Mini Diffusion Model for MNIST

Aishwarya Sahoo. Gargee Shah. Vaishnavi Daber.

Abstract

Diffusion Models (DMs) have emerged as a new set of models for several generating tasks in the past few years. DMs have also been able to address some of the most important drawbacks of Generative Adversarial Networks (GANs), like mode collapse, overhead of adversarial learning and failure to converge during optimization. The main fundamental step behind the DM is that during its training, we add Gaussian noise to the training data, and then learn to recover the original data from the noisy one which generates a new sample. DMs take inspiration from non-equilibrium thermodynamics, and so have high computational complexity intrinsically during the training stage due to iterative estimation and gradient computations. The computational cost becomes even more huge when dealing with images and videos in particular. For instance, Denoising Diffusion Probabilistic Model (Ho et al., 2020) takes about 4 days and over 2 weeks to train on eight V100 GPUs, for 64 × 64 and 256 × 256 resolution datasets, respectively [1] . It is also noteworthy that evaluating a pre-trained model takes considerable computational costs and time because the model may need to run for multiple steps to generate a sample. This makes practical applications of DMs difficult, especially in resource constrained environments. The most advanced and exciting results have been possible only with training diffusion models with enormous amounts of computational power and data. The restrictive time and data scale has created a bottleneck for further research due to the need of such high-end privileged resources. Through this project, we aim to study and explore creation of lightweight diffusion models, while also maintaining a standard minimum quality requirement. We look forward to comparative studies of different methods employed during our experiments to make the model lightweight.

1. Introduction

A bit about deep generative modelling. What are we trying to achieve. Reducing inference speed by reducing inference speed of the model using quantization and pruning and sampling by using advanced sampling techniques.

1.1. Sampling

The basic framework of diffusion models is creating data from gaussian noise. DMs take inspiration from Brownian Motion and Stochastic Differential Equations (SDE) that smoothly transform complex data distribution to a known prior distribution by injecting a small amount of noise into the data, and a corresponding reverse-time SDE that transforms the prior data distribution back into the data distribution by slowly removing the noise. The critical step in inference, in particular, is to find the solution to the reverse-time SDE where we use numerical SDE solvers to generate samples.

The forward process in DMs is defined by the SDE:

$$\mathrm{d}\mathbf{X} = \mathbf{f}(\mathbf{X}, t)\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}$$

Here, $\mathbf{f}$ is a vector function, and $d\mathbf{w}$ is a sample from a spherical Gaussian with variance dt with time steps from 0 to T. Similarly, the backward process in DMs is also an SDE of the form:

$$\mathrm{d}\mathbf{X} = \big(\mathbf{f}(\mathbf{X}, t) - g^2(t)\nabla_{\mathbf{X}} \log p(\mathbf{X}, t)\big)\,\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}$$

The naive numerical method would be the Euler's method of numerical integration to solve the reverse of a particular Markovian diffusion process. In this project, we use the reverse process by a Non-Markovian diffusion process, leading to implicit method of solving the reverse-time SDE. We further improve the implicit method by using momentum mechanics and adaptive updating as in Adam Optimizer used widely across deep learning.

1.2. Quantization

Diffusion models have achieve success in image generation through iterative noise estimation and sampling process using deep neural networks. However, due to high memory consumption and computation intensity of the noise estimation affects the efficiency of diffusion model and results in slow inference speed. Thus, enters Quantization a method to bring model to a reasonable size, while maintaining the image quality and achieving good performance speed. This is especially important for device-based applications, where memory and number of computations are limited. Quantization is a process which converts floating-point weights and activations to a reduced bit-width representation, common choices are 8-bit or lower. Model quantization can be executed through two predominant approaches: post-training quantization (PTQ) and training-aware quantization (QAT).

Post-training quantization: It involves converting a pre-trained floating-point model into a quantized model, where the weights and activations are represented with reduced bit precision.

Training-aware quantization: Method of quantizing neural network weights during the training process. The goal is to optimize the quantization parameters such that the model's performance is not significantly affected when using reduced-precision (quantized) weights during inference.

