Bottleneck Conditional Density Estimation

Rui Shu[*]
Stanford University
ruishu@stanford.edu

Hung H. Bui
Adobe Research
hubui@adobe.com

Mohammad Ghavamzadeh
Adobe Research
ghavamza@adobe.com

Abstract

We propose a neural network framework for high-dimensional conditional density estimation. The Bottleneck Conditional Density Estimator (BCDE) is a variant of the conditional variational autoencoder (CVAE) that employs layer(s) of stochastic variables as the bottleneck between the input x and target y, where both are high-dimensional. The key to effectively train BCDEs is the hybrid blending of the conditional generative model with a joint generative model that leverages unlabeled data to regularize the conditional model. We show that the BCDE significantly outperforms the CVAE in MNIST quadrant prediction benchmarks in the fully supervised case and establishes new benchmarks in the semi-supervised case.

1 Introduction

Conditional density estimation (CDE) refers to the problem of estimating a conditional density $p(y|x)$ for the input vector x and target vector y. In contrast to classification where the target y is simply a discrete class label, y is typically continuous or high-dimensional in CDE. Furthermore, we want to estimate the full conditional density (as opposed to its conditional mean in regression), a task that becomes important when the conditional distribution has multiple modes. CDE problems where both x and y are high-dimensional have a wide range of important applications, including video prediction, cross-modality prediction (e.g., given one modality (e.g., images) predict other modalities (e.g., sounds, captions)), model estimation in model-based reinforcement learning, and so on.

Classical non-parametric conditional density estimators typically rely directly on local Euclidean distance in the original input and target space [7]. This approach quickly becomes ineffective in high-dimensions from both a computational and a statistical point of view. Recent advances in deep generative models have led to new parametric models for high-dimensional CDE tasks, namely the conditional variational autoencoders (CVAE) [21]. CVAEs have been applied to a variety of problems, such as MNIST quadrant prediction, segmentation [21], attribute-based image generation [26], and machine translation [27].

The original CVAE suffers from two statistical deficiencies. First, they inherently do not learn the distribution of the input x. We argue that in the case of high-dimensional input x where there might exist a latent low-dimensional representation (such as a low-dimensional manifold) of the data, recovering this structure is of importance, even if the task at hand is to learn about the conditional density $p(y|x)$. Second, for many conditional density estimation tasks, the acquisition of labeled points is costly, motivating the need for semi-supervised CDE. A purely conditional model would not be able to utilize any available unlabeled data[2]. We note that while variational methods [11, 18] have been applied to semi-supervised classification (where y is simply a class label) [12, 14], semi-supervised CDE (where y is high-dimensional) remains an open problem.

We focus on a set of deep conditional generative models, which we call *bottleneck conditional density estimators* (BCDEs). In BCDEs, the input x only influences the target y via layers of bottleneck

[*]Work was done at Adobe Research.

[2]We define a "labeled point" to be a paired (x, y) sample, and an "unlabeled point" to be unpaired x or y.

stochastic variables $z = \{z_i\}$ in the conditional generative path. The BCDE naturally has a joint generative sibling model which we denote the *bottleneck joint density estimator* (BJDE), where the bottleneck z generates x and y independently. Following [13], we propose a hybrid training framework that regularizes the conditionally-trained BCDE parameters toward the jointly-trained BJDE parameters. This is the key feature that enables semi-supervised learning for conditional density estimation in the BCDEs.

In addition to our core hybrid framework for training the BCDE, we propose a method to make the BJDE and BCDE recognition models more compact. We show that regularities within the BJDE and BCDE posteriors enables extensive parameter sharing within the recognition model. We call this approach *factored inference*. A more compact recognition model provides additional protection against over-fitting, making factored inference especially useful in the semi-supervised regime.

Using our BCDE hybrid training framework, we establish new benchmarks for the MNIST quadrant prediction task [21] in both the fully-supervised and semi-supervised regimes. Our experiments show that **1)** hybrid training improves performance for fully-supervised CDE, **2)** combining hybrid training and factored inference enables strong performance in semi-supervised CDE, and that **3)** hybrid training encourages the model to learn better representations of both the input and target.

2 Related work

2.1 Variational Autoencoders

The Variational Autoencoder (VAE) is a deep generative model for density estimation. It consists of a latent variable z with unit Gaussian prior $z \sim \mathcal{N}(0, I_k)$ which in turn generates a conditionally Gaussian observable vector $x|z \sim \mathcal{N}(\mu_\theta(z), \text{diag}(\sigma^2_\theta(z))$ where μ and σ^2 are non-linear neural networks and θ represents their parameters. The VAE can be seen as a non-linear generalization of the probabilistic PCA [24], and thus can recover non-linear manifolds in the data. VAE's flexibility however makes posterior inference of the latent variable intractable. This inference issue is addressed via a recognition model $q_\phi(z|x)$ which serves as an amortized variational approximation of the intractable posterior $p(z|x)$. Typically $q(\cdot|\cdot)$ is also a conditionally Gaussian distribution parameterized by neural networks. Learning the VAE is done by jointly optimizing the parameters of both the generative and recognition models so as to maximize an objective that resembles an autoencoder regularized reconstruction loss [11]

$$\sup_{\theta,\phi} \mathbb{E}_{x\sim\tilde{P}} \left(\mathbb{E}_{q(z|x)} \left[\ln p(x|z) \right] - \mathrm{D}_{\mathrm{KL}}(q_\phi(z|x)||p(z)) \right) \equiv \mathbb{E}_{x\sim\tilde{P}} \mathbb{E}_{q_\phi(z|x)} \left[\ln \frac{p(z)p_\theta(x|z)}{q_\phi(z|x)} \right]. \quad (1)$$

We note that the above objective can be rewritten in a form that exposes its connection to the variational lower bound of the log-likelihood

$$\sup_{\theta} \left(\mathbb{E}_{x\sim\tilde{P}} \ln p_\theta(x) - \inf_{\phi} \mathbb{E}_{x\sim\tilde{P}} \mathrm{D}_{\mathrm{KL}}(q_\phi(z|x) \| p_\theta(z|x)) \right). \quad (2)$$

