Inverting VAEs for Improved Generative Accuracy

Ian Gemp and Sridhar Mahadevan
College of Information and Computer Sciences
University of Massachusetts Amherst
Amherst, MA 01003
{imgemp,mahadeva}@cs.umass.edu

Mario Parente
Electrical and Computer Engineering
University of Massachusetts Amherst
Amherst, MA 01003
mparente@ecs.umass.edu

Abstract

Recent advances in semi-supervised learning with deep generative models have shown promise in generalizing from small labeled datasets ($\mathbf{x}_l, \mathbf{y}_l$) to large unlabeled ones ($\mathbf{x}_u$). When the codomain ($\mathbf{y}$) has known structure, a large un*featured* dataset ($\mathbf{y}_u$) is potentially available. We develop a parameter-efficient, deep semi-supervised generative model for the purpose of exploiting this untapped data source. Empirical results show improved performance in disentangling variable semantics as well as improved discriminative prediction on a new MNIST task.

1 Introduction

Semi-supervised learning aims to improve learning accuracy when a large source of unlabeled data ($\mathbf{x}_u$) is available in addition to a small labeled dataset ($\mathbf{x}_l, \mathbf{y}_l$). Under this setting, inductive learning specifically judges accuracy of the learned mapping $f : \mathbf{x} \to \mathbf{y}$ for all $\mathbf{x}$, while transductive learning focuses on accuracy of this mapping for only $\mathbf{x}_u$. Semi-supervised techniques have significantly improved inductive and transductive learning accuracy for applications ranging from website classification [1], to natural language processing [23], to image segmentation and search [4, 16].

Separately, deep probabilistic models leveraging advances in variational inference have made gains in modeling text [15], images [6], and speech [3]. The architecture common to these models is a deep generative model deemed the *variational autoencoder* [9]. Kingma et al. [2014] developed an extension of this architecture to semi-supervised tasks with excellent semi-supervised classification performance on MNIST [11].

One advantage of using generative models for semi-supervised learning is the availability of a mechanism for generating unobserved data conditioned on the labels. In addition to generating novel observations, this mechanism can be used for exploring the data manifold and typically reveals interesting structure suggesting semantics are disentangled during training.

While semi-supervised learning typically focuses on inductive and transductive learning for labels, in this work, we also stress the importance of inductive accuracy for data generation ($f^{-1} \sim g : \mathbf{y} \to \mathbf{x}$). It is the goal of generative accuracy that leads us to explore the possibility of identifying and exploiting a large source of un*featured* data ($\mathbf{y}_u$): $\mathbf{y}$ without corresponding $\mathbf{x}$. In general, if 1) we know the support of $\mathbf{y}$, 2) we have a strong prior belief on $p(\mathbf{y})$, and 3) labelled instances can be easily synthesized, then $\mathbf{y}_u$ constitutes a potential data source to be tapped. In this work, we consider MNIST where the support of $\mathbf{y}$ is a discrete finite set, but our approach generalizes to other supports (e.g., simplex). In this domain, we are able to exploit our prior knowledge of $\mathbf{y}$ to improve generative accuracy, which in turn, improves discriminative accuracy as well.

Our approach is essentially to "invert" the deep generative model used for standard semi-supervised learning and train it in reverse, thereby considering the flipped semi-supervised task of learning $g : \mathbf{y} \to \mathbf{x}$ given $(\mathbf{x}_l, \mathbf{y}_l)$ and $\mathbf{y}_u$. A key feature of our approach is that f and g are learned jointly with tied parameters for learning efficiency and more critically, model regularization.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

1.1 MNIST Untapped

The latent variable manifolds learned by training variational autoencoders on MNIST are typically smooth. For example, Kingma et al.'s M2 [10] appears to learn a smooth manifold that is easily travelled by considering any convex combination of the disentangled latent factors: traversing the manifold results in images that vary in stroke thickness, style, angle, and letter skew. This suggests that any latent vector in the cross-product of latent factors can be used in combination with any choice of digit class (e.g., $\mathbf{y}$ is a onehot vector) to generate a legitimate image of a digit. In other words, any $\mathbf{z} \times \mathbf{y}$ can be considered a viable source for generating digit images. We ought to be able to exploit this knowledge to better learn the inductive, f, and generative, g, mappings.

We appeal to intuition in the context of a vanilla autoencoder (see Figure 1) and carry over insights developed in that setting to the variational model where distributions replace point estimates. Although variational autoencoders have been shown to behave differently from vanilla autoencoders (specifically with regards to representation learning), our results suggest the insights gleaned are valuable.

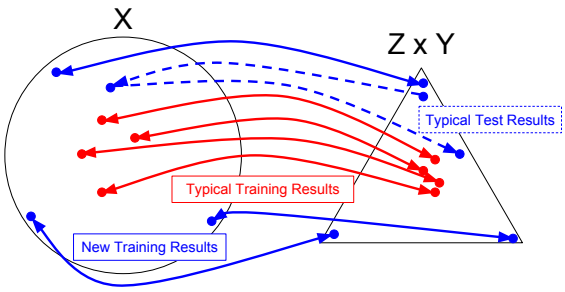

Figure 1: Typical autoencoder training roughly learns a one-to-one mapping only over the training set (solid red). When the model is introduced to new $(\mathbf{z}, \mathbf{y})$ combinations and asked to decode them, its predictions often map back to $(\mathbf{z}', \mathbf{y}')$ different from the original input (dashed, blue)! With our new approach, the model explicitly receives a signal to build a one-to-one mapping for $(\mathbf{z}, \mathbf{y})$ outside the training set (solid blue). The hope is that this encourages the model to be internally consistent.

