基于双目系统的目标跟踪与预测

行人检测

指导老师：王梁昊
组员：王兴路、邱增辉、罗启睿

Outline

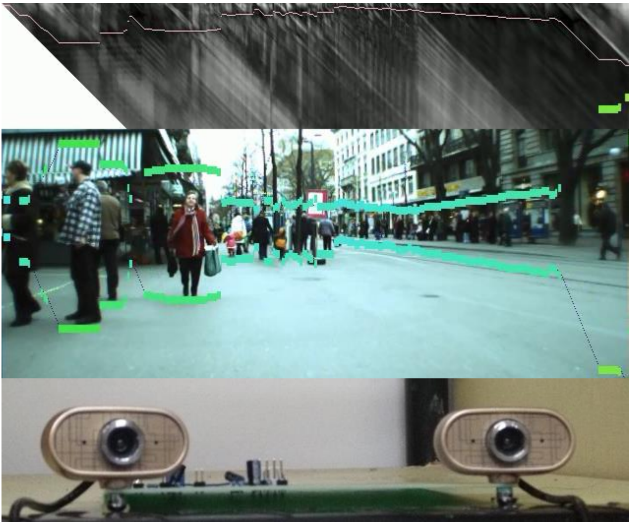
- Application
  - Whole System
  - People Flow Density Prediction
- Small Scale Pedestrian Detection
  - Data Augmentation
  - Why Degrade Performance?
  - Double Flow
  - Multiple Flow and Autoscale Reception Field

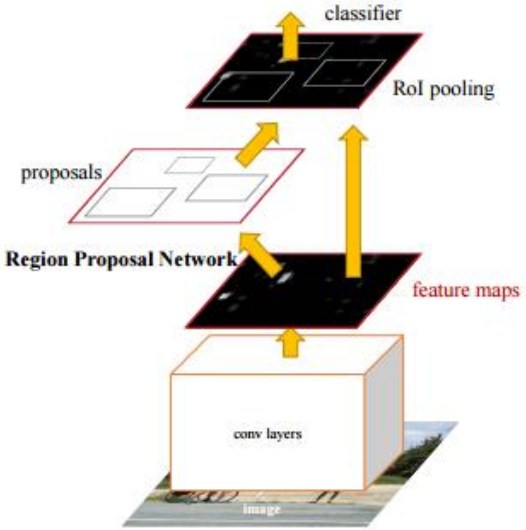
Outline

- Application
  - Whole System
  - People Flow Density Prediction
- Small Scale Pedestrian Detection
  - Data Augmentation
  - Why Degrade Performance?
  - Double Flow
  - Multiple Flow and Autoscale Reception Field

Whole System

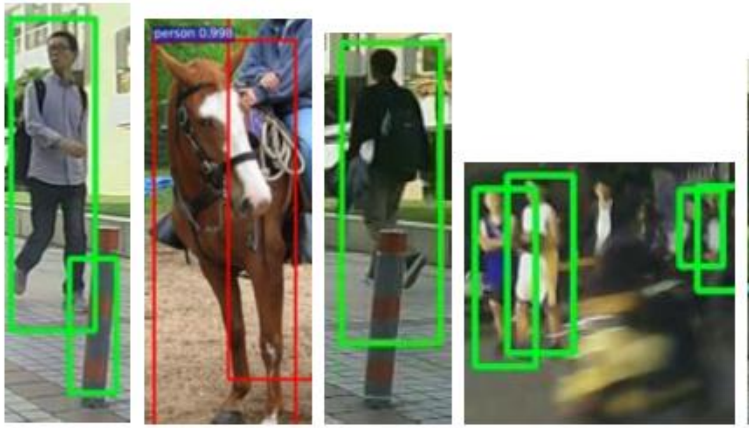
- Proposal in the Stixel world
- Detection by Faster-RCNN
- People Flow Density Prediction
- Then Focus on Scale Problem





People Flow Density Prediction

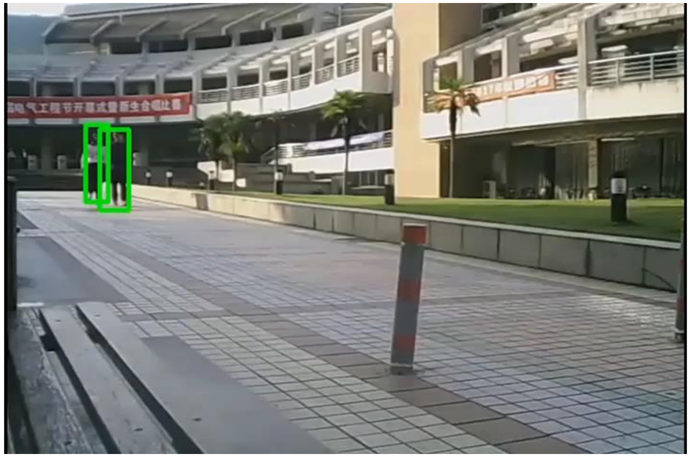
- Small Scale Pedestrian Detection
- Hard Negative Reduction
- Incorporate Prior into RPN Subnetwork
- Maintain Multiple Trackers
- Solve Association of Detections by Solving Assignment Problem, but how handle the noisy solutions.



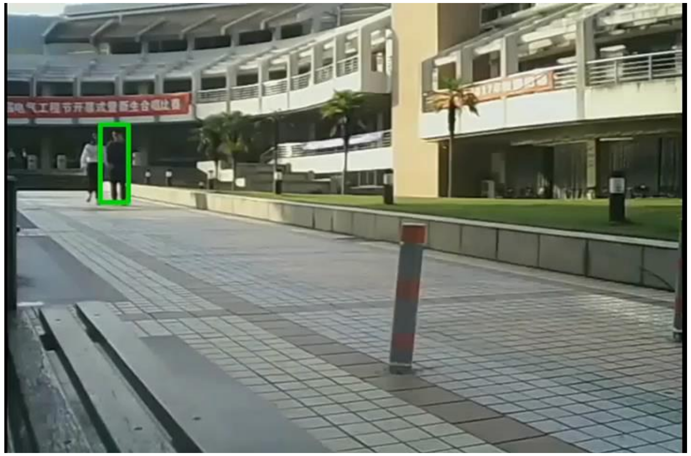
Outline

- Application
  - Whole System
  - People Flow Density Prediction
- Small Scale Pedestrian Detection
  - Data Augmentation
  - Why Degrade Performance?
  - Double Flow
  - Multiple Flow and Autoscale Reception Field

