1. Abstract

- **Problem Statement**

– **Ensuring compliance with mask-wearing is critical to limit the spread of airborne diseases like COVID-19.**

– **Manual monitoring is labor-intensive and prone to human error.**

– **An automated, real-time solution is needed to improve public health and safety.**

- **Proposed Method**

– **A Convolutional Neural Network (CNN)-based model was developed.**

– **Trained on a balanced dataset (with mask vs. without mask).**

– **Images were preprocessed (resized and normalized).**

– **The model includes convolutional, pooling, and dense layers for feature extraction and classification.**

- **Key Results & Contributions**

– **Training accuracy: 99.01%**

– **Validation accuracy: 93.38%**

– **Performance validated via confusion matrix and classification report.**

– **Lightweight and accurate model suitable for real-time deployment in surveillance and access control systems.**

2. Introduction

- **Background & Motivation**
- **COVID-19 increased the need for mask compliance in public spaces.**
- **Manual monitoring is slow, costly, and error-prone.**
- **Automated solutions can enhance safety and reduce human workload.**
- **Problem Statement**
- **Existing methods lack real-time accuracy or scalability.**
- **This work proposes a CNN-based system to automatically detect face mask usage from images.**

3. Related Work

- **Traditional ML methods (e.g., SVM, Decision Trees) required manual feature extraction.**
- **CNNs became dominant due to their ability to learn features automatically.**
- **Models like MobileNetV2 and ResNet have been used but are often too heavy for real-time use.**
- **This study proposes a lightweight CNN suitable for real-time, edge-device deployment.**

4. Methodology

- **Input: 100×100 RGB images**
- **Model Structure:**
- **Conv2D (32 filters, 3×3) → ReLU → MaxPooling (2×2)**
- **Conv2D (64 filters, 3×3) → ReLU → MaxPooling (2×2)**
- **Flatten → Dense (128) → ReLU → Dense (2) → Softmax**
- **Loss Function: Categorical Crossentropy**
- **Optimizer: Adam (IR = 0.001)**
- **Training: 10 epochs, batch size = 32**
- **Tools: TensorFlow/Keras on GPU (e.g., Google Colab)**

5. Experiments

➢ Dataset

Custom-labeled dataset with two classes: with_mask and without_mask.

Data split: 80% training, 20% validation.

Input images resized to 100×100 pixels and normalized.

