Imperial College
London

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Exploring Sampling Approaches for Training Energy-Based Models

Author:
Yiming Zhang

Supervisor:
Prof. Yingzhen Li

Second Marker:
Prof. Mark van der Wilk

September 2021

Abstract

The primary purpose of this project is to explore whether there are more effective sampling methods to help energy-based models train more effectively. In the energy-based model, due to the partition function, it is not easy to obtain the gradient of $p_\theta(x)$ using conventional methods. However, the MCMC sampling methods can be introduced to approximate the gradient with contrastive divergence.

In Yilun Du's work: using energy model for image generation (IGEBM) (`https://github.com/karimul/Riset-EBM`, and `https://github.com/yilundu/improved_contrastive_divergence`), and Will Grathwohl's work: using energy model for image classification (JEM) (`https://github.com/wgrathwohl/JEM`), they all used Langevin Dynamics as the only sampling method. Meanwhile, some improvement methods are based on the Langevin Dynamics, such as Metropolis-adjusted Langevin algorithm (MALA), Precondition, Tempering, etc. Some proofs and experiments can support the effectiveness of these improved methods in the field of statistics. However, it has not been researched to use improved sampling methods such as MALA and precondition in energy models. In this thesis, I applied MALA, precondition, and some methods in the sampling process of the energy model to explore whether these methods can effectively improve the training efficiency of the energy model.

During the experiments for image generation and image classification, compared with Langevin Dynamics, the improved sampling methods (MALA, Precondition, etc.) do no have obvious advantages. And these sampling methods that seem to improve training efficiency even have not yielded positive results. However, when we use the MALA and precondition methods to train a simple energy model (using Gaussian scatter points as the data set), compared to Langevin dynamics, these improved methods have obtained a great positive results: the training time is shorter and a better energy distribution is obtained.

The MALA and Precondition might not be suitable for deep energy model training because these improved methods based on Langevin Dynamics require that the step size s of MCMC be $0 < s << 1$, but in IGEBM and JEM, their step sizes are $s = 100$ and $s = 1$, respectively, which breaks the traditional inferences about MALA and precondition.

Acknowledgements

Many thanks for Yingzhen and her selfless help and encouragement throughout the project. I understand that my level is limited and the project requires much mathematical knowledge, but she did not express any dissatisfaction. Her continuous encouragement and comfort is the main reason I could complete this project. At the same time, I am also very grateful to my mother. In this particular year for me and my mother, she alone helped me bear much pressure from family accidents and constantly affirmed and supported me. Thanks.

Contents

Chapter 1

Introduction

Energy-Based-Models(EBMs), which are gradually becoming popular again, use energy distribution to execute tasks, such as generation, classification, regression, etc. With traditional optimization methods, it is hard to train EBMs due to the intractable normalizing constant (we will discuss in 2.1.4). However, from another point, by introducing the MCMC sampling methods, we can interestingly bypass the difficult problem and be more efficient in training EBMs. Langevin Dynamics is one of the most traditional sampling methods used in the energy-based model to calculate the contrastive divergence. However, there are many more efficient MCMC sampling methods have been proposed compared with Langevin Dynamics. This project aims to explore efficient sampling approaches to improve the efficiency and accuracy of energy-based models.

Basic knowledge

This thesis will begin with Gibbs distribution to illustrate the connection between energy and probability. And then some essential concepts about energy function and sampling will be presented, including that: the effect of the partition function for energy function, some examples of energy functions, the reason why MCMC methods are needed for the energy-based model, the inference for the gradient of the partition function, why the contrastive divergence can be used in the energy-based model and the effect of regularization during the training process. These parts are basic requirements for implementing sampling methods on energy-based models. And then, basic concepts about sampling are introduced, such as Markov Chain, transition kernel, metropolis-hastings acceptance ratio, and precondition. Moreover, the transition for Langevin dynamics from physis to Monte Carlo is introduced to discuss the connection between step size and variance. Furthermore, some parameters are needed for practical sampling, including burn-in, thinning steps, and parallel chains methods. And R hats are introduced to evaluate the MCMC method.

EBMs for image generation and classification with different sampling methods

Based on code by Yilun Du (`https://github.com/karimul/Riset-EBM`, and `https://github.com/yilundu/improved_contrastive_divergence`) and Will Grathwohl

`https://github.com/wgrathwohl/JEM`. I tried different sampling methods. MALA and precondition have achieved good results in a simple energy-based model (using Gaussian scatter points as the data set). However, these methods that were initially meant to improve training efficiency significantly did not play a positive role in the EBM of training images, whether an image generation task or an image classification task. Some discussions about why it is negative will be presented in the experimental section.

Chapter 2

Background

2.1 Energy based model

The Energy-Based Models(EBMs), is defined and discussed with the key concept: energy. LeCun et al. [1] has mentioned that the energy is defined to minimize the θ (will discuss in 2.1.2) during inference. "It captures dependencies by connecting scalar energy to each variable's configuration. Inference, i.e. making predictions or decisions, entails determining the values of observable variables and determining the values of remaining variables that minimize energy". During the training process, the θ is updated by an special energy function that connects the low energy to variables with high probability and higher energy with low probability. The loss function can be used to update the energy distribution via the energy functions. During the learning process, lots of choices of energy function can be designed. The quality of accessible energy functions is measured using a loss function that is minimised throughout learning. The numerous energy functions and loss functions available inside this common inference framework enable the creation of a wide range of probabilistic and non-probabilistic learning models.

Despite their current declining popularity, though, EBMs are known to be more difficult to train with familiar methods. While it is possible to train EBM systems, the most significant obstacle is figuring out how to sample the partition function. The sampling of the partition function through amortised generation is one method to training energy-based models. The aforementioned techniques are very similar to GANs as Bengio[2] proposes a separate network to produce samples, but they do not use the optimization strategy described in the introduction. In addition, amorticated generation is likely to collapse, particularly if the sampling network is trained without the often estimated or disregarded entropy component. Using MCMC sampling, an alternate method to estimating the partition function is to partition the marginal distribution of the sampler. In this way, training the EBMs seems possible and it can help people explore the Energy based model. Contrastive Divergence, a gradient-free MCMC-based method, was suggested by Based on the MCMC method, Hinton [3] illustrated the contrastive divergence, which is significant for training the EBMs, is a method to explore the partition function just by the

training dataset. And what's more, some methods for training energy based model, such as contrastive divergence, have been improved gradually. Combined with the persistent constrative divergence[4] illustrated by Tieleman, the MCMC's chain length can be much longer. And Yilun Du[5] has also illustrated that adding the missing KL term to the constrative divergence can improve the quality and stablity of EBM's training process.

2.1.1 From Gibbs Distributions to Energy

Before discussing the energy-based model, let us review the Gibbs distribution[3] shortly to introduce the connection between the energy and probability:

$$P(y) = \frac{1}{Z} \exp(-E(y))$$

$E(y)$ is a general energy function, which is very flexible as long as the corresponding partition function Z is finite. In a conditional Gibbs distribution $p(x|y)$, the energy function would depend on x as well as y. Note that the energy minimum is the most likely value for y. Exponential family distributions can be written as Gibbs distributions with energy $-\theta^T S(y) + C$, where C is any constant. Moreover, in the thesis, the notation will be used:

$$P(y) \propto \exp(-E(y)) \tag{2.1}$$

This formula is a shorthand for (2.1), where we leave the partition function implicit. In specific contexts, it is also helpful to introduce a temperature parameter τ, which is an essential parameter for the energy function:

$$P(y) = \frac{1}{Z} \exp\left(-\frac{1}{\tau} E(y)\right)$$

As the temperature approaches zero, the distribution approaches a point mass on the energy minimum or a mixture of a point mass on energy minima, which only happens when multiple local minima are global minima. Those local minima that are not global will not have a point mass. As the temperature approaches infinity, the distribution approaches the uniform distribution. Also, note that we should not characterize exponential families as having linear energy functions. They are linear in the sufficient statistics, but the adequate statistics may be a non-linear function of y.

2.1.2 Energy function and partition function

Now we have the connection from distribution to energy. At the usual time, we could make a similar inference with Gibbs distribution to define a probability distribution with energy function. The probability distribution can be defined as below:

There is a data point x, set $E_\theta(\mathbf{x}) \in R$ as energy function,

$$p_\theta(\mathbf{x}) = \frac{\exp\left(-E_\theta(\mathbf{x})\right)}{Z(\theta)}[1]$$

is the probability distribution, where $Z(\theta) = \int \exp\left(-E_\theta(\mathbf{x})\right) d\mathbf{x}$ denotes the partition function[6].

To arrive at the most accurate conclusions, the system must allocate the least amount of energy to the right response[1]. It is not necessary to include replies with lower energies as long as they are greater. Occasionally, however, the output of one system must be merged with the output of another or sent into the input of another system. Because energy is unmeasured, adding two independently trained energy-based models does not come with a certain outcome without a priori. The accumulation of energy for all potential outputs can only be converted into a normalised probability distribution by one consistent manner, which is divided by the partition function $Z(\theta)$ (only when the $Z(\theta) = \int \exp\left(-E_\theta(\mathbf{x})\right) d\mathbf{x}$ converge), the energy functions and domains that might be employed are limited.

2.1.3 Examples of EBM

There are many selections for choosing as energy functions, either linear or nonlinear. The exponential family is a typical example, which has a similar formula with energy function.

(1)Exponential family

A probability distribution of the exponential family has the following form:

$$P_\theta(x) = \frac{1}{Z_\theta} h(x) \exp\left(\theta^\top S(x)\right),$$

The $S(x)$ is a vector-valued function that maps x to a suitable collection of statistics, θ is an N-dimensional vector containing natural parameters, $h(x)$ is a base measure that does not depend on θ, and Z_θ is a normalizing constant such that $P(x)$ sums to one. The partition function is identified as $Z_\theta = \sum_{x \in \mathcal{X}} h(x) \exp\left(\theta^\top S(x)\right)$.

We can also make an interesting marginal inference for the exponential family. Marginal inference computes the expected sufficient statistics:

$$\begin{aligned} E_P[S(x)] &= \sum_x P(x) S(x) \\ &= \frac{1}{Z} \sum_x \exp\left(\theta^\top S(x)\right) S(x) \\ &= \nabla_\theta \log \left(\sum_x \exp\left(\theta^\top S(x)\right) \right) \\ &= \nabla_\theta \log Z \end{aligned}$$

Note that $logZ$ is a convex function of θ since the log-sum-exp function is convex [7]. As a consequence, for elements of the minimal exponential family, between the values of θ and $\nabla_\theta \log Z$, there is a one-to-one correlation. In other words, there is a one-to-one correspondence between the expected sufficient statistics of a distribution of the exponential family and its natural parameters.

Numerous well-known probability distributions are members of the exponential family, which is a collection of distributions for a given definition of $h(y)$ and $S(y)$.

(1.1)Gaussian Distribution

With a mean μ and a variance σ^2 on the distribution and $\mathbf{x} \in R$[8], the general form of its probability density function is: $P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$, where $\sigma\sqrt{2\pi}$ being the corresponding partition function.

(1.2) Ising Distribution

Over a set of spins $\mathbf{s} = (s_1, \ldots, s_N) \in \{-1, 1\}^N$, Ising probability distribution[9] is shown as:

$$P(\mathbf{s} \mid \mathbf{h}, \mathbf{K}) = \frac{1}{Z_I} \exp\left(\sum_{i=1}^{N} h_i s_i + \frac{1}{2} \sum_{i,j=1}^{N} J_{ij} s_i s_j\right)$$

with parameters $\mathbf{h} \in R^N$ and $\mathbf{K}$ an $N \times N$ matrix that is symmetric, $Z_I(\mathbf{h}, \mathbf{K})$ is the partition function that corresponds to it.

As we mentioned, the formula of Exponential family is similar with the energy function. All the exponential family function $P_\theta(y) = \frac{1}{Z_\theta} h(y) \exp\left(\theta^\top S(y)\right)$ could be represented as a distributions with the energy $-\theta^\top S(y) - \log h(y)$.

(2)Neural networks

Neural network is a popular method for machine learning task. Essentially, we can use θ to express the Neural network (N) as an nonlinear parameter, corresponding to input x, the energy function is shown as to $E(x) = N_\theta(x)$, whose partition function is $Z_\theta = \sum_{i=1}^{N} \exp\left(-N_\theta(x_i)\right) x_i$.

(3)Posterior probability

We can also using the Bayesian Inference as energy function. Using θ to represent the parameter, the prior distribution is p_θ, and $p(x \mid \theta)$ is the probability of data x parameterized by θ. And we can derive the posterior distribution:

$$p(\theta \mid X) = \frac{p(\theta) \prod_{i=1}^{N} p\left(x_i \mid \theta\right)}{\prod_{i=1}^{N} p(x_i)} \propto p(\theta) \prod_{i=1}^{N} p\left(x_i \mid \theta\right) \propto exp[-(-\log p(\theta) - \sum_{i=1}^{N} \log p(x_i \mid \theta))]$$

In this way, we can define the energy function as:

$$E(x) = -\log p(\theta) - \sum_{i=1}^{N} \log p(x_i \mid \theta)[10]$$

2.1.4 Maximum likelihood learning of EBMs

In this thesis, like most machine learning tasks, we wish to maximize this likelihood on training data. An obvious way to derive a learning algorithm is to minimize the negative log likelihood[11] loss during the energy model training process. Consider the following scenario: assume we getting samples $\mathbf{x}_i$ from $p_{\mathrm{d}}(\mathbf{x})$ for $i = 1, 2, \ldots, N$[12]. The log-likelihood function is shown in (2.2):

$$Log - Likelilood(\theta) = \frac{1}{N} \sum_{i=1}^{N} \log p_\theta(\mathbf{x}_i) = E_{p_{\mathrm{d}}}[\log p_\theta(\mathbf{x})] \, [11] \tag{2.2}$$

We can minimize the KL divergence (between $p_{\mathrm{d}}(\mathbf{x})$ and $p_{\boldsymbol{\theta}}(\mathbf{x})$) to maximize the likelihood[12]. And we need to notice that there is no relationship between the second term and $\boldsymbol{\theta}$, which means it is a constant.

$$-E_{\mathbf{x}\sim p_{\mathrm{d}}(\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x})] = D_{KL}(p_{\mathrm{d}}(\mathbf{x}) \| p_{\boldsymbol{\theta}}(\mathbf{x})) - E_{\mathbf{x}\sim p_{\mathrm{d}}(\mathbf{x})}[\log p_{\mathrm{d}}(\mathbf{x})] \quad [12]$$

Now we encounter a stumbling block: the likelihood of EBMs cannot be directly estimated using the maximum likelihood method owing to the intractable normalisation constant $Z_{\boldsymbol{\theta}}$[12]. However, with Markov Chain Monte Carlo sampling methods, which can accomplish likelihood maximisation using gradient ascent[13] approaches, we can also calculate the log-likelihood gradient. In the second part of the background chapter, we will discuss the details of MCMC methods in details. For inference, The gradient of an energy function's log probability can be divided into two parts.

$$\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{x}) = -\nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\mathbf{x}) - \nabla_{\boldsymbol{\theta}} \log Z_{\boldsymbol{\theta}} [12]$$

Now there is a challenge. The difficulty lies in estimating the second gradient component, which is computationally intractable. However, we can actually bypass this problem with the Monte Carlo estimate. With MCMC methods[12], for the term $\nabla_{\boldsymbol{\theta}} \log Z_{\boldsymbol{\theta}}$, the following expectation may be expressed:

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} \log Z_{\boldsymbol{\theta}} &= \nabla_{\boldsymbol{\theta}} \log \int \exp\left(-E_{\boldsymbol{\theta}}(\mathbf{x})\right) d\mathbf{x} \\
&= \left(\int \exp\left(-E_{\boldsymbol{\theta}}(\mathbf{x})\right) d\mathbf{x}\right)^{-1} \nabla_{\boldsymbol{\theta}} \int \exp\left(-E_{\boldsymbol{\theta}}(\mathbf{x})\right) d\mathbf{x} \\
&= \int \frac{\exp\left(-E_{\boldsymbol{\theta}}(\mathbf{x})\right)}{Z_{\boldsymbol{\theta}}} \left(-\nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\mathbf{x})\right) d\mathbf{x} \qquad [12] \\
&= \int p_{\boldsymbol{\theta}}(\mathbf{x}) \left(-\nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\mathbf{x})\right) d\mathbf{x} \\
&= E_{\mathbf{x}\sim p_{\boldsymbol{\theta}}(\mathbf{x})}\left[-\nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\mathbf{x})\right]
\end{aligned}$$

In the end, the log-likelihood gradient estimation based on MCMC estimation is:

$$\nabla_{\boldsymbol{\theta}} \log Z_{\boldsymbol{\theta}} \approx -\nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\tilde{\mathbf{x}})$$

where $\tilde{\mathrm{x}} \sim p_{\boldsymbol{\theta}}(\mathrm{x})$. In other words, to get unbiased MC estimations of the log likelihood gradient, what we should do is to collect stochastic samples from the model and use some optimizers to update our parameters.

