Training Discrete Deep Generative Models via Gapped Straight-Through Estimator

Ting-Han Fan [*1] **Ta-Chung Chi** [*2] **Alexander I. Rudnicky** [2] **Peter J. Ramadge** [1]

Abstract

While deep generative models have succeeded in image processing, natural language processing, and reinforcement learning, training that involves discrete random variables remains challenging due to the high variance of its gradient estimation process. Monte Carlo is a common solution used in most variance reduction approaches. However, this involves time-consuming resampling and multiple function evaluations. We propose a Gapped Straight-Through (GST) estimator to reduce the variance without incurring resampling overhead. This estimator is inspired by the essential properties of Straight-Through Gumbel-Softmax. We determine these properties and show via an ablation study that they are essential. Experiments demonstrate that the proposed GST estimator enjoys better performance compared to strong baselines on two discrete deep generative modeling tasks, MNIST-VAE and ListOps.

1. Introduction

Deep generative models (DGM) (Ruthotto & Haber, 2021; Kingma & Welling, 2019; Goodfellow et al., 2020; Rezende & Mohamed, 2015) are deep neural networks that are capable of high-dimensional probability distributions modeling and random samples generation. These properties are especially useful in applications such as image processing (Korshunov & Marcel, 2018; Song et al., 2021), speech processing (Oord et al., 2016), natural language processing (Radford et al., 2019; Chen et al., 2021), and reinforcement learning (Ho & Ermon, 2016; Li et al., 2017). Among these tasks, some of which involve inherently discrete components hence necessitating the need of modeling discrete random variables. For example, structure learning (Nangia & Bowman, 2018), generative text modeling (Yang et al., 2017), multi-agent control (Lowe et al., 2017), and control with discrete/integer variables (Tang & Agrawal, 2020; Fan & Wang, 2021). Training these discrete DGMs remains challenging mainly due to the discrete sampling process, which impedes the direct use of gradient backpropagation. Consequently, designing a high-quality gradient estimation technique for the discrete component becomes the key to success.

Existing gradient estimation techniques for discrete DGMs bifurcate into two paradigms: the REINFORCE estimator (Glynn, 1990; Williams, 1992) and the Straight-Through Gumbel-Softmax (STGS) (Maddison et al., 2017; Jang et al., 2017). The former is unbiased but with high variance, while the latter is of low variance but requires a continuous relaxation during the gradient computation (i.e., h in Eq. (6)). Despite the differences, the Monte Carlo variance reduction technique has been used in both methods; for example, Mnih & Gregor (2014); Mnih & Rezende (2016); Gu et al. (2016) for REINFORCE and Paulus et al. (2021) for STGS. While the variance is reduced, side-effects such as multiple resampling and function evaluations emerge, which are also the main drawbacks that we address.

In this paper, we introduce the Gapped Straight-Through (GST) estimator, a variant of the Straight-Through estimator (Bengio et al., 2013), that adds a careful logit perturbation process. We decide to improve upon the STGS paradigm since it is believed to have low variance and can leave the loss function unmodified (Eq. (2) vs. (5)). First, we show that STGS has a number of properties (§3) essential for good performance (see ablation study in §5.1.1). To our best knowledge, this is a new determination of the key properties that support STGS.

Second, the Gumbel randomness introduced in STGS is sufficient but not necessary for these properties. We show GST satisfies these properties with less randomness. In particular, GST first samples a random category $\boldsymbol{D}$ and then computes the deterministic perturbation given $\boldsymbol{D}$. This has the advantage of variance reduction while avoiding any resampling

[*]Equal contribution [1]Department of Electrical and Computer Engineering, Princeton University, Princeton, NJ, USA [2]Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA, USA. Correspondence to: Ting-Han Fan <tinghanf@princeton.edu>, Ta-Chung Chi <tachungc@andrew.cmu.edu>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

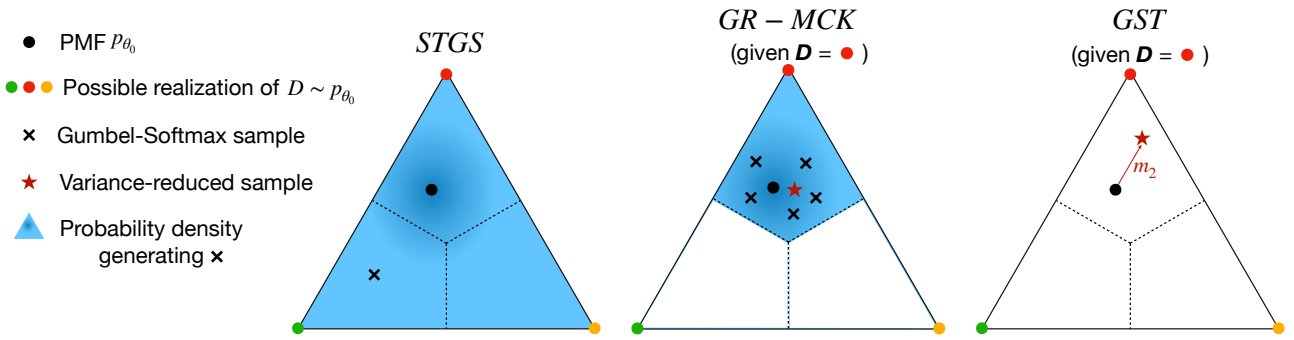

Figure 1. **Different estimators operating on the probability simplex.** The blue shaded region represents the probability density of the Gumbel-Softmax sample in the simplex, where the degree of darkness is proportional to the likelihood. The dashed lines separate three categories (top, left, right). STGS generates soft samples across all categories. GR-MCK (Paulus et al., 2021) chooses a random category $\boldsymbol{D} \sim p_{\theta_0}$ (e.g. top category), generates multiple (e.g. $K = 100$) Gumbel-Softmax samples in $\boldsymbol{D}$'s category (i.e., conditioning on $\boldsymbol{D}$), and averages over them to obtain a variance-reduced one. GST chooses a random category $\boldsymbol{D} \sim p_{\theta_0}$ and applies m_2 to get a variance-reduced soft sample given $\boldsymbol{D}$ (§4.2 and Figure 2). Note that m_2 is deterministic given $\boldsymbol{D}$, so the resulting sample is of low variance. All estimators generate the soft samples and convert them into the hard samples using the Straight-Through trick (e.g. Algorithm 1).

overhead, e.g., the Monte Carlo method (Paulus et al., 2021). Experiments show that GST achieves smaller test losses and variances on MNIST-VAE (Jang et al., 2017; Kingma et al., 2014) and better accuracies on ListOps unsupervised parsing (Nangia & Bowman, 2018). Figure 1 summarizes the difference between our proposed GST and the prior work.

We use bold symbols to denote random variables. For example, $\boldsymbol{D}$, $\boldsymbol{\xi}$, and $\boldsymbol{G}$ are the one-hot random sample, random source, Gumbel$(0, 1)$ random vector, respectively. The subscript i, with or without a bracket, denotes the ith entry of a vector; for example, $[p_\theta]_i$ and $\boldsymbol{G}_i$ are the ith entry of p_θ and $\boldsymbol{G}$, respectively. Lastly, θ denotes the trainable parameters of a neural network (NN), and logit_θ denotes the NN-parameterized vector before the Softmax function.

2. Gradient Estimation for Discrete DGM

Let $\boldsymbol{D}$ be a random variable from a discrete/categorical distribution p_θ parameterized by an NN with parameter θ (we will take $p_\theta = \text{Softmax}_1(\text{logit}_\theta)$ in §2.3). Without loss of generality, suppose $\boldsymbol{D}$ is one-hot and $\mathbb{P}(\boldsymbol{D} = e_i) = [p_\theta]_i$, where $\{e_1, ..., e_N\}$ is the standard basis in $\mathbb{R}^N$. Given an objective function $g : \{e_1, ..., e_N\} \to \mathbb{R}$, we want to minimize the expected value of g over the distribution of $\boldsymbol{D}$:

$$\min_{p_\theta} \mathbb{E}_{\boldsymbol{D}\sim p_\theta}[g(\boldsymbol{D})] = \min_{\theta} \mathbb{E}_{\boldsymbol{D}\sim p_\theta}[g(\boldsymbol{D})]. \tag{1}$$

Since p_θ is parameterized by θ, Eq. (1) needs to be optimized by an (unbiased) estimate of $\nabla_\theta \mathbb{E}[g(\boldsymbol{D})]$, which we will review in this section.

2.1. REINFORCE Estimator

The REINFORCE estimator (Glynn, 1990; Williams, 1992) is defined as follows:

$$\begin{aligned}\nabla_\theta \mathbb{E}[g(\boldsymbol{D})] &= \sum_{i=1}^{N} \nabla_\theta [p_\theta]_i g(e_i) \\ &\overset{(*)}{=} \sum_{i=1}^{N} [p_\theta]_i \frac{\nabla_\theta [p_\theta]_i}{[p_\theta]_i} g(e_i) = \mathbb{E}[\nabla_\theta \log p_\theta(\boldsymbol{D}) g(\boldsymbol{D})],\end{aligned} \tag{2}$$

where $p_\theta(\boldsymbol{D})$ is the probability of $\boldsymbol{D}$; i.e., $p_\theta(\boldsymbol{D}) = [p_\theta]_i$ if $\boldsymbol{D} = i$. The LHS of $(*)$ is unweighted while the RHS is written into an expectation with weights $1/[p_\theta]_i$. The REINFORCE estimator simply takes Eq. (3) as an unbiased estimate of $\nabla_\theta \mathbb{E}[g(\boldsymbol{D})]$.

$$\nabla_\theta \log p_\theta(\boldsymbol{D}) g(\boldsymbol{D}), \quad \boldsymbol{D} \sim p_\theta. \tag{3}$$

While Eq. (3) only requires a single sample from p_θ to establish an unbiased estimate, it is of high variance due to its importance weight $1/[p_\theta]_i$. The variance can be large when there exists small $[p_\theta]_i$ for some $i \in \{1, ..., N\}$.

