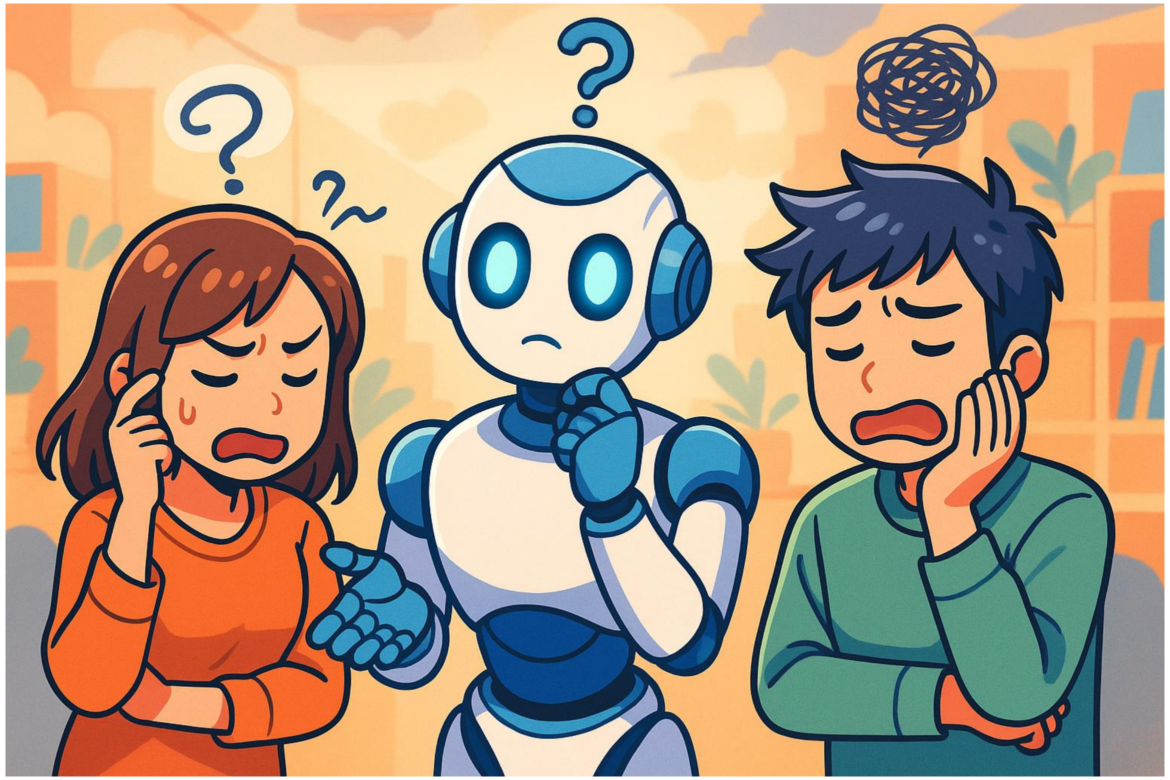
EMOTION RECOGNITION FROM FACES

WILLIE LEE BREAUX III

TIER 2



Problem: Machines can recognize faces but cannot understand human emotion.

Who Cares: Healthcare, customer service, education, and security that relies on human AI interaction.

Why is it Important: AI's ability to understand human emotion enables a natural, empathetic, and safer interaction between the two.

