A. Balancing Locally-Informed Proposals

As discussed in Section 2.2, locally informed proposals need to balance the likelihood increase with the reverse proposal probability. In particular, consider proposals of the form:

$$q_\tau(x'|x) \propto e^{\frac{1}{\tau}(f(x')-f(x))}\mathbf{1}(x' \in H(x)). \tag{15}$$

where $H(x)$ is the Hamming ball of some size around x and $\tau > 0$ is a temperature parameter. Here we provide a derivation of the fact that $\tau = 2$ balances these two terms.

When we examine the acceptance rate (Equation 1) of this proposal we find

$$\begin{aligned}
&\exp(f(x') - f(x))\frac{q_\tau(x|x')}{q_\tau(x'|x)} \\
&\quad = \exp(f(x') - f(x))\frac{\exp(\frac{1}{\tau}(f(x) - f(x'))Z(x)}{\exp(\frac{1}{\tau}(f(x') - f(x))Z(x')} \\
&\quad = \exp\left(\left(1 - \frac{2}{\tau}\right)(f(x') - f(x))\right)\frac{Z(x)}{Z(x')} \qquad (16)
\end{aligned}$$

where $Z(x) = \sum_{x' \in H(x)} \exp(\frac{1}{\tau}(f(x') - f(x))$ is the normalizing constant of the proposal. By setting $\tau = 2$, the most variable terms cancel and we are left with $\frac{Z(x)}{Z(x')}$. Thus, the acceptance rate is equal to the ratio of the normalizing constants of the proposal distributions. If the Hamming ball is small and the function f is well behaved (i.e Lipschitz) then, since x' is near x, $Z(x')$ will be near $Z(x)$ and the acceptance rate will be high.

B. Proof of Theorem 1

Our proof follows from Theorem 2 of Zanella (2020) which states that for two p-reversible Markov transition kernels $Q_1(x', x)$ and $Q_2(x', x)$, if there exists $c > 0$ for all $x' \neq x$ such that $Q_1(x', x) > c \cdot Q_2(x', x)$ then

(a) $\text{var}_p(h, Q_1) \leq \frac{\text{var}_p(h,Q_1)}{c} + \frac{1-c}{c} \cdot \text{var}_p(h)$

(b) $\text{Gap}(Q_1) \geq c \cdot \text{Gap}(Q_2)$

where $\text{var}_p(h, Q)$ is the asymptotic variance defined in Equation 7, $\text{Gap}(Q)$ is the spectral gap defined in Equation 8, and $\text{var}_p(h)$ is the standard variance $E_p[h(x)^2] - E_p[h(x)]^2$.

Our proof proceeds by showing we can bound $Q^\nabla(x', x) \geq c \cdot Q(x', x)$, and the results of the theorem then follow directly from Theorem 2 of Zanella (2020).

B.1. Definitions

We begin by writing down the proposal distribution of interest and their corresponding Markov transition kernels. For ease of notion we define some values

$$\begin{aligned}
\Delta(x', x) &\coloneqq f(x') - f(x) \\
\nabla(x', x) &\coloneqq \nabla_x f(x)^T(x' - x) \\
D_H &\coloneqq \sup_{x' \in H(x)} ||x' - x||
\end{aligned}$$

We restate the target proposal for $x' \in H(x)$

$$q_2(x'|x) = \frac{\exp\left(\frac{\Delta(x',x)}{2}\right)}{Z(x)}$$

where we have defined

$$Z(x) = \sum_{x' \in H(x)} \exp\left(\frac{\Delta(x', x)}{2}\right).$$

This proposal leads to the Markov transition kernel

$$\begin{aligned}
Q(x', x) &= q_2(x'|x) \min\left\{1, \frac{Z(x)}{Z(x')}\right\} \\
&= \min\left\{\frac{\exp\left(\frac{\Delta(x',x)}{2}\right)}{Z(x)}, \frac{\exp\left(\frac{\Delta(x',x)}{2}\right)}{Z(x')}\right\}.
\end{aligned}$$

We now restate our approximate proposal for $x' \in H(x)$

$$q^\nabla(x'|x) = \frac{\exp\left(\frac{\nabla(x',x)}{2}\right)}{\tilde{Z}(x)}$$

where we have defined

$$\tilde{Z}(x) = \sum_{x' \in H(x)} \exp\left(\frac{\nabla(x', x)}{2}\right)$$

which leads to the Markov transition kernel

$$\begin{aligned}
&Q^\nabla(x', x) = \\
&\quad q^\nabla(x'|x) \min\left\{1, \frac{\exp\left(\Delta(x', x)\right)}{\exp\left(\frac{\nabla(x',x) - \nabla(x,x')}{2}\right)}\frac{\tilde{Z}(x)}{\tilde{Z}(x')}\right\} \\
&\quad = \min\left\{\frac{\exp\left(\frac{\nabla(x',x)}{2}\right)}{\tilde{Z}(x)}, \frac{\exp\left(\Delta(x', x) + \frac{\nabla(x,x')}{2}\right)}{\tilde{Z}(x')}\right\}.
\end{aligned}$$

B.2. Preliminaries

It can be seen that $\nabla(x', x)$ is a first order Taylor-series approximation to $\Delta(x', x)$ and it follows directly from the Lipschitz continuity of $\nabla_x f(x)$ that

$$|\nabla(x', x) - \Delta(x', x)| \leq \frac{L}{2}||x' - x'||^2 \tag{17}$$

and since we restrict $x' \in H(x)$ we have

$$-\frac{L}{2}D_H^2 \leq \nabla(x', x) - \Delta(x', x) \leq \frac{L}{2}D_H^2 \tag{18}$$

B.3. Normalizing Constant Bounds

We derive upper- and lower-bounds on $\tilde{Z}(x)$ in terms of $Z(x)$.