We make two remarks regarding minimization of the term $\mathrm{D}_{\mathrm{KL}}(q_\phi(z|x)\|p_\theta(z|x))$ in Eq. (2). First, when $q(\cdot|\cdot)$ is a conditionally independent Gaussian, this approximation is at best only as good as the mean-field approximation which minimizes $\mathrm{D}_{\mathrm{KL}}(q\|p_\theta(z|x))$ over all independent Gaussian q. Second, this term serves as a form of amortized posterior regularization that encourages the posterior $p_\theta(z|x)$ to be close to an amortized variational family [2, 5, 6]. In practice, both θ and ϕ are jointly optimized in Eq. (1), and the reparameterization trick [11] is used to transform the expectation over $z \sim q_\phi(z|x)$ into $\epsilon \sim \mathcal{N}(0, I_k)$; $z = \mu_\phi(x) + \text{diag}(\sigma^2_\phi(x))\epsilon$ which leads to an easily obtained stochastic gradient.

2.2 Conditional VAEs (CVAEs)

In [21], the authors introduced the conditional version of variational autoencoders. The conditional generative model is similar to the VAE, except that the latent variable z and the generating distribution of $y|z$ are both conditioned on the input x. The conditional generative path is

$$z \sim p_\theta(z|x) = \mathcal{N}(z|\mu_{z,\theta}(x), \text{diag}(\sigma^2_{z,\theta}(x))), \quad (3)$$

$$y \sim p_\theta(y|x,z) = \mathcal{N}(y|\mu_{y,\theta}(x,z), \text{diag}(\sigma^2_{y,\theta}(x,z))) \text{ or } \textit{Bern}(y|\mu_{y,\theta}(x,z)). \quad (4)$$

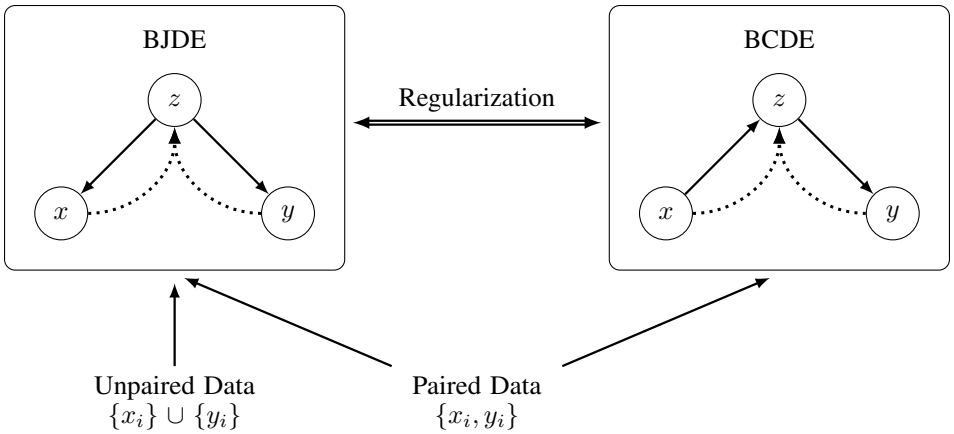


Figure 1: The hybrid training procedure that regularizes the BCDE toward the BJDE. This regularization enables the BCDE to indirectly leverage unpaired x and y for conditional density estimation.

where θ denotes the parameters of the neural networks used in the generative path. The CVAE is trained by maximizing a lower bound of the conditional likelihood

$$\ln p_\theta(y|x) \geq \mathbb{E}_{q_\phi(z|x,y)} \left[\ln \frac{p_\theta(z|x) p_\theta(y|x,z)}{q_\phi(z|x,y)} \right], \tag{5}$$

using the same technique for VAE [11, 18] but with a recognition network $q_\phi(z|x,y) = \mathcal{N}\left(z|\mu_\phi(x,y), \text{diag}\big(\sigma^2_\phi(x,y)\big)\right)$ taking both x and y as input.

While [21] demonstrated that CVAE can be applied to high-dimensional conditional density estimation, CVAE suffers from two limitations. First, CVAE cannot incorporate unlabeled data. Second, CVAE does not learn the distribution of its input and is thus more susceptible to over-fitting. To resolve these limitations, we present a new approach to conditional density estimation.

3 Neural bottleneck conditional density estimation

We provide a high-level overview of our approach (Fig. 1) which consists of a new architecture and a new training procedure. Our new architecture imposes a bottleneck constraint, resulting in a class of conditional density estimators that we call *bottleneck conditional density estimators* (BCDEs). Unlike the CVAE, the BCDE generative path prevents x from directly influencing y. Following the conditional training paradigm in [21], conditional/discriminative training of the BCDE means maximizing the lower bound of a conditional likelihood similar to Eq. (5),

$$\ln p_\theta(y|x) \geq \mathcal{C}(\theta, \phi; x, y) = \mathbb{E}_{q_\phi(z|x,y)} \left[\ln \frac{p_\theta(z|x) p_\theta(y|z)}{q_\phi(z|x,y)} \right]. \tag{6}$$

When trained over a dataset L of paired (x, y) samples, the overall conditional training objective is

$$\mathcal{C}(\theta, \phi) = \sum_{x,y \in L} \mathcal{C}(\theta, \phi; x, y). \tag{7}$$

However, this approach will suffer from the same limitations as CVAE in addition imposing a bottleneck that limits the flexibility of the generative model. Instead, we propose a *hybrid* joint and conditional training regime that takes advantage of the bottleneck architecture to avoid over-fitting and support semi-supervision.

One component in our hybrid training procedure tackles the problem of estimating the *joint* density $p(x, y)$. To do this, we use the joint counterpart of the BCDE: the bottleneck joint density estimator (BJDE). Unlike conditional models, the BJDE allows us to incorporate unpaired x and y data during training. Thus, the BJDE can be trained in a semi-supervised fashion. We will also show that the BJDE is well-suited for *factored inference* (§3.2)—a factorization procedure that makes the parameter space of the recognition model more compact.

