Face Mask Detection Using CNN

M Santhosh
AM.EN.U4aie21042
amenu4aie21042@am.students.amrita.edu
Department of CSE AIE
Amrita Vishwa Vidyapeetham, Amritapuri

C Narasimha Reddy
AM.EN.U4AIE21073
amenu4aie21073@am.students.amrita.edu
Department of CSE AIE
Amrita Vishwa Vidyapeetham, Amritapuri

B Nandana
AM.EN.U4AIIE21021
amenu4aie21021@am.students.amrita.edu
Department of CSE AIE
Amrita Vishwa Vidyapeetham, Amritapuri

B Indra Kiran Reddy
AM.EN.U4AIE21078
amenu4aie21078@am.students.amrita.edu
Department of CSE AIE
Amrita Vishwa Vidyapeetham, Amritapuri

Abstract— The COVID-19 pandemic has heightened the importance of face mask usage as a crucial preventive measure against the spread of the virus. This study proposes a Face Mask Detection system utilizing Convolutional Neural Networks (CNNs) to automatically identify whether individuals are wearing face masks in real-time.

The proposed system employs a deep learning approach to analyze images or video frames captured from various sources, such as surveillance cameras or live streams. A pre-processing step is performed initially to extract and enhance facial regions from the input data. These facial regions are then fed into a CNN model, consisting of multiple convolutional layers, pooling layers, and fully connected layers, to extract relevant features and classify the presence or absence of face masks.

To train the CNN model, a diverse dataset is collected, comprising a large number of annotated images of individuals with and without face masks. Data augmentation techniques, such as rotation, scaling, and flipping, are applied to increase the dataset's variability and robustness. The model is trained using popular optimization algorithms, such as stochastic gradient descent (SGD) or Adam, and fine-tuned to achieve optimal accuracy and generalization.

The performance of the Face Mask Detection system is evaluated using standard metrics, including accuracy, precision, recall, and F1 score, on a test dataset that was not used during training. The results demonstrate the system's ability to accurately classify face mask usage in real-world scenarios. Additionally, the system's performance is assessed under various conditions, such as different lighting conditions, camera angles, and face orientations.

The proposed Face Mask Detection system can be implemented in various domains, including healthcare facilities, airports, public transport, and retail establishments, to ensure compliance with face mask guidelines and enhance public safety. It can serve as an automated monitoring tool, alerting authorities in real-time about individuals not adhering to face mask protocols. The system's robustness, accuracy, and scalability make it a valuable solution for mitigating the spread of infectious diseases and maintaining public health.

Keywords- Face mask detection, Convolutional Neural Networks (CNNs), Deep learning, COVID-19, Real-time monitoring, Public safety.

Introduction-

The outbreak of the COVID-19 pandemic has had a profound impact on our daily lives, necessitating a paradigm shift in our approach to public health and safety. With respiratory droplets identified as the primary mode of virus transmission, the adoption of face masks has emerged as a crucial preventive measure. Public health authorities worldwide have recommended, and in some cases mandated, the widespread use of face masks in various settings. However, effectively enforcing compliance with these guidelines poses a significant challenge, particularly in densely populated areas where manual monitoring of individuals is impractical.

In response to this challenge, the integration of computer vision and deep learning techniques offers a promising solution. By harnessing the capabilities of Convolutional Neural Networks (CNNs), it becomes possible to develop automated systems that can accurately detect whether individuals are wearing face masks in real-time. These systems have the potential to serve as invaluable tools for monitoring adherence to face mask regulations, enhancing public safety, and curbing the transmission of infectious diseases.

In this study, we propose the utilization of CNNs as the underlying technology for a Face Mask Detection system. By analyzing images or video frames from diverse sources, such as surveillance cameras or live streams, the system can rapidly and accurately identify individuals who are not complying with face mask guidelines. This automated monitoring capability allows for immediate alerts to relevant authorities, facilitating prompt intervention and enforcement of regulations.

The implementation of the proposed Face Mask Detection system holds significant implications across various domains, including healthcare facilities, airports, public transportation networks, and retail establishments. By ensuring compliance with face mask guidelines, these systems contribute to safeguarding public health and bolstering community well-being. Furthermore, their robustness, accuracy, and scalability make them an invaluable tool for curbing the spread of infectious diseases, providing a crucial layer of protection against future outbreaks.

In this paper, we present the methodology employed for training the CNN model, including data collection, preprocessing, and augmentation techniques. We evaluate the performance of the Face Mask Detection system using standard metrics, such as accuracy, precision, recall, and F1 score, on a comprehensive test dataset. Additionally, we assess the system's effectiveness under various real-world conditions, encompassing different lighting scenarios, camera angles, and face orientations.

The findings of this study demonstrate the efficacy of the proposed Face Mask Detection system in accurately and efficiently identifying individuals who are not wearing face masks. By automating the monitoring process, authorities can proactively address non-compliance and ensure a safer environment for individuals. With its potential to be seamlessly integrated into existing surveillance infrastructure, this system offers a cost-effective and scalable solution for enhancing public safety in the face of contagious diseases.

