Semi-Supervised Learning with Hybrid Generative Models

Vincent Sham

January 9, 2026

Abstract

Supervised learning approaches require a large amount of labelled data, while semi-supervised learning approaches loosen the requirement and achieve a competitive performance with only a few labelled examples. We propose a new semi-supervised learning method that combines likelihood based and adversarial learning. Generally, generative models (e.g., normalizing flows) often perform worse in semi-supervised learning than discriminative models. We demonstrate that the normalizing flow's objective, negative log-likelihood, hurts semi-supervised learning (SSL) performance due to the mode-covering property. In practice, models trained with negative log-likelihood ignore about the low-density regions of the target distribution, yielding poor performance. Therefore, we propose to include adversarial learning in our unsupervised loss so that the models can better fit the low-density areas of the target distribution, resulting in a hybrid generative model. We empirically demonstrate that our method yields a competitive SSL performance while preserving the advantages of generative models.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Deep learning has had enormous success in classification tasks in computer vision [Krizhevsky et al., 2012, He et al., 2016]. Its performance can be on par with or even surpass human performance by leveraging a large amount of high-quality labelled data [Krizhevsky et al., 2012]. By training with a large amount of labelled data, a network can generalize well in testing and be transferred to other tasks successfully.

Unfortunately, obtaining labelled data is still expensive and time-consuming, which is a hurdle for some applications. For instance, annotating medical image data requires domain expertise and specific tools that are costly or dangerous to scale up. Also, human errors may occur during data annotation, and training on the low-quality labelled data can negatively impact performance. Thus, assuring the quality of labelled data is crucial, further increasing the cost of obtaining it. On the other hand, unlabelled data is often relatively cheap and easy to acquire. Therefore, incorporating unlabelled data to improve classification performance is an important research direction.

Semi-supervised learning (SSL) is an active research area and precedes deep learning [Chapelle et al., 2006]. It is a learning paradigm where only a small amount of labelled data is available, but there is a substantial amount of unlabelled data, assuming both labelled and unlabelled data

are drawn from the same distribution. By utilizing unlabelled data, the semi-supervised learning algorithm can gain further insights into the data distribution, leading to better prediction performance for unseen data. This approach is particularly useful when there is no other labelled dataset available with shared knowledge for transfer learning. For example, natural image data knowledge doesn't always transfer well to medical image data, making it challenging to develop a reliable model by pre-training it with a natural dataset and finetuning with a medical dataset.

Our focus is to build a semi-supervised learning model with generative models. In the past, Gaussian Mixture Models (GMMs) were adopted for semi-supervised learning, where each class conditional distribution is set to be Gaussian [Chapelle et al., 2006]. The likelihood of GMMs with labelled and unlabelled data is maximized, e.g., by iterative algorithms such as EM [Moon, 1996]. This classic method does not perform well with complex, high-dimensional data like images due to an insufficient inductive bias [Chapelle et al., 2006]. Recently, generative models parameterized by deep neural networks (i.e., deep generative models) have had tremendous success in learning complex distributions [Kingma and Welling, 2014, Goodfellow et al., 2014, Kobyzev et al., 2021]. Generated images from recent large-scale generative models are sufficiently plausible that humans struggle to detect if the images are "real" or not [Brock et al., 2019].

Normalizing flows [Kobyzev et al., 2021] are a family of deep generative models that transforms a complex distribution into a simple, tractable one with a series of invertible transformations. It allows us to perform both sampling and exact density evaluation efficiently. Semi-supervised learning with deep generative models like normalizing flows is not a new idea [Izmailov et al., 2020, Atanov et al., 2019]. However, its SSL performance is not competitive with other semi-supervised learning methods because learning how the data distributed is relatively challenging. Any discrepancy between generative models and data distributions potentially hurts SSL performance. On the other hand, normalizing flows can offer some advantages over other methods, such as conditional sampling and density evaluation. Thus, we propose a new semi-supervised learning method with

normalizing flows by introducing a hybrid generative method, where the loss objective combines the negative log-likelihood and adversarial learning. The additional adversarial learning allows the model to shift the focus to learn better decision boundaries so that its performance becomes competitive with other semi-supervised learning methods while preserving the advantages of the deep generative models.

In this thesis, we make the following contributions:

- We first investigate the issues of semi-supervised learning with normalizing flows and propose advancements over the previous work.
- We further analyze the training behaviour of semi-supervised learning with negative log-likelihood and propose a hybrid deep generative objective.
- We perform in-depth experiments on toy and benchmark datasets to demonstrate the SSL performance of our method.

Chapter 2

Related Works

Semi-supervised learning (SSL) is a learning problem where only a small amount of labelled data is available with a large amount of unlabelled data. The main goal of SSL is to improve supervised performance by utilizing additional unlabelled data in contrast to fully supervised learning methods, which only use labelled data. In general, all SSL methods can be defined as optimizing the following objective [Yang et al., 2022]:

$$L_{ssl} = \sum_{x,y \sim \mathcal{L}} L_{sup}(x, y) + \sum_{x \sim \mathcal{U}} L_{unsup}(x) + \sum_{x \sim \mathcal{L}+\mathcal{U}} L_{reg}(x)$$

where $\mathcal{L}$ and $\mathcal{U}$ denote labelled and unlabelled datasets, respectively. Each instance in the labelled dataset has both input value x and target value y, while each instance in the unlabelled dataset only contains an input value x.

Semi-supervised learning methods differ in their choices of the supervised loss, the unsupervised loss and regularization. The supervised learning loss highly depends on the types of target values. Most previous works focus on the classification problem in which the target values are discrete, while some works handle continuous target values [Farouk et al., 2009]. Recently, some works have been addressing more complex tasks, like object detection [Sohn et al., 2020] and semantic segmentation [Ouali et al., 2020], in which the labelled data is a lot harder to obtain. The unsuper-

vised losses typically encourage the model to leverage a significantly large amount of unlabelled data to learn the data distribution, which is beneficial for the supervised task. Finally, the regularization terms are related to any form of regularization that does not require label information. In this work, we restrict the scope to the classification problem. Our work mainly focuses on introducing a new unsupervised objective for semi-supervised learning. Therefore, here we will present three major SSL methods from the perspective of unsupervised loss: Variational Autoencoders (VAEs), Normalizing Flows (NFs) and Generative Adversarial Networks (GANs). We also cover the SSL assumptions and some SSL regularization methods for completeness. We suggest referring to Yang et al. [2022] for a more detailed review.

2.1 Semi-Supervised Learning Assumptions

While unlabelled data is relatively easy to acquire, it is not always the case that incorporating unlabelled data can improve performance. For instance, if the unlabelled data does not contain any information about the target value, there is no reason to use the unlabelled data. Even worse, if the labelled and unlabelled data are from different distributions, incorporating unlabelled data can lead to performance degradation. Therefore, SSL is applicable only when the extra information from unlabelled data is beneficial to the supervised task. Specifically, some essential assumptions [Chapelle et al., 2006, Yang et al., 2022] need to be satisfied to apply the SSL method:

- The Smoothess Assumption. *If two input points x_1, x_2 in a high-density region are close, then so should be their corresponding outputs y_1, y_2.* That is, if two points can connect with a path of high density, then their outputs are likely to be similar. On the other hand, if two points are not able to connect with a path of high density, they are separated by a low-density area. Therefore, they do not belong to the same cluster and their outputs are not necessarily similar. Note that two points belonging to different clusters may still have a close target value, but two points with different target values must be from different clusters based on the smoothness assumption.

- Low-density Separation Assumption. *The decision boundary should be in a low-density region.* The low-density separation assumption is closely related to the smoothness assumption. We assume that all classes are separated by decision boundaries lying in a low-density region. If decision boundaries are in a high-density region, two points with different target values can connect with a path of high density. That implies that two points with different target values are close, violating the smoothness assumption.

- The Manifold Assumption. *The (high-dimensional) data lie (roughly) on a low-dimensional manifold.* This assumption is related to a well-known machine learning problem: the curse of dimensionality. When the data lies in high-dimensional spaces, the volume of the space increases exponentially with the dimensionality such that the training data tends to be sparse. As a result, it is challenging to learn the data distribution or explore any pattern from the training data. In contrast, if the data lies on a low-dimensional manifold, our learning algorithm can instead learn to project the data into the corresponding low-dimensional space to avoid the issue.

The above assumptions help ensure the feasibility of SSL. Otherwise, additional unlabelled data is no longer useful for improving supervised learning performance. The first two assumptions describe the structure of the data distribution such that learning the input data distribution is sufficient to infer the output value. The last assumption ensures the effectiveness of learning the input data distribution from finite training data.

2.2 Classification

In general, there are two types of classifiers: discriminative classifiers and generative classifiers. The discriminative classifiers directly model the probability of a target class given the input (i.e., $p_{\mathcal{Y}}(y|x)$). On the other hand, the generative classifiers define a generative model to learn the conditional distribution (i.e., $p_{\mathcal{X}}(x|y)$) and use the Bayes' rule to compute $p_{\mathcal{Y}}(y|x)$ for classification [Chapelle et al., 2006]. Generative classifiers learn how the data is generated to categorize a signal,

while discriminative classifiers do not care about how the data is generated. Hence, discriminative classifiers can discard any class-irrelevant information and are more directly aligned with the goal of supervised learning. As a result, discriminative classifiers usually perform better than generative classifiers, but the latter has some unique features, such as conditional sampling for visualizing class features and likelihood estimation for detecting outliers and out-of-distribution data [Ardizzone et al., 2020]. Therefore, there is a need to improve the performance of generative classifiers, which is an important area of research.

2.3 SSL Unsupervised Loss

In semi-supervised learning, unsupervised losses encourage the model to learn the input data distribution, which is useful for inferring the target value under the smoothness assumption and the low-density separation assumption. In the past, classic generative models such as mixture models and graphical models were not very successful in learning the input data distribution for high-dimensional data due to complexity and scalability issues. With the power of deep learning, several deep generative models have been proposed to learn complex data distributions and have achieved tremendous success in many challenging domains. In the following, we summarize the three most common deep generative models in semi-supervised learning: Variational Autoencoders (VAEs) [Kingma and Welling, 2014], Normalizing Flows (NFs) [Kobyzev et al., 2021] and Generative Adversarial Networks (GANs) [Goodfellow et al., 2014].

2.3.1 VAEs

VAEs [Kingma and Welling, 2014] are deep generative models that can approximate inference and draw samples efficiently. They combine the idea of autoencoders and directed probabilistic models with variational inference. In particular, VAEs try to learn $-\log p_{\mathcal{X}}(x;\theta) = -\log \int_{\mathcal{Z}} p_{\mathcal{X}}(x|z;\theta)p_{\mathcal{Z}}(z)$. However, this integral is intractable. To avoid the intractability issue, variational inference is

adopted to approximate the posterior $q_{\mathcal{Z}}(z|x;\phi)$. Hence, the objective of VAEs is

$$\begin{aligned} L_{vae} &= \mathbb{E}_{z\sim q_{\mathcal{Z}}(z|x;\phi)}\left[\log q_{\mathcal{Z}}(z|x;\phi) - \log p_{\mathcal{Z}}(z) - \log p_{\mathcal{X}}(x|z;\theta)\right] \\ &\geq -\log p_{\mathcal{X}}(x;\theta) \end{aligned} \tag{2.1}$$

where $z \sim q_{\mathcal{Z}}(z|x;\phi)$ can be drawn and optimized by the "reparameterization trick".

VAEs consist of two components: the encoder and decoder parameterized by ϕ and θ, respectively. Intuitively, the encoder learns to encode the data x into a lower dimension distribution $q_{\mathcal{Z}}(z|x;\phi)$ in the latent space, while the decoder tries to recover the data x from $z \sim q_{\mathcal{Z}}(z|x;\phi)$ by minimizing the second term of Equation 2.1. The variational posterior distribution $q_{\mathcal{Z}}(z|x;\phi)$ is regularized to be similar to a pre-defined prior distribution $p_{\mathcal{Z}}(z)$ at the same time. For the objective Equation 2.1, the equality holds when $q_{\mathcal{Z}}(z|x;\phi) = p_{\mathcal{Z}}(z|x;\theta)$. The performance of VAEs highly depends on the tightness of the inequality Equation 2.1. Therefore, some works [Kingma et al., 2016] propose increasing the complexity of $q_{\mathcal{Z}}(z|x;\phi)$ to better approximate the true posterior $p_{\mathcal{Z}}(z|x;\theta)$.