2 Deep Semi^2-Supervised Generative Model

Motivated by the above problem, we consider the marriage of two probabilistic models to describe the data. The first is a probabilistic model (M2) that describes an MNIST image, $\mathbf{x}$, as being generated by a onehot vector $\mathbf{y}$ in addition to a latent vector $\mathbf{z}$. The joint distribution is assumed to factorize as $p(\mathbf{x}, \mathbf{y}, \mathbf{z}) = p(\mathbf{y})p(\mathbf{z})p(\mathbf{x}|\mathbf{y}, \mathbf{z})$, so the data are explained by the *generative process*:

$$p(\mathbf{y}) = Cat(\pi = 1/10), \quad p(\mathbf{z}) = U(z_{low}, z_{hi}), \quad p_\theta(\mathbf{x}|\mathbf{y}, \mathbf{z}) = f(\mathbf{x}; \mathbf{y}, \mathbf{z}, \theta). \tag{1}$$

Here, $p(\mathbf{y})$ and $p(\mathbf{z})$ are prior distributions and $f(\mathbf{x}; \mathbf{y}, \mathbf{z}, \theta)$ is a distribution whose parameters are functions of $\mathbf{y}$ and $\mathbf{z}$ (e.g., multivariate Bernoulli with $p = b_\theta(\mathbf{y}, \mathbf{z})$). We choose a uniform prior over the categories for $\mathbf{y}$, a clipped uniform prior for $\mathbf{z}$, and neural networks with weights θ for $b_\theta(\mathbf{y}, \mathbf{z})$.

The second is a probabilistic model that describes the *reverse process*: $\mathbf{z}$ and $\mathbf{y}$ are generated by $\mathbf{x}$,

$$q(\mathbf{x}) = Bern(p = 0.5), \quad q_\phi(\mathbf{y}|\mathbf{x}) = g(\mathbf{y}; \mathbf{x}, \phi), \quad q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y}) = h(\mathbf{z}; \mathbf{x}, \mathbf{y}, \phi) \tag{2}$$

where $q(\mathbf{x})$ is an uninformative, uniform prior. To define $q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y})$ with support limited to $[z_{low}, z_{hi}]$, we first draw an intermediate random variable, $\tilde{\mathbf{z}}$, from a diagonal Gaussian. We then pass $\tilde{\mathbf{z}}$ through a scaled sigmoid which conforms the Gaussian distribution to $[z_{low}, z_{hi}]$; to account for the change in probability density, we use techniques from normalizing flows [18] (see Appendix A). To model $q_\phi(\mathbf{y}|\mathbf{x})$ as a categorical distribution, we use the *gumbel-softmax / concrete* distribution [8, 13], which is a popular differentiable approximation. While $q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y})$ is unnecessary for generating $\mathbf{y}$, our reverse model formulation is actually a specific instance of an *auxiliary* generative model [12], which was shown to make the variational distribution more expressive. This term also serves an additional role described next.

Computing the exact posterior of the latent variables $\mathbf{y}$ and $\mathbf{z}$ (i.e., Bayesian inference) in the first model, and likewise $\mathbf{x}$ in the second model, is intractable due to the non-conjugate priors and non-linear dependencies (deep nets). Instead, we approximate the posterior distribution with a separate

non-linear function called a *recognition model* for inferring or "recognizing" the latent variables. One of our novel contributions is to reuse $q_\phi(\mathbf{y}|\mathbf{x})q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})$ as the recognition model for the forward generative model and $p(\mathbf{z})p_\theta(\mathbf{x}|\mathbf{y},\mathbf{z})$ for the reverse model.

To learn the parameters, θ and ϕ, we optimize variational lower bounds on the marginal likelihoods of our data samples. Lower bounds for the forward model are given by

$$\log p_\theta(\mathbf{x},\mathbf{y}) \geq \mathcal{J}_f^l = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})}\Big[\log p_\theta(\mathbf{x}|\mathbf{y},\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y}) + \log p(\mathbf{y}) + \log p(\mathbf{z})\Big] \quad (3)$$

$$\log p_\theta(\mathbf{x}) \geq \mathcal{J}_f^u = \mathbb{E}_{q_\phi(\mathbf{y},\mathbf{z}|\mathbf{x})}\Big[\log p_\theta(\mathbf{x}|\mathbf{y},\mathbf{z}) - \log q_\phi(\mathbf{y},\mathbf{z}|\mathbf{x}) + \log p(\mathbf{y}) + \log p(\mathbf{z})\Big] \quad (4)$$

respectively for labeled and unlabeled samples. We can take advantage of *unfeatured* labels by constructing a lower bound for the reverse model:

$$\log q_\phi(\mathbf{y}) \geq \mathcal{J}_r^u = \mathbb{E}_{p(\mathbf{z})p_\theta(\mathbf{x}|\mathbf{y},\mathbf{z})}\Big[\log q_\phi(\mathbf{y},\mathbf{z}|\mathbf{x}) - \log p_\theta(\mathbf{x}|\mathbf{y},\mathbf{z}) + \log q(\mathbf{x}) - \log p(\mathbf{z})\Big]. \quad (5)$$