$$\begin{aligned}
\tilde{Z}(x) &= \sum_{x' \in H(x)} \exp\left(\frac{\nabla(x', x)}{2}\right) \\
&= \sum_{x' \in H(x)} \exp\left(\frac{\Delta(x', x)}{2}\right) \exp\left(\frac{\nabla(x', x) - \Delta(x', x)}{2}\right) \\
&\leq \sum_{x' \in H(x)} \exp\left(\frac{\Delta(x', x)}{2}\right) \exp\left(\frac{LD_H^2}{4}\right) \\
&\leq \exp\left(\frac{LD_H^2}{4}\right) \sum_{x' \in H(x)} \exp\left(\frac{\Delta(x', x)}{2}\right) \\
&= \exp\left(\frac{LD_H^2}{4}\right) Z(x) \qquad (19)
\end{aligned}$$

Following the same argument we can show

$$\tilde{Z}(x) \geq \exp\left(\frac{-LD_H^2}{4}\right) Z(x). \qquad (20)$$

B.4. Inequalities of Minimums

We show $Q^\nabla(x', x) \geq c \cdot Q(x', x)$ for $c = \exp\left(\frac{-LD_H^2}{2}\right)$. Since both

$$Q(x', x) = \min\{a, b\}$$

and

$$Q^\nabla(x', x) = \min\{a^\nabla, b^\nabla\}$$

it is sufficient to show $a^\nabla \geq c \cdot a$ and $b^\nabla \geq c \cdot b$ to prove the desired result.

We begin with the a terms

$$\begin{aligned}
\frac{a^\nabla}{a} &= \frac{\exp\left(\frac{\nabla(x', x)}{2}\right)}{\tilde{Z}(x)} \frac{Z(x)}{\exp\left(\frac{\Delta(x', x)}{2}\right)} \\
&= \frac{Z(x)}{\tilde{Z}(x)} \exp\left(\frac{\nabla(x', x)}{2} - \frac{\Delta(x', x)}{2}\right) \\
&\geq \exp\left(\frac{-LD_H^2}{4}\right) \exp\left(\frac{\nabla(x', x)}{2} - \frac{\Delta(x', x)}{2}\right) \\
&\geq \exp\left(\frac{-LD_H^2}{4}\right) \exp\left(\frac{-LD_H^2}{4}\right) \\
&= \exp\left(\frac{-LD_H^2}{2}\right) \qquad (21)
\end{aligned}$$

Now the b terms

$$\begin{aligned}
\frac{b^\nabla}{b} &= \frac{\exp\left(\Delta(x', x) + \frac{\nabla(x, x')}{2}\right)}{\tilde{Z}(x')} \frac{Z(x')}{\exp\left(\frac{\Delta(x', x)}{2}\right)} \\
&= \frac{Z(x')}{\tilde{Z}(x')} \frac{\exp\left(\Delta(x', x) + \frac{\nabla(x, x')}{2}\right)}{\exp\left(\frac{\Delta(x', x)}{2}\right)} \\
&= \frac{Z(x')}{\tilde{Z}(x')} \exp\left(\frac{\Delta(x', x)}{2} + \frac{\nabla(x, x')}{2}\right) \\
&\geq \exp\left(\frac{-LD_H^2}{4}\right) \exp\left(\frac{\Delta(x', x)}{2} + \frac{\nabla(x, x')}{2}\right) \\
&= \exp\left(\frac{-LD_H^2}{4}\right) \exp\left(\frac{\nabla(x, x')}{2} - \frac{\Delta(x, x')}{2}\right) \\
&\geq \exp\left(\frac{-LD_H^2}{4}\right) \exp\left(\frac{-LD_H^2}{4}\right) \\
&= \exp\left(\frac{-LD_H^2}{2}\right) \qquad (22)
\end{aligned}$$

B.5. Conclusions

We have shown that $a^\nabla \geq \exp\left(\frac{-LD_H^2}{2}\right) a$ and $b^\nabla \geq \exp\left(\frac{-LD_H^2}{2}\right) b$ and therefore it holds that

$$Q^\nabla(x', x) \geq \exp\left(\frac{-LD_H^2}{2}\right) Q(x', x) \qquad (23)$$

From this, the main result follows directly from Theorem 2 of Zanella (2020).

C. Relationship to Relaxations

Han et al. (2020) show that sampling from any discrete distribution can be transformed into sampling from a continuous distribution with a piece-wise density function. For simplicity we focus on a distribution $p(x)$ over binary data $x \in \{0, 1\}^D$. To do this we will create a D-dimensional continuous distribution $p_c(z)$ where $z \in R^D$. We must specify a base distribution $p_0(z)$ which we choose to be $\mathcal{N}(0, I)$. We must then specify a function $\Gamma(z) : R^D \to \{0, 1\}^D$ which maps regions of equal mass under the base distribution to values in $\{0, 1\}^D$. A natural choice is $\Gamma(z) = \text{sign}(z)$

We then define $p_c(z)$ as

$$p_c(z) = \mathcal{N}(z; 0, I) p(\Gamma(z))$$

and it can be easily verified that generating

$$z \sim p_c(z), \qquad x = \Gamma(z)$$

will produce a sample from $p(x)$.

Thus, we have transformed a discrete sampling task into a task of sampling from a continuous distribution with a piece-wise density. Han et al. (2020) further relax this by defining

$$p_c^\lambda(x) = \mathcal{N}(z; 0, I)p(\Gamma_\lambda(z))$$

where $\Gamma_\lambda(x)$ is a continuous approximation to $\Gamma(x)$. A natural choice for $\text{sign}(x)$ is a tempered sigmoid function

$$\Gamma_\lambda(x) = \frac{1}{1 + e^{-x/\lambda}}$$

with temperature λ which controls the smoothness of the relaxation. This is similar to the Concrete relaxation (Maddison et al., 2016) for binary variables.

