Bilinear Models for Fine-grained Visual Recognition

Subhransu Maji

College of Information and Computer Sciences

University of Massachusetts, Amherst

October 2015

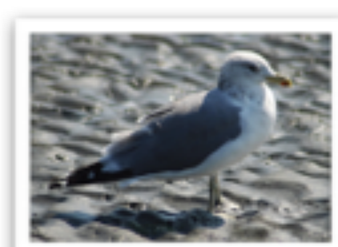
Fine-grained visual recognition

- Example: distinguish between closely related categories

California gull

Ringed beak gull

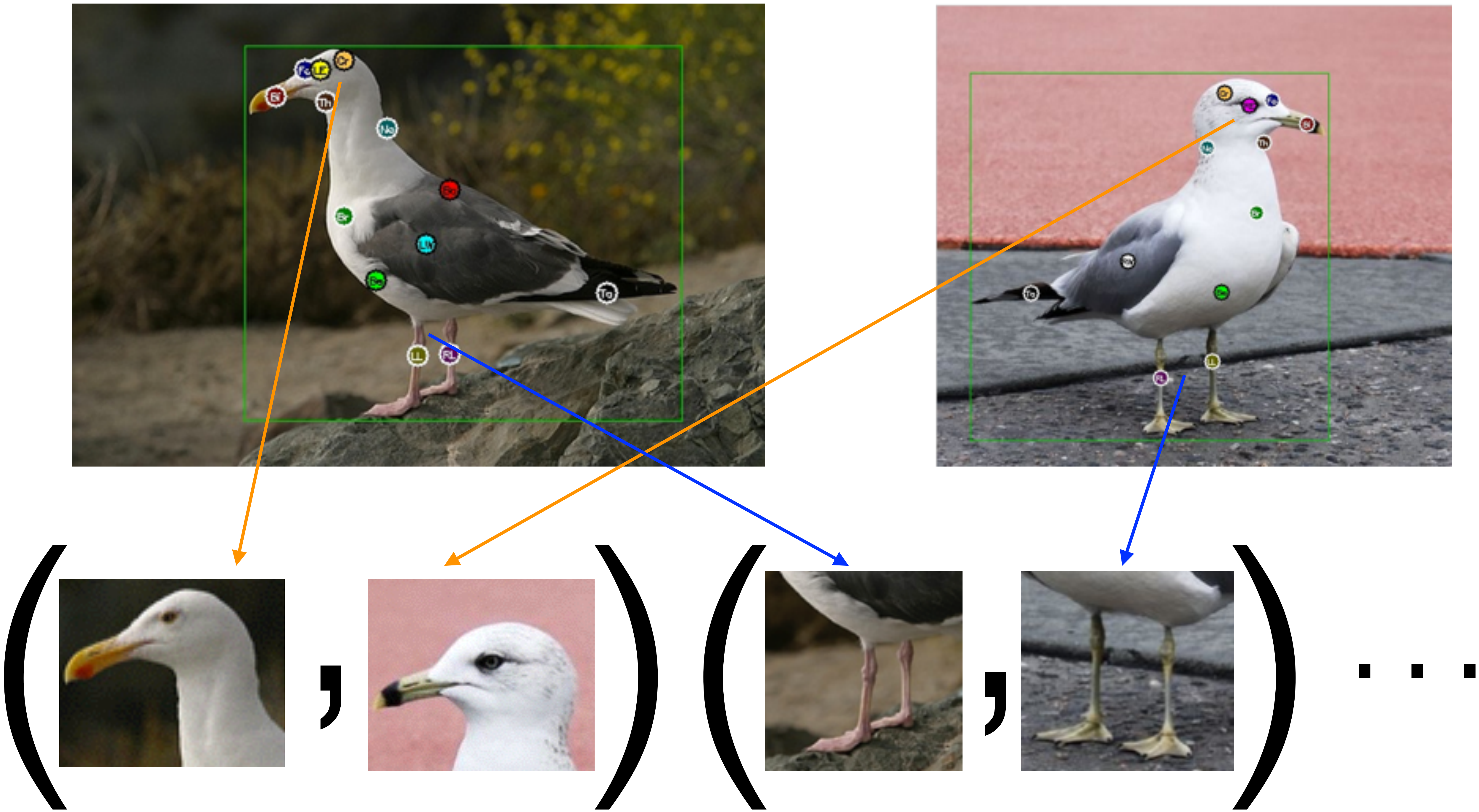





- Intra-category variation v.s. inter-category variation
  - location, pose, viewpoint, background, lighting, gender, season, etc

Part-based models

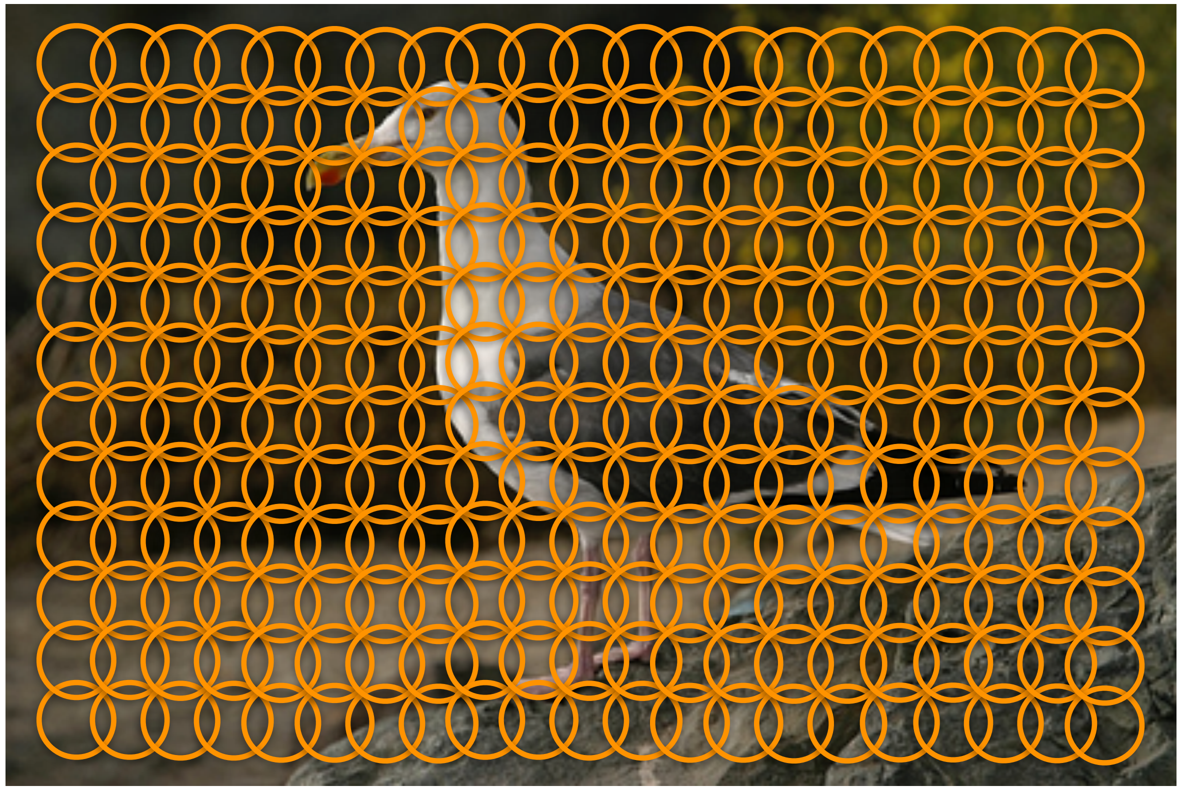
- Localize “parts” and compare corresponding locations



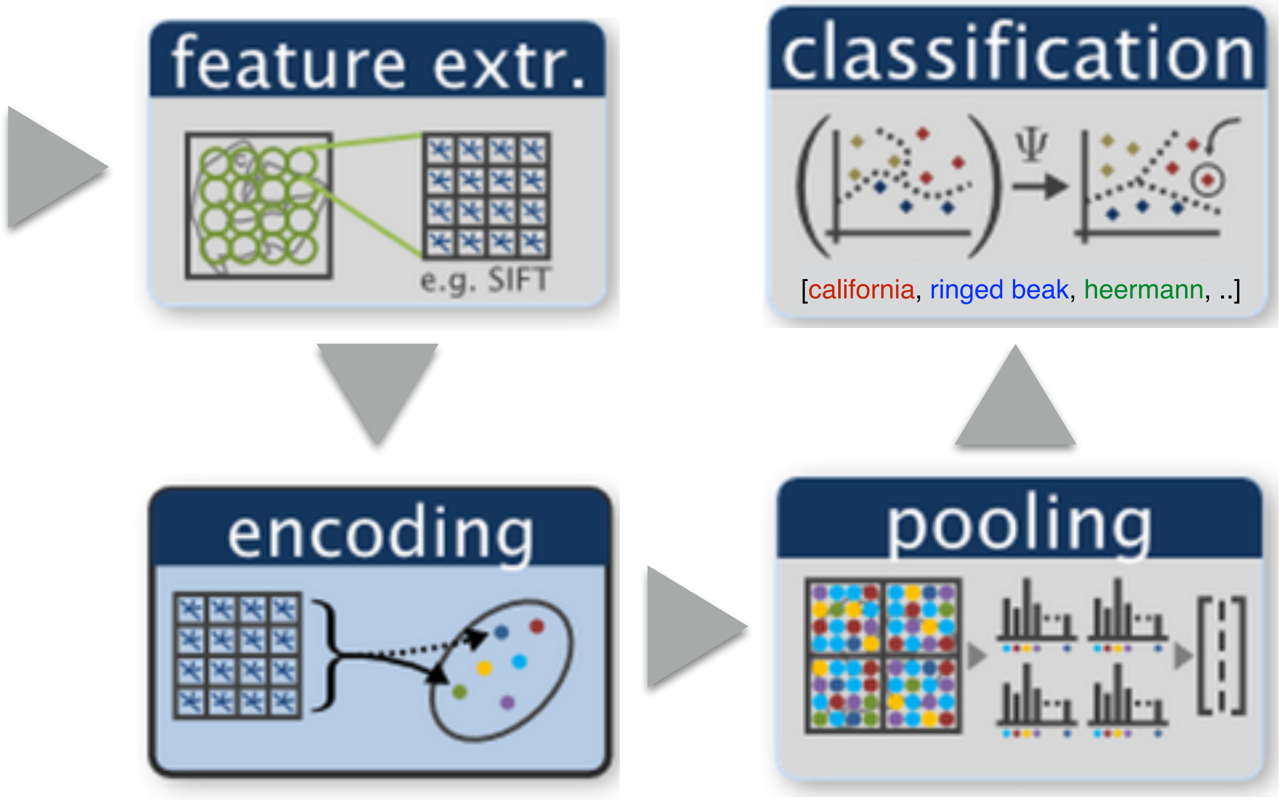
- Factor out the variation due to pose, viewpoint and location

Texture models

- Image as a collection of patches [bag-of-visual-words, Csurka et al 04]



dense sampling



- Orderless pooling and no explicit modeling of pose or viewpoint
- Invariances due to
  - choice of features (e.g. SIFT is robust to lighting changes)
  - encoding + pooling + classification
- E.g., Fisher-vectors work remarkably well for fine-grained tasks

Tradeoffs

- Part-based models [Zhang'14, Branson'14]
  - ✓ Offer the best recognition accuracy on many fine-grained recognition datasets (e.g., birds, cars, etc)
  - x Relatively slow since it involves part detection
  - x Needs part annotations for training. This can be time consuming and may require expert knowledge (especially for fine-grained domains). Parts may be hard to define them for some categories.
- Texture models [Perronnin'10]
  - ✓ Easy to deploy since they only need image labels for training
  - ✓ Fast CPU implementations
  - x Lower recognition accuracy
    - ➡ Pipelined procedure (features → encoding → classification) can be suboptimal. For example, the feature extractors are not learned.

- Can we get the best of both?

