LEARNING CONSISTENT DEEP GENERATIVE MODELS FROM SPARSE DATA VIA PREDICTION CONSTRAINTS

Anonymous authors
Paper under double-blind review

ABSTRACT

We develop a new framework for learning variational autoencoders and other deep generative models that balances generative and discriminative goals. Our framework optimizes model parameters to maximize a variational lower bound on the likelihood of observed data, subject to a task-specific prediction constraint that prevents model misspecification from leading to inaccurate predictions. We further enforce a consistency constraint, derived naturally from the generative model, that requires predictions on reconstructed data to match those on the original data. We show that these two contributions – prediction constraints and consistency constraints – lead to promising image classification performance, especially in the semi-supervised scenario where category labels are sparse but unlabeled data is plentiful. Our approach enables advances in generative modeling to directly boost semi-supervised classification performance, an ability we demonstrate by augmenting deep generative models with latent variables capturing spatial transformations.

1 INTRODUCTION

We develop broadly applicable methods for learning flexible models of high-dimensional data, like images, that are paired with (discrete or continuous) labels. We are particularly interested in *semi-supervised* learning (Zhu, 2005; Oliver et al., 2018) from data that is sparsely labeled, a common situation in practice due to the cost or privacy concerns associated with data annotation. Given a large and sparsely labeled dataset, we seek a single probabilistic model that *simultaneously* makes good predictions of labels and provides a high-quality generative model of the high-dimensional input data. Strong generative models are valuable because they can allow incorporation of domain knowledge, can address partially missing or corrupted data, and can be visualized to improve interpretability.

Prior approaches for the semi-supervised learning of deep generative models include methods based on *variational autoencoders* (VAEs) (Kingma et al., 2014; Siddharth et al., 2017), *generative adversarial networks* (GANs) (Dumoulin et al., 2017; Kumar et al., 2017), and hybrids of the two (Larsen et al., 2016; de Bem et al., 2018; Zhang et al., 2019). While these all allow sampling of data, a major shortcoming of these approaches is that they do not adequately use labels to inform the generative model. Furthermore, GAN-based approaches lack the ability to evaluate the learned probability density function, which can be important for tasks such as model selection and anomaly detection.

This paper develops a framework for training *prediction constrained variational autoencoders* (PC-VAEs) that minimize application-motivated loss functions in the prediction of labels, while simultaneously learning high-quality generative models of the raw data. Our approach is inspired by the prediction-constrained framework recently proposed for learning supervised topic models of "bag of words" count data (Hughes et al., 2018), but differs in four major ways. First, we develop scalable algorithms for learning a much larger and richer family of deep generative models. Second, we capture uncertainty in latent variables rather than simply using point estimates. Third, we allow more flexible specification of loss functions. Finally, we show that the generative model structure leads to a natural consistency constraint vital for semi-supervised learning from very sparse labels.

Our experiments demonstrate that *consistent prediction-constrained* (CPC) VAE training leads to prediction performance competitive with state-of-the-art discriminative methods on fully-labeled datasets, and *excels* over these baselines when given semi-supervised datasets where labels are rare.

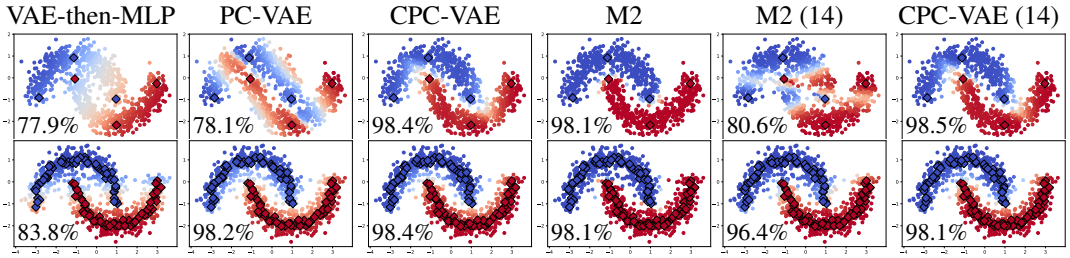


Figure 1: Predictions from SSL VAE methods on half-moon binary classification task, with accuracy in lower corner. Each dot indicates a 2-dim. feature vector, colored by predicted binary label. *Top:* 6 labeled examples (diamond markers), 994 unlabeled. *Bottom:* 100 labeled, 900 unlabeled. First 4 columns use $C = 2$ encoding dimensions, last 2 use $C = 14$. M2 (Kingma et al., 2014) classification accuracy *deterioriates* when increasing model capacity from 2 to 14, especially with only 6 labels (drop from 98.1% to 80.6% accuracy). In contrast, our CPC VAE approach is reliable at any model capacity, as it better aligns generative and discriminative goals.

2 Background: Deep Generative Models and Semi-supervision

We now describe VAEs as deep generative models and review previous methods for *semi-supervised learning* (SSL) of VAEs, highlighting weaknesses that we later improve upon. We assume all SSL tasks provide two training datasets: an unsupervised (or unlabeled) dataset $\mathcal{D}^U$ of N feature vectors x, and a supervised (or labeled) dataset $\mathcal{D}^S$ containing M pairs (x, y) of features x and label $y \in \mathcal{Y}$. Labels are often sparse ($N \gg M$) and can be discrete or continuous.

2.1 Unsupervised Generative Modeling with the VAE

The variational autoencoder (Kingma & Welling, 2014) is an unsupervised model with two components: a generative model and an inference model. The generative model defines for each example a joint distribution $p_\theta(x, z)$ over "features" (observed vector $x \in \mathbb{R}^D$) and "encodings" (hidden vector $z \in \mathbb{R}^C$). The "inference model" of the VAE defines an approximate posterior $q_\phi(z \mid x)$, which is trained to be close to the true posterior ($q_\phi(z \mid x) \approx p_\theta(z \mid x)$) but much easier to evaluate. As in Kingma & Welling (2014), we assume the following conditional independence structure:

$$p_\theta(x, z) = \mathcal{N}(z \mid 0, I_C) \cdot \mathcal{F}(x \mid \mu_\theta(z), \sigma_\theta(z)), \quad q_\phi(z \mid x) = \mathcal{N}(z \mid \mu_\phi(x), \sigma_\phi(x)). \tag{1}$$

The likelihood $\mathcal{F}$ is often multivariate normal, but other distributions may give robustness to outliers. The (deterministic) functions μ_θ and σ_θ, with trainable parameters θ, define the mean and covariance of the likelihood. Given any observation x, the posterior of z is approximated as normal with mean μ_ϕ and (diagonal) covariance σ_ϕ parameterized by ϕ. These functions can be represented as *multi-layer perceptrons* (MLPs), *convolutional neural networks* (CNNs), or other (deep) neural networks.

We would ideally learn generative parameters θ by maximizing the marginal likelihood of features x, integrating latent variable z. Since this is intractable, we instead maximize a variational lower bound:

$$\max_{\theta,\phi} \sum_{x \in \mathcal{D}} \mathcal{L}^{\text{VAE}}(x; \theta, \phi), \qquad \mathcal{L}^{\text{VAE}}(x; \theta, \phi) = \mathbb{E}_{q_\phi(z|x)} \left[\log \frac{p_\theta(x,z)}{q_\phi(z|x)} \right] \leq \log p_\theta(x). \tag{2}$$

This expectation can be evaluated via Monte Carlo samples from the inference model $q_\phi(z|x)$. Gradients with respect to θ, ϕ can be similarly estimated by the reparameterization "trick" of representing $q_\phi(z \mid x)$ as a linear transformation of standard normal variables (Kingma & Welling, 2014).

Throughout this paper, we denote variational parameters by ϕ. Because the factorization of q changes for more complex models, we will write $\phi^{z|x}$ to denote the parameters specific to factor $q(z|x)$.

