Project Title

<u>Glasses Detection System using Deep Learning</u>

Problem Definition (Week 4 Task 1)

In Week 4, I proposed and implemented a real-world AI application focused on visual recognition: detecting whether a person is wearing glasses or not from an image.

Objectives:

- Identify the presence of glasses on a person's face using a single image.
- Count and highlight people wearing glasses in group photographs.
- Extend the system to real-time detection using a webcam feed.

Use Cases:

- Attendance systems that require visual authentication.
- Surveillance systems to track individuals wearing or not wearing glasses.
- Assistive technologies in healthcare and education.

Dataset Used (Week 4 Task 2)

I utilized the publicly available "Glasses or No Glasses" dataset by Jeff Heaton.

Structure:

- Contains 5,000 facial images labeled from face-1.png to face-5000.png.
- Two CSV files accompany the dataset:
  - train.csv: used for model training (images 1 to 4500).
  - test.csv: used for model evaluation (images 4501 to 5000).
- Each CSV row includes an id and a glasses label (1 = wearing glasses, 0 = not wearing).

AI Pipeline and Implementation (Week 4 Task 3)

1. Initial Haarcascade Classifier Approach

I began with a classical computer vision approach using OpenCV's Haarcascade classifiers:

- Used haarcascade_frontalface_default.xml to detect faces.
- Used haarcascade_eye_tree_eyeglasses.xml to detect glasses-like regions.

Challenges Faced:

- The classifier actually detected eye regions, not glasses directly.
- This resulted in high false positives when glasses weren't present.

Solution:

- Added dynamic padding around detected eye regions to approximate a bounding box around glasses.
- While visually useful in simple cases, this method lacked robustness in diverse real-world scenarios.

Decision Justification:
I initially avoided training an object detection model like YOLO due to its requirement for labeled bounding box data. Haarcascade served as a functional but limited shortcut for demonstration.

---

2. Real-Time Detection via Webcam (Local Testing Only)

I implemented OpenCV-based webcam integration to test real-time glasses detection locally using the Haar-based method.

Technical Limitation:

- Google Colab does not support direct access to webcams.

Solution:

- I moved this part of the project to a local Python environment (PyCharm) and successfully ran real-time face and glasses detection with bounding boxes and labels.

---

3. CNN-Based Glasses Classifier (Deep Learning Model)

I then transitioned to a deep learning-based binary image classifier using TensorFlow and Keras.

Training Setup:

- Preprocessed all images to a uniform size of 100x100 pixels.
- Used only the training set (train.csv) and dynamically created image paths using the id column.
- Constructed a custom CNN architecture with the following layers:
    - Three convolutional layers with ReLU activation.
    - Batch Normalization and MaxPooling after each convolutional block.
    - Flatten layer followed by Dense and Dropout.
    - Final Dense output layer with sigmoid activation for binary classification.

Compilation:

- Loss: Binary Crossentropy
- Optimizer: Adam (learning rate: 0.0001)
- Metrics: Accuracy

Output:

- The trained model was saved as glasses_cnn_model.h5.

Challenges Faced During Model Training and Testing

File Path and Directory Errors:

- Windows file paths caused errors due to escape sequences like \V.
- Solved by converting all paths into raw strings (prefixing with r"").

Column Mismatch:

- Initial assumption of a file column in the CSV led to a KeyError.
- Corrected by generating filenames from the id column using a lambda function.

Input Shape Mismatch in Predictions:

- The trained model expected input shape matching 100x100 images.
- During testing, different image dimensions caused prediction failures.
- Resolved by resizing input images properly in both training and testing phases.

Final Testing and Integration in Google Colab

I deployed the trained CNN model in Colab to test it with new uploaded images:

Steps Followed:

1. Uploaded the glasses_cnn_model.h5 file.
2. Uploaded test images.
3. Preprocessed test images to 100x100 resolution.
4. Used Haarcascade to detect faces and possible eye regions.
5. If the CNN predicted "Wearing Glasses" and eye regions were found, a red bounding box was drawn around the glasses area using dynamic padding.
6. Displayed prediction label along with bounding boxes using Matplotlib.

Remaining Challenges:

- In a few edge cases, the Haarcascade failed to detect eye regions even when glasses were worn.
- The label "Wearing Glasses" was shown correctly, but no red box appeared.

Improvement:

- Updated logic to ensure the red box is drawn only when both the model's prediction is positive and the Haarcascade detects glasses.
- Prevented duplicate text by drawing the label only once per face.

Final Model Evaluation using Test Set

I used test.csv (images 4501–5000) to evaluate the model's performance:

- Achieved satisfactory prediction accuracy on previously unseen images.

- Visual testing showed the model generalized well to real faces with varied features.

Webcam Integration with CNN Model (Local Testing)

I successfully extended the CNN model into a local real-time webcam application:

- Captured live frames using OpenCV.
- Used Haarcascade to detect faces and eye regions.
- For each detected face, cropped and resized the face region.
- Predicted using the trained CNN whether glasses are present.
- If predicted positive, detected glasses region was highlighted with a red box and labeled "Wearing Glasses".

This provided a real-time demonstration of the model's practical utility.

Final Results and Achievements

Baseline (Haarcascade only):

- Easy to implement but limited in accuracy.
- Useful for prototyping and simple applications.

Deep Learning-Based Classifier:

- Significantly improved accuracy and reliability.
- Easily reusable via .h5 file in different environments.
- Combined with Haarcascade for bounding box overlays in final visualization.

Overall Achievements:

- Built and trained a functional CNN from scratch.
- Handled and resolved real-world data issues (paths, input shapes, etc.).
- Demonstrated working prediction on uploaded and real-time webcam images.

Real-World Applicability

This project has potential use in several practical domains:

- In education: Detecting student attentiveness or enforcing visual identification.
- In surveillance: Detecting or filtering people with or without glasses.
- In healthcare: Monitoring patients or users who require vision correction.
- In retail: Smart mirror systems that track users wearing glasses for AR try-on.

This end-to-end deep learning system demonstrates how even a simple classification task can be elevated with preprocessing, computer vision, model training, and smart integration.