A Theorem 3.1

Let us define the first-order approximation of ∇ as $\widehat{\nabla}_{\text{1st-order}} = \sum_i \sum_j \boldsymbol{\pi}_j \frac{\partial f(\boldsymbol{I}_j)}{\partial \boldsymbol{I}_j}(\boldsymbol{I}_i - \boldsymbol{I}_j)\frac{d\,\boldsymbol{\pi}_i}{d\,\boldsymbol{\theta}}$, which approximates $f(\boldsymbol{I}_i) - f(\boldsymbol{I}_j)$ in Equation 6 as $\frac{\partial f(\boldsymbol{I}_j)}{\partial \boldsymbol{I}_j}(\boldsymbol{I}_i - \boldsymbol{I}_j)$.

Theorem 3.1.

$$E[\widehat{\nabla}_{\mathit{ST}}] = \widehat{\nabla}_{\mathit{1st\text{-}order}}.$$

Proof. Based on the definition, we have:

$$\begin{aligned}
\widehat{\nabla}_{\text{1st-order}} &= \sum_i \sum_j \boldsymbol{\pi}_j \frac{\partial f(\boldsymbol{I}_j)}{\partial \boldsymbol{I}_j}(\boldsymbol{I}_i - \boldsymbol{I}_j)\frac{d\,\boldsymbol{\pi}_i}{d\,\boldsymbol{\theta}} \\
&= \sum_j \boldsymbol{\pi}_j \frac{\partial f(\boldsymbol{I}_j)}{\partial \boldsymbol{I}_j} \sum_i \boldsymbol{I}_i \frac{d\,\boldsymbol{\pi}_i}{d\,\boldsymbol{\theta}} - \sum_j \boldsymbol{\pi}_j \frac{\partial f(\boldsymbol{I}_j)}{\partial \boldsymbol{I}_j} \boldsymbol{I}_j \sum_i \frac{d\,\boldsymbol{\pi}_i}{d\,\boldsymbol{\theta}} \qquad (9)
\end{aligned}$$

Since $\sum_i \boldsymbol{\pi}_i = 1$, we have $\sum_i \frac{d\,\boldsymbol{\pi}_i}{d\,\boldsymbol{\theta}} = 0$. Also, since $\boldsymbol{\pi} = \sum_i \boldsymbol{\pi}_i \boldsymbol{I}_i$, we have $\frac{d\,\boldsymbol{\pi}}{d\,\boldsymbol{\theta}} = \sum_i \boldsymbol{I}_i \frac{d\,\boldsymbol{\pi}_i}{d\,\boldsymbol{\theta}}$. Thus, together with Equation 9, we have:

$$\begin{aligned}
\widehat{\nabla}_{\text{1st-order}} &= \sum_j \boldsymbol{\pi}_j \frac{\partial f(\boldsymbol{I}_j)}{\partial \boldsymbol{I}_j} \sum_i \boldsymbol{I}_i \frac{d\,\boldsymbol{\pi}_i}{d\,\boldsymbol{\theta}} \\
&= E[\frac{\partial f(\boldsymbol{D})}{\partial \boldsymbol{D}} \frac{d\,\boldsymbol{\pi}}{d\,\boldsymbol{\theta}}] = E[\widehat{\nabla}_{\text{ST}}].
\end{aligned}$$

□

B Theorem 3.2

Theorem 3.2.

$$E[\widehat{\nabla}_{\mathit{ReinMax}}] = \widehat{\nabla}_{\mathit{2rd\text{-}order}}.$$

Proof. Here, we aim to proof, $\forall k \in [1, n]$, we have $E[\widehat{\nabla}_{\text{ReinMax},k}] = \widehat{\nabla}_{\text{2rd-order},k}$. As defined in Equation 8, we have

$$\begin{aligned}
\widehat{\nabla}_{\text{2rd-order},k} &= \sum_i \sum_j \frac{\boldsymbol{\pi}_j}{2}(\frac{\partial f(\boldsymbol{I}_j)}{\partial \boldsymbol{I}_j} + \frac{\partial f(\boldsymbol{I}_i)}{\partial \boldsymbol{I}_i})(\boldsymbol{I}_i - \boldsymbol{I}_j)\frac{d\,\boldsymbol{\pi}_i}{d\,\boldsymbol{\theta}_k} \\
&= \sum_i \sum_j \frac{\boldsymbol{\pi}_j \boldsymbol{\pi}_i(\delta_{ik} - \boldsymbol{\pi}_k)}{2}(\frac{\partial f(\boldsymbol{I}_j)}{\partial \boldsymbol{I}_j} + \frac{\partial f(\boldsymbol{I}_i)}{\partial \boldsymbol{I}_i})(\boldsymbol{I}_i - \boldsymbol{I}_j) \\
&= \sum_j \frac{\boldsymbol{\pi}_j \boldsymbol{\pi}_k}{2}(\frac{\partial f(\boldsymbol{I}_j)}{\partial \boldsymbol{I}_j} + \frac{\partial f(\boldsymbol{I}_k)}{\partial \boldsymbol{I}_k})(\boldsymbol{I}_k - \boldsymbol{I}_j) \\
&= \frac{\boldsymbol{\pi}_k}{2}\frac{\partial f(\boldsymbol{I}_k)}{\partial \boldsymbol{I}_k}(\boldsymbol{I}_k - \sum_j \boldsymbol{\pi}_j \boldsymbol{I}_j) + \sum_j \frac{\boldsymbol{\pi}_j \boldsymbol{\pi}_k}{2}\frac{\partial f(\boldsymbol{I}_j)}{\partial \boldsymbol{I}_j}(\boldsymbol{I}_k - \boldsymbol{I}_j) \\
&= \frac{1}{2}E[\delta_{\boldsymbol{D}k}\frac{\partial f(\boldsymbol{D})}{\partial \boldsymbol{D}}(\boldsymbol{I}_{\boldsymbol{D}} - \sum_j \boldsymbol{\pi}_j \boldsymbol{I}_j)] + \frac{1}{2}E[\boldsymbol{\pi}_k \frac{\partial f(\boldsymbol{D})}{\partial \boldsymbol{D}}(\boldsymbol{I}_k - \boldsymbol{I}_{\boldsymbol{D}})] \\
&= \frac{1}{2}E[\frac{\partial f(\boldsymbol{D})}{\partial \boldsymbol{D}}(\boldsymbol{\pi}_k(\boldsymbol{I}_k - \boldsymbol{I}_{\boldsymbol{D}}) + \delta_{\boldsymbol{D}k}(\boldsymbol{I}_{\boldsymbol{D}} - \sum_i \boldsymbol{\pi}_i \boldsymbol{I}_i))] \qquad (10)
\end{aligned}$$