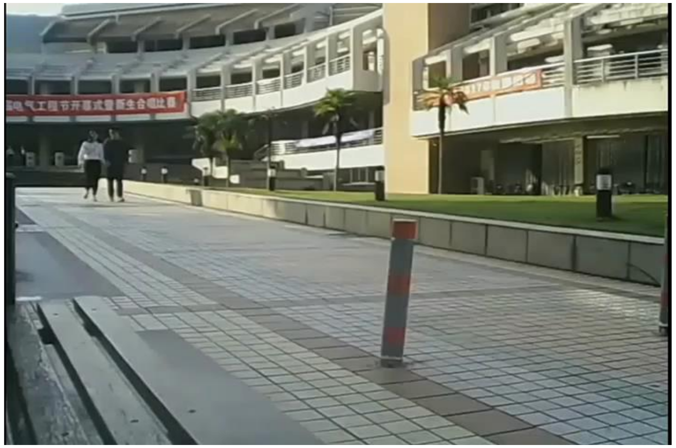
Data Augmentation



Train Without Data Augment



Data Augment: Use 0.8 Scale + Origin Scale



Just 0.8 Scale

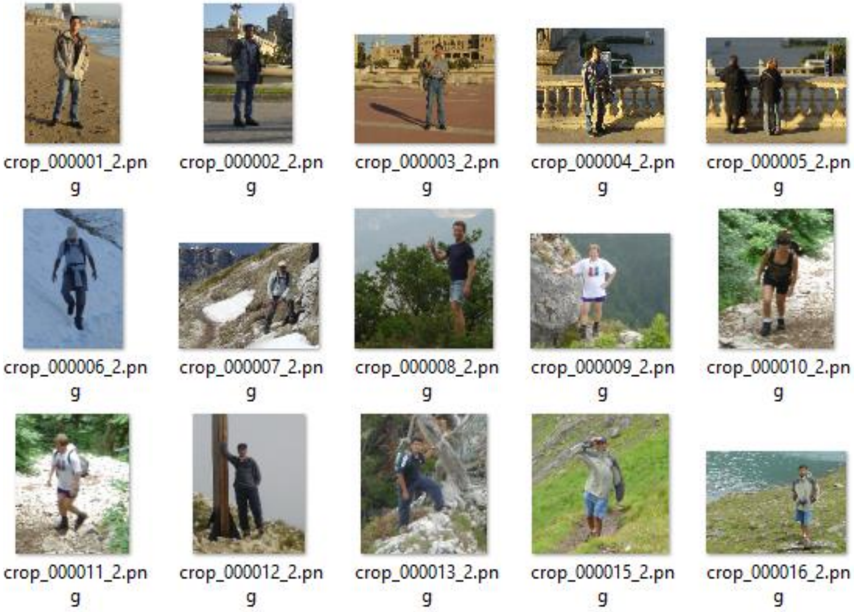
Data Augment Degrade Performance!

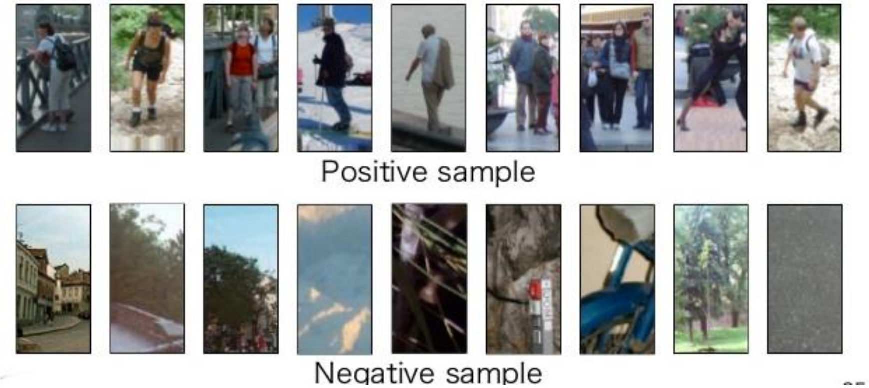
Data Augmentation

	Test on Origin Dataset	Test on 0.8 Scale Dataset	Test on Mixed Scale Dataset
Train on Origin Dataset	92.55%		
Train on 0.8 Scale Dataset	91.00%	81.19%	
Train on Mixed Scale Dataset	92.58%		88.95%

- Experiments Setting:
  - Evaluation Metric: $AP^{\mathrm{IoU}=0.5}, Ap\ at\ IoU = .50(PASCAL\ VOC\ Metric)$
  - Train img : Test img= 4964 : 548 $\sim$ 1: 9
  - Pos bbox : Neg bbox = 62900 : 62900 = 1 : 1
  - CONF_THRESH = 0.8   NMS_THRESH = 0.3

Inria Dataset

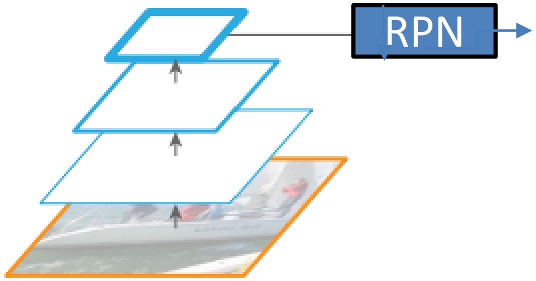



Positive sample

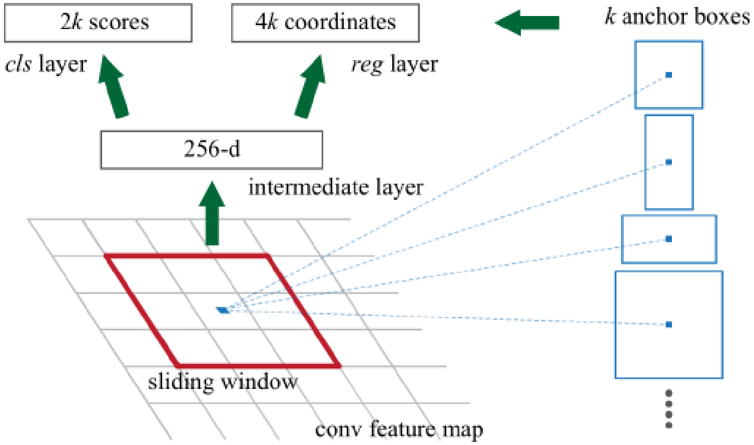
Negative sample

- Origin Image
- Cropped Pos and Neg Sample

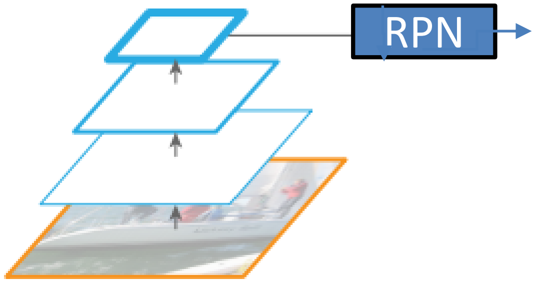
Why Degrade Performance?



