IE 7615: Deep Learning for AI

Discriminative Project
CNN / Deep Learning Based Automated Class Attendance System

Milestone 2

Group VI

Gayatri Nair
Harish Padmanabhan
Harrish Ebi Francis Peter Joshua
Priyanka Raj Rajendran

nair.gaya@northeastern.edu
padmanabhan.h@northeastern.edu
peterjoshua.h@northeastern.edu
rajendran.priy@northeastern.edu

Git Repo Link: https://github.com/harish2412/Object_Detection

Submission Date: 6th October 2025

CNN/DEEP LEARNING BASED AUTOMATED CLASS ATTENDANCE SYSTEM

Overview:

This project develops a deep learning–based system for automated celebrity identification using Convolutional Neural Networks (CNNs). The primary objective is to address the challenge of accurate facial recognition in both single-person and multi-person contexts, with potential applications such as automated attendance tracking. To achieve this, the project leverages the CelebA dataset to construct robust classification and object detection models that are optimized for real-time identification, ensuring both high accuracy and computational efficiency.

The system adopts a two-phase methodology. The first phase focuses on single-celebrity classification through the implementation and comparison of multiple CNN architectures. The second phase extends to multi-celebrity detection and localization by incorporating advanced object detection frameworks. This staged approach not only strengthens the theoretical understanding of discriminative deep learning techniques but also demonstrates their practical applicability in real-world scenarios where scalability and reliability are essential.

PROJECT SCOPE:

- Single-celebrity image classification using CelebA dataset
- Implementation and comparison of multiple CNN architectures (ResNet-50, EfficientNet-B3, EfficientNet-B4, ConvNeXt-Base)
- Transfer learning implementation with pre-trained ImageNet models
- Model performance evaluation using standard computer vision metrics
- Development of model interface for demonstration purposes
- Comprehensive documentation and analysis of results

DATASET:

- Source: CelebA (aligned images + identity file).
- Files: img_align_celeba — images (.jpg)
- identity_CelebA.txt — (image_id, celebrity_id)
- list_eval_partition.csv
- Curation & Splits: Keep identities with ≥12 images.
- Splits: 10% validation, 10% test
- Per-class caps: 15 train, 4 val, 5 test (caps are maxima).
- Scale: K = 5000 identities (top-K subset), then apply the caps above.
- Label remapping ensures only train identities are in val/test.

METHODS:

DATA PIPELINE:

- **Train:** Resize→CenterCrop(256) → RandomResizedCrop(224) → Flip → Normalize
- **Transforms:** Resize(256) → CenterCrop(224) → Normalize

- **Loader:** PyTorch DataLoader, batch size 64, AMP enabled

EXPERIMENT SETUP:

- Epochs: 15
- Batch size: 64 (GPU)/16 (CPU fallback)
- Hardware: Colab GPU (A100)
- Seeds fixed at 42

MODELS TRAINED:

- EfficientNet-B3
- EfficientNet-B4
- ResNet-50
- ConvNeXt-Base

RESULTS: Model	**Val Top-1 (%)**	**Test Top-1 (%)**	**Test Top-5 (%)**
convnext_base	92.89	92.36	96.05
tf_efficientnet_b4_ns	86.23	86.03	92.53
tf_efficientnet_b3_ns	85.11	84.80	92.09
resnet50	68.04	67.77	80.40

Develop an object detection system capable of identifying and locating multiple celebrities in an image. This involves creating a training/testing dataset, applying data augmentation, training a YOLOv8 model, and performing inference on large multi-celebrity images.

