Understanding visual representations

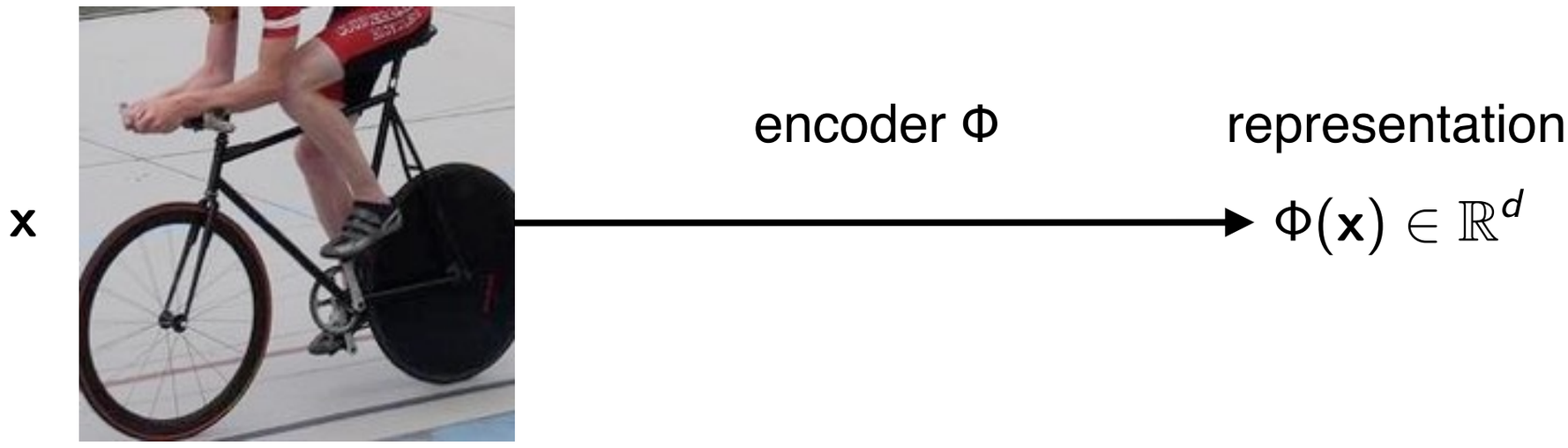
Andrea Vedaldi

Workshop on Deep Learning: Theory and Practice

July 2016

UNIVERSITY OF
OXFORD

Image representations



An **encoder** maps the data into a **vectorial representation**

Facilitate labelling of images, text, sound, videos, …

Understanding visual representations

Intro

Visualizing representations

Backpropagation networks and “deconvolution”

Representations: equivalence & transformations

Understanding visual representations

Intro

Visualizing representations

Backpropagation networks and “deconvolution”

Representations: equivalence & transformations

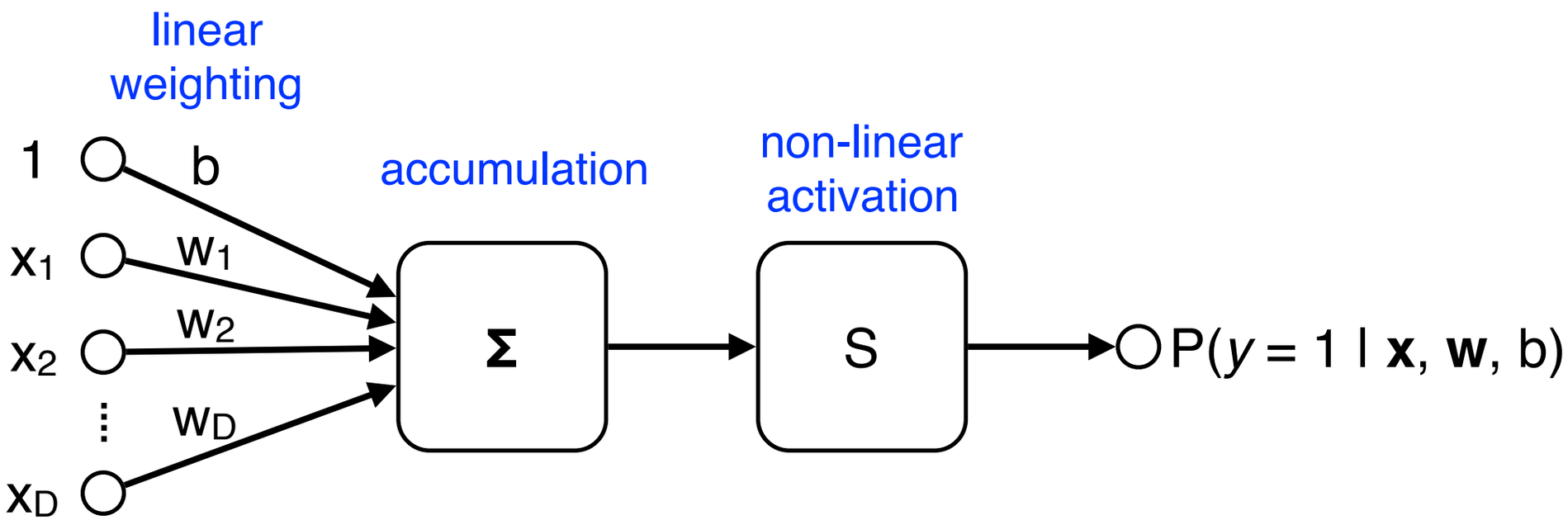
Convolutional neural networks

Origin (1950-60)

Perceptron

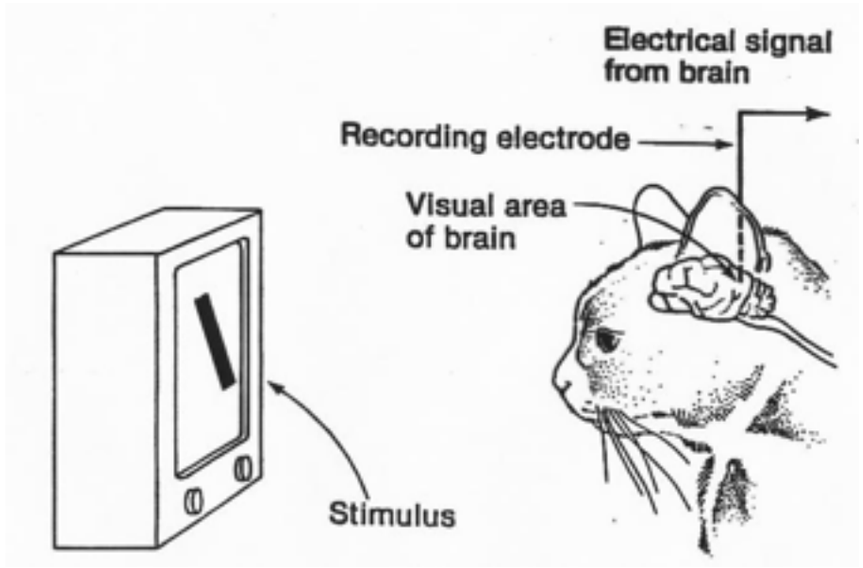
[Rosenblatt 57]

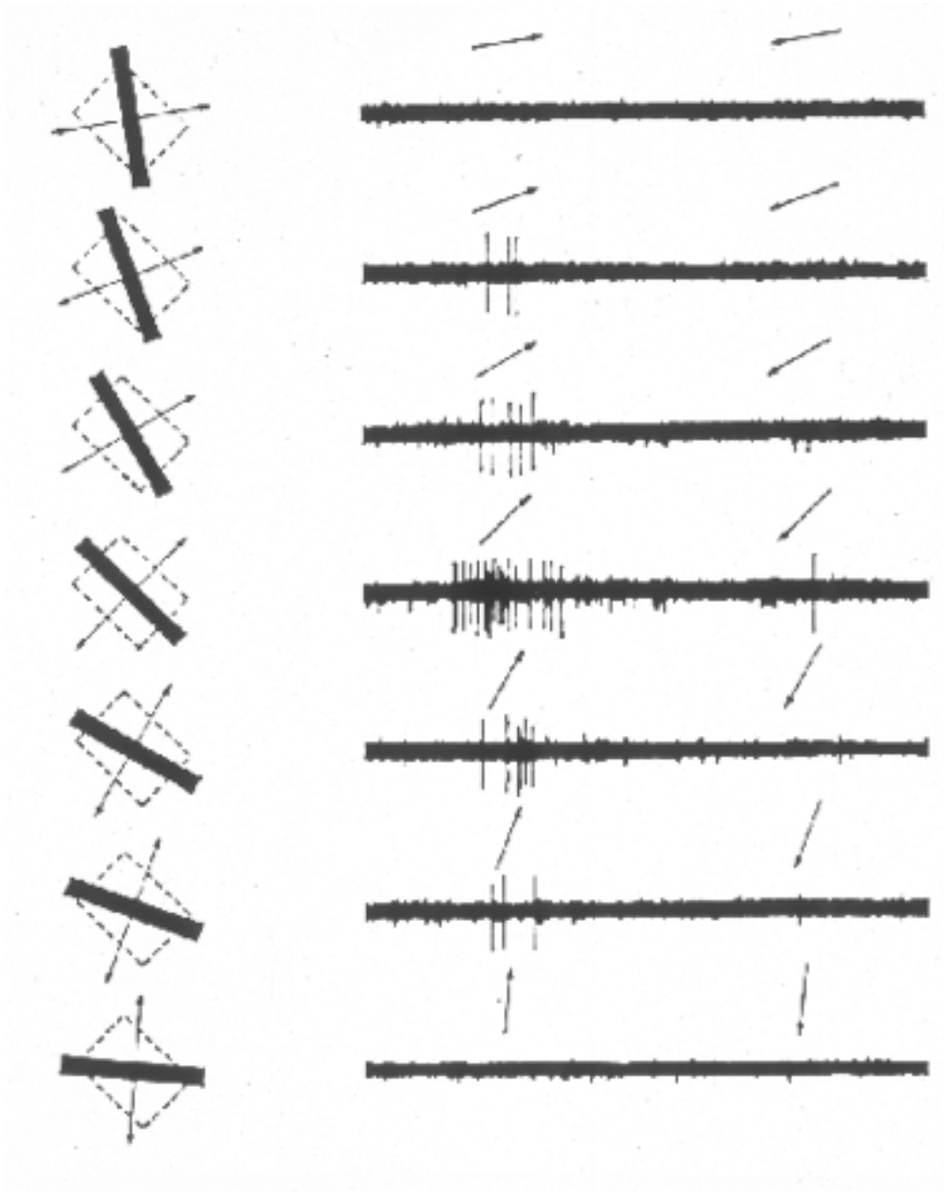
The goal is estimating the posterior probability of the binary label *y* of a vector **x**:

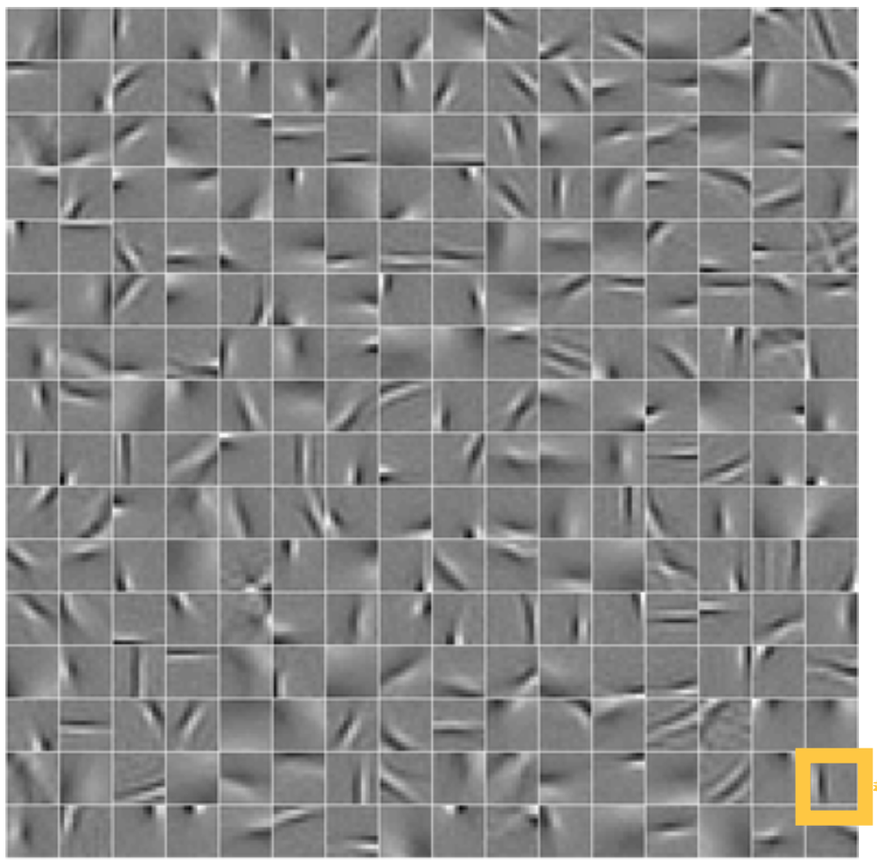


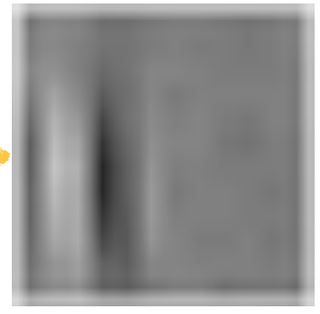
Discovery of oriented cells in the visual cortex

[Hubel and Wiesel 59]

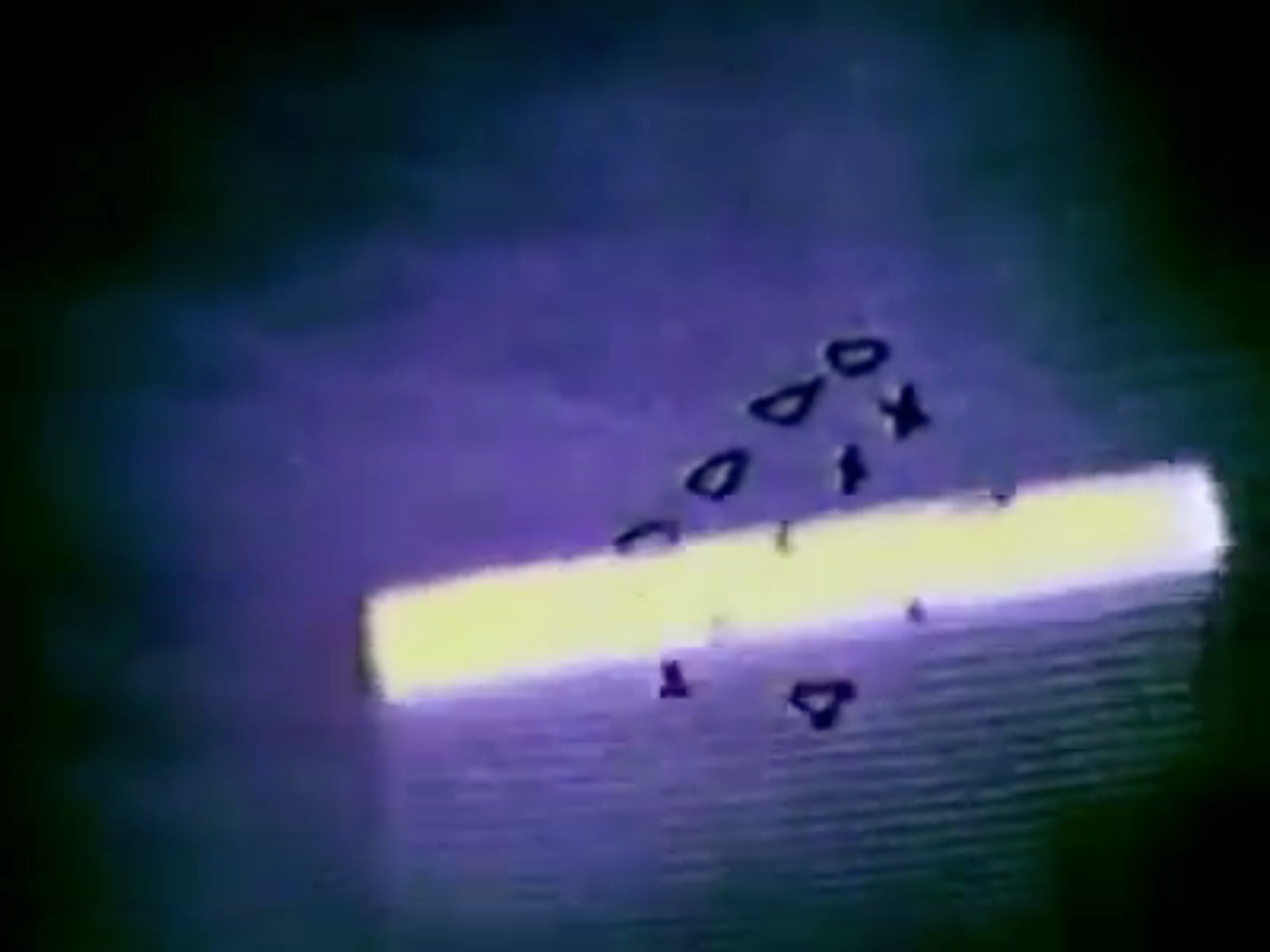






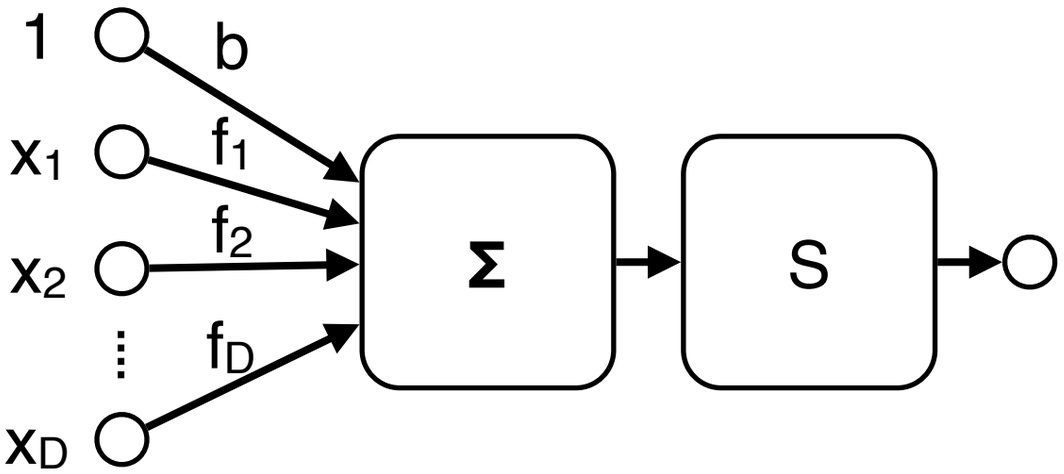


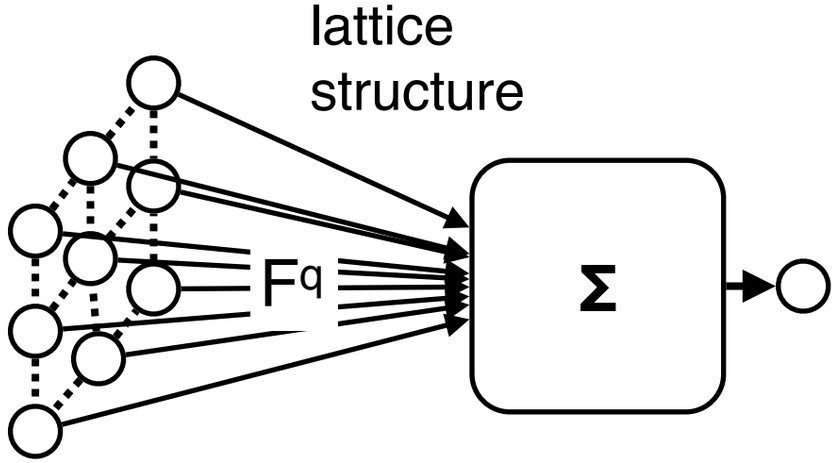
oriented filter

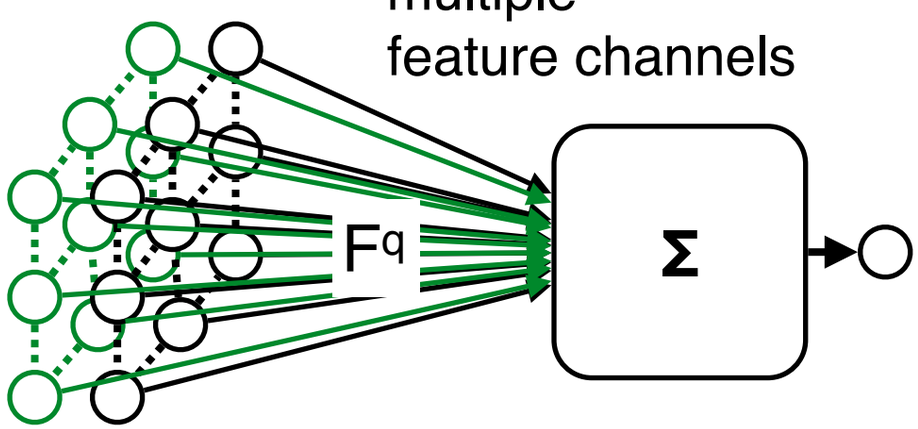


Linear convolution

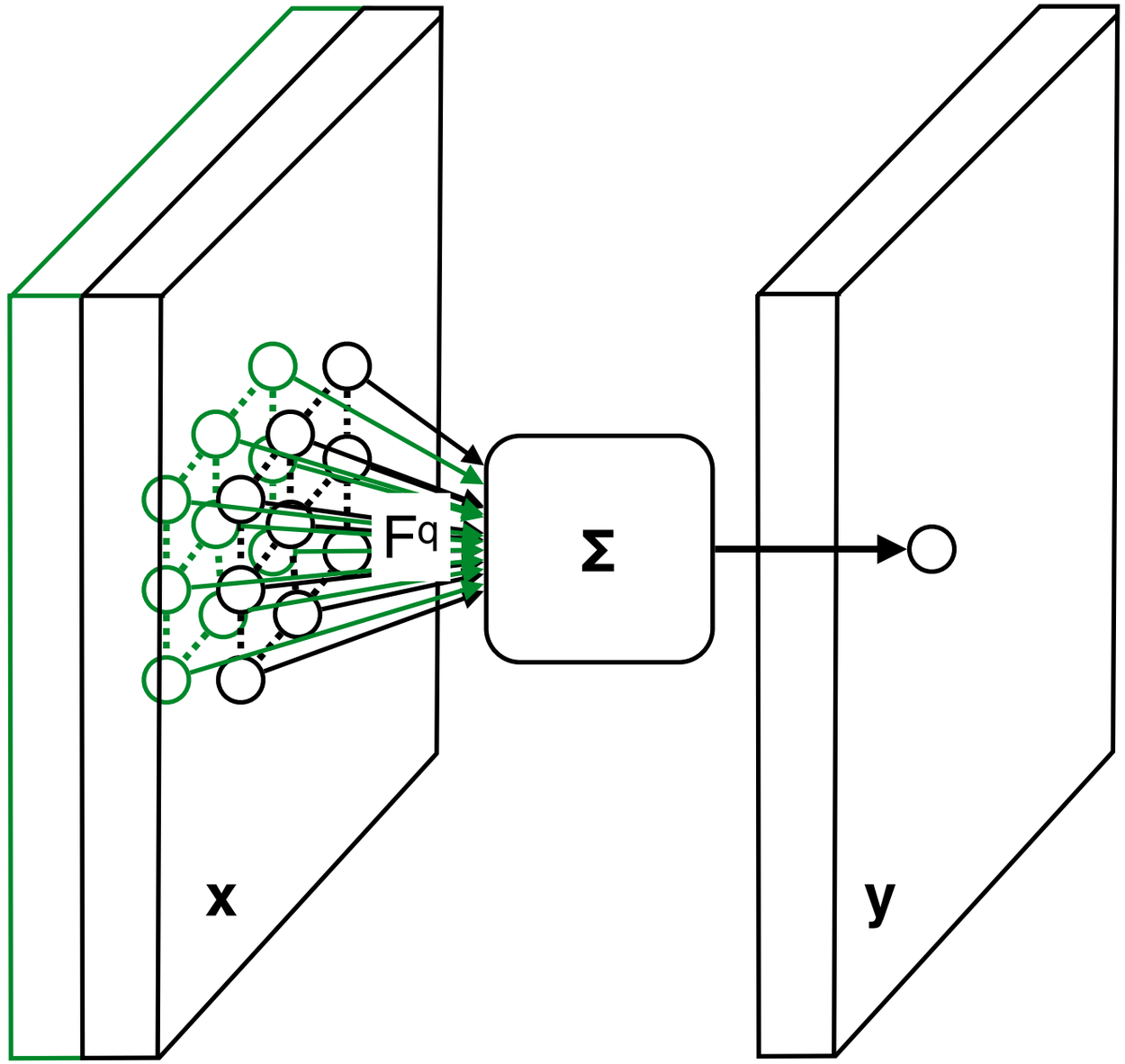
As a neural network





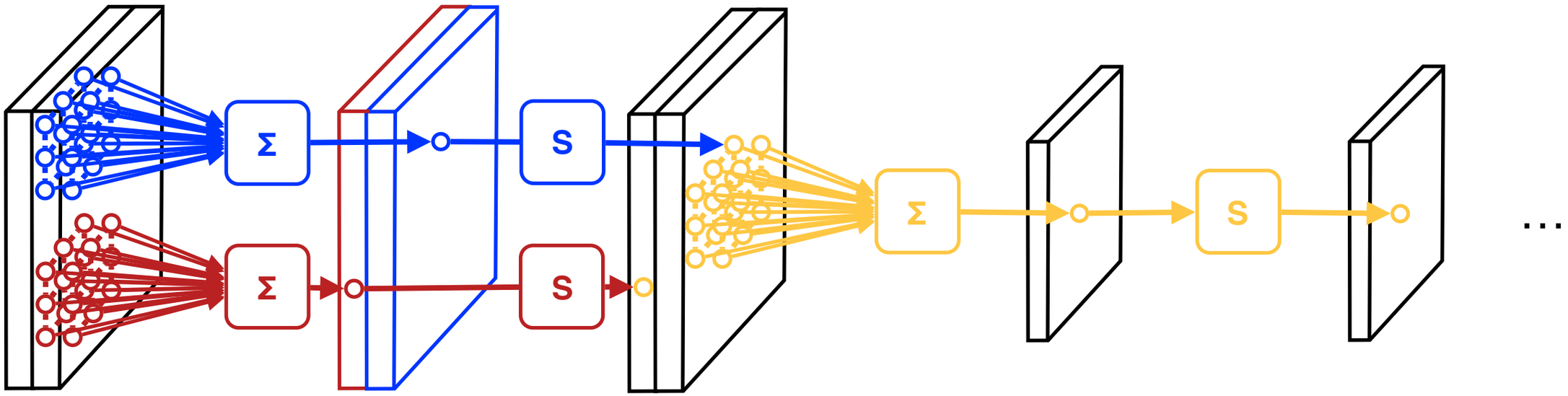


local and translation
invariant action



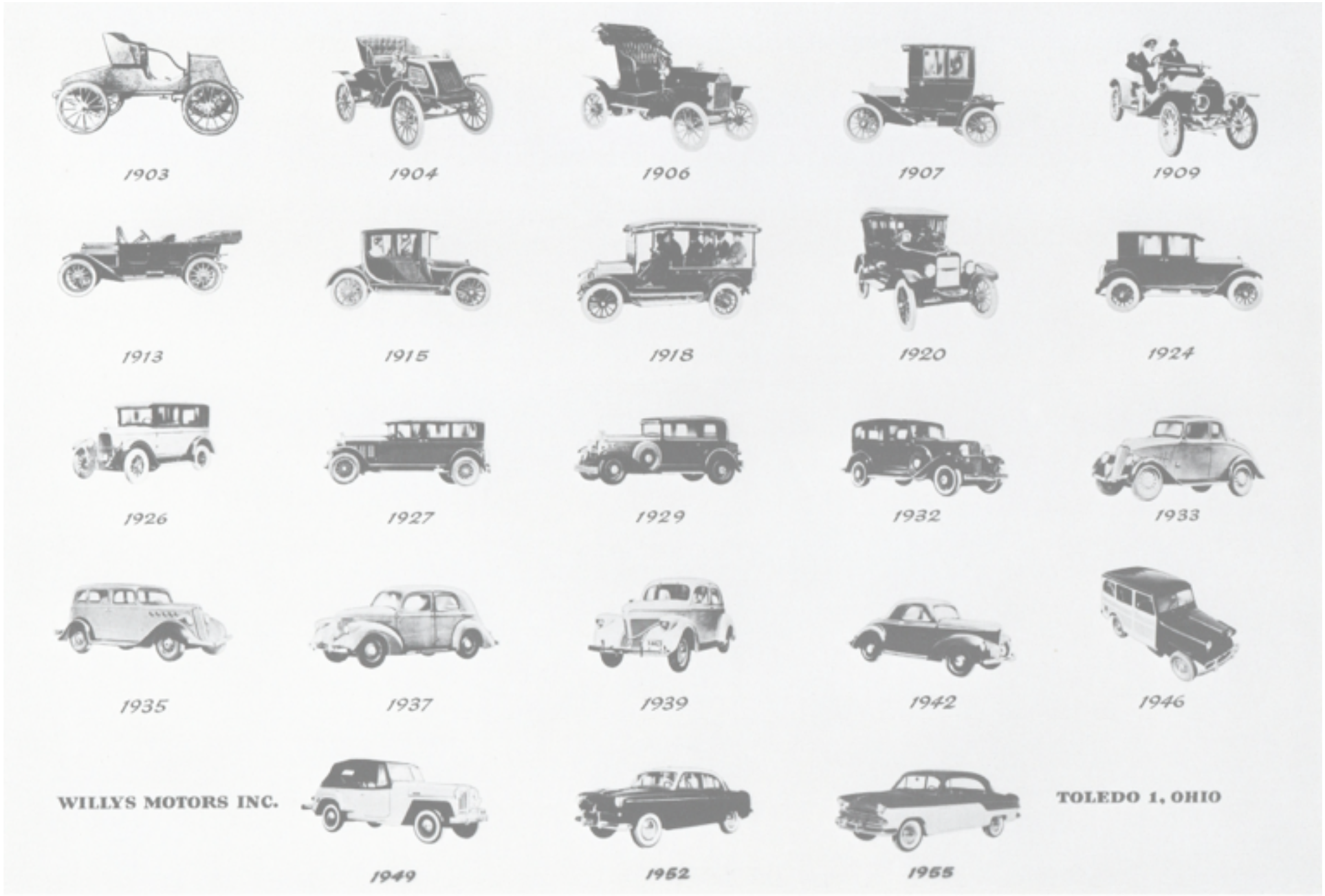
Deep architectures

Repeat linear / non-linear operators

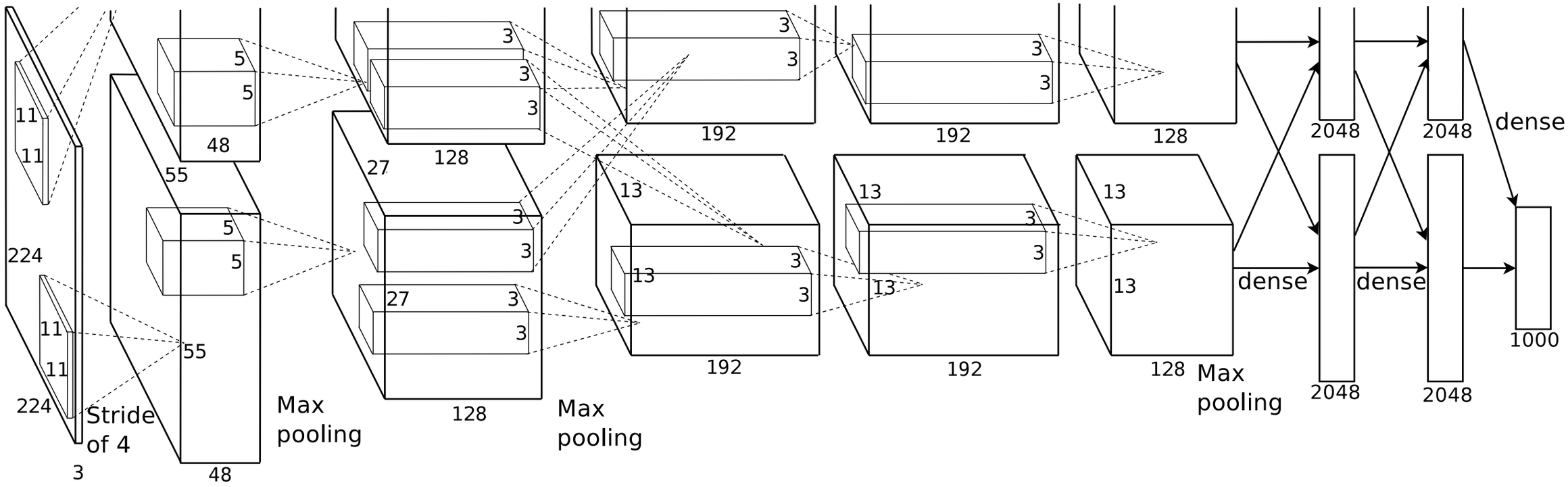


Modern convolutional networks

From AlexNet (2012) to ResNet (2015)



Modern convolutional nets



[AlexNet by Krizhevsky et al. 2012]

Excellent **performance** in image understanding tasks

Learn a sequence of **general-purpose representations**

Millions of parameters learned from data

The “**meaning**” of the representation is **unclear**

How deep is deep enough?

