Semi-supervised Learning using Deep Generative Models and Auxiliary Tasks

Jhosimar Arias Figueroa
jariasf03@gmail.com

Abstract

In this work, we propose a semi-supervised approach based on generative models to learn both feature representations and categories in an end-to-end manner. The learning process is guided by our proposed auxiliary task that performs assignments for the unlabeled data and regularizes the feature representations with the use of metric embedding methods. Our model is represented by a Gaussian Mixture Variational Autoencoder (GMVAE), in which, we model our categories with the Gumbel-Softmax distribution and we benefit from the autoencoder's architecture to learn feature representations. Experimental results show the effectiveness of our method on three datasets: MNIST, Fashion-MNIST and SVHN.

1 Introduction

Semi-supervised learning [1] aims to leverage limited amounts of labeled data, in conjunction with large amounts of unlabeled data to improve the model performance. In the last few years, Deep Neural Networks (DNNs) have been largely applied to solve this problem due to their ability of learning useful representations. The information of the unlabeled data can be incorporated into these networks through an unsupervised loss [2, 3], pseudo-labels [4, 5] or other auxiliary tasks [6–8]. Recently, Deep Generative Models (DGMs) have been widely used for semi-supervised learning. DGMs model complex high dimensional data by using latent variables. Numerous models are built on Variational Autoencoders (VAEs) [9, 10]. For instance, Kingma et al. [11] proposed the stacked generative model (M1+M2) whose latent space is the joint distribution over data and labels. Maaløe et al. [12] introduced the auxiliary deep generative model (ADGM) which utilizes an extra set of auxiliary latent variables to improve the variational lower bound. Further, Maaløe et al. [12] also propose the Skip Deep Generative Model (SDGM) which shows superior performance compared to an ADGM. Unlike these works, our probabilistic model is based on a GMVAE that does not require pre-training and compared to other GMVAE implementations [13, 14], ours is a modification of the M2 model that considers deterministic feature representations. In addition to that, we use the Gumbel-Softmax distribution [15, 16] to approximate the discrete latent variable.

In this paper, we propose a semi-supervised approach that considers the use of deep generative models to learn feature representations that are guided by our auxiliary tasks in the form of loss functions. Unlike previous semi-supervised probabilistic methods, where the discrete latent variable is considered observable when using labeled data [11, 12], we consider the complete model as unsupervised and we include the knowledge of the labeled data through our auxiliary tasks. Our total loss function is:

$$\mathcal{L}_{total} = \mathcal{L}_{var} + \mathcal{L}_{aux}, \tag{1}$$

where $\mathcal{L}_{var}$ is the variational loss given by the generative model and $\mathcal{L}_{aux}$ is our proposed auxiliary task loss. A general view of our proposed model is depicted in Fig. 1.

4th workshop on Bayesian Deep Learning (NeurIPS 2019), Vancouver, Canada.

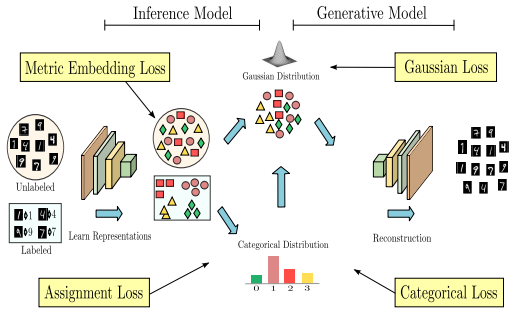
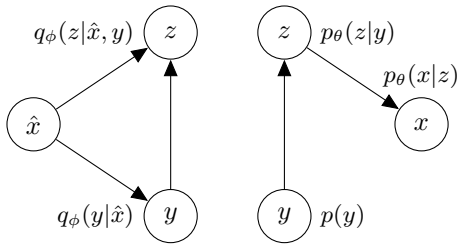


Figure 1: Illustration of the proposed model. Our encoder transforms the data into feature representations, which are used to infer a gaussian and categorical distribution. We take advantage of the small amount of labeled data to perform category assignments for the unlabeled data and regularize the feature space by using our auxiliary tasks.



