Convolutional Neural Networks for Computer Vision Applications

林彥宇 副研究員
Yen-Yu Lin, Associate Research Fellow

中央研究院 資訊科技創新研究中心
Research Center for IT Innovation, Academia Sinica

About Yen-Yu Lin

- Yen-Yu Lin, *Associate research fellow, CITI, Academia Sinica*

- Research interests:
  - Computer Vision (CV):

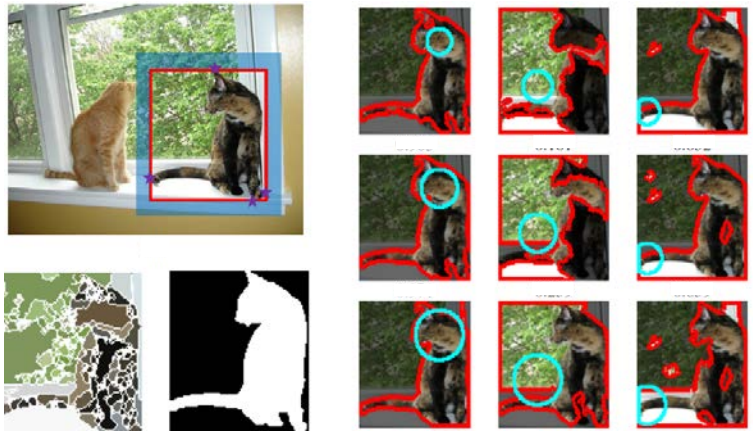
    Let computers see, recognize, and interpret the world like humans
  - Machine Learning (ML):

    A statistical way to learn how human visual system works
  - Goal: *Design **ML** methods to facilitate **CV** applications*

Research Topics 1/3

bicycles? ○
dogs? ✖
trains? ✖
persons? ○

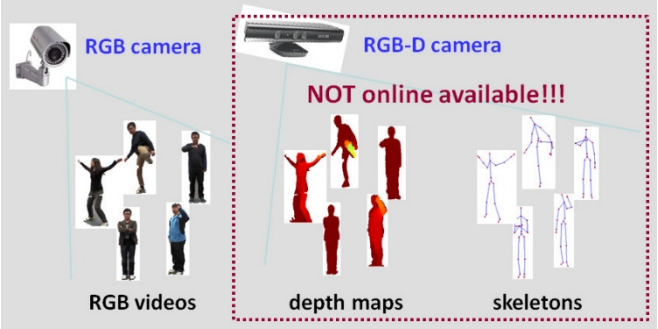
CV: *object recognition*
ML: *multiple kernel learning*
TPAMI'11, ICCV'09, NIPS'08
最核心、最基本的問題就是物件識別

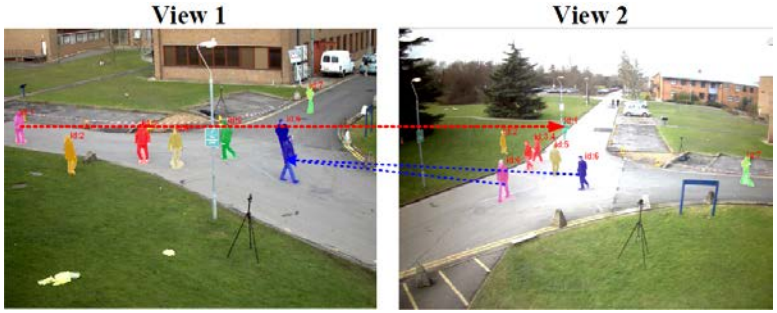


CV: *image segmentation*
ML: *graphical model*
CVPR'14, TIP'14, ACCV'12

CV: *face detection* **ML**: *multi-task boosting*
US Patent'07, CVPR'05, ECCV'04

Research Topics 2/3





CV: *action recognition*
ML: *low-rank reconstruction*
TIP'15, CVPR'14

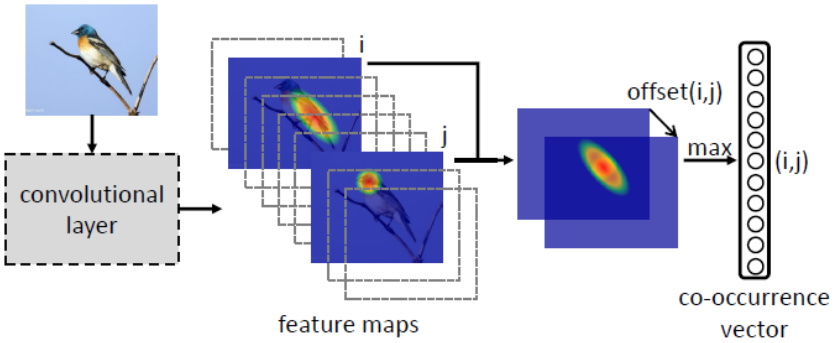
CV: *multi-view people counting*
ML: *transfer learning*
TIP'15, ACM MM'12

SIFT LIOP DAISY RI GB OURS!

CV: *image matching* **ML**: *energy minimization* 手機照相全景，找相同的點疊起來

CVPR'16, TPAMI'15, TIP'15, CVPR'15, CVPR'13

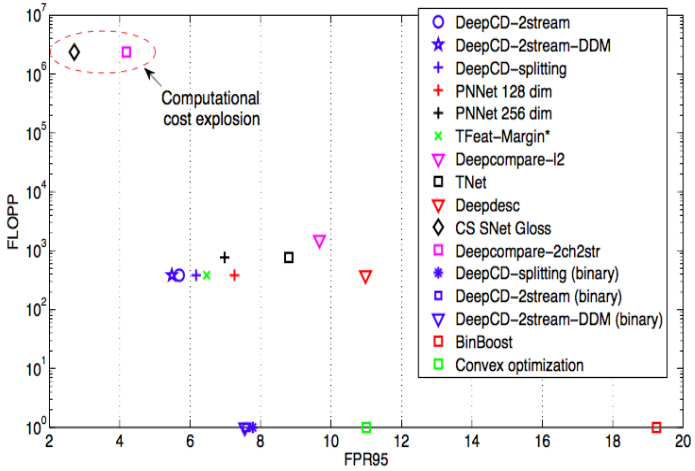
Research Topics 3/3



CV: *fine-grained object recognition*

ML: *CNNs with co-occurrence layer*

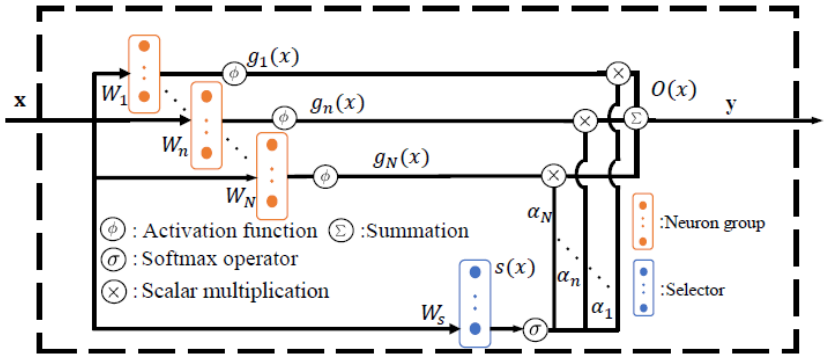
CVPR'17



CV: *patch descriptor learning*

ML: *CNNs with adaptive learning rate*

ICCV'17



CV: *gesture recognition*

ML: *DNNs with adaptive hidden layer*

AAAI'18

Outline

- Convolutional neural networks (CNNs)
- Representative CNN models
- CNN-based computer vision applications

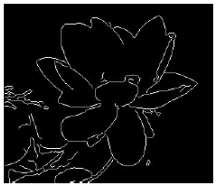
Outline

- Convolutional neural networks (CNNs)
  - Conventional approaches vs. deep learning
  - Neural networks
  - Convolutional neural networks
- Representative CNN models
- CNN-based computer vision applications

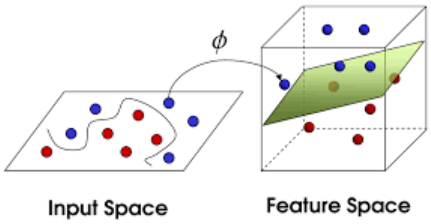
Conventional approach to object recognition

- Training phase

image collection → feature extraction → classifier training → trained classifier



histogram of oriented gradients (HoG)



support vector machines (SVMs)

dogs? ✖

flowers? ○

trains? ✖

persons? ✖

Conventional approach to object recognition

- Training phase

key step

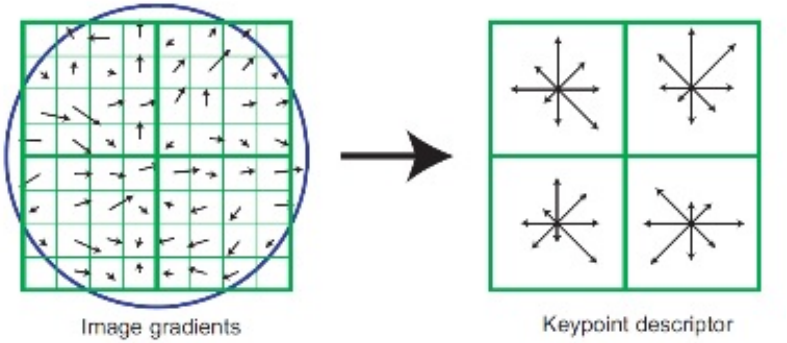
image collection → feature extraction → classifier training → trained classifier

- Testing phase

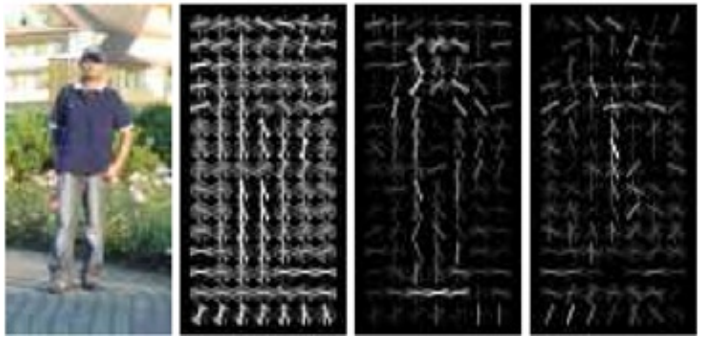
test image → feature extraction → trained classifier → prediction

Features are the keys

- Off-the-shelf visual features

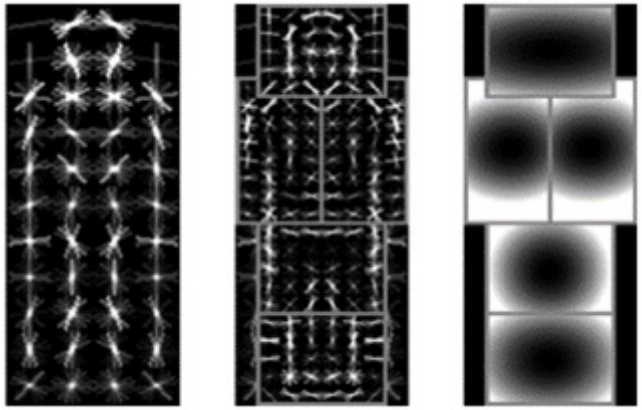


SIFT [Lowe, IJCV'04]
Citations: 43465



HoG [Dalal & Triggs, CVPR'05]
Citations: 20174

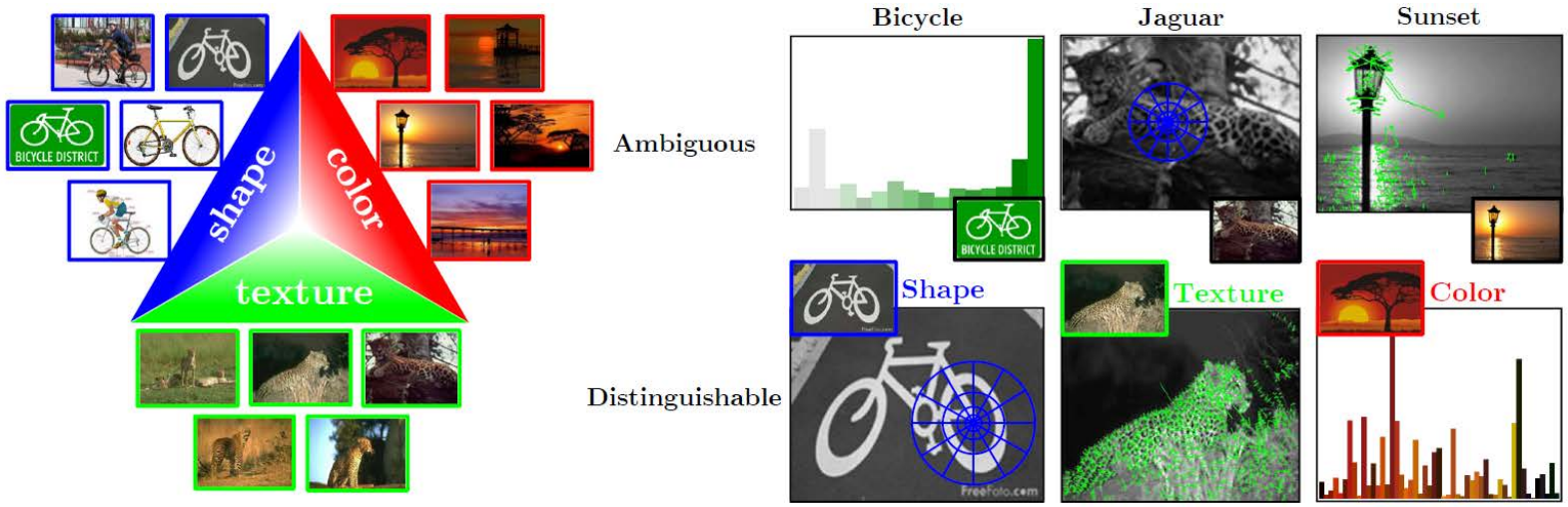
Constellation model [Fergus et al., CVPR'03]
Citations: 2551



DPM [Felzenszwalb et al., PAMI'10]
Citations: 5093

Features are the keys

- Features are the keys to recent progress in classification
- Are handcrafted features optimal?
- The optimal features for classification in general vary from task to task, even from category to category



Conventional approaches vs. Deep learning

- Conventional approaches
  - Fixed/engineered features + trainable classifier

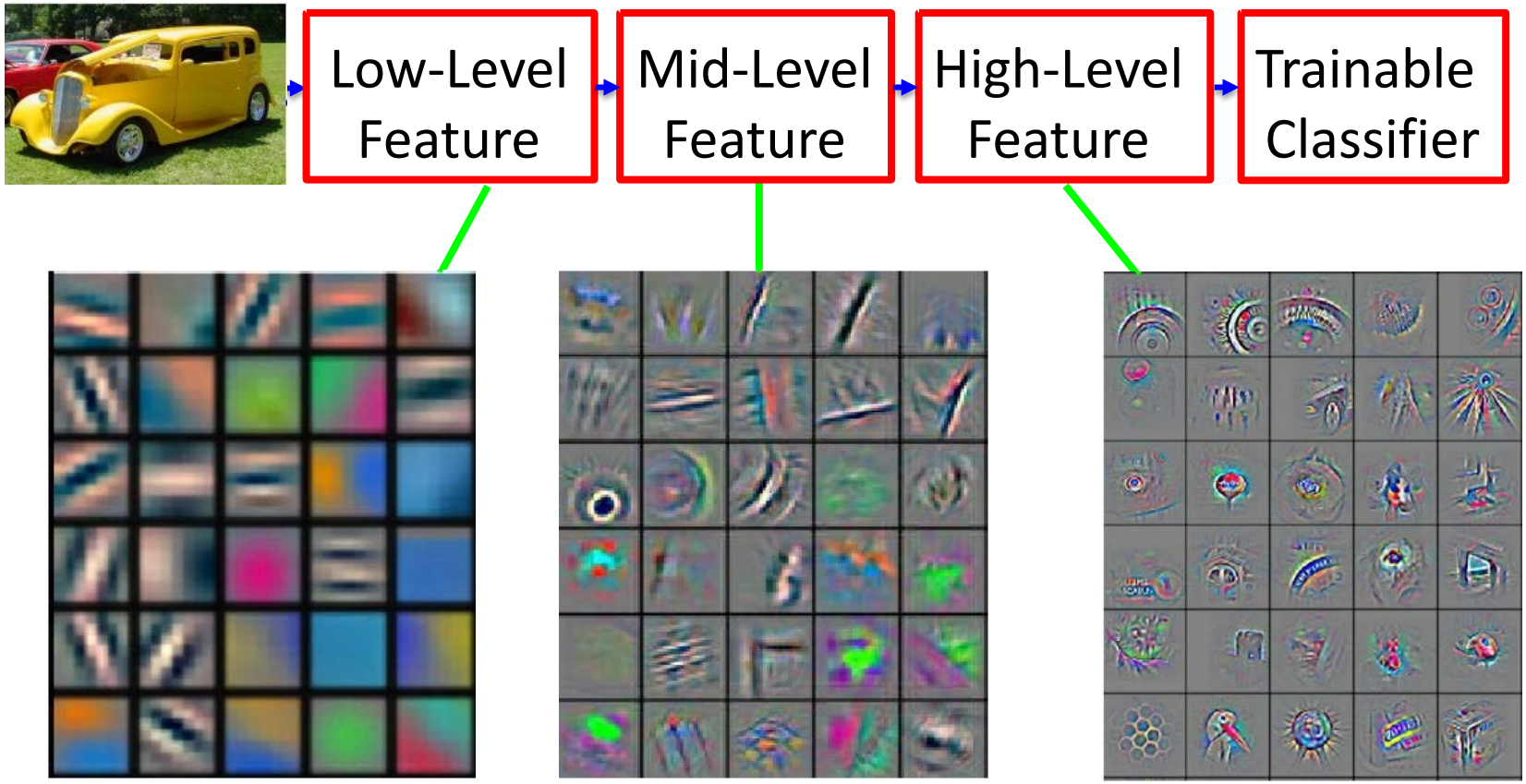
Hand-crafted Feature Extractor → "Simple" Trainable Classifier →

- Deep learning / End-to-end learning / Feature learning
  - Trainable features + trainable classifier

Trainable Feature Extractor → **Trainable** Classifier →

slide: Y LeCun & MA Ranzato

Deep learning = Learning hierarchical representations



slide: Y LeCun & MA Ranzato

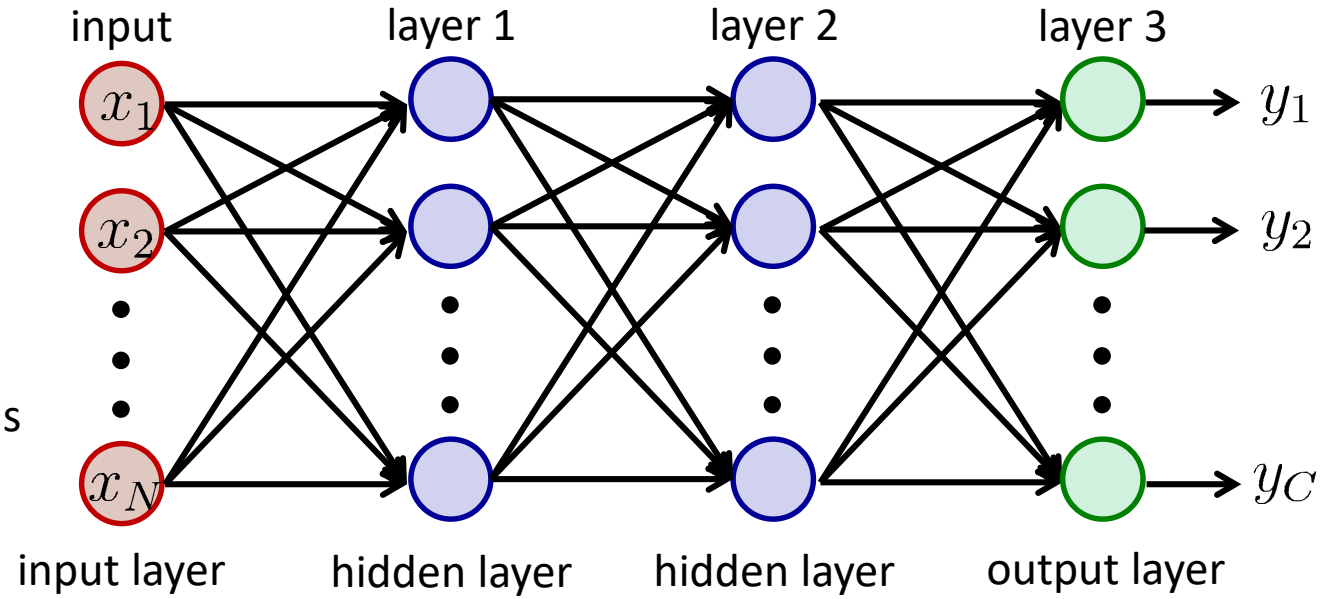
Outline

- Convolutional neural networks (CNN)
  - Conventional approaches vs. deep learning
  - Neural networks
  - Convolutional neural networks
- Representative CNN models
- CNN-based computer vision applications

Neural networks and neurons

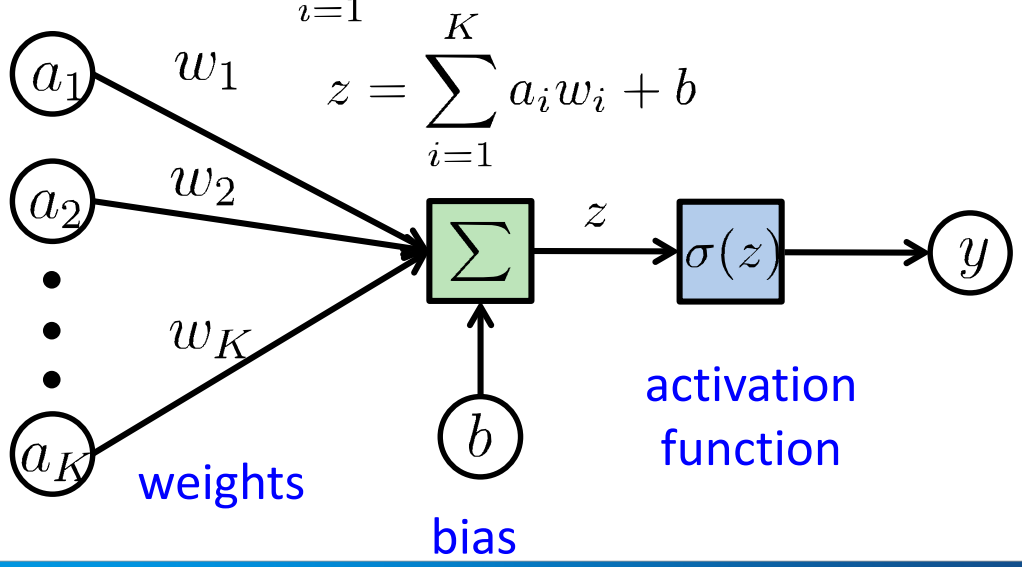
- Neural networks are presented as layers of interconnected neurons
  - Each layer of neurons takes messages from output of previous layer

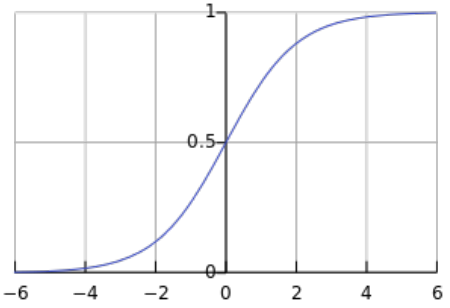
image of N pixels



A single neuron

- A function $f : R^K \mapsto R$
  - Map K inputs to 1 output
  - Compute the biased weighted sum
  - Apply a non-linear mapping function (activation function)
  - $f((a)) = \sigma(\sum_{i=1}^{K} a_i w_i + b)$, where $\sigma(z) = \frac{1}{1+\exp(-z)}$





sigmoid function

Training neural networks

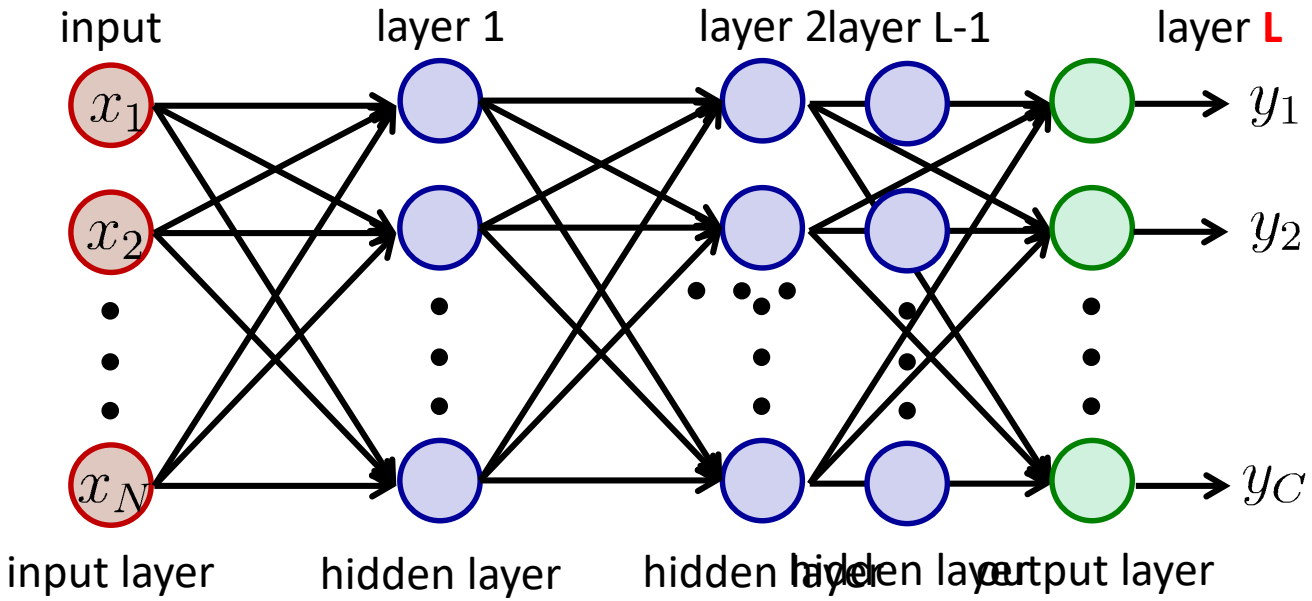
- Collect a set of labeled training data $D = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N}$
- Training neural networks: Finding network parameters $\theta = \{\mathbf{w}, \mathbf{b}\}$ to minimize the loss between true training label $\mathbf{y}_i$ and the estimated label, e.g.,

$$L(\theta) = \sum_{i=1}^{N} \|\mathbf{y}_i - g_{\mathbf{w}}(\mathbf{x}_i)\|^2$$

- Minimization can be done by gradient descent if $L(\cdot)$ is differentiable with respect to θ
- Back-propagation: a widely used method for optimizing multi-layer neural networks

What is deep neural networks (DNN)

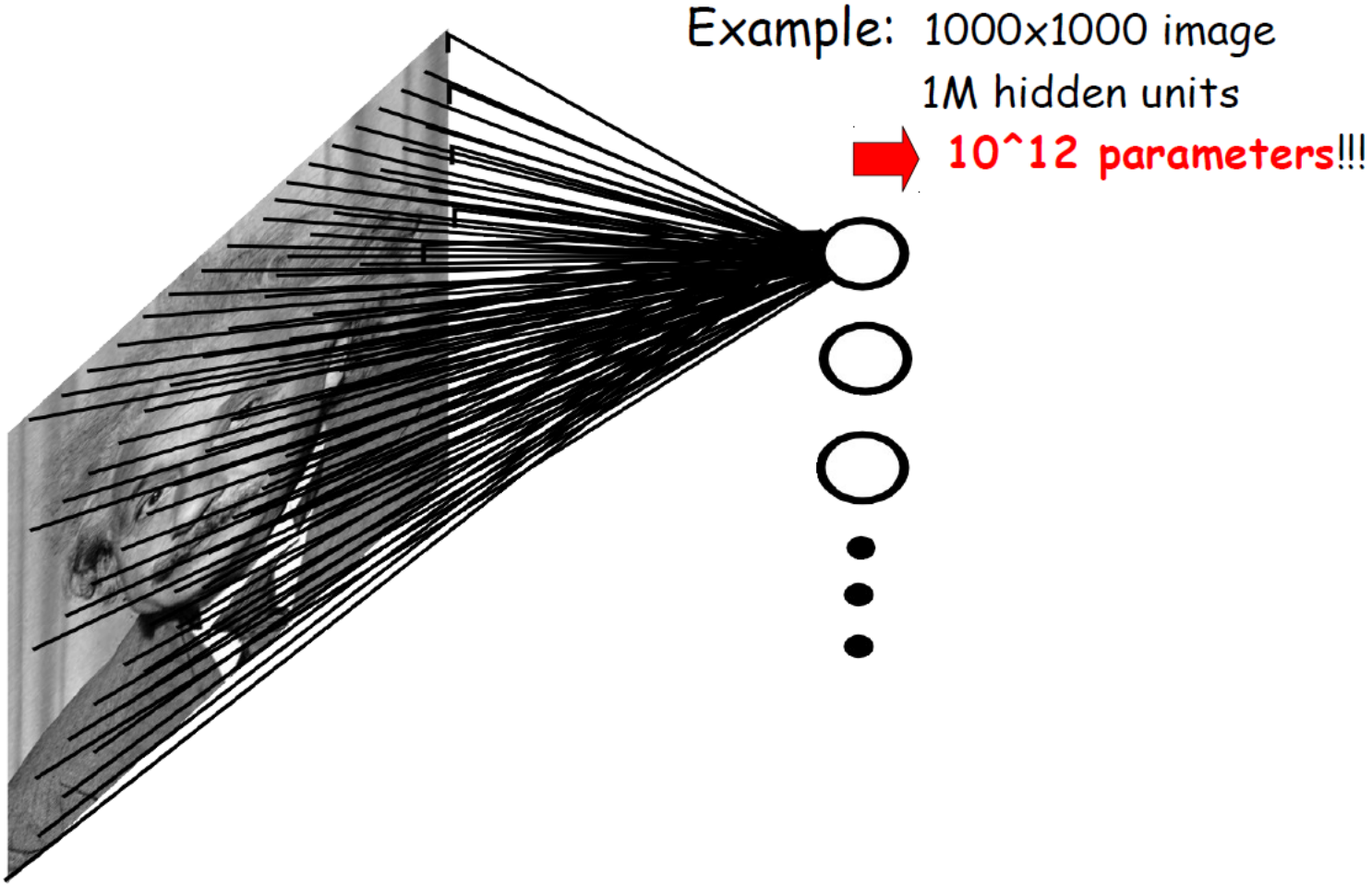
- DNN is neural networks with many hidden layers



Outline

- Convolutional neural networks (CNN)
  - Conventional approaches vs. deep learning
  - Neural networks
  - Convolutional neural networks
- Representative CNN models
- CNN-based computer vision applications

of parameters in fully connected NN

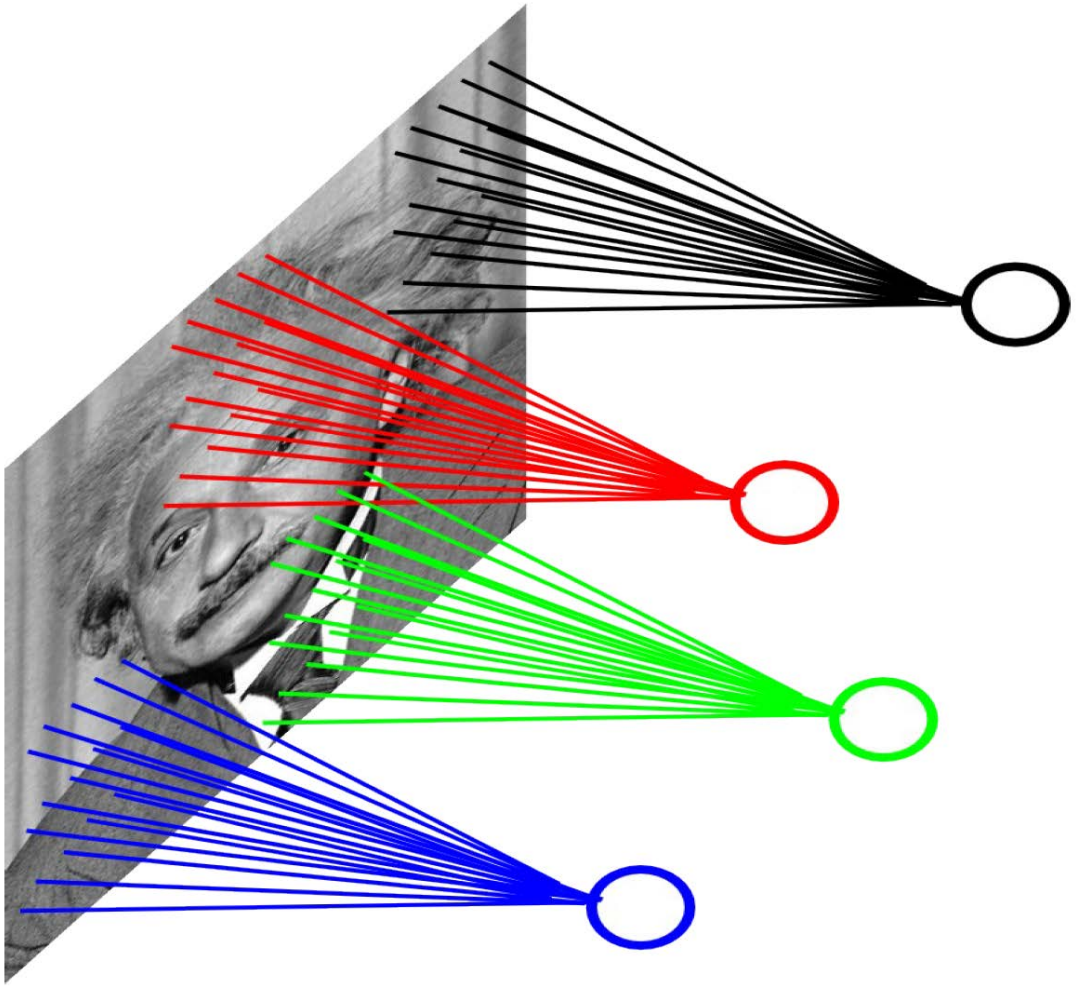


slide: MA Ranzato

Convolutional neural networks (CNN)

- CNN: a multi-layer neural network with
  1. Local connectivity
  2. Weight sharing
- Why local connectivity?
  - Spatial correlation is local (locality of spatial dependencies)
  - Reduce # of parameters
- Why weight sharing?
  - Statistics is at different locations (stationarity of statistics)
  - Reduce # of parameters

of parameters in fully connected NN



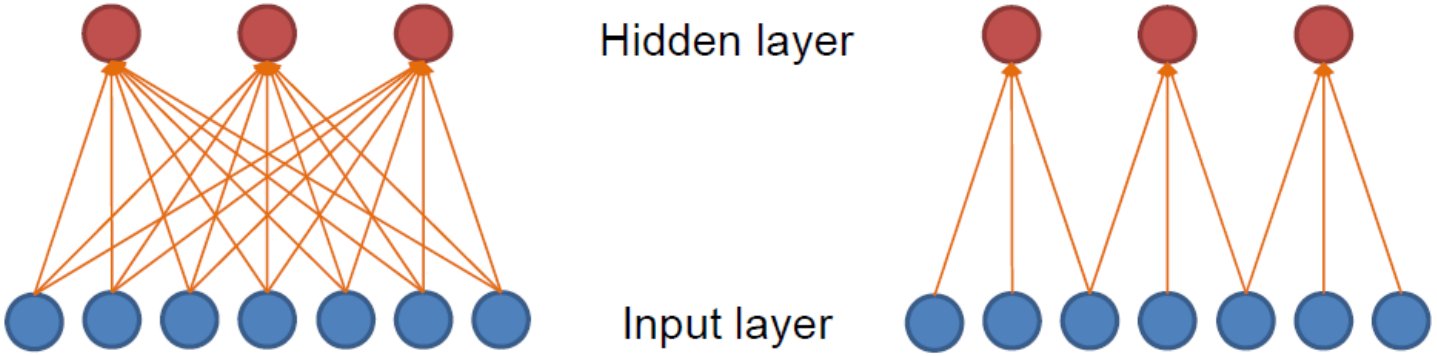
0x1000 image

nidden units

^12 parameters!!!

slide: MA Ranzato

CNN: Local connectivity



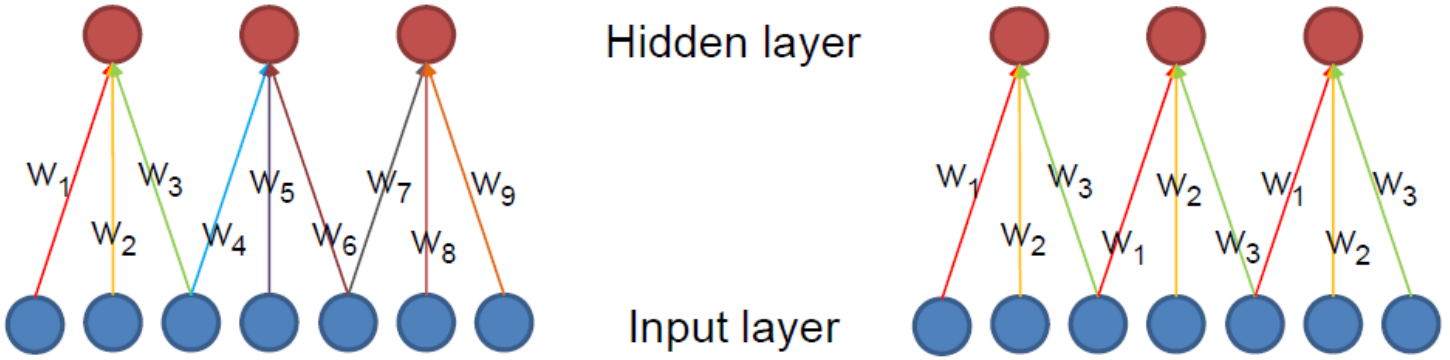
Global connectivity

Local connectivity

- # input units (neurons): 7
- # hidden units: 3
- Number of parameters
  - Global connectivity: 3 x 7 = 21
  - Local connectivity: 3 x 3 = 9

slide: J.-B. Huang

CNN: Weight sharing



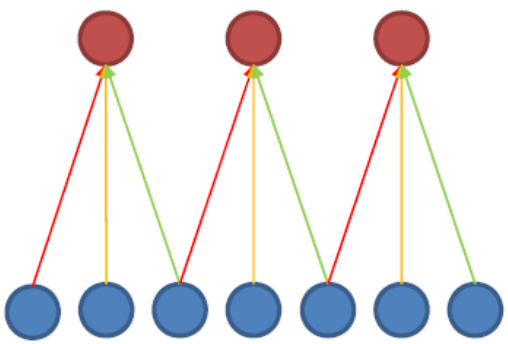
Without weight sharing

With weight sharing

- # input units (neurons): 7
- # hidden units: 3
- Number of parameters
  - Without weight sharing: 3 x 3 = 9
  - With weight sharing : 3 x 1 = 3

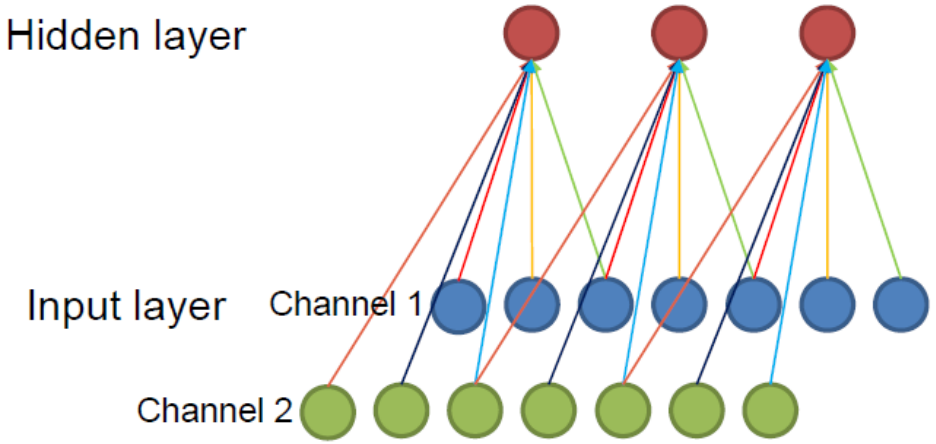
slide: J.-B. Huang

CNN with multiple input channels



Single input channel

Filter weights

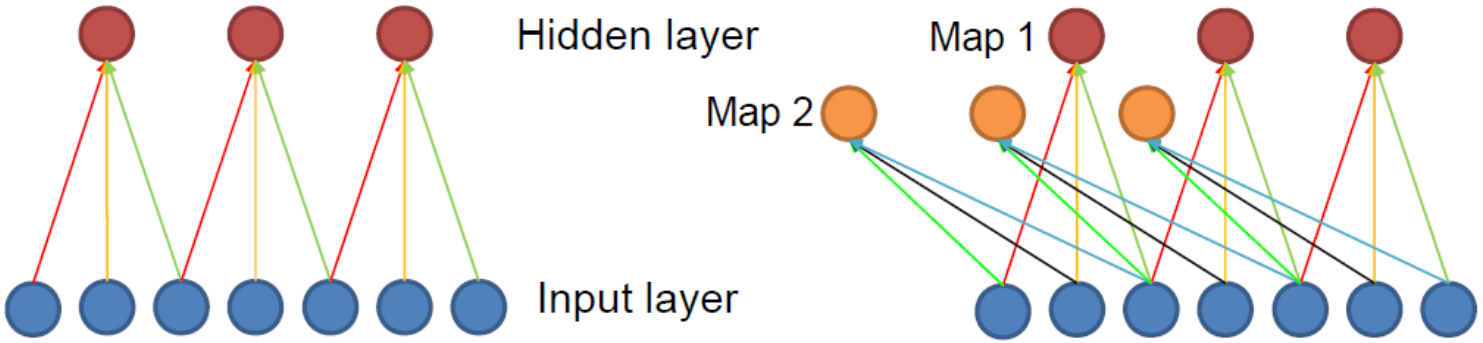


Multiple input channels

Filter weights

slide: J.-B. Huang

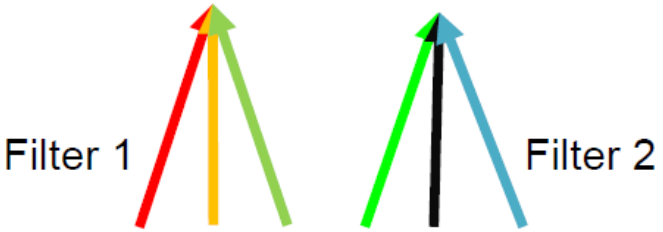
CNN with multiple output channels



Single output map

Filter weights

Multiple output maps



Filter weights

slide: J.-B. Huang

Putting them together

- Local connectivity
- Weight sharing
- Handling multiple input channels
- Handling multiple output maps

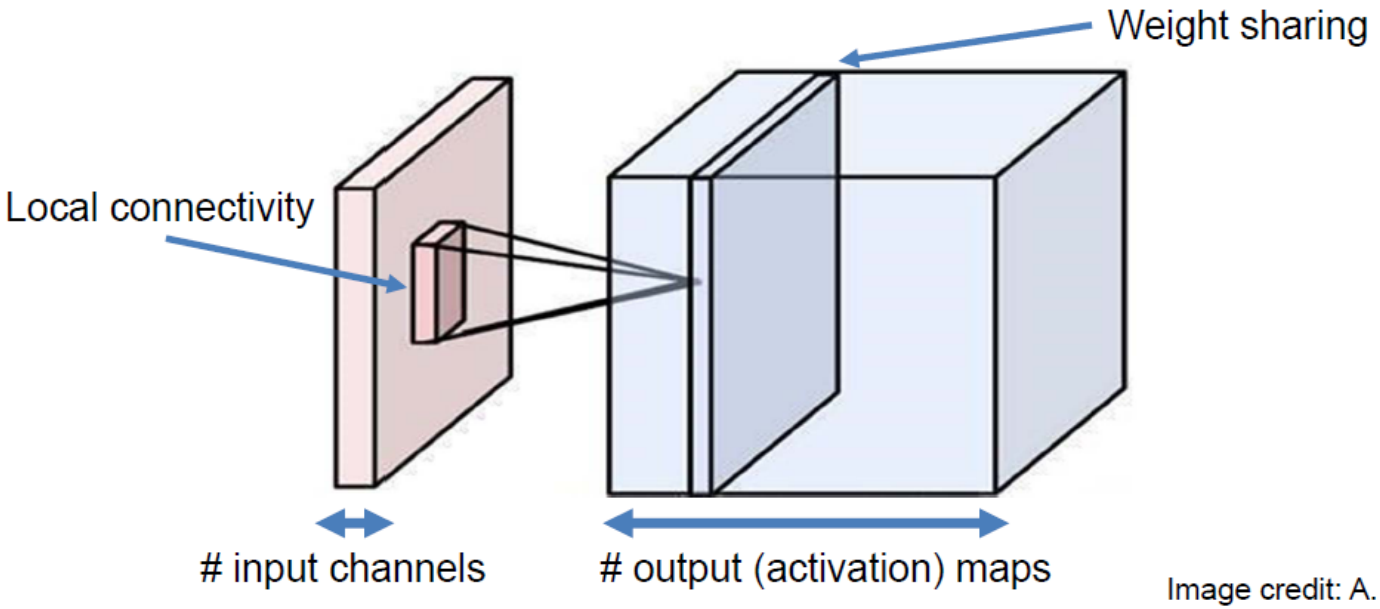
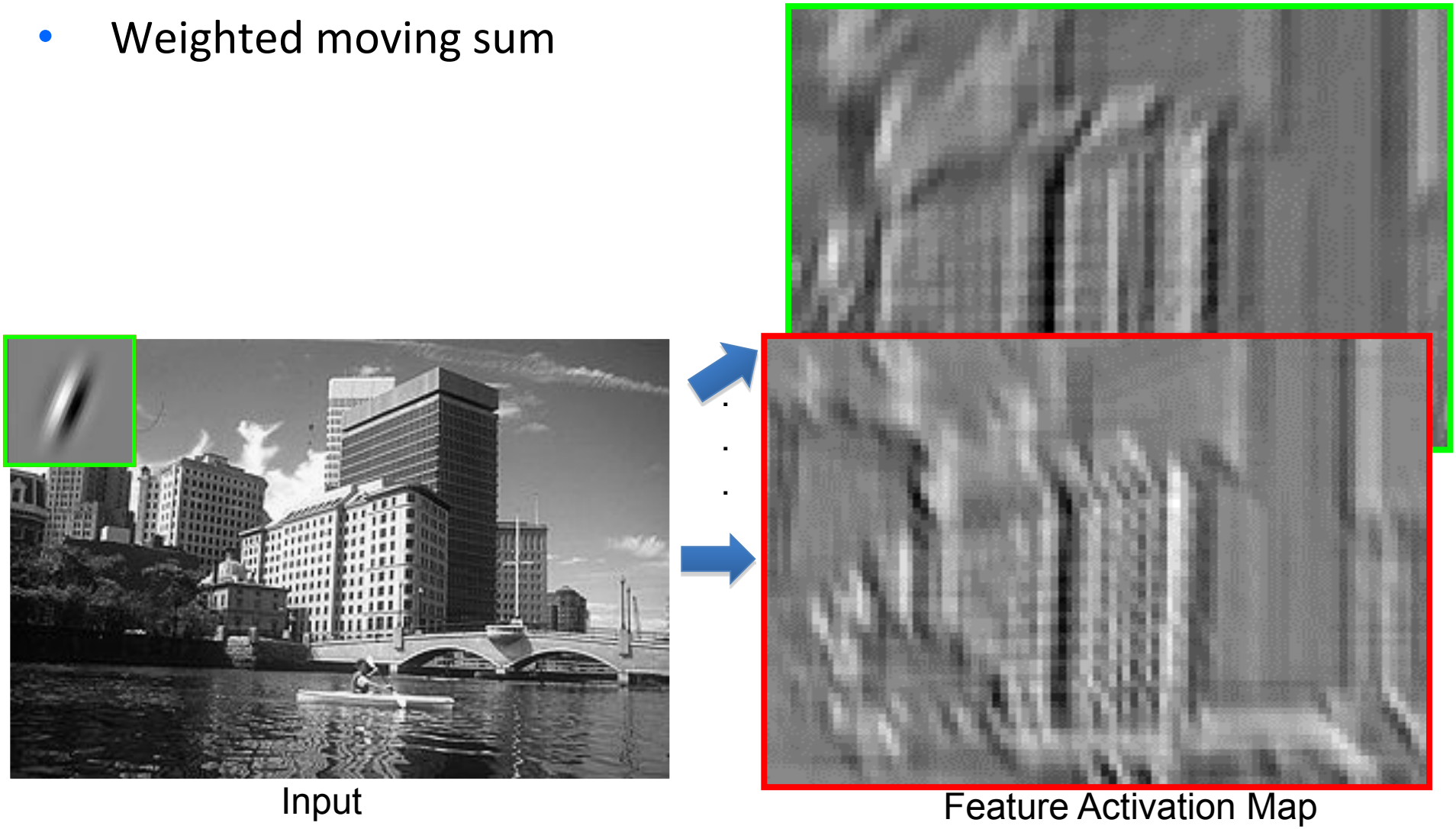


Image credit: A. Karpathy

slide: J.-B. Huang

What is a Convolution?

