



저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:



저작자표시. 귀하는 원저작자를 표시하여야 합니다.



비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.



변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 [이용허락규약\(Legal Code\)](#)을 이해하기 쉽게 요약한 것입니다.

[Disclaimer](#)

이 학 박 사 학 위 논 문

Deep learning methodologies  
for semi-supervised and unsupervised cases

준지도 및 비지도 학습에서의  
딥러닝 방법론 연구

2019년 8월

서울대학교 대학원

통계학과

김 동 하

Deep learning methodologies  
for semi-supervised and unsupervised cases

준지도 및 비지도 학습에서의  
딥러닝 방법론 연구

지도교수 김 용 대

이 논문을 이학박사 학위논문으로 제출함  
2019년 7월

서울대학교 대학원  
통계학과  
김 동 하

김동하의 이학박사 학위논문을 인준함  
2019년 7월

위 원 장 Myunghee Cho Paik (인)

---

부위원장 김 용 대 (인)

---

위 원 원 중 호 (인)

---

위 원 정 성 규 (인)

---

위 원 전 종 준 (인)

---

**Deep learning methodologies for  
semi-supervised and unsupervised cases**

**By**

**Dongha Kim**

**A Thesis**

**Submitted in fulfillment of the requirement**

**for the degree of**

**Doctor of Philosophy**

**in Statistics**

**Department of Statistics  
College of Natural Sciences  
Seoul National University  
August, 2019**

## ABSTRACT

# Deep learning methodologies for semi-supervised and unsupervised cases

Dongha Kim

The Department of Statistics

The Graduate School

Seoul National University

In this thesis, we propose two learning methodologies for deep learning models. We consider two cases: semi-supervised learning and unsupervised learning.

In semi-supervised learning, we spell out a new semi-supervised learning method, called *GAB*, that searches for a decision boundary whose neighborhood overlaps the least with the support of unlabeled data. We construct a formal measure of the degree of overlap between the neighborhood of a given decision boundary and the support of unlabeled data and develop an algorithm to learn the model, which minimizes this penalty term. We theoretically prove that *GAB* finds the Bayes classifier successively and

devise an algorithm with an approximated penalty term by generating artificial data near the current decision boundary based on an adversarial training technique. We empirically show that *GAB* not only competes well with the recent studies in prediction power but also requires much smaller computational resources.

In unsupervised learning, we propose a method for generative models maximizing the log-likelihood of observable variables directly by using the *EM* algorithm and an importance sampling algorithm, instead of employing variational inference. A novel feature of the proposed method is to develop a warm start technique by taking a convex combination of the expected complete log-likelihood and variational lower bound in the E-step, which stabilizes the learning procedure and thus results in superior performance. The proposed learning method called *VAEM* outperforms other variational methods in terms of the test log-likelihood without increasing computational cost much, generates more sharp and realistic images, and can be easily modified for nonstandard cases such as the presence of missing data which is not obvious for variational methods.

**Keywords:** Semi-supervised learning, Cluster assumption, Adversarial training, *GAB*, Unsupervised learning, *EM* algorithm, Importance sampling, Annealing strategy, *VAEM*

**Student Number:** 2012 – 23010

# Contents

<b>Abstract</b>	<b>i</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Estimation algorithm for semi-supervised deep learning</b>	<b>4</b>
2.1 Introduction . . . . .	4
2.2 Review of existing semi-supervised learning methods in deep learning . . . . .	8
2.2.1 <i>DGN</i> [Kingma et al., 2014] . . . . .	8
2.2.2 <i>Ladder network</i> [Rasmus et al., 2015] . . . . .	9
2.2.3 <i>VAT</i> [Miyato et al., 2017] . . . . .	10
2.2.4 <i>Bad GAN</i> [Dai et al., 2017] . . . . .	11
2.3 Proposed Method . . . . .	12
2.3.1 Motivation . . . . .	12
2.3.2 Theoretical Analysis . . . . .	14
2.3.3 Objective Function of <i>GAB</i> . . . . .	16
2.4 Generation of artificial samples . . . . .	17

2.4.1	Adversarial training . . . . .	17
2.4.2	Motivation . . . . .	18
2.4.3	Artificial sample generation for general classifier . . . . .	19
2.4.4	Finding adversarial direction . . . . .	20
2.5	Regularization term . . . . .	21
2.6	Experiments . . . . .	22
2.6.1	Prediction performance comparisons . . . . .	22
2.6.2	Effects of Tuning Parameters . . . . .	24
2.6.3	Computational Efficiency . . . . .	26
2.6.4	Quality of Artificial Samples . . . . .	26
<b>3</b>	<b>Estimation algorithm of deep generative model</b>	<b>32</b>
3.1	Introduction . . . . .	32
3.2	Review of existing methods for training deep generative models . . . . .	36
3.2.1	<i>VAE</i> [Kingma and Welling, 2013] . . . . .	36
3.2.2	<i>IWAE</i> [Burda et al., 2015] . . . . .	37
3.2.3	<i>HMC-DLGM</i> [Hoffman, 2017] . . . . .	38
3.3	Proposed method . . . . .	39
3.3.1	<i>VAEM</i> for complete data . . . . .	39
3.3.2	Role of Annealing . . . . .	41
3.3.3	<i>VAEM</i> for incomplete data . . . . .	43
3.4	Empirical analysis . . . . .	46
3.4.1	Experimental setup . . . . .	46
3.4.2	Performance results . . . . .	47

3.4.3	Image generation . . . . .	49
3.4.4	Ablation study . . . . .	49
3.4.5	<i>VAEM</i> for incomplete data . . . . .	50
<b>4</b>	<b>Conclusions</b>	<b>65</b>
	<b>Bibliography</b>	<b>67</b>
<b>A</b>	<b>Appendix A.</b>	<b>78</b>
A.1	Proof of Theorem 2.3.1. . . . .	78
A.2	Proof of proposition 2.4.1. . . . .	81
A.3	Extension of proposition 1 for the DNN classifier .	82
A.4	Model architectures . . . . .	84
A.4.1	MNIST . . . . .	84
A.4.2	SVHN, CIFAR10 and CIFAR100 . . . . .	84
<b>B</b>	<b>Appendix B.</b>	<b>86</b>
B.1	Generated Images . . . . .	86
	<b>Abstract (in Korean)</b>	<b>91</b>

# List of Tables

2.1	Test accuracies for <i>GAB</i> with and without the regularization term. The numbers in the parenthesis are the sizes of labeled data. . . . .	22
2.2	Comparison of prediction accuracies of various semi-supervised learning algorithms for benchmark datasets. $ \mathcal{L} $ is the number of labeled data and the results with * are implemented by us. . . . .	28
2.3	Test accuracies of MNIST with 100 labeled data for various values of $c, C$ . and $\alpha$ . The other parameters for each case are fixed at the optimal values. . . . .	29
2.4	Learning time per training epoch ratio compared to supervised learning with cross-entropy for CIFAR10. <i>Bad GAN</i> is operated without <i>PixelCNN++</i> . . . . .	29
3.1	Test log-likelihood values of the four learning methods: 1) <i>VAE</i> , 2) <i>VAEM</i> with $\alpha = 1$ and the initial $\theta$ and $\phi$ estimated by <i>VAE</i> , 3) <i>VAEM</i> with $\alpha = 1$ and random initial for $\theta$ and $\phi$ and 4) <i>VAEM</i> (with annealing $\alpha$ and random initial for $\theta$ and $\phi$ ). . . . .	53

3.2	Test log-likelihood between different models and learning methods with the <i>SG</i> prior. For <i>IWAE</i> , $K = 50$ is used for <i>MLP</i> and $K = 10$ is used for other models. . . . .	54
3.3	Test log-likelihood between different models and learning methods with the mixture prior distributions. . . . .	55
3.4	Test log-likelihood for static biMNIST. . . . .	56
3.5	Test log-likelihood for dynamic biMNIST. . . . .	56
3.6	Test log-likelihood for Omniglot. . . . .	57
3.7	Test log-likelihood for Caltech 101. . . . .	57
3.8	Margins of the test log-likelihood values from the Figure 3.6 for various number of hidden nodes. . . . .	57
A.1	CNN models used in experimental analysis over SVHN, CIFAR10, and CIFAR100. We used leaky ReLU activation function [Maas et al., 2013], and all the convolutional layers and fully-connected layers are followed by BN[Ioffe and Szegedy, 2015]. . . . .	85

# List of Figures

2.1	Demonstration of how the artificial samples generated by the adversarial training are distributed. We consider two cases: 3-class classification problem ( <b>Left</b> ) and 4-class classification problem ( <b>Right</b> ). True data and artificial data are colored blue and orange, respectively. . . . .	20
2.2	Examples of $P(y = 1 x)$ of smooth ( <b>Left</b> ) and wiggle ( <b>Right</b> ) cases. We plotted 3 points and their adversarial directions for each case. . . . .	23
2.3	( <b>Upper</b> ) 10 randomly sampled original MNIST dataset. ( <b>Middle and Lower</b> ) Artificial samples obtained by classifiers learned with and without the regularization term of <i>VAT</i> . . . . .	24
2.4	Trace plot of the test accuracies with the three semi-supervised learning methods and the supervised learning method with cross-entropy for MNIST with 20 labeled data. . . . .	25

2.5	( <b>Upper left</b> ) The scatter plot of synthetic data which consist of 1000 unlabeled data (gray) and 4 labeled data for each class (red and blue with black edge). ( <b>Upper right</b> ) Accuracies of unlabeled data for each epoch for <i>VAT</i> and <i>GAB</i> . We use 2-layered NN with 100 hidden units each. ( <b>Else</b> ) Artificial samples and classified unlabeled data by colors at the 20,40,60 and 80 training epochs of <i>GAB</i> . . . .	30
2.6	The number of epochs to achieve the pre-specified test accuracies (98%, 90% and 80%) with the three methods for ( <b>Left</b> ) MNIST (100), ( <b>Middle</b> ) SVHN (1000) and ( <b>Right</b> ) CIFAR10 (4000) settings. <i>Bad GAN</i> is operated without <i>PixelCNN++</i> for SVHN and CIFAR10 datasets. . . . .	31
2.7	100 randomly sampled artificial images and bad images with ( <b>Left</b> ) <i>GAB</i> and ( <b>Right</b> ) <i>Bad GAN</i> respectively. . . . .	31
3.1	Image generation with <i>VAEM</i> for static MNIST dataset. Images generated with ( <b>1st row</b> ) <i>MLP (SG)</i> and <i>MLP (MoG)</i> , ( <b>2nd row</b> ) <i>CNN (SG)</i> and <i>CNN (MoG)</i> and ( <b>3rd row</b> ) <i>PixelCNN (SG)</i> and <i>PixelCNN (MoG)</i> . . . . .	58

3.2	Image generation with <i>VAEM</i> for dynamic MNIST dataset. Images generated with <b>(1st row)</b> <i>MLP (SG)</i> and <i>MLP (MoG)</i> , <b>(2nd row)</b> <i>CNN (SG)</i> and <i>CNN (MoG)</i> and <b>(3rd row)</b> <i>PixelCNN (SG)</i> and <i>PixelCNN (MoG)</i> . . . . .	59
3.3	Image generation with <i>VAEM</i> for Omniglot dataset. Images generated with <b>(1st row)</b> <i>MLP (SG)</i> and <i>MLP (MoG)</i> , <b>(2nd row)</b> <i>CNN (SG)</i> and <i>CNN (MoG)</i> and <b>(3rd row)</b> <i>PixelCNN (SG)</i> and <i>PixelCNN (MoG)</i> . . . . .	60
3.4	Image generation with <i>VAEM</i> for Caltech 101 dataset. Images generated with <b>(1st row)</b> <i>MLP (SG)</i> and <i>MLP (MoG)</i> , <b>(2nd row)</b> <i>CNN (SG)</i> and <i>CNN (MoG)</i> and <b>(3rd row)</b> <i>PixelCNN (SG)</i> and <i>PixelCNN (MoG)</i> . . . . .	61
3.5	Test log-likelihood for various values of the sample size $m$ for <i>IS</i> in <i>VAEM</i> on static biMNIST <b>(top)</b> and Omniglot dataset <b>(bottom)</b> . The MLP architecture and the <i>SG</i> prior are used for the generative model. . . . .	62
3.6	Test log-likelihood of <i>VAEM</i> and <i>VAE</i> for varying numbers of hidden nodes in the proposed (or variational) model on static biMNIST dataset. The <i>MLP</i> architecture and the <i>SG</i> prior are used for the generative model. . . . .	63

3.7	Test log-likelihood of <i>missVAEM</i> and <i>missIWAE</i> for various missing rates on the static biMNIST dataset. The MLP architecture and the <i>SG</i> prior are used for the generative model. . . . .	64
B.1	Generated images for static MNIST dataset with various architectures and learning methods. <b>(1st row)</b> <i>MLP+VAE (SG)</i> , <i>MLP+VAE (MoG)</i> and <i>MLP+VP</i> , <b>(2nd row)</b> <i>CNN+VAE (SG)</i> , <i>CNN+VAE (MoG)</i> and <i>CNN+VP</i> , and <b>(3th row)</b> <i>PixelCNN+VAE (SG)</i> , <i>PixelCNN+VAE (MoG)</i> and <i>PixelCNN+VP</i> are considered. . . . .	87
B.2	Generated images for dynamic MNIST dataset with various architectures and learning methods. <b>(1st row)</b> <i>MLP+VAE (SG)</i> , <i>MLP+VAE (MoG)</i> and <i>MLP+VP</i> , <b>(2nd row)</b> <i>CNN+VAE (SG)</i> , <i>CNN+VAE (MoG)</i> and <i>CNN+VP</i> , and <b>(3th row)</b> <i>PixelCNN+VAE (SG)</i> , <i>PixelCNN+VAE (MoG)</i> and <i>PixelCNN+VP</i> are considered. . . . .	88
B.3	Generated images for Omniglot dataset with various architectures and learning methods. <b>(1st row)</b> <i>MLP+VAE (SG)</i> , <i>MLP+VAE (MoG)</i> and <i>MLP+VP</i> , <b>(2nd row)</b> <i>CNN+VAE (SG)</i> , <i>CNN+VAE (MoG)</i> and <i>CNN+VP</i> , and <b>(3th row)</b> <i>PixelCNN+VAE (SG)</i> , <i>PixelCNN+VAE (MoG)</i> and <i>PixelCNN+VP</i> are considered. . . . .	89

B.4 Generated images for Caltech 101 dataset with various architectures and learning methods. **(1st row)** *MLP+VAE (SG)*, *MLP+VAE (MoG)* and *MLP+VP*, **(2nd row)** *CNN+VAE (SG)*, *CNN+VAE (MoG)* and *CNN+VP*, and **(3th row)** *PixelCNN+VAE (SG)*, *PixelCNN+VAE (MoG)* and *PixelCNN+VP* are considered. . . . . 90

# Chapter 1

## Introduction

Deep learning is a class of models with deep architectures that originated from artificial neural networks; it also refers to a class of effective learning methods for training deep architectures. Deep learning has accomplished substantial success because of the development of deep architectures, learning techniques, and hardwares [Hinton et al., 2012; Ioffe and Szegedy, 2015; Kingma and Ba, 2014; Krizhevsky et al., 2012; Szegedy et al., 2015], and thus, has made remarkable progress in character and speech recognition, image classification, and many other applications [Chung et al., 2014; Seide et al., 2011; Sutskever et al., 2014].

In recent years, the demands for applying deep learning methodologies to various machine learning fields have increased, especially for semi-supervised learning and unsupervised learning. For highly complicated data such as image data with high resolution, it is hard to collect large amount of labeled data because of the

high cost; thus, semi-supervised learning methodologies that utilize large unlabeled data as well as small labeled data to train a classifier need to be devised.

Furthermore, it is important to establish the distribution of complex and high dimensional data [Goodfellow et al., 2014a; Kingma and Welling, 2013; Rezende et al., 2014]. Image or audio data in the real world are of high dimensionality but have distributions made by an intrinsic structure with much lower dimensions, that is, the distribution of these data is entangled with complex functions. Deep learning can realize the highly nonlinear functions and thus can obtain the distribution of high dimensional data.

In Chapter 2, we propose a new semi-supervised learning method, called *GAB* (generating artificial data on the decision boundary). Motivated by one of the standard assumptions in semi-supervised learning, called *cluster assumption*, we devise a specially designed regularization term that measures the degree of overlap between the neighborhood of a given decision boundary and the support of unlabeled data and develop a learning method which minimizes this regularization term. We theoretically prove that the proposed method can successively find the Bayes classifier. We also devise a method approximating the regularization term with only a classifier by generating artificial data near the current decision boundary based on the adversarial training technique. We demonstrate that *GAB* competes well with other recent studies and at the same time requires much less amount of computational resources.

In Chapter 3, we propose a new learning method to learn deep generative models. Instead of employing the variational inference, we maximize the marginal log-likelihood of observable data directly by using the EM algorithm and importance sampling (IS). The novel feature of the proposed learning method is that we apply the idea of warm-up by taking a convex combination of the expected complete log-likelihood and the ELBO function to stabilize the E-step with IS at the early stage of learning. We call this method the *VAEM* (variational annealed expectation-maximization) algorithm. We show that *VAEM* improves the performance in terms of the test log-likelihood significantly and generates more sharp and realistic images compared with the variational methods. In addition, *VAEM* can be easily modified for nonstandard cases such as the analysis for incomplete data, which is not obvious for variational methods.

In Chapter 4, we summarize the proposed methods of this thesis and discuss future work.

## Chapter 2

# Estimation algorithm for semi-supervised deep learning

### 2.1 Introduction

Deep learning has accomplished unprecedented success with the development of deep architectures, learning techniques, and hardware [Hinton et al., 2012; Ioffe and Szegedy, 2015; Kingma and Ba, 2014; Krizhevsky et al., 2012; Szegedy et al., 2015]. However, deep learning has also suffered from collecting large amount of labeled data which is both cost and time consuming. Thus, it has become important to develop semi-supervised methodologies that learn a classifier (or discriminator) by using small labeled data and large unlabeled data.

Various semi-supervised learning methods have been proposed for deep learning. Weston et al. [2012] employed a manifold embedding technique using the pre-constructed graph of unlabeled data, while Rasmus et al. [2015] used a specially designed auto-encoder to extract essential features for classification. The variational auto encoder [Kingma and Welling, 2013] was also used in the context of semi-supervised learning by maximizing the variational lower bound of both labeled and unlabeled data [Kingma et al., 2014; Maaløe et al., 2016]. Miyato et al. [2017] applied the idea of adversarial training to semi-supervised learning.

Recently, semi-supervised learning based on generative adversarial networks (*GAN*, Goodfellow et al. [2014a]) has received attention. For  $K$ -class classification problems, Salimans et al. [2016] and Kumar et al. [2017] solved the  $(K+1)$ -class classification problem, where the additional  $(K+1)$ th class consists of synthetic images made by a generator of *GAN* learned by unlabeled data. Dai et al. [2017] noticed that not a good generator but a bad generator that generates synthetic images that are much different from observed images is crucial for the success of semi-supervised learning. Dai et al. [2017] provided theoretical justifications of using a bad generator and developed a semi-supervised learning algorithm called *Bad GAN* which showed state-of-the-art performance over multiple benchmark datasets.

However, *Bad GAN* has several limitations. It needs two additional deep architectures besides the one for the classifier: bad sample generator and pre-trained density estimation. Learning

these multiple deep architectures requires a large amount of computations and consumes enormous memory. In particular, *PixelCNN++* [Salimans et al., 2017] is used for the pre-trained density estimation, which needs very large computational resources. Another difficulty in *Bad GAN* is that it requires a two-step learning procedure, the first step is to learn the *PixelCNN++* model and the second step is to learn the classifier and the bad generator.

In this thesis, we propose a new semi-supervised learning method which competes well with other state-of-the-art semi-supervised learning algorithms while needing much less amount of computational resources. In particular, the proposed method employs only one deep architecture, and hence, the corresponding learning phase is much easier and faster.

