REINTRODUCING STRAIGHT-THROUGH ESTIMATORS AS PRINCIPLED METHODS FOR STOCHASTIC BINARY NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Training neural networks with binary weights and activations is a challenging problem due to the lack of gradients and difficulty of optimization over discrete weights. Many successful experimental results have been achieved with empirical straight-through (ST) approaches, proposing a variety of ad-hoc rules for propagating gradients through non-differentiable activations and updating discrete weights. At the same time, ST methods can be truly derived as estimators in the stochastic binary network (SBN) model with Bernoulli weights. We advance these derivations to a more complete and systematic study. We analyze properties, estimation accuracy, obtain different forms of correct ST estimators for activations and weights, explain existing empirical approaches and their shortcomings, explain how latent weights arise from the mirror descent method when optimizing over probabilities. This allows to reintroduce, once empirical, ST methods as sound approximations, apply them with clarity and develop further improvements.

1 INTRODUCTION

Neural networks with binary weights and activations have much lower computation costs and memory consumption than their real-valued counterparts (Horowitz, 2014; Esser et al., 2016; Rastegari et al., 2016). They are therefore very attractive for applications in mobile devices, robotics and other resource-limited settings, in particular for solving vision and speech recognition problems (Bulat & Tzimiropoulos, 2017; Xiang et al., 2017).

The state of the art in training deep binary networks is established with methods using the empirical straight-through gradient estimation approach. Typically, the sign function is used to represent binary activations. This poses an immediate problem as the derivative of a step function like sign is zero almost everywhere and is not useful for training with backpropagation. The *empirical straight-through* approach consists in using the sign function on the forward pass and the derivative of some other *proxy* function on the backward pass. One popular solution is to use *identity* proxy, *i.e*. completely bypass sign on the backward pass (Krizhevsky & Hinton, 2011; Bengio et al., 2013; Courbariaux et al., 2015; Zhou et al., 2016), hence the name straight-through. Other proxy functions applied include $\tanh$ (Hinton, 2012; Raiko et al., 2015), clipped identity (Hubara et al., 2016; Lin et al., 2017; Rastegari et al., 2016; Alizadeh et al., 2019), or piece-wise quadratic $\mathrm{ApproxSign}$ (Esser et al., 2016; Liu et al., 2018), illustrated in Fig. A.1. These proxies can be used with deterministic or stochastic binarization (Courbariaux et al., 2015; Hubara et al., 2016; Krizhevsky & Hinton, 2011). This gives rise to a diversity of empirical ST methods, where various choices are studied purely experimentally (Alizadeh et al., 2019; Bethge et al., 2019; Tang et al., 2017).

Since binary weights can be also represented as a sign mapping of some real-valued *latent weights*, the same type of methods was adopted to handle optimization over binary weights (Shayer et al., 2017; Courbariaux et al., 2015; Hubara et al., 2016). However, often a different proxy is used for the weights along with clipping or not clipping latent weights to the interval $[-1, 1]$, generating additional unclear choices. With such obscurity of the concept of latent weights, Helwegen et al. (2019) argues that "latent weights do not exist" meaning that discrete optimization over binary weights needs to be considered and proposes an equally unjustified alternative scheme.

A more principled approach to handle binarization is to consider *stochastic binary networks* (SBNs), in which hidden units and/or weights are Bernoulli random variables. In these models the expected loss is a truly differentiable function of parameters (resp. weight probabilities) and its gradient can be estimated. Advanced unbiased gradient estimators exist (Grathwohl et al., 2018; Tucker et al., 2017; Yin & Zhou, 2019), which however have never been applied in practice to networks with deep binary dependencies due to increased variance in deep layers and complexity that grows quadratically with the number of layers (Shekhovtsov et al., 2020). ST can be defined in SBNs as a biased estimator, however this form is not well-known and no systematic study exists. More specifically, during our research on ST we found out that Tokui & Sato (2017); Shekhovtsov et al. (2020); Dai et al. (2017) already showed how to *derive* ST under clear approximations in SBNs. These results however are secondary in these papers and unnoticed in the works applying ST in practice, recent works on analysis of deterministic ST (Yin et al., 2019; Cheng et al., 2019) and works on alternative estimators (Cheng et al., 2019). They are not properly connected to empirical ST variants and do not perform any analysis. This is the main context for our work. More related works, including alternative approaches, are discussed in § A.

Contribution The goal of this work is to reintroduce straight-through estimators in a principled way, clarify, systematize and study the empirical ST approaches. We build on the derivations of ST for shallow (Tokui & Sato, 2017) and deep (Shekhovtsov et al., 2020) networks and the mirror descent optimization view for deterministic binary weights (Ajanthan et al., 2019).

We review sound results allowing to derive ST estimator in SBNs (§§ 2.1 and 4); Analyze this derived ST for models with one hidden layer and show shortcomings of and equivalences between existing empirical approaches (§ 2.2); Define latent weights in a sound optimization scheme and establish a rigorous correspondence between SGD with identity ST and powerful optimization methods such as mirror descent and variational Bayesian learning (§ 3); Experimentally study the accuracy of derived ST, confirming the theory and revealing useful insights (§ 5). Experimentally show that with the derived ST estimators and the proposed initialization and optimization, several common choices of gradient rules can be closely replicated, but unlike the empirical variants lead to equally well-performing methods (§ 5). As accompanying results, we analyze *Gumbel-Softmax* estimator (§ E) and the BayesBiNN method (Meng et al., 2020) that applies this estimator for SBNs, showing an unexpected relation of the latter to ST (§ F).

2 Single Layer Straight-Through

Notation Throughout the paper we will define several models that work with binary random variables. We will model random states $\boldsymbol{x} \in \{-1, 1\}^n$ using the noisy sign mapping:

$$x_i = \operatorname{sign}(a_i - z_i), \tag{1}$$

where z_i are real-valued independent noises with a fixed cdf F and a_i are (input-dependent) parameters. It is seen that the difference from deterministic binary models is only in the presence of injected noises $\boldsymbol{z}$. Equivalently, x_i can be described by probabilities $p(x_i{=}1 \,|\, a_i) = \mathbb{P}(a_i{-}z_i \geq 0 \,|\, a_i) = \mathbb{P}(z_i \leq a_i \,|\, a_i) = F(a_i)$. The noise distribution F will play an important role in understanding different schemes. If the noise is logistic, then F is the logistic sigmoid function σ.

2.1 Derivation

The ST was first introduced (Hinton, 2012; Krizhevsky & Hinton, 2011) in the context of a stochastic autoencoder model, highly relevant to date (*e.g.* Dadaneh et al., 2020), which we consider. We give a self-contained derivation of ST clarifying the key steps of Tokui & Sato (2017, sec. 6.4).

Let $\boldsymbol{y}$ denote observations. The *encoder network* parametrized by $\boldsymbol{\phi}$ computes logits $\boldsymbol{a}(\boldsymbol{y}; \boldsymbol{\phi})$ and computes a binary latent state $\boldsymbol{x}$ via (1). As noises $\boldsymbol{z}$ are independent, the conditional distribution factors as $p(\boldsymbol{x}|\boldsymbol{y}; \boldsymbol{\phi}) = \prod_i p(x_i|\boldsymbol{y}; \boldsymbol{\phi})$. The *decoder* reconstructs observations with $p^{\text{dec}}(\boldsymbol{y}|\boldsymbol{x}; \boldsymbol{\theta})$, another neural network parametrized by $\boldsymbol{\theta}$. The reconstruction loss of this autoencoder is defined as

$$\mathbb{E}_{\boldsymbol{y}\sim\text{data}}\big[\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x}|\boldsymbol{y};\boldsymbol{\phi})}[-\log p^{\text{dec}}(\boldsymbol{y}|\boldsymbol{x}; \boldsymbol{\theta})]\big]. \tag{2}$$

We will focus on the problem of computing the gradient of this loss w.r.t. the encoder parameters $\boldsymbol{\phi}$ (differentiating in $\boldsymbol{\theta}$ is easy as it does not affect the distribution if samples). The problem for a fixed

observation $\boldsymbol{y}$ takes the form

$$\frac{\partial}{\partial \boldsymbol{\phi}} \mathbb{E}_{\boldsymbol{x} \sim p(\boldsymbol{x};\boldsymbol{\phi})}[\mathcal{L}(\boldsymbol{x})] = \frac{\partial}{\partial \boldsymbol{\phi}} \mathbb{E}_{\boldsymbol{z}}[\mathcal{L}(\operatorname{sign}(\boldsymbol{a} - \boldsymbol{z}))], \tag{3}$$

where $p(\boldsymbol{x};\boldsymbol{\phi})$ is a shorthand for $p(\boldsymbol{x}|\boldsymbol{y};\boldsymbol{\phi})$ and $\mathcal{L}(\boldsymbol{x}) = -\log p^{\text{dec}}(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta})$. The reparametrization trick, *i.e.* to draw one sample of $\boldsymbol{z}$ in (3) and differentiate $\mathcal{L}(\operatorname{sign}(\boldsymbol{a} - \boldsymbol{z}))$ fails: since the loss as a function of $\boldsymbol{a}$ and $\boldsymbol{z}$ is not *continuously differentiable we cannot interchange the gradient and the expectation in* $\boldsymbol{z}$ *in the first place*[1]. If we nevertheless do make the interchange, we obtain that the gradient of $\operatorname{sign}(\boldsymbol{a} - \boldsymbol{z})$ is zero and the result is obviously incorrect. Instead, the following steps lead to an unbiased low variance estimator. From the LHS of (3) we express the derivative as

$$\frac{\partial}{\partial \boldsymbol{\phi}} \sum_{\boldsymbol{x}} \big(\prod_i p(x_i;\boldsymbol{\phi})\big)\mathcal{L}(\boldsymbol{x}) = \sum_{\boldsymbol{x}} \sum_i \big(\prod_{i' \neq i} p(x_{i'};\boldsymbol{\phi})\big)\big(\frac{\partial}{\partial \boldsymbol{\phi}} p(x_i;\boldsymbol{\phi})\big)\mathcal{L}(\boldsymbol{x}). \tag{4}$$

Then the *derandomization* (Owen, 2013, ch. 8.7) is applied which performs summation over x_i explicitly for the rest of the states in $\boldsymbol{x}$ fixed. Because x_i takes only two values, this gives

$$\sum_{x_i} \frac{\partial p(x_i;\boldsymbol{\phi})}{\partial \boldsymbol{\phi}} \mathcal{L}(\boldsymbol{x}) = \frac{\partial p(x_i;\boldsymbol{\phi})}{\partial \boldsymbol{\phi}} \mathcal{L}(\boldsymbol{x}) + \frac{\partial (1 - p(x_i;\boldsymbol{\phi}))}{\partial \boldsymbol{\phi}} \mathcal{L}(\boldsymbol{x}_{\downarrow i}) = \frac{\partial}{\partial \boldsymbol{\phi}} p(x_i;\boldsymbol{\phi})\big(\mathcal{L}(\boldsymbol{x}) - \mathcal{L}(\boldsymbol{x}_{\downarrow i})\big), \tag{5}$$

where $\boldsymbol{x}_{\downarrow i}$ denotes the full state vector $\boldsymbol{x}$ with the sign of x_i flipped. Since this expression is now invariant of x_i, we can multiply it with $1 = \sum_{x_i} p(x_i;\boldsymbol{\phi})$ and recollect the full gradient in the form:

$$\sum_x \big(\prod_{i'} p(x_{i'};\boldsymbol{\phi})\big) \sum_i \frac{\partial p(x_i;\boldsymbol{\phi})}{\partial \boldsymbol{\phi}} \big(\mathcal{L}(\boldsymbol{x}) - \mathcal{L}(\boldsymbol{x}_{\downarrow i})\big) = \mathbb{E}_{\boldsymbol{x} \sim p(\boldsymbol{x};\boldsymbol{\phi})} \sum_i \frac{\partial p(x_i;\boldsymbol{\phi})}{\partial \boldsymbol{\phi}} \big(\mathcal{L}(\boldsymbol{x}) - \mathcal{L}(\boldsymbol{x}_{\downarrow i})\big). \tag{6}$$

To obtain an unbiased estimate, it is sufficient to take one sample from the encoder $\boldsymbol{x} \sim p(\boldsymbol{x};\boldsymbol{\phi})$ and compute the sum in i. This estimator is known as *local expectations* (Titsias & Lázaro-Gredilla, 2015) and coincides in this case with GO-gradient (Cong et al., 2019) and RAM (Tokui & Sato, 2017).

However, evaluating $\mathcal{L}(\boldsymbol{x}_{\downarrow i})$ for all i may be impractical. A huge simplification is obtained if we assume that the change of the loss $\mathcal{L}$ when only a single latent bit x_i is changed can be approximated via linearization. When $\mathcal{L}$ is defined as a differentiable mapping $\mathbb{R}^n \to \mathbb{R}$ (as is the case in SBNs and autoencoders), which we evaluate on discrete inputs $\boldsymbol{x}$, we can approximate

$$\mathcal{L}(\boldsymbol{x}) - \mathcal{L}(\boldsymbol{x}_{\downarrow i}) \approx 2x_i \frac{\partial \mathcal{L}(\boldsymbol{x})}{\partial x_i}, \tag{7}$$

where we used that $x_i - (-x_i) = 2x_i$. Expressing $\frac{\partial}{\partial \boldsymbol{\phi}} p(x_i;\boldsymbol{\phi}) = x_i F'(a_i(\boldsymbol{\phi})) \frac{\partial}{\partial \boldsymbol{\phi}} a_i(\boldsymbol{\phi})$, we obtain

$$\frac{\partial p(x_i,\boldsymbol{\phi})}{\partial \boldsymbol{\phi}} (\mathcal{L}(\boldsymbol{x}) - \mathcal{L}(\boldsymbol{x}_{\downarrow i})) \approx 2F'(a_i(\boldsymbol{\phi}))\Big(\frac{\partial a_i(\boldsymbol{\phi})}{\partial \boldsymbol{\phi}}\Big) \frac{\partial \mathcal{L}(\boldsymbol{x})}{\partial x_i}. \tag{8}$$

If we now define that $\frac{\partial x_i}{\partial a_i} \equiv 2F'(a_i)$, the summation over i in (6) with the approximation (8) can be written in the form of a chain rule:

$$\sum_i 2F'(a_i(\boldsymbol{\phi}))\Big(\frac{\partial a_i(\boldsymbol{\phi})}{\partial \boldsymbol{\phi}}\Big) \frac{\partial \mathcal{L}(\boldsymbol{x})}{\partial x_i} = \sum_i \Big(\frac{\partial \mathcal{L}(\boldsymbol{x})}{\partial x_i}\Big)\Big(\frac{\partial x_i}{\partial a_i}\Big)\Big(\frac{\partial a_i(\boldsymbol{\phi})}{\partial \boldsymbol{\phi}}\Big). \tag{9}$$

To clarify, the estimator is already defined by the LHS of (9). We simply want to compute this expression by (ab)using the standard tools and this is the sole purpose of introducing $\frac{\partial x_i}{\partial a_i}$. Indeed the RHS of (9) is a product of matrices that can be efficiently computed by multiplying from right to left, *i.e.* by backpropagation. We thus obtained ST algorithm Alg. 1, *which matches exactly to the one described by Hinton (2012)*: to sample on the forward pass and use the derivative of the noise cdf on the backward pass. The coefficient 2 that occurred is due to that we used ± 1 encoding for $\boldsymbol{x}$.

2.2 Analysis

Below we present a summary of our analysis. Formal claims and their proofs are given in § B.

Invariances Observe that binary activations stay invariant under transformations: $\operatorname{sign}(a_i - z_i) = \operatorname{sign}(T(a_i) - T(z_i))$ for any strictly monotone mapping T. Consistently, *the ST gradient by Alg. 1 is also invariant to* T. In contrast, empirical straight-through approaches, in which the derivative proxy is hand-designed fail to maintain this property. In particular, rescaling the proxy leads to different estimators. Furthermore, when applying transform $T = F$ (the noise cdf), the backpropagation rule in line 5 of Alg. 1 becomes equivalent to using the identity proxy. Hence we see that a common description of ST in the literature "to back-propagate through the hard threshold function as if it had been the identity function" is also correct, *but only for the case of uniform noise*. Otherwise, and especially for deterministic ST, it is ambiguous as the result crucially depends on what transformations are applied under the hard threshold.

[1]The conditions refer to the Leibniz integral rule, other conditions may suffice *e.g.* for weak derivatives.

