Unsupervised feature learning with discriminative encoder

Gaurav Pandey and Ambedkar Dukkipati
Department of Computer Science and Engineering
Indian Institute of Science
Email: {*gauravp, ambedkar*}*@iisc.ac.in*

arXiv:1709.00672v1 [cs.CV] 3 Sep 2017

***Abstract*—In recent years, deep discriminative models have achieved extraordinary performance on supervised learning tasks, significantly outperforming their generative counterparts. However, their success relies on the presence of a large amount of labeled data. How can one use the same discriminative models for learning useful features in the absence of labels? We address this question in this paper, by jointly modeling the distribution of data and latent features in a manner that explicitly assigns zero probability to unobserved data. Rather than maximizing the marginal probability of observed data, we maximize the joint probability of the data and the latent features using a two step EM-like procedure. To prevent the model from overfitting to our initial selection of latent features, we use adversarial regularization. Depending on the task, we allow the latent features to be one-hot or real-valued vectors, and define a suitable prior on the features. For instance, one-hot features correspond to class labels, and are directly used for unsupervised and semi-supervised classification task, whereas real-valued feature vectors are fed as input to simple classifiers for auxiliary supervised discrimination tasks. The proposed model, which we dub dicriminative encoder (or DisCoder), is flexible in the type of latent features that it can capture. The proposed model achieves state-of-the-art performance on several challenging tasks.**

I. Introduction

Deep neural networks have achieved extraordinary performance for several challenging supervised learning tasks in recent years. Convolutional neural networks [1], [2] have greatly improved the state-of-the-art for several problems in computer vision including classification [3], object detection [4], semantic segmentation [5] etc. Similarly, recurrent neural networks (and its variants) [6] have greatly improved the state-of-the-art on several sequential modeling tasks, such as speech-to-text generation [7], language translation [8] etc.

However, access to large amount of labeled data has been crucial for the success of neural networks for most of the above tasks. Since obtaining labeled data in such large quantities is expensive, one may wonder if it is possible to make use of unlabeled data for learning meaningful representations. This question has been addressed by several researchers using a plethora of techniques.

Among these models, generative approaches model the joint distribution of the visible and the latent features. The marginal log-probability of the observed data is then maximized. Examples of such models in deep learning include restricted Boltzmann machines [9], variational autoencoders [10], [11] and generative adversarial networks [12], [13]. Variants of these models have been successfully used for learning categorical embeddings for unsupervised and semi-supervised classification tasks [14], [15], [10].

Another set of models use self-supervision for learning real valued embeddings of data [16], [17], [18]. These embeddings are then fed to a much smaller network for solving an auxiliary classification problem. The success of these models relies on the assumption that networks trained using self-supervision will learn meaningful embeddings that generalize well for other tasks.

In this paper, we propose a general approach for learning latent representations from unlabeled data. The model, which we refer to as discriminative encoder (or DisCoder) is completely specified by the encoding and the prior distribution on the latent features. The model makes no assumptions about the distribution of the latent features. However, in our experiments we limit ourselves to categorical and normally distributed latent features only, and show that these latent features consistently achieve the state-of-the-art performance on several tasks.

The rest of the paper is organized as follows: In Section II, we motivate the choice of our model, and discuss the steps involved in training the model. We also discuss an adversarial regularization strategy that prevents the model from getting stuck to its initial configuration. In Section III, we discuss other models that are used for unsupervised representation learning. In Section IV, we present the results achieved by our model on several tasks including clustering, semi-supervised and auxiliary classification.

II. The proposed model

In this section, we will motivate our choice of the model for unsupervised feature learning. We will discuss optimization strategies for the model that utilize minibatches of data, and are nearly as fast as models used in supervised learning. The observed features will be denoted by $\mathbf{x} = (x_1, \ldots, x_d)$, whereas the corresponding latent features will be denoted by $\mathbf{z} = (z_1, \ldots, z_m)$. In particular bold font will be used for vectors, while normal font will be used to denote the corresponding components. The distribution $p_\theta(\mathbf{z}|\mathbf{x})$ will be referred to as the encoding distribution, whereas $p_\theta(\mathbf{x}|\mathbf{z})$ will be referred to as the decoding distribution. The parameters of the networks will be denote by θ. The output of the encoding network will be denoted as ϕ. We will denote that the number

of samples in the training data by n, and the size of the latent features by K.

A. Motivation

A common approach to address the problem of unsupervised learning with latent features is by defining a joint distribution over the observed and the latent features. Such a model will be referred to as generative model in the sequel. One can then maximize the marginal distribution of the observed features.

$$p_\theta(\mathbf{x}) = \sum_{\mathbf{z}} p_\theta(\mathbf{x}, \mathbf{z}) \tag{1}$$

The MAP estimate of the latent features given the observed features can then be used as a representation for the data.

$$\hat{\mathbf{z}} = \arg\max_{\mathbf{z}} \log p_\theta(\mathbf{x}, \mathbf{z}) \tag{2}$$

The above approach is one of the most common approaches for modeling latent features, and has been successfully used in Gaussian mixture models (GMM), hidden Markov models (HMM), restricted Boltzmann machines (RBM), variational autoencoder (VAE), latent Dirichlet allocation (LDA) etc. Despite its almost ubiquitous success, this approach for learning latent features suffers from a fatal flaw: It is theoretically possible to maximize the $p_\theta(\mathbf{x})$ without changing the encoding distribution $p_\theta(\mathbf{z}|\mathbf{x})$ at all. This is particularly true when the choice of decoding distribution $p_\theta(\mathbf{x}|\mathbf{z})$ is flexible enough to fit the data distribution. In such a scenario, the model can choose to ignore the latent features $\mathbf{z}$ in the decoding distribution $p_\theta(\mathbf{x}|\mathbf{z})$ by equating $p_\theta(\mathbf{x}|\mathbf{z})$ to $p_\theta(\mathbf{x})$. When this happens,

$$p_\theta(\mathbf{z}|\mathbf{x}) = \frac{p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z})}{p_\theta(\mathbf{x})} = \frac{p_\theta(\mathbf{x})p_\theta(\mathbf{z})}{p_\theta(\mathbf{x})} = p_\theta(\mathbf{z})\,, \tag{3}$$

that is, the latent features $\mathbf{z}$ are completely independent of the observed features $\mathbf{x}$.