- Weighted moving sum

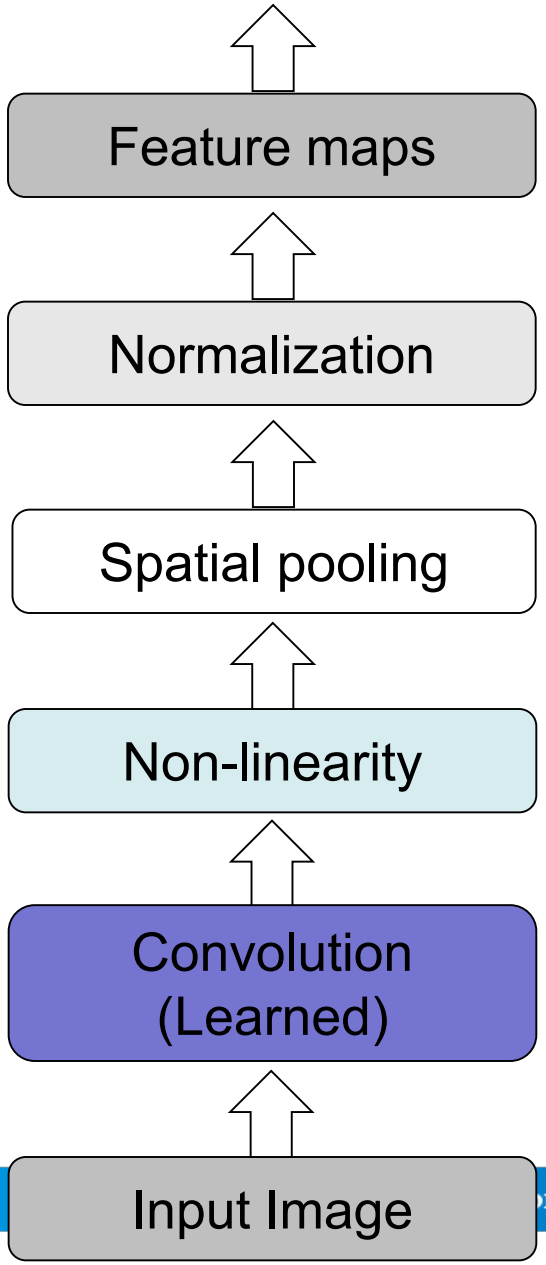


Input

Feature Activation Map

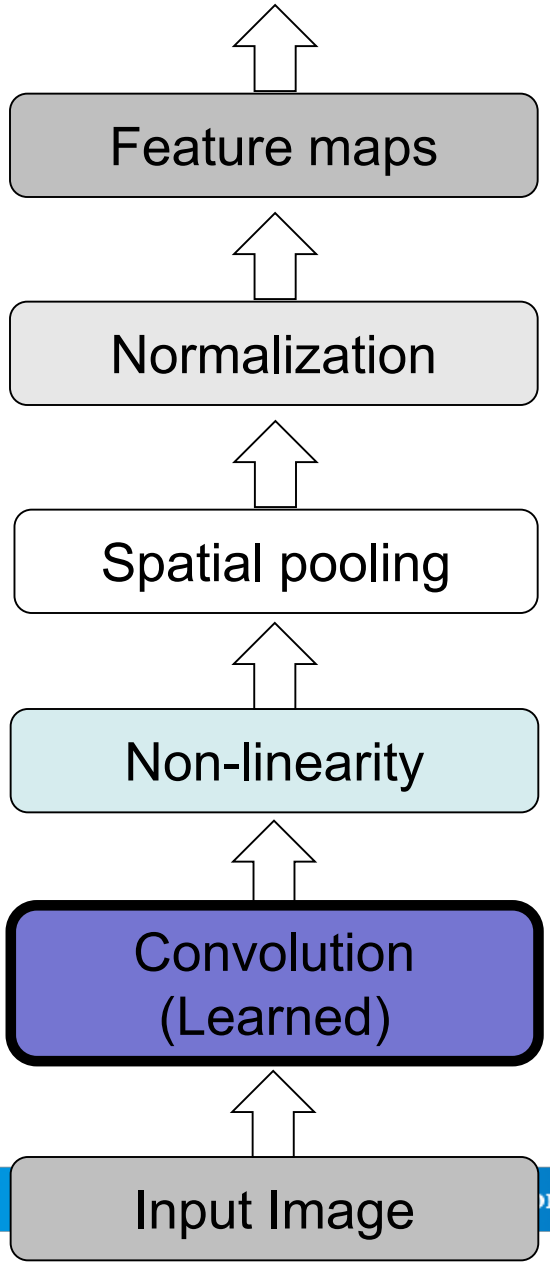
slide: S. Lazebnik

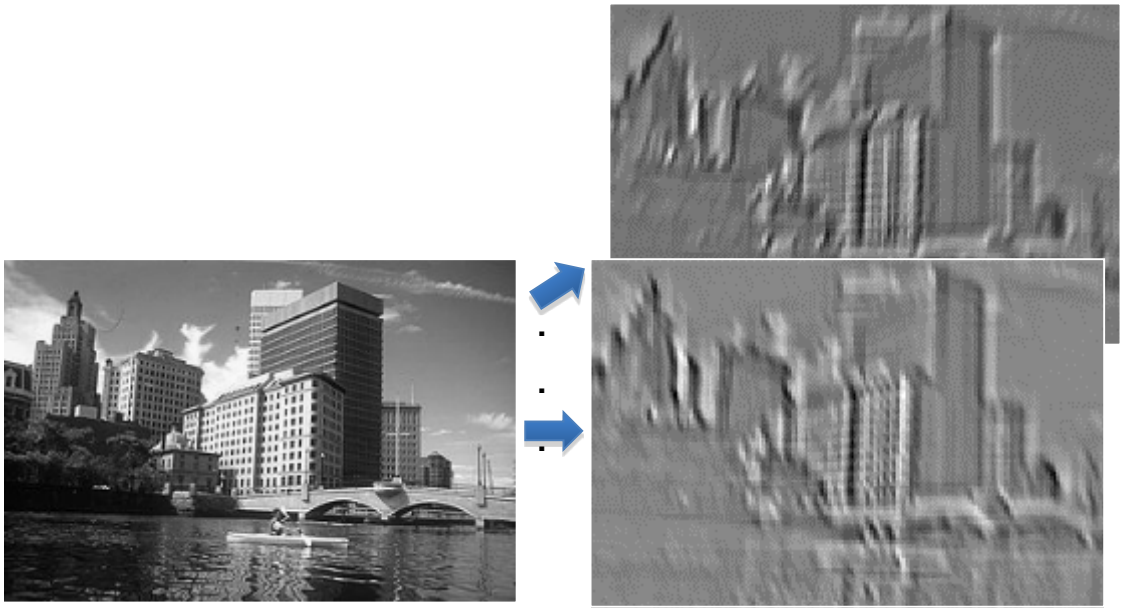
Convolutional Neural Networks



slide: S. Lazebnik

Convolutional Neural Networks



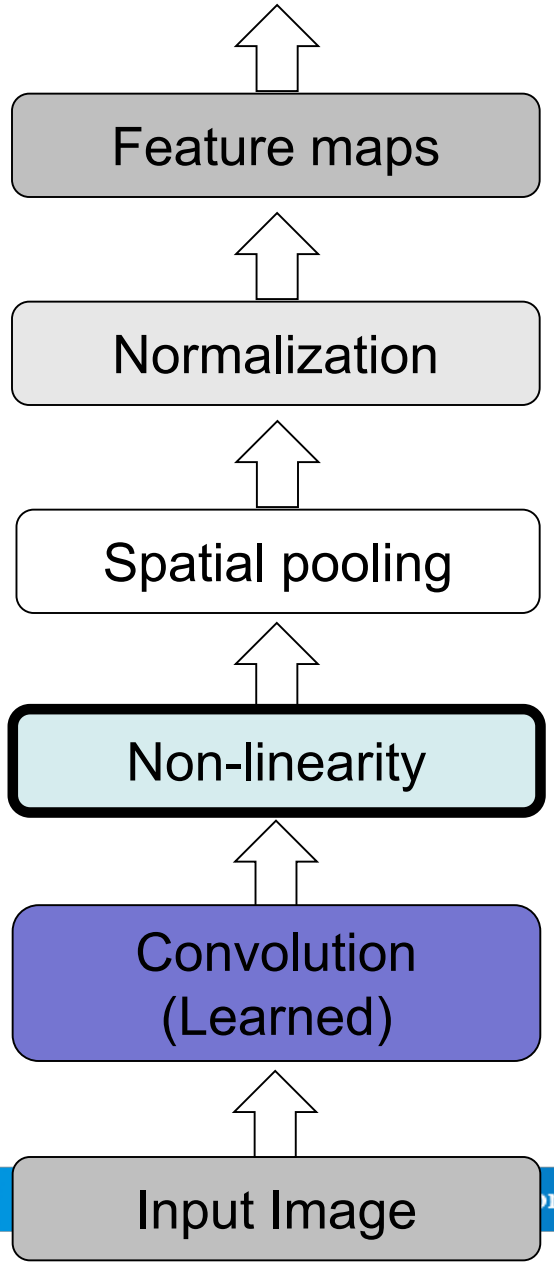


Input

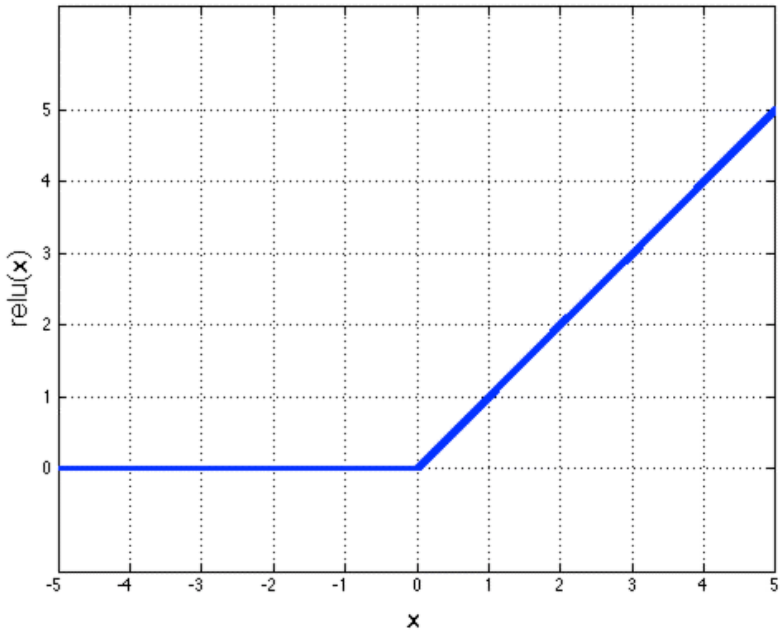
Feature Map

slide: S. Lazebnik

Convolutional Neural Networks

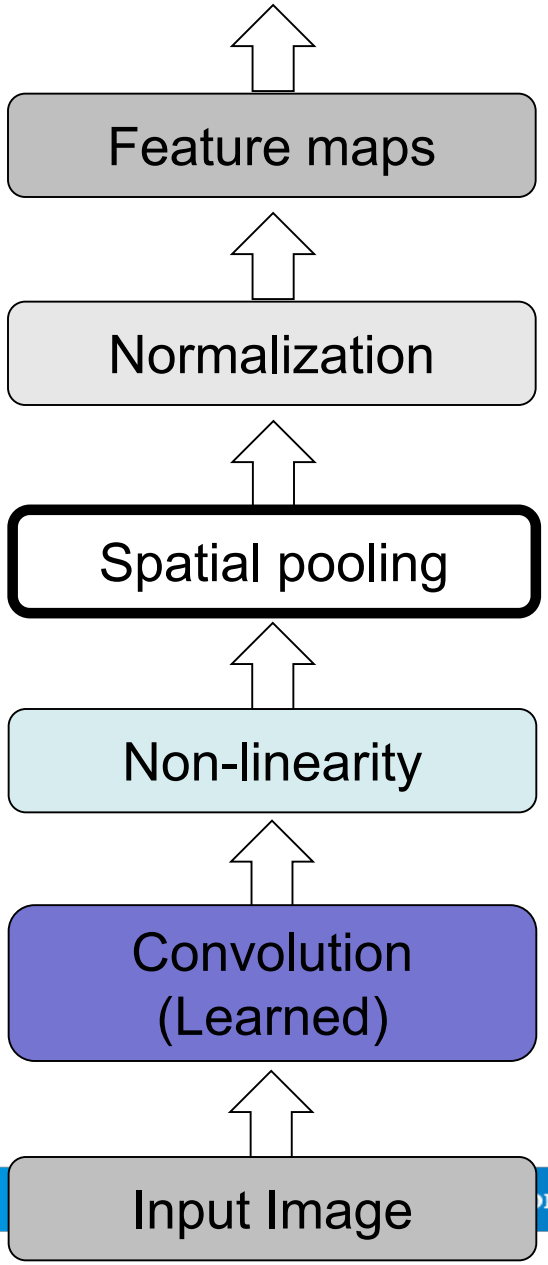


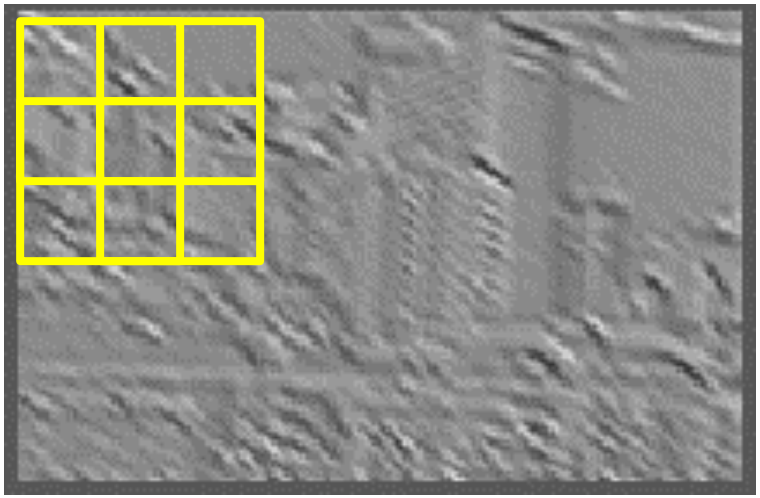
Rectified Linear Unit (ReLU)



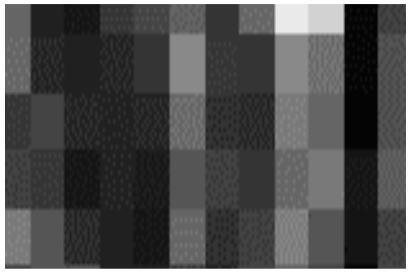
slide: S. Lazebnik

Convolutional Neural Networks





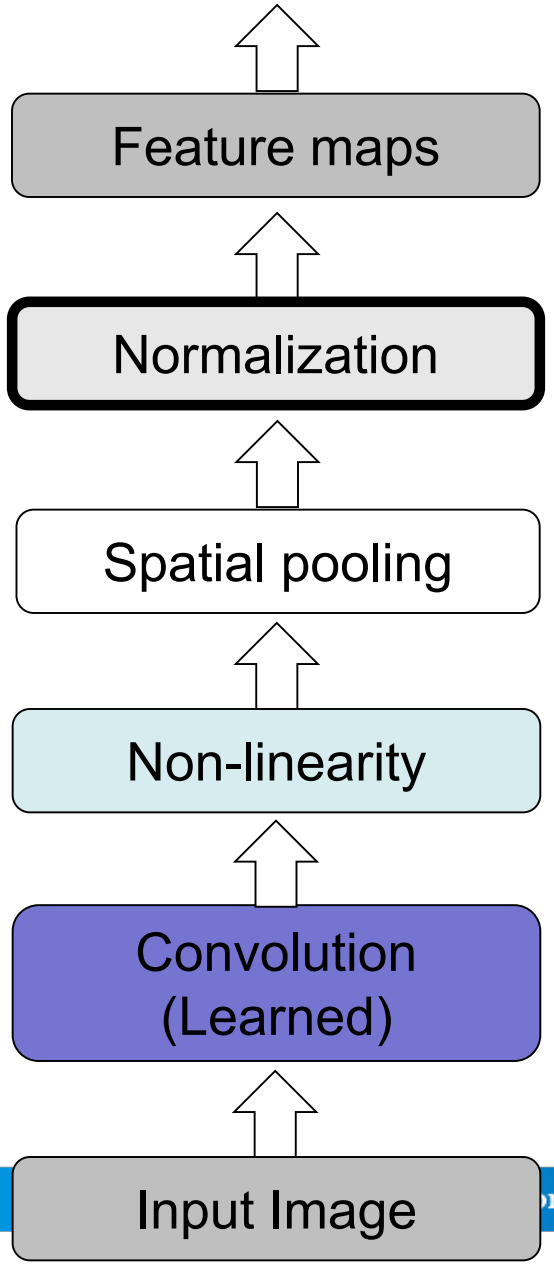
Max pooling

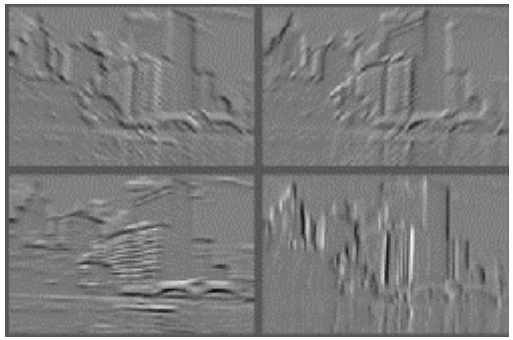


Max-pooling: a non-linear down-sampling

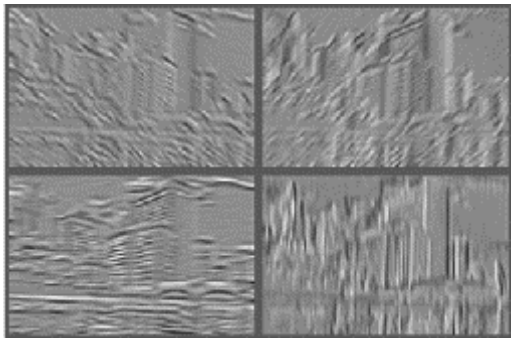
slide: S. Lazebnik

Convolutional Neural Networks





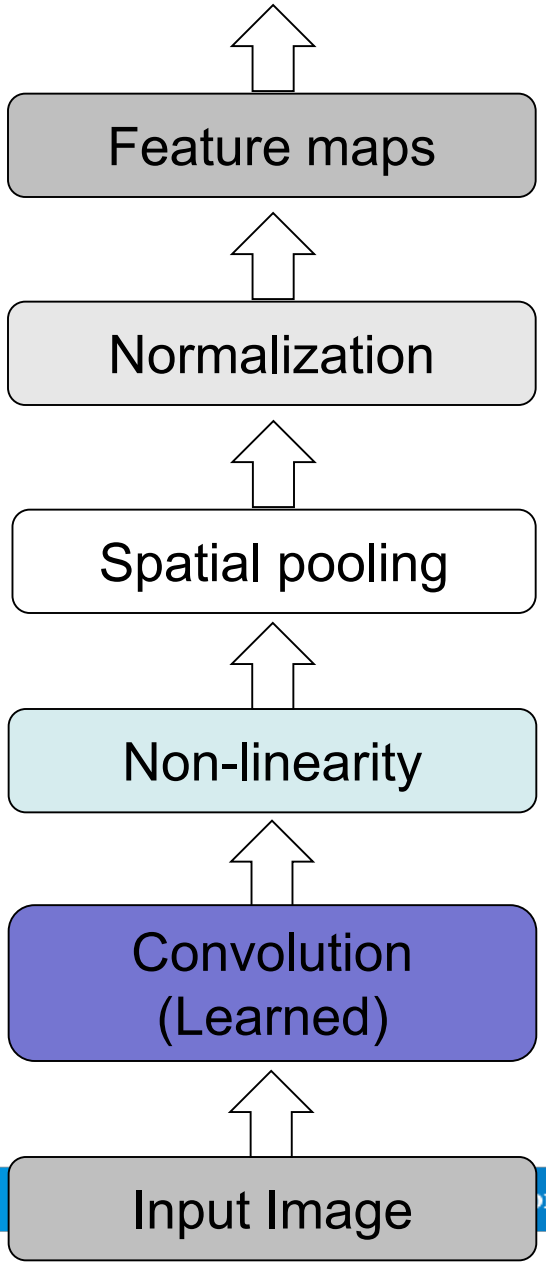
Feature Maps



Feature Maps
After Contrast
Normalization

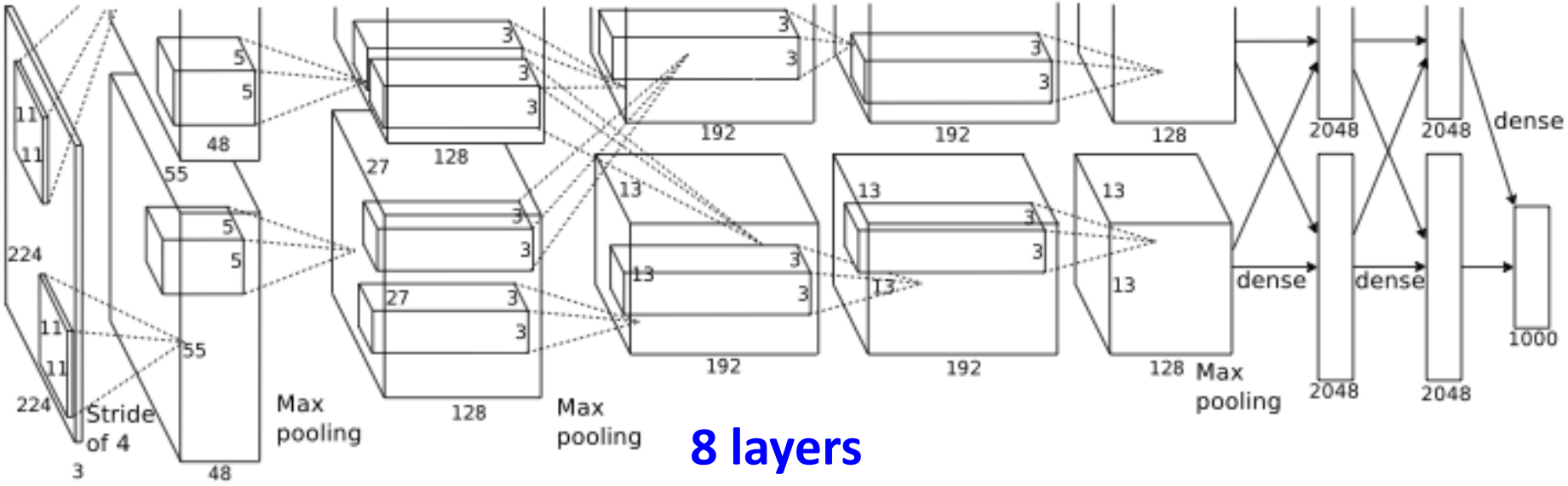
slide: S. Lazebnik

Convolutional Neural Networks



slide: S. Lazebnik

Modern CNN: AlexNet



8 layers

Input: 224*224*3=**150K**

Neurons: 290400+186624+64896+64896+43264+4096+4096+1000=**650K**

Weights: 11*11*3*48*2(35K)+5*5*48*128*2(307K)+128*3*3*192*4(884K)+192*3*3*192*2(663K)+192*3*3*128*2(442K)+6*6*128*2048*4(38M)+4096*4096(17M)+4096*1000(4M)=**60M**

- **More data (1.2M)**
- **Trained on two GPUs for a week**
- **Dropout**

slide: M. Sun

ImageNet ISLVRC 2012-2014: Object Recognition

有1000個選擇，讓你選5個類別，都選錯才算在錯誤率 Best non-convnet in 2012: 26.2%

Team	Year	Place	Error (top-5)	External data
SuperVision – Toronto (7 layers)	2012	-	16.4%	no
SuperVision	2012	1st	15.3%	ImageNet 22k
Clarifai – NYU (7 layers)	2013	-	11.7%	no
Clarifai	2013	1st	11.2%	ImageNet 22k
VGG – Oxford (16 layers)	2014	2nd	7.32%	no
GoogLeNet (19 layers)	2014	1st	6.67%	no
Human expert*			5.1%	

Team	Method	Error (top-5)
DeepImage - Baidu	Data augmentation + multi GPU	5.33%
PReLU-nets - MSRA	Parametric ReLU + smart initialization	4.94%
BN-Inception ensemble - Google	Reducing internal covariate shift	4.82%

Outline

- Convolutional neural networks (CNNs)
  - Conventional approaches vs. deep learning
  - Neural networks
  - Convolutional neural networks
- Representative CNN models
- CNN-based computer vision applications

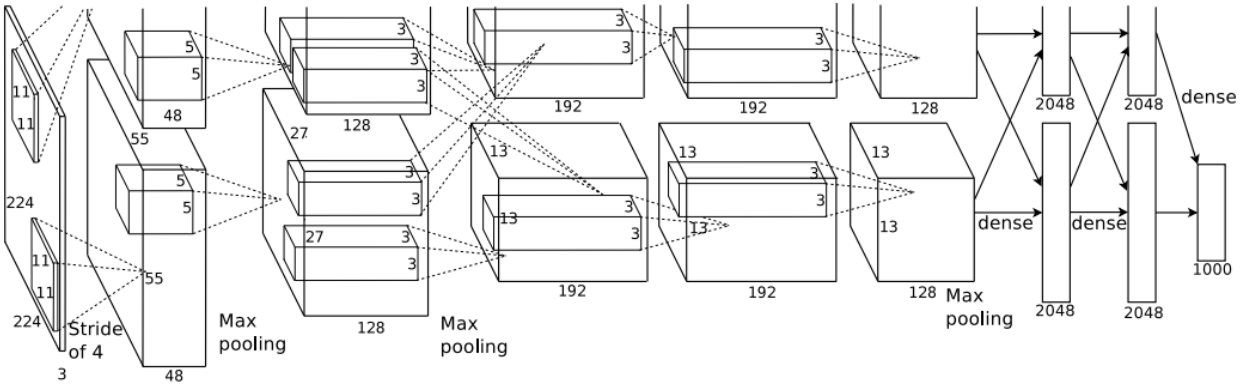
Outline

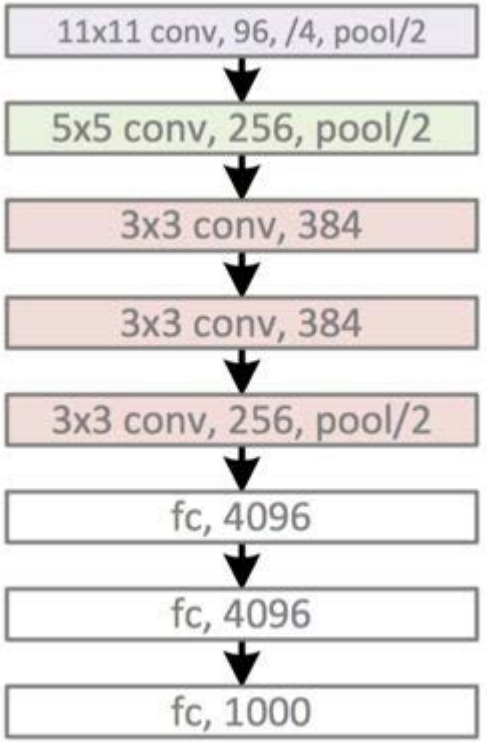
- Convolutional neural networks (CNNs)
- Representative CNN models
  - AlexNet
  - VGGNet
  - GoogleNet
  - ResNet
  - DenseNet
- CNN-based computer vision applications

AlexNet

[Krizhevsky et al., NIPS'12]

- Architecture overview
  - Five convolutional layers
  - Three fully connected layers



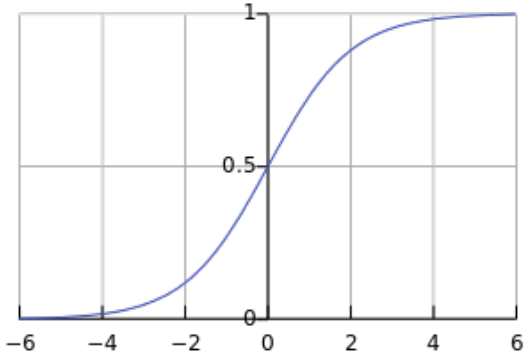


AlexNet

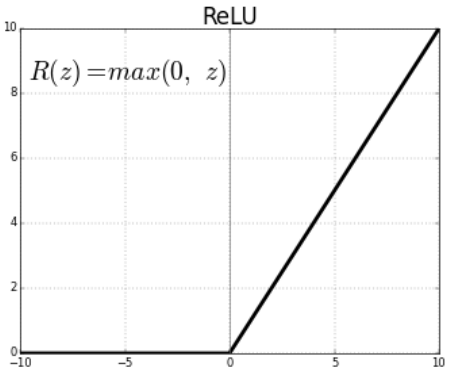
- Five major contributions:
  - Nonlinearity: ReLU
  - Multiple GPUs
  - Local response normalization (LRN)
  - Overlapping pooling
  - Reducing overfitting: data augmentation and dropout

AlexNet: ReLU and multiple GPUs

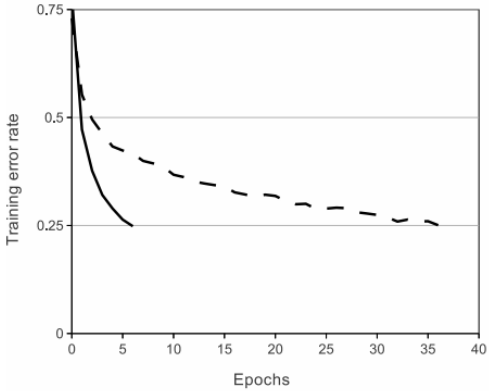
- Nonlinearity: Rectified Linear Units (ReLU)
  - Instead of tanh or sigmoid
  - Faster convergence and lower gradient vanishing



sigmoid function



ReLU



ReLU (solid line)
tanh (dashed line)

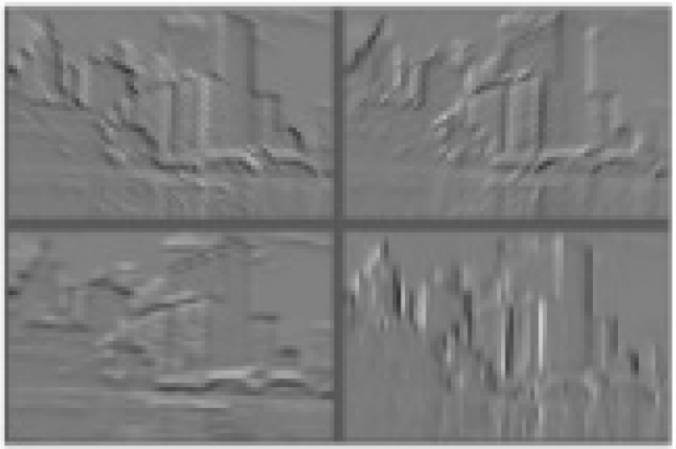
- Use multiple GPUs to speed up training

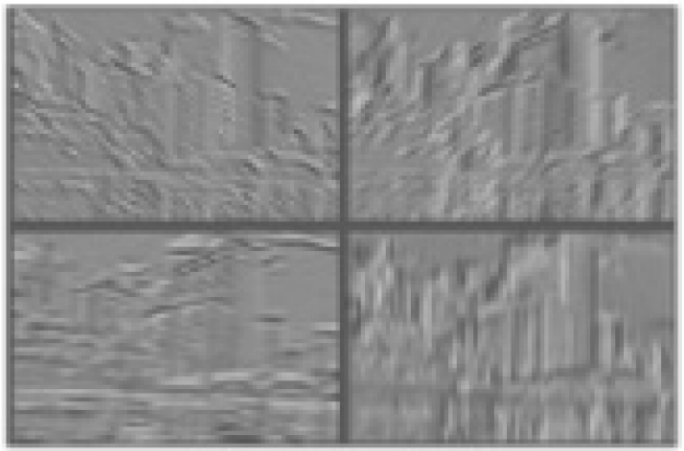
AlexNet: Local response normalization

- Local response normalization (LRN)

$$b^i_{x,y} = a^i_{x,y} / \left(k + \alpha \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} (a^j_{x,y})^2 \right)^{\beta}$$

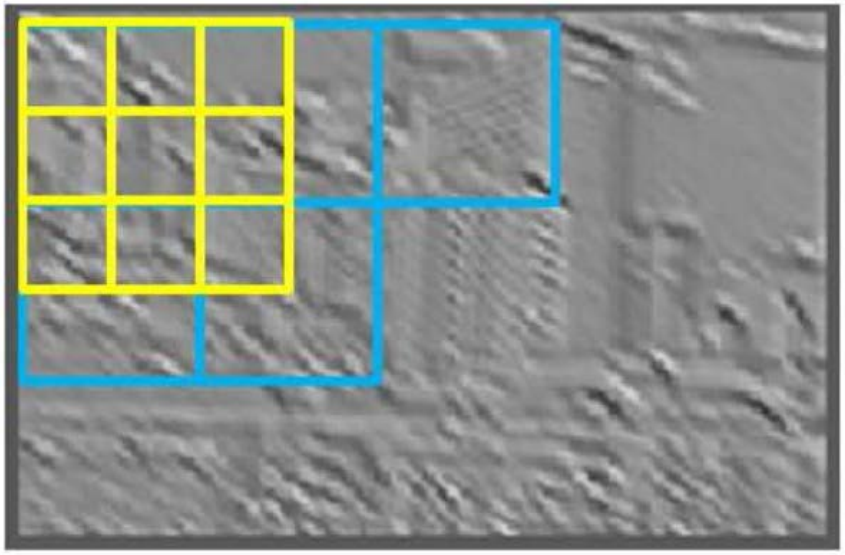
  - Reduce the top-1 and top-5 error rates by 1.4% and 1.2%





AlexNet: Overlapping pooling

- Overlapping pooling
  - Slightly alleviate the risk of overfitting
  - Reduce the top-1 and top-5 error rates by 0.4% and 0.3%

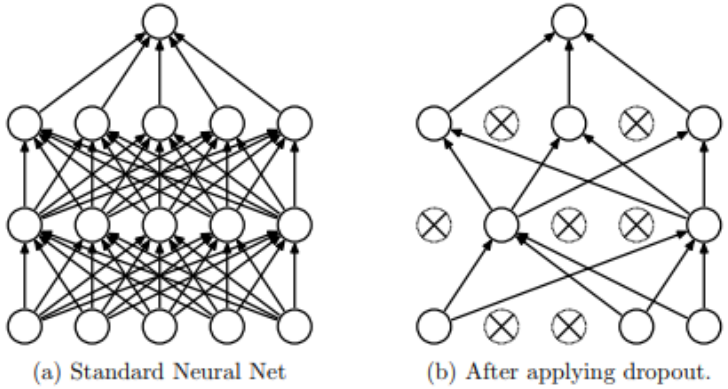


non-overlapping pooling

overlapping pooling

AlexNet: Data augmentation and dropout

- Reduce overfitting: data augmentation and dropout
- Data augmentation
  - Image translations and horizontal reflections
  - Altering the intensities of the RGB channels
- Dropout
  - Apply to fully connect layers
  - Setting the output of each hidden neuron to zero with probability 0.5 during the training stage



(a) Standard Neural Net (b) After applying dropout.

[Srivastava et al., JMLR'15]

AlexNet: Experimental results

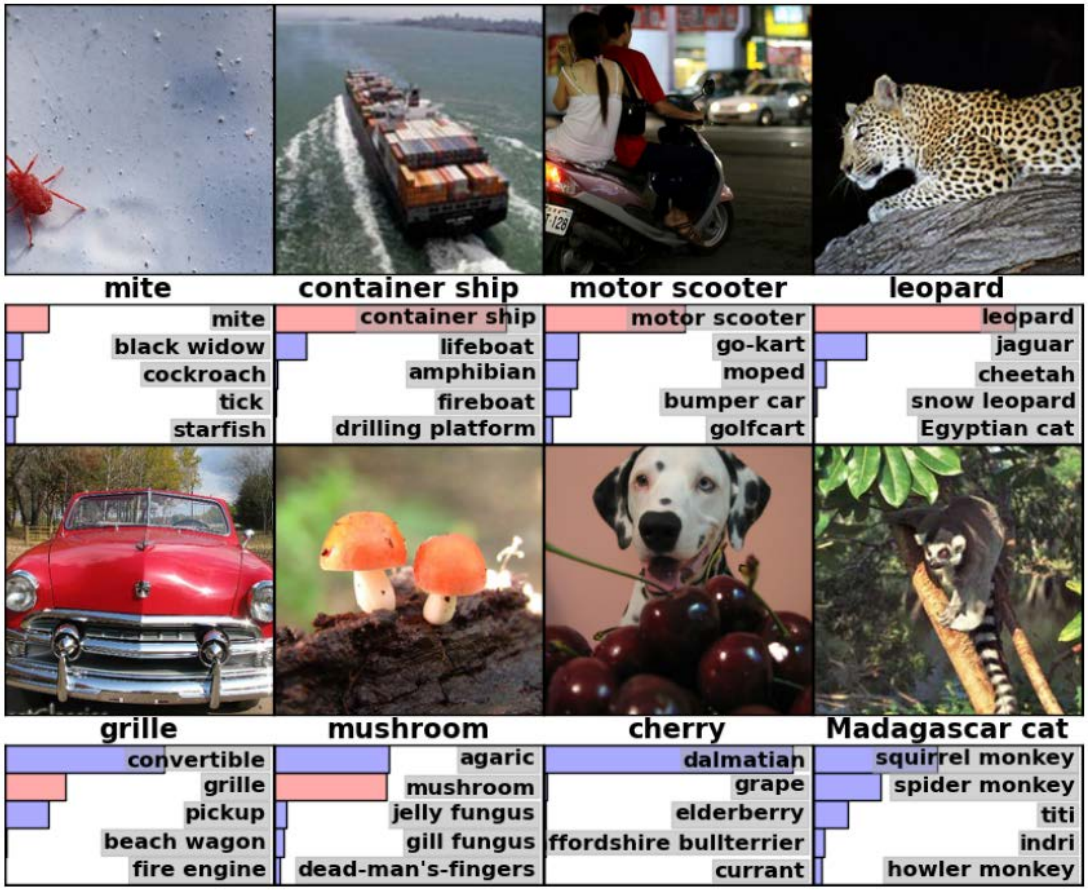
- ILSVRC-2012 competition
- Top-1 and top-5 errors

Model	Top-1 (val)	Top-5 (val)	Top-5 (test)
SIFT + FVs [7]	—	—	*26.2%*
1 CNN	40.7%	18.2%	—
5 CNNs	38.1%	16.4%	**16.4%**
1 CNN*	39.0%	16.6%	—
7 CNNs*	36.7%	15.4%	**15.3%**

- X CNNs: Average the predictions of X similar CNNs
- * denotes that models pre-trained to classify ImageNet 2011

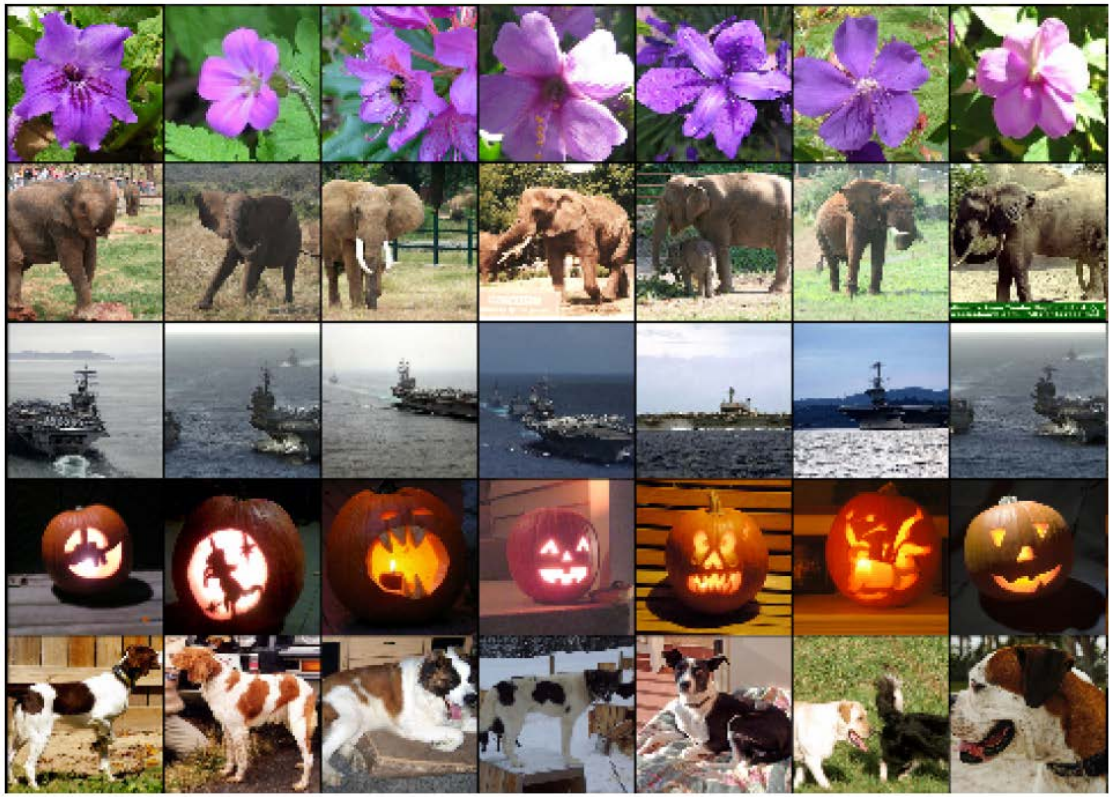
AlexNet: Experimental results

- Eight examples of top-5 prediction



AlexNet: Experimental results

- First column: Testing image
- The remaining columns: Its six nearest neighbors



Outline

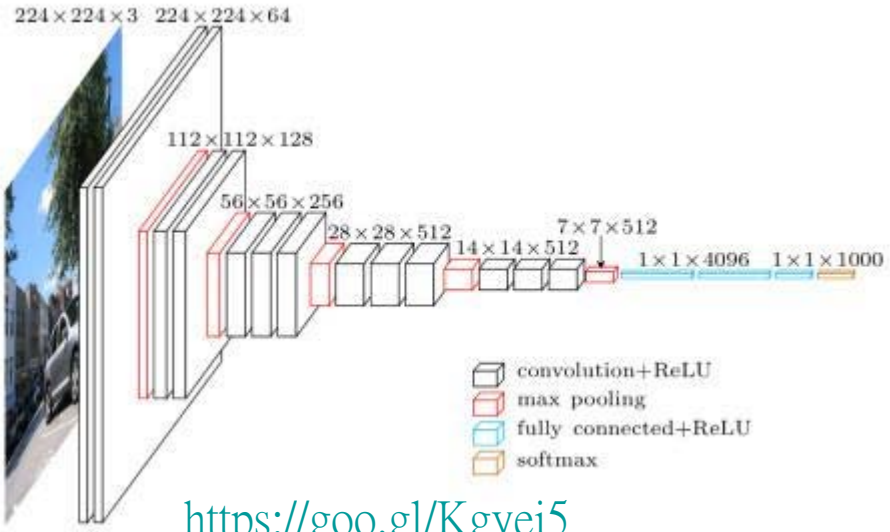
- Convolutional neural networks (CNNs)
- Representative CNN models
  - AlexNet
  - VGGNet
  - GoogleNet
  - ResNet
  - DenseNet
- CNN-based computer vision applications

VGG-Net

[Simonyan and Zisserman, ICLR'15]

- Architecture overview
  - Effect of CNN depths on accuracy
  - 16 layers or 19 layers
  - Deeper than AlexNet

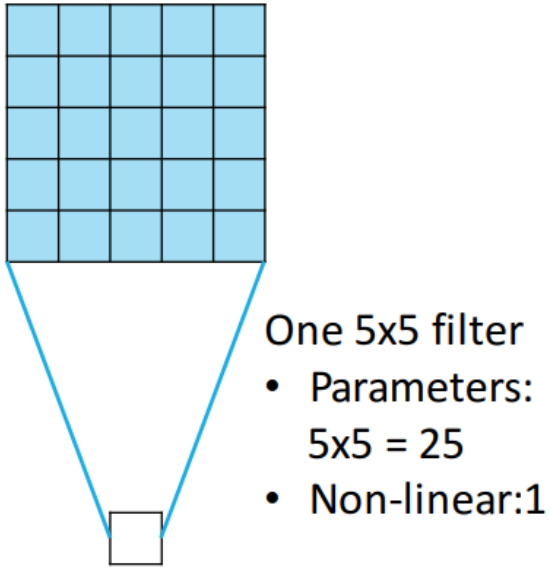
ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					



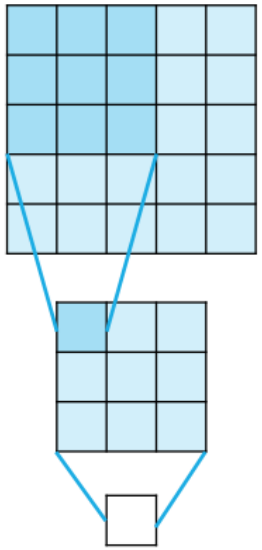
https://goo.gl/Kgyei5

VGG-Net

- 3x3 convolutional kernels – less parameters
  - Stacked convolutional layers have large receptive fields
  - More non-linearity
  - Less parameters to learn
  - More numbers of channels



One 5x5 filter
- Parameters: 5x5 = 25
- Non-linear:1



Two 3x3 filters
- Parameters: 3x3x2 = 18
- Non-linear:2

https://goo.gl/vAs3TZ

Six network configurations

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224×224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Performance analysis

ConvNet config. (Table 1)	smallest image side		top-1 val. error (%)	top-5 val. error (%)
	train (S)	test (Q)		
A	256	256	29.6	10.4
A-LRN	256	256	29.7	10.5
B	256	256	28.7	9.9
C	256	256	28.1	9.4
	384	384	28.1	9.3
	[256;512]	384	27.3	8.8
D	256	256	27.0	8.8
	384	384	26.8	8.7
	[256;512]	384	25.6	8.1
E	256	256	27.3	9.0
	384	384	26.9	8.7
	[256;512]	384	**25.5**	**8.0**

- A vs. A-LRN: LRN may not be effective
- A vs. B, C, D, E: The deeper, the better
- A vs. C: 1 x 1 convolutional layers work
- C vs. D: Spatial context is important
- Multiple scale training improves the performance

VGG-Net

- ILSVRC-2012 competition

Team	Year	Place	Error (top-5)	Uses external data	
SuperVision	2012	1st	16.4%	no	AlexNet
SuperVision	2012	1st	15.3%	Imagenet 22k	
Clarifai	2013	1st	11.7%	no	
Clarifai	2013	1st	11.2%	Imagenet 22k	
MSRA	2014	3rd	7.35%	no	
VGG	2014	2nd	7.32%	no	VGG-Net
GoogLeNet	2014	1st	6.67%	no	

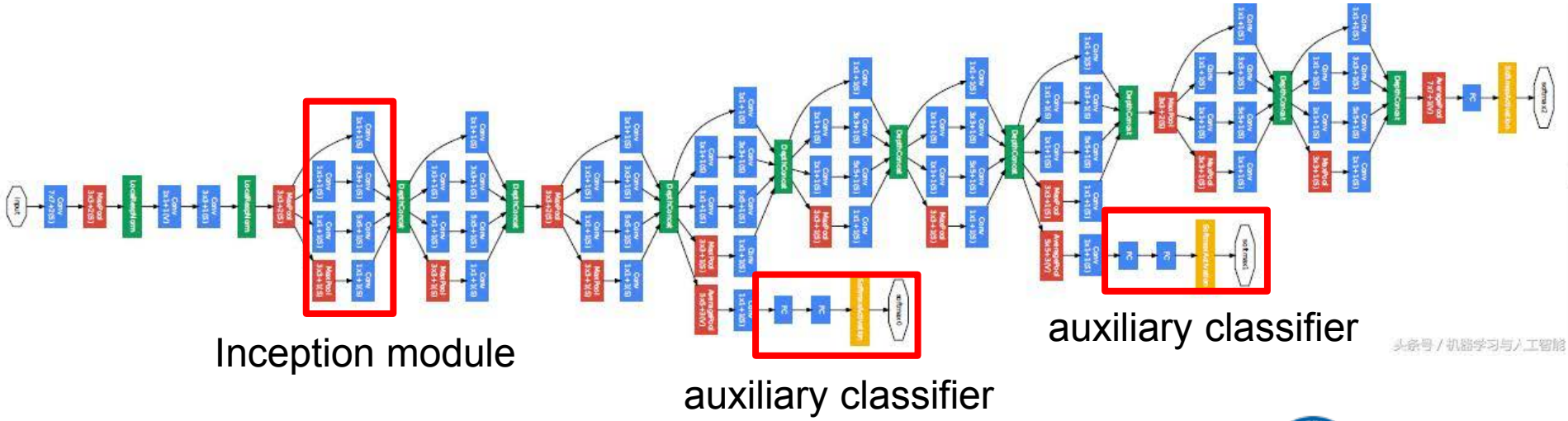
Outline

- Convolutional neural networks (CNNs)
- Representative CNN models
  - AlexNet
  - VGGNet
  - GoogleNet
  - ResNet
  - DenseNet
- CNN-based computer vision applications

GoogleNet (Inception V1)

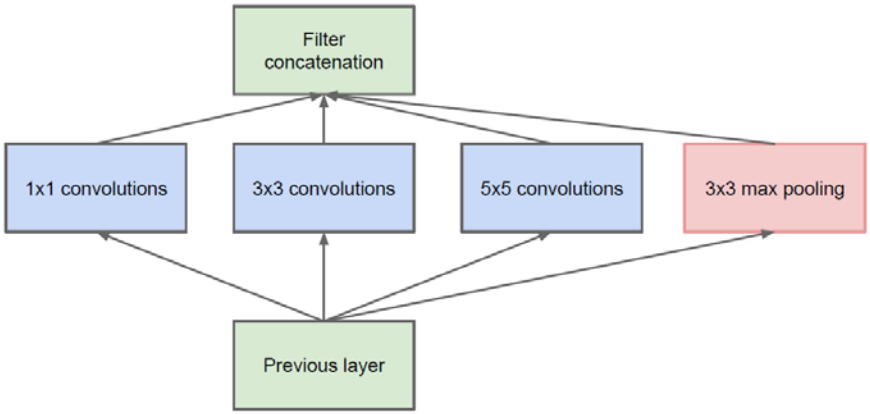
[Szegedy et al., CVPR'15]

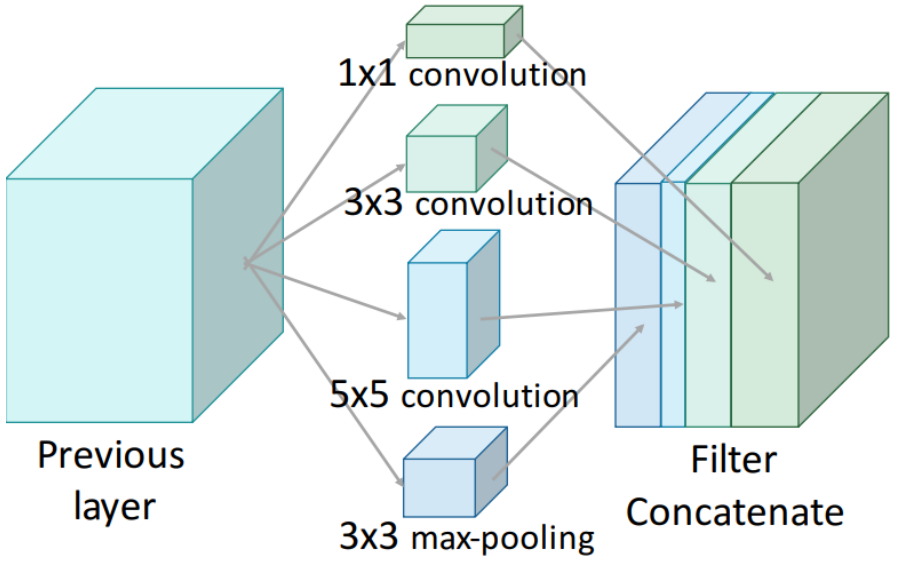
- Architecture overview
  - 22 layers
  - 12 times lesser parameters than AlexNet
  - Significantly more accurate than AlexNet
  - Inception module
  - Auxiliary classifiers



GoogleNet: Inception module

- Choose filter sizes of 1x1, 3x3, and 5x5
- Concatenate all feature maps
- Concatenate one additional pooling path, which is essential to the success of CNNs

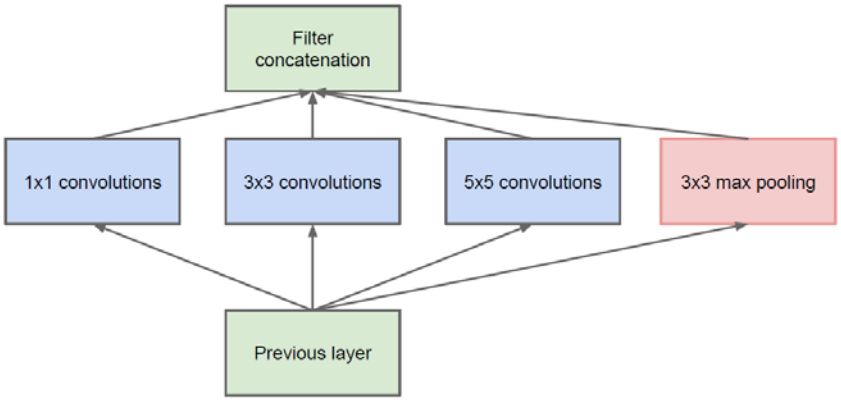


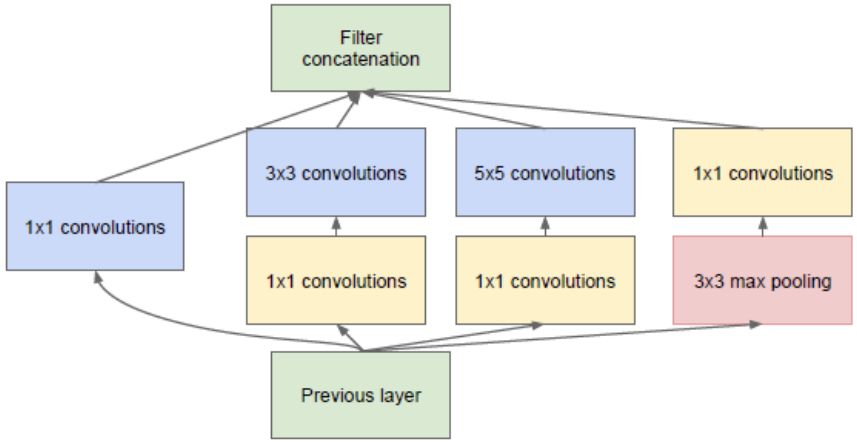


https://goo.gl/vAs3TZ

GoogleNet: Inception module

- Linear decrease in feature maps results in quadratic decrease in computation
- Use inexpensive 1x1 convolutional layers to reduce the number feature maps





GoogleNet: Inception module stacking

- Stacking inception modules
- Conventional convolutional layers in lower layers
- Max pooling for size reduction

