The Integration of Thermal and Visual Images for Human Detection Using Deep Learning

GFM M.Sc. Mid-term Defence

Qiao REN
First Supervisor: Dr. Siavash Hosseinyalamdary
Second Supervisor: Dr. Francesco Nex
Nov 21, 2018

Content

1. Problem Statement
2. Research Identification
3. Methodology
4. My Progress
5. Intermediate Result
6. What to do next

1. Problem Statement

- Main information sources: Visual Images and Thermal Images
- Drawbacks of visual and thermal images
- Sensor fusion: RGBD (RGBT)

(a) (b) (c) (d)

Content

1. Problem Statement
2. Research Identification
3. Methodology
4. My Progress
5. Intermediate Result
6. What to do next

2. Research Identification
- Research Objective

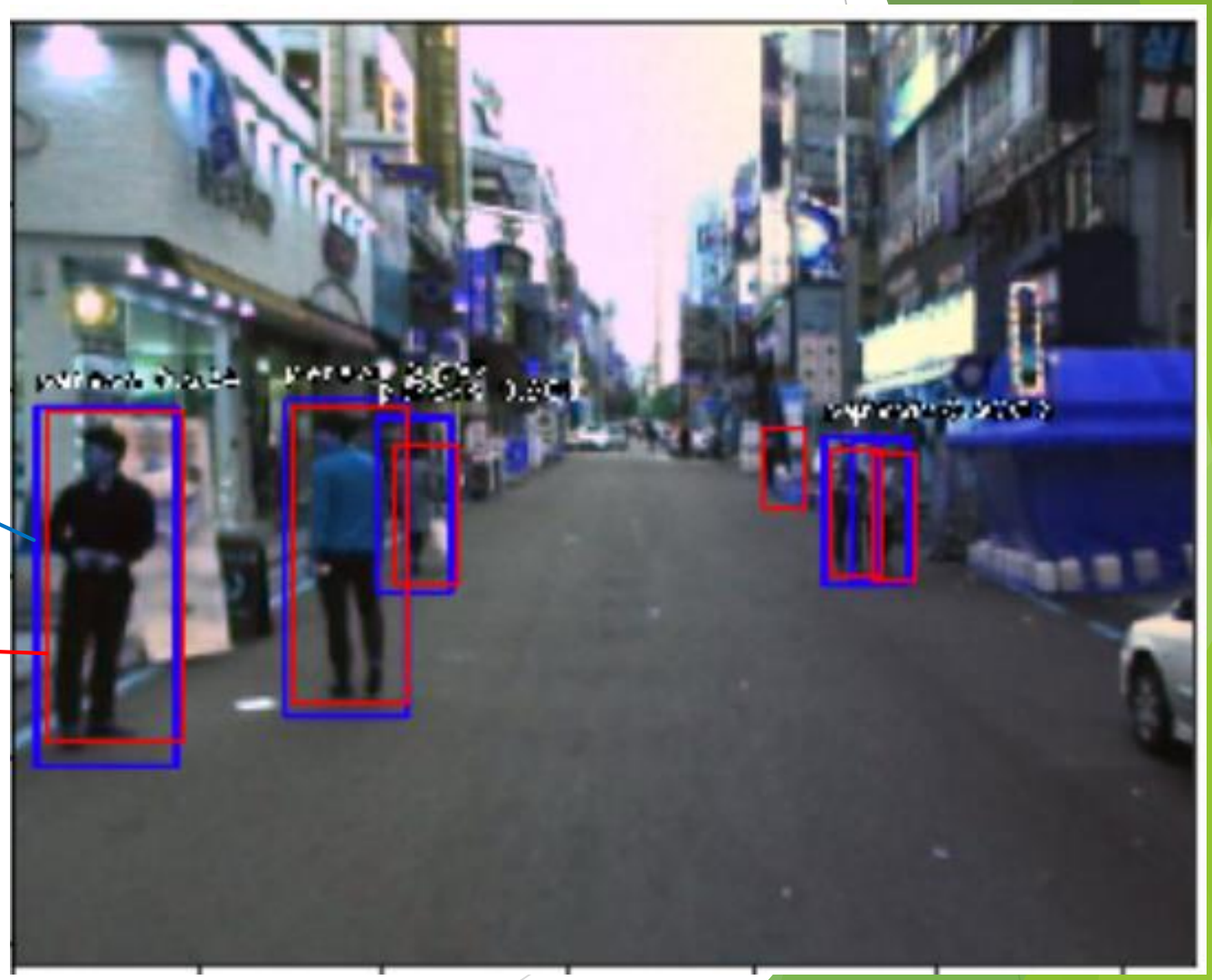
- Test: different sensor fusion architectures
- Approach: convolutional neural network (CNN)
- Applied in: human detection
- Input: thermal and visual images
- Provide: the most accurate architecture

2. Research Identification

- Output: bounding box

Blue bounding box: prediction

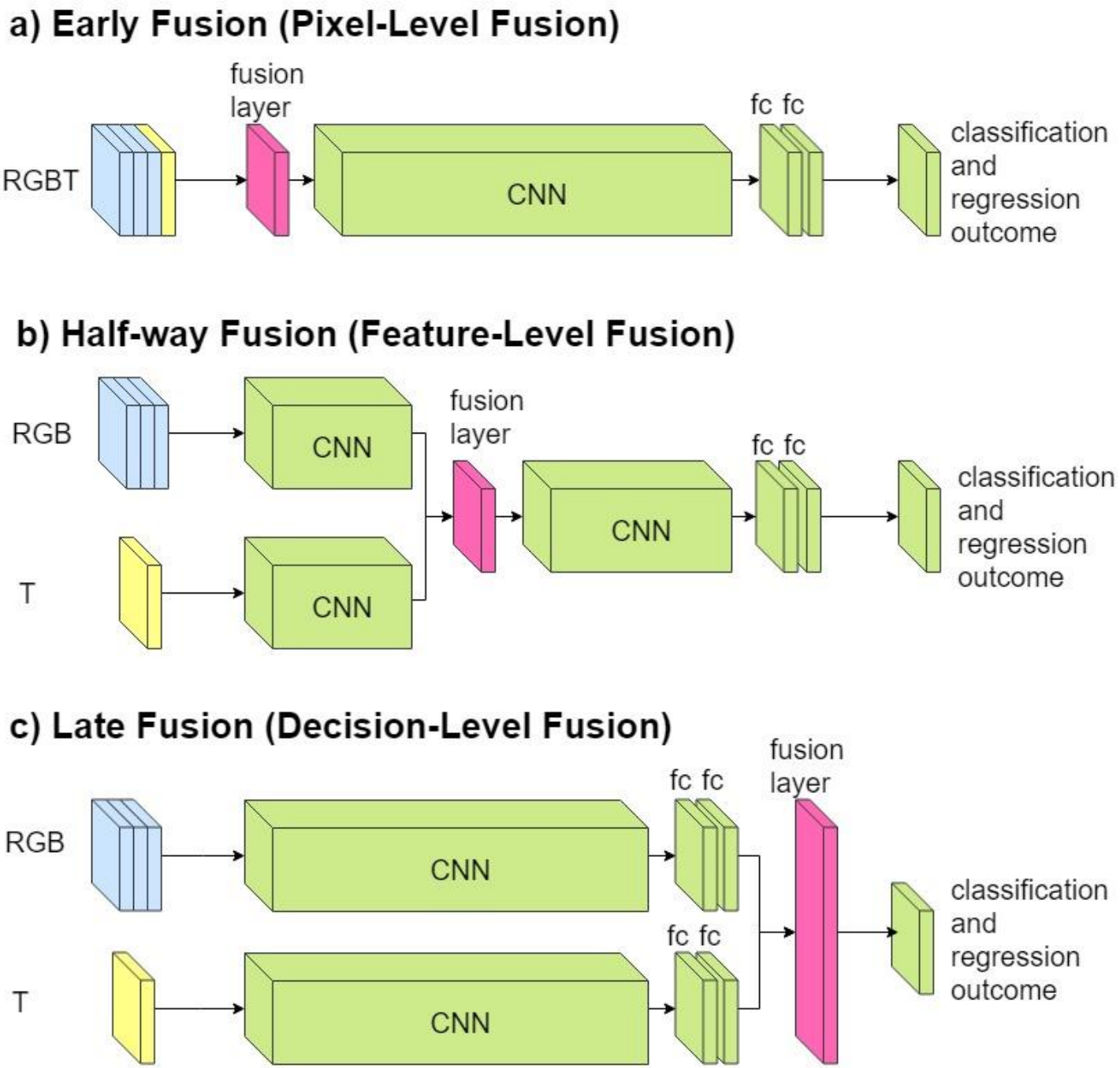
Red bounding box: annotation (ground truth)



Content

1. Problem Statement
2. Research Identification
3. Methodology
4. My Progress
5. Intermediate Result
6. What to do next

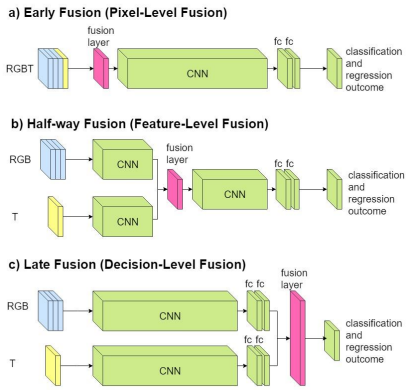
3. Methodology



legend

- a channel in visual images
- a channel in thermal images
- a fusion layer (concatenate layer)
- a convolutional layer, a fully connected layer (fc), a classification output layer or a regression output layer

3. Methodology-Retinanet



choose backbone and depth

load architecture of specified backbone

resnet

vgg

mobilenet

densenet

load weight for the specified backbone

weight randomly initialized

weight pretrained by other networks

ImageNet

kitti

coco

coco+kitti

costruct retinanet model

load selected 3 feature stage (c3,c4,c5) from backbone

create feature pyramid network layers (Focal Loss)

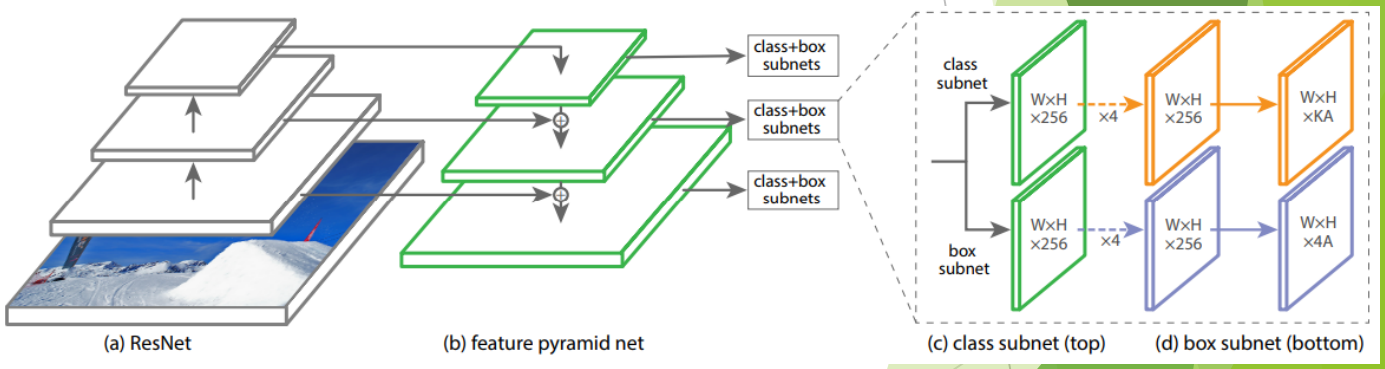
for all pyramid levels, build submodels:

classification model

regression model

Why choose retinanet?