One of the standard assumptions for semi-supervised learning is that the data tend to form discrete clusters, and points in the same cluster are more likely to share a label, which is called *cluster assumption* [Chapelle et al., 2009]. The proposed method is motivated by close investigation of this assumption. *Cluster assumption* implies that the optimal decision boundary is located in the low-density regions, that is, if the neighborhood of a given decision boundary includes many unlabeled data points, the decision boundary would not be optimal. Thus, the classifier needs to be trained so that the neighborhood of the decision boundary and unlabeled data overlap as little as possible. We devised a specially designed regularization term to achieve this goal. Its theoretical justification is given in Section 2.3. To measure the de-

gree of overlap between the neighborhood of the decision boundary and unlabeled data, we developed an algorithm to generate artificial samples near the decision boundary based on the idea of the adversarial training, which finds the direction for a given datum to which the probabilities of each class change most [Goodfellow et al., 2014b; Miyato et al., 2015, 2017]. Note that only a deep architecture for classification is needed to generate artificial data, and thus, the corresponding learning procedure is cheaper, easier, and faster. We call the proposed method *GAB* (Generating Artificial data on the decision Boundary).

By analyzing multiple benchmark datasets, we show that *GAB* competes well with the state-of-the-art algorithms with much fewer computations. Especially, for MNIST, the test accuracy of *GAB* is similar to those of *Bad GAN* [Dai et al., 2017] and *VAT* [Miyato et al., 2017] with 5 times and 7 times fewer training epochs, respectively.

In Section 2.2, we review the *Bad GAN* method briefly. In Section 2.3, the main ideas, theoretical analysis, and objective function of *GAB* are given, and a technique to generate artificial samples near a given decision boundary using the adversarial training is described in Section 2.4. A regularization term which is indispensable for *GAB* is explained in Section 2.5, and results of various numerical experiments are presented in Section 2.6.

## 2.2 Review of existing semi-supervised learning methods in deep learning

### 2.2.1 *DGN* [Kingma et al., 2014]

Kingma et al. [2014] devised a semi-supervised learning method, called *DGN* (deep generative networks), with a generative model employing the *VAE* (variational auto encoder) method [Kingma and Welling, 2013]. *DGN* first introduced how variational inference can be brought to bear upon the problem of semi-supervised classification. To be specific, the loss functions for a labeled datum  $(\mathbf{x}, y)$  and an unlabeled datum  $\mathbf{x}$  can be induced by the variational inference respectively given as

$$\begin{aligned} \log p(\mathbf{x}, y; \theta) &\geq \int \log \left[ \frac{p(\mathbf{x}|y, \mathbf{z}; \theta) \cdot p(y; \theta) \cdot p(\mathbf{z}; \theta)}{q(\mathbf{z}|\mathbf{x}, y; \phi)} \right] \cdot q(\mathbf{z}|\mathbf{x}, y; \phi) d\mathbf{z} \\ &=: -\mathcal{V}(\mathbf{x}, y) \end{aligned}$$

and

$$\begin{aligned} \log p(\mathbf{x}; \theta) &\geq \sum_y \int \log \left[ \frac{p(\mathbf{x}|y, \mathbf{z}; \theta) \cdot p(y; \theta) \cdot p(\mathbf{z}; \theta)}{q(\mathbf{z}|\mathbf{x}, y; \phi)} \right] \cdot q(y, \mathbf{z}|\mathbf{x}; \phi) d\mathbf{z} \\ &= \sum_y q(y|\mathbf{x}; \phi) \cdot (-\mathcal{L}(\mathbf{x}, y) + Ent(q(y|\mathbf{x}; \phi))) =: -\mathcal{W}(\mathbf{x}), \end{aligned}$$

where  $q(y|\mathbf{x}; \phi)$  is the variational distribution parametrized by  $\phi$ , and  $Ent(q(y|\mathbf{x}; \phi))$  is the entropy of  $q(y|\mathbf{x}; \phi)$ . The final loss function is formulated as

$$\mathbb{E}_{\mathbf{x}, y \sim \mathcal{L}^{tr}} \mathcal{V}(\mathbf{x}, y) + \mathbb{E}_{\mathbf{x} \sim \mathcal{U}^{tr}} \mathcal{W}(\mathbf{x}),$$

where  $\mathcal{L}^{tr}$  and  $\mathcal{U}^{tr}$  refer to labeled and unlabeled data, respectively.

Finally, *DGN* uses the conditional distribution  $q(y|\mathbf{x};\phi)$  as the classifier.

### 2.2.2 *Ladder network* [Rasmus et al., 2015]

Rasmus et al. [2015] proposed a learning technique for semi-supervised learning, called *Ladder network*, which utilizes an auto encoder. The goal of *adder network* is to learn feature vectors of a deep architecture so that they have condensed information for reconstructing as well as classifying original images. To do so, *ladder network* combines a discriminator for supervised learning and an auto encoder for unsupervised learning by sharing the feature vectors of each layer.

*Ladder network* devises a loss function that minimizes the supervised cost function for labeled data and the unsupervised cost function for unlabeled data simultaneously.  $p(y|\mathbf{x};\theta)$  is the conditional distribution with a deep architecture parametrized by  $\theta$ , and the highest feature vector of  $p(y|\mathbf{x};\theta)$  is denoted by  $f(\mathbf{x};\theta)$ . In addition,  $g(\mathbf{z};\phi)$  is defined as a decoder with a deep architecture parametrized by  $\phi$ . Then, the simplest cost function of *ladder network* is the sum of the negative cross-entropy function for labeled data and the reconstruction error function for unlabeled data, which is given as

$$-\mathbb{E}_{\mathbf{x},y\sim\mathcal{L}^{tr}} [\log p(y|\mathbf{x};\theta)] + \mathbb{E}_{\mathbf{x}\sim\mathcal{U}^{tr}} \|\mathbf{x} - g(f(\mathbf{x};\theta);\phi)\|^2,$$

where  $\mathcal{L}^{tr}$  and  $\mathcal{U}^{tr}$  are labeled and unlabeled data, respectively.

### 2.2.3 VAT [Miyato et al., 2017]

Recently, a simple but powerful idea for semi-supervised learning was proposed called *VAT* (virtual adversarial training, [Miyato et al., 2017]). *VAT* is a regularization method that is inspired by the adversarial training method [Goodfellow et al., 2014b] for supervised learning, which enhances the invariance of the classifier with respect to perturbations of input data.

Let  $p(y|\mathbf{x}; \theta)$  be a discriminator parametrized by  $\theta$ . Because it is intractable to consider all possible perturbations for a given datum  $\mathbf{x}$ , *VAT* proposes a representative direction for all perturbations called adversarial direction which is given as

$$\mathbf{r}_{\text{advr}}(\mathbf{x}, \epsilon) = \underset{\mathbf{r}; \|\mathbf{r}\| \leq \epsilon}{\operatorname{argmax}} D_{\text{KL}}(p(\cdot|\mathbf{x}; \theta) || p(\cdot|\mathbf{x} + \mathbf{r}; \theta)), \quad (2.1)$$

where  $D_{\text{KL}}$  is the Kullback-Leibler divergence and  $\epsilon > 0$  is a tuning parameter. In fact, the adversarial direction is the direction in which the conditional probabilities of each class change most. Then, the regularization term of *VAT* is given as

$$L^{\text{VAT}}(\theta; \hat{\theta}, \mathbf{x}, \epsilon) = D_{\text{KL}}(p(\cdot|\mathbf{x}; \theta^{\text{curr}}) || p(\cdot|\mathbf{x} + \mathbf{r}_{\text{advr}}(\mathbf{x}, \epsilon); \theta))$$

where  $\theta^{\text{curr}}$  is the current estimate of  $\theta$ . Combining with the cross-entropy term for labeled data, we get the final objective function of *VAT*:

$$-\mathbb{E}_{\mathbf{x}, y \sim \mathcal{L}^{\text{tr}}} [\log p(y|\mathbf{x}; \theta)] + \mathbb{E}_{\mathbf{x} \sim \mathcal{U}^{\text{tr}}} [L^{\text{VAT}}(\theta; \theta^{\text{curr}}, \mathbf{x}, \epsilon)], \quad (2.2)$$

where  $\mathcal{L}^{\text{tr}}$  and  $\mathcal{U}^{\text{tr}}$  refer to labeled and unlabeled data, respectively.

### 2.2.4 *Bad GAN* [Dai et al., 2017]

Dai et al. [2017] introduced a semi-supervised learning method called *Bad GAN* that trains a good discriminator with a bad generator. This procedure trains a generator as well as a discriminator simultaneously. Let  $\mathcal{D}_G(\phi)$  be the generated bad samples with a bad generator  $p_G(\cdot; \phi)$  parametrized by  $\phi$ . Here, the ‘bad generator’ is a deep architecture that generates samples different from observed data. Let  $p^{\text{pt}}(\cdot)$  be a pre-trained density estimation model. For a given discriminator with a feature vector  $v(x; \theta)$  of a given input  $x$  parameterized by  $\theta$ , *Bad GAN* learns the bad generator by minimizing the following:

$$\begin{aligned} & \mathbb{E}_{x \sim \mathcal{D}_G(\phi)} [\log p^{\text{pt}}(x) \mathbb{I}(p^{\text{pt}}(x) > \tau)] \\ & + \|\mathbb{E}_{x \sim \mathcal{U}^{\text{tr}}} v(x; \hat{\theta}) - \mathbb{E}_{x \sim \mathcal{D}_G(\phi)} v(x; \hat{\theta})\|^2 \end{aligned}$$

with respect to  $\phi$ , where  $\tau > 0$  is a tuning parameter,  $\mathcal{U}^{\text{tr}}$  is the unlabeled data, and  $\hat{\theta}$  is the current estimate of  $\theta$  and  $\|\cdot\|$  is the Euclidean norm.

In turn, to train the discriminator, we consider the  $K$ -class classification problem as the  $(K + 1)$ -class classification problem where the  $(K + 1)$ -th class is an artificial label of the bad samples generated by the bad generator. We estimate the parameter  $\theta$  in

the discriminator by minimizing the following:

$$\begin{aligned}
& -\mathbb{E}_{x,y\sim\mathcal{L}^{tr}} [\log p(y|x, y \leq K; \theta)] - \mathbb{E}_{x\sim\mathcal{U}^{tr}} \left[ \log \left\{ \sum_{k=1}^K p(k|x; \theta) \right\} \right] \\
& -\mathbb{E}_{x\sim\mathcal{D}_G(\phi)} [\log p(K+1|x; \theta)] - \mathbb{E}_{x\sim\mathcal{U}^{tr}} \left[ \sum_{k=1}^K p(k|x; \theta) \log p(k|x; \theta) \right]
\end{aligned} \tag{2.3}$$

for given  $\phi$ , where  $\mathcal{L}^{tr}$  is the labeled set. The second and the third terms in (2.3) are the cross-entropies between the unlabeled samples and the bad samples. The fourth term is similar to the entropy of the unlabeled data which is usually helpful for semi-supervised learning [Grandvalet and Bengio, 2005]. See Dai et al. [2017] for details of the objective function (2.3).

## 2.3 Proposed Method

### 2.3.1 Motivation

Let  $X \in \mathcal{X}$  and  $Y \in \{1, \dots, K\}$  be input and output random variables, where  $\mathcal{X}$  is a compact subset of  $\mathbb{R}^D$ . For a given function (i.e. model)  $f : \mathbb{R}^D \rightarrow \mathbb{R}^K$ , let  $C(\mathbf{x}; f)$  be the corresponding classifier given as  $C(\mathbf{x}; f) = \operatorname{argmax}_k f_k(\mathbf{x})$ , where  $f_k$  is the  $k$ -th element of  $f$ .

Let  $(X_1, Y_1), \dots, (X_n, Y_n)$  be  $n$  labeled data which are assumed to be random samples of  $(X, Y)$ . A standard supervised learning searches a model  $f$  that minimizes the empirical risk

$\sum_{i=1}^n I(C(X_i; f) \neq Y_i)$  over a given set of models  $\mathcal{F}$ . When the number of labeled data is small, the minimizer of the empirical

risk may not be unique and overfits frequently; thus, selecting the optimal one is difficult. However, under the *cluster assumption*, unlabeled data are helpful to find the optimal decision boundary.

To be more specific, let us define

$$D(f) := \bigcup_{k=1}^K \left\{ \mathbf{x} \in \mathcal{X} : f_k(\mathbf{x}) - \max_{k' \neq k} f_{k'}(\mathbf{x}) = 0 \right\}.$$

Ideally, *Bad GAN* searches a model that minimizes the empirical risk and at the same time minimizes  $\max_{\mathbf{x} \in D(f)} p(\mathbf{x})$ , where  $p(\mathbf{x})$  is the true density of  $X$ . To materialize this idea, however, we need to know the marginal density of  $X$  (or the generative model for  $X$ ), which is difficult and computationally demanding.

The main idea of *GAB* is that it considers candidate models  $f$  such that  $\mathcal{X}(f, \xi)$  is least overlapped with unlabeled data, where  $\mathcal{X}(f, \xi) = \{\mathbf{x} \in \mathcal{X} : d(D(f), \mathbf{x}) < \xi\}$  for given  $\xi > 0$ . Here,  $d(\cdot, \cdot)$  is a given metric and  $d(D(f), \mathbf{x}) = \min_{\mathbf{x}' \in D(f)} d(\mathbf{x}', \mathbf{x})$ . Then, *GAB* searches a model  $f$  that minimizes the empirical risk as well as the degree of overlap between  $\mathcal{X}(f, \xi)$  and unlabeled data. A technical bottleneck is the characterization of  $\mathcal{X}(f, \xi)$  for a given  $f$ , because  $f$  is highly nonlinear, and we resolve this problem by generating artificial samples on  $\mathcal{X}(f, \xi)$ . A novel feature is that *GAB* does not require the generative model of  $X$ , but it only requires generating artificial samples on  $\mathcal{X}(f, \xi)$ , which can be done effectively by use of only a given  $f$ . See Section 2.4 for details. In the next subsection, we theoretically show that *GAB* finds the optimal classifier under regularity conditions.

### 2.3.2 Theoretical Analysis

Let  $p_k^0(\mathbf{x})$  be the conditional density of  $X$  given  $Y = k$  and let  $\pi_k = \Pr(Y = k)$ . We assume that  $p_k(\mathbf{x})$  are continuous and positive on  $\mathcal{X}$ . The marginal density of  $X$  is given as  $p^0(\mathbf{x}) := \sum_{k=1}^K \pi_k p_k^0(\mathbf{x})$ . Let  $P_k^0(\mathbf{x}) := \Pr(Y = k | X = \mathbf{x})$  and  $D^{Bayes}$  be the decision boundary of the Bayes classifier  $\mathcal{C}^{Bayes}(\mathbf{x}) = \operatorname{argmax}_k P_k^0(\mathbf{x})$  :

$$D^{Bayes} := \bigcup_{k=1}^K \left\{ \mathbf{x} \in \mathcal{X} : P_k^0(\mathbf{x}) - \max_{k' \neq k} P_{k'}^0(\mathbf{x}) = 0 \right\}.$$

For given  $\epsilon > 0$ , define

$$\mathcal{X}_k(\epsilon) := \left\{ \mathbf{x} \in \mathcal{X} : P_k^0(\mathbf{x}) - \max_{k' \neq k} P_{k'}^0(\mathbf{x}) > \epsilon \right\},$$

$\mathcal{X}(\epsilon) := \cup_{k=1}^K \mathcal{X}_k(\epsilon)$ ,  $\delta(\epsilon) := \Pr(X \in \mathcal{X} - \mathcal{X}(\epsilon))$  and  $\rho(\epsilon) := d(\mathcal{X}(\epsilon), D^{Bayes})$ . For each  $\mathcal{X}_k(\epsilon)$ , we assume that there exists a partition  $\{\mathcal{X}_{kj}(\epsilon)\}_{j=1}^{m_k}$  such that  $\mathcal{X}_{kj}(\epsilon)$ s are open sets,  $\mathcal{X}_k(\epsilon) = \cup_{j=1}^{m_k} \mathcal{X}_{kj}(\epsilon)$  and  $\min_{j \neq j'} d(\mathcal{X}_{kj}(\epsilon), \mathcal{X}_{kj'}(\epsilon)) > 0$ . We denote the probability measure of truncated random variable  $X$  on  $\mathcal{X}(\epsilon)$  as  $P_\epsilon^0$ , whose density function is given as  $p_\epsilon^0(\mathbf{x}) \propto p^0(\mathbf{x}) \cdot I(\mathbf{x} \in \mathcal{X}(\epsilon))$ .

Let us consider a set  $\mathcal{F}$  of models such that all  $f \in \mathcal{F}$  are continuous and  $f^0 \in \mathcal{F}$ , where

$$f^0(\mathbf{x}) := \left( P_k^0(\mathbf{x}) - \max_{k' \neq k} P_{k'}^0(\mathbf{x}), k = 1, \dots, K \right).$$

For a given  $f \in \mathcal{F}$  and a positive constant  $\xi$ , let  $P_{f,\xi}$  be a probability measure on  $\mathcal{X}$  such that it has a density  $p_{f,\xi}$  with the support  $\mathcal{X}(f, \xi) := \cup_{\mathbf{x} \in D(f)} \mathcal{B}(\mathbf{x}, \xi) \cap \mathcal{X}$ , where  $\mathcal{B}(\mathbf{x}, \xi) = \{\mathbf{v} : d(\mathbf{x}, \mathbf{v}) < \xi\}$ . We assume that  $p_{f,\xi}(\mathbf{x})$  is bounded below by some positive constant.

Note that the Bayes decision boundary  $D^{Bayes}$  is located at  $\mathcal{X} - \mathcal{X}(\epsilon)$ . The following condition is essentially the *cluster assumption*.

**Condition 1.**  $\delta(\epsilon)$  is small enough to satisfy  $\delta(\epsilon) < \epsilon \cdot \min_{k,j} \Pr(X \in \mathcal{X}_{kj}(\epsilon))$ .

**Theorem 2.3.1.** *Define*

$$u_{\epsilon,\xi}(f) := \mathbb{E}_{X \sim P_\epsilon^0} \left[ I \left( \max_k f_k(X) > \epsilon \right) \right] \\ + \mathbb{E}_{X \sim P_{f,\xi}} \left[ I \left( \max_k f_k(X) < \epsilon \right) \right].$$

For  $\epsilon > 0$  and  $0 < \xi \leq \rho(\epsilon)$ , consider a subset  $\tilde{\mathcal{F}}(\epsilon, \xi)$  of  $\mathcal{F}$  given as

$$\tilde{\mathcal{F}}(\epsilon, \xi) := \left\{ f \in \mathcal{F} : f \in \operatorname{argmax}_{f \in \mathcal{F}} u_{\epsilon,\xi}(f) \right\}$$

Let  $\hat{f}$  be a maximizer of  $l_n(f) := \frac{1}{n} \sum_{i=1}^n I(C(X_i; f) = Y_i)$  on  $\tilde{\mathcal{F}}(\epsilon, \xi)$ . Then under Condition 1, we have

$$P^{(n)} \left\{ \Pr \left\{ C(X; \hat{f}) = C^{Bayes}(X) \right\} \geq 1 - \delta(\epsilon) \right\} \\ \geq 1 - Km \cdot \exp(-nc_*^2/2), \quad (2.4)$$

for some constant  $c_*$  not depending on  $n$ , where  $m = \sum_{k=1}^K m_k$  and  $P^{(n)}$  is the product probability measure of  $(X_1, Y_1), \dots, (X_n, Y_n)$ .

The proof is shown in Appendix A. Note that  $I(\max_k f_k(X) > \epsilon)$  and  $I(\max_k f_k(X) < \epsilon)$  cannot be simultaneously 1, and hence,  $u_{\epsilon,\xi}(f)$  becomes larger when the overlap of the supports of  $P_\epsilon^0$  and

$P_{f,\xi}$  is smaller. The inequality (2.4) means that  $C(\mathbf{x}; \hat{f})$  is equal to  $C^{Bayes}(\mathbf{x})$  for most of  $\mathcal{X}$  with the probability converging to 1 exponentially fast with the number of labeled data.