2.2. Reparameterization and Straight-Through

The major drawback of REINFORCE is that the randomness (p_θ) is coupled with the NN parameters (θ). This motivates the use of reparameterization to decouple the randomness and the parameter. Suppose that there exists a reparameterization of $\boldsymbol{D}$ (e.g. Eq. (7)) , written as $D(\theta, \boldsymbol{\xi})$, such that $D(\theta, \boldsymbol{\xi}) \in \{e_1, ..., e_N\}$ (i.e., one-hot) and that $D(\theta, \boldsymbol{\xi})$ is random with the source of randomness from $\boldsymbol{\xi}$. Under such reparameterization, Eq. (1) is re-written as

$$\min_{\theta} \mathbb{E}_{\boldsymbol{\xi}}[g(D(\theta, \boldsymbol{\xi}))]. \tag{4}$$

The gradient to optimize Eq. (4) becomes $\nabla_\theta \mathbb{E}_{\boldsymbol{\xi}}[g(D(\theta, \boldsymbol{\xi}))]$ with $\nabla_\theta g(D(\theta, \boldsymbol{\xi}))$ being its unbiased estimate:

$$\begin{aligned}\nabla_\theta \mathbb{E}_{\boldsymbol{\xi}}[g(D(\theta, \boldsymbol{\xi}))] &= \nabla_\theta \int f_\Xi(\xi) g(D(\theta, \xi)) d\xi \\ &= \int \nabla_\theta f_\Xi(\xi) g(D(\theta, \xi)) d\xi = \mathbb{E}_{\boldsymbol{\xi}}[\nabla_\theta g(D(\theta, \boldsymbol{\xi}))],\end{aligned} \tag{5}$$

where f_Ξ is the probability density of $\boldsymbol{\xi}$. Compared with Eq. (2), Eq. (5) does not involve an importance weighting procedure thanks to the decoupling of θ and $\boldsymbol{\xi}$. Thus, the gradient estimation from the reparameterization (i.e. $D(\theta, \boldsymbol{\xi})$) is believed to be of lower variance.

The final missing piece is to backpropagate through $D(\theta, \boldsymbol{\xi})$, which typically requires the differentiability w.r.t. θ. However, the discrete nature of $\boldsymbol{D}$ (i.e., $\boldsymbol{D} \in \{e_1, ..., e_N\}$) hinders the differentiability[1]. Fortunately, the Straight-Through estimator (Bengio et al., 2013; Chung et al., 2017), (Hinton et al., 2012)[lecture 15b] helps enforce the discreteness while maintaining differentiability. In particular, let θ be the NN's parameters and $\theta_0 = \text{stop\_grad}(\theta)$ be the parameters in the forward pass but with zero gradient in the backward pass (i.e., $\nabla_\theta \text{stop\_grad}(\Phi(\theta)) = \nabla_\theta \Phi(\theta_0) = 0$ for any $\Phi(\cdot)$). After then, the distribution vector p_{θ_0} is evaluated, and a one-hot vector $\boldsymbol{D} = D(\theta_0, \boldsymbol{\xi})$ is sampled from p_{θ_0}. Note $\boldsymbol{\xi}$ is the randomness of the sampling process and the random variable $\boldsymbol{D}$ is now represented by $D(\theta_0, \boldsymbol{\xi})$. With all the ingredients, the general form of the Straight-Through (ST) estimator can be written as

$$\begin{aligned}&D_{\text{ST}}(\theta, \boldsymbol{\xi}) \\ &= \text{stop\_grad}(D(\theta, \boldsymbol{\xi})) - \text{stop\_grad}(h(\theta, \boldsymbol{\xi})) + h(\theta, \boldsymbol{\xi}) \\ &= D(\theta_0, \boldsymbol{\xi}) - h(\theta_0, \boldsymbol{\xi}) + h(\theta, \boldsymbol{\xi}),\end{aligned} \tag{6}$$

where h is a differentiable function that depends on the NN parameter θ and (optionally) the randomness $\boldsymbol{\xi}$. Intuitively, h is the surrogate of $\boldsymbol{D} = D(\theta_0, \boldsymbol{\xi})$ and it allows $D_{\text{ST}}(\theta, \boldsymbol{\xi})$ to be differentiable. In the forward propagation we have $D_{\text{ST}}(\theta_0, \boldsymbol{\xi}) = \boldsymbol{D}$ while the backward propagation gives $\nabla_\theta D_{\text{ST}}(\theta, \boldsymbol{\xi}) = \nabla_\theta h(\theta, \boldsymbol{\xi})$. In other words, D_{ST} has the same value as $\boldsymbol{D}$ and therefore is random and one-hot during the function evaluation but is replaced by h during the gradient computation. In fact, Eq. (6) can be viewed as a continuous local approximation around the discrete and non-differentiable $D(\theta_0, \boldsymbol{\xi})$ by adding a curve $c(\theta) = h(\theta, \boldsymbol{\xi}) - h(\theta_0, \boldsymbol{\xi})$ such that $c(\theta_0) = 0$ and $c(\theta)$ is differentiable w.r.t θ. This builds the local structure around θ_0 and makes the differentiation possible around the neighborhood of $D(\theta_0, \boldsymbol{\xi})$.

Although Eq. (6) seems a bit abstract, a naive way to specify h is as follows.

$$D_{\text{ST-naive}}(\theta, \boldsymbol{\xi}) = D(\theta_0, \boldsymbol{\xi}) - p_{\theta_0} + p_\theta \tag{7}$$

Eq. (7) is reduced from Eq. (6) with $h(\theta, \boldsymbol{\xi}) = p_\theta$, where p_θ is the probability vector modeled by an NN. While being technically feasible, such a naive choice ignores the randomness $\boldsymbol{\xi}$ and is therefore unfavorable. A better choice is to design an $h(\theta, \boldsymbol{\xi})$ with a strong connection to $\boldsymbol{D}$ considering that the latter is replaced by the former during the gradient computation in Eq. (6). We will review a better choice in the next subsection.

2.3. Straight-Through Gumbel-Softmax (STGS)

Gumbel-Softmax (Jang et al., 2017; Maddison et al., 2017) provides a reasonable choice of $h(\theta, \boldsymbol{\xi})$ that is strongly correlated with $\boldsymbol{D}$ and is differentiable w.r.t the NN parameter θ. The construction is as follows.

Define $\text{Softmax}_\tau : \mathbb{R}^n \to \mathbb{R}^n$ as $[\text{Softmax}_\tau(x)]_i = e^{x_i/\tau} / \sum_{j=1}^n e^{x_j/\tau}$ and let $p_\theta = \text{Softmax}_1(\text{logit}_\theta)$. That is, the NN parameter θ generates the unnormalized log probability vector logit_θ, and logit_θ yields the discrete distribution p_θ through a Softmax function with temperature $\tau = 1$. The Gumbel-Max trick (Maddison et al., 2014) states that

$$\arg\max_{1 \le i \le N} [\text{logit}_\theta + \boldsymbol{G}]_i \sim p_\theta = \text{Softmax}_1(\text{logit}_\theta), \tag{8}$$

where $\boldsymbol{G}$ is an i.i.d. N-dimensional Gumbel$(0, 1)$ vector. Therefore, we can sample the discrete random variable $\boldsymbol{D}$ using Eq. (8). Since the gradient of $\arg\max$ in Eq. (8) is not useful (either 0 or undefined), Gumbel-Softmax approximates $\arg\max$ of Eq. (8) using another Softmax:

$$D_{\text{GS}}(\theta, \boldsymbol{\xi}) = \text{Softmax}_\tau(\text{logit}_\theta + \boldsymbol{G}) \tag{9}$$

After sampling $D(\theta_0, \boldsymbol{\xi})$ from Eq. (8) and constructing $h(\theta, \boldsymbol{\xi})$ from Eq. (9), we can substitute them into Eq. (6) to get the Straight-Through Gumbel-Softmax (STGS) estimator (Jang et al., 2017):

$$D_{\text{STGS}}(\theta, \boldsymbol{\xi}) = D(\theta_0, \boldsymbol{\xi}) - D_{\text{GS}}(\theta_0, \boldsymbol{\xi}) + D_{\text{GS}}(\theta, \boldsymbol{\xi}), \tag{10}$$

where $\boldsymbol{D} = D(\theta_0, \boldsymbol{\xi}) = \text{OneHot}(\arg\max_{1 \le i \le N} [\text{logit}_{\theta_0} + \boldsymbol{G}]_i)$ is the one-hot random sample using the Gumbel-Max in Eq. (8). $\boldsymbol{\xi}$ is the randomness from the Gumbel vector $\boldsymbol{G}$.

Note that Eq. (10) is reduced from Eq. (6) with $h(\theta, \boldsymbol{\xi}) = D_{\text{GS}}(\theta, \boldsymbol{\xi})$. This allows $\boldsymbol{D}$ and $h(\theta, \boldsymbol{\xi})$ to share the same randomness hence establishes a strong correlation, making Straight-Through Gumbel-Softmax Eq. (10) preferable to the naive choice in Eq. (7).

[1] We use the concept of *differentiablility* loosely in this paper. For example, a common misunderstanding is that the $\arg\max$ operation is not differentiable. However, it is differentiable almost everywhere with gradient 0 except for the case when the equality sign holds (Paulus et al., 2020). Nevertheless, we still consider it as *non-differentiable* in this paper.

2.4. Conditional Perspective on STGS

Although STGS successfully chooses an h that has a strong correlation with $\boldsymbol{D}$, there is still room for improvement. The key observation is that $\boldsymbol{D} = D(\theta_0, \boldsymbol{\xi})$ in Eq. (10) is "much less random" than its random source $\boldsymbol{\xi}$. That is, for a fixed discrete value e_i, there are multiple instances of $\boldsymbol{\xi}$ that lead to the same $D(\theta_0, \boldsymbol{\xi}) = e_i$. From a variance reduction perspective, it is tempting to set $h(\theta, \boldsymbol{\xi}) = h(\theta, D(\theta_0, \boldsymbol{\xi})) = h(\theta, \boldsymbol{D})$. In other words, it is enough to make the randomness of h solely come from $\boldsymbol{D}$ if there is a strong correlation between $h(\theta, \cdot)$ and $\boldsymbol{D}$.

