Differentiable Sampling of Categorical Distributions Using the CatLog-Derivative Trick: Supplementary Material

Lennert De Smet
KU Leuven

Emanuele Sansone
KU Leuven

Pedro Zuidberg Dos Martires
Örebro University

A Experimental Details

We will give more details about our experimental setup in this section. We used two machines for all experiments. The first one has access to 4x RTX 3080 Ti coupled with an Intel Xeon Gold 6230R CPU @ 2.10GHz and 256 GB of RAM. The second one has 4x GTX 1080 Ti with an Intel Xeon CPU E5-2630 v4 @ 2.20GHz and 128 GB of RAM. All synthetic experiments and the MNIST and F-MNIST datasets for the DVAE experiment were run on the former, while Omniglot and the neural-symbolic experiment used the latter machine. Note that every single run of each experiment only used a single GPU.

A.1 Synthetic Experiments

Modelling. Both synthetic experiments directly modelled their respective distributions via parametrised logits. That is, the parameters are given by a $D \times K$ matrix of logits, where D and K are the number of independent components and the dimension of each individual variable, respectively.

Hyperparameters. We optimised the temperature value for the Gumbel-Softmax for both synthetic experiments and the learning rate for all methods for the second one via a grid search over $\{10, 1, 0.1, 0.01\}$. This yielded a temperature value of 1 for the first synthetic experiment and of 0.1 for the second one. In addition to this choice of initial temperature, a standard exponential annealing scheme [21] was also used during the synthetic optimisation experiment, multiplying the initial temperature value with a factor $e^{-0.05}$ every 20 iterations. Moreover, the Gumbel-Softmax gradients also required removal of diverging values to maintain their usability during optimisation. Finally, the learning rates for all methods were chosen from the values $\{10, 5, 1, 0.1, 0.01\}$ while using the RMSProp optimiser and ended up being different for almost all methods. Both RLOO-F and IndeCateR managed to perform best for a higher learning rate of 5 while RLOO-S was stable and performed best for a learning rate of 1. Gumbel-Softmax needed the smallest learning rate of 0.01 as it would otherwise exhibit extremely unstable behaviour, likely due to the sensitivity of the loss function in conjunction with its bias. The choice The encoder architecture indeed has 2 true hidden layers and one output layer. The choice of two samples for IndeCateR was made as it is the lowest number of samples that allows a sample-equivalent to RLOO as it is a multi-sample estimator.

A.2 Discrete Variational Auto-Encoder

Datasets. Three datasets were used in this experiment with different forms of pre-processing. First, the usual MNIST [9] dataset was binarised, as per usual in the discrete gradient estimation literature [18, 21]. Binarised means that all pixels are normalised to values between 0 and 1 by dividing by 255 followed by replacing all values strictly higher than 0.5 with one and those lower with 0. For F-MNIST [23], we kept the continuous form of the data, only normalising the pixels to values between 0 and 1. In this way, we also had a dataset with a continuous supervision signal. The

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

last dataset, Omniglot [8], was both downscaled and binarised. Every Omniglot image usually has dimensions $105 \times 105 \times 3$, which we reduced to greyscale 28×28 followed by a binarisation step as outlined earlier. All test sets are constructed in the same way from the dedicated test partitions of the respective datasets.

Modelling. All datasets used the exact same neural architecture for the discrete variational auto-encoder (DVAE). As the data all had the same input size of 28×28, the first layer flattened the input to size 784 followed by three dense layers of sizes 384, 256 and 200. The first two of these layers used the ReLU activation function while the last one had no non-linearity. The output of this final 200-dimensional layer predicted the logits of a 200-dimensional Bernoulli variable, of which samples are drawn. These samples then formed the input to the decoder component of the DVAE, consisting of another three dense layers of size 256, 384 and 784. Again, the first two layers utilised ReLU activation functions while the final layer had a linear activation. Finally, following the literature, the output of the decoder is interpreted as the logits for 784 binary random variables and optimised using an ELBO loss function, which is an expected value. The correct probabilities are given by the normalised and binarised pixel values of the original image.

