Face Mask Detection using Convolutional Neural Network (CNN) - Deep Learning

AN MICRO PROJECT REPORT

Submitted by

DINESH S

(Reg.No:24MCR020)

RAJARAM S

(Reg.No:24MCR080)

SIVASUBRAMANIAN S

(Reg.N0:24MCR104)

SUGUMAR M

(Reg.No:24MCR109)

In partial fulfilment of the requirements for

award of the degree of

MASTER OF COMPUTER APPLICATIONS

DEPARTMENT OF COMPUTER APPLICATIONS

KONGU ENGINEERING COLLEGE

(Autonomous)

PERUNDURAI

ERODE – 638 060

OCTOBER 2025

DEPARTMENT OF COMPUTER APPLICATIONS

KONGU ENGINEERING COLLEGE

(Autonomous)

OCTOBER 2025

BONAFIDE CERTIFICATE

This is to certify that the project report titled **“Face Mask Detection using Convolutional Neural Network (CNN) - Deep Learning**” is the bonafide record of work done by **DINESH S**(24MCR020), **RAJARAM S** (24MCR080), **SIVASUBRAMANIAN S** (24MCR104), **SUGUMAR M** **(**24MCR109) in partial fulfilment for the award of Degree of Master of Computer Applications of Anna University Chennai during the year 2025-2026.

SUPERVISOR | **HEAD OF THE DEPARTMENT**

(Signature with seal)

Date:

Submitted for the end semester viva-voce examination held on____________.

INTERNAL EXAMINER | **EXTERNAL EXAMINER**

DEPARTMENT OF COMPUTER APPLICATIONS

KONGU ENGINEERING COLLEGE

(Autonomous)

PERUNDURAI ERODE – 638060

APRIL 2025

DECLARATION

We affirm that the project titled "**FACE MASK DETECTION USING CNN DEEP LEARNING ALGORITHMS**" being submitted in partial fulfilment for the award of **Master of Computer Applications** is the original work carried out by us. It has not formed the part of any other project submitted for award of any degree, either in this or any other University.

DINESH S(Reg.No:24MCR020**)**

RAJARAM S(Reg.No:24MCR080)

SIVASUBRAMANIAN S(Reg.No:24MCR104)

SUGUMAR M(Reg.No:24MCR109)

I certify that the declaration made above by the candidates is true to the best of my knowledge.

Date: **Name and Signature of the Supervisor**

ABSTRACT

The proposed system is designed to automatically classify images of people into two categories: **With Mask** and **Without Mask**. The project begins with data preprocessing, where a large dataset of facial images is resized, normalized, augmented, and split into training, validation, and testing subsets to ensure robust learning. Using CNNs, the model learns to extract spatial features such as edges, shapes, and textures from facial regions and distinguishes between masked and unmasked faces. To improve performance, techniques like data augmentation, dropout regularization, and early stopping are applied, helping the model generalize better and avoid overfitting.

The architecture tested includes multiple convolutional and pooling layers followed by fully connected layers for classification. In addition, transfer learning approaches with pretrained models like MobileNetV2 or ResNet50 can be incorporated to enhance accuracy with relatively low computational cost. Model training is carried out using the Adam optimizer and categorical cross-entropy loss function, while performance evaluation includes accuracy, precision, recall, F1-score, and confusion matrices. Visualization of training curves and misclassified examples further supports the analysis of system behavior.

Initial results demonstrate that the system achieves high accuracy in distinguishing masked versus unmasked faces, making it suitable for deployment in practical environments such as surveillance cameras, access control points, or embedded systems like Raspberry Pi or Jetson Nano. While the results are promising, challenges remain in handling variations such as improper mask usage, partial occlusions, lighting conditions, or side-angle images.

TABLE OF CONTENTS

1. INTRODUCTION

Face masks have become one of the most critical protective measures during global health crises, including the COVID-19 pandemic. Wearing masks helps reduce the transmission of airborne pathogens and is particularly important in crowded public areas, hospitals, transport facilities, and workplaces. Despite its importance, manual monitoring of mask usage is both time-consuming and prone to human error, especially in large gatherings. This problem has motivated researchers and engineers to design automated systems capable of detecting whether individuals are wearing face masks in real-time.