Algorithm 1: Straight-Through-Activations

/* $\boldsymbol{a}$ -- preactivation */
/* F -- activation noise cdf */
/* $\boldsymbol{x} \in \{-1, 1\}^n$ -- binary state */
1 **Forward(** $\boldsymbol{a}$ **)**
2 $\quad \boldsymbol{p} = F(\boldsymbol{a})$;
3 $\quad \boldsymbol{x} \sim 2\text{Bernoulli}(\boldsymbol{p}) - 1$;
4 **Backward(** $\frac{d\mathcal{L}}{d\boldsymbol{x}}$ **)**
$\quad$ /* $\frac{d\mathcal{L}}{d\boldsymbol{x}}$ -- loss grad in $\boldsymbol{x}$ */
5 $\quad \frac{d\mathcal{L}}{d\boldsymbol{a}} \equiv 2\,\text{diag}(F'(\boldsymbol{a}))\frac{d\mathcal{L}}{d\boldsymbol{x}}$;

Algorithm 2: Straight-Through-Weights

/* $\boldsymbol{\eta}$ -- mirror/latent weights */
/* F -- weight noise cdf */
/* $\boldsymbol{w} \in \{-1, 1\}^d$ */
1 **Forward(** η **)**
2 $\quad \boldsymbol{p} = F(\boldsymbol{\eta})$;
3 $\quad \boldsymbol{w} \sim 2\text{Bernoulli}(\boldsymbol{p}) - 1$;
4 **Backward(** $\frac{d\mathcal{L}}{d\boldsymbol{w}}$ **)**
$\quad$ /* $\frac{d\mathcal{L}}{d\boldsymbol{w}}$ -- loss grad in $\boldsymbol{w}$ */
5 $\quad \frac{d\mathcal{L}}{d\boldsymbol{\eta}} \equiv 2\frac{d\mathcal{L}}{d\boldsymbol{w}}$;

Bias Analysis I) When ST is unbiased? As we used linearization as the only biased approximation, it follows that *Alg. 1 is unbiased if the objective function $\mathcal{L}$ is multilinear in $\boldsymbol{x}$*. A simple counter-example, where ST is biased, is $\mathcal{L}(x) = x^2$. In this case the expected value of the loss is 1, independently of a that determines $x = \text{sign}(a - z)$ while the true gradient is zero. However the expected ST gradient is $\mathbb{E}[2F'(a)2x] = 4F'(a)(2F(a) - 1)$, which may be positive or negative depending on a. On the other hand, any function of binary variables has an equivalent multilinear expression. In particular, if we consider $\mathcal{L}(\boldsymbol{x}) = \|\boldsymbol{W}\boldsymbol{x} - \boldsymbol{y}\|^2$, analyzed by Yin et al. (2019), then $\tilde{\mathcal{L}}(\boldsymbol{x}) = \|\boldsymbol{W}\boldsymbol{x} - \boldsymbol{y}\|^2 - \sum_i x_i^2 \|\boldsymbol{W}_{:,i}\|^2 + \sum_i \|\boldsymbol{W}_{:,i}\|^2$ *coincides with $\mathcal{L}$ on all binary configurations and is multilinear*. It follows that ST applied to $\tilde{\mathcal{L}}$ gives an unbiased gradient estimate of $\mathbb{E}[\mathcal{L}]$. In the special case when $\mathcal{L}$ is linear in $\boldsymbol{x}$, the ST estimator is not only unbiased but has a zero variance, *i.e.*, it is exact. This surprising fact follows easily from the derivation.

II) How does using a different proxy than F in Line 5 of Alg. 1 affects the gradient in $\boldsymbol{\phi}$? Since $\text{diag}(F')$ occurs in the backward chain, we call estimators that uses a diagonal $\boldsymbol{\Lambda}$ instead of $\text{diag}(F')$ as *internally rescaled*. We show that *for any $\boldsymbol{\Lambda} \succcurlyeq 0$, the expected rescaled estimator has non-negative scalar product with the expected original estimator.* Thus using a different proxy, in particular identity, is not immediately destructive: if Alg. 1 was unbiased, the rescaled estimator may be biased but it is guaranteed to give an ascend direction in the expectation and the optimization can in principle succeed. However, assuming that Alg. 1 is biased (when $\mathcal{L}$ is not multi-linear) but still is an ascent direction, *the ascent property can no longer be guaranteed for the rescaled estimator*.

III) Does ST gradient give a valid ascent direction even when $\mathcal{L}$ is not multilinear? Assume that all partial derivatives $g_i(\boldsymbol{x}) = \frac{\partial \mathcal{L}(\boldsymbol{x})}{\partial x_i}$ are L-Lipschitz continuous for some L. We show that *expected ST gradient is an ascent direction if and only if $\left|\mathbb{E}_{\boldsymbol{x}}[g_i(\boldsymbol{x})]\right| > L$ for all i.*

IV) Can we decrease the bias? Assume that the loss function is applied after a linear transform of Bernoulli variables, *i.e.*, takes the form $\mathcal{L}(\boldsymbol{x}) = \ell(\boldsymbol{W}\boldsymbol{x})$. A typical initialization uses random $\boldsymbol{W}$ normalized by the size of the fan-in, *i.e.*, so that $\|\boldsymbol{W}_{k,:}\|_2 = 1\ \forall k$. In this case *the Lipschitz constant of gradients of $\mathcal{L}$ scales as $O(1/\sqrt{n})$, where n is the number of binary variables*. Therefore, *using more binary variables decreases the bias, at least at initialization*.

V) Does deterministic ST give an ascent direction? Let $\boldsymbol{g}^*$ be the deterministic ST gradient for the state $\boldsymbol{x}^* = \text{sign}(\boldsymbol{a})$ and $p^* = p(\boldsymbol{x}^*|\boldsymbol{a})$ probability of this state. *We show that deterministic ST gradient forms a positive scalar product with the expected ST gradient if $|g_i^*| \geq 2L(1-p^*)$ and with the true gradient if $|g_i^*| \geq 2L(1-p^*) + L$.* From this we conclude that deterministic ST positively correlates with the true gradient when $\mathcal{L}$ is multilinear, improves with the number of hidden units in the case described by IV and approaches expected stochastic ST Alg. 1 as units learn to be more deterministic so that the factor $(1 - p^*)$ decreases.

Variants Explained Using the invariance property, many works applying randomized ST estimators are easily seen to be equivalent to Alg. 1 (Raiko et al., 2015; Shen et al., 2018; Dadaneh et al., 2020). Furthermore, using different noise distributions for $\boldsymbol{z}$, we recover common choices of sign proxies used in empirical ST works as shown in Fig. A.1 (c-e). In our framework they correspond to *modeling choices* affecting the forward pass in the first turn.

Dai et al. (2017) perform a correct interchange of derivative and integral in (3) using *weak (distributional)* deriatives. After computing local expectations in this more complicated formalism, they

are back with finite differences (6) which they also propose to linearize as in (7). Thus their *distributional SGD is equivalent to common SGD with the ST estimator* Alg. 1.

3 LATENT WEIGHTS DO EXIST!

In this section we show that latent weights that did not get a proper substantiation in empirical ST methods (Hubara et al., 2016; Helwegen et al., 2019) can be clearly defined in SBNs and that several empirical update rules correspond to sound optimization schemes: projected gradient descent, mirror descent, variational Bayesian learning.

Let $\boldsymbol{w}$ be ± 1-Bernoulli weights with $p(w_i{=}1) = \theta_i$, let $\mathcal{L}(\boldsymbol{w})$ be the loss for a fixed training input. Consistently with the model for activations (1), we can define $w_i = \operatorname{sign}(\eta_i - z_i)$ in order to model weights w_i using parameters $\eta_i \in \mathbb{R}$ that we will call *latent weights*. We need to tackle two problems in order to optimize $\mathbb{E}_{\boldsymbol{w}\sim p(\boldsymbol{w}|\boldsymbol{\theta})}[\mathcal{L}(\boldsymbol{w})]$ in probabilities $\boldsymbol{\theta}$: i) how to estimate the gradient and ii) how to handle constraints $\boldsymbol{\theta} \in [0,1]^m$.

Projected Gradient A basic approach to handle constraints is the *projected gradient descent*:

$$\boldsymbol{\theta}^{t+1} := \operatorname{clip}(\boldsymbol{\theta}^t - \varepsilon \boldsymbol{g}^t, 0, 1), \quad \text{where} \quad \boldsymbol{g}^t = \nabla_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{w}\sim p(\boldsymbol{w}|\boldsymbol{\theta}^t)}[\mathcal{L}(\boldsymbol{w})] \tag{10}$$

and $\operatorname{clip}(\boldsymbol{x}, a, b) := \max(\min(\boldsymbol{x}, b), a)$ is the projection. Observe that for the uniform noise distribution on $[-1,1]$ with $F(z) = \operatorname{clip}(\frac{z+1}{2}, 0, 1)$, we have $\theta_i = p(w_i{=}1) = F(\eta_i) = \operatorname{clip}(\frac{\eta_i+1}{2}, 0, 1)$. Because this F is linear on $[-1,1]$, the update (10) can be equivalently reparametrized in $\boldsymbol{\eta}$ as

$$\boldsymbol{\eta}^{t+1} := \operatorname{clip}(\boldsymbol{\eta}^t - \varepsilon' \boldsymbol{h}^t, -1, 1), \quad \text{where} \quad \boldsymbol{h}^t = \nabla_{\boldsymbol{\eta}} \mathbb{E}_{\boldsymbol{w}\sim p(\boldsymbol{w}|F(\boldsymbol{\eta}))}[\mathcal{L}(\boldsymbol{w})] \quad \text{and} \quad \varepsilon' = 4\varepsilon. \tag{11}$$

The gradient in the latent weights, $\boldsymbol{h}^t$, can be estimated by Alg. 1 with the simplification $2F' = 1$. We obtained that *the method of Hubara et al. (2016, Alg.1) with stochastic rounding and with real-valued weights identified with $\boldsymbol{\eta}$ is equivalent to PGD on $\boldsymbol{\eta}$ with constraints $\eta \in [-1,1]^m$ and ST gradient estimate Alg. 1* and thus is a sound optimization scheme[2].

Mirror Descent As an alternative approach to handle constraints $\boldsymbol{\theta} \in [0,1]^m$, we study the application of mirror descent (MD) and connect it with the identity ST update variants. A step of MD is found by solving the following proximal problem:

$$\boldsymbol{\theta}^{t+1} = \min_{\boldsymbol{\theta}} \langle \boldsymbol{g}^t, \boldsymbol{\theta} - \boldsymbol{\theta}^t \rangle + \tfrac{1}{\varepsilon} D(\boldsymbol{\theta}, \boldsymbol{\theta}^t). \tag{12}$$

The divergence term $\frac{1}{\varepsilon} D(\boldsymbol{\theta}, \boldsymbol{\theta}^t)$ weights how much we trust the linear approximation $\langle \boldsymbol{g}^t, \boldsymbol{\theta} - \boldsymbol{\theta}^t \rangle$ when considering a step from $\boldsymbol{\theta}^t$ to $\boldsymbol{\theta}$. When the gradient is stochastic we speak about *stochastic mirror descent* (SMD) (Zhang & He, 2018; Azizan et al., 2019). A common choice of divergence to handle probability constraints is the KL-divergence $D(\theta_i, \theta_i^t) = \operatorname{KL}(\operatorname{Ber}(\theta_i), \operatorname{Ber}(\theta_i^t)) = \theta_i \log(\frac{\theta_i}{\theta_i^t}) + (1-\theta_i)\log(\frac{1-\theta_i}{1-\theta_i^t})$. Solving for stationary point of (12) gives

$$0 = \tfrac{\partial}{\partial \theta_i} f + \tfrac{1}{\varepsilon}\big(\log(\tfrac{\theta_i}{1-\theta_i}) - \log(\tfrac{\theta_i^t}{1-\boldsymbol{\theta}_i^t})\big). \tag{13}$$

Observe that when $F = \sigma$ we have $\log(\frac{\theta_i}{1-\theta_i}) = \eta_i$. Then the MD step can be written in the well-known convenient form using the latent weights $\boldsymbol{\eta}$ (natural parameters of Bernoulli distribution):

$$\boldsymbol{\theta}^t := \sigma(\boldsymbol{\eta}^t); \qquad \boldsymbol{\eta}^{t+1} := \boldsymbol{\eta}^t - \varepsilon \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}^t). \tag{14}$$

We thus have obtained the rule where on the forward pass $\boldsymbol{\theta} = \sigma(\boldsymbol{\eta})$ defines the sampling probability of $\boldsymbol{w}$ and on the backward pass the derivative σ that otherwise occurs in Line 5 of Alg. 1 *is bypassed exactly as if the identity proxy was used*. We define such ST rule for optimization in weights as Alg. 2. Its correctness is not limited to logistic noise. We show that for any strictly monotone noise distribution F there is a corresponding divergence function D.

Proposition 1. Common SGD in latent weights $\boldsymbol{\eta}$ using the *identity straight-through-weights* Alg. 2 implements SMD in the weight probabilities $\boldsymbol{\theta}$ with the divergence corresponding to F.

[2] In the projected SGD setting there are convergence guarantees at least in the convex setting.

The proof in § C is closely related to the work by Ajanthan et al. (2019). Differently from us, they considered a connection to deterministic ST. Their argument includes taking the limit in which F is squashed into the step function, which renders MD invalid. This problem does not occur in our formulation. Proposition 1 reveals that although Bernoulli weights can be modeled the same way as activations using the injected noise model $\boldsymbol{w} = \operatorname{sign}(\boldsymbol{\eta} - \boldsymbol{z})$, *the noise distribution F for weights does not reflect a modeling choice but the optimization proximity scheme*.

Despite the generality of Proposition 1, we view the KL divergence as a more reliable choice in practice. Azizan et al. (2019) have shown that the optimization with SMD has an inductive bias to find the closest solution to the initialization point as measured by the divergence used in MD, which has a strong impact on generalization. This suggests that MD with KL divergence will prefer higher entropy solutions, making more diverse predictions. By our equivalence results, using SGD on latent weights with logistic noise and identity straight-through Alg. 2 enjoys the same properties.

Variational Bayesian Learning Extending the results above, in § C.2 we study the variational Bayesian learning formulation and show the following.

Proposition 2. Common SGD in latent weights $\boldsymbol{\eta}$ *with a weight decay and identity straight-through-weights* Alg. 2 is equivalent to optimizing a factorized variational approximation to the weight posterior $p(\boldsymbol{w}|\text{data})$ using a composite SMD method.

We can see that powerful and sound learning techniques can be obtained in a form of simple update rules. Such simple learning rules are recently (contemporaneously) studied by Khan & Rue (2020). We emphasize the interplay with the identity-ST estimator and the connection to the implicit regularization discussed above. The recent work by Meng et al. (2020) also addresses the binary case but uses Gumbel-Softmax estimator. We discuss its shortcomings and an unexpected equivalence to deterministic ST in § F.

4 Deep Straight-Through

In this section we review the derivation of ST for deep SBNs (Shekhovtsov et al., 2020), extend it to binary weights using MD from § 3 and propose an initialization method based on the analysis in § 2.

Derivation Idea Consider an SBN model with L binary layers, recurrently defined as

$$\boldsymbol{x}^k = \operatorname{sign}(\boldsymbol{a}^k(\boldsymbol{x}^{k-1}; \boldsymbol{\theta}) - \boldsymbol{z}^k), \quad k = 1 \dots L, \tag{15}$$

where $\boldsymbol{a}^k$ are *pre-activations* and $\boldsymbol{z}^k$ are injected noises independent for all units. Let also $\mathcal{L}(\boldsymbol{x}^L; \boldsymbol{\theta})$ be the loss function for a given input and the last layer state $\boldsymbol{x}^L$. Similarly to one-layer case, the probability $p(\boldsymbol{x}^k|\boldsymbol{x}^{k-1})$ is conditionally independent with $p(x_i^k = 1|\boldsymbol{x}^{k-1}) = F(a_i^k(\boldsymbol{x}^{k-1}))$.

The challenge to derive an ST-like estimator for deep SBN is that simply applying the steps in § 2 to the k's layer in order to estimate gradient w.r.t. $\boldsymbol{a}^k$ is not straightforward because we would have to linearize the expectation of the whole network remainder and the loss as a function of $\boldsymbol{x}^k$. The idea of Shekhovtsov et al. (2020) is to linearize only the next layer's conditional probability $p(\boldsymbol{x}^{k+1}|\boldsymbol{x}^k)$ and then proceed recurrently, chaining derandomization as in § 2 and linearization steps. By doing so they infer a more accurate PSA estimator and from it the deep ST estimator. The ST estimator (Shekhovtsov et al., 2020, Alg. 2) *is identical to applying ST Alg. 1 to each binary layer and executing the automatic differentiation*. The insights from the derivation are twofold. First, since derandomization is performed recurrently, the variance for deep layers is significantly reduced. Second, we know which approximations contribute to the bias, they are indeed the linearizations

$$p(\boldsymbol{x}^{k+1}|\boldsymbol{x}^k) - p(\boldsymbol{x}^{k+1}|\boldsymbol{x}^k_{\downarrow i}) \approx 2x_i^k \frac{\partial p(\boldsymbol{x}^{k+1}|\boldsymbol{x}^k)}{\partial x_i^k} \qquad \text{and} \qquad \mathcal{L}(\boldsymbol{x}^L) - \mathcal{L}(\boldsymbol{x}^L_{\downarrow i}) \approx \frac{\partial \mathcal{L}(\boldsymbol{x}^L)}{\partial x_i^L}, \tag{16}$$

for all layers k and units i. The experiments (Shekhovtsov et al., 2020, Figs. 2, 3, 4) with real weights suggest that while for small networks these approximations lead to some degradation of gradient estimation accuracy during learning, for large networks ST performs on par with the PSA method that uses a much tighter approximation. This is well aligned with our analysis in § 2.2.

