

#Challenge.2 Human Detection

1. Introduction

The objective of this challenge is human detection, specifically localizing humans in provided images using bounding boxes. The key challenges include:

- Variability in Human Appearance Humans can appear in various poses, scales, and occlusions, making consistent detection difficult.
- Resource Constraints The use of pretrained models is strictly prohibited, requiring training from scratch using limited resources.
- Cluttered Backgrounds Scenes may contain complex or dynamic backgrounds, which interfere with robust human detection.

To tackle these issues, I adopt a **Faster R-CNN**-based detector with a **MobileNetV3-FPN backbone from TorchVision**, trained entirely from scratch with **no external pretrained weights**.

2. Model Pipeline

The model pipeline involves the following key steps:

1) Custom Implementation:

1. **Model Initialization:** Initialize Faster R-CNN with MobileNetV3-FPN, **setting weights=None** to avoid pretrained weight loading.
2. **Predictor Replacement:** Redefine the final classifier to support 2 classes (person and background).
3. **Training:** Use bounding box annotations formatted in the COCO style (i.e., [x, y, width, height]) and apply standard Faster R-CNN loss for end-to-end training.

2) Given by Challenge :

4. **Evaluation:** Compute mAP@0.5 using IoU-based matching and 11-point interpolated AP. The evaluation code and metric were provided and must not be altered.
 5. **Postprocessing:** Apply NMS and confidence filtering to yield final detection results. This logic is embedded in the provided evaluation script..
-

#Challenge.2 Human Detection

3. Pipeline Details

3.1 Model Initialization : From Scratch(i.e., training with randomly initialized weights without using pretrained models)

3.1.a) TorchVision Faster R-CNN The model is instantiated using:

```
# Load base model
base_model = fasterrcnn_mobilenet_v3_large_fpn(weights=None, weights_backbone=None)
in_features = base_model.roi_heads.box_predictor.cls_score.in_features
base_model.roi_heads.box_predictor = FastRCNNPredictor(in_features, num_classes)

#11 Model, optimizer and scheduler.
# You may change the settings.

#####
model = fasterrcnn_mobilenet_v3_large_fpn(weights=None,weights_backbone=None)
```

The function `**fasterrcnn_mobilenet_v3_large_fpn**` from `torchvision.models.detection` instantiates the entire Faster R-CNN detection pipeline. This includes the following major components:

- `GeneralizedRCNNTransform`: responsible for preprocessing input images (resize and normalize),
- `BackboneWithFPN`: consists of the **MobileNetV3 Large backbone** and the Feature Pyramid Network (FPN),
- `RegionProposalNetwork (RPN)`: proposes candidate object bounding boxes,
- `ROIHeads`: performs classification and bounding box regression based on the proposals.

By setting both `weights=None` and `weights_backbone=None`, all of the model's trainable parameters—including those in the backbone, FPN, RPN, and ROI heads—are randomly initialized from scratch. **No pretrained weights are loaded into any part of the model.**

#Challenge.2 Human Detection

3.1.b) Classification Head Replacement i modify the classification head using:

```
FastRCNNPredictor(in_features, num_classes=2)
```

```
num_classes = 2
in_features = model.roi_heads.box_predictor.cls_score.in_features
model.roi_heads.box_predictor = FastRCNNPredictor(in_features, num_classes)
```

3.1.c) Main pipeline:

```
GeneralizedRCNNTransform → BackboneWithFPN → RPN → RoI Align →
Classification Head
```

3.2 Training Configuration

3.2.a) Loss Function Loss computation is automatically handled by the Faster R-CNN model and includes:

- Classification Loss
- Bounding Box Regression Loss
- RPN Objectness Loss
- RPN Box Regression Loss

3.2.b) Optimization Strategy Adam optimizer is used with a learning rate of 0.0005, betas set to (0.9, 0.999), and weight decay of 0.0005. No learning rate scheduler was applied during training.

3.3 Evaluation Metric

3.3.a) IoU Matching Detection boxes are matched with GT boxes based on $\text{IoU} \geq 0.5$.

3.3.b) Mean Average Precision mAP@0.5 is calculated using 11-point interpolation across various recall levels.

#Challenge.2 Human Detection

4. Performance Evaluation

- **Metric:** mAP@0.5 on validation dataset.
- **Result:** Achieved mAP of **0.5431**, showing effective training from scratch without pretrained models.
But total evaluation time was approximately **119.50 seconds**.



5. Conclusion

While the Faster R-CNN model trained from scratch demonstrated effective human detection capabilities, several limitations were observed throughout the evaluation phase.

First, the **total inference time exceeded the challenge constraint** of 90 seconds, with a measured evaluation time of approximately **119.50 seconds**. This is likely due to the **two-stage nature of the Faster R-CNN architecture**, where the Region Proposal Network (RPN) and RoI head operate sequentially, making it inherently slower compared to single-stage detectors.

Second, some detection results exhibited **bounding box misalignment** with ground-truth (GT) annotations. In certain cases, the predicted box was noticeably larger or smaller than the GT box, suggesting that the regressor may have failed to converge tightly due to either ambiguous edges or lack of sufficient data.

Additionally, false positives were observed in test images that contained **no humans**. For example, one test image included a giraffe, which the model mistakenly classified as a human. This highlights a generalization issue, where the model occasionally responds to **human-like shapes or posture** despite having no pretrained semantic priors.

Overall, although the model successfully learned human detection without any pretrained weights, the trade-offs in speed and precision must be considered. Future improvements

#Challenge.2 Human Detection

could involve adopting faster single-stage architectures or incorporating hard negative mining to reduce confusion with non-human objects.