In general, the choice of decoding model is simple enough to prevent this from happening. For instance, in HMM, RBM, VAE and LDA, the decoding distribution factorizes completely, whereas in GMM, the decoding distribution is a Gaussian distribution. This forces the model to utilize the latent features to effectively model the data distribution.

Hence, generative models rely on weak decoding distributions for learning useful representations. In general, a good generative model doesn't guarantee a useful latent representation. The problem lies in the fact, that generative models spend most of their time optimizing $p_\theta(\mathbf{x})$, and may or may not choose to care about the encoding distribution $p_\theta(\mathbf{z}|\mathbf{x})$. To address this specific issue, we define a model that specifically assigns zero probability to any point $\mathbf{x}$ that doesn't occur in the training data. One can see that such a modeling strategy is intinsically used by discriminative models for supervised learning tasks. In particular, if $(\mathbf{x}^{(1)}, \mathbf{z}^{(1)}), \ldots, (\mathbf{x}^{(n)}, \mathbf{z}^{(n)})$ are the observed points and their labels, the joint log-likelihood $\mathcal{L}$ can be written as

$$\mathcal{L} = \sum_{i=1}^{n} \log p_\theta(\mathbf{x}^{(i)}) + \sum_{i=1}^{n} \log p_\theta(\mathbf{z}^{(i)}|\mathbf{x}^{(i)}) \tag{4}$$

In discriminative models, the first term in the above equation is completely independent of the parameters in the second term, and hence can be maximized independently. The maximum occurs, when $p_\theta(\mathbf{x}) = \frac{1}{n}$ for all the observed points, and 0 for any point that doesn't occur in the training data.

Unfortunately, when the labels $\mathbf{z}$ are unknown, the same strategy can't be used, since the model collapses to a single $\mathbf{z}$ with $p_\theta(\mathbf{z}|\mathbf{x}) = 1$ for all $\mathbf{x}$. To prevent this from happening, we couple the encoding distribution $p_\theta(\mathbf{z}|\mathbf{x})$ with a function $f(\mathbf{z})$, and define a new distribution

$$q_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{z}|\mathbf{x}) f(\mathbf{z}) \tag{5}$$

To obtain the value of f at $\mathbf{z}$, we force the distribution to satisfy the following constraints:

1) The marginal distribution of $q_\theta(\mathbf{x}, \mathbf{z})$ over $\mathbf{z}$ is $p_\theta(\mathbf{z})$. This is done to prevent the model from collapsing to a single $\mathbf{z}$. This also allows us to incorporate the prior information about the latent features into our model. For instance, if we know that the components of $\mathbf{z}$ should form a Markov chain, we can incorporate that information easily into the model.
2) The distribution q_θ assigns 0 probability to any point that doesn't occur in the training data. This is done to prevent the model from spending its time and effort in training $q_\theta(\mathbf{x})$. In the words of Vapnik, "One should solve the problem directly and never solve a more general problem as an intermediate step".

Lemma 1: Let q_θ be a distribution of the form given in (5) that satisfies the constraints above. Then

$$q_\theta(\mathbf{x}, \mathbf{z}) = \frac{p_\theta(\mathbf{z}|\mathbf{x})p_\theta(\mathbf{z})}{\sum_{j=1}^{n} p_\theta(\mathbf{z}|\mathbf{x}^{(j)})}\,, \tag{6}$$

for any $\mathbf{x}$ observed in the training data and, 0 otherwise.
This is the distribution that we will optimize for unsupervised feature learning in the rest of the paper. *Note that the model is completely specified by the choice of the prior $p_\theta(\mathbf{z})$ and the encoding distribution $p_\theta(\mathbf{z}|\mathbf{x})$.*

B. Discriminative Encoder (DisCoder)

In the previous section, we motivated the choice of the model used for unsupervised feature learning in this paper. The model will be referred to as discriminative encoder (or DisCoder) in the sequel. In this section, we will discuss the model in further detail, and will see, why the name *discriminative encoder* is apt for this model.

Given an $i.i.d$ sequence of unlabeled samples $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}$, the joint likelihood function can be written as a function of the corresponding latent features $\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(n)}$, and parameters of the model θ.

$$\mathcal{L}(\theta, \mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(n)}) = \sum_{i=1}^{n} \log q_\theta(\mathbf{x}^{(i)}, \mathbf{z}^{(i)})\,, \tag{7}$$

where $q_\theta(\mathbf{x}, \mathbf{z})$ is as defined in (5). In order to optimize the above objective, we use an EM-like procedure, that alternates between the selection of $\mathbf{z}^{(i)}$ and optimization with respect to θ.

In the selection step, for each $\mathbf{x}^{(i)}$ in the training data, we find the best $\mathbf{z}$, that is, the $\mathbf{z}$ that maximizes $\log q_\theta(\mathbf{x}^{(i)}, \mathbf{z})$. The quantity $\log q_\theta(\mathbf{x}^{(i)}, \mathbf{z})$ can be expanded as

$$\log q_\theta(\mathbf{x}^{(i)}, \mathbf{z}) = \log p_\theta(\mathbf{z}|\mathbf{x}^{(i)}) + \log p_\theta(\mathbf{z}) - \log \sum_{j=1}^{n} p_\theta(\mathbf{z}|\mathbf{x}^{(j)}) \tag{8}$$

Hence, the optimal latent representation should be such it maximizes $\log p_\theta(\mathbf{z}|\mathbf{x}^{(i)})$, but minimize $\log \sum_{j=1}^{n} p_\theta(\mathbf{z}|\mathbf{x}^{(j)})$. In other words, the latent representation should have high probability of being assigned to $\mathbf{x}^{(i)}$, but low probability of being assigned to any other $\mathbf{x}^{(j)}, j \neq i$.

