ION

INSIDE-OUTSIDE NET: DETECTING OBJECTS IN CONTEXT WITH SKIP POOLING AND RECURRENT NEURAL NETWORKS

SEAN BELL (CORNELL UNIVERSITY)
KAVITA BALA (CORNELL UNIVERSITY)
LARRY ZITNICK (MICROSOFT RESEARCH, NOW AT FAIR)
ROSS GIRSHICK (MICROSOFT RESEARCH, NOW AT FAIR)

Cornell University

Microsoft Research

ION TEAM




Sean Bell **Kavita Bala**

(Cornell University)

Larry Zitnick **Ross Girshick**

(Microsoft Research,
now both at FAIR)

SUMMARY: MS COCO DETECTION

Best Student Entry
(3rd Place Overall)

	test-competition	test-dev	Runtime
Competition	31.0%	31.2%	2.7 s
Post-Competition		33.1%	5.5 s

(single ConvNet model, no ensembling)

Key pieces:

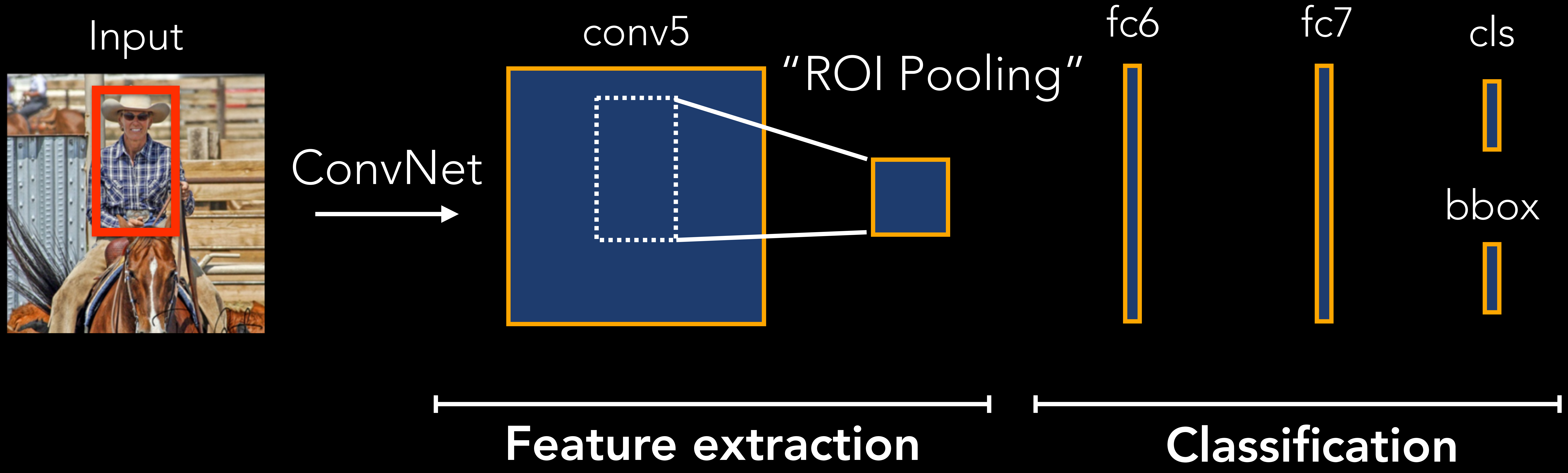
- New ION detector (+5.1 mAP)
- Better proposals, more data (+3.9 mAP)
- Better training/testing (+4.1 mAP)

Tech report: http://arxiv.org/pdf/1512.04143.pdf

ION DETECTOR

+5.1 mAP on COCO test-dev
compared to Fast R-CNN

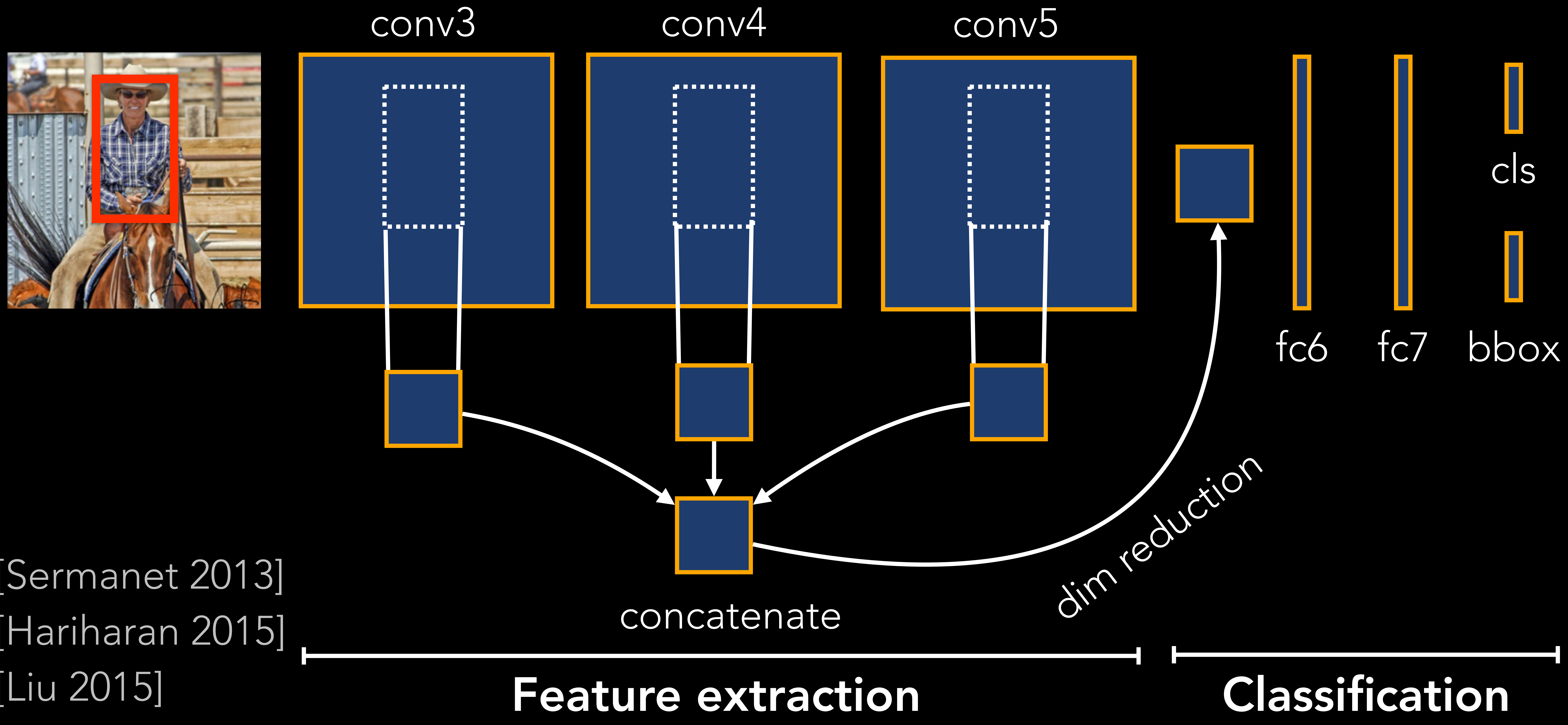
FAST R-CNN [GIRSHICK 2015]



Can we improve on feature extraction?

- For small objects, the footprint on conv5 might only cover a 1x1 cell, which gets upsampled to 7x7
- Only local features (inside the ROI) are used for classification

LET'S ADD SKIP CONNECTIONS



[Sermanet 2013]
[Hariharan 2015]
[Liu 2015]

PROBLEM: FEATURE AMPLITUDE

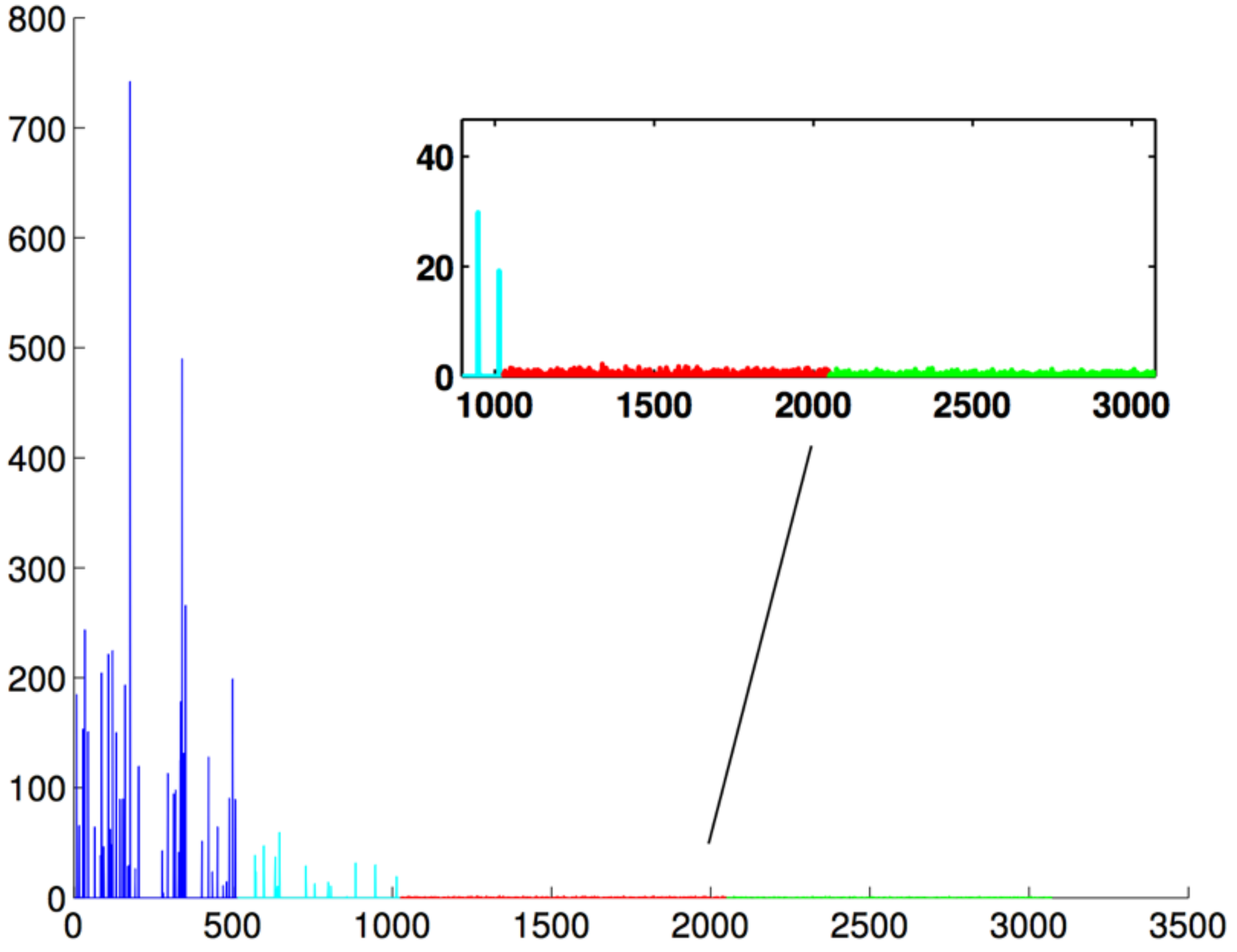
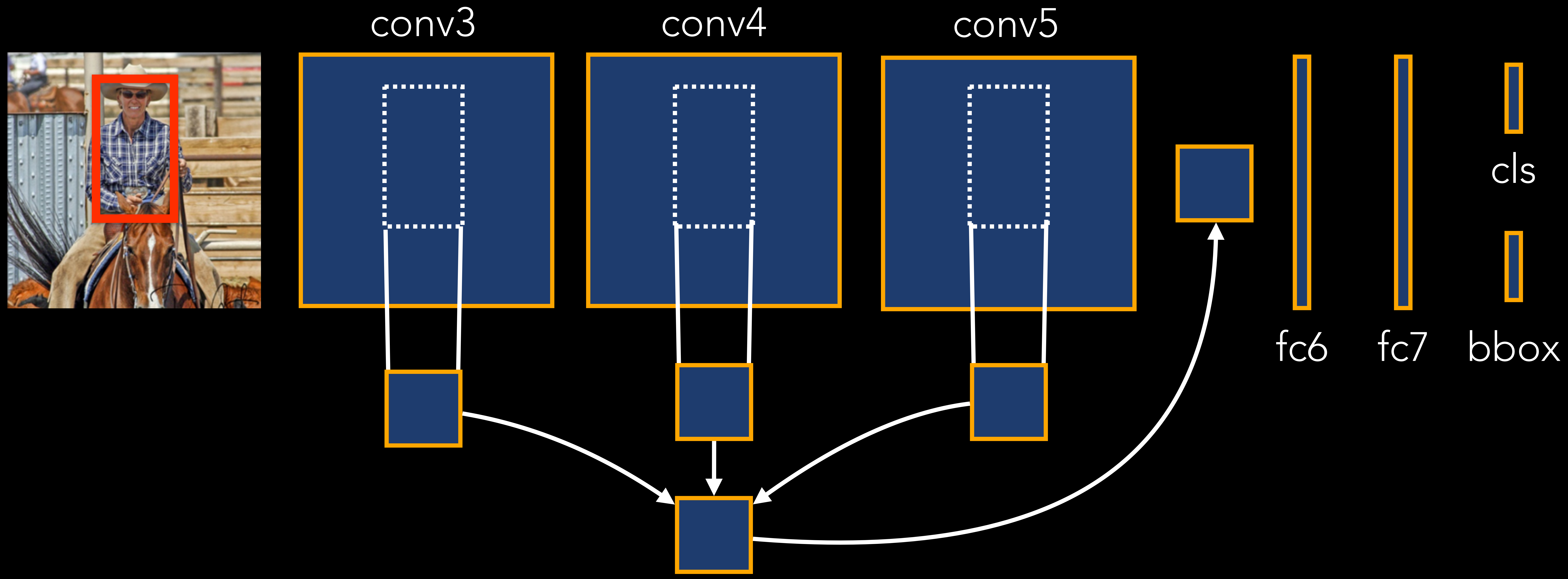


Figure 2: Features are in different scale. We show the features for a position from conv4_3, conv5_3, fc7 and pool6 when we concatenate them together.

