Color Palette Extraction & Image Captioning

Subtitle: Combining Image Captioning, Color Extraction, and Translation

A by Aya Joharji

Project Objectives

Image Captioning and Translation

- Generate descriptive captions for images.
- Translate captions to Arabic for multilingual support.

Color Palette Generation

- Extract dominant colors from uploaded images.
- Display colors visually and as hex codes.

Gradio Interface

- Build an interactive, user-friendly web tool.

Pipeline-Implementation Overview

1

Image Captioning Pipeline

- Model: Salesforce/blip-image-captioning-base
- Generates an English description of the image.

2

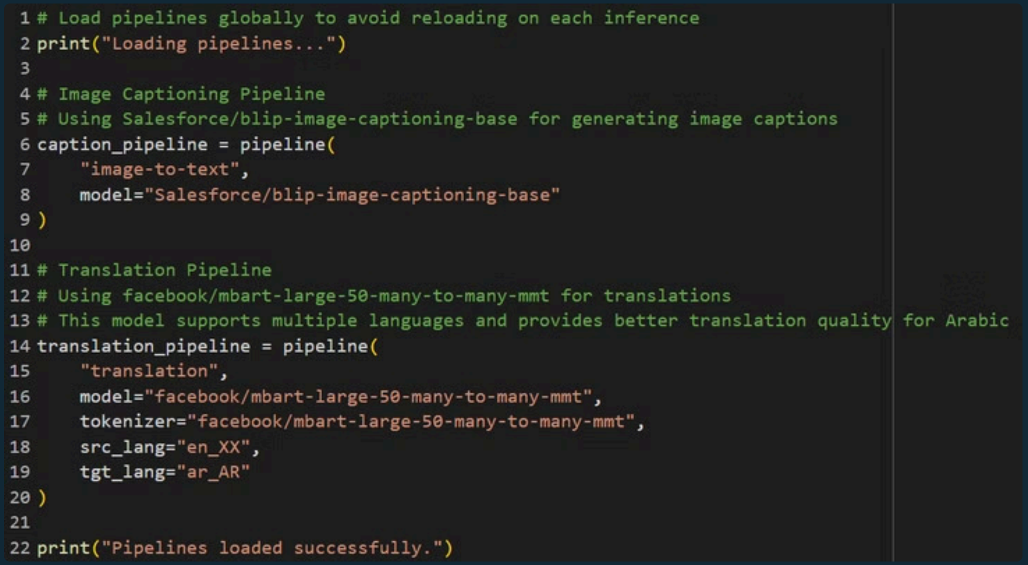
Translation Pipeline

- Model: facebook/mbart-large-50-many-to-many-mmt
- Translates captions from English to Arabic.

