Text-to-3D Generative AI on Mobile Devices: Measurements and Optimizations

Xuechen Zhang*
University of California, Riverside
Riverside, CA, USA
xzhan384@ucr.edu

Zheng Li*, Samet Oymak, Jiasi Chen
University of Michigan
Ann Arbor, MI, USA
{jimmyli,oymak,jiasi}@umich.edu

ABSTRACT

Emerging generative models can create 3D objects from text prompts. However, deploying these models on mobile devices is challenging due to resource constraints and user demand for real-time performance. We take a first step towards understanding the bottlenecks by performing a measurement study of three recent text-to-3D generative models (Point-E, Shap-E, and CLIP-Mesh) in terms of their runtime GPU memory usage, latency, and synthesis quality. We investigate the effectiveness of quantization and distillation techniques to overcome these challenges by speeding up inference execution, potentially at the expense of quality. We find that the Shap-E model is promising for mobile deployment, but requires further optimization in its bottleneck diffusion step for real-time performance, as well as reduced memory usage and load times. Further work is needed on custom optimizations for generative text-to-3D models, including targeting specific metrics at each computation stage, efficient representations of 3D objects, and adaptive network and system support for resource-hungry models.

CCS CONCEPTS

• **Computing methodologies** → **Artificial intelligence**; • **Human-centered computing** → **Ubiquitous and mobile computing**.

KEYWORDS

Text-to-3D, generative AI, mobile devices, latency, GPU memory

ACM Reference Format:
Xuechen Zhang* and Zheng Li*, Samet Oymak, Jiasi Chen. 2023. Text-to-3D Generative AI on Mobile Devices: Measurements and Optimizations. In *Workshop on Emerging Multimedia Systems (EMS '23), September 10, 2023, New York, NY, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3609395.3610594

1 INTRODUCTION

Recent advancements in AI have revolutionized various domains, including computer vision, natural language processing, and generative AI. A notable recent development arising from the intersection of these fields is the emergence of text-to-3D generative models. Different from popular image-to-3D generative models such as NeRF [15], text-to-3D models [10, 11, 16, 19, 21] have the remarkable capability to create novel 3D objects from arbitrary text descriptions. An example is shown in Figure 1. When combined with next-generation AR/VR mobile devices that allow users to view and interact with the generated 3D objects from different viewpoints [14], these generative AI models have huge potential in education, gaming, product design, etc.

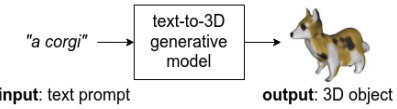


Figure 1: Example of a text-to-3D generative model.

However, several challenges impede the deployment of generative AI on resource-constrained mobile devices. Mobile devices typically have less memory and compute capabilities than the powerful servers that generative AI models are typically trained and tested on. Some generative models could have extremely large memory requirements (*e.g.*, Imagen [25] used in DreamFusion [21] contains 4.6 billion parameters), making them incompatible with mobile devices. These models typically also have high latency (*e.g.*, 12 hours for DreamFusion to generate a single 3D object on a NVIDIA V100 GPU) or low 3D object synthesis quality, which can hurt user experience. Hence, it is important to investigate the influence of these factors in order to develop effective strategies to accelerate the deployment of text-to-3D models on mobile devices.

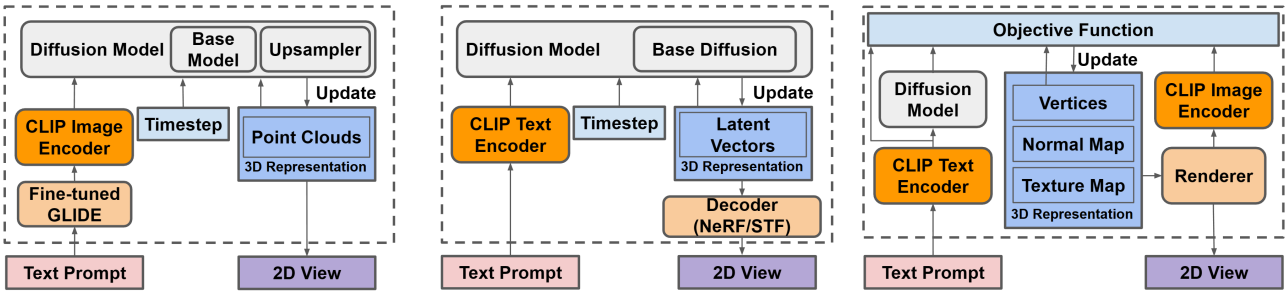
Towards this goal, in this paper we perform a measurement study of three recent text-to-3D generative models with potential for mobile deployment: Point-E [19], Shap-E [11] and CLIP-Mesh [16], all released in 2022-23. We selected these models because they are geared towards speed. Our measurements encompass three primary aspects: hardware compatibility, 3D object synthesis quality, and inference latency. Hardware compatibility is evaluated by tracking the runtime GPU memory usage of these models, and synthesis quality assessment involves measuring how well the generated 3D objects correspond to their original text descriptions [20]. We also investigate the effectiveness of standard quantization and distillation techniques on synthesis quality and speed of one of the models (Point-E). Overall, this work makes the following contributions:

- We measure the memory usage, latency, and synthesis quality of several recent text-to-3D generative models on an edge platform (NVIDIA Jetson Orin) and on a less powerful server GPU (NVIDIA T4 GPU) to understand their performance tradeoffs and feasibility for deployment on mobile devices. (§3)
- We assess the impact of distillation and quantization techniques to speed up deployment of Point-E on the Jetson, and find that both techniques significantly sacrifice synthesis quality for

*Equal contributors.

EMS '23, September 10, 2023, New York, NY, USA

ACM ISBN 979-8-4007-0303-4/23/09…$15.00
https://doi.org/10.1145/3609395.3610594



(a) Point-E (image-based configuration) (b) Shape-E (text-based configuration) (c) CLIP-Mesh

Figure 2: Architectures of the generative text-to-3D models studied in this paper (Point-E, Shap-E and CLIP-Mesh).

latency. Thus custom approaches are needed for the diffusion and transformer architectures of generative models. (§4)

- We analyze our findings and discuss future research directions. Overall, we find that Shap-E is promising in terms of synthesis quality and latency, but targeted optimizations at different stages of its processing are needed (*e.g.,* reduced latency of its diffusion step, reduced memory of its post-diffusion step) to meet end-to-end performance goals. (§5)

2 BACKGROUND AND RELATED WORK

The 3D representations used by the text-to-3D generative models and the architecture of these models will influence their performance. This section reviews these concepts.

2.1 3D Object Representations

Explicit representations: ***Point clouds*** represent a 3D object as a set of 3D points in space, and can also store other features (*e.g.,* color) at each point. The resolution of a point cloud is typically measured by the number of points it contains. ***3D meshes*** use vertices, edges and faces to represent the geometric structure of a 3D object. Feature vectors can also be associated with the vertices. The density of vertices, edges, and faces influences the smoothness of the mesh. Higher density results in more detailed 3D objects.

Implicit representations: ***Neural Radiance Fields (NeRF)*** [15] represent 3D objects as a continuous volumetric field using a multi-layer perceptron (MLP) [1]. To render a 2D view of a 3D object, for each pixel in the 2D view, NeRF emits a ray from the viewpoint along the viewing direction to pass through that pixel. NeRF subsequently queries the features (volume density and RGB color) of the 3D points at different distances along the rays using the MLP. Although NeRF can produce high quality 3D objects, it is usually inefficient due to its high latency and large size [17], which can inhibit mobile deployment. ***Signed Distance Fields (SDF)*** represent 3D objects by assigning a signed distance value to each point in 3D space. The sign of the value indicates whether the point is inside or outside the 3D object and the magnitude represents its distance to the nearest surface.

Among these 3D representations, explicit representations usually have low rendering latency, while implicit representations usually have high latency and memory usage due to intense computation. Although implicit representations are usually slower, they can produce more detailed and thus higher quality 3D objects because arbitrary points on the object can be rendered, rather than only the discrete locations represented by point clouds or meshes.

2.2 Text-to-3D Generative Models

Many current text-to-3D generative models rely on diffusion models as their foundation. Diffusion models learn to generate high-quality samples by starting from random noise and iteratively de-noising them to produce the final result. Text-to-3D generative models can be classified into two types based on how they utilize diffusion models. To generate a single 3D object, the first type (*e.g.,* Point-E [19], Shap-E [11], CLIP-Mesh [16]), involve a single run of a diffusion process, often acting on a representation of the 3D object. The second type (*e.g.,* DreamFusion [21], DreamFields [10]) employs multiple runs of a diffusion process, acting on a representation of the 2D rendered images of the 3D object. Since running a diffusion model is time-consuming, the first approach typically has lower latency (*e.g.,* the fastest Point-E model takes 16 seconds compared to Dreamfusion's 12 hours on an Nvidia V100 GPU; see Table 1 for details). As the models in the first category generally demonstrate lower latency, they are good candidates for mobile deployment, and we therefore focus on them in the rest of this paper.