Kingma et al. [2014] is one of the earliest works using conditional VAEs for semi-supervised learning. The objective of semi-supervised VAEs is to minimize the variational bound of $-\log p_{\mathcal{X}}(x,y;\theta)$ and $-\log p_{\mathcal{X}}(x;\theta)$ as a supervised and unsupervised loss, respectively. By learning the data distribution, the models can infer the label y based on the smoothness assumption. This means that if two points are close together, their labels should be the same.

$$\begin{aligned} L_{sup} &= \mathbb{E}_{q_{\mathcal{Z}}(z|x,y;\phi)}\left[\log q_{\mathcal{Z}}(z|x,y;\phi) - \log p_{\mathcal{Z}}(z) - \log p_{\mathcal{Y}}(y) - \log p_{\mathcal{X}}(x|y,z;\theta)\right] \\ &\geq -\log p_{\mathcal{XY}}(x,y;\theta) \end{aligned} \tag{2.2}$$

$$\begin{aligned} L_{vae} &= \mathbb{E}_{q_{\mathcal{Y}}(y|x;\phi)q_{\mathcal{Z}}(z|x,y;\phi)}[\log q_{\mathcal{Z}}(z|x,y;\phi) + \log q_{\mathcal{Y}}(y|x;\phi) - \log p_{\mathcal{Y}}(y) - \log p_{\mathcal{Z}}(z) \\ &\quad - \log p_{\mathcal{X}}(x|y,z;\theta)] \\ &\geq -\log p_{\mathcal{X}}(x;\theta) \end{aligned} \tag{2.3}$$

In this model, there are two cases: labelled data and unlabelled data. For labelled data, the model tries to optimize Equation 2.2, which is a simple extension of Equation 2.1 by introducing an additional variable y. For the unlabelled data, we treat the missing label as a latent variable for variational inference to have $q_{\mathcal{Y}}(y|x;\phi)$. Therefore, we can obtain the variational bound of $-\log p_{\mathcal{X}}(x;\theta)$ in Equation 2.3. As a result, the encoder can be used as a discriminative classifier by computing $\log q_{\mathcal{Y}}(y|x;\phi)$, while the model can also evaluate the upperbound of the log-likelihood of the data and generate conditional samples. Following Kingma et al. [2014], several works attempt to improve the performance of semi-supervised VAEs by increasing the expressiveness of the models. Maaløe et al. [2016] introduced additional auxiliary variables to VAEs, while Vahdat and Kautz [2020] introduced hierarchical structures in VAEs.

2.3.2 Normalizing Flows

Normalizing Flows [Kobyzev et al., 2021] are a family of deep generative models which can learn complicated data distributions. The main properties are that they can generate samples and evaluate the exact log-likelihood efficiently. Let's assume $\mathcal{X}$ and $\mathcal{Z}$ are the input space and latent space, respectively. A Normalizing Flow is a function $F : \mathcal{X} \mapsto \mathcal{Z}$ that utilizes the change of variables to transform a complex distribution $p_{\mathcal{X}}$ into a tractable distribution $p_{\mathcal{Z}}$ through an invertible transformation that allows computing the exact likelihood. Therefore, we can train Normalizing Flows by minimizing negative log-likelihood to approximate the target distribution $p_{\mathcal{X}}^{data}$.

$$L_{nll} = \mathbb{E}_{x\sim p_{\mathcal{X}}^{data}(x)}\left[-\log p_{\mathcal{X}}(x;\theta)\right] = \mathbb{E}_{x\sim p_{\mathcal{X}}^{data}(x)}\left[-\left(\log p_{\mathcal{Z}}(F_\theta(x)) + \log\left|\det\frac{\partial F_\theta(x)}{\partial x}\right|\right)\right] \tag{2.4}$$

where $p_{\mathcal{Z}}$ is a tractable probability density function, F is an invertible network parameterized by θ and $\frac{\partial F_\theta(x)}{\partial x}$ is the Jacobian of F. Since the typical choice of $p_{\mathcal{Z}}$ is Normal distribution, we call $F_\theta(\cdot)$ the normalizing direction.

To generate samples, we can draw samples $z \sim p_{\mathcal{Z}}$, and then apply the generator of the normalizing flow $G : \mathcal{Z} \mapsto \mathcal{X}$ to obtain the generated samples $x' = G_\theta(z)$. The generator of the normalizing flow is the inverse of the normalizing direction (i.e., $G_\theta = F_\theta^{-1}$) and we call $G_\theta(\cdot)$ the sampling or generating direction of a normalizing flow.

This formulation requires efficiently computing the inverse and Jacobian of transformations. Coupling layers [Dinh et al., 2017] are the most popular transformations that satisfy the two requirements. However, it is a simple transformation that limits the complexity, and the flows require compositing many coupling layers to increase the model complexity. Other studies [Ho et al., 2019, Kobyzev et al., 2021] introduce more complex transformations that can also mitigate the model complexity issue. See [Kobyzev et al., 2021] for a review.

Conditional distributions can be learned by conditional Normalizing Flows by introducing an additional class variable y to the flows: $-\log p_{\mathcal{X}}(x|y;\theta) = -\left(\log p_{\mathcal{Z}}(F_\theta(x,y)) + \log\left|\det \frac{\partial F_\theta(x,y)}{\partial x}\right|\right)$. Therefore, conditional Normalizing Flows can be adopted for semi-supervised learning by minimizing the negative log-likelihoods $\log p_{\mathcal{X}}(x,y;\theta)$ and $\log p_{\mathcal{X}}(x;\theta)$ as the supervised and unsupervised losses, respectively. The mixture distribution $p_{\mathcal{X}}(x,y;\theta)$ and marginal distribution $p_{\mathcal{X}}(x;\theta)$ are defined as $p_{\mathcal{X}}(x|y;\theta)p_{\mathcal{Y}}(y)$ and $\sum_y p_{\mathcal{X}}(x|y;\theta)p_{\mathcal{Y}}(y)$, respectively, where the prior distribution $p_{\mathcal{Y}}(y)$ can also be learnable.

$$L_{sup} = \mathbb{E}_{x,y\sim\mathcal{L}}\left[-\left[\log p_{\mathcal{X}}(x|y;\theta) + \log p_{\mathcal{Y}}(y)\right]\right] \tag{2.5}$$

$$L_{nll} = \mathbb{E}_{x\sim\mathcal{U}}\left[-\log p_{\mathcal{X}}(x;\theta)\right] = \mathbb{E}_{x\sim\mathcal{U}}\left[-\log \sum_y p_{\mathcal{X}}(x|y;\theta)p_{\mathcal{Y}}(y)\right] \tag{2.6}$$

By learning the data distribution, the hope is to infer the label y based on the smoothness assumption. The probability of the target class given the input $p_{\mathcal{Y}}(y|x;\theta)$ can be computed for classifica-

tion using the Bayes' rule:

$$p_{\mathcal{Y}}(y|x;\theta) = \frac{p_{\mathcal{X}}(x|y;\theta)p_{\mathcal{Y}}(y)}{\sum_y p_{\mathcal{X}}(x|y;\theta)p_{\mathcal{Y}}(y)}. \tag{2.7}$$

Izmailov et al. [2020], which is closely related to our work, adopts normalizing flows for semi-supervised learning. It parameterizes the distribution of the latent space $p_{\mathcal{Z}}$ as Gaussian Mixture Models and leaves the flow unconditional. Each Gaussian component corresponds to a class label. Therefore, the conditional distribution $\log p_{\mathcal{X}}(x|y;\theta)$ in 2.5 and 2.6 is the following:

$$\log p_{\mathcal{X}}(x|y;\theta) = \log \mathcal{N}(F_\theta(x); \mu_y, I) + \log \left| \det \frac{\partial F_\theta(x)}{\partial x} \right| \tag{2.8}$$

Since the underlying distribution in the latent space is a Gaussian Mixture Model, this model is called FlowGMM, and is the backbone model of our work. One of the advantages of FlowGMM is that we only require one forward pass to evaluate the summation of $\log p_{\mathcal{X}}(x;\theta)$ in 2.6 that can reduce the overall computation in training and testing.

Atanov et al. [2019] further improves the classification performance by conditioning the flow itself. However, it requires a number of forward passes equal to the number of classes to evaluate the summation of $p_{\mathcal{X}}(x;\theta)$ in Equation 2.6, vastly increasing the cost of training and evaluating. Therefore, Atanov et al. [2019] only specifies the last portion of the flow as conditional, called as semi-conditional Flows. As a result, only the conditional part requires the multiple passes for $\log p_{\mathcal{X}}(x;\theta)$ that can reduce the overall computations. It is important to understand that FlowGMMs are a specific type of semi-conditional Flows. The class-conditional affine transformation is applied only at the end of the flow, while the remainder of the flow is unconditional.

2.3.3 GANs

GANs [Goodfellow et al., 2014] have had success in generating high-resolution, plausible images [Karras et al., 2019, Brock et al., 2019]. They consist of two main components: a generator G and a discriminator D parameterized by θ and ϕ, respectively. The generator G is a network that transforms random noises to an output, which tries to approximate the target distribution. The discriminator represents the probability that the data came from the training distribution rather than the generator. The training paradigm is a minimax two-player game in which D learns to classify the output as either drawn from the actual target distribution or generated from G, while G tries to generate plausible data that can fool D by minimizing $\log(1 - D(G(z; \theta); \phi))$.

$$\min_{\theta} \max_{\phi} V(\theta, \phi) = \mathbb{E}_{x \sim p_{\mathcal{X}}^{data}(x)} \left[\log D(x; \phi)\right] + \mathbb{E}_{z \sim p_{\mathcal{Z}}(z)} \left[\log(1 - D(G(z; \theta); \phi))\right] \tag{2.9}$$

This objective is related minimizing Jensen-Shannon Divergence under some assumptions [Goodfellow et al., 2014]. Other objectives [Nowozin et al., 2016, Arjovsky et al., 2017] have been proposed to minimize different divergences or distance functions. In general, GANs are notoriously hard to train because we need to find the objective's saddle point. Furthermore, unlike VAEs or NFs, GANs are not able to evaluate the probability density.

Several previous works have incorporated adversarial training as an unsupervised loss in semi-supervised learning [Dai et al., 2017, Salimans et al., 2016]. With any standard classifier, we can include the generated samples from G in our training dataset, labeling them as a new class $y = K + 1$. Hence, besides learning to predict the correct label for labelled data, the model learns to classify the unlabelled data from one of the first K classes and the generated data from $K + 1$ class. The generator, at the same time, learns to generate samples that belong to one of the first K classes. This classifier can be viewed as a standard discriminator with an additional supervised objective on labelled data, and we call it the $(K + 1)$-class discriminator. The following equations are the objective of the $(K + 1)$-class discriminator D and the generator G parameterized by ϕ and

θ, respectively.

$$\min_{\phi} L_{sup_D} + L_{adv_D} \tag{2.10}$$

$$\min_{\theta} L_{adv_G} \tag{2.11}$$

where

$$L_{sup_D} = \mathbb{E}_{x,y\sim\mathcal{L}} \left[-y \log p_{\mathcal{K}}(y|x;\phi)\right] \tag{2.12}$$

$$L_{adv_D} = \mathbb{E}_{x\sim\mathcal{U}} \left[-\log p_{\mathcal{K}}(y \leq K|x;\phi)\right] + \mathbb{E}_{z\sim p_{\mathcal{Z}}(z)} \left[-\log p_{\mathcal{K}}(K+1|G(z;\theta);\phi)\right] \tag{2.13}$$

$$L_{adv_G} = \mathbb{E}_{z\sim p_{\mathcal{Z}}(z)} \left[-\log p_{\mathcal{K}}(y \leq K|G(z;\theta);\phi)\right] \tag{2.14}$$

where $\mathcal{K} = \{1, \ldots, K+1\}$ is the target space of the $(K+1)$-class discriminator D and $p_{\mathcal{K}}(\cdot|\mathcal{X};\phi)$ is the prediction probability over $K+1$ classes from D such that the first K classes are true target classes and the $(K+1)$-th class is the fake class. L_{sup_D} is the supervised loss applied on the labelled data only. The first term of L_{adv_D} is the adversarial loss for the discriminator to minimize the negative log probability of the unlabelled data belong to one of the target classes. The second term is to minimize the negative log probability of the $(K+1)$-th class (the fake class) for the generated data from the generator G. At the same time, to minimize the generator adversarial loss L_{adv_G}, the generator learns to generate samples that can make the discriminator believe that they belong to one of the K classes.