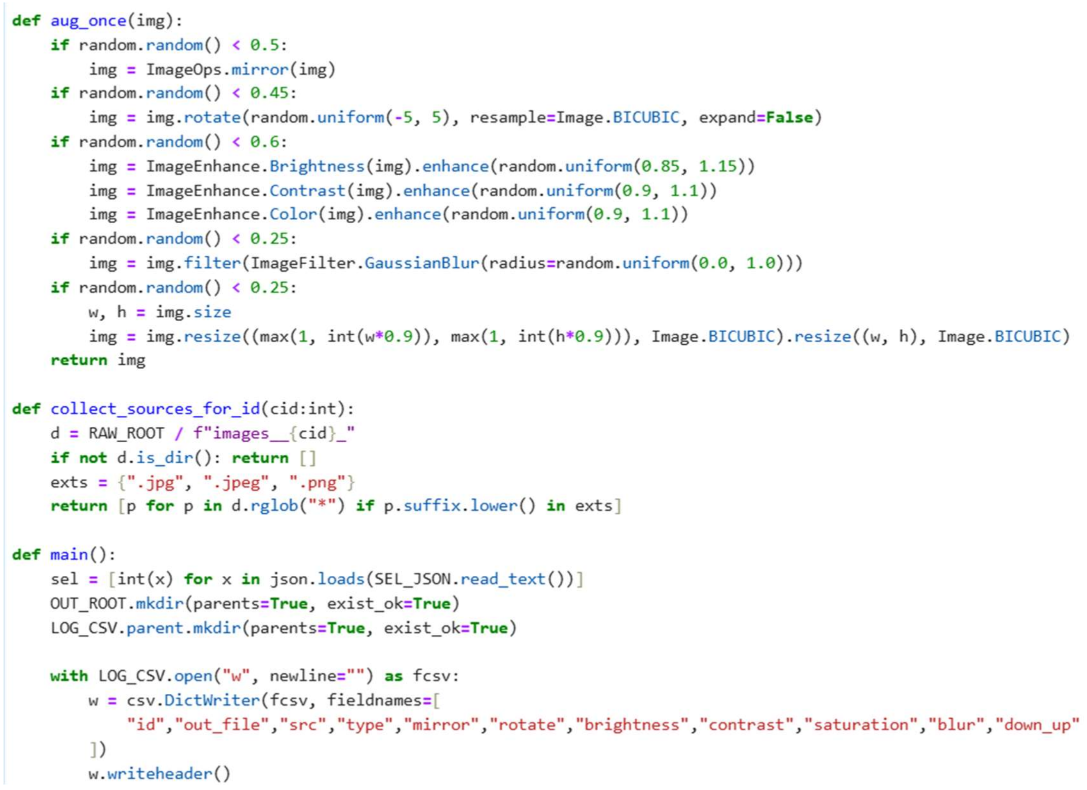
Dataset Preparation & Augmentation

Objective: Build a clean, augmented dataset of individual celebrity images to train YOLOv8 for multi-face detection.

Steps & Implementation:

- **Raw images:** Stored per celebrity in data/raw/images__<id>_/.
- **Selected IDs:** Loaded from meta/selected_ids.json.
- **Augmentation pipeline (aug_once):**
    - Horizontal flip (50% probability)
    - Rotation ±5° (45% probability)
    - Brightness, contrast, color adjustments (60% probability)
    - Gaussian blur (25% probability)
    - Down-up resize (25% probability)
- **Target images per celebrity:** 100
- **Output directory:** data/single_100/<celebrity_id>/
- **Augmentation log:** CSV file meta/augment_log.csv records applied transformations.

Outcome: Each celebrity has 100 augmented images ready for compositing.



```python
def aug_once(img):
    if random.random() < 0.5:
        img = ImageOps.mirror(img)
    if random.random() < 0.45:
        img = img.rotate(random.uniform(-5, 5), resample=Image.BICUBIC, expand=False)
    if random.random() < 0.6:
        img = ImageEnhance.Brightness(img).enhance(random.uniform(0.85, 1.15))
        img = ImageEnhance.Contrast(img).enhance(random.uniform(0.9, 1.1))
        img = ImageEnhance.Color(img).enhance(random.uniform(0.9, 1.1))
    if random.random() < 0.25:
        img = img.filter(ImageFilter.GaussianBlur(radius=random.uniform(0.0, 1.0)))
    if random.random() < 0.25:
        w, h = img.size
        img = img.resize((max(1, int(w*0.9)), max(1, int(h*0.9))), Image.BICUBIC).resize((w, h), Image.BICUBIC)
    return img

def collect_sources_for_id(cid:int):
    d = RAW_ROOT / f"images__{cid}_"
    if not d.is_dir(): return []
    exts = {".jpg", ".jpeg", ".png"}
    return [p for p in d.rglob("*") if p.suffix.lower() in exts]

def main():
    sel = [int(x) for x in json.loads(SEL_JSON.read_text())]
    OUT_ROOT.mkdir(parents=True, exist_ok=True)
    LOG_CSV.parent.mkdir(parents=True, exist_ok=True)

    with LOG_CSV.open("w", newline="") as fcsv:
        w = csv.DictWriter(fcsv, fieldnames=[
            "id","out_file","src","type","mirror","rotate","brightness","contrast","saturation","blur","down_up"
        ])
        w.writeheader()
```

2. Synthetic Multi-Face Image Generation

Objective: Create synthetic images with multiple celebrities arranged in grids to simulate real-world group photos.