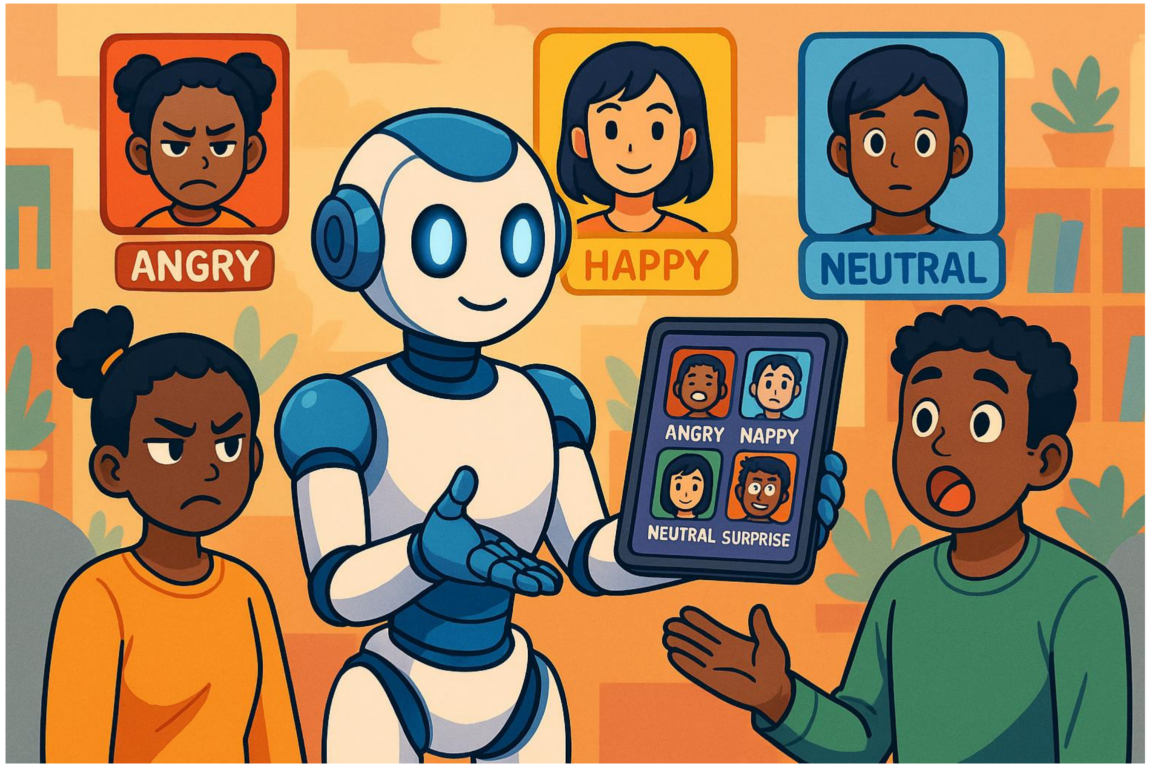
THE PROBLEM

SOLUTION OVERVIEW

Solution: Detect human facial emotions and classify them in real time.

How: By using a YOLO model trained on labeled facial emotion data to help machines accurately interpret human emotion.

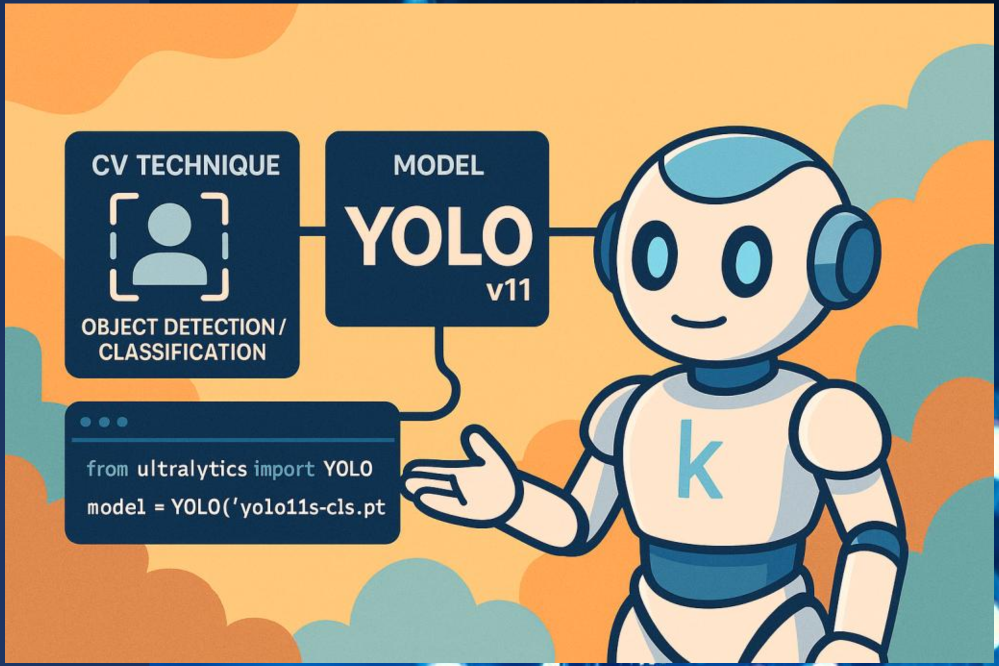
A Computer vision model system that will help machines recognize and respond to human expressions through facial emotion analysis.