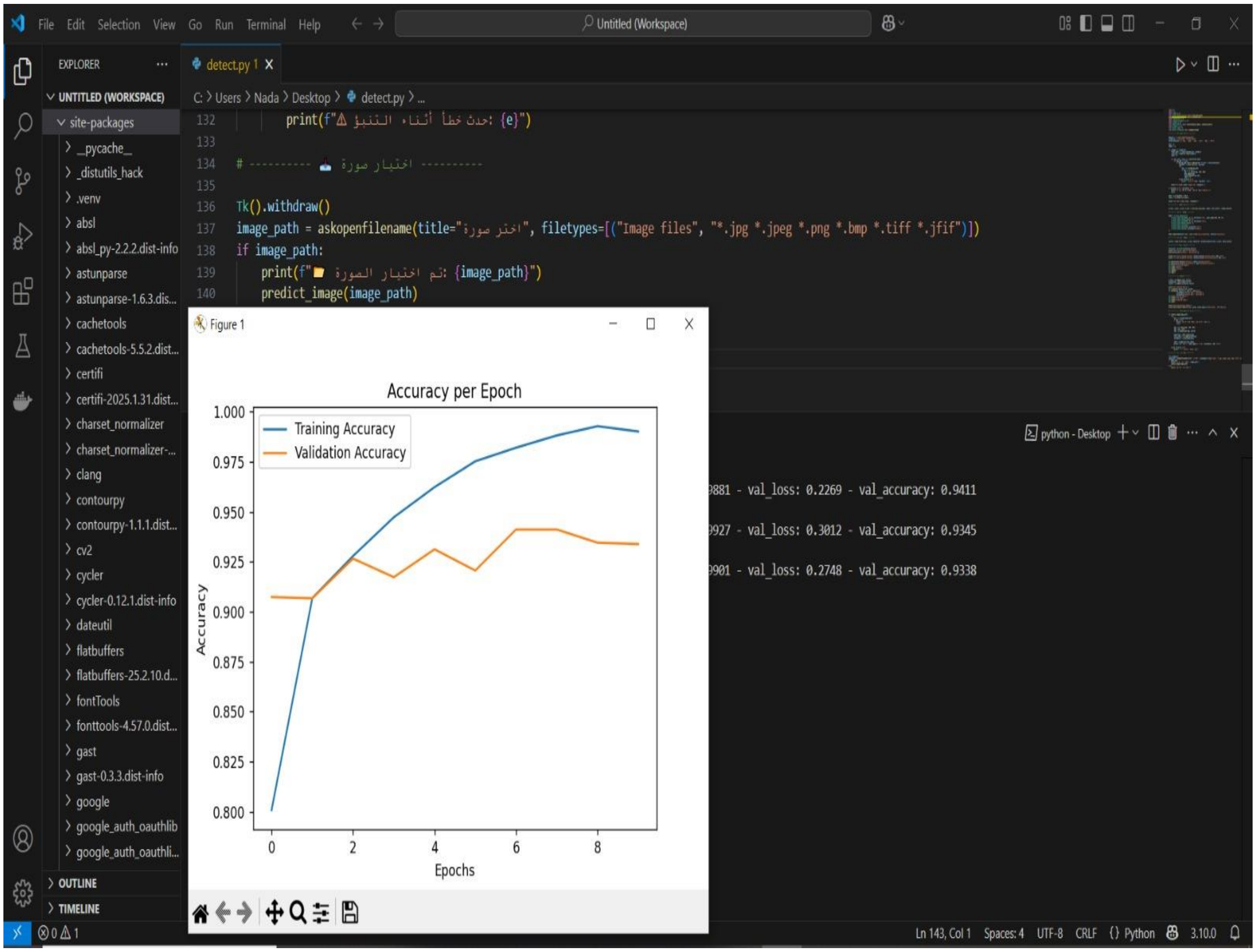
➢ Baseline

1. **Compared against logistic regression model.**
2. **Baseline performance: ~70% accuracy.**
3. **Highlighted need for more powerful deep learning models.**

