682 Report: Compact Diffusion Model for CIFAR-10

Tirth Bhagat
tbhagat@umass.edu

Shreya Birthare
sbirthare@umass.edu

1. Introduction

For the class project, we plan on compacting a diffusion model for CIFAR-10 so it would use less computation and memory compared to the baseline model. The motivation for this project is to improve the computational and memory efficiency of diffusion models, which are known for their high-quality image generation but often require substantial resources. Current diffusion models are typically deployed on expensive, power-hungry hardware, making them less feasible for real-world applications on resource-constrained devices. By compacting a diffusion model specifically for CIFAR-10, we aim to explore the potential trade-offs between model size, computational demands, and image quality. This project focuses on making diffusion models more practical for deployment in resource-constrained environments, such as mobile devices and edge platforms, through the use of knowledge distillation and quantization. The quality of compact of the compact model will be evaluated by the amount of compression, inference time, flops, and FID.

1.1. Dataset

For this project, we plan on using the CIFAR-10 dataset. The dataset contains 60000 32x32 color images in 10 classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck), with 6000 images per class. The data is split with 50000 images being a part of the training set and 10000 being in the test set. It is the ideal dataset as it has many unique examples for the model to train on and has a small image size, allowing for faster image generation.

2. Literature Review

2.1. Denoising Diffusion Probabilistic Models (DDPM)

Generative models have experienced rapid advancement over recent years, leading to impressive results across various domains, including image synthesis, audio generation, and text generation. Among these, Denoising Diffusion Probabilistic Models (DDPM) introduced by Ho et al. [2] have emerged as a prominent approach for generating high-quality images by leveraging the principles of nonequilibrium thermodynamics and Markovian processes. Unlike traditional generative adversarial networks (GANs) or autoregressive models, DDPMs operate through a diffusion process that iteratively adds noise to data and subsequently reverses this process to denoise samples, effectively generating new data points. Figure 13 (last page) shows the workflow on how the authors generate the image from random noise on the CelebA-HQ dataset. This approach offers promising advantages over GANs, such as avoiding adversarial training instability and mode collapse, making it well-suited for generating realistic images with detailed structures and textures. DDPMs achieve state-of-the-art sample quality, as demonstrated by their performance on benchmark datasets. For instance, the authors report an FID score of 3.17 on CIFAR-10, surpassing most models in the literature, including class-conditional models. When computed for the test set, the FID score remains strong at 5.24, outperforming many training set FID scores reported for comparable models. Additionally, the model's Inception score and negative log-likelihood scores highlight its capability for high-quality, lossless sample generation, solidifying DDPM's role as a leading approach in generative modeling. Despite DDPMs' strengths, the model's computational intensity and high memory demands pose limitations for real-world deployment, particularly in resource-limited environments. Each noise iteration in DDPMs adds computational overhead, making the model challenging to deploy in real-time or mobile contexts. To address these challenges, we plan to enhance the DDPM framework proposed above with knowledge distillation, and quantization. Knowledge distillation will involve training a compact student model to approximate the performance of the original model while operating more efficiently. Finally, quantization will downscale the precision of model weights, further lowering memory usage and computational demand, and facilitating deployment in resource-constrained environments. Together, these approaches will be evaluated to measure their impact on efficiency, performance, and scalability in image synthesis tasks.

2.2. Knowledge Distillation

In the paper "A Comprehensive Survey on Knowledge Distillation of Diffusion Models," [4] the authors provide an overview of the knowledge distillation technique for diffusion models, which can allow one to create a faster and smaller than the original without significantly sacrificing performance. Knowledge distillation can be categorized in three primary approaches: Diffusion-to-Field (D2F) Distillation, Diffusion-to-Generator (D2G) Distillation, and Accelerated Sampling Algorithms. In D2F, it distills the generative vector fields of a teacher model into a more efficient representation in a student model, thus making the student model more efficient(techniques include output and path distillation). D2G distillation uses a diffusion model to train another generator model. Accelerated Sampling Algorithms focus on training-free and training-based methods to speed up sampling.

The paper "Knowledge Diffusion for Distillation" by Tao Huang et al.[3] introduces DiffKD, a method for Knowledge Distillation (KD) aimed at bridging the representation gap between teacher and student models using diffusion models. Traditional KD methods rely on complex feature alignments or task-specific loss functions to reduce noise in student features, which can be limiting. DiffKD leverages diffusion models to denoise student features, viewing them as noisy versions of teacher features due to the student's lower capacity. The denoised student features are then used for more effective distillation.

