M2Det

A Single-Shot Object Detector based on Multi-Level Feature Pyramid Network

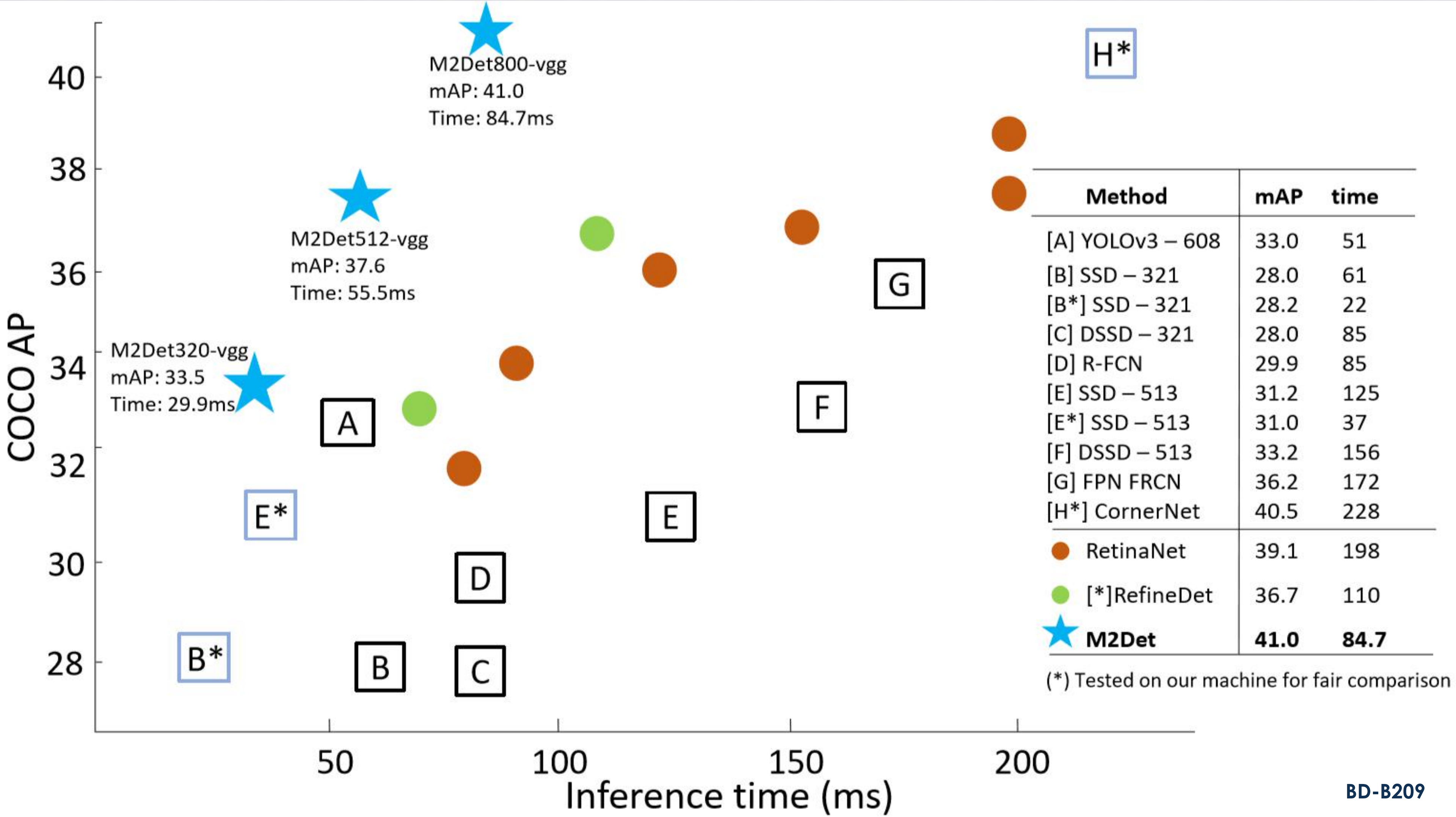
Reporter : GUOHUI XIANG

BD-B209

Introduction

Newly, in this work, we present **Multi-Level Feature Pyramid Network** (MLFPN) to construct more effective feature pyramids for detecting objects of different scales. **First**, we fuse multi-level features (i.e. multiple layers) extracted by backbone as the base feature. **Second**, we feed the base feature into a block of alternating joint Thinned U-shape Modules and Feature Fusion Modules and exploit the decoder layers of each U shape module as the features for detecting objects. **Finally**, we gather up the decoder layers with equivalent scales(sizes) to develop a feature pyramid for object detection, in which every feature map consists of the layers (features) from multiple levels.

