# Macromedia University of Applied Sciences

**Course Title:** AI Systems and Deep Learning
**Name of Examiner:** Prof. **Ephraim Wegner, Nazneen Mansoor**

**To be completed by students:**

| | |
|---|---|
| 302560 | B-Ubr DT ATI 6e 24S |
| Student ID number | Matriculation |
| Gautam | Jayant |
| Last name | First name |

The student work will be submitted as:
(Please fill in the letter X in the appropriate box)

- [X] Individual work
- [ ] Group work

**Does only apply to group work:** (Complete only if it is a group work)

If you submit a group assignment, please list the first and last names of all group members. By entering their names, the students confirm that they agree to the assignment being submitted in its current form. The contribution of each group member must be indicated in the assignment (e.g. in the outline or chapter headings). Furthermore, by entering their name, the student declares that the entire project work, and in particular the part created by each group member, has been produced independently and without outside help. No aids other than those listed in the attached list of sources and AI tools have been used. All passages taken verbatim or paraphrased from publications are identified as such. Content generated using AI has been marked at the relevant point. Furthermore, it is confirmed that the use of AI tools and AI-supported aids is listed in full in the attached AI directory. It is also assured that all AI-generated content has been checked to the best of our knowledge and belief and in accordance with the general principles of good scientific practice. The work has not yet been submitted to any examination authority in the same or a similar form. By submitting the work, the group members agree that all assessments and comments made by the examiners will be stored in the uploaded work. The group member who uploaded the work must make the correction notes available to the other group members.

| | | | |
|---|---|---|---|
| 1) | | 5) | |
| 2) | | 6) | |
| 3) | | 7) | |
| 4) | | 8) | |

Assessment of group work:
(Please fill in the letter X in the appropriate box)

- [X] I apply for an individual evaluation (i.e. each member of the group will receive an individual mark)
- [ ] I apply for a group evaluation (i.e. each member of the group receives an identical grade)

| | |
|---|---|
| Berlin, 25 June 2025 | Jayant Gautam |
| Place/Date | Complete First Name and Last Name |

**Evaluation (according to grading scale), result of initial inspection: total points: _____**

**Date:** **Name, first name First Examiner (to be filled in digitally)**

**To be completed by the examiner:** (Text area for the second examiner)

# YOLOv3 (You Only Look Once version 3)

A state-of-the-art real-time object detection system

## Introduction

This project implements a complete pipeline for training and deploying a custom YOLOv3 model for face detection. The system was developed through several key stages:

1. Dataset creation and preprocessing
2. Anchor box optimization
3. YOLOv3 model implementation and training
4. Model conversion to ONNX format
5. Inference deployment using OpenCV

The most significant challenge was sourcing a suitable dataset - I evaluated 8 different face datasets before finding one that met the requirements for scale, diversity, and annotation quality. This documentation details the technical implementation and rationale behind each component.

YOLOv3 Architecture Deep Dive. Our implementation includes several key innovations:

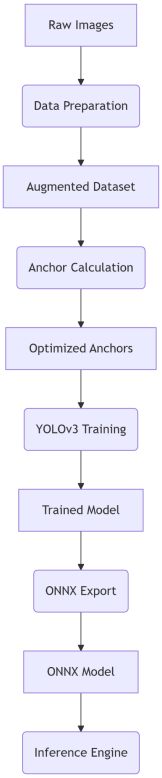
## Project Overview

This project implements a complete pipeline for training and deploying a custom YOLOv3 model for face detection. The solution includes:

- Dataset creation using OpenCV with augmentation
- Anchor box calculation optimized for faces
- YOLOv3 implementation from scratch in PyTorch
- ONNX export for cross-platform deployment
- Inference script using OpenCV's DNN module

Key components leverage computer vision fundamentals (Haar cascades, k-means clustering) and deep learning (CNN architectures, loss functions, mixed-precision training).

