**AI Engineer Task Report**

**Objective**

The goal of this task was to develop two AI-powered Gradio applications:

1. **Image-to-Text Generation** (Image Captioning)
2. **Text-to-Code Generation**

The report includes:

* Explanation of prompt engineering strategies and parameter choices
* Setup instructions
* Usage guidelines
* Examples and experimentation outcomes

**Image-to-Text Generation (Image Captioning)**

**🧪 Model Selection Process**

Initially, I compared the following vision-language models based on benchmark scores and community feedback:

* mPLUG-Owl2
* LLaVA-NeXT
* Qwen-VL-Chat
* deepseek-vl-1.3b-4bitill-qwen-1.5b

After evaluating benchmarks from official repositories, I chose **Qwen-VL-Chat** due to its competitive performance and the availability of a **quantized 4-bit version**, which significantly reduces memory usage without degrading output quality.

* **Qwen-VL-Chat-Int4** peak memory: **~11.82GB**
* **BF16 version**: **~22.60GB**

This made the Qwen-VL-Chat-Int4 model the most efficient choice for deployment in constrained environments (e.g., Kaggle/Colab).

**Setup Instructions**

1. Uninstall potential conflicting packages:

pip uninstall -y torch torchvision torchaudio

pip uninstall -y transformers transformers-stream-generator

pip uninstall -y accelerate optimum auto-gptq

pip uninstall -y einops gradio

1. Install compatible versions:

pip install -q torch==2.6.0+cu124 torchvision torchaudio==2.6.0+cu124 --index-url https://download.pytorch.org/whl/cu124

pip install -q transformers==4.51.3

pip install -q transformers-stream-generator==0.0.5

pip install -q accelerate==1.5.2

pip install -q optimum==1.25.3

pip install -q einops

pip install -q gradio

pip install -q auto-gptq==0.7.1 --no-build-isolation

**Prompt Engineering Strategy**

I began with the simple prompt:

Describe the image in detail.

I tested variations such as:

* "Focus on objects and actions in the image"
* "Describe every visible item in the image in detail"

However, longer prompts sometimes caused hallucinations — e.g., describing objects that weren’t in the image. The best balance between accuracy and richness came from keeping the prompt **short and clear**.

**Parameter Configuration**

Initial configuration:

* temperature=0.5: Balanced creativity
* top\_p=0.9: Slight randomness in output
* max\_new\_tokens=300: Reasonable length

**Observations:**

* Lowering temperature led to more **deterministic** outputs
* Increasing top\_p gave **creative**, but sometimes **less focused** results
* Reducing top\_p led to **precise**, but occasionally **repetitive** responses

**Interface**

* Built using **Gradio Blocks**
* Users can:
  + Upload an image
  + Adjust temperature, top\_p, and max\_new\_tokens via sliders
  + View generated description in a textbox

**Text-to-Code Generation**

**Model and Strategy**

Used: deepseek-ai/DeepSeek-R1-Distill-Qwen-1.5B

I experimented with:

* **Zero-shot prompting**: direct request
* **Few-shot prompting**: task + example outputs

The **few-shot approach consistently outperformed** zero-shot by guiding the model to follow a structured response format.

**Prompt Engineering**

The prompt included:

* A system message instructing the model to return only code
* A few Python examples (e.g., reverse string, check palindrome)
* The task statement in this format:
* Task: <user input>
* ```python

**Post-processing**

Despite prompt constraints, the model occasionally returned explanations or reasoning. To solve this:

* A post-processing step extracted only the Python code from markdown-style code blocks using regex.

**Parameter Configuration**

* temperature: **0.1 – 0.2**  
  To ensure deterministic, explanation-free code output
* top\_p: **0.7 – 0.9**  
  To balance creativity and consistency
* max\_new\_tokens: **512 – 1024**  
  To allow enough space for longer functions while avoiding excessive verbosity in case of hard code tasks it will require more than 1024 due to the explanation of steps.

**Interface**

* Developed with **Gradio Blocks**
* Users can:
  + Enter a task description
  + Adjust temperature, top-p, and max tokens
  + View clean code output

**Summary**

| **Task** | **Model Used** | **Prompting Approach** | **Post-Processing** | **Param Ranges** |
| --- | --- | --- | --- | --- |
| Image Captioning | Qwen-VL-Chat-Int4 (AutoGPTQ) | Simple direct prompt | None | Temp: 0.5–1.0Top-p: 0.8–1.0Tokens: 100–500 |
| Code Generation | DeepSeek-R1-Distill-Qwen-1.5B | Few-shot examples | Regex code extractor | Temp: 0.1–0.2Top-p: 0.7–0.9Tokens: 512–1024 |

Notebooks:  
<https://colab.research.google.com/drive/1ub2nk5eHeNvQKB97gGPwC-aiYn1qk070?usp=sharing>

<https://colab.research.google.com/drive/1Uwd3XEZBQsBBjrkzXKQ2INIyVgMesqqc?usp=sharing>