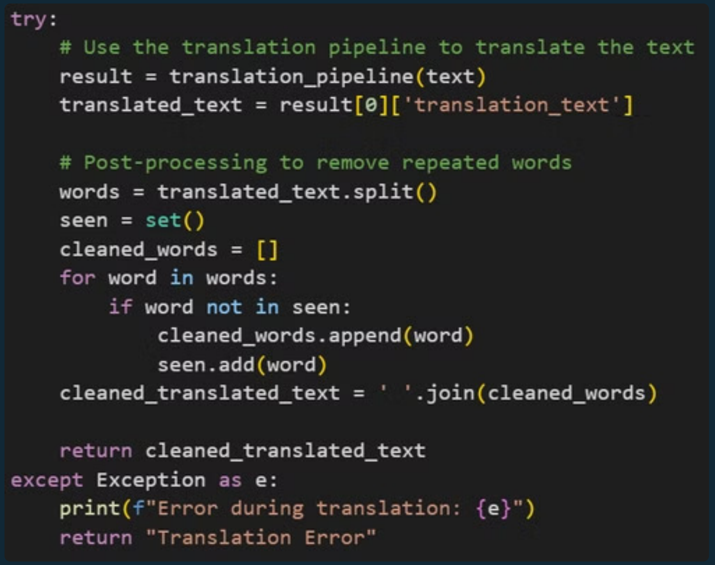
3

Color Palette Extraction

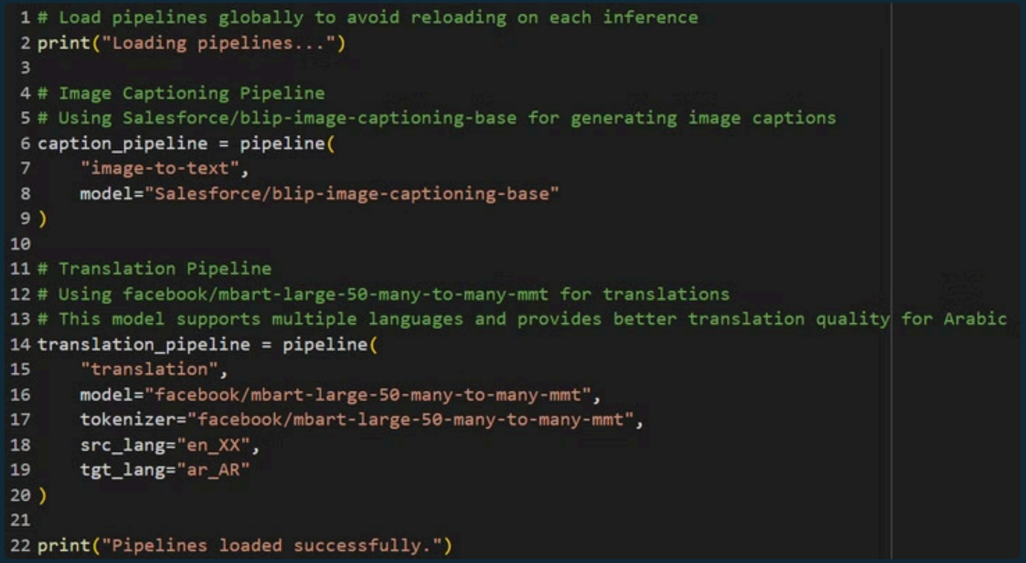
- Uses KMeans clustering to extract dominant colors.



```python
# Load pipelines globally to avoid reloading on each inference
print("Loading pipelines...")

# Image Captioning Pipeline
# Using Salesforce/blip-image-captioning-base for generating image captions
caption_pipeline = pipeline(
    "image-to-text",
    model="Salesforce/blip-image-captioning-base"
)

# Translation Pipeline
# Using facebook/mbart-large-50-many-to-many-mmt for translations
# This model supports multiple languages and provides better translation quality for Arabic
translation_pipeline = pipeline(
    "translation",
    model="facebook/mbart-large-50-many-to-many-mmt",
    tokenizer="facebook/mbart-large-50-many-to-many-mmt",
    src_lang="en_XX",
    tgt_lang="ar_AR"
)

print("Pipelines loaded successfully.")
```



```python
try:
    # Use the translation pipeline to translate the text
    result = translation_pipeline(text)
    translated_text = result[0]['translation_text']

    # Post-processing to remove repeated words
    words = translated_text.split()
    seen = set()
    cleaned_words = []
    for word in words:
        if word not in seen:
            cleaned_words.append(word)
            seen.add(word)
    cleaned_translated_text = ' '.join(cleaned_words)

    return cleaned_translated_text
except Exception as e:
    print(f"Error during translation: {e}")
    return "Translation Error"
```



```python
# Load pipelines globally to avoid reloading on each inference
print("Loading pipelines...")

# Image Captioning Pipeline
# Using Salesforce/blip-image-captioning-base for generating image captions
caption_pipeline = pipeline(
    "image-to-text",
    model="Salesforce/blip-image-captioning-base"
)

# Translation Pipeline
# Using facebook/mbart-large-50-many-to-many-mmt for translations
# This model supports multiple languages and provides better translation quality for Arabic
translation_pipeline = pipeline(
    "translation",
    model="facebook/mbart-large-50-many-to-many-mmt",
    tokenizer="facebook/mbart-large-50-many-to-many-mmt",
    src_lang="en_XX",
    tgt_lang="ar_AR"
)

print("Pipelines loaded successfully.")
```