(a) Inference Model (b) Generative Model

Figure 2: Probabilistic graphical model of our approach. The node $\hat{x}$ in the inference model represents a feature representation obtained from an encoder network.

2 Semi-supervised Learning using Deep Generative Models

For semi-supervised classification, Kingma et al. [11] proposed the stacked generative model M1+M2. This model is comprised of two approaches: a latent-feature discriminative model (M1) and a generative semi-supervised model (M2). The former is represented by a vanilla Variational Autoencoder (VAE) [9, 10] and is used to learn feature representations. The latter generates data from a discrete latent variable y, represented by a categorical distribution, and a continuous variable z, represented by a gaussian distribution. The M2 objective trains a discriminative network $q_\phi(y|x)$, inference network $q_\phi(z|x, y)$ and generative network $p_\theta(x|z, y)$ by maximizing a variational lower bound. For labeled data, the class variable y is observed and z is the latent variable to be inferred. For unlabeled data both y and z are latent variables to be inferred.

The M1+M2 model uses hierarchical stochastic layers that Kingma et al. [11] were not able to train end-to-end, and thus relied on pre-training. The problem of hierarchical stochastic variables is due to the posterior collapse observed in z [17–20], where the distribution $q_\theta(z|x, y)$ collapses to the prior $p(z)$. To avoid this problem, Figueroa et al. [21] replaced the stochastic latent variable of the M1 model with a deterministic one. The posterior collapse is also observed in the latent variable y, Shu et al. [22] performed an analysis of the M2 model for the problem of unsupervised clustering and found that it fails because the distribution $q_\phi(y|x)$ collapses to the prior $p(y)$. Similarly, Willetts et al. [23] found that this collapse also takes place over the subspace of unlabeled classes in the semi-unsupervised problem. To address this problem, Shu [24] proposed a modification of the M2 model, that explicitly models z as a Gaussian mixture conditioned on y.

3 Probabilistic Model

Our probabilistic model is similar to the M1+M2 model proposed by Kingma et al. [11], in which, we replace the stochastic latent variable of the M1 model with a deterministic one to learn feature representations, in a similar way to that proposed by Figueroa et al. [21]. Thus, our learned representations are defined as $\hat{x} = g(x)$, we use these representations as input to our probabilistic model. Our model is represented by a GMVAE, which is based on the modification proposed by Shu [24].

3.1 Generative Model

We use two latent variables to model the data, the continuous variable z and the discrete variable y, i.e., we have $p_\theta(x, z, y)$. Our generative model is defined as $p_\theta(x|z)p_\theta(z|y)p(y)$, cf. Fig. 2b,

$$p(y) = Cat(y|\pi), \tag{2}$$

$$p_\theta(z|y) = \mathcal{N}\left(z|\mu_\theta\left(y\right), \sigma^2_\theta\left(y\right)\right), \tag{3}$$

$$p_\theta(x|z) = \mathcal{N}\left(z|\mu_\theta\left(z\right), \sigma^2_\theta\left(z\right)\right) \text{ or } \mathcal{B}\left(z|\mu_\theta\left(z\right)\right), \tag{4}$$

where $p_\theta(x|z)$ is Gaussian ($\mathcal{N}$) or Bernoulli ($\mathcal{B}$) depending on whether x is continuous or discrete.

3.2 Variational Lower Bound

Computing the posterior $p_\theta(z, y|x)$ is intractable, thus we approximate this posterior with a tractable one, $q_\phi(z, y|\hat{x})$, by using variational inference [9]. Therefore, our inference model is defined as $q_\phi(z|\hat{x}, y)q_\phi(y|\hat{x})$, cf. Fig. 2a,

$$q_\phi(y|\hat{x}) = Cat(y|\pi_\phi(\hat{x})), \tag{5}$$

$$q_\phi(z|\hat{x}, y) = \mathcal{N}\left(z|\mu_\phi\left(\hat{x}, y\right), \sigma^2_\phi\left(\hat{x}, y\right)\right), \tag{6}$$