AlexNet (2012)

5 convolutional layers

3 fully-connected layers

How deep is deep enough?

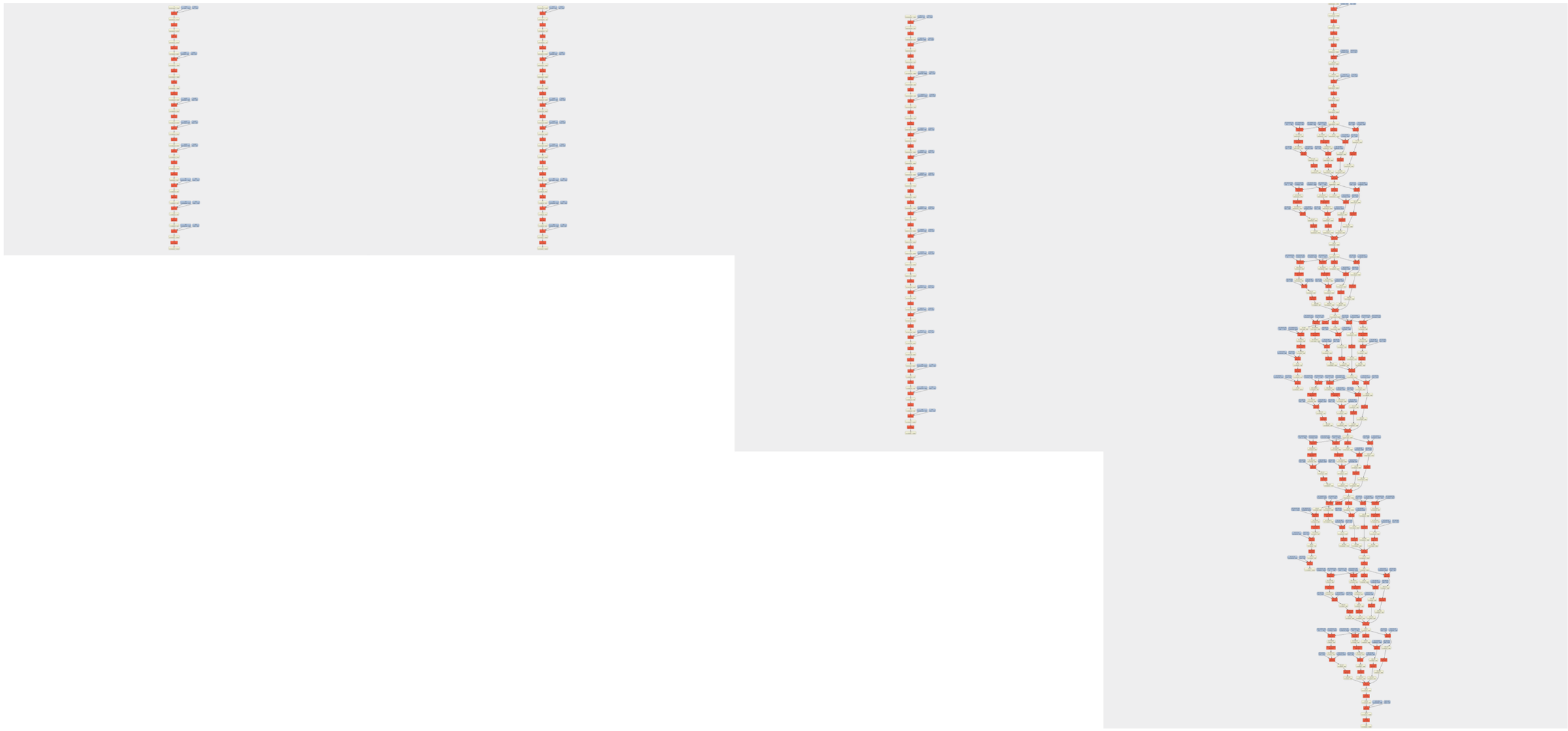
AlexNet (2012)

VGG-M (2013)

VGG-VD-16 (2014)

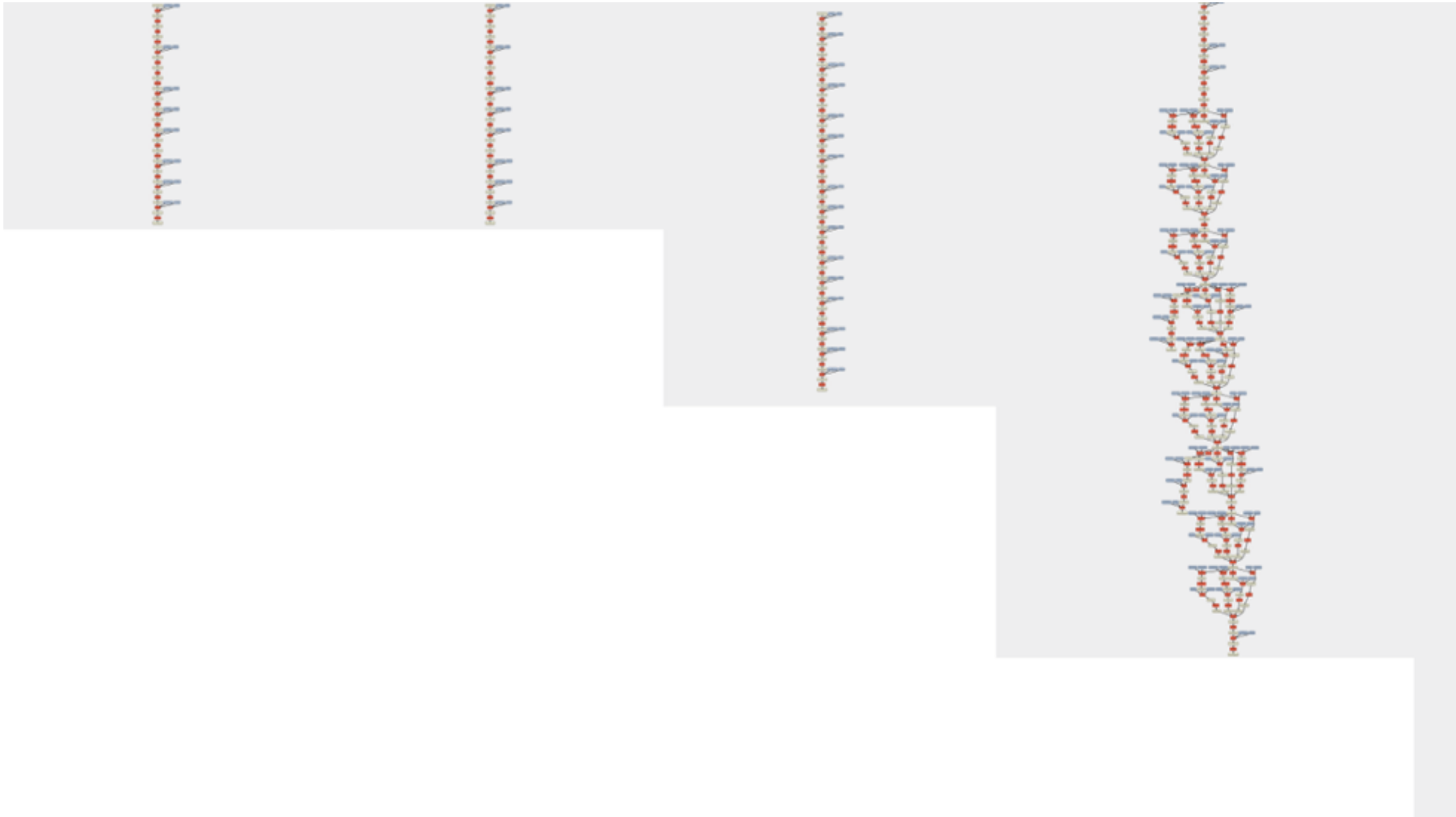
How deep is deep enough?

AlexNet (2012) VGG-M (2013) VGG-VD-16 (2014) GoogLeNet (2014)

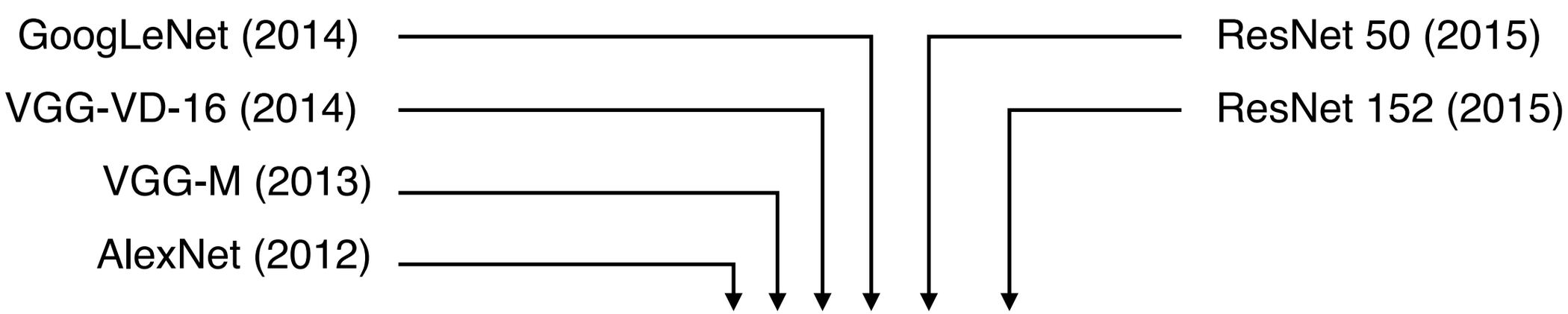


How deep is deep enough?

AlexNet (2012) VGG-M (2013) VGG-VD-16 (2014) GoogLeNet (2014)



How deep is deep enough?



16 convolutional layers

50 convolutional layers

152 convolutional layers

Krizhevsky, I. Sutskever, and G. E. Hinton. *ImageNet classification with deep convolutional neural networks*. In Proc. NIPS, 2012.

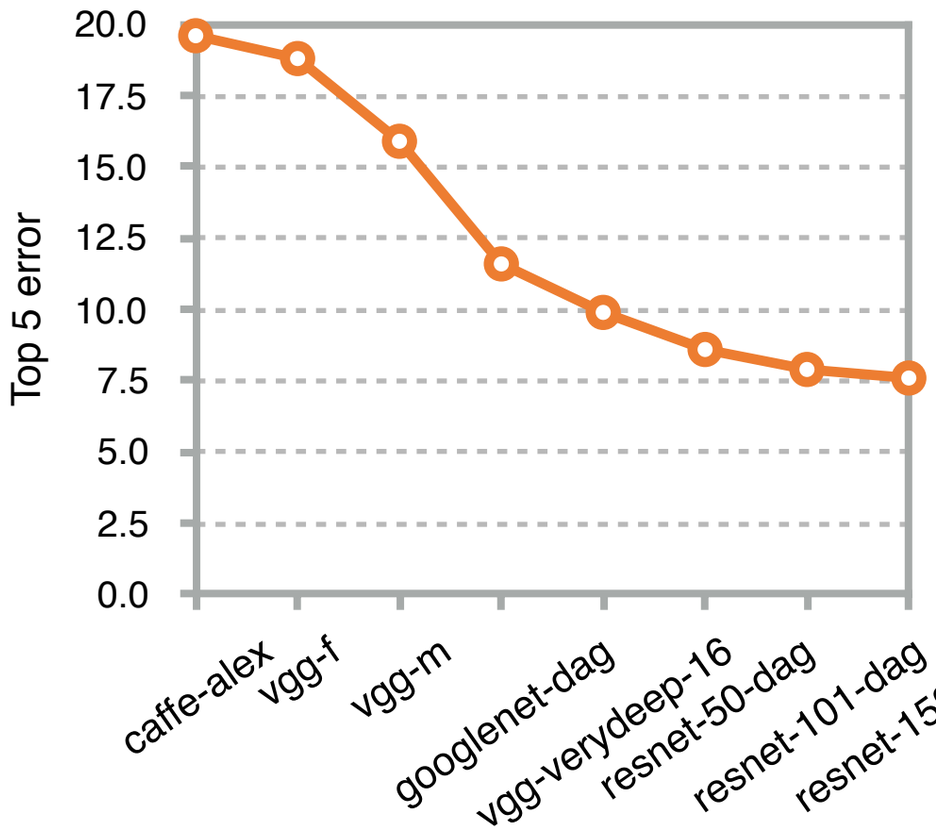
C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. *Going deeper with convolutions*. In Proc. CVPR, 2015.

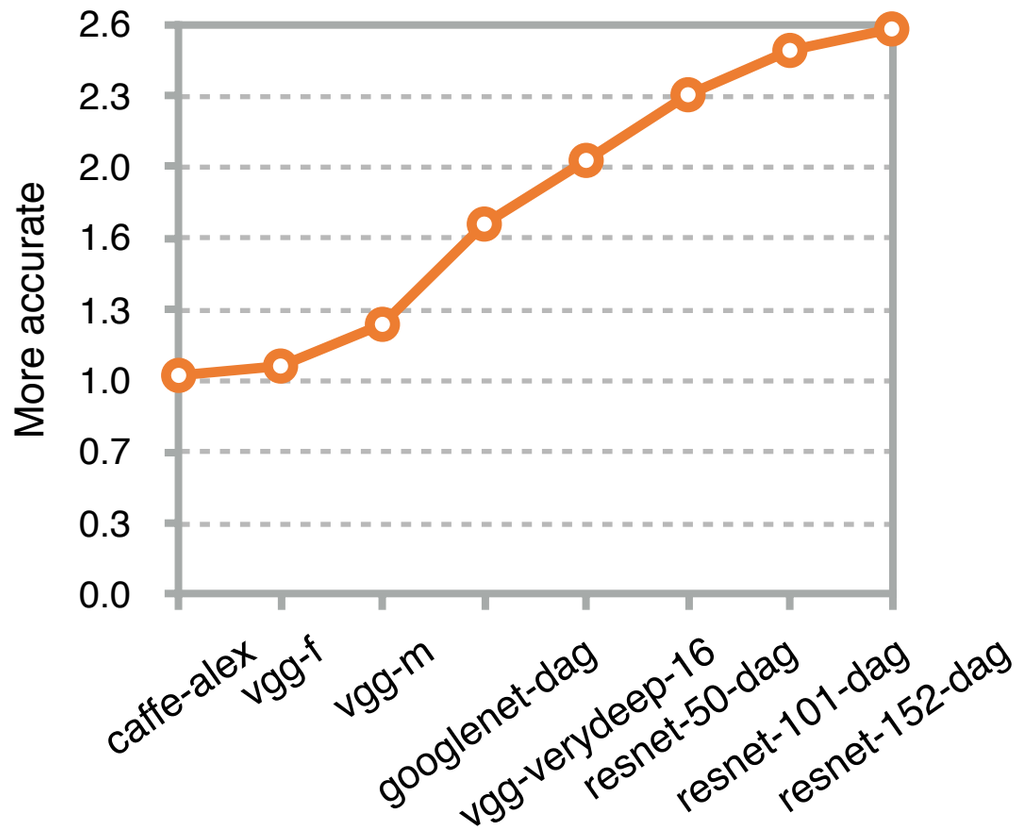
K. Simonyan and A. Zisserman. *Very deep convolutional networks for large-scale image recognition*. In Proc. ICLR, 2015.

K. He, X. Zhang, S. Ren, and J. Sun. *Deep residual learning for image recognition*. In Proc. CVPR, 2016.

Accuracy

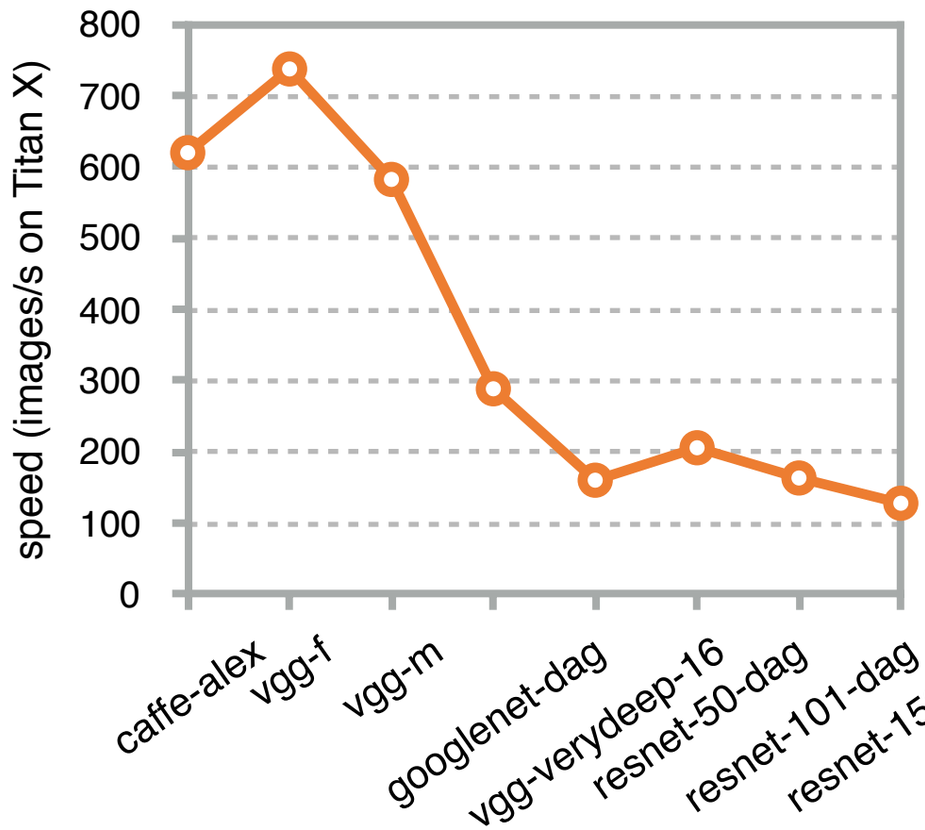
3 × more accurate in 3 years

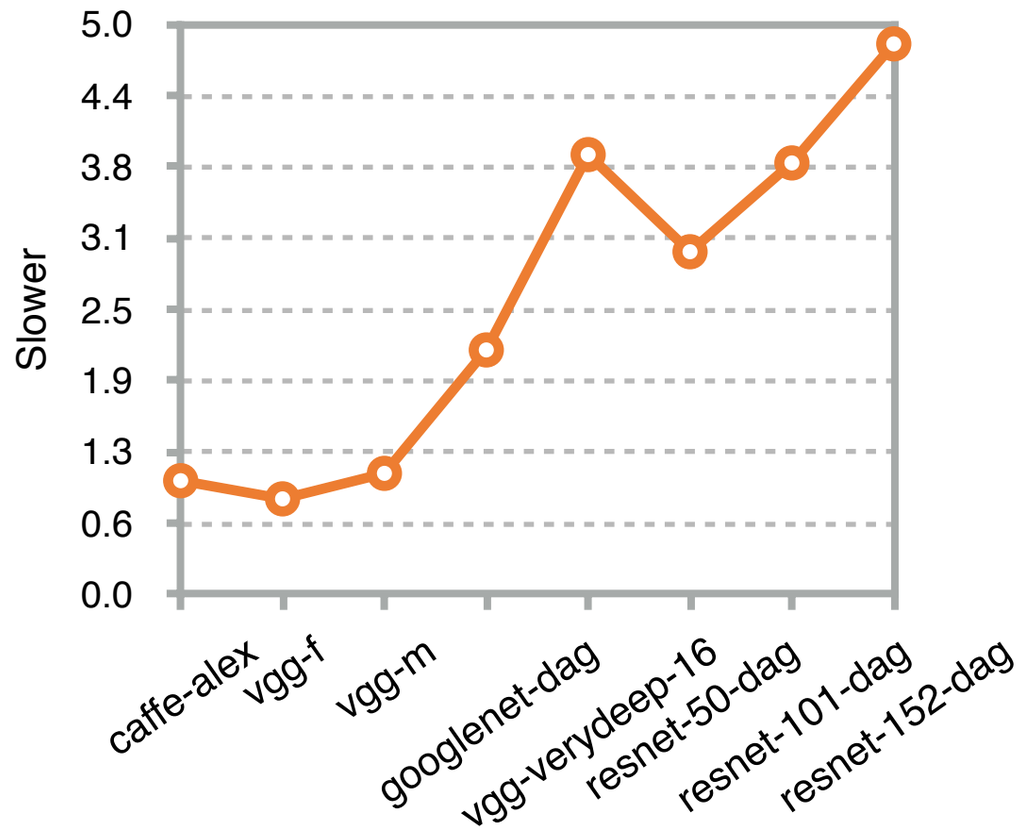




Speed

5 × slower





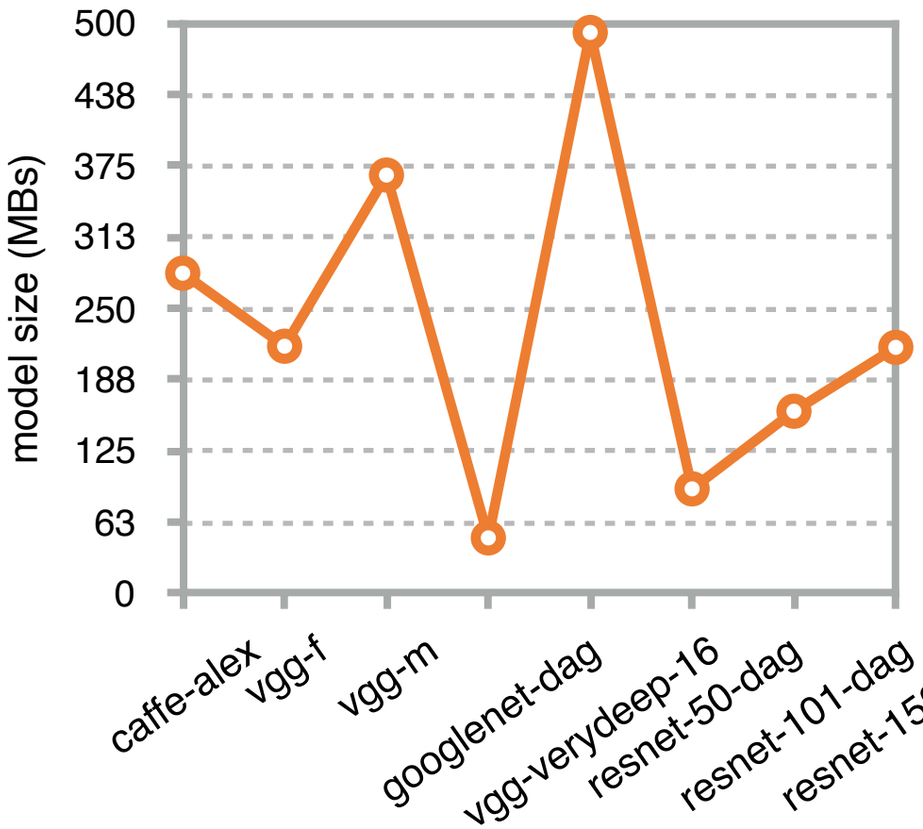
Remark: *101 ResNet* layers same size/speed as *16 VGG-VD* layers

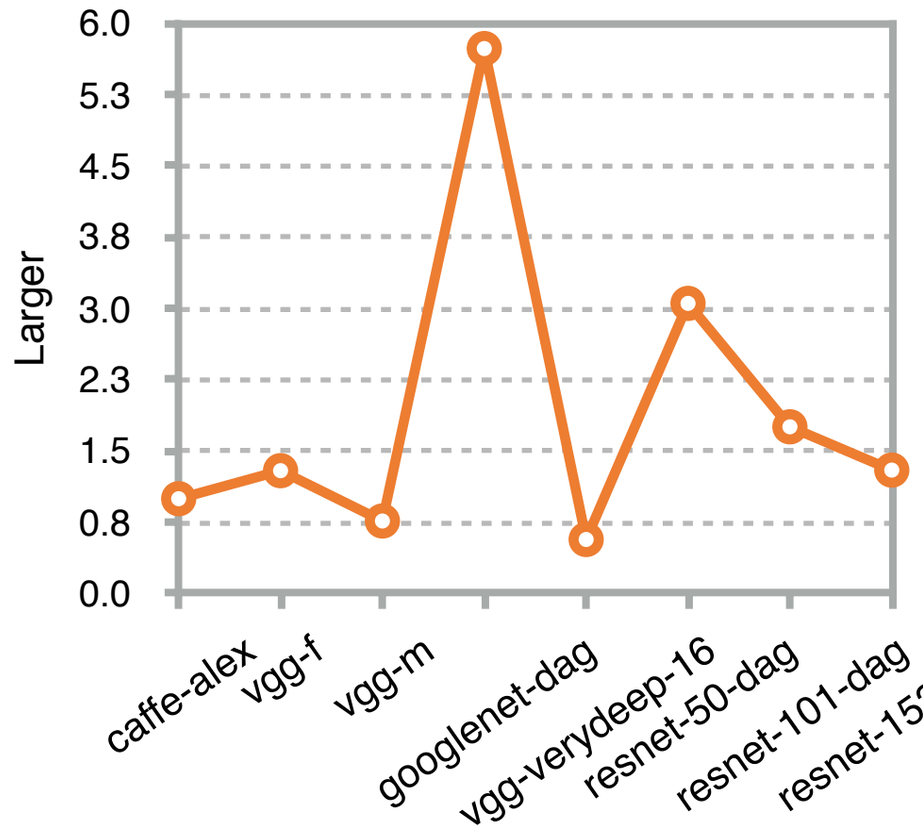
Reason: far fewer feature channels (quadratic speed/space gain)

Moral: optimize your architecture

Model size

Num. of parameters is about the same





Remark: *101 ResNet* layers same size/speed as *16 VGG-VD* layers

Reason: far fewer feature channels (quadratic speed/space gain)

Moral: optimize your architecture

Deep nets in vision: 2012-2015

Impact of deep learning in vision

- **2012** amazing results by AlexNet in the ImageNet challenge
- **2013-15** massive 3x improvement
- **2016-19** more improvements?

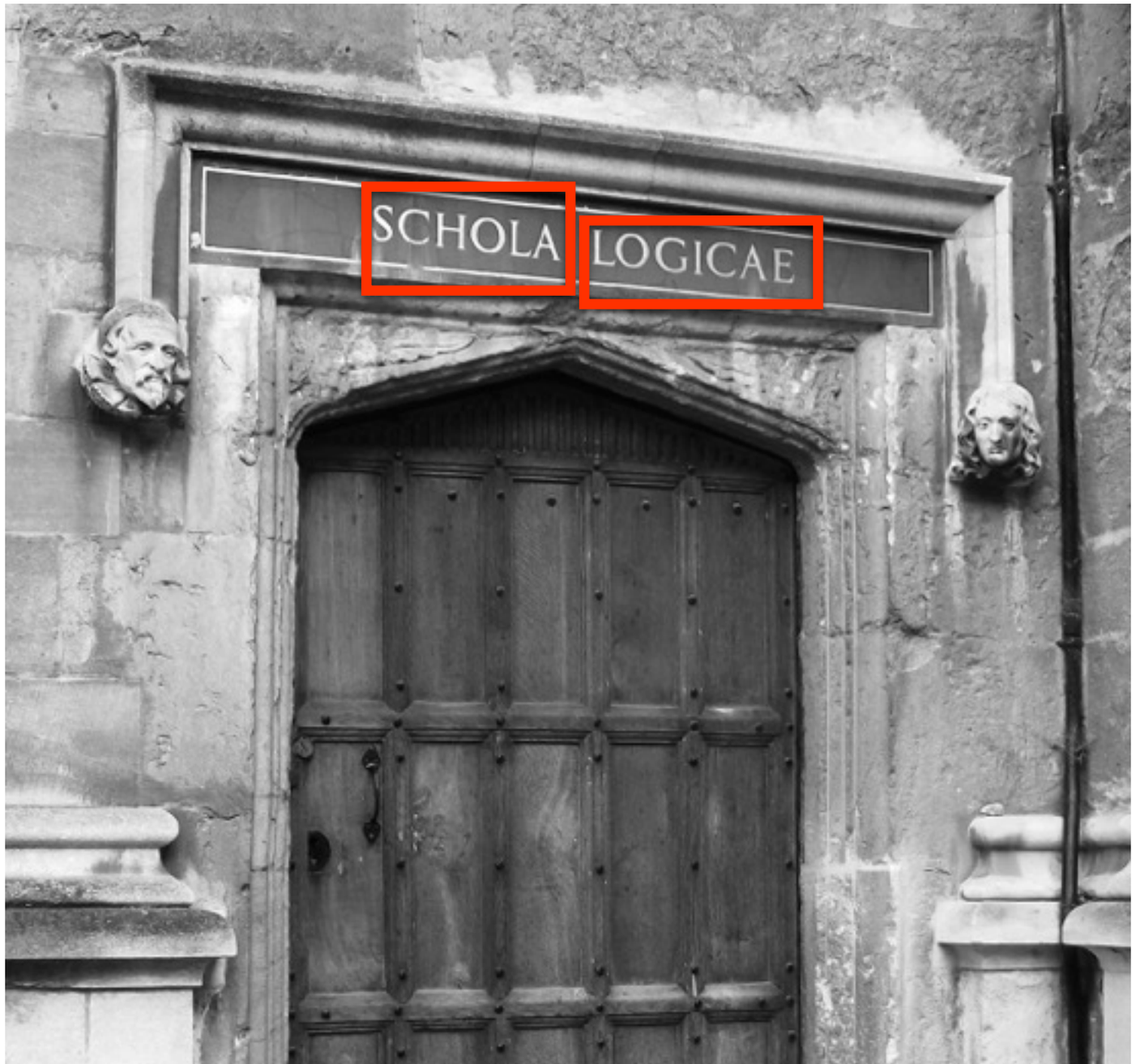
What have we learned

- Several **incremental tweaks** over the base AlexNet
- There is still space for improvements to the base model

Things that work

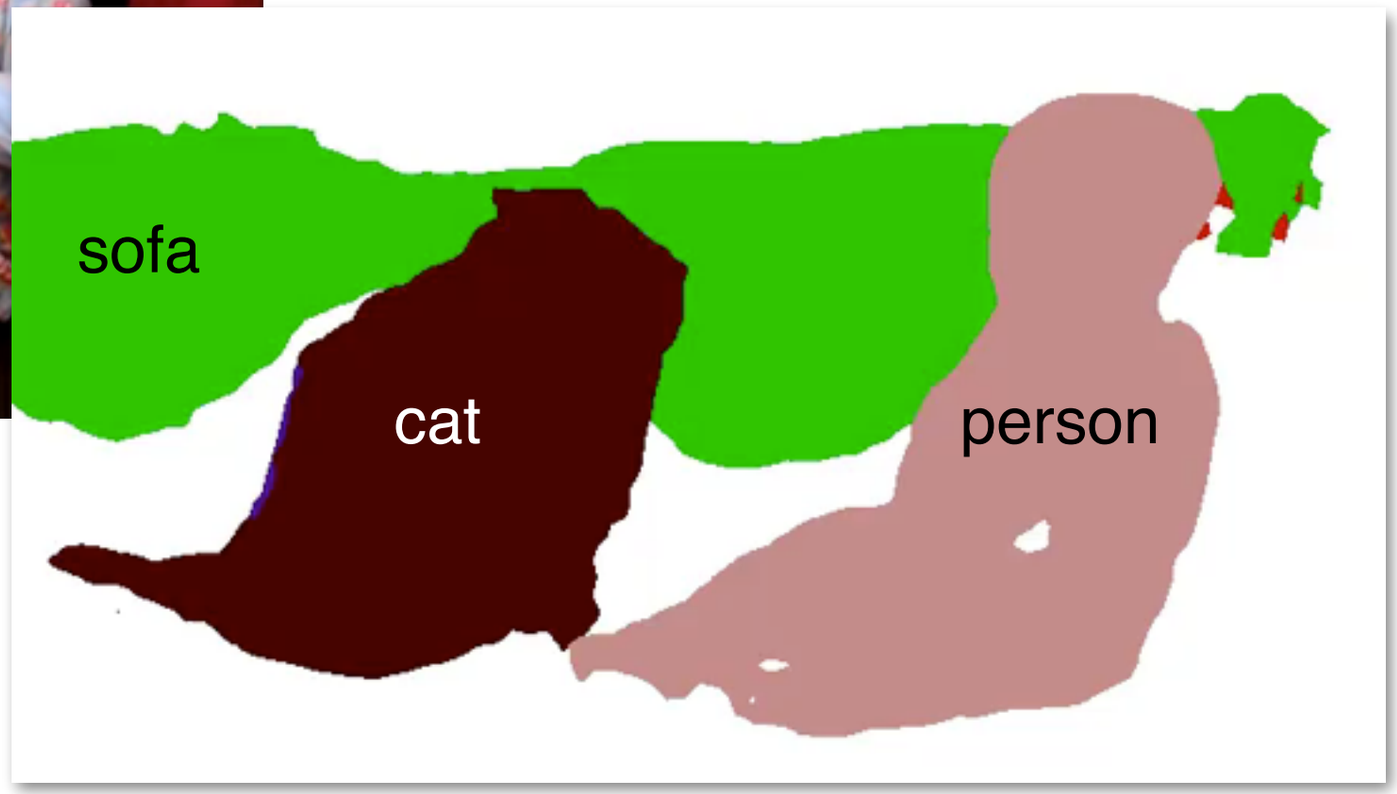
- Deeper architectures
- Smarter architectures (groups, low rank decompositions, …)
- Batch normalization
- Residual connections