The BJDE also serves as a way to regularize the BCDE, where the regularization constraint can be viewed as soft tying between the parameters of these two models' generative and recognition networks. It is via this regularization that the BCDE can benefit from unpaired x and y for conditional density estimation.

3.1 Bottleneck joint density estimation

In the BJDE, we wish to learn the joint distribution of x and y. The bottleneck is introduced in the generative path via the bottleneck variable z, which points to x and y (Figs. 2a to 2c). Thus, the variational lower bound of the joint likelihood is

$$\ln p_{\theta'}(x, y) \geq \mathcal{J}_{xy}(\theta', \phi'; x, y) = \mathbb{E}_{q_{\phi'}(z|x,y)} \left[\ln \frac{p(z) p_{\theta'}(x|z) p_{\theta'}(y|z)}{q_{\phi'}(z|x, y)} \right]. \tag{8}$$

We use $\{\theta', \phi'\}$ to indicate the parameters of the BJDE networks and reserve $\{\theta, \phi\}$ for the BCDE parameters. For samples where x or y is unobserved, we will need to compute the variational lower bound for the marginal likelihoods. Here, the bottleneck plays a critical role. If x were to directly influence y, any attempt to incorporate unlabeled y would require the recognition model to infer the unobserved x from the observed y—a conditional density estimation problem which may be as hard as our original task. In the bottleneck architecture, the conditional independence of x and y given z implies that only the low-dimensional bottleneck needs to be marginalized. Thus the usual variational lower bounds for the marginal likelihoods yield

$$\ln p_{\theta'}(x) \geq \mathcal{J}_x(\theta', \phi'; x) = \mathbb{E}_{q_{\phi'}(z|x)} \left[\ln \frac{p(z) p_{\theta'}(x|z)}{q_{\phi'}(z|x)} \right], \tag{9}$$

$$\ln p_{\theta'}(y) \geq \mathcal{J}_y(\theta', \phi'; y) = \mathbb{E}_{q_{\phi'}(z|y)} \left[\ln \frac{p(z) p_{\theta'}(y|z)}{q_{\phi'}(z|y)} \right]. \tag{10}$$

Since z takes on the task of reconstructing both x and y, the BJDE is sensitive to the distributions of x and y and learns a joint manifold over the two data sources. BJDE thus provides the following benefits: **1**) learning the distribution of x makes the inference of z given x robust to perturbations in the input, **2**) z becomes a joint-embedding of x and y, **3**) the model can leverage unlabeled data. Overall, when the observed paired and unpaired samples are i.i.d., the joint training objectives is,

$$\mathcal{J}(\theta', \phi') = \sum_{x,y \in L} \mathcal{J}_{xy}(\theta', \phi'; x, y) + \sum_{x \in U_x} \mathcal{J}_x(\theta', \phi'; x) + \sum_{y \in U_y} \mathcal{J}_y(\theta', \phi'; y),, \tag{11}$$

where L is a dataset of paired (x, y) samples, and U_x and U_y are data sets of unpaired samples.

3.2 Factored inference

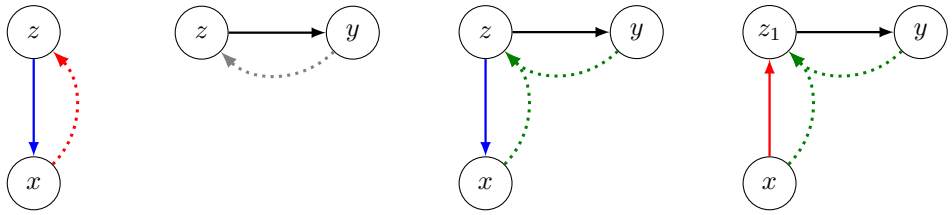
When training the BJDE in the semi-supervised regime, we introduce a factored inference procedure that reduce the number of parameters in the recognition model.

In the semi-supervised regime, the 1-layer BJDE recognition model requires approximating three posteriors: $p(z|x, y) \propto p(z)p(x, y|z)$, $p(z|x) \propto p(z)p(x|z)$, and $p(z|y) \propto p(z)p(y|z)$. The standard approach would be to assign one recognition network for each approximate posterior. This approach, however, does not take advantage of the fact that these posteriors share the same likelihood functions, i.e., $p(x, y|z) = p(x|z)p(y|z)$.

Rather than learning the three approximate posteriors independently, we propose to learn the approximate likelihood functions $\hat{\ell}(z; x) \approx p(x|z)$, $\hat{\ell}(z; y) \approx p(y|z)$ and let $\hat{\ell}(z; x, y) = \hat{\ell}(z; x)\hat{\ell}(z; y)$. Consequently, this factorization of the recognition model enables parameter sharing within the joint recognition model (which is beneficial for semi-supervised learning) and eliminates the need for constructing a neural network that takes both x and y as inputs. The latter property is especially useful when learning a joint model over multiple, heterogeneous data types (e.g. image, text, and audio).

In practice, we directly learn recognition networks for $q(z|x)$ and $\hat{\ell}(z; y)$ and perform factored inference as follows

$$q(z|x, y) \propto q_{\phi'}(z|x)\hat{\ell}_{\phi'}(z; y), \; q(z|y) \propto p(z)\hat{\ell}_{\phi'}(z; y), \tag{12}$$



(a) Joint: (x_u) (b) Joint: (y_u) (c) Joint: (x_l, y_l) (d) Conditional: (x_l, y_l)

Figure 2: The joint and conditional components of a 1-layer BCDE. Dotted lines represent recognition models. Colors indicate the parameters that are strictly tied within the joint model and softly tied between the joint and conditional models.

