circ\end{bmatrix}\right) \approx f'\left(\begin{bmatrix}\bullet\\\circ\\\circ\end{bmatrix}\right)\cdot\left(\begin{bmatrix}\circ\\\circ\\\bullet\end{bmatrix}-\begin{bmatrix}\bullet\\\circ\\\circ\end{bmatrix}\right)$
- This approximation has been known as the Euler's method, a first-order numerical ODE solver



How to Improve Straight-Through?

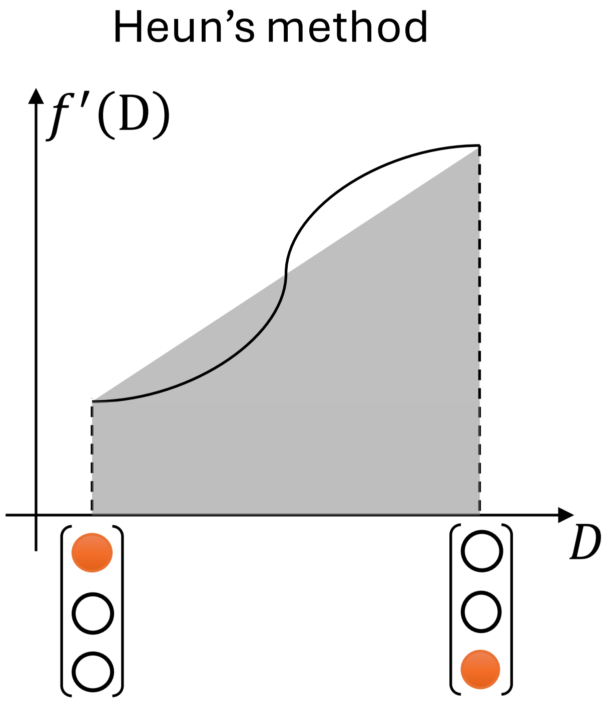
- Approximate  better



From Euler to Heun

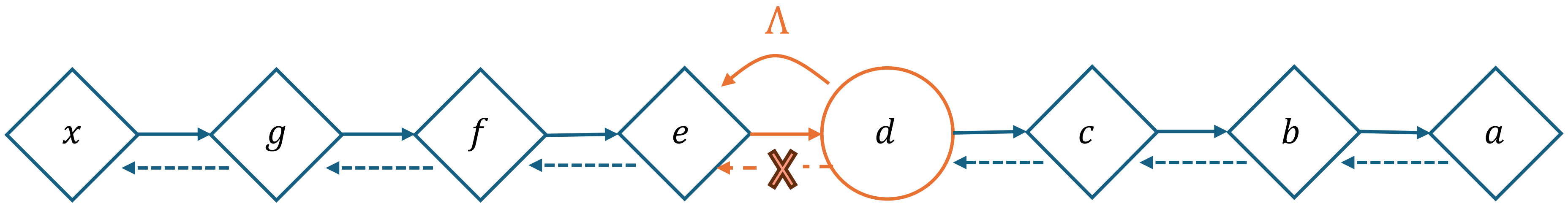
Heun's method approximate $f\left(\begin{bmatrix}\circ\\ \circ\\ \bullet\end{bmatrix}\right) - f\left(\begin{bmatrix}\bullet\\ \circ\\ \circ\end{bmatrix}\right) \approx \frac{1}{2}\cdot\left(f'\left(\begin{bmatrix}\bullet\\ \circ\\ \circ\end{bmatrix}\right) + f'\left(\begin{bmatrix}\circ\\ \circ\\ \bullet\end{bmatrix}\right)\right)\cdot\left(\begin{bmatrix}\circ\\ \circ\\ \bullet\end{bmatrix} - \begin{bmatrix}\bullet\\ \circ\\ \circ\end{bmatrix}\right)$

While Euler's method achieves first-order accuracy, Heun's method achieves *second-order accuracy without requiring second-order derivatives*.



ReinMax

We propose ReinMax to bridge discrete and back-propagation. It achieves second-order accuracy with little computation overheads.



$$\frac{\partial a}{\partial x} = \frac{\partial a}{\partial b} \cdot \frac{\partial b}{\partial c} \cdot \frac{\partial c}{\partial d} \cdot \cancel{\frac{\partial d}{\partial e}} \cdot \frac{\partial d}{\partial f} \cdot \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial x}$$

$$\frac{\partial a}{\partial x} \approx \frac{\partial a}{\partial b} \cdot \frac{\partial b}{\partial c} \cdot \frac{\partial c}{\partial d} \cdot \Lambda \cdot \frac{\partial d}{\partial f} \cdot \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial x}$$

Algorithm 1: ST.

