Name: Atif Ansari **Roll no:** 04 **Class:** D20B

AIDS 2 EXP 3

Aim:

To build a cognitive image-based application to understand context for a customer service application using image captioning and binary matrix representation.

Theory:

Cognitive image-based context recognition is an advanced field in computer vision and artificial intelligence that aims to interpret images by mimicking human cognitive processes. Unlike traditional image recognition, it focuses on understanding the context, emotions, and intentions within visual data, integrating principles from cognitive science, psychology, and neuroscience.

Key Concepts:

1. **Cognitive Image Processing:**
    - Involves analyzing images to extract meaningful information, simulating human perception.
    - Goes beyond simple classification to deduce contextual elements like scene, objects, and activities.
2. **Context Recognition:**
    - **Scene Understanding:** Identifying the environment (e.g., office, street, hospital).
    - **Object Recognition:** Detecting and classifying objects (e.g., people, vehicles, furniture).
    - **Activity Recognition:** Inferring actions or interactions (e.g., walking, conversing).
3. **Feature Extraction:**
    - **Low-Level Features:** Basic attributes like edges, colors, and textures.
    - **High-Level Features:** Complex attributes like object shapes, spatial relationships, and semantic meaning.
4. **Applications:**
    - **Customer Service:** Automating responses by understanding customer-submitted images (e.g., damaged products, receipts).

- **Healthcare:** Analyzing medical images for diagnosis.
- **Smart Cities:** Monitoring urban environments for traffic or crowd management.
- **Government:** Enhancing surveillance or document verification systems.

5. **Challenges:**
   - **Image Variability:** Variations in lighting, angles, and occlusions affect accuracy.
   - **Complex Scenes:** Scenes with multiple objects and interactions are hard to interpret.
   - **Contextual Ambiguity:** Images may lack clear context, requiring advanced reasoning.
6. **Future Directions:**
   - Improving model robustness for diverse image conditions.
   - Enabling real-time processing for dynamic applications.
   - Developing human-AI collaborative systems for context-aware assistance.

Objective:

To develop a cognitive image-based application that generates descriptive captions for images and represents them as binary matrices, enabling context understanding for customer service applications (e.g., interpreting customer-uploaded images for support queries).

Libraries Used:

- **NumPy:** For numerical operations and array manipulation.
- **Transformers (Hugging Face):** For loading the BLIP model and processor for image captioning.
- **PIL (Python Imaging Library):** For image loading and preprocessing.

Steps:

1. **Image Loading:** Load an input image using PIL.
2. **Binary Matrix Generation:** Convert the image to grayscale, apply a threshold to create a binary matrix, and display it.
3. **Context Recognition:** Use the BLIP (Bootstrapping Language-Image Pre-training) model to generate a descriptive caption for the image.

4. **Output Display:** Present the binary matrix and the generated caption to demonstrate context understanding.

Code:

```
import numpy as np
from transformers import BlipProcessor, BlipForConditionalGeneration
from PIL import Image
import warnings
from IPython.display import display  # For displaying images in Colab
import os

warnings.filterwarnings('ignore')

def print_binary_matrix(image_path, threshold=128, max_rows=20,
max_cols=20):
    """Convert image to binary matrix and print it"""
    try:
        image = Image.open(image_path).convert('L')
        image_array = np.array(image)
        binary_array = (image_array > threshold).astype(int)

        print(f"\nBinary Matrix for {image_path}:")
        for row in binary_array[:max_rows]:
            print(' '.join(str(cell) for cell in row[:max_cols]))

        return binary_array
    except Exception as e:
        print(f"Error processing image for binary matrix: {e}")
        return None

def describe_image(image_path):
    """Generate a caption for the image using BLIP model"""
    try:
        processor =
BlipProcessor.from_pretrained("Salesforce/blip-image-captioning-base",
use_fast=True)
        model =
BlipForConditionalGeneration.from_pretrained("Salesforce/blip-image-captio
ning-base")
```

```
            image = Image.open(image_path)
            inputs = processor(images=image, return_tensors="pt")
            out = model.generate(**inputs)
            description = processor.decode(out[0], skip_special_tokens=True)
            return description
        except Exception as e:
            print(f"Error generating image description: {e}")
            return "Unable to generate description"

def process_image(image_path):
    """Process image to generate binary matrix, description, and display
image"""
    print("="*50)
    print(f"Processing Image: {image_path}")
    print("="*50)

    # Display the image
    try:
        img = Image.open(image_path)
        print("\nDisplaying Image:")
        display(img)
    except Exception as e:
        print(f"Error displaying image: {e}")

    # Generate and print binary matrix
    binary_matrix = print_binary_matrix(image_path)

    # Generate and print image description
    description = describe_image(image_path)
    print(f"\nImage Description: {description}")

    return binary_matrix, description

if __name__ == "__main__":
    # Image paths in Colab
    image_paths = ["/content/image1.jpg"]

    # Verify images exist
    for image_path in image_paths:
        if not os.path.exists(image_path):
```

```
            print(f"Image not found: {image_path}. Please upload the image
to Colab.")
            continue
        try:
            binary_matrix, description = process_image(image_path)
        except Exception as e:
            print(f"Error processing {image_path}: {e}")
```

Code Explanation:

- **print_binary_matrix:**
  - Loads an image and converts it to grayscale using PIL.
  - Applies a threshold (128) to create a binary matrix (0s and 1s) based on pixel intensity.
  - Prints a portion of the matrix (first 20 rows/columns) for readability.
- **describe_image:**
  - Uses the BLIP model from Hugging Face to generate a descriptive caption for the image.
  - Processes the image with the BLIP processor and generates a caption using the pre-trained model.
- **process_image:**
  - Combines binary matrix generation and image captioning for a given image.
  - Prints both outputs to demonstrate low-level (binary) and high-level (contextual) image understanding.
- **Main Execution:**
  - Processes two placeholder images (image1.jpg, image2.jpg).
  - Includes error handling to ensure robustness if images are missing or invalid.

Output:

```
================================================
Processing Image: /content/image1.jpg
================================================

Displaying Image:
```

```
Binary Matrix for /content/image1.jpg:
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Fetching 1 files: 100% 1/1 [00:00<00:00, 62.62it/s]

Image Description: a blue bmw parked on a road next to a tree
```

Conclusion:

This experiment successfully built a cognitive image-based application for customer service using the BLIP model for context recognition and binary matrix representation for low-level image analysis. The application:

- Generates descriptive captions to understand image context (e.g., identifying a letter or a car in a scene), enabling automated processing of customer-uploaded images.
- Produces binary matrices to highlight low-level features (edges, shapes), useful for basic image analysis in customer service workflows. The BLIP model effectively captures high-level contextual information, while binary matrices provide a simple representation of image structure. This dual approach supports applications like automated complaint analysis (e.g., assessing damaged goods) or visual query processing in customer service.

Future Work:

- Integrate with customer service chatbots for real-time image-based query resolution.
- Enhance robustness to handle diverse image conditions (e.g., low lighting, occlusions).
- Incorporate object detection models (e.g., YOLO) for detailed scene analysis.
- Develop a user interface for seamless image uploads and responses.

This experiment demonstrates the potential of cognitive image processing in enhancing customer service applications through context-aware image analysis.