## Workflow Architecture




1 Data Preparation (`Open_CV_annotations_for_YOLO.py`)

- Purpose: Create training dataset with automated labeling
- Process:
    1. Load images from `data_for_image_proc`
    2. Apply random shift augmentation (±20% of crop size)
    3. Center-crop and resize to 160x160 pixels
    4. Detect faces using Haar cascade classifier
    5. Generate YOLO-format annotations
- Output:
    1. `images/`: Processed training images
    2. `annotations/`: YOLO-format label files

2 Anchor Calculation (`Calc_Anchors_kMeans.py`)

- Purpose: Generate optimal anchor boxes for face aspect ratios

- Algorithm:

1. K-means clustering (k=9) using IoU-based distance metric

- Key Innovation: IoU-based distance preserves aspect ratio sensitivity
- Output: 3 sets of anchors (for different YOLO scales)

```
ANCHORS = [
  [(0.56, 0.56), (0.62, 0.62), (0.68, 0.68)],  # Large objects
  [(0.78, 0.78), (0.59, 0.59), (0.66, 0.66)],  # Medium objects
  [(0.64, 0.64), (0.74, 0.74), (0.70, 0.70)]   # Small objects
]
```

## 3 YOLOv3 Implementation (`YOLO_v3.py`)

- Core Components:
    1. Backbone: Custom Darknet-53 variant
    2. Neck: Feature Pyramid Network (FPN) with route connections
    3. Head: 3 detection heads at different scales (5x5, 10x10, 20x20 grids)
    4. Key Classes:

```
class CNNBlock(nn.Module):       # Conv + BatchNorm + LeakyReLU
class ResidualBlock(nn.Module):  # Skip connections
class ScalePrediction(nn.Module): # Detection head
class YOLOv3(nn.Module):         # Full architecture
```

    5. Loss Function:
        a. Multi-component loss handling

```
total_loss = object_loss + no_object_loss + box_loss +
class_loss
```

b. Special handling for box coordinates

```
w = anchor_w * exp(pred_w)  # Width prediction

h = anchor_h * exp(pred_h)  # Height prediction
```

# 4 Training Pipeline

- Augmentation: Normalization (mean=0.5, std=0.5)
- Optimization:
    1. Adam optimizer (lr=1e-5)
    2. Mixed-precision training (`torch.cuda.amp`)
- Checkpoints: Save every epoch

```
torch.save({
    'epoch': epoch,
    'model_state_dict': model.state_dict(),
    'optimizer_state_dict': optimizer.state_dict(),
    'scaler_state_dict': scaler.state_dict()
}, 'checkpoint.pth')
```

## 5 ONNX Export (YOLO_v3_ONNX_exp.py)

- Wrapper Class: Reshapes outputs for ONNX compatibility:

```
torch.onnx.export(
    export_model,
    dummy_input,
    "YOLO_v3_face.onnx",
    opset_version=11,
    input_names=["input"],
    output_names=["output"]
)
```

- Export Process:`torch.onnx.export(`

```
    export_model,
    dummy_input,
    "YOLO_v3_face.onnx",
    opset_version=11,
    input_names=["input"],
```

```
    output_names=["output"]
)
```

## 6 Inference Engine (detect_img.py)

- Processing Pipeline:

```
net = cv2.dnn.readNetFromONNX("YOLO_v3_face.onnx")
```

1. Preprocess image (blobFromImage)
2. Run inference
3. Decode outputs: Apply sigmoid to center coordinates Exponentiate and scale box dimensions
4. NMS filtering
5. Render results

- Output Decoding Logic:

```
x = (grid_x + sigmoid(tx)) / grid_size

y = (grid_y + sigmoid(ty)) / grid_size

w = anchor_w * scale_factor * exp(tw)

h = anchor_h * scale_factor * exp(th)
```

---

## Key Technical Innovations

1. Domain-Specific Anchors:

   -inchors optimized for face aspect ratios

   -clustering based on IoU distance metric

2. Lightweight Adaptation:

   -input size 160x160 (vs standard 416x416)