Input: θ: softmax input, τ: temperature.
Output: D: one-hot samples.

$\pi_0 \leftarrow \text{softmax}(\theta)$
$D \leftarrow \text{sample\_one\_hot}(\pi_0)$
$\pi_1 \leftarrow \text{softmax}_\tau(\theta)$
`/* stop_gradient(·) duplicates its input and detaches it from backpropagation. */`
$D \leftarrow \pi_1 - \text{stop\_gradient}(\pi_1) + D$
return D

Algorithm 2: ReinMax.

Input: θ: softmax input, τ: temperature.
Output: D: one-hot samples.

$\pi_0 \leftarrow \text{softmax}(\theta)$
$D \leftarrow \text{sample\_one\_hot}(\pi_0)$
$\pi_1 \leftarrow \frac{D + \text{softmax}_\tau(\theta)}{2}$
$\pi_1 \leftarrow \text{softmax}(\text{stop\_gradient}(\ln(\pi_1) - \theta) + \theta)$
$\pi_2 \leftarrow 2 \cdot \pi_1 - \frac{1}{2} \cdot \pi_0$
$D \leftarrow \pi_2 - \text{stop\_gradient}(\pi_2) + D$
return D

```
pip install reinmax
```

```diff
from reinmax import reinmax
...
- y_hard = one_hot_multinomial(logits.softmax())
- y_soft_tau = (logits/tau).softmax()
- y_hard = y_soft_tau - y_soft_tau.detach() + y_hard
+ y_hard, y_soft = reinmax(logits, tau)
...
```

Outline

- Background
- Gradient Approximation: a Numerical ODE Perspective
- Discussions and Experiments

Discussions and Experiments

- Discussions and Experiments
  - Effectiveness of ReinMax
  - Comparisons with REINFORCE-style algorithms
  - Impact of Temperature Scaling
  - Downstream Applications
  - Efficiency

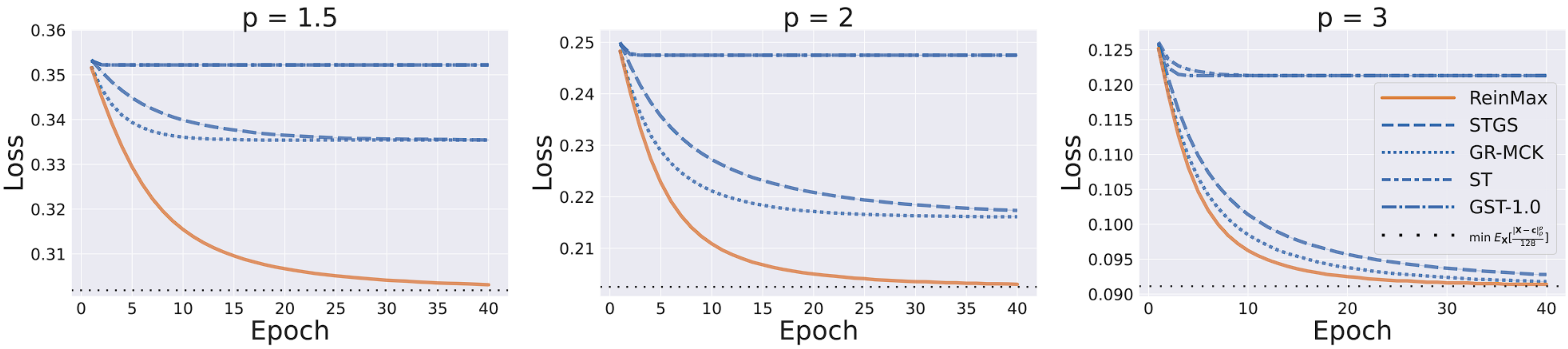
Effectiveness of ReinMax

Major Baselines

1. Straight-Through Gumbel-Softmax *(STGS)*
2. Straight-Through *(ST)*
3. Rao-Blackwellizing Gumbel-Softmax Straight-Through *(GR-MCK; ICLR'21)*
4. Gapped Straight-Through *(GST-1.0; ICML'22)*

Polynomial Programming

$$\min_{\theta} E_{D\sim P_\theta} \frac{|D-c|_p^p}{128} \text{ where } \theta \in R^{128\times 2},\ D \in \{0,1\}^{128}, \text{ and } D_i \overset{\text{iid}}{\sim} \text{softmax}(\theta_i)$$



ListOps and MNIST-VAE

Table 1: Performance on ListOps.

