IMAGE RECOGNITION AND CLASSIFICATION

INTRODUCTION

The "Image Recognition and Classification" project aims to classify images into predefined categories using convolutional neural networks (CNNs). It consists of two main functionalities: image classification and pneumonia detection. The project utilizes deep learning techniques to achieve high accuracy in image recognition tasks.

Image Classification: The project includes a robust image classification module that can accurately identify objects, scenes, or patterns within images. By training on large datasets, the CNN model can recognize various objects and assign them to specific categories.

Pneumonia Detection: Another critical functionality of the project is pneumonia detection. Using a specialized CNN architecture, the system can analyze chest X-rays and identify signs of pneumonia. Early detection of pneumonia can significantly impact patient outcomes.

OBJECTIVES

- To develop an image recognition system using CNNs.
- To classify images into specific categories with high accuracy.
- To implement a user-friendly interface for easy interaction with the system.
- To provide support for both general image classification and pneumonia detection tasks.

ABSTRACT

This project presents the development of an image recognition system that utilizes deep learning techniques for accurate classification of images. Two CNN models are trained and integrated into a Python application using Tkinter for the graphical user interface. The system allows users to upload images and select between image classification and pneumonia detection functionalities.

REQUIREMENTS

- Jupyter Notebook
- Python
- TensorFlow
- Tkinter
- PIL (Python Imaging Library)
- NumPy

Convolutional Neural Networks

Convolutional Neural Networks (CNNs) represent a specialized class of deep neural networks meticulously designed to excel in image recognition and classification tasks. These revolutionary architectures have reshaped the landscape of computer vision and pattern recognition, enabling unprecedented levels of accuracy and efficiency in analyzing visual data.

Key Components of CNNs:

Convolutional Layers:

The hallmark of CNNs, convolutional layers employ learnable filters or kernels to extract spatial features from input images. Through convolution operations, these layers capture patterns such as edges, textures, and shapes, progressively refining representations as information flows through the network.

Pooling Layers:

Pooling layers play a crucial role in spatial downsampling, reducing the dimensionality of feature maps while preserving essential information. Max pooling and average pooling are common pooling operations employed to extract dominant features and enhance translation invariance.

Activation Functions:

Non-linear activation functions like ReLU (Rectified Linear Unit) introduce non-linearity into the network, enabling CNNs to model complex relationships within image data. Other activation functions such as sigmoid and tanh may also be utilized in certain contexts.

Fully Connected Layers:

Fully connected layers, also known as dense layers, consolidate extracted features from preceding layers and map them to output classes. These layers facilitate high-level abstraction and decision-making, enabling CNNs to classify images into predefined categories.

Training Process:

The training of CNNs typically involves the following steps:

Initialization:

Initialize the weights and biases of the network parameters, often using random or pre-trained weights from existing models.

Forward Propagation: Propagate input images forward through the network, applying successive transformations via convolutional, pooling, and activation layers to extract hierarchical features.

Loss Computation:

Calculate the discrepancy between predicted outputs and ground truth labels using a suitable loss function, such as categorical cross-entropy for classification tasks.

Back-propagation:

Employ back-propagation algorithm to compute gradients of the loss function with respect to network parameters, facilitating parameter updates through optimization algorithms like Stochastic Gradient Descent (SGD) or Adam.

Parameter Update:

Adjust the weights and biases of the network parameters iteratively based on computed gradients, aiming to minimize the loss function and improve model performance.

Applications of CNN's:

- Image Classification
- Object Detection and Localization
- Facial Recognition
- Medical Imaging and Diagnosis
- Autonomous Vehicles
- Natural Language Processing (NLP)
- Robotics

INSTALLATION

Installation of Jupyter Notebook and Required Modules:

1. Install Python on your system.
2. Install Jupyter Notebook using pip:

```
pip install notebook
```

3. Install TensorFlow:

```
pip install tensorflow
```

4. Install Tkinter (usually included in Python distributions).
5. Install PIL and NumPy:

```
pip install pillow numpy
```

WORKFLOW

Data Collection:

The initial phase of the project involved collecting datasets from Kaggle, comprising two distinct sets:

Object Dataset: This dataset encompasses a diverse range of object categories, providing a rich source of imagery for image classification tasks.

