EX. no. 2 **Title: Black Dot Segmentation Using U-Net and Classical Image Processing**

Aim:

To segment black dots in grayscale images using classical image processing techniques and create a U-Net architecture for potential learning-based segmentation.

Procedure:

1. **Define and Build U-Net Model:**
   - Design a simplified U-Net model with an encoder, bottleneck, and decoder structure.
   - Use convolutional and transposed convolutional layers with ReLU activations.
   - Output a binary mask using a sigmoid-activated convolutional layer.
2. **Compile the U-Net Model:**
   - Use the Adam optimizer.
   - Set the loss function to binary cross-entropy for binary segmentation.
3. **Classical Image Processing for Black Dot Segmentation:**
   - Load the image in grayscale format.
   - Apply Gaussian Blur to reduce noise.
   - Perform Otsu's thresholding to separate the black dots from the background.
   - Use morphological closing (dilation followed by erosion) to remove small noise artifacts and enhance the black dot regions.
4. **Display Results:**
   - Plot the original grayscale image.

- Plot the resulting binary segmentation mask highlighting the black dots.

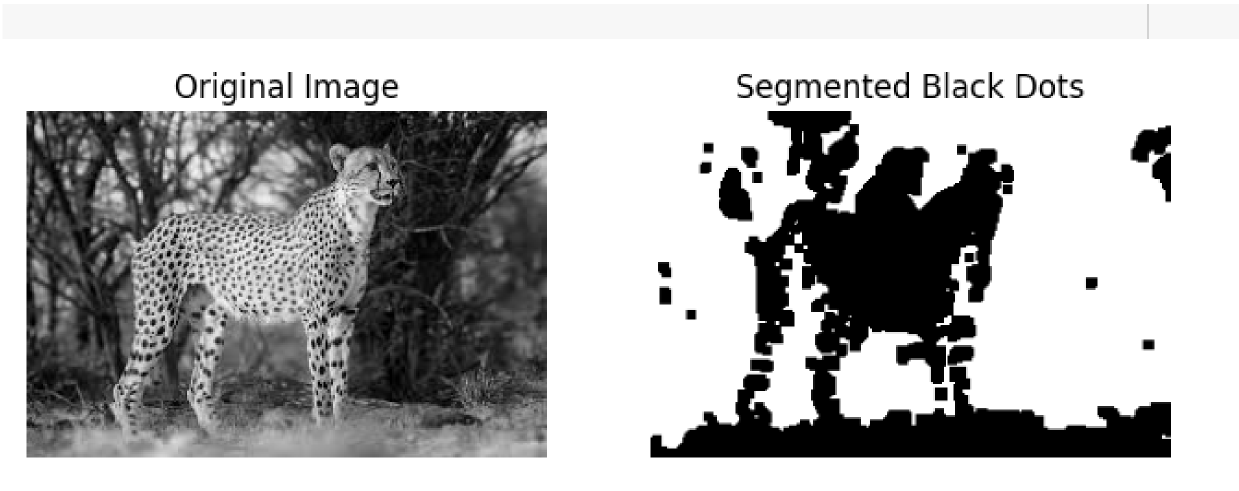
Code:

```
import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import layers

import numpy as np

import cv2

import matplotlib.pyplot as plt


# Define U-Net Model

def build_unet(input_shape=(128, 128, 1)):

    inputs = keras.Input(shape=input_shape)


    # Encoder

    x = layers.Conv2D(64, (3, 3), activation='relu', padding='same')(inputs)

    x = layers.MaxPooling2D((2, 2))(x)


    x = layers.Conv2D(128, (3, 3), activation='relu', padding='same')(x)

    x = layers.MaxPooling2D((2, 2))(x)


    # Bottleneck

    x = layers.Conv2D(256, (3, 3), activation='relu', padding='same')(x)
```

```
    # Decoder
    x = layers.Conv2DTranspose(128, (3, 3), strides=(2, 2), activation='relu', padding='same')(x)
    x = layers.Conv2DTranspose(64, (3, 3), strides=(2, 2), activation='relu', padding='same')(x)

    outputs = layers.Conv2D(1, (1, 1), activation='sigmoid')(x)

    model = keras.Model(inputs, outputs)
    return model

# Create and Compile Model
model = build_unet()
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Function to Predict Segmentation Mask
def segment_black_dots(image_path):
    img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

    # Gaussian Blur
    blurred = cv2.GaussianBlur(img, (5, 5), 0)

    # Otsu's Thresholding
    _, binary_mask = cv2.threshold(blurred, 0, 255, cv2.THRESH_BINARY_INV +
cv2.THRESH_OTSU)

    # Morphological Closing
```

```
    kernel = np.ones((3, 3), np.uint8)
    binary_mask = cv2.morphologyEx(binary_mask, cv2.MORPH_CLOSE, kernel, iterations=2)

    # Display Original and Segmented Images
    fig, axes = plt.subplots(1, 2, figsize=(8, 4))
    axes[0].imshow(img, cmap='gray')
    axes[0].set_title("Original Image")
    axes[0].axis("off")

    axes[1].imshow(binary_mask, cmap='gray')
    axes[1].set_title("Segmented Black Dots")
    axes[1].axis("off")

    plt.show()

# Example Usage
image_path = "/content/cheetah.jpg"
segment_black_dots(image_path)
```

Output:



Result:

Successfully segmented black dots from the grayscale image using Gaussian Blur, Otsu's thresholding, and morphological operations. A U-Net model was also defined for potential learning-based approaches.