TECHNICAL APPROACH

- CV Technique: Object Detection/ Classification
- Model: YOLOv11 Small (Ultralytics)
- Framework: Kaggle Networks with PyTorch, Ultralytics, OpenCV

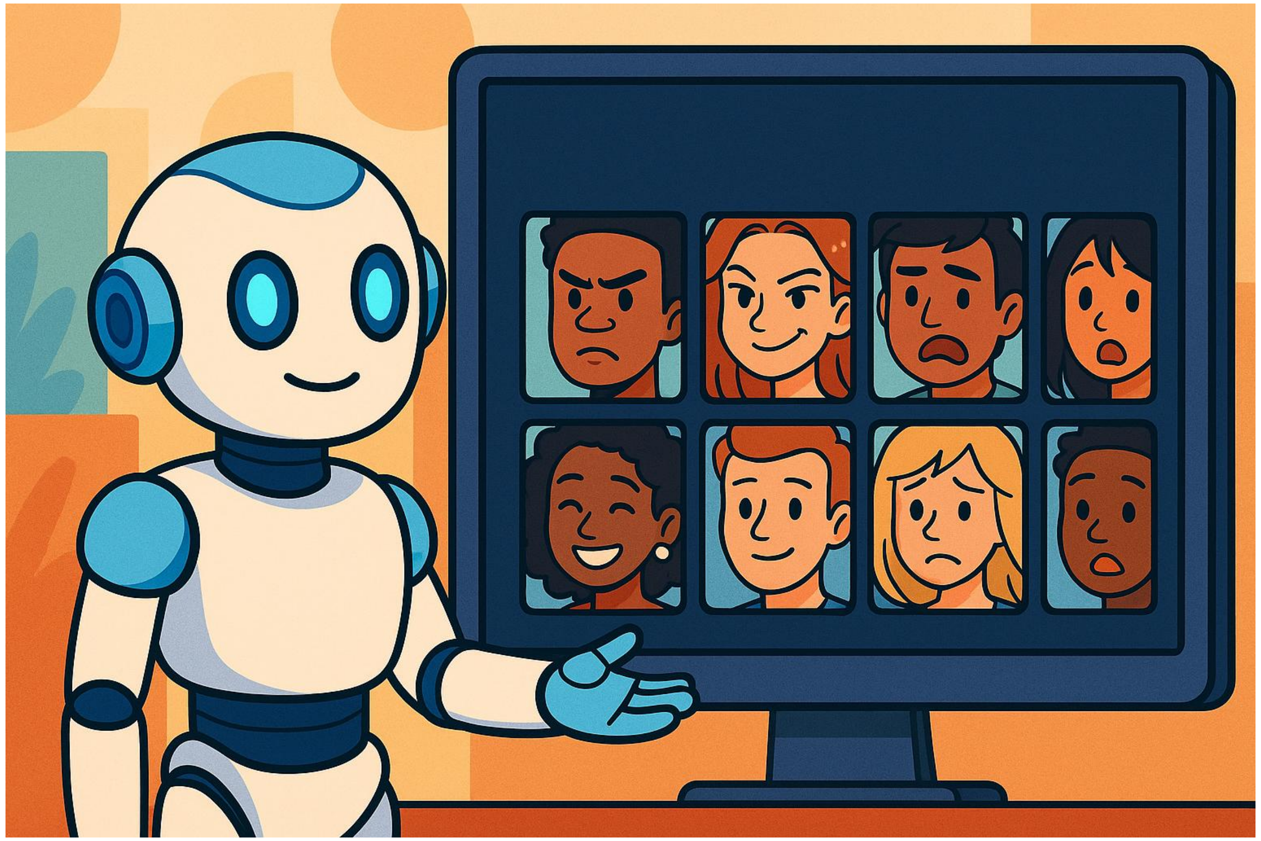
The YOLOv11 architecture supports both object detection and image classification. Variant YOLOv11s (The small variant) is faster, lightweight, and efficient, which is ideal for real time emotion recognition tasks. Using Kaggle Notebooks provides a convenient environment that allows direct access to the dataset from its source.



```
from ultralytics import YOLO
model = YOLO('yolo11s-cls.pt
```