Hyperparameters. As in the synthetic experiment, the Gumbel-Softmax required the most optimisation. The initial temperature value was set at 1 and annealed using an exponential decay with a factor of $e^{-0.01}$ applied at the start of every epoch. To guarantee stability, it was necessary to both limit the temperature to 0.1 in conjunction with removing diverging values from the gradients. This effort emphasises how the Gumbel-Softmax requires a lot of additional care to make it functional in practice compared to methods like RLOO or IndeCateR. Apart from the optimisation of the Gumbel-Softmax, we tried a learning rate of both 10^{-3} and 10^{-4} for all methods and ended up picking the latter because it generally provided lower final negated ELBO values due to its slower convergence. All methods used the Adam [7] optimiser. Note that all networks were initialised using the same random scheme and that no weight regularisation was applied to avoid obscuring the impact of using different estimators.

Additional plots and interpretations. We report additional results in Figure 1 for the DVAE, where we plot all metrics in terms of iterations. Using these figures, we can see that IndeCateR is very close to the performance of RLOO-F, but with orders of magnitude less samples (2 for IndeCateR compared to 800 for RLOO-F). This reduction in samples is also what makes IndeCateR perform better in time. The same overfitting behaviour can now also be seen on Omniglot for both IndeCateR and RLOO-F. However, in this case, the lowest negated test ELBO for both methods is lower than all competitors. This indicates that adding separate regularisation might further improve performance, as this regularisation is implicitly present in GS-S, GS-F and RLOO-S through either the continuous relaxations or higher variance.

A.3 Neural-Symbolic Optimisation

Dataset. The data for performing the MNIST addition experiment [16] on sequences of N digits is easily generated by randomly selecting N images from the MNIST dataset and concatenating them into a sequence. The supervision of such a sequence is equally easily obtained by summing up the labels of the sampled MNIST images. We only allow every MNIST image to appear once in all sequences, meaning the dataset contains $\lfloor 60000/N \rfloor$ sequences to learn from. A similar procedure holds for the test set, whose sequences we construct from the test set partition of the MNIST dataset.

Modelling. The neural network used for this experiment is a traditional LeNet [10] consisting of two convolutional layers with 6 and 16 filters of size 5 with ReLu activations followed by a flattening operation. Next, there are three dense layers of sizes 120, 84 and 10 of which the first two also have ReLU activations and the final one activates linearly. The output of this network for every image in the sequence yields the logits for the independent categorical distributions from which we sample. These samples are summed up and supervised to the correct sum. That is, a sampled sum is supervised to 1 if it matches the correct sum or 0 otherwise. More specifically, we optimise the negative log likelihood

$$-\log P(\sum_{i}^{N} D_i = s) = -\log \mathbb{E}_{\boldsymbol{D} \sim p(\boldsymbol{D})} \left[\mathbb{1}_{\sum_i^N D_i = s} \right]. \tag{A.1}$$