We design and train a powerful end-to-end one stage object detector we call M2Det by integrating it into the architecture of SSD, and achieve better detection performance than state-of-the-art one-stage detectors. Specifically, on MS-COCO benchmark, M2Det achieves **AP of 41.0** at speed of **11.8 FPS** with single-scale inference strategy and **AP of 44.2** with multi-scale inference strategy, which are the new state-of-the-art results among one-stage detectors.



Method	mAP	time
[A] YOLOv3 – 608	33.0	51
[B] SSD – 321	28.0	61
[B*] SSD – 321	28.2	22
[C] DSSD – 321	28.0	85
[D] R-FCN	29.9	85
[E] SSD – 513	31.2	125
[E*] SSD – 513	31.0	37
[F] DSSD – 513	33.2	156
[G] FPN FRCN	36.2	172
[H*] CornerNet	40.5	228
RetinaNet	39.1	198
[*]RefineDet	36.7	110
M2Det	**41.0**	**84.7**

(*) Tested on our machine for fair comparison

BD-B209

CONTENTS

1

BD-B209

Introduction

In general, **high-level features** in the deeper layers are more discriminative for **classification** subtask while **low-level features** in the shallower layers can be helpful for object **location** regression sub-task. **Moreover, low-level features are more suitable to characterize objects with simple appearances while high-level features are appropriate for objects with complex appearances**. In practice, the appearances of the object instances with similar size can be quite different. For example, a traffic light and a faraway person may have comparable size, and the appearance of the person is much more complex. Hence, each feature map (used for detecting objects in a specific range of size) in the pyramid mainly or only consists of single-level features will result in suboptimal detection performance.

The goal of this paper is to construct a more effective feature pyramid for detecting objects of different scales

2

BD-B209

Related Work

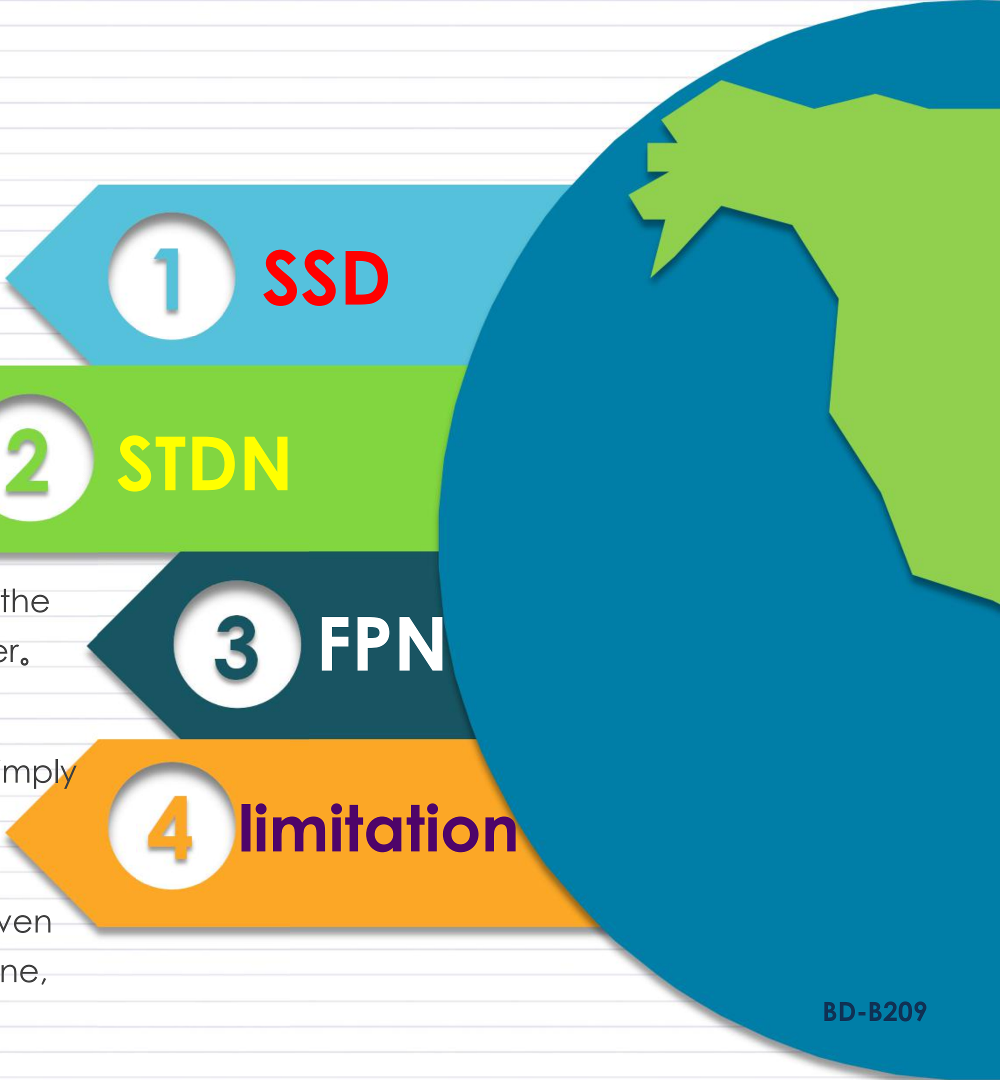
SSD directly and independently uses two layers of the backbone (i.e. VGG16) and four extra layers obtained by stride 2 convolution to construct the feature pyramid。

