Joint-stochastic-approximation Autoencoders with Application to Semi-supervised Learning

Wenbo He [1] **Zhijian Ou** [1]

Abstract

Our examination of existing deep generative models (DGMs), including VAEs and GANs, reveals two problems. First, their capability in handling discrete observations and latent codes is unsatisfactory, though there are interesting efforts. Second, both VAEs and GANs optimize some criteria that are indirectly related to the data likelihood. To address these problems, we formally present Joint-stochastic-approximation (JSA) autoencoders - a new family of algorithms for building deep directed generative models, with application to semi-supervised learning. The JSA learning algorithm directly maximizes the data log-likelihood and simultaneously minimizes the inclusive KL divergence the between the posteriori and the inference model. We provide theoretical results and conduct a series of experiments to show its superiority such as being robust to structure mismatch between encoder and decoder, consistent handling of both discrete and continuous variables. Particularly we empirically show that JSA autoencoders with discrete latent space achieve comparable performance to other state-of-the-art DGMs with continuous latent space in semi-supervised tasks over the widely adopted datasets - MNIST and SVHN. To the best of our knowledge, this is the first demonstration that discrete latent variable models are successfully applied in the challenging semi-supervised tasks.

1. Introduction

Semi-supervised learning (SSL) considers the problem of classification when only a small subset of the observations have corresponding class labels, and aims to leverage the large amount of unlabeled data to boost the classification performance. Several broad classes of methods for semi-supervised learning include generative models (Zhu, 2005), transductive SVM (Joachims, 1999), co-training (Blum & Mitchell, 1998), and graph-based methods (see (Zhu, 2005) for more introduction). In recent years, significant progress has been made on representation, learning and inference with Deep Generative Models (DGMs) (Dayan et al., 1995; Hinton et al., 2006; Kingma et al., 2014; Goodfellow et al., 2014; Li et al., 2015; Xu & Ou, 2016), and this stimulates an explosion of interest in utilizing DGMs for semi-supervised learning.

Semi-supervised learning with DGMs usually involves blending unsupervised learning and supervised learning. One justification is that the unsupervised loss (e.g. the negative marginal likelihood over the unlabeled data) provides additional regularization for the supervised loss over the labeled training data (Zhu, 2005; Erhan et al., 2010; Larochelle et al., 2012). Therefore, successful SSL methods often develop or adapt from unsupervised learning methods for DGMs.

DGMs define distributions over a set of variables, consisting of observation variable x and hidden variable (or say latent code) h, often organized in multiple layers. Early forms of DGMs dated back to works on Sigmoid Belief Networks (SBNs) (Saul et al., 1996), Helmholtz machines (Dayan et al., 1995), and probabilistic autoencoders (Zemel, 1994). In recent years, deep generative modeling techniques has been greatly advanced by inventing new models with new learning algorithms, such as Variational Autoencoders (VAEs) (Kingma et al., 2014), Generative Adversarial Networks (GANs) (Goodfellow et al., 2014), auto-regressive neural networks (Larochelle & Murray, 2011) and so on; all are originally proposed in the context of unsupervised learning. Two most prominent techniques are VAEs and GANs, both of which have been successfully adapted to semi-supervised learning. These two also represent two important classes of existing techniques in model representation, learning algorithm, and adaptation to SSL.

VAEs use prescribed generative models (Mohamed & Lakshminarayanan, 2016) and variational learning, i.e. maximize the variational lower bound of the data log-likelihood w.r.t. both the generative and inference models jointly. Adaptation to SSL is straightforward by introducing class label

[1]Department of Electronic Engineering, Tsinghua University, Beijing, China. Correspondence to: Zhijian Ou <ozj@tsinghua.edu.cn>.

y as another latent variable, in addition to h. Remarkably, continuous hidden variables is in dominant use with the reparameterization trick - even when the underlying modality is inherently discrete. Using discrete hidden variables in VAEs still remains a challenge, although there are some prior efforts (Mnih & Gregor, 2014; Jang et al., 2016; Maddison et al., 2016b; van den Oord et al., 2017). The application of VAEs to text data has been far less successful, and recently been improved in (Yang et al., 2017).

GANs use implicit generative models (Mohamed & Lakshminarayanan, 2016) and adversarial learning, i.e. minimize a lower bound on the Jensen-Shannon (JS) divergence between the generator distribution and data distribution, along with a discriminator (Nowozin et al., 2016). Adaptation to SSL either use the $(K+1)$-class discriminative objective (Salimans et al., 2016; Dai et al., 2017; Saatchi & Wilson, 2017) or still use K-class classifier but with various additional regularization terms (Springenberg, 2016; Li et al., 2017) . Remarkably, GANs lack the ability to infer the latent variable given the observation, and this limitation has been addressed by some recent studies (Dumoulin et al., 2016; Donahue et al., 2016). Learning GANs with discrete hidden variables remains unexplored. The application of GANs to discrete data is also rather restricted yet with some efforts (Kusner & Hernández-Lobato, 2016; Yu et al., 2016; Che et al., 2017; Saatchi & Wilson, 2017).

The above examination of VAEs and GANs in the context of unsupervised and semi-supervised learning reveals two problems with existing DGM techniques for SSL. First, their capability in handling discrete observations and latent codes is unsatisfactory, though there are interesting progresses. One fundamental reason is the difficulty of back-propagation of gradients through discrete random variables. Second, although maximum likelihood (ML) has been the de-facto standard for training generative models, both VAEs and GANs optimize some bounds that are indirectly related to the data likelihood.

To address these problems, we note that recently, a new learning algorithm called Joint-stochastic-approximation (JSA) is developed for a broad class of DGMs which are characterized by pairing a generative model (decoder) with an auxiliary inference model (encoder) which approximates the posterior inference in the generative model. The JSA learning algorithm directly maximizes the data log-likelihood and simultaneously minimizes the inclusive KL divergence the between the posteriori and the inference model. We call this new DGM technique by JSA autoencoders, or JAEs for short. Inspired by the success of unsupervised learning with JAEs in (Xu & Ou, 2016), we examine its adaptation to SSL in this paper, which addresses the above two problems with existing DGM techniques for SSL.

The contributions of this work can be summarized as:

(1) We formally introduce Joint-stochastic-approximation autoencoders (JAE) - a new family of algorithms for building deep directed generative models for semi-supervised tasks, and show its theoretical consistency in the nonparametric limit. Two distinctive features of JAEs are that they directly optimize the data log-likelihood and provide a simple, consistent and principled way to handle both discrete and continuous variables in latent and observation space.

(2) Synthetic experiments are given to help us analyze JAEs in-depth and understand their behaviors.

(3) We empirically show that JAEs with discrete latent space achieve comparable performance to other state-of-the-art DGMs with continuous latent space in semi-supervised tasks over the widely adopted datasets - MNIST and SVHN. To the best of our knowledge, this is the first demonstration that discrete latent variable models are successfully applied in the challenging semi-supervised tasks.

2. Related work

In this work, we are mainly concerned with semi-supervised learning with deep generative models. Currently most state-of-the-art SSL methods are based on DGMs. The main idea is that generative training over unlabeled data provides regularization for finding good classifiers (Zhu, 2005). From the perspective of regularization, virtual adversarial training (VAT) (Miyato et al., 2017) seeks virtually adversarial samples to smooth the output distribution of the classifier, temporal ensembling (Laine & Aila, 2016) and mean teacher (Tarvainen & Valpola, 2017) maintain running averages of label predictions and model weights respectively for regularization. These SSL methods also achieve good results. It can be seen that these SSL methods utilize different regularization from SSL with DGMs. Their combination could yield further performance improvement in practice.

SSL with DGMs often develops or adapts from unsupervised learning methods for DGMs. Recently there have emerged a bundle of DGMs, among which VAEs and GANs represent two important classes. For fitting a generative model to data, the optimization criterion used has profound effect on the behavior of the fitted model (Theis et al., 2016). Additionally, some auxiliary model are often introduced to facilitate the optimization, e.g. the inference model in variational learning and JSA learning, the discriminator in adversarial learning. In the following we list a few important optimization criteria (not a complete list) along with the models or learning algorithms which use them. For the sake of clarity, we omit to compare the criteria for optimizing the auxiliary models. Note that JSA learning is distinctive since it directly optimize the data log-likelihood.

- Maximizing the data log-likelihood, or equivalently minimizing the Kullback-Leibler (KL) divergence between the data distribution and the generative model, used by JSA (Xu & Ou, 2016)
- Maximizing the variational lower bound of the data log-likelihood, or equivalently minimizing the KL divergence between the inference model and the posteriori, used by the wake-sleep algorithm (Hinton et al., 1995), NVIL (Mnih & Gregor, 2014), VAEs (Kingma et al., 2014);
- Maximizing the importance sampling (IS) approximated lower bound of the data log-likelihood, used by reweighted wake-sleep (RWS) (Bornschein & Bengio, 2014), importance weighted autoencoders (IWAEs) (Burda et al., 2015);
- Minimizing the JS divergence between the generator distribution and the data distribution, used by GANs.
- Minimizing the Wasserstein distance between the generator distribution and the data distribution, used by WGAN (Arjovsky et al., 2017), WAE (Tolstikhin et al., 2017).