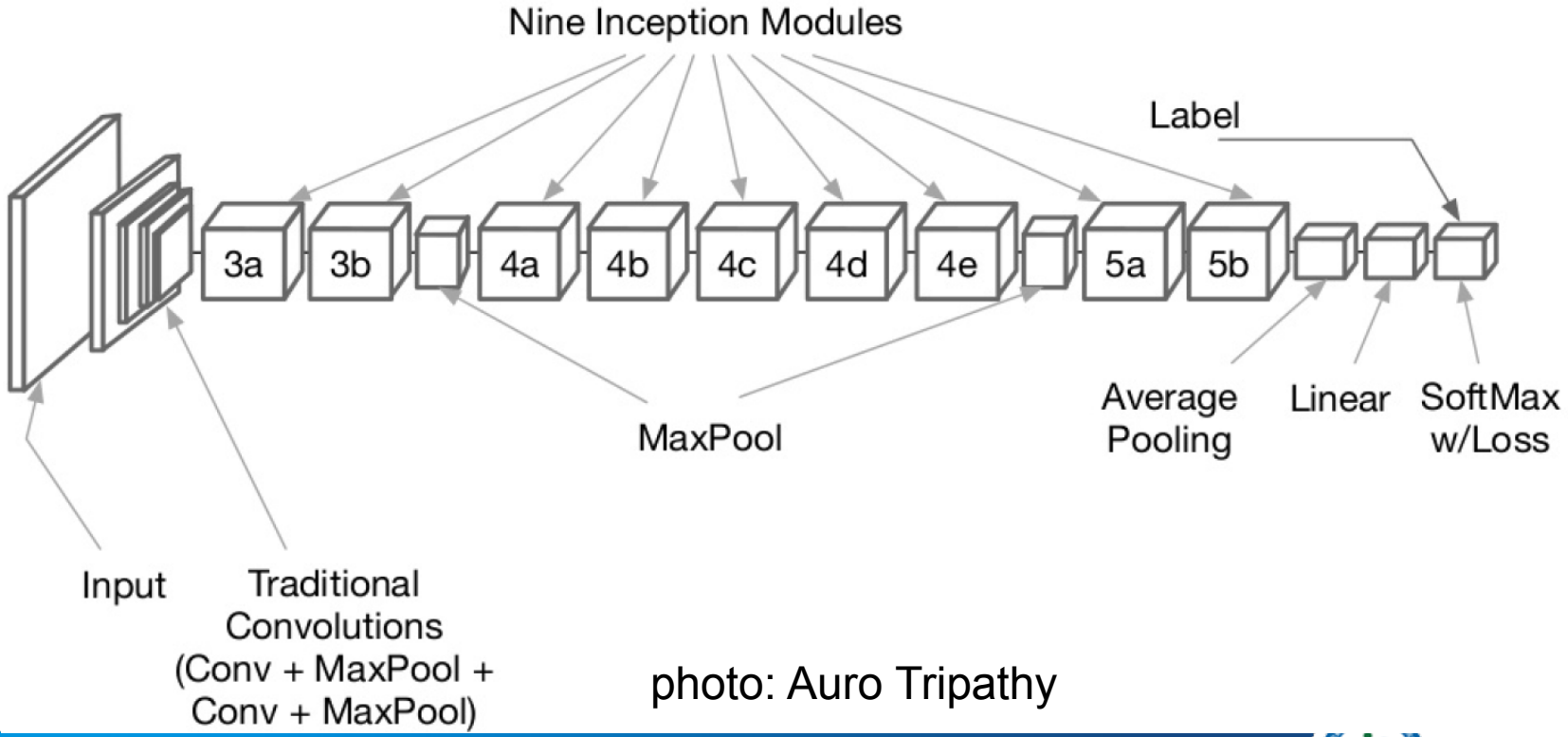


photo: Auro Tripathy

GoogleNet: Global average pooling

- Fully connected layers are prone to overfitting
- Average pooling has no parameters -> no overfitting
- Use average pooling instead of fully connected layers

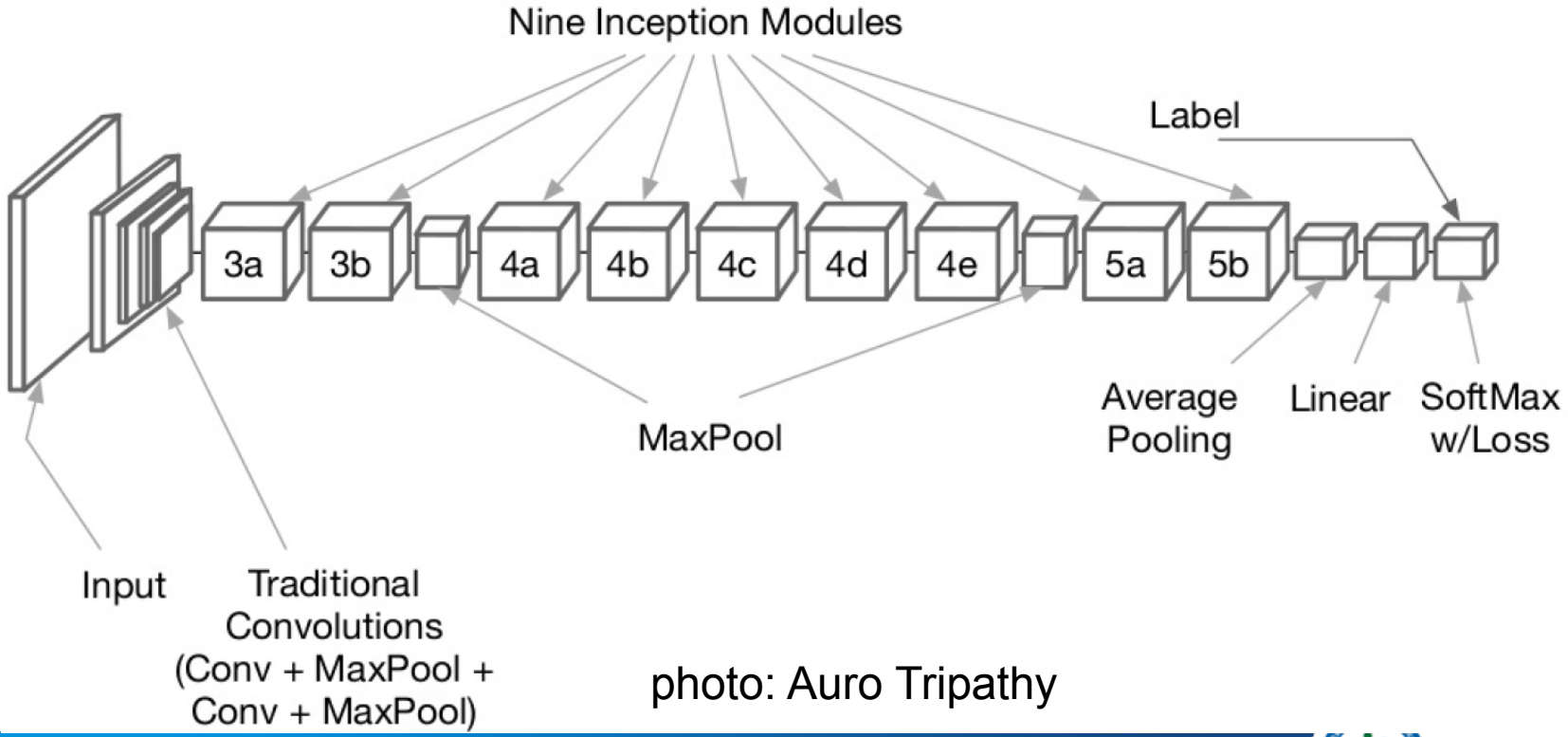


photo: Auro Tripathy

GoogleNet: Auxiliary classifier

- Deep networks result in vanishing gradients
- Auxiliary classifiers are appended

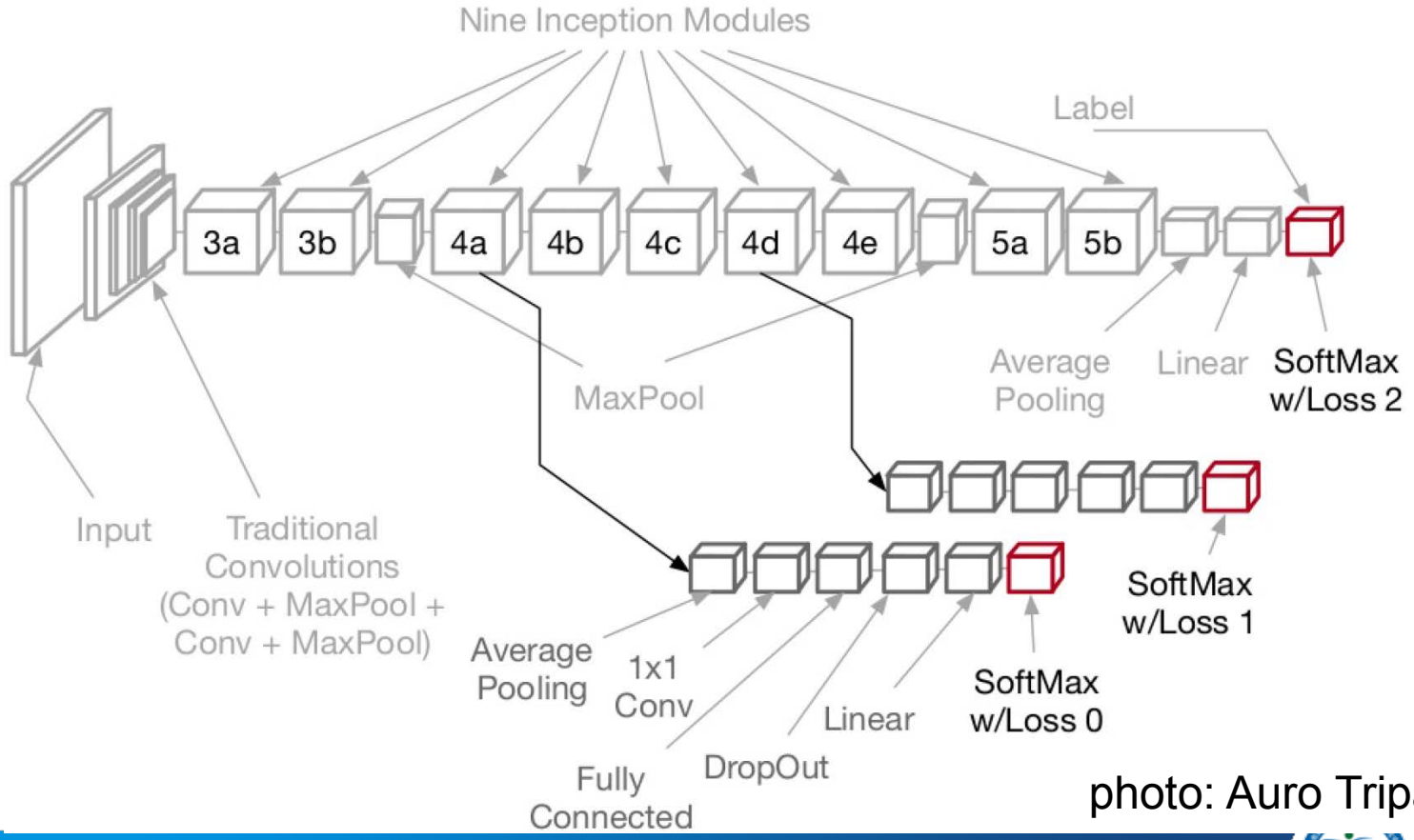


photo: Auro Tripathy

GoogleNet: Auxiliary classifier

- Intermediate layers become discriminative
- Intermediate losses alleviate the problem of vanishing graident

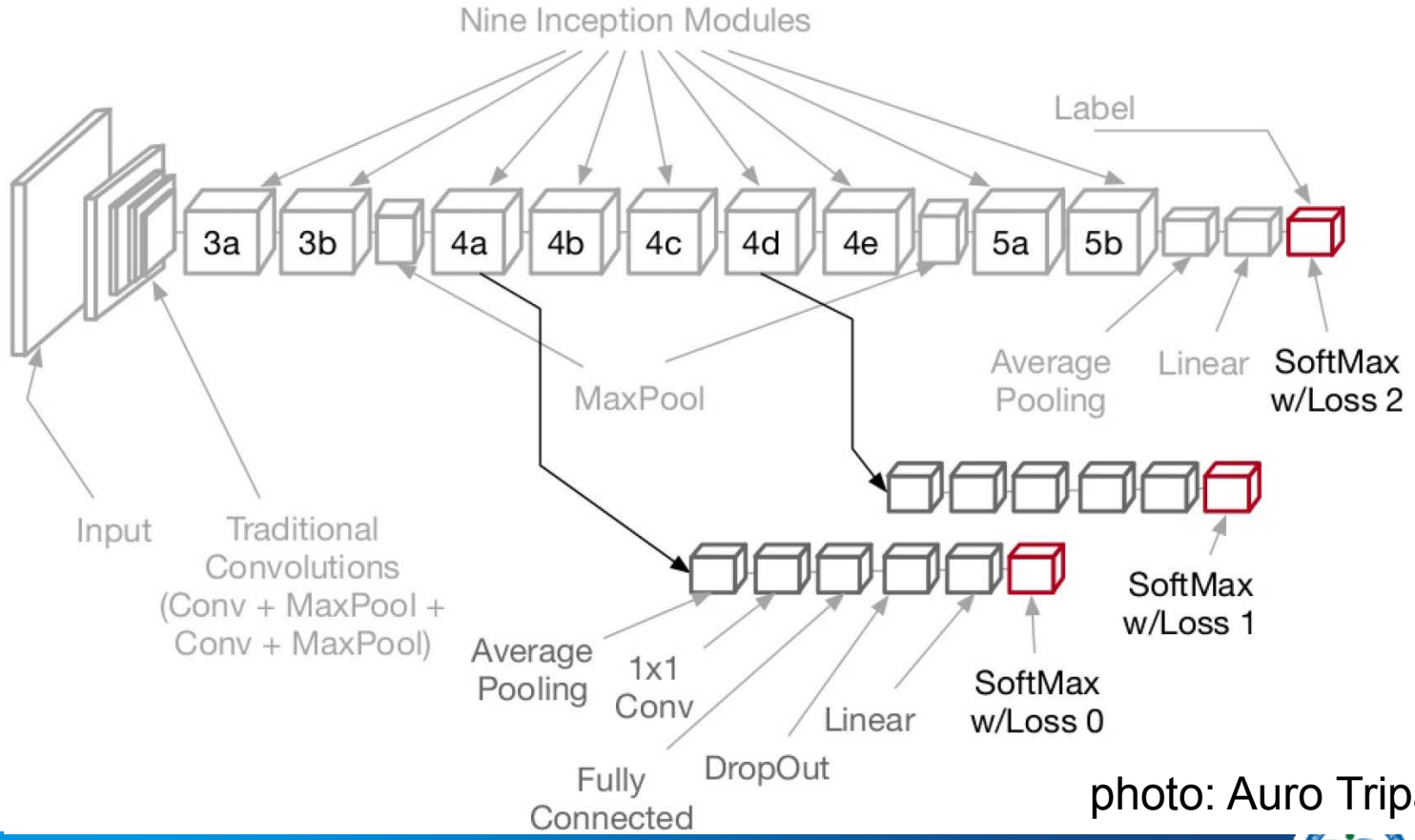


photo: Auro Tripathy

GoogleNet (Inception V1)

- ILSVRC-2012 competition

Team	Year	Place	Error (top-5)	Uses external data	
SuperVision	2012	1st	16.4%	no	AlexNet
SuperVision	2012	1st	15.3%	Imagenet 22k	
Clarifai	2013	1st	11.7%	no	
Clarifai	2013	1st	11.2%	Imagenet 22k	
MSRA	2014	3rd	7.35%	no	
VGG	2014	2nd	7.32%	no	VGG-Net
GoogLeNet	2014	1st	6.67%	no	GoogleNet

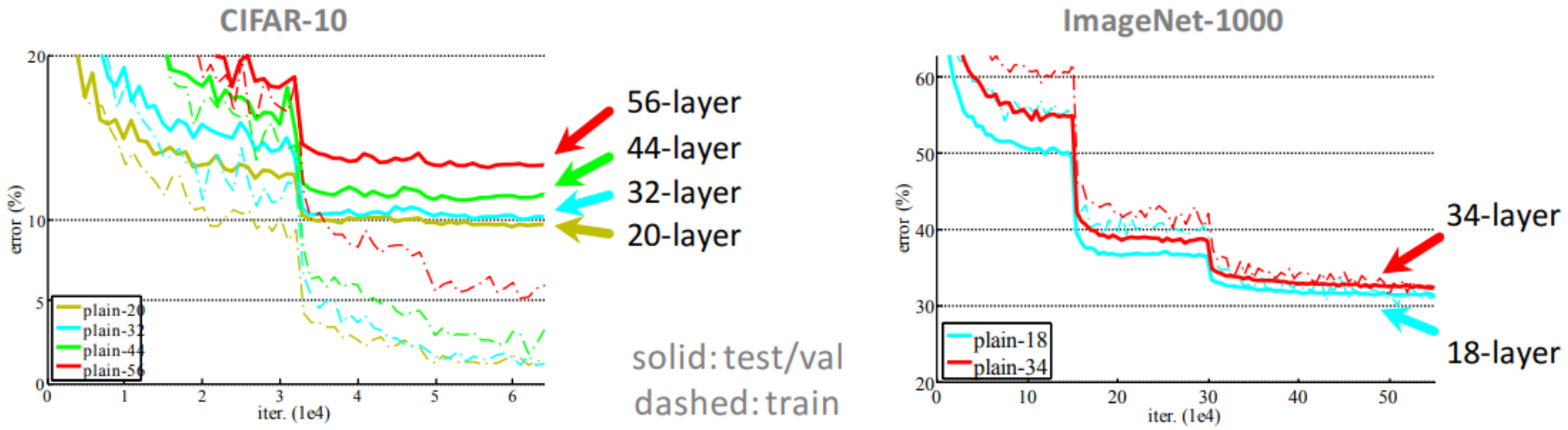
Outline

- Convolutional neural networks (CNNs)
- Representative CNN models
  - AlexNet
  - VGGNet
  - GoogleNet
  - ResNet
  - DenseNet
- CNN-based computer vision applications

ResNet

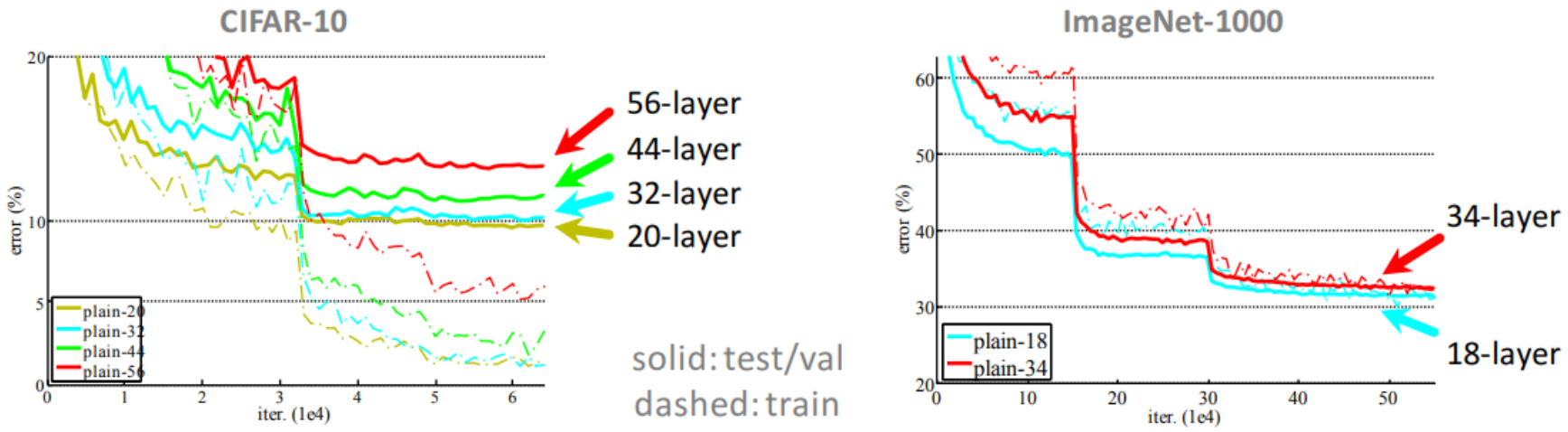
[He et al., CVPR'16]
Best Paper Award!

- The deeper, the better
  - Large receptive field size
  - High non-linearity
  - Better fitting power
- Really?
  - Performance drops when using an overly deep CNN model



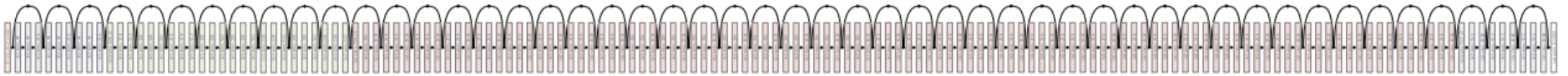
ResNet

- Overfitting?
  - ➢ No. Not only testing but also training errors increase!
- This is a general phenomenon, observed in many datasets
- It is caused by gradient exploding/vanishing



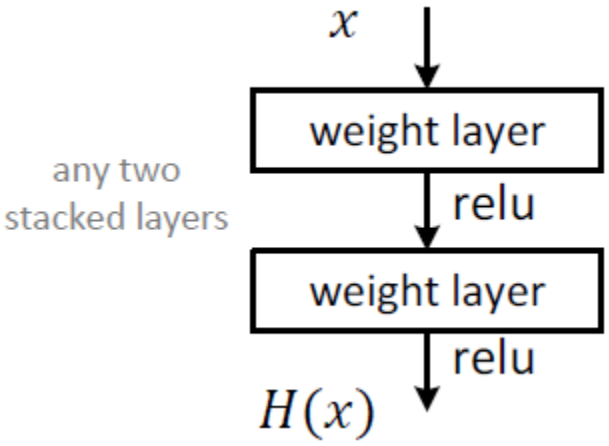
ResNet

- Architecture overview
  - 50, 101, 152 layers
  - Very deep, even more than 1,000 layers
  - It is constructed by stacking residual modules
  - Superior performances



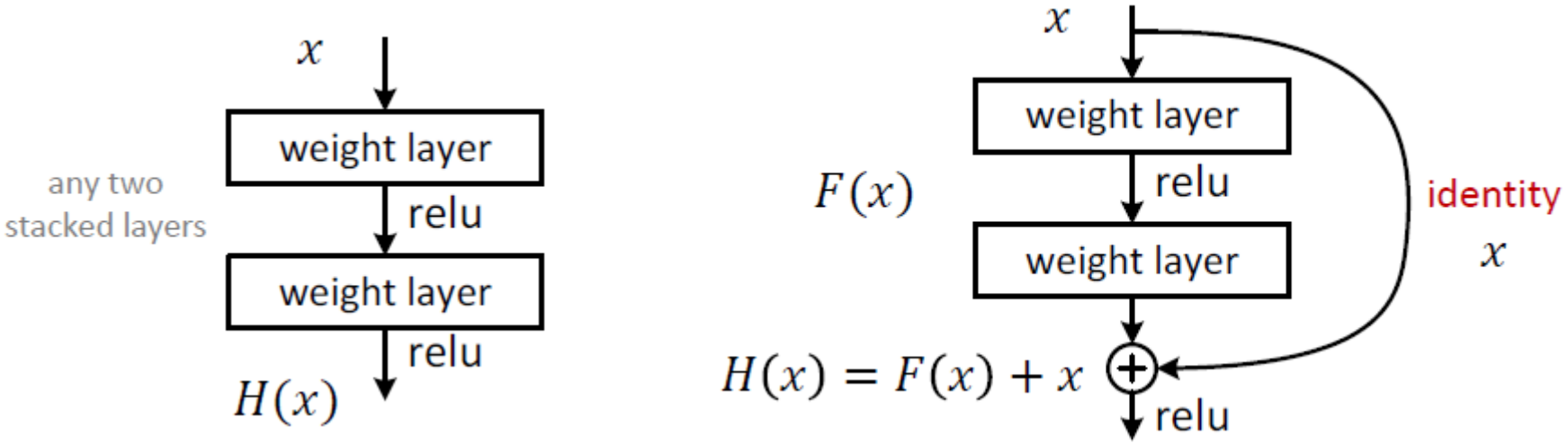
ResNet: Residual module

- The amount of changes (output - input) is fixed
- Adding more layers makes changes between two adjacent layers smaller
- Optimal mappings are close to identity
- Difficult to approximate an identity mapping due to ReLU



ResNet: Residual module

- Learn the residual F(x), instead of the desired output H(x)
- Residual: Difference between the input and the desired output
- Shortcuts connections



ResNet: Residual module

- Learn the residual F(x), instead of the desired output H(x)
- Residual: Difference between the input and the desired output
- Shortcuts connections

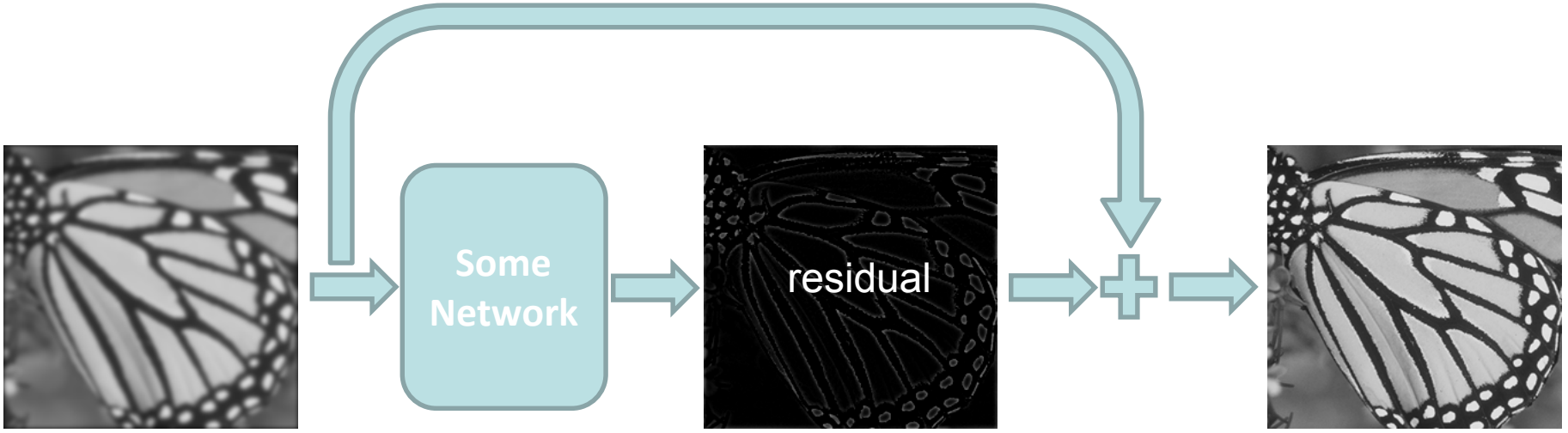


photo: H. Son

ResNet: Residual module

- A practical way to go deeper
- Inexpensive 1x1 convolutional layers for channel reduction

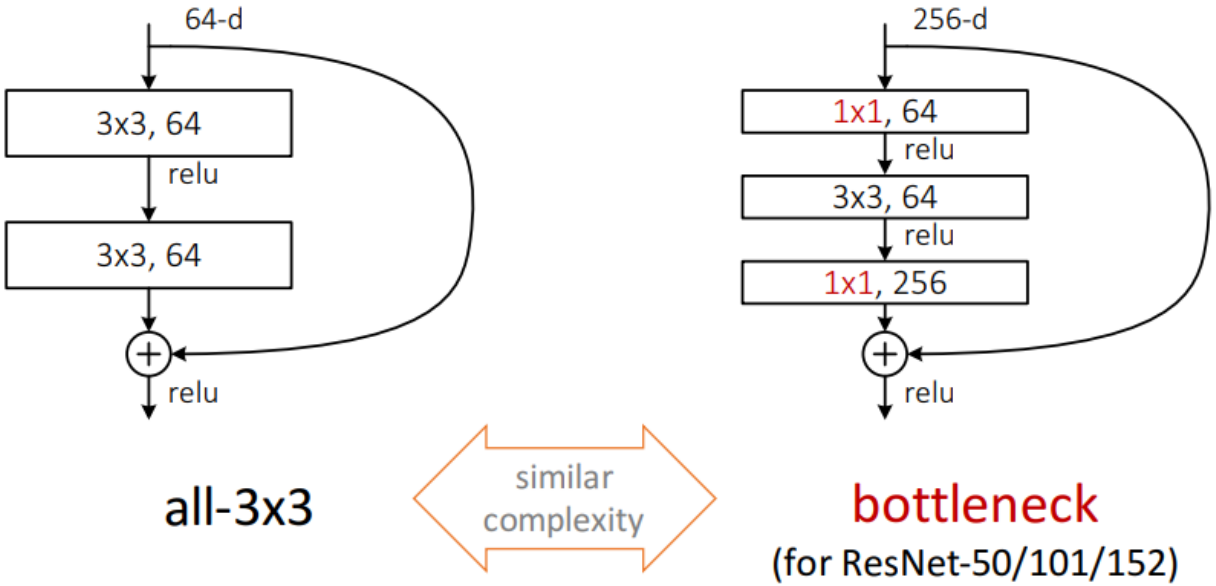
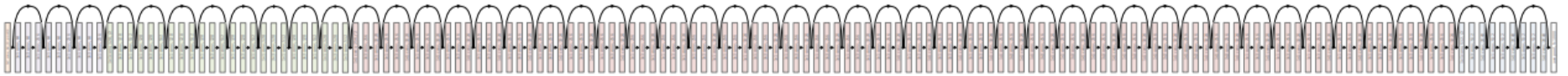


photo: H. Son

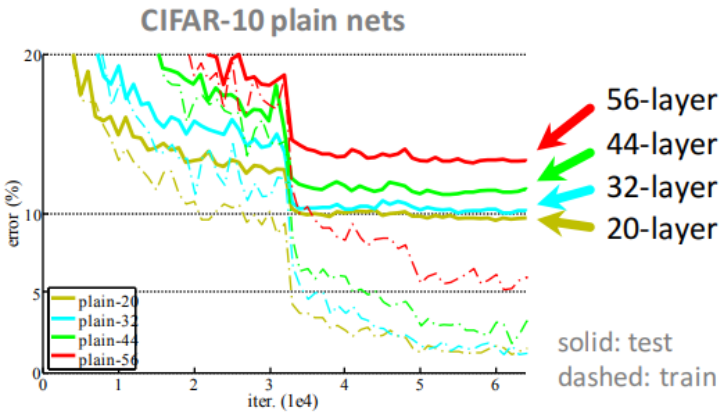
ResNet: Residual module

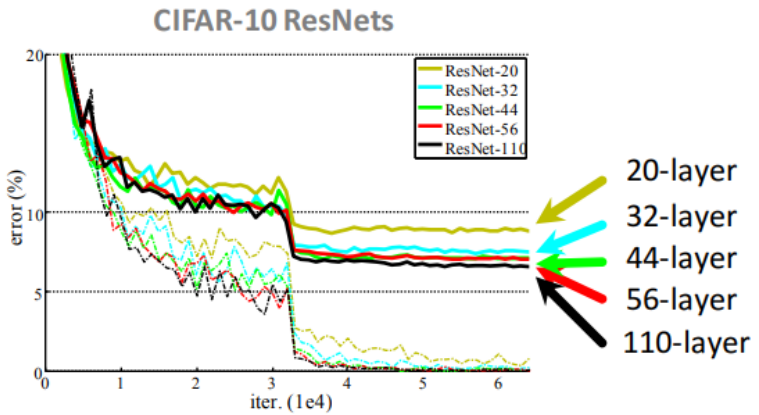
- Less parameters in a single layer
- Accelerate the training of deep networks
- Reduce the risk of vanishing gradient
- Increase depth of the network
- Achieve higher accuracy in many vision applications

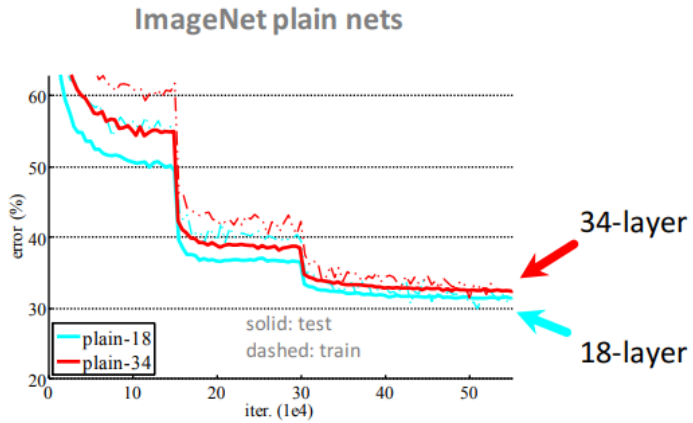


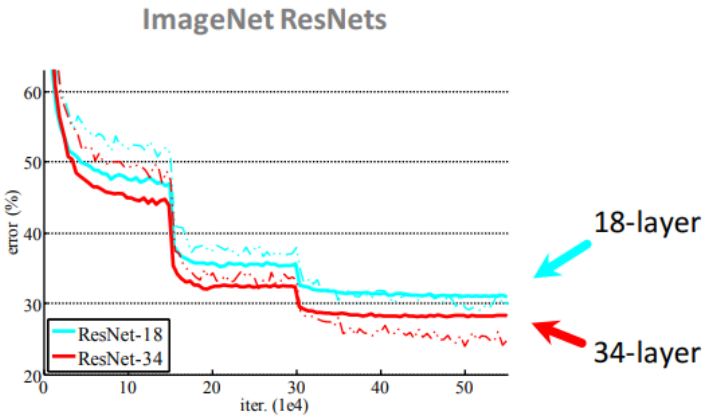
Experimental results: Convergence

- Convergence









Experimental results: Comparison with SOTA

method	top-5 err. (**test**)
VGG [41] (ILSVRC'14)	7.32
GoogLeNet [44] (ILSVRC'14)	6.66
VGG [41] (v5)	6.8
PReLU-net [13]	4.94
BN-inception [16]	4.82
ResNet (ILSVRC'15)	**3.57**

Experimental results: Depth vs. Error

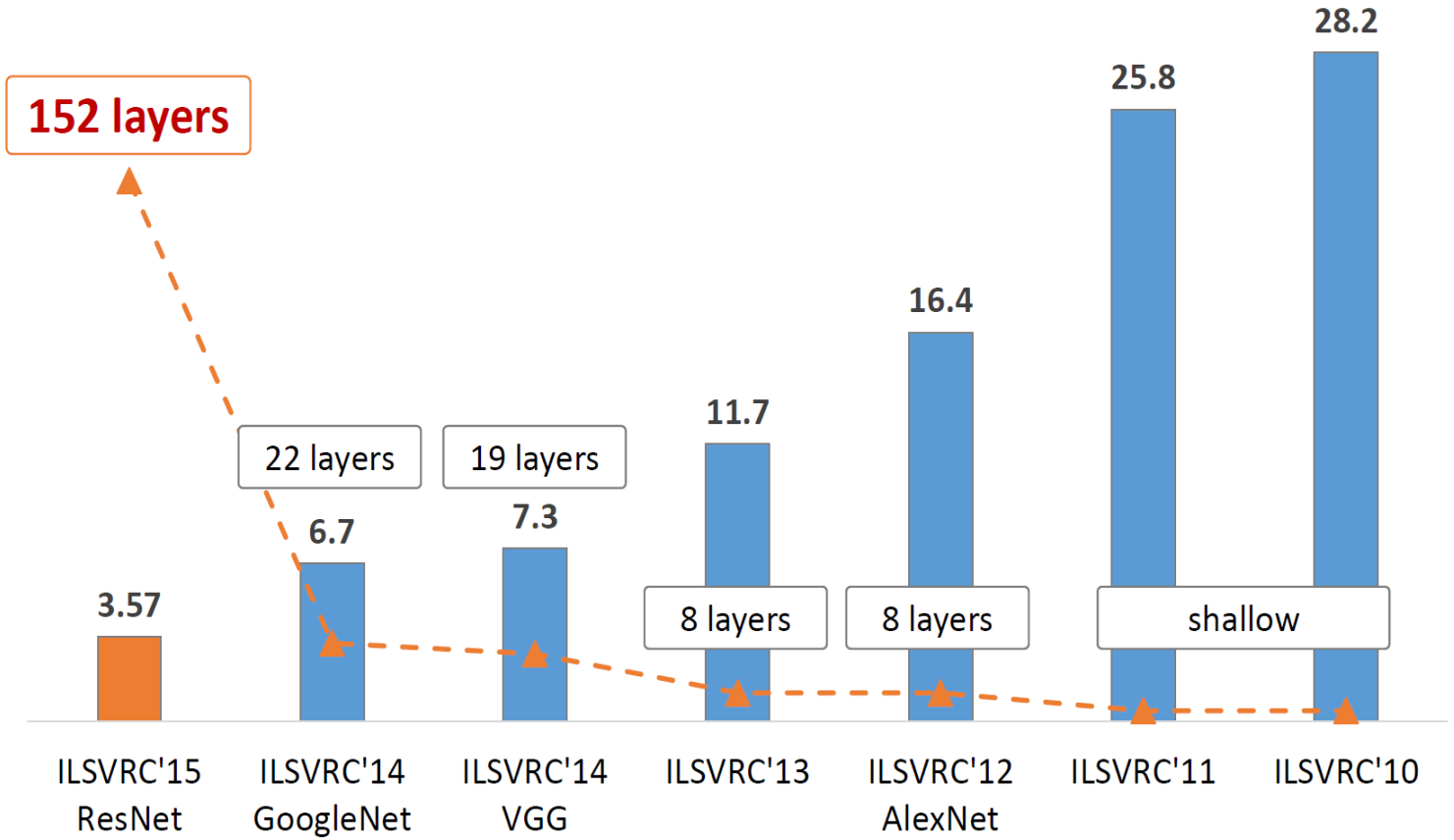


photo: He et al.

Revolution of depth

AlexNet, 8 layers
(ILSVRC 2012)

VGG, 19 layers
(ILSVRC 2014)

ResNet, 152 layers
(ILSVRC 2015)

photo: He et al.

Experimental results on more datasets and applications

- Applications: localization, detection, segmentation, …
- Datasets: ImageNet, MS COCO, …

task	2nd-place winner	MSRA	margin (relative)
ImageNet Localization (top-5 error)	12.0	9.0	**27%**
ImageNet Detection (mAP@.5)	53.6	62.1	**16%**
COCO Detection (mAP@.5:.95)	33.5	37.3	**11%**
COCO Segmentation (mAP@.5:.95)	25.1	28.2	**12%**

absolute 8.5% better!

photo: He et al.

Experimental results: Object detection

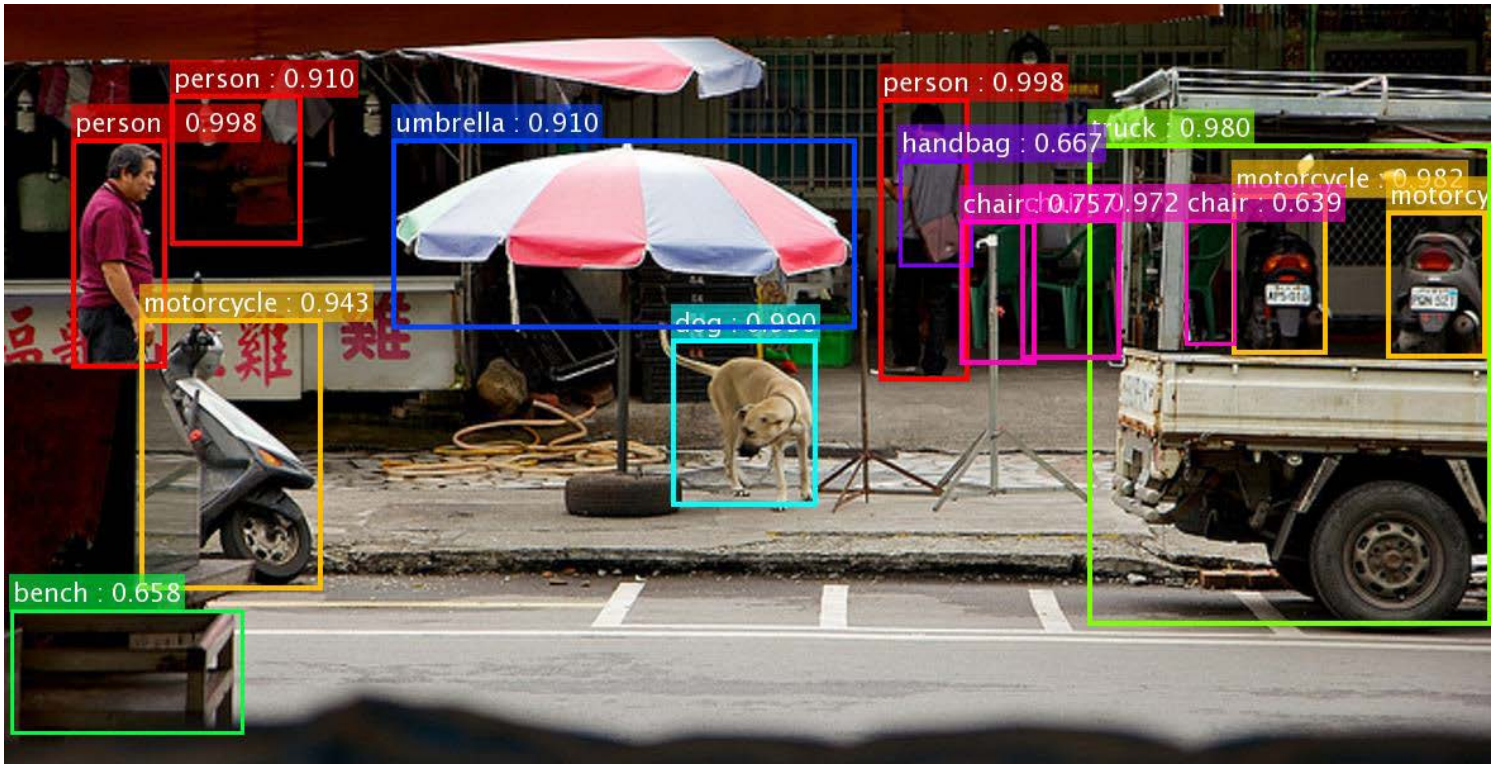


photo: He et al.

Experimental results: Instance segmentation

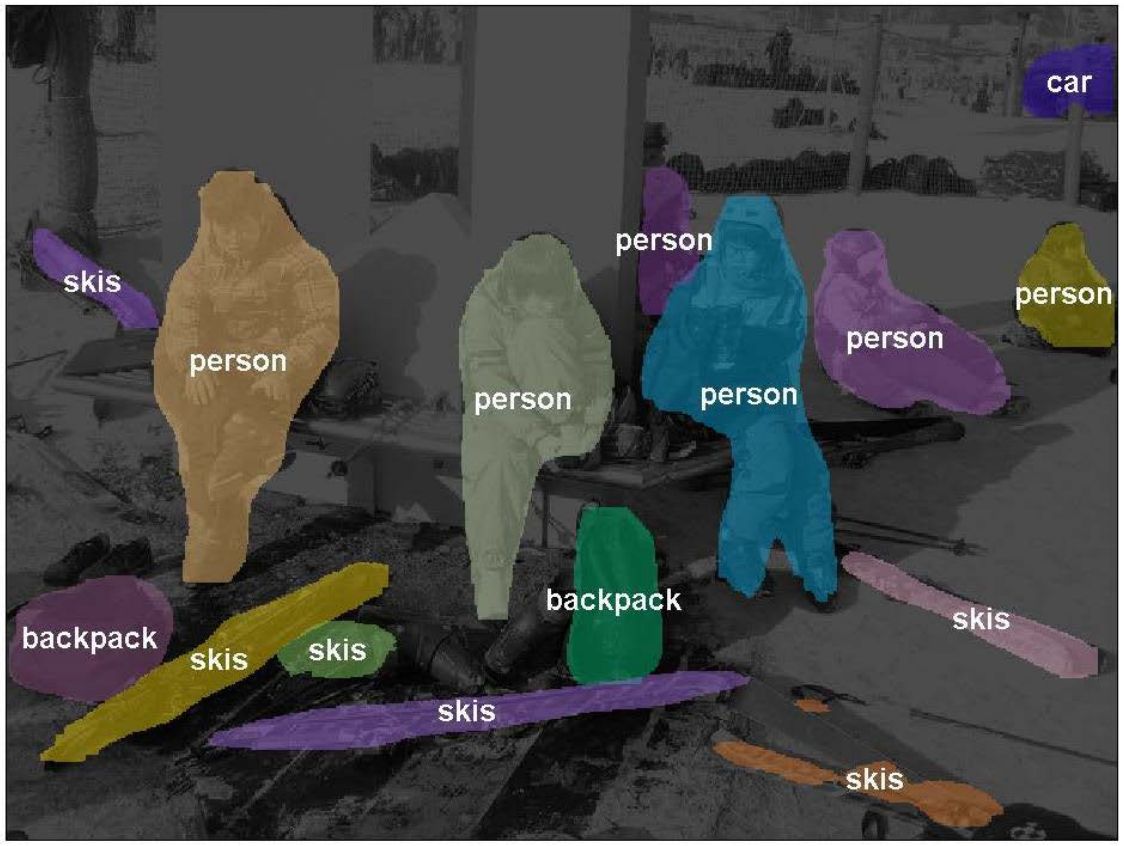


photo: He et al.

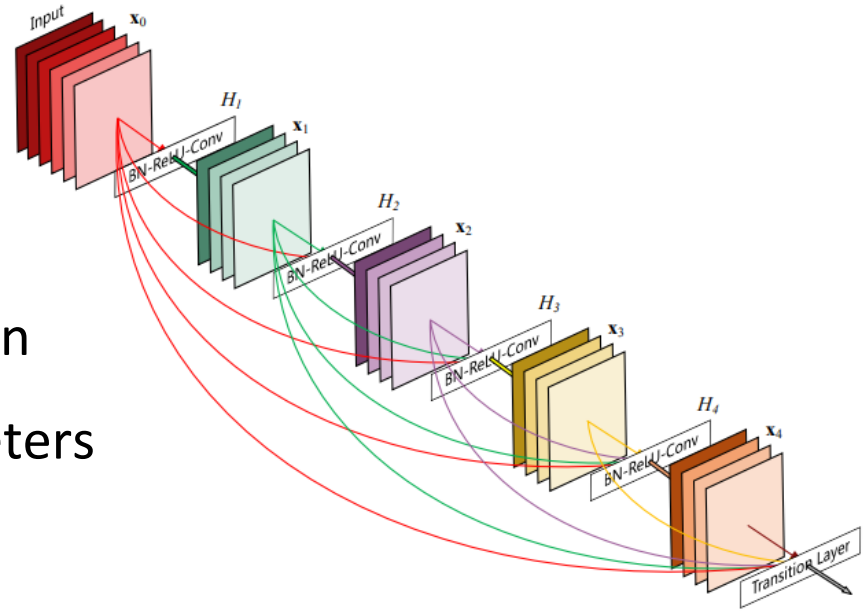
Outline

- Convolutional neural networks (CNNs)
- Representative CNN models
  - AlexNet
  - VGGNet
  - GoogleNet
  - ResNet
  - DenseNet
- CNN-based computer vision applications

DenseNet

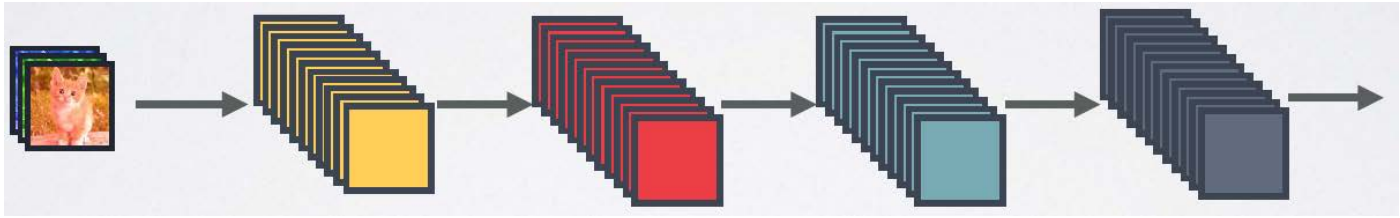
[Huang et al., CVPR'17]
Best Paper Award!

- Network architecture
- Densely connect each layer to every other layer
- For a layer, feature maps of all preceding layers are its input
- Advantages:
  - Alleviate vanishing gradient
  - Encourage feature reuse
  - Strengthen feature propagation
  - Reduce the number of parameters



CNNs vs. ResNet

- Conventional CNNs
  - A convolutional layer takes feature maps from its preceding layer



- ResNet
  - One additional path (identity mapping) with element-wise addition

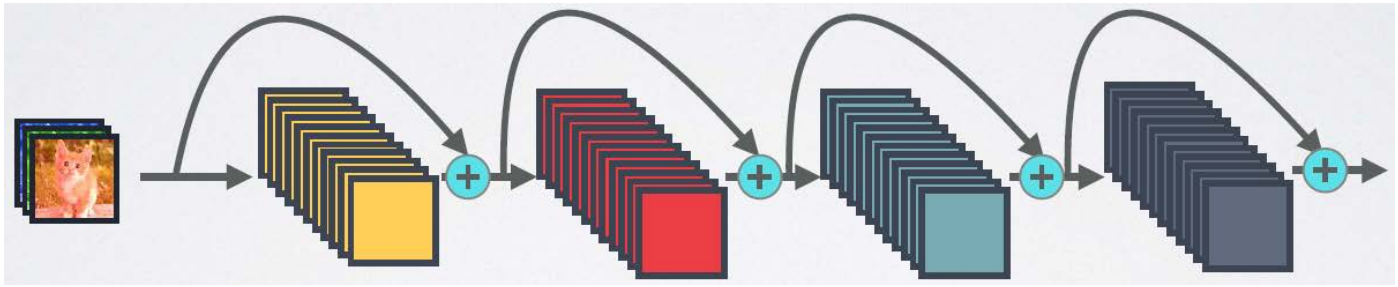


photo: Huang et al.

DenseNet

- Dense connectivity: a link between every layer pair
  - Each layer takes all preceding feature maps as input by channel-wise concatenation
  - Reduce the risk of gradient vanishing
  - Feature reuse
  - Fewer filters per layer
  - High computation cost and more parameters per filter?

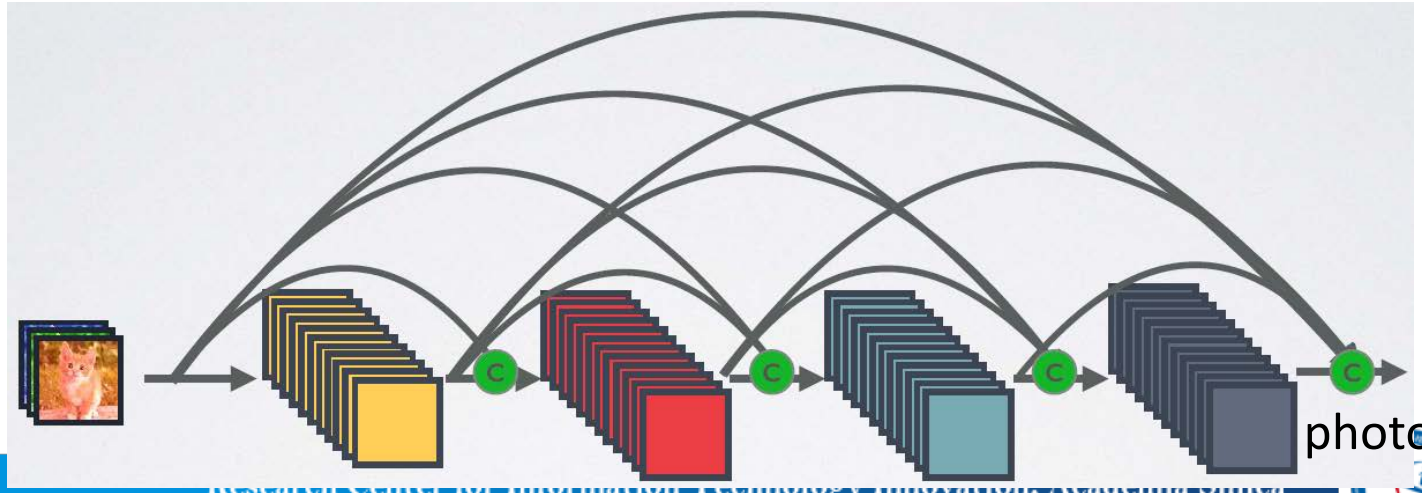


photo: Huang et al.

DenseNet

- 1x1 convolutional layer for channel (feature map) reduction
- Suppose each layer produces *k* feature maps
- There will be *l*x*k* channels at the *l*th convolutional layer
- Reduce to *4*x*k* channels before producing its *k* feature maps
- No pooling for map size reduction?

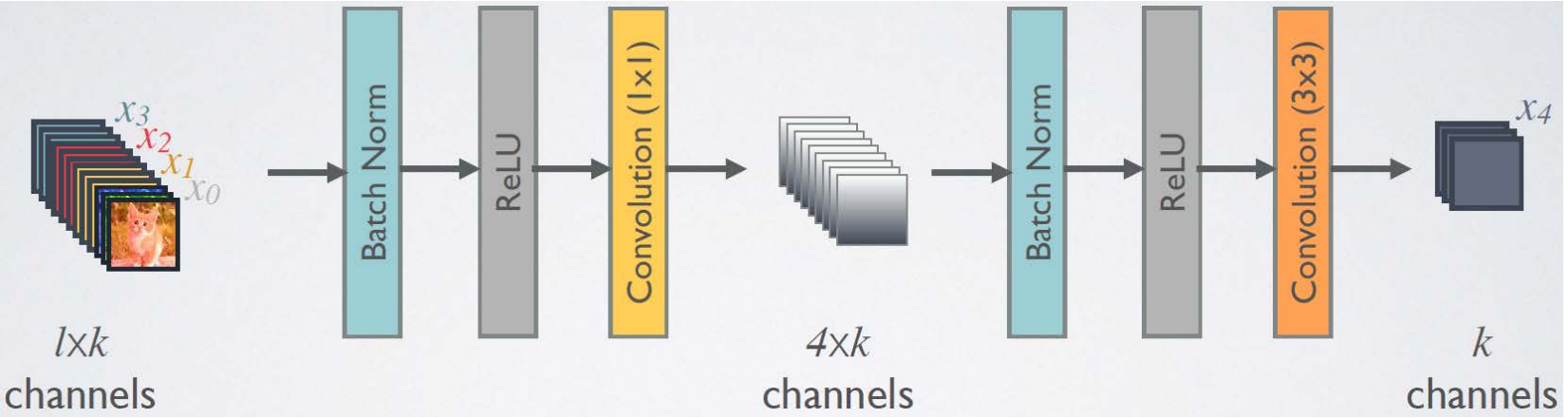


photo: Huang et al.