- Single feature map

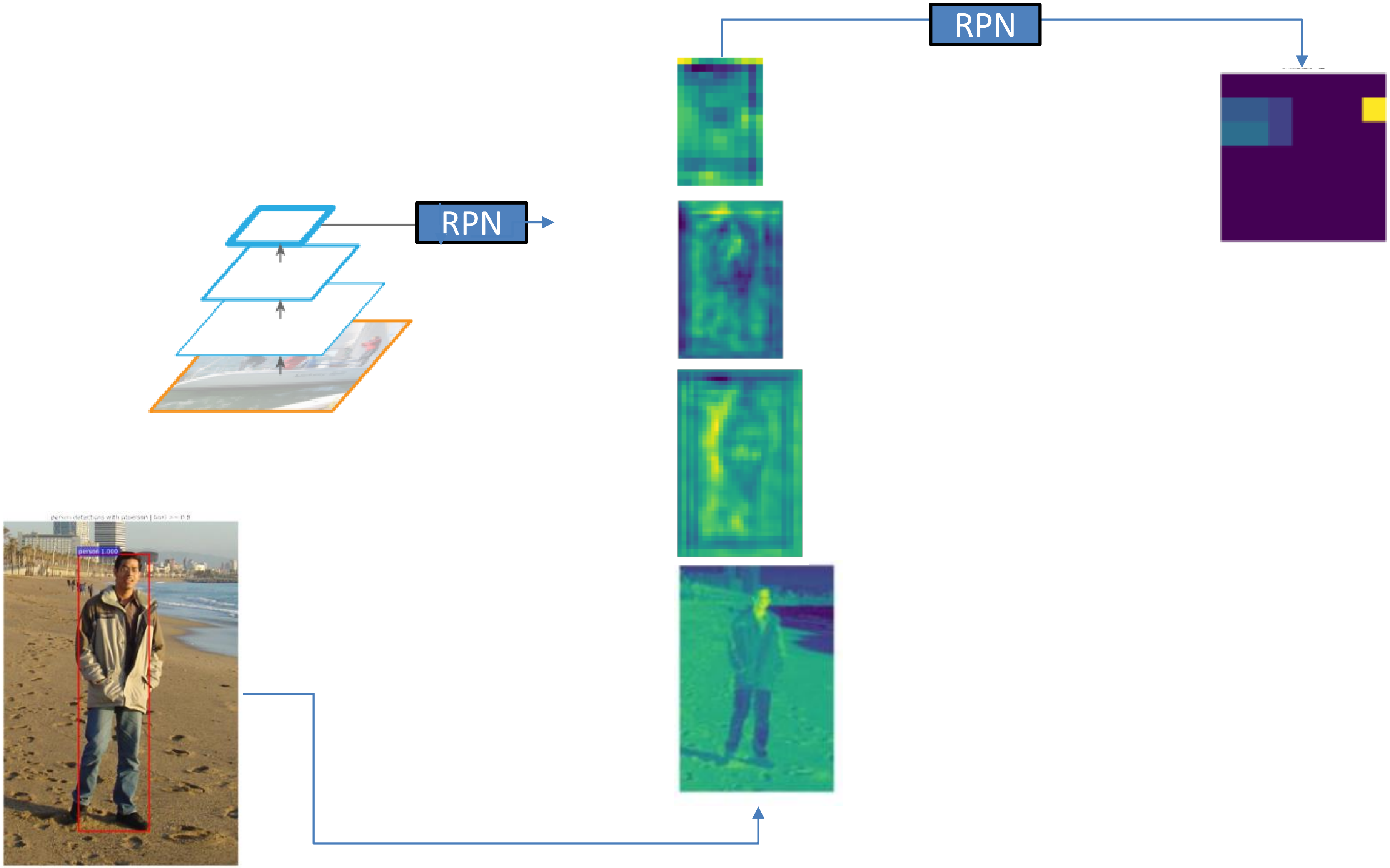


Scale Problem

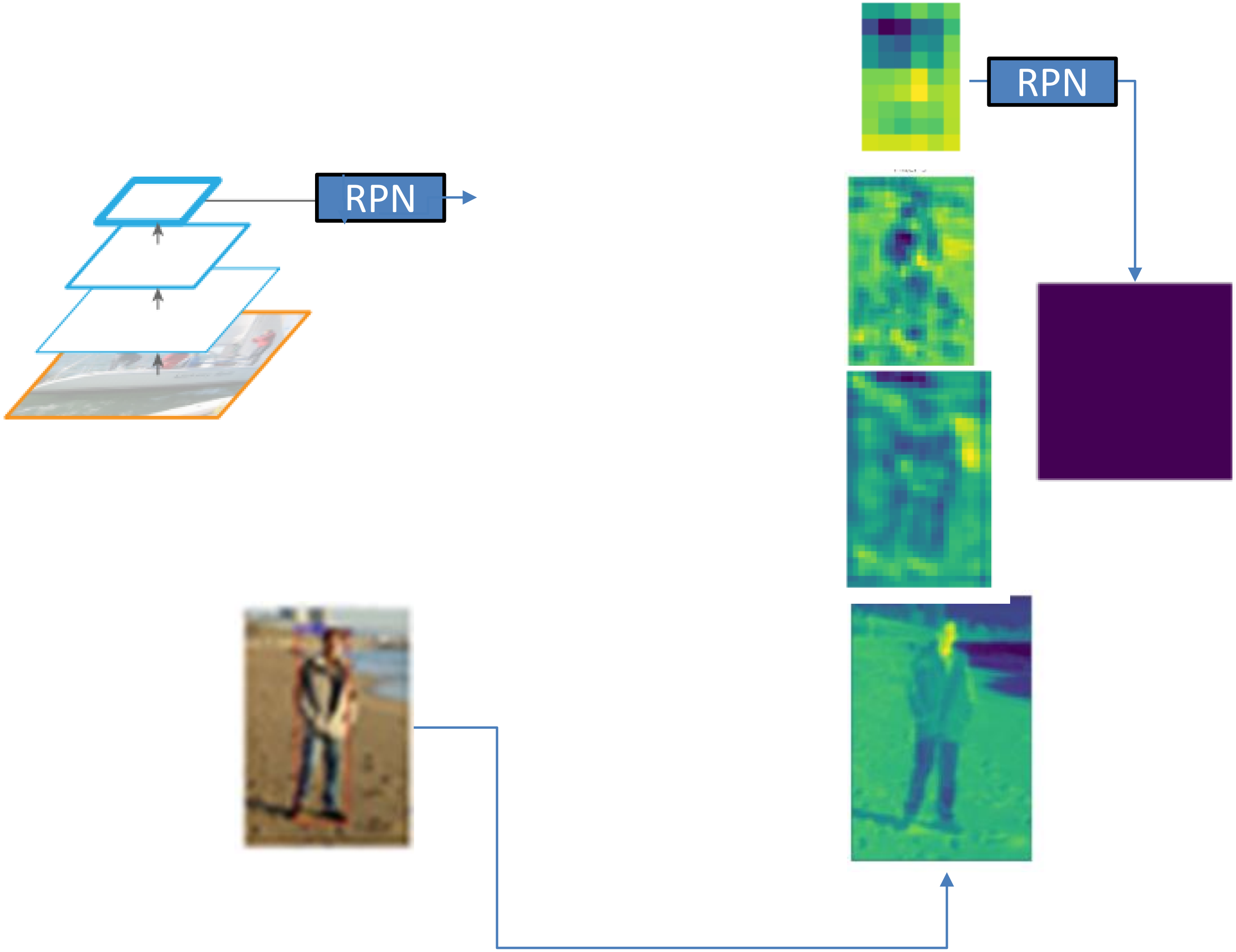


- Single feature map

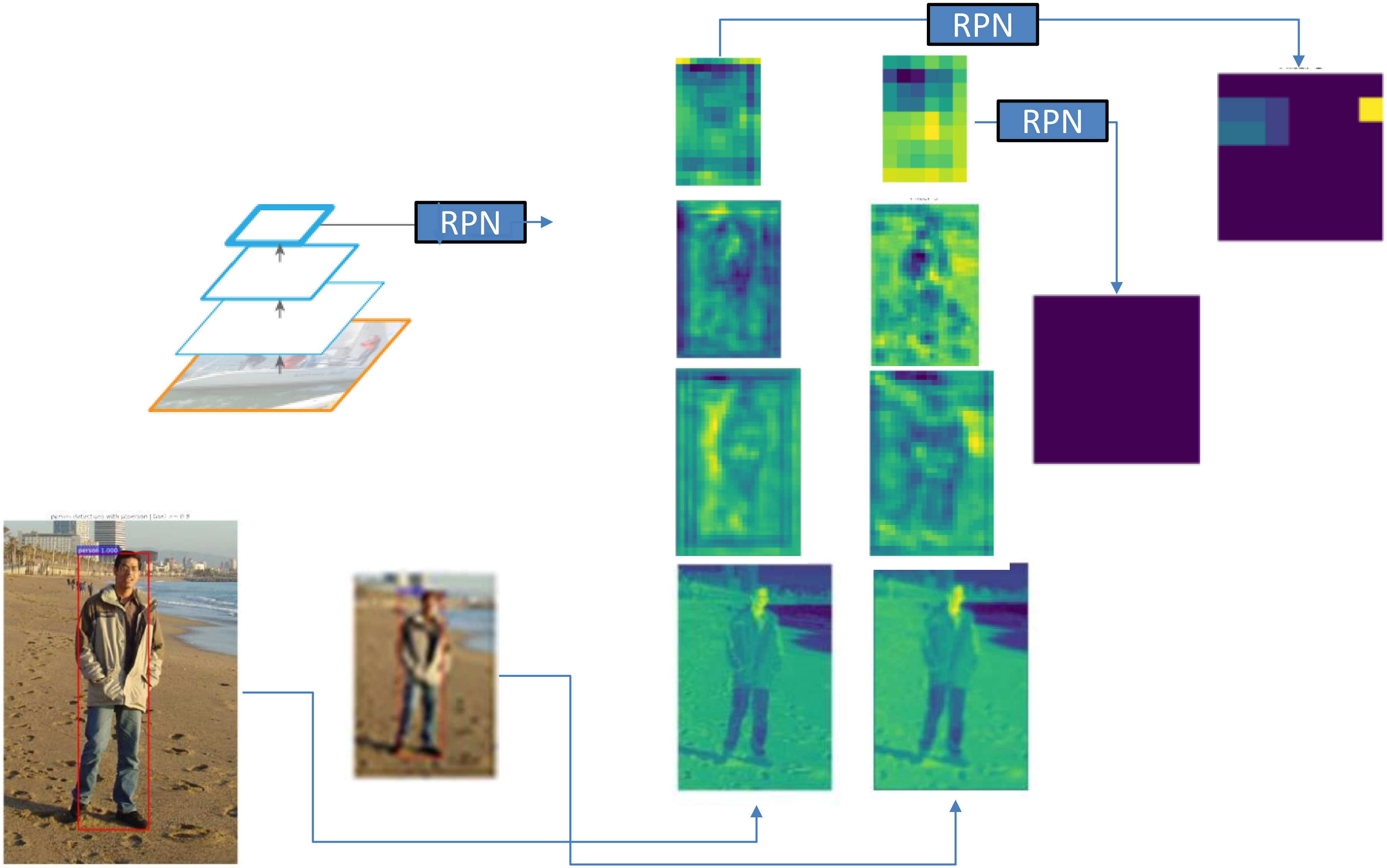
Feature Collapse



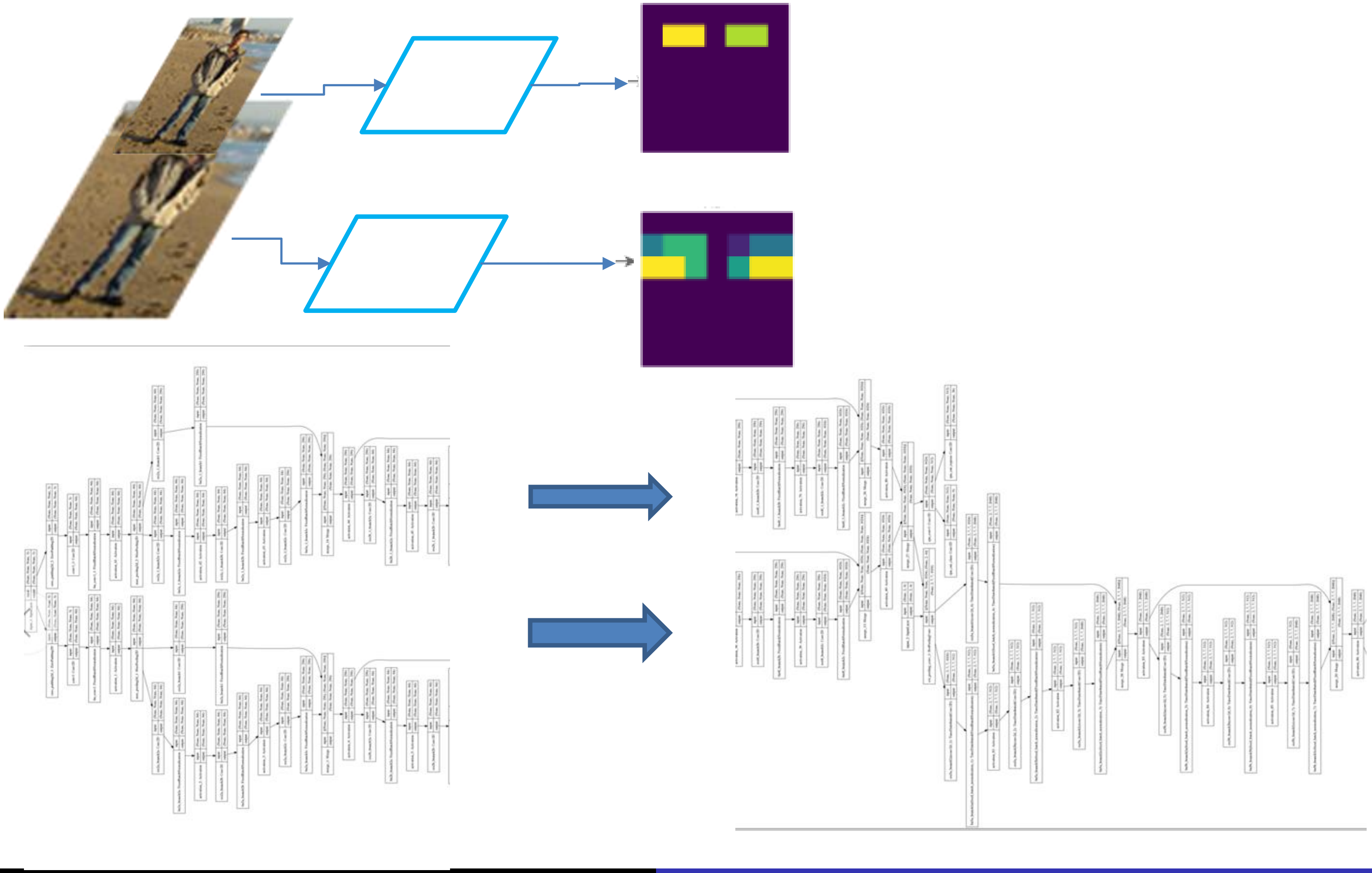
Feature Collapse



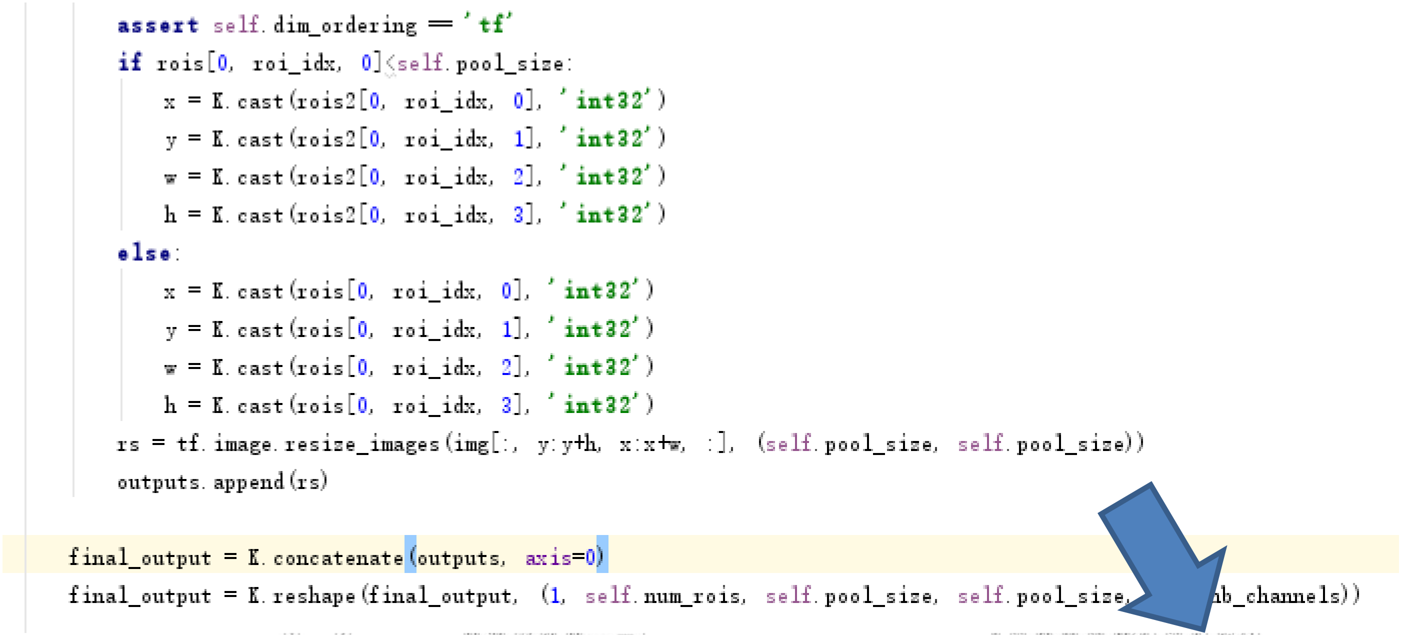
Feature Collapse