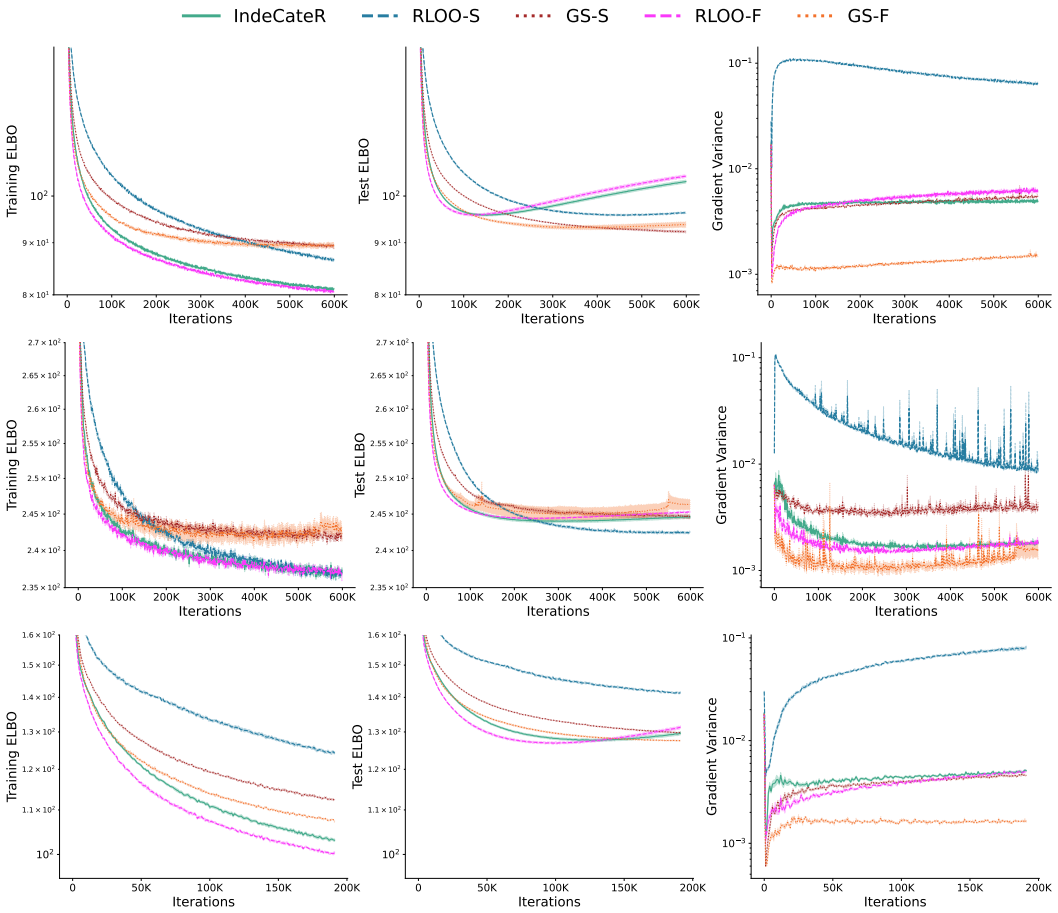

Figure 1: The top row shows negated training ELBO, test ELBO and gradient variance for the DVAE on MNIST – each plotted against iterations. Rows 2 and 3 show the same plots for F-MNIST (middle) and Omniglot (bottom). Omniglot only shows around 200K iterations because the dataset is around one third the size of MNIST and F-MNIST, and we fix the number of epochs to 1000.

Hyperparameters. We chose the standard Adam optimiser with a learning rate of 10^{-3} because of its dependable performance. The choice of 10 samples for IndeCateR was made so that every possible outcome of each digit is expected to appear once when sampling with uniform probabilities. The number of samples for the other methods is then fixed.

A.4 Brittleness of Gumbel-Softmax Estimates

During our experimental process, a number of noteworthy difficulties arose when applying the Gumbel-Softmax trick (GS). These difficulties show how applying the GS is far from plug-and-play and requires a substantial amount of hyperparameter tuning, at least to achieve performance comparable to other methods like RLOO.

Temperature tuning. While the bias of the GS can theoretically be controlled by annealing the temperature to zero, this process turns out to be rather cumbersome in practice. For example, in the DVAE experiments, we had to fix a lowest value for the temperature value beyond which it could not be annealed. If not, the ever smaller temperature would lead to numerical instability in the form of diverging gradients. Even when lower limiting the temperature, sporadic divergences of the gradient still needed to be clipped or they would instantly disrupt the full training process. In other words, finding a well-performing annealing scheme is non-trivial and does not guarantee a problem-free training process.