While performing PTQ on diffusion model we discovered that for quantized model inference time increased. Though we were able to achieve results in less steps compared to the normal model. There are several factors leading to this 1(a) The quantization process adds overhead during both training and inference. During training, the model needs to adapt to the quantization-induced errors, and during inference, extra operations are required for quantization and dequantization. 2(b) Though Quantized models are often smaller in size due to reduced precision, but this doesn't necessarily translate to faster inference. In some cases, memory access patterns and cache efficiency can impact performance.

To address these challanges, we explore the model with Q diffusion, a PTQ solution to compress the noise estimation in diffusion models, while maintaining performance to the full precision counterparts.

1.3. Pruning

Pruning is an optimization technique aimed at reducing computational complexity, model size, and inference time by removing redundant or less influential network components like weights, neurons, or layers. When applied specifically to Diffusion Probabilistic Models (DPMs), pruning becomes a technique to reduce the computational cost associated with generating samples or performing inference in these generative models. Pruning analyzes the diffusion process in DPMs to identify parts of the model that contribute less significantly to generating high-quality samples and prunes the Parameters or gradients that contribute less to the quality of generated samples. By reducing the number of parameters or connections, pruning aims to decrease computational complexity, potentially reducing the inference time and memory requirements during the generation of samples in diffusion models. Pruning can also lead to a decrease in the performance, in such cases, fine-tuning of the model can help recover the quality. There are various ways in which we can perform Pruning, such as magnitude-based pruning, where parameters with low magnitude are pruned; structural pruning, which removes entire units or channels; or iterative pruning, done over multiple steps, among other techniques.

2. Literature Review

2.1. Denoising Diffusion Probabilistic Models

[1] Diffusion Models are a subset of generative models, which show as much promise as Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs) in image generation. Diffusion Probabilistic Models are a parameterized Markov chains, which are trained using variational inference to produce samples matching the data over finite time. Transitions of chain are learned to reverse the diffusion process, which is again a Markov chain that gradually adds noise to the data in the opposite direction of sampling.

2.2. Denoising Diffusion Implicit Models

[2] Denoising Diffusion Implicit Models uses non-Markovian diffusion processes to sample faster while also maintaining the image quality by exploting the fact that the non-Markovian inference process lead to the same surrogate objective

2.3. Boosting Diffusion Models with an Adaptive Momentum Sampler

[3] The sampling process in Diffusion Probabilistic Models are vulnerable to erratic updates in discrete samples created in the reverse chain. An adaptive momentum sampler, inspired by the Adam optimizer, utilizes momentum mechanisms and adaptive updating to smooth the reverse process, and ensure stable generation, resulting in outputs of enhanced quality. A bit introduction about paper three.

2.4. EfficientDM: Efficient Quantization-Aware Fine-Tuning of low-bit diffusion models.

Quantization is one of the way to accelerate diffusion models. There are two approached to achieve this, post-training quantization (PTQ) and quantization-aware training (QAT). PTQ demonstrates efficiency in terms of both time and computation cost. However, it can outperform in low bit-width case studies. QAT dominants performance degradation but has performed good for computational and data resources. This paper further introduce a data-free, quantization-aware and parameter efficient fine-tuning framework for low-bit diffusion models to capitalize on the advantages while avoiding their respective drawbacks.

2.5. Q-Diffusion: Quantizing Diffusion Models.

Post-training quantization (PTQ) doesn't work well on the diffusion models in comparission of other tasks. This paper propose two techniques: time step-aware calibration data sampling and shortcut-splitting quantization. In our model we have considered time step-aware calibration to tackle the challenges identified in PTQ.

2.6. Structural Pruning for Diffusion Models

The paper[6] explores "Diff-Pruning," a method designed to compress Diffusion Probabilistic Models (DPMs) by addressing their computational complexity. These DPMs, while highly powerful in various generative tasks, often demand substantial computational resources during training and inference. Diff-Pruning introduces a unique approach by utilizing structural pruning specifically tailored for DPMs. At its core, Diff-Pruning employs a Taylor expansion method applied to pruned timesteps within the diffusion process. This strategy aims to identify and remove non-critical parameters, effectively reducing the computational load without compromising the model's generative performance. By discarding less informative diffusion steps and leveraging gradients from essential ones, Diff-Pruning achieves efficient compression. The effectiveness of Diff-Pruning is highlighted by the Empirical evaluations. It demonstrates an impressive 50% reduction in computational costs while requiring only 10-20% of the original training effort. Moreover, such a compression doesn't deteriorate the generative quality. This efficiency in parameter usage contributes to faster inference times, making the compressed model more suitable for real-time or resource-constrained applications.