2.3. Quantization

In the paper, "A Survey on Methods and Theories of Quantized Neural Networks" by Yunhui Guo [1], reviews techniques used for quantizing neural networks. Quantizing is a method that reduces the size and computation of the model by reducing the precisions of parameters and activations and gradients in the model (e.g. instead of representing parameters with float64, use float32 or float16) without substantial loss in performance. The author discusses deterministic approaches (rounding, vector quantization), and stochastic approaches, which use probabilistic methods, to quantize a network. The author states that quantization works not all the parameters in the model are important, especially in a deep network.

The paper, "Post-Training Quantization on Diffusion Models" by Yuzhang Shang et al.[5] explores accelerating diffusion models (DMs), which are computationally expensive due to lengthy iterative noise estimation and the use of large neural networks. The authors propose using Post-Training Quantization (PTQ) as a training-free method to compress noise estimation networks in DMs, making them faster and lighter without performance degradation. They introduce a Normally Distributed Time-step Calibration (NDTC), which samples calibration data from skew-normal distributions across time steps to better align with DM-specific structures. They are able to get a FID score of 7.14 using 4000 time steps on the CIFAR-10 dataset.

3. Method

Three factors contribute to the high memory usage and runtime of diffusion models. First, the number of time steps used by the model's scheduler. To generate an image, the model will need to pass the noisy input sampled from a Gaussian distribution into the model to produce a less noisy image and cycle it back to the model for t time steps. Typically, using more time steps produces better results but at the cost of inference speed. Second, model size affects performance; larger models can capture data distribution better but they can slow down the forward and backward passes of the model due to it having more parameters. To create a more compact diffusion model, the aim is to achieve comparable results to the baseline while using fewer timesteps and a smaller model architecture. Third, the data type of each layer can affect memory usage and runtime of the model. Increasing the precision requires increasing the number of bits in the computation, which requires more memory to store the representation and longer computations. Decreasing the number of bit representations will lead to less precision but less computational and memory demands. The teacher and student models will have to own their schedulers. The teacher has a maximum time step of t and the student has a maximum time step of t'. Since the goal is to make the teacher lightweight, it is the case that $t' < t$.

3.1. High Level Approach

The training process involves three machine learning models: a teacher model, a student model (with some layers having reduced precision), and the inceptionV3 model. The teacher model represents the model we aim to compress, while the student model is a distilled version with a smaller architecture and fewer time steps. The inceptionV3 model will be used as a feature extractor for comparing the features generated by the teacher and student models. Student and teacher models are defined using a U-Net architecture.

During training, the student model will receive a batch of n images from CIFAR-10, where each of the channels has been normalized in the range of [-1, 1] with a standard deviation of 0.5. For each image, n_i in 0 to n, a time step from 0 to t' is uniformly sampled called t'_i. Then a set of noisy images is created by repeatedly applying a small amount of Gaussian noise to the image n_i for t'_i times. The noisy images along with the time steps are passed into the student model, where the model tries to learn how to reconstruct the noisy image at the sampled time step. Then, it calculates the diffusion model loss, $Loss_1$, as the mean squared error between the noisy input and the student model's reconstruction output. Before propagating $Loss_1$ back into

the student model, $Loss_2$ needs to be calculated. To do this, the teacher model takes the same set of n CIFAR-10 images. And for each image, n_i a new noisy image will be created by repeatedly applying a small amount of Gaussian noise to n_i for $(t'_i \cdot \frac{t}{t'})$ times. Once created, the new noisy images are passed into the teacher model, which creates a reconstruction of the new noisy input images. Before passing the teacher and student model's output to the inceptionV3 model, the output is resized to 299x299, and normalized in the range of [0, 1] using the mean of 0.485, 0.456, 0.406 and standard deviation of 0.229, 0.224, 0.225 for each of the respective RGB channels. The student and teacher outputs are independently passed through inceptionV3 to extract feature score vectors of size 2048. The vectors are taken from the second-to-last layer (global average pooling) of the inceptionV3 model. Then teachers' feature vectors and students are compared using mean squared error or $Loss_2$. The combined loss of $Loss_1$ and $Loss_2$ is then backpropagated to train the student model.

Once the model has been trained, the quantization will be applied to the layers of the model.

The goal of this approach is to encourage the student to match the teacher's output distribution by comparing features created during the reconstruction of the input, ultimately aiming to replicate the teacher's performance with a compact model.