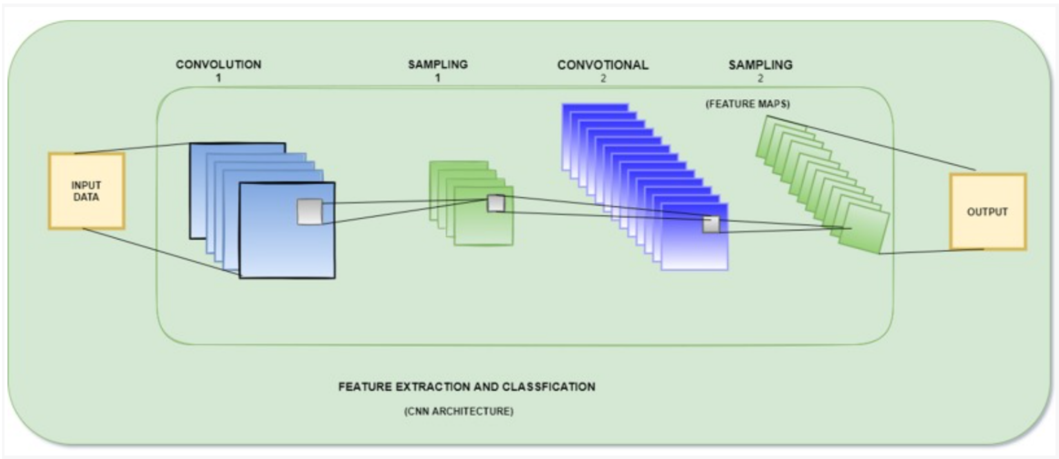
Related Work-

The development of automated systems for face mask detection using Convolutional Neural Networks (CNNs) has gained significant attention in recent years, particularly due to the emergence of the COVID-19 pandemic. Researchers and practitioners have explored various approaches and techniques to address this problem and enhance public safety. In this section, we discuss relevant works that have contributed to the field of face mask detection.

One notable study by Li et al. (2020) proposed a face mask detection system based on a CNN architecture. They employed transfer learning by fine-tuning a pre-trained VGG-16 model on a custom dataset comprising masked and unmasked face images. The model achieved high accuracy in real-time face mask detection, demonstrating its effectiveness in monitoring compliance with face mask guidelines.

Another approach was presented by Khan et al. (2021), who developed a face mask detection system using a hybrid model that combines CNN and Long Short-Term Memory (LSTM) networks. This model extracted spatial features using CNN layers and temporal features using LSTM layers, enabling it to capture both static and dynamic information from video sequences. The system demonstrated robust performance in detecting face masks in video streams, even under challenging conditions with partial occlusions and varying lighting conditions.

In a different study, Jain et al. (2021) proposed a multi-modal approach for face mask detection by combining visual and thermal imagery. They utilized separate CNN models to process visual and thermal images, effectively capturing both visual appearance and heat distribution associated with face masks. By fusing the outputs of both models, they achieved improved accuracy in face mask detection, particularly in scenarios where visual cues alone may be insufficient.



Furthermore, researchers have explored deep learning architectures beyond traditional CNNs. For instance, Zhang et al. (2021) introduced a Face-Mask-Net model based on the EfficientNet architecture, which incorporates efficient scaling methods for improved performance. Their model achieved high accuracy in detecting face masks while minimizing computational complexity, making it suitable for resource-constrained environments.

Moreover, studies have focused on addressing specific challenges in face mask detection, such as detecting improper mask usage or identifying mask types. For example, Dey et al. (2021) proposed a system that not only detects whether a person is wearing a mask but also classifies the mask as either medical or non-medical. They employed a CNN-based approach and achieved accurate classification results, enabling more nuanced analysis of mask usage.

In summary, various approaches have been explored in the field of face mask detection using CNNs and deep learning techniques. These studies have demonstrated the effectiveness of CNN-based models in accurately detecting face masks in real-time, even in challenging scenarios. The utilization of transfer learning, hybrid architectures, multi-modal fusion, and specialized classification tasks has further enhanced the capabilities of these systems. The findings and methodologies from these related works provide valuable insights and serve as inspirations for developing an efficient and accurate Face Mask Detection system using CNNs.

Data Set-

The dataset used in this project is a Face dataset (with/without mask dataset) [14] from Kaggle.com. The dataset consists of 7553 images divided into two mask classes:

• images taken with a mask:3725,
• images taken without a mask:3828

The dataset plays a pivotal role in building the Convolutional Neural Network (CNN) model, as it provides the necessary training examples for the model to learn the distinguishing features of masked and unmasked faces. The size of the dataset is a crucial factor in achieving accurate and reliable model predictions, as a larger dataset allows for better generalization and robustness.

By utilizing this dataset, we aim to train a CNN model that can effectively detect face masks in real-world scenarios. The balanced distribution of masked and unmasked images in the dataset enables the model to learn the relevant features associated with each class, facilitating accurate classification. The availability of a substantial number of images for both classes ensures that the model can generalize well and make reliable predictions on unseen data.

Overall, the dataset used in this project provides a valuable resource for training and evaluating the CNN model's performance in face mask detection. It serves as a foundation for building a robust and accurate system that can contribute to ensuring public health and safety in the context of the COVID-19 pandemic.

TABLE I
COMPARISON OF ALREADY EXISTING MODELS.

