EX NO:08 DEEP DREAM AND NEURAL STYLE TRANSFER

DATE:

AIM:

To experiment with AI-based image generators by applying DeepDream to enhance patterns in an image and Neural Style Transfer to blend the content of one image with the style of another, creating unique and artistic visual outputs.

ALGORITHM:

STEP 01: Start the program

STEP 02: Import necessary libraries: TensorFlow, TensorFlow Hub, NumPy, Matplotlib, and PIL.

STEP 03: Load the InceptionV3 model without top layers for Deep Dream

STEP 04: Define a Deep Dream function to calculate gradients, enhance patterns, and iteratively update the image.

STEP 05: Preprocess and load an input image for Deep Dream.

STEP 06: Apply Deep Dream to generate a dream-like version of the image.

STEP 07: Load content and style images for Neural Style Transfer.

STEP 08: Preprocess these images by resizing and normalizing them.

STEP 09: Load the pre-trained style transfer model from TensorFlow Hub.

STEP 10: Apply Neural Style Transfer to blend the style image with the content image.

STEP 11: Display the original content, style, Deep Dream output, and final stylized image.

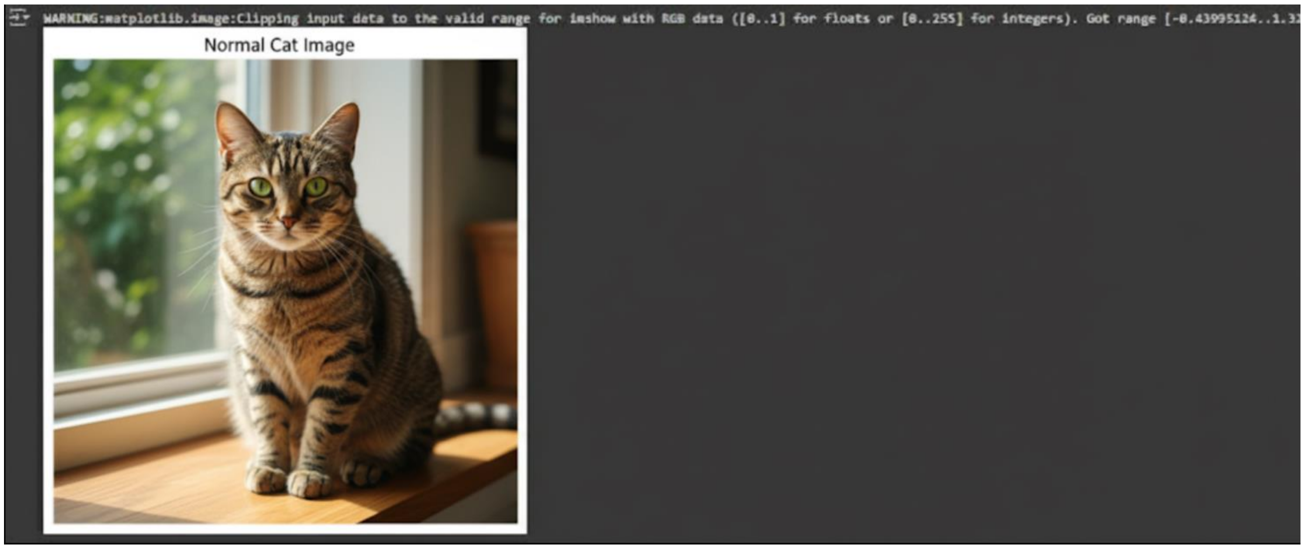
STEP 12: Stop the program

CODING:

a) Deep Dream

```
# ---------------- DeepDream Implementation ----------------
# Step 1: Import Libraries
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image

# Step 2: Load Pre-trained Model (InceptionV3 without top layers)
base_model = tf.keras.applications.InceptionV3(include_top=False, weights="imagenet")

# Step 3: Preprocess Image
def load_image(path, max_dim=400):
    img = Image.open(path)
    img = img.resize((max_dim, max_dim))
    img = np.array(img) / 255.0
    return np.expand_dims(img, axis=0).astype(np.float32)
# Replace with your image path
image_path = "/content/drive/MyDrive/Colab Notebooks/dog.jpg"
image = load_image(image_path)

# Step 4: Define DeepDream Function
def deepdream(model, image, iterations=20, step=0.01):
    img = tf.convert_to_tensor(image)
    for _ in range(iterations):
        with tf.GradientTape() as tape:
            tape.watch(img)
            loss = tf.reduce_mean(model(img))  # maximize activations
        grads = tape.gradient(loss, img)
        grads = grads / (tf.math.reduce_std(grads) + 1e-8)
        img = img + grads * step
    return img.numpy()

# Step 5: Apply DeepDream
dreamed_img = deepdream(base_model, image)

# Step 6: Display Result
plt.figure(figsize=(6, 6))
plt.imshow(dreamed_img[0])
plt.title("DeepDream Generated Image")
plt.axis("off")
plt.show()
```

