

## Topic 2 - Face Detection

Group 57

Jun 27, 2024

# Group Members

- NGUYEN DUY THAI - 175906
- PHAM HOANG DUY - 220607
- NGUYEN HOANG VINH QUANG - 219130
- PHAN VAN TAN - 219200
- TRUONG MINH NGHIA - 164626

- Introduction to Face Detection
- Proposed Solution
- Architecture Model Evaluation
- Architecture
- Data Processing
- Dataset
- Environment Setup
- Evaluation Metrics
- Training Custom YOLOv10-L Detector
- Benchmark
- Experimental Results
- Comparison with Other Models
- Test Environment
- Test Results: Inference on Image
- Test Results: Inference on Video

# Introduction to Face Detection

- **Definition:** Technology to identify and locate human faces in images and videos.
- **Importance:**
  - Foundation for facial recognition, emotion detection, and security systems.
  - Crucial in various applications like surveillance, user authentication, and personalized marketing.
- **How it Works:**
  - Uses algorithms and machine learning techniques.
  - Detects facial features and distinguishes them from other objects.
- **Advancements:**
  - Deep learning has enhanced accuracy and efficiency.
  - Modern systems are more robust and reliable.

# Proposed Solution

- **Model Selection:** Use YOLOv10-L, a state-of-the-art object detection model known for its speed and accuracy.
- **Data Collection:** Gather a dataset of diverse images containing faces, ensuring a balanced representation of different facial features.
- **Data Annotation:** Utilize the pre-labeled dataset downloaded from Kaggle, which includes about 32k images with clean labels in xywh (top-left x-coord, top-left y-coord, face-width, face-height) format for human face detection tasks.
- **Environment Setup:** Clone the YOLOv10-L repository and install dependencies in Google Colab for free GPU access.
- **Model Configuration:** Define the model architecture and configuration using a custom YAML file tailored for face detection.

# Proposed Solution

- **Training:** Train the YOLOv10-L model on the annotated dataset, optimizing for accuracy and performance.
- **Evaluation:** Assess the model's performance using metrics such as mAP (mean Average Precision) and adjust parameters as needed.
- **Inference:** Test the trained model on new images to validate its face detection capabilities.
- **Deployment:** Export the trained model weights for use in real-world applications.

# Architecture Model Evaluation

Model	Pros	Cons
<b>VGG16</b>	<ul style="list-style-type: none"><li>- Simplicity</li><li>- Strong Feature Extraction</li></ul>	<ul style="list-style-type: none"><li>- Computationally Intensive</li><li>- Not Specialized for Detection</li></ul>
<b>ResNet50</b>	<ul style="list-style-type: none"><li>- Residual Connections</li><li>- High Accuracy</li><li>- Scalability</li></ul>	<ul style="list-style-type: none"><li>- Complexity</li><li>- Resource Intensive</li></ul>
<b>YOLO</b>	<ul style="list-style-type: none"><li>- Real-Time Performance</li><li>- High Accuracy</li><li>- Unified Architecture</li></ul>	<ul style="list-style-type: none"><li>- Complexity</li><li>- Resource Intensive</li></ul>

- **YOLOv10 Detailed Structure:**

- **Backbone:**

- The backbone extracts features from images.
    - YOLOv10 uses an improved version of CSPNet (Cross Stage Partial Network) to improve the flow of information and use less computing power.

- **Neck:**

- The neck combines features from different backbone levels and passes them to the head.
    - It effectively mixes features from multiple scales(layers) using PAN (Path Aggregation Network) layers.

- **One-to-Many Head:**

- During training, this head makes several predictions for each object. This gives the model lots of information(features), helping it learn better.

- **One-to-One Head:**

- During inference, this head makes a single best prediction for each object. This removes the need for NMS (Non-Maximum Suppression), making the process faster.