Applications



Semantic image segmentation

Label individual pixels



Face analysis

Detection, verification, recognition, emotion, 3D fitting

E.g. VGG-Face

Text spotting

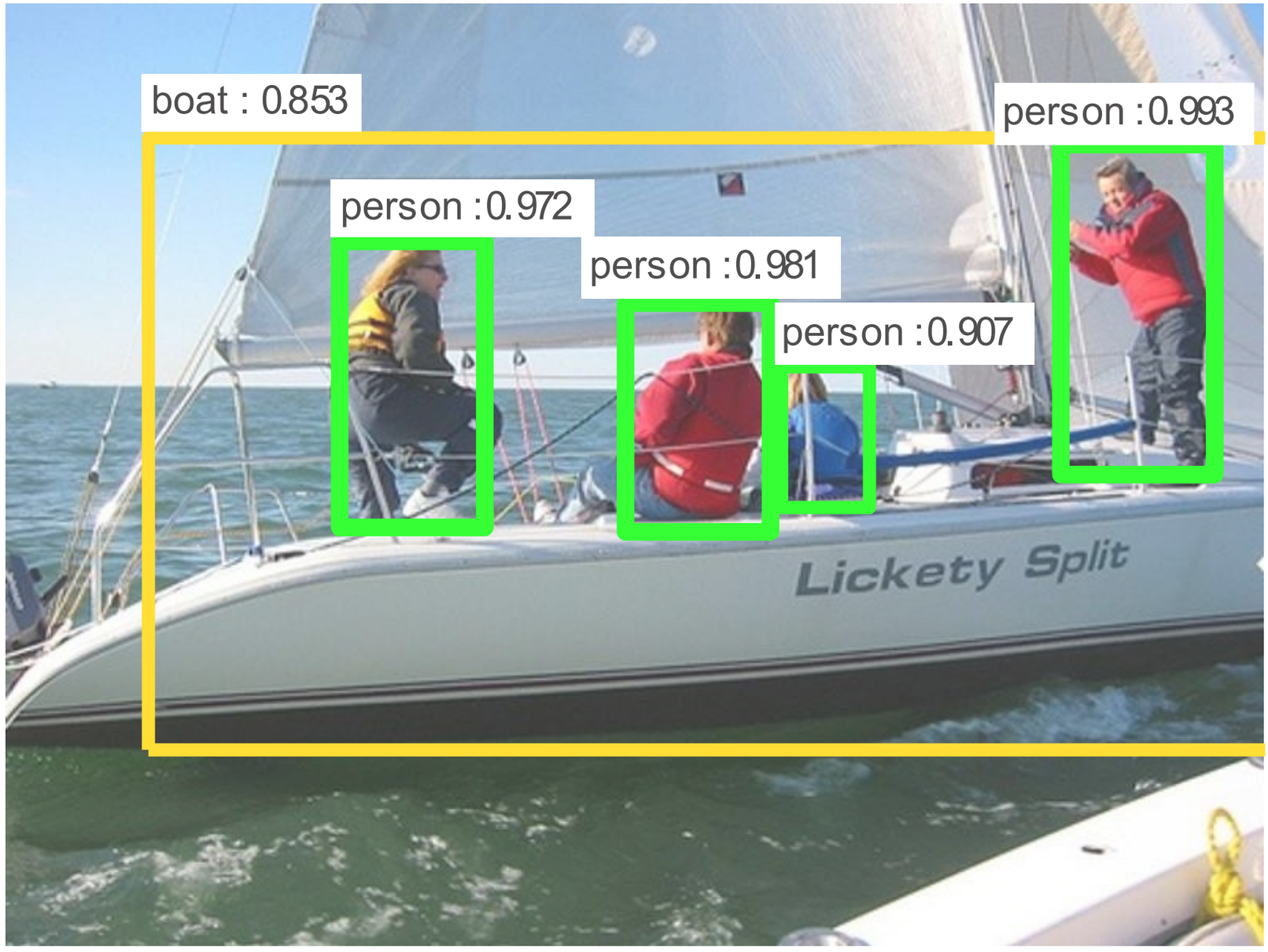
Detection, word recognition, character recognition

E.g. SynthText and VGG-Text

http://zeus.robots.ox.ac.uk/textsearch/#/search/

Object detection

Extract individual object instances



Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation
R. Girshick, J. Donahue, T. Darrell, J. Malik, CVPR 2014

Supervision required

Still a major limitation

Can get around in various ways, for example using **synthetic data**

Weakly-supervised learning

Use partial labelings

From this

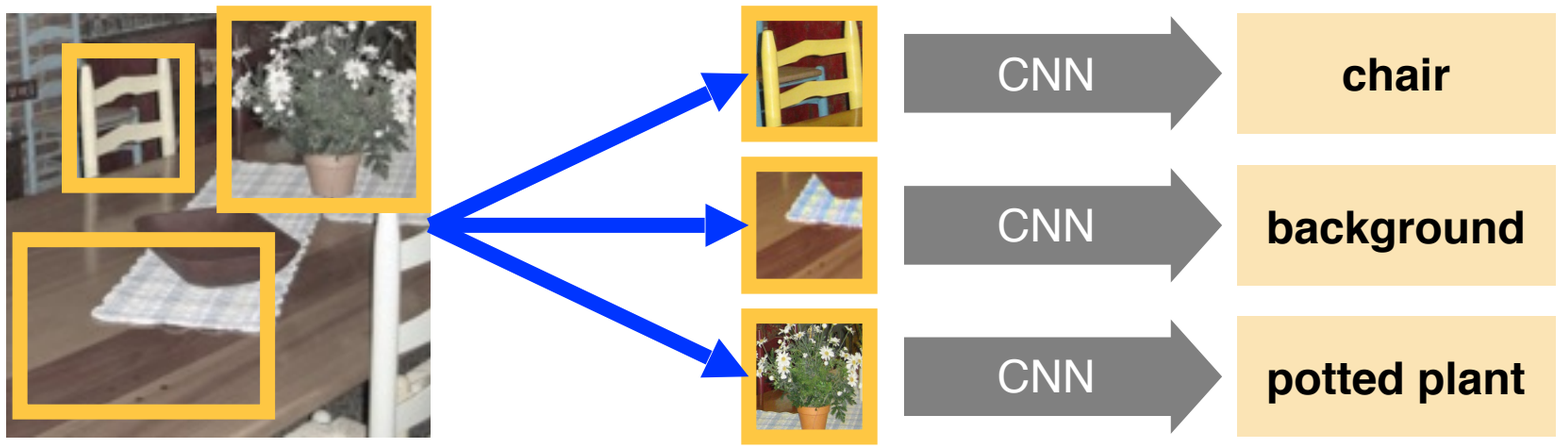
To this

There are
a cat, a person,
a chair

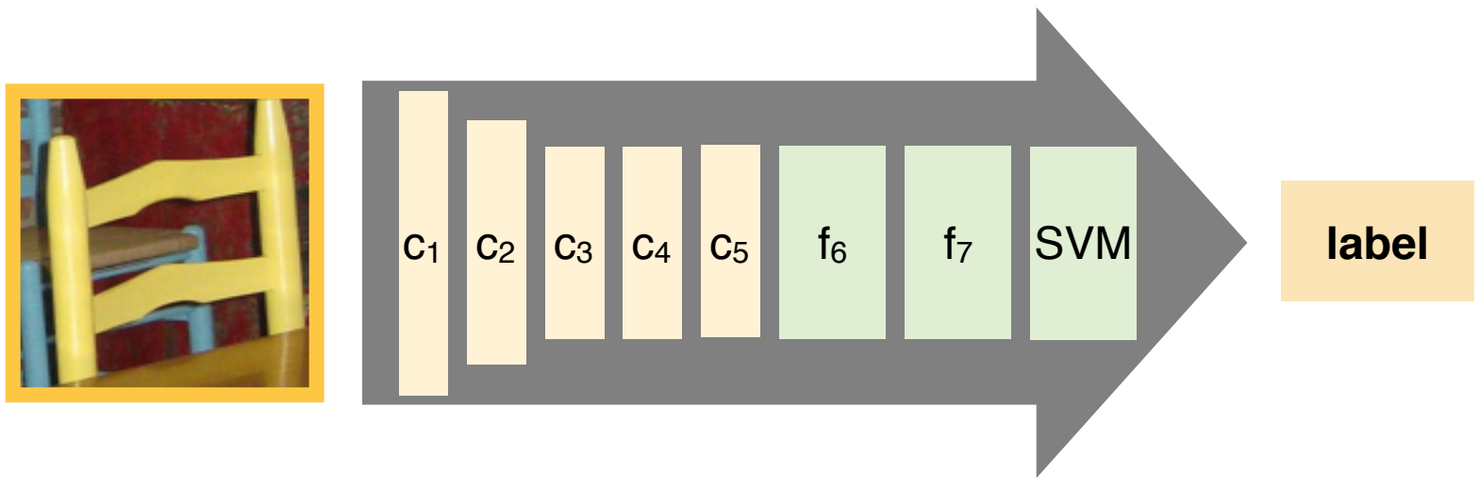
Detections with conv nets

Region-based Convolutional Neural Network (R-CNN)

Pros: simple and effective



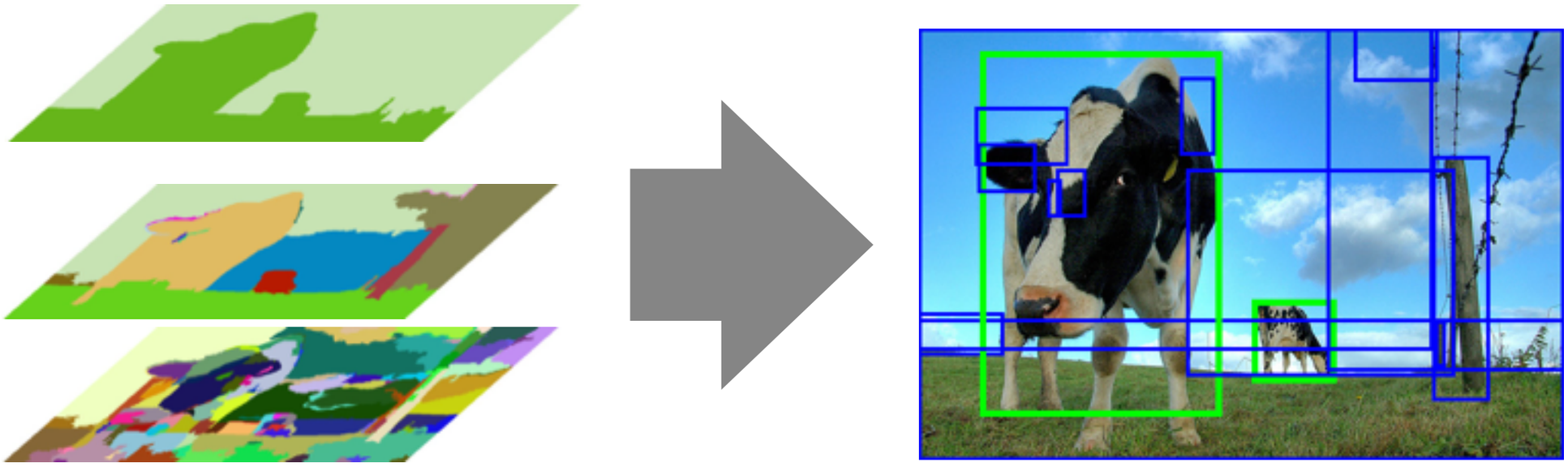
Cons: slow as the CNN is re-evaluated for each tested region



Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation
R. Girshick, J. Donahue, T. Darrell, J. Malik, CVPR 2014

Region proposals

Cut down the number of candidates



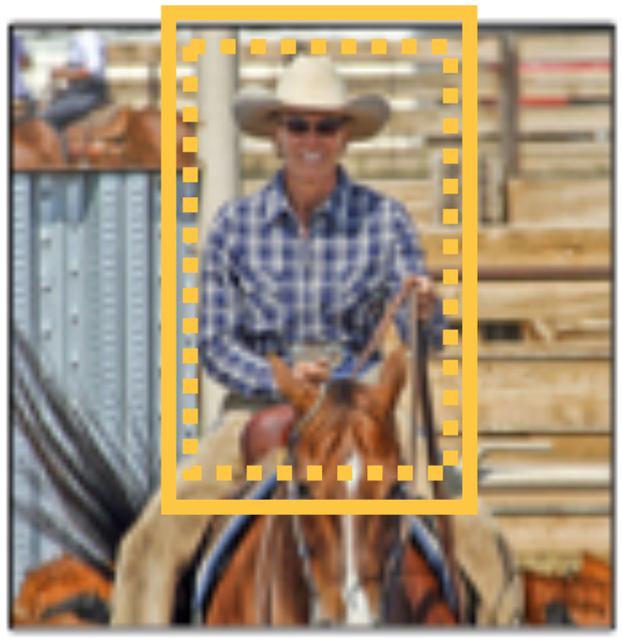
Proposal-method: Selective Search [van de Sande, Uijlings et al.]

- hierarchical segmentation
- each region generates a ROI
- ~ 2000 regions / image

From proposals to CNN features

Dilate, crop, reshape

Propose

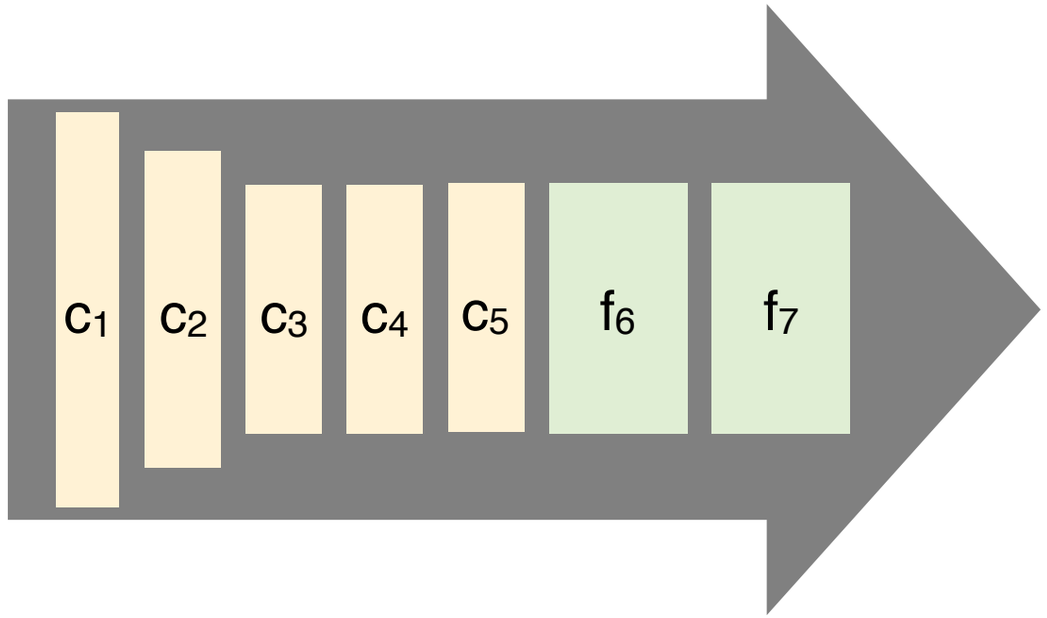


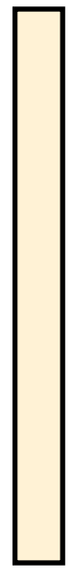
Dilate

Crop & scale
Anisotropic
227 x 227

From proposals to CNN features

Evaluate CNN





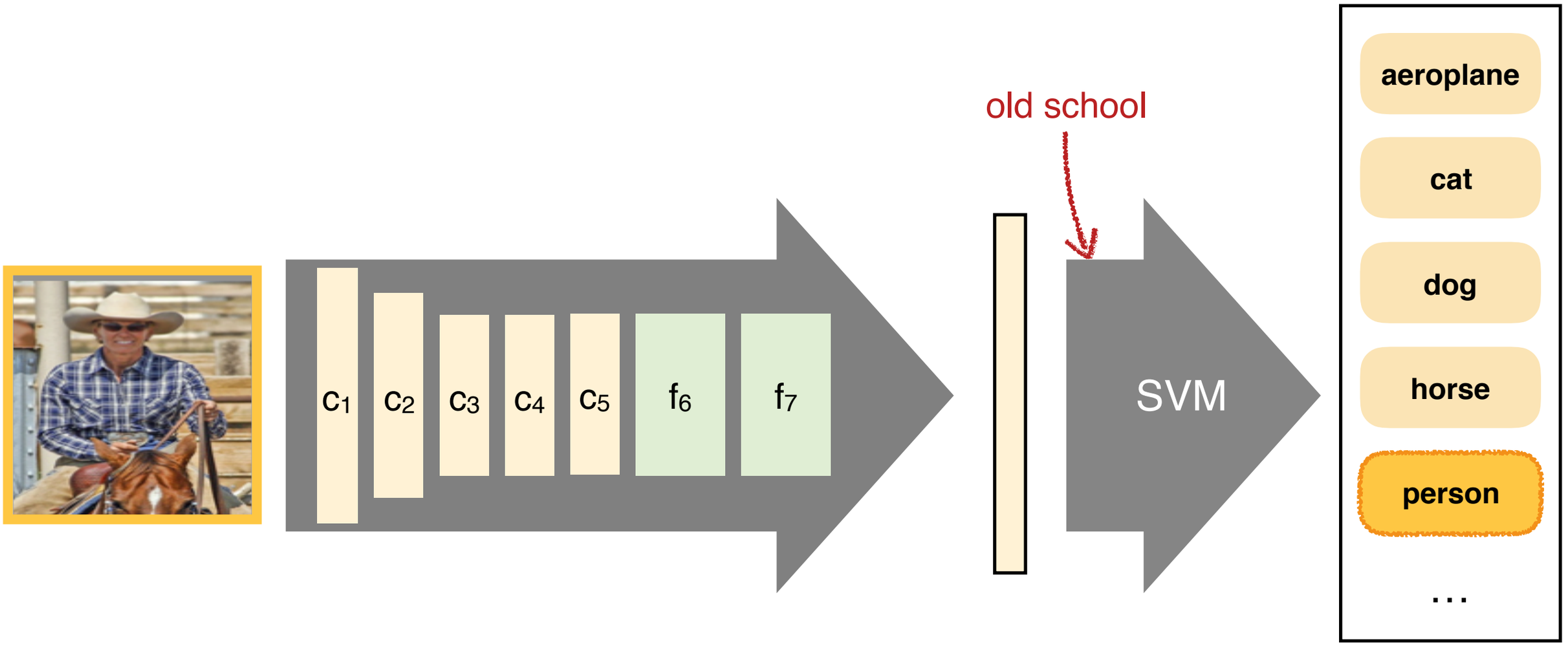
Scale
Anisotropic
227 x 227

CNN features
Up to FC-7
AlexNet

Feature vector
4096 D

Classification of a region

Run an SVM or similar on top



Scale
Anisotropic
227 x 227

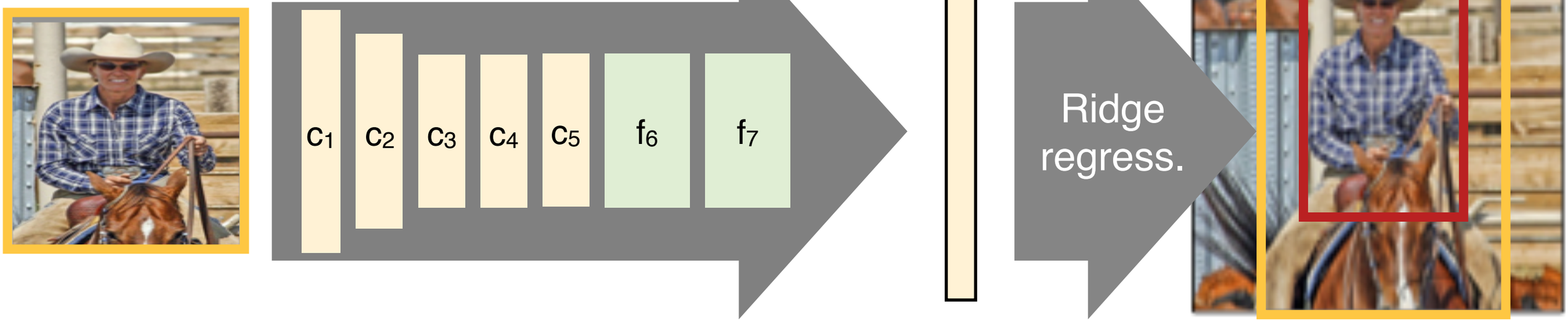
CNN features
Up to FC-7
AlexNet

Feature vector
4096 D

Label
One out of *N*

Region adjustment

Bounding-box regression



Scale
Anisotropic
227 x 227

CNN features
Up to FC-7
AlexNet

Feature vector
4096 D

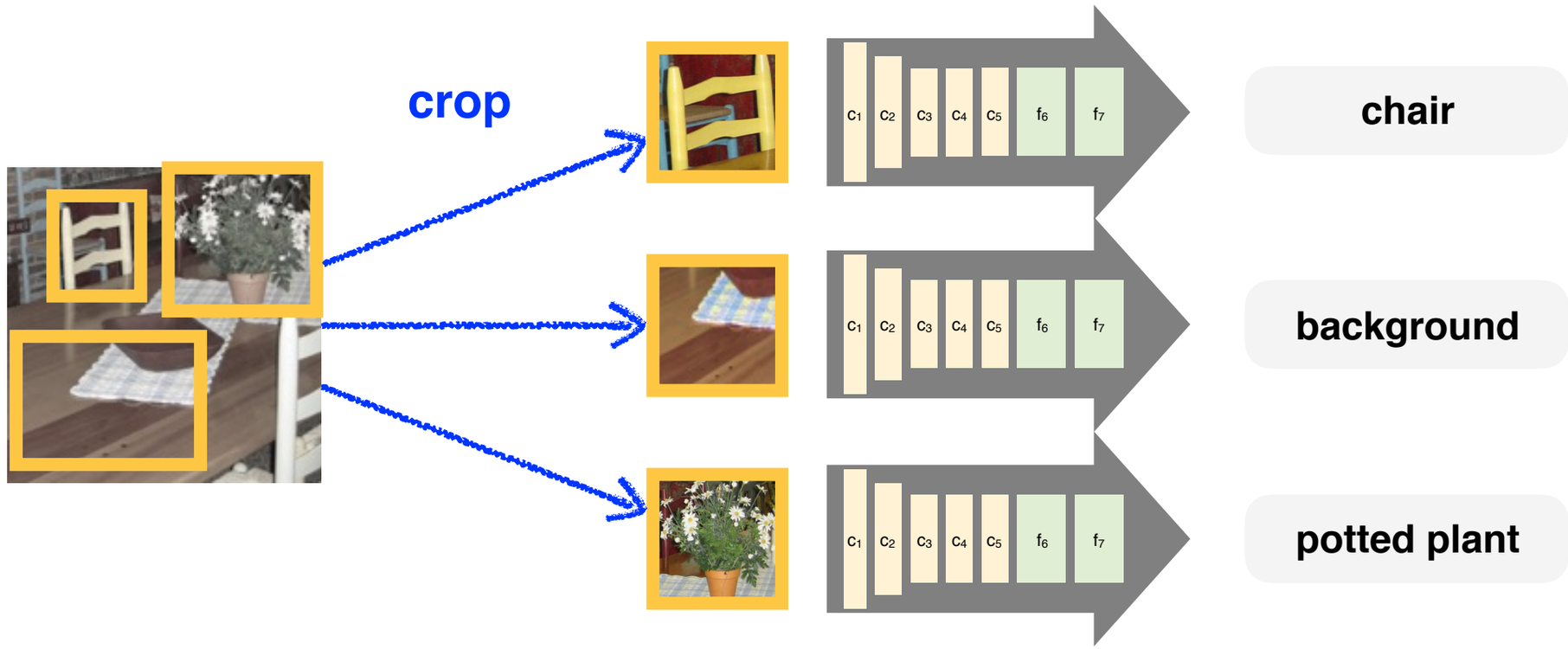
Box adjustment
dx1, dx2, dy1, dy2

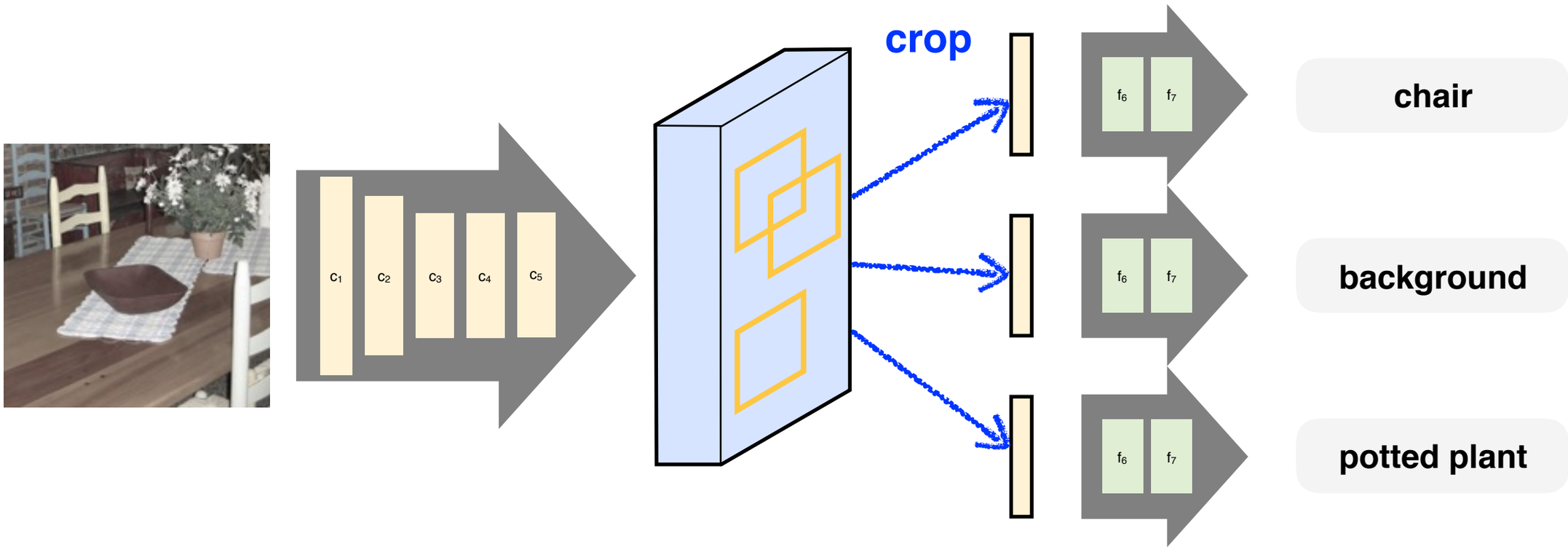
R-CNN results on PASCAL VOC

At the time of introduction (2013)

	VOC 2007	VOC 2010
DPM v5 (Girshick et al. 2011)	33.7%	29.6%
UVA sel. search (Uijlings et al. 2013)		35.1%
Regionlets (Wang et al. 2013)	41.7%	39.7%
SegDPM (Fidler et al. 2013)		40.4%
R-CNN (TorontoNet)	54.2%	50.2%
R-CNN (TorontoNet) + bbox regression	58.5%	53.7%
R-CNN (VGG-VD)	62.1%	
R-CNN (ONet) + bbox regression	66.0%	62.9%

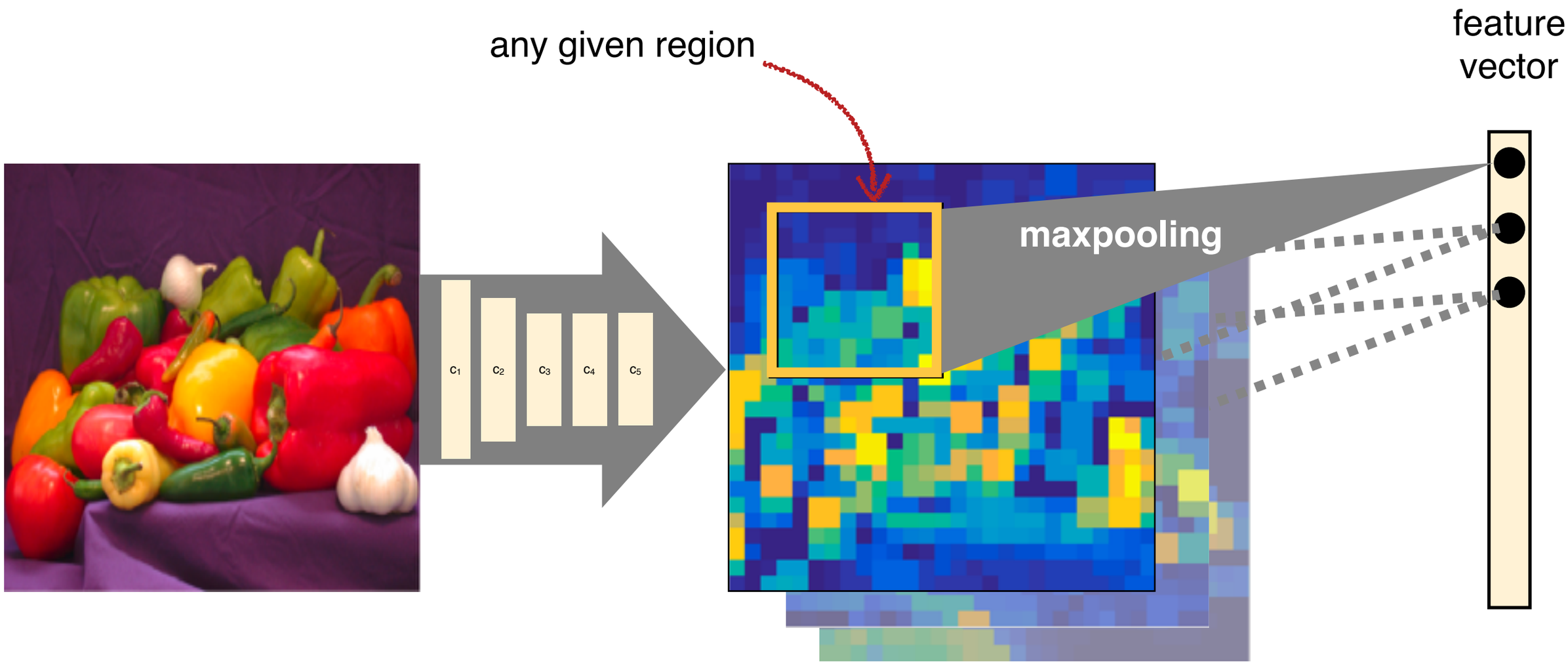
Accelerating R-CNN





The Spatial (Pyramid) Pooling layer

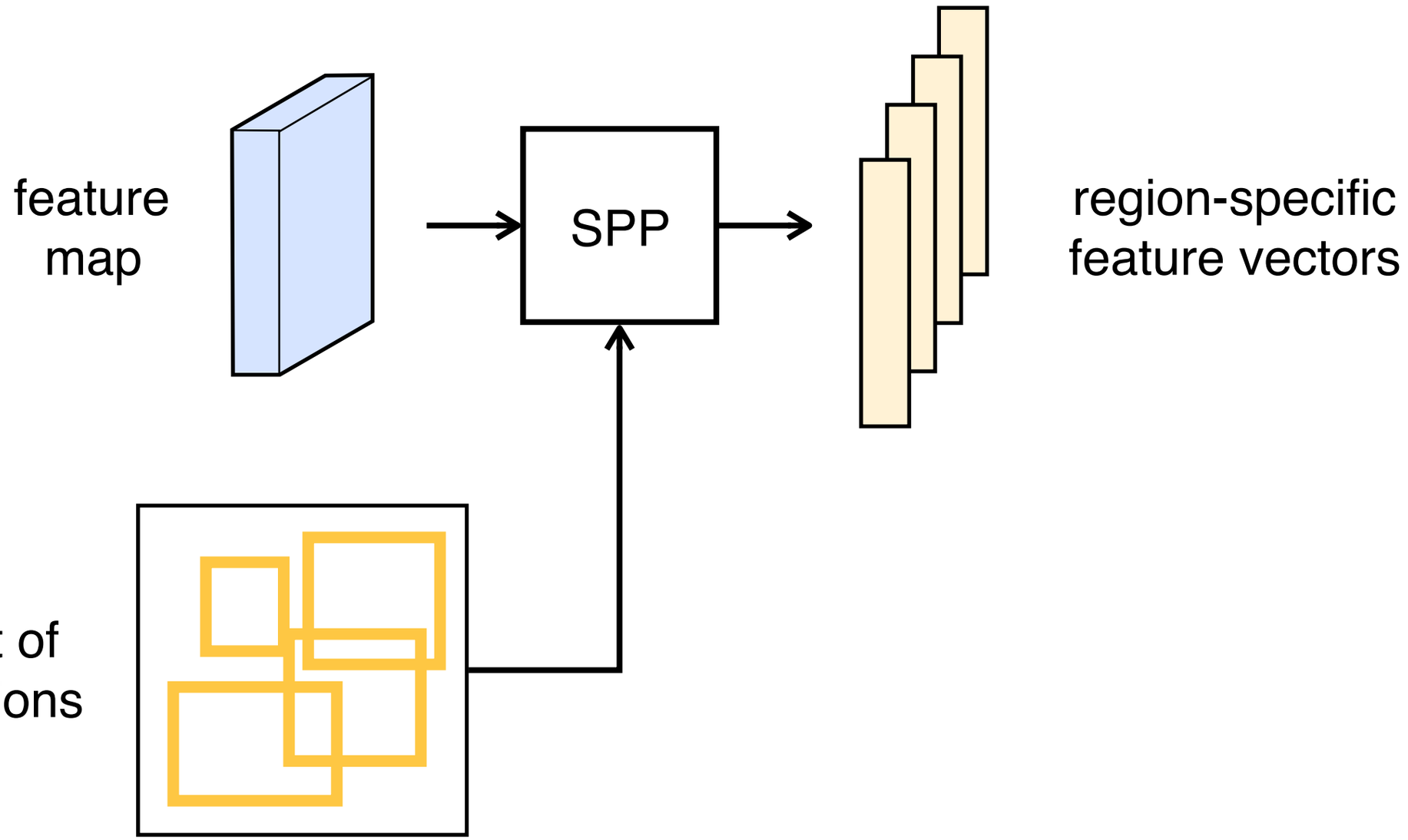
Max pooling in arbitrary regions



He, Zhang,Ren & Sun, “Spatial Pyramid Pooling (SPP) in Deep Convolutional Networks for Visual Recognition”, ECCV 2014