The diffusion models will be created using the diffuser library by Hugging-Face and trained using a pytorch framework.

3.2. Teacher Model

For this project, the teacher model is 'google/ddpm-cifiar10-32' (found in Hugging-Face) which is a DDPM based on the implementation of Ho et al.'s paper [2]. The model implements a U-net architecture consisting of 3 down-sampling blocks an attention block (down) with 128, 256, 256, and 256 channels. Each block has its respective up-sampling blocks. The model has a linear scheduler consisting of 1000 time steps.

3.3. Desired Model

To compare the results among the many experiments, we will need a standard model that is smaller than the teacher's architecture. For this project, we want distill to a model that contains one down-sampling block and an attention block with 128 and 256 channels respectively (and the corresponding up-sampling blocks). And has a linear scheduler consisting of 500 time steps. This is roughly half of the U-Net architecture and half the maximum time step of the teacher model. This model will be referred to as *Mini Model*.

4. Experiments

4.1. Baseline:

To see the effects of knowledge distillation, a set of baseline results must be established. In the context of the project, the baseline results would be created by training *Mini Model* without knowledge distillation and quantization, or just training the diffusion model as normal. *Mini Model* will be trained by computing and back-propagating $Loss_1$ only (and not $Loss_2$)as defined in the High-Level Approach. We will call this model as 'Baseline Model'.

4.2. Experiment 1: High-Level Approach without Quantization

For this experiment, we follow the high-level approach with the student model being *Mini Model* without quantization. The goal of this experiment is to see how well the model improves at image generation compared to the Baseline by learning the features of the teacher model output without quantization while retaining the inference speed of the smaller model. Here we use both $Loss_1$ and $Loss_2$ propagate the summation of both these losses back to the student model as described in the method section. We will call this model as 'Student Model'.

4.3. Exeriment 2: High-Level Approach with 8 bit Quantization

Using the student model from experiment 1, we will apply post training 8 bit quantization on the student model. Currently, the model is using float32 datatype to represent the model. However, to reduce memory usage, we can linearly map each layer's parameters to 8 bits. The float representation will map to the largest value in the 8 bit representation, the smallest float 32 representation will map to the smallest value under the 8 bit representation, and all other values will be assigned a value proportional to the scale. And to de-quantize, you can take the inverse of the linear mapping. To achieve this we use the BitsAndBytes library provided by HuggingFace. We will call this model as 'Quantized 8-bit Student Model (Q8-SM)'.

4.4. Expirment 3:High-Level Approach with 4 bit Quantization

Using the model from experiment 1, we will apply post training 4 bit quantization on the student model. Currently, the model is using float datatype to represent the model. However, to reduce memory usage, we can linearly map each layer's parameters to 4 bits. The float representation will map to the largest value in the 4 bit representation, the smallest float 32 representation will map to the smallest value under the 4 bit representation, and all other values will be assigned a value proportional to the scale. And to de-quantize, you can take the inverse of the linear mapping.

To achieve this we use the BitsAndBytes library provided by HuggingFace. We will call this model as 'Quantized 4-bit Student Model (Q4-SM)'.

4.5. Expirment 4: Gradual Knowledge Distillation without Quantization

For this experiment, instead of distilling the teacher model into the smallest model possible like in experiment 1, we will distill the teacher into a slightly smaller model. Then we let this slightly smaller student model become the teacher and it then distills further into a smaller model.The goal of this experiment is to see if this will lead to a better distillation process while retaining most of the baseline's learned distribution. We call this model as 'Gradual Distillation Student Model (GDSM)'.

4.6. Expirment 5: Gradual Knowledge Distillation with int8 Quantization

Using the model from experiment 4, we will apply post training 8 bit quantization on the student model. Currently, the model is using float datatype to represent the model. However, to reduce memory usage, we can linearly map each layer's parameters to 8 bits. The float representation will map to the largest value in the 8 bit representation, the smallest float 32 representation will map to the smallest value under the 8 bit representation, and all other values will be assigned a value proportional to the scale. And to de-quantize, you can take the inverse of the linear mapping. We call the model as 'Quantized 8-bit Gradual Distillation Student Model (Q8-GDSM)'.