- Different layers have very different amplitudes
- We must account for this to combine features
- L2 normalize to length 1, and then re-scale

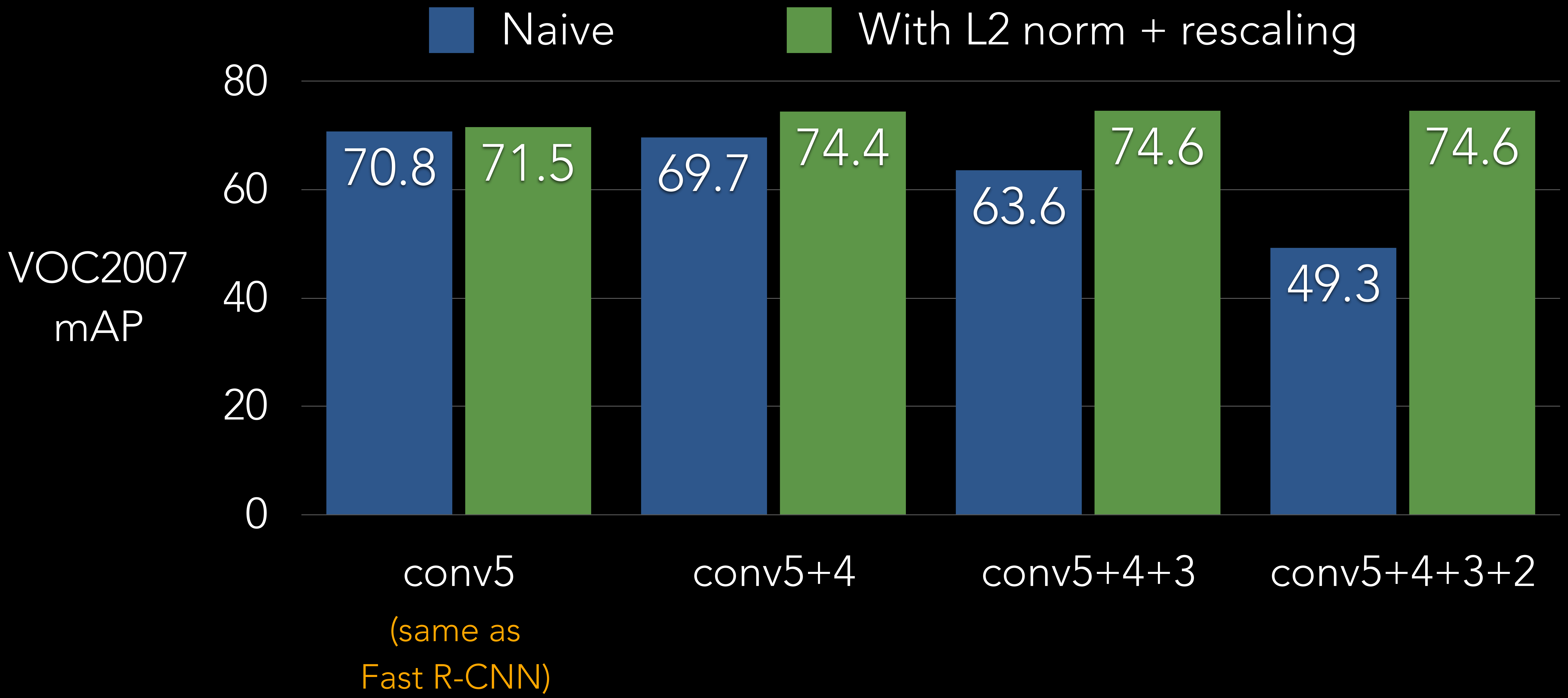
[Liu 2015]

COMBINING ACROSS LAYERS

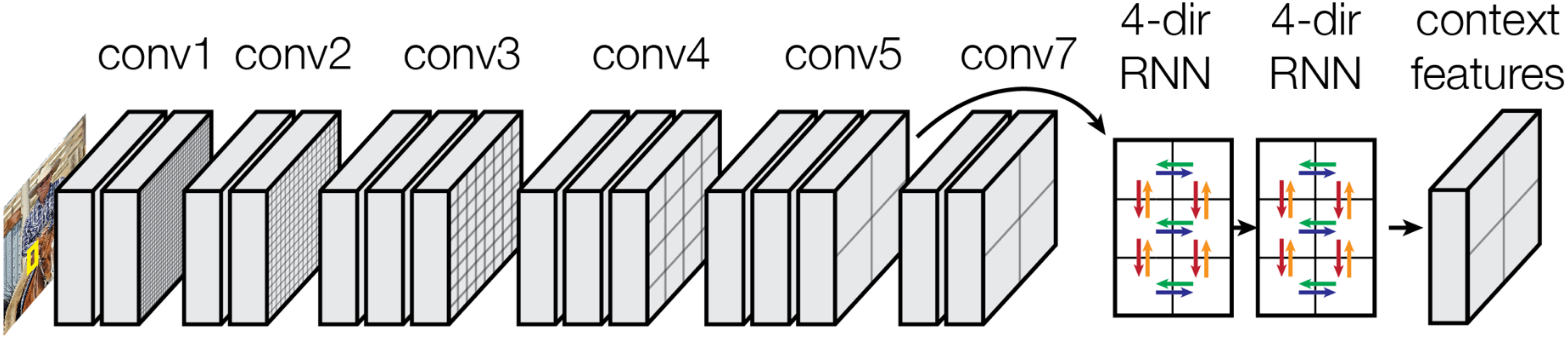


normalize, concatenate, **re-scale**

RESCALING FEATURE AMPLITUDES

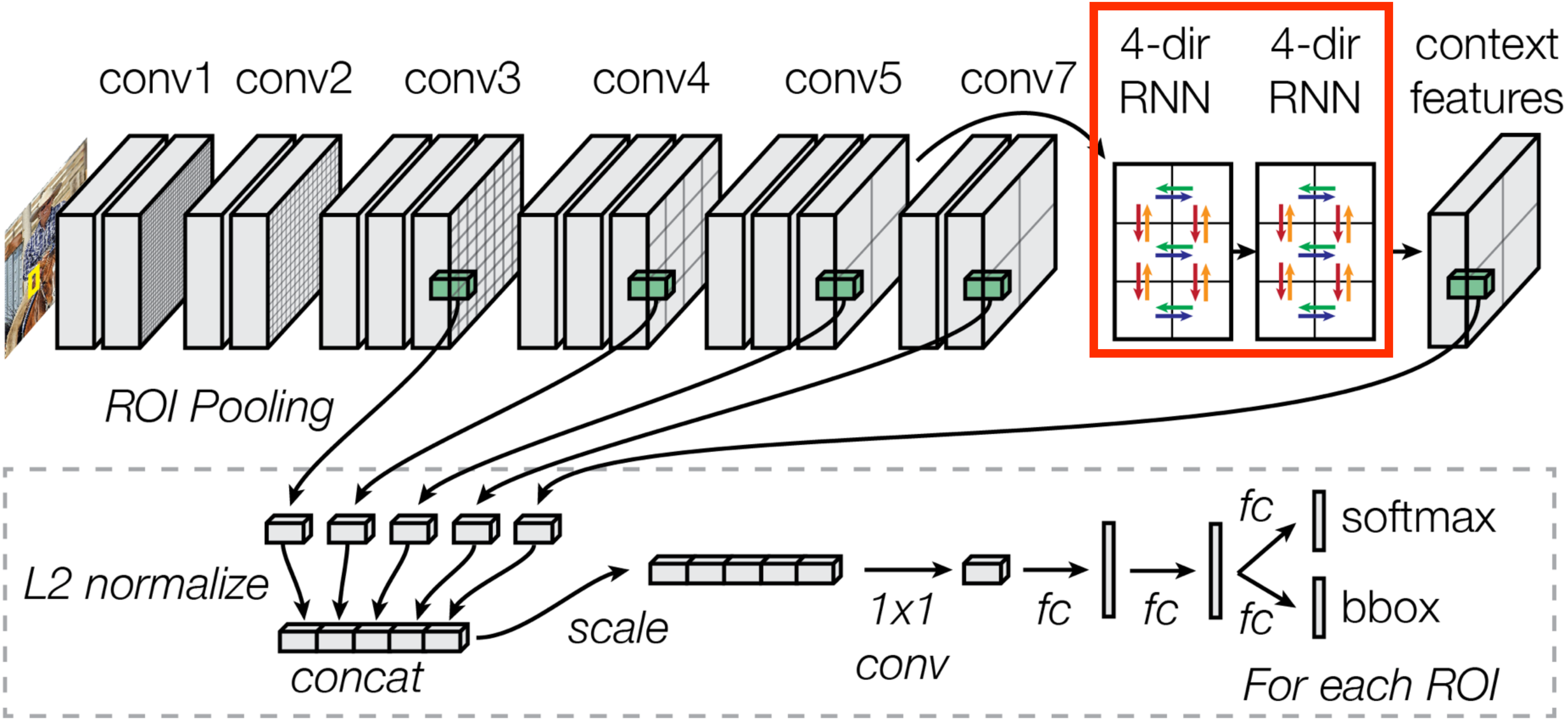


ION: INSIDE-OUTSIDE NET



Base ConvNet: VGG16 [Simonyan 2014]

ION: INSIDE-OUTSIDE NET



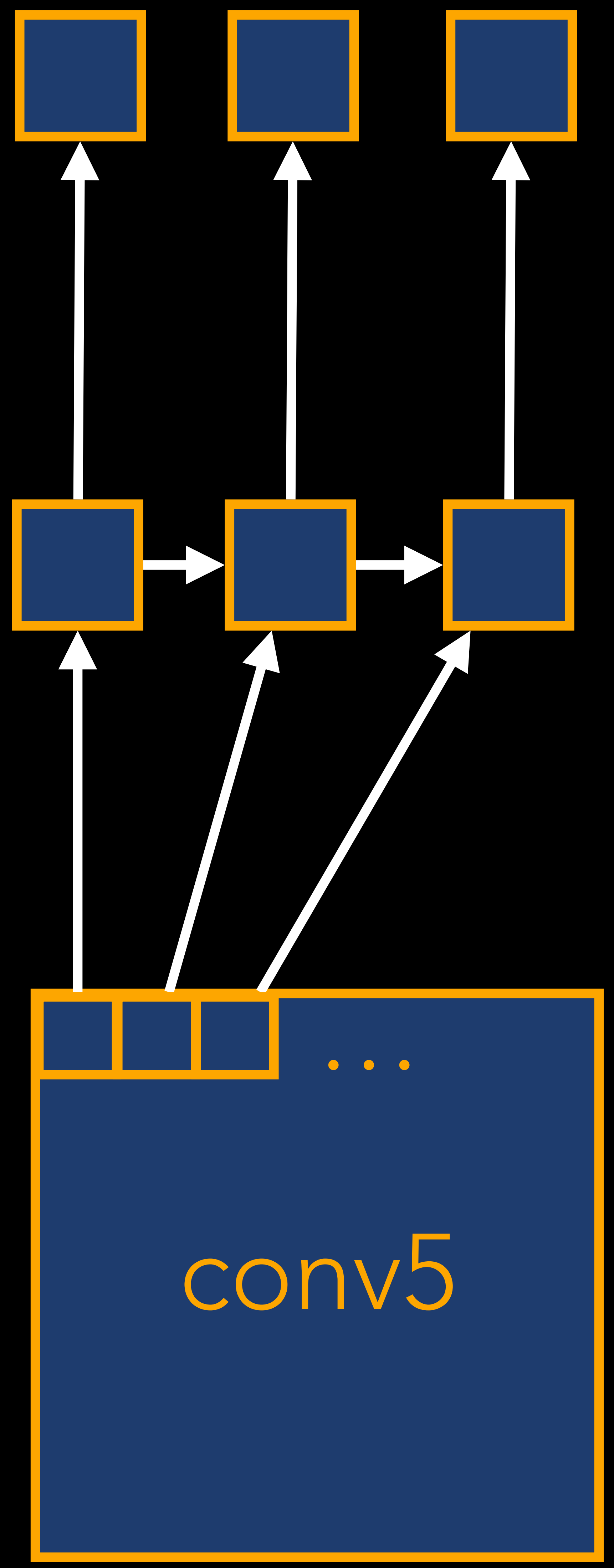
Base ConvNet: VGG16 [Simonyan 2014]

LATERAL RNN (MOVES ACROSS AN IMAGE)

Output
(which we interpret as
context features)