Once we have found the optimal latent representation for $\mathbf{x}^{(i)}$, we equate it to $\mathbf{z}^{(i)}$. Next, we optimize the objective with respect to the parameters of the model. In expanded form, the objective as a function of θ can be written as

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} \log p_\theta(\mathbf{z}^{(i)}|\mathbf{x}^{(i)}) - \sum_{i=1}^{n} \log \sum_{j=1}^{n} p_\theta(\mathbf{z}^{(i)}|\mathbf{x}^{(j)}) \tag{9}$$

For instance, when $p_\theta(\mathbf{z}|\mathbf{x})$ is normally distributed with mean $\phi(\mathbf{x})$ and variance $1/2$, the objective can be written as

$$\begin{aligned} -\mathcal{L}(\theta) = & \sum_{i=1}^{n} ||\phi(\mathbf{x}^{(i)}) - \mathbf{z}^{(i)}||^2 \\ & + \sum_{i=1}^{n} \log \sum_{j=1}^{n} \exp(-||\phi(\mathbf{x}^{(j)}) - \mathbf{z}^{(i)}||^2) \end{aligned} \tag{10}$$

Since $\mathbf{z}^{(i)}$ are fixed for this step, the terms that depend on $\mathbf{z}^{(i)}$ alone, have been removed from the objective. For a fixed $\mathbf{x}^{(i)}$, this step trains the network to maximize the probability of the $\mathbf{z}^{(i)}$ selected in the previous step while simultaneously lowering the probability of other $\mathbf{z}^{(j)}, j \neq i$.

Both the steps of training try to ensure that the learnt representations are as dissimilar to each other as possible, while simultaneously satisfying the requirement of prior distribution. Hence, the representations learn to capture those features that vary among the samples, while completely ignoring the features common to all the samples. Hence, we name the model as discriminative encoder (or DisCoder).

Semi-supervised learning: If we have access to a set of labelled samples, DisCoder can utilize those samples to improve upon the learnt embeddings. We achieve this by adding a term to the objective to maximize the conditional log-likelihood over the labelled samples. If we denote the set of labelled samples as $\{(\mathbf{x}_s^{(1)}, \mathbf{z}_s^{(1)}), \ldots, (\mathbf{x}^{(m)}, \mathbf{z}^{(m)})\}$, the new objective can be written as:

$$\mathcal{L}(\theta, \mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(n)}) \tag{11}$$

$$= \sum_{i=1}^{n} \log q_\theta(\mathbf{x}^{(i)}, \mathbf{z}^{(i)}) + \sum_{i=1}^{m} \log p_\theta(\mathbf{z}_s^{(i)}|\mathbf{x}_s^{(i)}), \tag{12}$$

C. Optimization

As mentioned in the previous section, the optimization proceeds in the following two steps:

1) Latent feature selection
2) Encoding network optimization

The latent feature selection step differs depending on the choice of the latent features, while the encoding network optimization step remains essentially the same irrespective of the latent features. We will first discuss the latent feature selection step for 2 special cases and then discuss the network optimization step.

Latent feature selection: While the proposed model is general enough to handle any encoding distribution, in this paper, we restrict ourselves to categorical and Gaussian distribution. Categorical distribution is used, when the latent features are one-hot vectors. For a fixed $\mathbf{x}^{(i)}$, the mean of the encoding distribution $\phi(\mathbf{x}^{(i)})$ is the output of the network. The latent feature selection step for categorically distributed latent features can be obtained by exponentiating equation (8)

$$\mathbf{z}^{(i)} = \arg\max_{\mathbf{z}} \frac{\prod_{k=1}^{K} \phi_k(\mathbf{x}^{(i)})^{z_k}}{\sum_{j=1}^{n} \prod_{k=1}^{K} \phi_k(\mathbf{x}^{(j)})^{z_k}} \tag{13}$$

In particular, $z_l^{(i)} = 1$, if

$$l = \arg\max_{k \in \{1,\ldots,K\}} \frac{\phi_k(\mathbf{x}^{(i)})}{\sum_{j=1}^{n} \phi_k(\mathbf{x}^{(j)})} \tag{14}$$

The last equation follows from the fact that only one component of $\mathbf{z}$ can be non-zero at a time. It can be computed efficiently and independently for each component.

For real valued latent features $\mathbf{z}$, we assume that the prior over $\mathbf{z}$ follows a uniform distribution over a sphere, while the encoding distributed is normally distributed with a constant variance λ, which is selected by cross-validation on a validation set for a secondary task. For a fixed $\mathbf{x}^{(i)}$, the mean of the encoding distribution $\phi(\mathbf{x}^{(i)})$ is the output of the network. The latent feature selection step for real valued vectors can be written as:

$$\begin{aligned} \mathbf{z}^{(i)} = & \arg\min_{\mathbf{z}} \frac{||\phi(\mathbf{x}^{(i)}) - \mathbf{z}||^2}{2\lambda} \\ & + \log \sum_{j=1}^{n} \exp\left(-\frac{||\phi(\mathbf{x}^{(j)}) - \mathbf{z}||^2}{2\lambda}\right) \end{aligned} \tag{15}$$

The above objective is a differentiable function of $\mathbf{z}$. Hence, we use stochastic gradient descent (SGD) for optimization. Moreover, we optimize the restriction of the third term in the objective over a small batch of 1000 samples only. In particular, we keep a queue of 1000 samples on which the model was trained recently, and use the samples in this queue for minimizing the objective.

Encoding network optimization: For fixed $\mathbf{x}$ and $\mathbf{z}$, the encoding distribution $p_\theta(\mathbf{z}|\mathbf{x})$ is a differentiable function of θ. Hence, we use minibatch SGD to maximize the objective $\mathcal{L}(\theta)$ defined in (9). Given a minibatch of samples, we optimize the restriction of the objective to this minibatch, while ignoring the examples from other minibatches. In particular, the summation inside the log of second term in (9) is evaluated over

the minibatch only. This greatly reduces the computational complexity of optimization from $\mathcal{O}(n^2)$ to $\mathcal{O}(nb)$, where b is the batchSize. Furthermore, the randomness introduced by the restriction serves to regularize the encoding network, and encourages it to explore other choices of $\mathbf{z}$.

D. Regularizing the model

As mentioned in the previous section, the training of the model alternates between latent feature selection and encoding network optimization. The latent features selected at the beginning of training are completely random. Now, if the encoding network is sufficiently powerful, and the encoding network optimization step proceeds for a sufficiently long time, the encoding network may attempt to fit to the initial features, despite the fact that they are completely random.