Such a variance reduction by conditioning is proposed in Paulus et al. (2021). The authors use an averaging over conditional distribution to make the randomness of h almost come from $\boldsymbol{D}$. To be more specific, we can first sample a $\boldsymbol{D} = D(\theta_0, \boldsymbol{\xi}) \sim p_{\theta_0}$ and then sample $\boldsymbol{J}_i(\theta) \overset{\text{i.i.d.}}{\sim} \text{logit}_\theta + \boldsymbol{G}|\boldsymbol{D}$. Finally, a rao-blackwellization scheme can be constructed as follows.

$$
\begin{aligned}
&D_{\text{GR-MCK}}(\theta, \boldsymbol{\xi}) = \\
&\boldsymbol{D} + \frac{1}{K}\sum_{i=1}^{K}[-\text{Softmax}_\tau(\boldsymbol{J}_i(\theta_0)) + \text{Softmax}_\tau(\boldsymbol{J}_i(\theta))] \quad (11)\\
&\overset{K\to\infty}{\to} \boldsymbol{D} - \mathbb{E}[D_{\text{GS}}(\theta_0, \boldsymbol{\xi})|\boldsymbol{D}] + \mathbb{E}[D_{\text{GS}}(\theta, \boldsymbol{\xi})|\boldsymbol{D}].
\end{aligned}
$$

When $K = 1$, Eq. (11) is identical to STGS. Therefore, Eq. (11) implies another construction of STGS: first sample $\boldsymbol{D} \sim p_{\theta_0}$ and then $\text{logit}_\theta + \boldsymbol{G}|\boldsymbol{D}$. Such a two-step process motivates the design of our GST estimator in §4. On the other hand, when K is large, Eq. (11) converges to the conditional expectation. $\mathbb{E}[D_{\text{GS}}(\theta, \boldsymbol{\xi})|\boldsymbol{D}]$'s randomness only depends on $\boldsymbol{D} = D(\theta_0, \boldsymbol{\xi})$, not $\boldsymbol{\xi}$. Hence, Eq. (11) reduces the randomness through conditioning and averaging.

3. Key Properties of STGS

Given the widespread success of STGS, we want to identify its good properties and use them to motivate our estimator design in §4. An ablation study is conducted in §5.1.1 to verify the usefulness of these properties.

3.0. Property 0: Following p_{θ_0}

We want to stress the most basic property of STGS: STGS follows $p_{\theta_0} = \text{Softmax}_1(\text{logit}_{\theta_0})$ during the forward propagation of a NN. This is because Eq. (10) reduces to $\boldsymbol{D}$ during the forward pass (i.e. $\theta = \theta_0$), and $\boldsymbol{D}$ is sampled by Gumbel-Max, yielding $\boldsymbol{D} \sim p_{\theta_0}$. As shown in Eq. (6), any discrete DGM based on the Straight-Through estimator satisfies this property. Since we focus on the family of STGS, we assume this property holds throughout this paper.

3.1. Property 1: Consistency

Recall that in Eq. (10), $\boldsymbol{D} = \text{OneHot}(\arg\max_{1\le i\le N}[\text{logit}_{\theta_0} + G]_i)$ is the one-hot sample using the Gumbel-Max trick and $D_{\text{GS}}(\theta_0, \boldsymbol{\xi}) = \text{Softmax}_\tau(\text{logit}_{\theta_0} + \boldsymbol{G})$ is its surrogate. Because the Softmax operation does not change the relative order of the input vector components, we have: $\arg\max_{1\le i\le N}[\boldsymbol{D}]_i = \arg\max_{1\le i\le N}[D_{\text{GS}}(\theta_0, \boldsymbol{\xi})]_i$. Thereby, we say a differentiable surrogate function h is consistent with $\boldsymbol{D}$ in Eq. (6) if:

$$
\arg\max_{1\le i\le N}[h(\theta_0, \boldsymbol{\xi})]_i = \arg\max_{1\le i\le N}[\boldsymbol{D}]_i. \quad (12)
$$

This makes intuitive sense as a surrogate should at least keep the supremacy of the largest component of the input.

3.2. Property 2: Zero-Gradient Perturbation

We now discuss the functional form of the differentiable surrogate, $h(\theta, \boldsymbol{\xi})$. One observation is that h should sit in the probability simplex $\Delta_{N-1} = \{(u_1, ..., u_N) : \sum_{i=1}^N u_i = 1,\ u_i \ge 0\ \forall\ i \in [1, ..., N]\}$, which can be achieved by applying a Softmax or Sparsemax (Martins & Astudillo, 2016). This is because h mimics $\boldsymbol{D}$ and $\boldsymbol{D}$ is on the vertices of Δ_{N-1}. Since we are more compatible with Softmax, we require $h = \text{Softmax}_\tau(\text{some logit})$. Then, the design of h boils down to the logit design. Gumbel-Softmax chooses its logit as a perturbed one: $\text{logit}_\theta + \boldsymbol{G}$. More generally, we introduce a perturbation function m and write h as

$$
h(\theta, \boldsymbol{\xi}) = \text{Softmax}_\tau(\text{logit}_\theta + m(\theta_0, \boldsymbol{\xi})). \quad (13)
$$

Note that $m(\theta_0, \boldsymbol{\xi}) = \boldsymbol{G}$ for Gumbel-Softmax, so m is a perturbation function that generalizes the Gumbel vector. To see why m is designed to be a function of θ_0 and $\boldsymbol{\xi}$, we first note that m has to depend on the sampling randomness $\boldsymbol{\xi}$ so that the surrogate h correlates well with the random one-hot sample $\boldsymbol{D}$. Secondly, the perturbation may depend on logit_θ, so the dependency on θ should also be included. Furthermore, θ is replaced by its zero-gradient version, θ_0, in order to maintain property 1 after a small gradient descent step in θ.[2]

Eq. (13) brings up the idea of *perturbed logits*. For a zero-gradient perturbation $m(\theta_0, \boldsymbol{\xi})$, we call $\text{logit}_\theta + m(\theta_0, \boldsymbol{\xi})$ the perturbed logits of logit_θ by m, or simply the perturbed logits. We will discuss the property of perturbed logits in the next subsection, which will be useful for our GST estimator proposed in §4.

[2] Let $h' = \text{Softmax}_\tau(\text{logit}_\theta + m(\theta, \boldsymbol{\xi}))$. After a gradient step, $m(\theta, \boldsymbol{\xi})$ is changed but $m(\theta_0, \boldsymbol{\xi})$ is not, so h is more likely to be consistent with $\boldsymbol{D}$ than h' is.

3.3. Property 3: Strict Gap Between Perturbed Logits

In §3.1, we focused only on the largest logit, but this leaves us wonder the actual difference between the largest and the other logits. The difference is important since we want to apply the perturbation correctly to reflect the logit difference. To simplify the problem, we study the difference between the top-2 largest perturbed logits of STGS. In particular, conditioning on the event that i is the index of the largest perturbed logit; i.e.,

$$\boldsymbol{D} = \text{OneHot}(\arg\max_{1\le j\le N}[\text{logit}_{\theta_0}]_j + \boldsymbol{G}_j) = e_i,$$

the expected gap between the top-2 largest perturbed logits is defined as

$$\text{Gap}(\theta_0|\boldsymbol{D}=e_i) = \mathbb{E}\left[[\text{logit}_{\theta_0}]_i + \boldsymbol{G}_i - \max_{j\neq i}\left([\text{logit}_{\theta_0}]_j + \boldsymbol{G}_j\right)\Big|\boldsymbol{D}=e_i\right]. \quad (14)$$

Lemma 1 gives an analytical expression of Eq. (14).

Lemma 1. *Gap*$(\theta_0|\boldsymbol{D}=e_i) = -\frac{\log(1-[p_{\theta_0}]_i)}{[p_{\theta_0}]_i} = \frac{\log\left(1+e^{\ell_i-s}\right)}{1-1/(1+e^{\ell_i-s})}$, *where* ℓ_i *is the shorthand of* $[logit_{\theta_0}]_i$ *and* $s=\log\left(\sum_{j\neq i}e^{\ell_j}\right)$ *is the log-sum-exponential of the unselected logits.*

A closer look into Lemma 1 shows that the expected gap largely depends on the logit difference $\ell_i - s$ and s is interpreted as *the effective unselected logit*. It is immediate to see that **(a)** the gap increases in $\ell_i - s$, **(b)** the gap converges to 1 when $\ell_i - s \ll 0$, and **(c)** the gap converges to $\ell_i - s$ as $\ell_i - s \gg 0$.

(a) follows from that larger $\ell_i - s$ implies larger logit difference and gap. **(b)** means the expected gap is strict and is lower-bounded by 1. **(c)** follows from that the Gumbel noises are negligible when $\ell_i - s$ is large, so the expected gap converges to the effective unperturbed logit difference.

(b) is probably the most counter-intuitive but also the most important property. Although the intuition might suggest a vanishing gap when $\ell_i - s \ll 0$, the event $\boldsymbol{D} = e_i$ turns out to put weight on large enough random gaps such that the expected gap ≥ 1. We provide more discussion on this in Appendix A.2. The strict expected gap is important for the τ-tempered Gumbel-Softmax, Eq. (9), to converge to $\boldsymbol{D}$ easier at low temperatures. That is, a strict gap implies $D_{\text{GS}}(\theta_0,\boldsymbol{\xi}) = \text{Softmax}_\tau(\text{logit}_{\theta_0}+\boldsymbol{G}) \overset{\tau\to 0}{\to} D(\theta_0,\boldsymbol{\xi}) = \boldsymbol{D}$ and justifies D_{GS} as a differentiable surrogate of $\boldsymbol{D}$.

