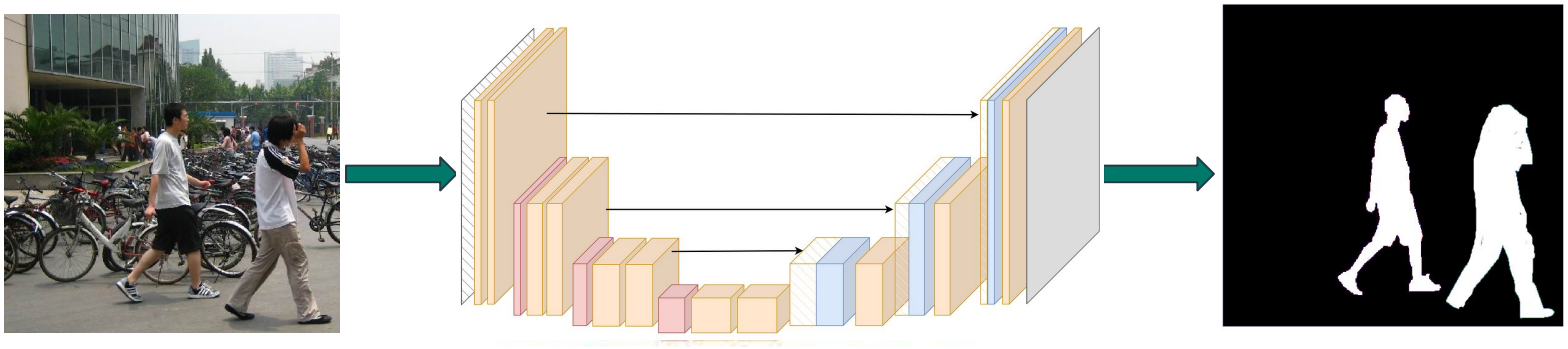
Powering up pedestrian segmentation with initial detection stage

Sharare Zolghadr, Donatas Vaiciukevičius, Kateryna Skurativska

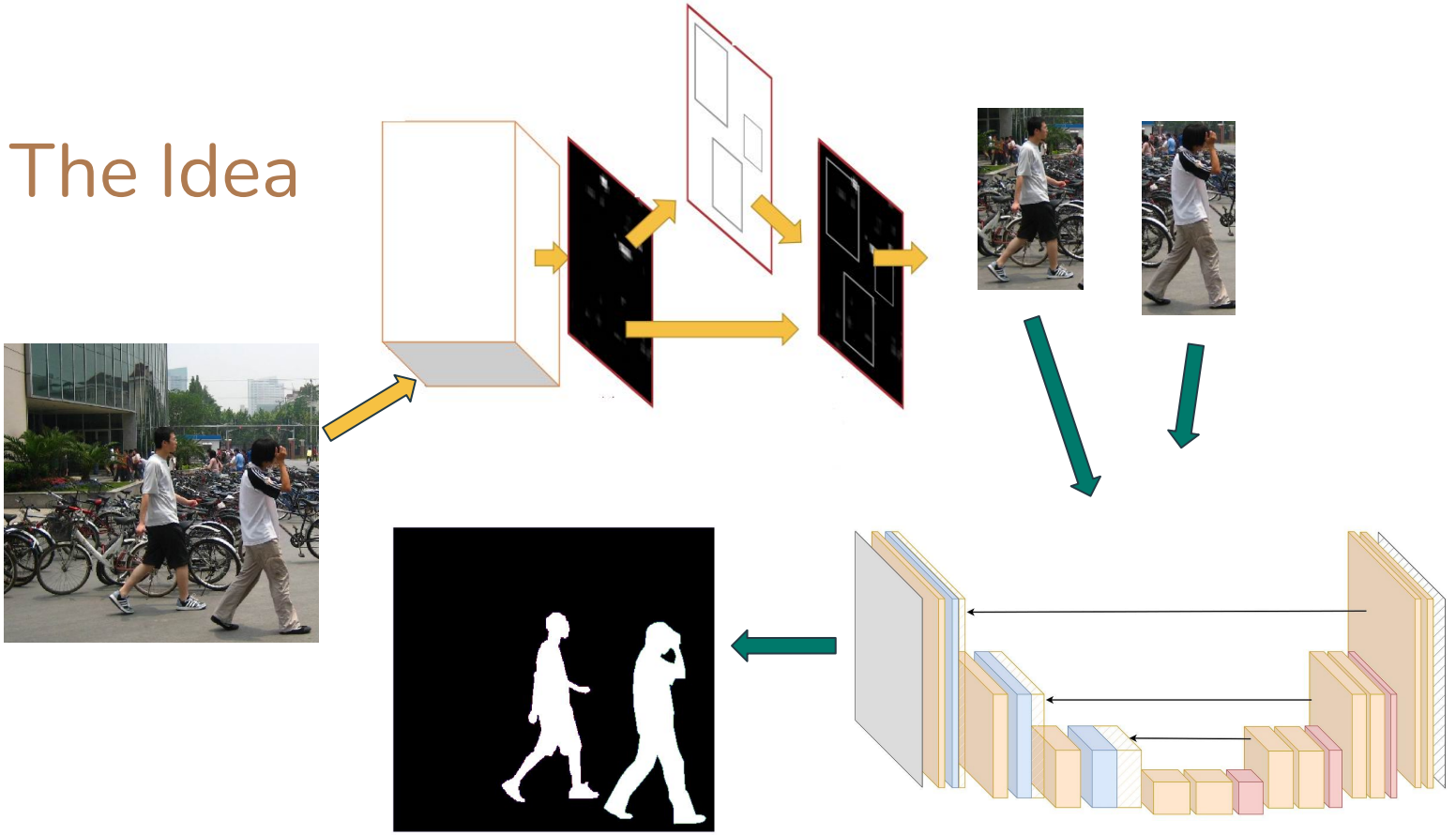
January 2024, UNIPD

Table of contents

The Idea



The Idea



Dataset Overview

Penn-Fudan pedestrian dataset:

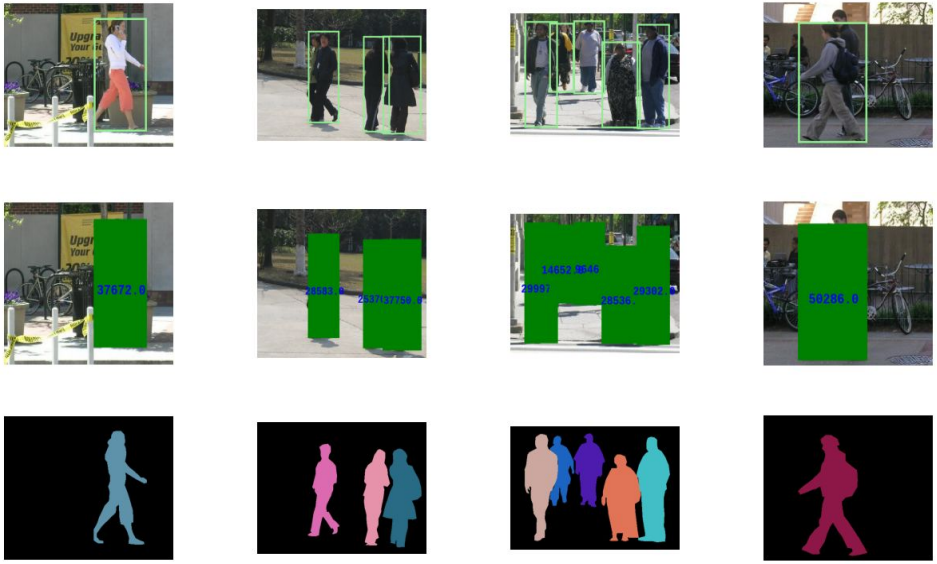


Image database consisting of ***170*** images with ***345*** labeled pedestrians:

- 96 images (University Pennsylvania)
- 74 images (Fudan University)

The heights of labeled pedestrians in this database fall into [180,390] pixels.
All labeled pedestrians are straight up.