Bias and insensitive objectives. As observed during the synthetic optimisation experiment, the loss function itself can seemingly have a significant impact on the quality of GS gradient estimates. This experiment used a regular mean squared error loss function, which does not have any non-differentiabilities in its domain. However, the value of the loss function itself does not change much for different sampled values. Since the GS estimates are biased, we conjecture that there are cases where the bias overpowers the information from the loss function. In theory, annealing the temperature should eventually make the bias a non-issue, but the numerical issues raised in the previous paragraph prevented us from acquiring a satisfactory solution.

B Extended Related Work

There are a plethora of works in the literature of gradient estimators, for an in-depth discussion we refer the reader to the survey in Mohamed et al. [17]. This section will extend the related work given in the main paper.

Continuous relaxations. Diametrically opposite to REINFORCE-based methods are those based on continuous relaxations of categorical distributions. The concrete distribution [15] is a continuous relaxation of the categorical distribution using the Gumbel-Softmax trick [6]. Compared to variance reduction techniques based on control variates for REINFORCE, the Gumbel-Softmax trick has the advantage of being a single-sample estimator. However, it has other drawbacks. Firstly, it gives us a biased estimate of the gradients. Secondly, it necessitates the tuning of the temperature hyperparameter – either through hyperparameter search or a specific annealing schedule [6, 15]. Thirdly, when the derivative of the function inside the expectation is zero almost everywhere, such as the reward in reinforcement learning, the Gumbel-Softmax trick always returns zero gradients. Even though techniques have been developed to improve upon some of these shortcomings, such as Rao-Blackwellising the concrete distribution [19], the obtained improvements seem marginal. For a more detailed account of the concrete distribution and related methods we refer the reader to Huijben et al. [5].

Coupling multiple samples. Coupling approaches make better use of multiple samples by making them dependent. Initially, these approaches were applied to the case of a multivariate binary distribution [2] by, for example, using a logistic reparametrisation together with antithetic sampling [24, 3, 26]. Categorical extensions of these binary cases have also been proposed. One line of work [1] extends the binary version by representing a categorical distribution as a series of binary ones. Another [25] utilises a Dirichlet reparametrisation together with different kinds of coupled sampling. Our Cat-Log Derivative trick is also orthogonal to coupling methods, as it currently only considers independent samples.

Control variate methods. While some control variate techniques were mentioned in the main paper, a couple of more intricate contributions need to be mentioned. RELAX [4] and REBAR [22] are hybrid methods that use continuous relaxations to construct a control variate for REINFORCE. The work of Titsias, Michalis and Shi, Jiaxin [21] introduces double control variates; a global one for all samples and a local one for each sample separately. Our Cat-Log Derivative trick is orthogonal to control variate approaches in general and therefore both strategies could benefit from each other.

Rao-Blackwellisation. Apart from the RAM and LEG gradient estimators, there are two more lines of work that apply Rao-Blackwellisation. The works of Lorberbom et al. [13, 14] use a finite difference approximation to compute the gradient. Other approaches [12, 11] globally sum categories while sampling from the remainder. In contrast, our approach conditions the sampling at the level of the variables and can be therefore again be regarded as a complementary strategy.

C Comparing CatLog and Local Expectation Gradients

Local Expectation Gradients (LEG) [20] and the CatLog-Derivative trick both try to decompose the overall gradient of an expectation by exploiting distributional knowledge. However, there are a couple of fundamental theoretical differences, which we collect in the following statement.

Proposition C.1. (LEG Trick and Estimator) Given a factorised distribution $p(\mathbf{X}) = \prod_{d=1}^{D} p(X_d \mid \mathbf{X}_{<d})$, LEG with respect to a shared parameter $\lambda \in \Lambda$ leads to the expression