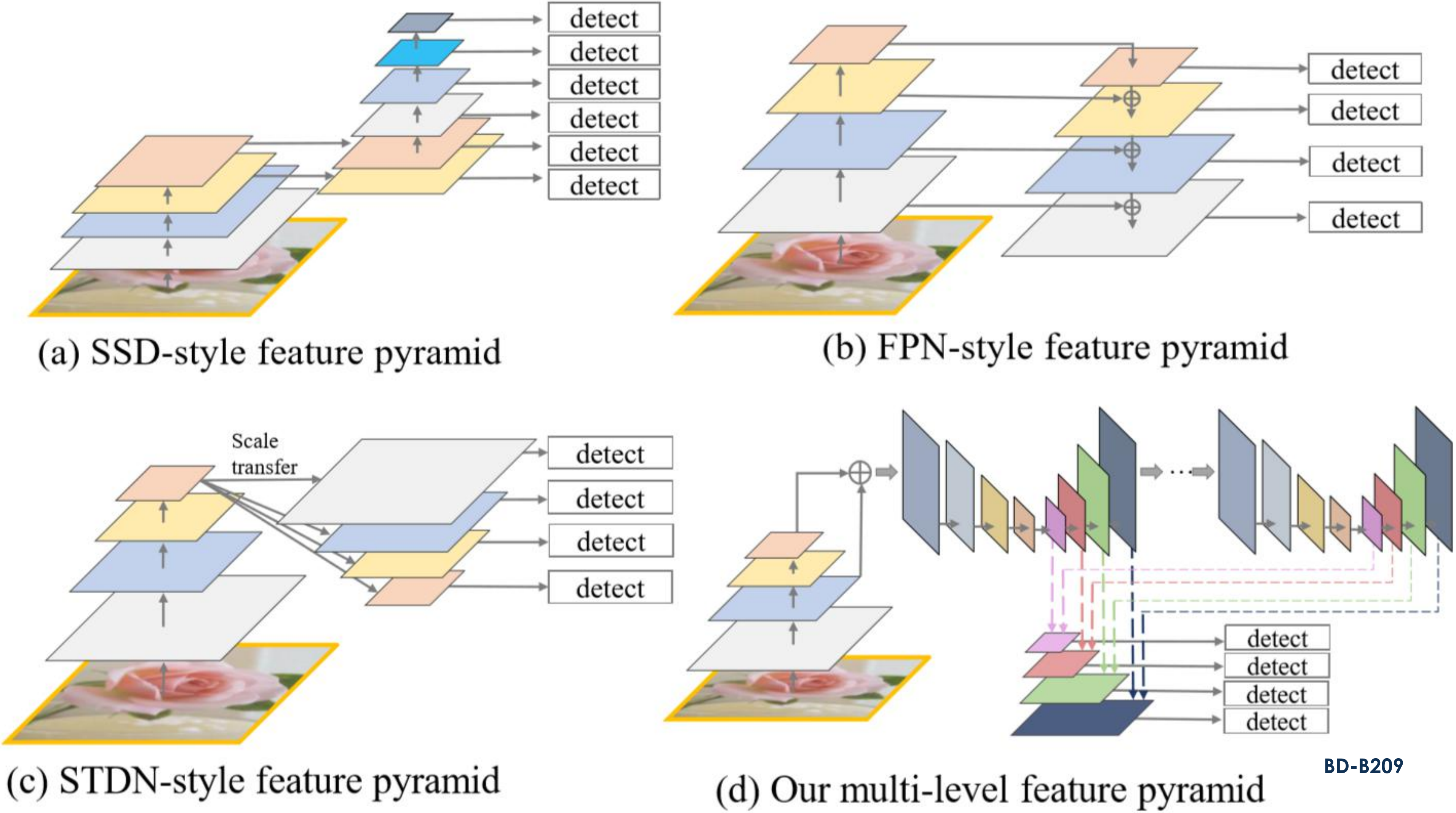
STDN only uses the last dense block of DenseNet to construct feature pyramid by pooling and scale-transfer operations。

FPN constructs the feature pyramid by fusing the deep and shallow layers in a top-down manner。

First, feature maps in the pyramid are not representative enough for the object detection task, instead they are simply constructed from the layers (features) of the backbone designed for object classification task.