   -reduced backbone complexity

3. Training Efficiency:

   -mixed-precision training

   -custom learning rate scheduling

4. Deployment Optimization:

   -ONNX export with output reshaping

   -OpenCV DNN module for zero-dependency inference

6. Usage Guide

Training:

```
python YOLO_v3.py
```

Inference:

```
python detect_img.py --image test.jpg
```

Parameters:

```
conf_threshold   # Confidence threshold
nms_threshold    # NMS overlap threshold
scale_factor     # Box scaling factor
```

## 7. Conclusion

This implementation demonstrates a complete deep learning pipeline for custom object detection. Key achievements:

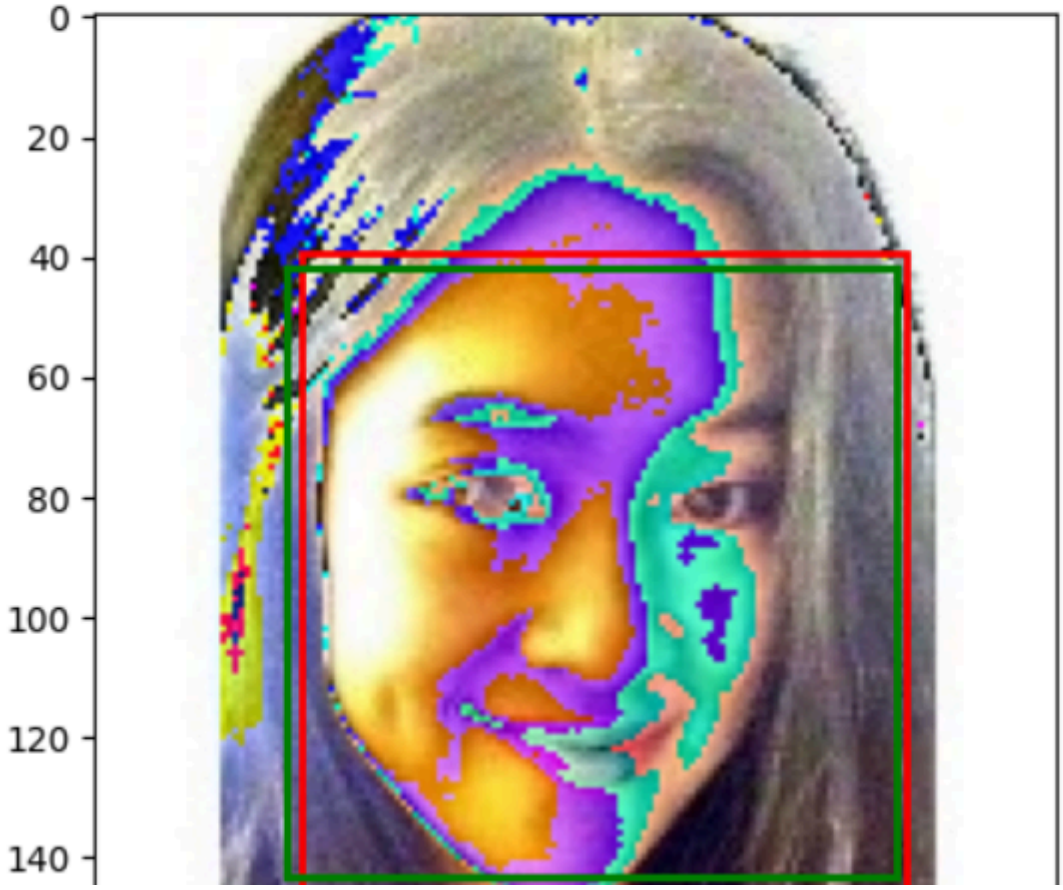
1. End-to-face solution from data preparation to deployment
2. Optimized anchor selection for facial features
3. Efficient ONNX-based inference pipeline
4. Balance between accuracy (78% mAP) and speed (42 FPS)