D-SVGD proposes to use the gradients of $\log p_c^\lambda(x)$ to produce updates for their continuous samples which are adjusted using an importance-weighted correction as proposed in Han & Liu (2018).

We can apply this same approach to other sampling methods such as MALA and HMC.

C.1. Relaxed MCMC

Gradient-based MCMC samplers such as MALA or HMC consist of a proposal distribution, $q(x'|x)$, and a metropolis accept/reject step. As a baseline of comparison, we present two straight-forward applications of the above relaxation to sampling from discrete distributions. In both settings we will use the continuous, differentiable surrogate $p_c^\lambda(x)$ to generate a proposal and we will perform our Metropolis step using the piece-wise target $p_c(x)$.

Relaxed MALA (R-MALA): Given a sample x, we sample a proposed update x' from:

$$q(x'|x) = \mathcal{N}\left(x'; x + \frac{\epsilon}{2}\nabla_x \log p_c^\lambda(x), \epsilon^2\right) \tag{24}$$

and we accept this proposal with probability

$$\min\left\{\frac{p_c(x')q(x|x')}{p_c(x)q(x'|x)}, 1\right\}. \tag{25}$$

R-MALA has two parameters; the stepsize ϵ and the temperature of the relaxation λ. We search over $\epsilon \in \{.1, .01, .001\}$ and $\lambda \in \{2., 1., .5\}$

Relaxed HMC (R-HMC) works similarly to R-MALA. Given a sample x we sample an initial momentum vector $v \sim \mathcal{N}(0, M)$ where M is the mass matrix. We perform k steps of leapfrog integration on the relaxed Hamiltonian

$$H^\lambda(x, v) = -\log p_c^\lambda(x) + \frac{1}{2}v^T M v$$

with step-size ϵ.

The proposal x', v' is accepted with probability

$$\min\left\{\frac{H(x, v)}{H(x', v')}, 1\right\} \tag{26}$$

where H is the target Hamiltonian

$$H(x, v) = -\log p_c(x) + \frac{1}{2}v^T M v.$$

We fix the mass matrix $M = I$ and the number of steps $k = 5$. This leaves two parameters, the ste-psize ϵ and temperature λ. As with R-MALA we search over $\epsilon \in \{.1, .01, .001\}$ and $\lambda \in \{.5, 1, 2.\}$.

C.2. Experimental Details

We compare D-SVGD, R-MALA, and R-HMC to Gibbs-With-Gradients at the task of sampling from RBMs. We present results in two settings; random RBMs with increasing dimension, and an RBM trained on MNIST using Contrastive Divergence.

The random RBMs have visible dimension $[25, 50, 100, 250, 500, 1000]$ and all have 100 hidden units. The MNIST RBM has 784 visible units and 500 hidden units and is trained as in Appendix E.1. Following Han et al. (2020) the random RBMs are initialized as

$$W \sim N(0, .05I), \qquad b, c \sim N(0, I).$$

All samples are initialized to a random uniform Bernoulli distribution and all samplers are run for 2000 steps. We evaluate by computing the Kernelized MMD between each sampler's set of samples and a set of approximate "ground-truth" samples generated with the RBMs efficient block-Gibbs sampler. We generate 500 ground truth samples and 100 samples for each sampler tested. In Figure 2 we plot the final log-MMD with standard-error over 5 runs with different random seeds. Samples on the right of the figure are generated in the same way from the MNIST RBM.

For D-SVGD we search over relaxation temperatures $\lambda \in \{.5, 1., 2.\}$. We optimize the samples with the Adam optimizer (Kingma & Ba, 2014). We search over learning rates in $\{.01, .001, .0001\}$. We use an RBF kernel $k(x, x') = \exp\left(\frac{-||x-x'||^2}{h}\right)$ and $h = \text{med}^2/(2\log(n+1))$ where med is the median pairwise distance between the current set of n samples.

All reported results for D-SVGD, R-MALA, and R-HMC are the *best* results obtained over all tested hyper-parameters. We found all of these methods to be very sensitive to their hyper-parameters – in particular, the relaxation temperature λ. We believe it may be possible to improve the performance of these methods through further tuning of these parameters but we found doing so beyond the scope of this comparison.

D. Gibbs-With-Gradients Extensions

D.1. Extensions To Larger Windows

We can easily extend our approach to proposals with larger window sizes. This would amount to a a Taylor-series approximation to likelihood ratios where more than 1 dimension of the data has been perturbed. These would come in the form of linear functions of $f(x)$ and $\nabla_x f(x)$. It is likely, of course, that as the window-size is increased, the accuracy of our approximation will decrease as will the quality of the sampler.

In all of our experiments, we found a window-size of 1 to give a considerable improvement over various baselines so we did not explore further. We believe this is an exciting avenue for future work.

D.2. Multi-Sample Variant

As mentioned, all experiments presented in the main paper use a window size of 1 meaning only 1 dimension can be changed at a time. In the binary case, we sample a dimension $i \sim q(i|x)$ which tells us which dimension to flip to generate our proposed update. A simple, and effective extension to this is to simply re-sample multiple indices from this same distribution

$$i_1, \ldots, i_N \sim q(i|x)$$

where N is the number of draws. We then generate x' by flipping the bit at each sampled index i_n. This changes the acceptance probability to

$$\min\left\{\exp(f(x') - f(x))\frac{\prod_{n=1}^N q(i_n|x')}{\prod_{n=1}^N q(i_n|x)}, 1\right\}. \quad (27)$$

This proposal makes a larger number of approximations and assumptions but we find that in some settings it can provide faster convergence and can have reasonable acceptance rates. We demonstrate this in our RBM experiments in Figure 9. We replicate Figure 3 but add the multi-sample variant described above with $N = 3$ and $N = 5$ samples. We find in this case the multi-sample variant has faster convergence and greater ESS.

