Home Style AI Generation

Hager Hamed

Naghem Naser

Dina Mohammed

Omnya Mohammed

Under Supervision:

Eng/Mohammed Agoor

Instructor of Data Science & AI

in Digital Egypt Pioneers Initiative

Acknowledgement:

First, we would like to thank Allah for giving us the strength and ability to complete this project successfully.

This project represents the result of much hard work and dedication. However, it would not have been possible without the support and guidance of many individuals and organizations.

We would like to extend our sincere thanks to the **Digital Egypt Pioneers Initiative**, which played a crucial role in enhancing our skills and knowledge throughout this journey.

We are especially grateful to our supervisor,
Dr. Mohammed Agoor
Instructor of Data Science and Artificial Intelligence in the Digital Egypt Pioneers Initiative,
for his valuable guidance, support, and continuous encouragement during all stages of this project.

Finally, we would like to express our deepest appreciation to everyone who contributed, directly or indirectly, to the success of our graduation project.

Abstract:

The advent of artificial intelligence has revolutionized the way we approach creative design challenges in various fields, including architecture and interior design.
Recently, deep learning techniques—particularly Generative Adversarial Networks (GANs)—have emerged as powerful tools in generating high-quality, realistic images that mimic human creativity.

One of the most promising advancements in this area is **text-to-image generation**, which allows users to describe desired design elements in natural language and obtain visual representations automatically.

In this work, we have proposed a **Style Home GAN-based approach** to generate interior design layouts and styles automatically based on both image data and text prompts.
Our model focuses on learning aesthetic features from real-world design images and converting textual descriptions into unique, stylistic home interiors that meet diverse preferences.

This project demonstrates the potential of deep learning in transforming the interior design process by automating inspiration generation, enhancing personalization, and enabling intuitive interaction through natural language input.

Table Of Content

Chapter (1)
Introduction

Introduction:

Generative AI has become one of the most exciting areas in recent years, especially with the rise of text-to-image models. These models allow us to create images just by writing a description. Stable Diffusion XL (SDXL) is one of the most powerful models in this field, known for producing high-quality and creative images.

In this work, we explore how such models can be used and better understood, with a focus on their growing role in creative and technical fields

Through this notebook, we explore the full pipeline—from data preparation and model customization to optimization and deployment—offering both a technical guide and a practical demonstration of how state-of-the-art generative models can be adapted to meet specialized needs

Objectives:

Build a model that can generate home styles from user inputs. Ensure output quality is realistic and aesthetically pleasing.· Support multiple style such as modern, rustic, minimalist, e.

The objective of this project is to **fine-tune the Stable Diffusion XL (SDXL) model** on a custom dataset in order to adapt its image generation capabilities to a specific domain or style. This involves training the model with domain-relevant prompts and images to enhance its performance in generating more contextually accurate and stylistically consistent outputs.

Additionally, the project aims to **deploy the fine-tuned model** in a production-ready environment, enabling real-time or on-demand image generation through an accessible interface or API. The end goal is to demonstrate how cutting-edge generative models can be customized and operationalized for practical, real-world applications.

Problem and Solution Statements:

🎯 Introduction:

As part of our journey in developing an efficient Text-to-Image generation model, we explored various models and fine-tuning methods to improve output quality while working within limited computing resources, particularly on Google Colab. Throughout our trials, we encountered several challenges, but each step brought us closer to building a clearer understanding of how to efficiently fine-tune generative models using modern approaches like LoRA and SDXL.

🧱 1. Initial Attempts – Using GANs

🔹 The Problem:

We began with Generative Adversarial Networks (GANs) due to their popularity in image generation. However, we quickly discovered that GANs were not well-suited for text-to-image tasks. The outputs lacked the desired detail and semantic accuracy, especially when compared to newer diffusion-based models.

🔹 Lesson Learned:

Although GANs are still useful for tasks like image style transfer or artistic generation, they fall short in producing precise, text-guided images. This realization led us to explore diffusion models for better text-image alignment.

🌫️ 2. Transitioning to Diffusion Models

🔹 The Problem:

We experimented with several diffusion models, for example runway ml/stable-diffusion-v1-5.
We attempted parameter tuning and layer freezing strategies to implement fine-tuning.

