# Scaling RBMs to High Dimensional Data with Invertible Neural Networks

**Will Grathwohl** [*1 2 3] **Xuechen Li** [*3] **Kevin Swersky** [3] **Milad Hashemi** [3] **Jörn-Henrick Jacobsen** [1 2]
**Mohammad Norouzi** [3] **Geoffrey Hinton** [3]

## Abstract

We combine invertible neural networks with RBMs to create a more tractable energy-based model which retains the power of recent scalable EBM variants while allowing for more efficient sampling and likelihood evaluation. We further find that replacing the Gaussian base distributions typically used in normalizing flows with an RBM leads to improved likelihood compared to a flow with a similar architecture possibly providing a pathway to more efficient, but still tractable generative models. We demonstrate the performance of our approach on small image datasets and compare to recent normalizing flows and EBMs.

## 1. Introduction

Restricted Botzmann Machines (RBMs) have had a long and rich history in the generative modeling community (Smolensky et al., 1986; Hinton, 2002; Hinton & Salakhutdinov, 2006). As a generative model they have many desirable properties including compositional structure (Hinton, 2002; Du & Mordatch, 2019) and the ability to be trained on unlabeled data, or data with missing values. Although they are unnormalized Energy-Based models, the structure of RBMs emits a tractable blocked Gibbs sampler, which enables *relatively* fast sampling and training, compared to other classes of Energy-Based Models (EBMs). While standard RBMs have been successful at modeling simple distributions, to successfully model more complicated data such as images, RBMs typically need to be stacked on top of each other to create a Deep Belief Network (Hinton et al., 2006; Salakhutdinov & Hinton, 2009). This greatly increases the model's expressive power but sampling must now be done sequentially, reducing the efficiency and increasing the bias of the training objective.

[*]Equal contribution [1]Department of Computer Science, University of Toronto [2]Vector Institute [3]Google AI Research. Correspondence to: Will Grathwohl <wgrathwohl@cs.toronto.edu>, Xuechen Li <lxuechen@google.com>.

Second workshop on *Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models* (ICML 2020), Virtual Conference

In recent years alternative classes of generative models have become more popular such as Normalizing Flows (Rezende & Mohamed, 2015; Deco & Brauer, 1995) and Variational Autoencoders (Kingma & Welling, 2013; Rezende et al., 2014). These models allow for more efficient sampling and likelihood computation (or estimation) at the cost of expressiveness. Despite this, considerable progress has been made in these more tractable models causing RBMs to fall out of favor.

In this work we propose a simple method to increase the scalability of RBMs without having to rely on sequential sampling. We train an RBM on top of a learned embedding given by an invertible neural network similar to those used to define Normalizing Flows (Kingma & Dhariwal, 2018; Dinh et al., 2016). The entire model is trained end-to-end to approximately maximize likelihood. We find that our EBM-flow hybrid models (which we refer to as EB-and-Flow) achieve better likelihoods than normalizing flows and RBMs while being easier to sample from and evaluate than recent EBM approaches.

## 2. Background

### 2.1. Energy-Based Models

An Energy-Based Model (EBM) is a model which represents a probability distribution as

$$p(x) = \frac{e^{-E(x)}}{Z}, \tag{1}$$

where $E$ is known as the *energy function* which maps the data to a scalar value and $Z$ is the normalizing constant. The normalizing constant is implicitly defined by the energy function as $Z = \int e^{-E(x)} dx$ so it is not modeled. This makes training and sampling challenging but gives great flexibility to the model.

### 2.2. Restricted Boltzmann Machines

An RBM is an EBM which defines a distribution over visible units $v$ and hidden units $h$ defined as

$$p(v) = \sum_h p(v, h) = \sum_h \frac{e^{-E(v,h)}}{Z}. \tag{2}$$

The visible or hidden units can be discrete or continuous. In this work we focus on Gaussian-Bernoulli RBMs (Cho et al., 2013) which have continuous $v$ and discrete $h$. The energy function for this model is defined as

$$E(v,h) = \frac{||v-b_v||^2}{2\sigma_v^2} + h^\top b_h + h^\top W \frac{v}{\sigma_v}, \quad (3)$$

with parameters $\{W, b_v, \sigma_v, b_h\}$. While the joint distribution $p(v,h)$ is unnormalized, the conditional distributions are not

$$p(v|h) = \mathcal{N}(v|b_v + \sigma_v h W^\top, \sigma_v^2), \quad (4)$$

$$p(h|v) = \text{Bernoulli}\left(h|\text{sigmoid}\left(\frac{v}{\sigma_v}W + b_h\right)\right), \quad (5)$$

allowing for an efficient blocked Gibbs sampler to be used to draw samples from $p(v,h)$.