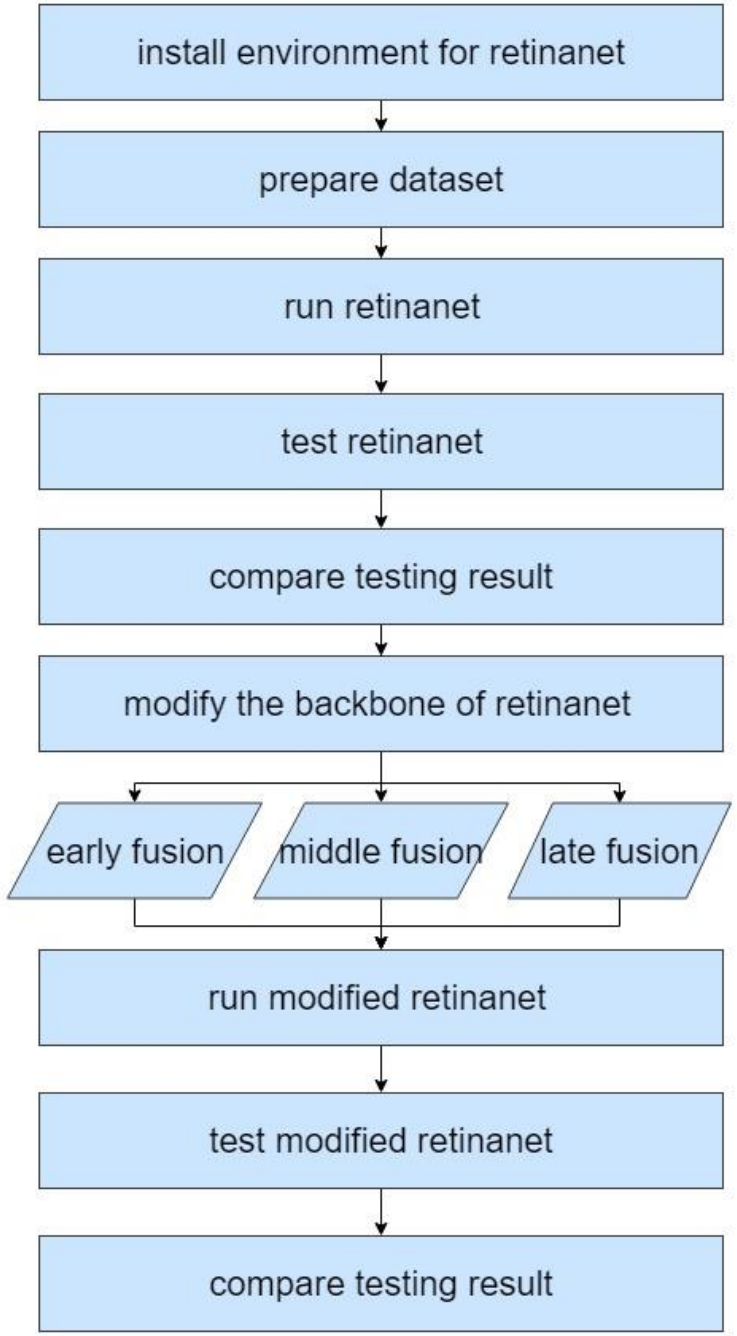
- solve two problems:
  - Problem 1: class imbalance
  - Solution: Focal Loss
  - Problem 2: high resolution maps (earlier layers) have low-level features
  - Solution: feature pyramid network
- Speed vs Accuracy on Coco
  - surpass yolo, SSD, DSSD, R-FCN (Lin,T., Goyal P., Girshick,R, 2018)



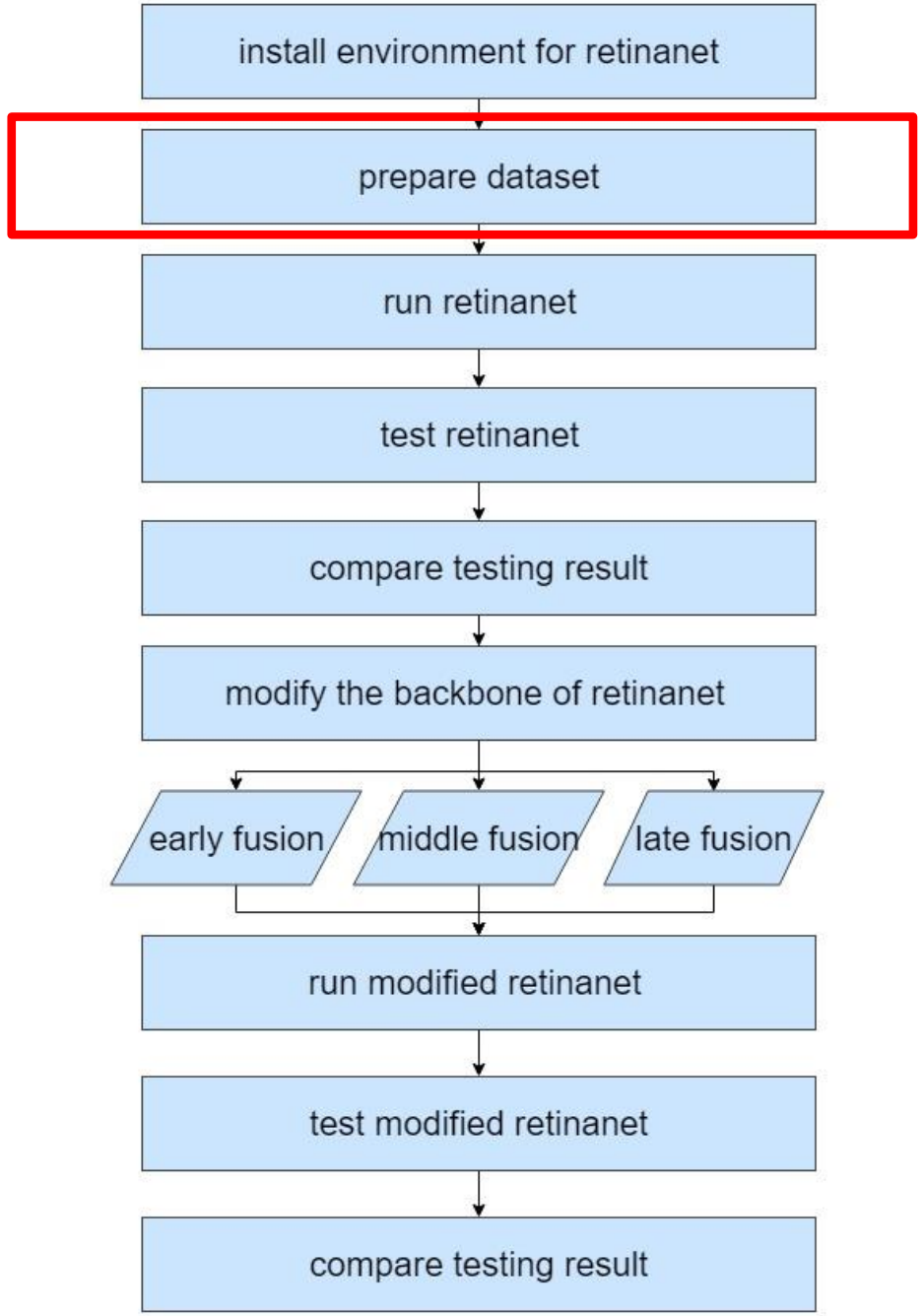
Content

1. Problem Statement
2. Research Identification
3. Methodology
4. My Progress
5. Intermediate Result
6. What to do next

Workflow


install environment for retinanet
prepare dataset
run retinanet
test retinanet
compare testing result
modify the backbone of retinanet
early fusion
middle fusion
late fusion
run modified retinanet
test modified retinanet
compare testing result

Workflow


install environment for retinanet
prepare dataset
run retinanet
test retinanet
compare testing result
modify the backbone of retinanet
early fusion
middle fusion
late fusion
run modified retinanet
test modified retinanet
compare testing result

Data Description

- KAIST (Korea Advanced Institute of Science and Technology)
- Image size: 512 pixels * 640 pixels
- Total: 95k colour-thermal pairs
- Total: 1182 annotated pedestrians

Train 50k
Set 00 / Day / Campus / 17,498 images
Set 01 / Day / Road / 8,035 images
Set 02 / Day / Downtown / 7,866 images
Set 03 / Night / Campus / 6,668 images
Set 04 / Night / Road / 7,200 images
Set 05 / Night / Downtown / 2,920 images

Validation: will be extracted

Test 45k
Set 06 / Day / Campus / 12,988 images
Set 07 / Day / Road / 8,141 images
Set 08 / Day / Downtown / 8,050 images
Set 09 / Night / Campus / 3,500 images
Set 10 / Night / Road / 8,902 images
Set 11 / Night / Downtown / 3,560 images

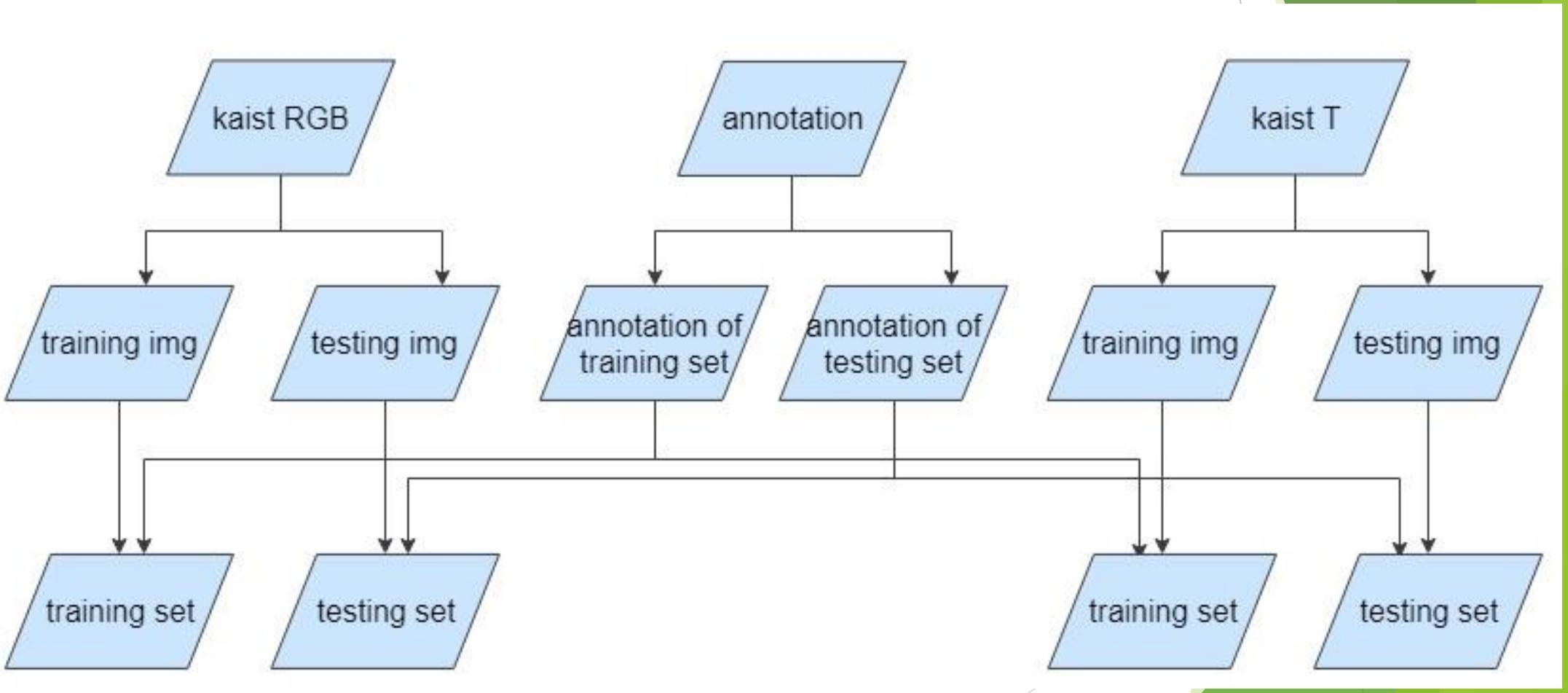
Improved annotation for test: 2250 images