The marginal likelihoods for the forward and reverse models, respectively, over the whole dataset are

$$\mathcal{J}_f = \sum_{(\mathbf{x},\mathbf{y})\sim\tilde{p}_l} \mathcal{J}_f^l + \sum_{\mathbf{x}\sim\tilde{p}_{u_x}} \mathcal{J}_f^u, \qquad \mathcal{J}_r = \sum_{\mathbf{y}\sim\tilde{p}_{u_y}} \mathcal{J}_r^u. \quad (6)$$

The predictive distribution for $\mathbf{y}$ appears in (4) and (5), but not (3). As in [10], we introduce an additional discriminative objective to each model that can be learned from the labeled data:

$$\mathcal{J}_f^d = \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\tilde{p}_l} L_y(q_\phi(\cdot|\mathbf{x}),\mathbf{y}), \quad \mathcal{J}_r^d = \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\tilde{p}_l} L_x(p_\theta(\cdot|\mathbf{y},\mathbf{z}),\mathbf{x}) \quad (7)$$

where L_y can, for example, either be $-\mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\tilde{p}_l} \log q_\phi(\mathbf{y}|\mathbf{x})$ or a discriminative loss on the mean of the distribution; L_x is treated similarly.

Technically, the objectives $\mathcal{J}_f$ and $\mathcal{J}_r$ arise from two different generative models. We could introduce an additional latent variable that interpolates between both models as in Conditional Bottleneck Density Estimation [20], capturing our uncertainty in the nature of the generative process: is $\mathbf{x}$ a result of $\mathbf{y}$ or vice versa? For simplicity, we chose to weight each model equally, leaving this question for future research:

$$\mathcal{J} = \mathcal{J}_f + \mathcal{J}_r - \alpha_f^d \mathcal{J}_{\mathrm{f}}^{\mathrm{d}} - \alpha_r^d \mathcal{J}_{\mathrm{r}}^{\mathrm{d}}. \quad (8)$$

In experiments, the M2 model uses the same architecture but without the $\mathcal{J}_r$ term. We learn the parameters θ and ϕ by maximizing (8) using Monte Carlo samples for the latent variables —a technique known as stochastic gradient variational Bayes [9] or stochastic backpropagation [19]. Pseudocode is given in Algorithm 1 in the Appendix where $\Gamma(g_\theta, g_\phi)$ returns a parameter update increment (e.g., $\Gamma = \text{SGD} \rightarrow -(g_\theta, g_\phi)$).

In the model just described, we treat $\mathbf{z}$ as a latent variable in the reverse process. In experiments, we instead treat $\mathbf{z}$ as observed and include it as part of our *unfeatured* dataset, $(\mathbf{y}_u, \mathbf{z}_u)$, which reinforces our prior on $\mathbf{z}$. The proposed model with $\mathbf{z}_u$ observed is implemented using the Theano [22], Lasagne, Parmesan, and Scikit-learn packages [17]—code available @ https://github.com/all-umass/untapped.

3 MNIST Experiment

MNIST digit recognition provides a well-known benchmark in machine learning for supervised, semi-supervised, and unsupervised learning. Here, we construct a new MNIST task where labeled data is only available for a subset of the digits, $0-4$. The model is not only expected to attribute correct semantics to $\mathbf{y}$ for digits $0-4$, but also $5-9$, which makes this task particularly challenging. In other words, the model should be able to generate appropriate images conditioned on one-hots (e.g., conditioning on $\mathbf{y} = [1,\ldots,0]$ should generate an image of a zero). Note that while the model is provided a signal to attribute the first five dimensions of $\mathbf{y}$ to $0-4$, the remaining dimensions are free to disentangle meaning as the model sees fit; the hope is that $5-9$ will be ascribed to some random permutation of the remaining dimensions and not distributed across the dimensions (e.g., 5 should not be represented by $y = [1/10,\ldots,1/10]$) or folded into $\mathbf{z}$. We use a 2d $\mathbf{z}$ as in [9] to capture variation in the digits.

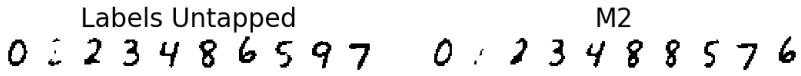

Figure 2: Digit Extraction (means): Our proposed model (left) and M2 (right) generate images (means, μ) conditioned on one-hot vectors ($\mathbf{y}$). The models are provided a supervised signal to assign digits $0-4$ to the first 5 dimensions. No signal is given for assigning the remaining dimensions. Notice, (left) assigns a unique digit to each dimension, while the digit 9 does not appear in (right).

Figure 2 demonstrates the tendency of our proposed model to encourage $\mathbf{y}$ towards the desired representation. All digits are generated by conditioning on the 10 possible one-hots and the mean of $\mathbf{z}$ over the training set: $\mathbf{x}' \sim p_\theta(\mathbf{x}|\mathbf{y}=[1,\ldots,0], \bar{\mathbf{z}} = \texttt{mean}(q_\phi(\mathbf{z}|\mathbf{x}_l, \mathbf{y}_l))$. In the M2 model, the digit 8's representation is distributed across the sixth and seventh dimensions of $\mathbf{y}$. In fact, the digit 9 does not appear to be represented by any one-hot representation in the M2 model. In contrast, our proposed model has attributed a unique digit to each possible one-hot vector: $0, 1, 2, 3, 4|\mathbf{8}, \mathbf{6}, \mathbf{5}, \mathbf{9}, \mathbf{7}$.

