Tensor-Train Diffusion Models

M. Baddar , M. Eigel

April 26, 2024

1 Abstract

In this work, we explore the application of fixed, low-rank tensor-train to Denoising Probabilistic Diffusion Models. We show the parametric noise can be modeled using Functional Tensor-Trains(FTT). We will also provide details on how the model can be trained using Riemannian Optimization algorithm for fixed-rank tensor-trains. The main objective is to develop a more efficient DDPM with respect to memory and training-time. The secondary objective is to conduct a comparative analysis to different FTT optimization methods in the context of DDPM.

2 Background

2.1 Denoising Diffusion Probabilisitce Models (DDPM)

DDPM models are one of the state-of-the-art generative models[HJA20]. Given a random variable $\mathbf{x}_0 \in \mathbb{R}^D$ with density $q(\mathbf{x}_0)$. The task is to find a set of parameters $\boldsymbol{\theta}$ for the approximate density function $p(\mathbf{x}_0; \boldsymbol{\theta})$ such that the log-likelihood $\log(p(\mathbf{x}_0; \boldsymbol{\theta}))$ is maximized.

2.2 Forward and Reverse Process

In DDPM, the training happens in two phases:

Forward Process In this phase, a set of intermediate latent variables are generated $\mathbf{x}_t \quad t = 0, 1, 2, .., T$, where $\mathbf{x}_t$is the input data and $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \quad \mathbf{0} \in \mathbb{R}^D, \mathbf{I} \in \mathbb{R}^{D \times D}$.

Given the following set of constants:

$$
\begin{aligned}
\beta_t, \quad & \leq \beta_t \leq 1 \\
\alpha_t &= 1 - \beta_t \\
\bar{\alpha}_t &= \prod_{i=1}^{T} \alpha_i
\end{aligned}
\tag{1}
$$

The latent variable $\mathbf{x}_t$ has the following distribution:

$$
\begin{aligned}
q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}\right) &= \mathcal{N}\left(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I}\right) \\
q\left(\mathbf{x}_{1:T} \mid \mathbf{x}_0\right) &= \prod_{t=1}^{T} q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}\right)
\end{aligned}
\tag{2}
$$

TBD : Add details for deriving the forward sampling equation: Which can be rewritten as

$$
q\left(\mathbf{x}_t \mid \mathbf{x}_0\right) = \mathcal{N}\left(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, \left(1-\bar{\alpha}_t\right)\mathbf{I}\right) \tag{3}
$$

Accordingly, in the forward phase for each time $t = 1, 2, \ldots T$, a corresponding $\mathbf{x}_t$ is generated by sampling from density in eq.(3). In another word, $\mathbf{x}_t$ can be sampled using the following equation (based on the Re-parameterization trick[KW22, The])

$$
\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon} \tag{4}
$$

Where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

Reverse Process This the process in which we try to train a model the reconstructs $\mathbf{x}_0^{'} \sim q$ f rom normal noise $\mathbf{x}_T$.

Add loss function derivation

The loss function for DDPM can be written as:

$$
L_{\text{simple}}\left(\theta\right) = \mathbb{E}_{t, \mathbf{x}_0, \boldsymbol{\epsilon}}\left[\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta\left(\mathbf{x}_t, t\right)\right\|^2\right] \tag{5}
$$

Where $\mathbf{x}_t$ is generated as in eq. (4),$t = 1, 2, \ldots, T$.(See eq (14) in[HJA20]) and the re-parameterization section in this article [Wha].

How the loss is calculated for each batch of the optimization loop (assuming an iterative optimization algorithm is applied):

1. The parameter values $\boldsymbol{\theta}$ are given, either as the initial or the updated values during optimization.

2. Sample t uniformly from $1, 2, \dots T$

3. Sample $\mathbf{x}_t$ based on eq. 4

4. Sample $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

5. Calculate $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$

6. Calculate the Loss as in eq.(5)

2.3 Modeling $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$

The core modeling task for Diffusion Models is to, a first design and architecture for a model that approximates $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$, then develop an optimization algorithm to find the optimal $\boldsymbol{\theta}$ That minimizes the loss in eq.(5).