4.7. Expirment 6: Gradual Knowledge Distillation with int4 Quantization

Using the model from experiment 4, we will apply post training 4 bit quantization on the student model. Currently, the model is using float datatype to represent the model. However, to reduce memory usage, we can linearly map each layer's parameters to 4 bits. The float representation will map to the largest value in the 8 bit representation, the smallest float 32 representation will map to the smallest value under the 4 bit representation, and all other values will be assigned a value proportional to the scale. And to de-quantize, you can take the inverse of the linear mapping. We call the model as 'Quantized 4-bit Gradual Distillation Student Model (Q4-GDSM)'.

5. Results

Note: The experiments were conducted using T4 GPUs on Google Colab.

5.1. Metrics

The results of the experiments will be judged through many metrics to see if the student model was successful. The first two metrics are file size and the total number of parameters of the model. Looking at the file and parameter size will give us an idea of how large the model is and idea of how long it will take an input to pass through the model. The next metrics are test loss and FID scores. Test loss will be the average $Loss_1$ (as defined in the High-Level Approach) on the CIFAR-10 test set; this score should roughly give us an idea of how the model can reconstruct noisy images at different time steps. FID or Fréchet inception distance evaluates how similar the generated images are to real images in terms of visual quality as well as the diversity of the images (the lower the FID is the better). In the experiments, the FID score will be determined using 10,000 images from the CIFAR-10 testing set and 10,000 images generated by the model. The last two metrics are Flops and inference time; these metrics will be used to measure the number of operations needed to run a single reconstruction at a given time step and the time needed to generate an image. For inference time plotting, we have run the pipeline for 5 times and took the average inferencing time taken for the 5 runs. We plot a graph that shows the effect of increasing the number of batch size for inferencing v/s time. We do this for all our models stated above.

5.2. Teacher Model Results

The 'google/ddpm-cifiar10-32' model contains 35,746,307 learnable parameters and has a file size of 143.2 MB and has a scheduler size of 1000 time steps. The validation loss on the CIFAR-10 test set came out to be 0.0302436245. The FID score of the model was 16.78. It takes 6.053 giga flops for an image at time step t to pass through the model. Figure1 is the inference speed . Below are images generated by the model.

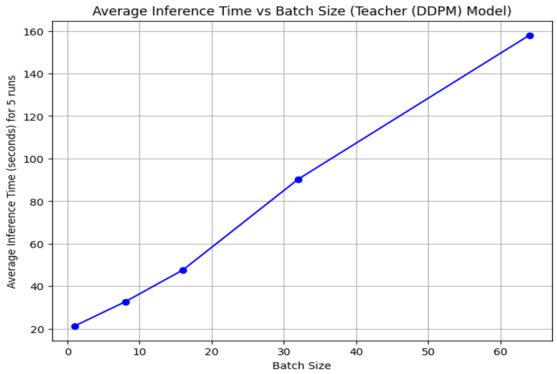


Figure 1

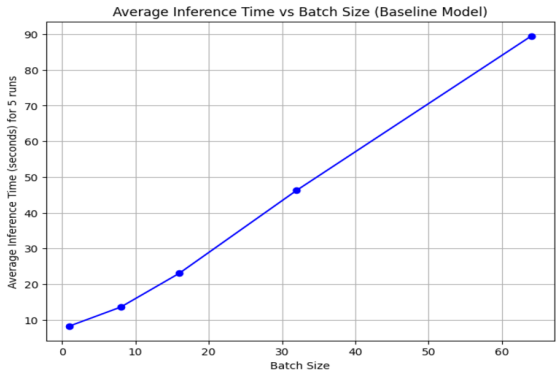
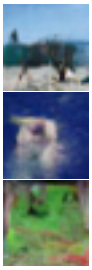


Figure 2

Qualitatively, the images generated by the diffusion are clear, and is easy to tell what they are at first glance. There are no visible distortions or blurriness in the images. From the graph above we see that when you pass the number of batches as 1 for inferencing it takes approx 20 seconds to generate an image and as we increase the batch size, the graph takes more and more time which is expected. For 64 batch size, we see that it takes an average of 160 seconds to generate the images.

5.3. Baseline Results

We trained the 'Baseline Model', using a learning rate of $2 \cdot 10^{-4}$ and a batch size of 128, over 20 epochs using Adam optimizer with a scheduler of 500-time steps. The baseline obtained a final validation loss of 0.04621237 and an FID score of 35. The model has 15915779 parameters and a file size of 60.77 MB. It takes the model 5.801 gigaflops for input to pass through the model once (one-time step). Figure 2 is an inference speed graph. Below are sample images generated by the baseline model.