- YOLOv10-L model configuration file (yolov10l.yaml):

*# Parameters*

*nc: 80 # number of classes*

*scales: # model compound scaling constants, i.e. 'model=yolov10l'*  
*# [depth, width, max\_channels]*

*l: [1.00, 1.00, 512] # YOLOv8l summary: 365 layers, 4369M*

*# YOLOv8.0n backbone*

**backbone:**

*# [from, repeats, module, args]*

*- [-1, 1, Conv, [64, 3, 2]] # 0-P1/2*

*- [-1, 1, Conv, [128, 3, 2]] # 1-P2/4*

*- [-1, 3, C2f, [128, True]]*

*- [-1, 1, Conv, [256, 3, 2]] # 3-P3/8*

- YOLOv10-L model configuration file (yolov10l.yaml):

- [-1, 6, C2f, [256, True]]
- [-1, 1, SCDwn, [512, 3, 2]] # 5-P4/16
- [-1, 6, C2f, [512, True]]
- [-1, 1, SCDwn, [1024, 3, 2]] # 7-P5/32
- [-1, 3, C2fCIB, [1024, True]]
- [-1, 1, SPPF, [1024, 5]] # 9
- [-1, 1, PSA, [1024]] # 10

*# YOLOv8.0n head*

head:

- [-1, 1, nn.Upsample, [None, 2, "nearest"]]
- [[-1, 6], 1, Concat, [1]] # cat backbone P4
- [-1, 3, C2fCIB, [512, True]] # 13
- [-1, 1, nn.Upsample, [None, 2, "nearest"]]

- **YOLOv10-L model configuration file (yolov10l.yaml):**

- `[[[-1, 4], 1, Concat, [1]] # cat backbone P3`
- `[-1, 3, C2f, [256]] # 16 (P3/8-small)`
- `[-1, 1, Conv, [256, 3, 2]]`
- `[[[-1, 13], 1, Concat, [1]] # cat head P4`
- `[-1, 3, C2fCIB, [512, True]] # 19 (P4/16-medium)`
- `[-1, 1, SCDwn, [512, 3, 2]]`
- `[[[-1, 10], 1, Concat, [1]] # cat head P5`
- `[-1, 3, C2fCIB, [1024, True]] # 22 (P5/32-large)`
- `[[[16, 19, 22], 1, v10Detect, [nc]] # Detect(P3, P4, P5)`

- **Preprocessing:**

- **Normalization:** Adjust image pixel values to a common scale to improve model performance.
- **Augmentation:** Apply techniques such as rotation, flipping, and scaling to increase dataset diversity.
- **Annotation:** Utilize the pre-labeled dataset downloaded from Kaggle, which includes about 32k images with clean labels.
- **Dataset Preparation:** Ensure balanced representation of various facial features and expressions.

- **Source:**
  - We used the Face Detection Dataset from Kaggle.
  - This dataset is specifically curated for training and testing face detection models.
- **Dataset Composition:**
  - **Training Set:** 26,300 images with annotated face locations.
  - **Validation Set:** 6,500 images with similar annotations.
- **Annotations:**
  - Each image comes with corresponding labels indicating face positions using bounding boxes.

- **Preparation:**

- Downloaded and extracted the dataset using a simple helper script.
- Ensured the removal of duplicate images and corresponding labels.

- **Directory Structure:**

- Organized as follows:

```
train
  images
  labels
validation
  images
  labels
test
  images
face-detect-datase.yaml
```

- **YAML Configuration:**

- Defined paths for training and validation data in a `face-detect-dataset.yaml`.
- Included class names and counts for model reference.
- **Training Data Path:**  
`/content/drive/MyDrive/face-detection-project/merged/images/train`
- **Validation Data Path:**  
`/content/drive/MyDrive/face-detection-project/merged/images/validation`
- **Class Names:** `['face']`
- **Number of Classes (nc):** 1

# Environment Setup

- **GPU Status Check:**

- Firstly we ensure the availability and readiness of the GPU for processing and running the YOLOv10 model.

- Command:

```
!nvidia-smi
```

```
[1] !nvidia-smi
```

```
↻ Sun Jun 30 15:33:10 2024
```

NVIDIA-SMI 535.104.05			Driver Version: 535.104.05		CUDA Version: 12.2	
GPU	Name	Perf	Persistence-M	Bus-Id	Disp.A	Volatile
Fan	Temp	Perf	Pwr:Usage/Cap		Memory-Usage	GPU-Util
						Uncorr. ECC
						Compute M.
						MIG M.
0	NVIDIA L4		Off	00000000:00:03.0	Off	0
N/A	44C	P8	12W / 72W	1MiB / 23034MiB		0%
						Default
						N/A

Processes:						
GPU	GI	CI	PID	Type	Process name	GPU Memory
ID	ID	ID				Usage
No running processes found						

Figure 1: GPU Status



# Environment Setup

- **Mounting Google Drive:**

- Next, we need to navigate to folder where the dataset is stored
- Command:

```
from google.colab import drive
drive.mount('/content/drive')
```

- **Display Current Directory:**

- Store current working directory's path in the HOME variable for reference.
- Command:

```
import os
HOME = os.getcwd()
print(HOME)
```

- **Install YOLOv10:**

- To set up the environment for running the YOLOv10 model, we need to install and clone the YOLOv10 repository. These installations are crucial for ensuring that all necessary tools and libraries are available for the model to function correctly.