2.7. Pruning convolutional neural networks for Resource Efficient Inference

The paper introduces an innovative pruning technique for convolutional neural networks (CNNs) designed to accelerate inference while preserving accuracy. It proposes a unique approach that combines greedy criteria-based pruning and backpropagation-driven fine-tuning, with a focus on transfer learning scenarios. This prunes entire feature maps and kernels. It leverages a new criterion based on Taylor expansion, approximating the change in the cost function induced by pruning network parameters. This criterion demonstrates superior performance compared to other criteria, particularly excelling in reducing the size of adapted 3D-convolutional filters while maintaining acceptable accuracy in tasks like recurrent gesture classification. This approach addresses the need for speed in inference, especially on resource-constrained platforms, making it a notable advancement in model compression techniques for neural networks.

3. Preliminaries

3.1. Denoising Diffusion Probabilistic Models

In DDPM, we approximate the data distribution $\mathbf{x_0} \sim q(\mathbf{x_0})$ by latent variable models:

$$p_\theta(\mathbf{x}_0) = \int p_\theta(\mathbf{x}_{0:T}) d\mathbf{x}_{1:T}, \text{ where}$$

$$p_\theta(\mathbf{x}_{0:T}) = p_\theta(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta^{(t)}(\mathbf{x_{t-1}}|\mathbf{x}_t)$$

where x_1, x_2, ... x_T are latent variables with the same dimensions with x_0.

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x_{t-1}}|\mathbf{x}_t), \text{ where}$$

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I})$$

Thus, the forward step of diffusion process from a data distribution to a Gaussian distribution for time step t can be expressed as:

$$\mathbf{x}_t = \sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{1-\alpha_t}\epsilon,$$

where $\alpha_t = \prod_{i=0}^{t}(1-\beta_i)$ and $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Then, a model is trained to model θ to fit the data distribution $\mathbf{x}_0$ by maximizing the variational lower-bound:

$$\max_\theta \mathbb{E}_{q(\mathbf{x}_0)}[\log p_\theta(\mathbf{x}_0)] \leq$$
$$\max_\theta \left(\mathbb{E}_{q(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_T)}[\log p_\theta(\mathbf{x}_{0:T}) - \log q(\mathbf{x}_{1:T}|\mathbf{x}_0)]\right)$$

where $q(\mathbf{x}_{1:T}|\mathbf{x}_0)$ is certain inference distribution, which could be calculated using Bayes Theorem. For the backward process, we sample through the function $p_\theta(\mathbf{x}_t|\mathbf{x}_{t-1}, t)$.

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}\Big(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \sigma_\theta(\mathbf{x}_t, t)\Big)$$

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\alpha_t}}\epsilon_\theta(\mathbf{x}_t, t)\right) + \sigma_t\epsilon_t \quad (1)$$

3.2. Pruning Mathematical Background:

Magnitude Estimation. Pruning based on the magnitude of gradients helps identify less impactful parameters. The L1-norm or L2-norm of gradients ∇L with respect to parameters θ can estimate their importance.

$$\text{Magnitude}(\theta) = \|\nabla L(\theta)\|$$

Thresholding. Parameters with magnitudes below a certain threshold (T) are pruned:

$$\text{Pruned Parameters} = \{\theta \mid \text{Magnitude}(\theta) < T\}$$

Taylor Approximation. Using a Taylor expansion to estimate the impact of pruning parameters on the loss (L) during the diffusion process.

$$\Delta L \approx \frac{\partial L}{\partial \theta} \cdot \Delta \theta$$

4. Methodology

4.1. Implicit Method

DDPM's dependence on the Markovian process results in an issue as the geenrative process is restricted to have similar number of processes as in the diffusion process. Also, the stochasticity causes uncertainity while synthesizing images, resulting in problems while interpolating images in the latent space. The Implicit Method, Denoising Diffusion Implicit Model (DDIM). generalizes the DDPM process as a non-Markovian process. Implicit method is a special case where we consider σ_t to be 0 for all iterations of t. Thus, the entire reversal process becomes deterministic in equation 1. Therefore, we can redefine the reversal process in the following way:

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t)\right)$$

In implicit method, we reduce our sampling trajectory from T steps to τ steps which is a subset of the timesteps $[1, 2, ...T]$. When the length of the sampling trajectory is much smaller than T, it results in accelerated sampling, which leads to faster inference speed.