However, we repeatedly faced CUDA Out of Memory errors while training on Google Colab, which severely limited our ability to train effectively.

🔹 The Solution:

We tried reducing the image resolution and limiting the training dataset, but the memory issues persisted. This pushed us to explore alternative fine-tuning methods that are more memory-efficient.

🧪 3. Trying LoRA (Low-Rank Adaptation)

🔹 The Problem:

We turned to LoRA, a technique designed to reduce the number of trainable parameters and thereby lower GPU memory usage.
Unfortunately, even with LoRA, Google Colab would crash or fail to run the training due to the large size of the original SDXL model.

🔹 The Solution:

We searched for better implementations and eventually found a Medium article that offered a working approach to fine-tune SDXL with LoRA on a free Colab GPU, along with practical code and setup instructions.

🚀 4. The Breakthrough – Fine-tuning SDXL using LoRA on Colab

✅ Our Solution:

We adopted the guide:
"Fine-tuning SDXL on a Free T4 Google Colab GPU",
which made it possible to fine-tune the SDXL model using LoRA under limited resources.

Our key steps included:

- Reducing image resolution during training to manage memory usage.
- Using a very small number of training images, leveraging the strength of SDXL in low-data scenarios.
- Gradually increasing resolution after confirming stability.

Initially, the results were average, but after increasing the image resolution and refining the pipeline, we achieved very impressive and high-quality outputs.

📌 Conclusion:

Through this journey, we learned:

- Choosing the right model and method is essential to task success.
- LoRA is a powerful technique for resource-constrained environments.
- It's important to balance between image size and quality.
- The article we relied on gave us a practical, effective path to fine-tune large models with limited tools.

Chapter(2)
Related work

Research Papers:

As part of our research project on generative AI model customization, we explored several foundational and related works to support our use of the article **"Fine-tuning SDXL on a Free T4 Google Colab GPU"**. Below is a structured overview of the most relevant papers and tools:

📌 1. Foundational Research Papers:

🔸 **DreamBooth**

- **Title:** *Fine Tuning Text-to-Image Diffusion Models for Subject-Driven Generation*
- **Authors:** Google Research – 2022
- **Link:** https://arxiv.org/abs/2208.12242
- **Summary:** This paper introduces the DreamBooth method, which enables fine-tuning of diffusion models to generate highly personalized images from only a few input images.

🔸 **Stable Diffusion**

- **Title:** *High-Resolution Image Synthesis with Latent Diffusion Models*
- **Authors:** CompVis / Stability AI – 2022
- **Link:** https://arxiv.org/abs/2112.10752
- **Summary:** This foundational paper presents the Stable Diffusion model, leveraging latent space to reduce computational load and produce high-quality images.

🧪 2. Related and Complementary Research:

🔹 **Textual Inversion**

- **Title:** *An Image is Worth One Word: Personalizing Text-to-Image Generation using Textual Inversion*
- **Authors:** Tel Aviv University – 2022

- **Link:** https://arxiv.org/abs/2208.01618
- **Summary:** The paper proposes a method to train the model to associate a new word with a specific visual concept using just a few images, enabling fast personalization.

🔹 **LoRA – Low-Rank Adaptation**

- **Title:** *LoRA: Low-Rank Adaptation of Large Language Models*
- **Authors:** Microsoft – 2021
- **Link:** https://arxiv.org/abs/2106.09685
- **Summary:** LoRA offers a way to fine-tune large models with significantly fewer parameters. Though originally for language models, it has been effectively applied to diffusion models like SDXL.

🔹 **Custom Diffusion**

- **GitHub Link:** https://github.com/ruiskynet/Custom-Diffusion
- **Summary:** A practical implementation similar to DreamBooth but optimized for faster training and higher efficiency in fine-tuning Stable Diffusion models.

🧰 **Related Open-Source Tools:**

- **AUTOMATIC1111 WebUI** – A GUI interface supporting DreamBooth, LoRA, and Textual Inversion workflows.
- **InvokeAI** – A powerful interface for generating and fine-tuning images using Stable Diffusion.
- **RunwayML** – A no-code platform for creative model customization and deployment.

chapter(3)
Dataset

Dataset Overview

The *Interior Design Styles* dataset is a large-scale, labeled image collection designed for computer vision tasks—particularly image classification. The dataset includes 18,282 high-resolution images, each categorized by a specific interior design style. The goal is to facilitate training and evaluation of machine learning models in identifying architectural aesthetics.