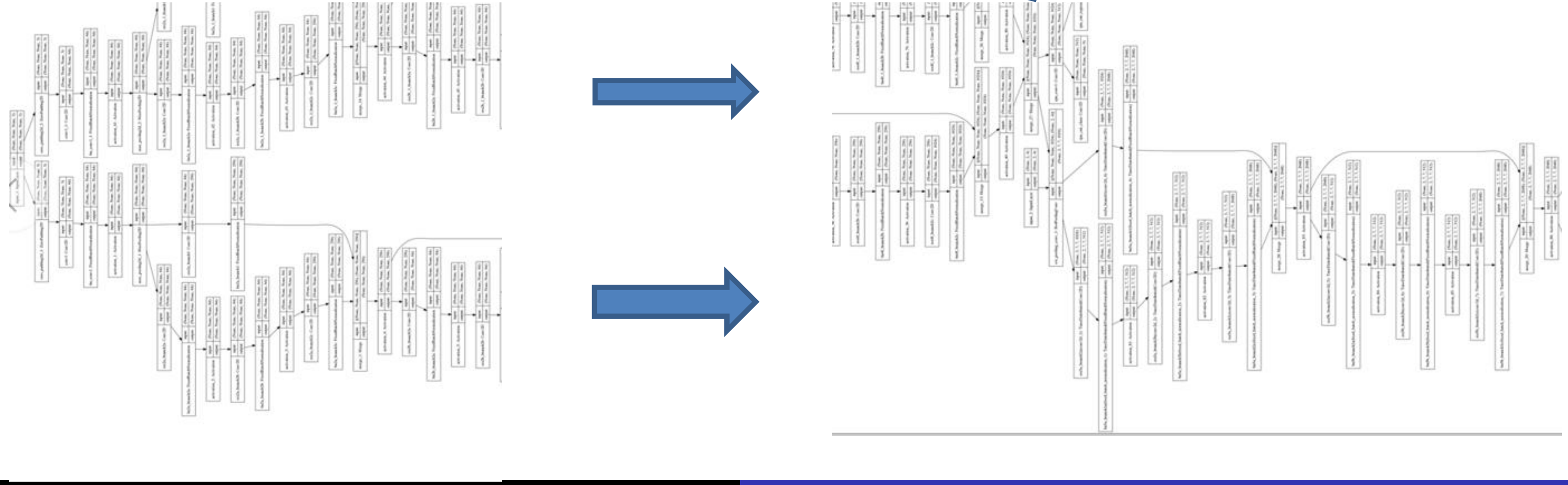
Double Flow



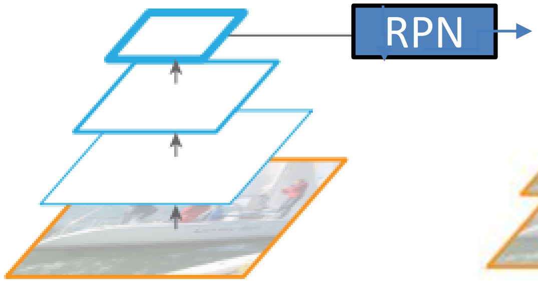
Double Flow



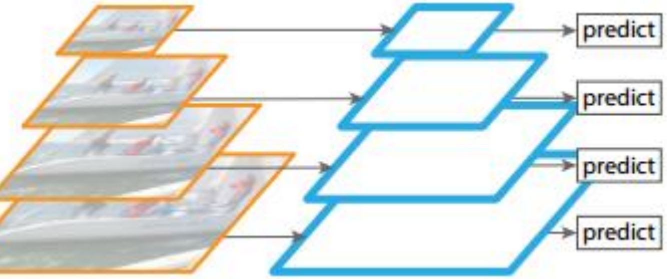
```
        assert self.dim_ordering == 'tf'
        if rois[0, roi_idx, 0]<self.pool_size:
            x = K.cast(rois2[0, roi_idx, 0], 'int32')
            y = K.cast(rois2[0, roi_idx, 1], 'int32')
            w = K.cast(rois2[0, roi_idx, 2], 'int32')
            h = K.cast(rois2[0, roi_idx, 3], 'int32')
        else:
            x = K.cast(rois[0, roi_idx, 0], 'int32')
            y = K.cast(rois[0, roi_idx, 1], 'int32')
            w = K.cast(rois[0, roi_idx, 2], 'int32')
            h = K.cast(rois[0, roi_idx, 3], 'int32')
        rs = tf.image.resize_images(img[:, y:y+h, x:x+w, :], (self.pool_size, self.pool_size))
        outputs.append(rs)

    final_output = K.concatenate(outputs, axis=0)
    final_output = K.reshape(final_output, (1, self.num_rois, self.pool_size, self.pool_size, nb_channels))
```