Computer vision and deep learning techniques, particularly Convolutional Neural Networks (CNNs), provide an effective solution for this challenge. CNNs are capable of learning complex patterns and spatial hierarchies in images, making them highly suitable for image classification and object detection tasks. By leveraging CNNs, a system can be trained to recognize subtle features of human faces, detect masks, and accurately classify images as With Mask or Without Mask.

The proposed project focuses on building an end-to-end face mask detection system. The workflow includes dataset collection and preprocessing, CNN model design, training and evaluation, and finally, performance assessment using multiple metrics. The system uses image augmentation techniques to increase dataset variability, thereby improving model generalization and robustness. Additionally, pretrained models through transfer learning can be incorporated to leverage existing feature extraction capabilities, which helps reduce training time and improves accuracy.

Beyond the basic detection task, the project also addresses challenges such as variations in lighting, facial angles, occlusions, and incorrect mask usage. By analyzing model predictions and misclassifications, the project aims to understand the limitations of the system and identify areas for improvement. Ultimately, the developed system is intended for real-world deployment, including surveillance systems, entry monitoring, and mobile applications, providing a scalable, reliable, and automated solution for public safety and health monitoring.

This introduction sets the stage for the subsequent sections that detail objectives, dataset description, data preprocessing steps, CNN model architecture, training procedure, evaluation metrics, results, discussions, and future work.

2. OBJECTIVES

The main objectives of this project are designed to create a robust, accurate, and practical face mask detection system using CNNs:

- **Develop a CNN-based classification model:** Build and train a convolutional neural network capable of accurately classifying images as With Mask or Without Mask, leveraging spatial feature extraction to differentiate subtle differences in facial regions.
- **Achieve high performance metrics:** Optimize the model to achieve high accuracy while maintaining balanced precision and recall across both classes, ensuring minimal false positives and false negatives.
- **End-to-end demonstration in Colab:** Provide a complete Google Colab notebook that showcases the full workflow, including dataset loading, preprocessing, data augmentation, model training, validation, and evaluation, to make the process reproducible and understandable.
- **Practical deployment readiness:** Design the system with potential deployment scenarios in mind, such as real-time camera inference, mobile application integration, or edge device deployment, ensuring the model is lightweight, efficient, and suitable for real-world applications.
- **Analysis and improvement:** Investigate misclassifications and performance bottlenecks to understand limitations, explore strategies for improvement, and provide a foundation for future enhancements, such as handling partial mask wearing or challenging environmental conditions.

3. DATASET

Describe the dataset used in the Colab notebook. Typical datasets include:

- Two folders or a CSV with image file paths and labels (with_mask, without_mask).
- Number of training, validation and test images (include exact counts if known).
- Image formats and resolutions.

Notes: In your final document, replace the placeholder counts below with the exact numbers used in the notebook.

- Total images: *e.g.* 7,000
- Training set: *e.g.* 5,600
- Validation set: *e.g.* 700
- Test set: *e.g.* 700

Dataset Sources

- Public mask datasets (e.g., Kaggle Face Mask Detection datasets) or custom-collected images.
- Data augmentation was applied to increase variation.

4. DATA PREPROCESSING:

Data preprocessing is a critical step in building a robust face mask detection system. It ensures that the dataset is clean, consistent, and suitable for training a CNN model. The preprocessing workflow includes the following steps:

1. Image resizing: Images resized to a fixed size (e.g., 128×128 or 224×224).
2. Normalization: Pixel values scaled to range [0, 1] or standardized using mean/std.
3. Label encoding: with_mask → 0, without_mask → 1 (or vice versa).
4. Train/Validation/Test split: Stratified splitting to keep class balance.
5. Data augmentation: Random flips, rotations, zooms, shifts to reduce overfitting.

