Multimodal Gradio Application (Complete Documentation Package)

1-README Documentation

Overview

This application provides a user-friendly Gradio interface for the DeepSeek-R1-Distill-Qwen-1.5B model, and llava-1.5-7b-hf supporting two main functionalities:

- Image Description: Generate detailed textual descriptions from uploaded images
- Code Generation: Produce functional code snippets from natural language prompts

Features

- Dual-mode interface for image analysis and code generation
- Adjustable generation parameters (temperature, top_p, max_new_tokens)
- Real-time parameter tuning with immediate feedback
- Optimized for Google Colab T4 GPU environment
- User-friendly interface with example prompts

Setup Instructions

Prerequisites

- Google Colab account (recommended) or local environment with GPU
- Python 3.8+
- CUDA-compatible GPU (T4 recommended, 12GB VRAM)

Installation Steps for Google Colab:

1. Open a new Google Colab notebook
2. Ensure GPU runtime is selected: Runtime → Change runtime type → Hardware accelerator → GPU (T4)
3. Install required dependencies:

```
python
!pip install transformers==4.36.0 gradio==4.8.0 torch torchvision pillow
```

Usage Guide

Starting the Application

```
python
python Objects_Task.py
```

The Hugging Face space (https://huggingface.co/spaces/momenn/Objects) runs on CPU, resulting in slower inference compared to GPU-based setups like Google Colab.

Unified Task Interface (Image Description + Code Generation):

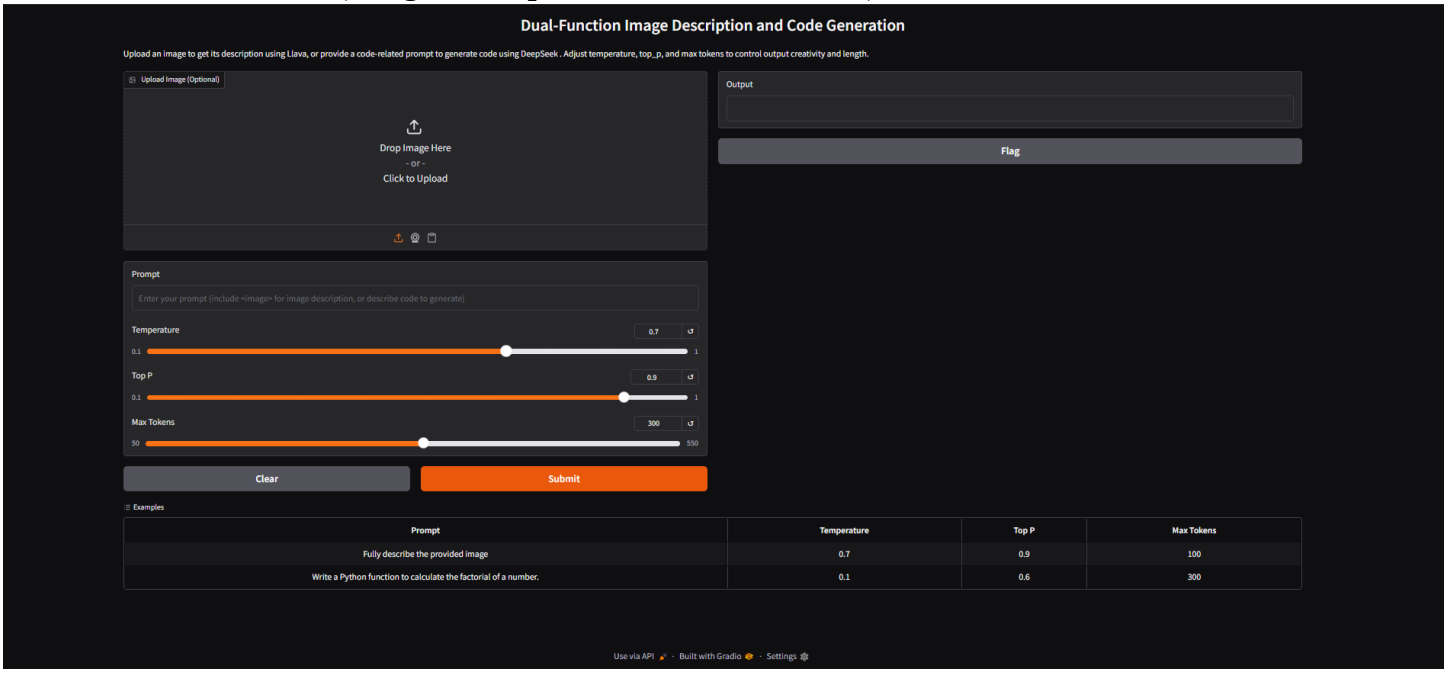


Image Description:

1. Upload Image: Click to upload JPG, PNG, or other supported formats
2. Custom Prompt: Modify the description prompt (optional)
3. Parameters: Temperature (0.1-1.0): Controls creativity vs. consistency - Top-p (0.1-1.0): Nucleus sampling parameter - Max New Tokens (50-500): Maximum length of generated description
4. Generate: Click to process the image

Code Generation:

1. Enter Prompt: Describe the code you want to generate
2. Parameters: Same as image description
3. Generate: Click to create code snippet

Quick prompt examples for quick testing.