In order to test discriminative generalization, we need to learn the permutation of $\mathbf{y}$ found by each model. To do this, we first generate canonical digits for 5-9 by conditioning on each of the corresponding one-hots. We then use a logistic regression classifier pre-trained on the entire dataset to compute a probability distribution over digits for every image, $p(\text{digit} = i|\text{image})$. If $j = \arg\max_i p(\text{digit} = i|\text{image})$ is unique to a single dimension, we assign that dimension the digit j. We use the highest probability to split ties otherwise. To measure discriminative generalization, we simply permute $\mathbf{y}$, and compute the cross-entropy over the predictions for digits 5-9. It is not surprising, given our model's learned representation, that training with $\mathbf{y}_u$ results in improved discriminative generalization. M2 achieves an average cross-entropy loss over 10 trials of 2.30 on this task while training with *unfeatured* $\mathbf{y}$ achieves 2.07.

Table 1: Test Error: Cross-entropy for predicted distribution over digits vs ground truth.

	M2	Untapped
Cross-Entropy	2.30	2.07

4 Related Work

Previous work attempted to exploit this same untapped resource by minimizing the Output Distribution Matching cost [21], which is the gap between the log marginal probability of the data and the evidence lower bound (ELBO): $KL(p(\mathbf{y})|q_\phi(\mathbf{y}|\mathbf{x}))$. A large portion of VAE research is devoted to shrinking this gap to tighten the lower bound, and so any gains made there should transfer to our framework.

Chen et al. [2] equipped the GAN minimax objective with a lower bound on the mutual information between the observed data ($\mathbf{x}$) and the latent code ($\mathbf{y}$ here) given $\mathbf{z}$:

$$\mathbf{I}(\mathbf{y}; \mathbf{x}' \sim p(\mathbf{x}|\mathbf{y}, \mathbf{z})) \geq \mathbb{E}_{p(\mathbf{y})p_\theta(\mathbf{x}|\mathbf{y},\mathbf{z})}\Big[\log q_\phi(\mathbf{y}|\mathbf{x})\Big] + H(\mathbf{y}) \tag{9}$$

If we fix $\mathbf{z}$ in our reverse model, this is equivalent to the first term in equation (5) sans $H(\mathbf{y})$. This objective can be also viewed as reconstruction error on the reverse model which is what motivated our proposed model. Δ-GAN [5] is also closely related as it explicitly trains with samples drawn from the prior $p(y)$, effectively creating an *unfeatured* dataset.

Our work can also be seen as strengthening the influence of the prior distribution. By directly feeding $\mathbf{y}_u \sim p(\mathbf{y})$ to our model, we are effectively treating $p(\mathbf{y})$ as "truth", therefore, this work follows in line with that of [14] and [7] where a GAN and scaled-KL-divergence term are used, respectively, to more harshly penalize deviations of the posterior from the prior.

5 Conclusion & Future Work

In this work, we identified a potentially untapped resource, *unfeatured* labels. We then proposed an extension to the semi-supervised variational autoencoder capable of leveraging this newfound

training signal. In future work, we will investigate our model's ability to improve performance on computer vision and NLP tasks. For example, image captioning requires assigning a short (limited length) description to an image. In this scenario, images are plentiful, yet captioned images are rare. Furthermore, captions follow a very specific structure enforced by the language grammar—this allows them to be synthesized which would provide a large *untapped* label source.

References

[1] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM, 1998.

[2] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2172–2180, 2016.

[3] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. In *Advances in neural information processing systems*, pages 2980–2988, 2015.

[4] Rob Fergus, Yair Weiss, and Antonio Torralba. Semi-supervised learning in gigantic image collections. In *Advances in neural information processing systems*, pages 522–530, 2009.

[5] Zhe Gan, Liqun Chen, Weiyao Wang, Yuchen Pu, Yizhe Zhang, and Lawrence Carin. Triangle generative adversarial networks. In *Advances in Neural Information Processing Systems*, pages 5251–5260, 2017.

[6] Ishaan Gulrajani, Kundan Kumar, Faruk Ahmed, Adrien Ali Taiga, Francesco Visin, David Vazquez, and Aaron Courville. Pixelvae: A latent variable model for natural images. *arXiv preprint arXiv:1611.05013*, 2016.

[7] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *In Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

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

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

[10] D.P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, pages 3581–3589, 2014.

[11] Yann LeCun, Corinna Cortes, and Christopher JC Burges. The mnist database of handwritten digits, 1998.

[12] Lars Maaløe, Casper Kaae Sønderby, Søren Kaae Sønderby, and Ole Winther. Auxiliary deep generative models. *arXiv preprint arXiv:1602.05473*, 2016.

[13] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

[14] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

[15] Yishu Miao, Lei Yu, and Phil Blunsom. Neural variational inference for text processing. In *Proc. ICML*, 2016.

[16] George Papandreou, Liang-Chieh Chen, Kevin Murphy, and Alan L Yuille. Weakly-and semi-supervised learning of a dcnn for semantic image segmentation. *arXiv preprint arXiv:1502.02734*, 2015.

[17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

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

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

[20] Rui Shu, Hung H Bui, and Mohammad Ghavamzadeh. Bottleneck conditional density estimation. *arXiv preprint arXiv:1611.08568*, 2016.