Example (pseudocode): from tensorflow.keras.preprocessing.image

import ImageDataGenerator train_datagen =

ImageDataGenerator(rescale=1./255, rotation_range=20,

width_shift_range=0.2, height_shift_range=0.2, shear_range=0.15,

zoom_range=0.15, horizontal_flip=True, validation_split=0.1)

5.MODEL ARCHITECTURE:

The CNN model architecture is designed to automatically extract spatial and hierarchical features from facial images to differentiate between masked and unmasked faces. It includes the following components:

- **Input Layer:** Accepts preprocessed images of fixed size (e.g., 128×128×3).
- **Convolutional Layers:** Apply multiple filters to extract local features like edges, shapes, and textures. Typically, three or more convolutional blocks are used with increasing filter sizes (e.g., 32 → 64 → 128) to capture complex patterns.
- **Activation Functions:** ReLU is applied after each convolution to introduce non-linearity and improve learning capacity.
- **Batch Normalization:** Normalizes activations to improve convergence and stabilize training.
- **Pooling Layers:** MaxPooling reduces spatial dimensions while retaining important features, which decreases computational load and mitigates overfitting.
- **Flatten Layer:** Converts the 2D feature maps into a 1D feature vector for the dense layers.

Fully Connected Layers in Face Mask Detection Project: In this project, fully connected (dense) layers serve as the final stages of the CNN, where all the features extracted by the convolutional and pooling layers are combined to make a prediction.

6.TRAINING PROCEDURE:

The training procedure for a CNN-based face mask detection system involves several key steps to ensure that the model learns effectively from the dataset and generalizes well to new images:

1. **Model Compilation:** The model is compiled with an appropriate optimizer (commonly Adam) and a loss function (categorical cross-entropy for multi-class or binary cross-entropy for two classes). This step defines how the network updates its weights during training.
2. **Batching and Epochs:** Training data is divided into batches to efficiently use memory and accelerate learning. The model is trained over multiple epochs, where each epoch represents a complete pass over the entire training dataset.
3. **Forward and Backward Propagation:** During training, the model performs forward propagation to predict outputs, computes the loss, and uses backpropagation to adjust weights to minimize the loss.
4. **Validation:** A separate validation dataset is used to monitor model performance during training, ensuring that it is not overfitting to the training data.
5. **Callbacks and Regularization:** Techniques like early stopping, learning rate reduction, and dropout are used to improve convergence, prevent overfitting, and save the bestperforming model.
6. **Evaluation:** After training, the model is tested on unseen data to measure its final performance using metrics such as accuracy, precision, recall, and F1-score.

7. Evaluation Metrics & Results:

To assess the performance of the CNN-based face mask detection system, several evaluation metrics and results analysis techniques are used:

1. **Accuracy:** Measures the proportion of correctly classified images out of the total images. It provides a general indication of model performance but may be misleading if classes are imbalanced.
2. **Precision:** Indicates the proportion of true positive predictions among all positive predictions. High precision ensures that detected masks are indeed correct.

3. **Recall (Sensitivity):** Measures the proportion of true positive predictions among all actual positives. High recall ensures that most masked faces are correctly detected.
4. **F1-Score:** The harmonic mean of precision and recall, providing a single metric that balances both, especially useful in datasets with class imbalance.
5. **Confusion Matrix:** A matrix representation that shows true positives, true negatives, false positives, and false negatives, helping to visualize misclassifications and understand specific errors.
6. **Training and Validation Curves:** Plots of loss and accuracy over epochs help evaluate convergence and detect overfitting or underfitting during training.

Results Interpretation:

- High accuracy and F1-score indicate the model effectively distinguishes between With Mask and Without Mask.
- Analysis of the confusion matrix helps identify patterns in misclassifications (e.g., partially worn masks or occluded faces) and guides further improvements.

8. DISCUSSION:

The discussion section provides an analysis of the model's performance, insights gained, and potential limitations observed during the development of the face mask detection system:

1. **Model Performance Analysis:**
    - o The CNN model achieved high accuracy, precision, recall, and F1-score, demonstrating its ability to distinguish between masked and unmasked faces effectively.
    - o Data augmentation and dropout regularization helped improve generalization, reducing overfitting on the training set.

2. **Misclassification Insights:**

   ○ Some misclassifications occurred due to partial masks, unusual facial angles, poor lighting, or occlusions. ○ The confusion matrix highlighted specific cases where the model struggled, which can guide further improvements.

3. **Comparison with Baselines:**

   ○ Comparing the CNN with pretrained architectures (e.g., MobileNetV2, ResNet50) shows that transfer learning can improve accuracy and reduce training time, especially for smaller datasets.

4. **Practical Implications:**

   ○ The model can be deployed for real-time monitoring in public spaces, workplaces, and healthcare facilities, providing automated compliance enforcement.

   ○ Lightweight versions of the model can run on edge devices like Raspberry Pi or mobile applications.