At the same time, based on the definition of $\widehat{\nabla}_{\text{ReinMax}}$, we have:

$$\begin{aligned}
E[\widehat{\nabla}_{\text{ReinMax},k}] &= E[\frac{\partial f(\boldsymbol{D})}{\partial \boldsymbol{D}}(2 \cdot \frac{\boldsymbol{\pi}_k + \delta_{\boldsymbol{D}k}}{2}(\boldsymbol{D}_k - \sum_i \frac{\boldsymbol{\pi}_i + \delta_{\boldsymbol{D}k}}{2}\boldsymbol{I}_i) - \frac{\boldsymbol{\pi}_k}{2}(\boldsymbol{D}_k - \sum_i \boldsymbol{\pi}_i \boldsymbol{I}_i))] \\
&= \frac{1}{2}E[\frac{\partial f(\boldsymbol{D})}{\partial \boldsymbol{D}}(\boldsymbol{\pi}_k(\boldsymbol{I}_k - \boldsymbol{I}_{\boldsymbol{D}}) + \delta_{\boldsymbol{D}k}(\boldsymbol{I}_k - \sum_i \boldsymbol{\pi}_i \boldsymbol{I}_i))] \qquad (11)
\end{aligned}$$

Since $\delta_{\boldsymbol{D}k}(\boldsymbol{I}_k - \sum_i \boldsymbol{\pi}_i \boldsymbol{I}_i) = \delta_{\boldsymbol{D}k}(\boldsymbol{I}_{\boldsymbol{D}} - \sum_i \boldsymbol{\pi}_i \boldsymbol{I}_i)$, together with Equation 10 and 11, we have:

$$E[\widehat{\nabla}_{\text{ReinMax},k}] = \widehat{\nabla}_{\text{2rd-order},k}$$

$\square$

C Remark 4.1

Remark 3.1. *When $\sum_i \boldsymbol{\phi}_i f(\boldsymbol{I}_i)$ is used as the baseline and $f(\boldsymbol{I}_i) - f(\boldsymbol{I}_j)$ is approximated as $\frac{\partial f(\boldsymbol{I}_j)}{\partial \boldsymbol{I}_j}(\boldsymbol{I}_i - \boldsymbol{I}_j)$, we mark the resulting first-order approximation of ∇ as $\widehat{\nabla}_{\textit{1st-order-avg-baseline}}$. Then, we have:*

$$E[\frac{\boldsymbol{\phi}_{\boldsymbol{D}}}{\boldsymbol{\pi}_{\boldsymbol{D}}}\widehat{\nabla}_{\textit{ST}}] = \widehat{\nabla}_{\textit{1st-order-avg-baseline}}$$

Proof. Using $\sum_i \boldsymbol{\phi}_i f(\boldsymbol{I}_i)$ as the baseline, we have:

$$\nabla = \sum_i (f(\boldsymbol{I}_i) - \sum_j \boldsymbol{\phi}_j f(\boldsymbol{I}_j)) \frac{d\,\boldsymbol{\pi}_i}{d\,\boldsymbol{\theta}} = \sum_i \sum_j \boldsymbol{\phi}_j (f(\boldsymbol{I}_i) - f(\boldsymbol{I}_j)) \frac{d\,\boldsymbol{\pi}_i}{d\,\boldsymbol{\theta}}$$

Approximating $f(\boldsymbol{I}_i) - f(\boldsymbol{I}_j)$ as $\frac{\partial f(\boldsymbol{I}_j)}{\partial \boldsymbol{I}_j}(\boldsymbol{I}_i - \boldsymbol{I}_j)$, we have:

$$\begin{aligned}
\widehat{\nabla}_{\text{1st-order-avg-baseline}} &= \sum_i \sum_j \boldsymbol{\phi}_j \frac{\partial f(\boldsymbol{I}_j)}{\partial \boldsymbol{I}_j}(\boldsymbol{I}_i - \boldsymbol{I}_j)\frac{d\,\boldsymbol{\pi}_i}{d\,\boldsymbol{\theta}} \\
&= \sum_j \frac{\boldsymbol{\phi}_j}{\boldsymbol{\pi}_j} \cdot \boldsymbol{\pi}_j \frac{\partial f(\boldsymbol{I}_j)}{\partial \boldsymbol{I}_j} \sum_i \boldsymbol{I}_i \frac{d\,\boldsymbol{\pi}_i}{d\,\boldsymbol{\theta}} \\
&= E[\frac{\boldsymbol{\phi}_{\boldsymbol{D}}}{\boldsymbol{\pi}_{\boldsymbol{D}}}\widehat{\nabla}_{\text{ST}}]
\end{aligned}$$