Due to the fact that random samples are not trivial[14], many studies focused on approaches for effective MCMC sampling. Some approaches, like Langevin MCMC[15], employ the log probability gradient with respect to x, which is the same as the negative energy gradient (we will discuss in the 2.2 part):

$$\nabla_{\mathbf{x}} \log p_{\boldsymbol{\theta}}(\mathbf{x}) = -\nabla_{\mathbf{x}} E_{\boldsymbol{\theta}}(\mathbf{x}) - \underbrace{\nabla_{\mathbf{x}} \log Z_{\theta}}_{=0} = -\nabla_{\mathbf{x}} E_{\boldsymbol{\theta}}(\mathbf{x})[12] \tag{2.3}$$

2.1.5 Implement MLE training on EBM in practice by contrastive divergence

Are all MCMC methods suitable for EBMs training? Not at all. Previously researches relied on MCMC techniques such as random walk (e.g. Gibbs sampling)[3], which have a long mixing time in these methods to deal with data with high dimensions, such as speeches and images.

To avoid the occurrence of mixing time during the MCMC sampling process, the dynamics system was introduced to MCMC methods, such as the Langevin dynamics (as shown in 2.4) [16], which collects samples by using the gradient of the energy function added by noise.

$$\mathbf{x}^n = \mathbf{x}^{n-1} - h\nabla_x E_\theta\left(\mathbf{x}^{n-1}\right) + v^n, v^n \sim \mathcal{N}(0, 2h) \tag{2.4}$$

We can define a distribution $\mathbf{x}^k \sim q_\theta$. As Welling and Teh demonstrate[16], when $K \to \infty$ and $h \to 0$, then $q_\theta \to p_\theta$ and this approach will sample from the energy function's distribution. We wish to simulate the data distribution p_d using the distribution specified by E, which will be used for minimizing the negative-log-likelihood of the data: $Log - Likelihood(\theta) = E_{\mathbf{x}\sim p_d}[-\log p_\theta(\mathbf{x})]$, where $-\log p_\theta(\mathbf{x}) = E_\theta(\mathbf{x}) - \log Z_\theta$[16]. We aim to try all the strengths for the EBM to let the energy distribution close to our target distribution p_θ. Meanwhile, we can know that using contrastive divergence[12] can get a gradient-based[17] on the target distribution. However, there are two significant points needed attention. The one is how we get the suitable samples to do contrastive divergence. Another one is how to change the energy distribution based on the contrastive divergence result. The MCMC method is beneficial here because it allows the Markov Chain to converge to the minimum local area with a low energy:

$$\nabla_\theta Log - Likelihood = E_{\mathbf{x}^+\sim p_d}\left[\nabla_\theta E_\theta\left(\mathbf{x}^+\right)\right] - E_{\mathbf{x}^-\sim p_\theta}\left[\nabla_\theta E_\theta\left(\mathbf{x}^-\right)\right][12] \tag{2.5}$$

The process of training energy based model by contrastive divergence can be regarded as a process to do negative log likelihood mixting process. The contrastive divergence has two significant term. As shown in the equation (2.5), the first term shows that some samples obeyed the distribution p_d will be selected. In another words, the p_d is actually the samples from our training dataset. And these samples will be used in the Markov Chain as the start point and positive samples. The second term shows that with some transition steps during the Markov Chain sampling process, some negative samples will be generated based on the positive samples and the distribution p_θ. Usually, with some steps in the Markov Chain, the negative samples are the group of samples from the low energy region, which is what we want for the calculating of constrative divergence. In the end, the result of constrative divergence will actually improve the energy that is from the negative samples $\mathbf{X^n}$, which is from the energy distribution $\mathbf{p}_\theta$. Meanwhile, constrative divergence will also narrow the energy which is from the postive samples $\mathbf{X^0}$, which can be also called the training dataset or original data.

We implement Langevin Dynamics (2.4) as an MCMC example to define $\mathbf{q}_\theta$ as a way of trying to represent $\mathbf{p}_\theta$:

$$\nabla_\theta Log - Likelihood_{\mathrm{ML}} \approx E_{\mathbf{x}^0 \sim p_\theta}\left[\nabla E_\theta\left(\mathbf{x}^0\right)\right] - E_{\mathbf{x}^n \sim q_\theta}\left[\nabla E_\theta\left(\mathbf{x}^n\right)\right].$$

Let us briefly review the previous discussion. Because contrastive divergence can approximate the gradient of the energy function, it can help us approximate the training process of the negative log-likelihood. Then, it can help the energy model to get the desired energy distribution by the back forward if the energy function is a neural network.

According to the formula of contrastive divergence, there are two terms. The first term uses the positive sample, which is the original data set during the training process. If the data set is an image set, then the positive sample here is the original image without any changes. The second term uses the negative sample, which uses the positive sample as the starting point and tries to do some sampling steps. If we can set an appropriate step size and an appropriate chain length through the Markov transfer process, we will get a sample from the local minimum energy through the Markov chain. And as a negative sample in the second term.

However, it is undoubtedly a challenge to ensure that the negative samples we hope can be obtained in a limited chain length. Because only when the negative sample is taken from the local minimum can we guarantee the contrast divergence's regular work. This is because the contrastive divergence will increase the area's energy where the negative sample is located and reduce the energy of the area where the positive samples are located. If the negative sample is not sampled in the local minimum area, it will have a negative impact on training.

2.1.6 Regularization

As said in the last part, we will implement the contrastive divergence to train the energy based model. But usually, the model is unstable during the beginning training process.

Arbitrary energy models may cause significant gradient fluctuations that make sampling by MCMC methods unstable, such as Langevin dynamics. It is helpful to regularise energy levels in L2 weakly for the positive and the negative in training. Since the dissimilarity between the positive and the negative would be kept but their real values will vary to numerically unstable stages[6]. While the partition function is usually integrable across input domains, L2 regularisation assures that the unnormalized distribution has a bounded magnitude. The algorithm is shown below.

Algorithm 1: Train the EBM with L2 loss

Input: data distribution $p_D(x)$, step size k, steps N_s, training epoch N_e
for *epoch* $i = 1$ *to* N_e **do**
  $\mathbf{x_0} \sim p_D$;
  $noise = \sqrt{2k}$;
  ▷ : σ is a huge variance to cover all distribution areas;
  $\mathbf{x}^0 \sim \mathbf{x_0} + \mathcal{N}(0, \sigma)$;
  ▷ Collect samples in q_θ with Langevin Dynamics;
  for *each step* $j = 1$ *to* N_s **do**
    $\mathbf{x}^j \leftarrow \mathbf{x}^{j-1} - k\nabla E_\theta\left(\mathbf{x}^{j-1}\right) + v, v \sim \mathcal{N}(0, 2k)$
  end
  ▷ $\Omega(.)$ means stopping gradient;
  $\mathbf{x}_i^- = \Omega\left(\mathbf{x}_i^k\right)$;
  ▷ Optimize $\mathcal{L}_{ML} + \alpha\mathcal{L}_2$ with respect to θ ;
  $\Delta\theta \leftarrow \nabla_\theta \frac{1}{N} \sum_i \alpha\left(E_\theta\left(\mathbf{x}_0\right)^2 + E_\theta\left(\mathbf{x}_i^-\right)^2\right) + E_\theta\left(\mathbf{x}_0\right) - E_\theta\left(\mathbf{x}_i^-\right)$;
  Update θ based on $\Delta\theta$ using Adam optimizer;
end

2.1.7 Persistent contrastive divergence

Commonly, contrastive divergence can obtain an accurate approximate gradient through approximate methods. Moreover, the contrastive divergence has a very minimal variance in two adjacent calculation processes, and the calculation speed is also breakneck, so it is very widely used in the sampling field of machine learning and artificial intelligence.

An example can describe the contrastive divergence of a single calculation. For the nth contrast divergence, the nth contrast divergence calculation is based on the n-1th calculation in the past, but there is no numerical constraint relationship with the n-2, n-3, etc. However, Tijmen Tieleman[18] proposed that calculating the contrastive divergence again based on the previous calculation results of the

contrastive divergence will get a better calculation result than a single contrastive divergence. Moreover, this contrastive divergence based on the previous calculation results can also extend the length of the Markov chain in disguise. This method will significantly improve using energy-based models to train extensive data (such as images). This method is called persistent contrastive divergence.

In other words, for the energy-based model that obtains negative samples through Markov chain sampling, the effect of single-step contrast divergence is not as good as that of n-step contrast divergence because the single-step contrastive divergence is equivalent to discarding the calculation result every time. However, based on the n-step contrast divergence, the contrastive divergence will be recalculated based on the previous n-step Markov chain.

Now what we need to pay most attention to is how to sample from the probability distribution of our energy model. According to the most mature method, it is to obtain samples from the target distribution through a Markov chain. However, if we are to calculate the contrastive divergence once, we redefine a Markov chain, and only when the length of this chain is very long can he get the samples we need. But this is not appropriate. When training the energy model, we need to consider that if the image data set is sampled, it will be very time-consuming to define the chain length of the Markov chain as a huge number (such as 1000). Moreover, the parameters of the model will only be updated slightly.

In this case, we must think of a Markov chain sampling result that can be used to help us calculate the contrastive divergence later. The main reason is that the result of the previous Markov chain is, to some extent, close to the last distribution energy. However, the parameters of the model have some slight changes. For a mixed model of multiple models, the result is not well due to the complex architecture. But for the energy-based model, this process will be very simple because we only need to sample one energy distribution.

In this case, for large data sets, such as images, we can use a mini-batch to train our model quickly. In this way, we only need to utilize a small number of data as positive samples each time and only a tiny number of negative samples for contrast divergence calculations. But each sampling result will be stored in a stack-like data structure, and we randomly sample nearly 95% of the data from the stack each time as the sampling starting point for negative samples. In this way, we store the sampling results of the past chain each time and use the previous results for sampling during the following sampling process. By updating the model in this form, the training process can be much more efficient and practical.

From the perspective of the entire training process, the essence of Persistent Contrastive Divergence(PCD)[18] is still an approximate approximation process, and it mainly emphasizes that in different Markov chains, we need to keep the previous results instead of emptying them. It performs new Markov chain sampling based on the

sampling results of previous samples. In this way, although each model update is still very subtle, it can significantly save the sampling time and improve the training progress, especially for larger data sets.

2.2 Sampling

2.2.1 Sampling vs Optimization

Machine learning are subjects that combine computer science and statistics in order to tackle inferential issues with such large size and complexity, where contemporary computing architecture is required. The algorithmic basis for these mixes are two broad computing methodologies with mathematical methods: optimization[19] and sampling(MCMC). The majority of research on these methodologies has been conducted independently, with optimization research focusing on estimation and prediction issues, with the goal of locating the global minimum in the distribution. All the points will converge to the minimum value. Moreover, The sampling study has concentrated on jobs that involve estimation of uncertainty, which samples points according to the probability. The number of samples from high probability regions is more than the number of low probability regions.

2.2.2 Markov Chain Monte Carlo (MCMC).

MCMC is a significant component in this thesis, and we will pay much attention on how to do more efficient MCMC sampling during the EBMs training.

Markov Chains.

The most important concept is Markov Chain[20]. A Markov chain in discrete time is composed with a variables sequence X_0, X_1, ..., which can be shown as:

$$\begin{aligned} T\left(x \to x'\right) &= p\left(X_t = x' \mid X_{t-1} = x, X_{t-2}, X_{t-3}, ..., X_0\right) \\ &= p\left(X_t = x' \mid X_{t-1} = x\right) \end{aligned} \text{[20]}$$

where $x_0 \sim p_0(x)$. The function p is called the transition kernel and $T(x \to x_0) = p(X_t = x_0 \mid X_{t-1} = x)$ is the Markov chain transition probability from x to x_0[20]. Within EBMs, gradient of energy function is the transition kernel.

We can infer the marginal distributions in the chain : $x_t \sim p_t(x)$ with the property that:

$$p_t\left(x'\right) = \sum_x p_{t-1}(x) T\left(x \to x'\right) \text{[20]}$$

These marginals combine to form an invariant distribution denoted by T with[20]:

$$p_\infty\left(x'\right) = \sum_x p_\infty(x) T\left(x \to x'\right) \quad \forall x \text{[20]}$$

To ensure $p_\infty = p^*$, the length of Markov chain should be as long as possible, and we can collect samples from distributions within a reasonable approximation distance of p^*, which means that, in an energy-based-model, the sampling result can be very close to our target distribution where the probability is high.
To ensure the $p_\infty = p^*$ have the right invariant distribution, there are two conditions for a Markov chain: ergodicity and detailed balance.
(1) One sufficient condition to be ergodicity[20]:

$$T^k(x \to x') > 0 \text{ for all } x, x' \in \mathcal{X} \text{ and some } k,$$

which indicates that each state may be reached in precisely k steps from any other state.
(2) Detailed balance[20]: $p^*(x)$ must satisfy the following criterion in order to be invariant:

$$p^*(x')\,T(x' \to x) = p^*(x)T(x \to x')$$

If both T and p^* meet ergodicity detailed balance[20], then the marginal of the T chain converges to $p*$.

Gibbs sampling and Metropolis-Hastings algorithm.

Let us go back to Gibbs again. Gibbs Sampling[21] is a way for collecting random samples in a multivariate distribution $p(x)$.
This procedure will iteratively (1) choose i (either at random or in turn) and (2) substitute x_i by a sample in the conditional distribution[20]:

$$p\left(x_i \mid \mathbf{x}_{\setminus i}\right) = p\left(x_i \mid x_1, \ldots, x_{i-1}, x_{i+1}, \ldots x_n\right) [20]$$

[20]
Is it possible to implement Gibbs sampling for training our EBMs? Prior to that, we may demonstrate the detailed Gibbs sampling balancing condition. The possibilities of transition are as follows: $T(\mathbf{x} \to \mathbf{x}') = \pi_i p\left(x_i' \mid \mathbf{x}_{\setminus i}\right)$, where π_i is the probability of choosing to update the i_{th} variable. Then:

$$T(\mathbf{x} \to \mathbf{x}')\,p(\mathbf{x}) = \pi_i p\left(x_i' \mid \mathbf{x}_{\setminus i}\right) \underbrace{p\left(x_i \mid \mathbf{x}_{\setminus i}\right) p\left(\mathbf{x}_{\setminus i}\right)}_{p(\mathbf{x})} [20]$$

and

$$T(\mathbf{x}' \to \mathbf{x})\,p(\mathbf{x}') = \pi_i p\left(x_i \mid \mathbf{x}_{\setminus i}'\right) \underbrace{p\left(x_i' \mid \mathbf{x}_{\setminus i}'\right) p\left(\mathbf{x}_{\setminus i}'\right)}_{p(\mathbf{x}')} [20]$$

Because $\mathbf{x}_{\setminus i}' = \mathbf{x}_{\setminus i}$, then detailed balance is satisfied.

From the transition process, we can notice that Gibbs sampling depends on the target distribution, which may cause very strong correlations between consecutive samples. Thus Gibbs can be slow, and conditionals may be intractable[22]. From this point of view, a global transition is a better choice.