[21] Ilya Sutskever, Rafal Jozefowicz, Karol Gregor, Danilo Rezende, Tim Lillicrap, and Oriol Vinyals. Towards principled unsupervised learning. *arXiv preprint arXiv:1511.06440*, 2015.

[22] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016. URL `http://arxiv.org/abs/1605.02688`.

[23] Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics, 2010.

A LogisticFlow

We use a normalizing flow [18] designed to warp a distribution to a d-dimensional hypercube, $[a, b]^d$. The flow function and its inverse are defined below in equations (10) and (13). Equation (16) allows linear time computation of the log-determinant of the Jacobian which is necessary to efficiently compute the log-density of the resulting distribution over the hypercube using only the original Gaussian density.

The scaled sigmoid transformation $\tilde{\mathbf{z}} \rightarrow \mathbf{z}$ is simply given by

$$\mathbf{n}_i(\tilde{\mathbf{z}}) = a + \frac{b-a}{1+e^{-\tilde{\mathbf{z}}_i}} = a + (b-a)\sigma(\tilde{\mathbf{z}}_i) \tag{10}$$

$$\mathbf{z}_i = \mathbf{n}_i(\tilde{\mathbf{y}}) \ \forall i \in \{1, \ldots, I-1\}. \tag{11}$$

The inverse transformation is then

$$\hat{\mathbf{z}}_i = \frac{\mathbf{z}_i - a}{b-a} \tag{12}$$

$$\mathbf{n}_i^{-1}(\mathbf{z}) = \log\Big(\frac{\hat{\mathbf{z}}_i}{1-\hat{\mathbf{z}}_i}\Big). \tag{13}$$

The log determinant of the Jacobian can then be computed as

$$\frac{\partial \mathbf{n}_i}{\partial \tilde{\mathbf{z}}_j} = \mathbb{I}(i=j)(b-a)\sigma(1-\hat{\mathbf{z}}_i)\sigma(\hat{\mathbf{z}}_i) \tag{14}$$

$$|\frac{\partial \mathbf{n}}{\partial \tilde{\mathbf{z}}}| = (b-a)^d \prod_i \sigma(1-\hat{\mathbf{z}}_i)\sigma(\hat{\mathbf{z}}_i) \tag{15}$$

$$\log|\frac{\partial \mathbf{n}}{\partial \tilde{\mathbf{z}}}| = d\log(b-a) + \sum_i \log(\sigma(1-\hat{\mathbf{z}}_i)) + \log(\sigma(\hat{\mathbf{z}}_i)). \tag{16}$$

B Alternative ELBO

As mentioned in the paper, we actually treat $\mathbf{z}$ as observed in the reverse model giving us $(\mathbf{y}_u, \mathbf{z}_u)$ pairs. This results in a minor modification to the evidence lower bound in equation (5).

$$\log q_\phi(\mathbf{y}, \mathbf{z}) \geq \mathbb{E}_{p_\theta(\mathbf{x}|\mathbf{y},\mathbf{z})}\Big[\log q_\phi(\mathbf{y}|\mathbf{x}) + \log q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y}) - \log p_\theta(\mathbf{x}|\mathbf{y},\mathbf{z}) + \log q(\mathbf{x})\Big] \tag{17}$$

C Algorithm

Algorithm 1 Learning the Model

while training() **do**
  $\mathcal{D} \leftarrow$ getRandomMiniBatch()
  $\mathcal{J} = 0$
  for all $\{\mathbf{x}_i, \mathbf{y}_i\} \in \mathcal{D}_{\{x,y\}}$ **do**
    $\mathbf{z}_i \sim q_\phi(\mathbf{z}|\mathbf{x}_i, \mathbf{y}_i)$
    $\mathcal{J} \mathrel{+}= \alpha_f \mathcal{J}_f^l - \alpha_f^d \mathcal{J}_f^d$
    $\mathbf{z}_i \sim p(\mathbf{z})$
    $\mathcal{J} \mathrel{+}= \alpha_r \mathcal{J}_r^l - \alpha_r^d \mathcal{J}_r^d$
  end for
  for all $\mathbf{x}_i \in \mathcal{D}_x$ **do**
    $\mathbf{y}_i \sim q_\phi(\mathbf{y}|\mathbf{x}_i),\ \mathbf{z}_i \sim q_\phi(\mathbf{z}|\mathbf{x}_i, \mathbf{y}_i)$
    $\mathcal{J} \mathrel{+}= \alpha^f \mathcal{J}_f^u$
  end for
  for all $\mathbf{y}_i \in \mathcal{D}_y$ **do**
    $\mathbf{z}_i \sim p(\mathbf{z}),\ \mathbf{x}_i \sim p_\theta(\mathbf{x}_i|\mathbf{y}_i, \mathbf{z}_i)$
    $\mathcal{J} \mathrel{+}= \alpha_r \mathcal{J}_r^u$
  end for
  $(g_\theta, g_\phi) \leftarrow (-\frac{\partial \mathcal{J}^\alpha}{\partial \theta}, -\frac{\partial \mathcal{J}^\alpha}{\partial \phi})$
  $(\theta, \phi) \leftarrow (\theta, \phi) + \Gamma(g_\theta, g_\phi)$
end while