Bilinear models for classification

- A bilinear model for classification is a four-tuple

$$\mathcal{B} = (f_A, f_B, \mathcal{P}, \mathcal{C})$$

feature extractor | pooling | classification

$$f : \mathcal{L} \times \mathcal{I} \rightarrow R^{c \times D}$$

image → local features

$f_A(l, \mathcal{I})$ "parts"

$f_B(l, \mathcal{I})$ "colors"

$$f_A(l, \mathcal{I})^T f_B(l, \mathcal{I})$$

$\mathcal{I}$

$$\text{bilinear}(l, \mathcal{I})$$

f_A

f_B

	beak	tail	belly	legs	belly
red					
blue					
gray					
blue					
black					

example "gray belly"

Bilinear models for classification

- A bilinear model for classification is a four-tuple

$$\mathcal{B} = (\underbrace{f_A, f_B}_{\text{feature extractor}}, \underbrace{\mathcal{P}}_{\text{pooling}}, \underbrace{\mathcal{C}}_{\text{classification}})$$

$$f : \mathcal{L} \times \mathcal{I} \rightarrow R^{c \times D}$$

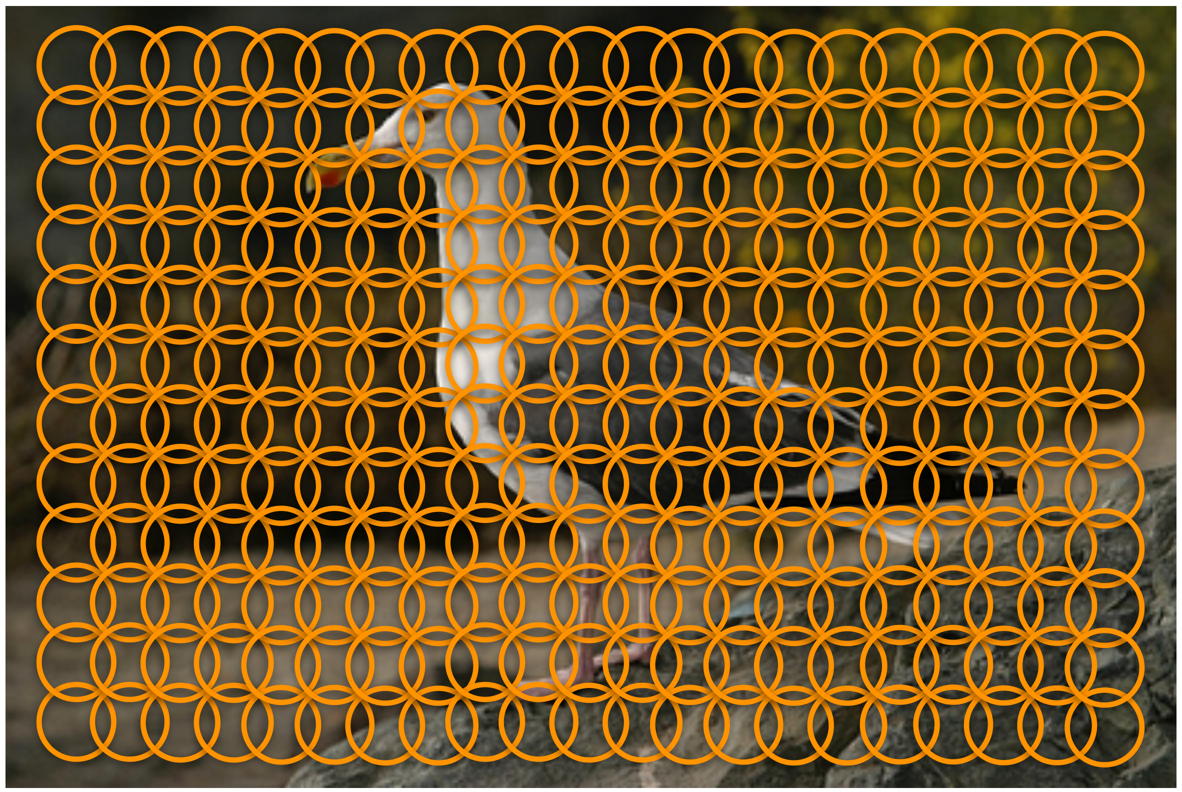
image $\longrightarrow$ local features $\xrightarrow{\text{pooling}}$ descriptor $\xrightarrow{\mathcal{C}}$ class

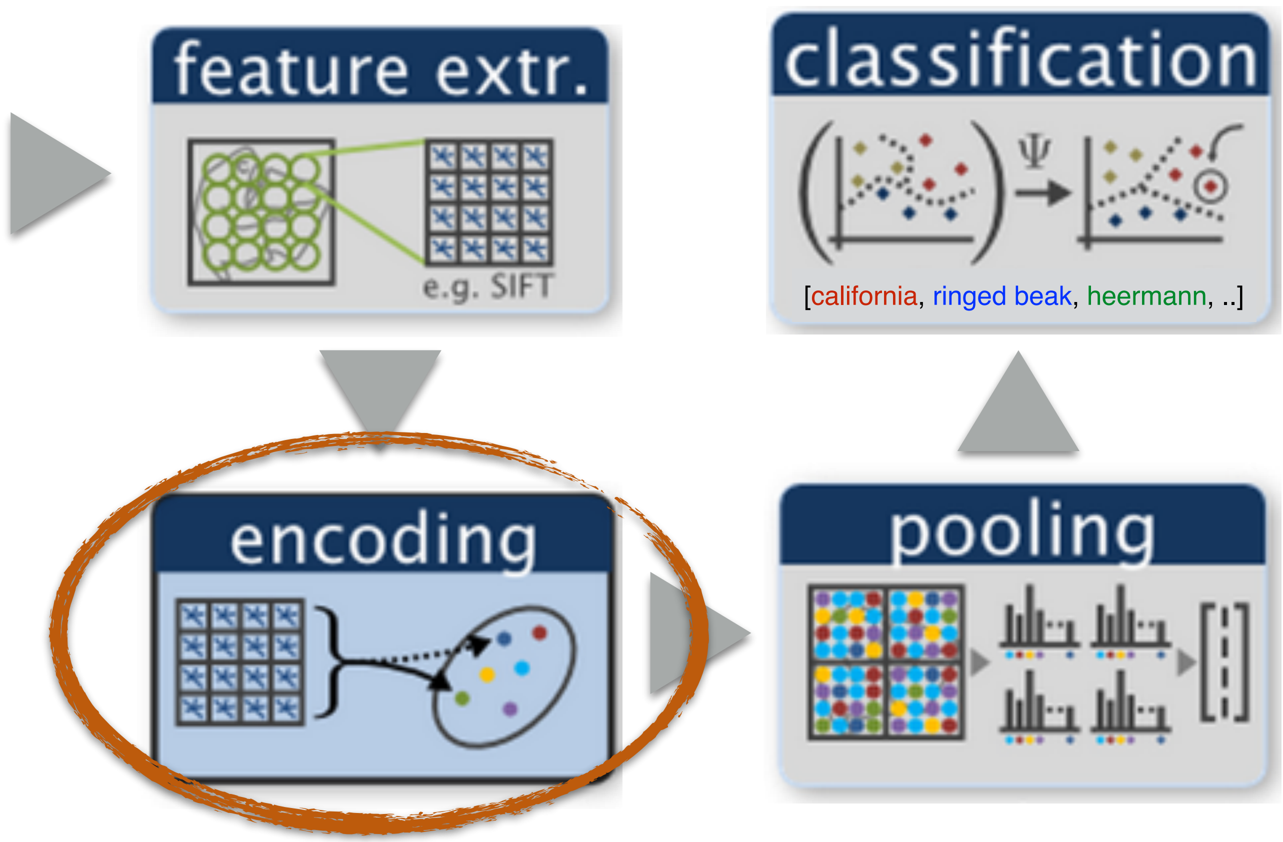
$$f_A(l, \mathcal{I}), \quad f_B(l, \mathcal{I}) \longrightarrow f_A(l, \mathcal{I})^T f_B(l, \mathcal{I}) \longrightarrow \sum_l \text{bilinear}(l, \mathcal{I})$$

$\mathcal{I}$ $\text{bilinear}(l, \mathcal{I})$ $\Phi(\mathcal{I})$

BoVW is a bilinear model

- Image is a collection of patches



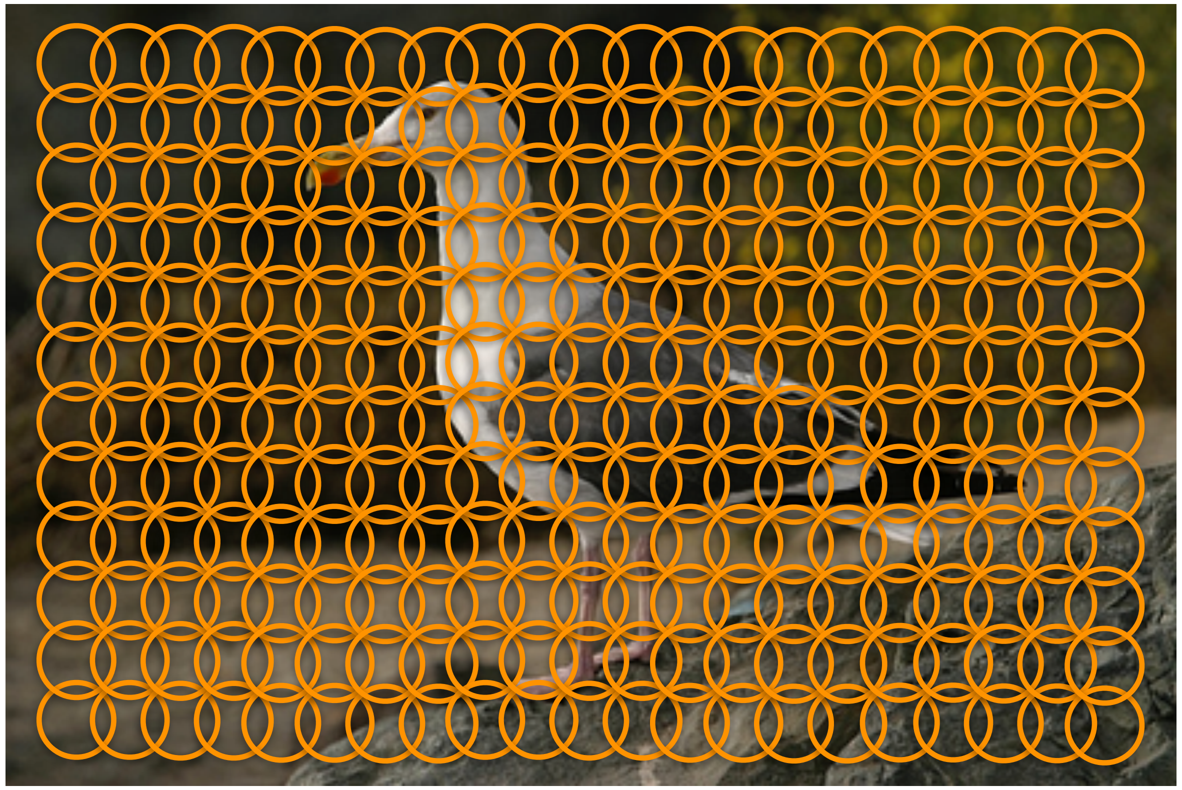


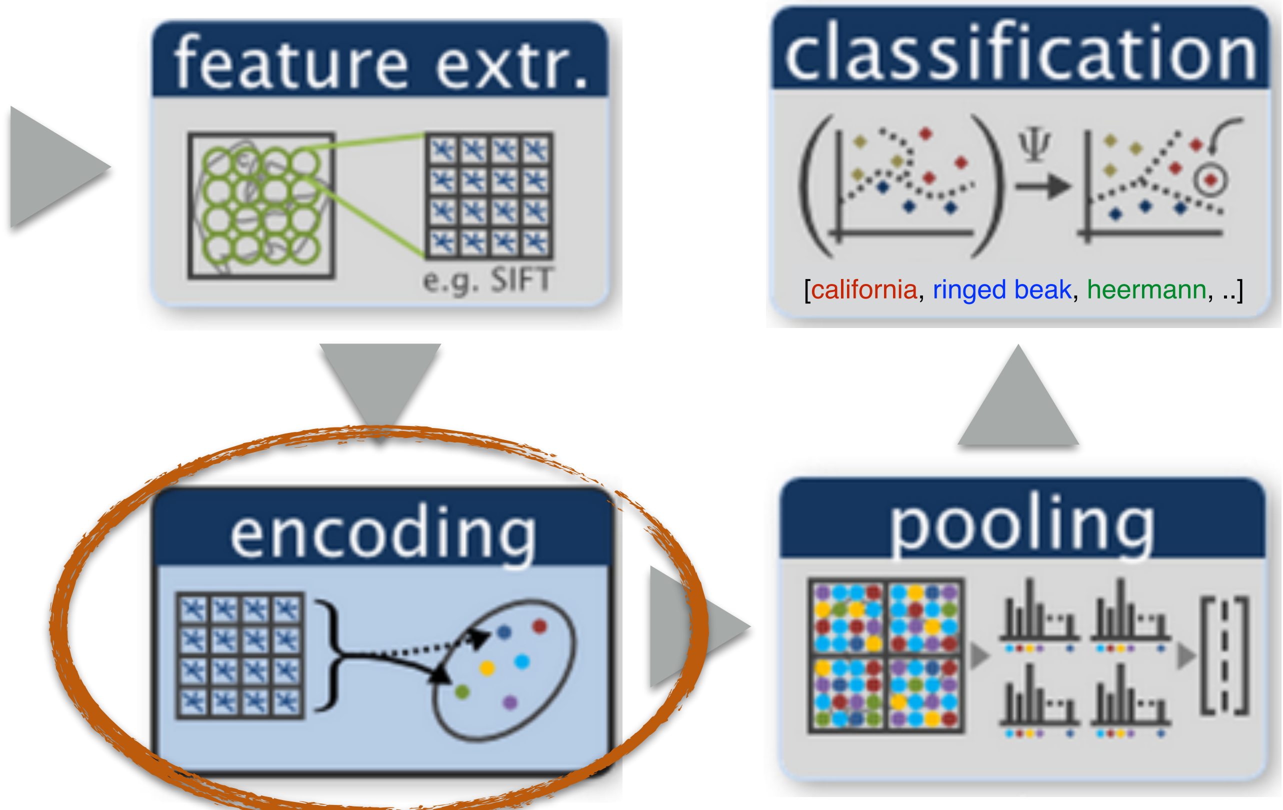
- Bag-of-visual words model [Csurka et al., 2004]
- Assign SIFT descriptor to the nearest center
  - Suppose η(**x**) = [0… 1 …], i.e., the binary assignment vector
- Then BoVW is a bilinear model

$$\mathcal{B} = (\eta(f_{\text{sift}}), 1, \mathcal{P}, \mathcal{C})$$

VLAD is a bilinear model

- Image is a collection of patches





- Vector of Locally Aggregated Descriptors (VLAD) [Je´gou et al., 10]
  - Locally encode each feature **x** as $(\mathbf{x} - \mu_k) \otimes \eta(\mathbf{x})$