Standard	BJDE:	$q_{\phi'}(z\|x,y)$	$q_{\phi'}(z\|y)$	$q_{\phi'}(z\|x)$	$p_{\theta'}(y\|z)$	$p_{\theta'}(x\|z)$
	BCDE:	$q_{\phi}(z\|x,y)$	–	$p_{\theta}(z\|x)$	$p_{\theta}(y\|z)$	–
Factored	BJDE:	–	$\hat{\ell}_{\phi'}(z;y)$	$q_{\phi'}(z\|x)$	$p_{\theta'}(y\|z)$	$p_{\theta'}(x\|z)$
	BCDE:	–	$\hat{\ell}_{\phi'}(z;y)$	$p_{\theta}(z\|x)$	$p_{\theta}(y\|z)$	–

Table 1: Soft parameter tying between the BJDE and BCDE. For each network within the BCDE, there is a corresponding network within the BJDE. We show the network correspondences with and without the application of factored inference. We regularize all BCDE networks to their corresponding BJDE network parameters.

where ϕ' parameterizes the recognition networks. To ensure proper normalization in Eq. (12), it is sufficient for $\hat{\ell}$ to be bounded. If the prior $p(z)$ belongs to an exponential family with sufficient statistics $T(z)$, we can parameterize $\hat{\ell}_{\phi'}(z;y) = \exp\left(\langle T(z), \eta_{\phi'}(y)\rangle\right)$, where $\eta_{\phi'}(y)$ is a network such that $\eta_{\phi'}(y) \in \{\eta | \{\langle T(z), \eta\rangle \ \ \forall z\}$ is upper bounded$\}$. Then the approximate posterior can be obtained by simple addition in the natural parameter space of the corresponding exponential family. When the prior and approximate likelihood are both Gaussians, this is exactly precision-weighted merging of the means and variances [9].

Factored inference offers a more compact recognition model at the cost of lower flexibility. We observe that when the supervised signal is low relative to the unsupervised signal, the benefits of a compact recognition model significantly outweigh its limitations.

3.3 Hybrid training

While the BJDE can be used directly for conditional density estimation, it is not expected to yield good performance. For classification, it has been observed that a generative model trained to estimate the joint distribution may yield sub-optimal performance when compared to a model that was trained discriminatively [15]. Indeed, both [12] and [14] incorporated additional discriminative training into their objective functions in order to successfully perform semi-supervised classification with deep generative models. The necessity of additional discriminative training is attributable to the joint model being mis-specified [13].

In other words, when the model is mis-specified, we should not expect the optimal joint model parameters to coincide with the optimal conditional model parameters. To address this, [13] proposed a principled hybrid blending of the joint and conditional models. At its core, [13]'s hybrid blending procedure regularizes the parameters θ of the conditional model toward the parameters θ' of the joint model by introducing a prior that softly ties θ' to θ. However, [13] considers models that only contain generative parameters.

We propose a variant of hybrid blending that extends to variational methods for conditional density estimation. Variational methods involve both generative and recognition networks. When blending is applied to models that contain both generative θ and recognition ϕ parameters, it may not be immediately obvious *a priori* how the conditional model parameters $\{\theta, \phi\}$ should be tied to the joint model parameters $\{\theta', \phi'\}$. We argue that it is reasonable to impose a prior that also ties some

elements of BCDE's θ to BJDE's ϕ'. In particular, since BJDE's $q_{\phi'}(z|x)$ and BCDE's $p_\theta(z|x)$ both serve the same function of projecting x into the space of z, it is reasonable to impose a prior that regularizes the BCDE generative network $p_\theta(z|x)$ toward the BJDE recognition network $q_{\phi'}(z|x)$. We thus propose the following hybrid objective function,

$$\mathcal{H}(\theta, \phi, \theta', \phi') = \mathcal{C}(\theta, \phi) + \mathcal{J}(\theta', \phi') - \Lambda(\theta, \phi, \theta', \phi'). \tag{13}$$

where Λ regularizes the BCDE parameters (θ, ϕ) toward the the BJDE parameters (θ', ϕ'). Details about how $\{\theta, \phi\}$ is tied to $\{\theta', \phi'\}$ are shown in Figure 2 and Table 1. When factored inference is applied, Table 1 shows additional parameter tying between BCDE and BJDE. Note that Λ regularization can be achieved in a variety of ways. In our experiments, we implement Λ regularization by initializing the conditional model with the joint model parameters and performing early-stopping. Note that this approach introduces one-way regularization of the BCDE by the BJDE. We are currently exploring alternative approaches that allow for two-way regularization.

4 Experiments

We evaluated the performance of our hybrid training procedure on the permutation-invariant MNIST quadrant prediction task [20, 21]. The MNIST quadrant prediction task is a conditional density estimation task where MNIST digits are partially occluded. The model is given the observed region and is evaluated by its perplexity on the occluded region. The MNIST quadrant prediction task consists of three sub-tasks depending on the degree of partial observability. 1-quadrant prediction: the bottom left quadrant is observed. 2-quadrant prediction: the left side is observed. 3-quadrant prediction: the bottom right quadrant is *not* observed.

In the fully-supervised case, the original MNIST training set $\{x'_i\}_{i=1}^{50000}$ is converted into our CDE training set $L = \{x_i, y_i\}_{i=1}^{50000}$ by splitting each image into its observed x and unobserved y regions. Note that the training set does not contain the original digit-class label information. In the n_l-label semi-supervised case, we randomly sub-sampled n_l pairs evenly across all 10 digits to create our labeled training set $L = \{x_i, y_i\}_{i=1}^{n_l}$. The remaining n_u paired samples are decoupled and put into our unlabeled training sets $U_x = \{x_i\}_{i=1}^{n_u}, U_y = \{y_i\}_{i=1}^{n_u}$. The MNIST digits are statically binarized by sampling from the Bernoulli distribution according their pixel values [19]. While the original validation set provides 10000 paired (x, y) samples, we always keep our labeled validation set at one-fifth the size of the labeled training set (n_l). The labeled test set size is always kept at 10000. The reported performances are averaged scores over multiple runs along with the standard error.