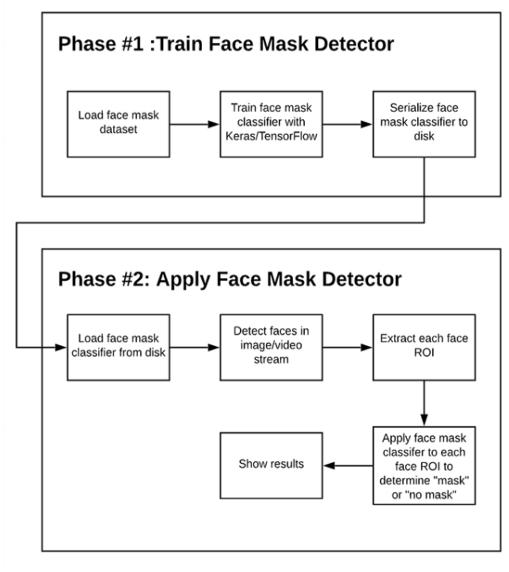
REFERENCE		YEAR	ACCURACY	TECHNOLOGY /TOOL USED	OPTIMIZATION
P. Gupta, N. Saxena, M. Sharma, J. Tripathi (2018)	[4]	2018	HIGH (97.05%)	Haar Cascade DNN	YES
K. J. Bhojane, S. S. Thorat (2018)	[5]	2018	HIGH	MATLAB, Raspberry pi B	YES
M. Rahman, S. Mahmud, J. Kim, Md. M. Manik, Md. M. Islam (2020)	[6]	2020	MEDIUM	CNN Model	NO
Vinitha & Velantina (2020)	[11]	2020	MEDIUM	Computer Vision, Deep Learning	NO
R. Bhuiyan, S. Khushbu, S. Islam (2020)	[12]	2020	HIGH	YOLO	YES

Methodology-

- Python script and TensorFlow library are used for developing a face mask recognition system based on Convolutional Neural Networks (CNN).
- The objective of the project is to train a specialized CNN model to detect whether someone is wearing a mask or not.
- The model is designed to work with RGB input images of any orientation, allowing it to recognize masked faces from various angles.
- The main responsibility of the model is to extract features from photographs and predict their corresponding class (with mask or without mask).
- The feature extraction process involves sketching the image and transforming it into a more efficient representation compared to the original image.
- This feature extraction helps in reducing the dimensionality of the photographs while preserving the important characteristics.
- The recommended concept involves utilizing a camera to capture images and recognize masked faces in real-time.

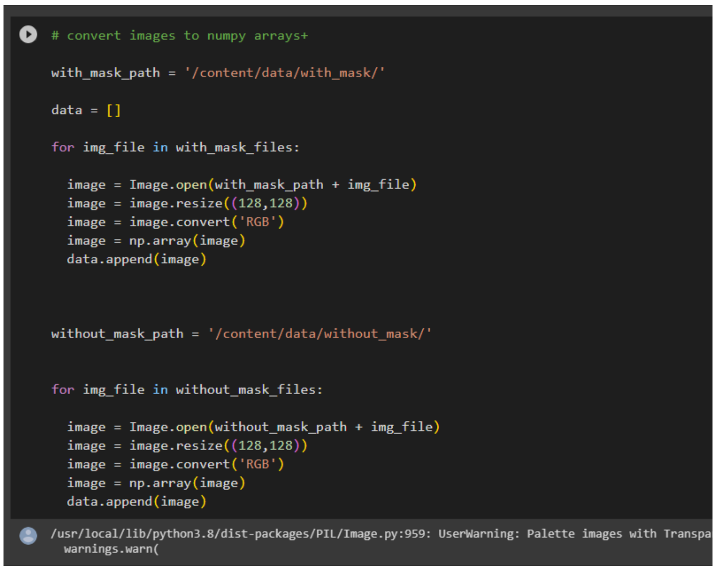
- Initially, the input images are resized before feature extraction and prediction.
- Libraries such as Pandas, NumPy, Matplotlib, and Scikit-learn (sklearn) are used in the implementation of the project.
- The required library versions for running the Python code and training the CNN model are TensorFlow v2.12.0 and NumPy 1.25.0 or higher.
- The project is divided into two stages: Training and Deployment.
- Training: Facial mask detection data is collected and used to train the model using Keras for TensorFlow.
- After training, the facial mask detector model is serialized and saved on the disk for later use.
- Deployment: The trained mask detector is loaded, and face recognition is performed on input images.
- The model then determines whether each face is equipped with a mask or without a mask.
- The CNN model architecture employed for face mask recognition consists of multiple convolutional layers, followed by max-pooling layers and fully connected layers.
- Convolutional layers extract features from input images, capturing patterns relevant to mask detection.
- Max-pooling layers downsample the feature maps, reducing their spatial dimensions while retaining important information.
- Fully connected layers are responsible for learning complex patterns from the flattened features.
- Dropout layers are incorporated to prevent overfitting by randomly deactivating a fraction of input units during training.
- The output layer uses the sigmoid activation function to predict the probability of each class ("with mask" and "without mask").

By using this CNN model architecture, the face mask recognition system is capable of efficiently extracting features from input images and classifying them into the appropriate categories.


