Topics in Neural Networks and Generative Modeling

Shounak Desai
Department of Computer Science
Golisano College of Computing and Information Sciences
Rochester Institute of Technology
Rochester, NY 14586
sd9649@rit.edu

***Index Terms*—Probability Theory; VAEs; GANs; DDPMs**

I. Introduction

This report covers all the contents covered in the CSCI - 799 Independent Study. The topic of the Study is Neural Networks and Generative Modeling. We covered Probability Theory, Variational AutoEncoders (VAEs), Generative Adversarial Networks (GANs), Denoising Diffusion Probabilistic Models (DDDPM) over the course of Spring 24' semester. This report will have the Literature Review of related papers of all the mentioned topics, Implementation of all these Models with their results and a summary of the course.

Understanding generative models in vision is pivotal for academia research as it provides a deep insight into probabilistic modeling, Bayesian inference, and latent variable modeling. It fosters a better grasp of complex generative processes underlying natural data distributions, facilitating the exploration of novel algorithms, model architectures, and optimization techniques. Mastery of generative models also opens avenues for investigating theoretical concepts like information theory, representation learning, and unsupervised learning, driving forward the frontiers of academic research in machine learning and computer vision.

II. Concept Review

A. Probability Theory

Understanding the foundational mathematics is crucial in generative models, making a strong grasp of probability theory essential. To delve into these concepts, we consulted Murphy's 2012 [12] book. Initially, we focused on fundamental formulas such as those for Naive Bayes, Discrete and Continuous distributions, Joint probability distributions. Let x be a random variable of interest, and let $p(x)$ be its density. For example, x could be an image and $p(x)$ could describe the distribution of possible natural images or the images in the given dataset.

Kullback-Leibler (KL) divergence is used to measure the difference between two probability distributions, quantifying how one distribution diverges from another. KL divergence is given by:

$$\mathrm{KL}(p\|q) = \sum_{x\in\mathcal{X}} p(x)\log\frac{p(x)}{q(x)} \tag{1}$$

Mutual information quantifies the amount of information that is common to two random variables, *X* and *Y*, measuring how much knowing one variable reduces uncertainty about the other. It is given by:

$$I(X;Y) = \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log\frac{p(x,y)}{p(x)p(y)} \tag{2}$$

where $p(x,y)$ is a joint probability distribution and $p(x)$ and $p(y)$ are marginal probabilities for x and y.

Maximum Likelihood Estimation (MLE) is a method to estimate the parameters of a statistical model by maximizing the likelihood function, finding the values that make the observed data most probable under the model. It is given by:

$$\hat{\theta}_{\text{MLE}} = \arg\max_{\theta}\prod_{i=1}^{n} p(x_i;\theta) \tag{3}$$

where p is a probability density function and θ are model's parameter that we want to estimate such as mean or variance.

MLE is a method used in statistical modeling to find parameter values that fit the distribution to some observed data. By taking the logarithm of the likelihood function, the calculations become simpler, and it is often used for optimization and parameter estimation in statistical inference. It is given by:

$$L(\theta) = \sum_{i=1}^{n}\log p(x_i;\theta) \tag{4}$$

As the course progressed and shifted towards generative models, we delved into topics like Evidence Lower Bound (ELBO), Monte Carlo inference, Importance Sampling, and Langevin Sampling.

The Evidence Lower Bound (ELBO) is a key concept in Variational Inference, representing a lower bound on the log likelihood of the observed data. It is used as an objective function to optimize the parameters of the variational distribution, balancing model complexity and data fit.

The Evidence Lower Bound (ELBO) is related to Maximum Likelihood Estimation (MLE) through the framework of variational inference in probabilistic models. ELBO serves as a lower bound on the log-likelihood of observed data, allowing for tractable estimation of complex posterior distributions. In variational inference, maximizing the ELBO is equivalent to minimizing the Kullback-Leibler (KL) divergence between the true posterior and an approximating distribution. This approach contrasts with MLE, which directly maximizes the log-likelihood of data, often leading to intractable solutions for complex models. ELBO thus provides a computationally feasible method for approximating posterior distributions in Bayesian inference. ELBO is give as,

$$\text{ELBO}(\theta, \phi) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{z \sim q_\phi(z|x)} \left[\log \frac{p(x, z; \theta)}{q_\phi(z|x)} \right] \quad (5)$$

where N is the total number of data points.

Monte Carlo inference is a method in Bayesian statistics that uses Monte Carlo sampling techniques to approximate posterior distributions or compute expectations in high-dimensional spaces, particularly when analytical solutions are intractable or computationally expensive. It involves generating samples from the posterior distribution and using these samples to estimate properties of interest, such as means, variances, or probabilities.

$$\mathbb{E}[f(X)] = \int f(x)p(x)\,dx \approx \frac{1}{S} \sum_{s=1}^{S} f(x_i) \quad (6)$$

where x_i are independent samples drawn from the distribution $p(x)$ and S is the number of samples.

Importance Sampling, a Monte Carlo technique for estimating properties of a target distribution by sampling from a different, easier-to-sample distribution. It assigns more weight to samples that are more likely under the target distribution, improving estimation accuracy for rare events or regions.

$$\mathbb{E}_{x \sim q_\phi}[f(x)] = \frac{1}{N} \sum_{i=1}^{N} \frac{p_\theta(x_i)}{q_\phi(x_i)} f(x_i) \quad (7)$$

Langevin Sampling: A Markov Chain Monte Carlo (MCMC) method used for sampling from probability distributions, particularly in Bayesian inference. It involves using a Langevin dynamics-based algorithm that combines random noise with gradient information to explore the distribution and generate samples.

$$x_{t+1} = x_t + \eta \nabla \log p_\theta(x_t) + \sqrt{2\eta}\xi_t \quad (8)$$

B. Variational Autoencoders (VAEs)

Variational Autoencoder (VAE) stands out as one of the pivotal topics in advanced Generative Models. The foundational work in this area traces back to Dayan's 1997 paper [3]. It provides foundational insights into hierarchical probabilistic models, focusing on how the brain perceives and recognizes patterns through hierarchical representations. This concept aligns with Variational Autoencoders (VAEs) in the sense that VAEs also operate on hierarchical latent spaces, where the encoder learns to represent data in a lower-dimensional latent space, akin to the hierarchical representations discussed in Dayan's work. Furthermore, the idea of recognition and inference in hierarchical models resonates with VAEs' encoding process, where the model learns to infer latent variables that best explain the observed data, thus establishing a conceptual link between Dayan's insights and the principles underlying VAEs.

Subsequently, Kingma and Welling's 2014 paper [10] , focusing specifically on VAEs, provided a detailed exposition of the VAE architecture and the crucial concept of the Evidence Lower Bound (ELBO) as its objective function. The Objective function of ELBO in this paper was give as,

$$\text{ELBO}(\theta, \phi) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x_i|z)] - \text{KL}[q_\phi(z|x)||p(z)] \quad (9)$$

here, $\text{ELBO}(\theta, \phi)$ represents the Evidence Lower Bound with parameters θ and ϕ. $\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x_i|z)]$ represents Reconstruction loss in VAE. $\text{KL}[q_\phi(z|x)||p(z)]$ represents the Kullback-Leibler (KL) divergence between the variational distribution $q_\phi(z|x)$ and the prior distribution $p(z)$ which represents Regularization Term. N represents the total number of datapoints.

The approximate posterior distribution $q(z|x)$ is often parameterized as a Gaussian distribution with a diagonal covariance matrix. This means that each dimension of the latent variable z s assumed to be independent and is characterized by its own mean and variance. The mean μ is given out as an output by the neural network as

$$\mu(x) = w_\mu(x)h(x) + b_\mu \quad (10)$$

where w is the weight matrix, b is the bias vector and $h(x)$ is the output of the neural network and the variance of $q(z|x)$ is given as

$$\begin{aligned} \sigma^2(x) &= \exp(z_{\sigma^2}(x)) \\ z_{\sigma^2}(x) &= w_{\sigma^2}(x)h(x) + b_{\sigma^2} \end{aligned} \quad (11)$$

where $z_{\sigma^2}(x)$ is the output of the neural network (interpreted as a log variance).

Thus, $q(z|x)$ can now be given as

$$q(z|x) = N(\mu(x), \sigma^2(x)) \quad (12)$$

After we get the distribution for $q(z|x)$ the reparameterization trick is used to enable the backpropagation of gradients through the stochastic sampling process. We have a latent variable z sampled from distribution of $q(z|x)$. The reparameterization trick rewrites the sampling operation as a deterministic function of a random variable ϵ sampled from a

fixed distribution $p(\epsilon)$ where ϵ is often sampled from a Normal distribution. Mathematically each sampled z is given as,

$$z = \mu(x) + \sigma(x)\epsilon \tag{13}$$

and this makes the end-to-end math ready to implement inside a neural network.

These seminal papers collectively offered a comprehensive understanding of both the theoretical underpinnings and practical implementation aspects of VAEs.

In delving into advanced Variational Autoencoder (VAE) techniques, I explored two key areas: Beta-VAE and Importance Weighted Autoencoders (IWAE). n theory, the IWAE advantage comes from a "tighter ELBO".

$$\text{Log Likelihood} \geq \text{ELBO\_IWAE} \geq \text{ELBO\_VAE}$$

so maximizing the IWAE ELBO should give you a "better model fit" in the sense of higher log likelihood. Beta-VAE is another critical aspect of advanced VAEs, as highlighted in Higgins et al. 2017 [6], it is the strategic multiplication of a scalar quantity β with the regularization term, a practice that significantly improves VAE efficacy. The objective function for Beta-VAE is,

$$\mathcal{L}(\theta, \phi) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x_i|z)] - \beta * \mathrm{KL}[q_\phi(z|x)||p(z)] \tag{14}$$

where β is the regularization hyper-parameter.