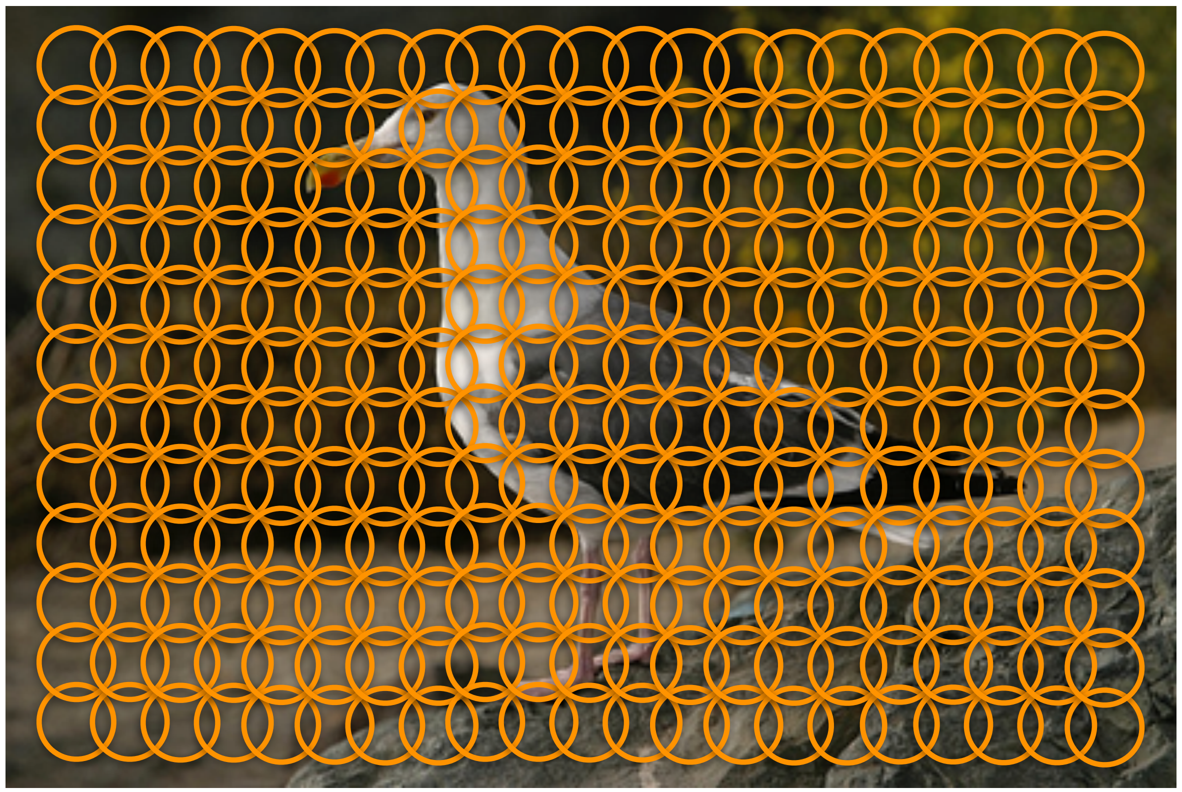
“kronecker product”

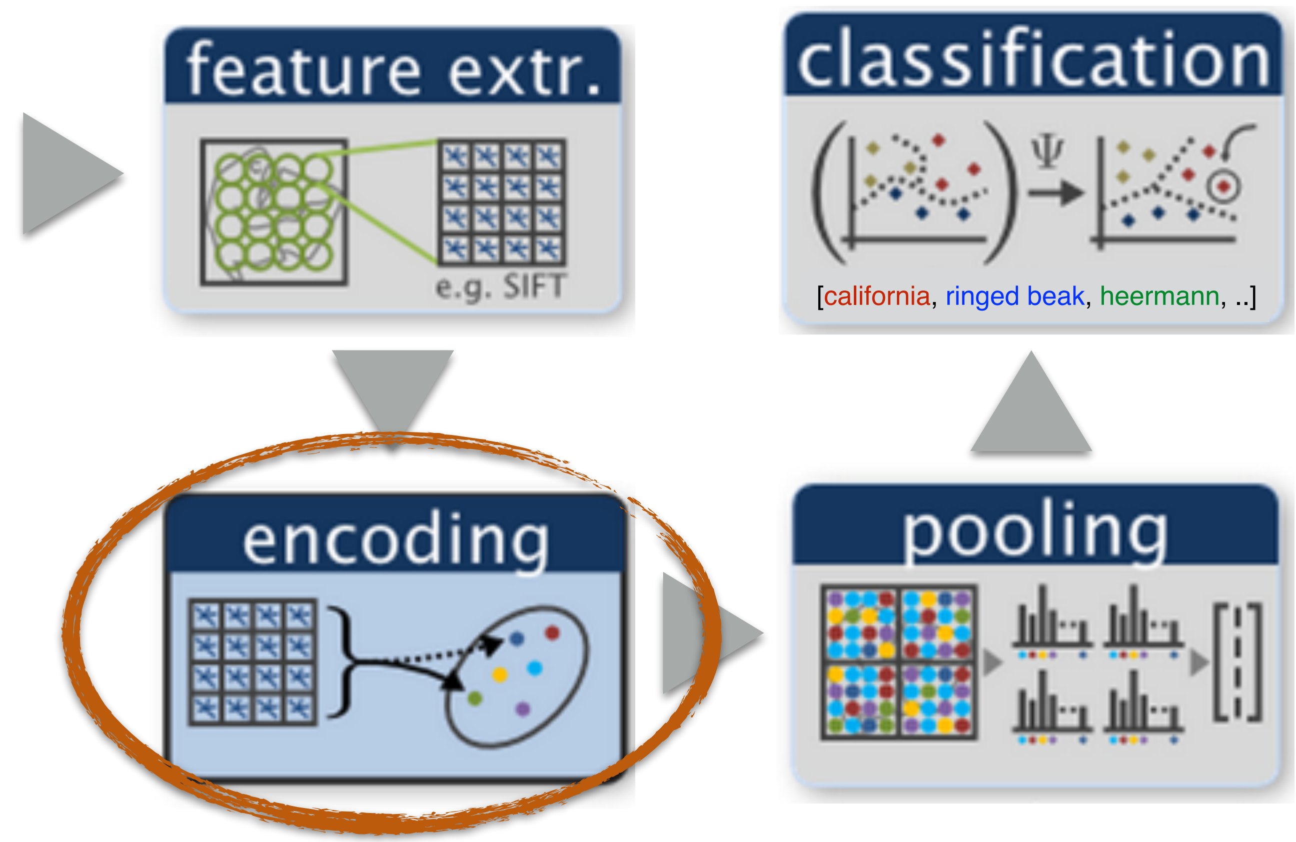
- VLAD is a bilinear model with

$$f_A = [\mathbf{x} - \mu_1; \mathbf{x} - \mu_2; \ldots; \mathbf{x} - \mu_k]$$
$$f_B = \mathrm{diag}(\eta(\mathbf{x}))$$

Fisher Vector is a bilinear model

- Image is a collection of patches





- Fisher vector (FV) models [Perronnin et al., 10]
  - Locally encode statistics of feature **x** weighted by η(**x**)

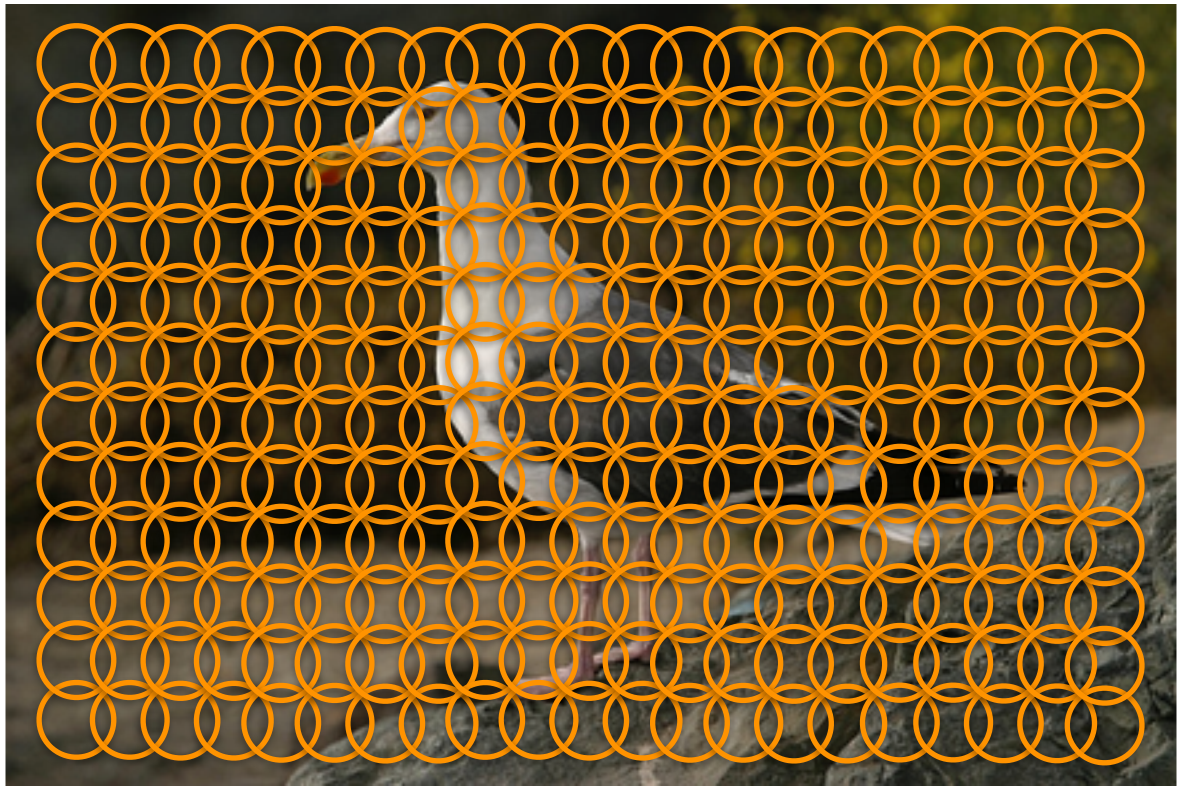
$$\alpha_i = \Sigma_i^{-\frac{1}{2}}(\mathbf{x} - \mu_i) \qquad \beta_i = \Sigma_i^{-1}(\mathbf{x}-\mu_i)\odot(\mathbf{x}-\mu_i) - 1$$

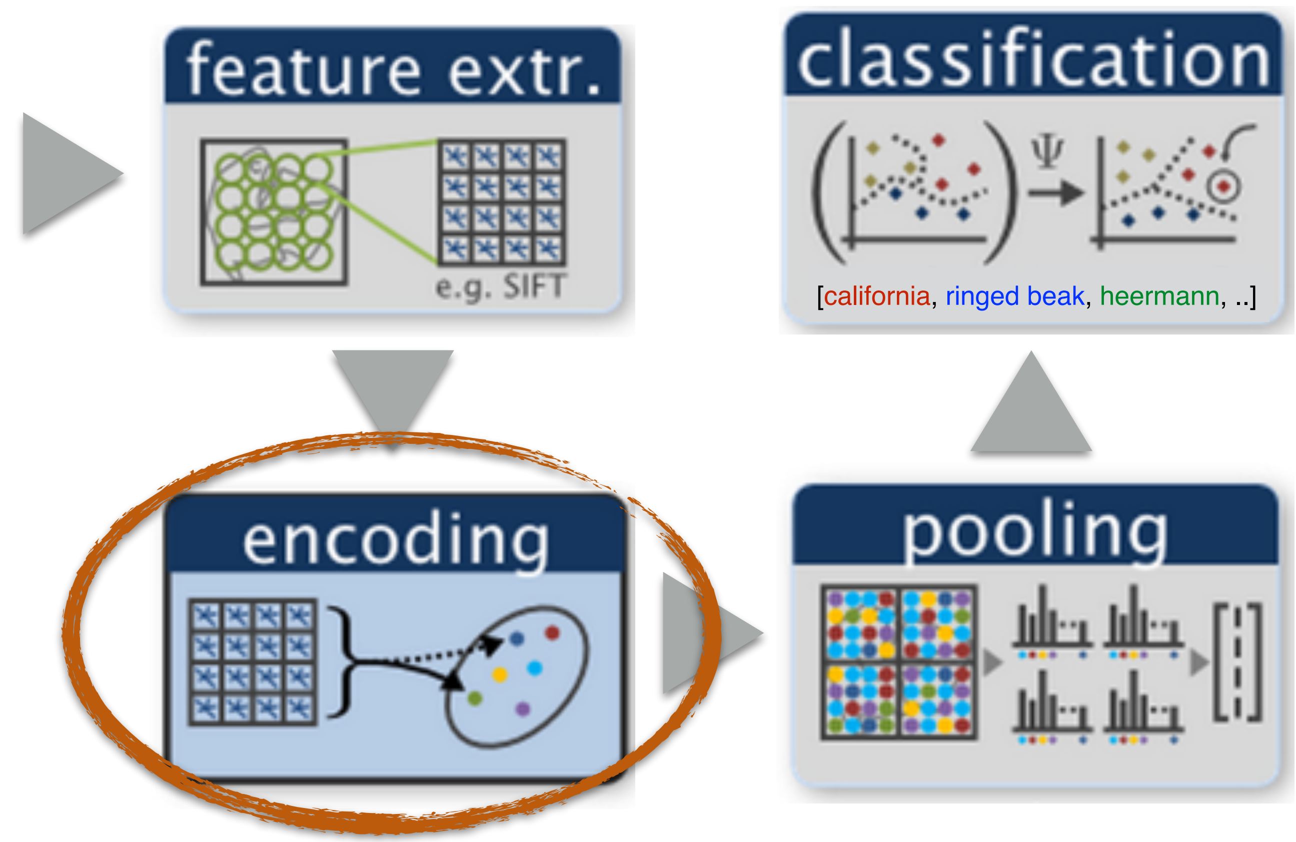
- FV is bilinear model with

$$f_A = [\alpha_1\ \beta_1; \alpha_2\ \beta_2; \ldots; \alpha_k\ \beta_k]$$
$$f_B = \mathrm{diag}(\eta(\mathbf{x})) \quad \text{“soft assignment”}$$

