L5: Transfer Learning with ConvNets

Xiaopeng HONG

CMV, OU

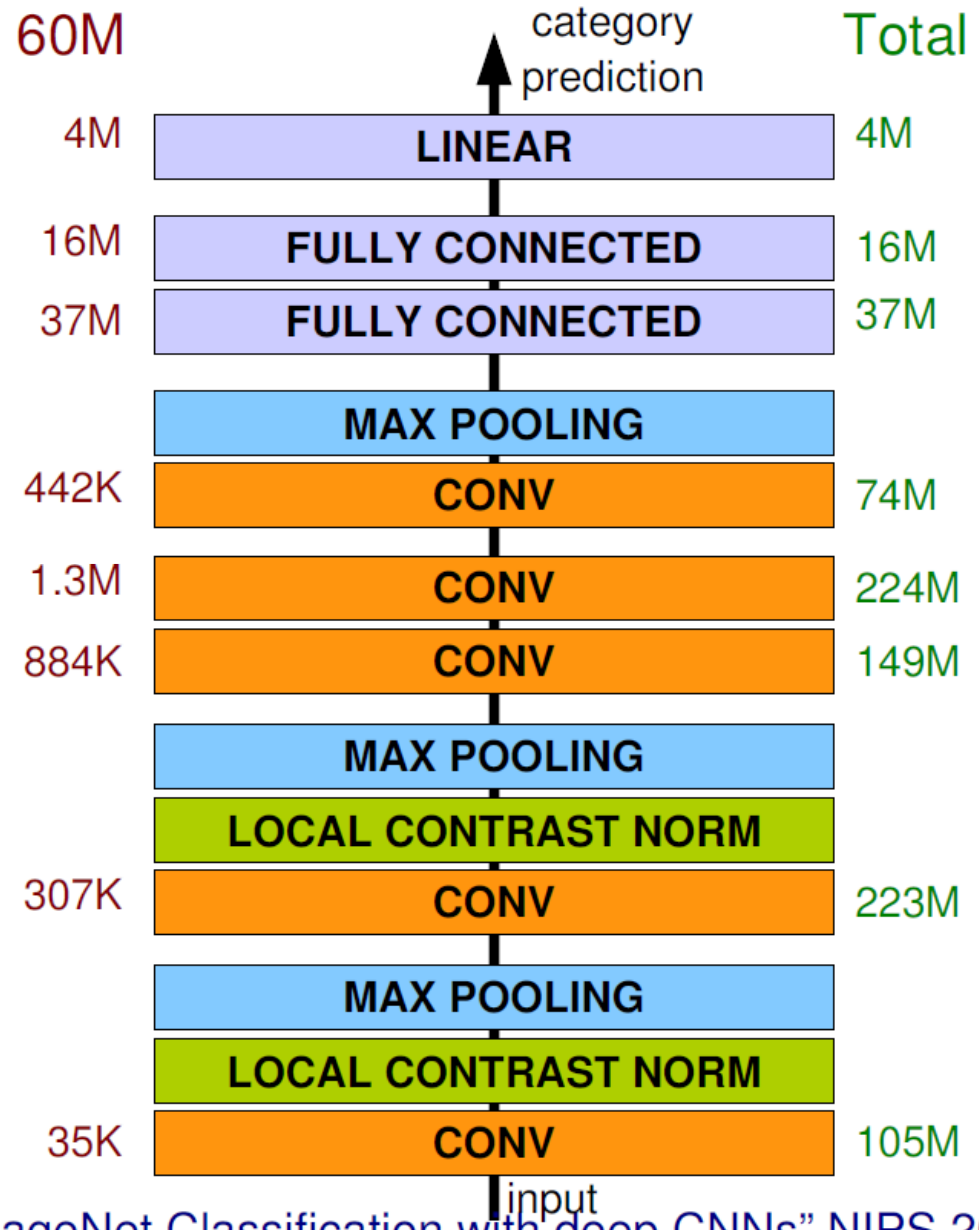
Acknowledgement

- A big portion of Slides are stolen from or inspired by
  - Feifei Li & Andrei Karpathy, and
  - Maxime Oquab's talks

Architecture

Total nr. params: 60M

Total nr. flops: 832M



Krizhevsky et al. “ImageNet Classification with deep CNNs” NIPS 2012

The 2014 ImageNet Large Scale Visual Recognition Challenge (ILSVRC)

- 1.2 million images in the training set, each labeled with one of 1000 categories
- The 100,000 test set images

Funny Q/A case

- Coffee room question:
  - *“Deep learning. How well do you think it would work for your computer vision problem?”*
- You may answer like:
  - “Pity I have neither the time, GPU programming skills nor large amount of labelled data to train my own network to quickly find out the answer”.

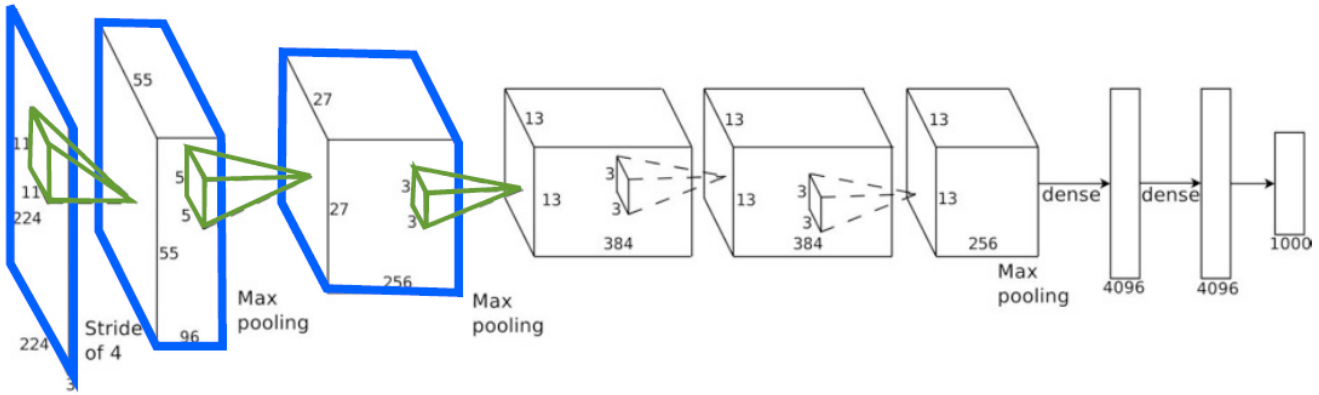
CNN Features off-the-shelf: an Astounding Baseline for Recognition

Ali Sharif Razavian Hossein Azizpour Josephine Sullivan Stefan Carlsson

An example

8-layer NN

[Krizhevsky et al.]



60 million parameters :

- ImageNet (1.2M images) : OK
- Pascal VOC (10k images) : ?

With limited training data, however, fully-supervised deep architectures with the representational capacity of (Krizhevsky et al., 2012) will generally dramatically overfit the training data. In fact, many conventional visual recognition challenges have tasks with few training examples; e.g., when a user is defining a category "on-the-fly" using specific examples, or for fine-grained recognition challenges (Welinder et al., 2010), attributes (Bourdev et al., 2011), and/or domain adaptation (Saenko et al., 2010).

DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition

---

Jeff Donahue*, Yangqing Jia*, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, Trevor Darrell

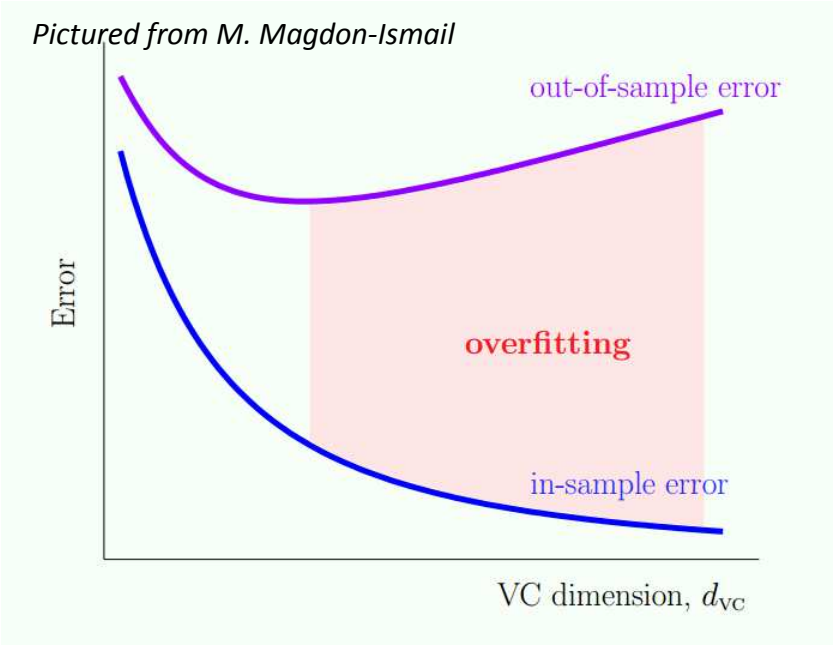
Overfitting

- Fitting the data more than is warranted

Overfitting generally occurs when a model is excessively complex, such as having too many parameters relative to the number of observations.
A model that has been overfit will generally have poor predictive performance, as it can exaggerate minor fluctuations in the data.

By Wikipedia

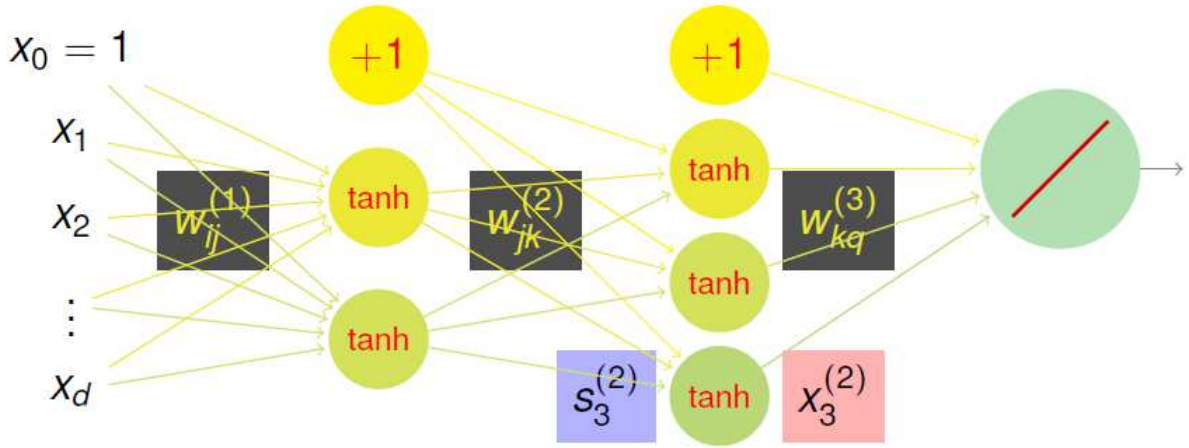
Pictured from M. Magdon-Ismail



Going for lower and lower E_in but results in higher and higher E_out

VC Dimension of Neural Network Model

roughly, with tanh-**like transfer functions**:
$d_{VC} = O(VD)$ where V = # of neurons, D = # of weights



- pros: can **approximate 'anything'** if enough neurons (V large)
- cons: can **overfit** if too many neurons

NNet: **watch out for overfitting!**

Usually

- We need a lot of data if we want to train/use CNNs

Usually

- We need a lot of data if we want to train/use CNNs

- What would happen if we refused to accept?

We Wondered…

- not whether one could train a deep network specifically for a given task,
- but if the features extracted by a deep network - one carefully trained on the diverse ImageNet database to perform the specific task of image classification
- could be exploited for a wide variety of vision tasks

More specifically

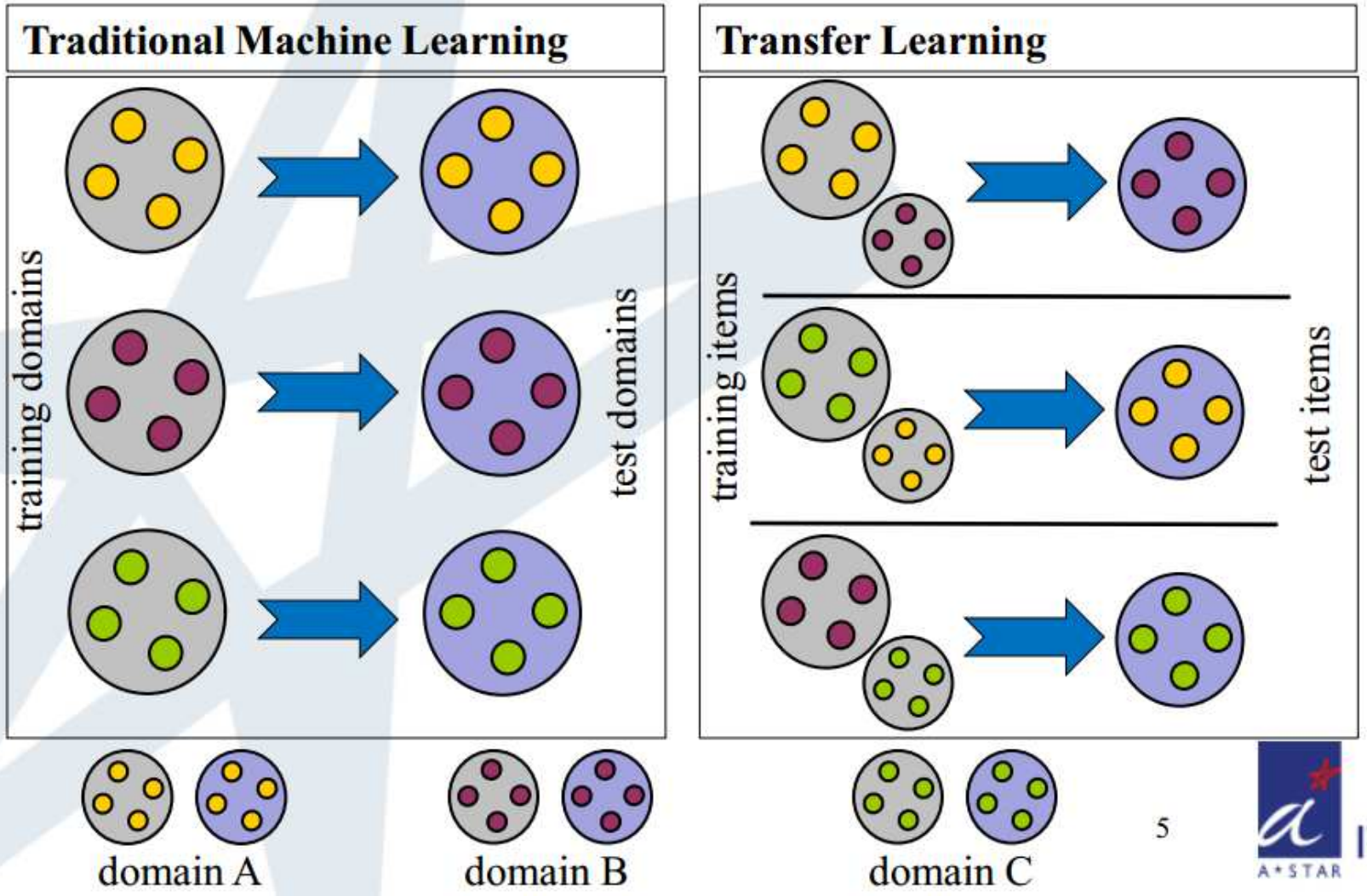
- We evaluate whether features extracted from the activation of a deep convolutional network trained in a fully supervised fashion on a large, fixed set of object recognition tasks can be repurposed to novel generic tasks.

http://www.forthillcompany.com/learning-alert-44-learning-transfer-summit-highlights/

Transfer learning

- Transfer learning aims to transfer knowledge between related source and target domains [Pan & Yang, TKDE,2010].
  - In computer vision, some try to overcome the deficit of training samples for some categories by adapting classifiers trained for other categories.
  - Other methods aim to cope with different data distributions in the source and target domains for the same categories, e.g. due to lighting, background and view-point variations.
- These and other related methods adapt classifiers or kernels while using standard image features.

Transfer learning



Strategy

1. Train on Imagenet

Strategy

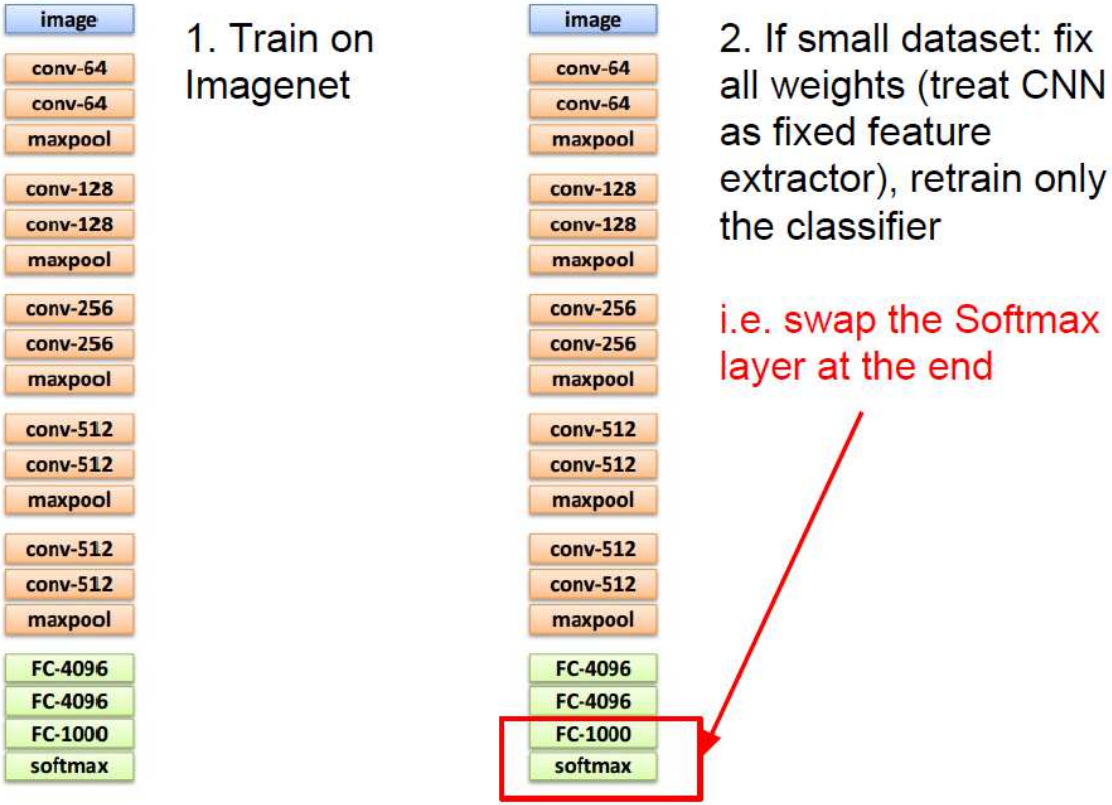


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
FC-1000
softmax
1. Train on Imagenet
image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
FC-1000
softmax
2. If small dataset: fix all weights (treat CNN as fixed feature extractor), retrain only the classifier
i.e. swap the Softmax layer at the end

Strategy

1. Train on Imagenet

2. If small dataset: fix all weights (treat CNN as fixed feature extractor), retrain only the classifier

i.e. swap the Softmax layer at the end

3. If you have medium sized dataset, **"finetune"** instead: use the old weights as initialization, train the full network or only some of the higher layers

retrain bigger portion of the network, or even all of it.

Strategy

1. Train on Imagenet

2. If small dataset: fix all weights (treat CNN as fixed feature extractor), retrain only the classifier

i.e. swap the Softmax layer at the end

3. If you have medium sized dataset, **"finetune"** instead: use the old weights as initialization, train the full network or only some of the higher layers

retrain bigger portion of the network, or even all of it.

tip: use only ~1/10th of the original learning rate in finetuning to player, and ~1/100th on intermediate layers

Transfer Learning & Fine tune

- Retrain only classifiers
- Add one or two full connected layers
- Fine-tune more

Transfer Learning & Fine tune

- Retrain only classifiers
- Add one or two full connected layers
- Fine-tune more

CNN Features off-the-shelf: an Astounding Baseline for Recognition

Ali Sharif Razavian Hossein Azizpour Josephine Sullivan Stefan Carlsson

---

DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition

---

Jeff Donahue*, Yangqing Jia*, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, Trevor Darrell

Why retrain the FC-Softmax layer?