While learning under most criteria is provably consistent given infinite model capacity and data, in practice it learns very different kinds of models with different behaviors, for example, VAEs' mode covering behavior, GANs' mode missing behavior. There are interesting efforts to design new criteria, e.g. connecting the best of GANs and VAEs (Makhzani et al., 2015; Mescheder et al., 2017; Pu et al., 2017) for better sample generation, but few evaluate the improvements over semi-supervised tasks. Remarkably, for adaptation of GANs to SSL using the $(K+1)$-class discriminative objective, it is observed that good semi-supervised classification performance and a good generator cannot be obtained at the same time (Salimans et al., 2016); and it is further analyzed that good semi-supervised learning indeed requires a bad generator (Dai et al., 2017). Nevertheless, among various criteria, maximum likelihood is still appealing due to its nice property (consistency, statistical efficiency, and functional invariance).

For SSL to make up for the lack of labeled training data, good matching of model assumption with the structure of data is critical. We need to handle different modalities in both observation and latent space. The most appropriate latent space may be discrete, continuous or even mixed, depending on the structure of data. However, learning with VAEs and GANs in discrete settings (consisting of three main cases) encounter some difficulties, lagging far behind the progress in continuous settings.

For GANs to work with discrete observations (Case I), it is difficult to propagating gradients back from the discriminator through the generated samples to the generator. For VAEs to work with discrete latent variables (Case II), there exists basically the same difficulty of back-propagation of gradients through discrete random variables[1]. There have some efforts to address this difficulty. For variational learning of discrete latent variable models, the REINFORCE-like trick is employed with various variance-reduction techniques in (Mnih & Gregor, 2014; Mnih & Rezende, 2016); VQ-VAE (van den Oord et al., 2017) approximates the gradient with the straight-through estimator (Bengio et al., 2013); a few studies utilize the Concrete (Maddison et al., 2016a) or Gumbel-softmax (Jang et al., 2016) distribution as a continuous approximation to a multinomial distribution, and then use 'reparameterization trick', although the gradients of these relaxations are biased. For learning GANs on discrete sequences, (Kusner & Hernández-Lobato, 2016) resorts to the Gumbel-softmax distribution; (Yu et al., 2016; Che et al., 2017) models the generation of the discrete sequence as a stochastic policy in reinforcement learning and perform gradient policy update. Remarkably, JAEs do not suffer from such difficulty, since optimizing some expectation w.r.t. discrete variables is solved by stochastic approximation, as we show in the following sections.

The application of VAEs to discrete observations (Case III) has no theoretical difficulty for gradient propagation, but has been far less successful (Bowman et al., 2016; Miao et al., 2016). It is found that the LSTM decoder in textual VAE does not make effective use of the latent code during training. This training collapse problem may reflect structure mismatch between the encoder and decoder, and is alleviated in (Yang et al., 2017) after controlling the contextual capacity of the decoder by using dilated CNN. Structure mismatch between the encoder and decoder causes an irreducible biased gap from the data log-likelihood for VAEs. While this mismatch also affects the statistical efficiency for JAEs, JAE learning is still consistent, since the MIS accept/reject mechanism in JAE learning will compensate for the mismatch.

3. Method

3.1. Background

Our method is an application of the stochastic approximation (SA) framework (Robbins & Monro, 1951), which basically provides a mathematical framework for stochastically solving a root finding problem, which has the form of expectations being equal to zeros. Suppose that the objective

[1]Gradient back-propagation through continuous random variables works by using the 'reparameterization trick'. Both VAEs and GANs use this trick in continuous settings to achieve lower variance in the gradients.

is to find the solution λ^* of $f(\lambda) = 0$ with

$$f(\lambda) = E_{z\sim p(\cdot;\lambda)}[F(z;\lambda)], \tag{1}$$

where $\lambda \in R^d$ is a parameter vector of dimension d, and z is an observation from a probability distribution $p(\cdot;\lambda)$ depending on λ, and $F(z;\lambda) \in R^d$ is a function of z. Given some initialization $\lambda^{(0)}$ and $z^{(0)}$, a general SA algorithm iterates as follows.

1. Generate $z^{(t)} \sim K_{\lambda^{(t-1)}}(z^{(t-1)}, \cdot)$, a Markov transition kernel that admits $p(\cdot;\lambda^{(t-1)})$ as the invariant distribution.
2. Set $\lambda^{(t)} = \lambda^{(t-1)} + \gamma_t F(z^{(t)};\lambda^{(t-1)})$, where γ_t is the learning rate.

During each SA iteration, it is possible to generate a set of multiple observations z by performing the Markov transition repeatedly and then use the average of the corresponding values of $F(z;\lambda)$ for updating λ, which is know as SA with multiple moves (Wang et al., 2017). This technique can help reduce the fluctuation due to slow-mixing of Markov transitions. The convergence of SA has been studied under various regularity conditions, e.g. satisfying that $\sum_{t=0}^{\infty}\gamma_t = \infty$ and $\sum_{t=0}^{\infty}\gamma_t^2 < \infty$. In practice, we can set a large learning rate at the early stage of learning and decrease to $1/t$ for convergence.

3.2. Joint-stochastic-approximation Learning

In the following we present the JSA learning algorithm in a more general form, which was originally proposed in (Xu & Ou, 2016) for learning a broad class of directed generative models.

Consider a generative model[2] $p_\theta(x,h) \triangleq p(h)p_\theta(x|h)$, consisting of observation variable x, hidden variables (or say latent code) h, and parameters θ. It is usually intractable to directly evaluate and maximize the marginal log-likelihood $logp_\theta(x)$, but it is well-known that we have

$$\frac{\partial}{\partial\theta}logp_\theta(x) = E_{p_\theta(h|x)}\left[\frac{\partial}{\partial\theta}logp_\theta(x,h)\right]$$

We can pair the generative model $p_\theta(x,h)$ with an auxiliary inference model $q_\phi(h|x)$, parameterized by ϕ, which approximates the posterior $p_\theta(h|x)$ in the generative model, and jointly train the two models. This basic idea has been proposed and enhanced many times - initially by Helmholtz machines and recently by NVIL (Mnih & Gregor, 2014), VAEs (Kingma et al., 2014), RWS (Bornschein & Bengio, 2014), IWAE (Burda et al., 2015), and so on. The distinctive key idea of JSA learning is that in addition to maximizing w.r.t. θ the marginal log-likelihood, it simultaneously minimizes w.r.t. ϕ the *inclusive* KL divergence $KL(p_\theta(h|x)||q_\phi(h|x))$ between the posteriori and the inference model, and Fortunately, we can use the SA framework to solve the optimization problem.

Suppose that data $\mathcal{D} = \{x_1, \cdots, x_n\}$, which consists of n observations drawn from the true but unknown data distribution $p_0(x)$ with support $\mathcal{X}$. $\tilde{p}(x) \triangleq \frac{1}{n}\sum_{k=1}^{n}\delta(x - x_n)$ denotes the empirical distribution. Then we can formulate the maximum likelihood learning as jointly optimizing

$$\begin{cases} \min\limits_{\theta} KL\left[\tilde{p}(x)||p_\theta(x)\right] \\ \min\limits_{\phi} KL\left[\tilde{p}(x)p_\theta(h|x)||\tilde{p}(x)q_\phi(h|x)\right] \end{cases} \tag{2}$$

By setting the gradients to zeros, the above optimization problem can be solved by finding the root for the following system of simultaneous equations:

$$\begin{cases} E_{\tilde{p}(x)p_\theta(h|x)}\left[\frac{\partial}{\partial\theta}logp_\theta(x,h)\right] = 0 \\ E_{\tilde{p}(x)p_\theta(h|x)}\left[\frac{\partial}{\partial\phi}logq_\phi(h|x)\right] = 0 \end{cases} \tag{3}$$

It can be shown that Eq.(3) exactly follows the form of Eq.(1), so that we can apply the SA algorithm to find its root and thus solve the optimization problem Eq.(2).