where $Cat(y|\pi_\phi(\hat{x}))$ is a categorical distribution that we approximate with the Gumbel-Softmax distribution [15, 16]. As aforementioned, we consider our probabilistic model as unsupervised, i.e., we require to infer both latent variables y and z. Therefore, we optimize the model by maximizing the following lower bound:

$$\log p_\theta(x) \geq \mathbb{E}_{q_\phi(y|\hat{x})}\left[\mathbb{E}_{q_\phi(z|\hat{x},y)}\left[\log p_\theta(x|z)\right] - \mathrm{KL}\left(q_\phi\left(z|\hat{x}, y\right)||p_\theta\left(z|y\right)\right)\right] - \mathrm{KL}\left(q_\phi\left(y|\hat{x}\right)||p\left(y\right)\right) \tag{7}$$

where the first term is given by the reconstruction loss ($\mathcal{L}_R$), the second term is the mixture of gaussians loss ($\mathcal{L}_G$) and the last term is denoted as the categorical loss ($\mathcal{L}_C$). Therefore, our variational loss is defined as $\mathcal{L}_{var} = \mathcal{L}_R + w_G\mathcal{L}_G + w_C\mathcal{L}_C$, where w_G and w_C are weights that specify the importance of the gaussian and categorical loss functions.

4 Auxiliary Tasks

Our main objective is to guide the learning process of our probabilistic model to perform semi-supervised classification. To achieve it, we propose two auxiliary tasks in the form of loss functions: the assignment loss ($\mathcal{L}_A$) and the metric embedding loss ($\mathcal{L}_M$). The first one considers the feature representations of the data to learn category assignments, and the second one regularizes the feature representations with the use of metric embedding methods. Therefore, our auxiliary task loss is given by $\mathcal{L}_{aux} = w_A\mathcal{L}_A + w_M\mathcal{L}_M$, where the weights w_A and w_M define the importance of each task.

4.1 Assignment Loss

Our first auxiliary task assigns labels to the unlabeled data. In order to do it, we use the distance-weighted k-nearest neighbor (k-NN) [25] on the feature space, i.e., we compare the features of the unlabeled data, $\hat{x}_u$, with the features of the labeled data, $\hat{x}_l$, and assign the label based on the contribution or weight of each of the k nearest neighbors. The assigned label for each unlabeled data is given by:

$$y = \arg\max_j \sum_{(x_i, y_i)\in\mathcal{D}_k} w_i \times I(j = y_i), \tag{8}$$

where y is the assigned label, $\mathcal{D}_k$ is the set of the k closest samples, w_i is the weight of the i sample and is given by $1/\mathrm{distance}(\hat{x}_u, \hat{x}_l)$. A similar assignment process was proposed by Hoffer et al. [26] and Figueroa et al. [8] but they considered only the nearest neighbor (1-NN). Once we obtained the assignments for the unlabeled data, we teach our network about these assignments with the cross-entropy loss that we denote as Assignment Loss ($\mathcal{L}_A$).

4.2 Metric Embedding Loss

Our second auxiliary task regularizes the feature space. In order to do it, we use metric learning methods [27, 28] on the feature space, in such a way that the distance between features that belong to the same category should be smaller than features of different categories. In our experiments, we use the lifted structured loss [28], but any metric learning method can be used. This loss function is applied after the assignment process because it requires all the data labeled. We denote this loss as the Metric Embedding Loss ($\mathcal{L}_M$).

Table 1: Semi-supervised test error (%) benchmarks on MNIST, SVHN and Fashion-MNIST for randomly and evenly distributed labeled data. The results with (*) on top use an additional extra set for training.