Too specific to the source task!!!

CNN Features off-the-shelf: an Astounding Baseline for Recognition

Ali Sharif Razavian Hossein Azizpour Josephine Sullivan Stefan Carlsson

- the CNN features used are trained only using ImageNet data
- the simple classifiers are trained using images specific to the task's dataset

CNN Features off-the-shelf: an Astounding Baseline for Recognition

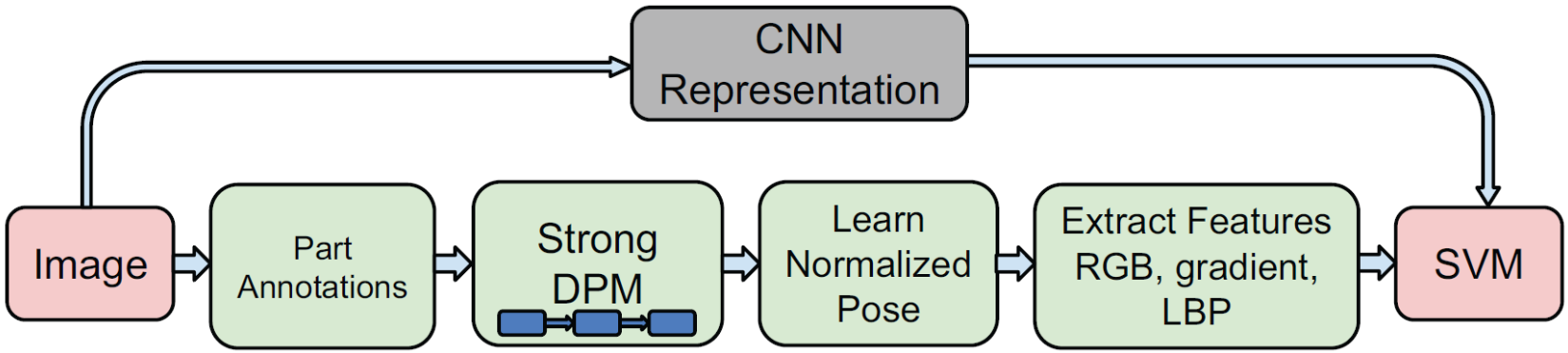
Ali Sharif Razavian Hossein Azizpour Josephine Sullivan Stefan Carlsson

- Use the publicly available trained CNN called OverFeat
- The structure of the network follows that of AlexNet.
  - The convolutional layers each contain 96 to 1024 kernels of size 3 $\times$3 to 7 $\times$7.
  - Half-wave rectification is used as the nonlinear activation function.
  - Max pooling kernels of size 3 $\times$3 and 5 $\times$5 are used at different layers to build robustness to intra-class deformations.
- We used the “large” version of the OverFeat network.
  - It takes as input color images of size 221 $\times$221.
- Train on ImageNet ILSVRC 2013 data

Implementation details

- The feature vector is further *L*2 normalized to unit length for all the experiments.
- **Direct use**
  - use the 4096 dimensional feature vector in combination with a Support Vector Machine (SVM) to solve different classification tasks (CNN-SVM).
- **further augment**
  - the training set by adding cropped and rotated samples and doing componentwise power transform and report separate results (CNNaug+SVM).

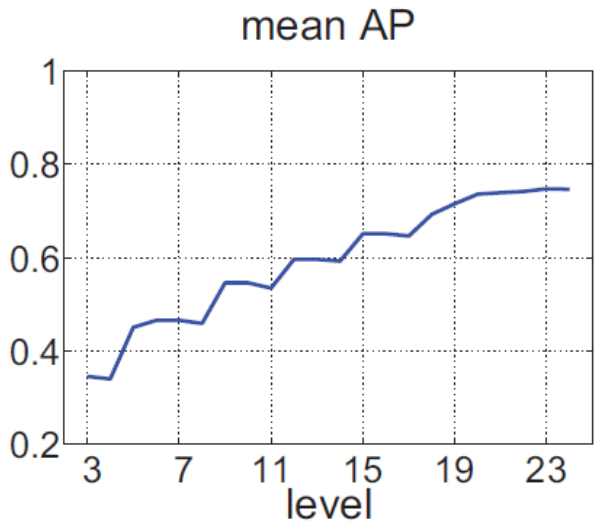
The off-the-shelf flowchart



PASCAL VOC2007

	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
GHM[8]	76.7	74.7	53.8	72.1	40.4	71.7	83.6	66.5	52.5	57.5	62.8	51.1	81.4	71.5	86.5	36.4	55.3	60.6	80.6	57.8	64.7
AGS[11]	82.2	83.0	58.4	76.1	**56.4**	**77.5**	**88.8**	69.1	**62.2**	61.8	64.2	51.3	**85.4**	**80.2**	91.1	48.1	61.7	**67.7**	86.3	70.9	71.1
NUS[39]	82.5	79.6	64.8	73.4	54.2	75.0	77.5	79.2	46.2	62.7	41.4	74.6	85.0	76.8	91.1	53.9	61.0	67.5	83.6	70.6	70.5
CNN-SVM	88.5	81.0	83.5	82.0	42.0	72.5	85.3	81.6	59.9	58.5	66.5	77.8	81.8	78.8	90.2	54.8	71.1	62.6	87.2	71.8	73.9
CNNaug-SVM	**90.1**	**84.4**	**86.5**	**84.1**	48.4	73.4	86.7	**85.4**	61.3	**67.6**	**69.6**	**84.0**	**85.4**	80.0	**92.0**	**56.9**	**76.7**	67.3	**89.1**	**74.9**	**77.2**

Table 1: **Pascal VOC 2007 Image Classification Results** compared to other methods which also use training data outside VOC. The CNN representation is not tuned for the Pascal VOC dataset. However, GHM [8] learns from VOC a joint representation of bag-of-visual-words and contextual information. AGS [11] learns a second layer of representation by clustering the VOC data into subcategories. NUS [39] trains a codebook for the SIFT, HOG and LBP descriptors from the VOC dataset. Oquab *et al*. [29] fixes all the layers trained on ImageNet then it adds and optimizes two fully connected layers on the VOC dataset and achieves better results (**77.7**) indicating the potential to boost the performance by further adaptation of the representation to the target task/dataset.

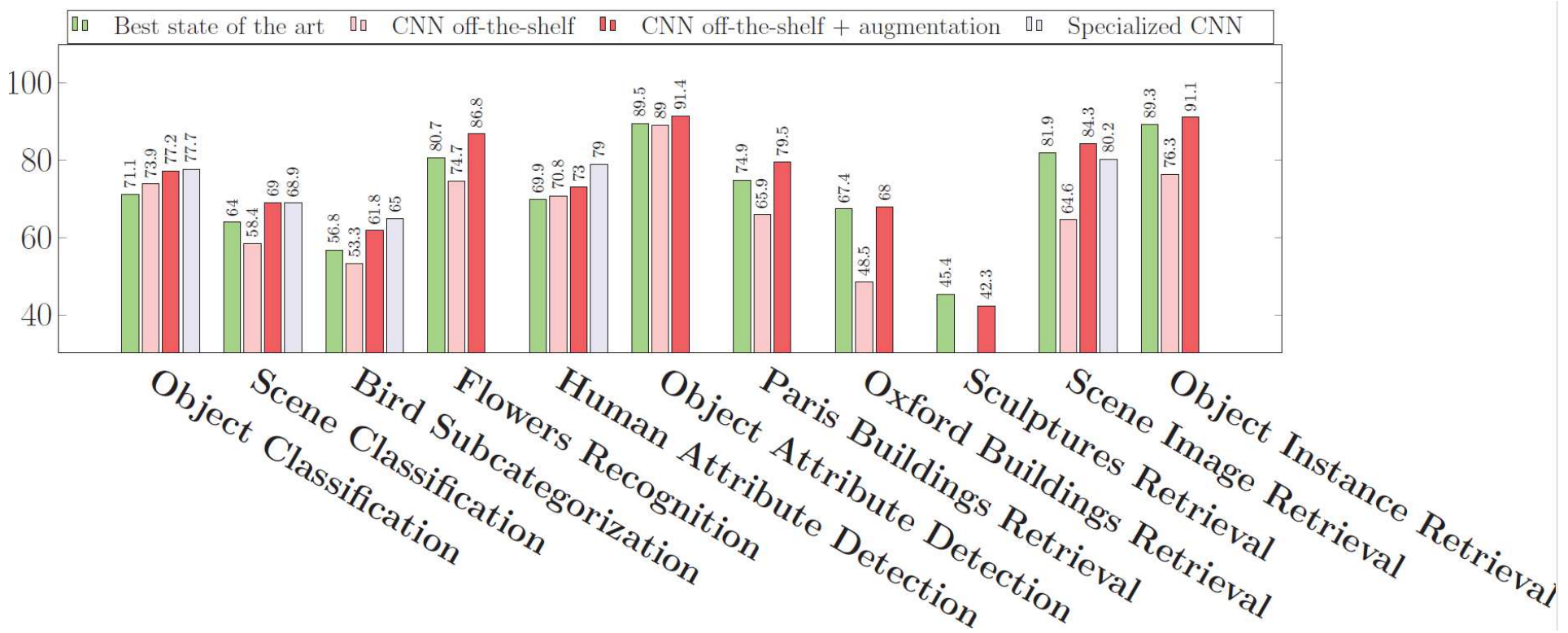


MIT-67 indoor scene dataset

Method	mean Accuracy
ROI + Gist[36]	26.1
DPM[30]	30.4
Object Bank[24]	37.6
RBow[31]	37.9
BoP[21]	46.1
miSVM[25]	46.4
D-Parts[40]	51.4
IFV[21]	60.8
MLrep[9]	64.0
CNN-SVM	58.4
CNNaug-SVM	**69.0**
CNN(AlexConvNet)+multiscale pooling [16]	68.9

Table 2: **MIT-67 indoor scenes dataset**. The MLrep [9] has a fine tuned pipeline which takes weeks to select and train various part detectors. Furthermore, Improved Fisher Vector (IFV) representation has dimensionality larger than 200K. [16] has very recently tuned a multi-scale orderless pooling of CNN features (off-the-shelf) suitable for certain tasks. With this simple modification they achieved significant average classification accuracy of **68.88**.

Results on Eleven Databases



DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition

Jeff Donahue*, Yangqing Jia*, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, Trevor Darrell

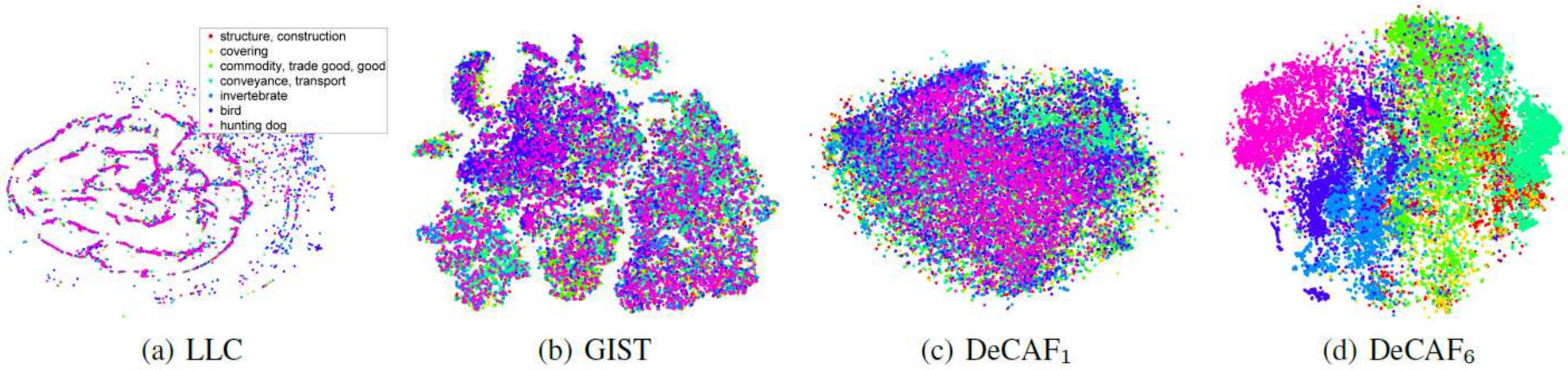
- Do features extracted from the CNN generalize to other datasets?
- How do these features perform versus depth?

DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition

Jeff Donahue*, Yangqing Jia*, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, Trevor Darrell

- Alex Net Stucture
- Trained on ILSVRC-2012 data

How good DeCAF?



(a) LLC (b) GIST (c) $DeCAF_1$ (d) $DeCAF_6$

Figure 1. This figure shows several t-SNE feature visualizations on the ILSVRC-2012 validation set. (a) LLC , (b) GIST, and features derived from our CNN: (c) DeCAF1, the first pooling layer, and (d) DeCAF6, the second to last hidden layer

ILSVRC-2012 to SUN-397

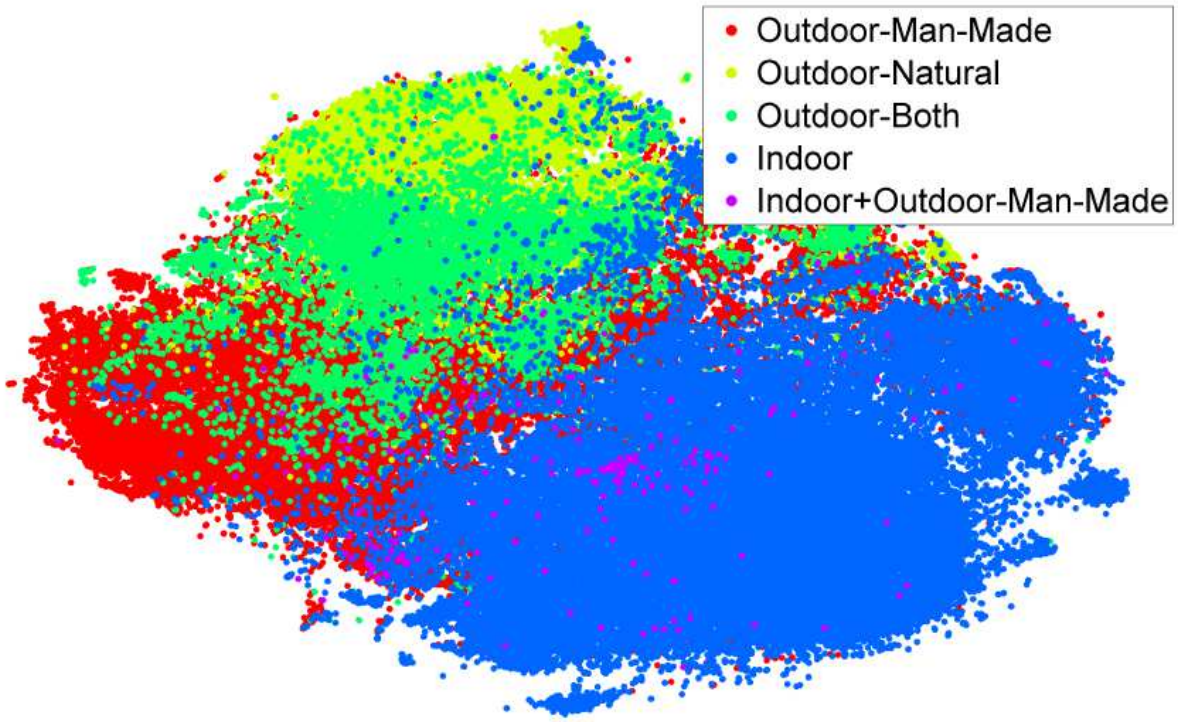
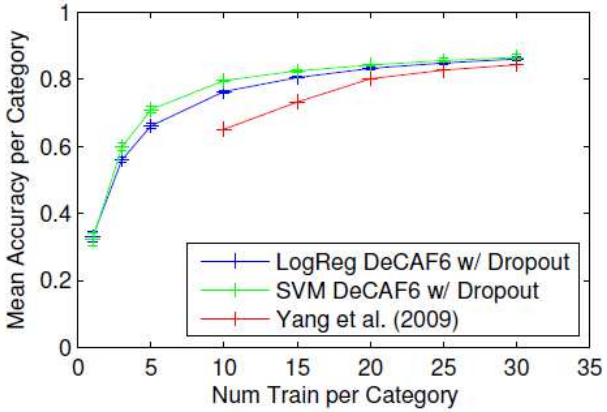


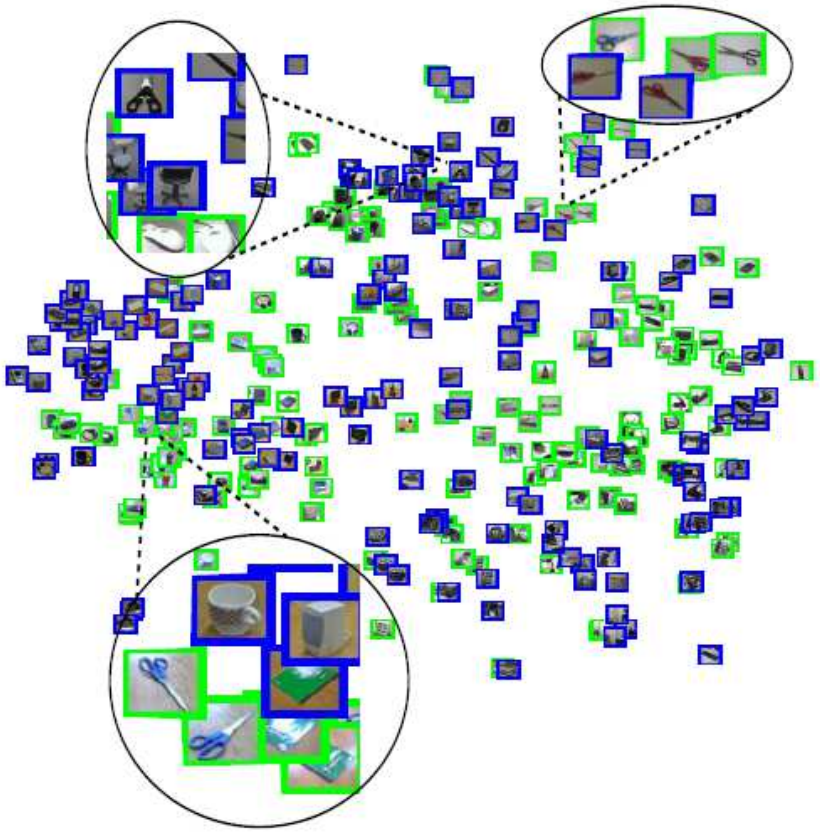
Figure 2. In this figure we show how our features trained on ILSVRC-2012 generalized to SUN-397 when considering semantic groupings of labels (best viewed in color).