The Spatial (Pyramid) Pooling layer

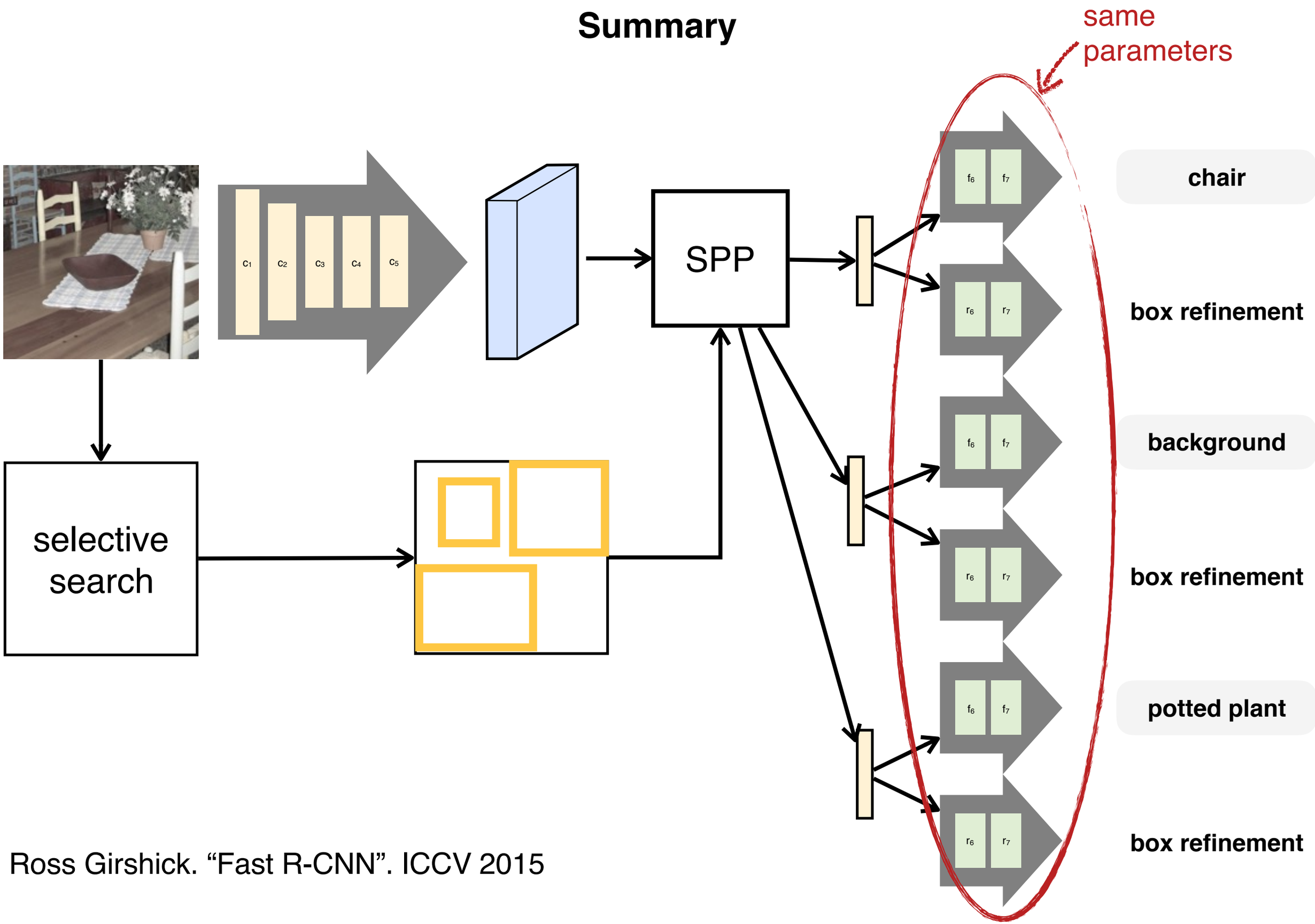
As a building block



He, Zhang,Ren & Sun, “Spatial Pyramid Pooling (SPP) in Deep Convolutional Networks for Visual Recognition”, ECCV 2014

Fast R-CNN

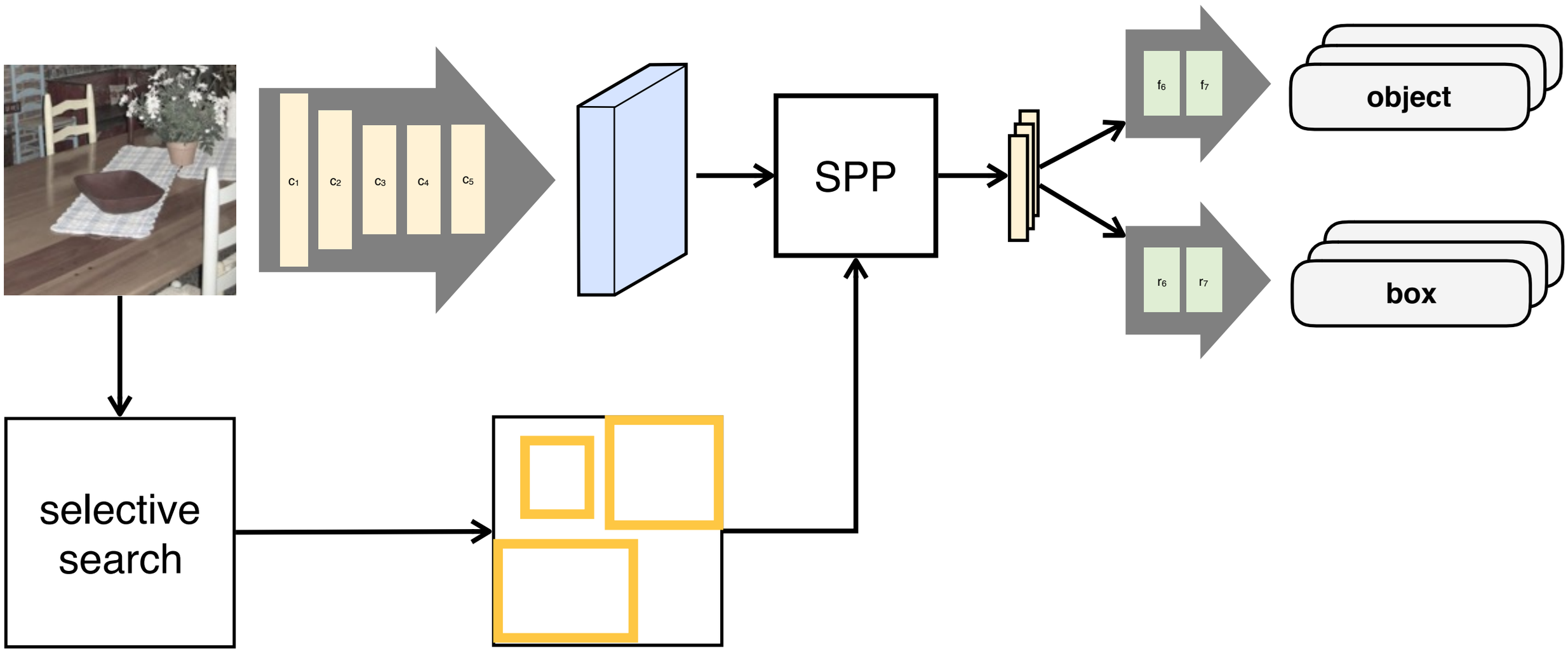
Summary



Ross Girshick. “Fast R-CNN”. ICCV 2015

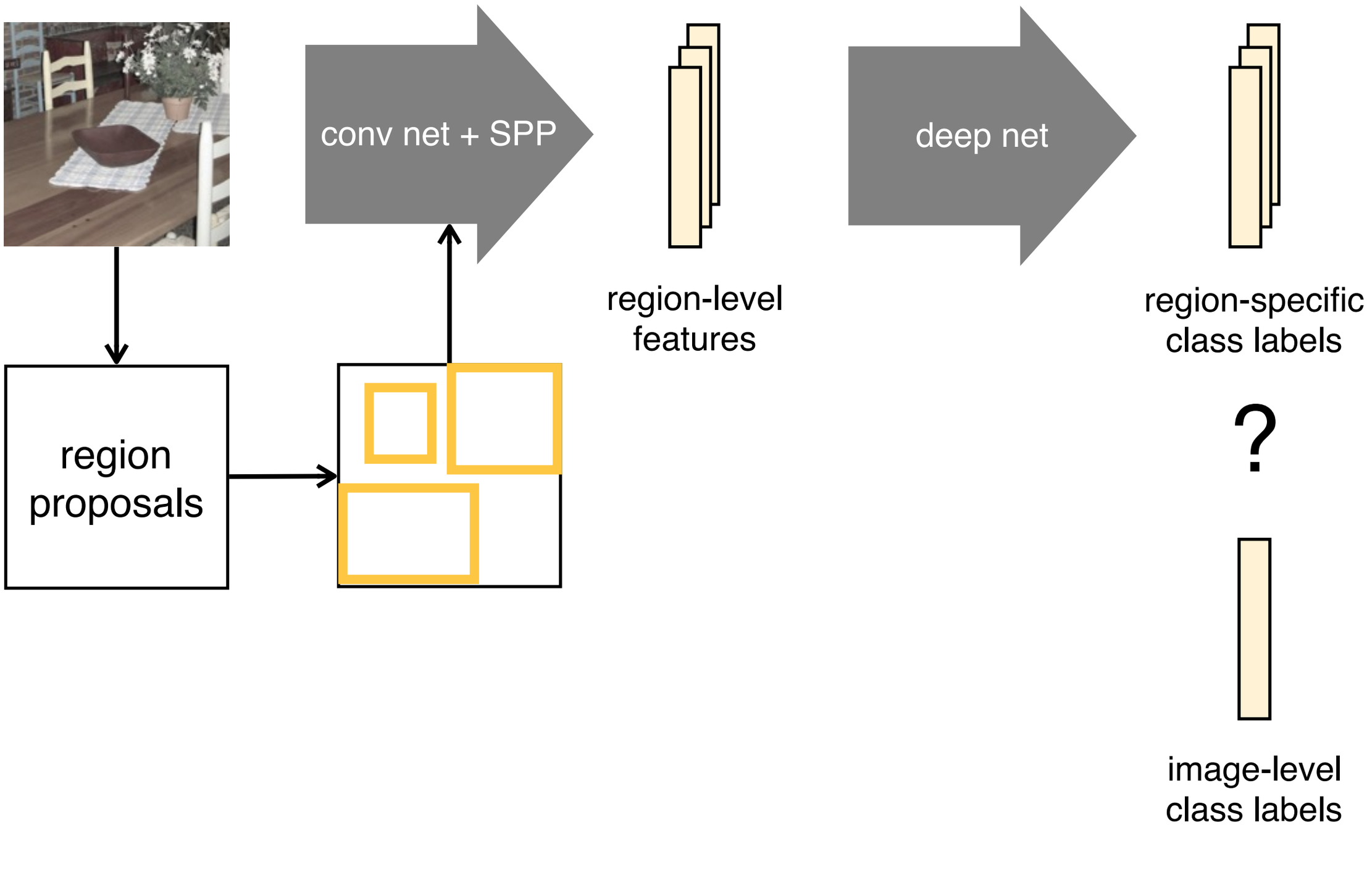
Fast R-CNN

Summary



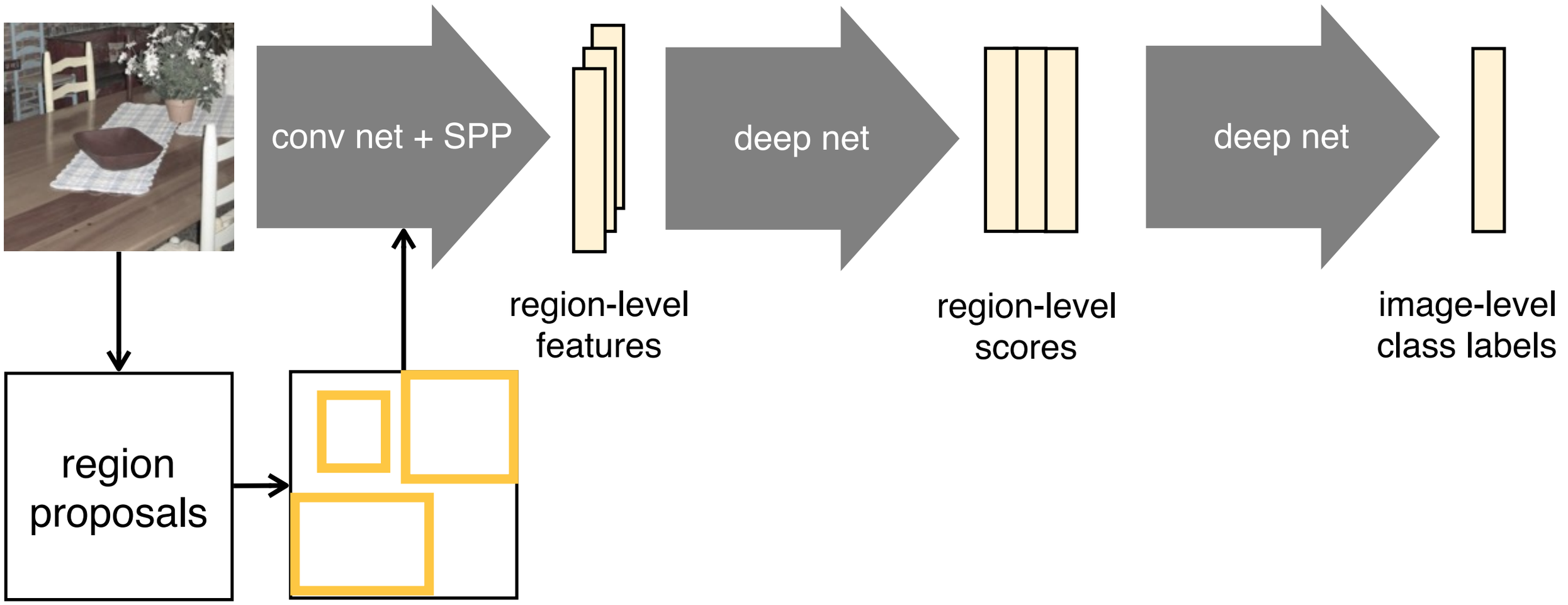
Object detection with deep networks

High-level view



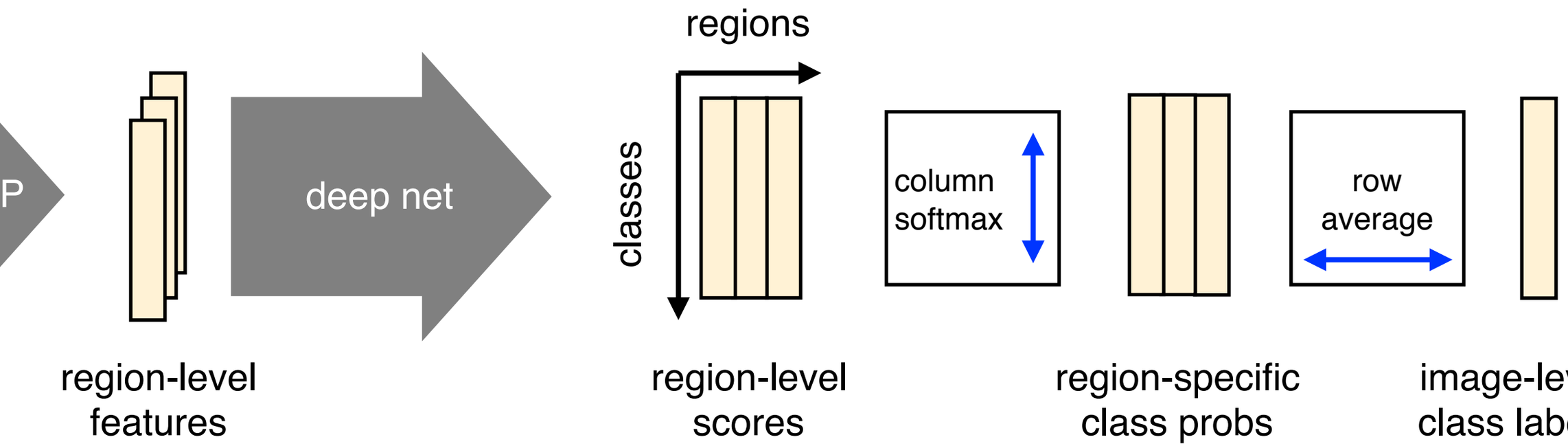
Towards weak supervision

Region scores to class labels



Towards weak supervision

Class labels = average of region labels

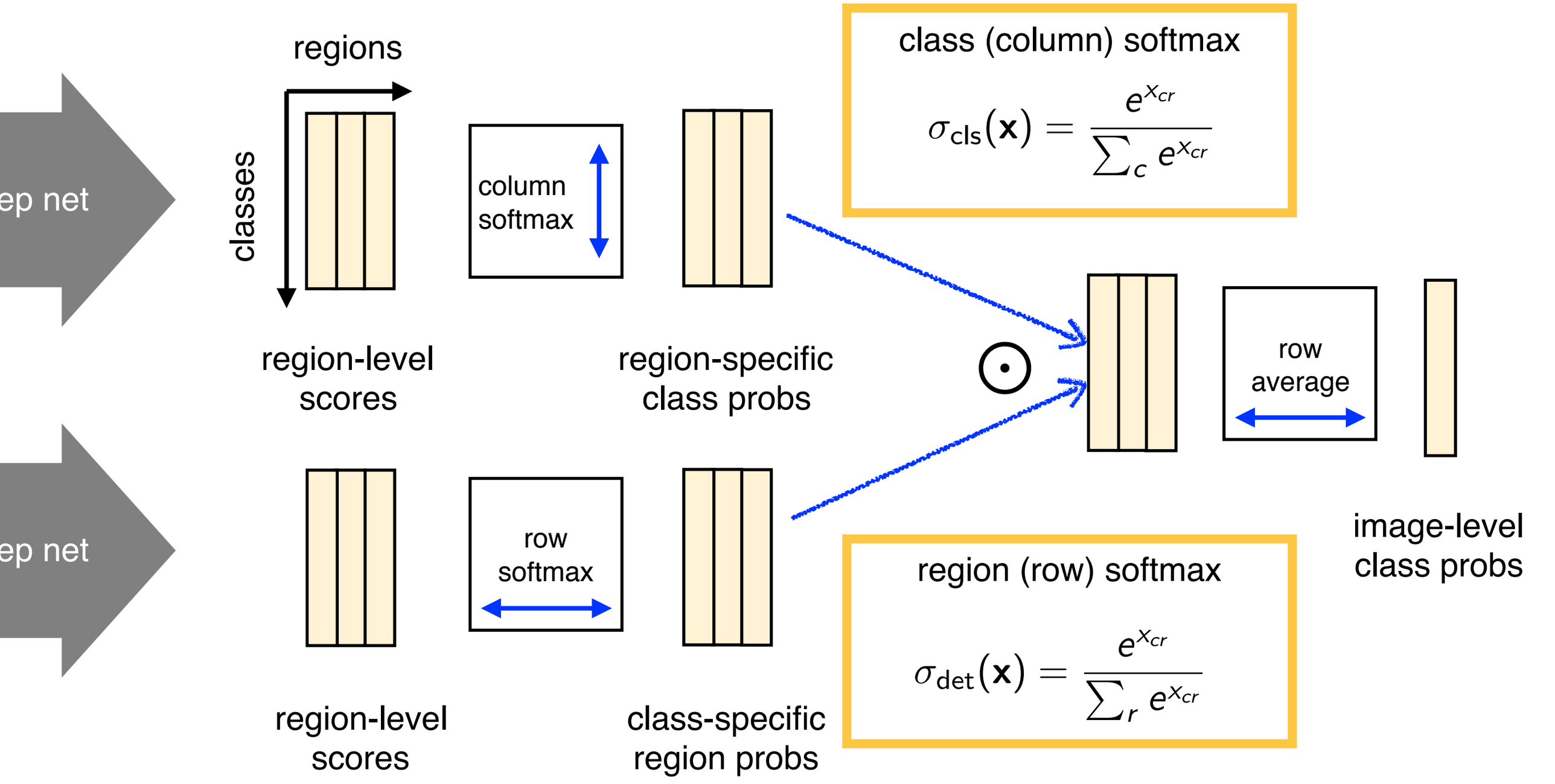


class (column) softmax

$$\sigma_{\text{cls}}(\mathbf{x}) = \frac{e^{x_{cr}}}{\sum_c e^{x_{cr}}}$$

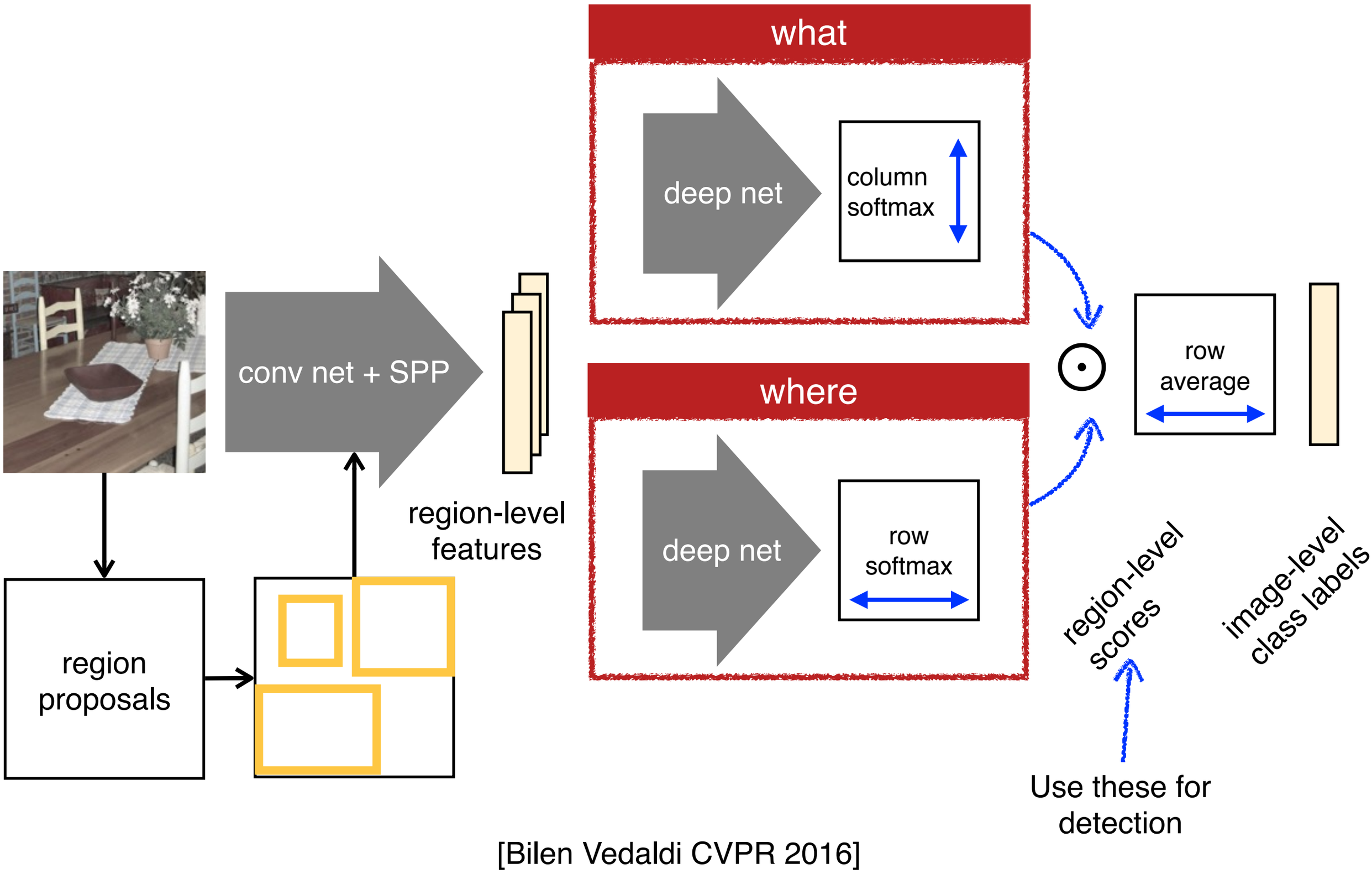
Towards weak supervision

Combine both class and region information



Two-streams weakly-supervised R-CNN

Overview



[Bilen Vedaldi CVPR 2016]

Results

Two streams vs state-of-the-art

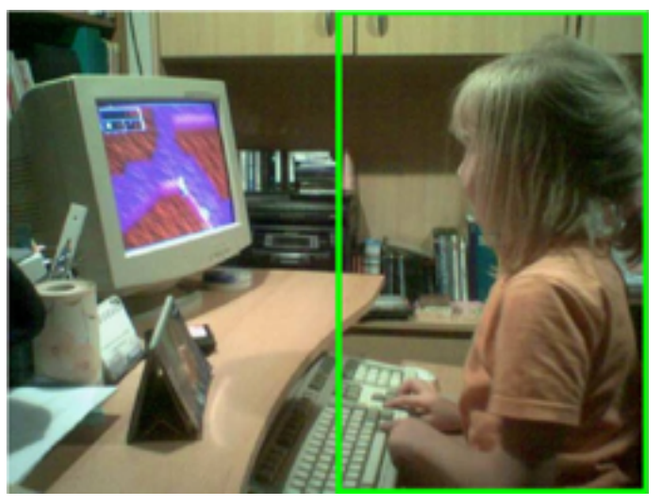
PASCAL VOC Detection (mAP)				
	Wang@ ECCV14	Bilen@ CVPR15	Cinbis@ PAMI16	Two Streams ++
PASCAL VOC07	30.2	27.7	31.6	**39.3**
PASCAL VOC10	27.4	-	-	**36.2**

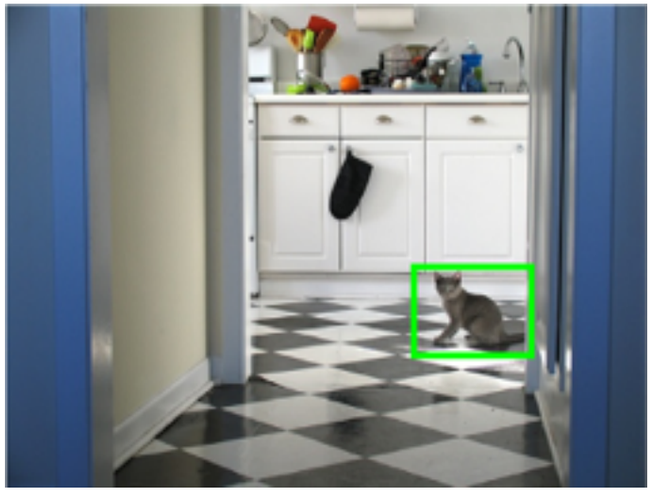
Two streams vs single stream

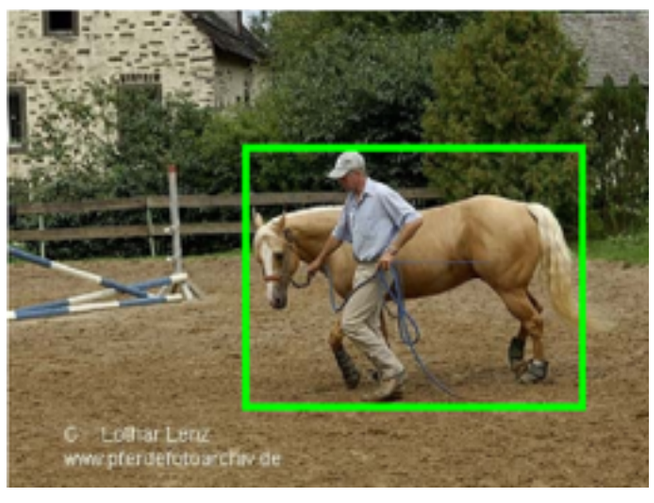
PASCAL VOC Detection (mAP)			
	Single Stream	Two Streams	Two Streams ++
PASCAL VOC07	21.6	30.9	39.3

Results

Good results







Failure modes

Understanding visual representations

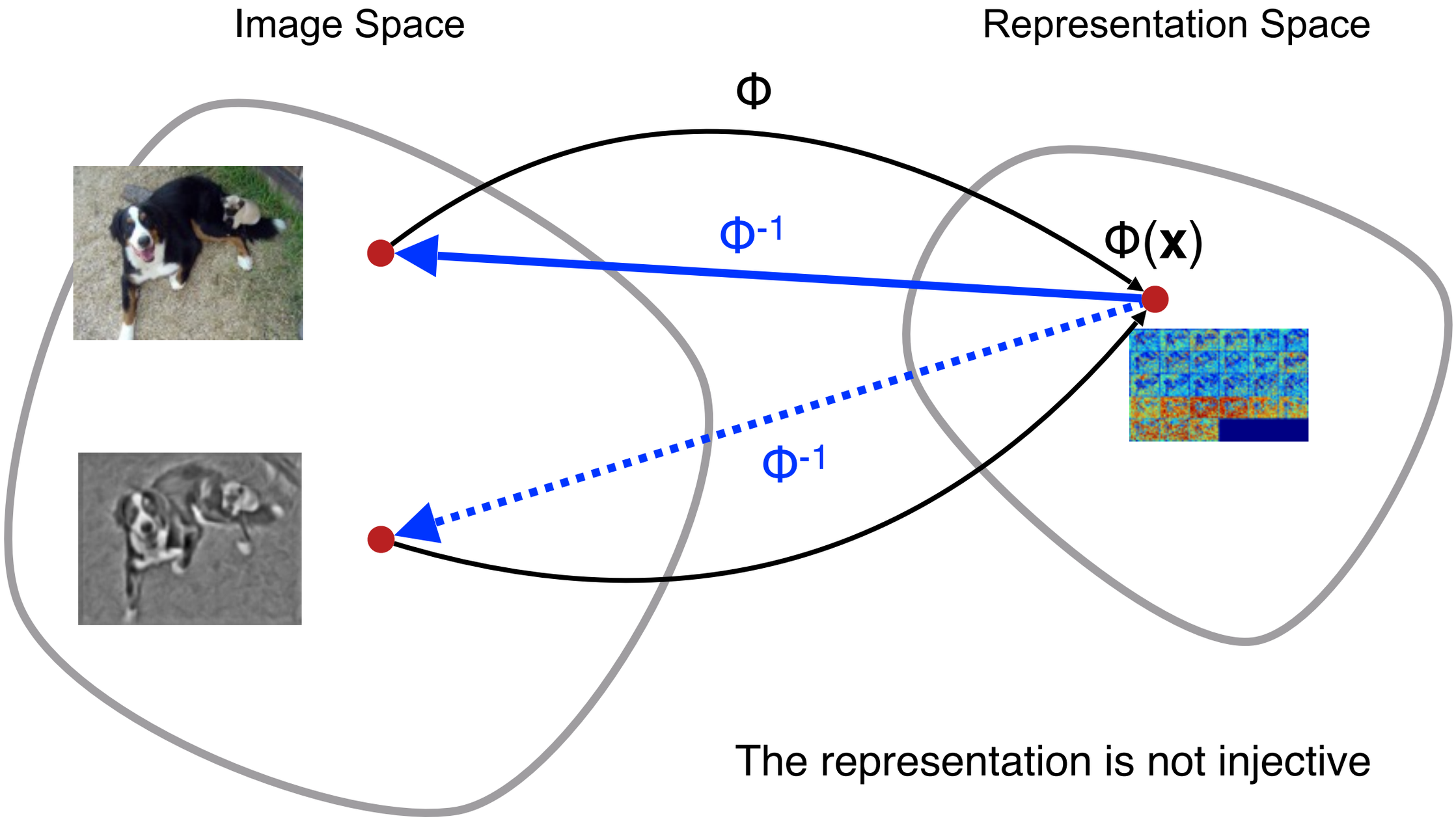
Intro

Visualizing representations

Backpropagation networks and “deconvolution”