To evaluate the prediction probability for classification, we assume that the input data is real in general. Therefore, we compute $p_{\mathcal{Y}}(y|x, y \leq K;\phi)$, assuming the label of the input data always fall into one of the target labels.

$$p_{\mathcal{Y}}(y|x, y \leq K;\phi) = \frac{p_{\mathcal{K}}(y|x;\phi)}{p_{\mathcal{Y}}(y \leq K|x;\phi)}$$

FMGANs [Salimans et al., 2016] is the first work to adopt the $K+1$ class discriminator framework for SSL and realize that the generator objective $\mathbb{E}_{z\sim p_{\mathcal{Z}}(z)}\left[-\log p_{\mathcal{K}}(y \leq K|G(z;\theta);\phi)\right]$ yields a poor semi-supervised learning performance. Instead, Salimans et al. [2016] uses the feature matching objective 2.15 for training the generator, resulting in a significantly better semi-supervised learning performance under this framework:

$$\min_{\theta} L_{adv_G} = ||\mathbb{E}_{x\sim U}\left[h_D(x;\phi)\right] - \mathbb{E}_{z\sim p_{\mathcal{Z}}(z)}\left[h_D(G(z;\theta);\phi)\right]||_2^2 \tag{2.15}$$

where $h_D(\cdot;\phi)$ denotes activations of the last layer of the discriminator D. The feature matching objective is a sub-optimal objective for training the generator because it only matches the first moment of the last layer activations of the discriminator between real and generated data. As a result, the generated samples do not look visually appealing.

A follow-up work, BadGAN [Dai et al., 2017], provided a theoretical justification that using the sub-optimal objective, Equation 2.15, on the generator can yield a better SSL result for the $(K+1)$-class discriminator. Intuitively, if the generator can perfectly learn the data distribution, the $(K+1)$-class discriminator struggles to identify the real data from fake and the discriminator's unsupervised objective, Equation 4.4, gets to the optimal point, where both terms in L_{adv_D} are equal to a constant. Consequently, the discriminator will completely ignore the unsupervised objective L_{adv_D} since it has reached the optimal point. Therefore, it will only focus on the supervised objective, Equation 2.12, which reduces to the training of supervised learning on labelled data. They further justify that the "bad" samples from the sub-optimal generator can improve the generalization error of the $(K+1)$-class discriminator. Consider L_{adv_D} with training the discriminator with "bad" generated samples. Since "bad" generated samples lie outside the true data distribution, the output logits of the true classes are high for the data inside the true data distribution and low for the data outside the true data distribution. Thus, the decision boundaries of the $(K+1)$-class discriminator are pushed outside the true data distribution and lie in the low-density area, further

improve the classification error because of the low-density separation assumption. Therefore, they propose additional penalties on the feature matching objective to force the generator to produce diversified sub-optimal samples. Despite reaching state-of-the-art semi-supervised learning performance, BadGAN can only generate unconditioned samples with bad visual quality and cannot evaluate the data's log-likelihood.

Several other works extend the $K + 1$ class discriminator framework by focusing on the data manifold regularization. ALIs [Dumoulin et al., 2017] introduce an additional inference network to learn a useful representation of the data in the latent space. Later, InvarGANs [Kumar et al., 2017] further regularize the discriminator by injecting the prediction consistency against the small changes of the data manifold. Instead of learning the global data manifold, Localized GANs [Qi et al., 2018] use local coordinate charts to parameterize the local geometry of data transformations for each data point such that the discriminator encourages to be invariant against any local changes.

Li et al. [2017] also realize that the $(K + 1)$-class discriminator framework can only work well with the sub-optimal generator, but instead, they propose a new framework that allows the optimal generator and discriminator to coexist. First, they argue that the issue arises from the two-players formulation because the $K+1$ class discriminator has to play two incompatible roles: distinguishing fake samples and predicting the label of real samples. In particular, the discriminator cannot converge to optimal for both tasks at the same time. Therefore, they propose a three-player framework to address this issue. They introduce a conditional generator G, a discriminator D and an additional classifier C. The classifier is used for predicting the label given the any real or fake input. Then, the discriminator now solely classifies the real labelled data pair from fake pairs (i.e., either or both input and label are generated from G and C, respectively). As a consequence, when converging, the generator can generate good conditional samples matching the true data distribution, while the classifier can also have a good classification performance. However, like other GANs, it is unable to evaluate the log-likelihood of the data.

2.4 SSL Regularization Loss

Besides the unsupervised loss, we can also introduce regularizations to improve the SSL performance. The main difference between unsupervised loss and regularization loss is that unsupervised loss focuses on learning the input distribution, while regularization loss targets regularizing the model's predictions. More specifically, we want our model to be confident in its predictions under the low-density separation assumption and invariant to any irrelevant perturbation unrelated to the supervised task under the smoothness assumption. The regularization can be applied to both labelled and unlabelled data.

2.4.1 Entropy Minimization

Entropy Minimization [Grandvalet and Bengio, 2004] is a regularization method to enforce the low-density separation assumption where the decision boundaries should lie in the low-density area. Specifically, it encourages the model to have confident (i.e., low entropy) predictions on unlabelled data by minimizing the entropy of the model's predictions in Equation 2.16. In this case, the entropy minimization prevents the model's decision boundaries from passing through any high-density area where the model would otherwise make high entropy predictions for any unlabelled data near the decision boundaries, enforcing the low-density separation assumption. This method, however, can lead to a model being overconfident in its predictions, especially in a case where the model misclassifies the unlabelled data, resulting in poor performance and calibration.

$$L_{entmin} = \mathbb{E}_{x\sim\mathcal{U}}\left[-\sum_{i\in\mathcal{Y}} p_{\mathcal{Y}}(i|x;\theta)\log p_{\mathcal{Y}}(i|x;\theta)\right] \tag{2.16}$$

2.4.2 Consistency Regularization

A popular SSL regularization, consistency regularization, encourages the model to make consistent predictions for all labelled and unlabelled data based on some distance functions R under some pre-specified perturbations $\mathcal{T}$. It is based on the smoothness assumption such that a data point with

different perturbations should be close in the manifold and hence, their predictions should also be close.

$$L_{con} = \mathbb{E}_{x \sim \mathcal{L}+\mathcal{U}} \left[R(p_{\mathcal{Y}}(y|x;\theta,\mathcal{T}_1), p_{\mathcal{Y}}(y|x;\theta,\mathcal{T}_2)) \right] \tag{2.17}$$

where $\mathcal{T}$ is the perturbation method, and $R(\cdot,\cdot)$ is a distance function for minimization. The popular options of distance functions are Mean Square Error (MSE), Kullback-Leiber Divergence (KLD) and Jensen-Shannon Divergence (JSD).

All the consistency regularization methods differ in the choice of the perturbation methods. For instance, we can regularize the model to have consistent predictions against the stochasticity of data augmentations, Gaussian noise, dropout, network initialization or training algorithm. Virtual adversarial training (VAT) [Miyato et al., 2019] combines the concepts of the adversarial attack for consistency regularization. VAT proposes introducing an additional adversarial noise to perturb the input such that it could change the model's predictions. The perturbed inputs are used for consistency regularization to encourage the model to be consistent against such adversarial attacks.

$$L_{VAT} = \mathbb{E}_{x \sim \mathcal{L}+\mathcal{U}} \left[R(p_{\mathcal{Y}}(y|x;\theta), p_{\mathcal{Y}}(y|x+\gamma^{adv};\theta) \right] \tag{2.18}$$

where

$$\gamma^{adv} = \arg \max_{\gamma;||\gamma|| \leq \delta} R(p_{\mathcal{Y}}(y|x;\theta), p_{\mathcal{Y}}(y|x+\gamma;\theta)$$

where $\delta > 0$ is to constrain the length of the additional adversarial noises.

2.4.3 Pseudo-labelling

Pseudo-labelling method [Lee, 2013] is another regularization method that produces pseudo-labels from other pre-trained models for unlabelled data as additional labelled data for supervised train-

ing. Unlike the consistency regularization that relies on the consistency contraint against some perturbations, the pseudo-labelling methods rely on pseudo-labels with high confidence from another pre-trained model, enforcing the model to also have high confident predictions on unlabelled dataset. The performance of pseudo-labelling methods highly depend on the quality of the pseudo-labels which tend to be noisy and often incorrect. Therefore, several methods [Rizve et al., 2021, Pham et al., 2021] have been proposed to improve the quality of pseudo labelling.

2.4.4 Self-supervised learning

Self-supervised learning has recently had huge success and achieved the state-of-the-art results in semi-supervised learning. Self-supervised learning first requires defining a pretext task to pre-train the model, followed by finetuning with supervised learning. Therefore, self-supervised learning methods highly depend on the design of the pretext tasks which require the domain knowledge. S4L [Zhai et al., 2019] introduces a framework of the learning paradigm with self-supervised and semi-supervised learning. SimClr [Chen et al., 2020] introduces contrastive learning as a domain-agnostic pretext task for self-supervised learning. Specifically, contrastive learning aims to learn to minimize the distances of the latent space between the same data sample with different augmentations while maximizing the distances between two different samples. BYOL [Grill et al., 2020] and SimSiam [Chen and He, 2021], on the other hand, adopt similarity learning as another pretext task that can achieve competitive results without the need of negative pairs, which can highly reduce the computations.

2.5 Summary

This chapter provides an introduction to the various methods of semi-supervised learning, focusing on different unsupervised losses and regularizations. We started by discussing the three primary assumptions that are crucial for the success of SSL. Next, we presented the three major unsupervised losses (VAEs, NFs, and GANs) that are closely related to our work and are used as our baselines.

Lastly, we discussed different SSL regularizations for the sake of completeness.

Chapter 3

Semi-Supervised Learning with Negative Log-Likelihood

Generative models with negative log-likelihood, such as FlowGMMs [Izmailov et al., 2020], have shown some success in semi-supervised learning. They can leverage a large amount of unlabelled data to learn the data distribution. This approach can enable us to infer the class label of unseen data based on the smoothness assumption. However, we find that FlowGMMs suffer considerably in some tasks due to an initialization issue. In this chapter, we analyze the cluster collapse issue in FlowGMMs and suggest optimizing the supervised likelihood for our supervised loss to improve classification performance. We further address the curse of dimensionality issue with the multi-scale architecture in the latent space of FlowGMM.

3.1 GMM clusters collapse

In semi-supervised learning, low-density separation is often a key assumption where we assume all the decision boundaries should lie in a low-density region. Hence, learning the data distribution of additional unlabelled data can help to improve classification performance. FlowGMMs learn the data distribution by minimizing the negative log-likelihood of labelled and unlabelled data as

the supervised loss and unsupervised loss, respectively. Their objective is:

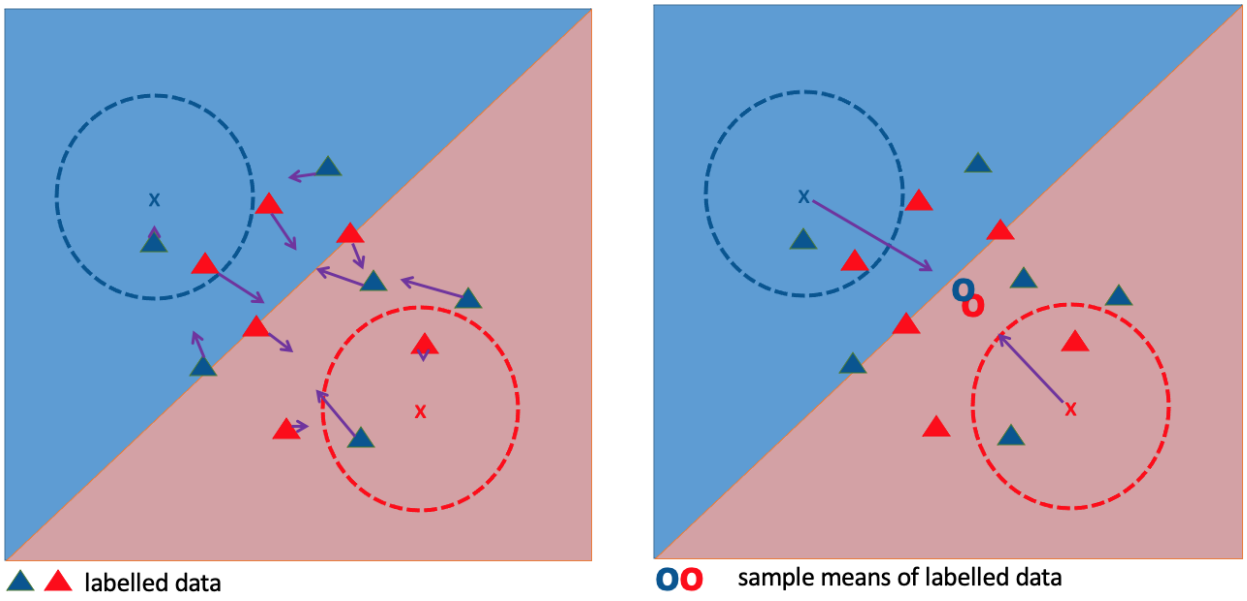
$$L = L_{sup} + L_{nll} \tag{3.1}$$

where

$$\begin{aligned} L_{sup} &= \mathbb{E}_{x,y\in\mathcal{L}}\left[-\log p_{\mathcal{XY}}(x,y;\theta)\right] \\ &= \mathbb{E}_{x,y\in\mathcal{L}}\left[-\log(p_{\mathcal{X}}(x|y;\theta)p_{\mathcal{Y}}(y))\right] \qquad (3.2) \\ L_{unsup} &= \mathbb{E}_{x\in\mathcal{U}}\left[-\log p_{\mathcal{X}}(x;\theta)\right] \\ &= \mathbb{E}_{x\in\mathcal{U}}\left[-\log\sum_{y\in\mathcal{Y}}(p_{\mathcal{X}}(x|y;\theta)p_{\mathcal{Y}}(y))\right] \qquad (3.3) \\ \log p_{\mathcal{X}}(x|y;\theta) &= \log\mathcal{N}(F_\theta(x);\mu_y,I) + \log\left|\det\frac{\partial F_\theta(x)}{\partial x}\right|. \qquad (3.4) \end{aligned}$$

Equation 3.2 and Equation 3.3 are the negative log-likelihood of labelled and unlabelled data, respectively. FlowGMMs transform the data distribution into a Gaussian Mixture Model in the latent space through an invertible function F_θ in Equation 3.4. Each GMM's cluster corresponds to a class label. In evaluation, FlowGMMs map the unseen data into the latent space and classify it based on its corresponding cluster. Their classification performance highly depends on the values of GMM parameters for satisfying the low-density separation assumption. If the clusters are too close, the decision boundary lies in a high-density area such that two points with different classes can connect with a path of high density. As a result, learning the input data distribution cannot provide any information about classification. On the other hand, if the clusters are too far away, it will be problematic because the model will assign high probabilities to all of its predictions which can cause overconfidence and calibration issues. Therefore, the GMM parameters in FlowGMMs are crucial to satisfy the low-density separation assumption for SSL performance.

Unfortunately, GMM parameters are very sensitive to the initialization. In classic Gaussian Mixture Models, it is well known that without careful initialization, the EM algorithm may lead to



(a) The loss gradient with respect to the labelled data in the latent space.

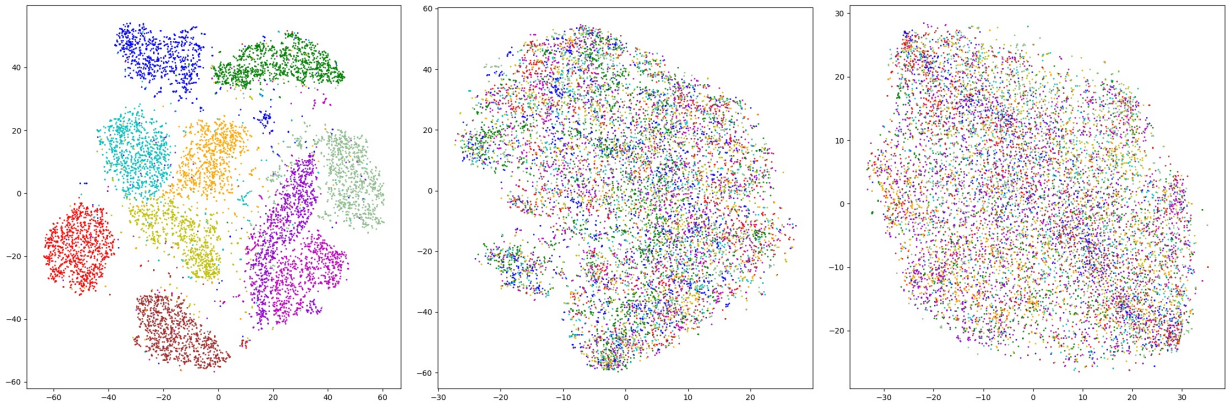
(b) The loss gradient with respect to the cluster means in the latent space.

Figure 3.1: An illustration of the gradient of $-\log p_{\mathcal{XY}}(x, y; \theta)$ on the labelled data and cluster means. The arrows, the triangles and the dotted circles represent the gradient directions, the labelled data and the clusters, respectively. Training FlowGMMs with $-\log p_{\mathcal{XY}}(x, y; \theta)$, the labelled data gets close to its corresponding cluster (left), while the cluster mean move toward the sample mean of the labelled data (right). Since there are no concrete clusters formed in the initialization, the clusters will collapse together.

poor local minima. Likewise, since the FlowGMMs' underlying distribution in the latent space is a GMM, it experiences a similar training issue. We illustrate this issue in Figure 3.1. We consider training a flow with SGD, and the objective function is negative log-likelihood. In Figure 3.1(left), the loss gradient with respect to the labelled data in the latent space points toward the corresponding cluster, so the labelled data in the latent space slowly moves toward its corresponding cluster. On the other hand, in Figure 3.1 (right), the loss gradient with respect to the cluster means points toward the center (sample mean) of its corresponding labelled data. Thus, if the labelled data isn't clearly clustered during initialization, the FlowGMM objective will encourage bringing the means of GMM clusters closer together. Once the clusters collapse together, separating them again is very difficult. In this situation, labelled data with different labels can connect together with a path of high density. Therefore, some labelled data can be close to incorrect clusters while still achieving

relatively low negative log-likelihood. As a result, the models do not satisfy the low-density separation assumption, leading to poor performance.

To see this in practice, we visualize the FlowGMM latent space of three widely used benchmark datasets: MNIST [LeCun and Cortes, 1998], SVHN [Netzer et al., 2011] or CIFAR10 [Krizhevsky, 2009] after initializations using the t-SNE plots [van der Maaten and Hinton, 2008] in Figure 3.2.



(a) MNIST (b) SVHN (c) CIFAR10

Figure 3.2: t-SNE graphs of FlowGMM's latent space at initialization.

We can clearly see that after the initialization, concrete clusters are formed in FlowGMM's latent space for each class of MNIST. However, there are no clusters for either SVHN or CIFAR10. Therefore, all the clusters may potentially get pulled toward the center during training. While the visualizations of the t-SNE plots are sometimes misleading [Wattenberg et al., 2016], we further demonstrate the issue by training FlowGMMs with the objective 3.1. Following the standard of the previous works [Oliver et al., 2018], we randomly draw 100, 1000, and 4000 labelled samples for MNIST, SVHN and CIFAR10, respectively, while leaving others as unlabelled data. Labelled and unlabelled data are drawn separately to optimize the first and second term of objective 3.1, respectively. The cluster means are initialized by K-Means in the latent space and trained by SGD along with the flow θ. In Table 3.1, the models struggle to classify even a small amount of the labelled training data correctly after 500 epochs in all three datasets (an epoch indicates one complete pass of the entire unlabelled dataset through the algorithm). The model cannot well-

separate the labelled training data implying that the clusters may be too close together even after training. To avoid this issue, Izmailov et al. [2020] sets the GMM parameters far away from each other and stops updating them after initialization. This is not an ideal solution because it is difficult to set a good initial value for the GMM parameters, but is somewhat effective.

Dataset (n_l/n_u)	Training Accuracy	Testing Accuracy
MNIST ($0.1k/59k$)	96.9	62.1
SVHN ($1k/72k$)	95.2	15.5
CIFAR10 ($4k/46k$)	75.3	13.1

Table 3.1: The SSL results of FlowGMMs on three benchmark datasets. n_l and n_u are the number of labelled and unlabelled training data on each dataset, respectively.

3.1.1 Supervised Likelihood

By minimizing the data likelihood $-\log p_{\mathcal{X},\mathcal{Y}}(x, y; \theta)$ as our supervised loss, the clusters in FlowGMM are prone to collapse in image datasets, as shown in the above section. Therefore, we need to encourage the clusters to push away from each other to satisfy the low-density separation assumption. Instead of minimizing the data likelihood, we suggest optimizing the supervised likelihood $-\log p_{\mathcal{Y}}(y|x; \theta)$ for our supervised loss:

$$
\begin{aligned}
-\log p_{\mathcal{Y}}(y|x;\theta) &= -\log \frac{p_{\mathcal{Z}}(F_\theta(x)|y)\left|\det \frac{\partial F_\theta(x)}{\partial x}\right| * p_{\mathcal{Y}}(y)}{\sum_{y'} p_{\mathcal{Z}}(F_\theta(x)|y')\left|\det \frac{\partial F_\theta(x)}{\partial x}\right| * p_{\mathcal{Y}}(y')} \\
&= -\log \frac{\mathcal{N}(f_\theta(x)|\mu_y, I) * p_{\mathcal{Y}}(y)}{\sum_{y'} \mathcal{N}(f_\theta(x)|\mu_{y'}, I) * p_{\mathcal{Y}}(y')}.
\end{aligned} \tag{3.5}
$$

Optimizing $-\log p_{\mathcal{Y}}(y|x; \theta)$ encourages the labelled data and the corresponding cluster to attract each other by maximizing the term of $\mathcal{N}(f_\theta(x)|\mu_y, I)$ in the numerator, while repulsing other clusters away by minimizing the terms of $\mathcal{N}(f_\theta(x)|\mu_{y'}, I)$ in the denominator. We illustrate the effect in Figure 3.3. The repulsive force will encourage the clusters to become well separated after training. On the other hand, optimizing $-\log p_{\mathcal{X}}(x; \theta)$ as the unsupervised loss can encourage the data

to move close to the high-density area in the latent space.

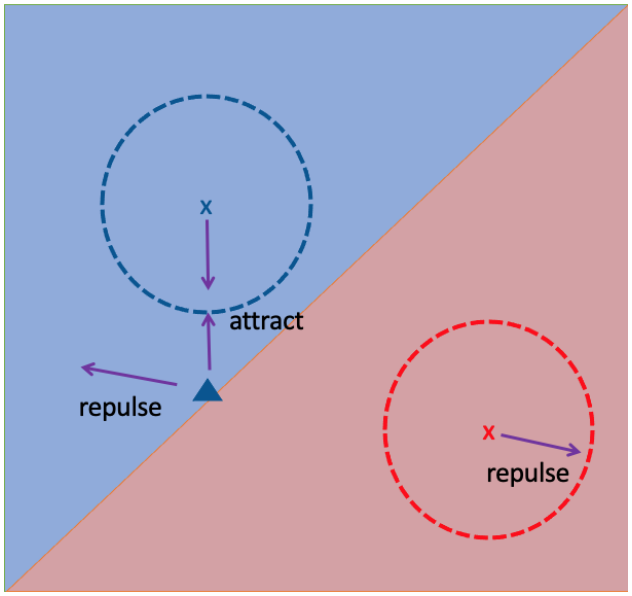


Figure 3.3: An illustration of the gradient of $-\log p_{\mathcal{Y}}(y|x;\theta)$ with respect to the labelled data and cluster means in the latent space. The arrows, the triangles and the dotted circles represent the gradient directions, the labelled data and the clusters, respectively.

Therefore, we modify the FlowGMM objective by the following. First, we replace the data likelihood $-\log p_{\mathcal{XY}}(x,y;\theta)$ with the supervised likelihood $-\log p_{\mathcal{Y}}(y|x;\theta)$ for our supervised loss as described above. Also, for the unsupervised loss term, we optimize the unsupervised likelihood $-\log p_{\mathcal{X}}(x;\theta)$ for both labelled and unlabelled data instead of solely unlabelled data.

$$\begin{aligned} L &:= w_{sup} * L_{sup} + w_{nll} * L_{nll} \\ &:= w_{sup} * \mathbb{E}_{x,y\in\mathcal{L}}\left[-\log p_{\mathcal{Y}}(y|x;\theta)\right] + w_{nll} * \mathbb{E}_{x\in\mathcal{L}+\mathcal{U}}\left[-\log p_{\mathcal{X}}(x;\theta)\right] \end{aligned} \tag{3.6}$$

By controlling the weights between these two terms, we can allow the clusters to move away from each other regardless of initialization. In Table 3.2, the models can reach 100.0% training accuracy for the labelled data with the supervised likelihood as the supervised loss. This results suggest that the supervised likelihood can avoid the cluster issue.