4. The Proposed Method

4.1. Near-deterministic Straight-Through Estimator

As discussed in §2.4, Paulus et al. (2021) propose conditional averaging for variance reduction. This is not very efficient in large-scale modeling as the averaging causes an extra computation of size $O(K)$ with K typically being 100 or higher. Nevertheless, in Eq. (11), the average converges to $\mathbb{E}[D_{\text{GS}}(\theta,\boldsymbol{\xi})|\boldsymbol{D}]$ at large K, which is a deterministic function in $\boldsymbol{D}$. This implies a good choice of deterministic function in $\boldsymbol{D}$, written as $h(\theta,\boldsymbol{D})$, might improve the performance. We thus reduce the general Straight-Through, Eq. (6) to the following near-deterministic Straight-Through estimator.

$$D_{\text{ST-det}}(\theta,\boldsymbol{D}) = \boldsymbol{D} - h(\theta_0,\boldsymbol{D}) + h(\theta,\boldsymbol{D}). \quad (15)$$

$\boldsymbol{D} = D(\theta_0,\boldsymbol{\xi})$ is generated from the random source $\boldsymbol{\xi}$. We see that Eq. (11) with $K\to\infty$ becomes a special case of Eq. (15) with $h(\theta,\boldsymbol{D}) = \mathbb{E}[D_{\text{GS}}(\theta,\boldsymbol{\xi})|\boldsymbol{D}]$. In the next subsection, we will find another $h(\theta,\boldsymbol{D})$ that also has a high correlation with $\boldsymbol{D}$ but without high resampling cost.

4.2. Gapped Straight-Through Estimator

We now present our Gapped Straight-through Estimator (GST). Due to the success of variance-reduction-type estimator (Paulus et al., 2021), it is enough to find a good deterministic function, $h(\theta,\boldsymbol{D})$, for the near-deterministic Straight-Through estimator, Eq. (15). Motivated by the success of Gumbel-Softmax, we choose $h(\theta,\boldsymbol{D})$ based on the properties discussed in §3.

According to property 1 & 2, we require $h(\theta,\boldsymbol{D})$ to satisfy

$$\arg\max_{1\le j\le N}[h(\theta,\boldsymbol{D})]_j = \arg\max_{1\le j\le N}[\boldsymbol{D}]_j$$
$$h(\theta,\boldsymbol{D}) = \text{Softmax}_\tau(\text{logit}_\theta + m(\theta_0,\boldsymbol{D})).$$

Since Softmax does not change the relative order of the input components, this can be reduced to

$$\arg\max_{1\le j\le N}[\text{logit}_\theta + m(\theta_0,\boldsymbol{D})]_j = \arg\max_{1\le j\le N}[\boldsymbol{D}]_j. \quad (16)$$

To realize Eq. (16), we design the perturbation m_1 that pushes the $\boldsymbol{D}$-selected logit $\langle\text{logit}_{\theta_0},\boldsymbol{D}\rangle$ to be the same as the largest logit. Note $\langle\cdot,\cdot\rangle$ denotes the inner product. Then, Eq. (16) is satisfied when $m(\theta_0,\boldsymbol{D}) = m_1(\theta_0,\boldsymbol{D})$:

$$m_1(\theta_0,\boldsymbol{D}) = \left(\max_{1\le j\le N}[\text{logit}_{\theta_0}]_j - \langle\text{logit}_{\theta_0},\boldsymbol{D}\rangle\right)\cdot\boldsymbol{D}. \quad (17)$$

Although a combination of Eq. (13) and Eq. (17) realizes properties 1 & 2, it cannot guarantee a strict gap between the top-2 perturbed logits (i.e., when $\boldsymbol{D}$ does not select the largest logit, m_1 makes the top-2 perturbed logits the same, and the gap size becomes zero). To enforce property 3's strict gap, we may either increase the selected logit or decrease the unselected ones. While both are mathematically correct, experiments show the latter gives stable results.

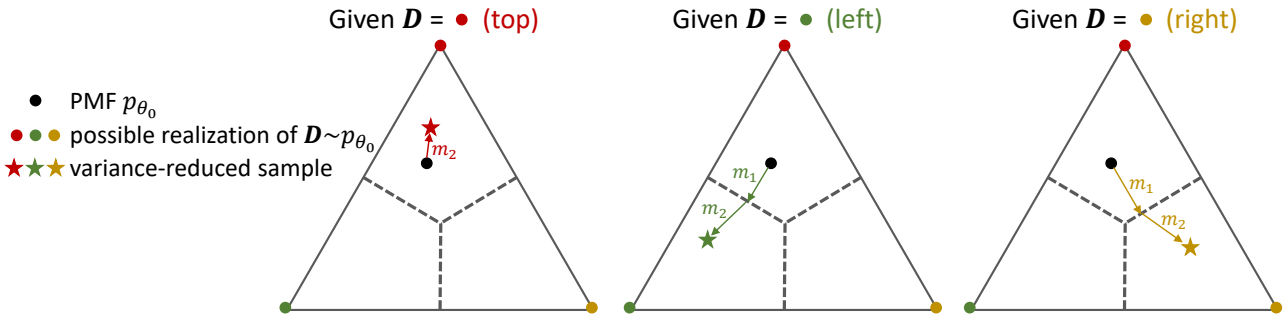

Figure 2. **The soft samples GST estimator on the probability simplex.** The dashed line separate three categories (top, left, right). First, GST chooses a random category $\boldsymbol{D} \sim p_{\theta_0}$, arriving at either the red, green or yellow point. Then, starting from p_{θ_0}, apply m_1 and m_2 to get a variance-reduced soft sample given $\boldsymbol{D}$. m_1 pushes the point to the boundary (if $\arg\max_j\ [p_{\theta_0}]_j \neq \arg\max_j\ [\boldsymbol{D}]_j$), and m_2 encourages a strict gap between the $\boldsymbol{D}$-selected logit and the unselected ones so that $\boldsymbol{D}$'s category stands out.

To enforce a strict gap (property 3 of §3), we choose to make the unselected logits smaller. Let $[m_2]_k$ be the decrease on the unselected logit at index k such that the gap size is at least $g \geq 0$. Then, for all k in the indices of unselected logits, we require the following (Note that the selected logit's value becomes $\max_{1\leq j\leq N} [\text{logit}_{\theta_0}]_j$ after applying m_1.)

$$\max_{1\leq j\leq N} [\text{logit}_{\theta_0}]_j - ([\text{logit}_{\theta_0}]_k - [m_2]_k) \geq g.$$

To enhance the sparsity, we also require $[m_2]_k \geq 0$ where $[m_2]_k = 0$ means the kth logit doesn't need the decrease. This avoids the unnecessary perturbation on the logit and makes it simpler to construct m_2. $m_2(\theta_0, \boldsymbol{D}, g)$ is defined as follows.

$$m_2(\theta_0, \boldsymbol{D}, g) = \left(\text{logit}_{\theta_0} + g - \max_{1\leq j\leq N} [\text{logit}_{\theta_0}]_j\right)_+ \cdot (1 - \boldsymbol{D}), \tag{18}$$

where $(x)_+ = \max(x, 0)$ and the product with $(1 - \boldsymbol{D})$ is element-wise (Hadamard product). Combining m_1 and m_2, we propose the Gapped Straight-Through estimator as

$$\begin{aligned} &D_{\text{GST}}(\theta, \boldsymbol{D}) = \boldsymbol{D} - h(\theta_0, \boldsymbol{D}) + h(\theta, \boldsymbol{D}) \\ &h(\theta, \boldsymbol{D}) = \text{Softmax}_\tau(\text{logit}_\theta + m_1(\theta_0, \boldsymbol{D}) - m_2(\theta_0, \boldsymbol{D}, g)). \end{aligned} \tag{19}$$

One might expect $g \approx 1$ due to the limiting behavior of Gumbel-Softmax's expected gap discussed in §3. Furthermore, according to Lemma 1, we may set the gap as $g = -\frac{\log(1-[p_{\theta_0}]_i)}{[p_{\theta_0}]_i}$ for $\boldsymbol{D} = e_i$. The experiments in §5 show that both produce very similar results.

4.3. Connection To Paulus et al. (2021)

Let $\mathbb{E}[X], \mathbb{V}[X]$ be the mean and variance of X. By the chain rule and the law of total variance, the variances of gradients under GR-MCK and our proposed GST are

$$\begin{aligned} \mathbb{V}[\nabla_{\text{GR-MCK}}] &= \mathbb{E}[\mathbb{V}[\nabla_{\text{GR-MCK}}|\boldsymbol{D}]] + \mathbb{V}[\mathbb{E}[\nabla_{\text{GR-MCK}}|\boldsymbol{D}]] \\ &= \underbrace{\frac{1}{K}\mathbb{E}\Big[\Big(\frac{\partial g(\boldsymbol{D})}{\partial \boldsymbol{D}}\Big)^2 \mathbb{V}\Big[\frac{\partial D_{GS}(\theta, \boldsymbol{\xi})}{\partial \theta}\Big|\boldsymbol{D}\Big]\Big]}_{(a)} \\ &\quad + \underbrace{\mathbb{V}\Big[\frac{\partial g(\boldsymbol{D})}{\partial \boldsymbol{D}}\mathbb{E}\Big[\frac{\partial D_{GS}(\theta, \boldsymbol{\xi})}{\partial \theta}\Big|\boldsymbol{D}\Big]\Big]}_{(b)}. \end{aligned}$$

$$\mathbb{V}[\nabla_{\text{GST}}] = \mathbb{V}\Big[\frac{\partial g(\boldsymbol{D})}{\partial \boldsymbol{D}}\frac{\partial D_{\text{GST}}(\theta, \boldsymbol{D})}{\partial \theta}\Big].$$

When $K = 1$, $\mathbb{V}[\nabla_{\text{GR-MCK}}]$ becomes $\mathbb{V}[\nabla_{\text{STGS}}]$. Therefore, GR-MCK reduces the variance of STGS by minimizing term (a), which decreases in K. On the other hand, $\mathbb{V}[\nabla_{\text{GST}}]$ is very similar to term (b). Since GST follows the key properties of STGS, we expect $\mathbb{E}\Big[\frac{\partial D_{GS}(\theta, \boldsymbol{\xi})}{\partial \theta}\Big|\boldsymbol{D}\Big] \approx \frac{\partial D_{\text{GST}}(\theta, \boldsymbol{D})}{\partial \theta}$, implying that GST may enjoy similar variance reduction as GR-MCK. This is verified through experiments in §5.

