CS152-BDL: Checkpoint 1
Tina Wu, Maha Mapara

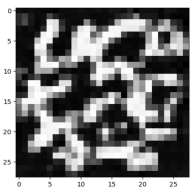
Purpose:
The goal of our project is to understand how Denoising Diffusion Probabilistic Models (DDPM) work and perform on a variety of image data sets.

Hypothesis:
We will compare the performance between diffusion models and variational autoencoders (VAEs) on the same image data sets. Performance will be compared through computational speed, and accuracy.

Evaluation plan:
We will use the MNIST, CIFAR-10 and ImageNet data sets. These are common data sets used in the literature for image synthesis tasks, and are easy to process and understand results on. We will measure training time, FID (Frechet Inception Distance), and ELBO (evidence lower bound).

Section A: Changelog
1. We will transition to using the HPC cluster. We tried to run a diffusion model on MNIST on our PCs and it took a lot of computational power and time.
2. We adapted code from Ho et al. [1] and Dhariwal et al's [2] papers on diffusion models. We followed Ho et al.'s U-Net architecture but got poor quality sample images on the MNIST data, likely due to issues with implementation. Moving to the HPC cluster will allow us to debug our code and train the model again without the issues of needing more GPUs.
3. Given how long training will take for diffusion models along with the implementation being very new to us, we will not use week 5 and 6 (from the project pitch) to look at more recent DDPM research in text to image and classifier guidance.
4. We will also delve deeper into improvements to U-Net architecture made by Dhariwal et al.[2] instead of Ho et al's [2] U-Net implementation.

Section B: Challenges
First, our main challenge is to debug our code for training DDPM on MINST. Our current model is underfitting as one of our sampled image shows:



We suspect that it is a data pre-processing issue, specifically rescaling or normalizing, and we are currently working on that. Second, neither of us have used the HPC cluster, so familiarizing ourselves with it will take some time. We are looking at the tutorials and other information available on the website. Third, training for DDPM takes longer than training for VAE. Therefore, we are prioritizing implementation and training of DDPM.

Section C: Timeline

By checkpoint 2, we plan on finishing DDPM training on the MNIST and CIFAR-10 datasets. Our expected timeline is as follows:

- Transition to using the HPC cluster (using conda on HPC, moving files to HPC) - Maha (by 11/12)
- Figuring out our code issues with DDPM on MNIST and what to ask help for- Maha and Tina (by early next week 11/15)
- Finish training DDPM on MNIST - Tina (by 11/18)
- Train DDPM on CIFAR-10 - Maha (by 11/20)
- We are leaving extra time for unexpected issues. If there is time, we will train DDPM on ImageNet, otherwise that will be done after checkpoint 2.

Section 1: Goals

Our main task with this project is image synthesis using DDPM. Our goal is to compare DDPM performance with VAE on the same image data sets. We expect DDPM to have better performance than VAE due to them having the following key differences:

- The forward process in DDPM has no learned parameters.
- The DDPM forward process progressively destroys all information about the input, such that the final distribution $q(x_t|x_0)$ is a standard gaussian by construction. This is typically not true with VAEs, where we want the latent variable z to contain some information about our input x.
- In DDPM, the additional input t makes one single model able to handle many different noise levels with a single set of shared parameters

This is an interesting project because diffusion models are an exciting area of research in generative models, especially for image synthesis tasks; many popular architectures use diffusion models for this purpose, like DALLE-2 and Imagen. This is a good opportunity to gain in-depth exposure about the architecture and modeling approach used for diffusion models. In particular, we were inspired by Dhariwal et al. 's [2] paper ‘Diffusion Models Beat GAN on Image Synthesis’, where the authors compare DDPM performance with GANs on image synthesis using ImageNet and LSUN image data. Instead of comparing DDPM to GAN, we will compare diffusion models with VAEs. This is because:

1. We have not seen a comparison between VAE and diffusion models in the literature on our choice of data sets.
2. The models have some similarities:
    a. Forward process that turns data into latent representations
    b. Reverse process that involves sampling some latent variable
    c. Training objective which is a lower bound on data likelihood
3. It will be interesting to see how different the model performances are, and evaluate the expected underperformance of VAEs based on our understanding of the key differences between DDPMs and VAEs.

Section 2: Methods

The probabilistic model we are focusing on is the Denoising Diffusion Probabilistic Model. There are two main steps involved: the forward process and the reverse process. The key random variables in these steps are q and p_θ.

First, the forward 'diffusion' process is about moving from an image in our data set (x_0) to a near standard Gaussian distribution (x_T), by adding standard Gaussian noise through a series of T steps (Markov chain of T steps).