This is a problem faced by almost all the approaches for unsupervised representation learning. For instance, in clustering based approaches [19], [20], [14], if the encoding network that maps the samples to clusters is sufficiently powerful, and the training of encoding network proceeds for a sufficiently long time, keeping the clusters fixed, any choice of the clusters can be predicted with absolute certainty. Even in the case of autoencoders, if the encoding and the decoding networks are sufficiently powerful, the autoencoder will be able to map any image to any representation, while the decoding network will be able to map the same representation back to the image. (In fact, in our experiments on 2D-representations of MNIST digits, we found that if the encoding and decoding networks are deep enough, the representations learnt by an autoencoder don't change much during the course of training.)

In general, a number of reasons prevent an unsupervised learning algorithm from getting stuck in the initial choice of representations:

1) The encoding network is often optimized for a single stochastic gradient descent update, and is immediately followed by a secondary step, such as clustering (or latent feature selection in our case).
2) The encoding network is regularized heavily to force it to learn simpler mappings.
3) Stochastic gradient descent training in neural networks prefers simple mappings that result in flat minima [21].

Badly selected latent features can be escaped easily, if the variance of the encoding distribution is high. This can be observed from equation (15) for normally distributed latent features. When λ is high, a slight change in the latent representation $\mathbf{z}$, doesn't change the objective by much. Hence, the stochasticity introduced by the optimization technique is often enough for the model to explore other choices for $\mathbf{z}$. However, as λ gets smaller, the expression gets more and more sharply trenched around $\phi(\mathbf{x}^{(i)})$. This makes it almost impossible for the model to assign any value other than $\phi(\mathbf{x}^{(i)})$ to $\mathbf{z}$, despite the stochasticity of training.

Hence, primarily for the above reason, we fix the variance of the encoding distribution to a constant, which is selected via cross-validation on a validation set for a secondary task, when the latent features are normally distributed. This approach works, since the latent space is continuous, and several choice of latent features have high probability under the encoding distribution. However, if we try to achieve the same for categorically distributed latent features, we end up with distributions that are almost uniform over all the labels. The DisCoder ends up learning nothing meaningful. The same is true for any choice of discrete distribution, since the latent features are far apart. To prevent this from happening, we use adversarial regularization for categorical latent features, which is discussed next.

Adversarial regularization: The adversarial regularization works as follows: While training the DisCoder to maximize $\sum_{i=1}^{n} \log q_\theta(\mathbf{x}^{(i)}, \mathbf{z}^{(i)})$ for real samples $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}$, we simultaneously train it to be totally confused about a set of *fake* samples $\bar{\mathbf{x}}^{(1)}, \ldots, \bar{\mathbf{x}}^{(n)}$. We force this by maximizing the probability over all the latent features for fake images. Hence, the new objective becomes:

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} \log q_\theta(\mathbf{x}^{(i)}, \mathbf{z}^{(i)}) + \sum_{i=1}^{n} \sum_{\mathbf{z}} p(\mathbf{z}) \log p_\theta(\mathbf{z}|\bar{\mathbf{x}}^{(i)}) \tag{16}$$

The above regularization forces the model to learn mappings from data to categories that are completely unsure about samples that don't occur in the training data. A similar scheme for regularization was also used in [14].

Note: The adversarial regularization used in this paper must not be confused with adversarial training used in [22]. We choose the term adversarial regularization for lack of a better name.

In order to complete the description of the model, we also need a way to generate fake samples. We use a generator to generate fake images, while we use negative sampling to generate fake documents.

Class-conditioned generation: In order to generate fake image samples, one can couple the encoder with a generator G, that takes Gaussian noise $\mathbf{n}$ and latent features $\mathbf{z}$ as input and generates fake samples $\bar{\mathbf{x}}$ as output, that is,

$$\bar{\mathbf{x}} = G(\mathbf{n}, \mathbf{z})$$

The generator is trained to generate samples $\bar{\mathbf{x}}$, such that the encoder can extract $\mathbf{z}$ back from $\bar{\mathbf{x}}$. We achieve this by training the generator to maximize $\log p_\theta(\mathbf{z}|\bar{\mathbf{x}} = G(\mathbf{n}, \mathbf{z}))$. Note that the discriminator is simultaneously being trained to be completely confused about the latent features of the fake examples. This creates an adversarial game akin to the adversarial game of generative adversarial networks [12]. The choice of the generative model allows us to visualize the latent features, and hence, determine if the model is learning anything meaningful. In particular, if we wish to determine the information captured by the i^{th} latent feature, we set it to 1 and the other latent features to 0, couple it with Gaussian noise and pass it through the generator.

Feature matching: Class-conditioned generation results in images that are sharp with the object immediately identifiable. Note that the encoding network is simultaneously being trained to be totally confused about the generated samples. However,

if the underlying class of the fake image is immediately identifiable, training a network to be confused about the image results in unstable training. Hence, while class conditioned generation is useful when the underlying task is to generate sharp images, it is not really suitable for classification tasks.

In order to address this problem, we use feature matching for classification tasks. Noise is passed through the generator to generate a batch of fake images $\bar{\mathbf{x}}$, that is, $\bar{\mathbf{x}} = G(\mathbf{n})$. The statistics of the fake batch is then compared with the statistics of the real batch. As suggested in [15], we pass the fake batch as well as the real batch through the encoding network, and compute the squared difference between the means of encodings of the real and fake batch for the penultimate layer. The generator is trained to minimize the distance between the average encodings for the penultimate layer. The reultant images aren't very sharp. However, this approach works quite well for classification task as has been observed in [15].

Negative Sampling: To generate fake documents, we randomly select words from the vocabulary, based on the frequency with which they occur in the corpus. The size of the document is Poisson distributed whose mean is the mean of the length of the documents that occur in the corpus

Algorithm 1 combines all the above steps for training the model.

E. Complexity of training

As mentioned in the previous sections, the complexity of each step (latent feature selection as well as network optimization) is quadratic in the size of the batch, that is, $\mathcal{O}(b^2)$. However, when the batch size is small enough (< 2000 samples), the empirical running time depends linearly on the batch size rather than quadratically. This is because, most of the running time is spent in forward and backpropagating through the network, which needs to be done only once for the entire batch). Very little time is utilized in the computation of the distribution of the Discoder or its gradient from the output of the network. Empirically, we observed that DisCoder takes approximately the same time as Improved GAN [15] for semi-supervised learning tasks.