Motivation:

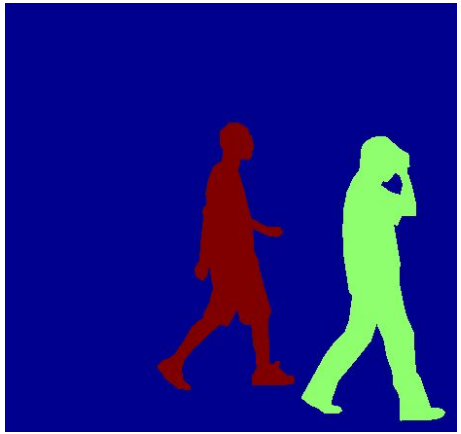
- Training models on small datasets requires less computational and time resources
- Compact size of dataset enables to speed up the process of exploring various models

Dataset Annotation

PennFudanPed:

- ➔ **PNGImages**
- ➔ **PedMasks**
- ➔ **Annotations**

FudanPed00009.img



FudanPed00009_mask.img

```
# Compatible with PASCAL Annotation Version 1.00
Image filename : "PennFudanPed/PNGImages/FudanPed00009.png"
Image size (X x Y x C) : 465 x 441 x 3
Database : "The Penn-Fudan-Pedestrian Database"
Objects with ground truth : 2 { "PASpersonWalking" "PASpersonWalking" }
# Note there may be some objects not included in the ground truth list for they are severe-occluded
# or have very small size.
# Top left pixel co-ordinates : (1, 1)
# Details for pedestrian 1 ("PASpersonWalking")
Original label for object 1 "PASpersonWalking" : "PennFudanPed"
Bounding box for object 1 "PASpersonWalking" (Xmin, Ymin) - (Xmax, Ymax) : (306, 138) - (453, 430)
Pixel mask for object 1 "PASpersonWalking" : "PennFudanPed/PedMasks/FudanPed00009_mask.png"

# Details for pedestrian 2 ("PASpersonWalking")
Original label for object 2 "PASpersonWalking" : "PennFudanPed"
Bounding box for object 2 "PASpersonWalking" (Xmin, Ymin) - (Xmax, Ymax) : (157, 124) - (298, 398)
Pixel mask for object 2 "PASpersonWalking" : "PennFudanPed/PedMasks/FudanPed00009_mask.png"
```

FudanPed00009.txt

Data preprocessing

1) Zero-padding in dimensions of the maximum size of the image was applied to all of the images. This resulted in a consistent image dimensions of 608x1024 (to fit the model's requirement of input dimensions should be divisible by 32).
2) As the ResNet model used for the Unet++ backbone was pre-trained on ImageNet, we have normalized the images to match its mean and standard deviation.
3) Split data into train, validation and test set:
    - Train set - 119 images
    - Validation set - 26 images
    - Test set - 25 images

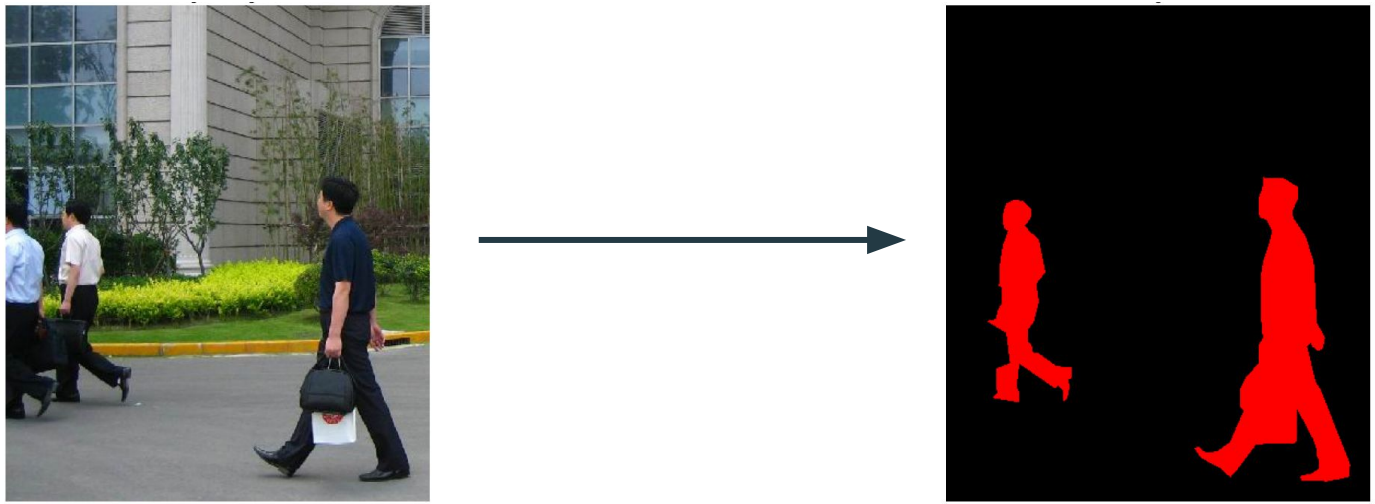
Data preprocessing

1) Zero-padding in dimensions of the maximum size of the image was applied to all of the images. This resulted in a consistent image dimensions of 608x1024.
2) As the ResNet model used for the Unet++ backbone was pre-trained on ImageNet, we have normalized the images to match its mean and standard deviation.
3) Random cropping augmentation was used to reduce the GPU memory usage and increase training speed. Crops of size 576x576 were generated, which both matched the input size to the U-Net in the original paper, as well as fit the model's requirement of input dimensions being divisible by 32.
4) Split data into train, validation and test set:
    - Train set - 119 images
    - Validation set - 26 images
    - Test set - 25 images

Semantic segmentation

Goal of semantic image segmentation is to label each pixel of an image with a corresponding class (Pedestrian vs background)

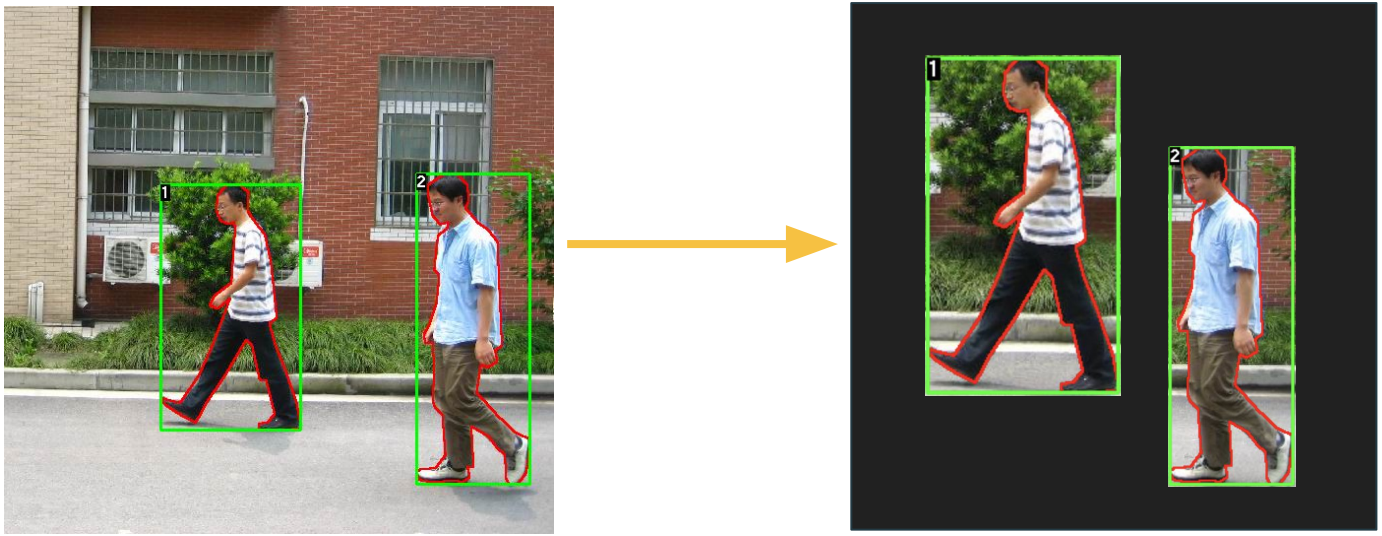
Four models were tested for semantic segmentation on full images:
U-Net or U-Net++ with ResNet-18 or ResNet-34 backbones.