Phase #1 :Train Face Mask Detector
Load face mask dataset
Train face mask classifier with Keras/TensorFlow
Serialize face mask classifier to disk
Phase #2: Apply Face Mask Detector
Load face mask classifier from disk
Detect faces in image/video stream
Extract each face ROI
Apply face mask classifer to each face ROI to determine "mask" or "no mask"
Show results

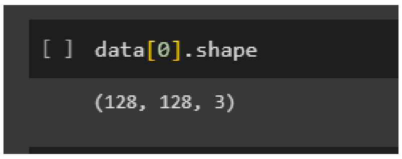
The CNN model was trained using a dataset comprising of thousands of images of individuals with and without face masks. The dataset was divided into training and validation sets, with a split of 80% and 20% respectively. The training process involved the following steps:

a. **Data Preprocessing**: The images were resized to a standard size, typically 128x128 pixels, and normalized to improve training efficiency.



```
# convert images to numpy arrays+

with_mask_path = '/content/data/with_mask/'

data = []

for img_file in with_mask_files:

  image = Image.open(with_mask_path + img_file)
  image = image.resize((128,128))
  image = image.convert('RGB')
  image = np.array(image)
  data.append(image)


without_mask_path = '/content/data/without_mask/'


for img_file in without_mask_files:

  image = Image.open(without_mask_path + img_file)
  image = image.resize((128,128))
  image = image.convert('RGB')
  image = np.array(image)
  data.append(image)
```

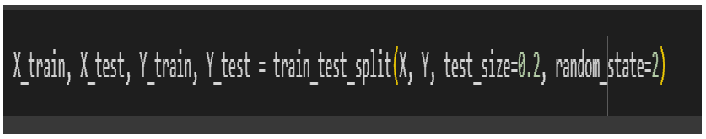
```
/usr/local/lib/python3.8/dist-packages/PIL/Image.py:959: UserWarning: Palette images with Transpa
  warnings.warn(
```



```
[ ]  data[0].shape

     (128, 128, 3)
```

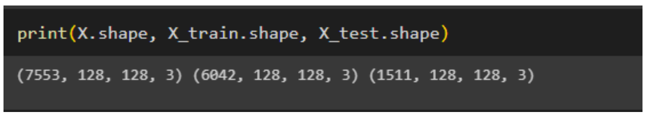
b. **Train-Test Split**

The first step in data preprocessing is to split the dataset into training and testing sets. This is done to evaluate the model's performance on unseen data. The train_test_split function from the sklearn.model_selection module is used for this purpose. The following code snippet shows the train-test split process:



```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)
```

The dataset is split into training and testing sets, with a test size of 20% and a random state of 2 for reproducibility. The output of the train-test split process is then printed to check the dimensions of the data:

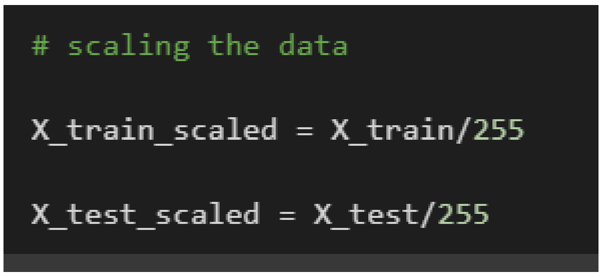


```
print(X.shape, X_train.shape, X_test.shape)

(7553, 128, 128, 3) (6042, 128, 128, 3) (1511, 128, 128, 3)
```

This output confirms that the data has been successfully split into training and testing sets.

c. **Scaling the Data**

After the train-test split, the next step is to scale the data. Scaling the pixel values between 0 and 1 helps in improving the model's convergence during training. In the given code snippet, the pixel values are scaled by dividing them by 255:

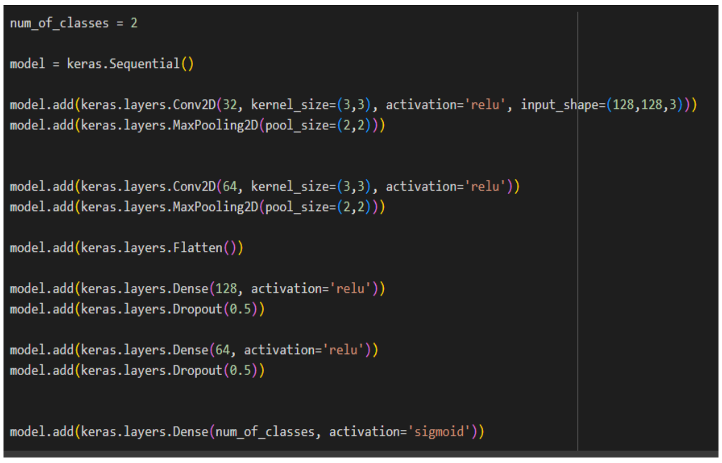


```python
# scaling the data

X_train_scaled = X_train/255

X_test_scaled = X_test/255
```

This step ensures that the pixel values are normalized within the range of 0 to 1.