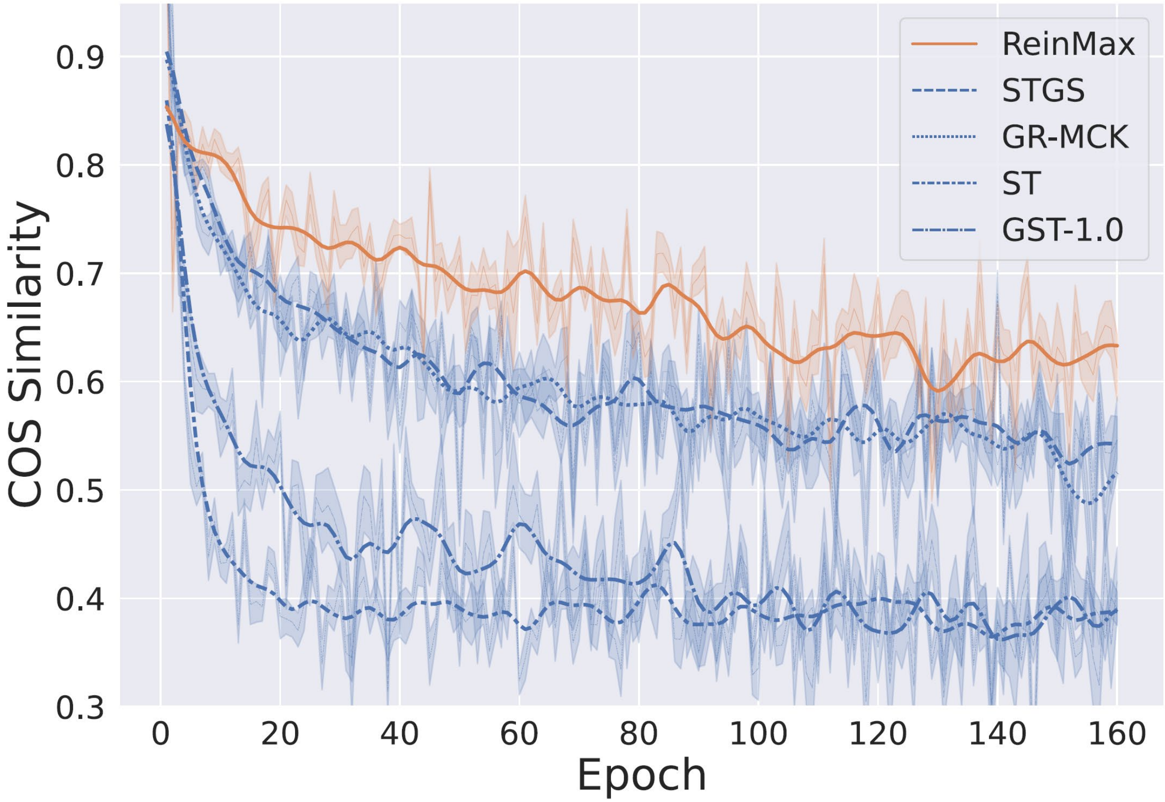
	STGS	GR-MCK	GST-1.0	ST	ReinMax
Valid Accuracy	66.95±3.05	66.53±0.58	66.28±0.52	66.51±0.76	**67.65±1.25**
Test Accuracy	67.30±2.50	66.53±0.86	66.30±0.62	66.26±0.48	**68.07±1.18**

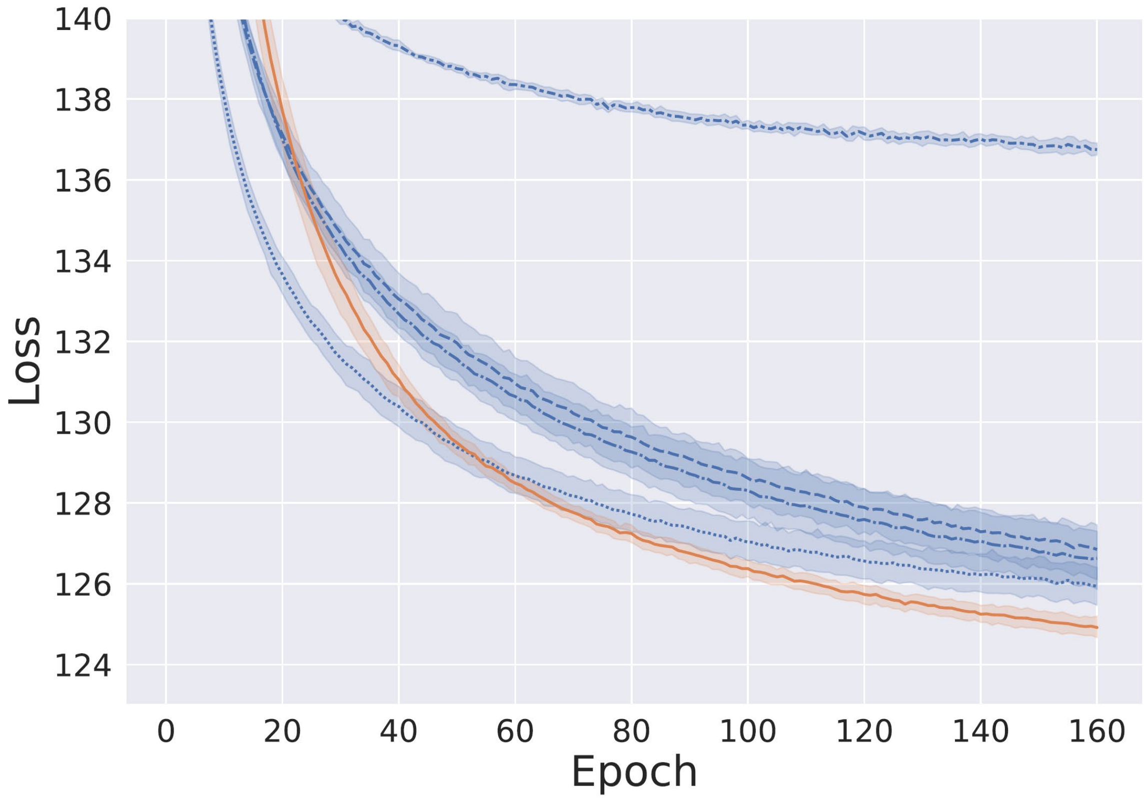
Table 2: Training –ELBO on MNIST ($N \times M$ refers to N categorical dim. and M latent dim.).

