Bottleneck Conditional Density Estimation

Rui Shu [1] **Hung H. Bui** [2] **Mohammad Ghavamzadeh** [3]

Abstract

We introduce a new framework for training deep generative models 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. Crucially, we propose a new hybrid training method that blends the conditional generative model with a joint generative model. Hybrid blending is the key to effective training of the BCDE, which avoids overfitting and provides a novel mechanism for leveraging unlabeled data. We show that our hybrid training procedure enables models to achieve competitive results in the MNIST quadrant prediction task in the fully-supervised setting, and sets new benchmarks in the semi-supervised regime for MNIST, SVHN, and CelebA.

1. Introduction

Conditional density estimation (CDE) refers to the problem of estimating a conditional density $p(y|x)$ for the input x and target 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), an important task the conditional distribution has multiple modes. CDE problems in which both x and y are high-dimensional have a wide range of important applications, including video prediction, cross-modality prediction (e.g. image-to-caption), model estimation in model-based reinforcement learning, and so on.

[1]Stanford University [2]Adobe Research [3]DeepMind (The work was done when all the authors were with Adobe Research). Correspondence to: Rui Shu <ruishu@stanford.edu>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017.

Classical non-parametric conditional density estimators typically rely on local Euclidean distance in the original input and target spaces (Holmes et al., 2012). This approach quickly becomes ineffective in high-dimensions from both computational and statistical points 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) (Sohn et al., 2015). CVAEs have been applied to a variety of problems, such as MNIST quadrant prediction, segmentation (Sohn et al., 2015), attribute-based image generation (Yan et al., 2015), and machine translation (Zhang et al., 2016).

But CVAEs suffer from two statistical deficiencies. First, they 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 low-dimensional representation (such as a low-dimensional manifold) of the data, recovering this structure is important, even if the task at hand is to learn the conditional density $p(y|x)$. Otherwise, the model is susceptible to overfitting. Second, for many CDE 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.[1] We note that while variational methods (Kingma & Welling, 2013; Rezende et al., 2014) have been applied to semi-supervised classification (where y is a class label) (Kingma et al., 2014; Maaløe et al., 2016), 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 influences the target y via layers of bottleneck stochastic variables $z = \{z_i\}$ in the 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. Motivated by Lasserre et al. (2006), 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.

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

Our BCDE hybrid training framework is a novel approach for leveraging unlabeled data for conditional density estimation. Using our BCDE hybrid training framework, we establish new benchmarks for the quadrant prediction task (Sohn et al., 2015) in the semi-supervised regime for MNIST, SVHN, and CelebA. Our experiments show that **1)** hybrid training is competitive for fully-supervised CDE, **2)** in semi-supervised CDE, hybrid training helps to avoid overfitting, performs significantly better than conditional training with unlabeled data pre-training, and achieves state-of-the-art results, and **3)** hybrid training encourages the model to learn better and more robust representations.

2. Background

2.1. Variational Autoencoders

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 an observable vector x. The observation is usually conditionally Gaussian $x|z \sim \mathcal{N}\big(\mu_\theta(z), \text{diag}(\sigma^2_\theta(z))\big)$, where μ and σ^2 are neural networks whose parameters are represented by θ.[2] VAE can be seen as a non-linear generalization of the probabilistic PCA (Tipping & Bishop, 1999), and thus, can recover non-linear manifolds in the data. However, VAE's flexibility makes posterior inference of the latent variables 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 $p_\theta(z|x)$. Learning in VAE's 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 (Kingma & Welling, 2013), i.e.,

$$\sup_{\theta,\phi} \mathbb{E}_{q_\phi(z|x)}\big[\ln p_\theta(x|z)\big] - \mathrm{D}_{\mathrm{KL}}\big(q_\phi(z|x) \,||\, p(z)\big). \quad (1)$$

We note that the objective Eq. (1) can be rewritten in the following form that exposes its connection to the variational lower bound of the log-likelihood

$$\sup_\theta \Big(\ln p_\theta(x) - \inf_\phi \mathrm{D}_{\mathrm{KL}}\big(q_\phi(z|x) \,||\, p_\theta(z|x)\big)\Big) = \sup_{\theta,\phi} \mathbb{E}_{q_\phi(z|x)}\left[\ln \frac{p_\theta(x,z)}{p_\phi(z|x)}\right]. \quad (2)$$

We make two remarks regarding the minimization of the term $\mathrm{D}_{\mathrm{KL}}\big(q_\phi(z|x) \,||\, p_\theta(z|x)\big)$ in Eq. 2. First, when $q(\cdot|\cdot)$ is a conditionally independent Gaussian, this approximation is at best as good as the mean-field approximation that minimizes $\mathrm{D}_{\mathrm{KL}}\big(q \;||\; p_\theta(z|x)\big)$ over all independent Gaussian q's. 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 (Dayan et al., 1995; Ganchev et al., 2010; Hinton et al., 1995). In practice, both θ and ϕ are jointly optimized in Eq. (1), and the reparameterization trick (Kingma & Welling, 2013) is used to transform the expectation over $z \sim q_\phi(z|x)$ into $\epsilon \sim \mathcal{N}(0, I_k)$; $z = \mu_\phi(x) + \text{diag}\big(\sigma^2_\phi(x)\big)\epsilon$, which leads to an easily obtained stochastic gradient.

