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

Image Captioning Pipeline

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

Translation Pipeline

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

Color Palette Extraction

- Uses KMeans clustering to extract dominant colors.

Justification for Image Captioning Model

- **Model chosen:** Salesforce/blip-image-captioning-base
- **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
- **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

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 clusters to RGB values & hex codes

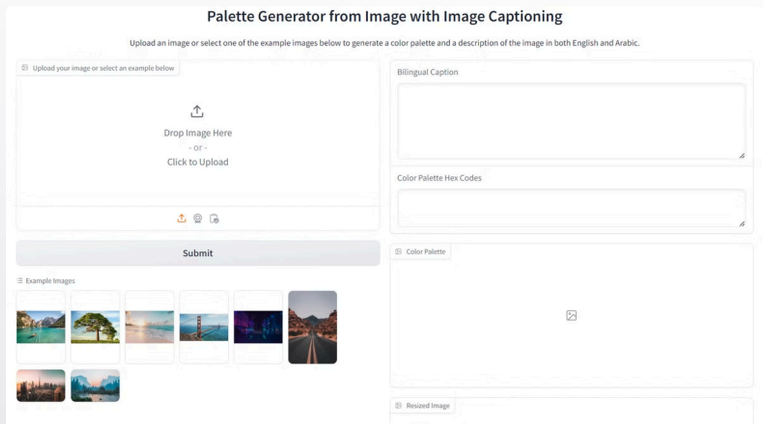
5

Visualization

Create a color palette image to represent the dominant colors.

Outputs:

- Hex Codes and Visual Palette for the dominant colors.



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.

