Compact Diffusion Models

Jenish Bajracharya
jbajracharya@umass.edu

Suvid Sahay
suvidsahay@umass.edu

Shivam Raj
shivamraj@umass.edu

Abstract

Diffusion models have demonstrated extensive capabilities in synthesizing high-quality images. These models often consist of millions of parameters, which poses a significant barrier to deploying them on resource-constrained devices, such as mobile phones or embedded systems. In this project, we aim to develop a compact diffusion model using popular model compression techniques, including quantization, distillation, and pruning. Our approach employs quantization aware knowledge distillation method as proposed by [16] to train a student model. The method was designed for deep learning framework however we adapt it to diffusion models. We experiment on a smaller and a larger UNet structure of DDPM [10] for which we achieve comparable results with 8bit quantization with the state-of the art methods. With 4 bit quantization our approach tends to fail and give a very high FID score.

1. Introduction

Generative models have demonstrated remarkable capabilities in synthesizing high-quality images, particularly with the advent of diffusion models [11, 25, 32]. These models operate by progressively adding noise to data and subsequently learning to reverse the process, enabling the generation of new data from noise. Their applications span a wide range of areas, including text-to-image generation [30], image synthesis [4, 29], video generation [12, 13], image editing [28, 42], and image translation [36]. This versatility has positioned diffusion models as a cornerstone of modern generative techniques.

Despite their impressive results, diffusion models often incur significant computational costs during training and inference due to their large parameter counts and iterative sampling processes. For instance, the seminal paper on diffusion models, Denoising Diffusion Probabilistic Models (DDPM) [1], describes a model with 35.7 million parameters, while the state-of-the-art DALLE2 [30], which employs four separate diffusion models, contains an astonishing 5.5 billion parameters. The combination of such massive model sizes and computationally intensive architectures makes these models particularly challenging to deploy in resource-constrained environments, such as edge devices, mobile platforms, or latency-sensitive real-time applications.

Furthermore, the inefficiency of the sampling process makes these deployment challenges even worse. Unlike other generative models such as Generative Adversarial Networks (GANs) [6] or Variational Autoencoders (VAEs) [17], which can generate outputs in a single forward pass, diffusion models typically require hundreds or thousands of iterative refinement steps to produce high-quality results. This inherently iterative nature leads to significantly higher computational costs, which are a critical barrier to broader adoption, especially in scenarios where rapid generation is essential.

Numerous efforts have been made to improve diffusion models, primarily focusing on three key areas: enhancing model architectures [26, 31, 40], accelerating sampling methods [35, 38], and optimizing training procedures [39]. For example, architectural improvements have incorporated attention mechanisms and hierarchical features, while faster sampling algorithms have reduced the number of iterations required for generation. However, these advancements often require significant retraining of the models, which is both computationally expensive and time-consuming.

To address these challenges, traditional model compression techniques offer a practical alternative. Knowledge distillation [9] transfers knowledge from a large, pre-trained teacher model to a smaller, more efficient student model, enabling the student to approximate the performance of the teacher while reducing the number of parameters. Similarly, pruning [8] removes redundant parameters, focusing computational resources on the most important parts of the model, while quantization [15] reduces precision levels (e.g., from 32-bit to 8-bit) to lower memory and computational requirements. These compression techniques have shown promise in reducing model complexity while maintaining performance in other domains, making them a compelling choice for addressing the inefficiencies of diffusion models.

In this paper we systematically examine existing model compression techniques and their applicability to diffusion

models. Building on this analysis, we introduce a novel hybrid approach to construct a compact diffusion model that effectively addresses the challenges of high computational costs and resource limitations. Our methodology incorporates a quantization-aware knowledge distillation framework to train a quantized student model capable of retaining the essential features of a larger full precision teacher model. By combining these strategies, the proposed approach achieves a substantial reduction in model size and computational complexity while delivering a marginal improvement in generative performance for some experiments. These advancements position diffusion models as viable candidates for deployment in practical scenarios, including edge computing, mobile applications, and time-sensitive tasks.

2. Background and Related Work

Due to their high computation and parameterization costs, several optimizations have been proposed in the literature to improve the efficiency of diffusion models. A limited number of prior works have explored mobile-friendly architectures via architectural and sampling efficiency for text-to-image diffusion models [20, 27, 43]. Additionally, extensive research has been conducted to enhance diffusion models in terms of architecture, sampling, and efficient training strategies [26, 31, 35, 38, 40].