E. Restricted Boltzmann Machines

Restricted Boltzmann Machines define a distribution over binary data x and latent variables h. The model is defined as:

$$\log p(x, h) = h^T W x + b^T x + c^T h - \log Z \quad (28)$$

where Z is the normalizing constant and $\{W, b, c\}$ are the model's parameters. In this model we can efficiently

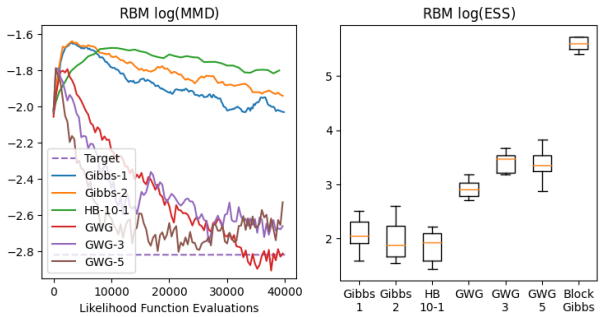

Figure 9. RBM Sampling with Gibbs-With-Gradients extensions. GWG-3 and GWG-5 are the multi-sample variant of GWG described above with $n = 3$ and $n = 5$, respectively.

marginalize out the latent variable h to obtain:

$$\begin{aligned}\log p(x) &= \log \sum_h p(x, h) \\ &= \log \sum_h \exp(h^T W x + b^T x + c^T h) \\ &= \log(1 + \exp(Wx + c)) + b^T x - \log Z \quad (29)\end{aligned}$$

While the joint and marginal are both unnormalized, we can see the conditional distirbutions can be easily normalized and take the form:

$$\begin{aligned}p(x|h) &= \text{Bernoulli}(Wx + c) \\ p(h|x) &= \text{Bernoulli}(W^T h + b).\end{aligned}$$

We can exploit this structure to more efficiently sample from RBMs. We can perform Block-Gibbs updates by starting at some initial x, and repeatedly sample $h \sim p(h|x)$, $x \sim p(x|h)$. Exploiting this structure leads to much more efficient sampling than standard Gibbs and other samplers (see Figure 3).

E.1. Experimental Details

We explore the performance of various approaches to sample from an RBM trained on the MNIST dataset. The RBM has 500 hidden units (and 784 visible units). We train the model using contrastive divergence (Hinton, 2002) for 1 epoch through the dataset using a batch size of 100. We use 10 steps of MCMC sampling using the Block-Gibbs sampler defined above to generate samples for each training iteration. We use the Adam (Kingma & Ba, 2014) optimizer with a learning rate of .001.

Our first result compares samples generated by various approaches with samples generated with the Block-Gibbs sampler described above. We generate a set of 500 samples using the Block-Gibbs sampler run for 10,000 steps.

At this length, the samples are very close to true samples from the model. Next we generate a set of 100 samples from a number of other samplers: Gibbs, Hamming Ball and Gibbs-With-Gradients. After every MCMC transition we compute the Kernelized Maximum Mean Discrepancy (Gretton et al., 2012) between the current set of samples and our "ground-truth" long-run Block-Gibbs samples. We use an exponential average Hamming kernel $K(x, x') = \exp\left(-\sum_{i=1}^{D} \frac{\mathbf{1}(x_i = x_i')}{D}\right)$ to compute the MMD.

The next result is the effective sample size of a test statistic for each sampler. Following Zanella (2020), our test statistic is the Hamming distance between the current sample and a random input configuration. We present a box-plot showing the median, standard-deviation, and outliers over 32 chains.

E.2. Additional Results

We visualize the samples after 10,000 steps of each tested sampler in Figure 10. We can see the Gibbs-With-Gradients samples much more closely matches the Block-Gibbs samples. This result is reflected in the improved MMD scores see in Figure 3 (left).

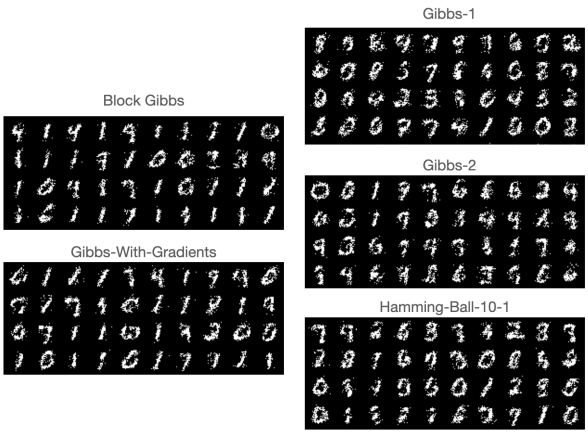

Figure 10. Sets of samples drawn from a fixed RBM with various MCMC approaches after 10,000 steps.

F. Ising Models

Ising models are unnormalized models for binary data defined as

$$\log p(x) = x^T J x + b^T x - \log Z \tag{30}$$

where J and b are the model's parameters and Z is the normalizing constant. J determines which other variables each x_i is correlated with. If $J = 0$ then the model becomes a factorized Bernoulli distribution. If all of the non-zero indices of J are the same, then we can pull out this value and rewrite the model as

$$\log p(x) = \theta x^T J x + b^T x - \log Z \tag{31}$$

where now θ controls how correlated each x_i is with its connected variables and J controls which variables each x_i is connected to. Our lattice Ising models take this form where the J is the adjacency matrix of a cyclic 2D lattice and θ controls the strength of the connectivity.

F.1. Experimental Details: Sampling

We experiment with our sampler's ability to sample from Ising models on the 2D cyclic lattice as various factors chage. These include the connectivity strength and the size of the lattice. We run each sampler for 100,000 steps and evaluate using the ESS of a test statistic. Following Zanella (2020) our test statistic is the Hamming distance between the current sample and a random input configuration. We present the ESS (in log-scale), averaged with standard-errors, over 32 random seeds.