DenseNet

- Dense block: a few densely connected convolutional layers
- Stack dense blocks
- Insert a convolutional layer and a pooling layer into two connected blocks

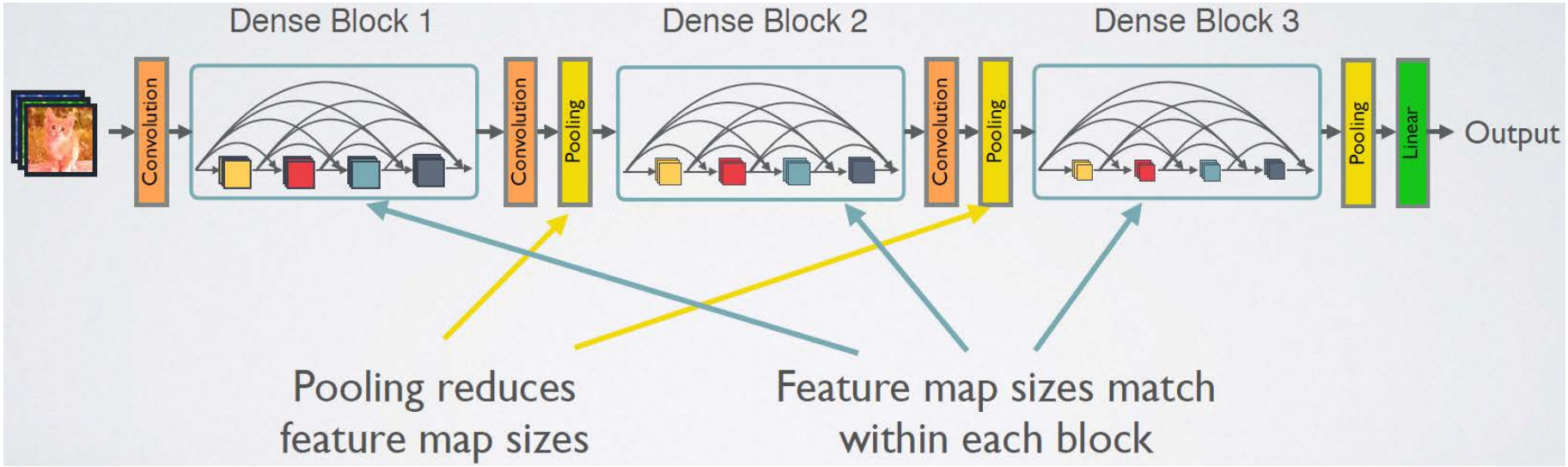


photo: Huang et al.

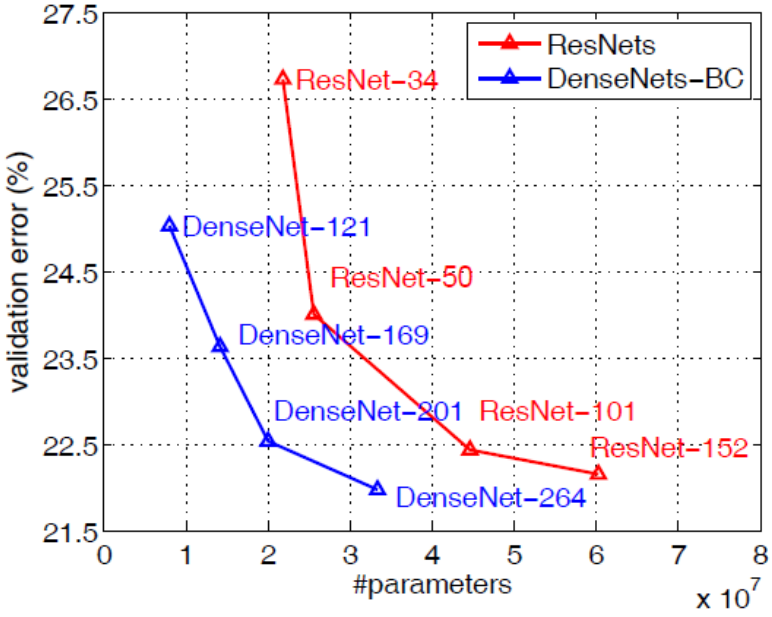
Experimental results: Comparison with SOTA

- ILSVRC, top-5 error

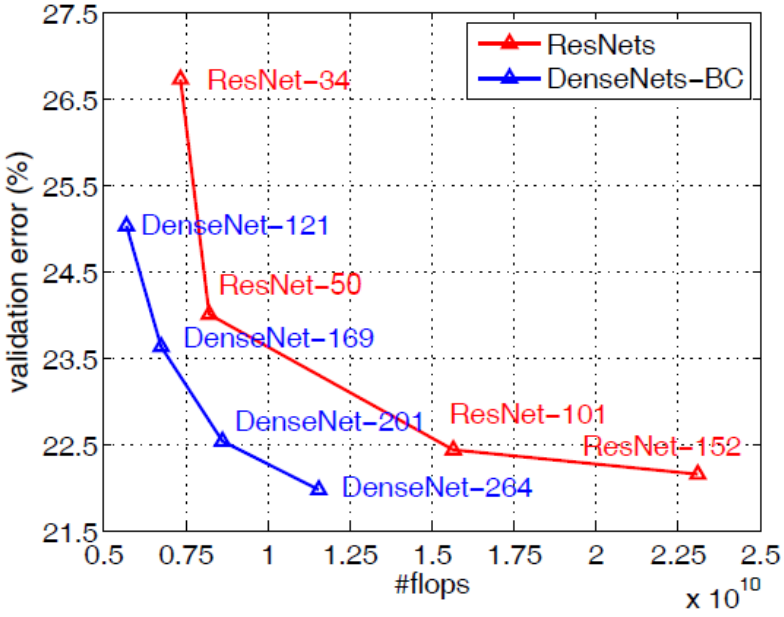
Method	Depth	Params	C10	C10+	C100	C100+	SVHN
Network in Network [22]	-	-	10.41	8.81	35.68	-	2.35
All-CNN [32]	-	-	9.08	7.25	-	33.71	-
Deeply Supervised Net [20]	-	-	9.69	7.97	-	34.57	1.92
Highway Network [34]	-	-	-	7.72	-	32.39	-
FractalNet [17]	21	38.6M	10.18	5.22	35.34	23.30	2.01
with Dropout/Drop-path	21	38.6M	7.33	4.60	28.20	23.73	1.87
ResNet [11]	110	1.7M	-	6.61	-	-	-
ResNet (reported by [13])	110	1.7M	13.63	6.41	44.74	27.22	2.01
ResNet with Stochastic Depth [13]	110	1.7M	11.66	5.23	37.80	24.58	1.75
	1202	10.2M	-	4.91	-	-	-
Wide ResNet [42]	16	11.0M	-	4.81	-	22.07	-
	28	36.5M	-	4.17	-	20.50	-
with Dropout	16	2.7M	-	-	-	-	1.64
ResNet (pre-activation) [12]	164	1.7M	11.26*	5.46	35.58*	24.33	-
	1001	10.2M	10.56*	4.62	33.47*	22.71	-
DenseNet ($k = 12$)	40	1.0M	**7.00**	5.24	**27.55**	24.42	1.79
DenseNet ($k = 12$)	100	7.0M	**5.77**	**4.10**	**23.79**	**20.20**	1.67
DenseNet ($k = 24$)	100	27.2M	**5.83**	**3.74**	**23.42**	**19.25**	**1.59**
DenseNet-BC ($k = 12$)	100	0.8M	**5.92**	4.51	**24.15**	22.27	1.76
DenseNet-BC ($k = 24$)	250	15.3M	**5.19**	**3.62**	**19.64**	**17.60**	1.74
DenseNet-BC ($k = 40$)	190	25.6M	-	**3.46**	-	**17.18**	-

+: data augmentation, *: run by the authors

Experimental results: Comparison with ResNet



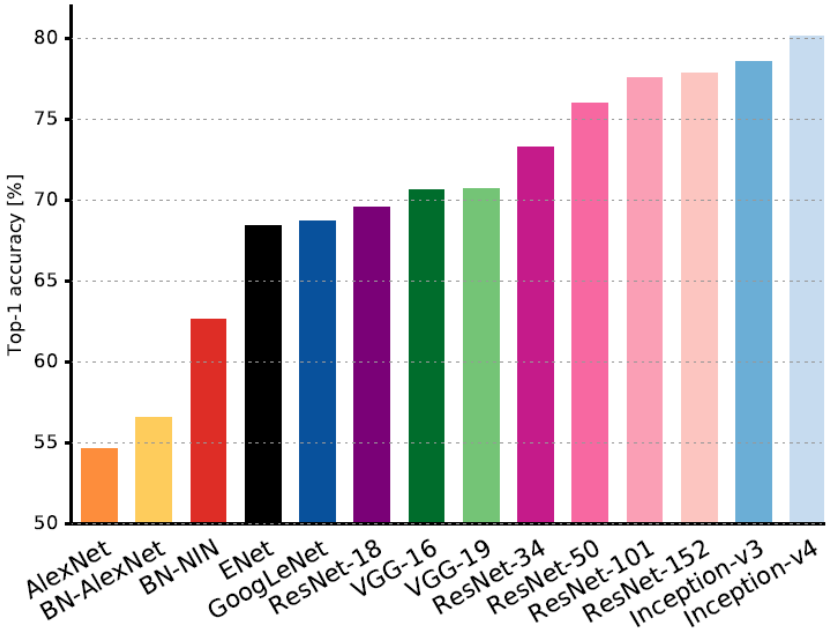
error vs. #parameters



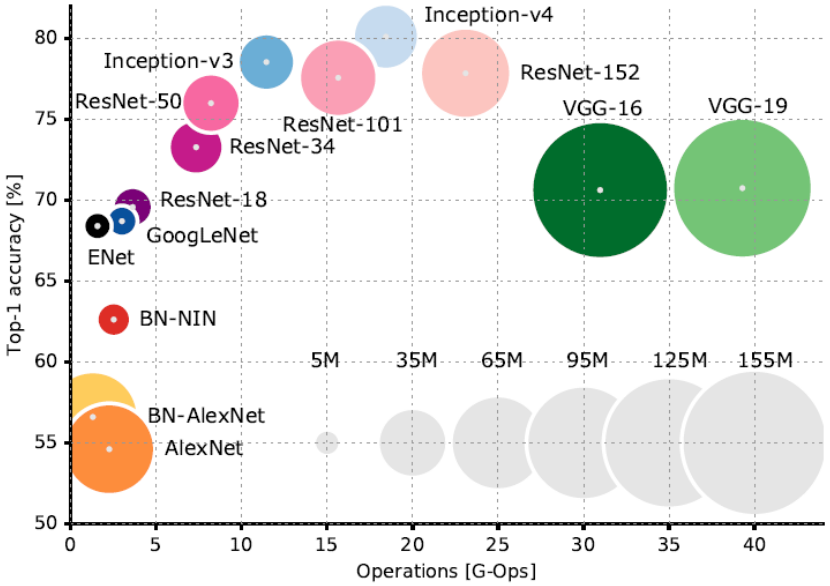
error vs. #flops

Performance comparison

[Canziani et al., arXiv'17]

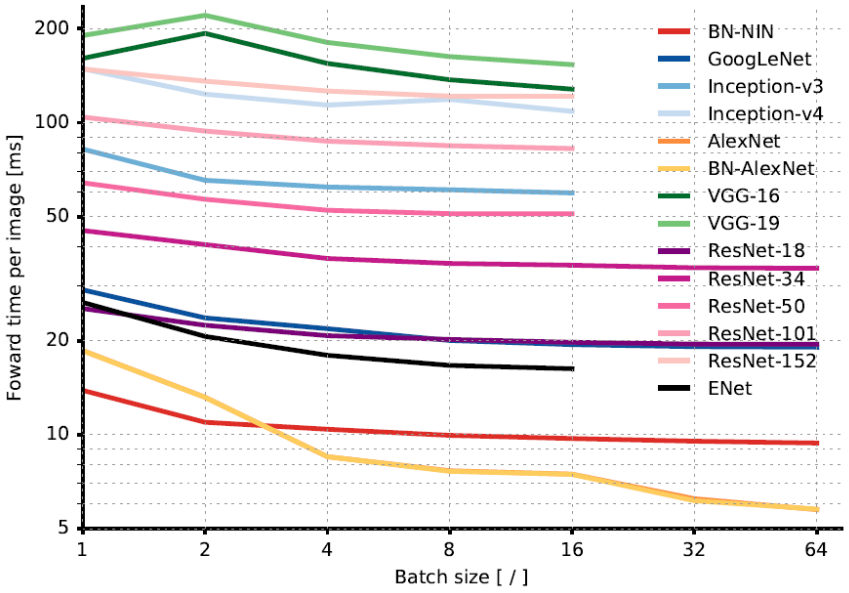


top-1 acc. vs. network

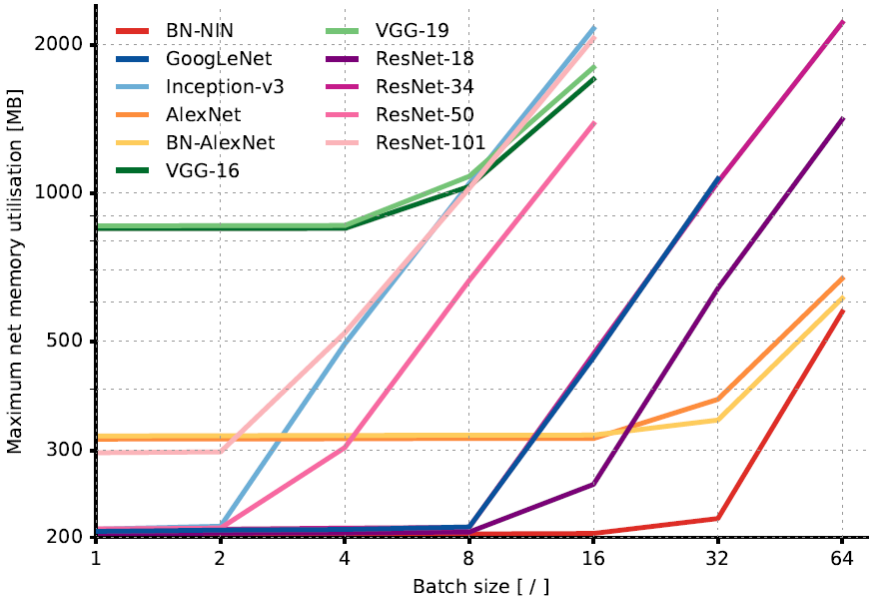


top-1 acc. vs. #operations with #parameters

Performance comparison



inference time vs. batch size



memory size vs. batch size

Performance comparison

模型名	AlexNet	VGG	GoogLeNet	ResNet
初入江湖	2012	2014	2014	2015
层数	8	19	22	152
Top-5错误	16.4%	7.3%	6.7%	3.57%
Data Augmentation	+	+	+	+
Inception(NIN)	–	–	+	–
卷积层数	5	16	21	151
卷积核大小	11,5,3	3	7,1,3,5	7,1,3,5
全连接层数	3	3	1	1
全连接层大小	4096,4096,1000	4096,4096,1000	1000	1000
Dropout	+	+	+	+
Local Response Normalization	+	–	+	–
Batch Normalization	–	–	–	+

https://goo.gl/sKfjPj

Outline

- Convolutional neural networks (CNNs)
- Representative CNN models
  - AlexNet
  - VGGNet
  - GoogleNet
  - ResNet
  - DenseNet
- CNN-based computer vision applications

Outline

- Convolutional neural networks (CNNs)
- Representative CNN models
- CNN-based computer vision applications
  - Find-grained object recognition
  - Object detection
  - Semantic segmentation
  - Image super resolution
  - Saliency detection
  - Image style transfer
  - Action and gesture recognition
  - Image matching and alignment

Introduction

[Shih et al., CVPR'17]

- Generic and fine-grained visual recognition
  - A large class number
  - Large intra-class variations
  - Subtle inter-class variations

做物件辨識就會碰到

fine-grained 才會碰到

不同類的東西長得可能很像

Caltech-UCSD Birds-200-2011

Crested Auklet

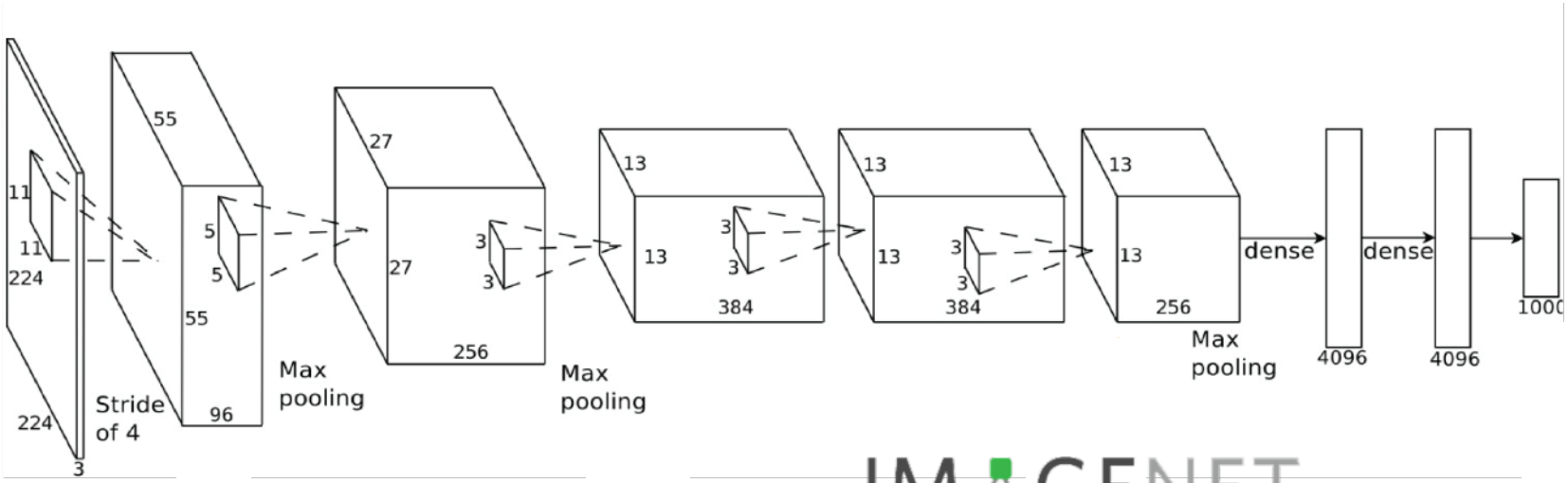
Least Auklet

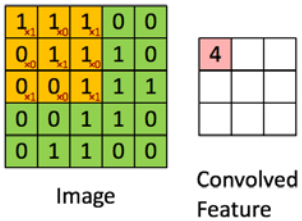
Rhinoceros Auklet

Discriminative Parts!

Convolutional Neural Networks

Most state-of-the-art recognition methods are developed upon CNNs.





http://ufldl.stanford.edu/wiki/images/6/6c/Convolution_schematic.gif

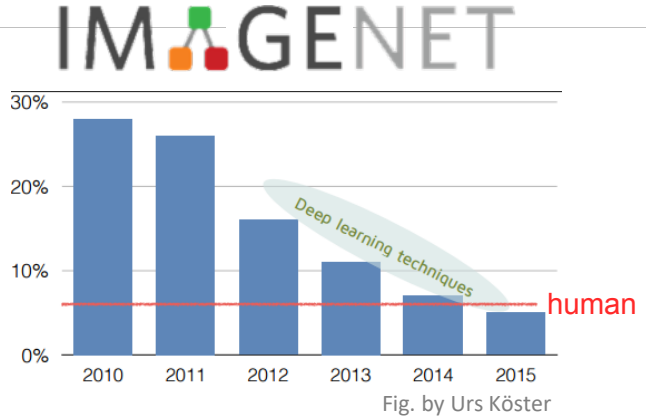
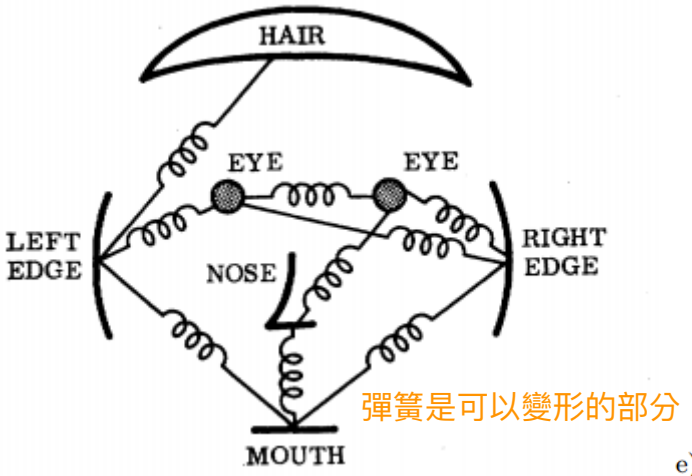


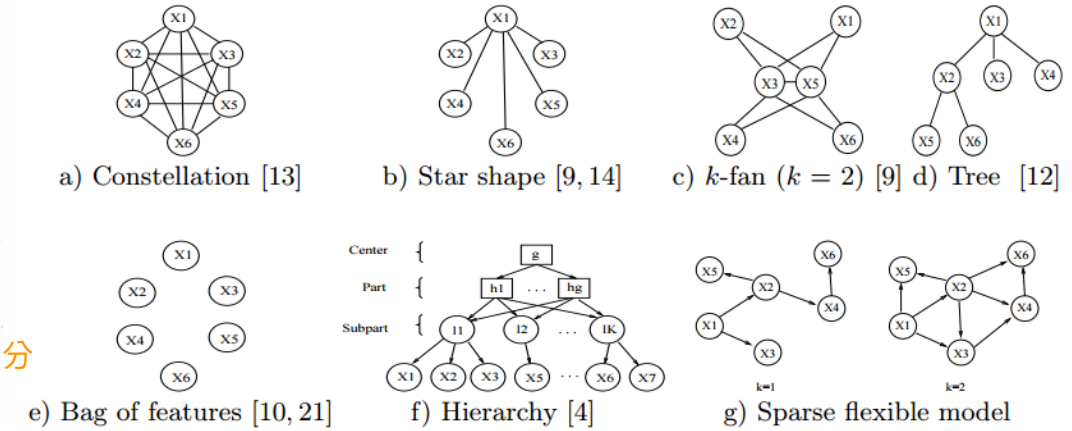
Fig. by Urs Köster

Part-based Method

- Appearance of parts　　形容一個object是由多個part組成
- Spatial relationships between parts



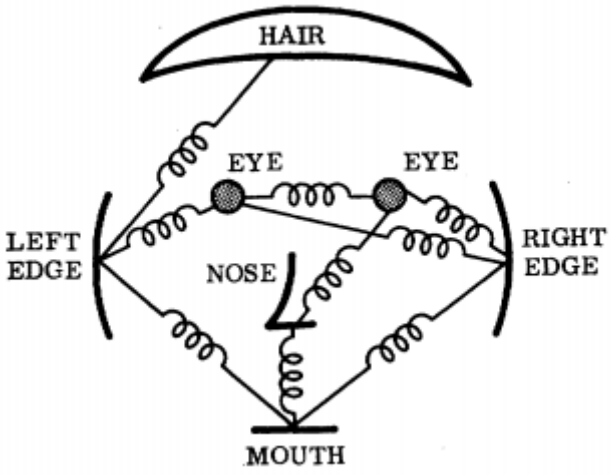
Pictorial models
[Fischler et al., 1973]



Graphical geometric models of priors
[Carneiro et al., 2006]

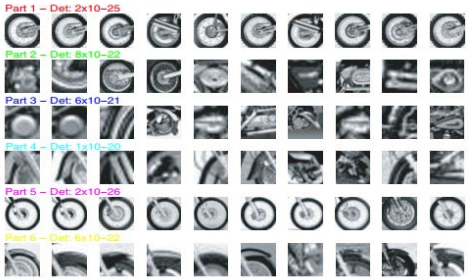
Part-based Method

- Appearance of parts
- Spatial relationships between parts



Pictorial models
[Fischler et al., 1973]

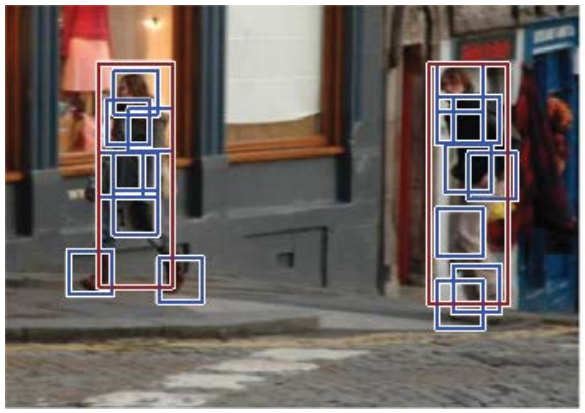
這邊本來有一台車



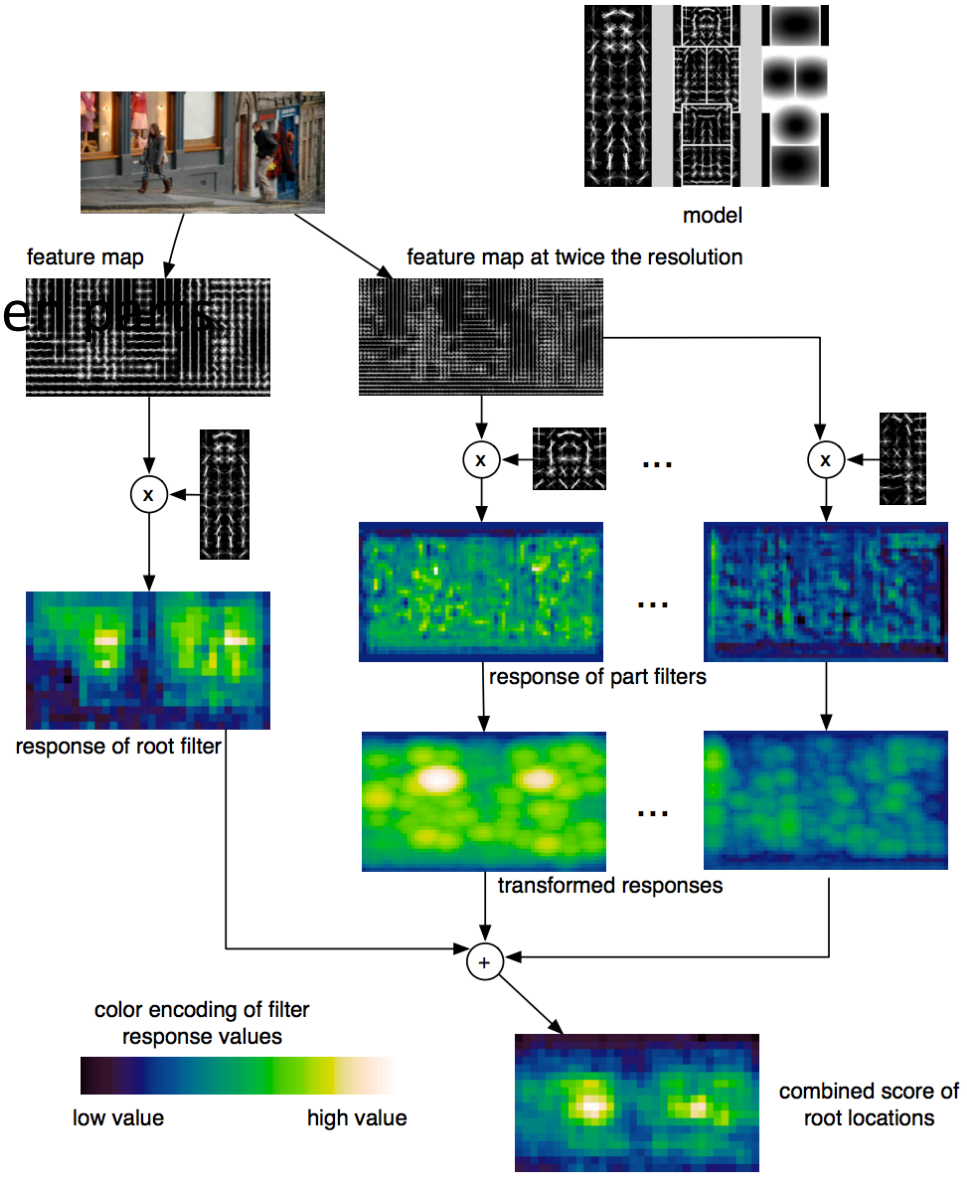
Constellation Model
[Fergus et al., 2007]

Part-based Method

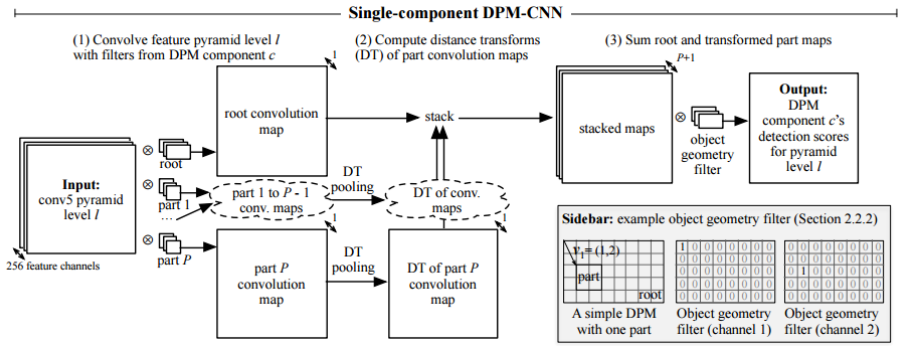
- Appearance of parts
- Spatial relationships between parts



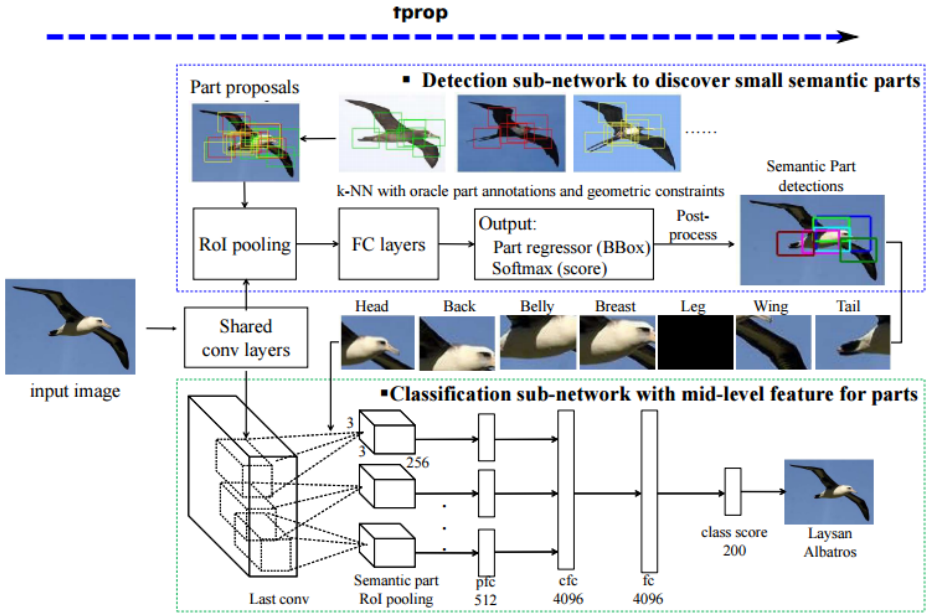
Deformable Part Model
[Felzenszwalb et al., 2010]



CNN + Part-based Method



DPM-CNN [Girshick et al., 2015]



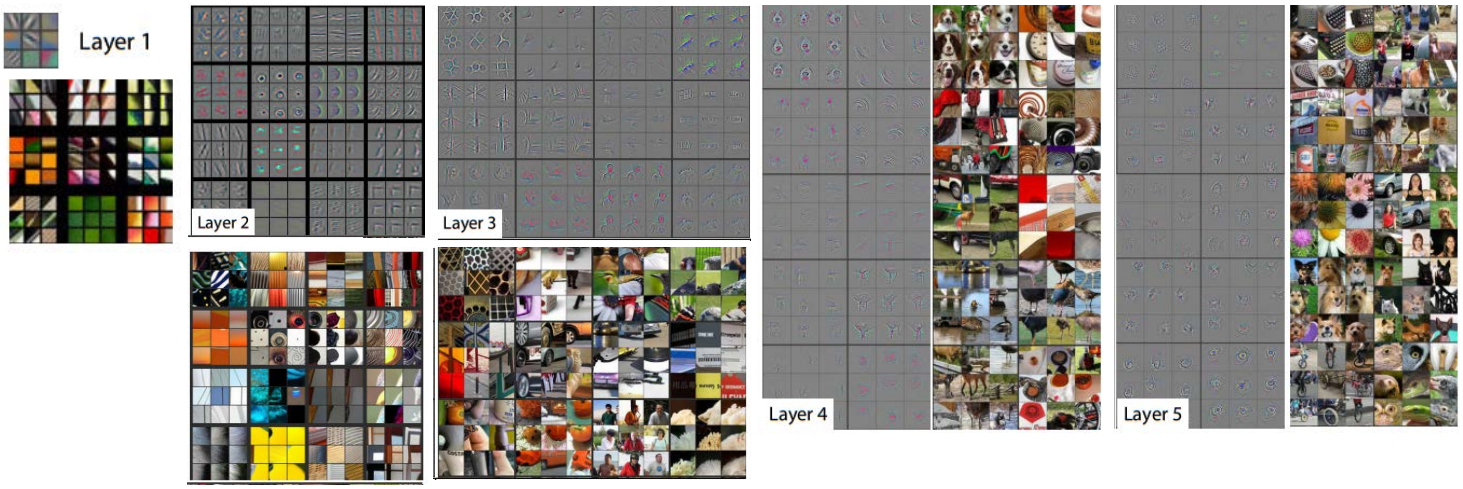
SPDA-CNN [Zhang et al., 2016]

Motivation

- Part-based method
  - Robust to large intra-class variation
  - More discriminative in the case of subtle inter-class variation

- Limitations
  - Pre-defined parts
  - High annotation cost
  - Spatial relationship is difficult to model in CNN

Idea

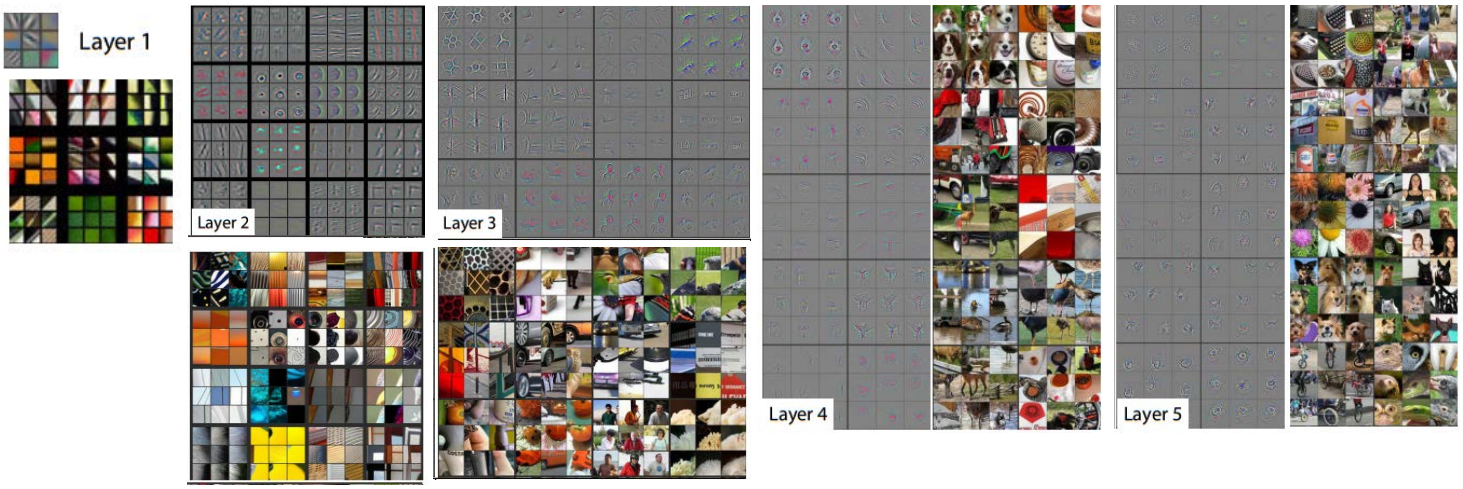


Visualizing and understanding convolutional networks
[Matthew D., and Fergus, 2014]

- Neuron: part detector 每個neuron就是代表某個part
- Feature map: the spatial occurrence of certain part

某個part是在哪邊出現

Idea

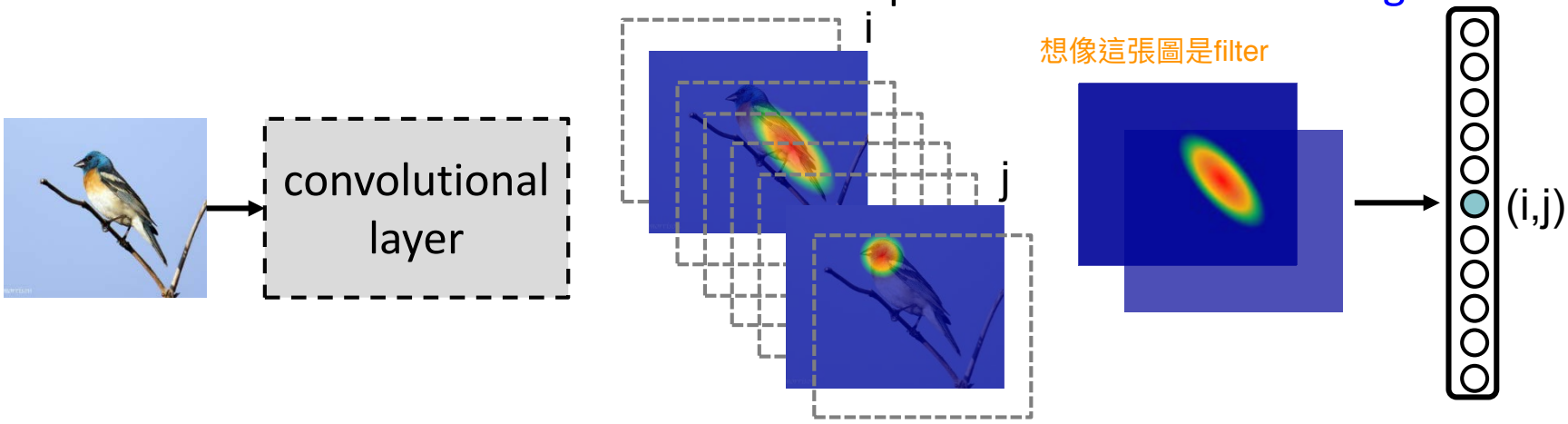


Visualizing and understanding convolutional networks
[Matthew D., and Fergus, 2014]

We introduce **the co-occurrence layer**
to encode the interaction between object parts.

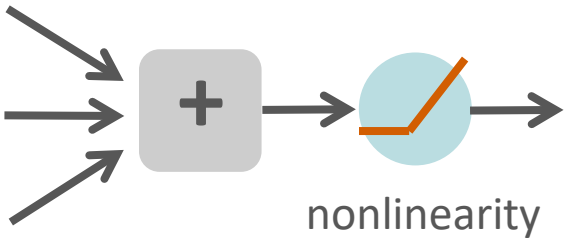
Our Approach

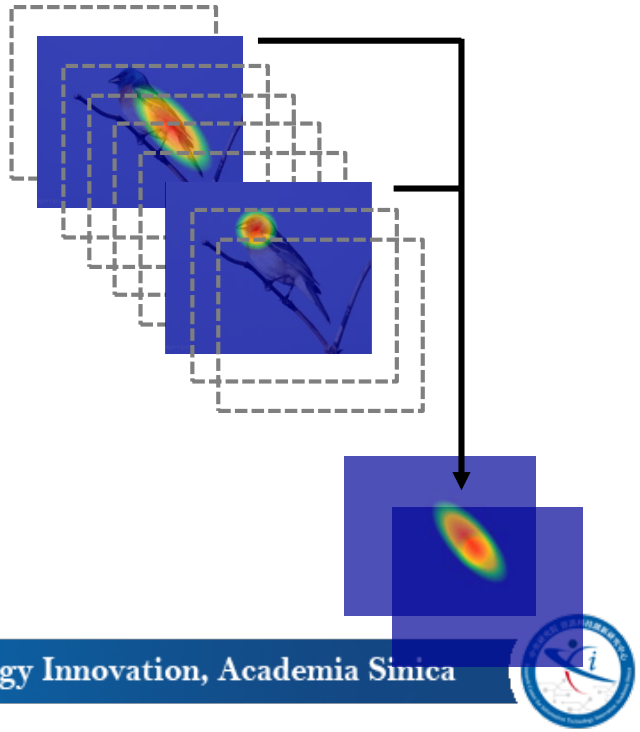
- We present a new network layer, the co-occurrence layer.
  - It generalize a convolutional layer to discover the co-occurrence between discriminative local parts.
- Idea sketch
  - Neuron: local part detector
  - Feature map: spatial occurrence likelihood of that part
  - Mutual convolution: treat feature maps as both filters and images



The Co-occurrence Layer

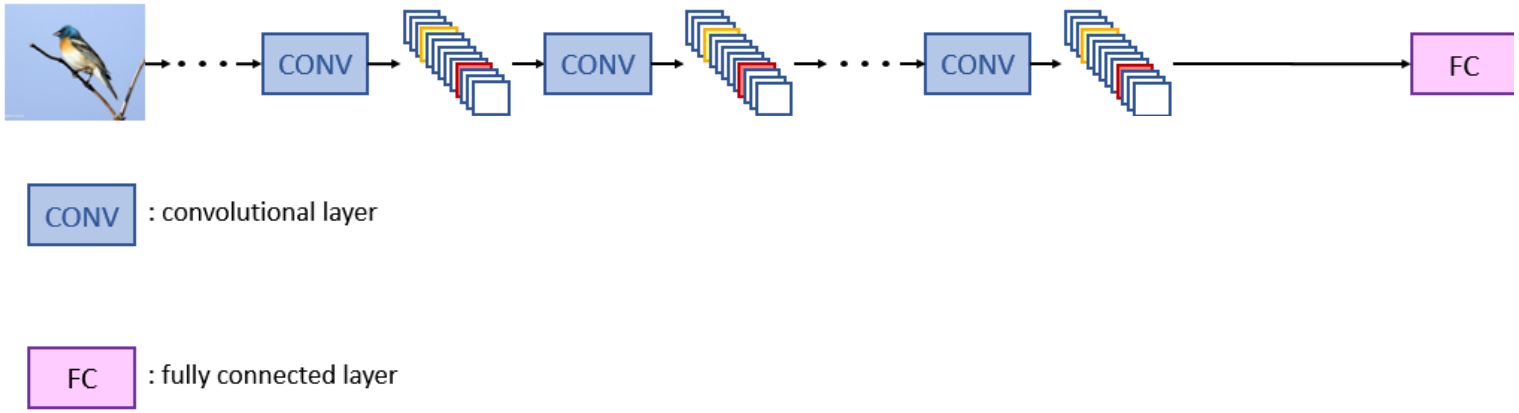
- Differentiable: the resultant CNN is end-to-end trainable
- Nonlinear across neurons
- Robust: rotation- and translation-invariant





The Co-occurrence Layer

- Part annotation free
- Flexible: applicable to various network architectures
- Extendable: coarse-to-fine co-occurrence detection



Experiments: Dataset

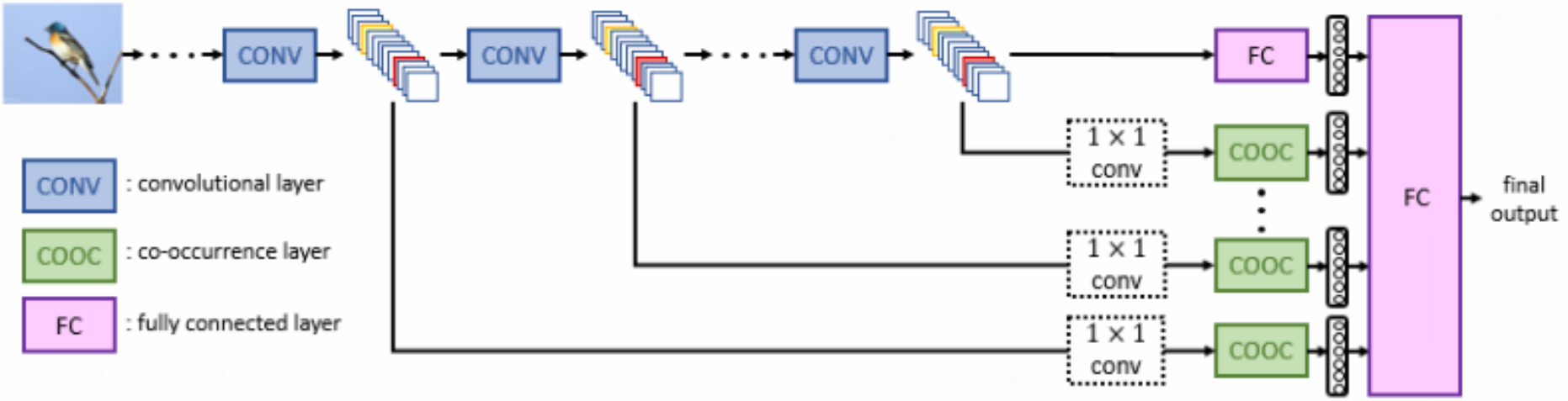
- CUB-200-2011 (Caltech-UCSD Birds-200-2011)

- 200 kinds of birds
- training: 5,994 images
- testing: 5,794 images

Experiments: Setting

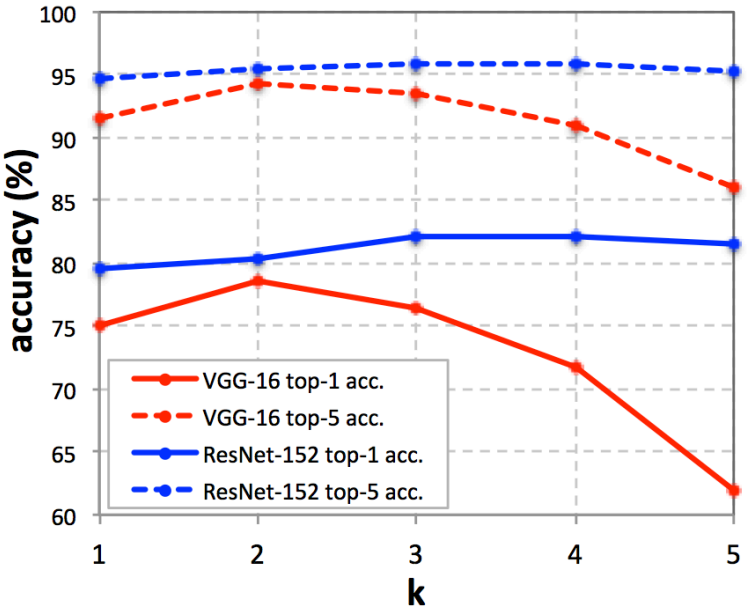
- Networks pre-trained on ImageNet
  - VGG-16
  - ResNet-152

- Implementation details:



Where to Apply the Co-occurrence Layer?

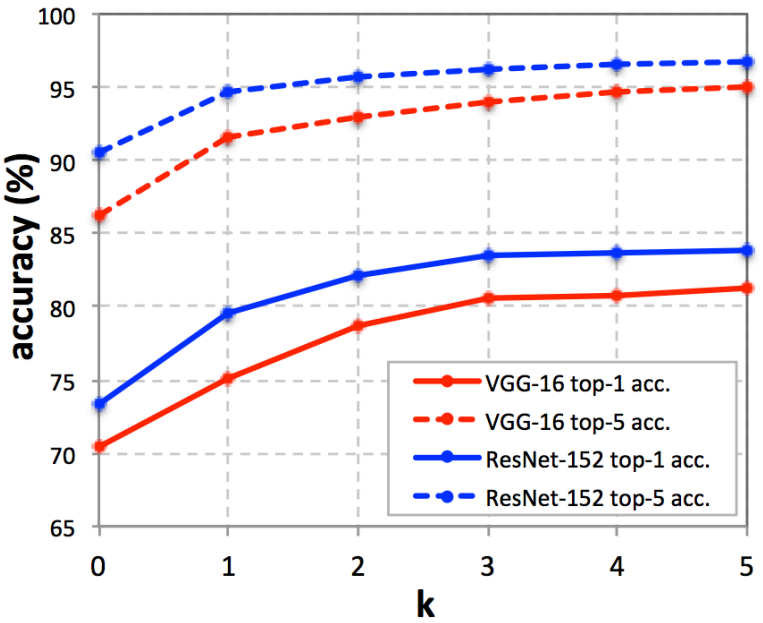
- Earlier convolutional layers tend to extract low-level concepts
- Later convolutional layers tend to extract high-level concepts
- Apply to the k^{th} to the last convolutional layer



- Baseline VGG-16: 70.4%
  VGG-16 + the 2nd to the last: 78.7%
- Baseline ResNet-152: 73.3%
  ResNet-152 + the 3rd to the last: 82.2%
- Lower feature dimensions!

Complementary Property

- Co-occurrence vectors from multiple layers
- Apply to the last k convolutional layers



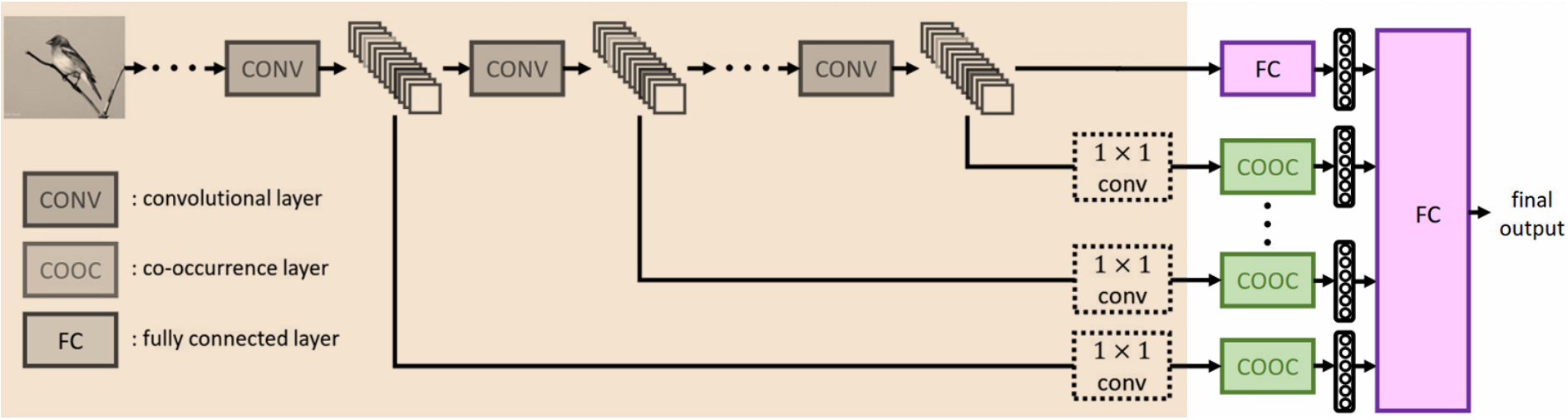
- Baseline VGG-16: 70.4%
  VGG-16 + 3 co-occurrence: 80.6%
  VGG-16 + 5 co-occurrence: 81.3%
- Baseline ResNet-152: 73.3%
  ResNet-152 + 3 co-occurrence: 83.6%
  ResNet-152 + 5 co-occurrence: 83.8%
- They are complementary
- Converge at the last 3 layers

Accuracy

method	network	part annotation	accuracy
Berg *et al.* [CVPR'13]	-	✓	56.9
Göering *et al.* [CVPR'14]	-	✓	57.8
Chai *et al.* [ICCV'13]	-	✓	59.4
Zhang *et al.* [ICCV'13]	-	✓	64.9
Liu *et al.* [CVPR'15]	Caffe		73.5
Zhang *et al.* [ECCV'14]	Caffe	✓	73.9
Branson *et al.* [BMVC'14]	Caffe	✓	75.7
Simon *et al.* [ICCV'15]	VGG		81.0
Krause *et al.* [CVPR'15]	VGG		82.0
Xiao *et al.* [CVPR'15]	AlexNet+VGG		77.9
Wang *et al.* [ICCV'15]	VGG×3		81.7
Lin *et al.* [ICCV'15]	VGG×2		84.1
Jaderberg *et al.* [NIPS'15]	Inception×4		84.1
Ours	VGG		**83.6**
Ours	ResNet-152		**85.8**

Visualization

- For each category, we find the most influential element in all co-occurrence vectors.
- Disply the corresponding pair of feature maps.