D Network Architectures & Training Setup

- Optimizer: Adam with gradient clipping $(-1, 1)$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1e-4$
- Monte Carlos samples to estimate expectation: 1
- Priors are all uniform distributions
- $z_{low}, z_{hi} = -1.5, 1.5$
- $\mathbf{x}$-discriminative loss: L2
- $\mathbf{y}$-discriminative loss: cross-entropy
- $\mathbf{z}$-dimensionality: 2
- Hidden unit nonlinearities: tanh
- $\mathbf{x} \rightarrow \mathbf{y}$ hidden units: $[50, 50]$
- $\mathbf{x} \rightarrow \mathbf{y}$ sampling distribution: Concrete distribution / Gumbel-softmax
- $\mathbf{x} \rightarrow \mathbf{y}$ output nonlinearity: None
- $(\mathbf{x}, \mathbf{y}) \rightarrow \mathbf{z}$ hidden units: $[10, 10]$
- $\mathbf{x} \rightarrow \mathbf{y}$ sampling distribution: diagonal Gaussian
- $(\mathbf{x}, \mathbf{y}) \rightarrow \mathbf{z}$ output nonlinearity: sigmoid
- $(\mathbf{y}, \mathbf{z}) \rightarrow \tilde{\mathbf{x}}$ hidden units: $[250, 500]$
- $\mathbf{x} \rightarrow \mathbf{y}$ sampling distribution: Bernoulli distribution
- $(\mathbf{y}, \mathbf{z}) \rightarrow \tilde{\mathbf{x}}$ output nonlinearity: None
- $(\alpha_f^d, \alpha_r^d) = (10, 0.1)$
- Batch size: 10000
- # of training epochs: 150
- Learning rate: 0.01

E ELBOs

E.1 Forward Model with Labeled Data

$$\log p_\theta(x,y) = \log\Big(\int_z p_\theta(x,y,z)dz\Big) \tag{18}$$
$$= \log\Big(\int_z p_\theta(x,y,z)\frac{q_\phi(z|x,y)}{q_\phi(z|x,y)}dz\Big) \tag{19}$$
$$= \log\Big(\mathbb{E}_{q_\phi(z|x,y)}\Big[\frac{p_\theta(x,y,z)}{q_\phi(z|x,y)}\Big]\Big) \tag{20}$$
$$\geq \mathbb{E}_{q_\phi(z|x,y)}\Big[\log\Big(\frac{p_\theta(x,y,z)}{q_\phi(z|x,y)}\Big)\Big] \tag{21}$$
$$= \mathbb{E}_{q_\phi(z|x,y)}\Big[\log p_\theta(x,y,z) - \log q_\phi(z|x,y)\Big] \tag{22}$$
$$= \mathbb{E}_{q_\phi(z|x,y)}\Big[\log p_\theta(x|y,z) + \log p(y) + \log p(z) - \log q_\phi(z|x,y)\Big] \tag{23}$$
$$= ELBO_f^l \tag{24}$$

$$KL\Big(q_\phi(z|x,y)\Big|p_\theta(z|x,y)\Big) = \mathbb{E}_{q_\phi(z|x,y)}\Big[\log\Big(\frac{q_\phi(z|x,y)}{p_\theta(z|x,y)}\Big)\Big] \tag{25}$$
$$= \mathbb{E}_{q_\phi(z|x,y)}\Big[\log q_\phi(z|x,y)\Big] - \mathbb{E}_{q_\phi(z|x,y)}\Big[\log p_\theta(z|x,y)\Big] \tag{26}$$
$$= \mathbb{E}_{q_\phi(z|x,y)}\Big[\log q_\phi(z|x,y)\Big] - \mathbb{E}_{q_\phi(z|x,y)}\Big[\log \frac{p_\theta(x,y,z)}{p_\theta(x,y)}\Big] \tag{27}$$
$$= \mathbb{E}_{q_\phi(z|x,y)}\Big[\log q_\phi(z|x,y) - \log p_\theta(x,y,z)\Big] + \log p_\theta(x,y) \tag{28}$$
$$= \mathbb{E}_{q_\phi(z|x,y)}\Big[\log q_\phi(z|x,y) - \log p_\theta(x|y,z) - \log p(y) - \log p(z)\Big] + \log p_\theta(x,y) \tag{29}$$
$$= -ELBO_f^l + \log p_\theta(x,y) \tag{30}$$
$$\Rightarrow ELBO_f^l = \log p_\theta(x,y) - KL\Big(q_\phi(z|x,y)\Big|p_\theta(z|x,y)\Big) \tag{31}$$

The ELBO bound on the marginal log likelihood becomes tighter as $q_\phi(z|x,y)$ better approximates $p_\theta(z|x,y)$.

E.2 Forward Model with Unlabeled Data

$$\log p_\theta(x) = \log\Big(\int_{y\times z} p_\theta(x,y,z)dydz\Big) \tag{32}$$
$$= \log\Big(\int_{y\times z} p_\theta(x,y,z)\frac{q_\phi(y,z|x)}{q_\phi(y,z|x)}dydz\Big) \tag{33}$$
$$= \log\Big(\mathbb{E}_{q_\phi(y,z|x)}\Big[\frac{p_\theta(x,y,z)}{q_\phi(y,z|x)}\Big]\Big) \tag{34}$$
$$\geq \mathbb{E}_{q_\phi(y,z|x)}\Big[\log\Big(\frac{p_\theta(x,y,z)}{q_\phi(y,z|x)}\Big)\Big] \tag{35}$$
$$= \mathbb{E}_{q_\phi(y,z|x)}\Big[\log p_\theta(x,y,z) - \log q_\phi(y,z|x)\Big] \tag{36}$$
$$= \mathbb{E}_{q_\phi(y,z|x)}\Big[\log p_\theta(x|y,z) + \log p(y) + \log p(z) - \log q_\phi(y,z|x)\Big] \tag{37}$$
$$= ELBO_f^u \tag{38}$$