When training our models, we optimized the various training objectives with *Adam* [10]. Although our training objective is based on the variational lower bound, our performance scoring metric is the negative conditional log-likelihood score which we approximate using importance sampling with 50 samples [21]. The initial learning rate is set to 0.001 and dropped by a factor of 10 when the validation score plateaus. Training was terminated with early-stopping when the validation score can no longer be improved using a given training objective. We use multi-layered perceptrons (MLPs) for all neural networks in the BCDE. All MLPs are batch-normalized [8] and parameterized with two hidden layers of 500 rectified linear units. All stochastic layers have 50 hidden units. The models were implemented in Python[3] using Theano [22] and Keras [4].

4.1 Conditional log-likelihood performance

Tables 2 to 4 show the performance comparisons between CVAE, hybridly-trained BCDE, and baseline variants of Eq. (13): conditional training ($\mathcal{C}$ only) and naïve pre-training (without J_{xy}). We also evaluate the performance of the BCDE models when factored inference is applied (Table 1), as well as the performances of 2-stochastic-layer BCDE models. The 2-L BCDE has the generative model $p(z_{1:2}, y|x) = p_\theta(z_1|x)p_\theta(z_2|z_1)p_\theta(y|z_2)$, and its 2-L BJDE counterpart has the generative model $p(z_{1:2}, x, y) = p_{\theta'}(z_1)p_{\theta'}(x|z_1)p_{\theta'}(z_2|z_1)p_{\theta'}(y|z_2)$. To deal with the multi-layer stochasticity, the BCDE and BJDE are trained using top-down inference [9].

In the fully-supervised regime, hybrid-trained BCDE achieves the best performance, significantly improves upon its conditionally-trained counterpart as well as previously reported result for

[3] github.com/ruishu/bcde

Models	$n_l = 50000$	$n_l = 25000$	$n_l = 10000$	$n_l = 5000$
CVAE [21]	63.91	-	-	-
1-L BCDE (conditional)	64.58 ± 0.05	66.90 ± 0.09	71.50 ± 0.04	77.28 ± 0.11
1-L BCDE (conditional + factored)	65.62 ± 0.06	68.88 ± 0.06	75.84 ± 0.09	82.08 ± 0.29
1-L BCDE (naïve pre-train)	63.95 ± 0.03	65.34 ± 0.05	67.40 ± 0.04	70.03 ± 0.15
1-L BCDE (hybrid)	$\mathbf{63.64 \pm 0.06}$	65.01 ± 0.08	66.91 ± 0.06	68.93 ± 0.20
1-L BCDE (hybrid + factored)	63.71 ± 0.05	$\mathbf{64.84 \pm 0.05}$	$\mathbf{66.23 \pm 0.05}$	$\mathbf{67.64 \pm 0.03}$
2-L BCDE (conditional)	62.45 ± 0.06	64.67 ± 0.06	68.33 ± 0.09	72.17 ± 0.08
2-L BCDE (conditional + factored)	63.85 ± 0.06	66.75 ± 0.09	72.91 ± 0.09	79.15 ± 0.25
2-L BCDE (naïve pre-train)	62.21 ± 0.02	63.60 ± 0.03	65.52 ± 0.11	67.49 ± 0.03
2-L BCDE (hybrid)	$\mathbf{61.79 \pm 0.02}$	63.12 ± 0.01	64.79 ± 0.10	66.35 ± 0.15
2-L BCDE (hybrid + factored)	61.84 ± 0.03	$\mathbf{62.95 \pm 0.06}$	$\mathbf{64.31 \pm 0.07}$	$\mathbf{65.46 \pm 0.12}$

Table 2: MNIST quadrant prediction task: 1-quadrant.

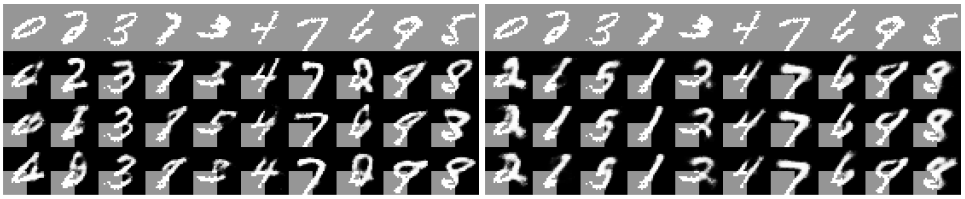
Models	$n_l = 50000$	$n_l = 25000$	$n_l = 10000$	$n_l = 5000$
CVAE [21]	44.73	-	-	-
1-L BCDE (conditional)	45.26 ± 0.01	47.03 ± 0.09	50.57 ± 0.11	54.17 ± 0.11
1-L BCDE (conditional + factored)	46.46 ± 0.09	49.32 ± 0.10	54.08 ± 0.19	58.33 ± 0.09
1-L BCDE (naïve pre-train)	44.59 ± 0.05	45.72 ± 0.07	47.46 ± 0.11	49.34 ± 0.08
1-L BCDE (hybrid)	$\mathbf{44.34 \pm 0.04}$	$\mathbf{45.36 \pm 0.05}$	46.94 ± 0.02	48.54 ± 0.09
1-L BCDE (hybrid + factored)	44.51 ± 0.02	45.40 ± 0.06	$\mathbf{46.62 \pm 0.05}$	$\mathbf{47.68 \pm 0.05}$
2-L BCDE (conditional)	43.83 ± 0.07	45.42 ± 0.03	48.18 ± 0.07	50.61 ± 0.08
2-L BCDE (conditional + factored)	45.51 ± 0.06	47.93 ± 0.04	52.47 ± 0.20	56.42 ± 0.08
2-L BCDE (naïve pre-train)	43.67 ± 0.03	44.56 ± 0.03	46.12 ± 0.05	47.77 ± 0.05
2-L BCDE (hybrid)	$\mathbf{43.13 \pm 0.04}$	$\mathbf{44.07 \pm 0.02}$	45.42 ± 0.02	46.79 ± 0.07
2-L BCDE (hybrid + factored)	43.48 ± 0.02	44.15 ± 0.05	$\mathbf{45.26 \pm 0.04}$	$\mathbf{46.07 \pm 0.07}$

Table 3: MNIST quadrant prediction task: 2-quadrants.