Second, each feature map in the pyramid is mainly or even solely constructed from single-level layers of the backbone, that is, it mainly or only contains single-level information。




detect
detect
detect
detect
detect
detect
(a) SSD-style feature pyramid
detect
detect
detect
detect
(b) FPN-style feature pyramid
Scale
transfer
detect
detect
detect
detect
(c) STDN-style feature pyramid
detect
detect
detect
detect
(d) Our multi-level feature pyramid

BD-B209

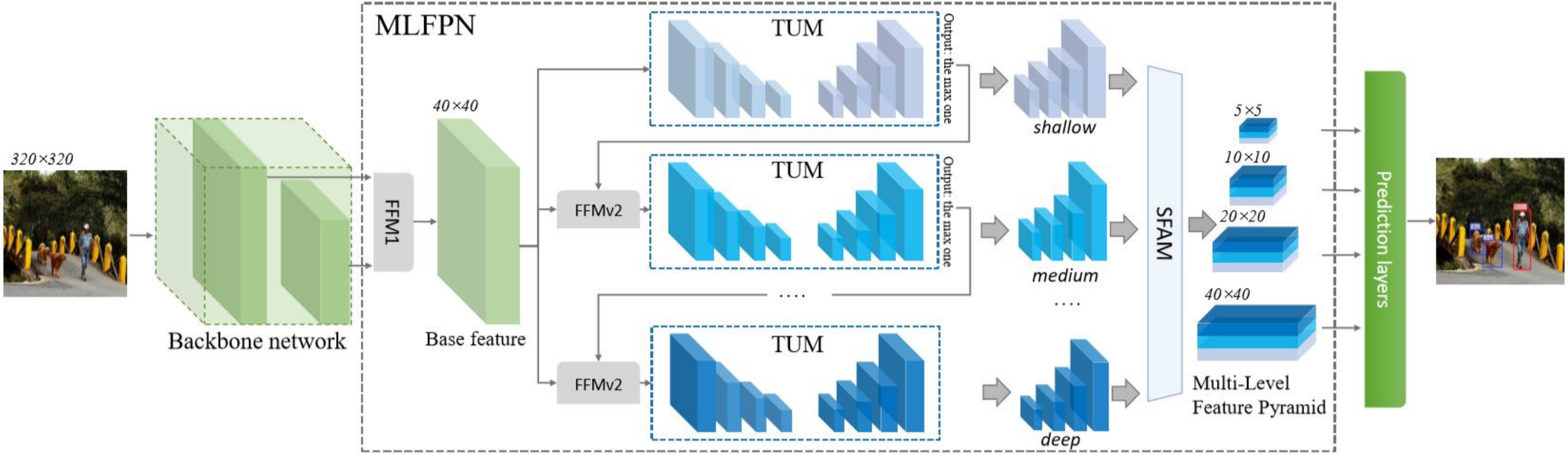
3

BD-B209

M2Det Model

M2Det uses the backbone and the Multi-Level Feature Pyramid Network (MLFPN) to **extract features** from the input image, and then similar to SSD, **produces dense bounding boxes and category scores** based on the learned features, followed by the **non-maximum suppression (NMS)** operation to produce the final results. MLFPN consists of three modules, i.e. Feature Fusion Module (**FFM**), Thinned U-shape Module (**TUM**) and Scale-wise Feature Aggregation Module (**SFAM**). FFMv1 enriches semantic information into base features by fusing feature maps of the backbone. Each TUM generates a group of multi-scale features, and then the alternating joint TUMs and FFMv2s extract multi-level multiscale features. In addition, SFAM aggregates the features into the multi-level feature pyramid through a scale-wise feature concatenation operation and an adaptive attention mechanism.