O2P is a bilinear model

- Image is a collection of patches

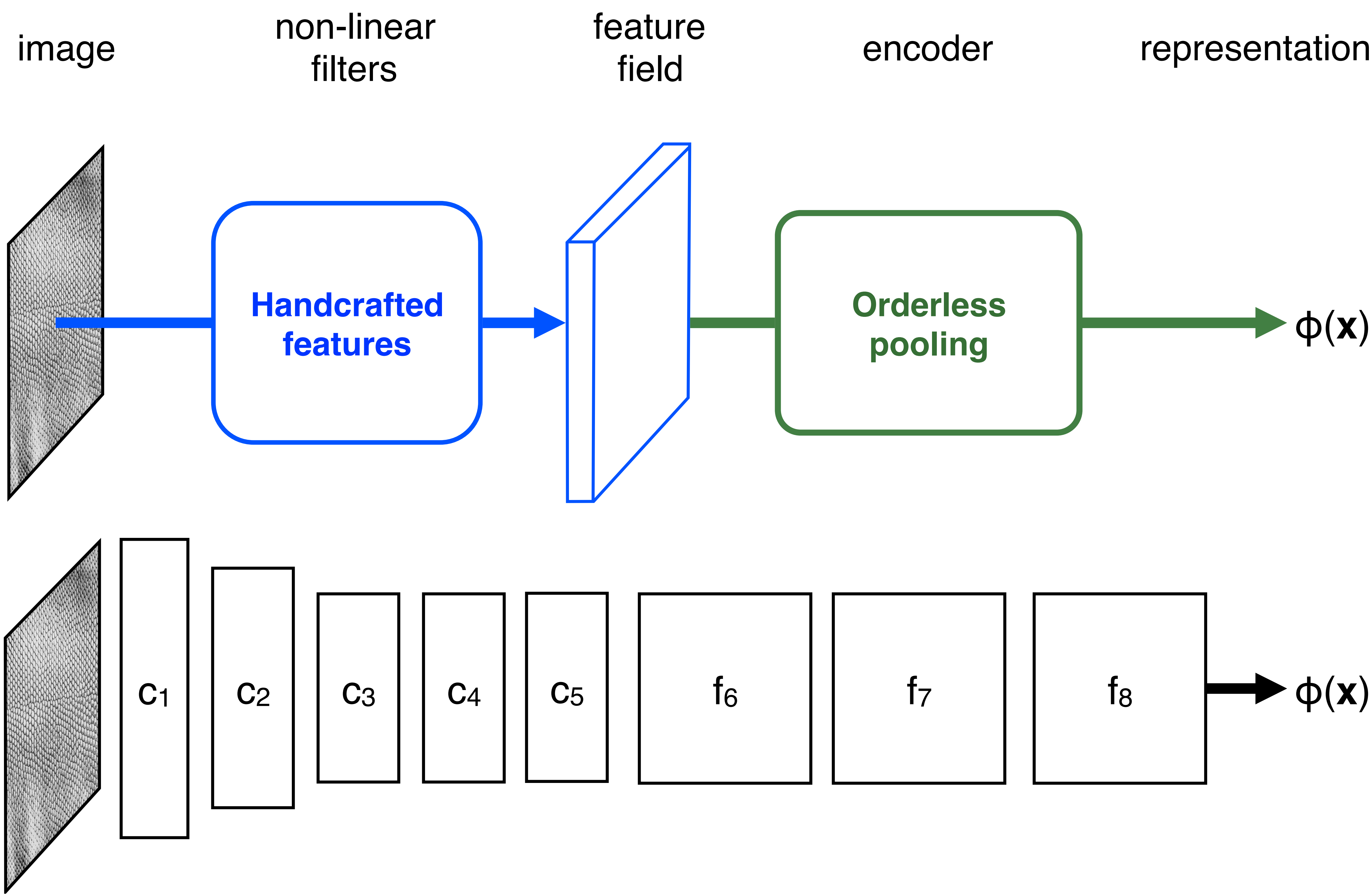




- Second order pooling [Carreira et al., 10]
  - Locally encode statistics of feature **x** weighted by **x** itself
    - ➡ Original formulation also proposes log non-linearity (maps the space of PSD matrices to an Euclidean space)
    - ➡ This is bilinear model with identical feature extractors