III. Related works

Unsupervised learning is a well studied problem in machine learning, and a plethora of techniques exist for addressing this problem, including probabilistic and non-probabilistic approaches. Most of the probabilistic models for unsupervised learning are generative models, that is, they explicitly model the probability of the observed data. In this paper, we will restrict ourselves to discuss only those models that are employed in deep learning.

A. Generative models

Representation learning in generative models proceeds by defining a joint distribution over the visible features and the latent features. One can then optimize the marginal log-likelihood (or its approximation) over the observed data. Examples of such models include Boltzmann machines

---

Algorithm 1: Minibatch training of DisCoder

Input: Observed features $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}$

Output: An encoder $p_\theta(\mathbf{z}|\mathbf{x})$ and generator G (if adversarial regularization is enabled)

Repeat the following steps until convergence

1) Randomly select a batch of observed features B.
2) Forward propagate them through the encoder to obtain the encoding distribution $p_\theta(\mathbf{z}|\mathbf{x}), \mathbf{x} \in B$.
3) Select the latent features $\mathbf{z_x}$ for all $\mathbf{x} \in B$ by maximizing $\log q_\theta(\mathbf{x}, \mathbf{z})$ for each $\mathbf{x}$ in the batch.
4) Train the encoding network to maximize
$$\mathcal{L}(\theta) = \sum_{\mathbf{x} \in B} \log q_\theta(\mathbf{x}, \mathbf{z_x})$$
5) If semi-supervised learning is enabled:
   a) Randomly select a batch B_s of labeled samples.
   b) Train the encoding network to maximize $\sum_{(\mathbf{x},\mathbf{z}) \in B_s} \log p_\theta(\mathbf{z}|\mathbf{x})$.
6) If class conditioned generation is enabled:
   a) Concatenate noise $\mathbf{n}$ with the latent features obtained in the previous step $\mathbf{z_x}$ and propagate them through the generator to get a batch of fake samples $\bar{B}$.
   b) Train the encoding network to maximize
   $$\sum_{\mathbf{x} \in \bar{B}} \sum_{\mathbf{z}} p(\mathbf{z}) \log p_\theta(\mathbf{z}|\bar{\mathbf{x}}^{(i)})$$
   c) Train the generator to maximize $\log p_\theta(\mathbf{z}|\bar{\mathbf{x}} = G(\mathbf{n}, \mathbf{z}))$
7) If feature matching is enabled:
   a) Forward propagate noise $\mathbf{n}$ through the generator to generate a batch of fake samples.
   b) Train the encoding network to maximize
   $$\sum_{\mathbf{x} \in \bar{B}} \sum_{\mathbf{z}} p(\mathbf{z}) \log p_\theta(\mathbf{z}|\bar{\mathbf{x}}^{(i)})$$
   c) Let the mean of the penultimate layer of the encoder for the fake batch be $\phi_{\bar{B}}$ and for the real batch be ϕ_B.
   d) Train the generator to minimize the squared distance between the average encodings for the real and fake batch for the penultimate layer.
8) If negative sampling is enabled:
   a) Create a batch of fake documents $\bar{B}$ by sampling words in the vocabulary based on their frequency with which they occur in the corpus.
   b) Train the encoding network to maximize
   $$\sum_{\mathbf{x} \in \bar{B}} \sum_{\mathbf{z}} p(\mathbf{z}) \log p_\theta(\mathbf{z}|\bar{\mathbf{x}}^{(i)})$$

(RBM [9], DBM [23]), variational autoencoders (VAE) [10], [11] and generative adversarial networks (GAN) [12].

RBM and DBM have proved to be excellent for unsupervised learning on simple datasets such as MNIST. However, training as well as inference in deep extensions of these models is quite challenging, especially when convolutional layers are employed in these models. This severely limits the application of these models for unsupervised feature learning.

In contrast, variational autoenoders [10], [11] and its deterministic variant [24] utilize deep neural networks for modeling, and are comparatively simpler to train. These models learn embeddings so as to minimize the reconstruction error. Variational autoencoders have been successfully used for a wide variety of tasks in deep learning, that include image captioning [25], semi-supervised learning [26] etc.

Among the generative models, the model most relevant to our work is GAN [12]. In a GAN, a discriminator is trained to distinguish between fake and real samples, while a generator is simultaneously trained to generate images from a latent distribution that fool the discriminator. Recent works involving GANs [13], [27] have explored methods for inferring the latent representations from the samples, and successfully utilized these latent representations for several auxiliary tasks.

Variants of GANs have also been used for unsupervised and semi-supervised learning tasks. In particular, CatGANs [14] learn a categorical latent representation for the data by maximizing the mutual information between the data and the categorical labels, while simultaneously maximizing the entropy over the class labels for fake images. Adversarial autoencoders [28] learn a mapping from the data to the latent space so as to minimize reconstruction error. However, unlike variational autoencoders, they use a GAN framework to enforce the prior on the latent features. This allows the model to learn hybrid (discrete + continuous) embeddings, which isn't possible for variational autoencoders.

B. Self-supervised learning:

Self-supervised learning proceeds by generating labels from the data using information present in the structure of the data. There has been a growing interest in such methods, since no extra effort is need for labeling these samples. For instance, Dosovitskiy *et al.* [29] train a model for training a convolutional network that assigns each image to its own class. In particular, each image is used as a seed to generate a class of images by applying various transformations. Hence, the output layer of the CNN grows linearly with the number of images in the dataset, severely limiting the scalability of the model.

Another set of methods force the model to learn the relative position of various patches in the image with respect to each other. For instance, jigsaw networks [16] permute the patches of the image, and train the network to predict the correct permutation. Similarly, context prediction networks [17] are trained to predict the correct relationship relationship between two patches. To successfully complete this task, the models are forced to learn representations that capture the global structure of the image.

Models that rely on videos as training data, attempt to learn features by exploiting the motion information present in videos. In particular, the approach in [17], identifies two patches that correspond to the same trajectory in a video, and minimizes the distance between their representations. This forces the model to learn feature that are invariant to orientation of the object in the video.

IV. Experiments

In order to evaluate the capability of the model for learning meaningful representations, we evaluate the learnt representations across several tasks. For each task, we evaluate our model against the state-of-the-art models for that task. The code for the experiments in this paper is available at *https://github.com/gauravpandeyamu/DisCoder*.

