EMOTION DETECTION: SELFIE CAPTURE

Mika Campbell, Kristen Mosley, Sydney Payne

SMAT 470

Dr. Erdi Kara

REAL TIME EXAMPLE

THE PROCESS

01

GAUGING UNDERSTANDING

Detecting emotions in a classroom environment : confused, bored, focused, distracted, understanding, neutral

02

DATASET ISSUES

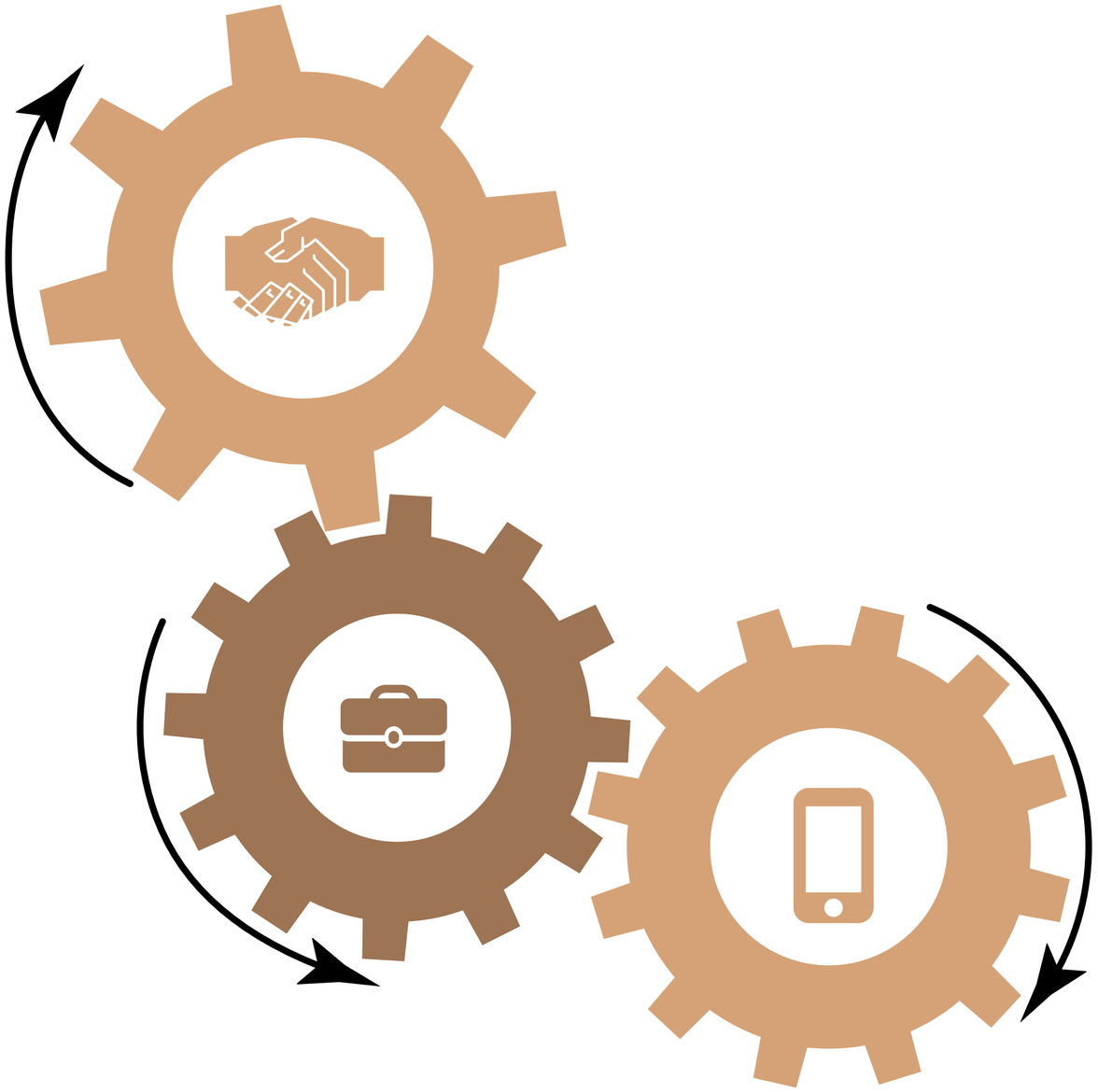
Difficult to determine the classes based only off of facial expressions

03

SELFIE CAPTURE

Focus shifted to a smaller - scale version of the project - more doable in the time frame and has more distinct classes