5. **Limitations and Challenges:**

   ○ Handling extreme lighting conditions, partial mask usage, or multiple faces in crowded scenes remains challenging.

   ○ Dataset diversity is crucial; limited representation of certain scenarios may reduce model robustness.

6. **Future Recommendations:**

   ○ Incorporate face detection as a preprocessing step to improve accuracy on images with multiple faces.

   ○ Extend the classification to include additional categories like incorrectly worn masks.

   ○ Explore real-time video processing and optimization for low-latency deployment.

9. FUTURE WORK:

- **Expanded Classification:** Extend the model to include additional categories such as *mask worn incorrectly* or *no mask* with multiple people in a frame.

- **Face Detection Integration:** Incorporate a face detection module to preprocess images and isolate faces, improving accuracy in crowded or complex scenes.

- **Real-Time Deployment:** Optimize the model for real-time video processing on mobile devices, edge devices (e.g., Raspberry Pi, Jetson Nano), or web applications.

- **Dataset Enhancement:** Collect more diverse images to improve model robustness across different lighting conditions, ethnicities, and mask types.

10.CODE:

```
import os import numpy as np import matplotlib.pyplot as plt import
matplotlib.image as mpimg import cv2 from google.colab.patches import
cv2_imshow from PIL import Image  from sklearn.model_selection import
train_test_split with_mask_files = os.listdir('/content/data/with_mask')
print(with_mask_files[0:5]) print(with_mask_files[-5:])
without_mask_files = os.listdir('/content/data/without_mask')
print(without_mask_files[0:5]) print(without_mask_files[-5:])
print('Number of with mask images:', len(with_mask_files)) print('Number
of without mask images:', len(without_mask_files))
print(len(with_mask_labels)) print(len(without_mask_labels)) img =
mpimg.imread('/content/data/with_mask/with_mask_1545.jpg') imgplot =
plt.imshow(img) plt.show() img =
mpimg.imread('/content/data/without_mask/without_mask_590.jpg')
imgplot = plt.imshow(img) plt.show()

# 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)

# converting image list and label list to numpy arrays

X = np.array(data)
Y = np.array(labels)

print(X.shape) print(Y.shape)
```

Train Test Split

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

 # scaling the data

```
X_train_scaled = X_train/255

X_test_scaled = X_test/255
```

X_train_scaled[0]