Diffusion Models Diffusion models, introduced in the seminal paper by Ho, Jain, and Abbeel [10], are a type of generative model that incrementally transform data distributions using a Markov chain. During the forward process, Gaussian noise is added to the data (x_0)progressively in a Markovian manner, gradually degrading the data's structure to obtain more noiser version of the data $(x_1, \ldots, x_T)$. This process is described by the equation:

$$q(x_t|x_{t-1}) = N(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t\mathbf{I}) \tag{1}$$

where t indexes the timesteps, $\beta_t \in (0, 1)$. As $T \to \infty$ the data distribution is a Gaussian distribution such that $x_T \sim N(\mathbf{0}, \mathbf{I})$. In many practical cases T is set to 1000 [10].

The image is further detailed by the marginal distribution $q(x_t|x_0)$, what allows to generated a noisy image in one-shot, such that

$$q(x_t|x_0) = N(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)\mathbf{I}) \tag{2}$$

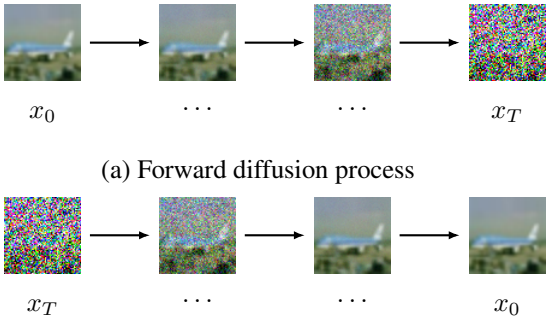
where:

$$\alpha_t = 1 - \beta_t, \quad \bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s, \tag{3}$$

Conversely, the reverse process, or denoising, attempts to reconstruct the original image from the noised state by estimating the conditional distribution $p_\theta(x_{t-1}|x_t)$ using a model parameterized by a neural network. In many state-of-the-art implementations [10, 37] the neural network used is a UNet [34]. This model produces the parameters $\mu_\theta(x_t, t)$ and $\Sigma_\theta(x_t, t)$ aimed at minimizing the differences between the true data and the estimated distribution through the equation:

$$p_\theta(x_{t-1}|x_t) = N(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) \tag{4}$$

The training of diffusion models involves optimizing a variational lower bound (ELBO) [10] on the data likelihood. This objective function integrates the Kullback-Leibler divergence D_{KL} to compare the modeled and actual distributions across various timesteps. This framework allows the diffusion model not only to learn the data distribution effectively but also to generate new data samples that reproduce the original the training data. The forward and reverse diffusion process in illustrated in Figure 1.



(a) Forward diffusion process

(b) Reverse diffusion process

Figure 1. **Forward and Reverse Diffusion Processes.** (a) The forward diffusion process gradually adds noise to an image, transitioning from x_0 to x_T where x_T is Gaussian. (b) The reverse diffusion process reconstructs the image, moving from x_T to x_0.

Quantization Quantization is a widely studied technique for reducing the computational and memory requirements of Deep Neural Networks. The core idea of quantization is to represent weights and activations using lower precision data types (e.g., 8-bit integers instead of 32-bit floating-point numbers) without significantly compromising model accuracy [7, 14]. Quantization techniques can be broadly categorized into two main approaches: post-training quantization (PTQ) and quantization-aware training (QAT). PTQ involves quantizing a pre-trained model without additional training, making it a quick and efficient method [15]. However, it often struggles with accuracy degradation, particularly for complex models. On the other hand, QAT integrates quantization during training, allowing the model to adapt to the lower precision representation and achieve higher accuracy, albeit at the cost of increased training complexity [18].

Knowledge Distillation Knowledge distillation is a another approach the help reduce model complexity and size. It is designed to transfer knowledge from a large, complex model (referred to as the "teacher") to a smaller to a more compact model (known as the "student"). First introduced by Hinton et al. [9], this approach allows the student model to achieve competitive performance while being computationally efficient. The core idea of knowledge distillation is to train the student model not only using the ground truth labels but also by mimicking the soft predictions produced by the teacher model. By learning from this additional information, the student model can generalize better and achieve performance comparable to the teacher, even with significantly fewer parameters. Knowledge distillation has been applied in several forms based on the domain in which it is deployed.

