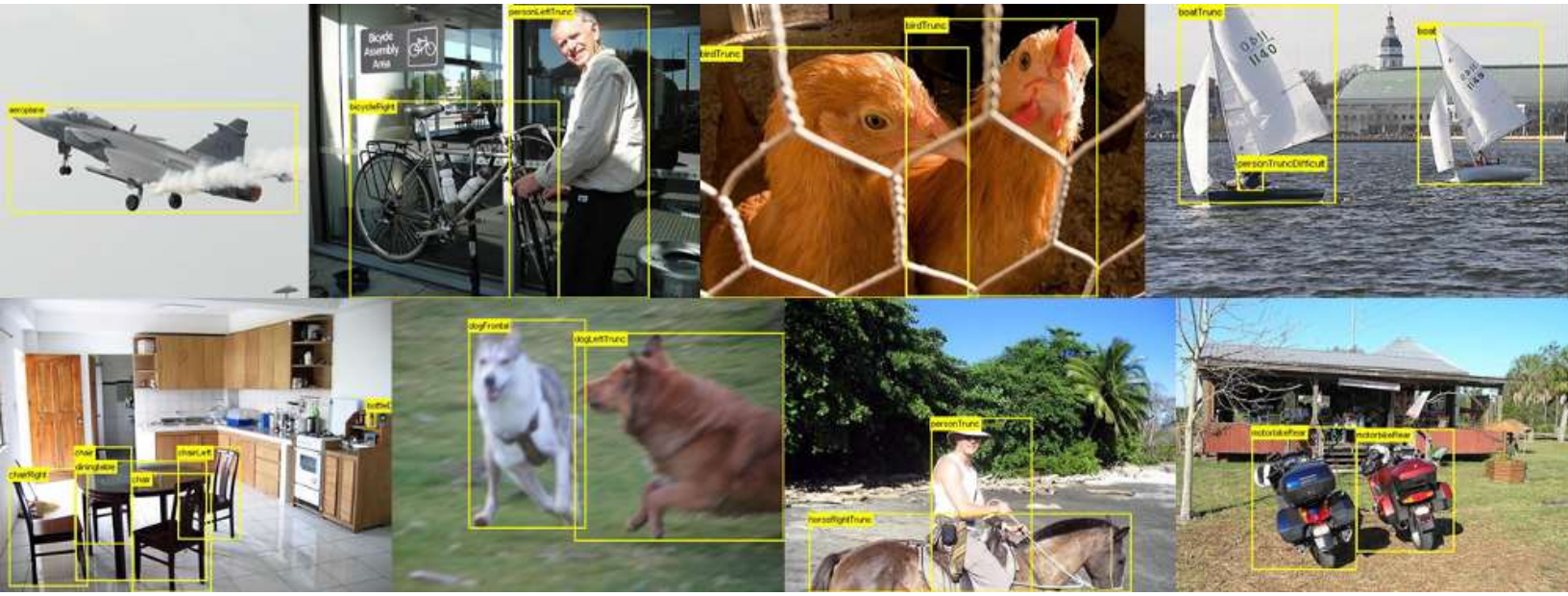
Review: ION —Inside-Outside Net, 2nd Runner Up in 2015 COCO Detection (Object Detection)

Best Student Entry and 2nd Runner Up in 2015 MS COCO Detection Challenge

SH Tsang Follow
Jan 2 · 7 min read



In this story, **ION (Inside-Outside Net)**, by **Cornell University** and **Microsoft Research**, is reviewed. By using skip pooling to extract information at multiple scales and levels of abstraction, and recurrent neural network (RNN) to capture the context feature, ION obtains **Best Student Entry** and **3rd Place** in **2015 MS COCO Detection Challenge**.

- **Inside**: Skip connections with L2 normalization.
- **Outside**: Stacked 4-direction RNNs for context.

As ION utilized a simple RNN called IRNN, it is also a good start to learn about RNN. It is published in **2016 CVPR** with more than **300 citations**. (SH Tsang @ Medium)