OUR METHOD



DATASET

Used Roboflow to annotate images

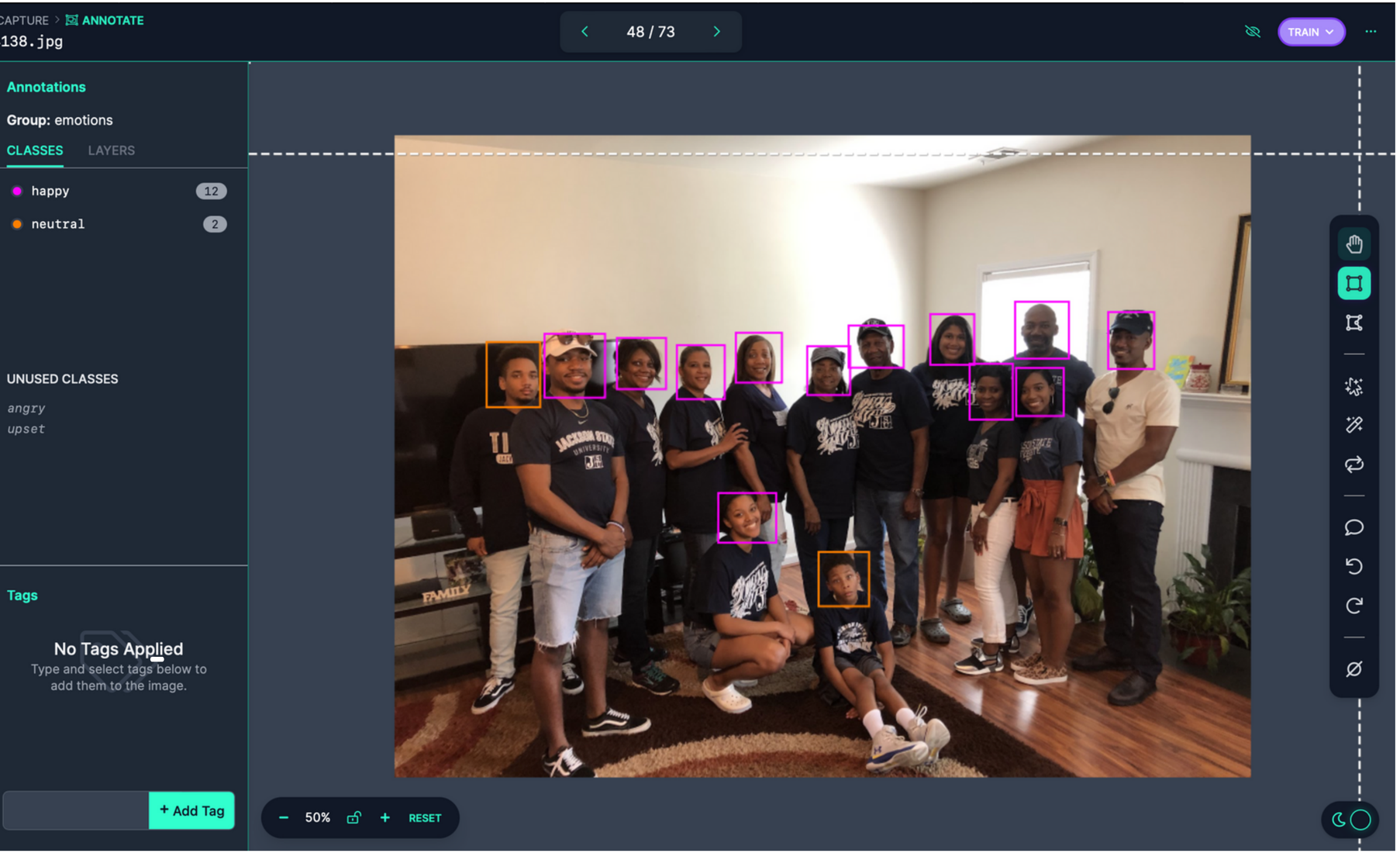
TRAIN MODEL

Used Colab to run selfie training and obtain the best weights and biases

MODEL IMPLEMENTATION

Used Jupyter notebook to run the model and obtain outputs

ANNOTATING



BOUNDING BOX

-rectangle drawn around an object in an image or video to indicate the object's location and size--commonly used to locate and isolate objects of interest within an image or video