Point-E [19] (released Dec. 2022) employs a two-step approach to generate a 3D model, which is represented as a point cloud. As shown in Figure 2a, it first transforms text into the corresponding embedding, and then generates a 3D object based on this embedding using a transformer-based diffusion model. In the first step, to transform the text into its corresponding embedding, there are two configurations that can be used, *text-only* and *image-based*. The text-only configuration directly feeds the text into a pretrained CLIP model [23], while the image-based configuration (shown in Figure 2a) first processes the text using a fine-tuned GLIDE model [18] to obtain its corresponding 2D image representation, which is then fed into a CLIP model to generate the final embedding. In the second step, the embedding, timestep, and the noised point clouds are encoded and fed as input to the diffusion model. The corresponding output is used in turn to update and denoise the point cloud. To enhance the quality of the generated point clouds, Point-E concatenates two diffusion models together. The first Base Model generates a point cloud with 1K points and the second Upsampler increases the resolution to 4K points, improving the overall object quality.

Shap-E [11] (released May 2023) has a similar transformer-based diffusion model architecture as Point-E (see Figure 2b). Shap-E also

has two configurations: *text-only* and *image-based*. Unlike Point-E, which employs an explicit 3D point cloud representation, Shap-E uses latent vectors to represent its 3D objects, requiring an extra volume rendering/decoding step. The diffusion model of Shap-E operates on the noised latent vectors to produce denoised latent vectors for the corresponding 3D models. The latent vectors are linearly projected to generate the parameters of an MLP responsible for Decoding. This MLP takes as input the coordinates of points in 3D space and outputs the corresponding feature vectors, such as RGB values, volume density, etc. To render the 3D model, Shap-E uses volume rendering with two possible configurations: NeRF-based [15] or STF-based (SDF with texture fields [5]).

CLIP-Mesh [16] (released March 2022) is a text-to-3D generative model that uses a mesh, a normal map and a texture map to represent 3D objects. The generation process (see Figure 2c) involves optimizing these three components based on an objective function that consists of three key terms: text-image similarity, image-image similarity, and a Laplacian regularizer. A diffusion model is run once to generate an image embedding to compute the image-image similarity. Text-image similarity and image-image similarity measure the relevance between the generated 3D object to the input text prompt. The Laplacian regularizer [8] is used to help maintain the geometry of the 3D model.

3 MEASUREMENTS

To assess the feasibility of deploying text-to-3D generative models on mobile devices, the models' ability to run on different edge hardware and their synthesis quality should be considered. The memory usage of a model determines its compatibility with specific devices, while its latency and synthesis greatly influence the overall user's experience. Therefore, we examine the generative models across three key dimensions: GPU memory usage (§3.2), latency (§3.3) and 3D object synthesis quality (§3.4), to explore their tradeoffs.

3.1 Experiment Setup

We conducted experiments on a more powerful NVIDIA T4 GPU with 4 vCPUs, and a weaker NVIDIA Jetson AGX Orin with 64 GB of memory. We chose the T4 as an approximation of limited mobile capacity, as it is less powerful than server GPUs like A100 and V100 where generative AI models are typically evaluated. To further assess mobile performance, we also ran the models on the Jetson AGX Orin. Our initial experiments were conducted on the more powerful T4, where it is easier to set up and run experiments, in order to identify hardware-independent bottlenecks such as memory usage. Our later experiments with the more promising text-to-3D models for mobile were conducted on the Jetson.

GPU memory. We measured the models' memory usage on a T4 GPU using the NVIDIA System Management Interface tool (`nvidia-smi`) that monitors the GPU status. The memory usage measured by `nvidia-smi` includes the overhead from PyTorch's caching allocator and other runtime factors, as opposed to only measuring the size of the model's parameter and architecture, so the memory usage we measured will be closer to the GPU memory usage in real application scenarios. The measurements were taken with batch size 1, with image size 64×64 for Shap-E. The representative memory traces of the text-only configurations were measured using the text input "a corgi", while the image-based configurations were measured using a corgi image as input, following prior examples [11, 19]. The latter does not include the GLIDE model.

Latency. Total latency is the time from when a text prompt is input to when a 3D object is output). We performed total latency measurements on the T4 and Jetson, and reported V100 results for comparison from [10, 11, 16, 19, 21]. All T4 results (except CLIP-Mesh) are measured with the full 5000-caption COCO test set [3]. Models deployed on Jetson and CLIP-Mesh on the T4 are measured using 100 random COCO test samples. For the latency breakdown measurements, we evaluated over 100 random samples from the COCO test set. We utilized the pre-trained models provided by the original papers [11, 19] wherever possible, except for Point-E (300M, text only) and GLIDE. The original large GLIDE model is not public, so we use a smaller pre-trained version (300 million parameters). Otherwise, we use the proper model architecture with randomly initialized weights, which does not change the latency. To measure loading latency, we first build the model architecture and then subsequently load a `.pt` file that contains pretrained weights. Results are averaged over 100 runs.

Synthesis quality. We adopt the CLIP R-Precision metric (ViT-B/32) [20] for evaluating text-to-3D synthesis quality, which has been found to have high correlation with human judgement. The CLIP R-Precision measures the compatibility of image-text pairs, which can also be thought of as the semantic similarity between the image and the caption. Specifically, it uses the CLIP model [23] to calculate the top-R retrieval accuracy when retrieving the matching text from 100 text candidates, using the generated image as a query. A higher score is better.