Dataset (n_l/n_u)	Training Accuracy	Testing Accuracy
MNIST ($0.1k/59k$)	100.0	93.1
SVHN ($1k/72k$)	100.0	71.9
CIFAR10 ($4k/46k$)	100.0	47.8

Table 3.2: The SSL results of FlowGMMs with supervised likelihood on three benchmark datasets. n_l and n_u are the number of labelled and unlabelled training data on each dataset, respectively.

3.2 The Curse of Dimensionality

Since the normalizing flow must be invertible, the dimensions between input and latent space are the same. While FlowGMMs define the distribution of the latent space as GMMs, the dimensionality of the latent space for classification is very high for images. Consequently, the model may suffer the curse of dimensionality, yielding poor SSL performance. To address this, we propose only adopting GMM as the underlying distribution for a portion of the latent space while leaving others as class independent Normal distribution. Specifically, we adopt the multi-scale architecture for normalizing flows [Dinh et al., 2017] and only specify the last scale of the latent space with the GMM. This results in forcing the class-related information to be in the last scale of the latent space, which contains most of the semantic information. Hence, only a portion of the latent space is used for classification, enforcing the manifold assumption.

3.3 Summary

In Sections 3.1 and 3.2, we discussed the problem of the original FlowGMM objective, which yielded poor SSL performance. We then proposed using the supervised likelihood and the multi-scale architecture to ensure that the model in the latent space satisfies the low-density assumption and the manifold assumption, respectively.

Chapter 4

Semi-Supervised Learning with the Hybrid Unsupervised Loss

With the above modifications, FlowGMMs are still not competitive with other discriminative methods in semi-supervised learning. In this chapter, we further analyze the impact of negative log-likelihood as an unsupervised loss in SSL. In particular, the models with negative log-likelihood severely underfit the areas near the decision boundaries, causing poor SSL performance. Therefore, we propose incorporating adversarial learning in the unsupervised loss such that the models can better learn the areas near the decision boundaries.

4.1 KL Divergence

Minimizing the negative log-likelihood is equivalent to minimizing an empirical estimate of the (forward) Kullback–Leibler divergence (Proposition 1). It is a popular method to learn a target distribution. The KL divergence is a type of divergence that measures how much one distribution differs from the target distribution. One of its main advantages is that we do not require the target distribution to compute the loss if we have access to draw samples from the target distribution. It is a typical scenario in practice where we only have the training dataset available, assuming drawn from the target distribution. The KL divergence is non-negative and reaches zero when

our generative model is identical to the target distribution. If the target model is in the family of our parametric generative model and sufficient data is available, minimizing the negative log-likelihood (or equivalent to minimizing the KL divergence) can guide our generative model to be identical to the target model. Unfortunately, it is impossible to ensure this in practice, especially, the normalizing flows require using restricted transformations (e.g. the coupling layers) for the sake of computational efficiency. With the limited memory budget, the KL divergence is always positive, which implies that our generative model can only approximate the target distribution. As a result, we need to understand the nature of the approximation.

4.2 Training Behaviour of KL Divergence

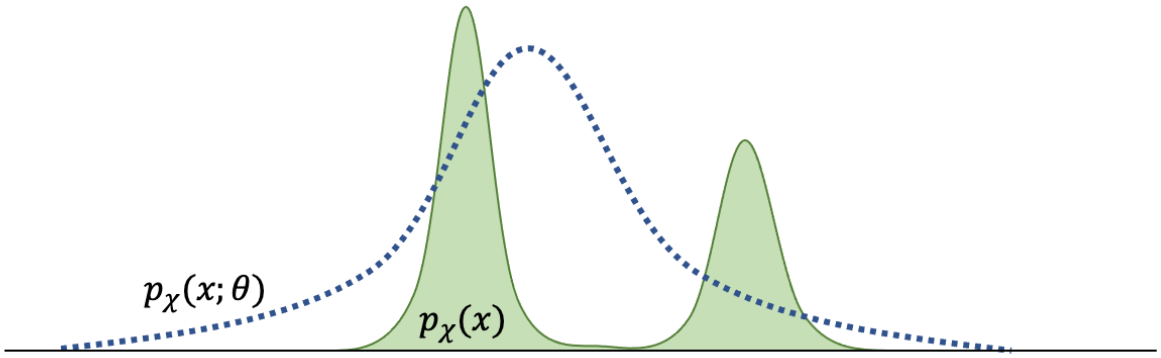


Figure 4.1: An illustration of the (forward) KL divergence. The parametric generative model (dotted line) is a single mode Gaussian distribution, while the target distribution (green mass) is a Gaussian Mixture Model with two modes. The resulting model trained with KL divergence assigns density to cover both modes of the target distribution.

Training with different divergence losses generally results in different generative models in practice. For instance, the forward KL Divergence encourages the resulting generative models to cover all the modes of the target distribution. As a result, models encourage assigning high density for all training data to satisfy the KL Divergence. On the other hand, KL Divergence does not penalize models giving excess density in low-density regions of the target distribution. That results in giving excess density in many regions to cover all the training data. The models typically ignore

low density regions of the target distribution and may incorrectly put significant mass in regions where the target distribution has little or no density, like in Figure 4.1. Consequently, this leads to a specific artifact when the models underfit the target distribution. We conjecture that such artifact due to the forward KL Divergence can hurt the SSL performance. In SSL, we assume that the target distribution follows the low-density separation assumption, which means the true decision boundaries lie in the low-density regions. Since models trained with KL divergence typically ignore the low-density regions of the target distribution in practice, the model may severely underfit in the areas near the decision boundaries, resulting in poor SSL performance.

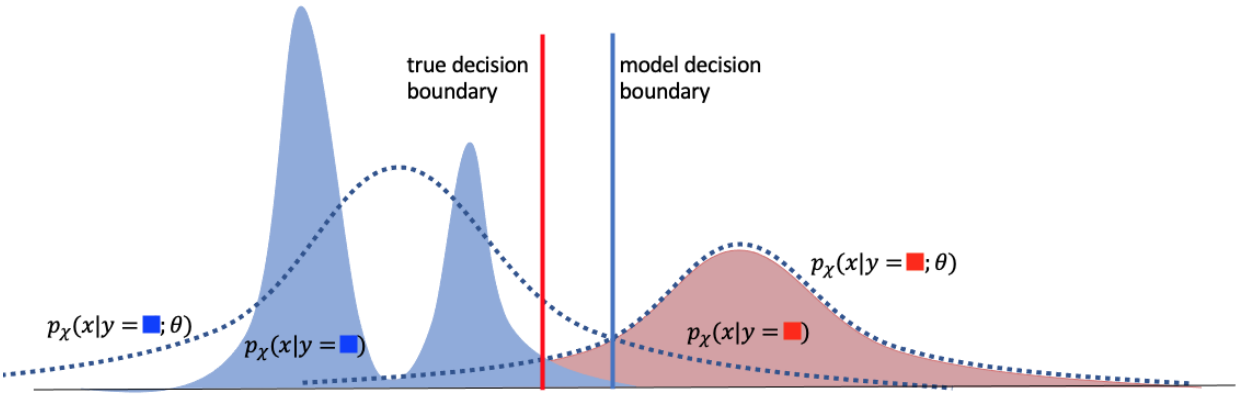


Figure 4.2: An illustration of the generative model trained with the KL divergence. The parametric conditional generative model (dotted line) is a single mode Gaussian distribution. The blue class and red class target distributions are a Gaussian Mixture Model with two modes and single mode, respectively. While the resulting model trained with KL divergence (dotted line) can fit the red class target distribution, it underfits the blue class target distribution due to the complexity issue. This results in a very different decision boundary (vertical line).

We provide an example that training with KL divergence leads poor decision boundaries. In Figure 4.2, our red class conditional generative model (a single-mode Gaussian distribution) has enough capacity to fit the red class target distribution (a single-mode Gaussian distribution) perfectly. On the other hand, our blue class conditional generative model (a single-mode Gaussian distribution) does not have enough capacity to fit the blue class target distribution, which is a double-mode GMM. As a result, the model has a heavier right tail than the blue class target distribution due to the side effect of KL divergence. The model decision boundary (the blue vertical line) shifts to the

right, mismatching the true decision boundary (the red vertical line). Figure 4.2 is a typical example in practice in which multiple modes exist within a class, while our generative model cannot perfectly fit the target distribution.

4.3 Training Behaviour of Jensen-Shannon Divergence

In the last section, we show that the KL divergence leads to poor SSL performance because it fails to learn the low-density regions where the decision boundaries lie. To improve the SSL performance, we propose considering other divergences that can better fit the low-density regions of the target distribution. Jensen-Shannon Divergence is a good choice that can take care of matching both the high-density and low-density regions between the generative models and target distribution.

$$\mathbb{E}_{x\sim p^{data}(x)}\left[\log \frac{p^{data}(x)}{m_{\mathcal{X}}(x)}\right] + \mathbb{E}_{x\sim p(x;\theta)}\left[\log \frac{p(x;\theta)}{m_{\mathcal{X}}(x)}\right] \quad \text{(JS divergence)}$$

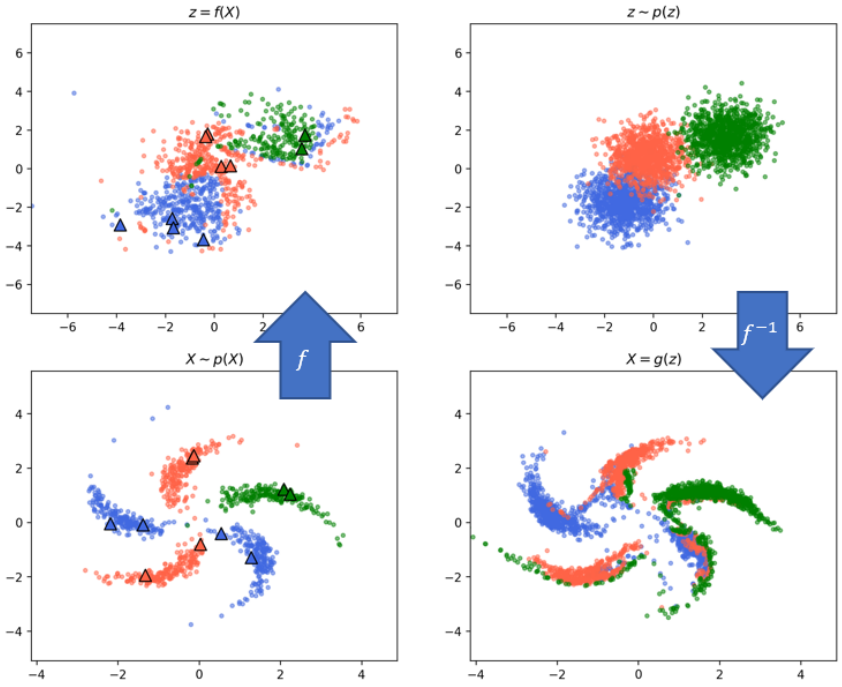
where $m_{\mathcal{X}}(x) = \frac{p^{data}(x)+p(x;\theta)}{2}$.

The first term is similar to the KL divergence that the generative model learns to match the density of the target distribution for each training data that can encourage mode-covering. Alternatively, the second term can penalize the model for allocating extra density in the regions where the target distribution has low or no density. It encourages the model to match the target distribution for each point drawn from the model. Therefore, even though the training dataset has no data to represent the regions of the low-density area, this term can still force the model to better match the target distribution if the model assigns high density in those areas.