[2] For discrete x, one can use a deep network to parameterize a Bernoulli or a discretized logistic distribution.

2.2. Conditional VAEs (CVAEs)

In Sohn et al. (2015), the authors introduce the conditional version of variational autoencoders. The conditional generative model is similar to VAE, except that the latent variable z and the observed vector y are both conditioned on the input x. The conditional generative path is

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

$$p_\theta(y \mid x, z) = \mathcal{N}\Big(y \mid \mu_{y,\theta}(x,z), \text{diag}\big(\sigma^2_{y,\theta}(x,z)\big)\Big), \quad (4)$$

and when we use a Bernoulli decoder is

$$p_\theta(y \mid x, z) = \text{Ber}\big(y \mid \mu_{y,\theta}(x,z)\big). \quad (5)$$

Here, 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

$$\ln p_\theta(y|x) \geq \mathbb{E}_{q_\phi(z|x,y)}\left[\ln \frac{p_\theta(z|x)p_\theta(y|x,z)}{q_\phi(z|x,y)}\right], \quad (6)$$

but with a recognition network $q_\phi(z|x,y)$, which is typically Gaussian $\mathcal{N}\Big(z|\mu_\phi(x,y), \text{diag}\big(\sigma^2_\phi(x,y)\big)\Big)$, and takes both x and y as input.

2.3. Blending Generative and Discriminative

It is well-known that a generative model may yield suboptimal performance when compared to the same model trained discriminatively (Ng & Jordan, 2002), a phenomenon attributable to the generative model being misspecified (Lasserre et al., 2006). However, generative models can easily handle unlabeled data in semi-supervised setting. This is the main motivation behind blending generative and discriminative models. Lasserre et al. (2006) proposed a principled method for hybrid blending by duplicating the parameter of the generative model into a discriminatively trained θ and a generatively trained $\tilde{\theta}$, i.e.,

$$p(\mathbf{X}_l, \mathbf{Y}_l, \mathbf{X}_u, \tilde{\theta}, \theta) = p(\tilde{\theta}, \theta)p(\mathbf{X}_u|\tilde{\theta})p(\mathbf{X}_l|\tilde{\theta})p(\mathbf{Y}_l|\mathbf{X}_l, \theta). \quad (7)$$

The discriminatively trained parameter θ is regularized toward the generatively trained parameter $\tilde{\theta}$ via a prior $p(\tilde{\theta}, \theta)$ that prefers small $\|\theta - \tilde{\theta}\|^2$. As a result, in addition to

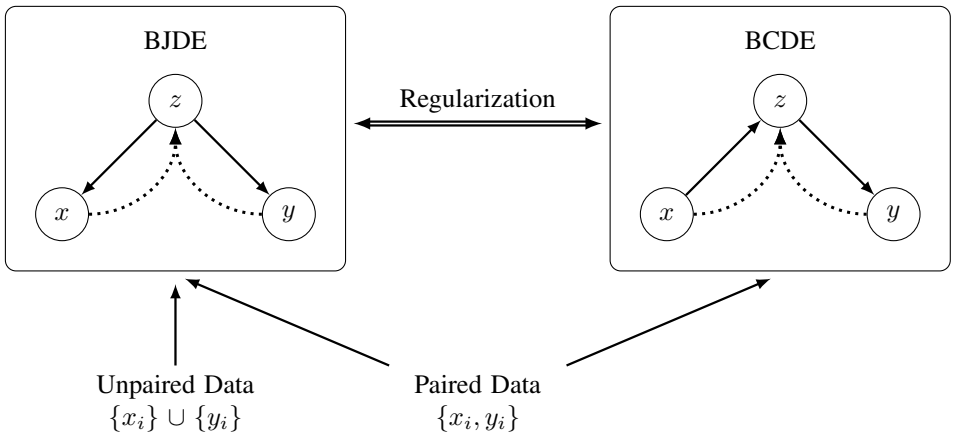


Figure 1. The hybrid training procedure that regularizes BCDE towards BJDE. This regularization enables the BCDE to indirectly leverage unpaired x and y for conditional density estimation.

learning from the labeled data $(\mathbf{X}_l, \mathbf{Y}_l)$, the discriminative parameter θ can be informed by the unlabeled data $\mathbf{X}_u$ via $\tilde{\theta}$, enabling a form of semi-supervised discriminatively trained generative model. However, this approach is limited to simple generative models (e.g., naive Bayes and HMMs), where exact inference of $p(y|x, \theta)$ is tractable.