4.2. Adaptive Method

Similar to accelerated optimization process in neural networks, we can also leverage the momentum method to accelerate our sampling of $\mathbf{x}_0$. We assume $\bar{\mathbf{x}}_t = \frac{\mathbf{x}_t}{\alpha_t}$ and $\mu = \left(\sqrt{\frac{1-\alpha_{t-1}-\sigma_t^2}{\alpha_{t-1}}}\right)$. We can redefine our reversed updates in the following manner:

$$d\bar{\mathbf{x}} = \mu \cdot \epsilon_\theta^{(t)}(\mathbf{x}_t) + \frac{\sigma_t}{\sqrt{\alpha_t - 1}} \cdot \epsilon_t \tag{2}$$

$$\begin{aligned}
\mathbf{v}_{t-1} &= (1-c) \cdot \mathbf{v}_t + c \cdot ||d\bar{\mathbf{x}}_t||_2^2 \\
\mathbf{m}_{t-1} &= a \cdot \mathbf{m}_t + b \cdot d\bar{\mathbf{x}}_t \\
\bar{\mathbf{x}}_{t-1} &= \bar{\mathbf{x}}_t + \frac{\mathbf{m}_{t-1}}{\sqrt{\mathbf{v}_t} + \zeta} \\
\mathbf{x}_{t-1} &= \alpha_{t-1}\left(\bar{\mathbf{x}}_t + \frac{\mathbf{m}_{t-1}}{\sqrt{\mathbf{v}_t} + \zeta}\right)
\end{aligned}$$

where, a is the damping coefficient and b is the history dependent velocity coefficient. $\mathbf{m}_t$ is the current momentum at t-th iteration. We start with prior momentum $\mathbf{m}_T = \mathbf{0}$. Both a, b control the strength of the velocity. c is the decay rate to control moving average of second-order moments, and $\mathbf{v}_T$ is the current moving average of second-order moments. We start with prior $\mathbf{v}_T = 1$. ζ is the smoothening term to avoid numerical instability.

For the final model and sampling process, we use the following algorithm:

Algorithm 1: Implcit Method Sampling with the Adaptive Momentum

```
Data: Initialization: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, $\mathbf{m} = \mathbf{0}$,
      $\mathbf{v}_T = \mathbf{1}$
for $t = T, \ldots, 1$ do
    Update $\bar{d}_{\mathbf{x}_t}$ based on Equation 2;
    $\mathbf{v}_{t-1} = (1-c) \cdot \mathbf{v}_t + c \cdot \|\bar{d}\mathbf{x}_t\|^2$;
    $\mathbf{m}_{t-1} = a \cdot \mathbf{m}_t + b \cdot d\bar{\mathbf{x}}_t$;
    $\mathbf{x}_{t-1} = \sqrt{\alpha_{t-1}}\left(\frac{\mathbf{x}_t}{\sqrt{\alpha}_t} + \frac{\mathbf{m}_{t-1}}{\sqrt{\mathbf{v}_{t-1}} + \zeta}\right)$;
Result: Return $x_0$
```

4.3. Quantization

PTQ is primarily concerned with reducing the memory footprint and computational requirements of a model after training whereas Time step-aware calibration, as implied by the name, is likely focused on improving or adjusting the model based on sequential or time-dependent data. Diffusion models often involve the concept of time embedding.The role of time embedding in diffusion models is to enable the model to learn and represent the evolution of data over time. Hence, the model needs to capture the data distribution that changes over different time steps and as a result PTQ does not work standard on diffusion models. The overall calibration workflow is:

Algorithm 2: Q-Diffusion Calibration

```
Input: Pretrained full precision diffusion model and
       quantized diffusion model [$W_\theta, \hat{W}_\theta$]
Input: Empty calibration dataset $D$
Input: Number of denoising sampling steps $T$
Input: Calibration sampling interval $c$, amount of
       calibration data per sampling step $n$
for $t = 1$ to $T$ do
    if $t\%c = 0$ then
        Sample $n$ intermediate inputs $x_{1t}, \ldots, x_{nt}$
         randomly at time step $t$ from $W_\theta$ and add
         them to $D$;
for $i = 1$ to $N$ do
    Retrieve corresponding weight ($w$) from full
     precision model $W_\theta$;
        $\hat{w} = s.clip(round(\frac{w}{s}), cmin, cmax)$
if do activation quantization then
    for $i = 1$ to $N$ do
        Update the activation quantizers step sizes
         of the $i$-th block with $\hat{W}\theta, W\theta, D$;
where,
$w$: Original weight of the model.
$s$: Scale factor, a positive scaling factor applied to
 the original weight before rounding and clipping.
$c_{\text{min}}$: Minimum allowed value after quantization.
 Values smaller than $c_{\text{min}}$ will be set to $c_{\text{min}}$.
$c_{\text{max}}$: Maximum allowed value after quantization.
 Values larger than $c_{\text{max}}$ will be set to $c_{\text{max}}$.
```

4.4. Pruning

One method of using pruning is to apply it over a pre-trained model. We have used the UNet architecture to build a Diffusion Model. We load the same model and apply different pruning methods. Inorder to avoid the inconsistencies in the layers like BatchNorm or Dropout that happen during training and testing of the model, we set the model to the evaluation mode. We use multiple approaches of Pruning.

Taylor Importance Method[7] : This method essentially uses the first order Taylor approximation. It prunes neural networks based on the Taylor expansion of the loss function concerning network parameters. The Taylor expansion is applied to the loss function with respect to the model parameters (weights) to estimate their importance. It evaluates the impact of individual weights on the overall loss function to identify less important weights for pruning.

The Taylor series expansion of a function f(x) around a point a is given by:

$$f(x) \approx f(a) + f'(a) \cdot (x - a) + \frac{f''(a)}{2!} \cdot (x - a)^2$$

In terms of Loss functions:

$$L(\theta) \approx L(\theta_0) + \nabla L(\theta_0) \cdot (\theta - \theta_0)$$

where L$f(\theta)$ is the is the loss function, $f(\theta)$ represents the model parameters (weights). We compute the gradient of the loss function with respect to the weights ∇ L(f(θ)).

Magnitude-based Importance Method: This is a technique used to prune neural networks by evaluating the importance of individual weights based on their magnitudes. It uses absolute values to identify the less important weights, assuming that weights with lower magnitudes contribute less to the network's performance. It also uses a threshold to prune the weights below it. It uses the thresholding function mentioned in Section 3.4.

We use a pruning ratio of 0.3 here. Having a higher pruning ratio might be beneficial to larger models, but for a smaller model, a higher pruning ratio might lead to overfitting and potentially deteriorating the performance. It is essential to note that we do not prune the last layer. Hence, while pruning we ignore the last layer so that the model can make accurate predictions. We can also group the channels where we wish to prune entire channels or feature maps together, where the dataset is more complicated. We initialise a prune object with the model, sample input and these parameters using MagnitudePruner which targets weights with lower values. We evaluate the magnitude of these gradients for each weight in the network using backpropagation. Post which, we prune the model iteratively.

After pruning, we should also rerun the model again to check if the model is performing well after the parameter reduction.

Diff Pruning[6]: While the basic pruning methods work well for convolutional neural networks, we find limitations to these methods when it comes to Diffusion Models. This is because DMs handle time-series or temporal data where the importance of features might vary dynamically over time. The static derivatives of Taylor Expansion or the absolute weights of Magnitude Importance do not approximate well with the dynamic nature of DMs. Hence, as mentioned in [7], there needs to be a structural pruning that uses temporal information associated with the pruning. Diff-pruning uses the Taylor Expansion method to prune the timesteps.