	AVG	8×4	4×24	8×16	16×12	64×8	10×30
STGS	105.20	126.85±0.85	101.32±0.43	99.32±0.33	100.09±0.32	104±0.41	99.63±0.63
GR-MCK	107.06	125.94±0.71	99.96±0.25	99.58±0.31	102.54±0.48	112.34±0.48	102.02±0.18
GST-1.0	104.25	126.35±1.24	101.49±0.44	98.29±0.66	98.12±0.57	102.53±0.57	98.64±0.33
ST	116.72	135.53±0.31	112.03±0.03	112.94±0.32	113.31±0.43	113.90±0.28	112.63±0.34
ReinMax	**103.21**	**124.66±0.88**	**99.77±0.45**	**97.70±0.39**	**98.06±0.53**	**100.71±0.70**	**98.37±0.44**

MNIST-VAE with 4 latent dims and 8 categorical dims

$Cos(\nabla, \widehat{\nabla}_{\text{method}})$





Discussions and Experiments

- Discussions and Experiments
  - Effectiveness of ReinMax
  - Comparisons with REINFORCE-style algorithms
  - Impact of Temperature Scaling
  - Downstream Applications
  - Efficiency

ReinMax v.s. REINFORCE

$$f(\mathrm{D}) \cdot \frac{\partial \log P_\theta(\mathrm{D})}{\partial \theta}$$

REIFORCE-style algorithms excel as they provide unbiased gradient estimations but may fall short in complex scenarios, since they only utilize the zero-order information.

$f(\mathrm{D})$ a scalar

$$\frac{\partial f(\mathrm{D})}{\partial \mathrm{D}} \cdot \Lambda \cdot \frac{\partial P_\theta(\mathrm{D})}{\partial \theta}$$

ReinMax, using more information, handles challenging scenarios better. Meanwhile, as a consequence of its estimation bias, ReinMax leads to slower convergence in some simple scenarios.

$\frac{\partial f(\mathrm{D})}{\partial D}$ a vector

ReinMax v.s. REINFORCE

RODEO: Gradient Estimation with Discrete Stein Operators

Table 6: Train $-$ELBO of 2×200 VAE on MNIST, Fashion-MNIST, and Omniglot. * Baseline results are referenced from Shi et al. (2022). K refers to the number of evaluations.