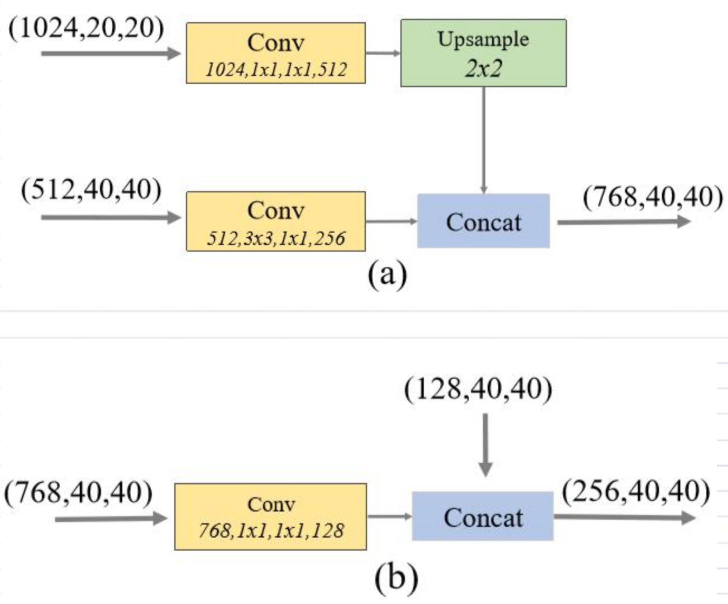
M2Det Model



An overview of the proposed M2Det(320×320). M2Det utilizes the backbone and the Multi-level Feature Pyramid Network (MLFPN) to extract features from the input image, and then produces dense bounding boxes and category scores. In MLFPN, FFMv1 fuses feature maps of the backbone to generate the base feature. Each TUM generates a group of multi-scale features, and then the alternating joint TUMs and FFMv2s extract multi-level multi-scale features. Finally, SFAM aggregates the features into a multi-level feature pyramid. In practice, we use 6 scales and 8 levels mostly.

FFMs

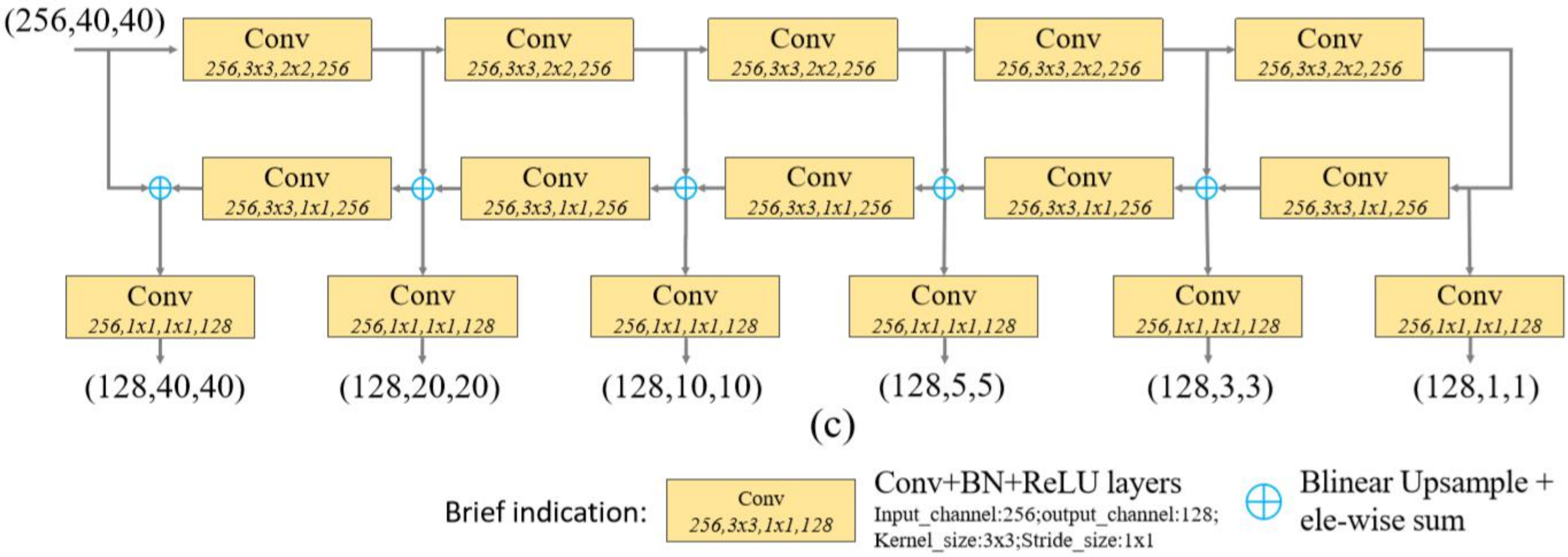
FFMs fuse features from different levels in M2Det. They use 1x1 convolution layers to compress the channels of the input features and use concatenation operation to aggregate these feature maps. Especially, since FFMv1 takes two feature maps with different scales in backbone as input, it adopts one upsample operation to rescale the deep features to the same scale before the concatenation operation. Meanwhile, FFMv2 takes the base feature and the largest output feature map of the previous TUM –these two are of the same scale – as input, and produces the fused feature for the next TUM。