Proposition 1. *If Eq.(3) is solvable, then we can apply the SA algorithm to find its root.*

Proof. This can be readily shown by recasting Eq.(3) in the form of $f(\lambda) = 0$, with $\lambda \triangleq (\theta,\phi)^T$, $z \triangleq (k, h_1, \cdots, h_n)^T$, $p(z;\lambda) \triangleq \frac{1}{n}\prod_{k=1}^{n}p_\theta(h_k|x_k)$, and

$$F(z;\lambda) \triangleq \begin{pmatrix} \frac{\partial}{\partial\theta}logp_\theta(x_k,h_k) \\ \frac{\partial}{\partial\phi}logq_\phi(h_k|x_k) \end{pmatrix},$$

where k denotes a uniform index variable over $1, \cdots n$. □

To apply the SA algorithm, we need to construct a Markov transition kernel $K_\lambda(z^{(t-1)}, \cdot)$ that admits $p(z;\lambda)$ as the invariant distribution. There are many options. Particularly, we can use the Metropolis independence sampler (MIS), with $p(z;\lambda)$ as the target distribution. However, the proposal $q(z;\lambda)$ is defined by first drawing k uniformly over $1, \cdots n$, and then only drawing $h_k \sim q_\phi(h_k|x_k)$ without changing other h_j for $j \neq k$. Given current sample $z^{(t-1)}$, MIS works as follow[3]:

[2] It is straightforward to develop the algorithm when the priori $p(z)$ depends on θ.

[3] We update one h_k at a time so that in $\frac{w(z)}{w(z^{t-1})}$, we can cancel the intractable $p_\theta(x_k)$ which is appeared in the importance ratio $w(z)$. In practice, we run SA with multiple moves.

1. Propose $z \sim q(z;\lambda)$,
2. Accept $z^t = z$ with probability

$$min\left\{1, \frac{w(z)}{w(z^{t-1})}\right\}, w(z) = \frac{p_\theta(h_k|x_k)}{q_\phi(h_k|x_k)}.$$

Since the parameters of the target and auxiliary models are jointly optimized based on the SA framework, the above method is referred to as JSA learning. It can be seen from the above derivation that JSA learning is general, which places no constrains on the handling of discrete variables for x and h. In the following, we provide more comments and comparisons with existing learning techniques.

First, note that as in JSA iterations, minimizing $KL(\tilde{p}(x)p_\theta(h|x)||\tilde{p}(x)q_\phi(h|x))$ w.r.t. ϕ encourages the inference model to chase the posteriori, which subsequently improves the sampling efficiency of using the inference model as the proposal for sampling the posteriori. Also the inclusive KL divergence ensures that $q_\phi(h|x) > 0$ wherever $p_\theta(h|x) > 0$, which makes $q_\phi(h|x)$ a valid proposal for sampling $p_\theta(h|x)$.

Second, note that adversarial learning of GANs involves finding a Nash equilibrium to a two-player non-cooperative game. Gradient descent may fail to converge, as analyzed in (Salimans et al., 2016). In contrast, Eq.(2) in JSA learning is not finding a Nash equilibrium, and thus is more stable.

Third, variational learning is to optimize the variational lower bound (VLB):

$$\max_{\theta,\phi} E_{\tilde{p}(x)q_\phi(h|x)} log\left[\frac{p_\theta(x,h)}{q_\phi(h|x)}\right]$$

While the gradient w.r.t. θ is well-behaved, the trouble is that the gradient w.r.t. ϕ is known to have high variance. To address this problem, there are a lot of efforts, as discussed in Related work.

Fourth, note that JSA learning mainly seeks ML estimates of θ, with an additional optimization over ϕ. So JSA estimator of θ enjoys the same theoretical properties as ML estimator, even if q_ϕ has finite capacity. Furthermore, if both p_θ and q_ϕ have infinite capacity, we will obtain not only the perfect generative model but also the perfect inference model. The following proposition shows the theoretical consistency of JSA learning in the nonparametric limit.

Proposition 2. *Suppose that $n \to \infty$, and $p_\theta(x,h)$ and $q_\phi(h|x)$ have infinite capacity, then we have (i) both KL divergences in Eq.(2) can be minimized to attain zeros. (ii) If both KL divergences in Eq.(2) attain zeros at (θ^*, ϕ^*), then we have $p_{\theta^*}(x) = p_0(x)$, $q_{\phi^*}(h|x) = p_{\theta^*}(h|x)$, $x \in \mathcal{X}$.*

Proof. By the property of the KL divergence, and $\tilde{p}(x) \to p_0(x)$ as $n \to \infty$. □

Algorithm 1 Semi-supervised learning with JAEs

repeat

Monte Carlo sampling: Draw a unsupervised minibatch $\mathcal{U} \sim \tilde{p}(x)p_\theta(y,h|x)$ and a supervised minibatch $\mathcal{S} \sim \tilde{p}(x,y)p_\theta(h|x,y)$;

SA updating: Update θ by ascending: $\frac{1}{|\mathcal{U}|}\sum_{(x,y,h)\sim\mathcal{U}} \frac{\partial}{\partial\theta} logp_\theta(x,y,h) + \frac{1}{|\mathcal{S}|}\sum_{(x,y,h)\sim\mathcal{S}} \frac{\partial}{\partial\theta} logp_\theta(x,y,h)$ Update ϕ by ascending: $\frac{1}{|\mathcal{U}|}\sum_{(x,y,h)\sim\mathcal{U}} \frac{\partial}{\partial\phi} logq_\phi(y,h|x) + \frac{1}{|\mathcal{S}|}\sum_{(x,y,h)\sim\mathcal{S}} \frac{\partial}{\partial\phi}\left[logq_\phi(y,h|x) + \alpha logq_\phi(y|x)\right]$

until convergence

Finally, note that $p_\theta(x|h)$ is often termed the decoder, and $q_\phi(h|x)$ the encoder. They could be defined either with multiple stochastic hidden layers such as SBNs (Saul et al., 1996), or with multiple deterministic hidden layers such as in VAEs (Kingma et al., 2014). JSA could be applied in both cases, resulting in JSA autoencoders, or JAEs for short. In this paper, we define them like those in VAEs[4]. Next, we examine SSL with JAEs.

3.3. Semi-supervised Learning with JAEs

In semi-supervised tasks, we consider the generative model $p_\theta(x,y,h) \triangleq p(y)p(h)p_\theta(x|h)$, where, with abuse of notation, the hidden variables are decomposed to the class label y and the latent code h. The inference model is generally denoted as $q_\phi(y,h|x)$. Suppose that among the data $\mathcal{D} = \{x_1, \cdots, x_n\}$, only a small subset of the observations, for example the first m observations, have class labels, $m \ll n$. Denote these labeled data as $\mathcal{L} = \{(x_1,y_1), \cdots, (x_m,y_m)\}$, with $\tilde{p}(x,y)$ representing the empirical distribution. Then we can formulate the semi-supervised learning as jointly optimizing

$$\begin{cases} \min_\theta KL\left[\tilde{p}(x)||p_\theta(x)\right] + KL\left[\tilde{p}(x,y)||p_\theta(x,y)\right] \\ \min_\phi KL\left[\tilde{p}(x)p_\theta(y,h|x)||\tilde{p}(x)q_\phi(y,h|x)\right] \\ + KL\left[\tilde{p}(x,y)p_\theta(h|x,y)||\tilde{p}(x,y)q_\phi(h|x,y)\right] \\ - \alpha \sum_{(x,y)\sim\mathcal{L}} logq_\phi(y|x) \end{cases} \quad (4)$$

which is similar to those SSL criteria used in (Larochelle et al., 2012; Kingma et al., 2014), which are defined by hybrids of generative and discriminative criteria. By setting the gradients to zeros, the above optimization problem can be solved by finding the root for the following system of

[4]In this manner, the storage for saving the latent codes per training observation is much reduced, as compared to (Xu & Ou, 2016).

simultaneous equations:

$$\begin{cases} E_{\tilde{p}(x)p_\theta(y,h|x)}\left[\frac{\partial}{\partial\theta}logp_\theta(x,y,h)\right] \\ + E_{\tilde{p}(x,y)p_\theta(h|x,y)}\left[\frac{\partial}{\partial\theta}logp_\theta(x,y,h)\right] = 0 \\ E_{\tilde{p}(x)p_\theta(y,h|x)}\left[\frac{\partial}{\partial\phi}logq_\phi(y,h|x)\right] \\ + E_{\tilde{p}(x,y)p_\theta(h|x,y)}\left[\frac{\partial}{\partial\phi}logq_\phi(h|x,y)\right] \\ + \alpha \sum_{(x,y)\sim\mathcal{L}} \frac{\partial}{\partial\phi}logq_\phi(y|x) = 0 \end{cases} \tag{5}$$

Similarly, it can be shown that Eq.(5) exactly follows the form of Eq.(1), so that we can apply the SA algorithm to find its root and thus solve the optimization problem Eq.(4). Also we can use MIS to draw samples from the posteriori distributions $p_\theta(y,h|x), p_\theta(h|x,y)$, while using the proposals, $q_\phi(y,h|x), p_\phi(h|x,y)$, from the inference model. The SA algorithm with multiple moves for SSL is shown in Algorithm 1.