A. Unsupervised and semi-supervised classification

We evaluate the one-hot embeddings learnt by the model for the task of clustering on MNIST, and 20-newsgroup dataset. We also evaluate the embeddings for the task of semi-supervised classification on CIFAR-10 dataset. In all our experiments, we use a batch size of 100 images, and Adam optimizer with a constant learning rate of 0.0002. Since neural networks are likely to get stuck in local optima, we repeat the experiment 10 times, and report the mean and the standard deviation.

MNIST: The MNIST [30] is a relatively simple dataset of digits from 0 to 9. We normalize the images, by dividing the pixel intensities by 255. We evaluate DisCoder on this dataset for the clustering task into 20 clusters. We use generative adversarial regularization for regularizing the DisCoder as discussed in Section II-D. The architecture of the generator and encoding network are provided in Table I. In order to evaluate the accuracy of the clustering algorithm, we need to associate each cluster with a label. Towards that end, we compute the intersection of the cluster with all the classes. The label of the class with the maximum intersection, is assigned to the cluster.

The clustering error achieved by the various models on MNIST dataset is shown in Table II. The accuracy is computed as follows: For each cluster i, we identify the sample $\mathbf{x}$ that maximizes $p(\mathbf{z} = i|\mathbf{x})$. Next, the label of the selected $\mathbf{x}$ is assigned to all the samples in this cluster. The test error is computed based on the labels assigned to the points using this procedure. Such a scheme is also used for evaluating clustering in [28].

For completeness, we also show the results, when adversarial regularization isn't used. As can be observed, adversarial regularization is extremely crucial for achieving good performance, when one-hot embeddings are used in DisCoder. Also noteworthy, is the high standard deviation of an unregularized DisCoder, which indicates that the model is overfitting to the initial assignment.

TABLE I
Network architecture for MNIST

Generator	Encoder
256 ConvT 4x4, relu	64 Conv 4x4, stride 2, leaky relu
128 ConvT 3x3, stride 2, bn, relu	128 Conv 4x4, stride 2, bn, lrelu
—	128 Conv 3x3, stride 1, bn, lrelu
64 ConvT 4x4, stride 2, bn, relu	256 Conv 3x3, stride 2, bn, lrelu
—	256 Conv 3x3, stride 1, bn, lrelu
1 ConvT 4x4, stride 2, sigmoid	nc Conv 4x4, stride 1, softmax

*ConvT=transposed convolution
*relu = rectified linear units
*lrelu=leaky rectified linear units
*bn=batch normalization
*nc=number of clusters

TABLE II
Clustering error on MNIST dataset for various models

Model	Percentage error
DEC (10 clusters) [20]	15.6
CatGAN (20 clusters) [14]	4.27
Adversarial autoencoder (32 clusters) [28]	4.10±1.13
Discoder (unregularized)	24.3±10.6
DisCoder (20 clusters)	**3.12±0.93**

The DisCoder achieves the state-of-the-art results for clustering on MNIST, significantly outperforming other methods. Since the generator is trained along with the encoder, we can visualize the images associated with each latent feature by setting the corresponding component of $\mathbf{z}$ to 1, concatenating it with noise and passing it through the generator. The resultant images are shown in Figure 1. As expected, each latent feature has learned to associate itself with images of a single class.

20 newsgroup: In order to ensure reproducibility, we use a preprocessed version of the dataset[1]. We use tf-idf representation for the documents. We train a DisCoder with negative sampling for clustering on this dataset as detailed in Section II-D. The encoding network consists of a single hidden layer with 1000 units. In order to compute clustering accuracy, we use the same strategy that we had used for MNIST.

The clustering accuracy achieved by the various models on 20 newsgroup dataset is shown in Table III. For LSA and LDA, we set the number of topics to the number of clusters. We assign each document to the topic with the highest probability for the given document. Again, as in the case of MNIST, a regularized DisCoder outperforms all the other models.

CIFAR-10: Next, we evaluate the performance of the DisCoder on semi-supervised learning task for CIFAR-10 dataset. CIFAR-10 is a relatively complicated image dataset with 50000 training samples belonging to 10 classes. There is high variability within each class, making it almost impossible to perform clustering on this dataset. Hence, we use the dataset for semi-supervised classification. We normalize the images to lie between -1 and 1. We use 400 examples per class as labelled data, and use the rest of the images as unlabelled.

[1]The pre-processed dataset is available at https://sites.google.com/site/renatocorrea02/textcategorizationdatasets

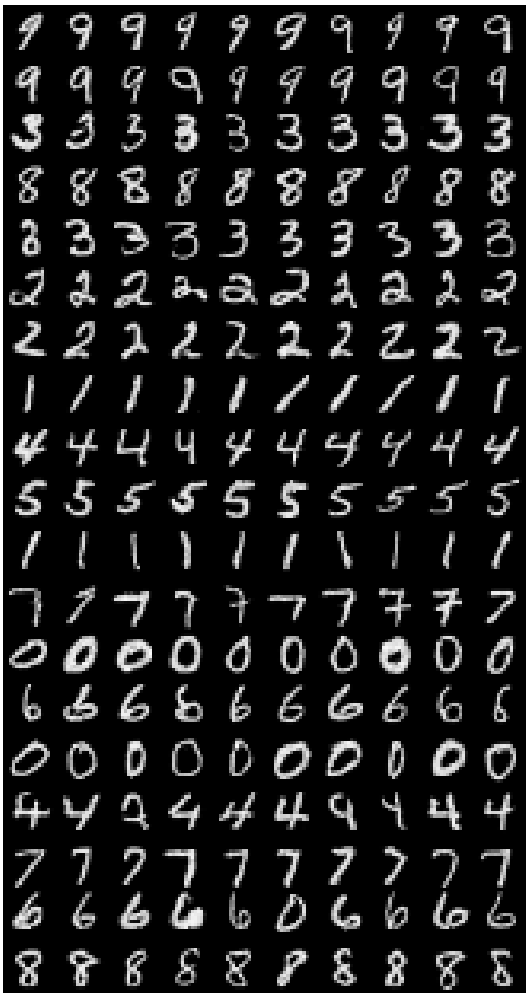

Fig. 1. Generated samples from the unsupervised MNIST generator conditioned on $\mathbf{z}$. Each row corresponds to a different component of $\mathbf{z}$ set to 1. Note that despite the fact that training was done in an unsupervised fashion, the generator was successfully able to associate different clusters with different classes.