Models	$n_l = 50000$	$n_l = 25000$	$n_l = 10000$	$n_l = 5000$
CVAE [21]	20.95	-	-	-
1-L BCDE (conditional)	20.91 ± 0.01	21.75 ± 0.05	23.37 ± 0.09	24.83 ± 0.05
1-L BCDE (conditional + factored)	22.04 ± 0.03	23.21 ± 0.04	25.10 ± 0.04	27.05 ± 0.05
1-L BCDE (naïve pre-train)	20.65 ± 0.03	21.20 ± 0.06	22.22 ± 0.09	22.99 ± 0.08
1-L BCDE (hybrid)	$\mathbf{20.43 \pm 0.01}$	$\mathbf{20.93 \pm 0.03}$	21.80 ± 0.06	22.58 ± 0.08
1-L BCDE (hybrid + factored)	20.78 ± 0.04	21.10 ± 0.02	$\mathbf{21.77 \pm 0.02}$	$\mathbf{22.49 \pm 0.11}$
2-L BCDE (conditional)	20.52 ± 0.02	21.05 ± 0.02	22.26 ± 0.05	23.48 ± 0.02
2-L BCDE (conditional + factored)	21.64 ± 0.04	22.90 ± 0.10	25.03 ± 0.11	26.49 ± 0.08
2-L BCDE (naïve pre-train)	20.45 ± 0.03	20.98 ± 0.01	21.84 ± 0.03	22.70 ± 0.06
2-L BCDE (hybrid)	$\mathbf{19.94 \pm 0.01}$	$\mathbf{20.40 \pm 0.05}$	21.16 ± 0.03	21.81 ± 0.03
2-L BCDE (hybrid + factored)	20.34 ± 0.02	20.60 ± 0.06	$\mathbf{21.12 \pm 0.02}$	$\mathbf{21.74 \pm 0.04}$

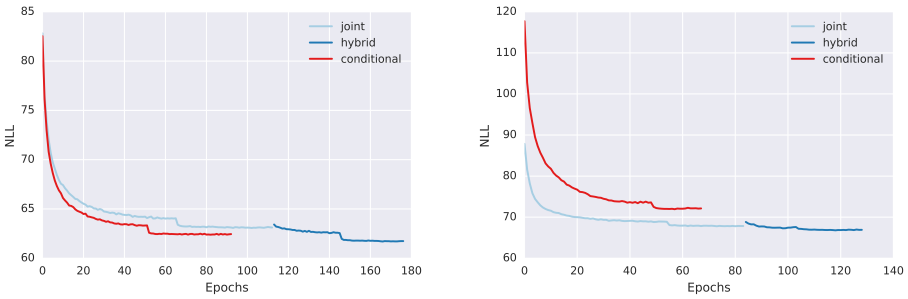
Table 4: MNIST quadrant prediction task: 3-quadrants. We report the test set negative conditional log-likelihood scores for the MNIST quadrant prediction task [20]. In addition to the baseline (CVAE), we perform an ablation study of the BCDE objective Fig. 13 and consider its variants. We perform the test in both fully-supervised ($n_l = 50000$) and semi-supervised ($n_l < 50000$) regimes, where n_l is the number of paired training points. For both the 1-layer and 2-layer BCDEs, the averaged performances over multiple runs and standard errors are reported.

CVAE [21]. In the semi-supervised regime, hybrid-trained BCDE continues to significantly outperform conditionally-trained BCDE. As the labeled training size n_l reduces, the benefits of having a more compact recognition model becomes more apparent; hybrid + factored inference achieves the best performance, out-performing hybrid on the $n_l = 5000$ 1-quadrant task by as much as 1 nat—a considerable margin [25]. Conditionally-trained BCDE performs very poorly in the semi-supervised tasks, likely due to over-fitting issues. The 2-layer BCDE generally outperforms 1-layer BCDE due to having a more expressive variational family [9, 17].



(a) Hybrid (b) Conditional

Figure 3: Comparison of conditional image generation for conditional versus hybrid 2-layer BCDE on the semi-supervised 1-quadrant task (3a-3b). Row 1 shows the original images. Rows 2-4 show three attempts by each model to sample y according to x (the bottom-left quadrant, indicated in gray). Hybrid training yields a higher-entropy conditional generative model that has lower perplexity.



(a) Fully-supervised (b) Semi-supervised

Figure 4: Performance on validation set versus epochs for the hybrid and conditional models. Since hybrid training is implemented by training on the joint objective followed by fine-tuning on the conditional objective, we color-code the joint training duration in a lighter color. We show plots for both the fully-supervised case ($n_l = 50000$) as well as the semi-supervised case ($n_l = 5000$).

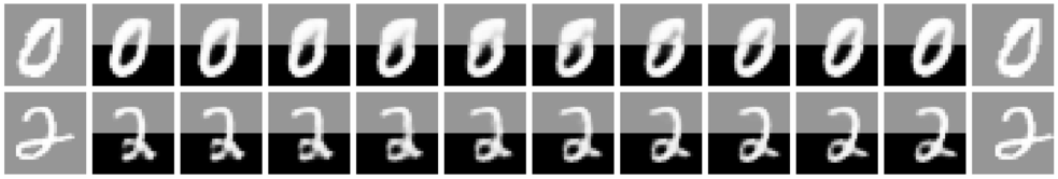
Conditionally generated samples from the hybrid and conditional models in Figure 3 shows that the models with lower perplexity tend to produce high-entropy conditional image generators that spreads the conditional probability mass over the target output space [23].

To investigate the behavior of regularization using the BJDE, we show an example of the validated negative conditional likelihood learning curve during training (Fig. 4) of conditional BCDE and hybrid BCDE. For hybrid BCDE, the curve is divided into two phases: joint (pre-training with Eq. (11)), and hybrid (fine-tuning with Eq. (7)). In the fully-supervised regime, model mis-specification results in BJDE initially being worse than conditionally-trained BCDE. In the semi-supervised regime, however, the benefit of incorporating unpaired data enables BJDE to outperforms BCDE even in the conditional task. In both cases, hybrid training consistently achieves the best performance.