3. Neural Bottleneck Conditional Density Estimation

While Sohn et al. (2015) has successfully applied the CVAE to CDE, CVAE suffers from two limitations. First, the CVAE does not learn the distribution of its input x, and thus, is far more susceptible to overfitting. Second, it cannot incorporate unlabeled data. To resolve these limitations, we propose a new approach to high-dimensional CDE that blends the discriminative model that learns the conditional distribution $p(y|x)$, with a generative model that learns the joint distribution $p(x, y)$.

3.1. Overview

Figure 1 provides a high-level overview of our approach that consists of a new architecture and a new training procedure. Our new architecture imposes a bottleneck constraint, resulting a class of conditional density estimators, we call it *bottleneck conditional density estimators* (BCDEs). Unlike CVAE, the BCDE generative path prevents x from directly influencing y. Following the conditional training paradigm in Sohn et al. (2015), conditional/discriminative training of the BCDE means maximizing the lower bound of a conditional likelihood similar to (6),i.e.,

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

When trained over a dataset of paired $(\mathbf{X}, \mathbf{Y})$ samples, the overall conditional training objective is

$$\mathcal{C}(\theta, \phi; \mathbf{X}, \mathbf{Y}) = \sum_{x,y \in \mathbf{X},\mathbf{Y}} \mathcal{C}(\theta, \phi; x, y). \tag{8}$$

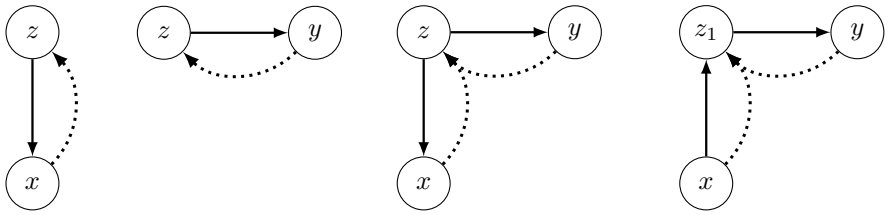
However, this approach suffers from the same limitations as CVAE and imposes a bottleneck that limits the flexibility of the generative model. Instead, we propose a *hybrid* training framework that takes advantage of the bottleneck architecture to avoid overfitting and supports semi-supervision.

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 to *factored inference* (see Section 3.4), i.e., 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. Via this regularization, BCDE benefits from unpaired x and y for conditional density estimation.

3.2. 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 (see Figs. 2(a) to 2(c)). Thus, the variational lower bound of the



(a) Joint: (x) (b) Joint: (y) (c) Joint: (x, y) (d) Conditional: (x, y)

Figure 2. The joint and conditional components of the BCDE. Dotted lines represent recognition models. The conditional model parameters are regularized toward the joint model's. The natural pairing of the conditional and joint parameters is described in Table 1.

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

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 correspondence among the networks with and without the application of factored inference. We regularize all the BCDE networks to their corresponding BJDE network parameters.

joint likelihood is

$$\ln p_{\tilde{\theta}}(x,y) \geq \mathcal{J}_{xy}(\tilde{\theta},\tilde{\phi};x,y) = \mathbb{E}_{q_{\tilde{\phi}}(z|x,y)}\left[\ln \frac{p(z)p_{\tilde{\theta}}(x|z)p_{\tilde{\theta}}(y|z)}{q_{\tilde{\phi}}(z|x,y)}\right]. \quad (9)$$

We use $\{\tilde{\theta},\tilde{\phi}\}$ to indicate the parameters of the BJDE networks and reserve $\{\theta,\phi\}$ for the BCDE parameters. For samples in which 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 in a non-trivial manner, 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 might 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

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

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

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. Thus, the BJDE provides the following benefits: **1**) learning the distribution of x makes the inference of z given x robust to perturbations in the inputs, **2**) z becomes a joint-embedding of x and y, **3**) the model can leverage unlabeled data. Following the convention in Eq. (8), the joint training objectives is

$$\mathcal{J}(\tilde{\theta},\tilde{\phi};\mathbf{X}_u,\mathbf{Y}_u,\mathbf{X}_l,\mathbf{Y}_l) = \mathcal{J}_x(\tilde{\theta},\tilde{\phi};\mathbf{X}_u) + \mathcal{J}_y(\tilde{\theta},\tilde{\phi};\mathbf{Y}_u) + \mathcal{J}_{xy}(\tilde{\theta},\tilde{\phi};\mathbf{X}_l,\mathbf{Y}_l), \quad (10)$$

where $(\mathbf{X}_l,\mathbf{Y}_l)$ is a dataset of paired (x,y) samples, and $\mathbf{X}_u$ and $\mathbf{Y}_u$ are datasets of unpaired samples.