Image Description Examples

Good prompts:

- "Describe this image in detail, including objects, colors, and scene composition"
- "Analyze this image and explain what's happening in the scene"
- "Provide a comprehensive description of all visual elements in this image"

Parameter recommendations:

- Temperature: 0.3-0.7 (balanced creativity)
- Top-p: 0.8-0.9 (diverse but coherent)
- Max tokens: 100-200 (detailed descriptions)

Code Generation Examples

Good prompts:

- "Create a Python function to calculate the factorial of a number"
- "Write a JavaScript function to validate email addresses using regex"
- "Generate a Function python code to make bubble sorting

Parameter recommendations:

- Temperature: 0.1-0.3 (precise, functional code)
- Top-p: 0.7-0.8 (focused generation)
- Max tokens: 300-400 (complete functions)

Troubleshooting

Common Issues

Slow Generation:

- Reduce max_new_tokens
- Use lower resolution images
- Ensure GPU is properly utilized

Poor Image Recognition:

- Verify image format is supported
- Check image quality and resolution
- Try different prompt formulations

Inconsistent Code Generation:

- Lower temperature (0.1-0.3)
- Be more specific in prompts
- Include programming language specification

2-Prompt Engineering & Parameter Strategy Guide

Overview

This document outlines the prompt engineering strategies and parameter optimization approaches used in the Gradio application.

Prompt Engineering Strategies

Effective Image Description Templates:

Template 1: Comprehensive Analysis "Analyze this image thoroughly and provide a detailed description including:

- Main subjects and objects
- Colors, lighting, and composition
- Setting and background elements
- Any text or notable details
- Overall mood or atmosphere"

Template 2: Structured Description "Describe this image systematically:

1. Detailed explanation about you sees in the image
2. identify objects you can see Secondary elements

Template 3: Contextual Analysis "Examine this image and describe:

- What is happening in the scene?
- Who or what are the main subjects?
- Where does this appear to be taking place?
- What details make this image unique or interesting?"

Image Prompt Optimization Results

Prompt Type	Detail Level	Inference speed
Fully describe the provided image	High	fast
Structured Template	High	medium
Contextual Analysis (What details make this image unique or interesting)	Very High	medium

Effective Code Generation Templates:

Template 1: Function Generation "Create a function that [SPECIFIC_TASK]. Requirements:

- Include proper error handling
- Add clear comments
- Use descriptive variable names
- Follow python best practices
- Include usage example"

Template 2: Implementation Guide "Implement [CONCEPT/ALGORITHM] :

1. Create the main function/class

2. Handle edge cases
3. Add comprehensive comments
4. Provide test cases
5. Explain time/space complexity"

Code Generation Optimization Results

Prompt Type	Detail Level	Inference speed
Simple request	low	medium
Structured requirements	High	medium
Best practices emphasis	Very High	slow

Parameter Tuning Strategy

Temperature Optimization

Image Description:

- Technical Content: temperature = 0.3 (high accuracy, ideal for scientific/technical images)
- Balanced Description: temperature = 0.7 (good detail and creativity for general images)
- Creative Writing: temperature = 0.9 (rich, imaginative descriptions for abstract/artistic content)

Code Generation:

- Production Code: temperature = 0.1 (precise, functional code suitable for deployment)
- General Development: temperature = 0.3 (balanced outputs for learning and prototyping)
- Creative Exploration: temperature = 0.5 (innovative solutions, alternative coding styles)

Top-p (Nucleus Sampling) Optimization

Image Description:

- Detailed Analysis: top_p = 0.8 (focused but diverse vocabulary)
- Creative Description: top_p = 0.9 (rich, varied descriptions)
- Technical Analysis: top_p = 0.7 (precise terminology)

Code Generation:

- Production Code: top_p = 0.7 (reliable, standard practices)
- Educational Examples: top_p = 0.8 (clear, well-explained)
- Algorithmic Solutions: top_p = 0.6 (focused, efficient)

Max New Tokens Optimization

Image Description:

- Standard Description: max_new_tokens = 150 (concise summary of image)
- Comprehensive Analysis: max_new_tokens = 300 (detailed exploration of image content)
- Technical Description: max_new_tokens = 400 (in-depth, may include hallucinations)