THE CODE – GOOGLE COLAB

GOOGLE COLAB – FINDING THE BEST MODEL

```
[1]  #clone YOLOv5 and
     !git clone https://github.com/ultralytics/yolov5  # clone repo
     %cd yolov5
     %pip install -qr requirements.txt # install dependencies
     %pip install -q roboflow

     import torch
     import os
     from IPython.display import Image, clear_output  # to display images

     print(f"Setup complete. Using torch {torch.__version__} ({torch.cuda.get_device_properties(0).name if torch.cu
```

```
[2]  from roboflow import Roboflow
     rf = Roboflow(api_key="pmVVllkKlvhJaFdCMBoi")
     project = rf.workspace("emotiondetection-bkn0e").project("selfie-capture")
     dataset = project.version(2).download("yolov5")
```

- Contains code to upload necessary packages
- Uploaded annotated pictures from Roboflow to use as our dataset

THE CODE – GOOGLE COLAB

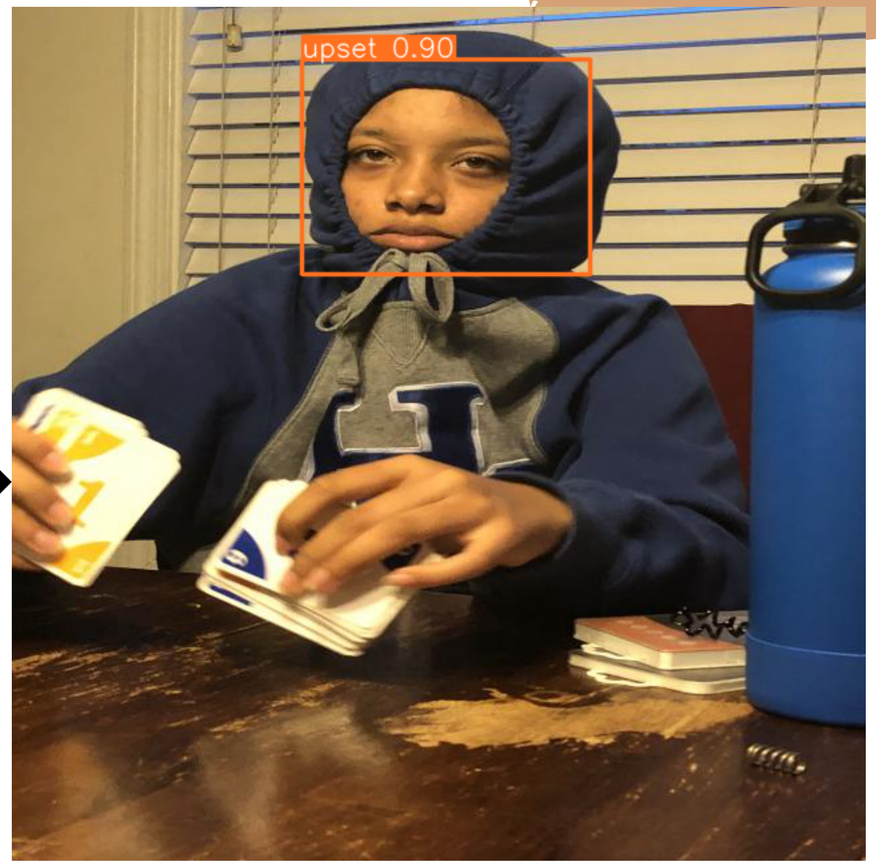
GOOGLE COLAB – FINDING THE BEST MODEL

```
[ ]  !python detect.py --weights runs/train/exp/weights/best.pt --img 640 --conf 0.1 --source {
```