Extensions We now can put all building blocks together. Our deep model extends the deep model (15) as follows. The weights $\boldsymbol{w}^k$ in each layer are ± 1 Bernoulli with probability $\boldsymbol{\theta}$.

Pre-activations $\boldsymbol{a}$ are extended to incorporate additional processing, importantly batch normalization (Ioffe & Szegedy, 2015):

$$\boldsymbol{a}^k = \text{BN}(\text{Linear}(\boldsymbol{x}^{k-1}, \boldsymbol{w}^k)), \tag{17}$$

where Linear is a binary fully connected or convolutional transform and BN has real-valued affine terms (scale, bias) enabled. The model is illustrated in Fig. D.2. Note that in the linearization of $p(\boldsymbol{x}^{k+1}|\boldsymbol{x}^k)$ in $\boldsymbol{x}^k$, we effectively differentiate the composite transform (Linear, BN, F, and finally the product forming $p(\boldsymbol{x}^{k+1}|\boldsymbol{x}^k)$) and approximate the finite difference $p(\boldsymbol{x}^{k+1}|\boldsymbol{x}^k) - p(\boldsymbol{x}^{k+1}|\boldsymbol{x}^k_{\downarrow i})$ using this linearization. Thus, extra non-linearities such as BN in principle worsen the approximation. On the other hand, the strongest non-linearity is introduced here by F. Following the arguments of § 2.2, a large number of hidden units in combination with normalization by BN should achieve a better approximation accuracy for linearizing F.

Initialization The role of the affine parameters $(\boldsymbol{s}, \boldsymbol{b})$ in BN is to reintroduce the scale and bias degrees of freedom removed by the normalization (Ioffe & Szegedy, 2015). In our model these degrees of freedom are important as they control the strength of pre-activation relative to noise. With the sign activation, they could be indeed equivalently represented as learnable bias and variance parameters of the noise since $\text{sign}(x_i s_i + b_i - z_i) = \text{sign}\left(x_i - \frac{z_i - b_i}{s_i}\right)$ assuming $s_i > 0$. Without the BN layer, the result of $\text{Linear}(\boldsymbol{x}^{k-1}, \boldsymbol{w}^k)$ is an integer in a range that depends on the size of $\boldsymbol{x}$. If the noise variance is set to 1, this will lead to vanishing gradients in a large network. With BN and its affine transform the right proportion can be learned, but it is important to initialize it so that the learning can make progress. We propose the following initialization. We set $s_i = 1$ and $b_i = 0$ (as default for BN) and *normalize the noise distribution* so that it has zero mean and $F'(0) = \frac{1}{2}$. This choice ensures that the Jacobian $2F'(\boldsymbol{a})$ in Line 5 of Alg. 1 at the mean value of pre-activations is the identity matrix and therefore gradients do not vanish.

We initialize weight probabilities θ_i as uniform in $[0, 1]$. The corresponding initialization of latent weights is then $\eta_i = F^{-1}(\theta_i)$. This initialization and the noise scaling above are the places where the empirical approaches have to do extra guessing when considering different ST proxies, obtain diverse results and attribute them to the quality of proxies (*e.g.* Esser et al., 2016).

5 Experiments

Stochastic Autoencoders Previous work has demonstrated that Gumbel-Softmax (biased) and ARM (unbiased) estimators give better results than ST on training variational autoencoders with Bernoulli latents (Jang et al., 2016; Yin & Zhou, 2019; Dadaneh et al., 2020). However, only the test performance was revealed to readers. We investigate in more detail what happens during the training. Except of studying the training loss under equal training setup, we can directly measure the gradient estimation accuracy of different variants of ST estimators as well as the reference methods in comparison to the true gradient. The latter can be estimated sufficiently accurately in models with a single Bernoulli layer using ARM with many samples (we use 1000).

We train a simple yet realistic variant of stochastic autoencoder for the task of text retrieval with binary representation on *20newsgroups* dataset. The autoencoder is trained by minimizing the reconstruction loss (2). The observed features $\boldsymbol{y} \in \mathbb{R}^d$ are word counts with $d = 10000$ words. The encoder maps them to Bernoulli probabilities of codes $\boldsymbol{x} \in \{0, 1\}^n$ using a neural network with structure $d - 512 - n$, *i.e.* with 512 hidden (deterministic) states and ReLU activations. The decoder network is symmetric: $n - 512 - d$. Please refer to § D.1 for full specification and auxiliary results.

For each estimator we perform the following protocol. First, we train the model with this estimator using Adam with $lr = 0.001$ for 1000 epochs. We then switch the estimator to ARM with 10 samples and continue training for 500 more epochs (denoted as ARM-10 correction phase). Fig. 1 top shows the training performance for different number of latent bits n. It is seen (esp. for 8 and 64 bits) that some estimators (esp. ST and det_ST) appear to make no visible progress, and even increase the loss, while switching them to ARM makes a rapid improvement. Does it mean that these estimators are bad and ARM is very good? An explanation of this phenomenon is offered in Fig. 2. The rapid improvement by ARM is possible because these estimators have accumulated a significant bias due to a systematic error component (*c.f.* example in § 2.2), possibly only is a subspace of parameters, which nevertheless can be easily corrected by an unbiased estimator.

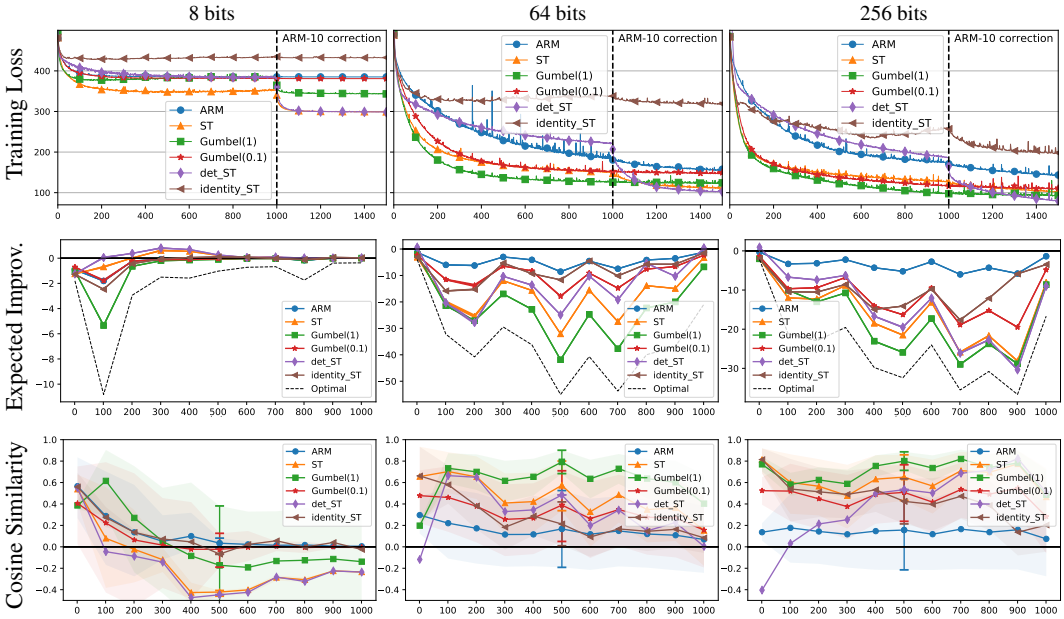

Figure 1: Comparison of the training performance and gradient estimation accuracy for a stochastic autoencoder with different number of latent Bernoulli units (bits). *Training Loss:* each estimator is applied for 100 epochs and then switched to ARM-10 in order to correct the accumulated bias. *Expected improvement*: lower is better (measures expected change of the loss), the dashed line shows the maximal possible improvement knowing the true gradient. *Cosine similarity*: higher is better, close to 1 means that the direction is accurate while below 0 means the estimated gradient is not an ascent direction; error bars indicate empirical 70% confidence intervals using 100 trials.

To measure the bias and alignment of directions as theoretically analyzed in § 2.2, we evaluate different estimators at the same parameter points, located along the learning trajectory of the reference ARM estimator. At each such point we compute the reference gradient by ARM-1000 and evaluate the quality of the 1-sample candidate estimator. To measure the quality we compute the *expected cosine similarity* and the *expected improvement*, defined respectively by:

$$\mathrm{ECS} = \mathbb{E}\Big[\langle \boldsymbol{g}, \tilde{\boldsymbol{g}} \rangle / (\|\boldsymbol{g}\| \|\tilde{\boldsymbol{g}}\|)\Big], \qquad \mathrm{EI} = -\mathbb{E}[\langle \boldsymbol{g}, \tilde{\boldsymbol{g}} \rangle] / \sqrt{\mathbb{E}[\|\tilde{\boldsymbol{g}}\|^2]}, \tag{18}$$

where $\boldsymbol{g}$ is the true gradient and $\tilde{\boldsymbol{g}}$ is an estimate. The expectation are taken over 100 trials and all batches. The detailed explanation of these metrics is deferred to § D.1. Briefly, EI metric models the objective loss locally using the proximal problem of an optimization method that can adapt the learning rate per each epoch. These measurements, displayed in Fig. 1 for different bit length, clearly show that with a small bit length biased estimators consistently start to point in the wrong direction. Identity ST and deterministic ST clearly introduce an extra bias to ST. However, when we increase the number of latent bits, the accuracy of all biased estimators improves, confirming our analysis **IV)**, **V)**. In the bottom right plot we see a clear confirmation of **V)** that deterministic ST improves as the network becomes more deterministic during the learning when there are sufficiently many units.

The practical takeaways are as follows: 1) biased estimators may perform significantly better than unbiased but might require a correction of the systematically accumulated bias. 2) with more units the ST approximation clearly improves and the bias has a less detrimental effect, requiring less correction 3) derived stochastic ST is indeed more accurate than other ST variants, however deterministic ST may have some additional useful properties for optimization not yet investigated.

Classification with Deep SBN In training deep SBNs, ARM becomes more computationally costly and the relaxation error of Gumbel-Softmax becomes too high. In this setup ST becomes more favorable and is the method of choice of many researchers. We want to show that the explainable ST with components that we derived: identity ST for weights, link to the noise model, connection to BN and the initialization scheme can do the job of the previous carefully selected but completely unclear approaches. We consider CIFAR-10 dataset and use the VGG-like architecture (Courbariaux

Table 1: Test accuracy for different methods on CIFAR-10 with the same/similar architecture. SBN can be tested either with zero noises (*det*) or using an ensemble of several samples (we use *10-sample*). Standard deviations are given w.r.t. to 4 learning trials with random initialization. The two quotations for Hubara et al. (2016) refer to their result with Torch7 implementation that uses randomized Htanh and Theano implementation that uses deterministic Htanh, respectively.

STOCHASTIC TRAINING		
Method	det	10-sample
Our SBN, logistic noise	89.6 ± 0.1	90.6 ± 0.2
Our SBN, uniform noise	89.7 ± 0.2	90.5 ± 0.2
Our SBN, triangular noise	89.5 ± 0.2	90.0 ± 0.3
Hubara et al. (2016) (rand.)	89.85	-
Peters & Welling (2018)	88.61	16-sample: 91.2
DETERMINISTIC TRAINING		
Rastegari et al. (2016)	89.83	-
Hubara et al. (2016) (det.)	88.60	-

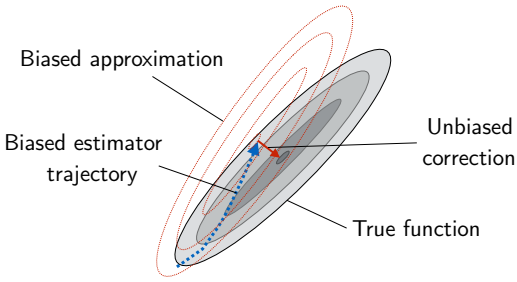

Figure 2: Schematic explanation of the optimization process using a biased estimator followed by a correction with an unbiased estimator. The figure applies more directly to estimators that do have a relaxed objective, *e.g.* Gumbel-Softmax. Initially, the biased estimator makes good progress, but then the value of the true loss function may start growing while the optimization steps nevertheless come closer to the optimal location in the parameter space.

et al., 2015; Hubara et al., 2016) specified in Fig. D.2. Our full learning setup is specified in § D.2. We trained SBNs with three choices of noise distributions corresponding to proxies used by prior work as in Fig. A.1 (c-e). Table 1 shows the test results in comparison with baselines.

We see that training with different choices of noise distributions, corresponding to different ST rules, all achieves equally good results. This is in contrast to empirical studies advocating specific proxies and is allowed by the consistency of the model, initialization and training. The identity ST at the correct place (ST-through-weights) works well and we know why. Comparing to empirical ST baselines (all except Peters & Welling), we see that there is no significant difference ('det' column) indicating that our derived ST method is on par with the well-guessed baselines. If the same networks we trained are tested in the stochastic mode (10-sample), there is a clear boost of performance, indicating an advantage of SBN models. Stochastic ST works better in the two experiments of Hubara et al. but deterministic ST also works well in a somewhat different learning setup of Rastegari et al.

There is a small gap to Peters & Welling in the stochastic mode (our 10-sample versus their 16-sample). Nevertheless, the results are very similar considering that they use a different estimation method, an initialization from a pretrained model and an early stopping (we believe otherwise they would overfit to the relaxation of the SBN considered). The takeaway message here is that ST can be considered in the context of SBN models as a simple but proper baseline. Since we achieve near 100% training accuracy, the optimization fully succeeds and thus the bias of ST is tolerable.

6 Conclusion

We have put many ST methods on a solid basis by deriving and explaining them from the first principles in the framework of SBNs. It is well-defined what they estimate and what the bias means. We obtained two different main estimators for propagating activations and weights, bringing the understanding which function they have, what approximations they involve and what are the limitations imposed by these approximations. The resulting methods in all cases are strikingly simple, no wonder they have been first discovered empirically long ago. We showed how our theory leads to a useful understanding of bias properties and to reasonable choices that allow for a more reliable application of these methods. We hope that researchers will continue to use these simple techniques, now with less guesswork and obscurity, as well as develop improvements to them. Our code will be available on `http://github.com`.

REFERENCES

Thalaiyasingam Ajanthan, Kartik Gupta, Philip HS Torr, Richard Hartley, and Puneet K Dokania. Mirror descent view for neural network quantization. *arXiv preprint arXiv:1910.08237*, 2019.

Milad Alizadeh, Javier Fernandez-Marques, Nicholas D. Lane, and Yarin Gal. An empirical study of binary neural networks' optimisation. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJfUCoR5KX.

Navid Azizan, Sahin Lale, and Babak Hassibi. Stochastic mirror descent on overparameterized nonlinear models: Convergence, implicit regularization, and generalization. *arXiv preprint arXiv:1906.03830*, 2019.

Yu Bai, Yu-Xiang Wang, and Edo Liberty. Proxquant: Quantized neural networks via proximal operators. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HyzMyhCcK7.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Joseph Bethge, Haojin Yang, Marvin Bornstein, and Christoph Meinel. Back to simplicity: How to train accurate BNNs from scratch? *CoRR*, abs/1906.08637, 2019. URL http://arxiv.org/abs/1906.08637.

E. Boros and P.L. Hammer. Pseudo-Boolean optimization. *Discrete Applied Mathematics*, 1-3(123):155–225, 2002.

Adrian Bulat and Georgios Tzimiropoulos. Binarized convolutional landmark localizers for human pose estimation and face alignment with limited resources. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

Suthee Chaidaroon and Yi Fang. Variational deep semantic hashing for text documents. In *SIGIR Conference on Research and Development in Information Retrieval*, pp. 75–84, 2017.

Pengyu Cheng, Chang Liu, Chunyuan Li, Dinghan Shen, Ricardo Henao, and Lawrence Carin. Straight-through estimator as projected wasserstein gradient flow. *arXiv preprint arXiv:1910.02176*, 2019.

Yulai Cong, Miaoyun Zhao, Ke Bai, and Lawrence Carin. GO gradient for expectation-based objectives. In *International Conference on Learning Representations*, 2019.

Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in neural information processing systems*, pp. 3123–3131, 2015.

Siamak Zamani Dadaneh, Shahin Boluki, Mingzhang Yin, Mingyuan Zhou, and Xiaoning Qian. Pairwise supervised hashing with Bernoulli variational auto-encoder and self-control gradient estimator. *ArXiv*, abs/2005.10477, 2020.

Bo Dai, Ruiqi Guo, Sanjiv Kumar, Niao He, and Le Song. Stochastic generative hashing. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 913–922, 2017.

Steven K. Esser, Paul A. Merolla, John V. Arthur, Andrew S. Cassidy, Rathinakumar Appuswamy, Alexander Andreopoulos, David J. Berg, Jeffrey L. McKinstry, Timothy Melano, Davis R. Barch, Carmelo di Nolfo, Pallab Datta, Arnon Amir, Brian Taba, Myron D. Flickner, and Dharmendra S. Modha. Convolutional networks for fast, energy-efficient neuromorphic computing. *Proceedings of the National Academy of Sciences*, 113(41):11441–11446, 2016. URL https://www.pnas.org/content/113/41/11441.

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

Alex Graves. Practical variational inference for neural networks. In *NeurIPS*, pp. 2348–2356. 2011.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *ICCV*, pp. 1026–1034, 2015.