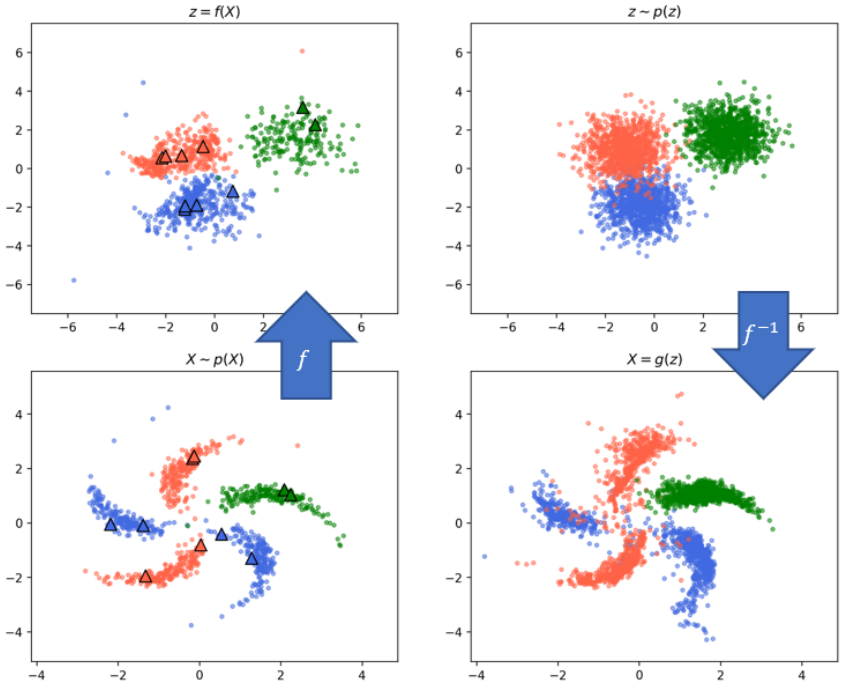
4.4 Adversarial Training

Unfortunately, JS Divergence is intractable in practice because the target distribution $p^{data}(x)$ is unknown. Adversarial training, first proposed by Goodfellow et al. [2014], is an approach for training a generative model. It introduces an additional network called the discriminator, which is trained to classify the data either from the target distribution or the generative model. The output value from the discriminator can be viewed as a pseudo loss for training the generative model to generate samples similar to the target distribution. Its objective is equivalent to Jensen-Shannon Divergence if the discriminator is optimal for any generative model [Goodfellow et al., 2014]. Hence, adversarial training can be a good estimator of JS Divergence. In fact, it is a variational lower bound of Jensen-Shannon Divergence [Nowozin et al., 2016]. While minimizing the variational lower bound may be a concern, we empirically found that the adversarial training can improve the generative model's SSL performance. We further found that the combination of negative log-likelihood and adversarial training produces the best performance.

In the pinwheel toy example [Pedregosa et al., 2011], with negative log-likelihood as the unsupervised loss (Figure 4.3a), we can see that the model fails to transfer data into three concrete clusters. Also, the generated samples do not match with the target distribution such that different class samples locate in the incorrect pin. We can also see that the generated samples form a path to connect two pins together. This artifact may be due to the mode-covering property of negative log-likelihood that causes the model to assign extra density in the low-density areas of the target distribution. With the hybrid unsupervised loss (Figure 4.3b), the model succeeds in mapping the data into three clusters, and the generated samples better match the target distribution.



(a) with nll only



(b) with the hybrid unsupervised loss

Figure 4.3: Illustration of FlowGMM SSL performance with different unsupervised loss on pinwheel toy dataset [Pedregosa et al., 2011]. Triangles and dots represent labelled and unlabelled data, respectively.

4.5 $K+1$-class Discriminator

Instead of using the ordinary discriminator to classify the data between real and fake, we apply the $K+1$-class discriminator framework, which has had success in SSL literature [Dai et al., 2017, Salimans et al., 2016]. While previous works utilize the $K+1$-class discriminator as their discriminative classifier, we focus on whether adversarial training improves the generative model's SSL performance. Assume the $K+1$-class discriminator and the generative model (i.e., the conditional normalizing flow) are parameterized by ϕ and θ, respectively:

$$\min_{\phi} L_{sup_D} + L_{adv_D} \tag{4.1}$$

$$\min_{\theta} L_{adv_G} \tag{4.2}$$

with

$$L_{sup_D} = \mathbb{E}_{x,y\sim\mathcal{L}}\left[-y \log p_{\mathcal{K}}(y|x;\phi)\right] \tag{4.3}$$

$$L_{adv_D} = \mathbb{E}_{x\sim\mathcal{U}}\left[-\log p_{\mathcal{K}}(y \leq K|x;\phi)\right] + \mathbb{E}_{x\sim p_{\mathcal{X}}(x;\theta)}\left[-\log p_{\mathcal{K}}(K+1|x;\phi)\right] \tag{4.4}$$

$$L_{adv_G} = \mathbb{E}_{x,y\sim p_{\mathcal{X}}(x|y;\theta),p_{\mathcal{Y}}(y)}\left[-\log p_{\mathcal{K}}(y|x;\phi)\right] \tag{4.5}$$

where $\mathcal{K} = \{1,\ldots,K+1\}$ is the target space of the $(K+1)$-class discriminator D and $p_{\mathcal{K}}(\cdot|\mathcal{X};\phi)$ is the prediction probability over $K+1$ classes from D such that the first K classes are true target classes and the $(K+1)$-th class is the fake class. The conditional normalizing flow as the generative model is differentiable in the sampling direction $G_\theta : \mathcal{Z} \mapsto \mathcal{X}$ such that $x = G_\theta(z) = F_\theta^{-1}(z)$ where $z \in p_{\mathcal{Z}}(z|y)$. The adversarial loss for the generative model is different from other works such that we minimize $\mathbb{E}_{x,y\sim p_{\mathcal{X}}(x|y;\theta),p_{\mathcal{Y}}(y)}\left[-\log p_{\mathcal{K}}(y|x;\phi)\right]$ instead of $\mathbb{E}_{x\sim p_{\mathcal{X}}(x;\theta)}\left[-\log p_{\mathcal{K}}(y \leq K|x;\phi)\right]$ because we are able to generate conditional samples from our conditional normalizing flow. Therefore, it not only guides the generative model to produce realistic samples but also encourages the model to generate samples matching its generated target class. Note that the L_{adv_D} is the same objective as the ordinary GAN objective such that $p_{\mathcal{K}}(y \leq K|x;\phi)$

is the probability of the data drawn from the target distribution while $p_{\mathcal{K}}(K+1|x;\phi)$ is the probability of the data drawn from the generative model. Therefore, L_{adv_D} is also equivalent to the Jensen-Shannon divergence if the discriminator is optimal for any generative model [Goodfellow et al., 2014]. However, in practice, this criterion is impossible to satisfy, so GANs are notoriously difficult to train. Therefore, including the negative log-likelihood as the unsupervised loss can compensate for the challenges of adversarial training.

4.6 Adversarial Training Near Decision Boundaries

The goal of adversarial training is to allow the generative model to better learn the low-density regions of the target distribution. If the decision boundaries indeed lie in the low-density areas of the target distribution, adversarial training, as an unsupervised loss, should improve SSL performance. The objective 4.5 encourages the model to match the target distribution in the areas where the samples are drawn. Thus, the model will allocate its capacity to learn the target distribution around the center of each cluster in the latent space. To further improve SSL performance, we suggest enforcing the model to focus on learning the target distribution near the decision boundaries. Since we know the location of the decision boundaries of FlowGMMs, we can instead draw samples near the decision boundaries for adversarial training. Therefore, we propose the following sampling method $\mathcal{G}$ for drawing samples near the decision boundary.

$$
\begin{aligned}
a, b &\sim Cat(K) \text{ where } a \neq b, \\
z_a, z_b &\sim \mathcal{N}(\mu_a, I), \mathcal{N}(\mu_b, I) \\
z' &= \beta * z_a + (1-\beta) * z_b \\
x' &= G_\theta(z') = F_\theta^{-1}(z') \\
y' &= [y'_1, \ldots, y'_K] = \left[\frac{\mathcal{N}(z'; \mu_i, I)}{\sum_j \mathcal{N}(z'; \mu_j, I)} \right]_{i \in \{1,\ldots,K\}}
\end{aligned}
\tag{4.6}
$$

where $\beta \in [0, 1]$ is a hyperparameter of the interpolation weight, y' is a posterior probability of the class given the latent value z' and all the elements of y' are sum to one, $\sum_i y'_i = 1$. Specifically, we randomly draw two points from two different components of the GMM. Then, we linearly interpolate between the two points. The interpolated point should lie near the decision boundary in the latent space. Finally, we apply the inverse of the flow to the interpolated point for generating the sample. With the proposed sampling method $\mathcal{G}$, we introduce the revised adversarial objective:

$$L^{int}_{adv_D} = \mathbb{E}_{x\sim\mathcal{U}}\left[-\log p_{\mathcal{K}}(y \leq K|x;\phi)\right] + \mathbb{E}_{x'\sim\mathcal{G}}\left[-\log p_{\mathcal{K}}(K+1|x';\phi)\right] \tag{4.7}$$

$$L^{int}_{adv_G} = E_{x',y'\sim\mathcal{G}}\left[-\sum_{k=1}^{K} y'_k \log p_{\mathcal{K}}(y'_k|x';\phi)\right]. \tag{4.8}$$

The revised adversarial objective for the discriminator (Equation 4.7) is equivalent to the previous one (Equation 4.4) with fake samples drawn from the proposed sampling method $\mathcal{G}$. On the other hand, the revised adversarial objective for the generative model (Equation 4.8) encourages the generative model to generate realistic samples matching its soft pseudo label y'. Essentially, the generative model focuses on learning the target distribution near the decision boundaries in order to improve SSL performance.

4.7 Pseudo-labeling

Our method has two components: the discriminator and the conditional normalizing flow, which both can produce the probability of a class given the input. Thus, each component can provide the pseudo-label on unlabelled data for the other. The pseudo-labelling method, a popular SSL method, relies on pseudo-labels with high confidence from another model, encouraging both models to have the consistent predictions on the unlabelled dataset. This can further improve the SSL performance in practice. Here we use the discriminative and generative classifiers to provide pseudo labels for

each other. We regularize the prediction probabilities of both models by using mean square error:

$$\min_{\theta,\phi} L_{PL} = \mathbb{E}_{x\sim\mathcal{U}}\left[\frac{1}{K}\sum_{y=1}^{K}\left[p_{\mathcal{K}}(y|x, y \leq K;\phi) - p_{\mathcal{Y}}(y|x;\theta)\right]^2\right] \tag{4.9}$$

4.8 Summary

In this chapter, we first demonstrated that training models with negative log-likelihood cause poor SSL performance due to the mode-covering property. Therefore, we proposed adopting a hybrid of generative objectives, including adversarial learning and negative log-likelihood, to train the models, improving the SSL performance. We also incorporated a popular SSL technique, pseudo-labelling, to enhance performance. we summarize the final SSL objective of the $K+1$-class discriminators and the conditional normalizing flows with the hybrid unsupervised loss in Table 4.1 and 4.2, respectively.

The final $(K+1)$-class discriminator D objective $\min_\phi w_{sup_D} * L_{sup_D} + w^{int}_{adv_D} * L^{int}_{adv_D} + w_{PL_D} * L_{PL_D}$	
L_{sup_D}	$\mathbb{E}_{x,y\sim\mathcal{L}}\left[-y \log p_{\mathcal{K}}(y|x;\phi)\right]$ The supervised loss for labelled data
$L^{int}_{adv_D}$	$\mathbb{E}_{x\sim\mathcal{U}}\left[-\log p_{\mathcal{K}}(y \leq K|x;\phi)\right] + \mathbb{E}_{x'\sim\mathcal{G}}\left[-\log p_{\mathcal{K}}(K+1|x';\phi)\right]$ The adversarial loss for real data (unlabelled data) against generated data with the interpolated sampling method $\mathcal{G}$
L_{PL_D}	$\mathbb{E}_{x\sim\mathcal{U}}\left[\sum_{y=1}^{K} R(p_{\mathcal{K}}(y|x, y \leq K;\phi), p_{\mathcal{Y}}(y|x;\theta))\right]$ The supervised loss with pseudo label for unlabelled data

Table 4.1: The summary of the $(K+1)$-class discriminator D objective.

