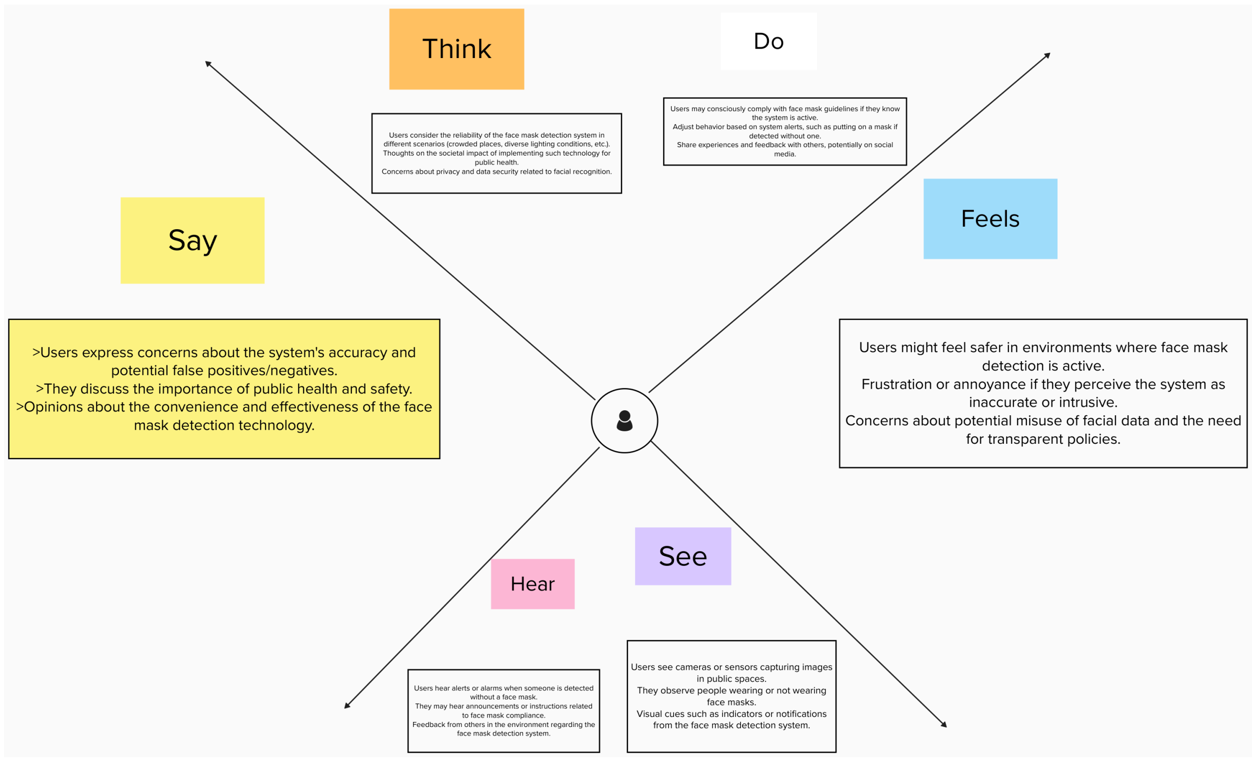
Face Mask Detection using CNN

Empathy map canvas:

Certainly! Here's an empathy map canvas specifically tailored to the context of a face mask detection project using machine learning:

1. See:
   - Users see cameras or sensors capturing images in public spaces.
   - They observe people wearing or not wearing face masks.
   - Visual cues such as indicators or notifications from the face mask detection system.
2. Hear:
   - Users hear alerts or alarms when someone is detected without a face mask.
   - They may hear announcements or instructions related to face mask compliance.
   - Feedback from others in the environment regarding the face mask detection system.
3. Say:
   - Users express concerns about the system's accuracy and potential false positives/negatives.
   - They discuss the importance of public health and safety.
   - Opinions about the convenience and effectiveness of the face mask detection technology.
4. Think:
   - Users consider the reliability of the face mask detection system in different scenarios (crowded places, diverse lighting conditions, etc.).
   - Thoughts on the societal impact of implementing such technology for public health.
   - Concerns about privacy and data security related to facial recognition.
5. Do:
   - Users may consciously comply with face mask guidelines if they know the system is active.
   - Adjust behavior based on system alerts, such as putting on a mask if detected without one.
   - Share experiences and feedback with others, potentially on social media.
6. Feel:
   - Users might feel safer in environments where face mask detection is active.
   - Frustration or annoyance if they perceive the system as inaccurate or intrusive.
   - Concerns about potential misuse of facial data and the need for transparent policies.

An empathy map is a living document that should evolve as you gather more insights from users, stakeholders, and the deployment of the face mask detection system. Regularly update the map to ensure that the technology aligns with the user's experiences and expectations.



Users consider the reliability of the face mask detection system in different scenarios (crowded places, diverse lighting conditions, etc.).
Thoughts on the societal impact of implementing such technology for public health.
Concerns about privacy and data security related to facial recognition.

Users may consciously comply with face mask guidelines if they know the system is active.
Adjust behavior based on system alerts, such as putting on a mask if detected without one.
Share experiences and feedback with others, potentially on social media.

>Users express concerns about the system's accuracy and potential false positives/negatives.
>They discuss the importance of public health and safety.
>Opinions about the convenience and effectiveness of the face mask detection technology.

Users might feel safer in environments where face mask detection is active.
Frustration or annoyance if they perceive the system as inaccurate or intrusive.
Concerns about potential misuse of facial data and the need for transparent policies.

Users hear alerts or alarms when someone is detected without a face mask.
They may hear announcements or instructions related to face mask compliance.
Feedback from others in the environment regarding the face mask detection system.