In SSL experiments in this paper, the inference network is implemented by $q_\phi(y,h|x) \triangleq q_\phi(y|x)q_\phi(h|x)$. This implementation benefits efficient posteriori inference, which is frequently executed in training. This approximation would be safe as the MIS accept/reject mechanism will compensate for the mismatch between the target distribution and the proposal distribution. This is also empirically validated in our experiments.

4. Experiments

To evaluate JAEs and compare with other SSL methods (mainly VAEs and GANs), we conduct a series experiments. We will release code to reproduce our experiments. Experimental details are provided in Appendix. It is worthwhile to emphasize that:

- In addition to giving the SSL results, we also present some unsupervised learning results, because SSL with DGMs usually involves unsupervised learning. These experiments help us to understand different learning behaviors of different SSL methods, instead of competing for unsupervised performance.

- In addition to showing the benchmarking results over the widely adopted image datasets, experiments with the synthetic datasets are useful for understanding different learning methods since we know the oracle model which creates the data.

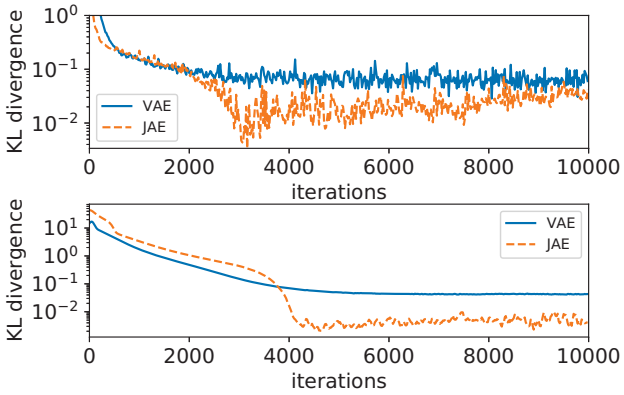

Figure 1. Results for factor analysis. **Upper:** KL divergences between $p_\theta(h|x)$ and $q_\phi(h|x)$ during training. **Lower:** KL divergences between the oracle $p_0(x)$ and the estimated $p_\theta(x)$ during training.

4.1. Factor Analysis

It is known that the encoders used in VAE training are usually not expressive enough to capture the true posterior distribution. They are often modeled as diagonal Gaussians whose means and variances are determined by NN transformations of the observations. Structure mismatch between the encoder and decoder causes an irreducible gap from the data log-likelihood for VAEs. While this mismatch also affects the statistical efficiency for JAEs, JAE learning is still consistent, since the MIS accept/reject mechanism in JAE learning will compensate for the mismatch.

To illustrate this difference, we conduct an experiment over a factor analysis (FA) synthetic dataset. We create 100 3d observations with 2d latent factors as follows:

$$x \sim \mu + Ph + N(0,R), R = 0.04 \times I_3,$$
$$h \sim N(0, I_2),$$
$$\mu = (-1,0,1)^T, P = \begin{pmatrix} 0.2 & 1 & 0.5 \\ 1 & 0.5 & 0.5 \end{pmatrix}^T,$$

We take the generative model $p_\theta(x,h)$ to be the above factor analysis model with unknown parameters $\theta = (\mu, P)$, so that the true posteriori $p_\theta(h|x)$ can be tractably calculated, which is a 2d Gaussian with correlation coefficient 0.66 in the above setting. Also the oracle $p_0(x,h)$ and assumed $p_\theta(x,h)$ are from the same family of models, so that we eliminate the possible bias caused by incorrect model assumption and focus on the effect of structure mismatch between the encoder and decoder.

For both VAE and JAE, the decoder $q_\phi(h|x)$ is implemented as a 2d Gaussian with diagonal covariance matrix. The means and variances are the four outputs from a 3-50-4

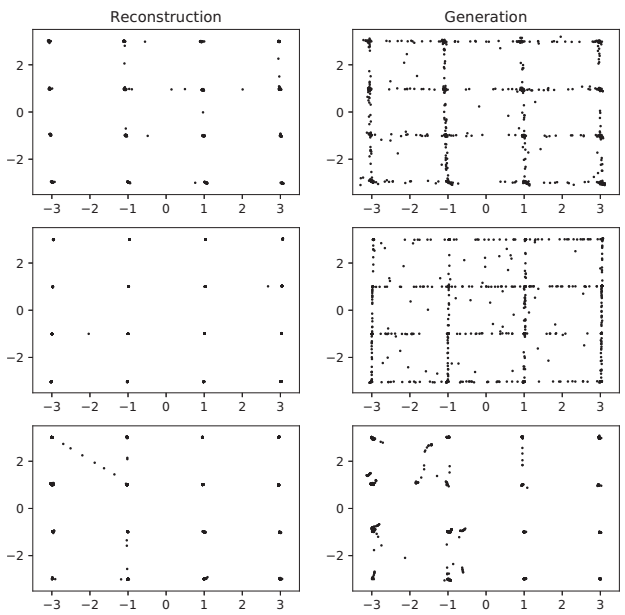

Figure 2. Comparison of a VAE with 2d Gaussian prior for latent code h (row 1), a JAE with 2d Gaussian prior (row 2) and a JAE with a mixture of 4d Bernoulli and 1d Gaussian prior (row 3).

neural network, fully connected, with ReLU activations at the hidden layer and linear activations at the output layer. It can be seen from Figure 1 that while there is structure mismatch between the encoder and decoder for both VAE and JAE, this mismatch prevents VAE from learning the data distribution. In contrast, JAE performs much better even with a mismatched decoder.

4.2. Gaussian Mixture Model

This experiment serves two purposes. First, it help us to understand the reconstruction behaviors of VAEs and JAEs, which also reflect the performances of the inference models. Second, it evaluates the capabilities of VAEs and JAEs for supporting discrete latent modalities. Note that for GANs, their reconstruction performance is still worse than autoencoders like VAEs, as evidenced in recent efforts to augment GANs' inference ability (Dumoulin et al., 2016). So we mainly compare VAEs and JAEs in this experiment, and also in the previous FA experiment for the same reason.

As similarly done in (Dumoulin et al., 2016), we create a synthetic dataset, created by randomly drawing 1600 datapoints from a 2D Gaussian mixture model (GMM) with 16 equally-weighted, low-variance (0.05^2) Gaussian components laid out on a grid This distribution exhibits lots of modes separated by large low-probability regions, which makes it a decently hard task for learning. It is also a good

Training Data	JAE	GAN	VAE
x+x	x+x+x+x+x x	x+x//*x x	xxx x x x x
x-x*x+x-x-x	x/x/x-x/x x	//- x+/x x /	xxx x x x x
x-x+x/x*x+x	x+x+x+x-x x	+ *x*x x x/	xxxxxxxxx x
x*x+x/x	x/x+x x	-x+xx//x*//	xxxxx x x x
x-x+x/x	x/x/x/x+x/x+	/x-x/- * /	xxxxxxxxx x
x*x-x-x-x*x*	x-x-x-x-x-x-	x/x+/x* - +	xxx x x x x
x/x	x-x+x-x/x-x-	x//- ///x	xxxxx x x x
x*x/x	x/x+x x	x/x x*x / x	xxxxxxxxx x
x+x*x+x-x/x/	x/x-x/x-x-x	+x*x/x/x	xxxxx x x x
x/x/x	x+x+x+x x x	/* x+ x	xxxxxxx x x
x/x/x	x/x+x/x x	xxx+/x+x/x	xxxxxxxxx x
x*x+x	x*x+x x x	/x x+x+x*x*x	xxxxxxx x x

Figure 3. Column 1: Part of the training data from the context free grammar; Column 2/3/4 : data generated by JAE, GAN and VAE respectively. Both GAN and VAE use the Gumbel-softmax trick. The GAN result is copied from (Kusner & Hernández-Lobato, 2016). The temperature $\{0.1, 0.01, 0.001\}$ is tested for Gumbel-softmax with VAE.

example of distributions, whose latent space contains both discrete and continuous modalities.

For both VAEs and JAEs, the inference network is implemented by a 3-layer fully connected neural network with 400 units per layer and ReLU activations. The generative network is also a 3-layer fully connected neural network with 200 units per layer and ReLU activations. We compare a VAE with 2d Gaussian prior for latent code h, a JAE also with 2d Gaussian prior, and a JAE with a mixture of 4d Bernoulli and 1d Gaussian prior. The results are summarized in Figure 2.

With 2d Gaussian prior, the JAE and VAE perform similarly for such continuous latent space, in terms of reconstruction and sample generation. However, for this data, the most appropriate latent space is in fact a mixture of discrete and continuous modalities, instead of being purely continuous. It can be seen that the JAE with the mixed latent code performs the best, which successfully discover not only the 16 discrete modes (represented by 4d Bernoulli) but also the variances surrounding each mode. This demonstrates that JAE can consistently handle both discrete and continuous latent codes.