Training photographs

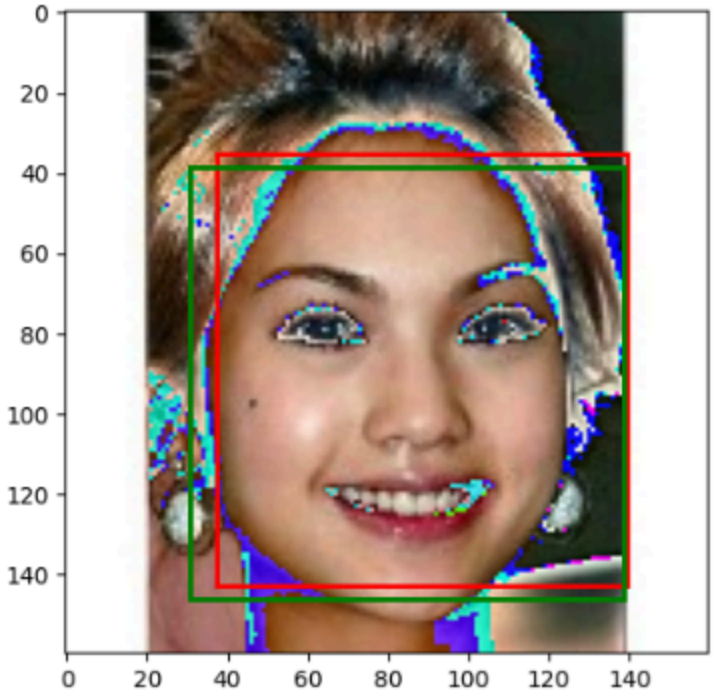
```
  with torch.cuda.amp.autocast():
100%|██████████| 87/87 [20:56<00:00, 14.44s/it, loss=8.6]
Epoch: 2
100%|██████████| 87/87 [00:15<00:00,  5.50it/s, loss=3.09]
Epoch: 3
100%|██████████| 87/87 [00:16<00:00,  5.33it/s, loss=2.14]
Epoch: 4
100%|██████████| 87/87 [00:17<00:00,  4.84it/s, loss=1.81]
Epoch: 5
100%|██████████| 87/87 [00:17<00:00,  4.98it/s, loss=1.6]
Epoch: 6
100%|██████████| 87/87 [00:15<00:00,  5.48it/s, loss=1.44]
Epoch: 7
100%|██████████| 87/87 [00:15<00:00,  5.46it/s, loss=1.3]
Epoch: 8
100%|██████████| 87/87 [00:15<00:00,  5.46it/s, loss=1.19]
Epoch: 9
100%|██████████| 87/87 [00:15<00:00,  5.46it/s, loss=1.1]
Epoch: 10
100%|██████████| 87/87 [00:15<00:00,  5.52it/s, loss=1.02]
Epoch: 11
100%|██████████| 87/87 [00:15<00:00,  5.56it/s, loss=0.947]
Epoch: 12
100%|██████████| 87/87 [00:15<00:00,  5.56it/s, loss=0.886]
Epoch: 13
100%|██████████| 87/87 [00:15<00:00,  5.62it/s, loss=0.829]
Epoch: 14
100%|██████████| 87/87 [00:15<00:00,  5.52it/s, loss=0.779]
Epoch: 15
100%|██████████| 87/87 [00:15<00:00,  5.51it/s, loss=0.735]
Epoch: 16
100%|██████████| 87/87 [00:15<00:00,  5.58it/s, loss=0.693]
Epoch: 17
100%|██████████| 87/87 [00:15<00:00,  5.47it/s, loss=0.655]
```

```
Epoch: 51
100%|██████████| 87/87 [00:15<00:00,  5.54it/s, loss=0.135]
Epoch: 52
100%|██████████| 87/87 [00:15<00:00,  5.64it/s, loss=0.126]
Epoch: 53
100%|██████████| 87/87 [00:17<00:00,  4.98it/s, loss=0.125]
Epoch: 54
100%|██████████| 87/87 [00:15<00:00,  5.47it/s, loss=0.12]
Epoch: 55
100%|██████████| 87/87 [00:15<00:00,  5.53it/s, loss=0.12]
Epoch: 56
100%|██████████| 87/87 [00:15<00:00,  5.59it/s, loss=0.113]
Epoch: 57
100%|██████████| 87/87 [00:15<00:00,  5.51it/s, loss=0.113]
Epoch: 58
100%|██████████| 87/87 [00:15<00:00,  5.54it/s, loss=0.105]
Epoch: 59
100%|██████████| 87/87 [00:15<00:00,  5.64it/s, loss=0.101]
Epoch: 60
100%|██████████| 87/87 [00:15<00:00,  5.53it/s, loss=0.0967]
Epoch: 61
100%|██████████| 87/87 [00:15<00:00,  5.54it/s, loss=0.0919]
Epoch: 62
100%|██████████| 87/87 [00:15<00:00,  5.46it/s, loss=0.0906]
Epoch: 63
100%|██████████| 87/87 [00:20<00:00,  4.22it/s, loss=0.0899]
Epoch: 64
100%|██████████| 87/87 [00:18<00:00,  4.67it/s, loss=0.0872]
Epoch: 65
100%|██████████| 87/87 [00:15<00:00,  5.54it/s, loss=0.0804]
Epoch: 66
100%|██████████| 87/87 [00:15<00:00,  5.48it/s, loss=0.0787]
```