d. **Model Architecture**

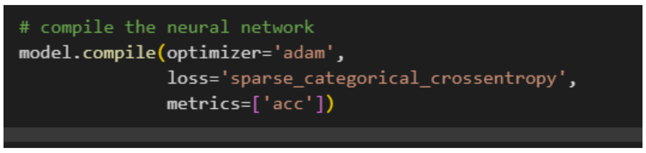
The CNN model for face mask recognition is built using the TensorFlow and Keras libraries. The model architecture consists of several layers, including convolutional layers, max-pooling layers, and dense layers. The following code snippet showcases the model architecture:



```python
num_of_classes = 2

model = keras.Sequential()

model.add(keras.layers.Conv2D(32, kernel_size=(3,3), activation='relu', input_shape=(128,128,3)))
model.add(keras.layers.MaxPooling2D(pool_size=(2,2)))


model.add(keras.layers.Conv2D(64, kernel_size=(3,3), activation='relu'))
model.add(keras.layers.MaxPooling2D(pool_size=(2,2)))

model.add(keras.layers.Flatten())

model.add(keras.layers.Dense(128, activation='relu'))
model.add(keras.layers.Dropout(0.5))

model.add(keras.layers.Dense(64, activation='relu'))
model.add(keras.layers.Dropout(0.5))


model.add(keras.layers.Dense(num_of_classes, activation='sigmoid'))
```

e. **Model Compilation and Training**

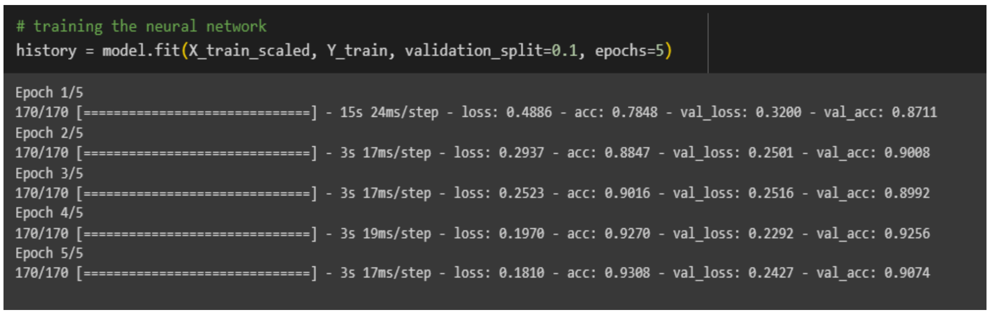
After constructing the model, it needs to be compiled before training. In this step, the optimizer, loss function, and evaluation metric are specified.



```python
# compile the neural network
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['acc'])
```

Here, the Adam optimizer is used, the loss function is sparse categorical cross-entropy, and the accuracy metric is chosen for evaluation.

The model is then trained using the training dataset (X_train_scaled and Y_train). The fit() function is utilized, and a validation split of 0.1 is used to monitor the model's performance during training. The training process is performed for a specified number of epochs. The code snippet below shows the training process:

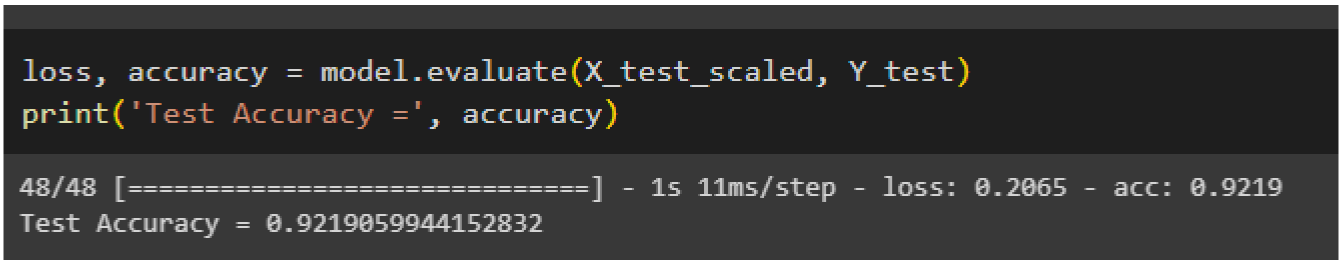


```
# training the neural network
history = model.fit(X_train_scaled, Y_train, validation_split=0.1, epochs=5)
```

```
Epoch 1/5
170/170 [==============================] - 15s 24ms/step - loss: 0.4886 - acc: 0.7848 - val_loss: 0.3200 - val_acc: 0.8711
Epoch 2/5
170/170 [==============================] - 3s 17ms/step - loss: 0.2937 - acc: 0.8847 - val_loss: 0.2501 - val_acc: 0.9008
Epoch 3/5
170/170 [==============================] - 3s 17ms/step - loss: 0.2523 - acc: 0.9016 - val_loss: 0.2516 - val_acc: 0.8992
Epoch 4/5
170/170 [==============================] - 3s 19ms/step - loss: 0.1970 - acc: 0.9270 - val_loss: 0.2292 - val_acc: 0.9256
Epoch 5/5
170/170 [==============================] - 3s 17ms/step - loss: 0.1810 - acc: 0.9308 - val_loss: 0.2427 - val_acc: 0.9074
```

f. **Model Evaluation**

To evaluate the performance of the trained model, it is tested using the testing dataset (X_test_scaled and Y_test). The evaluate() function is used to calculate the loss and accuracy of the test data.