Metropolis-Hastings[23] introduced a method to accept or reject a new state. For each step[20]:
(1) Beginning at the current state x_0,
(2) Using a proposal distribution $S\left(\mathbf{x_{new}} \mid \mathbf{x_0}\right) = S\left(\mathbf{x_0} \rightarrow \mathbf{x_{new}}\right)$ to introduce a new state x_{new} as the next situation,

(3) Adopt the latest situation by probability: $\min\left(1, \frac{p(\mathbf{x_{new}})S(\mathbf{x_{new}} \rightarrow \mathbf{x_0})}{p(\mathbf{x_0})S(\mathbf{x_0} \rightarrow \mathbf{x_{new}})}\right)$
(4) Otherwise retain the old state.
We can also show the detailed balance prove for Metropolis-Hastings. Transition kernel[20] can be shown as:

$$T\left(\mathbf{x_0} \rightarrow \mathbf{x_{new}}\right) = S\left(\mathbf{x_0} \rightarrow \mathbf{x_{new}}\right) \min\left(1, \frac{p\left(\mathbf{x_{new}}\right) S\left(\mathbf{x_{new}} \rightarrow \mathbf{x_0}\right)}{p(\mathbf{x_0}) S\left(\mathbf{x_0} \rightarrow \mathbf{x_{new}}\right)}\right)$$

with $T(\mathbf{x_0} \rightarrow \mathbf{x_{new}})$ the probability of being rejected.
We can suppose that$p\left(\mathbf{x_{new}}\right) S\left(\mathbf{x_{new}} \rightarrow \mathbf{x_0}\right) \leq p(\mathbf{x_0}) S\left(\mathbf{x_0} \rightarrow \mathbf{x_{new}}\right)$. without the generality loss. And we can show the following inference:

$$\begin{aligned} p(\mathbf{x_0}) T\left(\mathbf{x_0} \rightarrow \mathbf{x_{new}}\right) &= p(\mathbf{x_0}) S\left(\mathbf{x_0} \rightarrow \mathbf{x_{new}}\right) \cdot \frac{p\left(\mathbf{x_{new}}\right) S\left(\mathbf{x_{new}} \rightarrow \mathbf{x_0}\right)}{p(\mathbf{x_0}) S\left(\mathbf{x_0} \rightarrow \mathbf{x_{new}}\right)} \\ &= p\left(\mathbf{x_{new}}\right) S\left(\mathbf{x_{new}} \rightarrow \mathbf{x_0}\right) \end{aligned}$$

and

$$\begin{aligned} p\left(\mathbf{x_{new}}\right) T\left(\mathbf{x_{new}} \rightarrow \mathbf{x_0}\right) &= p\left(\mathbf{x_{new}}\right) S\left(\mathbf{x_{new}} \rightarrow \mathbf{x_0}\right) \cdot 1 \\ &= p\left(\mathbf{x_{new}}\right) S\left(\mathbf{x_{new}} \rightarrow \mathbf{x_0}\right) \end{aligned}$$

In some way, Metropolis-Hastings(MH) can be understood as a threshold for sampling, which reduces the strong correlations between consecutive samples. With the training process for EBMs, adding MH accept/reject ratio is helpful to locate the samples close to the target distribution region.

2.2.3 Langevin dynamics and Langevin Monte Carlo

However, is that enough? We need to notice that Gibbs sampling is just doing random sampling, and it is difficult to control the sampling direction, whose transition process is also called random walks[20]. The average distance by a random walk sampling method by n steps will cost $\sqrt{n}$ times[22][14]. Thus these random walk methods sampling from the distributions slowly, especially for the data with high dimensions (e.g. images), which means that it is beneficial to introduce the dynamical motion (e.g. Brownian motion) into our sampling methods. And in this way, the marginal distribution over position will correspond to the target distribution[20].

In this thesis, Monte Carlo sampling from R^d, energy function is shown as $f(x)$, which suggests that: $p(x) \propto e^{-f(x)}$. Our objective is to randomly select i.i.d. samples based on the target distribution with probability density $p(x)$.

Langevin Monte Carlo (LMC) finds the next position point by denoting the position of the sample at the m-th iteration as x^m.

$$x^{m+1} = x^m - h\nabla E\left(x^m\right) + n\xi_d^m,$$

where h is the time stepsize, $n = \sqrt{2h}$ is the noise scale, and ξ_d^m is generated i.i.d. from $\mathcal{N}\left(0, I_d\right)$, where I_d is identity matrix with size $d \times d$.
LMC may be understood as a discretization of the stochastic differential equation(SDE) by Euler-Maruyama[24]:

$$\mathrm{d}x_m = -\nabla E(x_m) + \sqrt{2}dB_t.$$

Note that the B_t is a d dimensions Brownian motion only with independent elements.

There is a connection between the time stepsize h and the noise sacle $\sqrt{2h}$. In physis, consider the Langevin equation as a description of Brownian motion. $x(t+\Delta t) = x(t) - \mu\frac{dU}{dx}\Delta t + \eta_{\Delta t}(t)$[25]. In Brownian motion, the noise amplitude is $D = \mu k_B T$[26] where $k_B T$ is the average kinetic energy. We can understand the μ as stepsize. With slightly movement by Δt, the term $\mu\frac{dU}{dx}\Delta t$ can be eliminated, the Brownian can be described in this equation: $x(t+\Delta t) = x(t) + \eta_{\Delta t}(t)$.

By n steps, the overall sampling time equals with $n\Delta t$. The sample is currently in the following position:

$$x(t+n\Delta t) = x(t)\eta_{\Delta t}(t) + \eta_{\Delta t}(t+\Delta t) + \ldots + \eta_{\Delta t}(t+(n-1)\Delta t).[25]$$

And we can infer that the mean square displacement is averaged as follows:

$$\begin{aligned}\left\langle [x(t+n\Delta t)]^2 \right\rangle &= \left\langle \left[\eta_{\Delta t}(t) + \eta_{\Delta t}(t+\Delta t) + \ldots + \eta_{\Delta t}(t+(n-1)\Delta t]^2 \right\rangle \right. \\ &= \left\langle \eta_{\Delta t}^2(t) \right\rangle + \left\langle \eta_{\Delta t}^2(t+\Delta t) \right\rangle + \ldots + \left\langle \eta_{\Delta t}^2(t+(n-1)\Delta t) \right\rangle \\ &= \left\langle \eta_{\Delta t}^2 \right\rangle n[25]\end{aligned} \tag{2.6}$$

As the total sampling time is $n\Delta t$, and with the mean square displacement of Brownian motion, the equation is satisfied[25]:

$$\left\langle [x(t+n\Delta t) - x(t)]^2 \right\rangle = 2Dn\Delta t \propto 2\mu n\Delta t. \tag{2.7}$$

The (2.5) and (2.6) reveal that:

$$\left\langle \eta_{\Delta t}^2 \right\rangle = 2D\Delta t \propto 2\mu\Delta t.$$

Return to LMC, without the influence of $k_B T$ we can inference that noise scale n is equal with $\sqrt{2h}$ easily.

Langevin Monte Carlo techniques combine Brownian motion with sampling, which is comparable to a gradient ascent approach with adding a Gaussian noise injection: the gradient part directs the chain toward locations of X with a high density of $E(X)$. And meanwhile, the injected noise avoids the chain from falling into to just

the local maximum during the sampling process.

Usually, the length of the Markov chain is large, and the discretization error is inevitable. During the constantly updated sampling process, the Markov chain with discretization error is not assured to converge to the right target distribution. This can be corrected by adding the Metropolis-Hastings adjustment (as discussed in 2.2.2) to set a threshold, which will decide whether accept or reject a new state. This method, which is called as Metropolis Adjusted Langevin Algorithm (MALA)[27], corresponds to using a Gaussian distribution, whose mean is $x_i - h\frac{\partial E(x_i)}{\partial x_i}$ and covariance matrix is $= 2h\mathbf{I}$:

$$T_{\mathrm{MALA}}\left(\mathbf{x}^{(t)} \rightarrow \mathbf{x}^{*}\right)=\mathcal{N}\left(\mathbf{x}^{*} ; x^{(t)}-h \frac{\partial E\left(x_{i}\right)}{\partial x_{i}}, 2h\mathbf{I}\right) \tag{2.8}$$

2.2.4 Practical MCMC

The Markov chain theory ensures that:

$$\frac{1}{M} \sum_{m=1}^{M} G\left(x_{t}\right) \rightarrow E[G(x)] \text{ where } M \rightarrow \infty$$

It is impossible to achieve $G \rightarrow \infty$ with finite computing resources, which means we have to set an infinite length for the Markov chain. Taking the LMC method as an example, which is shown as $x^{m+1}=x^{m}-h \nabla E\left(x^{m}\right)+n \xi_{d}^{m}$. To improve the efficiency and speed for the sampling process, some methods are likely to be used:

Burn-in

We can tolerate some steps to allow a Markov Chain to reach a zone of high probability (low energy) for sampling. The way is called the burn-in method: discarding first K samples from the Markov chain before convergence. If the chain length is defined as M, and after finishing the M-times transition for a Markov chain, we will discard the first K samples:
(1) Sampling stage: repeat M times sampling:

$$x^{m}=x^{m-1}-h \nabla E\left(x^{m-1}\right)+n \xi_{d}^{m-1}$$

(2) Discard the first K times sampling. Only save the sampling result from x^{M-K+1} to x^{M}. The other samples will be discarded.

Thinning steps

We may implement a Markov chain to collecte samples each time for a number of iterations to eliminate sample reliance. For example, we define the thinning step is k, and the length of the chain is M. For the M-times sampling, we will only save the

sampling :
(1) Sampling stage: repeat M times sampling:

$$x^m = x^{m-1} - h\nabla E\left(x^{m-1}\right) + n\xi_d^{m-1}$$

(2) Discard stage: Define the $n = \frac{M}{k}$. And we only save the $x^0, x^{1k}, x^{2k}, ..., x^{nk}$. The other samples will be discarded.

Parallel chains

Another way to reduce the dependence between samples is called parallel chains. For example, we can define K chains with the same length M to reduce the dependence from only one chain. For K chains, we can generate K different initial sample sets from the same distribution.
Sampling stage: repeat M times sampling:

$$x_1^m = x_1^{m-1} - h\nabla E\left(x_1^{m-1}\right) + n\xi_d^{m-1}$$

$$x_2^m = x_2^{m-1} - h\nabla E\left(x_2^{m-1}\right) + n\xi_d^{m-1}$$

$$...$$

$$x_K^m = x_K^{m-1} - h\nabla E\left(x_K^{m-1}\right) + n\xi_d^{m-1}$$

It is beneficial to combine the Thinning and Burn-in to parallel chains. After M times sampling, we can do R hat method to judge whether the sampling method is effective.

2.2.5 Preconditioned Langevin Dynamics

As just introduced, the traditional SGLD method will perform the next step of sampling with fixed step size and a fixed variance during the sampling process of the Markov chain. This sampling process is a random walk process because it has no relation to the previous sampling results when deciding how to sample the next step at each chain node.

However, due to the uneven energy distribution, different gradients at different positions may cause sluggish blending, especially when the gradient θ changes are very large or slow. In this case, it will mislead the sampling process of the Markov chain and affect the calculation of the contrast divergence, which in turn affects the training progress of the energy model. However, avoiding this situation is very difficult. If we use a deep learning neural network as our energy function, there will be many different layers in this energy function, and in each different layer, there will be many non-linear components. Such as non-linear activation function. The complexity of the model and the non-linear relationship will cause sluggish blending to often appear in the training process.

Since this slow mixing process often occurs in deep neural networks, it has gradually become a research hotspot, and many studies have proposed appropriate solutions.

Among them, a solution that may be possible is to use a fixed precondition matrix to multiply the step length every time that SGLD is calculated and update the step length [28]. The original intention of this design is to apply a predetermined condition to the sampling process, thereby affecting the sampling process of random walk. But this method is not desirable. The precondition's most basic original intention is to save the state x_k of the last intra-chain transfer and combine the information of the previous x_k to determine how to follow the new step size and variance in the next step x_{k+1}. In another study, Chunyuan Li[29] proposed a method based on RMSprop to design precondition SGLD.

Algorithm 2: Preconditioned SGLD with RMSprop

Input: stepsizes h, $\mathbf{v} \sim \mathcal{N}(0,1)$, $\alpha = 0.99$, $\lambda = 1e-5$, Energy function E
Initialize: $\mathbf{V} \leftarrow \mathbf{0}$;
for $k = 1, 2, ..., K$ **do**
  Estimate gradient $grad = \nabla_\theta E\left(\mathbf{x}^k\right)$;
  $V\left(\boldsymbol{\theta}_k\right) \leftarrow \alpha V\left(\boldsymbol{\theta}_{k-1}\right) + (1-\alpha) grad \odot grad$;
  $G\left(\boldsymbol{\theta}_k\right) \leftarrow \operatorname{diag}\left(\mathbf{1} \oslash \left(\lambda \mathbf{1} + \sqrt{V\left(\boldsymbol{\theta}_{t-1}\right)}\right)\right)$;
  $\tilde{\mathbf{x}}^{k+1} \leftarrow \mathbf{x}^k - G\left(\boldsymbol{\theta}_k\right) h \nabla_\theta E\left(\mathbf{x}^k\right) + \sqrt{2G\left(\boldsymbol{\theta}_k\right) h}\mathbf{v}$;
end
Return $\mathbf{x}^K$;

As this algorithm shows, there are two key steps:
The V is used to store the gradient results from the past states.

$$V\left(\boldsymbol{\theta}_k\right) \leftarrow \alpha V\left(\boldsymbol{\theta}_{k-1}\right) + (1-\alpha) grad \odot grad$$

The G is the precondition matrix, which is used to update the step size and variance in MCMC sampling.

$$G\left(\boldsymbol{\theta}_k\right) \leftarrow \operatorname{diag}\left(\mathbf{1} \oslash \left(\lambda \mathbf{1} + \sqrt{V\left(\boldsymbol{\theta}_{t-1}\right)}\right)\right)$$

Among them, the original intention of this design is to store the most recent gradient value for the next step and variance setting. Flat distributions tend to have small gradients, and steep distributions tend to have huge gradients. If each sampling is only based on the current gradient, it loses relevance to the previous gradient.

With this design, in a flat distribution area, the preconditioned matrix will cause the step size value to be tremendous, and in a steep distribution area, the step size value will be significantly reduced, helping the Markov chain sample.

2.2.6 Evaluate methods

R hat statistic

Comparing a chain's behaviour against that of other randomly initiated chains is one method to determine if it has converged to the equilibrium distribution, which is the reason why Gelman and Rubin developed the potential scale reduction statistic[30].

The $\hat{R}$ statistic[31] is used to calculate the ratio between the average variance of samples for all chains and the variance of data obtained from distinct chains. If all the chains achieve equilibrium states, the value $\hat{R}$ should be one[31]. If the chains do not converge to a single distribution, the $\hat{R}$ value is greater than one. Moreover, the higher the value of $\hat{R}$, the less satisfactory the convergence effect.

The $\hat{R}$ statistic is calculated with given a collection of M Markov chains, θ_m, each with N samples $\theta_m^{(n)}$. The estimated variance between chains is

$$B = \frac{N}{M-1}\sum_{m=1}^{M}\left(\bar{\theta}_m^{(\bullet)} - \bar{\theta}_\bullet^{(\bullet)}\right)^2, \text{where } \bar{\theta}_m^{(\bullet)} = \frac{1}{N}\sum_{n=1}^{N}\theta_m^{(n)} \text{and} \bar{\theta}_\bullet^{(\bullet)} = \frac{1}{M}\sum_{m=1}^{M}\bar{\theta}_m^{(\bullet)} [31]$$

The estimated variance between chains is,

$$W = \frac{1}{M}\sum_{m=1}^{M} s_m^2, \text{ where } s_m^2 = \frac{1}{N-1}\sum_{n=1}^{N}\left(\theta_m^{(n)} - \bar{\theta}_m^{(\bullet)}\right)^2$$

The variance estimator is a combination of the sample variances within and across chains, $\widehat{\text{var}}^+(\theta \mid y) = \frac{N-1}{N}W + \frac{1}{N}B$. And in the end, the $\hat{R}$ statistic is introduced as: $\hat{R} = \sqrt{\frac{\widehat{\text{var}}^+(\theta|y)}{W}}$.

2.3 Image generation with EBM

A review of generation model:

Throughout the history of machine learning, learning the distribution of data by using models and using the models to generate data has become a hot research topic. In this direction, many excellent models have been proposed. Such as the Variational Autoencoders (VAE model) proposed by Kingma and Welling and the Generative Adversarial Networks model (GAN model) proposed by Goodfellow. These two models have achieved much success in the direction of data generation. However, with the revival of the energy model, it has gradually returned to the research boom. Yilun Du proposed that Energy-based model can be used to generate high-quality image data. By designing the deep neural network as an energy function, we can complete the generation tasks we need through the energy model.

In the model's training process, the purpose of the model is to be able to allocate low Energy to real data, which is the input during model training, and to allocate high Energy to other points. In the training process, we need to use the aforementioned contrastive divergence to estimate the gradient, use the Markov chain Monte Carlo method to sample negative samples, and then calculate the contrastive divergence with the actual data to update the model. In the overall training process, comparing the traditional VAE and GANs models, the energy model has some advanced advantages in generating data.

The advantages of EBMs:

Stability

Unlike other generative models, an energy model is just a single model. It does not require multiple models to be mixed to form a so-called framework. The energy model only needs to obtain samples from actual data and use the Markov Monte Carlo method to sample negative samples and calculate contrast divergence to approximate the updated gradient. In this process, the energy model can avoid the unbalanced training process caused by separate networks. In the performance of VAE and GAN, posterior collapse often occurs. But the Energy-based-model can be avoided to a large extent.