Leveraging my foundational understanding of Importance Sampling from Murphy's, which greatly enhances VAE performance, I delved into the Burda et al. 2015 paper [2], which elucidates how importance sampling techniques augment VAE accuracy. These insights deepened my comprehension of cutting-edge VAE methodologies and their practical implications. In particular, I learned that Importance Weighted Autoencoders (IWAE) tighten the bound compared to the normal VAE ELBO, contributing significantly to the model's fidelity in capturing complex data distributions with more than 1 samples drawn from Latent distribution represented by *K*.

$$\text{IWAE}(\theta, \phi) = \mathbb{E}_{z\sim q_\phi(z|x)}\left[\log \frac{1}{K}\sum_{k=1}^{K} \frac{p_\theta(x, z^{(k)})}{q_\phi(z^{(k)}|x)}\right] \tag{15}$$

where k are number of different samples drawn from Encoder output for a single image.

C. Generative Adversarial Networks (GANs)

The Mackay and Gibbs 1995 paper [11] on Density Networks laid the groundwork for understanding probabilistic models and density estimation, particularly focusing on techniques for representing and modeling complex probability distributions. This foundational work in density networks and probabilistic modeling contributed to the development of advanced generative modeling techniques, including the Variaitonal AutoEncoders (VAE) and Generative Adversarial Network (GAN) introduced by Goodfellow et al. in 2014 [5]. Although it was introduced in 1995, efficient ways to train them at scale didn't really come along until VAEs and GANs. The concepts of modeling distributions and generating realistic data samples learned from density networks were instrumental in inspiring the architecture and training principles of GANs, which involve a generator network learning to mimic a target distribution through adversarial training with a discriminator network.

The Goodfellow et al. 2014 paper [5] introduced Generative Adversarial Networks (GANs) comprising two main components: the generator (G) and the discriminator (D). The generator learns to generate realistic data samples (x) from random noise (z), while the discriminator acts as a binary classifier distinguishing real and fake samples. The GAN objective function is given as:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))] \tag{16}$$

Here, $p_{\text{data}}(x)$ is the real data distribution, $p_z(z)$ is the noise distribution, $G(z)$ is the generated sample, and $\mathbb{E}$ denotes expectation. The goal is to find a Nash equilibrium where the generator produces realistic samples and the discriminator cannot distinguishe accurately between real and fake samples.

Further, exploring advanced GAN concepts, I delved into Odena et al.'s 2016 [13] work, addressing checkerboard artifacts through Resize-Convolution instead of using normal Deconvolution. The gradient calculation through this deconvolution process created the issue of checkerboard artifacts in the generated image. Hence, the authors came up with the idea of Resize-convolution where you can resize the image using nearest-neighbor interpolation or bilinear interpolation and then use the convolution as an alternative to Deconvolution which discourages the high frequency checkerboard artifacts. This paper particularly enlightened me about how training a good generative model requires good math AND smart decisions about the neural network architecture. One without the other will be a bad model.

The goal is to make $p_z(x) \approx p_{data}(x)$. While training the GAN model, the main probelm we face is mode collapse when the models starts to generate same labelled images if trained for a longer time than required. Arjovsky et al.'s 2017 [1] paper was about tackling mode collapse by incorporating Wasserstein Distance into GANs as a way of measuring how these two distributions. Lipschitz continuity helps ensure the stability and convergence of the training process by controlling the gradients of the discriminator. This constraint on the discriminator's Lipschitz constant improves the quality and diversity of the generated samples. This approach shifted the discriminator's objective from binary classification of real

or fake image to measuring the distance between generated and true distributions, significantly improving stability during training and completely avoiding the issue of Mode collapse. Lipschitz functions as a measure to control the slope of the weights in -*c* to *c* so we dont have to deal with skewed distributions making the Discriminator a predictor for distance between the true distribution and generated distribution . The objective function as stated by the authors in the paper is,

$$\min_G \max_D \mathbb{E}_{x\sim p_{\text{data}}(x)}[D(x)] - \mathbb{E}_{z\sim p_z(z)}[D(G(z))] \tag{17}$$

where G represents the generator network. D represents the discriminator network, constrained to be from a set of 1-Lipschitz functions denoted as $\mathcal{D}$ (also called as "Critic"). $p_{\text{data}}(x)$ is the distribution of real data. $p_z(z)$ is the noise distribution used by the generator. $\mathbb{E}_{x\sim p_{\text{data}}(x)}[D(x)]$ represents the expected value of the discriminator's output for real data samples. $\mathbb{E}_{z\sim p_z(z)}[D(G(z))]$ represents the expected value of the discriminator's output for generated data samples.

Additionally, I explored Conditional GANs as outlined in Odena et al.'s 2017 paper [14], which enables generating images based on specified labels, and Denton et al.'s 2015 work [4], introducing GAN training based on Laplacian Pyramids, showcasing diverse extensions and innovations within the GAN framework.

D. Denoising Diffusion Probabilistic Models (DDPMs)

The forefront of Generative Models is advanced by Ho et al. in 2020 [7], presenting a groundbreaking method for image generation that exhibits exceptional promise in producing high-quality images. This pioneering work builds upon the foundational principles established by Sohl-Dickstein et al. [15] in 2015, in their exploration of Deep Unsupervised Learning using Nonequilibrium Thermodynamics. Their seminal work introduced the concept of Diffusion, leveraging the laws of thermodynamics, which has since been refined and extended by Ho et al. in 2020 [7]. The core idea involves diffusing noise into an image until it converges to a Normal distribution for time *T* steps, followed by a reverse process where a Neural network model predicts the noise at each time step, ultimately restoring the image to its original state using the Langevin dynamics. The resulting trained model exhibits exceptional capability in generating high-quality and diverse samples.

Author stated that during the Reverse diffusion process, rather than predicting the mean of the noisy distribution at each time step *t* using a Neural network, predicting the noise will help us to predict the state of the image at every *t*. The objective function proposed in this paper is the Variational Lower bound on log likelihood which is the KL Divergence between the added noisy distributions *q* and generated distribution *p* at time *t*. p and q are given as

$$q(x_{[1:T]}|x_0) = \mathcal{N}(x_t; \sqrt{\tilde{\alpha}_t}x_0, (1-\tilde{\alpha})I) \tag{18}$$

$$p(x_{[0:T]}) = \mathcal{N}(x_t; \frac{1}{\sqrt{\alpha}_t}(x_t - \frac{1-\alpha_t}{\sqrt{1-\tilde{\alpha}_t}}), \frac{1-\tilde{\alpha}_{t-1}}{1-\tilde{\alpha}_t}\beta_t) \tag{19}$$

where time $t = 0$, x_0 is the original single image and at time step t it is the added gaussian noise image x_t.

This Variational Lower bound is represented as,

$$\mathcal{L}_{VLB} = \mathbb{E}_{x\sim[1:T]\sim q(x_{[1:T]}|x_0)}\left[\log \frac{q(x_{[1:T]}|x_0)}{p_\theta(x_{[0:T]})}\right] \tag{20}$$

where,

The authors derived a new Objective function using Markov chain properties from some algebraic re-arranging of terms to be,

$$\mathcal{L}_{VLB} = L_T + \sum_{t=2}^{T-2} L_t + L_0 \tag{21}$$

where,

- L_T = $\text{KL}[q(x_T|x_0)||p_\theta(x_T)]$
- L_t = $\text{KL}[q(x_t|x_{t+1}, x_0)||p_\theta(x_t|x_{t+1})]$
- L_0 = $-\log p_\theta(x_0|x_1)$

The authors then parameterized the L_t in order to simplify noise prediction with respect to noise predicted using L2 loss as,

$$L_t^{\text{simple}} = \mathbb{E}_{t\sim[1,T],x_0,\epsilon}[||\epsilon_t - \epsilon_\theta(\sqrt{\tilde{\alpha}_t}x_0 + \sqrt{1-\tilde{\alpha}}\epsilon_t, t)||^2] \tag{22}$$

In a related domain, Kadkhodaie and Simoncelli, 2021 [9] paper, present a simple proof that denoising using a squared reconstruction loss as shown in equation (22) is equivalent to score estimation. This strategy effectively regularizes and enhances the quality of solutions for linear inverse problems by incorporating the denoiser's deep understanding of signal structures and noise characteristics. Their work showcases significant advancements in tasks such as image reconstruction and super-resolution, underscoring the potential of leveraging deep learning priors to tackle challenging inverse problem scenarios.

Using Bayes theorem on score matching objective, it is very easy to implement a Conditional Diffusion Model using a normal Classifier such as VGG or Inception networks. Although in 2022, Ho and Salimans [8] introduced an innovative approach to construct a Conditional Diffusion model without reliance on such a Classifier. Instead, they pursued training both conditional and unconditional Diffusion models concurrently, leading to the development of a Classifier-Free Guidance Diffusion Model. This was achieved by modifying the objective function of noise prediction, incorporating a guidance strength parameter to guide the diffusion process effectively. This novel methodology not only avoids the need for a separate Classifier but also enhances the model's overall performance in generating high-quality and diverse samples.