200維

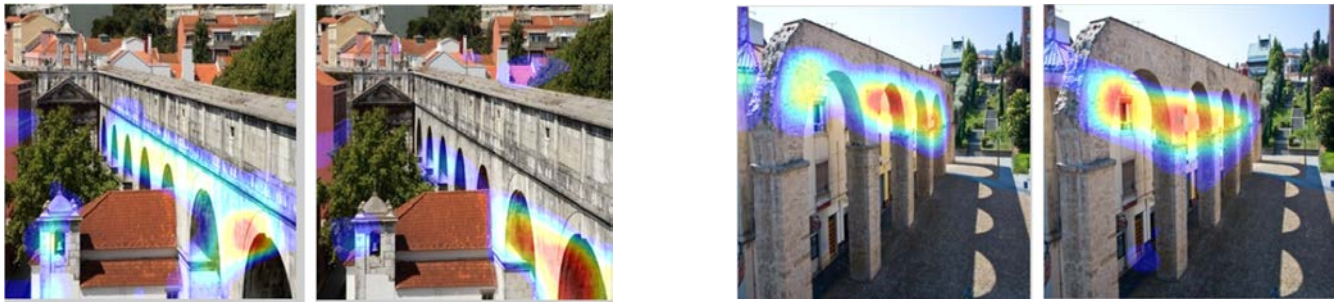
Visualization

Consistency in a class

- Make the recognition results more interpretable.
- Detected parts are consistent and meaningful.

Conclusions

- Breakthrough & impact
  - A new network layer for co-occurrence feature learning
  - Do not require the pre-defined local parts and the part number
  - Do not require part-level annotation
  - The state-of-the-art performance on the CUB-200-2011 dataset
- The source code is released at https://github.com/yafangshih/Deep-COOC.
- Receiver: III (Institute for Information Industry) via project "Scene understanding based on deep learning techniques"



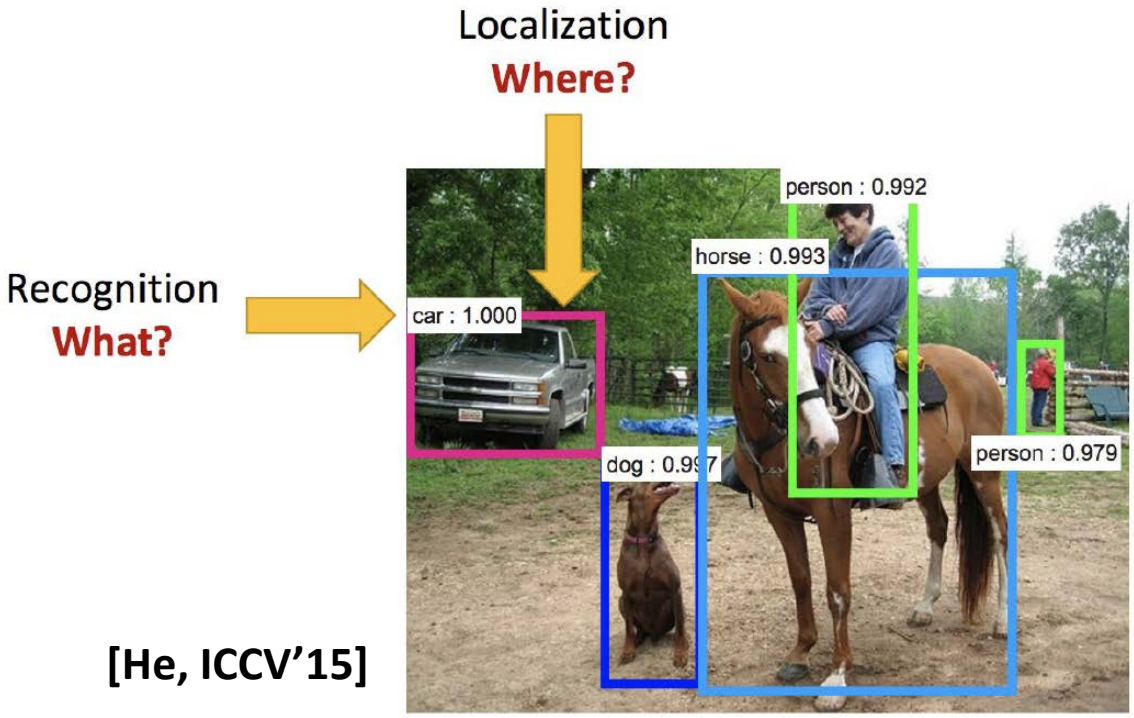
Y.-Y. Lin et al., Deep co-occurrence feature learning for visual object recognition, CVPR, 2017.

Outline

- Convolutional neural networks (CNNs)
- Representative CNN models
- CNN-based computer vision applications
  - Find-grained object recognition
  - Object detection
  - Semantic segmentation
  - Image super resolution
  - Saliency detection
  - Image style transfer
  - Action and gesture recognition
  - Image matching and alignment

Object Detection

- Goal: Detecting instances of semantic objects of certain classes
- Critical to high-level vision tasks such as surveillance, self-driving car, and image retrieval



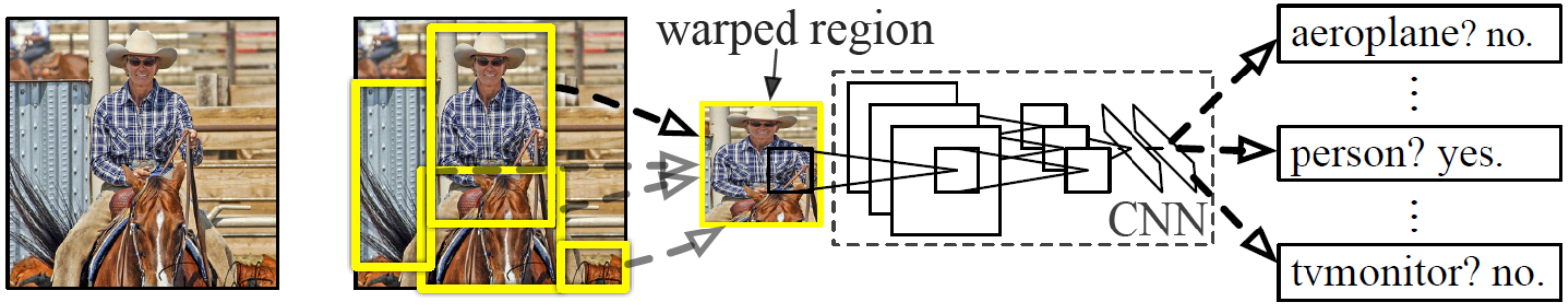
[He, ICCV'15]

R-CNN: Regions with CNN Features

[Girshick et al., CVPR'15]

- System overview

把一個detection的問題轉換成2000個分類問題



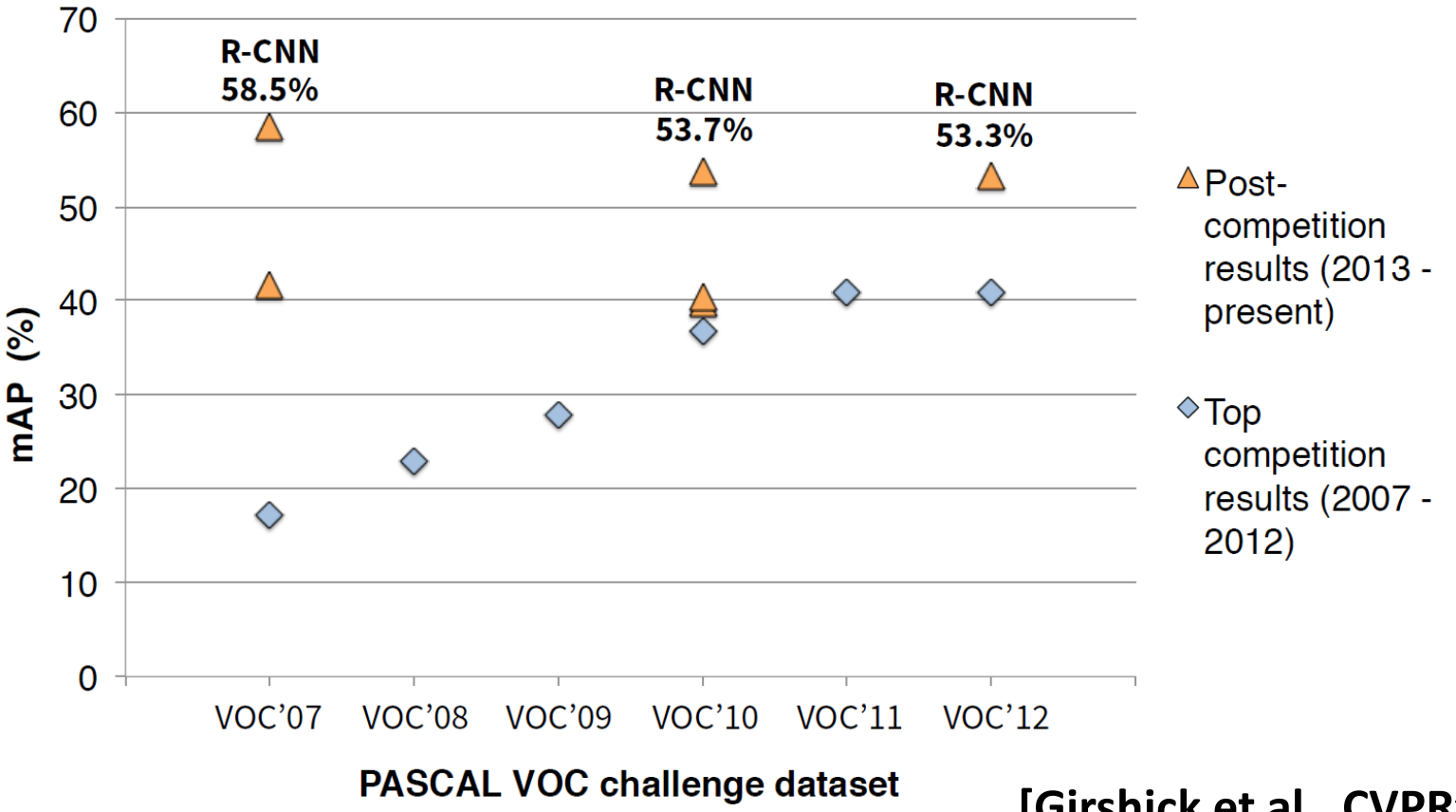
把一張圖先找2000個區塊出來

- Proposal extraction: Using selective search [Uijlin et al., IJCV'13]
- Compute CNN features in the layer 'fc7' of Caffe CNN
- Region classification: linear SVMs or a softmax classifer
- Regression-based bounding box refinement

Experimental Results

- Evaluation on Pascal VOC dataset

效果特別好

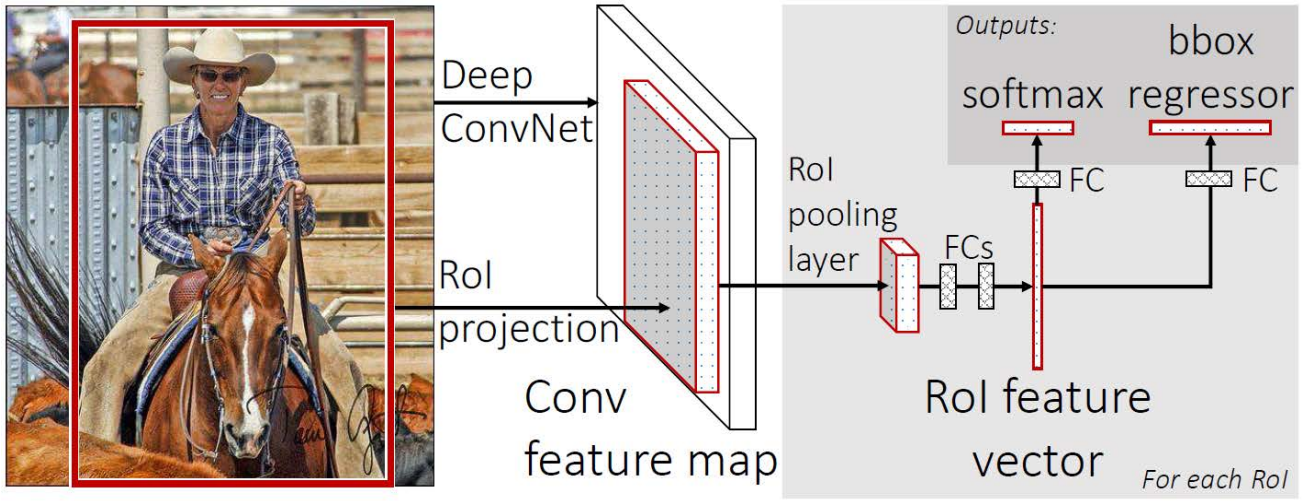


[Girshick et al., CVPR'15]

Fast R-CNN

[Girshick, ICCV'15]

- System overview



- Apply fully convolutional networks to the whole image
- RoI pooling: each proposal is pooled into a fix-size feature map
- Classification with a softmax layer
- Regression-based bounding box refinement

Experimental Results

- Evaluation on Pascal VOC dataset

	Fast R-CNN	R-CNN [1]	SPP-net [2]
Train time (h)	**9.5**	84	25
- Speedup	**8.8x**	1x	3.4x
Test time / image	**0.32s**	47.0s	2.3s
Test speedup	**146x**	1x	20x
mAP	**66.9%**	66.0%	63.1%

[Girshick et al., ICCV'15]

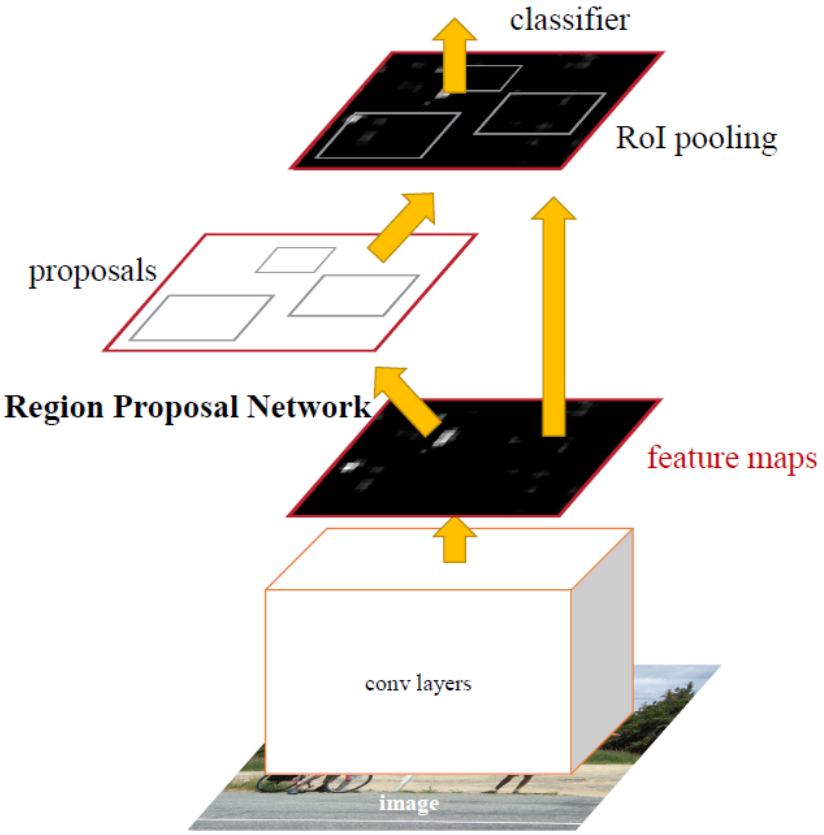
Faster R-CNN

[Ren et al., NIPS'15]

- Selective search is removed
- Region proposal network (RPN)
  - After the last conv. Layer
  - Produces proposals

原本用selective search 抓出的region改用network去抓
ＲＰＮ速度比Selective search快，因為可以用ＧＰＵ

- GPU acceleration is allowed



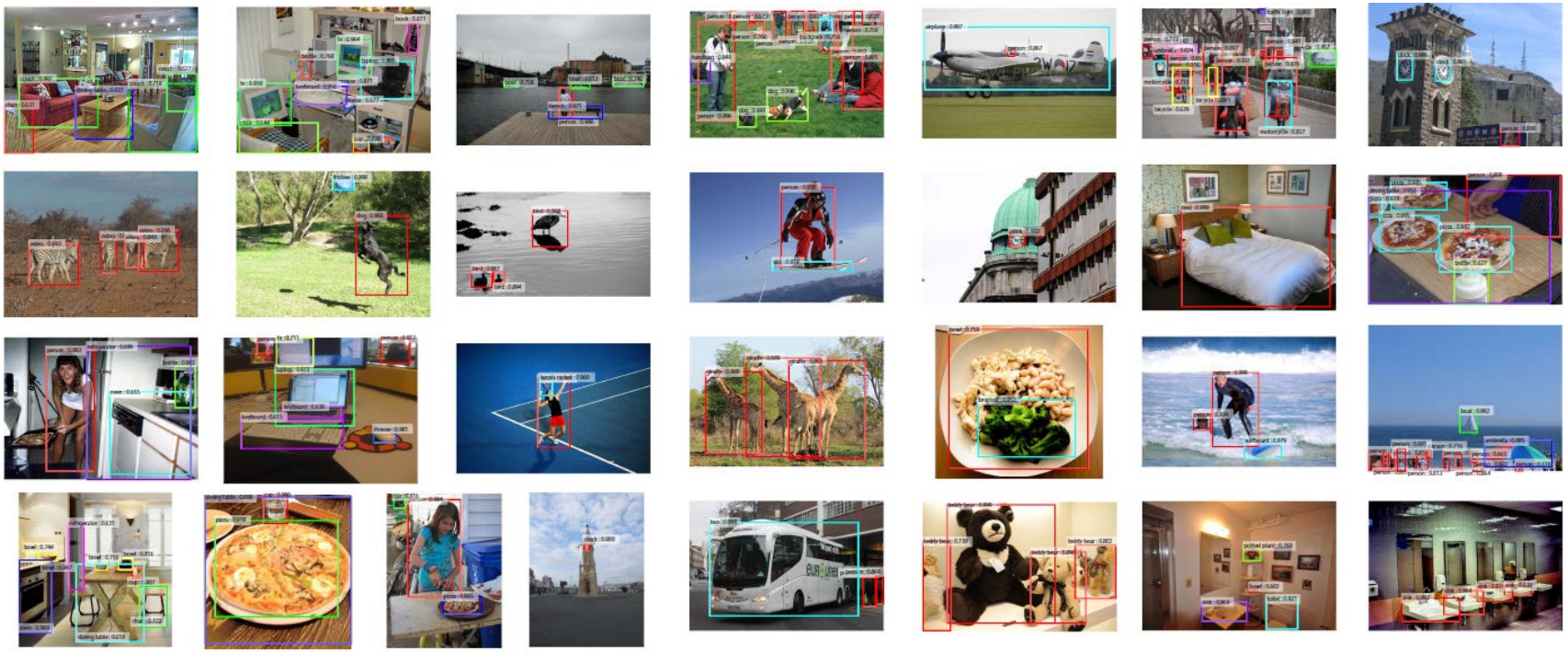
Experimental Results

- Evaluation on Pascal VOC dataset

	R-CNN	Fast R-CNN	Faster R-CNN
Test time per image (with proposals)	50 seconds	2 seconds	**0.2 seconds**
(Speedup)	1x	25x	**250x**
mAP (VOC 2007)	66.0	**66.9**	**66.9**

slide credit: Li et al.

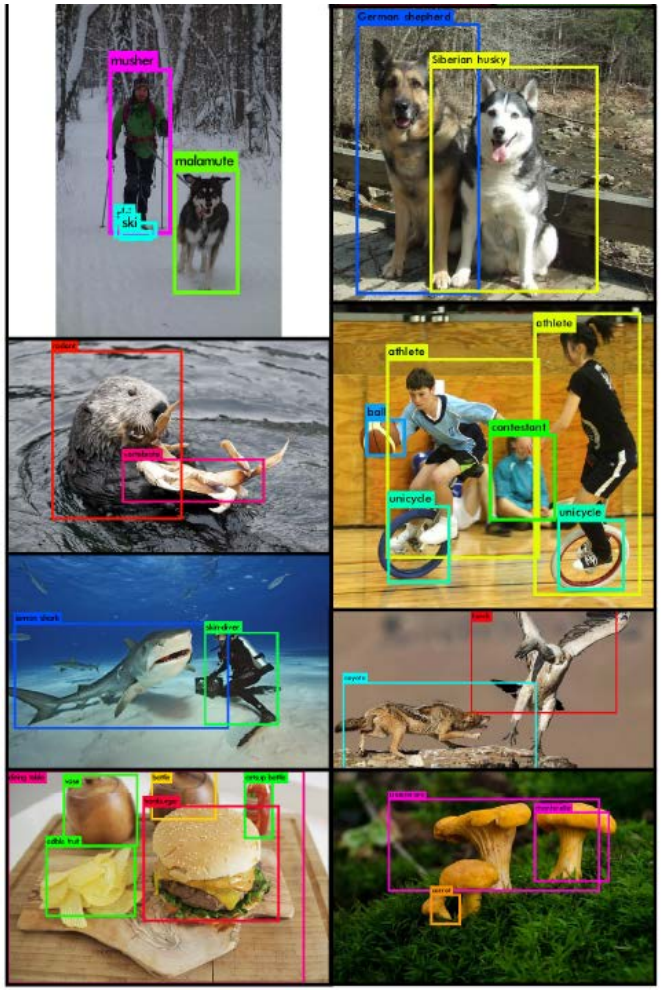
Experimental Results



YOLO9000

[Redmon & Farhadi, CVPR'17]

Detection Frameworks	Train	mAP	FPS
Fast R-CNN [5]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[15]	2007+2012	73.2	7
Faster R-CNN ResNet[6]	2007+2012	76.4	5
YOLO [14]	2007+2012	63.4	45
SSD300 [11]	2007+2012	74.3	46
SSD500 [11]	2007+2012	76.8	19
YOLOv2 288 × 288	2007+2012	69.0	91
YOLOv2 352 × 352	2007+2012	73.7	81
YOLOv2 416 × 416	2007+2012	76.8	67
YOLOv2 480 × 480	2007+2012	77.8	59
YOLOv2 544 × 544	2007+2012	**78.6**	40



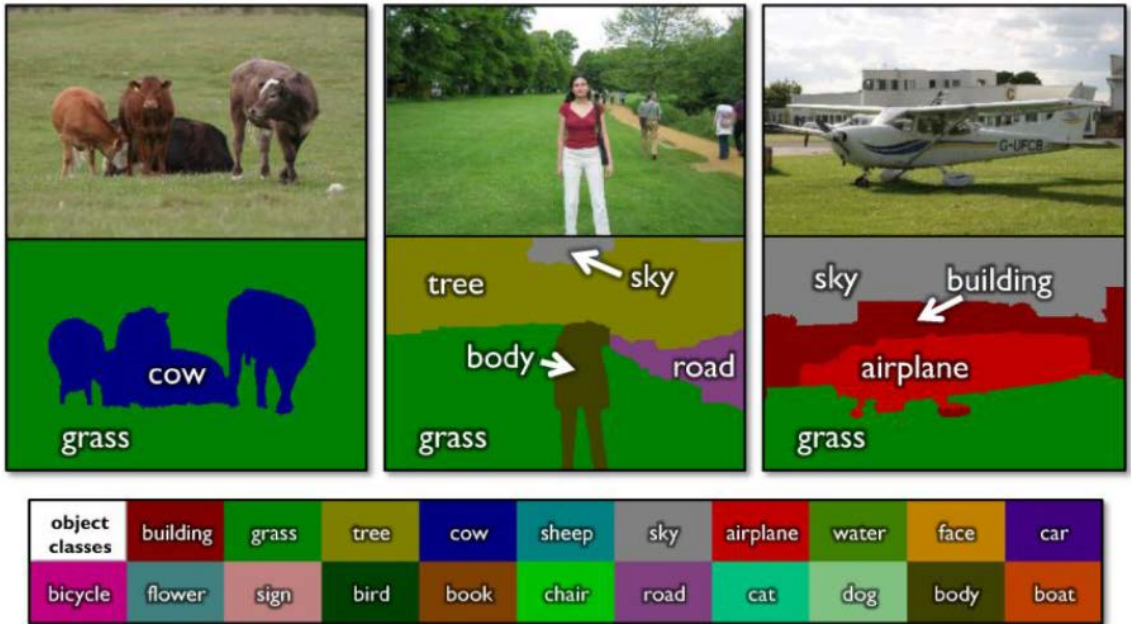
Outline

- Convolutional neural networks (CNNs)

- CNN-based computer vision applications
  - Object detection
  - Semantic segmentation
  - Image super resolution
  - Image style transfer
  - Action recognition
  - Image matching and alignment

Semantic Segmentation

- Goal: Label each pixel to one of predefined classes (or background)
- Critical to high-level vision tasks such as scene understanding, robot navigation, and image retrieval



[*Shotton et al., 2007*]

Introduction: Fully Convolutional Networks for Semantic Segmentation (FCN)

[Long et al. CVPR'15]

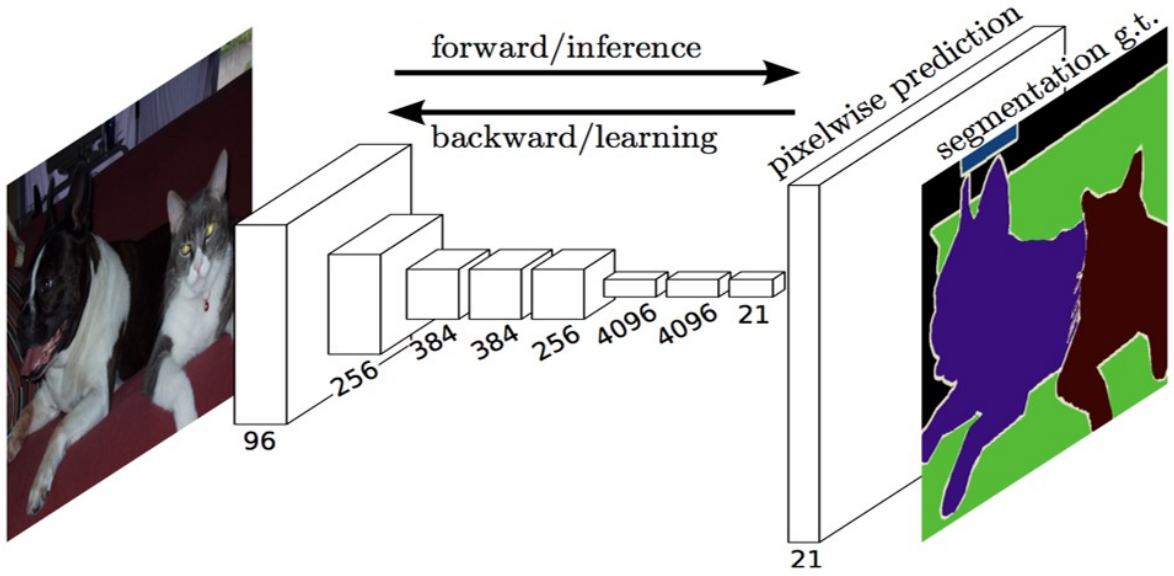
- Fully convolutional networks
  - There are only convolution layer and transposed convolution layer in the network architecture.
- Modify convolution nets for classifier
  - The networks were pre-trained on other datasets such as ImageNet for classification and then fine tuned with datasets for semantic segmentation such as Pascal VOC.

input image from Pascal VOC 2012

segmentation results

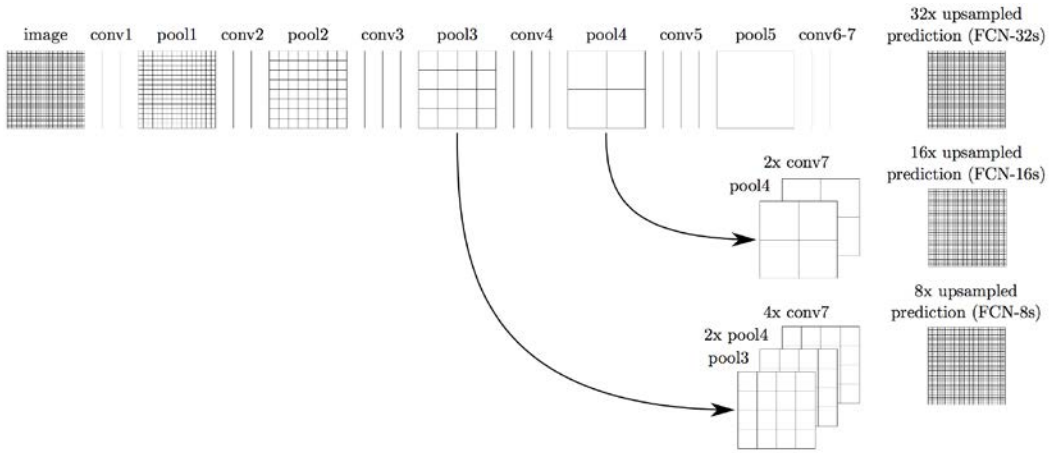
The Approach

- Use AlexNet, VGG 16-layer net, or GoogLeNet
- Discarding the final classifier layer
- Convert all fully connected layers to convolutions
- Append a 1×1 convolution with channel dimension 21 to predict scores for each of the PASCAL classes.

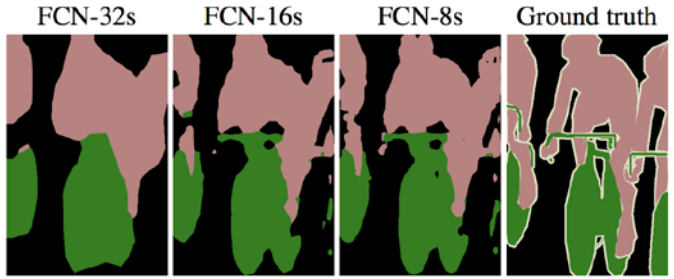


The Approach

- To make the predicted results better, adding the skips to combine the final prediction layer with lower layer with finer stride.



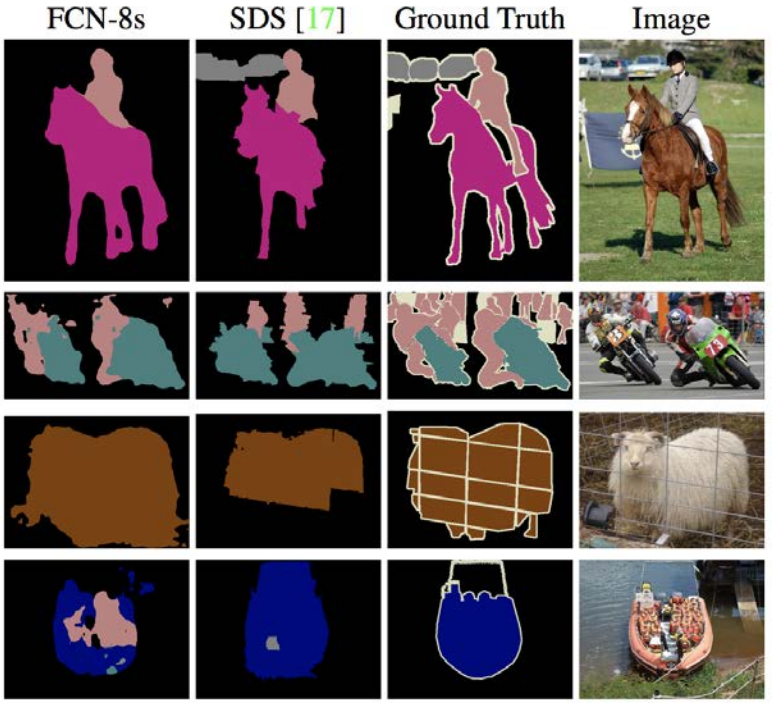
	pixel acc.	mean acc.	mean IU	f.w. IU
FCN-32s-fixed	83.0	59.7	45.4	72.0
FCN-32s	89.1	73.3	59.4	81.4
FCN-16s	90.0	75.7	62.4	83.0
FCN-8s	**90.3**	**75.9**	**62.7**	**83.2**



Experimental Results

- Dataset : Pascal VOC
  - Training : [Hariharan *et al*] collected labels for a larger set of 8498 PASCAL training images .
  - Testing : Test set of Pascal VOC 2011 and 2012

	mean IU VOC2011 test	mean IU VOC2012 test	inference time
R-CNN [12]	47.9	-	-
SDS [17]	52.6	51.6	~ 50 s
FCN-8s	**62.7**	**62.2**	~ **175 ms**

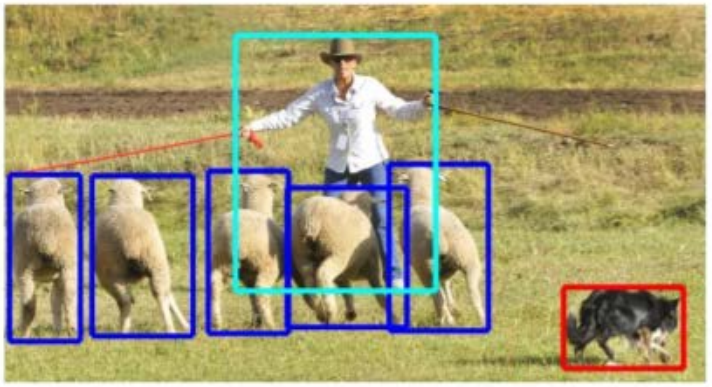


Outline

- Convolutional neural networks (CNNs)
- Representative CNN models
- CNN-based computer vision applications
  - Find-grained object recognition
  - Object detection
  - Semantic segmentation
  - Image super resolution
  - Saliency detection
  - Image style transfer
  - Action and gesture recognition
  - Image matching and alignment

Object Instance Segmentation

(a) Image classification



(b) Object localization

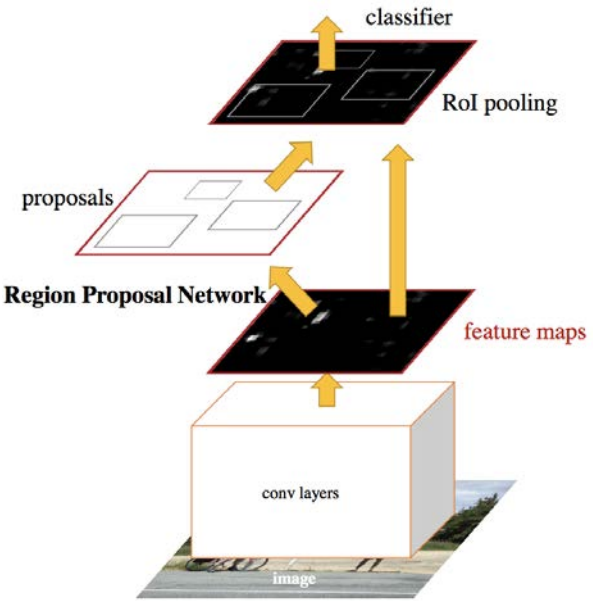
(c) Semantic segmentation

(d) Instance segmentation

Introduction: Mask-RCNN

[He et al. ICCV'17]

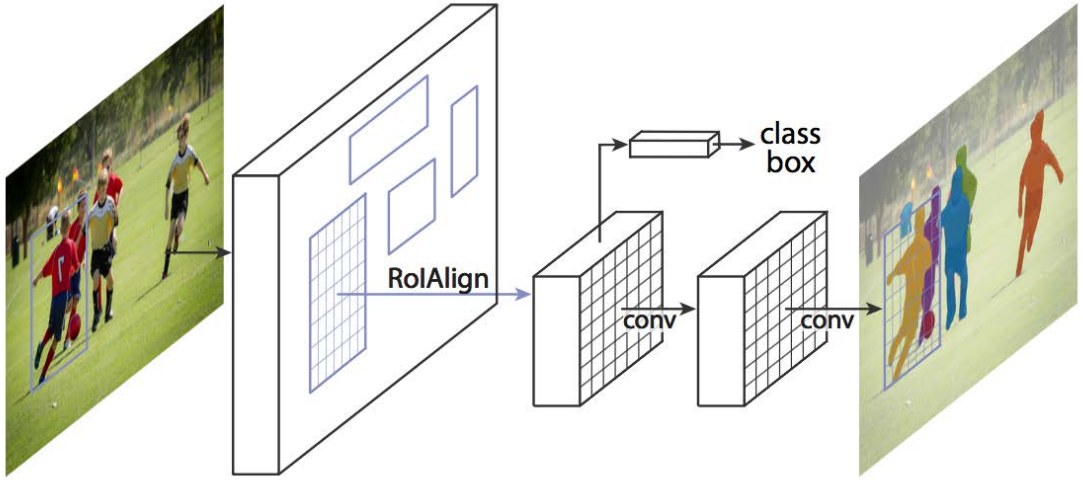
- Instance Segmentation
  - Correct detection of all objects.
  - Precisely Segmenting each objects.
- Faster R-CNN
  - A network for object detection



Upper : input image
Lower : output image
Image: Pascal VOC 2012

The Approach

- Use Resnet [He et al. CVPR'2016] and FPN [Lin et al. CVPR'2017] to get feature maps.
- Use RoIAlign rather than RoIPool.
- Add another branch for mask prediction.
- Use Fully convolution network to predict masks.



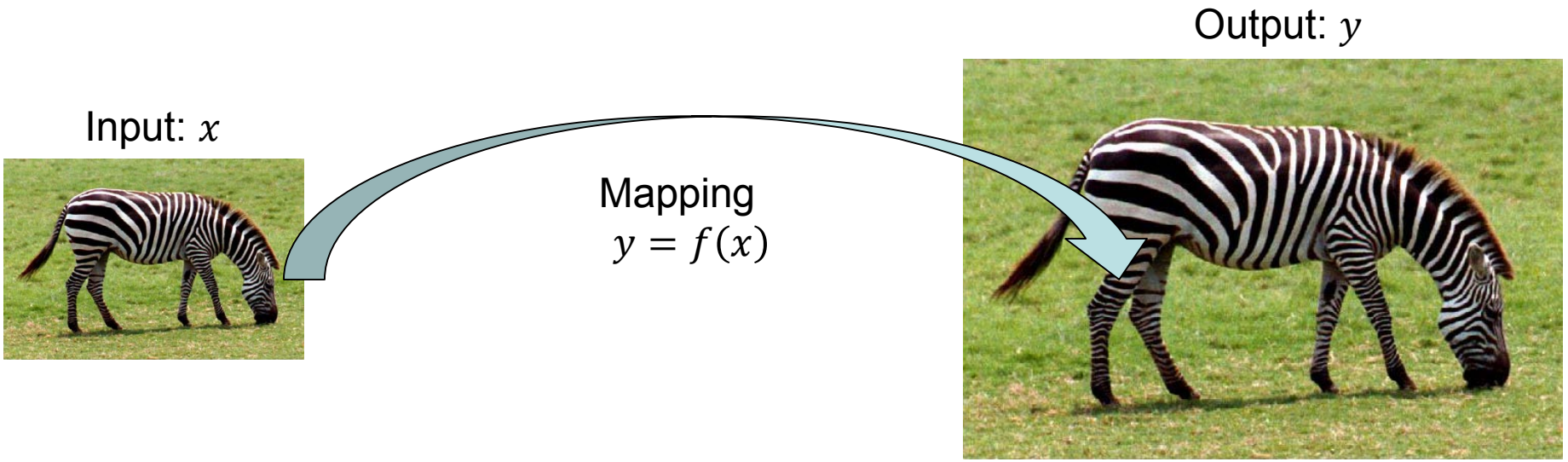
Experimental Results

Outline

- Convolutional neural networks (CNNs)
- Representative CNN models
- CNN-based computer vision applications
  - Find-grained object recognition
  - Object detection
  - Semantic segmentation
  - Image super resolution
  - Saliency detection
  - Image style transfer
  - Action and gesture recognition
  - Image matching and alignment

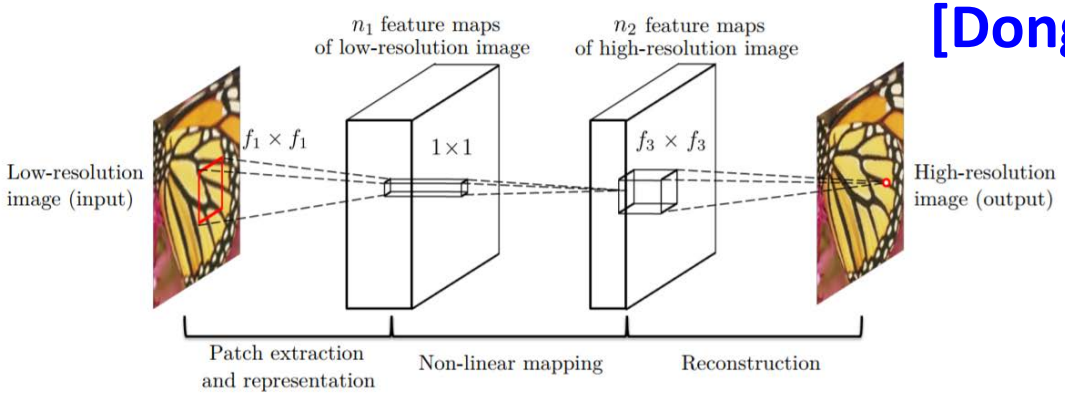
Image Super-Resolution

- Image super-resolution
  - Input: low-resolution (LR) images
  - Output: high-resolution (HR) images
  - Training goal: Learn the mapping between the inputs and outputs
  - Testing goal: Estimate the HR counterpart of a give LR image



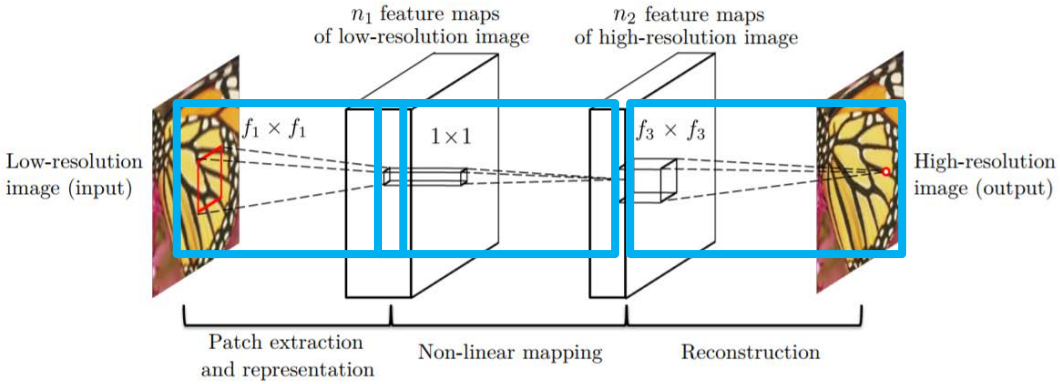
CNNs for Image Super-Resolution (SRCNN)

[Dong et al. ECCV'14 & PAMI'15]



- Learning the mapping with CNNs
  - CNNs: Feature learning and non-linear mapping
- Advantages:
  - High-speed and accuracy
  - No handcraft features
- Difference from AlexNet or VGGNet
  - Fully convolutional networks
  - No pooling operator because of preserving the size

CNNs for Image Super-Resolution (SRCNN)



- 3 layer CNNs: patch extraction, mapping, and reconstruction
- Patch extraction and representation:
  - Extract patches from the low-resolution image
  - Represent each patch as a high-dimensional feature vector with Conv.
- Non-linear mapping: nonlinearly maps each feature vector onto another feature vector
- Reconstruction : aggregate the patch-wise representations to generate the final high-resolution image

Experimental Results

- Dataset
  - Training set: 91 images [20]
  - Test set: Set5, Set14, BSD200

Set5

Eval. Mat	Scale	Bicubic	SC [48]	NE+LLE [4]	KK [24]	ANR [39]	A+ [39]	SRCNN
PSNR	2	33.66	-	35.77	36.20	35.83	36.54	**36.66**
	3	30.39	31.42	31.84	32.28	31.92	32.59	**32.75**
	4	28.42	-	29.61	30.03	29.69	30.28	**30.49**

Set14

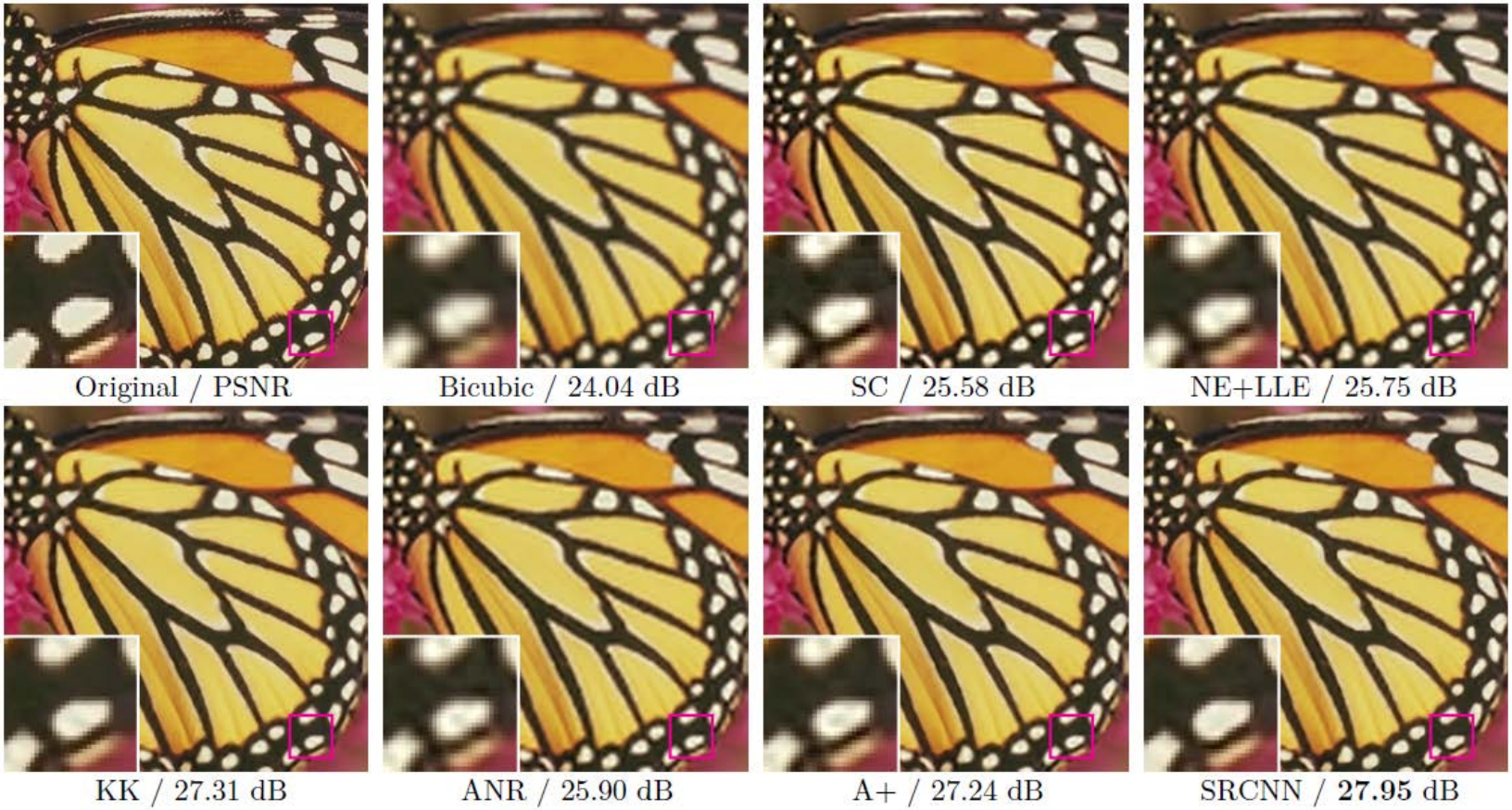
Eval. Mat	Scale	Bicubic	SC [48]	NE+LLE [4]	KK [24]	ANR [39]	A+ [39]	SRCNN
PSNR	2	30.23	-	31.76	32.11	31.80	32.28	**32.45**
	3	27.54	28.31	28.60	28.94	28.65	29.13	**29.30**
	4	26.00	-	26.81	27.14	26.85	27.32	**27.50**

BSD200

Eval. Mat	Scale	Bicubic	SC [48]	NE+LLE [4]	KK [24]	ANR [39]	A+ [39]	SRCNN
PSNR	2	28.38	-	29.67	30.02	29.72	30.14	**30.29**
	3	25.94	26.54	26.67	26.89	26.72	27.05	**27.18**
	4	24.65	-	25.21	25.38	25.25	25.51	**25.60**

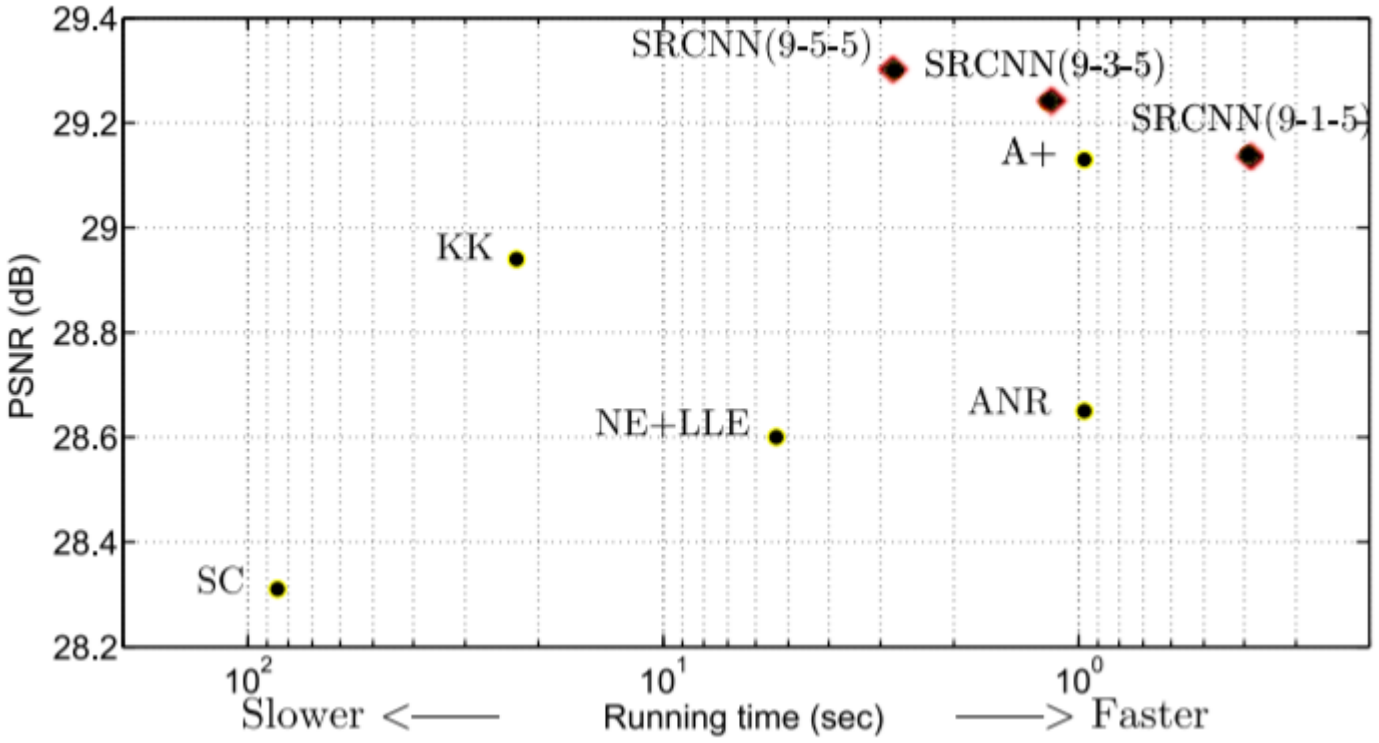
[20] Timofte et al., "Anchored neighborhood regression for fast example-based super-resolution", *ICCV, 2013*

Experimental Results



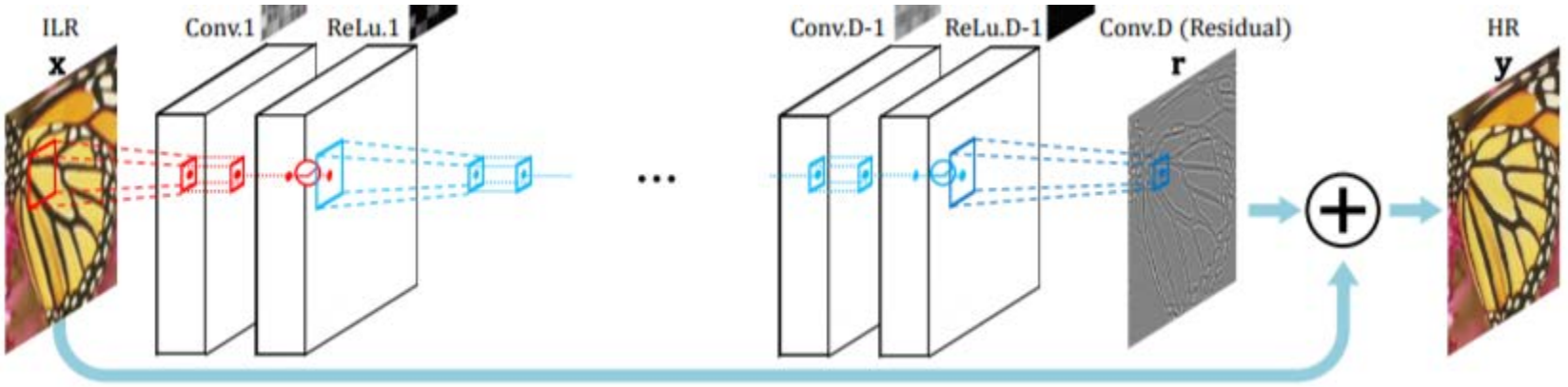
Experimental Results

- Running time vs. PSNR
  - Upper right: better
  - SRCNN: three types of size of Conv.