- **Install YOLOv10:**

- Command:

```
!pip install git+https://github.com/THU-MIG/yolov10.git
```

- **Download YOLOv10-L Pre-trained Weights:**

- This step involves creating a directory for storing the model weights and downloading the pre-trained weights for YOLOv10-L model. These weights are essential for initializing the model and can be used for both inference and fine-tuning.

- Command:

```
!mkdir -p {HOME}/weights
```

```
!wget -P {HOME}/weights -q https://github.com/jameslahm/yolov10/releases/download/v1.0/yolov10l.pt
```

- **Average Precision (AP):**
  - Measures precision and recall at various thresholds.
  - Calculates the weighted mean of precisions achieved at each threshold.
  - Provides a comprehensive view of model performance across different confidence levels.
- **Mean Average Precision (mAP):**
  - Combines AP scores over multiple IoU thresholds (e.g., 0.5 to 0.95).
  - Averages AP across all classes in the dataset.
  - Offers a comprehensive metric for overall model performance comparison.
- **AP@0.5:**
  - Measures precision and recall with a fixed Intersection over Union (IoU) threshold of 0.5.
  - Indicates how well the model distinguishes true positives from false positives.

- **P Curve (Precision Curve):**
  - Shows the precision at various confidence levels.
  - Helps in understanding how precision changes with different confidence thresholds.
- **PR Curve (Precision-Recall Curve):**
  - Plots precision against recall for different threshold values.
  - Provides insight into the trade-off between precision and recall.
- **R Curve (Recall Curve):**
  - Displays recall at various confidence levels.
  - Useful for understanding how recall varies with different confidence threshold
- **Confusion Matrix:**
  - This cell displays the confusion matrix generated during training.
  - The confusion matrix helps in visualizing the performance of the model by showing the accuracy of its predictions.

# Training Custom YOLOv10-L Detector

- **Model Summary:** YOLOv10-L with 628 layers, 25,766,870 parameters, and 127.2 GFLOPs.
- **Training:**
  - Total Epochs: 50
  - Batch Size: 8
  - Image Size: 640x640 pixels
  - Data Augmentation:
    - **Color Augmentation:** Adjustments in hue (hsv\_h=0.02), saturation (hsv\_s=0.8), and value (hsv\_v=0.5).
    - **Geometric Transformations:** Includes rotation (degrees=5), translation (translate=0.2), scaling (scale=0.6), and shearing (shear=2).
    - **Perspective Transform:** Small adjustments with perspective (perspective=0.001).
    - **Flipping:** Both vertical flipping (flipud=0.1) and horizontal flipping (fliplr=0.6).

- **Training:**

- Data Augmentation:

- **Mosaic:** Combines four images into one (mosaic=1.0).
    - **MixUp:** Merges two images into one (mixup=0.2).
    - **Copy-Paste:** Pastes objects from one image into another (copy\_paste=0.1).
    - **Auto Augment:** Uses RandAugment strategy for automatic augmentation.
    - **Erasing:** Randomly erases parts of images (erasing=0.5).
    - **Cropping:** Applies cropping with a fraction of 1.0 (crop\_fraction=1.0).

- **Training Results:**
  - Initial Epoch GPU Memory: 17.2G
  - Final Epoch GPU Memory: 14.7G
  - Total Training Time: 13.777 hours
- **Validation Results:**
  - Precision (P): 0.861
  - Recall (R): 0.669
  - mAP@0.5: 0.735
  - mAP@0.5:0.95: 0.42
- **Inference Speed:**
  - Preprocess: 0.2ms per image
  - Inference: 28.1ms per image
  - Postprocess: 0.1ms per image
- **Model Performance:** Efficient detection and robust accuracy, suitable for real-time face detection applications