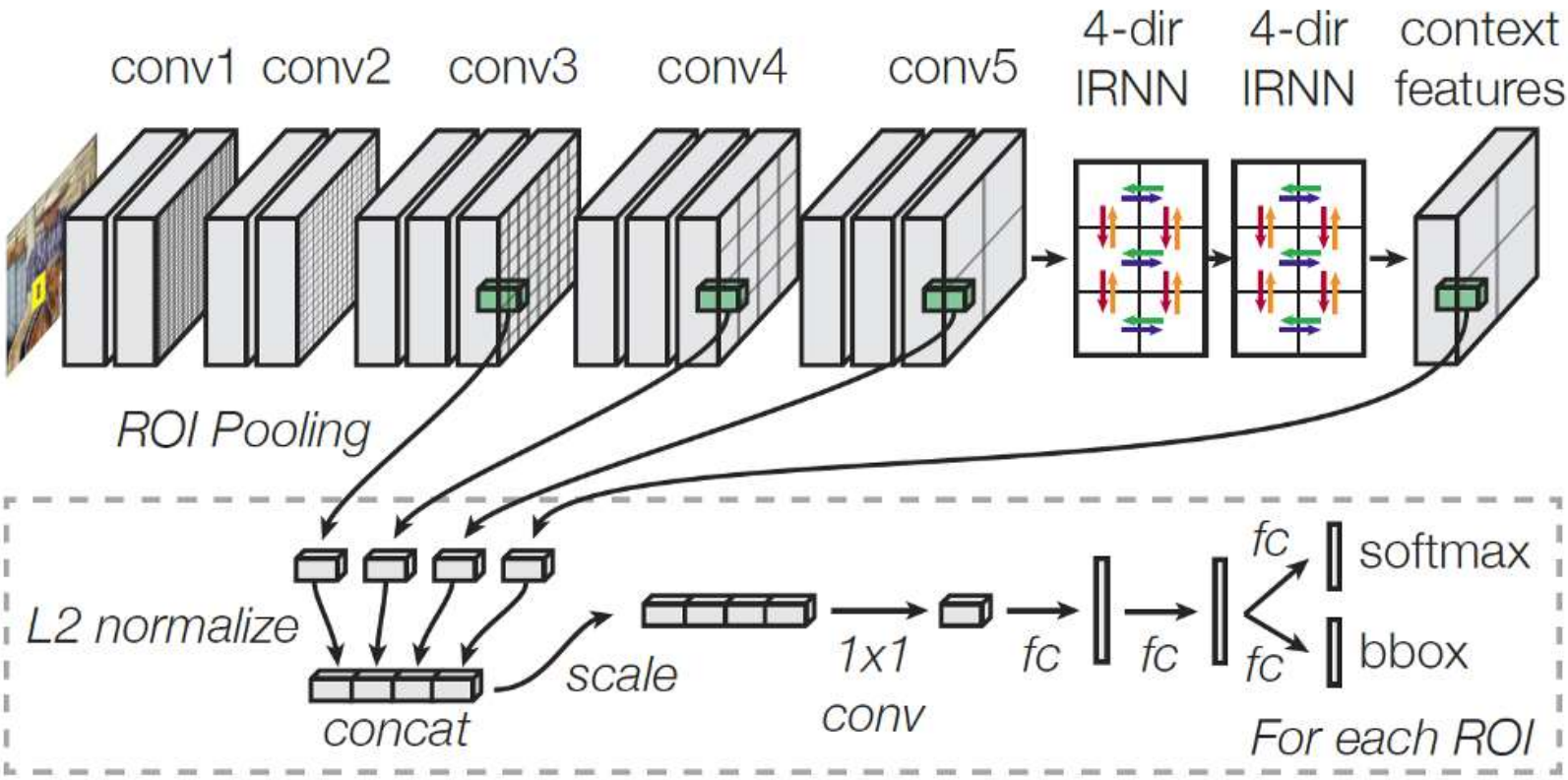
. . .

What Are Covered

1. **ION Architecture**
2. **Skip Pooling**
3. **From Vanilla RNN to 4-Direction IRNN**
4. **Some Design Evaluation**
5. **Results**

. . .

1. ION Architecture



ION Architecture

1.1. Fast R-CNN

In original Fast R-CNN, ROI pooling is performed at conv5 only.

- For small objects, the footprint on conv5 might only cover a 1×1 cell, which gets upsampled to 7×7.
- Only local features (inside the ROI) are used for classification.

1.2. ION

To address the above issues respectively,

- **Inside**: And the outputs of conv3 to conv5, they are L2 normalized, concatenated, re-scaled and dimension-reduced by 1×1 conv.
- **Outside**: At the output of conv5, there are 2 stacked IRNNs (A kind of RNN, that is composed of ReLUs and initialized with the identity matrix.) so as to make use of context features outside of the ROI pooling region.

It is noted that ION architecture is developed **based on Fast R-CNN using VGG16 backbone**. Thus, conv1 to conv5 at the beginning and fully connected (FC) layers at the end of the network, are from the original Fast R-CNN using VGG16 backbone.

. . .