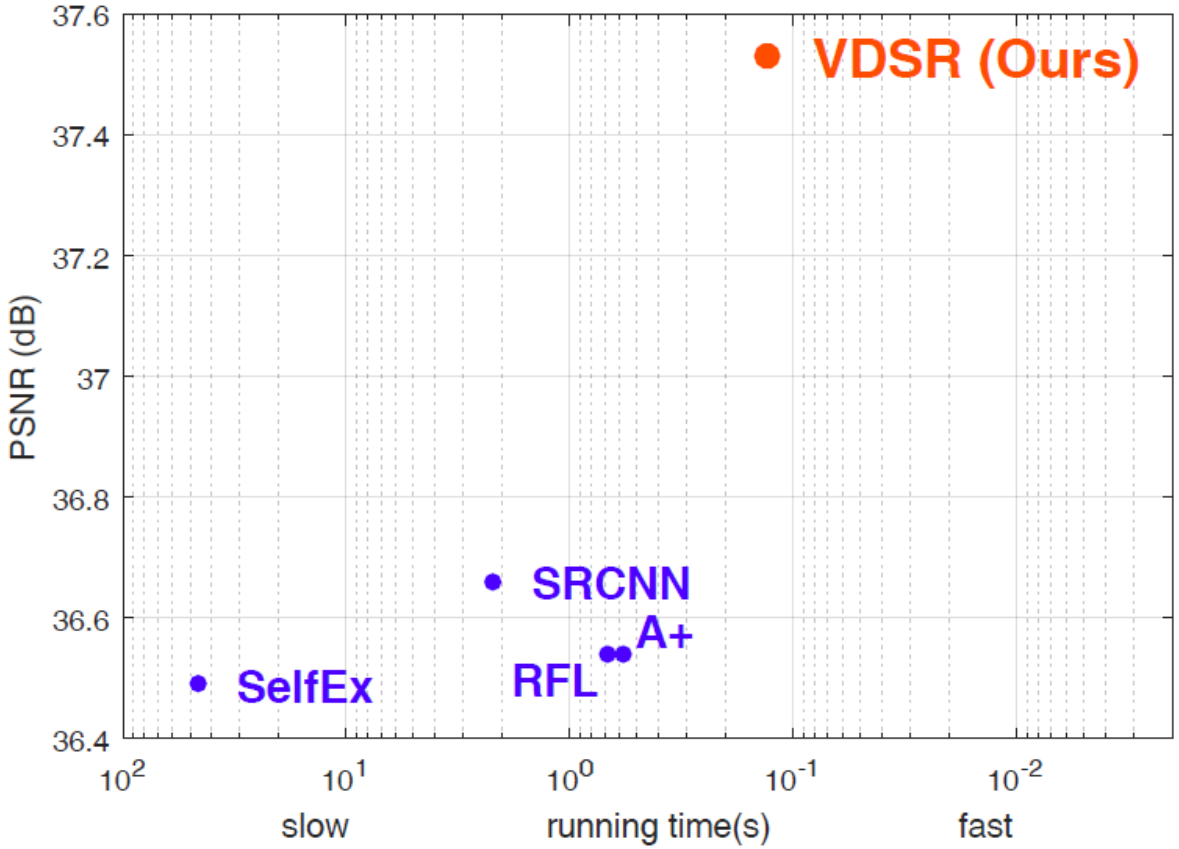
Very Deep Convolutional Networks for Image Super-Resolution (VDSR)

[Kim et al. CVPR'16]



- VDSR vs. SRCNN
  - 20 layers vs. 3 layers
  - Multi-scale learning vs. single-scale learning
- Faster and more stable training
  - Residual learning
  - High learning rates
  - Gradient clipping

Performance Comparison

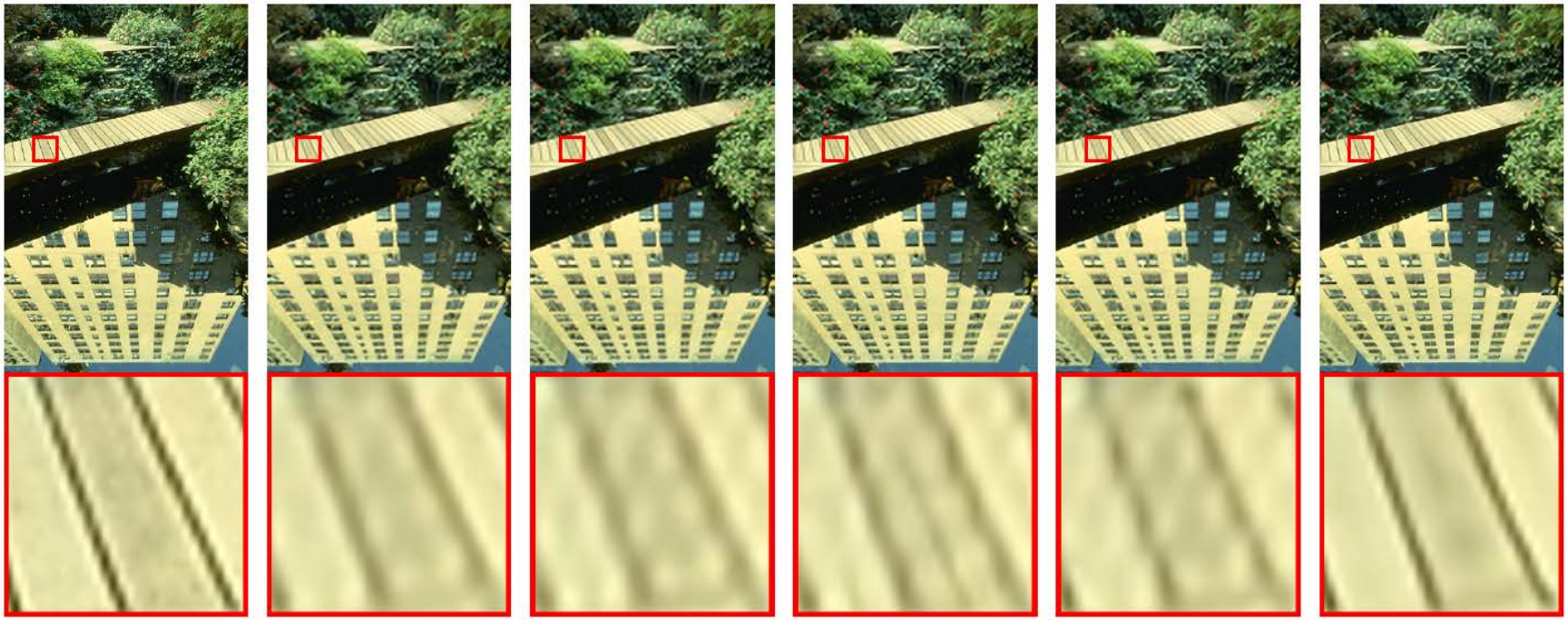


Experimental Results

Dataset	Scale	Bicubic PSNR/SSIM/time	A+ [22] PSNR/SSIM/time	RFL [18] PSNR/SSIM/time	SelfEx [11] PSNR/SSIM/time	SRCNN [5] PSNR/SSIM/time	VDSR (Ours) PSNR/SSIM/time
Set5	×2	33.66/0.9299/0.00	36.54/0.9544/0.58	36.54/0.9537/0.63	36.49/0.9537/45.78	36.66/0.9542/2.19	37.53/0.9587/0.13
	×3	30.39/0.8682/0.00	32.58/0.9088/0.32	32.43/0.9057/0.49	32.58/0.9093/33.44	32.75/0.9090/2.23	33.66/0.9213/0.13
	×4	28.42/0.8104/0.00	30.28/0.8603/0.24	30.14/0.8548/0.38	30.31/0.8619/29.18	30.48/0.8628/2.19	31.35/0.8838/0.12
Set14	×2	30.24/0.8688/0.00	32.28/0.9056/0.86	32.26/0.9040/1.13	32.22/0.9034/105.00	32.42/0.9063/4.32	33.03/0.9124/0.25
	×3	27.55/0.7742/0.00	29.13/0.8188/0.56	29.05/0.8164/0.85	29.16/0.8196/74.69	29.28/0.8209/4.40	29.77/0.8314/0.26
	×4	26.00/0.7027/0.00	27.32/0.7491/0.38	27.24/0.7451/0.65	27.40/0.7518/65.08	27.49/0.7503/4.39	28.01/0.7674/0.25
B100	×2	29.56/0.8431/0.00	31.21/0.8863/0.59	31.16/0.8840/0.80	31.18/0.8855/60.09	31.36/0.8879/2.51	31.90/0.8960/0.16
	×3	27.21/0.7385/0.00	28.29/0.7835/0.33	28.22/0.7806/0.62	28.29/0.7840/40.01	28.41/0.7863/2.58	28.82/0.7976/0.21
	×4	25.96/0.6675/0.00	26.82/0.7087/0.26	26.75/0.7054/0.48	26.84/0.7106/35.87	26.90/0.7101/2.51	27.29/0.7251/0.21
Urban100	×2	26.88/0.8403/0.00	29.20/0.8938/2.96	29.11/0.8904/3.62	29.54/0.8967/663.98	29.50/0.8946/22.12	30.76/0.9140/0.98
	×3	24.46/0.7349/0.00	26.03/0.7973/1.67	25.86/0.7900/2.48	26.44/0.8088/473.60	26.24/0.7989/19.35	27.14/0.8279/1.08
	×4	23.14/0.6577/0.00	24.32/0.7183/1.21	24.19/0.7096/1.88	24.79/0.7374/394.40	24.52/0.7221/18.46	25.18/0.7524/1.06

Table 3: Average PSNR/SSIM for scale factor ×2, ×3 and ×4 on datasets Set5, Set14, B100 and Urban100. Red color indicates the best performance and blue color indicates the second best performance.

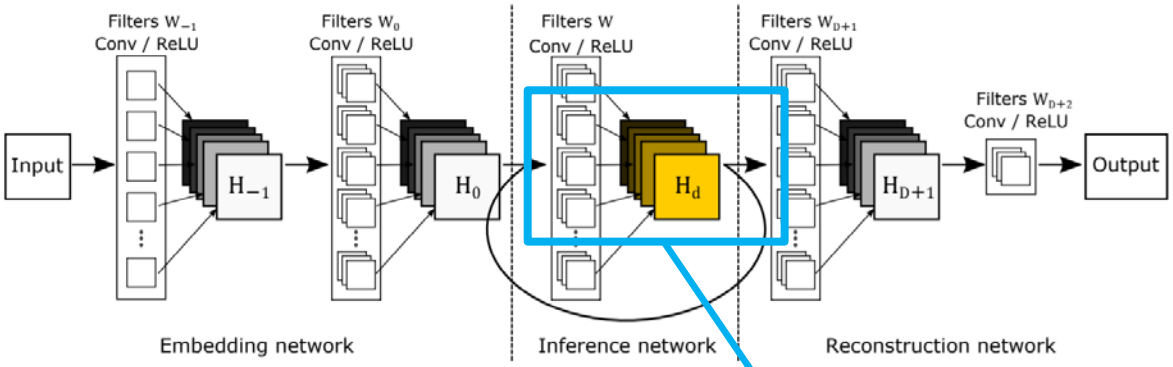
Experimental Results



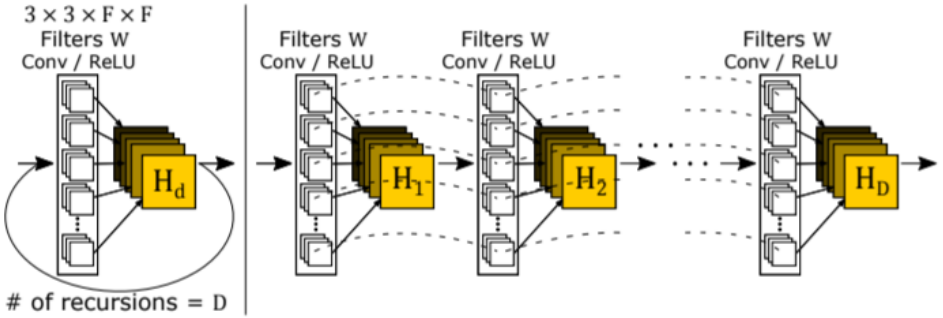
Ground Truth (PSNR, SSIM)	A+ [22] (22.92, 0.7379)	RFL [18] (22.90, 0.7332)	SelfEx [11] (23.00, 0.7439)	SRCNN [5] (23.15, 0.7487)	VDSR (Ours) (23.50, 0.7777)

Deeply-Recursive Convolutional Network for Image Super-Resolution (DRCN)

[Kim et al. CVPR'16]



- Difference from SRCNN
  - Inference network



- Inference network
  - Recursive conv. with the same filters
  - Very deep recursions: significantly boost the performance without increasing the parameters of CNNs

Experimental Results

Dataset	Scale	Bicubic PSNR/SSIM	A+ [29] PSNR/SSIM	SRCNN [5] PSNR/SSIM	RFL [23] PSNR/SSIM	SelfEx [10] PSNR/SSIM	DRCN (Ours) PSNR/SSIM
Set5	×2	33.66/0.9299	36.54/0.9544	36.66/0.9542	36.54/0.9537	36.49/0.9537	37.63/0.9588
	×3	30.39/0.8682	32.58/0.9088	32.75/0.9090	32.43/0.9057	32.58/0.9093	33.82/0.9226
	×4	28.42/0.8104	30.28/0.8603	30.48/0.8628	30.14/0.8548	30.31/0.8619	31.53/0.8854
Set14	×2	30.24/0.8688	32.28/0.9056	32.42/0.9063	32.26/0.9040	32.22/0.9034	33.04/0.9118
	×3	27.55/0.7742	29.13/0.8188	29.28/0.8209	29.05/0.8164	29.16/0.8196	29.76/0.8311
	×4	26.00/0.7027	27.32/0.7491	27.49/0.7503	27.24/0.7451	27.40/0.7518	28.02/0.7670
B100	×2	29.56/0.8431	31.21/0.8863	31.36/0.8879	31.16/0.8840	31.18/0.8855	31.85/0.8942
	×3	27.21/0.7385	28.29/0.7835	28.41/0.7863	28.22/0.7806	28.29/0.7840	28.80/0.7963
	×4	25.96/0.6675	26.82/0.7087	26.90/0.7101	26.75/0.7054	26.84/0.7106	27.23/0.7233
Urban100	×2	26.88/0.8403	29.20/0.8938	29.50/0.8946	29.11/0.8904	29.54/0.8967	30.75/0.9133
	×3	24.46/0.7349	26.03/0.7973	26.24/0.7989	25.86/0.7900	26.44/0.8088	27.15/0.8276
	×4	23.14/0.6577	24.32/0.7183	24.52/0.7221	24.19/0.7096	24.79/0.7374	25.14/0.7510

Table 1: Benchmark results. Average PSNR/SSIMs for scale factor ×2, ×3 and ×4 on datasets Set5, Set14, B100 and Urban100. Red color indicates the best performance and blue color refers the second best.

Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network (SRGAN)

[Ledig et al. CVPR'17]

- Previous work
  - Regard CNNs as the mapping function or generator
- Limitation:
  - Over-smoothing or blurring
  - Hard to recover the highly textured details
- SRGAN: generator + discriminator
  - Jointly learn the generator and discriminator
  - Make the results look more like the real-world images

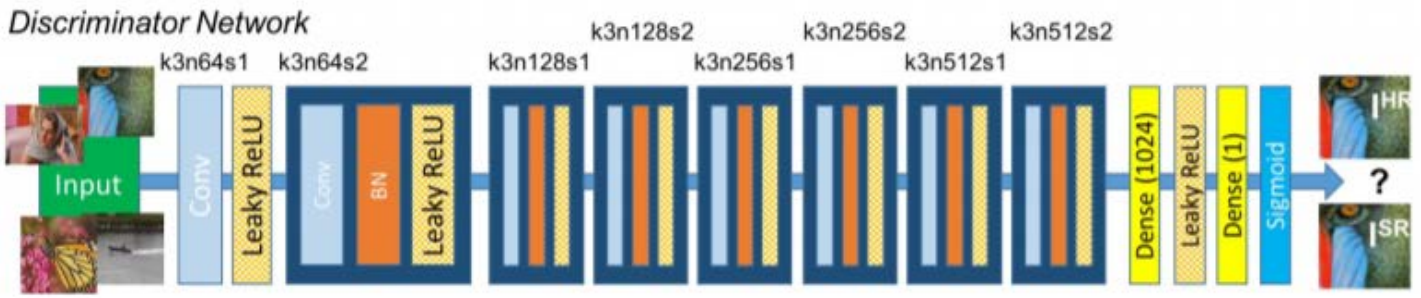
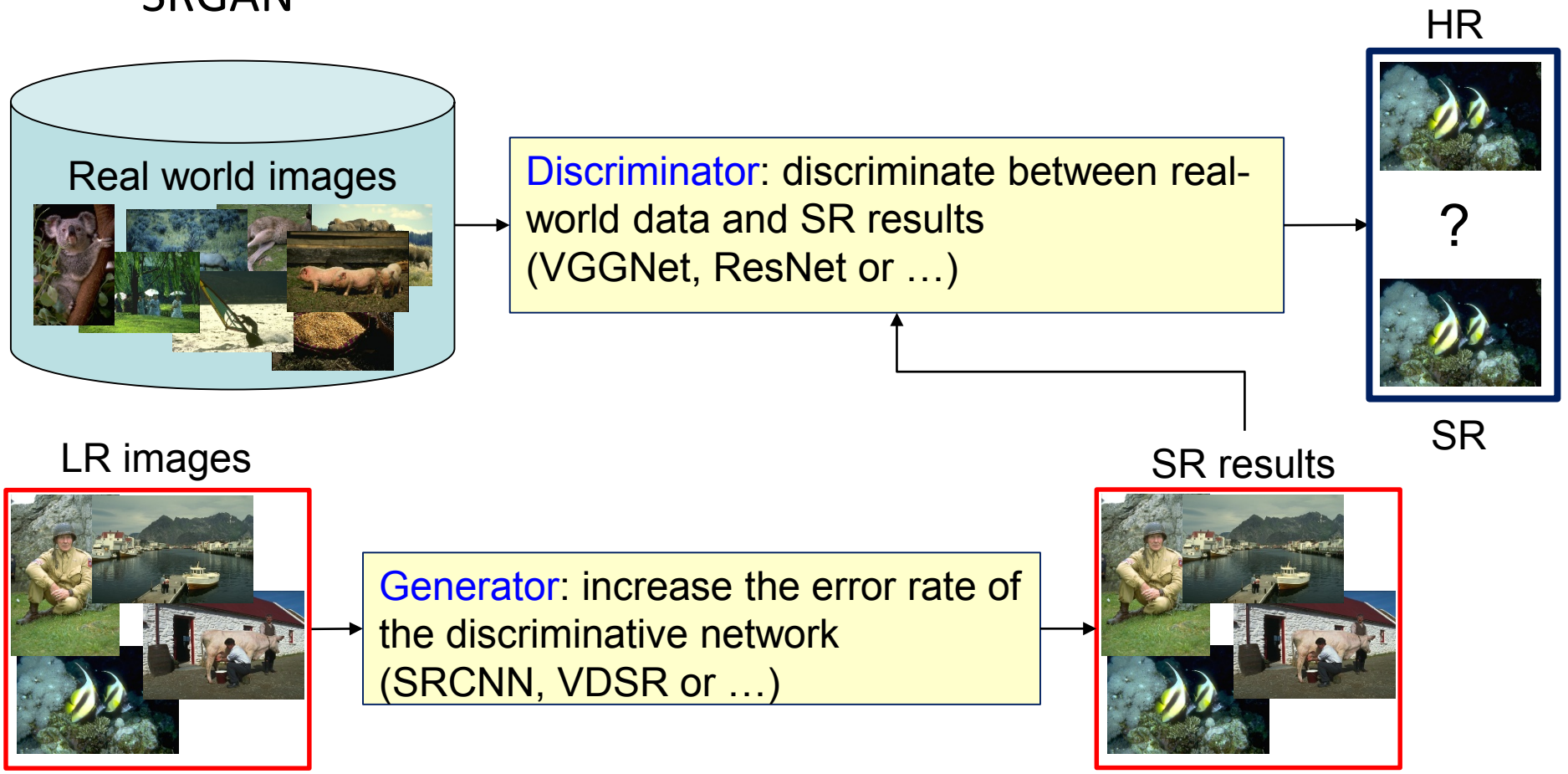


Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network (SRGAN)

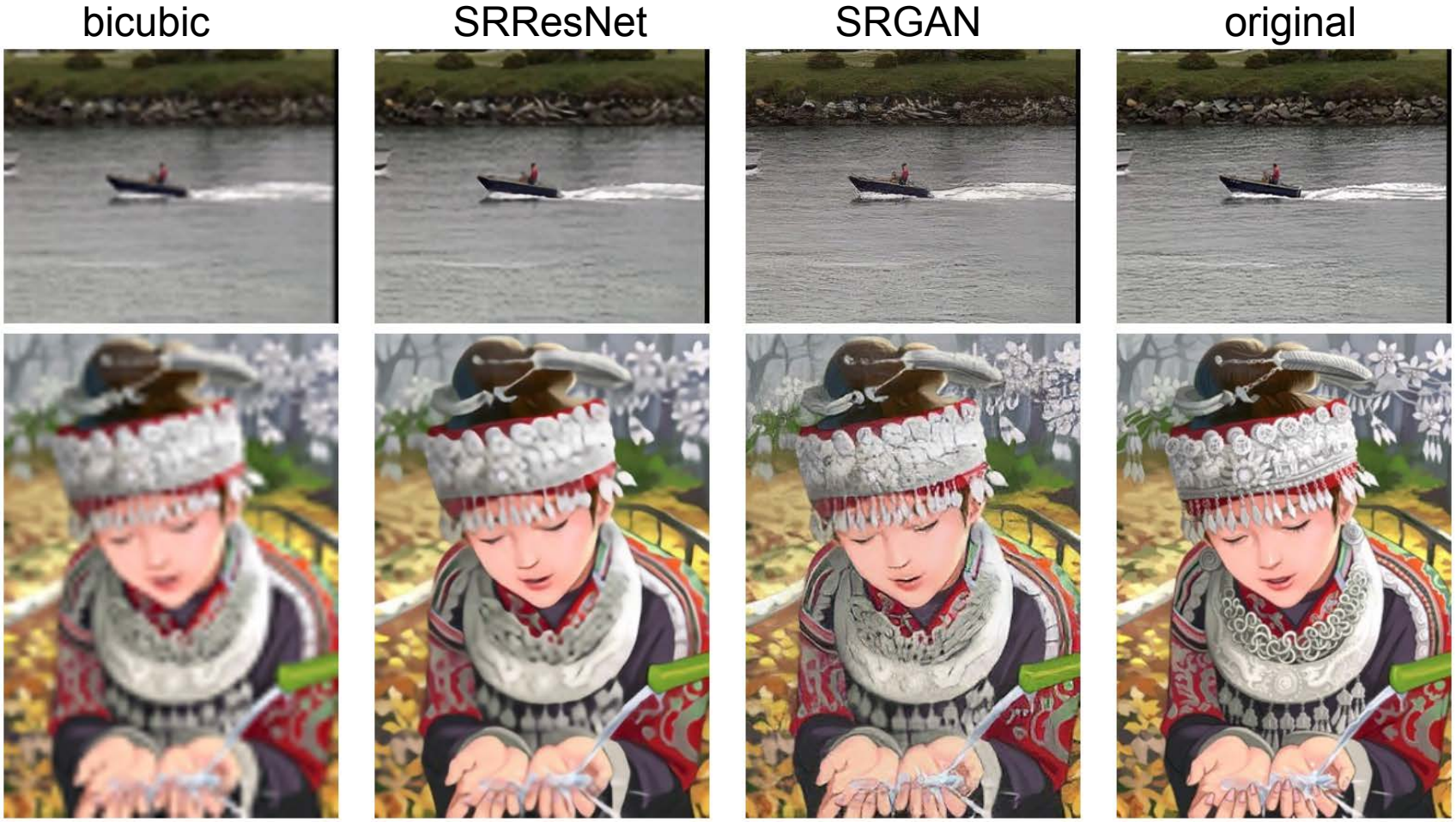
[Ledig et al. CVPR'17]

- SRGAN



Experimental Results

SRResNet: the state-of-the-art method but without the discriminator network



Outline

- Convolutional neural networks (CNNs)
- Representative CNN models
- CNN-based computer vision applications
  - Find-grained object recognition
  - Object detection
  - Semantic segmentation
  - Image super resolution
  - Saliency detection
  - Image style transfer
  - Action and gesture recognition
  - Image matching and alignment

Introduction

跟object segmentation 很像

[Hsu et al. BMVC'17]

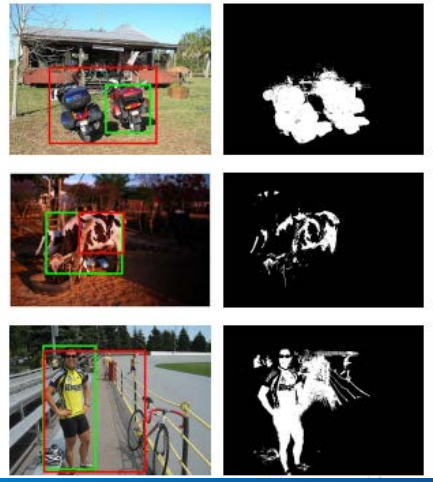
- Object saliency: Highlight salient objects in images

Image

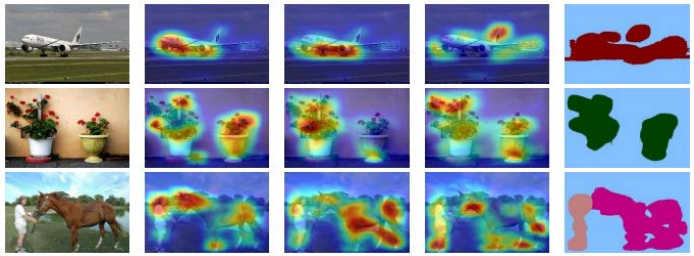
Saliency map

- Applications: visual tracking, object localization, semantic segmentation and so on

[Hong et al., ICML'15]



[Liu and Gong, IJCAI'17]



[Wei et al., CVPR'17]

Related work: Bottom-up approaches

- Aim to detect all objects attracting humans

Image 

Saliency map 

- Two types of object saliency prior:
  - Unsupervised object observation
  - Supervised prior learning from training data
- Limitations:
  - Ambiguous
  - Lack high-level semantic meaning

Related work: Top-down approaches

- Category-specific objects attracting humans
  - Detect objects belonging to the target category
- Category-specific information: learn the object concept from training data
- Advantage: High-quality saliency maps with semantic meaning
- Limitation: High annotation cost in training data collection

Goal

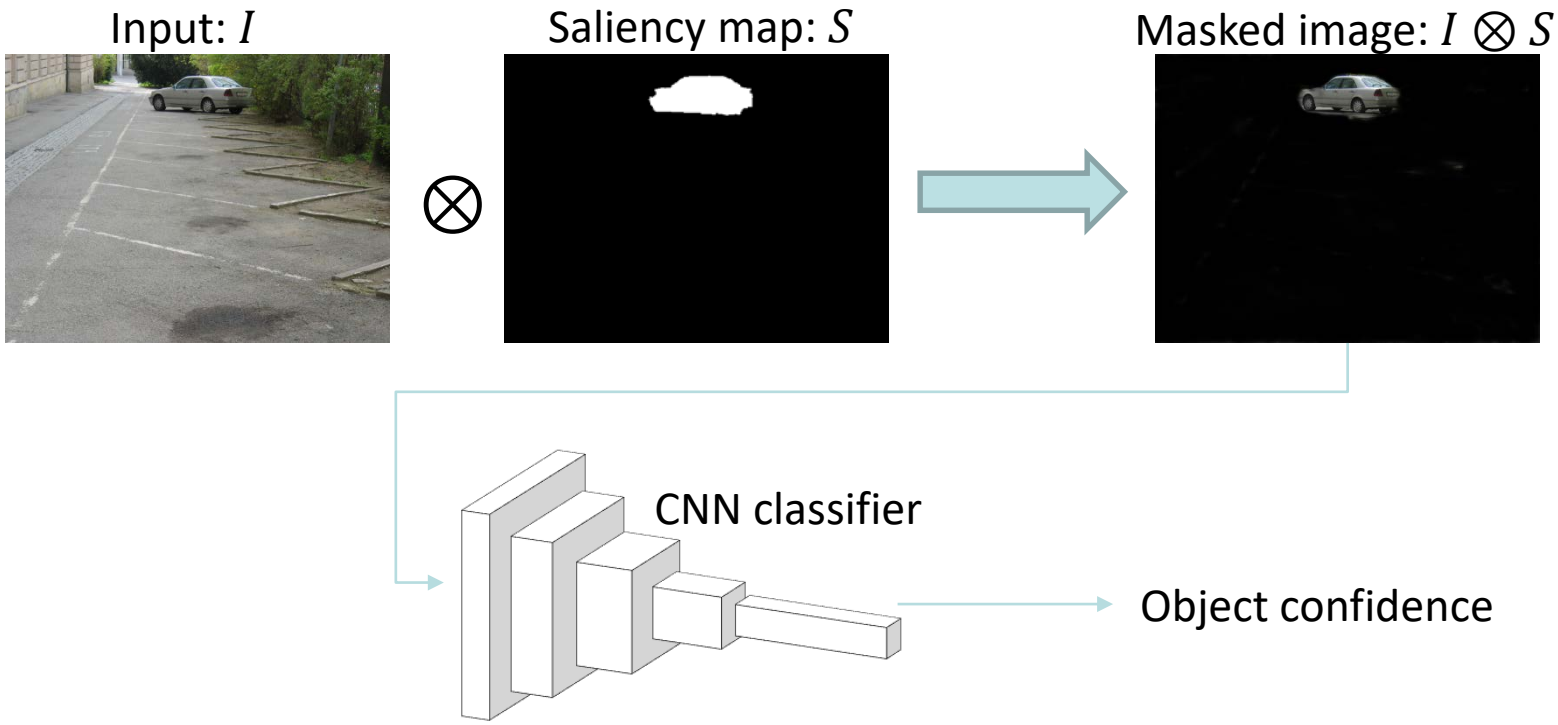
- A CNN-based weakly supervised method for top-down saliency detection
- Training data labeled in the image level, instead of segment level

bike

no bike

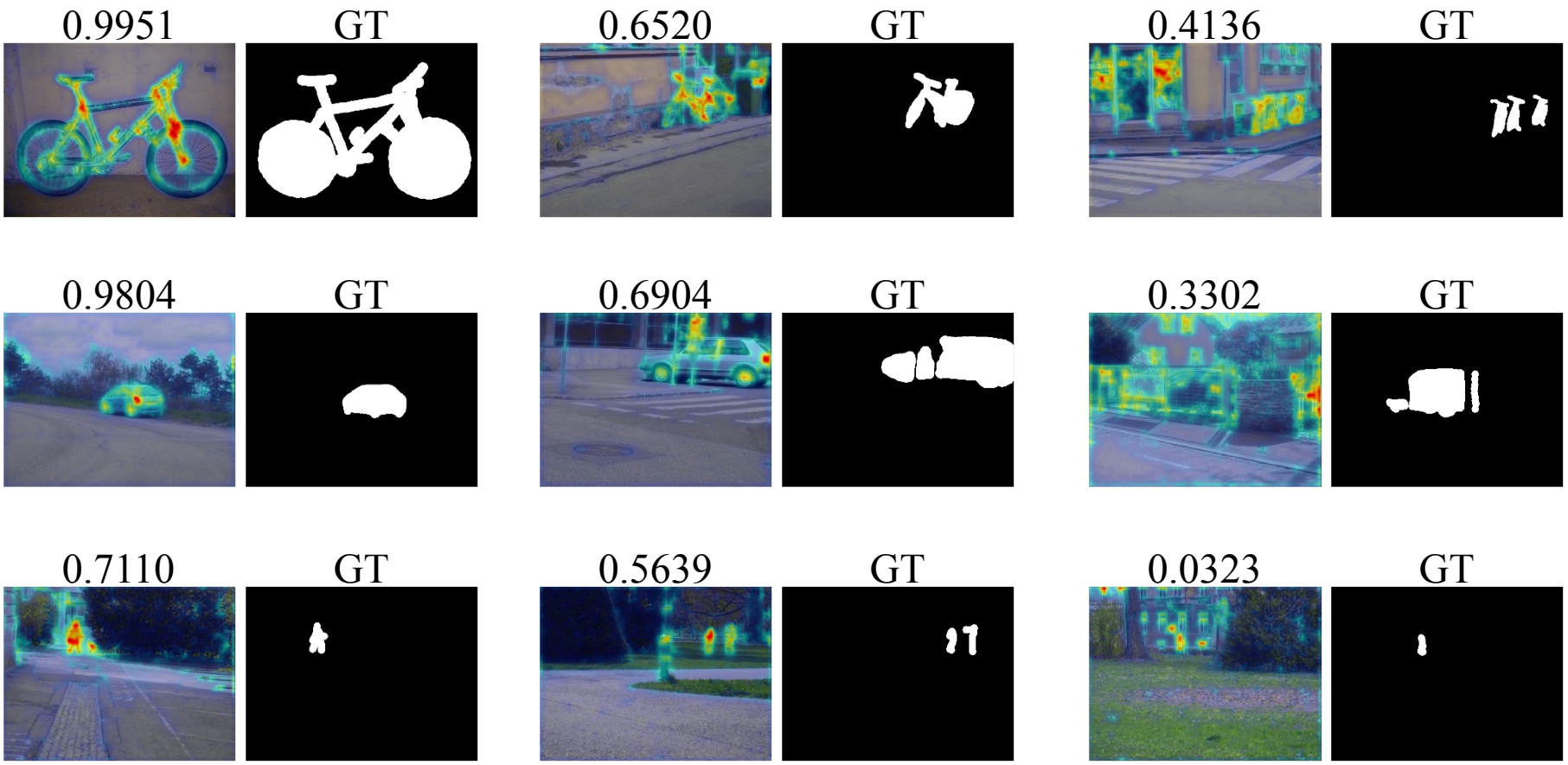
Observation

- Train a CNN classifier to separate bike images from non-bike one
- If the irrelevant background can be better removed, higher prediction score can be obtained



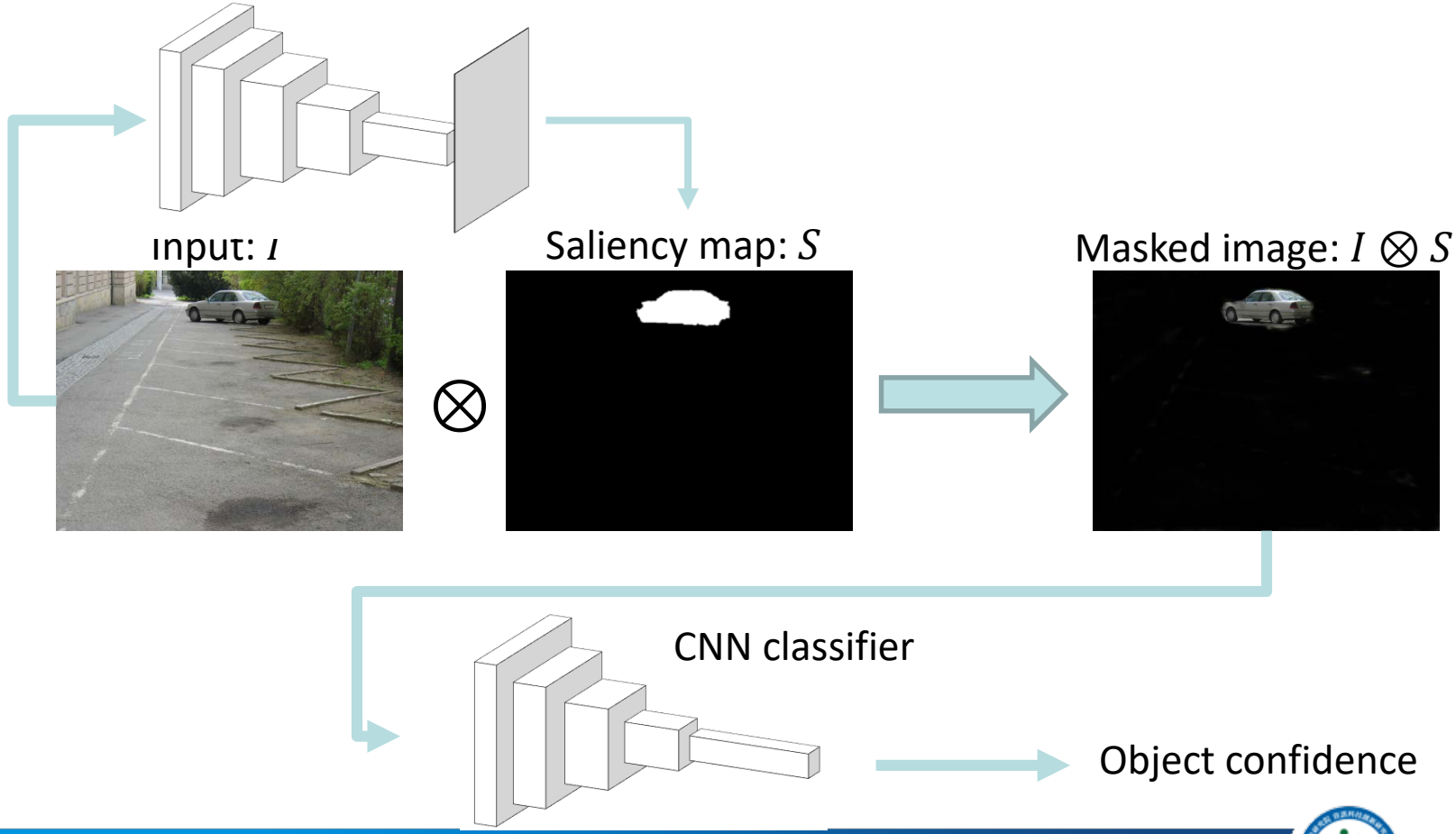
Observation

- Examples with scores shown on the top of the saliency maps



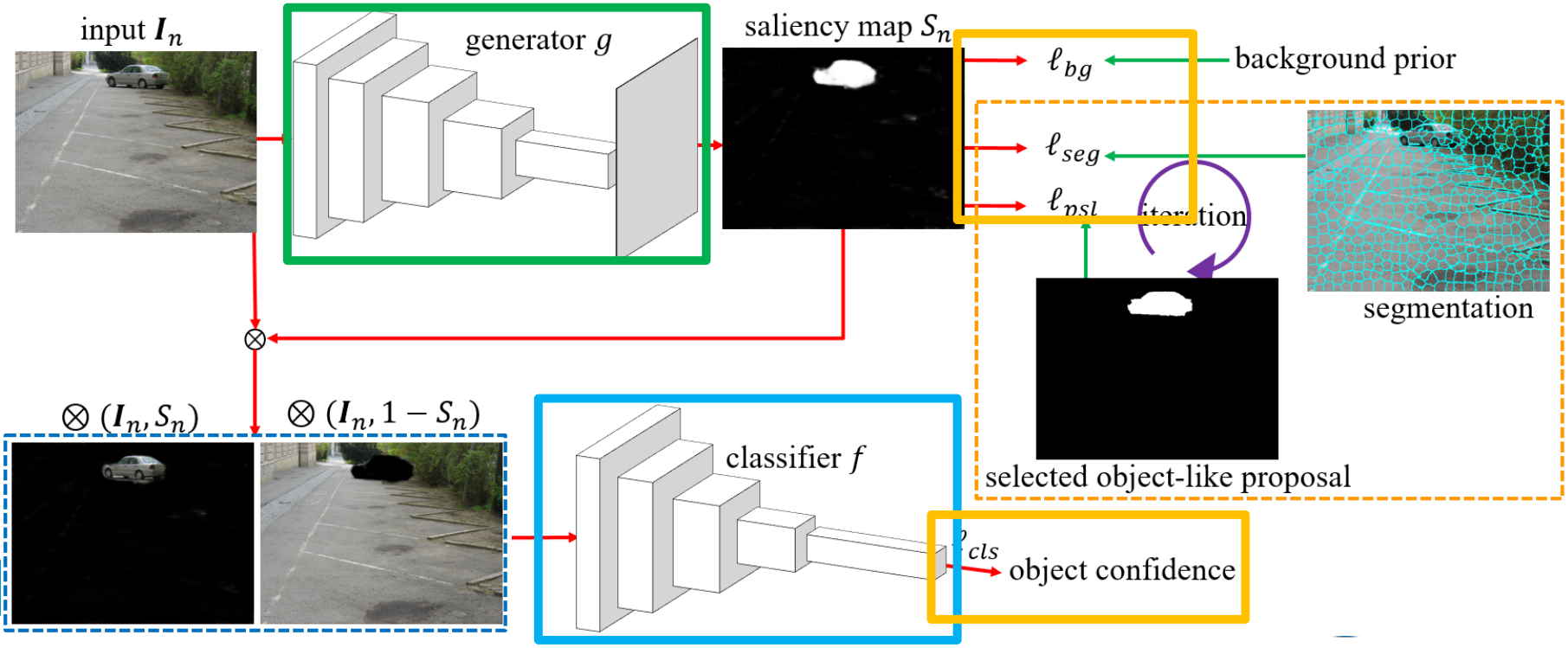
Idea

- Employ an FCN for generate saliency maps
- Let the trained a CNN classifier verify their qualities

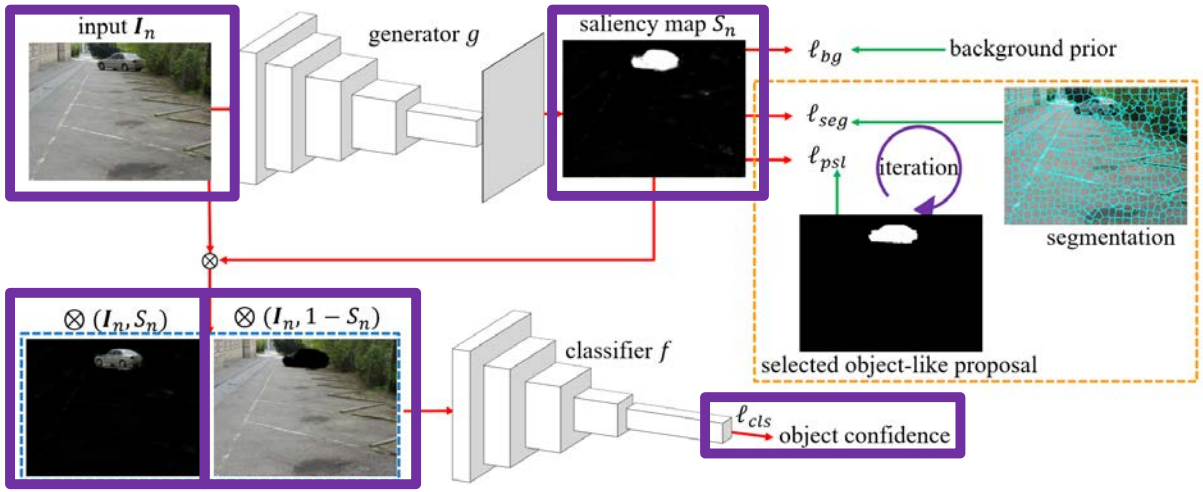


Approach

- Two collaborative CNN-based modules:
  - Image-level classifier
  - Pixel-level map generator
- Four loss functions: $\ell_{cls}, \ell_{bg}, \ell_{seg}, \ell_{pls}$



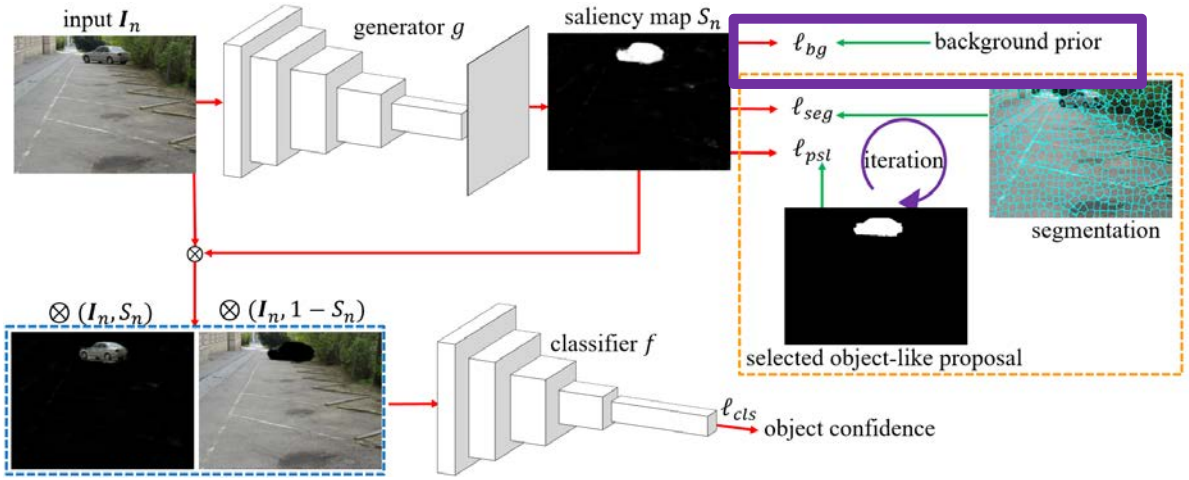
1st loss function: Classification loss



$$\ell_{cls}(I_n;\mathbf{w}) = \|f(\otimes(S_n, I_n)) - 1\|^2 + \|f(\otimes(1-S_n, I_n)) - 0\|^2$$

- Classification loss:
  - Highlight only discriminative regions

2nd loss function: Background loss

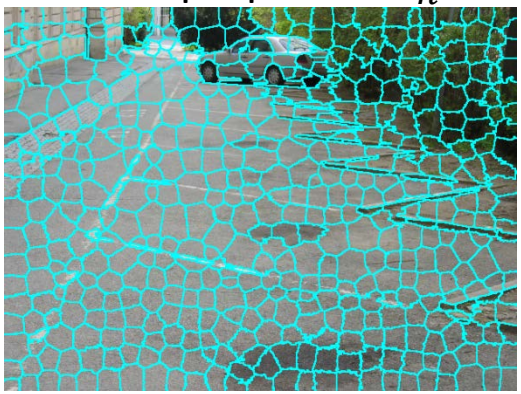


- Background loss: $$\ell_{bg}(I_n; \mathbf{w}) = \frac{1}{W \times H} \|S_n - \mathbf{0}\|^2$$
  - Reduce false alarms in saliency detection

3rd loss function: Segmentation loss

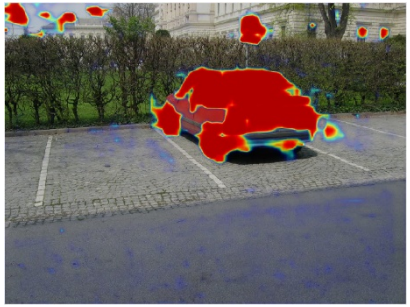
Input I_n

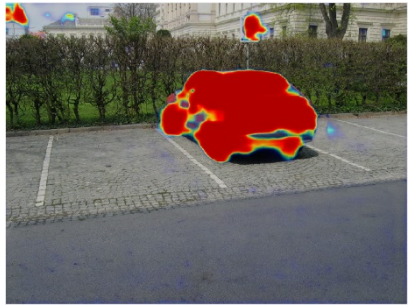
Superpixels M_n



Segmentation mask G_n

- Segmentation-based loss: $\ell_{seg}(I_n, M_n; \mathbf{w}) = \frac{1}{W \times H} \|S_n - G_n\|^2$
  - Reduce noise in a superpixel-wise manner

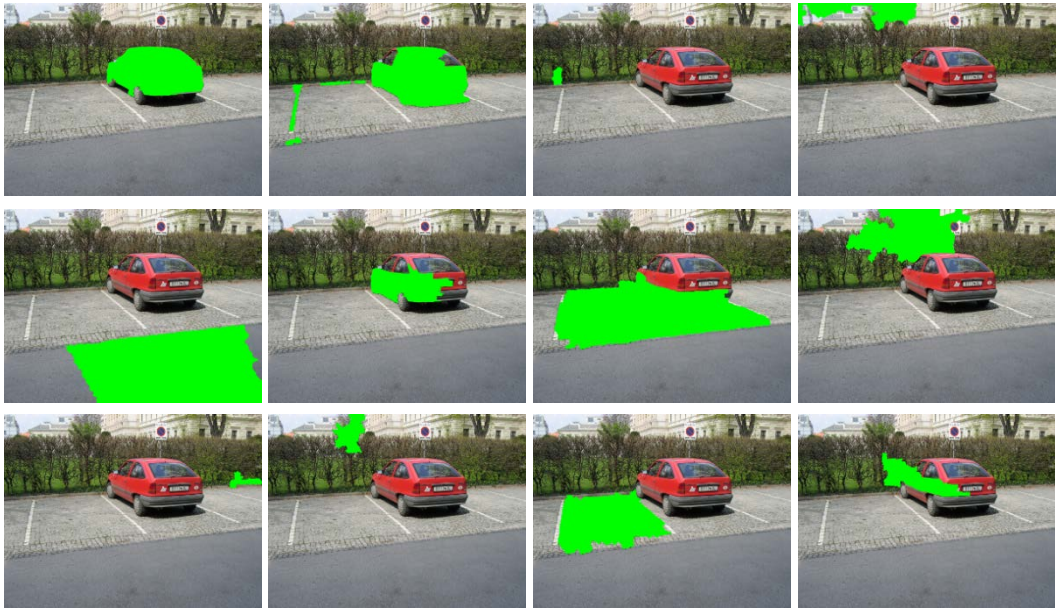




4th loss function: Proposal loss

- Object proposals: a set of object candidates
  - Each object is probably well-covered by at lease one proposal

Object proposals [Krahenbuhl and Koltun, ECCV'14]



4th loss function: Proposal loss

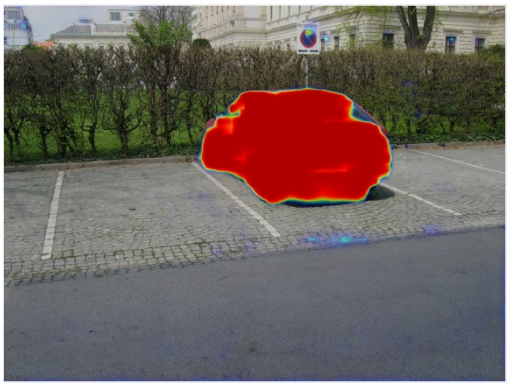
- Proposal loss: Select the most plausible proposal as ground truth

$$\ell_{psl}(I_n, O_n; \mathbf{w}) = \frac{1}{W \times H} \|S_n - O_n\|^2,$$
$$\text{where } O_n = \arg\min_{O \in \mathcal{O}_n} \|S_n - O\|^2.$$

- The object proposal covering the discriminative parts probably covers the non-discriminative parts at the same time

saliency map without ℓ_{psl}

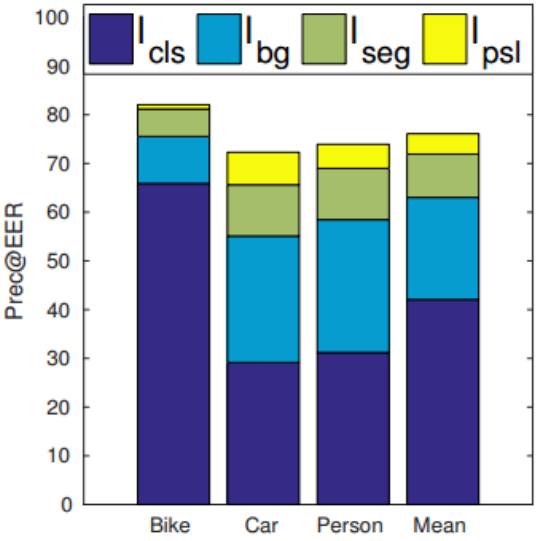
selected proposal



saliency map with ℓ_{psl}

Experimental results: Ablation study

- Test beds:
  - Graz02, PASCAL VOC-2007/2012
- Evaluation criterion: (Prec@EER)
  - Higher → Better



(a) Performance gain

(b) image

Experimental results on Graz02 dataset

- Prec@EER

Group	Method	Setting	Bike	Car	Person	Mean
Bottom-up	MB [ICCV'15]	US	54.7	39.0	52.0	48.6
	MST [CVPR'16]	US	50.1	38.8	51.3	46.7
	HDCT [CVPR'14]	FS	55.9	43.8	53.0	50.9
	DRFI [IJCV'16]	FS	51.3	49.6	59.6	53.5
Top-down	ILC [CVPR'10]	FS	71.9	64.9	58.6	65.1
	SP-Nei. [ICCV'09]	FS	72.2	72.2	66.1	70.2
	Shape mask [IJCV'12]	FS	61.8	53.8	44.1	53.2
	Patch-CRF [CVPR'12]	FS	62.4	60.0	62.0	61.3
	SP-CRF [BMVC'14]	FS	73.9	68.4	68.2	70.2
	LCCSC [BMVC'15]	FS	76.2	71.2	64.1	70.5
	R-ScSPM [CVPR'16]	FS	77.6	71.9	67.0	72.1
	R-ScSPM [CVPR'16]	WS	67.5	56.5	57.6	60.5
	CNN-Gen. [BMVC'17]	WS	78.9	66.6	64.2	69.9
	Ours	WS	**82.1**	**78.5**	**75.0**	**78.5**