4.3. Sequences of Discrete Elements

In this experiment, we compare VAEs, GANs and GANs for handling discrete observations. Particularly, we consider context free grammar (CFG) learning, as done in (Kusner & Hernández-Lobato, 2016). The CFG is: $S \to x||S+S||S-S||S*S||S/S$, which is used to create sequences with a maximum length of 12 characters. We pad all sequences with less than 12 characters with spaces. There are six characters $\{+, -, *, /, x, space\}$.

Figure 4. Conditional generation by the semi-supervised JAE over the MNIST dataset, using 60d Bernoulli prior. The leftmost column shows images from the test set. The other columns are generated by varying class label y for each column, and keeping latent h inferred from the leftmost column.

We implement VAE and JAE for this learning task. Both models use the same RNN-based seq2seq architecture for encoder-decoder. The decoder is basically a 2-layer (50-6) LSTM, representing $logp_\theta(x_{1...T}|h) = \sum_{t=1}^{T} logp_\theta(x_t|x_{t-1}, h)$, where x_t is the one-hot encoded character and $x_0 = \mathbf{0}$. The prior $p(h)$ is 20d Bernoulli with $\mu = 0.5$. The encoder is basically a 2-layer (50-50) LSTM, representing $q_\phi(h|x_{1...T})$. It calculates the posteriori of h by a single layer neural network form last state of the LSTM.

The sequence generation results are shown in Figure 3. We can see that the generative model of JAE has learned that x_1 should be the character 'x', and the character 'x' and other symbols $+, -, *, /$ should be generated alternately. VAE has learned the importance of 'x' and spaces, but omits other symbols. This result shows that JAEs are superior in learning with sequences of discrete elements.

4.4. Semi-supervised classification benchmarking

We evaluate SSL performances over the widely adopted image datasets - MNIST and SVHN, and strictly follow the experimental setup in previous benchmarking studies. MNIST contains 28x28 gray images of ten digits, consisting of 60k training samples and 10k test samples. SVHN contains 32x32x3 RGB images of ten digits, consisting of 600k training samples (including the extra set) and 26k test samples. We randomly choose 100 samples for MNIST and 1000 samples for SVHN as labeled data, the others in training set is unlabeled.

To demonstrate the ability of JAEs in handling discrete variables, we employ Bernoulli priors $p(h)$ for both binarized MNIST and SVHN. The JAE models run 10 times independently for each dataset with randomly selected labeled data. Table 1 compares the JAE results with state-of-the-art SSL methods. It can be seen that JAEs with discrete latent space achieve comparable SSL performance to other state-of-the-art DGMs with continuous latent space. When compare with the VAE, which uses Gumbel-softmax to handle categorical class but still uses Gaussian latent code (Jang et al., 2016), JAE performs much better on binarized MNIST data. In addition to superior SSL performance, we also show the ability of semi-supervised JAE to disentangle classes and styles. These resembles the two forms of analogical reasoning with VAEs (Kingma et al., 2014), but we show them with discrete latent space. Figure 4 shows the results by fixing the latent codes and then varying the class labels. Figure 5 shows the results by fixing the class label and then varying the latent codes over a range of values.

Figure 5. Class-conditional traversal in the discrete latent space. The center images in the two pictures are the reconstructions. Surrounding images are generated with several units of the latent codes flipped randomly. The number of flipped units follows the board distance to the center.

For SVHN, we find that it is beneficial to pre-process RGB images to gray images, and the error rate is around 6.80% on RGB images. For SVHN, we additionally utilize the discriminative confidence loss to regularize the JAE's inference network, by minimizing $\sum_{x\sim\tilde{p}(x)} \sum_y -q_\phi(y|x) log q_\phi(y|x)$.

5. Conclusion

In this paper, we formally present JSA autoencoders - a new family of algorithms for building deep directed generative models, with application to semi-supervised learning. We provide theoretical results and conduct a series of experiments to show its superiority such as being robust to structure mismatch between encoder and decoder, consistent handling of both discrete and continuous variables. Particularly we empirically show that JSA autoencoders with discrete latent space achieve comparable performance to other state-of-the-art DGMs with continuous latent space in semi-supervised tasks over the widely adopted datasets - MNIST and SVHN. In addition to variational learning and

Table 1. Comparison (% error) between state-of-the-art SSL methods over MNIST using 100 labels and SVHN using 1000 labels, without data augmentation and self-ensembling.

METHODS	MNIST	SVHN	CIFAR-10
LADDER NETWORK (2015)	1.06±0.37	-	20.40±0.47
CATGAN (2016)	1.91±0.10	-	19.58±0.45
IMPROVEDGAN (2016)	0.93±0.06	8.11±1.3	18.63±2.32
ALI (2016)	-	7.42±0.65	17.99±1.62
BADGAN (2017)	0.80±0.01	4.25±0.03	14.41±0.30
TRIPLEGAN (2017)	0.91±0.58	5.77±0.17	16.99±0.36
VAE (2014)	3.33±0.14	36.02±0.10*	-
SDGM (2016)	1.32±0.07	16.61±0.24*	-
ADGM (2016)	0.96±0.02	22.86*	-
ST GUMBEL-SOFTMAX (2016)	6.4	-	-
JAE+GAUSSIAN	1.98±0.07	12.80±0.32*	-
JAE+BERNOULLI	1.96±0.12	6.22±0.55*	44.8

Table 2. Evalution of log-likehood on binary MNIST dataset

METHODS	$logpx \geq$	$logp(x) =$
AVB(8-DIM)	-83.6±0.4	-91.2±0.6
AVB+AC(32-DIM)	-79.5±0.3	-80.2±0.4
VAE(32-DIM)	-87.2±0.2	-81.9±0.4
VAE+NF(T=80)	-85.1	-
VAE+HVI(T=16)	-88.3	-85.5
CONVVAE+HVI(T=16)	-84.1	-81.9
VAE+VGP(2HL)	-81.3	-
DRAW+VGP	-79.9	-
VAE+IAF	-80.8	-79.1
AUXILIARY VAE(L=2)	-83.0	-
IWAE(2L,K=50)	-	-82.9
RWS	-	-86.8
JAE(32-DIM)	-88.0	-80.3

adversarial learning, JSA learning provides another tool in the machine learning toolbox, which deserves more developments.

References

Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

Bengio, Y., Léonard, N., and Courville, A. C. Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR*, abs/1308.3432, 2013.

Blum, A. and Mitchell, T. M. Combining labeled and unlabeled data with co-training. pp. 92–100, 1998.

Bornschein, J. and Bengio, Y. Reweighted wake-sleep. *arXiv preprint arXiv:1406.2751*, 2014.

Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Józefowicz, R., and Bengio, S. Generating sentences from a continuous space. In *CoNLL*, 2016.

Burda, Y., Grosse, R., and Salakhutdinov, R. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

Che, T., Li, Y., Zhang, R., Hjelm, R. D., Li, W., Song, Y., and Bengio, Y. Maximum-likelihood augmented discrete generative adversarial networks. *arXiv: Artificial Intelligence*, 2017.

Chongxuan, L. I., Xu, K., Zhu, J., and Zhang, B. Triple generative adversarial nets. *neural information processing systems*, pp. 4088–4098, 2017.

Dai, Z., Yang, Z., Yang, F., Cohen, W. W., and Salakhutdinov, R. Good semi-supervised learning that requires a bad gan. *arXiv: Learning*, 2017.

Dayan, P., Hinton, G. E., Neal, R. M., and Zemel, R. S. The helmholtz machine. *Neural computation*, 7(5):889–904, 1995.

Donahue, J., Krähenbühl, P., and Darrell, T. Adversarial feature learning. *CoRR*, abs/1605.09782, 2016.

Dumoulin, V., Belghazi, I., Poole, B., Lamb, A., Arjovsky, M., Mastropietro, O., and Courville, A. Adversarially learned inference. *arXiv preprint arXiv:1606.00704*, 2016.

Erhan, D., Bengio, Y., Courville, A. C., Manzagol, P., Vincent, P., and Bengio, S. Why does unsupervised pretraining help deep learning? *Journal of Machine Learning Research*, 11:625–660, 2010.

Goodfellow, I. J., Pougetabadie, J., Mirza, M., Xu, B., Wardefarley, D., Ozair, S., Courville, A. C., and Bengio, Y. Generative adversarial networks. *arXiv: Machine Learning*, 2014.

Hinton, G. E., Dayan, P., Frey, B. J., and Neal, R. M. The "wake-sleep" algorithm for unsupervised neural networks. *Science*, 268(5214):1158–1161, 1995.

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

Jang, E., Gu, S., and Poole, B. Categorical reparameterization with gumbel-softmax. *arXiv: Machine Learning*, 2016.