Chest X-ray Images Dataset: Specifically curated for pneumonia detection, this dataset comprises chest X-ray images annotated with binary labels indicating the presence or absence of pneumonia

Data Preprocessing:

1) For Image Classification:

```
data_path = './Dataset/256_ObjectCategories'
IMAGE_SIZE = (224, 224)
BATCH_SIZE = 32
EPOCHS = 25

# Creating training and testing data generators
train_data = tf.keras.utils.image_dataset_from_directory(
    data_path,
    shuffle=True,
    image_size=IMAGE_SIZE,
    batch_size=BATCH_SIZE,
    validation_split=False
)

test_data = tf.keras.utils.image_dataset_from_directory(
    data_path,
    shuffle=True,
    image_size=IMAGE_SIZE,
    batch_size=32,
    validation_split=0.30,
    subset="validation",
    seed=89
)
```

2) **For Pneumonia Detection:**

```
train_path = './Dataset/chest_xray/train'
test_path = './Dataset/chest_xray/test'
```

```
IMAGE_SIZE = (180, 180)
BATCH_SIZE = 32
EPOCHS = 20

train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)
train_generator = train_datagen.flow_from_directory(
    train_path,
    target_size=IMAGE_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='binary'
)
test_datagen = ImageDataGenerator(rescale=1./255)

test_generator = test_datagen.flow_from_directory(
    test_path,
    target_size=IMAGE_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='binary'
)
```

Model Training and Saving:

Following data preprocessing, sequential models were constructed and trained for image classification and pneumonia detection tasks. The models were trained using the TensorFlow framework, leveraging the powerful capabilities of convolutional neural networks.

1) **For Image Classification:**

```
cnn_model = Sequential([layers.Rescaling(1./255)])
cnn_model.add(layers.Conv2D(filters = 16, kernel_size =3, padding='same', activation='relu'))
cnn_model.add(layers.MaxPool2D())
cnn_model.add(layers.Conv2D(filters = 32, kernel_size =3, padding='same', activation='relu'))
cnn_model.add(layers.MaxPool2D())
cnn_model.add(layers.Conv2D(filters = 64, kernel_size =3, padding='same', activation='relu'))
cnn_model.add(layers.MaxPool2D())
cnn_model.add(layers.Conv2D(filters = 128, kernel_size =3, padding='same', activation='relu'))
cnn_model.add(layers.MaxPool2D())
cnn_model.add(layers.Flatten())
cnn_model.add(layers.Dropout(0.2))
cnn_model.add(layers.Dense(units=128, activation='relu'))
cnn_model.add(layers.Dense(len(data_category)))
cnn_model.compile(optimizer='adam',
loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),metrics=['accuracy'])
cnn_model.fit(train_data,
              validation_data = test_data,
              epochs=EPOCHS)
```

2) **For Pneumonia Detection:**

```
cnn_model = Sequential()
cnn_model.add(layers.Conv2D(filters = 32, kernel_size =3, padding='same', activation='relu'))
cnn_model.add(layers.MaxPool2D())
cnn_model.add(layers.Conv2D(filters = 64, kernel_size =3, padding='same', activation='relu'))
cnn_model.add(layers.MaxPool2D())
cnn_model.add(layers.Conv2D(filters = 128, kernel_size =3, padding='same', activation='relu'))
cnn_model.add(layers.MaxPool2D())
cnn_model.add(layers.Flatten())
cnn_model.add(layers.Dense(units=512, activation='relu'))
cnn_model.add(layers.Dropout(0.5))
cnn_model.add(layers.Dense(units=1, activation='sigmoid'))
cnn_model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])

cnn_model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // BATCH_SIZE,
    epochs=EPOCHS,
    validation_data=test_generator,
    validation_steps=test_generator.samples // BATCH_SIZE)
```