- Command:

```
!yolo task=detect mode=train epochs=50 batch=12  
imgsz=640 plots=True model='/content/drive/MyDrive/  
face-detection-project/yolov10/weights/yolov10l.pt'  
data='/content/drive/MyDrive/face-detection-project/  
merged/face-detect-dataset.yaml' project='/content/  
drive/MyDrive/face-detection-project/runs/detect/  
train4/weights' name='train4' augment=True  
hsv_h=0.02 hsv_s=0.8 hsv_v=0.5 degrees=5 translate=0.2  
scale=0.6 shear=2 perspective=0.001 flipud=0.1  
fliplr=0.6 mosaic=1.0 mixup=0.2 copy_paste=0.1  
auto_augment=randaugument erasing=0.5 crop_fraction=1.0
```



# Benchmark

```
+ Code + Text
```

		Class	Images	Instances	Box(P	R	mAP50	mAP50-95):	100%	342	
		all	5460	46458	0.862	0.668	0.735	0.419			↑ ↓ ↻ ⌨ ⚙ 📄 🗑
Epoch	GPU_mem	box_om	cls_om	dfl_om	box_oo	cls_oo	dfl_oo	Instances	Size		
49/50	12.7G	1.176	0.5165	1.108	1.419	0.6591	1.071	9	640: 100%	3284/3284	[
	Class	Images	Instances	Box(P	R	mAP50	mAP50-95):	100%	342/342	[01:42<00:00,	3.35it/s]
	all	5460	46458	0.861	0.668	0.735	0.419				
Epoch	GPU_mem	box_om	cls_om	dfl_om	box_oo	cls_oo	dfl_oo	Instances	Size		
50/50	14.7G	1.175	0.5122	1.101	1.417	0.6588	1.065	1	640: 100%	3284/3284	[
	Class	Images	Instances	Box(P	R	mAP50	mAP50-95):	100%	342/342	[01:42<00:00,	3.35it/s]
	all	5460	46458	0.859	0.669	0.735	0.42				

50 epochs completed in 13.777 hours.  
Optimizer stripped from /content/drive/MyDrive/face-detection-project/runs/detect/train4/weights/train46/weights/last.pt, 52.2MB  
Optimizer stripped from /content/drive/MyDrive/face-detection-project/runs/detect/train4/weights/train46/weights/best.pt, 52.2MB

Validating /content/drive/MyDrive/face-detection-project/runs/detect/train4/weights/train46/weights/best.pt...

Ultralytics YOLOv8.1.34 Python-3.10.12 torch-2.3.0+cu121 CUDA:0 (NVIDIA L4, 22700MiB)

YOLOv10l summary (fused): 461 layers, 25717010 parameters, 0 gradients, 126.3 GFLOPs

Class	Images	Instances	Box(P	R	mAP50	mAP50-95):	0%	0/342	[00:00<?, ?it/s]	/usr/local/
return F.conv2d(input, weight, bias, self.stride,										
Class	Images	Instances	Box(P	R	mAP50	mAP50-95):	100%	342/342	[02:58<00:00,	1.91it/s]
all	5460	46458	0.486	0.64	0.393	0.233				

Speed: 0.2ms preprocess, 28.1ms inference, 0.0ms loss, 0.1ms postprocess per image  
Results saved to /content/drive/MyDrive/face-detection-project/runs/detect/train4/weights/train46  
💡 Learn more at <https://docs.ultralytics.com/modes/train>