Algorithm 1 The Proposed GST Estimator

Input : NN para. θ, temperature τ, mode $\in$ {hard,soft}
Output : $D_{\text{GST}}(\theta, \boldsymbol{D})$
$\boldsymbol{D}$ = sample_onehot_from(p_{θ_0}); $p_{\theta_0} = \text{Softmax}_1(\text{logit}_{\theta_0})$.
m_1 = Eq. (17); `/* consistency */`
m_2 = Eq. (18); `/* strict gap */`
$h(\theta, \boldsymbol{D}) = \text{Softmax}_\tau(\text{logit}_\theta + m_1 - m_2)$ by Eq. (19);
if *mode is hard* **then**
  return $\boldsymbol{D}$ - stop_gradient($h(\theta, \boldsymbol{D})$) + $h(\theta, \boldsymbol{D})$; `/* Straight-Through trick for hard samples */`
else
  return $h(\theta, \boldsymbol{D})$; `/* soft samples */`
end

5. Experiments

In this section, we compare our Gapped Straight-Through (GST) estimator with the Straight-Through-based model such as STGS (Jang et al., 2017) and its Monte Carlo variance reduction variant, GR-MC100 (Paulus et al., 2021). Note MC100 means taking 100 independent samples during the Monte Carlo sampling. All models generate *hard samples* using the Straight-Through trick. *Soft samples* are the outputs of their differentiable surrogate functions, e.g. Eq. (9) and $h(\theta, \boldsymbol{D})$ in Eq. (19).

We evaluate the models on two standard tasks, MNIST-VAE (Jang et al., 2017; Kingma et al., 2014) and ListOps unsupervised parsing (Nangia & Bowman, 2018). MNIST-VAE takes each digit of an MNIST image as a random variable and trains a Variational Auto Encoder (Kingma & Welling, 2019) to generate images. The latent space is represented by 30 categorical variables, each with 10 categories. i.e., 30 one-hot vectors with a total dimension of 30x10. ListOps is a dataset composed of prefix arithmetic expressions such as $\min(3, \max(5, 6))$, and the objective is to predict their evaluation results. The task is typically addressed by a tree-LSTM (Tai et al., 2015) that learns a distribution over latent parse trees. The categorical random variable models the distribution of the parent node over the plausible candidates (Choi et al., 2018). Since the latent space of MNIST-VAE accepts either discrete or continuous representations while ListOps' parent node selection is strictly discrete, the *soft sample* option is only allowed in MNIST-VAE and is forbidden in ListOps.[3]

5.1. MNIST-VAE

A variational autoencoder that generates the MNIST images can be trained by maximizing the evidence lower bound (ELBO) on the log-likelihood:

$$\begin{aligned}
&\log L_{\theta,\phi}(x) \\
&\geq \mathbb{E}_{\boldsymbol{D}^{1:30}\sim p_\theta(\cdot|x)}\left[\log \frac{L_\phi(x|\boldsymbol{D}^{1:30})\text{Pri}(\boldsymbol{D}^{1:30})}{p_\theta(\boldsymbol{D}^{1:30}|x)}\right] \\
&= \mathbb{E}_{\boldsymbol{D}^{1:30}\sim p_\theta(\cdot|x)} \log L_\phi(x|\boldsymbol{D}^{1:30}) - KL(p_\theta(\cdot|x)\|\text{Pri}(\cdot)) \\
&= \text{ELBO}_{\theta,\phi}(x).
\end{aligned}$$

x is an input image. $\boldsymbol{D}^{1:30} = [\boldsymbol{D}^1, ..., \boldsymbol{D}^{30}]$ is the concatenation of 30 categorical random variables, with each $\boldsymbol{D}^i$ having 10 categories. $p_\theta(D^{1:30}|x)$ is the encoder as well as the probability mass of $\boldsymbol{D}^{1:30}$ conditioning on the input image. $L_\phi(x|\boldsymbol{D}^{1:30})$ is the decoder that model likelihood of x given $\boldsymbol{D}^{1:30}$. $\text{Pri}(D^{1:30}) = 1/10^{30}$ is the prior of $\boldsymbol{D}^{1:30}$ and is chosen as the uniform distribution. We train the VAE with the loss function being the negative of ELBO:

$$-\mathbb{E}_{x\sim\mathcal{X}_{\text{train}}}\text{ELBO}_{\theta,\phi}(x), \quad \mathcal{X}_{\text{train}}: \text{ training data.}$$

[3]We release our code for both tasks at: https://github.com/chijames/GST.

and test the model by replacing $\mathcal{X}_{\text{train}}$ with the testing data $\mathcal{X}_{\text{test}}$. For simplicity, we train all tasks using the same neural network structure, batch size (=100), epochs (=40), optimizer (Adam, learning rate=0.001) and seeds ($\in$[0,1,...,9]). The only differences are the models (STGS, GR-MC100, GST) and temperatures $\tau \in [1, 0.5, 0.1]$.

5.1.1. Ablation Study

To start with, we conduct an ablation study of the properties in §3. We compare five estimators from which only one (GST-1.0) satisfies all properties. Because GST-1.0 performs much better than the others, the study justifies that a reasonable estimator should satisfy all properties at the same time.

Table 1 summarizes the estimators and their properties. ST is Eq. (7), which lacks the consistency because $\arg\max_i [p_{\theta_0}]_i$ may not be $\arg\max_i \boldsymbol{D}_i$. Gap-0.0 is Eq. (19) with $g = 0$ and thus the gap is zero by construction. NZ means "non-zero gradient for m"; i.e. $m = m(\theta, \boldsymbol{\xi})$ and hence $\nabla_\theta m \neq 0$.

Table 2 suggests that as long as an estimator violates any of the properties, its performance will worsen. Note that the best performing one, GST-1.0, satisfies all three properties.

Estimator	Consistency	$\nabla_\theta m = 0$	Gap > 0
ST	x	✓	✓
NZ-GST-0.0	✓	x	x
NZ-GST-1.0	✓	x	✓
GST-0.0	✓	✓	x
GST-1.0	✓	✓	✓

Table 1. **Estimators and properties 1, 2 & 3 of section 3.**

Estimator	Temperature	Neg. ELBO	Std.
ST	1.0	123.35	0.54
	0.5	133.84	0.66
NZ-GST-0.0	1.0	128.45	0.53
	0.5	139.02	0.47
NZ-GST-1.0	1.0	205.88	0.04
	0.5	205.90	0.05
GST-0.0	0.5	119.10	0.29
	1.0	115.48	0.81
GST-1.0	1.0	113.63	1.48
	0.5	108.43	1.08

Table 2. **Ablation study on the dev set.** The smaller (average) negative ELBO the better. Std. is the standard deviation.

5.1.2. Comparisons of Different Estimators

We now turn to the comparison of STGS, GR-MC100, and GST. Each of these estimators satisfies the properties in §3

and should perform reasonably well. The metrics we focus on are the negative ELBO and the standard deviation.

We can see that the GST estimators outperform all the other estimators at temperatures 1.0 and 0.5. Nevertheless, our GST estimator does not perform well under the low-temperature settings, which surprised us at first. After a closer inspection, we conclude that absolute temperature scales for different estimators are not directly comparable. Concretely, we calculate the entropy of the probability distribution calculated in Eq. (19). For the first 100 steps at temperature 0.1, the entropy of GR-MC100, STGS, and GST-1.2 are 0.36, 0.15, 0.0007, respectively. The effect is detrimental to GST-1.2 as it does not have enough exploration during the early stage of training. The inferior performance of STGS and GR-MC100 at temperature=0.01 also corroborates the effect. Still, this does not undermine the superior quality of the GST estimator, as we can fix the temperature at 1.0 or 0.5 and outperform the previous state-of-the-art GR-MC100 at its best setting. For interested readers, we offer a mixed temperature training strategy that can stabilize the training process of GST at low temperatures in Appendix A.3.

Finally, we notice that GST-pi may not perform better than GST-1.0 or GST-1.2 even though GST-pi follows Lemma 1: Eq. (19) with the expected gap of STGS, $g = -\frac{\log(1-[p_{\theta_0}]_i)}{[p_{\theta_0}]_i}$. Recall that the expected gap converges to 1 when $\ell_i - s \ll 0$ or equivalently, $[p_{\theta_0}]_i \to 0$. Hence, the limiting behavior of the expected gap ($g = 1$) might be more useful than the exactly expected gap at a certain $[p_{\theta_0}]_i$. It is enough to choose $g \approx 1$ for the MNIST-VAE task.

5.1.3. Empirical Variance Comparison

We evaluate the variance of the gradient estimators once at the end of each epoch on the MNIST-VAE task. Figure 3 shows our GST estimator achieves lower variances compared with STGS and GR-MC100 throughout the entire training process.

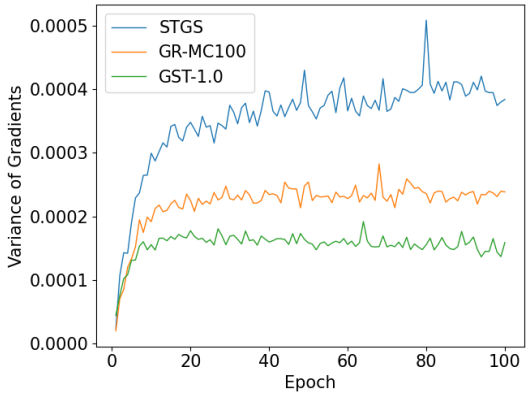

Figure 3. Variance Comparison on MNIST-VAE.