Koen Helwegen, James Widdicombe, Lukas Geiger, Zechun Liu, Kwang-Ting Cheng, and Roeland Nusselder. Latent weights do not exist: Rethinking binarized neural network optimization. In *Advances in neural information processing systems*, pp. 7531–7542, 2019.

Geoffrey Hinton. Lecture 15d - Semantic hashing : 3:05 - 3:35, 2012. URL https://www.cs.toronto.edu/~hinton/coursera/lecture15/lec15d.mp4.

M. Horowitz. Computing's energy problem (and what we can do about it). In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pp. 10–14, 2014.

Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In *Advances in neural information processing systems*, pp. 4107–4115, 2016.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, volume 37, pp. 448–456, 2015.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Emtiyaz Khan and Haavard Rue. Learning algorithms from Bayesian principles. August 2020. Draft v. 0.7.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

Alex Krizhevsky and Geoffrey E. Hinton. Using very deep autoencoders for content-based image retrieval. In *ESANN*, 2011.

Wu Lin, Mohammad Emtiyaz Khan, and Mark Schmidt. Fast and simple natural-gradient variational inference with mixture of exponential-family approximations. In *ICML*, volume 97, Jun 2019.

Xiaofan Lin, Cong Zhao, and Wei Pan. Towards accurate binary convolutional neural network. In *Advances in Neural Information Processing Systems 30*, pp. 345–353. 2017.

Zechun Liu, Baoyuan Wu, Wenhan Luo, Xin Yang, Wei Liu, and Kwang-Ting Cheng. Bi-real net: Enhancing the performance of 1-bit CNNs with improved representational capability and advanced training algorithm. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 722–737, 2018.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. 2016. arxiv:1611.00712.

Xiangming Meng, Roman Bachmann, and Mohammad Emtiyaz Khan. Training binary neural networks using the Bayesian learning rule, 2020.

Ricardo Nanculef, Francisco Mena, Antonio Macaluso, Stefano Lodi, and Claudio Sartori. Self-supervised Bernoulli autoencoders for semi-supervised hashing, 2020.

Arkadiĭ Semenovich Nemirovsky and David Borisovich Yudin. Problem complexity and method efficiency in optimization. 1983.

Art B. Owen. *Monte Carlo theory, methods and examples*. 2013.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. 2019.

Jorn WT Peters and Max Welling. Probabilistic binary neural networks. *arXiv preprint arXiv:1809.03368*, 2018.

Tapani Raiko, Mathias Berglund, Guillaume Alain, and Laurent Dinh. Techniques for learning binary stochastic feedforward neural networks. In *ICLR*, 2015. URL `http://arxiv.org/abs/1406.2989`.

Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. XNOR-Net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pp. 525–542. Springer, 2016.

Wolfgang Roth, Günther Schindler, Holger Fröning, and Franz Pernkopf. Training discrete-valued neural networks with sign activations using weight distributions. In *European Conference on Machine Learning (ECML)*, 2019.

Oran Shayer, Dan Levi, and Ethan Fetaya. Learning discrete weights using the local reparameterization trick. *arXiv preprint arXiv:1710.07739*, 2017.

Alexander Shekhovtsov, Viktor Yanush, and Boris Flach. Path sample-analytic gradient estimators for stochastic binary networks, 2020.

Dinghan Shen, Qinliang Su, Paidamoyo Chapfuwa, Wenlin Wang, Guoyin Wang, Ricardo Henao, and Lawrence Carin. NASH: toward end-to-end neural architecture for generative semantic hashing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pp. 2041–2050, 2018.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 15:1929–1958, 2014.

Wei Tang, Gang Hua, and Liang Wang. How to train a compact binary neural network with high accuracy? In *AAAI*, 2017.

Michalis K. Titsias and Miguel Lázaro-Gredilla. Local expectation gradients for black box variational inference. In *International Conference on Neural Information Processing Systems*, pp. 2638–2646, 2015.

Seiya Tokui and Issei Sato. Evaluating the variance of likelihood-ratio gradient estimators. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, pp. 3414–3423, 2017.

George Tucker, Andriy Mnih, Chris J Maddison, John Lawson, and Jascha Sohl-Dickstein. REBAR: Low-variance, unbiased gradient estimates for discrete latent variable models. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *NeurIPS*, pp. 2627–2636. 2017.

Xu Xiang, Yanmin Qian, and Kai Yu. Binary deep neural networks for speech recognition. In *INTERSPEECH*, 2017.

Mingzhang Yin and Mingyuan Zhou. ARM: Augment-REINFORCE-merge gradient for stochastic binary networks. In *ICLR*, 2019.

Penghang Yin, Jiancheng Lyu, Shuai Zhang, Stanley Osher, Yingyong Qi, and Jack Xin. Understanding straight-through estimator in training activation quantized neural nets. *arXiv preprint arXiv:1903.05662*, 2019.

Siqi Zhang and Niao He. On the convergence rate of stochastic mirror descent for nonsmooth nonconvex optimization. *arXiv: Optimization and Control*, 2018.

Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016.

Reintroducing Straight-Through Estimators as Principled Methods for Stochastic Binary Networks (Appendix)

Contents

A Related Work

B Analysis of ST with 1 Hidden Layer
- B.1 Invariances
- B.2 Bias Analysis

C Mirror Descent and Variational Mirror Descent
- C.1 Mirror Descent
- C.2 Latent Weight Decay Implements Variational Bayesian Learning

D Details of Experiments
- D.1 Stochastic Autoencoder
- D.2 Classification with Deep Stochastic Binary Networks

E Gumbel Softmax and ST Gumbel-Softmax

F Analysis of BayesBiNN

A Related Work

Hinton's vs Bengio's ST The name *straight-through* and the first experimental comparison was proposed by Bengio et al. (2013). Referring to Hinton's lecture, they describe the idea as "*simply to back-propagate through the hard threshold function (1 if the argument is positive, 0 otherwise) as if it had been the identity function*". In the aforementioned lecture (Hinton, 2012), however we find a somewhat different description: "*during the forward pass we stochastically pick a binary value using the output of the logistic, and then during the backward pass we pretend that we've transmitted the real valued probability from the logistic*". We can make two observations: 1) different variants appeared early on and 2) many subsequent works (*e.g.* Yin et al., 2019) attribute these two variants in the exact opposite way, adding to the confusion.

ST Analysis Yin et al. (2019) analyzes deterministic ST variants. The theoretical analysis is applicable to 1 hidden layer model with quadratic loss and the input data following a Gaussian distribution. The input distribution assumption is arguably artificial, however it allows to analyze the expected loss and its gradient. They show that population ST gradients using ReLU and clipped ReLU proxy correlate positively with the true population gradient and allow for convergence while identity ST does not. In § B we show that in the SBN model, a simple correction of the quadratic loss function makes the base ST estimator unbiased and all rescaled estimators including identity are ascent directions in the expectation. Also note that the approach to analyze deterministic ST methods by considering the expectation over the input has a principle limitation for extending to deep models: the expectation over the input of a deterministic network with two hidden binary layers is still non-smooth (non-differentiable) in the parameters of the second layer.

Cheng et al. (2019) shows for networks with 1 hidden layer that STE is approximately related to the projected Wasserstein gradient flow method proposed there.

On the weights side of the problem, Ajanthan et al. (2019) connected mirror descent updates for constrained optimization (*e.g.*, $\boldsymbol{w} \in [0,1]^m$) with straight-through methods. The connection of deterministic straight-through for weights and proximal updates was also observed in Bai et al. (2019). Mirror Descent has been applied to variational Bayesian learning of continuous weights *e.g.* in Lin et al. (2019), taking the form of update in natural parameters with the gradient in the mean parameters, same as in our case.

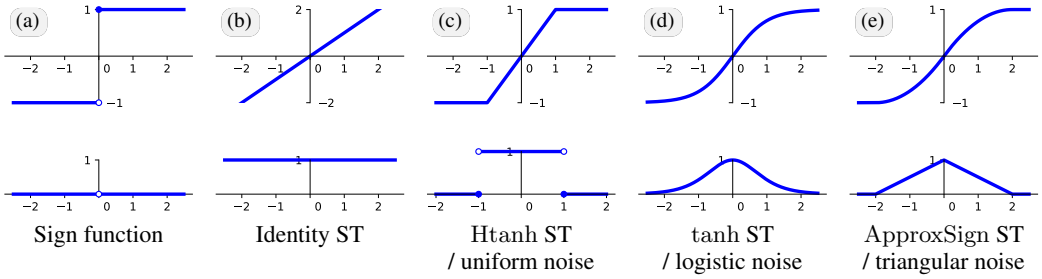

Figure A.1: The sign function and different proxy functions for derivatives used in empirical ST estimators. Variants (c-e) can be obtained by choosing the noise distribution in our framework. Specifically for a real-valued noise z with cdf F, in the upper plots we show $\mathbb{E}_z[\mathrm{sign}(a-z)] = 2F-1$ and, respectively, twice the density, $2F'$ in the lower plots. Choosing *uniform* distribution for z gives the density $p(z) = \frac{1}{2}\mathbb{1}_{z\in[-1,1]}$ and recovers the common Htanh proxy in (c). The *logistic* noise has cdf $F(z) = \sigma(2z)$, which recovers tanh proxy in (d). The *triangular* noise has density $p(z) = \max(0, |(2-x)/4|)$, which recovers a scaled version of ApproxSign (Liu et al., 2018) in (e). The scaling (standard deviation) of the noise in each case is chosen so that $2F'(0) = 1$ as explained in § 4. The identity ST form in (b) we recover as latent weight updates with mirror descent.

Alternative Estimators For deep binary networks several gradient estimation approaches are based on stochastic gradients of analytically smoothed/approximated loss (Peters & Welling, 2018; Roth et al., 2019). There is however a discrepancy between analytic approximation and the binary samples used at the test time. Shekhovtsov et al. (2020, Fig. 4) show that such relaxed objectives may indeed significantly diverge during the training. To obtain good results, a strong dropout regularization and/or pretraining is needed (Peters & Welling, 2018; Roth et al., 2019). Despite these difficulties they demonstrate on par or improved results, especially when using average prediction over multiple noise samples at test time.

B Analysis of ST with 1 Hidden Layer

B.1 Invariances

We have the following simple yet desirable and useful property. It is easy to observe that binary activations admit equivalent reformulations as

$$\mathrm{sign}(a_i - z_i) = \mathrm{sign}(T(a_i) - T(z_i)) \tag{19}$$

for any strictly monotone mapping $T\colon \mathbb{R} \to \mathbb{R}$.

Proposition B.1. The gradient computed by Alg. 1 is invariant to equivalent transformations under sign as in (19).

Proof. Let us denote the transformed noise as $\tilde{z}_i = T(z_i)$, its cdf as G and the transformed activations as $\tilde{a}_i = T(a_i)$. The sampling probability in line 2 of Alg. 1 does not change since after the transformation it computes $p = G(\tilde{a}_i) = \mathbb{P}(\tilde{z}_i \le \tilde{a}_i \,|\, \tilde{a}_i) = \mathbb{P}(z_i \le a_i \,|\, a_i) = F(a_i)$. The gradient returned by line 5 does not change since we have $\frac{d}{da_i}G(T(a_i)) = F'(a_i)$. □

In contrast, empirical straight-through approaches where the proxy is hand-designed fail to maintain this property. In particular, in the deterministic straight-through approach transforms such as $\mathrm{sign}(a_i) = \mathrm{sign}(T(a_i))$ while keeping the proxy of sign used in backprop fixed lead to different gradient estimates. This partially explains why many proxies have been tried, *e.g.* ApproxSign (Liu et al., 2018), and their scale needed tuning. Another pathological special case that leads to a confusion between identity straight-through and other forms is as follows.

Corollary B.1. Let F be strictly monotone. Then letting $T = F$ leads to $T(z_i)$ being uniformly distributed. Let $\tilde{a}_i = T(a_i)$. In this case the backpropagation rule in line 5 of Alg. 1 can be interpreted as replacing the gradient of $\mathrm{sign}(\tilde{a}_i - T(z_i))$ in $\tilde{a}_i$ with just identity.

Indeed, since $\tilde{z}_i = T(z_i)$ is uniform, we have $G' = 1$ on $(0, 1)$ and $\tilde{a}_i = F(a_i)$ is guaranteed to be in $(0, 1)$ by strict monotonicity. The gradient back-propagated by usual rules through $\tilde{a}_i$ (outside

of the ST Alg. 1) encounters derivative of F as before. Hence we see that the description "to back-propagate through the hard threshold function as if it had been the identity function" could be misleading as the resulting estimator crucially depends on what transformations are applied under the hard threshold despite they do not affect the network predictions in any way. We refer to the variant by Bengio et al. (2013) as identity-ST, as it specifically uses the identity proxy for the gradient in the pre-sigmoid activation.

B.2 Bias Analysis

I) Since the only approximation that we made was linearization of the objective $\mathcal{L}$, we have the following basic property.

Proposition B.2. If the objective function $\mathcal{L}$ is multilinear[3] in the binary variables $\boldsymbol{x}$, then Alg. 1 is unbiased.

Proof. In this case (7) holds as equality. □

While extremely simple, this is an important point for understanding the ST estimator. As an immediate consequence we can easily design counter-examples where ST is wrong.

Example 1. Let $a \in \mathbb{R}$, $x = \text{sign}(a - z)$ and $\mathcal{L}(x) = x^2$. In this case the expected value of the loss is 1, independent of a. The true gradient is zero. However the expected ST gradient is $\mathbb{E}[2F'(a)2x] = 4F'(a)(2\mathcal{L}(a) - 1)$ and can be positive or negative depending on a.

Example 2 (Tokui & Sato 2017)**.** Let $\mathcal{L}(x) = x - \sin(2\pi x)$. Then the finite difference $\mathcal{L}(1) - \mathcal{L}(0) = 1$ but the derivative $\frac{\partial \mathcal{L}}{\partial x} = 1 - 2\pi\cos(2\pi x) = -1$. In this failure example, ST, even in the expectation, will point to exactly the opposite direction of the true gradient.

An important observation from the above examples is that the result of ST is not invariant with respect to reformulations of the loss that preserve its values in all binary points. In particular, we have that $\mathcal{L} \equiv 1$ in the first example and $\mathcal{L}(x) \equiv x$ in the second example for any $x \in \{-1, 1\}$. If we used these equivalent representations instead, the ST estimator would have been correct.

More generally, any real-valued function of binary variables has a unique polynomial (and hence multilinear) representation (Boros & Hammer, 2002) and therefore it is possible to find a loss reformulation such that the ST estimator will be unbiased. Unfortunately, this representation is intractable in most cases, but it is tractable, *e.g.*, for a quadratic loss, useful in regression and autoencoders with a Gaussian observation model.

Proposition B.3. Let $\mathcal{L}(\boldsymbol{x}) = \|\boldsymbol{W}\boldsymbol{x} - \boldsymbol{y}\|^2$. Then the multilinear equivalent reformulation of $\mathcal{L}$ is given by

$$\tilde{\mathcal{L}}(\boldsymbol{x}) = \|\boldsymbol{W}\boldsymbol{x} - \boldsymbol{y}\|^2 - \textstyle\sum_i x_i^2 \|\boldsymbol{W}_{:,i}\|^2 + \sum_i \|\boldsymbol{W}_{:,i}\|^2, \tag{20}$$

where $\boldsymbol{W}_{:,i}$ is the i'th column of $\boldsymbol{W}$.

Proof. By expanding the square and using the identity $x_i^2 = 1$ for $x_i \in \{-1, 1\}$. □

Simply adjusting the loss using this equivalence and applying ST to it, fixes the bias problem.

II) Next we ask the question, whether dropping the multiplier $\text{diag}(F'(\boldsymbol{a}))$ or changing it by another multiplier, which we call an *(internal) rescaling* of the estimator, can lead to an incorrect estimation.

Proposition B.4. If instead of $\text{diag}(F'(\boldsymbol{a}))$ any positive semidefinite diagonal matrix $\boldsymbol{\Lambda}$ is used in Alg. 1, the expected rescaled estimator preserves non-negative scalar product with the original estimator.

Proof. We write the chain (9) in a matrix form as $\boldsymbol{J}_1^\mathsf{T}\boldsymbol{\Lambda}_0(\boldsymbol{a})\boldsymbol{J}_2^\mathsf{T}(\boldsymbol{x})$, with the Jacobians $\boldsymbol{J}_1 = \frac{\partial \boldsymbol{a}}{\partial \boldsymbol{\phi}}$, $\boldsymbol{\Lambda}^0 = \text{diag}(F'(\boldsymbol{a}))$ and $\boldsymbol{J}_2(\boldsymbol{x}) = \frac{\partial \mathcal{L}(\boldsymbol{x})}{\partial \boldsymbol{x}}$. The modified gradient with $\boldsymbol{\Lambda}$ is then defined as $\boldsymbol{J}_1^\mathsf{T}\boldsymbol{\Lambda}(\boldsymbol{a})\boldsymbol{J}_2^\mathsf{T}(\boldsymbol{x})$.