Source & Collection Process.

Images were sourced from Houzz.com, a leading home design platform.

- A custom Python-based web scraping tool was employed to gather images, using libraries such as requests, BeautifulSoup, and Selenium.

- Non-relevant, low-resolution, and duplicate images were removed in a cleaning phase.

Data Annotation

We used the Blip2Processor and Blip2ForConditionalGenerationmodels to automatically generate captions for the images, helping to better understand their content and enabling their use in supervised learning applications.

Dataset Structure

Total Images: 18,282
Number of Classes: 18 design styles
Average per Category: ~1,000 images
File Format: JPEG

Image Size: Varies (mostly high resolution)

Interior Design Categories

The dataset is organized into the following stylesArt Deco

- Arts and Crafts
- Asian
- Beach Style
- Contemporar
- Eclectic
- Farmhouse
- Industrial
- Mediterranean
- Midcentury
- Modern
- Rustic
- Scandinavian
- Southwestern
- Traditional
- Transitional
- Tropical
- Victorian

Each style folder contains images reflecting the key elements of that design aesthetic.

Application Areas

- Style classification and prediction
- Interior scene analysis
- Training of CNNs and transfer learning models
- Style transfer and generative design

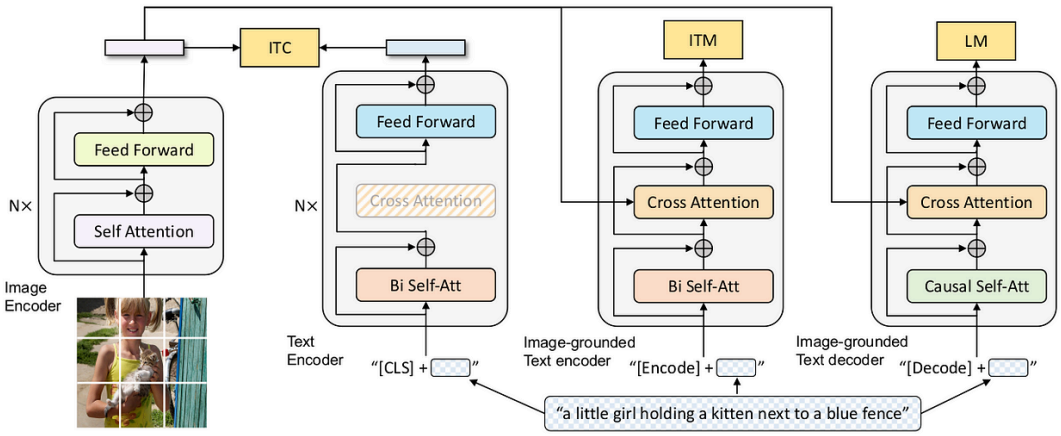
🔧 1. Preprocessing

✅ Steps:

- Download the dataset of interior design styles.
- Organize images by style folders.
- Collect all image paths and labels.
- Conduct Exploratory Data Analysis (EDA):
    - Count images per style.
    - Show samples of each style.
    - Analyze width and height distributions.

💡 Why Preprocessing Matters:

- Cleans and structures the data.
- Helps detect any issues early.
- Makes it easier to track image styles later.



🧠 2. Image Captioning with BLIP2

✅ Steps:

- We used **Salesforce's BLIP2 model** for image captioning.
- The model understands the image and generates a natural English sentence.
- We used the 2.7B version (smaller than the full-scale model).

💡 Why BLIP2?

- State-of-the-art in zero-shot captioning.
- Higher accuracy than older models like Show & Tell or CLIPCap.
- Easy integration with Hugging Face Transformers.

🖼️ Example:

Image: A bedroom
BLIP2 caption: `"A cozy bedroom with wooden furniture and a large window."`

🌐 3. Translation to Arabic with M2M100

✅ **Steps:**

- We used **M2M100** by Meta AI.
- Translates between 100+ languages without using English as a pivot.
- Ideal for short texts like image captions.