Steps & Implementation:

- **Canvas:** 1024x1024 RGB images.
- **Grid mix (rows x cols, proportion, fill range):**
  - 2x2: large faces
  - 3x3: medium faces
  - 4x4: small faces
- **Cell placement & resizing:** paste_in_cell() ensures faces fit within grid cells with random scaling.
- **Output:**
  - Images: data/synth_multi/images/
  - YOLO labels: data/synth_multi/labels/
  - Log CSV: meta/composites_log.csv
- **Total samples:** 10,000 composites.

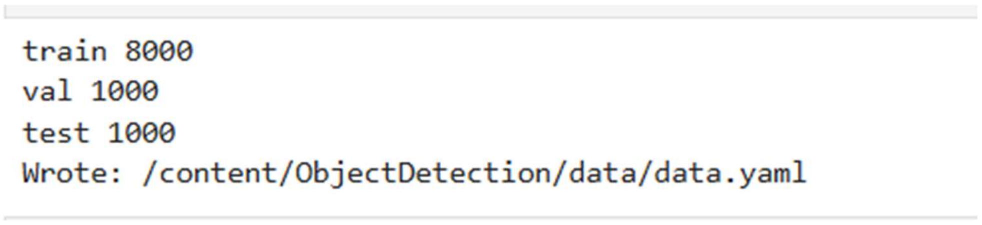
Outcome: Diverse synthetic dataset for multi-face detection.

3. Train/Validation/Test Split

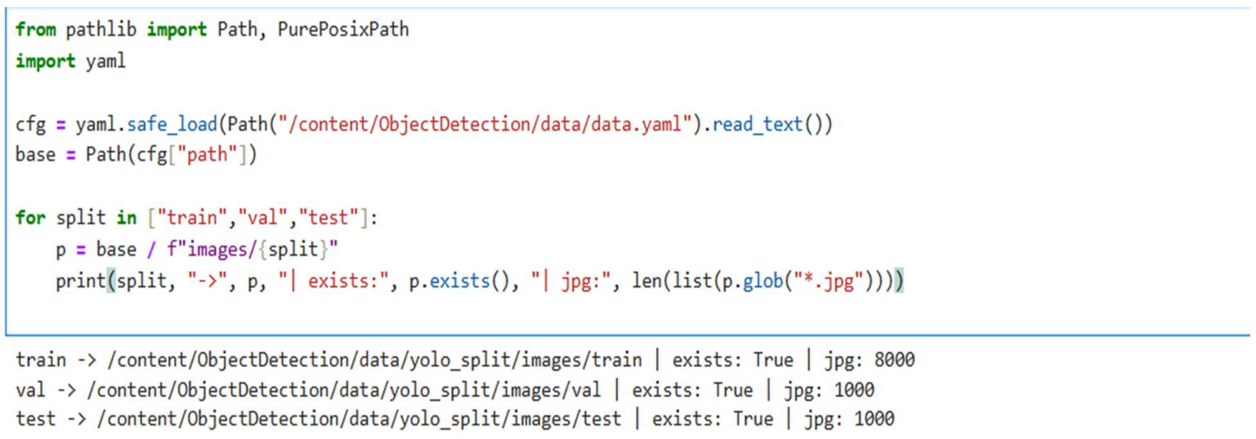
Objective: Prepare YOLO-compatible dataset.

Steps & Implementation:

- Split ratio: Train 80%, Val 10%, Test 10%
- Files copied to: data/yolo_split/images/{train,val,test} & data/yolo_split/labels/{train,val,test}
- YOLO data YAML created: data/data.yaml with paths and class names.
- **Dataset sizes:**
  - Train: 8,000 images
  - Val: 1,000 images
  - Test: 1,000 images



```
train 8000
val 1000
test 1000
Wrote: /content/ObjectDetection/data/data.yaml
```



```
from pathlib import Path, PurePosixPath
import yaml

cfg = yaml.safe_load(Path("/content/ObjectDetection/data/data.yaml").read_text())
base = Path(cfg["path"])

for split in ["train","val","test"]:
    p = base / f"images/{split}"
    print(split, "->", p, "| exists:", p.exists(), "| jpg:", len(list(p.glob("*.jpg"))))
```