### 2.3.3 Objective Function of *GAB*

Based on the results of Theorem 2.3.1, we propose an objective function for *GAB*, which is the sum of surrogated versions of  $l_n$  and  $u_{\epsilon,\xi}$ . Let  $g(\mathbf{x}; \theta) \in \mathbb{R}^K$  be the pre-softmax vector of a given deep architecture parametrized by  $\theta$ . Then, the proposed objective function for *GAB* is given as

$$\begin{aligned} & -\mathbb{E}_{\mathbf{x},y \sim \mathcal{L}^{tr}} [\log p(y|\mathbf{x}; \theta)] + \mathbb{E}_{\mathbf{x} \sim \mathcal{U}^{tr}} [L^{\text{Unl}}(\theta; \mathbf{x})] \\ & + \mathbb{E}_{\mathbf{x} \sim \mathcal{D}(\theta^{\text{curr}})} [L^{\text{Art}}(\theta; \mathbf{x})] + \mathbb{E}_{\mathbf{x} \sim \mathcal{U}^{tr}} [L^{\text{Reg}}(\theta; \mathbf{x})], \end{aligned} \quad (2.5)$$

where  $p(k|\mathbf{x}; \theta) = \exp(g_k(x; \theta)) / \sum_{k'=1}^K \exp(g_{k'}(x; \theta))$ ,  $\mathcal{L}^{tr}$  and  $\mathcal{U}^{tr}$  are the sets of labeled and unlabeled samples respectively,  $\mathcal{D}(\theta^{\text{curr}})$  is the set of generated artificial samples on  $\mathcal{X}(g(\cdot; \theta^{\text{curr}}), \xi)$ ,

$$\begin{aligned} L^{\text{Unl}}(\theta; \mathbf{x}) &= - \sum_{k=1}^K \left[ \frac{\exp(g_k(\mathbf{x}; \theta))}{\sum_{k'=1}^K \exp(g_{k'}(\mathbf{x}; \theta))} \right. \\ & \quad \left. \times \log \frac{\exp(g_k(\mathbf{x}; \theta))}{\sum_{k'=1}^K \exp(g_{k'}(\mathbf{x}; \theta))} \right], \\ L^{\text{Art}}(\theta; \mathbf{x}) &= - \log \frac{1}{1 + \sum_{k=1}^K \exp(g_k(\mathbf{x}; \theta))}, \end{aligned}$$

and  $L^{\text{Reg}}(\theta; \mathbf{x})$  is an additional regularization term that is explained in Section 5.

The sum of the first three terms of (2.5) is a surrogated version of  $-(l_n + u_{\epsilon,\xi})$  in Theorem 2.3.1. To be more concrete, we use cross-entropy with the labeled samples, which is a well-known surrogate

function of the 0-1 loss. In contrast,  $L^{\text{Unl}}$  forces the maximum value of  $g_k(\mathbf{x}; \theta)$ s to be large for the unlabeled samples, and  $L^{\text{Art}}$  encourages the maximum value of  $g_k(\mathbf{x}; \theta)$ s to be smaller than 0 for the artificial samples. Thus, by minimizing  $L^{\text{Unl}} + L^{\text{Art}}$ , we have the function  $g$  such that  $\max_k g_k$  is large on unlabeled data and  $\min_k g_k$  is small on artificial data. Note that a function maximizing  $u_{\epsilon, \xi}$  also has such properties.

For  $\mathcal{D}(\theta^{\text{curr}})$ , we propose a novel method to generate artificial samples on  $\mathcal{X}(g(\cdot; \theta^{\text{curr}}), \xi)$  in Section 2.4.

## 2.4 Generation of artificial samples

### 2.4.1 Adversarial training

Adversarial training is used to train the model being less sensitive to data perturbation toward the adversarial direction [Goodfellow et al., 2014b; Miyato et al., 2015, 2017]. Here, the adversarial direction for a given datum is the direction to which the probabilities of each class change the most:

$$\mathbf{r}_{\text{advr}}(\mathbf{x}, c) = \underset{\mathbf{r}; \|\mathbf{r}\| \leq c}{\operatorname{argmax}} D_{\text{KL}}(p(\cdot | \mathbf{x}; \theta^{\text{curr}}) || p(\cdot | \mathbf{x} + \mathbf{r}; \theta^{\text{curr}})), \quad (2.6)$$

where  $c > 0$  is the maximum radius of perturbation,  $p(\cdot | \mathbf{x}; \theta)$  is a classifier parametrized by  $\theta$ , and  $\theta^{\text{curr}}$  is the current estimate of  $\theta$ . We adopt the idea of the adversarial direction to generate artificial data near the current decision boundary.

## 2.4.2 Motivation

In this subsection, we explain how the adversarial direction can be used to generate samples near the current decision boundary. For simplicity, we only consider the linear decision boundary. For the decision boundary made by the DNN model with ReLU-like activation function, see Appendix A.

Let us consider the 2-class linear logistic regression model parametrized by  $\eta = \{w, b\}$ , that is,

$$p(y = 1|\mathbf{x}; \eta) = \left(1 + \exp(-b - w' \mathbf{x})\right)^{-1}.$$

Note that the decision boundary is  $\{\mathbf{x} : b + w' \mathbf{x} = 0\}$ , and for any given  $\mathbf{x}$ , the distance between  $\mathbf{x}$  and the decision boundary is  $|b + w' \mathbf{x}|/\|w\|$ . The key result is that moving  $\mathbf{x}$  toward the adversarial direction,  $r_{\text{advr}}(\mathbf{x}, c)$  is equivalent to moving  $\mathbf{x}$  toward the decision boundary which is stated in the following proposition. The proof is provided in Appendix A.

**Proposition 2.4.1.** *For a sufficiently small  $c > 0$ , we have*

$$\text{sign}(w' \mathbf{x} + b) \cdot \text{sign}\left(w' r_{\text{advr}}(\mathbf{x}, c)\right) = -1.$$

Proposition 2.4.1 implies that

$$\frac{|b + w' \mathbf{x}|}{\|w\|} > \frac{|b + w'(\mathbf{x} + r_{\text{advr}}(\mathbf{x}, c))|}{\|w\|}$$

unless  $|b + w' \mathbf{x}| = 0$ . Hence, we can treat

$$\mathbf{x} + C \frac{r_{\text{advr}}(\mathbf{x}, c)}{\|r_{\text{advr}}(\mathbf{x}, c)\|}$$

for appropriately choosing  $C > 0$  as a sample closer to the decision boundary.

### 2.4.3 Artificial sample generation for general classifier

Motivated by Proposition 2.4.1, we propose an artificial sample generator as follows. Let  $C > 0$  be fixed and  $\hat{\theta}$  be the current estimate of  $\theta$ . For a given input  $\mathbf{x}$  and a classifier  $p(\cdot|\mathbf{x};\hat{\theta})$ , we calculate the adversarial direction  $\mathbf{r}_{\text{advr}}(\mathbf{x}, c)$  for a given  $c$  by (2.6). Then, we consider

$$\mathbf{x}^A := \mathbf{x} + C \frac{\mathbf{r}_{\text{advr}}(\mathbf{x}, c)}{\|\mathbf{r}_{\text{advr}}(\mathbf{x}, c)\|}$$

as an artificial datum close to the decision boundary. We generate artificial samples for all unlabeled data. In practice, we apply the same  $C$  to all unlabeled data and choose  $C$  based on the validation data accuracy. In Figure 2.1, we illustrate how the artificial samples generated by the proposed method are distributed for the multi-class problem. Using the perfect classifier, we can clearly see that most generated samples are located well near the decision boundary.

It may happen that a generated sample is not sufficiently close to the decision boundary to be a 'good' artificial sample, in particular when  $C$  is too large or too small. To avoid such a situation, we exclude  $\mathbf{x}^A$ , which satisfies the following condition:

$$\max_k p(k|\mathbf{x}^A; \theta^{\text{curr}}) > 1 - \alpha$$

for a pre-specified  $\alpha > 0$ . In the experimental analysis, we set the optimal  $\alpha$  with the validation data.

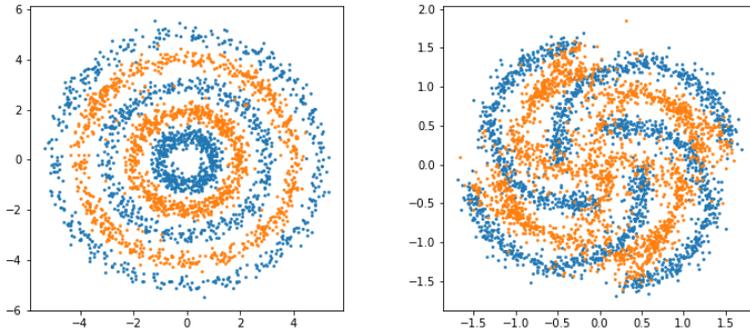


Figure 2.1: Demonstration of how the artificial samples generated by the adversarial training are distributed. We consider two cases: 3-class classification problem (**Left**) and 4-class classification problem (**Right**). True data and artificial data are colored blue and orange, respectively.

#### 2.4.4 Finding adversarial direction

Miyato et al. [2017] proposed the fast approximation method to calculate the adversarial direction  $\mathbf{r}_{\text{advr}}(\mathbf{x}, c)$  using the second-order Taylor expansion. Let us define

$$H(\mathbf{x}, \hat{\theta}) = \nabla \nabla D_{\text{KL}}(p(\cdot | \mathbf{x}; \theta^{\text{curr}}) || p(\cdot | \mathbf{x} + \mathbf{r}; \theta^{\text{curr}})) |_{\mathbf{r}=\mathbf{0}}.$$

They stated that  $\mathbf{r}_{\text{advr}}$  emerges as the first dominant eigenvector  $\mathbf{v}(\mathbf{x}, \hat{\theta})$  of  $H(\mathbf{x}, \hat{\theta})$  with magnitude  $c$ . However, there always exist two dominant eigenvectors,  $\pm \mathbf{v}(\mathbf{x}, \hat{\theta})$ , and the sign should be selected carefully. Therefore, we slightly modify the approximation

method of Miyato et al. [2017] by

$$\mathbf{r}_{\text{adv}}(\mathbf{x}, c) = \underset{\mathbf{r} \in \{\pm \mathbf{v}(\mathbf{x}, \theta^{\text{curr}})\}}{\text{argmax}} D_{\text{KL}}(p(\cdot|\mathbf{x}; \theta^{\text{curr}}) || p(\cdot|\mathbf{x} + \mathbf{r}; \theta^{\text{curr}})).$$

## 2.5 Regularization term

For generated samples by adversarial training to be ‘good’ artificial samples, the adversarial directions should be toward the decision boundary. While this always happens for the linear model by the Proposition 2.4.1, adversarial directions could be opposite to the decision boundary for deep models that are highly nonlinear. To avoid such undesirable cases as much as possible, it would be helpful to smoothen the deep model with a regularization term. In this thesis, we used the regularization term used in the *VAT* [Miyato et al., 2017] method to smoothen the deep model:

$$L^{\text{Reg}}(\theta; \mathbf{x}) = D_{\text{KL}}(p(\cdot|\mathbf{x}; \theta^{\text{curr}}) || p(\cdot|\mathbf{x} + r_{\text{adv}}(\mathbf{x}, c); \theta)). \quad (2.7)$$

By doing so, we can save computing time because the adversarial directions are reused.

The adversarial direction obtained by maximizing the KL divergence is sensitive to local fluctuations of class probabilities which is exemplified in Figure 2.2. The regularization term (2.7) is helpful to find an adversarial direction which is toward the decision boundary by eliminating unnecessary local fluctuations of class probabilities. In Figure 2.3, we compared artificial samples generated with and without the regularization term (2.7) for the MNIST dataset. While the artificial samples generated without

Table 2.1: Test accuracies for *GAB* with and without the regularization term. The numbers in the parenthesis are the sizes of labeled data.

Data	MNIST (100)	SVHN (1000)	CIFAR10 (4000)
w/ $L^{\text{Reg}}$	98.89	95.94	85.31
w/o $L^{\text{Reg}}$	83.6	90.21	68.32

the regularization term (2.7) are visually similar to the given input digits, those generated with the regularization term (2.7) look like mixtures of two different digits, which means that the artificial data with the regularization term are located near the decision boundary.

A better performance of generating ‘good’ artificial samples results in better prediction accuracies. Table 2.1 compares the prediction accuracies of *GAB* with and without the regularization term, which clearly shows that the regularization is necessary for *GAB*.

## 2.6 Experiments

### 2.6.1 Prediction performance comparisons

We compare the prediction performances of *GAB* over the benchmark datasets with other semi-supervised learning algorithms. We

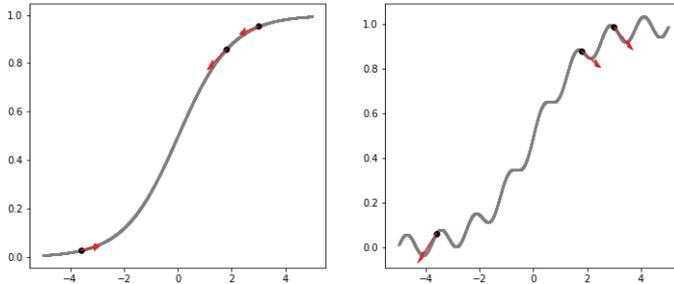


Figure 2.2: Examples of  $P(y = 1|x)$  of smooth (**Left**) and wiggly (**Right**) cases. We plotted 3 points and their adversarial directions for each case.

considered the most widely used datasets: MNIST [LeCun et al., 1998], SVHN [Marlin et al., 2010], CIFAR10 [Krizhevsky and Hinton, 2009], and CIFAR100 [Krizhevsky and Hinton, 2009]. For fair comparison, we used the same architectures as those used in Miyato et al. [2017] for MNIST, SVHN, and CIFAR10. See Appendix A for details. The optimal tuning parameters  $(c, C, \alpha)$  in *GAB* are chosen based on the validation data accuracy. We used the *Adam* algorithm [Kingma and Ba, 2014] to update the parameters and did not use any data augmentation techniques. The results are summarized in Table 2.2, which shows that *GAB* achieves the state-of-the-art accuracies for MNIST (20) and SVHN (500, 1000) and competitive accuracies with the state-of-the-art methods for other settings. For CIFAR100, which is more complex, *GAB* also outperforms *VAT*. Here, we used the large model described in Appendix A. Because *GAB* only needs one deep architecture, we can



Figure 2.3: **(Upper)** 10 randomly sampled original MNIST dataset. **(Middle and Lower)** Artificial samples obtained by classifiers learned with and without the regularization term of *VAT*.

conclude that *GAB* is a powerful and computationally efficient method.

Another advantage of *GAB* is its stability with respect to the learning phase. With small labeled data, Figure 2.4 shows that the test accuracies of each epoch tend to fluctuate and are degraded for *VAT* and *Bad GAN*, while *GAB* provides a much more stable result. This may be partly because the artificial samples help to stabilize the objective function.

### 2.6.2 Effects of Tuning Parameters

*GAB* introduces three tuning parameters  $c$ ,  $C$ , and  $\alpha$ , where  $c$  is the constant used to find the adversarial direction,  $C$  is the radius to generate artificial samples, and  $\alpha$  is used to determine whether an artificial sample is 'good'. We investigated the sensitivities of prediction performances with respect to the changes of the val-

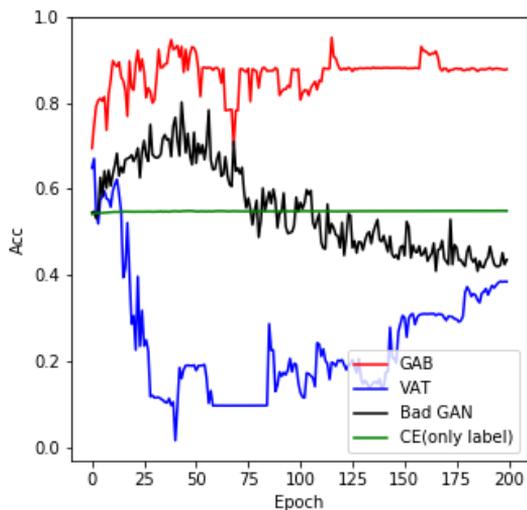


Figure 2.4: Trace plot of the test accuracies with the three semi-supervised learning methods and the supervised learning method with cross-entropy for MNIST with 20 labeled data.

ues of these tuning parameters. When we vary one of the tuning parameters, the other parameters are fixed at the optimal values chosen by the validation data. The results are reported in Table 2.3. Unless  $\alpha$  is too small or too large, the prediction performances do not change much. For  $c$  and  $C$ , care should be taken. When  $c$  is larger than the optimal  $C$  (i.e. 2) or when  $C$  is smaller than the optimal  $c$  (i.e. 1.5), the prediction performances are suboptimal. Apparently, choosing  $c$  and  $C$  with  $c$  being slightly smaller than  $C$  gives the best result.

### 2.6.3 Computational Efficiency

We investigated the computational efficiency of *GAB* in view of learning speed and computation time per training epoch. For *Bad GAN*, we did not use *PixelCNN++* on SVHN and CIFAR10 datasets because the pre-trained *PixelCNN++* models are not publicly available. Without *PixelCNN++*, *Bad GAN* is similar to *FM-GAN* [Salimans et al., 2016]. Figure 2.6 shows bar plots for the numbers of epochs needed to achieve the pre-specified test accuracies. We can clearly see that *GAB* requires much less epochs.

We also calculated the ratios of the computing time of each semi-supervised learning algorithm over the computing time of the corresponding supervised learning algorithm for the CIFAR10 dataset, whose results are summarized in Table 2.4. These ratios are almost same for different datasets. The computation time of *GAB* is less than *Bad GAN* and competitive to *VAT*. From the results of Figure 2.6 and Table 2.4, we can conclude that *GAB* achieves the pre-specified performances efficiently. Note that the learning time of *PixelCNN++* is not considered for this experiment, so comparison of computing time of *GAB* and *Bad GAN* with *PixelCNN++* is meaningless.

### 2.6.4 Quality of Artificial Samples

We investigated how ‘good’ artificial samples generated by *GAB* are. The left two plots of Figure 2.5 show the scatter plot of the synthetic data and the trace plot of prediction accuracies of *GAB*

and *VAT*. The right four plots of Figure 2.5 show the scatter plots with generated artificial samples at various epochs. We can clearly see that artificial samples are distributed near the current decision boundary. We also compared artificial images generated by *GAB* and bad images generated by *Bad GAN* for the MNIST data at the end of the learning procedure. In Figure 2.7, the images by *GAB* do not look like real images and do not seem to be collapsed, which indicates that *GAB* consistently generates diverse and good artificial samples. *Bad GAN* also generates diverse bad samples but some ‘realistic’ images can be found.

Table 2.2: Comparison of prediction accuracies of various semi-supervised learning algorithms for benchmark datasets.  $|\mathcal{L}|$  is the number of labeled data and the results with \* are implemented by us.

Data	Test acc.(%)			
	MNIST		SVHN	
$ \mathcal{L} $	20	100	500	1000
<i>DGN</i> [Kingma et al., 2014]	-	96.67	-	63.98
<i>Ladder</i> [Rasmus et al., 2015]	-	98.94	-	-
<i>FM-GAN</i> [Salimans et al., 2016]	83.23	99.07	81.56	91.89
<i>FM-GAN-Tan</i> [Kumar et al., 2017]	-	-	95.13	95.61
<i>Bad GAN</i> [Dai et al., 2017]	80.16*	99.20	-	95.75
<i>VAT</i> [Miyato et al., 2017]	67.04*	98.64	-	93.17
<i>Tri-GAN</i> [LI et al., 2017]	95.19	99.09	-	94.23
<i>CCLP</i> [Kamnitsas et al., 2018]	-	<b>99.25</b>	-	
<i>GAB</i>	<b>96.32</b>	98.89	<b>95.21</b>	<b>95.94</b>

Data	Test acc.(%)		
	CIFAR10		CIFAR100
$ \mathcal{L} $	1000	4000	8000
<i>Ladder</i> [Rasmus et al., 2015]	-	79.6	-
<i>FM-GAN</i> [Salimans et al., 2016]	78.13	81.37	-
<i>FM-GAN-Tan</i> [Kumar et al., 2017]	80.48	83.80	-
<i>Bad GAN</i> [Dai et al., 2017]	-	<b>85.59</b>	-
<i>VAT</i> [Miyato et al., 2017]	-	85.13	35.89*
<i>Tri-GAN</i> [LI et al., 2017]	-	83.01	-
<i>CCLP</i> [Kamnitsas et al., 2018]	-	81.43	-
<i>GAB</i>	78.44	85.31	<b>36.11</b>

Table 2.3: Test accuracies of MNIST with 100 labeled data for various values of  $c$ ,  $C$ , and  $\alpha$ . The other parameters for each case are fixed at the optimal values.