Method	MNIST	SVHN	Fashion-MNIST	
	100	1000	100	1000
M1+TSVM [11]	11.82 ($\pm$ 0.25)	55.33 ($\pm$ 0.11)	-	-
M1+M2 [11]	3.33 ($\pm$ 0.14)	36.02 ($\pm$ 0.10)	-	-
SS-Clustering [8]	1.66 ($\pm$ 0.20)	20.66 ($\pm$ 1.09)	-	-
SDGM [12]	1.32 ($\pm$ 0.07)	16.61* ($\pm$ 0.24)	-	-
ADGM [12]	0.96 ($\pm$ 0.02)	22.86*	-	-
Proposed	1.94 ($\pm$ 0.49)	14.78 ($\pm$ 0.40)	26.88 ($\pm$ 0.86)	15.85 ($\pm$ 0.69)

5 Experiments

5.1 Datasets

We evaluate our method on three image datasets. MNIST [29]: a dataset containing a total of $70,000$ handwritten digits with $60,000$ training and $10,000$ testing samples, each being a 28×28 image. SVHN [30]: consists of $99,289$ house number digits with $73,257$ training and $26,032$ testing samples of size $3 \times 32 \times 32$. There is an additional extra set of $531,131$ samples for training. However, we do not consider this additional set in our experiments. Fashion-MNIST [31]: This dataset has the same number of images and the same image size of MNIST, but it is more complicated. Instead of digits, it consists of various types of fashion products. For image pre-processing, we normalize the image intensities to be in the range of $[0, 1]$ for all the datasets, and consequently use the sigmoid function in the last layer of the decoder. We use the pre-defined train/test splits of each data set, from the training set we consider 80% for training and 20% for validation. Furthermore, we randomly chose 100 labeled examples for MNIST and Fashion-MNIST, and 1000 for SVHN and Fashion-MNIST that are fixed for the whole training phase. We ensure that all classes have the same number of labeled images.

5.2 Implementation Details

We employ convolutional neural networks to represent our encoder and decoder networks and fully connected multilayer networks to infer our latent variables. Our inference and generative models are parameterized by three neural networks: gaussians model $q_\phi(z|\hat{x}, y)$, categorical model $q_\phi(y|\hat{x})$ and generative model $p_\theta(x|z, y)$. Detailed model architectures can be found in Appendix A.

For training we use Adam [32] optimizer with a learning rate of 0.001. We iterate for 200 epochs and at each epoch we consider a different random permutation of our data. We use a batch size of 200 for MNIST and Fashion-MNIST, and 400 for SVHN. Half of the batch is composed by labeled data that is randomly selected. The feature size (f_{sz}) is 100 for both MNIST and SVHN, and 200 for Fashion-MNIST, the gaussian size (g_{sz}) is 100 for MNIST, 150 for SVHN and 50 for Fashion-MNIST, we use 5 nearest neighbors in the assignment process for MNIST and 3 for SVHN and Fashion-MNIST. The weights of the auxiliary tasks are $w_A = 5$ and $w_M = 3$ for SVHN, $w_A = w_M = 1$ for MNIST and $w_A = 1$ and $w_M = 0.1$ for Fashion-MNIST. The margin used in the metric embedding loss is 2 for MNIST, 0.5 for SVHN and 1 for Fashion-MNIST. We use the mean square error in the reconstruction loss for all the datasets. Finally, we smooth the temperature used in Gumbel-Softmax from 1 to 0.5 in the first 50 epochs. All the hyperparameters were found in the validation set using random search.

5.3 Quantitative Results

We run our method with 5 random seeds and report the average performance. All the results of the related works were reported from the original papers. The character '-' means that results for that metric or setup were not executed. We compared our method with probabilistic generative models applied to semi-supervised classification.

As we can see in Table 1, for the MNIST dataset, our method obtained comparable results with SDGM [12] and ADGM [12], which are models that use skip connections between the input data and all the latent variables in the inference model. We can see the effectiveness of our method on the

Table 2: Semi-supervised test error (%) of our probabilistic model when we train it only with the assignment loss ($\mathcal{L}_A$) and jointly with the metric embedding loss ($\mathcal{L}_M$) on the MNIST, SVHN and Fashion-MNIST datasets.

Auxiliary Task	MNIST	SVHN	Fashion-10
	100	1000	100
$\mathcal{L}_A$	10.16	58.77	29.98
$\mathcal{L}_A + \mathcal{L}_M$	1.94	14.78	26.88

Table 3: Clustering performance, ACC(%) and NMI(%), on MNIST and Fashion-MNIST.