Qualitatively, the images generated by the diffusion model are fuzzy and difficult to distinguish, especially the third image. However, it has learned how to differentiate the foreground and background when generating images. And learned some of the basic features of certain classes such as a bird and horses. As expected we get a linear curve as we increase the batch size. However we see that the inference time of this baseline model is much quicker than that of the teacher model taking an average time of almost 10s when batch size is 1 and 90s when batch size is 64.

5.4. Experiment 1 Results

We used the same hyperparameters as the baseline model to train our 'Student Model' (used a learning rate of $2 \cdot 10^{-4}$, batch size of 128, over 20 epochs using Adam optimizer). The student model 15915779 parameters and has a file size of 60.77 MB. The model performs 5.801 gigaflops of calculation. Figure 3 shows the average inference time (generating an image) against the number of batches. The model achieved a validation loss of 0.04121237 and a FID score of 31.86.

Below are sample images generated by the baseline model:

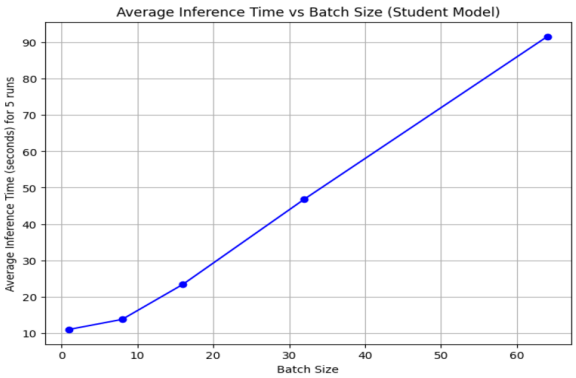


Figure 3

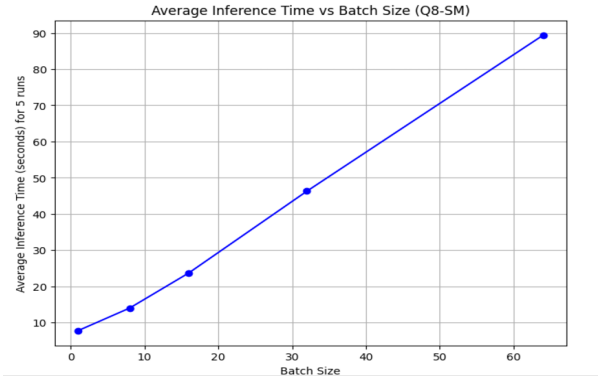


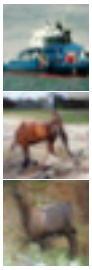
Figure 4

Qualitatively, the image generated by the model are much more clearer than the baseline results as the subjects are recognizable with some distortion. And like the baseline results, the background setting is also clear. Again, the graph is an increasing curve as expected and the average time to generate the image when batch size is 1 is close to 10s and when its 64, the time is close to 90s.

5.5. Experiment 2 Results

After applying 8-bit linear mapping quantized on the layers of the trained student model from experiment 1,our Q8-SM model achieved a validation loss of 0.04334, and a FID score of 33.08. The new model has 12760451 parameters, but a file size of 27.38mb. The model performs 5.801 giga flops of calculation. Figure 4 shows the average inference time (generating a image) against the number of batches.

Below are sample images generated by the baseline model:



From the graph we see that the average inferencing time when the batch size is 1 is a little less than 10s (8s approx) and when its 64, the average time is 90s approx. The inferencing time between the Student Model and 8Q-SM is not significantly different, by very less margin the graph is different indicating quantizing to 8-bit does not drastically impact the inference time.

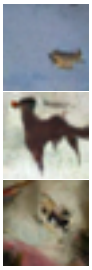
From the images, we can see the subject clearly but with distortion and fuzziness.

5.6. Experiment 3 Results

After applying 4-bit linear mapping quantized on the layers of the trained student model from experiment 1,our Q4-SM model achieved a validation loss of *0.47368*, and a FID score of 36.232. The new model has 12760451 parameters, but a file size of 25.88mb. The model performs 5.801 giga flops of calculation. Figure5 shows the average inference time (generating a image) against the number of batches.

Below are sample images generated by the baseline model:

From the graph we can see that with the Q4-SM model, the inference time decrease visibly from 10s (to approx 6s as seen by us during our runs) when batch size is 1. When batch size is 64, we see that it decreases from 90s to approx 86s. This indicates that 4-bit quantization not only makes the model lighter but somewhat decreases the inference time too.