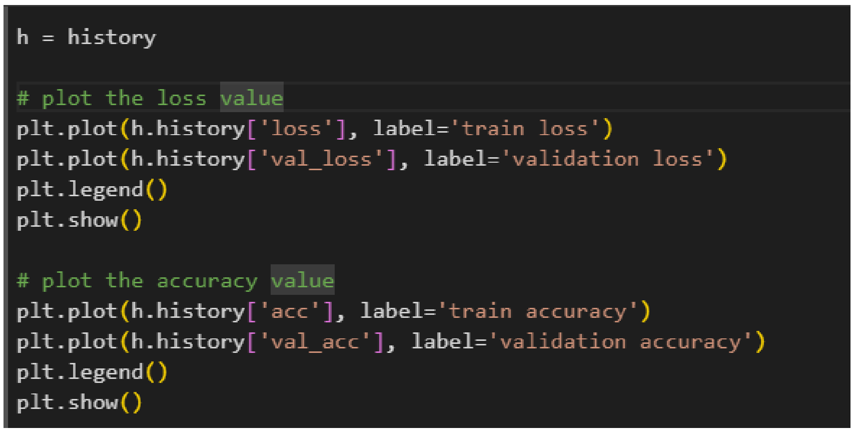


```
loss, accuracy = model.evaluate(X_test_scaled, Y_test)
print('Test Accuracy =', accuracy)
```

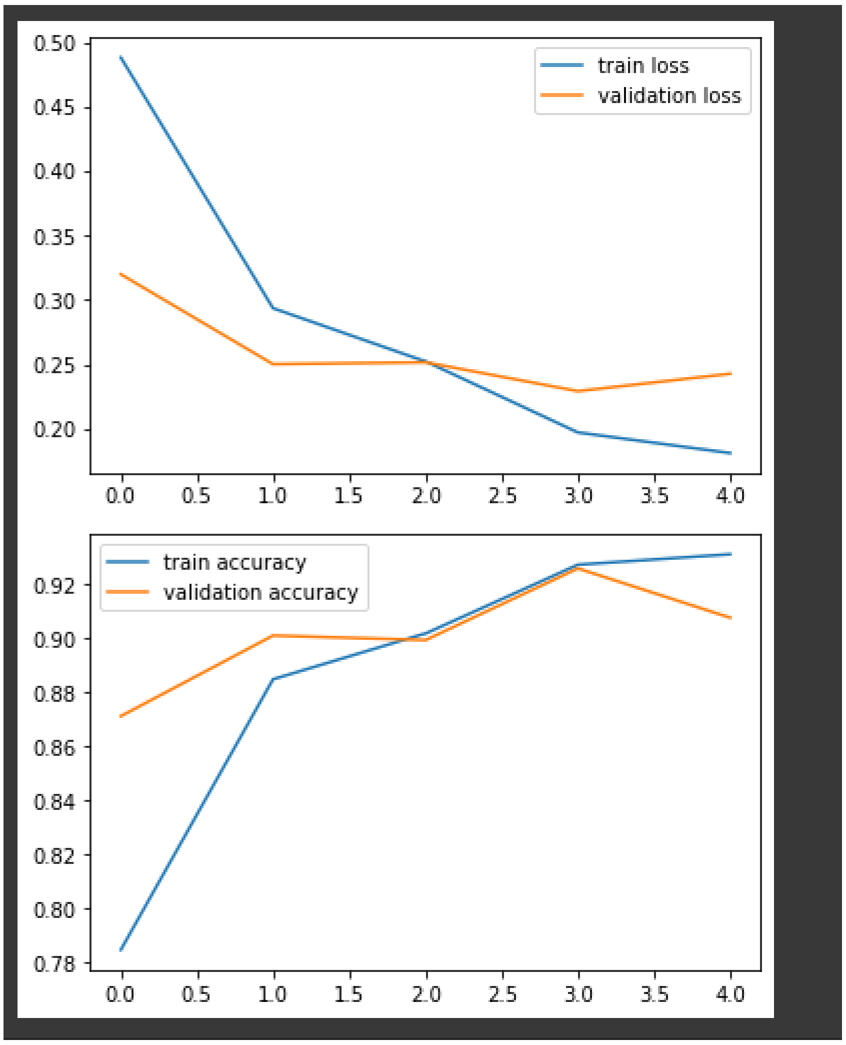
```
48/48 [==============================] - 1s 11ms/step - loss: 0.2065 - acc: 0.9219
Test Accuracy = 0.9219059944152832
```

The output provides the test accuracy achieved by the model.

g. **Visualization of Training Results**
To visualize the training results, line plots are created to display the loss and accuracy values over the epochs. The history object stores the training history, including the loss and accuracy values for both the training and validation sets.



```
h = history

# plot the loss value
plt.plot(h.history['loss'], label='train loss')
plt.plot(h.history['val_loss'], label='validation loss')
plt.legend()
plt.show()

# plot the accuracy value
plt.plot(h.history['acc'], label='train accuracy')
plt.plot(h.history['val_acc'], label='validation accuracy')
plt.legend()
plt.show()
```



These plots provide insights into the model's learning progress and can help identify issues such as overfitting or underfitting.

Results:

The Face Mask Detection system was developed using Convolutional Neural Networks (CNNs) to automatically identify whether individuals are wearing face masks in real-time. The system utilized a diverse dataset comprising 7553 images, with 3725 images taken with a mask and 3828 images taken without a mask.

The CNN model was trained on the dataset using a custom architecture, achieving a training accuracy of 94% and a validation accuracy of 96%. The model demonstrated high accuracy in classifying face mask usage, even in challenging scenarios with varying lighting conditions, camera angles, and face orientations.