We are interested in the scalar product between the expected gradient estimates:

$$\langle \mathbb{E}[\boldsymbol{J}_1^\mathsf{T}\boldsymbol{\Lambda}_0\boldsymbol{J}_2^\mathsf{T}], \mathbb{E}[\boldsymbol{J}_1^\mathsf{T}\boldsymbol{\Lambda}\boldsymbol{J}_2^\mathsf{T}]\rangle, \tag{21}$$

[3]E.g. $x_1x_2x_3$ is trilinear and thus qualifies but x_1^2 is not multi-linear.

where the expectation is over $\boldsymbol{x}$. Since neither $\boldsymbol{J}_1$ nor $\boldsymbol{\Lambda}$, $\boldsymbol{\Lambda}_0$ depend on $\boldsymbol{x}$, we can move the expectations to $\boldsymbol{J}_2$. Let $\bar{\boldsymbol{J}}_2 = \mathbb{E}\big[\frac{\partial \mathcal{L}(\boldsymbol{x})}{\partial \boldsymbol{x}}\big]$. Then the scalar product between the expected estimates becomes

$$\langle \boldsymbol{J}_1^\mathsf{T} \boldsymbol{\Lambda}_0 \bar{\boldsymbol{J}}_2^\mathsf{T}, \boldsymbol{J}_1^\mathsf{T} \boldsymbol{\Lambda} \bar{\boldsymbol{J}}_2^\mathsf{T} \rangle = \mathrm{Tr}(\bar{\boldsymbol{J}}_2 \boldsymbol{\Lambda} \boldsymbol{J}_1 \boldsymbol{J}_1^\mathsf{T} \boldsymbol{\Lambda}_0 \bar{\boldsymbol{J}}_2^\mathsf{T}). \tag{22}$$

Notice that $\boldsymbol{J}_1 \boldsymbol{J}_1^\mathsf{T}$ is positive semi-definite, $\boldsymbol{\Lambda}_0$ is also positive semi-definite since it is diagonal with non-negative entries. It follows that $\boldsymbol{R} = \boldsymbol{\Lambda} \boldsymbol{J}_1 \boldsymbol{J}_1^\mathsf{T} \boldsymbol{\Lambda}_0$ is positive semidefinite and that $\bar{\boldsymbol{J}}_2 \boldsymbol{R} \boldsymbol{J}_2^\mathsf{T}$ is positive semi-definite. Its trace is non-negative. □

We obtained that the use of an internal rescaling, in particular identity instead of F', is not too destructive: if Alg. 1 was unbiased, the rescaled estimator may be biased but it is guaranteed to give an ascend direction in the expectation so that the optimization can in principle succeed. However, assuming that Alg. 1 is biased (when $\mathcal{L}$ is not multi-linear) but gives an ascent direction in the expectation, *the ascent direction property cannot be longer guaranteed for the rescaled gradient*.

III) Next, we study whether the ST gradient is a valid ascent direction even when $\mathcal{L}$ is not multi-linear.

Proposition B.5. Let $\mathcal{L}(\boldsymbol{x})$ be such that its partial derivative $g_i = \frac{\partial \mathcal{L}}{\partial x_i}$ as a function of x_i is Lipschitz continuous for all i with a constant L. Then the expected ST gradient is an ascent direction for any $\boldsymbol{a}(\boldsymbol{\phi})$ and $\mathcal{L}(\boldsymbol{x})$ if and only if

$$\big|\mathbb{E}[g_i]\big| > L \text{ for all } i. \tag{23}$$

Proof. Sufficiency (if part). The true gradient using the local expectation form (6) expresses as

$$\mathbb{E}\Big[\textstyle\sum_i \big(\frac{\partial a_i}{\partial \boldsymbol{\phi}}\big)\big(p_z(a_i)\big) x_i \big(\mathcal{L}(\boldsymbol{x}) - \mathcal{L}(\boldsymbol{x}_{\downarrow i})\big)\Big] = \mathbb{E}[J\Delta], \tag{24}$$

where the expectation is w.r.t. $\boldsymbol{x} \sim p(\boldsymbol{x}; \boldsymbol{\phi})$ and we introduced the matrix notation $\boldsymbol{J} = \big(\frac{\partial \boldsymbol{a}}{\partial \boldsymbol{\phi}}\big)^\mathsf{T} \mathrm{diag}(p_z(\boldsymbol{a}))$, and $\Delta_i = x_i\big(\mathcal{L}(\boldsymbol{x}) - \mathcal{L}(\boldsymbol{x}_{\downarrow i})\big)$. The ST gradient replaces Δ_i with $2g_i(\boldsymbol{x})$. Since in both cases $\boldsymbol{J}$ does not depend on $\boldsymbol{x}$, the expectation can be moved to the last term. Respectively, let us define $\bar{\boldsymbol{\Delta}} = \mathbb{E}[\boldsymbol{\Delta}]$ and $\bar{\boldsymbol{g}} = \mathbb{E}[\boldsymbol{g}]$. The scalar product between the true gradient and the expected ST gradient can then be expressed as

$$\langle \boldsymbol{J}\bar{\boldsymbol{\Delta}}, \boldsymbol{J}\bar{\boldsymbol{g}} \rangle = \mathrm{Tr}(\boldsymbol{J}\bar{\boldsymbol{g}}\bar{\boldsymbol{\Delta}}^\mathsf{T}\boldsymbol{J}^\mathsf{T}). \tag{25}$$

From the relation

$$x_i(\mathcal{L}(\boldsymbol{x}) - \mathcal{L}(\boldsymbol{x}_{\downarrow i})) = \int_{-1}^{1} g_i(\boldsymbol{x}) \mathrm{d}x_i \tag{26}$$

and Lipschitz continuity of g_i in x_i we have bounds

$$2(g_i(\boldsymbol{x}) - L) \leq x_i(\mathcal{L}(\boldsymbol{x}) - \mathcal{L}(\boldsymbol{x}_{\downarrow i})) \leq 2(g_i(\boldsymbol{x}) + L). \tag{27}$$

It follows that

$$2(\mathbb{E}[\boldsymbol{g}] - L) \leq \mathbb{E}[\boldsymbol{\Delta}] \leq 2(E[\boldsymbol{g}] + L), \tag{28}$$

coordinate-wise. The outer product $\bar{\boldsymbol{g}}\bar{\boldsymbol{\Delta}}^\mathsf{T}$ is positive semidefinite iff $\bar{g}_i \bar{\Delta}_i \geq 0$ for all i. According to bounds above, this holds true if

$$(\forall i \,|\, \bar{g}_i \geq 0) \quad 2(|\bar{g}_i| - L) \geq 0 \tag{29}$$

$$(\forall i \,|\, \bar{g}_i < 0) \quad 2(|\bar{g}_i| + L) \leq 0, \tag{30}$$

or simply $(\forall i)\ |\bar{g}_i| \geq L$.

Necessity (only if part). We want to show that the requirements (23), which are simultaneous for all coordinates of $\boldsymbol{g}$, cannot be relaxed unless we make some further assumptions about $\boldsymbol{a}$ or $\mathcal{L}$. Namely, if $\exists i^*$ such that $\bar{g}_{i^*}\bar{\Delta}_{i^*} < 0$, then there exists $\boldsymbol{a}$ such that $\langle \boldsymbol{J}\bar{\boldsymbol{g}}, \boldsymbol{J}\bar{\boldsymbol{\Delta}} \rangle < 0$. *I.e.* a single wrong direction can potentially be rescaled by the downstream Jacobians to dominate the contribution of other components. This is detailed in the following steps.

Assume $(\exists i^*)\ |\bar{g}_{i^*}| < L$. Then exists $\mathcal{L}(\boldsymbol{x})$ such that the bounds (27) are tight (*e.g.* $\mathcal{L}(x) = x^2$) and therefore there will hold $\bar{g}_{i^*}\bar{\Delta}_{i^*} < 0$. Since $\boldsymbol{\Lambda} = \mathrm{diag}(p_z(\boldsymbol{a}))$ is positive semi-definite, $\boldsymbol{\Lambda}\bar{\boldsymbol{g}}\bar{\boldsymbol{\Delta}}^\mathsf{T}\boldsymbol{\Lambda}$ will preserve the non-positive sign of the component (i^*, i^*). There exists $\boldsymbol{a}(\boldsymbol{\phi})$ such that $\frac{\partial \boldsymbol{a}}{\partial \boldsymbol{\phi}}$ scales down all coordinates $i \neq i^*$ and scales up i^* such that the $\mathrm{Tr}(\boldsymbol{J}\bar{\boldsymbol{g}}\bar{\boldsymbol{\Delta}}^\mathsf{T}\boldsymbol{J}^\mathsf{T})$ is dominated by the entry (i^*, i^*). The resulting scalar product between the expected gradient and the true gradient thus can be negative. □

IV) Next we study, a typical use case when hidden binary variables are combined using a linear layer, initialized randomly. A typical initialization procedure would rescale the weights according to the size of the fan-in for each output.

Proposition B.6. Assume that the loss function is applied after a linear normalized transform of Bernoulli variables, *i.e.*, takes the form

$$\mathcal{L}(\boldsymbol{x}) = \ell(\boldsymbol{W}\boldsymbol{x}), \tag{31}$$

where $\boldsymbol{W} \in \mathbb{R}^{K\times n}$ is a matrix of normally distributed weights, normalized to satisfy $\|W_{k,:}\|_2^2 = 1$ $\forall k$. Then the expected Lipschitz constant of gradients of $\mathcal{L}$ scales as $O(\frac{1}{\sqrt{n}})$.

Proof. Let $\boldsymbol{u} = \boldsymbol{W}\boldsymbol{x}$ and let $\frac{\partial \ell}{\partial \boldsymbol{u}}$ be Lipschitz continuous with constant L. The gradient of $\mathcal{L}$ expresses as

$$g_i = \frac{\mathrm{d}\mathcal{L}(\boldsymbol{x})}{\mathrm{d}x_i} = \langle \frac{\partial \ell(\boldsymbol{u})}{\partial \boldsymbol{u}}, \boldsymbol{W}_{:,i}\rangle. \tag{32}$$

By assumptions of random initialization and normalization, $W_{k,i} \sim \mathcal{N}(0, \frac{1}{n})$. If we consider $|g_i|$ in the expectation over initialization we obtain that

$$\mathbb{E}_{\boldsymbol{W}}\left[|g_i(\boldsymbol{x}) - g_i(\boldsymbol{y})|\right] = \mathbb{E}_{\boldsymbol{W}}\left[\langle \ell'(\boldsymbol{W}\boldsymbol{x}) - \ell'(\boldsymbol{W}\boldsymbol{y}), \boldsymbol{W}_{:,i}\rangle\right] \leq L\mathbb{E}_{\boldsymbol{W}}\left[\|\boldsymbol{W}_{:,i}\|\right] = LK\sqrt{\frac{2}{n\pi}}. \tag{33}$$

Therefore g_i has expected Lipschitz constant $LK\sqrt{\frac{2}{n\pi}}$. □

The normal distribution assumption is not principal for conclusion of $O(\frac{1}{\sqrt{n}})$ dependance. Indeed, for any distribution with a finite variance it would hold as well, differing only in the constant factors. We obtain an important corollary.

Corollary B.2. As we increase the number of hidden binary units n in the model, the bias of ST decreases, at least at initialization.

V) Finally, we study conditions when a deterministic version of ST gives a valid ascent direction.

Proposition B.7. Let $\boldsymbol{x}^* = \mathrm{sign}(\boldsymbol{a})$. Let $g_i = \frac{\partial \mathcal{L}(\boldsymbol{x})}{\partial x_i}$ be Lipschitz continuous with constant L. Let $\boldsymbol{g}^* = \boldsymbol{g}(\boldsymbol{x}^*)$ and $p^* = p(\boldsymbol{x}^*|\boldsymbol{a})$. The deterministic ST gradient at $\boldsymbol{x}^*$ forms a positive scalar product with the expected stochastic ST gradient if

$$|g_i^*| \geq 2(1-p^*)L \;\; \forall i. \tag{34}$$

Proof. Similarly to Proposition B.5, let $\boldsymbol{J} = \left(\frac{\partial \boldsymbol{a}}{\partial \boldsymbol{\phi}}\right)^{\mathsf{T}} \mathrm{diag}(p_z(\boldsymbol{a}))$. The scalar product between the expected ST gradient and the deterministic ST gradient is given by

$$\langle \boldsymbol{J}\mathbb{E}[\boldsymbol{g}(\boldsymbol{x})], \boldsymbol{J}\boldsymbol{g}^*\rangle = \mathrm{Tr}\left(\mathbb{E}[\boldsymbol{g}(\boldsymbol{x})]\boldsymbol{g}^{*\mathsf{T}}\boldsymbol{J}^{\mathsf{T}}\right). \tag{35}$$

In order for it to be non-negative we need $\mathbb{E}[g(\boldsymbol{x})_i]g_i^* \geq 0$ $\forall i$. Observe that $\mathbb{E}[g(\boldsymbol{x})_i]$ is a sum that includes g_i^* with the weight p^*. We therefore need

$$\textstyle\sum_{\boldsymbol{x}\neq\boldsymbol{x}^*} p(\boldsymbol{x}|\boldsymbol{a})g(\boldsymbol{x})_i g_i^* + p^* {g_i^*}^2 \geq 0. \tag{36}$$

From Lipschitz continuity of g_i we have the bound $|g(\boldsymbol{x})_i - g_i^*| \leq L|x_i - x_i^*|$, or using that $|x_i - x_i^*| \leq 2$ we have

$$g_i^* - 2L \leq g(\boldsymbol{x})_i \leq g_i^* + 2L. \tag{37}$$

Therefore

$$g(\boldsymbol{x})_i g_i^* \geq {g_i^*}^2 - 2L|g_i^*|. \tag{38}$$

We thus can lower bound (36) as

$$\textstyle\sum_{\boldsymbol{x}\neq\boldsymbol{x}^*} p(\boldsymbol{x}|\boldsymbol{a})(|g_i^*| - 2L)|g_i^*| + p^* {g_i^*}^2 = -2L|g_i^*|(1-p^*) + {g_i^*}^2. \tag{39}$$

This lower bound is non-negative if

$$|g_i^*| \geq 2L(1-p^*). \tag{40}$$

□

Compared to Proposition B.5, this condition has an extra factor of $2(1-p^*)$. Since p^* is the product of probabilities of all units x_i^*, we expect initially $p^* \ll 1$. This condition improves at the same rate with the increase in the number of hidden units as the case covered by Proposition B.6. In addition it becomes progressively more accurate as units learn to be more deterministic, because in this case the factor $(1-p^*)$ decreases. However, note that this proposition describes the gap between deterministic ST and stochastic ST. And even when this gap diminishes, the gap between ST and the true gradient remains.

We can obtain a similar sufficient condition for the scalar product between deterministic ST and the executed true gradient, that (unlike the direct combination of Proposition B.5 and Proposition B.7) ensures an ascent direction.

Proposition B.8. Let $\boldsymbol{x}^* = \text{sign}(\boldsymbol{a})$. Let $g(\boldsymbol{x})_i = \frac{\partial \mathcal{L}(\boldsymbol{x})}{\partial x_i}$ be Lipschitz continuous with constant L. Let $\boldsymbol{g}^* = \boldsymbol{g}(\boldsymbol{x}^*)$ and $p^* = p(\boldsymbol{x}^*|\boldsymbol{a})$. The deterministic ST gradient at $\boldsymbol{x}^*$ forms a positive scalar product with the true gradient if

$$|g_i^*| \geq 2(1-p^*)L + L \;\; \forall i. \tag{41}$$

Proof. The proof is similar to Proposition B.7, only in this case we need to ensure $\mathbb{E}[\Delta_i]g_i^* \geq 0$. Using (28) we get the bounds

$$2(\mathbb{E}[\boldsymbol{g}] - L) \leq \mathbb{E}[\boldsymbol{\Delta}] \leq 2(E[\boldsymbol{g}] + L), \tag{42}$$

And using additionally (37) we get

$$2(p^* g_i^* + (1-p^*)(g_i^* - 2L) - L) \leq \mathbb{E}[\Delta_i] \leq 2(p^* g_i^* + (1-p^*)(g_i^* + 2L) + L). \tag{43}$$

Collecting the terms

$$2(g_i^* - (1-p^*)2L - L) \leq \mathbb{E}[\Delta_i] \leq 2(g_i^* + (1-p^*)2L + L). \tag{44}$$

Multiplying by g_i^* we obtain that a sufficient condition for $\mathbb{E}[\Delta_i]g_i^* \geq 0$ is

$$|g_i^*| \geq (1-p^*)2L + L. \tag{45}$$

□

C Mirror Descent and Variational Mirror Descent

C.1 Mirror Descent

Mirror descent is a widely used method for constrained optimization of the form $\min_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x})$, where $\mathcal{X} \subset \mathbb{R}^n$, introduced by Nemirovsky & Yudin (1983). Let $\Phi : \mathcal{X} \to \mathbb{R}$ be strongly convex and continuously differentiable on $\mathcal{X}$, called a *mirror map*. *Bregman divergence* $D_\Phi(\boldsymbol{x}, \boldsymbol{y})$ associated with Φ is defined as

$$D_\Phi(\boldsymbol{x}, \boldsymbol{y}) = \Phi(\boldsymbol{x}) - \Phi(\boldsymbol{y}) - \langle \nabla\Phi(\boldsymbol{y}), \boldsymbol{x} - \boldsymbol{y} \rangle. \tag{46}$$

An update of MD algorithm can be written as:

$$\boldsymbol{x}^{t+1} = \arg\min_{\boldsymbol{x}\in\mathcal{X}} \langle \boldsymbol{x}, \nabla f(\boldsymbol{x}^t) \rangle + \frac{1}{\varepsilon} D_\Phi(\boldsymbol{x}, \boldsymbol{x}^t). \tag{47}$$

In the unconstrained case when $\mathcal{X} = \mathbb{R}^n$ or in the case when the critical point is guaranteed to be in $\mathcal{X}$ (as typically ensured by the design of D_Φ), the solution can be found from the critical point equations, leading to the general form of iterates

$$\begin{aligned} \nabla\Phi(\boldsymbol{x}^{t+1}) &= \nabla\Phi(\boldsymbol{x}^t) - \varepsilon \nabla f(\boldsymbol{x}^t) \\ \boldsymbol{x}^{t+1} &= (\nabla\Phi)^{-1} \left(\nabla\Phi(\boldsymbol{x}^t) - \varepsilon \nabla f(\boldsymbol{x}^t) \right). \end{aligned} \tag{48}$$

Proposition 1. Common SGD in latent weights $\boldsymbol{\eta}$ using the *identity straight-through-weights* Alg. 2 implements SMD in the weight probabilities $\boldsymbol{\theta}$ with the divergence corresponding to F.