The revised predicted noise function for this model is given as,

$$\tilde{\epsilon_\theta}(z_\lambda, c) = (1 + w)\epsilon_\theta(z_\lambda, c) - w\epsilon_\theta(z_\lambda) \tag{23}$$

where w is guidance strength and c is class label. This noise term is then used in the equation (22) as derived for unconditional DDPM. The first term contributes more to the objective function when the class label is correct making the model better in conditional sense and the second term contributes for unconditional model training. In conclusion, this model is being trained for both conditional and unconditional Diffusion model together and meeting the objective of circumventing the use of a different Classifier.

III. Experiment

A. Variational Autoencoders

1) Variational Autoencoder (VAE): The Encoder module processes input x to produce the latent space representation z, while the Decoder module reconstructs x using the sampled z obtained through the reparameterization trick. The Encoder, parameterized by ϕ, calculates the posterior distribution $q_\phi(x|z)$, and the Decoder, parameterized by θ, computes the conditional distribution $p_\theta(z|x)$. These components are integral to the Variational Autoencoder (VAE) framework, where the objective function ELBO (as defined in equation (14)) plays a central role.

- *Architecture:* The Encoder architecture utilizes standard Fully Connected layers with input dimensions equal to the input image height h multiplied by the width w, incorporating 3 hidden layers. This configuration yields an output comprising the mean μ and the logarithm of variance $\log(\sigma)$ for a Latent dimension of 20. On the other hand, the Decoder accepts the latent dimension 20 as its input layer dimension. Similar to the Encoder, the Decoder also comprises 3 hidden layers, with its output layer generating the reconstructed output x, maintaining the same dimensions as the input to the Encoder. During the VAE implementation, my professor recommended incorporating an additional learning parameter into the Neural network as the diagonal covariance matrix in pixel space.

- *Results:* The model was trained for 10 Epochs, 0.0003 learning rate and a batch size of 32 on the MNIST dataset. Fig. (1) shows the randomly generated samples for MNIST dataset using a trained VAE model.

2) Importance Weighted Autencoder (IWAE): Importance Weighted Autoencoder (IWAE) extends the Variational Autoencoder (VAE) framework by incorporating importance sampling. Similar to VAE, IWAE utilizes the Encoder and Decoder modules parameterized by ϕ and θ for computing the posterior $q_\phi(z|x)$ and the generative distribution $p_\theta(z|x)$, respectively. The key distinction lies in the objective function, where IWAE introduces importance sampling, as illustrated in equation (15).

Fig. 1. VAE output

- *Architecture*: I used the same Encoder and Decoder for this implementation.
- *Results*: The model was trained for 10 Epochs, 0.0003 learning rate and a batch size of 32 on the MNIST dataset. Fig. (2) shows the randomly generated samples for MNIST dataset given by a trained IWAE.

Fig. 2. IWAE output

As we can see, the IWAE gave more confident outputs in generating MNIST digits for the same hyperparameters. IWAE is superior to VAE because it tightens the Evidence Lower Bound (ELBO), ensuring a more precise approximation of the true data distribution and leading to improved model quality and representation learning.

B. Generative Adversarial Networks

1) Generative Adversarial Network (GAN): Generative Adversarial Networks (GANs) are composed of two components: the Generator and the Discriminator. The goal is to minimize the loss of the Generator while maximizing that of the Discriminator. The objective function employed during implementation is derived from equation (16). The key strategy in GAN training involves updating the Discriminator for k=5 steps followed by training the Generator for 1 step. This alternating training procedure accelerates Discriminator learning, leading to improved Generator performance overall.

- *Architecture:* Generator is a simple MLP with input layer equal to the noise dimensions, one hidden layer and output layer equal to the image dimension as it should be capable to generate an image. Discriminator is also a simple MLP which takes in the input of input dataset image or generated image. It has one hidden layer and it outputs if the given input image to Discriminator is from True distribution or generated distribution with one neuron in output layer.
- *Result:* The GAN was trained for 30 Epochs with learning rate 0.0003 as given in the paper, and Batch size of 128. Fig. (3) shows the output of the GAN.

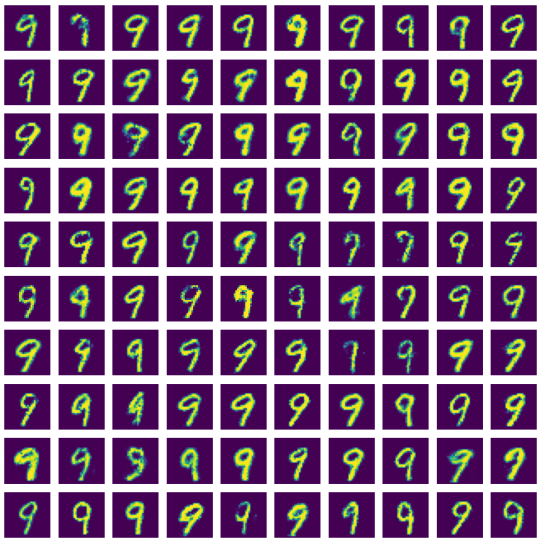


Fig. 3. Generated samples from GAN

As you can see, even if the model was trained for 30 epochs, it gave me an output of only a single label ”9” which shows the issue of mode collapse in training the GAN model. Hence, Wasserstein GAN solves this problem of model collapse.

Just to check the problem of Mode collapse i trained the same model for 15 epochs to check the difference between the outputs. Fig. (4) shows the output for the same which generates different labels for ”7” and ”9”. Eventually, after 30 epochs the model converges into Mode collapse as shown in Fig. (3).

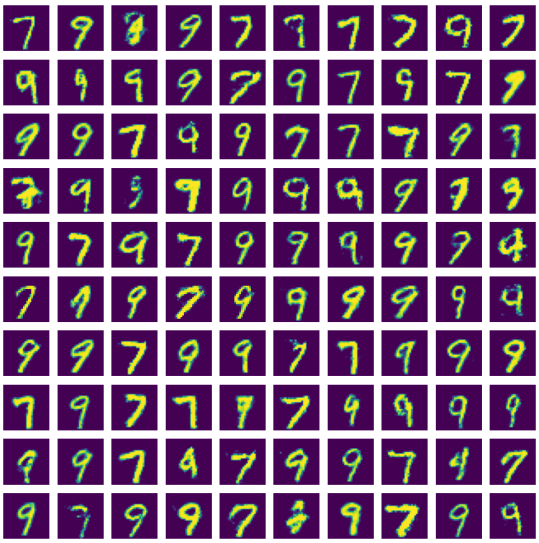


Fig. 4. GAN before Mode collapse

2) Wasserstein Generative Adversarial Network: Wasserstein Generative Adversarial Networks (WGANs) employ the Wasserstein distance in their objective function, as indicated by equation (17). To enforce the Lipschitz constraint, WGANs utilize a clipping method to control the Discriminator’s weights, typically setting the clipping constant to -0.001 and 0.001. Similar to Vanilla GANs, the training regimen involves updating the Discriminator for 5 steps for every 1 step of Generator training.

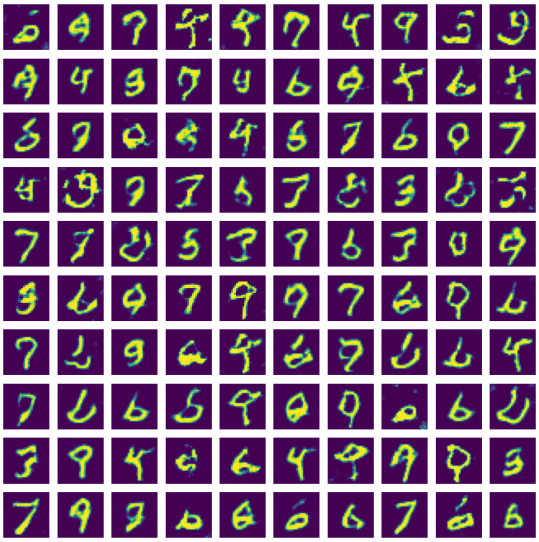


Fig. 5. Generated samples from WGAN

- *Architecture:* I have utilized Convolutional layers in both the Generator and Discriminator architectures to enhance feature extraction and deepen my understanding of Convolutional layer usage in PyTorch. The Generator comprises 2 Convolutional layers with a stride of 1, Max-

Pool layers with a stride of 2, and Batch Normalization for efficient processing. Following this, the Generator includes a fully-connected layer with one hidden layer and an output layer with dimensions matching the image size. Similarly, for the Discriminator, I've employed 2 Convolutional layers with a stride of 1, MaxPool layers with a stride of 2, and Batch Normalization. The output from these layers is then flattened and passed through a fully connected layer, consisting of one hidden layer and an output layer with a single neuron, as it predicts the Wasserstein distance.

- *Results:* The WGAN model was trained for 30 epochs, following the same duration as the GAN implementation. The learning rate was set to 0.0005, as specified in the referenced paper, and a batch size of 128 was used for training. Fig. 5 displays the results of random samples generated from a trained WGAN model.
  The results are conclusive in demonstrating that WGAN effectively mitigates the issue of mode collapse, enabling the generation of distinct labeled images from the MNIST dataset. As outlined in the referenced paper, with extended training, the "critic" becomes proficient at producing a wide range of samples, thus entirely circumventing mode collapse.