- running an inference with trained weights

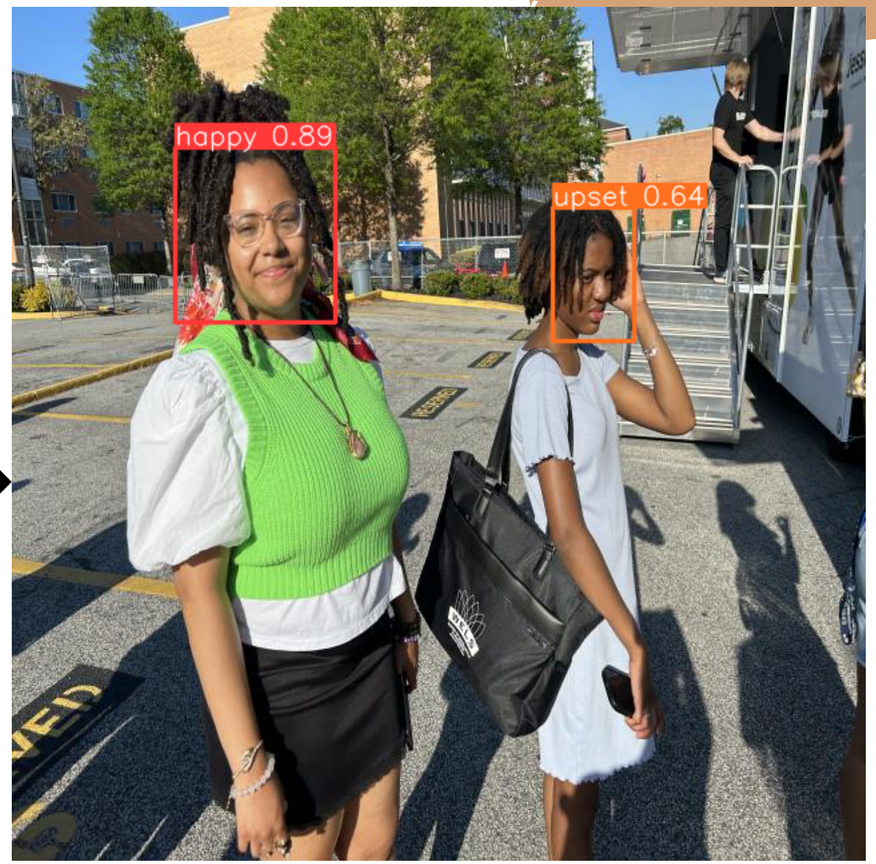
INPUT

OUTPUT



INPUT

OUTPUT



THE CODE – JUPYTER NOTEBOOK

IMPLEMENTING THE MODEL

```
model = torch.hub.load('ultralytics/yolov5', 'custom', path='best_face_model.pt', force_reload = True)
```

Process Video or Camera Feed In Real Time

In [3]:
```
#access already saved video
cap = cv2.VideoCapture('IMG_0956.mov')

#access camera
#cap = cv2.VideoCapture(0)
fps = int(cap.get(cv2.CAP_PROP_FPS))
print("Press 'c' to save the frame when you're happy with your pose.")
print("Press 'q' to quit.")
print("After saving a frame, press 's' to stop saving more frames, or press any other key to continue.")
#i=0
while True:
    #i+=1
    # grab a single frame from the camera
    ret, frame = cap.read()

    results = model(frame)
    cv2.imshow('YOLO', np.squeeze(results.render()))

    # Capture the frame when you press 'c'
    key = cv2.waitKey(1) & 0xFF
    if key == ord('c'):
        cv2.imwrite("selfie.jpg", frame)
        frame_saved = True
        print("Frame saved.")

        key = cv2.waitKey(0) & 0xFF
        if key == ord('s'):
            break

    # check if the user has pressed the 'q' key to quit
    if key == ord('q'):
        break

# release the camera and close all windows
cap.release()
cv2.destroyAllWindows()
```

```
Press 'c' to save the frame when you're happy with your pose.
Press 'q' to quit.
After saving a frame, press 's' to stop saving more frames, or press any other key to continue.
```

- Runs the model using our best model
- Accesses camera/ video gallery as the new input data for our model