Hidden state

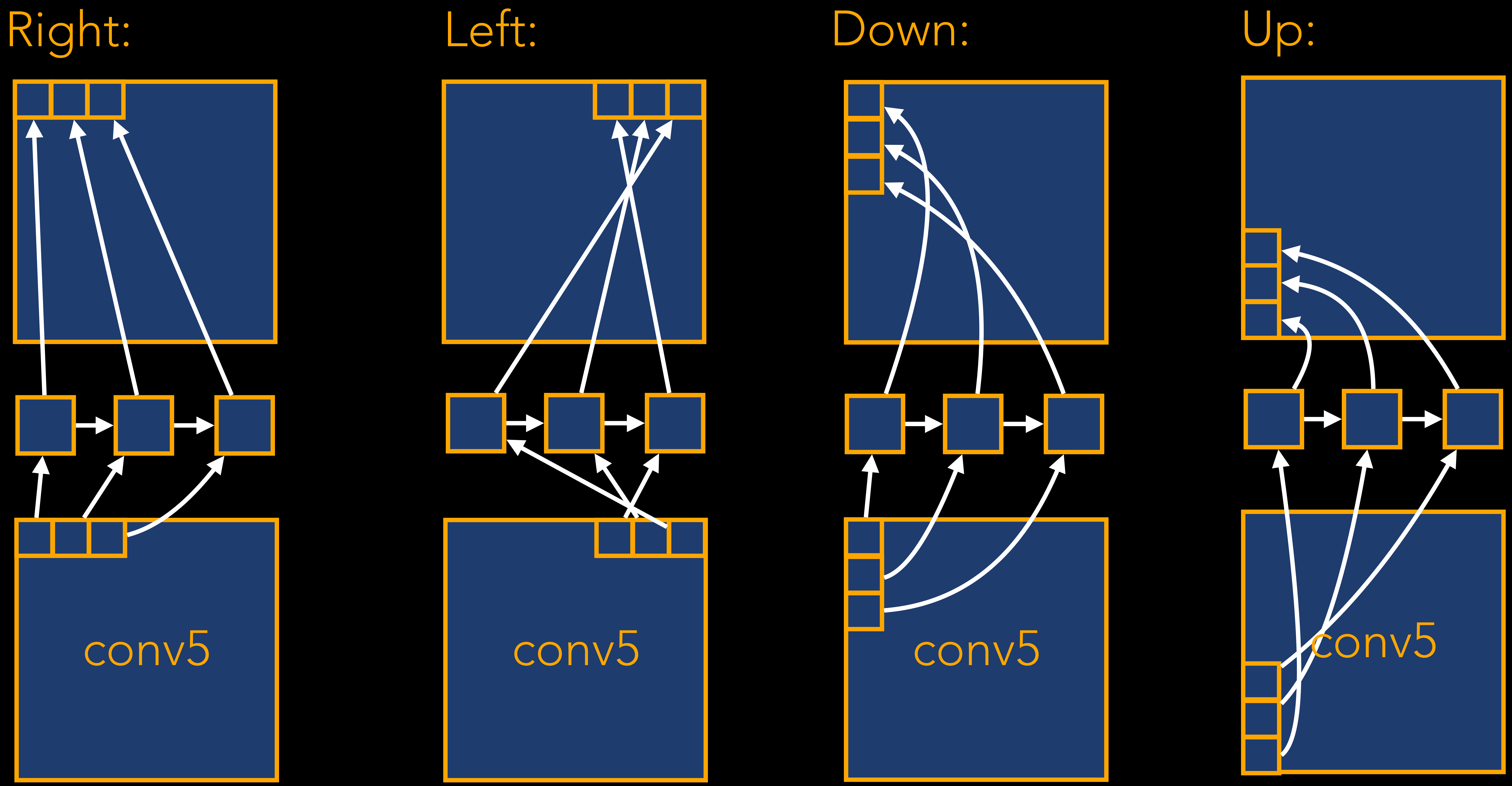
Convolutional
features



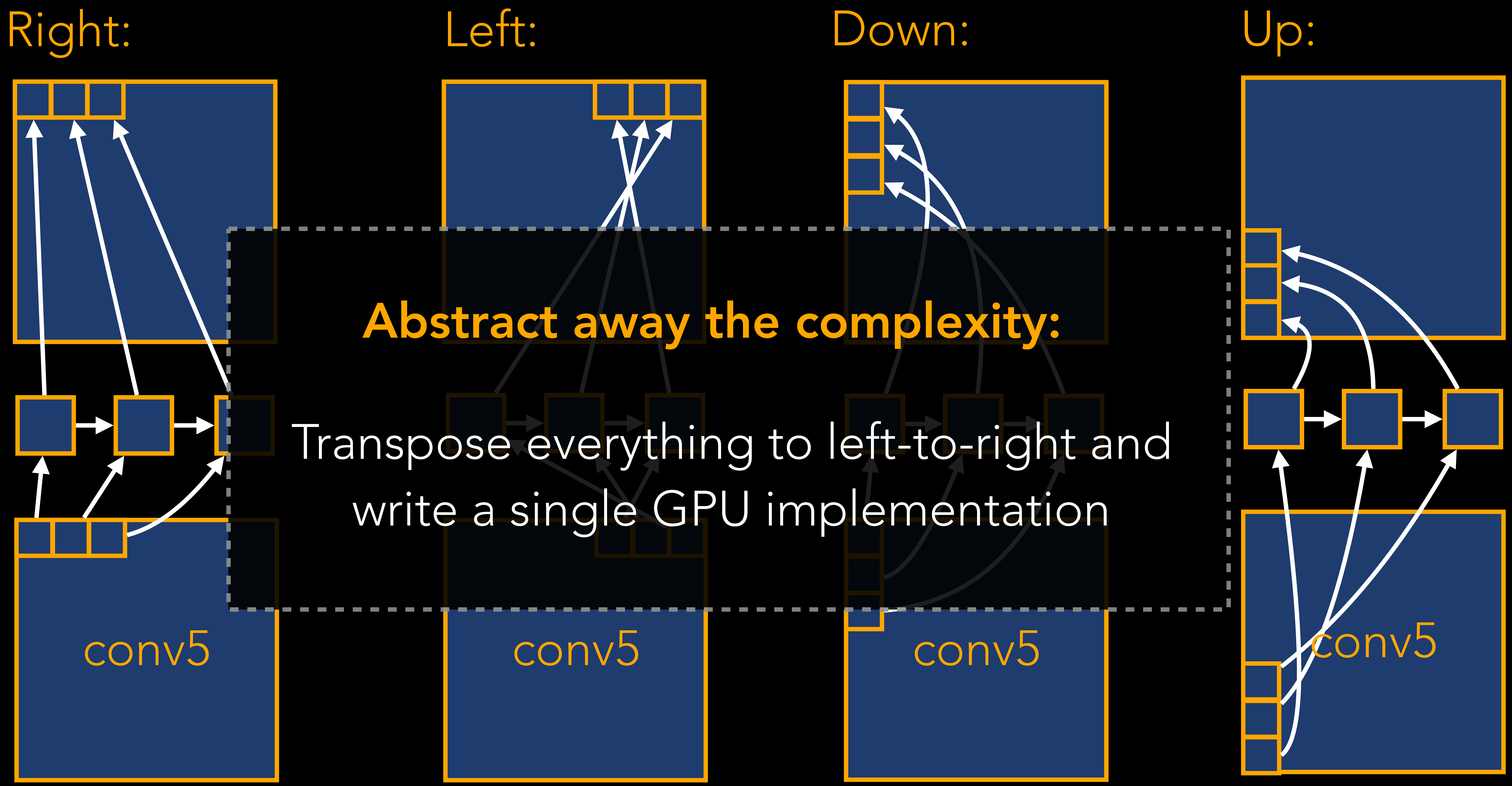
- Repeat for each row
- Can compute each column in parallel
- We can also move in 4 different directions

[Schuster 1997], [Graves 2009], [Byeon 2015], [Visin 2015]

RNN IMPLEMENTATION



RNN IMPLEMENTATION

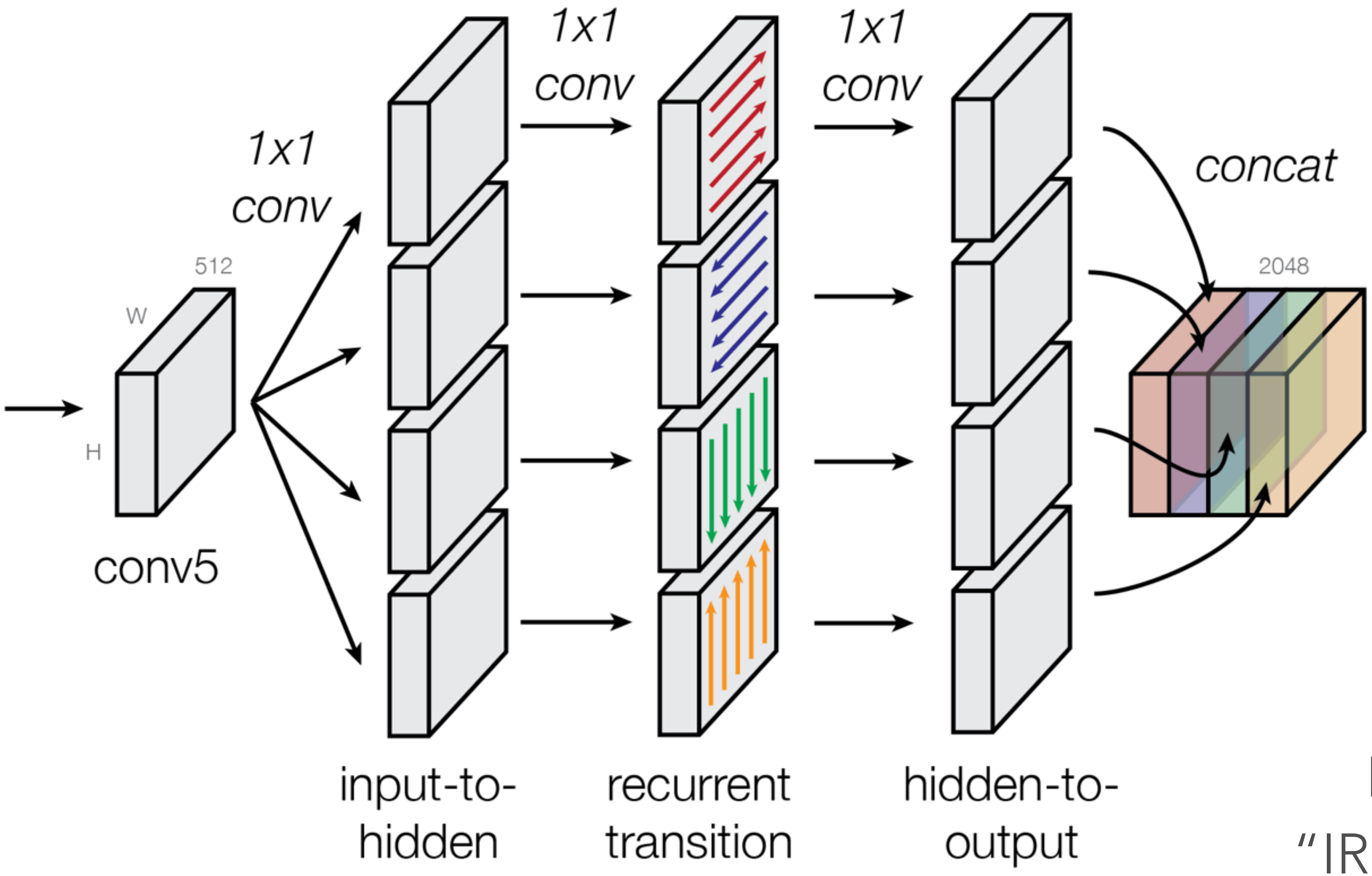


Abstract away the complexity:

Transpose everything to left-to-right and write a single GPU implementation

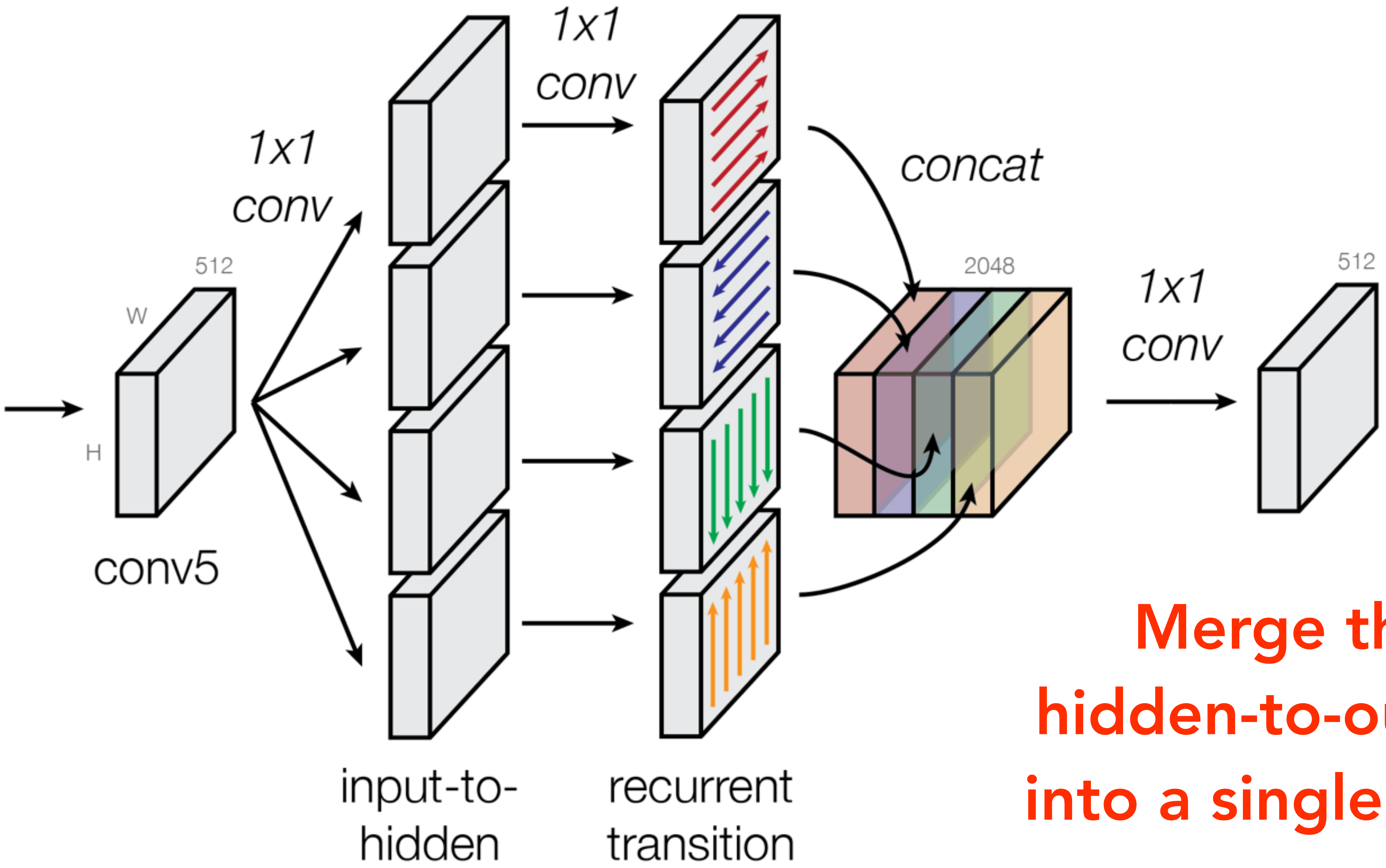
RNN IMPLEMENTATION

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



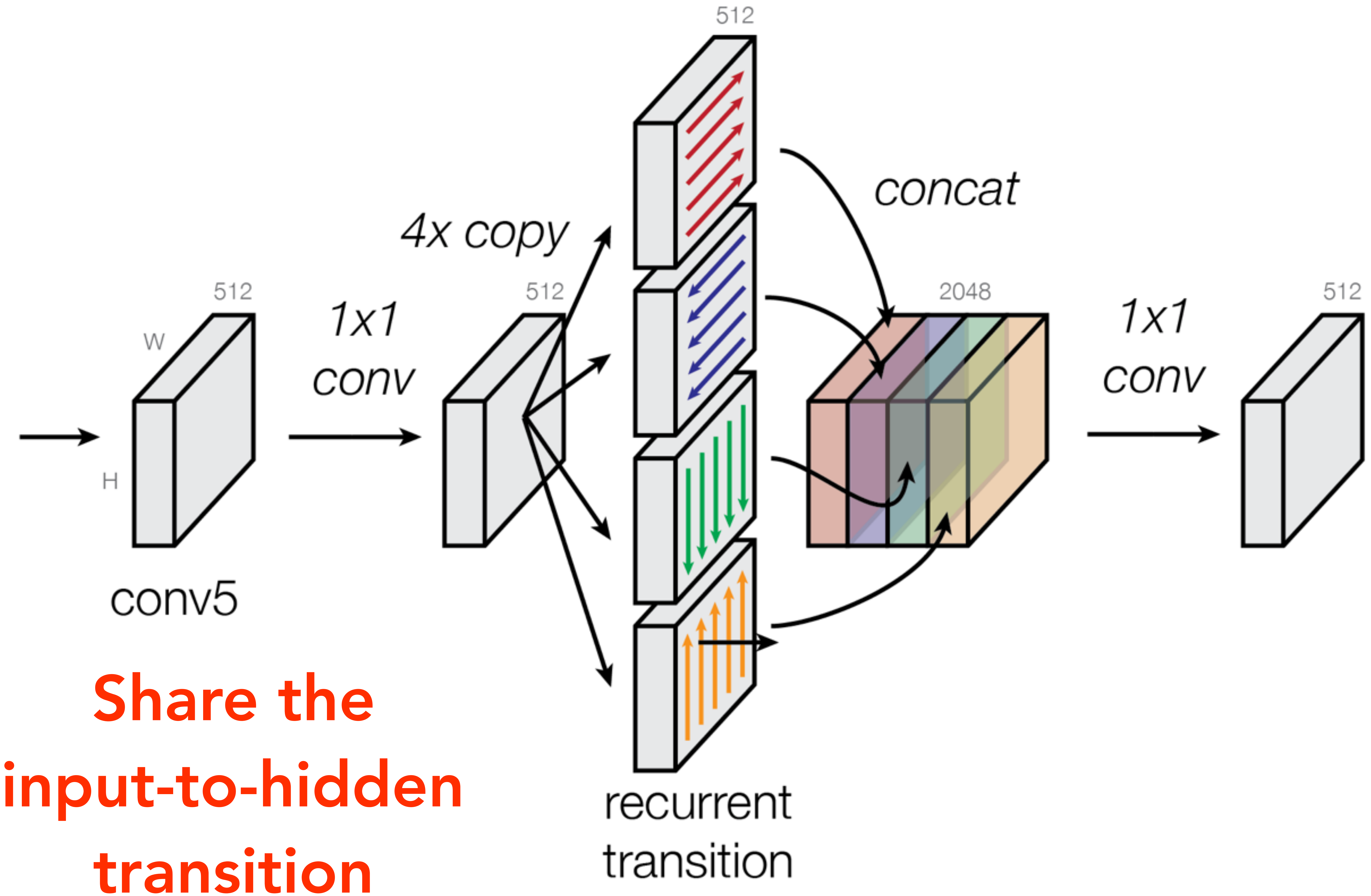
ReLU RNN:
"IRNN" [Le 2015]