$$KL\Big(q_\phi(y,z|x)\Big|p_\theta(y,z|x)\Big) = \mathbb{E}_{q_\phi(y,z|x)}\Big[\log\Big(\frac{q_\phi(y,z|x)}{p_\theta(y,z|x)}\Big)\Big] \tag{39}$$

$$= \mathbb{E}_{q_\phi(y,z|x)}\Big[\log q_\phi(y,z|x)\Big] - \mathbb{E}_{q_\phi(y,z|x)}\Big[\log p_\theta(y,z|x)\Big] \tag{40}$$

$$= \mathbb{E}_{q_\phi(y,z|x)}\Big[\log q_\phi(y,z|x)\Big] - \mathbb{E}_{q_\phi(y,z|x)}\Big[\log \frac{p_\theta(x,y,z)}{p_\theta(x)}\Big] \tag{41}$$

$$= \mathbb{E}_{q_\phi(y,z|x)}\Big[\log q_\phi(y,z|x) - \log p_\theta(x,y,z)\Big] + \log p_\theta(x) \tag{42}$$

$$= \mathbb{E}_{q_\phi(y,z|x)}\Big[\log q_\phi(y,z|x) - \log p_\theta(x|y,z) - \log p(y) - \log p(z)\Big] + \log p_\theta(x) \tag{43}$$

$$= -ELBO_f^u + \log p_\theta(x) \tag{44}$$

$$\Rightarrow ELBO_f^u = \log p_\theta(x) - KL\Big(q_\phi(y,z|x)\Big|p_\theta(y,z|x)\Big) \tag{45}$$

The ELBO bound on the marginal log likelihood becomes tighter as $q_\phi(y,z|x)$ better approximates $p_\theta(y,z|x)$.

E.3 Reverse Model with Labeled Data

$$\log q_\phi(x,y) = \log\Big(\int_z q_\phi(x,y,z)dz\Big) \tag{46}$$

$$= \log\Big(\int_z q_\phi(x,y,z)\frac{p(z)}{p(z)}dz\Big) \tag{47}$$

$$= \log\Big(\mathbb{E}_{p(z)}\Big[\frac{q_\phi(x,y,z)}{p(z)}\Big]\Big) \tag{48}$$

$$\geq \mathbb{E}_{p(z)}\Big[\log\Big(\frac{q_\phi(x,y,z)}{p(z)}\Big)\Big] \tag{49}$$

$$= \mathbb{E}_{p(z)}\Big[\log q_\phi(x,y,z) - \log p(z)\Big] \tag{50}$$

$$= \mathbb{E}_{p(z)}\Big[\log q_\phi(z|x,y) + \log q_\phi(y|x) + \log q(x) - \log p(z)\Big] \tag{51}$$

$$= ELBO_r^l \tag{52}$$

$$KL\Big(p(z)\Big|q_\phi(z|x,y)\Big) = \mathbb{E}_{p(z)}\Big[\log\Big(\frac{p(z)}{q_\phi(z|x,y)}\Big)\Big] \tag{53}$$

$$= \mathbb{E}_{p(z)}\Big[\log p(z)\Big] - \mathbb{E}_{p(z)}\Big[q_\phi(z|x,y)\Big] \tag{54}$$

$$= \mathbb{E}_{p(z)}\Big[\log p(z)\Big] - \mathbb{E}_{p(z)}\Big[\log \frac{q_\phi(x,y,z)}{q_\phi(x,y)}\Big] \tag{55}$$

$$= \mathbb{E}_{p(z)}\Big[\log p(z) - \log q_\phi(x,y,z)\Big] + \log q_\phi(x,y) \tag{56}$$

$$= \mathbb{E}_{p(z)}\Big[\log p(z) - \log q_\phi(z|x,y) - \log q_\phi(y|x) - \log q(x)\Big] + \log q_\phi(x,y) \tag{57}$$

$$= -ELBO_r^l + \log q_\phi(x,y) \tag{58}$$

$$\Rightarrow ELBO_r^l = \log q_\phi(x,y) - KL\Big(p(z)\Big|q_\phi(z|x,y)\Big) \tag{59}$$

We present the KL using $p(z)$ instead of $p(z|x,y)$ for practical reasons—we do not have a mechanism for easily sampling from $p(z|x,y)$ or computing its density. The ELBO bound on the marginal

log likelihood becomes tighter as $p(z)$ better approximates $q_\phi(z|x,y)$. This will likely not be a tight bound, so we expect maximizing this ELBO will do more harm than good. Experiments (not included) support this intuition, so we have omitted it from the model.