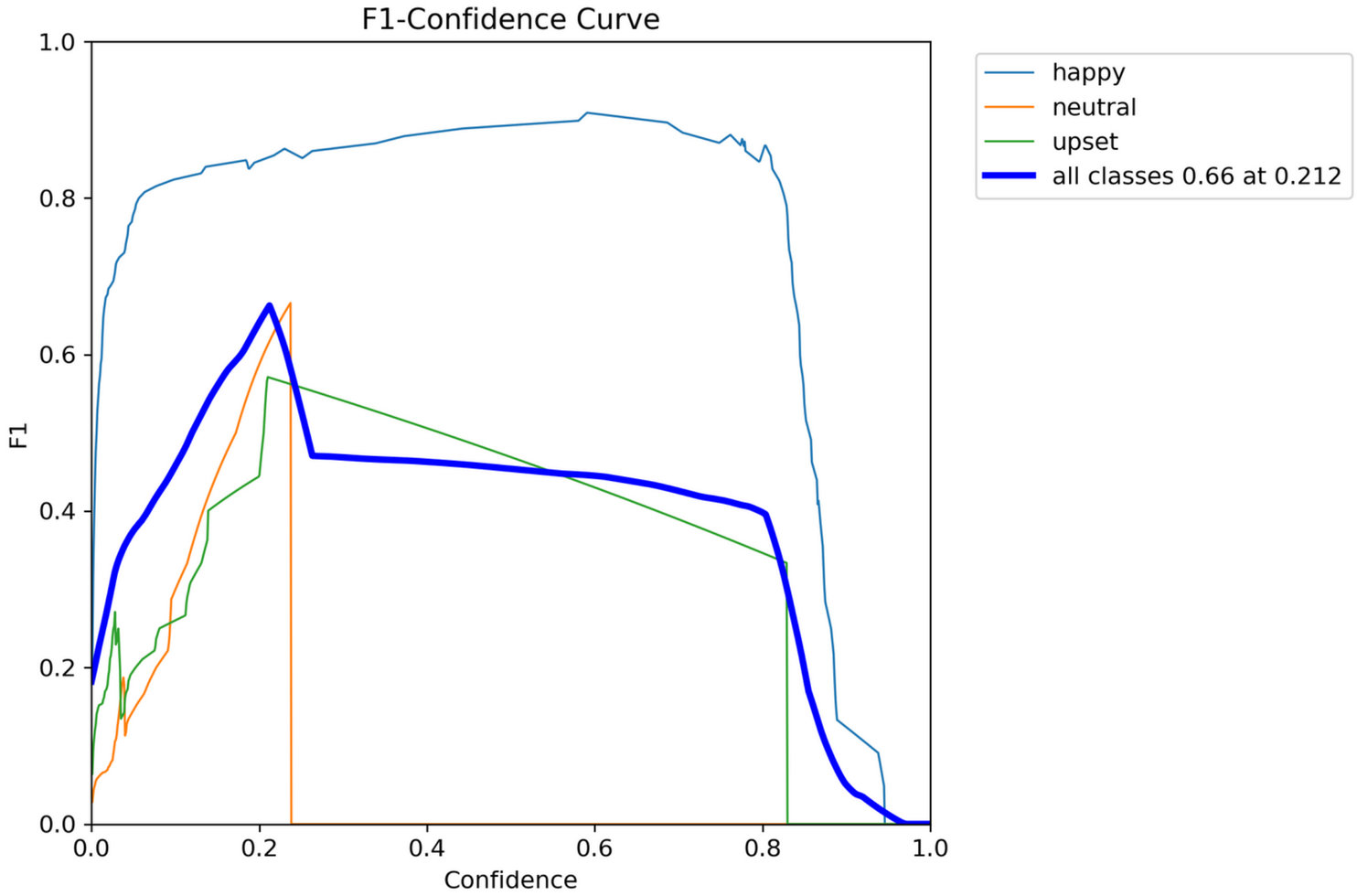
MODEL SUMMARY

```
Model summary: 157 layers, 7018216 parameters, 0 gradients, 15.8 GFLOPs
                 Class     Images  Instances          P          R      mAP50
                   all         21         49      0.609      0.626      0.666
                 happy         21         42      0.733      0.979      0.951
               neutral         21          2      0.615        0.5      0.554
                 upset         21          5      0.479        0.4      0.492
```