- Response-based Distillation: This is the standard approach, where the student learns directly from the soft predictions of the teacher [9].
- Feature-based Distillation: Here, the student is trained to replicate intermediate representations (features) of the teacher network, capturing finer-grained information about the input data [33].
- Relation-based Distillation: This approach focuses on preserving the relationships between multiple layers or data samples, enabling the student to understand higher-order dependencies in the data [41].

Luhman and Luhman (2021) [24] introduce a knowledge distillation (KD) approach that distills a Denoising Diffusion Implicit Model (DDIM) sampler into a Gaussian model, reducing sampling requirements to a single network function evaluation (NFE). DDIM, a commonly employed deterministic sampling method for Variance-Preserving (VP) diffusion models, enables accelerated and efficient sampling by approximating the diffusion process with fewer intermediate steps. This strategy showcases how KD can simplify the sampling phase in diffusion models without significant quality loss, making it a useful approach for improving diffusion model efficiency in resource-constrained environments.

.

Pruning Pruning techniques have been deployed in the literature to accelerate deep networks and reduce size[21, 23]. Pruning techniques can be broadly categorized into two main types: unstructured pruning and structured pruning. Structure pruning eliminates parameters and substructure such as filters, channels or layers from the network while unstructured pruning masks parameters by zeroing which results in a sparse weight matrix. Pruning in diffusion model posses significant challenges due to the interleaving nature of training.

Previous work done by Fang *et al* [5] introduced a structural pruning method called diff-pruning which employed several techniques such as channel-wise pruning, layer importance evaluation and fine tuning after pruning. While diff-pruning does reduce computational cost by 10-15 % and also reduces the model size significantly, it introduces challenges in maintaining high quality, consistent generation across time steps. Their model does not necessarily improve upon the number of parameters as other pruning techniques however their models do improve upon the FID score (5.29) and SSIM score (0.932).

3. Method

In this section, we first explain Quantization-aware knowledge distillation method and how are we using it on compressing a diffusion model. Specifically, we will work with Denoising Diffusion Probabilistic Model (DDPM) on the CIFAR-10 dataset.

3.1. Quantization-aware knowledge distillation (QKD):

1. **Quantization:** As proposed by Kim et al[16] we implement a trainable uniform quantization method as our baseline approach. We quantize both weights and activations in this process by introducing two trainable parameters to define the interval values for each layer's weight parameters and input activations. We then use a combination of different losses to train these parameters. For k-bit quantization, the weight and activation quantizers are defined as follows.

2. **Weight Quantizer:** In line with the QKD framework [16], where weights can assume both positive and negative values, each weight is quantized to an integer within the range $[-2^{k-1}, 2^{k-1}-1]$. Prior to rounding, weights are constrained within this range using a clamping function $F_W(w) \triangleq F(w, -2^{k-1}, 2^{k-1}-1)$, defined as:

$$f(z) = \begin{cases} \max, & \text{if } x > \max \\ \min, & \text{if } x < \min \\ x, & \text{otherwise.} \end{cases} \tag{5}$$

where F_W is the clamping function, and I_W is the trainable interval parameter. A trainable parameter, I_W, is used to represent the interval value for weight quantization, which is optimized together with the network's weights. The quantization level for a given weight w is determined by rounding F_W. The full quantization-dequantization process for the network weights is defined as follows:
The quantized weight $\hat{w}$ is defined as:

$$\hat{w} = Q_W(w) = \left\lfloor F_W\left(\frac{w}{I_W}\right) + \frac{1}{2} \right\rfloor \times I_W \tag{6}$$