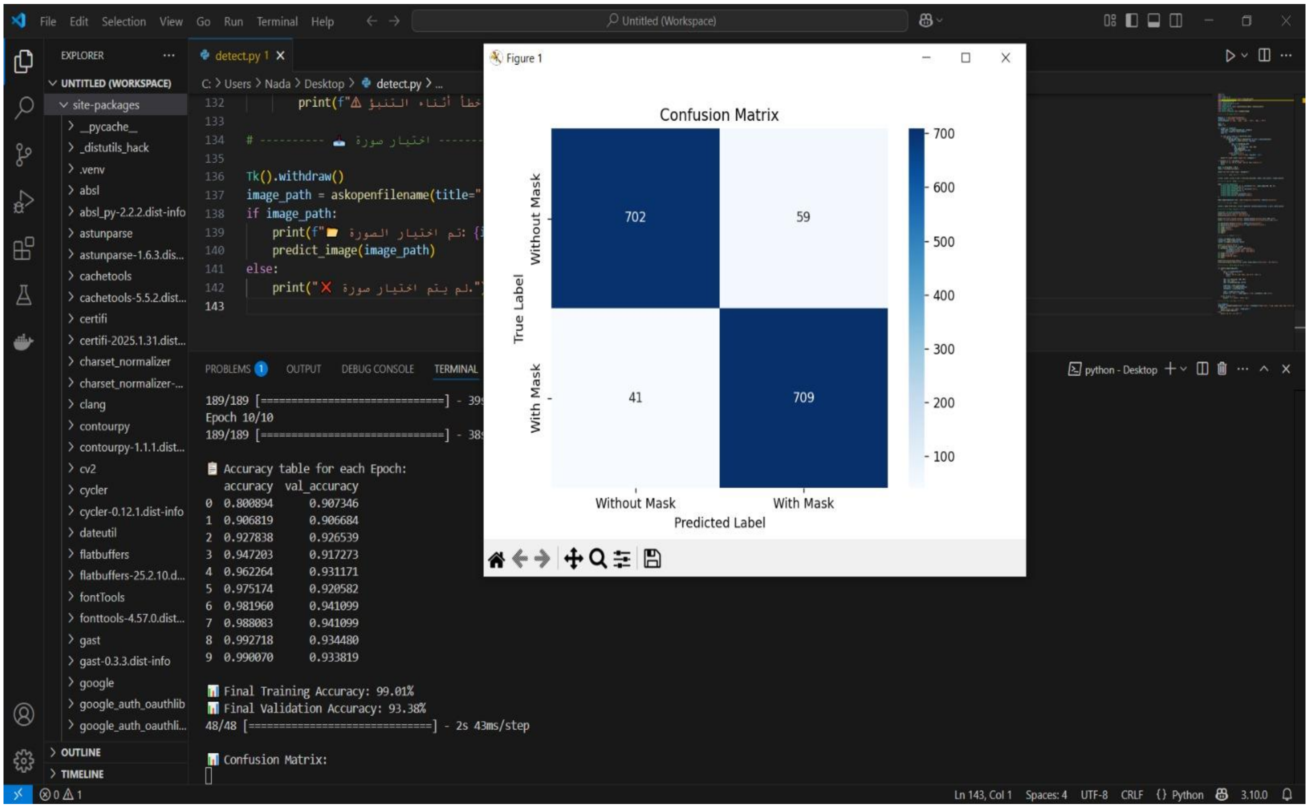
➢ Evaluation Metrics

- ☐ **Accuracy**
- ☐ **Precision**
- ☐ **Recall**

❑ F1-score


File Edit Selection View Go Run Terminal Help
Untitled (Workspace)
EXPLORER
UNTITLED (WORKSPACE)
site-packages
detect.py 1
C: > Users > Nada > Desktop > detect.py > ...
Tk().withdraw()
if image_path:
predict_image(image_path)
Figure 1
Accuracy per Epoch
Training Accuracy
Validation Accuracy
Accuracy
Epochs
val_loss: 0.2269 - val_accuracy: 0.9411
val_loss: 0.3012 - val_accuracy: 0.9345
val_loss: 0.2748 - val_accuracy: 0.9338
python - Desktop
Ln 143, Col 1 Spaces: 4 UTF-8 CRLF {} Python 3.10.0