Justification for Image Captioning Model

- **Model chosen:** Salesforce/blip-image-captioning-base
- **Pre-trained on Diverse Data**: The model is pre-trained on large datasets of images and corresponding descriptions, which means it has learned a wide variety of visual concepts.
- **Key capabilities:** Generates high-quality, natural language descriptions of images
- **Advantages:** Balances accuracy and computational efficiency
- **Suitability:** Generates concise, informative captions that provide context for the visual content

Justification for Translation Model

- **Model selected:** facebook/mbart-large-50-many-to-many-mmt
- **Multilingual capabilities:** Supports over 50 languages, including Arabic. can handle translations directly between different languages without relying solely on English as an intermediate step, which improves the translation quality.
- **Alignment with project objectives:** Creates a bilingual tool, making output accessible to a broader audience
- **Pre-trained nature:** Ensures high-quality translations without requiring significant fine-tuning

Post-processing for Arabic: Additional processing is applied to remove repeated words and improve the overall quality of the translation, ensuring it sounds natural and concise.

Justification for Color Extraction

Color Extraction: KMeans Clustering:

- KMeans was chosen due to its simplicity and effectiveness in segmenting an image into dominant color clusters.
- It efficiently identifies key colors in an image, making it ideal for generating a representative palette.
- This method is widely used in image processing for color analysis and is computationally feasible for real-time applications.

Image Captioning and Translation Pipeline

1

Input Image

User uploads image

2

Image Captioning

Describes image

3

Translation

Translates to Arabic

Image Captioning Model:

- Describes the image using a pre-trained captioning model.
- Example: "Sunset over Mountains."

Translation Model:

- Translates generated captions to Arabic.
- Ensures accessibility for Arabic-speaking users.

Color Palette Extraction Steps

1

Preprocessing

Resize and flatten the image

2

Normalization

Normalize RGB values to [0, 1].

3

Clustering

Use KMeans to extract 8 dominant colors

4

Conversions

Convert RGB back to the original scale.

Each RGB value is converted into a hexadecimal color code

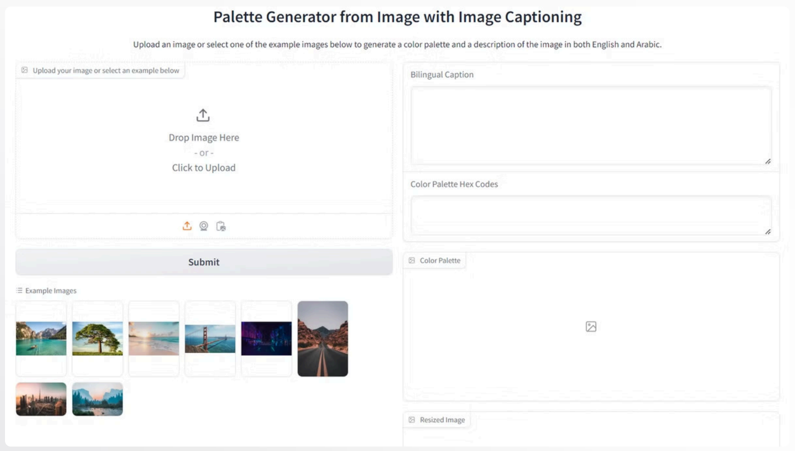
5

Visualization

Create a color palette image to represent the dominant colors.

Outputs:

- Hex Codes and Visual Palette for the dominant colors.


Palette Generator from Image with Image Captioning

Upload an image or select one of the example images below to generate a color palette and a description of the image in both English and Arabic.