We can also analytically sum out the hidden variables to produce an EBM for the marginal distribution of visible units with energy:

$$E(v) = \frac{\|v-b_v\|^2}{2\sigma^2} - \text{softplus}\left(W^\top v/\sigma + b_h\right)^\top \mathbf{1}.$$

RBMs are trained with gradient decent by estimating

$$\nabla_\theta \log p(v) = -\nabla_\theta E(v) + \mathbb{E}_{p(v)}[\nabla_\theta E(v)], \quad (6)$$

where $\nabla_\theta E(v)$ can be easily computed. The samples in the expectation come from a Gibbs chain by repeated sampling from Equations 4 and 5. The chain can be seeded from data samples, giving the Contrastive Divergence (CD) algorithm (Hinton, 2002). Recent work has also proposed starting the chain from random noise (Nijkamp et al., 2019b). A persistent chain can be used (Tieleman, 2008; Du & Mordatch, 2019; Grathwohl et al., 2019) for a lower-bias estimate which we use in this work.

### 2.3. Normalizing Flows

A Normalizing Flow (NF) is a generative model for data which works by drawing a sample $z \sim p(z)$ where $p(z)$ is an easily sampled distribution with a closed form density, referred to as the base distribution. This sample $z$ is then passed through a function $f^{-1}$ to give us our data $x = f^{-1}(z)$. When $f$ is bijective, then we can compute $\log p(x)$ as

$$\log p(x) = \log p(f(x)) + \log \left|\det \frac{\partial f(x)}{\partial x}\right|. \quad (7)$$

Most progress in NF research focuses on designing maximally expressive invertible architectures with efficient Jacobian log determinant computation (Rezende & Mohamed, 2015; Dinh et al., 2014; 2016; Kingma & Dhariwal, 2018; Grathwohl et al., 2018; Behrmann et al., 2018; Chen et al., 2019).

### 2.4. Issues with EBMs

While EBMs have shown many advantages over tractable likelihood models and are becoming one of the premier approaches to generative modeling (Du & Mordatch, 2019; Grathwohl et al., 2019; Nijkamp et al., 2019b; Song & Ermon, 2019), they have limitations that make them challenging to work with. The energy-based parameterization is very flexible, but sampling and likelihood evaluation require Markov Chain Monte Carlo (MCMC) techniques. Given the unconstrained nature of the energy functions in these recent models, the gradient-based samplers typically used have difficulty mixing (Nijkamp et al., 2019a) making likelihood evaluation a futile task (Du & Mordatch, 2019). This is particularly problematic because samples and likelihood are the current standard for evaluating generative models, making it unclear how these recent EBMs compare with other classes of generative models. For this reason, it would be desirable to train a model which retains the flexibility of EBMs while enabling a tractable way to accomplish both of these tasks.

## 3. Related Work

**Base Distributions for Flows.** Typically a Gaussian base density is used for NF models which imposes topological constraints on the data distributions that can be modeled (Falorsi et al., 2018). Recently, some alternatives have been explored. Izmailov et al. (2019) train NF models with a Gaussian Mixture base distribution for semi-supervised learning. This leads to strong performance at this task, but the parameters of the base distribution could not be learned online with the flow model. Autoregressive base distributions have also been shown to improve sample quality as well as likelihoods (Mahajan et al., 2020) at the cost of slower sampling and added model complexity. On MNIST, our approach provides a larger benefit.