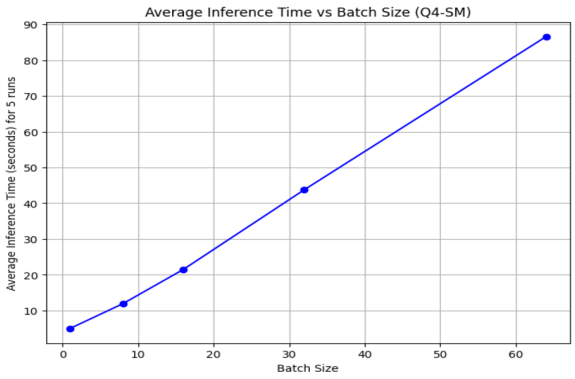
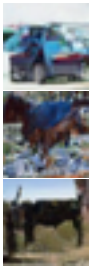


Figure 5

The generated images are difficult to distinguish. It has ability create some objects using simple shapes and colors.

5.7. Experiment 4 Results

After performing the same experiment with the same hyperparameters as the student model, it had 15915779 parameters and a file size of 60.77 MB. The model performs 5.801 gigaflops of calculation. Figure 6 shows the average inference time (generating an image) against the number of batches. The model achieved a validation loss of 0.04334 and a FID score of 32.95. Below are sample images generated by the model:



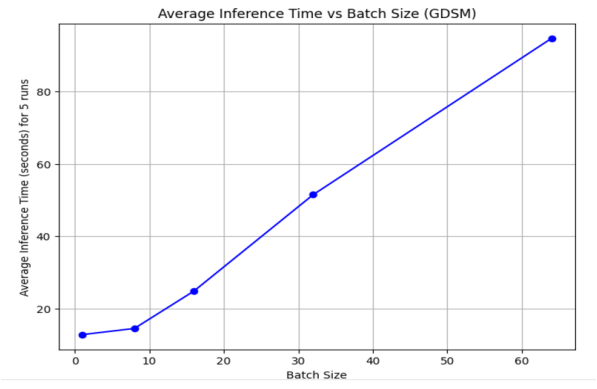
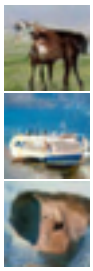


Figure 6

5.8. Experiment 5 Results

After applying 8-bit linear mapping quantized on the layers of the trained student model from experiment 4, the model achieved a validation loss of 0.03764, and a FID score of 33.22. The new model has 12760451 parameters with a file size of 27.38 MB. The model performs 5.801 gigaflops of calculation. Figure 7 shows the average inference time (generating a image) against the number of batches.

Below are sample images generated by the baseline model:



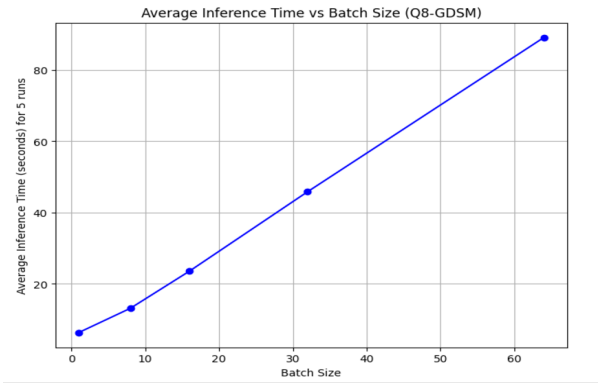


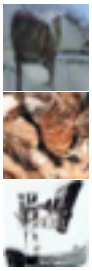
Figure 7

The above figure shows similar results to Figure 7 from experiment 4, we do not see a drastic change between the two. We also see that this variant of knowledge distillation performs more or less the same as the Student Model indicating that we need not make the model unnecessarily complex.

5.9. Experiment 6 Results

After applying 4-bit linear mapping quantized on the layers of the trained student model from experiment 4, the model achieved a validation loss of 0.041988, and a FID score of 35.64. The new model has 12760451 parameters with a file size of 25.88 MB. The model performs 5.801 giga flops of calculation. The Figure shows the average inference time (generating a image) against the number of batches.