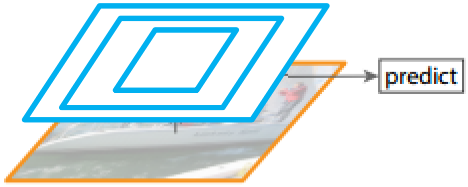
Related Method



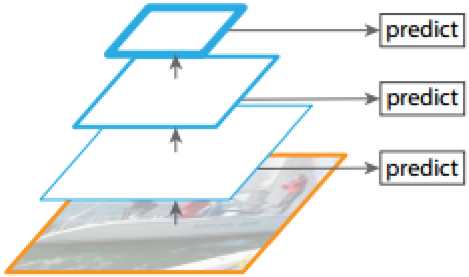
- Single feature map



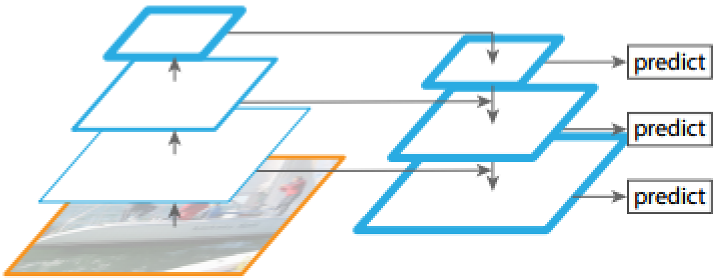
- Featurized image pyramid



- Filter pyramid



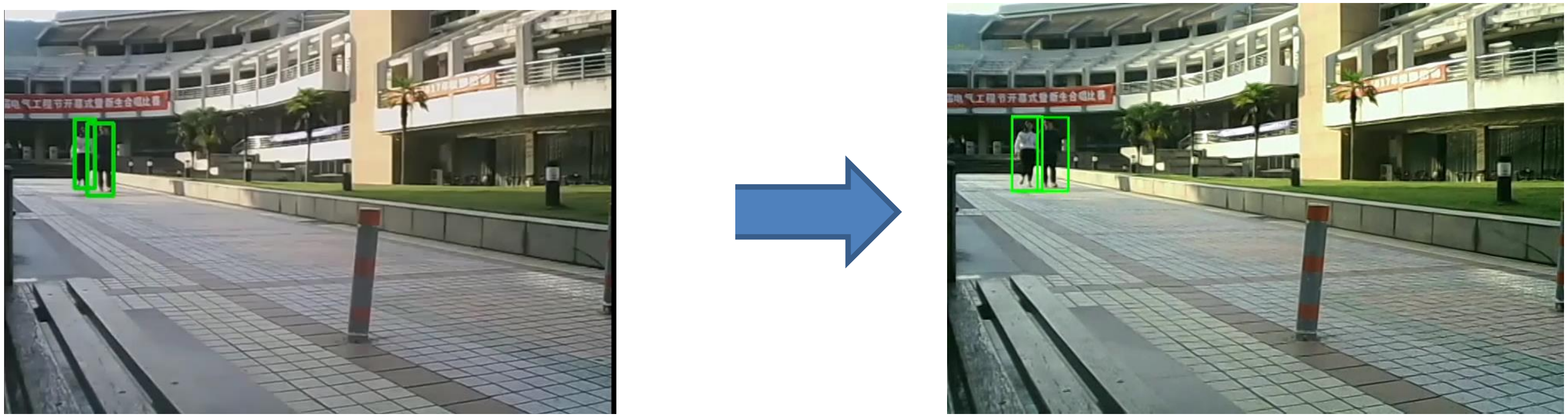
- Pyramidal feature hierarchy

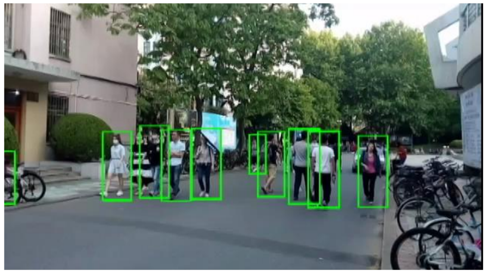


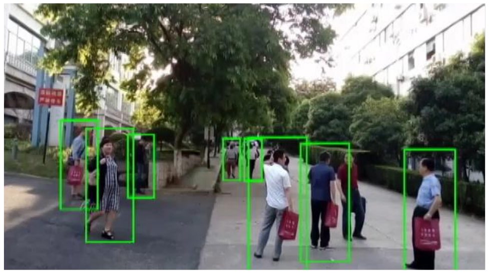
- Single feature map

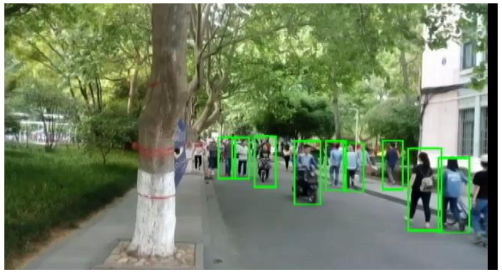
- Lin T Y, Dollár P, Girshick R, et al. Feature Pyramid Networks for Object Detection[J]. arXiv preprint arXiv:1612.03144, 2016. (CVPR 2017)

Small Scale Pedestrian Detection





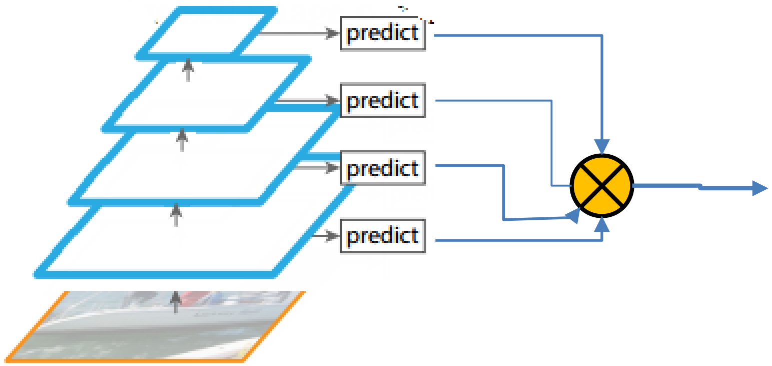




Double Flow

	Test on Origin Dataset	Test on 0.8 Scale Dataset	Test on Mixed Scale Dataset
Origin FRCNN	92.55%	81.19%	88.95%
Double Flow	92.58%	91.67%	88.21%