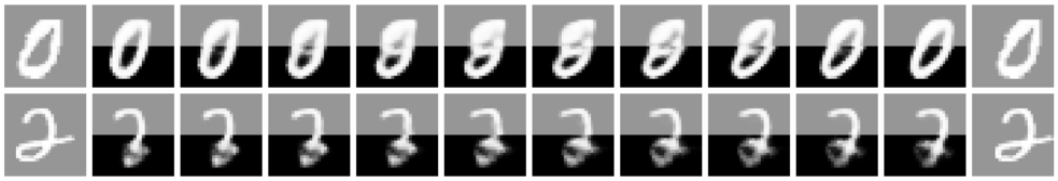
4.2 Conditional latent space walk

The 2-layer BJDE, by design, learns to use z_1 as a joint manifold for x and y. Hybrid training should ideally preserve z_1 as a joint manifold. To evaluate if this is the case, we propose a variant of the MNIST quadrant task called the *shift-sensitive top-bottom task*. In this task, the objective is to learn to predict the the bottom half of the MNIST digit (y) when given the top half (x). However, we introduce structural perturbation to both the top and bottom halves of the image in our training, validation, and test sets by randomly shifting each pair (x, y) horizontally by the same number of pixels (shift varies between $\{-4, -3, \ldots, 3, 4\}$). We then train the 2-layer BCDE using either the conditional or hybrid objective in the fully-supervised regime.

Latent space interpolation is a popular approach for evaluating the manifold structure of the latent space [16, 3, 1]. To evaluate whether z_1 is a manifold of y in the BCDE, we perform a conditional latent space walk as follows. Two x's are selected and deterministically projected onto the z_1 latent space using the mean of $p_\theta(z_1|x)$. Then, we linearly interpolate between the two latent representations,



(a) Hybrid



(b) Conditional

Figure 5: Comparison of interpolation of y for hybrid versus conditional 2-layer BCDE on the fully-supervised top-down task with full-image shift (5a-5b). This task was chosen for easy visual evaluation. We take some x (the top half of an image) and interpolate between x shifted two pixels left and right. Interpolation was performed in the space of z_1 and the resulting y was generated. To facilitate comparison, the interpolation of x was created using a VAE and is shown in gray

and deterministically decode the interpolated representations in the space of y using the means of $p_\theta(z_2|z_1)$ and then $p_\theta(y|z_2)$. In Figure 5, we chose to interpolate between an x shifted two pixels to the left and the same x shifted two pixels to the right. This setup was chosen for easy visual evaluation. Because hybrid training regularizes the BCDE toward the BJDE, hybridly-trained BCDE retains the manifold properties of the BJDE, producing meaningful bottom-half digits during z_1 latent space interpolation. In comparison, the conditionally-trained BCDE tends to produce illegible bottom-half digits, suggesting its failure in learning a joint manifold.

4.3 Robustness of representation

Models	Without shift	With shift	Difference
2-L BCDE (conditional)	41.57 ± 0.03	42.95 ± 0.04	1.38
2-L BCDE (hybrid)	40.70 ± 0.01	41.81 ± 0.03	1.11
2-L BCDE (factored + hybrid)	40.99 ± 0.04	41.89 ± 0.02	0.90

Table 5: Shift-invariant top-bottom prediction. Performance with and without structural corruption reported, along with the performance difference. Hybrid training makes the model more robust against corruption in x.

Since hybrid training makes BCDE aware of the distribution of x, it enables the model to be robust against corruption of x. To demonstrate this, we investigate another variant of the MNIST quadrant task called the *shift-invariant top-bottom* task. This task is similar to the shift-sensitive top-bottom task, but with one key difference: we *only* introduce structural noise to the top half of the image in our training, validation, and test sets. The goal is thus to learn that the prediction of y (which is always centered) is invariant to the shifted position of x. To eliminate the benefit of unlabeled data, we only perform this task in the fully-supervised regime.

Table 5 shows that hybrid training makes the BCDE performance more robust against structural corruption in x. Because of its more compact recognition model, factored + hybrid is less vulnerable to over-fitting, resulting in a smaller performance gap between performance on the corrupted and uncorrupted data.

To understand why hybrid training makes the BCDE robust against corruption, we compare the latent space representations of x for hybrid versus conditional (Fig. 6). We randomly select two top halves of the digit "3" and shifted each image horizontally between -4 and +4 pixels, for a total of two sets of nine x's. We deterministically projected each resulting x into the space of z_1

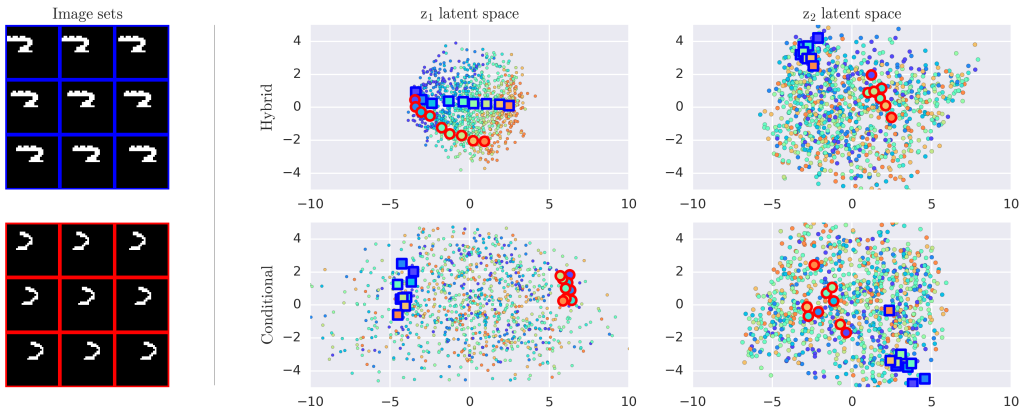


Figure 6: Evaluation of shift-invariance in the latent space of hybrid and conditionally-trained 2-layer BCDEs. PCA plots of the latent space subregion for all x's whose class label $= 3$ are shown. Two sets of images (demarcated by line color) are projected into the z_1 and z_2 latent spaces by both models. Fill color indicates the degree of shift: blue $= -4$, orange $= +4$.

and z_2 using the learned mean networks, in a manner similar to the latent space walk experiment. For easy visualization, we show the PCA projections of the latent space sub-region populated by the projections of all digits 3 and color-coded all points based on the degree of shifting. The plot for hybrid-BCDE's z_1 representation of x shows that the model learns to untangle shift from other features. This in turn helps the hybrid-BCDE learns a shift-invariant z_2 (which only has to reconstruct the un-shifted y), possibly by collapsing the axis along which z_1 was shift-sensitive. In contrast, the conditionally-trained BCDE uses both z_1 and z_2 to learn only a representation of y. Thus, its learned representations are not aware of the systematic shift in x, and do not yield a shift-invariant z_2.

