A IMPLEMENTING GRADIENT ESTIMATORS BY MODIFYING BACKPROPAGATION

An advantage of the GRMC-K estimator is the ease with which it can be implemented using automatic differentiation software. Here, we provide a pseudo code template for such an implementation.

```
class GRMCK(Function):

    def forward(logits, tau, k):
        sample = sampleOnehotCategorical(logits)
        save_for_backward(sample, logits, tau, k)
        return sample

    def backward(grad_output):
        sample, logits, tau, k = self.saved_objects
        logZ = logsumexp(logits)
        maxgumbel = getGumbel(logZ, k)
        tgumbels = getTruncatedGumbel(
                logits, k, sample, maxgumbel)
        gumbels = mergeGumbels(
                maxgumbel, tgumbels, sample)
        J = getSmaxJacobian(gumbels + logits).mean(0)
        return grad_output.matmul(J)
```

B IMPLEMENTING GRADIENT ESTIMATORS WITH THE SURROGATE LOSS FRAMEWORK

In this section, we consider an alternative framework for implementing the gradient estimators presented in the main body. This framework is due to (Schulman et al., 2015) and known as the surrogate loss framework. The key idea is that after the forward pass through a stochastic computation graph, all sampling decisions have been taken. Therefore, any gradient can be written as resulting from the differentiation of a surrogate objective in a deterministic computation graph.

Our exposition in the main body only considered a simplified scenario with a single discrete random variable. Therefore, we present here two cases, involving a layer of multiple and a cascade of discrete random variables. These two cases are general, because any case can be reduced to either of these two or a combination of them.

For ease of exposition, we again do not consider any direct dependence of f on the parameters of interest θ. The extension to this case is straight-forward and follows from basic calculus.

We also introduce the following notation to denote the stop of gradient flow. For $X^* = \texttt{stop\_gradient}(X)$ indicates that the gradient flow is interrupted at X and no gradient information is passed backward.

B.1 PARALLEL CASE

Let $D^1, \dots, D^m$ be a sequence of independent random variables. For $j \leq m$, let D^j be a discrete random variable $D^j \in \{0,1\}^n$ in a one-hot encoding, $\sum D^j_i = 1$, with distribution given by $p_{\theta^j}(D^j) \propto \exp({D^j}^T \theta)$ where $\theta^j \in \mathbb{R}^n$. Further, let S^j_τ be defined analogously to equation (3). Given a continuously differentiable $f : \mathbb{R}^{mn} \to \mathbb{R}$, we wish to minimize

$$\min_\theta \mathbb{E}\left[f(D^1, \dots D^m)\right], \tag{14}$$

where the expectation is taken over all m random variables.

In this setting, ∇_{REINF} can be computed by differentiating the following surrogate objec-

tive,

$$f(D^{1*}, \dots D^{m*}) \sum_{j=1}^{m} \log p_{\theta^j}(D^j) \tag{15}$$

In this setting, ∇_{GS} can be computed by differentiating the following surrogate objective,

$$f(S_\tau^1, \dots S_\tau^m) \tag{16}$$

In this setting, ∇_{ST} can be computed by differentiating the following surrogate objective,

$$\sum_{j=1}^{m} \left(\frac{\partial f(D^1, \dots D^m)}{\partial D^j} \right)^* \mathrm{softmax}_\tau(\theta^j) \tag{17}$$

In this setting, ∇_{STGS} can be computed by differentiating the following surrogate objective,

$$\sum_{j=1}^{m} \left(\frac{\partial f(D^1, \dots D^m)}{\partial D^j} \right)^* S_\tau^j \tag{18}$$

In this setting, $\nabla_{\mathrm{GRMC}K}$ can be computed by differentiating the following surrogate objective,

$$\sum_{j=1}^{m} \left(\frac{\partial f(D^1, \dots D^m)}{\partial D^j} \right)^* \left[\frac{1}{K} \sum_{k=1}^{K} S_\tau^{jk} \right] \tag{19}$$