Below are sample images generated by the baseline model:



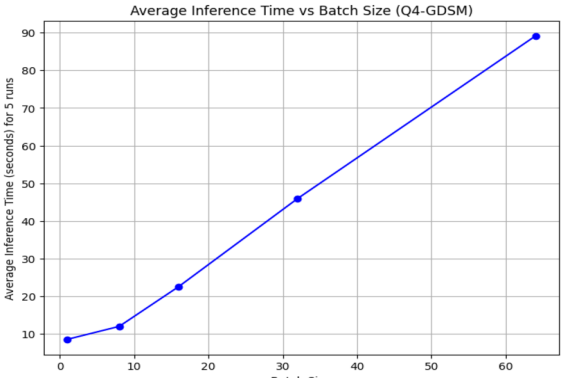


Figure 8

From the graph we see that Q4-GDSM Model has an average inferencing time of almost 10s when batch size is 1 and 90s when batch size is 64, indicating that the model performs more or less same as the student model in terms of the inference time.

5.10. Interpreting Results

Looking at the results, in terms of image generation quality. It is the case none of the created models has reached the teacher's FID score of 16.78. However, this was expected, reducing the model architecture can reduce the model's expressive power and learning abilities. Reducing the number of time steps makes it more difficult for the model to reconstruct images as it has less time to create and learn the process. However, there is evidence in the results of the experiments that show that the smaller model can learn a more sophisticated distribution from a larger model while retaining the benefits of having a smaller size. When comparing the results, from experiment 1 to the baseline model, there could be some quality improvements if knowledge distillation is used. The distilled model had nearly 3.1 points of improvement over the baseline when incorporating the student and teacher features in the training loss. Interestingly enough, having followed a gradual distillation process in experiment 4, performed just as well or slightly worse than the model from experiment 1 (as the FID score is slightly worse than experiment 1). In the scope of this report, we could conclude that the gradual distillation process worked no better than the initial proposed distillation process. One possible explanation is that the features learned by the intermediate model were not better or as good as what the teachers have learned, and because of this, it could have "misguided" the student model during the training process, which could explain the decrease in performance.

Looking at figure 9 and figure 10 in the context of memory and inference time produces interesting results. In this case, the teacher model was the largest, while the quantized models from experiments 2,3, 5, and 6 had the least amount of memory with the 8-bit quantized models being 27.38 MB and the 4-bit quantization having a model size of 25.88 MB (while the teacher had 143.2 MB, and baseline and other experiments had 60.77 MB). It seems that reducing the model architecture has significantly reduced the file size by a significant amount and the number of parameters by more than 55% for all models from the 35717891 in the teacher. However, despite the large decrease in memory and architecture size, the number of flops (as shown in the figure 11) for a single input to pass through the model for a one-time step is relatively the same for all the models (roughly a 5% decrease compared to the teacher). Despite, having similar flops, the referencing times for experimental models are faster than the teacher because all the experimental models have have maximum scheduler length of 500 vs 1000 in the teacher.

The teacher and the experimental results show issues with the size of the architecture and image generation quality. Running a large model with many more timesteps allows for images better-looking images. And decreasing the model size as done in the experiments, allows for faster and lighter-weight models at the expense of image quality. Interestingly, quantizing the model using 8-bit quantization has significantly reduced the model size with a minimal decrease in image quality. But 4-bit quantization was able to reduce the model size by a small amount more than 8-bit quantization but had affected larger negative effect on the image quality.

6. Conclusion

From the experiments, there is a trade-off between image quality, and memory and inference times. Larger models allow for better image generation but at the expense of time and memory. On the other hand, decreasing the model architecture or quantization can reduce the model memory consumption and generate images faster but at a cost of lower quality. However, knowledge distillation can help smaller models create better images by having a teacher model help guide them to learn the "good features" while maintaining the benefits of having a smaller model.