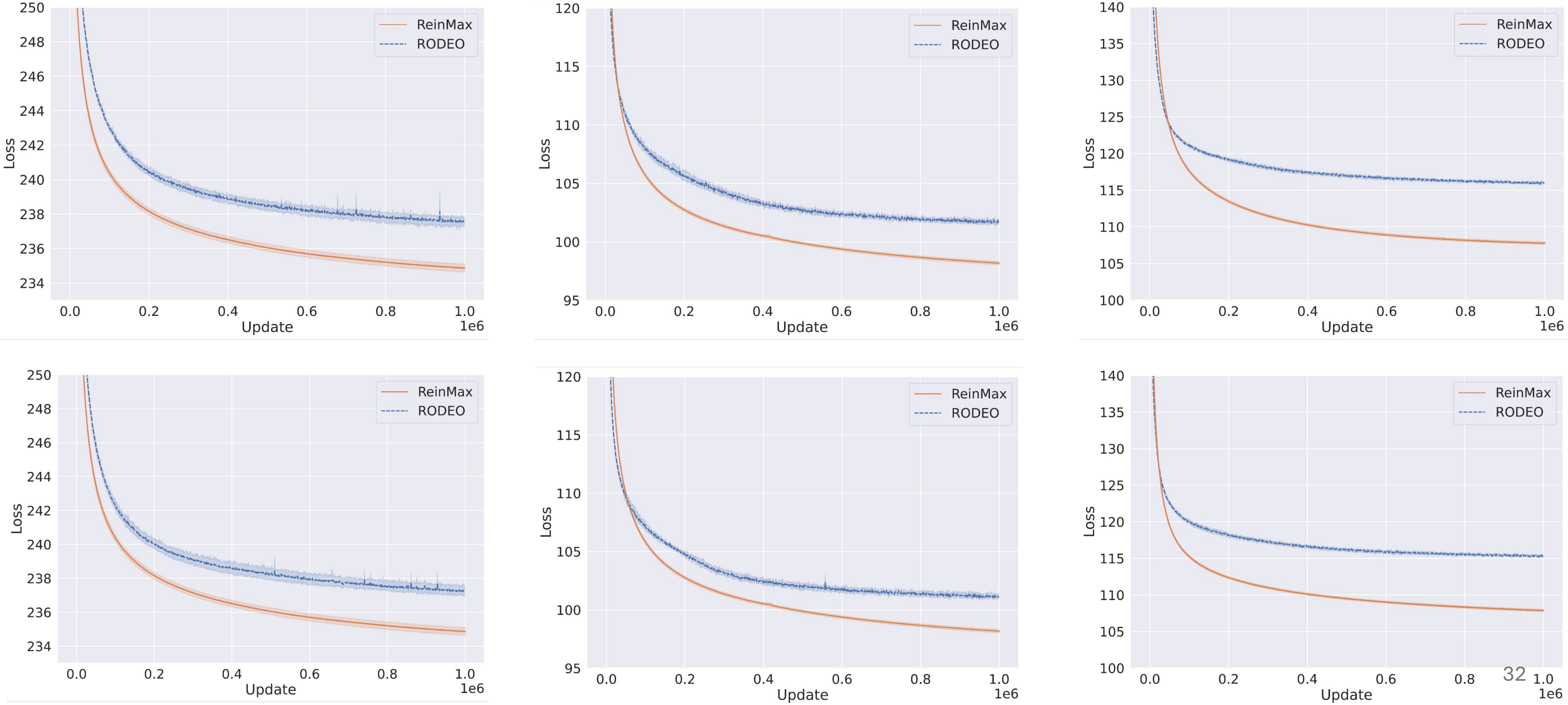
		RELAX*	ARMS*	DisARM*	Double CV*	RODEO*	ReinMax
K=3	MNIST	101.99±0.04	100.84±0.14	/	100.94±0.09	100.46±0.13	**97.83±0.36**
	Fashion-MNIST	237.74±0.12	237.05±0.12	/	237.40±0.11	236.88±0.12	**234.53±0.42**
	Omniglot	115.70±0.08	115.32±0.07	/	115.06±0.12	115.01±0.05	**107.51±0.42**
K=2	MNIST	/	/	102.75±0.08	102.14±0.06	101.89±0.17	**98.05±0.29**
	Fashion-MNIST	/	/	237.68±0.13	237.55±0.16	237.44±0.09	**234.86±0.33**
	Omniglot	/	/	116.50±0.04	116.39±0.10	115.93±0.06	**107.79±0.27**