B.2 Sequential Case

Let $D^1, \dots, D^m$ be a sequence of non-independent random variables. For $j \leq m$, let D^j be a discrete random variable $D^j \in \{0,1\}^n$ in a one-hot encoding, $\sum D_i^j = 1$, with distribution given by $p_{\theta^j}(D^j) \propto \exp({D^j}^T \theta^j)$ where $\theta^j \in \mathbb{R}^n$. For $2 \leq j \leq m$, let $\theta^j = h(D^{j-1})$, where $h : \mathbb{R}^n \to \mathbb{R}^n$ is a continuously differentiable function. Given a continuously differentiable $f : \mathbb{R}^{mn} \to \mathbb{R}$, we wish to minimize

$$\min_\theta \mathbb{E}\left[f(D^1, \dots, D^m) \right] \tag{20}$$

In this setting, ∇_{REINF} and ∇_{GS} can be computed by differentiating the surrogate objective given in the parallel case.
In this setting, ∇_{ST} can be computed by differentiating the following surrogate objective,

$$L_m \coloneqq \left(\frac{\partial f(D^1, \dots, D^m)}{\partial D^m} \right)^* \mathrm{softmax}_\tau(\theta^m) \tag{21}$$

$$L_j \coloneqq \left(\left(\frac{dL_{j+1}(D^1, \dots, D^m)}{dD^j} \right)^* + \left(\frac{\partial f(D^1, \dots, D^m)}{\partial D^j} \right)^* \right) \mathrm{softmax}_\tau(\theta^j) \tag{22}$$

In this setting, ∇_{STGS} can be computed by differentiating the following surrogate objective,

$$L_m \coloneqq \left(\frac{\partial f(D^1, \dots, D^m)}{\partial D^m} \right)^* \mathrm{softmax}_\tau(\theta^m + G^m) \tag{23}$$

$$L_j \coloneqq \left(\left(\frac{dL_{j+1}(D^1, \dots, D^m)}{dD^j} \right)^* + \left(\frac{\partial f(D^1, \dots, D^m)}{\partial D^j} \right)^* \right) \mathrm{softmax}_\tau(\theta^j + G^j) \tag{24}$$

In this setting, $\nabla_{\mathrm{GRMC}K}$ can be computed by differentiating the following surrogate objective,

$$L_m \coloneqq \left(\frac{\partial f(D^1, \dots, D^m)}{\partial D^m} \right)^* \left[\frac{1}{K} \sum_{k=1}^{K} \left(\mathrm{softmax}_\tau(\theta^m + G^{mk}) \right) \right] \tag{25}$$

$$L_j \coloneqq \left(\left(\frac{dL_{j+1}(D^1, \dots, D^m)}{dD^j} \right)^* + \left(\frac{\partial f(D^1, \dots, D^m)}{\partial D^j} \right)^* \right) \left[\frac{1}{K} \sum_{k=1}^{K} \left(\mathrm{softmax}_\tau(\theta^j + G^{jk}) \right) \right] \tag{26}$$

C PROOFS FOR THE PROPOSITIONS

In this section, we provide derivations for all the propositions given in the main body.

C.1 PROPOSITION 1

The derivation is based on Jensen's inequality and the law of iterated expectations.

Proof.

$$\mathbb{E}\left[\left\|\nabla_{\mathrm{GR}}-\nabla_\theta\right\|^2\right]=\mathbb{E}\left[\left\|\mathbb{E}\left[\nabla_{\mathrm{STGS}}|D\right]-\nabla_\theta\right\|^2\right] \tag{27}$$

$$=\mathbb{E}\left[\left\|\mathbb{E}\left[\nabla_{\mathrm{STGS}}-\nabla_\theta|D\right]\right\|^2\right] \tag{28}$$

$$\leq\mathbb{E}\left[\mathbb{E}\left[\left\|\nabla_{\mathrm{STGS}}-\nabla_\theta\right\|^2|D\right]\right] \tag{29}$$

$$=\mathbb{E}\left[\left\|\nabla_{\mathrm{STGS}}-\nabla_\theta\right\|^2\right] \tag{30}$$

□

The inequality is strict whenever $\mathrm{var}\left[\nabla_{\mathrm{STGS}}|D\right]>0$, which is the case if $\tau<\infty$ and $|\theta_i|<\infty$ for all $i\leq n$.

C.2 PROPOSITION 2

The derivation is based on Jensen's inequality and the linearity of expectations. For ease of exposition, denote by $\nabla_{\mathrm{STGS}}\left(S^k|D\right)$ a particular realization of the ST-GS estimator for a given D.