2. Skip Pooling

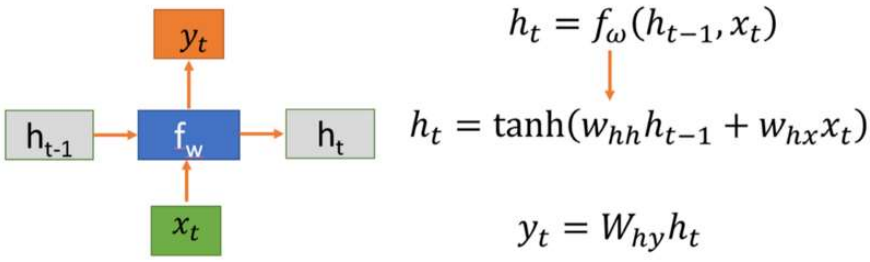
- **ROI mapping is performed at multiple layers, from conv3 to conv5**, as well as the **context feature computed by 2 IRNNs**.
- However, to match the input of FC layer, a size of $512\times7\times7$ shape is needed. Thereby, after concatenation, **1×1 convolution** is performed to **reduce the dimension** to $512\times7\times7$.
- But the **earlier layers usually have larger values than the later layers**, which is mentioned in ParseNet. Thus, each pooled ROI is **L2-normalized and re-scale back** up by empirically determined scale, prior to concatenation.

. . .

3. From Vanilla RNN to 4-Direction IRNN

3.1. Vanilla RNN (Plain Tanh RNN)

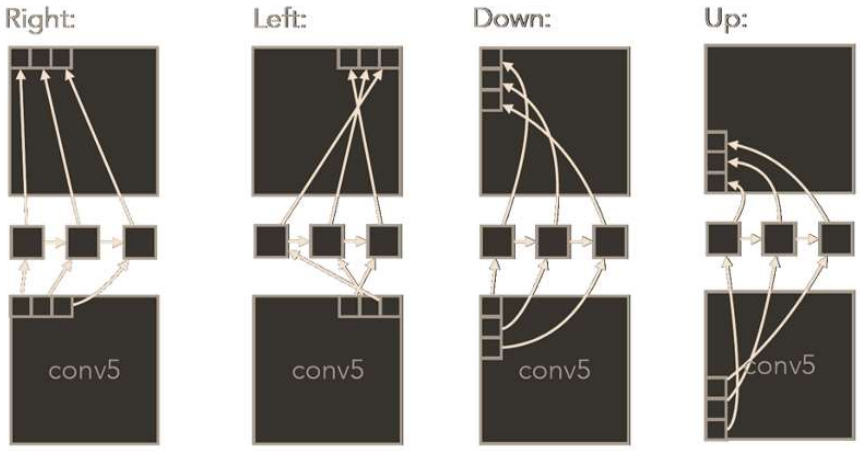
In **Vanilla RNN** (**Plain Tanh RNN**), tanh is used for activation:



3.2. 4-Direction IRNN

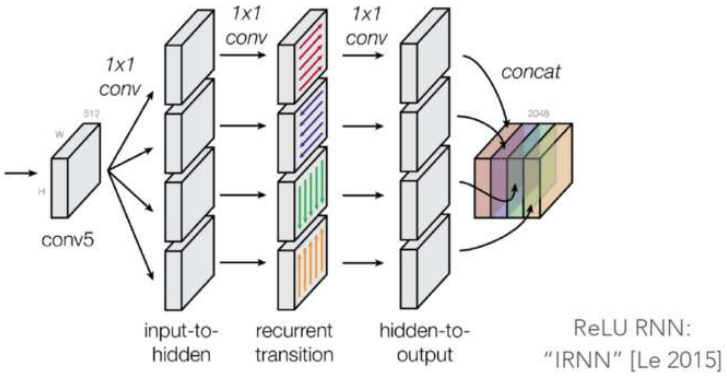
And the team from Prof. Hinton suggested IRNN, which is a kind of RNN, that is composed of ReLUs and initialized with the identity matrix. (If interested, you can visit the paper in arXiv, the paper called "A Simple Way to Initialize Recurrent Networks of Rectified Linear Units".)

Since it is done within an image, it is a lateral RNN. An example is as shown below. This step is repeated for each row (Right/Left) or column (Down/Up).