Temperature	Estimator	Neg. ELBO	Std.
1.0	STGS	122.96	3.08
	GR-MC100	120.65	2.95
	GST-0.8	113.61	1.96
	GST-1.0	113.63	1.48
	GST-1.2	**112.58**	**1.11**
	GST-pi	112.72	1.63
0.5	STGS	118.96	2.51
	GR-MC100	117.88	3.01
	GST-0.8	111.54	1.30
	GST-1.0	108.43	1.08
	GST-1.2	**107.33**	**0.69**
	GST-pi	109.42	1.32
0.1	STGS	127.70	3.99
	GR-MC100	123.23	3.55
	GST-*	X	X
0.01	STGS	141.61	3.53
	GR-MC100	130.04	3.63
	GST-*	X	X

Table 3. **Comparison of estimators on MNIST-VAE.** The results are evaluated over ten different random seeds. GST-* denotes the GST-based estimators. X means the estimator cannot converge.

5.1.4. Comparisons with REINFORCE-based estimators

To better position our work in the literature of discrete gradient estimation techniques, we additionally compare our estimators, GST-1.0 and GST-1.2, with several state-of-the-art REINFORCE-based strong baselines (Dong et al., 2020; 2021). In Table 4, we closely follow the network architecture (e.g., sizes, activations) and all the hyperparameters (e.g., batch size, epochs) by carefully inspecting the released code at `https://github.com/google-research/google-research/tree/master/disarm`. We can see that our estimator outperforms all the baselines.

Estimator	Neg. ELBO
RLOO	104.03 ± 0.23
DisARM-Tree	103.10 ± 0.25
STGS	97.32 ± 0.20
GR-MC100*	110.74 ± 1.23
GST-1.0	$\mathbf{96.09 \pm 0.25}$
GST-1.2	96.16 ± 0.32

Table 4. **Comparison of Estimators on MNIST-VAE with Dong et al. (2021).** The identical implementation (5 random seeds, 200 batch size, 5×10^5 training steps, 32 latent variables with 64 categories, optimizer, and architecture) is adopted. *GR-MC100 fails (diverges and yields computational errors) around epoch 40 to 70 for all seeds at all temperatures. Here, we report the numbers at temperature=0.5 right before it fails. We hypothesize that the failure can be attributed to its complicated resampling process.

5.1.5. Computational Efficiency

Table 5 is the computational efficiency comparison for the MNIST-VAE task on one Nvidia 1080-Ti GPU. GST is as efficient as STGS. In contrast, GR-MCK performs a resampling for K times, consuming more memory. For the 1000 sample case, it is not possible for the GPU to parallelize all computation at once, so longer time is needed.

Estimator	Allocated GPU Mem.	Time
STGS	0.023 GB	5.03 sec/epoch
GST-1.0	0.023 GB	5.04 sec/epoch
GR-MC100	0.081 GB	5.09 sec/epoch
GR-MC1000	0.602 GB	8.73 sec/epoch

Table 5. Computational Efficiency in Terms of Memory and Speed.

5.2. ListOps

For ListOps (Nangia & Bowman, 2018), we use the same model architecture as in Choi et al. (2018). In particular, we want to optimize:

$$\mathop{\mathbb{E}}_{q_\theta(T|x)}[\log p_\phi(y|T,x)] \tag{20}$$

Both q_θ and p_ϕ are modeled by NNs. Given an arithmetic expression with M tokens, $m = m_1, m_2, \cdots, m_M$, q_θ is responsible for sampling a parse tree T by merging a pair of adjacent tokens m_i, m_{i+1} at a time. In other words, m_i and m_{i+1} are merged into one new representation, and the total length is reduced by 1. After $M-1$ operations, the expression will end up with only one representation, from which the model predicts the final evaluation results and optimizes the parameters using the cross-entropy loss. As the construction of T requires discrete sampling, STGS is used with hard samples for training (Choi et al., 2018; Paulus et al., 2021). To validate the effectiveness of our GST estimator, we replace STGS with our estimator and measure the final accuracy of answer prediction.

Table 6 shows that the estimators perform differently on the ListOps task. We first note that GST-1.0 performs the best (in both accuracy and variance) among all other variants in the GST family, corroborating our expected gap derivation in §3.3. Therefore, we conclude that 1.0 can be chosen as the default gap for GST. Next, we compare GST-1.0 with STGS, the baseline estimator without an additional variance reduction technique. We can see that GST consistently outperforms STGS across all temperatures with lower variances. The performance difference is especially pronounced when a very low temperature (0.01) is used. Finally, we compare GST with the strongest baseline, GR-MC100. We achieve modest but better performances on two temperatures, 1.0 and 0.1. When the lowest temperature of 0.01 is used, GST performs on par with GR-MC100.

Temperature	Estimator	Accuracy	Std.
1.0	STGS	0.659	0.006
	GR-MC100	0.651	0.009
	GST-0.8	0.660	0.010
	GST-1.0	**0.662**	**0.005**
	GST-1.2	0.660	0.011
	GST-pi	0.659	0.006
0.1	STGS	0.645	0.014
	GR-MC100	0.637	0.049
	GST-0.8	0.664	0.020
	GST-1.0	**0.664**	**0.012**
	GST-1.2	0.660	0.018
	GST-pi	0.659	0.015
0.01	STGS	0.479	0.258
	GR-MC100	**0.662**	**0.002**
	GST-0.8	0.660	0.012
	GST-1.0	0.661	0.004
	GST-1.2	0.661	0.004
	GST-pi	0.652	0.018

Table 6. **Comparison of estimators on ListOps.** We set the maximum sequence length M to 100. We run five different seeds to get the average accuracy and standard deviation.

6. Conclusion

We introduce the Gapped Straight-Through (GST) estimator, a gradient estimator for discrete random variables. We derive the properties of Straight-Through Gumbel-Softmax and use them to design the GST estimator. An ablation study shows these properties are essential for good performance. Compared to prior work, the GST estimator enables variance reduction without resampling. Empirically, the GST estimator works with reduced variances and better performance on the MNIST-VAE and ListOps tasks. We hope the GST estimator helps accelerate the training speed of downstream applications while keeping low variance and good performance.

Acknowledgement

We thank the anonymous reviewers for their insightful feedback and suggestions.

References

Abramowitz, M., Stegun, I. A., and Romer, R. H. Handbook of mathematical functions with formulas, graphs, and mathematical tables, 1988.

Bengio, Y., Léonard, N., and Courville, A. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. d. O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Choi, J., Yoo, K. M., and Lee, S.-g. Learning to compose task-specific tree structures. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Chung, J., Ahn, S., and Bengio, Y. Hierarchical multiscale recurrent neural networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

Dong, Z., Mnih, A., and Tucker, G. Disarm: An antithetic gradient estimator for binary latent variables. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 18637–18647. Curran Associates, Inc., 2020.

Dong, Z., Mnih, A., and Tucker, G. Coupled gradient estimators for discrete latent variables. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 24498–24508. Curran Associates, Inc., 2021.

Fan, T.-H. and Wang, Y. Soft actor-critic with integer actions. *arXiv preprint arXiv:2109.08512*, 2021.

Glynn, P. W. Likelihood ratio gradient estimation for stochastic systems. *Communications of the ACM*, 33 (10):75–84, 1990.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

Gu, S., Levine, S., Sutskever, I., and Mnih, A. Muprop: Unbiased backpropagation for stochastic neural networks. In Bengio, Y. and LeCun, Y. (eds.), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

Hinton, G., Srivastava, N., and Swersky, K. Neural networks for machine learning. *Coursera, video lectures*, 2012.

Ho, J. and Ermon, S. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29:4565–4573, 2016.

Jang, E., Gu, S., and Poole, B. Categorical reparameterization with gumbel-softmax. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

Kingma, D. P. and Welling, M. An introduction to variational autoencoders. *arXiv preprint arXiv:1906.02691*, 2019.

Kingma, D. P., Mohamed, S., Rezende, D. J., and Welling, M. Semi-supervised learning with deep generative models. In *Advances in neural information processing systems*, pp. 3581–3589, 2014.

Korshunov, P. and Marcel, S. Deepfakes: a new threat to face recognition? assessment and detection. *arXiv preprint arXiv:1812.08685*, 2018.

Li, Y., Song, J., and Ermon, S. Infogail: Interpretable imitation learning from visual demonstrations. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 3815–3825, 2017.

Lowe, R., WU, Y., Tamar, A., Harb, J., Pieter Abbeel, O., and Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Maddison, C. J., Tarlow, D., and Minka, T. A* sampling. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

Maddison, C. J., Mnih, A., and Teh, Y. W. The concrete distribution: A continuous relaxation of discrete random variables. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

Martins, A. F. T. and Astudillo, R. F. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pp. 1614–1623. JMLR.org, 2016.

Mnih, A. and Gregor, K. Neural variational inference and learning in belief networks. In Xing, E. P. and Jebara, T. (eds.), *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pp. 1791–1799, Bejing, China, 22–24 Jun 2014. PMLR.

Mnih, A. and Rezende, D. J. Variational inference for monte carlo objectives. In *Proceedings of the 33rd International Conference on International Conference on*

Machine Learning - Volume 48, ICML'16, pp. 2188–2196. JMLR.org, 2016.

Nangia, N. and Bowman, S. R. Listops: A diagnostic dataset for latent tree learning. *arXiv preprint arXiv:1804.06028*, 2018.

Oord, A. v. d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

Paulus, M. B., Choi, D., Tarlow, D., Krause, A., and Maddison, C. J. Gradient estimation with stochastic softmax tricks. *arXiv preprint arXiv:2006.08063*, 2020.

Paulus, M. B., Maddison, C. J., and Krause, A. Rao-blackwellizing the straight-through gumbel-softmax gradient estimator. In *International Conference on Learning Representations*, 2021.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Rezende, D. and Mohamed, S. Variational inference with normalizing flows. In *International conference on machine learning*, pp. 1530–1538. PMLR, 2015.