Users see cameras or sensors capturing images in public spaces.
They observe people wearing or not wearing face masks.
Visual cues such as indicators or notifications from the face mask detection system.

Brainstorm and Idea Prioritization:

Brainstorming and idea prioritization are essential steps in developing a face mask detection system. The approach is as follows :

Brainstorming Ideas:

1. Multimodal Detection:
   - Combine visual data from cameras with other sensor data (thermal, infrared) for more accurate detection.
2. Real-time Feedback:
   - Provide instant feedback to individuals without masks through visual alerts or notifications.
3. Mobile App Integration:
   - Develop a mobile app for users to receive alerts and track their compliance history.
4. Privacy Filters:
   - Implement privacy filters or techniques to ensure that only mask presence is detected without storing or transmitting facial features.
5. Edge Computing:
   - Explore edge computing solutions to process data locally, reducing latency and dependence on centralized servers.
6. Integration with Access Control Systems:
   - Integrate face mask detection with access control systems to regulate entry into

certain spaces.
7. Adaptive Learning:
   - Implement machine learning models that adapt and improve over time based on new data and feedback.
8. Cultural Sensitivity:
   - Consider cultural norms regarding facial coverings to avoid biases in the detection algorithm.
9. Dashboard for Monitoring:
   - Create a centralized dashboard for administrators to monitor real-time compliance in various locations.
10. Public Awareness Campaign:
    - Develop a campaign to educate the public about the benefits of face mask detection for public health.

Idea Prioritization:
1. Critical Functionality:
   - Prioritize features directly related to the accurate detection of face masks to ensure the system's primary function is robust.
2. User Experience:
   - Give priority to features that enhance user experience, such as clear and non-intrusive alerts.
3. Privacy and Security Measures:
   - Place a high priority on implementing privacy and security measures to address user concerns.
4. Regulatory Compliance:
   - Ensure that the system aligns with data protection laws and ethical guidelines, making it compliant with relevant regulations.
5. Integration with Existing Systems:
   - If the system will be integrated into existing infrastructure, prioritize compatibility and ease of integration.
6. Scalability:
   - Consider the scalability of the system to handle varying crowd sizes and different deployment environments.
7. Continuous Improvement:
   - Prioritize features that support ongoing learning and improvement of the face mask detection model.
8. Cost-effectiveness:

   - Evaluate the cost-effectiveness of different features and prioritize those that provide the most value for the investment.
9. Public Perception:
   - Consider features that positively influence public perception and acceptance of the face mask detection system.
10. Feedback Mechanisms:
   - Prioritize features that enable the collection of user feedback for continuous refinement.

Certainly! Here's a template for a proposed solution for a face mask detection system:

Title: Face Mask Detection System Proposal

Executive Summary:

Brief overview of the proposed face mask detection system, its objectives, and anticipated benefits.

Objectives:

1. Develop an accurate and reliable face mask detection system to enhance public health and safety.
2. Implement a user-friendly solution that integrates seamlessly into various environments.
3. Address privacy concerns through the responsible use of facial data and adherence to data protection regulations.

Key Features:

1. Real-time Detection:
   - Utilize advanced machine learning algorithms for real-time face mask detection.
   - Achieve high accuracy in diverse scenarios, including crowded places and varying lighting conditions.

2. Multimodal Integration:
   - Combine visual data from cameras with other sensor data (thermal, infrared) for enhanced accuracy.
   - Explore the use of edge computing for faster processing and reduced dependence on

centralized servers.

3. User Feedback Mechanisms:
   - Provide instant visual and/or auditory feedback to individuals without masks.
   - Develop a mobile application for users to receive notifications and track their compliance history.

4. Privacy Measures:
   - Implement privacy filters to ensure only mask presence is detected without storing or transmitting facial features.
   - Adhere to strict privacy and security protocols, including encryption and secure data transmission.

5. Integration with Access Control Systems:
   - Integrate the face mask detection system with access control systems for regulated entry into certain spaces.
   - Explore compatibility with existing infrastructure for seamless integration.

Implementation Plan:

1. Phase 1: Research and Development
   - Conduct a thorough analysis of existing face mask detection technologies.
   - Identify and evaluate machine learning models suitable for accurate detection.

2. Phase 2: Prototype Development
   - Develop a prototype of the face mask detection system.
   - Test the prototype in controlled environments to assess accuracy and reliability.

3. Phase 3: Integration and Testing
   - Integrate the system with different camera and sensor types.
   - Conduct extensive testing in real-world scenarios to validate performance.

4. Phase 4: Privacy and Security Compliance
   - Implement robust privacy measures to address user concerns.
   - Ensure compliance with data protection laws and ethical guidelines.

5. Phase 5: User Interface and Experience Enhancement

- Design and implement a user-friendly interface for administrators and end-users.
- Optimize feedback mechanisms to enhance user experience.

6. Phase 6: Deployment and Monitoring
   - Deploy the face mask detection system in pilot locations.
   - Establish a centralized monitoring dashboard for administrators.

Evaluation and Feedback:

1. Conduct regular evaluations to measure the accuracy and reliability of the system.
2. Gather user feedback through surveys and other feedback mechanisms.
3. Iterate on the solution based on continuous improvement principles.

Budget and Resources:

Provide an estimate of the budget required for the development, testing, and deployment of the face mask detection system. Outline the key resources, including personnel and technology infrastructure.

Conclusion:
This template serves as a starting point and can be customized based on the specific requirements and context of your face mask detection project.

Face Mask Detection Using CNN: Solution Architecture

1. Data Collection:
   - Collect a diverse dataset of images containing individuals with and without face masks.
   - Annotate the dataset to indicate the presence or absence of masks.