Simplicity

Because the energy model is a separate training object, compared to VAE and GANS, if we can only use fewer parameters, it will significantly reduce the complexity of the model and simplify the training process. It can also avoid the occurrence of over-fitting. In addition, the singularity of the energy model can also facilitate the internal structure instead of mixing up many models like other models. Therefore, the energy model can maintain good consistency during the parameter transfer process.

Less time for training

Due to the simple model structure, energy models often have fewer model parameters. In the training process, simple models tend to take up less memory. If we use GPU to train our energy model, compared to VAE and GANS generative models, perhaps a GPU with a smaller capacity is enough to complete the training task. In addition, when calculating the contrastive divergence, since only a single model needs to be trained, we can significantly reduce the training time and get better results. If combined with the random optimization process and mini-batch, it will save training time.

Despite its advantages, the energy model is still complicated to train in high-dimensional data sets. One of the most challenging steps is how to calculate the contrastive divergence by collecting suitable samples. This report used Langevin dynamics, MALA, and PSGLD as sampling methods and compared the final results.

2.4 Image classification with EBM

At the moment, a significant performance difference is existing between the most powerful generative model method and particular solution for different particular issue in our daily life.
One possibility is that most downstream jobs are fundamentally distinct and that the advanced generative models are different from the advanced discriminative architectures. Therefore, even if we train the discriminator and generator separately, the gap in

their performance will be huge.

Will Grathwohl[32] proposed the Joint energy-based model. He used the energy model as a possible generative model for downstream identification tasks. Compared with other complex hybrid models, the energy model can be more easily integrated with the discriminative module, and a practical classification structure can be built.

In the process of training JEM, two critical key points need our attention:

(1): The joint energy model(JEM) shows us a framework that can combine labels and data into the same energy distribution. (2): Meanwhile, the JEM can also be understood as a model that mixes the generative model and the discriminant model into one at the same time.

2.4.1 Something is hiding in the classify

In the machine learning research, a classifier problem can be described as the following process: suppose the data set has N labels, and D is the distribution of our data set. What we need to do is to map the distribution of R^D to R^K labels. In this mapping transition process, every data point will be mapped from distribution D to distribution K through this logical mapping. Usually, this kind of mapping can be expressed as the familiar softmax transfer function:

$$prop_\theta(y \mid \mathbf{X}) = \frac{\exp\left(g_\theta(\mathbf{X})y\right)}{\sum_{y'} \exp\left(g_\theta(\mathbf{X})\ y'\right)}[32] \tag{2.9}$$

Among them, $g_\theta(X)y$ represents the case where the label of $g_\theta(X)$ is y.

Our main goal is to reapply the logical mapping described in formula (2.9) to the energy model. Without changing the energy function, we can refer to this logical mapping to establish a joint energy model distribution: x is the real data, and y is the label to the x.

$$prop_\theta(\mathbf{X}, y) = \frac{\exp\left(g_\theta(\mathbf{X})y\right)}{Z(\theta)} \tag{2.10}$$

As we mentioned in the first section, $Z(\theta)$ is a partition function. This function can be regarded as an unknown normalization factor. And we can know that for each (x, y), there is existing $g_\theta(x, y) = -g_\theta(x)y$. We can calculate the edge distribution of x and then eliminate y. So we can get a probability distribution about x:

$$prop_\theta(\mathbf{x}) = \sum_y prop_\theta(\mathbf{x}, y) = \frac{\sum_y \exp\left(g_\theta(\mathbf{x})y\right)}{Z(\theta)}[32] \tag{2.11}$$

Please note that in this case, we can use LogSumExp(.) to use logical mapping to define our energy equation. Where x is a real data, and y is the label corresponding to this x.

$$E_\theta(\mathbf{x}) = -\operatorname{LogSumExp}_y\left(g_\theta(\mathbf{x})y\right) = -\log\sum_y \exp\left(g_\theta(\mathbf{x})y\right) \tag{2.12}$$

In traditional classifiers, arbitrarily scaling this logical mapping $g_\theta(x)$ will not affect the performance of the entire model. However, if we scale a logical mapping of the original data in the joint energy model, it will affect the $log\ prop_\theta x$. As a result, we utilize the additional degrees of freedom inherent in the logic mapping to define the density function of the input example as well as the combined density of the example and the label[32].

In the end, we can use $g_\theta(x,y)/g_\theta(x)$ (in Equation 2.10 and Equation 2.11) to calculate $g_\theta(y \mid x)$. This is what we can find that the original partition function has been eliminated. And the result of elimination is exactly the expression of softmax. Therefore, JEM can be called a generative model with many discriminators hidden.

2.4.2 How to optimize JEM

Through the derivation of Joint Energy Based model, we can design a particular generative model. In this generative model, there are also many discriminators with good performance. Because our model can be expressed as $prop_\theta(y \mid x)$, we can train the model as a classifier. In addition, $prop_\theta(x)$, $prop_\theta(x,y)$ is not regularized because there is a partition function in the equation, which makes it not easy to obtain their maximum likelihood estimates. But if we can derive the relationship mathematically and eliminate the partition function in the denominator, we can regularize $prop_\theta(x)$, $prop_\theta(x,y)$ in another way.

Referring to equations (2.10) and (2.11), we can design a likelihood function as follows:

$$\log prop_\theta(x,y) = \log prop_\theta(x) + \log prop_\theta(y \mid x)[32] \tag{2.13}$$

The first term can be calculated by the contrastive divergence to estimate the gradient of x. Based on the section 2.1.5, we can estimate the log-likelihood of singal x corresponding to the θ:

$$\frac{\partial \log prop_\theta(\mathbf{x})}{\partial\theta} = E_{prop_\theta(\mathbf{x}')}\left[\frac{\partial E_\theta\left(\mathbf{x}'\right)}{\partial\theta}\right] - \frac{\partial E_\theta(\mathbf{x})}{\partial\theta}[32]$$

This equation means that we can approximate the gradient by sampling with the Markov chain Monte Carlo method. And the cross-entropy is used to optimized the $\log prop(y \mid x)$. And the sampling methods like SGLD are used to estimate the gradient, which is respect to $\operatorname{LogSumExp}_y(.)$ mentioned in the equation(2.12).

Chapter 3

Goals and Approaches

As part of the background said, the energy-based model has been used for image generation and classification. However, recently, all the popular directions aim to improve the training process from the point of the loss function, such as improving the contrastive divergence. It is meaningful to explore the some advanced sampling methods on training image generation/classification model by energy-based model. In this thesis, our main goal is to explore, evaluate, and discuss if some advanced sampling methods can improve the training quality for using energy-based model to generate and classify images. There are some necessary approaches to prepare for the evaluation part.

3.1 Residual Neural network

As mentioned in background part 2.2, the neural network is a popular choice as an energy-based model's energy function. Among the lots kinds of neural networks architecture, the residual neural network is prevalent and suitable for training models with images.

The network gradually degrades as network depth increases.

The depth of neural networks is already regarded as a critical factor influencing network performance. However, gradient disappearance and explosion interfere with the training of extremely deep networks. To a large extent, the introduction of normalization solves this problem, such as accelerating a deep neural network to converge. However, there will also be some issues, and the network will tend to decay. With the increasing depth of the deep neural network, accuracy will become saturant and decreasing rapidly[32]. Overfitting is not the cause of this, and with more layers added into the deep network, the errors of accuracy will be more serious.

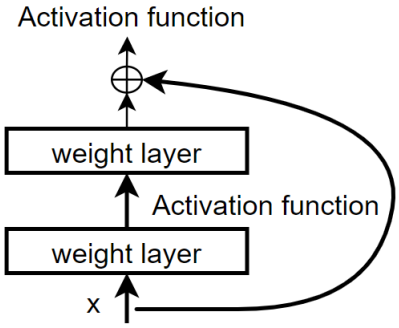


Figure 3.1: Residual Block

Learning residual.

To address this issue, the author(Kaiming He)[33] proposed the concept of deep residual learning, in which the stacked network is fitted using a brand new residual mapping rather than the implicit mapping originally intended. If the implicit mapping that is being considered is denoted by $H(x)$. The residual mapping can then be expressed as follows:

$$H(x) + x$$

Kaiming He demonstrated how much easier it is to fit this new mapping. For example, when fitting an identity map, setting the residual to zero is clearly more convenient than stacking multiple nonlinear layers to fit.
If we denote the residual connection as $F(x)$, we can obtain it using the previous formula.

$$F(X) = H(x) - x$$

Add x to the right of the equal sign to obtain

$$F(X) + x = H(x)$$

That is, the mapping relationship discovered by conventional neural networks is actually $F(X) + x$. This type of mapping is possible through the use of shortcut connections during the network's construction. Shortcut connections (3.1) serve as identity mapping in the ResNet model. As a result, its implementation is quite straightforward—just add the input to the output.
In theory, increasing the number of layers in a neural network should not degrade the network—for example, when the additional layers are all identity mappings. However, the network is degraded, which means that the solver will have difficulty fitting the identity map using the multi-layer network. While the identity mapping is unlikely to be the optimal choice in real-world situations, if the optimal choice is very close to the identity mapping, the residual network can easily learn the separation scheme of <identity mapping + perturbation>. Instead of a comprehensive optimal mapping.

3.2 Advanced tips for training EBMs

3.2.1 Multi-scale energy based model

Yilun Du[5] has illustrated some important and beneficial tips during the process for training energy based model. For example, spectral normalisation can be used to prevent energy divergence. And with multi-scale image input, the final energy value is the sum of the different scales' images energy, which can improve the generation quality and stable the training process. Energy is distributed unevenly across the surface, with deep and shallow pits. The lower the energy, the deeper the pit. Because it is difficult for the energy model to achieve the perfect distribution we expect during the early stages of training, the training time will be lengthy, and the change in the energy model's distribution may not always proceed as expected. However, by using inputs of various scales, obtaining the energy at various scales, and combining them, the robustness of the energy model can be significantly improved during the initial training stage.

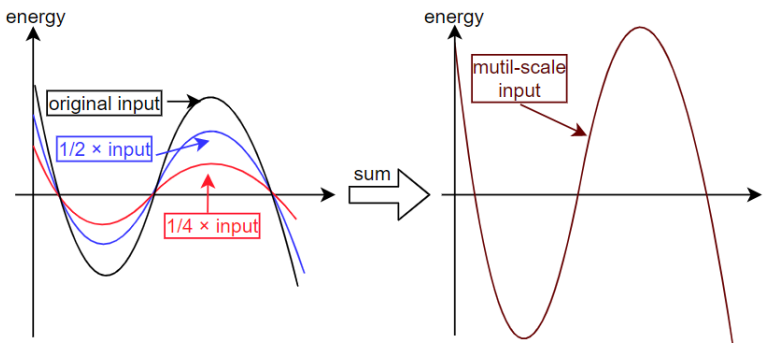


Figure 3.2: multi scale energy distribution

As shown in Figure 3.2, with the original input energy, the energy distribution is very simple, with a small difference between high and low energy. However, if a multi-scale input is used and the energy values from the various scales are added together to yield the final energy value (the sum of different scale energy), the difference between high and low energy is significantly increased. This also serves as an excellent preparation for more precise sampling.

3.2.2 Data augmentation

At the same time, data augmentation is used to the image data set for training. In data science research, image augmentation methods are always adopted to enhance the quality of data gathered by either adding significantly modified samples of previously obtained data or by generating new synthetic data based on the obtained data. It serves as a regularizer and prevents the model from overfitting[34]. It is strongly linked to data analysis oversampling. The image generation model employs picture enhancement techniques such as geometric changes, flipping, color alteration, cutting, rotating, noisy input, and random erasing.

3.2.3 Improved Contrastive Divergence

Yilun Du proposed the Improved Contrastive Divergence on paper "Improved Contrastive Divergence Training of Energy-Based Model."[5] In this paper, he discussed that there is a missing term in Contrastive Divergence. Now review the contrastive divergence equation (from section 2.1.5) for training the energy-based model.

$$\nabla_\theta \log - \text{Likelihood}_{\text{ML}} \approx E_{\mathbf{P}(x)}\left[\nabla E_\theta\left(\mathrm{x}\right)\right] - E_{\mathbf{q}(x')}\left[\nabla E_\theta\left(\mathrm{x}'\right)\right]$$

But let's also see the contrastive divergence equation illustrated by Hitton[3].

$$\mathrm{KL}\left(p_D(\boldsymbol{x}) \| p_\theta(\boldsymbol{x})\right) - \mathrm{KL}\left(q_\theta(\boldsymbol{x}) \| p_\theta(\boldsymbol{x})\right)[5]$$

And we can see, for the first term in constrastive divergence, the two term is equal with each other:

$$\frac{\partial \mathrm{KL}\left(p_D(\boldsymbol{x}) \| p_\theta(\boldsymbol{x})\right)}{\partial \theta} = -E_{p_D(\boldsymbol{x})}\left[\frac{\partial E_\theta(\boldsymbol{x})}{\partial \theta}\right][5]$$

But for the second term in constrastive divergence,

$$\frac{\mathrm{KL}\left(q_\theta(\boldsymbol{x}) \| p_\theta(\boldsymbol{x})\right)}{\partial \theta} = \frac{\partial q\left(\boldsymbol{x}'\right)}{\partial \theta} \frac{\partial \mathrm{KL}\left(q_\theta\left(\boldsymbol{x}'\right) \| p_\theta\left(\boldsymbol{x}'\right)\right)}{\partial q_\theta\left(\boldsymbol{x}'\right)} - E_{q_\theta(\boldsymbol{x}')}\left[\frac{\partial E_\theta\left(\boldsymbol{x}'\right)}{\partial \theta}\right][5]$$

There is a missing term $\frac{\partial q(\boldsymbol{x}')}{\partial \theta} \frac{\partial \mathrm{KL}(q_\theta(\boldsymbol{x}') \| p_\theta(\boldsymbol{x}'))}{\partial q_\theta(\boldsymbol{x}')}$[5] in the loss function for training the energy based model. What the paper suggested is adding this term to the loss function for training the EBMs. And in the evaluation part of this paper, it showed the KL-term can improve the stablity of energy based model. And in this thesis, the KL-term is used to the unsupervised generation part.

3.2.4 Replay buffer

Replay buffer is an important part in this thesis. Because the datasets are images and all of them have huge size, it is very time-consuming if the chain length is set as 1000 or larger. But if the chain length is just set as 50 or 100, the sampling process can not arrive any area with low energy but just stay at the start point.

To solve this problem, the replay buffer was proposed[6] based on the persistent Langevin dynamics(PCD)[18]. We can assume the replay buffer as a stack. For each batch's training, the replay buffer will save the sampling result as the new input during the training process. The oldest sampling result will be deleted from the Replay Buffer if the stack capacity has reached the maximum.

The energy-based model will randomly select samples from the replay buffer at the start of each batch training. Use these samples as a jumping-off point to begin sampling. The contrastive divergence is then calculated using the final sampling result and the original image. We can extend the length of the Markov chain at a low cost by using a replay buffer.

3.3 Sampling methods for image

In this part, I will propose some possible advanced methods for training the energy-based model with images.

3.3.1 Suitable hyperparameters

Typically, the advanced MCMC methods are based on optimization. For example, CA Bhardwaj[35] proposed the <Adaptively Preconditioned Stochastic Gradient Langevin Dynamics> as a new optimization method for neural network training. But if we use the MCMC methods as sampling methods, we need to consider some key points:

Not all datasets can be sampling efficient with tiny step size.

Usually, the step size is defined as around 0.0001 to 0.01. But it is not suitable for images if we want to use the energy-based model. Let's review the Langevin dynamics for sampling equation:

$$\mathbf{x}^{k+1} \leftarrow \mathbf{x}^{k} - h\nabla_{\theta} E\left(\mathbf{x}^{k}\right) + \sqrt{2h}\mathbf{v}$$

Usually, the term v is obeyed as a normal distribution $N \sim (0, 1)$. As we can see, the new state x^{k+1} is updated according to the difference from the last term x^{k}. And the difference is made up by the two term: $h\nabla_{\theta} E\left(\mathbf{x}^{k}\right)$ and $\sqrt{2h}\mathbf{v}$. Among them, the h term is step size and the $\sqrt{2h}\mathbf{v}$ is variance. The main reason is that the value of gradient term $\nabla_{\theta} E\left(\mathbf{x}^{k}\right)$ can not be decided during the training process and it will depend both on the dataset and the energy based model. In this thesis, for Cifar-10 and Cifar-100 data,, the suitable step size should be limited between 10 and 100 to sample because the absolute value of term $\nabla_{\theta} E\left(\mathbf{x}^{k}\right)$ is always around 10^{-4}.