**Unstructured EBMs.** EBMs impose very few constraints on model architecture. Recently, energy functions based on unstructured neural networks have achieved impressive performance in terms of sample quality as well as downstream discriminative tasks (Du & Mordatch, 2019; Grathwohl et al., 2019; Song & Ou, 2018). Training these models requires sampling using MCMC from the model distribution, which is notoriously difficult for unstructured energy functions, leading to costly and unstable optimization. These difficulties can partially be side-stepped by training with alternative objectives such as Score Matching (Song & Ermon, 2019; Li et al., 2019) or objectives based on Stein Discrepancies (Grathwohl et al., 2020) – the latter also providing a compelling method for model evaluation. However, despite this progress, reliably training large-scale EBMs and evaluating their likelihoods is still an open problem.

## 4. The Best of Both Worlds

We define a new model for data $x$. We first sample $v$ from a Gaussian-Bernoulli RBM $p(v) = \sum_h p(v, h)$. We then pass $v$ through an invertible neural network with tractable Jacobian log-determinant, $f_\theta^{-1}$, defining a model:

$$p(x) = \sum_h p(x, h) = \sum_h p(v, h) \left| \det \frac{\partial f_\theta(x)}{\partial x} \right|, \quad (8)$$

where $v = f_\theta(x)$. Overall, this gives

$$\log p(x) = -\frac{\|v - b_v\|^2}{2\sigma^2} + \text{softplus}\left(W^\top v/\sigma + b_h\right)^\top \mathbf{1} + \log \left| \det \frac{\partial f_\theta(x)}{\partial x} \right| - \log Z.$$

With respect to $\theta$, we can easily optimize $\log p(x)$ since the gradients with respect to $\theta$ do not depend on $\log Z$. With respect to the RBM parameters $\phi = \{W, b_h, b_v\}$, we estimate $\nabla_\phi \log p(x)$ using Persistent Contrastive Divergence (PCD) (Hinton, 2002; Tieleman, 2008) using a replay buffer (Du & Mordatch, 2019; Grathwohl et al., 2019).

Each step of PCD learning requires only 2 matrix multiplies. Assuming $f_\theta$ is a large neural network, then PCD with 20 sampling steps per training training iteration adds negligible computational overhead compared to training a NF model with a Gaussian base distribution. Pseudocode for our training procedure can be found in Algorithm 1.

**Algorithm 1** EB-and-Flow Training

**Require:** Invertible net $f_\theta$, RBM $p_\phi(v, h)$, replay buffer $B$, number of MCMC steps $n$
  **for** $x$ in training data **do**
    Compute $v = f_\theta(x)$
    Compute $g = \nabla_v \log p_\phi(v)$
    Update $\theta$ with
      $\nabla_\theta \log p(v) = g^\top \nabla_\theta f_\theta(x) + \nabla_\theta \log |\nabla_x f_\theta(x)|$
    Sample $v' \sim B$, remove from $B$
    $\hat{v} =$ MCMC sample for $n$ steps from $v'$
    Update $\phi$ with $\nabla_\phi E(v) - E_{\hat{v}}[\nabla_\phi E(\hat{v})]$
    Add $\hat{v}$ to $B$
  **end for**

## 5. Model Structure

When training normalizing flows, the invertible model is a mapping $f : \mathbb{R}^d \to \mathbb{R}^d$. Traditionally, variables are "factored out" as transformations are added. This means we apply one invertible mapping $f_1$ to get $v_1' = f_1(x)$. We then split the features of $v_1'$ into two groups $v_1, v_1^+$, We then apply our second transformation to $v_1^+$ giving $v_2'$ which is split into $v_2, v_2^+$, and $v_2^+$ is passed to the next transformation, and so on. This gives us $L$ separate outputs $v_1, \ldots, v_L$ where $L$ is the number of times the variables are factored out.

When $D$ is small, we can simply concatenate all $v_i$ together into one vector $v$ and use an RBM on top to define $p(v, h)$. Alternatively we can give each $v_i$ its own independent RBM, which defines the overall product distribution $p(v_1, h_1) \cdots p(v_L, h_L)$. Under this model each $v_i$ is independent of all $v_{\setminus i}$.