Method	MNIST		Fashion-10	
	ACC	NMI	ACC	NMI
GMVAE [13]	82.31	-	-	-
IDEC [33]	88.06	86.72	52.90	55.70
VaDE [14]	94.46	87.60	57.80	63.00
JULE-RC [34]	96.40	91.30	56.30	60.80
DEPICT [35]	96.50	91.70	39.20	39.20
DualAuto [36]	97.80	94.10	66.20	64.50
Proposed (Avg.)	92.46	88.16	55.97	54.55
Proposed (Best)	97.36	93.44	59.05	57.56

SVHN dataset, where we outperformed all the probabilistic methods without the use of the additional extra training set. We also reported results for the Fashion-MNIST, where we did not get as good results as MNIST because this is a harder dataset.

Besides the comparison with state-of-the-art methods, we performed an ablation study over the auxiliary tasks. To do so, we trained our probabilistic model only with the assignment loss ($\mathcal{L}_A$) and with both assignment loss ($\mathcal{L}_A$) and metric embedding loss ($\mathcal{L}_M$). Table. 2 shows the results of this study, we can see that with the assignment loss we are able to obtain reasonable results on MNIST and Fashion-MNIST. However, it is not very useful on the SVHN dataset. Recall that this loss depends on the feature representations to obtain assignments for classification. Therefore, if the feature representations are not learned properly, the assignment loss will misclassify the data. According to the results, regularizing the feature space with the metric embedding loss is of great importance to improve the performance of our model and achieve competitive results.

5.3.1 Unsupervised Clustering

Our proposed model can be applied to clustering. To do so, we train our probabilistic model without labeled data. We examine the clustering performance on the MNIST and Fashion-MNIST datasets. For this task, we consider a different set of hyperparameters. We use a batch size of 64 for both datasets. The feature size (f_{sz}) is 100 for MNIST and 200 for Fashion-MNIST, the gaussian size (g_{sz}) is 150 for MNIST and 50 for Fashion-MNIST. Finally, the weight of the gaussian loss (w_G) is set to 5. We train our models with the binary cross entropy loss for reconstruction. This set of hyperparameters gives us better results than those used in the semi-supervised task.

We evaluate our clustering results with two metrics commonly used in the literature, clustering accuracy (ACC) and normalized mutual information (NMI) [37]. We report the average and best results of 5 random seeds. We compare our method with probabilistic generative models [13, 14] and models that use convolutional networks [33–36]. As we can see in Table. 3, our average results are competitive with the state-of-the-art and the best results outperform different methods. Compared to probabilistic generative models, our model outperforms the GMVAE proposed by Dilokthanakul et al. [13] and our best run outperforms VaDE [14] on the MNIST and Fashion-MNIST datasets, these methods are very related to us because they also model the data with a mixture of gaussians.

5.4 Qualitative Results

Besides classification, we also evaluated the generative part of our model. Fig. 3 shows how our model can generate random images for each category. Fig. 4 depicts the feature representations of all the datasets using t-SNE [38], we reduced the dimensionality of the feature representations to 2D and plot the test sets. Different colors indicate ground-truth classes. We can see that for MNIST and SVHN datasets we obtained a good separability of the categories. However, for Fashion-MNIST the separability is not very clear because some samples of different categories are mixed.

(a) MNIST (b) SVHN (c) Fashion-MNIST

Figure 3: Generated images of our proposed model trained on MNIST, SVHN and Fashion-MNIST datasets.

(a) MNIST (b) SVHN (c) Fashion-MNIST

Figure 4: Visualization of the feature representations of MNIST, SVHN and Fashion-MNIST test set.

6 Conclusions

In this paper, we proposed a semi-supervised method that combines the variational loss given by a GMVAE with our proposed auxiliary tasks that assign labels to the unlabeled data and regularize the feature space using metric embedding methods. Our approach is not restricted to GMVAE and can be applied to other models like M2 [11] or ADGM [12]. Experimental results show that our approach can obtain competitive results by adding loss functions that drive the learning process of our generative model. One problem of the current version of our approach is the fact that we require to change the weights of some loss functions in order to get good results. Future work focuses on the application of methods to avoid posterior collapse like InfoVAE [39]. Furthermore, we can improve the assignment process using more complex techniques.