Model configurations: In order to identify potential inefficiencies within the models and enable future optimization efforts to focus on specific inefficient stages, all models are divided into several parts for measurement. For notational simplicity, we include CLIP when measuring Point-E's Base Model and Upsampler, and Shap-E's Base Diffusion. The breakdown of the models and the configurations are described below.

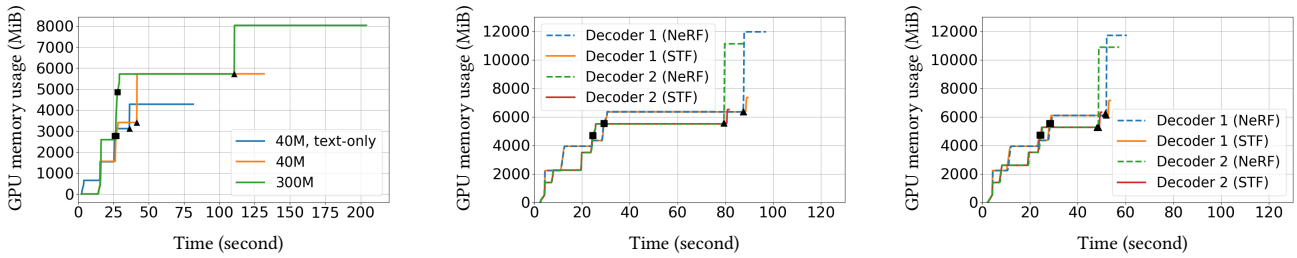
- **Point-E** has a choice of how many parameters in the Base Model (40, 300, or 1000 million) as well as whether *text-only* or *image-based* configuration is used (see §2.2). Unless explicitly noted, image-based is used.
- **Shap-E** similarly has *text-only* or *image-based* configurations. There are also two decoder options with slightly different underlying models, which we call *Decoder 1* and *Decoder 2*,[1] each of which has two representation options: NeRF or STF.

3.2 GPU Memory

Mobile devices are known to have limited GPU memory size, which presents a significant constraint when deploying generative models with large model sizes. Due to this limitation, it is essential to consider GPU memory usage when evaluating the models' feasibility of mobile deployment.

Point-E. We measured three different Point-E models with suitable model size for mobile deployment. The parameter count is shown in Table 5 in the Appendix, and traces of runtime memory

[1] "Transmitter" and "decoder", respectively, in the Shap-E code.



(a) Point-E (b) Shap-E (300M) (c) Shap-E (300M, text-only)

Figure 3: GPU memory usage. The time before the square points is for model loading and building. For Point-E (Shap-E), square points denote starting Base Model (diffusion process) execution and triangle points denote starting the Upsampler (Decoder). Generally, the model loading and post-diffusion steps consume the most GPU memory and need further optimization.

usage are shown in Figure 3a. Larger parameter count leads to a larger model size and thus higher memory usage, as expected. Also, image-based models have higher memory usage than the text-only model, even with similar model parameter counts, but this turns out to due to the influence of a Point-E hyperparameter (the guidance scale), which has different defaults for image-based vs. text-only configurations. Finally, although the Upsampler only has 40M parameters, it needs three times as much memory as the Base Model, possibly because it has to hold a high-resolution point cloud in memory. These results suggest that a potential area of further optimization for Point-E is the Upsampler, as it consumes a significant fraction of the memory and time, as does the model's initial load.

Shap-E. We tested eight different Shap-E configurations (see Figures 3b, 3c and Table 6). From the results, it can be seen that although NeRF rendering tends to produce slightly higher quality 3D objects than STF rendering, it requires substantially more memory, which is likely due to its more complex representation of the 3D object [11]. Further, we observe that despite the fact that the Decoder 1 has twice as many parameters as Decoder 2, its GPU memory usage is only slightly greater. Overall, these results suggest that NeRF representations require significant optimization in terms of memory usage, as does the initial model loading step.

Summary. Many models or configurations are quite large (8-12 GB) and optimizing their memory usage on mobile devices is needed. Popular mobile devices such as the Samsung S10 and Pixel 4a have limited available memory, with a significant portion already allocated for the baseline OS and background processes. For example, The Samsung S10 has 7.7 GB memory, but it already consumes 3.7 GB for normal usage. Regarding what parts of the models require optimizations, our measurements of Point-E and Shap-E show that the diffusion step is not the bottleneck for memory usage, but rather the post-processing steps after diffusion (Upsampler or Decoder, respectively).