5 Conclusion

We presented the BCDE, a new framework for high-dimensional conditional density estimation where multiple layers of stochastic neural networks are used as the bottleneck between the input and the output data. To train the BCDE, we proposed a new hybrid training procedure where the BCDE is regularized towards its joint counterpart, the BJDE. The bottleneck constraint implies that only the bottleneck needs to be marginalized when missing either the input x or the output y during training, thus enabling the joint model to be trained in a semi-supervised fashion. The regularization effect between the conditional and the joint model in our hybrid training procedure helps the BCDE, itself a conditional model, to become more robust and can learn from unpaired data in semi-supervised setting. To reduce the complexity of the recognition models of the joint and the conditional models during hybrid training, we introduced factored inference, a technique that leads to more parameter sharing among the recognition networks.

Our experiments showed that the hybridly-trained BCDE established new benchmark performances on the MNIST quadrant prediction task in both the fully-superivsed and semi-supervised regime. Hybrid training enables the BCDE to learn from unpaired data, which significantly improves performance in the semi-supervised regime. When the supervisory signal is very weak, factored inference prevents over-fitting by additionally tying parameters within the BJDE recognition networks. To understand the BCDE's strong performance in the fully-supervised regime, we showed that hybrid training transfers the joint embedding properties of the BJDE to the BCDE, allowing the BCDE to learn better representations of both the input x and output y. By learning a better representation of x, the BCDE also becomes robust to perturbations in the input.

The success of the BCDE hybrid training framework makes it a prime candidate for other high-dimensional conditional density estimation tasks, especially in semi-supervised settings. The advantages of a more compact model when using factored inference also merits further consideration; an interesting line of future work will be to apply factored inference to joint density estimation tasks which involve multiple, heterogeneous data sources.

References

[1] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio. Generating Sentences from a Continuous Space. *arXiv:1511.06349*, 2015.

[2] P. Dayan, G. Hinton, R. Neal, and R. Zemel. The Helmholtz Machine. *Neural computation*, 1995.

[3] L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using Real NVP. *arXiv:1605.08803*, 2016.

[4] C. François. Keras. `https://github.com/fchollet/keras`, 2016.

[5] K. Ganchev, J. Graca, J. Gillenwater, and B. Taskar. Posterior regularization for structured latent variable models. *JMLR*, 2010.

[6] G. Hinton, P. Dayan, B. Frey, and R. Radford. The "wake-sleep" algorithm for unsupervised neural networks. *Science*, 1995.

[7] M. P. Holmes, A. G. Gray, and C. L. Isbell. Fast Nonparametric Conditional Density Estimation. *arXiv:1206.5278*, 2012.

[8] S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv:1502.03167*, 2015.

[9] C. Kaae Sønderby, T. Raiko, L. Maaløe, S. Kaae Sønderby, and O. Winther. Ladder Variational Autoencoders. *arXiv:1602.02282*, 2016.

[10] D. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980*, 2014.

[11] D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. *arXiv:1312.6114*, 2013.

[12] D. P. Kingma, D. J. Rezende, S. Mohamed, and M. Welling. Semi-Supervised Learning with Deep Generative Models. *arXiv:1406.5298*, 2014.

[13] J. Lasserre, C. Bishop, and T. Minka. Principled hybrids of generative and discriminative models. In *The IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[14] L. Maaløe, C. Kaae Sønderby, S. Kaae Sønderby, and O. Winther. Auxiliary Deep Generative Models. *arXiv:1602.05473*, 2016.

[15] A. Ng and M. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *Neural Information Processing Systems*, 2002.

[16] A. Radford, L. Metz, and S. Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *arXiv:1511.06434*, 2015.

[17] R. Ranganath, D. Tran, and D. M. Blei. Hierarchical Variational Models. *ArXiv e-prints*, 1511.02386, Nov. 2015.

[18] D. Rezende, S. Mohamed, and D. Wierstra. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. *ArXiv e-prints*, 1401.4082, Jan. 2014.

[19] R. Salakhutdinov and I. Murray. On the quantitative analysis of deep belief networks. *International Conference on Machine Learning*, 2008.

[20] K. Sohn, W. Shang, and L. H. Improved multimodal deep learning with variation of information. *Neural Information Processing Systems*, 2014.

[21] K. Sohn, X. Yan, and H. Lee. Learning structured output representation using deep conditional generative models. *Neural Information Processing Systems*, 2015.

[22] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv:1605.02688*, 2016.

[23] L. Theis, A. van den Oord, and M. Bethge. A note on the evaluation of generative models. *arXiv:1511.01844*, 2015.

[24] M. Tipping and C. Bishop. Probabilistic Principal Component Analysis. *J. R. Statist. Soc. B*, 1999.

[25] Y. Wu, Y. Burda, R. Salakhutdinov, and R. Grosse. On the Quantitative Analysis of Decoder-Based Generative Models. *arXiv:1611.04273*, 2016.

[26] X. Yan, J. Yang, K. Sohn, and H. Lee. Attribute2Image: Conditional Image Generation from Visual Attributes. *arXiv:1512.00570*, 2015.

[27] B. Zhang, D. Xiong, J. Su, H. Duan, and M. Zhang. Variational Neural Machine Translation. *arXiv:1605.07869*, 2016.