TUMs

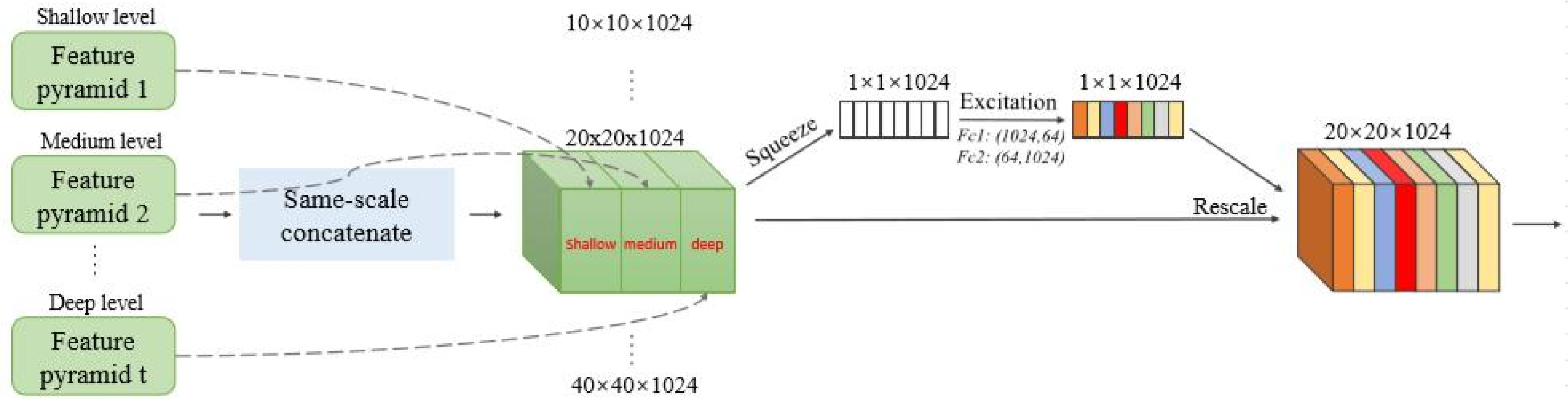
Different from FPN and RetinaNet ,TUM adopts a thinner U-shape structure. The encoder is a series of 3x3 convolution layers with stride 2. And the decoder takes the outputs of these layers as its reference set of feature maps, while the original FPN chooses the output of the last layer of each stage in ResNet backbone. In addition, we **add 1x1 convolution layers after upsample and elementwise sum operation at the decoder branch to enhance learning ability and keep smoothness for the features .**All of the outputs in the decoder of each TUM form the multi-scale features of the current level. As a whole, the outputs of stacked TUMs form the multi-level multi-scale features, while the front TUM mainly provides shallow-level features, the middle TUM provides mediumlevel features, and the back TUM provides deep-level features.

TUMs



(256,40,40)
Conv
256,3x3,2x2,256
Conv
256,3x3,2x2,256
Conv
256,3x3,2x2,256
Conv
256,3x3,2x2,256
Conv
256,3x3,2x2,256
Conv
256,3x3,1x1,256
Conv
256,3x3,1x1,256
Conv
256,3x3,1x1,256
Conv
256,3x3,1x1,256
Conv
256,3x3,1x1,256
Conv
256,1x1,1x1,128
Conv
256,1x1,1x1,128
Conv
256,1x1,1x1,128
Conv
256,1x1,1x1,128
Conv
256,1x1,1x1,128
Conv
256,1x1,1x1,128
(128,40,40)
(128,20,20)
(128,10,10)
(128,5,5)
(128,3,3)
(128,1,1)
(c)
Brief indication:
Conv
256,3x3,1x1,128
Conv+BN+ReLU layers
Input_channel:256;output_channel:128;
Kernel_size:3x3;Stride_size:1x1
Blinear Upsample +
ele-wise sum

SFAM



The first stage of SFAM is to concatenate features of the equivalent scale together along the channel dimension. However, simple concatenation operations are not adaptive enough. In the second stage, we introduce a channel-wise attention module to encourage features to focus on channels that they benefit most. Following SE block, we use global average pooling to generate channel-wise statistics $z \in R^C$ at the squeeze step. And to fully capture channel-wise dependencies, the following excitation step learns the attention mechanism via two fully connected layers:

$$\mathbf{s} = \mathbf{F}_{ex}(\mathbf{z}, \mathbf{W}) = \sigma(\mathbf{W}_2 \delta(\mathbf{W}_1 \mathbf{z}))$$

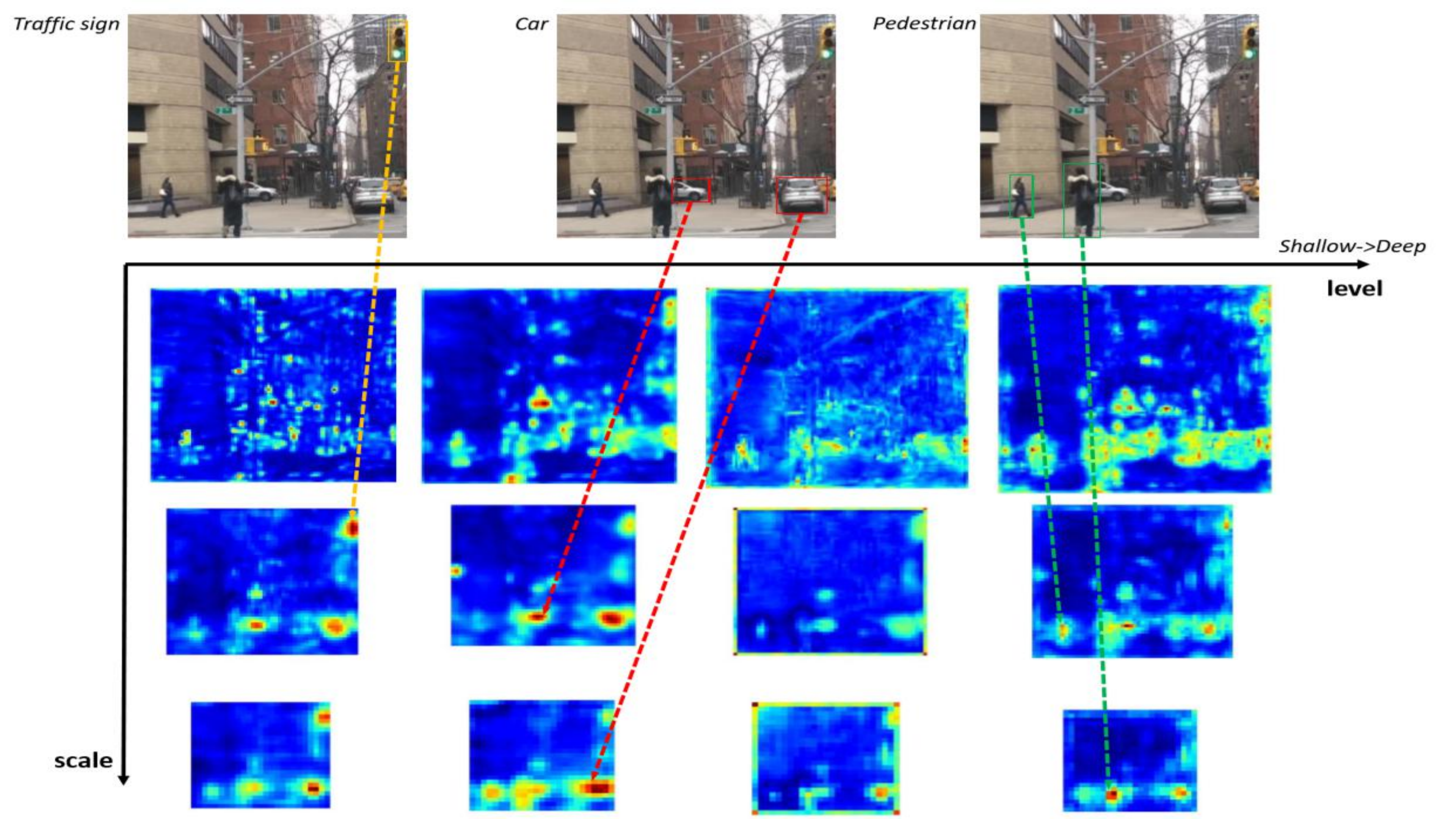
Network Configurations

We assemble M2Det with two kinds of backbones. Before training the whole network, the backbones need to be pre-trained on the ImageNet 2012 dataset .(All of the default configurations of MLFPN contain 8 TUMs, each TUM has 5 striding-Convs and 5 Upsample operations, so it will output features with 6 scales. To reduce the number of parameters, we only allocate 256 channels to each scale of their TUM features, so that the network could be easy to train on GPUs. As for input size, we follow the original SSD, RefineDet and RetinaNet, i.e., 320, 512 and 800.