Bernoulli VAE

Fashion MNIST | MNIST | Omniglot

K=2

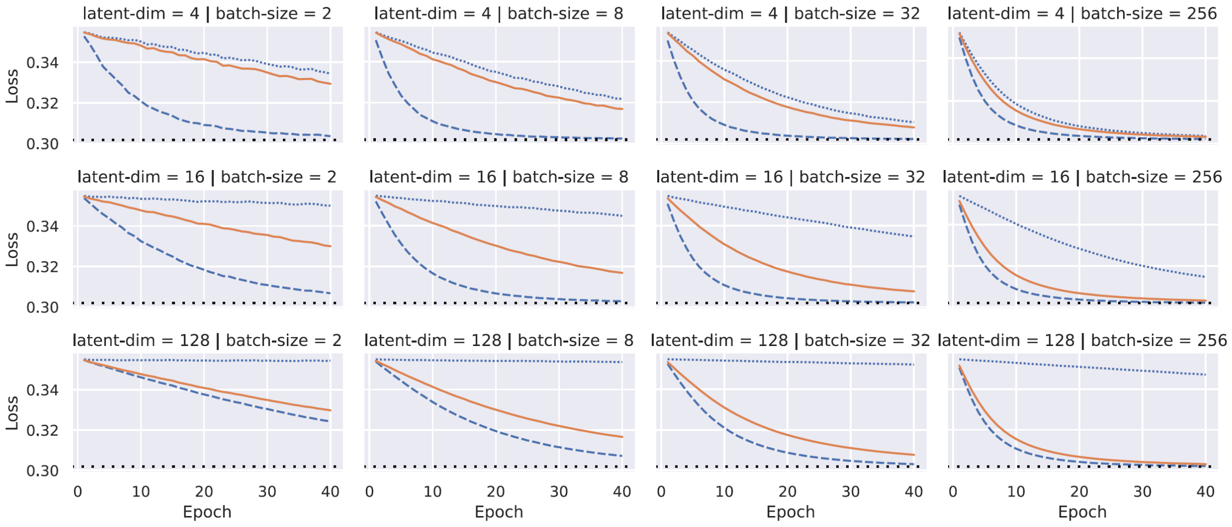
K=3



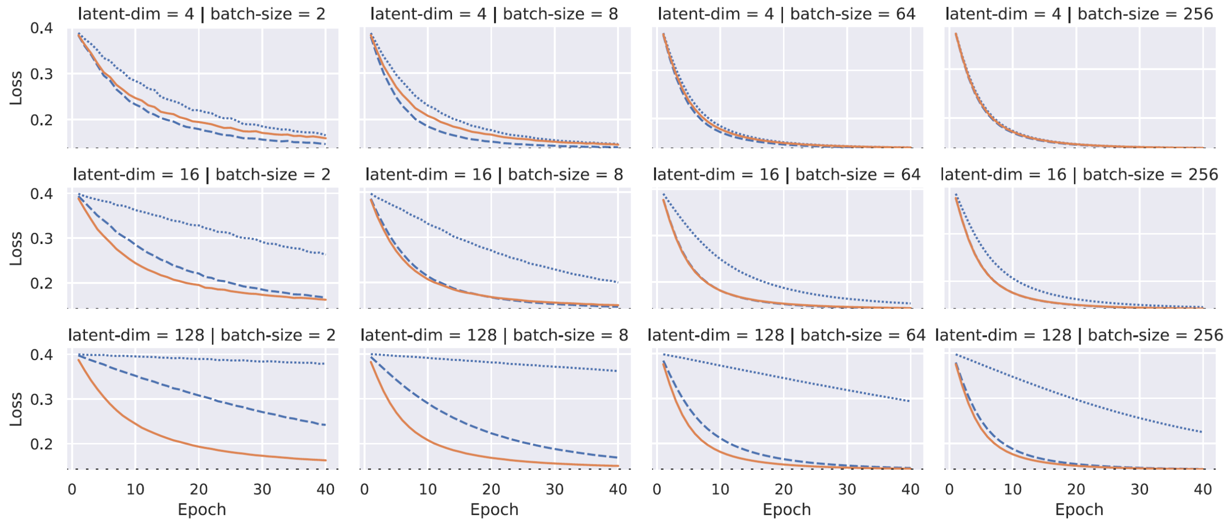
ReinMax v.s. REINFORCE

$$\min_{\theta} E_{D\sim P_\theta} \frac{|D-c|_p^p}{L} \text{ where } \theta \in R^{L\times 2},\ D \in \{0,1\}^L, \text{ and } D_i \overset{\text{iid}}{\sim} \text{softmax}(\theta_i)$$

$$c = [0.45, 0.45, \cdots, 0.45]$$



$$c = [\frac{0.5}{L}, \frac{1.5}{L}, \cdots, \frac{L-0.5}{L}]$$



ReinMax RODEO REINFORCE

ReinMax v.s. REINFORCE

$$f(\mathrm{D}) \cdot \frac{\partial \log P_\theta(\mathrm{D})}{\partial \theta}$$

REIFORCE-style algorithms excel as they provide unbiased gradient estimation but may fall short in complex scenarios, since they only utilize the zero-order information.

$f(\mathrm{D})$ a scalar

$$\frac{\partial f(\mathrm{D})}{\partial \mathrm{D}} \cdot \Lambda \cdot \frac{\partial P_\theta(\mathrm{D})}{\partial \theta}$$

ReinMax, using more information, handles challenging scenarios better. Meanwhile, as a consequence of its estimation bias, ReinMax leads to slower convergence in some simple scenarios.