2. Data Preprocessing:
   - Resize images to a consistent resolution.
   - Normalize pixel values to a common scale (e.g., 0 to 1).
   - Augment the dataset through techniques like rotation, flipping, and zooming to increase diversity.

3. Model Architecture:

- **Input Layer:**
  - Accept input images with dimensions consistent with the preprocessed dataset.
- Use multiple channels for color images (e.g., three channels for RGB).

- **Convolutional Layers:**
  - Stack multiple convolutional layers for feature extraction.
  - Apply activation functions (e.g., ReLU) after each convolutional layer.

- **Pooling Layers:**
  - Intersperse pooling layers (e.g., MaxPooling) to reduce spatial dimensions and extract important features.

- **Flatten Layer:**
  - Flatten the output to a 1D vector for input to the fully connected layers.

- **Fully Connected Layers:**
  - Add one or more dense layers for classification.
  - Use dropout layers to reduce overfitting.

- **Output Layer:**
  - Single neuron with a sigmoid activation function for binary classification (mask or no mask).

4. Model Training:
   - Split the dataset into training, validation, and test sets.
   - Train the CNN using backpropagation and optimization algorithms (e.g., Adam).
   - Monitor metrics such as accuracy, precision, recall, and F1 score during training.

5. Model Evaluation:
   - Evaluate the trained model on the test set to assess its generalization performance.
   - Fine-tune the model if needed based on evaluation results.

6. Deployment:
   - Convert the trained model to a deployable format (e.g., TensorFlow SavedModel or ONNX format).
   - Deploy the model on an inference server or edge device capable of real-time processing.

7. Integration with Cameras:
  - Interface the deployed model with cameras or video streams for real-time face mask detection.
  - Ensure compatibility with different camera types and resolutions.

8. User Interface :
  - Develop a user interface for administrators or end-users to visualize the detection results.
  - Include features for monitoring and managing the system.

9. Privacy and Security Measures:
  - Implement privacy filters to process images without storing or transmitting facial features.
  - Apply encryption and secure transmission protocols to protect data.

10. Continuous Improvement:
  - Set up a feedback loop to gather user feedback and update the model periodically for continuous improvement.

Note:
- This architecture provides a general framework and can be adapted based on specific requirements and constraints.
- The model complexity and the number of layers may need adjustment based on the size and complexity of the dataset.
- Consider ethical and legal aspects, including privacy concerns, when deploying face mask detection systems.

This architecture assumes a binary classification (mask or no mask). If more detailed information is required (e.g., mask types or incorrect mask usage), the architecture may need to be adjusted accordingly.

Face Mask Detection System - Data Flow Diagram:
A Data Flow Diagram (DFD) is a visual representation of the flow of data within a system. It's a useful tool for understanding the processes and interactions that occur. Here's a template for a DFD to determine the requirements of a face mask detection

system:

Level 0: System Overview

- **Processes:**
  - **Face Mask Detection System:**
    - Receives input from cameras/sensors.
    - Processes the data to detect face masks.
    - Provides feedback/alerts based on the detection results.
    - Interfaces with the user interface/dashboard.

- **Data Stores:**
  - **Face Mask Data Store:**
    - Stores historical data related to face mask detection.
  - **Configuration Data Store:**
    - Stores system configuration settings.

- **External Entities:**
  - **Camera/Sensor:**
    - Provides input images or data to the system.
  - **Administrator/User:**
    - Interacts with the system through the user interface.

- **Data Flows:**
  - **Input Data Flow:**
    - Camera/Sensor data to the Face Mask Detection System.
  - **Output Data Flow:**
    - Face mask detection results to the User Interface.
    - Face mask data to the Face Mask Data Store.
    - Configuration settings to the Face Mask Detection System.

Level 1: Face Mask Detection Process

- **Processes:**
  - **Capture and Preprocess:**
    - Captures images/data from cameras/sensors.

    - Preprocesses the data for face mask detection.
  - **Face Mask Detection Algorithm:**
    - Applies the CNN-based face mask detection algorithm.
    - Outputs detection results.
  - **Alert Generation:**
    - Generates alerts or notifications based on detection results.
  - **User Interface Integration:**
    - Integrates with the user interface to display results.

- **Data Stores:**
  - *(Same as Level 0)*

- **External Entities:**
  - *(Same as Level 0)*

- **Data Flows:**
  - **Input Data Flow:**
    - Camera/Sensor data to Capture and Preprocess.
  - **Intermediate Data Flows:**
    - Preprocessed data to Face Mask Detection Algorithm.
    - Detection results to Alert Generation and User Interface Integration.
  - **Output Data Flows:**
    - Alerts/notifications to the User Interface.
    - Face mask detection results to the Face Mask Data Store.

Level 2: Face Mask Data Store

- **Processes:**
  - *(Same as Level 1)*

- **Data Stores:**
  - **Face Mask Data Store:**
    - *(Same as Level 0)*

- **External Entities:**
  - *(Same as Level 0)*

- **Data Flows:**
  - **Input Data Flow:**
    - Face mask detection results to the Face Mask Data Store.

#### Level 2: User Interface Integration

- **Processes:**
  - *(Same as Level 1)*

- **Data Stores:**
  - *(Same as Level 0)*

- **External Entities:**
  - *(Same as Level 0)*

- **Data Flows:**
  - **Input Data Flow:**
    - Detection results to User Interface Integration.
  - **Output Data Flow:**
    - Alerts/notifications to the User Interface.

Technology Stack details: Face Mask Detection
### a) Technical Architecture:

A typical technical architecture for face mask detection involves several components working together. Here's a simplified overview:

1. **Input Source:**
   - Cameras and/or sensors capture images or video feeds.