2.2 Two-Stage SSL: Maximize Feature Likelihood then Train Predictor

One way to employ the VAE for a semi-supervised task is a *two-stage* "VAE-then-MLP". First, train a VAE to maximize the unsupervised likelihood (2) of all observed features x (both labeled $\mathcal{D}^S$ and unlabeled $\mathcal{D}^U$). Second, we define a label-from-code *predictor* $\hat{y}_w(z)$ that maps each learned code representation z to a predicted label $y \in \mathcal{Y}$. We use an MLP with weights w, though any predictor could do. Let $\ell_S(y, \hat{y})$ be a loss function, such as cross-entropy, appropriate for the prediction task. We train the predictor to minimize the loss: $\min_w \sum_{x,y \in \mathcal{D}^S} \mathbb{E}_{q_\phi(z|x)} \left[\ell_S(y, \hat{y}_w(z)) \right]$. Importantly, this second stage uses only the small labeled dataset and relies on fixed parameters ϕ from stage one.

While "VAE-then-MLP" is a simple common baseline (Kingma et al., 2014), it has a key disadvantage: Labels are only used in the second stage, and thus a *misspecified* generative model in stage one will likely produce inferior predictions. Fig. 1 illustrates this weakness.

2.3 Semi-supervised VAEs: Maximize Joint Likelihood of Labels and Features

To overcome the weakness of the two-stage approach, previous work by Kingma et al. (2014) presented a VAE-inspired model called "M2" focused on the *joint* generative modeling of labels y and data x. M2 has two components: a generative model $p_\theta(x, y, z)$ and an inference model $q_\phi(y, z \mid x)$. Their generative model is factorized to sample labels (with frequencies π) first, and then features x:

$$p_\theta(x, y, z) = \mathcal{N}(z \mid 0, I_C) \cdot \text{Cat}(y \mid \pi) \cdot \mathcal{F}(x \mid \mu_\theta(y, z), \sigma_\theta(y, z)). \tag{3}$$

The M2 inference model sets $q_\phi(y, z \mid x) = q_{\phi^{y|x}}(y \mid x) q_{\phi^{z|x,y}}(z \mid x, y)$, where $\phi = (\phi^{y|x}, \phi^{z|x,y})$.

To train M2, Kingma et al. (2014) maximize the likelihood of all observations (labels and features):

$$\max_{\theta, \phi^{y|x}, \phi^{z|x,y}} \quad \textstyle\sum_{x,y \in \mathcal{D}^S} \mathcal{L}^S(x, y; \theta, \phi^{z|x,y}) + \sum_{x \in \mathcal{D}^U} \mathcal{L}^U(x; \theta, \phi^{y|x}, \phi^{z|x,y}). \tag{4}$$

The first, "supervised" term in Eq. (4) is a variational bound for the feature-and-label *joint* likelihood:

$$\mathcal{L}^S(x, y; \theta, \phi^{z|x,y}) = \mathbb{E}_{q_{\phi^{z|x,y}}(z|x,y)} \left[\log \frac{p_\theta(x,y,z)}{q_{\phi^{z|x,y}}(z|x,y)} \right] \leq \log p_\theta(x, y). \tag{5}$$

The second, "unsupervised" term is a variational lower bound for the features-only likelihood $\log p_\theta(x) \geq \mathcal{L}^U$, where $\mathcal{L}^U = \mathbb{E}_{q_\phi(y,z|x)} \left[\log \frac{p_\theta(x,y,z)}{q_\phi(y,z|x)} \right]$ can be simply expressed in terms of $\mathcal{L}^S$:

$$\mathcal{L}^U(x; \theta, \phi^{y|x}, \phi^{z|x,y}) = \textstyle\sum_{y \in \mathcal{Y}} q_{\phi^{y|x}}(y \mid x) \left(\mathcal{L}^S(x, y; \theta, \phi^{z|x,y}) - \log q_{\phi^{y|x}}(y \mid x) \right). \tag{6}$$

As with the unsupervised VAE, both terms in the objective can be computed via Monte Carlo sampling from the variational posterior, and gradients can be estimated via the reparameterization trick.

M2's prediction dilemma and heuristic fix. After training parameters θ, ϕ, we need to predict labels y given test data x. M2's structure assumes we make predictions via the inference model's discriminator density $q_{\phi^{y|x}}(y \mid x)$. However, the discriminator's parameter $\phi^{y|x}$ is only informed by the *unlabeled* data when using the objective above (it is not used to compute $\mathcal{L}^S$). We cannot expect accurate predictions from a parameter that does not touch *any* labeled examples in the training set.

To partially overcome this issue, Kingma et al. (2014) and later work use a weighted objective:

$$\max_{\theta, \phi} \sum_{x,y \in \mathcal{D}^S} \left(\alpha \log q_{\phi^{y|x}}(y \mid x) + \lambda \mathcal{L}^S(x, y; \theta, \phi^{z|x,y}) \right) + \sum_{x \in \mathcal{D}^U} \mathcal{L}^U(x; \theta, \phi^{y|x}, \phi^{z|x,y}). \tag{7}$$

This objective biases the inference model's discriminator to do well on the labeled set via an extra loss term (weighted by hyperparameter $\alpha > 0$). We can further include $\lambda > 0$ to balance the supervised and unsupervised terms. Originally, Kingma et al. (2014) fix $\lambda = 1$ and tune α to achieve good performance. Later, Siddharth et al. (2017) tuned λ to improve performance. Maaløe et al. (2016) used this same $\alpha \log q(y \mid x)$ term for labeled data to train VAEs with auxiliary variables.

Disadvantage: What Justification? While the $\mathcal{L}^S$ and $\mathcal{L}^U$ terms in Eq. (7) have a rigorous justification as maximizing the data likelihood under the assumed generative model, the first term ($\alpha \log q(y \mid x)$) is not justified by the generative or inference model. In particular, suppose the training data were fully labeled: we would ignore the $\mathcal{L}^U$ terms altogether, and the remaining terms would *decouple* the parameters $\theta, \phi^{z|x,y}$ from the discriminator parameters $\phi^{y|x}$. This is deeply unsatisfying: We want a single model guided by *both* generative and discriminative goals, not two separate models. Even in partially-labeled scenarios, including this α term does not adequately balance generative and discriminative goals, as we demonstrate in later examples. An overly flexible yet misspecified generative model may go astray and compromise predictions.

Disadvantage: Runtime Cost. Another disadvantage is that the computation of $\mathcal{L}^U$ in Eq. (6) is *expensive*. If labels are discrete, computing this sum exactly is possible but requires a sum over all $L = |\mathcal{Y}|$ possible class labels, computing a Monte Carlo estimate of $\mathcal{L}^S$ for each one. While Monte Carlo approximations can avoid the explicit sum in Eq. (6), they may make gradients too noisy.

Extensions. Siddharth et al. (2017) showed how $\mathcal{L}^S$ and $\mathcal{L}^U$ could be extended to any desired conditional independence structure for $q_\phi(y, z \mid x)$, generalizing the label-then-code factorization $q_\phi(y \mid x) q_\phi(z \mid x, y)$ of Kingma et al. (2014). While importance sampling leads to likelihood bounds,

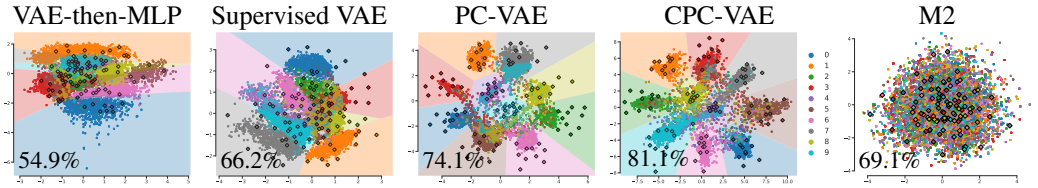


Figure 2: Semi-supervised learning of 2-dim. encodings of MNIST digits, with accuracy in lower corner. All methods use 100 labeled examples and 49,900 unlabeled examples. Each observed image x is mapped to its most likely 2-dim. encoding z and colored by true label y. Labeled examples are emphasized. Where applicable, we also show class decision boundaries. *Baselines (from left):* 2-stage unsupervised VAE-then-MLP (Sec. 2.2) and a "supervised" VAE maximizing joint likelihood $\log p(x, y)$ (a special case of our PC method with $\lambda = 1$, Sec. 3.1). *Our methods:* Prediction constrained VAE (PC-VAE with $\lambda = 25$, Sec. 3.1) and consistent prediction constrained VAE (CPC-VAE, Sec. 3.2). *Competitors:* M2 from Kingma et al. (2014), which intentionally decouples label y from "style" z, has limited accuracy due to imbalance of discriminative and generative goals.

the overall objective still has two undesirable traits. First, it is expensive, requiring either marginalization of y to compute $\mathcal{L}^U$ in Eq. (6) or marginalization of z to compute $q(y|x) = \int q_\phi(y, z|x)dz$. Second, the approach requires the heuristic inclusion of the discriminator loss $\alpha \log q(y \mid x)$. While recent parallel work by Gordon & Hernández-Lobato (2020) also tries to improve SSL for VAEs, their approach couples discriminative and generative terms only distantly through a joint prior over parameters and still requires expensive sums over labels when computing generative likelihoods.