$\frac{\partial f(\mathrm{D})}{\partial D}$ a vector

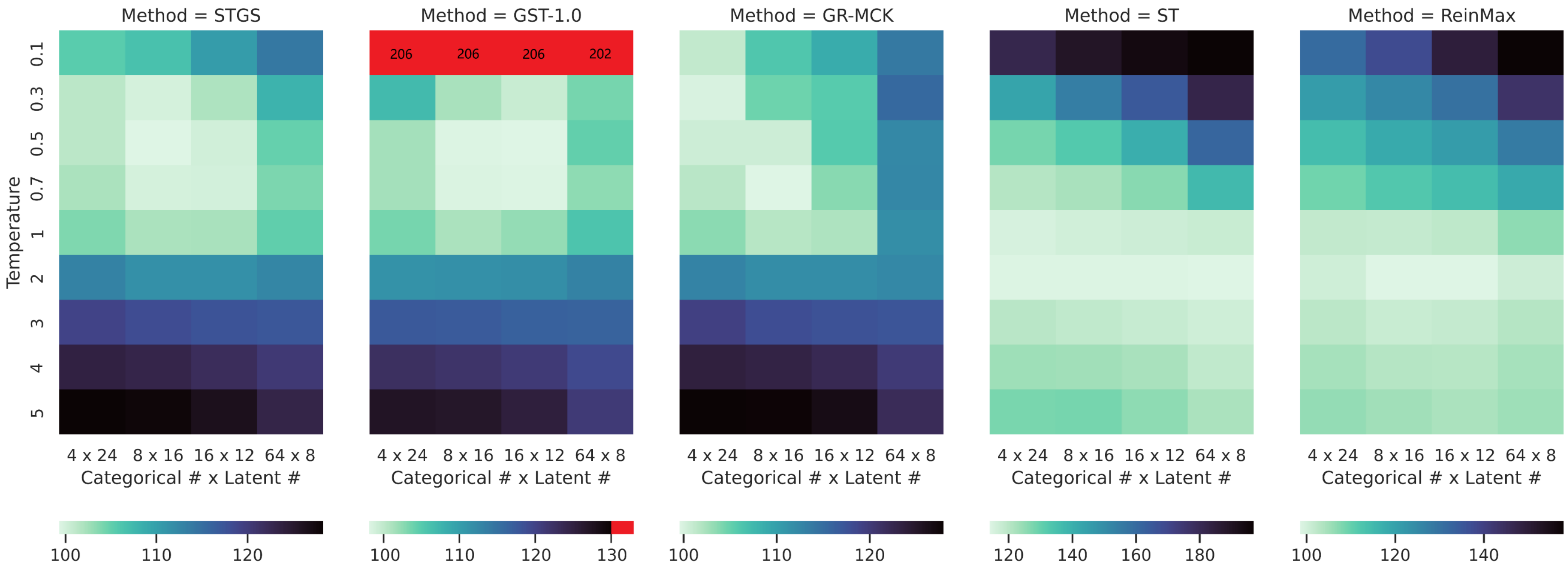
Discussions and Experiments

- Discussions and Experiments
  - Effectiveness of ReinMax
  - Comparisons with REINFORCE-style algorithms
  - Impact of Temperature Scaling
  - Downstream Applications
  - Efficiency

Impact of Temperature Scaling

For STGS, temperature scaling helps to control the bias of the gradient estimation

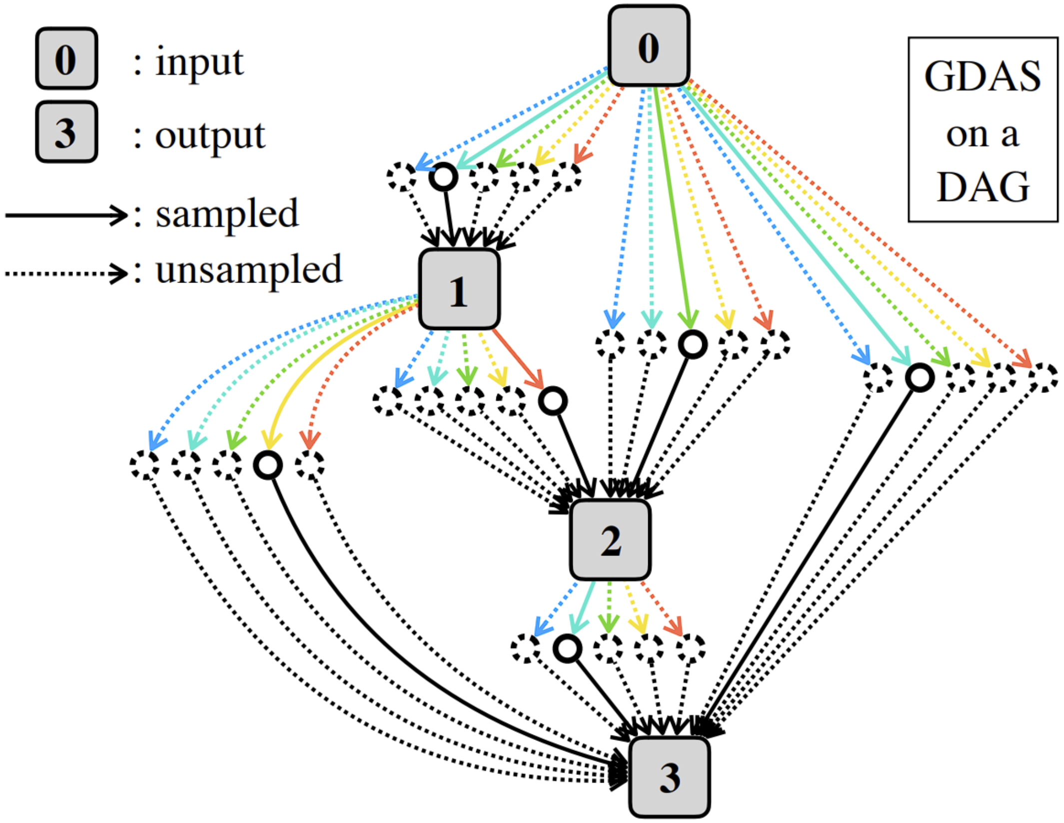
For ST/ReinMax, temperature scaling helps to reduce the variance of the gradient estimation