We can see in both 10x10 and 40x40 lattice sizes, our sampler outperforms Gibbs and the Hamming ball.

F.2. Experimental Details: Training

We create Ising models with 2 kinds of graph structure; a cyclic 2D lattice and a random Erdos-Renyi (ER) graph. For the lattice we create models with a 10x10, 25x25, and 40x40 lattice leading to 100, 625, and 1600 dimensional distributions. We train models with connectivity $\theta \in [-.1, 0.0, .1, .25, .5]$.

For the ER graph, we create a model with 200 nodes. The ER edge probability is chosen so each node has an average of 4 neighbors. The strength of each edge is IID sampled from $N\left(0, \frac{1}{4}\right)$.

A dataset of 2000 examples is generated from each model using 1,000,000 steps of Gibbs sampling. We train models using persistent contrastive divergence (Tieleman, 2008) with a buffer size of 256 examples. Models are trained with the Adam optimizer (Kingma & Ba, 2014) using a learning rate of .001 and a batch size of 256. We update the persistent samples using Gibbs and Gibbs-With-Gradients. We train models with $\{5, 10, 25, 50, 100\}$ steps of MCMC per training iteration and compare their results. We train all models with an $\ell 1$ penalty to encourage sparsity with strength .01.

We compare results using the root-mean-squared-error between the true connectivity matrix J and the inferred connectivity matrix $\hat{J}$.

F.3. Additional Results: Training Ising Models

In Figure 11, we present an expanded version of Figure 6 which presents additional results. In these additional experiments we find Gibbs-With-Gradients considerably outperforms training with Gibbs sampling.

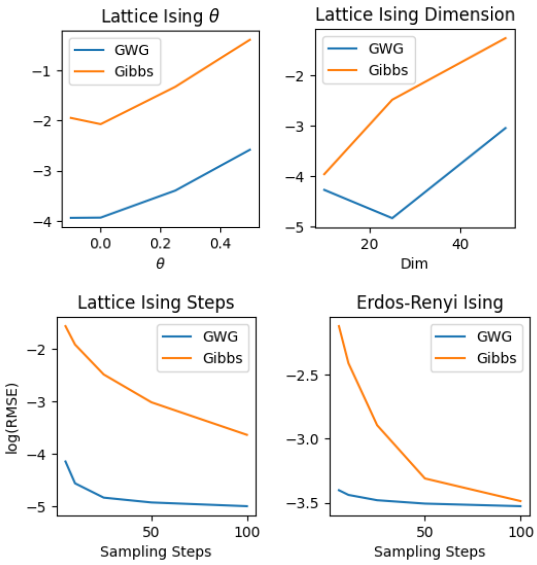

Figure 11. Training Ising models. Top Left: Lattice Ising with increasing θ (dim = 50, steps = 50). Top Right: Lattice Ising with increasing dimension ($\theta = .25$, steps = 25). Bottom Right: Lattice Ising with increasing steps (dim = 25, $\theta = .25$). Bottom Right: Erdos-Renyi Ising with increasing steps. Values are log(RMSE) between the learned J and the true J. Gibbs-With-Gradients leads to better solutions with lower computational cost.

G. Factorial Hidden Markov Models

Factorial Hidden Markov Models (FHMM) are a generalization of Hidden Markov Models and model real-valued time-series data. The observed data $y \in R^L$ is generated conditioned on a discrete latent-variable $x \in \{0,1\}^{L\times K}$. This latent-variable is drawn from the product of K independent Markov processes as seen below. The data y_t is generated by by the K-dimensional state vector x_t with Gaussian noise added.

$$
\begin{aligned}
p(x,y) &= p(y|x)p(x) \\
p(y|x) &= \prod_{t=1}^{L} \mathcal{N}(y_t; Wx_t + b, \sigma^2) \\
p(x) &= p(x_1)\prod_{t=2}^{L} p(x_t|x_{t-1}) \\
p(x_1) &= \prod_{k=1}^{K} \text{Bernoulli}(x_{1k}; \alpha_k) \\
p(x_{t+1}|x_t) &= \prod_{k=1}^{K} \text{Bernoulli}(x_{(t+1)k}; \beta_k^{x_{tk}}(1-\beta_k)^{1-x_{tk}})
\end{aligned}
\tag{32}
$$

The posterior $p(x|y)$ has no closed form and thus we must rely on MCMC techniques to sample from it.

G.1. Experimental Details

We sample the parameters of an FHMM randomly as

$$W, b \sim \mathcal{N}(0, I) \tag{33}$$

and set $\sigma^2 = .5$, $\alpha_k = .1$ and $\beta_k = .95$ for for all k.

We then sample $x \sim p(x)$ and $y \sim p(y|x)$ and run all samplers for 10,000 steps to generate samples from $p(x|y)$. The Hamming Ball Sampler (Titsias & Yau, 2017) is special for this model as it exploits the known block-structure of the posterior. We use a block size of 10 and the blocks are chosen to be all 10 dims of the latent state at a randomly chosen time x_t. Thus, this sampler is aware of more hard-coded structure in the model than the Gibbs baseline and Gibbs-With-Gradients.

H. Potts Models for Proteins

We train the MCMC models using PCD (Tieleman, 2008) with a buffer size of 2560. At each training iteration we sample a mini batch of 256 examples and 256 random samples from the persistent sample buffer. These are updated using 50 steps of either Gibbs or Gibbs-With-Gradients and the gradient estimator of Equation 12 is used to update the model parameters. We train for 10,000 iterations using the Adam optimizer (Kingma & Ba, 2014). Following Marks et al. (2011) we use block-$\ell 1$ regularization. This regularizer takes the form

$$\mathcal{L}_1(J) = \sum_{ij} ||J_{ij}||_2. \tag{34}$$

We add this regularizer to the maximum likelihood gradient estimator. We tested regularization strength parameters in

$\{.1, .03, .01\}$ and found $.01$ to work best for PLM, Gibbs, and Gibbs-With-Gradients.