Caltech-101

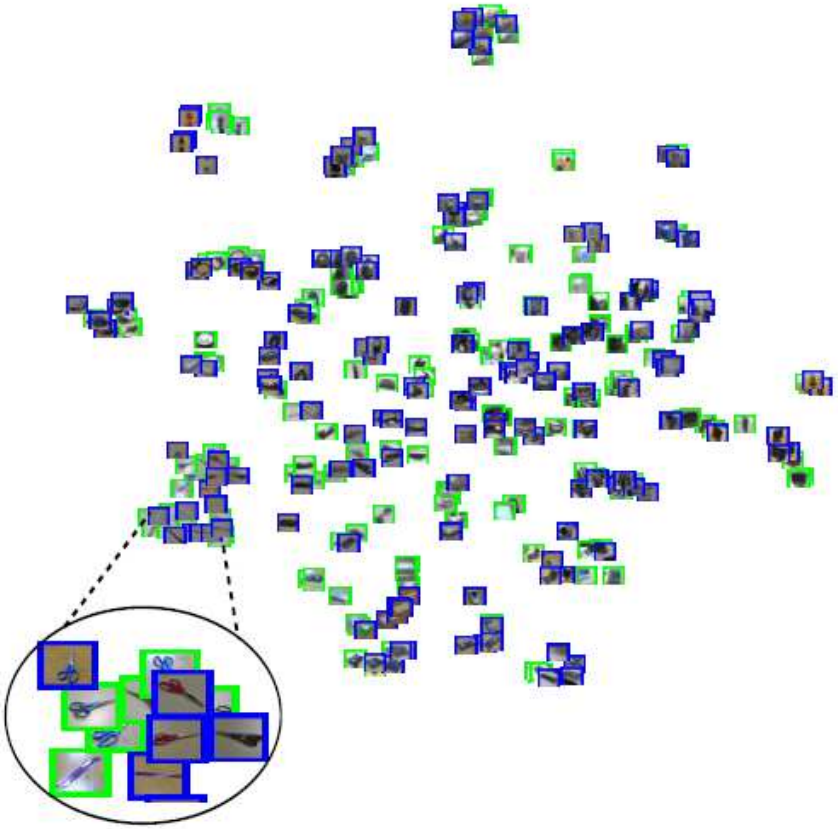
	$DeCAF_5$	$DeCAF_6$	$DeCAF_7$
LogReg	63.29 ± 6.6	84.30 ± 1.6	84.87 ± 0.6
LogReg with Dropout	-	86.08 ± 0.8	85.68 ± 0.6
SVM	77.12 ± 1.1	84.77 ± 1.2	83.24 ± 1.2
SVM with Dropout	-	$\mathbf{86.91 \pm 0.7}$	85.51 ± 0.9
Yang et al. (2009)		84.3	
Jarrett et al. (2009)		65.5	



Domain adaptation



(a) SURF features



(b) $DeCAF_6$

Caltech-UCSD bird dataset

Method	Accuracy
$DeCAF_6$	58.75
DPD + $DeCAF_6$	**64.96**
DPD (Zhang et al., 2013)	50.98
POOF (Berg & Belhumeur, 2013)	56.78

Table 2. Accuracy on the Caltech-UCSD bird dataset.

SUN-397 large-scale scene recognition database

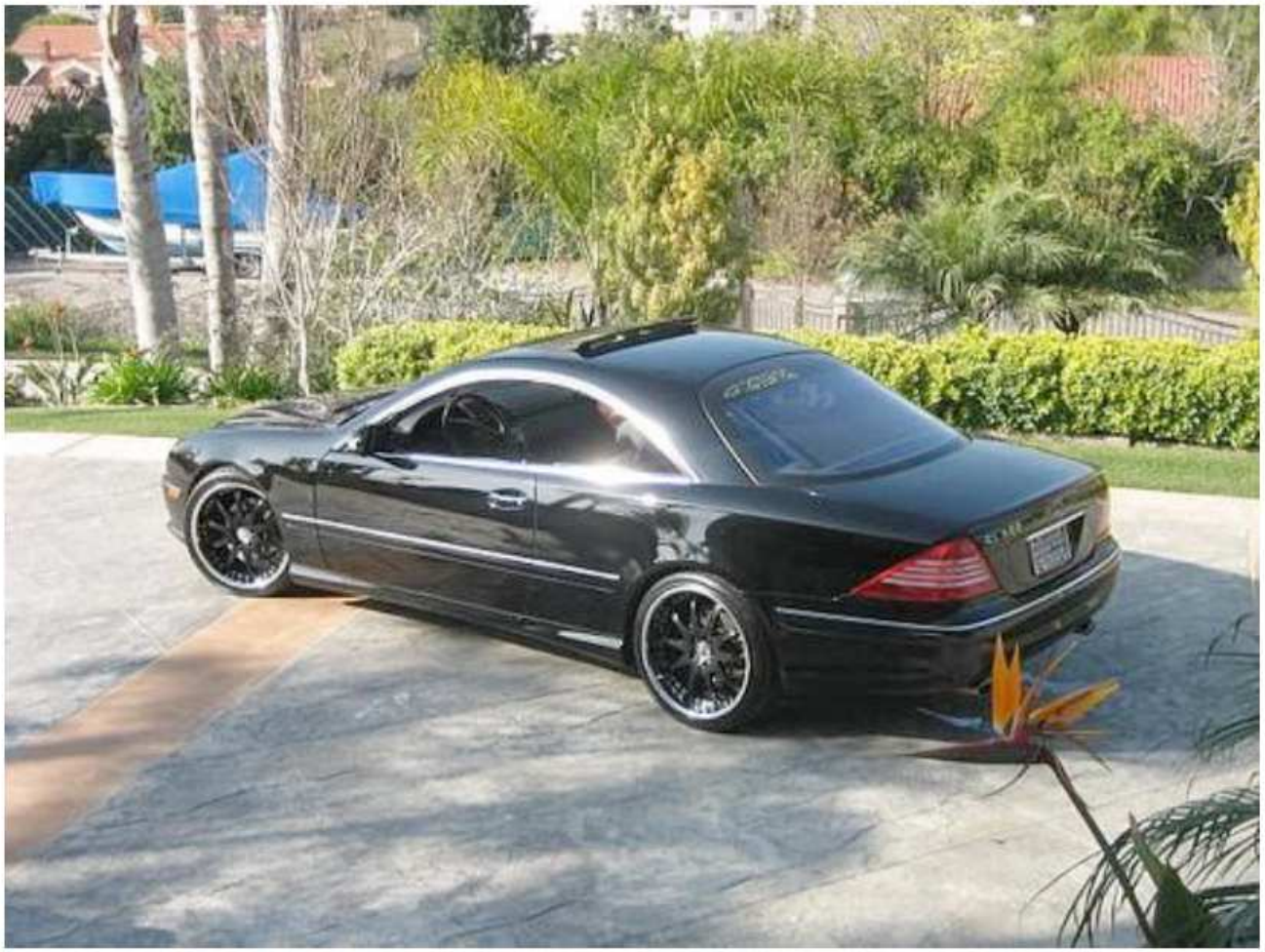
	DeCAF_6	DeCAF_7
LogReg	$\mathbf{40.94 \pm 0.3}$	40.84 ± 0.3
SVM	39.36 ± 0.3	40.66 ± 0.3
Xiao et al. (2010)	38.0	

Transfer Learning & Fine tune

- Retrain only classifiers
- Add one or two full connected layers
- Fine-tune more

Maxime Oquab, Léon Bottou, Ivan Laptev, Josef Sivic. Learning and transferring mid-level image representions using convolutional neural networks. CVPR14

Image classification (easy)



Is there a **car** ?

Source : Pascal VOC dataset

Image classification (harder)

Is there a **boat** ?

Source : Pascal VOC dataset

Image classification (harder)

Is there a **boat** ?

Source : Pascal VOC dataset

Image classification (v.hard)

Is there
a **person** ?

Source : Pascal VOC dataset

Image classification (v.hard)

Source : Pascal VOC dataset

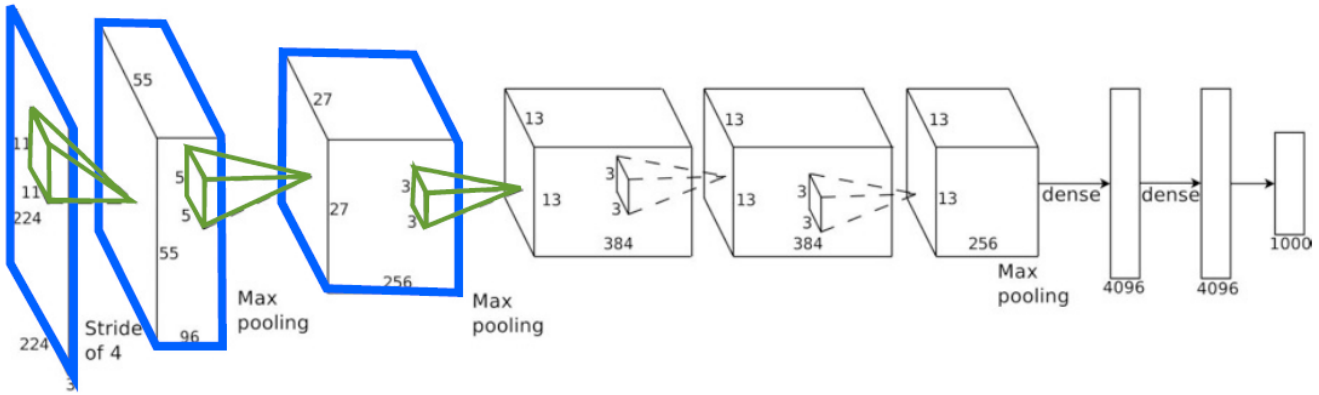
Pascal VOC vs. ImageNet classification

Pascal VOC :
complex scenes
20 object classes
10k images

ImageNet :
object-centric
1000 object classes
1.2M images

Slides from Maxime Oquab

8-layer NN
[Krizhevsky et al.]



60 million parameters :
- ImageNet (1.2M images) : OK
- Pascal VOC (10k images) : ?

Pascal VOC : different task

Car examples from
Pascal VOC

Typical car examples
from ImageNet

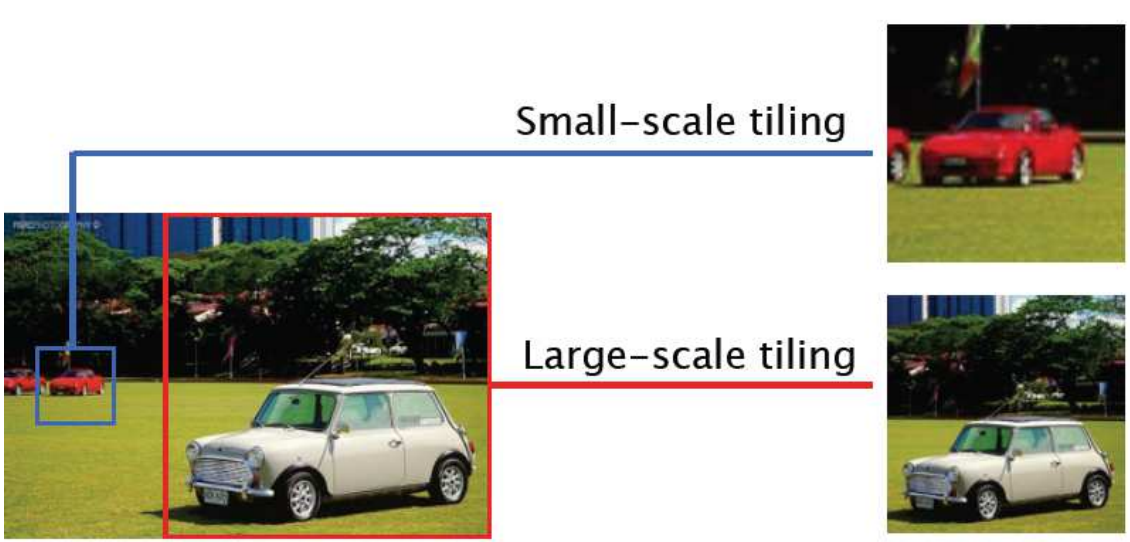
Pascal VOC : different task

Car examples from
Pascal VOC

Typical car examples
from ImageNet

Solution : multi-scale patch tiling

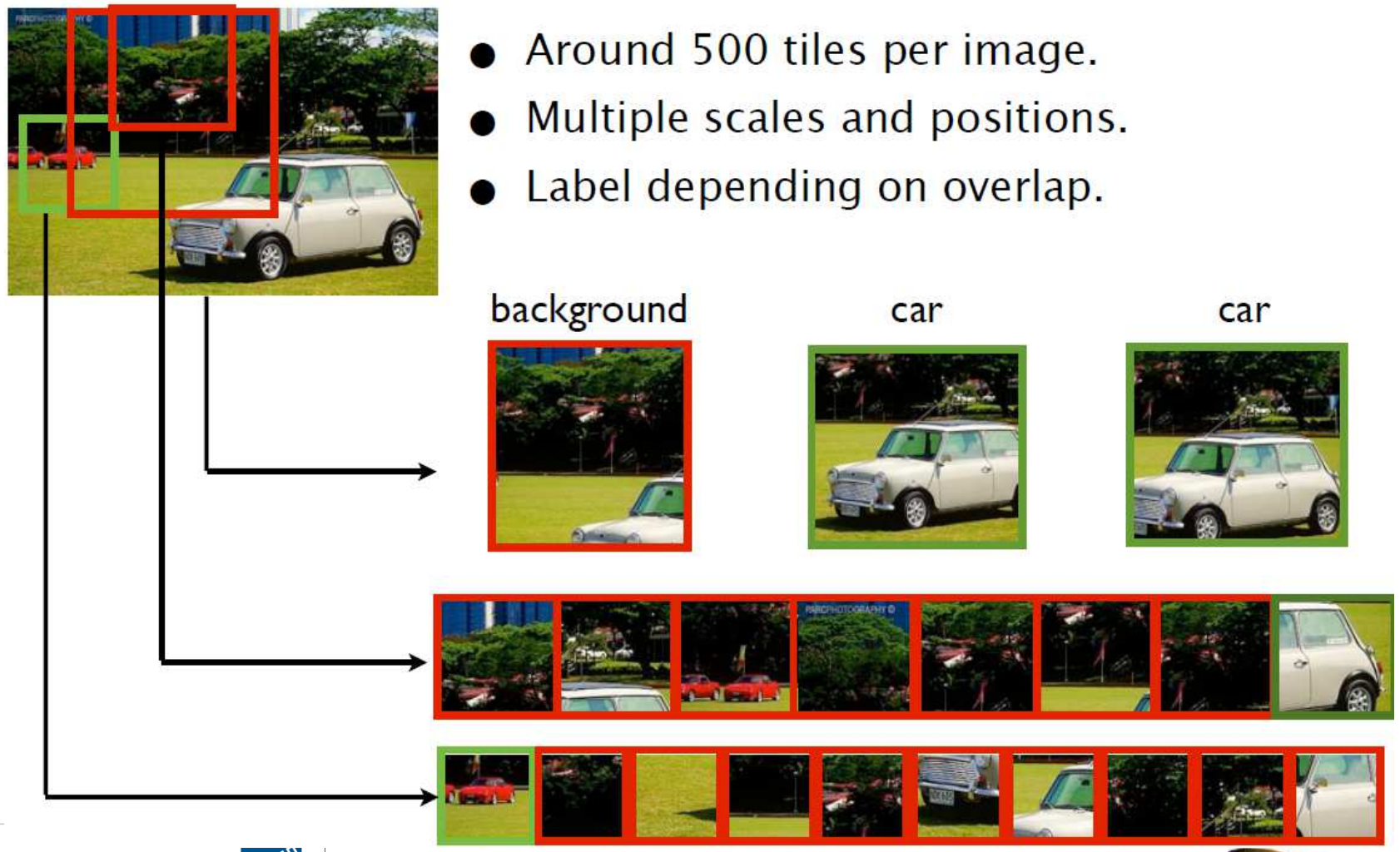
- Goal : obtain a **dataset that looks like ImageNet**



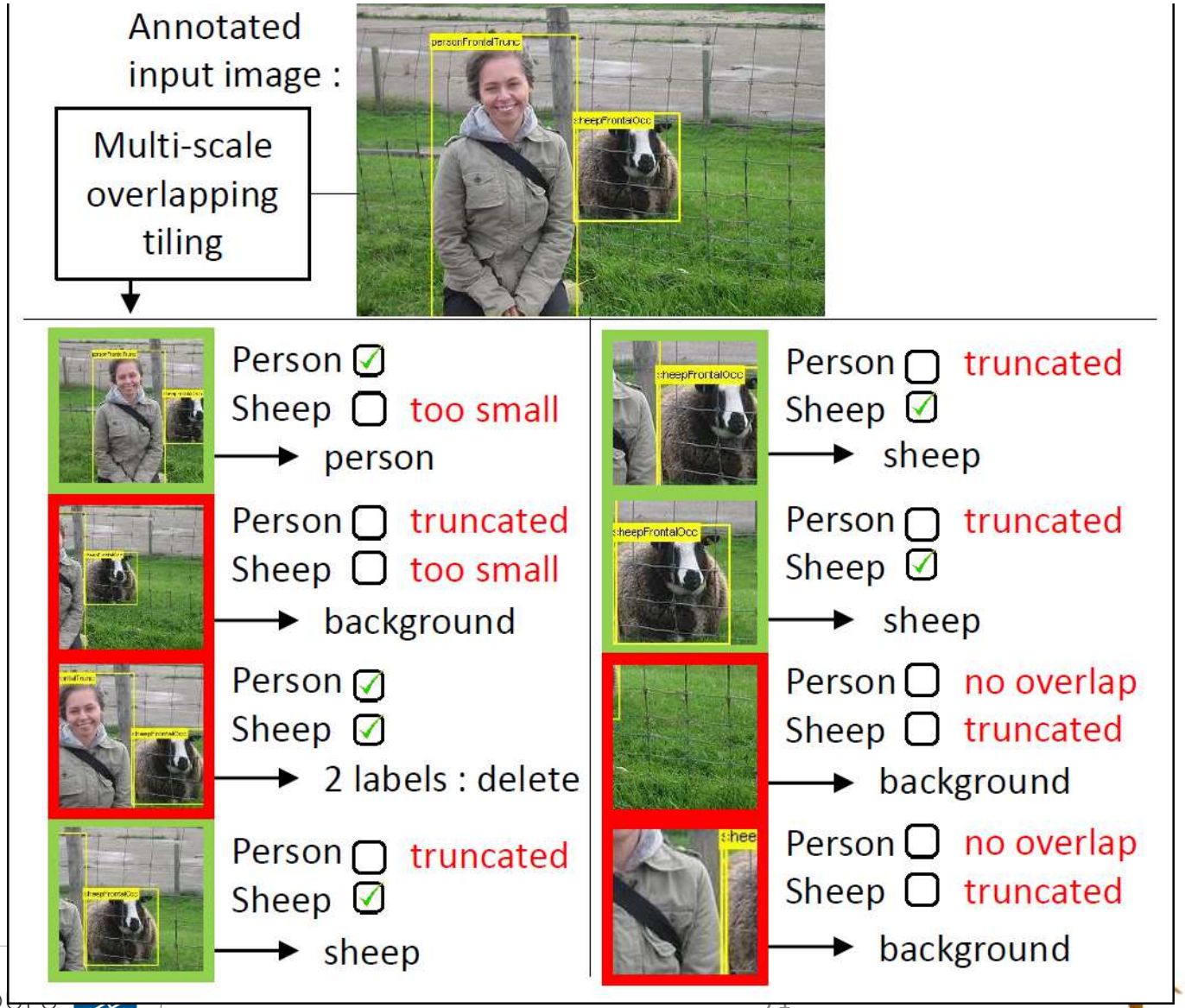
Typical Pascal VOC car example ... **... in disguise**

Typical car examples from ImageNet

Solution : multi-scale patch tiling



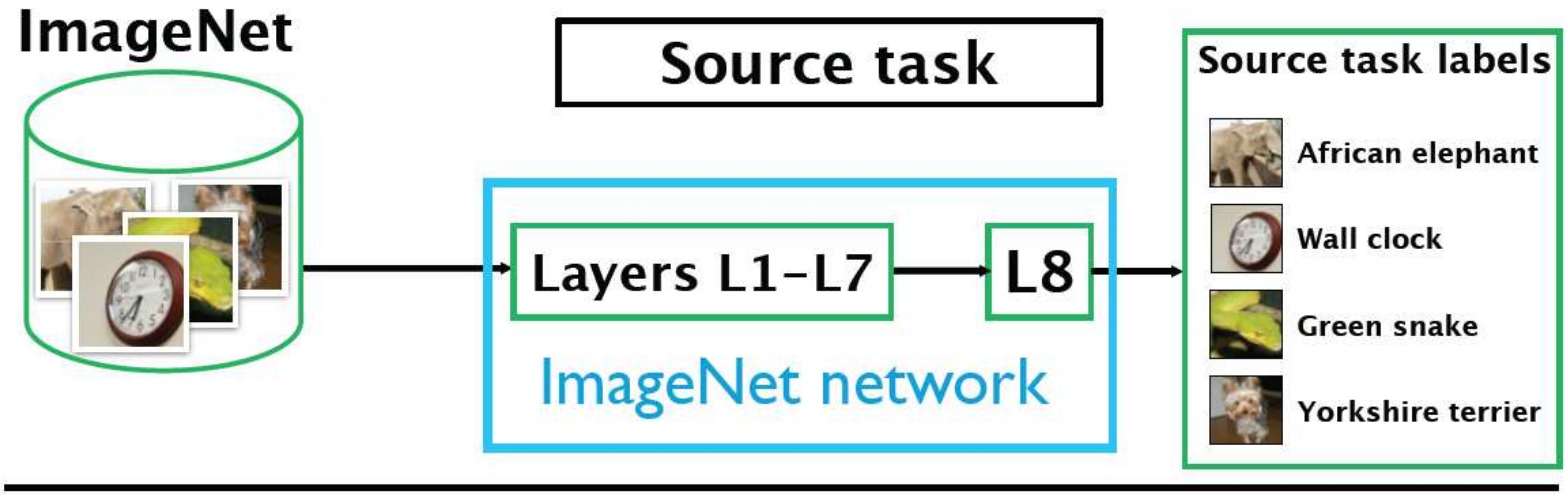
Generating training data for the target task