3.3 Latency Measurements

When considering text-to-3D generative models for interactive usage, low latency is a crucial factor for smooth user experience. Thus, measuring model latency is important when evaluating the feasibility of a model for mobile deployment. Our measurements (total latency in Table 1, breakdown of latency in Figure 4, and loading time in Table 2), indicate that Shap-E and Point-E have

Model	V100	T4	Jetson
DreamFields [10]	~ 200 hr	/	/
DreamFusion [21]	~ 12 hr	/	/
CLIP-Mesh	~ 17 min	~ 52 min	/
Point-E (40M, text-only)	16 sec	56 sec	3.1 min
Point-E (40M)	1.0 min	3.3 min	9.7 min
Point-E (300M, text-only)	25 sec	1.3 min	5.2 min
Point-E (300M)	1.2 min	/	/
Point-E (1B)	1.5 min	/	/
Shap-E (300M, text-only, decoder 1, NeRF)	13 sec	34 sec	2.5 min

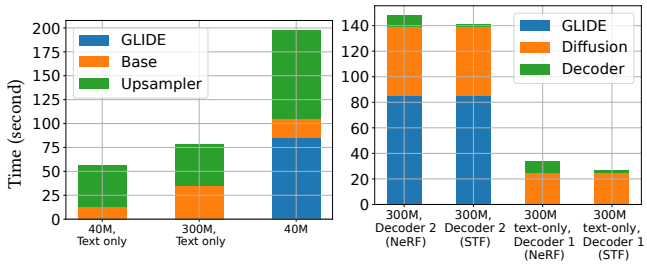
Table 1: Average total latency of generating a single 3D object, for different text-to-3D generative models and devices. Shap-E generally has the lowest latency.

lower latency than CLIP-Mesh, making them more suitable for interactive deployments.

Point-E. From Table 1, it is clear that the Point-E model variants with more parameters have higher latency. For example, Point-E (40M, text-only) took 3.1 minutes to generate one 3D object on the Jetson, but Point-E (300M, text-only) took 5.2 minutes. Also, given the same parameter count, the image-based configuration has significantly higher latency than the text-only configuration (for example, about 3x for Point-E (40M) on the Jetson). As shown in the latency breakdown in Figure 4a, this difference caused by the inclusion of the GLIDE model in the image-based configuration and by the extra latency of the Upsampler (which in turn is due to the associated guidance scale hyperparameter, similar to §3.2).

Figure 4a also shows that the higher output dimension of the Upsampler causes it to have a higher latency relative to the Base Model, despite the similarity in the number of model parameters (also seen in the Fig. 3a trace). The ratio of the Upsampler to the Base Model's latency is larger when the model is smaller (green vs. orange bars in the figure). Finally, as seen in the load time measurements of Table 2, although model variants with higher parameter counts have slightly higher latency, different parameter counts do not significantly influence the loading time of the Base Model, suggesting that most of the loading time is a fixed cost dependent on the model architecture.

Shap-E. As seen in Table 1, Shap-E has the lowest latency among the three candidate models for mobile deployment (Point-E, Shap-E and CLIP-Mesh). As shown in the latency breakdown in Figure 4b, the diffusion process is a more significant source of latency than



(a) Point-E (b) Shap-E

Figure 4: Latency breakdown of the Point-E and Shap-E models measured on a T4 GPU, excluding loading time. Shap-E generally has lower latency than Point-E. Image-based configurations take significantly longer due to the GLIDE model.

Model Component	Load Latency (s)
GLIDE	164.95
Point-E (Base Model, 40M, text-only)	12.95
Point-E (Base Model, 40M)	13.37
Point-E (Base Model, 300M)	14.14
Point-E (Upsampler)	13.24
Shap-E (Base Diffusion, 300M, text-only)	17.38
Shap-E (Base Diffusion, 300M)	17.95
Shap-E (Decoder 1)	8.51
Shap-E (Decoder 2)	5.22

Table 2: Mean latency of model loading measured on T4 GPU. Point-E and Shap-E have similar overall loading time. GLIDE's load time (for image-based configurations) is large.

decoding. For decoding, using the STF representation is slightly faster than NeRF, but not by much. Similar to Point-E, for Shap-E, the text-only model has much faster diffusion process than the image-based model due to the inclusion of the GLIDE model.

In terms of loading latency in Table 2, Shap-E's loading latency is generally similar to Point-E's overall. While Shap-E's Base Diffusion is slower to load than Point-E's Base Model, the post-diffusion step of Decoder is faster than Point-E's Upsampler. Loading Decoder 1 is slower than loading Decoder 2. This loading only needs to happen once, when the text-to-3D application is first launched.

Summary. Shap-E is more promising than Point-E for mobile deployment, taking less time for execution (at the expense of consuming more GPU memory). The latency is not necessarily proportional to the number of model parameters, as shown by Point-E's Upsampler's having a higher latency due to its high output dimension. Finally, image-based configurations are very slow due to the high load time and execution latency of the GLIDE model.