$$\begin{aligned}
&\mathbb{E}_{\mathbf{X}\sim p(\mathbf{X})}\left[f(\mathbf{X})\partial_\lambda \log p(\mathbf{X})\right] \\
&= \sum_{d=1}^{D} \mathbb{E}_{\mathbf{X}_{\neq d}\sim p(\mathbf{X}_{\neq d})}\left[\mathbb{E}_{X_d\sim p(X_d|\mathrm{MB}(X_d))}\left[f(\mathbf{X})\partial_\lambda \log p(X_d \mid \mathbf{X}_{<d})\right]\right] \qquad \text{(cf. Eq. 8 in [20])} \\
&= \sum_{d=1}^{D} \mathbb{E}_{\mathbf{X}_{<d}\sim p(\mathbf{X}_{<d})}\Bigg[\mathbb{E}_{X'_d\sim p(X'_d|\mathbf{X}_{<d})}\Bigg[\\
&\qquad \mathbb{E}_{\mathbf{X}_{>d}\sim p(\mathbf{X}_{>d}|X'_d,\mathbf{X}_{<d})}\left[\mathbb{E}_{X_d\sim p(X_d|\mathbf{X}_{\neq d})}\left[f(\mathbf{X})\partial_\lambda \log p(X_d \mid \mathbf{X}_{<d})\right]\right]\Bigg]\Bigg] \qquad \text{(C.1)}
\end{aligned}$$

where $\mathrm{MB}(X_d)$ is the Markov blanket of variable X_d and

$$p(X_d \mid \mathrm{MB}(X_d)) = p(X_d \mid \mathbf{X}_{\neq d}) = \frac{p(X_d \mid \mathbf{X}_{<d}) \prod_{j>d} p(X_j \mid \mathbf{X}_{<j})}{\sum_{x_\delta \in \Omega(X_d)} p(x_\delta \mid \mathbf{X}_{<d}) \prod_{j>d} p(X_j \mid \mathbf{X}_{<j})} \qquad \text{(C.2)}$$

is a weighting distribution. The Monte Carlo estimate corresponding to the LEG Trick in Equation (C.1) (cf. Algorithm 1 in [20]) given N pivot samples is given by the following expression:

$$\mathbb{E}_{\mathbf{X}\sim p(\mathbf{X})}\left[f(\mathbf{X})\partial_\lambda \log p(\mathbf{X})\right] \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{d=1}^{D}\sum_{x_\delta\in\Omega(X_d)} p(x_\delta \mid \mathbf{x}^{(n)}_{\neq d}) f(\mathbf{x}^{(n)}_{\neq d}, x_\delta)\partial_\lambda \log p(x_\delta \mid \mathbf{x}^{(n)}_{<d})$$

Additionally, the computational complexity is $\mathcal{O}(N \cdot D^2 \cdot K)$, with $K = \max_d(\mid \Omega(X_d) \mid)$.

Proof. We start by recalling the proof for the first equality in Equation (C.1).

$$\begin{aligned}
\mathbb{E}_{\mathbf{X}\sim p(\mathbf{X})}\left[f(\mathbf{X})\partial_\lambda \log p(\mathbf{X})\right] &= \mathbb{E}_{\mathbf{X}\sim p(\mathbf{X})}\left[f(\mathbf{X})\partial_\lambda \log \prod_{d=1}^{D} p(X_d \mid \mathbf{X}_{<d})\right] \\
&= \sum_{d=1}^{D} \mathbb{E}_{\mathbf{X}\sim p(\mathbf{X})}\left[f(\mathbf{X})\partial_\lambda \log p(X_d \mid \mathbf{X}_{<d})\right] \\
&= \sum_{d=1}^{D} \mathbb{E}_{\mathbf{X}_{\neq d}\sim p(\mathbf{X}_{\neq d})}\left[\underbrace{\mathbb{E}_{X_d\sim p(X_d|\mathbf{X}_{\neq d})}\left[f(\mathbf{X})\partial_\lambda \log p(X_d \mid \mathbf{X}_{<d})\right]}_{=\, g(\mathbf{X}_{\neq d})}\right]. \qquad \text{(C.3)}
\end{aligned}$$

By noting that $p(X_d \mid \mathbf{X}_{\neq d}) = p(X_d \mid \mathrm{MB}(X_d))$, we obtain the first result.