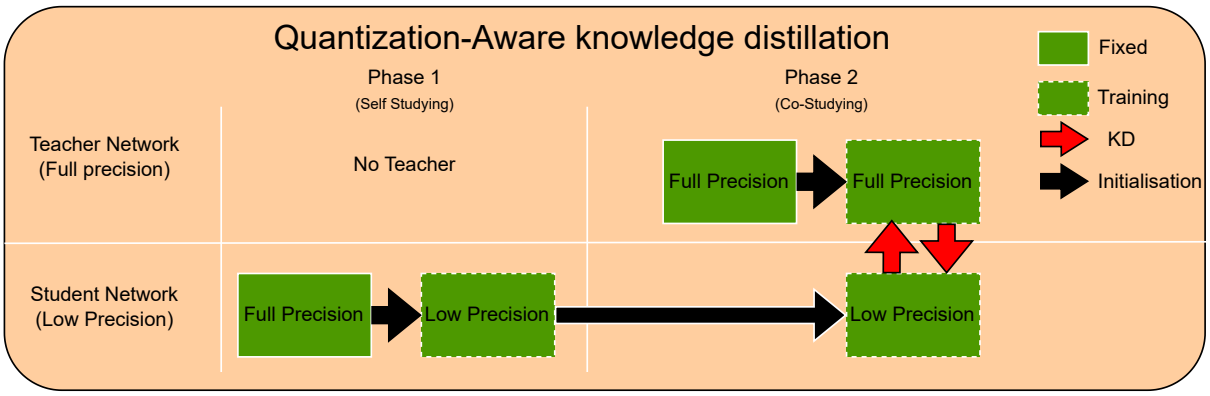


Figure 2. The overall QKD process with Self studying phase for a good starting point for teacher model followed by Co-Studying phase to jointly train Teacher and student model

where $\lfloor . \rfloor$ is he flooring operation. The dequantization step, involving multiplication by I_W, restores the quantized values ($\hat{w}$) to their original range. This step is commonly employed to simulate the impact of quantization.

3. **Activation Quantizer:** Since ReLU is commonly used as the activation function in modern neural networks, the activation values are non-negative. Therefore, to quantize the activations, we employ a quantization function with a range of $[0, 2^k - 1]$. The activation quantizer Q_X is defined as follows:

$$\hat{x} = Q_X(x) = \left\lfloor F_X\left(\frac{x}{I_X}\right) + \frac{1}{2} \right\rfloor \times I_X \tag{7}$$

In this context, $F_X(x) \triangleq F(x, 0, 2^{k-1})$, where x denotes the activation value and I_X represents the interval for activation quantization. Prior to training, the interval values I_W and I_X for each layer are initialized using the minimum and maximum values of weights and activations obtained from a single forward pass, following an approach similar to TF-Lite [2].

Since these quantizers are non-differentiable, we employ a straight-through estimator (STE) [3] to enable gradient backpropagation through the quantization layer. The STE approximates the gradient $\frac{d\hat{x}}{dx}$ as 1, allowing the gradient of the loss function L with respect to x, $\frac{dL}{dx}$, to be approximated by $\frac{dL}{d\hat{x}}$.

$$\frac{dL}{dx} = \frac{dL}{d\hat{x}} \cdot \frac{d\hat{x}}{dx} \approx \frac{dL}{d\hat{x}} \tag{8}$$

3.2. Quantization Aware knowledge Distillation for diffusion models (DDPM):

We propose a 2 phase approach for quantization aware knowledge distillation in diffusion models. We describe the approach as follows:

1. **Phase 1: Self Studying**
Directly performing KD on low precision network leads leads to the model performing poorly. This is because the poor representative power of KD leads to the network learning getting trapped in a poor local minima. To mitigate this issues we provide an initial starting point to the lower precision network by training it for certain number of epochs where the combined loss function is defined as follows:

$$L_{\text{combined}} = \alpha \cdot L_{\text{pixel}} + (1 - \alpha) \cdot L_{\text{SSIM}} \tag{9}$$

where:

- L_{pixel} is the pixel-wise loss, which can either be the L1 loss or the L2 (MSE) loss:

$$L_{\text{pixel}} = \begin{cases} \frac{1}{N}\sum_{i=1}^{N} |y_i - \hat{y}_i| & \text{(L1 Loss)} \\ \frac{1}{N}\sum_{i=1}^{N} (y_i - \hat{y}_i)^2 & \text{(L2 Loss)} \end{cases} \tag{10}$$

- L_{SSIM} represents the Structural Similarity Index Measure (SSIM) loss, computed as:

$$L_{\text{SSIM}} = 1 - \text{SSIM}(y, \hat{y}), \tag{11}$$

where $\text{SSIM}(y, \hat{y})$ measures the perceptual similarity between the predicted image $\hat{y}$ and the ground truth image y.