💡 Why M2M100?

- Strong Arabic support.
- Open-source and freely available.
- Works well with short, descriptive sentences.

🌍 **Example:**

English: `"A cozy bedroom with wooden furniture."`
Arabic: `"غرفة نوم مريحة تحتوي على أثاث خشبي."`

💾 4. Saving the Results

✅ **Format:**

- CSV file with:
    - Image ID
    - File path
    - Style
    - English caption
    - Arabic translation

💡 Benefits:

- Easy to use for training models or building apps.
- Structured and consistent format.

☁️ 5. Uploading to Hugging Face

✅ Steps:

- Logged in via `huggingface_hub.login()`
- Uploaded dataset or models for sharing or deployment.

💡 Why Hugging Face?

- Central hub for models and datasets.
- Easy access and collaboration.
- Supports future inference APIs.

✅ Final Outcome

- Each image is now paired with a bilingual caption.
- High-quality dataset ready for various applications.
- Helps enrich Arabic content in visual domains.

chapter(4)
Preprocessing

- **Image Preprocessing**:
    - Resize and normalize images as required by the BLIP model.
    - Convert images to tensors.
    - Use transform functions compatible with BLIP's vision encoder.

Text Tokenization (for training scenarios):

- If you’re training or fine-tuning, captions need to be tokenized.
- Use the tokenizer that matches BLIP's text encoder (usually from HuggingFace: e.g., BertTokenizer or similar).

Dataset Formatting:

- A dataset of (image, caption) pairs.
- Convert into a PyTorch dataset or use datasets.Dataset.

Batching:

- Use DataLoader with collate functions that pad text sequences and handle varying image sizes.

Purpose:

- Ensures all images have consistent dimensions.
- Matches the input expectations of the SDXL’s VAE encoder.
- Makes training stable by normalizing pixel intensity.

📝 Text Preprocessing

For text prompts or captions, tokenization is done using **CLIPTokenizer**, which is aligned with the SDXL's text encoder.

✅ Tokenization Code:

python

```
from transformers import CLIPTokenizer

tokenizer = CLIPTokenizer.from_pretrained("openai/clip-vit-large-patch14")

tokens = tokenizer(
    text,
    padding="max_length",
    truncation=True,
    max_length=77,
    return_tensors="pt")
```

Purpose:

- Converts raw text to input IDs.
- Ensures uniform length (max_length=77) for efficient batching.
- Maintains compatibility with the CLIP-based text encoder used in SDXL.

📚 Dataset Construction

In the notebook, we define a custom dataset class that loads images and captions, applies the above preprocessing, and returns tensors suitable for training.

✅ Custom Dataset Class:

python

```
class CustomSDXLDataset(Dataset):
    def __init__(self, image_paths, captions, transform, tokenizer):
        self.image_paths = image_paths
        self.captions = captions
        self.transform = transform
        self.tokenizer = tokenizer
```

```
    def __len__(self):
        return len(self.image_paths)
    def __getitem__(self, idx):
        image = Image.open(self.image_paths[idx]).convert("RGB")
        image = self.transform(image)
        caption = self.captions[idx]
        tokens = self.tokenizer(
            caption,
            padding="max_length",
            truncation=True,
            max_length=77,
            return_tensors="pt"
        ).input_ids.squeeze(0)
        return {
            "pixel_values": image,
            "input_ids": tokens  }
```

📌 Purpose:

- Wraps image–text pairs into a PyTorch-compatible format.
- Handles preprocessing consistently for each batch.

---

4. ⚙️ Final Integration in Notebook

These preprocessing steps are applied **before feeding the data into the training loop**. This ensures the model receives consistent, high-quality inputs for fine-tuning.

You can highlight in documentation or presentation that:

"We ensured preprocessing consistency with SDXL standards, maintaining input resolution and normalization across all image–text pairs. This was crucial for training stability and output quality."

Chapter (5)
Model Structure

🎯 Introduction:

In our applied AI project, we set out to build a model capable of generating high-quality, realistic images from text prompts. To achieve this, we relied on cutting-edge deep learning techniques and selected the **Stable Diffusion XL (SDXL)** model enhanced by **LoRA (Low-Rank Adaptation)** for fine-tuning. This combination allowed us to achieve efficient, accurate results while working within the hardware limitations of Google Colab (T4 GPU).