The final conditional normalizing flow objective $\min_{\theta,\mu_y,p_{\mathcal{Y}}(y)} w_{sup_G} * L_{sup_G} + w_{nll} * L_{nll} + w^{int}_{adv_G} * L^{int}_{adv_G} + w_{PL_G} * L_{PL_G}$	
L_{sup_G}	$\mathbb{E}_{x,y\in\mathcal{L}}\left[-\log p_{\mathcal{Y}}(y\|x;\theta)\right]$ The supervised likelihood with threshold for labelled data
L_{nll}	$\mathbb{E}_{x\sim\mathcal{L}+\mathcal{U}}\left[-\log p_{\mathcal{X}}(x;\theta)\right]$ The negative log-likelihood for both labelled and unlabelled data
$L^{int}_{adv_G}$	$\mathbb{E}_{x',y'\sim\mathcal{G}}\left[-\sum_k y'_k \log p_{\mathcal{K}}(y'_k\|x';\phi)\right]$ The adversarial loss for generated data with the interpolated sampling method $\mathcal{G}$
L_{PL_G}	$\mathbb{E}_{x\sim\mathcal{U}}\left[\frac{1}{K}\sum_{y=1}^{K}\left[p_{\mathcal{K}}(y\|x, y\leq K;\phi) - p_{\mathcal{Y}}(y\|x;\theta)\right]^2\right]$ The supervised loss with pseudo label for unlabelled data

Table 4.2: The summary of the conditional normalizing flow objective.

Chapter 5

Experiments

Our goal is to evaluate semi-supervised learning performance of generative models with hybrid unsupervised objectives on three widely used benchmark datasets, namely MNIST [LeCun and Cortes, 1998], SVHN [Netzer et al., 2011], and CIFAR10 [Krizhevsky, 2009]. We also perform ablation studies to learn the effect of each component of the final objective.

5.1 Experimental Protocol

Following the standard SSL protocol [Oliver et al., 2018], we randomly sample 100, 1000 and 4000 samples as labelled datasets for MNIST, SVHN and CIFAR10, respectively, and leave the rest as unlabelled data, and use the standard data splits for evaluation. The labelled data are chosen to have a balanced number of examples for each class. The results are reported with the average of five different runs. In all experiments, our generative models follow the same network architecture as in FlowGMMs. They adopt RealNVP as the normalizing flow architecture and Gaussian Mixture Model as the underlying prior distribution. Our $K+1$ class discriminators follow the same network architecture as in FMGANs. The weights of the final objective in Section 4.8 are tuned based on the separated validation set for MNIST. The same set of weights is used for SVHN and CIFAR10. The model architectures and training details are in Appendix A and B.

5.2 Main Results

Models	Dataset (n_l/n_u)		
	MNIST ($0.1k/59k$)	SVHN ($1k/72k$)	CIFAR-10 ($4k/46k$)
Generative classifiers			
M1+M2 VAE SSL Kingma et al. [2014]	96.7	64.0	-
SDGM Maaløe et al. [2016]	98.7	83.4	-
ADGM Maaløe et al. [2016]	99.0	77.1	-
FlowGMM Izmailov et al. [2020]	91.7	67.2	42.1
Semi-cond Flow Atanov et al. [2019]	98.1	-	-
Our model (NF generator f)	**99.1**	**93.2**	**73.6**
Discriminative classifiers			
CatGAN Springenberg [2016]	98.1	-	80.4
Ladder network Rasmus et al. [2015]	98.9	-	79.6
FM GAN Salimans et al. [2016]	99.1	91.9	81.4
Bad GAN Dai et al. [2017]	**99.2**	**95.7**	**85.6**
Our model ($(K+1)$-class discriminator D)	99.1	94.0	81.9

Table 5.1: Accuracy of state-of-the-art methods on three benchmark datasets. n_l and n_u are the number of labelled and unlabelled training data on each dataset, respectively.

We compare the results of our model with state-of-the-art SSL methods on the three benchmark datasets in Table 5.1. Since our model has two components, we report the results with two separate categories: generative classifier and discriminative classifier. Our generative model outperforms all generative classifiers including FlowGMMs by a large margin in all three datasets and becomes competitive with discriminative classifiers in MNIST and SVHN but CIFAR10. Since CIFAR10 is a relatively challenging dataset that contains many variations within classes, we conjecture that the generative classifier is not sufficient to learn the data distribution. On the other hand, our discriminative component is competitive with other discriminative classifier methods, and is always better than our generative component. BadGANs perform the best among other classifiers. However, they are discriminative classifiers, which perform better than generative classifiers in general. On the other hand, our method improves the generative classifiers' performance, which is competitive with other discriminative classifiers. Furthermore, unlike discriminative classifiers, it enables us

to perform conditional sampling and likelihood estimation. We provide some conditional samples for MNIST (Figure 5.1), SVHN (Figure 5.2) and CIFAR10 (Figure 5.3) drawn from our models. The samples have a variety of styles with good quality for each class.

Figure 5.1: Conditional samples of the model trained with MNIST.

Figure 5.2: Conditional samples of the model trained with SVHN.

Figure 5.3: Conditional samples of the model trained with CIFAR10.

5.3 Ablation Studies

Models (generative classifier)	Dataset (n_l/n_u)		
	MNIST ($0.1k/59k$)	SVHN ($1k/72k$)	CIFAR-10 ($4k/46k$)
supervised loss only	80.7	63.3	45.7
+ nll only	93.1	71.9	47.8
+ adv loss only		unstable	
+ nll + adv loss	96.1	80.2	60.7
+ nll + adv loss + pseudo labels	99.1	93.5	73.5
FlowGMM	91.7	67.2	42.1

Table 5.2: Accuracy with different component of our model objective.

We analyze the effect of each component of our model objective in our ablation studies in Table 5.2. Our baseline is trained with the supervised objective on labelled data only. First, we incorporate the unlabelled data by minimizing the negative log-likelihood. Its performance is better than the baseline. This result suggests that learning the distribution from additional unlabelled data can improve the SSL performance. With the adversarial loss as the only unsupervised loss, the SSL results become unstable. We conjecture that the adversarial loss does not enforce the model to map the labelled and unlabelled data to one of the GMM clusters in the latent space. Therefore, the model is unable to make a good prediction in the latent space. By incorporating the hybrid generative loss, the negative log-likelihood and the adversarial loss, the SSL performance is better than the model with the negative log-likelihood. This result aligns with our analysis that the negative log-likelihood fails to learn the low-density regions of the target distribution. Since we assume the true decision boundaries lie in the low-density areas, the model will obtain poor decision boundaries. With the additional adversarial loss, the model becomes capable of learning the low-density regions of the target distribution. Finally, our generative model attains the best outcomes by utilizing the pseudo labels.

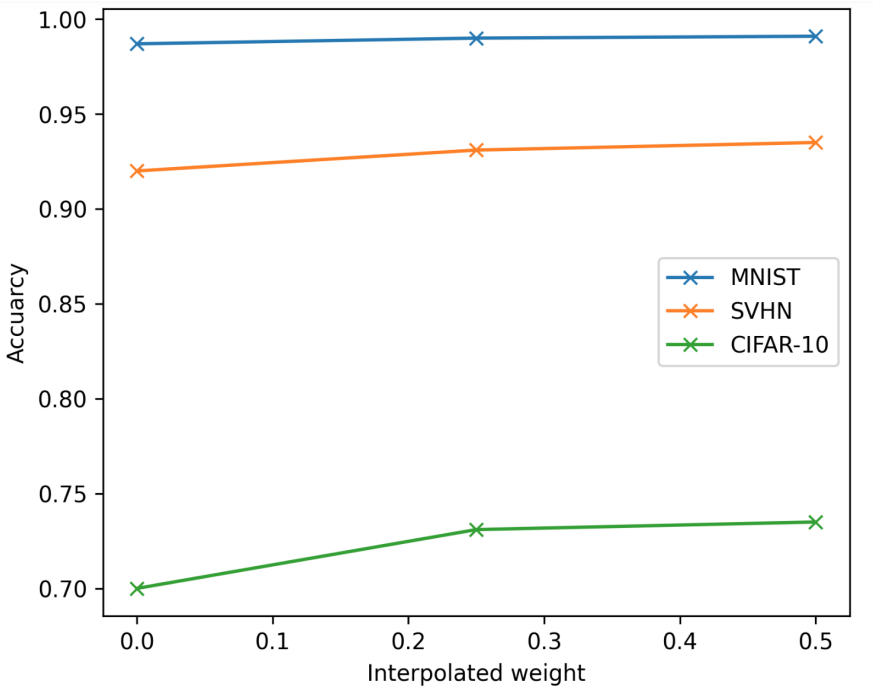


Figure 5.4: The SSL performance with different interpolated weights.

We further study the interpolated weight β of the interpolated sampling method 4.6 in Figure 5.4. In all three datasets, the performance improves slightly when the interpolated weight β increases. It performs best with β equal to 0.5 such that samples drawn for adversarial training should lie near the decision boundaries. We conjecture that it encourages the model to focus on learning the target distribution near the region of decision boundaries, resulting in a better performance.

5.4 Summary

This chapter demonstrated that our method achieves state-of-the-art performance compared to other generative classifiers. Although the performance is still slightly worse than other discriminative classifiers, it can allow us to generate class-conditional samples with good quality. Through our ablation studies, we examined each component of our model and found that they effectively enhance our backbone models, FlowGMMs.

Chapter 6

Conclusion

We proposed a new semi-supervised learning method with normalizing flows by introducing a hybrid generative method that combines minimum negative log-likelihood and adversarial learning. Generative models offer advantages such as sampling and density estimation, but they often perform worse in semi-supervised learning than discriminative models. FlowGMMs, SSL models with normalizing flows, are the backbone models of our method. We showed that FlowGMMs suffer considerably in some datasets due to the initialization issue. To address this issue, we modified the FlowGMMs' objective such that it no longer gets stuck on a bad local minima regardless of the initialization. Further, we indicated that negative log-likelihood as an unsupervised loss has a bad impact on SSL performance due to the mode-covering property. Models trained with negative log-likelihood are not sensitive to about the low-density regions of the target distribution in practice. The model may severely underfit the areas near the decision boundaries, resulting in poor SSL performance. Therefore, we proposed to include adversarial learning in our unsupervised loss so that the models can better fit the low-density areas of the target distribution, resulting in a hybrid generative model. We empirically demonstrated that our method yields a competitive SSL performance while preserving the advantages of generative models, and we performed ablation studies to further understand the effect of each component of our method.

6.1 Limitations

Although our SSL method outperforms other generative classifiers, it has an additional adversarial training term compared to the baseline model, FlowGMM, that requires searching the optimal weight of the loss. This is problematic in SSL because we usually assume only a limited amount of labelled data. Therefore, we may not have the luxury of obtaining a hold-out validation set for hyperparameter searching. Cross-validation may be a viable solution, but it is insufficient and incurs a substantial computational cost. In our experiments, we found that our method works best for all three benchmark datasets with the set of hyperparameter $w_{sup_G} = 1.0, w_{nll} = 0.01 * \frac{1}{dim}, w^{int}_{adv_G} = 0.01$. Thus, this hyperparameter set can be a good starting point for other dataset.

Likelihood-based generative models allow us to estimate the density of the data, which can be used for out-of-distribution (OOD). Although there is literature that likelihood alone is insufficient for determining whether data is out of distribution [Nalisnick et al., 2019], some state-of-the-art OOD methods [Choi et al., 2018, Havtorn et al., 2021] rely on pre-trained likelihood-based generative models. However, with additional supervised learning and adversarial training, our method does not obtain competitive results in density estimation, which may potentially hurt the OOD performance.

6.2 Future Work

As we demonstrated that a hybrid unsupervised loss could improve SSL performance, testing the impact of different generative model objectives such as f-GANs [Nowozin et al., 2016] and Wasserstein GANs [Arjovsky et al., 2017] can be an interesting direction for future work. Also, we can try expanding the complexity of the generative model. For instance, our current method defines the number of clusters in GMM as equal to the total classes. Therefore, one single cluster in the latent space must represent all the data in one class. It can be very challenging, especially in CIFAR10, where each class has wide varieties. Therefore, to further improve the SSL performance,

we could expand the number of clusters of GMM in the latent space. This change can increase the complexity of the latent space distribution such that more than one cluster can associate with the same class.

Bibliography

L. Ardizzone, R. Mackowiak, C. Rother, and U. Köthe. Training Normalizing Flows with the Information Bottleneck for Competitive Generative Classification. In *Neural Information Processing Systems*, 2020.

M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein Generative Adversarial Networks. In *International Conference on Machine Learning*, 2017.

A. Atanov, A. Volokhova, A. Ashukha, I. Sosnovik, and D. Vetrov. Semi-Conditional Normalizing Flows for Semi-Supervised Learning. In *ICML Workshop on Invertible Neural Nets and Normalizing Flows, International Conference on Machine Learning*, 2019.

A. Brock, J. Donahue, and K. Simonyan. Large Scale GAN Training for High Fidelity Natural Image Synthesis. In *International Conference on Learning Representations*, 2019.