$\square$

D Remark 4.2

Remark 3.2. *In Equation 8, we approximate $f(\boldsymbol{I}_k) - f(\boldsymbol{I}_i)$ as $\frac{1}{2}(\frac{\partial f(\boldsymbol{I}_i)}{\partial \boldsymbol{I}_i} + \frac{\partial f(\boldsymbol{I}_k)}{\partial \boldsymbol{I}_k})(\boldsymbol{I}_k - \boldsymbol{I}_i)$, and mark the resulting second-order approximation of ∇_k as $\widehat{\nabla}_{\textit{2rd-order-wo-baseline},k} = \boldsymbol{\pi}_k \sum_i \boldsymbol{\pi}_i \frac{1}{2}(\frac{\partial f(\boldsymbol{I}_i)}{\partial \boldsymbol{I}_i} + \frac{\partial f(\boldsymbol{I}_k)}{\partial \boldsymbol{I}_k})(\boldsymbol{I}_k - \boldsymbol{I}_i)$, Then, we have:*

$$E[\widehat{\nabla}_{\textit{ReinMax}}] = \widehat{\nabla}_{\textit{2rd-order-wo-baseline}}$$

Proof. Here, we aim to proof, $\forall k \in [1, n]$, we have $E[\widehat{\nabla}_{\text{ReinMax},k}] = \widehat{\nabla}_{\text{2rd-order-wo-baseline},k}$.

$$\begin{aligned}
\widehat{\nabla}_{\text{2rd-order-wo-baseline},k} &= \boldsymbol{\pi}_k \sum_i \boldsymbol{\pi}_i \frac{1}{2}(\frac{\partial f(\boldsymbol{I}_i)}{\partial \boldsymbol{I}_i} + \frac{\partial f(\boldsymbol{I}_k)}{\partial \boldsymbol{I}_k})(\boldsymbol{I}_k - \boldsymbol{I}_i) \\
&= \boldsymbol{\pi}_k \sum_i \boldsymbol{\pi}_i \frac{1}{2} \frac{\partial f(\boldsymbol{I}_i)}{\partial \boldsymbol{I}_i}(\boldsymbol{I}_k - \boldsymbol{I}_i) + \boldsymbol{\pi}_k \sum_i \boldsymbol{\pi}_i \frac{1}{2} \frac{\partial f(\boldsymbol{I}_k)}{\partial \boldsymbol{I}_k}(\boldsymbol{I}_k - \boldsymbol{I}_i) \\
&= E[\frac{\partial f(\boldsymbol{D})}{\partial \boldsymbol{D}} \frac{\boldsymbol{\pi}_k(\boldsymbol{I}_k - \boldsymbol{I}_{\boldsymbol{D}}) + \delta_{\boldsymbol{D}k}(\boldsymbol{I}_k - \sum_i \boldsymbol{\pi}_i \boldsymbol{I}_i)}{2}] = E[\widehat{\nabla}_{\text{ReinMax},k}]
\end{aligned}$$

$\square$

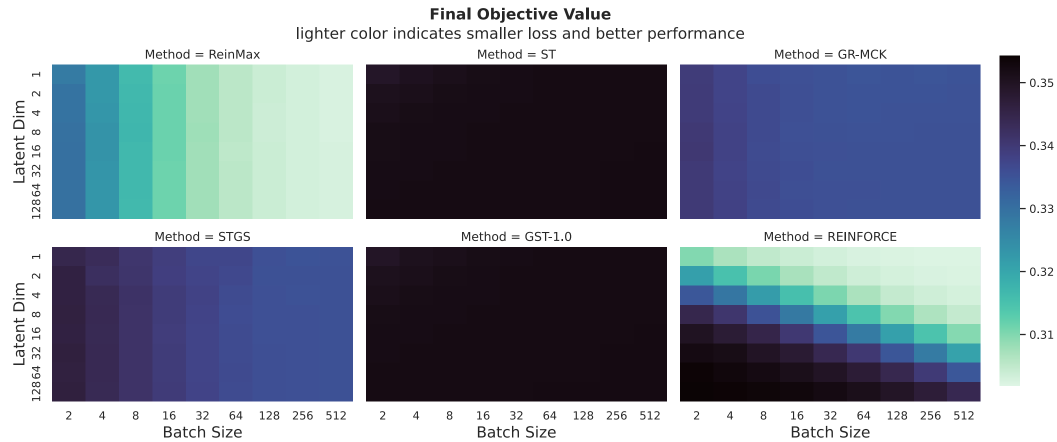

Figure 7: Polynomial programming loss after 40 epochs, with different batch sizes and random variable counts (L), i.e., $\min_{\boldsymbol{\theta}} E[\frac{\|\boldsymbol{X}-\boldsymbol{c}\|_{1.5}^{1.5}}{L}]$, where $\boldsymbol{\theta} \in \mathcal{R}^{L\times 2}, \boldsymbol{X} \in \{0,1\}^L$, and $\boldsymbol{X}_i \overset{\text{iid}}{\sim}$ Multinomial(softmax($\boldsymbol{\theta}_i$)). More details are elaborated in Section 6.