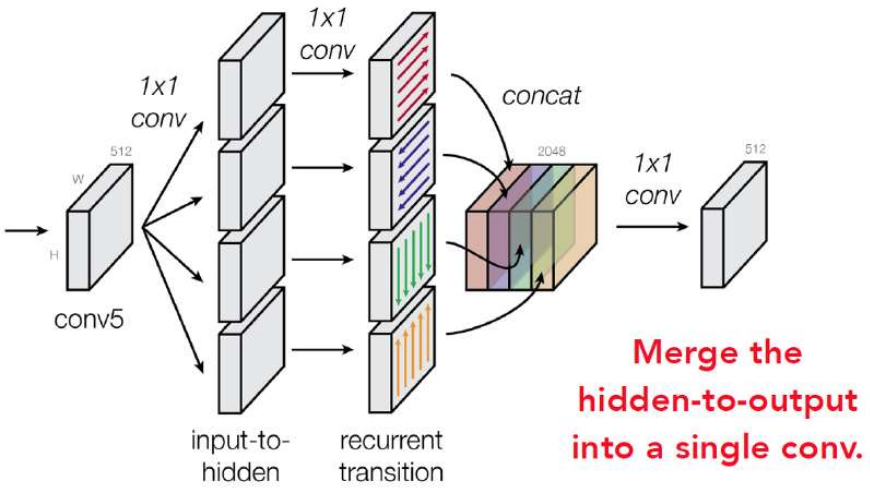


And **a naive version of 4-direction IRNN** using the ReLU-based IRNN is as follows:

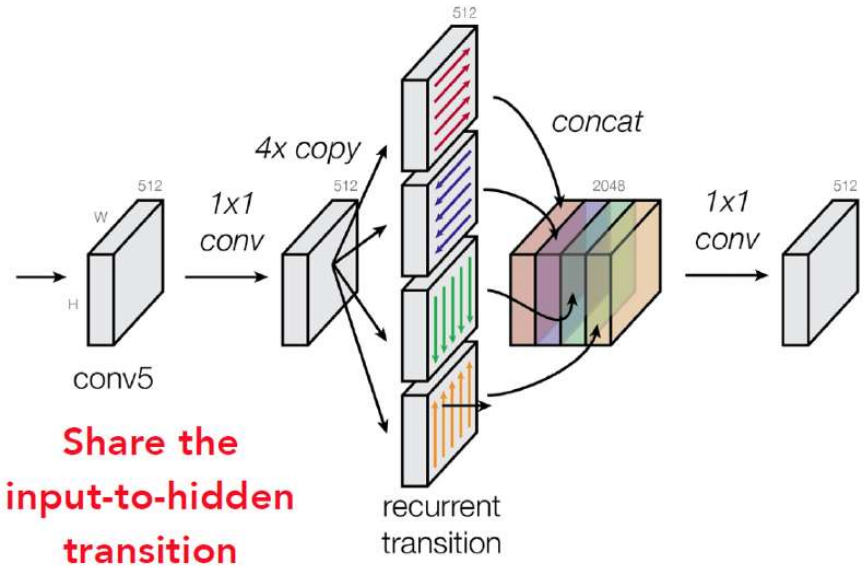
$$h_t^{\text{right}} = \max(W_{hh}^{\text{right}} h_{t-1}^{\text{right}} + W_{xh}^{\text{right}} x_t, 0) \qquad y_t^{\text{right}} = \max(W_{hy}^{\text{right}} h_t^{\text{right}}, 0)$$