$c$	1.	1.5	2.	4.
Test acc.	97.94	98.89	98.61	95.65
$C$	1	2.	4.	6.
Test acc.	89.54	98.89	98.79	98.55
$\alpha$	0.001	0.01	0.1	0.2
Test acc.	98.65	98.89	98.77	98.71

Table 2.4: Learning time per training epoch ratio compared to supervised learning with cross-entropy for CIFAR10. *Bad GAN* is operated without *PixelCNN++*.

Method	<i>VAT</i>	<i>GAB</i>	<i>Bad GAN</i>
Time ratio	1.37	2.09	3.20

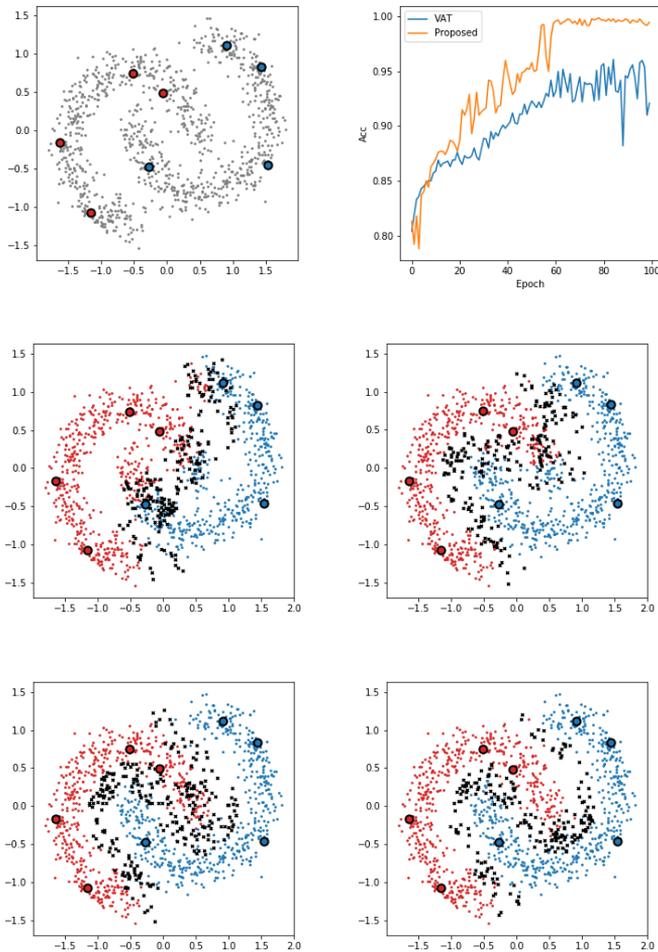


Figure 2.5: **(Upper left)** The scatter plot of synthetic data which consist of 1000 unlabeled data (gray) and 4 labeled data for each class (red and blue with black edge). **(Upper right)** Accuracies of unlabeled data for each epoch for *VAT* and *GAB*. We use 2-layered NN with 100 hidden units each. **(Else)** Artificial samples and classified unlabeled data by colors at the 20,40,60 and 80 training epochs of *GAB*.

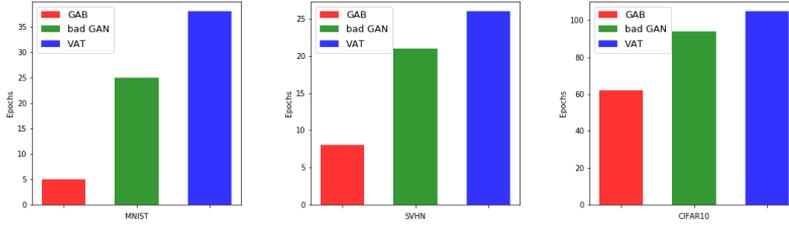


Figure 2.6: The number of epochs to achieve the pre-specified test accuracies (98%, 90% and 80%) with the three methods for **(Left)** MNIST (100), **(Middle)** SVHN (1000) and **(Right)** CIFAR10 (4000) settings. *Bad GAN* is operated without *PixelCNN++* for SVHN and CIFAR10 datasets.

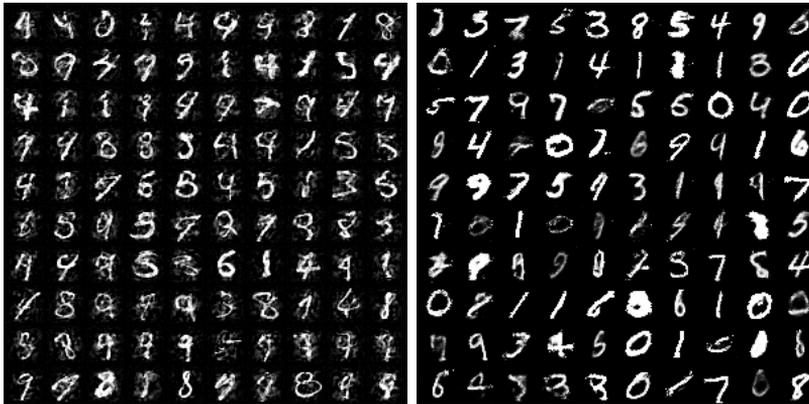


Figure 2.7: 100 randomly sampled artificial images and bad images with **(Left)** *GAB* and **(Right)** *Bad GAN* respectively.

## Chapter 3

# Estimation algorithm of deep generative model

### 3.1 Introduction

Probabilistic generative models with deep neural networks have achieved tremendous success for modeling high dimensional data mainly because of the development of the variational auto encoding framework (*VAE*, Kingma and Welling [2013]; Rezende et al. [2014]). *VAE* models the distribution of an observable random vector  $\mathbf{x}$  of high dimension by introducing a lower dimensional latent vector  $\mathbf{z}$  such that  $p(\mathbf{x}; \theta) = \int_{\mathbf{z}} p(\mathbf{x}|\mathbf{z}; \theta)p(\mathbf{z}; \theta)d\mathbf{z}$ , where  $\theta$  is the parameter of the model. Furthermore, it estimates the parameter by maximizing the lower bound of the marginal log-likelihood, called *ELBO* (evidence lower bound).

To be more specific, the gradient of the marginal log-likelihood

with respect to  $\theta$  is given as

$$\nabla_{\theta} \log p(\mathbf{x}; \theta) = \int p(\mathbf{z}|\mathbf{x}; \theta) \cdot \nabla_{\theta} \log p(\mathbf{x}, \mathbf{z}; \theta) d\mathbf{z}. \quad (3.1)$$

Thus, the central concern to learn the model is to approximate the above integration, which is computationally challenging because  $p(\mathbf{z}|\mathbf{x}; \theta)$  is complicated. *VAE* replaces the model posterior  $p(\mathbf{z}|\mathbf{x}; \theta)$  with a more tractable variational posterior distribution  $q(\mathbf{z}|\mathbf{x}; \phi)$  approximating  $p(\mathbf{z}|\mathbf{x}; \theta)$  and calculates a Monte Carlo estimate of (3.1), which turns out to be equivalent to optimize the *ELBO* function [Kingma and Welling, 2013; Rezende et al., 2014]. To approximate the model posterior more closely, more complex but still tractable families of variational distributions have been proposed to yield a tighter lower bound of the marginal log-likelihood [Burda et al., 2015; Cremer et al., 2017; Kingma et al., 2016a; Rezende and Mohamed, 2015; Salimans et al., 2015; Sønderby et al., 2016]. These approaches have achieved much success in representation tasks as well as the performance task.

Even if the approximation of (3.1) becomes more accurate by using a more complex variational model, the bias caused by the discrepancy between  $p(\mathbf{z}|\mathbf{x}; \theta)$  and  $q(\mathbf{z}|\mathbf{x}; \phi)$  is inevitable, and thus, the resulting estimated model would be sub-optimal. In addition, there is a case where the variational approach is hard to apply. An example is incomplete data whose details are given in Section 4.3. Hence, there is still a need to approximate (3.1) directly without resorting to the variational approach.

An alternative method is to approximate (3.1) directly by uti-

lizing the *MCMC* method [Neal, 1993] to generate samples from the model posterior  $p(\mathbf{z}|\mathbf{x}; \theta)$  [Hoffman, 2017; Salimans et al., 2015; Wolf et al., 2016]. The *MCMC* method builds a Markov chain whose stationary distribution is the model posterior and uses the samples from the Markov chain to approximate (3.1). Although the strength of this approach lies in its accurate approximation of the model posterior, it is quite slow compared with the variational approach, which is one of the inherent drawbacks of *MCMC* based methods, and thus relatively hard to be scaled up to large architectures and/or datasets.

In this thesis, we present a new method based on the *EM* algorithm and importance sampling (*IS*) to learn a deep generative model. The central idea is simple. The *EM* algorithm essentially replaces (3.1) by

$$\int p(\mathbf{z}|\mathbf{x}; \theta^{\text{curr}}) \cdot \nabla_{\theta} \log p(\mathbf{x}, \mathbf{z}; \theta) d\mathbf{z}, \quad (3.2)$$

where  $\theta^{\text{curr}}$  is the current estimate of  $\theta$ . We use *IS* to approximate the integration (3.2). That is, we draw multiple samples  $\mathbf{z}_1, \dots, \mathbf{z}_m$  from a proposal distribution  $q(\mathbf{z})$ , calculate the corresponding weights  $w_1, \dots, w_m$ , and approximate (3.2) as  $\frac{1}{m} \sum_{h=1}^m w_h \cdot \nabla_{\theta} \log p(\mathbf{x}, \mathbf{z}_h; \theta)$ . This naive *IS* approach, however, seldom works well in practice because *IS* suffers when the posterior  $p(\mathbf{z}|\mathbf{x}; \theta)$  has multi-modality or  $\mathbf{z}$  is of high dimensionality, resulting in high variance of the approximated estimate of (3.2) [Bengtsson et al., 2008; Dowling et al., 2018; Tokdar and Kass, 2010]. This would be why not much studies have used *IS* to learn deep generative

models.

The novel feature of the proposed learning method is that we apply the idea of *warm-up* [Bowman et al., 2015] by taking a convex combination of the expected complete log-likelihood and the *ELBO* function to stabilize the E-step with *IS* at the early stage of learning. That is, we replace  $p(\mathbf{z}|\mathbf{x}; \theta^{\text{curr}})$  in (3.2) by  $\alpha \cdot p(\mathbf{z}|\mathbf{x}; \theta^{\text{curr}}) + (1 - \alpha) \cdot q(\mathbf{z}|\mathbf{x}; \phi)$  for some  $\alpha \in [0, 1]$  and use *IS* to approximate the corresponding integration at each E-step with  $q(\mathbf{z}|\mathbf{x}; \phi)$  as the proposal distribution. We start the algorithm with  $\alpha = 0$  and increase  $\alpha$  up to 1 as the iteration proceeds, which stabilizes the learning procedure and allows the parameter  $\theta$  to move to  $\hat{\theta}^{\text{MLE}}$  incrementally, where  $\hat{\theta}^{\text{MLE}}$  is the maximizer of the marginal log-likelihood. We call the proposed method the *VAEM* (variational annealed expectation-maximization) algorithm. By analyzing various benchmark datasets, we show empirically that *VAEM* does strengthen the power of performances in terms of the test log-likelihood significantly and consistently compared with the variational methods. Moreover, we illustrate that *VAEM* generates more sharp images and is robust to the choice of the architecture for the variational distribution.

Another appealing feature of *VAEM* is that it can be modified easily for nonstandard cases such as incomplete images or images with different resolutions. This is because *VAEM* is devised to directly maximize the marginal log-likelihood, and the marginal log-likelihood of incomplete data is easily formulated. On the contrary, the variational based methods are hard to be extended for

such cases since the choice of the variational model for incomplete data is not obvious. We demonstrate this feature in Section 3.3.3.

In Section 3.2, we provide brief explanations of related works, and the detailed descriptions of the *VAEM* algorithm for both complete and incomplete data scenarios are given in Section 3.3. The results of numerical experiments including both quantitative and qualitative analyses are presented in Section 3.4.

## 3.2 Review of existing methods for training deep generative models

Here, we give brief explanations of learning methods of deep generative models using variational and *MCMC* techniques.

### 3.2.1 *VAE* [Kingma and Welling, 2013]

The *VAE* (variational auto encoder) method (Kingma and Welling [2013]; Rezende et al. [2014]) is one of the most popular algorithms to learn deep generative models based on the variational inference. *VAE* maximizes the lower bound of the marginal log-likelihood, called *ELBO* derived from Jensen’s inequality with a variational distribution  $q(\mathbf{z}|\mathbf{x}; \phi)$  parametrized by  $\phi$  given as

$$\begin{aligned} \log p(\mathbf{x}; \theta) &= \log \int q(\mathbf{z}|\mathbf{x}; \phi) \cdot \frac{p(\mathbf{x}, \mathbf{z}; \theta)}{q(\mathbf{z}|\mathbf{x}; \phi)} d\mathbf{z} \\ &\geq \int q(\mathbf{z}|\mathbf{x}; \phi) \cdot \log \left[ \frac{p(\mathbf{x}, \mathbf{z}; \theta)}{q(\mathbf{z}|\mathbf{x}; \phi)} \right] d\mathbf{z} =: ELBO(\theta, \phi; \mathbf{x}). \end{aligned}$$

*VAE* approximates the *ELBO* function by using the Monte Carlo method,

$$\widehat{ELBO}(\theta, \phi; \mathbf{x}) := \frac{1}{m} \sum_{h=1}^m \log \left[ \frac{p(\mathbf{x}, \mathbf{z}_h; \theta)}{q(\mathbf{z}_h | \mathbf{x}; \phi)} \right],$$

where  $\mathbf{z}_h \sim q(\mathbf{z} | \mathbf{x}; \phi)$  for  $h = 1, \dots, m$ , and maximizes the  $\widehat{ELBO}$  function with respect to  $\theta$  and  $\phi$ .

Note that *ELBO* can be rewritten as

$$ELBO(\theta, \phi; \mathbf{x}) = \log p(\mathbf{x}; \theta) - KL[q(\mathbf{z} | \mathbf{x}; \phi) \| p(\mathbf{z} | \mathbf{x}; \theta)],$$

where  $KL(\cdot \| \cdot)$  is the Kullback-Leibler divergence. If the variational distribution is equal to the model posterior, which means the KL divergence is zero, then the *VAE* method yields the maximum likelihood estimate  $\hat{\theta}^{\text{MLE}}$  exactly. In contrast, the estimated model  $p(\mathbf{x}; \hat{\theta})$  by *VAE* might not be close to the  $p(\mathbf{x}; \hat{\theta}^{\text{MLE}})$  if the discrepancy between  $p(\mathbf{z} | \mathbf{x}; \theta)$  and  $q(\mathbf{z} | \mathbf{x}; \phi)$  is not small.

### 3.2.2 *IWAE* [Burda et al., 2015]

In recent years there have been several follow-up studies to enhance the *VAE* method. One of the most popular improvements is called *IWAE* (importance-weighted auto encoder, Burda et al. [2015]) which is a variational inference strategy capable of producing arbitrarily tight *ELBO*s. Instead of the *ELBO* function, for a given observable datum  $\mathbf{x}$ , *IWAE* uses the following function with multi samples of a given variational distribution parametrized by

$\phi$ ,  $q(\mathbf{z}|\mathbf{x}; \phi)$ , given as

$$\begin{aligned} \log p(\mathbf{x}; \theta) &\geq \mathbb{E}_{\mathbf{z}_1, \dots, \mathbf{z}_K \sim q(\mathbf{z}|\mathbf{x}; \phi)} \left[ \log \left( \frac{1}{K} \sum_{k=1}^K \frac{p(\mathbf{x}, \mathbf{z}_k; \theta)}{q(\mathbf{z}_k|\mathbf{x}; \phi)} \right) \right] \\ &=: IWAE(\mathbf{x}; \theta, \phi), \end{aligned}$$

where  $K$  is the number of samples from the variational distribution  $q(\mathbf{z}|\mathbf{x}; \phi)$ . Like *VAE*, in practice, *IWAE* approximates the above lower bound function by using the Monte Carlo method and maximizes the approximated lower bound given as

$$IW\hat{A}E(\mathbf{x}; \theta, \phi) := \log \left( \frac{1}{K} \sum_{k=1}^K \frac{p(\mathbf{x}, \mathbf{z}_k; \theta)}{q(\mathbf{z}_k|\mathbf{x}; \phi)} \right),$$

where  $\mathbf{z}_1, \dots, \mathbf{z}_K \sim q(\mathbf{z}|\mathbf{x}; \phi)$ . *IWAE* maximizes the above term with respect to  $\theta$  and  $\phi$ .

It is known that the lower bound function  $IWAE(\mathbf{x}; \theta, \phi)$  is a more tight lower bound of the marginal likelihood than that of *VAE* [Cremer et al., 2017]. Cremer et al. [2017] establishes that the *IWAE* function is the standard *ELBO* function whose variational distribution is more complex than  $q(\mathbf{z}|\mathbf{x}; \phi)$  used in *VAE*.

### 3.2.3 *HMC-DLGM* [Hoffman, 2017]

Another stream of training deep generative models is to directly maximize the log-likelihood by using the *MCMC* method. It is known that the Hamiltonian Monte Carlo method (*HMC*, Duane et al. [1987]; Neal [1993]; Neal et al. [2011]) is a powerful *MCMC* algorithm to learn deep generative models. To estimate  $\nabla_{\theta} \log p(\mathbf{x}; \theta)$  in (3.1), Hoffman [2017] utilized an *HMC* algorithm

to sample from the model posterior  $p(\mathbf{z}|\mathbf{x}; \theta)$ . For this purpose, the model posterior is replaced by the refined distribution,  $\tilde{q}$ , as

$$\tilde{q}_{\epsilon,L,K}(\mathbf{z}|\mathbf{x}) = \int_{\tilde{\mathbf{z}}} q(\tilde{\mathbf{z}}|\mathbf{x}; \phi) \cdot HMC_{\epsilon,L,K}(\mathbf{z}|\tilde{\mathbf{z}}, \mathbf{x}, \theta) d\tilde{\mathbf{z}},$$

where  $HMC_{\epsilon,L,K}(\mathbf{z}|\tilde{\mathbf{z}}, \mathbf{x}, \theta)$  is the distribution of the last sample from an  $K$ -step  $HMC$  chain with step size vector  $\epsilon$  and  $L$  leapfrog steps per iteration. Despite the accurate sampling from the model posterior, this approach is difficult to scale up to complex architectures or large datasets because of heavy computation.

### 3.3 Proposed method

In this section, we describe the learning procedure of *VAEM* in detail. We spell out the proposed method for both complete and incomplete data scenarios. Let  $\mathbf{x}_1, \dots, \mathbf{x}_n$  be i.i.d. observed data with  $p$  dimension, and  $p(\mathbf{x}; \theta) = \int p(\mathbf{x}, \mathbf{z}; \theta) d\mathbf{z}$  be a generative model with a latent variable  $\mathbf{z}$  parametrized by  $\theta \in \Theta$ . We denote a proposal model given a datum  $\mathbf{x}$  and a parameter  $\phi \in \Phi$  by  $q(\mathbf{z}|\mathbf{x}; \phi)$ .

The *VAEM* algorithm consists of three steps: 1) expectation step (E-step), 2) maximization step (M-step), and 3) proposal step (P-step).