Gradio Interface Design

1

2

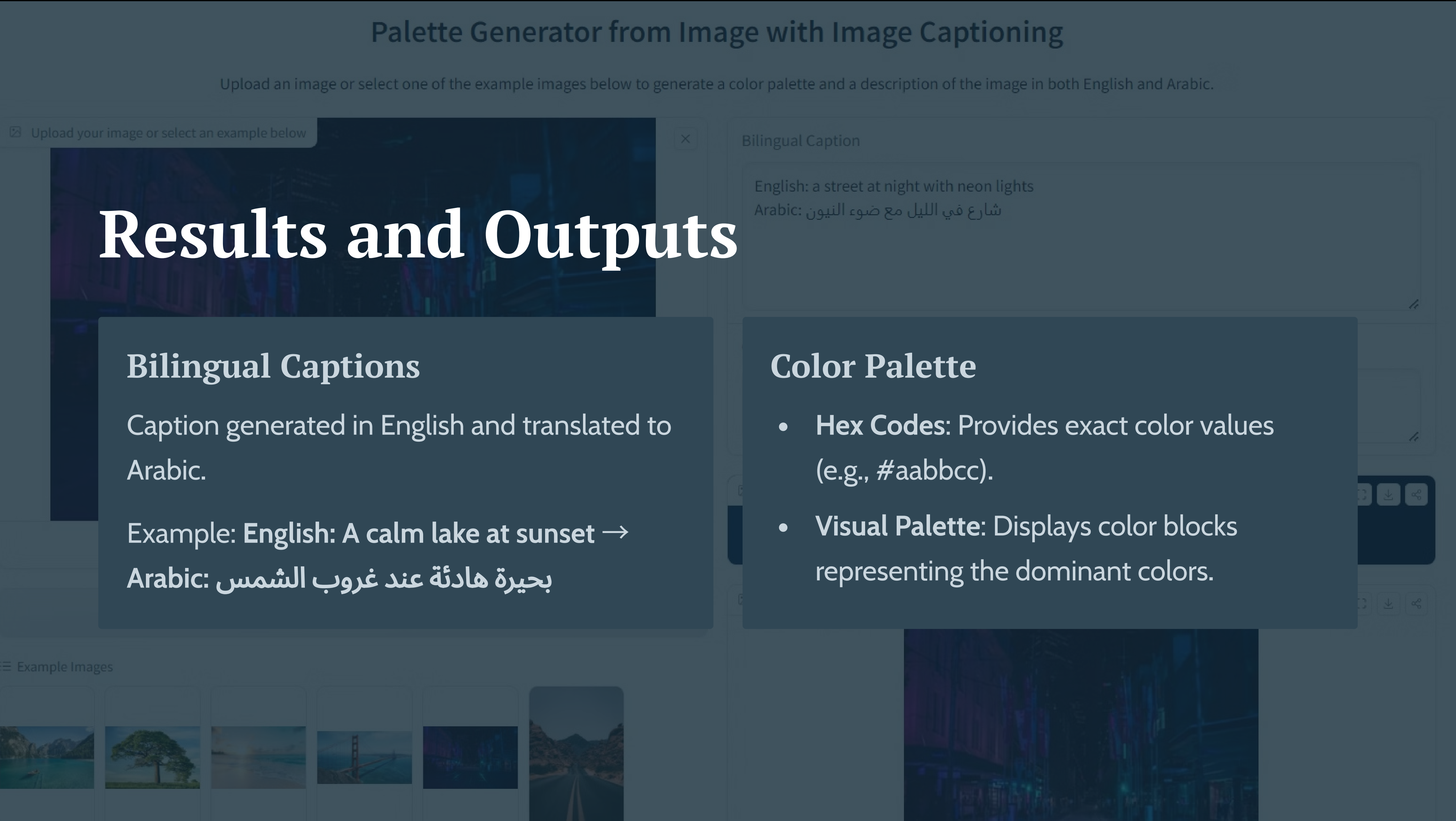
1 Interactive Elements

Image Input: Upload or select example images.

Outputs: Bilingual Caption (English & Arabic), Hex Codes of Dominant Colors, Visual Palette Image, Resized Image

2 User Flow

Upload image → Click Submit → View outputs.



Palette Generator from Image with Image Captioning

Upload an image or select one of the example images below to generate a color palette and a description of the image in both English and Arabic.