SOURCE CODE

After successful model training, a user-friendly Python application was developed using the Tkinter library. This application allows users to interactively utilize the trained models for image recognition and classification tasks.

```
# Importing the Libraries
#tkinter module for GUI development
import tkinter as tk
from tkinter import filedialog, messagebox
# PIL library for image processing
from PIL import ImageTk, Image
# numpy for numerical operations
import numpy as np
# TensorF for deep learning tasks
import tensorflow as tf
from keras.preprocessing import image
#custom class names from a module
from resources.class_names import class_1, class_2

# Load pre-trained models
image_classifier_model =
tf.keras.models.load_model('./notebooks/models/image_classifier_model.keras')
pneumonia_detection_model =
tf.keras.models.load_model('./notebooks/models/pneumonia_detection_model.keras')

# image dimensions
IMAGE_SIZE = (180, 180)


# Loading class names
image_classification_names = class_1
pneumonia_detection_names = class_2
```

```
# Function to handle selection of functionalities
def select_function(selection):
    if selection == 1:
        image_classification()
    elif selection == 2:
        pneumonia_detection()

# Function to upload image for image classification
def upload_image_classification():
    filepath = filedialog.askopenfilename(initialdir="./test_cases/inputs/Image_classification/")
    if filepath:
        if check_image_file(filepath):
            display_image_classification(filepath)
        else:
            messagebox.showerror("Error", "Invalid file format. Please select an image file.")

# Function to check if a file is a valid image file
def check_image_file(filepath):
    try:
        with Image.open(filepath):
            return True
    except:
        return False

# Function to open window for image classification
def image_classification():
    window01 = tk.Toplevel(root)
    window01.title("Image Classification")
    window01.geometry("400x200")
    window01.configure(bg="black")
```

```
    bg_image = ImageTk.PhotoImage(Image.open("./resources/images/image_classification_bg.jpg").resize((400, 200)))
    bg_label = tk.Label(window01, image=bg_image)
    bg_label.place(x=0, y=0, relwidth=1, relheight=1)

    upload_button = tk.Button(window01, text="Upload Image", command=upload_image_classification,
                              bg="black", fg="yellow", highlightbackground="black", borderwidth=3,
                              font=("Times New Roman", 12))
    upload_button.place(relx=0.5, rely=0.5, anchor=tk.CENTER)

    exit_button = tk.Button(window01, text="Exit", command=window01.destroy,
                            bg="black", fg="yellow", highlightbackground="black", borderwidth=2,
                            font=("Times New Roman", 10))
    exit_button.place(relx=0.5, rely=0.8, anchor=tk.CENTER)

    window01.mainloop()

# Function to display selected image for image classification
def display_image_classification(filepath):
    image = Image.open(filepath)
    image = image.resize((300, 300))
    photo = ImageTk.PhotoImage(image)

    window11 = tk.Toplevel(root)
    window11.title("Image Display")
    window11.geometry("500x400")
    window11.configure(bg="black")

    image_label = tk.Label(window11, image=photo)
    image_label.image = photo
```

```
    image_label.pack()

    predict_image(filepath, window11)

# Function to predict class of image
def predict_image(filepath, window):
    img_load = tf.keras.utils.load_img(filepath, target_size=IMAGE_SIZE)
    img_arr = tf.keras.utils.img_to_array(img_load)
    img_batch = np.expand_dims(img_arr, 0)

    predict = image_classifier_model.predict(img_batch)
    score = tf.nn.softmax(predict)
    class_name = image_classification_names[np.argmax(score)].split(".")
    class_text = tk.Label(window, text=f"Predicted image as: {class_name[1]}",
                          font=("Times New roman", 13, "bold"), fg="yellow",  bg="black")
    class_text.pack(pady=10)

    exit_button = tk.Button(window, text="Exit", command=window.destroy,
                            bg="black", fg="yellow", highlightbackground="black",