First attempt

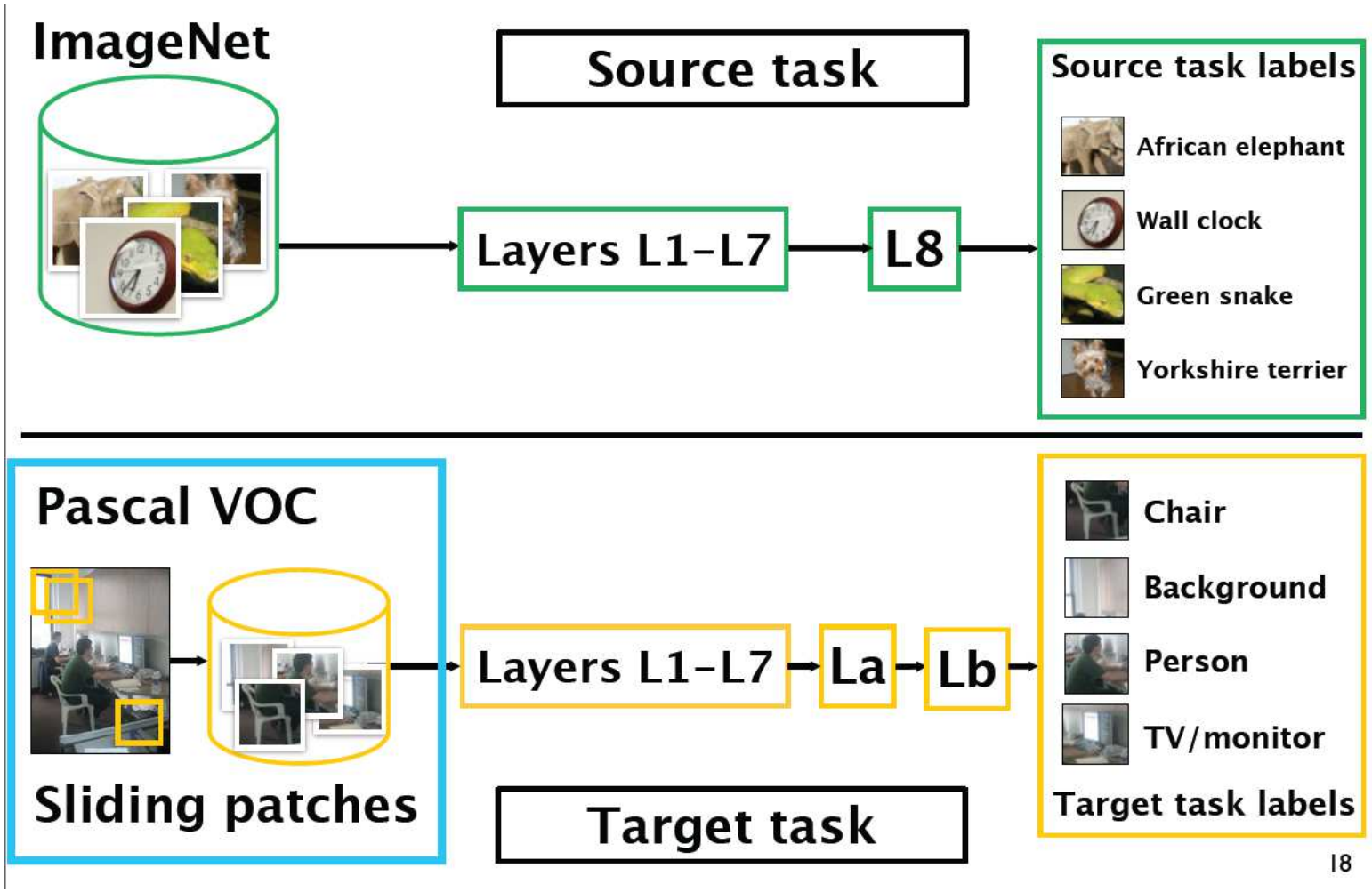
- Train CNN on Pascal VOC patches :
  - Result : 70.9% mAP.
  - We observe **overfitting**.
  - State of the art : 82.2% mAP (NUS-PSL).
- How to benefit from the power of neural networks ?
- We propose **transfer learning**.

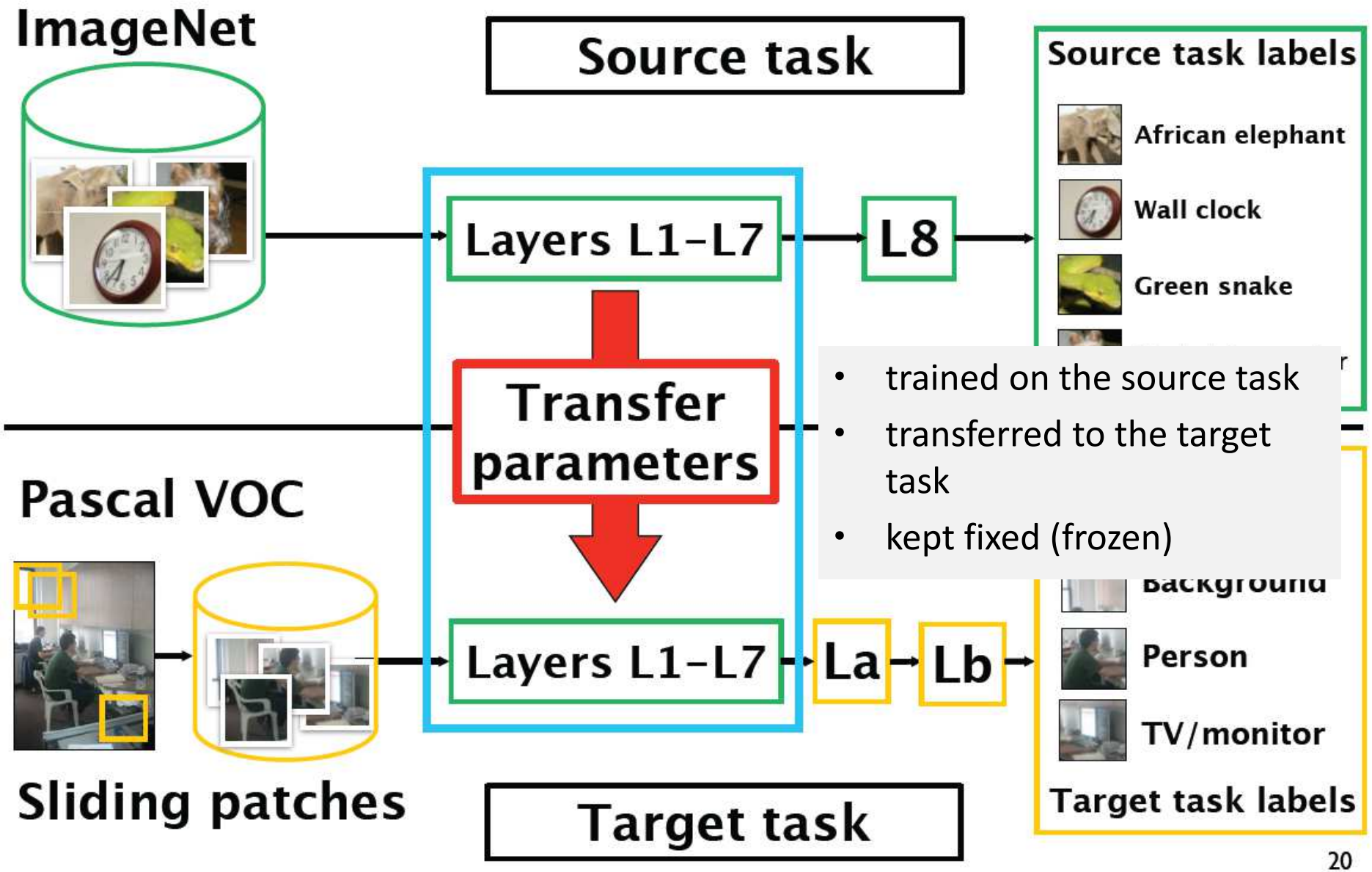
Transfer learning

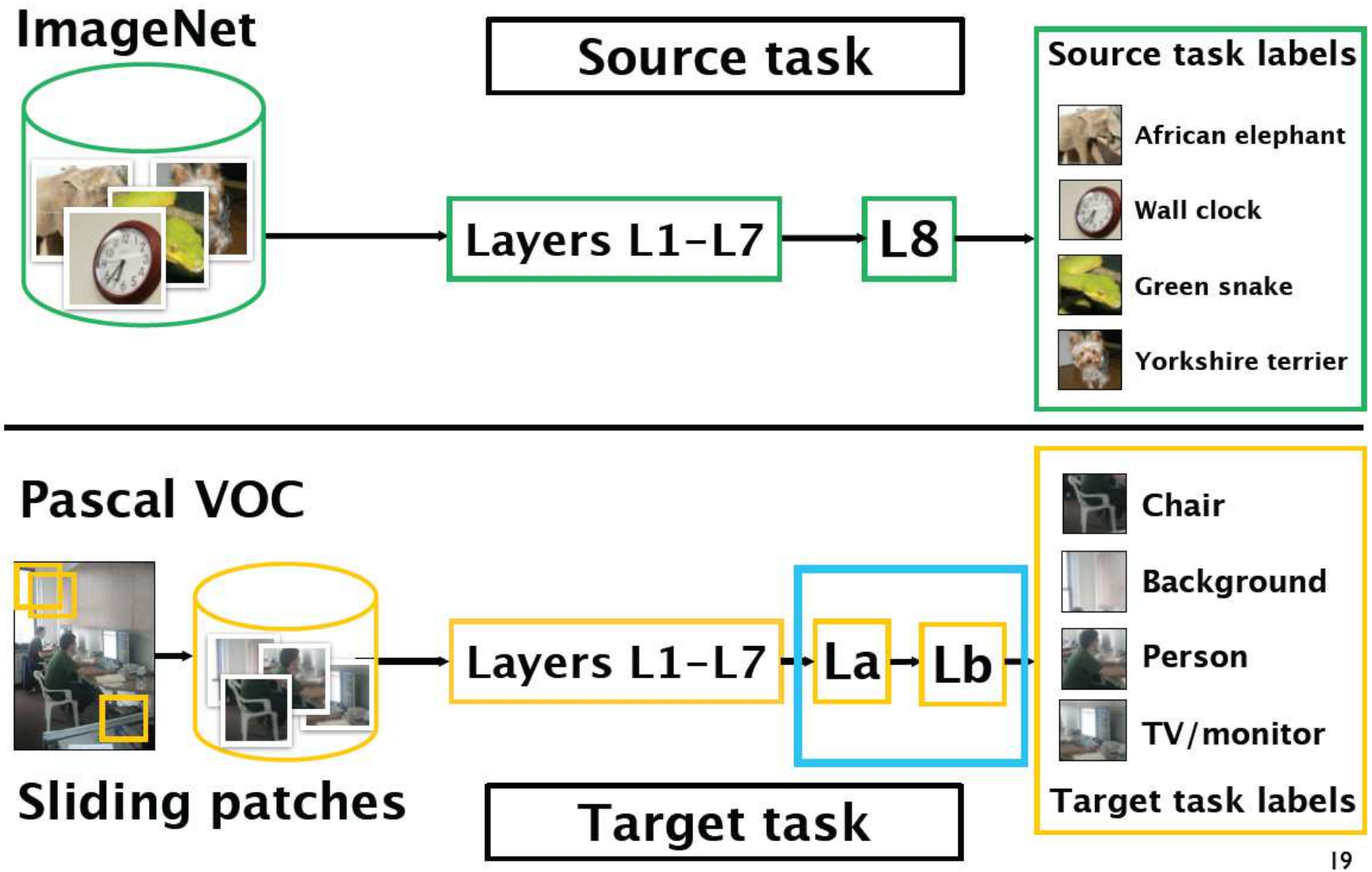


- Layers 6, 7, and 8 are full connected layers
- Layer 8 has the 'SoftMax' function as the non-linear activation functions.

Transfer learning




ImageNet
Source task
Source task labels
African elephant
Wall clock
Green snake
Layers L1–L7
L8
Transfer parameters
trained on the source task
transferred to the target task
kept fixed (frozen)
Pascal VOC
Layers L1–L7
La
Lb
Person
TV/monitor
Sliding patches
Target task
Target task labels
20


ImageNet
Source task
Source task labels
African elephant
Wall clock
Green snake
Yorkshire terrier
Layers L1-L7
L8
Pascal VOC
Chair
Background
Person
TV/monitor
Target task labels
Layers L1-L7
La
Lb
Sliding patches
Target task
19

directly trained on the Pascal VOC 2012 training data (NO PRETRAIN) without using any external data from ImageNet.

Second attempt (with pre-training)

- After pre-training on the ILSVRC-2012 dataset, we obtain 78.7% mean AP (no pre-train : 70.9%).
- Significantly better but can we improve more ?

	plane	bike	bird	boat	btl	bus	car	cat	chair	cow	table	dog	horse	moto	pers	plant	sheep	sofa	train	tv	mAP
NUS-PSL [48]	**97.3**	**84.2**	80.8	**85.3**	**60.8**	**89.9**	**86.8**	89.3	**75.4**	77.8	**75.1**	83.0	87.5	**90.1**	95.0	57.8	79.2	**73.4**	**94.5**	**80.7**	82.2
NO PRETRAIN	85.2	75.0	69.4	66.2	48.8	82.1	79.5	79.8	62.4	61.9	49.8	75.9	71.4	82.7	93.1	59.1	69.7	49.3	80.0	76.7	70.9
PRE-1000C	93.5	78.4	*87.7*	80.9	57.3	85.0	81.6	*89.4*	66.9	73.8	62.0	*89.5*	83.2	87.6	*95.8*	*61.4*	79.0	54.3	88.0	78.3	78.7

+18 % **+14 %**

- Observe large boosts for dog and bird classes.
- Well-represented groups in ILSVRC-2012.

59 species of birds and 120 breeds

Slides from Maxime Oquab

Pre-training data

- Inspect 22k classes of the ImageNet tree:
  - «furniture» subtree contains **chairs, dining tables, sofas**

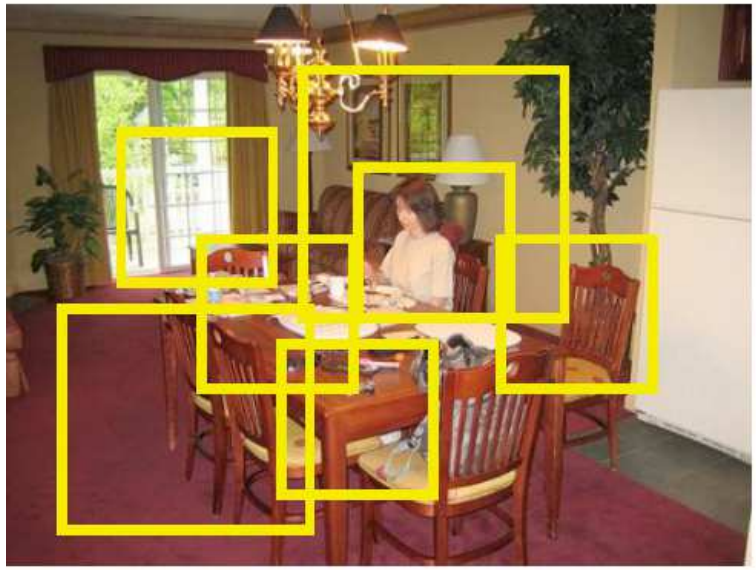
  - «hoofed mammal» subtree contains **sheep, horses, cows**

  - ...
- Add 512 classes to the pre-training,
- Result improves from 78.8% to **82.8%** mAP.
- All scores increase, targeted classes improve more.

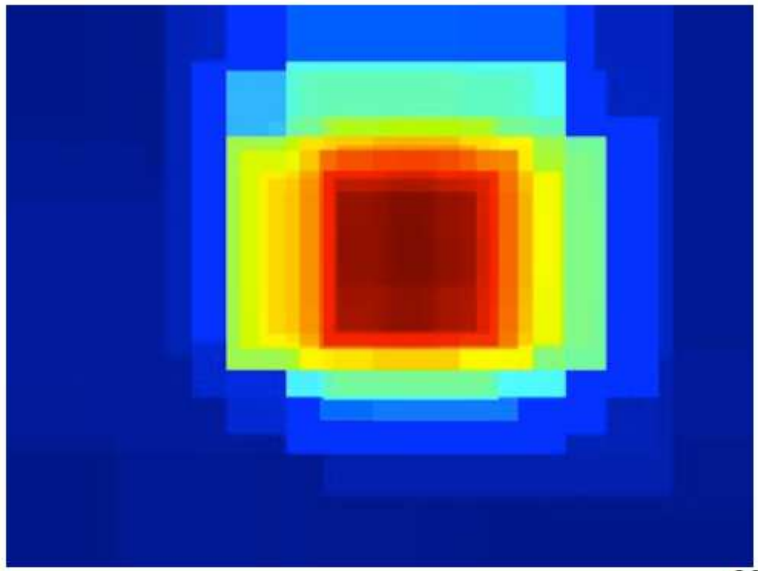
Slides from Maxime Oquab

Computing scores at test time

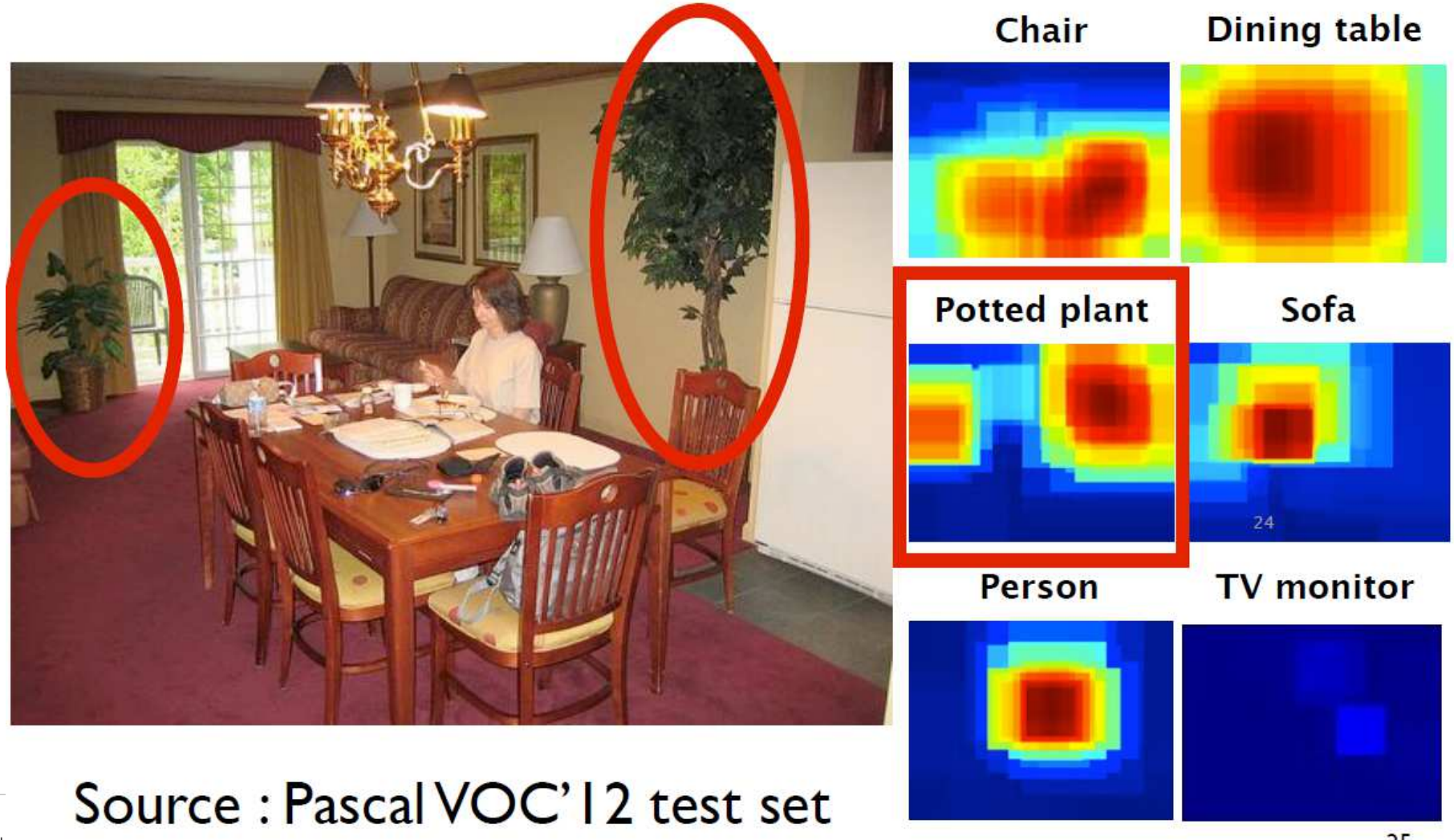
- We extract 500 multi-scale patches.
- **Image score = sum of all patch scores.**
- **Pixel score = sum of overlapping patches scores (heat maps**