RNN IMPLEMENTATION



Merge the hidden-to-output into a single conv.

RNN IMPLEMENTATION



Share the
input-to-hidden
transition

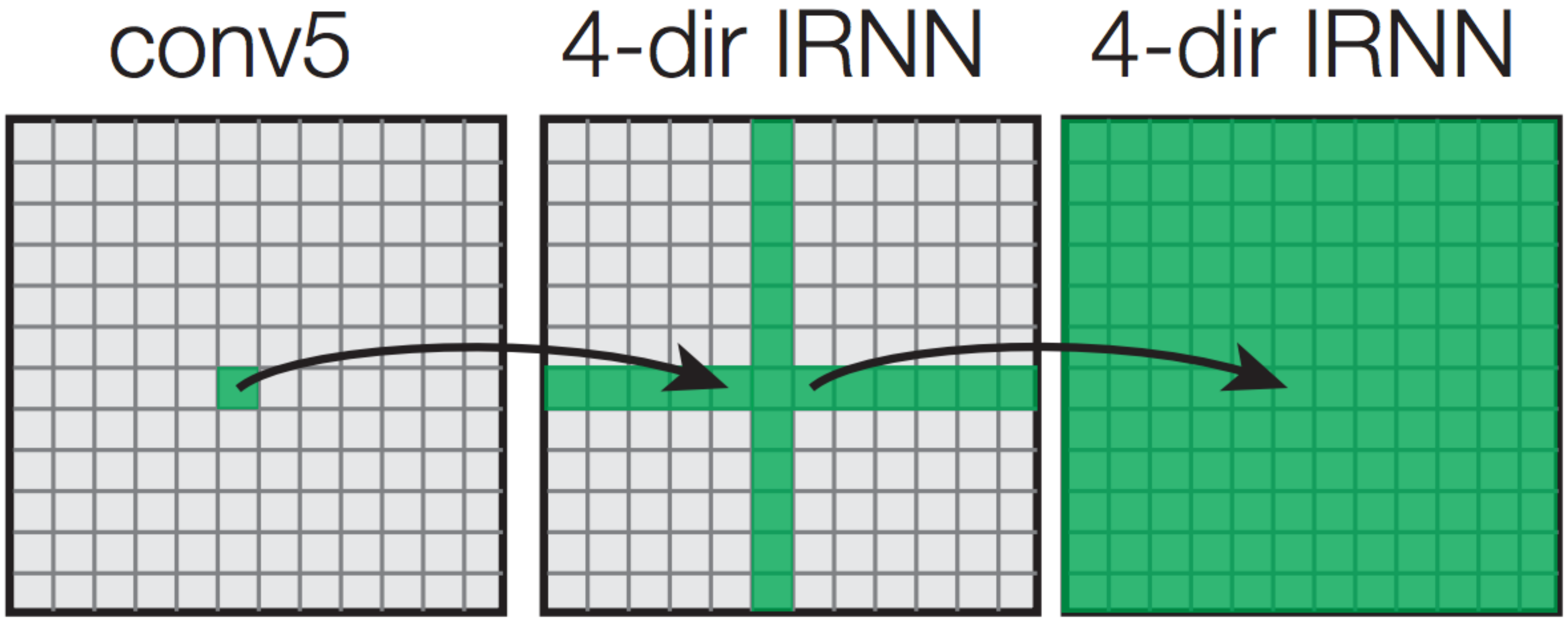
RNN IMPLEMENTATION

Our final architecture:

Features used by our detector

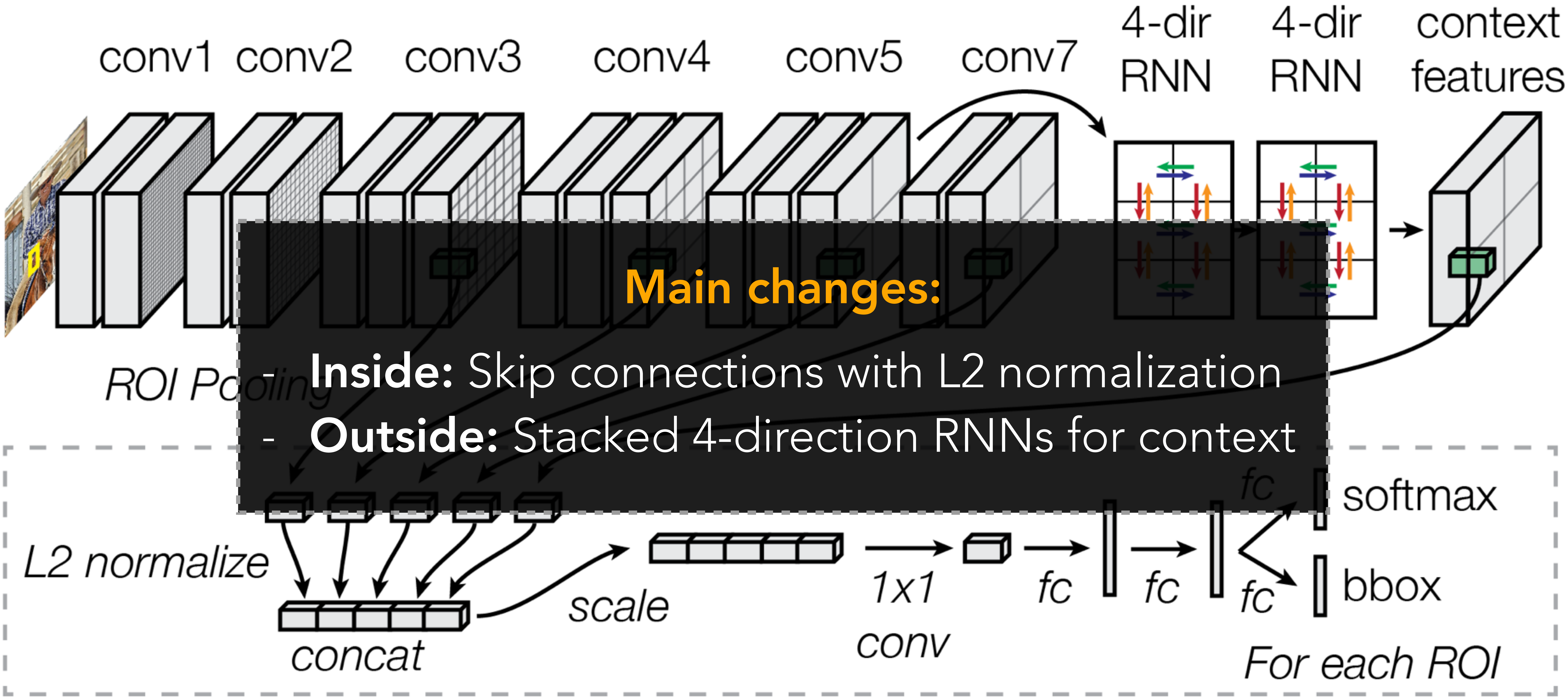
Stack 2 RNNs together

RNN: SPATIAL DEPENDENCY



(d) two 4-direction IRNN layers

ION: INSIDE-OUTSIDE NET



Main changes:

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

Base ConvNet: VGG16 [Simonyan 2014]

BETTER PROPOSALS, MORE DATA

+3.9 mAP on COCO test-dev,
compared to Selective Search

REGION PROPOSAL NETWORK (RPN)

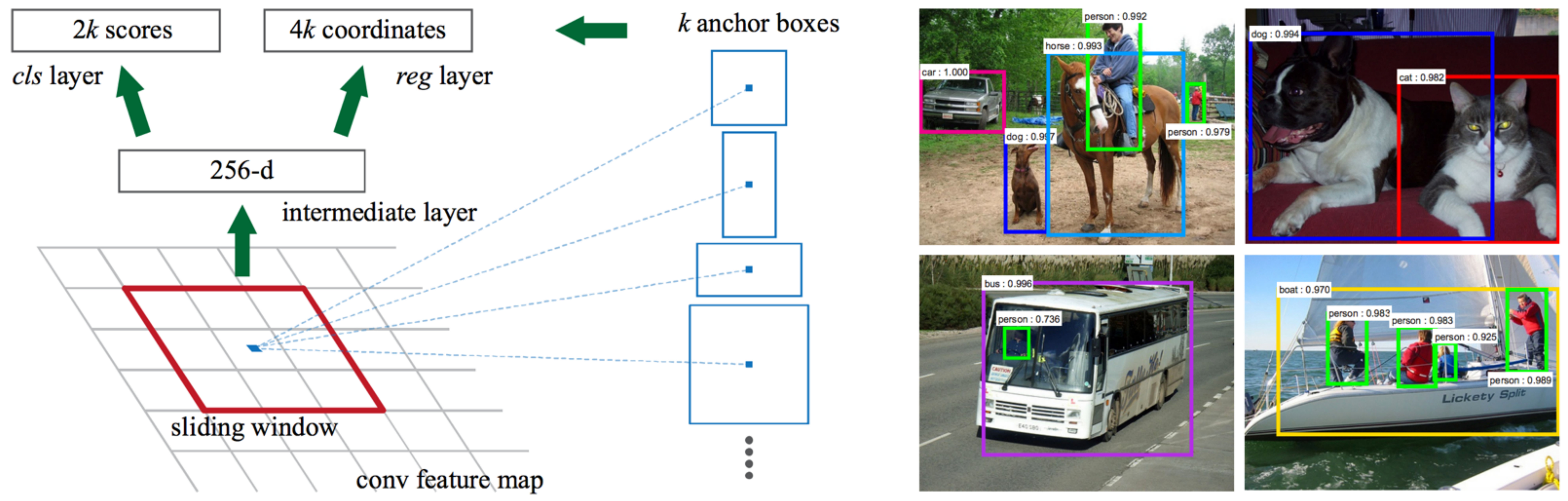


Figure 1: **Left**: Region Proposal Network (RPN). **Right**: Example detections using RPN proposals on PASCAL VOC 2007 test. Our method detects objects in a wide range of scales and aspect ratios.

Faster R-CNN [Ren 2015]

REGION PROPOSAL NETWORK (RPN)

- Original RPN [Ren 2015] used **9 anchors**: 3 scales x 3 aspect ratios. RPN works well for VOC, but not COCO

- We extend this to **22 anchors:** 7 scales x 3 aspect ratios, and 32x32

	Avg. Recall
Selective Search [Uijlings 2013]	41.7%
MCG [Arbelaez 2014]	51.6%
RPN with 10 anchors [Ren 2015]	39.9%
RPN with 22 anchors	**44.1%**

- We *mix* MCG with RPN, which performs better than either individually (1000 of each for training, 2000 of each for testing)

BETTER TRAINING/TESTING

+4.1 mAP on COCO test-dev,
compared to Fast R-CNN setup

TRAINING IMPROVEMENTS

- No dropout (+0.6 mAP)
- Train for longer with larger mini-batches
  4 images (512 ROIs total) / batch (+0.8 mAP)
- Regularize with semantic segmentation predictions (+1.3 mAP)
  (see tech report)

(mAP on test-dev)

TESTING IMPROVEMENTS

- We use **iterative box regression and weighted voting**, from MR-CNN [Gidaris 2015]
    - Helps on PASCAL (+2.0 mAP)
    - *Reduces* score on COCO (-0.5 mAP), since COCO requires precise localization
    - New thresholds: NMS: ~0.45, voting: ~0.85 (+1.3 mAP)
- **Left-right flips:** evaluate on original and flipped image and average (+0.8 mAP)


(a) Step 1
(b) Step 2
(c) Step 3
(d) Step 4
(e) Step 5

[Gidaris 2015]

COMPARISON TO RESNET (WINNER) [HE 2015]

training data	COCO train		COCO trainval	
test data	COCO val		COCO test-dev	
mAP	@.5	@[.5, .95]	@.5	@[.5, .95]
baseline Faster R-CNN (VGG-16)	41.5	21.2		
baseline Faster R-CNN (ResNet-101)	48.4	27.2		
+box refinement	49.9	29.9		
+context	51.1	30.0	53.3	32.2
+multi-scale testing	53.8	32.5	**55.7**	**34.9**
ensemble			**59.0**	**37.4**

Table 9. Object detection improvements on MS COCO using Faster R-CNN and ResNet-101.