Figure 2: Train model

# Benchmark

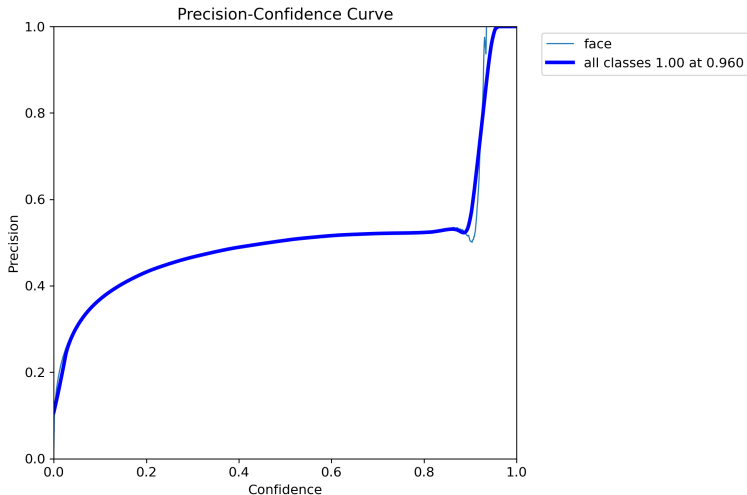


Figure 3: P curve

# Benchmark

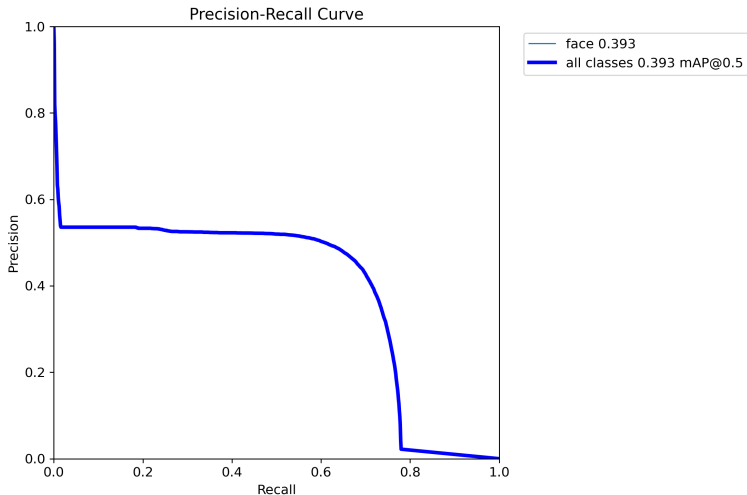


Figure 4: PR curve

# Benchmark

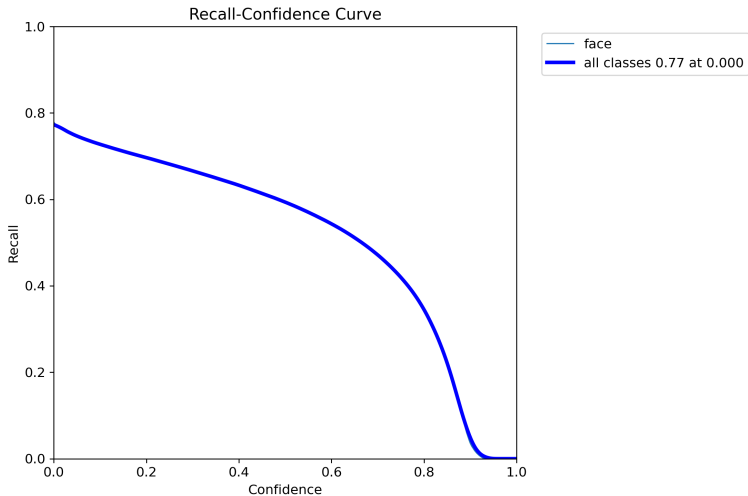


Figure 5: R curve

# Benchmark

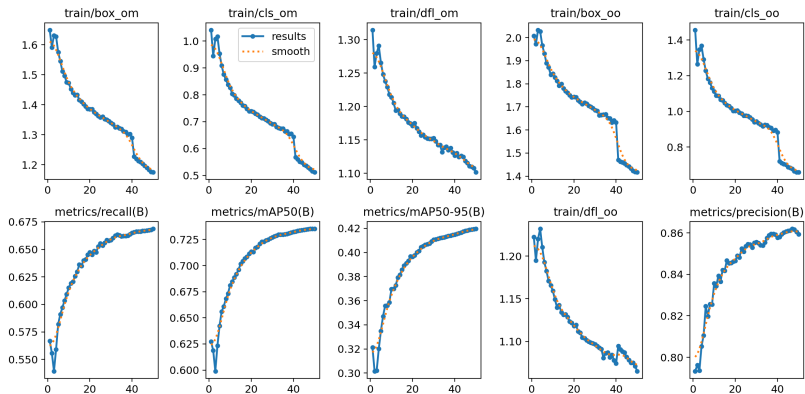


Figure 6: Train model result

# Benchmark

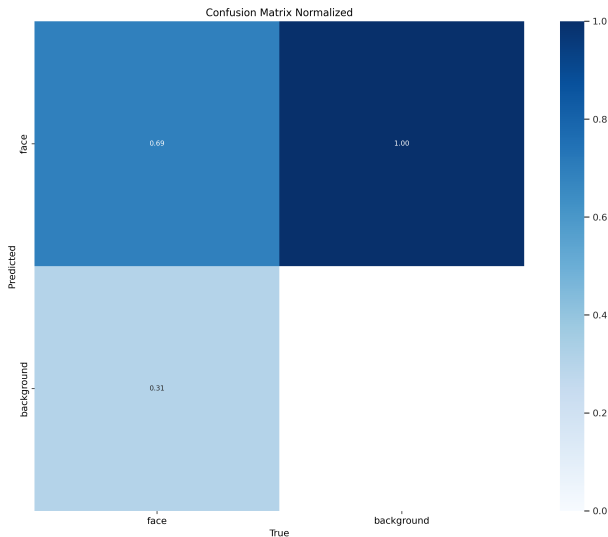


Figure 7: Confusion Matrix

- **Accuracy and Speed:**

- Our model showed competitive precision and recall rates compared to other leading models.
- The average precision (mAP@0.5) was 73.5%, while the mAP@0.5:0.95 reached 41.9%.
- The inference speed was efficient, making our model suitable for real-time applications.

- **Resource Utilization:**

- The model demonstrated efficient GPU memory usage, with a peak of 17.2G during training.
- The combination of precision, speed, and resource efficiency highlights the robustness of our model for face detection tasks.

- **Hardware:** NVIDIA L4 GPU
- **Software:** Ultralytics YOLOv8.1.34, Python 3.10.12, Torch 2.3.0+cu121
- **Detection Performance:**
  - Generated images and screenshots demonstrate the detection performance on test data.
  - Model: YOLOv10
  - Confidence threshold: 0.25
  - Results show the model identifying multiple faces with high accuracy.



# Test Results: Inference on Image

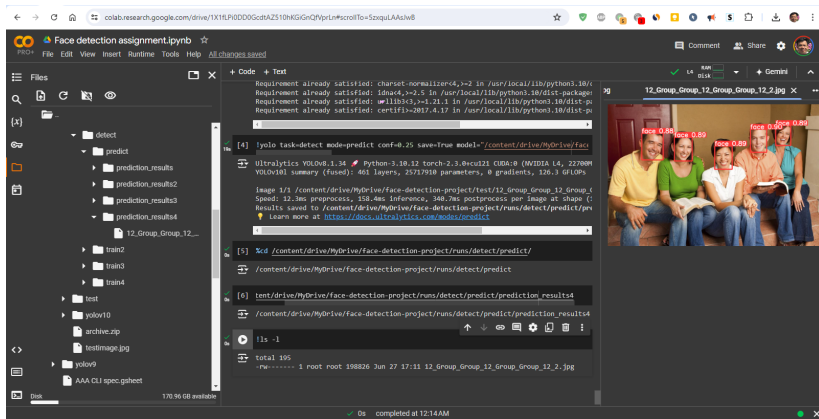
- **Inference Details:**

- Model: YOLOv10l
- Parameters: 46 layers, 25,717,910 parameters, 126.3 GFLOPs
- Inference speed: 158.4ms per image
- Command:  

```
!yolo task=detect mode=predict conf=0.25 save=True  