E.4 Reverse Model with Unlabeled Data

$$\log q_\phi(y) = \log\Big(\int_{x\times z} q_\phi(x,y,z)dxdz\Big) \tag{60}$$

$$= \log\Big(\int_{x\times z} q_\phi(x,y,z)\frac{p_\theta(x,z|y)}{p_\theta(x,z|y)}dxdz\Big) \tag{61}$$

$$= \log\Big(\mathbb{E}_{p_\theta(x,z|y)}\Big[\frac{q_\phi(x,y,z)}{p_\theta(x,z|y)}\Big]\Big) \tag{62}$$

$$\geq \mathbb{E}_{p_\theta(x,z|y)}\Big[\log\Big(\frac{q_\theta(x,y,z)}{p_\theta(x,z|y)}\Big)\Big] \tag{63}$$

$$= \mathbb{E}_{p_\theta(x,z|y)}\Big[\log q_\phi(x,y,z) - \log p_\theta(x,z|y)\Big] \tag{64}$$

$$= \mathbb{E}_{p_\theta(x,z|y)}\Big[\log q_\phi(z|x,y) + \log q_\phi(y|x) + \log q(x) - \log p_\theta(x|y,z) - \log p(z)\Big] \tag{65}$$

$$= \mathbb{E}_{p_\theta(x|y,z)p(z)}\Big[\log q_\phi(z|x,y) + \log q_\phi(y|x) + \log q(x) - \log p_\theta(x|y,z) - \log p(z)\Big] \tag{66}$$

$$= ELBO_r^u \tag{67}$$

$$KL\Big(p_\theta(x,z|y)\Big|q_\phi(x,z|y)\Big) = \mathbb{E}_{p_\theta(x,z|y)}\Big[\log\Big(\frac{p_\theta(x,z|y)}{q_\phi(x,z|y)}\Big)\Big] \tag{68}$$

$$= \mathbb{E}_{p_\theta(x,z|y)}\Big[\log p_\theta(x,z|y)\Big] - \mathbb{E}_{q_\phi(y,z|x)}\Big[\log q_\phi(x,z|y)\Big] \tag{69}$$

$$= \mathbb{E}_{p_\theta(x,z|y)}\Big[\log p_\theta(x,z|y)\Big] - \mathbb{E}_{q_\phi(y,z|x)}\Big[\log \frac{q_\phi(x,y,z)}{q_\phi(y)}\Big] \tag{70}$$

$$= \mathbb{E}_{p_\theta(x,z|y)}\Big[\log p_\theta(x,z|y) - \log q_\phi(x,y,z)\Big] + \log q_\phi(y) \tag{71}$$

$$= \mathbb{E}_{p_\theta(x,z|y)}\Big[\log p_\theta(x|y,z) + \log p(z) - \log q_\phi(z|x,y) \tag{72}$$

$$\qquad - \log q_\phi(y|x) - \log q(x)\Big] + \log q_\phi(y)$$

$$= -ELBO_r^u + \log q_\phi(y) \tag{73}$$

$$\Rightarrow ELBO_r^u = \log q_\phi(y) - KL\Big(p_\theta(x,z|y)\Big|q_\phi(x,z|y)\Big) \tag{74}$$

The ELBO bound on the marginal log likelihood becomes tighter as $p_\theta(x,z|y)$ better approximates $q_\phi(x,z|y)$.

F Motivation for Untapped Starting with M2

In the M2 model [9], the latent factors z are assumed to be independent of the class labels y. This is important for modeling purposes. For example, z may represent the style of a handwritten digit while y may denote the number represented by that digit. We can write any digit we like with any given style, therefore, it is natural to think of these two variables as independent. This implies that the joint distribution over data, labels, and latent factors can be simplified as follows:

$$p(x,y,z) = p(z)p(y|z)p(x|y,z) \tag{75}$$

$$= p(z)p(y)p(x|y,z). \tag{76}$$

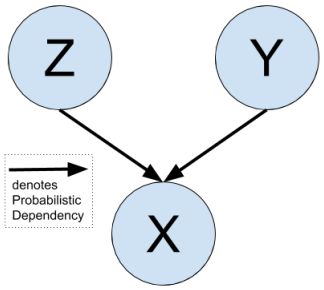

Figure 3: Forward Generative Graphical Model

This generative process directly implies the corresponding graphical model (see Figure 3).

In variational inference, we propose an approximate distribution for the posterior over the latent variables, which in this case are y and z. Notice that, by d-separation, z and y are not necessarily independent given x, so we cannot simplify the chain rule factorization of the posterior:

$$q(y, z|x) = q(y|x)q(z|x, y) \tag{77}$$

Moreover, this particular choice for the factorization was made because it explicitly represents the discriminative distribution over labels, $q(y|x)$, which is convenient for the auxiliary classification task. This recognition process directly implies the corresponding recognition architecture (see Figure 4).

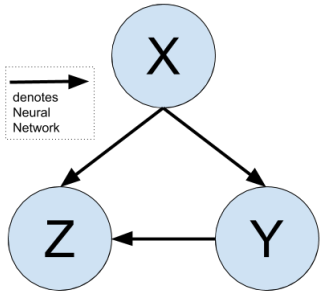

Figure 4: Forward Recognition Network Architecture

This explains why Kingma et al. chose this structure for their M2 model. Given their choice, we would like to define a reverse model without creating any additional networks. This cuts down on the number of weights we need to learn which helps prevent overfitting. The forward model treats y and z as latent variables and x as observed. In the reverse, we will do the reverse! The variables y and z will be observed and x will be latent. We can reuse the networks in the forward recognition model to define the generative process for the reverse model:

$$q(x, y, z) = q(x)q(y|x)q(z|x, y). \tag{78}$$

The recognition model for the reverse model will be used to approximate the posterior over x. In this case, we can reuse the generative model for the forward model: $p(x|y, z)$.