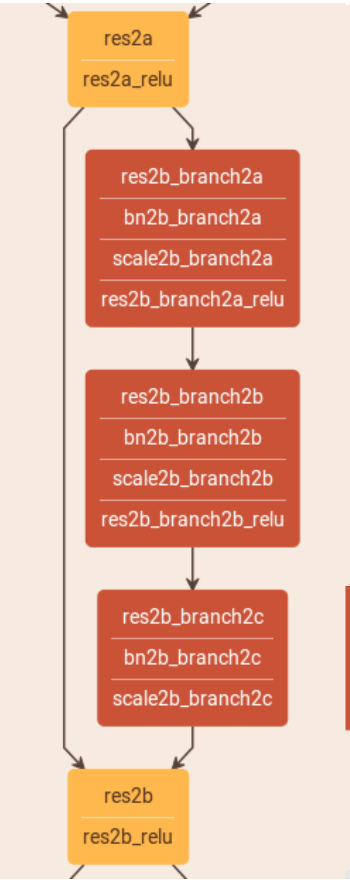
Multiple Flow

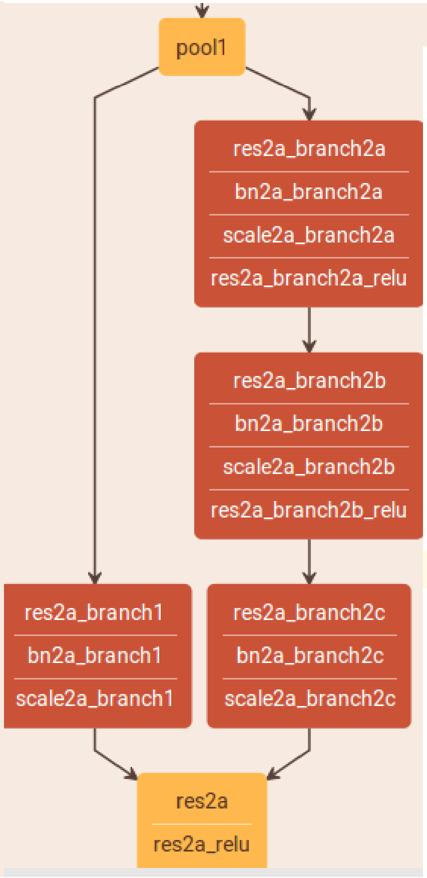


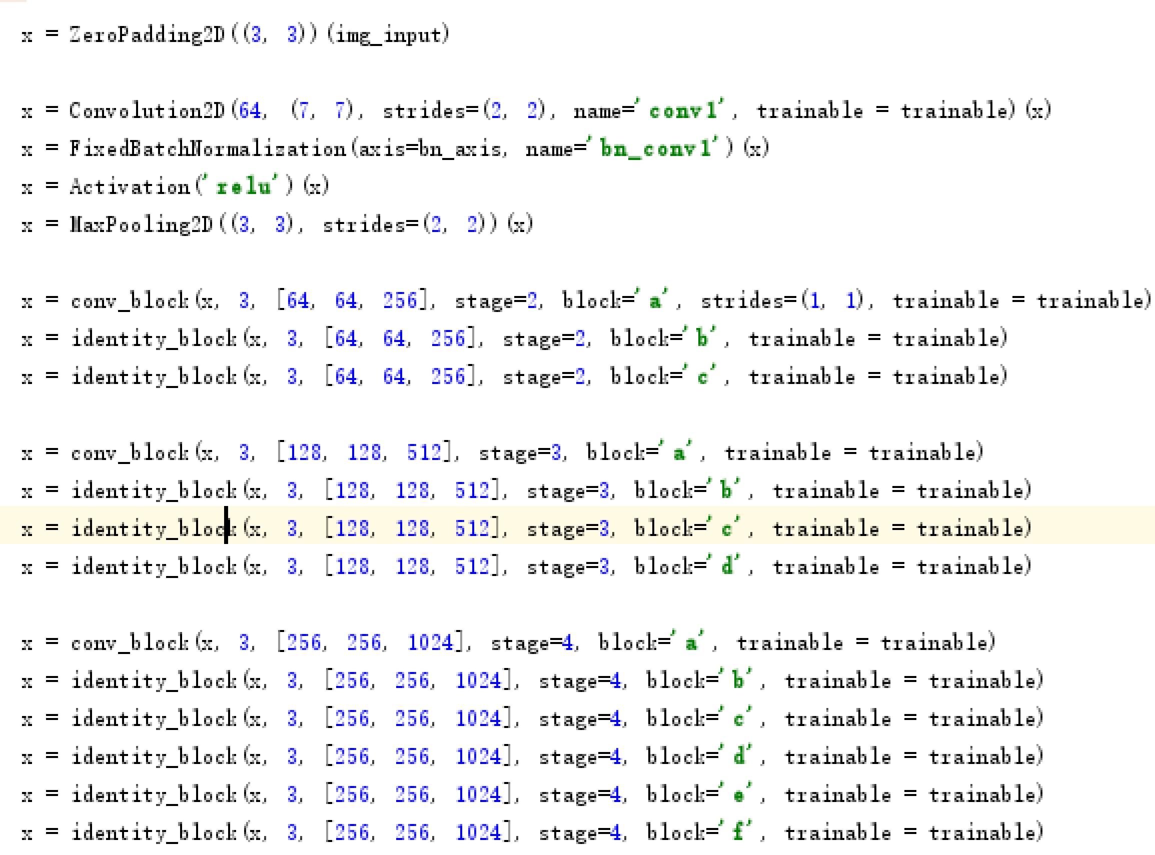
- Get Multi-Scale Feature Map in one Model
- New Concatenate Mode in ResNet
- Auto Select Different Scale
- Select According to ROI’s size
- then ROI pooling on Channel Dimension to connect to FC layer

Thank You!

ResNet







```
x = ZeroPadding2D((3, 3))(img_input)

x = Convolution2D(64, (7, 7), strides=(2, 2), name='conv1', trainable = trainable)(x)
x = FixedBatchNormalization(axis=bn_axis, name='bn_conv1')(x)
x = Activation('relu')(x)
x = MaxPooling2D((3, 3), strides=(2, 2))(x)

x = conv_block(x, 3, [64, 64, 256], stage=2, block='a', strides=(1, 1), trainable = trainable)
x = identity_block(x, 3, [64, 64, 256], stage=2, block='b', trainable = trainable)
x = identity_block(x, 3, [64, 64, 256], stage=2, block='c', trainable = trainable)

x = conv_block(x, 3, [128, 128, 512], stage=3, block='a', trainable = trainable)
x = identity_block(x, 3, [128, 128, 512], stage=3, block='b', trainable = trainable)
x = identity_block(x, 3, [128, 128, 512], stage=3, block='c', trainable = trainable)
x = identity_block(x, 3, [128, 128, 512], stage=3, block='d', trainable = trainable)

x = conv_block(x, 3, [256, 256, 1024], stage=4, block='a', trainable = trainable)
x = identity_block(x, 3, [256, 256, 1024], stage=4, block='b', trainable = trainable)
x = identity_block(x, 3, [256, 256, 1024], stage=4, block='c', trainable = trainable)
x = identity_block(x, 3, [256, 256, 1024], stage=4, block='d', trainable = trainable)
x = identity_block(x, 3, [256, 256, 1024], stage=4, block='e', trainable = trainable)
x = identity_block(x, 3, [256, 256, 1024], stage=4, block='f', trainable = trainable)
```

- Identity block
- Conv block