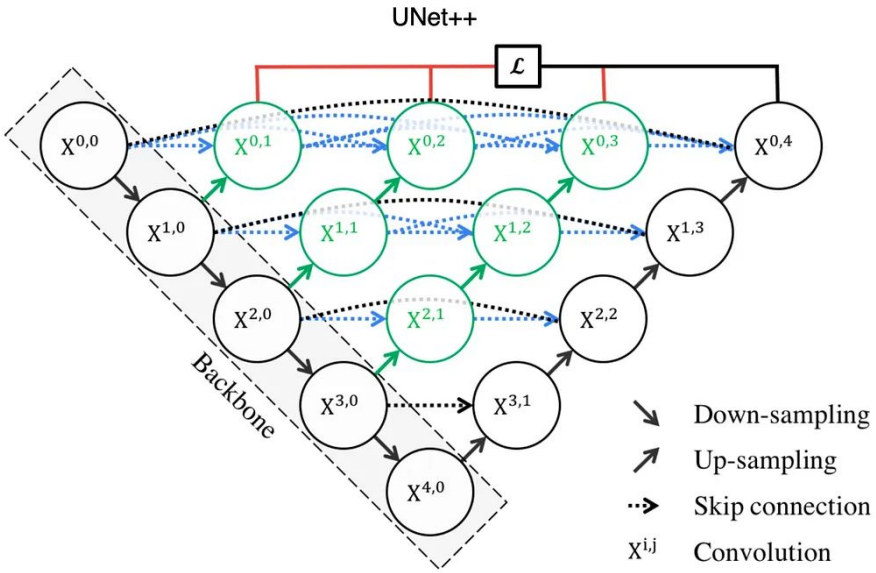
Semantic segmentation

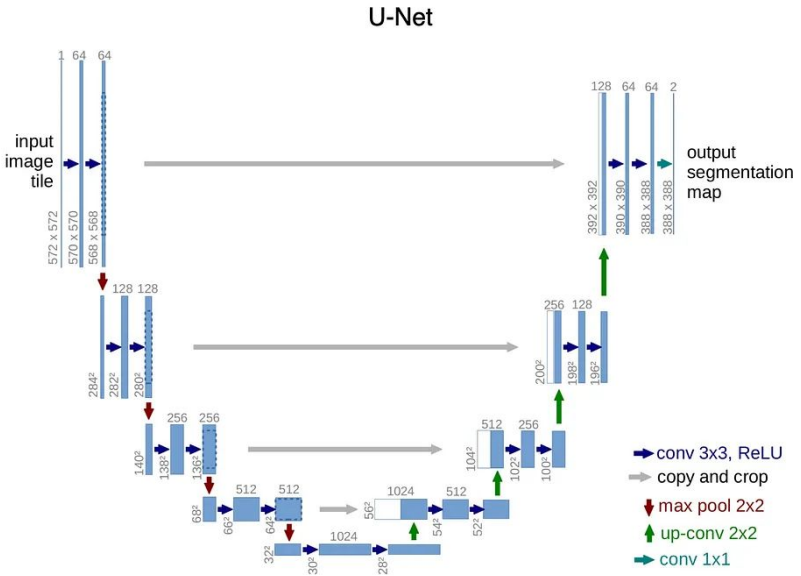
Goal of semantic image segmentation is to label each pixel of an image with a corresponding class of what is being represented.

Four models were tested for semantic segmentation on full images:
U-Net or U-Net++ with ResNet-18 or ResNet-34 backbones.



U-Net and U-Net++

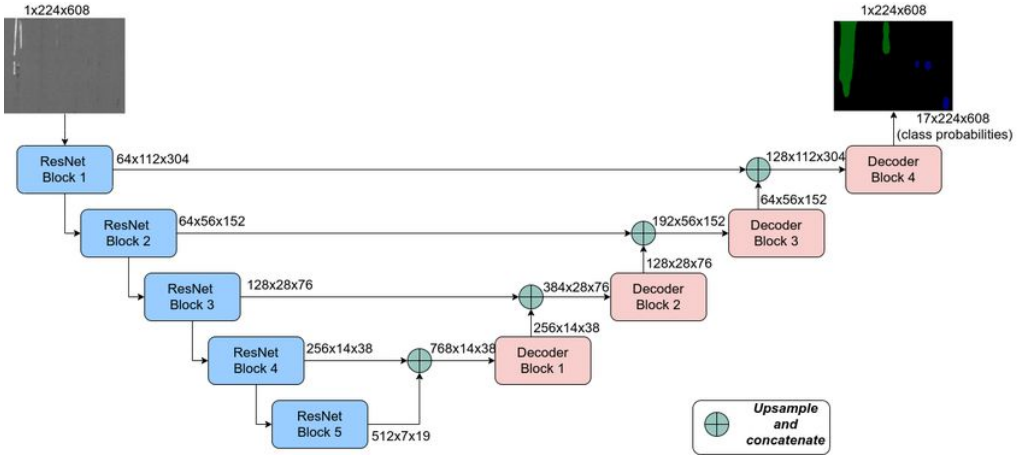




ResNet-18 and ResNet-34

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

U-Net with ResNet-18 backbone



Combination:

The U-Net structure with a ResNet backbone integrates the ResNet architecture into the encoder part of the U-Net.

Benefits:

The ResNet backbone brings the benefits of residual connections and the ability to capture complex features, enhancing the U-Net's feature extraction capabilities.

The combination aims to leverage the strengths of both architectures for better performance in semantic segmentation tasks.

Model setup

The ResNet backbone is initialized with weights pre-trained on ImageNet.

Number of epochs	70-90 with early stopping
Criterion	Focal Loss $= -\alpha_t(1-p_t)^{\gamma}\log(p_t)$ (alpha = 0.25, gamma = 2.0)
Optimizer	Adam(learning rate= 0.0005, weights_decay = 0.0005, patience = 20)
Scheduler	ReduceLROnPlateu('min', patience=10, factor=0.1, verbose=True)

Model setup

The ResNet backbone is initialized with weights pre-trained on ImageNet.
We did not perform fine-tuning for the models, as after initialization, only model encoder weights were pretrained

Number of epochs	250
Criterion	Focal Loss $= -\alpha_t(1-p_t)^{\gamma}\log(p_t)$ (alpha = 0.25, gamma = 2.0)
Optimizer	Adam(learning rate= 0.0005, weights_decay = 0.0005, patience = 20)
Scheduler	ReduceLROnPlateu('min', patience=10, factor=0.1, verbose=True)