#### 3.3.1 *VAEM* for complete data

**E-step** Let  $\theta^{\text{curr}}$  and  $\phi^{\text{curr}}$  be the current estimated parameters of the generative model and the proposal model, respectively. Then, the E-step for a datum  $\mathbf{x}$  and an annealing controller  $\alpha \in [0, 1]$  is

formulated by

$$Q(\theta|\theta^{\text{curr}}; \mathbf{x}, \alpha) := \int \log p(\mathbf{x}, \mathbf{z}; \theta) \cdot [\alpha \cdot p(\mathbf{z}|\mathbf{x}; \theta^{\text{curr}}) + (1 - \alpha) \cdot q(\mathbf{z}|\mathbf{x}; \phi^{\text{curr}})] d\mathbf{z}.$$

We apply *self-normalized IS* and the Monte Carlo method to estimate  $Q(\theta|\theta^{\text{curr}}; \mathbf{x}, \alpha)$  by drawing  $m+m'$  samples  $\mathbf{z}_1, \dots, \mathbf{z}_m, \mathbf{z}'_1, \dots, \mathbf{z}'_{m'}$  from the current proposal distribution  $q(\mathbf{z}|\mathbf{x}; \phi^{\text{curr}})$ :

$$\begin{aligned} \hat{Q}(\theta|\theta^{\text{curr}}, \phi^{\text{curr}}; \mathbf{x}, \alpha) &:= \sum_{h=1}^m \frac{\alpha \cdot w_h}{\sum_{h'=1}^m w_{h'}} \cdot \log p(\mathbf{x}, \mathbf{z}_h; \theta) \\ &\quad + \sum_{h'=1}^{m'} \frac{1 - \alpha}{m'} \cdot \log p(\mathbf{x}, \mathbf{z}'_{h'}; \theta), \end{aligned} \tag{3.3}$$

where  $w_h := \frac{p(\mathbf{x}, \mathbf{z}_h; \theta^{\text{curr}})}{q(\mathbf{z}_h|\mathbf{x}; \phi^{\text{curr}})}$ . Then, the final approximated E-step for all train data is the sum of all  $\hat{Q}(\theta|\theta^{\text{curr}}, \phi^{\text{curr}}; \mathbf{x}_i, \alpha)$ :

$$\hat{Q}(\theta|\theta^{\text{curr}}, \phi^{\text{curr}}; \alpha) := \sum_{i=1}^n \hat{Q}(\theta|\theta^{\text{curr}}, \phi^{\text{curr}}; \mathbf{x}_i, \alpha). \tag{3.4}$$

We assign  $\alpha$  to zero in the initial stage and increase it incrementally up to one as the iteration proceeds. The details of the annealing strategy are illustrated in the experimental analysis in Section 3.4.1.

**M-step** In the M-step, we update  $\theta$  by maximizing the term (3.4) with respect to  $\theta$  :

$$\max_{\theta \in \Theta} \hat{Q}(\theta|\theta^{\text{curr}}, \phi^{\text{curr}}; \alpha)$$

In practice, we update the parameter by one step iteration of an SGD based algorithm rather than the full optimization.

**P-step** We update  $\phi$  by minimizing the KL divergence between  $q(\mathbf{z}|\mathbf{x}; \phi)$  and  $p(\mathbf{z}|\mathbf{x}; \theta^{\text{curr}})$ , which is equivalent to minimizing  $-ELBO(\theta^{\text{curr}}, \phi; \mathbf{x})$ . Let  $\hat{R}(\phi|\hat{\theta}^{\text{curr}}) := -\sum_{i=1}^n ELBO(\theta^{\text{curr}}, \phi; \mathbf{x}_i)$ . In the P-step, we update  $\phi$  by minimizing the term  $\hat{R}(\phi|\hat{\theta}^{\text{curr}})$  with respect to  $\phi \in \Phi$ .

Algorithm 1 outlines the algorithm of *VAEM*.

**Remark 3.3.1. Computational efficiency issue** *At a glance, VAEM seems to require more computation resources than VAE, but it is not true. VAE needs to compute  $ELBO(\theta, \phi; \mathbf{x})$ , which requires computation of two kinds of terms:  $p(\mathbf{x}, \mathbf{z}_h; \theta)$ s and  $q(\mathbf{z}_h|\mathbf{x}; \phi)$ s. On a closer look, we can easily see that all terms to be calculated in VAEM are exactly the same as those in VAE, which means that VAEM is competitive in terms of computational efficiency.*

### 3.3.2 Role of Annealing

Annealing strategy stabilizes *IS* in the early stage of learning and thus makes the parameter  $\theta$  gradually move toward  $\hat{\theta}^{\text{MLE}}$ , which could be explained by the trade-off between bias and variance. Instead of (3.2), *VAEM* uses an approximation of the expected gradient of the joint log-likelihood with a mixture of  $p(\mathbf{z}|\mathbf{x}; \theta^{\text{curr}})$  and  $q(\mathbf{z}|\mathbf{x}; \phi^{\text{curr}})$ . Then, the bias and the variance of  $\nabla_{\theta} \hat{Q}^{\text{mod}}$  are

given as

$$\begin{aligned}\text{Var} &:= \text{Var}_q \left[ \nabla_{\theta} \hat{Q}^{\text{mod}} \right] \\ &= \alpha \cdot \text{Var}_q \left[ \nabla_{\theta} \hat{Q}_p^{\text{mod}} \right] + (1 - \alpha) \cdot \text{Var}_q \left[ \nabla_{\theta} \hat{Q}_q^{\text{mod}} \right]\end{aligned}$$

and

$$\begin{aligned}\text{Bias} &:= \mathbb{E}_q \left[ \nabla_{\theta} \hat{Q}^{\text{mod}} \right] - \nabla_{\theta} Q \\ &\approx (1 - \alpha) \cdot \left( \mathbb{E}_q \left[ \nabla_{\theta} \hat{Q}_q^{\text{mod}} \right] - \nabla_{\theta} Q \right),\end{aligned}$$

where

$$\nabla_{\theta} \hat{Q}_p^{\text{mod}} := \sum_{h=1}^m \frac{w_h}{\sum_{h'=1}^m w_{h'}} \cdot \nabla_{\theta} \log p(\mathbf{x}, \mathbf{z}_h; \theta),$$

$$\nabla_{\theta} \hat{Q}_q^{\text{mod}} := \sum_{h=1}^m \frac{1}{m} \cdot \nabla_{\theta} \log p(\mathbf{x}, \mathbf{z}_h; \theta),$$

and  $\mathbb{E}_q$  and  $\text{Var}_q$  refer to the expectation and variance with respect to the distribution  $q(\mathbf{z}|\mathbf{x}; \phi)$ , respectively. Note that if  $\alpha$  becomes small, then the variance decreases and the bias increases, and vice versa, because the variance of  $\nabla_{\theta} \hat{Q}_p^{\text{mod}}$  is larger than that of  $\nabla_{\theta} \hat{Q}_q^{\text{mod}}$  in general. At the initial learning stage, the current  $\theta$  is far from  $\hat{\theta}^{\text{MLE}}$ , which results in the value of the variance being large because of the large value of  $|\nabla_{\theta} p(\mathbf{x}, \mathbf{z}; \theta)|$ . A large variance of  $\nabla_{\theta} \hat{Q}^{\text{mod}}$  hampers the training procedure; thus, we need to assign  $\alpha$  as being small at the cost of the bias to stabilize the learning. On the contrary, after sufficient iterations proceed, the current  $\theta$  becomes close to  $\hat{\theta}^{\text{MLE}}$ , which means the curvature of  $\log p(\mathbf{x}, \mathbf{z}; \theta)$  is smooth and the value of the variance becomes small. Therefore, a

large value of  $\alpha$  does not disturb the learning procedure anymore, and it is desirable to increase  $\alpha$  for reducing the bias.

To demonstrate the validity of the proposed annealing strategy clearly, we compared the test log-likelihood values of the four learning methods - 1) *VAE*, 2) *VAEM* with  $\alpha = 1$  and the initial  $\theta$  and  $\phi$  estimated by *VAE*, 3) *VAEM* with  $\alpha = 1$  and random initial for  $\theta$  and  $\phi$ , and 4) *VAEM* (with annealing  $\alpha$  and random initial for  $\theta$  and  $\phi$ ) over three benchmark datasets, whose results are given in Table 3.1. First of all, *VAEM* has the highest test log-likelihood values consistently which confirms that the annealing strategy is necessary.

### 3.3.3 *VAEM* for incomplete data

Here, we illustrate the *VAEM* method for incomplete data called *missVAEM*. For a given datum  $\mathbf{x} \in \mathbb{R}^p$  and set  $o \subset \{1, \dots, p\}$ , let us assume the situation that we only observe  $\mathbf{x}^{(o)} = (x_j, j \in o)$ . Denote the missing variable of  $\mathbf{x}$  by  $\mathbf{x}^{(m)} := (x_j, j \in o^c)$ . Let  $\mathbf{x}_1^{(o)}, \dots, \mathbf{x}_n^{(o)}$  be i.i.d. observed data and

$$p(\mathbf{x}^{(o)}; \theta) = \int p(\mathbf{x}^{(o)} | \mathbf{z}; \theta) \cdot p(\mathbf{x}^{(m)} | \mathbf{z}, \mathbf{x}^{(o)}; \theta) p(\mathbf{z}; \theta) d\mathbf{z} d\mathbf{x}^{(m)}$$

be a likelihood of a generative model with a latent variable  $\mathbf{z}$  parametrized by  $\theta \in \Theta$ . For simplicity, we assume that  $\mathbf{x}$ s are independent given  $\mathbf{z}$ , that is,  $\mathbf{x}^{(o)}$  and  $\mathbf{x}^{(m)}$  are independent given  $\mathbf{z}$  for all  $o \subset \{1, \dots, p\}$  (Note that the conditional independent assumption almost holds for generative models except for autoregressive models.). We also denote a proposal distribution given

a datum  $\mathbf{x}^{(o)}$  and parameter  $\phi \in \Phi$  by  $q(\mathbf{z}, \mathbf{x}^{(m)} | \mathbf{x}^{(o)}; \phi)$ .

**Choice of proposal distribution** We propose to use the following proposal distribution for  $q(\mathbf{z}, \mathbf{x}^{(m)} | \mathbf{x}^{(o)}; \phi)$ :

$$q(\mathbf{z}, \mathbf{x}^{(m)} | \mathbf{x}^{(o)}; \phi) := p(\mathbf{x}^{(m)} | \mathbf{z}; \theta^{\text{curr}}) \cdot q(\mathbf{z} | \tilde{\mathbf{x}}; \phi)$$

where  $\tilde{\mathbf{x}} = (\mathbf{x}^{(o)}, \tilde{\mathbf{x}}^{(m)})$  is a completion vector for some reasonably imputed value  $\tilde{\mathbf{x}}^{(m)}$  of  $\mathbf{x}^{(m)}$ , and  $q(\mathbf{z} | \mathbf{x}; \phi)$  has the same architecture of  $q$  in *VAEM* for complete data. In this thesis, we calculate  $\tilde{\mathbf{x}}^{(m)}$  as follow:

- Calculate the mean vector of the distribution  $q(\mathbf{z} | \mathbf{x}^{(o)}, \mathbf{0}; \phi^{\text{curr}})$  and denote it by  $\tilde{\mathbf{z}}$ .

- Define  $\tilde{\mathbf{x}}^{(m)}$  be the mean vector of the distribution  $p(\mathbf{x}^{(m)} | \tilde{\mathbf{z}}; \theta^{\text{curr}})$ .

The goal is to estimate  $\theta$ , which maximizes the sum of log-likelihood functions  $\sum_{i=1}^n \log p(\mathbf{x}_i^{(o)}; \theta)$ .

**E-step** Let  $\theta^{\text{curr}}$  and  $\phi^{\text{curr}}$  be the current estimated parameters of a generative model and a proposal model, respectively. Then, the E-step for an observed datum  $\mathbf{x}^{(o)}$  and an annealing controller  $\alpha \in [0, 1]$  is formulated by

$$Q^{\text{miss}}(\theta | \theta^{\text{curr}}; \mathbf{x}^{(o)}, \alpha) := \int \log p(\mathbf{x}^{(o)}, \mathbf{x}^{(m)}, \mathbf{z}; \theta) \cdot \left[ \alpha \cdot p(\mathbf{z}, \mathbf{x}^{(m)} | \mathbf{x}^{(o)}; \theta^{\text{curr}}) + (1 - \alpha) \cdot q(\mathbf{z}, \mathbf{x}^{(m)} | \mathbf{x}^{(o)}; \phi^{\text{curr}}) \right] d\mathbf{z}.$$

We apply *self-normalized IS* and Monte Carlo method to estimate  $Q^{\text{miss}}(\theta | \theta^{\text{curr}}; \mathbf{x}^{(o)}, \alpha)$  by drawing  $m + m'$  random tuples

$(\mathbf{z}_1, \mathbf{x}_1^{(m)}), \dots, (\mathbf{z}_m, \mathbf{x}_m^{(m)}), (\mathbf{z}'_1, \mathbf{x}'_1^{(m)'})', \dots, (\mathbf{z}'_m, \mathbf{x}'_m^{(m)'})'$  from the current proposal model  $q(\mathbf{z}, \mathbf{x}^{(m)} | \mathbf{x}^{(o)}; \phi^{\text{curr}})$ :

$$\begin{aligned} \hat{Q}^{\text{miss}}(\theta | \theta^{\text{curr}}, \phi^{\text{curr}}, \mathbf{x}^{(o)}, \alpha) &:= \sum_{h=1}^m \frac{\alpha \cdot w_h}{\sum_{h'=1}^m w_{h'}} \cdot \log p(\mathbf{x}^{(o)}, \mathbf{x}_h^{(m)}, \mathbf{z}_h; \theta) \\ &+ \sum_{h'=1}^{m'} \frac{1 - \alpha}{m} \cdot \log p(\mathbf{x}^{(o)}, \mathbf{x}_{h'}^{(m)'}, \mathbf{z}'_{h'}; \theta) \end{aligned} \quad (3.5)$$

where

$$w_h := \frac{p(\mathbf{x}^{(o)}, \mathbf{x}_h^{(m)}, \mathbf{z}_h; \theta^{\text{curr}})}{q(\mathbf{z}_h, \mathbf{x}_h^{(m)} | \mathbf{x}^{(o)}; \phi^{\text{curr}})} = \frac{p(\mathbf{z}_h, \mathbf{x}^{(o)}; \theta^{\text{curr}})}{q(\mathbf{z}_h | \tilde{\mathbf{x}}; \phi^{\text{curr}})}.$$

Then, the final approximated E-step for all train data is the sum of all  $\hat{Q}^{\text{miss}}(\theta | \theta^{\text{curr}}, \phi^{\text{curr}}, \mathbf{x}_i^{(o)}, \alpha)$ s:

$$\hat{Q}^{\text{miss}}(\theta | \theta^{\text{curr}}, \phi^{\text{curr}}, \alpha) := \sum_{i=1}^n \hat{Q}^{\text{miss}}(\theta | \theta^{\text{curr}}, \phi^{\text{curr}}, \mathbf{x}_i^{(o)}, \alpha). \quad (3.6)$$

We assign  $\alpha$  to zero in the initial step and increase it incrementally up to one.

**M-step** In M-step, we maximize the term calculated in the E-step, i.e. (3.6), with respect to  $\theta$ .

$$\max_{\theta \in \Theta} \hat{Q}^{\text{miss}}(\theta | \theta^{\text{curr}}, \phi^{\text{curr}}, \alpha)$$

**P-step** We learn the proposal model by minimizing the KL divergence between  $q(\mathbf{z}, \mathbf{x}^{(m)} | \mathbf{x}^{(o)}; \phi^{\text{curr}})$  and  $p(\mathbf{z}, \mathbf{x}^{(m)} | \mathbf{x}^{(o)}; \theta^{\text{curr}})$

with respect to  $\phi$ , which is equivalent to minimizing the following:

$$\begin{aligned}
R^{\text{miss}}(\phi|\hat{\theta}^{\text{curr}}, \mathbf{x}^{(o)}) &:= \int q(\mathbf{z}, \mathbf{x}^{(m)}|\mathbf{x}^{(o)}; \phi) \\
&\quad \cdot \log \left[ \frac{q(\mathbf{z}, \mathbf{x}^{(m)}|\mathbf{x}^{(o)}; \phi)}{p(\mathbf{z}, \mathbf{x}^{(o)}, \mathbf{x}^{(m)}; \theta^{\text{curr}})} \right] d\mathbf{z}d\mathbf{x}^{(m)} \\
&= \int q(\mathbf{z}, \mathbf{x}^{(m)}|\mathbf{x}^{(o)}; \phi) \\
&\quad \cdot \log \frac{q(\mathbf{z}|\tilde{\mathbf{x}}; \phi)}{p(\mathbf{z}, \mathbf{x}^{(o)}; \theta^{\text{curr}})} d\mathbf{z}d\mathbf{x}^{(m)}.
\end{aligned}$$

Like P-step in *VAEM*, we calculate the estimate of

$R^{\text{miss}}(\phi|\hat{\theta}^{\text{curr}}, \mathbf{x}^{(o)})$  using the Monte Carlo method,  $\hat{R}^{\text{miss}}(\phi|\hat{\theta}^{\text{curr}}, \mathbf{x}^{(o)})$ , and minimize the sum  $\sum_{i=1}^n R^{\text{miss}}(\phi|\hat{\theta}^{\text{curr}}, \mathbf{x}_i^{(o)})$  with respect to  $\phi \in \Phi$ .

## 3.4 Empirical analysis

### 3.4.1 Experimental setup

We performed quantitative as well as qualitative analyses to assess the performances of *VAEM* in comparison to other methods by analyzing four image datasets: static biMNIST Larochelle and Murray [2011], dynamic biMNIST Salakhutdinov and Murray [2008], OMNIGLOT Lake et al. [2015], and Caltech 101 Silhouette Marlin et al. [2010].

We refer to Tomczak and Welling [2017] for architectures of generative and proposal models. We considered three deep architectures for  $p(\mathbf{x}|\mathbf{z}; \theta)$ : 1) fully connected neural nets (*MLP*) with the gating mechanisms [Dauphin et al., 2016], 2) convolutional nets

(*CNN*), and 3) *PixelCNN* [Oord et al., 2016]. We fixed the dimensions of the latent vector  $\mathbf{z}$  to 40. For the prior distribution  $p(\mathbf{z}; \theta)$ , we considered two classes of distributions: 1) standard Gaussian distribution  $\mathcal{N}(0, I)$  (*SG*) and 2) mixture of Gaussian distribution with learnable parameters,  $\sum_{k=1}^K \pi_k \mathcal{N}(\mu_k, \Sigma_k)$  (*MoG*). For *MoG*, the covariance matrices are set to be diagonal and the number of mixture components is fixed to 500. In all cases, we model the proposal distribution with *MLP* with two hidden layers of 300 nodes and the gating mechanisms.

We choose the optimal tuning parameter,  $m$ , which is the number of samples for *IS*, using validation data. We fix the tuning parameter  $m'$ , the number of samples for Monte Carlo method, to 1 because we found that  $m'$  does not affect the performances seriously. For the optimization algorithm, the *Adam* algorithm [Kingma and Ba, 2014] is used with the learning rate  $5 \cdot 10^{-4}$  and mini-batches of size 100. For annealing in *VAEM*, we start with the annealing controller  $\alpha$  being zero and increase it by 0.01 after every epoch of the training phase up to 1. The initial values of the parameters of the generative and proposal models are designed according to Glorot and Bengio [2010].

### 3.4.2 Performance results

We compared the test log-likelihood values of *VAEM* and other methods, where the test log-likelihood values are calculated by the method in Burda et al. [2015]; Rezende et al. [2014]; Tomczak and Welling [2017]. First, we compared *VAEM* to *VAE* and *IWAE*

[Burda et al., 2015] with the *SG* prior distribution, whose results are presented in Table 3.2. First of all, *VAEM* always significantly outperforms *VAE*, which amply supports the finding that MLE is valuable and worth pursuing. Besides, *VAEM* competes well with *IWAE*, which is surprising because the variational model of *IWAE* is much more complex than that of *VAEM* [Cremer et al., 2017]. That is, *VAEM* achieves similar performances as *IWAE* with a much simpler proposal model. This advantage comes from the fact that *VAEM* tries to find the MLE which is less sensitive to the choice of the proposal model.