- x_0 is sampled from a real data distribution q(x) i.e $x_0 \sim q(x)$
- Gaussian noise is added with a variance of β_t to x_{t-1}
- This results in a new latent variable x_t with distribution $q(x_t|x_{t-1})$
- $q(x_t|x_{t-1}) = \mathcal{N}(x_t; \mu_t = \sqrt{(1-\beta_t)}\, x_{t-1}, \Sigma_t = \beta_t \mathbf{I})$
- The posterior probability is: $q(x_{1:T}|x_0) = \Pi^T_{t=1}\, q(x_t|x_{t-1})$
- Using reparameterization trick, we can instead get $x_t \sim q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{(\propto_t)}\, x_0, (1-\propto_t)\mathbf{I})$
    - This way we can sample our latent variable x_t at any arbitrary timestep.

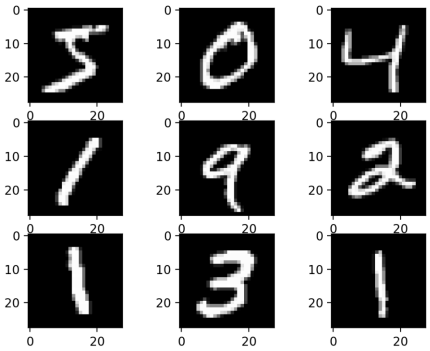
Second, the reverse diffusion process is about approximating the reverse 'denoising' process $[q(x_{t-1}|x_t)]$ using a neural network, p_θ, such that we can sample x_T from $\mathcal{N}(0,\mathbf{I})$. Then it runs the reverse process and acquires a sample from $q(x_0)$. This generates a novel data point from the original data distribution.

- $q(x_{t-1}|x_t)$ is the reverse distribution which is approximated using p_θ because it is intractable
- As $q(x_{t-1}|x_t)$ is Gaussian, we choose p_θ to be Gaussian for a small β_t
- $p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t)\, x_{t-1}, \Sigma_\theta(x_t, t))$
- We can go from x_T to the data distribution using
    - $p_\theta(x_{0:T}) = p_\theta(x_T)\, \Pi^T_{t=1} p_\theta(x_{t-1}|x_t)$

Additionally, image data that we use can come from any distribution which is why DDPM is a very flexible model and applicable on a large range of datasets. Our focus method is DDPM using U-Net architecture to achieve a better performance than VAE on our image synthesis task. VAE, therefore, is our baseline model to compare with the DDPM results.

Section 3: Experiments

The first of three datasets we will be using is MNIST [3], which is an image dataset composed of handwritten digits of 0-9. It is 28x28 pixels in grayscale, with digits centered in the image.



This dataset is widely available and widely used in research. For our purposes, we will load it using the ‘load_dataset’ method from the ‘datasets’ library.

The summary statistics for the MNIST data are:

Num. example	Num dimensions	Range
60000	28 x 28	[0, 255]

Our primary reason for using this dataset is as our trial. This is because MNIST is clear and easy to interpret. Among all image datasets, MNIST is very simple (gray-scale) and takes a shorter time to train, which is appropriate for implementing the algorithm and working through coding issues. Also, MNIST can comprehensively be used to analyze image quality. Therefore, we can compare the results of the diffusion model on MNIST against VAE on MNIST easily.

For our intended experimental results, first, we will be plotting ELBO against iterations for both VAE and diffusion models for the training and testing sets. This is our loss function, which tells how good the model fits the dataset and whether there are signs of overfitting or underfitting. Second, we will tabulate FID scores for both models. This measures how good the qualities of the generated samples are, by calculating the difference between feature vectors of real images and the feature vectors of generated images.The lower the FID score, the fewer distorsions in the

image and better the image quality. Therefore, the model that produces lower FID will have a better performance.

Our implementation plan is to use pytorch for mathematical computations and operations in batch iterations, such as calculating loss and performing gradient descent. As mentioned earlier, we are currently implementing U-Net for our model architecture, which is very new to us. Moreover, we are writing the majority of the code other than U-Nett and batch operations, such as performing gradient descent. The code we are writing can be broken down into 3 major pieces: forward process, reverse process, and training loss. In the forward process, we are trying our different variance schedules (β_t) and sampling q. In the reverse process, we are writing loops to sample p. Lastly, we have a loss function that governs the gradient descent in pytorch.

References

[1] Jonathan Ho, Ajay Jain and Pieter Abbeel. Denoising Diffusion Probabilistic Models.arXiv:2006.11239v2, 2020.

[2] Prafulla Dhariwal and Alex Nichol. Diffusion Models Beat GANs on Image Synthesis. arXiv:2105.05233v4, 2021.

[3] LeCun, Yann and Cortes, Corinna and Burges, CJ. MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist, 2010.