If $v_i$ is spatially structured (as it would be using a convolutional model) we can define $p(v', h_i)$ with a convolutional RBM (Lee et al., 2009). We use traditional RBMs for our MNIST and Fashion MNIST experiments and use convolutional RBMs for our CIFAR10 experiments. Specifically, we use convolutional RBMs which share a spatially structured hidden state. Full details of this RBM architecture are found in Appendix B.

### 5.1. Conditional Versions

To incorporate side information, we may include additional visible units to the Gaussian-Bernoulli RBM component of the EB-and-Flow model following (Larochelle & Bengio, 2008). For instance, given an image $x$ and one-hot label $y$, the log-likelihood can be defined as

$$\log p(x, y) = -\frac{\|v - b_v\|^2}{2\sigma^2} + y^\top b_y + \text{softplus}\left(W^\top v/\sigma + V^\top y + b_h\right)^\top \mathbf{1} + \log \left| \det \frac{\partial f_\theta(x)}{\partial x} \right| - \log Z,$$

where $V$ is a weight matrix and $b_y$ is the bias for the labels. The contrastive divergence learning algorithm can still be used to approximately maximize this log-likelihood, since efficient blocked Gibbs sampling is available (Larochelle & Bengio, 2008).

Given both a labeled set $\mathcal{D}_l$ and unlabeled set $\mathcal{D}_u$, we may optimize the following joint objective

$$\sum_{x \in \mathcal{D}_u} \log \sum_y p(x, y) + \lambda \sum_{(x,y) \in \mathcal{D}_l} \log p(x, y). \quad (9)$$

We leave the investigation of this model as future work.

## 6. Experiments

We run a number of experiments to demonstrate the performance of our approach. We train EB-and-Flow models using the invertible network architectures from NICE (Dinh et al., 2014) and Glow (Kingma & Dhariwal, 2018). We first explore how our approach compares with other EBMs on sampling and likelihood evaluation. Next we explore likelihood computed on held-out test data and compare our

approach with standard RBMs and NF models. On each of these tasks we find our model performs favorably. Full details of model architectures, baselines, and hyperparameters can be found in Appendix A.

### 6.1. Likelihood Evaluation

We evaluate our models by finding an upper-bound on likelihood using Annealed Importance Sampling (AIS) (Neal, 2001) and a lower-bound using RAISE (Burda et al., 2015). These methods run MCMC chains which slowly anneal from a tractable distribution to our model's distribution. For recent EBM models, many chains must be run (over 300,000 was used in Du & Mordatch (2019) and still the bounds are very loose). We find that EB-and-Flow models are much easier to evaluate. As seen in Table 1, we can arrive at bounds within .001 bit/dim (bit/dim is typically reported up to 2 decimal places) using 1000 steps meaning that EB-and-Flow models can be reliably compared with tractable likelihood models like NFs and VAEs.

| Model | Low-Bound | Up-Bound | # Iterations |
|---|---|---|---|
| EBM | 3.92 | 4.45 | 300k |
| EB-and-Flow | 1.0060 | 1.0061 | 1k |

*Table 1.* Likelihood evaluation results on MNIST.

### 6.2. Sampling

We compare the ease of sampling from our model with a standard RBM trained on MNIST, FashionMNIST, and CIFAR10. As can be seen in Figure 1 we observe much faster mixing, higher quality, and more diverse samples. We can see our chain quickly mixes between the various modes of the data distribution producing a varied set of samples relatively quickly.

### 6.3. Likelihood Evaluation

We compare EB-and-Flow Models with standard NFs, RBMs, and Flows with autoregressive base distributions (Mahajan et al., 2020) on the MNIST, Fashion MNIST (Xiao et al., 2017), and CIFAR10 (Krizhevsky et al., 2009) datasets. As seen in Table 2, across various architectures on MNIST and Fashion MNIST we find that EB-and-Flow outperforms the baselines in terms of test set log-likelihood. We see that we are able to obtain competitive results on CIFAR10 but do not outperform the state-of-the-art or flows with standard Gaussian base distributions. We believe this has to do with difficulties training our latent RBM on higher-dimensional data. Izmailov et al. (2019) also experienced these difficulties leading them to fix the base distribution throughout training and refine it post-hoc. We expect this performance gap could be closed in this way or by using improved RBM training techniques (Qiu et al., 2020). We leave this for further work.