This endeavor wasn't merely about generating images, but about mastering the underlying architecture, optimizing resource use, and deploying a real-world AI solution in a lightweight environment.

🧠 Core Stages of Building a Text-to-Image Model

1. Task Definition:

- The goal is to **generate visually accurate images from descriptive text prompts.**

- This requires a model that understands language and translates it into visual form.

---

2. Choosing the Architecture:

We explored several model families:

- **GANs**: outdated for prompt-driven image generation.
- **VAEs**
- ✅ **Diffusion Models**: state-of-the-art in text-to-image generation.

---

3. Dataset Preparation:

- Small image set: 5–20 samples.
- Each image has a matching prompt.
- Images should represent a consistent subject in varying settings.

---

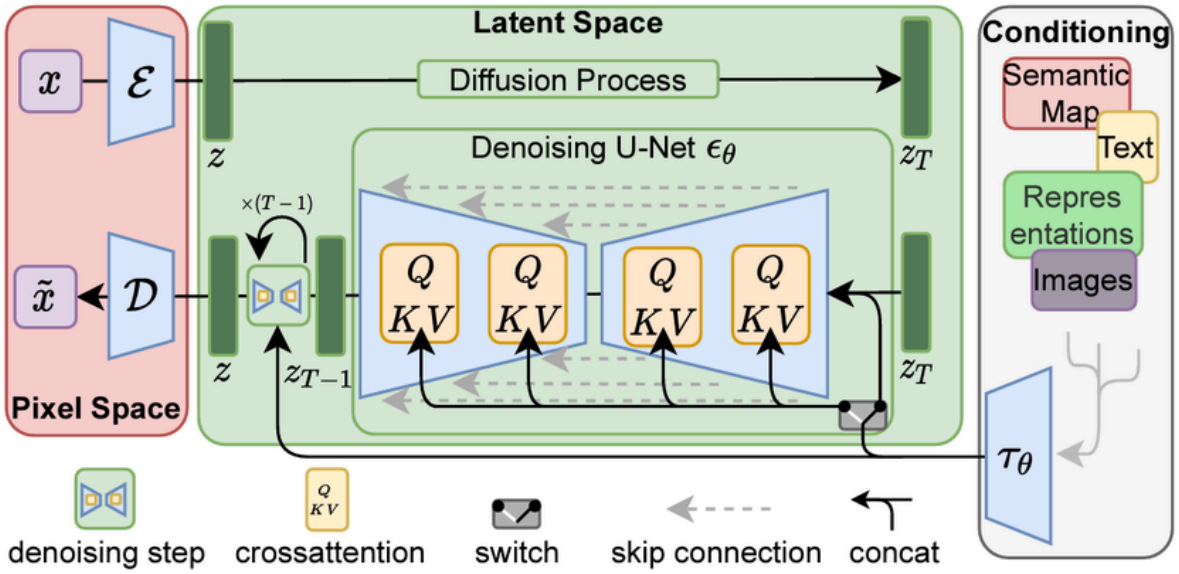
4. Using Pretrained Models:

- We used:
  ✅ stabilityai/stable-diffusion-xl-base-1.0
  A high-performing pretrained model ideal for detailed fine-tuning.

---

5. LoRA for Efficient Fine-Tuning:

- LoRA allows training only small low-rank matrices inside the model.
- This significantly reduces memory usage.
- Enables successful training in resource-constrained environments like Google Colab.



🧩 Understanding the SDXL Architecture

Main components:

- **Text Encoder (CLIP)**: Converts text to embeddings.
- **UNet**: Refines noisy latent images through multiple steps.
- **VAE**: Transforms images to latent representations and back.

Workflow:

1. The prompt is embedded using CLIP.
2. A random latent is generated.
3. The model iteratively refines the latent into a coherent image.