- $\alpha \in [0, 1]$ is the weighting factor that controls the balance between the pixel-wise loss and the SSIM loss.

This combined loss is useful for tasks such as image reconstruction, where both pixel-level accuracy and perceptual similarity are important.

In the self studying phase, our model is initialised with a good starting parameters which leads to find a good local minima in the second phase.

2. **Phase 2: Co-studying(online studying)**
   In this phase, we jointly train the low-precision student model and the full-precision teacher model using an online method. The loss function consists is a combination of Mean Squared Error (MSE) loss and Kullback–Leibler (KL) Divergence loss which aligns the probability distributions of the teacher (D_T) and the student (D_S) at the pixel level. Using softened probabilities with a temperature T. the KL divergence is computed as:

$$KL(D_T \,||\, D_S) = \sum_i D_T(i) \log \left(\frac{D_T(i)}{D_S(i)} \right) \tag{12}$$

   where $D_T(i)$ is the teacher's probability distribution for pixel i, $D_S(i)$ is the student's probability distribution for pixel i.
   We then compute the loss for each network as follows:

$$L_S^{\text{KD}} = L_S^{\text{combined}} + T^2 \times KL(z_T \parallel z_S; T) \tag{13}$$

$$L_T^{\text{KD}} = L_T^{\text{combined}} + T^2 \times KL(z_S \parallel z_T; T) \tag{14}$$

   where T is the temperature parameter, which softens the distributions for better alignment, and the scaling factor T^2 ensures consistent gradient magnitudes during optimization.

4. Results

Implementation Details We perform a comprehensive evaluation of our proposed method on the CIFAR10[19] dataset. We compare our proposed quantization aware knowledge distillation for DDPM methods on 4, 8-bits width against our baseline model. We adapted our baseline models to reflect the UNet used in DDPM [10]. Our first baseline model *SmallDiffusion* inherits the UNet from DDPM but with smaller parameter size. Our second baseline model LargeDiffusion is reflective of the UNet used for DDPM. We evaluate our proposed models based on model size, the number of parameters, and the Friechet Inception Distance (FID) score. We show our results against few of the state of the art compressed diffusion models. However our results are not comparable with the state-of-the-art implementation.

Setting Our *LargeDiffusion(5 million parameters)* and *SmallDiffusion(300k parameters)* models are compared against the following methods for performance evaluation:

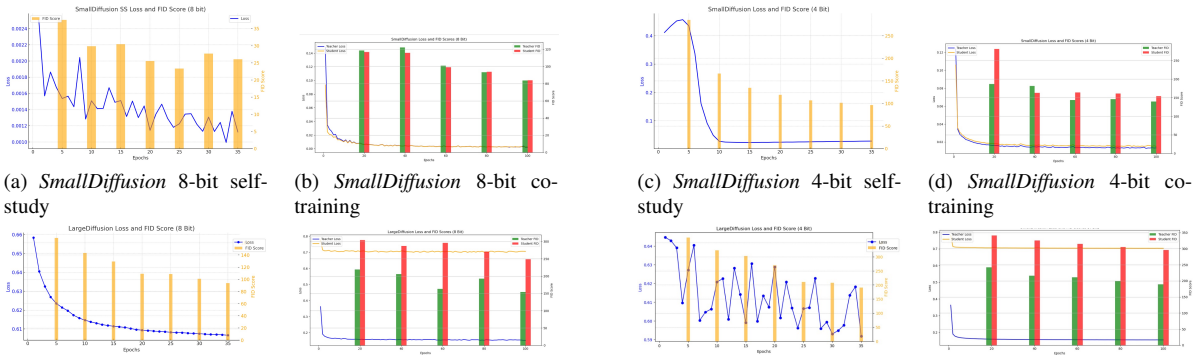
1. **Baseline (BL):** The baseline model for quantization is the UNet architecture used in DDPM.
2. **Self-Studying (SS):** We perform direct knowledge distillation on the quantized UNet to train the student model without additional supervision.
3. **Co-Training (CO):** In this method we co-train the low bit student model from the self-studying phase with a teacher model.