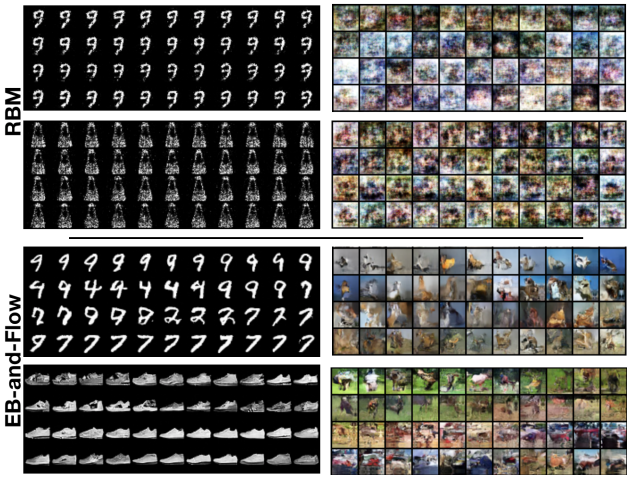


*Figure 1.* Consecutive Gibbs samples from a markov chain that has been burned in for 1000 iterations. Top: Standard RBM. Bottom: EB-and-Flow. Top left: MNIST. Bottom left: Fashion MNIST. Right: CIFAR10.

| | MNIST | Fashion | Cifar10 |
|---|---|---|---|
| NF | 1.87/1.05<br>NICE/Glow | 4.05/2.96<br>NICE/Glow | 3.35<br>Glow |
| EB-and-Flow | 1.80/1.01<br>NICE/Glow | 4.03/2.90<br>NICE/Glow | 3.36<br>Glow |
| Flow+AR | 1.03<br>Glow | N/A<br>Glow | 3.31<br>Glow |
| RBM | 2.11 | 4.55 | 5.47 |

*Table 2.* Unconditional Density Estimation. All CIFAR10 models use the Glow architecture.

## 7. Conclusion

In this work we have presented a new type of EBM that retains the flexibility and desirable properties of previous EBM approaches while addressing several key issues of these prior methods; namely sampling and evaluation. By leveraging invertible neural networks we have created a more tractable EBM which outperforms a flow-based baseline with the same architecture, indicating that the additional flexibility provided by the RBM improves modeling performance. We are excited about the potential of our approach to make invertible models more efficient by allowing them to use smaller networks and by the potential of scaling our approach up to tackle more challenging datasets.

## References


Behrmann, J., Grathwohl, W., Chen, R. T., Duvenaud, D., and Jacobsen, J.-H. Invertible residual networks. *arXiv preprint arXiv:1811.00995*, 2018.

Burda, Y., Grosse, R., and Salakhutdinov, R. Accurate and conservative estimates of mrf log-likelihood using reverse annealing. In *Artificial Intelligence and Statistics*, pp. 102–110, 2015.

Chen, T. Q., Behrmann, J., Duvenaud, D. K., and Jacobsen, J.-H. Residual flows for invertible generative modeling. In *Advances in Neural Information Processing Systems*, pp. 9913–9923, 2019.

Cho, K. H., Raiko, T., and Ilin, A. Gaussian-bernoulli deep boltzmann machine. In *The 2013 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–7. IEEE, 2013.

Deco, G. and Brauer, W. Nonlinear higher-order statistical decorrelation by volume-conserving neural architectures. *Neural Networks*, 8(4):525–535, 1995.

Dinh, L., Krueger, D., and Bengio, Y. NICE: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

Du, Y. and Mordatch, I. Implicit generation and generalization in energy-based models. *arXiv preprint arXiv:1903.08689*, 2019.

Falorsi, L., de Haan, P., Davidson, T. R., De Cao, N., Weiler, M., Forré, P., and Cohen, T. S. Explorations in homeomorphic variational auto-encoding. *arXiv preprint arXiv:1807.04689*, 2018.

Grathwohl, W., Chen, R. T., Bettencourt, J., Sutskever, I., and Duvenaud, D. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*, 2018.