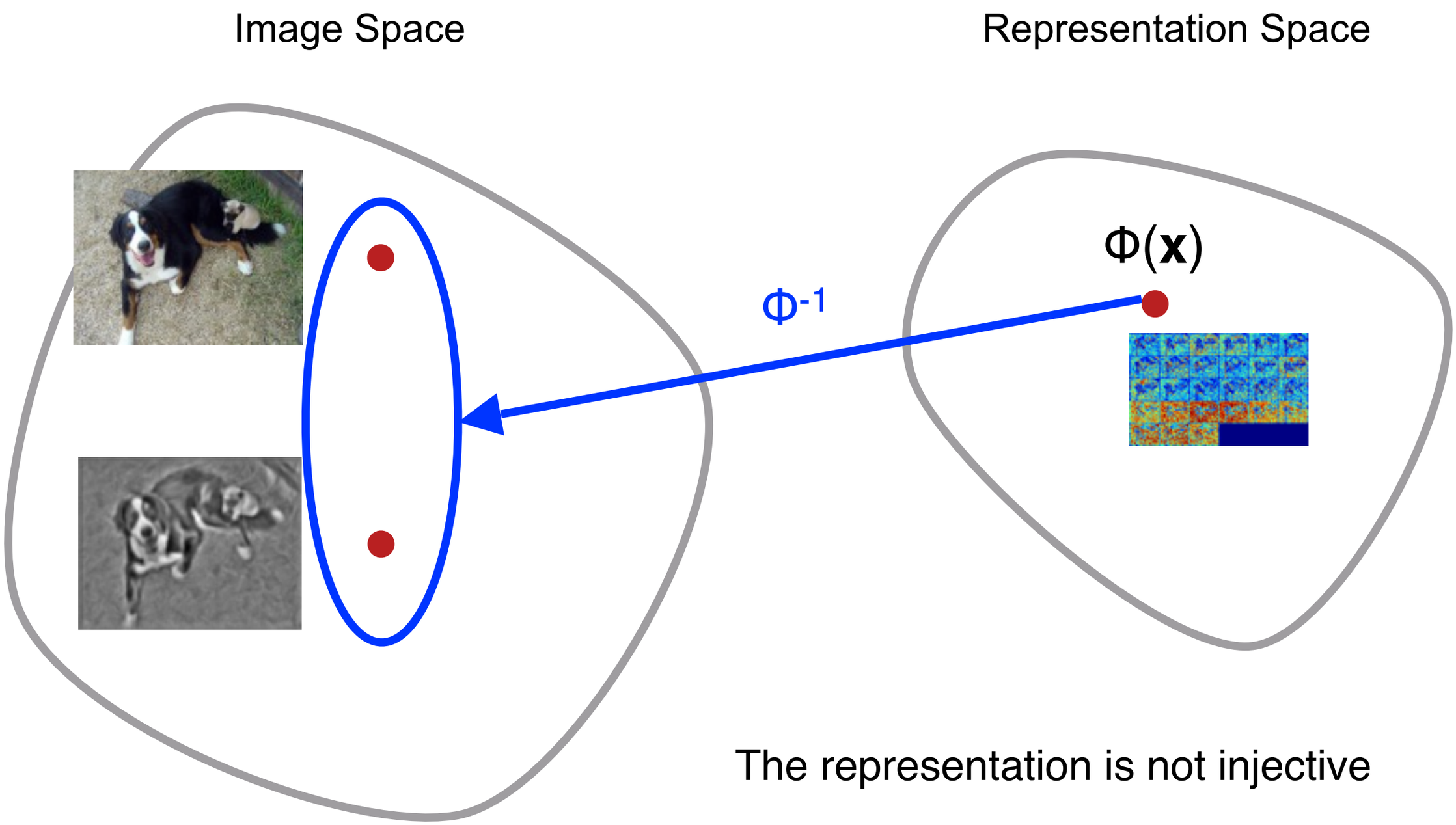
Representations: equivalence & transformations

Visualization: Pre-Image



The representation is not injective

Visualization: Pre-Image



The representation is not injective

The reconstruction ambiguity **provides useful information about the representation**

Finding a Pre-Image

A simple yet general and effective method

$$\min_{\mathbf{x}} \|\Phi(\mathbf{x}) - \Phi_0\|_2^2$$

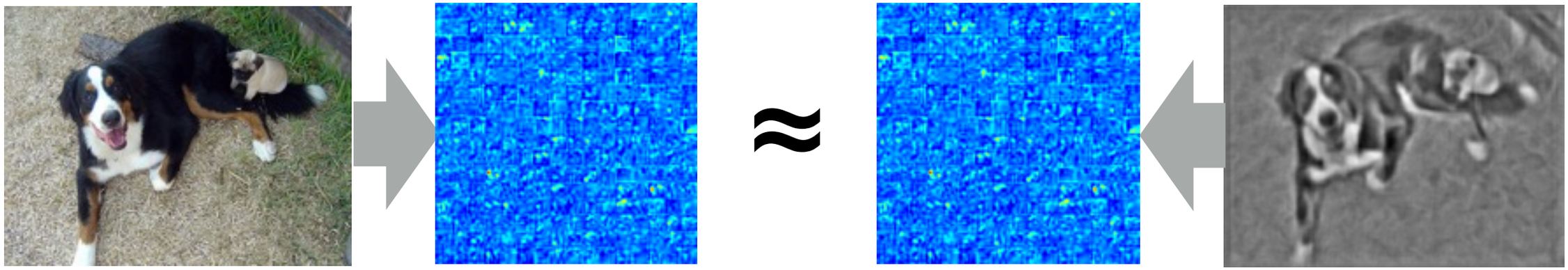


Image | Representation | Reconstruction | Pre-Image

Start from **random noise**

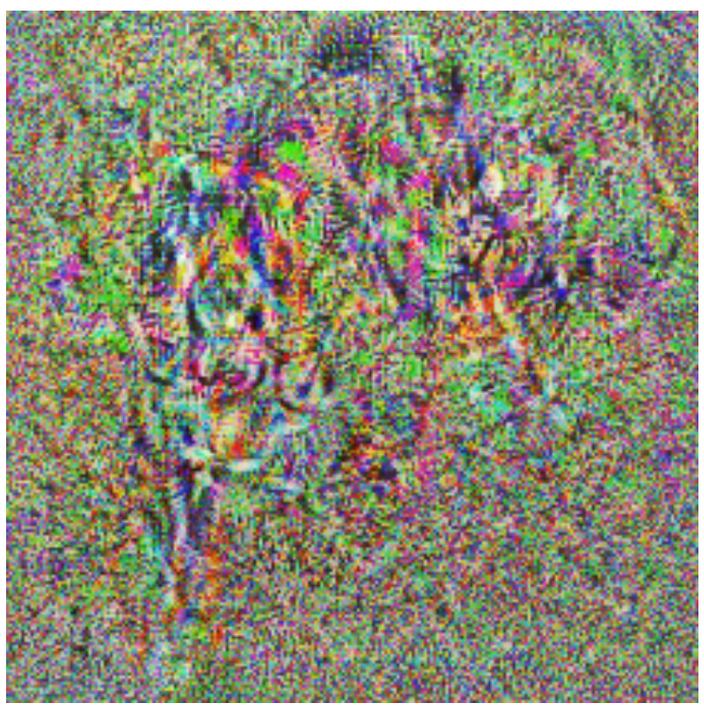
Optimize using stochastic **gradient descent**

Finding a Pre-Image

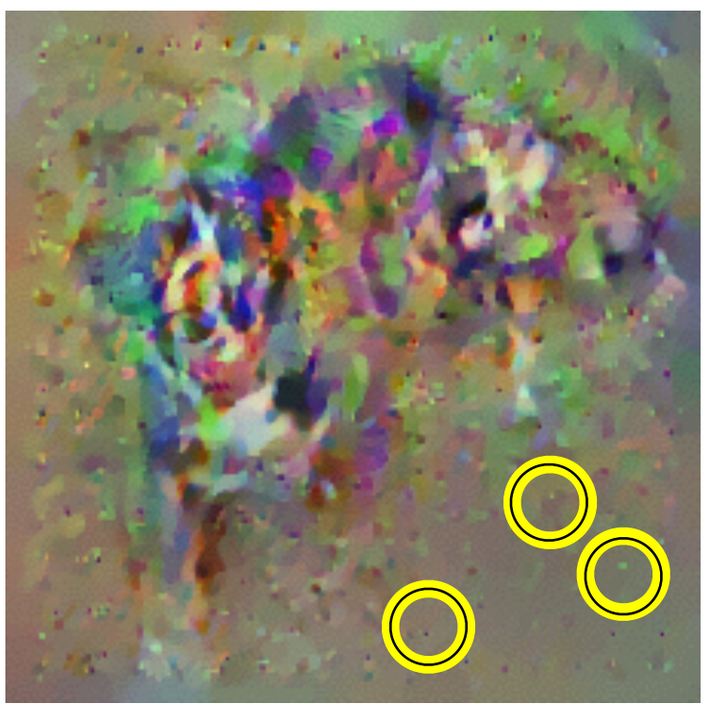
A simple yet general and effective method

$$\min_{\mathbf{x}} \|\Phi(\mathbf{x}) - \Phi_0\|_2^2$$

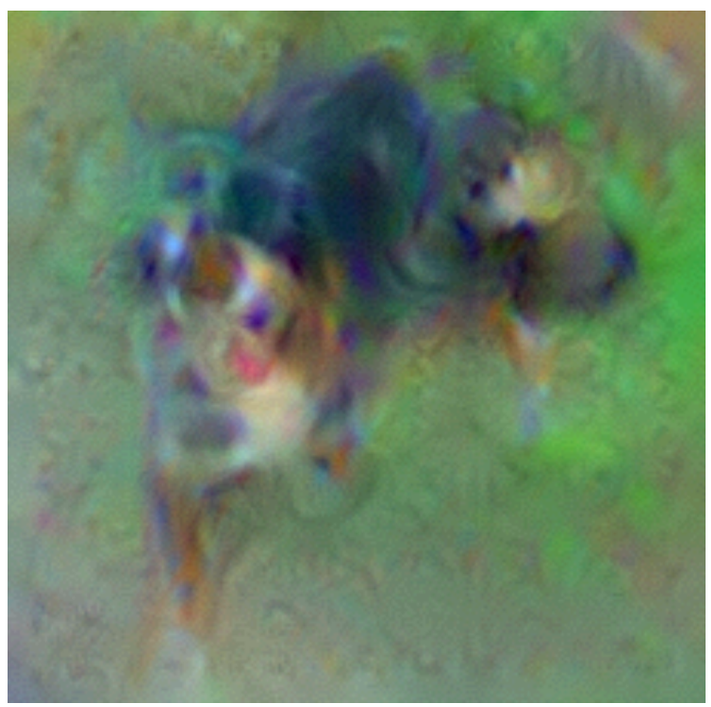
No prior



TV-norm β = 1



TV-norm β = 2



Related Work

Analysis tools

Visualizing higher-layer features of a deep network
Ethan et al. 2009
[intermediate features]

Deep inside convolutional networks
Simonyan et al. 2014
[deepest features, aka “deep dreams”]

DeConvNets
Zeiler et al. In ECCV, 2014
[intermediate features]

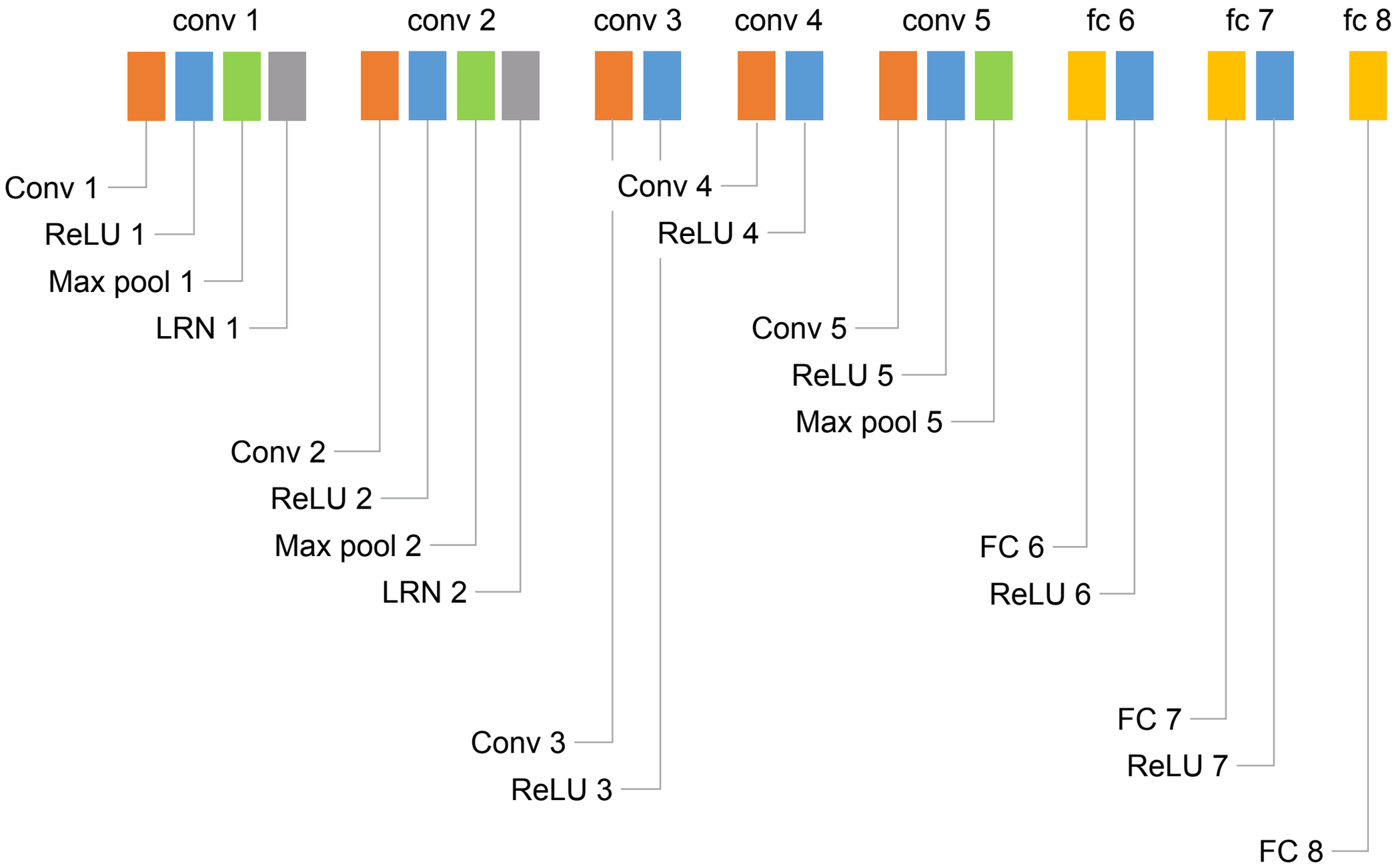
Understanding neural networks through deep visualisation
Yosinksi et al. 2015
[intermediate features]

Artistic tools

Google’s “inceptionsm”
Mordvintsev et al. 2015

Style synthesis and transfer
Gatys et al. 2015

Inversion



AlexNet
[Krizhevsky et al. 2012]