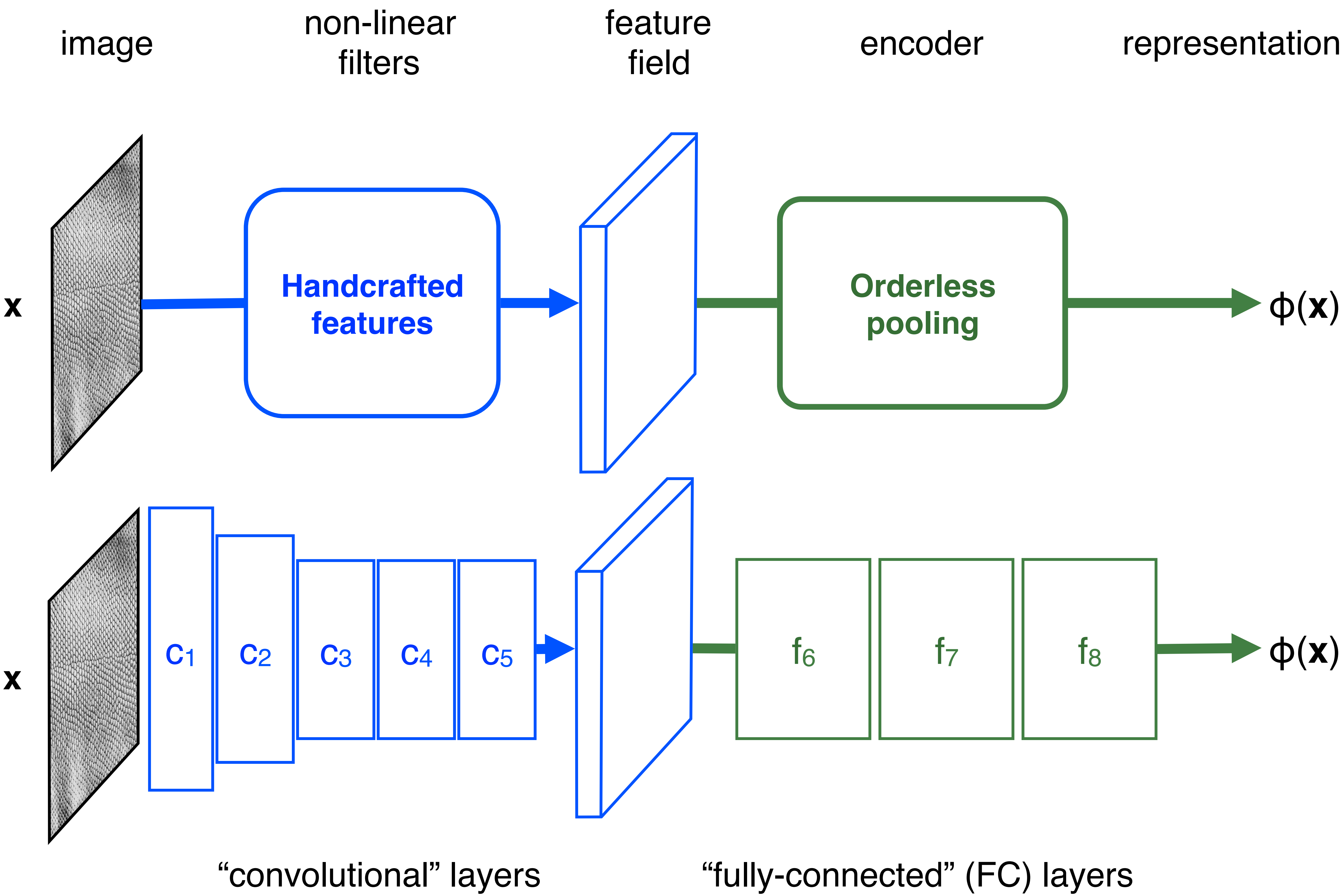
$$f_A = f_B = \mathbf{x}$$

Texture representations vs CNNs

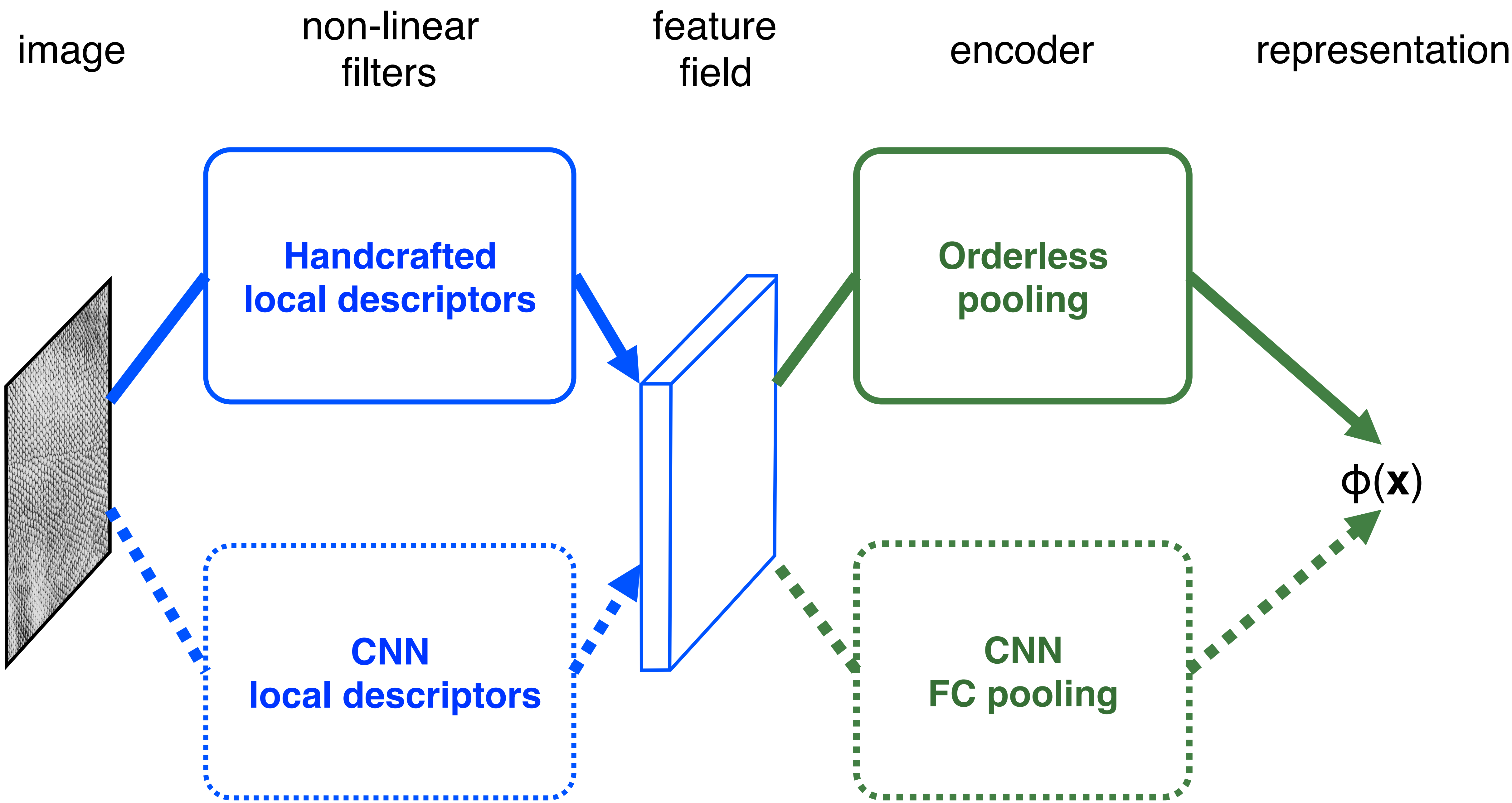


Cimpoi et al., CVPR 15

Texture representations vs CNNs

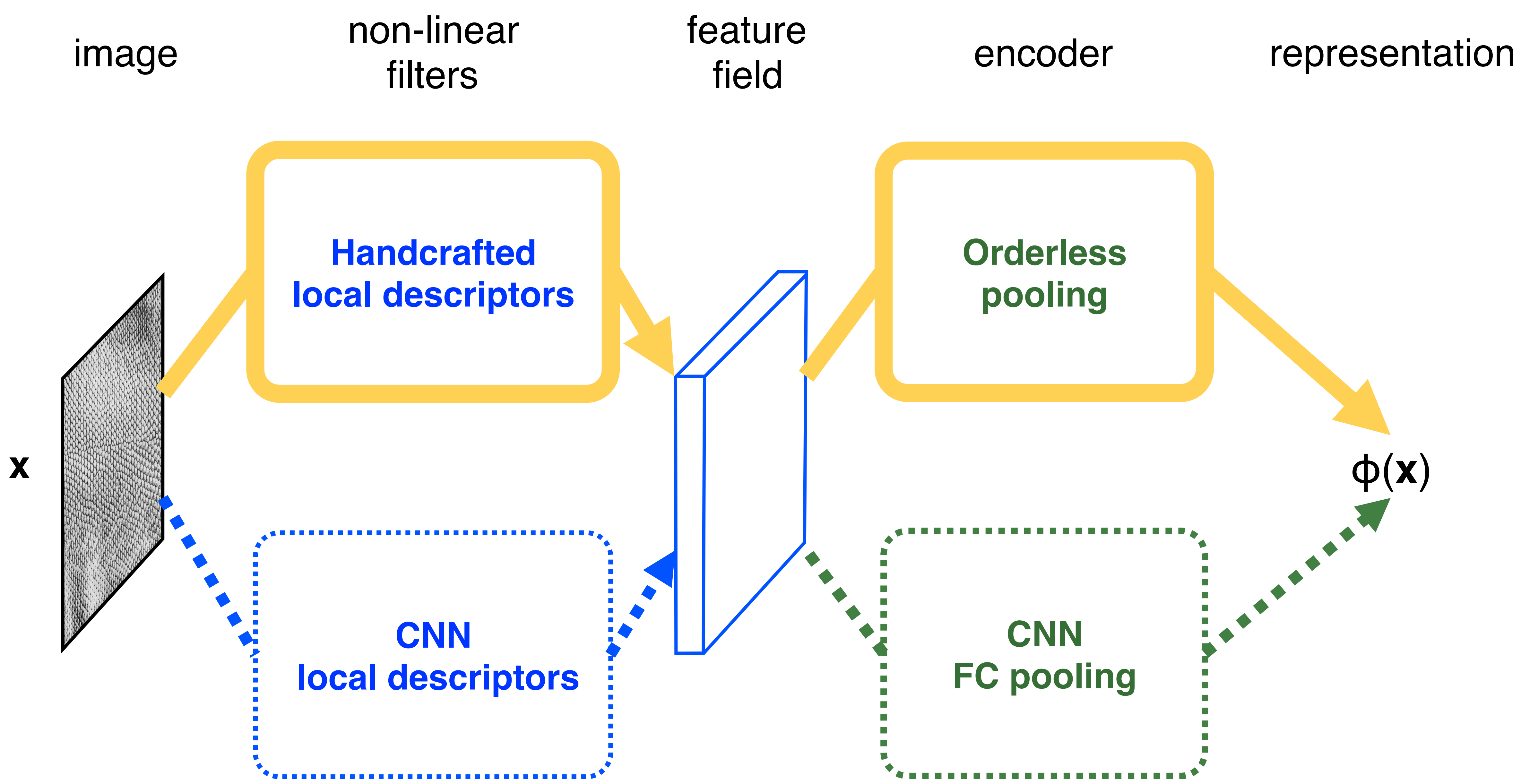


Mix and match



Mix and match

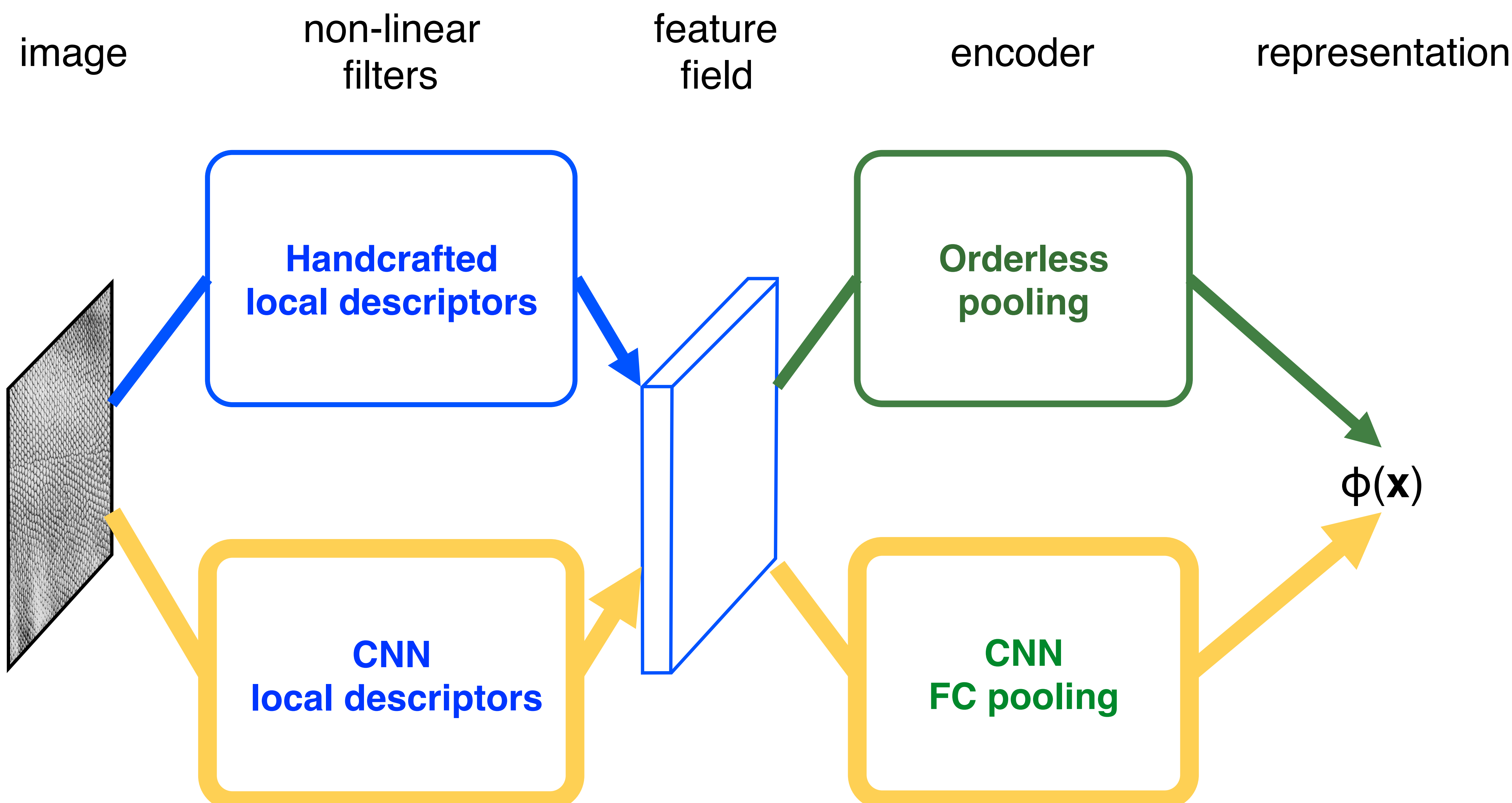
Standard texture representation



[Sivic and Zisserman 03, Csurka et al. 04, Perronnin and Dance 07, Perronnin et al. 10, Jegou et al. 10]

Mix and match

Standard application of CNN

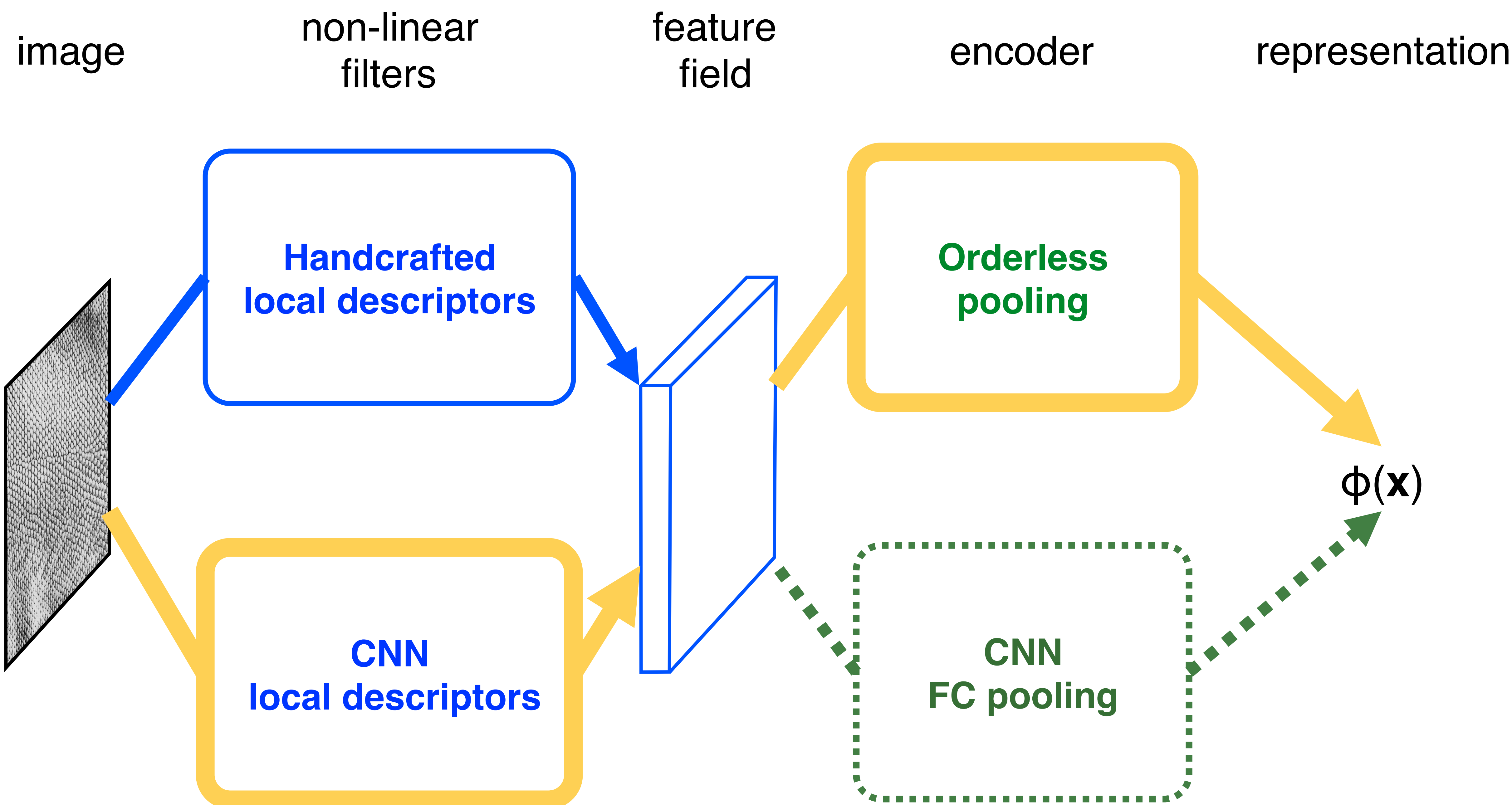


FC-CNN

[Chatfield et al. 14, Girshick et al. 2014, Gong et al. 14, Razavin et al. 14]