O. Chapelle, B. Schölkopf, and A. Zien. *Semi-Supervised Learning*. MIT Press, 2006.

T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A Simple Framework for Contrastive Learning of Visual Representations. In *International Conference on Machine Learning*, 2020.

X. Chen and K. He. Exploring Simple Siamese Representation Learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

H. Choi, E. Jang, and A. A. Alemi. WAIC, but Why? Generative Ensembles for Robust Anomaly Detection. In *arXiv preprint, arXiv:1810.01392*, 2018.

Z. Dai, Z. Yang, F. Yang, W. W. Cohen, and R. Salakhutdinov. Good Semi-Supervised Learning that Requires a Bad GAN. In *Neural Information Processing Systems*, 2017.

L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density Estimation using Real NVP. In *International Conference on Learning Representations*, 2017.

V. Dumoulin, I. Belghazi, B. Poole, O. Mastropietro, A. Lamb, M. Arjovsky, and A. Courville. Adversarially Learned Inference. In *International Conference on Learning Representations*, 2017.

M. F. A. Farouk, F. Schwenker, and G. Palm. Semi-Supervised Learning for Regression with Co-training by Committee. In *International Conference on Artificial Neural Networks*, 2009.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Networks. In *Neural Information Processing Systems*, 2014.

Y. Grandvalet and Y. Bengio. Semi-Supervised Learning by Entropy Minimization. In *Neural Information Processing Systems*, 2004.

J. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. A. Pires, Z. D. Guo, M. G. Azar, B. Piot, K. Kavukcuoglu, R. Munos, and M. Valko. Bootstrap Your Own Latent: A New Approach to Self-Supervised Learning. In *Neural Information Processing Systems*, 2020.

J. D. Havtorn, J. Frellsen, S. Hauberg, and L. Maaløe. Hierarchical VAEs know what they don't know. In *the International Conference on Machine Learning*, 2021.

K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2016.

J. Ho, X. Chen, A. Srinivas, Y. Duan, and P. Abbeel. Flow++: Improving Flow-Based Generative Models with Variational Dequantization and Architecture Design. In *International Conference on Learning Representations*, 2019.

P. Izmailov, P. Kirichenko, M. Finzi, and A. G. Wilson. Semi-Supervised Learning with Normalizing Flows. In *International Conference on Machine Learning*, 2020.

T. Karras, S. Laine, and T. Aila. A Style-Based Generator Architecture for Generative Adversarial Networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations*, 2014.

D. P. Kingma, D. J. Rezende, S. Mohamed, and M. Welling. Semi-Supervised Learning with Deep Generative Models. In *Neural Information Processing Systems*, 2014.

D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling. Improving Variational Inference with Inverse Autoregressive Flow. In *Neural Information Processing Systems*, 2016.

I. Kobyzev, S. J.D. Prince, and M. A. Brubaker. Normalizing Flows: An Introduction and Review of Current Methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(11): 3964–3979, 2021.

A. Krizhevsky. Learning Multiple Layers of Features from Tiny Images. 2009.

A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Neural Information Processing Systems*, 2012.

A. Kumar, P. Sattigeri, and P. T. Fletcher. Semi-Supervised Learning with GANs: Manifold Invariance with Improved Inference. In *Neural Information Processing Systems*, 2017.

Y. LeCun and C. Cortes. The MNIST Database of Handwritten Digits. 1998.

D. Lee. Pseudo-Label : The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks. In *ICML Workshop on Challenges in Representation Learning*, 2013.

C. Li, K. Xu, J. Zhu, and B. Zhang. Triple Generative Adversarial Nets. In *Neural Information Processing Systems*, 2017.

L. Maaløe, C. K. Sønderby, S. K. Sønderby, and O. Winther. Auxiliary Deep Generative Models. In *International Conference on Machine Learning*, 2016.

Takeru Miyato, Shin ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual Adversarial Training: A Regularization Method for Supervised and Semi-Supervised Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(8):1979–1993, 2019.

T. K. Moon. The Expectation-Maximization Algorithm. *IEEE Signal Processing Magazine*, 13 (6):47–60, 1996.

E. Nalisnick, A. Matsukawa, Y. W. Teh, D. Gorur, and B. Lakshminarayanan. Do Deep Generative Models Know What They Don't Know? In *International Conference on Learning Representations*, 2019.

Y. Netzer, T. Wang, A. Coates, A Bissacco, B. Wu, and A. Y. Ng. Reading Digits in Natural Images with Unsupervised Feature Learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

S. Nowozin, B. Cseke, and R. Tomioka. f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization. In *Neural Information Processing Systems*, 2016.

A. Oliver, A. Odena, C. Raffel, E. D. Cubuk, and I. Goodfellow. Realistic Evaluation of Deep Semi-Supervised Learning Algorithms. In *Neural Information Processing Systems*, 2018.

Y. Ouali, C. Hudelot, and M. Tami. Semi-Supervised Semantic Segmentation with Cross-Consistency Training. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot,

and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

H. Pham, Z. Dai, Q. Xie, and Q. V. Le. Meta Pseudo Labels. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

G. Qi, L. Zhang, H. Hu, M. Edraki, J. Wang, and X. Hua. Global versus Localized Generative Adversarial Nets. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.

A. Rasmus, M. Berglund, M. Honkala, H. Valpola, and T. Raiko. Semi-Supervised Learning with Ladder Networks. In *Neural Information Processing Systems*, 2015.

M. N. Rizve, K. Duarte, Y. S. Rawat, and M. Shah. In Defense of Pseudo-Labeling: An Uncertainty-Aware Pseudo-label Selection Framework for Semi-Supervised Learning. In *International Conference on Learning Representations*, 2021.

T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved Techniques for Training GANs. In *Neural Information Processing Systems*, 2016.

K. Sohn, Z. Zhang, C. Li, H. Zhang, C. Lee, and T. Pfister. A Simple Semi-Supervised Learning Framework for Object Detection. In *arXiv preprint, arXiv:2005.04757*, 2020.

J. T. Springenberg. Unsupervised and Semi-Supervised Learning with Categorical Generative Adversarial Networks. In *International Conference on Learning Representations*, 2016.

A. Vahdat and J. Kautz. NVAE: A Deep Hierarchical Variational Autoencoder. In *Neural Information Processing Systems*, 2020.

L. van der Maaten and G. Hinton. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.

M. Wattenberg, F. Viégas, and I. Johnson. How to Use t-SNE Effectively. *Distill*, 2016.

X. Yang, Z. Song, I. King, and Z. Xu. A Survey on Deep Semi-Supervised Learning. pages 1–20, 2022.

X. Zhai, A. Oliver, A. Kolesnikov, and L. Beyer. S4L: Self-Supervised Semi-Supervised Learning. In *IEEE/CVF International Conference on Computer Vision*, 2019.

Appendix A

Architecture

In all experiments, we adopt CNN and RealNVP [Dinh et al., 2017] as the architecture of $(K+1)$-class discriminator and normalizing flow. Specifically, our $(K+1)$-class discriminator has the same architecture as Salimans et al. [2016], Dai et al. [2017] for the SSL evaluation protocol [Oliver et al., 2018]. Note that the discriminator with $K+1$ outputs is over-parameterized. That is, subtracting the same value over all the output logits does not change the softmax output. Therefore, in Salimans et al. [2016], the $(K+1)$-class discriminator has K outputs instead of $K+1$ outputs, where we set $K+1$-th logit always equal zero. On the other hand, our normalizing flow has a multi-scale architecture with 3 scales. Each scale has multiple actnorm and affine coupling layers, containing 1 hidden layer CNN. We summarize the details of the model's architecture in Table A.1, A.2 and A.3.

$(K+1)$-class Discriminator D	**Normalizing Flow G**
MLP 1000 units, IReLU, Gaussian noise	Uniform Dequantization, Space To Depth
MLP 500 units, IReLU, Gaussian noise	(Actnorm, Channelwise Mask, Coupling 128 units) $\times 8$
MLP 250 units, IReLU, Gaussian noise	Factor Out, Space to Depth
MLP 250 units, IReLU, Gaussian noise	(Actnorm, Channelwise Mask, Coupling 128 units) $\times 8$
MLP 250 units, IReLU, Gaussian noise	Factor Out, Space to Depth
MLP 10 units, softmax, Gaussian noise	(Actnorm, Channelwise Mask, Coupling 128 units) $\times 8$
	Space to Depth, Coupling 512 units, GMM 256 units

Table A.1: Network Architecture for MNIST.

$(K+1)$**-class Discriminator** D	**Normalizing Flow** G
Gaussian noise, 0.2 Dropout	Uniform Dequantization, Space To Depth
$(3 \times 3$ Conv 64 units, IRelU) $\times 3$	(Actnorm, Channelwise Mask, Coupling 512 units) $\times 8$
0.5 Dropout	Factor Out, Space to Depth
$(3 \times 3$ Conv 128 units, IRelU) $\times 3$	(Actnorm, Channelwise Mask, Coupling 512 units) $\times 8$
0.5 Dropout	Factor Out, Space to Depth
$(3 \times 3$ Conv 128 units, IRelU) $\times 3$	(Actnorm, Channelwise Mask, Coupling 512 units) $\times 8$
Global pooling, MPL 10 units, softmax	Space to Depth, Coupling 512 units, GMM 768 units

Table A.2: Network Architecture for SVHN.

$(K+1)$**-class Discriminator** D	**Normalizing Flow** G
Gaussian noise, 0.2 Dropout	Uniform Dequantization, Space To Depth
$(3 \times 3$ Conv 96 units, IRelU) $\times 3$	(Actnorm, Channelwise Mask, Coupling 128 units) $\times 16$
0.5 Dropout	Factor Out, Space to Depth
$(3 \times 3$ Conv 192 units, IRelU) $\times 3$	(Actnorm, Channelwise Mask, Coupling 128 units) $\times 16$
0.5 Dropout	Factor Out, Space to Depth
$(3 \times 3$ Conv 192 units, IRelU) $\times 3$	(Actnorm, Channelwise Mask, Coupling 128 units) $\times 16$
Global pooling, MPL 10 units, softmax	Space to Depth, Coupling 1024 units, GMM 768 units

Table A.3: Network Architecture for CIFAR10.

Appendix B

Training Details

In all experiments, we adopt Adam optimizer with learning rate to train both $(K+1)$-class discriminator and normalizing flow. We optimize the GMM parameters of the latent space distribution jointly with the normalizing flow parameters. We also tried to optimize the GMM parameters with the EM algorithm but did not find any significant difference. The maximum training epoch is 1200, where an epoch indicates one complete pass of the entire unlabelled dataset through the algorithm. We use a batch size of 100 labelled data and 200 unlabelled data for each min-batch for all experiments. The hyperparameters are tuned based on the separated validation set for MNIST (i.e., $w_{sup_G} = 1.0, w_{nll} = 0.01 * \frac{1}{dim}, w_{adv_G}^{int} = 0.01, w_{PL_G} = 1.0, w_{sup_D} = 1.0, w_{adv_D}^{int} = 1.0, w_{PL_D} = 0.01$). The same set of hyperparameters is used for SVHN and CIFAR10.

Appendix C

Propositions

Proposition 1. *Minimizing the negative log-likelihood is equivalent to minimizing an empirical estimate of the (forward) Kullback–Leibler divergence.*

$$
\begin{aligned}
&\min_{\theta} \mathbb{E}_{x \sim p^{data}(x)} \left[-\log p(x;\theta) \right] \text{ (negative log-likelihood)} \\
\Rightarrow &\min_{\theta} \int_{x \in \mathcal{X}} -p^{data}(x) \log p(x;\theta) dx \\
\Rightarrow &\min_{\theta} \int_{x \in \mathcal{X}} -p^{data}(x) \log p(x;\theta) dx + \int_{x \in \mathcal{X}} p^{data}(x) \log p^{data}(x) dx \\
\Rightarrow &\min_{\theta} \int_{x \in \mathcal{X}} p^{data}(x) \log \frac{p^{data}(x)}{p(x;\theta)} dx \\
\Rightarrow &\min_{\theta} \mathbb{E}_{x \sim p^{data}(x)} \left[\log \frac{p^{data}(x)}{p(x;\theta)} \right] \text{ (KL divergence)}
\end{aligned} \tag{C.1}
$$

where $p^{data}(x)$ and $p(x;\theta)$ are the target distribution and the parametric generative model, respectively.