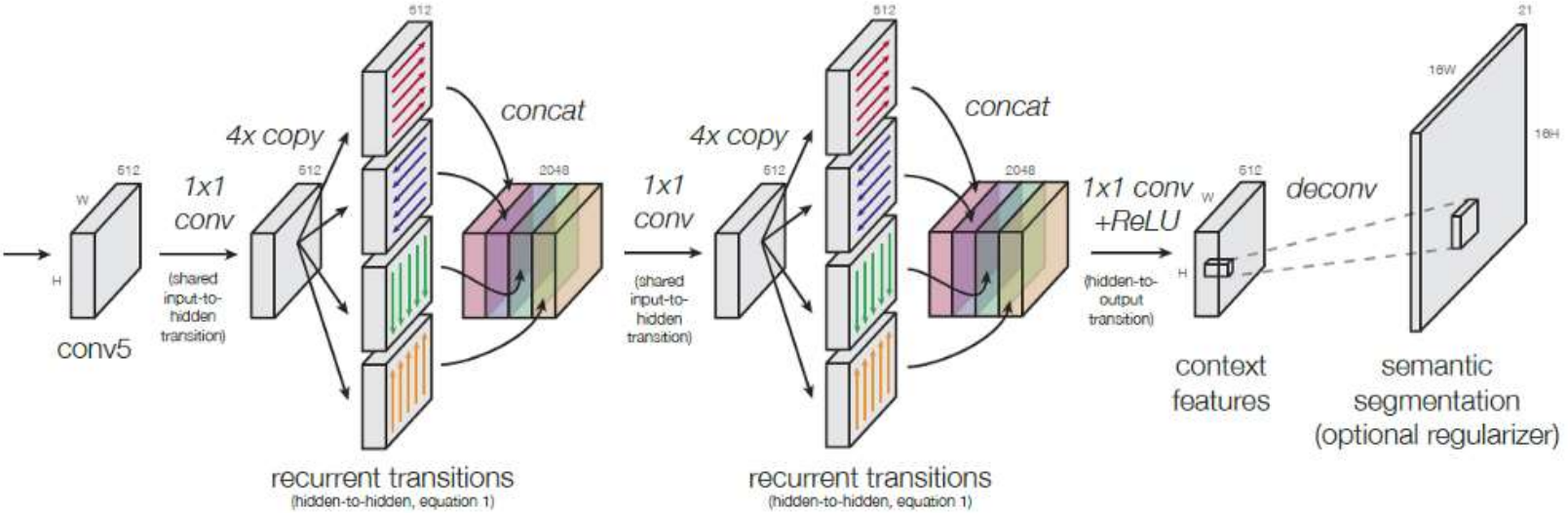
The equations above are those for right-direction IRNN, similar for left, up and down directions. To make it efficient, authors simplify the above 4-direction IRNN. To simply, first, **the hidden-to-output is merged into a single conv, i.e. concatenation followed by 1×1 convolution**:



Second, the input-to-hidden is also simplified by **1×1 convolution and shared the conv with 4 recurrent transitions**:



Finally, **two modified IRNNs are stacked together** to extract the context features:



Two Stacked IRNN

After the above modifications, the RNN equation becomes:

$$h_{i,j}^{\text{right}} \leftarrow \max\left(\mathbf{W}_{hh}^{\text{right}} h_{i,j-1}^{\text{right}} + h_{i,j}^{\text{right}}, 0\right)$$

As we can see, **there is no input *x*, because the input is implicitly done by the 1×1 convolution.** And it is found that the ***W* can also be removed due to similar detection performance**:

$$h_{i,j}^{\text{right}} \leftarrow \max\left(h_{i,j-1}^{\text{right}} + h_{i,j}^{\text{right}}, 0\right)$$

This is like an accumulator, but with ReLU after each step. Hence, the IRNN consist of repeated steps of: accumulate, ReLU, accumulate, etc. Note that this is not the same as an integral/area image, since each step has ReLU.

3.3. Segmentation Loss

As shown in the figure above, since the object detection dataset also contains the semantic segmentation label, during training, **semantic segmentation is also performed using context features.** Just like FCN, The deconvolution upsamples by 16× with a 32×32 kernel, and an extra softmax loss layer is added with a weight of 1. This loss is acted as a **regularizer**.

During testing, this segmentation path is removed. Therefore, inference time is the same as the network trained without segmentation loss.

3.4. Two-Stage Training

As mentioned, the network is based on Fast R-CNN using VGG16 as backbone, thus, pre-trained VGG16 on ImageNet are used for those common layers. And the network is trained with conv1 to conv5 frozen. Then, the network is trained with conv1 and conv2 frozen.

. . .

4. Some Design Evaluation

Ablation study is performed on PASCAL VOC 2007 dataset. All are trained using union of 2007 trainval and 2012 trainval, and tested on 2007 test set.

4.1. Pool From Which Layers?

ROI pooling from:				Merge features using:	
C2	C3	C4	C5	1x1	L2+Scale+1x1
			✓	*70.8	71.5
		✓	✓	69.7	74.4
	✓	✓	✓	63.6	**74.6**
✓	✓	✓	✓	59.3	**74.6**

Combining Features from Different Layers.

- As shown above, combining conv3, conv4 and conv5 with L2 normalization has 74.6% mAP which is the highest performance.

4.2. How should we normalize feature amplitude?

L2 Normalization method	Seg.	Scale:	
		Learned	Fixed
Sum across channels	✓	76.4	76.2
Sum over all entries	✓	76.5	**76.6**

Approaches to Normalizing Feature Amplitude.

- As in ParseNet, it sums over channels and performs one normalization per spatial location. Or instead, sum over all entries in each pooled ROI and normalize it as a single blob.
- During scaling, a fixed scale or a learnt scale per channel?

- It is found that all of these approaches perform about the same.