C. Denoising Diffusion Probabilistic Model (DDPM)

In a DDPM the first process involves adding noise to the image up to timestep t and then sampling from the conditional distribution $q(x_{t-1}|x_t, x_0)$ after this noise addition. The second process entails moving backward in time from $t = T$ to $t = 1$, utilizing a neural network to forecast $p(x_{t-1}|x_t)$. This implementation concept of DDPM has proven to be successful in generating high-quality and diverse samples by predicting noise at each time step, thereby enhancing the overall output quality.

- *Architecture:* The architecture of DDPM is indeed more intricate compared to VAEs or GANs. It incorporates UNets and a Multi-head Self-Attention mechanism to handle time embeddings. These embeddings are concatenated with the output from Convolutional layers within the ResNet block. The UNet structure comprises three main modules: Downblock, Midblock, and Upblock. Downblock and Midblock share similar architecture, involving ResNet blocks and Self-Attention layers. The Downblock further includes downsampling and passes its output to the Midblock, which contains ResNet blocks, a Self-Attention layer, and another ResNet block. The resulting output from Midblock is then forwarded to Upblock for upsampling, followed by ResNet and Self-Attention blocks. Ultimately, this process generates noise predictions at time t, parameterized by θ. The objective function of DDPM is determined by Mean Squared Error (MSE), calculated between the noise at time t and the model-predicted noise at time t, as referenced from equation (22).
- *Result:* The implementation involved using 1000 time steps (T) to add noise gradually until reaching the Normal distribution state. Then, starting from T, the neural network generated the final image at time 0. The parameter β, representing the mean of noise at $t = 1$, varied from 0.0001 to 0.002 over time T. The program ran for 15 epochs with a batch size of 128 and a learning rate of 0.0001, as specified in the paper by Ho et al. [2020]. In Fig. 6, the images at $t = 999$ are filled with noise, approaching a Normal distribution. As time progresses from 999 to 0, the model demonstrates the ability to generate diverse samples. Fig. 7 showcases an image at $t = 350$, providing strong evidence of effective denoising by the model. Finally, Fig. 8 illustrates that the DDPM model successfully denoised the image, producing high-quality and diverse samples.

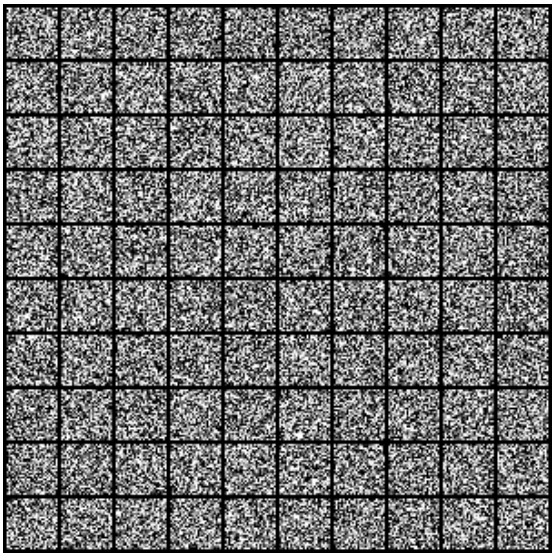


Fig. 6. DDPM generated image at t = 999

I also plot the graph for noise prediction over the number of epochs as this is a normal optimization problem and it becomse important to make the model predict noise with least error possible, hence the loss graph for noise prediction looks like it decreases over number of epochs. Fig. 9 shows the loss graph.

IV. Summary & Future interests

This course provided me with a deep dive into the mathematical foundations essential for formulating objective functions in Generative Models like VAEs, GANs, and DDPMs. Through this study, I gained a profound appreciation for the pivotal role of Probability theory in shaping these model families, discerning the nuanced contrasts between VAEs, GANs, and DDPMs, including their respective strengths and limitations.

VAEs excel in capturing latent space distributions via an Encoder-Decoder framework, primarily optimizing the

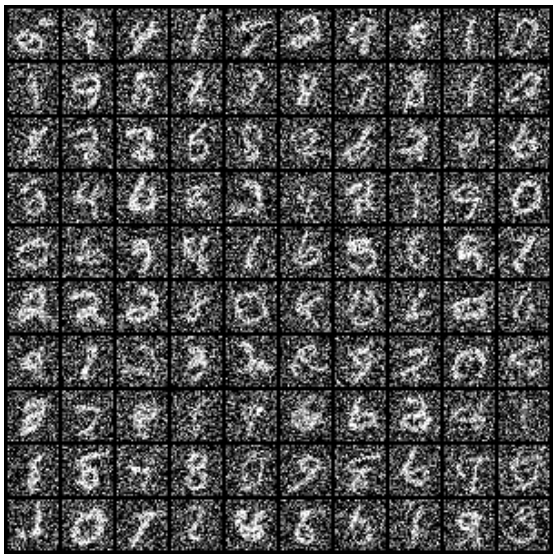


Fig. 7. DDPM generated image at $t = 350$

Fig. 8. DDPM generated image at $t = 0$

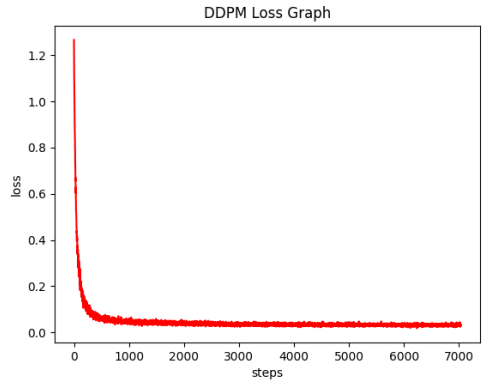


Fig. 9. DDPM Loss graph

Evidence Lower Bound (ELBO). While VAEs offer speed, they may lack accuracy, prompting the adoption of IWAE, which leverages importance sampling on both recognition and generative model distributions, leading to tighter bounds and enhanced image generation accuracy.

GANs is another subdomain of Generative models which uses 2 neural networks; Generator and Discriminator playing a game of min-max where the objective is minimize the loss of generator and maximize the loss of discriminator. The goal of a GAN is "Fool the discriminator". GAN converges into a problem of Mode collapse where the model if trained longer than needed, starts to give an output with same kind of image. To overcome this problem Wasserstein GAN is used which finds the distance between the True distrobution with respect to Generated distribution. GANs have shown promise to generate diverse samples compared to VAEs but comes with the problems of model training.

DDPMs represent a cutting-edge domain, leveraging noise injection and neural networks to predict noise evolution at each time step, drawing inspiration from Langevin sampling and Score matching techniques. This approach has demonstrated superior results in generating high-quality, diverse samples compared to VAEs and GANs.

My hands-on experience implementing these models in PyTorch enriched my understanding of their underlying mathematics, fostering a keen interest in advancing the field of Generative Models. Exploring the intricacies of these models has equipped me to identify and tackle low-level challenges, igniting my passion for future research and problem-solving in this domain.

Acknowledgment

I am immensely grateful to my advisor, Dr. Richard Lange, for his invaluable guidance and support throughout my academic journey. His dedication and expertise have provided me with a profound understanding of essential concepts and mathematical principles in our field. Dr. Lange's mentorship has not only inspired me but also guided me towards pursuing a path in academia. The knowledge and skills I have acquired under his tutelage have played a pivotal role in my decision to undertake a thesis for my Master's program.

References

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223. PMLR, 06–11 Aug 2017.

[2] Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders, 2016.

[3] Peter Dayan. Recognition in hierarchical models. In Felipe Cucker and Michael Shub, editors, *Foundations of Computational Mathematics*, pages 43–62, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

[4] Emily L Denton, Soumith Chintala, arthur szlam, and Rob Fergus. Deep generative image models using a laplacian pyramid of adversarial networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[5] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[6] Irina Higgins, Loic Matthey, Arka Pal, Christopher P Burgess, Xavier Glorot, Matthew M Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. *ICLR (Poster)*, 3, 2017.

[7] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020.

[8] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance, 2022.

[9] Zahra Kadkhodaie and Eero P. Simoncelli. Solving linear inverse problems using the prior implicit in a denoiser, 2021.

[10] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022.

[11] David J.C. Mackay and Mark N. Gibbs. *Density networks*, page 129–145. Oxford University Press, Inc., USA, 2000.