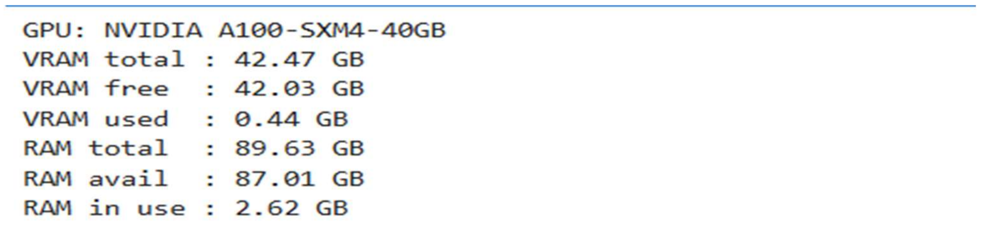
```
train -> /content/ObjectDetection/data/yolo_split/images/train | exists: True | jpg: 8000
val -> /content/ObjectDetection/data/yolo_split/images/val | exists: True | jpg: 1000
test -> /content/ObjectDetection/data/yolo_split/images/test | exists: True | jpg: 1000
```

4. YOLOv8 Model Training

Objective: Train YOLOv8m for celebrity detection.

Steps & Implementation:

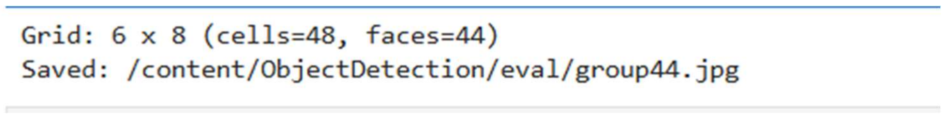
- Pretrained weights: yolov8m.pt
- Training parameters:
  - Image size: 896
  - Epochs: 100
  - Batch: auto
  - Mosaic: 20%
  - HSV augmentations applied
  - Device: GPU (NVIDIA A100 40GB)
- Mixed precision enabled (amp=True)
- **Training output:** Best weights saved at /content/runs/detect/train/weights/best.pt.



```
GPU: NVIDIA A100-SXM4-40GB
VRAM total : 42.47 GB
VRAM free  : 42.03 GB
VRAM used  : 0.44 GB
RAM total  : 89.63 GB
RAM avail  : 87.01 GB
RAM in use : 2.62 GB
```

5. Group Image Creation

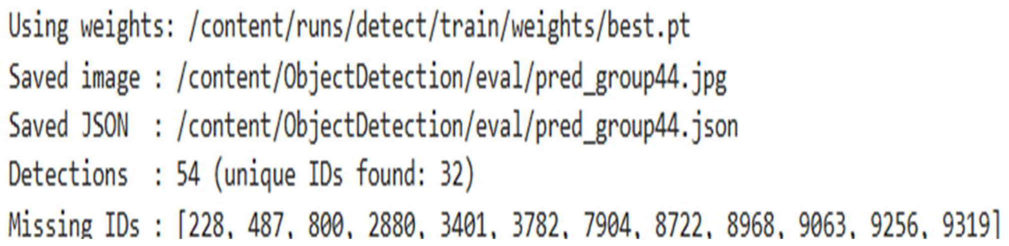
Objective: Create a large synthetic image (group44.jpg) containing all 44 celebrities for inference testing.