Proof.

$$\mathbb{E}\left[\left\|\nabla_{\mathrm{GRMC}K}-\nabla_\theta\right\|^2\right]=\mathbb{E}\left[\left\|\frac{1}{K}\sum_{k=1}^{K}\nabla_{\mathrm{STGS}}\left(S^k|D\right)-\nabla_\theta\right\|^2\right] \tag{31}$$

$$=\mathbb{E}\left[\left\|\frac{1}{K}\sum_{k=1}^{K}\left(\nabla_{\mathrm{STGS}}\left(S^k|D\right)-\nabla_\theta\right)\right\|^2\right] \tag{32}$$

$$\leq\mathbb{E}\left[\frac{1}{K}\sum_{k=1}^{K}\left\|\nabla_{\mathrm{STGS}}\left(S^k|D\right)-\nabla_\theta\right\|^2\right] \tag{33}$$

$$=\frac{1}{K}\sum_{k=1}^{K}\mathbb{E}\left[\left\|\nabla_{\mathrm{STGS}}\left(S|D\right)-\nabla_\theta\right\|^2\right] \tag{34}$$

$$=\mathbb{E}\left[\left\|\nabla_{\mathrm{STGS}}-\nabla_\theta\right\|^2\right] \tag{35}$$

□

The inequality is strict whenever $K>1$ and $\mathrm{var}\left[\nabla_{\mathrm{STGS}}|D\right]>0$, which is the case if $\tau<\infty$ and $|\theta_i|<\infty$ for all $i\leq n$.

C.3 PROPOSITION 3

The derivation is based on the law of total variance.

Proof.

$$\text{var}\left[\overline{\nabla}^{1:B}_{\text{GRMC}K}\right] = \mathbb{E}\left[\text{var}\left[\overline{\nabla}^{1:B}_{\text{GRMC}K}|D_{1:B}, X_{1:B}\right]\right] + \text{var}\left[\mathbb{E}\left[\overline{\nabla}^{1:B}_{\text{GRMC}K}|D_{1:B}, X_{1:B}\right]\right] \tag{36}$$

$$= \mathbb{E}\left[\text{var}\left[\frac{1}{B}\sum_{b=1}^{B}\nabla^{b}_{\text{GRMC}K}\middle|D_b, X_b\right]\right] + \text{var}\left[\mathbb{E}\left[\frac{1}{B}\sum_{b=1}^{B}\nabla^{b}_{\text{GRMC}K}\middle|D_b, X_b\right]\right] \tag{37}$$

$$= \mathbb{E}\left[\frac{1}{B^2}\sum_{b=1}^{B}\text{var}\left[\nabla^{b}_{\text{GRMC}K}\middle|D_b, X_b\right]\right] + \text{var}\left[\mathbb{E}\left[\frac{1}{B}\sum_{b=1}^{B}\nabla^{b}_{\text{GRMC}K}\middle|D_b, X_b\right]\right] \tag{38}$$

$$= \frac{1}{B}\,\mathbb{E}\left[\text{var}\left[\nabla_{\text{GRMC}K}|D, X\right]\right] + \text{var}\left[\frac{1}{B}\sum_{b=1}^{B}\mathbb{E}\left[\nabla^{b}_{\text{GRMC}K}|D_b, X_b\right]\right] \tag{39}$$

$$= \frac{1}{B}\,\mathbb{E}\left[\frac{1}{K}\,\text{var}\left[\nabla_{\text{STGS}}|D, X\right]\right] + \frac{1}{B}\,\text{var}\left[\mathbb{E}\left[\nabla_{\text{GRMC}K}|D, X\right]\right] \tag{40}$$

$$= \frac{1}{BK}\,\mathbb{E}\left[\text{var}\left[\nabla_{\text{STGS}}|D, X\right]\right] + \frac{1}{B}\,\text{var}\left[\nabla_{\text{GR}}\right] \tag{41}$$