3.3. Blending Joint and Conditional Deep Models

Because of potential model mis-specifications, the BJDE is not expected to yield good performance if applied to the conditional task. Thus, we aim to blend the BJDE and BCDE models in the spirit of Lasserre et al. (2006). However, we note that (7) is not directly applicable since the BCDE and BJDE are two different models, and not two different views (discriminative and generative) of the same model. Therefore, it is not immediately clear how to tie the BCDE and BJDE parameters together. Further, these models involve conditional probabilities parameterized by deep networks and have no closed form for inference.

Any natural prior for the BCDE parameter θ and the BJDE parameter $\tilde{\theta}$ should encourage $p_{\text{BCDE}}(y|x,\theta)$ to be close to $p_{\text{BJDE}}(y|x,\tilde{\theta})$. In the presence of the latent variable z, it is then natural to encourage $p(z|x,\theta)$ to be close to $p(z|x,\tilde{\theta})$ and $p(y|z,\theta)$ to be close to $p(y|z,\tilde{\theta})$. However, enforcing the former condition is intractable as we do not have a closed form for $p_{\text{BJDE}}(z|x,\tilde{\theta})$. Fortunately, an approxima-

tion of $p_{\text{BJDE}}(z|x,\tilde{\theta})$ is provided by the recognition model $q(z|x,\tilde{\phi})$. Thus, we propose to softly tie together the parameters of networks defining $p(z|x,\theta)$ and $q(z|x,\tilde{\phi})$. This strategy effectively leads to a joint prior over the model network parameters, as well as the recognition network parameters $p(\tilde{\phi},\tilde{\theta},\phi,\theta)$.

As a result, we arrive at the following hybrid blending of deep stochastic models and its variational lower bound

$$\begin{aligned}\ln p(\mathbf{X}_l,\mathbf{Y}_l,\mathbf{X}_u,\mathbf{Y}_u,\tilde{\theta},\tilde{\phi},\theta,\phi) \geq \ln p(\tilde{\theta},\tilde{\phi},\theta,\phi) + \\ \mathcal{J}_x(\tilde{\theta},\tilde{\phi};\mathbf{X}_u) + \mathcal{J}_y(\tilde{\theta},\tilde{\phi};\mathbf{Y}_u) + \\ \mathcal{J}_x(\tilde{\theta},\tilde{\phi};\mathbf{X}_l) + \mathcal{C}(\theta,\phi;\mathbf{X}_l,\mathbf{Y}_l). \end{aligned} \tag{11}$$

We interpret $\ln p(\tilde{\theta},\tilde{\phi},\theta,\phi)$ as a ℓ_2-regularization term that softly ties the joint parameters $(\tilde{\theta},\tilde{\phi})$ and conditional parameters (θ,ϕ) in an appropriate way. For the BCDE and BJDE, there is a natural one-to-one mapping from the conditional parameters to a subset of the joint parameters. For the joint model described in Fig. 2(c) and conditional model in Fig. 2(d), the parameter pairings are provided in Table 1. Formally, we define $\gamma = \{\theta,\phi\}$ and use the index $\gamma_{a|b}$ to denote the parameter of the neural network on the Bayesian network link $b \to a$ in the BCDE. For example $\gamma_{z|x} = \theta_{z|x}$, $\gamma_{z|x,y} = \phi_{z|x,y}$. Similarly, let $\tilde{\gamma} = \{\tilde{\theta},\tilde{\phi}\}$. In the BJDE, the same notation yields $\tilde{\gamma}_{z|x} = \tilde{\phi}_{z|x}$. The hybrid blending regularization term can be written as

$$\ln p(\theta,\phi,\tilde{\theta},\tilde{\phi}) = -\frac{\lambda}{2}\sum_{i\in I}\|\gamma_i - \tilde{\gamma}_i\|_2^2 + \text{const}, \tag{12}$$

where I denotes the set of common indices of the joint and conditional parameters. When the index is $z|x$, it effectively means that $p(z|x,\theta)$ is softly tied to $q(z|x,\tilde{\theta})$, i.e.,

$$\|\gamma_{z|x} - \tilde{\gamma}_{z|x}\|_2^2 = \|\theta_{z|x} - \tilde{\phi}_{z|x}\|_2^2 \,.$$

Setting $\lambda = 0$ unties the BCDE from the BJDE, and effectively yields to a conditionally trained BCDE, while letting $\lambda \to \infty$ forces the corresponding parameters of the BCDE and BJDE to be identical.