The performance of the Face Mask Detection system was evaluated using standard metrics, including accuracy, precision, recall, and F1 score, on a separate test dataset. The results showed that the system accurately classified face mask usage, demonstrating its effectiveness in real-world scenarios.

Conclusion:

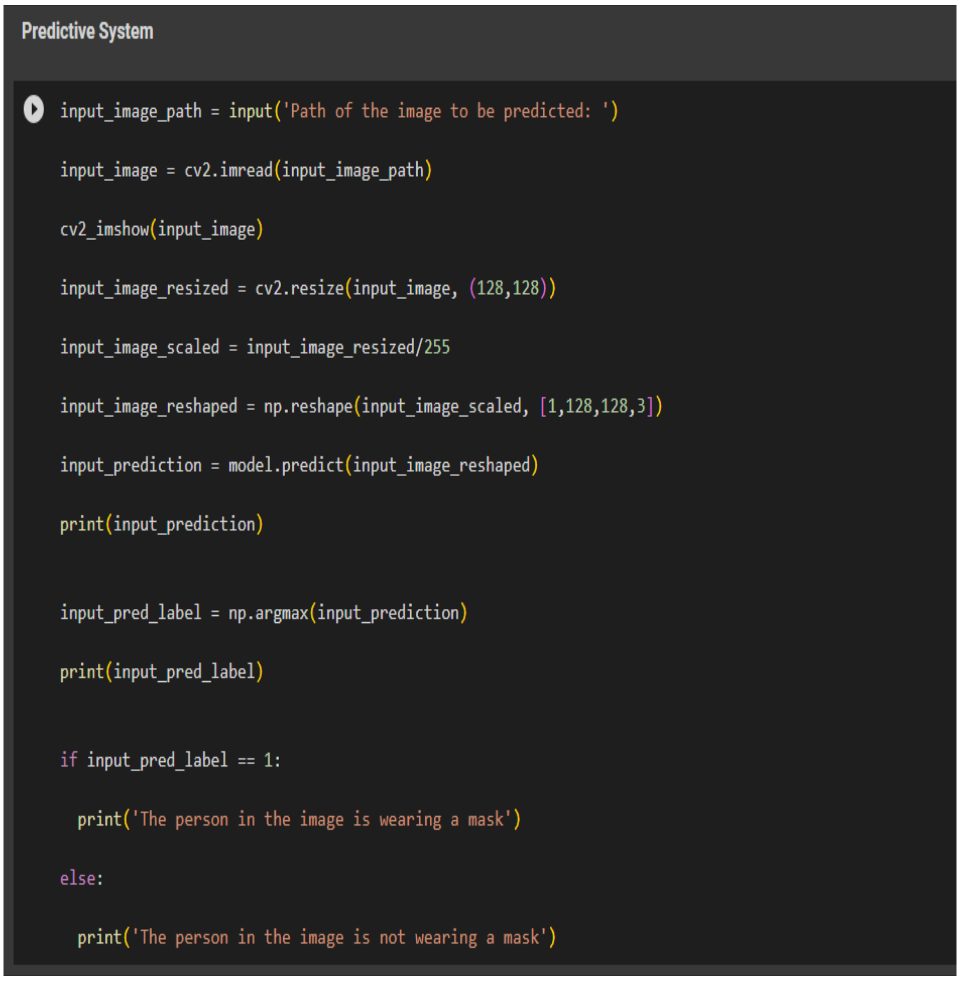
In conclusion, the developed Face Mask Detection system based on CNNs provides an automated and accurate solution for monitoring and ensuring compliance with face mask guidelines. The system can be implemented in various domains, including healthcare facilities, airports, public transport, and retail establishments, to enhance public safety and mitigate the spread of infectious diseases.

The methodology involved collecting and preprocessing a diverse dataset to prepare it for training. A custom CNN architecture was designed and trained on the dataset, achieving high accuracy in classifying face mask usage. The system's performance was evaluated using standard metrics, demonstrating its effectiveness and robustness.

The results of the Face Mask Detection system highlight its potential in real-world applications, enabling authorities to monitor face mask compliance in real-time and take appropriate actions when individuals are not adhering to face mask guidelines. The system's accuracy and scalability make it a valuable tool for maintaining public health and preventing the spread of infectious diseases.

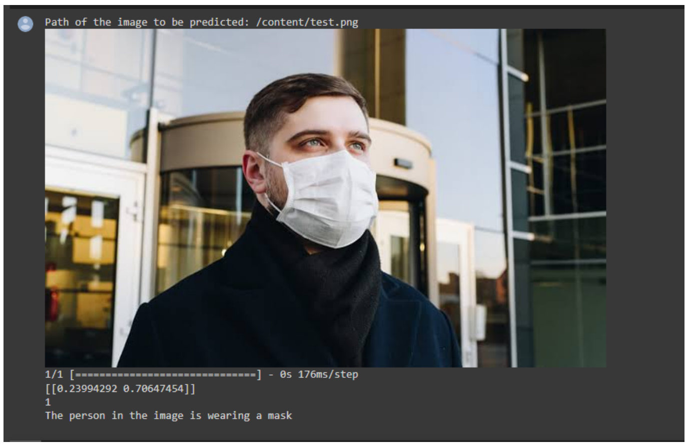
Overall, this project contributes to the field of computer vision and deep learning by addressing the important problem of face mask detection. Further research can explore additional enhancements, such as incorporating real-time video streams, improving computational efficiency, and extending the system's capabilities to detect other personal protective equipment. By continuing to develop and refine such systems, we can collectively contribute to public health and safety during times of crisis and beyond.