Results and Outputs

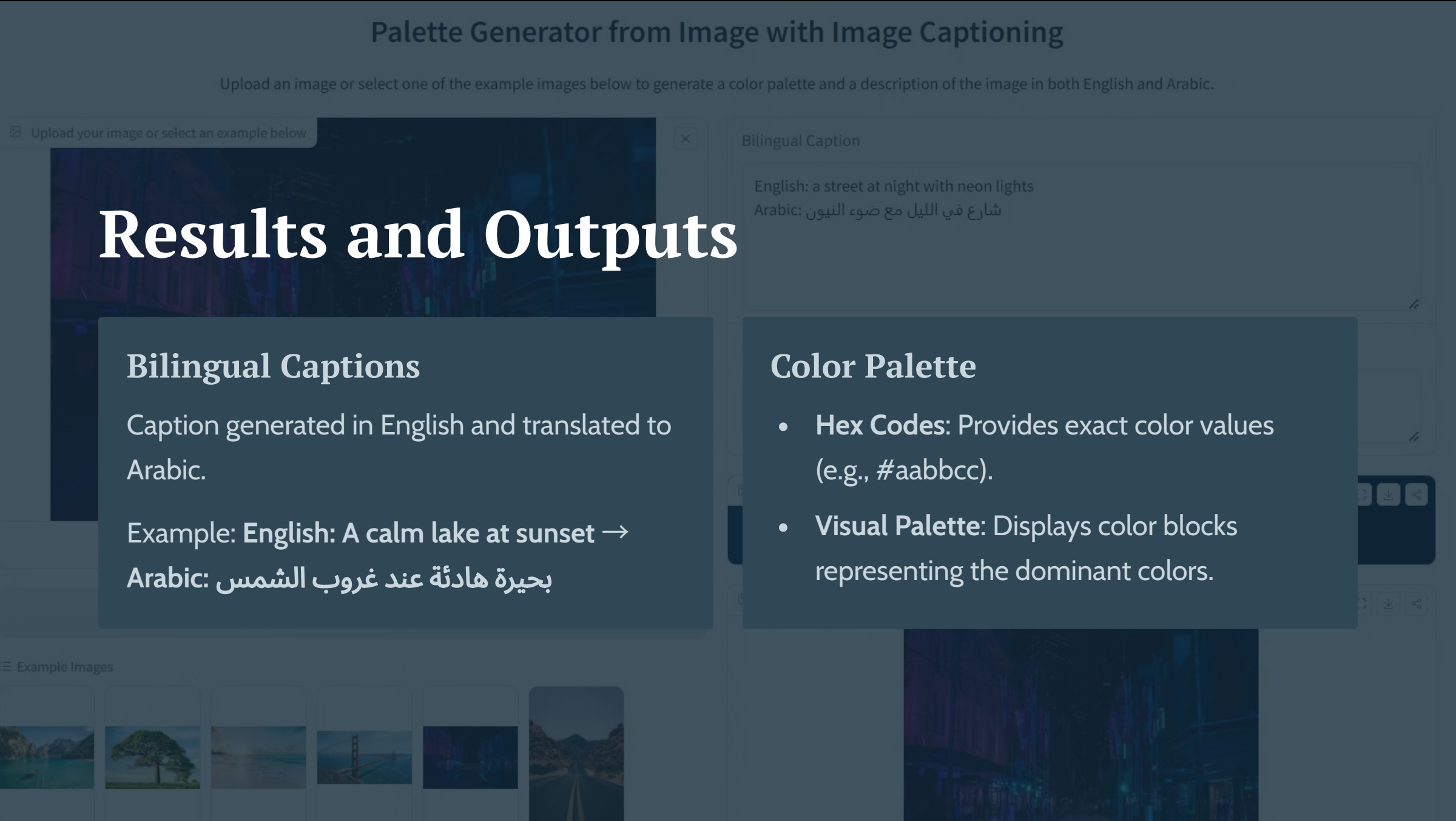
Bilingual Captions

Caption generated in English and translated to Arabic.

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

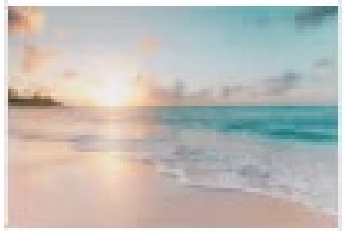
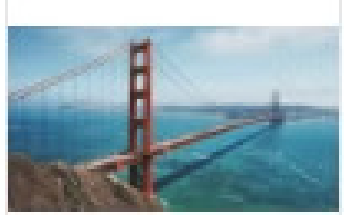
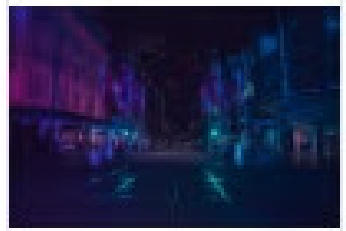
Color Palette

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



Example Images




User Experience and Applications

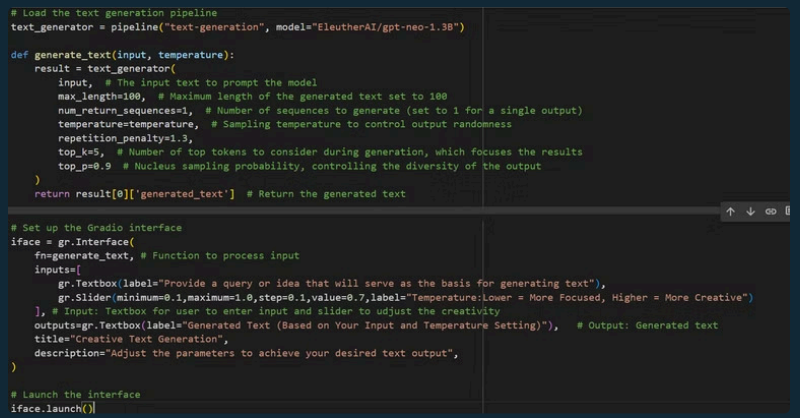
User Experience

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

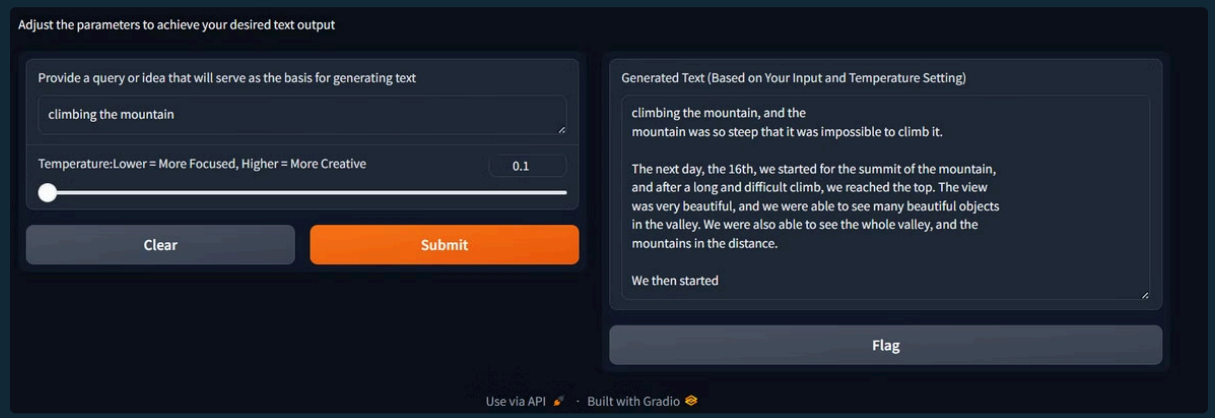


```
# Load the text generation pipeline
text_generator = pipeline("text-generation", model="EleutherAI/gpt-neo-1.3B")

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()
```



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.