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

Xuechen Zhang\textsuperscript{1*}, Zheng Li\textsuperscript{2*}, Samet Oymak\textsuperscript{2}, Jiasi Chen\textsuperscript{2}

\textsuperscript{1}University of California, Riverside
\textsuperscript{2}University of Michigan, Ann Arbor
\textsuperscript{*}co-first authors
Text-to-3D Generative AI

“"A pumpkin”" ➔ Text-to-3D Generative AI ➔ Output: 3D Model

Application Scenarios:
- Gaming
- Product Design

Input: Text Prompt
Text-to-3D Generative AI

Problems:
Not ready for mobile deployment due to resource constraints (memory, compute, energy, etc.)

E.g., DreamFusion takes 12 hours to generate a 3D object on a NVIDIA V100 GPU
Motivation

We want to deploy Text-to-3D generative AI on mobile devices while ensuring good user experience.

- Low Latency
- Low Memory Usage
- High 3D Object Synthesis Quality
Motivation

Low Latency
Low Memory Usage
High Synthesis Quality

Optimization

Measurements to identify bottlenecks
Background

Text - to - 3D

Natural Language Processing

Generative AI

Computer Vision
Background: 3D Representations

3D

Computer Vision
Background: 3D Representations

Explicit Representation
- Point Clouds
- 3D Meshes

Implicit Representation
- NeRF
- SDF
Background: Explicit Representations

Point Clouds

3D Meshes

Usually use discrete locations represented by points, edges etc.

Low Latency
Low Memory Usage
Low Synthesis Quality
Background: Implicit Representations

NeRF: Neural Radiance Fields

\[ F_\Theta : (x, d) \rightarrow (c, \sigma) \]

5D Input: Position + Direction
Output: Color + Density

High Latency due to Computation
High Memory Usage due to Computation
High Synthesis Quality

SDF: Signed Distance Field
Background: 3D Representations

<table>
<thead>
<tr>
<th>Explicit Representations</th>
<th>Implicit Representations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latency</td>
<td>↑</td>
</tr>
<tr>
<td>Memory Usage</td>
<td>↑</td>
</tr>
<tr>
<td>Synthesis Quality</td>
<td>↑</td>
</tr>
</tbody>
</table>
Background

to

Generative AI
Background: Diffusion Model

Reverse Diffusion Process

\[ p_\theta(x_{t-1} | x_t) \]

\[ q(x_t | x_{t-1}) \]

Forward Diffusion Process

Many steps of an expensive machine learning model (e.g. Unet, ViT) is needed to learn the reverse diffusion process.
Diffusion Model Overview

Point-E (Dec. 2022)

- Diffusion Model
- Base Model
- Upsampler
- CLIP Image Encoder
- Timestep
- Point Clouds
- 3D Representation
- Fine-tuned GLIDE
- Text Prompt
- 2D View

Shap-E (May 2023)

- Diffusion Model
- Base Diffusion
- Latent Vectors
- 3D Representation
- CLIP Text Encoder
- Timestep
- Decoder (NeRF/STF)
- Text Prompt
- 2D View
Measurements

What are the **bottlenecks** to deploy text-to-3D models on mobile devices?

What to measure?

**Optimization Goals:**
- Low **Latency**
- Low **Memory Usage**
- Good **Synthesis Quality**
Measurement Setup

Hardware:
NVIDIA T4 GPU (weak server GPU)
NVIDIA Jetson AGX Orin (mobile GPU)

Dataset:

![COCO Logo](image)
Measurement Setup: Model Configurations

For Point-E and Shap-E:

Parameter count for Diffusion:
- 40M
- 300M
- 1B

Conditioning options:
- Text-only
  - Text \(\rightarrow\) 3D
- Image-conditional (Default)
  - Text \(\rightarrow\) 2D \(\rightarrow\) 3D
Latency-Quality Tradeoff

Synthesis quality:
- Image-conditional > Text-only

Latency:
- Text-only < Image-conditional
Latency Breakdown

**Point-E**

<table>
<thead>
<tr>
<th>Latency (s)</th>
<th>GLIDE</th>
<th>Base</th>
<th>Upsampler</th>
</tr>
</thead>
<tbody>
<tr>
<td>40M, Text only</td>
<td><img src="image1" alt="Bar Graph" /></td>
<td><img src="image2" alt="Bar Graph" /></td>
<td><img src="image3" alt="Bar Graph" /></td>
</tr>
<tr>
<td>300M, Text only</td>
<td><img src="image4" alt="Bar Graph" /></td>
<td><img src="image5" alt="Bar Graph" /></td>
<td><img src="image6" alt="Bar Graph" /></td>
</tr>
<tr>
<td>Image-conditional</td>
<td><img src="image7" alt="Bar Graph" /></td>
<td><img src="image8" alt="Bar Graph" /></td>
<td><img src="image9" alt="Bar Graph" /></td>
</tr>
</tbody>
</table>