set06-11 1/20 in each set

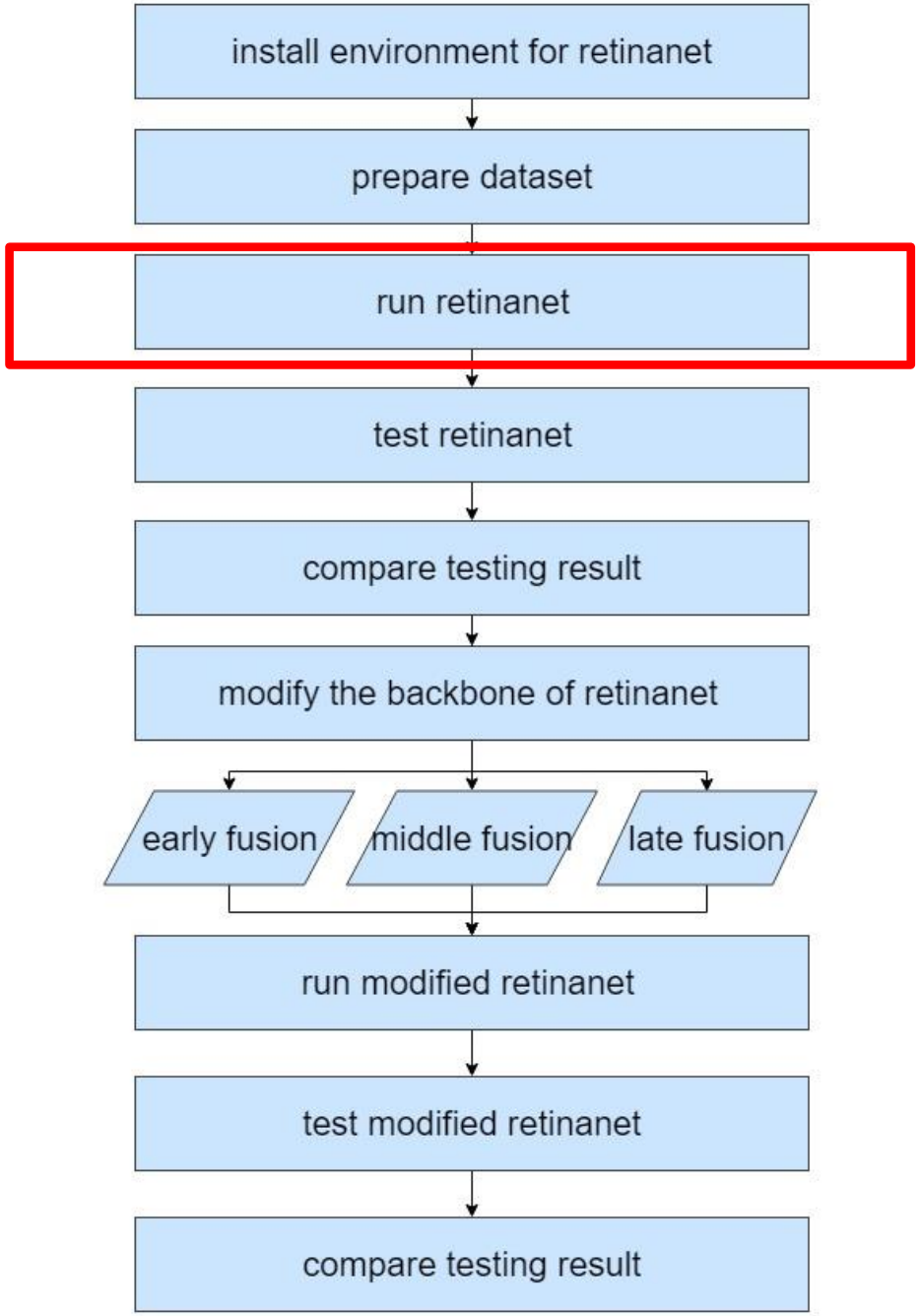
(a)

(b)

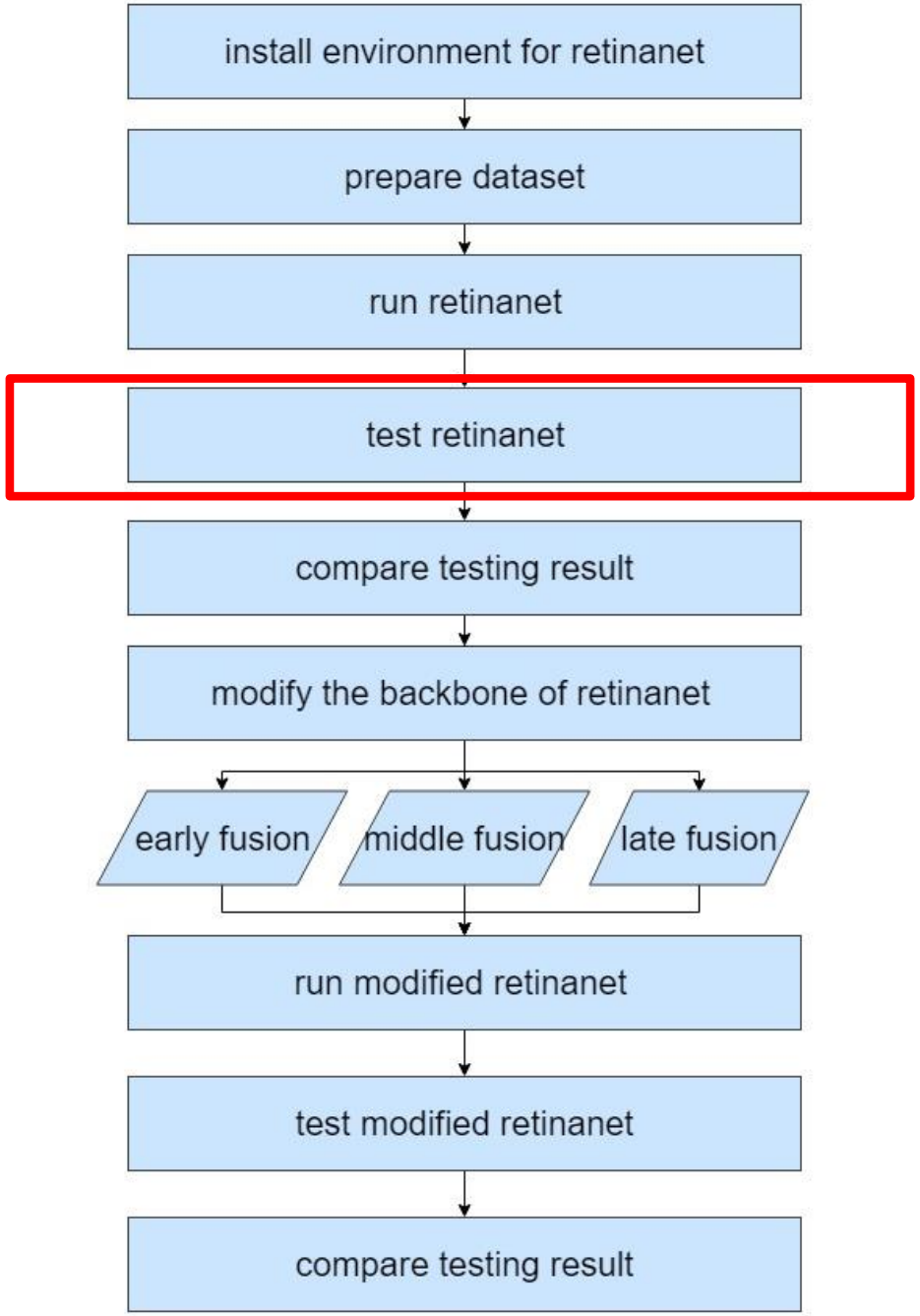
Prepare dataset



Workflow

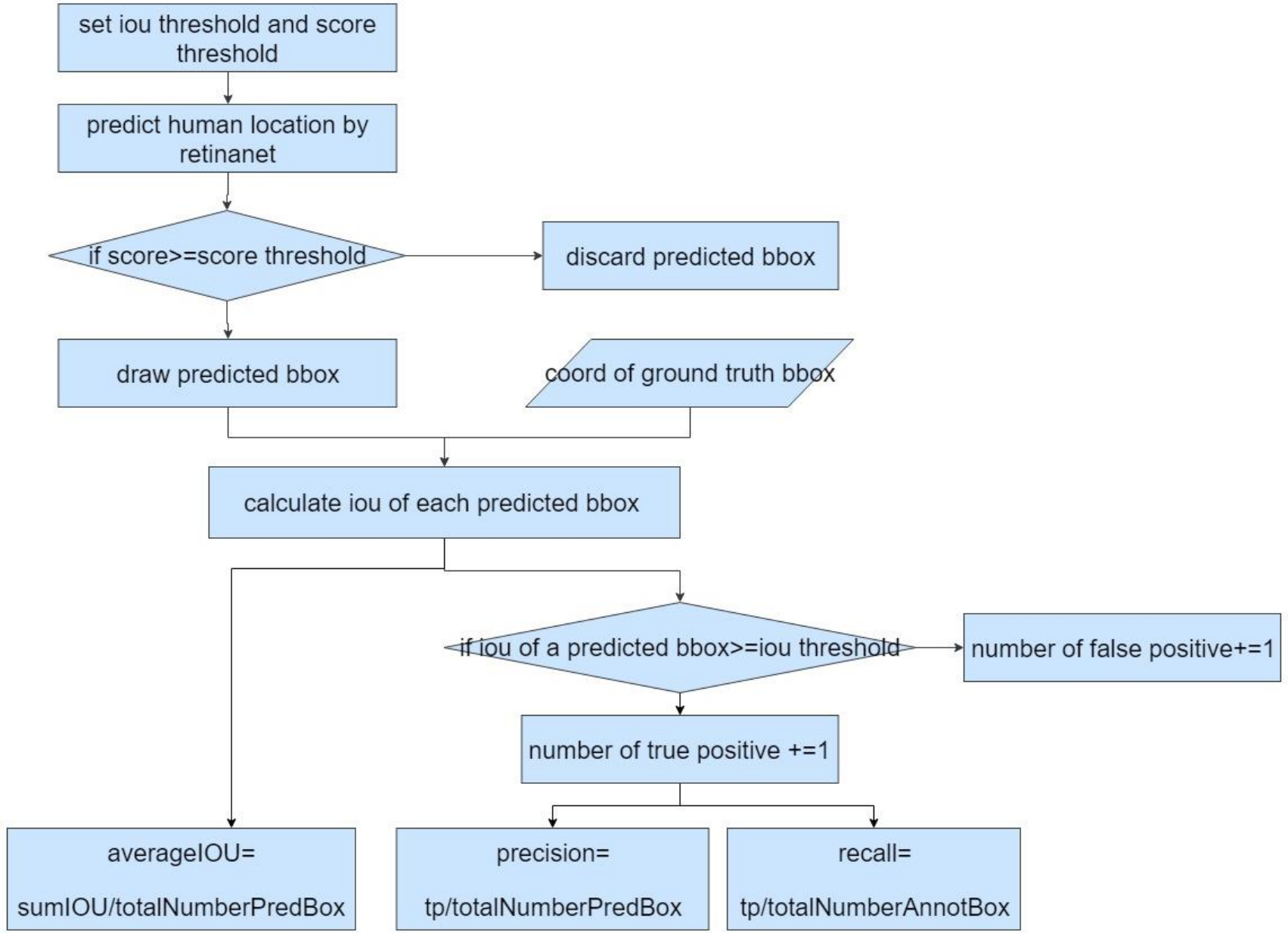

install environment for retinanet
prepare dataset
run retinanet
test retinanet
compare testing result
modify the backbone of retinanet
early fusion
middle fusion
late fusion
run modified retinanet
test modified retinanet
compare testing result