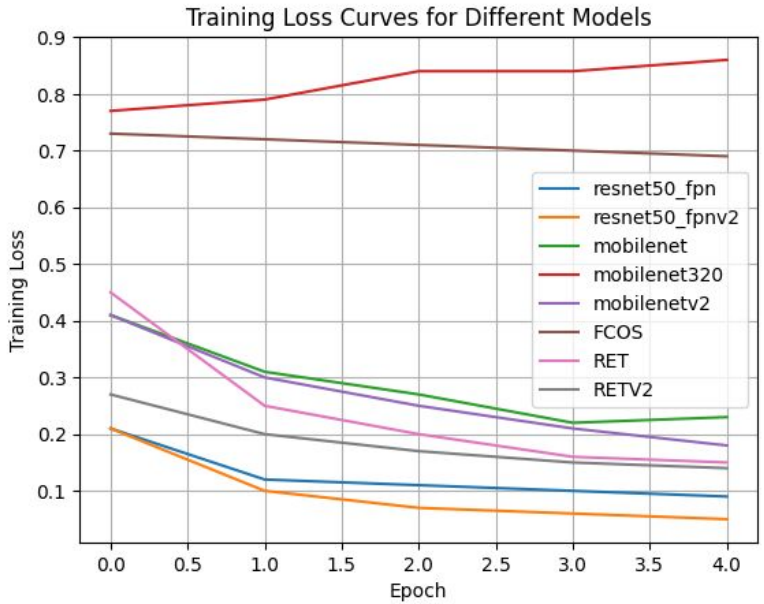
Results

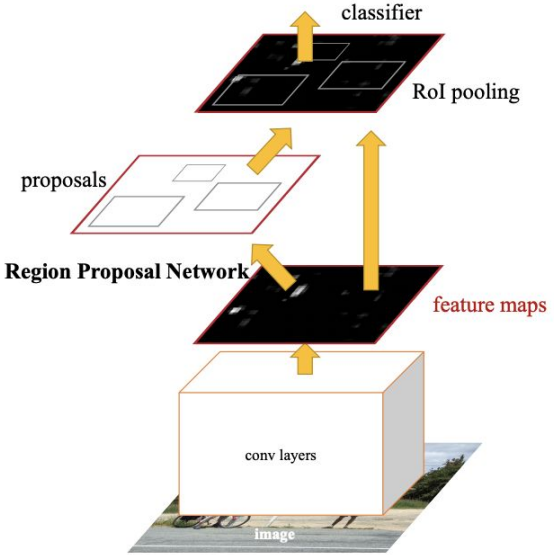
Model	Backbone	IoU	F1 Score
U-Net	ResNet-18	**0.608**	**0.748**
U-Net	ResNet-34	0.550	0.702
U-Net++	ResNet-18	0.588	0.733
U-Net++	ResNet-34	0.584	0.729

Model	Backbone	Train-Test IoU Delta	Train-Test F1 Delta
U-Net	ResNet-18	0.267	0.180
U-Net	ResNet-34	0.269	0.188
U-Net++	ResNet-18	0.282	0.192
U-Net++	ResNet-34	0.272	0.188

Condition	IoU	F1 Score
Random weights	0.559	0.711
ImageNet weights	**0.608**	**0.748**
ImageNet weights + frozen encoder	0.601	0.746

Object detection (Approach)





After evaluating various pre-trained models, we found Faster R-CNN to be optimal for Pen-Fudan pedestrian detection. The pedestrian detection model uses ResNet50 FPN V2 serves as the chosen backbone model, leveraging its Feature Pyramid Network (FPN) architecture to capture hierarchical features essential for detecting pedestrians aids in overcoming scale variations present in pedestrian instances within the dataset

Object detection (Approach)

Model Name	Backbone Name	Number of Parameters	Training Time	AP@[0.50:0.95]	AP@[0.50]	AP@[0.75]	AP@[small]	AP@[medium]	AP@[large]
Faster RCNN Model 1	ResNet50 FPN	17844249	25 s	0.784	0.990	0.923	0.525	0.717	0.799
Faster RCNN Model 2	ResNet50 FPN V2	19748121	42.0 s	0.816	**0.991**	**0.943**	**0.75**	**0.724**	**0.827**
Faster RCNN Model 3	MobileNet V3 Large FPN	15982677	11.0 s	0.756	0.975	0.906	0.000	0.648	0.781
Faster RCNN Model 4	MobileNet V3 Large 320 FPN	15982677	10.0 s	0.553	0.904	0.678	0.000	0.227	0.604
FCOS Model 5	ResNet50 FPN	8814688	27.0s	0.731	0.980	0.927	0.501	0.591	0.749
RetinaNet Model 6	ResNet50 FPN	10560087	37.0 s	0.693	0.984	0.883	0.402	0.662	0.708
RetinaNet Model 7	ResNet50 FPN V2	14690903	37.0 s	0.728	0.974	0.859	0.701	0.669	0.738
Faster RCNN Model	MobileNet V2 Large	82,352,981	29.0s	0.366	0.794	0.219	0.000	0.103	0.412

Object detection (Approach)

Modifications or Enhancements:

1. Learning Rate Scheduling: Integrate dynamic learning rate scheduling mechanisms to adaptively fine-tune the learning rate throughout the training process. This implementation facilitates faster convergence and mitigates the risk of overshooting optimal parameter values.

2. Post-Processing Techniques: Implement post processing techniques like Non-Maximum Suppression(3) (NMS) were implemented to refine pedestrian detection results. This step contributed to the reduction of redundant bounding boxes by filtering out redundant bounding boxes with low scores, enhancing the accuracy of localized pedestrian instances.

3. Anchor Box Optimization: Experiment with anchor box sizes and aspect ratios. Customize anchor scales to better match the distribution of object sizes in your dataset

4. Modification of Output Layer and Freezing Layers: Adjust the output layer to match the number of classes in the Pen-Fudan dataset. Freeze some layers of the model, especially the earlier layers (backbone). Freezing is useful when the target dataset is small or similar to the original pre-training dataset. This helps retain features learned from the original dataset.

.

Object detection (Metrics)

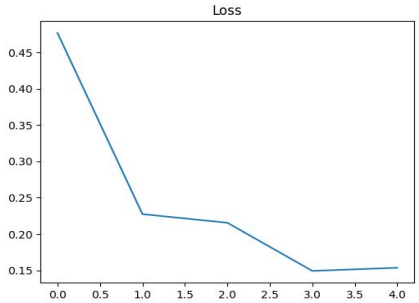
Loss Function Used for Training

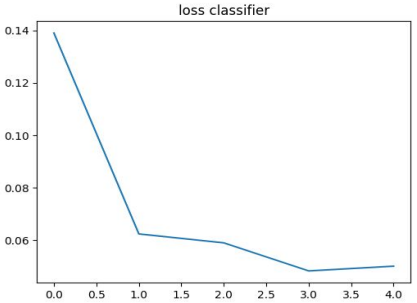
Region Proposal Network (RPN) Loss: The RPN is responsible for proposing candidate object regions.

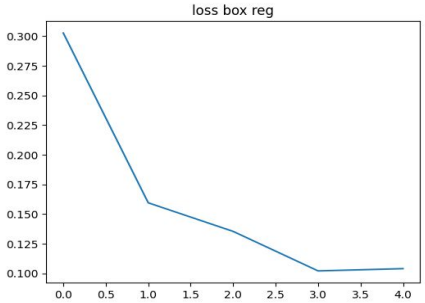
- **Anchor Classification Loss (Binary Cross-Entropy Loss):** Measures how well the RPN classifies anchors as foreground (containing an object) or background.
- **Anchor Regression Loss (Smooth L1 Loss):** Measures the difference between the predicted bounding box coordinates and the ground truth coordinates for positive (foreground) anchors.

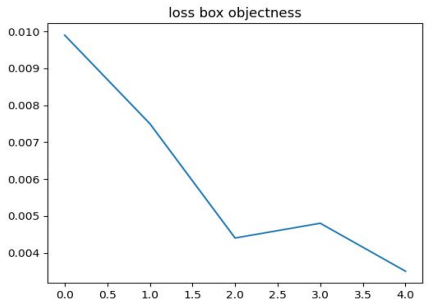
Region-based ROI Heads Loss: After the RPN proposes candidate regions, the RoI (Region of Interest) heads are responsible for refining these proposals and predicting the class of the objects within them.

- **Classification Loss (Cross-Entropy Loss):** Measures how well the model classifies the objects within the proposed RoIs.
- **Regression Loss (Smooth L1 Loss):** Measures the difference between the predicted bounding box coordinates and the ground truth coordinates for the objects within the proposed RoIs.