Combining ResNet101 and ION is potentially complimentary

Our single-model (post-competition) result: **33.1% mAP**

CONCLUSION

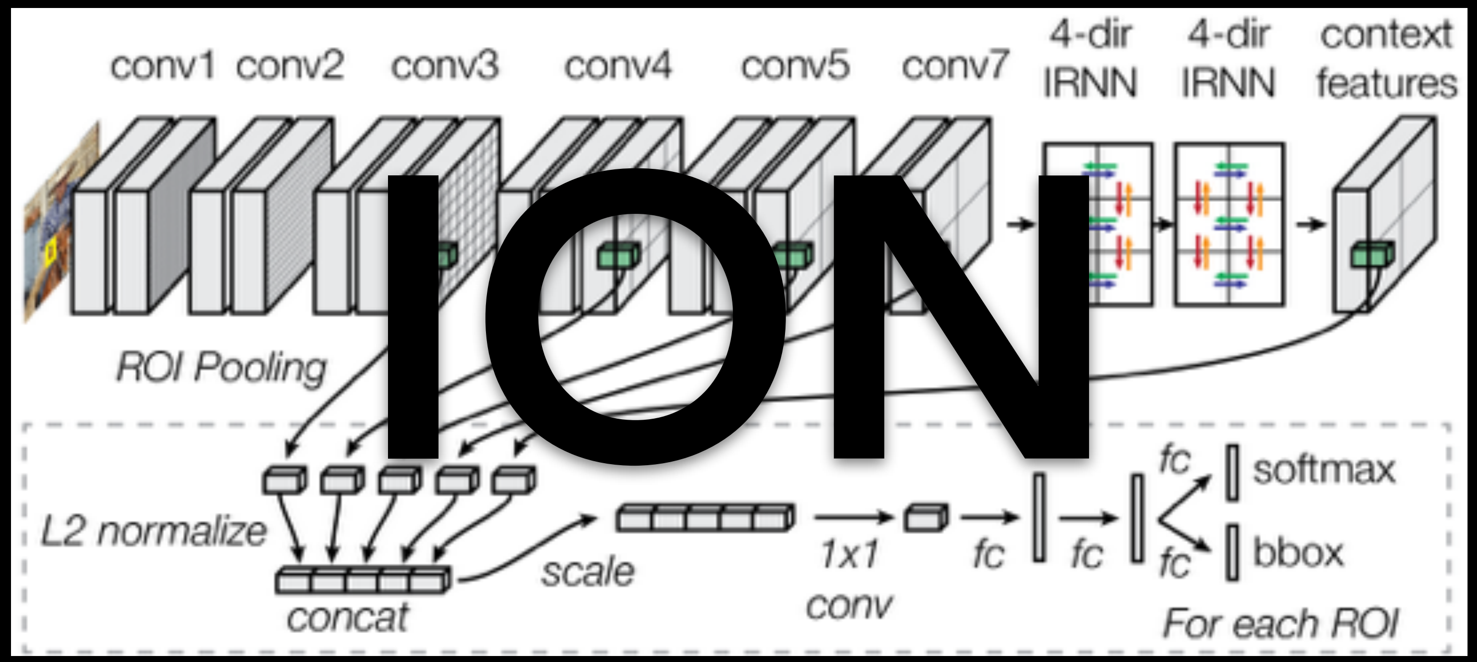
Improvement breakdown:

- New ION detector (+5.1 mAP)
- Better proposals, more data (+3.9 mAP)
- Better training/testing (+4.1 mAP)

Thanks:

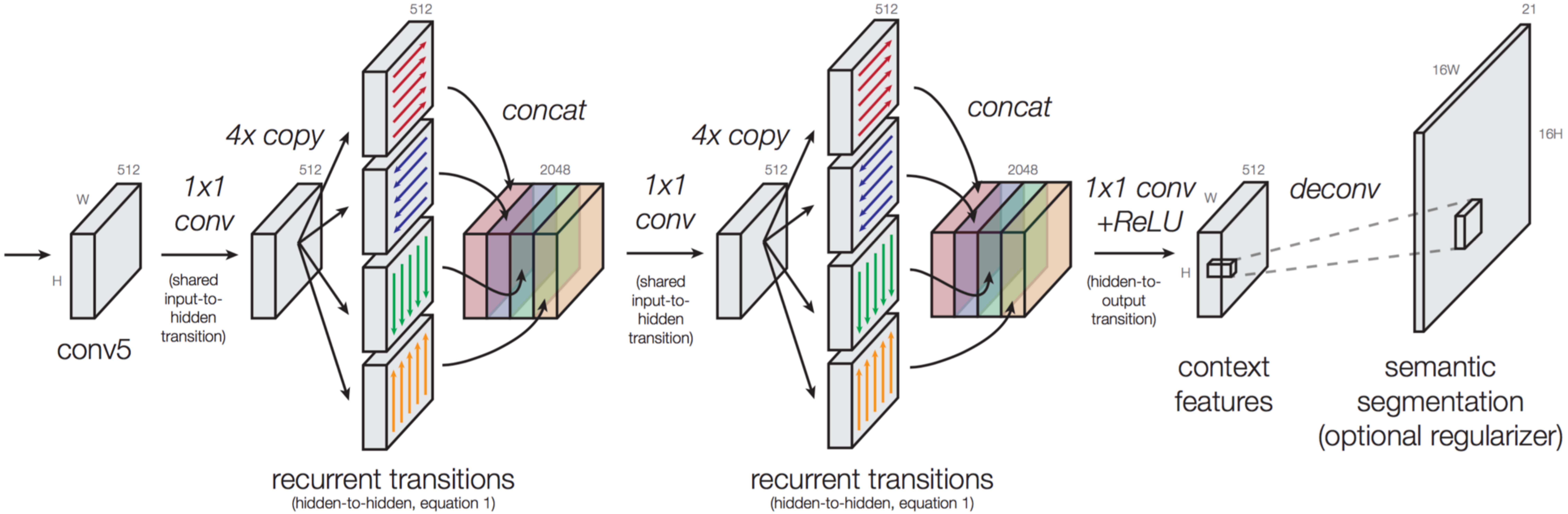
- NVIDIA (GPU Donation)
- Microsoft Research (Internship)

Tech Report: http://arxiv.org/pdf/1512.04143.pdf



EXTRA SLIDES

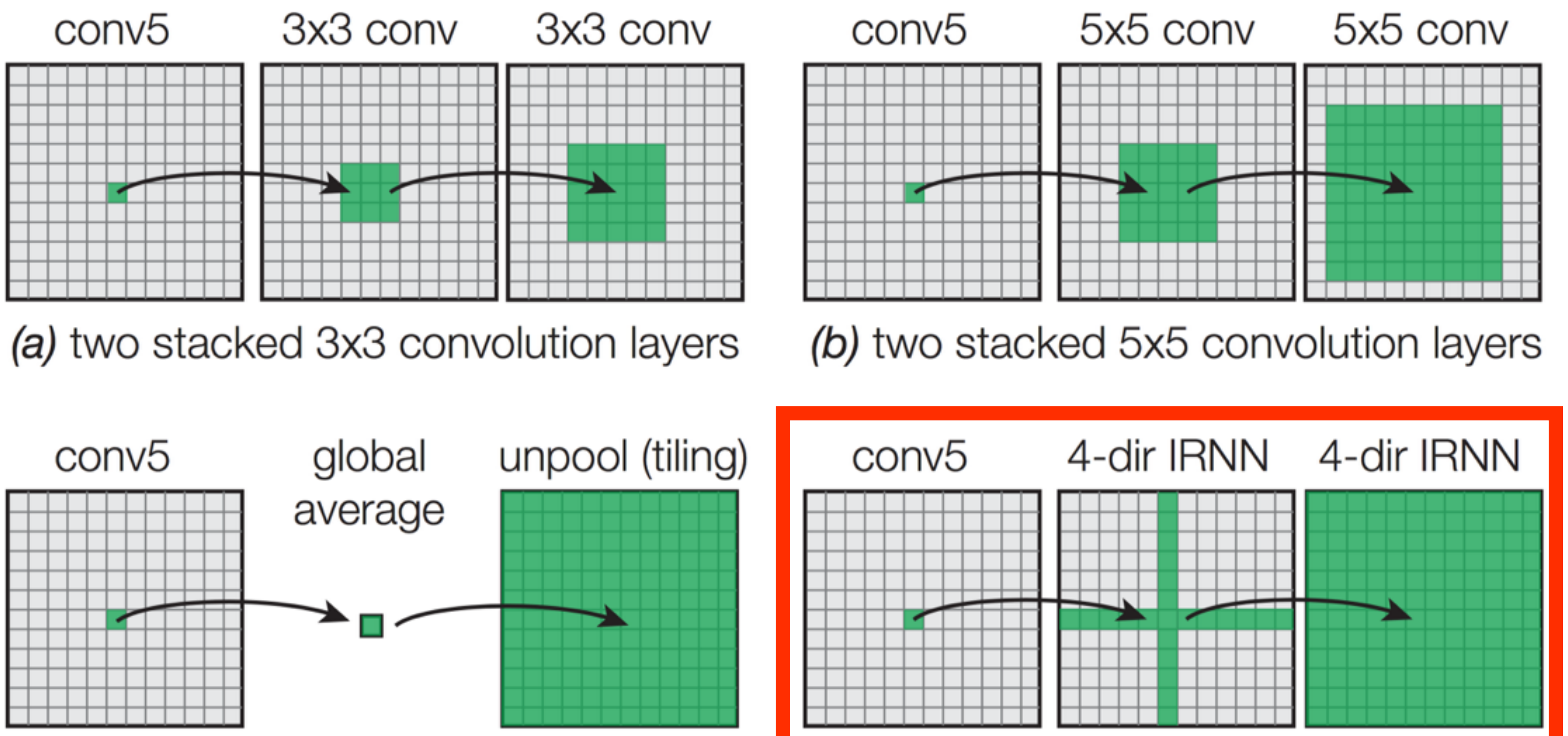
SURPRISING FIND: H2H TRANSITION NOT NEEDED



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

We use H2H for our submission, but there is barely any drop without it!

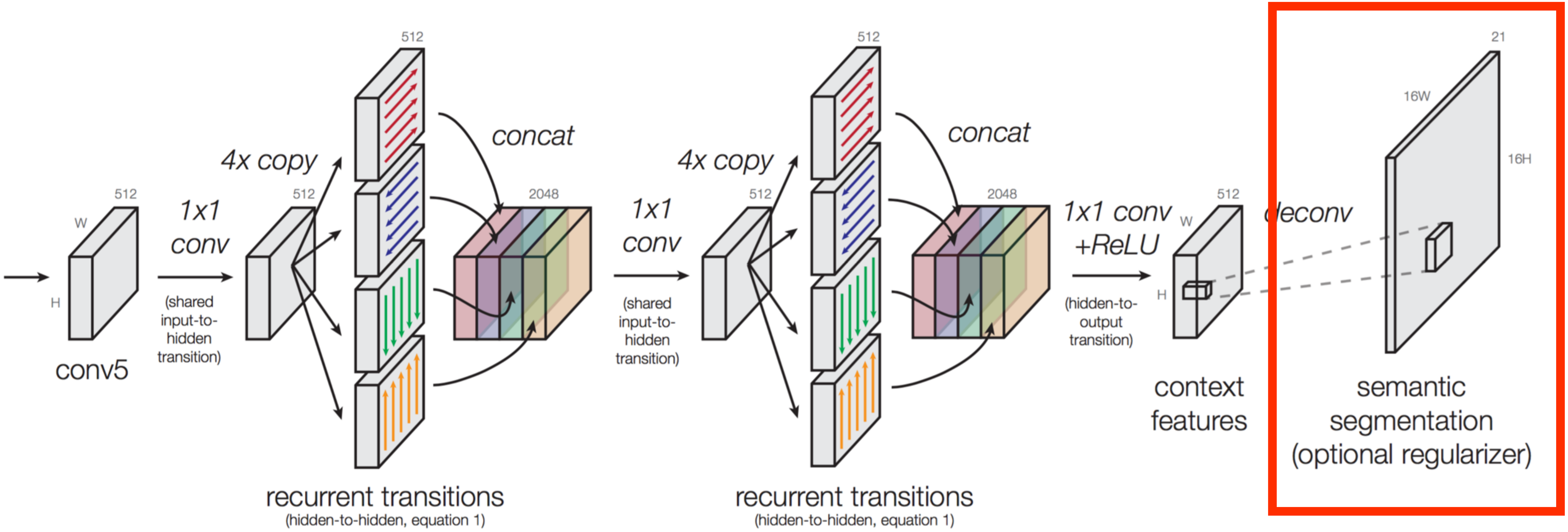
WHAT ABOUT OTHER CONTEXT METHODS?



(a) two stacked 3x3 convolution layers
(b) two stacked 5x5 convolution layers
(c) global averaging and unpooling
(d) two 4-direction IRNN layers

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**

IS SEGMENTATION LOSS WORTH IT?



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**

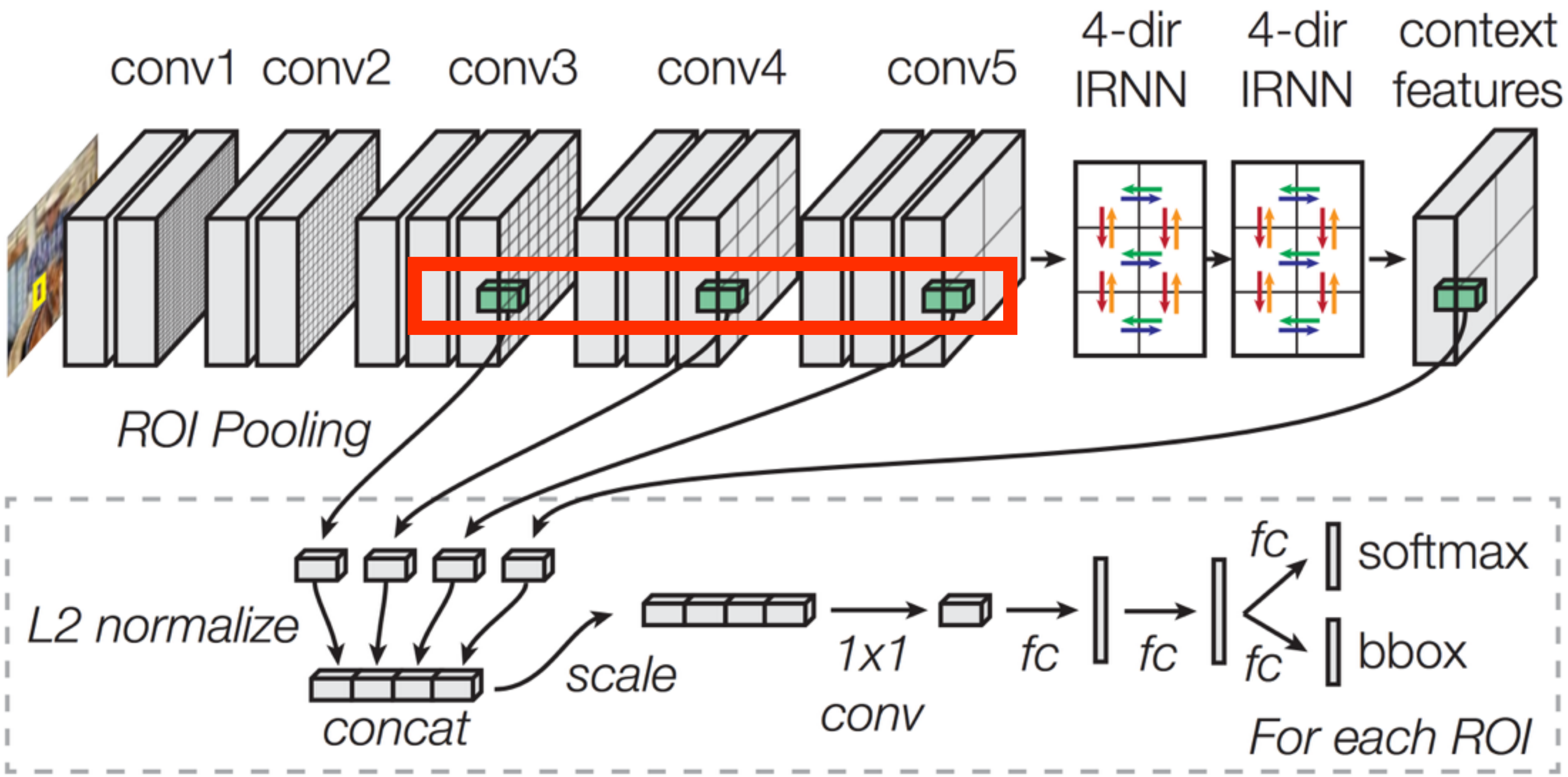
Test: +1mAP, same speed

Train: 1.5x-2x slower

HOW MANY RNN LAYERS?