TABLE III
Clustering accuracy on 20 newsgroup dataset for various models

Model	Percentage accuracy on 20 newsgroup
K-means	40.4
LSA [31]	57.25
LDA [32]	55.6
Discoder (unregularized)	35.11±8.13
DisCoder	**61.43±2.24**

The architecture of the generator and the encoding network used for this task is given in Table IV. The architecture of the encoding network is essentially the same as the architecture of discriminator in Improved GAN [15]. In particular, the magnitude of weights in all the convolution layers of the encoding network are held fixed during training, and leaky rectified nonlinearities with a slope of .2 follow every convolution layer.

TABLE IV
Network architecture for CIFAR10

Generator	Encoder
512 ConvT 4x4, relu	dropout(.2)
——	96 Conv 3x3, stride 1, wn, lrelu
——	96 Conv 3x3, stride 1, wn, lrelu
——	96 Conv 3x3, stride 2, wn, lrelu
256 ConvT 4x4, stride 2, bn, relu	dropout(.5)
——	192 Conv 3x3, stride 1, wn, lrelu
——	192 Conv 3x3, stride 1, wn, lrelu
——	192 Conv 3x3, stride 2, wn, lrelu
128 ConvT 4x4, stride 2, bn, relu	dropout(.5)
——	192 Conv 3x3, stride 1, wn, lrelu
——	192 Conv 1x1, stride 1, wn, lrelu
——	192 Conv 1x1, stride 1, wn, lrelu
——	Global average pooling
3 ConvT 4x4, stride 2, tanh	10 linear,wn, softmax

*wn= weight normalization

TABLE V
Performance of various models on CIFAR-10 dataset for the task of semi-supervised classification.

Model	Percentage error using 4000 labeled samples
Ladder network [33]	20.4±0.47
CatGAN [14]	19.58±0.46
Improved GAN [15]	18.63±2.32
Discoder (unregularized)	31.33±4.4
DisCoder	**17.24±0.43**

Batch normalization is used in the layers of the generator.

We use feature matching to generate the fake images for our experiments on semi-supervised learning. The classification error achieved by the various models on CIFAR-10 is shown in Table V. DisCoder achieves the lowest classification error among all the models, significantly outperforming both Imrpoved GAN and CatGAN.

We use class-conditioned generation to visualize samples associated with each class learnt by the generator. To generate each row, we set the corresponding component of $\mathbf{z}$ to 1, concatenate it with noise, and pass it through the generator. The corresponding images for the 10 classes are shown in Figure 2. One can observe that for many classes, the generator is able to capture the shape of the objects quite effectively, which has been known to be quite challenging for CIFAR-10. For comparison, we have shown the images generated by feature matching when trained on CIFAR-10 for the same task. As one can observe, most of the images are meaningless blobs.

SVHN: We also evaluate the performance of the DisCoder for semi-supervised learning on the Street View House Numbers (SVHN) dataset [34]. The dataset consists of cropped images of digits extracted from house numbers in Google Street View images. As is the case with CIFAR-10, the dataset has high variability, and hence, requires a few labelled examples per class to achieve satisfactory classification result. We use 100 examples per class as labelled images, while the rest of the images are used as unlabelled images.

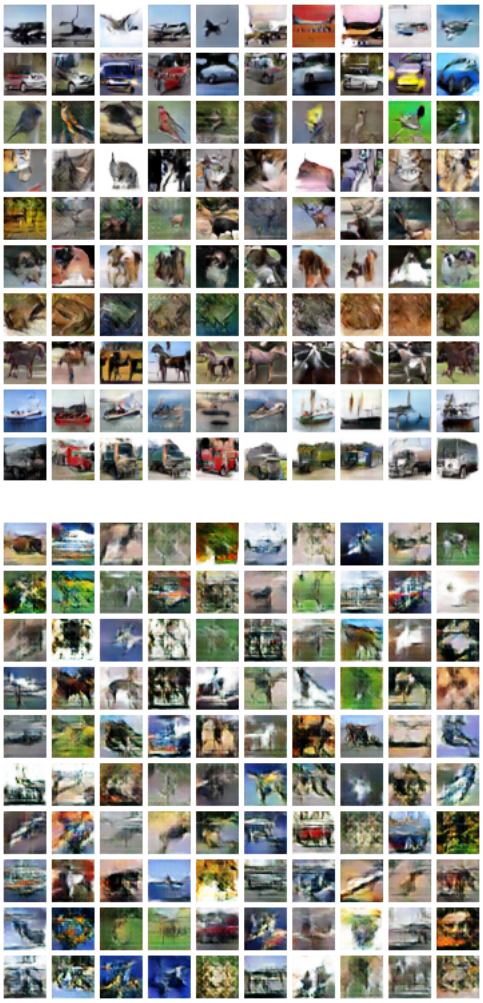

Fig. 2. (Top) Generated samples from the semi-supervised CIFAR-10 generator using class-conditioned generation. Each row corresponds to a different component of $\mathbf{z}$ set to 1. Since this problem is semi-supervised, $\mathbf{z}$ isn't exactly a latent feature. Note that for many classes, the generator is able to preserve the shape of the objects, which is known to be quite challenging for CIFAR-10 images. (Bottom) Samples generated by the generator using feature matching for the same task. Note that the shape of the objects isn't preserved in the images.

The architecture of the generator and the encoding network used for this task is given in Table VI. Again, the architecture of the encoding network is essentially the same as the architecture of discriminator in Improved GAN [15]. Feature matching is used to generate the fake images for our experiments on semi-supervised learning. The classification error achieved by the various models on SVHN is shown in Table VII. As can be observed, regularized DisCoder outperforms Improved GAN by a significant margin, despite using the same network architecture. The samples generated using feature matching are given in Figure 3.