Interestingly, Eq. (11) does not contain the term $\mathcal{J}_{xy}$. Since explicit training of $\mathcal{J}_{xy}$ may lead to learning a better joint embedding in the space of z, we note the following generalization of Eq. (11) that trades off the contribution between $\mathcal{J}_{xy}$ and $[\mathcal{J}_x + \mathcal{C}]$,

$$\begin{aligned}&\ln p(\mathbf{X}_l,\mathbf{Y}_l,\mathbf{X}_u,\mathbf{Y}_u,\tilde{\theta},\tilde{\phi},\theta,\phi) \\ &\geq \mathcal{H}(\tilde{\theta},\tilde{\phi},\theta,\phi;\mathbf{X}_l,\mathbf{Y}_l,\mathbf{X}_u,\mathbf{Y}_u) \\ &= \ln p(\tilde{\theta},\tilde{\phi},\theta,\phi) + \\ &\quad \mathcal{J}_x(\tilde{\theta},\tilde{\phi};\mathbf{X}_u) + \mathcal{J}_y(\tilde{\theta},\tilde{\phi};\mathbf{Y}_u) + \\ &\quad \alpha\cdot\mathcal{J}_{xy}(\tilde{\theta},\tilde{\phi};\mathbf{X}_l,\mathbf{Y}_l) + \\ &\quad (1-\alpha)\cdot\Big[\mathcal{J}_x(\tilde{\theta},\tilde{\phi};\mathbf{X}_l) + \mathcal{C}(\theta,\phi;\mathbf{X}_l,\mathbf{Y}_l)\Big]. \end{aligned} \tag{13}$$

Intuitively, the equation computes the lower bound of $p(\mathbf{X}_l,\mathbf{Y}_l)$, either using the joint parameters $\tilde{\theta},\tilde{\phi}$ or factorizes $p(\mathbf{X}_l,\mathbf{Y}_l)$ into $p(\mathbf{X}_l)p(\mathbf{Y}_l \mid \mathbf{X}_l)$ before computing the lower bound of $p(\mathbf{Y}_l \mid \mathbf{X}_l)$ with the conditional parameters. A proof that the lower bound holds for any $0 \leq \alpha \leq 1$ is provided in Appendix B. For simplicity, we set $\alpha = 0.5$ and do not tune α in our experiments.

3.4. Factored Inference

The inference network $q_\phi(z|x,y)$ is usually parameterized as a single neural network that takes both x and y as input. Using the precision-weighted merging scheme proposed by Sønderby et al. (2016), we also consider an alternative parameterization of $q_\phi(z|x,y)$ that takes a weighted-average of the Gaussian distribution $q_\phi(z|x)$ and a Gaussian likelihood term $\hat{\ell}(z;y)$ (see Appendix A). Doing so offers a more compact recognition model and more sharing parameters between the BCDE and BJDE (e.g., see the bottom two rows in Table 1), but at the cost of lower flexibility for the variational family $q_\phi(z|x,y)$.

4. Experiments

We evaluated the performance of our hybrid training procedure on the permutation-invariant quadrant prediction task (Sohn et al., 2014; Sohn et al., 2015) for MNIST, SVHN, and CelebA. The quadrant prediction task is a conditional density estimation problem where an image data set is partially occluded. The model is given the observed region and is evaluated by its perplexity on the occluded region. The quadrant prediction task consists of four sub-tasks depending on the degree of partial observability. 1-quadrant prediction: the bottom left quadrant is observed. 2-quadrant prediction: the left half is observed. 3-quadrant prediction: the bottom right quadrant is *not* observed. Top-down prediction: the top half is observed.

In the fully-supervised case, the original MNIST training set $\{x'_i\}_{i=1}^{50000}$ is converted into our CDE training set $\{\mathbf{X}_l,\mathbf{Y}_l\} = \{x_i,y_i\}_{i=1}^{50000}$ by splitting each image into its observed x and unobserved y regions according to the quadrant prediction task. Note that the training set does not contain the original class label information. In the n_l-label semi-supervised case, we randomly sub-sampled n_l pairs to create our labeled training set $\{x_i,y_i\}_{i=1}^{n_l}$. The remaining n_u paired samples are decoupled and put into our unlabeled training sets $\mathbf{X}_u = \{x_i\}_{i=1}^{n_u}$, $\mathbf{Y}_u = \{y_i\}_{i=1}^{n_u}$. Test performance is the conditional density estimation performance on the entire test set, which is also split into input x and target y according to the quadrant prediction task. Analogous procedure is used for SVHN and CelebA.

For comparison against Sohn et al. (2015), we evaluate the performance of our models on the MNIST 1-

Models	$n_l = 50000$	$n_l = 25000$	$n_l = 10000$	$n_l = 5000$
CVAE (Sohn et al., 2015)	63.91	-	-	-
BCDE (conditional)	62.45 ± 0.02	64.50 ± 0.03	68.23 ± 0.05	71.66 ± 0.06
BCDE (naïve pre-train)	$\mathbf{62.00 \pm 0.02}$	63.27 ± 0.04	65.14 ± 0.05	67.13 ± 0.04
BCDE (hybrid)	62.16 ± 0.03	$\mathbf{62.90 \pm 0.02}$	$\mathbf{64.08 \pm 0.03}$	65.10 ± 0.03
BCDE (hybrid + factored)	62.81 ± 0.05	63.47 ± 0.02	64.16 ± 0.02	$\mathbf{64.64 \pm 0.05}$

Table 2. MNIST quadrant prediction task: 1-quadrant. We report the test set loss (IW=100) and standard error.