2. **Preprocessing:**
   - The input data undergoes preprocessing, including resizing, normalization, and augmentation to prepare it for the model.

3. **Face Mask Detection Model:**
   - A Convolutional Neural Network (CNN) processes the preprocessed data to detect the presence or absence of face masks.

4. **Output Processing:**
   - The model's output is processed to generate alerts or notifications based on the detection results.

5. **User Interface:**
   - A user interface displays real-time results, alerts, and potentially historical data for administrators or end-users.

6. **Storage:**
   - Face mask detection results and related data can be stored for analysis, monitoring, and future improvements.

7. **Integration:**
   - Integration points with external systems, such as access control or alarm systems, for a more comprehensive solution.

8. **Security and Privacy Measures:**
   - Implement encryption, secure data transmission, and privacy filters to address security and privacy concerns.

### b) Open Source Frameworks:

1. **TensorFlow:**
   - TensorFlow is an open-source machine learning framework widely used for building and training deep learning models, including face mask detection models.

2. **PyTorch:**
   - PyTorch is another popular deep learning framework known for its flexibility and dynamic computational graph, making it suitable for face mask detection tasks.

3. **OpenCV:**
   - OpenCV (Open Source Computer Vision Library) provides tools for image and video processing, which can be used for tasks like image preprocessing in face mask detection.

4. **Keras:**

   - Keras is a high-level neural networks API that runs on top of TensorFlow. It simplifies the process of building and training neural networks, including face mask detection models.

### c) Third-party APIs:

1. **Microsoft Azure Cognitive Services:**
   - Offers pre-trained models for face detection, which can be customized for face mask detection.

2. **Google Cloud Vision API:**
   - Provides powerful image analysis capabilities, including face detection, which can be utilized for face mask detection.

3. **AWS Rekognition:**
   - Amazon Rekognition offers facial analysis features, allowing developers to build applications for face detection, analysis, and recognition.

### d) Cloud Deployment:

1. **AWS (Amazon Web Services):**
   - Utilize AWS for cloud deployment, leveraging services like Amazon EC2 for hosting, Amazon S3 for storage, and AWS Lambda for serverless computing.

2. **Microsoft Azure:**
   - Azure provides services such as Azure Virtual Machines, Azure Blob Storage, and Azure Functions for deploying and running face mask detection applications.

3. **Google Cloud Platform (GCP):**
   - GCP offers resources like Google Compute Engine, Google Cloud Storage, and Cloud Functions for deploying and scaling face mask detection systems.

4. **Edge Computing :**
   - Consider deploying the model on edge devices (e.g., IoT devices, edge servers) for real-time processing without relying solely on cloud resources.

Face Mask Detection: Project Planning.

Project planning for face mask detection involves defining the project scope, identifying key tasks, allocating resources, setting timelines, and establishing a framework for project execution. Here's the step-by-step project planning:

1. **Define Project Objectives:**
   - Clearly articulate the goals and objectives of the face mask detection project. Identify the problem statement and the desired outcomes.

2. **Scope Definition:**
   - Define the boundaries of the project. Specify the features and functionalities of the face mask detection system. Determine what is in and out of scope.

3. **Stakeholder Identification:**
   - Identify all stakeholders involved, including project sponsors, end-users, developers, and any other relevant parties.

4. **Risk Assessment:**
   - Identify potential risks and challenges associated with the project. Develop a risk mitigation plan to address and minimize these risks.

5. **Resource Allocation:**
   - Allocate human resources, including project managers, developers, and testers.
   - Identify hardware, software, and other resources required for the project.

6. **Technology Selection:**
   - Choose the technology stack, including the programming language, frameworks, and libraries for face mask detection.

7. **Data Collection and Annotation:**
   - Plan the collection of a diverse dataset containing images with and without face masks.
   - Establish a process for annotating the dataset for training and testing the model.

8. **Model Development:**
   - Outline the steps for developing the face mask detection model, including data preprocessing, model architecture selection, and training.

9. **Testing and Validation:**
   - Develop a testing plan to validate the accuracy and reliability of the face mask detection model.
   - Include unit testing, integration testing, and real-world scenario testing.

10. **User Interface Design:**
   - Plan the design and development of the user interface for administrators and end-users.
   - Consider features such as real-time monitoring, alert displays, and historical data visualization.

11. **Privacy and Security Measures:**
   - Plan and implement privacy filters, encryption, and other security measures to address privacy concerns associated with facial data.

12. **Integration with External Systems:**
   - If applicable, plan the integration of the face mask detection system with external systems such as access control or alarm systems.

13. **Deployment Strategy:**
   - Develop a deployment plan outlining the steps for deploying the face mask detection system in different environments.
   - Consider whether deployment will be on the cloud, edge devices, or a combination of both.

14. **User Training and Documentation:**
   - Develop a plan for training end-users and administrators on how to use the face mask detection system.
   - Create documentation for system configuration, troubleshooting, and maintenance.

15. **Monitoring and Continuous Improvement:**
   - Plan for ongoing monitoring of the system's performance.
   - Establish procedures for collecting user feedback and implementing continuous improvements to the model.

16. **Compliance and Ethics:**
   - Ensure that the project adheres to relevant data protection laws, ethical guidelines,

and industry standards.

17. **Project Timeline:**
   - Develop a detailed project timeline with milestones and deadlines for each phase of the project.

18. **Communication Plan:**
   - Define a communication plan outlining how project updates will be communicated to stakeholders.

19. **Budgeting:**
   - Develop a budget that includes costs associated with resources, technology, training, and any other project-related expenses.