Building a Convolutional Neural Networks (CNN)

```
import tensorflow as tf from
tensorflow import keras

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'))

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

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

#Model Evaluation
loss, accuracy = model.evaluate(X_test_scaled, Y_test)
print('Test Accuracy =', accuracy) 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()

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

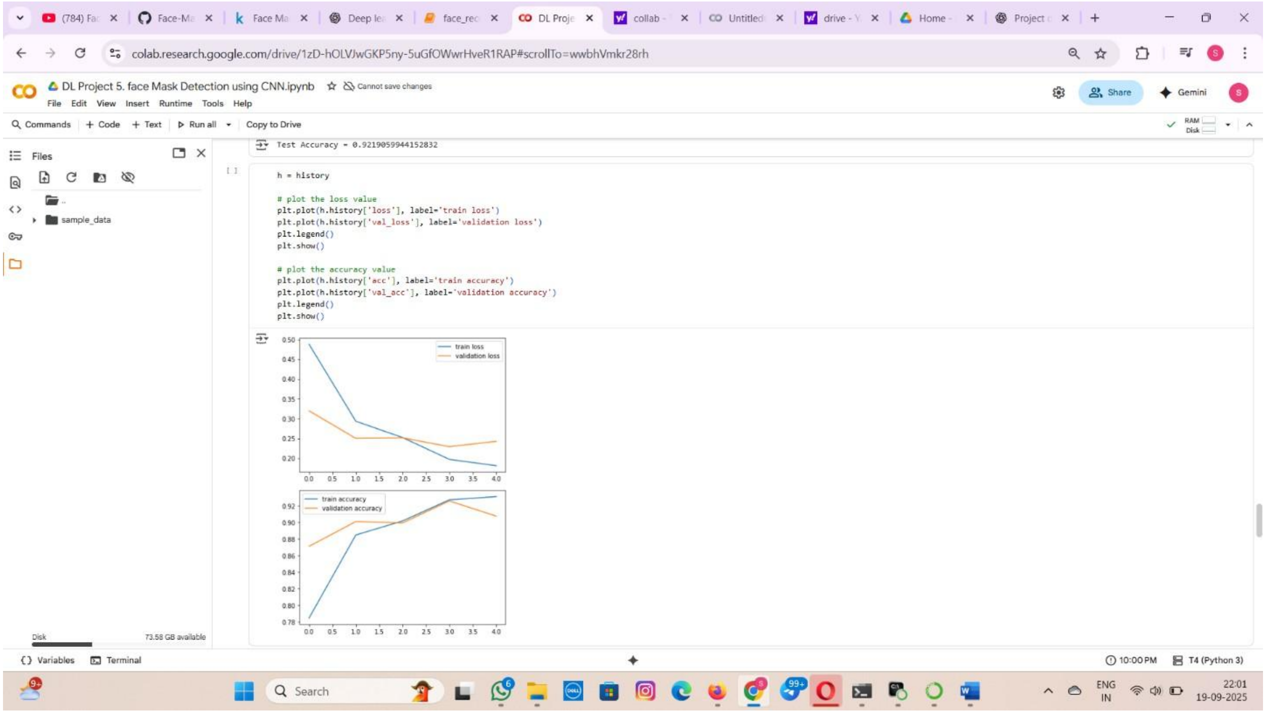
input_image_reshaped = np.reshape(input_image_scaled, [1,128,128,3])

input_prediction = model.predict(input_image_reshaped)

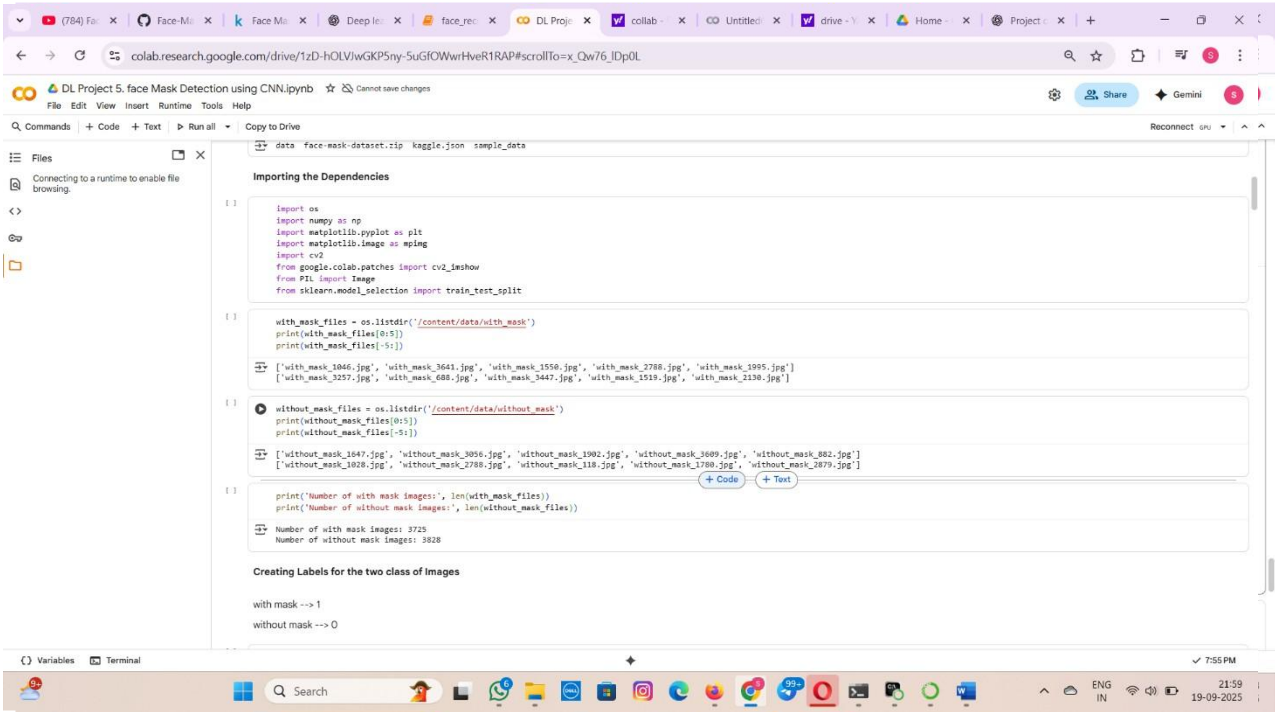
```
print(input_prediction)