Joachims, T. Transductive inference for text classification using support vector machines. In *ICML*, 1999.

Kingma, D. P., Rezende, D. J., Mohamed, S., and Welling, M. Semi-supervised learning with deep generative models. In *NIPS*, 2014.

Kusner, M. J. and Hernández-Lobato, J. M. Gans for sequences of discrete elements with the gumbel-softmax distribution. *CoRR*, abs/1611.04051, 2016.

Laine, S. and Aila, T. Temporal ensembling for semi-supervised learning. *arXiv: Neural and Evolutionary Computing*, 2016.

Larochelle, H. and Murray, I. The neural autoregressive distribution estimator. In *AISTATS*, 2011.

Larochelle, H., Mandel, M. I., Pascanu, R., and Bengio, Y. Learning algorithms for the classification restricted boltzmann machine. *Journal of Machine Learning Research*, 13(1):643–669, 2012.

Li, C., Xu, T., Zhu, J., and Zhang, B. Triple generative adversarial nets. In *Advances in Neural Information Processing Systems 30*, pp. 4091–4101. Curran Associates, Inc., 2017.

Li, Y., Swersky, K., and Zemel, R. S. Generative moment matching networks. *international conference on machine learning*, pp. 1718–1727, 2015.

Maaloe, L., Sonderby, C. K., Sonderby, S. K., and Winther, O. Auxiliary deep generative models. *international conference on machine learning*, pp. 1445–1453, 2016.

Maddison, C. J., Mnih, A., and Teh, Y. W. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016a.

Maddison, C. J., Mnih, A., and Teh, Y. W. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv: Learning*, 2016b.

Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., and Frey, B. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

Mescheder, L., Nowozin, S., and Geiger, A. Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks. *arXiv preprint arXiv:1701.04722*, 2017.

Miao, Y., Yu, L., and Blunsom, P. Neural variational inference for text processing. In *ICML*, 2016.

Miyato, T., ichi Maeda, S., Koyama, M., and Ishii, S. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *CoRR*, abs/1704.03976, 2017.

Mnih, A. and Gregor, K. Neural variational inference and learning in belief networks. *international conference on machine learning*, pp. 1791–1799, 2014.

Mnih, A. and Rezende, D. J. Variational inference for monte carlo objectives. *international conference on machine learning*, pp. 2188–2196, 2016.

Mohamed, S. and Lakshminarayanan, B. Learning in implicit generative models. *arXiv: Machine Learning*, 2016.

Nowozin, S., Cseke, B., and Tomioka, R. f-gan: Training generative neural samplers using variational divergence minimization. *neural information processing systems*, pp. 271–279, 2016.

Pu, Y., Chen, L., Dai, S., Wang, W., Li, C., and Carin, L. Symmetric variational autoencoder and connections to adversarial learning. *CoRR*, abs/1709.01846, 2017.

Rasmus, A., Valpola, H., Honkala, M., Berglund, M., and Raiko, T. Semi-supervised learning with ladder networks. *neural information processing systems*, pp. 3546–3554, 2015.

Robbins, H. and Monro, S. A stochastic approximation method. *The Annals of Mathematical Statistics*, pp. 400–407, 1951.

Saatchi, Y. and Wilson, A. G. Bayesian gan. *arXiv: Learning*, 2017.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., and Alec Radford, X. C. Improved techniques for training gans. *arXiv: Learning*, 2016.

Saul, L. K., Jaakkola, T., and Jordan, M. I. Mean field theory for sigmoid belief networks. *Journal of artificial intelligence research*, 4(1):61–76, 1996.

Springenberg, J. T. Unsupervised and semi-supervised learning with categorical generative adversarial networks. *international conference on learning representations*, 2016.

Tarvainen, A. and Valpola, H. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in neural information processing systems*, pp. 1195–1204, 2017.

Theis, L., Den Oord, A. V., and Bethge, M. A note on the evaluation of generative models. *international conference on learning representations*, 2016.

Tolstikhin, I. O., Bousquet, O., Gelly, S., and Schölkopf, B. Wasserstein auto-encoders. *CoRR*, abs/1711.01558, 2017.

van den Oord, A., Vinyals, O., and Kavukcuoglu, K. Neural discrete representation learning. In *NIPS*, 2017.

Wang, B., Ou, Z., and Tan, Z. Learning trans-dimensional random fields with applications to language modeling. *IEEE transactions on pattern analysis and machine intelligence*, 2017.

Xu, H. and Ou, Z. Joint stochastic approximation learning of helmholtz machines. *arXiv preprint arXiv:1603.06170*, 2016.

Yang, Z., Hu, Z., Salakhutdinov, R., and Berg-Kirkpatrick, T. Improved variational autoencoders for text modeling using dilated convolutions. *arXiv: Neural and Evolutionary Computing*, 2017.

Yu, L., Zhang, W., Wang, J., and Yu, Y. Seqgan: Sequence generative adversarial nets with policy gradient. *national conference on artificial intelligence*, pp. 2852–2858, 2016.

Zemel, R. S. *A minimum description length framework for unsupervised learning.* University of Toronto, 1994.

Zhu, X. Semi-supervised learning literature survey. 2005.

A. Visualization

Figure 6 demonstrates the reconstruction result of SVHN data in test set trained by semi-JAE with 220d Bernoulli $p(h)$.

Figure 7 demonstrates the manifold of MNIST dataset, images are represented as a 60d $p(h|x)$ probabilistic vector trained by unsupervised JAE with Bernoulli prior. It suggests that JAE can extract class features and benefit semi-supervised learning.

B. Training Details

B.1. Factor Analysis Dataset

For the synthetic factor analysis dataset, we used neural networks for the inference network with one hidden layer, containing 50 rectified linear units. The output of the inference network is a 4d vector, involve 2d for mean values and 2d for variances of the Gaussian $q(h|x)$. The generative network $x = f(h)$ is a factor generate process that $x = \hat{P}h + \mu$. Architectures for JAE and VAE is the same. Parameters are initializer by the Xavier initialization. We optimizes JAE and VAE by Adam($\alpha = 10^{-2}, \beta_1 = 0.9, \beta_2 = 0.999$) with learning rate 0.01. The size of the dataset is 100, and we run 10000 iterations with full batch data on it.

B.2. Almost Discrete Gaussian Mixture Dataset

Inference networks and generative networks we used are fully connected neural networks for the almost discrete Gaussian mixture dataset. Inference networks for VAE and JSA are networks with 2 hidden layers with shape 400-400 and the ReLU activation. Generative networks are networks 2 hidden layers with shape 200-200 and the ReLU activation. Parameters are initializer by the Xavier initialization. The variance of Gaussian noise attached to the output of JAE's inference network is selected to be 0.05, and is decreased to 0.01 after 1000 iterations. We use the Adam optimizer ($\alpha = 0.05, \beta_1 = 0.9, \beta_2 = 0.999$). The size of the dataset is 1600, batch size is 100, and we run 10000 iterations on it.

B.3. Sequences of Discrete Elements Dataset

Architecture of the models for the discrete sequence dataset is shown in Figure 8. Numbers on the picture denotes widths of layers. We show 4 time steps but LSTM networks run 12 time steps for every sequence. Squares denote LSTM layer and the circle denotes the fully connected layer. All layers without annotation explicitly use tanh activation. Parameters are initializer by the Xavier initialization. SGD optimizers with learning rate 0.0005 are used to train networks. The prior $p(h)$ for JAE and VAE is selected to be 20d Bernoulli distribution with $\mu = 0.5$. The size of the dataset is 5000, batch size is 100, and we run 10000 iterations on it.

B.4. MNIST

We leverage 60d Bernoulli prior of the hidden variables for MNIST. Figure 9 demonstrates networks for MNIST. Dense denotes fully connected layer and noise denotes the addition of gauss ion noise with variance 0.3^2. All layers without annotation explicitly use ReLU activation. Parameters are initializer by the Xavier initialization. The optimizer is SGD with learning rate 0.001. The MIS process work without cache and restart for every datapoint with 10-step warm up. After 100 epochs, the learning rate decays to 0.995 times it for every epoch. We use 100 randomly selected images as labeled data, and use the other 50k images as unlabeled data. The accuracy is evaluated on the test set with 10k images. The algorithm runs $1.2 * 10^5$ iterations, and for every iteration parameters are updated by 100 labeled data and 100 unlabeled data. For first 500 iterations, only supervised learning is executed.