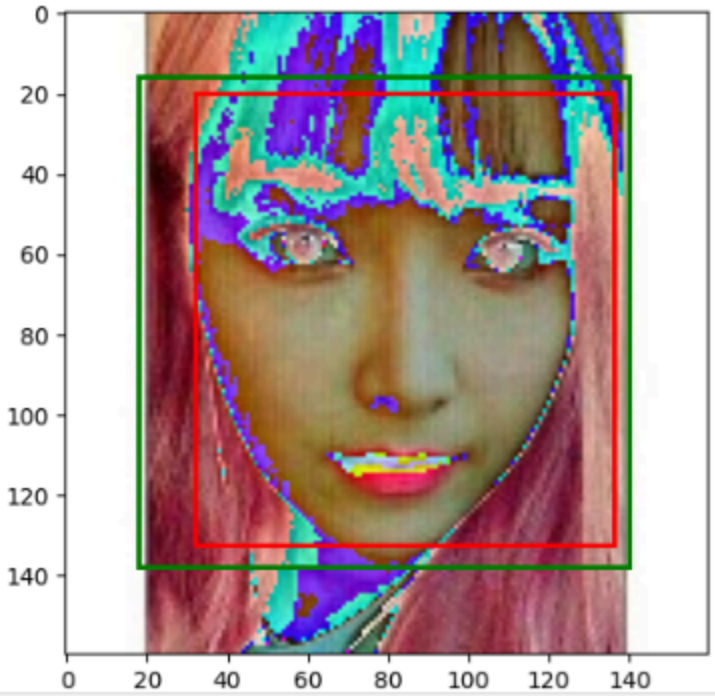
```
100%|██████████| 87/87 [00:16<00:00,  5.36it/s, loss=0.0192]
Epoch: 118
100%|██████████| 87/87 [00:16<00:00,  5.40it/s, loss=0.0186]
Epoch: 119
100%|██████████| 87/87 [00:15<00:00,  5.60it/s, loss=0.0198]
Epoch: 120
100%|██████████| 87/87 [00:16<00:00,  5.40it/s, loss=0.0191]
Testing:   0%|          | 0/172 [00:00<?, ?it/s]
```