Regarding the second equality in Equation (C.1), we observe that

$$\mathbb{E}_{\mathbf{X}_{\neq d}\sim p(\mathbf{X}_{\neq d})}\left[g(\mathbf{X}_{\neq d})\right] \qquad \text{(C.4)}$$

in Equation (C.3) can be equivalently rewritten as

$$\begin{aligned}
\mathbb{E}_{\mathbf{X}_{\neq d}\sim p(\mathbf{X}_{\neq d})}\left[g(\mathbf{X}_{\neq d})\right] &= \mathbb{E}_{\mathbf{X}\sim p(\mathbf{X})}\left[g(\mathbf{X}_{\neq d})\right] \\
&= \mathbb{E}_{\mathbf{X}_{<d}\sim p(\mathbf{X}_{<d})}\mathbb{E}_{X'_d\sim p(X'_d|\mathbf{X}_{<d})}\mathbb{E}_{\mathbf{X}_{>d}\sim p(\mathbf{X}_{>d}|X'_d,\mathbf{X}_{<d})}\left[g(\mathbf{X}_{\neq d})\right] \qquad \text{(C.5)}
\end{aligned}$$

By plugging this result into Equation (C.3) we obtain the desired result for the second equality in Equation (C.1).

The Monte Carlo estimate trivially follows by sampling N pivots from the first three expectations in Equation (C.1) and by taking the exact weighted average of score evaluations.

Regarding computational complexity, we observe that computing the weighting function in the Monte Carlo estimate requires $\mathcal{O}(D)$ evaluations. Also, the estimate requires to iterate over three summations, thus having a computational requirement of $\mathcal{O}(N \cdot D \cdot K)$. By combining these two results, we obtain the overall complexity of $\mathcal{O}(N \cdot D^2 \cdot K)$. □

Table 1: Main differences between LEG and CatLog. The different nature of the applied tricks result in different computational complexities and related sampling strategies.

Name	Trick	Complexity	Relation to Sampling
LEG	Equation (C.6)	$\mathcal{O}(N \cdot D^2 \cdot K)$	Gibbs sampling
CatLog	Equation (C.7)	$\mathcal{O}(N \cdot D \cdot K)$	Ancestral sampling

Note that this result does not immediately follow from the expressions provided by Titsias, Michalis and Lázaro-Gredilla, Miguel [20], as they do not explicitly study the case of a shared parameter space across different variables. We do show, however, that their assumption of a separate parameter space can be removed, allowing a one-to-one comparison to the CatLog-Derivative trick. We give an overview of the main differences between LEG and the CatLog-Derivative trick in Table 1. To be precise, we will refer to the following two expressions for LEG and CatLog that clearly indicate the differences between the two.

$$\text{LEG} = \sum_{d=1}^{D} \mathbb{E}_{(\mathbf{X}_{<d}, X'_d, \mathbf{X}_{>d}) \sim p(\mathbf{X}_{<d}, X'_d, \mathbf{X}_{>d})} \left[\mathbb{E}_{\mathbf{X}_d \sim p(X_d \mid \mathbf{X}_{\neq d})} \left[f(\mathbf{X}) \partial_\lambda \log p(X_d \mid \mathbf{X}_{<d}) \right] \right] \tag{C.6}$$

$$\text{CatLog} = \sum_{d=1}^{D} \mathbb{E}_{(\mathbf{X}_{<d}, X_d, \mathbf{X}_{>d}) \sim p(\mathbf{X}_{<d}, X_d, \mathbf{X}_{>d})} \left[f(\mathbf{X}_{\neq d}, X_d) \partial_\lambda \log p(X_d \mid \mathbf{X}_{<d}) \right] \tag{C.7}$$

While the most important theoretical difference is the presence of the weighing factor $p(X_d \mid \text{MB}(X_d)) = p(X_d \mid \mathbf{X}_{\neq d})$, there is also a practical difference. Indeed, IndeCateR allows for new samples to be drawn for each variable while Algorithm 1 in [20] does not do so. This difference translates into a significant differential in performance as observed in the neural-symbolic experiment.