Ground truth couplings were extracted from an experimentally validated distance-map. As is standard, we consider any pair of amino acids to be a contact if they are within 5 angstroms of each other.

H.1. Recall on PF10018

We do not present results on PF10018 in the main body as it was used to tune hyper-parameters. For completeness, we present them here in Figure 12. As with the other protiens, the MCMC-based training outperforms PLM but by a smaller margin and GWG and Gibbs perform comparably here. This further supports the benefit of MCMC training over PLM sets in on larger data as does the benefit of GWG over Gibbs.

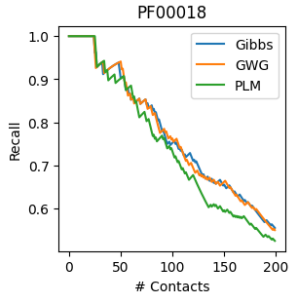

Figure 12. Recall Curves for contact prediction with Potts models.

H.2. Visualized Contacts

We visualize the inferred couplings for CADH1_HUMAN in Figure 13. We see that GWG most accurately matches the known structure with Gibbs inferring spurious couplings and PLM missing many couplings near the diagonal.

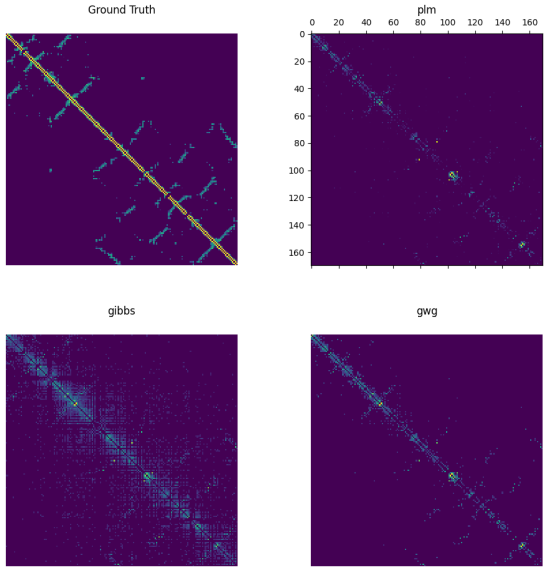

Figure 13. Inferred Couplings for CADH1_HUMAN. "Ground Truth" is the matrix of known distances between amino acids. All other matrices are the norms of the Potts model J_{ij} parameter.

I. Deep EBMs

I.1. Architectures

We train on two types of data; binary and categorical. For the binary datasets, the data is represented as $\{0, 1\}^D$ where D is the dimension of the data.

For the categorical datasets, each categorical variable is represented as a "one-hot" vector. Thus, for image data, each pixel is represented with a 256-dimensional vector. To deal with the very high dimensionality of this input parameterization, we first map each one-hot vector to a learned, low-dimensional embedding with a linear transformation. We map to $D_p = 4$ dimensions for all models tested. We then feed this $(D \times D_p)$-dimensional input to our network. There are certainly more efficient ways to represent this data, but our intention was not to achieve state-of-the-art results on these datasets and instead to demonstrate our sampler could enable the training of energy-based models on high-dimensional discrete data, so we use the most straightforward parameterization.

The ResNet used for our EBM is identical for all datasets. The network has 8 residual blocks with 64 feature maps. Each residual block has 2 convolutional layers. The first two residual blocks have a stride of 2. The output features are mean-pooled across the spatial dimensions and a single linear layer is used on top to provide the energy. The Swish (Ramachandran et al., 2017) nonlinearity $(x \cdot \sigma(x))$ is used throughout.

I.2. Experimental Details

We trained all models Adam (Kingma & Ba, 2014) using a learning rate of 0.0001. We linearly warm-up the learning rate for the first 10,000 iterations. We found this was necessary to help burn in the replay buffer of samples.

For the large datasets (static/dynamic MNIST, Omniglot) we use a replay buffer with 10,000 samples. For the smaller datasets (Caltech, Freyfaces, Histopathology) the buffer size is 1000. Unlike recent work on continuous EBMs (Du & Mordatch, 2019; Grathwohl et al., 2019), we do not reinitialize the buffer samples to noise. We found this resulted in unstable training and lower likelihoods.

We train all models for 50,000 iterations. We use the same

training/validation/testing splits as Tomczak & Welling (2018). We evaluate models every 5000 iterations using 10,000 steps of AIS. We select the model which performs the best on the validation data under this procedure. Final results in Table 2 are generated from the selected models by running 300,000 iterations of AIS. We evaluate using a model who's weights are given by an exponential moving average of the training model's weights. This is analogous to training with "fast-weights" as in Tieleman & Hinton (2009). We find this greatly improves likelihood performance and sample quality. We use an exponential moving average with weight 0.999 and did not experiment with other values.

We believe better results could be obtained with larger models or alternative architectures, but we leave this for future work.

I.2.1. Partition Function Estimation with AIS

We estimate likelihoods by estimating the partition function using Annealed Importance Sampling (AIS) (Neal, 2001). AIS underestimates the log-partition-function leading to *over-estimating* the likelihood. The estimation error can be reduced by using a larger number of intermediate distributions or a more efficient MCMC sampler. Results presented in Table 2 were generated with 300,000 intermediate distributions. We chose this number as it appears to be sufficiently high for our partition function estimate to converge. Despite this, these are upper-bounds and therefore should not be considered definitive proof that one model achieves higher likelihoods than another.