But following the equation, the variance should be correlated to the step size h. But if we set the h as 100, the variance is enormous and will destroy the samples which are drawn from the replay buffer. To overcome this issue, we must provide a fixed variance for a particular energy function and data set in order to guarantee the authenticity of the replay buffer samples. So the variance should be limited as a small value and should balance the variance and the gradient.

3.3.2 Stochastic gradient langevin dynamics

Stochastic gradient Langevin Dynamics(SGLD)[36], proposed by Welling, is a popular method for bayesian learning. The method has been used in many energy-based models as the primary sampling method. For example, Yilun Du[6] and Will Grathwohl [32] have used the SGLD method as their only sampling method to collect the negative samples from the replay buffer. And according to their result, the SGLD can work very well and achieve the training aim. Yilun Du uses the SGLD method as the sampling method to train the energy-based model for image generation, and Will Grathwohl uses it to implement the joint energy-based model for image classification.

3.3.3 Tempering

The main idea of tempering Langevin dynamics for Bayesian learning is suggested from Chandra R, Jain K, Deo R V, et al. [37]. Tempering can also be used in the energy-based model sampling process. Take the one dimension distribution as an example. The energy distribution can be described as a curve function. Each point on the x-axis corresponds to a single energy value.

Why do we need to mention tempering? The main reason is that tempering can

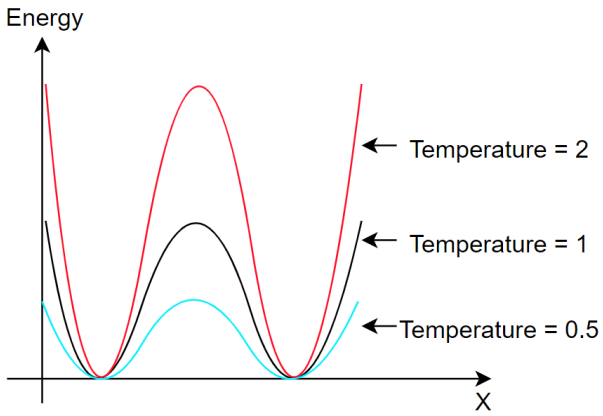


Figure 3.3: One dimension energy tempering

change the gradient by changing the energy distribution. As the picture shown3.3, the energy value is calculated by this formula: Tempering Energy = Original Energy $\times$ Temperature. Assuming the original energy distribution is the black curve. With this method, like the red curve, the gradient of the Energy will sharply multiply by two. So during the sampling process, if we set the temperature > 1, the energy difference between two points will be much possible to have a more significant gap, which can sharp the energy distribution and enlarge the gradient of the energy function. In other words, scale down the Energy with a low temperature < 1 can narrow the energy difference between two near points.

If the original energy distribution is very flat, we can set the temperature > 1 to sharp the gradient, according to the sampling method equation:

$$\mathbf{x}^{k+1} \leftarrow \mathbf{x}^{k} - h\nabla_{\theta} E\left(\mathbf{x}^{k}\right) + \mathbf{v}$$

the term $h\nabla_{\theta} E\left(\mathbf{x}^{k}\right)$ will be enlarged to sample data far away the starting point. Especially for training some datasets with low gradients, such as Cifar-10. In the evaluation section, the gradient of this dataset is always limited to about 10^{-4}, and to achieve a new suitable state which is far away from the start point; it is needed to set a large step size or set the enlarge the temperature.

3.3.4 Metropolis-adjusted Langevin algorithm for image (MALA)

As mentioned in part 2.2.2, the Metropolis-Hastings algorithm is an effective method to help the Markov chain converge. In the transition process of every two adjacent

states, the reliability of the Markov chain sampling process can be improved by introducing a mechanism of whether to reject or accept. As shown in the analysis Figure 3.4:

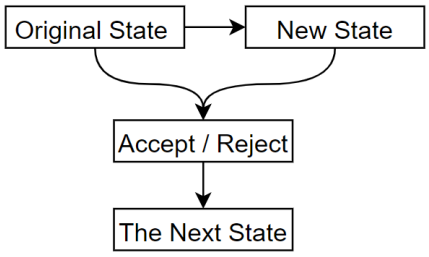


Figure 3.4: MALA process

The Metropolis-Hastings algorithm's first step needs to get the next state based on the original state. If the SGLD is used, the next state x^{k+1} is calculated by this formula.

$$\mathbf{x}^{k+1} \leftarrow \mathbf{x}^{k} - h\nabla_{\theta}E\left(\mathbf{x}^{k}\right) + \mathbf{v}$$

However, typically, not all the new states can be accepted as the new state. Just thinking that, during the SGLD transition, if the step size is a fixed value, we can not make sure the new transition direction will be what. Furthermore, usually, it will be a transition like random walk[38]. However, if there is a mechanism to judge whether this new state is worth accepting as the next state, or it can be judged whether this new state is too far from the previous state and should be rejected, the mechanism will improve the converge speed on the Markov chain and act a very critical assisting role.

Metropolis-Hastings algorithm can play this critical role in Langevin dynamics, and this method has been proposed by Atchadé Y F[39]. For the statistics in Markov Chain Monte Carlo, Metropolis-adjusted Langevin algorithm can combine the two fundamental mechanisms (where the one is a random walk and the other one is accepted or reject) with improving the sampling task and speed to converge to the aim probability distribution. The two mechanisms are:

All the new states should be transformed and proposed by the Langevin dynamics, and the gradient $E\left(\mathbf{x}^{k}\right)$ should be the target probability distribution. In an energy-based model, the gradient should be the gradient of the energy function.

The Metropolis-Hastings algorithm will judge the new state, and the decision is whether accept or reject the new state proposed by the Langevin dynamics. In the process of judging, the energy probability density will be used, but not the gradient of the energy function.

With an unofficial description, Langevin dynamics will ask the transition process (random walk) to close to the areas with high probability (corresponding to the low energy in the EBM) based on gradient of target probability function. Then, the Metropolis Hasting algorithm will act as a judge mechanism if the new state is

valuable to help the random walk mix to the target probability distribution. In this way, the Metropolis Hasting algorithm will improve the accuracy to converge to the target distribution. How does the judging mechanism work? There are some steps list below:

The first step. Based on the formula: $\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k - h\nabla_\theta E\left(\mathbf{x}^k\right) + \mathbf{v}$, we can get the new sample $\mathbf{x}^{k+1}$.
The second step. Nevertheless, in MALA, the new sample result is not the end. Moreover, we can note it as a temporary state or a state of waiting for judgment. It can be expressed as $\tilde{\mathbf{x}}^{k+1}$.
The third step. Metropolis-Hastings algorithm proposes the judging mechanism based on the equation:

$$\alpha := \min\left\{1, \frac{E\left(\tilde{X}_{k+1}\right) q\left(X_k \mid \tilde{X}_{k+1}\right)}{E\left(X_k\right) q\left(\tilde{X}_{k+1} \mid X_k\right)}\right\}$$

And from X_k to $\bar{X}_{k+1}$, the transition probability density $q\left(\tilde{x}^{k+1} \mid x^k\right)$ can be shown as:

$$q\left(\tilde{x}^{k+1} \mid x^k\right) \propto \exp\left(-\frac{1}{4h}\left\|\tilde{x}^{k+1} - x^k - h\nabla \log E(x)\right\|_2^2\right)$$

The forth step. Metropolis-Hastings will create a uniform distribution u, which is drawn from the interval $[0, 1]$. And if the $u \leq \alpha$, the MALA will accept the state $\tilde{x}^{k+1}$ as the new state x^{k+1}:

$$x^{k+1} = \tilde{x}^{k+1}$$

Else, the new state will be rejected, and the MALA result will use the old state as the final state:

$$x^{k+1} = x$$

As shown above, the MALA will help the Langevin dynamics close to the region with high probability (where the energy is low), whose new states are more likely to be accepted in the transition process.

3.3.5 Combination the tempering with MALA

Based on the last part discussion, MALA will correspond to the acceptance/rejection mechanism to decide whether to receive the new state as the next state. However, if the acceptance ratio is meager (about 0.1 or even lower) for a long time, the Markov Chain might have been trapped in some areas with low energy, and the other places near this low-energy region have high energy. As shown in the figure 3.5.

In this case, if the MALA's has been very low, the Markov chain will always be stuck in a bottomless "pit," no matter how Langevin dynamics will try to "climb out" of this bottomless pit due to probability density MALA's mechanism will reject the difference. When this happens, how do we know that this has happened, and to

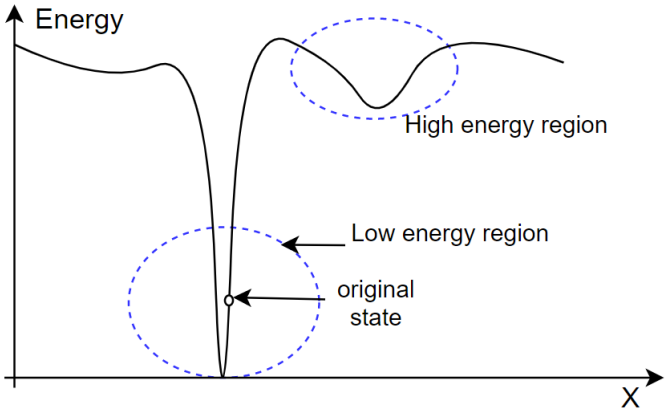


Figure 3.5: High/Low energy region

alleviate this problem?

There are some options to alleviate in theory. Firstly, we can scale down the step size to allow the new state to move away from the deep "pit" gradually. But with the small step size, it will be very slow to move away from the pit. There is another way to improve the acceptance ratio and help the original state to get out of the "deep pit": tempering. Imagine turning all the energy into a fraction of the original—for example, one-tenth. Then the distribution of this energy will become flattered, and the difference between the high-energy and low-energy regions will also become smaller. We can use more vivid words to describe that the depth of the originally bottomless pit will become smaller, but the depth of the initially shallow pit will not change much. Let us take an elementary example. Assuming the original low-energy value is -10, and the high-energy value is 1. If we change these two energy values with a tempering hyperparameter as 0.1, then the low energy value will be -10 $\times$0.1 = -1, and the change will be 9. The high energy value will be 1$\times$0.1 = 0.1, and the difference is only 0.9.

We can design a minimum threshold for the acceptance rate. Once the acceptance rate is lower than this threshold, we need to perform a tempering transformation on the energy distribution: Energy $=$ Old Energy $\times$ temperature $(0 <$ temperature $< 1)$.

3.4 Precondition for sampling

Precondition matrix

As mentioned in chapter 2, the precondition is an effective tool in the Markov Monte Carlo process. It can dynamically update the step size and variance required for the next state, and the updated benchmark is based on the gradient of all states experienced before this chain.

However, due to the complexity of the model and the limitations of the mini-batch,

it is impossible to set the length of a single Markov chain to be very long, which will limit the role of precondition to some extent. Moreover, it is also necessary to consider whether the precondition should be applied to the variance. If we apply a significant variance to the original image for image data, the original energy distribution might be destroyed. In other words, our training set will be invalidated by precondition. This situation is existing, and it is also mentioned in the evaluation part later. Because it is difficult to determine the magnitude of the preconditioned matrix, if this precondition matrix is only a small weight, it will only play a harmful role in the sampling process. For example, if we set the step size to 100, then it can sample usually. But if our precondition matrix is often only a number around 0.0001, then the result of division with the step size can reach around 10^6. For a model that can be sampled commonly only when the step size is 100, such a step size will cause the progress of the model training to be meaningless or even destroyed.

Nevertheless, such an attempt is not meaningless. For short Markov chains, if precondition can help the sampling process to a great extent, it will also significantly improve the training efficiency of the energy model.

Annealing

There is another way. We can gradually reduce the step size because as the Markov chain Monte Carlo converges, the sampling result will get closer and closer to the place where the energy is low. This usually happens in the second half of a Markov chain. However, if we still use the same step length from the first half in the second half, it may cause the point where it should be escaped from this low-energy region.

In other words, we can also gradually reduce the learning rate to avoid the latter half of the Markov chain from escaping the low-energy area due to the large step size, thereby causing errors.

3.5 Unconditional/Conditional EBM

In the image generation part, we use the deep residual network as our energy model. When using a deep residual network, we can choose to include the data label as part of our training set. In the image generation stage, we can use labels to enhance the quality of our image generation. But for JEM, we must add the label as part of the training set input to do the classification task.

Yilun Du's image generation model and Will Grathwohl's JEM are used in the subsequent experiments. Among them, the code references the official code of Yilun Du: `https://github.com/yilundu/improved_contrastive_divergence`, `https://github.com/rosinality/igebm-pytorch`, and `https://github.com/karimul/Riset-EBM`.

For JEM, I quoted Will Grathwohl's official code: `https://github.com/wgrathwohl/JEM` for experimentation. Thanks for their open-source code.

My contributions are mainly to try advanced sampling methods (such as MALA and PSGLD), and experiment with different hyperparameter combinations, and draw corresponding conclusions.

Chapter 4

Experimentation

4.1 EBM for Gaussian scatters

In this section, our goal is to justify if the langevin dynamics, Metropolis-adjusted Langevin algorithm and precondition langevin dynamics will work in a simple energy based model. We have implemented different sampling methods on a simple Energy-based model. The energy function is specified as a fully connected neural network. The target distribution is 4-mixture Gaussian Distribution, whose centers are $[10, 0], [0, 10], [-10, 0], [0, -10]$, and the mean and variance for each center is 0 and 1 (the gray points, figure [4.1]). In this experimentation, we implement the algorithm 1 on page 7, and generate 1000 training samples from the target distribution as the x_{train}. The green points are the initial points for sampling(x^0), which is calculated by $x^0 = x_{train} + Normal(0, 15)$.

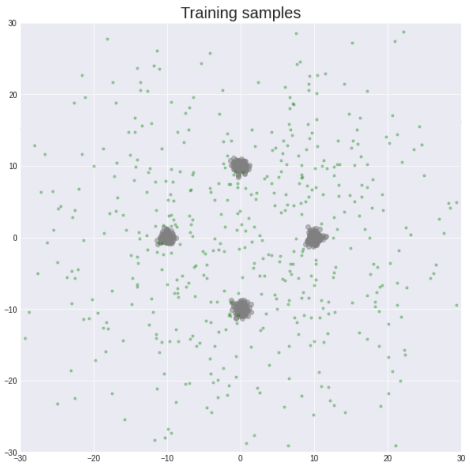


Figure 4.1: Training samples

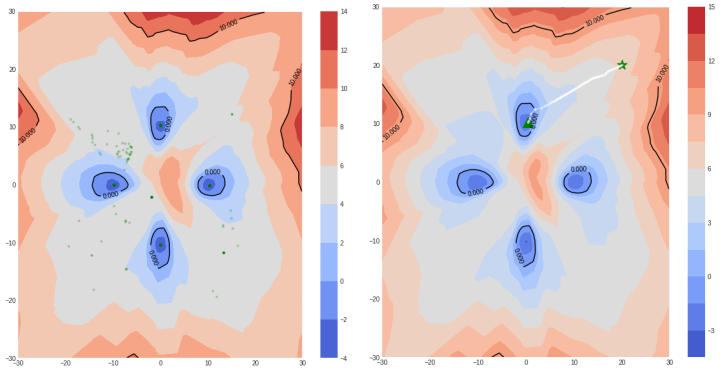
We set steps = 5000, training epoch = 300, stepsize = 0.01 for all sampling methods, and with different methods, we tested the R hat value and plot two graphs (red to high energy and blue to low energy). One is the sampling scatters map corresponding to the energy map, showing the relationship between the sampling result and the energy. Another one is the scatter trace map (from pentagram to triangle), which can help us figure out the sampling trend from high-energy regions to low-energy regions. Finally, we use the R hat method, with ten independent chains, to evaluate these sampling methods.

4.1.1 Gradient Descent

Make a review on gradient descent (GD). GD is an optimization method based on gradient that updates state with the negative gradient direction. From the resulting graph(Figure [4.2]), we can notice that almost all the samples trend to locate the local minimum point quickly. We have tested the R-hat for sampling with the gradient descent method. The R-hat value is 2891.63, which shows that the GD is not a suitable sampling method.