Algorithm 3: Diff-Pruning Algorithm

```
Input: A pretrained diffusion model $\theta$, a dataset $X$,
       a threshold $T$, and a pruning ratio $p\%$
Output: The pruned diffusion model $\theta'$
$L_{\text{max}} = 0$;
$x = \text{mini-batch}(X)$;
$\epsilon \sim \mathcal{N}(0,1)$;
// Accumulating gradients over
   partial steps with the threshold
   $T$
for $t$ in $[0,1,2,\ldots,T]$ do
    $L_t = \left\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x + \sqrt{1-\bar{\alpha}_t}\epsilon, t)\right\|^2$;
    $L_{\text{max}} = \max(L_{\text{max}}, L_t)$;
    if $L_t/L_{max} \leq T$ then
        break;
    end
    $\nabla_{\theta_{ik}} L_t(\theta, x) = \text{back-propagation}(L_t(\theta, x))$;
end
// Estimating the importance of
   sub-structure $\theta_i$ with the
   accumulated $t$-step gradients
$I(\theta_i, x) = \sum_k \left|\theta_{ik} \cdot \sum_{s=0}^{t} \nabla_{\theta_{ik}} L_s(\theta, x)\right|$;
// Pruning and finetuning
Remove $p\%$ channels in each layer to obtain $\theta'$;
Finetune the pruned model $\theta'$ on $X$;
return $\theta'$;
```

We realise that DMs have high level information at larger t values and finer details at lower t values. While backpropagating the gradients, we apply an additional threshold or criteria to identify less important weights for pruning based on the loss values. Here we set the threshold or criteria as 0.05. We use this threshold to reduce the less-impactful weights. We further estimate the importance matrix based on the accumulated gradients. We would thus aim to balance the removal of less important components while retaining the model's generative capabilities by incorporating a threshold-based criterion and importance estimation based on accumulated gradients. This gives us better results than the basic importance methods.

5. Experiments and Results

To verify the effectiveness of the proposed method, we evaluate it with two widely adopted network structures: DDIM and DDPM. For experiments with DDPM and DDIM, we evaluate it on MNIST dataset. The performance of diffusion models is evaluated with Mean Squared Error (MSE). Results are obtained by sampling 50, 000 images and evaluating them with ADM's TensorFlow evaluation suite. All experiments are conducted utilizing NVIDIA Tesla K80 GPUs and implemented with the PyTorch framework.

5.1. Datasets

MNIST is a subset of a larger set available from MNIST. The MNIST database of handwritten digits has a training set of 60,000 examples, and a test set of 10,000 examples.

5.2. Configurations

Model. We use UNet in our noise predictor network due to its downsampling and upsampling nature. We also use an exponential moving average (EMA) over model parameters.

Baseline. We use standard model with no pruning and quantization and Euler's method of numerical integration as our baseline model.

Sampling Experiments. We assume 1000 timesteps for DDPM. For Implcit Method, the image quality is preserved for 500 timesteps and gets corrupted beyond that. For Adaptive Method.

Quantization Configuration. Model used : DDPM. Batch size : 128. Calibration Interval Ratio : 20.

Pruning Configuration. Model used : DDPM. Batch size : 128. Pruning Ratio : 0.3. Loss Threshold : 0.05. Timesteps: 1000

5.3. Performance and Measurements

These are the results of the experiments conducted for sampling. Here, DDPM stands for Euler's Method for Numerical Integration, DDIM stands for Implicit Method for Numerical Integration while AM stands for Adaptive Momentum.

Sampler	Sampling Time (s) $\downarrow$	MSE Loss $\downarrow$
DDPM	518.782574726	1.08
DDIM	271.17745043500054	1.09
DDPM + AM	518.682356709244343	1.08
DDIM + AM	271.14398500862945	1.09

Table 1. Performance Metrics for Sampling

Method	Inference Time (ms)	MSE$\downarrow$
DDPM	67.234	1.09
DDPM + Quantization	56.29	1.11
DDIM + Quantization	50.818	1.09

Table 2. Performance Metrics for Quantization

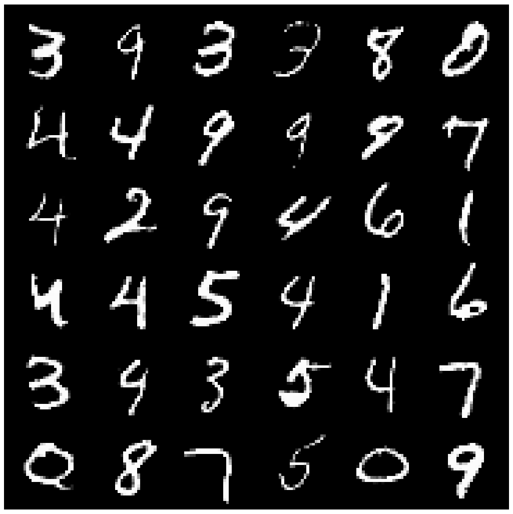

Figure 1. Sampled Image with DDPM (1000 steps)