English: a street at night with neon lights
Arabic: شارع في الليل مع ضوء النيون

Results and Outputs

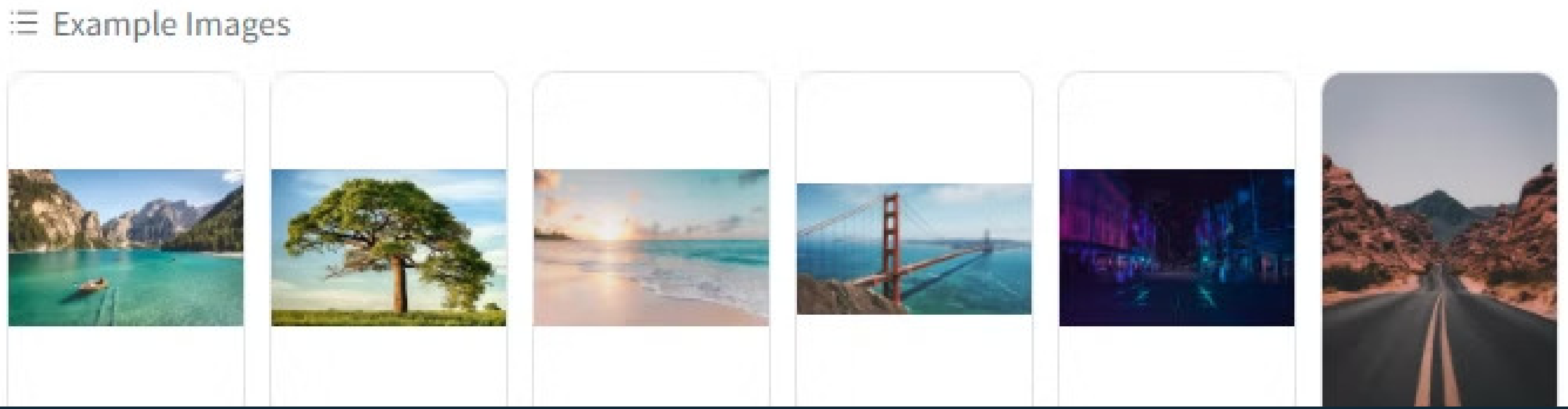
Bilingual Captions

Caption generated in English and translated to Arabic.

Example: **English: A calm lake at sunset → Arabic: بحيرة هادئة عند غروب الشمس**

Color Palette

- **Hex Codes**: Provides exact color values (e.g., #aabbcc).
- **Visual Palette**: Displays color blocks representing the dominant colors.



User Experience and Applications

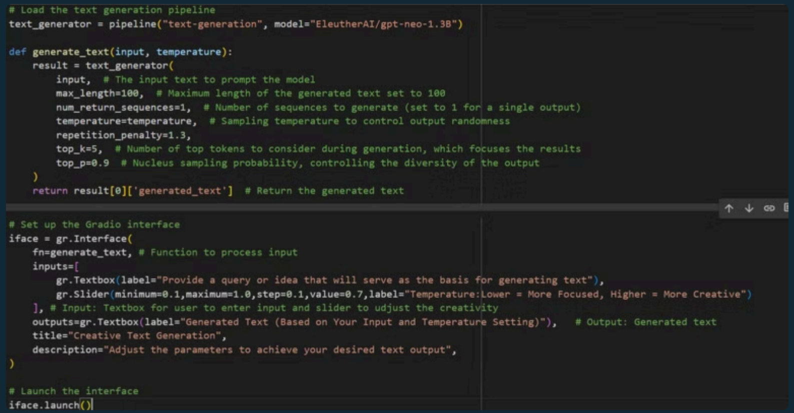
User Experience

Gradio Interface: Simple, interactive, and accessible. Examples Provided: Users can explore without needing to upload images.