Code Generation:

- Incomplete Output (reasoning cut-off): max_new_tokens = 150 (test premature ending)
- Full Function/Script: max_new_tokens = 300 (complete code generation with explanations or comments)

Quality Assessment Metrics

Image Description Quality Indicators

- Accuracy: Factual correctness of described elements
- Completeness: Coverage of visible elements
- Detail Level: Depth of description
- Coherence: Logical flow and organization
- Specificity: Use of precise, descriptive language

Code Quality Indicators

- Functionality: Code executes without errors
- Readability: Clear structure and naming
- Documentation: Adequate comments and explanations
- Best Practices: Follows language conventions
- Completeness: Includes error handling and edge cases

3-Example Usage and Demonstrations

Overview

This guide provides practical examples, demonstration scenarios, and use cases to showcase the capabilities of the Gradio application.

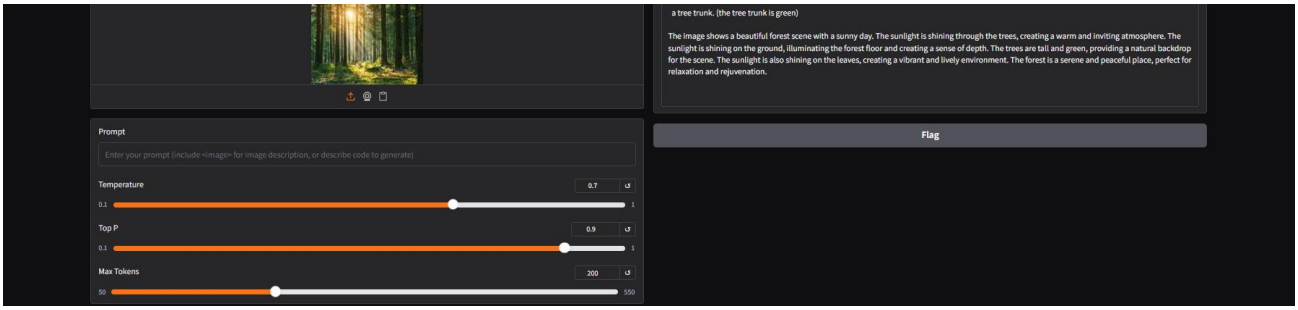
Image Description Examples

Example 1: Landscape Photography

Input Image: Forest landscape **Prompt:** "Provide a detailed description of this image"

Parameters:

- Temperature: 0.7
- Top-p: 0.9
- Max tokens: 200



Generated Output: "The image shows a beautiful forest scene with a sunny day. The sunlight is shining through the trees, creating a warm and inviting atmosphere. The sunlight is shining on the ground, illuminating

the forest floor and creating a sense of depth. The trees are tall and green, providing a natural backdrop for the scene. The sunlight is also shining on the leaves, creating a vibrant and lively environment. The forest is a serene and peaceful place, perfect for relaxation and rejuvenation.

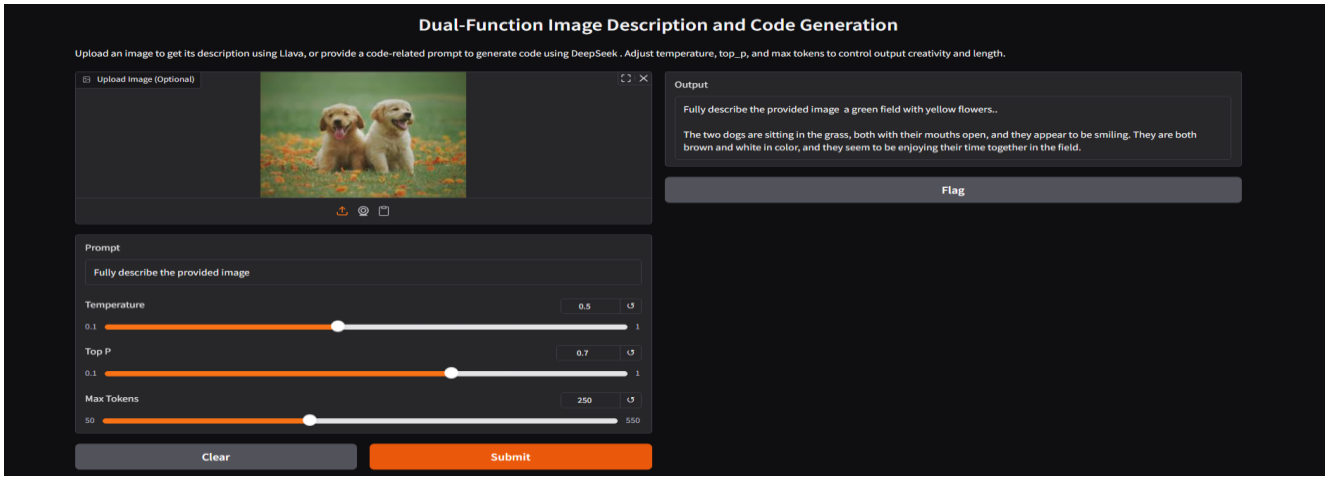
Quality Assessment: - Detailed color description - Composition analysis – Creative description

