

# AL/ML Internship - Problem Test

## Problem Statement:

Create a small LLM application to generate Mobile UI Designs.

## Overview of the Project:

The goal of this project is to develop a retrieval-augmented generation (RAG) program that can produce mobile user interface designs. Based on user-provided text inputs, the application creates user interface (UI) graphics using Stable Diffusion for computer vision and a Large Language Model (LLM) for natural language processing.

## Objective:

- **Model Selection & Training:** Produce UI designs using pretrained models.
- Develop an application that allows users to enter text and obtain UI design graphics using Gradio UI or a Python-based command-line interface.
- **Evaluation:** Assess the design quality based on visual clarity and user input response.
- **Text Input:** The LLM (GPT-Neo) is used to generate descriptive text based on the user's input query.
- **Image Generation:** The Stable Diffusion model interprets this text to generate corresponding UI design images.

## Steps Followed:

### 1. Dataset Exploration

- **Dataset Used:** Mobile UI Design Dataset from Hugging Face <https://huggingface.co/datasets/mrtoy/mobile-ui-design>.
- **Dataset Exploration:** The dataset contains various mobile UI screenshots with metadata. Each data entry has:
  - Image of the UI design.
  - Bounding boxes for different objects in the UI.
  - Metadata regarding the dimensions, objects, etc.
- We used the Hugging Face `datasets` library to load and inspect the data:

```
from datasets import load_dataset

# Load the dataset
dataset = load_dataset("mrtoy/mobile-ui-design")

print(dataset['train'].column_names)
print(dataset['train'][0])
```

## 2. Model Selection

- **LLM for Text Generation:** We selected GPT-Neo (125M parameters) from EleutherAI for generating design descriptions from user input.
- **Stable Diffusion for Image Generation:** The Stable Diffusion model from the `diffusers` library was used for generating high-quality images based on text descriptions.
- **Dependencies:**
  - `transformers` for GPT-Neo.
  - `diffusers` for Stable Diffusion.
  - `datasets` for loading the Mobile UI dataset.
  - `gradio` for creating a simple user interface.

## 3. Model Loading and Text Generation

- We first load the text-generation model (GPT-Neo):

```
# Load LLM for text generation (GPT-Neo as an example)
model_name = "EleutherAI/gpt-neo-125M"
model = AutoModelForCausalLM.from_pretrained(model_name)
tokenizer = AutoTokenizer.from_pretrained(model_name)
```

- Why GPT-Neo: GPT-Neo is a model built by EleutherAI that is open-source and offers powerful text-generation capabilities. For this project:

**Pretrained on General Text:** GPT-Neo is pre-trained on a vast corpus, allowing it to understand and generate relevant design descriptions from short input queries.

**Flexibility:** We can fine-tune the model if needed, but the current setup uses the pretrained model directly for generating text based on user input.

- Then, we use a pipeline to generate a description from a user query:

```
def generate_design(query, num_images=3, guidance_scale=7.5, num_inference_st
    # Using LLM to generate text based on input
    nlp_model = pipeline('text-generation', model=model_name)
    generated_text = nlp_model(query, max_length=50)[0]['generated_text']
    print("Generated text description:", generated_text)
```

## 4. Image Generation with Stable Diffusion

- After generating the design description, Stable Diffusion was used to generate mobile UI images:

```
# Load Stable Diffusion for image generation
pipe = StableDiffusionPipeline.from_pretrained("CompVis/stable-diffusion-v1-4").to("cuda")
```

- **Why Stable Diffusion:** Stable Diffusion is a state-of-the-art image-generation model:

**High-Quality Images:** It can generate photorealistic images based on text prompts.

**Efficiency:** The model runs efficiently on GPUs, making it ideal for generating multiple high-quality UI designs in a reasonable amount of time.

**Flexibility in Image Generation:** By tweaking parameters like `guidance_scale` and `num_inference_steps`, we can control the level of creativity and refinement in the generated images.