ROI pooling from:					Seg.	# IRNN layers		
C2	C3	C4	C5	IRNN		1	2	3
				✓	✓		70.6	
			✓	✓	✓	74.3		
		✓	✓	✓	✓	75.8	76.2	
	✓	✓	✓	✓	✓	76.1	76.5	75.9
✓	✓	✓	✓	✓	✓		**76.8**	

WHY CONV3, CONV4, CONV5?



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

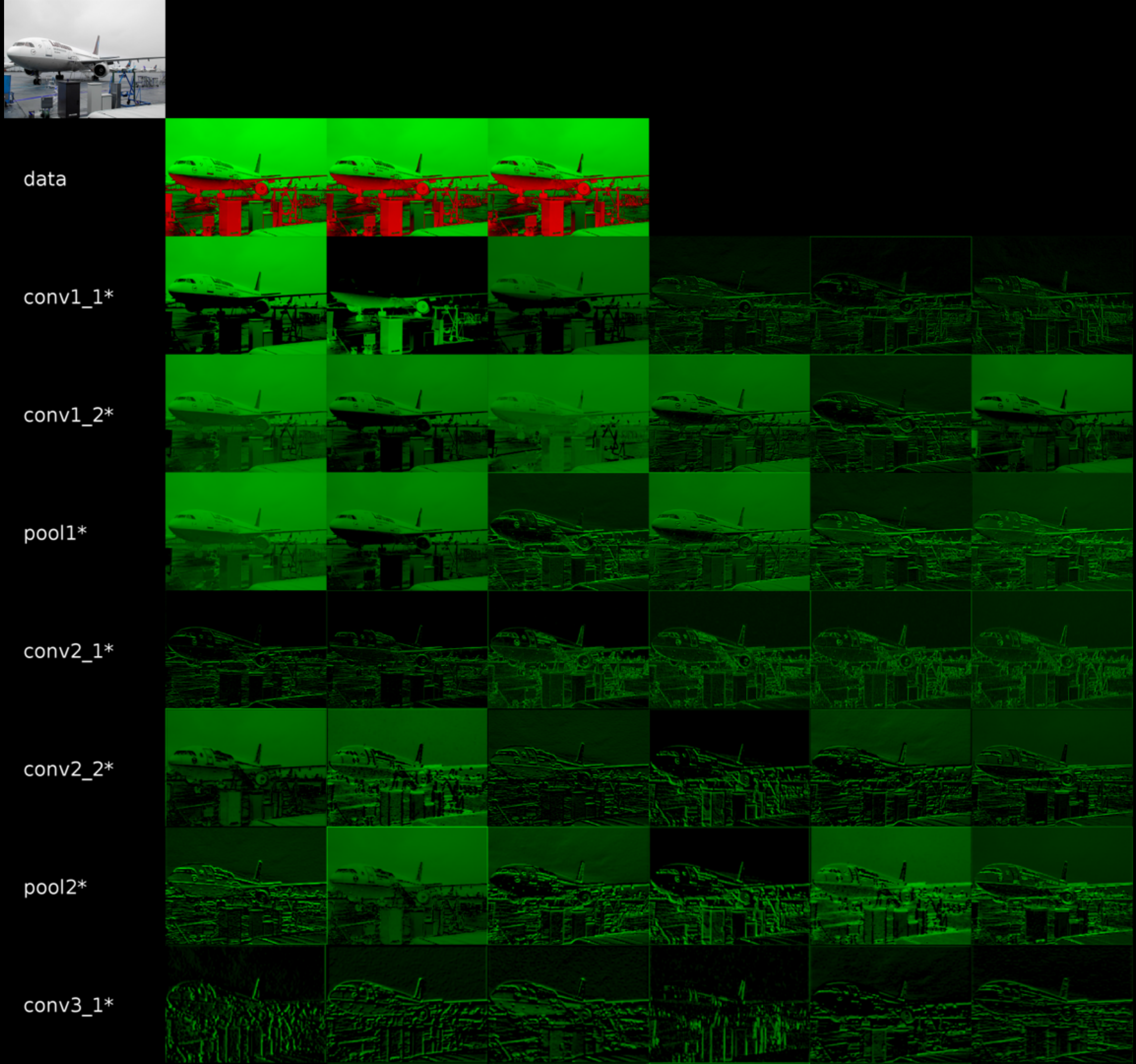
RESULTS (VOC 2007 TEST)

METHOD	MAP
FAST R-CNN [GIRSHICK 2014]	70.0
FASTER R-CNN [GIRSHICK 2015]	73.2
CONV3+CONV4+CONV5	**75.6**
+ RNN + SEGMENTATION LOSS	**76.5**
+ SECOND BBOX REGRESSION + WEIGHTED VOTING	**78.5**
— DROPOUT	**79.2**