Workflow

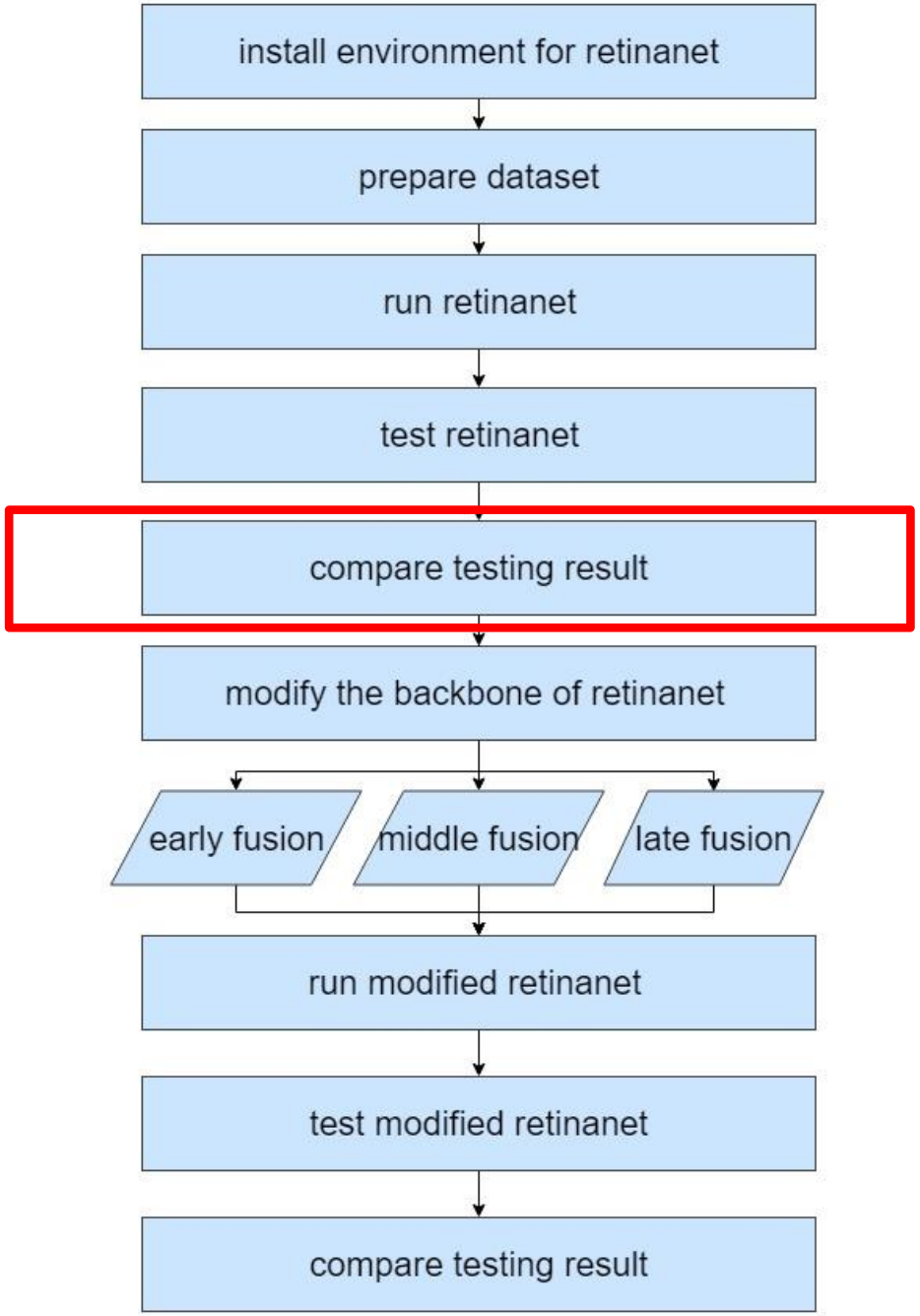

install environment for retinanet
prepare dataset
run retinanet
test retinanet
compare testing result
modify the backbone of retinanet
early fusion
middle fusion
late fusion
run modified retinanet
test modified retinanet
compare testing result

Test Retinanet

- Maunal test
  - Average IOU
  - Precision
  - recall
- Evaluate.py



Workflow

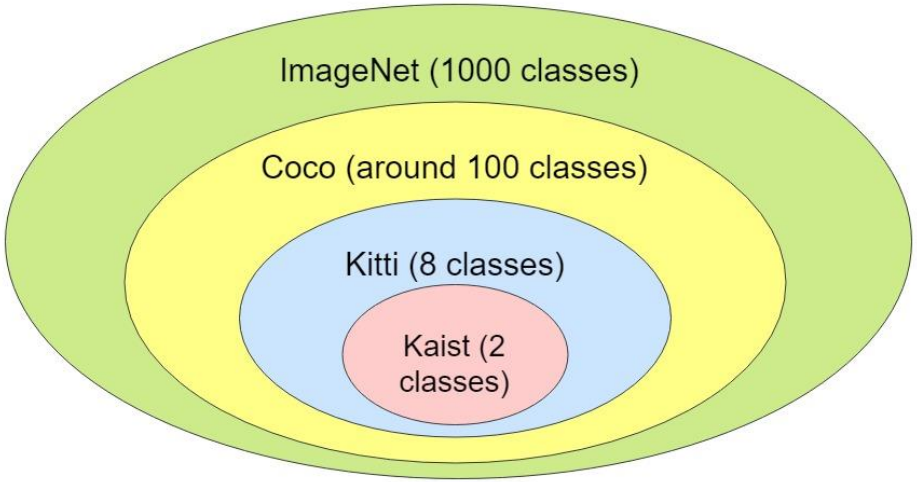

install environment for retinanet
prepare dataset
run retinanet
test retinanet
compare testing result
modify the backbone of retinanet
early fusion
middle fusion
late fusion
run modified retinanet
test modified retinanet
compare testing result

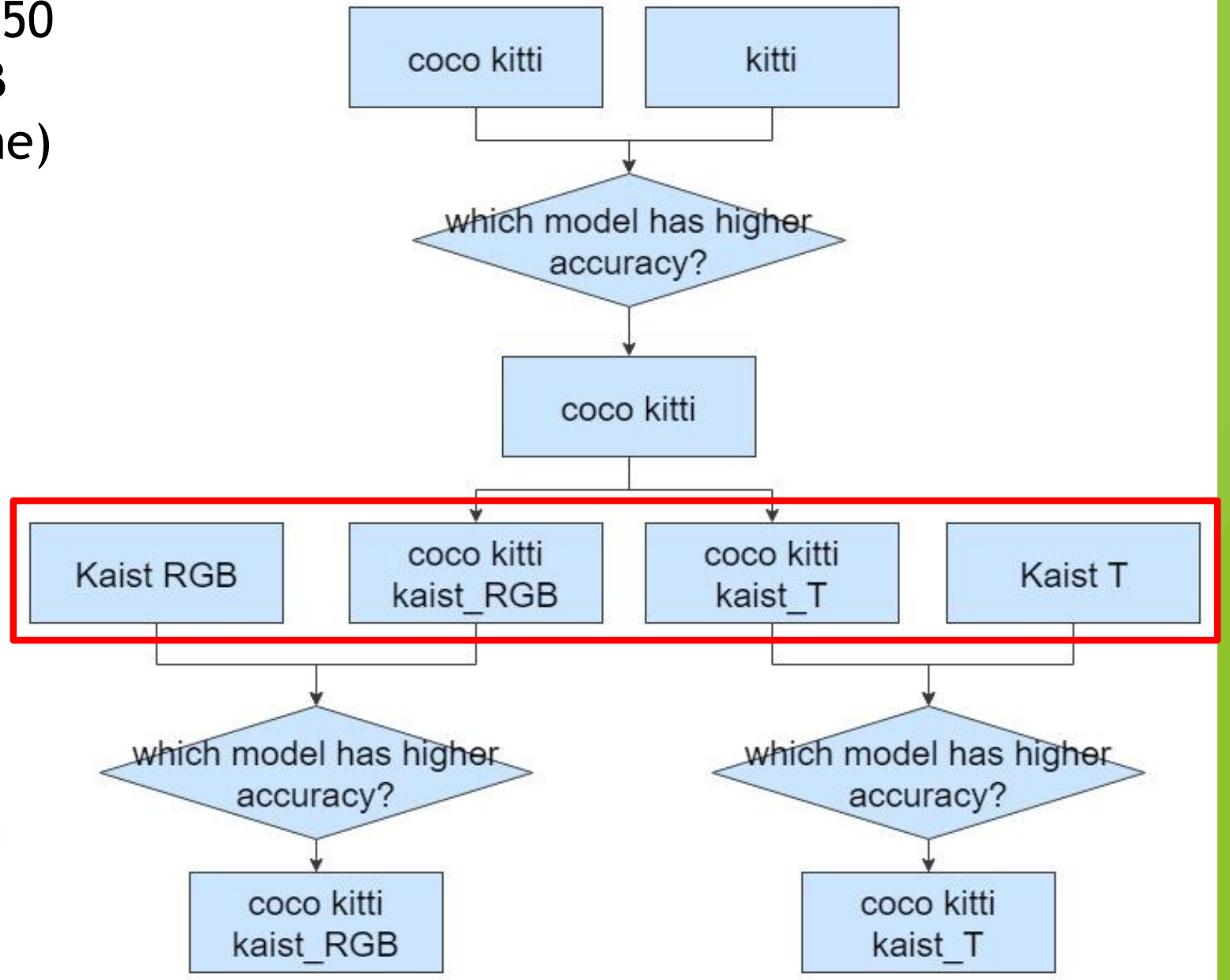
Content

1. Problem Statement
2. Research Identification
3. Methodology
4. My Progress
5. Intermediate Result
6. What to do next

Backbone of all models:Resnet 50
ImageNet (default)+KaistT/RGB
Coco+Kitti+KaistT/RGB (finetune)