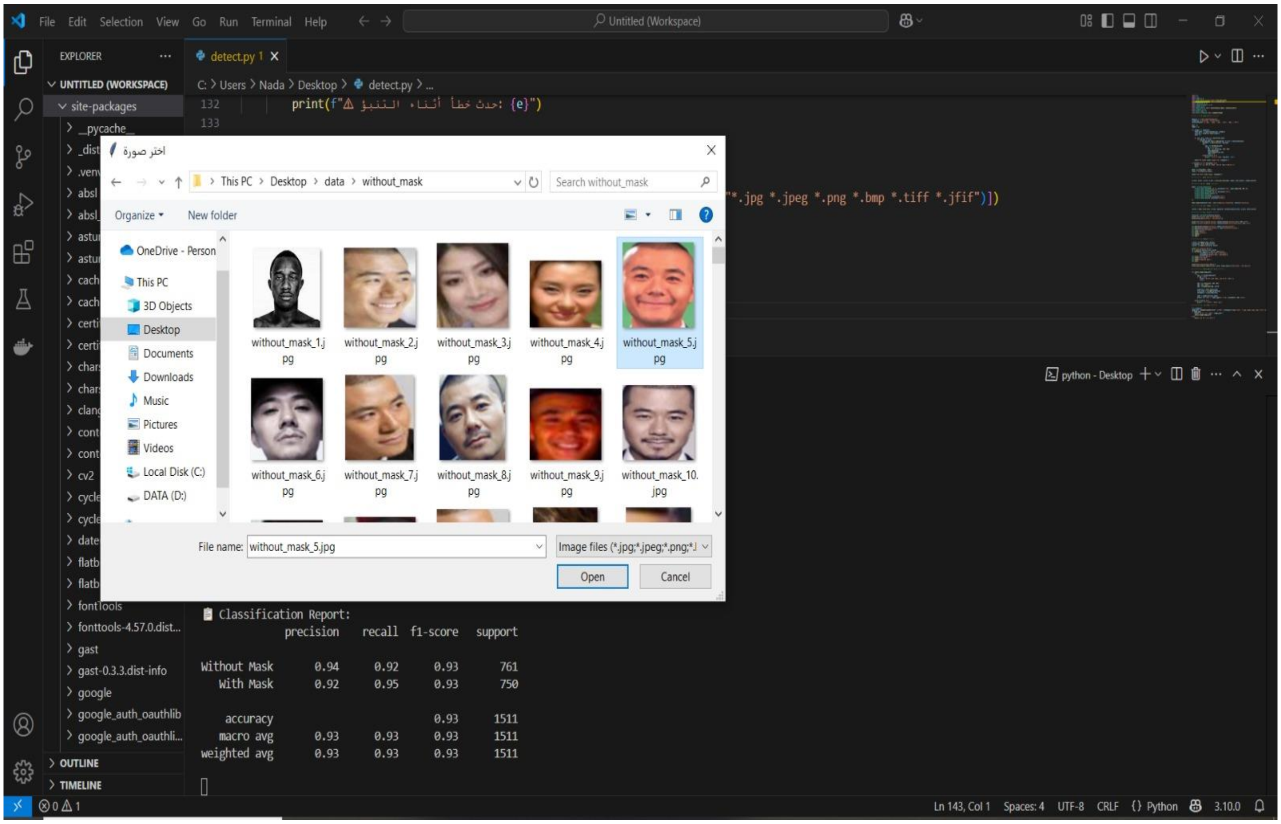
Confusion Matrix (Confusion Matrix Heatmap)