Ruthotto, L. and Haber, E. An introduction to deep generative modeling. *GAMM-Mitteilungen*, pp. e202100008, 2021.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

Tai, K. S., Socher, R., and Manning, C. D. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 1556–1566, Beijing, China, July 2015. Association for Computational Linguistics.

Tang, Y. and Agrawal, S. Discretizing continuous action space for on-policy optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 5981–5988, 2020.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

Yang, Z., Hu, Z., Salakhutdinov, R., and Berg-Kirkpatrick, T. Improved variational autoencoders for text modeling using dilated convolutions. In *International conference on machine learning*, pp. 3881–3890. PMLR, 2017.

A. Appendix

A.1. Proof of Lemma 1

Fact 1. *Let $\ell = logit_{\theta_0}$ be the shorthand of the logit vector. The conditional expression of the Gumbel-Softmax's logit is:*

$$[\ell + \boldsymbol{G}|\boldsymbol{D} = e_i]_j \overset{d}{=} \begin{cases} -\log \frac{\boldsymbol{E}_i}{Z} & \text{if } j = i \\ -\log\left(\frac{\boldsymbol{E}_j}{e^{\ell_j}} + \frac{\boldsymbol{E}_i}{Z}\right) & \text{o.w.} \end{cases}, \quad j \in [1, ..., n], \tag{21}$$

$Z = \sum_j e^{\ell_j}$ is the partition function. $\boldsymbol{E}_i \overset{i.i.d.}{\sim} Exp(1)$ is a standard exponential random variable.

Proof. The expression is given in Paulus et al. (2021). Here, we provide a proof based on the joint density function. Let $\boldsymbol{E} = [\boldsymbol{E}_1, ..., \boldsymbol{E}_N]$ be an N-dimensional standard exponential random vector. Because $\boldsymbol{G} \overset{d}{=} -\log \boldsymbol{E}$, we have

$$\ell + \boldsymbol{G} \overset{d}{=} \ell - \log \boldsymbol{E} = -\log \frac{\boldsymbol{E}}{e^\ell},$$

where $\boldsymbol{E}/e^\ell$ uses elementwise exponential and division. It is enough to focus on $\boldsymbol{E}/e^\ell$ conditioning on the ith element being the *smallest* entry; namely,

$$e_i = \boldsymbol{D} = \arg\max_j \ell_j + \boldsymbol{G}_j = \arg\min_j \boldsymbol{E}_j/e^{\ell_j} = \arg\min_j \boldsymbol{X}_j. \tag{22}$$

Since each element $\boldsymbol{X}_j = \boldsymbol{E}_j/e^{\ell_j}$ follows Exp(e^{ℓ_j}). Let $\lambda_j = e^{\ell_j}$, the joint density of $\boldsymbol{X} = \boldsymbol{E}/e^\ell$ without conditioning is

$$f(x_1, ..., x_N) = \Pi_{j=1}^N \lambda_j e^{-\lambda_j x_j}.$$

Note that $\arg\min_j \boldsymbol{E}_j/\lambda_j \sim \text{Exp}(\sum_j \lambda_j)$. In order to derive the conditional density, we might want to take out the density of the smallest entry. Namely,

$$\begin{aligned} \prod_j \lambda_j e^{-\lambda_j x_j} &\overset{(*)}{=} \lambda_i e^{-\sum_j \lambda_j x_i} \prod_{j \neq i} \underbrace{\lambda_j e^{-\lambda_j (x_j - x_i)}}_{\text{is a density if } x_j \geq x_i} \\ &\overset{(**)}{=} \sum_i \frac{\lambda_i}{\sum_j \lambda_j} \sum_j \lambda_j e^{-\sum_j \lambda_j x_i} \prod_{j \neq i} \lambda_j e^{-\lambda_j (x_j - x_i)} \\ &\overset{(***)}{=} \sum_i \mathbb{P}(\boldsymbol{X}_i = \min_j \boldsymbol{X}_j) f(x_i | \boldsymbol{X}_i = \min_j \boldsymbol{X}_j) \prod_{j \neq i} f(x_j | \boldsymbol{X}_j \geq x_i) \\ &= \sum_i \mathbb{P}(\boldsymbol{X}_i = \min_j \boldsymbol{X}_j) f(x_1, ..., x_N | \boldsymbol{X}_i = \min_j \boldsymbol{X}_j). \end{aligned} \tag{23}$$

Note that $(*)$ holds for all i if we ignore making $\lambda_j e^{-\lambda_j (x_j - x_i)}$ a density. Because $\mathbb{P}(\boldsymbol{X}_i = \min_j \boldsymbol{X}_j) = \lambda_i / \sum_j \lambda_j$, $(**)$ rewrite the equation so that the expression is more relevant to $\min_j \boldsymbol{X}_j$ at Eq. (22). Finally, $(***)$ simply identifies $f(x_i | \boldsymbol{X}_i = \min_j \boldsymbol{X}_j) = \sum_j \lambda_j e^{-\sum_j \lambda_j x_i}$ and $f(x_j | \boldsymbol{X}_j \geq x_i) = \lambda_j e^{-\lambda_j (x_j - x_i)}$.

Since Eq. (23) implies

$$f(x_1, ..., x_N | \boldsymbol{X}_i = \min_j \boldsymbol{X}_j) = \sum_j \lambda_j e^{-\sum_j \lambda_j x_i} \prod_{j \neq i} \lambda_j e^{-\lambda_j (x_j - x_i)} = Z e^{-Z x_i} \prod_{j \neq i} e^{\ell_j} e^{-e^{\ell_j} (x_j - x_i)},$$

the random vector $\boldsymbol{E}/e^\ell$ conditioning on $\boldsymbol{D} = e_i$ is

$$[\boldsymbol{E}/e^\ell | \boldsymbol{D} = e_i]_j \overset{d}{=} \begin{cases} \frac{\boldsymbol{E}_i}{Z} & \text{if } j = i \\ \frac{\boldsymbol{E}_j}{e^j} + \frac{\boldsymbol{E}_i}{Z} & \text{o.w.} \end{cases}.$$

Taking $-\log(\cdot)$ on this expression concludes the proof. □

Lemma 1. *Gap*$(\theta_0|\boldsymbol{D}=e_i) = -\frac{\log(1-[p_{\theta_0}]_i)}{[p_{\theta_0}]_i} = \frac{\log\left(1+e^{\ell_i-s}\right)}{1-1/(1+e^{\ell_i-s})}$, *where* $s = \log\left(\sum_{j\neq i} e^{[logit_{\theta_0}]_j}\right)$ *is the log-sum-exponential of the unselected logits.*

Proof. Write the expected gap using the conditional logit expression Eq. (21).

$$\begin{aligned}
\text{Gap}(\theta_0|\boldsymbol{D}=e_i) =& \mathbb{E}\left[\min_{j\neq i} \ell_i + \boldsymbol{G}_i - (\ell_j+\boldsymbol{G}_j)\Big|\boldsymbol{D}=e_i\right] \\
\overset{(21)}{=}& \mathbb{E}\min_{j\neq i} -\log\left(\frac{\boldsymbol{E}_i}{Z}\right) + \log\left(\frac{\boldsymbol{E}_j}{e^{\ell_j}}+\frac{\boldsymbol{E}_i}{Z}\right) \\
\overset{[p_{\theta_0}]_j=e^{\ell_j}/Z}{=}& \mathbb{E}\min_{j\neq i} \log\left(1+\frac{\boldsymbol{E}_j}{\boldsymbol{E}_i[p_{\theta_0}]_j}\right) = \mathbb{E}\log\left(1+\frac{1}{\boldsymbol{E}_i}\min_{j\neq i}\frac{\boldsymbol{E}_j}{[p_{\theta_0}]_j}\right) \\
=& \mathbb{E}\log\left(1+\frac{\boldsymbol{E}_0}{\boldsymbol{E}_i\sum_{j\neq i}[p_{\theta_0}]_j}\right) = \mathbb{E}\log\left(1+\frac{\boldsymbol{E}_0}{\boldsymbol{E}_i(1-[p_{\theta_0}]_i)}\right).
\end{aligned} \tag{24}$$

The last line follows from $\frac{\boldsymbol{E}_j}{[p_{\theta_0}]_j} \sim \text{Exp}([p_{\theta_0}]_j)$ and $\min_j \text{Exp}(\lambda_j) \overset{d}{=} \text{Exp}(\sum_j \lambda_j)$. Note $\boldsymbol{E}_0 \sim \text{Exp}(1)$ is an exponential random variable that is independent of $\boldsymbol{E}_i$.

Consider the expectation of Eq. (24) w.r.t. $\boldsymbol{E}_0$. It is enough to evaluate the following.

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{E}_0}\left[\log\left(\frac{E_0}{r}+1\right)\right] &= \int_0^\infty \log\left(\frac{t}{r}+1\right)e^{-t}dt \\
&= e^r\int_0^\infty \log\left(\frac{t+r}{r}\right)e^{-(t+r)}dt \\
&= e^r\int_r^\infty \log\left(\frac{t}{r}\right)e^{-t}dt \\
&= e^r\left(\left[\log\left(\frac{t}{r}\right)e^{-t}\right]_{t=\infty}^{t=r} + \int_r^\infty \frac{1}{t}e^{-t}dt\right) = e^r\Gamma(0,r).
\end{aligned} \tag{25}$$

The last line uses integration by parts. Notice $\Gamma(0,r)=\int_r^\infty e^{-t}/t$ is the exponential integral as well as a special case of the incomplete gamma function $\Gamma(s,r)$. Since E denotes an exponential random variable, we write the exponential integral as $\Gamma(0,r)$. Thus, Eq. (24) becomes $\mathbb{E}e^{\boldsymbol{E}_i(1-[p_{\theta_0}]_i)}\Gamma(0,\boldsymbol{E}_i(1-[p_{\theta_0}]_i))$, and we turn to another expectation:

$$\begin{aligned}
\mathbb{E}e^{\boldsymbol{E}_i r}\Gamma(0,\boldsymbol{E}_i r) &= \int_0^\infty e^{rx}\Gamma(0,rx)e^{-x}dx = \int_0^\infty e^{(r-1)x}\int_{rx}^\infty \frac{1}{t}e^{-t}dtdx \\
&= \int_0^\infty\int_0^{t/r}\frac{1}{t}e^{-t}e^{(r-1)x}dxdt = \int_0^\infty \frac{1}{r-1}\frac{1}{t}\left(e^{-t/r}-e^{-t}\right)dt \\
&= \frac{1}{r-1}\lim_{z\to 0^+}\Gamma(0,z/r)-\Gamma(0,z) = \frac{\log(r)}{r-1}.
\end{aligned} \tag{26}$$

The last equality uses the following series expansion (Abramowitz et al., 1988)[p. 229, 5.1.11] and that $z\to 0^+$:

$$\Gamma(0,z) = -\gamma - \log(z) - \sum_{k=1}^\infty \frac{(-z)^k}{kk!} = -\gamma - \log(z) + O(z),$$

where $O(z)$ is the Big-O notation when $z\to 0^+$, indicating the set $\{f:\ |f(z)|\leq Mz \text{ as } z\to 0^+\}$ for $M>0$. This implies $\Gamma(0,z/r)-\Gamma(0,z) = \log(r)+O(z)$ and hence the limit of Eq. (26) goes to $\log(r)/(r-1)$.