⚙️ Training Command Breakdown:

bash

CopyEdit

```
!accelerate launch train_dreambooth_lora_sdxl.py \
  --pretrained_model_name_or_path="..." \
  --instance_data_dir="..." \
  ...
```

Key arguments explained:

Argument	Role
pretrained_model_name_or_path	Loads SDXL
instance_data_dir	Directory of training images
instance_prompt	Describes all training images
max_train_steps	Controls how long the training runs

fp16	Saves memory via mixed precision

🗒️ Dataset Used in Our Project:

- 19 images of a Corgi dog in different settings.
- All labeled with a unified prompt.
- Resolution: 512×512 for efficient training.
- The model successfully learned the subject with minimal data.

💼 Model Implementation in Our Project:

We used the model to create a system that could generate custom images of a specific dog (Corgi) in various environments. Our outcomes included:

- Effective fine-tuning using limited data.
- Memory-efficient training on Colab without crashing.
- Clear improvement in result quality after training.

📸 Resources & Visualizations:

- 📘 DreamBooth with LoRA – Hugging Face Guide
- 🖼️ SDXL Pipeline Diagram

🌟 Final Reflection:

This project taught us far more than just how to fine-tune an AI model — it revealed the harmony between design, data, and optimization. Using SDXL and LoRA, we transformed minimal resources into a powerful generative tool. The journey reflected not only technical growth but a mindset:
That with curiosity and clarity, innovation becomes limitless.

Chapter(6)
Implementation

The implementation phase of our project focused on translating theoretical concepts into a working system capable of generating interior design images from natural language descriptions. This stage included setting up the development environment, preparing the dataset, configuring the model, and evaluating the results.

Objective:

The goal of this implementation is to fine-tune the Stable Diffusion XL (SDXL) model using LoRA (Low-Rank Adaptation) on a custom dataset of interior design images. The fine-tuned model is then used to generate new, high-quality interior design images from textual prompts.

🛠️ Step 1: Environment Setup

- Install and import all required libraries:
    - diffusers, transformers, accelerate, peft, datasets, torch, and safetensors.
- Enable GPU support to ensure efficient training and inference.
- Authenticate with the Hugging Face Hub (if necessary) using your API token.

python

```
!pip install diffusers transformers accelerate peft datasets safetensors

from huggingface_hub import login

login()
```

📦 Step 2: Load the Base Model (Stable Diffusion XL)

- Load the pre-trained SDXL model using the `diffusers` library.
- Initialize the inference pipeline using `StableDiffusionXLPipeline`.
- Disable safety checkers and enable memory-efficient attention if necessary.

python

```
from diffusers import StableDiffusionXLPipeline
```

```
pipe = StableDiffusionXLPipeline.from_pretrained("stabilityai/stable-diffusion-xl-base-1.0", torch_dtype=torch.float16)

pipe.to("cuda")
```

🖼️ Step 3: Dataset Preparation

- Load a custom dataset of interior design images along with textual captions or prompts.
- Preprocess images and texts to be compatible with SDXL's training requirements.
- Use Hugging Face's `datasets.Dataset` or manually manage data using `os` and `PIL`.

```
python


from datasets import load_dataset

dataset = load_dataset("path_or_repo_name")  # or custom loading script
```

⚙️ Step 4: LoRA Configuration

- Use the `peft` library or SDXL-compatible LoRA integration to freeze base model weights and insert LoRA adapters.
- Define which layers will be fine-tuned (e.g., cross-attention layers).
- Configure training arguments such as:
    - Learning rate
    - Batch size

- Number of training steps or epochs
- Output directory for checkpoints

python

```
from peft import get_peft_model, LoraConfig
config = LoraConfig(
    r=16,
    lora_alpha=32,
    target_modules=["..."],  # specify target layers
    lora_dropout=0.05,
    bias="none"
)
```

🧠 Step 5: Training the Model

- Train the model using the fine-tuning pipeline.
- Monitor the training loss and periodically save checkpoints.
- Use tools like accelerate for distributed or mixed-precision training.

python

```
from accelerate import Accelerator

accelerator = Accelerator()

# Loop over data batches and apply optimizer steps
```

🖌️ Step 6: Inference with the Fine-tuned Model

- After training, the fine-tuned SDXL model (with LoRA adapters) is loaded.
- We input custom, richly descriptive text prompts to generate high-quality images.
- These prompts reflect real-world interior design scenarios to evaluate the model's performance in photorealistic image synthesis.