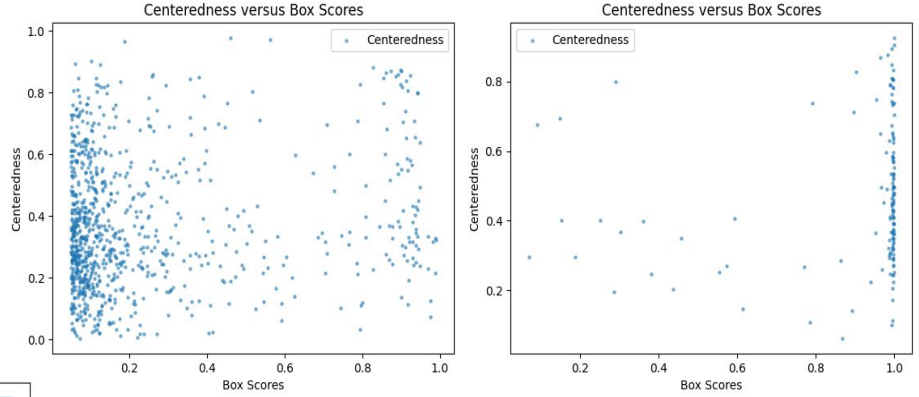




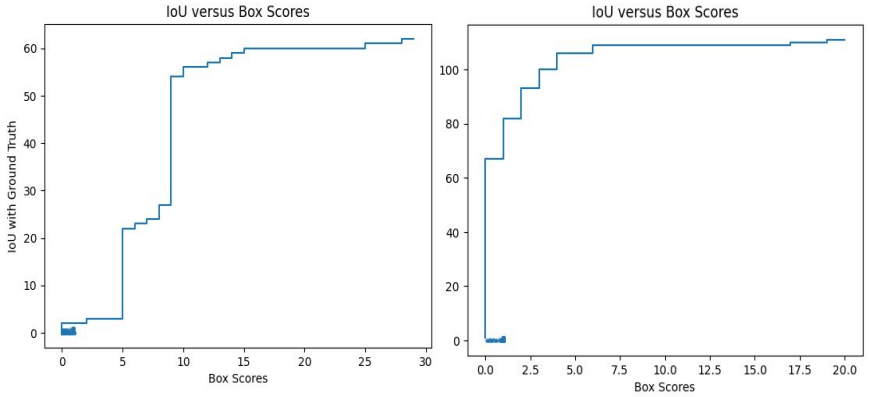
Object detection (Metrics)

$$\text{Center-ness*} = \sqrt{\frac{\min(l,r)}{\max(l,r)} \times \frac{\min(t,b)}{\max(t,b)}}$$

we employed a comprehensive set of evaluation metrics, according to the standards established by the MS COCO Dataset. Our evaluation metrics are IoU (Intersection over Union), Average Precision, Average Recall, and the Precision-Recall curve.



$$IoU = \frac{Area\ of\ Intersection}{Area\ of\ Union}$$



mAP, or mean Average Precision, refers to the average precision calculated across various IoU values, typically ranging from 0.50 to 0.95. This principle holds similarly for Average Recall as well

Object detection (Results)

Model	AP@[IoU=0.75]	AP@small
Simple Faster R-CNN	0.643	0.500
Improved Model	0.943	0.750

This selected model showcased exceptional performance, attaining the highest area under the precision-recall curve and mean average accuracy. Notably, it demonstrated remarkable proficiency in detecting small objects. Additionally, it achieved top-tier scores in classification accuracy and IoU, affirming its efficacy for pedestrian detection tasks in intricate and challenging scenarios.

Metric	Value
AP @[IoU=0.50:0.95 , area=all , maxDets=100]	0.674
AP @[IoU=0.50 , area=all , maxDets=100]	0.960
AP @[IoU=0.75 , area=all , maxDets=100]	0.853
AP @[IoU=0.50:0.95 , area=small , maxDets=100]	0.50
AP @[IoU=0.50:0.95 , area=medium , maxDets=100]	0.155
AP @[IoU=0.50:0.95 , area=large , maxDets=100]	0.696
AR @[IoU=0.50:0.95 , area=large , maxDets=100]	0.738

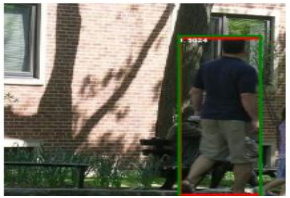
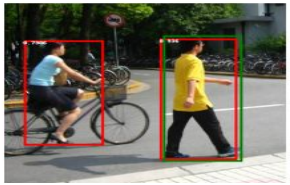
Object detection (Results)

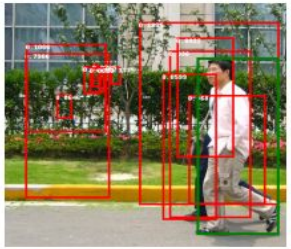
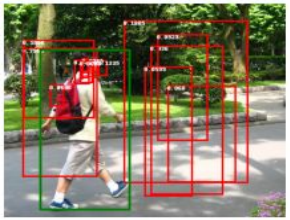
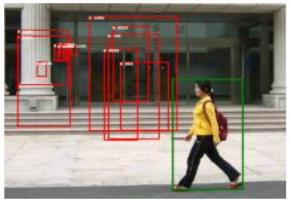
Failure Cases Analysis in Pedestrian Detection

Despite the overall commendable performance of our pedestrian detection model on the Pen-Fudan dataset, it is crucial to delve into instances where the model faces challenges and exhibits suboptimal results. The most frequent failures arise in the following scenarios:

- **Single Detection Recall at Low IoU** (maxDets=1) One notable area of concern arises in scenarios where the model is constrained to output only one detection per object suggests a challenge in accurately localizing objects when limited to a single detection.
- **Small Objects** typically exhibit fewer distinguishable features, making precise detection challenging.



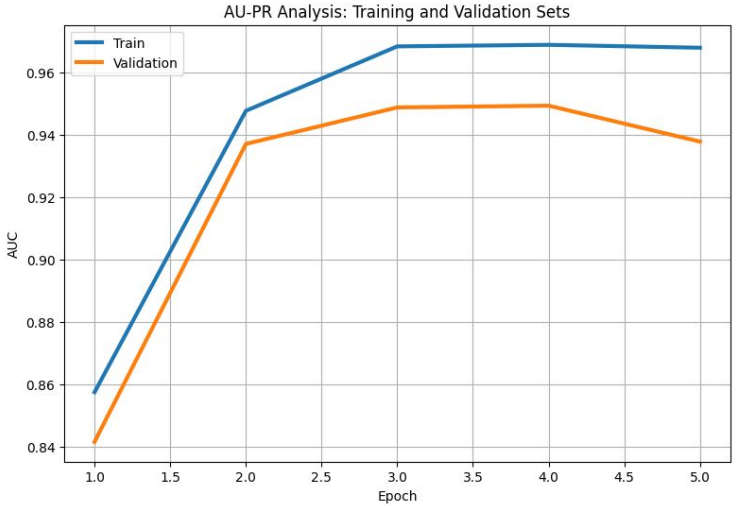




Object detection (Results)