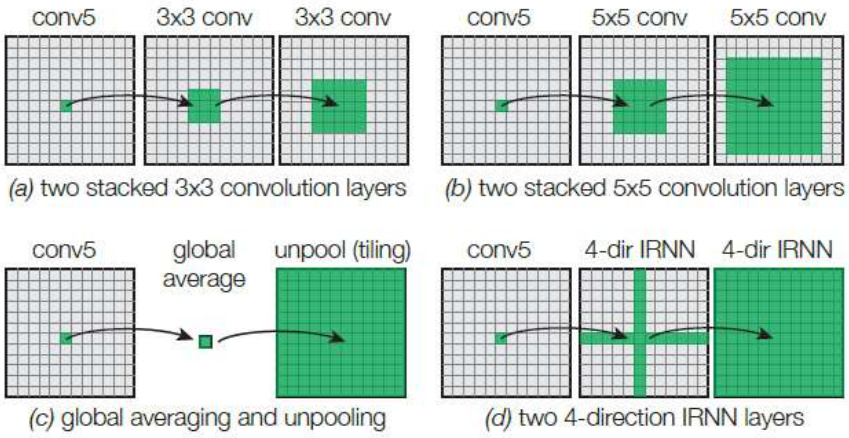
4.3. How much does segmentation loss help?

ROI pooling from:					Use seg. loss?	
C2	C3	C4	C5	IRNN	No	Yes
				✓	69.9	70.6
			✓	✓	73.9	74.2
		✓	✓	✓	75.1	76.2
	✓	✓	✓	✓	75.6	76.5
✓	✓	✓	✓	✓	74.9	**76.8**

Effect of Segmentation Loss.

- Performance is consistently better with segmentation loss as shown above.

4.4. How should we incorporate context?



Receptive Field of Different Layer Types.

Context method	Seg.	mAP
(a) 2x stacked 512x3x3 conv		74.8
(b) 2x stacked 256x5x5 conv		74.6
(c) Global average pooling		74.9
(d) 2x stacked 4-dir IRNN		**75.6**
(a) 2x stacked 512x3x3 conv	✓	75.2
(d) 2x stacked 4-dir IRNN	✓	**76.5**

Comparing Approaches to Adding Context.

- Different types of conv are tried after conv5. 2 stacked 3×3 and 2 stacked 5×5 are just further convolution. The receptive field is based on the filter size. For global average pooling, it has an input-size receptive field but the output is of the same value. For IRNN, it has an input-size receptive field, and the output value is varied at different positions.
- And of course, using IRNN obtains the highest mAP.

4.5. Which IRNN architecture?

ROI pooling from: C3	C4	C5	IRNN	Seg.	# units	Include W_{hh}? Yes	No
✓	✓	✓	✓	✓	128	76.4	75.5
✓	✓	✓	✓	✓	256	**76.5**	75.3
✓	✓	✓	✓	✓	512	**76.5**	76.1
✓	✓	✓	✓	✓	1024	76.2	76.4

Varying The Hidden Transition.

Variation	mAP
Our method	**76.5**
(a) Left-right then up-down	**76.5**
(b) Pool out of both IRNNs	75.9
(c) Combine 2x stacked 512x3x3 conv and IRNN	**76.5**

Other Variations.

- Including Whh and having 256 number of hidden units obtains the best results. And using only left and right directions IRNN for the first IRNN, and only up-down directions IRNN has the same results as two stacked 4-direction IRNN. Nevertheless, excluding

Whh, using 512 number of hidden units, and two stacked 4-direction IRNN are used finally.

. . .