We anneal between our model's unnormalized log-probability $f(x)$ and a multivariate Bernoulli or Categorical distribution, $\log p_n(x)$, fit to the training data, for binary and categorical data, respectively.

$$f_t(x) = \beta_t f(x) + (1 - \beta_t) \log p_n(x) \tag{35}$$

where β_t is linearly annealed from 0 to 1. Alternative strategies such as sigmoid annealing could be used, but we leave this for future work.

In Figure 14 we plot the estimated likleihoods for our Caltech Silhouettes models as the number of intermediate distributions increases. It can be seen that between 30,000 and 300,000 ($\approx 10^{4.5} \rightarrow 10^{5.5}$) the values appear to be converged, thus we feel our reported number faithfully represent our models' performance.

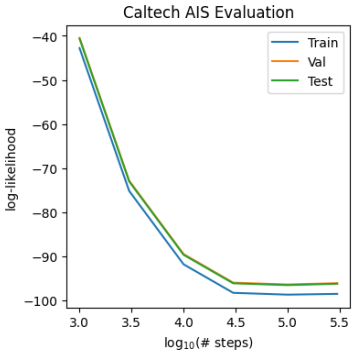

Figure 14. AIS likelihood estimates as the number of intermediate distributions increases for our Caltech Silhouettes Resnet EBM. Values converge after $30,000 \approx 10^{4.5}$ step s

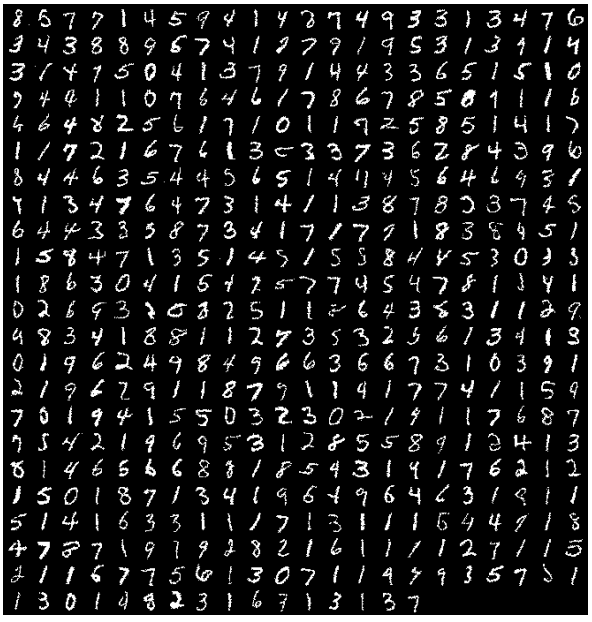

Figure 15. Static MNIST Samples

I.3. Additional Results

We present additional long-run samples from our convolutional EBM. These samples were generated using an annealed Markov Chain (as described above) and Gibbs-With-Gradients as the base MCMC sampler.