B.5. SVHN

We leverage 220d Bernoulli prior of the hidden variables for SVHN. Figure 10 demonstrates networks for SVHN. BN denotes batch normalization. Dense denotes fully connected layer. And the keep probability for dropout we used is 0.8. All layers without annotation explicitly use ReLU activation. The variance of the Gaussian noise attached to the output of JAE's inference network is selected to be 0.1. Parameters are initializer by the Xavier initialization and biases are set to be zero. Regulation are added into the criterion. Minimizing $H(p(y|x))$ for unlabeled data and the pseudo loss $KL(\sum_h p(x, y, h)||q(z|x)p(x))$ improves the performance of the network. The optimizer is SGD with learning rate 0.00005. After 10^5 iterations, the learning rate decays to 0.995 times it for every 100 iterations. The MIS process work without cache and restart for every datapoint with 10-step warm up. We use 1000 randomly selected images as labeled data,

and use the other 600k images as unlabeled data. The accuracy is evaluated on the test set with 26k images. The algorithm runs $1.2 * 10^5$ iterations, and for every iteration parameters are updated by 50 labeled data and 500 unlabeled data. For first 1000 iterations, only supervised learning is executed.

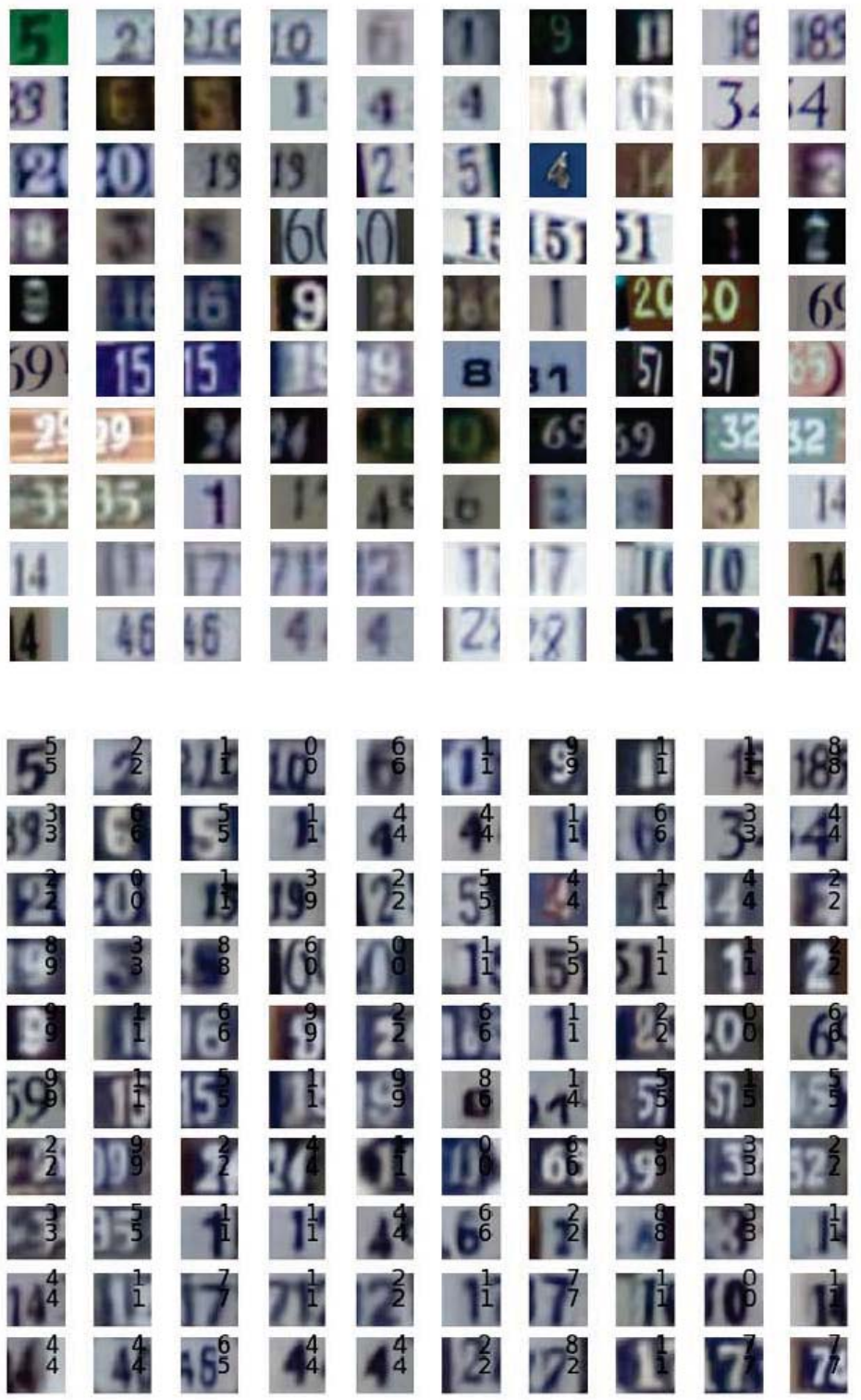

Figure 6. Reconstructions (lower) for the images in the SVHN test set (upper), with 220d Bernoulli $p(h)$. The two digits at the right of the lower picture are the true labels and predicted labels respectively.

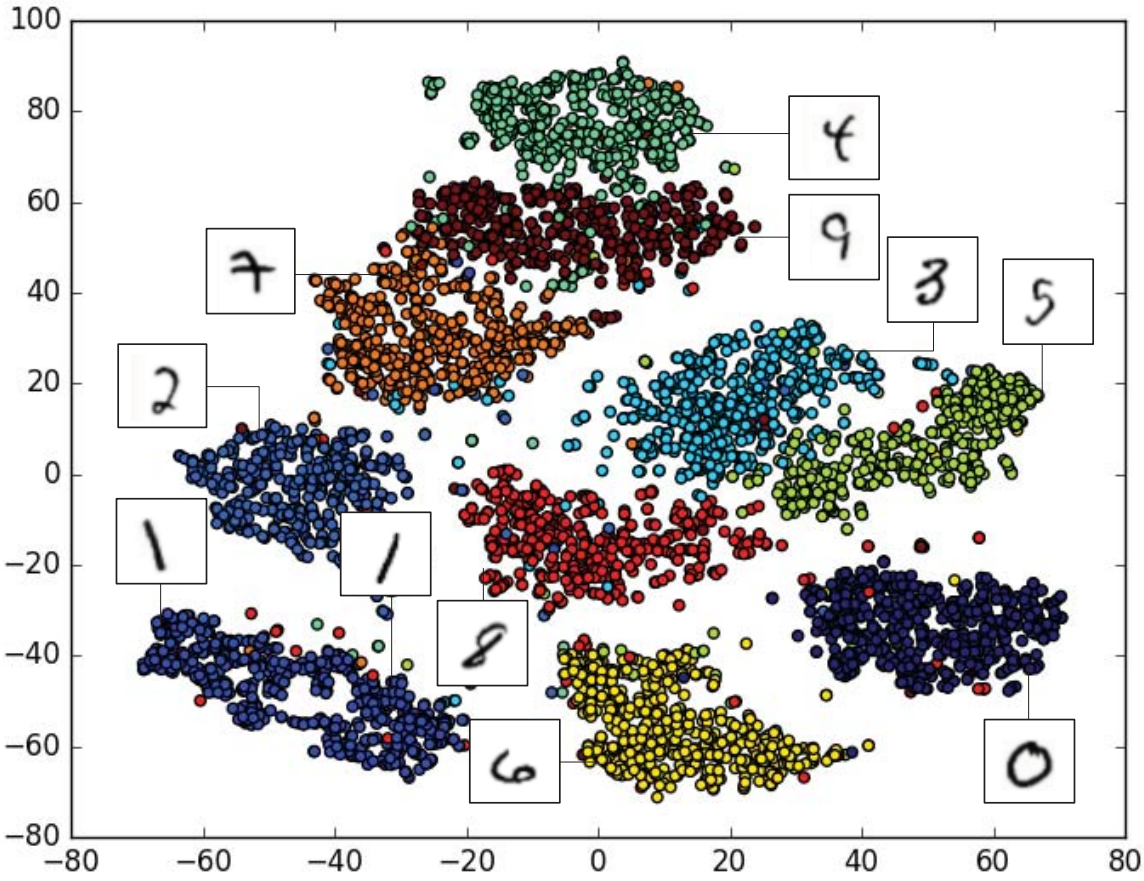

Figure 7. t-SNE of 60d latent codes inferred by unsupervised JAE with 60d Bernoulli $p(h)$. Different colors represent different class variables in MNIST dataset.