- ➢ **Implementation**

- **Programming Language: Python**
- **Framework: TensorFlow / Keras**

- **GUI: Tkinter used for image selection and prediction (GUI interface)**

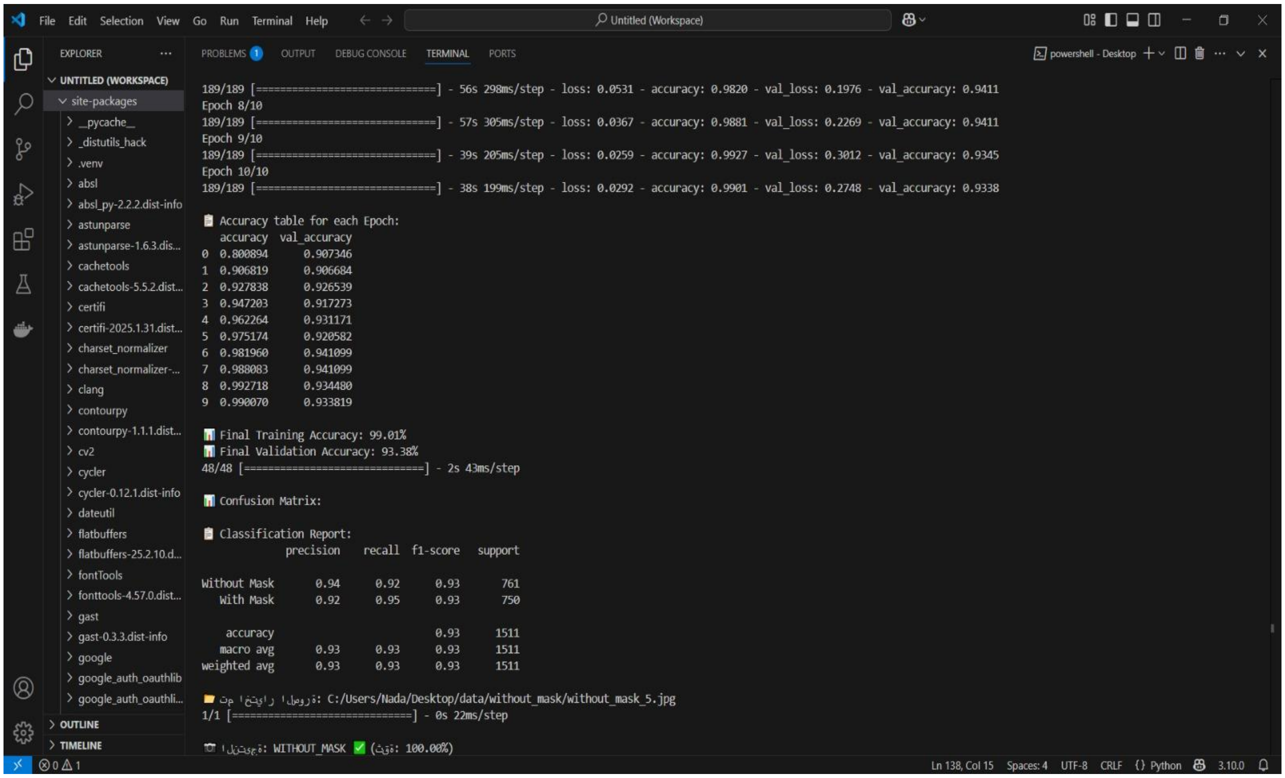


- **Environment: GPU-enabled (Google Colab)**

➢ **Training Configuration**

– **Optimizer: Adam, learning rate = 0.001**
– **Loss Function: Categorical Crossentropy**
– **Epochs: 10**
– **Batch Size: 32**

– **Visualization of accuracy per epoch (Accuracy vs. Epochs plot)**



➢ Results

- **Training Accuracy: 99.01%**
- **Validation Accuracy: 93.38%**
- **Confusion Matrix:**
- **With Mask: 709 correct, 41 incorrect**
- **Without Mask: 702 correct, 59 incorrect**
- **Achieved strong results with low computational cost.**

6. Discussion (Model Performance)

- **Performs well on validation data with high accuracy.**
- **Detects mask usage across different faces and conditions.**

➢ Limitations

- **Dataset lacks diversity in lighting and ethnicity.**
- **May struggle in more complex real-world settings.**