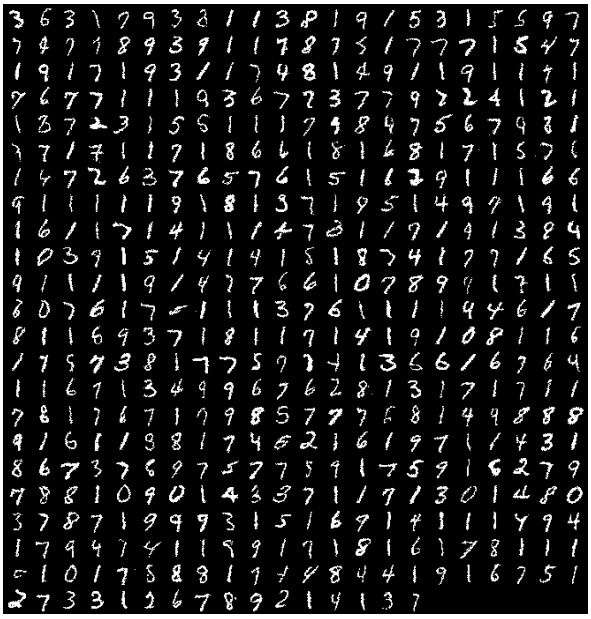

Figure 16. Dynamic MNIST Samples

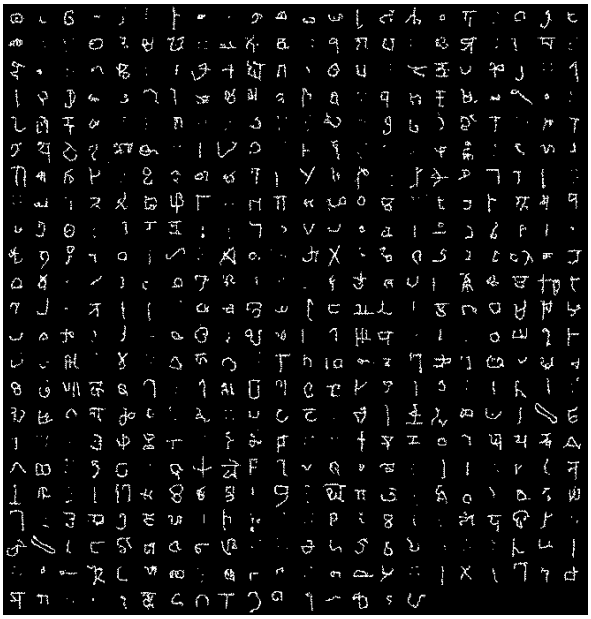

Figure 17. Omniglot Samples

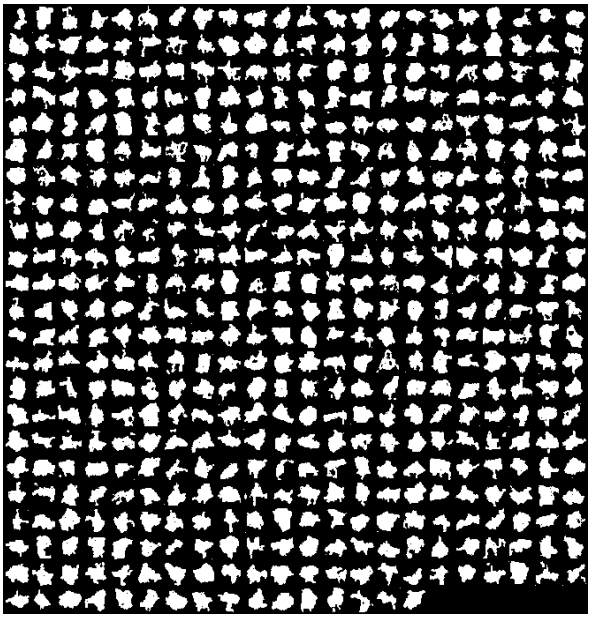

Figure 18. Caltech Silhouette Samples

Figure 19. Freyfaces Samples

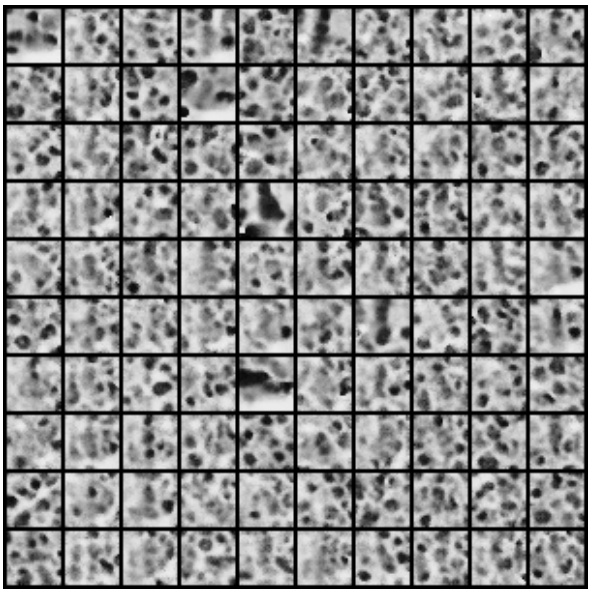

Figure 20. Histopathology Samples

J. Preliminary Text EBM Results

There has recently been interest in learning Energy-Based Models of text data. An EBM for text could enable non-autoregressive generation and more flexible conditioning than autoregressive models. For example, an EBM trained on language pairs $p(x, y)$ could be used to translate in either direction without retraining and could be structured as $\log p(x, y) = f_\theta(x)^T g_\phi(y) - \log Z$ so that each language component could be trained separately. We also have much more architectural freedom when specifying EBMs meaning f_θ is free to be a CNN, RNN, transformer, or MLP.

A few works have had success training and applying text EBMs. Deng et al. (2020) find that EBMs can be used to improve the generative modeling performance of large-scale transformer language models and He et al. (2021) find that Joint Energy-Based Models (Grathwohl et al., 2019) can improve the calibration of text classifiers. Because of the discrete structure of the text data, both of these works train using Noise Contrastive Estimation (Gutmann & Hyvärinen, 2010) using a pretrained autoregressive language model as the noise distribution. NCE requires a noise distribution which can be sampled from and enables exact likelihood computation. Thus, these approaches rely on and are limited by the quality of these autoregressive models.

Ideally, we could train a text EBM on its own, without an auxiliary model. One way to do this is to use the gradient estimator of Equation 12 but the MCMC sampling task is very challenging. Text models typically have a vocabulary above 10,000 words so the size of the discrete sample space is tremendous. Further, as noted in Section 8, to apply Gibbs sampling to a model like this we would need to evaluate the energy function over 10,000 times to perform a single step of sampling!

We believe Gibbs-With-Gradients can provide an avenue to train and sample from these kinds of models. As a preliminary experiment we train non-autoregressvive language models on a shortened version of the Penn Tree Bank dataset (Taylor et al., 2003). This is a dataset of short sentences with 10,000 words. We cut out all sentences with greater than 20 words and pad all shorter sentences with an "end of sequence" token. We feel this simplified setting is sufficient for a proof-of-concept as the configuration space is very large and Gibbs sampling is not feasible.

Our model consists of a bidirectional LSTM (Gers et al., 1999) with 512 hidden units. We project the 10,000 words to an embedding of size 256 with a learned mapping. To compute the energy, we take the last hidden-state from each direction, concatenate them together to a 1024-dimensional vector. We project this to 512 dimensions with a linear layer, apply a Swish nonlinearity and then map to 1 dimension with another linear layer.

We train using PCD with a buffer size of 1000 and we use 20 steps of MCMC with Gibbs-With-Gradients to update the samples at every training iteration. Besides this, training was identical to our image EBMs in section 8.

We compare with a simple autoregressive language model which is based on an LSTM with 512 hidden units and use a learned word embedding of size 256.

We find the autoregressive model slightly outperforms the EBM. The test-set log-likelihoods of the EBM and autoregressive model are -77.16 and -74.0, respectively. For comparison, a uniform distribution over possible tokens obtains -184.21 and a Categorical distribution fit to the training data obtains -100.05.

While we are aware these are *far* from state-of-the-art language modelling results, we believe they demonstrate that Gibbs-With-Gradients can enable MCMC-trained EBMs to successfully model text data with large vocabulary sizes. At every step, the sampler has $10,000 \times 20 = 200,000$ choices for possible updates. Despite this massive sampling space, we find our acceptance rates during training are just above 70% making our approach at least 3500 times more efficient than Gibbs sampling.

We believe improvements could be obtained through larger models and more tuning. To further scale this approach, we believe we will need to develop further approximations which make sampling from very large categorical distributions more efficient and numerically stable. We leave this for future work.