SYSTEM ARCHITECTURE

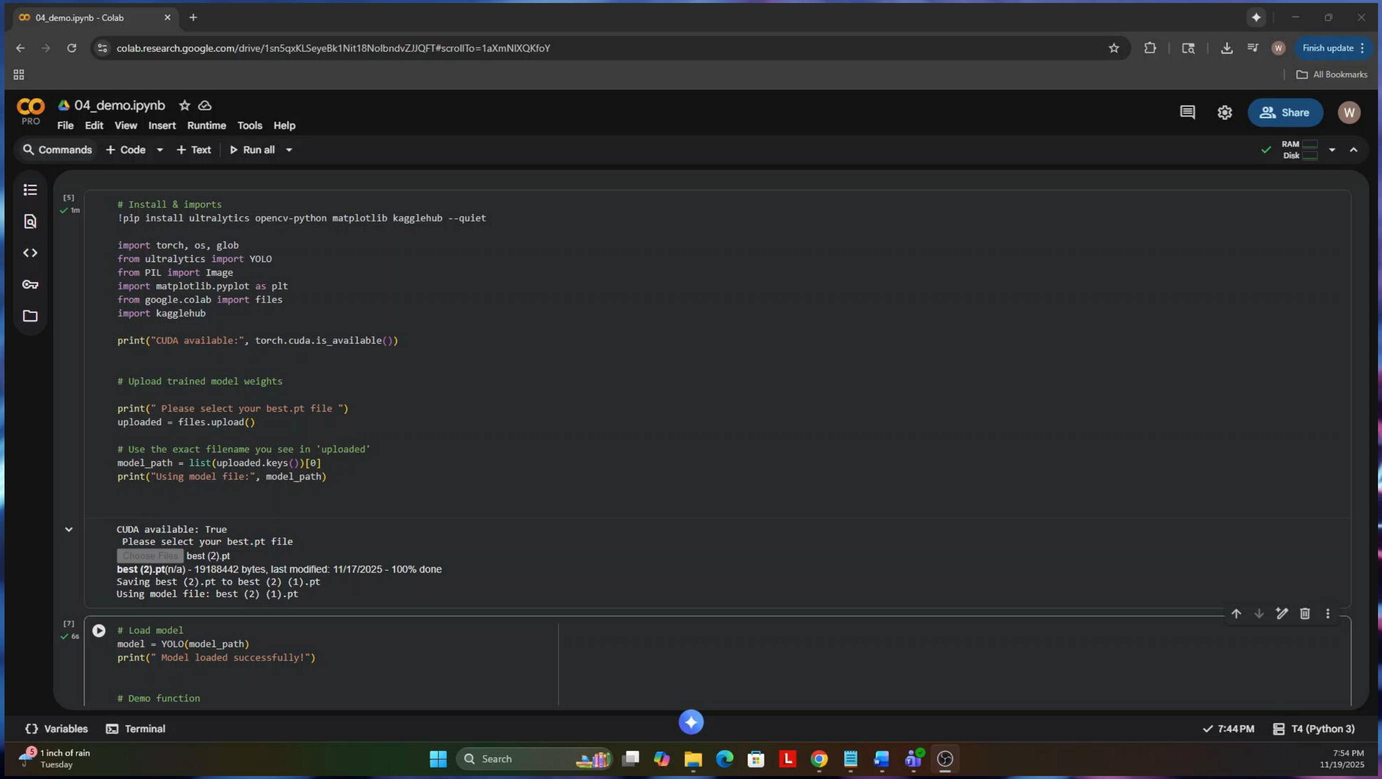
1. Input Stage
    1. User/webcam provides an image of a human face.
    2. Image is preprocessed
2. YOLOv11 Detection Model
    1. Model detects the location of the face in the image
    2. Bounding box coordinates extracted
    3. YOLOv11s (small) chosen for speed/accuracy
3. Emotion Classification
    1. Model assigns one of the eight AffectNet emotion labels:
       (Angry, Happy, Neutral, Sad, Surprised, Fear, Disgust, Contempt)
4. Output Generated
    1. Bounding Boxes drawn on images.
    2. Labels/confidence displayed.
    3. Output shown on screen for demo/analysis



Architecture Flow:

INPUT IMAGE → YOLO FACE DETECTOR → EMOTION CLASSIFIER → LABELED OUTPUT

DEMO



```
[5]
# Install & imports
!pip install ultralytics opencv-python matplotlib kagglehub --quiet

import torch, os, glob
from ultralytics import YOLO
from PIL import Image
import matplotlib.pyplot as plt
from google.colab import files
import kagglehub

print("CUDA available:", torch.cuda.is_available())


# Upload trained model weights

print(" Please select your best.pt file ")
uploaded = files.upload()

# Use the exact filename you see in 'uploaded'
model_path = list(uploaded.keys())[0]
print("Using model file:", model_path)
```

```
CUDA available: True
 Please select your best.pt file
Choose Files best (2).pt
best (2).pt(n/a) - 19188442 bytes, last modified: 11/17/2025 - 100% done
Saving best (2).pt to best (2) (1).pt
Using model file: best (2) (1).pt
```

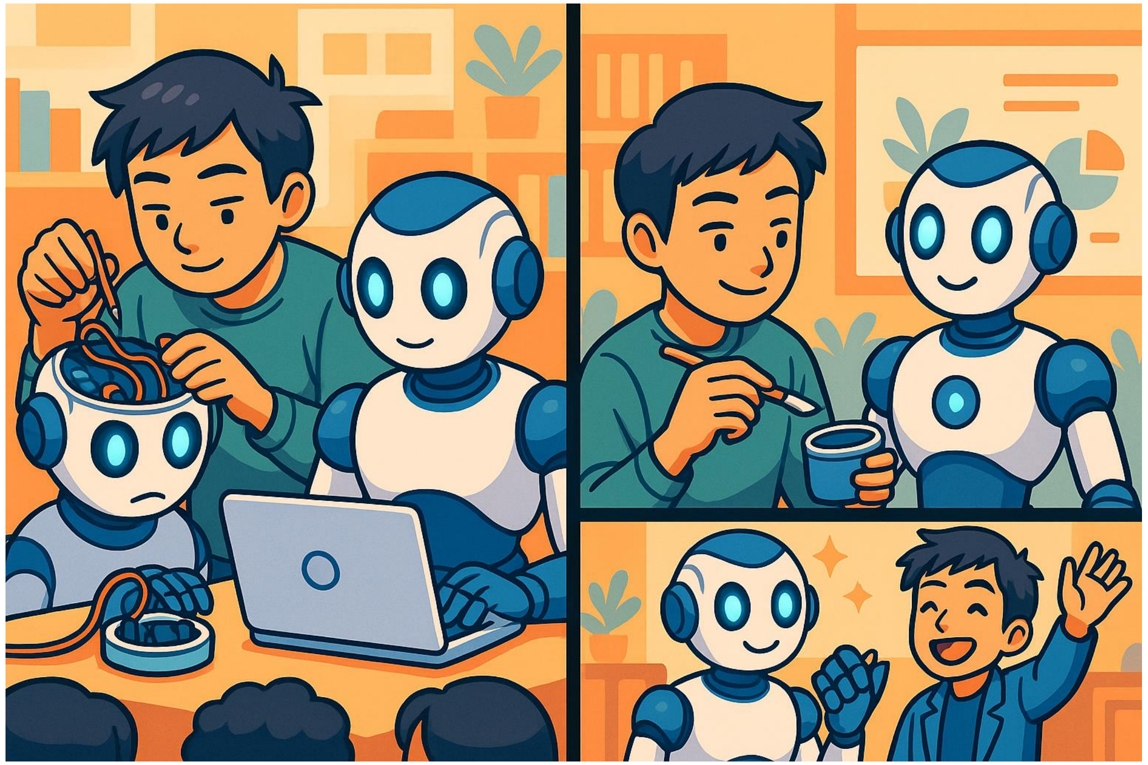
```
[7]
# Load model
model = YOLO(model_path)
print(" Model loaded successfully!")


# Demo function
```