References

[1] O. Chapelle, B. Schlkopf, and A. Zien. *Semi-Supervised Learning*. The MIT Press, 1st edition, 2010. ISBN 0262514125, 9780262514125.

[2] Y. Grandvalet and Y. Bengio. Semi-supervised learning by entropy minimization. In *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, 2004.

[3] J. Weston, F. Ratle†, and R. Collobert. Deep learning via semi-supervised embedding. In *International Conference on Machine Learning (ICML)*, 2008.

[4] D.-H. Lee. Pseudo-label : The simple and efficient semi-supervised learning method for deep neural networks. *ICML 2013 Workshop : Challenges in Representation Learning (WREPL)*, 2013.

[5] A. Iscen, G. Tolias, Y. Avrithis, and O. Chum. Label propagation for deep semi-supervised learning. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[6] A. Rasmus, M. Berglund, M. Honkala, H. Valpola, and T. Raiko. Semi-supervised learning with ladder networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[7] J. Zhao, M. Mathieu, R. Goroshin, and Y. LeCun. Stacked what-where auto-encoders. In *International Conference on Learning Representations (ICLR)*, 2016.

[8] J. A. Figueroa and A. R. Rivera. Learning to cluster with auxiliary tasks: A semi-supervised approach. In *Conference on Graphics, Patterns and Images (SIBGRAPI)*, 2017.

[9] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

[10] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning (ICML)*, volume 22, 2014.

[11] 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 (NIPS)*, 2014.

[12] L. Maaløe, C. K. Sønderby, S. K. Sønderby, and O. Winther. Auxiliary deep generative models. In *International Conference on Machine Learning (ICML)*, 2016.

[13] N. Dilokthanakul, P. A.M. Mediano, M. Garnelo, M. C.H. Lee, H. Salimbeni, K. Arulkumaran, and M. Shanahan. Deep unsupervised clustering with gaussian mixture variational cvprs. In *arXiv preprint arXiv:1611.02648*, 2016.

[14] Z. Jiang, Y. Zheng, H. Tan, B. Tang, and H. Zhou. Variational deep embedding: A generative approach to clustering. In *International Joint Conference on Artificial Intelligence*, 2017.

[15] E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations (ICLR)*, 2016.

[16] C. J. Maddison, A. Mnih, and Y. W. Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations (ICLR)*, 2016.

[17] C. K. Sønderby, T. Raiko, L. Maaløe, S. K. Sønderby, and O. Winther. Ladder variational autoencoders. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[18] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[19] L. Maaløe, M. Fraccaro, and O. Winther. Semi-supervised generation with cluster-aware generative models. In *arXiv preprint arXiv:1704.00637*, 2017.

[20] X. Chen, D. P. Kingma, T. Salimans, Y. Duan, P. Dhariwal, J. Schulman, I. Sutskever, and P. Abbeel. Variational lossy autoencoder. In *International Conference on Learning Representations (ICLR)*, 2017.

[21] J. A. Figueroa and A. R. Rivera. Is simple better?: Revisiting simple generative models for unsupervised clustering. In *Second workshop on Bayesian Deep Learning (NIPS)*, 2017.

[22] R. Shu, J. Brofos, and C. Langlotz. A note on deep variational models for unsupervised clustering. 2017.

[23] M. Willetts, S. J Roberts, and C. C Holmes. Semi-unsupervised learning with deep generative models: Clustering and classifying using ultra-sparse labels. *arXiv preprint arXiv:1901.08560*, 2019.

[24] R. Shu. Gaussian mixture vae: Lessons in variational inference, generative models, and deep nets. `http://ruishu.io/2016/12/25/gmvae/`, 2016.

[25] S. A. Dudani. The distance-weighted k-nearest-neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-6(4):325–327, April 1976. ISSN 0018-9472. doi: 10.1109/TSMC.1976.5408784.

[26] E. Hoffer and N. Ailon. Semi-supervised deep learning by metric embedding. In *arXiv preprint arXiv:1611.01449*, 2016.