CNN person classifier



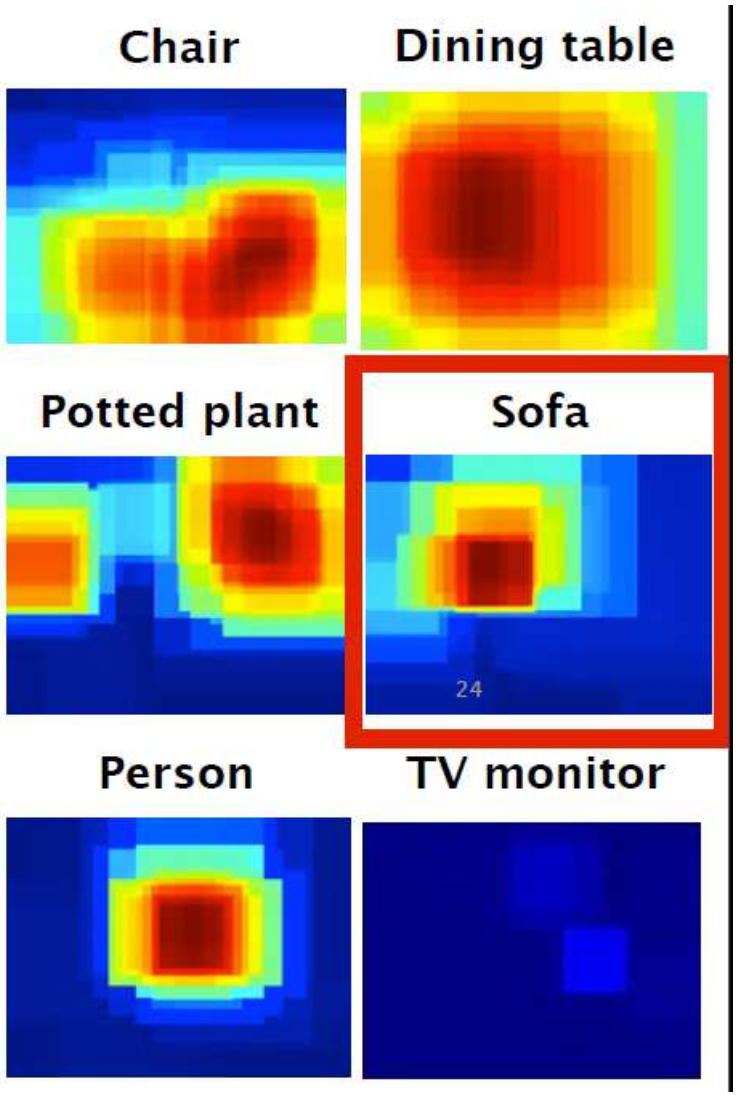
Qualitative results



Source : Pascal VOC'12 test set

Slides from Maxime Oquab

Qualitative results



Source : Pascal VOC'12 test set

Quantitative results

Pascal VOC'12 object classification :

	plane	bike	bird	boat	btl	bus	car	cat	chair	cow	table	dog	horse	moto	pers	plant	sheep	sofa	train	tv	mAP
NUS-PSL [48]	**97.3**	**84.2**	80.8	**85.3**	**60.8**	**89.9**	**86.8**	89.3	**75.4**	77.8	**75.1**	83.0	87.5	**90.1**	95.0	57.8	79.2	**73.4**	**94.5**	**80.7**	82.2
NO PRETRAIN	85.2	75.0	69.4	66.2	48.8	82.1	79.5	79.8	62.4	61.9	49.8	75.9	71.4	82.7	93.1	59.1	69.7	49.3	80.0	76.7	70.9
PRE-1000C	93.5	78.4	*87.7*	80.9	57.3	85.0	81.6	*89.4*	66.9	73.8	62.0	*89.5*	83.2	87.6	*95.8*	*61.4*	79.0	54.3	88.0	78.3	78.7
PRE-1000R	93.2	77.9	83.8	80.0	55.8	82.7	79.0	84.3	66.2	71.7	59.5	83.4	81.4	84.8	95.2	59.8	74.9	52.9	83.8	75.7	76.3
PRE-1512	94.6	82.9	**88.2**	84.1	60.3	89.0	84.4	**90.7**	72.1	**86.8**	69.0	**92.1**	**93.4**	88.6	**96.1**	**64.3**	**86.6**	62.3	91.1	79.8	**82.8**

State of the art : **82.2**

No pre-training baseline : **70.9**

1000 ILSVRC classes : **78.7**

1512 classes (our best) : **82.8**

Pascal VOC'12 object classification :

	plane	bike	bird	boat	btl	bus	car	cat	chair	cow	table	dog	horse	moto	pers	plant	sheep	sofa	train	tv	mAP
NUS-PSL [48]	**97.3**	**84.2**	80.8	**85.3**	**60.8**	**89.9**	**86.8**	89.3	**75.4**	77.8	**75.1**	83.0	87.5	**90.1**	95.0	57.8	79.2	**73.4**	**94.5**	**80.7**	82.2
NO PRETRAIN	85.2	75.0	69.4	66.2	48.8	82.1	79.5	79.8	62.4	61.9	49.8	75.9	71.4	82.7	93.1	59.1	69.7	49.3	80.0	76.7	70.9
PRE-1000C	93.5	78.4	*87.7*	80.9	57.3	85.0	81.6	*89.4*	66.9	73.8	62.0	*89.5*	83.2	87.6	*95.8*	*61.4*	79.0	54.3	88.0	78.3	78.7
PRE-1000R	93.2	77.9	83.8	80.0	55.8	82.7	79.0	84.3	66.2	71.7	59.5	83.4	81.4	84.8	95.2	59.8	74.9	52.9	83.8	75.7	76.3
PRE-1512	94.6	82.9	**88.2**	84.1	60.3	89.0	84.4	**90.7**	72.1	**86.8**	69.0	**92.1**	**93.4**	88.6	**96.1**	**64.3**	**86.6**	62.3	91.1	79.8	**82.8**

State of the art : 82.2

No pre-training baseline : 70.9

1000 ILSVRC classes : 78.7

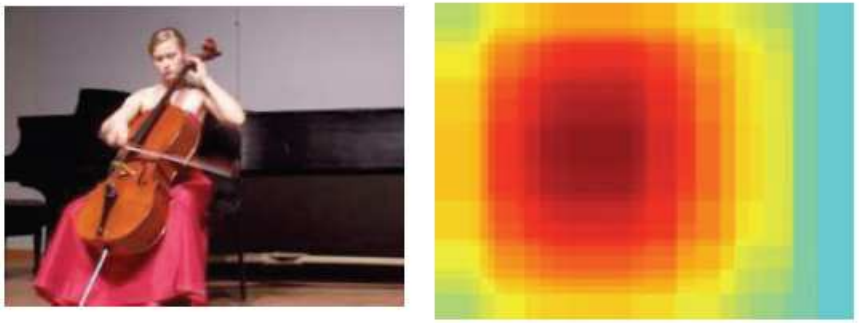
Random 1000 classes : 76.3

1512 classes (our best) : 82.8

1,000 ImageNet classes selected, this time, at random among all the 22,000 available ImageNet classes

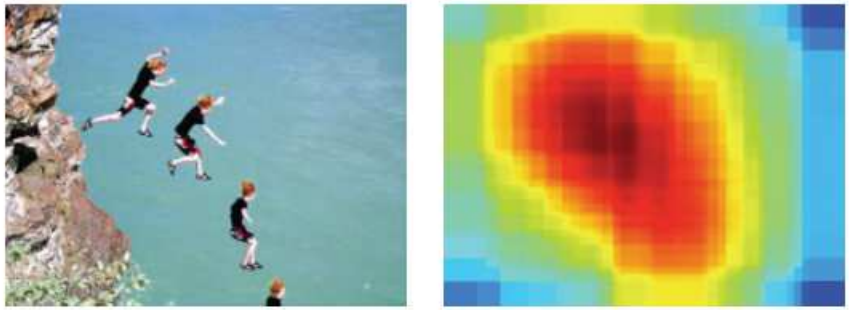
Different task : action classification (still images)

playing instrument

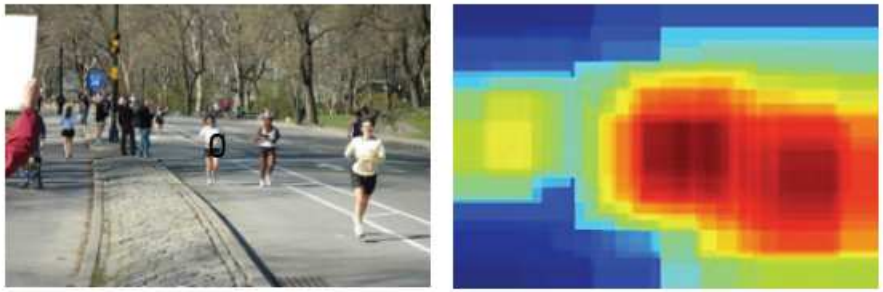


playing instrument

jumping



running



Source : Pascal VOC'12 Action classification test set

State-of-the-art 70.2% mAP result

38

- **Transfer learning with CNNs avoids overfitting**
- • See also : [Girshick et al.'14], [Sermanet et al.'13], [Donahue et al. '13],
- [Zeiler & Fergus '13], [Razavian et al. '14], [Chatfield et al. '14]
- **•We study the effect of pre-training data :**
- • More pre-training data => better
- • Related pre-training data => even better
- **•Transfer to action classification.**
- **•http://www.di.ens.fr/willow/research/cnn/**
- • Implementation (Torch7 modules) available soon
- • Includes efficient and flexible GPU training code

Transfer Learning & Fine tune

- Retrain only classifiers
- Add one or two full connected layers
- Fine-tune more

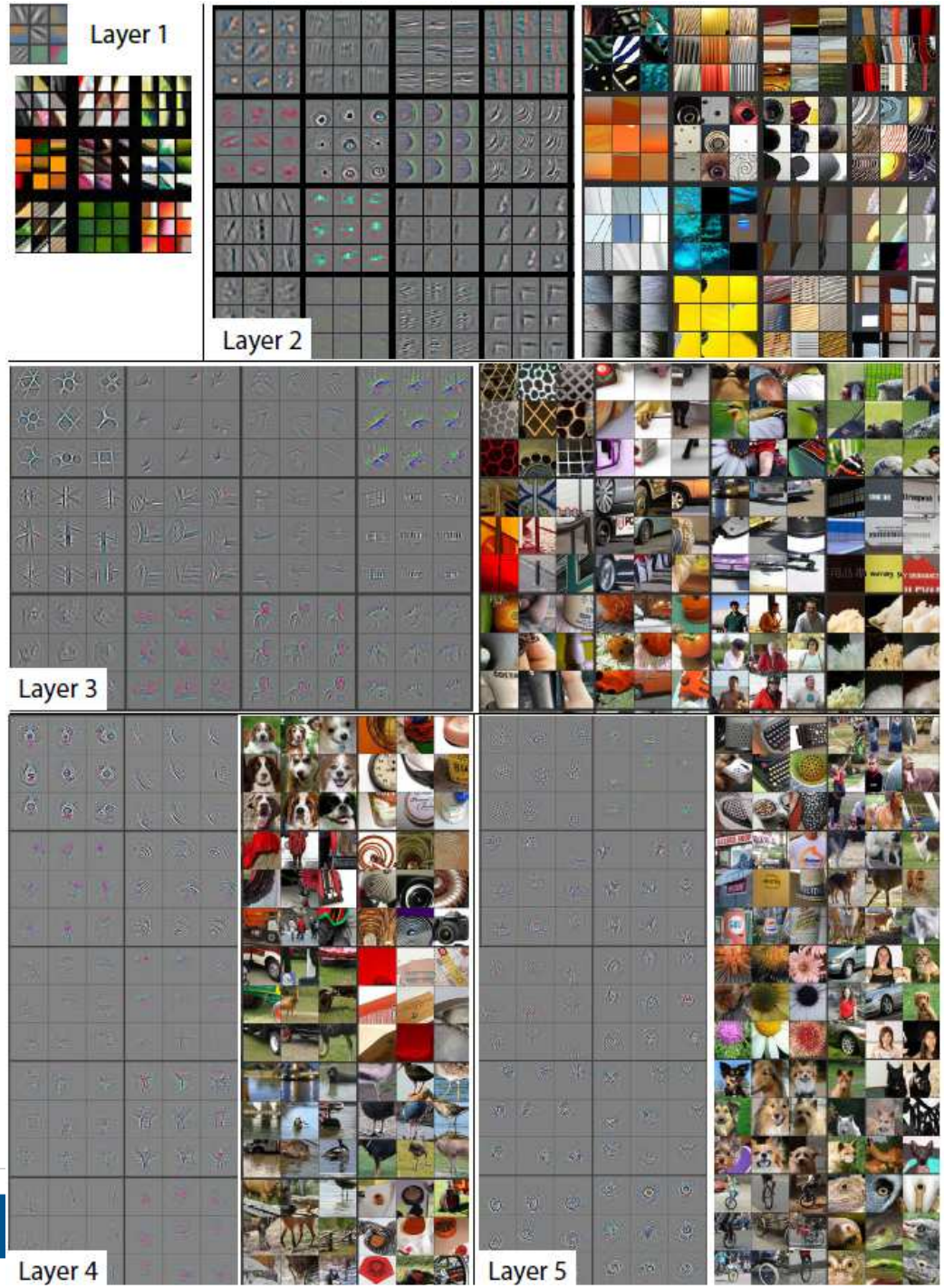
How transferable are features in deep neural networks?

Jason Yosinski,[1] Jeff Clune,[2] Yoshua Bengio,[3] and Hod Lipson[4]

Visualizing and Understanding Convolutional Networks
[Zeiler and Fergus, 2013]

	Cal-101 (30/class)	Cal-256 (60/class)
SVM (1)	44.8 ± 0.7	24.6 ± 0.4
SVM (2)	66.2 ± 0.5	39.6 ± 0.3
SVM (3)	72.3 ± 0.4	46.0 ± 0.3
SVM (4)	76.6 ± 0.4	51.3 ± 0.1
SVM (5)	$\mathbf{86.2 \pm 0.8}$	65.6 ± 0.3
SVM (7)	$\mathbf{85.5 \pm 0.4}$	$\mathbf{71.7 \pm 0.2}$
Softmax (5)	82.9 ± 0.4	65.7 ± 0.5
Softmax (7)	$\mathbf{85.4 \pm 0.4}$	$\mathbf{72.6 \pm 0.1}$

Q: If we were to use only one layer, which layer should we transfer from?


Layer 1
Layer 2
Layer 3
Layer 4
Layer 5

Intuitive Interpretations of each layers

- Layer 1: Edge
- Layer 2: Corner/edge color conjunctions
- Layer 3: More complex invariances, capturing similar textures
- Layer 4: Significant variation, more class-specific
- Layer 5: Entire objects with significant pose variation

One common phenomenon

- Many deep neural networks trained on natural images exhibit a curious phenomenon in common:
  - on the first layers they learn features similar to Gabor filters and color blobs.
  - Such first-layer features appear not to be specific to a particular dataset or task, but general in that they are applicable to many datasets and tasks.
  - It shows the generality of the features from the first layer(s).

How transferable are features in deep neural networks?

Jason Yosinski,[1] Jeff Clune,[2] Yoshua Bengio,[3] and Hod Lipson[4]

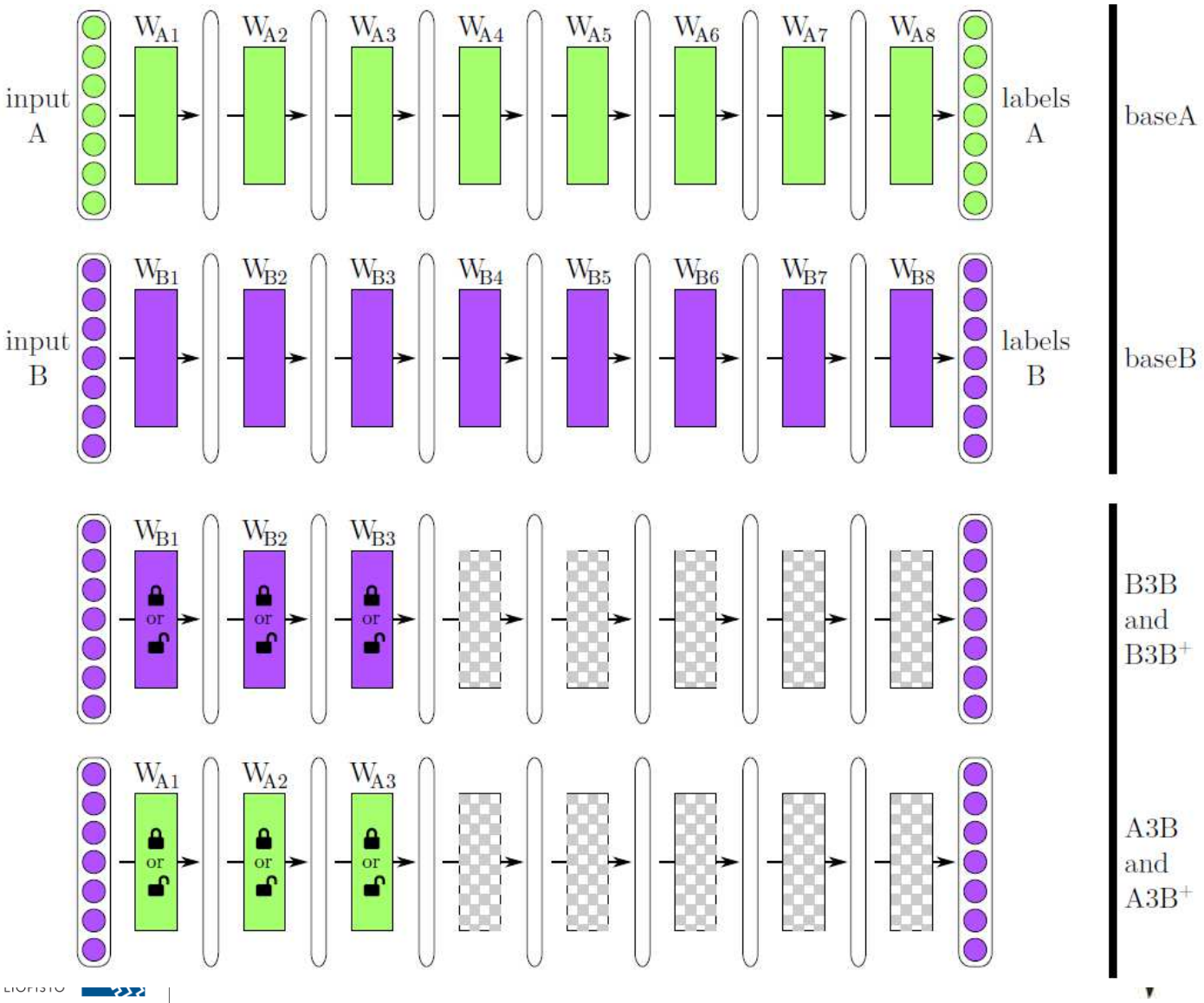
- In this paper we experimentally quantify the generality versus specificity of neurons in each layer of a deep convolutional neural network and report a few surprising results.
- Transferability is negatively affected by two distinct issues:
  - (1) the specialization of higher layer neurons to their original task at the expense of performance on the target task, which was expected
  - (2) optimization difficulties related to splitting networks between co-adapted neurons, which was not expected.