3 Prediction-Constrained Learning with Consistency

We now highlight two experiments that demonstrate disadvantages of prior SSL methods, and contrast them with our new approaches. In Fig. 1 we show the predictive accuracy of several SSL methods on the widely-used "half-moon" task, where the goal is to to predict a binary label y given 2-dimensional features x. We focus on the top row, which shows results given only 6 labeled examples (3 of each class) but hundreds of unlabeled examples. Notably, while M2 has 98.1% accuracy with a small encoding space ($C = 2$), if the generative model is too flexible ($C = 14$) it learns overly complex structure that does not help label-from-feature predictions, dropping accuracy to only 80.6%. In contrast, our *consistent prediction constrained* (CPC) VAE gets over 98% accuracy with either $C = 2$ or $C = 14$. We have verified it maintains 98% even at $C = 50$, while M2 shows further instability.

Second, in Fig. 2 we show SSL methods for classifying images of MNIST digits (LeCun et al., 2010), given only 10 labeled examples per digit. We seek models with highly accurate label-from-feature predictions, as well as interpretable relationships between the encoding z and these predictions. When forced to use a 2-dimensional latent space, M2 has worse accuracy and (by design) no apparent relationship between encoding z and label y. In contrast, our CPC approach offers noticeable advantages over all baselines in both accuracy and interpretability of the encoding space.

3.1 Prediction Constrained Training for VAEs

We develop a framework for *jointly* learning a strong generative model of features x, and making label-given-feature predictions $\hat{y}(x)$ of uncompromised quality, by requiring predictions to meet a user-specified quality threshold. Our *prediction constrained* training objective enables end-to-end estimation of all parameters while incorporating the *same* task-specific prediction rules and loss functions that will be used in heldout evaluation ("test") scenarios. Our goals are similar to previous work on end-to-end approximate inference for task-specific losses with simpler probabilistic models (Lacoste-Julien et al., 2011; Stoyanov et al., 2011), but our approach yields simpler algorithms.

Generative model. Our generative model does not include labels y, only features x and encodings z. Their joint distribution $p_\theta(x, z)$ factorizes as the unsupervised VAE of Eq. (1), and we also use the inference model $q_\phi(z \mid x)$ defined in Eq. (1). While M2 included the labels y in its generative model (Kingma et al., 2014), our goals are different: we wish to make label-given-feature predictions, but we are not interested in label marginals or other distributions over y that do not condition on x.

Label-from-feature prediction. To predict labels y from features x, we use a predictor similar to the two-stage method of Sec. 2.2. We first sample an encoding $z \sim q_\phi(z|x)$ from the learned inference

model, and then transform this encoding z to a label via the predictor function $\hat{y}_w(z)$ with parameter w. By sharing random variable z, the generative model is involved in label-from-feature predictions.

Constrained PC objective. Unlike the two-stage model, our approach does not do post-hoc prediction with a previously learned generative model. Instead, we train the predictor simultaneously with the generative model via a new, *prediction-constrained* (PC) objective:

$$\max_{\theta, \phi^{z|x}, w} \quad \sum_{x \in \mathcal{D}^U \cup \mathcal{D}^S} \mathcal{L}^{\text{VAE}}(x; \theta, \phi^{z|x}), \quad \text{subj. to:} \quad \frac{1}{M} \sum_{x,y \in \mathcal{D}^S} \underbrace{\mathbb{E}_{q_\phi(z|x)}[\ell_S(y, \hat{y}_w(z))]}_{\mathcal{P}(x,y;\phi^{z|x},w)} \leq \epsilon. \tag{8}$$

The constraint requires that any feasible solution achieve average prediction loss less than ϵ on the labeled training set. Both the loss function and scalar threshold $\epsilon > 0$ can be set to reflect task-specific needs (e.g., classification must have a certain false positive rate or overall accuracy). The loss function may be any differentiable function, and need not equal the log-likelihood of discrete labels as assumed by previous work specialized to supervision of topic models (Hughes et al., 2018).

Unconstrained PC objective. Using the KKT conditions, we define an equivalent unconstrained objective that maximizes the unsupervised likelihood but penalizes inaccurate label predictions:

$$\max_{\theta, \phi^{z|x}, w} \quad \textstyle\sum_{x \in \mathcal{D}^U \cup \mathcal{D}^S} \mathcal{L}^{\text{VAE}}(x; \theta, \phi^{z|x}) - \lambda \sum_{x,y \in \mathcal{D}^S} \mathcal{P}(x, y; \phi^{z|x}, w). \tag{9}$$

Here λ is a positive Lagrange multiplier chosen to ensure that the target prediction constraint is achieved; smaller loss tolerances ϵ require larger penalty multipliers λ. This PC objective, and gradients for parameters θ, ϕ, w, can be estimated via Monte Carlo samples from $q_\phi(z \mid x)$.

Justification. While the PC objective of Eq. (9) may look superficially similar to Eq. (7), we emphasize two key differences. First, our objective couples a generative likelihood and a prediction loss via the shared variational parameters $\phi^{z|x}$. This makes both generative and discriminative performance depend on the *same* learned encoding z. (Later we show how to partition z so some entries are discriminative, while others affect generative "style" only.) In contrast, the M2 objective uses a label-given-features conditional to make predictions that does not share *any* of its parameters $\phi^{y|x}$ with the supervised likelihood $\mathcal{L}^S$. Second, our objective is more affordable: no term requires an expensive marginalization over labels. This is key to scaling to big unlabeled datasets, and also enables tractable learning from datasets whose labels are continuous or multi-dimensional.

Hyperparameters. The major hyperparameter influencing PC training is the constraint multiplier $\lambda \geq 0$. Setting $\lambda = 0$ leads to unsupervised maximum likelihood training (or MAP training, given priors on θ) of a classic VAE. Setting $\lambda = 1$ and choosing a probabilistic loss $-\log p(y \mid z)$ produces a "supervised VAE" that maximizes the joint likelihood $p_\theta(x, y)$. But as illustrated in Fig. 2, because features x have much higher dimension than labels y, the resulting model may have weak predictive performance. Satisfying the strong prediction constraint of Eq. (8) typically requires $\lambda \gg 1$, and in practice we use validation data to select the best of several candidate λ values. If a task motivates a concrete tolerance ϵ, we can test an increasing sequence of λ values until the constraint is satisfied.

We emphasize that although Eq. (9) is easier to optimize, we prefer to think of the constrained problem in Eq. (8) as the "primary" objective, because our applied goals are to satisfy discriminative quality first; a generative model that predicts poorly is not plausible. Furthermore, the constrained objective is far more natural for semi-supervised learning. The choice of ϵ need not be concerned by the relative sizes of the labeled and unlabeled datasets. In contrast, if either $|\mathcal{D}^U|$ or $|\mathcal{D}^S|$ changes, the value of λ may need to change dramatically to reach the same prediction quality.

3.2 Enforcing Consistent Predictions from Generative Model Reconstructions

While the PC objective is effective given sufficient labeled data, it may generalize poorly when labels are very sparse (see Fig. 1). This fundamental problem arises because in the PC objective of Eq. (8), the parameters w of the predictor $\hat{y}_w(z)$ are only directly informed by the labeled training data.