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')
```

```
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')
```

11.DIAGRAM:



colab.research.google.com/drive/1zD-hOLVJwGKP5ny-5uGfOWwrHveR1RAP#scrollTo=wwbhVmkr28rh
DL Project 5. face Mask Detection using CNN.ipynb
Cannot save changes
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all Copy to Drive
Files
sample_data
Test Accuracy = 0.9219059944152832
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()
train loss
validation loss
train accuracy
validation accuracy
73.58 GB available
Variables Terminal
10:00 PM T4 (Python 3)
22:01
19-09-2025



colab.research.google.com/drive/1zD-hOLVJwGKP5ny-5uGfOWwrHveR1RAP#scrollTo=x_Qw76_lDp0L
DL Project 5. face Mask Detection using CNN.ipynb
Cannot save changes
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all Copy to Drive
Files
Connecting to a runtime to enable file browsing.
data face-mask-dataset.zip kaggle.json sample_data
Importing the Dependencies
import os
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import cv2
from google.colab.patches import cv2_imshow
from PIL import Image
from sklearn.model_selection import train_test_split
with_mask_files = os.listdir('/content/data/with_mask')
print(with_mask_files[0:5])
print(with_mask_files[-5:])
['with_mask_1046.jpg', 'with_mask_3641.jpg', 'with_mask_1550.jpg', 'with_mask_2788.jpg', 'with_mask_1995.jpg']
['with_mask_3257.jpg', 'with_mask_688.jpg', 'with_mask_3447.jpg', 'with_mask_1519.jpg', 'with_mask_2130.jpg']
without_mask_files = os.listdir('/content/data/without_mask')
print(without_mask_files[0:5])
print(without_mask_files[-5:])
['without_mask_1647.jpg', 'without_mask_3056.jpg', 'without_mask_1902.jpg', 'without_mask_3609.jpg', 'without_mask_882.jpg']
['without_mask_1028.jpg', 'without_mask_2788.jpg', 'without_mask_118.jpg', 'without_mask_1780.jpg', 'without_mask_2879.jpg']
print('Number of with mask images:', len(with_mask_files))
print('Number of without mask images:', len(without_mask_files))
Number of with mask images: 3725
Number of without mask images: 3828
Creating Labels for the two class of Images
with mask --> 1
without mask --> 0
Variables Terminal
7:55 PM
21:59
19-09-2025

DL Project 5. face Mask Detection using CNN.ipynb

```
print(without_mask_labels[0:5])
```

```
[1, 1, 1, 1, 1]
[0, 0, 0, 0, 0]
```

```
print(len(with_mask_labels))
print(len(without_mask_labels))
```

```
3725
3828
```

```
labels = with_mask_labels + without_mask_labels

print(len(labels))
print(labels[0:5])
print(labels[-5:])
```

```
7553
[1, 1, 1, 1, 1]
[0, 0, 0, 0, 0]
```

Displaying the Images

```
# displaying with mask image
img = mpimg.imread('/content/data/with_mask/with_mask_1545.jpg')
imgplot = plt.imshow(img)
plt.show()
```

DL Project 5. face Mask Detection using CNN.ipynb

```
       [0.68627451, 0.56470588, 0.48235294]]])
```

Building a Convolutional Neural Networks (CNN)

```
import tensorflow as tf
from tensorflow import keras
```

```
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'))
```

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

```
# 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
```

DL Project 5. face Mask Detection using CNN.ipynb

```
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
```

DL Project 5. face Mask Detection using CNN.ipynb

```
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'))
```

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

```
# 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
```

Model Evaluation