Intuitive notions

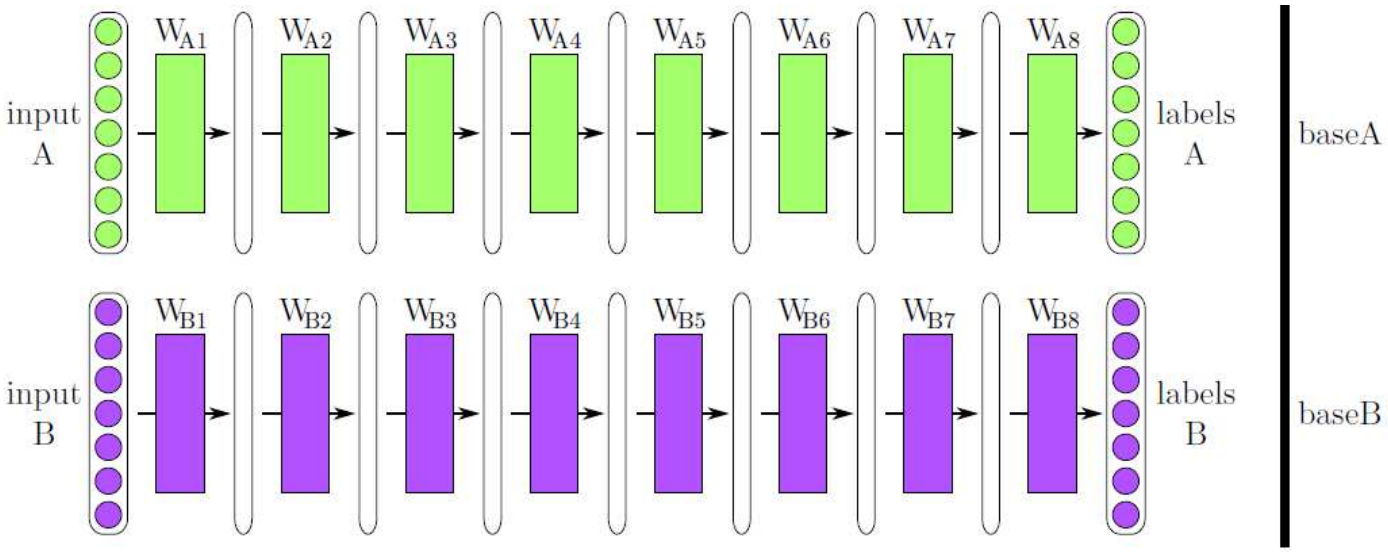
- first-layer features → general
  - Because finding these standard features on the first layer seems to occur regardless of the exact cost function and natural image dataset, we call these first-layer features general
- the last-layer features → specific
  - a network with an N-dimensional softmax output layer that has been successfully trained toward a supervised classification objective
  - each output unit will be specific to a particular class

- If first-layer features are general and last-layer features are specific, there must be a transition from general to specific somewhere in the network.
- This observation raises a few questions:
  - Can we quantify the degree to which a particular layer is general or specific?
  - Does the transition occur suddenly at a single layer, or is it spread out over several layers?
  - Where does this transition take place: near the first, middle, or last layer of the network?

Experimental treatments and controls



Experimental treatments and controls



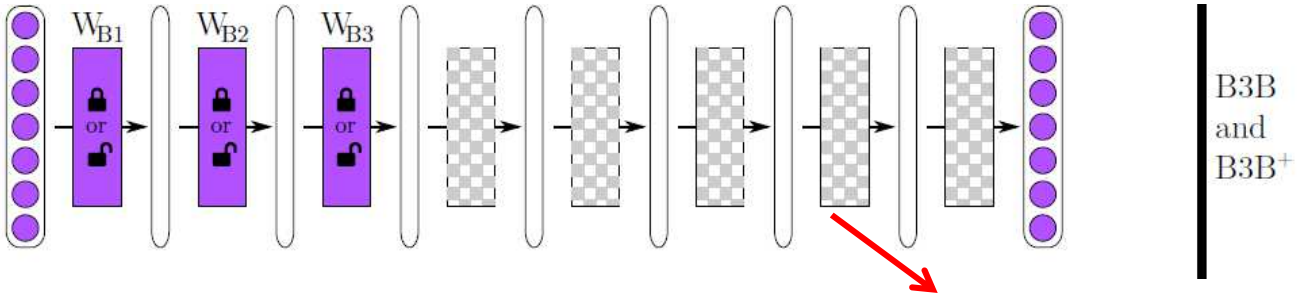
Top two rows: The base networks are trained using standard supervised backprop on only half of the ImageNet dataset (first row: A half, second row: B half).

The labeled rectangles (e.g. WA1) represent the weight vector learned for that layer, with the color indicating which dataset the layer was originally trained on.

The vertical ellipsoidal bars between weight vectors represent the activations of the network at each layer.

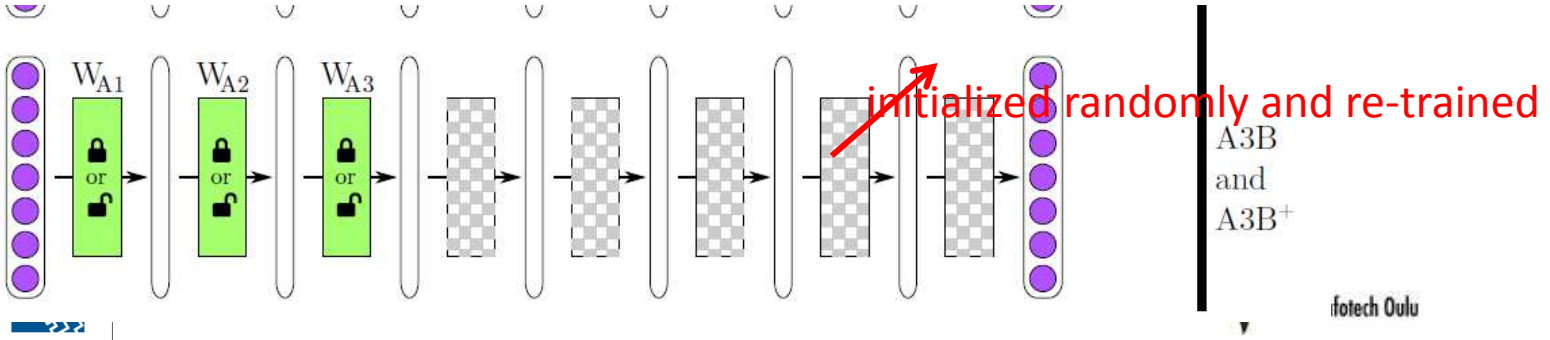
Experimental treatments and controls

In the selffer network control, the first n weight layers of the network (in this example, n = 3) are copied from a base network (e.g. one trained on dataset B)
the upper 8 − n layers are randomly initialized, and then the entire network is trained on that same dataset (in this example, dataset B).
The first n layers are either locked during training (“frozen” selffer treatment B3B) or allowed to learn (“fine-tuned” selffer treatment B3B+).
This treatment reveals the occurrence of fragile coadaptation,
when neurons on neighboring layers co-adapt during training in such a way that cannot be rediscovered when one layer is frozen.



Experimental treatments and controls

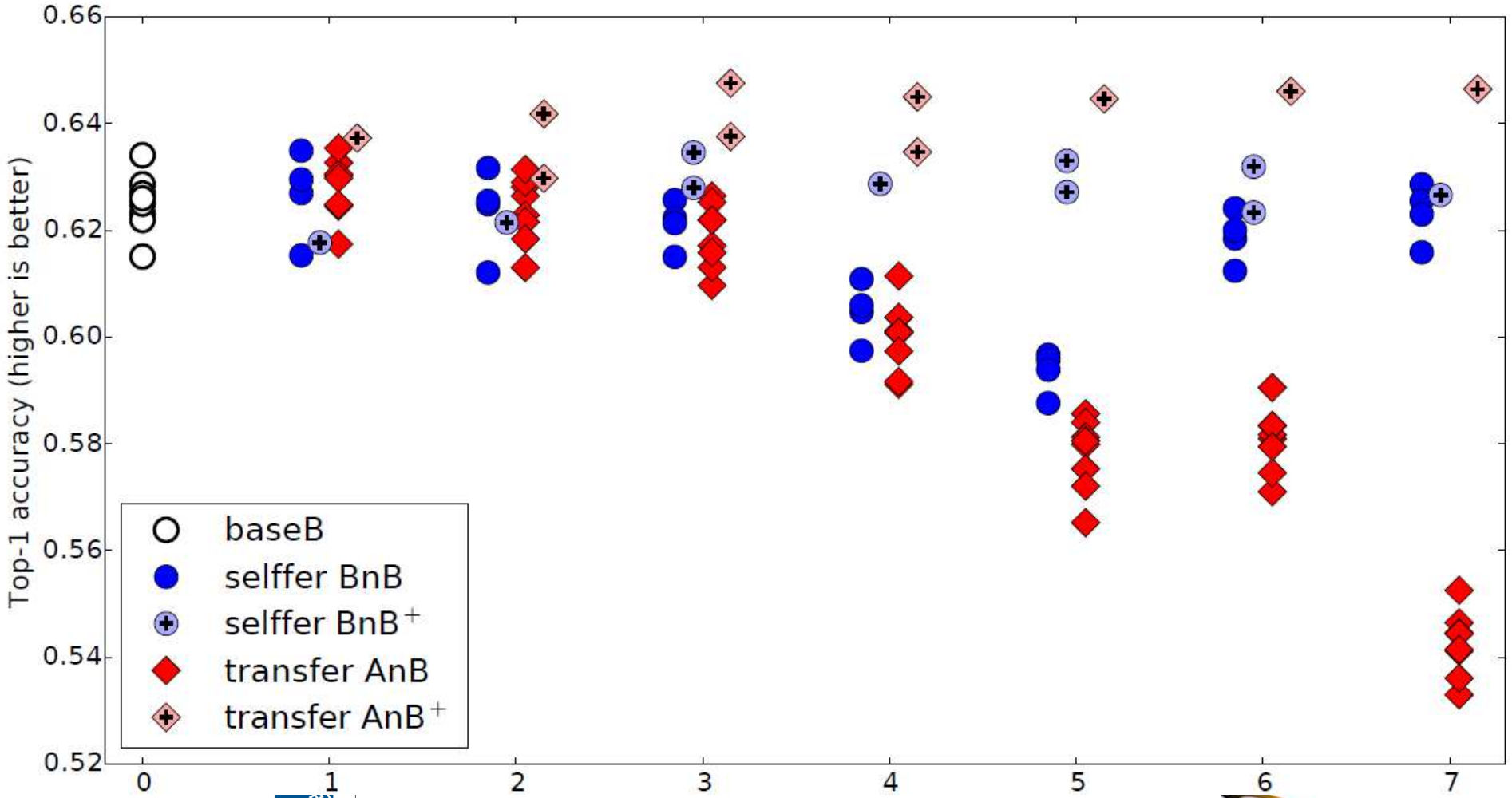
The transfer network experimental treatment is the same as the selffer treatment
except that the first n layers are copied from a network trained on one dataset (e.g. A) and then the entire network is trained on the other dataset (e.g. B).
This treatment tests the extent to which the features on layer n are general or specific.



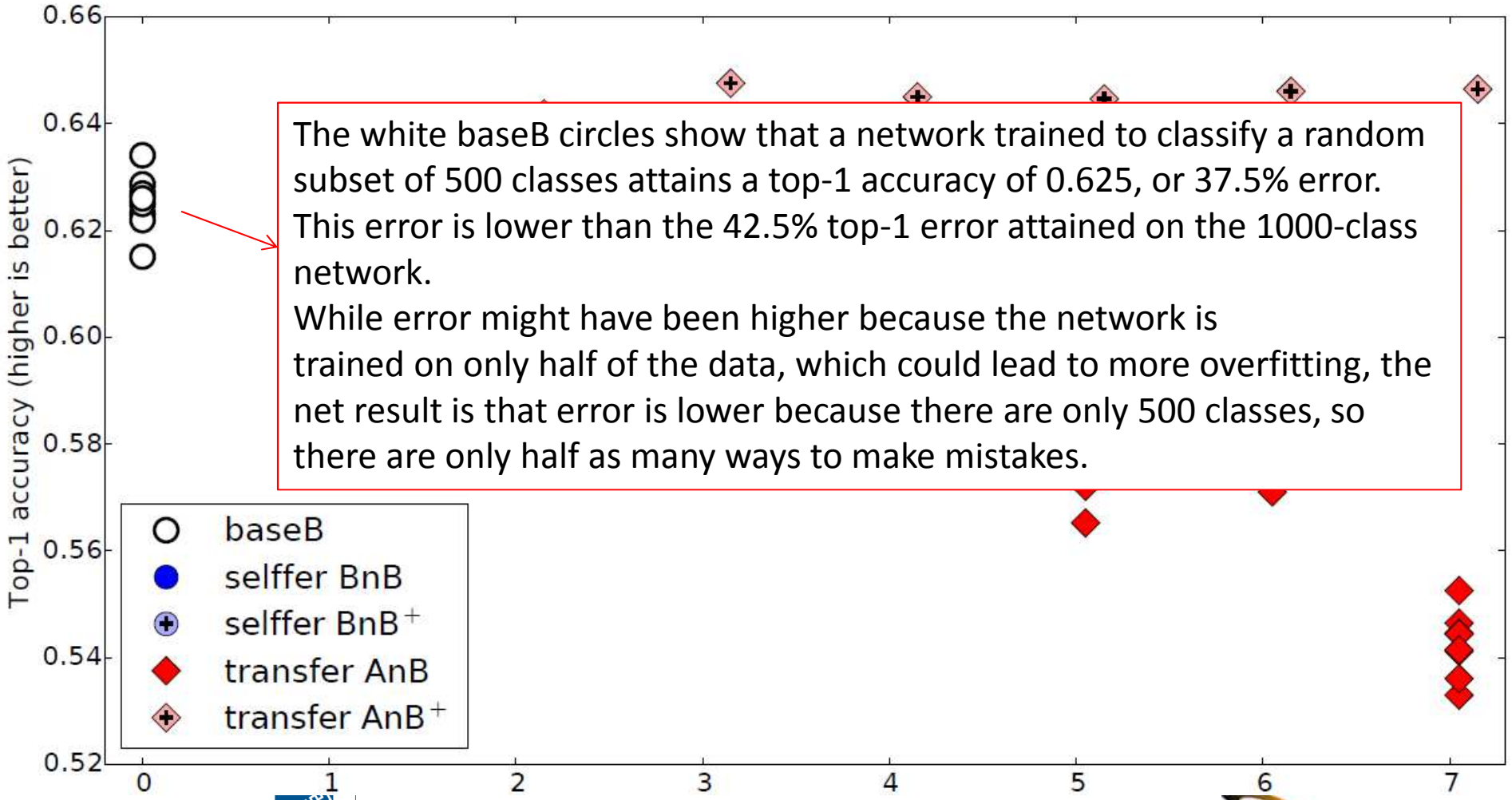
ImageNet Split

- To create base and target datasets that are similar to each other, we randomly assign half of the 1000 ImageNet classes to A and half to B. (Similar)
  - 500 classes each
- ImageNet allowed us to create a special split of the dataset into two halves that are as semantically different from each other as possible:
  - with dataset A containing only man-made entities
  - B containing natural entities.
  - 551 classes in the man-made group
  - 449 in the natural group.

Main results



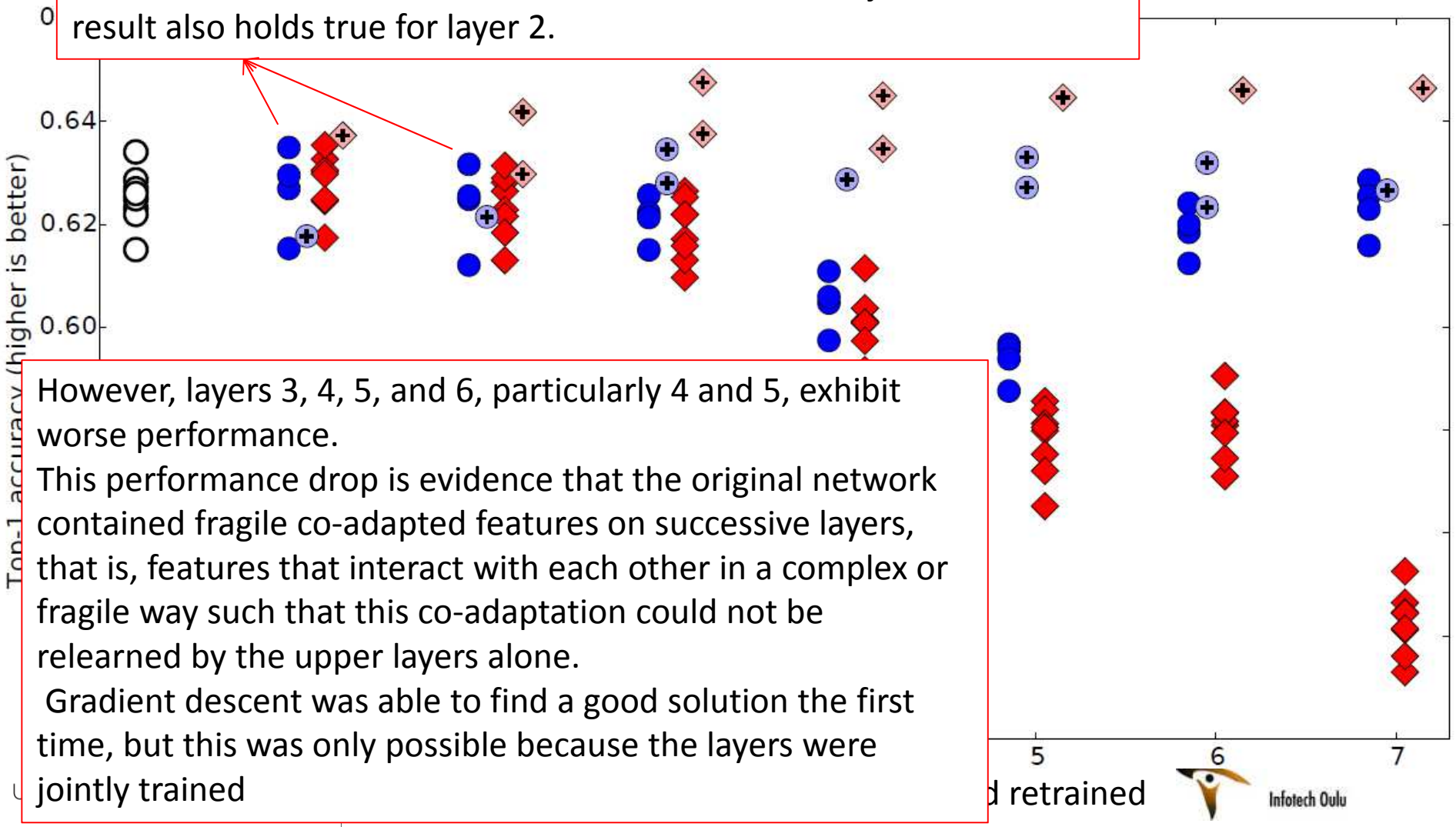
Main results



The white baseB circles show that a network trained to classify a random subset of 500 classes attains a top-1 accuracy of 0.625, or 37.5% error. This error is lower than the 42.5% top-1 error attained on the 1000-class network.