model="/content/drive/MyDrive/face-detection-project/  
runs/detect/train4/weights/train46/weights/best.pt"  
source="/content/drive/MyDrive/face-detection-project/  
test/12_Group_Group_12_Group_Group_12_2.jpg"  
project="/content/drive/MyDrive/face-detection-project/  
runs/detect/predict" name="prediction_results"
```
- Results saved to:  

```
/content/drive/MyDrive/face-detection-project/runs/  
detect/predict/prediction_results4
```
- Learn more at: [Ultralytics Documentation](#)

# Test Results: Inference on Image



The screenshot displays a Google Colab workspace titled "Face detection assignment.ipynb". The interface includes a file explorer on the left, a code editor in the center, and a terminal at the bottom. The file explorer shows a directory structure with folders like "detect", "predict", and "test", and files like "testimage.jpg". The code editor contains several cells, including a requirements list, a JupyterLab command to run a face detection model, and a terminal output showing the execution of the command. The terminal output includes the command `!yolo task=detect mode=predict conf=0.25 save=True model="/content/drive/MyDrive/face-detection-project/runs/detect/predict/weights/ultralytics_yolo10l.pt" image="/content/drive/MyDrive/face-detection-project/test/12_Group_Group_12_Group_12_2.jpg" results="/content/drive/MyDrive/face-detection-project/runs/detect/predict/prediction_results4` and the output `total 195`. A preview of the detected image is shown on the right, displaying five people with bounding boxes and confidence scores (e.g., 0.88, 0.89, 0.89, 0.89, 0.88) indicating successful face detection.