Proof. We start from the defining equation of MD update in the form (48). In order for (48) to match common SGD on $\boldsymbol{\eta}$ with $\eta_i = F^{-1}(\theta_i)$, the mirror map Φ must satisfy $\nabla\Phi(\boldsymbol{\theta}) = F^{-1}(\boldsymbol{\theta})$, where F^{-1} is coordinate-wise. We can therefore consider coordinate-wise mirror maps $\Phi\colon \mathbb{R} \to \mathbb{R}$. The inverse F^{-1} exists if F is strictly monotone, meaning that the noise density is non-zero on the

support. Finding the mirror map Φ explicitly is not necessary for our purpose, however in 1D case it can be expressed simply as $\Phi(x) = \int_0^x F^{-1}(\eta)d\eta$. With this coordinate-wise mirror map, the MD update can be written as

$$\boldsymbol{\eta}^{t+1} = \boldsymbol{\eta}^t - \varepsilon \frac{d\mathcal{L}}{d\boldsymbol{\theta}}\Big|_{\boldsymbol{\theta}=F(\boldsymbol{\eta}^t)}. \tag{49}$$

Thus MD on $\boldsymbol{\theta}$ takes the form of a descent step on $\boldsymbol{\eta}$ with the gradient $\frac{d\mathcal{L}}{d\boldsymbol{\theta}}$. A common SGD on $\boldsymbol{\eta}$ would use the gradient $\frac{d\mathcal{L}}{d\boldsymbol{\eta}} = \frac{\partial\boldsymbol{\theta}}{\partial\boldsymbol{\eta}}\frac{\partial\mathcal{L}}{\partial\boldsymbol{\theta}}$. Thus (49) bypasses the Jacobian $\frac{\partial\boldsymbol{\theta}}{\partial\boldsymbol{\eta}}$. This is exactly what Alg. 2 does. More precisely, when applying the same derivations that we used to obtain ST for activations in order to estimate $\frac{d\mathcal{L}}{d\boldsymbol{\theta}}$, since $F(\eta_i) = \theta_i$, we obtain that the factor $\frac{\partial}{\partial\boldsymbol{\theta}}p(w_i;\theta)$ present in (6) expresses as

$$\frac{dF(\boldsymbol{\eta})}{d\boldsymbol{\theta}} = \frac{\partial F(F^{-1}(\boldsymbol{\theta}))}{\partial\boldsymbol{\theta}} = 1 \tag{50}$$

and thus can be omitted from the chain rule as defined in Alg. 2. □

C.2 Latent Weight Decay Implements Variational Bayesian Learning

In the Bayesian learning setting we consider a model with binary weights $\boldsymbol{w}$ and are interested in estimating $p(\boldsymbol{w}|D)$, the posterior distribution of the weights given the data D and the weights prior $p(\boldsymbol{w})$. In the variational Bayesian (VB) formulation, this difficult and multi-modal posterior is approximated by a simpler one $q(\boldsymbol{w})$, commonly a fully factorized distribution, by minimizing $\mathrm{KL}(q(\boldsymbol{w})\|p(\boldsymbol{w}|D))$. Let $q(\boldsymbol{w}) = \mathrm{Ber}(\boldsymbol{w};\boldsymbol{\theta})$ and $p(\boldsymbol{w}) = \mathrm{Ber}(\boldsymbol{w};\frac{1}{2})$, both meant component-wise, i.e. fully factorized. Then the VB problem takes the form

$$\arg\min_{\boldsymbol{\theta}} \big\{ -\mathbb{E}_{(\boldsymbol{x}^0,\boldsymbol{y})\sim\text{data}}\big[\mathbb{E}_{\boldsymbol{w}\sim\mathrm{Ber}(\boldsymbol{\theta})}\big[\log p(\boldsymbol{y}|\boldsymbol{x}^0;\boldsymbol{w})\big]\big] + \tfrac{1}{N}\mathrm{KL}(\mathrm{Ber}(\boldsymbol{\theta})\|\mathrm{Ber}(\tfrac{1}{2}))\big\}, \tag{51}$$

where we have rewritten the data likelihood as expectation and hence the coefficient $1/N$ in front of the KL term appeared. This problem is commonly solved by SGD taking one sample from the training data and one sample of $\boldsymbol{w}$ and applying backpropagation (Graves, 2011). We can in principle do the same by applying an estimator for the gradient in $\boldsymbol{\theta}$.

The trick that we apply, different from common practices, is not to compute the gradient of the KL term but to keep this term explicit throughout to the proximal step leading to a *composite* MD (Zhang & He, 2018). With this we have

Proposition 2. Common SGD in latent weights $\boldsymbol{\eta}$ *with a weight decay and identity straight-through-weights* Alg. 2 is equivalent to optimizing a factorized variational approximation to the weight posterior $p(\boldsymbol{w}|\text{data})$ using a composite SMD method.

Proof. Expanding data log-likelihood as the sum over all data points, we get

$$\log p(D\,|\,\boldsymbol{w}) = \textstyle\sum_i \log p(x_i\,|\,\boldsymbol{w}) =: \sum_i l_i(\boldsymbol{w}). \tag{52}$$

When multiplying with $\frac{1}{N}$, the first term becomes the usual expected data likelihood, where the expectation is in training data and weights $\boldsymbol{w} \sim q(\boldsymbol{w})$. Expanding also the parametrization of $q(\boldsymbol{w}) = \mathrm{Ber}(\boldsymbol{w}\,|\,\boldsymbol{\theta})$, the variational inference reads

$$\arg\min_{\boldsymbol{\theta}} \big\{ -\mathbb{E}_{\boldsymbol{w}\sim\mathrm{Ber}(\boldsymbol{\theta})}\big[\tfrac{1}{N}\textstyle\sum_i l_i(\boldsymbol{w})\big] + \tfrac{1}{N}\mathrm{KL}(q(\boldsymbol{w})\|p(\boldsymbol{w})) + const\big\}. \tag{53}$$

We employ mirror descent to handle constraints $\boldsymbol{\theta} \in [0,1]^m$ similar to the above but now we apply it to this composite function, linearizing only the data part and keeping the prior KL part non-linear. Let

$$\boldsymbol{g} = \frac{1}{|I|}\sum_{i\in I}\nabla_{\boldsymbol{\theta}}\mathbb{E}_{\boldsymbol{w}\sim\mathrm{Ber}(\boldsymbol{\theta})}l_i(\boldsymbol{w})$$

be the stochastic gradient of the data term in the weight probabilities $\boldsymbol{\theta}$ using a min-batch I. The SMD step subproblem reads

$$\min_{\boldsymbol{\theta}} \big\{\boldsymbol{g}^\mathsf{T}\boldsymbol{\theta} + \tfrac{1}{\varepsilon}\mathrm{KL}(\mathrm{Ber}(\boldsymbol{\theta})\|\mathrm{Ber}(\boldsymbol{\theta}^t)) + \tfrac{1}{N}\mathrm{KL}(\mathrm{Ber}(\boldsymbol{\theta})\|\mathrm{Ber}(\tfrac{1}{2}))\big\}. \tag{54}$$

We notice that $\mathrm{KL}(\mathrm{Ber}(\boldsymbol{\theta})\|\mathrm{Ber}(\frac{1}{2})) = -H(\mathrm{Ber}(\boldsymbol{\theta}))$, the negative entropy, and also introduce the prior scaling coefficient $\lambda = \frac{1}{N}$ in front of the entropy, which may optionally be lowered to decrease the regularization effect. With these notations, the composite proximal problem becomes

$$\min_{\boldsymbol{\theta}} \left\{\boldsymbol{g}^\mathsf{T}\boldsymbol{\theta} + \tfrac{1}{\varepsilon}\mathrm{KL}(\mathrm{Ber}(\boldsymbol{\theta})\|\mathrm{Ber}(\boldsymbol{\theta}^t)) - \lambda H(\mathrm{Ber}(\boldsymbol{\theta}))\right\}. \tag{55}$$

The solution is found from the critical point equation in $\boldsymbol{\theta}$:

$$\nabla_{\boldsymbol{\theta}}\big(\boldsymbol{g}^\mathsf{T}\boldsymbol{\theta} + \tfrac{1}{\varepsilon}\mathrm{KL}(\mathrm{Ber}(\boldsymbol{\theta})\|\mathrm{Ber}(\boldsymbol{\theta}^t)) - \lambda H(\mathrm{Ber}(\boldsymbol{\theta}))\big) = 0 \tag{56a}$$

$$g_i + \tfrac{1}{\varepsilon}\left(\log\tfrac{\theta_i}{1-\theta_i} - \log\tfrac{\theta_i^t}{1-\theta_i^t}\right) + \lambda\log\tfrac{\theta_i}{1-\theta_i} = 0 \tag{56b}$$

$$(\varepsilon\lambda + 1)\log\tfrac{\theta_i}{1-\theta_i} = \log\tfrac{\theta_i^t}{1-\theta_i^t} - \varepsilon g_i \tag{56c}$$

$$\log\tfrac{\theta_i}{1-\theta_i} = \tfrac{1}{\varepsilon\lambda+1}\log\tfrac{\theta_i^t}{1-\theta_i^t} - \tfrac{\varepsilon}{\varepsilon\lambda+1}g_i. \tag{56d}$$

For the natural parameters we obtain:

$$\boldsymbol{\eta} = \tfrac{\boldsymbol{\eta}^t - \varepsilon\boldsymbol{g}}{\varepsilon\lambda+1} = \boldsymbol{\eta}^t - \tfrac{\varepsilon}{\varepsilon\lambda+1}\Big(\lambda\boldsymbol{\eta}^t + \boldsymbol{g}\Big). \tag{57}$$

We can further drop the correction of the step size $\frac{\varepsilon}{\varepsilon\lambda+1}$ since $\varepsilon\lambda + 1 \approx 1$ and the step size will need to be selected by cross validation anyhow. This gives us an update of the form

$$\boldsymbol{\eta} = \boldsymbol{\eta}^t - \varepsilon(\boldsymbol{g} + \lambda\boldsymbol{\eta}^t), \tag{58}$$

which is in the form of a standard step in any SGD or adaptive SGD optimizer. The difference is that the gradient in probabilities $\boldsymbol{\theta}$ is applied to make step in logits $\boldsymbol{\eta}$ and the prior KL divergence contributes the *logit decay* λ, which in this case is the *latent weight decay*. Since the ST gradient in $\boldsymbol{\theta}$ differs from the ST gradient in $\boldsymbol{\eta}$ by the factor $\mathrm{diag}(F')$, the claim of Proposition 2 follows. □

D Details of Experiments

D.1 Stochastic Autoencoder

It was shown in the literature that semantic hashing using binary hash codes can achieve superior results using learned hash codes, in particular based on variational autoencoder (VAE) formulation, *e.g.*, recent works of Chaidaroon & Fang (2017); Dadaneh et al. (2020); Nanculef et al. (2020). We consider a simplified yet realistic model, similar to the unsupervised setup in the mentioned works. However, we do not use the variational evidence lower bound formulation (also referred to as VAE), just a plain stochastic autoencoder (2). It can be seen that the variational formulation adds to the objective (2) the negative entropy of the encoder distribution (KL to the uniform Bernoulli prior). This entropy can be computed in closed form and consequently has an analytic gradient that does not need the techniques we study. It also changes he optimization problem, leading to an automatic selection of the number of bits efficiently used by the encoder while the remaining bits are found to be in the posterior collapse (their distribution $q(x_i|\boldsymbol{y})$ is uniform and does not depend on the input $\boldsymbol{y}$). This property is in contradiction with our goal to study the performance of estimators with the increase of the latent code size. In practice, the balance between the data term and the KL prior is often controlled by an extra hyper-parameter β (in the public implementation of Nanculef et al. (2020) one can find $\beta = 0.015$ is used). As the KL prior term imposes lots of issues that are orthogonal to this study, we chose to test with plain stochastic autoencoders, that correspond to the setting $\beta = 0$ in β-VAE.

Dataset The *20Newsgroups* data set[4] is a collection of approximately 20,000 text documents, partitioned (nearly) evenly across 20 different newsgroups. In our experiments we do not use the partitioning. We used the processed version of the dataset denoted as Matlab/Octave on the dataset's web site. It contains bag-of-words representations of documents given by one sparse word-document count matrix. We worked with the training set that contains 11269 documents in the bag of words representation.

[4] `http://qwone.com/~jason/20Newsgroups/`

Table D.1: List of estimators evaluated in the stochastic autoencoder experiment.

Name	Details
ARM	State-of-the-art unbiased estimator Yin & Zhou (2019).
Gumbel(τ)	Relaxation-based biased estimator Jang et al. (2016) with temperature parameter $\tau = 1$ or 0.1.
ST	Hinton's ST variant in Alg. 1.
det_ST	Deterministic version of ST setting the noise $\boldsymbol{z} = 0$ always during training.
identity_ST	Identity ST variant described by Bengio et al. (2013).

Preprocessing We keep only the 10000 most frequent words in the training set to reduce the computation requirements. Each of the omitted rare words occurs not more than in 10 documents.

Reconstruction Loss Let $\boldsymbol{y} \in \mathbb{N}^d$ be the vector of word counts of a document and $\boldsymbol{x} \in \{0,1\}^n$ be a latent binary code representing the topic that we will learn. The decoder network given the code $\boldsymbol{x}$ deterministically outputs word frequencies $\boldsymbol{f} \in [0,1]^d$, $\sum_i f_i = 1$ and the reconstruction loss $-\log p^{\text{dec}}(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta})$ is defined as

$$-\textstyle\sum_i y_i \log f_i, \tag{59}$$

i.e., the negative log likelihood of a generative model, where word counts $\boldsymbol{y}$ follow multinomial distribution with probabilities $\boldsymbol{f}$ and the number of trials equal to the length of the document. The encoder $p(\boldsymbol{x}|\boldsymbol{f};\boldsymbol{\phi})$ obtains word frequencies form $\boldsymbol{y}$ and maps them deterministically to Bernoulli probabilities $p(x_i|\boldsymbol{f};\boldsymbol{\phi})$. The loss of the autoencoder (2) is then

$$\textstyle\sum_{\boldsymbol{y}\sim\text{data}} \mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{x}|\boldsymbol{y})} - \log p^{\text{dec}}(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta}). \tag{60}$$

Networks Specs The encoder network takes on the input word frequencies $\boldsymbol{f} \in \mathbb{R}^d$ and applies the following stack: FC($d \times 512$), ReLU, FC($512 \times n$), where FC is a fully connected layer. The output is the vector of logits of Bernoulli latent bits. The decoder network is symmetric: FC($n \times 512$), ReLU, FC($512 \times d$), Softmax. Its input is a binary latent code $\boldsymbol{x}$ and output is the word probabilities $\boldsymbol{f}$. Standard weight initialization is applied to all linear layers $\boldsymbol{W}$ setting $W_{i,j} \sim \mathcal{U}[-1/\sqrt{k}, 1/\sqrt{k}]$, where k is the number of input dimensions to the layer. This is a standard initialization scheme (He et al., 2015), which is consistent with the assumptions we make in Proposition B.6 and hence important for verification of our analysis.

Estimators Estimators evaluated in this experiment are described in Table D.1. As detailed in § 2, in the identity ST we still draw random samples in the forward pass like in Alg. 1 but omit the multiplication by F'. Alg. 1 is correctly instantiated for the $\{0,1\}$ rather than ± 1 encoding in all cases. For the ARM-10 correction phase and ARM-1000 ground truth estimation, the average of ARM estimates with the respective number of samples is taken.

Optimizer We used Adam (Kingma & Ba, 2014) optimizer with a fixed starting learning rate $lr = 0.001$ in both phases of the training. When switching to the ARM-10 correction phase, we reinitialize Adam in order to reset the running averages.