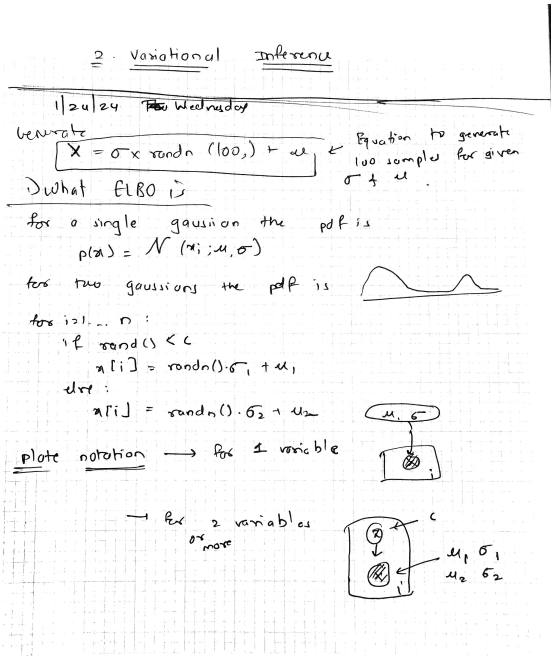
[12] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

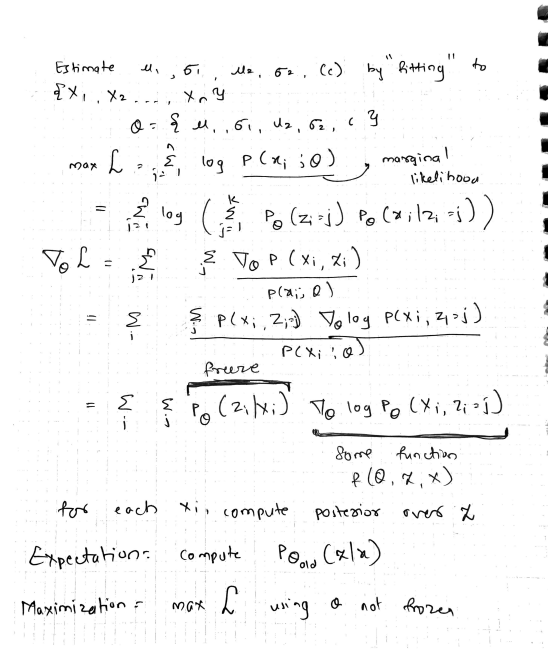
[13] Augustus Odena, Vincent Dumoulin, and Chris Olah. Deconvolution and checkerboard artifacts. *Distill*, 2016.

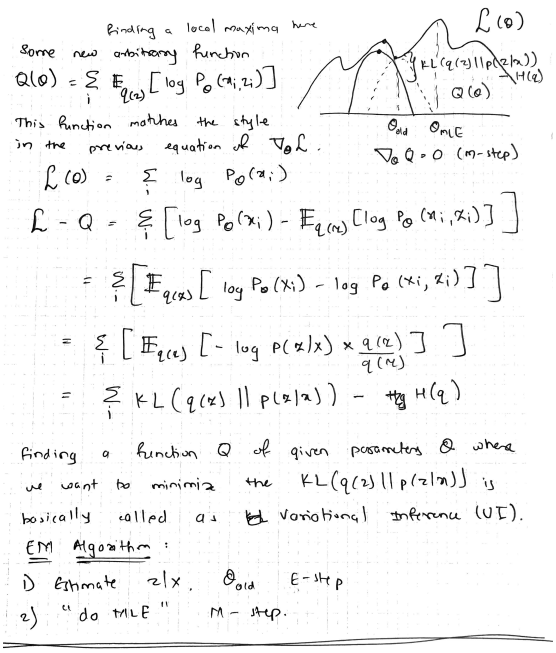
[14] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier GANs. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2642–2651. PMLR, 06–11 Aug 2017.

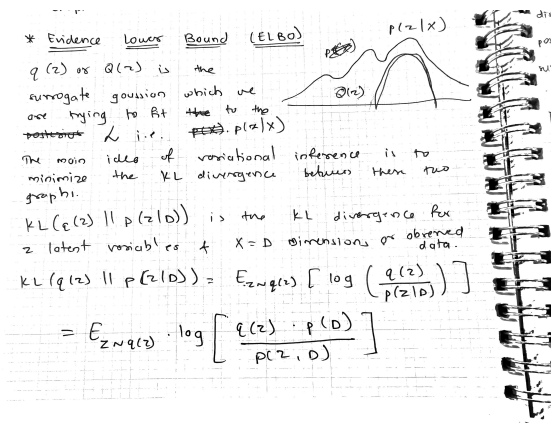
[15] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2256–2265, Lille, France, 07–09 Jul 2015. PMLR.

V. Appendix









* Evidence Lower Bound (ELBO)

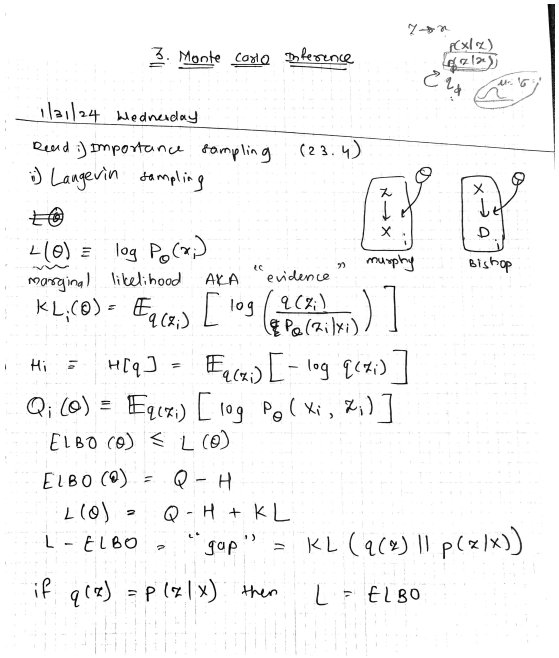
$q(z)$ or $Q(z)$ is the surrogate gaussian which we are trying to fit to the ~~posterior~~ $p(z|x)$. i.e. $p(z|x)$

The main idea of variational inference is to minimize the KL divergence between these two graphs.

$KL(q(z) \,||\, p(z|D))$ is the KL divergence for z latent variables & $X = D$ dimensions or observed data.

$$KL(q(z) \,||\, p(z|D)) = E_{z \sim q(z)} \left[\log \left(\frac{q(z)}{p(z|D)} \right) \right]$$

$$= E_{z \sim q(z)} \cdot \log \left[\frac{q(z) \cdot p(D)}{p(z, D)} \right]$$



3. Monte Carlo Inference

1/31/24 Wednesday

Read i) Importance sampling (23.4)

ii) Langevin sampling

$L(\theta) \equiv \log P_\theta(x_i)$

marginal likelihood AKA "evidence"

$$KL_i(\theta) = E_{q(z_i)} \left[\log \left(\frac{q(z_i)}{P_\theta(z_i|x_i)} \right) \right]$$

$H_i = H[q] = E_{q(z_i)} [-\log q(z_i)]$

$Q_i(\theta) \equiv E_{q(z_i)} [\log P_\theta(x_i, z_i)]$

$ELBO(\theta) \leq L(\theta)$

$ELBO(\theta) = Q - H$

$L(\theta) = Q - H + KL$

$L - ELBO$ = "gap" $= KL(q(z) \,||\, p(z|x))$

if $q(z) = p(z|x)$ then $L = ELBO$

murphy

Bishop

$$= E_{z \sim q(z)} \left[\log \left(\frac{q(z)}{p(z|D)} \right) \right] + E_{z \sim q(z)} [\log p(D)]$$

$$= -E_{z \sim q(z)} \left[\log \left(\frac{p(z,D)}{q(z)} \right) \right] + \log p(D)$$

Hence,

KL = $-ELBO(q)$ + $\log p(D)$

- KL: distance between posterior & surrogate
- $-ELBO(q)$: → negative must ≤ 0 → smaller than $\log p(D)$ → Evidence Lower Bound.
- $\log p(D)$: marginal → evidence → ≤ 0 → a constant quantity

$$\boxed{ELBO(q) = E_{z \sim q(z)} \left[\log \left(\frac{p(z,D)}{q(z)} \right) \right]}$$

variational inference is

$$q^*(z) = \underset{q(z) \in Q}{\operatorname{argmin}} \left(KL(q(z) \,||\, p(z|D)) \right)$$

for Q being family of different but simple Gaussians.

Given: $P_\theta(x)$ family of distributions $\{x_1, \ldots, x_n\}$

Goal: Find θ^* MLE / Best fit

Catch: really have $P_\theta(x)$ $P_\theta(x|z)$ "latent variable model"

Algorithm (EM):

init θ_0 {random}:

$L = Q + KL - H$ ($KL \to 0$, H not θ)

for $i = 1 \ldots$ steps:

define $q_t(z) = P_{\theta_t}(z|x)$

compute $\frac{\partial Q}{\partial \theta_t}$, set to 0

$$Q(\theta_t) = E_{q(z)} [\log P_{\theta_t}(x, z)]$$

$\theta_t \leftarrow \operatorname{argmax} Q$

Algorithm (VAE, SGVB):

init θ_0

$ELBO = L - KL$

$= Q + H$

for $i = 1 \ldots$ steps:

$\theta_t \leftarrow \theta_{t-1} + \eta \nabla ELBO$

Assume, $q(x) \approx q(z|x)$

$ELBO = Q + H = L - KL(q(z) \,||\, p(z|x))$

$= \mathbb{E}_{q(z|x)}\left[\log \frac{p(x,z)}{q(z|x)}\right]$

$= \mathbb{E}_{q(z|x)}\left[\log \frac{p(z)}{q(z|x)}\right] + \mathbb{E}_{q(z|x)}\left[\log p(x_i|z)\right]$

$= -KL(q(z|x) \,||\, p(z)) +$ "auto-encoder reconstruction loss"

(regularization)

want $q(z|x) = p(z|x)$

$x \rightarrow [\quad] \rightarrow q(z|x)$

"guess z given x"

"recognition model" → Dayan 1997

KL divergence for two univariate gaussians

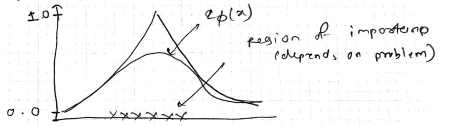
~~$KL(p||q) = \log$~~

$$KL(q\,||\,p) = \log \frac{\sigma_p}{\sigma_q} + \frac{\sigma_q^2 + (\mu_q - \mu_p)^2}{2\sigma_p^2} - \frac{1}{2}$$

$p(x) = \mathcal{N}(\mu_p, \sigma_p) \quad q(x) = \mathcal{N}(\mu_q, \sigma_q)$

$\mu = \underset{x}{\operatorname{argmax}}\, h(x)\,p(x)$

Using this mean μ, use q of a Normal distribution μ i.e. $\mathcal{N}(\mu, 1)$.