[27] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[28] H. O. Song, Y. Xiang, S. Jegelka, and S. Savarese. Deep metric learning via lifted structured feature embedding. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[29] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

[30] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2011.

[31] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017.

[32] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[33] X. Liu J. Yin X. Guo, L. Gao. Improved deep embedded clustering with local structure preservation. In *International Joint Conference on Artificial Intelligence*, 2017.

[34] J. Yang, D. Parikh, and D. Batra. Joint unsupervised learning of deep representations and image clusters. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[35] K. G. Dizaji, A. Herandi, and H. Huang. Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization. In *arXiv preprint arXiv:1704.06327*, 2017.

[36] X. Yang, C. Deng, F. Zheng, J. Yan, and W. Liu. Deep spectral clustering using dual autoencoder network. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[37] W. Xu, X. Liu, and Y. Gong. Document clustering based on non-negative matrix factorization. In *International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, pages 267–273, 2003.

[38] L. v. d. Maaten and G. Hinton. Visualizing high-dimensional data using t-sne. In *The Journal of Machine Learning Research*, pages 2579–2605, 2008.

[39] S. Zhao, J. Song, and S. Ermon. Infovae: Balancing learning and inference in variational autoencoders. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019.

Appendix

A Network Architectures

Table 4 provides a description of the encoder architecture used to learn feature representations, $\hat{x} = g(x)$. The discriminative network $q_\phi(y|\hat{x})$, is given by the set of linear layers: $f_{sz} - f_{sz} - c$, where f_{sz} is the feature size of $\hat{x}$, c is the number of classes, the output vector is given by the Gumbel-Softmax distribution. The architecture of the inference network $q_\phi(z|\hat{x}, y)$, is $(f_{sz} + c) - z_{sz} - z_{sz} - z_{sz}$, where the input is the concatenation of the features and categories, z_{sz} is the size of the gaussian latent variable, its output vector is given by the reparameterization trick of the gaussian distribution. The architecture of $p_\theta(z|y)$ is given by two linear layers for μ and σ^2 respectively. All the linear layers are followed by Leaky ReLU non-linearity. Finally, the architecture of the generative model $p_\theta(x|z)$ is given in Table 5.

Table 4: **Neural Network Architecture of ($\hat{x} = g(x)$):** Input is an image, all the layers are followed by Batch Normalization and Leaky ReLU non-linearity. Convolutional layers are specified as (filters@filterSize, stride, padding). The output is a feature vector of size f_{sz}.

Dataset	layer1	layer2	layer3	layer4	layer5
MNIST	$(16@5, 1, 1)$	$(32@5, 1, 1)$	$(64@4, 2, 2)$	$(128@4, 2, 2)$	$(f_{sz}@5, 1, 1)$
SVHN	$(16@5, 1, 1)$	$(32@5, 1, 1)$	$(64@4, 2, 2)$	$(128@4, 2, 2)$	$(f_{sz}@6, 1, 1)$
Fashion-10	$(16@5, 1, 1)$	$(32@5, 1, 1)$	$(64@4, 2, 2)$	$(128@4, 2, 2)$	$(f_{sz}@5, 1, 1)$

Table 5: **Neural Network Architecture of $p_\theta(x|z)$:** The input is a vector of size f_{sz}. All the layers are followed by Batch Normalization and ReLU non-linearity. Convolutional layers are specified as (filters@filterSize, stride, padding). The output is an image.

Dataset	layer1	layer2	layer3	layer4	layer5
MNIST	$(128@5, 1, 1)$	$(64@4, 2, 2)$	$(32@4, 2, 2)$	$(16@5, 1, 1)$	$(1@5, 1, 1)$
SVHN	$(128@6, 1, 1)$	$(64@4, 2, 2)$	$(32@4, 2, 2)$	$(16@5, 1, 1)$	$(3@5, 1, 1)$
Fashion-10	$(128@5, 1, 1)$	$(64@4, 2, 2)$	$(32@4, 2, 2)$	$(16@5, 1, 1)$	$(1@5, 1, 1)$