4

BD-B209

Method	Backbone	Input size	MultiScale	FPS	Avg. Precision, IoU:			Avg. Precision, Area:		
					0.5:0.95	0.5	0.75	S	M	L
two-stage:										
Faster R-CNN (Ren et al. 2015)	VGG-16	~1000×600	False	7.0	21.9	42.7	-	-	-	-
OHEM++ (Shrivastava et al. 2016)	VGG-16	~1000×600	False	7.0	25.5	45.9	26.1	7.4	27.7	40.3
R-FCN (Dai et al. 2016)	ResNet-101	~1000×600	False	9	29.9	51.9	-	10.8	32.8	45.0
CoupleNet (Zhu et al. 2017)	ResNet-101	~1000×600	False	8.2	34.4	54.8	37.2	13.4	38.1	50.8
Faster R-CNN w FPN (Lin et al. 2017a)	Res101-FPN	~1000×600	False	6	36.2	59.1	39.0	18.2	39.0	48.2
Deformable R-FCN (Dai et al. 2017)	Inc-Res-v2	~1000×600	False	-	37.5	58.0	40.8	19.4	40.1	52.5
Mask R-CNN (He et al. 2017)	ResNeXt-101	~1280×800	False	3.3	39.8	62.3	43.4	22.1	43.2	51.2
Fitness-NMS (Tychsen-Smith and Petersson 2018)	ResNet-101	~1024×1024	True	5.0	41.8	60.9	44.9	21.5	45.0	57.5
Cascade R-CNN (Cai and Vasconcelos 2018)	Res101-FPN	~1280×800	False	7.1	42.8	62.1	46.3	23.7	45.5	55.2
SNIP (Singh and Davis 2018)	DPN-98	-	True	-	45.7	67.3	51.1	29.3	48.8	57.1
one-stage:										
SSD300* (Liu et al. 2016)	VGG-16	300×300	False	43	25.1	43.1	25.8	6.6	25.9	41.4
RON384++ (Kong et al. 2017)	VGG-16	384×384	False	15	27.4	49.5	27.1	-	-	-
DSSD321 (Fu et al. 2017)	ResNet-101	321×321	False	9.5	28.0	46.1	29.2	7.4	28.1	47.6
RetinaNet400 (Lin et al. 2017b)	ResNet-101	~640×400	False	12.3	31.9	49.5	34.1	11.6	35.8	48.5
RefineDet320 (Zhang et al. 2018)	VGG-16	320×320	False	38.7	29.4	49.2	31.3	10.0	32.0	44.4
RefineDet320 (Zhang et al. 2018)	ResNet-101	320×320	True	-	38.6	59.9	41.7	21.1	41.7	52.3
M2Det (Ours)	VGG-16	320×320	False	33.4	33.5	52.4	35.6	14.4	37.6	47.6
M2Det (Ours)	VGG-16	320×320	True	-	38.9	59.1	42.4	24.4	41.5	47.6
M2Det (Ours)	ResNet-101	320×320	False	21.7	34.3	53.5	36.5	14.8	38.8	47.9
M2Det (Ours)	ResNet-101	320×320	True	-	**39.7**	**60.0**	**43.3**	**25.3**	**42.5**	**48.3**
YOLOv3 (Redmon and Farhadi 2018)	DarkNet-53	608×608	False	19.8	33.0	57.9	34.4	18.3	35.4	41.9
SSD512* (Liu et al. 2016)	VGG-16	512×512	False	22	28.8	48.5	30.3	10.9	31.8	43.5
DSSD513 (Fu et al. 2017)	ResNet-101	513×513	False	5.5	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet500 (Lin et al. 2017b)	ResNet-101	~832×500	False	11.1	34.4	53.1	36.8	14.7	38.5	49.1
RefineDet512 (Zhang et al. 2018)	VGG-16	512×512	False	22.3	33.0	54.5	35.5	16.3	36.3	44.3
RefineDet512 (Zhang et al. 2018)	ResNet-101	512×512	True	-	41.8	62.9	45.7	25.6	45.1	54.1
CornerNet (Law and Deng 2018)	Hourglass	512×512	False	4.4	40.5	57.8	45.3	20.8	44.8	56.7
CornerNet (Law and Deng 2018)	Hourglass	512×512	True	-	42.1	57.8	45.3	20.8	44.8	56.7
M2Det (Ours)	VGG-16	512×512	False	18.0	37.6	56.6	40.5	18.4	43.4	51.2
M2Det (Ours)	VGG-16	512×512	True	-	42.9	62.5	47.2	28.0	47.4	52.8
M2Det (Ours)	ResNet-101	512×512	False	15.8	38.8	59.4	41.7	20.5	43.9	53.4
M2Det (Ours)	ResNet-101	512×512	True	-	**43.9**	**64.4**	**48.0**	**29.6**	**49.6**	**54.3**
RetinaNet800 (Lin et al. 2017b)	Res101-FPN	~1280×800	False	5.0	39.1	59.1	42.3	21.8	42.7	50.2
M2Det (Ours)	VGG-16	800×800	False	11.8	41.0	59.7	45.0	22.1	46.5	53.8
M2Det (Ours)	VGG-16	800×800	True	-	**44.2**	**64.6**	**49.3**	**29.2**	**47.9**	**55.1**


Traffic sign
Car
Pedestrian
Shallow->Deep
level
scale

Game Over

THANK YOU!

PRESENTED BY XGH