* What if P_θ is not normalized?

$Z_p = \int_R P_\theta(x)\,dx \quad ; \quad Z_q =$

$q_\phi(x) = \frac{\tilde{q}_\phi(x)}{Z_q} \quad ; \quad Z_q = \frac{\tilde{q}_\phi(x)}{q_\phi(x)}$

Now,

$$\frac{Z_p}{Z_q} = \frac{1}{Z_q}\int_R \tilde{P}_\theta(x)\,dx$$

$$= \frac{q_\phi(x)}{\tilde{q}_\phi(x)}\int_R \tilde{P}_\theta(x)\,dx$$

$$= \int_R \tilde{P}_\theta(x)\,\frac{q_\phi(x)}{\tilde{q}_\phi(x)}\,dx$$

* Importance Sampling

Used to calculate the Expected value of $h(x)$. (Probability distribution of x.)

$$\mathbb{E}_{P_\theta}[h(x)] = \sum_{i=1}^{\infty} h(x_i)\,P_\theta(x_i)$$

($h(x_i)$: Random variable Function of x.)

$$\mathbb{E}_{P_\theta}[h(x)] = \int_R h(x)\,P_\theta(x)\,dx$$

Problems → 1) can't sample from P_θ

2) Inefficient to sample from P_θ

Variance reduction algorithm.

3) P_θ is not

Use a sampling / proposal distribution $q_\phi(x)$ (subjective)

$$\mathbb{E}_{P_\theta}[h(x)] = \int_R h(x)\,P_\theta(x)\,\frac{q_\phi(x)}{q_\phi(x)}\,dx$$

$$\mathbb{E}_{q_\phi}[h(x)] = \int_R h(x)\,\frac{P_\theta(x)}{q_\phi(x)}\,q_\phi(x)\,dx$$

Using law of large numbers (importance weight / likelihood ratio)

$$\mathbb{E}_{q_\phi}[h(x)] \approx \frac{1}{N}\sum_{i=1}^{N} h(x_i)\,\frac{P_\theta(x_i)}{q_\phi(x_i)}$$

* But how do we choose q_ϕ?

→ We would choose $q(x)$ that is similar to $h(x) * p(x)$

$$= \int_R \frac{\tilde{P}_\theta(x)}{\tilde{q}_\phi(x)}\,q_\phi(x)\,dx$$

Using law of large Numbers

$$= \mathbb{E}_{q_\phi}\left[\frac{\tilde{P}_\theta(x)}{\tilde{q}_\phi(x)}\right]$$

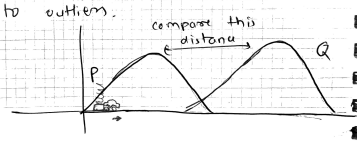
Hence, $\frac{Z_p}{Z_q} \approx \frac{1}{N}\sum_{i=1}^{N} \frac{\tilde{P}_\theta(x)}{\tilde{q}_\phi(x)}$

This is called as Self Normalized Importance Sampling

* Wasserstein distance (Earth movers distance)

Data type- continuous ; Range- $(0, +\infty)$

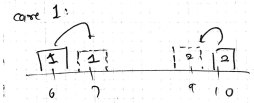
- It is the amount of work needed to transform one distribution into the other.
- Area between the PDFs.
- It is sensitive to outliers.



$W(P_g, P_g) = \inf_{\gamma \in \Pi(P_g, P_g)} \mathbb{E}_{(x,y)\sim\gamma} [\|x - y\|]$ (infimum)

$\Pi(P_g, P_g)$ → transport plan

case 1:

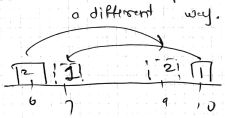


	7	9
6	1	0
10	0	1

Π_1 transport plan 1

$1 + 1 = 2$

case 2: Both end up at same final places but in a different way.



	7	9
6	0	1
10	1	0

Π_2 transport plan 2

$3 + 3 = 6$

Similarly many such transport plans can be calculated but only the one with minimum amount of work is selected i.e. $\pi(2) \rightarrow \Pi_1$. (infimum)

Finally, we approximate the supremum over all the functions with a parameterized family of function D_w.

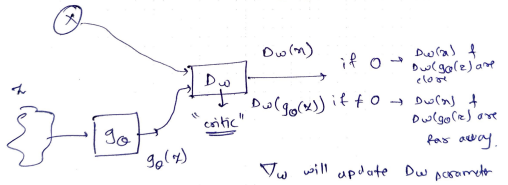
$\min_{\theta} \max_{w \in W} \mathbb{E}[D_w(x) - D_w(g_\theta(z))]$ s.t $\|D_w\|_L \leq$

$= \min_{\theta} \max_{w \in W} \mathbb{E}_{x \sim P_x}[D_w(x)] - \mathbb{E}_{z \sim P_z} D_w(g_\theta(z))]$

s.t. $\|D_w\|_L \leq k$.

This becomes my loss function for WGAN.

$\|D_w\|_L \leq k$ $\{-c, c\}$ clipping of weights



∇_w will update D_w parameters

∇_θ will update g_θ parameters

WGAN

* Wasserstein GAN

Arjovsky et. al. (2017)

$W(P_r, P_g) = \inf_{\gamma \in \Pi(P_r, P_g)} \mathbb{E}_{(x,y)\sim\gamma} [\|x - y\|]$

This is intractable.

Hence, using Kantorovich – Rubinstein duality this equation is equivalent to

$W(P_r, P_\theta) = \sup_{\|D\|_L \leq 1} \mathbb{E}[D(x) - D(g_\theta(z))]$

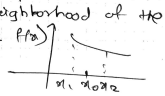
supremum (maximum) is taken over all 1– Lipschitz functions.

WGAN objective → $\min_{\theta} \{W(P_r, P_\theta)\}$

what is $\sup_{\|D\|_L \leq 1}$?

→ It is telling you that the slope of the curve should not exceed 1 in the neighborhood of the point of interest.

$\frac{f(x_1) - f(x_2)}{x_1 - x_2} \leq 1$ ← Lipschitz $f(x)$ constant



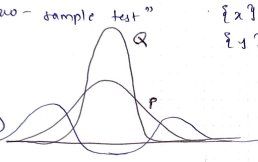
3/25/24 Monday

Lipschitz constraint

"Two-sample test"

$\{x_1 \ldots x_n\} \sim P \quad x \in \mathbb{R}^d$

$\{y_1 \ldots y_m\} \sim Q \quad y \in \mathbb{R}^d$



for the given P & Q distributions, we have $f(x)$ test function to check how far these distributions are.

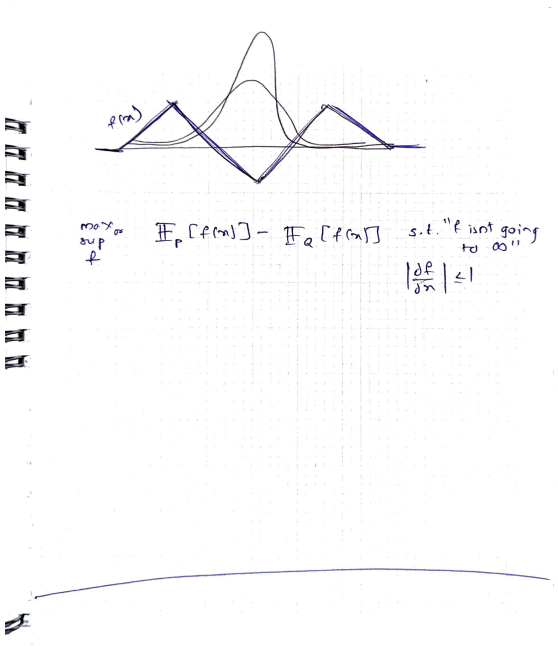
$f_1(x) = x \left| \frac{1}{n}\sum_{i=1}^{n} x_i - \frac{1}{m}\sum_{j=1}^{m} y_j \right|$ if same μ, $P = Q$

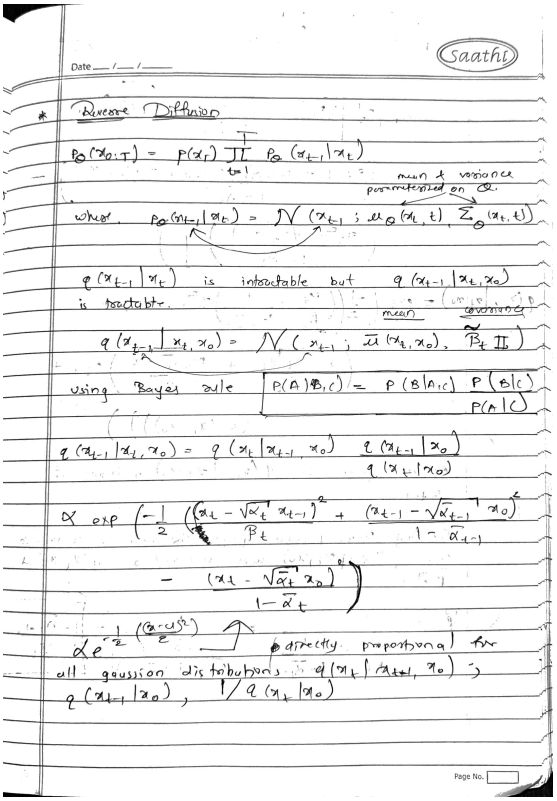
$f_2(x) = (x-\mu)^2 \left| \frac{1}{n}\sum_{i=1}^{n} (x_i - \mu_x)^2 - \frac{1}{m}\sum_{j=1}^{m} (y_j - \mu_\theta)^2 \right|$ if same σ, $P = Q$