Second, we investigated the log-likelihood performances of the generative models with the *MoG* prior. We also considered the *VampPrior* method (*VP*, Tomczak and Welling [2017]), which uses the prior as the one maximizing the *ELBO* function. It turns out that the resulting prior becomes a mixture distribution depending on learnable prototype data, called *pseudo inputs*. *VP* is known to achieve better performances than *VAE* with the *MoG* prior. The results are summarized in Table 3.3. For all datasets and all architectures, *VAEM* achieves the highest test log-likelihood values. Note that *VAEM* does not need any learnable *pseudo inputs* which are required for *VP*. That is, *VAEM* has memory efficiency as well as superior test log-likelihood performances.

Additionally, we compared *VAEM* with other recent studies. We gathered the results of recent studies from the experimental analysis in Tomczak and Welling [2017]. Although simpler architectures are used, the results of Table 3.4 to 3.7 show that *VAEM*

competes well with recent state-of-the-art methods, which again confirms the usefulness of *VAEM*.

### 3.4.3 Image generation

We verified whether the deep generative model trained by *VAEM* generates realistic images. The generated images of each dataset with *VAEM* are depicted in Figure 3.1 to Figure 3.4. Images generated by other methods are given in the Appendix B. The images generated by *VAEM* are quite sharp and realistic, which indicates that *VAEM* is good at not only log-likelihood performance but also image generation.

### 3.4.4 Ablation study

We investigated the effect of size of samples  $m$  for *IS* in the E-step of *VAEM*. We compared the test log-likelihood values of *VAEM* for various values of  $m$ , where the *MLP* architecture for  $p(\mathbf{x}|\mathbf{z};\theta)$  and the *SG* prior are used. We analyzed the static biMNIST and Omniglot datasets, and the results are given in Figure 3.5. More samples for *IS* does not always improve the performance, and a possible explanation would be that a large sample size leads the generative model to be stuck at a bad local minimum.

Theoretically, the approximation (3.3) of the E-step converges to its limit as the size of the samples  $m$  for *IS* goes to infinite and

the limit does not depend on the choice of the proposal model. This observation leads us to expect that *VAEM* is more robust to the choice of the architecture for the proposal model than the variational based methods. To confirm this conjecture, we compared *VAEM* and *VAE* by varying the numbers of hidden nodes of the proposed (and variational) model. For example, if the number of hidden nodes is 200, the architecture of the proposal model becomes 784-200-200-40. The results in Figure 3.6 and Table 3.8 clearly indicate that *VAEM* outperforms *VAE* consistently, and the margin becomes larger as the number of hidden nodes becomes smaller.

### 3.4.5 *VAEM* for incomplete data

As mentioned in Section 3.1, an additional advantage of *VAEM* compared with variational based methods is the ease of being modified for nonstandard cases. We modified the *VAEM* algorithm for incomplete data called *missVAEM* which is proposed in Section 3.3.3.

For the variational based methods, Mattei and Frelsen [2018] proposed a modification for incomplete data called *missIWAE*, where all missing observations are imputed by 0 in the variational distribution. Even though *missIWAE* is known to perform reasonably well, the corresponding *ELBO* may not be a good lower bound of the marginal log-likelihood because imputation of 0 is hard to be justified theoretically.

We compared *missVAEM* and *missIWAE* of Mattei and Frelsen

[2018] over the static biMNIST dataset for various missing rates. Only  $K = 1$  for *missIWAE*, is considered in the experiment because of the fair comparison in terms of the complexity of proposal (or variational) distribution of  $\mathbf{z}$ . The results are presented in Figure 3.7, which clearly shows that *missVAEM* is robust to the rate of missing data while *missIWAE* becomes worse dramatically as the missing rate increases.

---

**Algorithm 1:** *VAEM* algorithm

---

**Require::** Train dataset:  $\mathcal{D} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$

**Require::** Generative model with joint distribution:

$$p(\mathbf{x}, \mathbf{z}; \theta)$$

**Require::** Proposal model:  $q(\mathbf{z}|\mathbf{x}; \phi)$

**Require::** Size of samples for *IS*:  $m$

**Require::** Size of mini-batch:  $n_{mb}$

**Require::** SGD based learning algorithm:  $\mathcal{L}(\cdot, \cdot)$

**Initialization::** Parameters  $\theta^{\text{curr}}$  and  $\phi^{\text{curr}}$

**Initialization::** Annealing controller  $\alpha = 0$ , increment

$c > 0$  and number of update criteria  $n_u$

**while** *Parameter  $\theta^{\text{curr}}$  converges* **do**

    Sample  $\tilde{\mathbf{x}}_1, \dots, \tilde{\mathbf{x}}_{n_{mb}}$  from  $\mathcal{D}$

**E-step** Calculate  $\hat{Q}^{\text{VAEM}}(\theta|\theta^{\text{curr}}, \phi^{\text{curr}}; \alpha) :=$

$$\sum_{i=1}^{n_{mb}} \hat{Q}^{\text{VAEM}}(\theta|\theta^{\text{curr}}, \phi^{\text{curr}}; \tilde{\mathbf{x}}_i, \alpha).$$

**M-step** Update  $\theta^{\text{curr}}$  as  $\theta^{\text{new}}$ :

$$\theta^{\text{new}} = \mathcal{L}(\hat{Q}^{\text{VAEM}}(\theta|\theta^{\text{curr}}, \phi^{\text{curr}}; \alpha), \theta^{\text{curr}}).$$

**P-step** Calculate

$$\hat{R}(\phi|\hat{\theta}^{\text{curr}}) := - \sum_{i=1}^{n_{mb}} \text{ELBO}(\theta^{\text{curr}}, \phi; \mathbf{x}_i)$$
 and update

$\phi^{\text{curr}}$  as  $\phi^{\text{new}}$ :

$$\phi^{\text{new}} = \mathcal{L}(\hat{R}(\phi|\hat{\theta}^{\text{curr}}), \phi^{\text{curr}}).$$

    After every  $n_u$  updates,  $\alpha \leftarrow \min(\alpha + c, 1)$

**end**

---

Table 3.1: Test log-likelihood values of the four learning methods:  
 1) *VAE*, 2) *VAEM* with  $\alpha = 1$  and the initial  $\theta$  and  $\phi$  estimated by *VAE*, 3) *VAEM* with  $\alpha = 1$  and random initial for  $\theta$  and  $\phi$  and 4) *VAEM* (with annealing  $\alpha$  and random initial for  $\theta$  and  $\phi$ ).

Method	<i>VAE</i>	<i>VAEM</i> ( $\alpha = 1$ ) <i>w/ VAE init.</i>	<i>VAEM</i> ( $\alpha = 1$ )	<i>VAEM</i>
static biMNIST	-88.21	-88.16	-88.41	-87.70
Omniglot	-108.46	-108.31	-106.84	-106.69
Caltech 101	-119.67	-116.72	-119.24	-116.31

Table 3.2: Test log-likelihood between different models and learning methods with the *SG* prior. For *IWAE*,  $K = 50$  is used for *MLP* and  $K = 10$  is used for other models.

Dataset	static biMNIST	dynamic biMNIST	Omniglot	Caltech 101
Model	MLP			
VAEM (SG)	-87.70	-84.35	-106.69	-116.31
VAE (SG)	-88.21	-85.31	-108.46	-119.67
IWAE (SG)	-87.72	-83.97	-106.39	-117.75
Model	CNN			
VAEM (SG)	-83.71	-81.89	-100.50	-106.68
VAE (SG)	-84.63	-84.08	-101.33	-109.24
IWAE (SG)	-83.35	-83.38	-100.07	-107.01
Model	PixelCNN			
VAEM (SG)	-80.87	-79.42	-91.61	-86.73
VAE (SG)	-81.25	-79.51	-91.64	-98.52
IWAE (SG)	-81.27	-79.02	-90.93	-86.07

Table 3.3: Test log-likelihood between different models and learning methods with the mixture prior distributions.

Dataset	static biMNIST	dynamic biMNIST	Omniglot	Caltech 101
Model	MLP			
VAEM (MoG)	-85.39	-82.91	-105.96	-113.07
VAE (MoG)	-85.90	-83.92	-107.22	-117.08
VP	-85.85	-82.95	-107.21	-116.19
Model	CNN			
VAEM (MoG)	-81.87	-80.64	-98.85	-98.40
VAE (MoG)	-82.55	-81.12	-100.24	-101.16
VP	-81.95	-80.88	-99.60	-103.31
Model	PixelCNN			
VAEM (MoG)	-79.54	-78.55	-90.40	-85.55
VAE (MoG)	-79.97	-78.76	-90.61	-87.48
VP	-79.73	-78.61	-90.48	-86.53

Table 3.4: Test log-likelihood for static biMNIST.

Method	Test LL
<i>VAE (L=1) + NF</i> [Rezende and Mohamed, 2015]	-85.10
<i>AVB+AC (L=1)</i> [Mescheder et al., 2017]	-80.20
<i>VLAE</i> [Chen et al., 2016]	-79.30
<i>VAE+IAF</i> [Kingma et al., 2016b]	-79.88
<i>PixelHVAE (L=2) + VP</i> [Tomczak and Welling, 2017]	-79.78
<i>PixelCNN+VAEM (SG)</i>	-80.87
<i>PixelCNN+VAEM (MoG)</i>	-79.54

Table 3.5: Test log-likelihood for dynamic biMNIST.

Method	Test LL
<i>VAE (L=2) + VGP</i> [Tran et al., 2015]	-81.32
<i>CAGEM-0 (L=2)</i> [Maaløe et al., 2017]	-81.60
<i>LVAE (L=5)</i> [Sønderby et al., 2016]	-81.74
<i>VLAE</i> [Chen et al., 2016]	-78.53
<i>VAE+IAF</i> [Kingma et al., 2016b]	-79.10
<i>PixelHVAE (L=2)+VP</i> [Tomczak and Welling, 2017]	-78.45
<i>PixelCNN+VAEM (SG)</i>	-79.45
<i>PixelCNN+VAEM (MoG)</i>	-78.55

Table 3.6: Test log-likelihood for Omniglot.

Method	Test LL
<i>VR-MAX</i> ( $L=2$ ) [Li and Turner, 2016]	-103.72
<i>IWAE</i> ( $L=2$ ) [Burda et al., 2015]	-103.38
<i>LVAE</i> ( $L=5$ ) [Sønderby et al., 2016]	-102.11
<i>VLAE</i> [Chen et al., 2016]	-89.83
<i>PixelHVAE</i> ( $L=2$ ) + <i>VP</i> [Tomczak and Welling, 2017]	-89.76
<i>PixelCNN+VAEM</i> ( <i>SG</i> )	-91.61
<i>PixelCNN+VAEM</i> ( <i>MoG</i> )	-90.40

Table 3.7: Test log-likelihood for Caltech 101.

Method	Test LL
<i>IWAE</i> ( $L=1$ ) [Burda et al., 2015]	-117.21
<i>VR-MAX</i> ( $L=1$ ) [Li and Turner, 2016]	-117.10
<i>VLAE</i> [Chen et al., 2016]	-78.53
<i>PixelHVAE</i> ( $L=2$ ) + <i>VP</i> [Tomczak and Welling, 2017]	-86.22
<i>PixelCNN+VAEM</i> ( <i>SG</i> )	-86.73
<i>PixelCNN+VAEM</i> ( <i>MoG</i> )	-85.55

Table 3.8: Margins of the test log-likelihood values from the Figure 3.6 for various number of hidden nodes.

Num. of hidden nodes	300	200	100	50
Margin	0.463	0.478	0.780	0.781

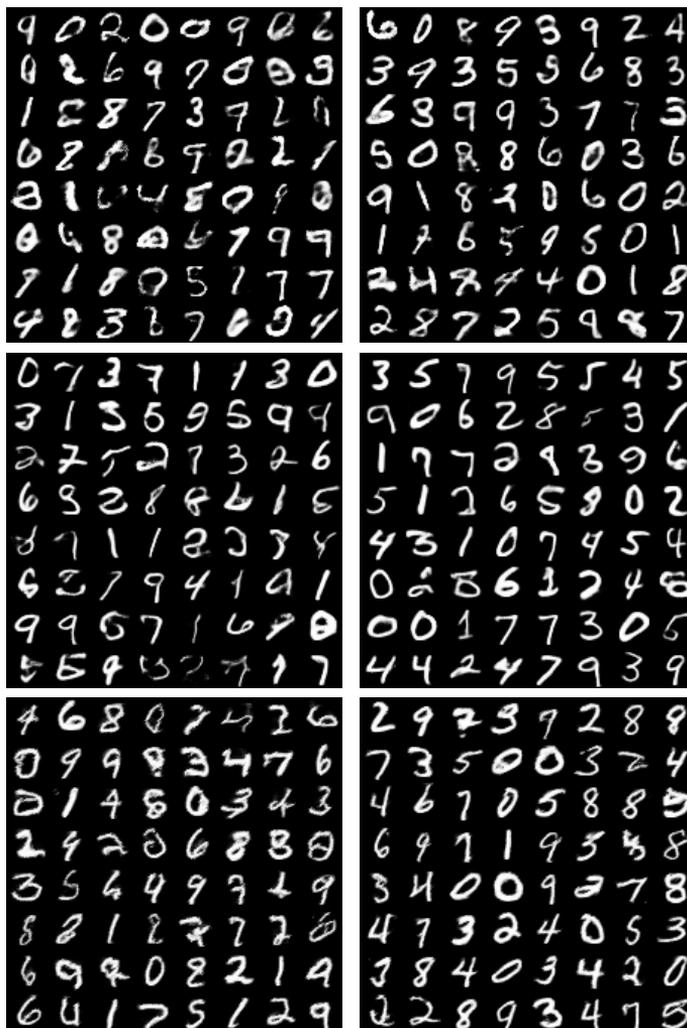


Figure 3.1: Image generation with VAE for static MNIST dataset. Images generated with (1st row) MLP (SG) and MLP (MoG), (2nd row) CNN (SG) and CNN (MoG) and (3rd row) PixelCNN (SG) and PixelCNN (MoG).



Figure 3.2: Image generation with VAE for dynamic MNIST dataset. Images generated with (1st row) MLP (SG) and MLP (MoG), (2nd row) CNN (SG) and CNN (MoG) and (3rd row) PixelCNN (SG) and PixelCNN (MoG).



Figure 3.3: Image generation with VAE for Omniglot dataset. Images generated with (1st row) *MLP (SG)* and *MLP (MoG)*, (2nd row) *CNN (SG)* and *CNN (MoG)* and (3rd row) *PixelCNN (SG)* and *PixelCNN (MoG)*.

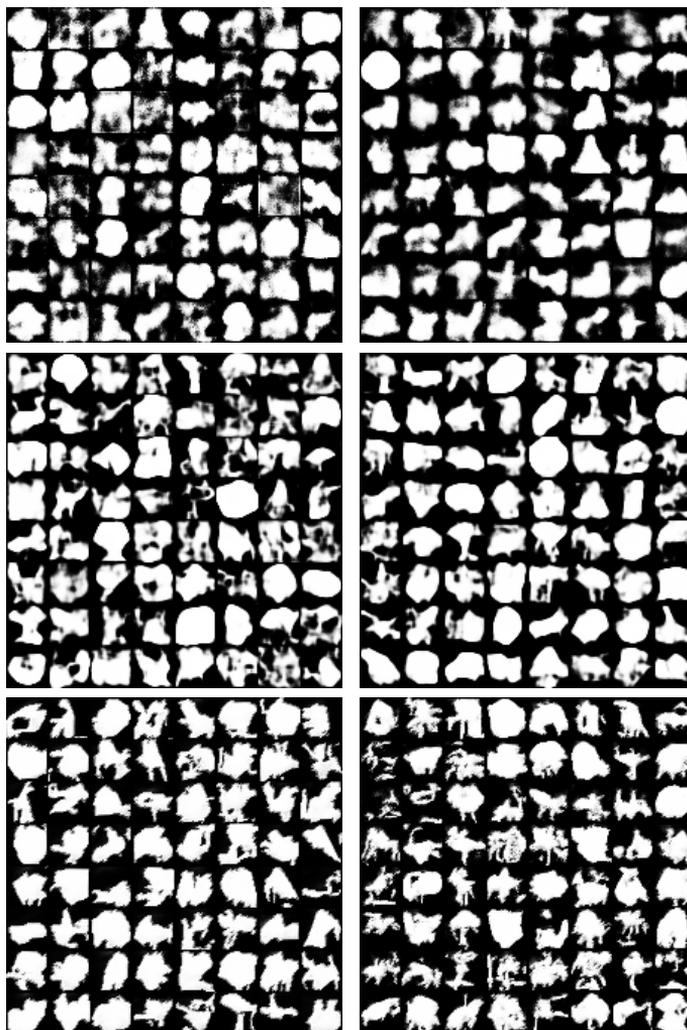


Figure 3.4: Image generation with *VAEM* for Caltech 101 dataset. Images generated with (1st row) *MLP (SG)* and *MLP (MoG)*, (2nd row) *CNN (SG)* and *CNN (MoG)* and (3rd row) *PixelCNN (SG)* and *PixelCNN (MoG)*.

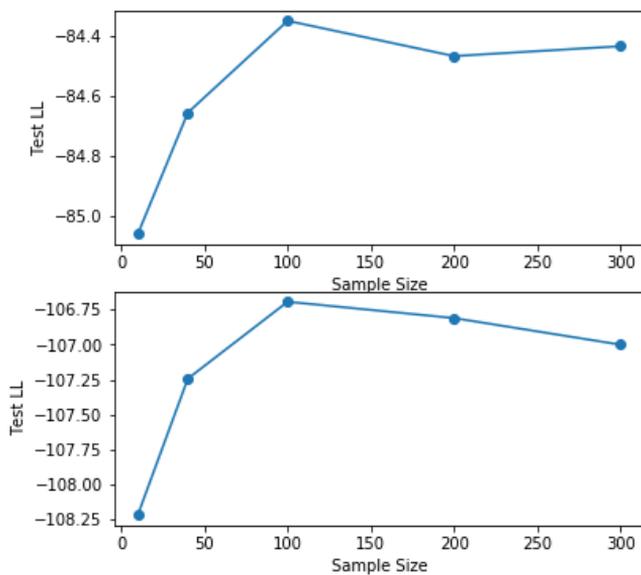


Figure 3.5: Test log-likelihood for various values of the sample size  $m$  for *IS* in *VAEM* on static biMNIST (**top**) and Omniglot dataset (**bottom**). The MLP architecture and the *SG* prior are used for the generative model.

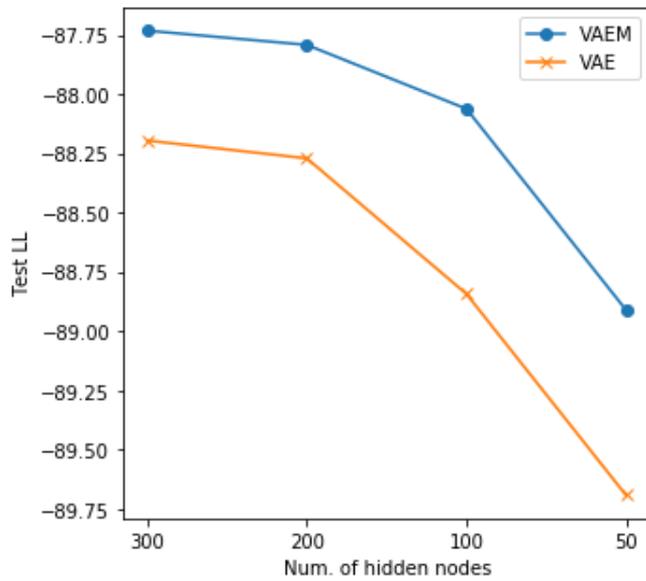


Figure 3.6: Test log-likelihood of *VAEM* and *VAE* for varying numbers of hidden nodes in the proposed (or variational) model on static biMNIST dataset. The *MLP* architecture and the *SG* prior are used for the generative model.