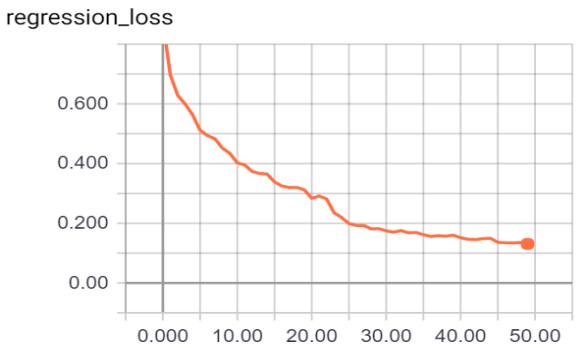
	cocokitti	kitti
Pedestrian with average precision	0.44	0.34
Cyclist with average precision	0.46	0.39



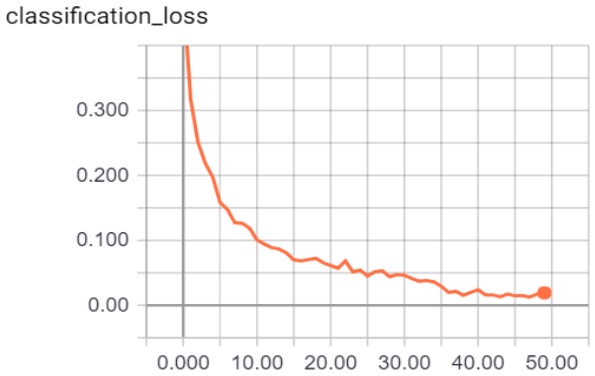


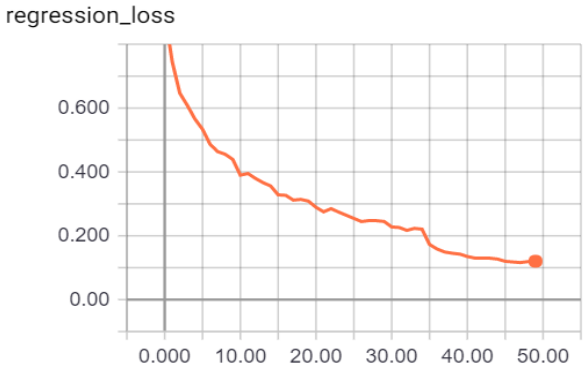
Classification Loss and Regression Loss

KaistT

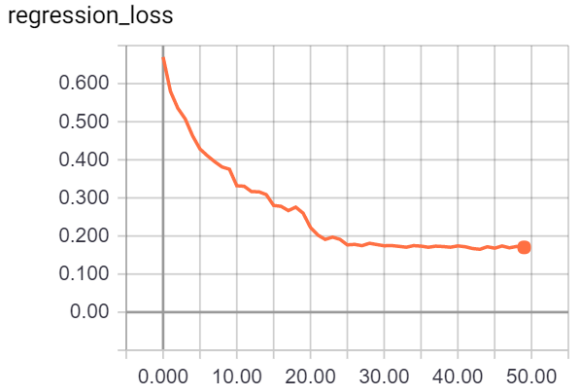


KaistRGB

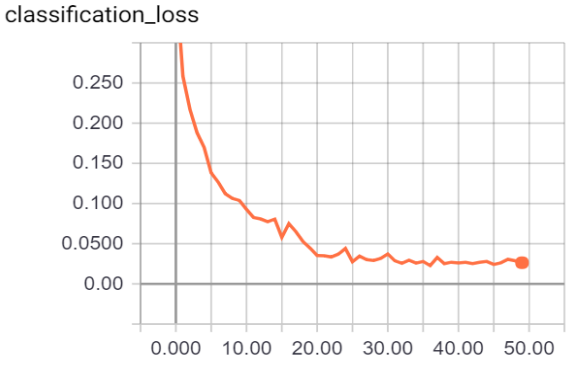




CocoKittiKaistT



CocoKittiKaistRGB

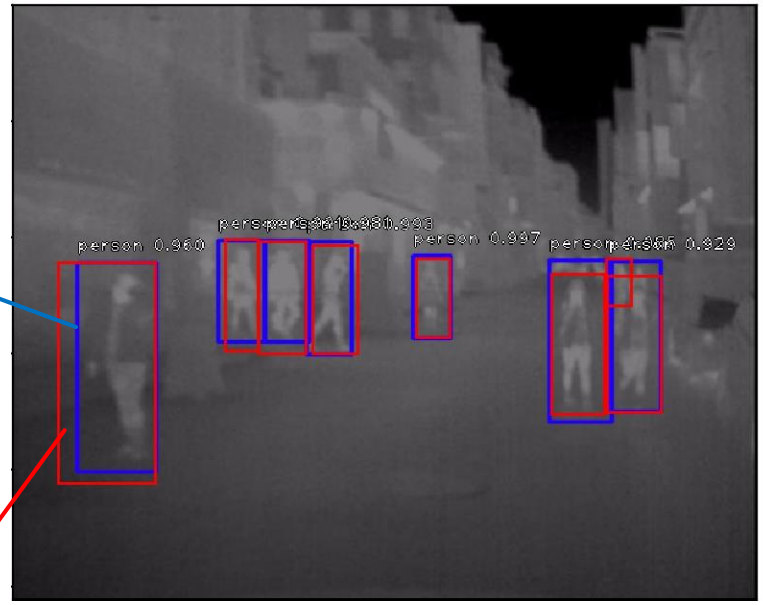


Will include validation loss

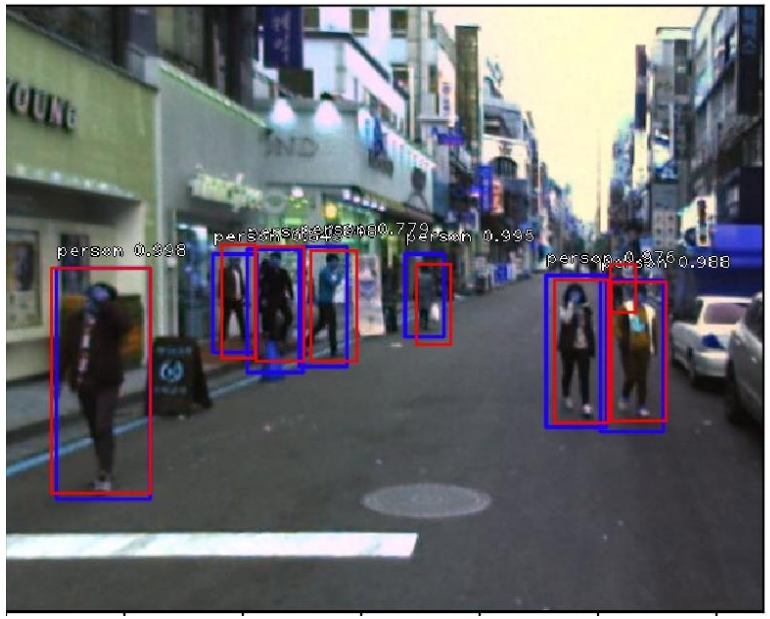
Blue bounding box: prediction

Red bounding box: annotation (ground truth)

