NEURAL NETWORK PROJECT

ABOUT:

THE PROJECT ENTAILS CREATING AN IMAGE CLASSIFICATION SYSTEM USING NEURAL NETWORKS TO IDENTIFY CATS OR DOGS IN IMAGES. IT TRAINS ON PRE-CLASSIFIED IMAGES, ALLOWING USERS TO CLASSIFY NEW IMAGES AND DISPLAYS RESULTS WITH EXPLANATIONS. THE SYSTEM ALSO CHECKS IMAGE ORTHOGONALITY FOR RELIABILITY AND CALCULATES ACCURACY, COMPARING EXPECTED AND ACTUAL CLASSIFICATIONS.

- **PROJECT AMMING LANGUAGE:** PYTHON
- **NEURAL NETWORK:** PSUDOINVERSE
- **SUPERVIDIOP:** DR/ GHADA ELTAWEEL

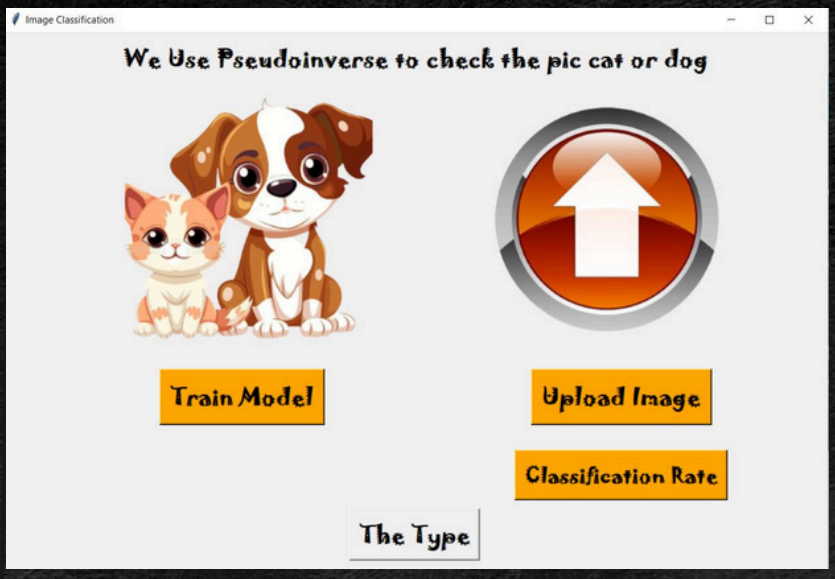
PREPARED BY :

- MOHAMED NABIH EID
- NADA KENAWY YEHIA
- MENNAALLAH ZAKARIA MAHFOUZ
- ESRAA ABDELNASSER KHALEL
- NOURAN ABDO NOUH
- MENNAALLAH OMAR ELSAYED
- MAYADA MOHY ELDEEN SOLIMAN
- HABEBA AHMED ELSAYED