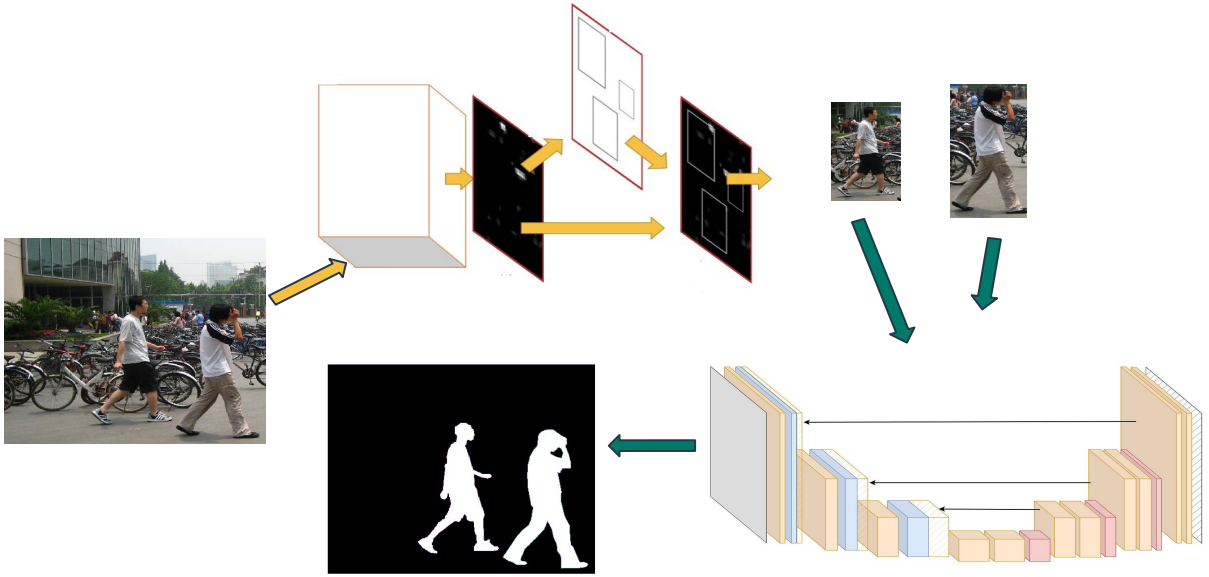
Improvements

- To enhance the model's performance on **small objects**, the following strategies were implemented: **Feature Extraction Enhancements** (utilizing ResNet FPN V2 (5) as our backbone to provide valuable features), **Augmentation Techniques**, and **anchor size optimization**. This allowed us to achieve an Average Precision (AP) at IoU threshold [0.50:0.95] with maxDets=100 for small objects, recorded at 0.750. This metric showcases the model's effectiveness in accurately detecting and localizing pedestrians, particularly in scenarios where objects are relatively small within the image.

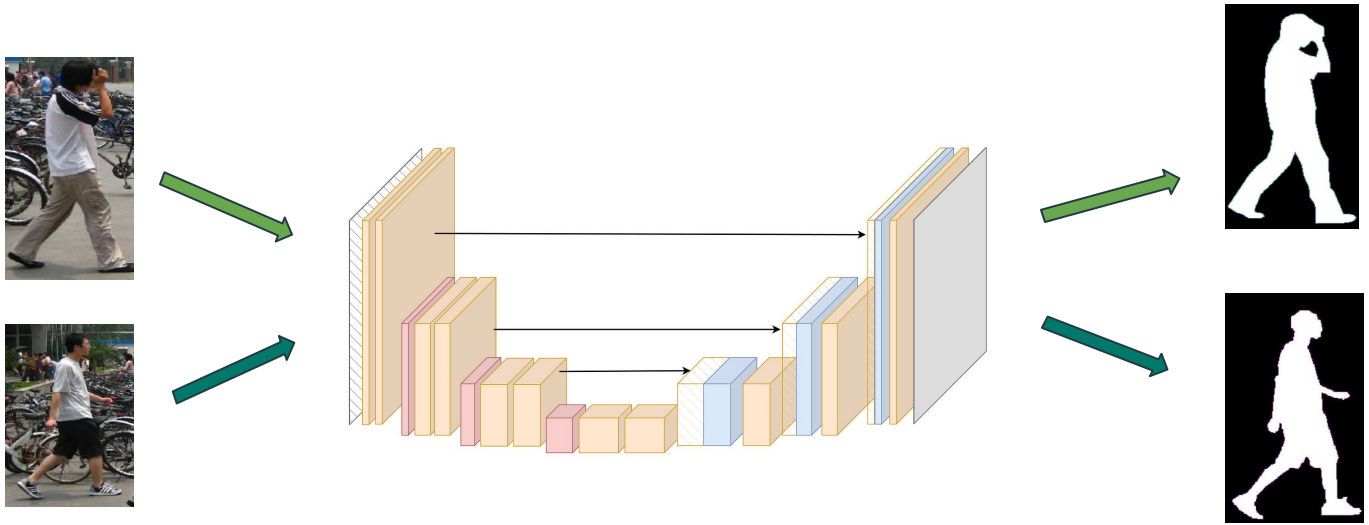
- For **single-object detection**, Post-Processing Refinement, such as **non-maximum suppression** (NMS)(see fig 5), was applied to address overlapping or redundant detections that could contribute to decreased recall. Additionally, **IoU Threshold Optimization** was performed to explore the impact of different IoU thresholds on recall and assess the trade-off with precision.



