**کد شبیه سازی مقاله و فراخوانی دیتاست**

**نرگس خاتمی**

import cv2

import numpy as np

import tkinter as tk

from tkinter import filedialog

from PIL import ImageTk, Image

# Image enhancement functions

def histogram\_equalization(img):

return cv2.equalizeHist(img)

def clahe(img):

clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))

return clahe.apply(img)

def bpdhe(img):

hist, bins = np.histogram(img.flatten(), 256, [0, 256])

cdf = hist.cumsum()

cdf\_normalized = cdf \* hist.max() / cdf.max()

# Calculate the histogram equalization transform

lut = np.interp(np.arange(256), bins[:-1], cdf\_normalized).astype(np.uint8)

# Apply the transform to the input image

equalized\_img = cv2.LUT(img, lut)

return equalized\_img

def agcwd(img, gamma=1, alpha=0.5, beta=0.25):

# Calculate the image gradient

gradient\_x = cv2.Sobel(img, cv2.CV\_64F, 1, 0, ksize=3)

gradient\_y = cv2.Sobel(img, cv2.CV\_64F, 0, 1, ksize=3)

gradient\_magnitude = cv2.magnitude(gradient\_x, gradient\_y)

# Calculate the adaptive gamma correction weights

weight = np.power(gradient\_magnitude, alpha)

weight = cv2.normalize(weight, None, 0, 255, cv2.NORM\_MINMAX)

weight = np.power(weight, beta)

# Apply gamma correction to the input image

img\_gamma = np.power(img / 255.0, gamma)

# Apply the weighting distribution to the gamma-corrected image

img\_corrected = cv2.multiply(img\_gamma, weight)

# Normalize the corrected image to the range [0, 255]

img\_corrected = cv2.normalize(img\_corrected, None, 0, 255, cv2.NORM\_MINMAX)

img\_corrected = img\_corrected.astype(np.uint8)

return img\_corrected

def fcce(img, alpha=0.25, beta=0.75, gamma=2.5, epsilon=1e-5):

# Convert the input image to floating-point format

img = img.astype(np.float32) / 255.0

# Calculate the local mean and standard deviation

local\_mean = cv2.blur(img, (3, 3))

local\_std = cv2.sqrt(cv2.blur(img \* img, (3, 3)) - local\_mean \* local\_mean)

# Calculate the contrast gain function

contrast\_gain = alpha + beta \* (local\_std / (local\_std + epsilon))

# Apply the contrast gain to enhance the image

enhanced\_img = np.power(img, gamma) \* contrast\_gain

# Normalize the enhanced image to the range [0, 255]

enhanced\_img = cv2.normalize(enhanced\_img, None, 0, 255, cv2.NORM\_MINMAX)

enhanced\_img = enhanced\_img.astype(np.uint8)

return enhanced\_img

def ecs(img, population\_size=10, max\_iterations=100, pa=0.25, alpha=1.5, sigma=0.2):

# Convert the input image to floating-point format

img = img.astype(np.float32) / 255.0

# Initialize the cuckoo population with random solutions

population = np.random.rand(population\_size, \*img.shape) \* 255.0

for iteration in range(max\_iterations):

print(f"Iteration {iteration+1}/{max\_iterations}")

# Evaluate the fitness of the population

fitness = fitness\_function(population, img)

# Sort the population based on fitness in descending order

sorted\_indices = np.argsort(fitness)[::-1]

population = population[sorted\_indices]

fitness = fitness[sorted\_indices]

# Generate new solutions using Levy flights

new\_population = levy\_flight(population, alpha, sigma)

# Evaluate the fitness of new solutions

new\_fitness = fitness\_function(new\_population, img)

# Replace low-fitness solutions with new solutions

num\_replacements = int(pa \* population\_size)

replace\_indices = np.random.choice(population\_size, size=num\_replacements, replace=False)

population[replace\_indices] = new\_population[replace\_indices]

fitness[replace\_indices] = new\_fitness[replace\_indices]