E Forward Euler Method and Heun's Method

For simplicity, we consider a simple function $g(x) : \mathcal{R} \to \mathcal{R}$ that is three times differentiable on $[t_0, t_1]$. Now, we proceed to a simple introduction to approximate $\int_{t_0}^{t_1} g'(x)dx$ with the Forward Euler Method and the Heun's Method. For a detailed introduction to numerical ODE methods, please refer to Ascher & Petzold (1998).

Forward Euler Method. Here, we approximate $g(t_1)$ with the first-order Taylor expansion, i.e., $g(t_1) = g(t_0) + g'(t_0) \cdot (t_1 - t_0) + O((t_1 - t_0)^2)$, then we have $\int_{t_0}^{t_1} g'(x)dx \approx g'(t_0)(t_1 - t_0)$. Since we used the first-order Taylor expansion, this approximation has first-order accuracy.

Heun's Method. First, we approximate $g(t_1)$ with the second-order Taylor expansion:

$$g(t_1) = g(t_0) + g'(t_0) \cdot (t_1 - t_0) + \frac{g''(t_0)}{2} \cdot (t_1 - t_0)^2 + O((t_1 - t_0)^3). \tag{12}$$

Then, we show that we can match this approximation by combining the first-order derivatives of two samples. Taylor expanding $g'(t_1)$ to the first-order, we have:

$$g'(t_1) = g'(t_0) + g''(t_0) \cdot (t_1 - t_0) + O((t_1 - t_0)^2)$$

Therefore, we have:

$$g(t_0) + \frac{g'(t_0) + g'(t_1)}{2}(t_1 - t_0) = g(t_0) + g'(t_0) \cdot (t_1 - t_0) + \frac{g''(t_0)}{2} \cdot (t_1 - t_0)^2 + O((t_1 - t_0)^3).$$

It is easy to notice that the right-hand side of the above equation matches the second-order Taylor expansion of $g(t_1)$ as in Equation 12. Therefore, the above approximation (i.e., approximating $g(t_1) - g(t_0)$ as $\frac{g'(t_0)+g'(t_1)}{2}(t_1 - t_0)$) has second-order accuracy.

Connection to $f(\boldsymbol{I}_i) - f(\boldsymbol{I}_j)$ in Equation 6. By setting $g(x) = f(x \cdot \boldsymbol{I}_i + (1-x) \cdot \boldsymbol{I}_j)$, we have $g(1) - g(0) = f(\boldsymbol{I}_i) - f(\boldsymbol{I}_j)$. Then, it is easy to notice that the forward Euler Method approximates $f(\boldsymbol{I}_i) - f(\boldsymbol{I}_j)$ as $\frac{\partial f(\boldsymbol{I}_j)}{\partial \boldsymbol{I}_j}(\boldsymbol{I}_i - \boldsymbol{I}_j)$ and has first-order accuracy. Also, the Heun's Method approximates $f(\boldsymbol{I}_i) - f(\boldsymbol{I}_j)$ as $\frac{1}{2}(\frac{\partial f(\boldsymbol{I}_i)}{\partial \boldsymbol{I}_i} + \frac{\partial f(\boldsymbol{I}_j)}{\partial \boldsymbol{I}_j})(\boldsymbol{I}_i - \boldsymbol{I}_j)$ and has second-order accuracy.

F Experiment Details

F.1 Baselines

Here, we consider four methods as our major baselines:

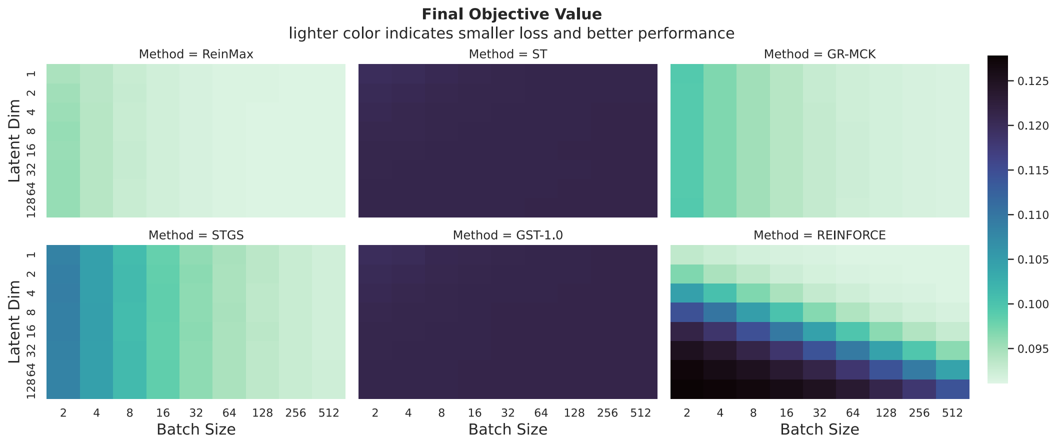

Figure 8: Polynomial programming loss after 40 epochs, with different batch sizes and random variable counts (L), i.e., $\min_{\boldsymbol{\theta}} E[\frac{\|\boldsymbol{X}-\boldsymbol{c}\|_3^3}{L}]$ where $\boldsymbol{\theta} \in \mathcal{R}^{L\times 2}, \boldsymbol{X} \in \{0,1\}^L$, and $\boldsymbol{X}_i \overset{\text{iid}}{\sim} \text{Multinomial}(\text{softmax}(\boldsymbol{\theta}_i))$. More details are elaborated in Section 6.