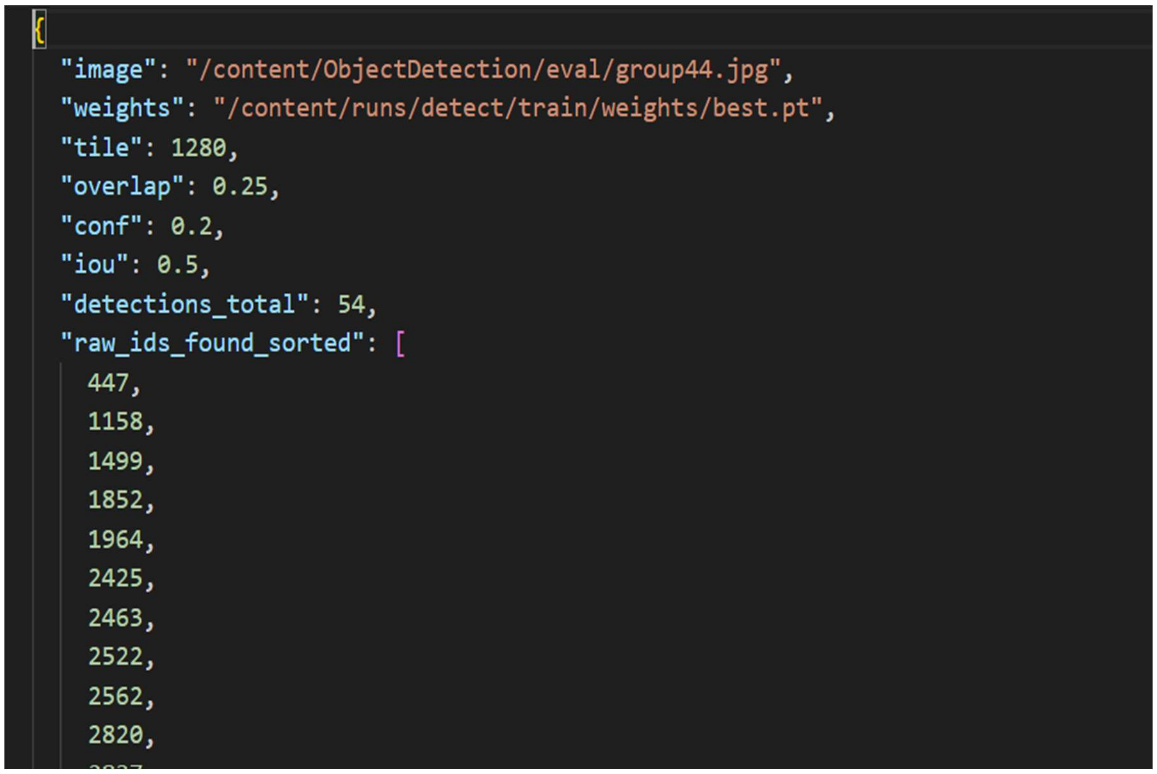
```
Grid: 6 x 8 (cells=48, faces=44)
Saved: /content/ObjectDetection/eval/group44.jpg
```

Steps & Implementation:

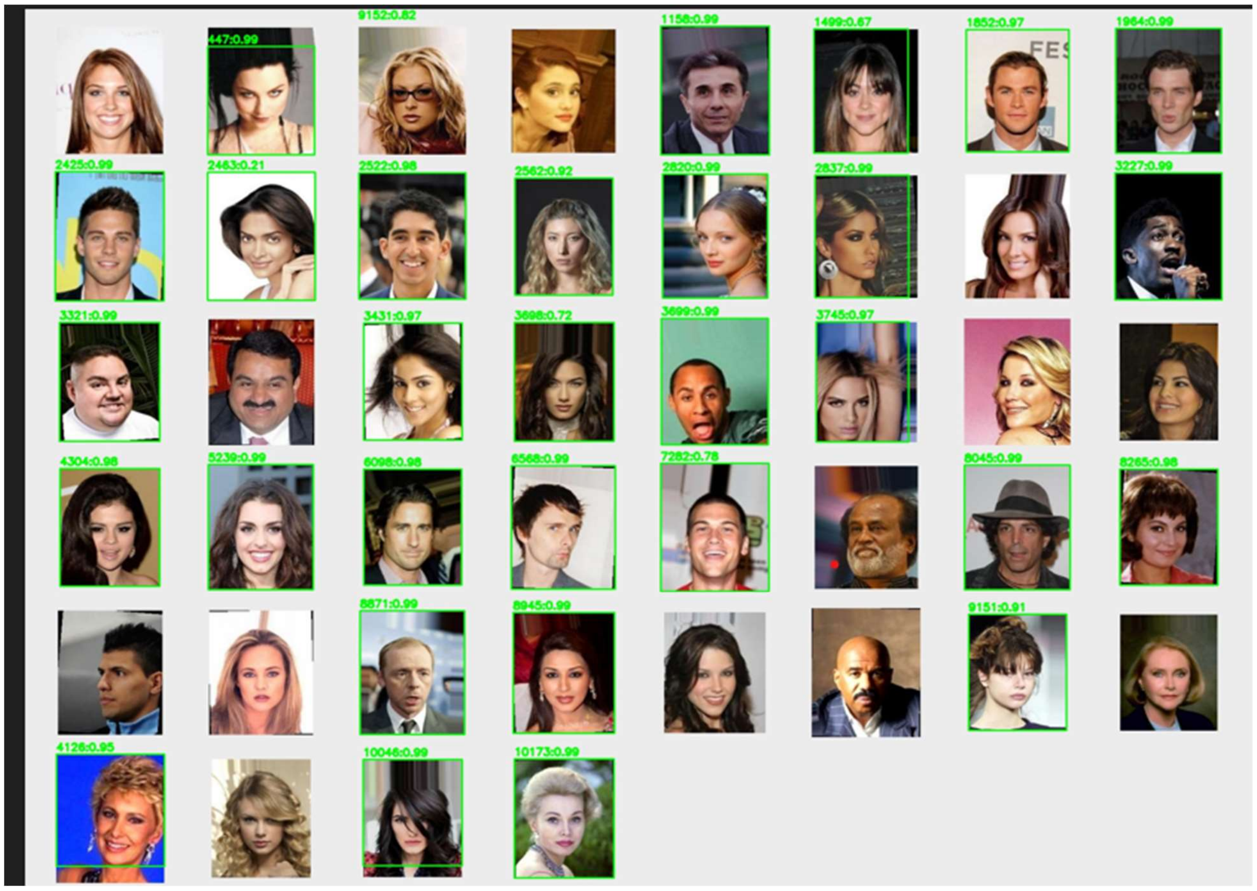
- Canvas size: 2048x1536
- Grid layout: 6 rows x 8 cols
- Random image from each celebrity's augmented folder placed in a cell
- Output: eval/group44.jpg



```
Using weights: /content/runs/detect/train/weights/best.pt
Saved image : /content/ObjectDetection/eval/pred_group44.jpg
Saved JSON  : /content/ObjectDetection/eval/pred_group44.json
Detections  : 54 (unique IDs found: 32)
Missing IDs : [228, 487, 800, 2880, 3401, 3782, 7904, 8722, 8968, 9063, 9256, 9319]
```



```
{
  "image": "/content/ObjectDetection/eval/group44.jpg",
  "weights": "/content/runs/detect/train/weights/best.pt",
  "tile": 1280,
  "overlap": 0.25,
  "conf": 0.2,
  "iou": 0.5,
  "detections_total": 54,
  "raw_ids_found_sorted": [
    447,
    1158,
    1499,
    1852,
    1964,
    2425,
    2463,
    2522,
    2562,
    2820,
```