□

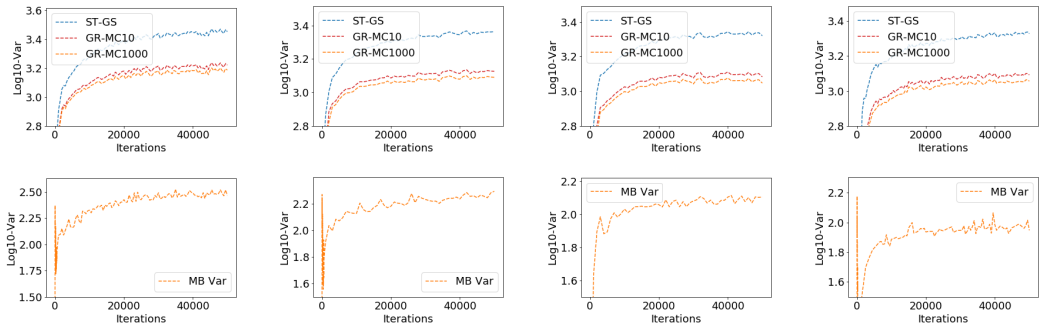
D Experimental Details

D.1 Unsupervised Parsing on ListOps

For our unsupervised parsing expeiment on ListOps, we use the basic version of the model described in Choi et al. (2017) with an embedding dimension and hidden dimension of 128. We do not use the *leaf-rnn*. We do not use the *intra-attention module*. We do not use dropout, but set weight decay to be $1e-4$. Because our interest is in using this experiment primarily as a testbed to evaluate the effectiveness of different gradient estimators for this model at different temperatures and for trees of different depth, we use a very simple experimental set-up. We rely on stochastic gradient descent without momentum to train all models. We use grid search to determine an optimal learning rate from $\{0.1, 0.2, \ldots 1.0\}$ and set the temperature τ to be in $\{0.01, 0.1, 1.0\}$. We repeat five independent random runs at each setting and report the mean over the five runs. We train for ten epochs and set the batch size to be equal to the maximum sequence length L.

Havrylov et al. (2019) also consider unsupervised parsing on ListOps with a variant of the model in Choi et al. (2017). They achieve near perfect accuracy, albeit in a highly customized experimental set-up. We list the most important differences below:

- Havrylov et al. (2019) does not use single-evaluation estimators, we do: They report near perfect accuracy only when using the self-critical baseline. This baseline requires an additional forward pass. All their single-evaluation results are in a similar ballpark as ours accounting for the additional differences below.
- Havrylov et al. (2019) uses extensive hyperparameter tuning, we do not: They tune learning rate, learning rate schedule, weight decay, entropy regularisation, variance reduction hyper-parameters, optimizer (Adadelta), number of updates for PPO, leaf transformations and train for 300 epochs. In contrast, we only tune the (constant) learning rate via gridsearch and use SGD (see above) for each temperature and train for ten epochs.
- Havrylov et al. (2019) uses customized training procedures, we are simply plug-in: They use PPO, gradient normalization, different control variates and entropy regularization. We simply plug our estimator into the model from Choi et al. (2017).
- Havrylov et al. (2019) uses a model with more parameters than us: We do not use any leaf LSTM. It improves performance (Choi et al., 2017), but may also confound tree learning (e.g., leaf-LSTM may learn to solve the task, making tree obsolete), so we do not use it.
- Havrylov et al. (2019) uses more training data than us: We reserved 10% of the training set for validation, while they use less than 2%.



(a) (b) (c) (d)

Figure 3: Our estimator (GR-MCK) effectively reduces the variance over the entire training trajectory at all arities. The variance reduction compares favorable to the minibatch variance. Columns correspond to arities, i.e. (a) binary, (b) 4-ary, (c) 8-ary, (d) 16-ary. First row, log10-trace of MC covariance matrix for various gradient estimators over iterations. Second row, log10-trace of MB covariance matrix over iterations (same for all gradient estimators).

D.2 Generative Modelling with Variational Auto-Encoders

We trained variational auto-encoders with n-ary discrete random variables with values on the corners of the hypercube $\{-1, 1\}^{\log_2(n)}$. The model with arity $\{2, 4, 8, 16\}$ included $\{240, 120, 80, 60\}$ random variables respectively.

All models were optimized using stochastic gradient descent with momentum for 50000 steps on minibatches of size 20 and 200 respectively. Hyperparameters were randomly sampled and the best setting was selected from twenty independent runs. Learning rate and momentum were randomly sampled from $\{5, 6, \dots 50\} \times 10^{-4}$ and $(0, 1)$ respectively. We did not anneal the learning rate during training. For regularising the network, we used weight-decay, which was randomly sampled from $\{0, 10^{-1}, 10^{-2} \dots, 10^{-6}\}$. The temperature was randomly sampled from $[0.1, 1.0]$ and not annealed throughout training.

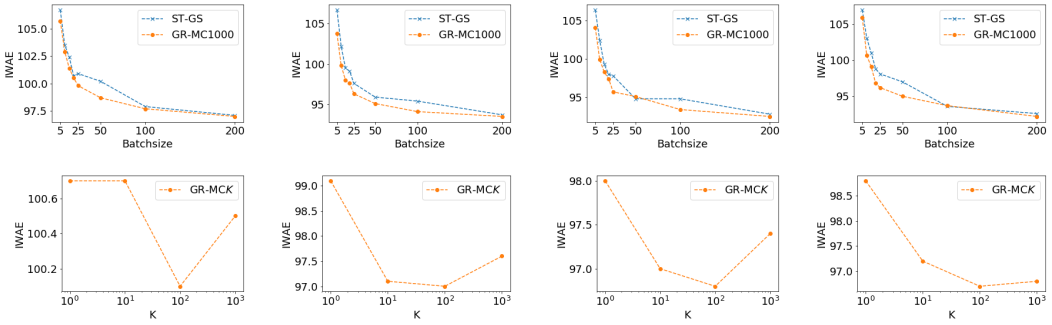
All models were evaluated on the validation and test set using the importance-weighted bound on the log-likelihood described in Burda et al. (2015) with 5000 samples.

To estimate the variance of a gradient estimator in the VAE experiment we used 5000 randomly sampled minibatches of size 20, for each of which we performed 100 independent forward passes and then computed the associated gradient for the parameters of the inference network. We then summed the variance to get a singe scalar measurement.

To estimate the bias of a gradient estimator in the VAE experiment, we proceeded as above to approximate the expectation for a gradient estimator. We approximated the true gradient by following this procedure for the REINFORCE algorithm.

To assess training speed, we measured the average number of iterations needed to achieve a pre-specified loss threshold on the validation set. In particular, we ran multiple independent runs under the same experimental conditions for all gradient estimators. Among only runs that achieved the threshold within the total budget, we report the average number of iterations taken to cross the threshold.

E Additional Figures



(a) (b) (c) (d)

Figure 4: Increasing the number of Monte Carlo samples K to reduce variance in gradient estimation tends to improve performance. The performance difference tends to be larger at smaller batch sizes. Columns correspond to arities, i.e. (a) binary, (b) 4-ary, (c) 8-ary, (d) 16-ary. First row, IWAE on test set for best validated model trained at various batch sizes. Second row, IWAE on test set for best validated model trained at various K at batch size 20.

Table 3: Our estimator, GR-MCK, consistently achieves better performance across arities and batchsizes. The outperformance tends to be larger at smaller batchsizes. Best bound on the negative log-likelihood selected on the validation set from 20 independent runs at randomly searched hyperparameters.

		BINARY		4-ARY		8-ARY		16-ARY	
	ESTIMATOR	VALID.	TEST	VALID.	TEST	VALID.	TEST	VALID.	TEST
BATCH-SIZE 5	ST-GS	107.7	106.7	107.8	106.7	107.5	106.4	108.1	107.0
	GR-MC1000	**106.7**	**105.7**	**104.7**	**103.8**	**105.1**	**104.1**	**107.0**	**105.9**
BATCH-SIZE 10	ST-GS	104.4	103.5	103.2	102.2	103.5	102.4	104.1	103.1
	GR-MC1000	**103.7**	**102.9**	**100.8**	**99.8**	**100.9**	**99.9**	**101.8**	**100.7**
BATCH-SIZE 15	ST-GS	103.4	102.4	100.4	99.5	100.3	99.3	101.9	101.0
	GR-MC1000	**102.3**	**101.4**	**99.0**	**98.0**	**99.2**	**98.3**	**100.2**	**99.1**
BATCH-SIZE 20	ST-GS	101.5	100.7	100.0	99.1	99.0	98.0	99.8	98.8
	GR-MC1000	**101.3**	**100.5**	**98.4**	**97.6**	**97.5**	**96.5**	**97.8**	**96.8**
BATCH-SIZE 25	ST-GS	101.7	100.9	98.6	97.6	98.8	97.8	99.0	98.1
	GR-MC1000	**100.7**	**99.8**	**97.2**	**96.3**	**96.6**	**95.7**	**97.1**	**96.2**
BATCH-SIZE 50	ST-GS	101.2	100.2	96.7	95.9	**95.7**	**94.8**	98.0	97.0
	GR-MC1000	**99.5**	**98.7**	**96.0**	**95.1**	95.9	95.1	**95.9**	**95.0**
BATCH-SIZE 100	ST-GS	98.8	97.9	96.3	95.4	95.7	94.8	**94.4**	**93.6**
	GR-MC1000	**98.5**	**97.7**	**95.0**	**94.1**	**94.3**	**93.4**	94.6	93.7
BATCH-SIZE 200	ST-GS	97.9	97.1	94.5	93.7	93.6	92.8	93.4	92.6
	GR-MC1000	**97.8**	**97.0**	**94.3**	**93.5**	**93.2**	**92.5**	**93.1**	**92.2**