Models	$n_l = 50000$	$n_l = 25000$	$n_l = 10000$	$n_l = 5000$
CVAE (Sohn et al., 2015)	44.73	-	-	-
BCDE (conditional)	43.91 ± 0.01	45.49 ± 0.03	48.16 ± 0.02	50.83 ± 0.04
BCDE (naïve pre-train)	$\mathbf{43.53 \pm 0.02}$	44.42 ± 0.04	45.81 ± 0.01	47.49 ± 0.06
BCDE (hybrid)	$\mathbf{43.56 \pm 0.02}$	$\mathbf{44.10 \pm 0.02}$	45.23 ± 0.02	46.39 ± 0.03
BCDE (hybrid + factored)	44.07 ± 0.02	44.41 ± 0.02	$\mathbf{45.02 \pm 0.04}$	$\mathbf{45.86 \pm 0.06}$

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

Models	$n_l = 50000$	$n_l = 25000$	$n_l = 10000$	$n_l = 5000$
CVAE (Sohn et al., 2015)	20.95	-	-	-
BCDE (conditional)	20.64 ± 0.01	21.27 ± 0.01	22.44 ± 0.03	23.72 ± 0.04
BCDE (naïve pre-train)	20.37 ± 0.01	20.87 ± 0.02	21.65 ± 0.02	22.32 ± 0.05
BCDE (hybrid)	$\mathbf{20.31 \pm 0.01}$	20.69 ± 0.02	21.36 ± 0.02	22.27 ± 0.02
BCDE (hybrid + factored)	20.43 ± 0.01	$\mathbf{20.56 \pm 0.01}$	$\mathbf{21.16 \pm 0.01}$	$\mathbf{21.81 \pm 0.03}$

Table 4. MNIST quadrant prediction task: 3-quadrant.

Models	$n_l = 10000$	$n_l = 5000$
BCDE (conditional)	4657 ± 48	4845 ± 33
BCDE (naïve pre-train)	4547 ± 23	4627 ± 13
BCDE (hybrid)	$\mathbf{4213 \pm 21}$	$\mathbf{4392 \pm 13}$
BCDE (hybrid + factored)	4700 ± 146	5030 ± 165

Table 5. SVHN prediction task: Top-Down.

Models	$n_l = 20000$	$n_l = 10000$
BCDE (conditional)	5805 ± 2	5817 ± 3
BCDE (naïve pre-train)	5784.8 ± 0.5	5793 ± 1
BCDE (hybrid)	5778.6 ± 0.4	5781.3 ± 0.5
BCDE (hybrid + factored)	$\mathbf{5776.1 \pm 0.3}$	$\mathbf{5780.3 \pm 0.6}$

Table 6. CelebA prediction task: Top-Down.

quadrant, 2-quadrant, and 3-quadrant prediction tasks. The MNIST digits are statically-binarized by sampling from the Bernoulli distribution according to their pixel values (Salakhutdinov & Murray, 2008). We use a sigmoid layer to learn the parameter of the Bernoulli observation model.

We provide the performance on the top-down prediction task for SVHN and CelebA. We used a discretized logistic observation model Kingma et al. (2016) to model the pixel values for SVHN and a Gaussian observation model with fixed variance for CelebA. For numerical stability, we rely on the implementation of the discretized logistic distribution described in Salimans et al. (2017).

In all cases, we extracted a validation set of 10000 samples for hyperparameter tuning. While our training objective uses a single (IW=1) importance-weighted sample (Burda et al., 2015), we measure performance using IW=100 to get a tighter bound on the test log-likelihood (Sohn et al., 2015). We run replicates of all experiments and report the mean performance with standard errors. For a more expressive variational family (Ranganath et al., 2015), we use two stochastic layers in the BCDE and perform inference via top-down inference (Sønderby et al., 2016). We use multi-layered perceptrons (MLPs) for MNIST and SVHN, and convolutional neural networks (CNNs) for CelebA. All neural networks are batch-normalized (Ioffe & Szegedy, 2015) and updated with *Adam* (Kingma & Ba, 2014). The number of training epochs is determined based on the validation set. The dimensionality of each stochastic layer is 50, 100, and 300 for MNIST, CelebA, and SVHN respectively. All models were implemented in Python[3] using Tensorflow (Abadi, 2015).

4.1. Conditional Log-Likelihood Performance

Tables 2 to 6 show the performance comparisons between the CVAE and the BCDE. For baselines, we use the CVAE, the BCDE trained with the conditional objective, and the BCDE initialized via pre-training $\mathcal{J}_x(\cdot)$ and $\mathcal{J}_y(\cdot)$ using the available x and y data separately (and then trained conditionally). Against these baselines, we measure the performance of the BCDE (with and without factored inference)

[3] github.com/ruishu/bcde

trained with the hybrid objective $\mathcal{H}(\cdot)$. We tuned the regularization hyperparameter $\lambda = \{10^{-3}, 10^{-2}, \ldots, 10^{3}\}$ on the MNIST 2-quadrant semi-supervised tasks and settled on using $\lambda = 10^{-2}$ for all tasks.