Inversion

Original Image

Inversion

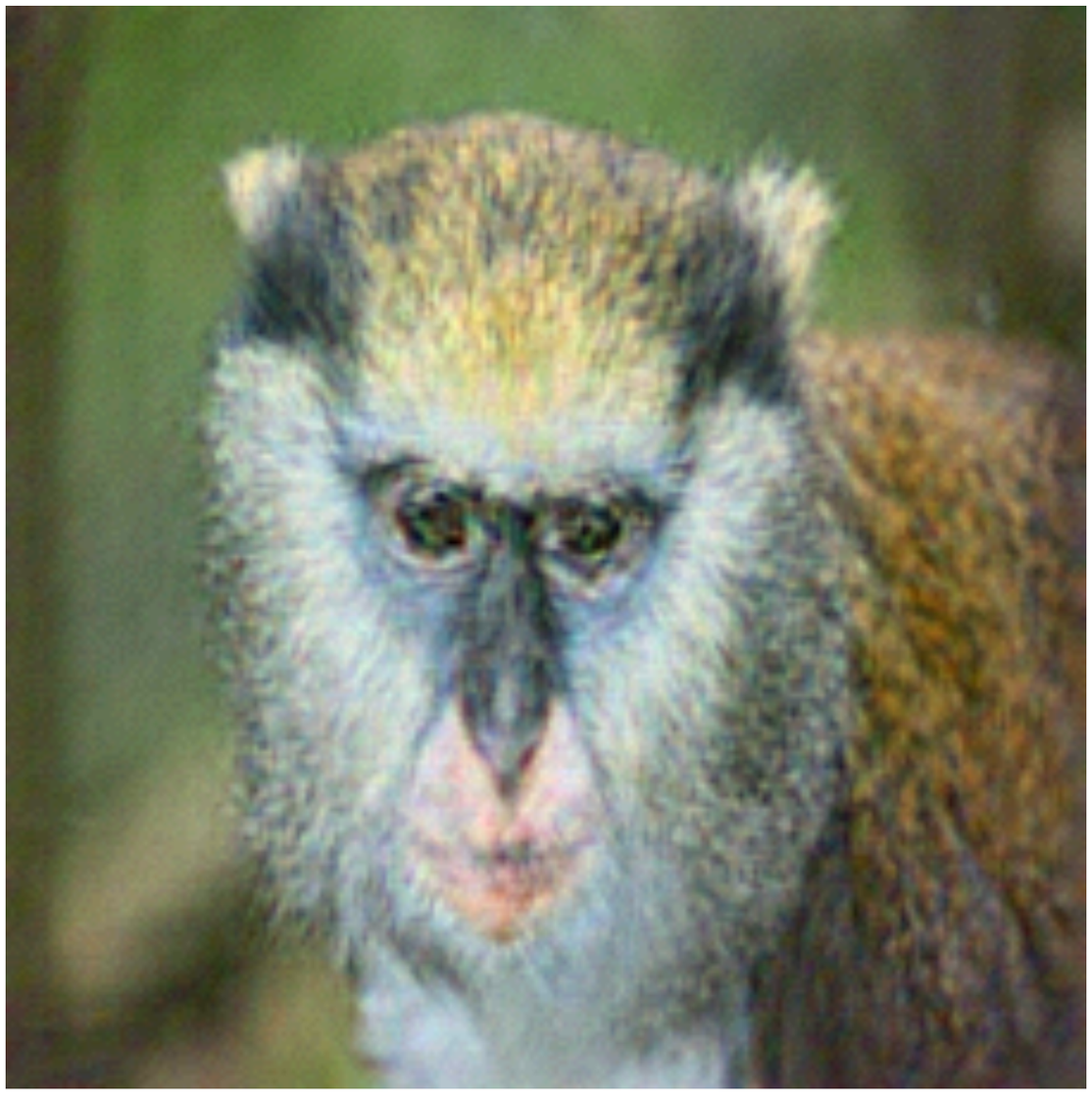
Original Image

Inversion

Original Image

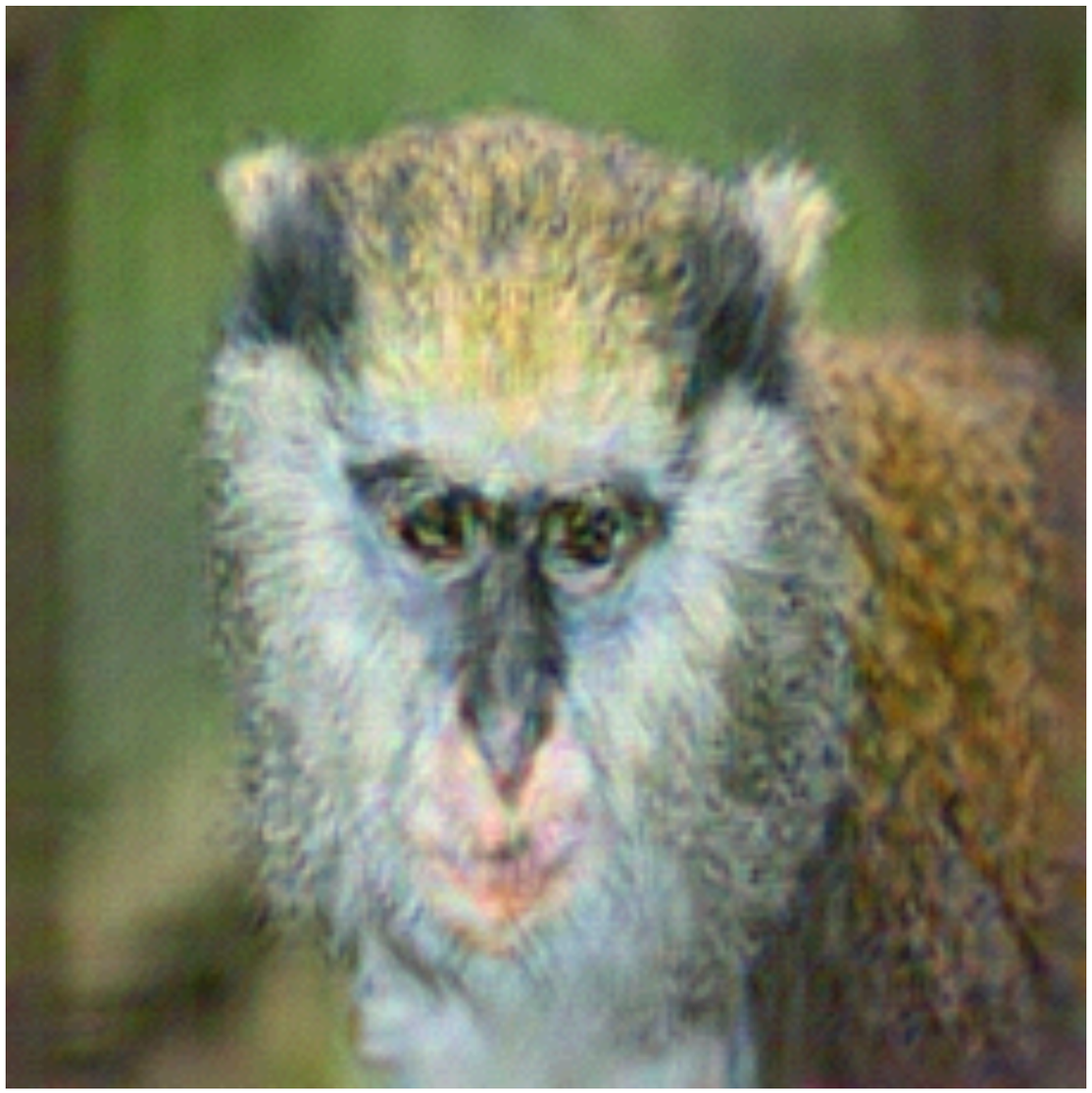
Inversion

Original Image



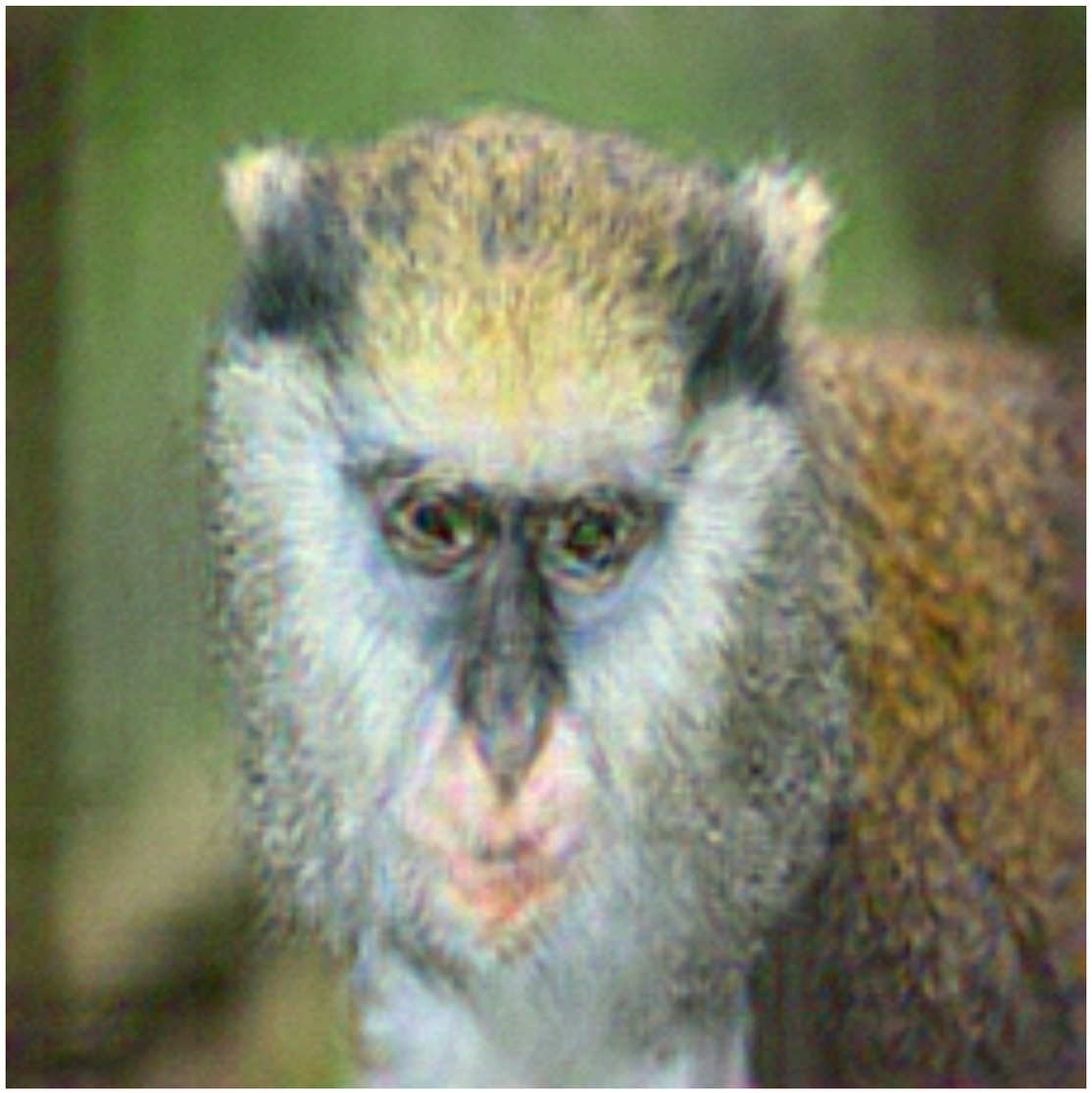
Inversion

Original Image



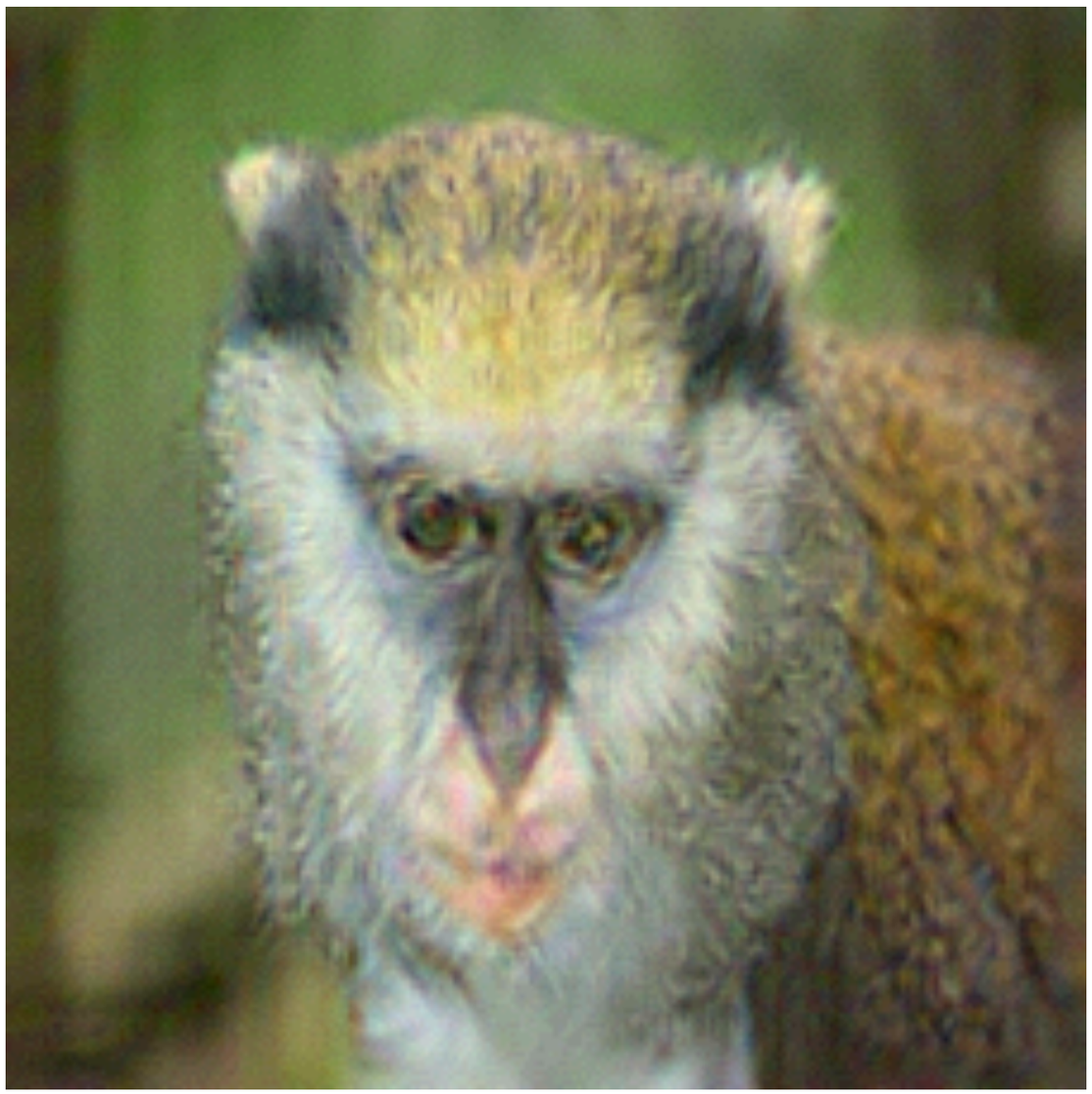
Inversion

Original Image



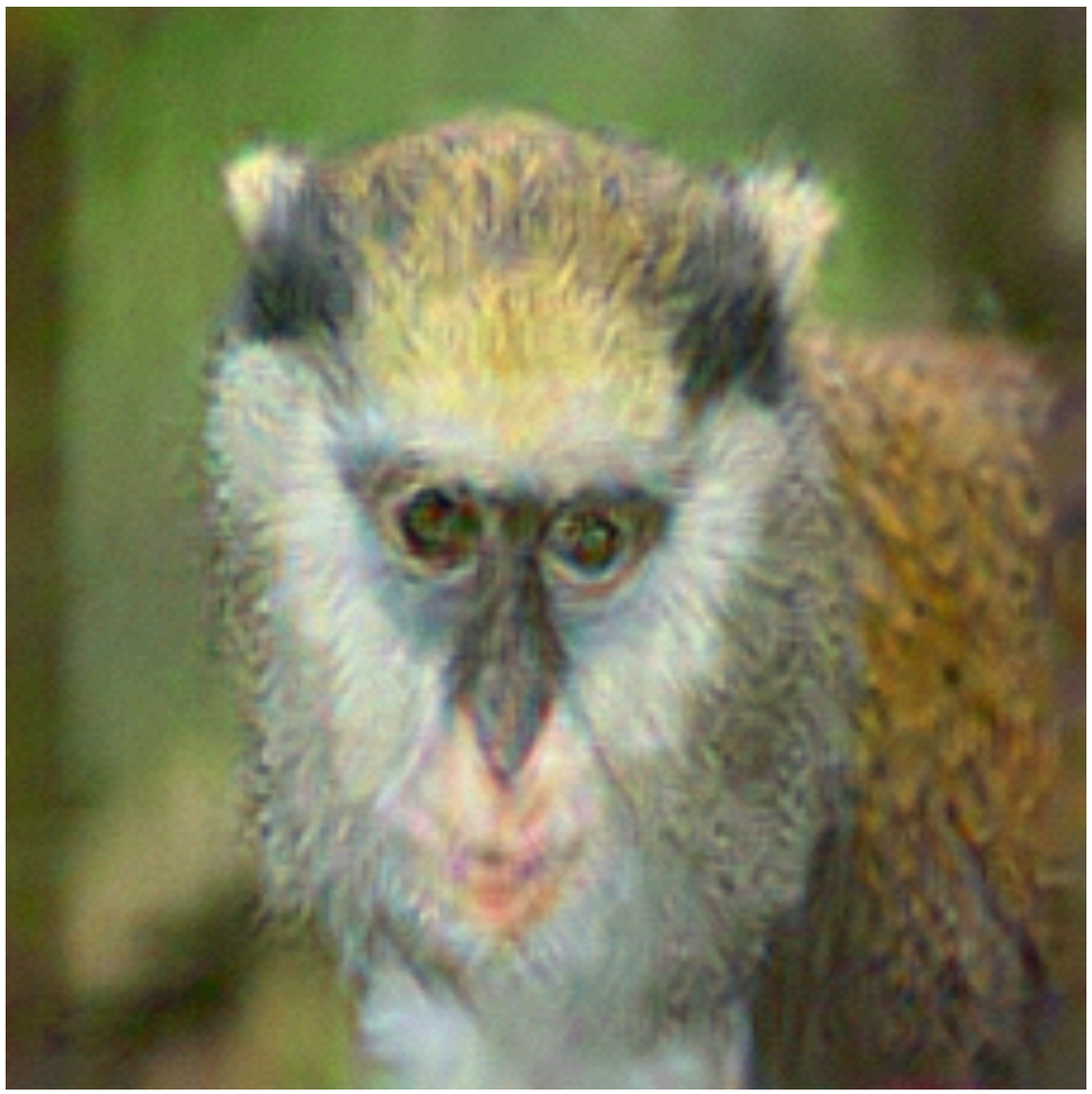
Inversion

Original Image



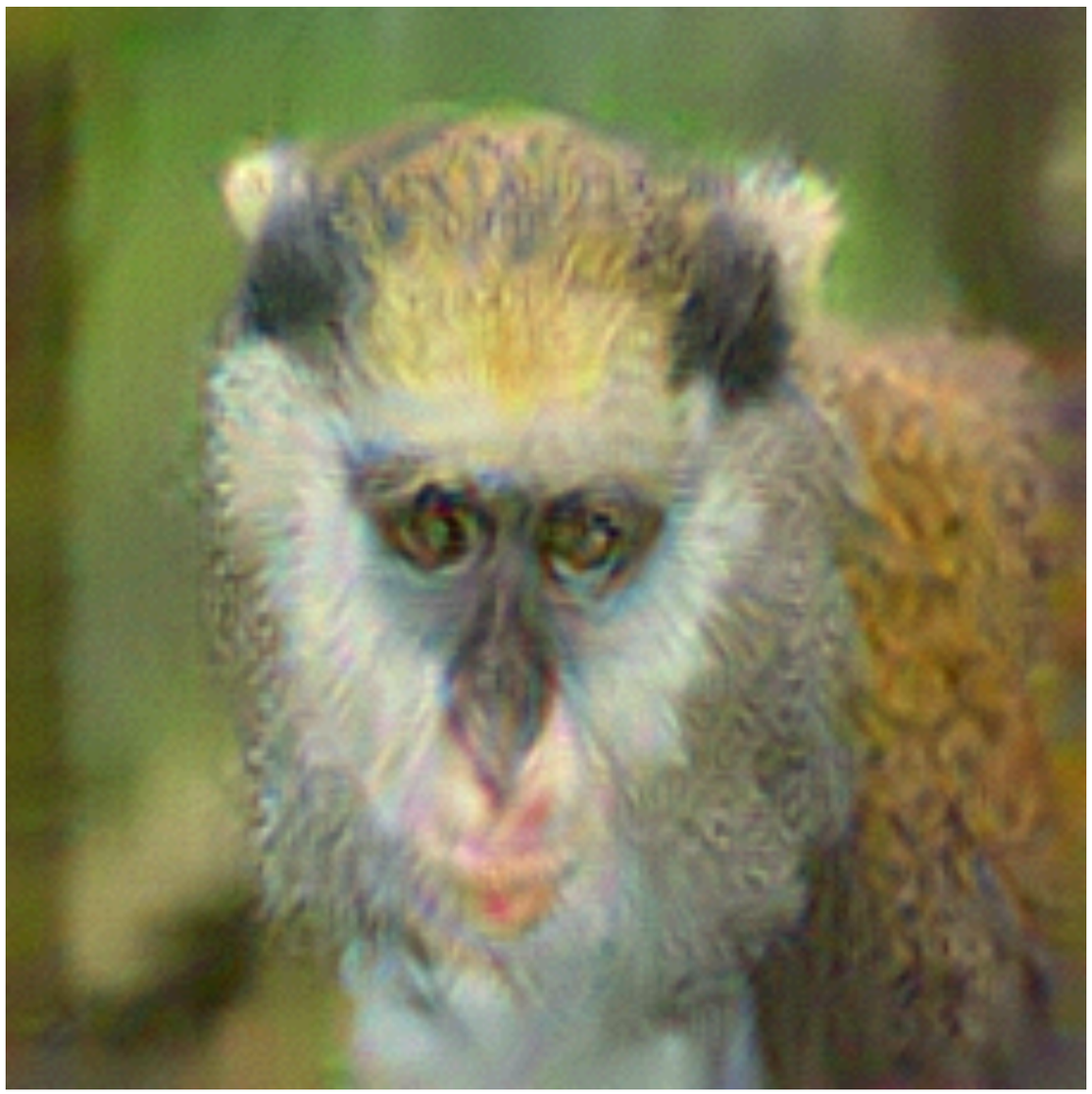
Inversion

Original Image



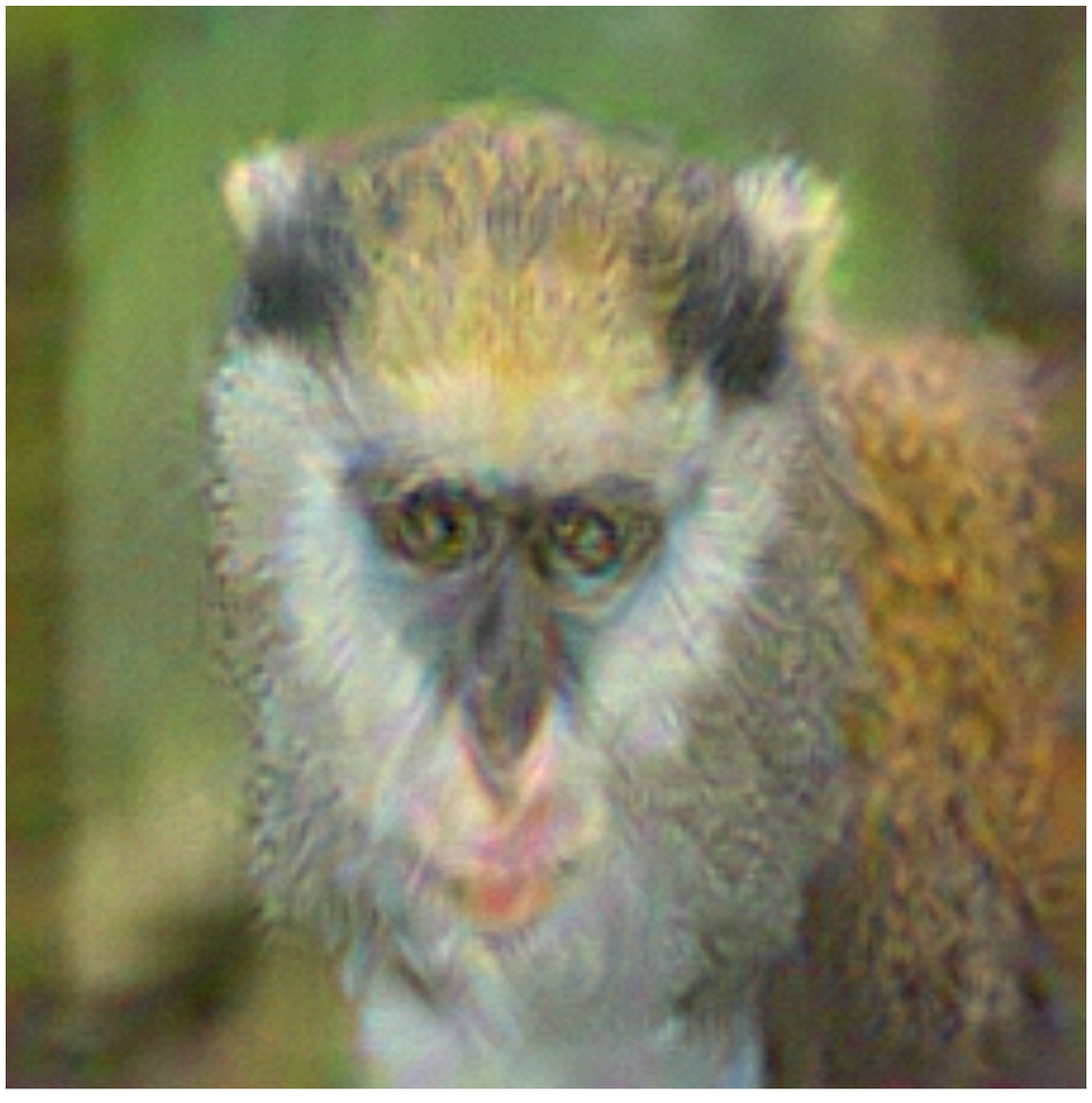
Inversion

Original Image



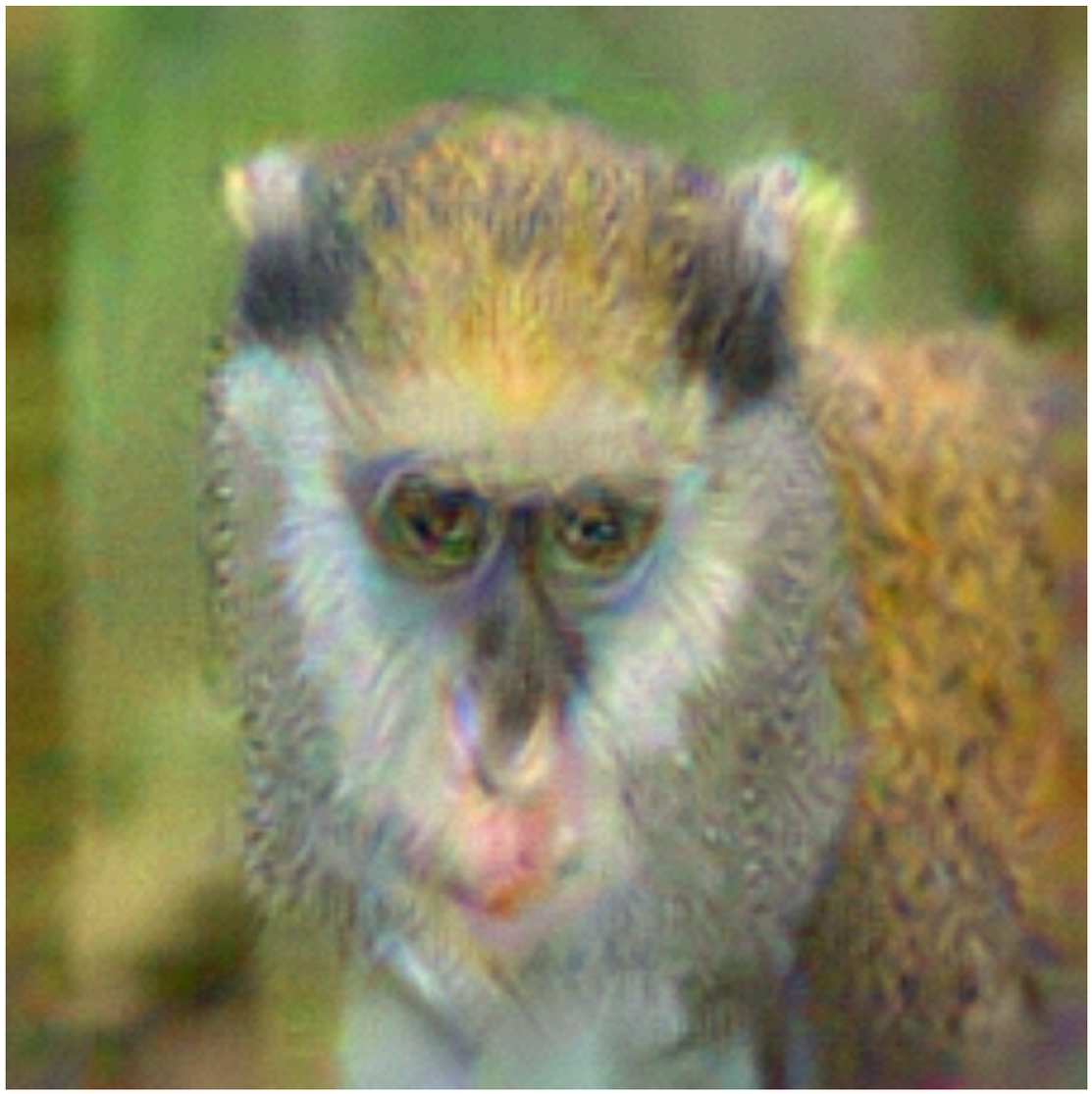
Inversion

Original Image



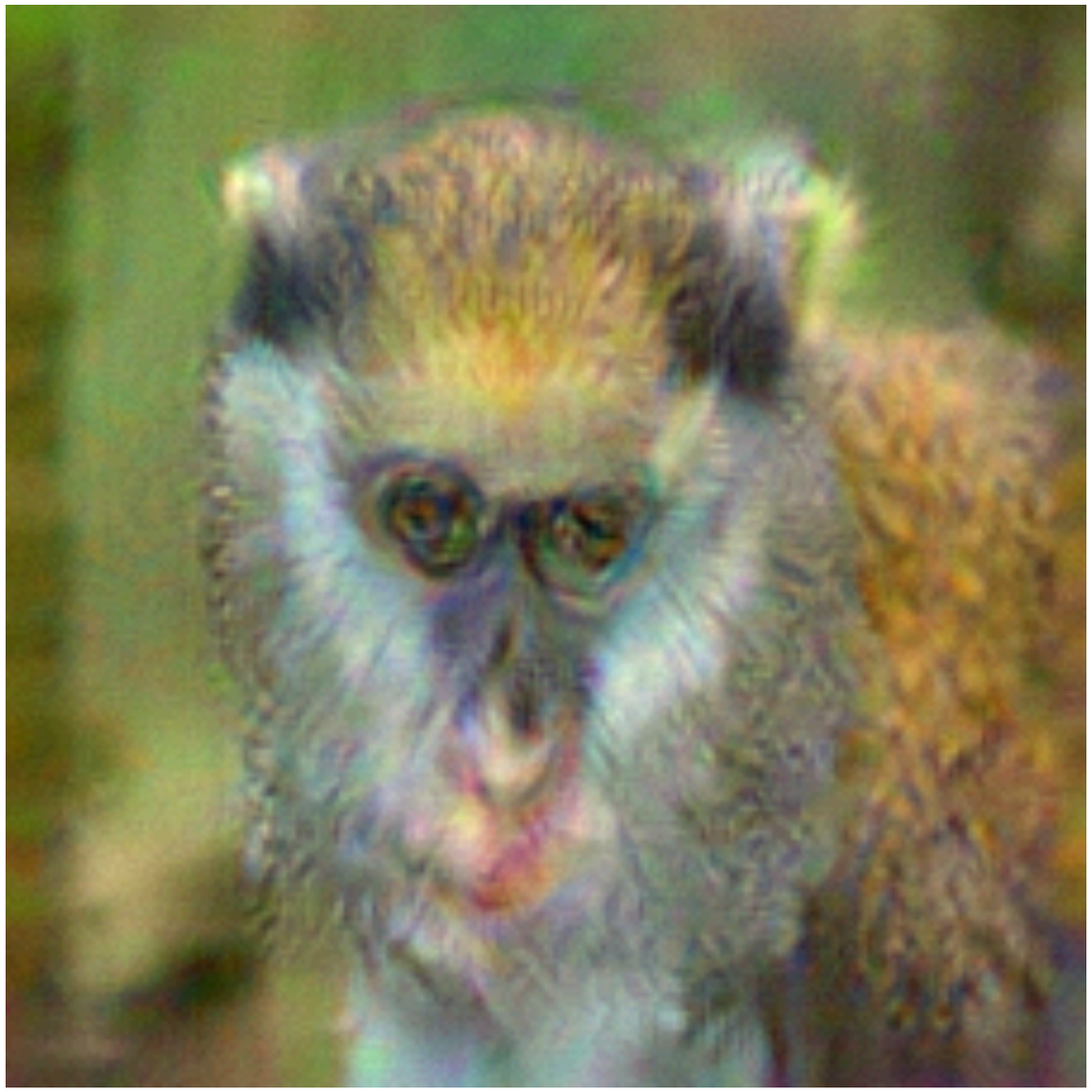
Inversion

Original Image



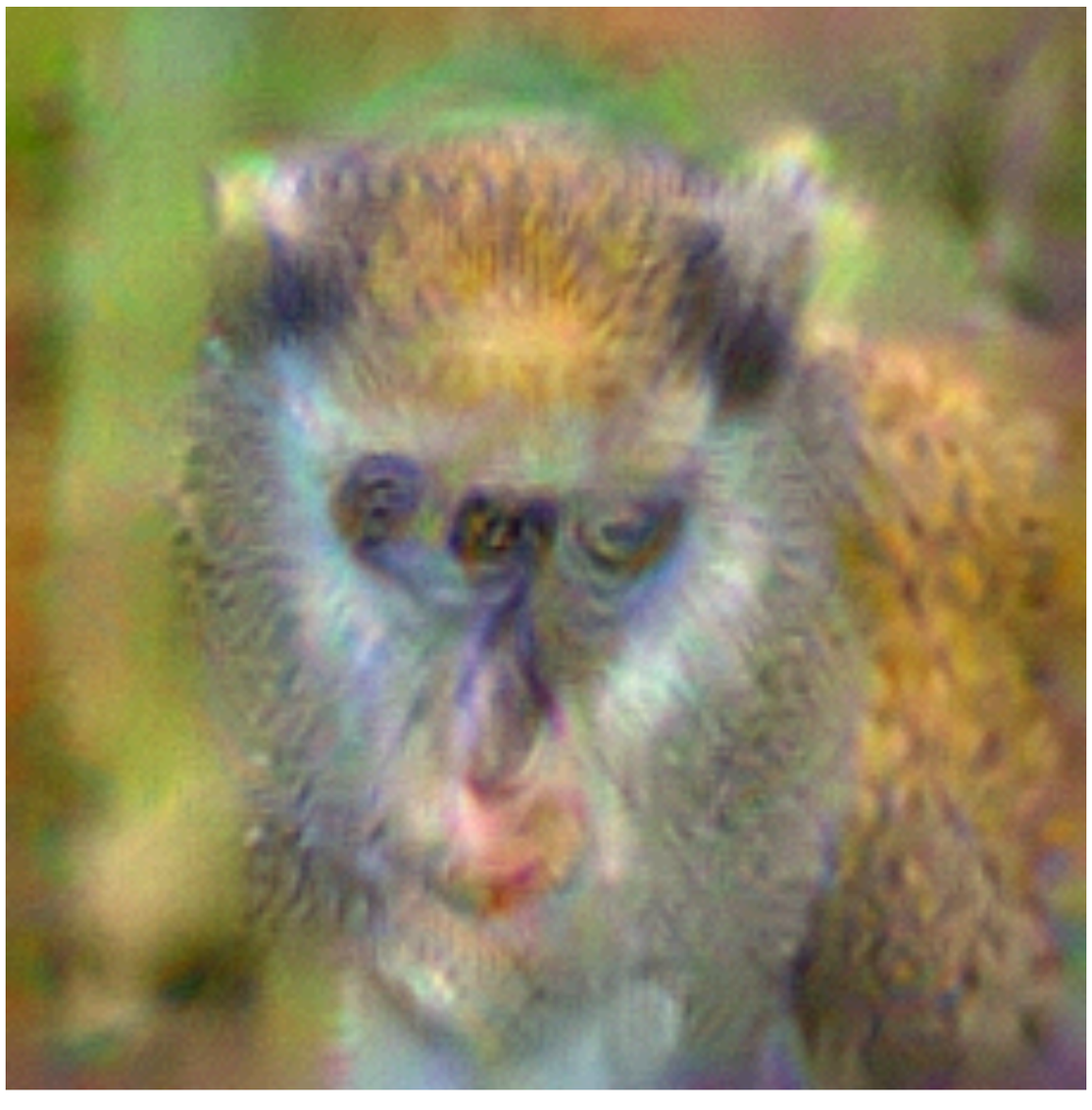
Inversion

Original Image



Inversion

Original Image



Inversion

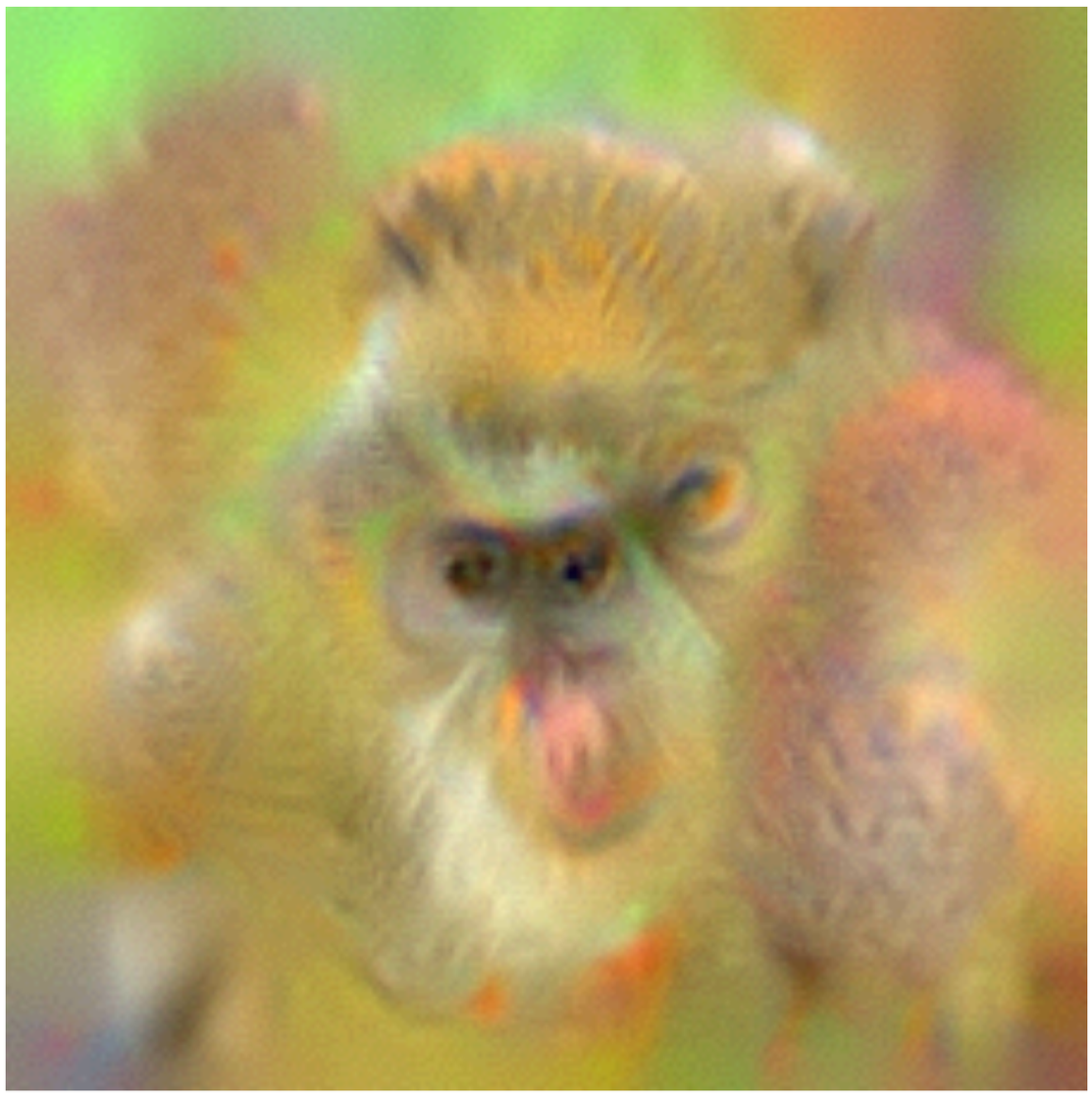
Original Image

Inversion

Original Image

Inversion

Original Image

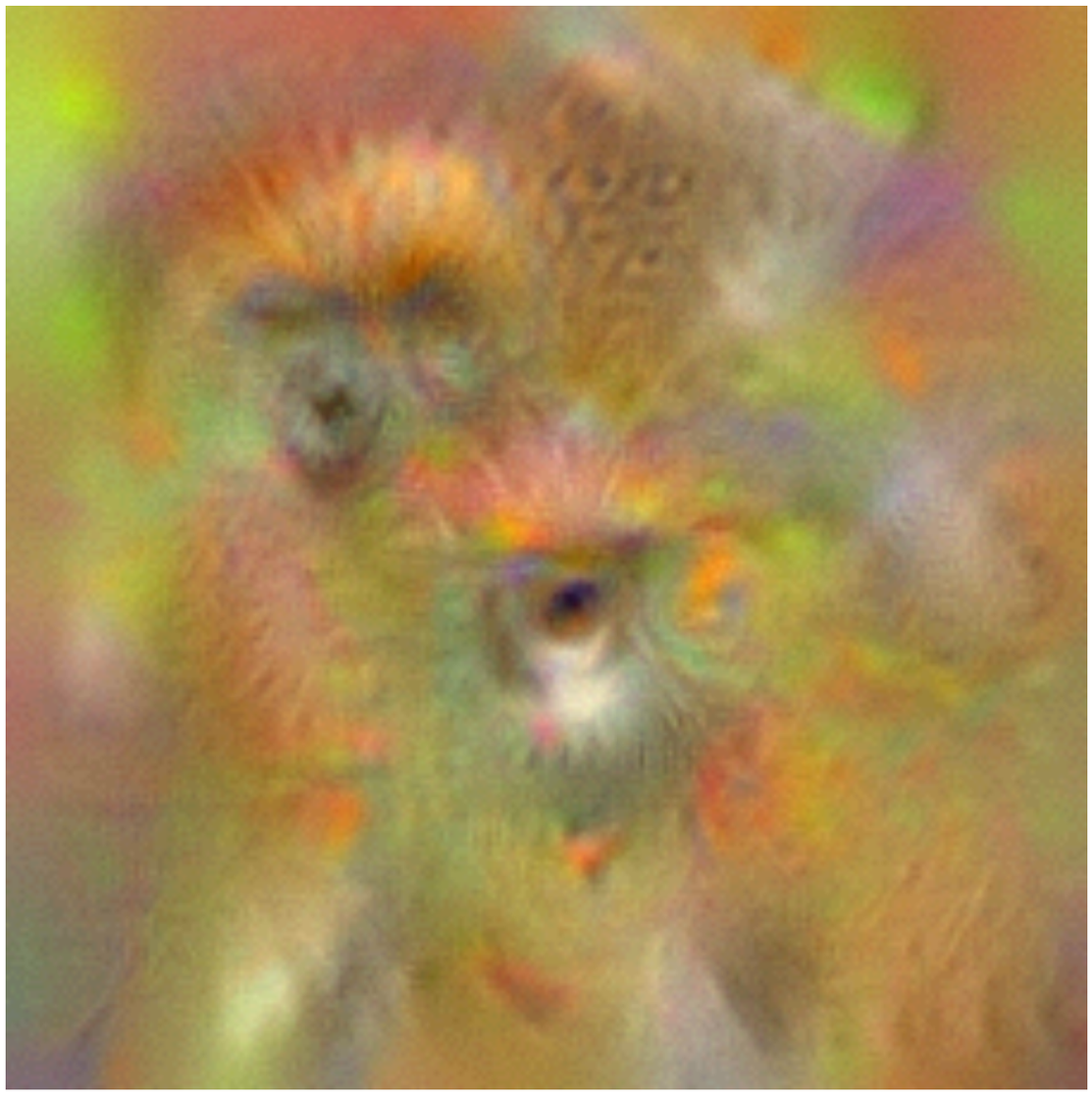


Inversion

Original Image

Inversion

Original Image



Inversion

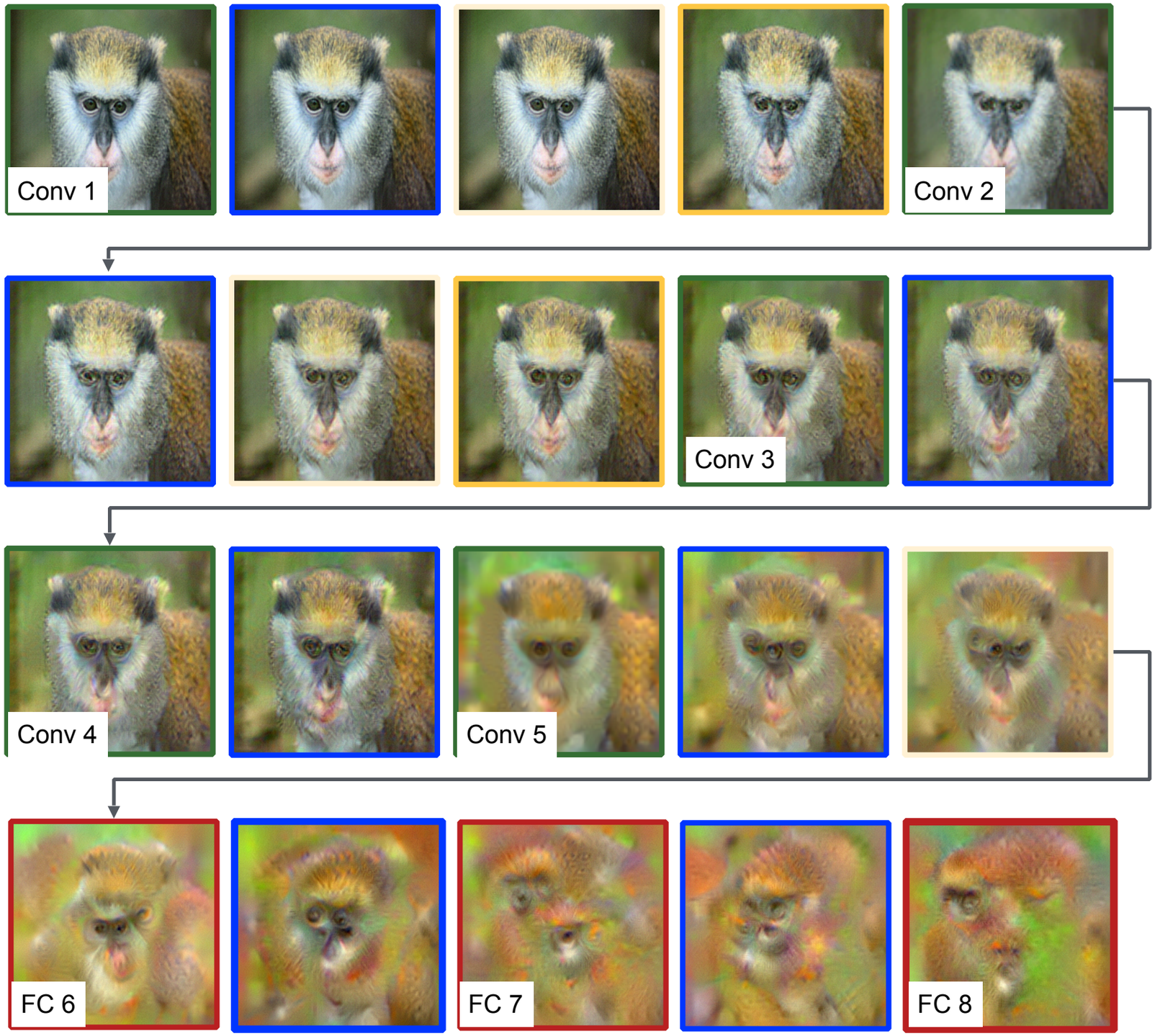
Original Image

Inversion

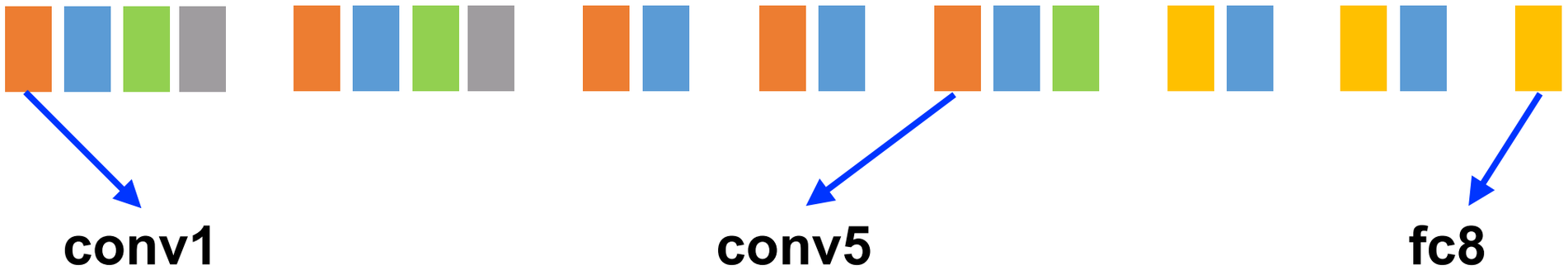