For both *LargeDiffusion* and *SmallDiffusion* models, we conduct four experiments each, evaluating both 4-bit and 8-bit quantization. We train the baseline model for up to 100 epochs with learning rate of 10^{-3}. We use 35 epochs for the SS phase. The original paper divides the learning rate by 10 after every 10 epochs however we use a constant learning rate for the SS phase. We use 100 epochs for the CO phase with the same learning rate as BL phase.

We summarize the results from our experiments are give in Figure 3. At every 20 epoch we report the FID score and report the loss. We do not report the results for the baseline. After 100 epochs the loss is – and the FID score is –. We present the results for our *SmallDiffusion* and *LargeDiffusion* for each of the methods varying the quantization bit. The results for the 4 experiments:

1. ***SmallDiffusion* with 8-bit quantization**: The **SS** phase shows significant improvement for the student model, where the FID score reaches 25 after 35 epochs. In the **CO** phase, the FID score for the student starts higher than the initial FID, with the final FID reaching around 80 for both the teacher and student models.
2. ***SmallDiffusion* with 4-bit quantization**: Similar to the 8-bit quantization, the **SS** phase shows that the FID score for the student model drops to 100 after 35 epochs. However, in the **CO** phase, the FID score for the student starts higher than the initial FID, and the final FID reaches around 150 for both the teacher and student models, which is higher than in the SS phase.
3. ***LargeDiffusion* with 8-bit quantization**: For 8-bit quantization, the **SS** phase shows a slight decrease in the loss function, and the FID score drops below 100. However, in the **CO** phase, the FID score reaches 150 for the teacher model and 250 for the student model, as the loss for both models saturates at very different thresholds.
4. ***LargeDiffusion* with 4-bit quantization**: Similar to the 8-bit quantization, the **SS** phase shows that the FID score for the student model drops to 200 after 35 epochs. However, in the **CO** phase, the FID score for the student starts higher than the initial FID, with the final FID reaching around 300 for the student and 200 for the teacher model, which is higher than in the SS phase.

We now compare our results with the state-of-the-art implementations. For quantization, the state of the art methods are linear-quant [22], Squant [22], and Q-Diffusion [22]. We give the summary in Table 1 approaches achieve comparable results against the baseline model they use to compare. Both our approaches SS and CO achieve close results to the baseline models we trained. However for the 4-bit quantization, our FID scores are very large.



(a) *SmallDiffusion* 8-bit self-study (b) *SmallDiffusion* 8-bit co-training (c) *SmallDiffusion* 4-bit self-study (d) *SmallDiffusion* 4-bit co-training

(e) *LargeDiffusion* 8-bit self-study (f) *LargeDiffusion* 8-bit co-training (g) *LargeDiffusion* 4-bit self-study (h) *LargeDiffusion* 4-bit co-training

Figure 3. Loss and FID scores for various methods. The top row depicts the loss and FID scores across 4 and 8 bit quantization for our *SmallDiffusion* model. The second row represents the same variation for our *LargeDiffusion* model.

Model	Method	Size (MB)		FID	
		Bit = 8	Bit = 4	Bit = 8	Bit = 4
Small Diffusion	Baseline (Teacher)	1.14	1.14	26.48	72.31
	Self-Studying (Student)	0.07	0.14	26.56	96.38
	Co-Learning (Student)	0.07	0.14	83.88	154
Large Diffusion	Baseline	20.07	20.07	23.37	49.58
	Self-Studying	1.21	2.51	93	190
	Co-Learning	1.21	2.51	250.98	300.09
Full Precision Baseline		143.2	143.2	4.22	4.22
Linear Quant		35.8	17.9	4.71	141.0
SQuant		35.8	17.9	4.61	160.0
Q-Diffusion		35.8	17.9	4.27	5.09

Table 1. Evaluation against state-of-the-art quantization methods. Our methods are not directly comporable to the state-of-the-art methods however, we see that the SS method for 8 bit quantization performs comparable to the 8bit quantization for the state-of-the-art methods.

5. Conclusion

In this paper, we explored various methods associated with deploying diffusion models in resource-constrained environments. By using popular model compression techniques like quantization and knowledge distillation, we proposed a quantization-aware knowledge distillation framework initially developed for deep neural networks to adapted for diffusion models. We validated our results through extensive experimentation on both small and large UNet architectures within the DDPM framework.