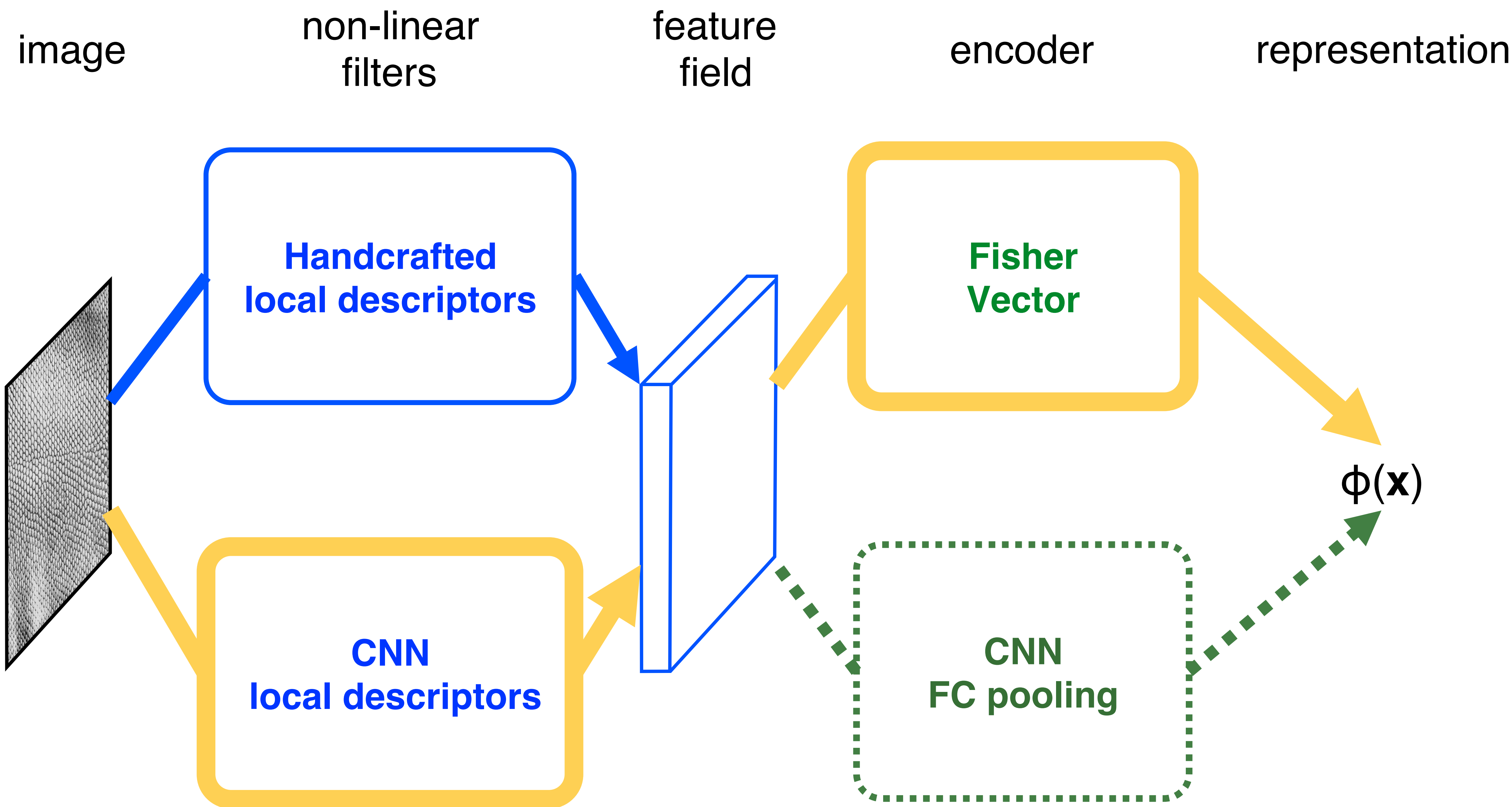
Mix and match

Order-less pooling of CNN local descriptors



Mix and match

CNN descriptors pooled by Fisher Vector



FV-CNN

CNNs for texture recognition

Texture recognition accuracy

Dataset	FV-SIFT	FC-CNN	FV-CNN	FV-CNN (VD)
KT-2b	70.8	71.0	**71.0**	**81.8**
FMD	59.8	70.3	**72.6**	**79.8**
DTD	58.6	58.8	**66.7**	**72.3**

Cimpoi et al., CVPR 15

Using the very deep model from Oxford VGG group that performed among the best on LSVRC 2014 (ImageNet classification challenge)

http://www.robots.ox.ac.uk/~vgg/research/very_deep/

Scenes as textures

- MIT Indoor dataset (67 classes)

Prev. best: **70.8**%
Zhou et al., NIPS 14

FV-CNN **81.0**%
Cimpoi et al., CVPR 15

- Domain specific CNNs with texture models

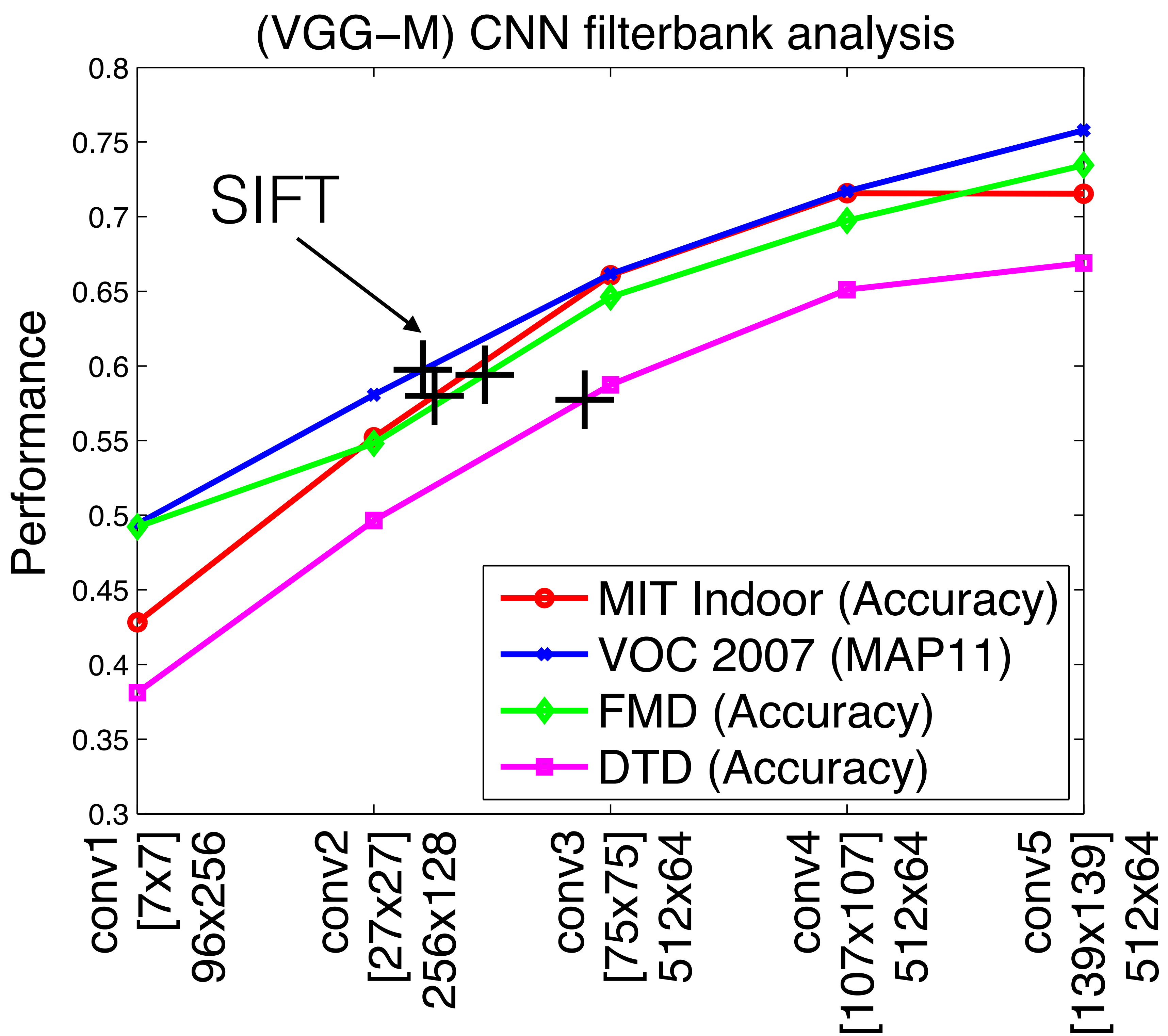
	Accuracy (%)		
CNN	FC-CNN	FV-CNN	FC+FV-CNN
PLACES	65.0	67.6	73.1
CAFFE	58.6	69.7	71.6
VGG-M	62.5	74.2	74.4
VGG-VD	67.6	**81.0**	80.3

(no data augmentation)

The advantage of domain specific training disappears with FV-CNN

Better CNNs lead to better performance and FV is better than FC

SIFT vs. CNN filter banks with FV



Cimpoi et al., CVPR 15

Learning features for texture models

- The features in the texture models (e.g. FV) are not learned
  - Hand crafted (e.g. SIFT), or CNN but trained with a different architecture (e.g. fully-connected layers)
  - The GMM parameters are learned in an unsupervised manner
- Can we learn the features for FV models?
  - Computing the gradients of the bilinear feature with respect to the feature **x** is nasty since both **f*A*** and **f*B*** depend on **x** via the GMM parameters

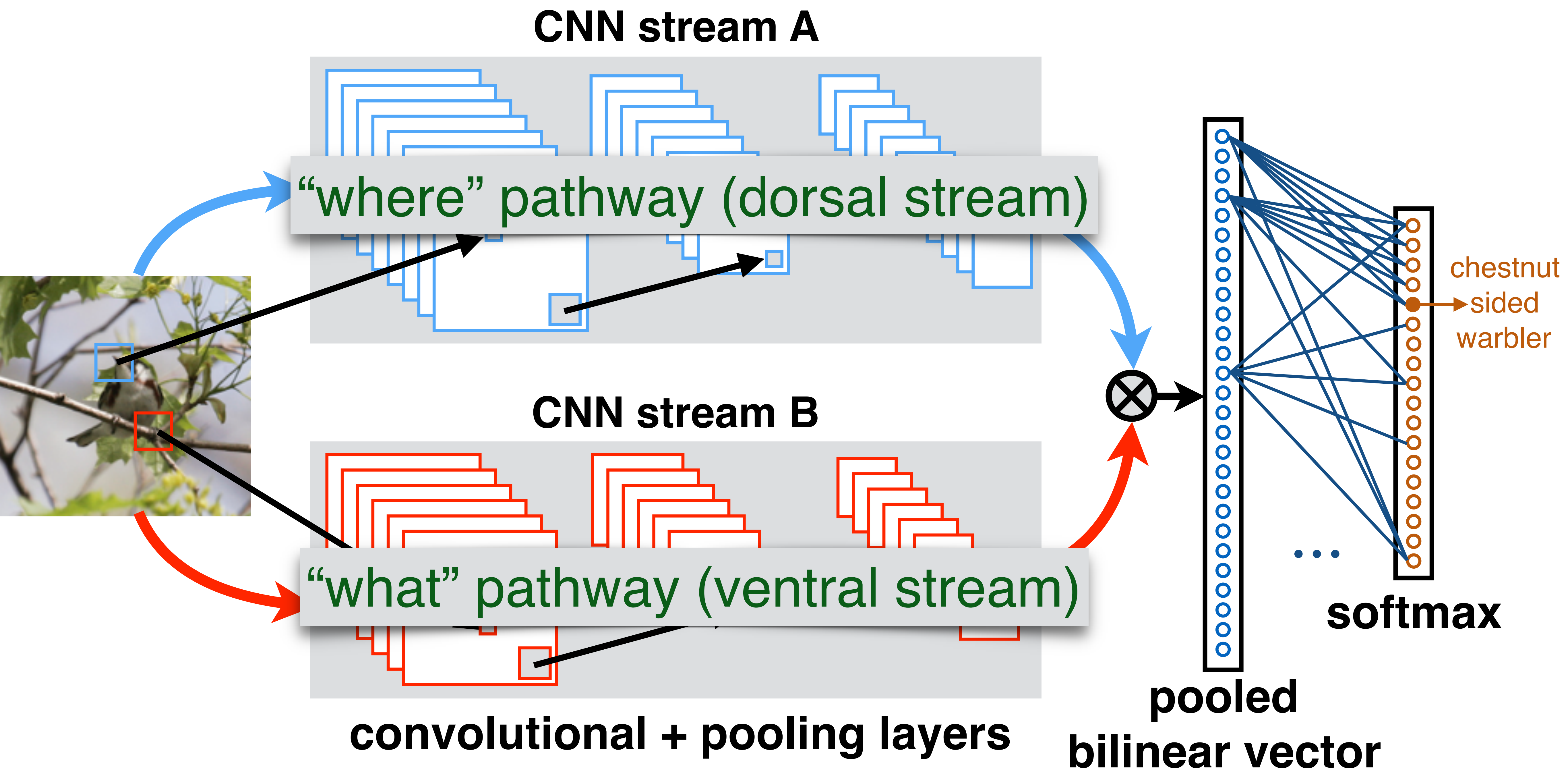
$$\alpha_i = \Sigma_i^{-\frac{1}{2}}(\mathbf{x} - \mu_i) \qquad \beta_i = \Sigma_i^{-1}(\mathbf{x}-\mu_i)\odot(\mathbf{x}-\mu_i)-1$$

$$\begin{aligned} f_A &= [\alpha_1\ \beta_1; \alpha_2\ \beta_2; \ldots; \alpha_k\ \beta_k] \\ f_B &= \mathrm{diag}(\eta(\mathbf{x})) \end{aligned}$$

  - Hard to compute the gradients
    - ➡ Partial attempt : Sydorov et al. [CVPR14] learn parameters of the GMM for FV-SIFT discriminatively but *not* the features themselves

Bilinear CNN model

- Generalization: decouple f_A and f_B by using separate feature functions



http://arxiv.org/abs/1504.07889

Bilinear CNN model training

- Back-propagation though the bilinear layer is easy

$$\frac{d\ell}{dA} \leftarrow B\left(\frac{d\ell}{d\mathbf{z}}\frac{d\mathbf{z}}{d\mathbf{y}}\frac{d\mathbf{y}}{d\mathbf{x}}\right)^T$$

$$\mathbf{x} = A^T B \xrightarrow{\text{sqrt}} \mathbf{y} \xrightarrow{\ell_2} \mathbf{z}$$

$$\frac{d\ell}{dB} \leftarrow A\left(\frac{d\ell}{d\mathbf{z}}\frac{d\mathbf{z}}{d\mathbf{y}}\frac{d\mathbf{y}}{d\mathbf{x}}\right)$$