Evaluation For each bit length we save the encoder and decoder parameter vectors $\boldsymbol{\phi}, \boldsymbol{\theta}$ every 100 epochs along the ARM training trajectory. At each such point, offline to the training, we first apply ARM-1000 in order to obtain an accurate estimate of the true gradient $\boldsymbol{g}$. We then evaluate each of the 1-sample estimators, including ARM itself (= ARM-1 for that matter). The next question we discuss is how to measure the estimator accuracy. Clearly, if we just consider the expected local performance such as $\mathbb{E}[\langle \boldsymbol{g}, \tilde{\boldsymbol{g}} \rangle]$, unbiased estimators win regardless how high is their variance. This is therefore not appropriate for measuring their utility in optimization. We evaluate three metrics tailored for comparison of biased and unbiased estimators.

Cosine Similarity This metric evaluates the expected cosine similarity, measuring alignment of directions:

$$\mathbb{E}\Big[\langle \boldsymbol{g}, \tilde{\boldsymbol{g}} \rangle / (\|\boldsymbol{g}\| \|\tilde{\boldsymbol{g}}\|)\Big], \tag{61}$$

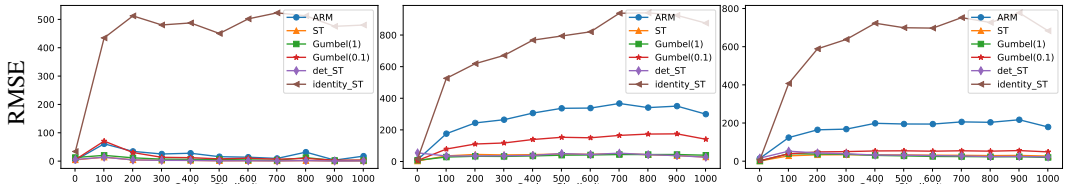

Figure D.1: Root Mean Squared error of different estimators for the same reference trajectories as Fig. 1.

where the expectation is over all training data batches and 100 stochastic trials of the estimator $\tilde{\boldsymbol{g}}$. This metric is well aligned with our theoretical analysis § 2.2. It is however does not measure how well the gradient length is estimated. If the length has a high variance, this may hinder the optimization but would not be reflected in this metric.

Expected Improvement To estimate the utility of the estimator for optimization, we propose to measure the expected optimization improvement using the same proximal problem objective that is used in SGD or SMD to find the optimization steps. Namely, let $\boldsymbol{g} = \nabla_{\boldsymbol{\phi}}\mathcal{L}(\boldsymbol{\phi}^t)$ be the true gradient at the current point. The common SGD step is defined as

$$\boldsymbol{\phi}^{t+1} = \boldsymbol{\phi}^t + \arg\min_{\Delta\boldsymbol{\phi}} \left(\langle g, \Delta\boldsymbol{\phi}\rangle + \tfrac{1}{2\varepsilon}\|\Delta\boldsymbol{\phi}\|^2\right). \tag{62}$$

The optimal solution is given by $\Delta\boldsymbol{\phi} = -\varepsilon\boldsymbol{g}$. Since instead of $\boldsymbol{g}$, only an approximation is available to the optimizer, we allow it to use the solution $\Delta\boldsymbol{\phi} = -\alpha\hat{\boldsymbol{g}}$, where $\hat{\boldsymbol{g}}$ is an estimator of $\boldsymbol{g}$ and α is one scalar parameter to adopt the step size. We consider the expected change of the proxy objectives:

$$\mathbb{E}\Big[\langle \boldsymbol{g}, -\alpha\hat{\boldsymbol{g}}\rangle + \tfrac{\alpha^2}{2\varepsilon}\|\hat{\boldsymbol{g}}\|^2\Big]. \tag{63}$$

The parameter α correspond to a learning rate that can be tuned or adapted during learning. We set it optimistically for each estimator by minimizing the expected objective (63), which is a simple quadratic function in α. One scalar α is thus estimated for one measuring point (*i.e.* for one expectation over all training batches and all 100 trials). As such, it is not overturning too much to each estimator. The optimal α is given by

$$\alpha = \varepsilon\mathbb{E}[\langle \boldsymbol{g}, \hat{\boldsymbol{g}}\rangle]/\mathbb{E}[\|\hat{\boldsymbol{g}}\|^2] \tag{64}$$

and the value of the objective for this optimal α is

$$-\tfrac{\varepsilon}{2}\mathbb{E}[\langle \boldsymbol{g}, \hat{\boldsymbol{g}}\rangle]^2/\mathbb{E}[\|\hat{\boldsymbol{g}}\|^2]. \tag{65}$$

For the purpose of comparing estimators, $-\frac{\varepsilon}{2}$ is irrelevant and the comparison can be made on the square root of (65). We obtain an equivalent metric that is the expected loss decrease normalized by the RMS of the gradients:

$$-\mathbb{E}[\langle \boldsymbol{g}, \hat{\boldsymbol{g}}\rangle]/\sqrt{\mathbb{E}[\|\hat{\boldsymbol{g}}\|^2]}. \tag{66}$$

Confer to common adaptive methods which divide the step-length exactly by the square root of a running average of second moment of gradients, in particular Adam (applied per-coordinate there). This suggests that this metric is more tailored to measure the utility of the estimator for optimization. For brevity, we refer to (66) as *expected improvement*. Note also that in (66) we preserve the sign of $\mathbb{E}[\langle \boldsymbol{g}, \hat{\boldsymbol{g}}\rangle]$ and if the estimator is systematically in the wrong direction, we expect to measure a positive value in (66), *i.e.* predicting objective ascent rather than descent.

Root Mean Squared Error It is rather common to measure the error of biased estimators as

$$\text{RMSE} = \sqrt{\mathbb{E}[\|\boldsymbol{g} - \tilde{\boldsymbol{g}}\|^2]}. \tag{67}$$

This metric however may be less indicative and less discriminative of the utility of the estimator for optimization. In Fig. D.1 it is seen that RMSE of ARM estimator can be rather high, especially with more latent bits, yet it performs rather well in optimization.

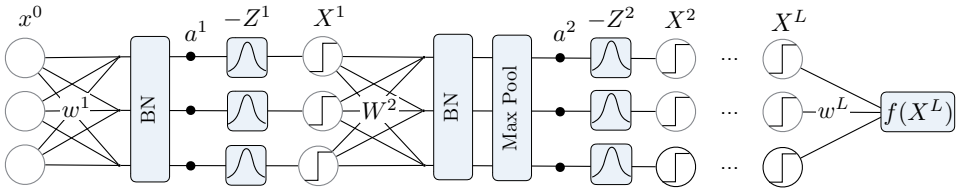

Figure D.2: Stochastic Binary Network: first and last layer have real-valued weights. BN layers have real-valued scale and bias parameters that can adjust scaling of activations relative to noise. Z are independent injected noises with a chosen distribution. Binary weights W_{ij} are random ± 1 Bernoulli(θ_{ij}) with learnable probabilities θ_{ij}. In experiments we consider SBN with a convolutional architecture same as Courbariaux et al. (2015); Hubara et al. (2016): $(2{\times}128\text{C3}) - \text{MP2} - (2{\times}256\text{C3}) - \text{MP2} - (2{\times}512\text{C3}) - \text{MP2} - (2{\times}1024\text{FC}) - 10\text{FC} - \text{softmax}$.

D.2 Classification with Deep Stochastic Binary Networks

The verification of ST estimator in training deep neural networks with mirror descent is conducted on CIFAR-10 dataset[5].

Dataset The dataset consists of 60000 32x32 color images divided in 10 classes, 6000 images per class. There is a predefined training set of 50000 examples and test set of 10000 examples.

Preprocessing During training we use standard augmentation for CIFAR-10, namely random horizontal flipping and random cropping of 32×32 region with a random padding of 0-4 px on each side.

Network The network structure and layer specifications are shown in (D.2).

Optimizer We use Adam optimizer (Kingma & Ba, 2014) in all the experiments. The initial learning rate $\gamma = 0.01$ is used for 300 epochs and then we divide it by 10 at epochs 300 and 400 and stop at epoch 500. This is fixed for all models. All other Adam hyper-parameters such as $\beta_1, \beta_2, \varepsilon$ are set to their correspondent default values in the PyTorch Paszke et al. (2019) framework.

Training Loss Let the network softmax prediction on the input image $\boldsymbol{x}^0$ with noise realizations in all layers $\boldsymbol{z}$ be denoted as $p(\boldsymbol{x}|\boldsymbol{z}, \boldsymbol{x}^0)$. The training loss for the stochastic binary network is the expected loss under the noises:

$$\mathbb{E}_{\boldsymbol{x}^0 \sim \text{data}}\big[\mathbb{E}_{\boldsymbol{z}}[-\log p(\boldsymbol{x}|\boldsymbol{z}, \boldsymbol{x}^0)]\big]. \tag{68}$$

The training procedure is identical to how the neural networks with dropout noises are trained (Srivastava et al., 2014): one sample of the noise is generated alongside each random data point.

Evaluation At the test time we can either set $\boldsymbol{z} = 0$ to obtain a deterministic binary network (denoted as 'det'). We can also consider the network as a stochastic ensemble and obtain the prediction via the expected predictive distribution

$$\mathbb{E}_{\boldsymbol{z}}[p(\boldsymbol{x}|\boldsymbol{z}, \boldsymbol{x}^0)], \tag{69}$$

approximated by several samples. In the experiments we report performance in this mode using 10 samples. We observed that increasing the number of samples further improves the accuracy only marginally. We compute the mean and standard deviation for the obtained accuracy values by averaging the results over 4 different random learning trials for each experiment.

[5] https://www.cs.toronto.edu/~kriz/cifar.html

E GUMBEL SOFTMAX AND ST GUMBEL-SOFTMAX

Gumbel Softmax (Jang et al., 2016) or Concrete relaxations (Maddison et al., 2016) are techniques that overcome the problem of non-differentiability of $\arg\max$ (in binary case equivalent to sign in (1)) by approximating it with a smooth function whose degree of smoothness is controlled by a temperature-like parameter $\tau > 0$, where the limit $\tau \to 0$ recovers the original non-smooth function.

We analyze this estimator for the case of a single neuron and show the following properties, which to our knowledge were not published before:

- **I)** GS estimator is asymptotically unbiased as $\tau \to +0$ and the bias decreases at the rate $O(\tau)$. But for any fixed τ it is biased even when $\mathcal{L}$ is linear.
- **II)** For any given noise realization $z \neq \eta$ the GS gradient norm approaches zero at the exponential rate $O(\frac{1}{\tau}c^{\frac{1}{\tau}})$, where $c = e^{-|\eta - z|} < 1$.
- **III)** The probability of the event that gradient norm is at least ε is asymptotically $O(\tau \log \frac{1}{\varepsilon})$. Thus the probability to observe a non-zero gradient up to numerical precision quickly vanishes.
- **IV)** The variance of GS estimator grows at the rate $O(\frac{1}{\tau})$.
- **V)** The ST Gumbel-softmax estimator (Jang et al., 2016) is biased even asymptotically even for a linear loss.

Properties II and III are easy to extend to the case of layer with multiple units since they apply just to the factor $\frac{\partial}{\partial \eta}\sigma_\tau(\eta - z)$, which is present independently at all units. Property II can be extended to deep networks with L layers of Bernoulli variables, in which case the chain derivative will encounter L such factors and we obtain that the gradients will vanish at the rate $O(\tau^L)$.

The proofs are given below. Basically all these facts should convince the reader of the following: it is not possible to use a very small τ, not even with an annealing schedule starting from $\tau = 1$. For very small τ, the most likely consequence would be to never encounter a non-zero gradient during the whole training. For moderately small τ the variance would be prohibitively high. Indeed, Jang et al. (2016) anneals τ only down to 0.5 in their experiments.

Definitions The *Gumbel-max* reparametrization constructs a sample from a categorical distribution with class probabilities π_i as

$$x = \text{one_hot}(\arg\max_i(\log \pi_i + \Gamma_i)), \tag{70}$$

where x is 1-hot encoding of the class and Γ_i are independent Gumbel noises. This reparametrization is exact and is not differentiable since it outputs discrete samples. The *Gumbel-Softmax* replaces the hard $\arg\max$ indicator with softmax to compute the relaxed sample $y \in \mathbb{R}^n$ as

$$y = \text{softmax}((\tfrac{1}{\tau}(\log \pi_i + \Gamma_i))_i). \tag{71}$$

We will focus on the binary case. We can express then

$$x_1 = [\![\log \pi_1 + \Gamma_1 \geq \log \pi_0 + \Gamma_0]\!] = [\![\eta - z \geq 0]\!], \tag{72}$$

where $\eta = \log \frac{\pi_1}{1-\pi_1}$ and $z = \Gamma_0 - \Gamma_1$. As expected z has logistic distribution and we recover that $p(x_1{=}1) = \sigma(\eta)$.

The relaxation y takes a similar form. By dividing over the numerator in (71) we obtain:

$$y_1 = \frac{1}{1+\exp(-(\eta - z)/\tau)} = \sigma(\tfrac{\eta - z}{\tau}) =: \sigma_\tau(\eta - z) \tag{73}$$

$$y_0 = 1 - y_1. \tag{74}$$

The loss function $\mathcal{L}$ is extended to the simplex of (y_1, y_0) by

$$\tilde{\mathcal{L}}(y_1, y_0) = \mathcal{L}(y_1). \tag{75}$$

Recall that ST estimates the gradient in η by sampling x_1 and composing

$$\frac{\partial \mathcal{L}}{\partial x_1}\frac{\partial \sigma(\eta)}{\partial \eta}. \tag{76}$$

The Gumbel-Softmax estimates the gradient by sampling z and computing

$$\hat{G}_\tau := \frac{d}{d\eta}\tilde{\mathcal{L}}(y(\eta)) = \frac{d}{d\eta}\mathcal{L}(\sigma_\tau(\eta - z)) = \frac{\partial \mathcal{L}(y_1)}{\partial y_1}\frac{\partial \sigma_\tau(\eta - z)}{\partial \eta}. \tag{77}$$

So the gradient of the relaxed loss is multiplied with the gradient of stretched (assuming $\tau < 1$) and shifted sigmoid.

Proposition E.1. The estimator is asymptotically unbiased as $\tau \to 0$ and the bias decreases at the rate $O(\tau)$.

Proof. Let us denote

$$\hat{g}_t = \mathbb{E}_z[\hat{g}_t] = \int_{-\infty}^{\infty} \frac{d}{d\eta}\mathcal{L}(\sigma_\tau(\eta - z))p_Z(z)\mathrm{d}z. \tag{78}$$

Note that the limit $\lim_{\tau\to 0+} g_t(x)$ cannot be simply interchanged with the integral above – no qualification theorem allows this. We apply the following reformulation. The derivative $\frac{d}{d\eta}\mathcal{L}(\sigma_\tau(\eta - z))$ expands as

$$\mathcal{L}'(\sigma_\tau(\eta - z))\sigma_\tau(\eta - z)(1 - \sigma_\tau(\eta - z))\tfrac{1}{\tau}. \tag{79}$$

We make a change of variables $v = \sigma_\tau(\eta - z)$ in the integral. This gives $z = \eta - \tau\mathrm{logit}(v)$ and $\mathrm{d}z = -\tau\frac{1}{v(1-v)}\mathrm{d}v$. Substituting and cancelling part of the terms, we obtain

$$\hat{g}_t = \int_0^1 \mathcal{L}'(v)p_Z(\eta - \tau\mathrm{logit}(v))\mathrm{d}v. \tag{80}$$

With this expression we can now interchange the limit and the integral using the dominated convergence theorem. In order to apply it we need to show that there exist an integrable function $\bar{g}(v)$ such that

$$|\mathcal{L}'(v)p_Z(\eta - \tau\mathrm{logit}(v))| < \bar{g}(v) \tag{81}$$

for all $\tau > 0$. Observe that

$$\sup_{v\in[0,1]} |p_Z(\eta - \tau\mathrm{logit}(v))| = \sup_{u\in\mathbb{R}} |p_Z(\eta - \tau u)| = \sup_{y\in\mathbb{R}} |p_Z(y)| = p_Z(0) = \tfrac{1}{4}, \tag{82}$$

where we used that the maximum of standard logistic density is attained at zero. We can therefore let $\bar{g}(v) = \mathcal{L}'(v)/4$. Since $\mathcal{L}'(v)$ is the derivative of $\mathcal{L}$, it is integrable on $[0, 1]$. Therefore the conditions of the dominated convergence theorem are satisfied and we have

$$\lim_{\tau\to 0+} \hat{g}_\tau = \int_0^1 \mathcal{L}'(v) \lim_{\tau\to 0+} p_Z(\eta - \tau\mathrm{logit}(v))\mathrm{d}v \tag{83a}$$

$$= \int_0^1 \mathcal{L}'(v)p_Z(\eta)\mathrm{d}v = (\mathcal{L}(1) - \mathcal{L}(0))p_Z(\eta), \tag{83b}$$

which is the correct value of the gradient.