While error might have been higher because the network is trained on only half of the data, which could lead to more overfitting, the net result is that error is lower because there are only 500 classes, so there are only half as many ways to make mistakes.

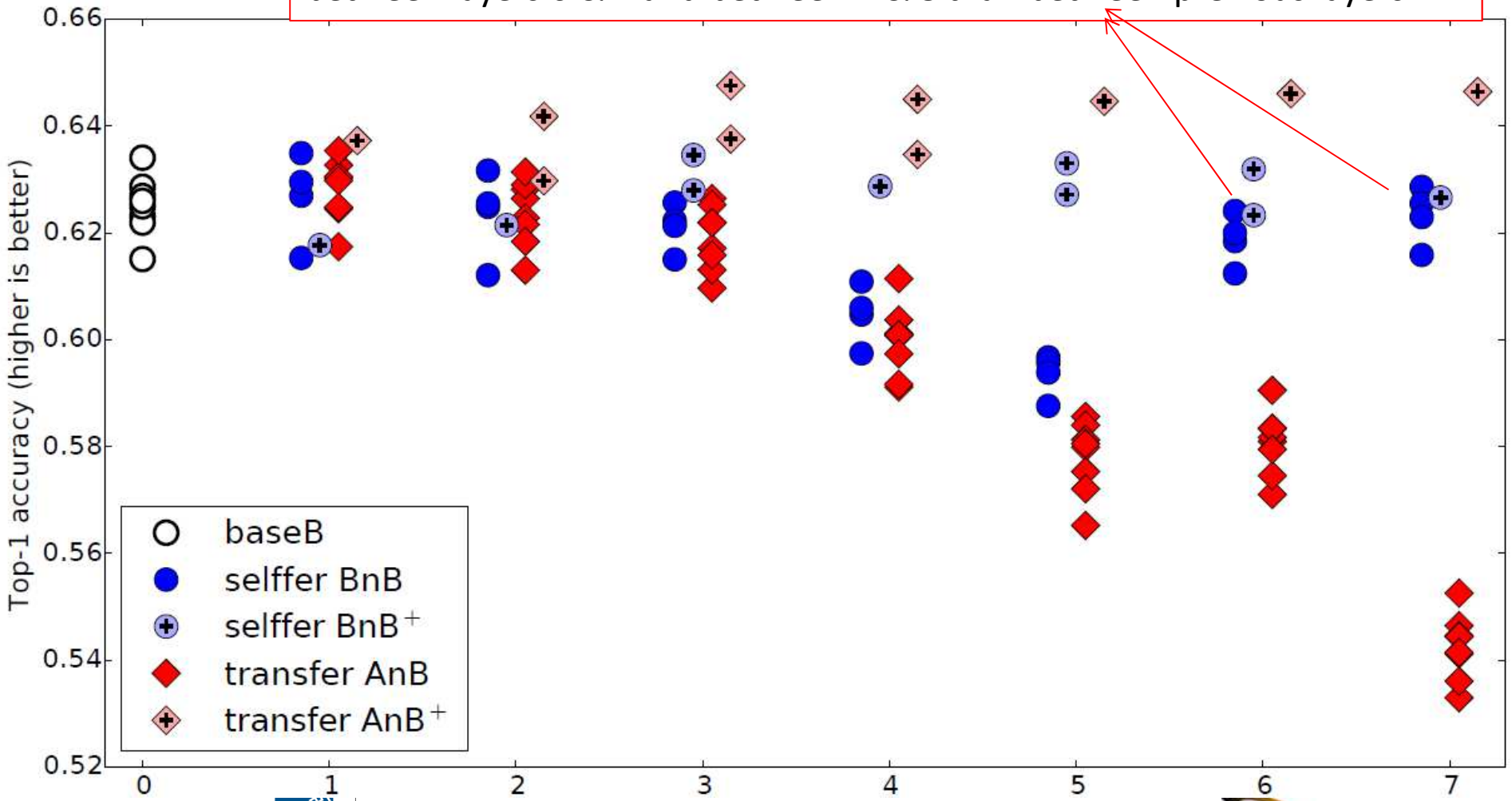
As expected, performance at layer one is the same as the baseB points. That is, if we learn eight layers of features, save the first layer of
learned Gabor features and color blobs, reinitialize the whole network, and retrain it toward the same task, it does just as well. This result also holds true for layer 2.



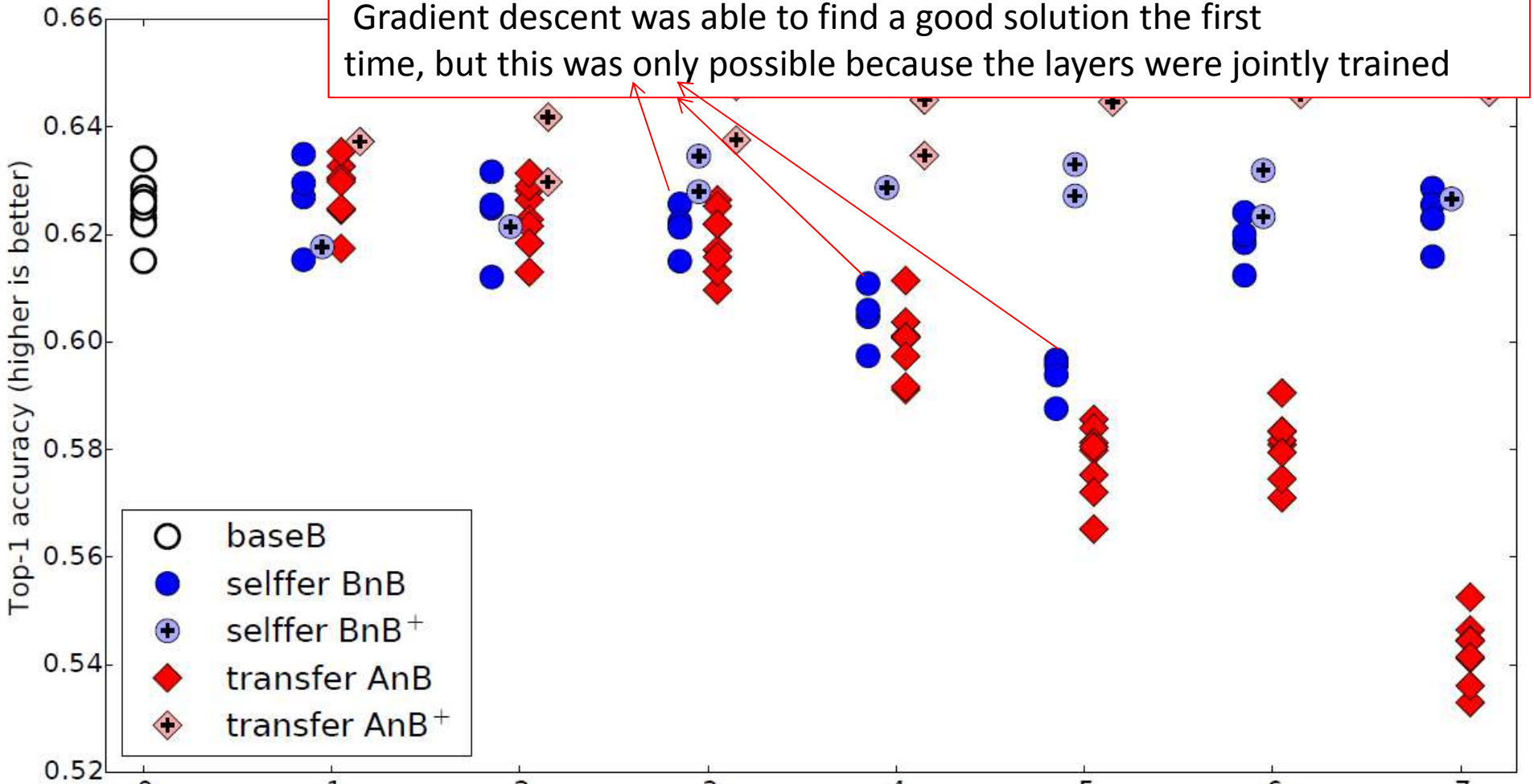
However, layers 3, 4, 5, and 6, particularly 4 and 5, exhibit worse performance.
This performance drop is evidence that the original network contained fragile co-adapted features on successive layers, that is, features that interact with each other in a complex or fragile way such that this co-adaptation could not be relearned by the upper layers alone.
Gradient descent was able to find a good solution the first time, but this was only possible because the layers were jointly trained

Infotech Oulu

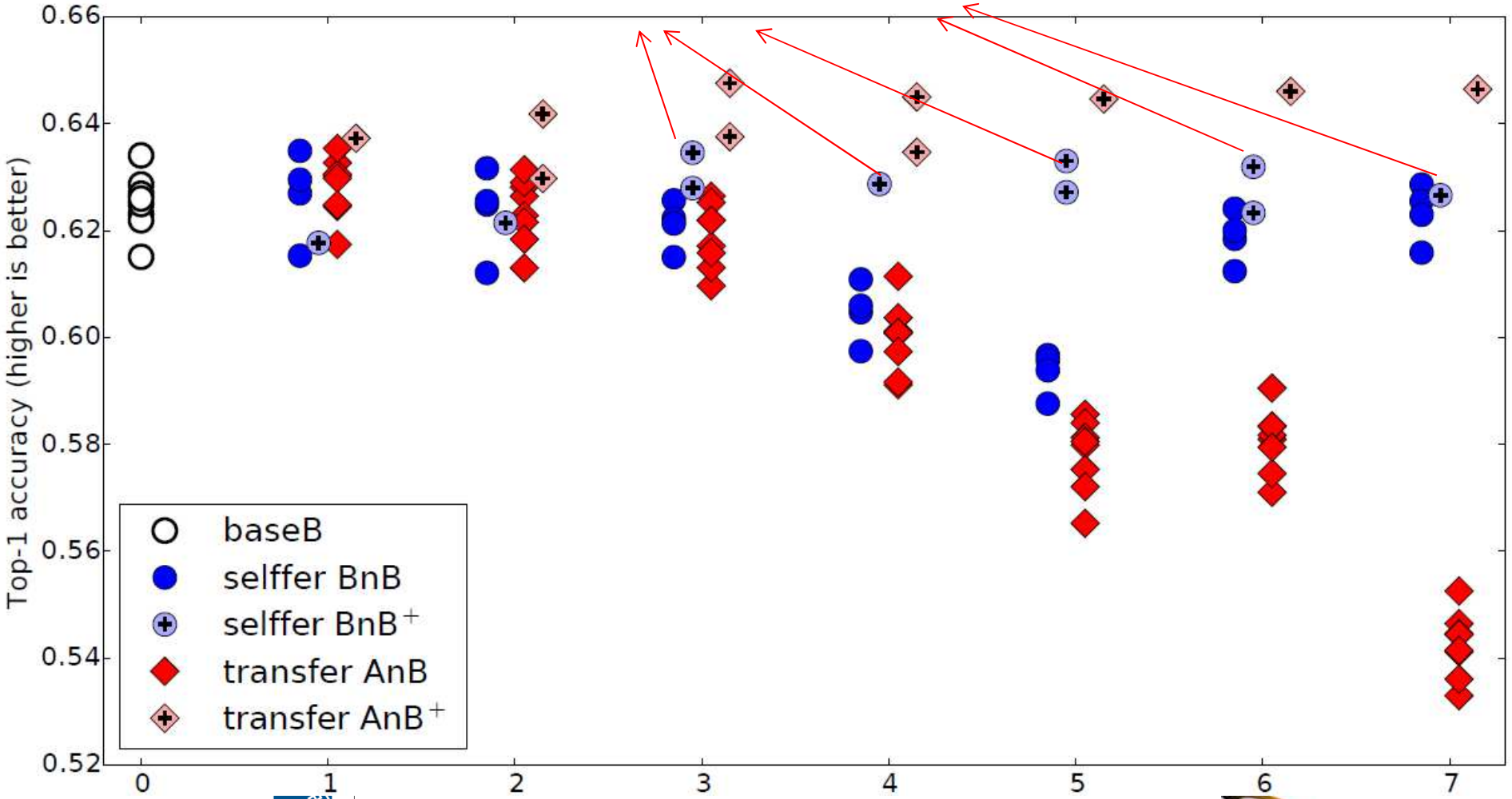
By layer 6 performance is nearly back to the base level, as is layer 7.
As we get closer and closer to the final, 500-way softmax output layer 8, there is less to relearn, and apparently relearning these one or two layers is simple enough for gradient descent to find a good solution.
Alternately, we may say that there is less co-adaptation of features between layers 6 & 7 and between 7 & 8 than between previous layers.



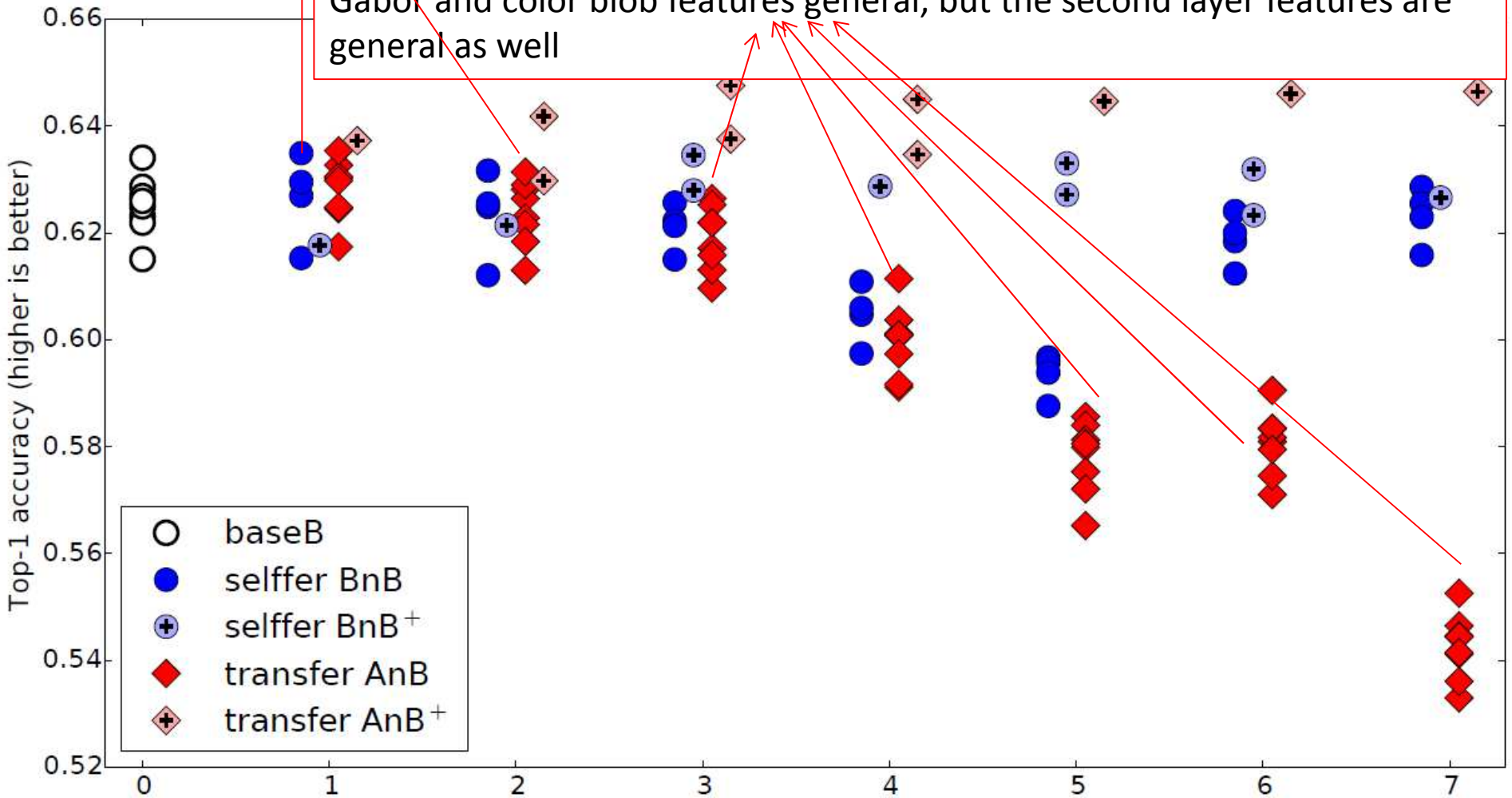
However, layers 3, 4, 5, and 6, particularly 4 and 5, exhibit worse performance.
This performance drop is evidence that the original network contained fragile co-adapted features on successive layers, that is, features that interact with each other in a complex or fragile way such that this co-adaptation could not be relearned by the upper layers alone.
Gradient descent was able to find a good solution the first time, but this was only possible because the layers were jointly trained



The light blue BnB+ points show that when the copied, lower-layer features also learn on the target dataset (which here is the same as the base dataset), performance is similar to the base case.
Such fine-tuning thus prevents the performance drop observed in the BnB networks.

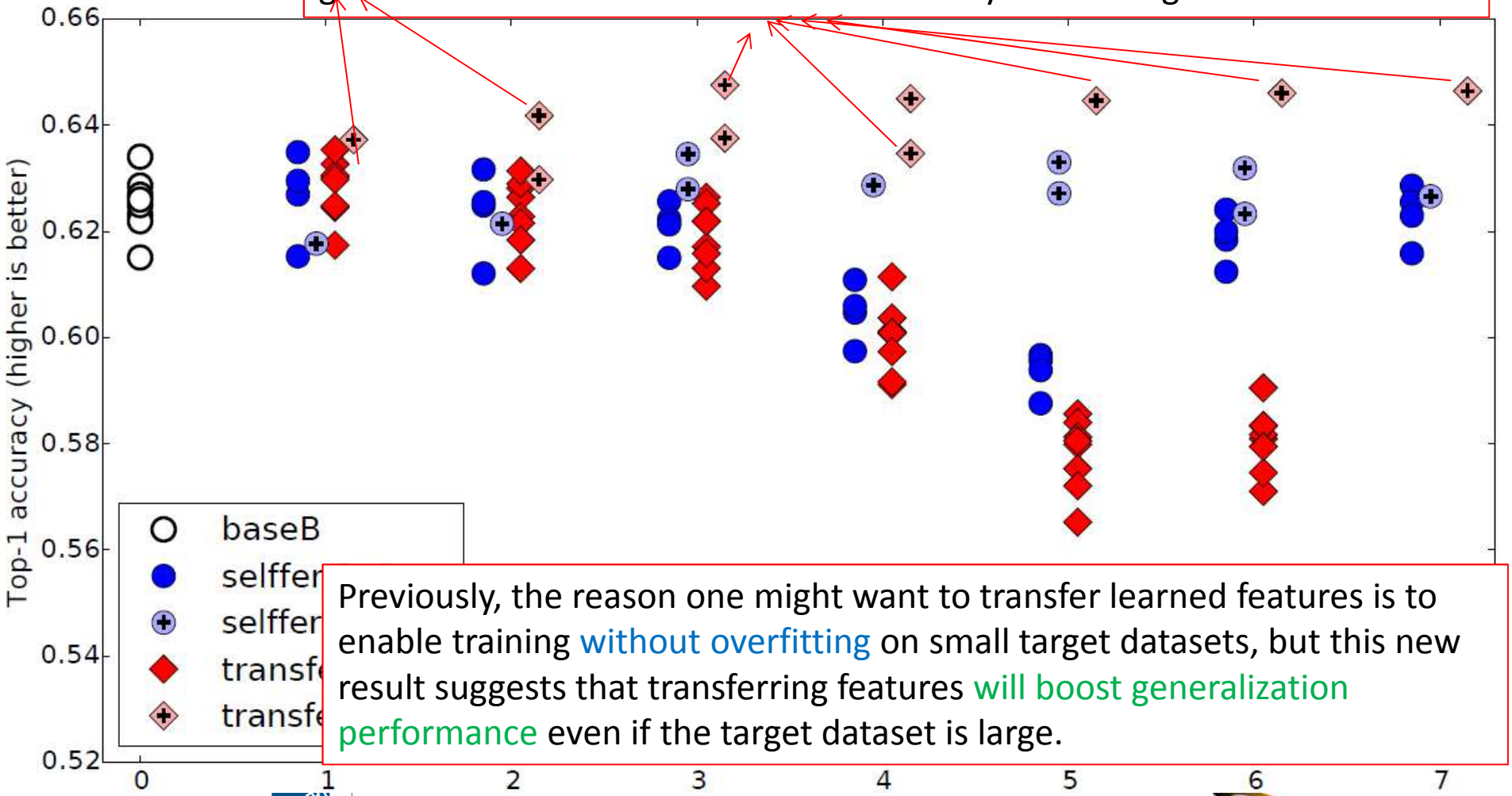


The dark red AnB diamonds show the effect we set out to measure in the first place: the transferability of features from one network to another at each layer.
Layers one and two transfer almost perfectly from A to B, giving evidence that, at least for these two tasks, not only are the first-layer Gabor and color blob features general, but the second layer features are general as well