5. Results

5.1. PASCAL VOC 2007

Method	Boxes	R	W	D	Train	Time	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
FRCN [9]	SS				07+12	0.3s	70.0	77.0	78.1	69.3	59.4	38.3	81.6	78.6	86.7	42.8	78.8	68.9	84.7	82.0	76.6	69.9	31.8	70.1	74.8	80.4	70.4
Faster [32]	RPN				07+12	0.2s	73.2	76.5	79.0	70.9	65.5	52.1	83.1	84.7	86.4	52.0	81.9	65.7	84.8	84.6	77.5	76.7	38.8	73.6	73.9	83.0	72.6
MR-CNN [8]	SS+EB		✓		07+12	30s	78.2	80.3	84.1	78.5	70.8	**68.5**	**88.0**	85.9	87.8	60.3	85.2	73.7	87.2	86.5	**85.0**	76.4	48.5	76.3	75.5	85.0	81.0
ION [ours]	SS				07+12	0.8s	74.6	78.2	79.1	76.8	61.5	54.7	81.9	84.3	88.3	53.1	78.3	71.6	85.9	84.8	81.6	74.3	45.6	75.3	72.1	82.6	81.4
ION [ours]	SS	✓			07+12	0.8s	75.6	79.2	83.1	77.6	65.6	54.9	85.4	85.1	87.0	54.4	80.6	73.8	85.3	82.2	82.2	74.4	47.1	75.8	72.7	84.2	80.4
ION [ours]	SS	✓	✓	✓	07+12	1.2s	77.6	79.7	83.4	78.1	65.7	62.0	86.5	85.8	88.8	60.2	83.4	75.1	86.5	87.3	82.1	79.7	48.3	77.0	75.3	85.3	82.4
ION [ours]	SS+EB	✓	✓	✓	07+12	2.0s	79.4	82.5	86.2	79.9	71.3	67.2	88.6	87.5	88.7	60.8	84.7	72.3	87.6	**87.7**	83.6	82.1	53.8	**81.9**	74.9	85.8	81.2
ION [ours]	SS	✓			07+12+S	0.8s	76.5	79.2	79.2	77.4	69.8	55.7	85.2	84.2	89.8	57.5	78.5	73.8	87.8	85.9	81.3	75.3	49.7	76.9	74.6	85.2	82.1
ION [ours]	SS	✓	✓		07+12+S	1.2s	78.5	80.2	84.7	78.8	72.4	61.9	86.2	86.7	89.5	59.1	84.1	74.7	88.9	86.9	81.3	80.0	50.9	80.4	74.1	86.6	83.3
ION [ours]	SS	✓	✓	✓	07+12+S	1.2s	79.2	80.2	85.2	78.8	70.9	62.6	86.6	86.9	89.8	61.7	86.9	76.5	88.4	87.5	83.4	80.5	52.4	78.1	**77.2**	**86.9**	**83.5**
ION [ours]	SS+EB	✓	✓	✓	07+12+S	2.0s	**80.1**	**84.2**	**87.2**	**82.1**	**74.8**	67.1	85.0	**88.0**	**89.3**	**60.4**	**86.1**	**76.3**	**88.7**	86.3	83.5	**82.2**	**55.5**	80.5	75.3	86.5	83.3

PASCAL VOC 2007 Test Set (07: 07 trainval, **12:** 12 trainval, **S:** Segmentation labels, **R:** 4-Dir IRNN, **W:** Two rounds of box regression and weighted voting, **D:** remove all dropout, **SS:** SelectiveSearch, **EB:** EdgeBoxes, **RPN:** region proposal network, **Time:** per image, excluding proposal generation.)

- With all techniques included, ION obtains 80.1% mAP, and it takes 2.0s per image which is much faster than the state-of-the-art MR-CNN of 30s.

5.2. PASCAL VOC 2012

Method	Boxes	R	W	D	Train	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
FRCN [9]	SS				07++12	68.4	82.3	78.4	70.8	52.3	38.7	77.8	71.6	89.3	44.2	73.0	55.0	87.5	80.5	80.8	72.0	35.1	68.3	65.7	80.4	64.2
Faster [32]	RPN				07++12	70.4	84.9	79.8	74.3	53.9	49.8	77.5	75.9	88.5	45.6	77.1	55.3	86.9	81.7	80.9	79.6	40.1	72.6	60.9	81.2	61.5
FRCN+YOLO [31]	SS				07++12	70.4	83.0	78.5	73.7	55.8	43.1	78.3	73.0	89.2	49.1	74.3	56.6	87.2	80.5	80.5	74.7	42.1	70.8	68.3	81.5	67.0
HyperNet [21]	RPN				07++12	71.4	84.2	78.5	73.6	55.6	53.7	78.7	79.8	87.7	49.6	74.9	52.1	86.0	81.7	83.3	81.8	48.6	73.5	59.4	79.9	65.7
MR-CNN [8]	SS+EB			✓	07+12	73.9	85.5	82.9	76.6	57.8	62.7	79.4	77.2	86.6	55.0	79.1	62.2	87.0	83.4	84.7	78.9	45.3	73.4	65.8	80.3	74.0
ION [ours]	SS	✓	✓	✓	07+12	74.7	86.9	84.5	75.2	58.2	57.7	80.5	78.3	90.4	54.4	79.9	60.5	88.4	83.0	83.0	81.2	50.7	77.3	67.6	83.5	72.3
ION [ours]	SS+EB	✓	✓	✓	07+12	76.4	88.0	84.6	77.7	63.7	**63.6**	80.8	80.8	90.9	55.5	81.9	60.9	89.1	84.9	84.2	83.9	53.2	79.8	67.4	84.4	72.9
ION [ours]	SS	✓	✓	✓	07+12+S	76.4	87.5	84.7	76.8	63.8	58.3	**82.6**	79.0	90.9	57.8	82.0	64.7	88.9	86.5	84.7	82.3	51.4	78.2	**69.2**	**85.2**	73.5
ION [ours]	SS+EB	✓	✓	✓	07+12+S	**77.9**	**88.3**	**85.7**	**80.5**	**67.2**	**63.6**	82.5	**82.0**	**91.4**	**58.2**	**84.1**	**65.3**	**90.1**	**87.3**	**85.0**	**84.4**	**53.6**	**80.7**	69.1	84.6	**74.7**