Revisiting the generative model, let $x \sim p_\theta(\cdot \mid z')$ and $\bar{x} \sim p_\theta(\cdot \mid z')$ be two observations sampled from the *same* latent code z'. Even if the true label y of x is uncertain, we know that for this model to be useful for predictive tasks, $\bar{x}$ must have the *same* label as x. We formalize this relationship via a *consistency constraint* requiring label predictions for common-code data pairs $(x, \bar{x})$ to approximately match. The regularization provided by this constraint may dramatically boost performance.

Given features x, our method predicts labels by sampling $z \sim q_\phi(z \mid x)$ from the approximate posterior, and then applying our predictor $\hat{y}_w(z)$. Alternatively, given x we can first *simulate* alternative features $\bar{x}$ with matching code z by sampling from the inference and generative models, and then predict the label associated with $\bar{x}$. We constrain the label predictions y for x, and $\bar{y}$ for $\bar{x}$, to be similar via a consistency penalty function $\ell_C(y, \bar{y})$. For the classification tasks considered below, we use a cross-entropy consistency penalty ℓ_C. Given this penalty, we constrain the maximum values of the following consistency costs on *unlabeled* and *labeled* examples, respectively:

$$\mathcal{C}^U(x; \theta, \phi, w) \triangleq \mathbb{E}_{q_\phi(z|x)} \left[\mathbb{E}_{p_\theta(\bar{x}|z)} \left[\mathbb{E}_{q_\phi(\bar{z}|\bar{x})} \left[\ell_C(\hat{y}_w(z), \hat{y}_w(\bar{z}))\right]\right]\right], \tag{10}$$

$$\mathcal{C}^S(x, y; \theta, \phi, w) \triangleq \mathbb{E}_{q_\phi(z|x)} \left[\mathbb{E}_{p_\theta(\bar{x}|z)} \left[\mathbb{E}_{q_\phi(\bar{z}|\bar{x})} \left[\ell_C(y, \hat{y}_w(\bar{z}))\right]\right]\right]. \tag{11}$$

Consistent PC: Unconstrained objective. To train model parameters, we apply a consistency cost to all observed feature vectors, both unlabeled and labeled. The overall objective becomes:

$$\max_{\theta,\phi,w} \sum_{x \in \mathcal{D}^U \cup \mathcal{D}^S} \mathcal{L}^{\text{VAE}}(x; \theta, \phi) - \gamma \mathcal{C}^U(x; \theta, \phi, w) + \sum_{x,y \in \mathcal{D}^S} -\lambda \mathcal{P}(x, y; \phi, w) - \gamma \mathcal{C}^S(x, y; \theta, \phi, w),$$

where $\mathcal{L}$ is the unsupervised likelihood, $\mathcal{P}$ is the predictor loss, and $\mathcal{C}$ are the consistency constraints. Here, $\gamma > 0$ is a scalar Lagrange multiplier for the consistency terms, with similar interpretation as λ.

Aggregate Label Consistency. For SSL applications, we find it is also useful to regularize our model with an *aggregate label consistency* constraint, which forces the distribution of label predictions for unlabeled data is align with a known target distribution π. This discourages predictions on ambiguous unlabeled examples from collapsing to a single value. We define the aggregate consistency loss as: $\ell_A(\pi, \mathbb{E}_{x \sim \mathcal{D}^U, z \sim q(z|x)}[\hat{y}_w(z)])$, and again use a cross-entropy penalty. If the target distribution of labels π is unknown, we set it to the empirical distribution of the labeled data.

Related work on consistency. Recently popular SSL image classifiers focused on discriminative goals will train the weights of a CNN to minimize a modified objective that penalizes both label accuracy and a notions of "consistency" or smoothness on unlabeled data. Examples include smoothness under adversarial perturbations (Miyato et al., 2019), consistency under label-invariant transformations (Laine & Aila, 2017), and smoothness in predictions when interpolating between training examples (Berthelot et al., 2019). Such regularization can deliver competitive discriminative performance, but does not meet our goal of generative modeling (e.g., these approaches cannot handle input features with missing values). In broader machine learning contexts, previous work has enforced "cycle-consistency" in generative adversarial methods for images (Zhu et al., 2017; Zhou et al., 2016) or biomedical data (McDermott et al., 2018). Inspired by these innovations, others have developed cycle-consistent objectives for VAEs (Jha et al., 2018) where the focus was on accurately reconstructing encoding vectors z. Our work is differentiated by its focus on semi-supervised learning, and by the use of cycle-consistency to enforce similarity of predicted labels y. It is easier for domain-experts to select a task-relevant penalty for label similarity than for reconstructed features x or encodings z. Recently, Miller et al. (2019) developed discriminative regularization for VAEs with a direct data-to-label prediction model. Their objective penalizes label distributions inconsistent with a reconstruction-to-prediction distribution and does not handle SSL. Our approach emphasizes instead code-to-label prediction and emphasizes SSL.

3.3 Improved Generative Models: Robust Likelihoods and Spatial Transformers

As our approach is applicable to any generative model, we can incorporate prior knowledge of the data domain to improve both generative and discriminative performance. We consider two examples: likelihoods that model noisy pixels, and explicit affine transformations to model image deformations.

Noise Likelihoods. In place of a normal (or other common) likelihood for pixel intensities, we use a "Noise-Normal" likelihood in our image models. This likelihood is defined as a parameterized 2-component mixture (with pixel-specific weights) of a truncated Normal distribution and a uniform distribution over pixel intensities. This approach allows our model to avoid sensitivity to outliers and noise in the observed images, and boosts the performance of our CPC method for SSL.

Spatial Transformer VAE. Our spatial transformer VAE retains the structure of a standard VAE, but reinterprets the latent code z as two components. We denote the first 6 latent dimensions as z_t, and associate these with 6 affine transformation parameters capturing image translation, rotation, scaling, and shear. The generative model maps each value into a fixed range and creates an affine transformation matrix, M_t, by applying the transformations in a fixed order. The remainder of the

Source	Method	MNIST (100)	SVHN (1000)	NORB (1000)
Tables 1-2 of Kingma et al. (2014)	M1 + M2	96.67% ($\pm$0.14)	63.98% ($\pm$0.10)	-
Table 2 of Maaløe et al. (2016)	ADGM	**99.04**% ($\pm$0.02)	77.14%	89.94% ($\pm$0.05)
Table 2 of Maaløe et al. (2016)	SDGM	98.68% ($\pm$0.07)	83.39% ($\pm$0.24)	90.60% ($\pm$0.04)
Gordon & Hernández-Lobato (2020)	Blended M2	93.05% ($\pm$0.73)	-	-
Tables 3-4 of Miyato et al. (2019)	VAT	98.64% ($\pm$0.03)	**94.23**% ($\pm$0.32)	-
ours, using labeled-set only	MLP / WRN	73.91% ($\pm$1.45)	87.7% ($\pm$1.02)	86.7% ($\pm$1.32)
ours	**CPC VAE**	96.68% ($\pm$0.54)	**94.22**% ($\pm$0.62)	**92.00**% ($\pm$1.21)

Table 1: Test accuracy of SSL methods. Our results show mean (std. dev.) across 10 samples of the labeled set. The labeled-set-only neural net (MLP on MNIST; WRN otherwise) has roughly the same size as our CPC VAE.