# Return the best solution (cuckoo) as the enhanced image

best\_solution = population[0]

enhanced\_img = np.clip(best\_solution, 0, 255).astype(np.uint8)

return enhanced\_img

# Fitness function for ECS (example: mean squared error)

def fitness\_function(population, img):

errors = np.mean((population - img) \*\* 2, axis=(1, 2))

return -errors

# Levy flight function for generating new solutions

def levy\_flight(population, alpha, sigma):

levy = np.random.power(alpha, size=population.shape)

levy = np.sign(np.random.randn(\*population.shape)) \* sigma \* (levy \*\* (-1 / alpha))

new\_population = population + levy

new\_population = np.clip(new\_population, 0, 255)

return new\_population

# GUI functions

def upload\_image():

file\_path = filedialog.askopenfilename(filetypes=(("Image files", "\*.jpg;\*.jpeg;\*.png"), ("All files", "\*.\*")))

if file\_path:

global original\_img\_gray, original\_img\_tk

img = cv2.imread(file\_path)

original\_img\_gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

original\_img\_pil = Image.fromarray(img)

original\_img\_tk = ImageTk.PhotoImage(original\_img\_pil)

input\_label.config(image=original\_img\_tk)

input\_label.image = original\_img\_tk

def enhance\_image():

selected\_method = method\_var.get()

if selected\_method == "Histogram Equalization":

processed\_img = histogram\_equalization(original\_img\_gray)

elif selected\_method == "CLAHE":

processed\_img = clahe(original\_img\_gray)

elif selected\_method == "BPDHE":

processed\_img = bpdhe(original\_img\_gray)

elif selected\_method == "AGCWD":

processed\_img = agcwd(original\_img\_gray)

elif selected\_method == "FCCE":

processed\_img = fcce(original\_img\_gray)

elif selected\_method == "ECS":

processed\_img = ecs(original\_img\_gray)

return

# Call the display\_processed\_image() function with the processed image and title

display\_processed\_image(processed\_img, selected\_method)

processed\_img\_label = None

processed\_img\_rgb = cv2.cvtColor(processed\_img, cv2.COLOR\_GRAY2RGB)

processed\_img\_pil = Image.fromarray(processed\_img\_rgb)

processed\_img\_tk = ImageTk.PhotoImage(processed\_img\_pil)

# Display the processed image in the new window

processed\_img\_label.config(image=processed\_img\_tk)

processed\_img\_label.image = processed\_img\_tk

processed\_img\_label = tk.Label(output\_window, image=processed\_img\_tk)

processed\_img\_label.pack()

# Create the main window

window = tk.Tk()

window.title("Image Enhancement - Created by Narges Khatami")

# Upload image button

upload\_button = tk.Button(window, text="Upload Image", command=upload\_image)

upload\_button.pack()

# Original image display

original\_img\_gray = None

original\_img\_tk = None

input\_label = tk.Label(window)

input\_label.pack()

# Enhancement method selection

method\_var = tk.StringVar(window)

method\_var.set("Histogram Equalization")

method\_options = ["Histogram Equalization", "CLAHE", "BPDHE", "AGCWD", "FCCE", "ECS"]

method\_menu = tk.OptionMenu(window, method\_var, \*method\_options)

method\_menu.pack()

# Apply enhancement button

enhance\_button = tk.Button(window, text="Enhance", command=enhance\_image)

enhance\_button.pack()

# Processed image display

output\_label = tk.Label(window)

output\_label.pack()

def display\_processed\_image(processed\_img, title):

processed\_window = tk.Toplevel()

processed\_window.title(title)

processed\_img\_rgb = cv2.cvtColor(processed\_img, cv2.COLOR\_GRAY2RGB)

processed\_img\_pil = Image.fromarray(processed\_img\_rgb)

processed\_img\_tk = ImageTk.PhotoImage(processed\_img\_pil)

processed\_img\_label = tk.Label(processed\_window, image=processed\_img\_tk)

processed\_img\_label.pack()

processed\_window.mainloop()

# Start the GUI main loop

window.mainloop()