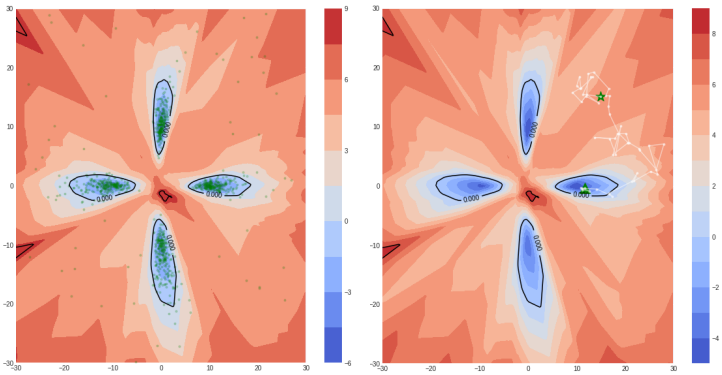
Algorithm 3: Gradient Descent

Input: $\mathbf{x}^0$, stepsizes h, Energy function E
for $k = 0, 1, 2, ..., K-1$ **do**
  $\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k - h\nabla_\theta E\left(\mathbf{x}^k\right)$;
end
Return $\mathbf{x}^K$;



(a) GD scatter (b) SGLD scatter

Figure 4.2: scatters and trace for Gradient Descent (R Hat = 2891.63)



(a) SGLD scatter (b) SGLD trace

Figure 4.3: scatters and trace for SGLD (R Hat = 19.21)

4.1.2 SGLD

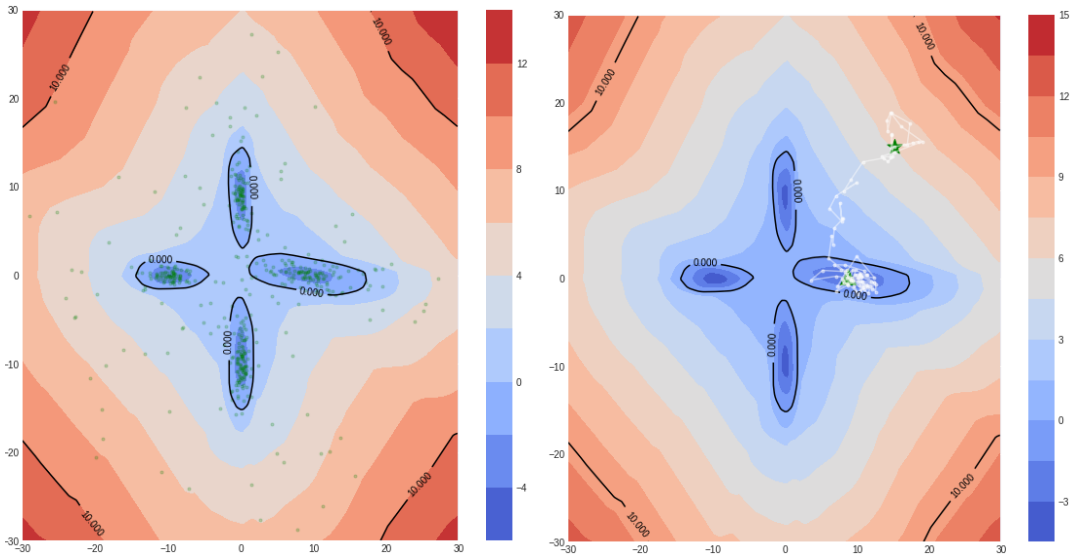
Compared with the traditional Gradient Descent, the main difference between Stochastic Gradient Langevin Dynamic (SGLD)[36], and Gradient Descent is that there is a noise term added for each transition. The result (Figure[4.3]) shows that the scatters will move to the area with low energy but not converge to the local minimum point. The R hat value for SGLD sampling is 19.21.

Algorithm 4: Stochastic Gradient Langevin Dynamic

Input: stepsizes h, $\mathbf{v} \sim \mathcal{N}(0,1)$, Energy function E
for $k = 0, 1, 2, ..., K-1$ **do**
  $\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k - h\nabla_\theta E\left(\mathbf{x}^k\right) + \sqrt{2h}\mathbf{v}$;
end
Return $\mathbf{x}^K$;

4.1.3 Metropolis-Adjusted Langevin Algorithm

Compared with SGLD, MALA (Figure: [4.4]) introduced the acceptance ratio. Metropolis-Hastings (as mentioned in page 9) was used in this method. We can understand the ratio as a threshold to accept or reject some stages. For a current stage x^k and a new stage $\tilde{x}^{k+1}$, if $\tilde{x}^{k+1}$ was accepted by the Metropolis Hastings, $x^{k+1} = \tilde{x}^{k+1}$. Else, $x^{k+1} = x^k$. R-hat evaluation value is 6.46. Compared with SGLD, this method can accelerate the sampling speed to locate at the region with low energy.



(a) MALA scatters graph (b) MALA trace graph

Figure 4.4: scatters and trace for MALA (R hat = 6.46)

Algorithm 5: Metropolis-Adjusted Langevin Algorithm

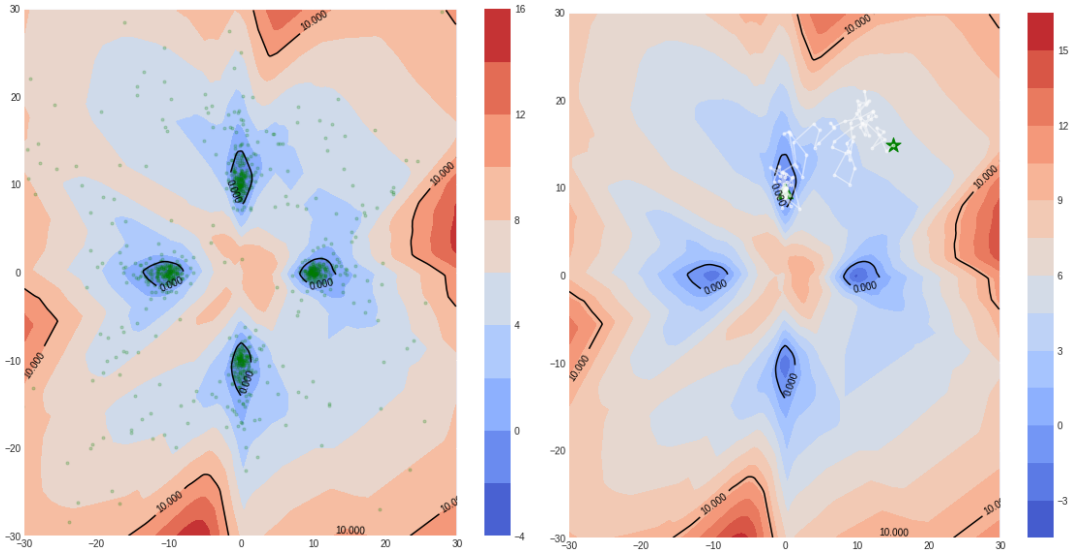
Input: acceptance ratio $u = 0.5$, stepsizes h, $\mathbf{v} \sim \mathcal{N}(0, 1)$
for $k = 0, 1, 2, ..., K - 1$ **do**
  $\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k - h\nabla_\theta E\left(\mathbf{x}^k\right) + \sqrt{2h}\mathbf{v}$;
  $\tilde{\mathbf{x}}^{k+1} \leftarrow \mathbf{x}^k - h\nabla_\theta E\left(\mathbf{x}^k\right) + \sqrt{2h}\mathbf{v}$;
  $\alpha := \min\left\{1, \frac{E(\tilde{x}_{k+1})q(x_k|\tilde{x}_{k+1})}{E(x_k)q(\tilde{x}_{k+1}|x_k)}\right\}$, where
  $q\left(x' \mid x\right) \propto \exp\left(-\frac{1}{4\tau}\left\|x' - x - h\nabla \log E(x)\right\|_2^2\right)$;
  Draw λ from the uniform distribution on $[0, 1]$;
  if $\lambda \leq \alpha : x^{k+1} := \tilde{x}^{k+1}$;
  else : $x^{k+1} := x^k$;
end
Return $\mathbf{x}^K$;

4.1.4 Preconditioned SGLD

Preconditioned SGLD(pSGLD)[29] is also an efficient method to accelerate the speed of sampling to the region with low energy. Compared with MALA, the main difference is that pSGLD sets the precondition, but not a threshold, to update the noise and stepsize, which seeks to create a local transform so the ratio of curvature is equal in all directions. Following this, pSGLD was proposed to use the same preconditioner as in RMSprop. This preconditioner merely estimates a diagonal matrix and is updated progressively using just the current gradient information. From the result (Figure [4.5]), compared with SGLD, we can notice that most scatter are located in the region with low energy more efficiency. Moreover, the trace graph suggests that the point will quickly locate to a region with low energy to accelerate the EBM training process. The corresponding R hat value is 6.49, which is much better than SGLD.

Algorithm 6: Preconditioned SGLD with RMSprop

Input: stepsizes h, $\mathbf{v} \sim \mathcal{N}(0, 1)$, $\alpha = 0.99$,$\lambda = 1e - 5$
Initialize: $\mathbf{V} \leftarrow \mathbf{0}$;
for $k = 1, 2, ..., K$ **do**
  Estimate gradient $grad = \nabla_\theta E\left(\mathbf{x}^k\right)$;
  $V\left(\boldsymbol{\theta}_k\right) \leftarrow \alpha V\left(\boldsymbol{\theta}_{k-1}\right) + (1 - \alpha) grad \odot grad$;
  $G\left(\boldsymbol{\theta}_k\right) \leftarrow \operatorname{diag}\left(\mathbf{1} \oslash \left(\lambda\mathbf{1} + \sqrt{V\left(\boldsymbol{\theta}_{t-1}\right)}\right)\right)$;
  $\tilde{\mathbf{x}}^{k+1} \leftarrow \mathbf{x}^k - G\left(\boldsymbol{\theta}_k\right) h\nabla_\theta E\left(\mathbf{x}^k\right) + \sqrt{2G\left(\boldsymbol{\theta}_k\right) h}\mathbf{v}$;
end
Return $\mathbf{x}^K$;



(a) PSGLD scatters graph (b) MALA trace graph

Figure 4.5: scatters and trace for pSGLD, R hat = 6.49

4.2 Image generation model

In this part, we implement different sampling methods to justify whether the MALA and PSLGD are suitable for image generation. To evaluate the result, the FID and IS scores are used to judge if the result is better than the result of Langevin dynamics.

Firstly, we need to implement the image generation model based on the Yilun Du[6][5]. In the subsequent experiment process, some essential components require our special attention.

Replay Buffer

As discussed before, replay buffer is a crucial point to achieve persistent contrastive divergence. Not only can it store the results of previous samples, more importantly, in the subsequent training iterations, but we also need to randomly extract the results of the last sampling from the replay buffer as the starting point for the next step of sampling. In this way, the chain length of the Markov chain is increased. Because of the existence of the replay buffer, we may be able to implement mini-batches and use small samples for training. At the same time, it also avoids wasting a lot of training time because of an excessively long Markov chain.

Step size and Variance

We need to break the relationship between the step size and the square in the Langevin dynamic. I quoted Yilun Du's code. The reference codes are:`https://`

`github.com/yilundu/improved_contrastive_divergence` and `https://github.com/karimul/Riset-EBM`. In his model, the sampling step is 100 for cifar-10 data set, and the standard deviation value is only 0.0001. In the experiment of Yilun Du[5], this is not the result of random setting but the optimal value extracted through a large number of experiments. One possible reason is that the complexity of the energy equation is high, and the value of the gradient is minimal. Since the training set is image data, if we only use a smaller step size in the sampling process, multiplying with a smaller gradient will likely cause the Markov chain to stay in a certain state.

Algorithm 7: Train the EBM with L2 loss

H **Input:** data distribution $p_D(x)$, step size k, steps N_s, training epoch N_e, replay buffer B, data augmentation $D(.)$, stop gradient $\Omega(.)$, Energy based model $E_\theta(.)$, noise scale n, mini-batch size S

$n = 0.0001$;
$B \leftarrow \emptyset$;
for *epoch* $i = 1$ *to* N_e **do**
  $\mathbf{x}^+ \sim p_D$;
  $\mathbf{x_0}^- \sim B$;
  ▷ : We need to use the data augmentation on the start point of sampling:$x_0{}^-$;
  $x_0{}^- = D(x_0{}^-)$;
  ▷ : Now we need to implement the sampling method to sample the negative samples for calculating the contrastive divergence.;
  ▷ : Take the Langevin Dynamics as an example;
  for *step* $j = 0$ *to* $N_s - 1$ **do**
    $\mathbf{x}_{j-1}{}^- = \Omega(\mathbf{x}_{j-1}{}^-)$;
    $\mathbf{x}_j{}^- \leftarrow \mathbf{x}_{j-1}{}^- - k\nabla E_\theta\left(\mathbf{x}_{j-1}{}^-\right) + v, v \sim \mathcal{N}(0, n)$
  end
  ▷ $\Omega(.)$ means stopping gradient;
  $\mathbf{x}^- = \Omega\left(\mathbf{x}_j^-\right)$;
  ▷ Optimize $\mathcal{L}_{CD} + \alpha\mathcal{L}_2$ with respect to θ ;
  $\mathcal{L}_{CD} = \frac{1}{S}\sum_{mini-batch}\left(E_\theta\left(\boldsymbol{x}^+\right) - E_\theta\left(\boldsymbol{x}^-\right)\right)$;
  $\mathcal{L}_2 = \frac{1}{S}\sum_{mini-batch}\left(E_\theta\left(\mathrm{x}^+\right)^2 + E_\theta\left(\mathrm{x}^-\right)^2\right)$;
  ▷ Optimize $\mathcal{L}_{CD} + \alpha\mathcal{L}_2$ with respect to θ ;
  $\Delta\theta \leftarrow \nabla_\theta(\mathcal{L}_{CD} + \alpha\mathcal{L}_2)$;
  Update θ based on $\Delta\theta$ using Adam optimizer;
  ▷Update replay buffer B;
  $B = B \cup x_i^-$
end

The length of chain in one batch

In the sampling process, we have to face a genuine problem, which is time. If we train a mini-batch each time, we have to set up a Markov chain with a chain length of

nearly 1000 for sampling, then it will be a very time-consuming task, especially for image data. Take the training process of Cifar-10 as an example. If we use Cifar-10 for training and set the size of the mini-batch to 128, then if the length of the chain is set to 60 and training on an RTX 3090 GPU with 21G video memory, the training time length of a mini-batch is 5s. If the chain length is set to 1000, the training time of a mini-batch will be close to one minute.

4.2.1 Implement different sampling method for image generation

Based on the experiments on Gaussian scatters, we will implement the energy-based model on the image data set. There are three different sampling methods.

Algorithm 8: Langevin Dynamic for image

H **Input:** step size h, negative sample x^0 from Replay buffer, chain length K, noise scale $n = 0.0001$, Energy based model $E_\theta(.)$

for $k = 0, 1, 2, ..., K-1$ **do**

$v \sim N(0, n)$;

$\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k - h\nabla_\theta E\left(\mathbf{x}^k\right) + \mathbf{v}$;

end

Return $\mathbf{x}^{K-1}$;

Algorithm 9: Preconditioned Langevin dynamics for image

H **Input:** step size h, negative sample x^0 from Replay buffer, chain length K, noise scale $n = 0.0001$, Energy based model $E_\theta(.)$, $\alpha = 0.9$,$\lambda = 1$

Initialize: $\mathbf{V} \leftarrow \mathbf{0}$;

for $k = 1, 2, ..., K$ **do**

$v \sim N(0, n)$;

Estimate gradient $grad = \nabla_\theta E\left(\mathbf{x}^k\right)$;

$V \leftarrow \alpha V + (1-\alpha) grad \odot grad$;

$G \leftarrow \text{diag}\left(\mathbf{1} \oslash \left(\lambda \mathbf{1} + \sqrt{V}\right)\right)$;

$\tilde{\mathbf{x}}^{k+1} \leftarrow \mathbf{x}^k - Gh\nabla_\theta E\left(\mathbf{x}^k\right) + v$;

end

Return $\tilde{\mathbf{x}}^{K-1}$;

Now we will experiment with the Energy-based model on images based on algorithm 7 with three different sampling methods (Algorithms 8 to 10). Among them, we use FID and IS values as the main criteria for evaluation. For the hyper-parameters, all settings are the same for these three different experiments: step size $s = 100$, variance $v = 0.0001$, chain's length $l = 60$, learning rate for model optimization $lr = 0.0001$.