**Shap-E**

<table>
<thead>
<tr>
<th>Latency (s)</th>
<th>GLIDE</th>
<th>Diffusion</th>
<th>Decoder</th>
</tr>
</thead>
<tbody>
<tr>
<td>300M, Decoder 2 (NeRF)</td>
<td><img src="image10" alt="Bar Graph" /></td>
<td><img src="image11" alt="Bar Graph" /></td>
<td><img src="image12" alt="Bar Graph" /></td>
</tr>
<tr>
<td>300M, Decoder 2 (STF)</td>
<td><img src="image13" alt="Bar Graph" /></td>
<td><img src="image14" alt="Bar Graph" /></td>
<td><img src="image15" alt="Bar Graph" /></td>
</tr>
<tr>
<td>300M, text-only, Decoder 1 (NeRF)</td>
<td><img src="image16" alt="Bar Graph" /></td>
<td><img src="image17" alt="Bar Graph" /></td>
<td><img src="image18" alt="Bar Graph" /></td>
</tr>
<tr>
<td>300M, text-only, Decoder 1 (STF)</td>
<td><img src="image19" alt="Bar Graph" /></td>
<td><img src="image20" alt="Bar Graph" /></td>
<td><img src="image21" alt="Bar Graph" /></td>
</tr>
</tbody>
</table>

**Diffusion** is a latency bottleneck!
Implicit representation can save memory usage during generation.
Model Optimization

What to optimize?

Diffusion process!
Model Optimization

How to optimize?

- Distillation
- Quantization
- Neural Architecture Search, Pruning, etc.

Can be generalized for other diffusion based models
Model Optimization: Distillation

Speed up the model by reducing steps

$t = 1$

$z_{3/4} = f(z_1; \eta)$

$z_{1/2} = f(z_{3/4}; \eta)$

$z_{1/4} = f(z_{1/2}; \eta)$

$x = f(z_{1/4}; \eta)$

$t = 0$

Teacher

Student

$x = f(z_1; \theta)$
Model Optimization: Distillation

Speed up the model by reducing steps

\[ t = 1 \]
\[ z_{3/4} = f(z_1; \eta) \]
\[ z_{1/2} = f(z_{3/4}; \eta) \]
\[ z_{1/4} = f(z_{1/2}; \eta) \]
\[ x = f(z_{1/4}; \eta) \]

\[ t = 0 \]

Teacher

Student

Distillation

Distillation

Distillation

\[ x = f(z_1; \theta) \]
Model Optimization: Distillation

Speed up the model by reducing steps

Point-E results:

Synthesis quality severely degrades at lower latency.
Model Optimization: Quantization

Speed up the model and reduce memory usage by using lower precision parameters: 32 bit → 8 bit Quantization

Point-E results:

<table>
<thead>
<tr>
<th>Library</th>
<th>Layers</th>
<th>Quality ↑</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>n/a</td>
<td>15.4%</td>
<td>×1</td>
</tr>
<tr>
<td>TensorRT</td>
<td>Linear</td>
<td>10.2%</td>
<td>×1.3</td>
</tr>
<tr>
<td>TensorRT</td>
<td>All</td>
<td>1.7%</td>
<td>×1.8</td>
</tr>
<tr>
<td>PyTorch (FBGEMM)</td>
<td>Linear</td>
<td>11%</td>
<td>×1.3</td>
</tr>
</tbody>
</table>
Model Optimization: Quantization

**Speed up the model and reduce memory usage by using lower precision parameters:** 32 bit → 8 bit

**Quantization**

**Point-E results:**

<table>
<thead>
<tr>
<th>Library</th>
<th>Layers</th>
<th>Quality</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>n/a</td>
<td>15.4%</td>
<td>×1</td>
</tr>
<tr>
<td>TensorRT</td>
<td>Linear</td>
<td>10.2%</td>
<td>×1.3</td>
</tr>
<tr>
<td>TensorRT</td>
<td>All</td>
<td>1.7%</td>
<td>×1.8</td>
</tr>
<tr>
<td>PyTorch (FBGEMM)</td>
<td>Linear</td>
<td>11%</td>
<td>×1.3</td>
</tr>
</tbody>
</table>

May need custom per-layer quantization
Summary

Custom optimization (e.g. distillation, quantization) of text-to-3D models needed for mobile deployment.

Shap-E outperforms Point-E on mobile devices, possibly due to its efficient implicit representation.

Synthesis quality:

Text $\rightarrow$ 2D $\rightarrow$ 3D > Text $\rightarrow$ 3D