## 5. Application Development

- **Gradio Interface:** For a user-friendly interface, Gradio was used to allow users to input a query, generate a description using GPT-Neo, and then create an image using Stable Diffusion.

```
import gradio as gr

def generate_design(query):
    nlp_model = pipeline('text-generation', model=model_name)
    generated_text = nlp_model(query, max_length=50)[0]['generated_text']
    generated_image = pipe(generated_text).images[0]
    return generated_image

gr.Interface(fn=generate_design, inputs="text", outputs="image").launch()
```

### Dependencies:

The application depends on the following libraries:

- `transformers`: For loading the GPT-Neo model.
- `diffusers`: For the Stable Diffusion model used in image generation.
- `datasets`: For loading the Mobile UI design dataset.
- `gradio`: For creating a simple user interface.
- `torch`: Required for running both GPT-Neo and Stable Diffusion on GPU.

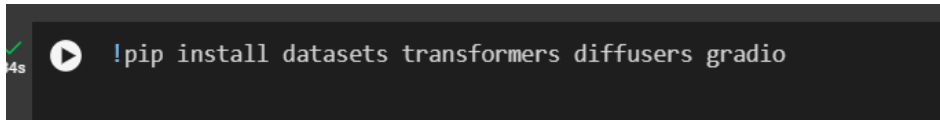
### Performance Evaluation:

- **Design Accuracy:** The generated designs are evaluated based on their similarity to the user-provided text description.
- **Visual Quality:** The images generated are reviewed for clarity and fidelity.
- **Speed:** GPU support significantly improves the time taken to generate the images. GPU is critical for efficiently generating images, especially for large models like Stable Diffusion.

## Running the Application in Google Colab:

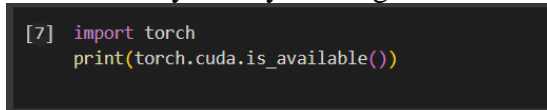
To simplify the process of running the project, we can use Google Colab for a cloud-based environment with GPU support. Below are the steps specific to Colab:

1. **Open a Google Colab Notebook:**
  - Go to Google Colab.
2. **Install Dependencies:** Run the following commands to install required libraries:

A screenshot of a Google Colab terminal window. It shows a green checkmark and a play button icon on the left, followed by the command `!pip install datasets transformers diffusers gradio`.

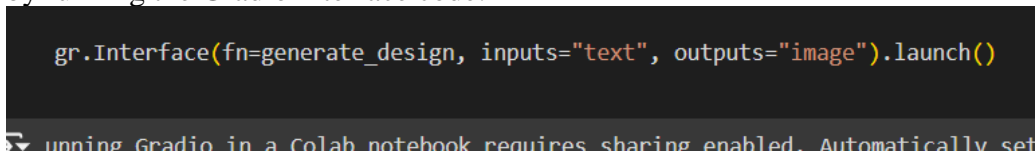
```
!pip install datasets transformers diffusers gradio
```

3. **Check GPU Availability:** Ensures that the Colab notebook is using a GPU runtime. We can verify this by running:

A screenshot of a Google Colab terminal window showing the code to check GPU availability. It includes a prompt `[7]` followed by `import torch` and `print(torch.cuda.is_available())`.

```
[7] import torch
print(torch.cuda.is_available())
```

4. **Running the Gradio UI:** We can also launch the Gradio-based interface from Colab by running the Gradio interface code:

A screenshot of a Google Colab terminal window showing the code to launch the Gradio interface. It includes `gr.Interface(fn=generate_design, inputs="text", outputs="image").launch()`. Below the code, a message states: "Running Gradio in a Colab notebook requires sharing enabled. Automatically set..."

```
gr.Interface(fn=generate_design, inputs="text", outputs="image").launch()

Running Gradio in a Colab notebook requires sharing enabled. Automatically set
```