Algorithm 10: Tempering MALA

H **Input:** Minimum threshold of acceptance ratio $u = 0.5$, step size h, negative sample x^0 from Replay buffer, chain length K, noise scale $n = 0.0001$, Energy based model $E_\theta(.)$, temperature $tem = 1$

for $k = 0, 1, 2, ..., K-1$ **do**

$E = E\,/tem\;\; v \sim N(0, n)$;

$\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k - h\nabla_\theta E\left(\mathbf{x}^k\right) + v$;

$\tilde{\mathbf{x}}^{k+1} \leftarrow \mathbf{x}^k - h\nabla_\theta E\left(\mathbf{x}^k\right) + v$;

$\alpha := \min\left\{1, \frac{E(\tilde{x}_{k+1})q(x_k|\tilde{x}_{k+1})}{E(x_k)q(\tilde{x}_{k+1}|x_k)}\right\}$, where

$q\left(x' \mid x\right) \propto \exp\left(-\frac{1}{4\tau}\left\|x' - x - h\nabla \log E(x)\right\|_2^2\right)$;

Draw λ from the uniform distribution on $[0, 1]$;

if $\lambda \leq \alpha$: $x^{k+1} := \tilde{x}^{k+1}$;

else : $x^{k+1} := x^k$;

if $\alpha \leq u$: $tem = tem \times 10$;

end

Return $\mathbf{x}^{K-1}$;

Based on the FID curve graph 4.6, only using the Langevin Dynamics as the sampling method has a good effect. But it is not reasonable and different with the result of Gaussian scatters. It is very time consuming for training the energy based model to achieve the best performance. If we want to achieve a great performance for a data set, we need to set the epoch as 100 and the training time is about 1 to 2 days on V100 with 32G video memory. Table 4.1 shows the result on four different data sets with four different sampling methods. All the results are based on the unconditional energy-based model.

Table 4.1: FID results on four data set with different sampling methods (training 100 epoch on each trying)

FID	Cifar-10	SVHN	Mnist	Cats
Langevin dynamics	61	68	117	76
MALA	375	311	274	311
Tempering MALA	413	362	308	324
Precondition(Lambda = 1)	107	155	173	161

Based on the result of MALA, no matter if the MALA is tempered, the acceptance ratio is always near to 0 in the second half of training. An acceptance ratio close to 0 means that the new state of sampling will always be rejected. This result is exceptionally unreasonable. Because logically, not all sampling images should be rejected. Moreover, during the SGLD sampling process, I also calculated the acceptance ratio but did not use this mechanism to reject or accept the new state. In the first 100 batches of training using SGLD, the acceptance ratio was not high at the beginning and remained between 0.7 and 1.0. But after the 100th batch training,

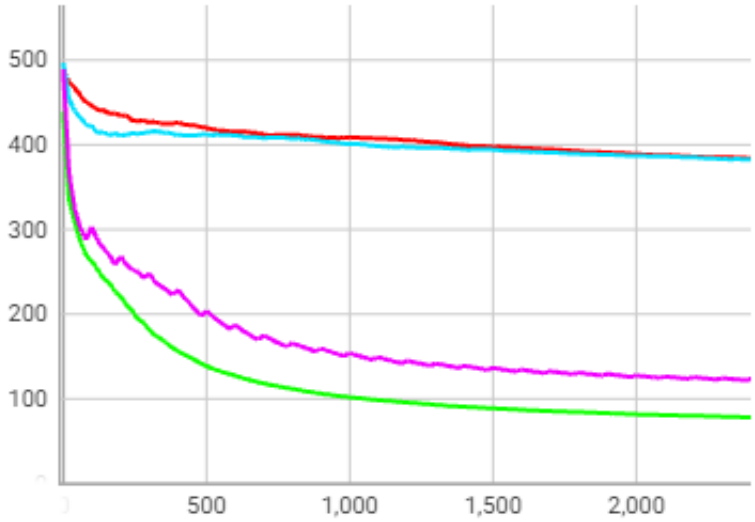
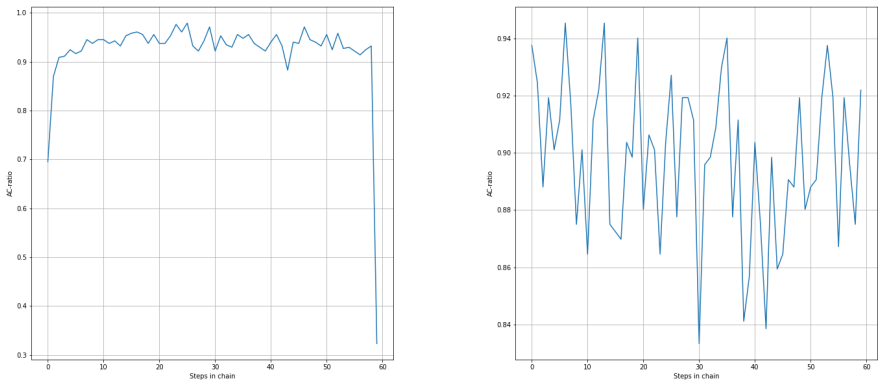


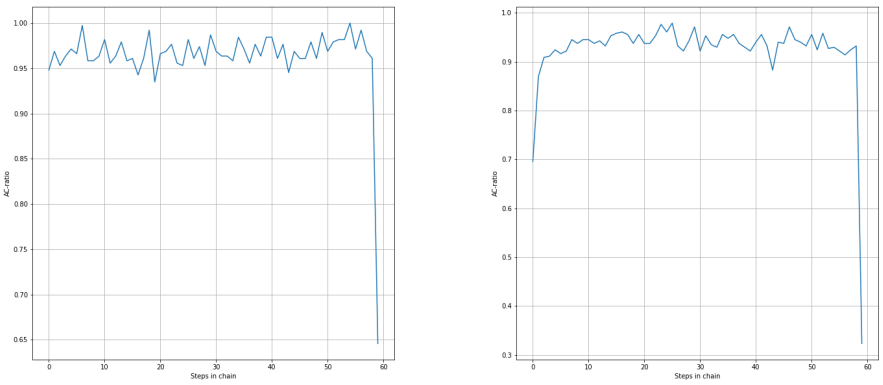
Figure 4.6: FID curve for the first epoch on cifar-10. Green: SGLD, Pink: PSGLD, Blue: tempering MALA, Red: MALA. X-axis: the iteration of mini batch, Y-axis: FID

the acceptance ratio has been maintained between 0.95 and 0.99. As for MALA, in the first 100 batches, the acceptance ratio is the same as SGLD. But after the 100th batches, the acceptance ratio will decrease gradually and will close to 0 in the end as shown in figure 4.11 and 4.8, which shows the acceptance ratio changes in one chain, both SGLD and MALA has the same acceptance ratio during the first 100 batch training process.



(a) The 0th batch (b) The 50th batch

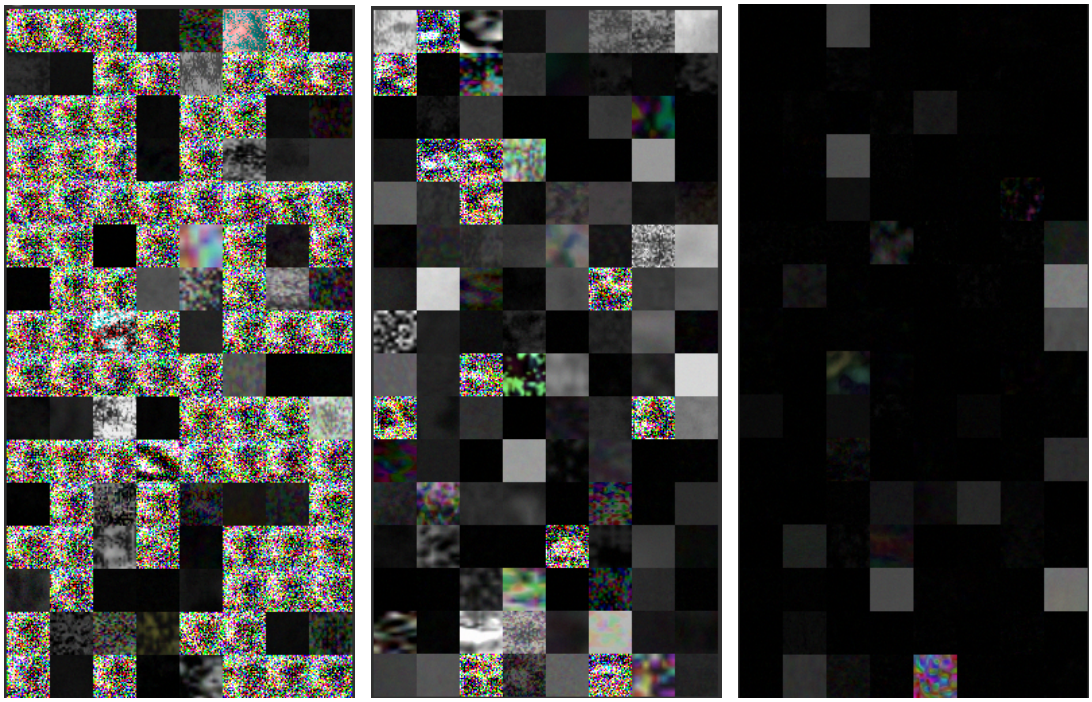
Figure 4.7: SGLD acceptance ratio but not used in sampling



(a) The 0th batch (b) The 50th batch

Figure 4.8: MALA acceptance ratio and used in sampling

But since the 150th batch, the acceptance ratio of MALA will decrease but there are no changes on SGLD, which means if the sampling methods begin to accept or reject samples for a long time, the ac ratio will decrease significantly.

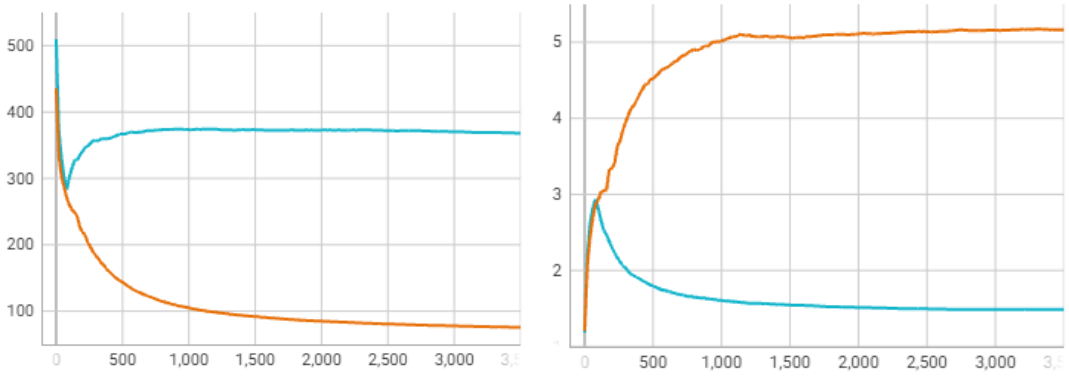


(a) At the beginning (b) Mid-training (c) End of training

Figure 4.9: MALA will always reject with the training going on.

But it is also possible to explain this performance in another way; when MALA is not given a proper initial distribution, the energy model will produce a false distribution,

which will lead to the phenomenon of constant rejection. So if we use the SGLD as a sampling method for the first 100th batches to give the EBM an initial distribution, the MALA might work. But in the experiment, the effect is not as we thought. As shown in Figure **??**, once when we use mala, the FID will start to rise. We also tried to set more than 100 batches to use SGLD, such as the first 1000 and the first 5000. But the result is the same, MALA will cause the value of FID to rise all the time, and the resulting image will become more and more blurred. We also tried to set more than 100 batches to use SGLD, such as the first 1000 and the first 5000. But the result is the same, MALA will cause the value of FID to rise all the time, and the resulting image will become more and more blurred.

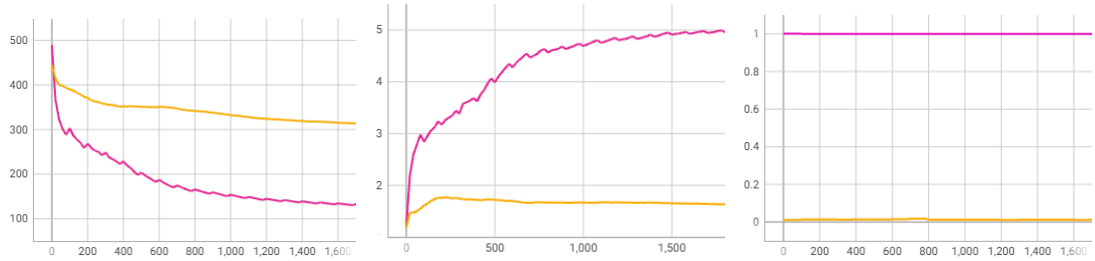


(a) FID curve (b) IS curve

Figure 4.10: Blue: SGLD used in first 100th batch and MALA used in the later; Yellow: Always SGLD

As for the precondition Langevin Dynamics, the value of precondition parameter G should have a significant increasing trend, but in the experiment result, it will only increase a little, and the magnitude of G is tiny, remaining at around 0.001. Please note that the value of our step size itself is enormous, set to 100. If the value of G is small, as the denominator of the original step size, the new step size (step size $s = s/G$) will be enormous, which is illogical. Because the role of G is to dynamically change the step size, allowing the Markov chain to stay more in low-energy areas.

There is a possibility that the value of Lambda is not set correctly. Because when calculating G, we need to add a Lambda to prevent G from becoming 0. In the experiment, we can limit the value of G to an approximate value of 1 by setting $Lambda = 1$. But this loses the original meaning of precondition. Because before adding Lambda, the value of precondition has been about 0.001. After adding 1, it is also a value close to 1, which will almost not affect the step size.



(a) FID curve (b) IS curve (c) 1/G curve

Figure 4.11: Pink: PSGLD Lambda = 1.0; Yellow: Lambda = 0.01

4.2.2 Implement different sampling method for image classification

As for the Joint energy based model, the main difference is that the model is optimize with respect to the $\frac{\partial \log prop_\theta(\mathrm{x,y})}{\partial \theta}$, and there is a equation based on the section 2.4.

$$\log prop_\theta(x, y) = \log prop_\theta(x) + \log prop_\theta(y \mid x) \tag{4.1}$$

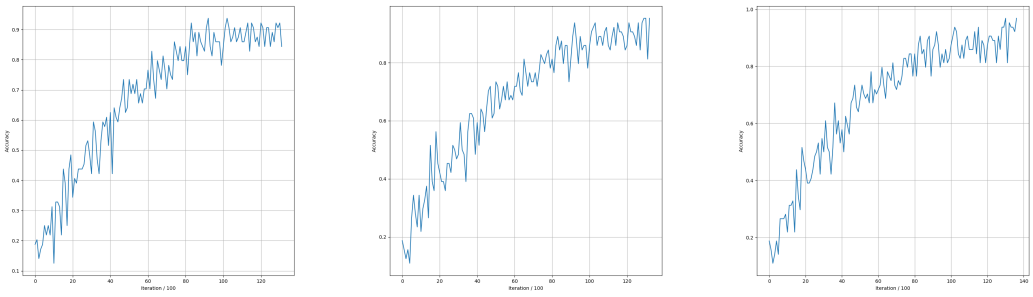
For the second term, we can calculate it by standard cross-entropy (classification result with real label). For the first term, we need to use some sampling methods to do contrastive divergence to estimate the gradient with respect to $\log prop_\theta(\mathrm{x})$.

$$\frac{\partial \log prop_\theta(\mathrm{x})}{\partial \theta} = E_{prop_\theta(\mathrm{x}')}\left[\frac{\partial E_\theta(\mathrm{x}')}{\partial \theta}\right] - \frac{\partial E_\theta(\mathrm{x})}{\partial \theta}$$

There is almost no difference with different sampling methods on the Cifar-10 data set, as the table 4.2 shown.

Table 4.2: Accuracy on Cifar-10 with different sampling methods (training 100 epoch on each trying)

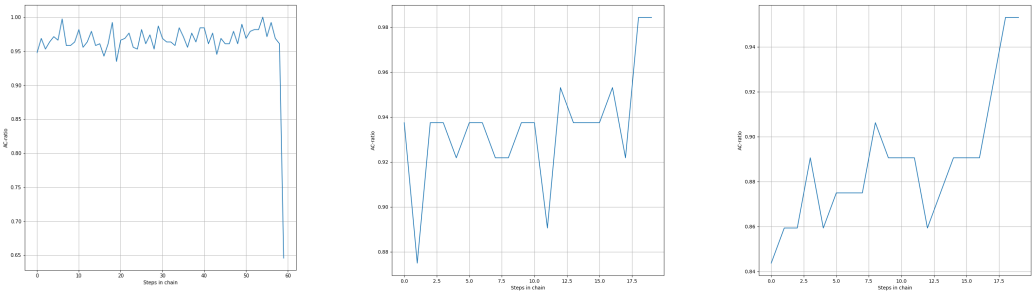
Cifar-10	SGLD	PSGLD(Lambda = 1)	MALA
Accuracy	0.89	0.89	0.88
FID	57.1	59.4	56.4



(a) SGLD (b) PSGLD (c) MALA

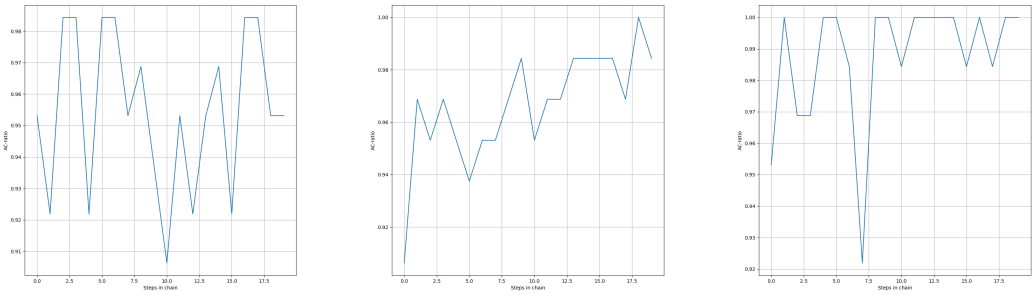
Figure 4.12: Accuracy curve for three different sampling methods

From the result 4.12, there are almost no clear differences between the four sampling methods. It is also difficult to understand and the performance is not in logic.