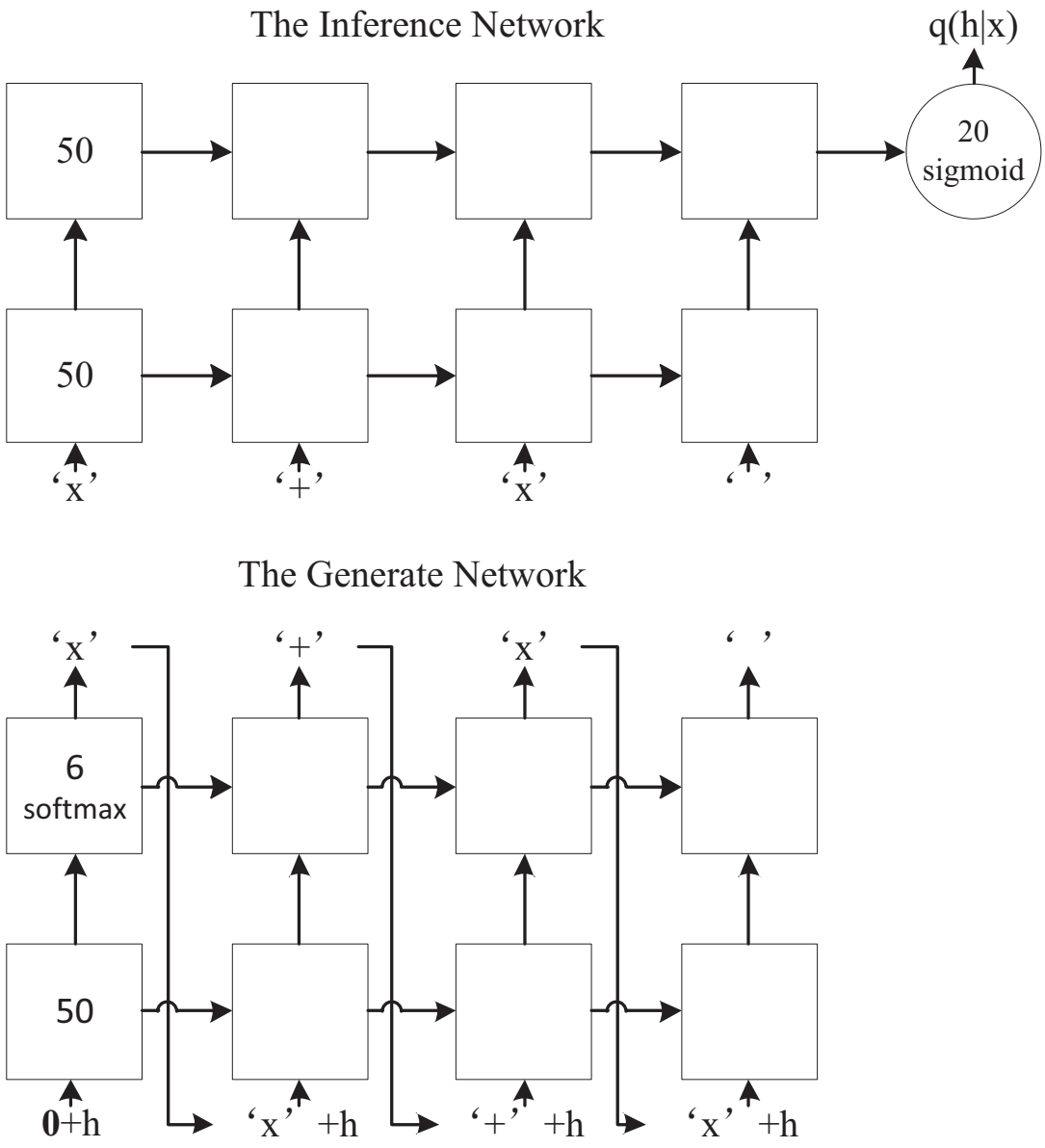

Figure 8. Architecture of the JAE model on context free grammar dataset.

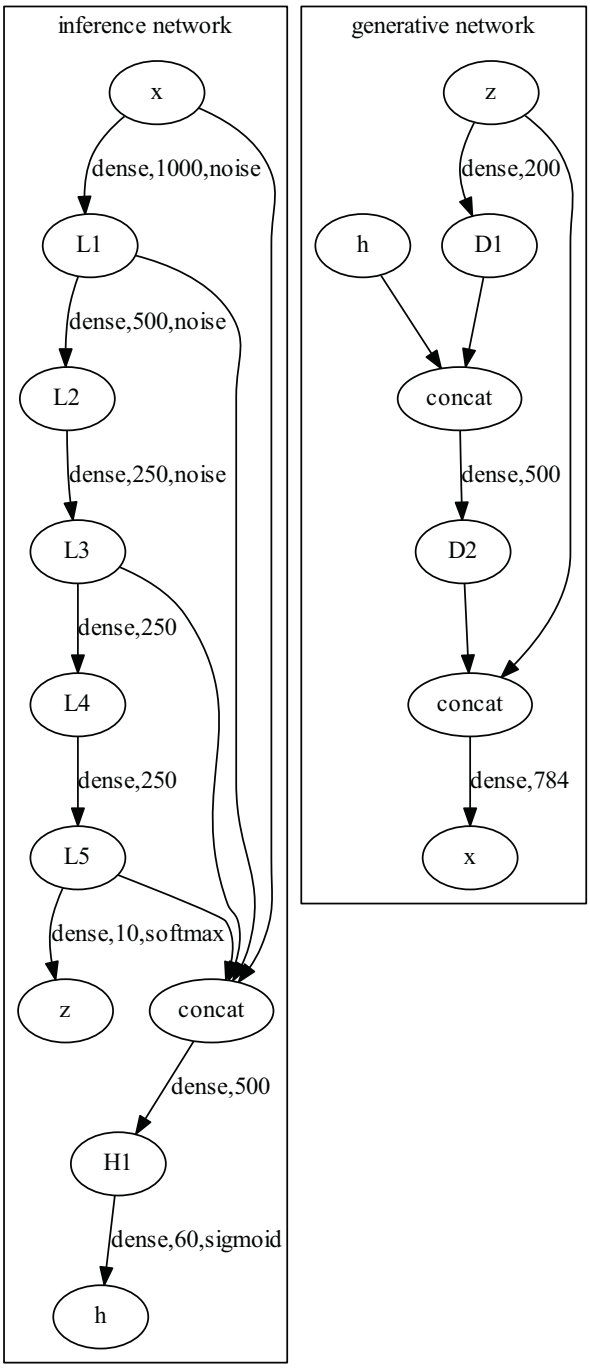

Figure 9. Architecture of the semi-JAE model on MNIST.

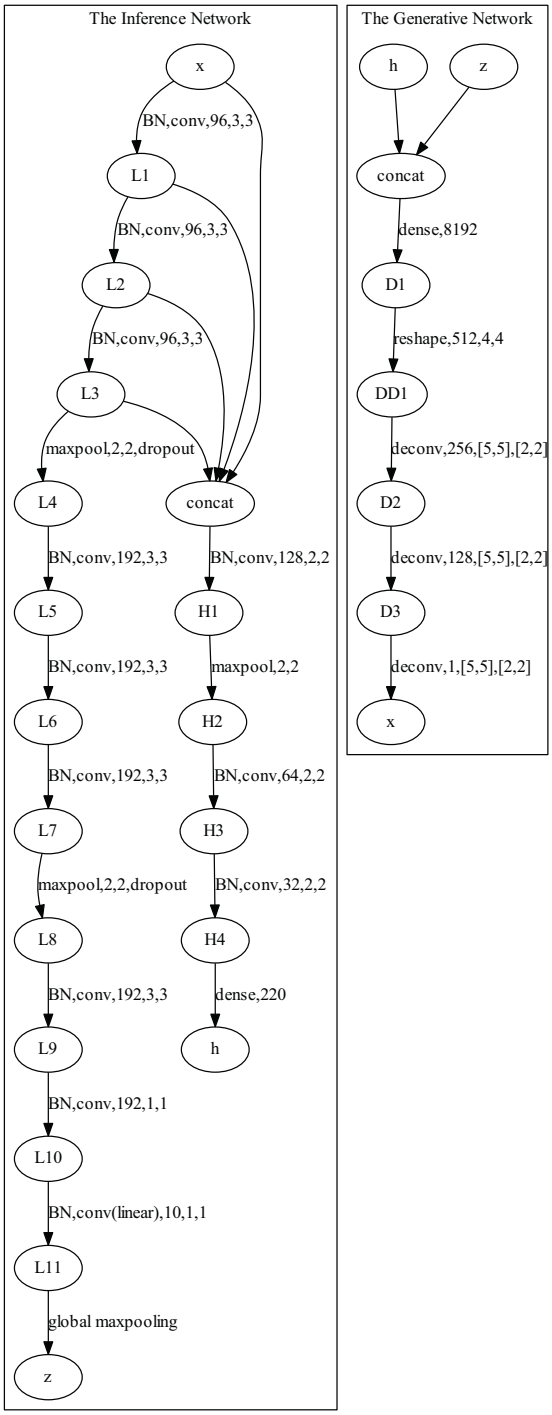

Figure 10. Architecture of the semi-JAE model on SVHN.