- Straight-Through (ST; Bengio et al., 2013) backpropagate through the sampling function as if it had been the identity function.
- Straight-Through Gumbel-Softmax (STGS; Jang et al., 2017) integrates the Gumbel reparameterization trick to approximate the gradient.
- Gumbel-Rao Monte Carlo (GR-MCK; Paulus et al., 2021) leverages the Monte Carlo method to reduce the variance introduced by the Gumbel noise in STGS. To obtain the optimal performance for this baseline, we set the number of Monte Carlo samples to 1000 in most experiments. Except in our discussions of efficiency, we set the number of Monte Carlo samples to 100, 300, and 1000 for a more comprehensive comparisons.
- Gapped Straight-Through (GST-1.0; Fan et al., 2022) aims to reduce the variance of STGS and constructs a deterministic term to replace the Monte Carlo samples used in GR-MCK. Here, as suggested in (Fan et al., 2022), we set the gap (a hyper-parameter) as 1.0.

GST-1.0 Performance. Despite GST-1.0 achieving good performance on most settings of MNIST-VAE, it fails to maintain this performance on polynomial programming and unsupervised parsing, as discussed before. At the same time, a different variant of GST (i.e., GST-p) achieves a significant performance boost over GST-1.0 on polynomial programming. However, on MNIST-VAE and ListOps, GST-p achieves an inferior performance. Upon discussing with the author of the GST-1.0, we suggest that this phenomenon is caused by different characteristics of GST-1.0 and GST-p.

This observation verifies our intuition that, without understanding the mechanism of ST, different applications have different preferences on its configurations. Meanwhile, ReinMax achieves consistent improvements in all settings, which greatly simplifies future algorithms developments.

F.2 Hyper-Parameters

Without specifically, we conduct full grid search for all methods in all experiments, and report the best performance (averaged with 10 random seeds on MNIST-VAE and 5 random seeds on ListOps). The hyper-parameter search space is summarized in Table 5.

Table 5: Hyper-parameter search space.

Hyperparameters	Search Space
Optimizer	{Adam(Kingma & Ba, 2015), RAdam(Liu et al., 2020)}
Learning Rate	{0.001, 0.0007, 0.0005, 0.0003}
Temperature	{0.1, 0.3, 0.5, 0.7, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5}

Table 6: Test −ELBO on MNIST. Hyper-parameters are chosen based on Train −ELBO.

	AVG	8×4	4×24	8×16	16×12	64×8	10×30
STGS	106.89	128.09±0.79	103.60±0.45	99.32±0.33	102.49±0.32	106.20±0.46	101.61±0.54
GR-MCK	109.03	127.90±0.71	102.76±0.33	102.12±0.29	104.23±0.65	113.54±0.50	103.62±0.13
GST-1.0	106.85	128.20±1.12	103.95±0.49	101.44±0.32	101.28±0.59	105.44±0.62	100.78±0.44
ST	118.85	137.06±0.51	113.41±0.49	114.25±0.29	114.48±0.56	115.43±0.29	118.46±0.18
ReinMax	**105.74**	**126.89±0.79**	**102.40±0.43**	**100.63±0.41**	**100.85±0.50**	**102.91±0.67**	**100.75±0.50**

Table 7: Test −ELBO on MNIST. Hyper-parameters are chosen based on Test −ELBO.

	AVG	8×4	4×24	8×16	16×12	64×8	10×30
STGS	107.15	128.09±0.79	103.25±0.22	101.44±0.32	102.29±0.39	106.20±0.46	101.61±0.54
GR-MCK	108.87	127.86±0.54	102.40±0.37	101.59±0.22	104.22±0.63	113.54±0.50	103.62±0.13
GST-1.0	106.55	128.03±1.02	103.63±0.24	100.67±0.34	101.04±0.39	105.44±0.62	**100.51±0.37**
ST	118.79	137.05±0.36	113.23±0.43	114.11±0.31	114.48±0.56	115.43±0.29	118.46±0.18
ReinMax	**105.60**	**126.29±0.32**	**102.40±0.43**	**100.45±0.26**	**100.84±0.56**	**102.91±0.68**	100.69±0.48

Table 8: Train −ELBO on MNIST. Hyper-parameters are chosen based on Test −ELBO.

	AVG	8×4	4×24	8×16	16×12	64×8	10×30
STGS	105.31	126.85±0.85	101.81±0.14	99.32±0.33	100.22±0.47	104.02±0.41	99.63±0.63
GR-MCK	107.37	126.53±0.55	100.47±0.31	99.75±0.29	103.11±0.58	112.34±0.48	102.02±0.18
GST-1.0	104.60	126.63±1.16	102.11±0.24	98.40±0.34	98.76±0.41	102.53±0.57	99.14±0.30
ST	117.76	136.75±0.22	112.09±0.50	113.06±0.26	113.31±0.43	113.90±0.28	117.46±0.09
ReinMax	**103.40**	**124.92±0.38**	**99.77±0.45**	**98.06±0.31**	**98.51±0.54**	**100.71±0.70**	**98.40±0.48**

Polynomial Programming. As this problem is relatively simple, we set the learning rate to 0.001 and the optimizer to Adam, and only tune the temperature hyper-parameter.