Next, we obtain the series representation of the estimator bias in the asymptote $\tau \to 0+$. We approximate $p_Z(\eta - \tau\mathrm{logit}(v))$ with its Taylor series around $\tau = 0$:

$$p_Z(\eta - \tau\mathrm{logit}(v)) = p_Z(\eta) + c_1\mathrm{logit}(v)\tau + c_2\mathrm{logit}^2(v)\tau^2 + O(\tau^3) \tag{84}$$

where

$$c_1 = p_Z(\eta)\tfrac{e^\eta - 1}{e^\eta + 1}; \quad c_2 = p_Z(\eta)\tfrac{-4e^\eta + e^{2\eta} + 1}{2(e^\eta + 1)^2}. \tag{85}$$

This is obtained using mathematica. We use this expansion in the integral representation (80). Observing that $g = p_Z(\eta)\int_0^1 \mathcal{L}'(v)\mathrm{d}v$, the zero order term becomes the true gradient. It follows that the bias of G_τ is asymptotically

$$c_1\Big(\int_0^1 \mathcal{L}'(v)\mathrm{logit}(v)\mathrm{d}v\Big)\tau + c_2(x)\Big(\int_0^1 \mathcal{L}'(v)\mathrm{logit}^2(v)\mathrm{d}v\Big)\tau^2 + O(\tau^3). \tag{86}$$

In the case when $\mathcal{L}$ is linear, the first order term vanishes because $\mathcal{L}'$ is constant and $\mathrm{logit}(v)$ is odd about $\frac{1}{2}$. However τ^2 and higher order even terms do not vanish, therefore the estimator is still biased even for linear objectives. □

This property sounds as a good and desirable one, but it's advantage is practically nullified by the next properties.

Proposition E.2. For any given realization $z \neq \eta$ the norm of Gumbel-Softmax gradient estimate asymptotically vanishes at the rate $O(\frac{1}{\tau}c^{1/\tau})$ with $c = e^{-|x|} < 1$.

Proof. Considering z fixed and denoting $x = \eta - z$, we need to check the asymptotic behavior of

$$\frac{\mathrm{d}}{\mathrm{d}x}\sigma_\tau(x) = \frac{1}{\tau}p_Z(x/\tau) = \frac{e^{-\frac{x}{\tau}}}{\tau(1+e^{-x/\tau})^2} \tag{87}$$

as $\tau \to 0+$. Since p_Z is symmetric, we may assume $x > 0$ without loss of generality. The denominator is then asymptotically just τ. Therefore the ratio is asymptotically $O(\frac{1}{\tau}c^{1/\tau})$. □

For small x, where c is close to one, the term $1/\tau$ dominates at first. In particular for $z = \eta$, we get $c = 1$ and the asymptote is $O(1/\tau)$. So while for most of the noise realization the gradient magnitude quickly vanishes, it is compensated by a significant grows at rate $1/\tau$ around $z = \eta$. In practice it means, most of the time a value of gradient close to zero is measured and occasionally, very rarely, a value of $O(1/\tau)$ is obtained.

Since the gradient is asymptotically unbiased, such quick diminishing for any fixed ε has to be compensated by a rapid growth

Proposition E.3. The probability to observe the gradient of norm at least ε is asymptotically $O(\tau \log(\frac{1}{\varepsilon}))$, where the asymptote is $\tau \to 0$, $\varepsilon \to 0$.

Proof. We want to analyze the probability

$$P = \mathbb{P}(\frac{\mathrm{d}}{\mathrm{d}\eta}\sigma_\tau(\eta - z) \geq \varepsilon) \tag{88}$$

when z is distributed logistically. Let $s = \sigma_\tau(\eta - z)$. Then $\frac{\mathrm{d}}{\mathrm{d}\eta}\sigma_\tau(\eta - z) = s(1-s)$. The equality $s(1-s) = \varepsilon$ holds for $s^* = \frac{1}{2}(1 - \sqrt{1-4\varepsilon})$. This implies

$$z_{1,2} = \eta \pm \tau \mathrm{logit}(s^*). \tag{89}$$

The inequality $s(1-s) \geq \varepsilon$ holds in the interval $[z_1, z_2]$. Thus the probability in question is given by

$$P = F_Z(z_2) - F_Z(z_1). \tag{90}$$

As $\tau\mathrm{logit}(s^*) \to 0$ for $\tau \to 0$, we have asymptotically that

$$P = p_Z(\eta)\tau\mathrm{logit}(s^*). \tag{91}$$

Lastly note that $\mathrm{logit}(\frac{1}{2}(1 - \sqrt{1-4\varepsilon}))$ is asymptotically $O(-\log\varepsilon)$ for $\varepsilon \to 0$. □

Proposition E.4. The variance of GS estimator grows as $O(\frac{1}{\tau})$.

Proof. We first show that the second moment of the estimator G_τ has the following asymptotic expansion for $\tau \to 0+$:

$$p_Z(\eta)\Big(\int_0^1 \mathcal{L}'(v)^2 v(1-v)\Big)\frac{1}{\tau} + c_1\Big(\int_0^1 \mathcal{L}'(v)^2 v(1-v)\mathrm{logit}(v)\mathrm{d}v\Big) \tag{92}$$

$$+ c_2\Big(\int_0^1 \mathcal{L}'(v)^2 v(1-v)\mathrm{logit}^2(v)\mathrm{d}v\Big)\tau + O(\tau^2). \tag{93}$$

The second moment expresses as

$$\mathbb{E}[G_\tau^2] = \int_{-\infty}^{\infty}(\frac{\partial}{\partial\eta}\mathcal{L}(\sigma_\tau(\eta - z)))^2 p_Z(z)\mathrm{d}z \tag{94}$$

$$= \int_{-\infty}^{\infty}\Big(\mathcal{L}'(\sigma_\tau(\eta - z))\sigma_\tau(\eta - z)(1 - \sigma_\tau(\eta - z))\frac{1}{\tau}\Big)^2 p_Z(z)dz \tag{95}$$

We perform the same substitution of variables: $v = \sigma_t(\eta - z)$, $dv = -v(1-v)\frac{1}{\tau}\mathrm{d}z$ to obtain

$$\mathbb{E}[G_\tau^2] = \int_0^1 \mathcal{L}'(v)v(1-v)\frac{1}{\tau}\mathcal{L}'(v)p_Z(\eta - \tau\mathrm{logit}(v))\mathrm{d}v \tag{96}$$

$$= \frac{1}{\tau}\int_0^1 \mathcal{L}'(v)^2 v(1-v)p_Z(\eta - \tau\mathrm{logit}(v))\mathrm{d}v. \tag{97}$$

We perform the same Taylor expansion for $p_Z(\eta - \tau\mathrm{logit}(v))$ around $\tau = 0$ as in Proposition E.1 and combine the terms to obtain the expansion as claimed. The variance is therefore dominated by the $O(\frac{1}{t})$ term of the second moment. □

Unlike ST, for linear objective $\mathcal{L}$, this estimator is biased. Furthermore, unlike ST, even for a single neuron it still has non-zero variance.

ST Gumbel-Softmax Considering that relaxed variables deviate from binary samples on the forward pass, Jang et al. (2016) also proposd the following empirical modification related to our main topic on ST methods. Their *ST Gumbel-Softmax* estimator replaces the gradient estimator with

$$\frac{\partial \mathcal{L}(x_1)}{\partial x_1} \frac{\partial y_1}{\partial \eta}, \tag{98}$$

where both x_1 and y_1 are computed using the same logistic noise z. Since $\frac{\partial \mathcal{L}(x_1)}{\partial x_1}$ does not vary with z unless x changes, to compute the expected gradient we can marginalize over z locally in the two regions: $z < \eta$ and $z > \eta$. Let's consider the first case. We have:

$$\int_{-\infty}^{\eta} \frac{1}{\tau} \sigma_\tau(\eta - z)(1 - \sigma_\tau(\eta - z)) p_Z(z) \mathrm{d}z = \tag{99}$$

we make a substitution $z = \eta - \tau \mathrm{logit}(v)$. Then $v = \sigma_\tau(\eta - z)$ and $\mathrm{d}z = -\tau \frac{1}{v(1-v)} \mathrm{d}v$. The integral expresses as

$$-\int_{1}^{0.5} p_Z(\eta - \tau \mathrm{logit}(v)) \mathrm{d}v. \tag{100}$$

The limit of this integral with $\tau \to 0$ is $\frac{1}{2} p_Z(\eta)$. And the same holds for the case $x_1 = 0$, *i.e.*, $z > \eta$. What we obtained, is that ST Gumbel-Softmax in the simple case of linear objective and in the favorable limit $\tau \to 0$ underestimates the true gradient by $1/2$.

To summarize, ST Gumbel-Softmax is more expensive as it involves sampling from Logistic distribution and keeping the samples, it is biased, even asymptotically as $\tau \to 0$ and even in the case of linear objective. It is also more noisy than ST as the gradient depends on the value of z and not only the binary state. In particular for one neuron it is still stochastic while ST becomes deterministic.

F Analysis of BayesBiNN

BayesBiNN Meng et al. (2020) algorithm is derived from the same variational Bayesian learning problem formulation that we consider § C.2. The update step is also the same as they differentiate the entropy regularization in closed form similarly to our composite MD. The only essential difference is the use of Gumble-Softmax estimator to estimate the gradient in the probabilities $\boldsymbol{\theta}$ used to update the natural parameters $\boldsymbol{\eta}$. We will show that the Gumbel-Softmax estimator is used incorrectly, which leads to some surprising algorithm behavior.

First, it is easy to see that the properties of Gumbel-Softmax estimator § E for gradient in $\boldsymbol{\eta}$ apply as well to the gradient in $\boldsymbol{\theta}$. Indeed, for Bernoulli distribution we have

$$\frac{\partial \boldsymbol{\eta}}{\partial \boldsymbol{\theta}} = \left(\frac{\partial \boldsymbol{\theta}}{\partial \boldsymbol{\eta}} \right)^{-1} = \mathrm{diag}(p_z(\boldsymbol{\theta}))^{-1}. \tag{101}$$

This link between the two gradients does not depend on the relaxation parameter τ and therefore the asymptotic properties § E apply to the Gumbel-Softmax gradient in $\boldsymbol{\theta}$ as well. With parameter $\tau = 10^{-10}$, the probability to measure a gradient larger than a numerical precision scales as $O(\tau)$ and practically diminishes. Even if such a rare event occurs, the value of the measured gradient scales as $O(1/\tau)$. The experiments of Meng et al. (2020) could not possibly be successful without a technical issue that we discuss next.

The BayesBiNN algorithm (Meng et al., 2020, Table 1 middle) performs the steps (in our notation):

$$\boldsymbol{w}_b := \tanh(\tfrac{\boldsymbol{\eta} - \boldsymbol{z}}{\tau}); \tag{102a}$$

$$\boldsymbol{g} := \nabla_{\boldsymbol{w}_b} \mathcal{L}; \tag{102b}$$

$$\boldsymbol{\eta} := (1 - \alpha)\boldsymbol{\eta} - \alpha \boldsymbol{s} \odot \boldsymbol{g}, \tag{102c}$$

where $\boldsymbol{g}$ denotes the gradient of the average min-batch loss $\mathcal{L}$, which is evaluated using softly binarized weights $\boldsymbol{w}_b$, $\odot$ is a component-wise product and $\boldsymbol{s}$ is a scaling factor originating from Gumbel-Softmax estimator and is discussed below.

The actual scaling factor $\boldsymbol{s}$ used in the experiments Meng et al. (2020) according to the published code adds a technical ε in (Meng et al., 2020, Eq. (9)) in the implementation of Gumbel-Softmax

estimator, presumably for a numerical stability. Coincidentally, $\varepsilon = 10^{-10}$ is used. The resulting scaling factor becomes:

$$s_i = \frac{N(1-(w_b)_i^2+\varepsilon)}{\tau(1-\tanh(\eta_i)^2+\varepsilon)}, \tag{103}$$

where N is the size of the complete training set.

Proposition F.1. With the setting of the hyper-parameters for τ, ε (Meng et al., 2020, Table 7) in large-scale experiments (MNIST, CIFAR10, CIFAR100), the BayesBiNN algorithm is equivalent to SGD with deterministic identity straight-through and latent weight decay.

Proof. For simplicity we will assume that η and g as scalars. For the general vector case the arguments would apply coordinate-wise.

First, we analyze the nominator of (103). From the asymptotic expansion of

$$1 - \tanh[\log(x)]^2 = \frac{4}{x^2} + O(\frac{1}{x^3}), \text{for } x \to \infty, \tag{104}$$

substituting $\log(x) = \frac{|\eta - z|}{\tau}$, we obtain

$$1 - w_b^2 \sim 4 \exp\big(-2\frac{|\eta-z|}{\tau}\big). \tag{105}$$

For example, for $\eta = 5$ and $z = 0$ we have that $(1 - w_b^2) \approx 4e^{-10^{11}}$. Therefore in the nominator the part $(1 - w_b^2)$ is negligibly small compared to ε and even to the floating point precision. This applies so long as $|\eta - z| \gg \tau = 10^{-10}$, which we expect to hold with high probability for two reasons: 1) η will be shown to grow significantly during the first iterates and 2) the probability of the noise matching η to this accuracy even for $\eta = 0$ is of the order $O(\tau)$.

The denominator of (103) satisfies the bounds

$$\tau\varepsilon \leq \tau(1 - \tanh(\eta)^2 + \varepsilon) \leq \tau(1 + \varepsilon), \tag{106}$$

from which we can conclude that $s \geq \frac{N\varepsilon}{\tau} = N$. However for a moderately large η the denominator drops quickly, *e.g.* for $\eta = 5$, we have $\tau(1 - \tanh(\eta)^2 + \varepsilon) < \tau(2 \cdot 10^{-4} + \varepsilon) \approx 2 \cdot 10^{-4}\tau$. And the asymptote for $|\eta| \to \infty$ is $\tau(4e^{-2|\eta|} + \varepsilon)$.

Since η is initialized uniformly in $[-10, 10]$ and receives updates of order at least $\alpha N g \approx 5g$ (for the initial $\alpha = 10^{-4}$ used), during the first iterates $|\eta|$ can be expected to grow significantly until we reach the asymptote $e^{-2|\eta|} \ll \varepsilon$, which is when $|\eta| > 5 \log 10 \approx 11$. After reaching this asymptote, we will have $s \approx \frac{N\varepsilon}{\tau\varepsilon} = \frac{N}{\tau}$ and we may expect the growth of η to stabilize around $|\eta| \approx \alpha \frac{N}{\tau}|g| \approx 10^{10}$.

The first consequence of this is that the scaling factor that was supposed to implement Gumbel-Softmax gradient, just becomes the constant $\frac{N}{\tau}$. In particular an inadvertent factor $\frac{1}{\tau}$ occurs. It is tempting to conclude that this up-scaling of the gradient corresponds to solving the variational Bayesian problem with the data evidence part up-scaled respectively, *i.e.*, completely dominating the the KL prior part, but this is not exactly so because also the true stochastic scaling of the gradient is modified.

The second consequence is that the natural parameters η have huge magnitudes during the training, and we have that $|z| \ll |\eta|$ with high probability, therefore the noise plays practically no role even in the forward pass of BayesBiNN. In this mode the BayesBiNN algorithm becomes equivalent to

$$w_b := \text{sign}(\eta); \tag{107a}$$

$$g := \nabla_{w_b}\mathcal{L}; \tag{107b}$$

$$\eta := (1 - \alpha)\eta - \alpha\frac{N}{\tau}g. \tag{107c}$$

It is seen that the forward pass and the gradient implement the deterministic straight-through with identity derivative and that the update has a form of SGD with a latent weight decay and with the gradient of data evidence up-scaled by $\frac{1}{\tau}$. These huge step-sizes do not destroy the learning since sign is invariant to a global rescaling of η. □

Proposition F.2. The result of the algorithm (107) does not depend on the values of τ and N.

Proof. Denoting $\bar{\boldsymbol{\eta}} = \frac{\tau}{N}\boldsymbol{\eta}$, we can equivalently rewrite (107) as

$$\boldsymbol{w}_b := \operatorname{sign}(\bar{\boldsymbol{\eta}}); \tag{108a}$$

$$\boldsymbol{g} := \nabla_{\boldsymbol{w}_b}\mathcal{L}; \tag{108b}$$

$$\bar{\boldsymbol{\eta}} := (1-\alpha)\bar{\boldsymbol{\eta}} - \alpha\boldsymbol{g}. \tag{108c}$$

This algorithm and the resulting binary weights $\boldsymbol{w}_b$ do not depend on τ, N. □

This is perhaps somewhat unexpected, but it makes sense indeed. The initial BayesBiNN algorithm of course depends on N and τ. However due to the issue with the implementation of Gumbel Softmax estimator for sufficiently small value of τ it falls into a regime which is significantly different from the Bayesian learning rule and is instead more accurately described by (107). In this regime, it produces the result not dependent on the particular values of τ and N. While we do not know what problem it is solving in the end, it is certainly not solving the variational Bayesian learning problem. This is so because the variational Bayesian learning problem and its solution do depend on N in a critical way. The algorithm (108a) indeed does not solve any variational problem as there is no variational distribution involved (nothing sampled). Yet the decay term $-\alpha\bar{\boldsymbol{\eta}}$ stays effective: if the data gradient becomes small, the decay term implements some small "forgetting" of the learned information and may be responsible for an improved generalization observed in the experiments (Meng et al., 2020). However there are other differences to the baselines, *e.g.* SGD vs. Adam that might be also contributing to the generalization difference.