Original Image

Inverting a Deep CNN

Original Image



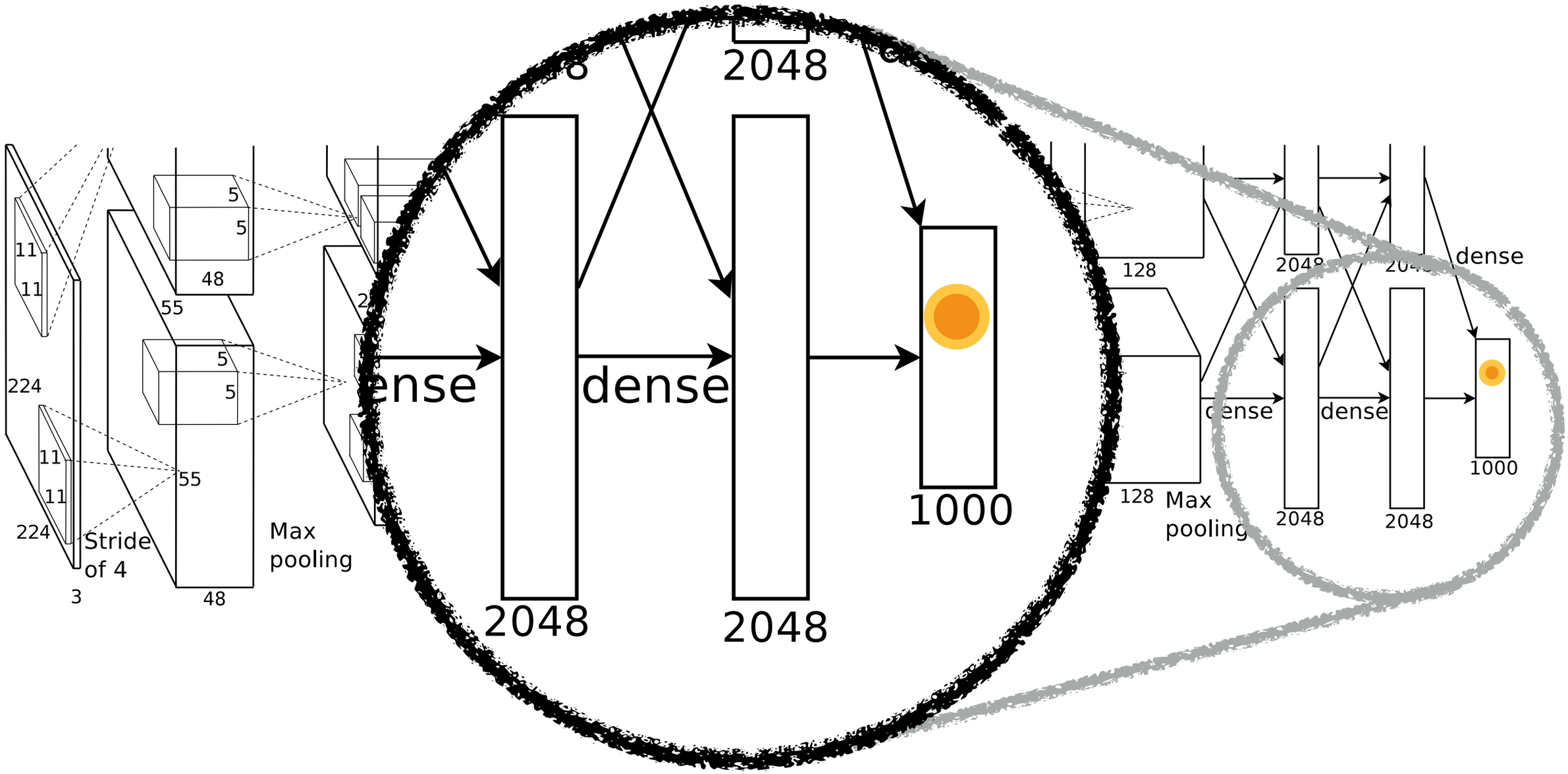
CNNs = visual codes?

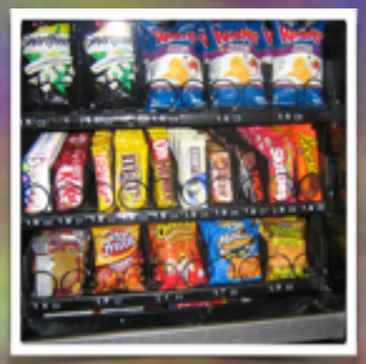


Activation Maximization

Look for an image that maximally activates a **specific feature component**

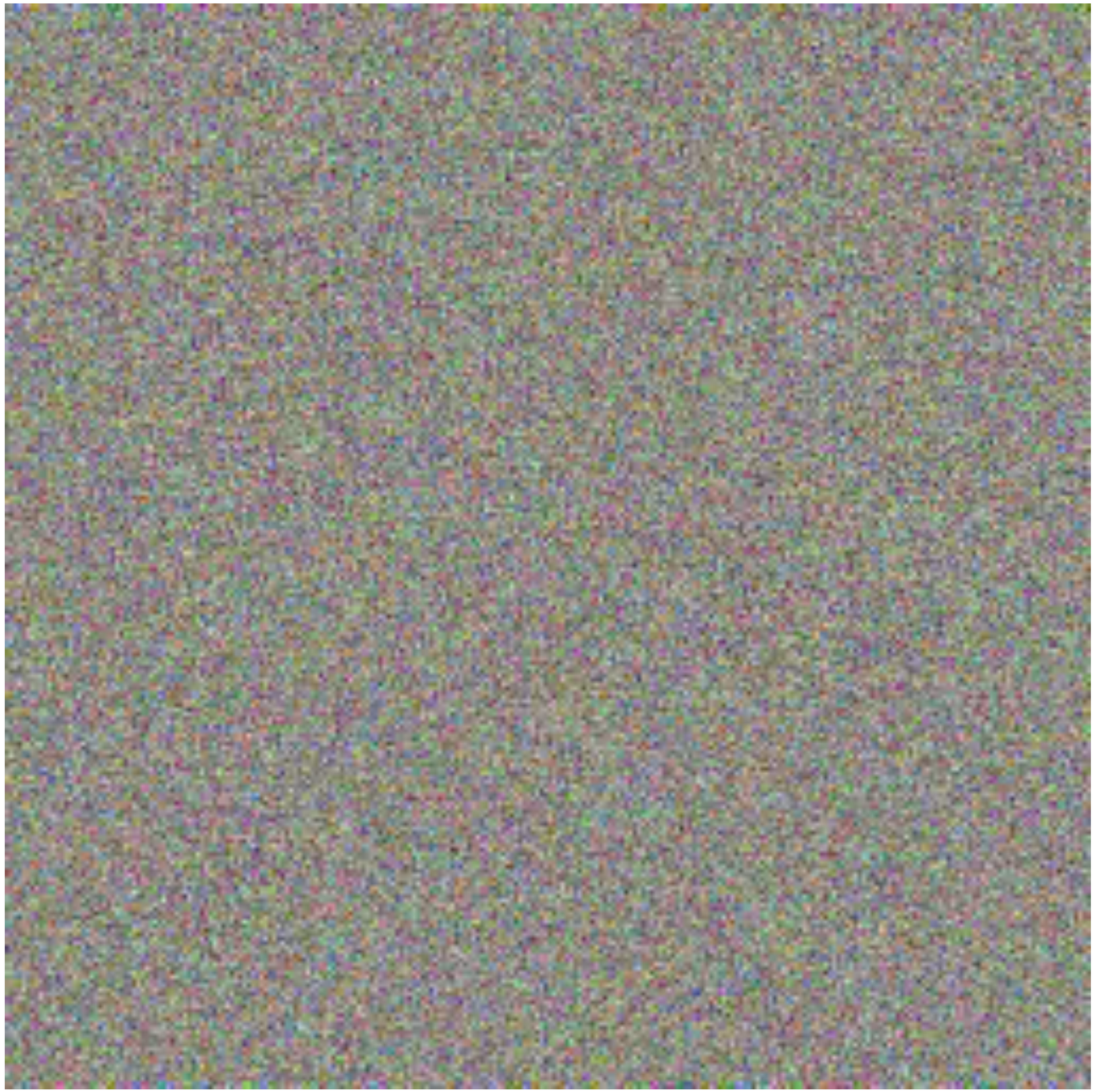
$$\min_{\mathbf{x}} -\langle \mathbf{e}_k, \Phi(\mathbf{x}) \rangle + R_{TV}(\mathbf{x}) + R_{\alpha}(\mathbf{x})$$

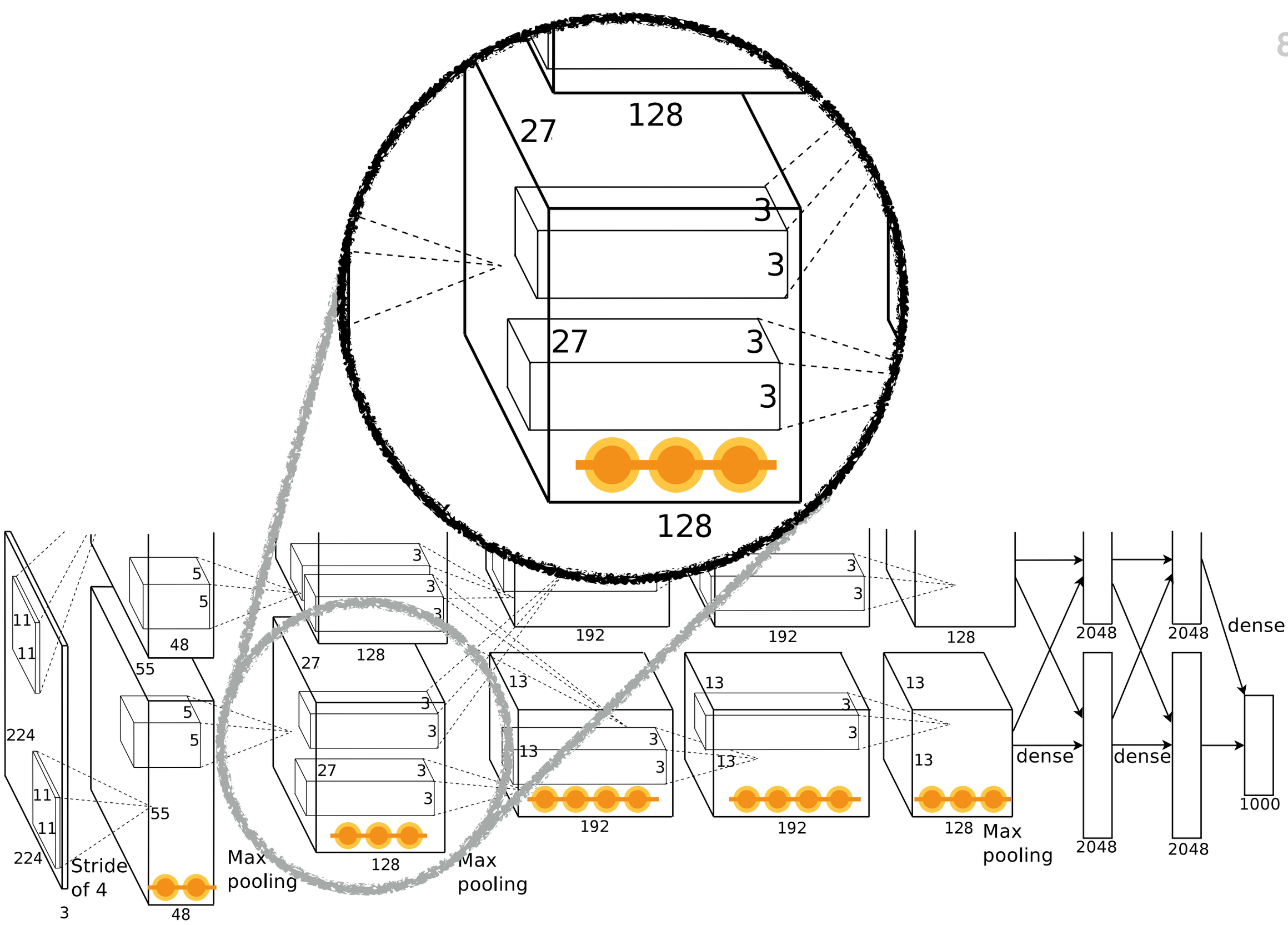




Remember: the starting point is white noise

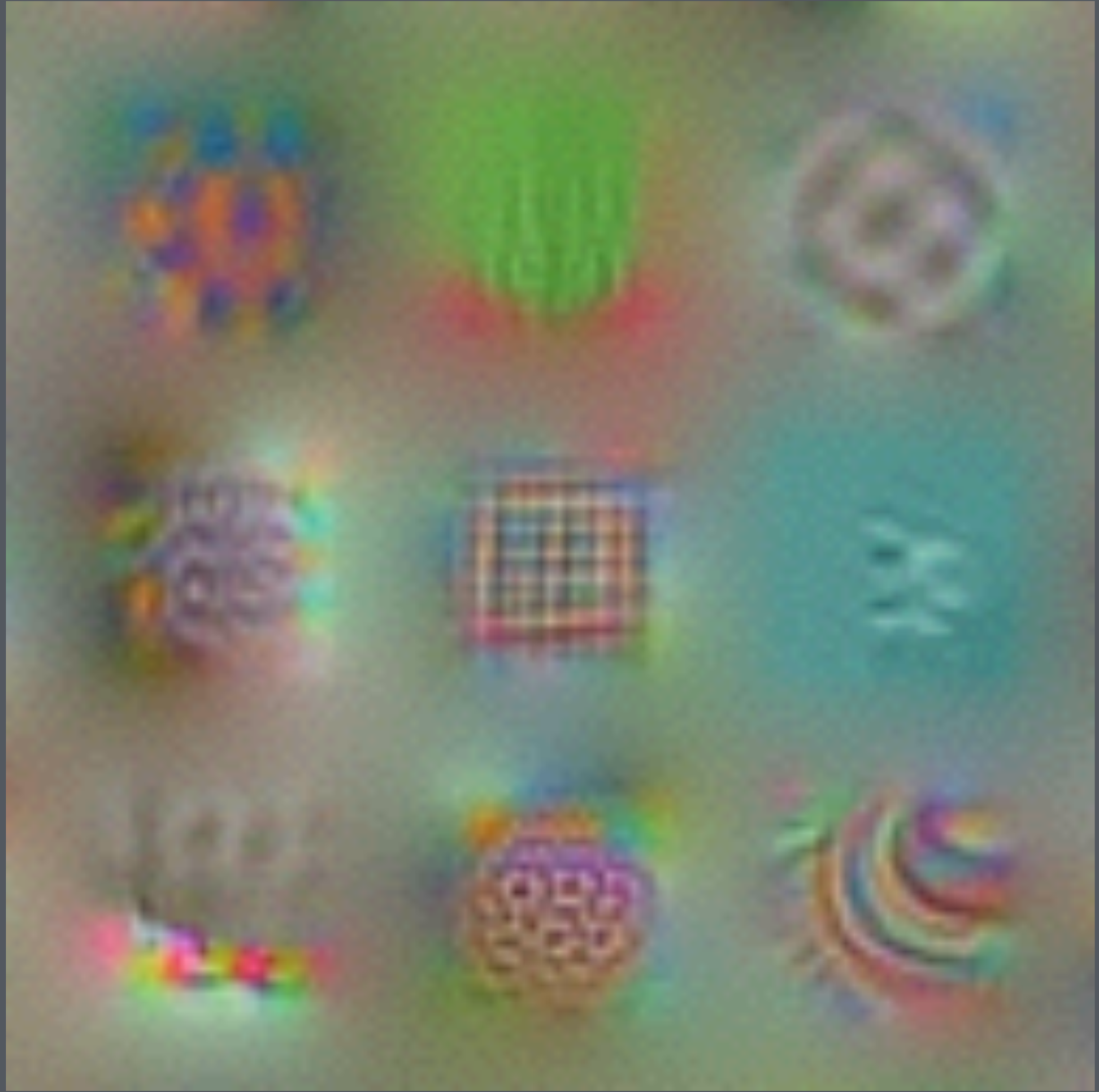
Not an image!



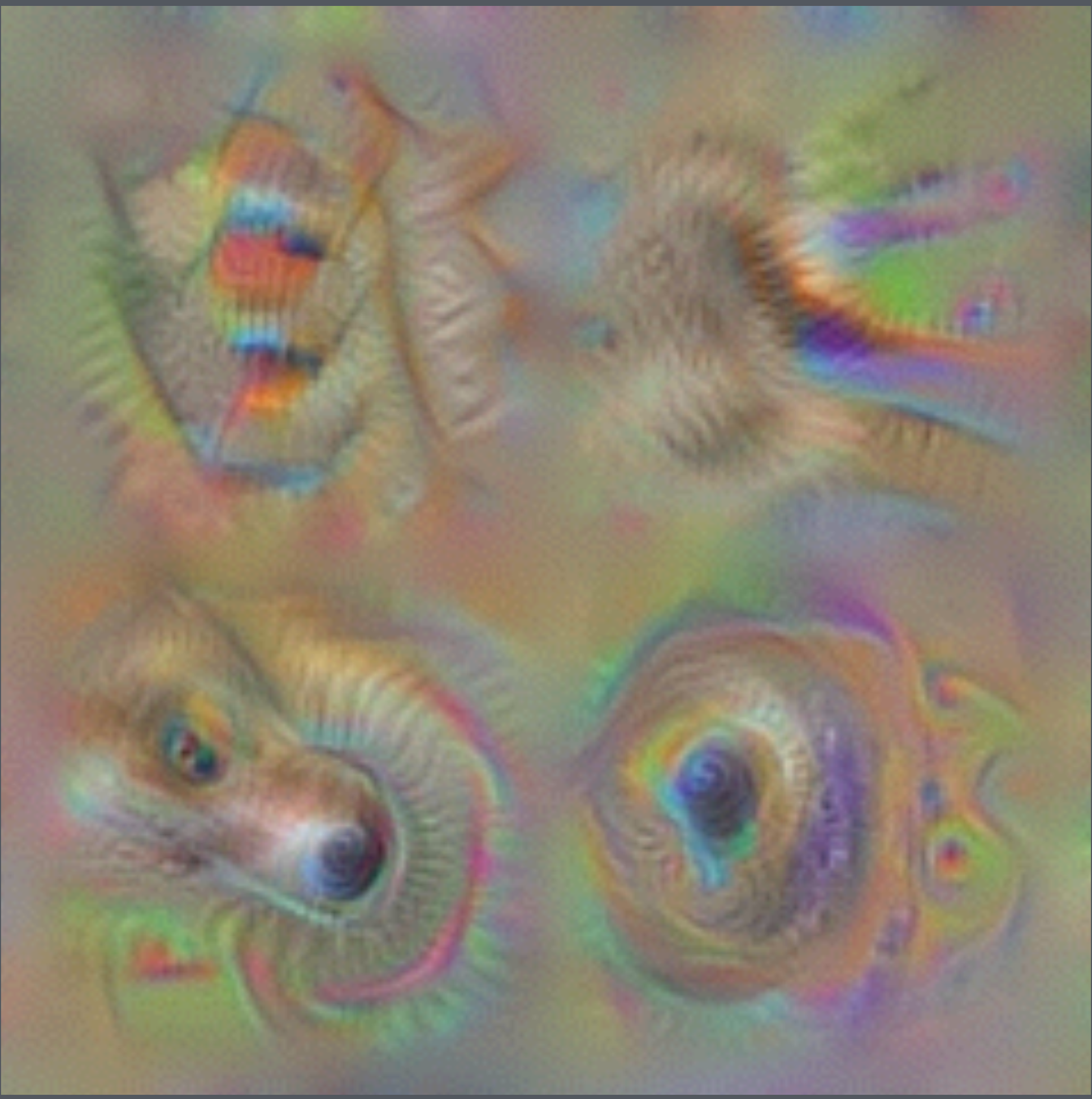

128
27
3
3
27
3
3
128
11
11
224
224
11
11
55
55
5
5
48
48
Stride
of 4
3
Max
pooling
128
27
3
3
3
3
27
3
3
128
Max
pooling
192
192
13
13
3
3
13
13
3
3
192
192
128
13
13
Max
pooling
dense
dense
2048
2048
2048
2048
dense
1000

conv1

conv2

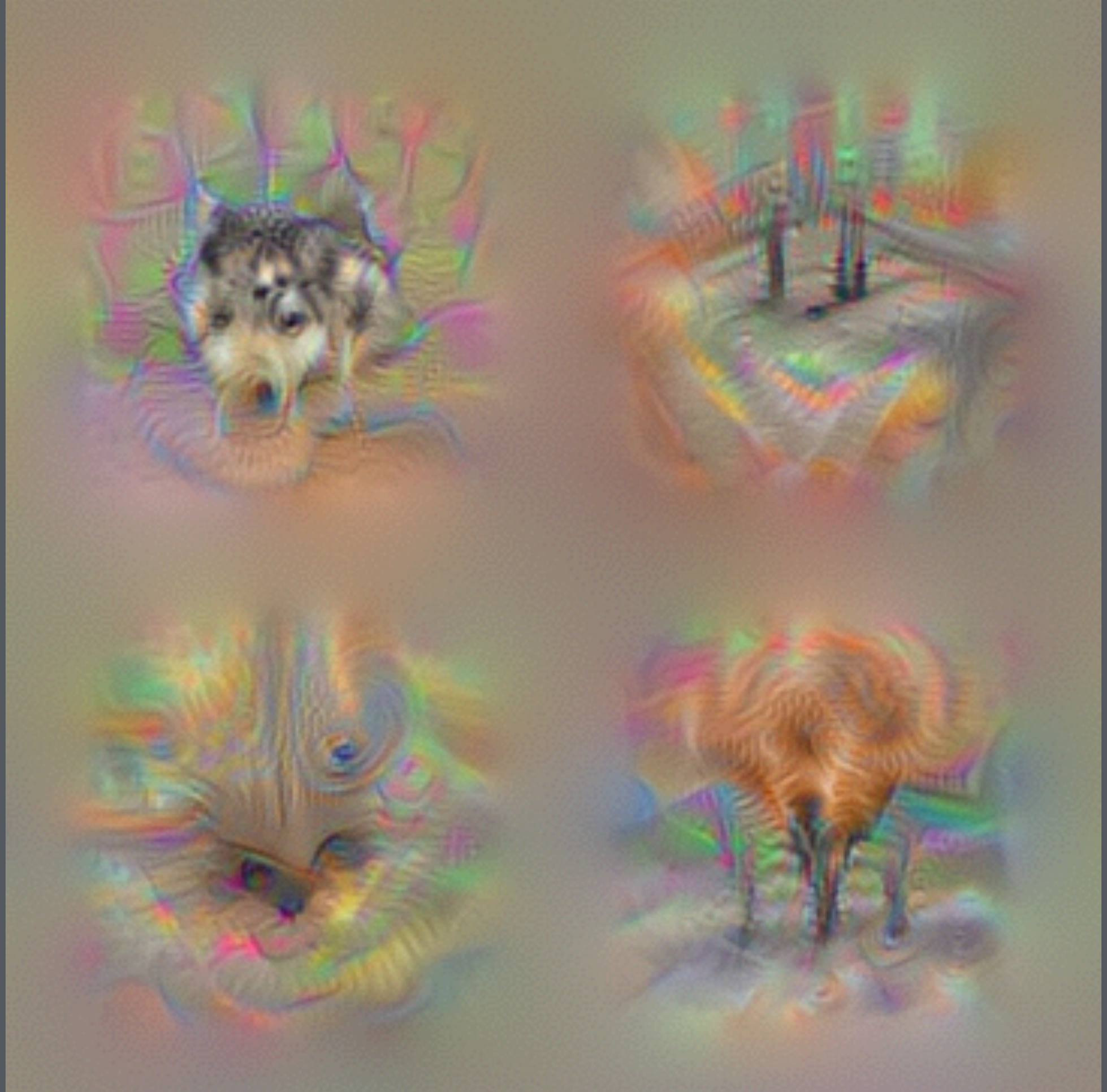


conv3



conv4

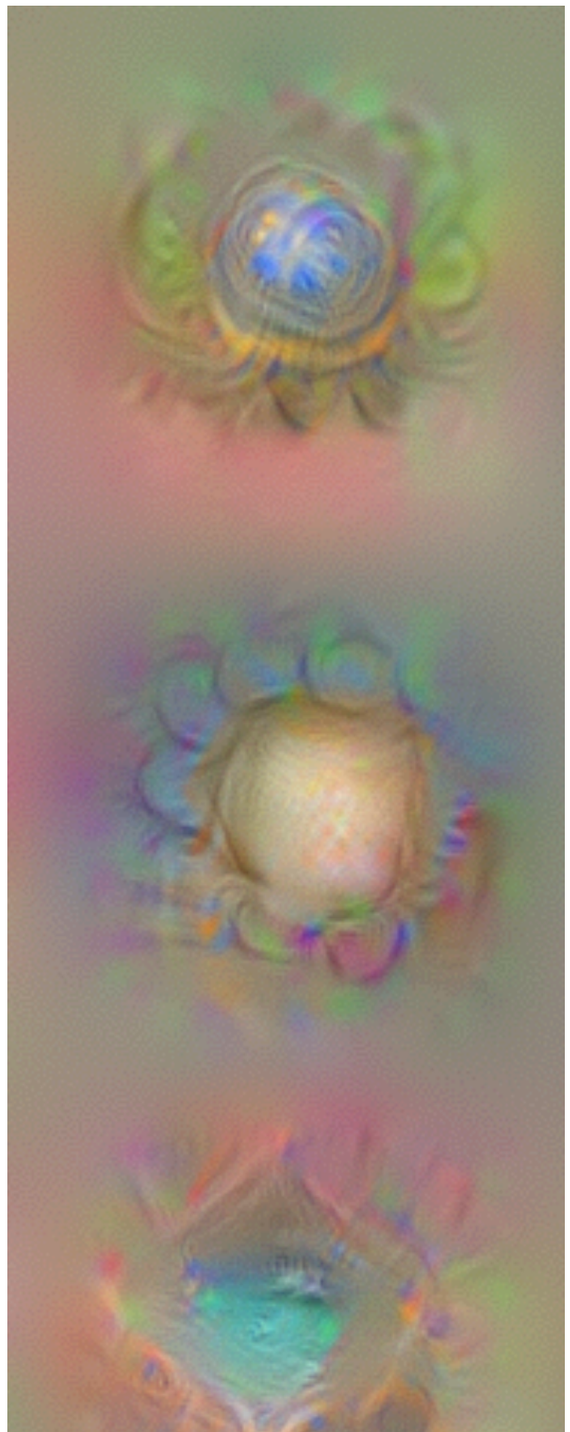
conv5



Network comparison

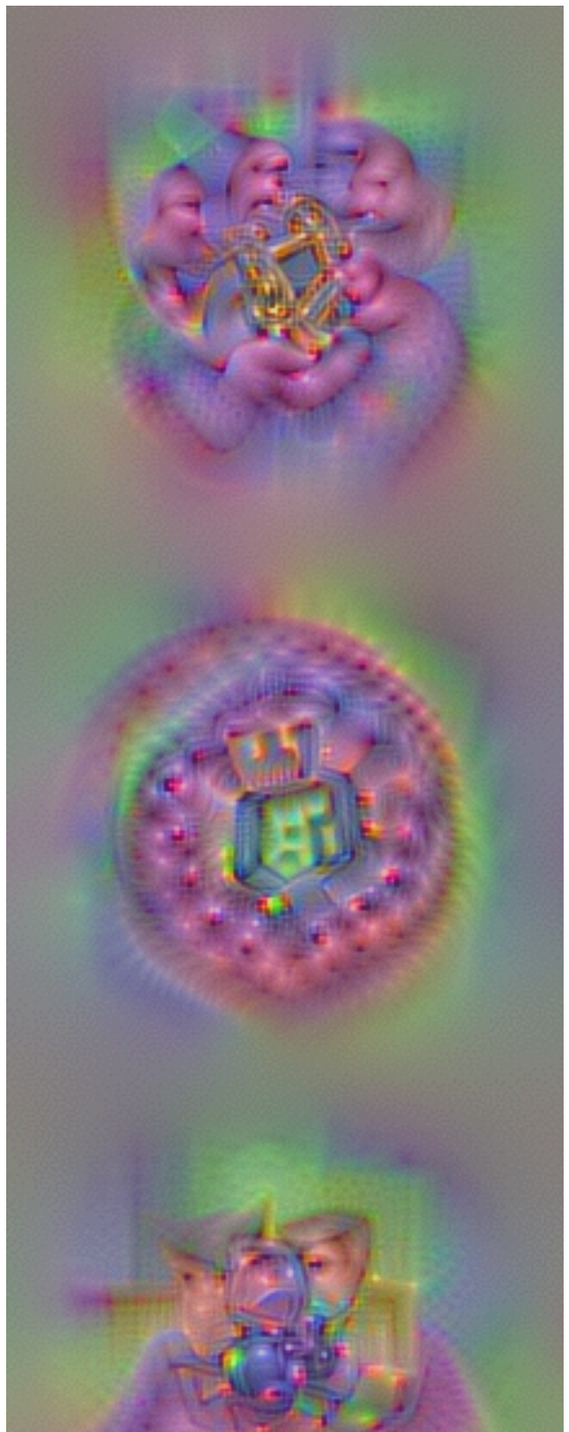
“conv5” features

AlexNet



VGG-M

VGG-VD



Caricaturization

[Google Inceptionism 2015, Mahendran et al. 2016]