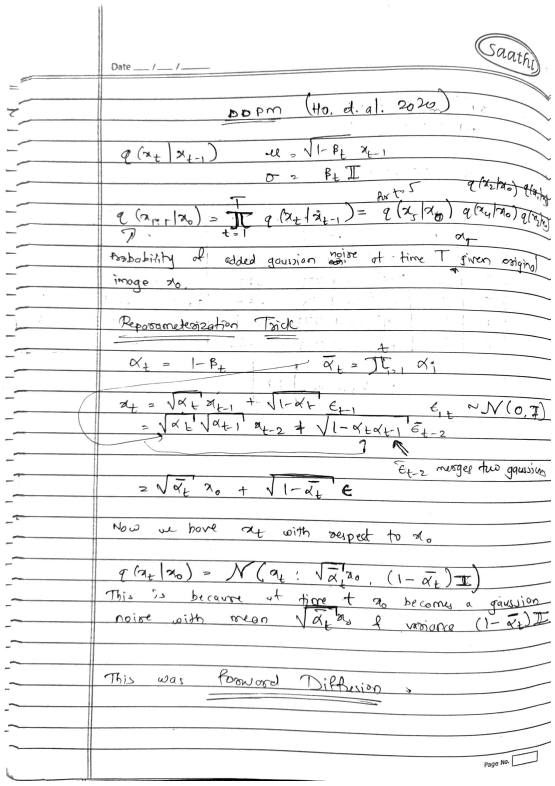
$f_3(x) = (x-\mu)^3$ | ...

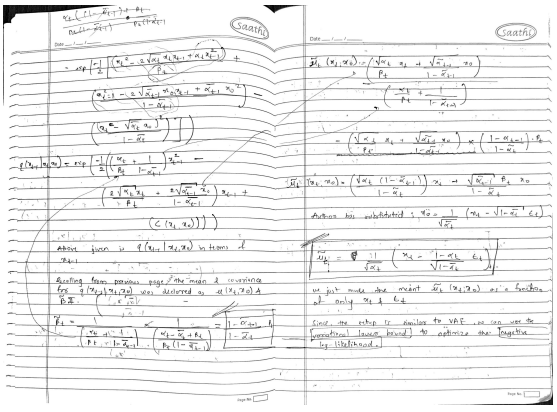
if $\forall f \; \mathbb{E}_P[f(x)] = \mathbb{E}_Q[f(y)]$ then $P = Q$

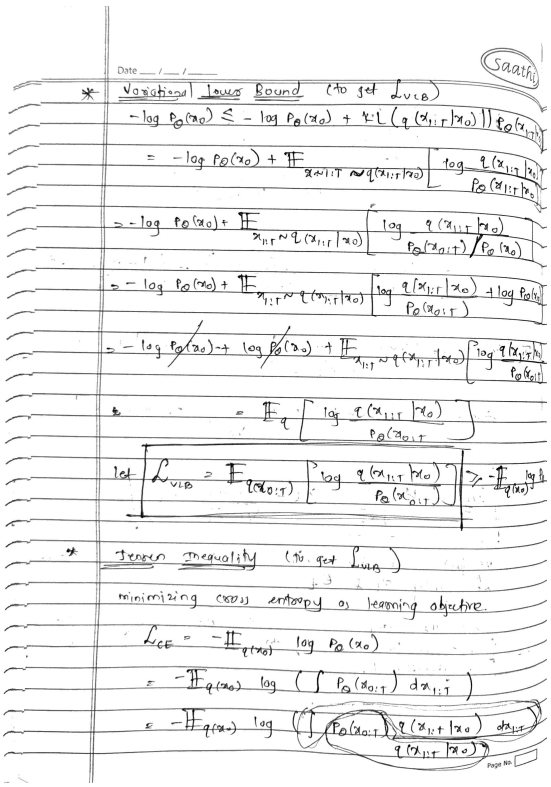
if $\exists f \; \mathbb{E}_P[f] \neq \mathbb{E}_Q[f]$ then $P \neq Q$


f(x)
max or sup f
$\mathbb{E}_p[f(x)] - \mathbb{E}_q[f(x)]$ s.t. "f isn't going to ∞"
$\left|\frac{df}{dx}\right| \leq 1$


Saathi
Reverse Diffusion
$p_\theta(x_{0:T}) = p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)$
mean & variance parameterized on θ.
where $p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t,t), \Sigma_\theta(x_t,t))$
$q(x_{t-1}|x_t)$ is intractable but $q(x_{t-1}|x_t,x_0)$ is tractable.
$q(x_{t-1}|x_t,x_0) = \mathcal{N}(x_{t-1}; \tilde{\mu}(x_t,x_0), \tilde{\beta}_t \mathbf{I})$
mean, variance
using Bayes rule $P(A|B,C) = \frac{P(B|A,C)\,P(B|C)}{P(A|C)}$
$q(x_{t-1}|x_t,x_0) = \frac{q(x_t|x_{t-1},x_0)\,q(x_{t-1}|x_0)}{q(x_t|x_0)}$
$\propto \exp\left(-\frac{1}{2}\left(\frac{(x_t - \sqrt{\alpha_t}x_{t-1})^2}{\beta_t} + \frac{(x_{t-1} - \sqrt{\bar{\alpha}_{t-1}}x_0)^2}{1-\bar{\alpha}_{t-1}} - \frac{(x_t - \sqrt{\bar{\alpha}_t}x_0)^2}{1-\bar{\alpha}_t}\right)\right)$
$\propto e^{-\frac{1}{2}\left(\frac{(x-\mu)^2}{\sigma^2}\right)}$ directly proportional for
all gaussian distributions, $q(x_t|x_{t+1},x_0)$ / $q(x_{t-1}|x_0)$ / $q(x_t|x_0)$
Page No.


Saathi
DDPM (Ho, et. al. 2020)
$q(x_t|x_{t-1})$ $\mu = \sqrt{1-\beta_t}\,x_{t-1}$
$\sigma = \beta_t \mathbf{I}$
$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1}) = q(x_1|x_0)\,q(x_2|x_1)\,q(x_3|x_2)\ldots$ Art to S
α_t
Probability of added gaussian noise at time T given original image x_0.
Reparameterization Trick
$\alpha_t = 1-\beta_t$ $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$
$x_t = \sqrt{\alpha_t}\,x_{t-1} + \sqrt{1-\alpha_t}\,\epsilon_{t-1}$ $\epsilon_t \sim \mathcal{N}(0,\mathbf{I})$
$= \sqrt{\alpha_t}\sqrt{\alpha_{t-1}}\,x_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\,\bar{\epsilon}_{t-2}$
$\bar{\epsilon}_{t-2}$ merges two gaussian
$= \sqrt{\bar{\alpha}_t}\,x_0 + \sqrt{1-\bar{\alpha}_t}\,\epsilon$
Now we have x_t with respect to x_0
$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)\mathbf{I})$
This is because at time t x_0 becomes a gaussian noise with mean $\sqrt{\bar{\alpha}_t}x_0$ & variance $(1-\bar{\alpha}_t)\mathbf{I}$
This was Forward Diffusion
Page No.


Saathi
Saathi



Variational lower Bound (to get $\mathcal{L}_{VLB}$)

$$-\log p_\theta(x_0) \le -\log p_\theta(x_0) + KL\left(q(x_{1:T}|x_0) \,||\, p_\theta(x_{1:T}|x_0)\right)$$

$$= -\log p_\theta(x_0) + \mathbb{E}_{x_{1:T}\sim q(x_{1:T}|x_0)}\left[\log \frac{q(x_{1:T}|x_0)}{p_\theta(x_{1:T}|x_0)}\right]$$

$$= -\log p_\theta(x_0) + \mathbb{E}_{x_{1:T}\sim q(x_{1:T}|x_0)}\left[\log \frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})/p_\theta(x_0)}\right]$$

$$= -\log p_\theta(x_0) + \mathbb{E}_{x_{1:T}\sim q(x_{1:T}|x_0)}\left[\log \frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})} + \log p_\theta(x_0)\right]$$

$$= -\log p_\theta(x_0) + \log p_\theta(x_0) + \mathbb{E}_{x_{1:T}\sim q(x_{1:T}|x_0)}\left[\log \frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})}\right]$$

$$= \mathbb{E}_q\left[\log \frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})}\right]$$

let $\mathcal{L}_{VLB} = \mathbb{E}_{q(x_{0:T})}\left[\log \frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})}\right] \ge -\mathbb{E}_{q(x_0)}\log p_\theta(x_0)$