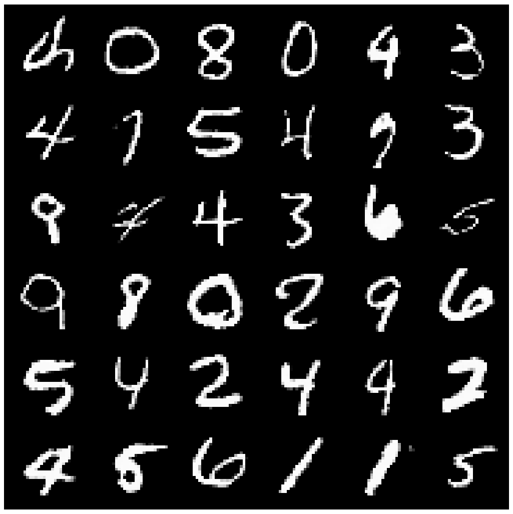

Figure 2. Oversampled Image with DDIM (1000 steps)

Method	Params	MACs	MSE
Diff Pruning	759K	11,070M	1.11
Taylor Importance	1.05M	15,840M	1.11
Mag. Importance	1.07M	16,142M	1.11
Base Model	1K	16,293M	1.11

Table 3. Performance Metrics for Pruning

6. Conclusion

In this project, we have implemented a training free sampler, and employed standard techniques to reduce inference speed of the model. Through both the techniques combined, we manage to increase the inference by 100%. In future works, we propose other complex numerical integration methods like Exponential Integrator and Reduced

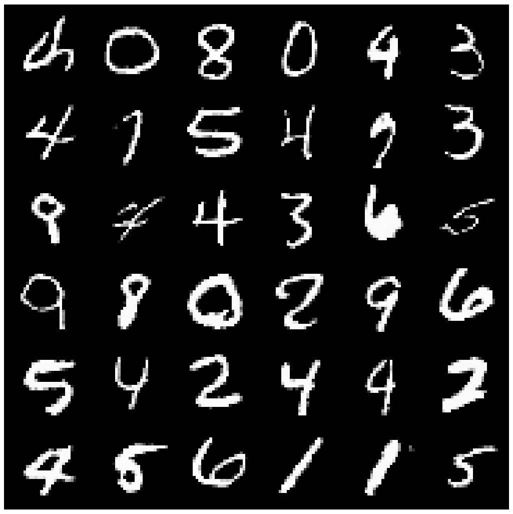

Figure 3. Sampled Image with DDIM (500 steps)

Augmentation Implicit Low-rank methods to achieve faster sampling.

Code available at: https://github.com/aishwaryax/diffusion-model to visit the website.

Video available at: https://github.com/aishwaryax/diffusion-model/project_video.mp4

References

[1] Song, J.; Meng, C.; and Ermon, S (2020) *Denoising diffusion implicit models.*, arXiv preprint arXiv:2010.02502.

[2] Song, J.; Meng, C.; and Ermon, S (2020) *Denoising diffusion implicit models.*, arXiv preprint arXiv:2010.02502.

[3] Xiyu Wang, Anh-Dung Dinh, Daochang Liu, Chang Xu (2023) *Boosting Diffusion Models with an Adaptive Momentum Sampler.*, arXiv preprint arXiv:2308.11941.

[4] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, Jan Kautz (2017) *Pruning convolutional neural networks for Resource Efficient Inference*

Xiuyu Li, Yijiang Liu, Long Lian, Huanrui Yang, Zhen Dong, Daniel Kang, Shanghang Zhang, Kurt Keutzer (2023) *Q-Diffusion: Quantizing Diffusion Models*

Yefei He, Jing Liu, Weijia Wu, Hong Zhou, Bohan Zhuang (2023) *EfficientDM: Efficient quantization-aware fine-tuning of low-bit diffusion models*