3.3 Region Based CNNs_part1
(Detection Problems)

1.2.2 Region Based CNNs(RCNNs)

CNNs that detect different types of objects in an image (Hybrid of classification and Regression problems)

Pioneer works

- *Viola-Jones*
- *HOG Detector*
- *DPM:* Deformable Parts Model

Two stage detectors

- RCNN
- *SPP-Net* (Spatial Pyramid Pooling)
- Fast RCNN
- Faster RCNN (FRCN)
- FPN
- RFCN (Region Fully Connected Network)
- Mask RCNN

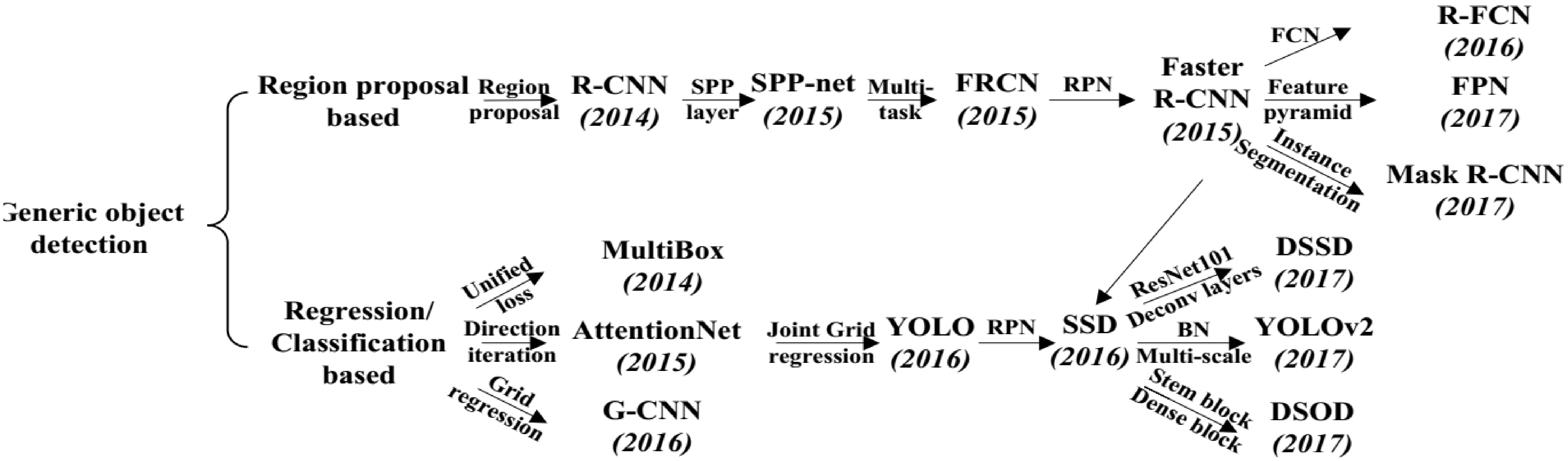
Single stage detectors

1. YOLO
2. SSD
3. YLOLO2, YOLO9000
4. Retina Net
5. YOLOv3
6. Center Net
7. EfficientDet
8. YOLOv4
9. Swin Transformer
10. YOLOX

* S. A. Zaidi (2021)

Evolutionary history of RCNNs 2014-2021

* Zhong-Qiu Zhao, 2019



Retina Net(2018) → YOLOv3(2018) →

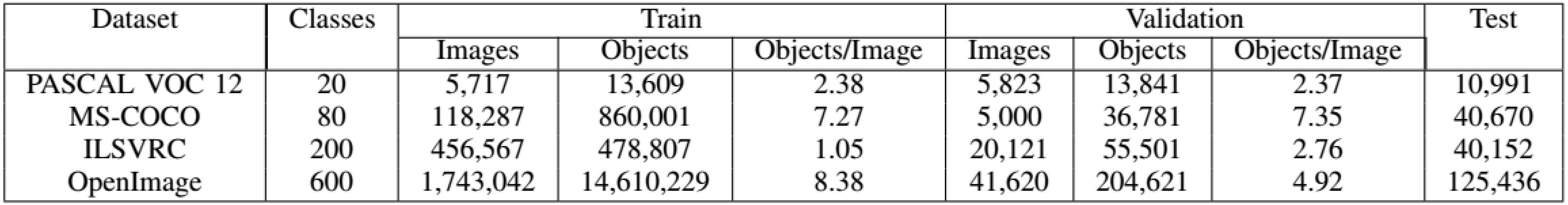
Center Net (2019) →

EfficientDet(2020) → YOLOv4 (2020) →

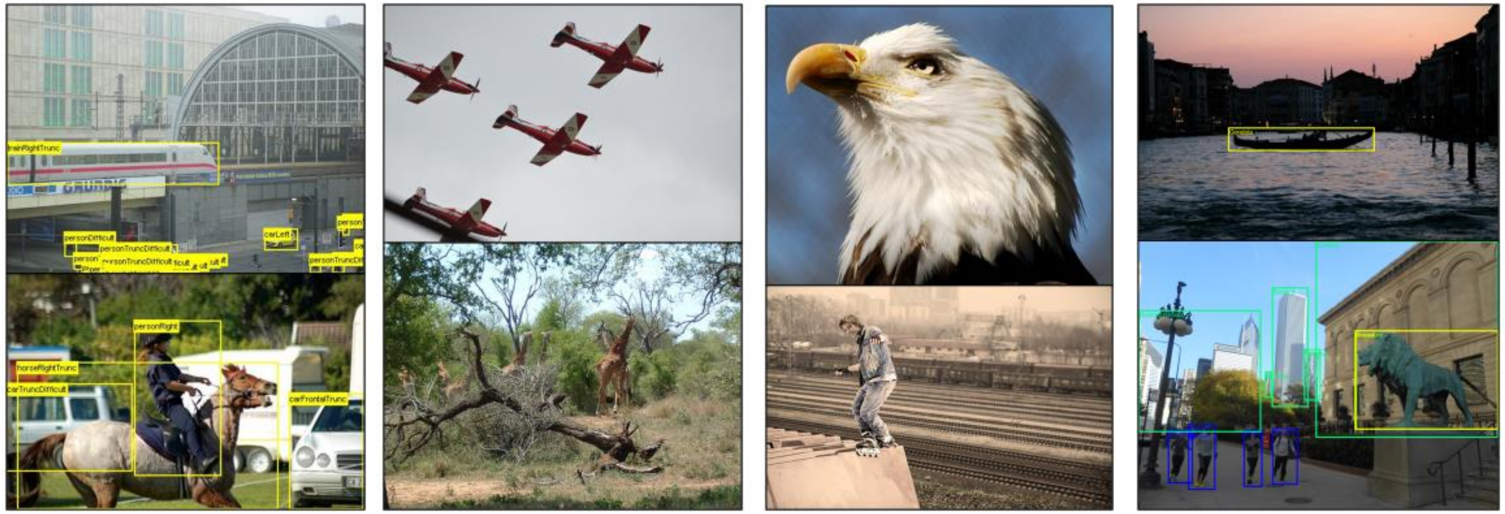
Swin Transformer (2021)

Data-sets for object detections

TABLE I: Comparison of various object detection datasets.



Dataset	Classes	Train			Validation			Test
		Images	Objects	Objects/Image	Images	Objects	Objects/Image	
PASCAL VOC 12	20	5,717	13,609	2.38	5,823	13,841	2.37	10,991
MS-COCO	80	118,287	860,001	7.27	5,000	36,781	7.35	40,670
ILSVRC	200	456,567	478,807	1.05	20,121	55,501	2.76	40,152
OpenImage	600	1,743,042	14,610,229	8.38	41,620	204,621	4.92	125,436



(a) PASCAL VOC 12 (b) MS-COCO (c) ILSVRC (d) OpenImage

Fig. 2: Sample images from different datasets.

Performance Comparison

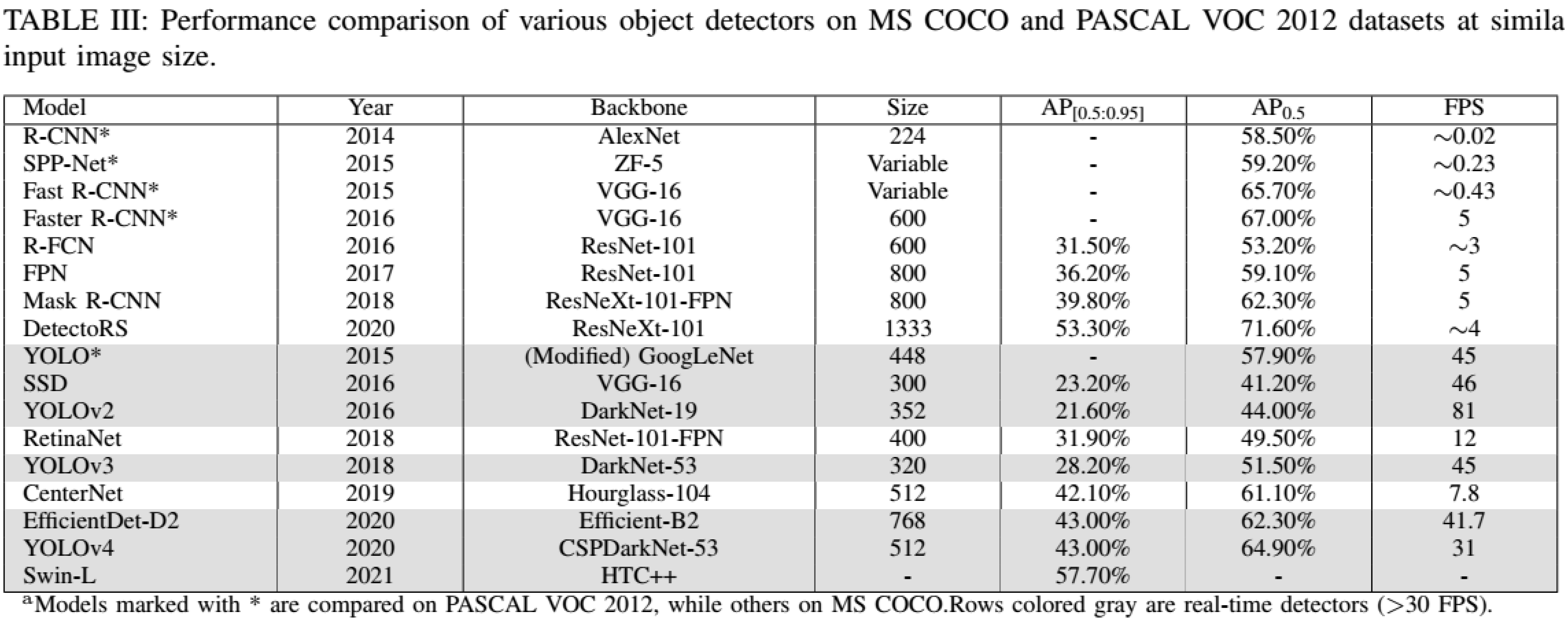


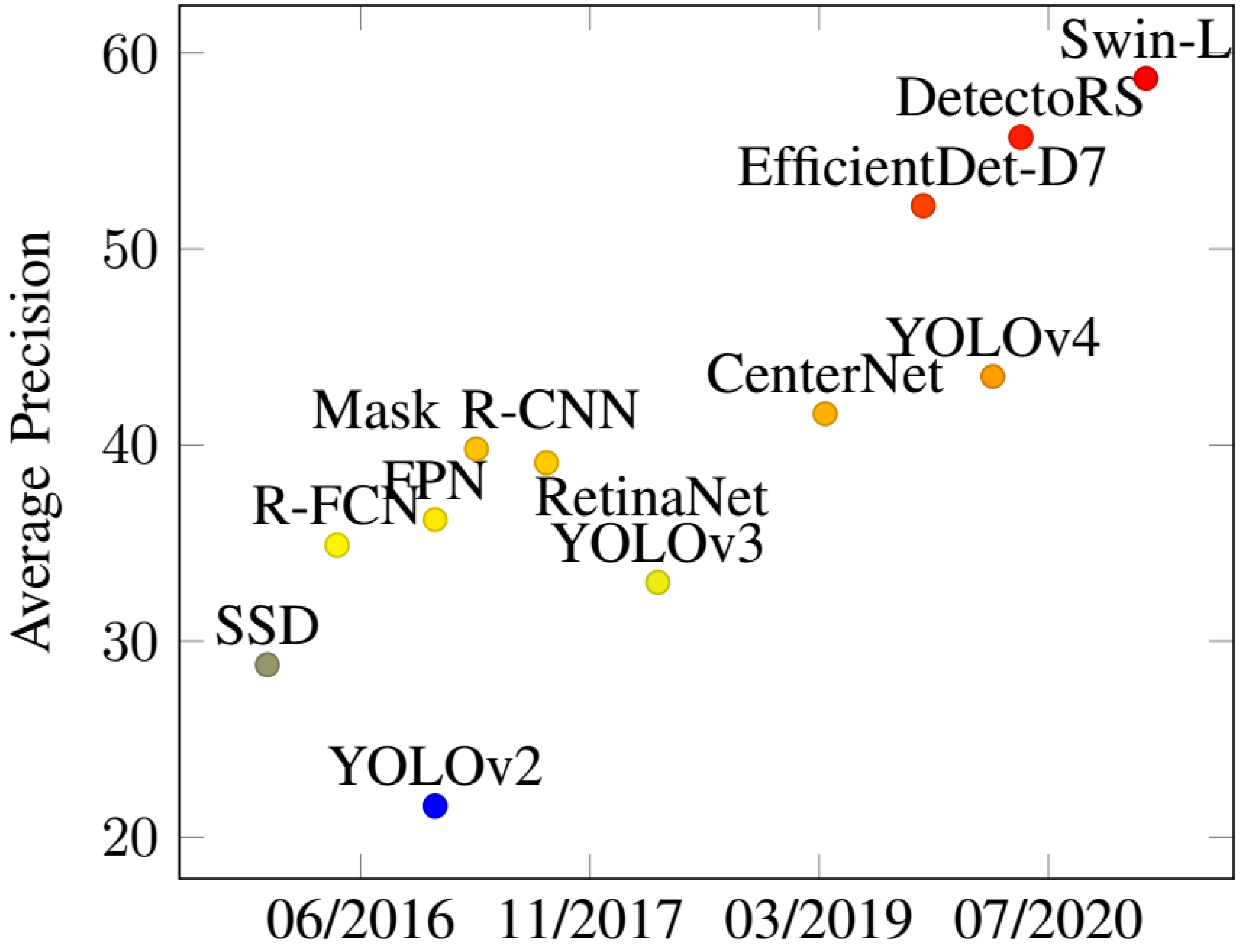
TABLE III: Performance comparison of various object detectors on MS COCO and PASCAL VOC 2012 datasets at simila input image size.

Model	Year	Backbone	Size	$AP_{[0.5:0.95]}$	$AP_{0.5}$	FPS
R-CNN*	2014	AlexNet	224	-	58.50%	∼0.02
SPP-Net*	2015	ZF-5	Variable	-	59.20%	∼0.23
Fast R-CNN*	2015	VGG-16	Variable	-	65.70%	∼0.43
Faster R-CNN*	2016	VGG-16	600	-	67.00%	5
R-FCN	2016	ResNet-101	600	31.50%	53.20%	∼3
FPN	2017	ResNet-101	800	36.20%	59.10%	5
Mask R-CNN	2018	ResNeXt-101-FPN	800	39.80%	62.30%	5
DetectoRS	2020	ResNeXt-101	1333	53.30%	71.60%	∼4
YOLO*	2015	(Modified) GoogLeNet	448	-	57.90%	45
SSD	2016	VGG-16	300	23.20%	41.20%	46
YOLOv2	2016	DarkNet-19	352	21.60%	44.00%	81
RetinaNet	2018	ResNet-101-FPN	400	31.90%	49.50%	12
YOLOv3	2018	DarkNet-53	320	28.20%	51.50%	45
CenterNet	2019	Hourglass-104	512	42.10%	61.10%	7.8
EfficientDet-D2	2020	Efficient-B2	768	43.00%	62.30%	41.7
YOLOv4	2020	CSPDarkNet-53	512	43.00%	64.90%	31
Swin-L	2021	HTC++	-	57.70%	-	-

[a]Models marked with * are compared on PASCAL VOC 2012, while others on MS COCO.Rows colored gray are real-time detectors (>30 FPS).

AP: Average Precision FPS: Frame per second

Performance of Object Detectors on MS COCO dataset.



(1) Pioneer Works

1. *Viola-Jones*
2. *HOG Detector*
3. *DPM:* Deformable Parts Model

Viola-Jones:

- Primarily designed for face detection, in 2001 Viola-Jones as an object detector was an accurate and powerful detector.
- It combined multiple techniques like Haar-like features, integral image, Adaboost and cascading classifier.
- Viola Jones algorithm is still used in small devices as it is very efficient and fast.

HOG: Histogram of Oriented Gradients

- In 2005, Dalal and Triggs proposed HOG.
- HOG extracts gradient and its orientation of the edges to create a feature table.
- The image is divided into grids and the feature table is then used to create histogram for each cell in the grid.
- HOG features are generated for the region of interest and fed into a linear SVM classifier for detection.
- The detector was proposed for pedestrian detection; however, it could be trained to detect various classes.

DPM: Deformable Parts Model

- DPM was introduced by Felzenszwalb et al. and was the winner Pascal VOC challenge in 2009.
- It used individual “part” of the object for detection and achieved higher accuracy than HOG.
- It follows the philosophy of *divide and rule*; parts of the object are individually detected during inference time and a probable arrangement of them is marked as detection.

(1) Two stage Object detectors

- RCNN
- *SPP-Net* (Spatial Pyramid Pooling)
- Fast RCNN
- Faster RCNN (FRCN)
- FPN
- RFCN (Region Fully Connected Network)
- Mask RCNN

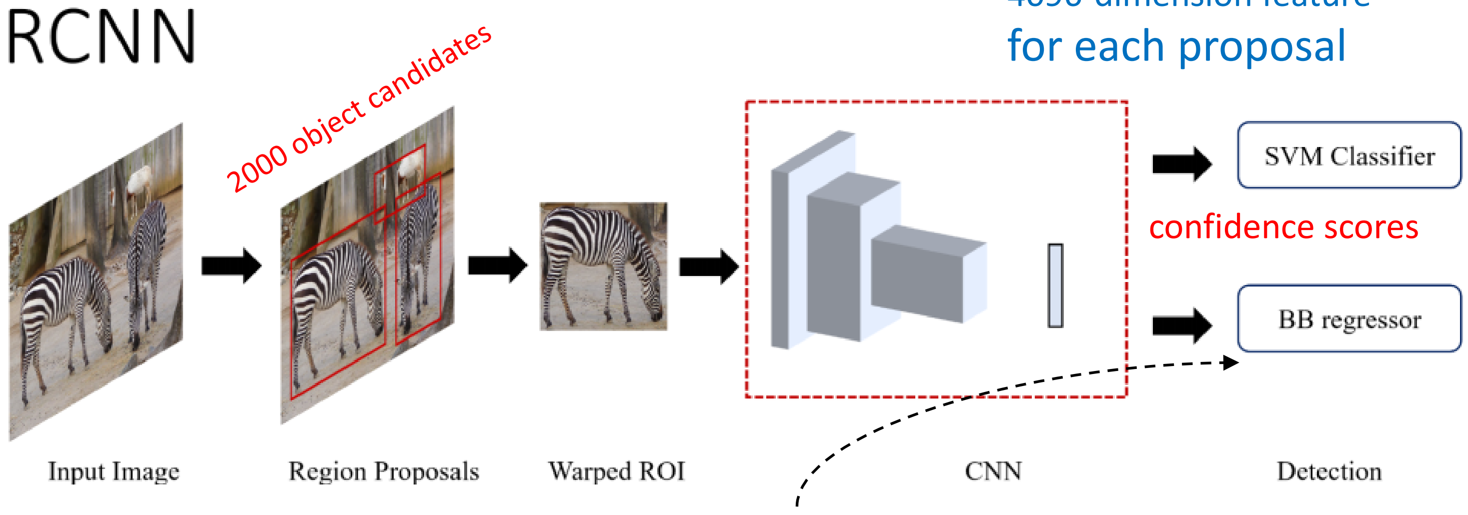
- A network which has a separate module to generate region proposals is termed as a two-stage detector.
- These models try to find an arbitrary number of objects proposals in an image during the first stage and then classify and localize them in the second.
- As these systems have two separate steps, they generally take longer to generate proposals, have complicated architecture and lacks global context.

Region-based Convolutional Neural Network (RCNN)

*R. Girshick 2014

A mean-subtracted input image is first passed through the region proposal module, which produces 2000 object candidates.
This module find parts of the image which has a higher probability of finding an object using Selective Search.
These candidates are then warped and propagated through a CNN network, which extracts a 4096-dimension feature vector for each proposal.
Girshick et al. used AlexNet as the backbone architecture of the detector.
The feature vectors are then passed to the trained, class-specific Support Vector Machines (SVMs) to obtain confidence scores.
Non-maximum suppression (NMS) is later applied to the scored regions, based on its IoU and class.
Once the class has been identified, the algorithm predicts its bounding box using a trained bounding-box regressor which predicts four parameters i.e., center coordinates of box along with its width and height .

A multistage training process



2000 object candidates

4096-dimension feature
for each proposal

confidence scores

Bounding-Box regressor
four parameters i.e., center coordinates of box along with its width and height

Region Proposal Networks abbreviated as RPN.
To generate these so called “proposals” for the region where the object lies, a small network is slide over a convolutional feature map that is the output by the last convolutional layer.
This module find parts of the image which has a higher probability of finding an object using Selective Search.

SPP-Net (SPP-Net is considerably faster than the R-CNN model with comparable accuracy)

*K. He , 2015

SPP(Spatial Pyramid Pooling)-net only shifted the convolution layers of CNN before the region proposal module and added a pooling layer, thereby making the network independent of size/aspect ratio and reducing the computations.

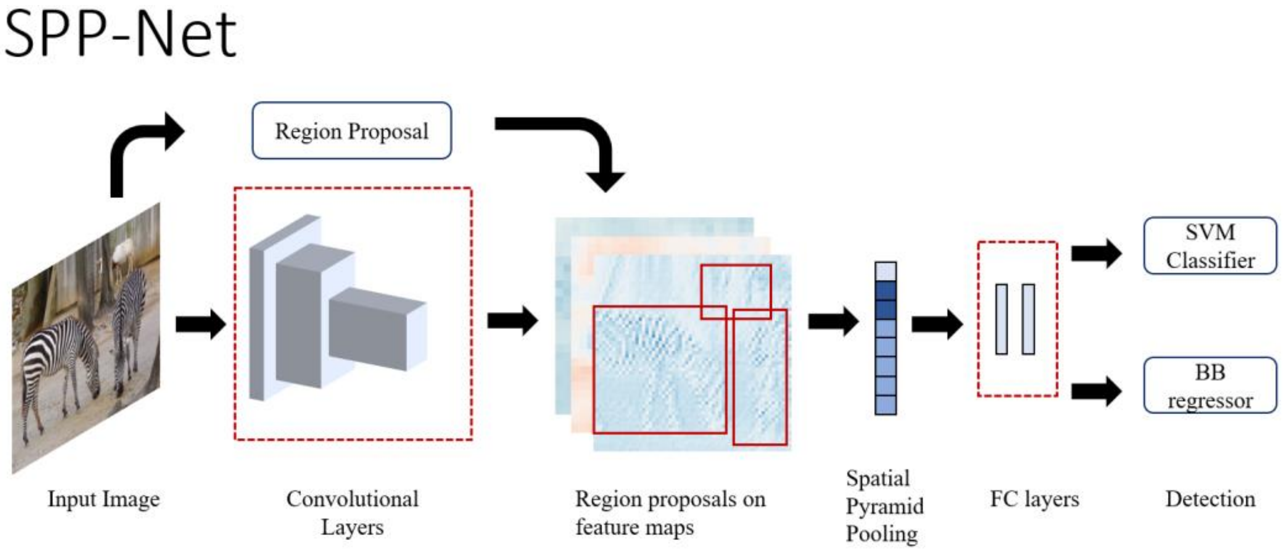
The selective search algorithm is used to generate candidate windows.

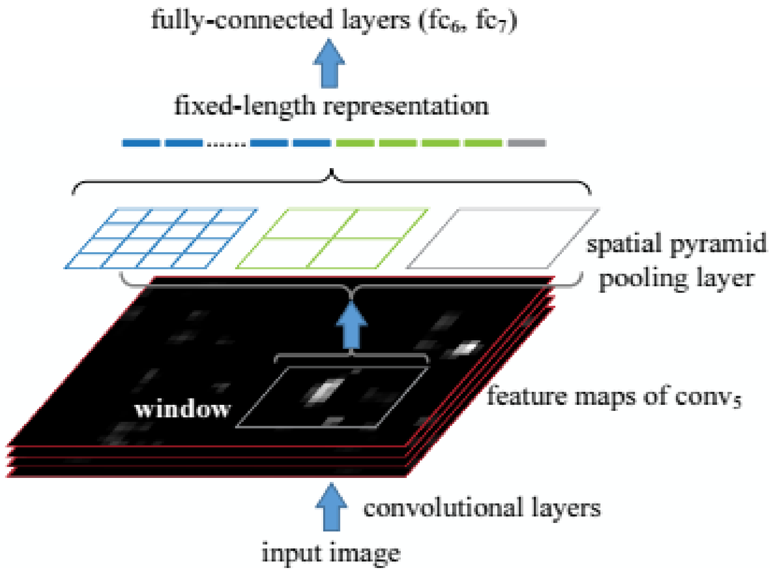
Feature maps are obtained by passing the input image through the convolution layers of a ZF-5 network.

The candidate windows are then mapped on to the feature maps, which are subsequently converted into fixed length representations by spatial bins of a pyramidal pooling layer.

This vector is passed to the fully connected layer and ultimately, to SVM classifiers to predict class and score.

Similar to R-CNN, SPP-net has as post processing layer to improve localization by bounding box regression.





fully-connected layers (fc_6, fc_7)
fixed-length representation
spatial pyramid pooling layer
window
feature maps of $conv_5$
convolutional layers
input image

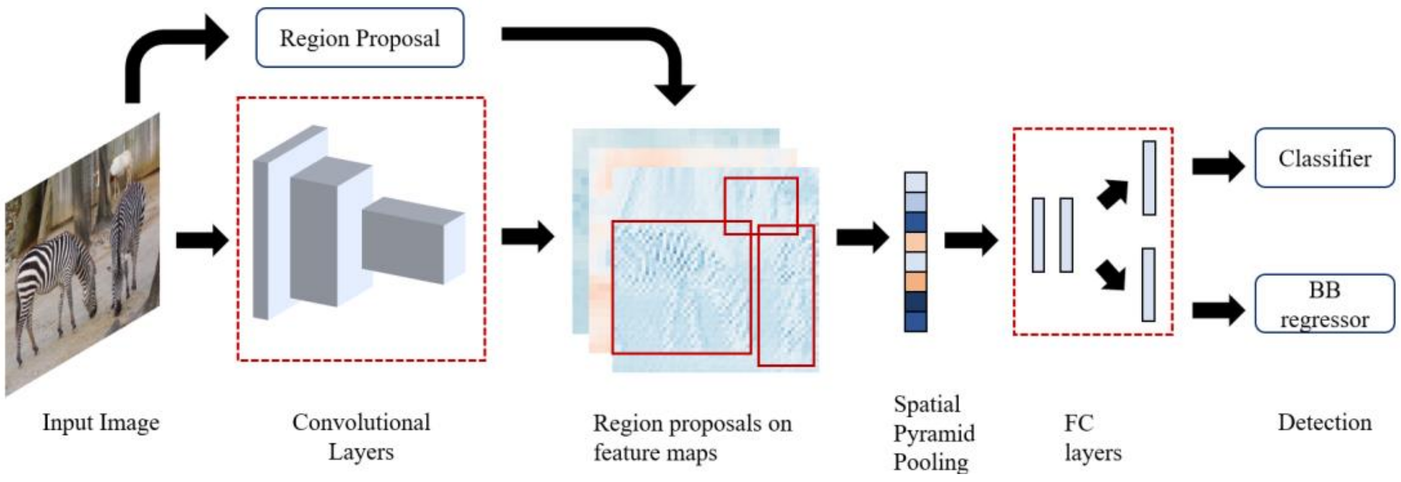
Fast RCNN

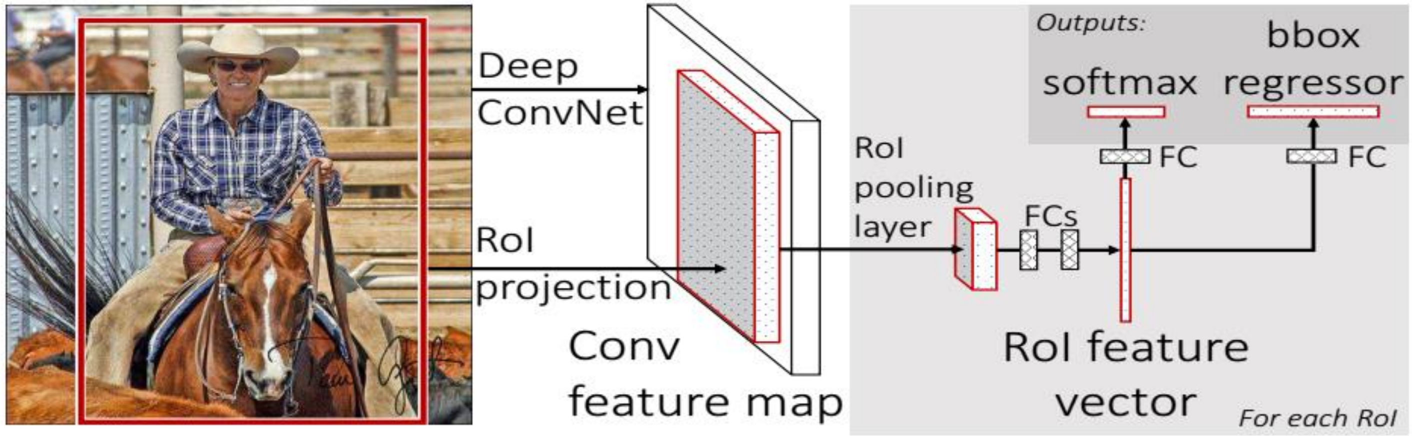
*R. Girshick 2015

- R-CNN/SPPNet need to train multiple systems separately.
- Fast R-CNN solved this by creating a single end-to-end trainable system.
- The network takes as input an image and its object proposals.
- The image is passed through a set of convolution layers and the object proposals are mapped to the obtained feature maps.
- Girshick replaced pyramidal structure of pooling layers from SPP-net with a single spatial bin, called RoI (Region of interest) pooling layer.
- The RoI pooling layer is a special case of the SPP layer, which has only one pyramid level.
- This layer is connected to 2 fully connected layer and then branches out into a *N+1*-class SoftMax layer and a bounding box regressor layer, which has a fully connected layer as well.
- The model also changed the loss function of bounding box regressor from L2 to smooth L1 to better performance, while introducing a multi-task loss to train the network.

Fast RCNN

a multi-task loss to train the network





It simplified training procedure, removed pyramidal pooling and introduces a new loss function. The object detector, without the region proposal network, reported near real time speed with considerable accuracy

Faster RCNN

* S. Ren..2016

Faster RCNN takes an arbitrary input image and outputs a set of candidate windows.

Each such window has an associated *objectness score* which determines likelihood of an object.

Unlike its predecessors which used image pyramids to solve size variance of objects, RPN introduces Anchor boxes.

It used multiple bounding boxes of different aspect ratios and regressed over them to localize object.

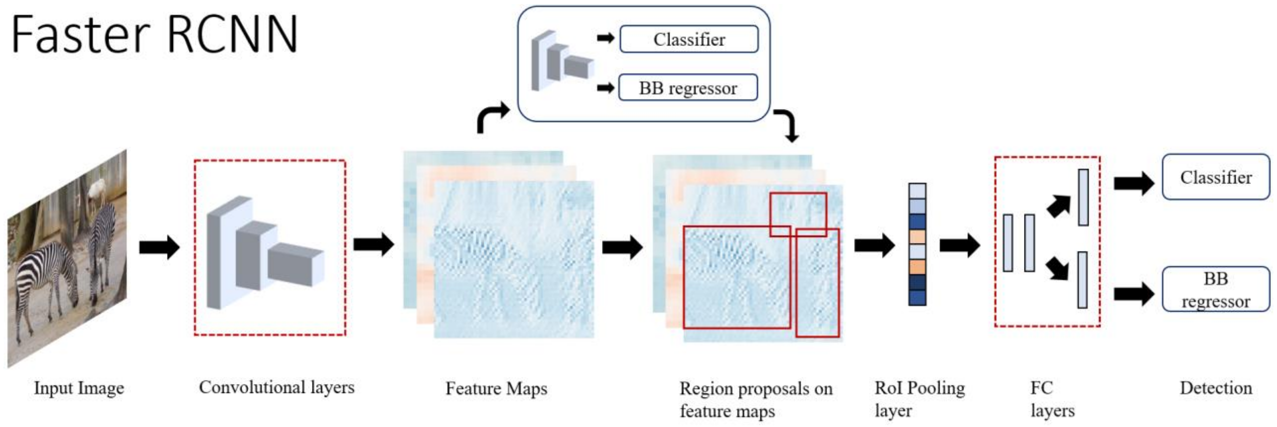
The input image is first passed through the CNN to obtain a set of feature maps.

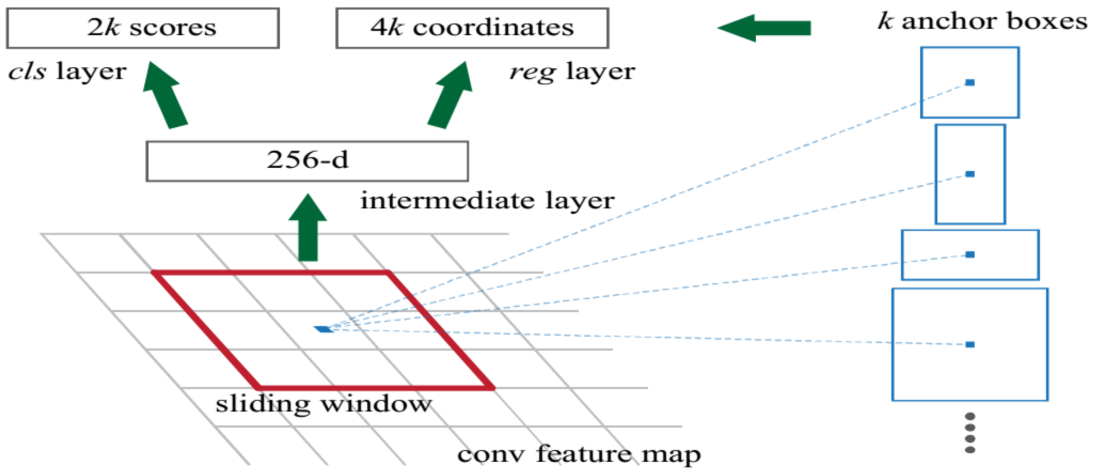
These are forwarded to the RPN, which produces bounding boxes and their classification.

Selected proposals are then mapped back to the feature maps obtained from previous CNN layer in RoI pooling layer, and ultimately fed to fully connected layer, which is sent to classifier and bounding box regressor.

Faster R-CNN is essentially Fast R-CNN with RPN as region proposal module.

Faster R-CNN improved the detection accuracy over the previous state-of-art by more than 3% . It fixed the bottleneck of slow region proposal and ran in near real time at 5 frames per second.

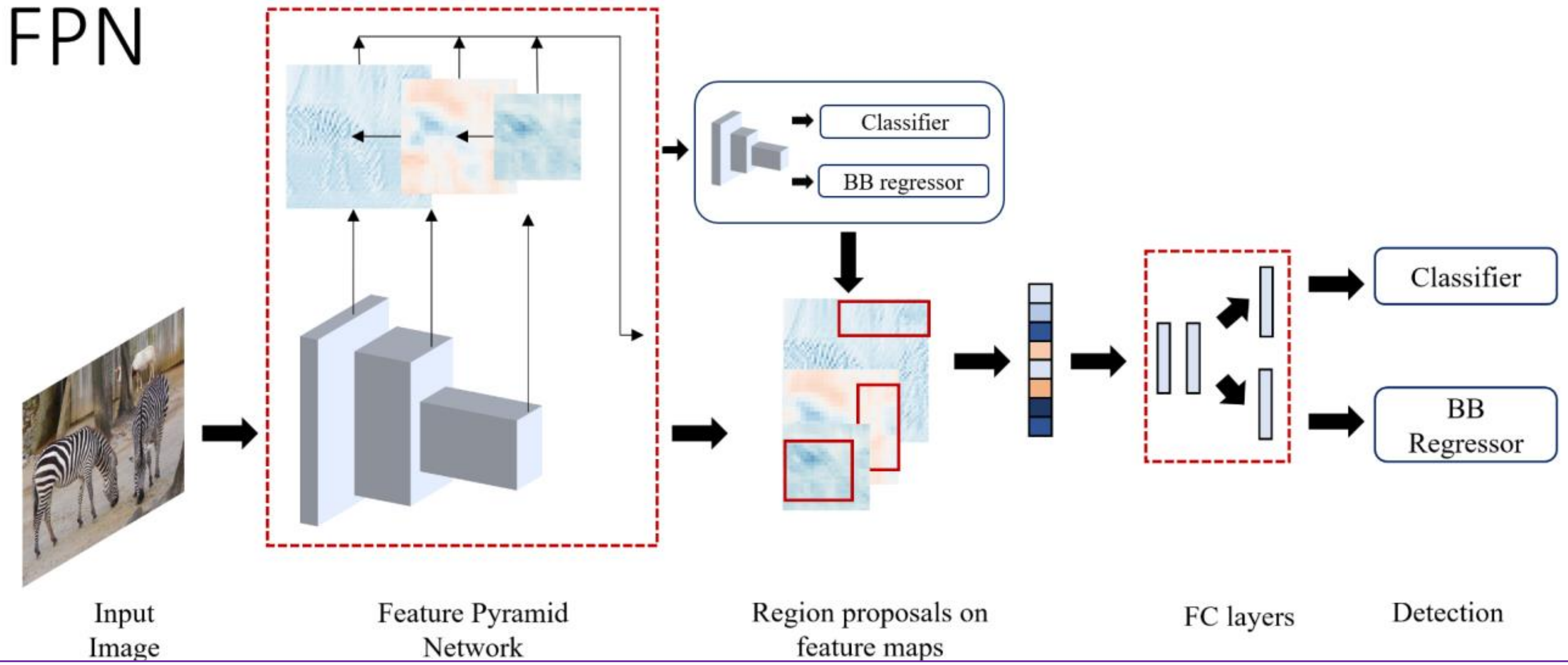




The RPN in Faster R-CNN K predefined anchor boxes are convoluted with each sliding window to produce fixed-length vectors which are taken by cls and reg layer to obtain corresponding outputs

FPN(Feature Pyramid Network)

*T.-Y. Lin... 2017



FPN has a top-down architecture with lateral connections to build high-level semantic features at different scales.

The FPN has two pathways, a bottom-up pathway which is a ConvNet computing feature hierarchy at several scales and a top-down pathway which up samples coarse feature maps from higher level into high resolution features.

These pathways are connected by lateral connection by a *1x1* convolution operation to enhance the semantic information in the features.

FPN is used as a region proposal network (RPN) of a ResNet-101 based Faster R-CNN here.

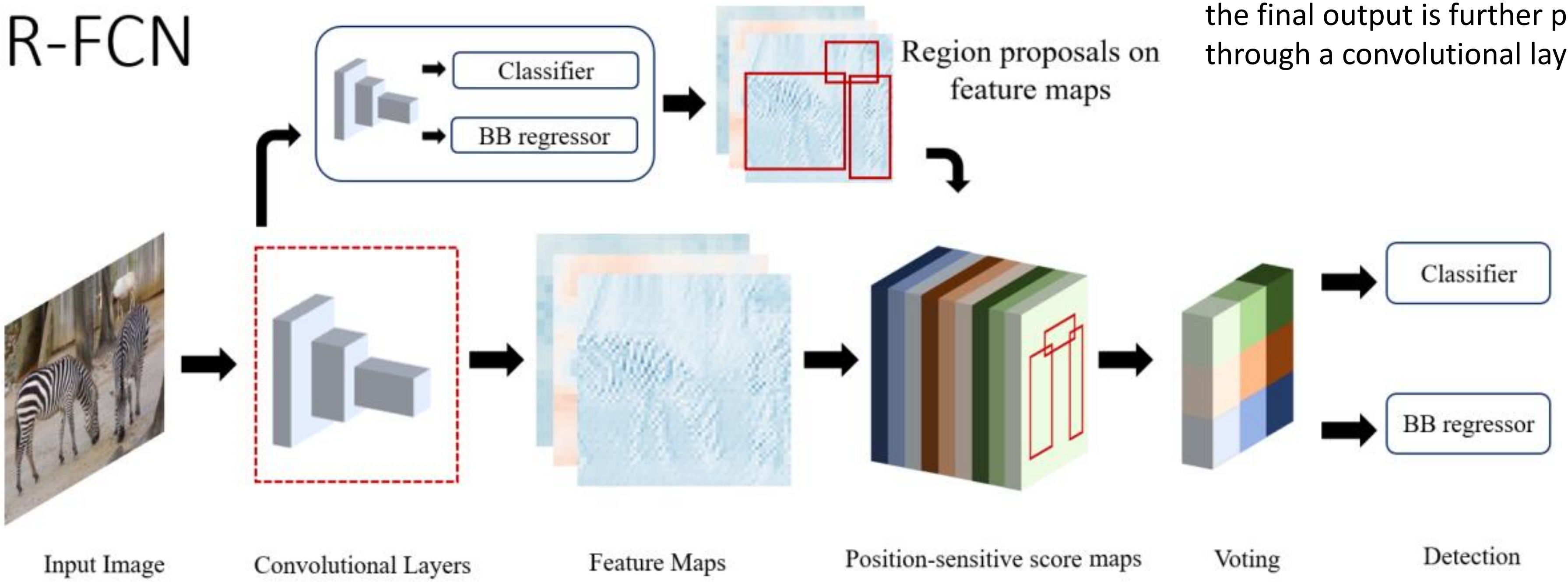
FPN could provide high-level semantics at all scales, which reduced the error rate in detection.

It became a standard building block in future detections models and improved accuracy their accuracy across the table. It also lead to development of other improved networks

RFCN(Region-based Fully Convolutional Network)

* J. Dai..2016

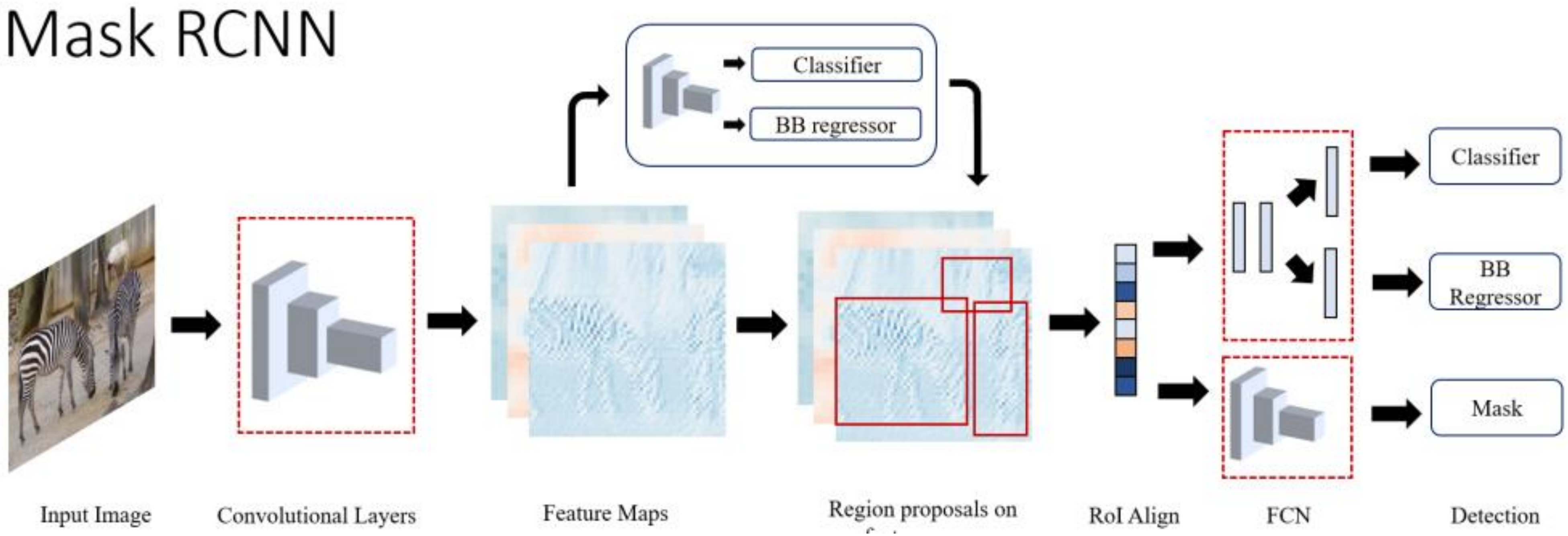
the final output is further processed through a convolutional layer



R-FCN detector is a combination of four convolutional networks.
The input image is first passed through the ResNet-101 to get feature maps.
An intermediate output (Conv4 layer) is passed to a Region Proposal Network (RPN) to identify RoI proposals. while the final output is further processed through a convolutional layer and is input to classifier and regressor.
The classification layer combines the generated the position-sensitive map with the RoI proposals to generate predictions while the regression network outputs the bounding box details.
R-FCN is trained in a similar 4 step fashion as Faster-RCNN [44] whilst using a combined cross-entropy and box regression loss.

Mask RCNN

* K. He...2018



Mask R-CNN extends on the Faster R-CNN by adding another branch in parallel for pixel-level object instance segmentation.

The branch is a fully connected network applied on RoIs to classify each pixel into segments with little overall computation cost.

It uses similar basic Faster R-CNN architecture for object proposal, but adds a mask head parallel to classification and bounding box regressor head.

The authors chose the ResNeXt-101 as its backbone along with the feature Pyramid Network (FPN) for better accuracy and speed.

The loss function of Faster R-CNN is updated with the mask loss and as in FPN, it uses 5 anchor boxes with 3 aspect ratio.

Overall training of Mask R-CNN is similar to faster R-CNN

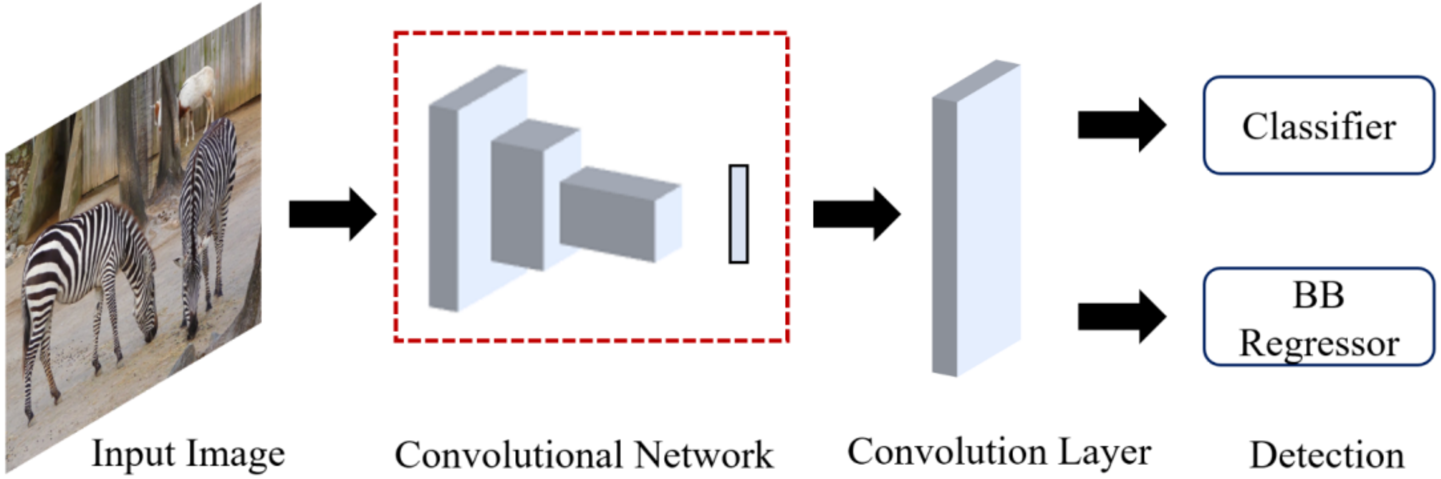
3. One stage object detectors

1. YOLO
2. SSD
3. YLOLO2, YOLO9000
4. Retina Net
5. YOLOv3
6. Center Net
7. EfficientDet
8. YOLOv4
9. Swin Transformer
10. YOLOx

- Single-stage detectors classify and localize semantic objects in a single shot using dense sampling.
- They use predefined boxes/keypoints of various scale and aspect ratio to localize objects.
- It edges two-stage detectors in real-time performance and simpler design.

YOLO (You Onl YOLO

* J. Redmon ...2016



YOLO reframed it as a regression problem, directly predicting the image pixels as objects and its bounding box attributes.

In YOLO, the input image is divided into a *S x S* grid and the cell where the object's center falls is responsible for detecting it.

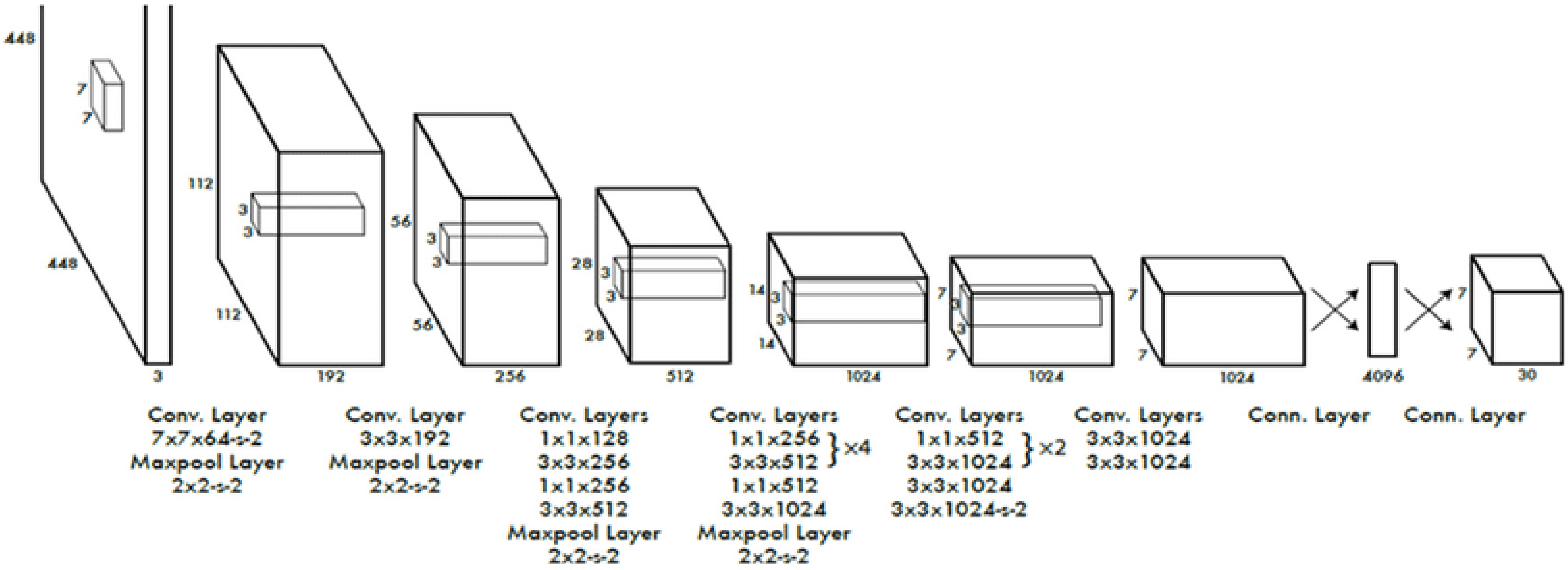
A grid cell predicts multiple bounding boxes, and each prediction array consists of 5 elements: center of bounding box – x and y, dimensions of the box – w and h, and the confidence score.

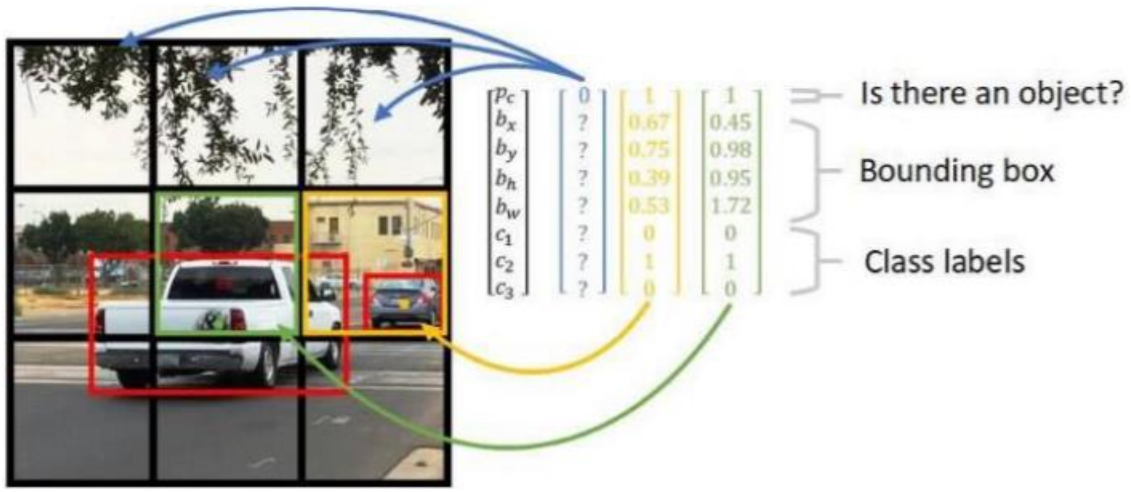
YOLO was inspired from the GoogLeNet model for image classification, which uses cascaded modules of smaller convolution networks.

It is pre-trained on ImageNet data till the model achieves high accuracy and then modified by adding randomly initialized convolution and fully connected layers.

At training time, grid cells predict only one class as it converges better, but it is be increased during the inference time.

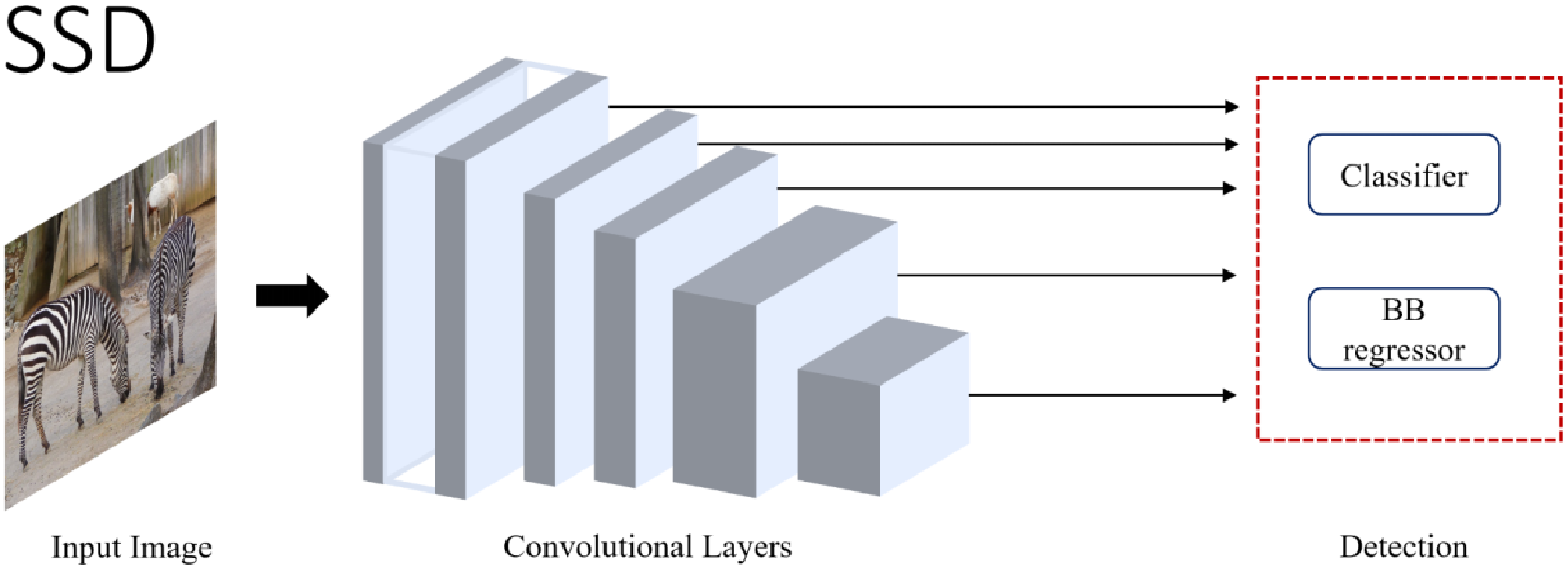
Multitask loss, combined loss of all predicted components, is used to optimize the model.


448
7
7
448
3
112
3
3
112
192
56
3
3
56
256
28
3
3
28
512
14
3
3
14
1024
7
3
3
7
1024
7
7
1024
4096
7
7
30
Conv. Layer
7x7x64-s-2
Maxpool Layer
2x2-s-2
Conv. Layer
3x3x192
Maxpool Layer
2x2-s-2
Conv. Layers
1x1x128
3x3x256
1x1x256
3x3x512
Maxpool Layer
2x2-s-2
Conv. Layers
1x1x256
3x3x512
}×4
1x1x512
3x3x1024
Maxpool Layer
2x2-s-2
Conv. Layers
1x1x512
3x3x1024
}×2
3x3x1024
3x3x1024-s-2
Conv. Layers
3x3x1024
3x3x1024
Conn. Layer
Conn. Layer


p_c b_x b_y b_h b_w c_1 c_2 c_3
0 ? ? ? ? ? ? ?
1 0.67 0.75 0.39 0.53 0 1 0
1 0.45 0.98 0.95 1.72 0 1 0
Is there an object?
Bounding box
Class labels

SSD(SingleShot Detector)

*W. Liu, 2016



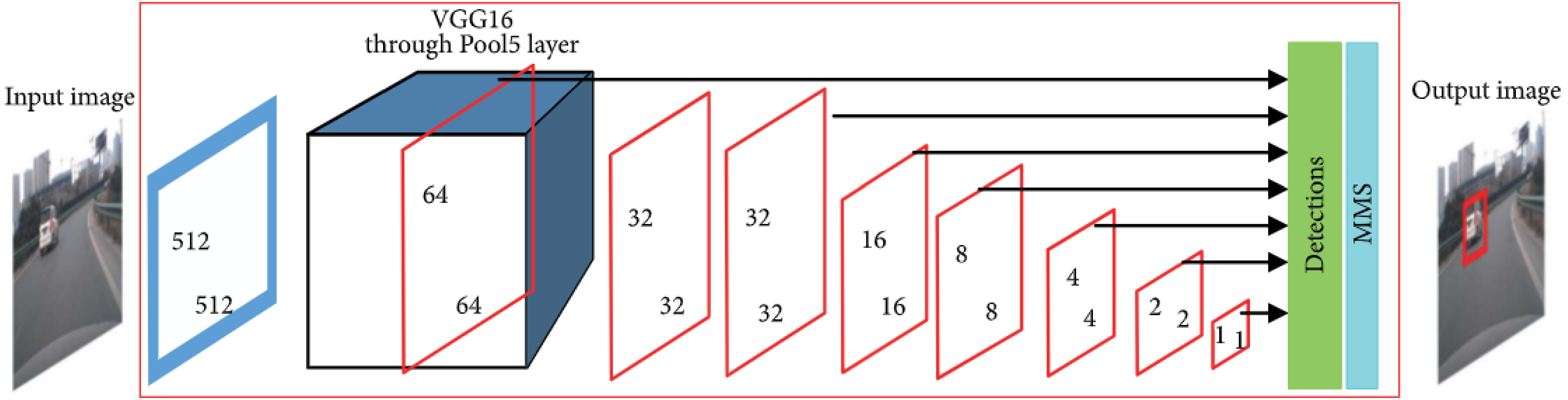
SSD was the first single stage detector that matched accuracy of contemporary two stage detectors like Faster R-CNN [44], while maintaining real time speed.

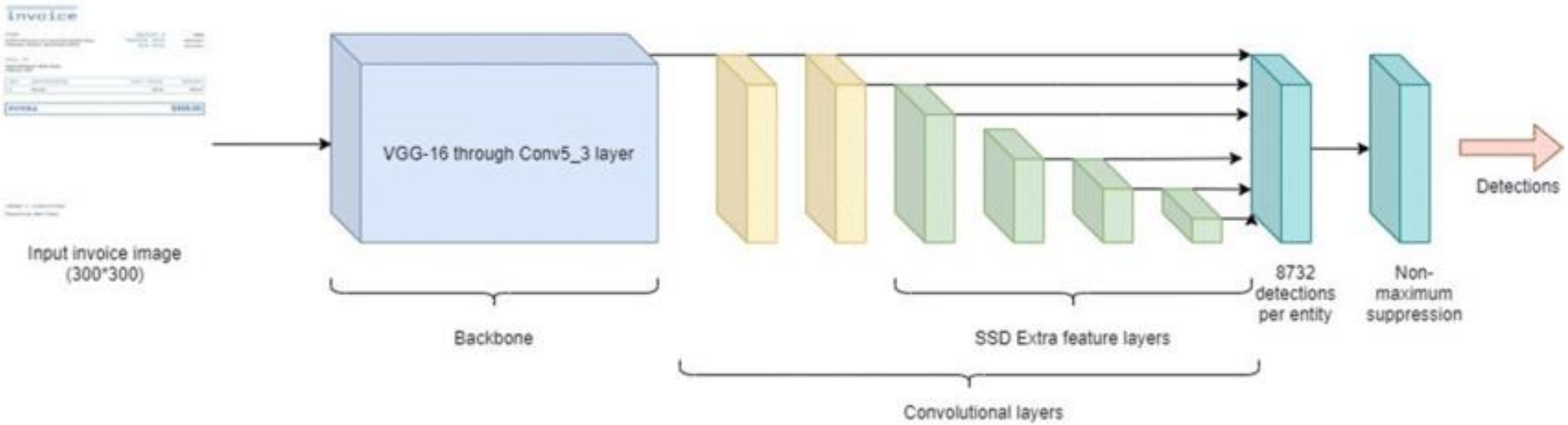
SSD was built on VGG-16, with additional auxiliary structures to improve performance.

These auxiliary convolution layers, added to the end of the model, decrease progressively in size. SSD detects smaller objects earlier in the network when the image features are not too crude, while the deeper layers were responsible for offset of the default boxes and aspect ratios.

Even though SSD was significantly faster and more accurate than both state-of-art networks like YOLO and Faster R-CNN, it had difficulty in detecting small objects.

This issue was later solved by using better backbone architectures like ResNet and other small fixes

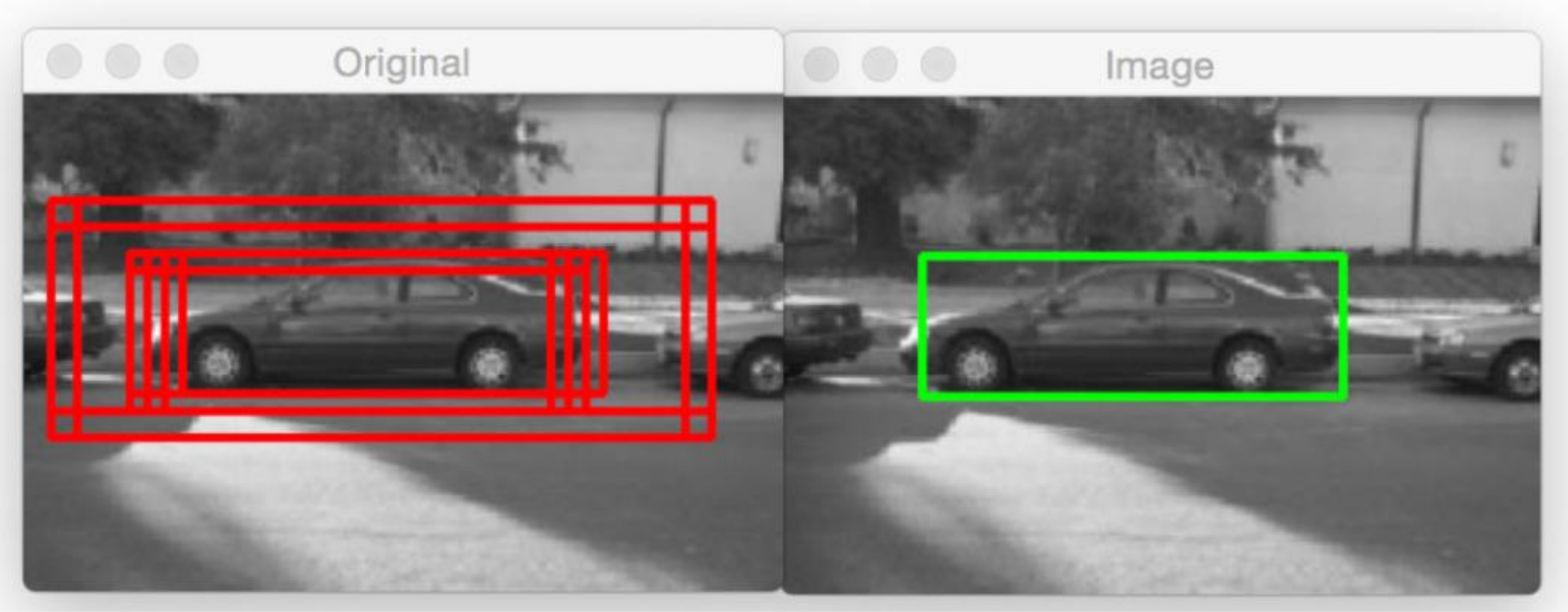

VGG16
through Pool5 layer
Input image
512
512
64
64
32
32
32
32
16
16
8
8
4
4
2
2
1
1
Detections
MMS
Output image


invoice
Input invoice image
(300*300)
VGG-16 through Conv5_3 layer
Backbone
SSD Extra feature layers
Convolutional layers
8732
detections
per entity
Non-
maximum
suppression
Detections

NMS: Non-maximum Suppression

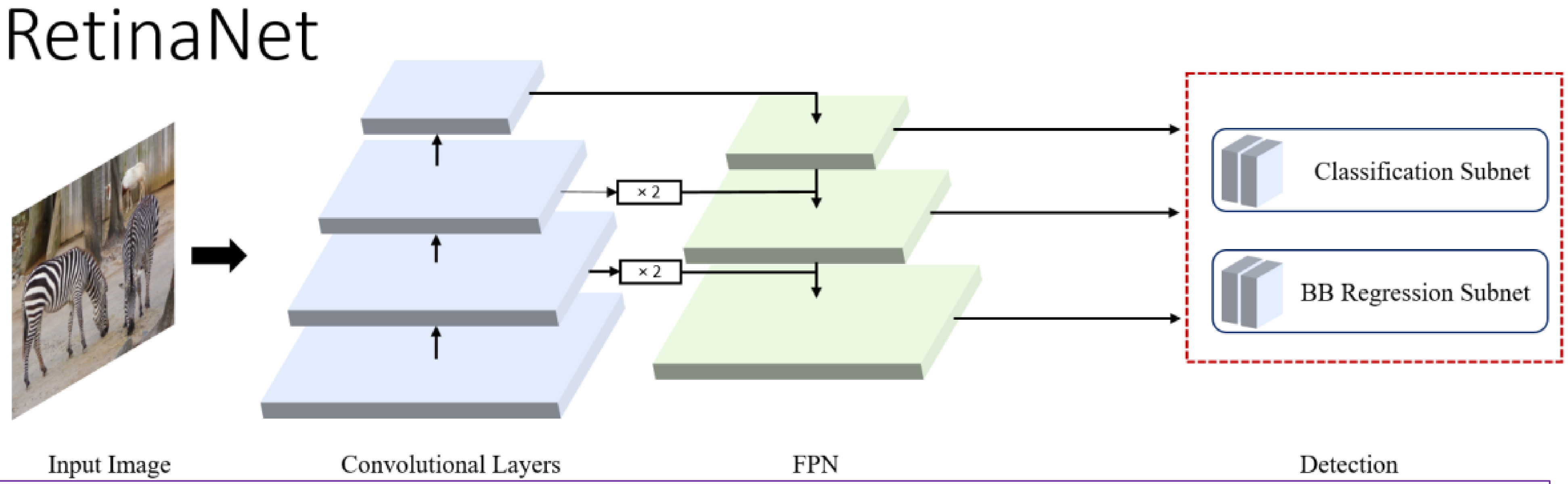
Non max suppression is **a technique used mainly in object detection that aims at selecting the best bounding box out of a set of overlapping boxes**.

The first step in NMS is to remove all the predicted bounding boxes that have a detection probability that is less than a given NMS threshold.



RetinaNet

*Lin 2020



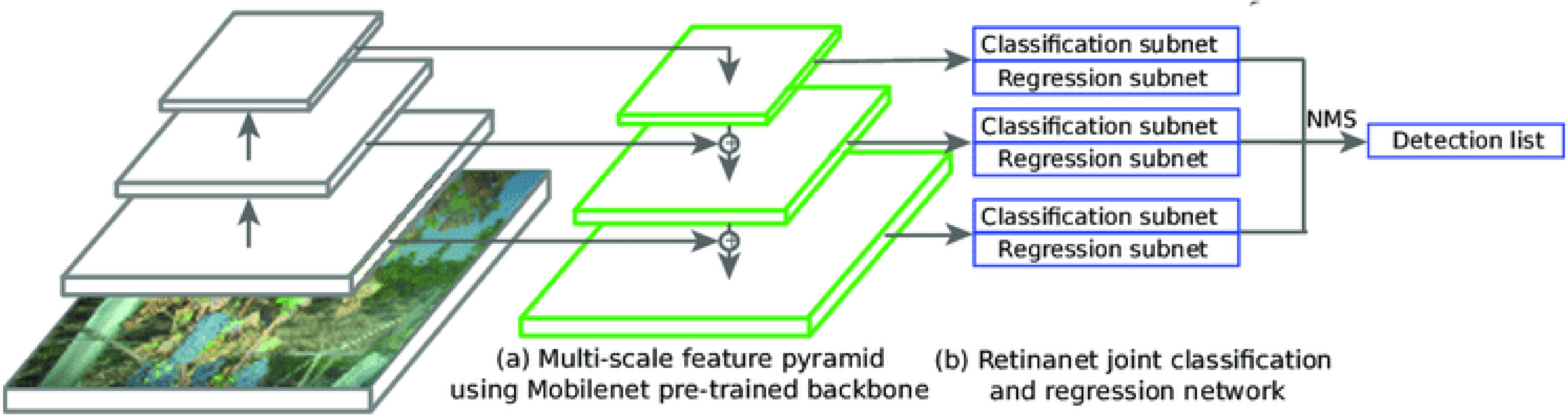
Given the difference between the accuracies of single and two stage detectors, Lin et al. suggested that the reason single stage detectors lag is the “extreme foreground-background class imbalance”

They proposed a reshaped cross entropy loss, called Focal loss as the means to remedy the imbalance.

Focal loss parameter reduces the loss contribution from easy examples.

The authors demonstrate its efficacy with the help of a simple, single stage detector, called RetinaNet, which predicts objects by dense sampling of the input image in location, scale and aspect ratio.

It uses ResNet augmented by Feature Pyramid Network (FPN) as the backbone and two similar subnets - classification and bounding box regressor.

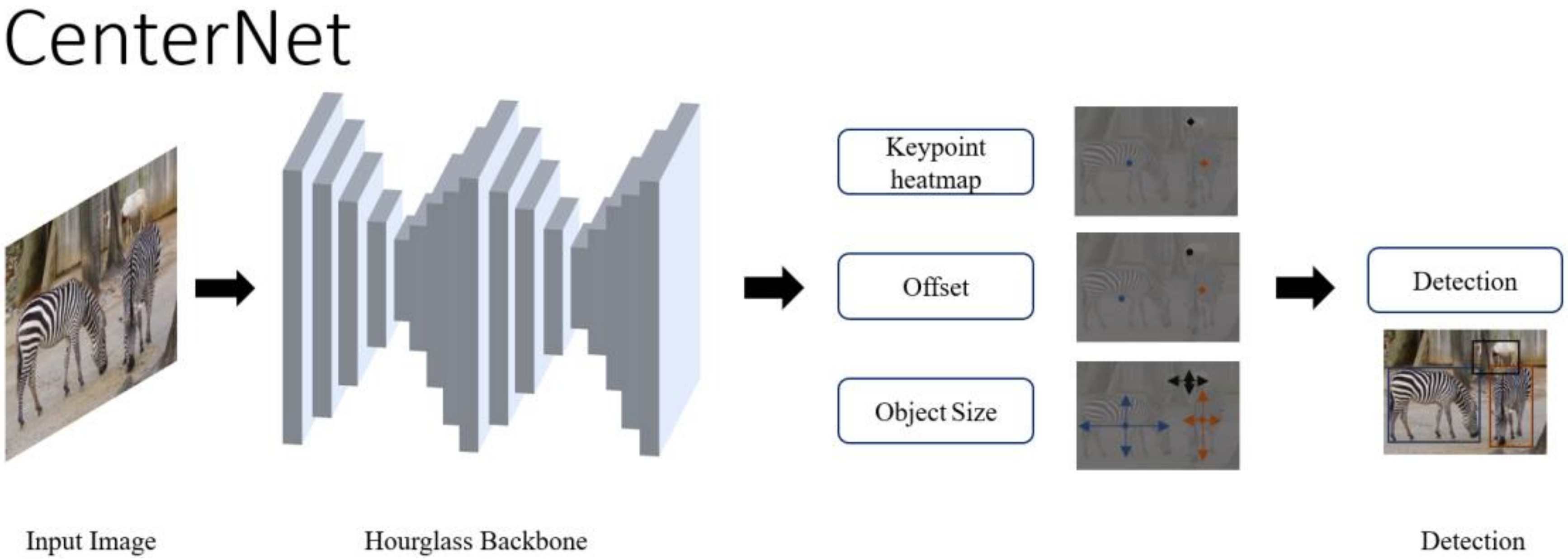


(a) Multi-scale feature pyramid using Mobilenet pre-trained backbone

(b) Retinanet joint classification and regression network

CenterNet

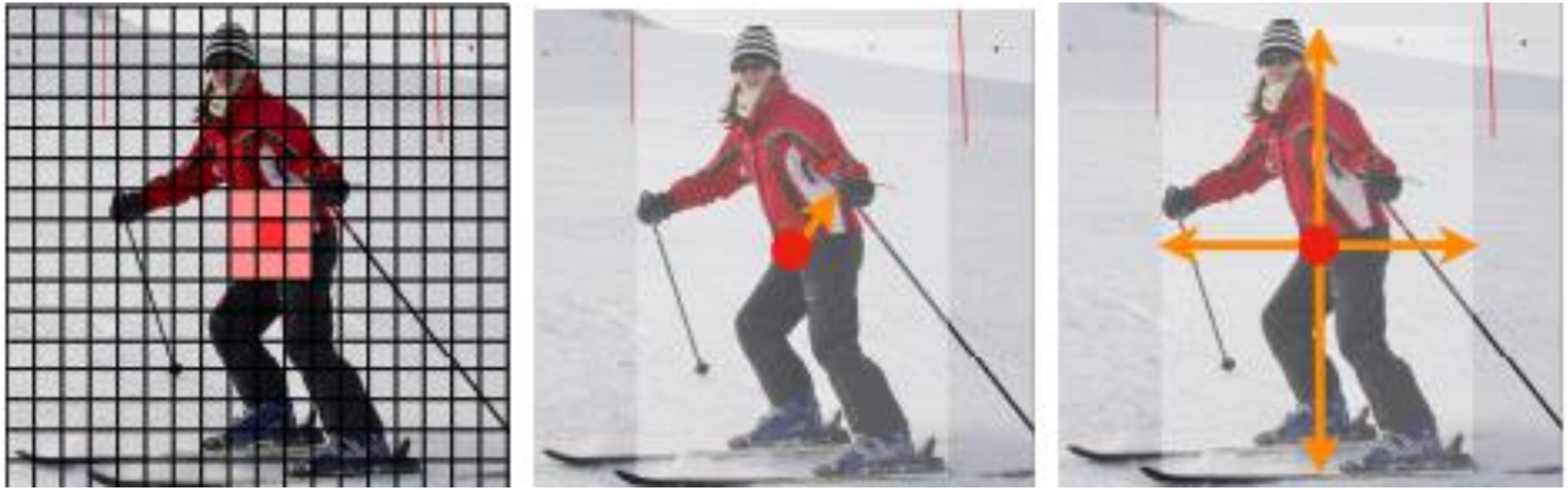
X. Zhou…"Objects as points." 2019



Zhou et al. in [68] takes a very different approach of modelling objects as points, instead of the conventional bounding box representation, CenterNet predicts the object as a single point at the center of the bounding box.

The input image is passed through the FCN that generates a heatmap, whose peaks correspond to center of detected object.

It uses a ImageNet pretrained stacked Hourglass-101 as the feature extractor network and has 3 heads – heatmap head to determine the object center, dimension head to estimate size of object and offset head to correct offset of object point (overlap)



Center Point

Offset Point

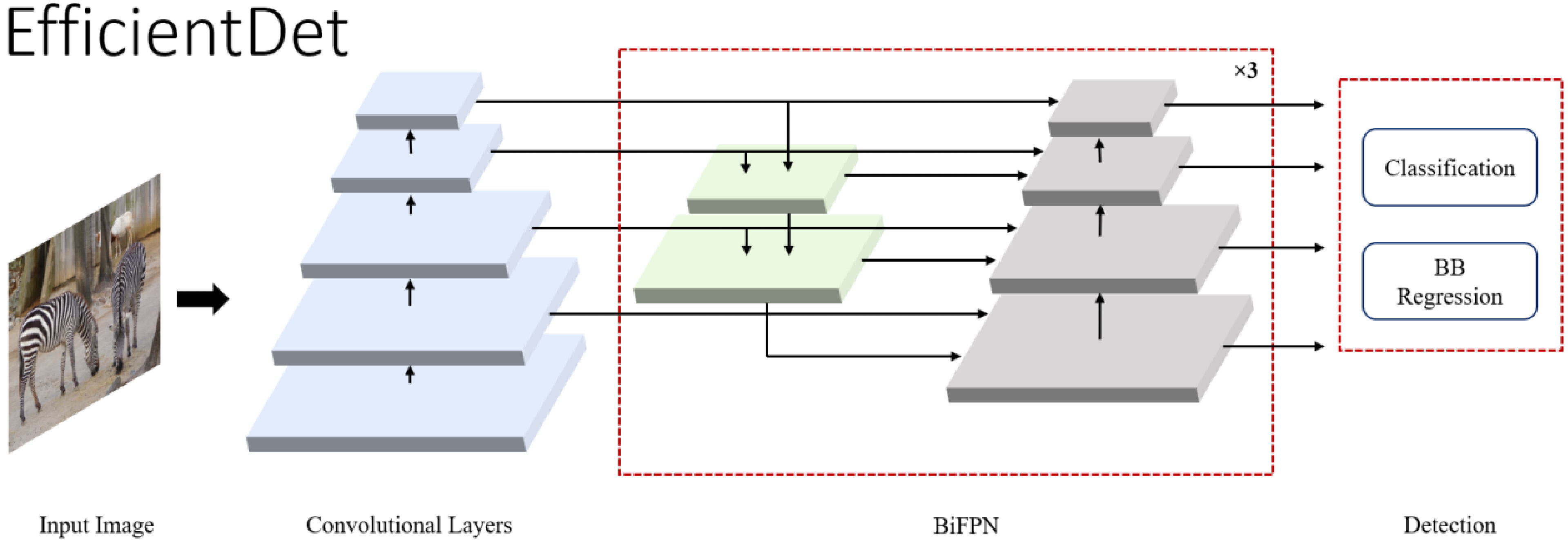
Rectangle

Offset Head

- This head is used to recover from the discretization error caused due to the downsampling of the input.
- After the prediction of the center points, we have to map these coordinates to a higher dimensional input image.
- This will cause a value disturbance as the original image pixel indices are integers and we will be predicting the float values.
- So to solve this issue they predict the local offsets O_hat. These local offset values are shared between objects present in an image.

EfficientDet

*M. Tan 2020



EfficientDet utilizes EfficientNet as the backbone network with multiple sets of BiFPN layers stacked in series as feature extraction network.
it builds towards the idea of scalable detector with higher accuracy and efficiency. It introduces efficient multi-scale features, BiFPN and model scaling. BiFPN is bi-directional feature pyramid network with learnable weights for cross connection of input features at different scales.

.

Yolox

*Zheng Ge ...2021

We switch the YOLO detector to an anchor-free manner and conduct other advanced detection techniques, i.e., *a decoupled head and the leading label assignment strategy SimOTA to achieve state-of-the-art results across a large scale range of models*

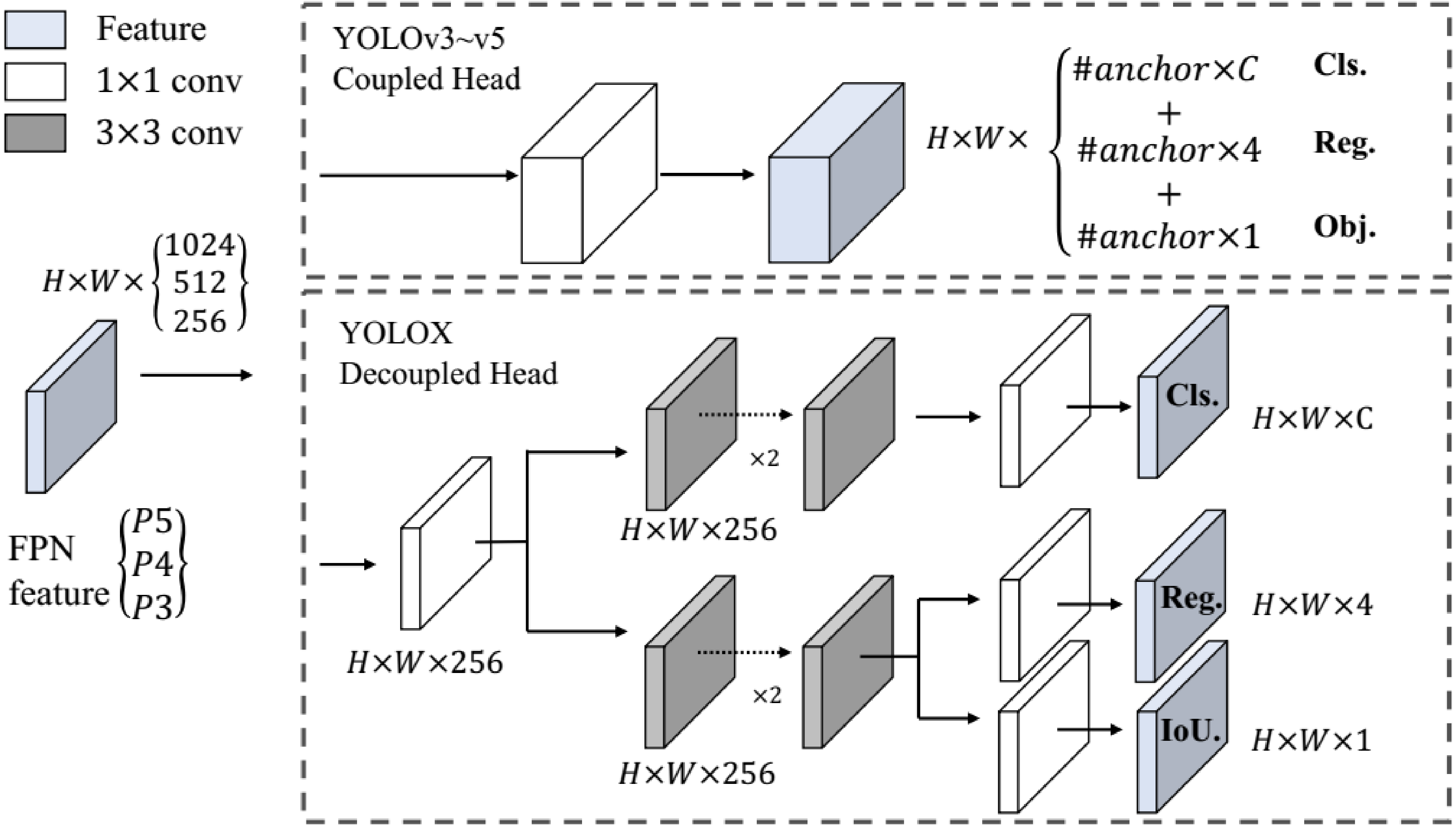
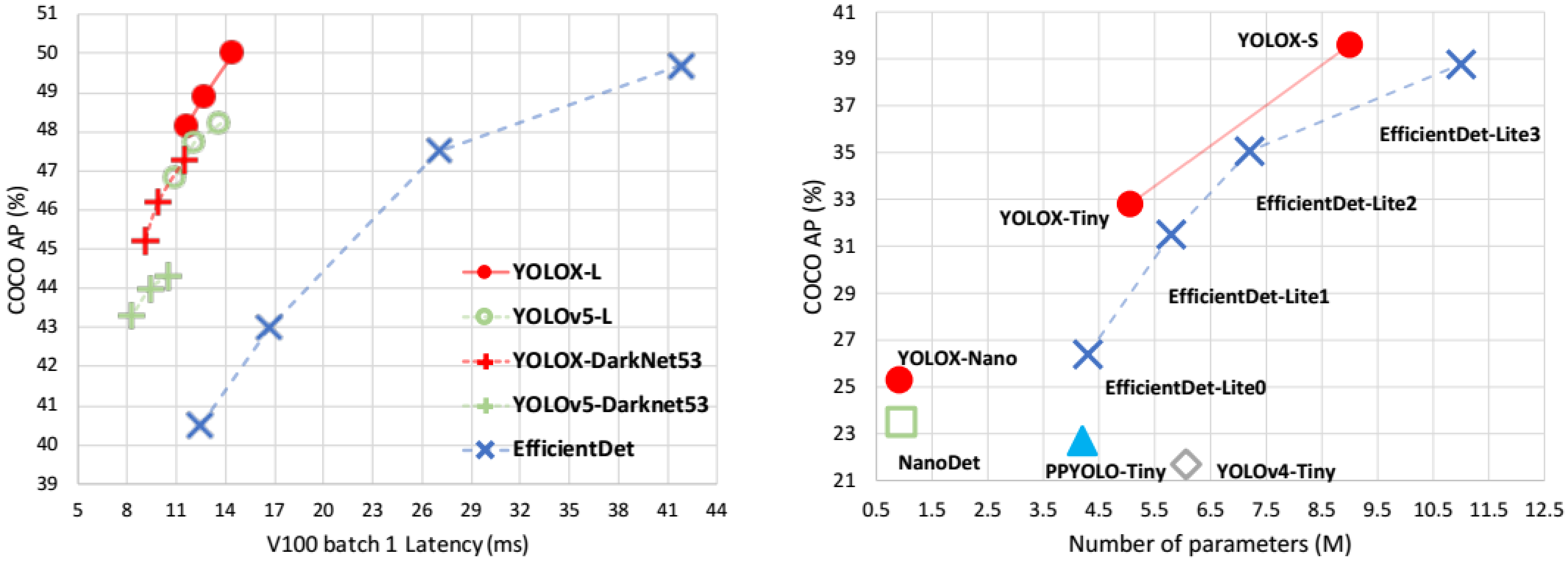
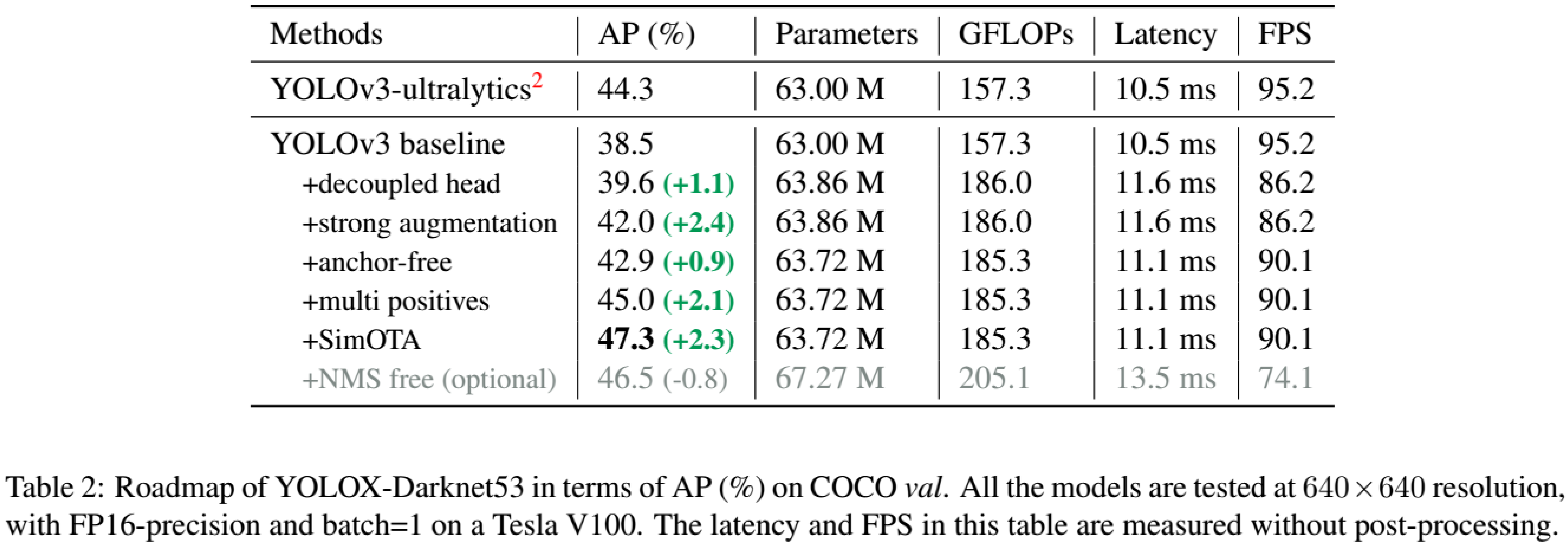


Illustration of the difference between YOLOv3 head and the proposed decoupled head. For each level of FPN feature, we first adopt a 1 × 1 conv layer to reduce the feature channel to 256 and then add two parallel branches with two 3 × 3 conv layers each for classification and regression tasks respectively. IoU branch is added on the regression branch.

YOLOx



Speed-accuracy trade-off of accurate models (top) and Size-accuracy curve of lite models on mobile devices (bottom) for YOLOX and other state-of-the-art object detectors



Methods	AP (%)	Parameters	GFLOPs	Latency	FPS
YOLOv3-ultralytics[2]	44.3	63.00 M	157.3	10.5 ms	95.2
YOLOv3 baseline	38.5	63.00 M	157.3	10.5 ms	95.2
+decoupled head	39.6 **(+1.1)**	63.86 M	186.0	11.6 ms	86.2
+strong augmentation	42.0 **(+2.4)**	63.86 M	186.0	11.6 ms	86.2
+anchor-free	42.9 **(+0.9)**	63.72 M	185.3	11.1 ms	90.1
+multi positives	45.0 **(+2.1)**	63.72 M	185.3	11.1 ms	90.1
+SimOTA	**47.3** **(+2.3)**	63.72 M	185.3	11.1 ms	90.1
+NMS free (optional)	46.5 (-0.8)	67.27 M	205.1	13.5 ms	74.1

Table 2: Roadmap of YOLOX-Darknet53 in terms of AP (%) on COCO *val*. All the models are tested at 640 × 640 resolution, with FP16-precision and batch=1 on a Tesla V100. The latency and FPS in this table are measured without post-processing.

Open Problems in RCNNs

- **AutoML** :The use of automatic neural architecture search (NAS) for determining the characteristics of object detector
- **Lightweight detectors:** While lightweight networks have shown great promise by matching classification errors with the full-fledged models, they still lack in detection accuracy by more than 50%.
- **Weakly supervised/few shot detection:** Most of the state of-the-art object detection models are trained on millions of bounding box annotated data, which is unscalable as annotating data requires time and resources.
- **Domain transfer:** Domain transfer refers to use of a model trained on labeled image of a particular source task on a separate, but related target task. It encourages reuse of trained model and reduces reliance on the availability of a large dataset to achieve high accuracy.
- **3D object detection:** 3D object detection is a particularly critical problem for autonomous driving. Even though models have achieved high accuracy, deployment of anything below human level performance will bring up safety concerns.
- **Object detection in video:** Object detectors are designed to perform on individual image which lack correlation between themselves. Using spatial and temporal relationship between the frames for object recognition is an open problem.

3.4 Layer Wise Design Algorithms_part1

Model Compressing

Model Compressing

In model compressing some layers and Feature Maps(units)

Some Definitions:

Filter Part: The convolution and pooling layers (before the flat layer) which extract features from the spatial or temporal correlated inputs on an image or any other input signal. In addition for the sequenced inputs, If there are one or more than one RNN modules in the filter part, they are considered in the filter part, too.

FC Part: one, two, or more than two fully connected layers which are designed after the "Filter Part" which build an classifier or regression model for the extracted feature space.

Algorithm3:

1- Find and remove the last layers of the "Filter Part" which do not increase the SI(SmI), significantly.

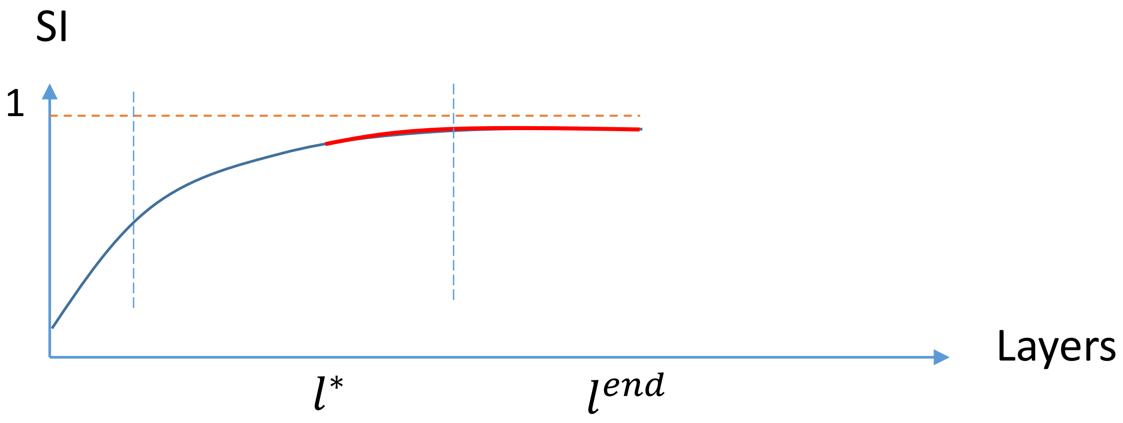
2- Find and remove all feature maps(units) of the last layer of the filter part which do not increase the SI(SmI), significantly.

3- Find and remove some other layers or feature maps form the filter part that by removing them, the separation index at the last layer of the filter part does not drop significantly

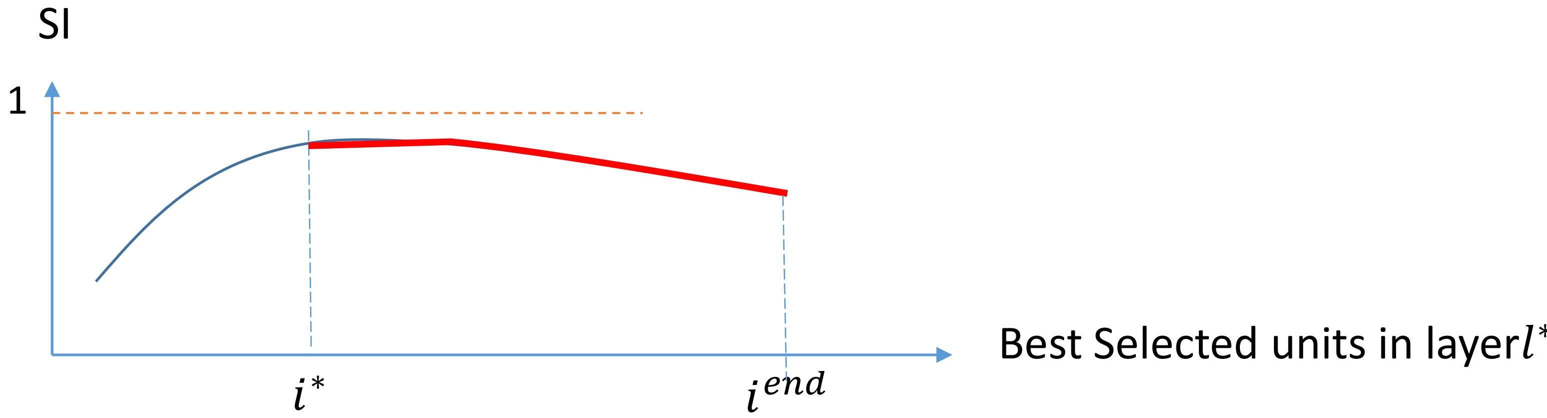
3- Design a new "FC part" with respect the number of units at the last later of the "filter part".

Compressing Procedure

1.Compute SI(SmI) for layers of the DNN and then remove all last layers (after layer l^*) which do no increase SI(SmI), significantly.



2.Remove all extra feature maps(units) at layer l^* which do no increase SI(SmI), significantly.



3. Flat the units at layer l^* and add Fully Connoted layers and fine tune

TABLE VIII
Comparison with the prior art of VGG-16 on CIFAR-10.

Compressing method	**Retraining needed**	**FLOPs**	**Pruned**	**Accuracy**
VGG16-base	-	313.7M (0.0%)	0.0%	94.04%
PFEC **[17]**	Yes	206M (34.3%)	63.3%	93.40%
VP **[45]**	Yes	190M (39.1%)	73.4%	93.18%
SS **[46]**	Yes	183.13M (41.6%)	۷۳,۲%	۹۳,۰۲%
GAL-0.05 **[47]**	No	189.49M (39.6%)	77.2%	92.03%
CP **[48]**	No	107.58M (65.1%)	77.6%	92.03%
HRFM **[49]**	Yes	108.61M (65.3%)	82.1%	92.34%
GAL-0.1 **[47]**	No	171.89M (45.2%)	82.2%	90.73%
Our method	**No**	**75.6M (76.0%)**	**87.5%**	**93.49%**
HRFM **[49]**	Yes	73.70M (76.5%)	88.2%	91.23%

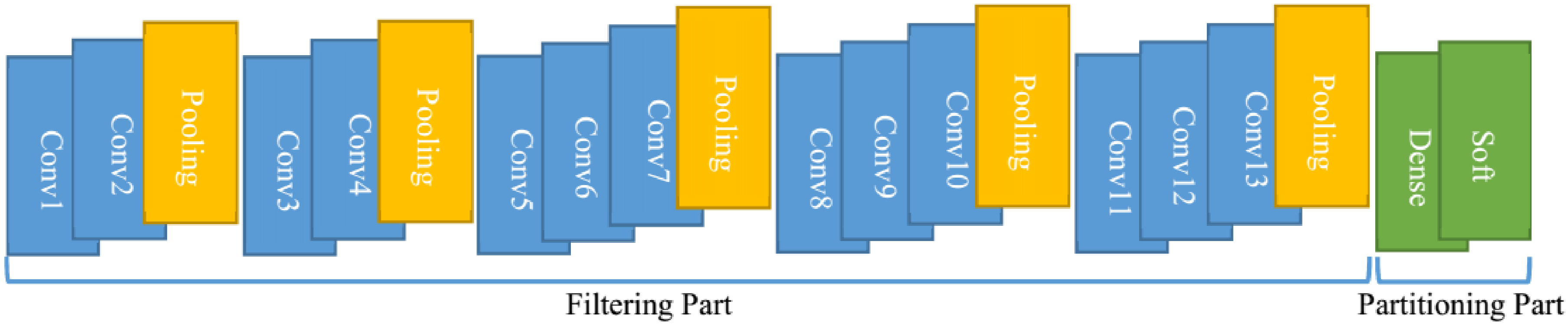


Fig. 5. The architecture of the "VGG-16" network.

TABLE IX
Comparison with the prior art of GoogLeNet on CIFAR-10.

Compressing method	**Retraining needed**	**FLOPS**	**Pruned**	**Accuracy**
GoogleNet-base	-	1.52B(0.0%)	0.0%	95.05%
PFEC **[17]**	Yes	1.0B(32.9%)	42.9%	94.54%
HRFM **[49]**	Yes	0.69B(54.9%)	55.4%	94.53%
GAL-Apo **[50]**	No	0.76B(50.0%)	53.7%	92.11%
GAL-0.05 **[48]**	No	0.94B(38.2%)	49.3%	93.93%
HRFM **[49]**	Yes	0.45B(70.4%)	69.8%	94.07%
Our method	**No**	**0.39B(74.4%)**	**77.6%**	**95.00%**

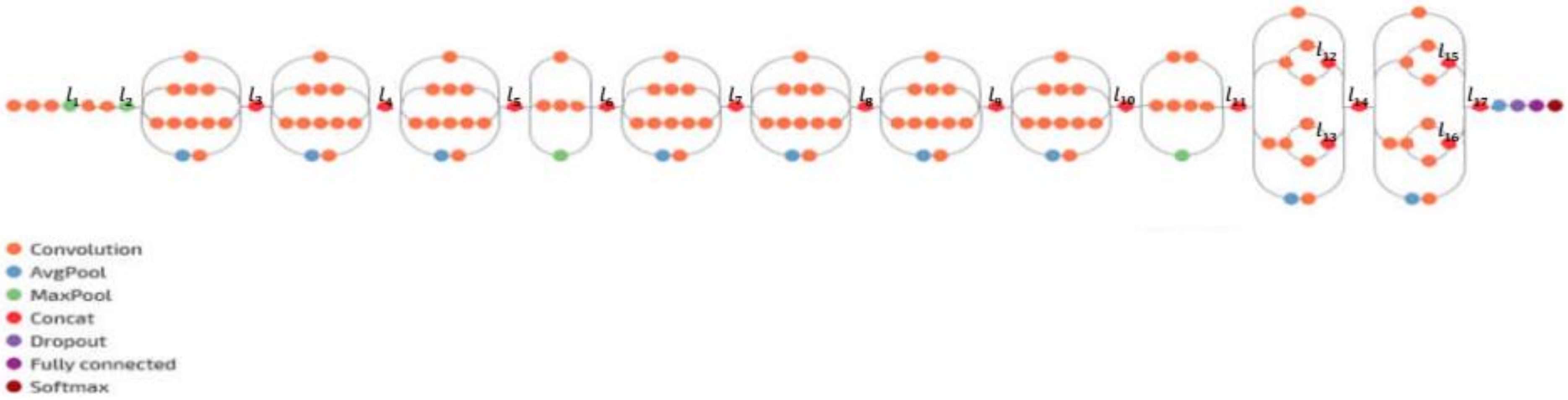
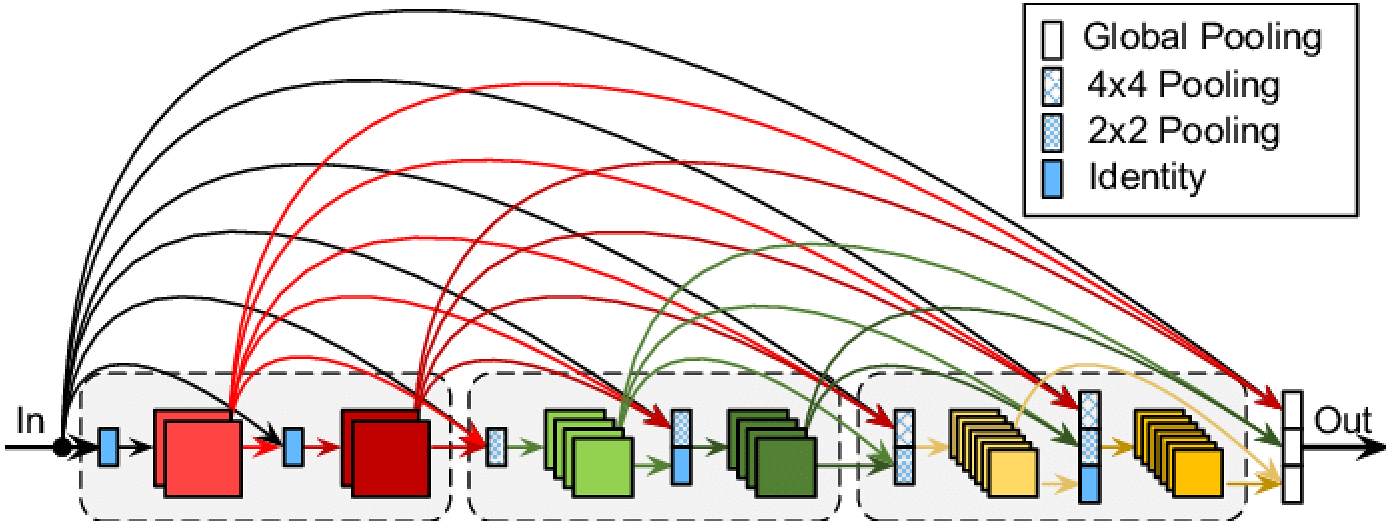


Fig. 7. The architecture of the "Inception V3" network.

TABLE X
Comparison with the prior art of DenseNet-40 on CIFAR-10.

Compressing method	**Retraining needed**	**FLOPS**	**Pruned**	**Accuracy**
DenseNet-base	-	0.29B(0.0%)	0.0%	94.22%
ECNS **[51]**	Yes	0.12B(58.3%)	67.2%	94.35%
HRFM **[49]**	Yes	0.11B(61.8%)	55.1%	94.53%
GAL-0.05 **[48]**	No	0.13B(55.6%)	57.9%	92.11%
VP **[45]**	No	0.16B(45.8%)	60.7%	93.16%
Our method	**No**	**0.07B(76.1%)**	**78.8%**	**94.17%**



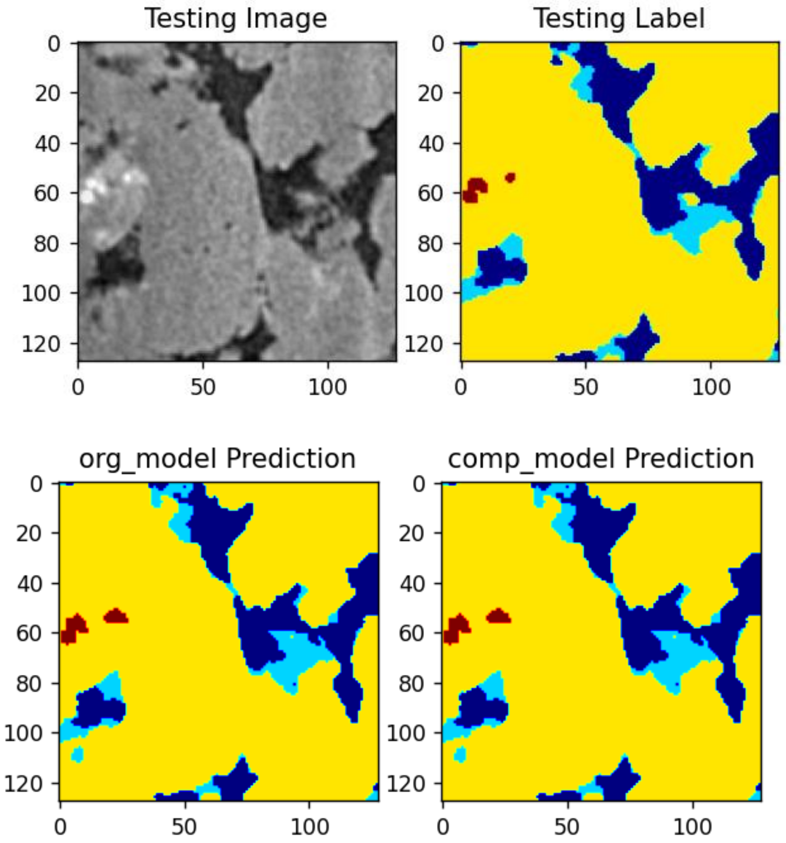
Model Compressing in Image Segmentation

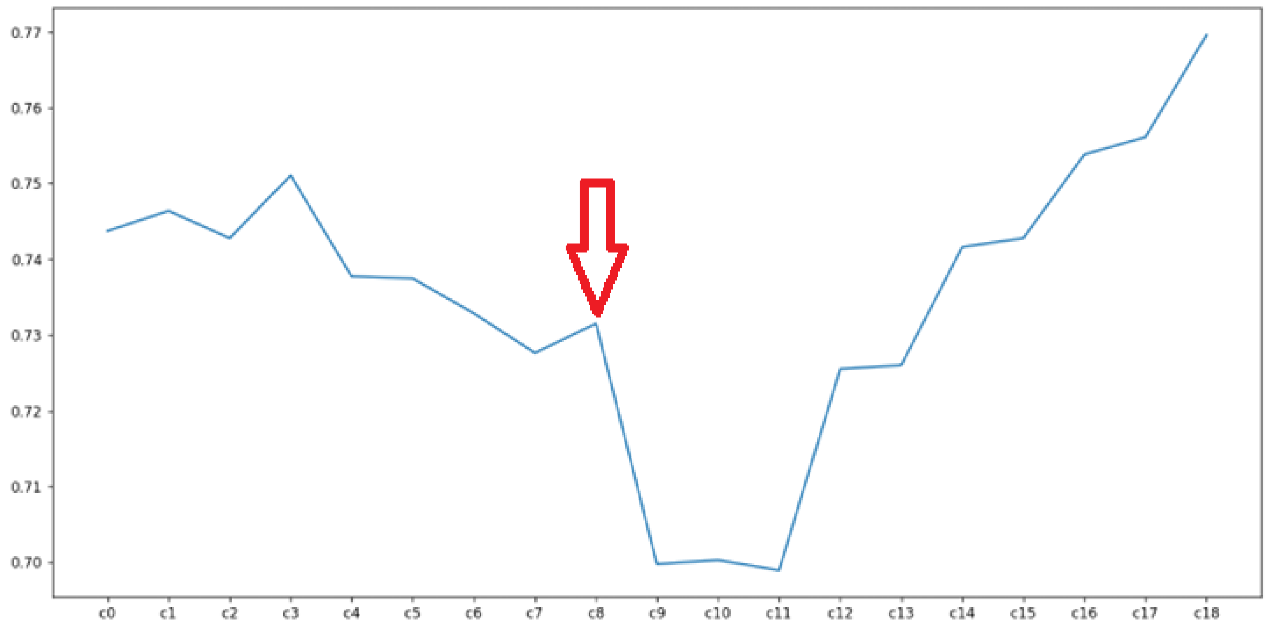
Algorithm

- ✓ Obtain the feature maps at the all layers of U-Net (or any DNN for the Segmentation purpose).
- ✓ Calculate Pairwise Distance Matrix For examples at each layer of the network.
- ✓ At each layer and for each input sample, determine the index of the closest vector in the distance matrix. This index indicates the most similar feature vector in the feature space to the given input sample's vector.
- ✓ Compute the Separation Index (SI) for different layers.
- ✓ Find a critical layer where a significant drop occurs, pruning the U-Net architecture from this layer.

Using simple U-Net– Multi-class segmentation

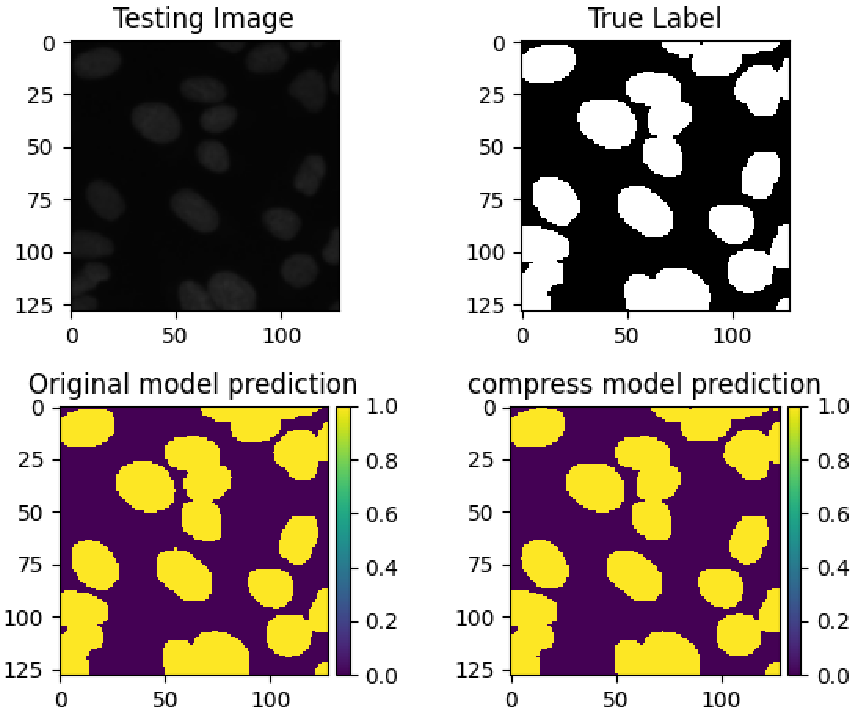
Unet Model	*Pram_num*	*accuracy*	*Mean IOU*	*Class1 IOU*	*Class2 IOU*	*Class3 IOU*	*Class4 IOU*
Original model	*1,946,756*	*96.95533514022827*	*0.88590944*	*0.91697925*	*0.7158169*	*0.9696021*	*0.9412393*
Compressed model	*155,268*	*96.52481079101562*	*0.87311*	*0.91641974*	*0.67240185*	*0.96412665*	*0.9394917*

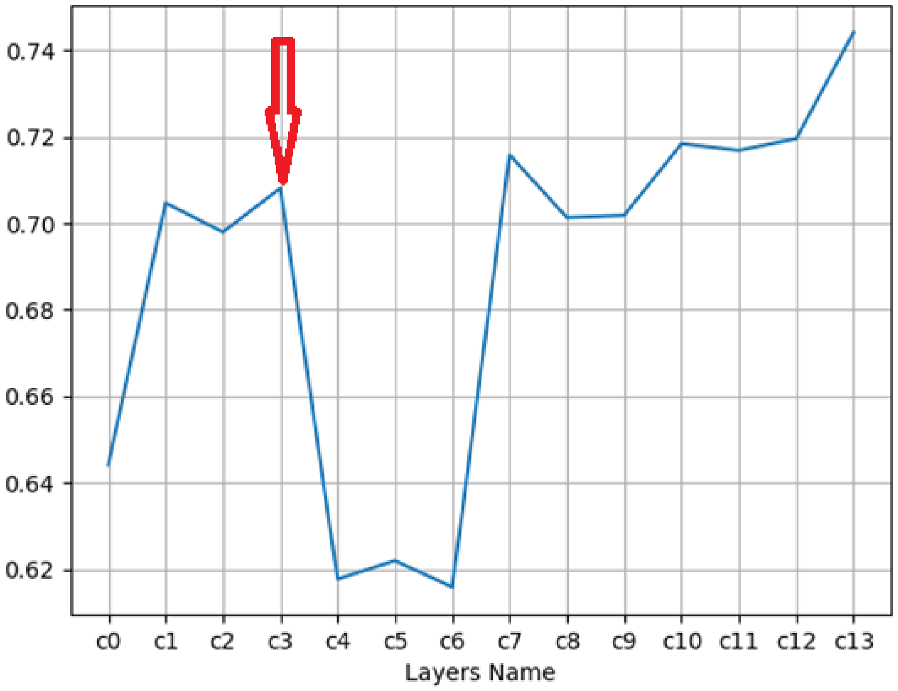




Using MobileNet in encoder path – Binary segmentation

	Trainable pram	*Overall mean IoU*	*Class1 Mean IoU*	*Class2 Mean IoU*
U-NET	*11,725,617*	*0.92313635*	*0.9560986*	*0.8901641*
Compress U-NET	*2,743,521*	*0.9254801*	*0.95738968*	*0.89357052*





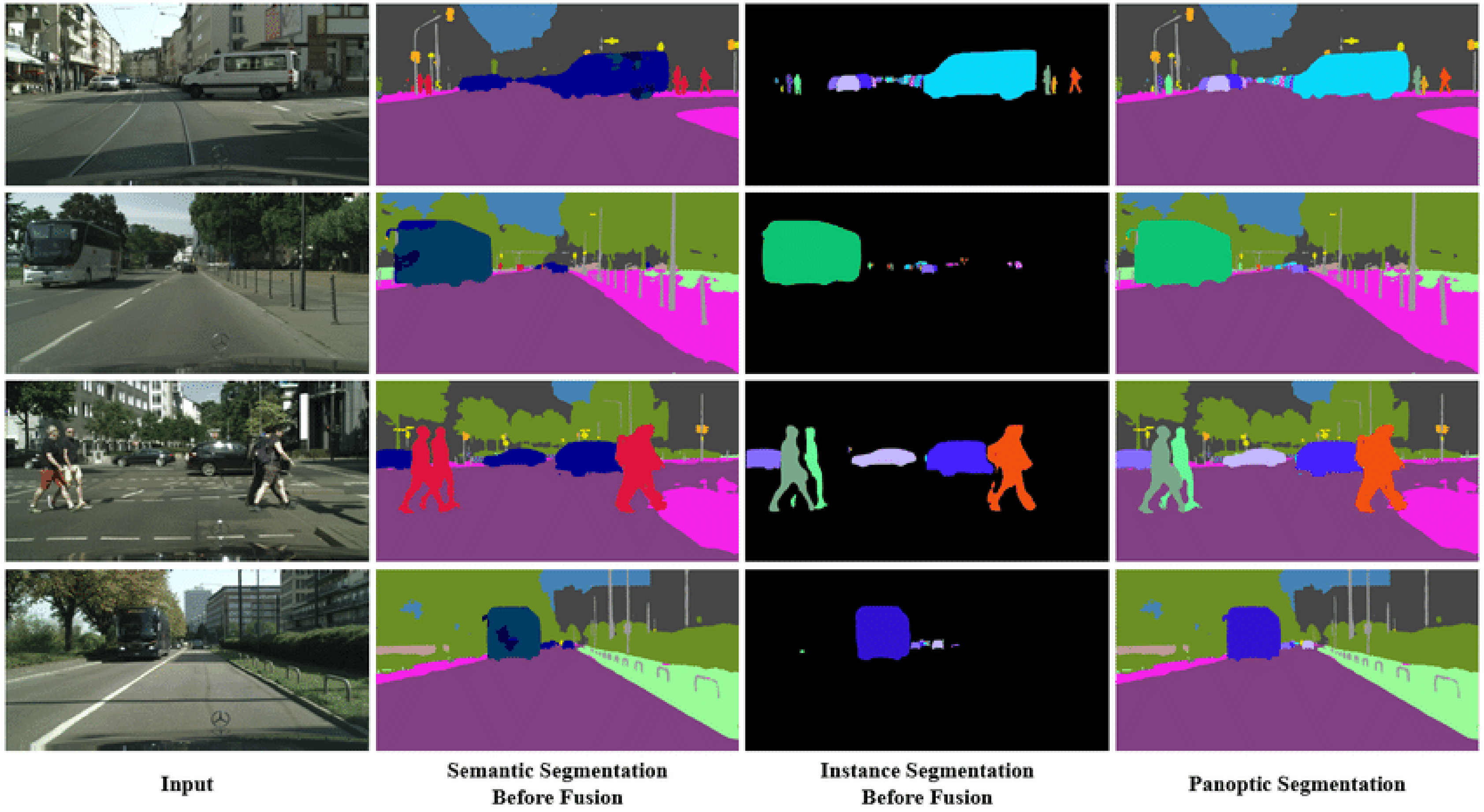
3.5 Region Based CNNs_part2

1.2.3 DNNs for Segmentation

DNNs which broke down an image into various subgroups called Image segments

There are three manners for segmentation: Semantic segmentation, Instance segmentation, and Panoptic segmentation

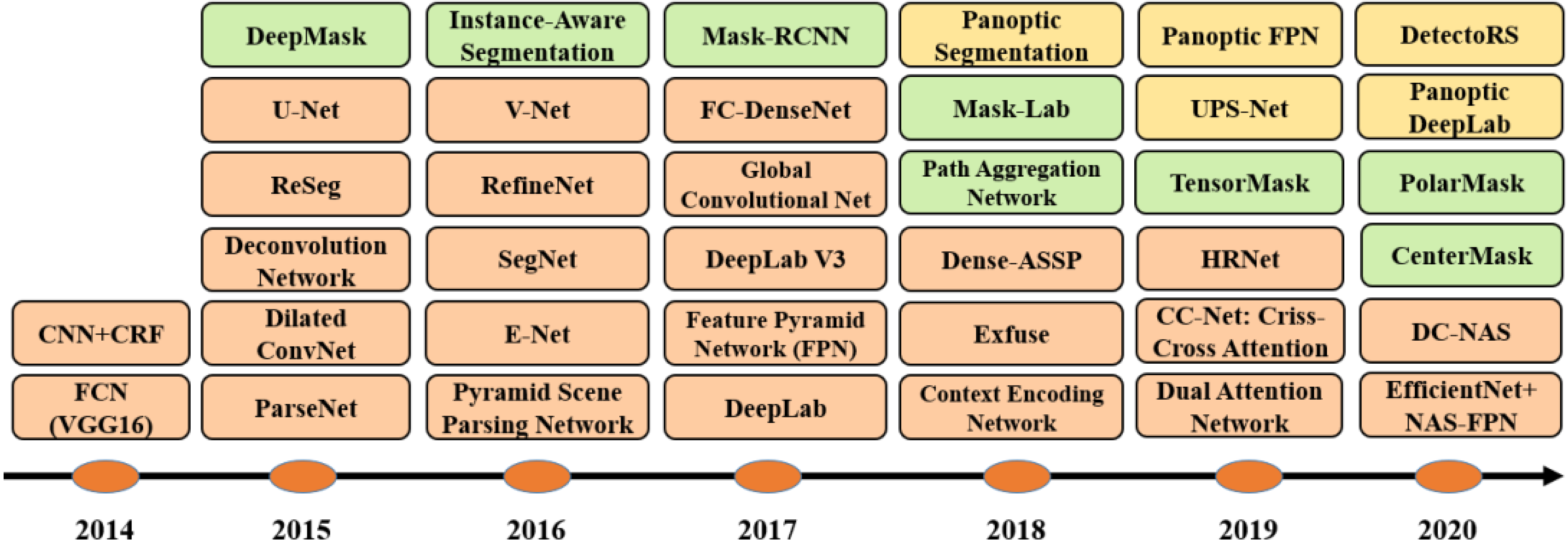
Panoptic segmentation is **proposed to unify the typically distinct tasks of semantic segmentation and instance segmentation**. The proposed task requires the generation of a rich and complete coherent scene segmentation, which is an important step towards a real-world visual system.


Input
Semantic Segmentation Before Fusion
Instance Segmentation Before Fusion
Panoptic Segmentation

Semantic Seg. Instance Seg. Panoptic Seg.

A timeline of DL-based Segmentation algorithms

*S. Minaee 2020



2014	2015	2016	2017	2018	2019	2020
	DeepMask	Instance-Aware Segmentation	Mask-RCNN	Panoptic Segmentation	Panoptic FPN	DetectoRS
	U-Net	V-Net	FC-DenseNet	Mask-Lab	UPS-Net	Panoptic DeepLab
	ReSeg	RefineNet	Global Convolutional Net	Path Aggregation Network	TensorMask	PolarMask
	Deconvolution Network	SegNet	DeepLab V3	Dense-ASSP	HRNet	CenterMask
CNN+CRF	Dilated ConvNet	E-Net	Feature Pyramid Network (FPN)	Exfuse	CC-Net: Criss-Cross Attention	DC-NAS
FCN (VGG16)	ParseNet	Pyramid Scene Parsing Network	DeepLab	Context Encoding Network	Dual Attention Network	EfficientNet+ NAS-FPN

The timeline of DL-based segmentation algorithms for 2D images, from 2014 to 2020.

Orange, green, and yellow blocks refer to semantic, instance, and panoptic segmentation algorithms respectively

Datasets for Segmentation

- 2D images RGB

1. PASCAL Visual Object Classes (VOC)
2. PASCAL Context
3. Microsoft Common Objects in Context (MS COCO)
4. Cityscapes
5. ADE20K /MIT Scene Parsing (SceneParse150)
6. SiftFlow
7. Stanford background
8. Berkeley Segmentation Dataset (BSD)
9. Youtube-Objects
10. KITTI

- 2.5 D images RGB –D

1. NYU-D V2
2. SUN-3D
3. SUN RGB-D
4. UW RGB-D Object Dataset
5. ScanNet

Using RGB and Depth

- 3 D images

1. Stanford 2D-3D
2. ShapeNet Core
3. Sydney Urban
4. Objects Dataset:

3D image datasets are popular in robotic, medical image analysis, 3D scene analysis, and construction applications.
Three dimensional images are usually provided via meshes or other volumetric representations, such as point clouds.

RGB-D

Metrics For Segmentation Models

- **Pixel accuracy**

where P_{ij} is the number of pixels of class *i* predicted as belonging to class *j*.

$$\text{PA} = \frac{\sum_{i=0}^{K} p_{ii}}{\sum_{i=0}^{K} \sum_{j=0}^{K} p_{ij}}$$

- **Mean Pixel Accuracy (MPA)**

MPA is the extended version of PA, in which the ratio of correct pixels is computed in a per-class manner and then averaged over the total number of classes

$$\text{MPA} = \frac{1}{K+1} \sum_{i=0}^{K} \frac{p_{ii}}{\sum_{j=0}^{K} p_{ij}}$$

- **Intersection over Union (IoU)**

$$\text{IoU} = J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

(IoU) or the **Jaccard Index** is one of the most commonly used metrics in semantic segmentation. It is defined as the area of intersection between the predicted segmentation map and the ground truth, divided by the area of union between the predicted segmentation map and the ground truth

- **Mean-IoU**

  the average IoU over all classes

-

- **Precision / Recall / F1 score**

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}},$$

$$\text{F1-score} = \frac{2 \text{ Prec Rec}}{\text{Prec} + \text{Rec}}$$

- **Dice coefficient**

$$\text{Dice} = \frac{2|A \cap B|}{|A| + |B|} \qquad \text{Dice} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}} = \text{F1}$$

Comparison

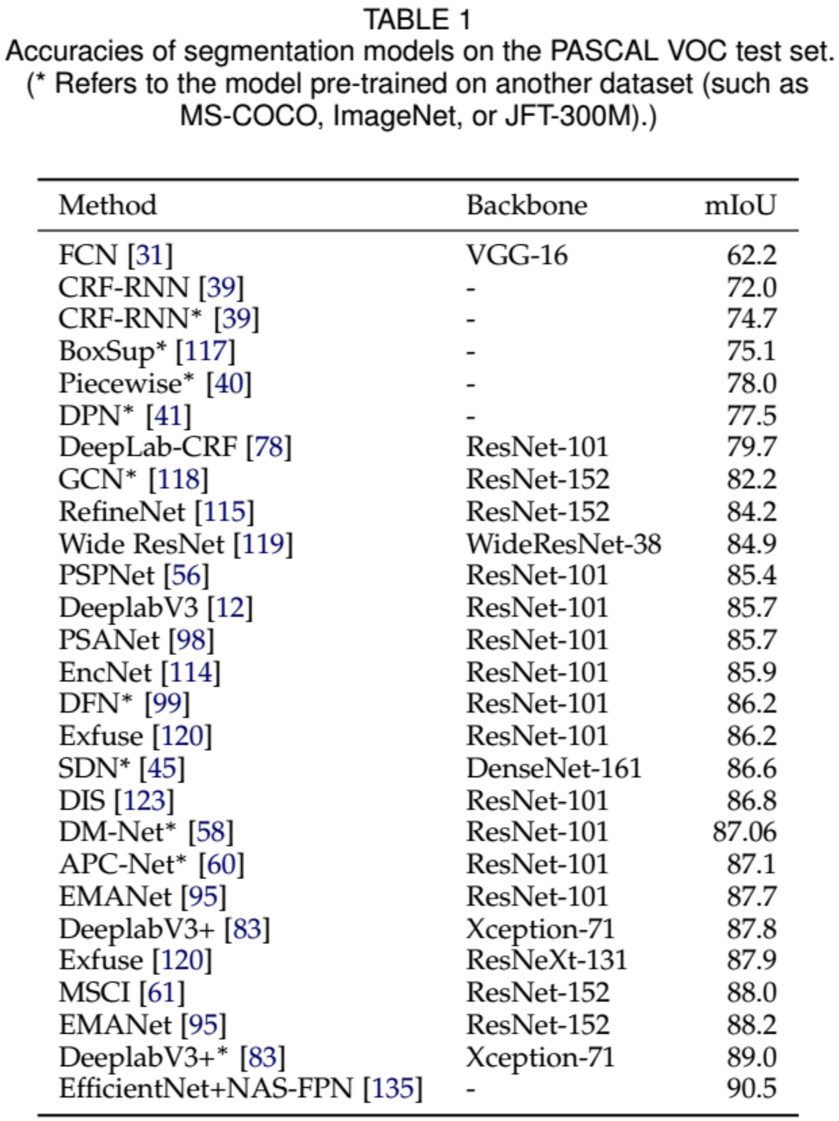


TABLE 1
Accuracies of segmentation models on the PASCAL VOC test set. (* Refers to the model pre-trained on another dataset (such as MS-COCO, ImageNet, or JFT-300M).)

Method	Backbone	mIoU
FCN [31]	VGG-16	62.2
CRF-RNN [39]	-	72.0
CRF-RNN* [39]	-	74.7
BoxSup* [117]	-	75.1
Piecewise* [40]	-	78.0
DPN* [41]	-	77.5
DeepLab-CRF [78]	ResNet-101	79.7
GCN* [118]	ResNet-152	82.2
RefineNet [115]	ResNet-152	84.2
Wide ResNet [119]	WideResNet-38	84.9
PSPNet [56]	ResNet-101	85.4
DeeplabV3 [12]	ResNet-101	85.7
PSANet [98]	ResNet-101	85.7
EncNet [114]	ResNet-101	85.9
DFN* [99]	ResNet-101	86.2
Exfuse [120]	ResNet-101	86.2
SDN* [45]	DenseNet-161	86.6
DIS [123]	ResNet-101	86.8
DM-Net* [58]	ResNet-101	87.06
APC-Net* [60]	ResNet-101	87.1
EMANet [95]	ResNet-101	87.7
DeeplabV3+ [83]	Xception-71	87.8
Exfuse [120]	ResNeXt-131	87.9
MSCI [61]	ResNet-152	88.0
EMANet [95]	ResNet-152	88.2
DeeplabV3+* [83]	Xception-71	89.0
EfficientNet+NAS-FPN [135]	-	90.5

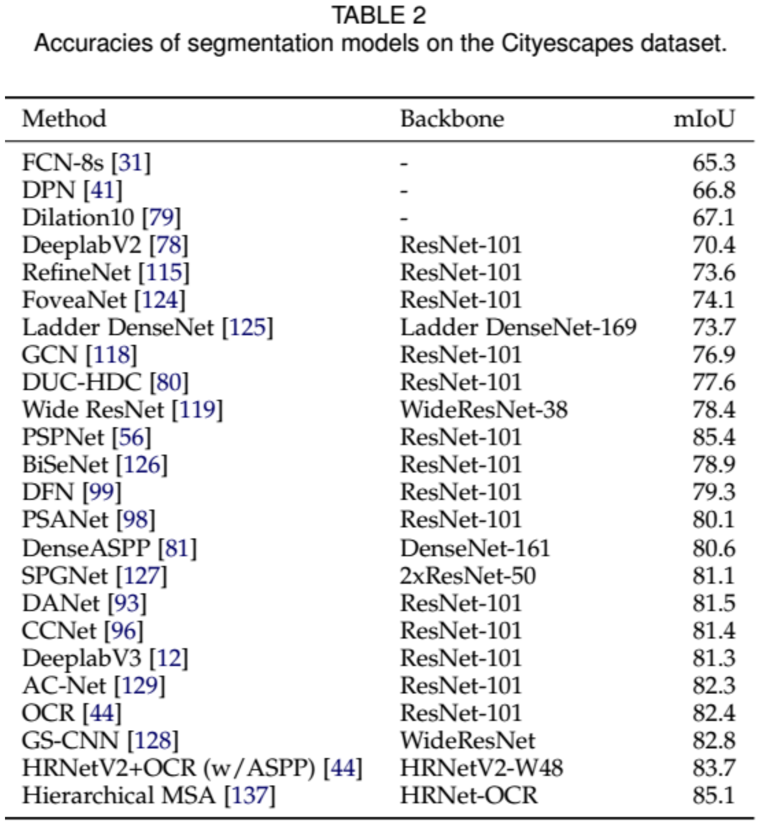


TABLE 2
Accuracies of segmentation models on the Cityescapes dataset.

Method	Backbone	mIoU
FCN-8s [31]	-	65.3
DPN [41]	-	66.8
Dilation10 [79]	-	67.1
DeeplabV2 [78]	ResNet-101	70.4
RefineNet [115]	ResNet-101	73.6
FoveaNet [124]	ResNet-101	74.1
Ladder DenseNet [125]	Ladder DenseNet-169	73.7
GCN [118]	ResNet-101	76.9
DUC-HDC [80]	ResNet-101	77.6
Wide ResNet [119]	WideResNet-38	78.4
PSPNet [56]	ResNet-101	85.4
BiSeNet [126]	ResNet-101	78.9
DFN [99]	ResNet-101	79.3
PSANet [98]	ResNet-101	80.1
DenseASPP [81]	DenseNet-161	80.6
SPGNet [127]	2xResNet-50	81.1
DANet [93]	ResNet-101	81.5
CCNet [96]	ResNet-101	81.4
DeeplabV3 [12]	ResNet-101	81.3
AC-Net [129]	ResNet-101	82.3
OCR [44]	ResNet-101	82.4
GS-CNN [128]	WideResNet	82.8
HRNetV2+OCR (w/ASPP) [44]	HRNetV2-W48	83.7
Hierarchical MSA [137]	HRNet-OCR	85.1

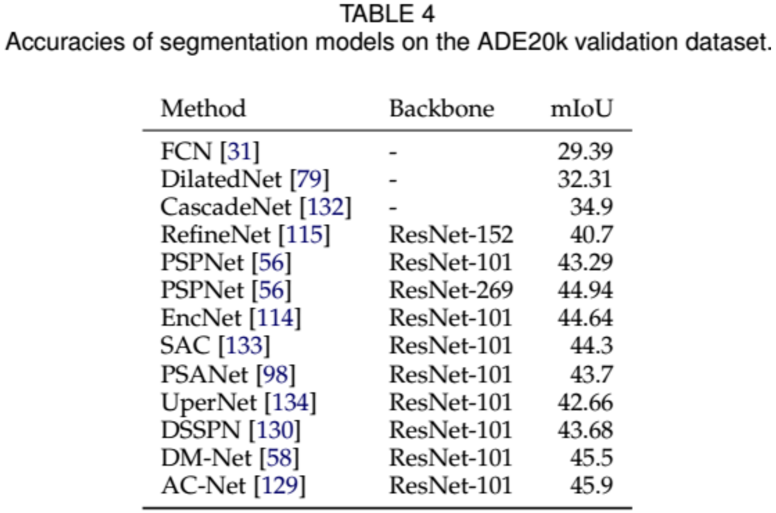


TABLE 4
Accuracies of segmentation models on the ADE20k validation dataset.

Method	Backbone	mIoU
FCN [31]	-	29.39
DilatedNet [79]	-	32.31
CascadeNet [132]	-	34.9
RefineNet [115]	ResNet-152	40.7
PSPNet [56]	ResNet-101	43.29
PSPNet [56]	ResNet-269	44.94
EncNet [114]	ResNet-101	44.64
SAC [133]	ResNet-101	44.3
PSANet [98]	ResNet-101	43.7
UperNet [134]	ResNet-101	42.66
DSSPN [130]	ResNet-101	43.68
DM-Net [58]	ResNet-101	45.5
AC-Net [129]	ResNet-101	45.9

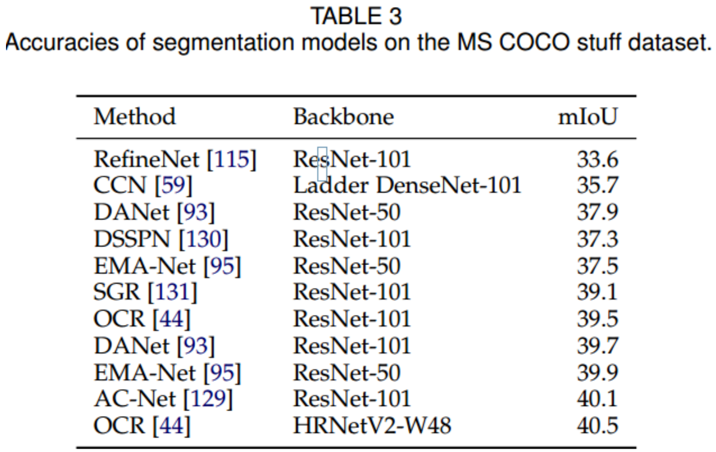


TABLE 3
Accuracies of segmentation models on the MS COCO stuff dataset.

Method	Backbone	mIoU
RefineNet [115]	ResNet-101	33.6
CCN [59]	Ladder DenseNet-101	35.7
DANet [93]	ResNet-50	37.9
DSSPN [130]	ResNet-101	37.3
EMA-Net [95]	ResNet-50	37.5
SGR [131]	ResNet-101	39.1
OCR [44]	ResNet-101	39.5
DANet [93]	ResNet-101	39.7
EMA-Net [95]	ResNet-50	39.9
AC-Net [129]	ResNet-101	40.1
OCR [44]	HRNetV2-W48	40.5

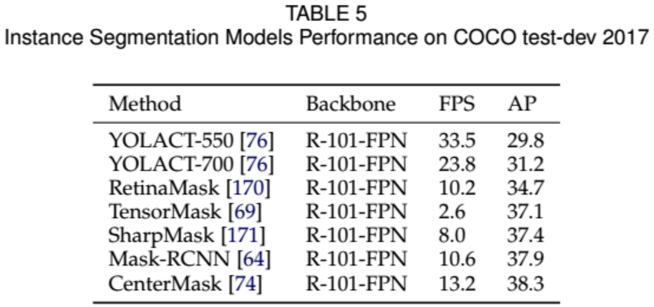


TABLE 5
Instance Segmentation Models Performance on COCO test-dev 2017

Method	Backbone	FPS	AP
YOLACT-550 [76]	R-101-FPN	33.5	29.8
YOLACT-700 [76]	R-101-FPN	23.8	31.2
RetinaMask [170]	R-101-FPN	10.2	34.7
TensorMask [69]	R-101-FPN	2.6	37.1
SharpMask [171]	R-101-FPN	8.0	37.4
Mask-RCNN [64]	R-101-FPN	10.6	37.9
CenterMask [74]	R-101-FPN	13.2	38.3

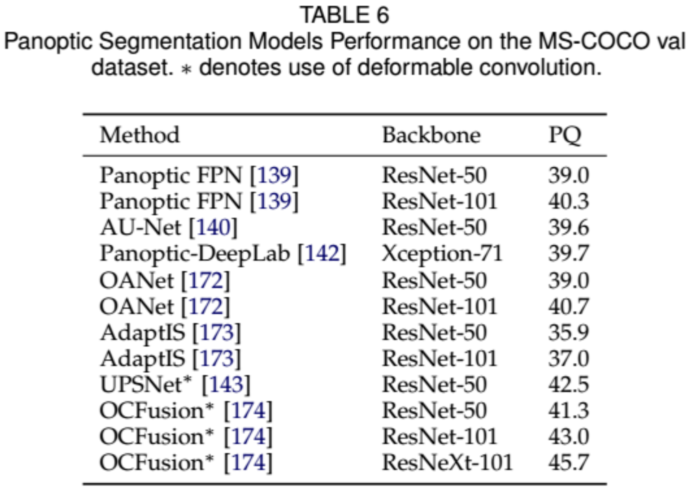


TABLE 6
Panoptic Segmentation Models Performance on the MS-COCO val dataset. ∗ denotes use of deformable convolution.

Method	Backbone	PQ
Panoptic FPN [139]	ResNet-50	39.0
Panoptic FPN [139]	ResNet-101	40.3
AU-Net [140]	ResNet-50	39.6
Panoptic-DeepLab [142]	Xception-71	39.7
OANet [172]	ResNet-50	39.0
OANet [172]	ResNet-101	40.7
AdaptIS [173]	ResNet-50	35.9
AdaptIS [173]	ResNet-101	37.0
UPSNet* [143]	ResNet-50	42.5
OCFusion* [174]	ResNet-50	41.3
OCFusion* [174]	ResNet-101	43.0
OCFusion* [174]	ResNeXt-101	45.7

Taxonomy of DNNs for Segmentation

1. Fully Conventional Neural Networks
2. Convolutional Models With Graphical Models
3. Encoder-Decoder Based Models
4. Multi-Scale and Pyramid Network Based Models
5. R-CNN Based Models (for Instance Segmentation)
6. Dilated Convolutional Models and DeepLab Family
7. Recurrent Neural Network Based Models
8. Attention-Based Models
9. Generative Models and Adversarial Training
10. **CNN Models With Active Contour Models**

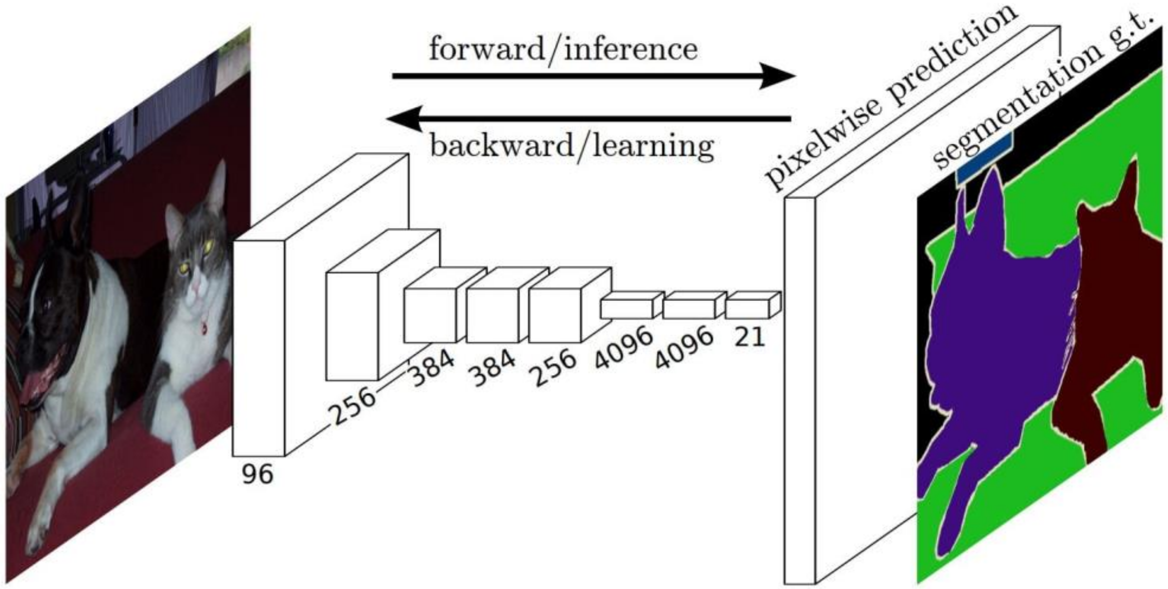
1. Fully Conventional Neural Networks

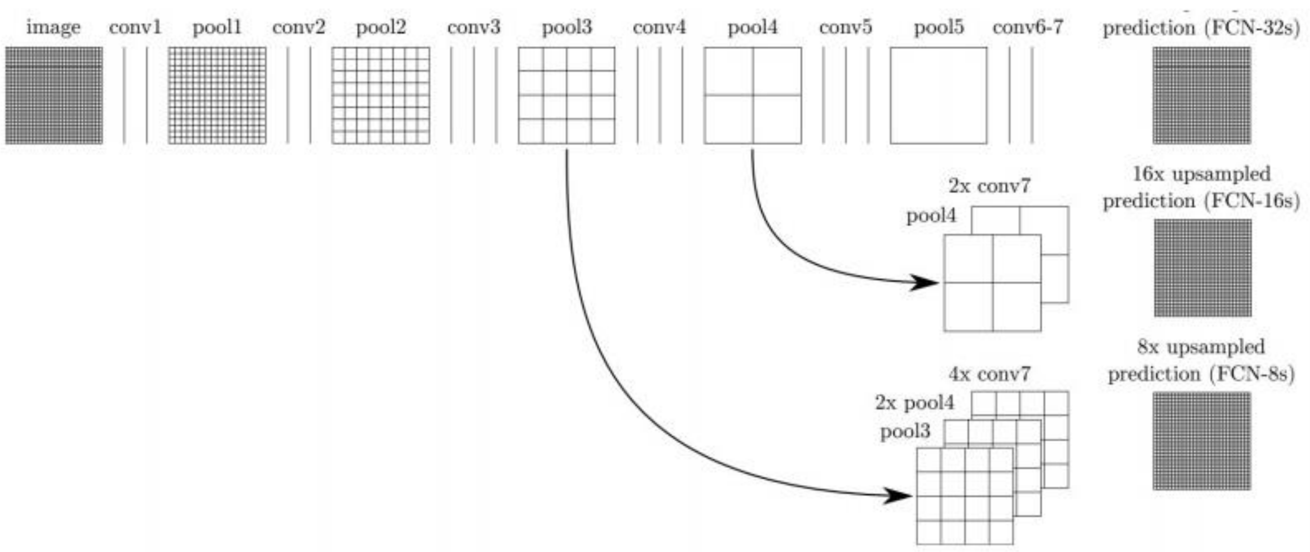
It includes only convolutional layers

The applied backbones are CNN architectures such as VGG16 and GoogLeNet

FCN

J. Long... 2015

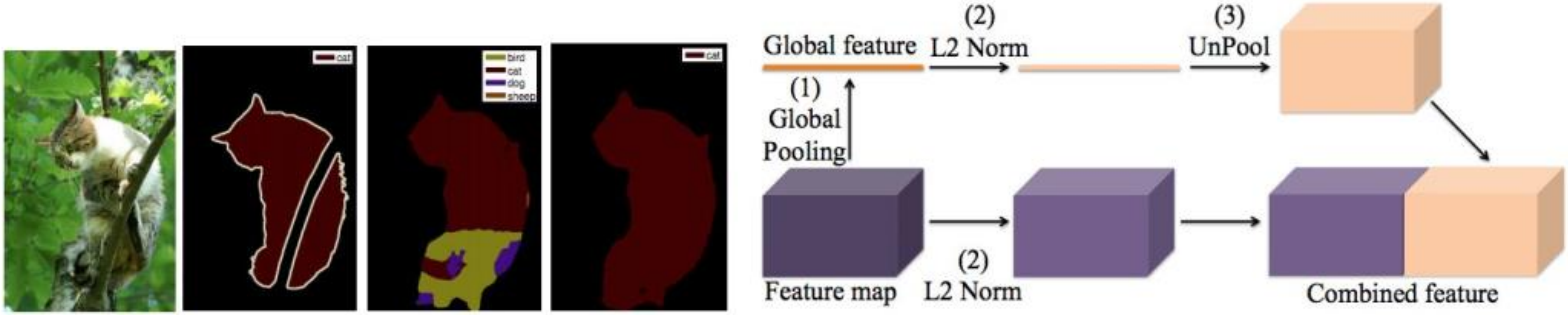




Through the use of skip connections in which feature maps from the final layers of the model are up-sampled and fused with feature maps of earlier layers, the model combines semantic information (from deep, coarse layers) and appearance information (from shallow, fine layers) in order to produce accurate and detailed segmentations.

ParseNet

W. Liu..2015



(a) Image (b) Truth (c) FCN (d) ParseNet (e) ParseNet contexture module overview.

ParseNet adds global context to FCNs by using the average feature for a layer to augment the features at each location.

The feature map for a layer is pooled over the whole image resulting in a context vector.

This context vector is normalized and un-pooled to produce new feature maps of the same size as the initial ones.

2. Convolutional Models With Graphical Models

As discussed, FCN ignores potentially useful scene-level semantic context. To integrate more context, several approaches incorporate probabilistic graphical models, such as Conditional Random Fields (CRFs) and Markov Random Field (MRFs), into DL architectures.

CNN+CRF (Condition Random Field)

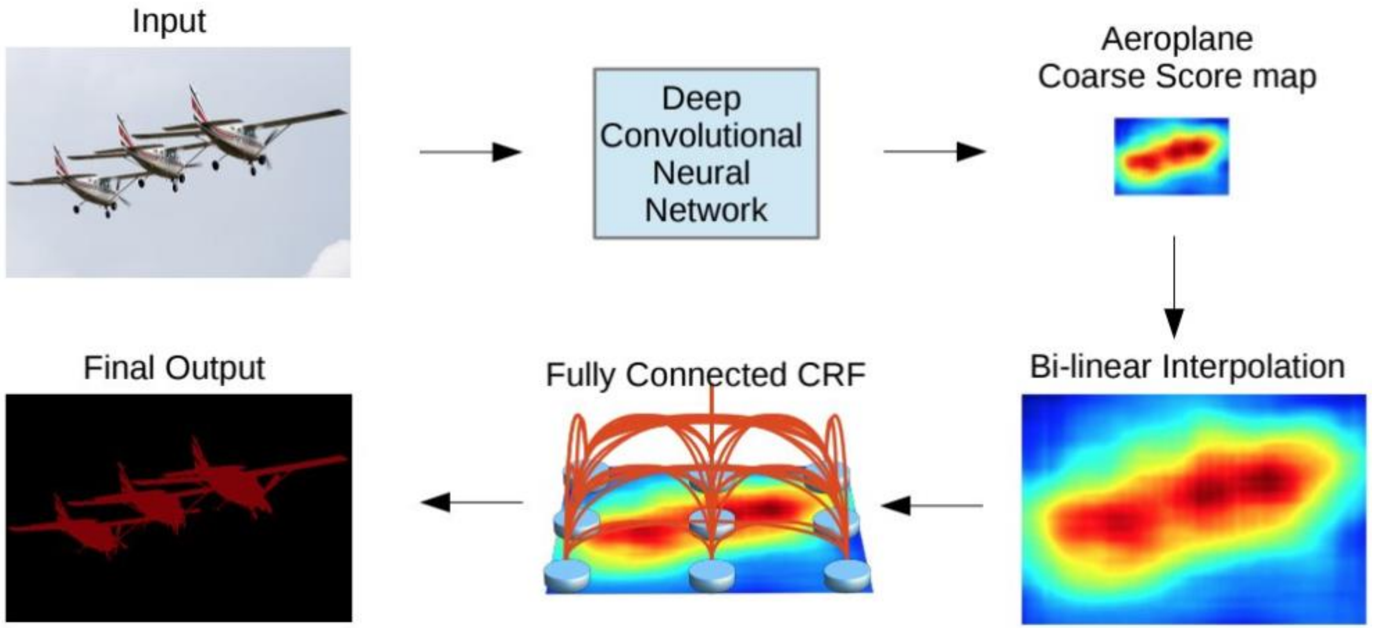
Chen..2014

Chen proposed a semantic segmentation algorithm based on the combination of CNNs and fully connected CRFs. (CNN+CRF)

They showed that responses from the final layer of deep CNNs are not sufficiently localized for accurate object segmentation.

To overcome the poor localization property of deep CNNs, they combined the responses at the final CNN layer with a fully-connected CRF.

They showed that their model is able to localize segment boundaries at a higher accuracy rate than it was possible with previous methods.



The coarse score map of a CNN is upsampled via interpolated interpolation, and fed to a fully-connected CRF to refine the segmentation result.

Conditional random fields (CRFs) are **a class of statistical modeling methods used for structured prediction**. Whereas a classifier predicts a label for a single sample without onsidering "neighbouring" samples, a CRF can take context into account.

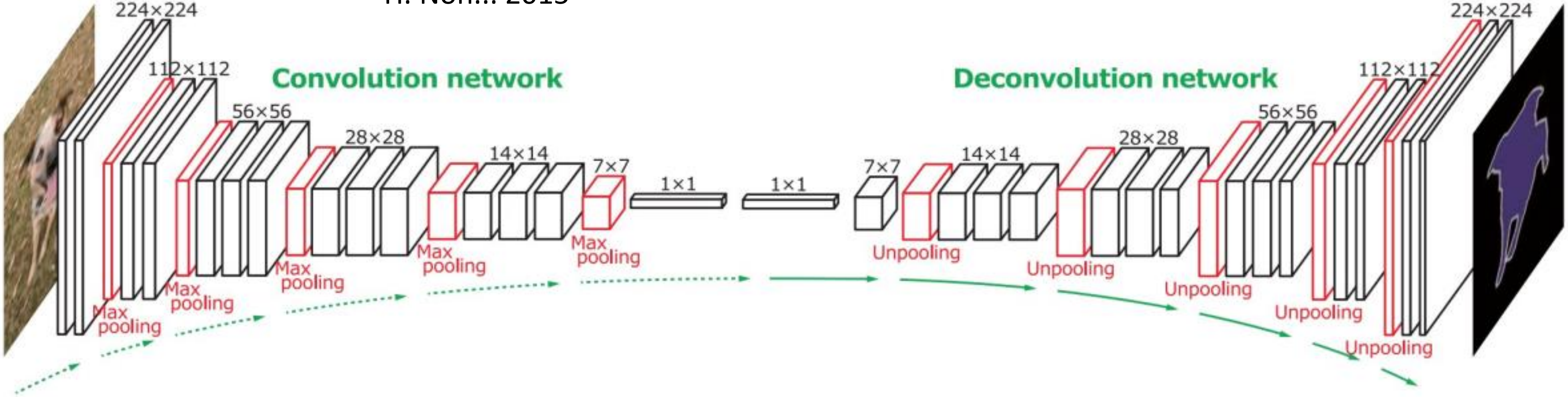
3. Encoder-Decoder Based Models

image segmentation based on the convolutional encoder-decoder architecture.

A. Encoder-Decoder Models for General Segmentation

DSS (De-convolutional semantic segmentation)

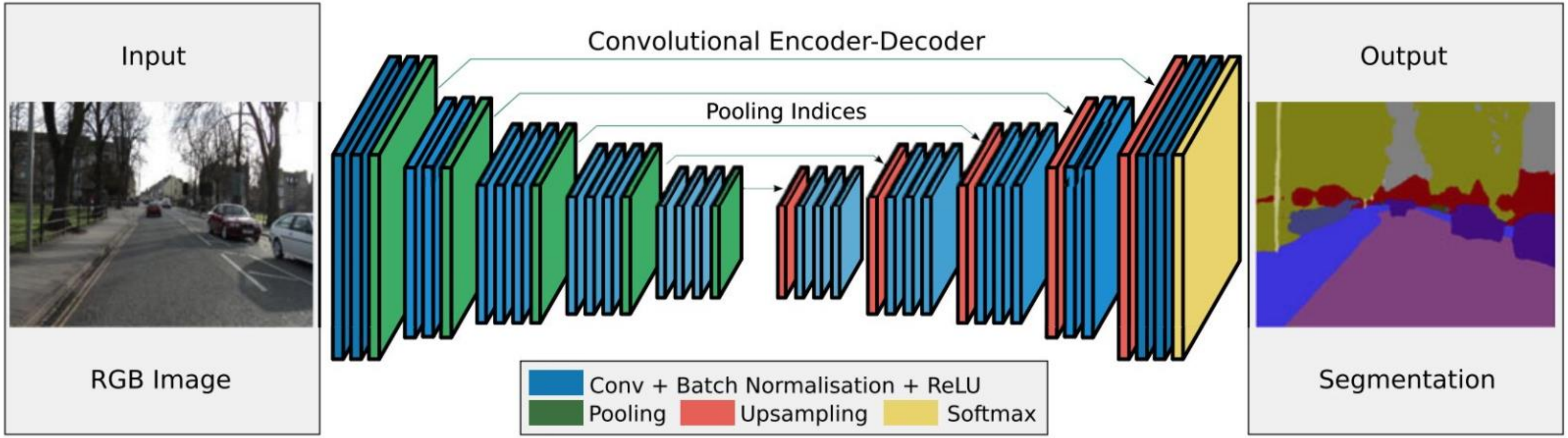
H. Noh... 2015



A convolution network based on the VGG 16-layer net, is a multi-layer deconvolution network to generate the accurate segmentation map.

SegNet

A. Kendall ...2015



SegNet has no fully-connected layers; hence, the model is fully convolutional. A decoder up-samples its input using the transferred pool indices from its encoder to produce a sparse feature map(s)

HRNET(high-resolution network)

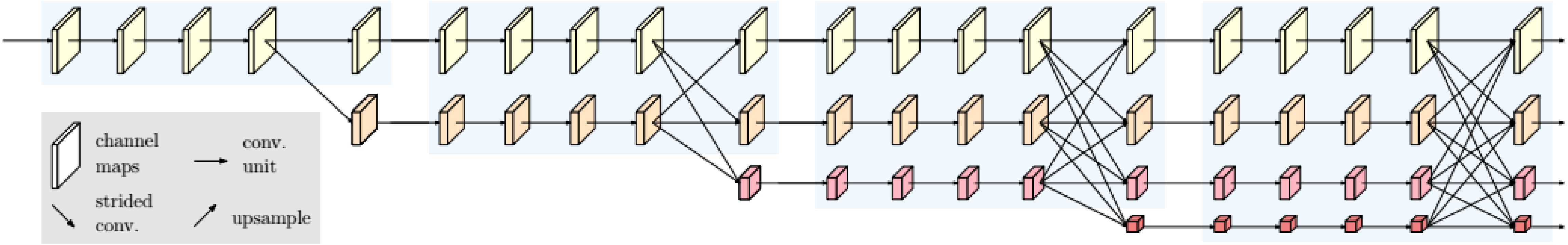
Y. Yuan...2019

Satge1
(high resolution Conv.)

Satge2
(Two resolution Blocks)

Satge3
(Three resolution Blocks)

Satge4
(Four resolution Blocks)



HRNet consists of parallel high-to-low resolution convolution streams with repeated information exchange across multi-resolution steams.

There are four stages:

The 1st stage consists of high-resolution convolutions.

The 2nd (3rd, 4th) stage repeats two-resolution (three-resolution, four-resolution) blocks.

B. Encoder-Decoder Models for Medical and Biomedical Image Segmentation

UNet

Ronneberger....2015

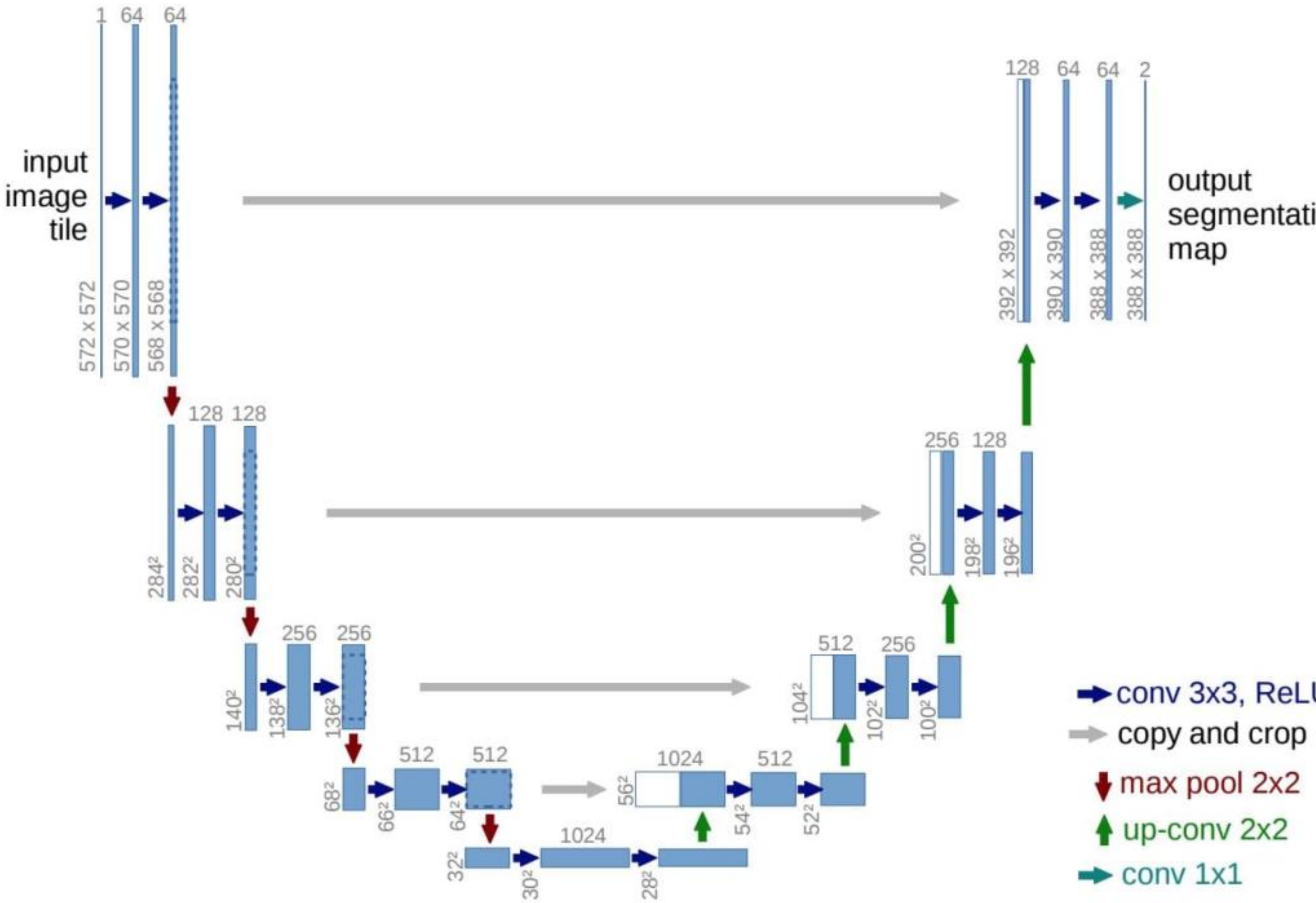
U-Net is proposed for segmenting biological microscopy images.

Their network and training strategy relies on the use of data augmentation to learn from the very few annotated images effectively.

The U-Net architecture comprises two parts, a contracting path to capture context, and a symmetric expanding path that enables precise localization.

The down-sampling or contracting part has a FCN-like architecture that extracts features with 3 × 3 convolutions. The up-sampling or expanding part uses up-convolution (or deconvolution), reducing the number of feature maps while increasing their dimensions.

Feature maps from the down-sampling part of the network are copied to the up-sampling part to avoid losing pattern information.

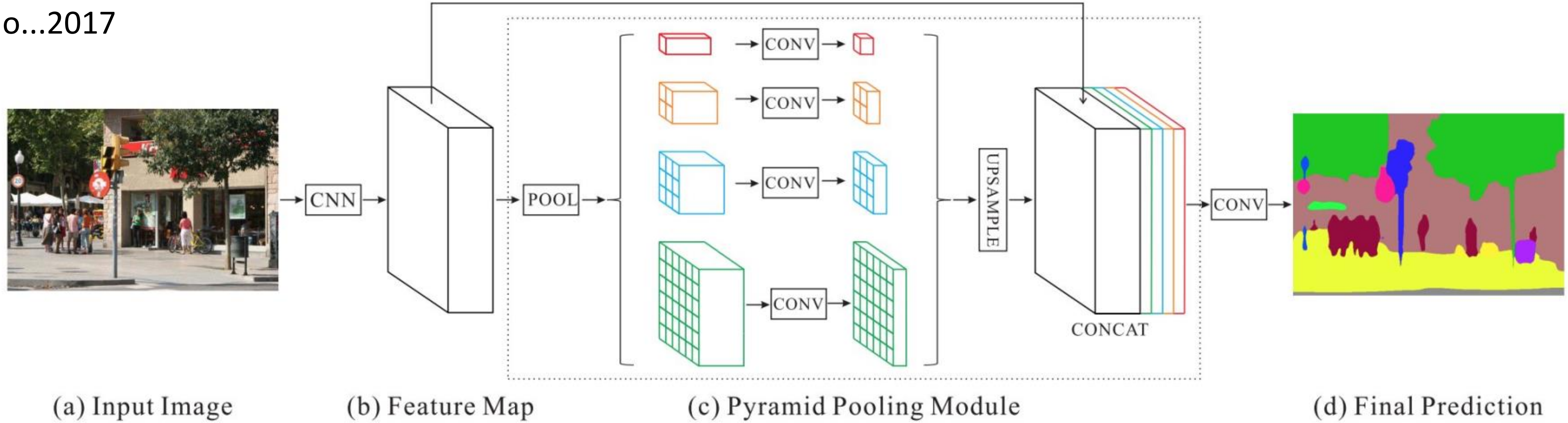


4. Multi-Scale and Pyramid Network Based Models

DNNs whose Backbone is from CNNs with Multi-Scale and Pyramid Network

PSPN Pyramid scene parsing network

Zhao...2017



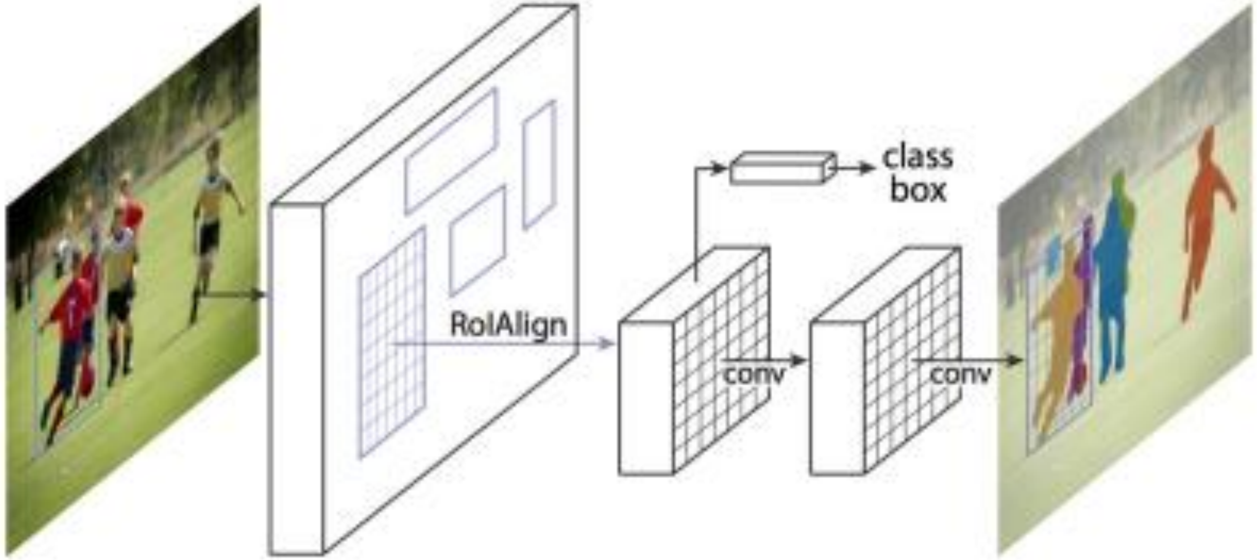
(a) Input Image (b) Feature Map (c) Pyramid Pooling Module (d) Final Prediction

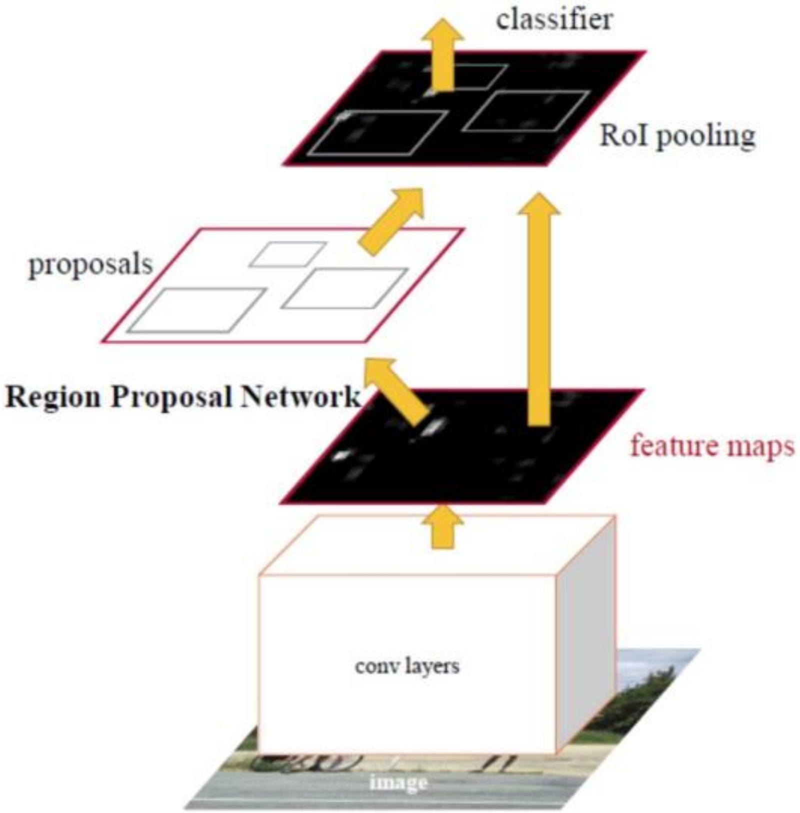
A CNN produces the feature map and a pyramid pooling module aggregates the different sub-region representations. Up-sampling and concatenation are used to form the final feature representation from which, the final pixel-wise prediction is obtained through convolution.

5. R-CNN Based Models (for Instance Segmentation)

Mask R-CNN

K. He...2017





Mask R-CNN architecture for instance segmentation

MaskLab(Instance Segmentation)

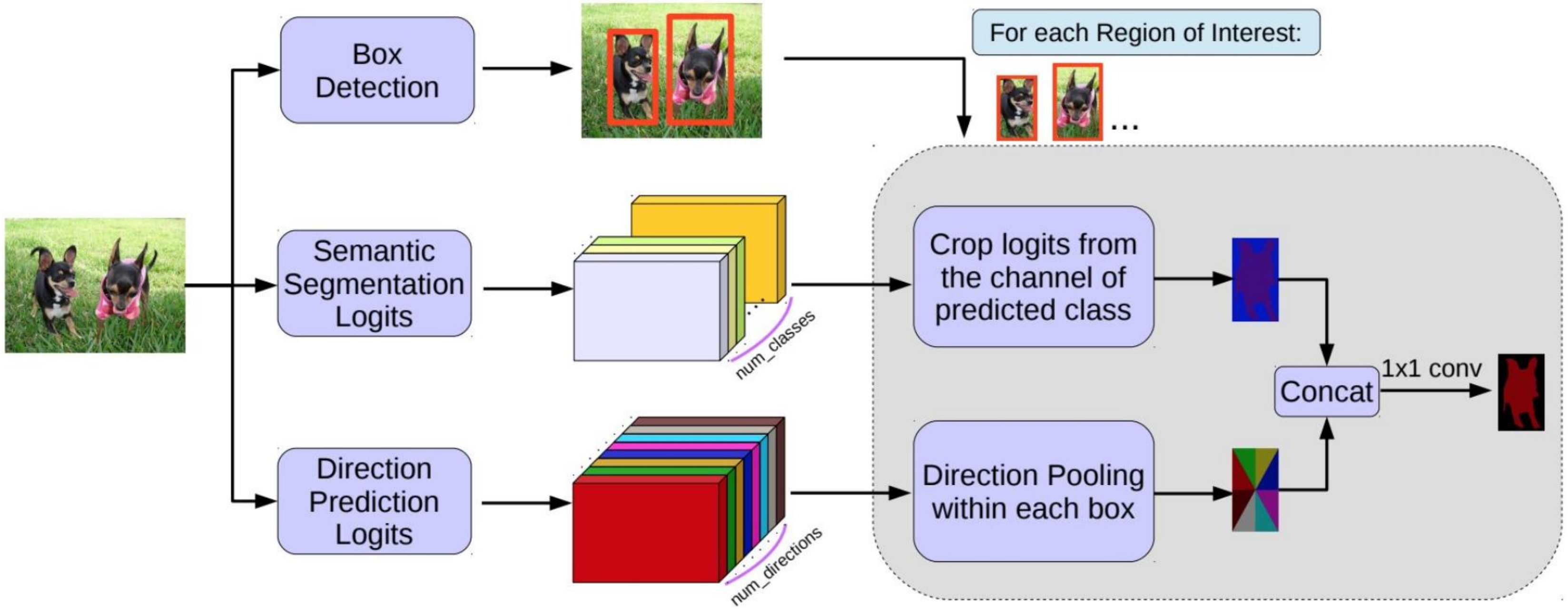
L. Chen…2018

A model by refining object detection with semantic and direction features based on Faster R-CNN.

This model produces three outputs, box detection, semantic segmentation, and direction prediction.

Building on the FasterRCNN object detector, the predicted boxes provide accurate localization of object instances.

Within each region of interest, MaskLab performs foreground/background segmentation by combining semantic and direction prediction.



Box Detection
For each Region of Interest:
...
Semantic Segmentation Logits
num_classes
Crop logits from the channel of predicted class
Concat
1x1 conv
Direction Prediction Logits
num_directions
Direction Pooling within each box

MaskLab generates three outputs—refined box predictions (from Faster R-CNN), semantic segmentation logits for pixel-wise classification, and direction prediction logits for predicting each pixel’s direction toward its instance center.

6. Dilated Convolutional Models and DeepLab Family

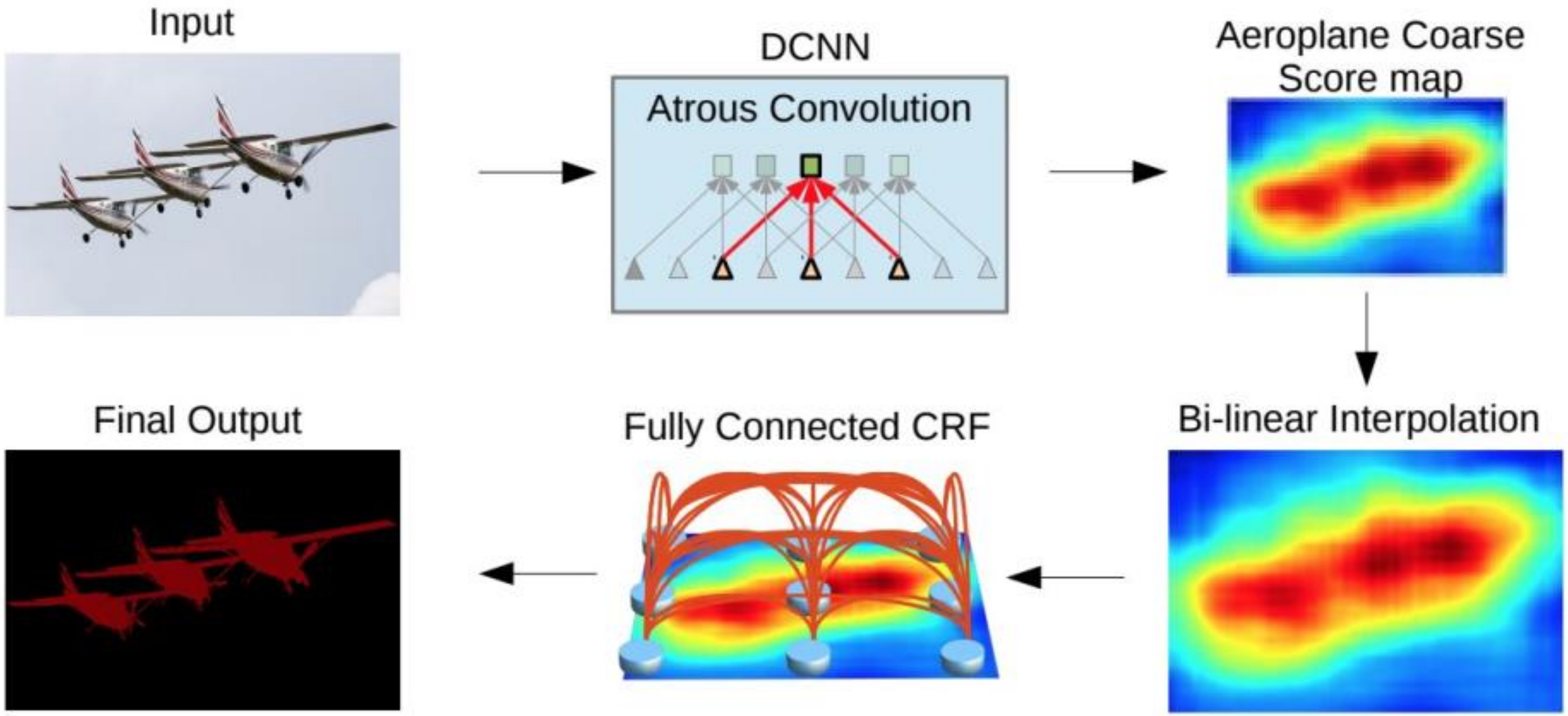
DeepLab

Chen. 2017

Similar to

CNN+CRF

2014



A CNN model such as VGG-16 or ResNet-101 is employed in fully convolutional fashion, using dilated convolution.

A bilinear interpolation stage enlarges the feature maps to the original image resolution. Finally, a fully connected CRF(Conditional Random Field) refines the segmentation result to better capture the object boundaries.

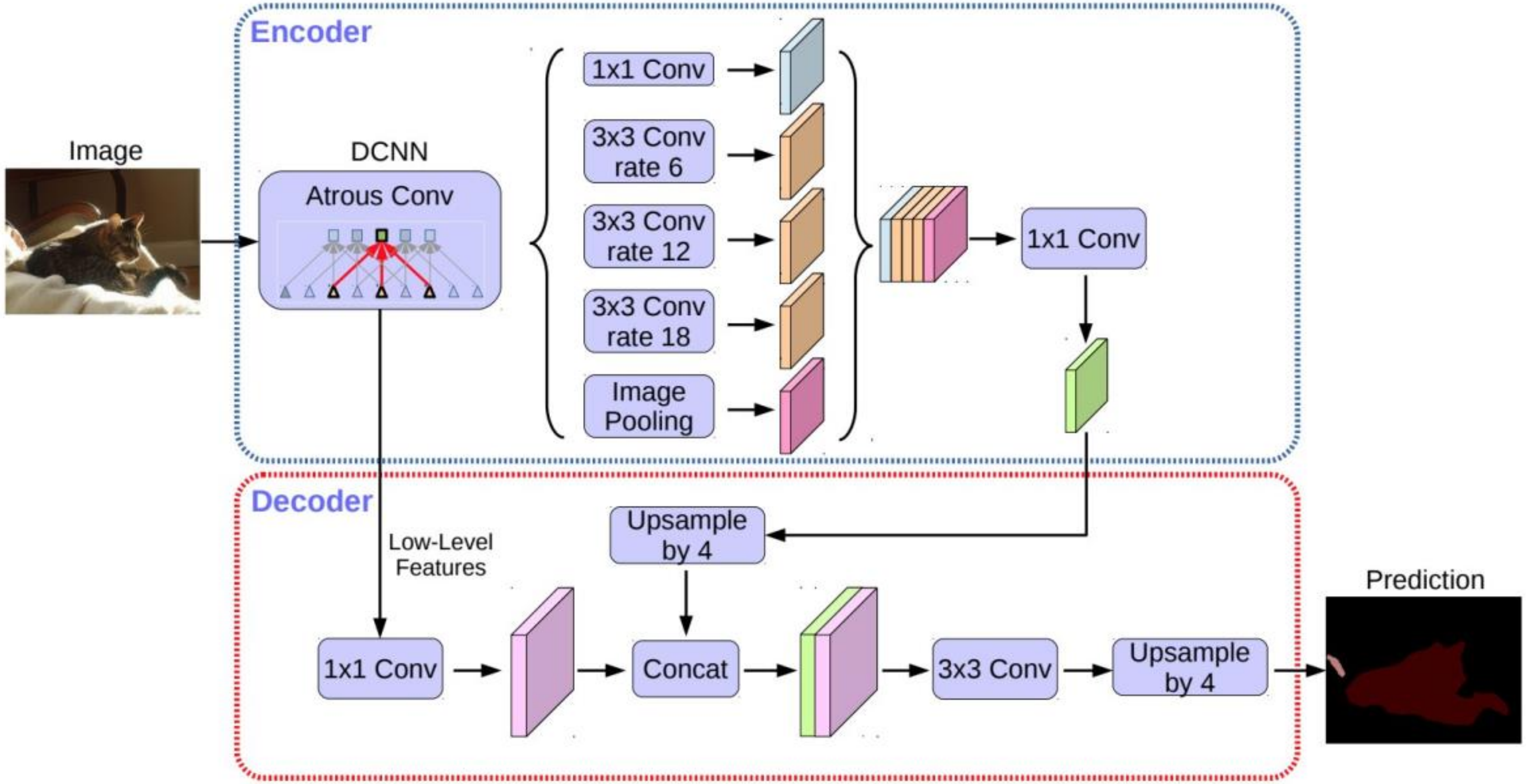
DeepLabv3+

Chen....2018

Deeplabv3+ uses an encoder-decoder architecture, including atrous separable convolution, composed of a depthwise convolution (spatial convolution for each channel of the input) and pointwise convolution (1×1 convolution with the depthwise convolution as input).

They used the DeepLabv3 framework as encoder.

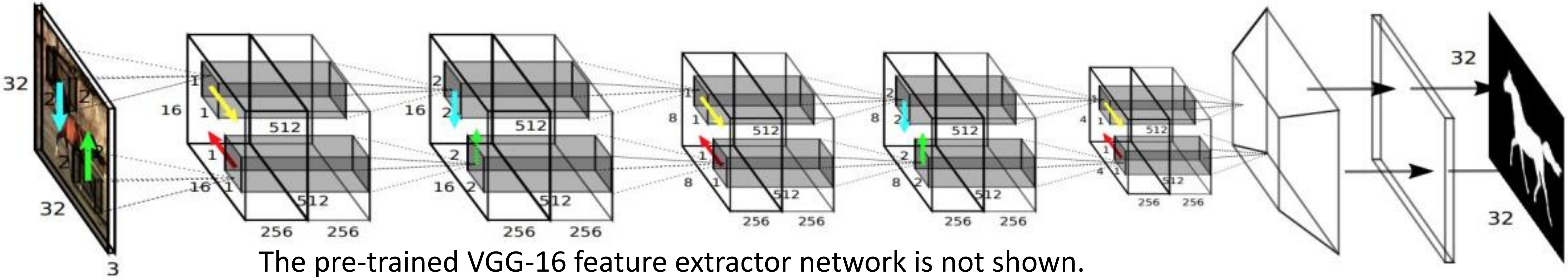
The most relevant model has a modified Xception backbone with more layers, dilated depthwise separable convolutions instead of max pooling and batch normalization.



7. Recurrent Neural Network Based Models

ReSeg

Visin ..2016



The pre-trained VGG-16 feature extractor network is not shown.

This model is mainly based on ReNet (which was developed for image classification).

Each ReNet layer is composed of four RNNs that sweep the image horizontally and vertically in both directions, encoding patches/activations, and providing relevant global information.

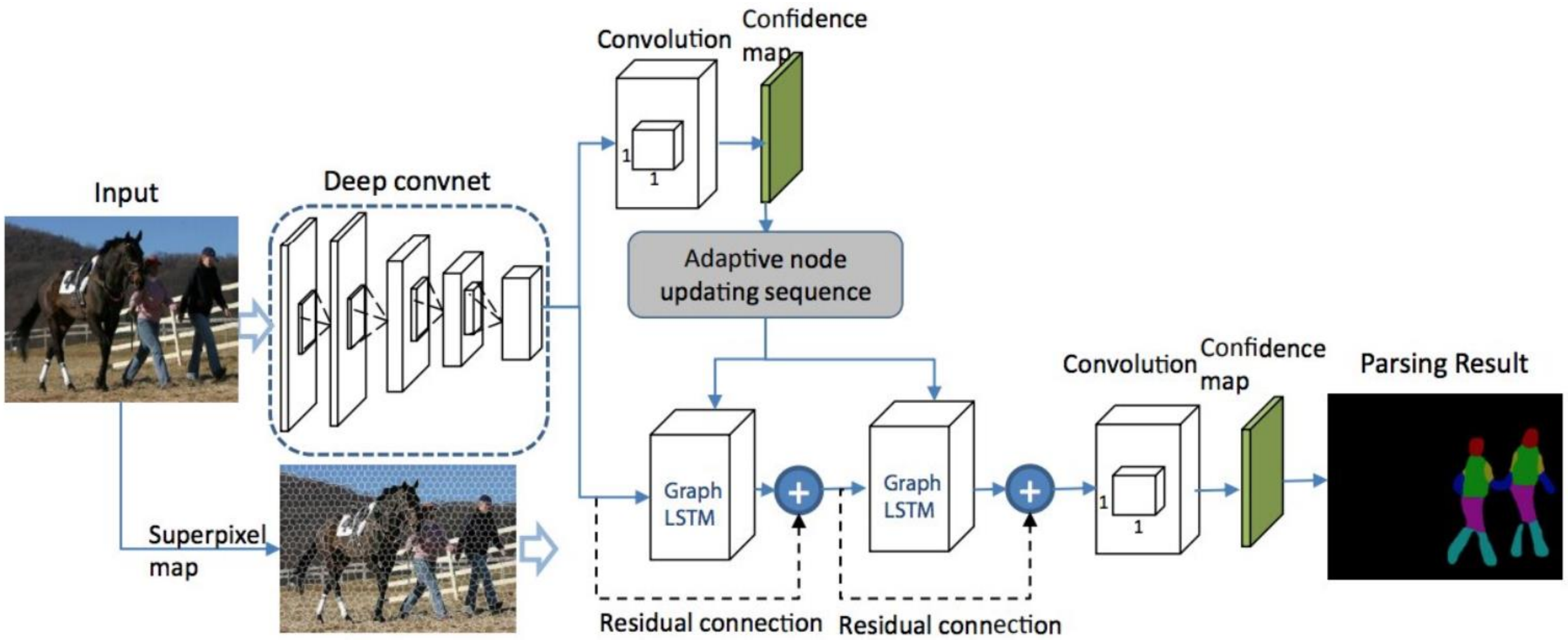
To perform image segmentation with the ReSeg model, ReNet layers are stacked on top of pre-trained VGG-16 convolutional layers that extract generic local features.

ReNet layers are then followed by up-sampling layers to recover the original image resolution in the final predictions.

Gated Recurrent Units (GRUs) are used because they provide a good balance between memory usage and computational power.

Graph-LSTM

Liang 2016



A semantic segmentation model based on the Graph LSTM network is proposed. (Graph LSTM : A generalization of LSTM from sequential data or multidimensional data to general graph-structured data).

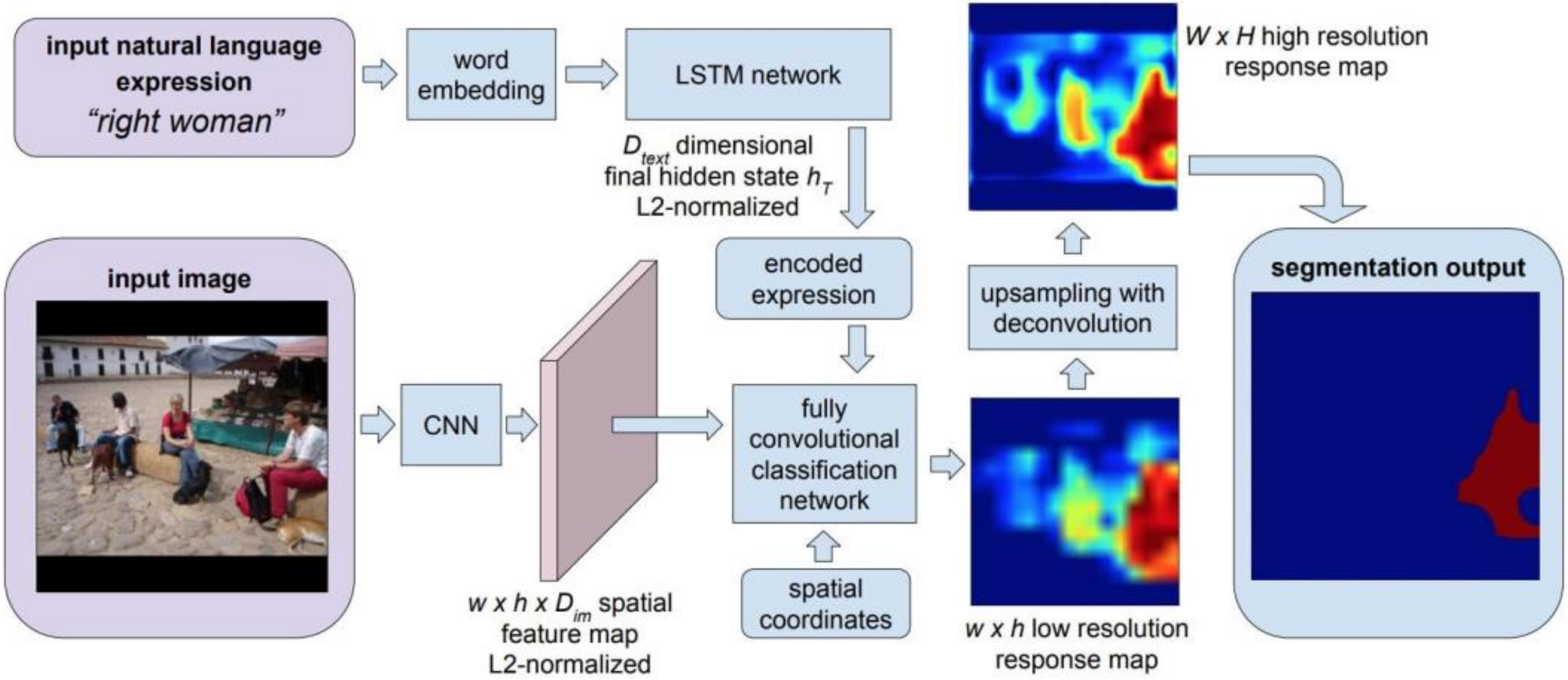
Instead of evenly dividing an image to pixels or patches in existing multi-dimensional LSTM structures (e.g., row, grid and diagonal LSTMs), they take each arbitrary-shaped super pixel as a semantically consistent node, and adaptively construct an undirected graph for the image, where the spatial relations of the super pixels are naturally used as edges.

To adapt the Graph LSTM model to semantic segmentation, LSTM layers built on a super-pixel map are appended on the convolutional layers to enhance visual features with global structure context.

CNN+LSTM

R. Hu...2016

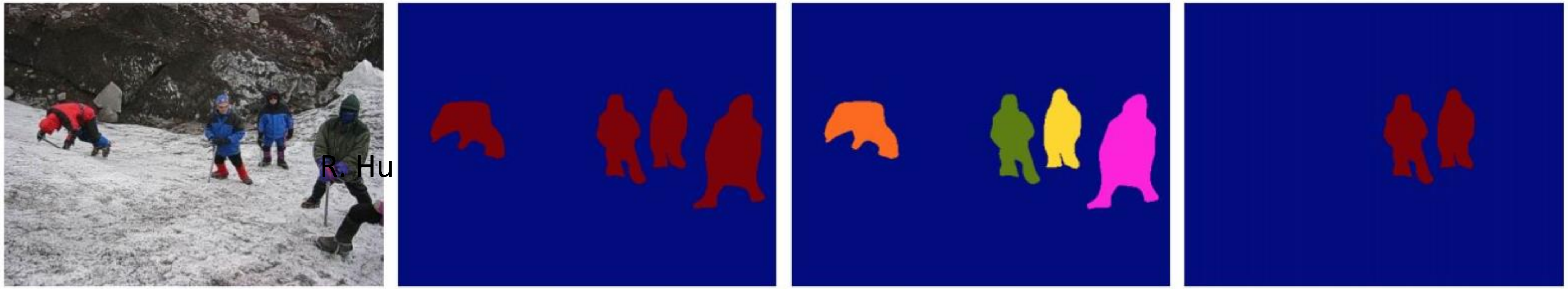
The CNN+LSTM architecture for segmentation from natural language expressions.


input natural language expression
"right woman"
word embedding
LSTM network
D_{text} dimensional final hidden state h_T L2-normalized
W x H high resolution response map
input image
CNN
w x h x D_{im} spatial feature map L2-normalized
encoded expression
fully convolutional classification network
spatial coordinates
upsampling with deconvolution
w x h low resolution response map
segmentation output

To produce pixel-wise segmentation for language expression, they propose an end-to-end trainable recurrent and convolutional model that jointly learns to process visual and linguistic information.

In the considered model, a recurrent LSTM network is used to encode the referential expression into a vector representation, and an FCN is used to extract a spatial feature map from the image and output a spatial response map for the target object.

An example segmentation result of this model (for the query "people in blue coat”) is shown.



R. Hu

(a) input image

(b) object class segmentation of class ***people***

(c) object instance segmentation of class ***people***

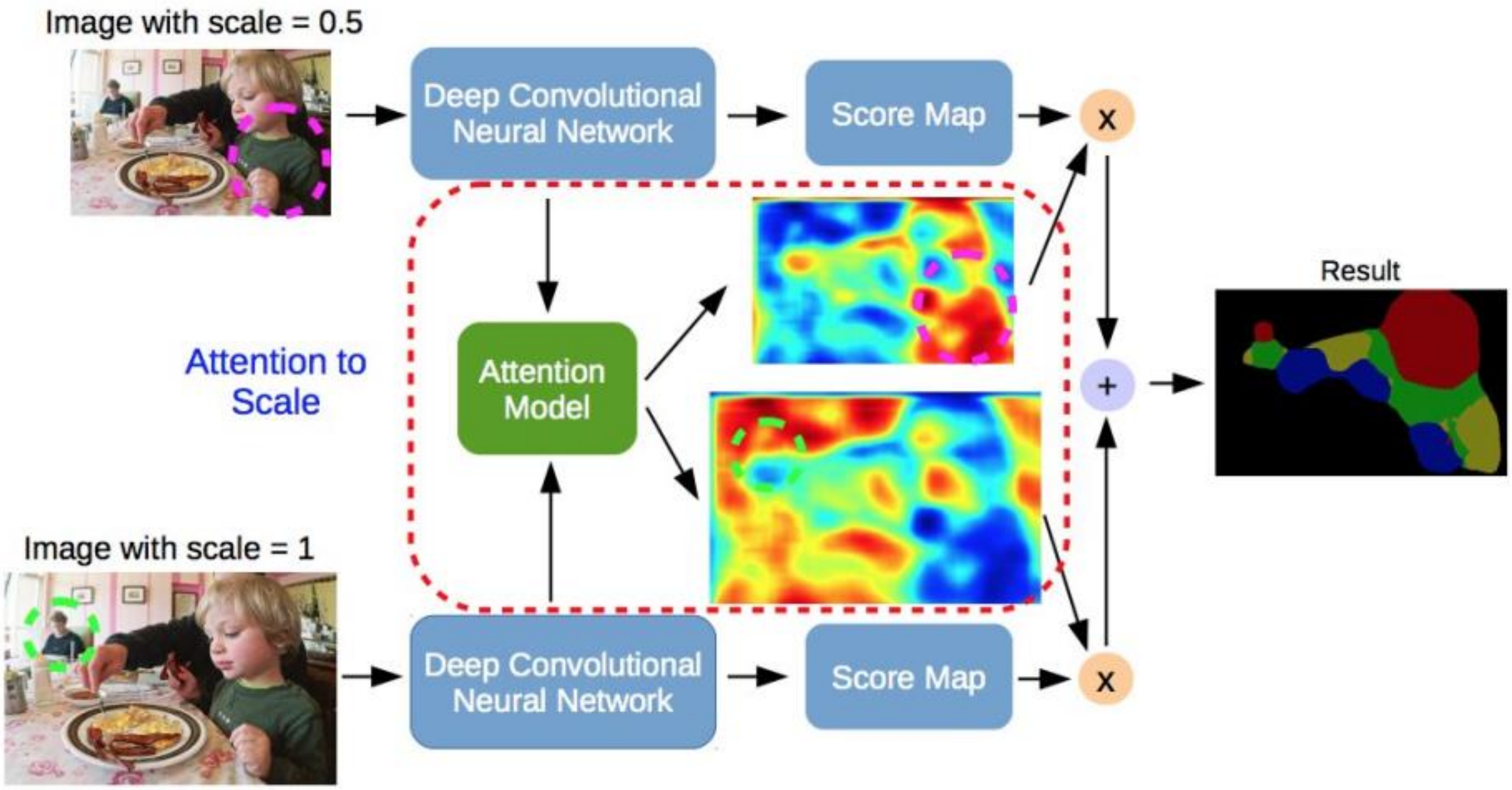
(d) segmentation from expression *“people in blue coat”*

Segmentation masks generated for the query “people in blue coat”

8. Attention-Based Models

Attention-based semantic segmentation model

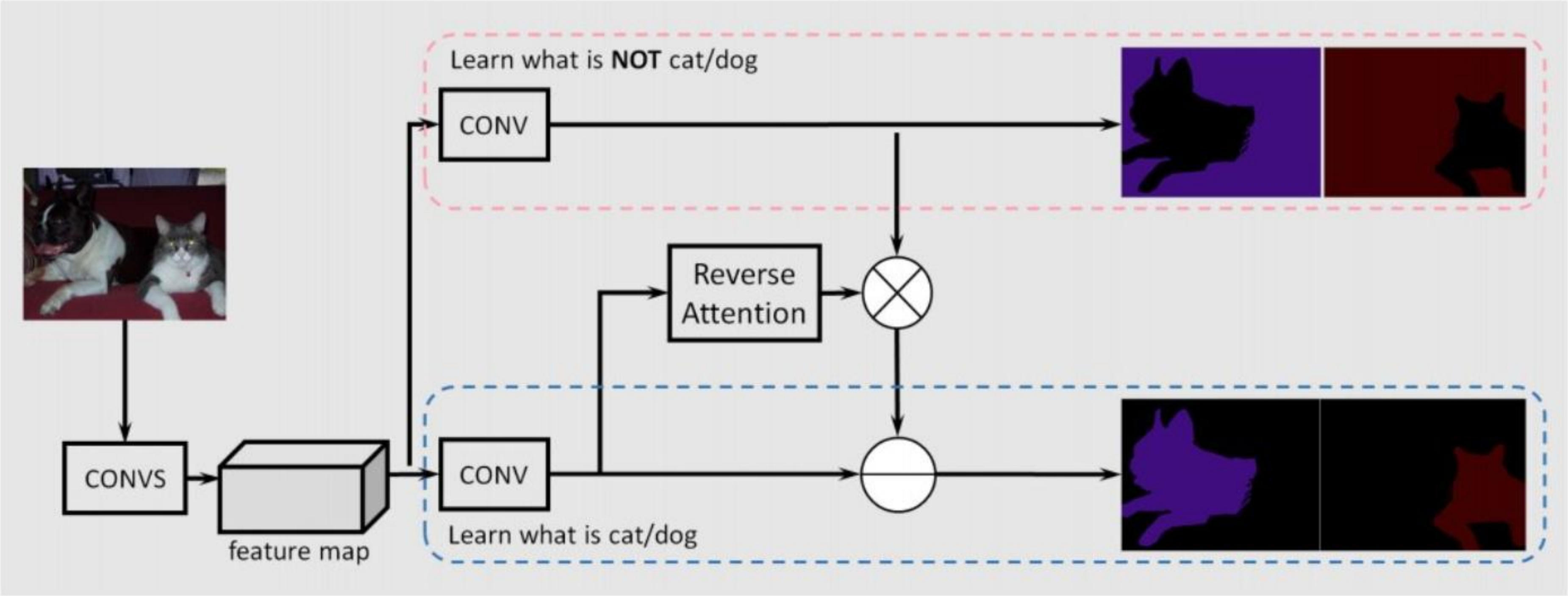
Chen ...2016



The attention model learns to assign different weights to objects of different scales.
e.g., the model assigns large weights on the small person (green dashed circle) for features from scale 1.0, and large weights on the large child (magenta dashed circle) for features from scale 0.5.

Semantic segmentation with reverse attention

Huang....2017



In contrast to other works in which convolutional classifiers are trained to learn the representative semantic features of labeled objects, Huang *et al* proposed a semantic segmentation approach using reverse attention mechanisms. Their Reverse Attention Network (RAN) architecture trains the model to capture the opposite concept (i.e., features that are not associated with a target class) as well. The RAN is a three-branch network that performs the direct, and reverse-attention learning processes simultaneously.

Other categories

9 Generative Models and Adversarial Training

C. Yu.. 2018:“Learning a discriminative feature network for semantic segmentation”

N. Souly....2016, “Semi supervised semantic segmentation using generative adversarial network’”

C. Hung....2018“Adversarial learning for semi-supervised semantic segmentation,”
Y. Xue.....2018 “Segan: Adversarial network with multi-scale loss for medical image segmentation,”

Majurski...2019“Cell image segmentation using generative adversarial networks, transfer learning, and augmentations,”

10 CNN Models With Active Contour Models
Chen...2019“Learning active contour models for medical image segmentation,”
Le...1028 “Reformulating level sets as deep recurrent neural network approach to semantic segmentation,”

Rupprecht...2016 “**Deep active contours**,”

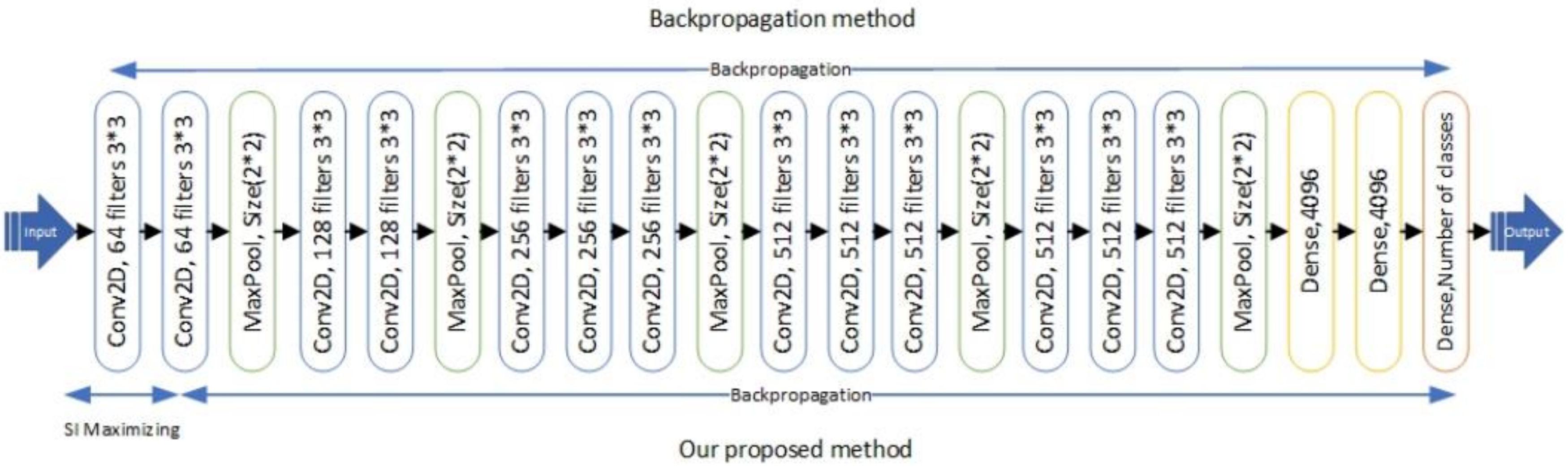
3.6 Layer Wise Design Algorithms_part2

Layer-wise forward learning

3.6.1 Forward learning in the first layer

For classification problems but the method can be generalized for regression problems

Forward learning in the first layer



$$\{\boldsymbol{x}^q\}_{q=1}^{Q} \quad \xrightarrow{\text{Filters: } \{(\boldsymbol{\theta}_l, b_l)\}_{l=1}^{M} + \text{Relu}} \quad \{\mathbf{z}^q\}_{q=1}^{Q}$$

Input layer Quasi - LS First conv.layer

Figure 3: Applying M convolutional filters and biases to the input patterns and then activating the convolved units by Relu function the feature maps are resulted.

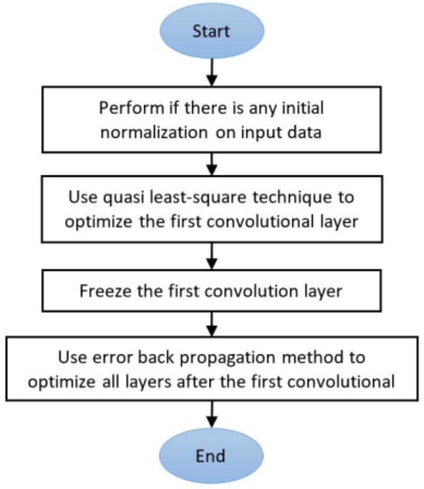

Start
Perform if there is any initial normalization on input data
Use quasi least-square technique to optimize the first convolutional layer
Freeze the first convolution layer
Use error back propagation method to optimize all layers after the first convolutional
End

Figure 2: This flowchart indicates the learning method in which the quasi-LS is utilized to learn the first convolution layer, and other layers are learned by an error backpropagation method.

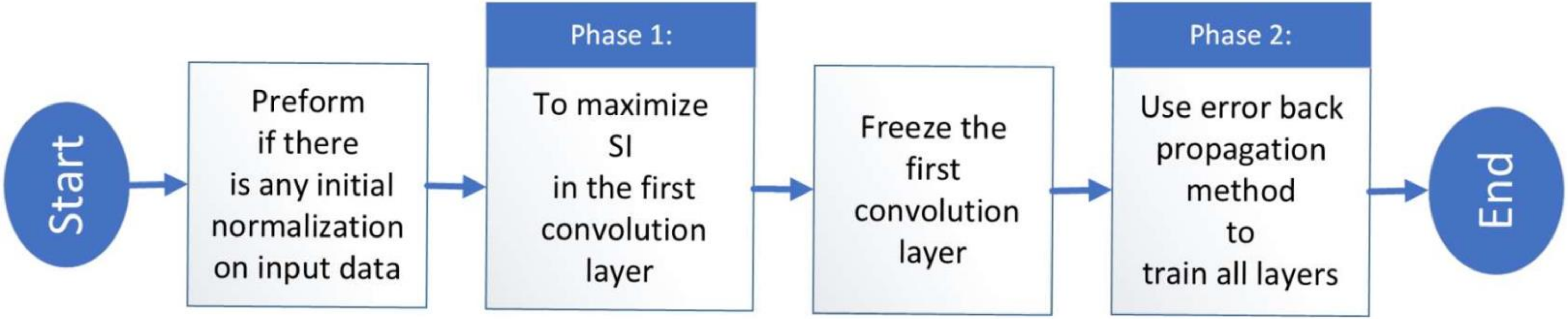


Figure 2. This flowchart indicates the learning method in which the quasi-LS is utilized to learn the first convolution layer (phase 1), and other layers are learned by an error backpropagation algorithm (phase 2).

Algorithm to train the first layer by Quasi Least Square (QLS) technique

1. Define the matrix of patches from the input data: X
2. Subtract the matrix from their mean: $\tilde{X} = X - mean(X)$
3. Define Auto-correlation matrix of patches: $W_{np} = \tilde{X}^{T}\tilde{X}$
4. Apply SVD technique to get Eigen values and Eigen vectors: $W = \sum_{i=1}^{n_p} \lambda_i v_i v_i^T$
5. Rank Eigen vectors by their Eigen values and then initiate all filters parameters and the biases by Eigen vectors, their negatives and the mean of patches.

   $$\left\{\{\boldsymbol{\theta}^j, b^j\}\right\}_{j=1}^{n_F} \quad n_F\text{:number of filters}$$

6. Based on the filter parameters at the first (convolutional)layer compute the second layer and for each example(anchor) find nearest neighbor for both positive and negative examples and define triplet loss for all data batch.

   $$J_{triplet} = \sum_{i=1}^{m}\left(\left\|x_i^2 - x_{ipos}^2\right\|^2\right)\text{-}\sum_{i=1}^{m}\left(\left\|x_i^2 - x_{ineg}^2\right\|^2\right)$$

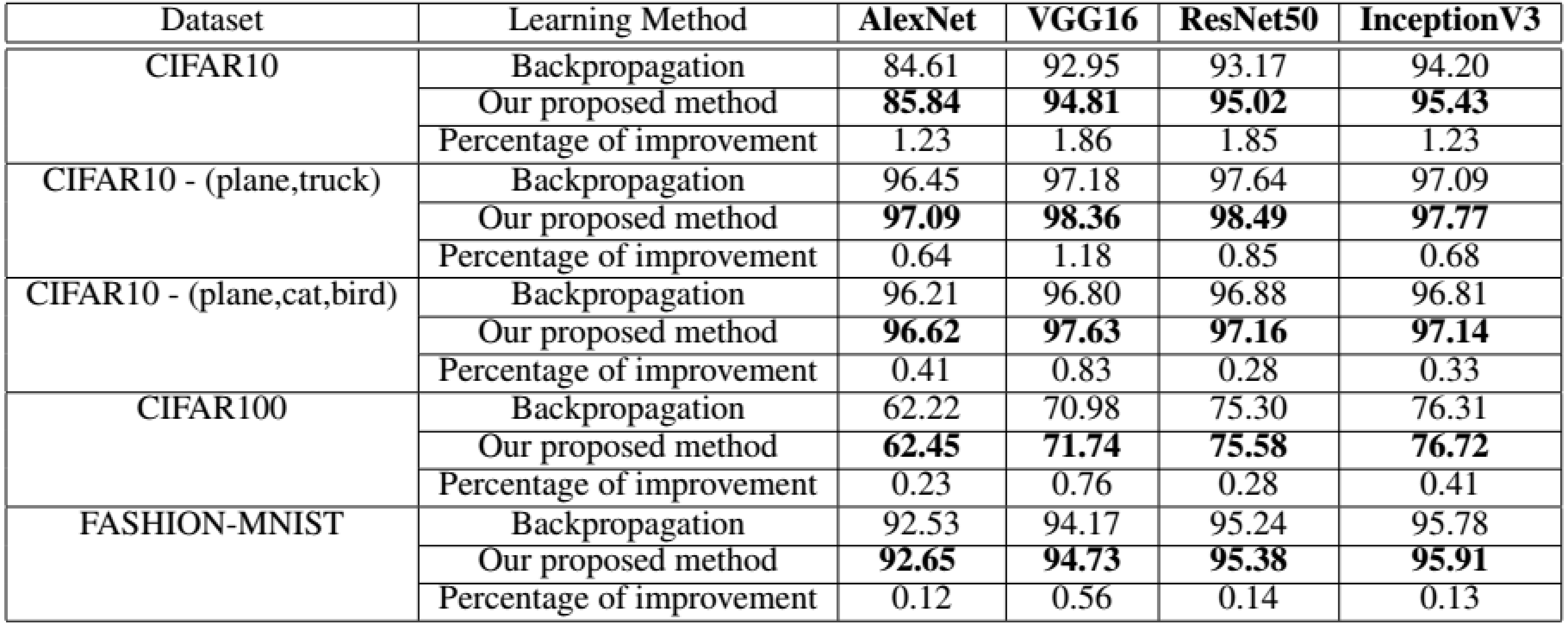
7. Use a QLS technique to minimize $J_{triplet}$ and update all filters.

   $$\boldsymbol{\theta}^j := \boldsymbol{\theta}^j + \mu\,\tilde{\boldsymbol{\theta}}^j \qquad b^j := b^j + \mu\tilde{b}^j \qquad \mu\text{=learning rate}$$

8. Repeat steps 6 and 7 for several epochs to get maximum possible SI on the second layer.

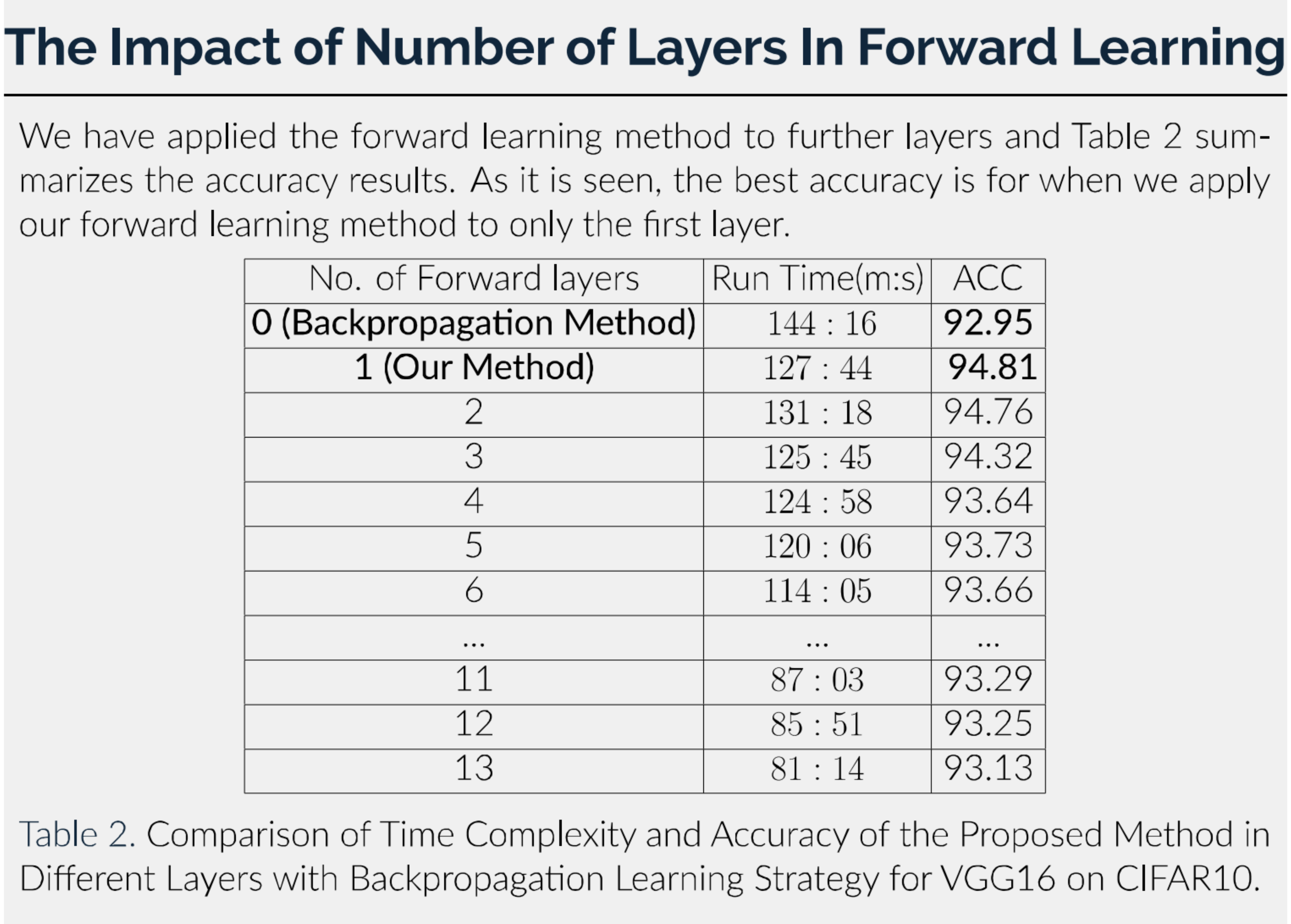
Now, one can train further layers by Backpropagation Algorithms.

Comparison between our method and backpropagation algorithm



Dataset	Learning Method	**AlexNet**	**VGG16**	**ResNet50**	**InceptionV3**
CIFAR10	Backpropagation	84.61	92.95	93.17	94.20
	Our proposed method	**85.84**	**94.81**	**95.02**	**95.43**
	Percentage of improvement	1.23	1.86	1.85	1.23
CIFAR10 - (plane,truck)	Backpropagation	96.45	97.18	97.64	97.09
	Our proposed method	**97.09**	**98.36**	**98.49**	**97.77**
	Percentage of improvement	0.64	1.18	0.85	0.68
CIFAR10 - (plane,cat,bird)	Backpropagation	96.21	96.80	96.88	96.81
	Our proposed method	**96.62**	**97.63**	**97.16**	**97.14**
	Percentage of improvement	0.41	0.83	0.28	0.33
CIFAR100	Backpropagation	62.22	70.98	75.30	76.31
	Our proposed method	**62.45**	**71.74**	**75.58**	**76.72**
	Percentage of improvement	0.23	0.76	0.28	0.41
FASHION-MNIST	Backpropagation	92.53	94.17	95.24	95.78
	Our proposed method	**92.65**	**94.73**	**95.38**	**95.91**
	Percentage of improvement	0.12	0.56	0.14	0.13

Table 5: Comparing the accuracy of our proposed method in different architectures and datasets with backpropagation method



The Impact of Number of Layers In Forward Learning

We have applied the forward learning method to further layers and Table 2 summarizes the accuracy results. As it is seen, the best accuracy is for when we apply our forward learning method to only the first layer.

No. of Forward layers	Run Time(m:s)	ACC
0 (Backpropagation Method)	144 : 16	**92.95**
1 (Our Method)	127 : 44	**94.81**
2	131 : 18	94.76
3	125 : 45	94.32
4	124 : 58	93.64
5	120 : 06	93.73
6	114 : 05	93.66
...	...	...
11	87 : 03	93.29
12	85 : 51	93.25
13	81 : 14	93.13

Table 2. Comparison of Time Complexity and Accuracy of the Proposed Method in Different Layers with Backpropagation Learning Strategy for VGG16 on CIFAR10.

3.6.2 Layer-wise forward learning

Layer Wise Forward Learning

Algorithm

```
For L=1:$L_{final}$
    While $\mathrm{SI}\ (Data^{l})$ has tendency to be increased:
        For $l = 1{:}\,L$
            $\boldsymbol{g}_{\boldsymbol{p}^{l}} = \frac{\partial \mathrm{Loss}_{\mathrm{triplet}}(Data^{l})}{\boldsymbol{p}^{l}}$
            $\boldsymbol{p}^{l} := \boldsymbol{p}^{l} - \gamma \boldsymbol{g}_{\boldsymbol{p}^{l}}$
```

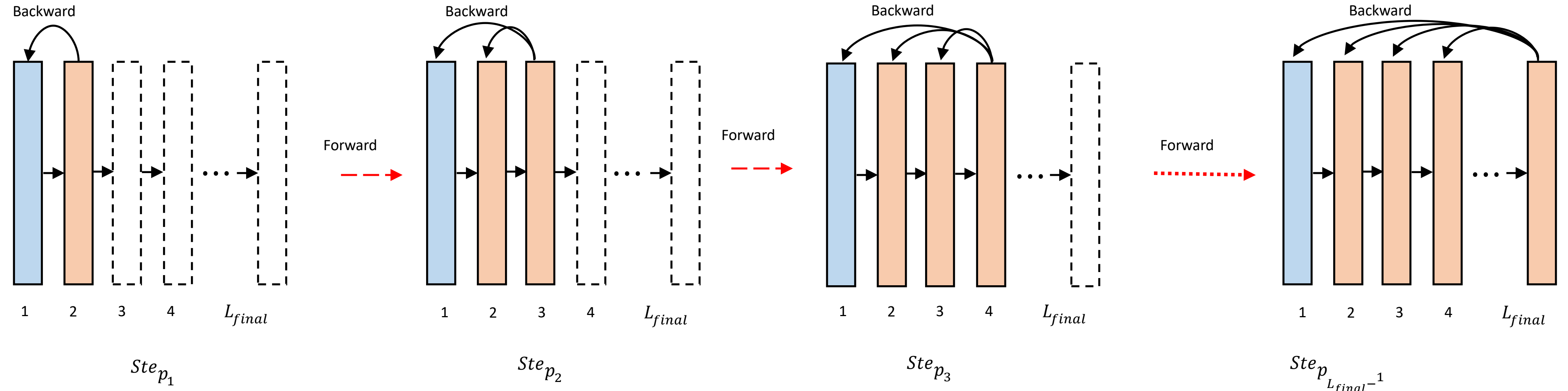
$Data^{l} = \{(\boldsymbol{x}_i^L, \boldsymbol{y}_i^L)\}$ $\quad Data\ at\ layer\ l$

Triplet Loss

$$\mathrm{Loss}_{\mathrm{triplet}}(Data^{L}) = \frac{1}{m}\sum_{i=1}^{m}\left(\left\|\boldsymbol{x}_i^L - \boldsymbol{x}_{i_p^*}^L\right\|^2 - \left\|\boldsymbol{x}_i^L - \boldsymbol{x}_{i_n^*}^L\right\|^2\right)$$

$$i_p^* = \underset{\forall q \neq i}{\arg\ min} \frac{1}{\delta(\boldsymbol{y_i}, \boldsymbol{y}_q) + \varepsilon}\left\|\boldsymbol{x}_i^L - \boldsymbol{x}_q^L\right\|^2 \qquad \varepsilon \to 0^{+}$$

$$i_n^* = \underset{\forall q \neq i}{\arg\ min} \frac{1}{1 - \delta(\boldsymbol{y_i}, \boldsymbol{y}_q) + \varepsilon}\left\|\boldsymbol{x}_i^L - \boldsymbol{x}_q^L\right\|^2$$



Comparison between our method and backpropagation algorithm

VGG۱٦	ResNet٥۹	InceptionV۳	EfficientNetB۰		
۹۲.۹۵	۹۳.۱۷	۹۴.۲۰	۹۸.۱۱	Backpropagation	CIFAR۱۰
۹۳.٦۷	**۹٤.٦٥**	**۹٤.٦۳**	**۹۸.۲٤**	Our method	
۷۰.۹۸	۷۵.۳۰	۷۶.۳۱	۸۸.۱۴	Backpropagation	CIFAR۱۰۰
۷۱.۱۹	**۷٥.٤۷**	**۷٦.٤۹**	**۸۸.۱۷**	Our method	
۹۴.۱۷	۹۵.۳۸	۹۵.۹۱	۹۷.۲۳	Backpropagation	Fashion-MNIST
۹٤.٥۲	**۹٦.٦۱**	**۹٦.۲۱**	**۹۷.٤۱**	Our method	

index is feasible by training the first layers with the Forward learning approach.

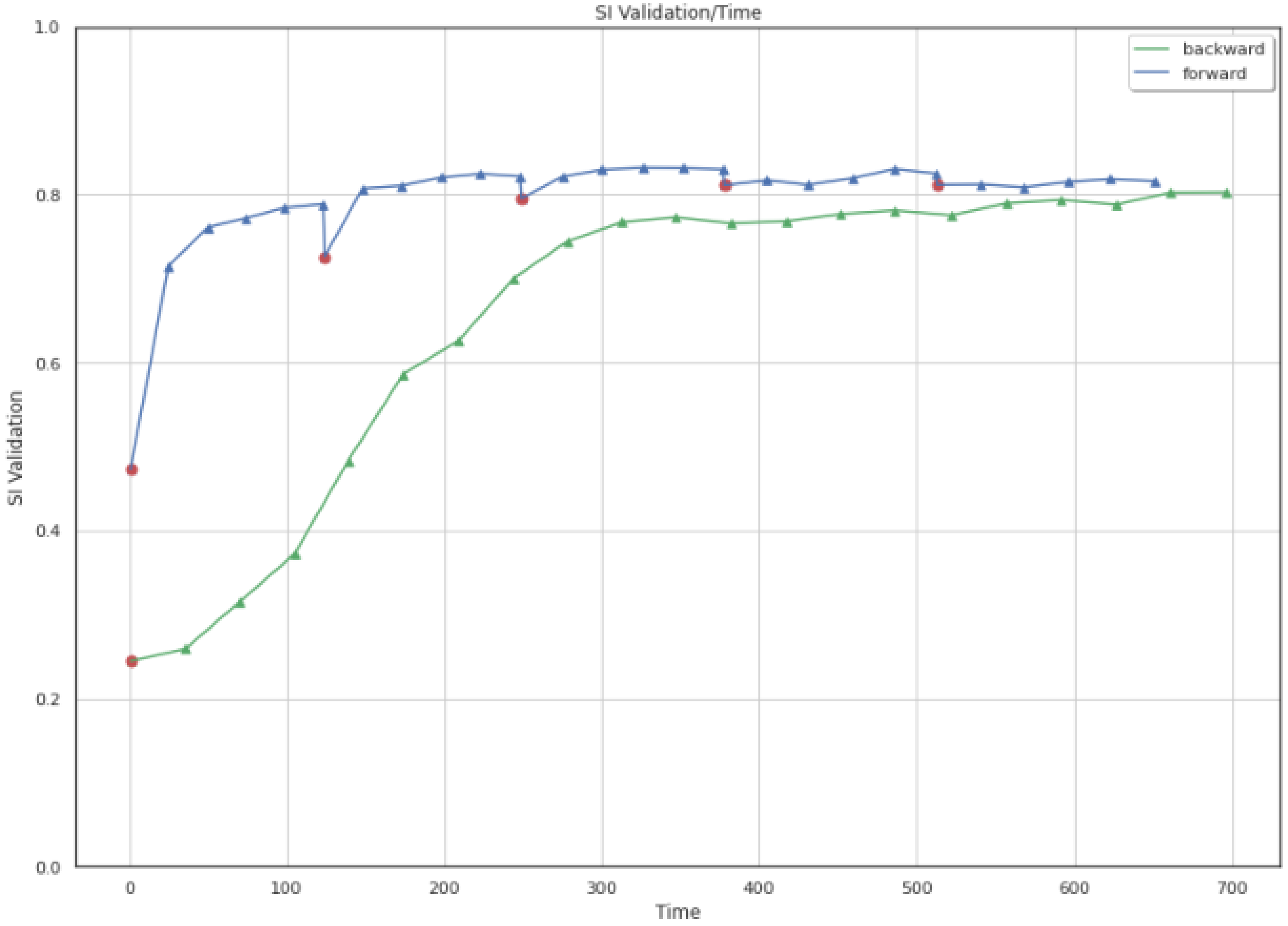
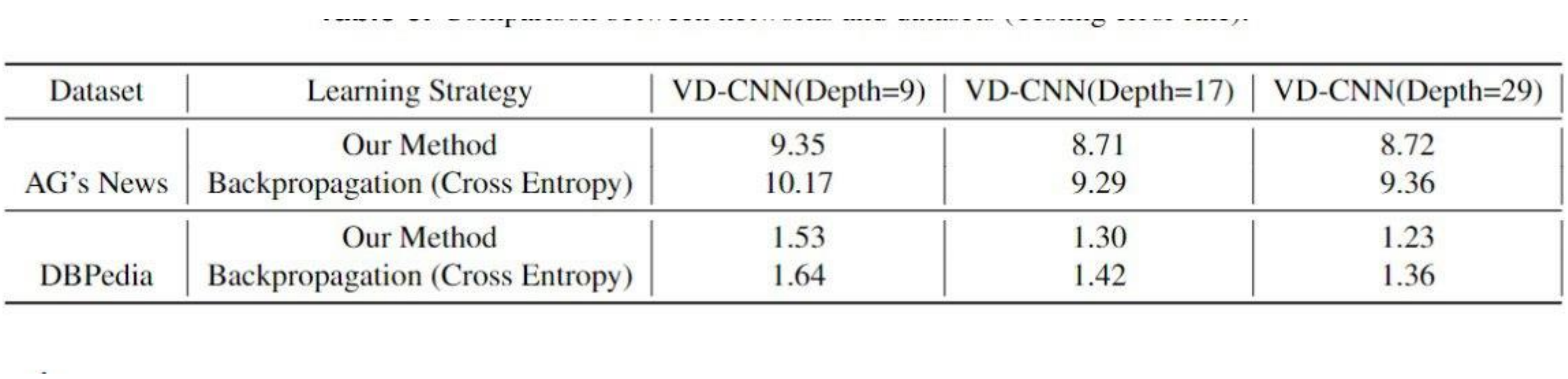


Figure 2. Comparing the separation index of the forward learning and backward learning approaches over time. Each triangle shows a single epoch in the learning process and the red dots in the Forward learning method are convolutional layers.

The Text Classification Error rate



Dataset	Learning Strategy	VD-CNN(Depth=9)	VD-CNN(Depth=17)	VD-CNN(Depth=29)
AG’s News	Our Method Backpropagation (Cross Entropy)	9.35 10.17	8.71 9.29	8.72 9.36
DBPedia	Our Method Backpropagation (Cross Entropy)	1.53 1.64	1.30 1.42	1.23 1.36