➢ Implications

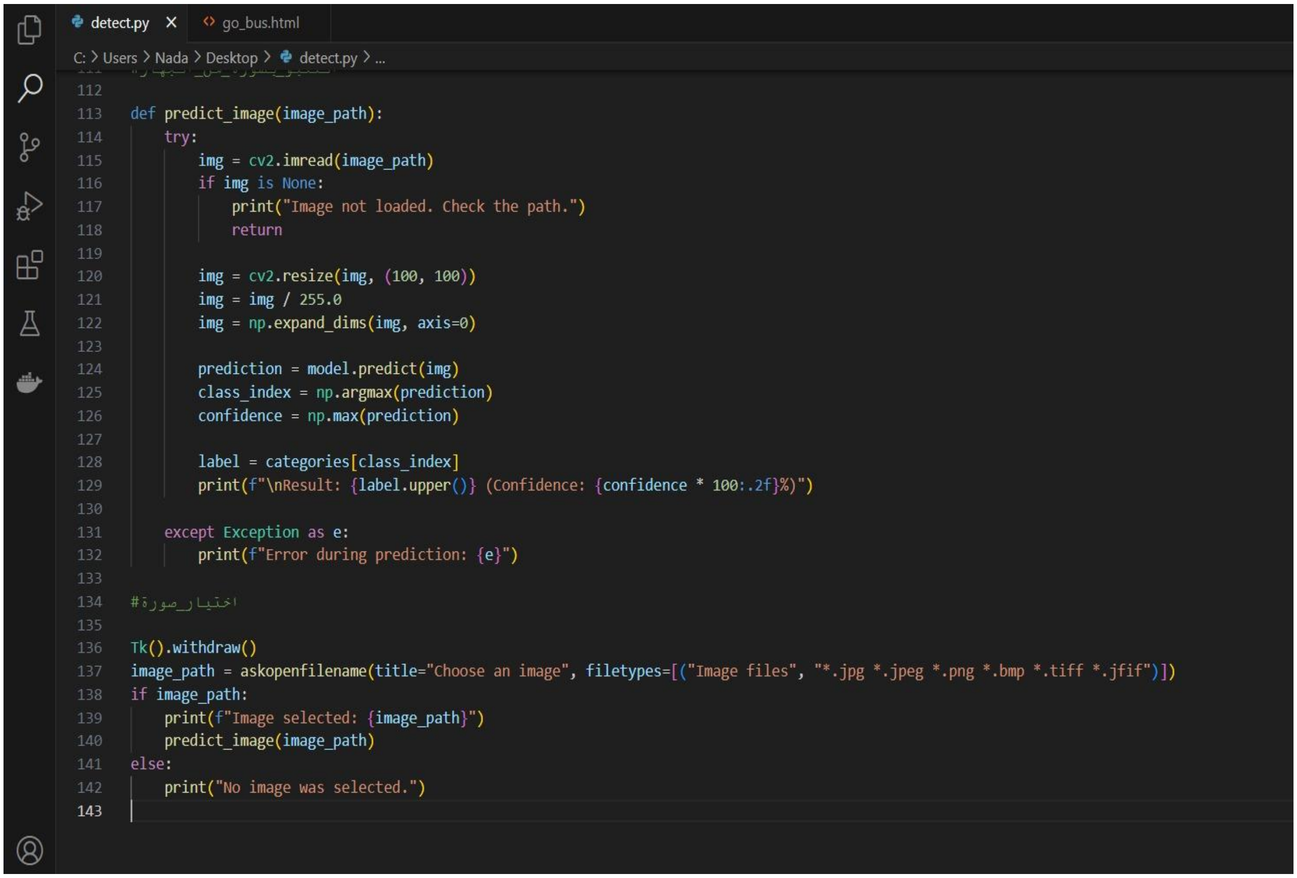
- **Shows CNNs can be reliable for real-time public safety tools.**
- **Suitable for deployment on lightweight devices.**

7. Source Code

```python
import os
import cv2
import numpy as np
from sklearn.model_selection import train_test_split
from keras.utils import to_categorical
import tensorflow as tf
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns
from tkinter import Tk
from tkinter.filedialog import askopenfilename

#تحميل_البيانات

data_dir = "C:/Users/Nada/Desktop/data"
categories = ["with_mask", "without_mask"]
valid_extensions = ['.jpg', '.jpeg', '.png', '.jfif', '.bmp', '.tiff']

data = []
labels = []

for category in categories:
    folder_path = os.path.join(data_dir, category)
    class_num = categories.index(category)
    count = 0

    for root, dirs, files in os.walk(folder_path):
        for img_name in files:
            if any(img_name.lower().endswith(ext) for ext in valid_extensions):
                img_path = os.path.join(root, img_name)
                try:
                    img = cv2.imread(img_path)
                    if img is not None:
                        img = cv2.resize(img, (100, 100))
                        data.append(img)
                        labels.append(class_num)
                        count += 1
                except Exception as e:
                    print(f"Error in image: {img_path} - {e}")

    print(f"Loaded {count} images from '{category}'")

if len(data) == 0 or len(labels) == 0:
    print("No images were loaded. Check the path and folders.")
    exit()
```

```python
        print("No images were loaded. Check the path and folders.")
        exit()

data = np.array(data) / 255.0
labels = to_categorical(labels)

print(f"\nTotal loaded images: {len(data)}")

#تقسيم_البيانات

X_train, X_test, y_train, y_test = train_test_split(data, labels, test_size=0.2, random_state=42)

#بناء_النموذج

model = tf.keras.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(100, 100, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(2, activation='softmax')
])

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

#تدريب_النموذج

history = model.fit(X_train, y_train, epochs=10, validation_data=(X_test, y_test), batch_size=32)

#نتائج_التدريب

history_df = pd.DataFrame(history.history)
print("\nAccuracy table for each training epoch:")
print(history_df[['accuracy', 'val_accuracy']])

print(f"\nFinal training accuracy: {history.history['accuracy'][-1] * 100:.2f}%")
print(f"Final validation accuracy: {history.history['val_accuracy'][-1] * 100:.2f}%")

plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Accuracy per Epoch')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