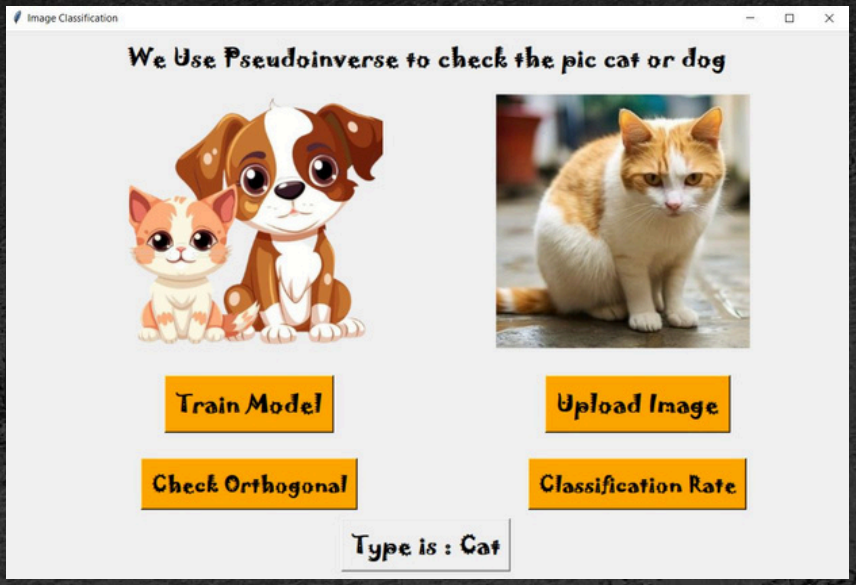
GUI

First Window

After Train Model

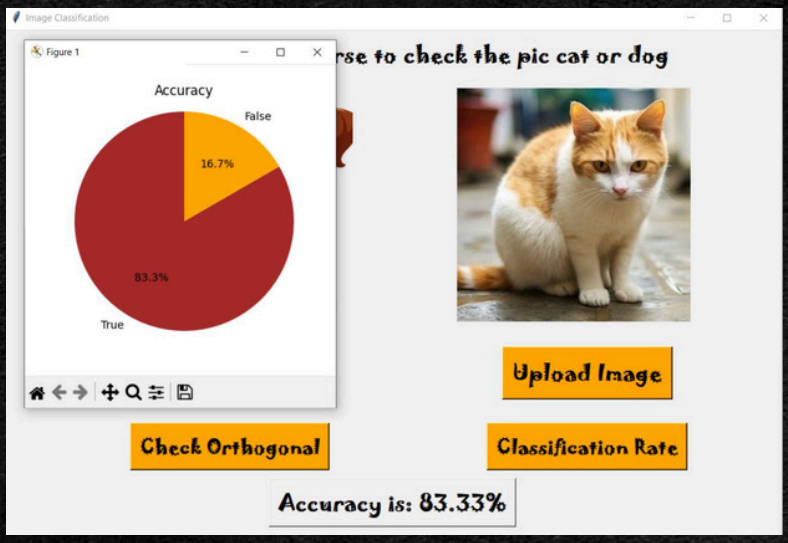


After Upload Image

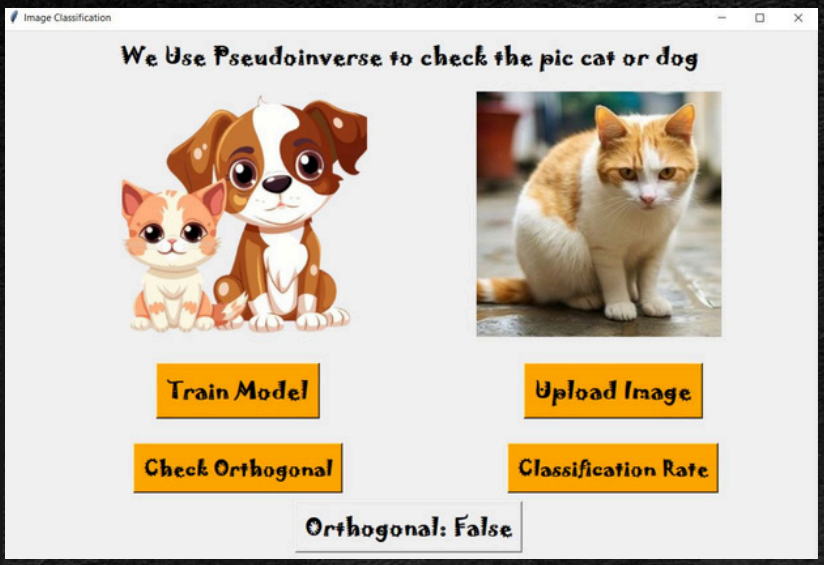


GUI

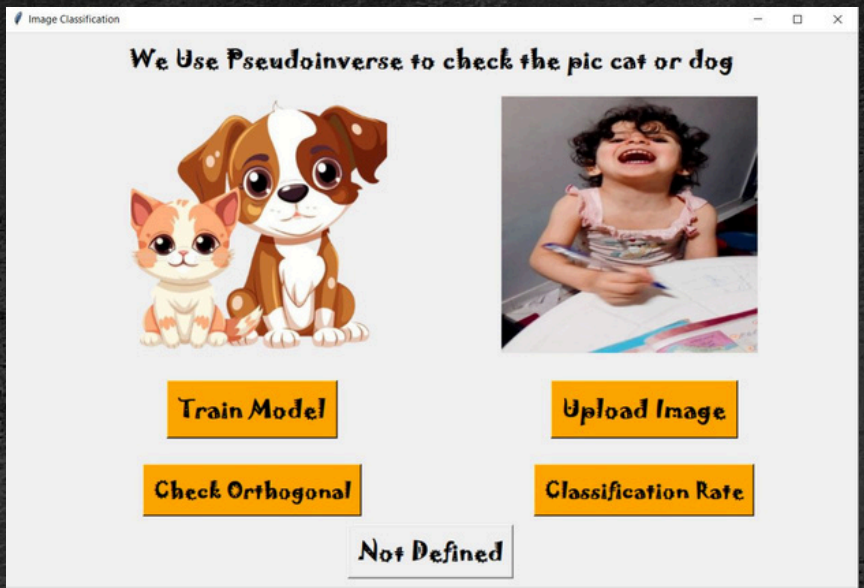
Classification Rate



Check Orthogonal



Upload Unidentified Image



CODE

```python
import tkinter as tk
import tkinter.filedialog as tkFileDialog
from PIL import Image, ImageTk
import numpy as np
import cv2
import matplotlib.pyplot as plt

# Define global variables
P = []  # Training data (input vectors)
T = []  # Training labels (target vectors)
weights = np.array([])  # Weights matrix
image = None
status_text = "The Type"

def open_image():
    global image
    path = tkFileDialog.askopenfilename(filetypes=[("Image Files", ".jpg .png .gif")])
    if path:
        im = Image.open(path)
        tkimage = ImageTk.PhotoImage(im)
        image = tkimage
        image_label.config(image=tkimage)
        classification(path)
```

```python
def accuracy():
    global weights
    inp = []
    counter = 0
    for i in range(1, 7):
        inp=get_features(cv2.imread(f"test2/{i}.jpg", cv2.IMREAD_GRAYSCALE))
        inp = np.array((inp))
        inp = inp.T
        n = np.dot(weights, inp)
        counter +=  1 if (n >= 0 and i <= 3 ) or (n < 0 and i >3 ) else 0

    accuracy_percentage = round((counter/6.0)*100,2)
    text2 = f"Accuracy is: {accuracy_percentage}%"
    status_label.config(text=text2)
    # Plotting pie chart
    labels = ['True', 'False']
    sizes = [accuracy_percentage, 100 - accuracy_percentage]