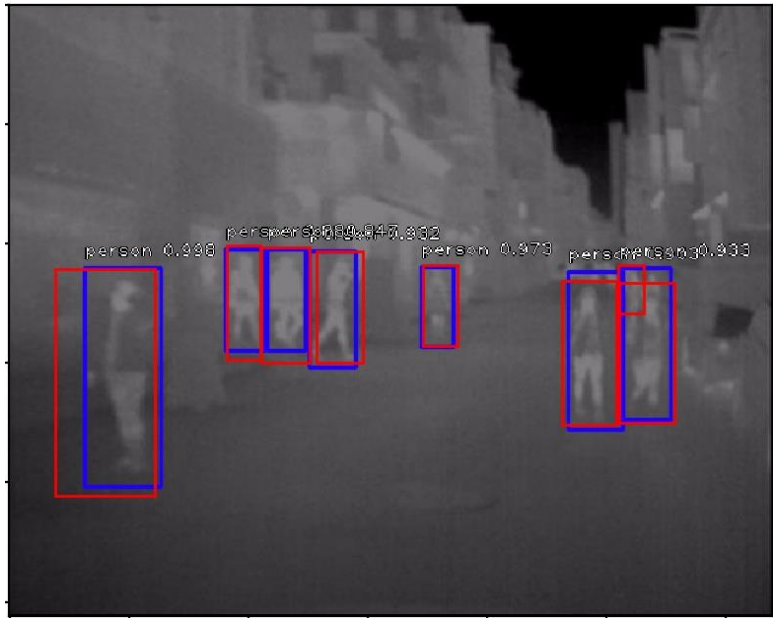
KaistT



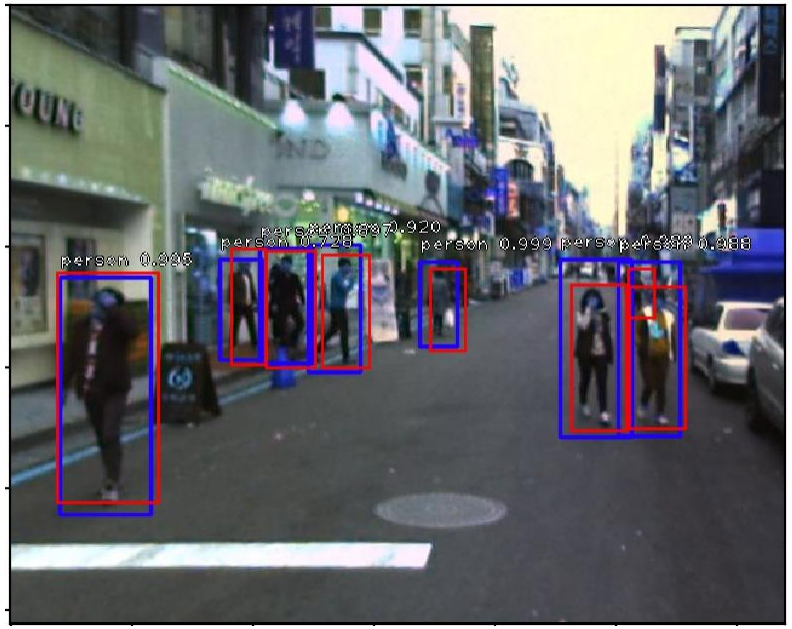
KaistRGB



CocoKittiKaistT

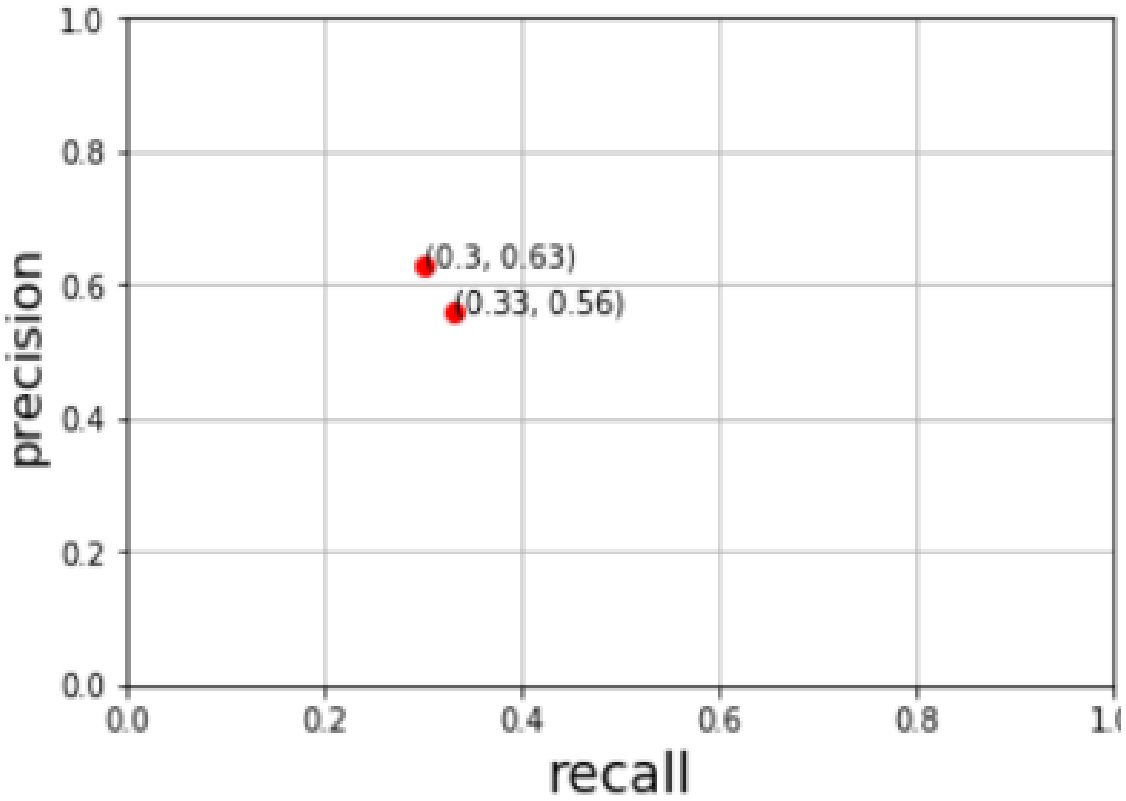


CocoKittiKaistRGB

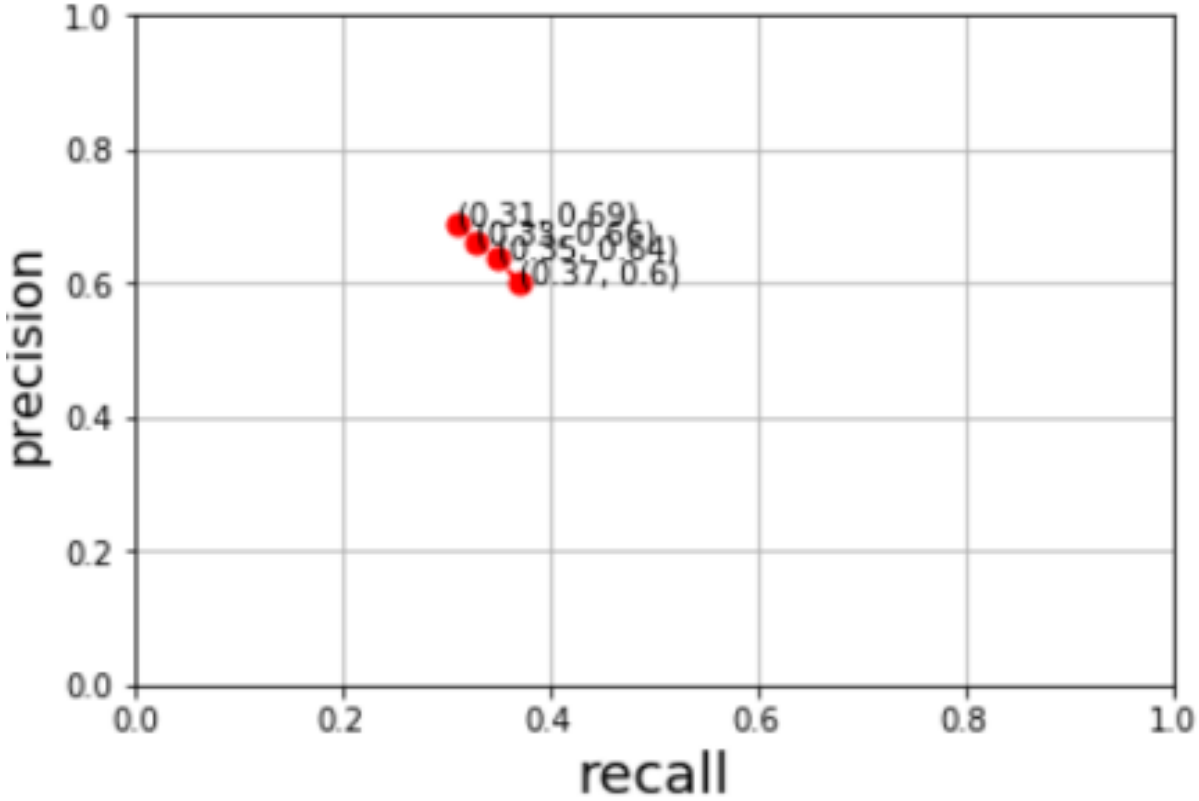


Precision-Recall Graph when iou_threshold=0.7

KaistT



CocoKittiKaistT

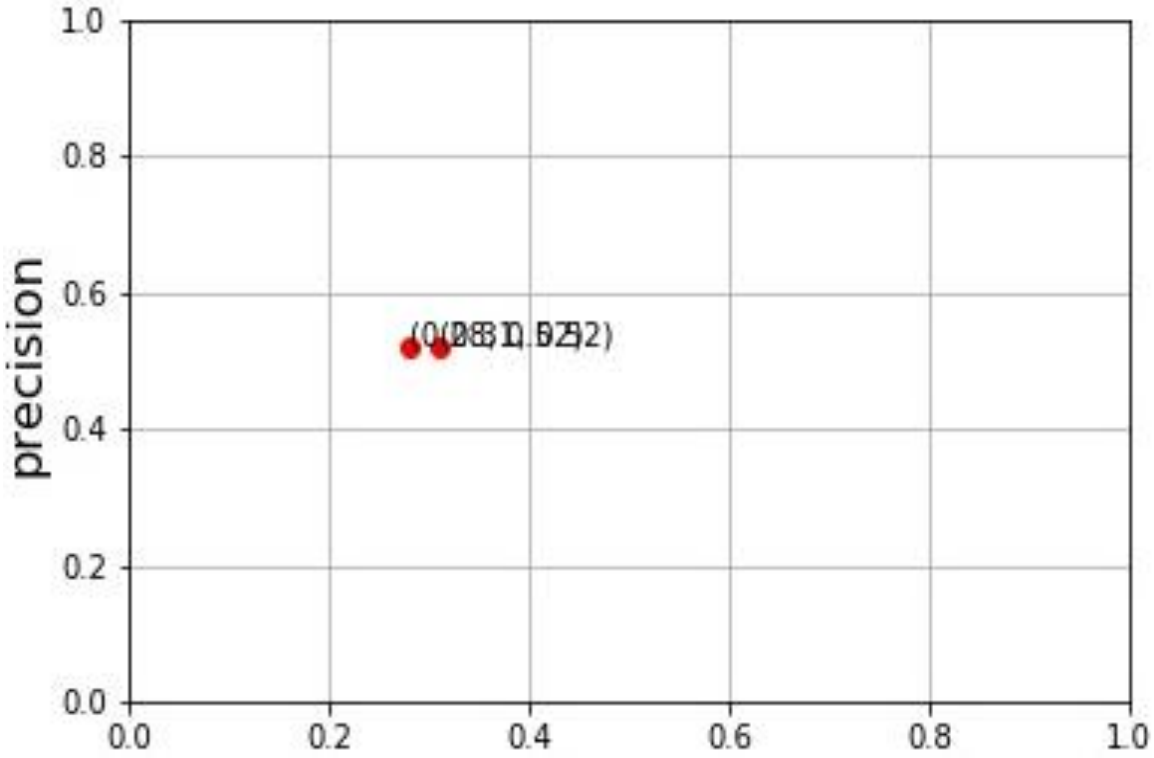


Score>=0.5 recall:0.33 precision: 0.56
Score>=0.6 recall: precision:
Score>=0.7 recall: precision:
Score>=0.8 recall:0.30 precision: 0.63

Score>=0.5 recall:0.37 precision: 0.60
Score>=0.6 recall:0.35 precision: 0.64
Score>=0.7 recall:0.33 precision: 0.66
Score>=0.8 recall:0.31 precision: 0.69

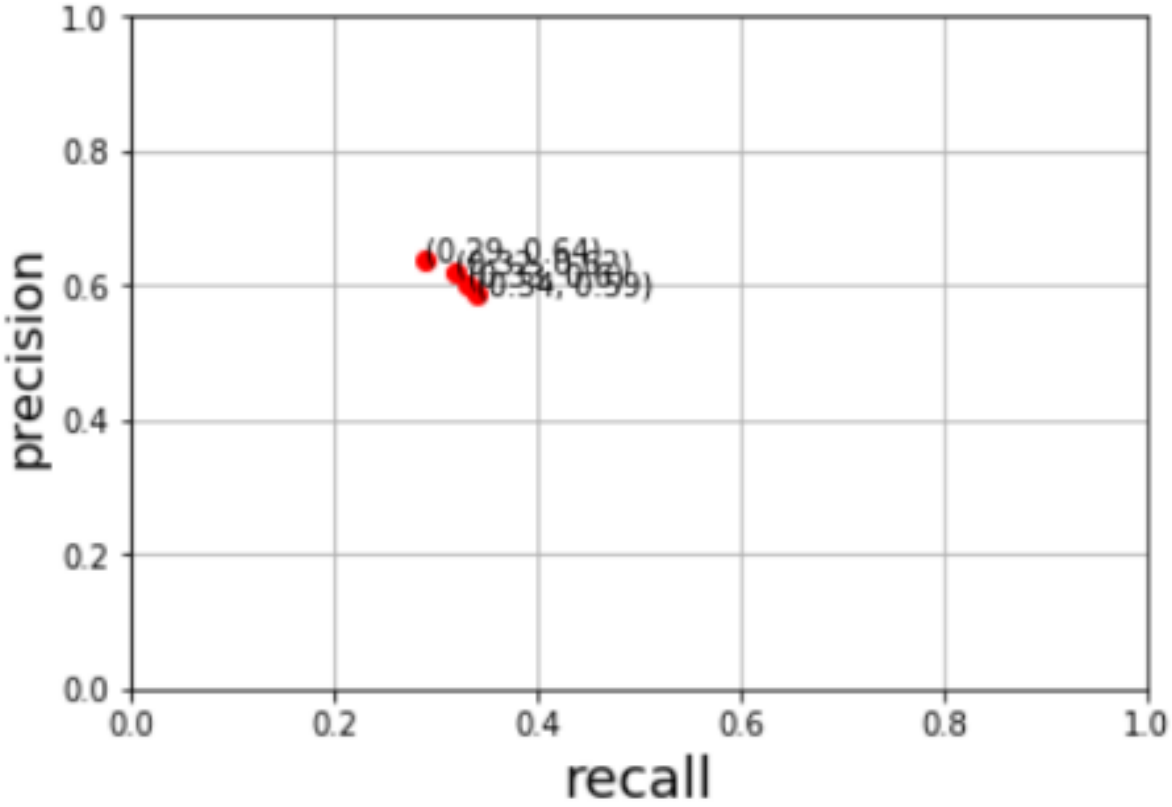
Precision-Recall Graph when iou_threshold=0.7