ACTIVATIONS

Input

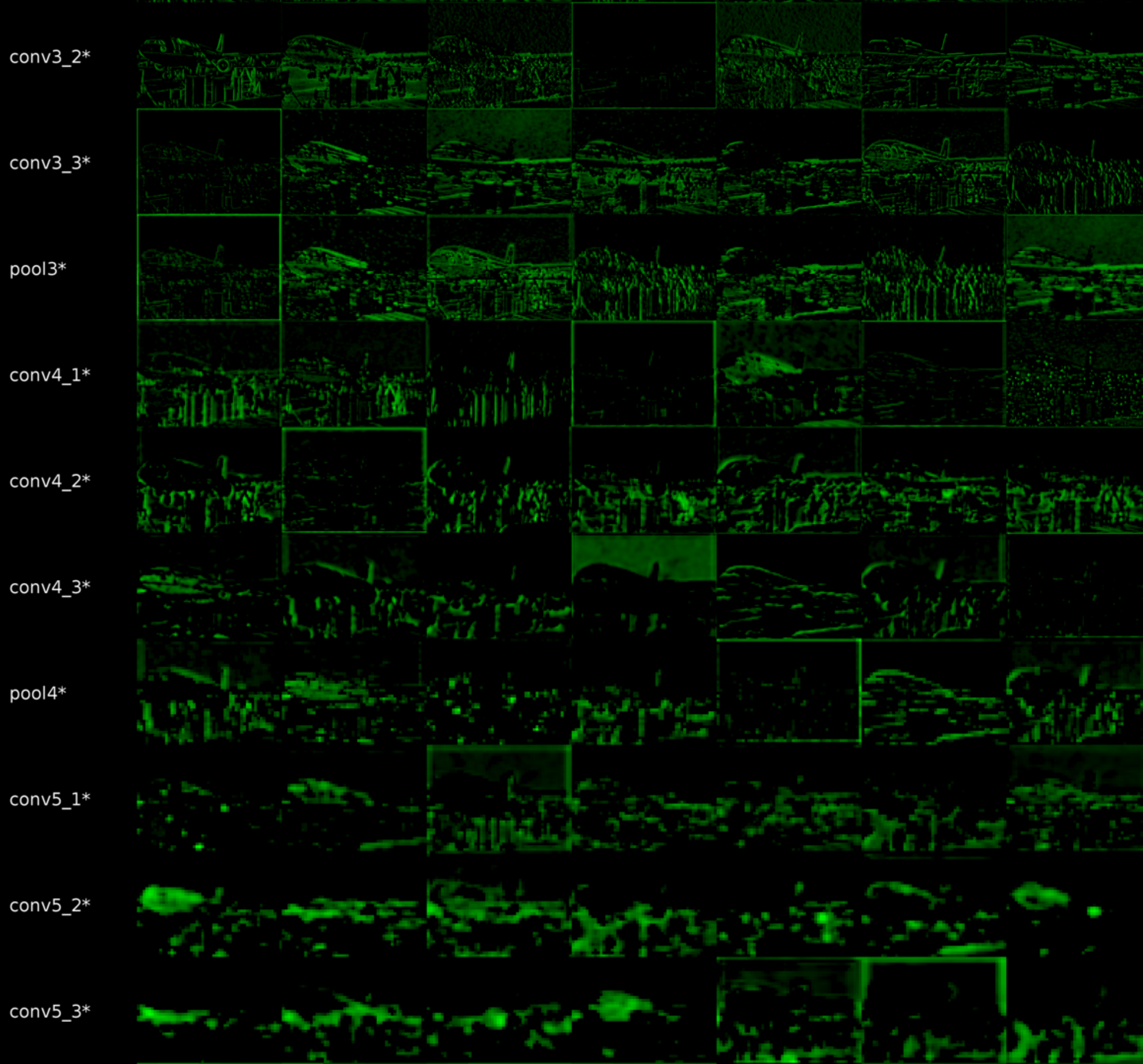
Positive Negative



ACTIVATIONS

Input

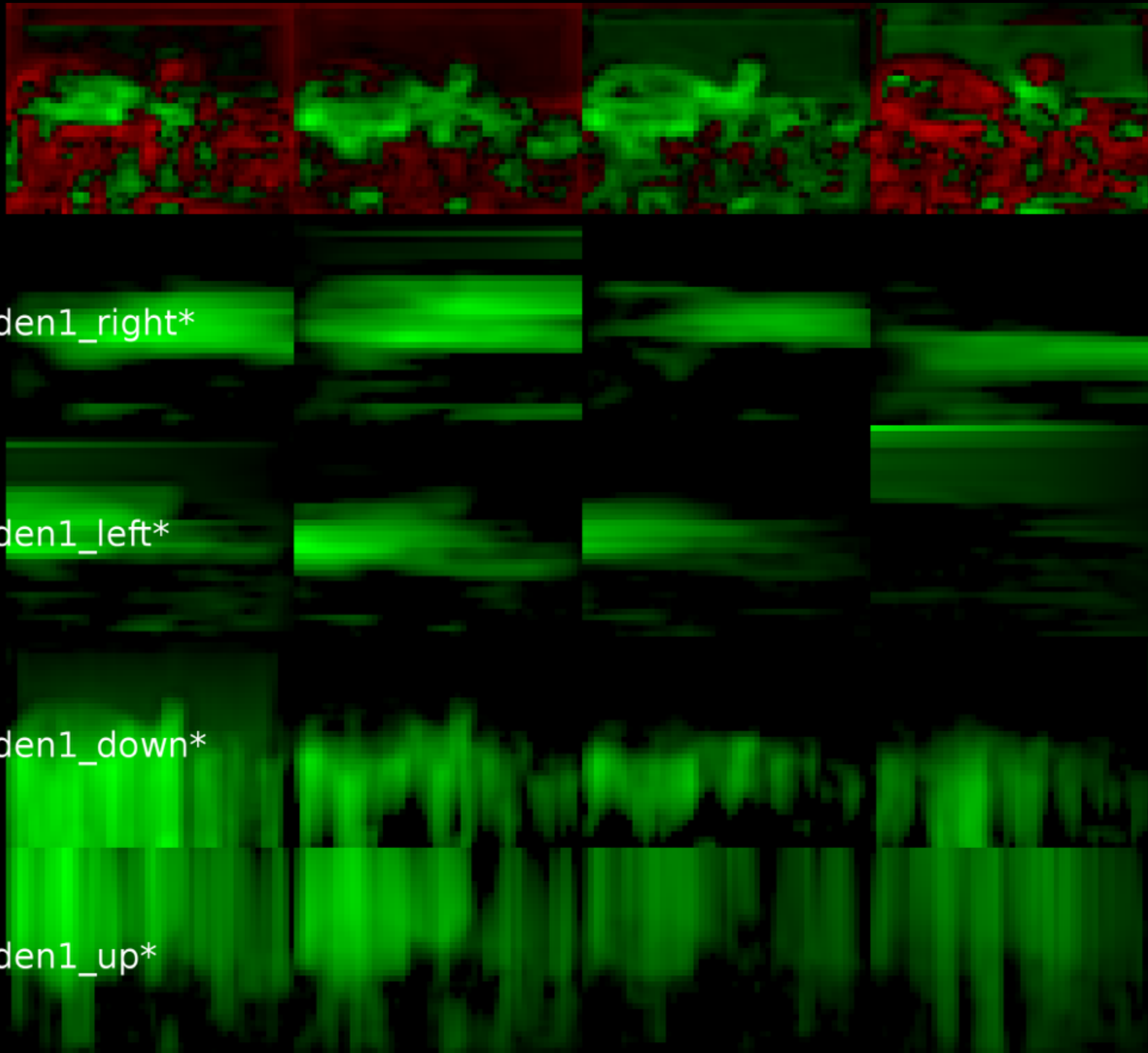
Positive Negative



RNN ACTIVATIONS

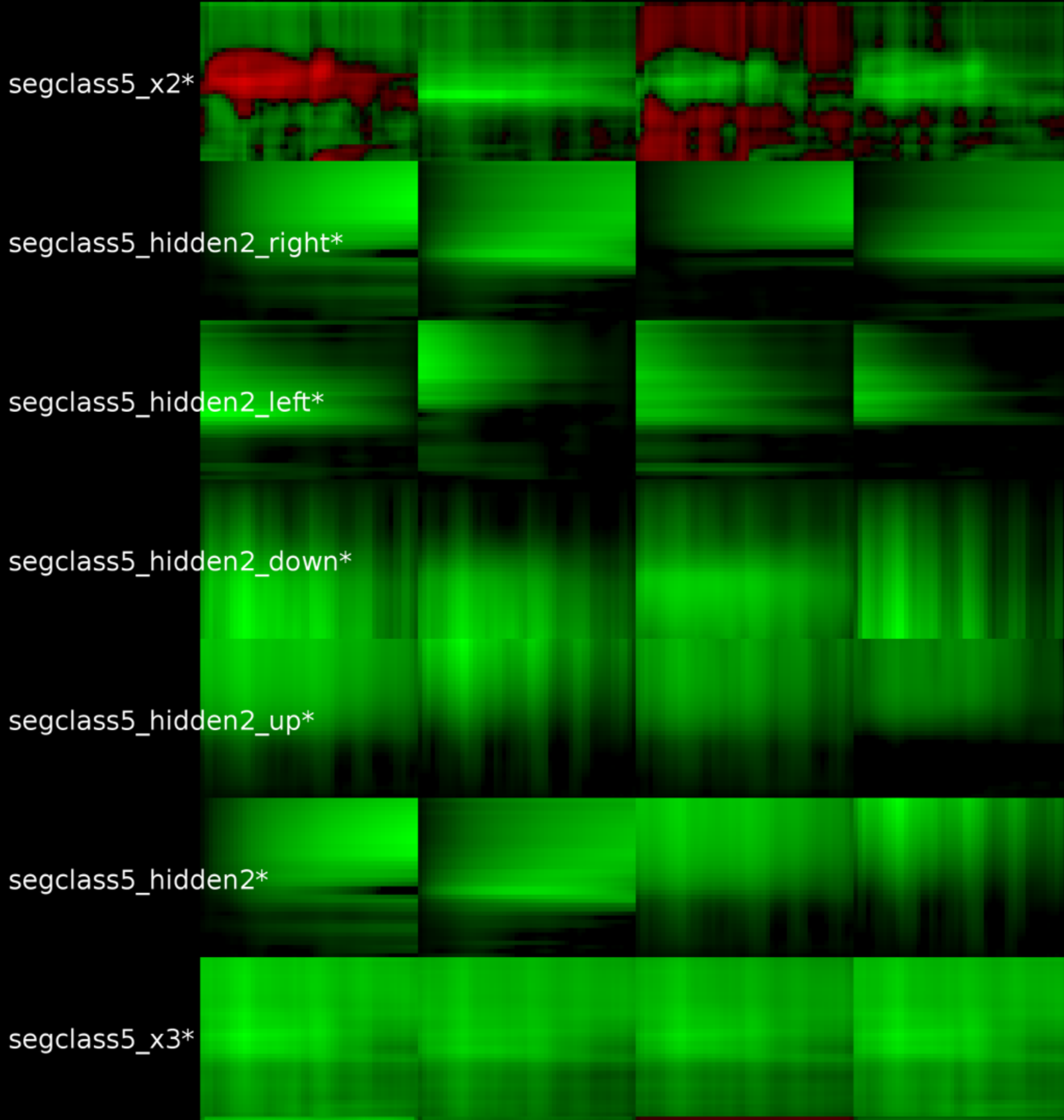
Input

Positive Negative



RNN ACTIVATIONS

Input

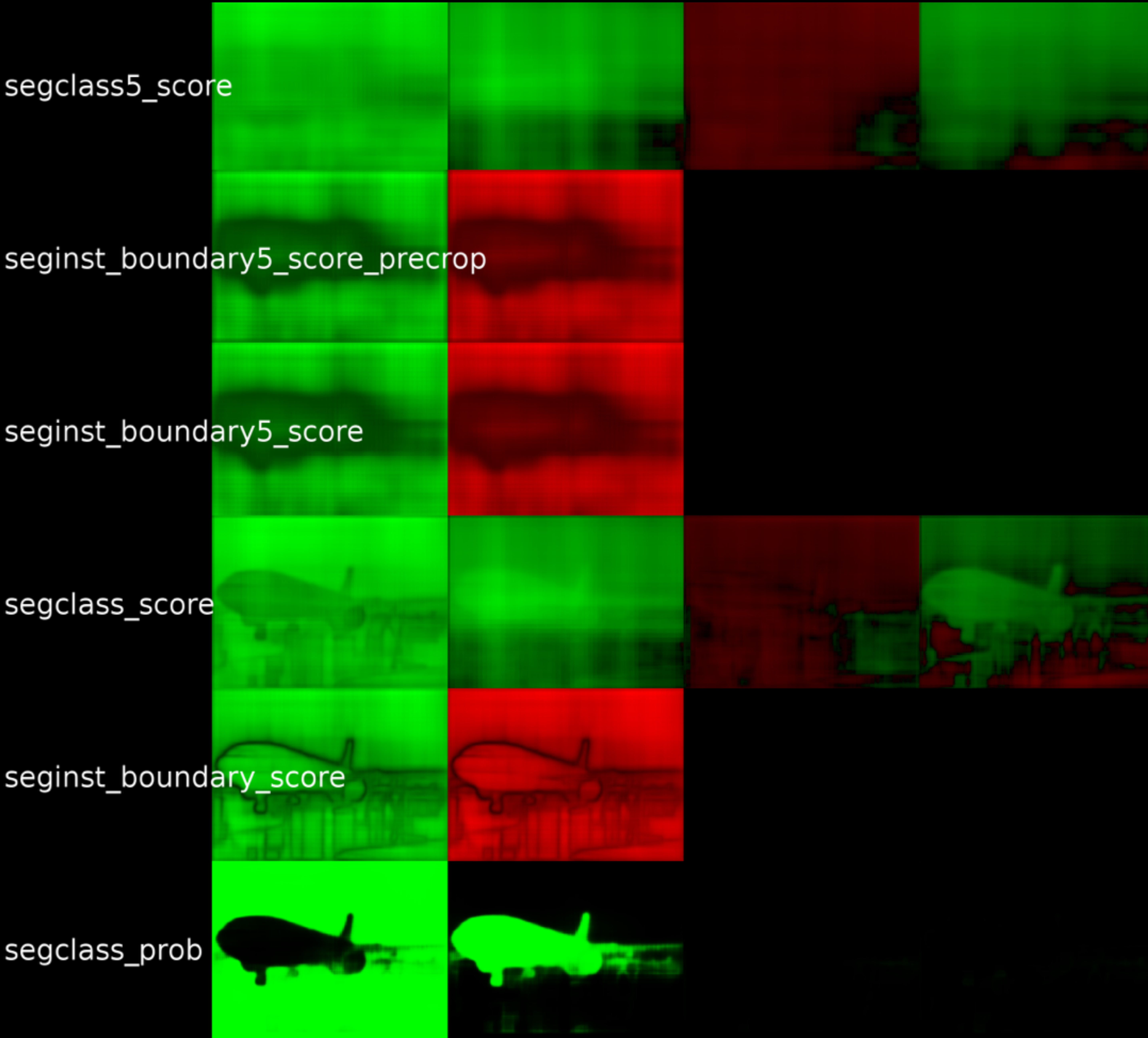


Positive Negative

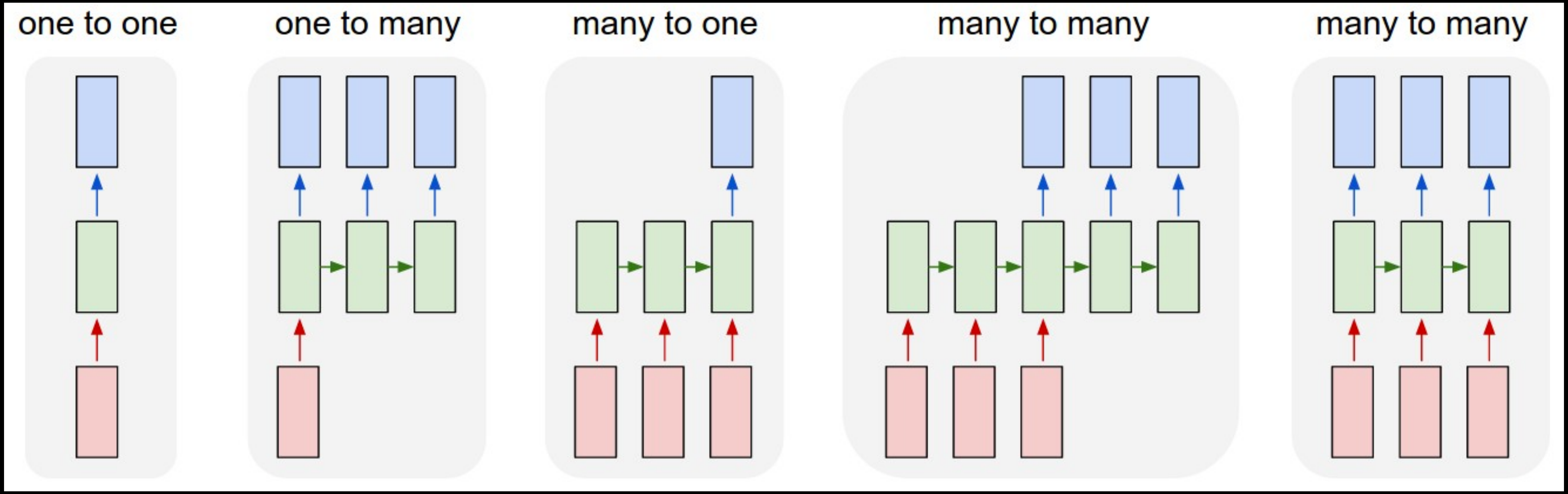
RNN ACTIVATIONS

Input

Positive Negative

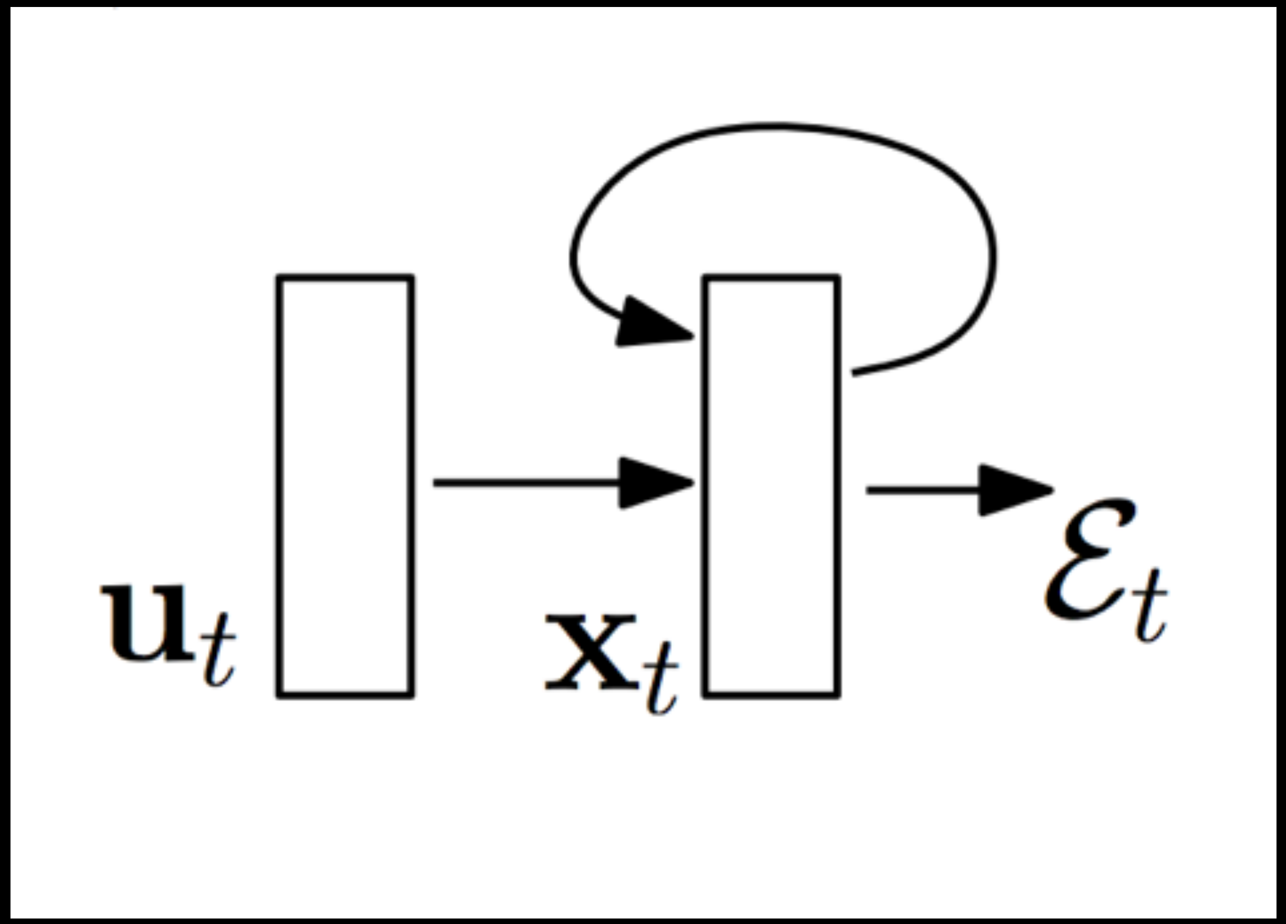


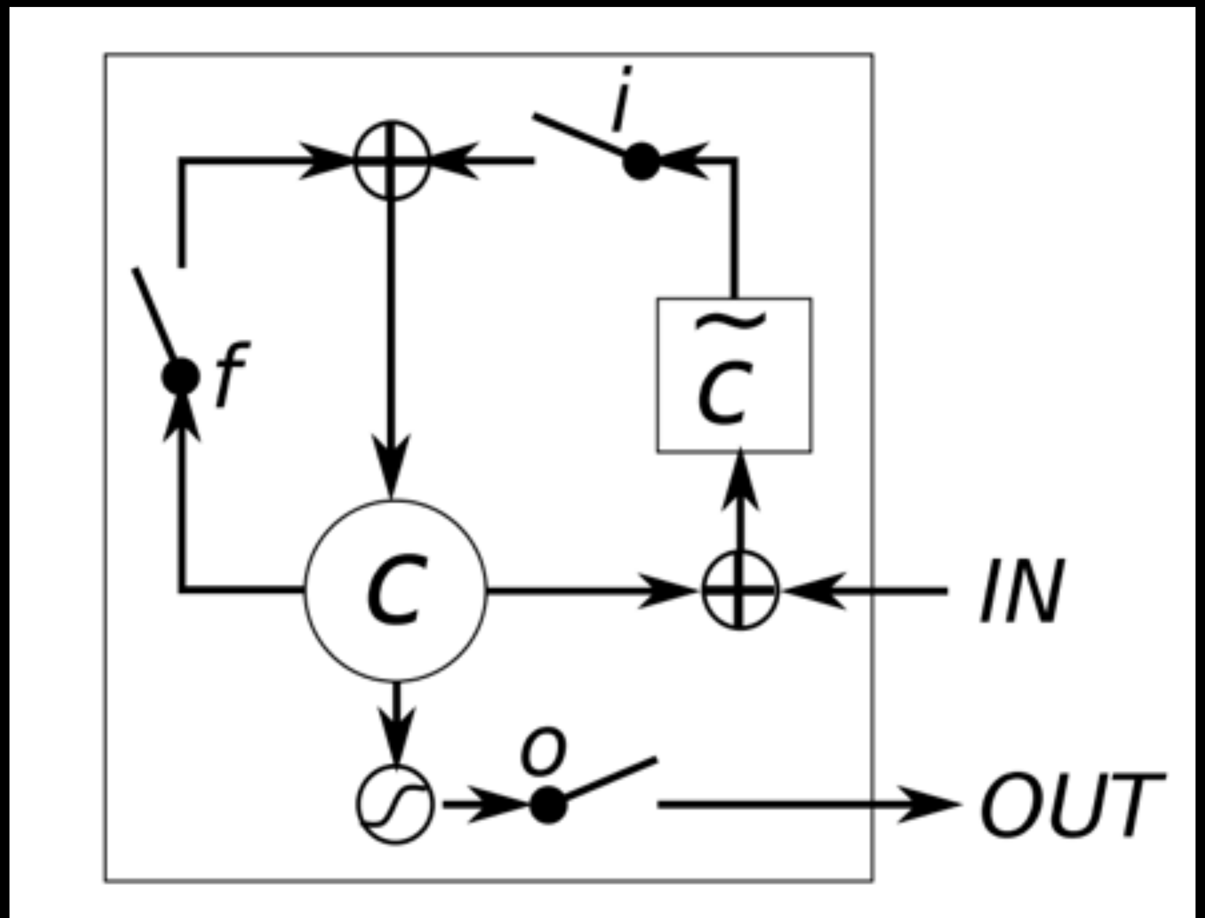
RECURRENT NEURAL NETWORKS

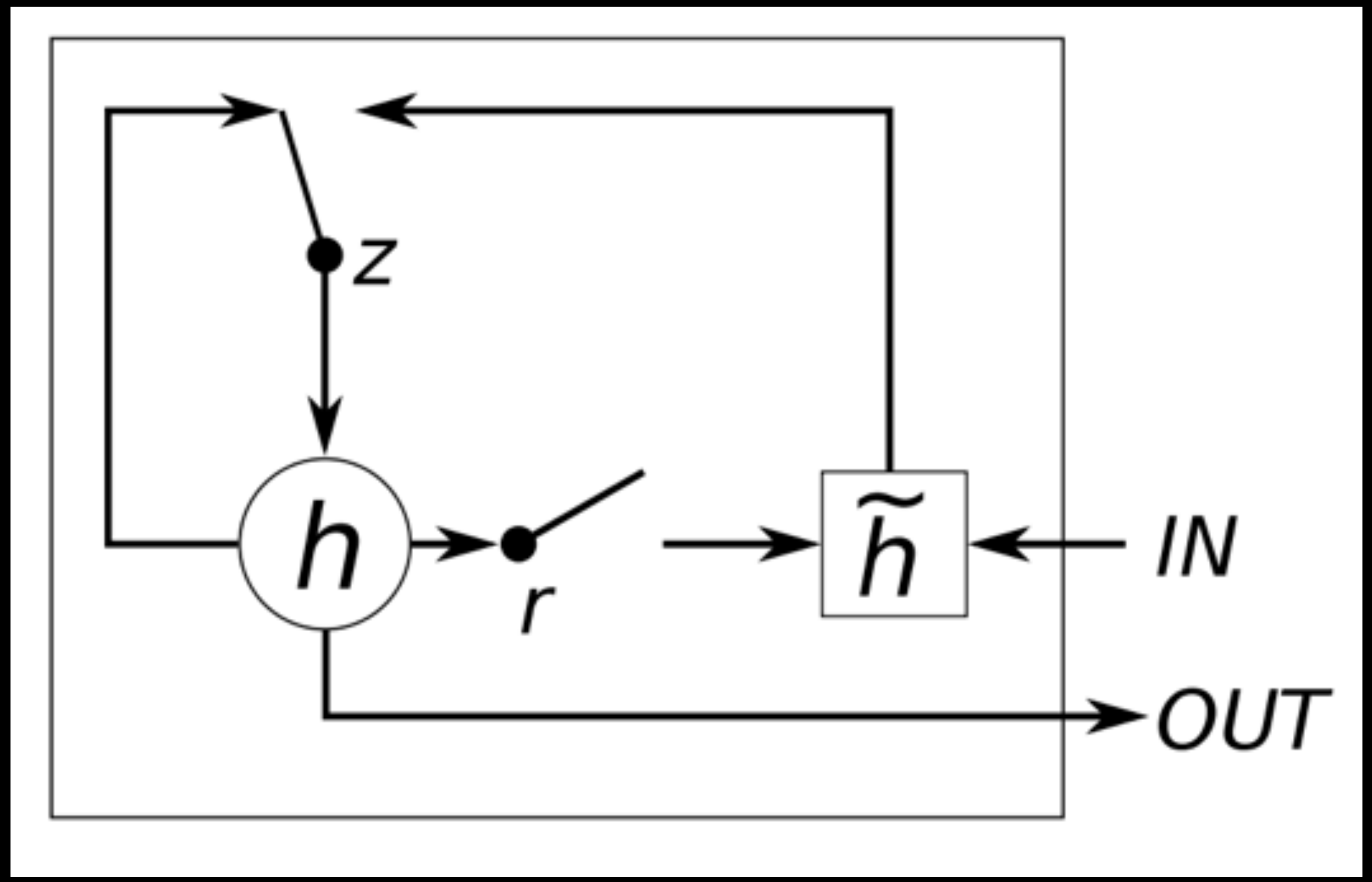


[Karpathy 2015]

TYPES OF RNNS







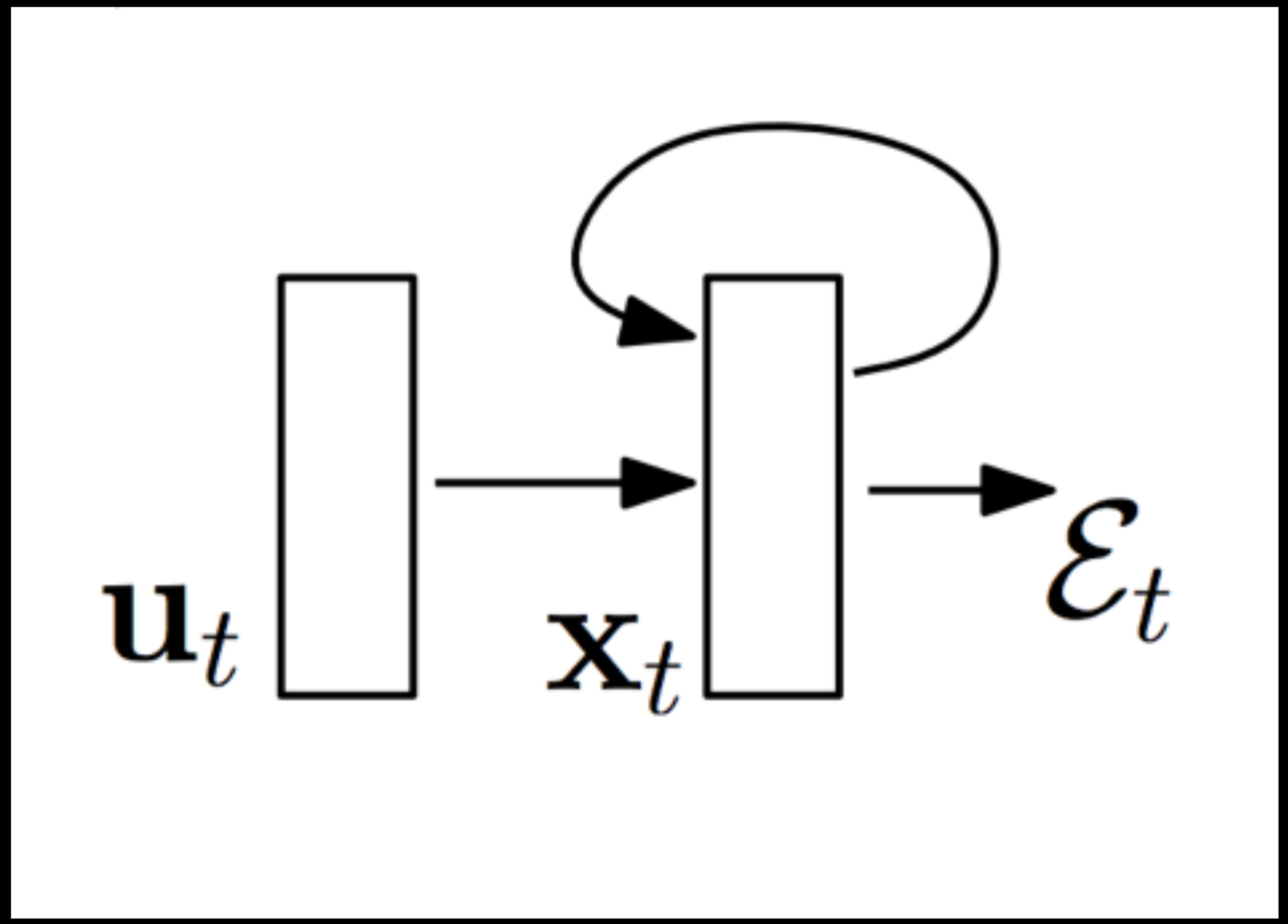
"Vanilla"
RNN
(tanh)

LSTM
(Long Short-Term Memory)

GRU
(Gated Recurrent Unit)