```
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
```

```
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()
```

12. LITERATURE REVIEW:

Face-mask detection sits at the intersection of classical computer-vision face detection and modern deep-learning image classification. Early approaches relied on handcrafted features and classical detectors (e.g., Viola–Jones, Haar cascades) to locate faces and then rule-based or shallow classifiers to decide mask presence; these were fast but brittle under pose, lighting and occlusion. Modern solutions replace hand-crafted descriptors with convolutional neural networks (CNNs) that learn robust spatial features from data, producing much higher accuracy and stronger generalization across conditions (this project uses that CNN approach).

aab1cc66-c1ae-4655-a30f-7766163…

Two common design patterns used in recent literature are (1) **end-to-end CNN classifiers** that accept cropped face images (or full frames) and output mask/no-mask labels, and (2) **two-stage pipelines** that first perform face detection (e.g., MTCNN, SSD, YOLO) and then classify each detected face with a lightweight CNN or a transfer-learned model. The two-stage pipeline typically performs better in crowded scenes and multi-person frames because the detector focuses the classifier on face regions. Transfer learning using pretrained backbones such as MobileNetV2 or ResNet50 is widely adopted to improve accuracy while lowering training time and compute needs; MobileNet variants are particularly popular for edge deployment because of their favorable accuracy/latency tradeoffs.

- Chollet, F. *Deep Learning with Python* — good background on Keras/CNN workflows.
- Howard et al., *MobileNet* — for lightweight backbones suitable for edge devices.
- He et al., *ResNet* — for deeper residual networks often used in transfer learning.

13. CONCLUSION:

The face mask detection project successfully demonstrates the use of Convolutional Neural Networks (CNNs) to automatically classify facial images as `With Mask` or `Without Mask`. Through careful data preprocessing, model design, and training, the system achieves high accuracy, precision, recall, and F1-score. The project highlights the effectiveness of CNNs in extracting spatial and hierarchical features from images, making them suitable for real-world applications such as surveillance, access control, and public health monitoring. Challenges such as partial mask usage, occlusions, and varying lighting conditions were identified, providing insights into potential areas for improvement. Overall, the project proves that deep learning-based mask detection can be a reliable, automated solution for monitoring compliance in various settings.

14. REFERENCES:

- Chollet, F. (2017). *Deep Learning with Python*. Manning Publications.
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.
- Howard, A. G., et al. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv preprint arXiv:1704.04861*.
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *CVPR 2016*.
- Kaggle. (2020). Face Mask Detection Dataset. https://www.kaggle.com
- TensorFlow Documentation. (2025). Image Classification with Keras. https://www.tensorflow.org/

15. SUSTAINABLE DEVELOPMENT:

This face mask detection project strongly aligns with the goals of sustainable development by addressing public health, innovation, and responsible technology use. It supports **Sustainable Development Goal (SDG) 3 – Good Health and Well-Being** by promoting safety and hygiene in public spaces through intelligent monitoring of mask compliance. During health crises such as pandemics, this technology can play a vital role in reducing disease transmission by ensuring preventive measures are followed effectively. The system acts as an assistive tool for authorities and organizations to maintain healthy environments, especially in hospitals, educational institutions, and public transport systems.

Furthermore, the project contributes to **SDG 9 – Industry, Innovation, and Infrastructure** by showcasing how artificial intelligence and computer vision can be applied to build innovative and efficient monitoring systems. The use of lightweight deep learning models, such as MobileNet, supports **energy-efficient computing** and can be deployed on low-power edge devices. This reduces dependency on high-resource cloud infrastructure, leading to lower carbon emissions and sustainable use of technology resources.

The system also aligns with **SDG 11 – Sustainable Cities and Communities**, as it can be integrated into smart city initiatives for maintaining public safety and promoting responsible behavior. Implementing such AI-based systems in urban areas enhances the resilience and sustainability of cities by preparing them to manage future health emergencies more effectively.

16. Appendix:

The appendix includes the key sections of the Google Colab notebook used for this project, providing step-by-step guidance on reproducing the results:

1. **Data Loading and Preprocessing:** Code for loading images, resizing, normalization, label encoding, splitting into training/validation/test sets, and applying data augmentation.
2. **CNN Model Definition:** Implementation of the CNN architecture, including convolutional layers, pooling layers, fully connected layers, dropout, and output layer.
3. **Model Compilation and Training:** Code for compiling the model with the Adam optimizer, defining loss functions, setting batch size and epochs, and using callbacks like EarlyStopping and ModelCheckpoint.
4. **Evaluation and Metrics:** Scripts for calculating accuracy, precision, recall, F1-score, generating confusion matrices, and plotting training/validation curves.
5. **Visualization:** Examples of visualizing correctly classified and misclassified images to understand model behavior.