    colors = ['brown', 'orange']  # تحديد الألوان لكل قطعة بالترتيب
    plt.figure(figsize=(4, 4))
    plt.pie(sizes, labels=labels, autopct='%2.1f%%', startangle=90 ,colors=colors)
    plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
    plt.title('Accuracy')
    plt.show()
```

```python
def start():
    global weights, T, P
    for i in range(10):
        P.append(get_features(cv2.imread(f"data2/cat.{i}.jpg", cv2.IMREAD_GRAYSCALE)))
        T.append(1)
        P.append(get_features(cv2.imread(f"data2/dog.{i}.jpg", cv2.IMREAD_GRAYSCALE)))
        T.append(-1)
    P = np.array(P) # Convert the list to numpy array
    T = np.array(T) # Convert the list to numpy array
    weights = np.dot(T, np.dot(np.linalg.inv(np.dot(P, P.T)), P))
    print("the train is done")
    check_classification_rate()
    # Create button to select and classify image
    button_select = tk.Button(button_frame, text="Upload Image", command=lambda: open_image(), background="orange", font=("Jokerman", 20, "bold"))
    button_select.grid(row=0, column=1, padx=100, pady=20)
    button_classification = tk.Button(button_frame, text="Classification Rate", command=lambda: accuracy(), background="orange", font=("Jokerman", 18, "bold"))
    button_classification.grid(row=1, column=1, padx=20, pady=10)

def get_features(image):
    new = conv_relu(image)
    new = pooling(new)
    new = conv_relu(new)
    new = pooling(new)
    new = conv_relu(new)
    new = pooling(new)
    new = conv_relu(new)
    new = pooling(new)
    new = conv_relu(new)
    new = pooling(new)
    new = flatten(new)
    return new
```