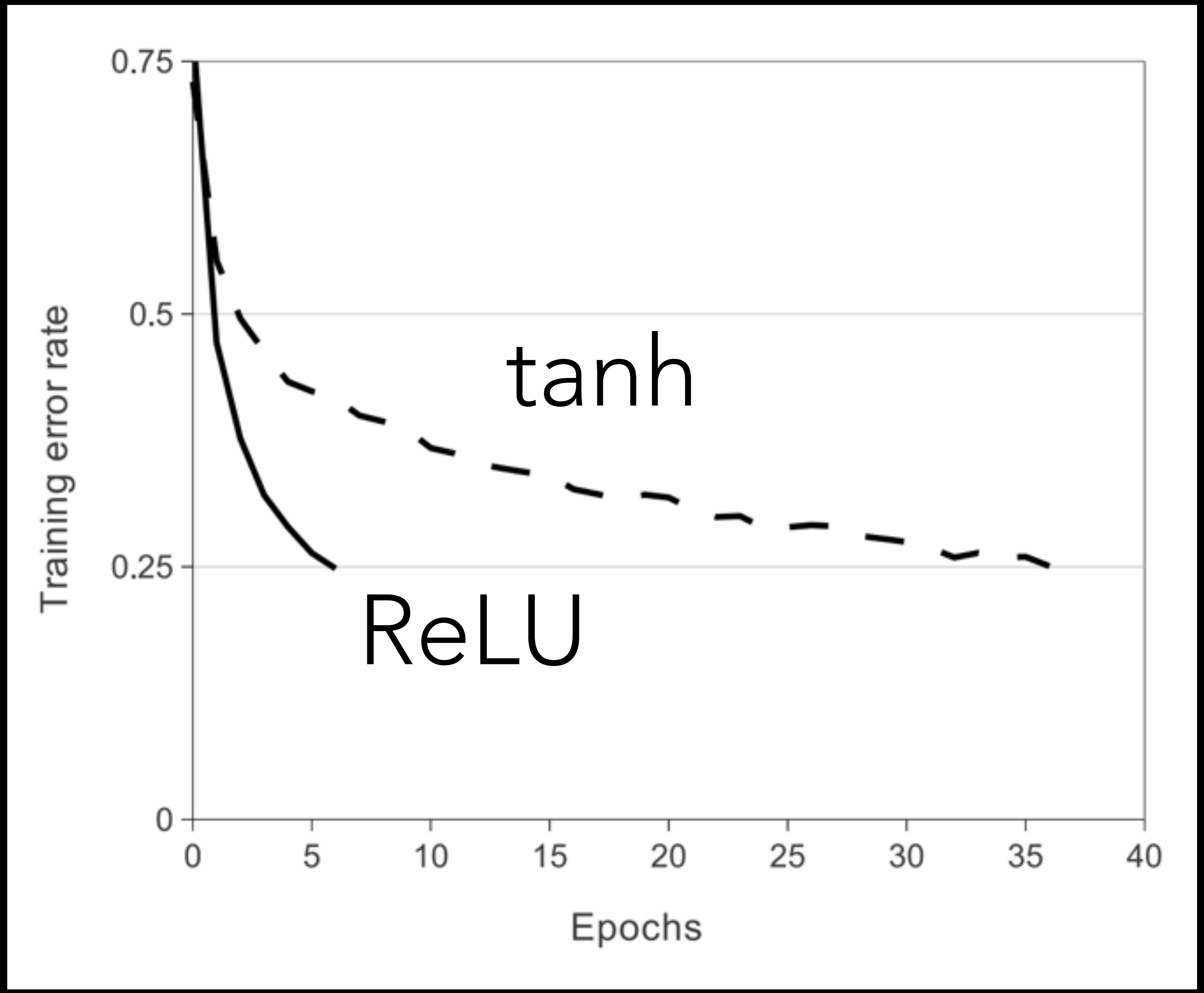
[Rumelhart 1986], [Hochreiter and Schmidhuber 1997], [Cho 2014]

CAN WE USE RELU WITH AN RNN?



"Vanilla"
RNN
(tanh)

- Replacing tanh with ReLU gave huge gains for AlexNet
- Is there some way to use ReLU with RNNs?



[Krizhevsky 2012]

A Simple Way to Initialize Recurrent Networks of Rectified Linear Units

Quoc V. Le, Navdeep Jaitly, Geoffrey E. Hinton
Google

Methods	Test perplexity
LSTM (512 units)	68.8
IRNN (4 layers, 512 units)	69.4
IRNN (1 layer, 1024 units + linear projection with 512 units before softmax)	70.2
RNN (4 layer, 512 tanh units)	71.8
RNN (1 layer, 1024 tanh units + linear projection with 512 units before softmax)	72.5

Table 3: Performances of recurrent methods on the 1 billion word benchmark.