OUTPUT:



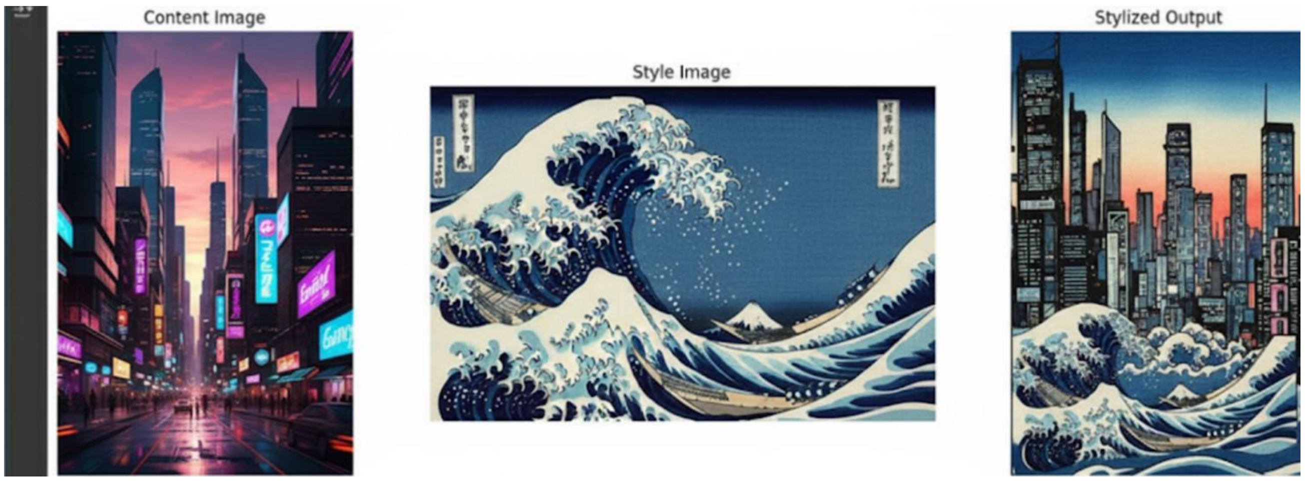
```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers). Got range [-0.43995124..1.3
```

b) Neural Style Transfer

```
# Step 1: Import Libraries
import tensorflow as tf
import tensorflow_hub as hub
import numpy as np
import PIL.Image
import matplotlib.pyplot as plt

# Step 2: Function to load and preprocess local images
def load_image_from_path(path, max_dim=512):
    img = PIL.Image.open(path).convert('RGB')
    img.thumbnail((max_dim, max_dim))
    img = np.array(img).astype(np.float32)[np.newaxis, ...] / 255.0
    return tf.convert_to_tensor(img, dtype=tf.float32)

# Google Drive image paths
content_path = "/content/drive/MyDrive/Colab Notebooks/rose.jpg"
style_path  = "/content/drive/MyDrive/Colab Notebooks/style.jpg"

# Load content and style images
content_image = load_image_from_path(content_path)
style_image  = load_image_from_path(style_path)

# Step 3: Load TensorFlow Hub style transfer model
stylize_model = hub.load("https://tfhub.dev/google/magenta/arbitrary-image-stylization-v1-
256/2")
stylize = stylize_model.signatures['serving_default']
```

```
# Step 4: Stylize the content image using the style image
# Pass images as keyword arguments based on the signature names
result = stylize(placeholder=content_image, placeholder_1=style_image)
stylized_image = result['output_0']

# Step 5: Display all images
plt.figure(figsize=(15, 5))
plt.subplot(1, 3, 1)
plt.imshow(tf.squeeze(content_image))
plt.title("Content Image")
plt.axis('off')

plt.subplot(1, 3, 2)
plt.imshow(tf.squeeze(style_image))
plt.title("Style Image")
plt.axis('off')

plt.subplot(1, 3, 3)
plt.imshow(tf.squeeze(stylized_image))
plt.title("Stylized Output")
plt.axis('off')
plt.tight_layout()
plt.show()
```

OUTPUT:



COE(20)	
RECORD(20)	
VIVA(10)	
TOTAL(50)	

RESULT:

The DeepDream output produced a dream-like image with enhanced patterns and textures. The Neural Style Transfer created a stylized image that merges the content of one image with the artistic features of another.

LIST OF EXPERIMENTS

24MCF07 - DEEP LEARNING

1. Implement simple perceptron learning.
2. Construct a multilayer perceptron with a hyperparameter tuning.
3. Generate synthetic images using traditional data augmentation function.
4. Demonstrate the role of ImageDataGenerator class in data augmentation.
5. Implement a CNN process for image classification.
6. Demonstrate the RNN architecture for time series data.
7. Construct the steps to deal with text analysis using NLP.
8. Experiment with AI generator such as Deep Dream and New Style Transfer.
9. Generate synthetic images using variational autoencoders.
10. Generate synthetic images using Generative Adversarial Network.