20. **Documentation:**
   - Establish a documentation strategy for recording project specifications, design decisions, and other important information.

21. **Project Kickoff:**
   - Conduct a project kickoff meeting to ensure that all team members understand the project objectives, roles, and responsibilities.

22. **Project Management Tools:**
   - Choose and set up project management tools for tasks, timelines, and collaboration (e.g., Jira, Trello, or Asana).

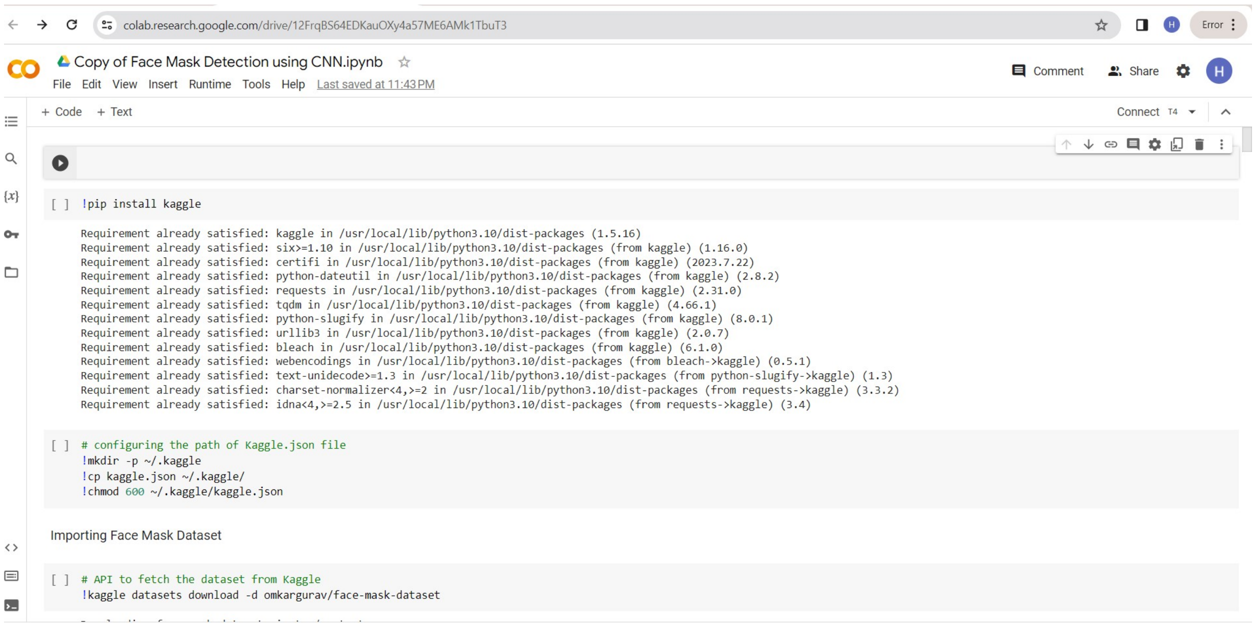
23. **Quality Assurance:**
   - Implement a quality assurance plan to ensure that the face mask detection system meets the defined standards and specifications.

24. **Legal and Regulatory Compliance:**
   - Ensure compliance with legal and regulatory requirements, especially concerning privacy and data protection.

25. **Project Review and Closure:**
   - Plan for a project review at the end to evaluate the success of the face mask detection system against the defined objectives.

- Document lessons learned and ensure a smooth transition for ongoing maintenance and support.