3.4 3D Object Synthesis Quality

The preceding metrics (memory usage and latency) may trade off with the synthesis quality of the generated 3D objects. We obtain the CLIP-R-Precision of all three models [11, 16, 19] evaluated on the COCO dataset and plot them alongside our latency measurements in Figure 5, for different model configurations and devices. Data points towards the upper left corner of the plot are desirable for lower latency and high 3D object synthesis quality. Several observations can be made. Although CLIP-Mesh, DreamFusion, and DreamFields have the highest latencies, they can produce results with better

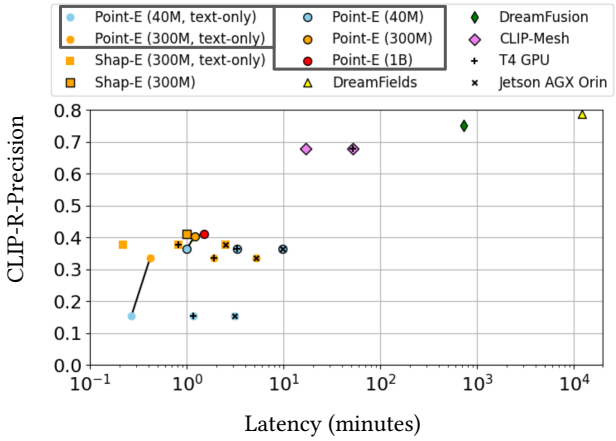


Figure 5: Quality-latency tradeoffs of text-to-3D generative models. Absence of a "+" or "x" means measurements from a V100. Latency measurements are from Table 1. Shap-E and CLIP-Mesh exhibit the best tradeoff on the Pareto frontier.

quality. Point-E and Shap-E have similar synthesis quality, and image-based models achieve better synthesis quality than text-only ones. Moreover, more parameters leads to better model quality (dots connected by solid lines). Finally, running the models on T4 or Jetson increases the latency for the same synthesis quality. Overall, Shap-E and CLIP-Mesh exhibit good tradeoffs between quality and time on the Pareto frontier.

4 MODEL OPTIMIZATION

To address the preceding latency challenges, we performed an initial investigation into general model speedup techniques, specifically distillation (§4.1) and quantization (§4.2). We focused on Point-E (40M, text-only) as it has relatively low latency and size based on our preceding measurements, and Shap-E was released very recently (May 2023). Since Shap-E has a similar diffusion architecture to Point-E, we believe that its performance would be similar. The evaluations are conducted on 100 random test samples from COCO.

4.1 Distillation

Model distillation trains a small lightweight model to mimic the behavior and knowledge of a larger complex model. However, standard distillation methods for classifiers [2, 9] cannot be directly applied to the unique structure of diffusion models, which serve as the basic architecture for most text-to-3D generative models. This is because diffusion models require temporal continuity across multiple steps for sampling. Later work has studied distillation for diffusion models, with a recent work [26] successfully distilling 2D generative diffusion models down to as few as 4 iterations while maintaining perceptual quality, but for 2D images only. Can similar techniques work for 3D object generation? Following [26], we distilled the behavior of original Point-E with N steps into a new model requiring only $N/2$ steps for each distillation training round. By repeating this distillation training procedure, we can potentially create faster models capable of generating high-quality outputs.

Our experiments (see Table 3) demonstrate that even with just one distillation training round (*i.e.,* distilled the teacher model with N steps into a new model with $N/2$ steps), the new model with 74

Iterations	Quality ↑	Inference Latency
150 (original)	15.4%	56 sec
74	12.4%	30 sec
36	6.8%	17 sec

Table 3: Distilled Point-E (40M, text-only) on T4 GPU.

Library	Layers	Quality ↑	Speed
Original	n/a	15.4%	×1
TensorRT	Linear	10.2%	×1.3
TensorRT	All	1.7%	×1.8
PyTorch (FBGEMM)	Linear	11%	×1.3

Table 4: Quantized Point-E (40M, text-only) on T4 GPU.

steps can accelerate the diffusion process by nearly 2x while maintaining over 80% of the original quality. However, further distillation tends to decrease the latency and leads to greater performance degradation, losing more than half of the original quality.

4.2 Quantization

Quantization reduces model size by representing model parameters with fewer bits. Applying standard quantization techniques to Point-E is nontrivial because of its unique transformer architecture and diffusion model. For example, existing quantization packages, such as PyTorch's Dynamic Quantization [22], don't support the multi-head attention layers used in transformer architectures such as Point-E. Diffusion models are also challenging to quantize due to the variable output distributions of the noise estimation networks across diffusion iterations, which are prone to degradation [13]. Recent works [13, 27] attempt to tackle these challenges, but their focus have been primarily on classic diffusion models with the UNet architecture for 2D images, rather than 3D objects.