MNIST-VAE. Following the previous study (Dong et al., 2020, 2021; Fan et al., 2022), we used 2-layer MLP as the encoder and the decoder. We set the hidden state dimension of the first-layer and the second-layer as 512 and 256 for the encoder, and 256 and 512 for the decoder. For our experiments on MNIST-VAE with 32 latent dimensions and 64 categorical dimensions, we set the batch size to 200, training steps to 5×10^5, and activation function to LeakyReLU, in order to be consistent with the literature. For other experiments, we set the batch size to 100, the activation function to ReLU, and training steps to 9.6×10^4 (i.e., 160 epochs).

ListOps. We followed the same setting of Fan et al. (2022), i.e., used the same model configuration as in Choi et al. (2017) and set the maximum sequence length to 100.

F.3 Hardware and Environment Setting

Most experiments (except efficiency comparisons) are conducted on Nvidia P40 GPUs. For efficiency comparisons, we measured the average time cost per batch and peak memory consumption on quadratic programming and MNIST-VAE on the same system with an idle A6000 GPU. Also, to better reflect the efficiency of gradient estimators, we skipped all parameter updates in this experiment.

F.4 Additional Results on Polynomial Programming

Here, we visualized the heat map for polynomial programming with various batch sizes and latent dimensions in Figure 7 (for $p = 1.5$) and Figure 8 (for $p = 3$). We visualized the training curve for polynomial programming with various batch sizes and latent dimensions in Figure 9 (for $p = 1.5$), Figure 10 (for $p = 2$), and Figure 11 (for $p = 3$).

F.5 Additional Results on MNIST-VAE

In our discussions in Section 6, we focused on the training ELBO only. Here, we provide a brief discussion on the test ELBO.

Choosing Hyper-parameter Based on Training Performance. Similar to Table 2, for each method, we select the hyper-parameter based on its training performance. The Test $-$ELBO in this setting is summarized in 6. Despite the model being trained without dropout or other overfitting reduction techniques, ReinMax maintained the best performance in this setting.

Choosing Hyper-parameter Based on Test Performance. We also conduct experiments by selecting hyper-parameters directly based on their test performance. In this setting, the test $-$ELBO is summarized in Table 7, and the training $-$ELBO is summarized in Table 8. ReinMax achieves the best performance in all settings except the test performance of the setting with 10 categorical dimensions and 30 latent dimensions.