Performance and Final Submission:
Screenshots

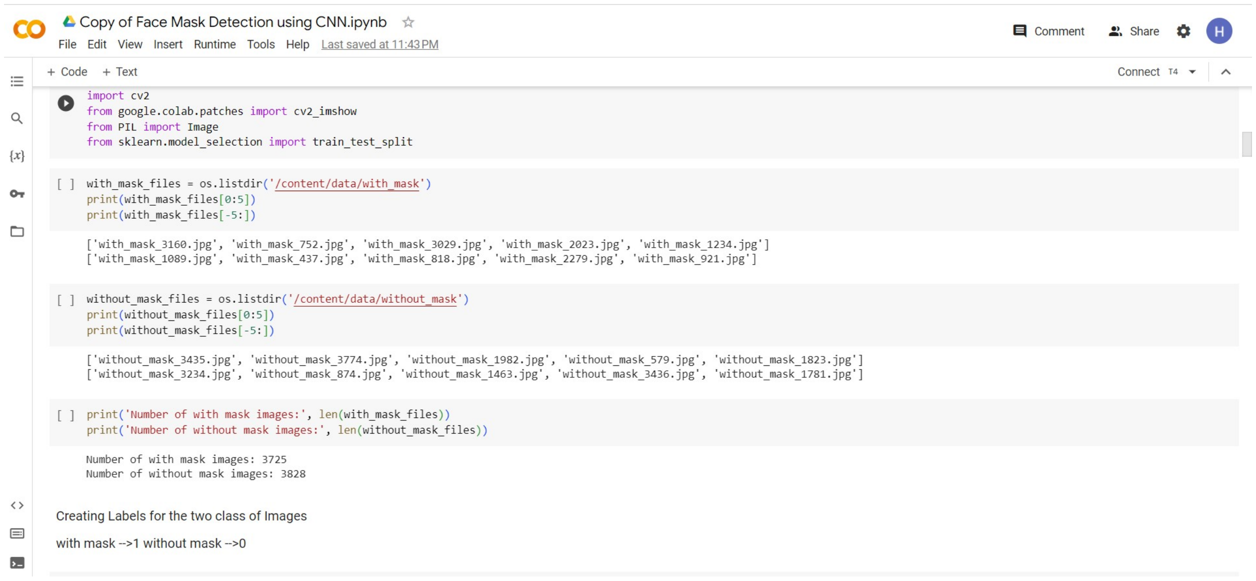

colab.research.google.com/drive/12FrqBS64EDKauOXy4a57ME6AMk1TbuT3
Copy of Face Mask Detection using CNN.ipynb
File Edit View Insert Runtime Tools Help Last saved at 11:43 PM
Comment Share
+ Code + Text
Connect T4

```
!pip install kaggle

Requirement already satisfied: kaggle in /usr/local/lib/python3.10/dist-packages (1.5.16)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.10/dist-packages (from kaggle) (1.16.0)
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages (from kaggle) (2023.7.22)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.10/dist-packages (from kaggle) (2.8.2)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from kaggle) (2.31.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from kaggle) (4.66.1)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.10/dist-packages (from kaggle) (8.0.1)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.10/dist-packages (from kaggle) (2.0.7)
Requirement already satisfied: bleach in /usr/local/lib/python3.10/dist-packages (from kaggle) (6.1.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.10/dist-packages (from bleach->kaggle) (0.5.1)
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.10/dist-packages (from python-slugify->kaggle) (1.3)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->kaggle) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->kaggle) (3.4)
```

```
# configuring the path of Kaggle.json file
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
```

Importing Face Mask Dataset

```
# API to fetch the dataset from Kaggle
!kaggle datasets download -d omkargurav/face-mask-dataset
```


Copy of Face Mask Detection using CNN.ipynb
File Edit View Insert Runtime Tools Help Last saved at 11:43 PM
Comment Share
+ Code + Text
Connect T4

```
import cv2
from google.colab.patches import cv2_imshow
from PIL import Image
from sklearn.model_selection import train_test_split
```

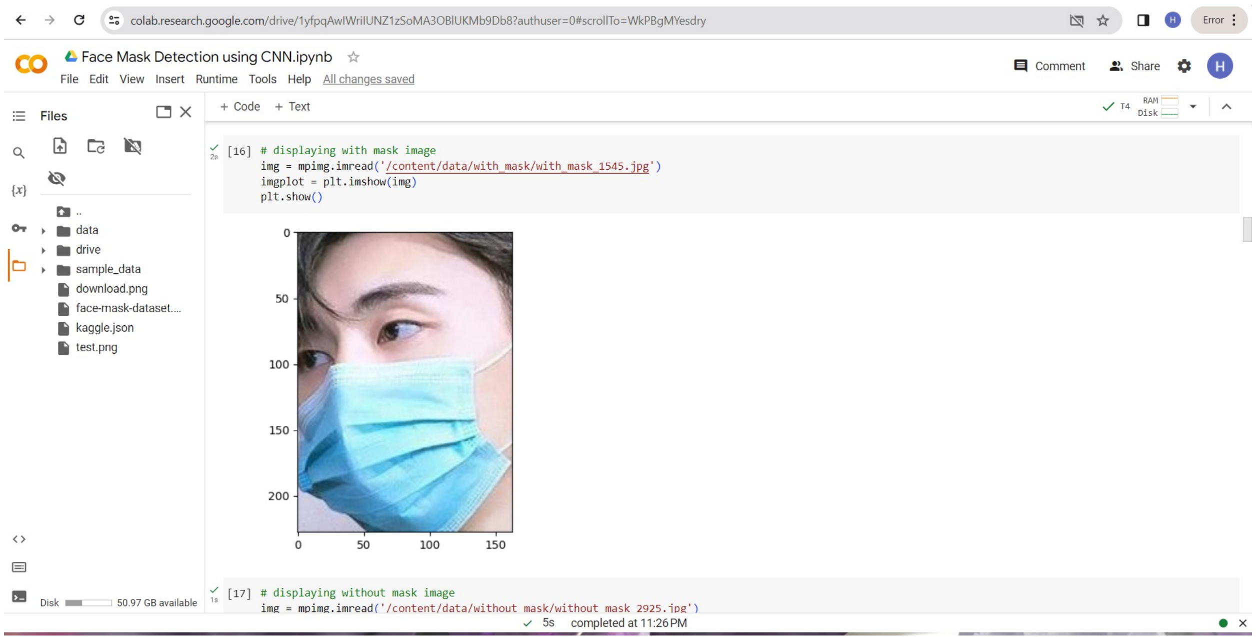
```
with_mask_files = os.listdir('/content/data/with_mask')
print(with_mask_files[0:5])
print(with_mask_files[-5:])

['with_mask_3160.jpg', 'with_mask_752.jpg', 'with_mask_3029.jpg', 'with_mask_2023.jpg', 'with_mask_1234.jpg']
['with_mask_1089.jpg', 'with_mask_437.jpg', 'with_mask_818.jpg', 'with_mask_2279.jpg', 'with_mask_921.jpg']
```

```
without_mask_files = os.listdir('/content/data/without_mask')
print(without_mask_files[0:5])
print(without_mask_files[-5:])

['without_mask_3435.jpg', 'without_mask_3774.jpg', 'without_mask_1982.jpg', 'without_mask_579.jpg', 'without_mask_1823.jpg']
['without_mask_3234.jpg', 'without_mask_874.jpg', 'without_mask_1463.jpg', 'without_mask_3436.jpg', 'without_mask_1781.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
```