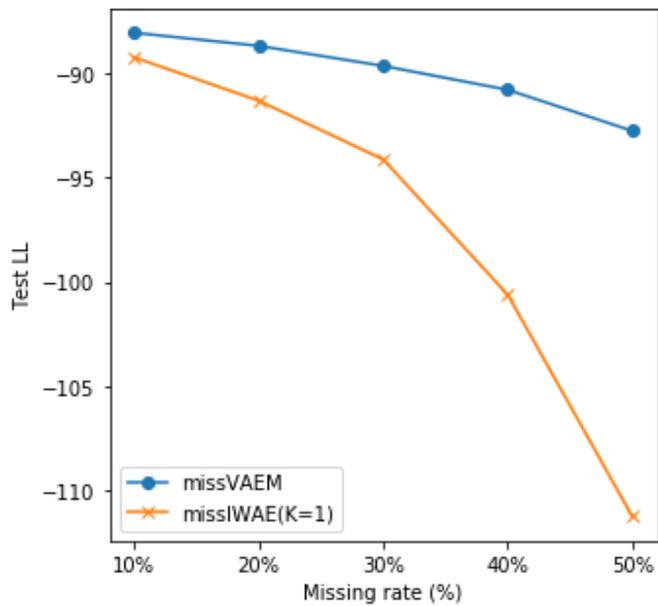


Figure 3.7: Test log-likelihood of *missVAEM* and *missIWAE* for various missing rates on the static biMNIST dataset. The MLP architecture and the *SG* prior are used for the generative model.

## Chapter 4

# Conclusions

In this thesis, we proposed a method called *GAB* for semi-supervised learning and a method called *VAEM* for unsupervised learning. *GAB* utilizes artificial data distributed near the current decision boundary, and we developed an algorithm to generate the artificial data by adopting the adversarial training method. Provided the *cluster assumption*, we theoretically prove that *GAB* finds the Bayes classifier successfully. In numerical experiments, we show that *GAB* achieves almost the state-of-the-art performances with much fewer epochs. Unlike *Bad GAN*, *GAB* only needs to learn a discriminator. Hence, it could be extended without much effort to other learning problems. For example, *GAB* can be modified easily for recurrent neural networks and hence can be applied to sequential data. We will leave this extension for future work.

*VAEM* introduces an annealing procedure by taking the weighted average of the expected complete log-likelihood and the *ELBO*

function at the E-step, which stabilizes the learning procedure and thus improves performances. Furthermore, *VAEM* can be modified easily for nonstandard cases such as the presence of missing data, which is not obvious for variational based methods. Similar to the analysis with missing data done in Section 4.3, it would be possible to apply *missVAEM* to train a deep generative model based on image data with different resolutions, whose results will be reported elsewhere. In future work, we hope to apply *VAEM* to the disentanglement problem [Achille and Soatto, 2018; Chen et al., 2018; Higgins et al., 2017; Kim and Mnih, 2018]. We expect that replacing the reconstruction loss in the variational based methods by the E-step loss function in *VAEM* would result in a better trade-off between density estimation and disentanglement.

# Bibliography

Alessandro Achille and Stefano Soatto. Emergence of invariance and disentanglement in deep representations. *J. Mach. Learn. Res.*, 19(1):1947–1980, January 2018. ISSN 1532-4435. URL <http://dl.acm.org/citation.cfm?id=3291125.3309612>.

Thomas Bengtsson, Peter Bickel, and Bo Li. Curse-of-dimensionality revisited: Collapse of the particle filter in very large scale systems. 2008.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015.

Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542, 2009.

Tian Qi Chen, Xuechen Li, Roger B Grosse, and David K Duve-

naud. Isolating sources of disentanglement in variational autoencoders. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 2610–2620. Curran Associates, Inc., 2018.

Xi Chen, Diederik P Kingma, Tim Salimans, Yan Duan, Prallha Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. Variational lossy autoencoder. *arXiv preprint arXiv:1611.02731*, 2016.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Chris Cremer, Quaid Morris, and David Duvenaud. Reinterpreting importance-weighted autoencoders. *arXiv preprint arXiv:1704.02916*, 2017.

Zihang Dai, Zhilin Yang, Fan Yang, William W Cohen, and Ruslan R Salakhutdinov. Good semi-supervised learning that requires a bad gan. In *Advances in Neural Information Processing Systems*, pages 6513–6523, 2017.

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. *arXiv preprint arXiv:1612.08083*, 2016.

Matthew Dowling, Josue Nassar, Petar M Djurić, and Mónica F

- Bugallo. Improved adaptive importance sampling based on variational inference. In *2018 26th European Signal Processing Conference (EUSIPCO)*, pages 1632–1636. IEEE, 2018.
- Simon Duane, Anthony D Kennedy, Brian J Pendleton, and Duncan Roweth. Hybrid monte carlo. *Physics letters B*, 195(2): 216–222, 1987.
- Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS’10)*. Society for Artificial Intelligence and Statistics, 2010.
- Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014a.
- Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014b.
- Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. In *Advances in neural information processing systems*, pages 529–536, 2005.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on

imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, volume 3, 2017.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

Matthew D. Hoffman. Learning deep latent Gaussian models with Markov chain Monte Carlo. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1510–1519, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. URL <http://proceedings.mlr.press/v70/hoffman17a.html>.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Konstantinos Kamnitsas, Daniel Castro, Loic Le Folgoc, Ian Walker, Ryutaro Tanno, Daniel Rueckert, Ben Glocker, Antonio Criminisi, and Aditya Nori. Semi-supervised learning via

- compact latent space clustering. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2459–2468, 2018.
- Hyunjik Kim and Andriy Mnih. Disentangling by factorising. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2649–2658, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL <http://proceedings.mlr.press/v80/kim18b.html>.
- Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
- Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, pages 3581–3589, 2014.
- Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4743–4751. Curran Associates, Inc., 2016a.

- Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, pages 4743–4751, 2016b.
- Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.
- Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- Abhishek Kumar, Prasanna Sattigeri, and Tom Fletcher. Semi-supervised learning with gans: manifold invariance with improved inference. In *Advances in Neural Information Processing Systems*, pages 5534–5544, 2017.
- Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.
- Hugo Larochelle and Iain Murray. The neural autoregressive distribution estimator. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 29–37, 2011.
- Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

- Chongxuan LI, Taufik Xu, Jun Zhu, and Bo Zhang. Triple generative adversarial nets. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4088–4098. 2017.
- Yingzhen Li and Richard E Turner. Rényi divergence variational inference. In *Advances in Neural Information Processing Systems*, pages 1073–1081, 2016.
- Lars Maaløe, Casper Kaae Sønderby, Søren Kaae Sønderby, and Ole Winther. Auxiliary deep generative models. *arXiv preprint arXiv:1602.05473*, 2016.
- Lars Maaløe, Marco Fraccaro, and Ole Winther. Semi-supervised generation with cluster-aware generative models. *arXiv preprint arXiv:1704.00637*, 2017.
- Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3, 2013.
- Benjamin Marlin, Kevin Swersky, Bo Chen, and Nando Freitas. Inductive principles for restricted boltzmann machine learning. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 509–516, 2010.
- Pierre-Alexandre Mattei and Jes Frelsen. missiwae: Deep generative modelling and imputation of incomplete data. *arXiv preprint arXiv:1812.02633*, 2018.

- Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks. *arXiv preprint arXiv:1701.04722*, 2017.
- Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, Ken Nakae, and Shin Ishii. Distributional smoothing with virtual adversarial training. *arXiv preprint arXiv:1507.00677*, 2015.
- Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *arXiv preprint arXiv:1704.03976*, 2017.
- Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.
- Radford M Neal. Probabilistic inference using markov chain monte carlo methods. 1993.
- Radford M Neal et al. Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2, 2011.
- Aaron Van Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of

*Proceedings of Machine Learning Research*, pages 1747–1756, New York, New York, USA, 20–22 Jun 2016. PMLR. URL <http://proceedings.mlr.press/v48/oord16.html>.

Antti Rasmus, Mathias Berglund, Mikko Honkala, Harri Valpola, and Tapani Raiko. Semi-supervised learning with ladder networks. In *Advances in Neural Information Processing Systems*, pages 3546–3554, 2015.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1530–1538, Lille, France, 07–09 Jul 2015. PMLR. URL <http://proceedings.mlr.press/v37/rezende15.html>.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

Ruslan Salakhutdinov and Iain Murray. On the quantitative analysis of deep belief networks. In *Proceedings of the 25th International Conference on Machine Learning, ICML '08*, pages 872–879, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-205-4. doi: 10.1145/1390156.1390266. URL <http://doi.acm.org/10.1145/1390156.1390266>.

Tim Salimans, Diederik Kingma, and Max Welling. Markov chain monte carlo and variational inference: Bridging the gap. In *In-*

*ternational Conference on Machine Learning*, pages 1218–1226, 2015.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2234–2242, 2016.

Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517*, 2017.

Frank Seide, Gang Li, and Dong Yu. Conversational speech transcription using context-dependent deep neural networks. In *Twelfth annual conference of the international speech communication association*, 2011.

Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. Ladder variational autoencoders. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 3738–3746. Curran Associates, Inc., 2016. URL <http://papers.nips.cc/paper/6275-ladder-variational-autoencoders.pdf>.

Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. Ladder variational autoencoders. In

*Advances in neural information processing systems*, pages 3738–3746, 2016.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

Surya T Tokdar and Robert E Kass. Importance sampling: a review. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(1):54–60, 2010.

Jakub M Tomczak and Max Welling. Vae with a vampprior. *arXiv preprint arXiv:1705.07120*, 2017.

Dustin Tran, Rajesh Ranganath, and David M Blei. The variational gaussian process. *arXiv preprint arXiv:1511.06499*, 2015.

Jason Weston, Frédéric Ratle, Hossein Mobahi, and Ronan Collobert. Deep learning via semi-supervised embedding. In *Neural Networks: Tricks of the Trade*, pages 639–655. Springer, 2012.

Christopher Wolf, Maximilian Karl, and Patrick van der Smagt. Variational inference with hamiltonian monte carlo. *arXiv preprint arXiv:1609.08203*, 2016.

# Appendix A

## Appendix A.

### A.1 Proof of Theorem 2.3.1.

The proof consists of three steps.

[Step 1.] First we show that if  $f \in \tilde{\mathcal{F}}(\epsilon, \xi)$  then  $\mathcal{X}(\epsilon) \cap \mathcal{X}(f, \xi) = \emptyset$ . It is easy to check that  $u_{\epsilon, \xi}(f^0) = 2$ , which means  $u_{\epsilon, \xi}(f) = 2$  for  $f \in \tilde{\mathcal{F}}(\epsilon, \xi)$ . Let assume that  $\mathcal{X}(\epsilon) \cap \mathcal{X}(f, \xi) \neq \emptyset$  for some  $f \in \tilde{\mathcal{F}}(\epsilon, \xi)$ . Since  $\mathcal{X}(\epsilon)$  and  $\mathcal{X}(f, \xi)$  are open sets, there exists an open ball  $B$  such that

$$B \subset \mathcal{X}(\epsilon) \cap \mathcal{X}(f, \xi).$$

Note that  $P_\epsilon^0(B) > 0$  and  $P_{f, \xi}(B) > 0$ . Because  $I(\max_k f_k(\mathbf{x}) > \epsilon) \cdot I(\max_k f_k(\mathbf{x}) < \epsilon) = 0$ , we have  $u_{\epsilon, \xi}(f) < 2$ , which is a contradiction. Therefore,  $\mathcal{X}(\epsilon) \cap \mathcal{X}(f, \xi) = \emptyset$ .

We highlight that if  $f$  satisfies  $\mathcal{X}(\epsilon) \cap \mathcal{X}(f, \xi) = \emptyset$ , for any  $j, k$ ,

$C(\mathbf{x}; f)$  is constant on each  $\mathcal{X}_{kj}(\epsilon)$ .

[Step 2.] Let  $\mathcal{W}$  be the subset of  $\mathcal{X}^n \times \mathcal{Y}^n$  such that

$$\begin{aligned} \sum_{i=1}^n I(X_i \in \mathcal{X}_{kj}(\epsilon), Y_i = k) > \max_{k' \neq k} \left\{ \sum_{i=1}^n I(X_i \in \mathcal{X}_{kj}(\epsilon), Y_i = k') \right\} \\ + \sum_{i=1}^n I(X_i \in \mathcal{X} - \mathcal{X}(\epsilon)) \end{aligned} \quad (\text{A.1})$$

for all  $(k, j)$ , where  $\mathcal{Y} := \{1, \dots, K\}$ . Note that  $C^{\text{Bayes}}(\mathbf{x}) = k$  on  $\mathcal{X}_{kj}(\epsilon)$ . Hence,

$$\begin{aligned} \sum_{i=1}^n I(Y_i = C^{\text{Bayes}}(X_i), X_i \in \mathcal{X}_{kj}(\epsilon)) &= \sum_{i=1}^n I(Y_i = k, X_i \in \mathcal{X}_{kj}(\epsilon)) \\ &> \sum_{i=1}^n I(Y_i = k', X_i \in \mathcal{X}_{kj}(\epsilon)) + \sum_{i=1}^n I(X_i \in \mathcal{X} - \mathcal{X}(\epsilon)) \end{aligned}$$

for any  $k' \neq k$  on  $\mathcal{W}$ . Hence, if there exists a tuple  $(k, j)$  such that  $C(\mathbf{x}; \hat{f}) = k' \neq k$  on  $\mathbf{x} \in \mathcal{X}_{kj}(\epsilon)$ , we have the following inequalities on  $\mathcal{W}$ ,

$$\sum_{i=1}^n (Y_i = C^{\text{Bayes}}(X_i)) \geq \sum_{i=1}^n \sum_{k,j} I(X_i \in \mathcal{X}_{k,j}(\epsilon), Y_i = k)$$

and

$$\begin{aligned}
\sum_{i=1}^n I(Y_i = C(X_i; \hat{f})) &\leq \sum_{i=1}^n I(Y_i = C(X_i; \hat{f}), X_i \in \mathcal{X}(\epsilon)) \\
&\quad + \sum_{i=1}^n I(X_i \in \mathcal{X} - \mathcal{X}(\epsilon)) \\
&\leq \sum_{i=1}^n \sum_{(\tilde{k}, \tilde{j}) \neq (k, j)} I(X_i \in \mathcal{X}_{\tilde{k}\tilde{j}}(\epsilon), Y_i = \tilde{k}) \\
&\quad + \sum_{i=1}^n I(X_i \in \mathcal{X}_{kj}(\epsilon), Y_i = k') \\
&\quad + \sum_{i=1}^n I(X_i \in \mathcal{X} - \mathcal{X}(\epsilon)).
\end{aligned}$$

Therefore, with (A.1), we have

$$\sum_{i=1}^n (Y_i = C^{Bayes}(X_i)) > \sum_{i=1}^n I(Y_i = C(X_i; \hat{f}))$$

on  $\mathcal{W}$ , which contradicts the definition of  $\hat{f}$ . Thus,  $C(\mathbf{x}; \hat{f}) = C^{Bayes}(\mathbf{x})$  on  $\mathbf{x} \in \mathcal{X}(\epsilon)$ . Because  $\Pr\{X \in \mathcal{X}(\epsilon)\} = 1 - \delta(\epsilon)$

$$\Pr\left\{C(X; \hat{f}) = C^{Bayes}(X)\right\} \geq 1 - \delta(\epsilon)$$

on  $\mathcal{W}$ .

[Step 3.] To complete the proof, it suffices to  $P^{(n)}(\mathcal{W}^c) \leq Km \cdot \exp(-nc_*^2/2)$ . Let

$$\begin{aligned}
W_{i,(k,j,k')} &= I(X_i \in \mathcal{X}_{kj}(\epsilon), Y_i = k) - I(X_i \in \mathcal{X}_{kj}(\epsilon), Y_i = k') \\
&\quad - I(X_i \in \mathcal{X} - \mathcal{X}(\epsilon))
\end{aligned}$$

for  $k' \neq k$ . Note that

$$\begin{aligned}
\mathbb{E}(W_{1,(k,j,k')}) &= \int_{\mathcal{X}_{kj}(\epsilon)} \left(P_k^0(\mathbf{x}) - P_{k'}^0(\mathbf{x})\right) p^0(\mathbf{x}) d\mathbf{x} - \delta(\epsilon) \\
&> \epsilon \cdot \Pr(X \in \mathcal{X}_{kj}(\epsilon)) - \delta(\epsilon) > 0
\end{aligned}$$

by use of Condition 1. Let  $c_*^2 = \min_{k,j} \epsilon \cdot P^0(X \in \mathcal{X}_{kj}(\epsilon)) - \delta(\epsilon)$ .

Then, Hoeffding's inequality implies that

$$P^{(n)} \left\{ \sum_{i=1}^n W_{i,(k,j,k')} < 0 \right\} \leq \exp(-nc_*^2/2).$$

By the union bound, we have

$$P^{(n)}(\mathcal{W}^c) \leq P^{(n)} \left\{ \min_{k,j,k'} \sum_{i=1}^n W_{i,(k,j,k')} < 0 \right\} \leq Km \cdot \exp(-nc_*^2/2),$$

which completes the proof.  $\square$

## A.2 Proof of proposition 2.4.1.

Without loss of generality, we assume that  $w' \mathbf{x} + b > 0$ , that is,  $p(y = 1 | \mathbf{x}; \eta) > p(y = 0 | \mathbf{x}; \eta)$ . We will show that there exists  $c > 0$  such that  $w' r^*(\mathbf{x}, c) < 0$ . Let  $p_1$  and  $p_0$  be abbreviated notations of  $p(y = 1 | \mathbf{x}; \eta)$  and  $p(y = 0 | \mathbf{x}; \eta)$ , respectively. Note that

$$\begin{aligned} \operatorname{argmax}_{r, \|r\| \leq c, w'r > 0} KL(\mathbf{x}, r; \eta) &= c \frac{w}{\|w\|} (=: r_1^*) \quad \text{and} \\ \operatorname{argmax}_{r, \|r\| \leq c, w'r < 0} KL(\mathbf{x}, r; \eta) &= -c \frac{w}{\|w\|} (=: r_2^*). \end{aligned}$$

Therefore, we have to show

$$KL(\mathbf{x}, r_2^*; \eta) > KL(\mathbf{x}, r_1^*; \eta).$$

By simple calculation, we can get the following:

$$\begin{aligned} KL(\mathbf{x}, r_2^*; \eta) - KL(\mathbf{x}, r_1^*; \eta) &= -p_1 w' (r_2^* - r_1^*) \\ &\quad + \log \left[ \frac{\exp(w'(\mathbf{x} + r_2^*) + b) + 1}{\exp(w'(\mathbf{x} + r_1^*) + b) + 1} \right]. \end{aligned}$$

Using the Taylor's expansion up to the third-order, we obtain the following:

$$\begin{aligned} \log \left[ \exp \left( w'(\mathbf{x} + r) + b \right) + 1 \right] &= \log \left[ \exp \left( w' \mathbf{x} + b \right) + 1 \right] + p_1 w' r \\ &\quad + \frac{1}{2} p_1 p_0 r' w w' r - \frac{1}{6} p_1 p_0 (p_1 - p_0) \\ &\quad \times \sum_{i,j,k=1}^p w_i w_j w_k r_i r_j r_k \\ &\quad + o(\|r\|^3). \end{aligned}$$

So,

$$\begin{aligned} \log \frac{\exp \left( w'(\mathbf{x} + r_2^*) + b \right) + 1}{\exp \left( w'(\mathbf{x} + r_1^*) + b \right) + 1} &= p_1 w' (r_2^* - r_1^*) \\ &\quad + \frac{1}{3} p_1 p_0 (p_1 - p_0) c^3 \|w\|^3 + o(c^3). \end{aligned}$$

Thus, we have the following equations:

$$\begin{aligned} KL(\mathbf{x}, r_2^*; \eta) - KL(\mathbf{x}, r_1^*; \eta) &= \frac{1}{3} p_1 p_0 (p_1 - p_0) c^3 \|w\|^3 + o(c^3) \\ &= C \cdot c^3 + o(c^3). \end{aligned}$$

Therefore, there exists  $c^* > 0$  such that  $KL(\mathbf{x}, r_2^*; \eta) > KL(\mathbf{x}, r_1^*; \eta)$  for  $\forall 0 < c < c^*$ .  $\square$

### A.3 Extension of proposition 1 for the DNN classifier

Consider a binary classification DNN model with ReLU-like activation function  $p(y = 1|\mathbf{x}; \theta) = (1 + \exp(-g(\mathbf{x}; \theta)))^{-1}$  parameterized by  $\theta$ . Here, the ReLU-like function is the activation function

which is piece-wise linear, such as ReLU [Nair and Hinton, 2010], lReLU [Maas et al., 2013] and PReLU [He et al., 2015]. Because  $g(\mathbf{x}; \theta)$  is piecewise linear, we can write  $g(\mathbf{x}; \theta)$  as

$$g(\mathbf{x}; \theta) = \sum_{j=1}^N \mathbb{I}(\mathbf{x} \in \mathcal{A}_j) \cdot (w_j \mathbf{x} + b_j),$$

where  $\mathcal{A}_j$  is a linear region and  $N$  is the number of linear regions.