Experimental results on Pascal VOC

- Prec@EER

Method	Setting	Avg.	A.P.	Bike.	Bird	Boat	Bottle.	Bus	Car	Cat	Chair	Cow	D.T.	Dog	Horse	M.B.	P.S.	P.P.	Sheep	Sofa	Train	TV
Patch-CRF [CVPR'12]	FS	15.6	14.7	28.1	9.8	6.1	2.2	24.1	30.2	17.3	6.2	7.6	10.3	11.5	12.5	24.1	36.7	2.2	20.4	12.3	26.1	10.2
SP-CRF [BMVC'14]	FS	40.4	46.5	45.0	33.1	**60.2**	25.8	48.4	31.4	64.4	19.8	32.2	**44.7**	30.1	41.8	72.1	33.0	40.5	38.6	12.2	64.6	23.6
Examplar [CVPR16]	FS	56.2	55.9	37.9	45.6	43.8	**47.3**	**83.6**	57.8	69.4	22.7	**68.5**	37.1	72.8	63.7	69.0	57.5	**43.9**	66.6	38.3	**75.1**	**56.7**
GMP [CVPR15]	WS	48.1	48.9	42.9	37.9	47.1	31.4	68.4	39.9	66.2	**27.2**	54.0	38.3	48.5	56.5	70.1	43.2	42.6	52.2	34.8	68.1	43.4
Exc. BP [ECCV16]	WS	45.3	50.7	32.5	48.4	30.2	36.8	59.3	36.6	54.4	21.6	57.6	40.4	59.0	47.5	61.4	48.4	28.7	57.5	35.8	48.7	51.5
CNN-Gen. [BMVC'17]	WS	50.0	64.5	46.7	50.2	29.6	0.0	75.3	60.1	73.4	16.0	39.5	40.9	**81.8**	59.9	72.5	72.0	37.6	58.9	45.3	43.5	32.9
Ours	WS	**56.8**	**71.6**	**47.7**	**64.6**	32.4	0.0	77.8	**69.4**	**81.9**	19.5	48.9	39.9	76.8	**71.3**	**75.0**	**87.4**	42.0	**75.8**	**67.8**	54.0	32.1

- Run time

Method	MB [ICCV'15]	MST [CVPR'16]	Patch-CRF [CVPR'12]	SP-CRF [BMVC'14]
Time (Sec)	0.0263	0.1142	3.094	30.2928
Speedup	1.742×	7.563×	204.9×	2006×

Method	Examplar [CVPR16]	CNN-Gen. [BMVC'17]	Exc. BP [ECCV'16]	Ours
Time (Sec)	2.1470	5.295	0.0632	**0.0151**
Speedup	142.2×	348.9×	4.185 ×	**1**×

Visualization results on Graz02 dataset

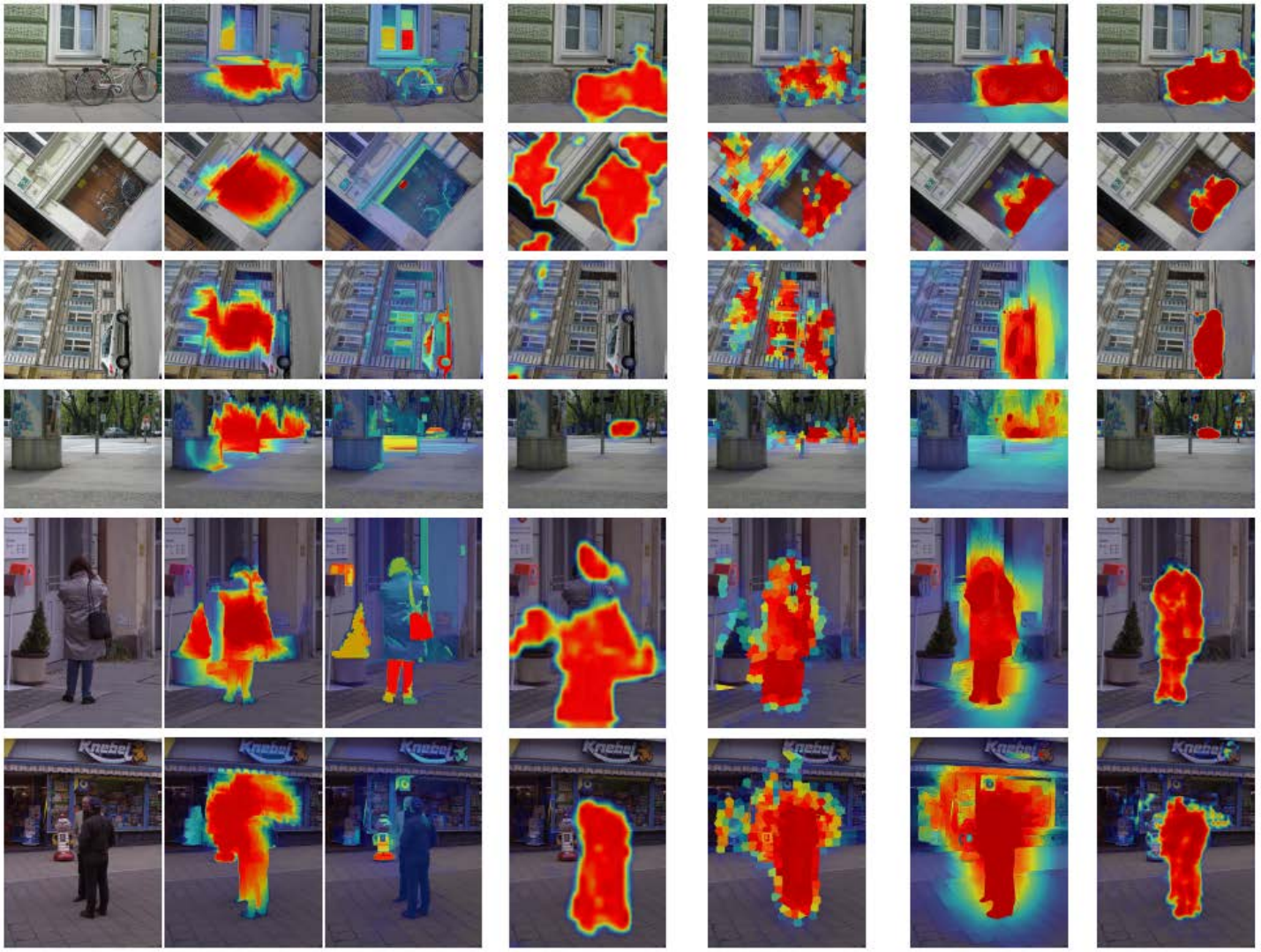


Image MB [ICCV'15] DRFI [IJCV'16] Patch-CRF [CVPR'12] SP-CRF [BMVC'14] CNN-Gen. [BMVC'17] Ours

Visualization results on Pascal VOC

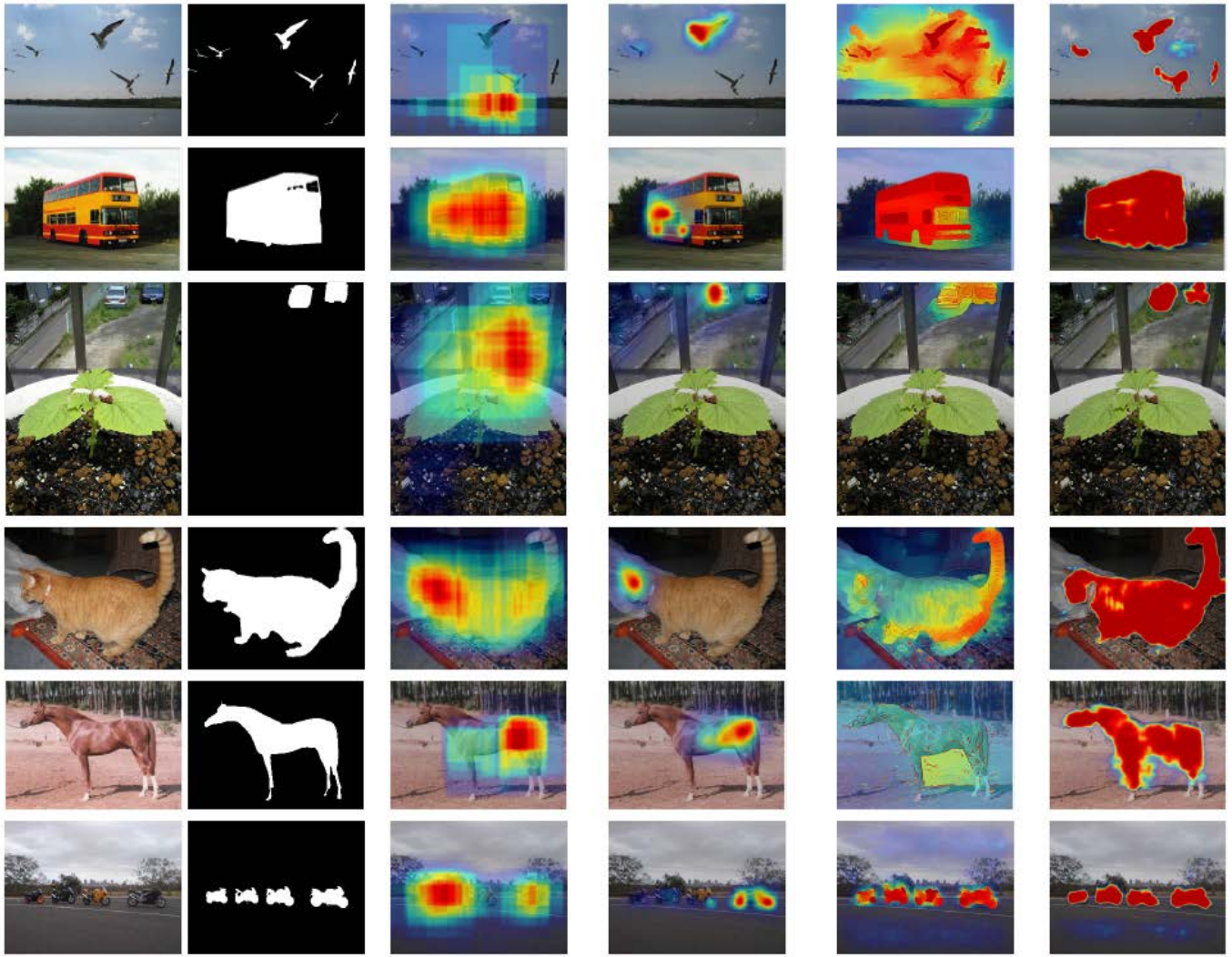


Image Mask Examplar [CVPR'16] Exc. BP. [ECCV'16] CNN-Gen. [BMVC'17] Ours

Outline

- Convolutional neural networks (CNNs)
- Representative CNN models
- CNN-based computer vision applications
  - Find-grained object recognition
  - Object detection
  - Semantic segmentation
  - Image super resolution
  - Saliency detection
  - Image style transfer
  - Action and gesture recognition
  - Image matching and alignment

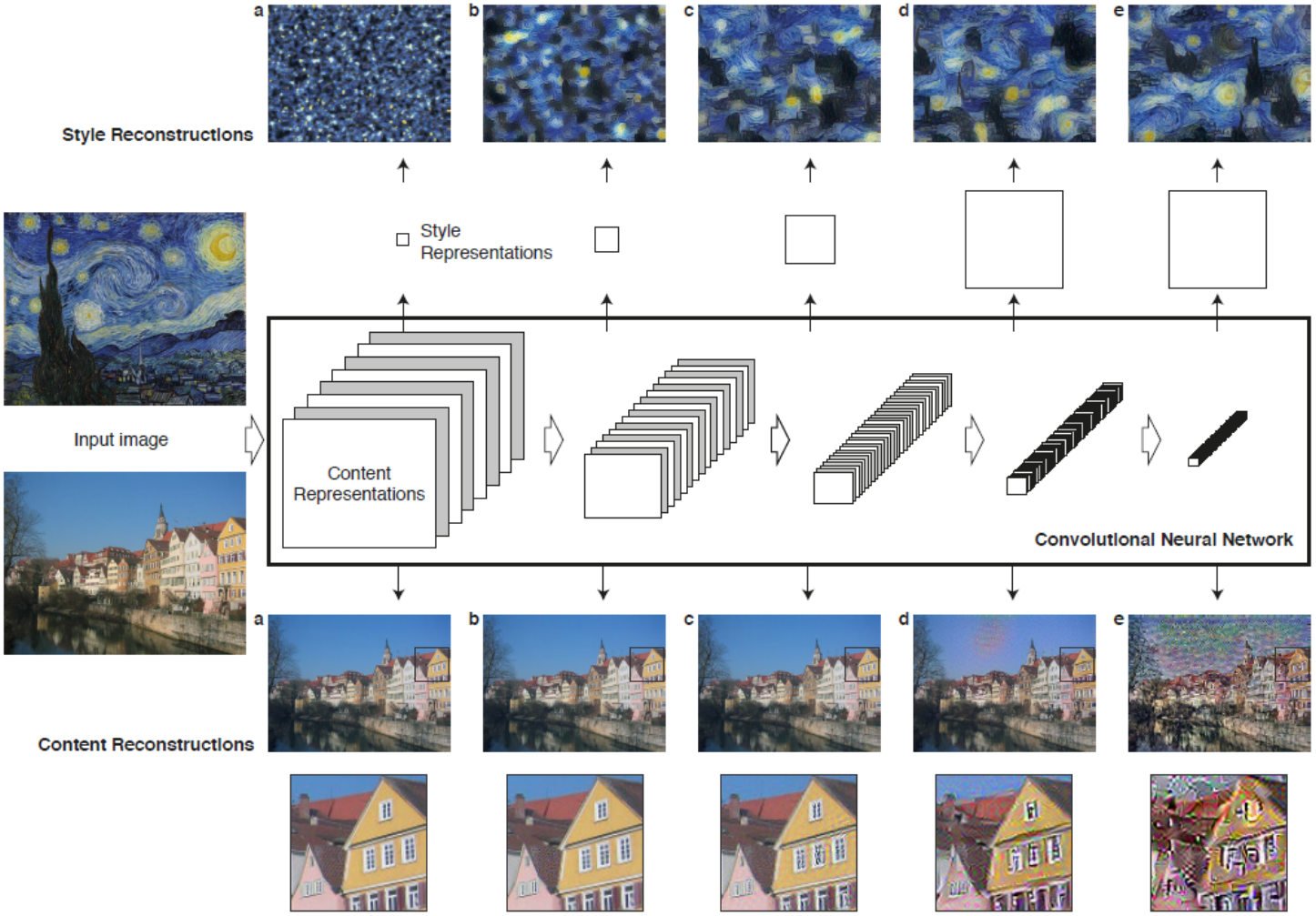
Image Style Transfer

- Given a **content image** and a **style image**, generate a new image that
  - Matches the CNN features of the content image
  - Matches the Gram matrices of the style image

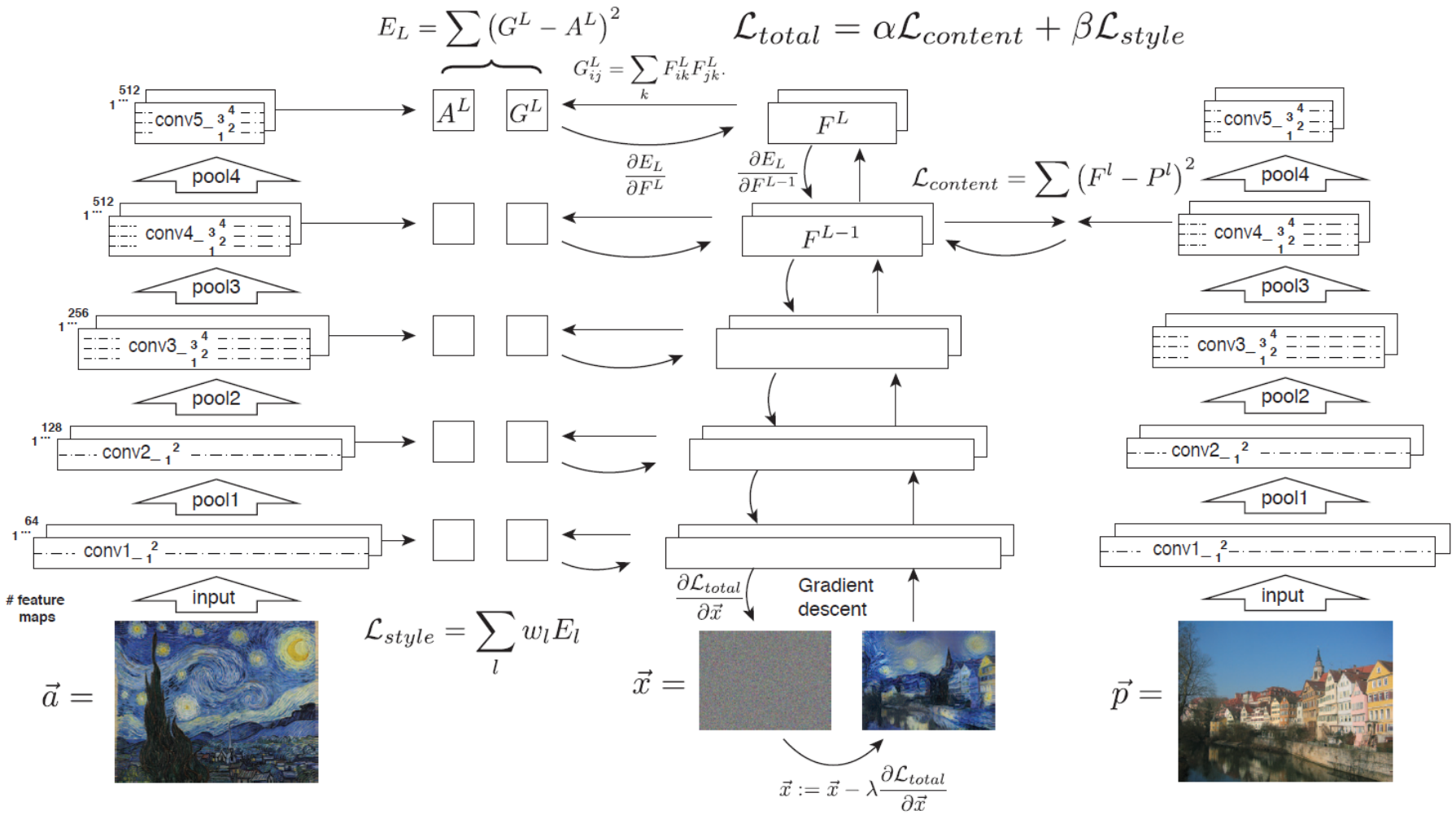
content image style image stylized image

Image Style Transfer using CNNs

[Gatys et al, CVPR'16]



Network Architecture



Experimental Results

Outline

- Convolutional neural networks (CNNs)
- Representative CNN models
- CNN-based computer vision applications
  - Find-grained object recognition
  - Object detection
  - Semantic segmentation
  - Image super resolution
  - Saliency detection
  - Image style transfer
  - Action and gesture recognition
  - Image matching and alignment

Action Recognition

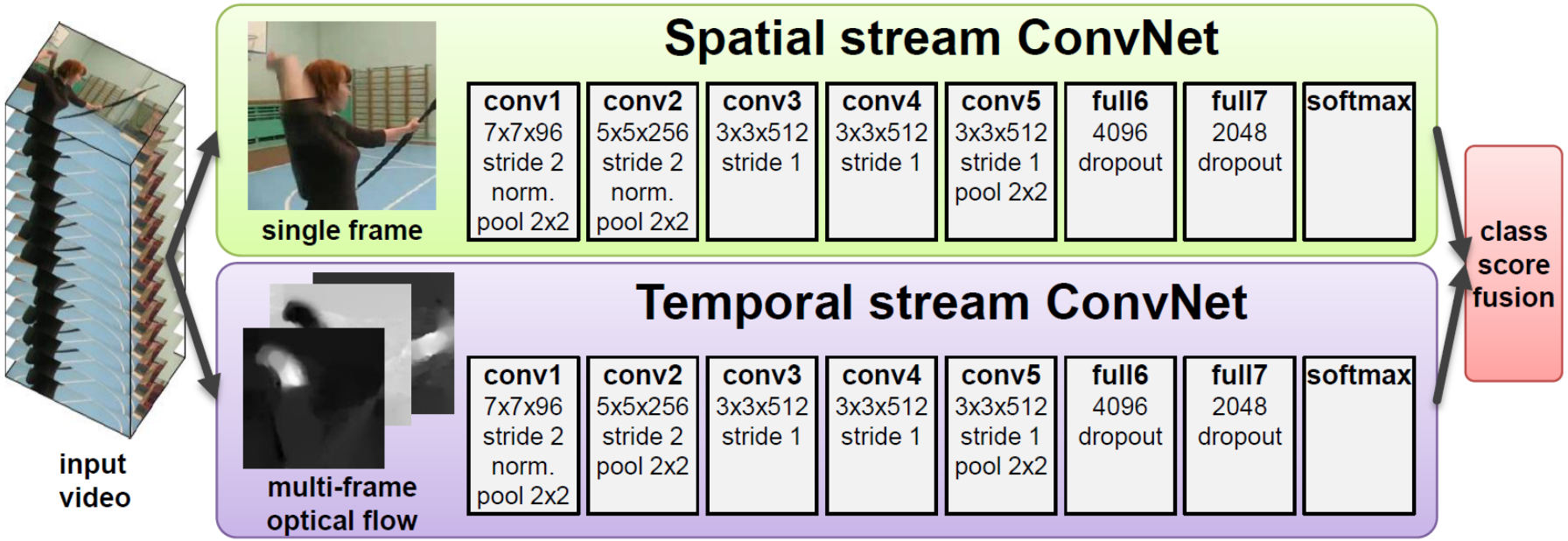
- Predict the behavior of a person in a video sequence

UCF Sports 101 Dataset

Two-Stream CNNs for Action Recognition

[Simonyan and Zisserman, NIPS'14]

- A two-stream CNN architecture
  - Spatial stream networks: Single-frame RGB image
  - Temporal stream networks: Multi-frame dense optical flow



Experimental Results

- Recognition rates on two benchmarks: HMDB 51 and UCF 101

Method	UCF-101	HMDB-51
Improved dense trajectories (IDT) [26, 27]	85.9%	57.2%
IDT with higher-dimensional encodings [20]	**87.9%**	61.1%
IDT with stacked Fisher encoding [21] (based on Deep Fisher Net [23])	-	**66.8%**
Spatio-temporal HMAX network [11, 16]	-	22.8%
"Slow fusion" spatio-temporal ConvNet [14]	65.4%	-
Spatial stream ConvNet	73.0%	40.5%
Temporal stream ConvNet	83.7%	54.6%
Two-stream model (fusion by averaging)	86.9%	58.0%
Two-stream model (fusion by SVM)	**88.0%**	**59.4%**

Learning Adaptive Hidden Layers for Mobile Gesture Recognition

[Hu et al., AAAI'18]

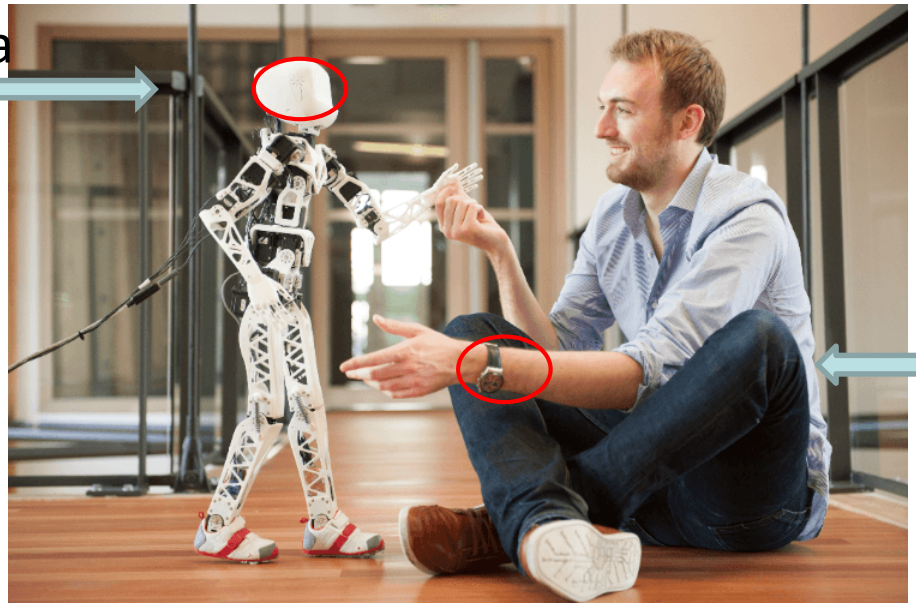
- Gesture recognition serves an intuitive and convenient way for human-machine interaction in various environments.
- Accurate gesture recognition becomes quite challenging due to variations caused by
  - Diverse user behaviors
  - Ubiquitous environments

source: Chalearn Gesture challenge

Introduction

- Such challenges can be potentially resolved by using extra information from sensing modalities.

camera



smart-watch

Our goal is to leverage multi-modal signals captured by diverse mobile sensors.

source: sculpteo

Outline

- Introduction
- Observation & Motivation
- Related Work
- Approach
- Experimental results
- Conclusions

Observation & Motivation – (1)

- Observations
  - Optimal modality typically varies from gesture to gesture
  - Mobile contexts suffer from large modality-specific environment variations

We address these issues by predicting the output with the model conditioned on the input. (adaptive learning method)

Observation & Motivation – (2)

- Adaptive learning methods
  - Pro: Robust to both intra-modal & inter-modal variations
  - Con: A large number of parameters

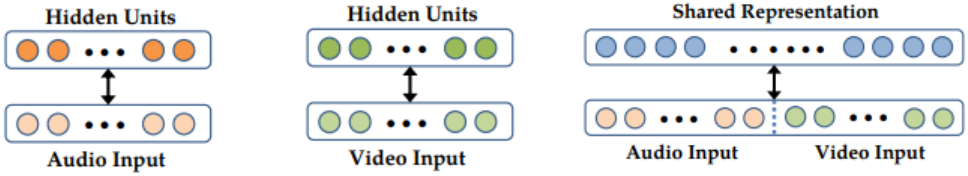
- We introduce a new adaptive learning method with linearly increased number of parameters.

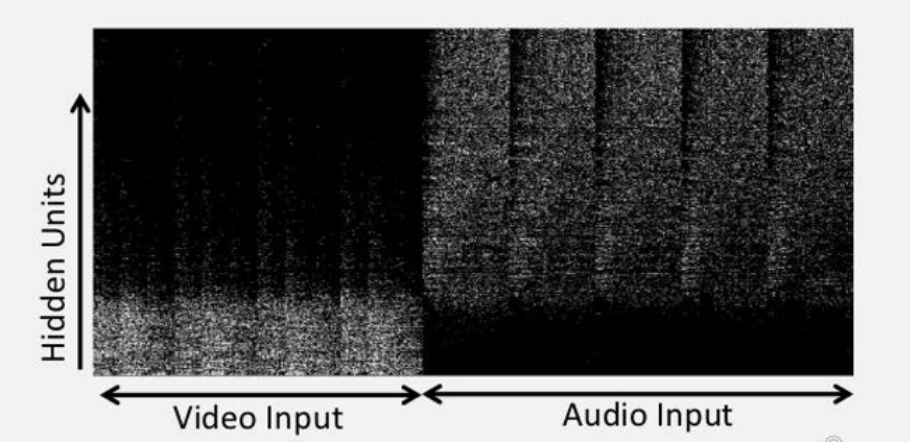
Outline

- Introduction
- Observation & Motivation
- Related Work
- Approach
- Experimental results
- Conclusions

Related Work – (1)

- Concatenation: a naïve solution to multi-modal learning



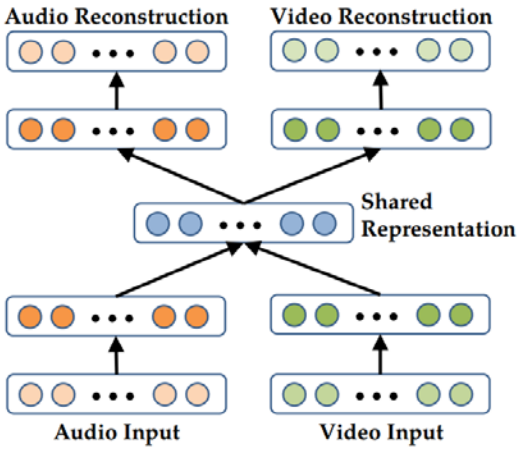


Only unimodal features are learned!

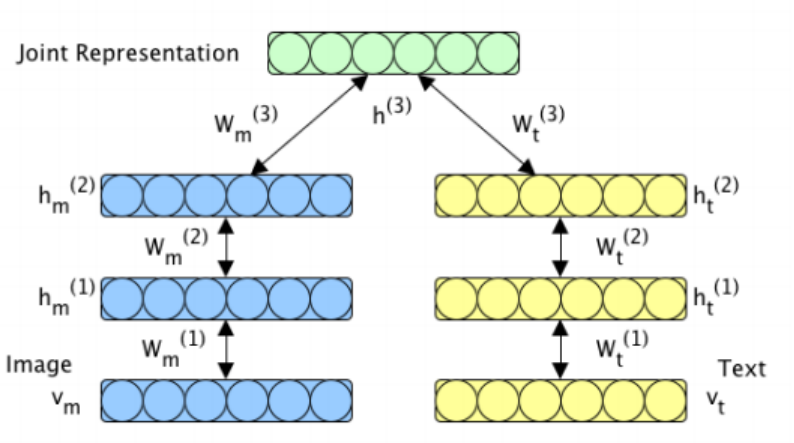
[Ngiam et al. ICML'11]

Related Work – (2)

- These works learn modality-specific networks first followed by learning joint network.
- These approaches seek an immutable combination of multi-modal information.



[Ngiam et al., ICML'11]

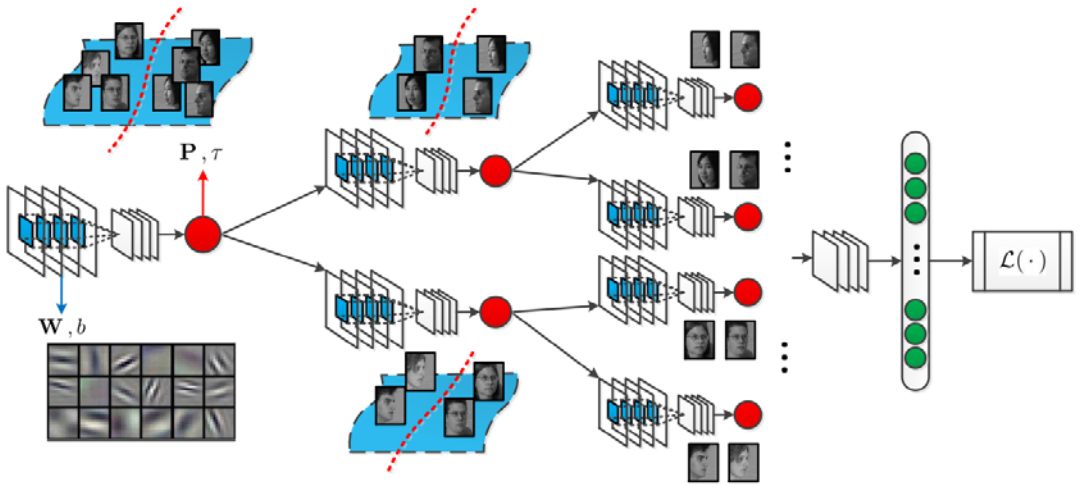


[Srivastava et al., NIPS'12]

Related work – (3)

- This work learns tree-structured sub-networks for adaptive learning.
- It provides exponentially many sub-networks with exponentially increased number of parameters.

We introduce an adaptive learning method with exponentially many paths and linearly increased number of parameters.



[Xiang et al., ICCV'15]

Outline

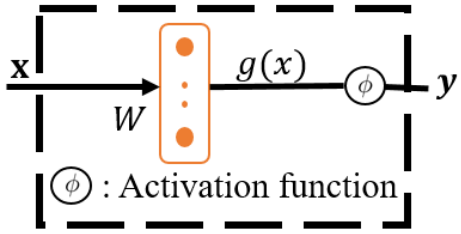
- Introduction
- Observation & Motivation
- Related Work
- Approach
- Experimental results
- Conclusions

Approach

- We present a new network layer, called adaptive hidden layer (AHL).
  - It generalizes a hidden layer in DNNs and can dynamically generate an appropriate activation map for a given input.

- An AHL consists of
  - Multiple neuron groups
  - One extra selector

Forward pass

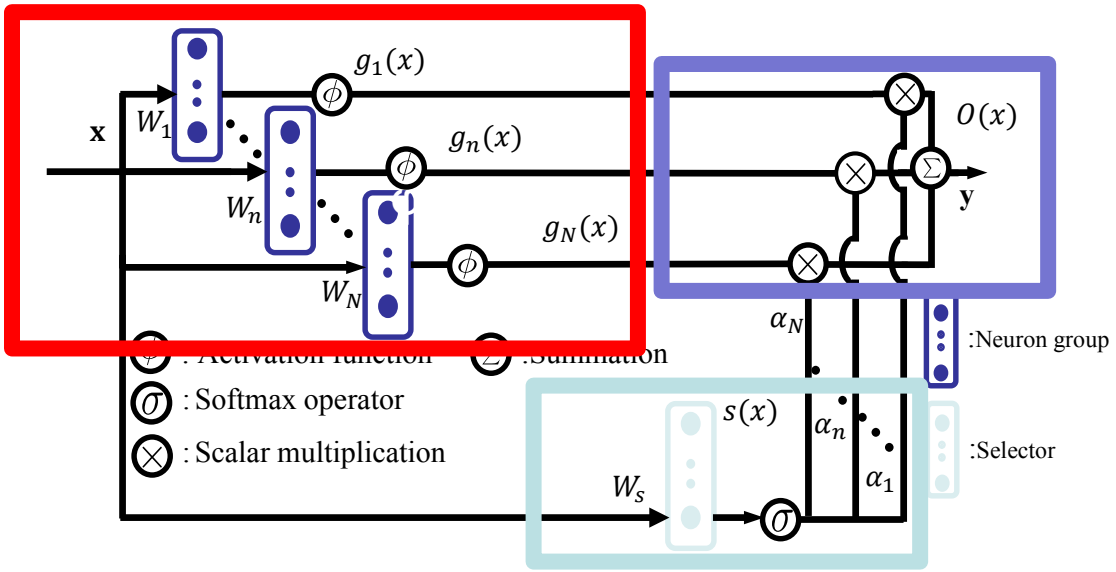


$$y = \phi(W * x + b)$$

$$g_n(x) = \phi(W_n * x + b_n) \quad \text{for } n = 1,2,\dots,N$$

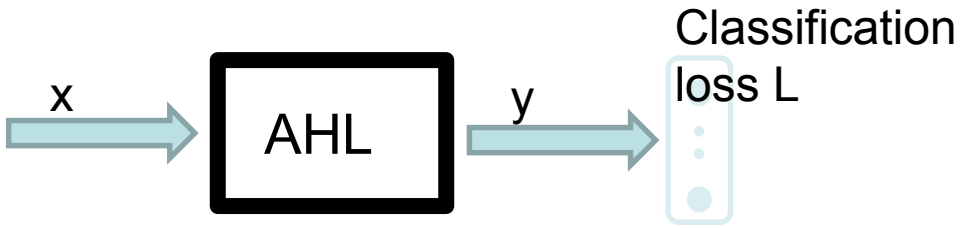
$$\alpha = \sigma(W_s * x + b_s)$$

$$y = \sum_{n=1}^{N} \alpha_n g_n(x) \in \mathbb{R}^M$$



Backward propagation

- Backward propagation is a widely used method for optimizing multi-layer neural networks .
- By ordinary back propagation, we need $\frac{\partial l}{\partial W_n} = \frac{\partial l}{\partial y}\frac{\partial y}{\partial W_n}$ and $\frac{\partial l}{\partial W_s} = \frac{\partial l}{\partial y}\frac{\partial y}{\partial W_s}$.
  - l is the classification loss.
- $\frac{\partial l}{\partial y}$ can be computed by using the chain rule.



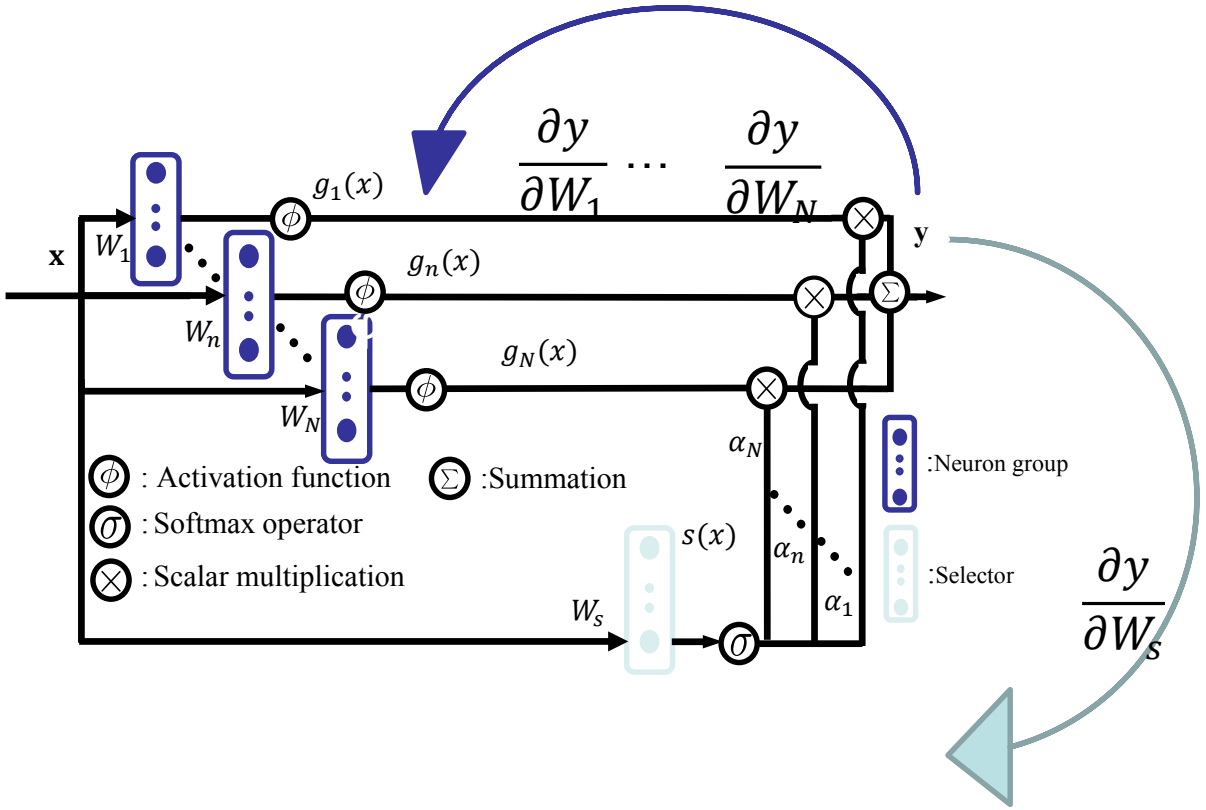
Backward propagation

$$\frac{\partial y}{\partial W_n} = \frac{\partial \sum_{i=1}^{N} \alpha_i g_i(x)}{\partial W_n} = \alpha_n \frac{\partial g_n(x)}{\partial W_n}$$

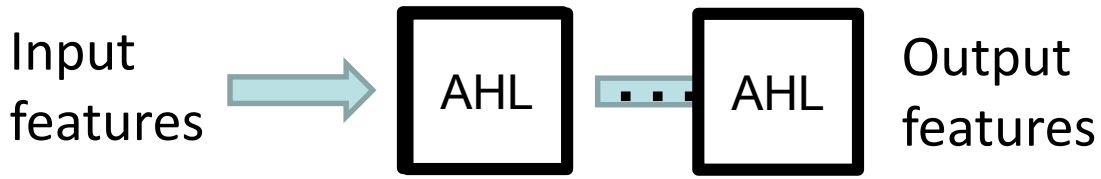
$$\frac{\partial y}{\partial W_s} = \frac{\partial \sum_{i=1}^{N} \alpha_i g_i(x)}{\partial W_s} = \sum_{i=1}^{N} \frac{\partial \alpha_i}{\partial W_s} g_i(x)$$

$$= \sum_{i=1}^{N} \frac{\partial \alpha_i}{\partial s(x)} \frac{\partial s\,(x)}{\partial W_s} g_i(x)$$

$$= \sum_{i=1}^{N} \frac{\partial \alpha_i}{\partial s(x)} \frac{\partial\,(W_s * x + b_s)}{\partial W_s} g_i(x)$$



Generalization - stacking AHLs



- Scalable: it provides exponentially many forward paths with linearly increased number of parameters.

The Adaptive Hidden Layer

- Differentiable: the resultant network is end-to-end trainable.
- Scalable: it provides exponentially many forward paths with linearly increased number of parameters.
- Robust: Exponentially many paths tackles intra-modal and inter-modal variations.

Outline

- Introduction
- Observation & Motivation
- Related Work
- Approach
- Experimental Results
- Conclusions

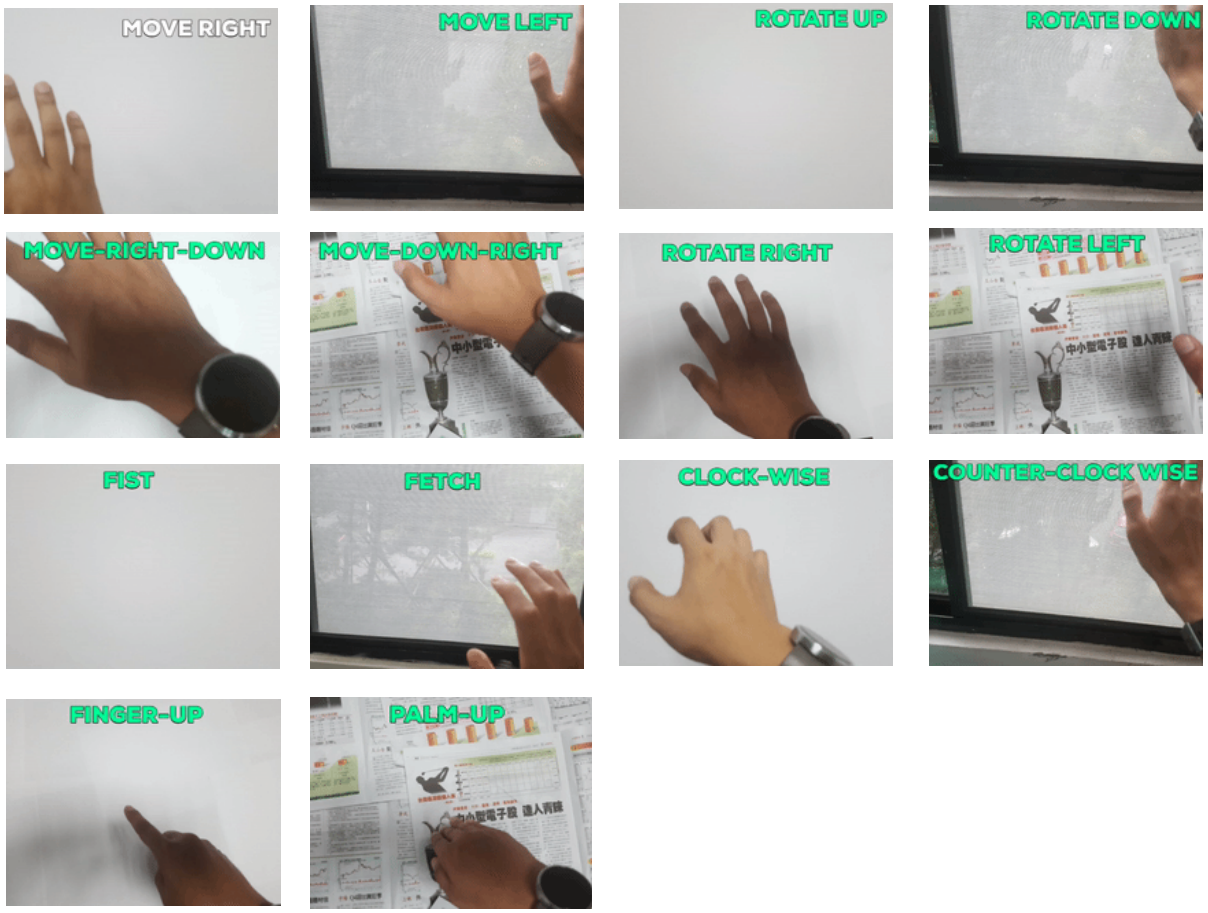
Datasets – (1)

- IsoGD dataset
  - Multi-modal - RGB and Depth data
  - Large scale - 47749 gestures and 249 labels
  - User Independent: the uses in the training set will not appear in the validation set.

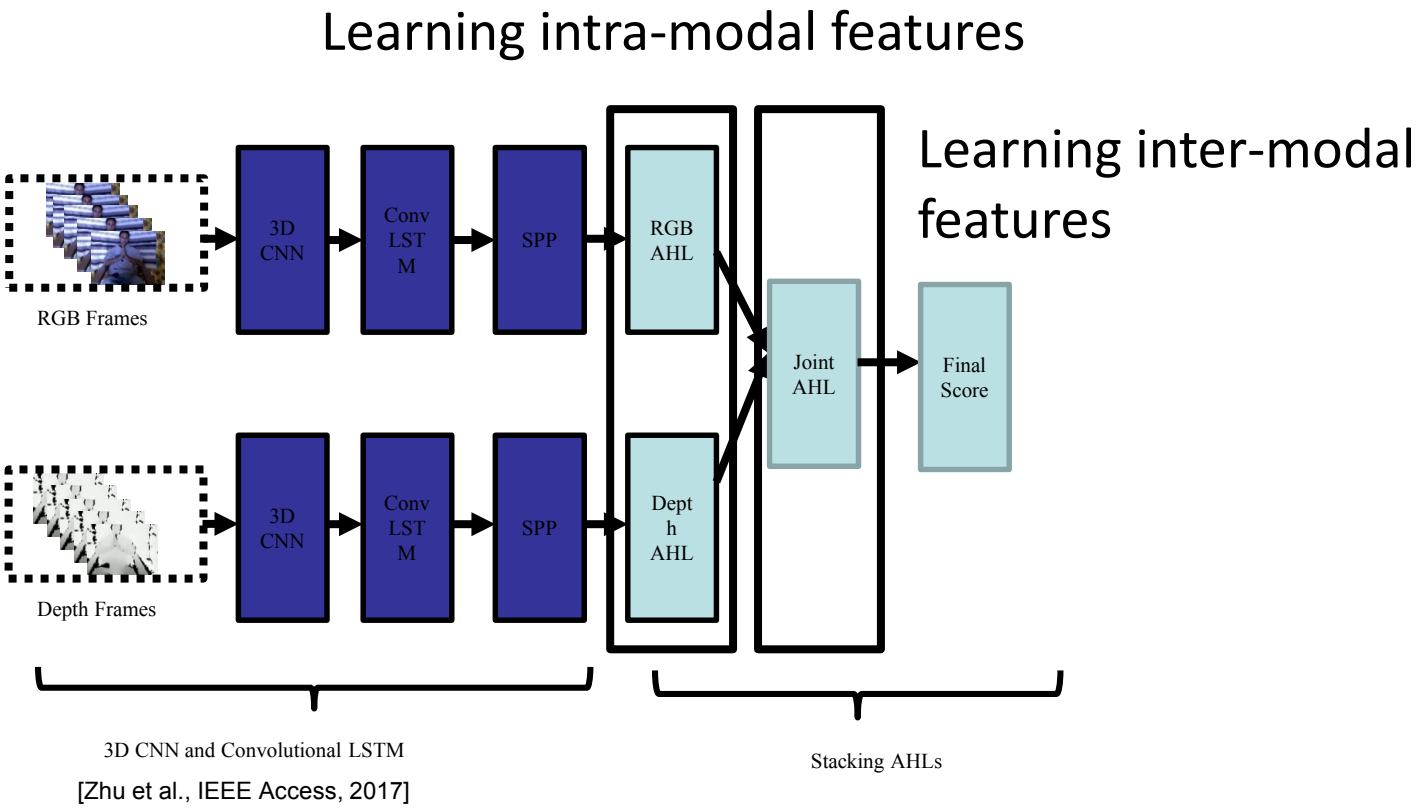
Datasets – (2)

- Our collected dataset
  - Multi-modal: Visual and Motion data
  - Four types of environmental variations are included.
    - Two types of variations are related to the visual modality.
      - Lighting condition: sufficient versus poor
      - Background: clean versus noise
    - The other two types of variations are related to the motion modality.
      - Angle of the performing plane: horizontal versus vertical
      - The degree of wear tightness: tight versus loose

Datasets – (3)

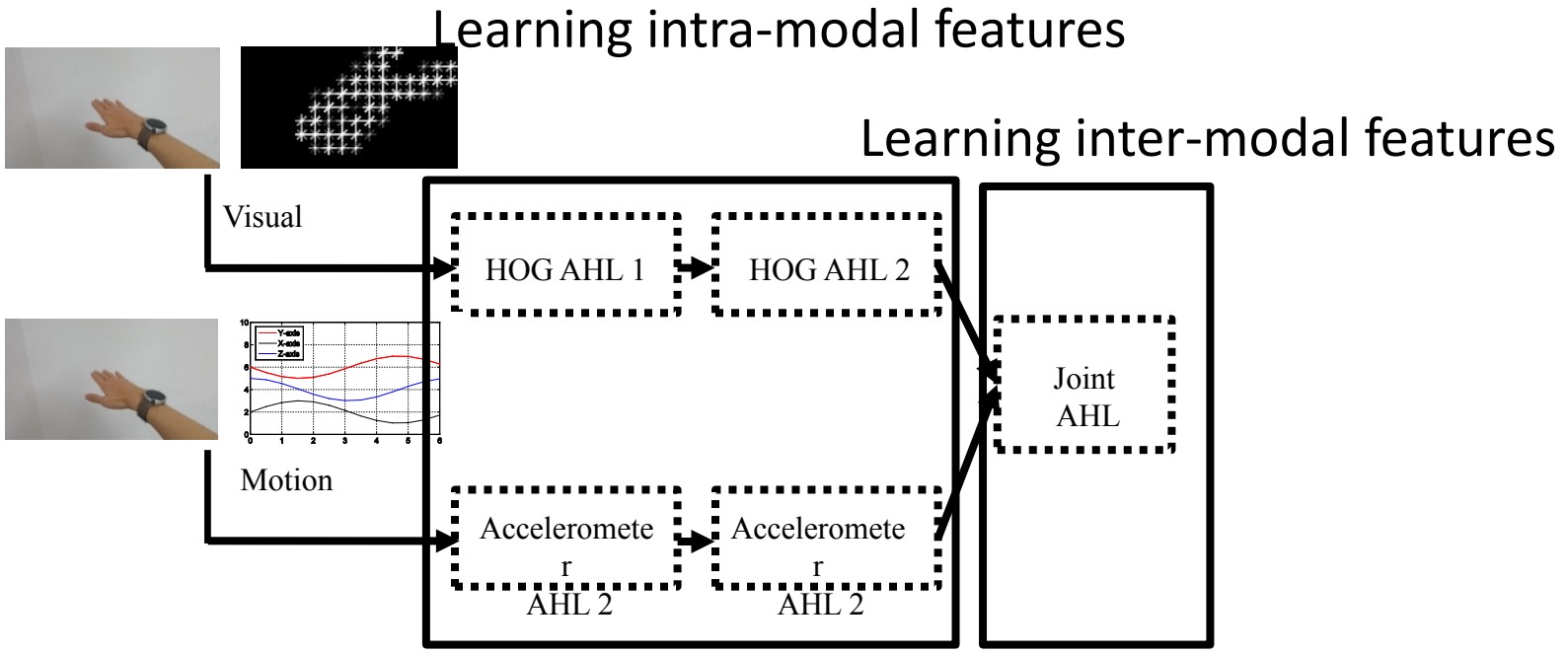

MOVE RIGHT
MOVE LEFT
ROTATE UP
ROTATE DOWN
MOVE-RIGHT-DOWN
MOVE-DOWN-RIGHT
ROTATE RIGHT
ROTATE LEFT
FIST
FETCH
CLOCK-WISE
COUNTER-CLOCK WISE
FINGER-UP
PALM-UP

Resultant network – IsoGD dataset



[Zhu et al., IEEE Access, 2017]

Resultant network – Our dataset



Performance evaluation – (1)

approach	accuracy
C3D (RGB only)	37.30%
C3D+ConvLSTM (RGB only)	43.88%
DEEP C3D+ConvLSTM (RGB only)	43.56%
ours (RGB only)	**44.88%**
C3D (Depth only)	40.50%
C3D+ConvLSTM (Depth only)	44.66%
DEEP C3D+ConvLSTM (Depth only)	47.99%
ours (Depth only)	**48.96%**
C3D (RGB+Depth)	49.20%
C3D+ConvLSTM (RGB+Depth)	51.02%
DEEP C3D+ConvLSTM (RGB+Depth)	51.40%
ours (RGB+Depth)	**54.14%**

Table 2: Recognition rates on the IsoGD dataset.