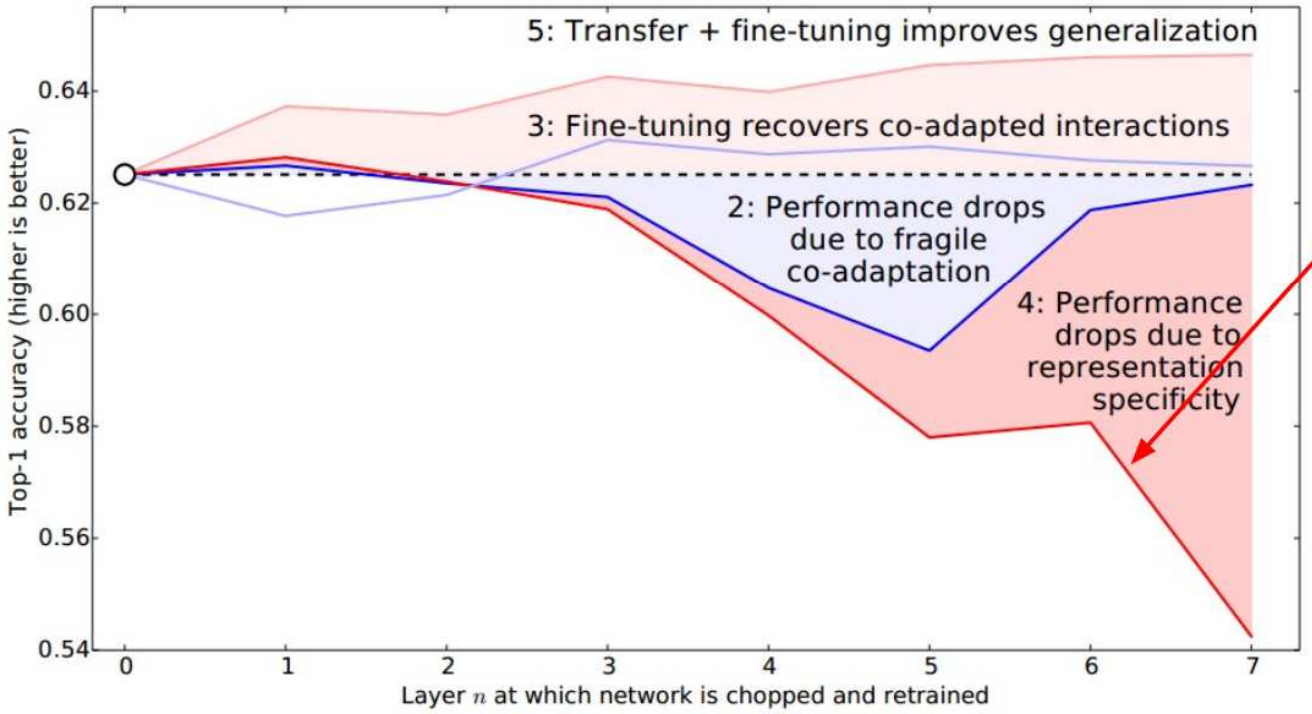
Main results

The light red AnB+ diamonds show a particularly surprising effect: that transferring features and then fine-tuning them results in networks that generalize better than those trained directly on the target dataset.



Previously, the reason one might want to transfer learned features is to enable training without overfitting on small target datasets, but this new result suggests that transferring features will boost generalization performance even if the target dataset is large.

How transferable are features in deep neural networks?
[Yosinski et al., 2014]

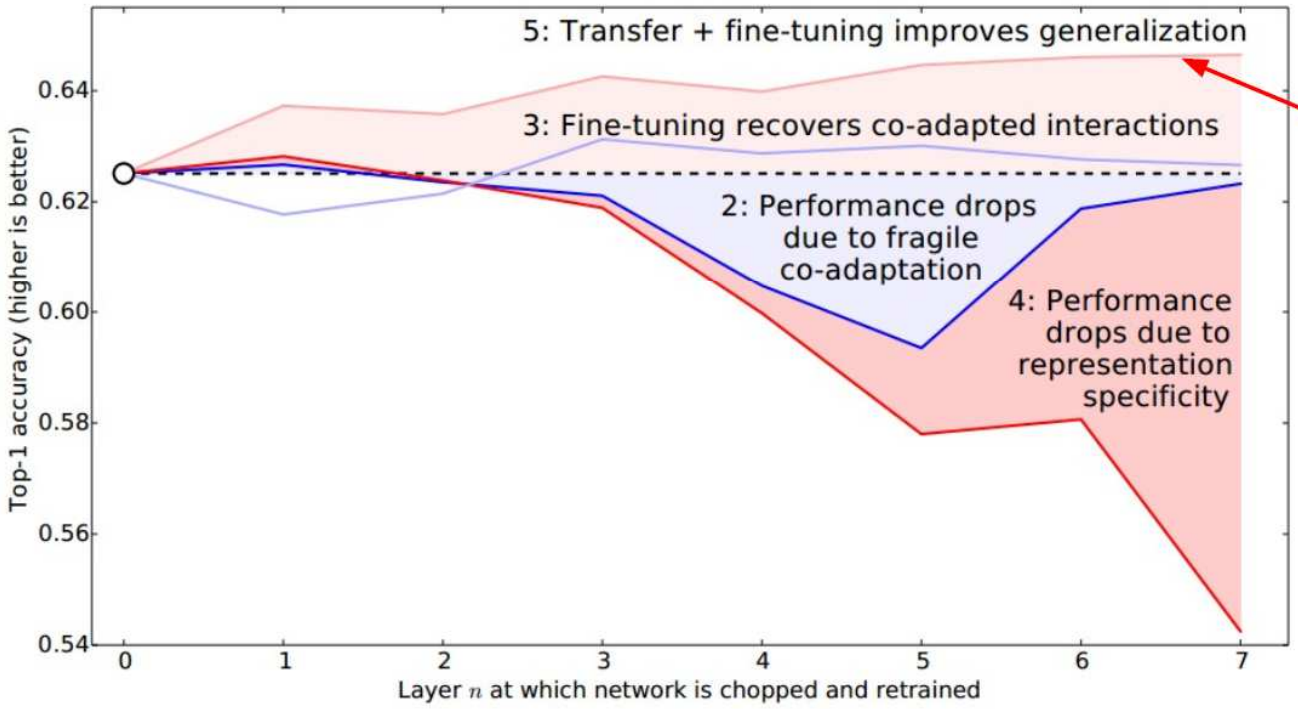


Split ImageNet classes in half to two sets: A/B.

Train on A, fix the first n layers, reinit layers n+, train on B, test on B val.

=> performance degrades because representation higher up is too A-specific

How transferable are features in deep neural networks?
[Yosinski et al., 2014]

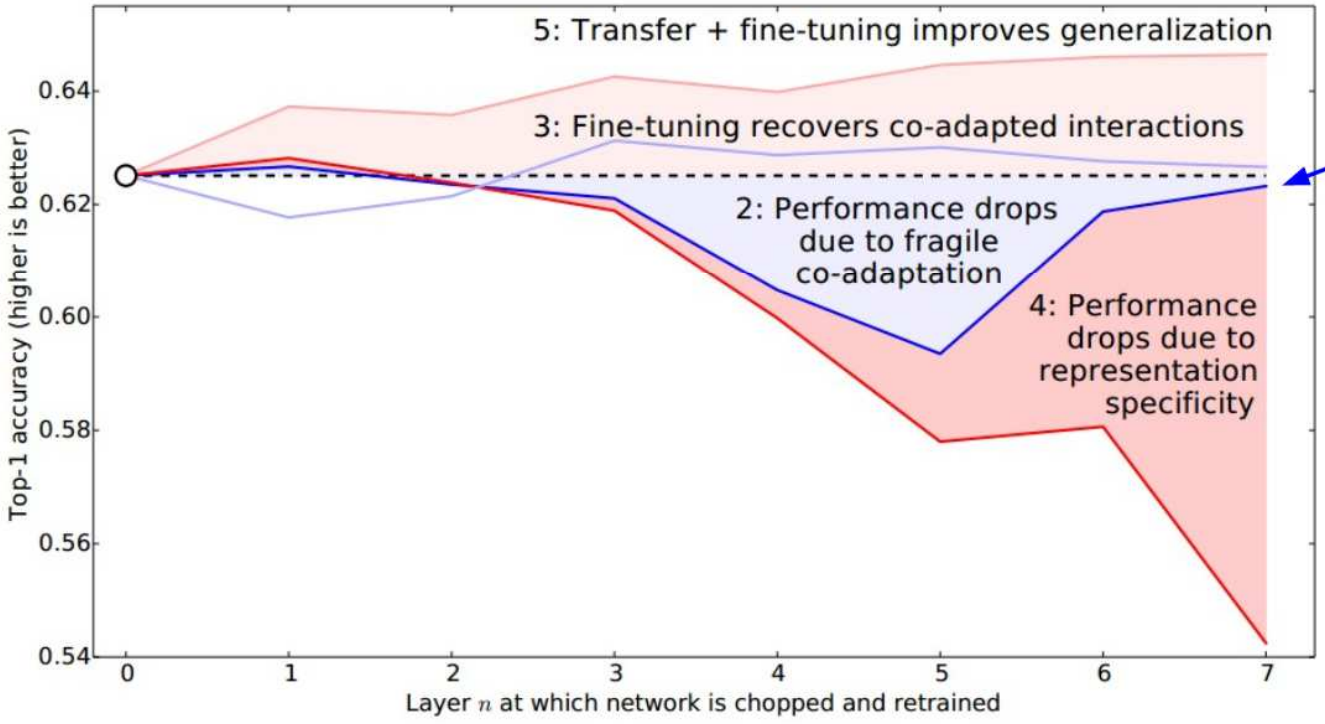


Split ImageNet classes in half to two sets: A/B.

Train on A, reinit layers n+, train on B, test on B val.

=> the information from once seeing data from A seems to linger, gives better generalization

How transferable are features in deep neural networks?
[Yosinski et al., 2014]

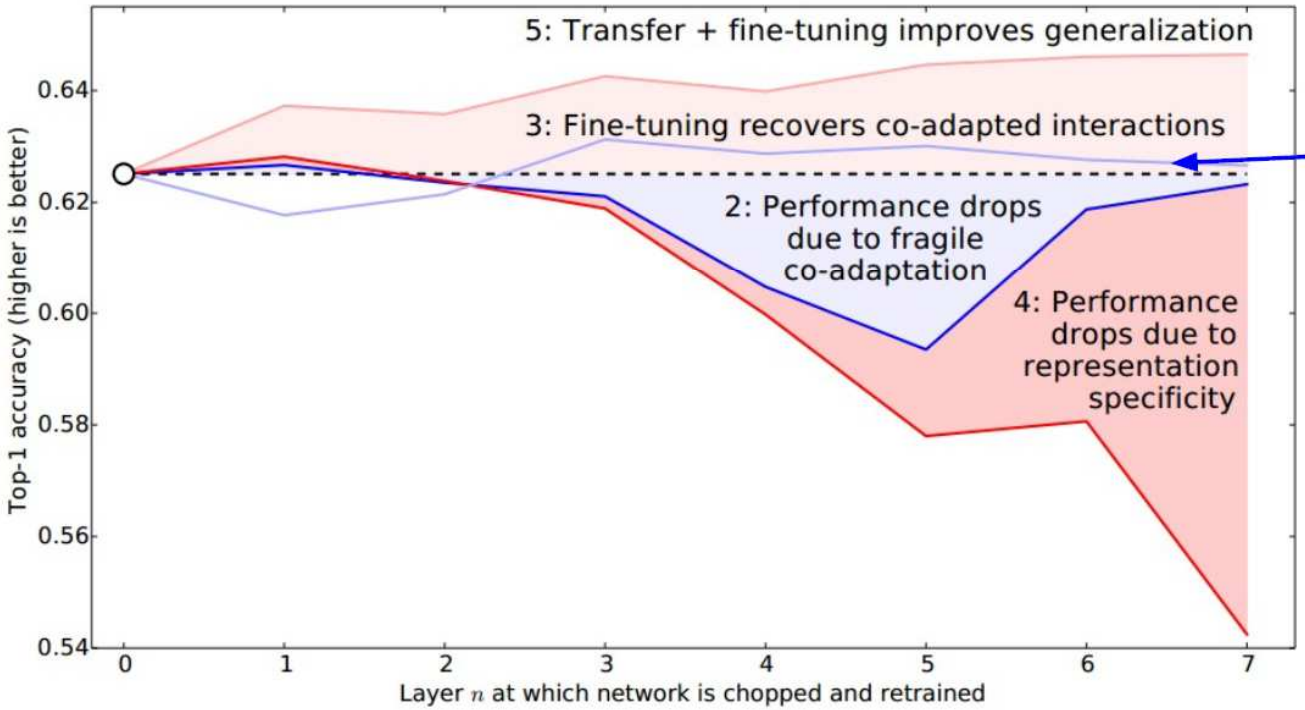


Split ImageNet classes in half to two sets: A/B.

Train on B, fix the first n layers, re-initialize layers n+, train on B again and test on B val

=> performance degrades when n = 4. wat.

How transferable are features in deep neural networks?
[Yosinski et al., 2014]

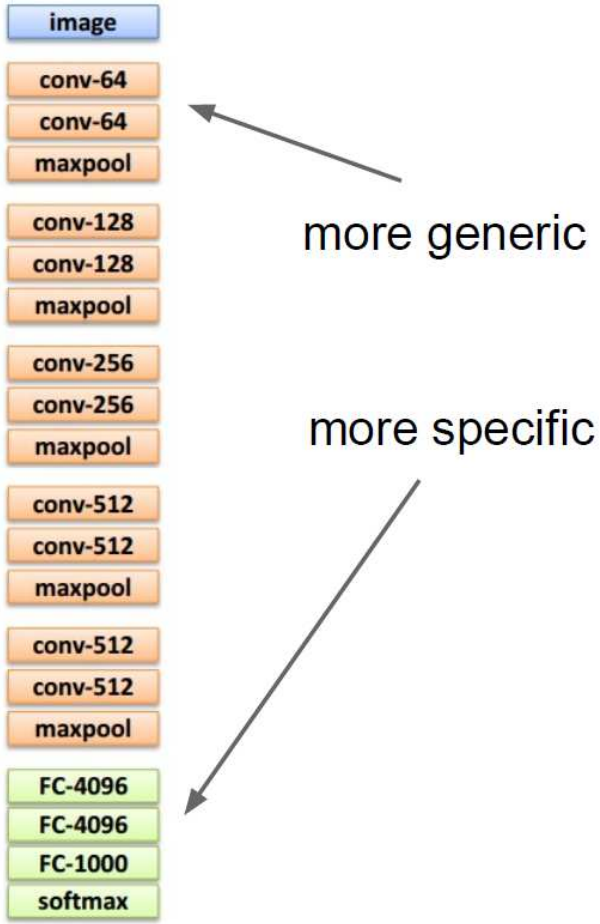


Split ImageNet classes in half to two sets: A/B.

Train on B, reinitialize layers n+, train on B again and test on B val.

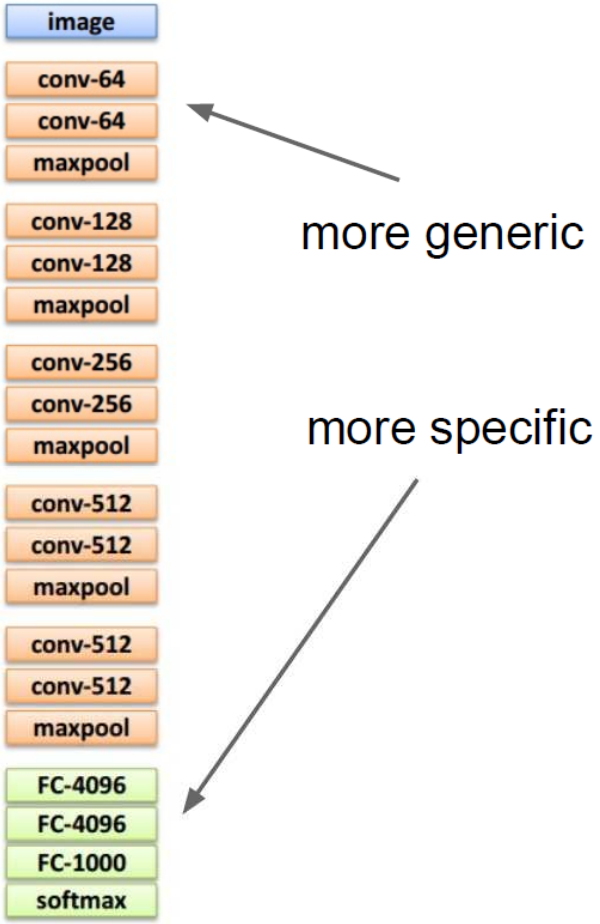
=> performance doesn't degrade anymore

Take home messages



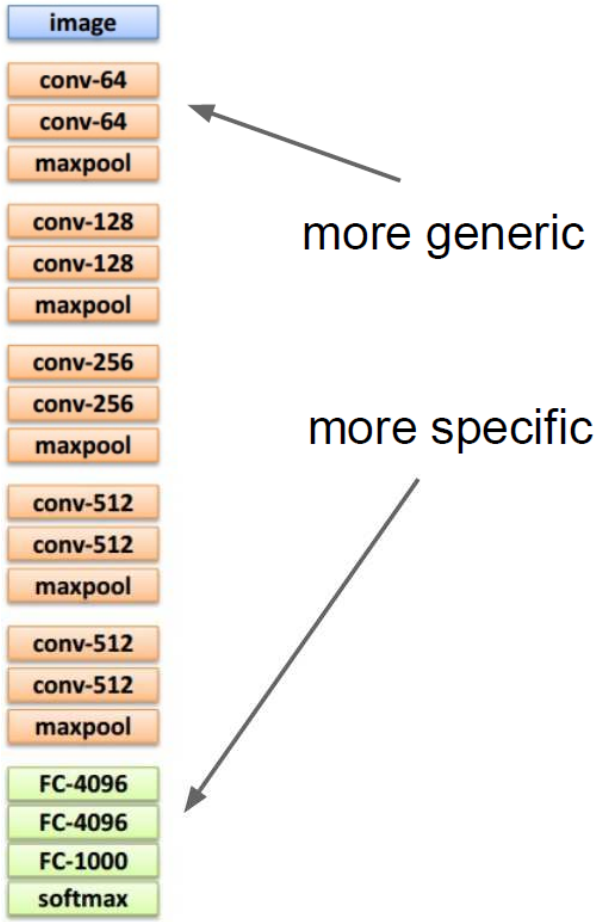
	very similar dataset	very different dataset
very little data	?	?
quite a lot of data	?	?

Take home messages



	very similar dataset	very different dataset
very little data	Use Linear Classifier on top layer	?
quite a lot of data	Finetune a few layers	?

Take home messages



	very similar dataset	very different dataset
very little data	Use Linear Classifier on top layer	You're in trouble… Try linear classifier from different stages
quite a lot of data	Finetune a few layers	Finetune a larger number of layers

Some other issue!

- R1 Q2: Why the pre-trained network that is trained cifar10 dataset is utilized?

- R1 Q4: Although the method achieves state-of-the art, it seems lack of novelty
- R2 Q2: The authors need to show the scientific contributions in tracking or CDBN.

Summary

- We talked about the transfer learning with the CovNets
  - Retrain only classifiers
  - Add one or two full connected layers
  - Fine-tune more