Method	MNIST (100)	Method	MNIST (100)
CPC (2 layer)	**96.68**% ($\pm$0.54)	**M1 + M2** (Kingma et al., 2014)	**96.67**% ($\pm$0.14)
CPC (2 layer, w/o aggregate loss)	94.27% ($\pm$3.78)	M2 (1 layer, $\alpha = 0.1$) (Kingma et al., 2014)	88.03% ($\pm$1.71)
CPC (2 layer, w/o transforms)	91.93% ($\pm$1.65)	M2 (2 layer, $\alpha = 0.1$)	83.32% ($\pm$5.22)
CPC (4 layer, w/o transfroms)	93.78% ($\pm$2.25)	M2 (4 layer, $\alpha = 0.1$)	47.05% ($\pm$8.13)
PC (2 layer)	80.49% ($\pm$3.31)	M2 (4 layer, tuned to $\alpha = 10$)	68.15% ($\pm$3.43)

Table 2: Ablation study for SSL on MNIST with 100 labels and encoding size $C = 50$. Unless cited, all results come from our implementation, where encoder and decoder have 1000 units per hidden layer. All our introduced techniques improve accuracy. M2's accuracy deteriorates when its networks are overparameterized, even after tuning α on validation set instead of using default from Kingma et al. (2014), while our method remains stable.

latent code, z_*, is used to generate parameters for independent, per-pixel likelihoods. Assuming normal likelihoods, the output parameters for the pixel with coordinates (i, j) are $\mu_\theta(z_*)_{ij}$, $\sigma_\theta(z_*)_{ij}$.

We re-orient our per-pixel likelihoods according to the affine transform M_t. The parameters of the likelihood for pixel (i, j) will use the decoder outputs at coordinate (i', j'), where M_t defines the affine mapping from (i', j') to (i, j). We apply this transformation in a (sub) differentiable way via a *spatial transformer layer* (Jaderberg et al., 2015) that takes as input M_t and the appropriate parameter maps $\mu_\theta(z_*)$, $\sigma_\theta(z_*)$, and outputs a final set of parameters for the individual pixel likelihoods. As (i', j') may not correspond to integer coordinates, we use bilinear interpolation over an appropriate representation of the likelihood parameters, and appropriately pad the size of the decoder output.

We further account for this special structure in our prediction and consistency constraints. For many applications, we have prior knowledge that small affine transforms should not affect the class of an image, and thus we can define consistency constraints that condition on z_* but not z_t.

4 Experiments

We assess our consistent prediction-constrained (CPC) VAE on two key goals: accurate prediction of labels y given features x (especially when labels are rare) and useful generative modeling of x. We compare to ablations of our own method (without consistency, without spatial transformations) and to external baselines. We report each method's mean and standard deviation in classification accuracy across 10 samples of the labeled set. We trained using ADAM (Kingma & Ba, 2014), with each minibatch containing 50% labeled and 50% unlabeled data. Hyperparameter search used Optuna (Akiba et al., 2019) to maximize accuracy on a labeled validation set. We used either MLP or wide residual nets (WRN) (Zagoruyko & Komodakis, 2016). Reproducible details are in appendices.

SSL classification on MNIST with thorough internal comparisons. In Table 2 we compare several variations of our CPC methods and the M2 model on an SSL version of MNIST (LeCun et al., 2010, 10 classes,$|\mathcal{D}^S| = 100$,$|\mathcal{D}^U| = 49900$, 10000 validation, 10000 test).

SSL classification on SVHN and NORB. Table 1 compares our methods on two standard SSL tasks: Street-View Housing Numbers (SVHN) (Netzer et al., 2011, 10 classes, $|\mathcal{D}^S| = 1000$, $|\mathcal{D}^U| = 62257$, 10000 validation, 26032 test) and the NYU Object Recognition Benchmark (NORB) (LeCun et al., 2004, 5 classes, $|\mathcal{D}^S| = 1000$, $|\mathcal{D}^U| = 21300$, 2000 validation, 24300 test).

Across all evaluations, we can conclude:

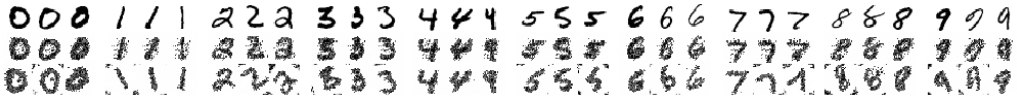
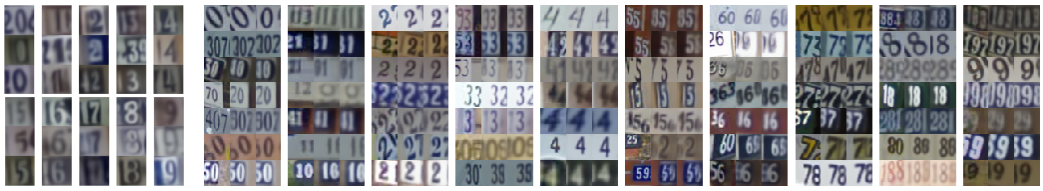


Figure 3: Sampled reconstructions used to compute the consistency loss during training. *Top:* Original image. *Middle:* Sampled reconstructions using a "Noise-Normal" likelihood. *Bottom:* Sampled reconstructions with spatial affine transformations sampled from the prior.



(a) Prior samples (b) Reconstructions

Figure 4: Visualizations of generative model performance on SVHN for a prediction and consistency constrained VAE incorporating latent affine transformations, trained on the SVHN dataset with all labels. (4a) Samples from the learned generative model conditioned on class (more are shown in supplement B). Samples are chosen via rejection sampling in the latent space with a threshold of 95% confidence in the target class. (4b) Reconstructions of images in the held-out test set. Each triplet shows the original image (left), the reconstruction (middle), and an aligned reconstruction (right) obtained by fixing the learned affine transform variables to the global mean.

Both consistency and prediction constraints are needed for high accuracy. In Table 2, PC alone gets 80% accuracy on 100-label MNIST, while adding consistency yields 97% for CPC. The benefits of CPC over PC in both accuracy and latent interpretability are visible in Figs. 1-2. Our *aggregate label consistency* improves robustness, reducing the *variance* in CPC accuracy (from 3.78^2 to 0.54^2).

CPC training delivers strong improvements in SSL prediction quality over baselines. In Table 1, our CPC achieves 94.22% and 92.0% on the challenging 1000-label SVHN and NORB benchmarks, which surpasses by >1.4% all reported baselines while being *reliable* across runs. The M1+M2 baseline (Kingma et al., 2014) is not a coherent generative model, but rather a discriminative ensemble of multiple models. It performs well on MNIST, but very poorly on the more challenging SVHN.

CPC delivers better generative performance; it is not all about prediction. We improve on unsupervised VAEs by explicitly learning latent representations informed by class labels (Fig. 2). In Fig. 3 and Fig. 4, we show visually-plausible class-conditional samples from our best CPC models. Additional visualizations of learned VAE and CPC-VAE models are included in the supplement.

With improved generative models, CPC can improve predictions. Fig. 4 shows that including spatial transformations allows learning a canonical orientation and scale for each digit. This generative improvement boosts classifier accuracy (e.g., MNIST improves from 91.9% to to 96.7% in Table 2).

5 Conclusion

We have developed a new optimization framework for semi-supervised VAEs that can balance discriminative and generative goals. Across image classification tasks, our CPC-VAE method delivers superior accuracy and label-informed generative models with visually-plausible samples. Unlike previous efforts to enforce constraints on latent variable models, such as expectation constraints (Mann & McCallum, 2010), posterior regularization (Zhu et al., 2014; 2012), posterior constraints (Ganchev et al., 2010), or prediction constraints for topic models (Hughes et al., 2018), our approach is the only one that coherently and simultaneously treats *uncertainty* in latent variables z, applies to flexible "deep" non-conjugate models, and offers scalable training and test evaluation via amortized inference.

A further contribution is demonstrating the necessity of *consistency* for improving discrimination. Our CPC approach is an antidote to model misspecification: the constraints on prediction quality and consistency prevent the model from learning a generative model that is unaligned with the classification task or that overfits with more flexible generative models (as M2 is vulnerable to do). As we show with spatial transformers, our work lets improvements in generative model quality directly improve semi-supervised label prediction, helping realize the promise of deep generative models.

REFERENCES

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin Raffel. MixMatch: A Holistic Approach to Semi-Supervised Learning. In *Advances in Neural Information Processing Systems*, 2019. URL http://arxiv.org/abs/1905.02249.

Rodrigo de Bem, Arnab Ghosh, Thalaiyasingam Ajanthan, Ondrej Miksik, N. Siddharth, and Philip Torr. A Semi-supervised Deep Generative Model for Human Body Analysis. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, 2018. URL http://openaccess.thecvf.com/content_eccv_2018_workshops/w11/html/de_A_Semi-supervised_Deep_Generative_Modelfor_Human_Body_Analysis_ECCVW_2018_paper.html.

Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. Adversarially Learned Inference. In *International Conference on Learning Representations (ICLR)*, 2017.

Michael Figurnov, Shakir Mohamed, and Andriy Mnih. Implicit reparameterization gradients. In *Advances in Neural Information Processing Systems*, 2018.

Kuzman Ganchev, João Graça, Jennifer Gillenwater, and Ben Taskar. Posterior Regularization for Structured Latent Variable Models. *Journal of Machine Learning Research*, 11:2001–2049, 2010.

Jonathan Gordon and José Miguel Hernández-Lobato. Combining deep generative and discriminative models for Bayesian semi-supervised learning. *Pattern Recognition*, 100, 2020.

Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. In *Proceedings of the 17th International Conference on Neural Information Processing Systems*, NIPS'04, pp. 529–536, Cambridge, MA, USA, 2004. MIT Press.

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 *International Conference on Learning Representations (ICLR)*, 2017.

Michael C. Hughes, Gabriel Hope, Leah Weiner, Thomas H. McCoy, Roy H. Perlis, Erik B. Sudderth, and Finale Doshi-Velez. Semi-Supervised Prediction-Constrained Topic Models. In *Artificial Intelligence and Statistics*, 2018. URL http://proceedings.mlr.press/v84/hughes18a.html.

Max Jaderberg, Karen Simonyan, Andrew Zisserman, and koray kavukcuoglu. Spatial transformer networks. In *Advances in Neural Information Processing Systems*, 2015. URL http://papers.nips.cc/paper/5854-spatial-transformer-networks.pdf.

Ananya Harsh Jha, Saket Anand, Maneesh Singh, and V. S. R. Veeravasarapu. Disentangling Factors of Variation with Cycle-Consistent Variational Auto-encoders. In *European Conference on Computer Vision (ECCV)*. Springer International Publishing, 2018.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*, 2014. URL http://arxiv.org/abs/1412.6980.

Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations*, 2014. URL http://arxiv.org/abs/1312.6114.

Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, 2014. URL https://papers.nips.cc/paper/5352-semi-supervised-learning-with-deep-generative-models.pdf.

Abhishek Kumar, Prasanna Sattigeri, and Tom Fletcher. Semi-supervised Learning with GANs: Manifold Invariance with Improved Inference. In *Advances in Neural Information Processing Systems*, 2017. URL https://papers.nips.cc/paper/7137-semi-supervised-learning-with-gans-manifold-invariance-with-improved-inference.pdf.

Simon Lacoste-Julien, Ferenc Huszár, and Zoubin Ghahramani. Approximate inference for the loss-calibrated bayesian. In *Artificial Intelligence and Statistics*, 2011.

Samuli Laine and Timo Aila. Temporal Ensembling for Semi-Supervised Learning. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/pdf?id=BJ6oOfqge.

Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In *International Conference on Machine Learning*, pp. 1558–1566, 2016. URL http://proceedings.mlr.press/v48/larsen16.html.

Y LeCun, F. J. Huang, and L. Bottou. Learning Methods for Generic Object Recognition with Invariance to Pose and Lighting. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2004. URL http://yann.lecun.com/exdb/publis/pdf/lecun-04.pdf.

Yann LeCun, Corinna Cortes, and CJ Burges. MNIST handwritten digit database, 2010. URL http://yann.lecun.com/exdb/mnist/.

Lars Maaløe, Casper Kaae Sønderby, Søren Kaae Sønderby, and Ole Winther. Auxiliary Deep Generative Models. *arXiv:1602.05473 [cs, stat]*, 2016. URL http://arxiv.org/abs/1602.05473.

Gideon S Mann and Andrew McCallum. Generalized expectation criteria for semi-supervised learning with weakly labeled data. *Journal of Machine Learning Research*, 11(Feb):955–984, 2010.

Matthew B A McDermott, Tom Yan, Tristan Naumann, Nathan Hunt, Harini Suresh, Peter Szolovits, and Marzyeh Ghassemi. Semi-Supervised Biomedical Translation with Cycle Wasserstein Regression GANs. In *Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, pp. 8, 2018. URL https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/viewFile/16938/15951.

Andrew C Miller, Ziad Obermeyer, John P Cunningham, and Sendhil Mullainathan. Discriminative Regularization for Latent Variable Models with Applications to Electrocardiography. In *International Conference on Machine Learning*, pp. 10, 2019. URL https://proceedings.mlr.press/v97/miller19a/miller19a.pdf.

Takeru Miyato, Shin-Ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual Adversarial Training: A Regularization Method for Supervised and Semi-Supervised Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(8):1979–1993, 2019. URL https://ieeexplore.ieee.org/document/8417973/.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading Digits in Natural Images with Unsupervised Feature Learning. In *NeurIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011. URL http://ufldl.stanford.edu/housenumbers.

Avital Oliver, Augustus Odena, Colin Raffel, Ekin D. Cubuk, and Ian J. Goodfellow. Realistic Evaluation of Deep Semi-Supervised Learning Algorithms. *arXiv:1804.09170 [cs, stat]*, 2018. URL http://arxiv.org/abs/1804.09170.

N. Siddharth, Brooks Paige, Jan-Willem van de Meent, Alban Desmaison, Noah D. Goodman, Pushmeet Kohli, Frank Wood, and Philip H. S. Torr. Learning Disentangled Representations with Semi-Supervised Deep Generative Models. In *Advances in Neural Information Processing Systems*, 2017. URL http://arxiv.org/abs/1706.00400.

Veselin Stoyanov, Alexander Ropson, and Jason Eisner. Empirical risk minimization of graphical model parameters given approximate inference, decoding, and model structure. In *Artificial Intelligence and Statistics*, 2011.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In Edwin R. Hancock Richard C. Wilson and William A. P. Smith (eds.), *Proceedings of the British Machine Vision Conference (BMVC)*, pp. 87.1–87.12. BMVA Press, September 2016. ISBN 1-901725-59-6. doi: 10.5244/C.30.87. URL https://dx.doi.org/10.5244/C.30.87.

Xiang Zhang, Lina Yao, and Feng Yuan. Adversarial Variational Embedding for Robust Semi-supervised Learning. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 139–147, Anchorage AK USA, 2019. ACM.

Tinghui Zhou, Philipp Krahenbuhl, Mathieu Aubry, Qixing Huang, and Alexei A. Efros. Learning Dense Correspondence via 3D-Guided Cycle Consistency. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 117–126, Las Vegas, NV, USA, 2016. IEEE.

Jun Zhu, Amr Ahmed, and Eric P Xing. MedLDA: Maximum margin supervised topic models. *The Journal of Machine Learning Research*, 13(1):2237–2278, 2012.

Jun Zhu, Ning Chen, and Eric P Xing. Bayesian inference with posterior regularization and applications to infinite latent SVMs. *Journal of Machine Learning Research*, 15(1):1799–1847, 2014.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2242–2251, Venice, 2017. IEEE.

Xiaojin Zhu. Semi-Supervised Learning Literature Survey. Technical Report Technical Report 1530, Department of Computer Science, University of Wisconsin Madison., 2005.

A CODE AVAILABILITY

During the review process, anonymized code for our methods is available. After review, we will publish all code under a permissive open source license.