                            borderwidth=2, font=("Times New Roman", 10))
    exit_button.pack()

# Function to upload a image for pneumonia detection
def upload_pneumonia_detection():
    filepath =
filedialog.askopenfilename(initialdir="./test_cases/inputs/pneumonia_detection/")
    if filepath:
        if check_image_file(filepath):
            display_pneumonia_detection(filepath)
        else:
            messagebox.showerror("Error", "Invalid file format. Please select an image
file.")

# Function to open window for pneumonia detection
```

```
def pneumonia_detection():
    window02 = tk.Toplevel(root)
    window02.title("Pneumonia Detection")
    window02.geometry("400x200")
    window02.configure(bg="black")

    bg_image = ImageTk.PhotoImage(Image.open("./resources/images/pneumonia_detection_bg.jpg").resize((400, 200)))
    bg_label = tk.Label(window02, image=bg_image)
    bg_label.place(x=0, y=0, relwidth=1, relheight=1)

    upload_button = tk.Button(window02, text="Upload Image", command=upload_pneumonia_detection,
                              bg="black", fg="yellow", highlightbackground="black", borderwidth=3,
                              font=("Times New Roman", 12))
    upload_button.place(relx=0.5, rely=0.5, anchor=tk.CENTER)

    exit_button = tk.Button(window02, text="Exit", command=window02.destroy,
                            bg="black", fg="yellow", highlightbackground="black", borderwidth=2,
                            font=("Times New Roman", 10))
    exit_button.place(relx=0.5, rely=0.8, anchor=tk.CENTER)

    window02.mainloop()

# Function to display selected image for pneumonia detection
def display_pneumonia_detection(filepath):
    image = Image.open(filepath)
    image = image.resize((300, 300))
    photo = ImageTk.PhotoImage(image)

    window22 = tk.Toplevel(root)
```

```
    window22.title("Image Display")
    window22.geometry("500x400")
    window22.configure(bg="black")

    image_label = tk.Label(window22, image=photo)
    image_label.image = photo
    image_label.pack()

    predict_pneumonia(filepath, window22)

# Function to predict the pneumonia
def predict_pneumonia(filepath, window):
    img_load = image.load_img(filepath, target_size=IMAGE_SIZE)
    img_arr = image.img_to_array(img_load)
    img_batch = np.expand_dims(img_arr, axis=0)

    predict = pneumonia_detection_model.predict(img_batch)
    class_name = pneumonia_detection_names[int(np.round(predict[0][0]))]
    class_text = tk.Label(window, text=f"Predicted as: {class_name}",
                          font=("Times New roman", 13, "bold"), fg="yellow",  bg="black")
    class_text.pack(pady=10)

    exit_button = tk.Button(window, text="Exit", command=window.destroy,
                            bg="black", fg="yellow", highlightbackground="black",
                            borderwidth=2, font=("Times New Roman", 10))
    exit_button.pack()

# Creating main root window for GUI
root = tk.Tk()
root.title("Image Recognition System")
root.geometry("400x200")
```

```
background_image = ImageTk.PhotoImage(Image.open("./resources/images/root_bg.jpg").resize((400, 200)))
canvas = tk.Canvas(root, width=400, height=200)
canvas.pack(fill="both", expand=True)
canvas.create_image(0, 0, anchor=tk.NW, image=background_image)

var = tk.IntVar()
radio_button1 = tk.Radiobutton(root, text="Image Classification", variable=var, value=1)
radio_button1.place(relx=0.5, rely=0.3, anchor=tk.CENTER)

radio_button2 = tk.Radiobutton(root, text="Pneumonia Detection", variable=var, value=2)
radio_button2.place(relx=0.5, rely=0.5, anchor=tk.CENTER)

select_button = tk.Button(root, text="Select", command=lambda: select_function(var.get()),bg="black", fg="yellow",
                          highlightbackground="black", borderwidth=2, font=("Times New Roman", 12))
select_button.place(relx=0.5, rely=0.7, anchor=tk.CENTER)

exit_button = tk.Button(root, text="Exit", command=root.destroy, bg="black", fg="yellow",
                        highlightbackground="black", borderwidth=2, font=("Times New Roman", 10))
exit_button.place(relx=0.5, rely=0.9, anchor=tk.CENTER)
root.mainloop()
```