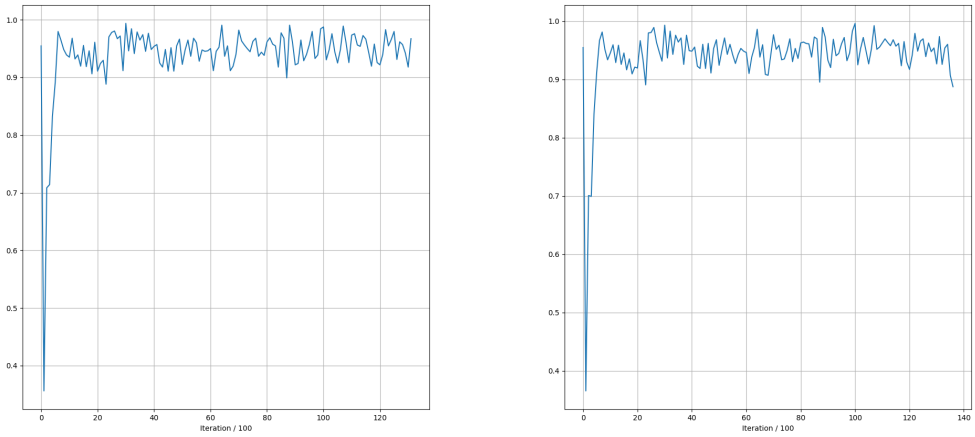
(a) 0th (b) 5000th (c) 10000th

Figure 4.13: Acceptance ratio (in chain) for MALA in 0th, 5000th, 10000th iteration



(a) 0th (b) 5000th (c) 10000th

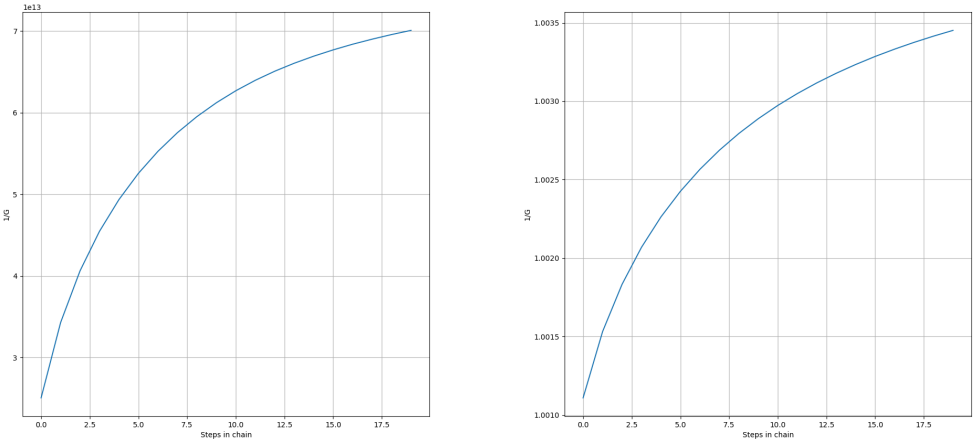
Figure 4.14: Acceptance ratio (in chain) for Langevin Dynamics in 0th, 5000th, 10000th iteration



(a) SGLD (b) MALA

Figure 4.15: The overall trend of acceptance ratio for Langevin Dynamics and MALA

We also collect the acceptance ratio (in the chain) for MALA (Figure 4.13) and SGLD (Figure 4.14), where MALA used the ratio to accept or reject but SGLD did not use it. The overall trend curve of the acceptance ratio has shown in Figure 4.16, which shows there is almost no difference for the two sampling methods. For MALA, the training process is slow because for each sampling step because compared with SGLD sampling, and the MALA needs one more step to calculate the gradient of the proposal state $\tilde{x}$. From the result, we can notice that MALA might not be needed in JEM because both MALA and SGLD have the same result, but MALA needs more time to sample.



(a) Lambda = 0.01 (b) Lambda = 1

Figure 4.16: The value of $1/G$ for Lambda = 0.01 or 1

For PSGLD, there are some interesting points. If we set the Lambda as 1, the value of G is also close to 1, which can explain the reason why the performances of PSGLD and SGLD are similar to each other. But if we set the Lambda as 0.0001, the accuracy after the first epoch will achieve 0.36, but the accuracy of SGLD after the first epoch will only achieve 0.26. However, with the training process going on, the value of G will increase to a terrible number (Figure 4.16), and the loss value will always suddenly diverge to $1e^8$ in one batch. Then the energy distribution will be destroyed as well.

This performance shows that it might be beneficial for the JEM to be set a huge step size during the first one epoch training process. But as for the reason why the loss will diverge suddenly, one of the possible reasons is that the large step size destroys the energy distribution and makes the distribution pretty steep.

Chapter 5

Conclusion

We designed the energy-based model in the experimental module and trained it with Gaussian scattered points as the data set. SGLD, MALA, PSGLD methods are used in sampling methods. We verified the usability of MALA and PSGLD in the simple energy model. And compared with SGLD, the MALA and PSGLD positively influence the sampling result and training process.

For image generation by EBM

We also refer to Yilun Du's code and replace the SGLD method by using MALA, tempering MALA, and PSGLD. However, these sampling methods initially thought to be effective have brought very negative effects on the model.

MALA will always reject the new state in the second half of training and the acceptance ratio. Regardless of whether the energy is tempered or step size is annealed, the acceptance ratio of MALA will gradually approach 0. This result also means that the Markov chain is stuck at a certain point and cannot perform state transfer.

For PSGLD, because the gradient is minimal, the value of the preconditioned matrix will remain at a minimum order of magnitude. During the experiment, the value of the preconditioned matrix has been maintained at about 0.0001 and the original step size $s = 100$. However, the update rule of step size s is: $s = s \times G$; a small precondition matrix will cause the new step size s to become extremely small.

However, please note that all sampling methods use a huge step size $s = 100$. But in the inference of MALA and PSGLD, the step size s should be limited to $0 < s << 1$. Moreover, if and only if the step size in this interval, the MALA and PSGLD can positively affect the model. However, during the training process, if we set the step size in this interval, such as step size $s = 10^{-5}$, the model will not train because there is only a very slight transition distance in the total Markov Chain. One of the main reasons is that the gradient of the energy-based model we used is too small, which means we have to use a considerable step size, such as 100, to make up for

the negative effect of the slight gradient.

For image classification by EBM

Based on the experimental results, I think the main reason for the negative impact is to use an inappropriate step size. And this idea got barely plausible support in JEM's experiment. In the JEM experiment, I used the official code of the author of JEM Will Grathwohl, and the default step size in the code is 1. Compared to 100, 1 is much smaller. I think this is one of the main reasons MALA can be used in JEM and not cause negative effects. Although MALA does not play any role in JEM, it will even extend the training time, and it will take more time to get the same accuracy as Langevin dynamics, but it will not cause problems that happened in the image generation model.

Moreover, the preconditioned matrix plays a minimal role on JEM as well. The original intention of the precondition design is to increase the step size in areas with high energy and reduce the step size in areas with low energy. But in the image classification part, because the step size we set is 1, and the value of the preconditioned matrix is often around 10^{-3}, if the lambda is set as 0.001, the updated step size s is about 1000, which will destroy the sampling process. And if the lambda is set as 1, there will be no difference between the updated step size and the original step size. To sum up, the method of precondition will overly rely on the value of lambda.

However, is it feasible to set the step size to a smaller value and then match the precondition matrix value? Assuming that the value of precondition matrix G is 10^{-2}, and we can set the value of step size to 100 and then keep $s \times g$ around 1. But this is not feasible. And the effect of training is not as good as SGLD.

The main reason is that the preconditioned matrix has a small value and has a small range of change. Taking the above setting as an example, if we want to keep the $s \times g$ about 1, the approximate change interval of the value of $s \times g$ is only 1.1 to 0.9. This method has a negative effect on training and also lengthens the training time required for search.

Chapter 6

Future work

Energy function architecture

After the experimental results, I think that if we want to apply MALA or precondition in the sampling process of the energy-based model, the most important thing is to design a sampling method that can use a small step size for sampling in the energy model, which means the architecture might need to be modified. And the desired neural network should not be deep but wide.

Try to prevent the MCMC from being trapped.

It is more like a preset for Langevin dynamics that uses a large step size, which transfers the Markov chain through forced methods from one area to another. With training, energy distribution gradually takes shape, and this mandatory effect has also been weakened to a certain extent. If we use MALA or PSGLD as the sampling methods, the Markov chain might be trapped in a place with very low energy. Because MALA will reject the new state, and PSGLD will make the value of G tremendous, which will cause the Markov chain to be unable to escape this area. And the energy in this area will become lower and lower as a result.

Therefore, logically, when SGLD sampling can no longer optimize the model, we can use MALA as a new sampling method to continue training the model. But the premise is to use a smaller step size and maintain the relationship between step size and variance.

Larger batch size

For the initial distribution, although I tried to use the 50th epoch model for MALA training, it brought negative effects to the model, and the FID would get higher and higher. However, one possible main reason is that the batch size is small. Due to hardware memory limitations, the size of the mini-batch cannot be set to 500 or higher. If we can have a larger batch size and a very mature initial distribution, we may use MALA in the sampling method.

References

[1] Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006. pages 3, 5

[2] Taesup Kim and Yoshua Bengio. Deep directed generative models with energy-based probability estimation. *arXiv preprint arXiv:1606.03439*, 2016. pages 3

[3] Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002. pages 3, 4, 8, 26

[4] Tijmen Tieleman. Training restricted boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th international conference on Machine learning*, pages 1064–1071, 2008. pages 4

[5] Yilun Du, Shuang Li, Joshua Tenenbaum, and Igor Mordatch. Improved contrastive divergence training of energy based models. *arXiv preprint arXiv:2012.01316*, 2020. pages 4, 25, 26, 38, 39

[6] Yilun Du and Igor Mordatch. Implicit generation and modeling with energy based models. 2019. pages 5, 10, 26, 27, 38

[7] Martin J Wainwright and Michael Irwin Jordan. *Graphical models, exponential families, and variational inference*. Now Publishers Inc, 2008. pages 6

[8] Nathaniel R Goodman. Statistical analysis based on a certain multivariate complex gaussian distribution (an introduction). *The Annals of mathematical statistics*, 34(1):152–177, 1963. pages 6

[9] Adrien Wohrer. Ising distribution as a latent variable model. *Physical Review E*, 99(4):042147, 2019. pages 6

[10] Richard O Duda, Peter E Hart, and Nils J Nilsson. Subjective bayesian methods for rule-based inference systems. In *Readings in artificial intelligence*, pages 192–199. Elsevier, 1981. pages 7

[11] Md Mostafa Kamal Sarker, Hatem A Rashwan, Farhan Akram, Syeda Furruka Banu, Adel Saleh, Vivek Kumar Singh, Forhad UH Chowdhury, Saddam Abdulwahab, Santiago Romani, Petia Radeva, et al. Slsdeep: Skin lesion segmentation based on dilated residual and pyramid pooling networks. In

International Conference on Medical Image Computing and Computer-Assisted Intervention, pages 21–29. Springer, 2018. pages 7

[12] Yang Song and Diederik P Kingma. How to train your energy-based models. *arXiv preprint arXiv:2101.03288*, 2021. pages 7, 8

[13] Laurent Younes. On the convergence of markovian stochastic algorithms with rapidly decreasing ergodicity rates. *Stochastics: An International Journal of Probability and Stochastic Processes*, 65(3-4):177–228, 1999. pages 7

[14] Random walks. URL `https://www.mit.edu/~kardar/teaching/projects/chemotaxis(AndreaSchmidt)/random.htm`. pages 8, 14

[15] Ulf Grenander and Michael I Miller. Representations of knowledge in complex systems. *Journal of the Royal Statistical Society: Series B (Methodological)*, 56 (4):549–581, 1994. pages 8

[16] Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688. Citeseer, 2011. pages 8

[17] Richard Turner. Cd notes. 2005. URL `http://www.gatsby.ucl.ac.uk/~turner/Notes/ContrastiveDivergence/CDv3.pdf`. pages 8

[18] Tijmen Tieleman. Training restricted boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th international conference on Machine learning*, pages 1064–1071, 2008. pages 10, 11, 26

[19] Yi-An Ma, Yuansi Chen, Chi Jin, Nicolas Flammarion, and Michael I Jordan. Sampling can be faster than optimization. *Proceedings of the National Academy of Sciences*, 116(42):20881–20885, 2019. pages 12

[20] Random walks. URL `http://www.gatsby.ucl.ac.uk/teaching/courses/ml1-2015/lect12-handout.pdf`. pages 12, 13, 14

[21] Alan E Gelfand. Gibbs sampling. *Journal of the American statistical Association*, 95(452):1300–1304, 2000. pages 13

[22] Don Van Ravenzwaaij, Pete Cassey, and Scott D Brown. A simple introduction to markov chain monte–carlo sampling. *Psychonomic bulletin & review*, 25(1): 143–154, 2018. pages 13, 14

[23] Siddhartha Chib and Edward Greenberg. Understanding the metropolis-hastings algorithm. *The american statistician*, 49(4):327–335, 1995. pages 14

[24] Wenbo Gong, Yingzhen Li, and José Miguel Hernández-Lobato. Meta-learning for stochastic gradient mcmc. *arXiv preprint arXiv:1806.04522*, 2018. pages 15

[25] Erik Nijkamp. *A Dance with the Langevin Equation*. PhD thesis, UCLA, 2018. pages 15

[26] Bart G de Grooth. A simple model for brownian motion leading to the langevin equation. *American journal of physics*, 67(12):1248–1252, 1999. pages 15

[27] Luke Tierney. A note on metropolis-hastings kernels for general state spaces. *Annals of applied probability*, pages 1–9, 1998. pages 16

[28] Mark Girolami and Ben Calderhead. Riemann manifold langevin and hamiltonian monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):123–214, 2011. pages 18

[29] Chunyuan Li, Changyou Chen, David Carlson, and Lawrence Carin. Preconditioned stochastic gradient langevin dynamics for deep neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016. pages 18, 37

[30] Andrew Gelman, Donald B Rubin, et al. Inference from iterative simulation using multiple sequences. *Statistical science*, 7(4):457–472, 1992. pages 18

[31] Notation for samples, chains, and draws. URL `https://mc-stan.org/docs/2_21/reference-manual/notation-for-samples-chains-and-draws.html`. pages 19

[32] Will Grathwohl, Kuan-Chieh Wang, Jörn-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swersky. Your classifier is secretly an energy based model and you should treat it like one. *arXiv preprint arXiv:1912.03263*, 2019. pages 21, 22, 23, 27

[33] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016. pages 24

[34] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):1–48, 2019. pages 25

[35] Chandrasekaran Anirudh Bhardwaj. Adaptively preconditioned stochastic gradient langevin dynamics. *arXiv preprint arXiv:1906.04324*, 2019. pages 27

[36] Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688. Citeseer, 2011. pages 27, 36

[37] Rohitash Chandra, Konark Jain, Ratneel V Deo, and Sally Cripps. Langevin-gradient parallel tempering for bayesian neural learning. *Neurocomputing*, 359:315–326, 2019. pages 28

[38] Frank Spitzer. *Principles of random walk*, volume 34. Springer Science & Business Media, 2013. pages 29

[39] Yves F Atchadé. An adaptive version for the metropolis adjusted langevin algorithm with a truncated drift. *Methodology and Computing in applied Probability*, 8(2):235–254, 2006. pages 29