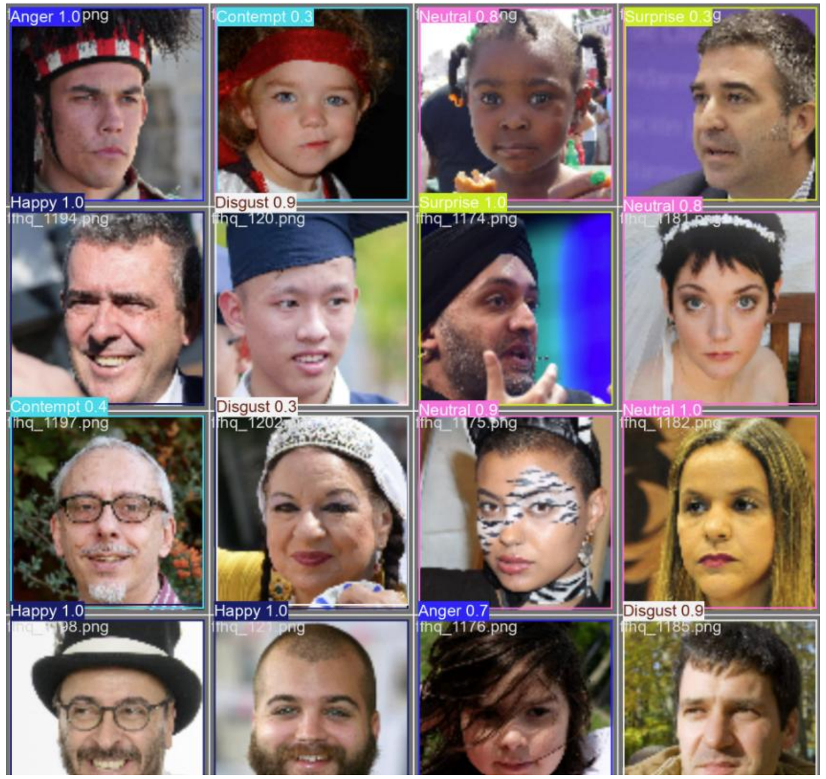
04_demo.ipynb - Colab
colab.research.google.com/drive/1sn5qxKLSeyeBk1Nit18NolbndvZJJQFT#scrollTo=1aXmNlXQKfoY
04_demo.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Share
RAM Disk
Variables Terminal
7:44 PM T4 (Python 3)
1 inch of rain Tuesday
Search
7:54 PM 11/19/2025

RESULTS (METRICS)

Metric	Score
Precision (P)	0.757
Recall (R)	0.773
mAP@50	0.849
mAP@50-95	0.849

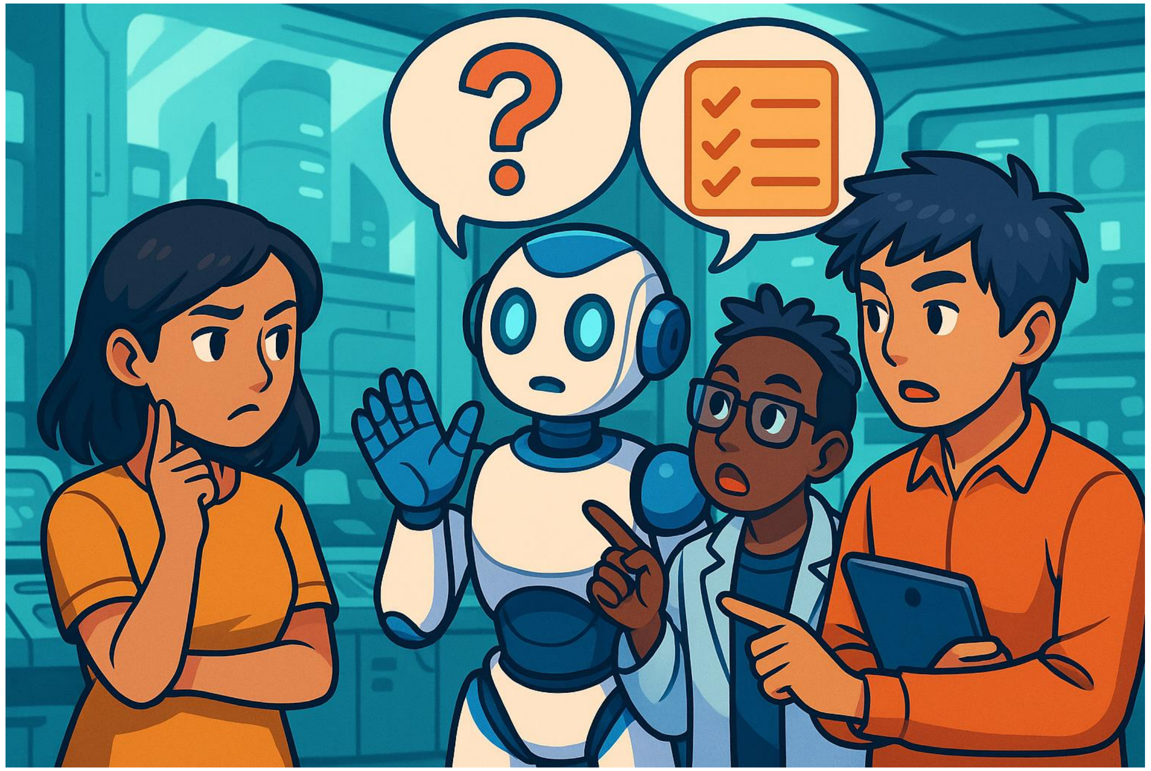
RESULTS (EXAMPLES)