Download a ZIP file here: `https://www.dropbox.com/s/b43xcsnhs5kevue/PC-VAE-REVIEW-RELEASE.zip?dl=1`

Browse in your web browser here: `https://anonymous.4open.science/repository/0fa4ef53-5e92-40df-ad0a-bcdab28f6df0/pcvae/`

B DETAILS AND VISUALIZATIONS OF GENERATIVE MODELS

B.1 NOISE-NORMAL LIKELIHOOD

As discussed in Sec. 4.3, we use a "Noise-Normal" distribution as the pixel likelihood for many of our experiments. We define this distribution to be a parameterized two-component mixture of a *truncated-normal* distribution and a *uniform* distribution. We will use ρ to denote the mixture probability of the Normal component, and μ and σ to denote the mean and standard deviation of the truncated-normal, respectively. The generative model (or decoder) predicts a distinct outlier probability $(1-\rho)$ for each pixel. We assume that pixel intensities are defined on the domain $[-1, 1]$ and rescale our datasets to match. We can write the *probability density function* of the Noise-Normal distribution via the standard normal PDF $\phi(\cdot)$, and standard normal CDF $\Phi(\cdot)$, as follows:

$$f(x \mid \rho, \mu, \sigma) = \rho \left(\frac{\phi\left(\frac{x-\mu}{\sigma}\right)}{\Phi\left(\frac{1-\mu}{\sigma}\right) - \Phi\left(\frac{-1-\mu}{\sigma}\right)} \right) + (1-\rho)\left(\frac{1}{2}\right). \tag{12}$$

We can similarly express the *cumulative distribution function* of the Noise-Normal distribution as:

$$F(x \mid \rho, \mu, \sigma) = \rho \left(\frac{\Phi\left(\frac{x-\mu}{\sigma}\right) - \Phi\left(\frac{-1-\mu}{\sigma}\right)}{\Phi\left(\frac{1-\mu}{\sigma}\right) - \Phi\left(\frac{-1-\mu}{\sigma}\right)} \right) + (1-\rho)\left(\frac{x+1}{2}\right). \tag{13}$$

In order to propagate gradients through the sampling process of the noise-normal distribution, we use the *implicit reparameterization gradients* approach of Figurnov et al. (2018). Given a sample x drawn from this distribution, we compute the gradient with respect to the parameters ρ, μ, and σ as:

$$\nabla_{\rho,\mu,\sigma} x = \frac{-\nabla_{\rho,\mu,\sigma} F(x \mid \rho, \mu, \sigma)}{f(x \mid \rho, \mu, \sigma)}. \tag{14}$$

When fitting the parameters of this distribution using gradient descent, we enforce the constraints that $\rho \in [0, 1]$, $\mu \in [-1, 1]$, and $\sigma > 0$. To do this, we optimize unconstrained parameters ρ_*, μ_*, σ_*, and then define $\rho = \text{sigmoid}(\rho_*)$, $\mu = \tanh(\mu_*)$, and $\sigma = \text{softplus}(\sigma_*)$.

B.2 SPATIAL TRANSFORMER VAE

We now describe how to sample affine transformations M_t for use in our generative model of images. As described in Sec. 4.3, the latent transformation code z_t has 6 real-valued dimensions, each corresponding to one of the following 6 affine transformation parameters:

- $z_t^{(1)} \rightarrow$ *horizontal translation*,
- $z_t^{(2)} \rightarrow$ *vertical translation*,
- $z_t^{(3)} \rightarrow$ *rotation*,
- $z_t^{(4)} \rightarrow$ *shear*,
- $z_t^{(5)} \rightarrow$ *horizontal scale*,
- $z_t^{(6)} \rightarrow$ *vertical scale*.

To constrain our transformations to a fixed range of plausible values, we construct M_t using parameters $\bar{z}_t^{(i)} = \tanh(z_t^{(i)})$ that are first mapped to the interval $[-1, +1]$, and then linearly rescaled to an appropriate range via hyperparameters $\alpha^{(1)}, \ldots, \alpha^{(6)}$. Figure 5 illustrates that the induced prior for

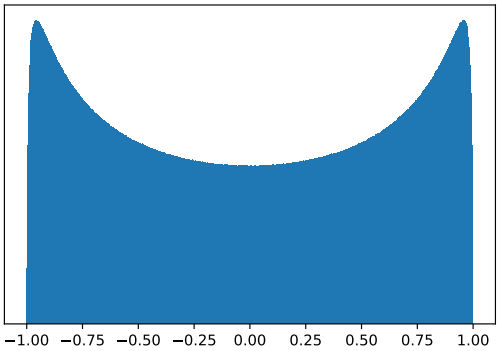


Figure 5: Prior distribution for latent parameters $\bar{z}_t^{(i)} = \tanh(z_t^{(i)})$ used to represent affine transformations.

$\bar{z}_t^{(i)}$ is heaviest for extreme values, encouraging aggressive augmentation when sampling from the prior. The mapping function could be changed to modify this distribution for other applications.

Given these latent transformation parameters, we define an affine transformation matrix M_t as follows:

$$M_t = \begin{bmatrix} 1 & 0 & \alpha^{(1)}\bar{z}_t^{(1)} \\ 0 & 1 & \alpha^{(2)}\bar{z}_t^{(2)} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos(\alpha^{(3)}\bar{z}_t^{(3)}) & -\sin(\alpha^{(3)}\bar{z}_t^{(3)}\alpha^{(4)}\bar{z}_t^{(4)}) & 0 \\ \sin(\alpha^{(3)}\bar{z}_t^{(3)}) & \cos(\alpha^{(3)}\bar{z}_t^{(3)}\alpha^{(4)}\bar{z}_t^{(4)}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} (\alpha^{(5)})^{\bar{z}_t^{(5)}} & 0 & 0 \\ 0 & (\alpha^{(6)})^{\bar{z}_t^{(6)}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (15)$$

To determine the parameters of the likelihood function for the pixel at coordinate (i, j), we use the generative model (or decoder) output at the pixel (i', j') for which

$$\begin{bmatrix} i \\ j \\ 1 \end{bmatrix} = M_t \begin{bmatrix} i' \\ j' \\ 1 \end{bmatrix}. \quad (16)$$

This corresponds to applying horizontal and vertical scaling, followed by rotation and shear, followed by translation. We use the *spatial transformer layer* proposed by Jaderberg et al. (2015) with bilinear interpolation to apply this transformation with non-integer pixel coordinates. For the Noise-Normal distribution we independently interpolate the ρ, μ, and σ^2 parameters.

B.3 Class-conditional sampling

A standard VAE generates data by sampling $z \sim \mathcal{N}(0, I)$, and then sampling $x \sim \mathcal{N}(\mu_\theta(z), \sigma_\theta(z))$, or an alternative like the Noise-Normal likelihood. For the PC-VAE or CPC-VAE, we can further sample images conditioned on a particular class label. As labels are not explicitly part of the generative model, we accomplish this by sampling images that would be confidently predicted as the target class. We use a rejection sampler, repeatedly sampling $z \sim \mathcal{N}(0, I)$ until a sample meets the criteria: $p_w(y \mid z) > 1 - \epsilon$, for some target threshold ϵ. We typically use $\epsilon = 0.05$ in our experiments.

MNIST digit samples for models with a 2-D latent space. Fig. 2 in the main text shows 2-dimensional latent space encodings of the MNIST dataset using several different models. We provide a complementary visualization of generative models in Fig. 6, where we compare class-conditional samples for three of these models. The unsupervised VAE's encodings of some classes (e.g., 2's and 4's and 8's and 9's) are not separated, and samples thus frequently appear to be the wrong class. Model M2 (Kingma et al., 2014) explicitly encodes the class label as a latent variable, but nevertheless many sampled images do not visually match the conditioned class. In contrast, for our CPC-VAE model almost all samples are easily recognized as the target class.

(a) Unsupervised VAE (b) CPC-VAE (c) M2

Figure 6: Class-conditional samples of the 10 possible digit classes in the MNIST dataset. Each column shows multiple samples from one specific digit class. From left to right, each panel shows samples from a standard unsupervised VAE, our CPC-VAE, and model M2 (Kingma et al., 2014). All models use a 2-dimensional latent code, and are trained on the MNIST dataset with 100 labeled examples (10 per class).

Figure 7: Class-conditional samples of the 10 possible digit classes in the SVHN dataset, extending figure 4a. The generative model was trained on the fully labeled SVHN dataset with prediction and consistency constraints. Samples were chosen via rejection sampling in the latent space with a threshold of 95% confidence in the target class.

C Experimental Protocol

Here we provide details about models and experiments which did not fit into the primary paper.