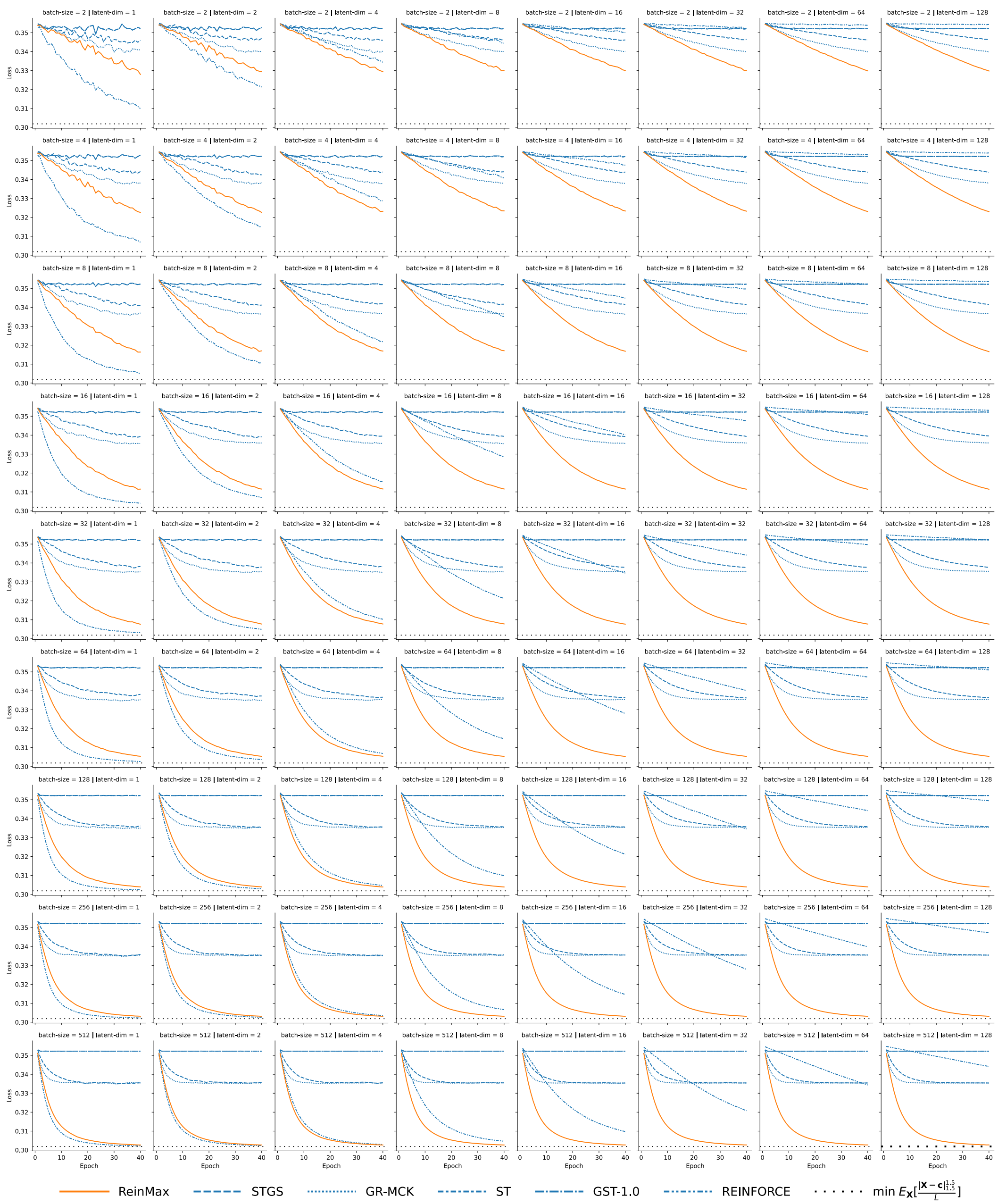

Figure 9: Polynomial programming training curve, with different batch sizes and random variable counts (L), i.e., $\min_{\boldsymbol{\theta}} E[\frac{\|\boldsymbol{X}-\boldsymbol{c}\|_{1.5}^{1.5}}{L}]$, where $\boldsymbol{\theta} \in \mathcal{R}^{L\times 2}, \boldsymbol{X} \in \{0,1\}^L$, and $\boldsymbol{X}_i \overset{\text{iid}}{\sim} \text{Multinomial}(\text{softmax}(\boldsymbol{\theta}_i))$. More details are elaborated in Section 6.