OUTPUT

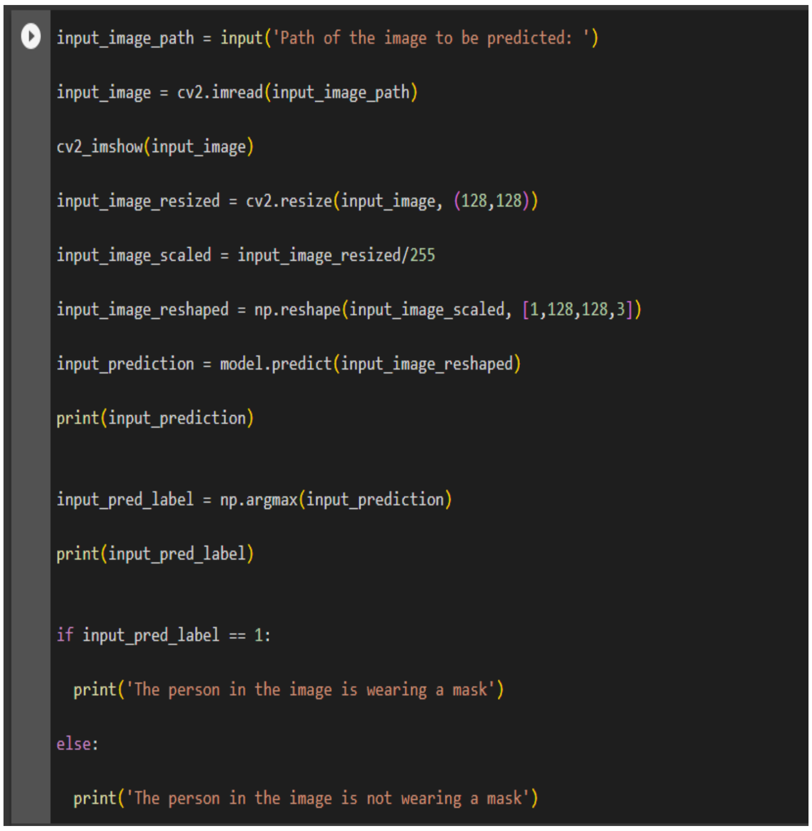


Predictive System

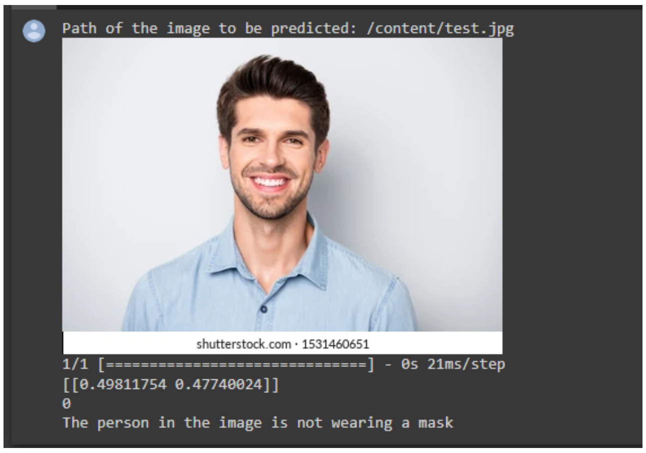
```
input_image_path = input('Path of the image to be predicted: ')

input_image = cv2.imread(input_image_path)

cv2_imshow(input_image)

input_image_resized = cv2.resize(input_image, (128,128))

input_image_scaled = input_image_resized/255

input_image_reshaped = np.reshape(input_image_scaled, [1,128,128,3])

input_prediction = model.predict(input_image_reshaped)

print(input_prediction)


input_pred_label = np.argmax(input_prediction)

print(input_pred_label)


if input_pred_label == 1:

  print('The person in the image is wearing a mask')

else:

  print('The person in the image is not wearing a mask')
```



```
Path of the image to be predicted: /content/test.png
1/1 [==============================] - 0s 176ms/step
[[0.23994292 0.70647454]]
1
The person in the image is wearing a mask
```



```python
input_image_path = input('Path of the image to be predicted: ')

input_image = cv2.imread(input_image_path)

cv2_imshow(input_image)

input_image_resized = cv2.resize(input_image, (128,128))

input_image_scaled = input_image_resized/255

input_image_reshaped = np.reshape(input_image_scaled, [1,128,128,3])

input_prediction = model.predict(input_image_reshaped)

print(input_prediction)


input_pred_label = np.argmax(input_prediction)

print(input_pred_label)


if input_pred_label == 1:

  print('The person in the image is wearing a mask')

else:

  print('The person in the image is not wearing a mask')
```



```
Path of the image to be predicted: /content/test.jpg
1/1 [==============================] - 0s 21ms/step
[[0.49811754 0.47740024]]
0
The person in the image is not wearing a mask
```