```python
plt.legend()
plt.show()

#التقييم

y_true = np.argmax(y_test, axis=1)
y_pred_prob = model.predict(X_test)
y_pred = np.argmax(y_pred_prob, axis=1)

print("\nConfusion Matrix:")
cm = confusion_matrix(y_true, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=['Without Mask', 'With Mask'],
            yticklabels=['Without Mask', 'With Mask'])
plt.title('Confusion Matrix')
plt.ylabel('True Label')
plt.xlabel('Predicted Label')
plt.show()

print("\nClassification Report:")
print(classification_report(y_true, y_pred, target_names=['Without Mask', 'With Mask']))

#التنبؤ_بصورة_من_الجهاز

def predict_image(image_path):
    try:
        img = cv2.imread(image_path)
        if img is None:
            print("Image not loaded. Check the path.")
            return

        img = cv2.resize(img, (100, 100))
        img = img / 255.0
        img = np.expand_dims(img, axis=0)

        prediction = model.predict(img)
        class_index = np.argmax(prediction)
        confidence = np.max(prediction)
```




```python
def predict_image(image_path):
    try:
        img = cv2.imread(image_path)
        if img is None:
            print("Image not loaded. Check the path.")
            return

        img = cv2.resize(img, (100, 100))
        img = img / 255.0
        img = np.expand_dims(img, axis=0)

        prediction = model.predict(img)
        class_index = np.argmax(prediction)
        confidence = np.max(prediction)

        label = categories[class_index]
        print(f"\nResult: {label.upper()} (Confidence: {confidence * 100:.2f}%)")

    except Exception as e:
        print(f"Error during prediction: {e}")

#اختيار_صورة

Tk().withdraw()
image_path = askopenfilename(title="Choose an image", filetypes=[("Image files", "*.jpg *.jpeg *.png *.bmp *.tiff *.jfif")])
if image_path:
    print(f"Image selected: {image_path}")
    predict_image(image_path)
else:
    print("No image was selected.")
```

7. Conclusion (Key Contributions)

- **Built an accurate, lightweight CNN model with GUI.**
- **Achieved strong results with low computational cost.**

8. Future Work

- **Add data augmentation.**
- **Use transfer learning for better generalization.**
- **Extend to related tasks like emotion or identity detection.**

9. References

1. **Chollet, F. Deep Learning with Python, Manning Publications, 2018 https://www.manning.com/books/deep-learning-with-python**
2. **Howard et al. "MobileNets: Efficient CNNs for Mobile Vision", arXiv:1704.04861 https://arxiv.org/abs/1704.04861**
3. **He et al. "Deep Residual Learning for Image Recognition", arXiv:1512.03385 https://arxiv.org/abs/1512.03385**
4. **WHO – "Mask Use in the Context of COVID-19", Dec 2020 WHO Mask Guidelines**