Grathwohl, W., Wang, K.-C., Jacobsen, J.-H., Duvenaud, D., Norouzi, M., and Swersky, K. Your classifier is secretly an energy based model and you should treat it like one. *arXiv preprint arXiv:1912.03263*, 2019.

Grathwohl, W., Wang, K.-C., Jacobsen, J.-H., Duvenaud, D., and Zemel, R. Cutting out the middle-man: Training and evaluating energy-based models without sampling. *arXiv preprint arXiv:2002.05616*, 2020.

Hinton, G. E. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

Hinton, G. E. and Salakhutdinov, R. R. Reducing the dimensionality of data with neural networks. *science*, 313 (5786):504–507, 2006.

Hinton, G. E., Osindero, S., and Teh, Y.-W. A fast learning algorithm for deep belief nets. *Neural computation*, 18 (7):1527–1554, 2006.

Izmailov, P., Kirichenko, P., Finzi, M., and Wilson, A. G. Semi-supervised learning with normalizing flows. *arXiv preprint arXiv:1912.13025*, 2019.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kingma, D. P. and Dhariwal, P. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pp. 10215–10224, 2018.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Larochelle, H. and Bengio, Y. Classification using discriminative restricted boltzmann machines. In *Proceedings of the 25th international conference on Machine learning*, pp. 536–543, 2008.

Lee, H., Grosse, R., Ranganath, R., and Ng, A. Y. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th annual international conference on machine learning*, pp. 609–616, 2009.

Li, Z., Chen, Y., and Sommer, F. T. Annealed denoising score matching: Learning energy-based models in high-dimensional spaces. *arXiv preprint arXiv:1910.07762*, 2019.

Mahajan, S., Bhattacharyya, A., Fritz, M., Schiele, B., and Roth, S. Normalizing flows with multi-scale autoregressive priors. *arXiv preprint arXiv:2004.03891*, 2020.

Neal, R. M. Annealed importance sampling. *Statistics and computing*, 11(2):125–139, 2001.

Nijkamp, E., Hill, M., Han, T., Zhu, S.-C., and Wu, Y. N. On the anatomy of mcmc-based maximum likelihood learning of energy-based models. *arXiv preprint arXiv:1903.12370*, 2019a.

Nijkamp, E., Zhu, S.-C., and Wu, Y. N. On learning non-convergent short-run mcmc toward energy-based model. *arXiv preprint arXiv:1904.09770*, 2019b.

Qiu, Y., Zhang, L., and Wang, X. Unbiased contrastive divergence algorithm for training energy-based latent variable models. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=r1eyceSYPr.

Rezende, D. J. and Mohamed, S. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

Salakhutdinov, R. and Hinton, G. Deep boltzmann machines. In *Artificial intelligence and statistics*, pp. 448–455, 2009.

Smolensky, P., McClelland, J., and Rumelhart, D. *Parallel distributed processing: Explorations in the microstructure of cognition*. MIT Press, 1986.

Song, Y. and Ermon, S. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems*, pp. 11895–11907, 2019.

Song, Y. and Ou, Z. Learning neural random fields with inclusive auxiliary generators. *arXiv preprint arXiv:1806.00271*, 2018.

Tieleman, T. Training restricted boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th international conference on Machine learning*, pp. 1064–1071, 2008.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

# A. Experimental Details

## A.1. Architectures

The NICE architectures used were exactly as in Dinh et al. (2014). For MNIST and Fashion MNIST the Glow models we used have two levels of features. Each level consists of 8 affine-coupling blocks with 1x1 convolutions and the hidden dimension of the couple blocks was 512. For CIFAR10 we use 3 levels of features and 16 blocks per level. Our CIFAR10 models used convolutional RBMs with 100 hidden channels. We detail this RBM structure in Appendix B.

For the MNIST and Fashion MNIST we use a single fully-connected RBM whose input is a concatenation of the features from the invertible network. These RBMs had 512 hidden units. The baseline RBMs also have 512 hidden units.

## A.2. Training

MNIST and Fashion MNIST models and baselines were trained for 250 epochs using a batch size of 128. CIFAR10 models were trained for 500 epochs with the same batch size. We use the Adam (Kingma & Ba, 2014) optimizer with default hyperparameters and learning rate .001.