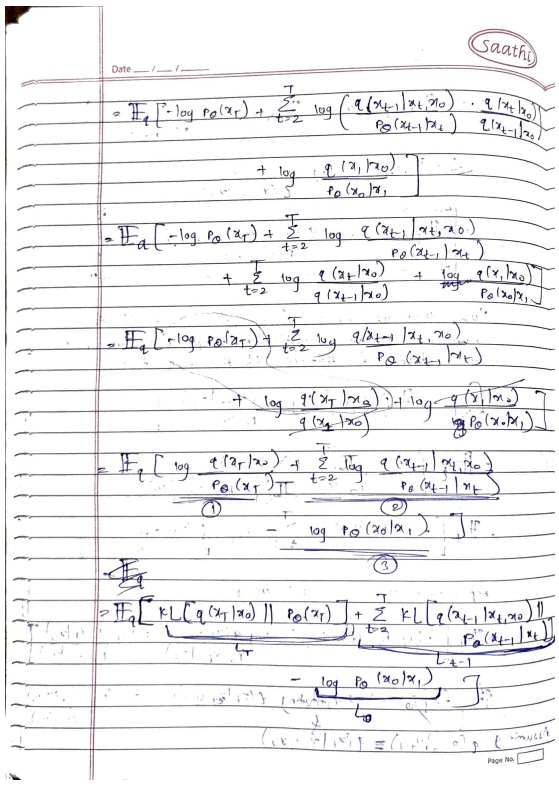
Jensen inequality (to get $\mathcal{L}_{VLB}$)

minimizing cross entropy as learning objective.

$$\mathcal{L}_{CE} = -\mathbb{E}_{q(x_0)}\log p_\theta(x_0)$$

$$= -\mathbb{E}_{q(x_0)}\log\left(\int p_\theta(x_{0:T})\, dx_{1:T}\right)$$

$$= -\mathbb{E}_{q(x_0)}\log\left(\int \frac{p_\theta(x_{0:T})\, q(x_{1:T}|x_0)}{q(x_{1:T}|x_0)}\, dx_{1:T}\right)$$



$$= \mathbb{E}_q\left[-\log p_\theta(x_T) + \sum_{t=2}^{T}\log\frac{q(x_{t-1}|x_t,x_0)}{p_\theta(x_{t-1}|x_t)}\cdot\frac{q(x_t|x_0)}{q(x_{t-1}|x_0)} + \log\frac{q(x_1|x_0)}{p_\theta(x_0|x_1)}\right]$$

$$= \mathbb{E}_q\left[-\log p_\theta(x_T) + \sum_{t=2}^{T}\log\frac{q(x_{t-1}|x_t,x_0)}{p_\theta(x_{t-1}|x_t)} + \sum_{t=2}^{T}\log\frac{q(x_t|x_0)}{q(x_{t-1}|x_0)} + \log\frac{q(x_1|x_0)}{p_\theta(x_0|x_1)}\right]$$

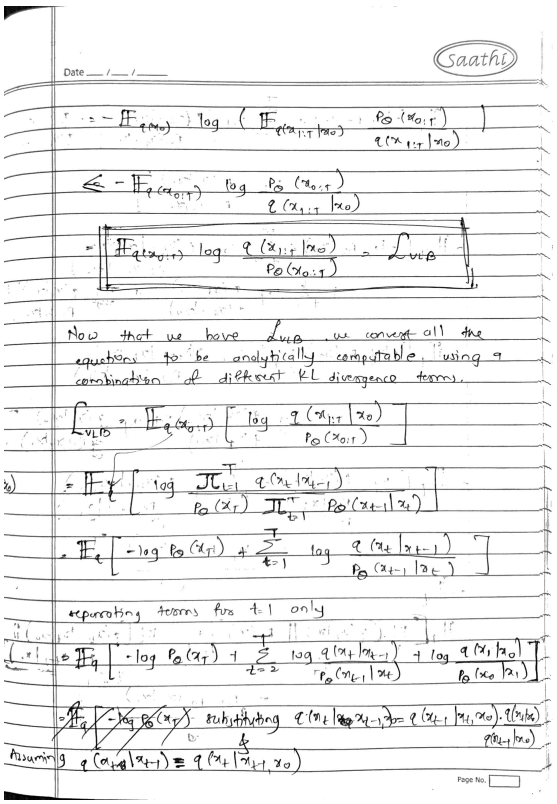
$$= \mathbb{E}_q\left[-\log p_\theta(x_T) + \sum_{t=2}^{T}\log\frac{q(x_{t-1}|x_t,x_0)}{p_\theta(x_{t-1}|x_t)} + \log\frac{q(x_T|x_0)}{q(x_1|x_0)} + \log\frac{q(x_1|x_0)}{p_\theta(x_0|x_1)}\right]$$

$$= \mathbb{E}_q\left[\log\frac{q(x_T|x_0)}{p_\theta(x_T)} + \sum_{t=2}^{T}\log\frac{q(x_{t-1}|x_t,x_0)}{p_\theta(x_{t-1}|x_t)} - \log p_\theta(x_0|x_1)\right]$$

① ② ③

$$= \mathbb{E}_q\left[KL\left(q(x_T|x_0)\,||\,p_\theta(x_T)\right) + \sum_{t=2}^{T}KL\left(q(x_{t-1}|x_t,x_0)\,||\,p_\theta(x_{t-1}|x_t)\right) - \log p_\theta(x_0|x_1)\right]$$

L_T L_{t-1} L_0



$$= -\mathbb{E}_{q(x_0)}\log\left(\mathbb{E}_{q(x_{1:T}|x_0)}\frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)}\right)$$

$$\le -\mathbb{E}_{q(x_{0:T})}\log\frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)}$$

$$= \mathbb{E}_{q(x_{0:T})}\log\frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})} = \mathcal{L}_{VLB}$$

Now that we have $\mathcal{L}_{VLB}$, we convert all the equations to be analytically computable using a combination of different KL divergence terms.

$$\mathcal{L}_{VLB} = \mathbb{E}_{q(x_{0:T})}\left[\log\frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})}\right]$$

$$= \mathbb{E}_q\left[\log\frac{\prod_{t=1}^{T}q(x_t|x_{t-1})}{p_\theta(x_T)\prod_{t=1}^{T}p_\theta(x_{t-1}|x_t)}\right]$$

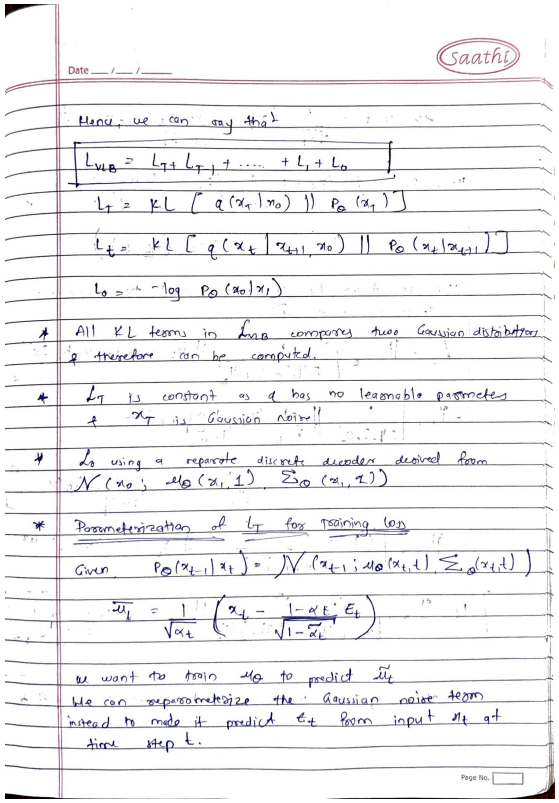
$$= \mathbb{E}_q\left[-\log p_\theta(x_T) + \sum_{t=1}^{T}\log\frac{q(x_t|x_{t-1})}{p_\theta(x_{t-1}|x_t)}\right]$$

separating terms for $t=1$ only

$$= \mathbb{E}_q\left[-\log p_\theta(x_T) + \sum_{t=2}^{T}\log\frac{q(x_t|x_{t-1})}{p_\theta(x_{t-1}|x_t)} + \log\frac{q(x_1|x_0)}{p_\theta(x_0|x_1)}\right]$$

$$= \mathbb{E}_q\left[-\log p_\theta(x_T) \text{ substituting } q(x_t|x_{t-1}) = q(x_{t-1}|x_t,x_0)\cdot\frac{q(x_t|x_0)}{q(x_{t-1}|x_0)}\right.$$

Assuming $q(x_{t-1}|x_t) \equiv q(x_t|x_{t-1},x_0)$



Hence, we can say that

$$\mathcal{L}_{VLB} = L_T + L_{T-1} + \dots + L_1 + L_0$$

$$L_T = KL\left[q(x_T|x_0)\,||\,p_\theta(x_T)\right]$$

$$L_t = KL\left[q(x_t|x_{t+1},x_0)\,||\,p_\theta(x_t|x_{t+1})\right]$$

$$L_0 = -\log p_\theta(x_0|x_1)$$

* All KL terms in $\mathcal{L}_{VLB}$ compares two Gaussian distributions & therefore can be computed.
* L_T is constant as q has no learnable parameters & x_T is Gaussian noise.
* L_0 using a separate discrete decoder derived from $\mathcal{N}(x_0; \mu_\theta(x_1,1), \Sigma_\theta(x_1,1))$
* Parameterization of L_t for training loss

Given $p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t,t), \Sigma_\theta(x_t,t))$

$$\tilde{\mu}_t = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_t\right)$$

We want to train μ_θ to predict $\tilde{\mu}_t$

We can reparameterize the Gaussian noise term instead to make it predict ϵ_t from input x_t at time step t.