Fully-supervised regime. By comparing in the fully-supervised regime for MNIST (Tables 2 to 4, $n_l = 50000$), we show that the hybrid BCDE achieves competitive performance against the pretrained BCDE and out-performs previously reported results for CVAE (Sohn et al., 2015).

Semi-supervised regime. As the labeled training size n_l reduces, the benefit of having the hybrid training procedure becomes more apparent. The BCDEs trained with the hybrid objective function tend to significantly improve upon its conditionally-trained counterparts.

On MNIST, hybrid training of the factored BCDE achieves the best performance. Both hybrid models achieve over a 1-nat difference than the pre-trained baseline in some cases—a significant difference for binarized MNIST (Wu et al., 2016). Conditional BCDE performs very poorly in the semi-supervised tasks due to overfitting.

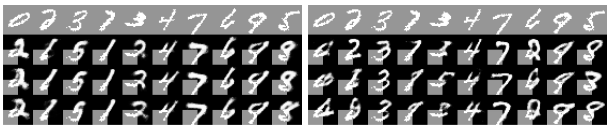
On CelebA, hybrid training of the factored BCDE also achieves the best performance. Both hybrid models significantly out-perform the conditional baselines and yield better visual predictions than conditional BCDE (see Appendix C). The hybrid models also outperform pre-trained BCDE with only half the amount of labeled data.

On SVHN, the hybrid BCDE with standard inference model significantly out-performs the conditional baselines. However, the use of factored inference results in much poorer performance. Since the decoder is a discretized logistic distribution with learnable scale, it is possible that the factored inference model is not expressive enough to model the posterior distribution.

Model entropy. In Figure 3, we sample from $p_\theta(y|x)$ for the conditional BCDE and the hybrid BCDE. We show that the conditionally-trained BCDE achieves poorer performance because it learns a lower-entropy model. In contrast, hybrid training learns a lower perplexity model, resulting in a high-entropy conditional image generator that spreads the conditional probability mass over the target output space (Theis et al., 2015).

4.2. Conditional Training Overfits

To demonstrate the hybrid training's regularization behavior, we show the test set performance during training (Fig. 4) on the 2-quadrant MNIST task ($n_l = 10000$). Even with pre-trained initialization of parameters, models that were trained conditionally quickly overfit, resulting in poor test set performance. In contrast, hybrid training regularizes the conditional model toward the joint model, which is much more resilient against overfitting.



(a) Conditional (b) Hybrid

Figure 3. Comparison of conditional image generation for the conditional versus hybrid BCDE on the semi-supervised 1-quadrant task. 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 model that has lower perplexity.

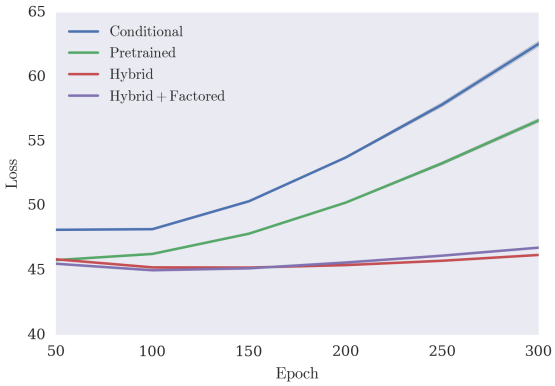


Figure 4. Comparison of the BCDE variants on the 2-quadrant MNIST prediction task with $n_l = 10000$ labeled points. In contrast to conditional training, hybrid training is less susceptible to overfitting.

4.3. Robustness of Representation

Since hybrid training encourages the BCDE to consider the distribution of x, we can demonstrate that models trained in a hybrid manner are robust against structured perturbations of the data set. To show this, we experimented with two variants of the MNIST quadrant task called the *shift-sensitive* and *shift-invariant* top-bottom prediction tasks. In these experiments, we set $\lambda = 0.1$.

4.3.1. Shift-Sensitive Estimation

In the shift-sensitive task, the objective is to learn to predict the bottom half of the MNIST digit (y) when given the top half (x). However, we introduce structural perturbation to 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 BCDE using either the conditional or hybrid objective in the fully-

supervised regime. Note that compared to the original top-down prediction task, the perplexity of the conditional task remains the same after the perturbation is applied.

Models	No Shift	Shift	Δ
Conditional	41.59 ± 0.02	44.02 ± 0.03	2.43
Hybrid	41.33 ± 0.01	43.51 ± 0.01	2.17
Hybrid + Factored	41.20 ± 0.02	43.19 ± 0.02	1.99

Table 7. Shift-sensitive top-bottom MNIST prediction. Performance with and without structural corruption reported, along with the performance difference. Hybrid training is robust against structural perturbation of (x, y).