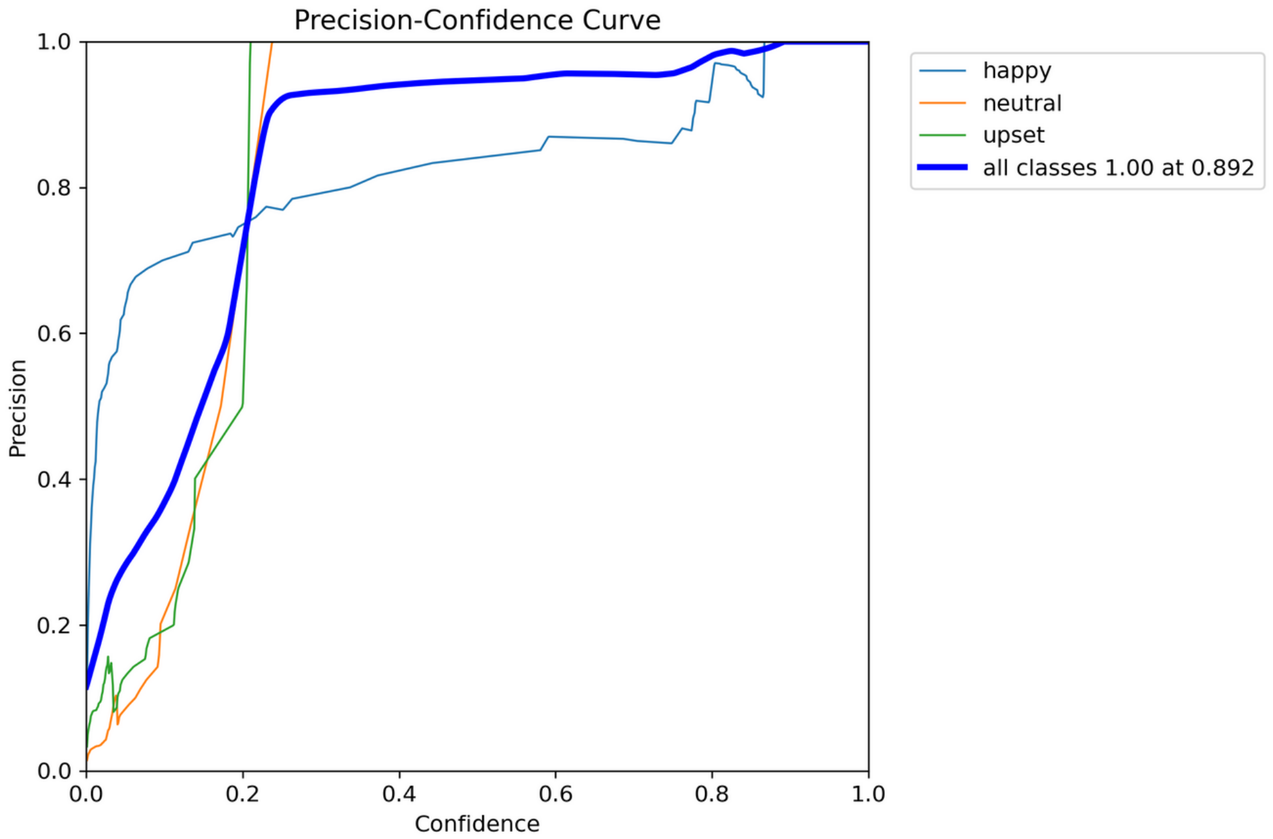
MEAN AVERAGE PRECISION

Calculates the average precision for each epoch and averages the values to provide an overall score for the ranking algorithm's performance