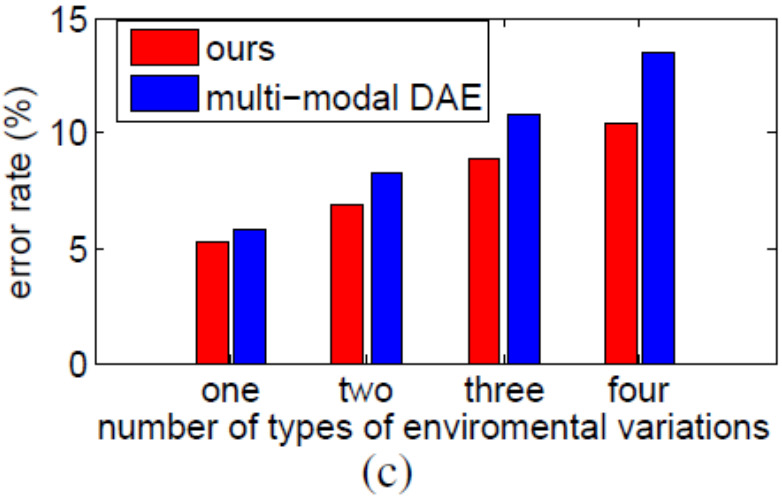
Performance evaluation – (2)

approach	accuracy
DAE + HOG	81.52%
DAE + ACCE	76.24%
DAE + conc. feat.	82.34%
multi-modal DAE	86.48%
double-sized multi-modal DAE	86.02%
ours	**90.57%**

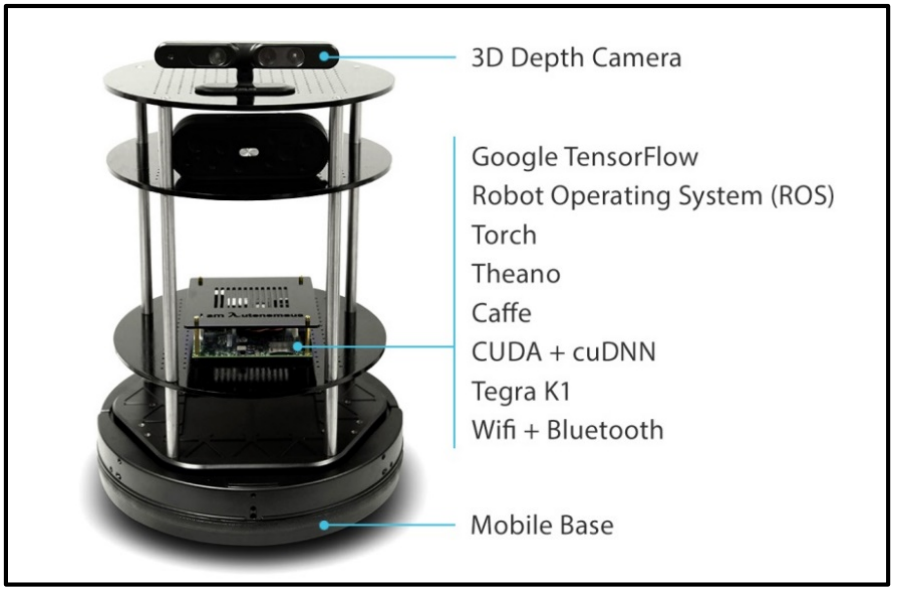
Table 3: Recognition rates on the dataset we collected.

Effect of Data Variations

- We adjust the number of variation types to control the degrees of data variations.
- Adaptive multi-modal learning enabled by our approach is crucial in mobile applications where large variations are frequently present.



Prototype Systems



Outline

- Introduction
- Observation & Motivation
- Related Work
- Approach
- Experimental results
- Conclusions

Conclusions

- We have presented a new network layer, called the adaptive hidden layer (AHL), which is composed of multiple neuron groups and an extra selector.

- The experimental results on two datasets show that by stacking multiple AHLs, our approach outperforms all competing methods.

Outline

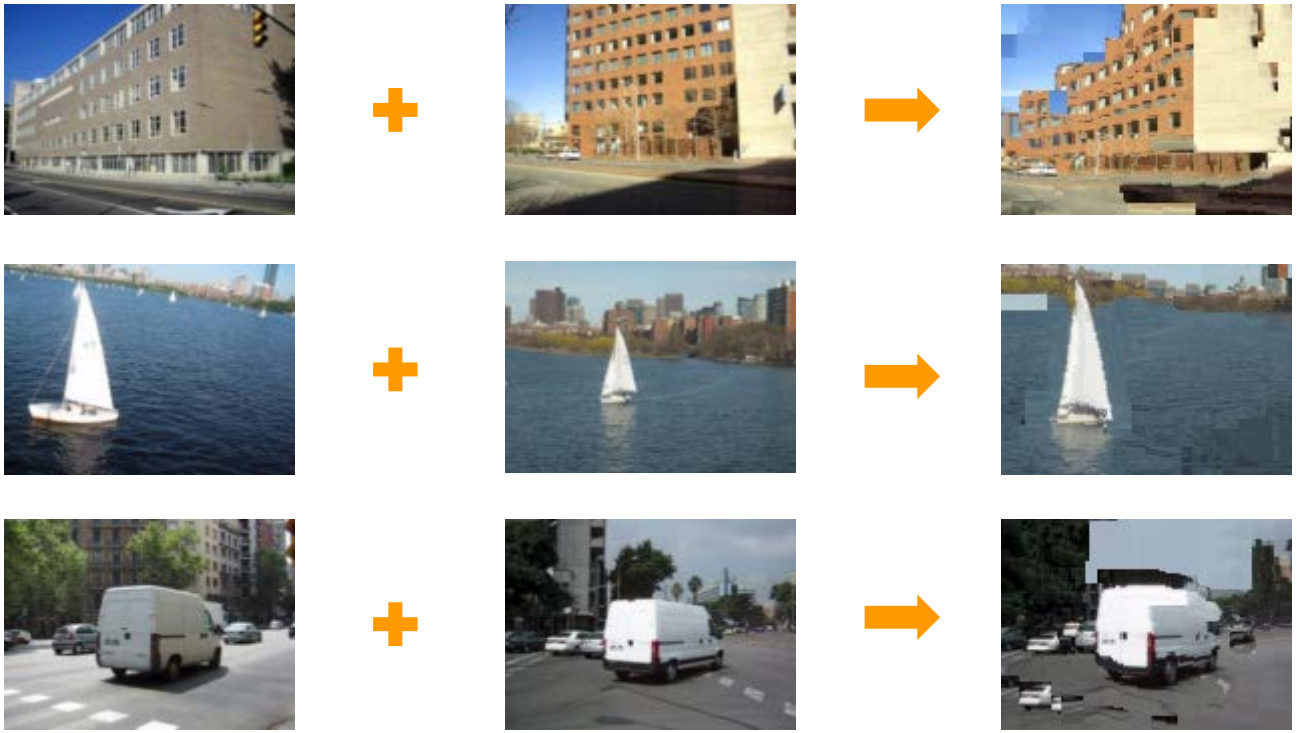
- Convolutional neural networks (CNNs)
- Representative CNN models
- CNN-based computer vision applications
  - Find-grained object recognition
  - Object detection
  - Semantic segmentation
  - Image super resolution
  - Saliency detection
  - Image style transfer
  - Action and gesture recognition
  - Image matching and alignment

Image Matching

- Seek common objects and regions between images
- Conventional pipeline:
  - Salient point detection + Description + Matching

Image Alignment

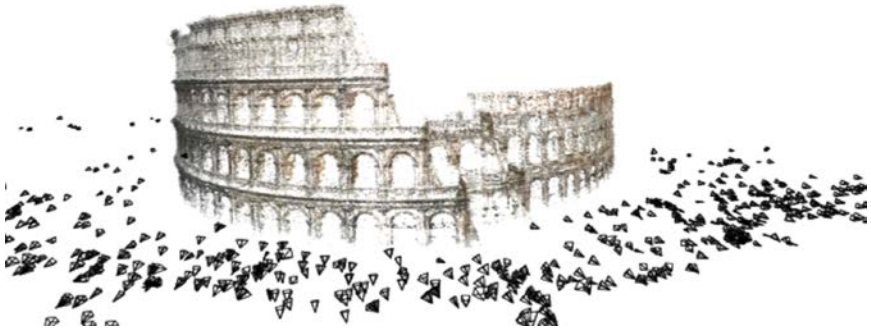
- Image matching: sparse matching on interest points
- Image alignment: ***dense*** matching on pixels



[Liu et al., PAMI'11]

Various Applications

3D Reconstruction



Agarwal et al. 2009

Image Retrieval

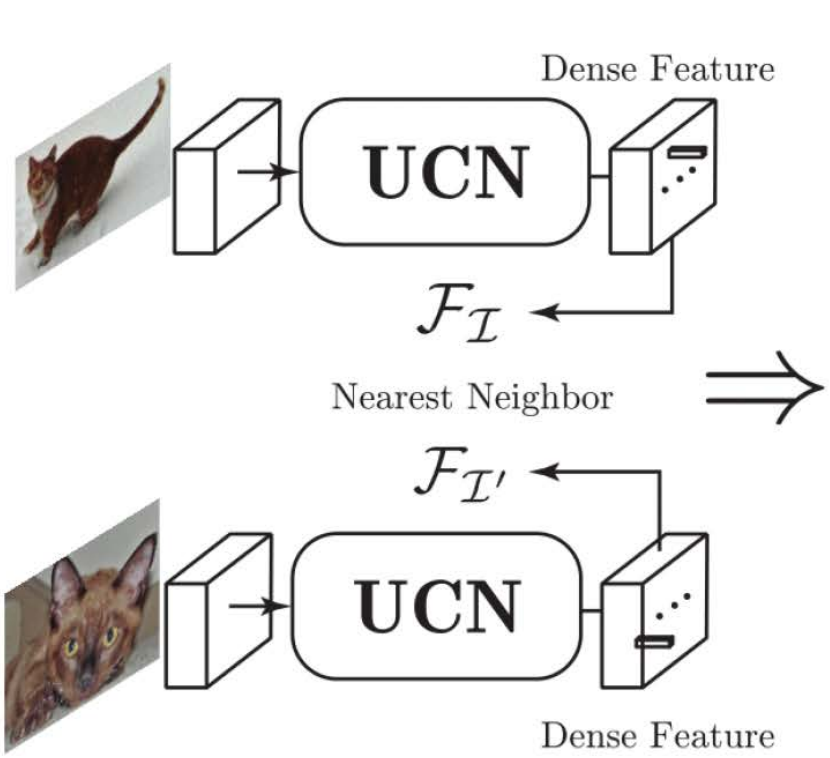
Zheng et al. 2015

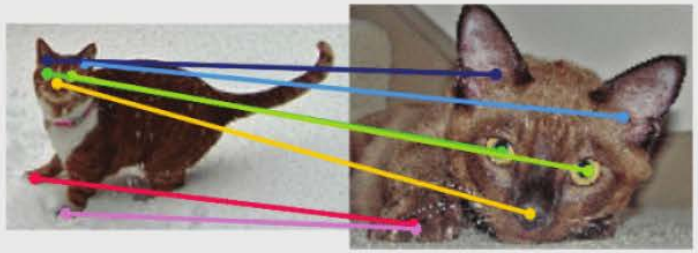
Panorama Stitching

Brown and Lowe 2007

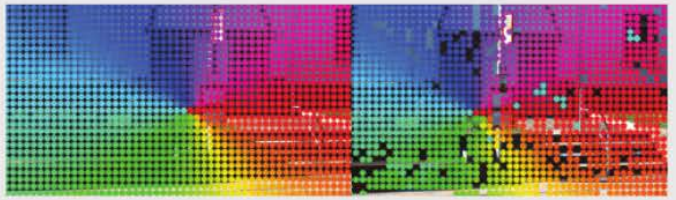
Universal Correspondence Network (UCN)

[Choy et al, NIPS'16]





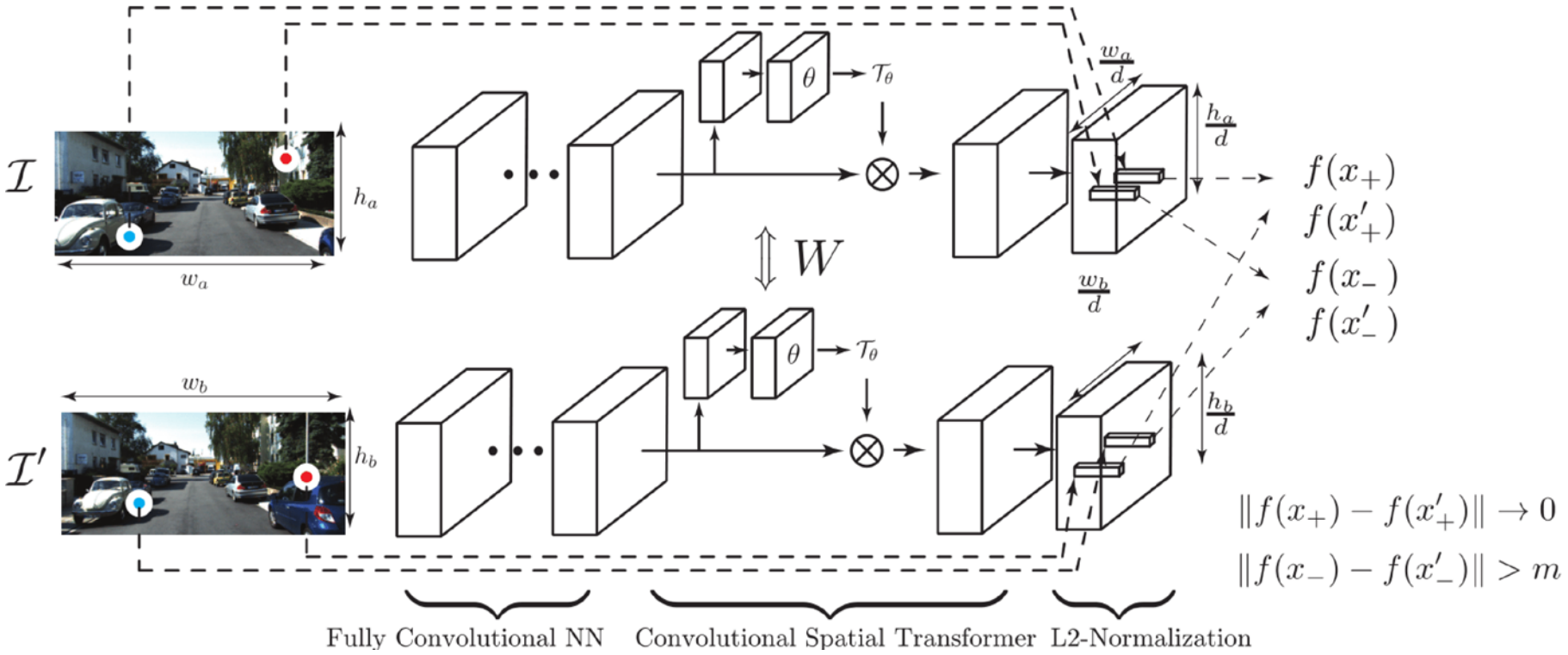
Accurate semantic correspondences



Dense accurate geometric correspondences

- Convolutional spatial transformer
- Fully convolutional
- Learns feature directly
- Active hard negative mining

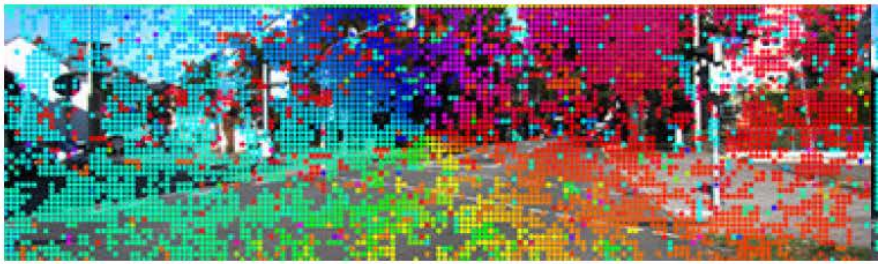
Network Architecture



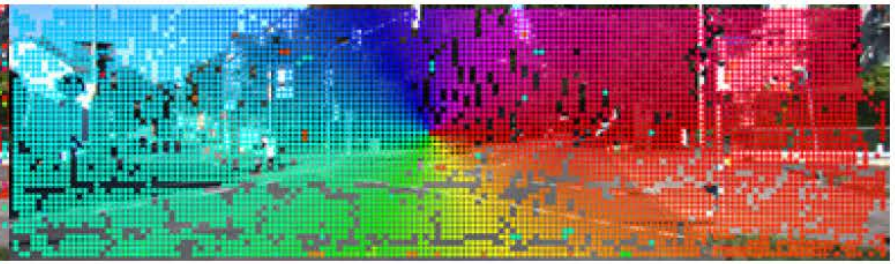
Experimental Results

(a) Original image pair and keypoints

(b) SIFT [23] NN matches



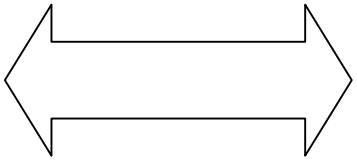
(c) DAISY [30] NN matches



(d) Ours-HN NN matches

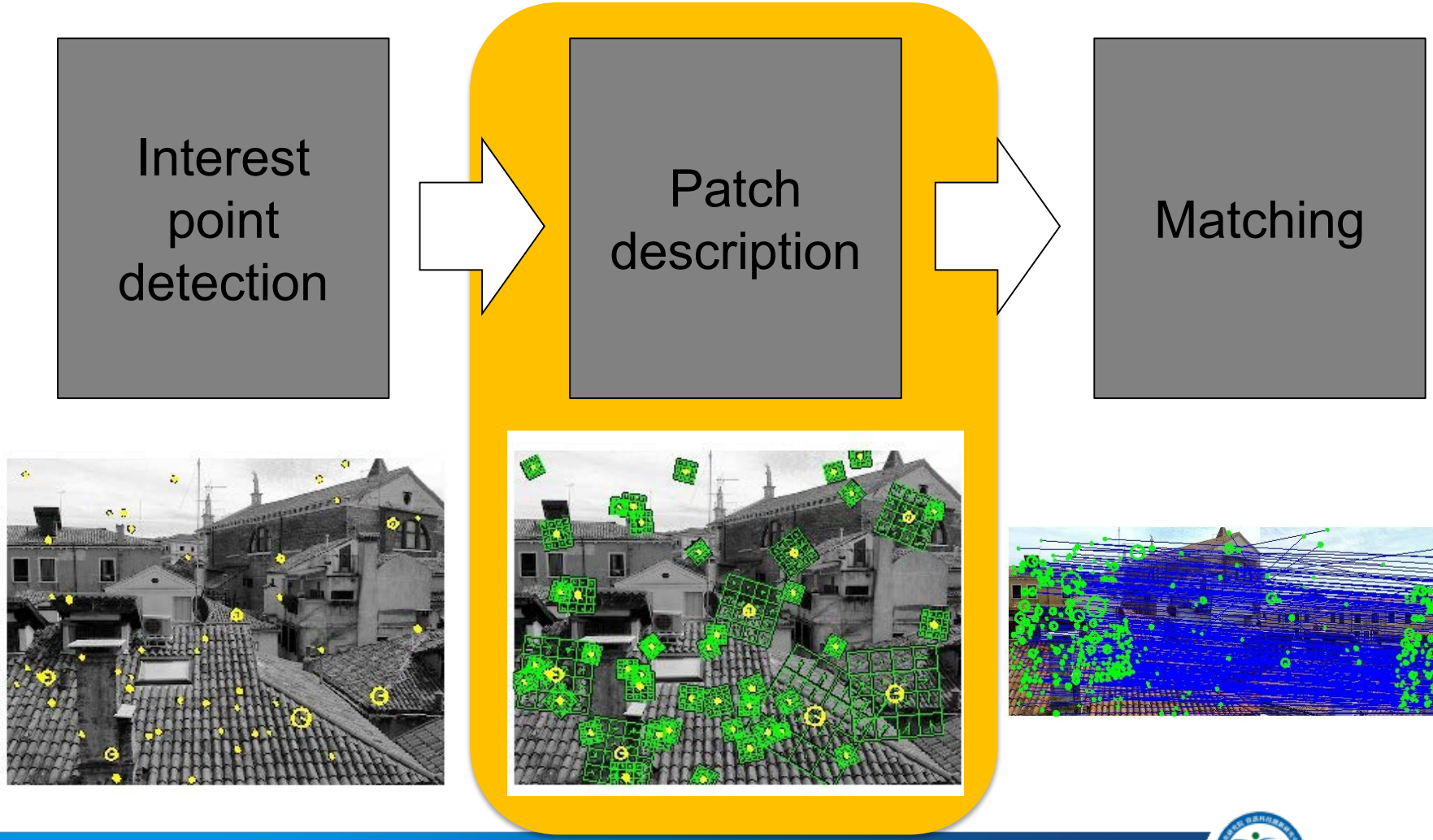
Introduction: Image Matching

- Consider two images
- Same object? Same place? Same building?



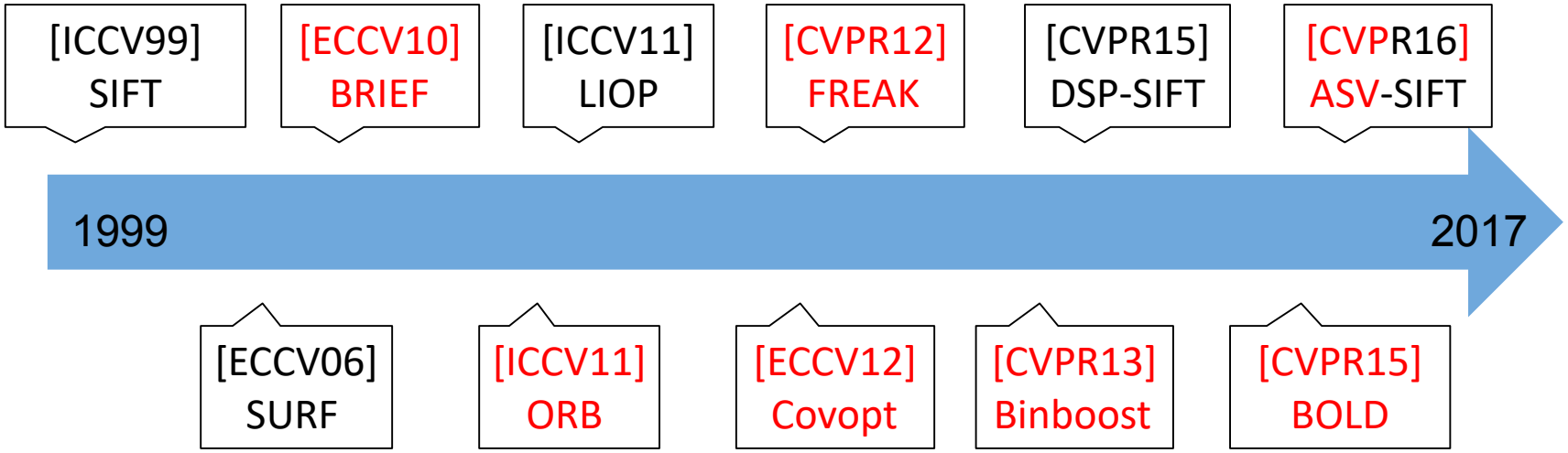
DeepCD (Deep complementary descriptor) for Image Matching

[Yang et al., ICCV'17]



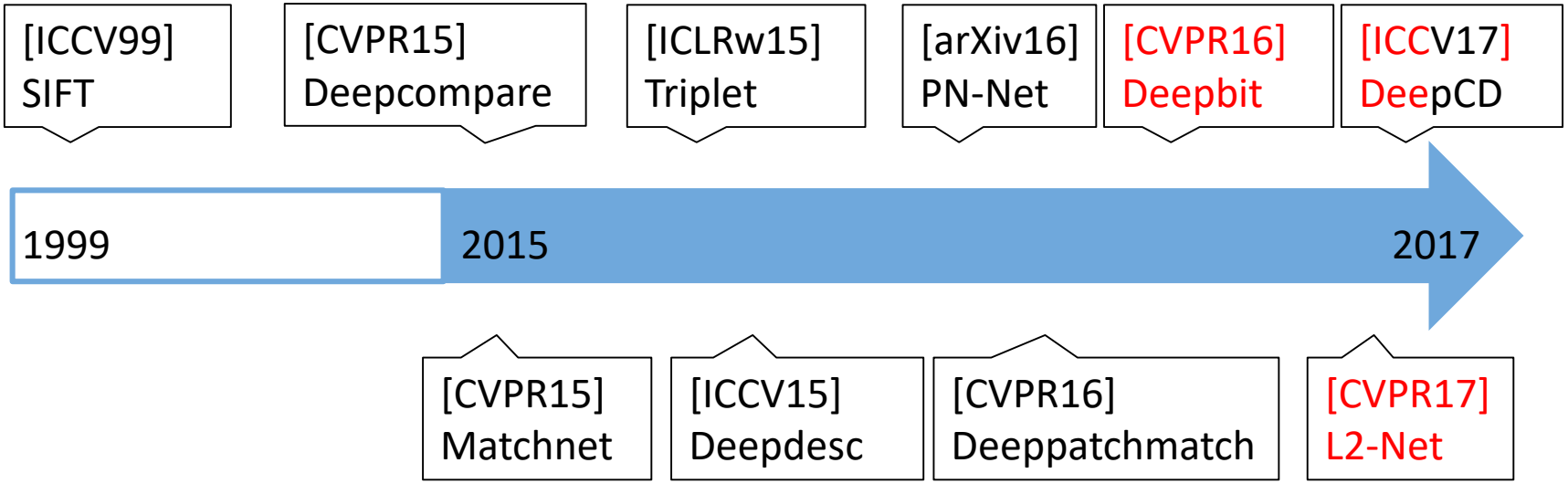
Hand-crafted Descriptors

- Manual design
- Human observation (translation, scaling, stability, ...)
- Sub-optimal results



Descriptors Derived by Deep Learning

- Data-driven
- Complex network slow metric learning
- Good performance



real-valued descriptor **binary descriptor**

Motivation

High performance:

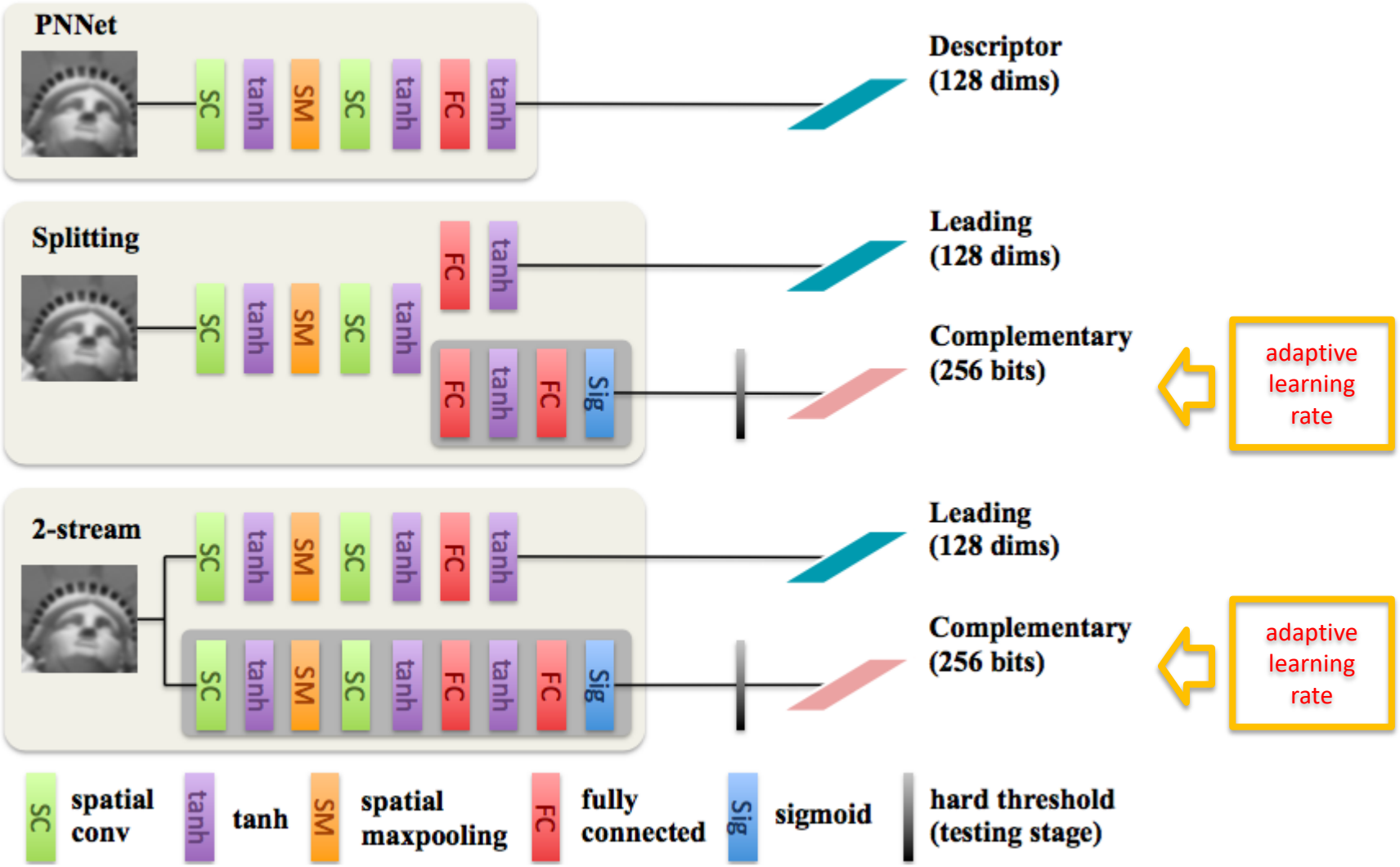
Requiring a high-dimensional descriptor or a complex distance metric

Compactness:

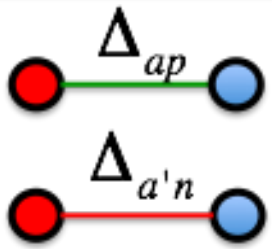
Requiring a low-dimensional descriptor or a binary descriptor

Can we substantially enhance the performance by adding one additional, compact descriptor?

Idea


PNNet
SC
tanh
SM
SC
tanh
FC
tanh
Descriptor
(128 dims)
Splitting
SC
tanh
SM
SC
tanh
FC
tanh
FC
tanh
FC
Sig
Leading
(128 dims)
Complementary
(256 bits)
adaptive
learning
rate
2-stream
SC
tanh
SM
SC
tanh
FC
tanh
SC
tanh
SM
SC
tanh
FC
tanh
FC
Sig
Leading
(128 dims)
Complementary
(256 bits)
adaptive
learning
rate
SC
spatial
conv
tanh
tanh
SM
spatial
maxpooling
FC
fully
connected
Sig
sigmoid
hard threshold
(testing stage)

Siamese Target and Triplet Target

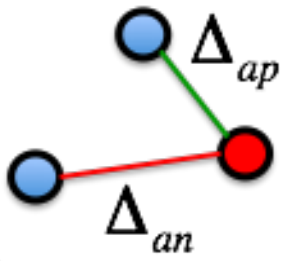


Siamese Target:

$$\Delta_{ap} \rightarrow 0$$

$$\Delta_{a'n} \rightarrow d_{\max}$$

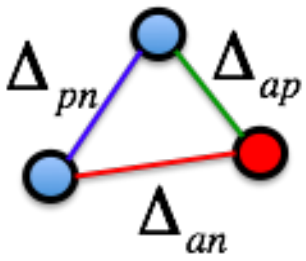
positive pair (a,p) and negative pair (a’,n) are separated



Triplet Target:

$$\left(\Delta_{ap}, \Delta_{an}\right) \rightarrow \left(0, d_{\max}\right)$$

positive pair (a,p) and negative pair (a,n) are linked



All Connected Triplet Target:

$$\left(\Delta_{ap}, \Delta_{an}, \Delta_{pn}\right) \rightarrow \left(0, d_{\max}, d_{\max}\right)$$

Notation

$$\Delta_{ij} \triangleq D(d(q_i), d(q_j))$$

distance Δ_{ij}
(q_i,q_j for different patches)

- We take L2-norm as our distance metric D since it is efficient.

$$d = \arg\min_{d} \sum_{ij} L(\Delta_{ij})$$

learned descriptor: d

- L is the loss function.

Triplet Training

Step.1

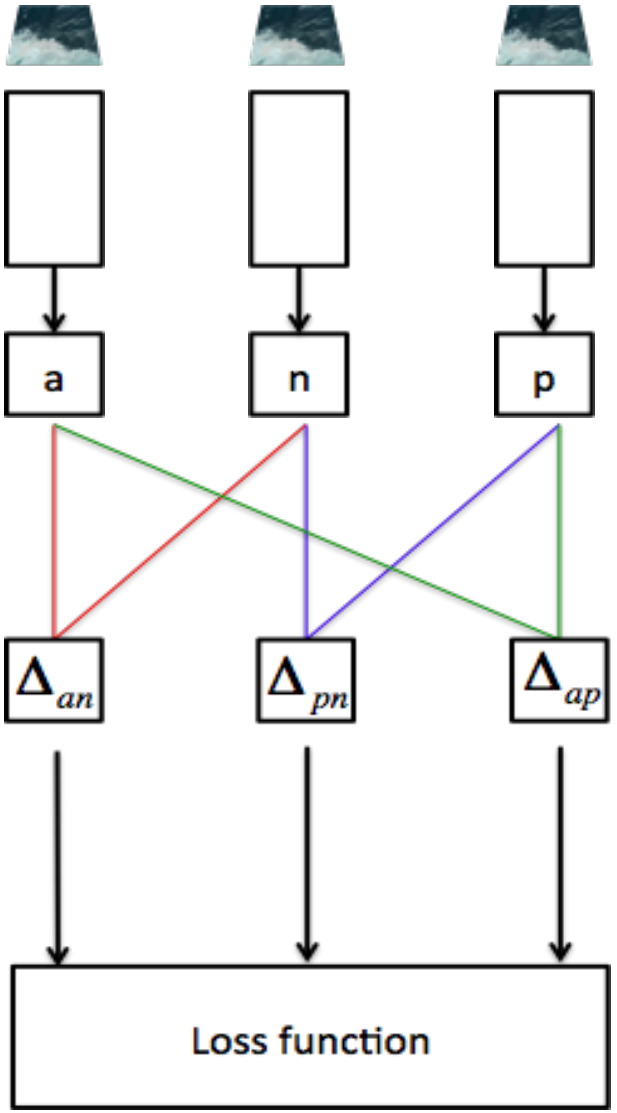
Shared networks

Step.2

Compute distances

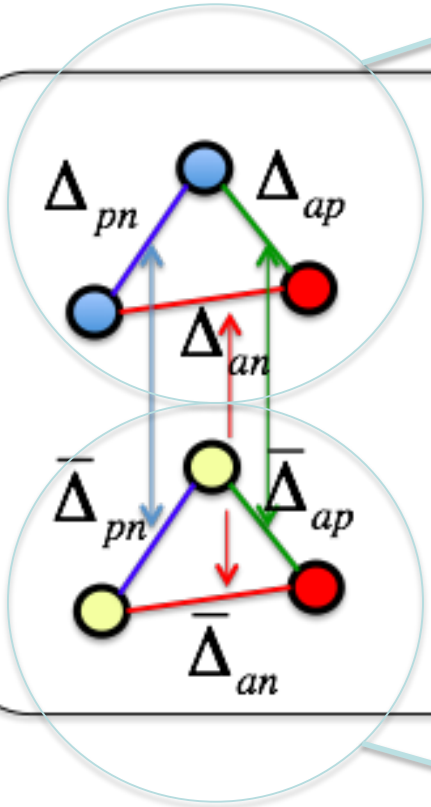
Step.3

Loss function



DeepCD Triplet

Triplet under leading descriptor



DeepCD Target:

$$\left(\sqrt{\Delta_{ap}\bar{\Delta}_{ap}}, \sqrt{\Delta_{an}\bar{\Delta}_{an}}, \sqrt{\Delta_{pn}\bar{\Delta}_{pn}}\right) \rightarrow \left(0, d_{\max}, d_{\max}\right)$$

Triplet under complementary descriptor

Joint Loss Notation

$$\Delta_{ij} \triangleq D(d(q_i), d(q_j))$$ **leading distance**

$$\overline{\Delta}_{ij} \triangleq \overline{D}(\overline{d}(q_i), \overline{d}(q_j))$$ **complementary distance**

$$J(q_i, q_j) = L(\Delta_{ij}) + \lambda L'(\Phi(\Delta_{ij}, \overline{\Delta}_{ij}))$$ **Joint optimization**

product
late fusion

$$(d, \overline{d}) = \arg\min_{d,\overline{d}} \sum_{ij} J(q_i, q_j)$$

Joint Loss Function

$$J(q_a, q_p, q_n) = L(\Delta_{ap}, \Delta_{an}, \Delta_{pn}) + \lambda L'\left(\sqrt{\Delta_{ap}\overline{\Delta}_{ap}}, \sqrt{\Delta_{an}\overline{\Delta}_{an}}, \sqrt{\Delta_{pn}\overline{\Delta}_{pn}}\right)$$

SoftPN (no hard margin, using ratio to separate positive pair and negative pair)

$$L(\delta^+, \delta_1^-, \delta_2^-) = \left[e^{\delta^+} / \left(e^{\min(\delta_1^-, \delta_2^-)} + e^{\delta^+}\right)\right]^2 + \left(\left[e^{\min(\delta_1^-, \delta_2^-)} / \left(e^{\min(\delta_1^-, \delta_2^-)} + e^{\delta^+}\right)\right] - 1\right)^2$$

DeepCD Training

Step.1

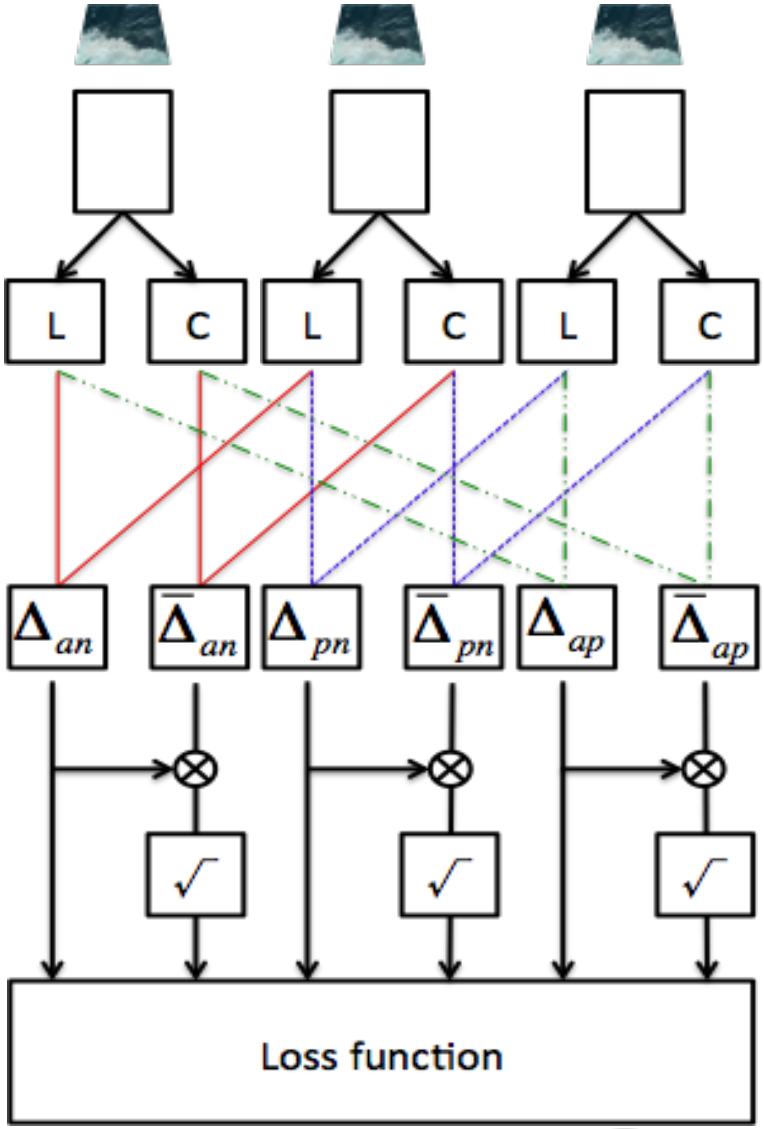
Shared network

Step.2

Compute distances

Step.3

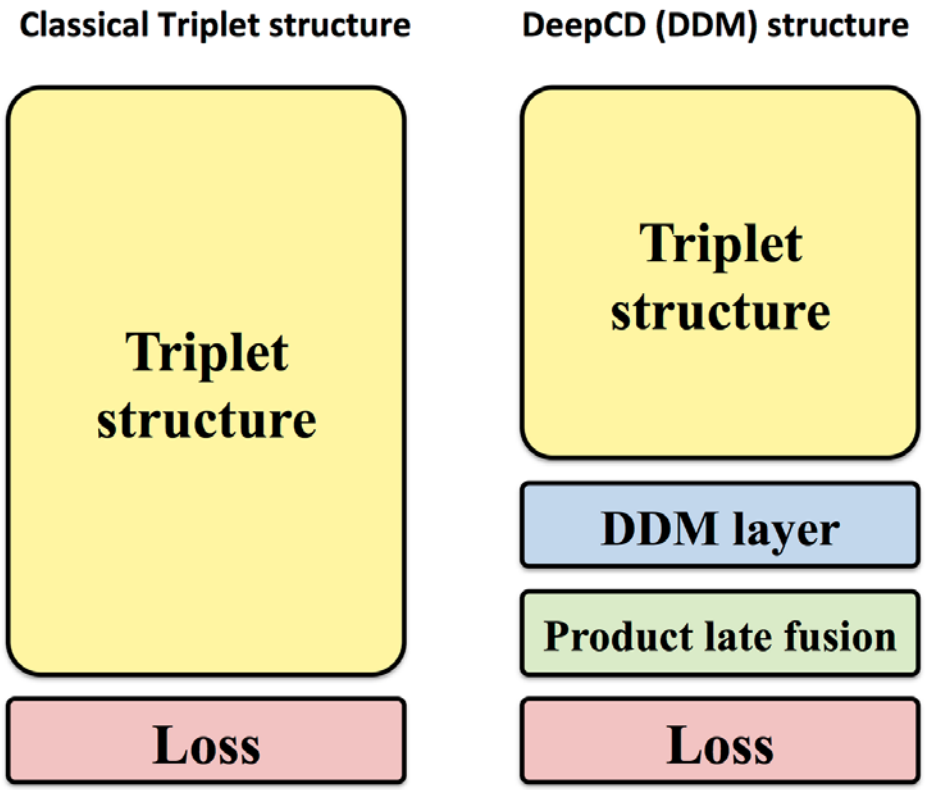
Loss function



Data Dependent Module (DDM)

How to enhance the complementary property?

Adaptive learning rate is applied to the complementary stream



Experiment

- Evaluation on three datasets
- Brown dataset (cross dataset transfer)
  - Cropped patches, FPR95
  - Three subsets: liberty, notredame, yosemite, ~500k patch pairs
- Strecha dataset (perspective change)
  - Dense patches, mAP
  - 2 image sequences, 7~10 magnitudes
- Oxford dataset (key point detection involved)
  - DoG detection, mAP
  - 48 images in 8 sequences with different variations and magnitudes

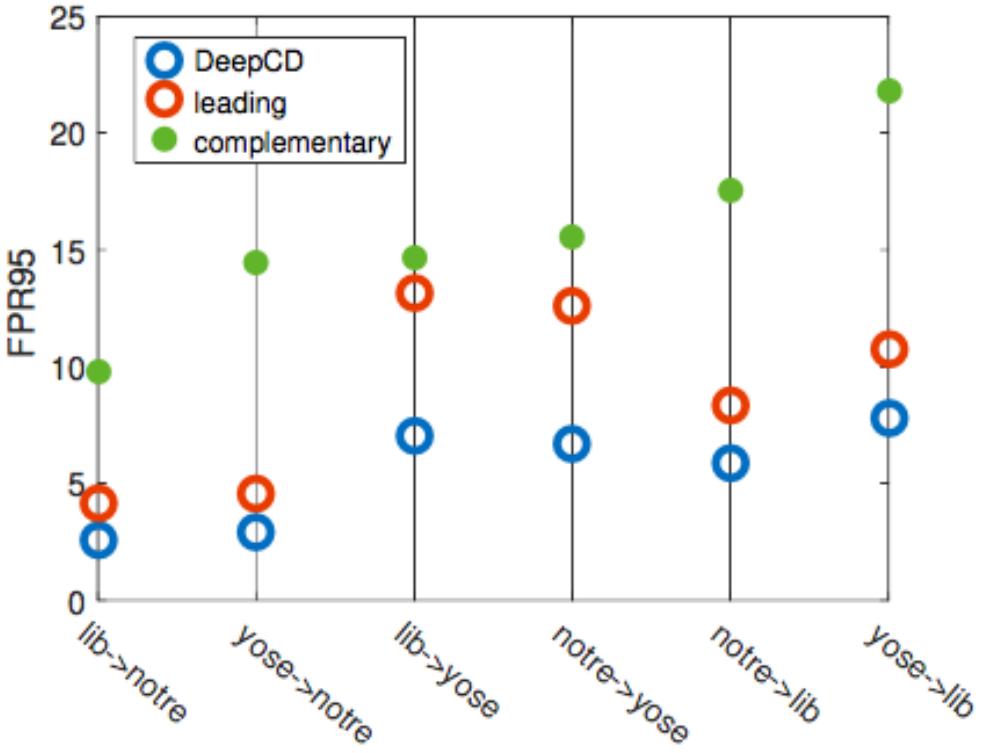
Results on Brown Dataset

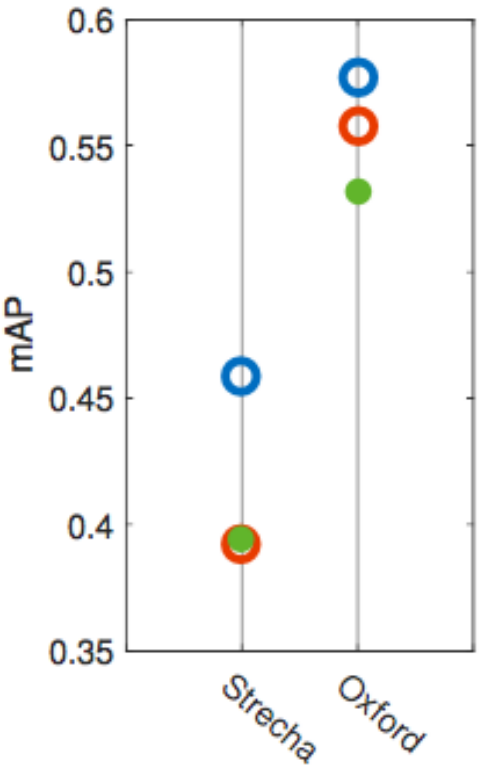
Test	Train	des.	2-stream PNNet +CBP	Deep comp	Deep desc	TNet	PNNet		TFeat Margin*	DeepCD splitting	DeepCD 2-stream	DeepCD 2-stream (DDM)
		bytes	256	512	128	256	128	256	128		128+32	
Notre.	Lib.		16.99	4.54	5.16	3.91	3.81	3.71	3.12	2.98	2.87	**2.59**
	Yose.		18.22	5.58		5.43	4.45	4.23	3.85	3.51	3.02	**2.95**
Yose.	Lib.		34.46	13.24	18.21	10.65	9.55	8.99	7.82	8.27	7.78	**7.03**
	Notre.		30.70	13.02		9.47	7.74	7.21	7.08	6.85	**6.65**	6.69
Lib.	Notre.		21.33	8.79	9.56	9.91	8.27	8.13	7.22	6.32	6.08	**5.85**
	Yose.		32.47	12.84		13.45	9.76	9.65	9.79	9.10	**7.76**	7.82
mean			25.70	9.67	10.98	8.8	7.26	6.98	6.47	6.17	5.69	**5.48**

Table 1. FPR95 for real-valued descriptors on the Brown dataset.

Complementary property

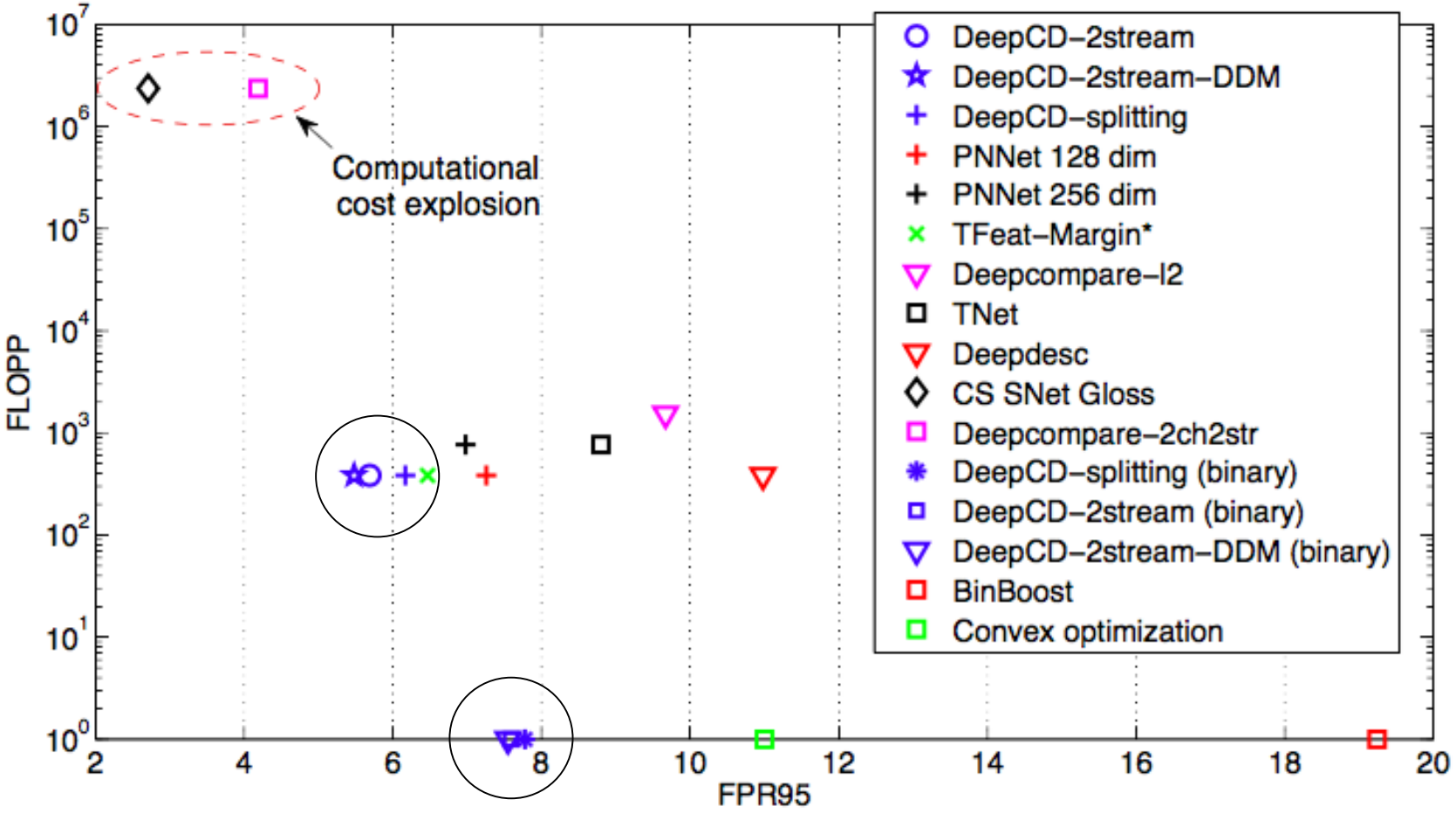
The complementary descriptor is weak, but it can improve the leading descriptor by late fusion.





Low Error vs. Low Computational Cost

Low error (FPR95) vs. Low computational cost (FLOPP)



Remarks

1. We use an **extra binary descriptor** for learning the complementary information.

2. Both **compactness** and **performance** are achieved.

3. A **dynamic learning rate module (DDM)** is proposed to enhance complementary properties.

4. Source code is available at https://github.com/shamangary/DeepCD

Thank You for Your Attention!

Yen-Yu Lin (林彥宇)

URL: http://cvlab.citi.sinica.edu.tw/

Email: yylin@citi.sinica.edu.tw