We use PCD with a replay buffer to train our RBMs. We use a replay buffer of size 10000 and use 25 steps of Gibbs sampling to update the particles at each training iteration.

# B. Shared Convolutional RBMs

For high dimensional data, the invertible networks used to specify normalizing flow will typically "factor out" groups of features as they apply more invertible transformations. This leaves us with $f(x) = (v_1, \dots, v_L)$. When convolutional models are used, each $v_i$ is spatially structured with its on height, width, and depth. We could unwrap these into vectors, concatenate them together to $v$ and build a fully connected RBM on top of them. This gives the following joint energy function:

$$E(v, h) = \frac{||v - b_v||^2}{2\sigma_v^2} + h^\top b_h + h^\top W \frac{v}{\sigma_v}. \tag{10}$$

Alternatively, we can treat them all separately and given each of them their own weight matrix, leading to the *identical* energy function:

$$E(v, h) = \frac{||v - b_v||^2}{2\sigma_v^2} + h^\top b_h + \sum_{i=1}^{L} h^\top W_i \frac{v_i}{\sigma_v}. \tag{11}$$

For fully connected RBMs this interpretation is pointless, but it is not when dealing with convolutional RBMs. A

convolutional RBM replaces the weight matrix $W$ with a filter $\Omega$ and defines the energy function:

$$E(v,h) = \frac{||v-b_v||^2}{2\sigma_v^2} + h^\top b_h + h^\top \left(\Omega * \frac{v}{\sigma_v}\right). \quad (12)$$

When the input $v$ is split into $L$ spatially structured $v_i$ each with their own height, width, and channels, we can define a new energy function which uses $L$ separate filters to define an RBM with a single hidden tensor state:

$$E(v,h) = \frac{||v-b_v||^2}{2\sigma_v^2} + h^\top b_h + \sum_{i=1}^{L} h^\top \left(\Omega_i * \frac{v_i}{\sigma_v}\right) \quad (13)$$

where each $\Omega_i$ has the same number of output filters, and the kernel size and/or stride is chosen to make sure that the width and height of $\Omega_i * v_i$ is identical. For example, the invertible network for our CIFAR10 model outputs 3 $v_i's$ with height and width equal to $(16,16),(8,8),(4,4)$. Thus $\Omega_1$ has a 5x5 kernel and stride 4, $\Omega_2$ has a 3x3 kernel and stride 2, and $\Omega_3$ has a 3x3 kernel and stride 1. Each kernel has 100 filters leading $h$ to have size $(4,4,100)$. This can be seen pictorially in Figure 2.

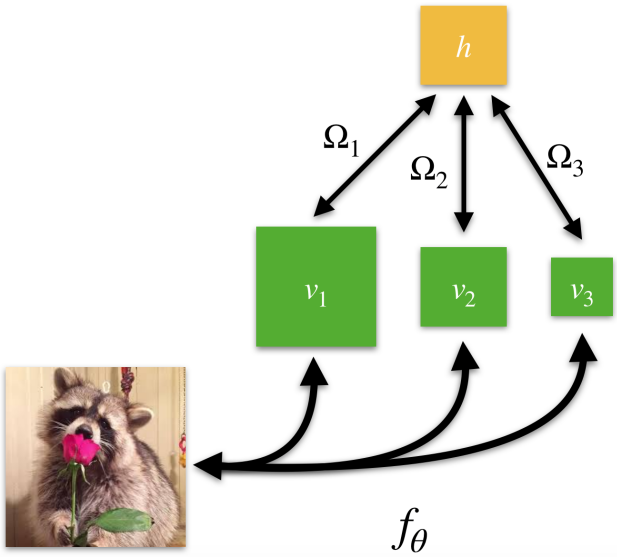


*Figure 2.* Visualization of our shared convolutional RBM structure. The image is mapped to spatially structured $v_1, v_2, v_3$ by the invertible neural network $f_\theta$. Each $v_i$ is then convolved with a distinct filter $\Omega_i$ which maps to the shared hidden state $h$.