KaistRGB



Score>=0.5 recall:0.31 precision: 0.52
Score>=0.6 recall: precision:
Score>=0.7 recall: precision:
Score>=0.8 recall:0.28 precision: 0.52

CocoKittiKaistRGB



Score>=0.5 recall:0.34 precision: 0.59
Score>=0.6 recall:0.33 precision: 0.60
Score>=0.7 recall:0.32 precision: 0.62
Score>=0.8 recall:0.29 precision: 0.64

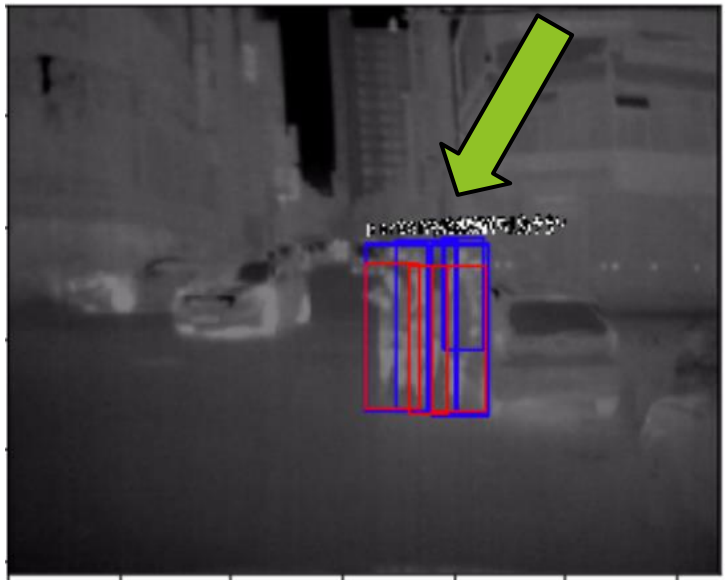
Average IOU when IOU threshold=0.7

	kaistT	kaistRGB	cocokittikaistT	cocokittikaistRGB
score threshold=0.5	0.693	0.662	0.704	0.692
score threshold=0.8	0.724	0.690	0.739	0.715

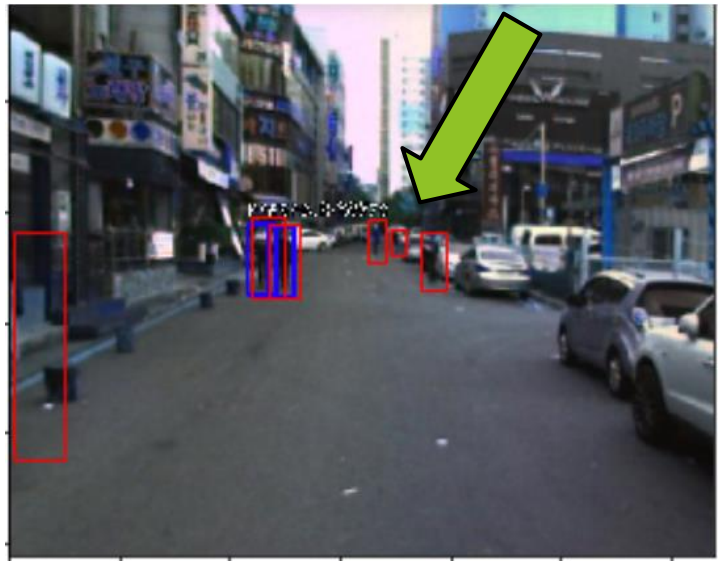
Train: set 0-5
Test: set 08/v000 with improved annotation

Half person

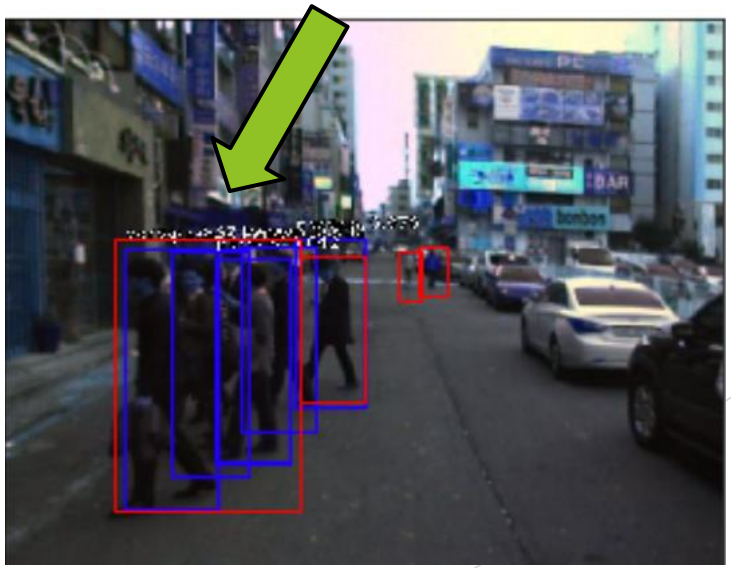
occluded person
(able to detect but iou<threshold)



Persons are too small



Annotation is imperfect



Content

1. Problem Statement
2. Research Identification
3. Methodology
4. My Progress
5. Intermediate Result
6. What to do next

What to do next?

choose backbone and depth

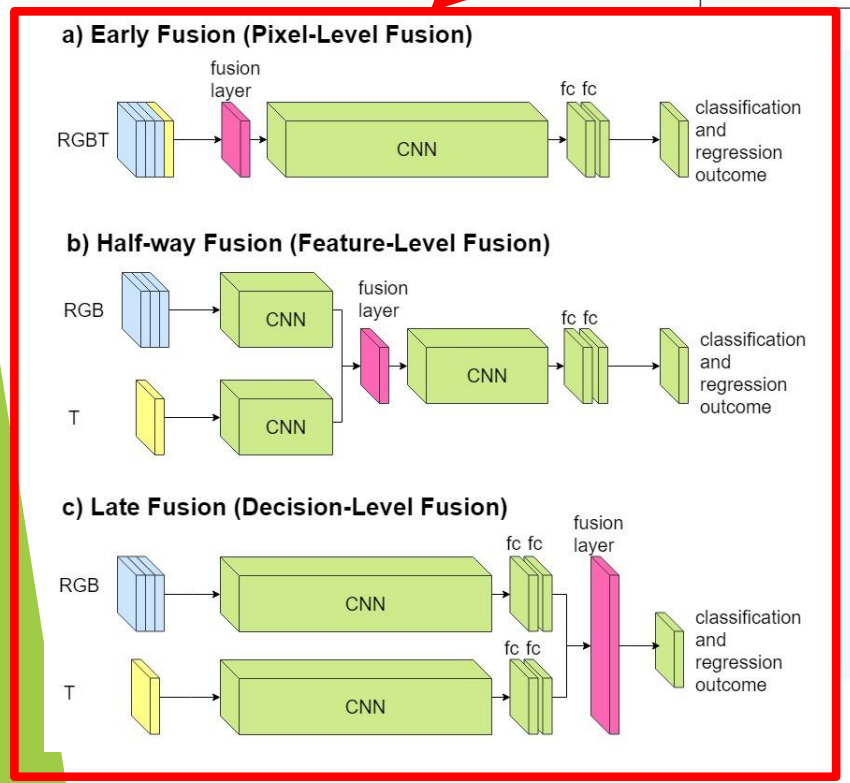
load architecture of specified backbone

load weight for the specified backbone

resnet | vgg | mobilenet | densenet

weight randomly initialized | weight pretrained by other networks

ImageNet | kitti | coco | coco+kitti



costruct retinanet model

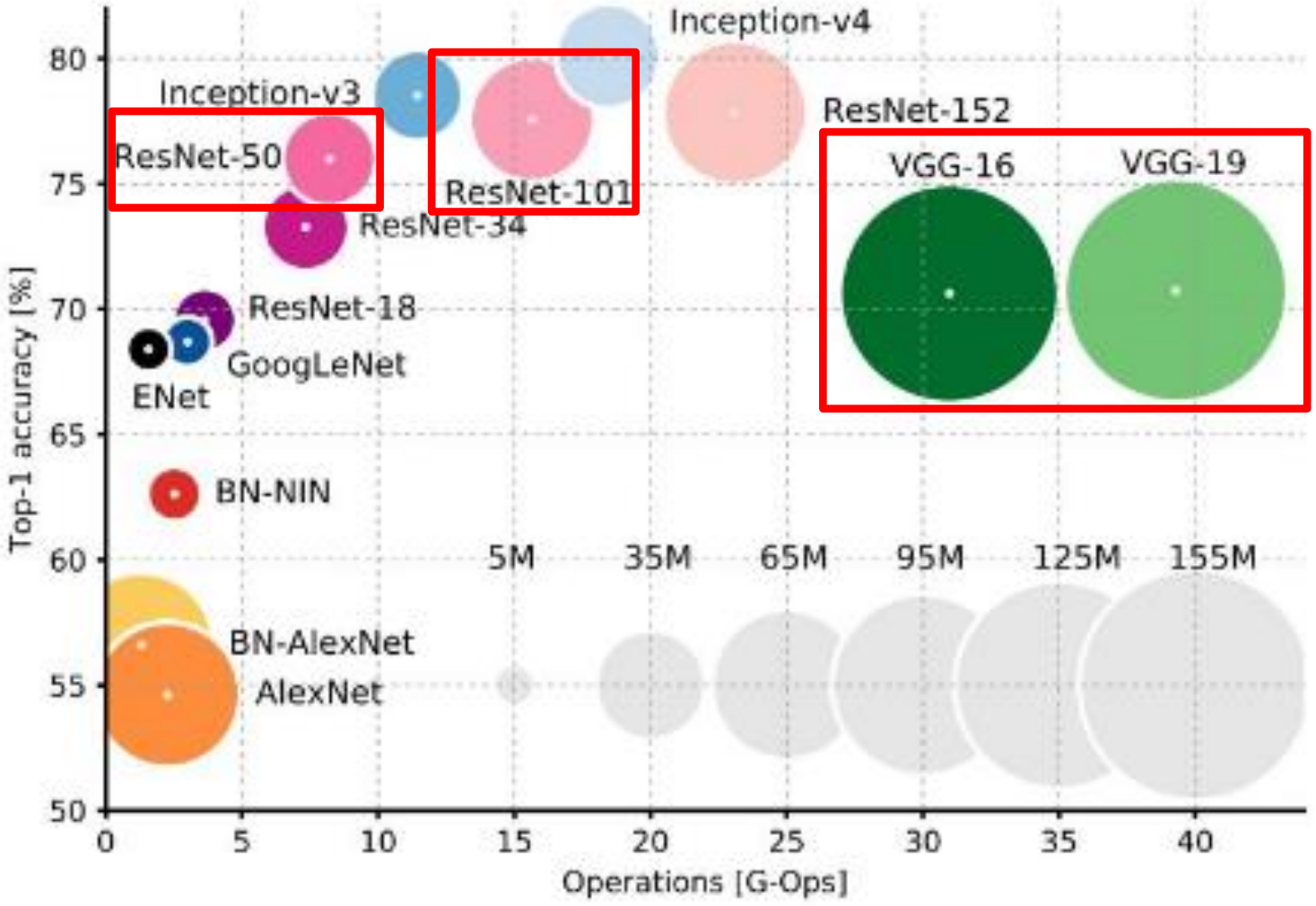
load selected 3 feature stage (c3,c4,c5) from backbone