Our results demonstrated that 8-bit quantization enables significant model size reduction and computational efficiency while maintaining generative quality comparable to state-of-the-art methods. This is evident from the FID score as the Self-study method achieves an FID score of 26.56 as compared to 26.48 of the baseline model. We believe that fine-tuning our 8bit method could optimize the method better. However, our experiments also revealed limitations with 4-bit quantization, where generative performance degraded substantially, as evident by high FID scores. This could possibly be because of various reasons. We primarily suspect that this could be because of the Markov chains that diffusion models deploy. Additionally we believe that our choice of loss function could be improved to further get better results. We believe that further refinement are required to our approach viable for diffusion models.

References

[1] https://huggingface.co/google/ddpm-cifar10-32. 1

[2] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mane, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viegas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: Large-scale machine learning on heterogeneous distributed systems, 2016. 4

[3] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation, 2013. 4

[4] Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis. *CoRR*, abs/2105.05233, 2021. 1

[5] Gongfan Fang, Xinyin Ma, and Xinchao Wang. Structural pruning for diffusion models, 2023. 3

[6] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. In *Advances in neural information processing systems*, 2014. 1

[7] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep learning with limited numerical precision, 2015. 2

[8] Song Han, Jeff Pool, John Tran, and William J Dally. Learning both weights and connections for efficient neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015. 1

[9] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 1, 3

[10] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020. 1, 2, 5

[11] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020. 1

[12] Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P. Kingma, Ben Poole, Mohammad Norouzi, David J. Fleet, and Tim Salimans. Imagen video: High definition video generation with diffusion models, 2022. 1

[13] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J. Fleet. Video diffusion models, 2022. 1

[14] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks: Training neural networks with low precision weights and activations, 2016. 2

[15] Benoit Jacob, Steven Kligys, Bo Chen, Ming Zhu, Ming Tang, Andrew Howard, and Hartwig Adam. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 2

[16] Jangho Kim, Yash Bhalgat, Jinwon Lee, Chirag Patel, and Nojun Kwak. Qkd: Quantization-aware knowledge distillation, 2019. 1, 3

[17] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 1

[18] Raghuraman Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A whitepaper, 2018. 2

[19] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009. 5

[20] Alexander C. Li, Mihir Prabhudesai, Shivam Duggal, Ellis Brown, and Deepak Pathak. Your diffusion model is secretly a zero-shot classifier, 2023. 2

[21] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets, 2017. 3

[22] Xiuyu Li, Yijiang Liu, Long Lian, Huanrui Yang, Zhen Dong, Daniel Kang, Shanghang Zhang, and Kurt Keutzer. Q-diffusion: Quantizing diffusion models, 2023. 5

[23] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming, 2017. 3

[24] Eric Luhman and Troy Luhman. Knowledge distillation in iterative generative models for improved sampling speed, 2021. 3

[25] Alex Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models, 2021. 1

[26] Hao Phung, Quan Dao, and Anh Tran. Wavelet diffusion models are fast and scalable image generators, 2023. 1, 2

[27] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis, 2023. 2

[28] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion, 2022. 1

[29] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents, 2022. 1

[30] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents, 2022. 1

[31] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2022. 1, 2

[32] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2022. 1

[33] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets, 2015. 3

[34] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015. 2

[35] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models, 2022. 1, 2

[36] Hiroshi Sasaki, Chris G. Willcocks, and Toby P. Breckon. Unit-ddpm: Unpaired image translation with denoising diffusion probabilistic models, 2021. 1

[37] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models, 2022. 2

[38] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models, 2022. 1, 2

[39] Zhendong Wang, Yifan Jiang, Huangjie Zheng, Peihao Wang, Pengcheng He, Zhangyang Wang, Weizhu Chen, and Mingyuan Zhou. Patch diffusion: Faster and more data-efficient training of diffusion models, 2023. 1

[40] Xingyi Yang, Daquan Zhou, Jiashi Feng, and Xinchao Wang. Diffusion probabilistic model made slim, 2022. 1, 2

[41] Junho Yim, Donggyu Joo, Jihoon Bae, and Junmo Kim. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7130–7138, 2017. 3

[42] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models, 2023. 1

[43] Yang Zhao, Yanwu Xu, Zhisheng Xiao, Haolin Jia, and Tingbo Hou. Mobilediffusion: Instant text-to-image generation on mobile devices, 2024. 2