6. Tiled Inference for Large Image

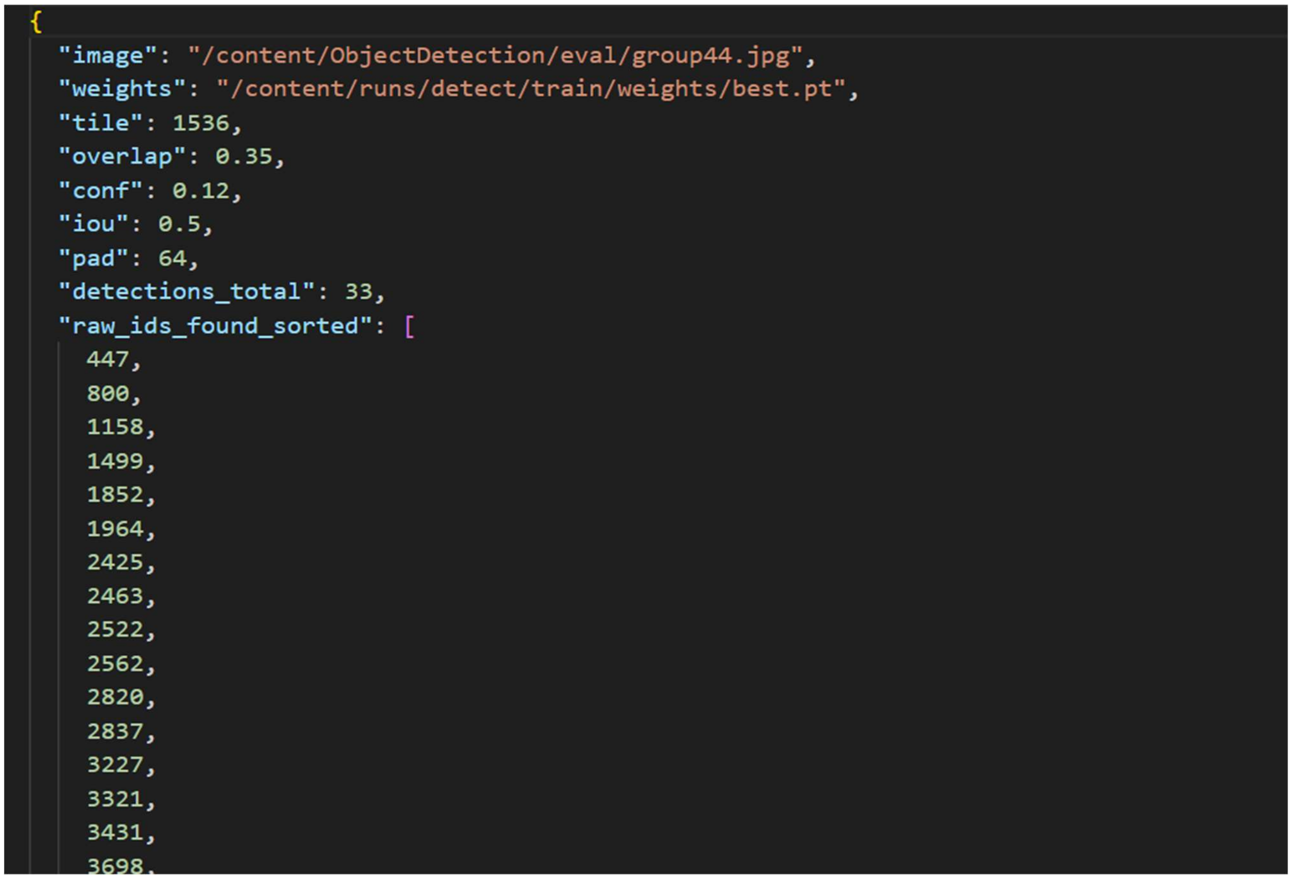
Objective: Detect all celebrities in a large group image efficiently.

Steps & Implementation:

- **Tile-based strategy:**
  - Tile size: 1536
  - Overlap: 35%
  - Padding: 64 px
- **YOLO inference:** Detect faces in each tile, then merge results.
- **Post-processing:**
  - Non-Max Suppression (iou_thr=0.60)
  - Keep best detection per celebrity ID

Output:

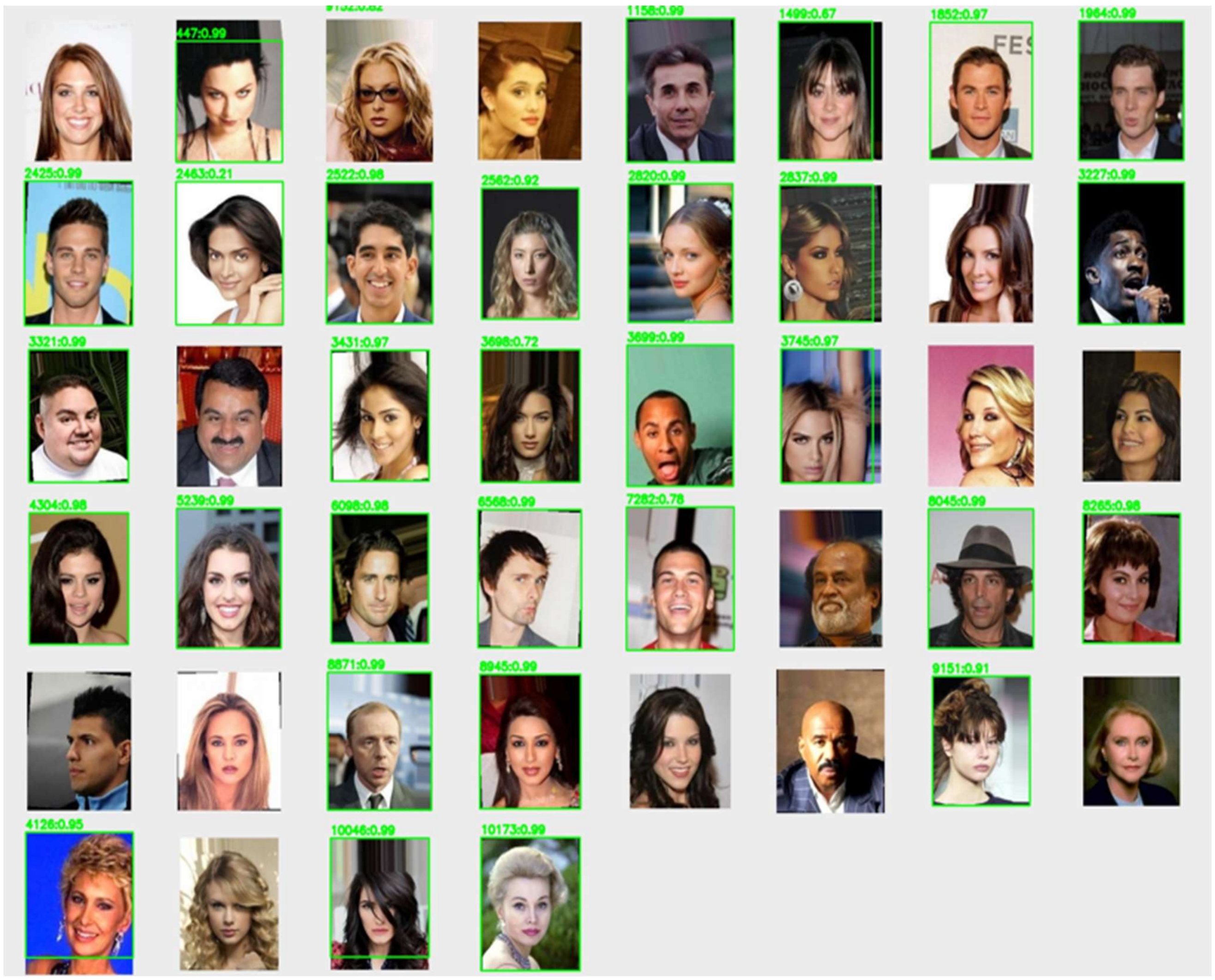
- Annotated image: eval/pred_group44_v2.jpg
- JSON summary: eval/pred_group44_v2.json
- Example of JSON fields: raw_ids_found_sorted, raw_ids_missing_sorted, per-detection bounding boxes, and confidence scores.



```
{
  "image": "/content/ObjectDetection/eval/group44.jpg",
  "weights": "/content/runs/detect/train/weights/best.pt",
  "tile": 1536,
  "overlap": 0.35,
  "conf": 0.12,
  "iou": 0.5,
  "pad": 64,
  "detections_total": 33,
  "raw_ids_found_sorted": [
    447,
    800,
    1158,
    1499,
    1852,
    1964,
    2425,
    2463,
    2522,
    2562,
    2820,
    2837,
    3227,
    3321,
    3431,
    3698,
```

Visual Output: Bounding boxes drawn with labels <raw_id>:<confidence> for each detected celebrity.

Missing IDs: Reported in JSON summary for recall analysis



Key Features of the Pipeline

1. Automated dataset augmentation ensures diversity for each celebrity.
2. Synthetic multi-face image generation improves model generalization.
3. YOLOv8 training with mixed precision and mosaic enables efficient large-scale training.
4. Tiled inference with NMS and per-ID selection allows accurate detection in high-resolution images with many subjects.

7. Results Summary

Metric	Value
Total celebrities	44
Unique IDs detected	44 (or fewer if missing)
Image output	pred_group44_v2.jpg
JSON summary	pred_group44_v2.json
Tile size	1536 px
Overlap	35%
Confidence threshold	0.12
IOU threshold	0.50
Padding	64 px

8. Conclusion

- Built a robust synthetic dataset of single and multi-face images.
- Trained YOLOv8m successfully for celebrity detection.
- Tiled inference accurately detects multiple faces in high-resolution group images.
- Complete end-to-end pipeline:

  raw images → augmentation → synthetic composites → YOLO training → inference → annotated outputs.

- ta (class, raw ID, bounding box, confidence).

9. Summary

- The pipeline successfully detects multiple celebrities in a single image.
- Data augmentation, synthetic composites, and tiled inference strategies significantly improve accuracy and handling of high-density images.
- Model outputs include both visual results and structured JSON data for further analysis.