For given  $\mathbf{x}$ , suppose  $g(\mathbf{x}; \theta) > 0$ . If  $g(\mathbf{x}; \theta)$  is estimated reasonably, we expect that  $g(\mathbf{x}; \theta)$  is decreasing if  $\mathbf{x}$  moves toward the decision boundary. A formal statement of this expectation would be that  $\mathbf{x} - r \nabla_{\mathbf{x}} g(\mathbf{x}; \theta)$  can arrive at the decision boundary for a finite value of  $r > 0$ , where  $\nabla_{\mathbf{x}}$  is the gradient with respect to  $\mathbf{x}$ . Of course, for  $\mathbf{x}$  with  $g(\mathbf{x}; \theta) < 0$ , we expect that  $\mathbf{x} + r \nabla_{\mathbf{x}} g(\mathbf{x}; \theta)$  can arrive at the decision boundary for a finite value of  $r > 0$ . We say that  $\mathbf{x}$  is *normal* if there is  $r > 0$  such that  $\mathbf{x} - r \nabla_{\mathbf{x}} g(\mathbf{x}; \theta) \text{sign}\{g(\mathbf{x}; \theta)\}$  is located at the decision boundary. We say that a linear region  $\mathcal{A}_j$  is *normal* if all  $\mathbf{x}$  in  $\mathcal{A}_j$  are *normal*. We expect that most of  $\mathcal{A}_j$  are *normal* if  $g(\mathbf{x}; \theta)$  is reasonably estimated so that the probability decreases or increases depending on  $\text{sign}\{g(\mathbf{x}; \theta)\}$  if  $\mathbf{x}$  is getting closer to the decision boundary.

The following proposition proves that the adversarial direction is toward the decision boundary for all  $\mathbf{x}$ s in *normal* linear regions.

**Lemma A.3.1.** *If a linear region  $\mathcal{A}_j$  is normal, then for any  $\mathbf{x} \in \text{int}(\mathcal{A}_j)$ , there exists  $c > 0$  and  $C > 0$  such that*

$$\mathbf{x}^A = \mathbf{x} + C \frac{r_{adv}(\mathbf{x}, c)}{\|r_{adv}(\mathbf{x}, c)\|}$$

*is on the decision boundary.*

**Proof)** Take  $\tilde{c} > 0$  such that  $\mathbf{x} + r \in \mathcal{A}_{\tilde{j}}$  for  $\forall r \in B(\mathbf{x}, \tilde{c})$ . Then, by Proposition 1, there exists  $0 < c^* < \tilde{c}$  such that for  $\forall 0 < c < c^*$ ,

$$\begin{aligned} r_{\text{advr}}(\mathbf{x}, c) &= c \cdot \text{sign}(-b_{\tilde{j}} - w'_{\tilde{j}}\mathbf{x}) \cdot \frac{w_{\tilde{j}}}{\|w_{\tilde{j}}\|} \\ &\propto -\nabla_{\mathbf{x}}g(\mathbf{x}; \theta)\text{sign}\{g(\mathbf{x}; \theta)\}. \end{aligned}$$

$\mathbf{x}$  is *normal*, thus there exists  $C > 0$  such that

$$\mathbf{x}^A = \mathbf{x} + C \frac{r_{\text{advr}}(\mathbf{x}, c)}{\|r_{\text{advr}}(\mathbf{x}, c)\|}$$

belongs to the decision boundary. □

## A.4 Model architectures

All model architectures used in the experiments are based on Miyato et al. [2017].

### A.4.1 MNIST

For the MNIST dataset, we used fully connected NN with four hidden layers, whose numbers of nodes were (1200, 600, 300, and 150) with ReLU activation function [Nair and Hinton, 2010]. All the fully connected layers are followed by BN[Ioffe and Szegedy, 2015].

### A.4.2 SVHN, CIFAR10 and CIFAR100

For SVHN, CIFAR10, and CIFAR100 datasets, we used the CNN architectures. More details are in Table A.1.

SVHN	CIFAR10	CIFAR100
$32 \times 32$ RGB images		
$3 \times 3$ conv. 64 lReLU	$3 \times 3$ conv. 96 lReLU	$3 \times 3$ conv. 128 lReLU
$3 \times 3$ conv. 64 lReLU	$3 \times 3$ conv. 96 lReLU	$3 \times 3$ conv. 128 lReLU
$3 \times 3$ conv. 64 lReLU	$3 \times 3$ conv. 96 lReLU	$3 \times 3$ conv. 128 lReLU
$2 \times 2$ max-pool, stride 2		
dropout, $p = 0.5$		
$3 \times 3$ conv. 128 lReLU	$3 \times 3$ conv. 192 lReLU	$3 \times 3$ conv. 256 lReLU
$3 \times 3$ conv. 128 lReLU	$3 \times 3$ conv. 192 lReLU	$3 \times 3$ conv. 256 lReLU
$3 \times 3$ conv. 128 lReLU	$3 \times 3$ conv. 192 lReLU	$3 \times 3$ conv. 256 lReLU
$2 \times 2$ max-pool, stride 2		
dropout, $p = 0.5$		
$3 \times 3$ conv. 128 lReLU	$3 \times 3$ conv. 192 lReLU	$3 \times 3$ conv. 512 lReLU
$1 \times 1$ conv. 128 lReLU	$1 \times 1$ conv. 192 lReLU	$1 \times 1$ conv. 256 lReLU
$1 \times 1$ conv. 128 lReLU	$1 \times 1$ conv. 192 lReLU	$1 \times 1$ conv. 128 lReLU
global average pool, $6 \times 6 \rightarrow 1 \times 1$		
dense $128 \rightarrow 10$	dense $192 \rightarrow 10$	dense $128 \rightarrow 100$
10-way softmax		100-way softmax

Table A.1: CNN models used in experimental analysis over SVHN, CIFAR10, and CIFAR100. We used leaky ReLU activation function [Maas et al., 2013], and all the convolutional layers and fully-connected layers are followed by BN[Ioffe and Szegedy, 2015].

## Appendix B

# Appendix B.

### B.1 Generated Images

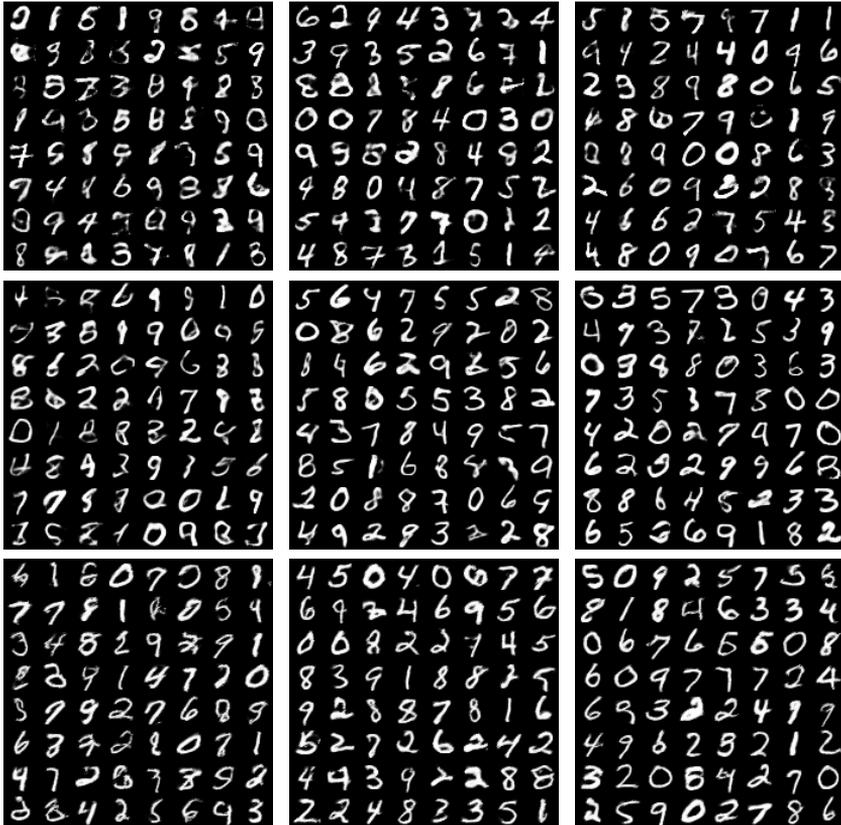


Figure B.1: Generated images for static MNIST dataset with various architectures and learning methods. **(1st row)** *MLP+VAE (SG)*, *MLP+VAE (MoG)* and *MLP+VP*, **(2nd row)** *CNN+VAE (SG)*, *CNN+VAE (MoG)* and *CNN+VP*, and **(3th row)** *PixelCNN+VAE (SG)*, *PixelCNN+VAE (MoG)* and *PixelCNN+VP* are considered.

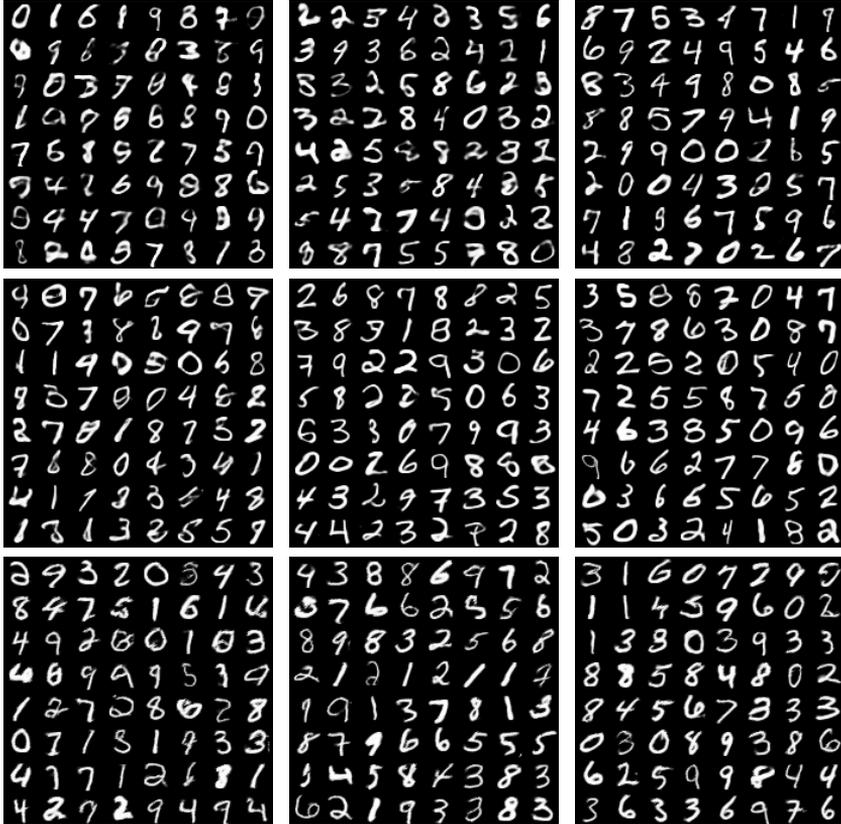


Figure B.2: Generated images for dynamic MNIST dataset with various architectures and learning methods. **(1st row)** *MLP+VAE (SG)*, *MLP+VAE (MoG)* and *MLP+VP*, **(2nd row)** *CNN+VAE (SG)*, *CNN+VAE (MoG)* and *CNN+VP*, and **(3th row)** *PixelCNN+VAE (SG)*, *PixelCNN+VAE (MoG)* and *PixelCNN+VP* are considered.

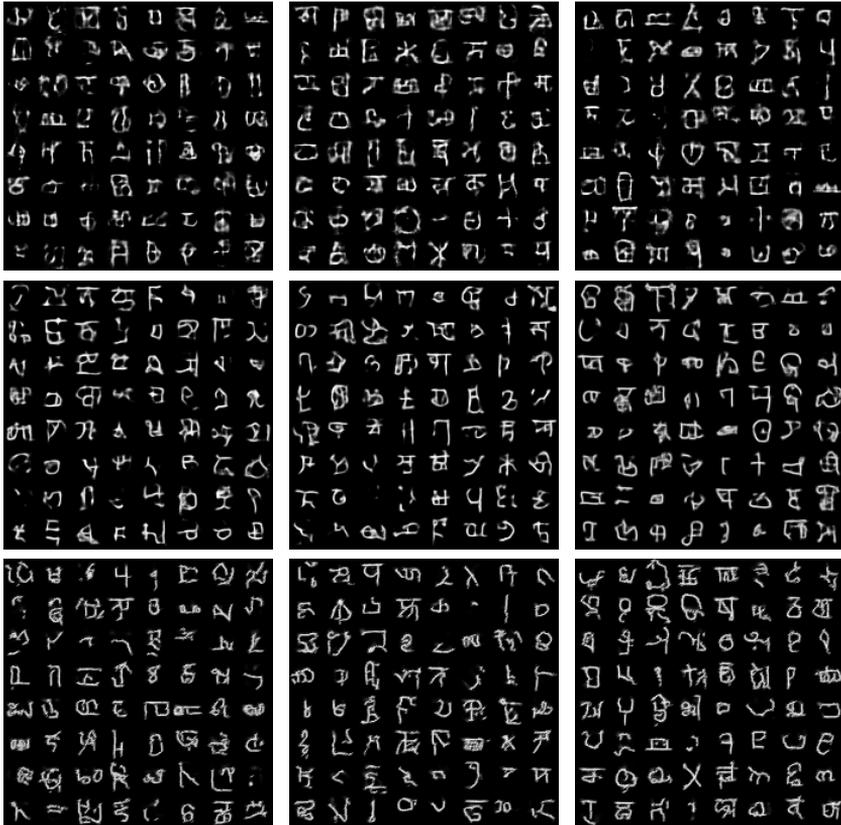


Figure B.3: Generated images for Omniglot dataset with various architectures and learning methods. **(1st row)** *MLP+VAE (SG)*, *MLP+VAE (MoG)* and *MLP+VP*, **(2nd row)** *CNN+VAE (SG)*, *CNN+VAE (MoG)* and *CNN+VP*, and **(3th row)** *PixelCNN+VAE (SG)*, *PixelCNN+VAE (MoG)* and *PixelCNN+VP* are considered.

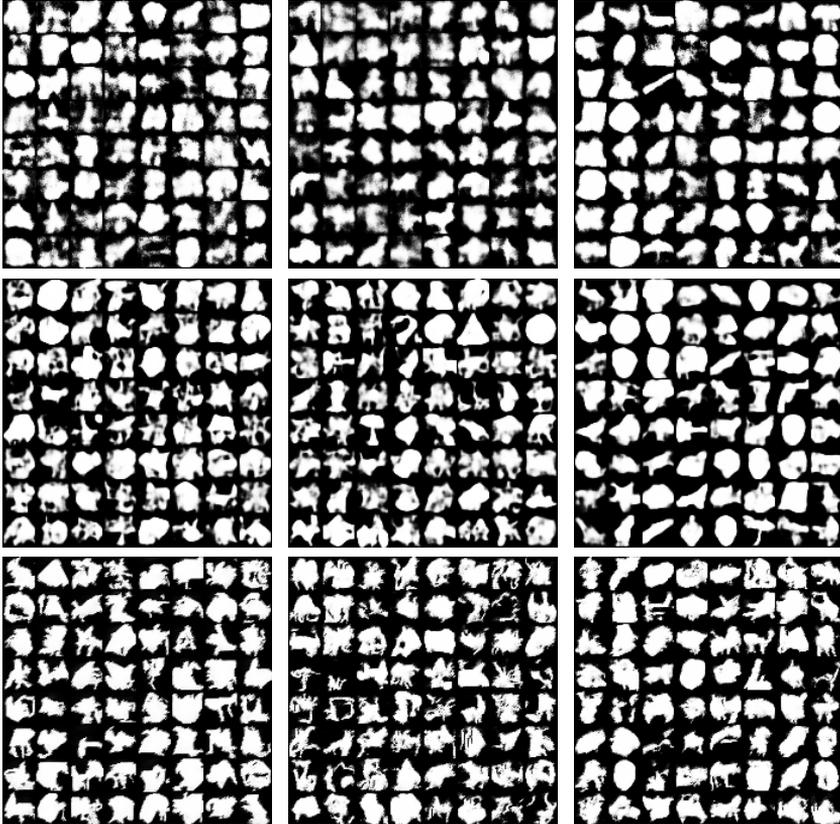


Figure B.4: Generated images for Caltech 101 dataset with various architectures and learning methods. (**1st row**)  $MLP+VAE$  ( $SG$ ),  $MLP+VAE$  ( $MoG$ ) and  $MLP+VP$ , (**2nd row**)  $CNN+VAE$  ( $SG$ ),  $CNN+VAE$  ( $MoG$ ) and  $CNN+VP$ , and (**3th row**)  $PixelCNN+VAE$  ( $SG$ ),  $PixelCNN+VAE$  ( $MoG$ ) and  $PixelCNN+VP$  are considered.

# 국문초록

본 논문은 준지도 학습과 비지도 학습에서 딥러닝 모형을 학습하는 새로운 방법론을 제안한다. 첫째로, 준지도 학습에서는 현재의 분류기의 결정 경계 근방의 인공 자료가 실제 자료와 가능한 겹치지 않는 방향으로 분류기를 학습하는 새로운 준지도 학습 방법을 제시한다. 이를 구현하기 위해 결정 경계 근방과 실제 자료의 중첩 정도를 측정할 수 있는 새로운 지수를 개발하였고, 해당 지수를 최소화하는 방향으로 분류기를 찾는 것을 본 방법론의 목표로 한다. 또한 제안한 중첩 지수를 최소화하는 분류기의 결정 경계는 베이스 분류기의 결정 경계와 거의 같음을 이론적으로 증명하였다. 중첩 지수의 근사를 위해서는 결정 경계 근방의 인공 자료를 생성하는 과정이 필요한데, 본 논문에서는 대립 훈련 방법을 응용한 인공 자료를 생성하는 방법을 제안한다. 본 연구에서 새롭게 제안한 준지도 방법론이 우수한 분류기를 효율적으로 잘 추정할 수 있음을 다양한 벤치마크 자료들에 적용하여 실험적으로 입증하였다.

둘째로, 비지도 학습에서는 EM 알고리즘과 중요도 표집을 통해 관측 변수의 우도 함수를 직접적으로 최대화하는 학습 방법을 제안한다. 특히 워 스타트 방법을 응용하여 E-단계에서 로그 결합

우도의 기댓값 함수와 변분 하한 함수의 가중 평균을 사용하는 새로운 방법을 개발하였으며, 제시한 방법이 우수한 모형을 안정적으로 추정할 수 있게 해줌을 실험적으로 증명하였다. 준지도 학습과 마찬가지로 본 연구에서 제안한 비지도 학습 방법론이 타 방법론에 비해 우수한 성능을 가지고 있으며, 결측 자료가 존재할 때에도 잘 적용될 수 있음을 실험적으로 증명하였다.

**주요어:** 준지도 학습, 군집 가정, 대립 훈련, GAB, 비지도 학습, EM 알고리즘, 중요도 표집, 변분 추정법, VAEM

**학 번:** 2012-23010