B. Auxiliary classification

Finally, we train the Discoder to obtain real valued embeddings for the STL-10 dataset. The dataset consists of $100,000$

TABLE VI
Network architecture for SVHN

Generator	Encoder
512 ConvT 4x4, relu	dropout(.2)
—	64 Conv 3x3, stride 1, wn, lrelu
—	64 Conv 3x3, stride 1, wn, lrelu
—	64 Conv 3x3, stride 2, wn, lrelu
256 ConvT 4x4, stride 2, bn, relu	dropout(.5)
—	128 Conv 3x3, stride 1, wn, lrelu
—	128 Conv 3x3, stride 1, wn, lrelu
—	128 Conv 3x3, stride 2, wn, lrelu
128 ConvT 4x4, stride 2, bn, relu	dropout(.5)
—	128 Conv 3x3, stride 1, wn, lrelu
—	128 Conv 1x1, stride 1, wn, lrelu
—	128 Conv 1x1, stride 1, wn, lrelu
—	Global average pooling
3 ConvT 4x4, stride 2, tanh	10 linear,wn, softmax

*wn= weight normalization

TABLE VII
Performance of various models on SVHN dataset for the task of semi-supervised classification.

Model	Percentage error using 4000 labeled samples
SDGM [35]	16.61±0.24
ADGM [35]	22.86
Improved GAN [15]	8.11±1.13
Discoder (unregularized)	16.33±2.42
DisCoder	**5.22±0.12**

Fig. 3. Samples generated by the generator using feature matching for semi-supervised learning on SVHN.

unlabelled images of size 64×64 of vehicles and animals. We normalize the images to lie between -1 and 1. The encoding network consists of 3 convolution layers with $64, 128$ and 256 filters respectively of filter size 5×5. The last two layers are fully connected layers with 512 and 10 neurons respectively. We use weight normalization with fixed norm for all but the last layer.

We use a batch size of 500 images. Adam optimizer with a learning rate of $.0003$ is used. The encoding network is trained for 800 epochs. In particular, the objectives in (15) and (10) are optimized alternatively. No adversarial regularization is used for training. After training, the weights of all but the last layer are frozen. Finally, the last layer is trained using labelled data to minimize classification loss.

The trained network achieves a classificiation accuracy of 71.2% on STL-10 dataset. In contrast, the state of the art on STL-10 using fully supervised training is 70.1% [36].

V. Conclusion

In this paper, we introduced a method for encoding the data in an unsupervised manner. We demonstrated the results for the cases when the encoding distribution is either categorical or Gaussian. The learnt embeddings were either used directly in an unsupervised or semi-supervised learning problem or fed to a fully connected neural network for auxiliary classification. Using the learnt embeddings, we were able to achieve the state-of-the-art performance for several well known datasets.

While the objective of an autoencoder encourages it to learn embeddings that minimize reconstruction error, the objective function of a DisCoder encourages the embeddings to be dissimilar to each other. To achieve this task, the DisCoder is forced to capture features that are most discriminative among the samples. For one-hot features, we regularize the model using adversarial regularization to prevent it from overfitting to our initial selection of latent features. However, no such regularization is needed for real valued features because of the continuity of feature space and the stochastic nature of optimization.

In this paper, we have dealt with categorical and Gaussian encoding distributions only. However, the model definition makes no assumptions about the encoding distribution or prior. It will be interesting to explore the utility of the model for handling more complicated embeddings (for instance, a latent Markov chain) in the future.

References

[1] K. Fukushima and S. Miyake, "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition," in *Competition and cooperation in neural nets*. Springer, 1982, pp. 267–285.

[2] Y. LeCun, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel, "Handwritten digit recognition with a back-propagation network," in *Advances in Neural Information Processing Systems*. Citeseer, 1990.

[3] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM international conference on Multimedia*. ACM, 2014, pp. 675–678.

[4] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1440–1448.

[5] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.

[6] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[7] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE, 2013, pp. 6645–6649.

[8] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[9] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural computation*, vol. 14, no. 8, pp. 1771–1800, 2002.

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

[11] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," in *Proceedings of The 31st International Conference on Machine Learning*, 2014, pp. 1278–1286.

[12] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[13] J. Donahue, P. Krähenbühl, and T. Darrell, "Adversarial feature learning," 2016.

[14] J. T. Springenberg, "Unsupervised and semi-supervised learning with categorical generative adversarial networks," *arXiv preprint arXiv:1511.06390*, 2015.

[15] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, X. Chen, and X. Chen, "Improved techniques for training gans," in *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Curran Associates, Inc., 2016, pp. 2234–2242. [Online]. Available: http://papers.nips.cc/paper/6125-improved-techniques-for-training-gans.pdf

[16] M. Noroozi and P. Favaro, "Unsupervised learning of visual representations by solving jigsaw puzzles," in *European Conference on Computer Vision*. Springer, 2016, pp. 69–84.

[17] C. Doersch, A. Gupta, and A. A. Efros, "Unsupervised visual representation learning by context prediction," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1422–1430.

[18] X. Wang and A. Gupta, "Unsupervised learning of visual representations using videos," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2794–2802.

[19] J. Yang, D. Parikh, and D. Batra, "Joint unsupervised learning of deep representations and image clusters," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5147–5156.

[20] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *Proceedings of The 33rd International Conference on Machine Learning*, 2016, pp. 478–487.

[21] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, "On large-batch training for deep learning: Generalization gap and sharp minima," *arXiv preprint arXiv:1609.04836*, 2016.

[22] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[23] R. Salakhutdinov and G. Hinton, "Deep boltzmann machines," in *Artificial Intelligence and Statistics*, 2009, pp. 448–455.

[24] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle *et al.*, "Greedy layer-wise training of deep networks."

[25] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *International Conference on Machine Learning*, 2015, pp. 2048–2057.

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

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

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

[29] A. Dosovitskiy, J. T. Springenberg, M. Riedmiller, and T. Brox, "Discriminative unsupervised feature learning with convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2014, pp. 766–774.

[30] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[31] T. K. Landauer, P. W. Foltz, and D. Laham, "An introduction to latent semantic analysis," *Discourse processes*, vol. 25, no. 2-3, pp. 259–284, 1998.

[32] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[33] A. Rasmus, M. Berglund, M. Honkala, H. Valpola, and T. Raiko, "Semi-supervised learning with ladder networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 3546–3554.

[34] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning."

[35] L. Maaløe, C. K. Sønderby, S. K. Sønderby, and O. Winther, "Auxiliary deep generative models," *arXiv preprint arXiv:1602.05473*, 2016.

[36] K. Swersky, J. Snoek, and R. P. Adams, "Multi-task bayesian optimization," in *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2013, pp. 2004–2012. [Online]. Available: http://papers.nips.cc/paper/5086-multi-task-bayesian-optimization.pdf