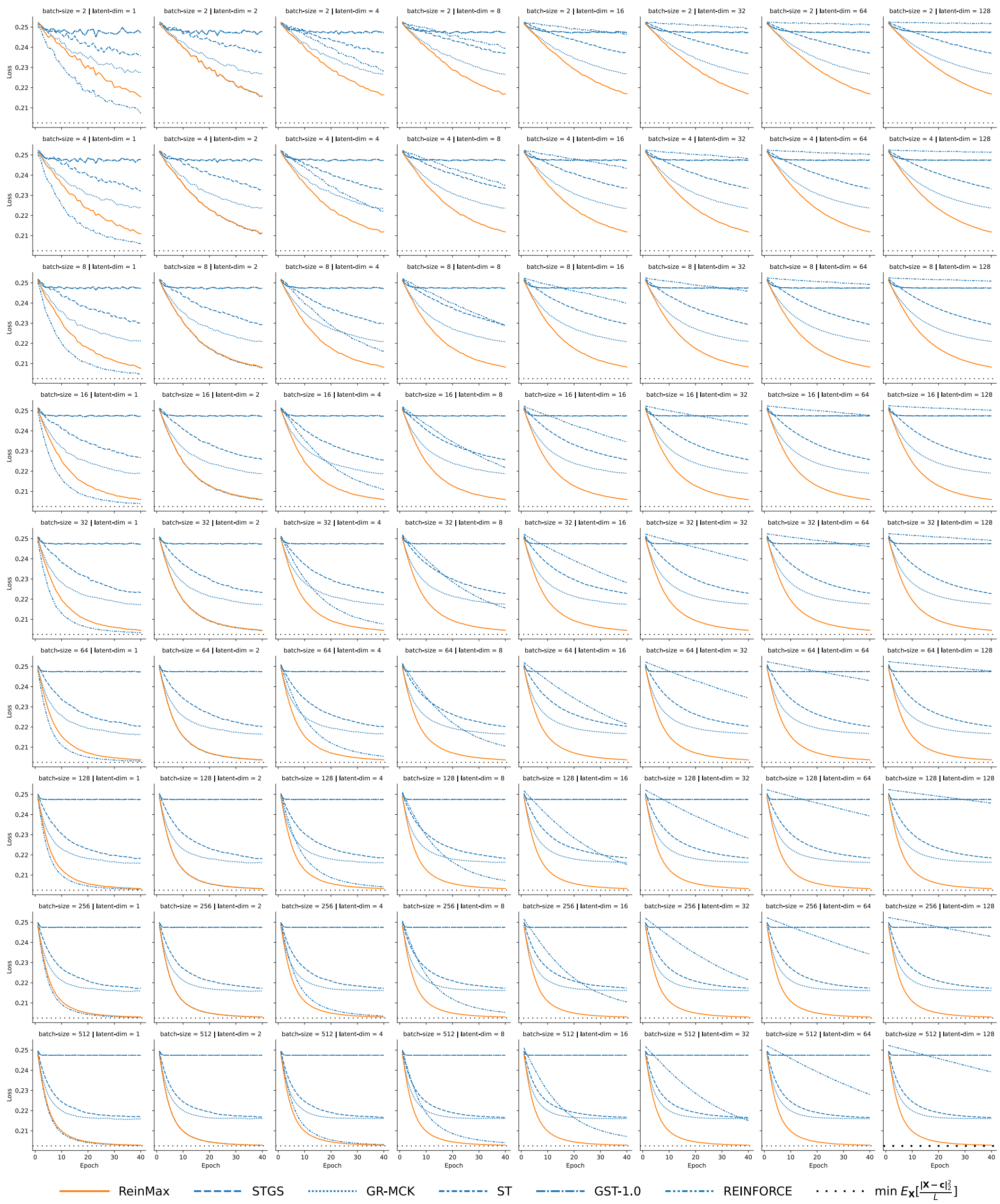

Figure 10: Quadratic programming training curve, with different batch sizes and random variable counts (L), i.e., $\min_{\boldsymbol{\theta}} E[\frac{\|\boldsymbol{X}-\mathbf{c}\|_2^2}{L}]$, where $\boldsymbol{\theta} \in \mathcal{R}^{L\times 2}, \boldsymbol{X} \in \{0,1\}^L$, and $\boldsymbol{X}_i \overset{\text{iid}}{\sim}$ Multinomial(softmax($\boldsymbol{\theta}_i$)). More details are elaborated in Section 6.

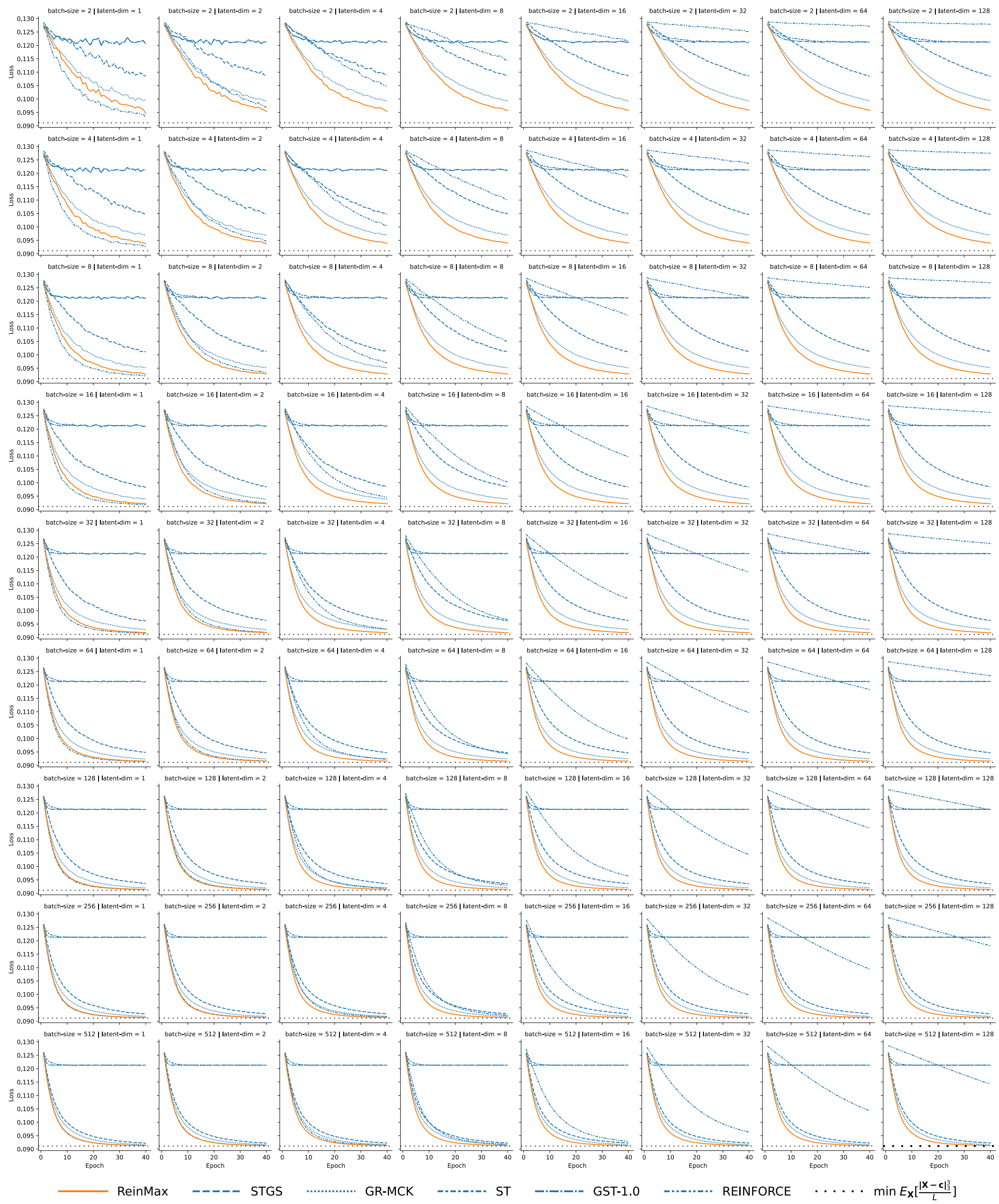

Figure 11: Polynomial programming training curve, with different batch sizes and random variable counts (L), i.e., $\min_{\boldsymbol{\theta}} E[\frac{\|\boldsymbol{X}-\mathbf{c}\|_3^3}{L}]$, where $\boldsymbol{\theta} \in \mathcal{R}^{L\times 2}, \boldsymbol{X} \in \{0,1\}^L$, and $\boldsymbol{X}_i \overset{\text{iid}}{\sim}$ Multinomial(softmax($\boldsymbol{\theta}_i$)). More details are elaborated in Section 6.