Discussions and Experiments

- Discussions and Experiments
  - Effectiveness of ReinMax
  - Comparisons with REINFORCE-style algorithms
  - Impact of Temperature Scaling
  - Downstream Applications
  - Efficiency

Differentiable Architecture Search



The task Architecture Search is formulated as edge searching on a DAG:

1. Each node indicates a data
2. Edge of different colors indicate different operations (e.g., pooling / CNNs / …)
3. STGS used as gradient estimator in GDAS

Differentiable Architecture Search

Table 4: Performance on NATS-Bench. * Baseline results are referenced from Dong et al. (2020a).

	CIFAR-10		CIFAR-100		ImageNet-16-120	
	validation	test	validation	test	validation	test
GDAS + STGS*	89.68±0.72	93.23±0.58	68.35±2.71	68.17±2.50	39.55±0.00	39.40±0.00
GDAS + ReinMax	**90.01±0.12**	**93.44±0.23**	**69.29±2.34**	**69.41±2.24**	**41.47±0.79**	**42.03±0.41**

Discussions and Experiments

- Discussions and Experiments
  - Effectiveness of ReinMax
  - Comparisons with REINFORCE-style algorithms
  - Impact of Temperature Scaling
  - Downstream Applications
  - Efficiency

Wall-Clock Efficiency Comparisons

Table 4: Average time cost (per epoch) / peak memory consumption on quadratic programming (QP) and MNIST-VAE. QP is configured to have 128 binary latent variables and 512 samples per batch. MNIST-VAE is configured to have 10 categorical dimensions and 30 latent dimensions.

	ReinMax	ST	STGS	GST-1.0	GR-MCK_{100}	GR-MCK_{300}	GR-MCK_{1000}
QP	0.2s / 6.5Mb	0.2s / 5.0Mb	0.2s / 5.5Mb	0.2s / 8.0Mb	0.8s / 0.3Gb	2.2s / 1Gb	6.6s / 3Gb
MNIST-VAE	5.2s / 13Mb	5.2s / 13Mb	5.2s / 13Mb	5.2s / 13Mb	5.2s / 76Mb	5.2s / 0.2Gb	5.4s / 0.6Gb

Take Aways

ST $(\mathrm{D} \leftarrow \mathrm{P}_{\theta}(\mathrm{D}) - \mathrm{P}_{\theta}(\mathrm{D}).detach() + \mathrm{D})$ works as a first-order approximation to the gradient

ReinMax achieves second-order accuracy without any second-order derivatives, yielding better gradient estimation with negligible computation overheads

Take Aways

ST $(\mathrm{D} \leftarrow \mathrm{P}_{\theta}(\mathrm{D}) - \mathrm{P}_{\theta}(\mathrm{D}).detach() + \mathrm{D})$ works as a first-order approximation to the gradient

ReinMax achieves second-order accuracy without any second-order derivatives, yielding better gradient estimation with negligible computation overheads

Under the guidance of ReinMax, we have a follow-up work on Mixture-of-Expert training. Feel free to stop by!

ReinMax
Poster Section 2

SparseMixer @ WANT Workshop
Oral Section: 11:30->12:00pm
Poster Section: 5:00->5:30pm

MSR is Hiring! FTE & Intern headcounts available! Feel free to reach out :)