The application provides two main functionalities: image classification and pneumonia detection. Users can upload images through an intuitive graphical user interface (GUI), and the application utilizes the trained models to predict the respective classes or diagnoses.

SNAPSHOTS OF PROJECT

root window:

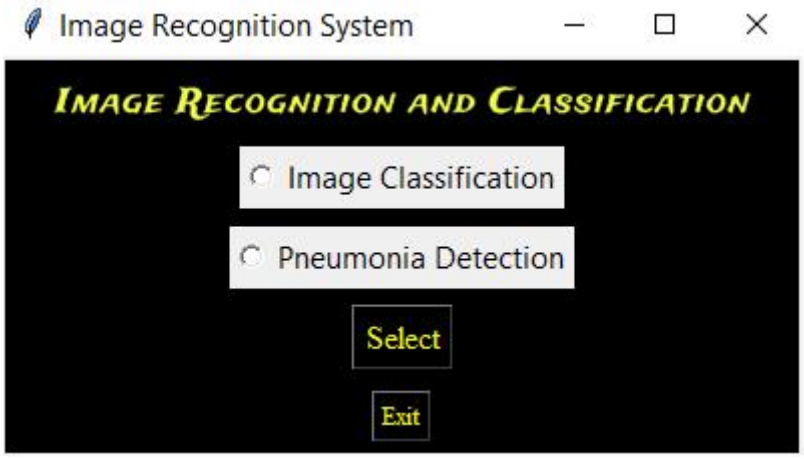
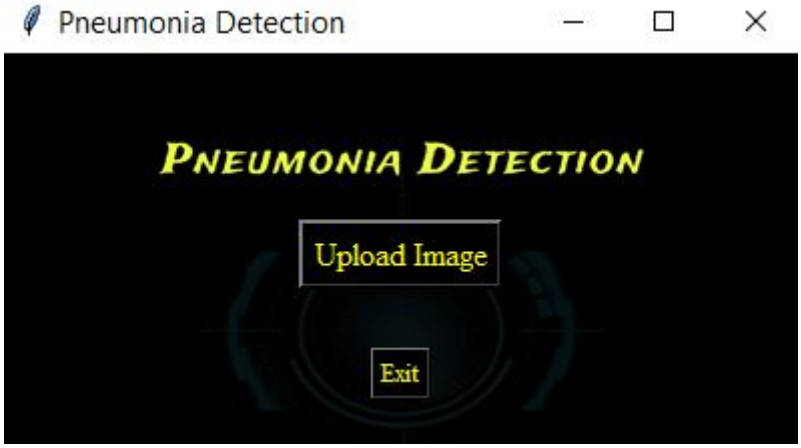


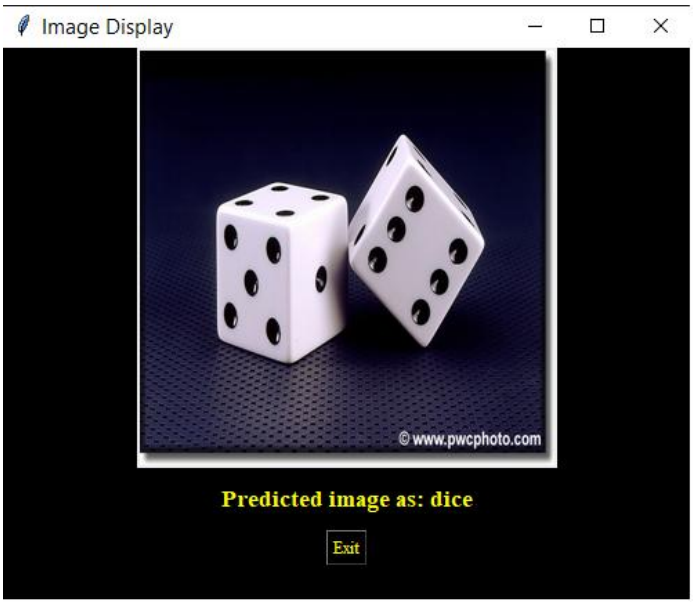
Image classification window:

Pneumonia Detection:



Output images:

1) Image Classification output:



2) Pneumonia Detection output: