R2: Relation to Tokui & Sato (2017) [31]. Their RAM estimator for each discrete variable and each its state needs to recompute the states of all dependent variables [31, Alg.1] and therefore scales quadratically with the number of variables despite the sample of the noises being drawn only once. It is proposed in order to study the quality of control variate techniques. We have compared with several tractable control-variate techniques in Appendix Fig. C.3 and Fig. C.5.

The Gumble-max reparametrization resolves dependencies between latent noises only. When flipping the discrete state z_i for fixed latent noises ϵ_{-i}, the dependent discrete variables and the objective function may still change causing the quadratic complexity of [31 Alg.1]. To avoid confusion, we should clarify that the Gumble-max reparametrization is not differentiable and the "reparameterization trick" is never used in their RAM method in contrast to what the abstract says.

In context of L109 "extending the linearization construction [31]" we mean extending their derivation of Straight-Through [31 sec. 6.4], where it is assumed that the loss function f is differentiable in each discrete variable z_i (and does not depend on it through a chain of other discrete variables) and thus can be linearized [31 eq. 8]. This derivation is applicable to one layer only, in which case it matches our ST. Hence there is nothing to compare experimentally. This result is a side observation in this paper, indeed not noticed in alternative explanations of ST [4, 36].

R2, R4: Variational autoencoders. Thanks, we will correct to "deep autoencoders with stochastic binary codes".

R3: Experimental setup. We will describe the generation procedure. Points of class 1 (resp. 2) are uniformly distributed above $y = 0$ (resp. below $y = \cos(x)$). The implementation is available in gradeval/expclass.py. The data is shown in Fig. B.1 (a). We have experimented with several configurations varying the number of units and layers. Generally, with a smaller number of units ARM gets more accurate and ST gets less accurate, but the overall picture stays. The displayed results are actually for a 5-5-5 configuration as described in Appendix C.1 (mentioning 5-3-3 in L224 is a typo). We will extend the appendix to show more cases varying the number of units / layers.

Performace. We address only the training performance and not the generalization performance, which in practice involves batch norm, pretraining and architecture search. However the method does achieve the best accuracy and the fastest convergence in iterations in comparison with other training methods under the same setup.

The ST method is previously proposed. We disagree. As we discuss in the related work, it has been proposed as a practical hack in several different variants. Other works attempted studying its properties. We derive it for deep models. In our view we are the first to propose it as a formal method.

Test set performance. In our experiments all hyperparameters including the learning rates are tuned exclusively on the training set (see Appendix C.2). Hence the validation set provides an unbiased estimate of the test error.

ARM on CIFAR. ARM has a prohibitive complexity for deep models (see L90). We expect it to have high variance for deep models as well. See also appendix L643-655 comparing to MuProp.

Computation complexity. Our complexity analysis (Proposition 2, proof in Appendix B.5) shows that the required computation for all flips has the same complexity as standard forward propagation. Additionally, in Appendix B.5 we show how to overload backprop operations to achieve the FLOPs complexity as low as 2x standard backprop. Additionally, in L664 of appendix we report all running times with the currently provided implementation (using GPU but suboptimal).

Single layer case. The single layer case is well covered in the literature [5, 30, 31], we also detail it in Appendix B2 (L453-461).

R4. More advanced experiments. We face here the situation that ST methods have been already applied successfully to deep residual networks, e.g. on ImageNet. We do not expect to beat them simply by an improved gradient estimator without dealing with learning schedules, pretraining, designing special architectures, etc. We will explore this and other applications in the future work.

Beyond logistic noise. All theory applies seamlessly to any continuous noise distribution. The logistic noise is really being used only in eq. (49-51) to optimize the implementation (affects constant complexity factors).

Feedback. Thanks, we will discuss limitations and possible applications.