- Allows end-to-end training
- Added two normalization layers inspired by “improved Fisher vector” [Perronnin et al., 10]
  - Square-root normalization $(\mathbf{y} \leftarrow \operatorname{sign}(\mathbf{x})\sqrt{|\mathbf{x}|})$
  - l_2 normalization $(\mathbf{z} \leftarrow \mathbf{y}/||\mathbf{y}||_2)$
  - Both these improve performance (see arXiv report for details)

http://arxiv.org/abs/1504.07889

Experiments

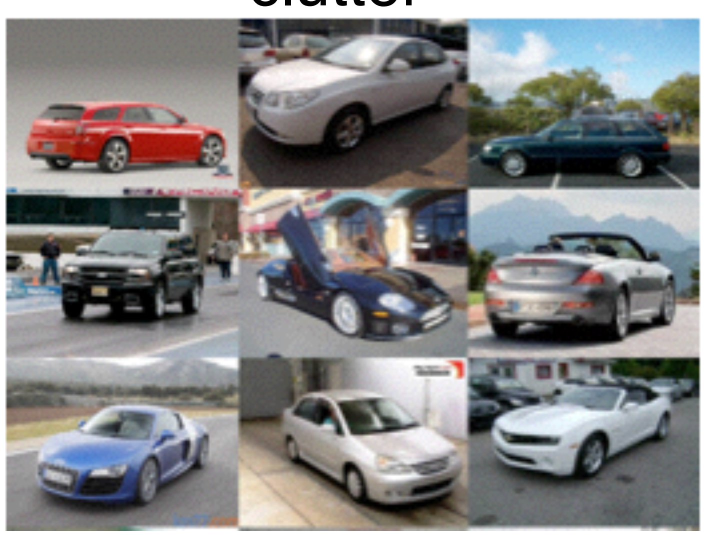
- We consider two CNN models initialized from ImageNet
  - VGG-M (5 convolutional layers + 2 fully connected layers)
  - VGG-D (13 convolutional layers + 2 fully connected layers)
- Methods considered in addition to the state-of-the-art:
  - FV-SIFT: Fisher-vector with SIFT features
  - FC-CNN: Features from the penultimate layer of a CNN
  - FV-CNN: Fisher-vector with CNN features [Cimpoi et al., CVPR 15]
  - B-CNN: Bilinear model with two CNNs feature extractors
- Trained using image labels only (no part or bounding-box annotations)
- Datasets:

small, clutter

CUB 200-2011
200 species, 11,788 images

FGVC Aircraft
100 variants, 10,000 images

clutter



Stanford cars
196 models, 16,185 images

Results: Birds classification

- Per-image accuracy on CUB 200-2011 dataset
- Setting: provided with only the image at test time

“shape” models

“texture” models (orderless)

accuracy improves

Method	w/o ft	w/ ft
FV-SIFT	18.8	-
FC-CNN (M)	52.7	
FC-CNN (D)	61.0	
FV-CNN (M)	61.1	
FV-CNN (D)	71.3	
B-CNN (M,M)	72.0	
B-CNN (M,D)	80.1	
B-CNN (D,D)	80.1	
SoTA	**84.1** [4], 66.7 [1], 73.9 [2], 75.7 [3]	

[1] Multi-scale FV-CNN (D), Cimpoi et al., CVPR 15
[2] Part-based R-CNNs, Zhang et al., ECCV 14 (+ part bounding-boxes during training)
[3] Pose normalized CNNs, Branson et al., BMVC 14 (+ landmarks during training)
[4] Spatial Transformer Networks, Jaderberg et al., NIPS 2015

Results: Birds classification

- Per-image accuracy on CUB 200-2011 dataset
- Setting: provided with only the image at test time

"shape" models

"texture" models (orderless)

accuracy improves

Method	w/o ft	w/ ft
FV-SIFT	18.8	-
FC-CNN (M)	52.7	58.8
FC-CNN (D)	61.0	70.4
FV-CNN (M)	61.1	
FV-CNN (D)	71.3	
B-CNN (M,M)	72.0	
B-CNN (M,D)	80.1	
B-CNN (D,D)	80.1	
SoTA	**84.1** [4], 66.7 [1], 73.9 [2], 75.7 [3]	

fine-tuning helps

[1] Multi-scale FV-CNN (D), Cimpoi et al., CVPR 15
[2] Part-based R-CNNs, Zhang et al., ECCV 14 (+ part bounding-boxes during training)
[3] Pose normalized CNNs, Branson et al., BMVC 14 (+ landmarks during training)
[4] Spatial Transformer Networks, Jaderberg et al., NIPS 2015

Results: Birds classification

- Per-image accuracy on CUB 200-2011 dataset
- Setting: provided with only the image at test time

"shape" models

"texture" models (orderless)

accuracy improves

Method	w/o ft	w/ ft
FV-SIFT	18.8	-
FC-CNN (M)	52.7	58.8
FC-CNN (D)	61.0	70.4
FV-CNN (M)	61.1	64.1
FV-CNN (D)	71.3	74.7
B-CNN (M,M)	72.0	
B-CNN (M,D)	80.1	
B-CNN (D,D)	80.1	
SoTA	**84.1** [4], 66.7 [1], 73.9 [2], 75.7 [3]	

indirect fine-tuning helps

[1] Multi-scale FV-CNN (D), Cimpoi et al., CVPR 15
[2] Part-based R-CNNs, Zhang et al., ECCV 14 (+ part bounding-boxes during training)
[3] Pose normalized CNNs, Branson et al., BMVC 14 (+ landmarks during training)
[4] Spatial Transformer Networks, Jaderberg et al., NIPS 2015

Results: Birds classification

- Per-image accuracy on CUB 200-2011 dataset
- Setting: provided with only the image at test time

"shape" models

"texture" models (orderless)

accuracy improves

Method	w/o ft	w/ ft
FV-SIFT	18.8	-
FC-CNN (M)	52.7	58.8
FC-CNN (D)	61.0	70.4
FV-CNN (M)	61.1	64.1
FV-CNN (D)	71.3	74.7
B-CNN (M,M)	72.0	78.1
B-CNN (M,D)	80.1	**84.1**
B-CNN (D,D)	80.1	84.0
SoTA	**84.1** [4], 66.7 [1], 73.9 [2], 75.7 [3]	

fine-tuning helps

[1] Multi-scale FV-CNN (D), Cimpoi et al., CVPR 15

[2] Part-based R-CNNs, Zhang et al., ECCV 14 (+ part bounding-boxes during training)

[3] Pose normalized CNNs, Branson et al., BMVC 14 (+ landmarks during training)

[4] Spatial Transformer Networks, Jaderberg et al., NIPS 2015

Results: Aircraft classification

- Per-image accuracy on FGVC aircraft dataset
- Setting: provided with only the image at test time

low clutter + big objects → localization is less important

Method	w/o ft	w/ ft
FV-SIFT	61.0	-
FC-CNN (M)	44.4	57.3
FC-CNN (D)	45.0	74.1
FV-CNN (M)	64.3	70.1
FV-CNN (D)	70.4	77.6
B-CNN (M,M)	72.7	77.9
B-CNN (M,D)	78.4	83.9
B-CNN (D,D)	76.8	**84.1**
SoTA	72.5 [1], 80.7 [2]	

"shape" models

"texture" models (orderless)

accuracy improves

fine-tuning helps (much more)

indirect fine-tuning helps

fine-tuning helps

[1] Symbiotic segmentation, Y. Chai et al., ICCV 15 (+ object bounding-boxes during training)
[2] Fisher vector SIFT++ , Gosselin et al., Pattern Recognition 14

Results: Cars classification

- Per-image accuracy on Stanford cars dataset
- Setting: provided with only the image at test time

more
clutter + small objects
→ localization is
important

Method	w/o ft	w/ ft
FV-SIFT	59.2	-
FC-CNN (M)	37.3	58.6
FC-CNN (D)	36.5	79.8
FV-CNN (M)	70.8	77.2
FV-CNN (D)	75.2	85.7
B-CNN (M,M)	77.8	86.5
B-CNN (M,D)	83.9	91.3
B-CNN (D,D)	82.9	90.6
SoTA	**92.6** [1], 82.7 [2], 78.0 [3]	

"shape" models

"texture" models (orderless)

accuracy improves

fine-tuning helps (much more)

indirect fine-tuning helps

fine-tuning helps

[1] Fine-grained recognition without part annotations, Krause et al., CVPR 15 (+ object bounding-boxes during training)

[2] Fisher vector SIFT++ , Gosselin et al., Pattern Recognition 14

[3] Symbiotic segmentation, Y. Chai et al., ICCV 15 (+ object bounding-boxes during training)

Most confused birds

Most confused aircrafts

C-47

DC-3

Design and development [edit]

The C-47 differed from the civilian DC-3 in numerous modifications, including being fitted with a cargo door and strengthened floor, along with a shortened tail cone for glider-towing shackles, and an astrodome in the cabin roof.[3][4]

During World War II, the armed forces of many countries used the C-47 and modified DC-3s for the transport of troops, cargo, and wounded. The U.S. Naval designation was R4D. More than 10,000 aircraft were produced in Long Beach and Santa Monica, California and Oklahoma City, Oklahoma. Between March 1943 and August 1945 the Oklahoma City plant produced 5,354 C-47s.[2][5]

source: wikipedia

737-300

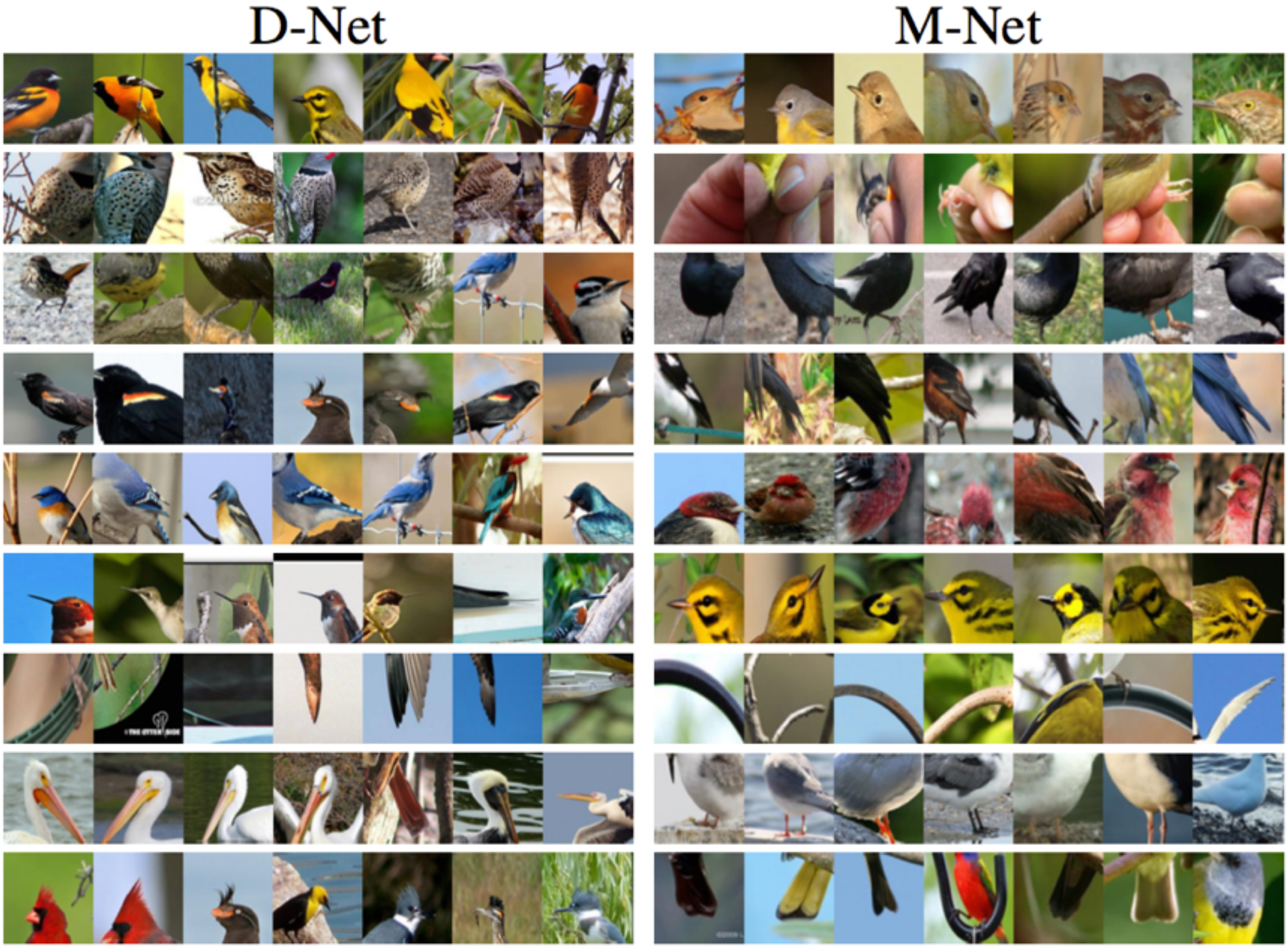
737-500

767-200

767-300

Most confused cars

What is learned [birds]