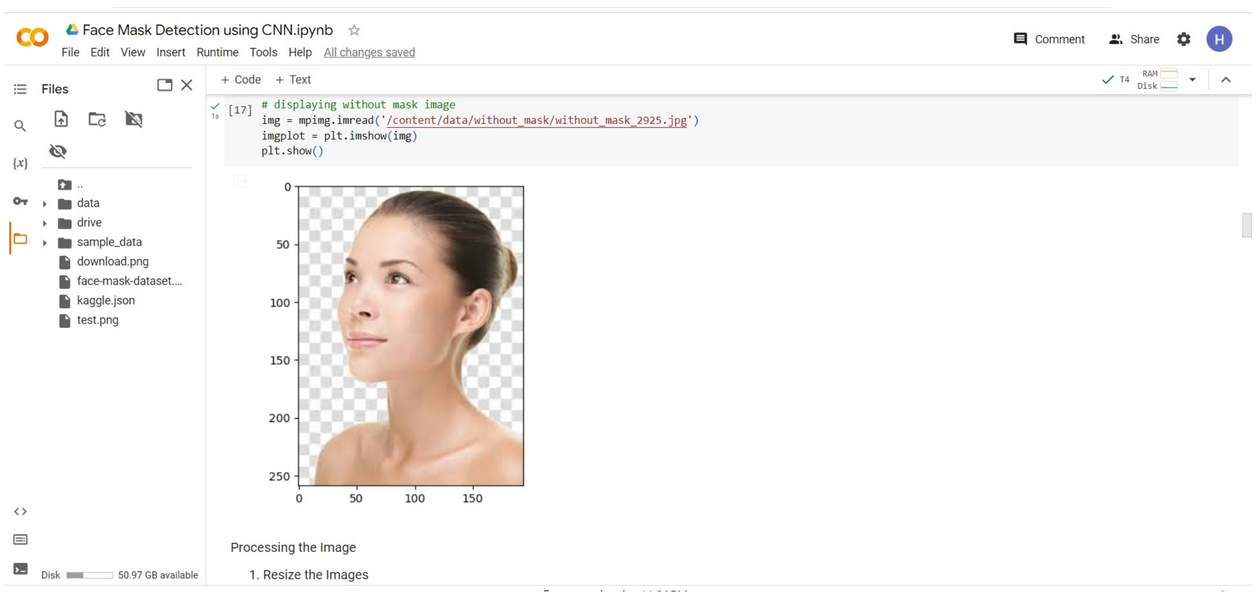
Creating Labels for the two class of Images

with mask -->1 without mask -->0


colab.research.google.com/drive/1yfpqAwIWrilUNZ1zSoMA3OBlUKMb9Db8?authuser=0#scrollTo=WkPBgMYesdry
Face Mask Detection using CNN.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Files
data
drive
sample_data
download.png
face-mask-dataset....
kaggle.json
test.png
Disk 50.97 GB available
5s completed at 11:26 PM

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

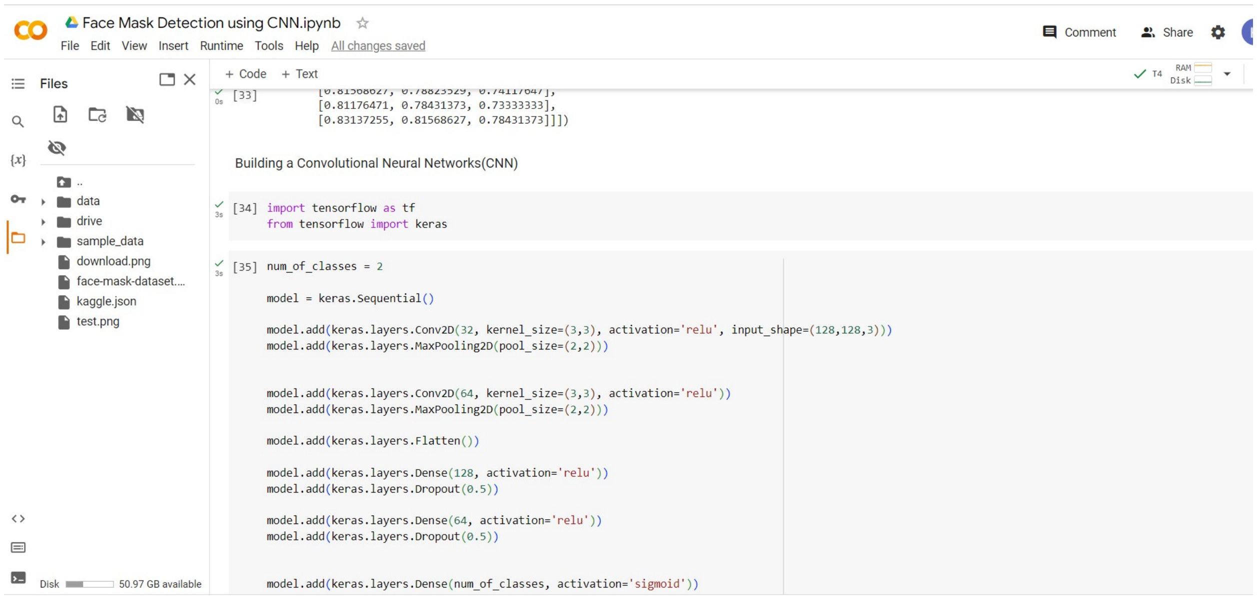
```
[17] # displaying without mask image
     img = mpimg.imread('/content/data/without_mask/without_mask_2925.jpg')
```



```
[17] # displaying without mask image
     img = mpimg.imread('/content/data/without_mask/without_mask_2925.jpg')
     imgplot = plt.imshow(img)
     plt.show()
```