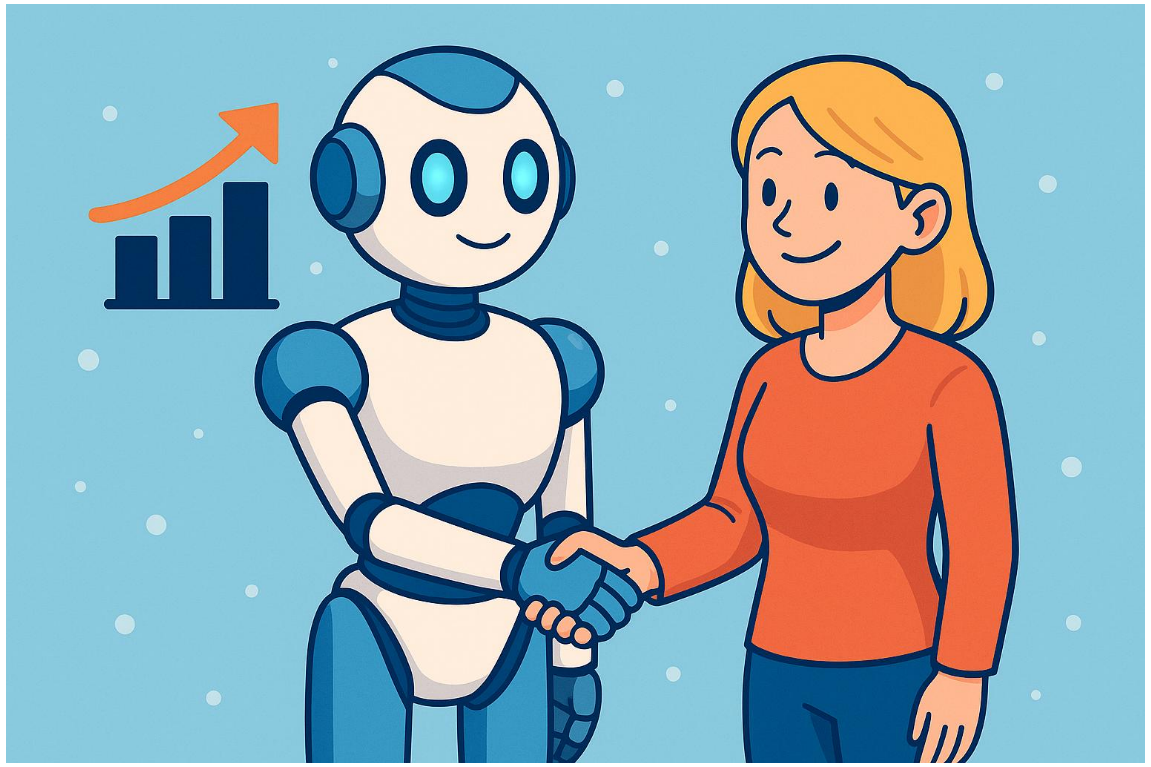


LEARNINGS

I have learned how to move from pretrained to training to evaluation to demo for building an AI model. Gained a better understanding of how model size, GPU type, and dataset size affect training time. Learned the difference between image detection vs. image classification. Lastly, I acquired skills in managing large files outside GitHub and running models in Colab with GPU acceleration (T4- A100).



FUTURE WORKS



- Integrate webcam input so the system identifies live.
- Extend the system to detect multiple faces in the same scene.
- Connect the model to a small robot or a virtual assistant.
- Research the possibility of using emotional patterns to flag high-risk situations.
- Add body pose recognition to read body language.
- Improve realism for use in different professional fields.
- Train with larger data (specialized, clinical)