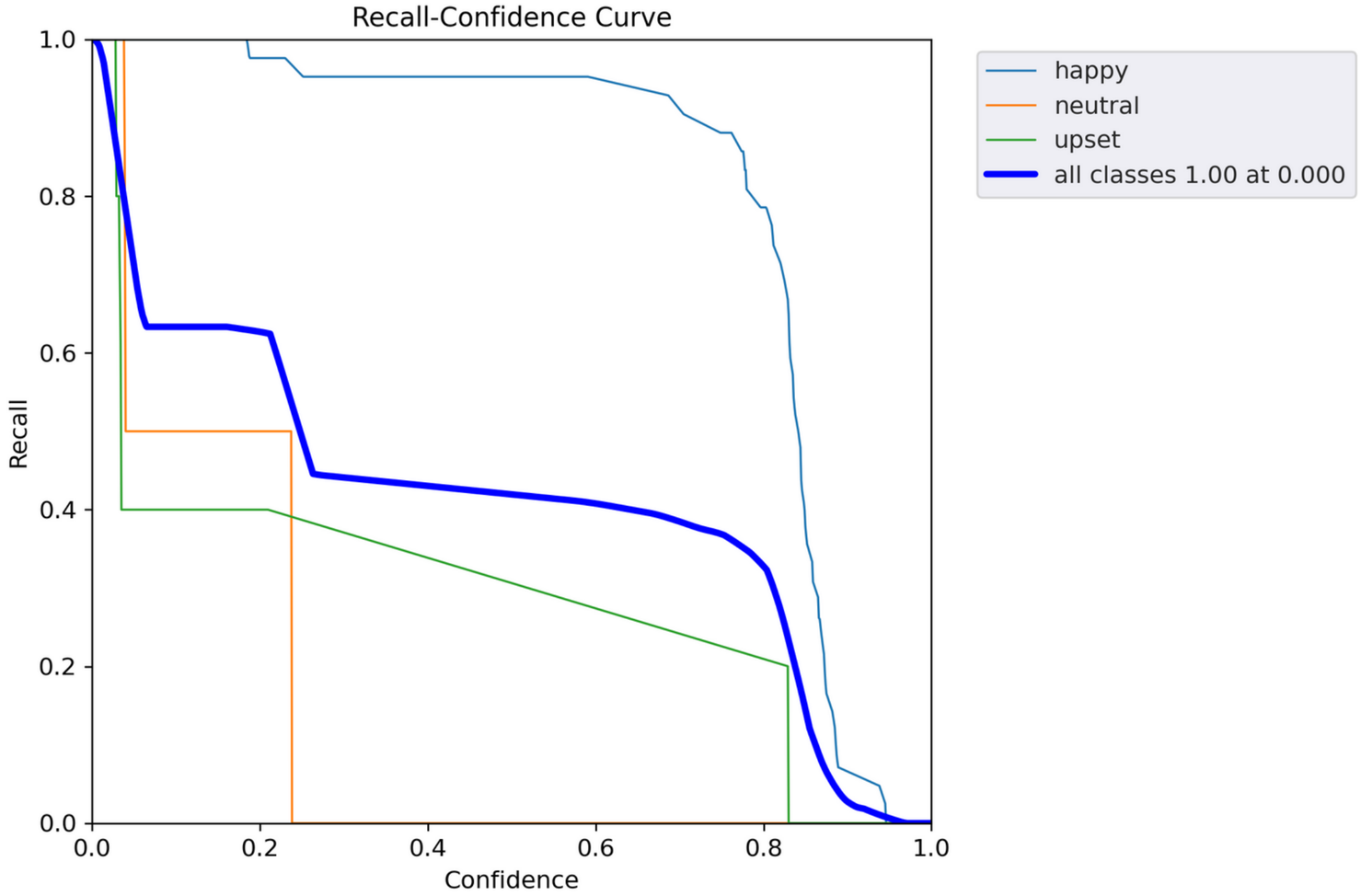
F-1 SCORES


F1-Confidence Curve
1.0
0.8
0.6
0.4
0.2
0.0
F1
0.0
0.2
0.4
0.6
0.8
1.0
Confidence
happy
neutral
upset
all classes 0.66 at 0.212

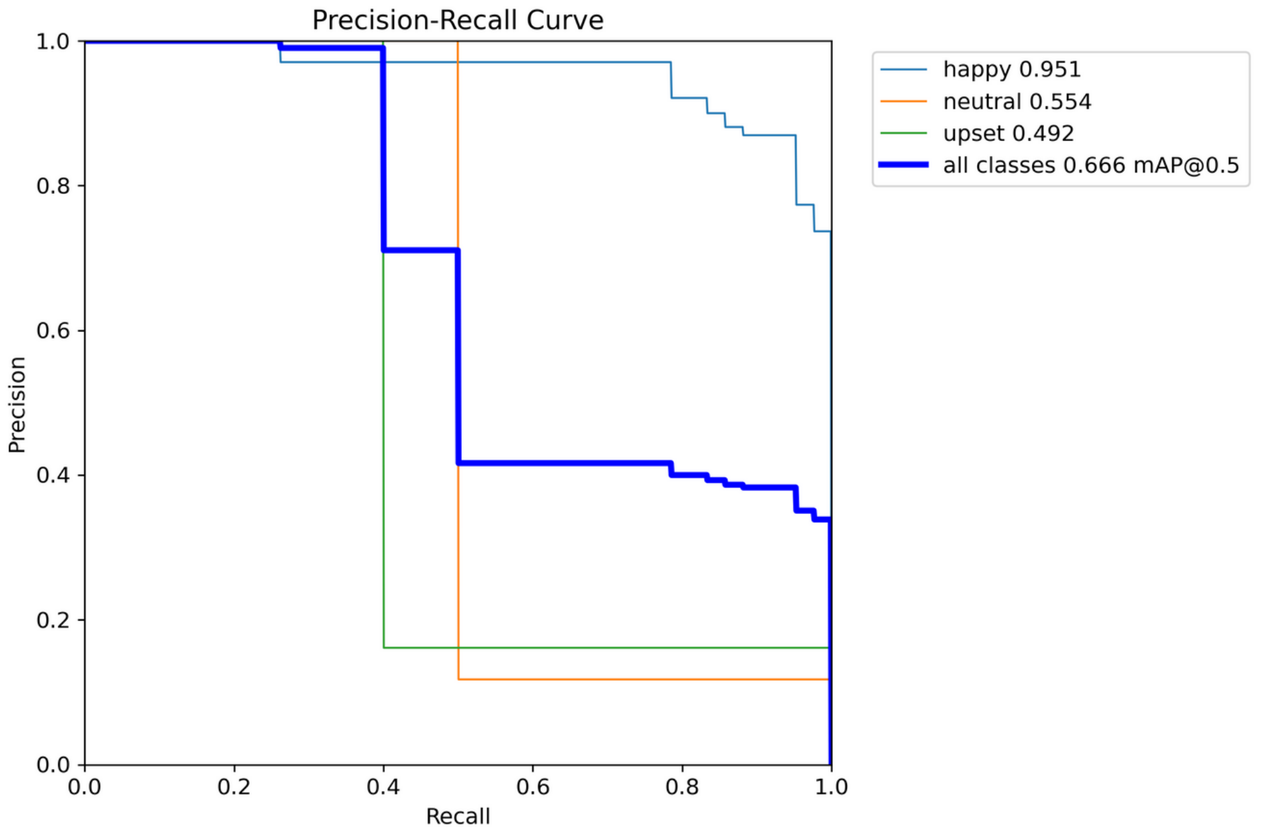
PRECISION


Precision-Confidence Curve
Precision
Confidence
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
happy
neutral
upset
all classes 1.00 at 0.892