Processing the Image

1. Resize the Images

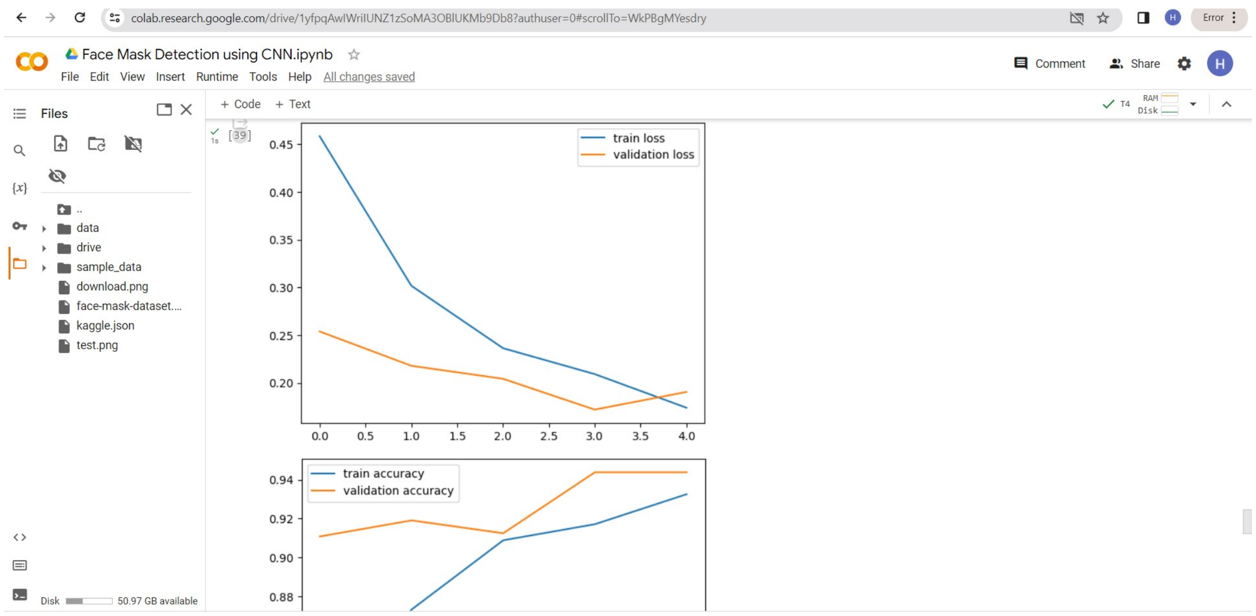


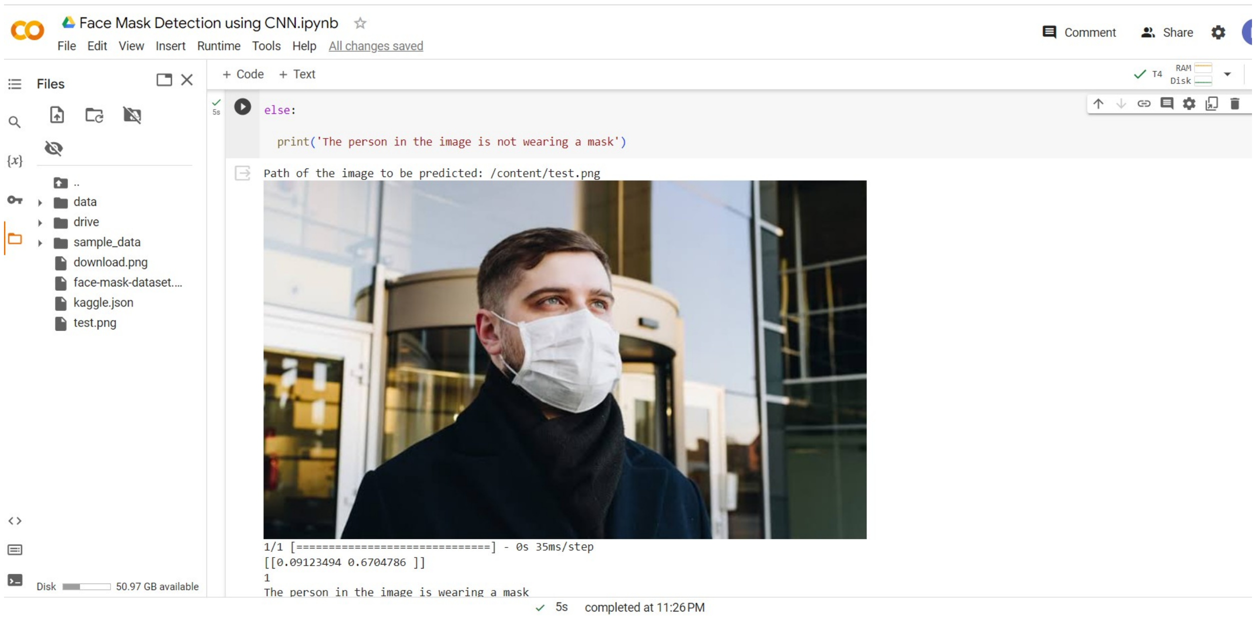
```
[33]         [0.81568627, 0.78823529, 0.74117647],
             [0.81176471, 0.78431373, 0.73333333],
             [0.83137255, 0.81568627, 0.78431373]]])
```

Building a Convolutional Neural Networks(CNN)

```
[34] import tensorflow as tf
     from tensorflow import keras
```

```
[35] 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'))
```





```
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 35ms/step
[[0.09123494 0.6704786 ]]
1
The person in the image is wearing a mask
```

Project link(Google colab):
https://colab.research.google.com/drive/1yfpqAwIWriIUNZ1zSoMA3OBlUKMb9Db8?usp=sharing

Thanking you
Diptonil Das
21BCB0044 VIT VELLORE