C.1 Hyperparameter optimization

The hyperparameter search for all models, including the CPC-VAE and various baselines, used Optuna (Akiba et al., 2019) to achieve the best accuracy on a labeled validation set. For our 2-layer and 4-layer M2 experiments, we used our own implementation (available in our code release) and followed the hyperparameters used by the original authors. For the 4-layer variant, we tested 10 different settings of α, ranging from 0.05 to 50, reporting both the the result using the original suggested value for α (0.1) and the best value for α we found for the setting (10). For M2, we also dynamically reduced the learning rate when the validation loss plateaued.

C.2 Network architectures

For our PC-VAE and CPC-VAE models of the MNIST data, we use fully-connected encoder and decoder networks with two hidden layers, 1000 hidden units per layer, and softplus activation functions. Like the M2 model (Kingma et al., 2014), we use a 50-dimensional latent space. The original M2 experiments used networks with a single hidden layer of 500 units. We compare this to replications with networks matching ours, as well as 4-layer networks.

For the SVHN and NORB datsets, we adapt the wide-residual network architecture (WRN-28-2) (Zagoruyko & Komodakis, 2016) that was proposed as a standard for semi-supervised deep learning research in Oliver et al. (2018). In particular, we use this architecture for our encoder with two notable changes: We replace the final global average pooling layer with 3 dense layers with 1000 hidden

units, and add a final dense layer that outputs means and variances for the latent space. We find that the dense layers provide the capacity needed for accomplishing both generative and discriminative tasks with a single network. For the decoder network we use a "mirrored" version of this architecture, reversing the order of layer sizes used, replacing convolutions with transposed convolutions, and removing pooling layers. We maintain the residual structure of the network. Our best classification results with this architecture were achieved with a latent space dimension of 200.

C.3 BETA-VAE REGULARIZATION

As an additional form of regularization for our model, we allow our hyperparameter optimization to adjust a weight on the KL-divergence term in the variational lower bound, which we call $\boldsymbol{\beta}$ as in previous work (Higgins et al., 2017):

$$\mathcal{L}_{\beta}^{\text{VAE}}(x;\theta,\phi) = \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z)\right] + \boldsymbol{\beta} \cdot \mathbb{E}_{q_\phi(z|x)}\left[\log \frac{p(z)}{q_\phi(z \mid x)}\right] \tag{17}$$

This allows us to encourage $q_\phi(z \mid x)$ to more closely conform to the prior, which may be necessary to balance the scale of the objective, depending on the likelihoods used and the dimensionality of the dataset.

C.4 PREDICTION MODEL REGULARIZATION

We add two standard regularization terms to the prediction model used in our constraint, $\hat{y}_w(y \mid z)$. The first is an ℓ_2 regularizer on the regression weights, $||w||_2^2$, to help reduce overfitting. The second is an entropy penalty. As $\hat{y}_w(z)$ defines a categorical distribution over labels, we compute this as: $-\mathbb{E}_{\hat{y}_w(y|z)}[\log \hat{y}_w(y \mid z)]$, which has been shown to be helpful for semi-supervised learning in Grandvalet & Bengio (2004) and was used as part of the standardized training framework of Oliver et al. (2018). We allowed our hyperparameter optimization approach to select appropriate weights for both terms.

C.5 IMAGE PRE-PROCESSING

For all of our image datasets, we rescale the inputs to the range [-1, 1]. For our NORB classification results, we downsample each image to 48x48 pixels. For our SVHN classification results, we convert images to greyscale to reduce the representational load on our generative model. Before the grayscale conversion, we apply contrast normalization to better disambiguate the colors within each image.

For the SVHN and NORB results, we follow the recommendation of a recent survey of semi-supervised learning methods (Oliver et al., 2018) and apply a single data augmentation technique: random translations by up to 2-pixels in each direction. For generative results, we retained the original color images and trained with full labels.

C.6 LIKELIHOODS

For all of our image datasets, we use the Noise-Normal likelihood for our CPC methods. For all experiments on toy data (e.g. half-moon), we used a normal likelihood.

For our implementation of M2 for extensive experiments on MNIST we retained the Bernoulli likelihood used by the original authors (Kingma et al., 2014). That is, we rescaling each pixel's numerical intensity value to the unit interval [0,1], and then sampled binary values from a Bernoulli with probability equal to the intensity.

C.7 SUMMARY OF HYPERPARAMETER SETTINGS FOR FINAL RESULTS

Table 3 below provides all hyperparameter settings used in our experiments.

Hyperparameter	**MNIST (100)**	**SVHN (1000)**	**NORB (1000)**
Encoder/decoder	2 FC layers	WRN-28-2 + 3 FC	WRN-28-2 + 3 FC
Fully connected layer size	1000 units	1000 units	1000 units
Network activations	Softplus	Leaky ReLU	Leaky ReLU
Latent dimension	50	200	200
Pixel likelihood	Noise-Normal	Noise-Normal	Noise-Normal
Prediction multiplier λ	25	140	80
Consistency multiplier γ	4.25λ	1.25λ	4λ
Aggregate consistency penalty	0.1λ	0.2λ	0.2λ
β**-VAE weight**	1	1.3	2
Predictor reg. ($||w||_2^2$**)**	1	1	1
Entropy reg. ($\mathbb{E}_{p_w(y\|z)}[\log p_w(y\|z)]$**)**	0.5λ	0.5λ	0.5λ
Translation range ($\alpha^{(1)} = \alpha^{(2)}$**)**	$0.2 \times$ (image-width)	$0.2 \times$ (image-width)	$0.2 \times$ (image-width)
Rotation range ($\alpha^{(3)}$**)**	0.4 rad	0.5 rad	0.4 rad
Shear range ($\alpha^{(4)}$**)**	0.2 rad	0.2 rad	0.2 rad
Scale range ($\alpha^{(5)} = \alpha^{(6)}$**)**	1.5	1.5	1.5
Optimizer	ADAM	ADAM	ADAM
Learning rate	3×10^{-4}	3×10^{-4}	3×10^{-4}

Table 3: Hyperparameter settings for semi-supervised learning experiments with our CPC-VAE.

D DATASET DETAILS

For each dataset considered in our paper, we provide a more detailed overview of its contents and properties.

D.1 MNIST

Overview. We consider a 10-way exclusive categorization task for MNIST digits.

We use 28-by-28 pixel grayscale images.

Public availability. We will make code to extract our version available after publication.

Data statistics. Statistics for MNIST are shown in Table 4.

split	num. examples	label distribution
labeled train	100	[0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1]
unlabeled train	49900	[0.1 0.11 0.1 0.1 0.1 0.09 0.1 0.1 0.1 0.1]
labeled valid	10000	[0.1 0.11 0.1 0.1 0.1 0.09 0.1 0.1 0.1 0.1]
labeled test	10000	[0.1 0.11 0.1 0.1 0.1 0.09 0.1 0.1 0.1 0.1]

Table 4: MNIST dataset.

D.2 SVHN

Overview. We consider a 10-way exclusive categorization task for SVHN digits.

We use 32x32 pixel grayscale images.

Public availability. We will make code to extract our version available after publication.

Data statistics. Statistics for SVHN are shown in Table 5.

split	num. examples	label distribution
labeled train	1000	[0.10 0.10 0.10 0.10 0.10 0.10 0.10 0.10 0.10 0.10]
unlabeled train	62257	[0.07 0.19 0.15 0.12 0.10 0.09 0.08 0.08 0.07 0.06]
labeled valid	10000	[0.07 0.19 0.14 0.12 0.10 0.09 0.08 0.08 0.07 0.06]
labeled test	26032	[0.07 0.20 0.16 0.11 0.10 0.09 0.08 0.08 0.06 0.06]

Table 5: SVHN dataset.

D.3 NORB

Overview.

We use 48x48 pixel grayscale images.

Public availability. We will make code to extract our version available after publication.

Data statistics. Statistics for NORB are shown in Table 6.

split	num. examples	label distribution
labeled train	1000	[0.2 0.2 0.2 0.2 0.2]
unlabeled train	21300	[0.2 0.2 0.2 0.2 0.2]
labeled valid	2000	[0.2 0.2 0.2 0.2 0.2]
labeled test	24300	[0.2 0.2 0.2 0.2 0.2]

Table 6: NORB dataset.