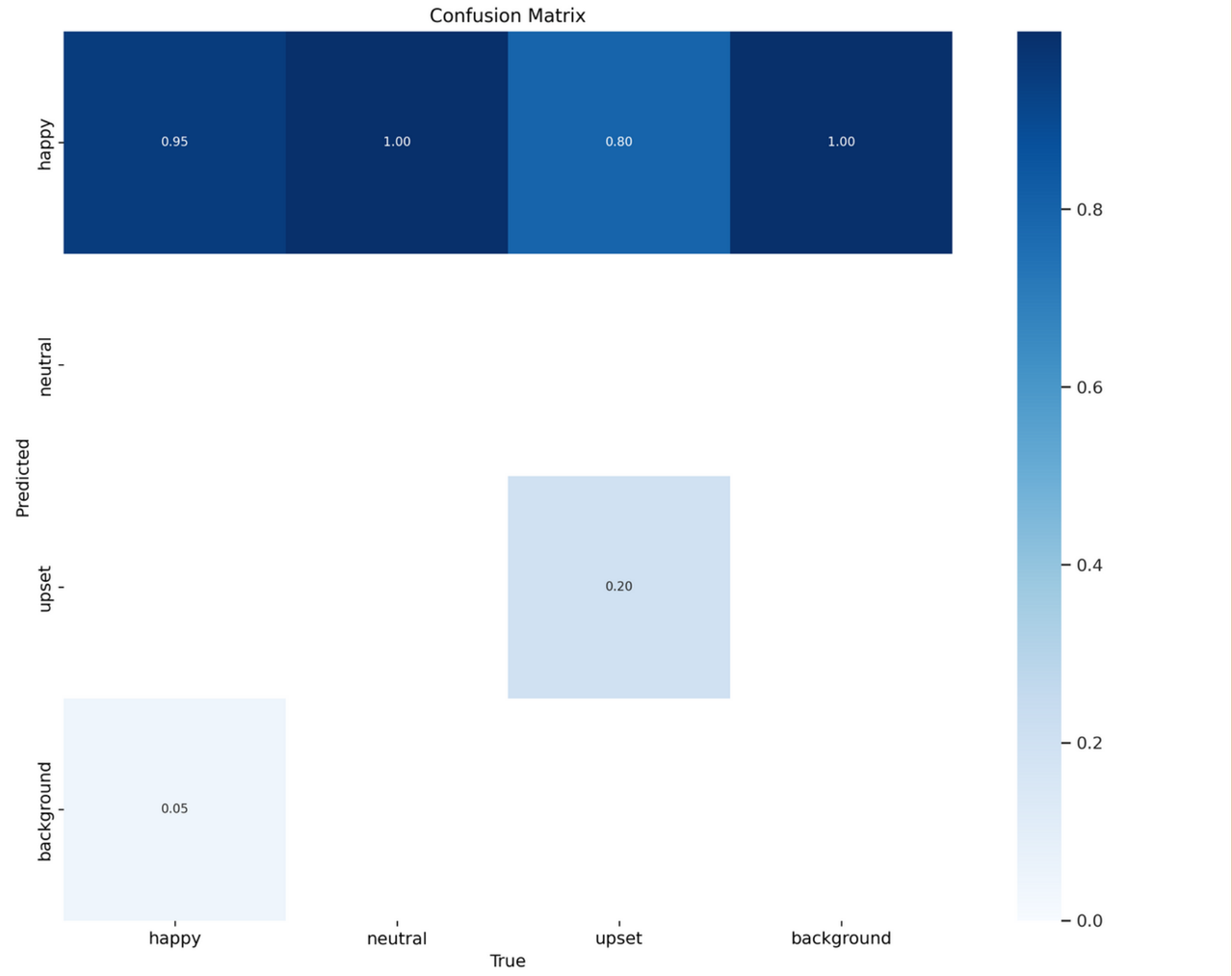
RECALL


Recall-Confidence Curve
Recall
Confidence
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
happy
neutral
upset
all classes 1.00 at 0.000

PRECISION – RECALL CURVE



CONFUSION MATRIX



USE CASES

BODY CAMS

STUDENT UNDERSTANDING IN CLASS

FUTURE EXPLORATION

CREATE A BETTER DATA SET

CAPTURE WHILE RECORDING

CREATING A DEPLOYABLE APP

THANK YOU