Time: 120 seconds

Example 2: Entities Recognition

Input Image: Two dogs in a field with flowers **Prompt:** "Analyze this urban scene systematically" **Parameters:**

- Temperature: 0.5
- Top-p: 0.7
- Max tokens: 250



Generated Output: " The two dogs are sitting in the grass, both with their mouths open, and they appear to be smiling. They are both brown and white in color, and they seem to be enjoying their time together in the field."

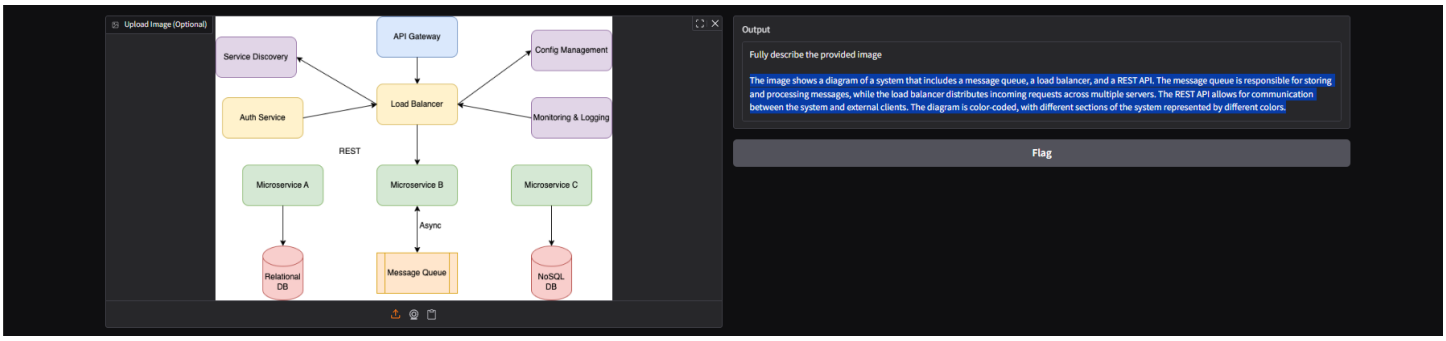
Quality Assessment: - Accurate scene identification - Less Detailed color description

Time: 90 seconds

Example 3: Technical Diagram

Input Image: Software architecture diagram **Prompt:** "Describe this technical diagram, focusing on components and relationships" **Parameters:**

- Temperature: 0.3
- Top-p: 0.6
- Max tokens: 300



Generated Output: " The image shows a diagram of a system that includes a message queue, a load balancer, and a REST API. The message queue is responsible for storing and processing messages, while the load balancer distributes incoming requests across multiple servers. The REST API allows for communication between the system and external clients. The diagram is color-coded, with different sections of the system represented by different colors."

Code Generation Examples

Example 1: Python Data Processing Function

Prompt: " Write a Python function to calculate the factorial of a number."

Parameters:

- Temperature: 0.1
- Top-p: 0.6
- Max tokens: 100

Generated Code:

```python
```python

def factorial(n):
    if n < 0:
        raise ValueError("Negative numbers are not allowed.")
    elif n == 0:
        return 1
    else:
        result = 1
        for i in range(2, n + 1):
            result *= i
        return result
```

Quality Assessment: Complete functionality - Proper error handling example provided

Time: 10 seconds

Example 2: Create bubble sort function

Prompt: "Write a Python function to fetch and display user data from an API with error handling"
```

Parameters:

- Temperature: 0.2
- Top-p: 0.7
- Max tokens: 500

Generated Code:

```
```python

def bubble_sort(arr):
    # Return early if the array is empty or has one element
    if len(arr) <= 1:
        return arr
    # Perform bubble sort
    for i in range(len(arr)):
        swapped = False  # Track if any swaps are made in this pass
        for j in range(0, len(arr) - i - 1):
            if arr[j] > arr[j + 1]:
                # Swap adjacent elements if they are in the wrong order
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
                swapped = True
        # If no swaps were made in this pass, the array is already sorted
        if not swapped:
            break
  return arr
```

Quality Assessment: Problem clearly understood and restated before implementation- Comprehensive error handling - Clear documentation

Time: 20 seconds
```