In this subsection, we will show the different Neural Network architecture used for different realizations of Diffusion Models and how they are optimized. Also, we will go through the Positional Time Encoding used to embed time into such a model,

2.3.1 Positional Time Embedding

TBD : [Dem, AGe, VSP+23]

2.3.2 ResNet and UNet for modeling $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$

TBD: [HJA20, SDWMG15]

2.4 Functional Tensor-Trains (FTT)

In this section and overview of Functional Tensor Trains is introduced. The details of how tensor-trains can approximate a given function, along with different optimization algorithms, will be illustrated TBD : Add details

2.5 Optimization of Function Tensor Trains

In this subsection, an overview of different optimization methods for Functional Tensor Trains.

Add details for each algorithm, in the corresponding subsection

2.5.1 Alternating Linear Schema

2.5.2 Riemannian Optimization

3 Model

3.1 Architecture

3.2 Optimization

4 Experiments

4.1 Plan

1. Isolation experiments for parametric noise model
2. Use a ResNet, Unet for training such data (for validation)
3. Apply TT opt to isolated data coming from different distributions (the effect of distribution on tt opt)
4. The effect of rank and number of cores,
5. The effect of opt algorithm parameters
6. Drawing a loss landscape for the original DDPM and TT ones
7. Analyzing the convergence of each optimization method
8. Analyzing the number of computations with each method
9. Analyze the memory footprint for each
10. As a plan B , if the TT-DDPM model is not better. This work can result in a comparative analysis to different TT-OPT methods applied to the context of Diffusion Models examples : https://arxiv.org/pdf/2404.16154 , https://arxiv.org/pdf/2404.06455 , https://arxiv.org/pdf/2212.10797

convergence analysis examples https://arxiv.org/abs/2208.05314 https://openreview.net/pdf?id=v https://www.igpm.rwth-aachen.de/Download/reports/pdf/IGPM423.pdf https://www.jstor.org/stable https://youtu.be/4WDedaz$_T$V4?$si = ksqwnfZMYNF_U595https://www.cis.upenn.edu/cis6100/Ri Wirth - optim - Riemann.pdf$**https://arxiv.org/abs/1712.09913**

4.2 Analysis of Gradient Descent Optimization with Neural Networks Model

4.3 Optimization with Gradient Descent for Tensor-Train Model

4.4 Optimization with Alternating Linear Scheme for Tensor-Train Model

4.5 Optimization with Riemannian Gradient Descent for Tensor-Train Model

Why ? OPTIMIZATION METHODS ON RIEMANNIAN MANIFOLDS AND THEIR APPLICATION TO SHAPE SPACE https://www.uni-muenster.de/AMM/num/wirth/fi
"Even if an embedding is known, one might hope that a Riemannian optimization method performs more efficiently since it exploits the underlying geometric structure of the manifold. For this purpose, various methods have been devised, from simple gradient descent on manifolds [25] to sophisticated trust region methods [5]."

4.6 Results

References

[AGe] A gentle introduction to positional encoding in transformer models, part 1 - machinelearningmastery.com. https://machinelearningmastery.com/a-gentle-introduction-to-positional-encoding-in-transformer-models-part-1/. (Accessed on 04/26/2024).

[Dem] Demystifying diffusion models. generative models learns a... — by rucha apte — medium. https://ruchaa.medium.com/demystifying-diffusion-models-c7e0689a7ca8. (Accessed on 04/26/2024).

[HJA20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020.

[KW22] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022.

[SDWMG15] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan,

and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics, 2015.

[The] The reparameterization trick. https://gregorygundersen.com/blog/2018/04/29/reparameterization/kingma2013auto. (Accessed on 04/26/2024).

[VSP+23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[Wha] What are diffusion models? — lil'log. https://lilianweng.github.io/posts/2021-07-11-diffusion-models/. (Accessed on 04/25/2024).