References

[1] Alek Dimitriev and Mingyuan Zhou. Carms: Categorical-antithetic-reinforce multi-sample gradient estimator. *NeurIPS*, 2021.

[2] Aleksandar Dimitriev and Mingyuan Zhou. Arms: Antithetic-reinforce-multi-sample gradient for binary variables. *ICML*, 2021.

[3] Zhe Dong, Andriy Mnih, and George Tucker. Disarm: An antithetic gradient estimator for binary latent variables. *NeurIPS*, 33, 2020.

[4] Will Grathwohl, Dami Choi, Yuhuai Wu, Geoffrey Roeder, and David Duvenaud. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. *ICLR*, 2018.

[5] Iris AM Huijben, Wouter Kool, Max B Paulus, and Ruud JG Van Sloun. A review of the gumbel-max trick and its extensions for discrete stochasticity in machine learning. *IEEE TPAMI*, 2022.

[6] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *ICLR*, 2017.

[7] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[8] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[9] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[10] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

[11] Chen Liang, Mohammad Norouzi, Jonathan Berant, Quoc V Le, and Ni Lao. Memory augmented policy optimization for program synthesis and semantic parsing. *NeruIPS*, 2018.

[12] Runjing Liu, Jeffrey Regier, Nilesh Tripuraneni, Michael Jordan, and Jon Mcauliffe. Rao-blackwellized stochastic gradients for discrete distributions. *ICML*, 2019.

[13] Guy Lorberbom, Andreea Gane, Tommi Jaakkola, and Tamir Hazan. Direct optimization through argmax for discrete variational auto-encoder. *NeurIPS*, 2019.

[14] Guy Lorberbom, Chris J Maddison, Nicolas Heess, Tamir Hazan, and Daniel Tarlow. Direct policy gradients: Direct optimization of policies in discrete action spaces. *NeurIPS*, 2020.

[15] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *ICLR*, 2017.

[16] Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. Deepproblog: Neural probabilistic logic programming. *advances in neural information processing systems*, 31, 2018.

[17] Shakir Mohamed, Mihaela Rosca, Michael Figurnov, and Andriy Mnih. Monte carlo gradient estimation in machine learning. *JMLR*, 2020.

[18] Mathias Niepert, Pasquale Minervini, and Luca Franceschi. Implicit mle: backpropagating through discrete exponential family distributions. *NeurIPS*, 2021.

[19] Max B Paulus, Chris J Maddison, and Andreas Krause. Rao-blackwellizing the straight-through gumbel-softmax gradient estimator. *ICLR*, 2021.

[20] Titsias, Michalis and Lázaro-Gredilla, Miguel. Local Expectation Gradients for Black Box Variational Inference. *NeurIPS*, 2015.

[21] Titsias, Michalis and Shi, Jiaxin. Double control variates for gradient estimation in discrete latent variable models. *AISTATS*, 2022.

[22] George Tucker, Andriy Mnih, Chris J Maddison, John Lawson, and Jascha Sohl-Dickstein. Rebar: Low-variance, unbiased gradient estimates for discrete latent variable models. *NeurIPS*, 2017.

[23] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[24] Mingzhang Yin and Mingyuan Zhou. Arm: Augment-reinforce-merge gradient for stochastic binary networks. *ICLR*, 2019.

[25] Mingzhang Yin, Yuguang Yue, and Mingyuan Zhou. Arsm: Augment-reinforce-swap-merge estimator for gradient backpropagation through categorical variables. *ICML*, 2019.

[26] Mingzhang Yin, Nhat Ho, Bowei Yan, Xiaoning Qian, and Mingyuan Zhou. Probabilistic best subset selection via gradient-based optimization. *arXiv*, 2020.