EX. NO: 03

DATE :

SYNTHETIC IMAGE GENERATION USING TRADITIONAL DATA AUGMENTATION

AIM:

To generate multiple synthetic images from a single input image using traditional data augmentation techniques such as rotation, shifting, flipping, zooming, and shearing.

ALGORITHM:

Step 1: Import the necessary libraries (`tensorflow.keras`, `matplotlib`, `numpy`, etc.).

Step 2: Upload a sample image using the Google Colab file uploader.

Step 3: Load the uploaded image and preprocess it:

- Resize it to a fixed dimension (e.g., 224x224).
- Convert it to a NumPy array.
- Expand dimensions to simulate batch input.

Step 4: Initialize an `ImageDataGenerator` with augmentation parameters:

- Rotation
- Width/height shift
- Zoom
- Shear
- Horizontal flip

Step 5: Use the generator (`flow`) to create and visualize multiple augmented versions of the image using a loop and `matplotlib`.

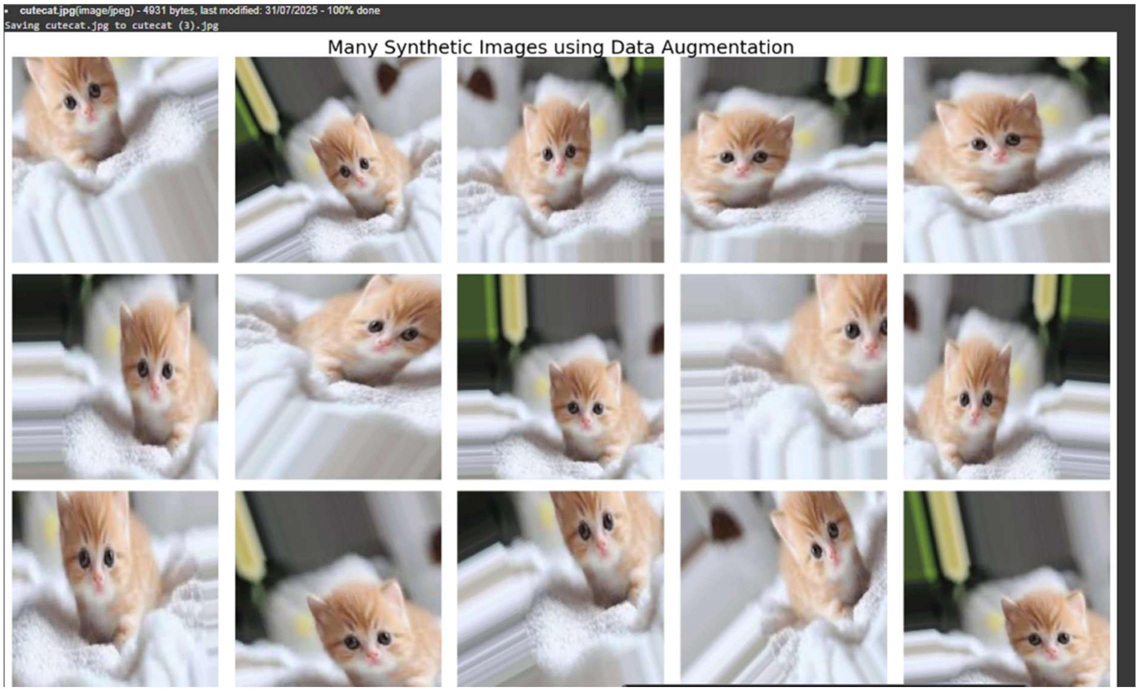
Step 6 (Optional): Save a specific number of generated images to a folder in your Colab environment.

PROGRAM:

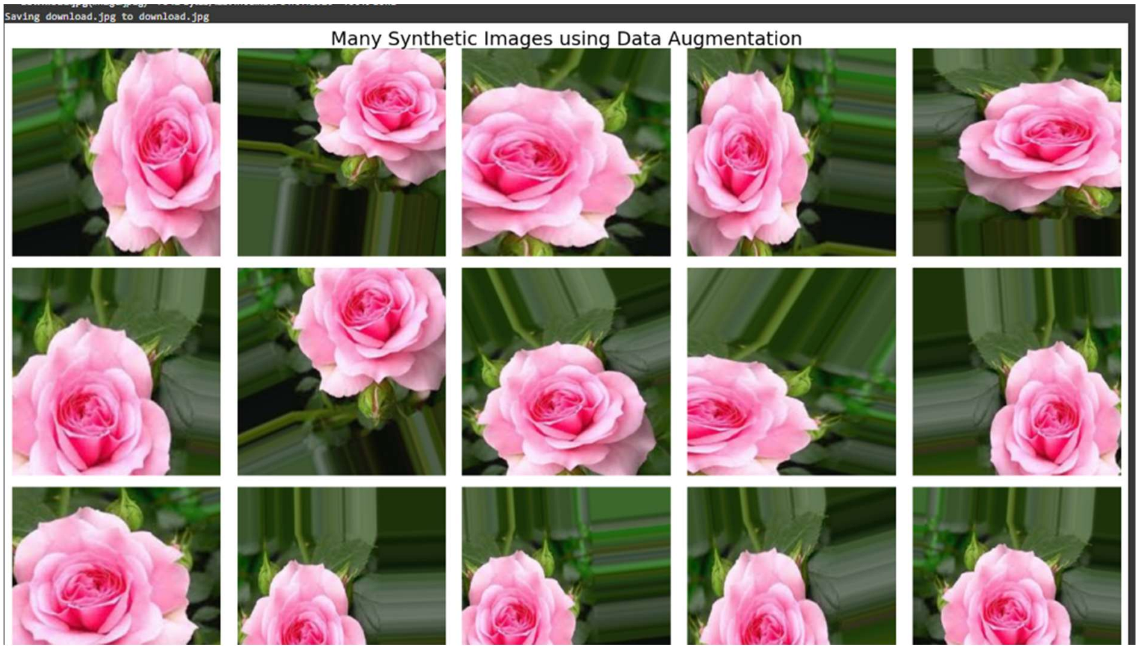
```
#  STEP 1: Install required libraries (usually pre-installed in Colab)
!pip install -q matplotlib pillow

#  STEP 2: Import libraries
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img, img_to_array
import os
from google.colab import files

#  STEP 3: Upload an image
uploaded = files.upload()  # Choose a file like cat.jpg
img_path = list(uploaded.keys())[0]

#  STEP 4: Load and preprocess the image
img = load_img(img_path, target_size=(224, 224))  # Resize image
img_array = img_to_array(img)  # Convert to numpy array
img_array = np.expand_dims(img_array, axis=0)  # Add batch dimension

#  STEP 5: Create ImageDataGenerator with traditional augmentations
datagen = ImageDataGenerator(
   rotation_range=40,
   width_shift_range=0.3,
   height_shift_range=0.3,
   shear_range=0.2,
   zoom_range=0.3,
   horizontal_flip=True,
   fill_mode='nearest'
)
```

```
#  STEP 6: Generate and display MORE augmented images
aug_iter = datagen.flow(img_array, batch_size=1)

num_images = 25  # You can increase this value (e.g., 36, 49)
rows = int(np.sqrt(num_images))
cols = int(np.ceil(num_images / rows))

plt.figure(figsize=(15, 15))
for i in range(num_images):
    batch = next(aug_iter)
    image_aug = batch[0].astype('uint8')
    plt.subplot(rows, cols, i + 1)
    plt.imshow(image_aug)
    plt.axis('off')

plt.suptitle("Many Synthetic Images using Data Augmentation", fontsize=18)
plt.tight_layout()
plt.show()

#  STEP 7: (Optional) Save many augmented images to disk
save_dir = '/content/many_augmented_images'
os.makedirs(save_dir, exist_ok=True)

datagen_save = ImageDataGenerator(
    rotation_range=40,
    width_shift_range=0.3,
    height_shift_range=0.3,
    shear_range=0.2,
    zoom_range=0.3,
    horizontal_flip=True,
    fill_mode='nearest'
)

i = 0
for batch in datagen_save.flow(img_array, batch_size=1, save_to_dir=save_dir, save_prefix='aug', save_format='jpg'):
    i += 1
    if i >= 50:  # Save 50 augmented images
        break

print(f" Saved {i} augmented images to: {save_dir}")
```

OUTPUT:



```
•  cutecat.jpg(image/jpeg) - 4931 bytes, last modified: 31/07/2025 - 100% done
Saving cutecat.jpg to cutecat (3).jpg
```



```
Saving download.jpg to download.jpg
```

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

RESULT:

Multiple synthetic images are successfully generated and displayed using traditional augmentation methods, increasing data variability for better model generalization.