We conducted experiments using integer quantization [28] on Point-E and present the results of post-training quantization to 8-bit integers (see Table 4). PyTorch supports multiple approaches to quantizing a deep learning model including TensorRT and FBGEMM (which we evaluated on the CPU as it doesn't support GPU). The quantization scheme in TensorRT is symmetric per-channel for weights, and symmetric per-tensor for activations, whereas FBGEMM is asymmetric. Compared to typical single-precision floating point (32-bit) models, 8-bit integer quantization enables 4× reduction in the model size. However, directly applying these techniques to Point-E led to a significant decrease in performance with commensurate speed improvement.

5 DISCUSSION AND OPPORTUNITIES

Which models are most promising for mobile? Among the recent generative models we evaluated, Shap-E exhibited the lowest latency while delivering reasonable synthesis quality. Additionally, the availability of multiple decoder options makes it more adaptable to different goals and device specifications, although the memory usage is still too high for off-the-shelf mobile devices such as smartphones, and mobile GPU programming is challenging. CLIP-Mesh is another viable alternative, although significant latency reductions are needed to be tenable on mobile devices.

The text-to-3D models generally have different configurations (*e.g.,* number of parameters, whether an intermediate image embedding is needed, output format). Which of these configuration options have the most impact on performance? Generally, larger model size somewhat increased latency or GPU memory, but not directly proportional to the parameter count, while also improving synthesis quality (in terms of CLIP-R-Precision). Therefore, when deploying these methods in practice, selecting larger models is advisable wherever feasible. A more significant configuration parameter is whether the intermediate image embedding is needed (text-only vs. image-based configurations). Although the image-based configurations generally had higher synthesis quality, there was also a noticeable increase in latency. Therefore text-based configurations are advisable, although further exploration is needed on the associated configuration-specific hyperparameters (such as guidance scale) that can also impact performance.

What are the current bottlenecks of mobile deployment? Our study shows that different stages of the generative 3D models have different bottlenecks. The initial model loading step generally consumes significant memory, as does the post-diffusion step. In general, the GPU memory usage is quite high (8-12 GB), compared to what is currently available (*e.g.,* 4 GB on a Hololens 2) so well-provisioned edge devices are needed to run these generative models, or remote help from a cloud/edge server is needed. The diffusion step, on the other hand, consumes a significant fraction of the latency. Overall, this suggests that a one-size-fits-all approach of minimizing latency or memory alone will not work, and targeted optimization with specific objectives are needed at each stage of the model, in order to meet end-to-end performance goals.

What are the unique challenges of 3D generative AI? The 3D representation and how it is generated greatly impact overall performance. Changing the 3D representation from implicit neural representations [15] can significantly reduce latency (as MobileNeRF [4] did to textured polygons, or Point-E did to point clouds). This points to the need for further 3D-specific optimizations to achieve further gains, such as viewport adaptation, intelligent configuration of the AI models when running locally, and dynamic adaptive 3D streaming when network support is needed. Related techniques have been studied in the context of 360-degree streaming or 2D object detection [6, 7, 12, 24], but the rapidly changing architectures and 3D representations in generative text-to-3D models present new challenges.

6 CONCLUSIONS

As research on generative AI and text-to-3D models specifically is advancing rapidly, this work provides a checkpoint on their resource utilization (memory, latency, and synthesis quality) on server and embedded system GPUs. Our measurements showed trade-offs between memory, latency, and synthesis quality, with Shap-E achieving the best balance compared to Point-E and CLIP-Mesh. Certain compute steps of Shap-E (model loading and diffusion) consumed significant resources (memory and time, respectively), motivating further innovations. Application of standard distillation and quantization techniques to the diffusion step improved speed at the expense of lower synthesis quality. Therefore, multi-objective, customized optimization of text-to-3D generative AI models is needed to realize their vast potential on mobile devices.

ACKNOWLEDGEMENTS

This work was supported in part by NSF CAREER 1942700 and CCF-2212426.

REFERENCES

[1] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. 2021. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *IEEE/CVF ICCV*.

[2] Defang Chen, Jian-Ping Mei, Hailin Zhang, Can Wang, Yan Feng, and Chun Chen. 2022. Knowledge distillation with the reused teacher classifier. In *IEEE/CVF CVPR*.

[3] Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. 2015. Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325* (2015).

[4] Zhiqin Chen, Thomas Funkhouser, Peter Hedman, and Andrea Tagliasacchi. 2023. Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. In *IEEE/CVF CVPR*.

[5] Jun Gao, Tianchang Shen, Zian Wang, Wenzheng Chen, Kangxue Yin, Daiqing Li, Or Litany, Zan Gojcic, and Sanja Fidler. 2022. Get3d: A generative model of high quality 3d textured shapes learned from images. *NeurIPS* (2022).