Table 7 shows that hybrid training consistently achieves better performance than conditional training. Furthermore, the hybridly trained models were less affected by the introduction of the perturbation, demonstrating a higher degree of robustness. Because of its more compact recognition model, hybrid + factored is less vulnerable to overfitting, resulting in a smaller performance gap between performance on the shifted and original data.

4.3.2. Shift-Invariant Estimation

The shift-invariant 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.

Models	No Shift	Shift	Δ
Conditional	41.59 ± 0.02	42.99 ± 0.04	1.40
Hybrid	41.33 ± 0.01	42.53 ± 0.02	1.20
Hybrid + Factored	41.20 ± 0.02	42.20 ± 0.02	1.00

Table 8. Shift-invariant top-bottom MNIST prediction. Performance with and without structural corruption reported, along with the performance difference. Hybrid training is robust against structural corruption of x.

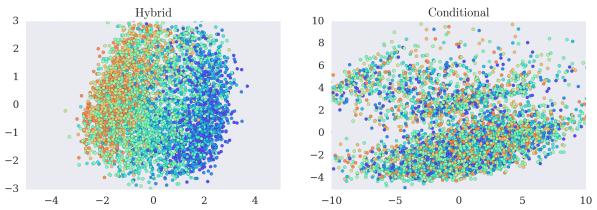


Figure 5. Visualization of the latent space of hybrid and conditionally-trained BCDEs. PCA plots of the latent space sub-region for all x's whose class label $= 2$ are shown. Fill color indicates the degree of shift: blue $= -4$, orange $= +4$.

Table 8 shows similar behavior to Table 7. Hybrid training continues to achieve better performance than conditional models and suffer a much smaller performance gap when structural corruption in x is introduced.

In Fig. 5, we show the PCA projections of the latent space sub-region populated by digits 2 and color-coded all points based on the degree of shift. We observe that hybrid training versus conditional training of the BCDE result in very different learned representations in the stochastic layer. Because of regularization toward the joint model, the hybrid BCDE's latent representation retrains information about x and learns to untangle shift from other features. And as expected, conditional training does not encourage the BCDE to be aware of the distribution of x, resulting in a latent representation that is ignorant of the shift feature of x.

5. Conclusion

We presented a new framework for high-dimensional conditional density estimation. The building blocks of our framework are a pair of sibling models: the Bottleneck Conditional Density Estimator (BCDE) and the Bottleneck Joint Density Estimator (BJDE). These models use layers of stochastic neural networks as bottleneck between the input and output data. While the BCDE learns the conditional distribution $p(y|x)$, the BJDE learns the joint distribution $p(x, y)$. The bottleneck constraint implies that only the bottleneck needs to be marginalized when either the input x or the output y are missing during training, thus, enabling the BJDE to be trained in a semi-supervised fashion.

The key component of our framework is our hybrid objective function that regularizes the BCDE towards the BJDE. Our new objective is a novel extension of Lasserre et al. (2006) that enables the principle of hybrid blending to be applied to deep variational models. Our framework provides a new mechanism for the BCDE, a conditional model, to become more robust and to learn from unlabeled data in semi-supervised conditional density estimation.

Our experiments showed that hybrid training is competitive in the fully-supervised regime against pre-training, and achieves superior performance in the semi-supervised quadrant prediction task in comparison to conditional models, achieving new state-of-the-art performances on MNIST, SVHN, and CelebA. Even with pre-trained weight initializations, the conditional model is still susceptible to overfitting. In contrast, hybrid training is significantly more robust against overfitting. Furthermore, hybrid training transfers the nice embedding properties of the BJDE to the BCDE, allowing the BCDE to learn better and more robust representation of the input x. The success of our hybrid training framework makes it a prime candidate for other high-dimensional conditional density estimation problems, especially in semi-supervised settings.

References

Abadi, Martín, et. al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL http://tensorflow.org/. Software available from tensorflow.org.

Burda, Y., Grosse, R., and Salakhutdinov, R. Importance Weighted Autoencoders. *arXiv preprints:1509.00519*, 2015.

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

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

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

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

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

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

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

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

Kingma, Diederik P., Salimans, Tim, and Welling, Max. Improving variational inference with inverse autoregressive flow. *CoRR*, abs/1606.04934, 2016. URL http://arxiv.org/abs/1606.04934.

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

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

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

Ranganath, R., Tran, D., and Blei, D. M. Hierarchical Variational Models. *ArXiv e-prints*, 1511.02386, November 2015.

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

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

Salimans, Tim, Karpathy, Andrej, Chen, Xi, and Kingma, Diederik P. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *CoRR*, abs/1701.05517, 2017. URL http://arxiv.org/abs/1701.05517.

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

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

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

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

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

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

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

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