CODE

```python
def conv_relu(image):
    mask = [[-1, -1, 1], [0, 1, -1], [0, 1, 1]]
    size1 = len(image) - 2
    size2 = len(image[0]) - 2
    new_image = [[0 for _ in range(size2)] for _ in range(size1)]
    for i in range(size1):
        for j in range(size2):
            x = 0
            for k in range(3):
                x += (image[i + k][j + 0] * mask[k][0] + image[i + k][j + 1] * mask[k][1] + image[i + k][j + 2] * mask[k][2])
            new_image[i][j] = x if x > 0 else 0
    return new_image

def pooling(image):
    size1 = int(len(image) / 2)
    size2 = int(len(image[0]) / 2)
    new_image = [[0 for _ in range(size2)] for _ in range(size1)]
    for i in range(0, size1):
        for j in range(0, size2):
            x = 0
            for k in range(2):
                x += (image[(i * 2) + k][(j * 2) + 0] + image[(i * 2) + k][(j * 2) + 1]) / 4
            new_image[i][j] = int(x)
    return new_image

def flatten(image):
    new_image = []
    for row in image:
        for el in row:
            new_image.append(el)
    return new_image
```

```python
def minDistance(p):
    global P
    minDistance = -1
    for input in P:
        dis = np.linalg.norm(np.array(p) - np.array(input))
        if minDistance == -1:
            minDistance = dis
        elif dis < minDistance:
            minDistance = dis

    return minDistance

def classification(path):
    global image, weights, text
    p = np.array(get_features(cv2.imread(path, cv2.IMREAD_GRAYSCALE)))
    distance = minDistance(p)
    if distance > 1000:
        text = "Not Defined"
    else:
        p = p.T
        n = np.dot(weights, p)
        text = "Type is : Cat" if n >= 0 else "Type is : Dog"
    status_label.config(text=text)
    # Create buttons to check orthogonal and classification rate
    button_orthogonal = tk.Button(button_frame, text="Check Orthogonal", command=lambda: check_orthogonal(), background="orange", font=("Jokerman", 18, "bold"))
    button_orthogonal.grid(row=1, column=0, padx=20, pady=10)

```

```python
def check_orthogonal():
    global P
    orthogonal = np.allclose(np.dot(P, P.T), np.eye(len(P)), atol=1e-8)
    status_label.config(text=f"Orthogonal: {orthogonal}")

def check_classification_rate():
    global weights, P, T
    predictions = np.dot(weights, P.T)
    accuracy = np.mean(np.sign(predictions) == T)
    print(f"Classification Rate: {accuracy * 100:.2f}%")

# Create a tkinter window
window = tk.Tk()
window.title("Image Classification")
window.geometry("1000x650")

# Create a frame for images
image_frame = tk.Frame(window)
image_frame.pack()

# Create a label for title
title_label = tk.Label(image_frame, text="We Use Pseudoinverse to check the pic cat or dog", font=("Jokerman", 20, "bold"), padx=10, pady=10)
title_label.pack()

```

```python
# Load the second image
image2 = Image.open("DogCat.png")
photo2 = ImageTk.PhotoImage(image2)
image_label2 = tk.Label(image_frame, image=photo2)
image_label2.pack(side="left", padx=80, pady=10)

# Load the first image
image = Image.open("download.png")
photo = ImageTk.PhotoImage(image)
image_label = tk.Label(image_frame, image=photo)
image_label.pack(side="left", padx=50, pady=10)

# Create a frame for the buttons and status label
button_frame = tk.Frame(window)
button_frame.pack()

# Create button to load and train the model
button_train = tk.Button(button_frame, text="Train Model", command=lambda: start(), background="orange", font=("Jokerman", 20, "bold"))
button_train.grid(row=0, column=0, padx=150,pady=20)

# Create a label for status
status_label = tk.Label(button_frame, text=status_text, font=("Jokerman", 20, "bold"), padx=10, pady=10, relief=tk.RAISED, borderwidth=2)
status_label.grid(row=2, columnspan=2)

# Start the tkinter event loop
window.mainloop()
```