PASCAL VOC 2012 Test Set (07: 07 trainval, **12:** 12 trainval, **S:** Segmentation labels, **R:** 4-Dir IRNN, **W:** Two rounds of box regression and weighted voting, **D:** remove all dropout, **SS:** SelectiveSearch, **EB:** EdgeBoxes, **RPN:** region proposal network, **Time:** per image, excluding proposal generation.)

- Similar to VOC 2007, with all techniques included, ION obtains 77.9% mAP.

5.3. MS COCO

Method	Boxes	RW	D	Train	Avg. Precision, IoU: 0.5:0.95	0.50	0.75	Avg. Precision, Area: Small	Med.	Large	Avg. Recall, # Dets: 1	10	100	Avg. Recall, Area: Small	Med.	Large
FRCN [9]*	SS			train	20.5	39.9	19.4	4.1	20.0	35.8	21.3	29.5	30.1	7.3	32.1	52.0
FRCN [9]*	SS	✓		train	20.0	40.3	18.1	4.1	19.6	34.5	20.8	29.1	29.8	7.4	31.9	50.9
ION [ours]	SS	✓		train	23.0	42.0	23.0	6.0	23.8	37.3	23.0	32.4	33.0	9.7	37.0	53.5
ION [ours]	SS	✓	✓	train	23.6	43.2	23.6	6.4	24.1	38.3	23.2	32.7	33.5	10.1	37.7	53.6
ION [ours]	SS	✓	✓	train+S	24.9	44.7	25.3	7.0	26.1	40.1	23.9	33.5	34.1	10.7	38.8	54.1
ION [ours]	SS	✓	✓	train+S	24.6	46.3	23.3	7.4	26.2	38.8	23.7	33.9	34.6	11.7	40.0	53.8
ION comp.†	MCG+RPN	✓	✓	trainval35k+S	31.2	53.4	32.3	12.8	32.9	45.2	27.8	43.1	45.6	23.6	50.0	63.2
ION post.†	MCG+RPN	✓	✓	trainval35k+S	**33.1**	**55.7**	**34.6**	**14.5**	**35.2**	**47.2**	**28.9**	**44.8**	**47.4**	**25.5**	**52.4**	**64.3**

MS COCO 2015 Test-Dev (**comp:** competition submission result, **post:** post-competition result)

- Using **only a single model** (no ensembling), and MCG+RPN for region proposals, **31.2% mAP** is obtained for MS COCO competition submission result.
- And **average precision (AP) and average recall (AR) for small objects are greatly improved** from 4.1% to 7.4% and from 7.4% to 11.7% respectively compared with Fast R-CNN (F-RCN).
- With left-right flipping and adjusting training parameters, **33.1% mAP** is obtained for **post-competition result**.

. . .

Hope everyone enjoys reading my stories, happy deep learning, and wish you all Happy New Year!

. . .

References

[2016 CVPR] [ION]
Inside-Outside Net: Detecting Objects in Context with Skip Pooling and Recurrent Neural Networks

My Related Reviews

Image Classification

[LeNet] [AlexNet] [ZFNet] [VGGNet] [SPPNet] [PReLU-Net] [DeepImage] [GoogLeNet / Inception-v1] [BN-Inception / Inception-v2] [Inception-v3] [Inception-v4] [Xception] [MobileNetV1] [ResNet] [Pre-Activation ResNet] [RiR] [RoR] [Stochastic Depth] [WRN] [FractalNet] [Trimps-Soushen] [PolyNet] [ResNeXt] [DenseNet]

Object Detection

[OverFeat] [R-CNN] [Fast R-CNN] [Faster R-CNN] [DeepID-Net] [R-FCN] [SSD] [DSSD] [YOLOv1] [YOLOv2 / YOLO9000]

Semantic Segmentation

[FCN] [DeconvNet] [DeepLabv1 & DeepLabv2] [ParseNet] [DilatedNet] [PSPNet]