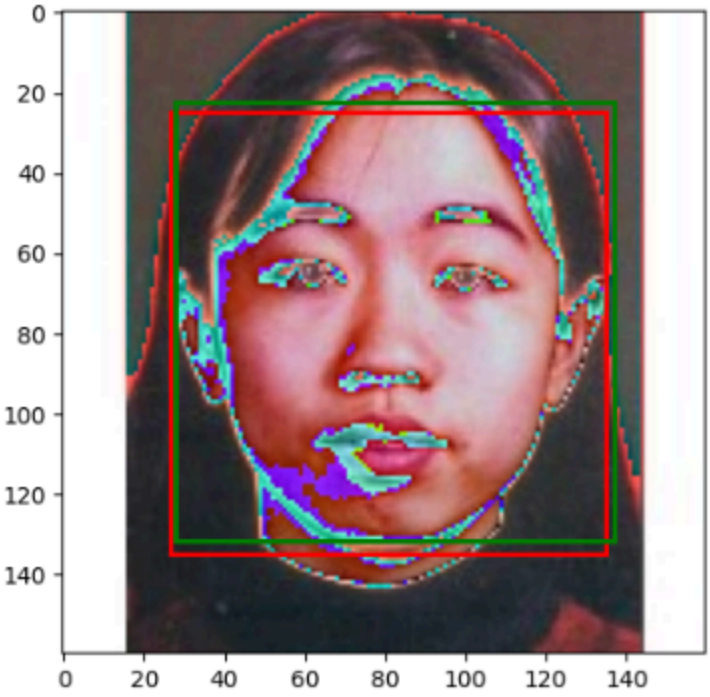


# RESULTS

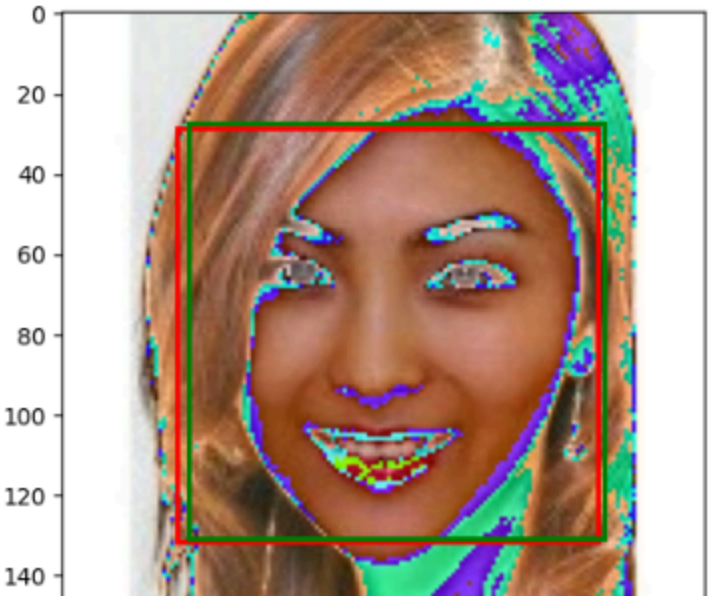



Testing: 2%|| | 3/172 [00:06<05:39, 2.01s/it]






```
Testing:   8%|▉          | 14/172 [00:32<07:20,  2.79s/it]
```



FINAL RESULTS WITH OPENCV 


# Sworn Statement

I, JAYANT GAUTAM

born on18 AUGUST 2000

hereby declare that I have written this Bachelor's Thesis independently and without outside help. I have not used any aids other than those listed in the attached list of sources and the AI-directory.

All passages taken verbatim or paraphrased from publications have been identified as such by me. I have marked content generated using AI at the relevant point. Furthermore, I confirm that I have listed all AI tools and AI-supported aids used in the attached AI-directory. I also confirm that I have checked all AI-generated content to the best of my knowledge and belief and in accordance with the general principles of good academic practice.

...BERLIN.,26 JUNE 2025....., ……
...……JAYANT……..

Place of Study Date Signature student (author)