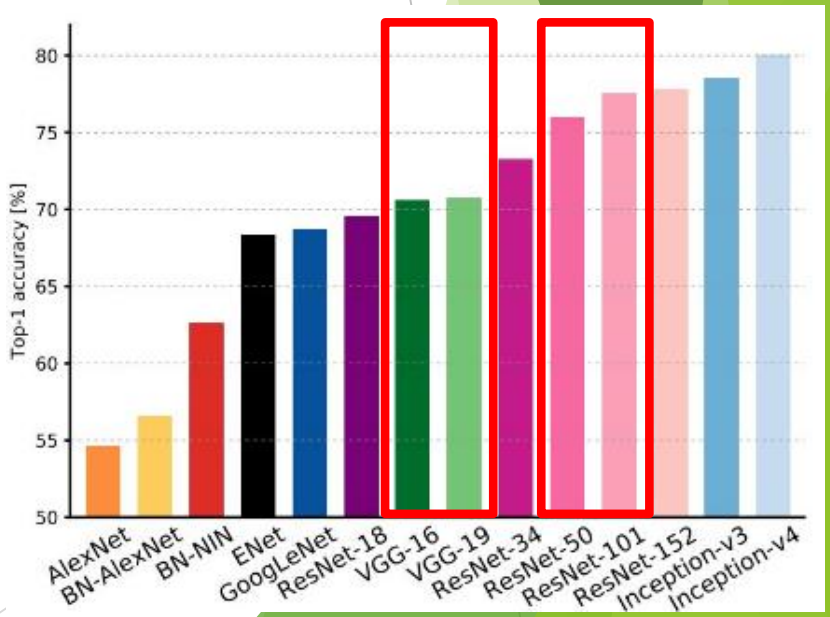
create feature pyramid network layers (Focal Loss)

for all pyramid levels, build submodels:

classification model | regression model

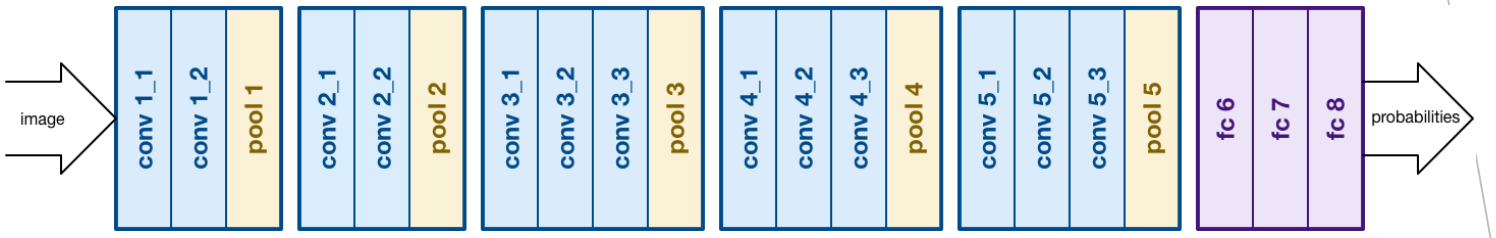
Which backbone to choose? Resnet, VGG



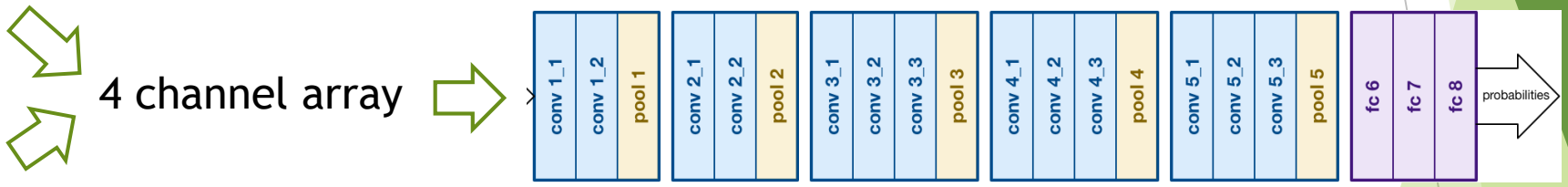


I am working on:

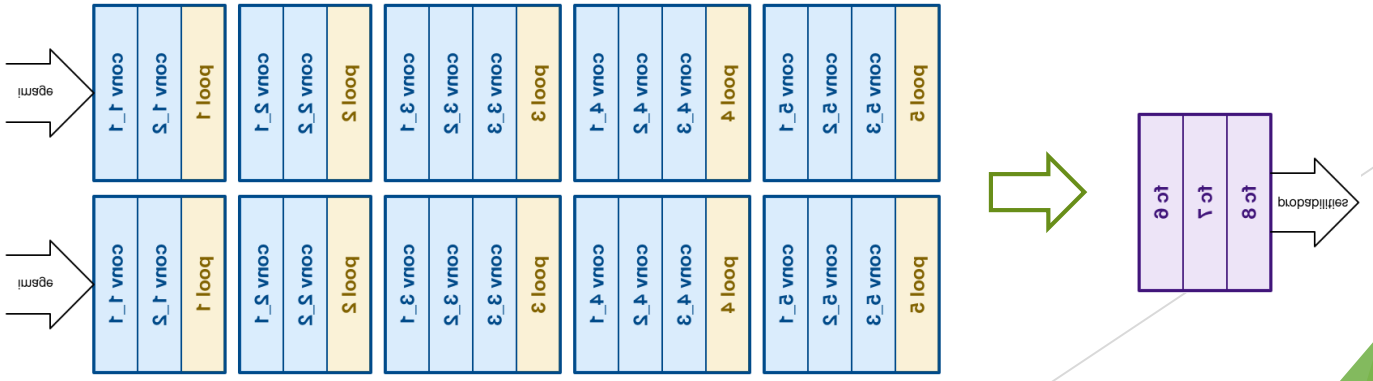
VGG original



VGG early



VGG late



VGG

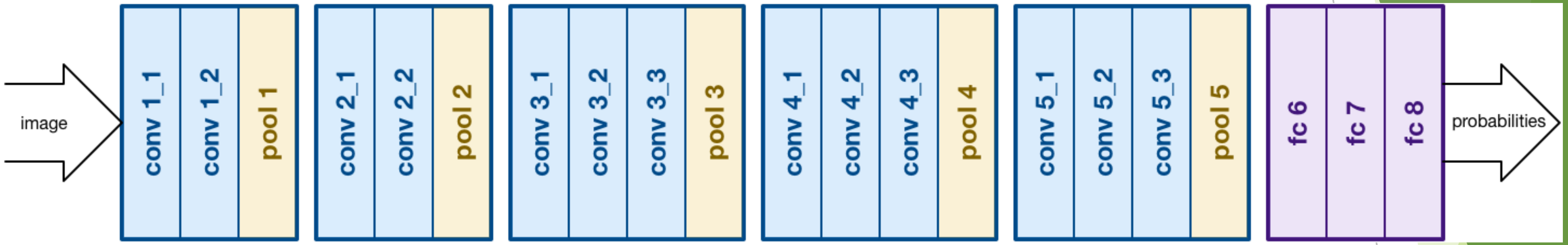
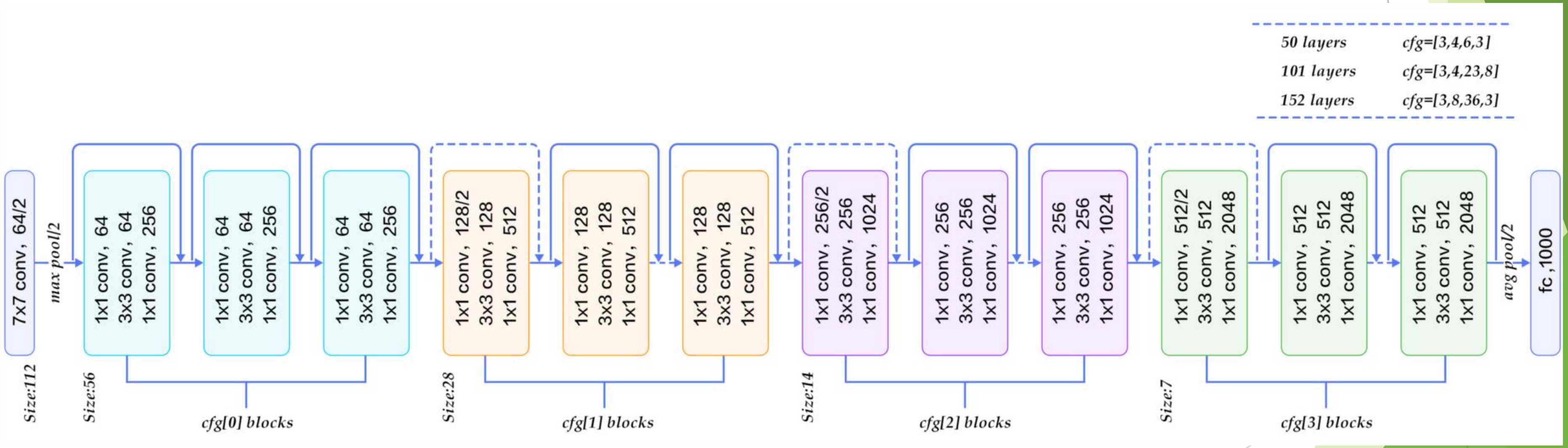


image
conv 1_1
conv 1_2
pool 1
conv 2_1
conv 2_2
pool 2
conv 3_1
conv 3_2
conv 3_3
pool 3
conv 4_1
conv 4_2
conv 4_3
pool 4
conv 5_1
conv 5_2
conv 5_3
pool 5
fc 6
fc 7
fc 8
probabilities

resnet


50 layers cfg=[3,4,6,3]
101 layers cfg=[3,4,23,8]
152 layers cfg=[3,8,36,3]
7x7 conv, 64/2
max pool/2
1x1 conv, 64
3x3 conv, 64
1x1 conv, 256
1x1 conv, 128/2
3x3 conv, 128
1x1 conv, 512
1x1 conv, 128
1x1 conv, 256/2
3x3 conv, 256
1x1 conv, 1024
1x1 conv, 256
1x1 conv, 512/2
3x3 conv, 512
1x1 conv, 2048
1x1 conv, 512
avg pool/2
fc, 1000
Size:112
Size:56
Size:28
Size:14
Size:7
cfg[0] blocks
cfg[1] blocks
cfg[2] blocks
cfg[3] blocks

What to do next?

- Training:
  - Modify backbone architecture：
    - Firstly vgg, then resnet
    - Firstly early, then late, finally halfway
  - Find out optimal value of hyperparameters
- Validation: take 20% data from set 0-5
- Testing:
  - Whole testing dataset
  - Understand why Evaluate.py gives very low mAP
  - Evaluate computational complexity
- report


THANK YOU

hyperparameters

- hyperparameters for building up network:
  - 1) Number of hidden layers and units
  - 2) Weight initialization

- hyperparameters to train network:
  - 1)Learning rate: speed vs convergence
  - 2)Number of epochs: observe validation accuracy
  - 3)Batch size: speed vs accuracy