EX:No.2 DATE: 21/2/25	**Simple CNN for Image Segmentation**

AIM:

To build a simple CNN for image segmentation.

ALGORITHM:

1. **Start**
2. **Import Libraries**
   a. Use TensorFlow/Keras, NumPy, etc.
3. **Load & Preprocess Dataset**
   a. Load images and corresponding masks.
   b. Normalize both images and masks.
   c. Resize to fixed dimensions.
   d. Split into train/val/test sets.
4. **Build CNN Segmentation Model**
   a. Input → Conv2D → ReLU → MaxPooling
   b. Repeat for more layers
   c. UpSampling2D → Conv2D (to reconstruct image)
   d. Output layer with sigmoid or softmax
5. **Compile Model**
   a. Use binary_crossentropy or categorical_crossentropy
   b. Optimizer: adam, Metric: accuracy or IoU
6. **Train Model**
   a. Fit model with training data (image, mask pairs)
7. **Evaluate & Predict**
   a. Evaluate with test data
   b. Predict masks for new images
8. **Stop**

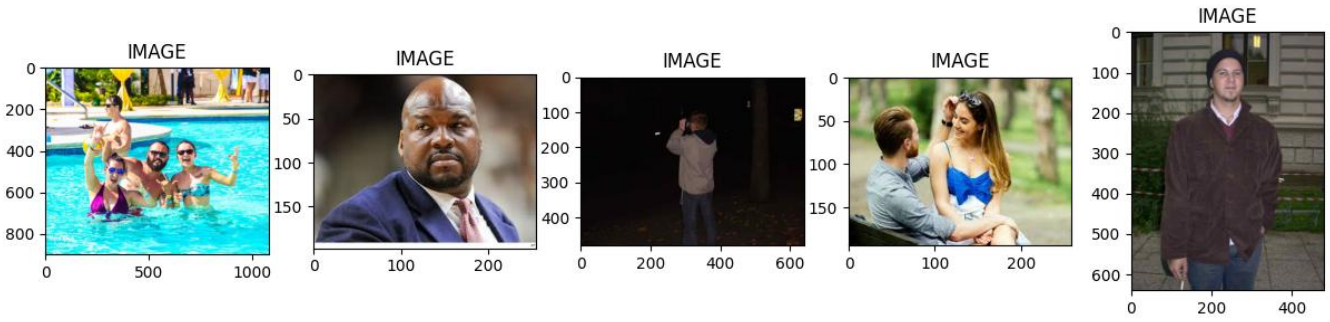
CODE:

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import torch

import cv2

from sklearn.model_selection import train_test_split

from tqdm import tqdm


TRAIN_DATA_PATH = '/kaggle/working/Human-Segmentation-Dataset-master/train.csv'

DATA_DIR = '/kaggle/working'
```

```
# Select the device to train on
DEVICE = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
# Define hyperparameters
EPOCHS = 10        # number of epochs
LR = 0.001         # Learning rate
IMG_SIZE = 320     # Size of image
BATCH_SIZE = 32    # Batch size
# Define pretrained encoder model and weights
ENCODER = 'timm-efficientnet-b0'
WEIGHTS = 'imagenet'
TRAIN_DATA_PATH = 'Human-Segmentation-Dataset-master/train.csv'  # Change the path
DATA_DIR = '/kaggle/working'

# Select the device to train on
DEVICE = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")

# Define hyperparameters
EPOCHS = 10        # number of epochs
LR = 0.001         # Learning rate
IMG_SIZE = 320     # Size of image
BATCH_SIZE = 32    # Batch size

# Define pretrained encoder model and weights
ENCODER = 'timm-efficientnet-b0'
WEIGHTS = 'imagenet'
df = pd.read_csv(TRAIN_DATA_PATH)
print(df.shape)
df.head()
sample = df.iloc[np.random.randint(0, df.shape[0], size=5)]

def generate_sample_images(sample):
    imgs = sample.images
    _, ax = plt.subplots(1, 5, figsize=(15,3))
```

```
    ax = ax.flatten()
    for i, image in enumerate(imgs):
        image = cv2.imread(image)
        image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        ax[i].set_title("IMAGE")
        ax[i].imshow(image)
def generate_sample_masks(sample):
    masks = sample.masks
    _, ax = plt.subplots(1, 5, figsize=(15,3))
    ax = ax.flatten()
    for i, mask in enumerate(masks):
        mask = cv2.imread(mask, cv2.IMREAD_GRAYSCALE) / 255.0
        ax[i].set_title("GROUND TRUTH")
        ax[i].imshow(mask, cmap='gray')
generate_sample_images(sample)
generate_sample_masks(sample)
```

OUTPUT:



RESULT:

Thus the program has been completed and verified successfully.