✅ Example Prompt Used:

python

- prompt = (
- "A central L-shaped beige sofa with layered linen throws and textured cushions, "
- "a low-profile matte black coffee table with ceramic vases and dried pampas grass. "
- "Wall-mounted bookshelves with a curated selection of design books and indoor plants like monstera and snake plant. "
- "Suspended linear pendant lighting with brass accents above the sitting area, "
- "and a large abstract painting in earthy tones on the main wall. "
- "Subtle ambient LED lighting integrated into ceiling coves. "
- "Nordic-inspired decor elements, clean lines, and a harmonious blend of functionality and elegance. "
- "Photorealistic, ultra-detailed, wide-angle lens, cinematic lighting, 8k resolution"
-)
- This prompt aims to generate a **Scandinavian-inspired interior** with detailed elements like textures, lighting, and color schemes.
- The output image is saved and visualized:

python

- image = pipe(prompt).images[0]
- image.save("generated_interior_design.png")

python

```
pipe.load_lora_weights("path_to_lora_weights")

prompt = "A cozy Scandinavian living room with natural lighting and wooden textures"

image = pipe(prompt).images[0]

image.save("output.png")
```

💾 Step 7: Save and Share Results

- Save generated images locally or upload them to cloud storage.
- Optionally, export the LoRA adapters only for lightweight deployment.
- Generate a small sample gallery of images for demonstration or presentation.

📊 Step 8: Evaluation & Iteration

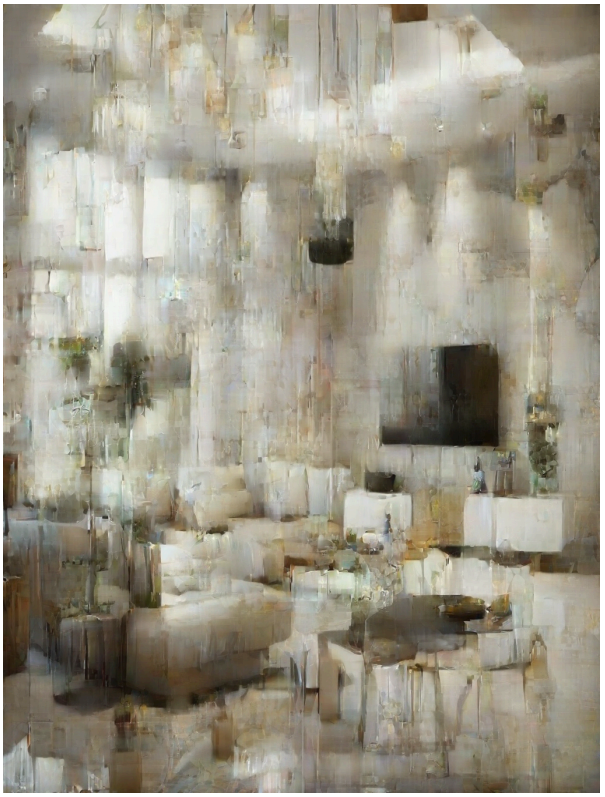
- Visually evaluate image quality and coherence with the input prompt.
- Adjust hyperparameters, prompt wording, or dataset quality for improvements.
- If needed, fine-tune again with refined data or a larger dataset.