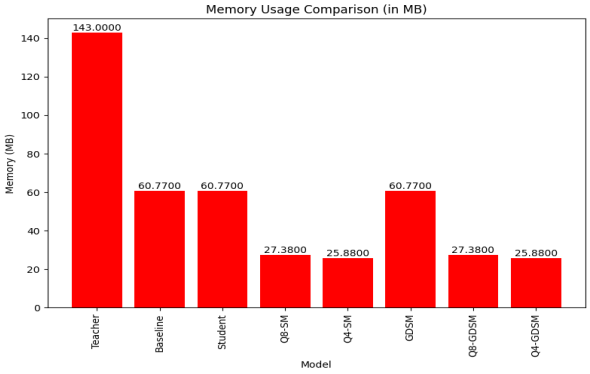


Figure 9

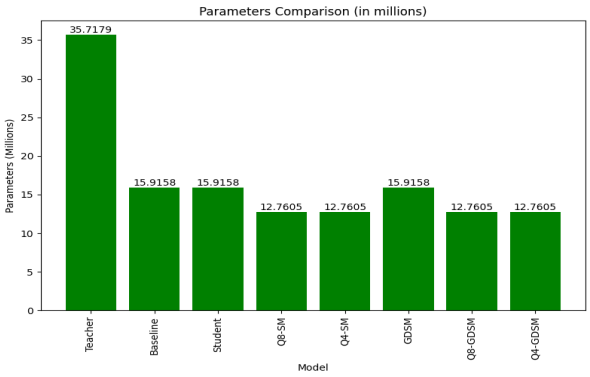


Figure 10

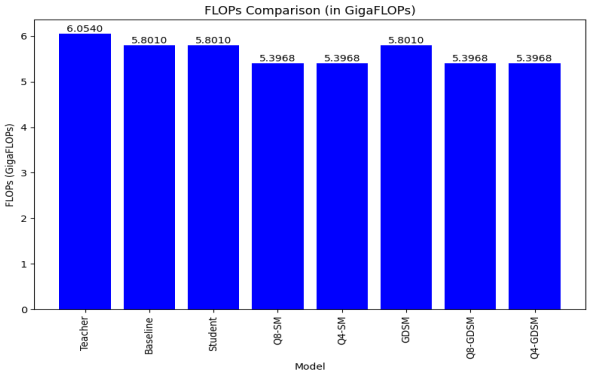


Figure 11

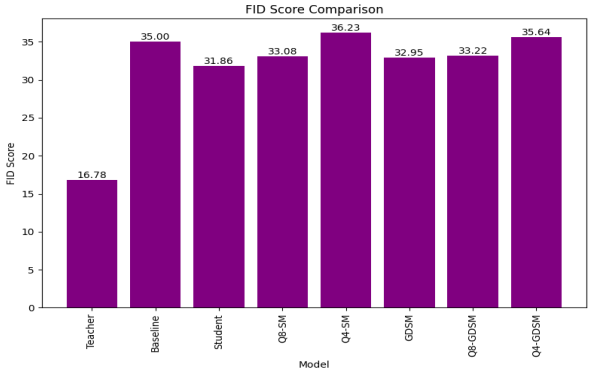


Figure 12

References

[1] Yunhui Guo. A survey on methods and theories of quantized neural networks. *arXiv preprint arXiv:1808.04752*, 2018.

[2] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[3] Tao Huang, Yuan Zhang, Mingkai Zheng, Shan You, Fei Wang, Chen Qian, and Chang Xu. Knowledge diffusion for distillation. In *Advances in Neural Information Processing Systems*, pages 65299–65316. Curran Associates, Inc., 2023.

[4] Weijian Luo. A comprehensive survey on knowledge distillation of diffusion models. *arXiv preprint arXiv:2304.04262*, 2023.

[5] Yuzhang Shang, Zhihang Yuan, Bin Xie, Bingzhe Wu, and Yan Yan. Post-training quantization on diffusion models, 2023.

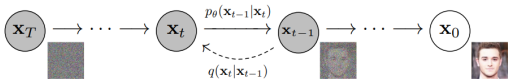


Figure 13. Directed Graphical Model for Diffuser Model by Ho et al. [2]

CIFAR-10 32 × 32 (100 DDIM steps)					
Method	**#Params ↓**	**MACs ↓**	**FID ↓**	**SSIM ↑**	**Train Steps ↓**
Pretrained	35.7M	6.1G	4.19	1.000	800K
Scratch Training	19.8M	3.4G	9.88	0.887	100K
Scratch Training			5.68	0.905	500K
Scratch Training			5.39	0.905	800K
Random Pruning			5.62	0.926	100K
Magnitude Pruning			5.48	0.929	100K
Taylor Pruning			5.56	0.928	100K
Ours ($\mathcal{T} = 0.00$)	19.8M	3.4G	5.49	0.932	100K
Ours ($\mathcal{T} = 0.02$)			5.44	0.931	100K
Ours ($\mathcal{T} = 0.05$)			**5.29**	**0.932**	100K

Figure 14. Diffusion Pruning on CIFAR10 Dataset evaluated on number of parameters, number of Multiply-Add Accumulations, FID Score, SSIM index and Training Steps