Detection-Segmentation pipeline

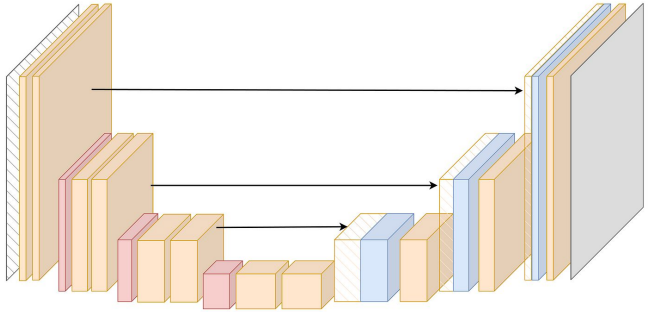


Detection-Segmentation pipeline



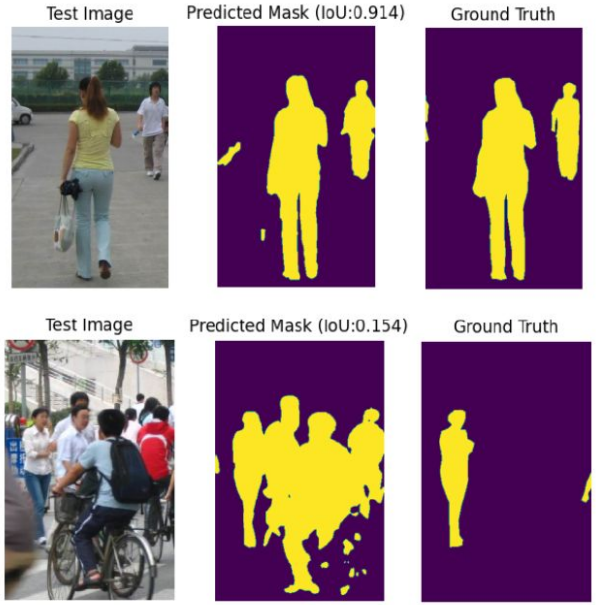
Detection-Segmentation pipeline

288x288

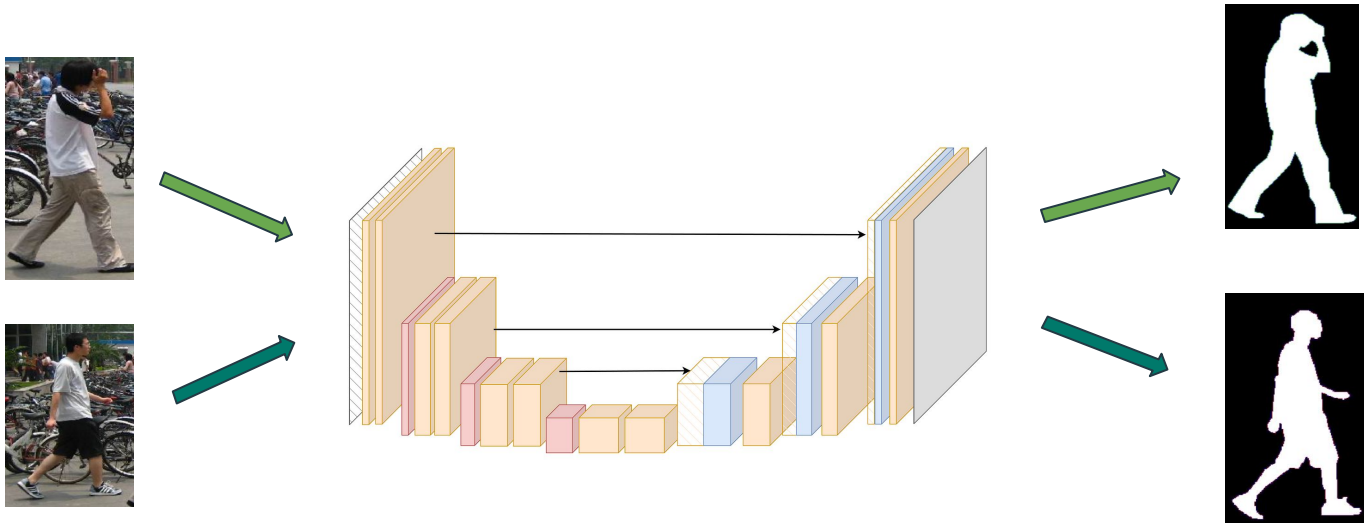


Detection-Segmentation pipeline

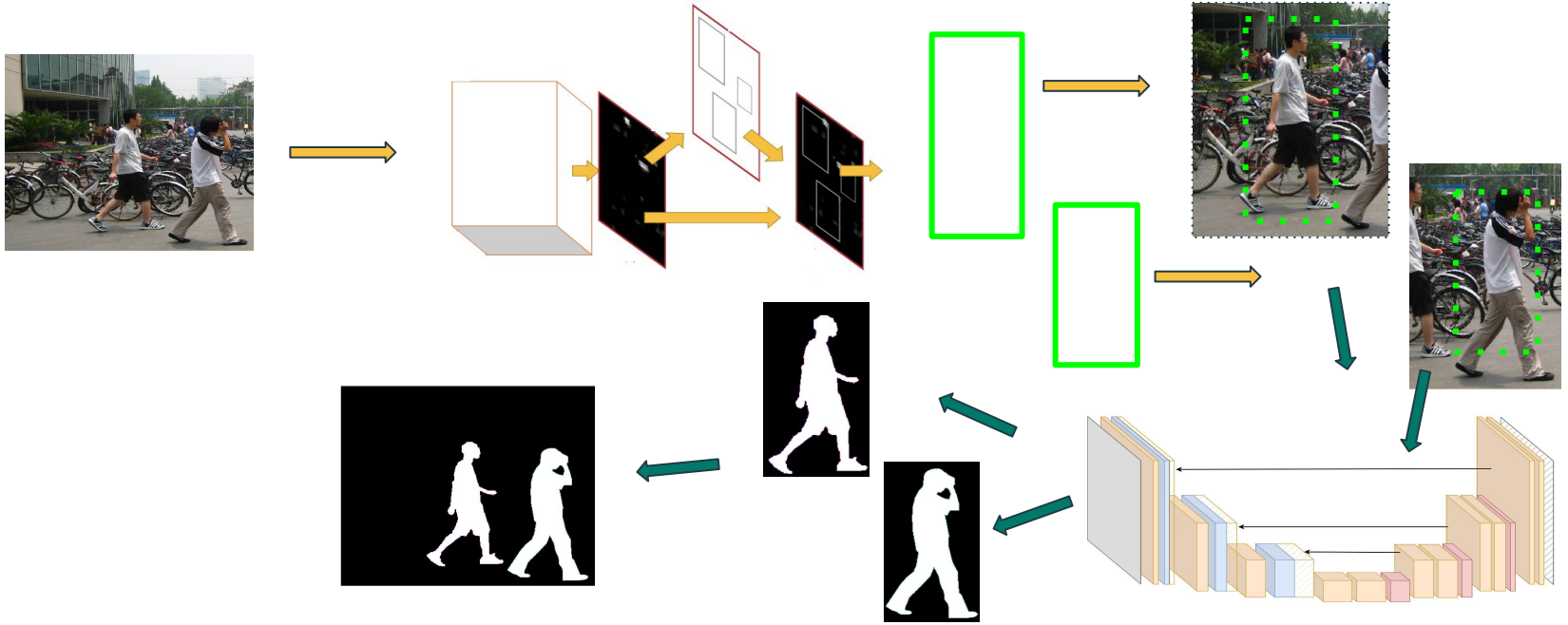
Model	Backbone	IoU	F1 Score
U-Net	ResNet-18	0.626	0.745
U-Net	ResNet-34	0.607	0.731
U-Net++	ResNet-18	**0.637**	**0.756**
U-Net++	ResNet-34	0.631	0.751