Figure 8: Google Colab workspace

# Test Results: Inference on Image

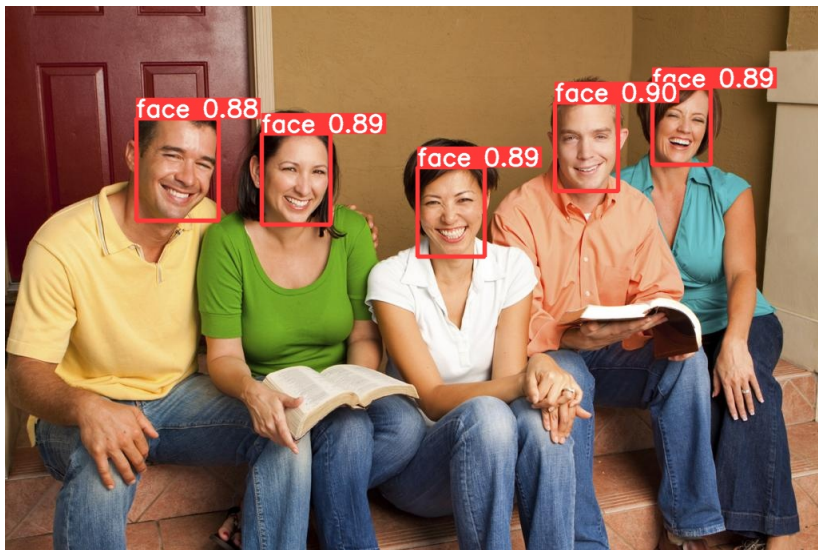


Figure 9: Inference on image

# Test Results: Inference on Image

- **Inference Details:**

- Model: YOLOv10l
- Parameters: 461 layers, 25,717,910 parameters, 126.3 GFLOPs
- Inference speed: 155.5ms per image
- Command:

```
!yolo task=detect mode=predict conf=0.25 save=True
model="/content/drive/MyDrive/face-detection-project/
runs/detect/train4/weights/train46/weights/best.pt"
source="/content/drive/MyDrive/face-detection-project/
test/runs/AOS-group.png"
project="/content/drive/MyDrive/face-detection-project/
runs/detect/predict" name="prediction_results"
```
- Results saved to:  
/content/drive/MyDrive/face-detection-project/runs/  
detect/predict/prediction\_results5
- Learn more at: [Ultralytics Documentation](#)

# Test Results: Inference on Image



Figure 10: Inference on image

# Test Results: Inference on Video

- Command:

```
!yolo task=detect mode=predict conf=0.25 save=True  
model="/content/drive/MyDrive/face-detection-project  
/runs/detect/train4/weights/train46/weights/best.pt"  
source="/content/drive/MyDrive/face-detection-project  
/test/WALK-NEW-YORK-City-USA-vlog.mp4" project="/content  
drive/MyDrive/face-detection-project/runs/detect/predict  
name="prediction_results"
```

- Log:

```
Streaming output truncated to the last 5000 lines.  
video 1/1 (frame 76799/81795) /content/drive/MyDrive/  
face-detection-project/test/WALK-NEW-YORK-City-USA-vlog  
384x640 1 face, 16.6ms
```

# Test Results: Inference on Video

- Log:

```
video 1/1 (frame 76800/81795) /content/drive/MyDrive/  
face-detection-project/test/WALK-NEW-YORK-City-USA-vlog  
384x640 2 faces, 16.8ms
```

```
video 1/1 (frame 77973/81795) /content/drive/MyDrive/  
face-detection-project/test/WALK-NEW-YORK-City-USA-vlog  
384x640 (no detections), 16.7ms
```

```
...
```

```
Speed: 2.3ms preprocess, 17.3ms inference, 1.1ms  
postprocess per image at shape (1, 3, 384, 640)
```

```
Results saved to /content/drive/MyDrive/face-detection-  
project/runs/detect/predict/prediction_results6
```

- Play the inference result video:

<https://www.youtube.com/watch?v=HwbmiKk6k3I>

# Test Results: Inference on Video



Figure 11: Inference on video



# Conclusion

- Face detection is a vital technology
- Wide range of applications
- Project aims to contribute to this field

# Questions?

- Open for any questions or discussions

- Dataset: Face Detection Dataset
- Information: Train set - 26,300 images, Test set - 6,500 images
- Evaluation Metric: AP, AP@0.5