Substitute Eq. (25) and (26) into Eq. (24), we arrive at

$$\text{Gap}(\theta_0|\boldsymbol{D}=e_i) = -\frac{\log(1-[p_{\theta_0}]_i)}{[p_{\theta_0}]_i}.$$

Furthermore, let $s = \log\left(\sum_{j\neq i} e^{\ell_j}\right)$ be the log-sum-exponential of the unselected logits. Then, $1-[p_{\theta_0}]_i = e^s/(e^{\ell_i}+e^s) = 1/(1+e^{\ell_i - s})$ and $[p_{\theta_0}]_i = 1 - 1/(1+e^{\ell_i-s})$. So we have an expression in logit difference:

$$\text{Gap}(\theta_0|\boldsymbol{D}=e_i) = \frac{\log\left(1+e^{\ell_i-s}\right)}{1-1/(1+e^{\ell_i-s})}.$$

□

A.2. An Alternative View to Lemma 1

Here, we provide an alternative view to Lemma 1 based on the equivalent unselected logit $s = \log(\sum_{j\neq i} e^{\ell_j})$. The analysis is less complicated than the proof of Lemma 1 and should give more intuition.

Recall that the gap between the top-2 largest logit is $\boldsymbol{G}_i + \ell_i - \max_{j\neq i}(\boldsymbol{G}_j + \ell_j)$ and that $\ell_j + \boldsymbol{G}_j \overset{d}{=} -\log(\boldsymbol{E}_j/e^{\ell_i})$ where $\boldsymbol{E}_j$'s are i.i.d. $\text{Exp}(1)$ variables. Thus, the gap is reorganized as

$$\begin{aligned}
&\boldsymbol{G}_i + \ell_i - \max_{j\neq i}(\boldsymbol{G}_j+\ell_j) \overset{d}{=} \boldsymbol{G}_i + \ell_i - \max_{j\neq i}\left(-\log\frac{\boldsymbol{E}_j}{e^{\ell_j}}\right)\\
=&\boldsymbol{G}_i + \ell_i - \left(-\log\left(\min_{j\neq i}\frac{\boldsymbol{E}_j}{e^{\ell_j}}\right)\right) \overset{d}{=} \boldsymbol{G}_i + \ell_i - \left(-\log\left(\frac{\boldsymbol{E}_0}{e^s}\right)\right)\\
=&\boldsymbol{G}_i + \ell_i - (\boldsymbol{G}_0 + s).
\end{aligned} \tag{27}$$

The second line uses $\frac{\boldsymbol{E}_j}{e^{\ell_j}} \sim \text{Exp}(e^{\ell_j})$ and $\min_{j\neq i}\text{Exp}(e^{\ell_j}) \overset{d}{=} \text{Exp}(e^s)$. $\boldsymbol{E}_0 \sim \text{Exp}(1)$ and is independent of $\boldsymbol{G}_i$. The last line defines $\boldsymbol{G}_0 = -\log(\boldsymbol{E}_0) \sim \text{Gumbel}(0,1)$ as another gumbel random variable.

This means that the smallest gap is equivalent to the perturbed difference between ℓ_i and $s = \log(\sum_{j\neq i} e^{\ell_j})$, and s becomes the equivalent logit that summarizes all unselected logits.

Since $\boldsymbol{G}_i,\ \boldsymbol{G}_0 \overset{i.i.d.}{\sim} \text{Gumbel}(0,1)$, we know $\boldsymbol{X} = \boldsymbol{G}_i - \boldsymbol{G}_0 \sim \text{Logistic}(0,1)$ and Eq. (27) is simply $\boldsymbol{X} - (s-\ell_i)$. Therefore, conditioning on $\boldsymbol{G}_i + \ell_i$ being the largest perturbed logit (i.e., $\boldsymbol{D} = e_i$), the expected gap is

$$\text{Gap}(\theta_0|\boldsymbol{D}=e_i) = \mathbb{E}[\boldsymbol{X} - d|\boldsymbol{X} - d \geq 0], \qquad d = s - \ell_i.$$

Note that $\text{Logistic}(0,1)$ has a density function $f(x) = e^{-x}/(1+e^{-x})^2$. We know $\mathbb{P}(\boldsymbol{X}\geq d) = 1 - 1/(1+e^{-d})$. The conditional expectation is evaluated as:

$$\begin{aligned}
&\mathbb{P}(\boldsymbol{X}\geq d)\mathbb{E}[\boldsymbol{X}|\boldsymbol{X}\geq d] = \int_d^\infty x\frac{e^{-x}}{(1+e^{-x})^2}dx = \int_d^\infty x\frac{e^x}{(1+e^x)^2}dx \overset{u=e^x}{=} \int_{1+e^d}^\infty \log(u-1)\frac{du}{u^2}\\
=&\frac{d}{1+e^d} + \int_{1+e^d}^\infty \frac{1}{u}\frac{1}{u-1}du = \frac{d}{1+e^d} - \log\frac{e^d}{1+e^d} = d\left(1-\frac{1}{1+e^{-d}}\right) + \log\left(1+e^{-d}\right).
\end{aligned}$$

Thereby, the expected gap equals to

$$\mathbb{E}[\boldsymbol{X} - d|\boldsymbol{X}\geq d] = d + \frac{\log\left(1+e^{-d}\right)}{1-1/(1+e^{-d})} - d = \frac{\log\left(1+e^{-d}\right)}{1-1/(1+e^{-d})} = \frac{\log\left(1+e^{\ell_i-s}\right)}{1-1/(1+e^{\ell_i-s})}, \tag{28}$$

which is exactly the same as Lemma 1.

Recall the expected gap converges to $\ell_i - s$ when $\ell_i - s \gg 0$ and to 1 when $\ell_i - s \ll 0$. We can understand this using the property of logistic distribution $\boldsymbol{X} \sim \text{Logistic}(0,1)$.

(a) When $-d = \ell_i - s \gg 0$, d is very negative and the conditional expectation $\mathbb{E}[\boldsymbol{X} - d|\boldsymbol{X}\geq d]$ is almost an unconditioned one, $\mathbb{E}[\boldsymbol{X} - d]$. Because $\mathbb{E}[\boldsymbol{X}] = 0$, the expected gap converges to $-d = \ell_i - s$.

(b) When $-d = \ell_i - s \ll 0$, d is very positive. Because $\text{Logistic}(0,1)$ has an exponential tail (this is evident from its density function), the conditional expectation $\mathbb{E}[\boldsymbol{X} - d|\boldsymbol{X}\geq d]$ is approximated by the exponential distribution: $\mathbb{E}[\boldsymbol{Y} - d|\boldsymbol{Y}\geq d]$ with $\boldsymbol{Y}\sim\text{Exp}(1)$. Since $\boldsymbol{Y}|\boldsymbol{Y}\geq d \overset{d}{=} \text{Exp}(1) + d$, the expected gap converges to 1 in this situation.

A.3. Additional Experiments for MNIST-VAE at Low Temperatures

As shown in section 5.1.2, the estimators are not easy to train at low temperatures. Still, low temperatures are potentially beneficial because the estimators are nearly one-hot, which reduces the functional mismatch between discrete one-hot vectors and the Softmax outputs. Thus, we demonstrate a training technique to assist the training at low temperatures below.

- When mod(batch index,M)$=0$, train at low temperatures (e.g., temp.=0.1 or 0.01).
- When mod(batch index,M)$\neq 0$, train at temperature mid_temp.
- Test at low temperatures.

M controls the frequency of the low-temperature training step. The mid temperature mid_temp is a temperature value at which the estimator is trained without any issue. The presence of mid_temp stabilizes the model after the potentially hazardous low-temperature training. We hence call this method the mixed temperature training.

For the experiments in Table 7, we select $(M, \text{mid\_temp})=(20, 0.5)$. Compared with Table 3, the mixed temperature training allows both GST and GR-MC100 to properly perform at low temperatures. It also improves the STGS at temperature 0.1 (Avg. Neg. ELBO ≈ 128 without mixed training and ≈ 119 with mixed training).

Temperature	Estimator	Neg. ELBO	Std.
0.1	STGS	119.38	2.36
	GR-MC100	117.70	3.64
	GST-0.8	111.33	0.95
	GST-1.0	107.73	1.10
	GST-1.2	107.36	0.78
	GST-pi	109.47	1.05
0.01	STGS	121.75	2.36
	GR-MC100	117.69	2.13
	GST-0.8	110.82	1.03
	GST-1.0	107.90	1.04
	GST-1.2	107.86	1.06
	GST-pi	109.08	1.26

Table 7. **Comparison of estimators on MNIST-VAE with mixed temperature training.** The results are evaluated over ten different random seeds.