[6] Yu Guan, Chengyuan Zheng, Xinggong Zhang, Zongming Guo, and Junchen Jiang. 2019. Pano: Optimizing 360 video streaming with a better understanding of quality perception. In *ACM SIGCOMM*.

[7] Bo Han, Yu Liu, and Feng Qian. 2020. ViVo: Visibility-aware mobile volumetric video streaming. In *ACM MobiCom*.

[8] Jon Hasselgren, Jacob Munkberg, Jaakko Lehtinen, Miika Aittala, and Samuli Laine. 2021. Appearance-Driven Automatic 3D Model Simplification. In *Eurographics Symposium on Rendering*.

[9] Tao Huang, Shan You, Fei Wang, Chen Qian, and Chang Xu. 2022. Knowledge distillation from a stronger teacher. *arXiv preprint arXiv:2205.10536* (2022).

[10] Ajay Jain, Ben Mildenhall, Jonathan T Barron, Pieter Abbeel, and Ben Poole. 2022. Zero-shot text-guided object generation with dream fields. In *IEEE/CVF CVPR*.

[11] Heewoo Jun and Alex Nichol. 2023. Shap-e: Generating conditional 3d implicit functions. *arXiv preprint arXiv:2305.02463* (2023).

[12] Kyungjin Lee, Juheon Yi, and Youngki Lee. 2023. FarfetchFusion: Towards Fully Mobile Live 3D Telepresence Platform. In *ACM MobiCom*.

[13] Xiuyu Li, Long Lian, Yijiang Liu, Huanrui Yang, Zhen Dong, Daniel Kang, Shanghang Zhang, and Kurt Keutzer. 2023. Q-diffusion: Quantizing diffusion models. *arXiv preprint arXiv:2302.04304* (2023).

[14] MacRumors. 2023. Apple Now Seeking Generative AI Engineers for AR/VR Applications. https://www.macrumors.com/2023/06/02/apple-seeking-generative-ai-engineers-for-ar-vr/. (2023).

[15] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. 2021. Nerf: Representing scenes as neural radiance fields for view synthesis. *Commun. ACM* 65, 1 (2021), 99–106.

[16] Nasir Mohammad Khalid, Tianhao Xie, Eugene Belilovsky, and Tiberiu Popa. 2022. CLIP-Mesh: Generating textured meshes from text using pretrained image-text models. In *SIGGRAPH Asia*.

[17] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics* 41, 4 (2022), 1–15.

[18] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. 2021. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741* (2021).

[19] Alex Nichol, Heewoo Jun, Prafulla Dhariwal, Pamela Mishkin, and Mark Chen. 2022. Point-E: A System for Generating 3D Point Clouds from Complex Prompts. *arXiv preprint arXiv:2212.08751* (2022).

[20] Dong Huk Park, Samaneh Azadi, Xihui Liu, Trevor Darrell, and Anna Rohrbach. 2021. Benchmark for compositional text-to-image synthesis. In *NeurIPS*.

[21] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. 2022. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988* (2022).

[22] PyTorch. 2023. Dynamic Quantization. https://pytorch.org/tutorials/recipes/recipes/dynamic_quantization.html. (2023).

[23] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *ICML*.

[24] Xukan Ran, Haolianz Chen, Xiaodan Zhu, Zhenming Liu, and Jiasi Chen. 2018. Deepdecision: A mobile deep learning framework for edge video analytics. In *IEEE INFOCOM 2018-IEEE conference on computer communications*. IEEE, 1421–1429.

[25] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. 2022. Photorealistic text-to-image diffusion models with deep language understanding. *NeurIPS* (2022).

[26] Tim Salimans and Jonathan Ho. 2022. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512* (2022).

[27] Yuzhang Shang, Zhihang Yuan, Bin Xie, Bingzhe Wu, and Yan Yan. 2023. Post-training quantization on diffusion models. In *IEEE/CVF CVPR*.

[28] Hao Wu, Patrick Judd, Xiaojie Zhang, Mikhail Isaev, and Paulius Micikevicius. 2020. Integer quantization for deep learning inference: Principles and empirical evaluation. *arXiv preprint arXiv:2004.09602* (2020).

APPENDIX

A NUMBER OF MODEL PARAMETERS

Tables 5 and 6 show the number of parameters for each component of the Point-E and Shap-E models, respectively.

Point-E Component	Number of Parameters
Base Model (40M, text-only)	40,333,836
Base Model (40M)	40,466,956
Base Model (300M)	311,778,316
Base Model (1B)	1,244,311,564
Upsampler	40,470,540

Table 5: Number of model parameters for Point-E

Shap-E Component	Number of Parameters
GLIDE	~ 3.5 B
Diffusion Model (300M, text-only)	315,692,032
Diffusion Model (300M)	315,956,224
Decoder 1	443,771,357
Decoder 2	226,076,870

Table 6: Number of model parameters for Shap-E