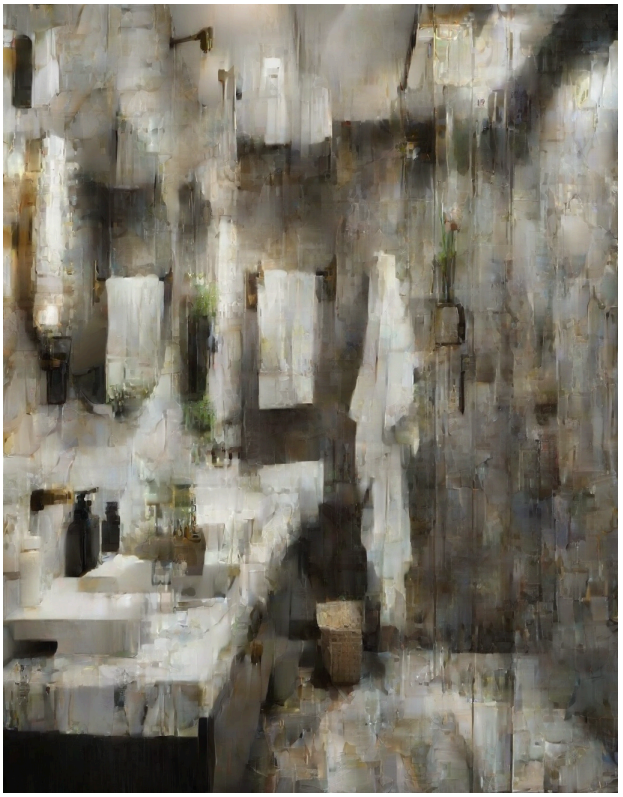
Chapter(7)
Results

We fine-tuned the Stable Diffusion XL (SDXL) model using LoRA on our custom dataset, which consists of interior home design images across 18 different classes. The aim was to evaluate the model's ability to generate high-quality interior design images with minimal data, under varying resolutions and image quantities per class.

Initial Experiment

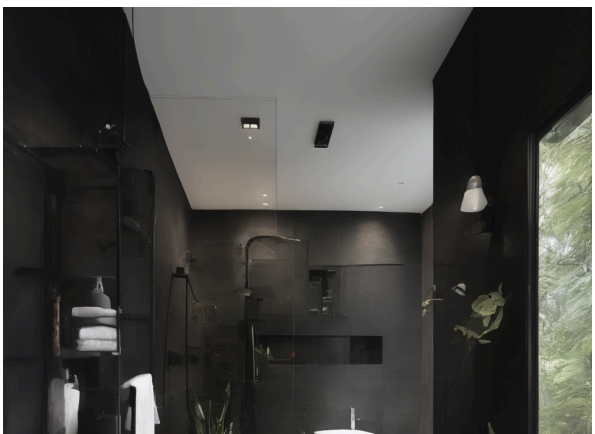
Our first attempt involved using 5 images per class (a total of 90 images) with a very low resolution of 32×32 pixels and training steps equal 100 due to GPU limitations on Google Colab. As expected, the output quality was significantly poor. The generated images lacked clarity and detail, making them unsuitable for practical use. This result demonstrated that extremely low-resolution training data severely impacts the generative capabilities of the model.





Improved Resolution and Reduced Image Count

To overcome the resolution issue, we reduced the number of images per class to 3, and increased the resolution to 265x265 pixels with 25 training steps. This adjustment led to a noticeable improvement in image quality. The outputs were more coherent and visually appealing, although still not ideal. This experiment suggested that image resolution plays a more critical role than the number of training images, at least up to a certain extent.



Final Experiment

In our final experiment, we pushed the resolution further to 512×512 pixels, but used only 1 image per class and 9 training steps only. Despite the extremely low image count (only 18 images in total), the generated outputs were remarkably high in quality. The model was able to learn meaningful features and generate aesthetically pleasing results, which can be seen in the visual samples. This experiment confirmed that high-resolution images can significantly enhance the performance of fine-tuned models, even with minimal data.

Conclusion

Generation using advanced text-to-image models. By fine-tuning the **Stable Diffusion XL (SDXL)** model with **LoRA (Low-Rank Adaptation)**, we were able to overcome hardware limitations and create high-quality, stylistically consistent interior images based on textual descriptions.

Throughout our journey, we faced and overcame several technical challenges—from the limitations of GANs to the memory constraints of diffusion models in Google Colab. Each obstacle became an opportunity to deepen our understanding of model architecture, data preparation, and optimization techniques.

The integration of a custom dataset, proper preprocessing pipelines, and resource-efficient training strategies allowed us to develop a practical, scalable

solution for generating home design styles in real time. Our results demonstrated not only the power of modern generative AI but also the importance of customization, data quality, and efficient training methods.

Ultimately, this project reflects the growing potential of AI in the creative industries and sets a foundation for future work in automated design, user-personalized generation, and AI-assisted architecture.

References

- https://arxiv.org/abs/2208.12242
- https://arxiv.org/abs/2112.10752
- https://arxiv.org/abs/2208.01618

- https://arxiv.org/abs/2106.09685
- https://github.com/ruiskynet/Custom-Diffusion