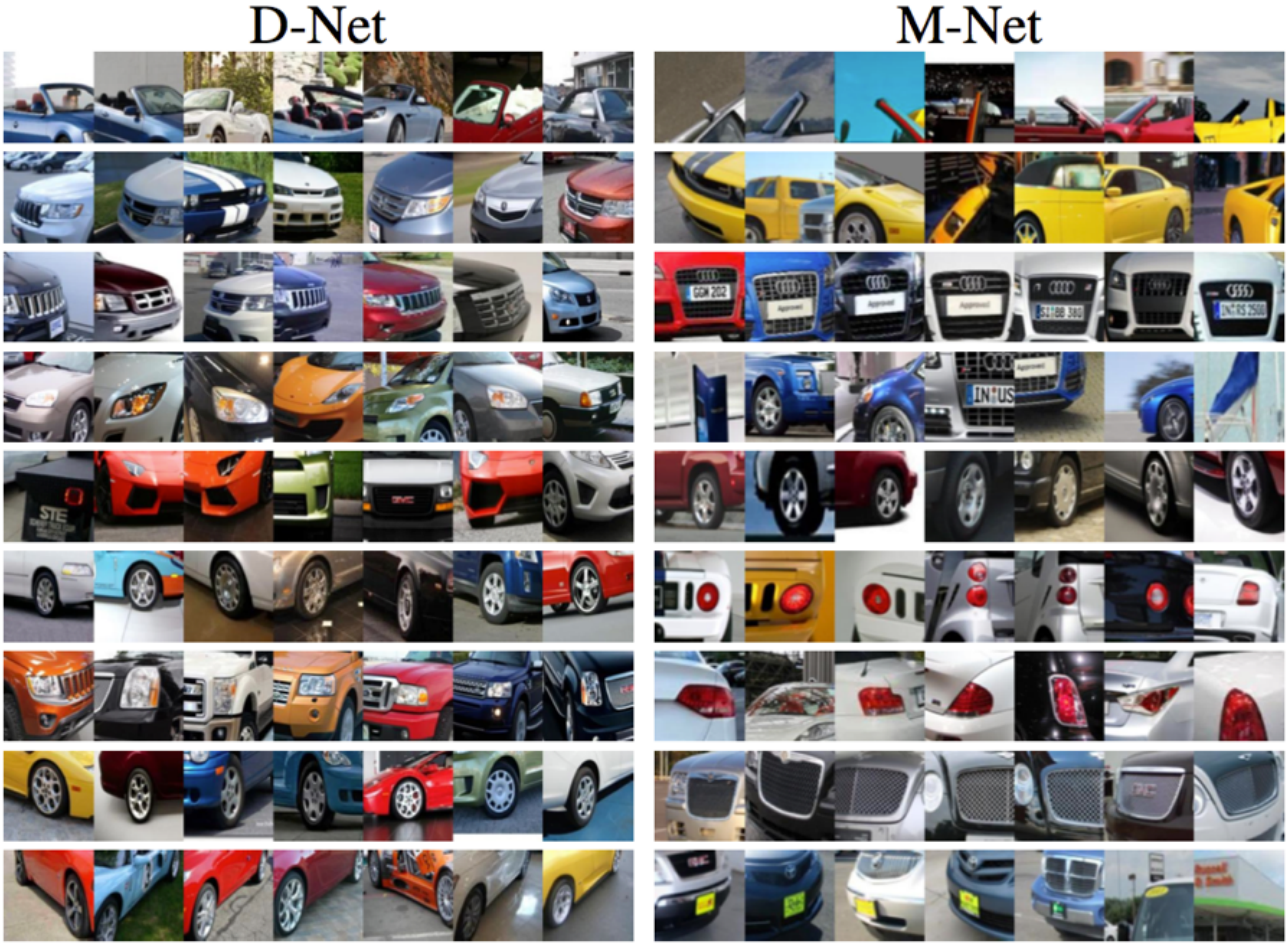


What is learned [airplanes]

D-Net

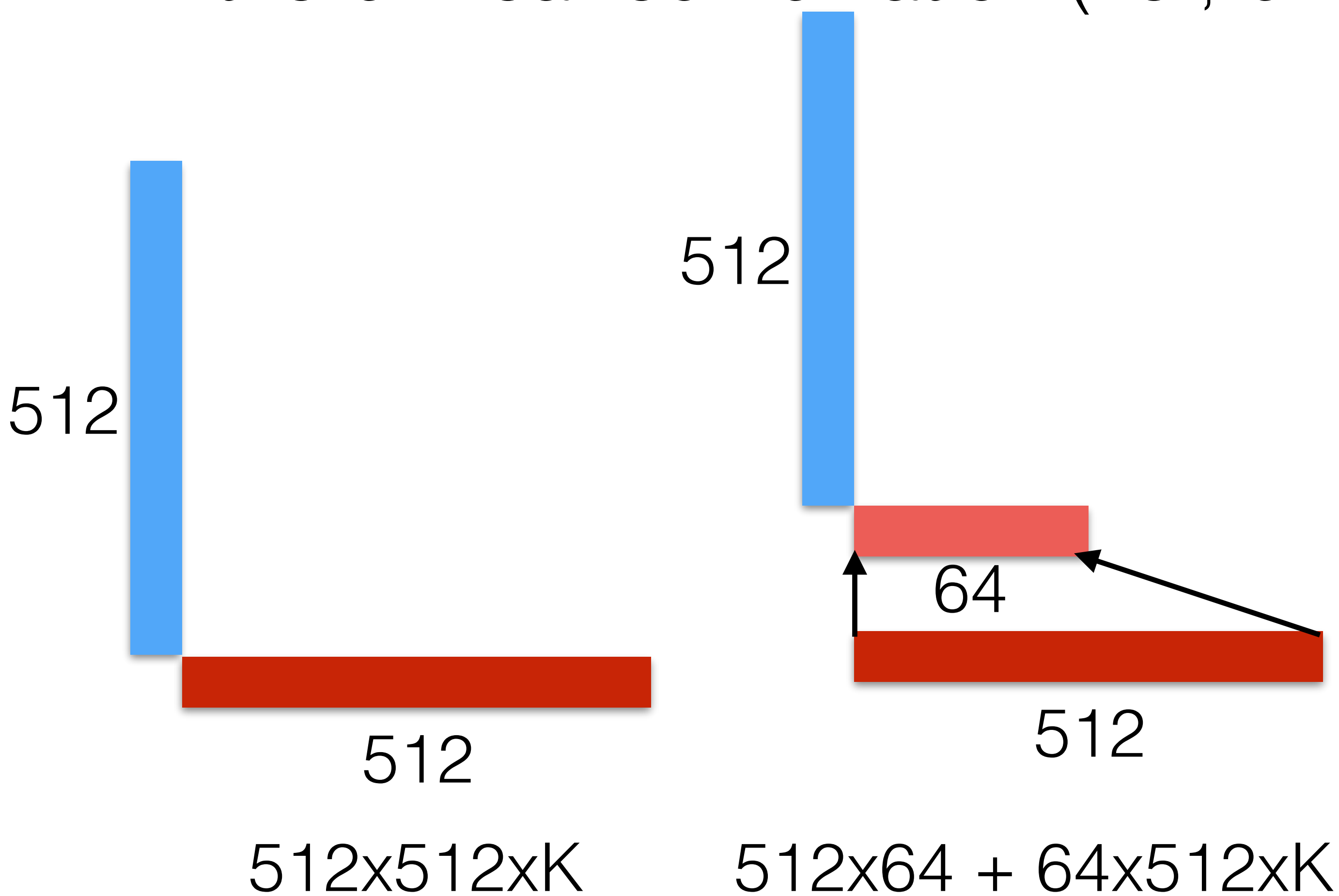
M-Net

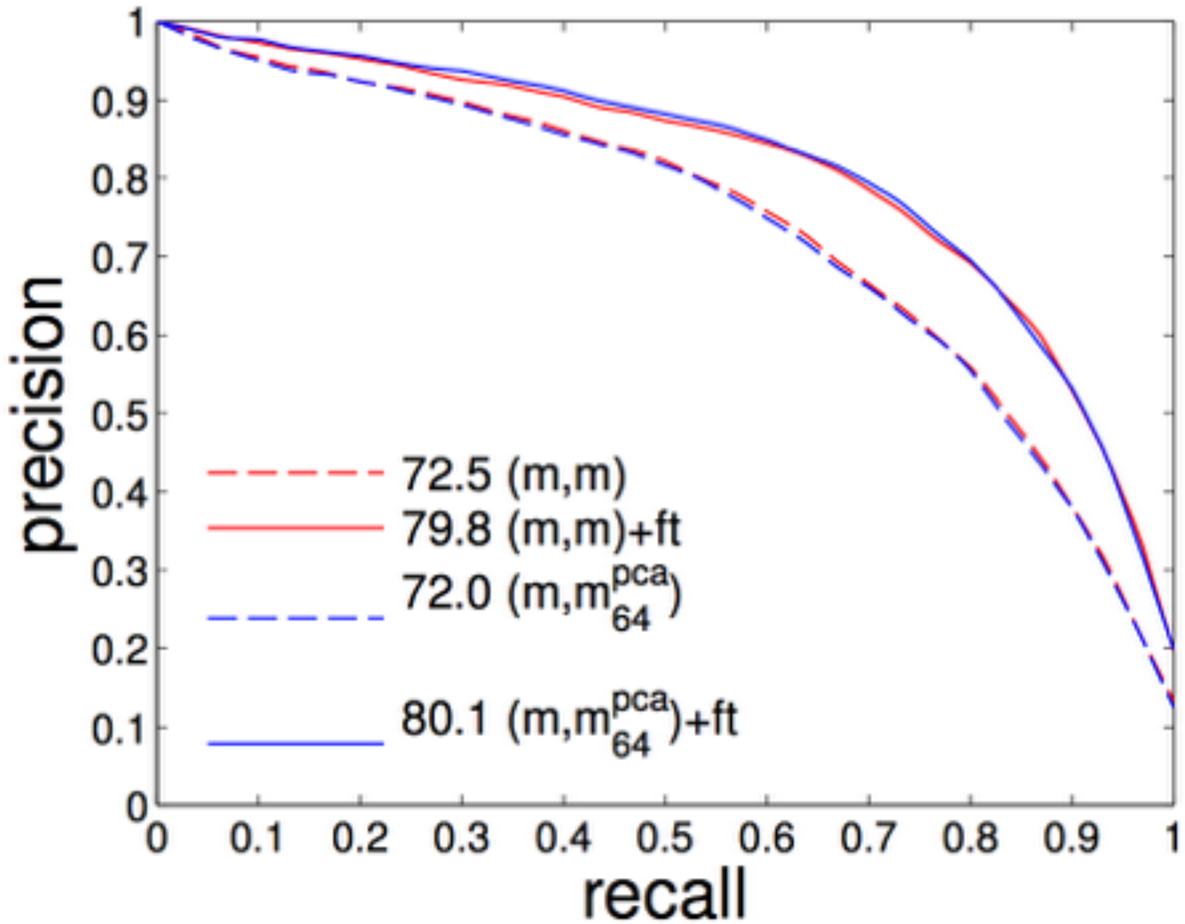
What is learned [cars]



Symmetric vs. asymmetric models

- The B-CNN(M,M) is symmetric — fine-tuning will keep them symmetric
  - 2x faster than asymmetric model but sub-optimal
- Breaking the symmetry
  - Dropout — made it worse
  - Dimensionality reduction — reduce the output of one CNN before the bilinear combination (i.e., bilinear classifier)





Summary

- **Bilinear CNN models :**
  - generalize both *texture methods* and *part-based methods*
  - training is requires *only* image labels
  - *fairly efficient* at test time — our MatConvNet based B-CNN (D, M) runs at 8 fps on a Tesla K40 GPU
  - code is available at http://vis-www.cs.umass.edu/bcnn

- **Inverting B-CNN (D,D) :**
  - images that match bilinear responses of **relu1_1, relu2_1, relu3_1, relu4_1, relu5_1** layers of **vgg-verydeep-16** model

"equivalent" images

References

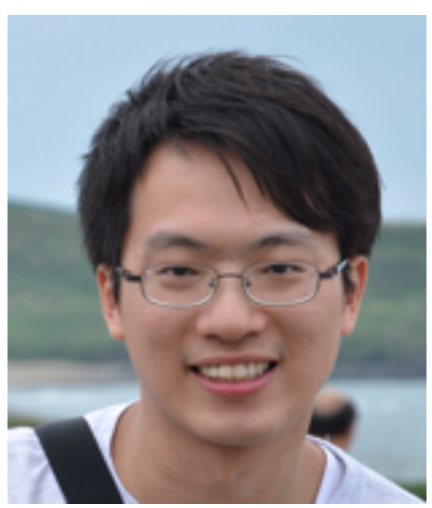
- **Deep Filter Banks for Texture Recognition and Segmentation,** Mircea Cimpoi, Subhransu Maji, Andrea Vedaldi, CVPR 2015

Mircea Cimpoi

Andrea Vedaldi

code available!

- **Bilinear CNN Models for Fine-grained Visual Recognition,** Tsung-Yu Lin, Aruni Roy Chowdhury, Subhransu Maji, ICCV 2015



Tsung-yu Lin

Aruni RoyChowdhury

code available!