Detection-Segmentation pipeline

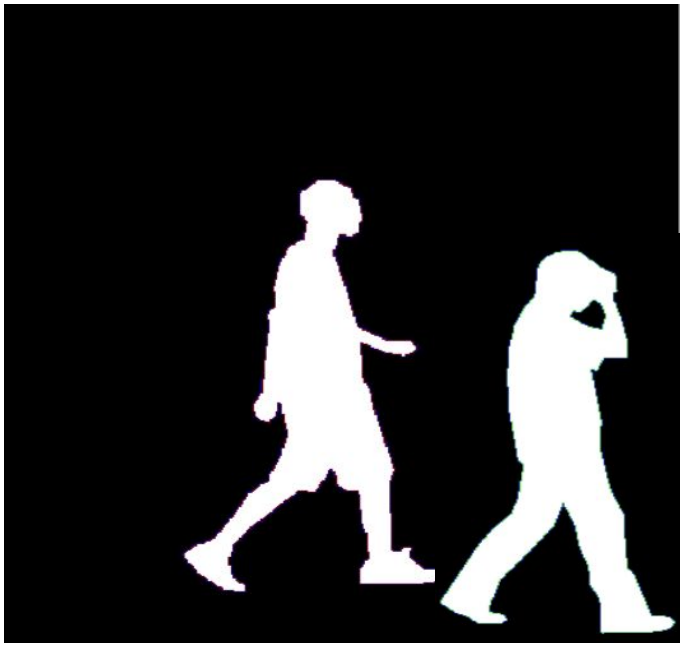


Detection-Segmentation pipeline

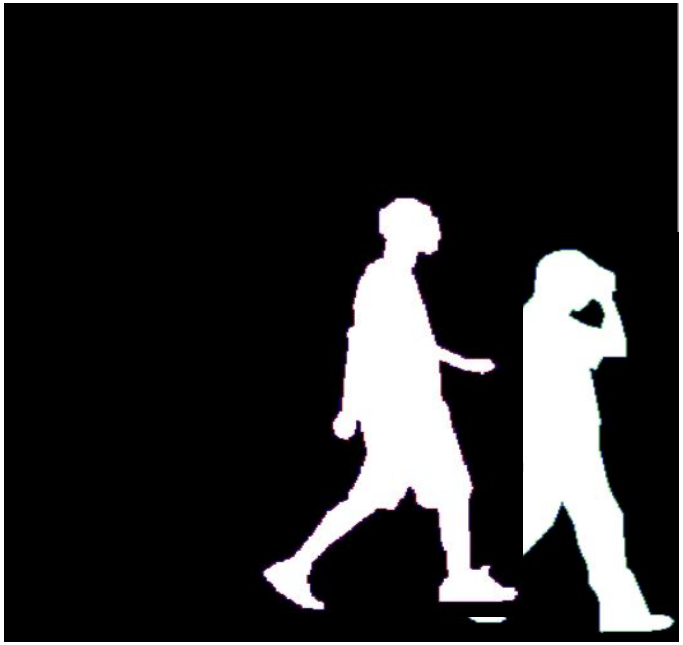


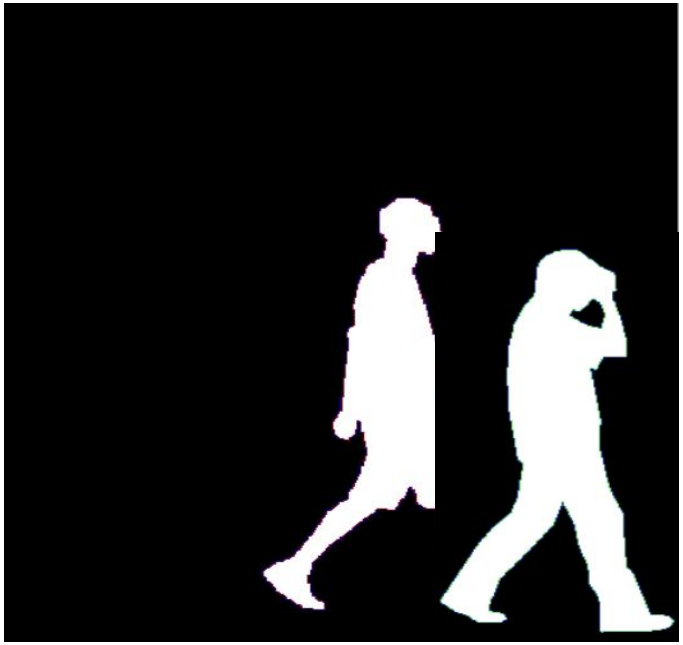
Mask stitching

Mask stitching

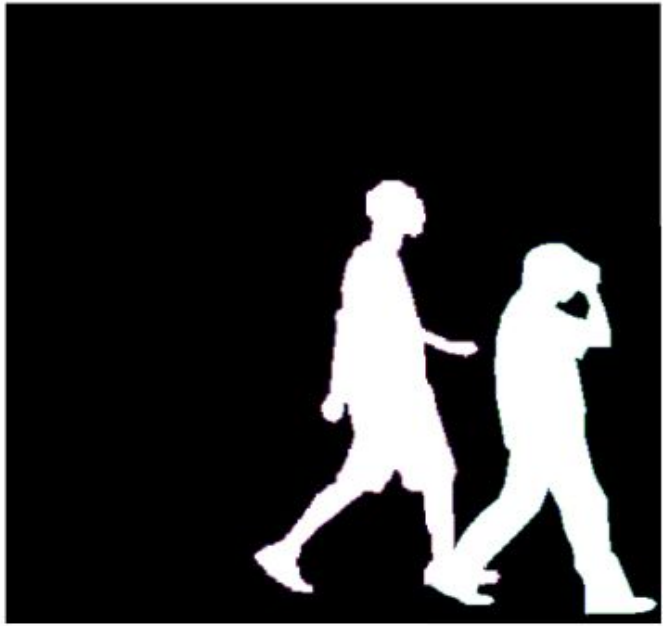


Mask stitching

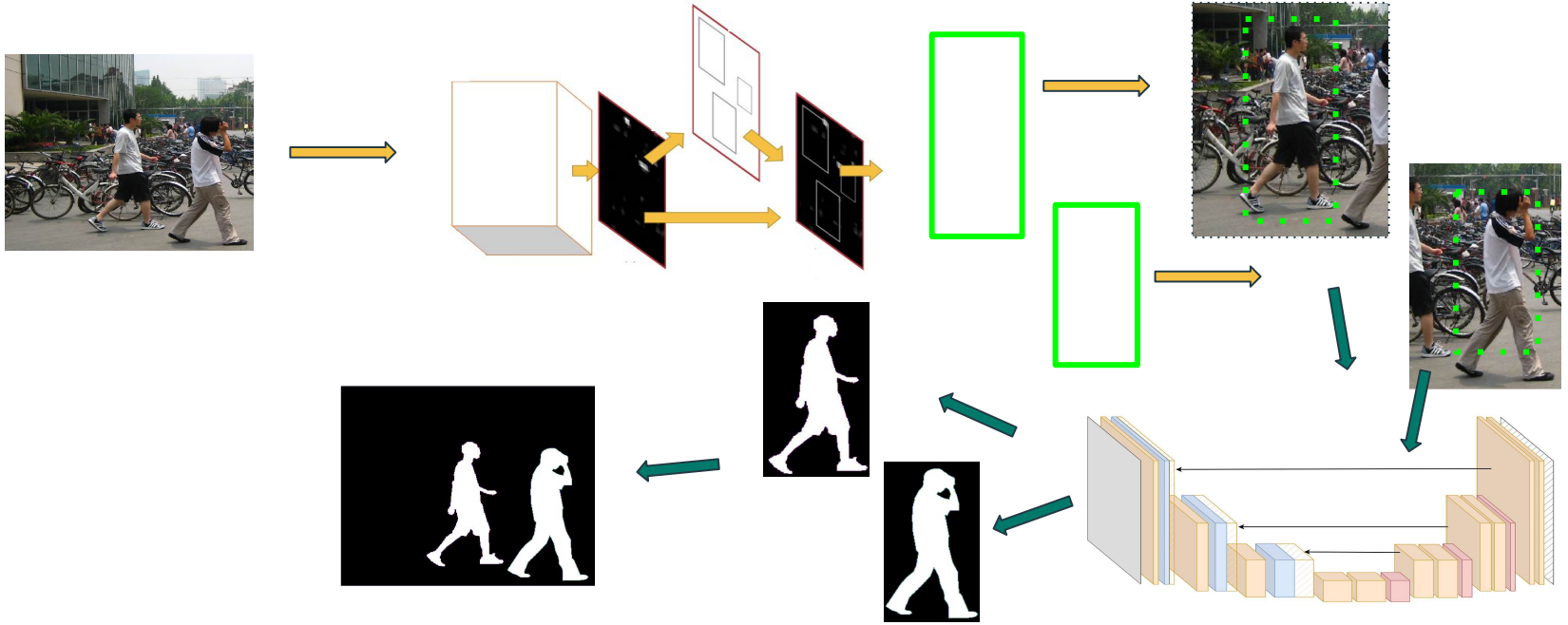




Mask stitching

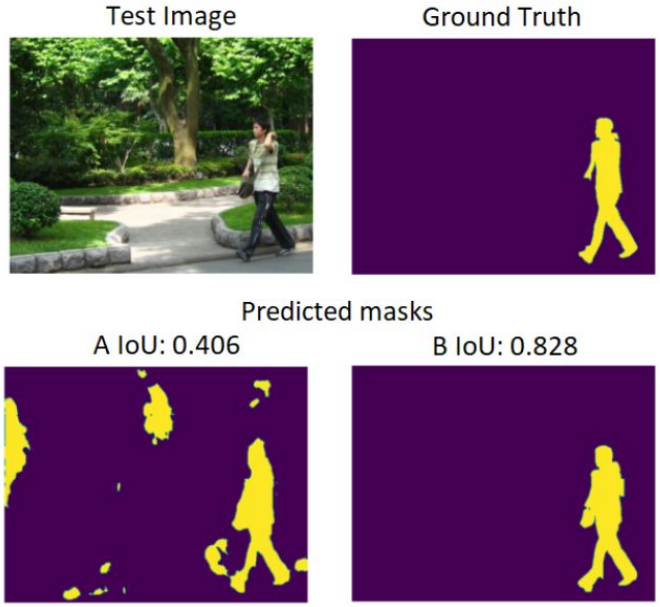


Detection-Segmentation pipeline



Detection-Segmentation pipeline

IoU	F1 Score
0.629	0.765



Conclusions and future work

1. For simple semantic segmentation smaller U-Net model with Resnet-18 backbone performed better
2. Segmentation pipeline powered by an object detection stage achieved slightly better quantitative results then just segmentation

Future work on limitations:

1. Conduct a more extensive hyperparameter search to find optimal configurations for the existing model.
2. Data augmentation (flipping, rotating, color altering and scaling)
3. Experiment with different models (FCN, LinkNet, SegNet)
4. Using another dataset for more reliable quantitative results

THANK YOU
FOR YOUR ATTENTION

Sharare Zolghadr, Donatas Vaiciukevičius, Kateryna Skurativska

January 2024, UNIPD