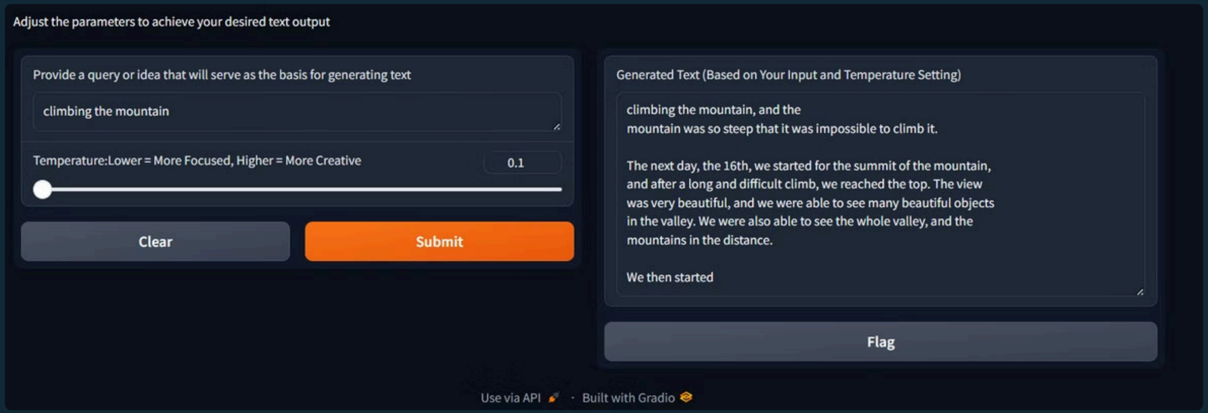
Applications

Design and Aesthetics: Extract color palettes for design inspiration. Multilingual Accessibility: Catering to both English and Arabic-speaking users.

Previous Work Overview



```
# Load the text generation pipeline
text_generator = pipeline("text-generation", model="EleutherAI/gpt-neo-1.3B")

def generate_text(input, temperature):
    result = text_generator(
        input,  # The input text to prompt the model
        max_length=100,  # Maximum length of the generated text set to 100
        num_return_sequences=1,  # Number of sequences to generate (set to 1 for a single output)
        temperature=temperature,  # Sampling temperature to control output randomness
        repetition_penalty=1.3,
        top_k=5,  # Number of top tokens to consider during generation, which focuses the results
        top_p=0.9  # Nucleus sampling probability, controlling the diversity of the output
    )
    return result[0]['generated_text']  # Return the generated text

# Set up the Gradio interface
iface = gr.Interface(
    fn=generate_text, # Function to process input
    inputs=[
        gr.Textbox(label="Provide a query or idea that will serve as the basis for generating text"),
        gr.Slider(minimum=0.1,maximum=1.0,step=0.1,value=0.7,label="Temperature:Lower = More Focused, Higher = More Creative")
    ], # Input: Textbox for user to enter input and slider to udjust the creativity
    outputs=gr.Textbox(label="Generated Text (Based on Your Input and Temperature Setting)"),   # Output: Generated text
    title="Creative Text Generation",
    description="Adjust the parameters to achieve your desired text output",
)

# Launch the interface
iface.launch()
```



Adjust the parameters to achieve your desired text output

Provide a query or idea that will serve as the basis for generating text: climbing the mountain

Temperature:Lower = More Focused, Higher = More Creative: 0.1

Generated Text (Based on Your Input and Temperature Setting):

climbing the mountain, and the
mountain was so steep that it was impossible to climb it.

The next day, the 16th, we started for the summit of the mountain,
and after a long and difficult climb, we reached the top. The view
was very beautiful, and we were able to see many beautiful objects
in the valley. We were also able to see the whole valley, and the
mountains in the distance.

We then started

Previous Work

Description: The previous project generated text using the following text-generation pipeline.

text_generator = pipeline("text-generation", model="EleutherAI/gpt-neo-1.3B")

GitHub and Hugging Face Links

GitHub Repository:

https://github.com/AyaJoharji/Color-Palette-Extraction-Image-Captioning

Hugging Face Space:

https://huggingface.co/spaces/ayajoharji/Color_PaletteExtraction_and_ImageCaptioning

Summary and Future Work

Summary

- Successfully implemented a tool that combines image captioning, translation, and color extraction.
- Gradio integration makes it accessible and easy to use.

Future Enhancements

- Adding support for additional languages.