Emphasise patterns that are detected by a certain representation

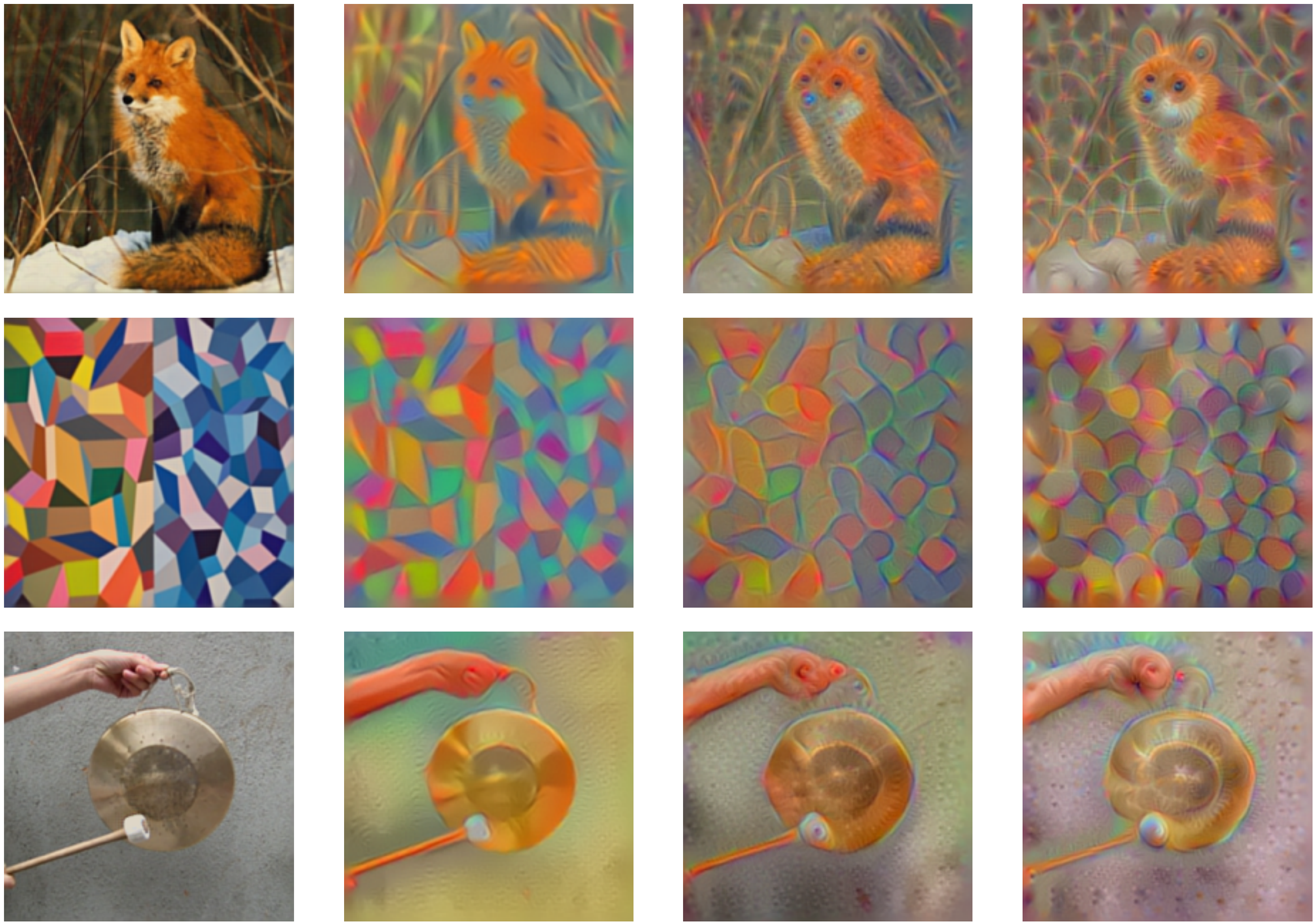
$$\min_{\mathbf{x}} -\langle \Phi(\mathbf{x}_0), \Phi(\mathbf{x}) \rangle + R_{TV}(\mathbf{x}) + R_{\alpha}(\mathbf{x})$$

Key differences:

- the starting point **is** the image $\mathbf{x}_0$
- particular configurations of features are emphasized, not individual features

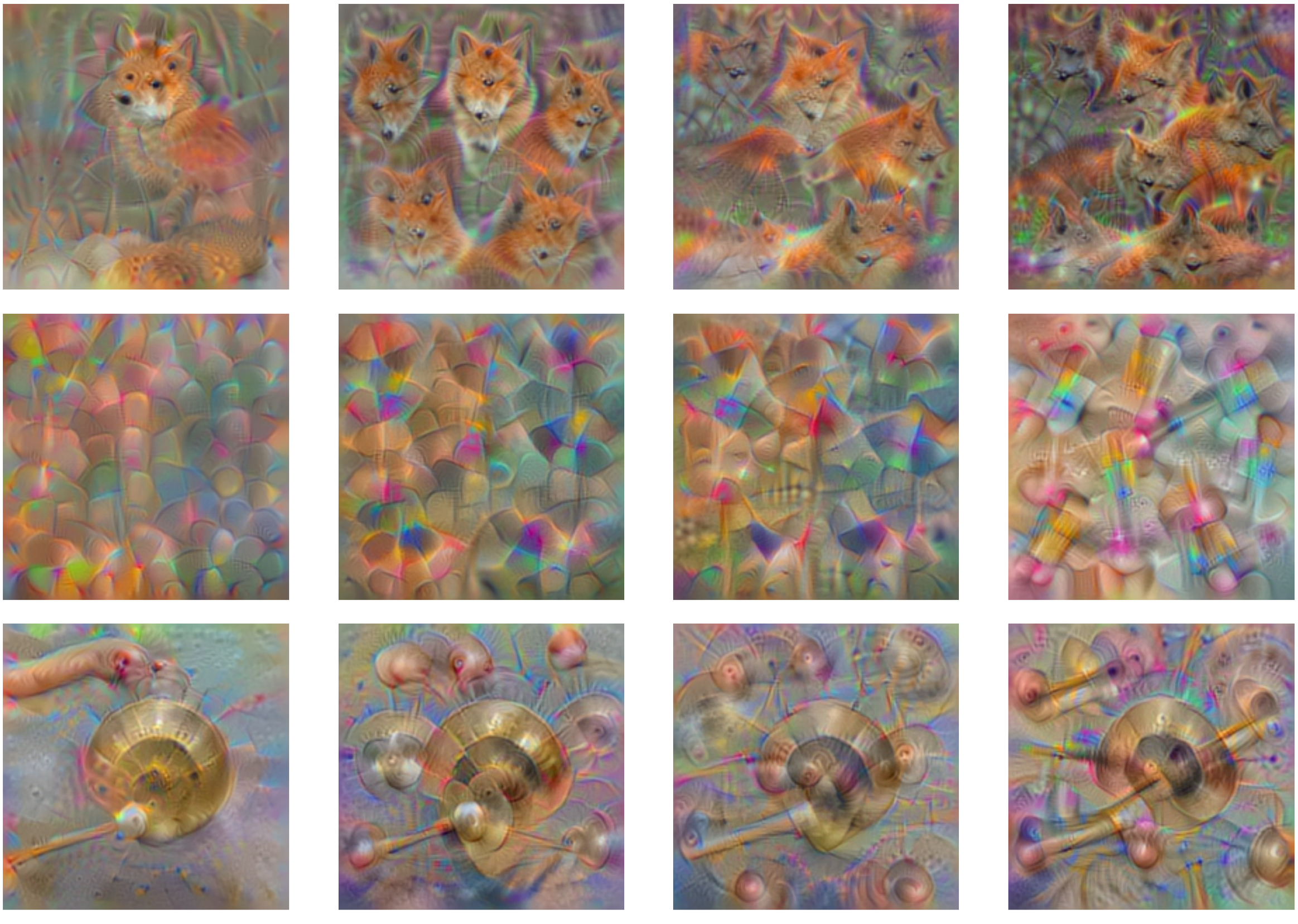
Caricaturization (VGG-M)

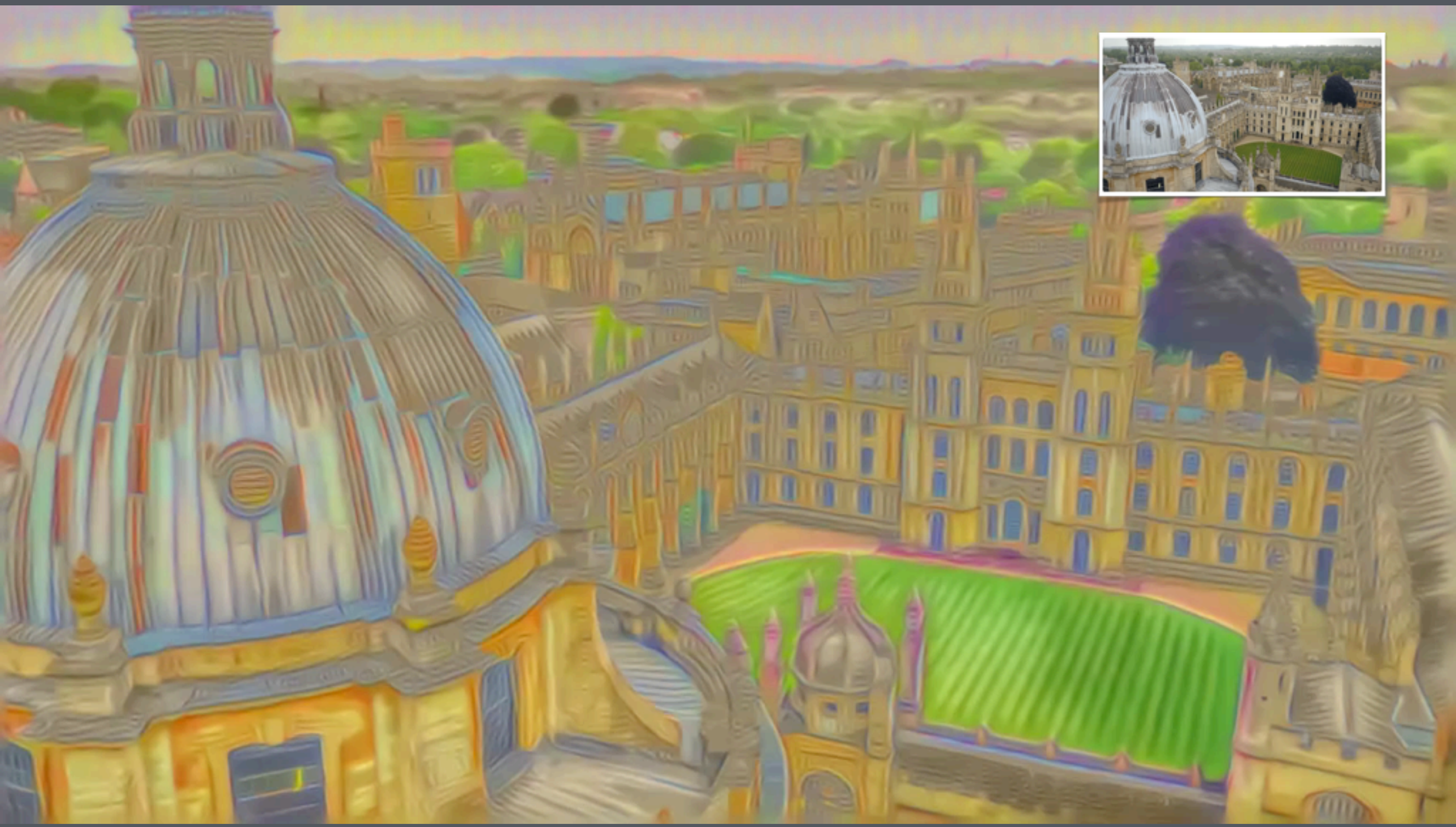
input | conv2 | conv3 | conv4



Caricaturization (VGG-M)

conv5 fc6 fc7 fc8





Understanding visual representations

Intro

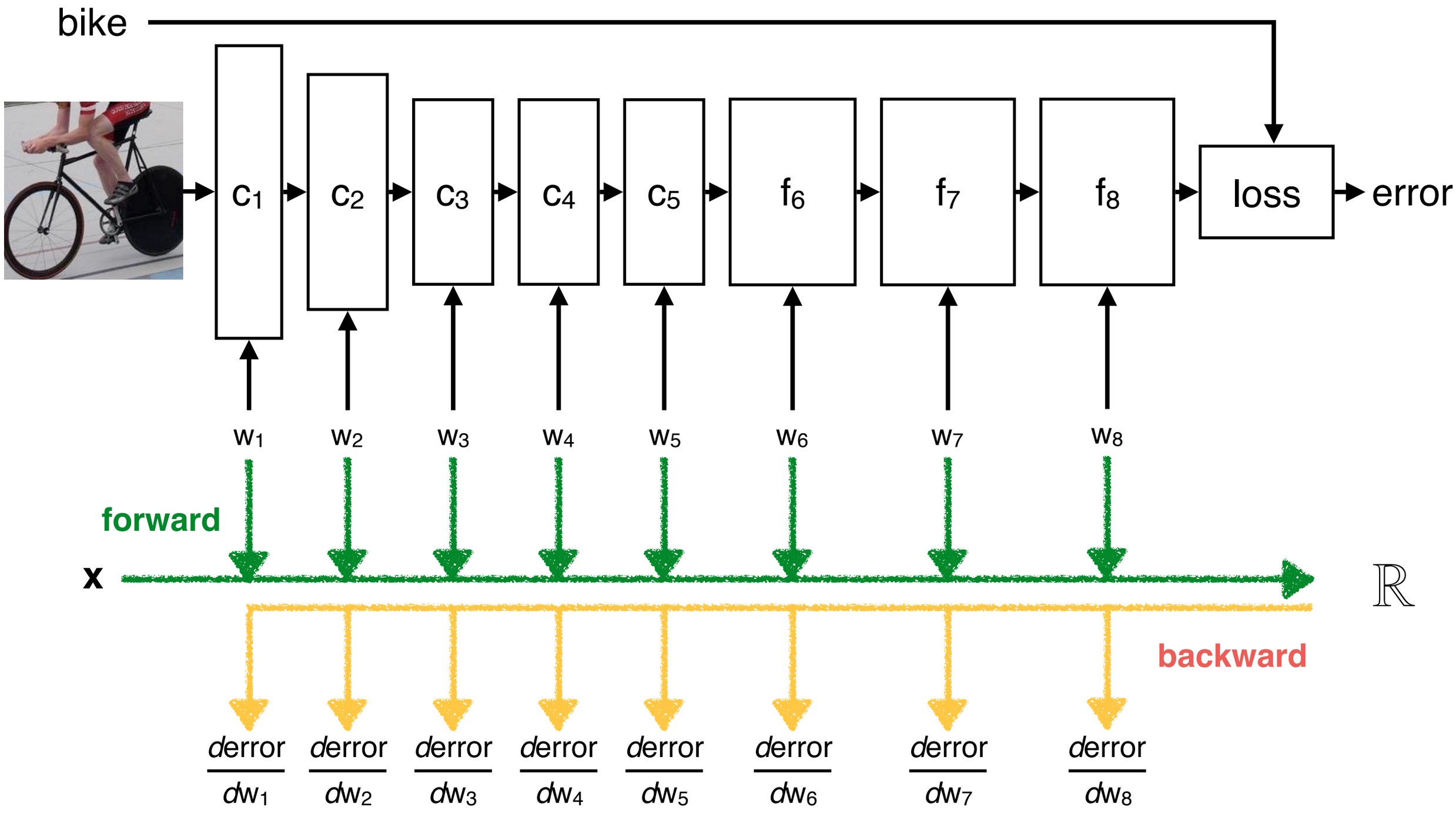
Visualizing representations

Backpropagation networks and “deconvolution”

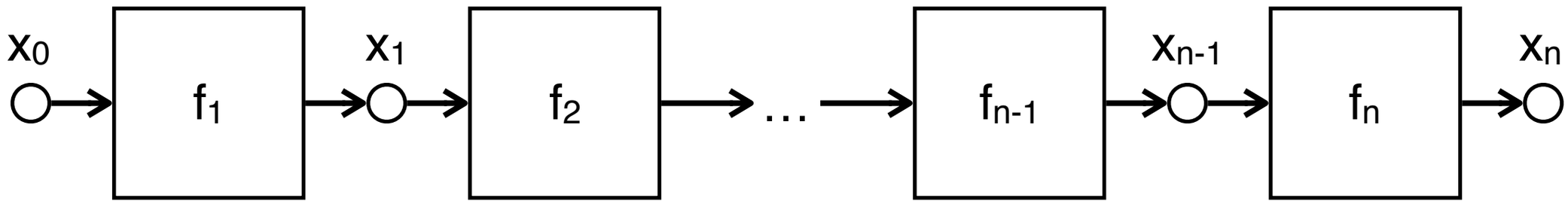
Representations: equivalence & transformations

Backpropagation

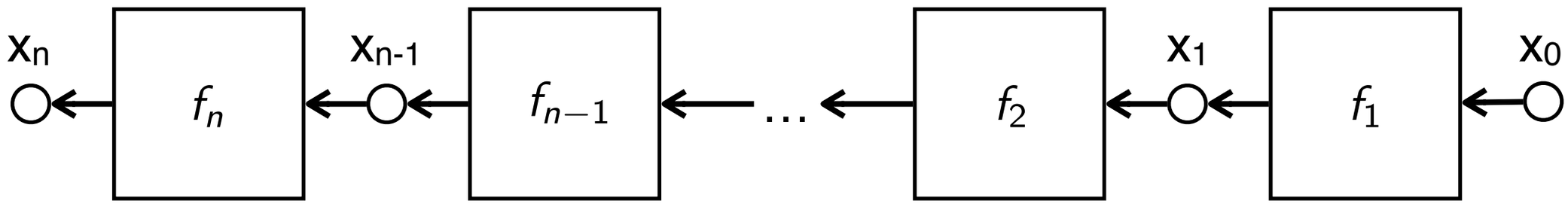
Compute derivatives using the chain rule



Chain rule: scalar version



Chain rule: scalar version



A composition of *n* functions

$$\mathbf{x}_n = (\ f_n \circ f_{n-1} \circ \ldots \circ f_2 \circ f_1\ )\ (\mathbf{x}_0)$$

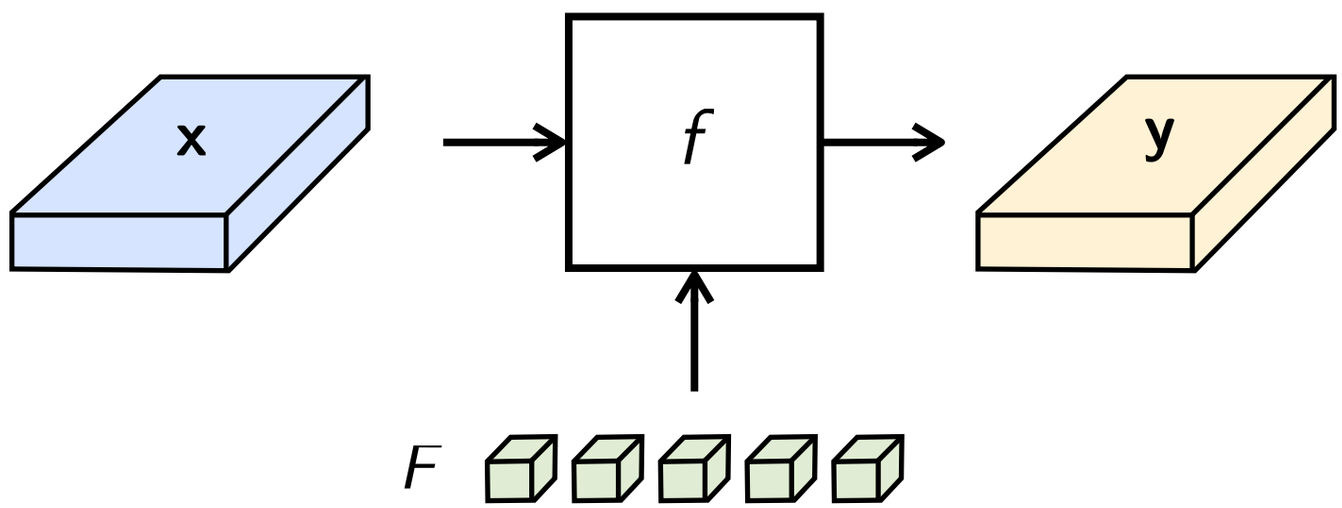
$$\frac{dx_n}{dx_0} = \frac{df_n}{dx_{n-1}} \times \frac{df_{n-1}}{dx_{n-2}} \times \ldots \times \frac{df_2}{dx_1} \times \frac{df_1}{dx_0}$$

Derivative ← chain rule

Tensor-valued functions

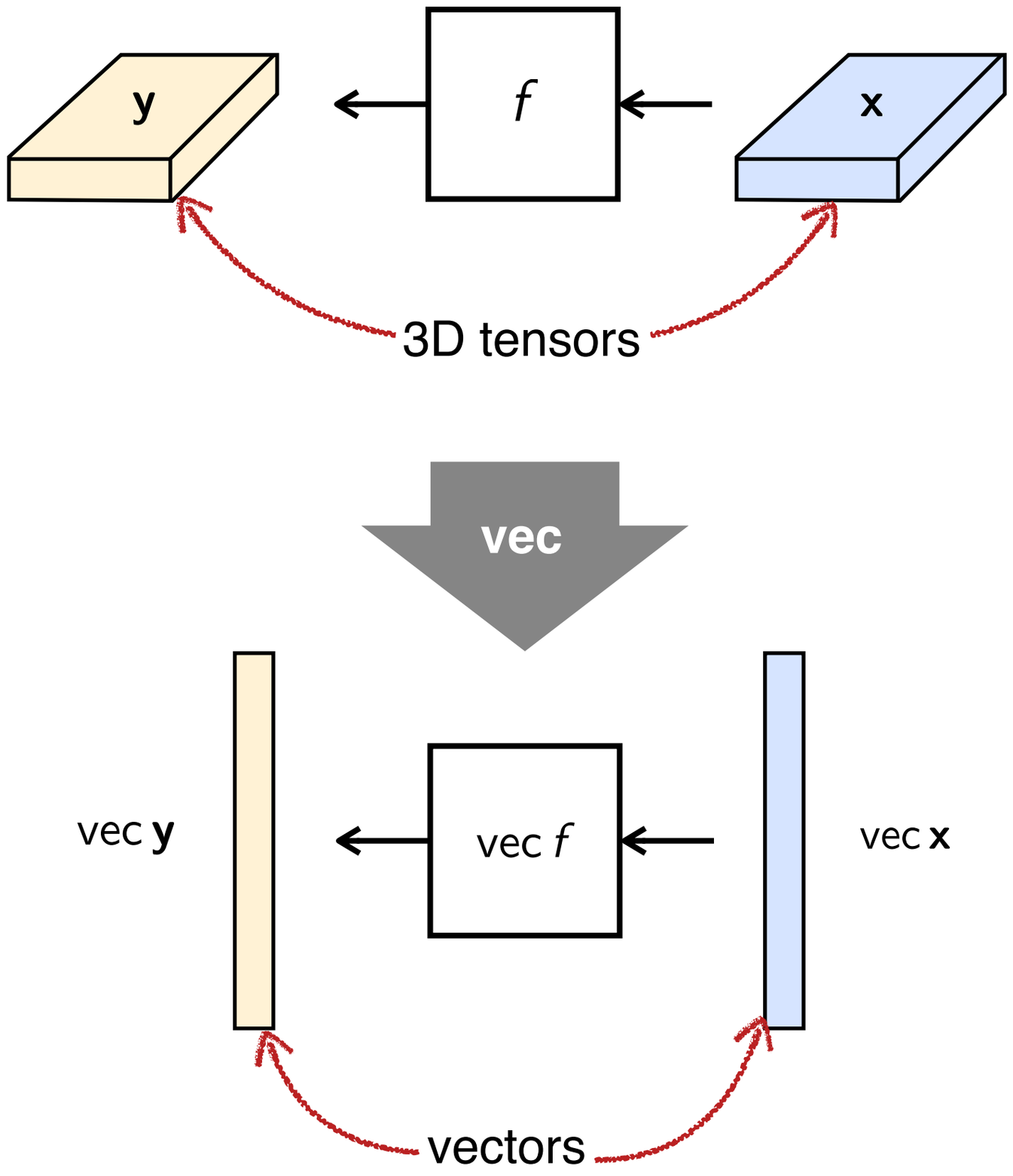
E.g. linear convolution = bank of 3D filters

$$\mathbf{y} = F * \mathbf{x} + b$$

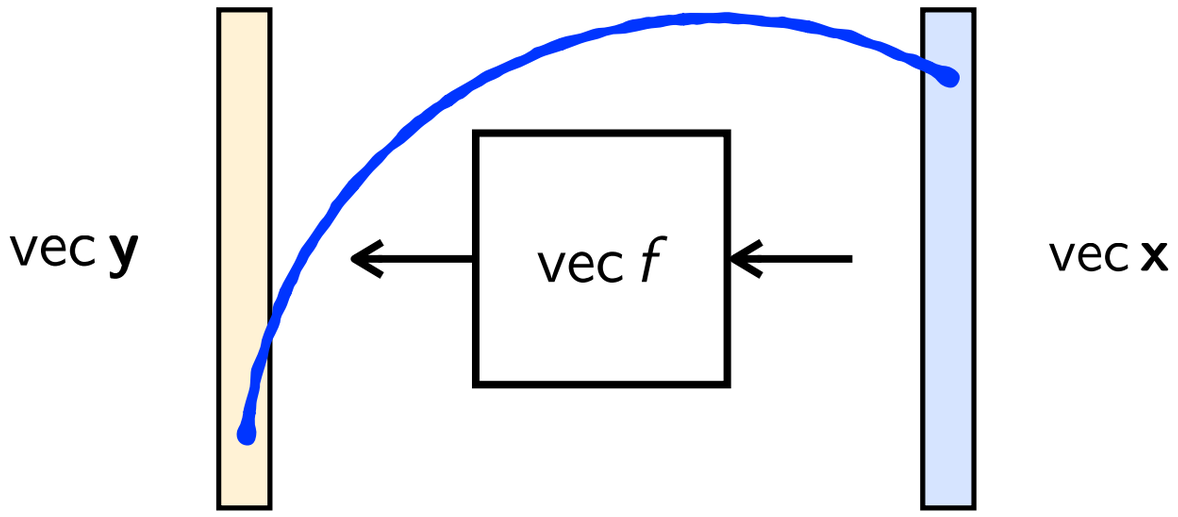


	height	width	channels	instances
input **x**	H	W	C	1 or N
filters F	H_f	W_f	C	K
output **y**	$H - H_f + 1$	$W - W_f + 1$	K	1 or N

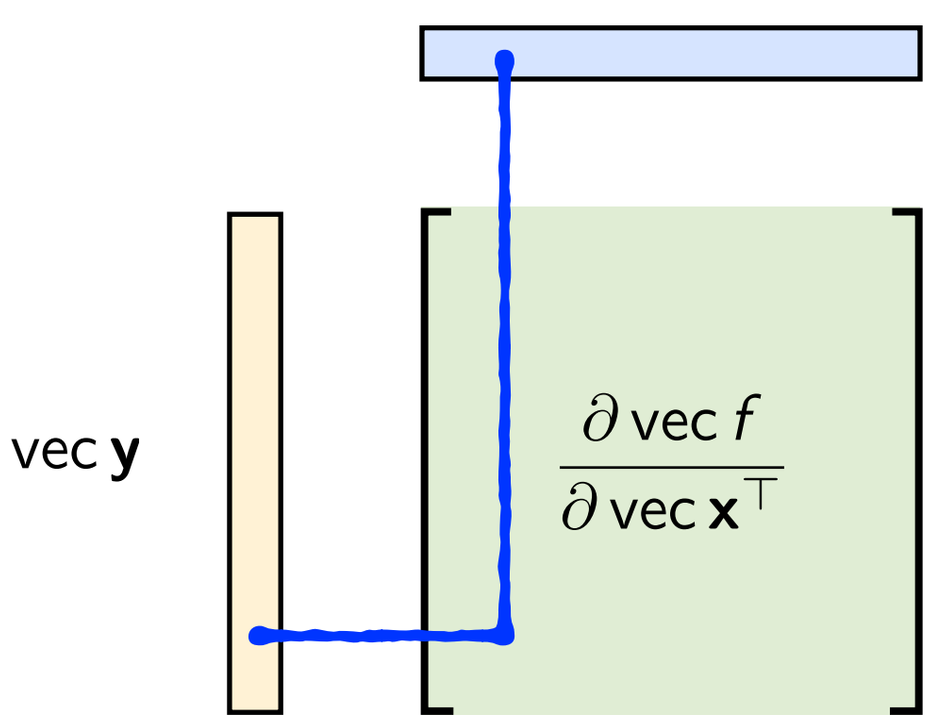
Vector representation



Derivative of tensor-valued functions



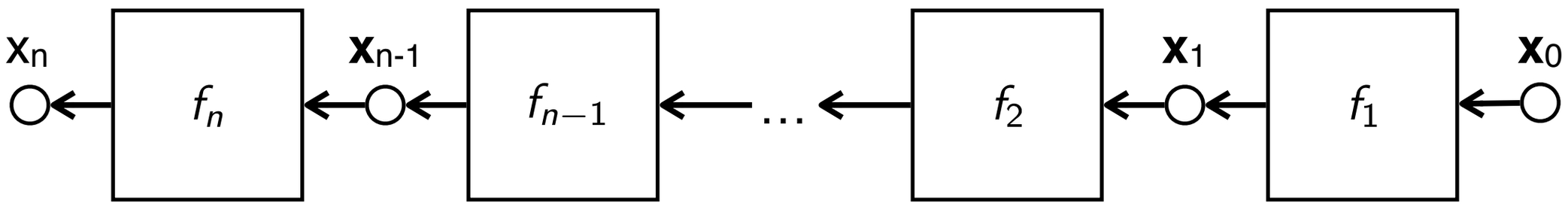
Derivative (Jacobian): every output element w.r.t. every input element!



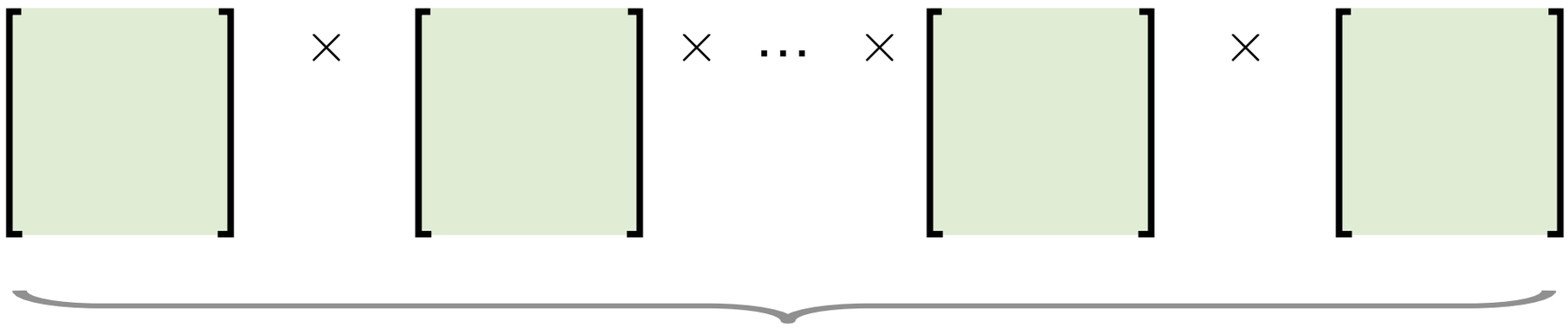
The **vec** operator
allows us to use
a familiar matrix notation
for the derivatives

Chain rule: tensor version

Using vec() and matrix notation

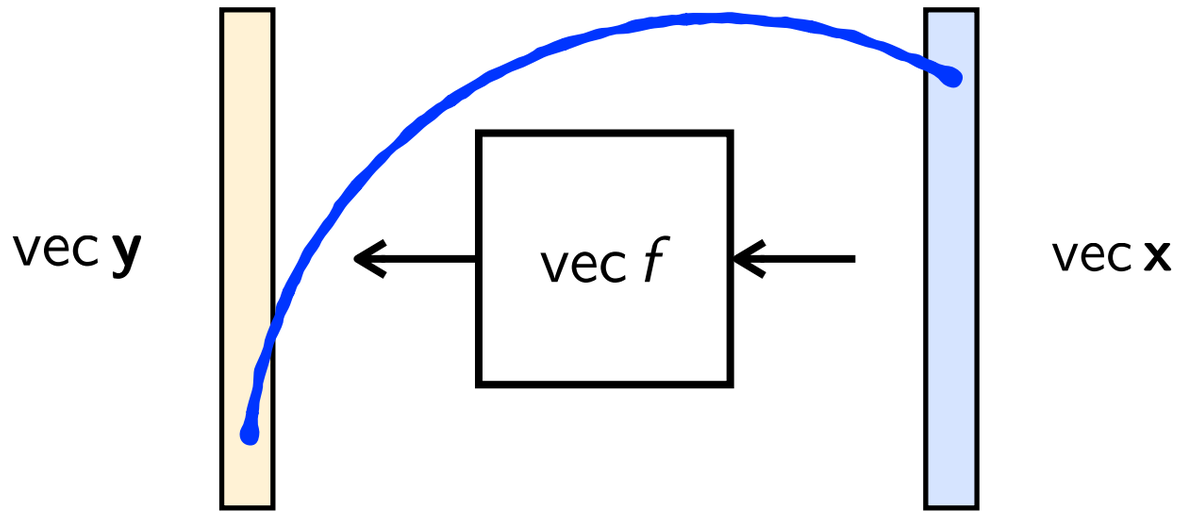


$$\frac{d \operatorname{vec} f_n}{d \operatorname{vec} \mathbf{x}_{n-1}^\top} \times \frac{d \operatorname{vec} f_{n-1}}{d \operatorname{vec} \mathbf{x}_{n-2}^\top} \times \cdots \times \frac{d \operatorname{vec} f_2}{d \operatorname{vec} \mathbf{x}_1^\top} \times \frac{d \operatorname{vec} f_1}{d \operatorname{vec} \mathbf{x}_0^\top}$$

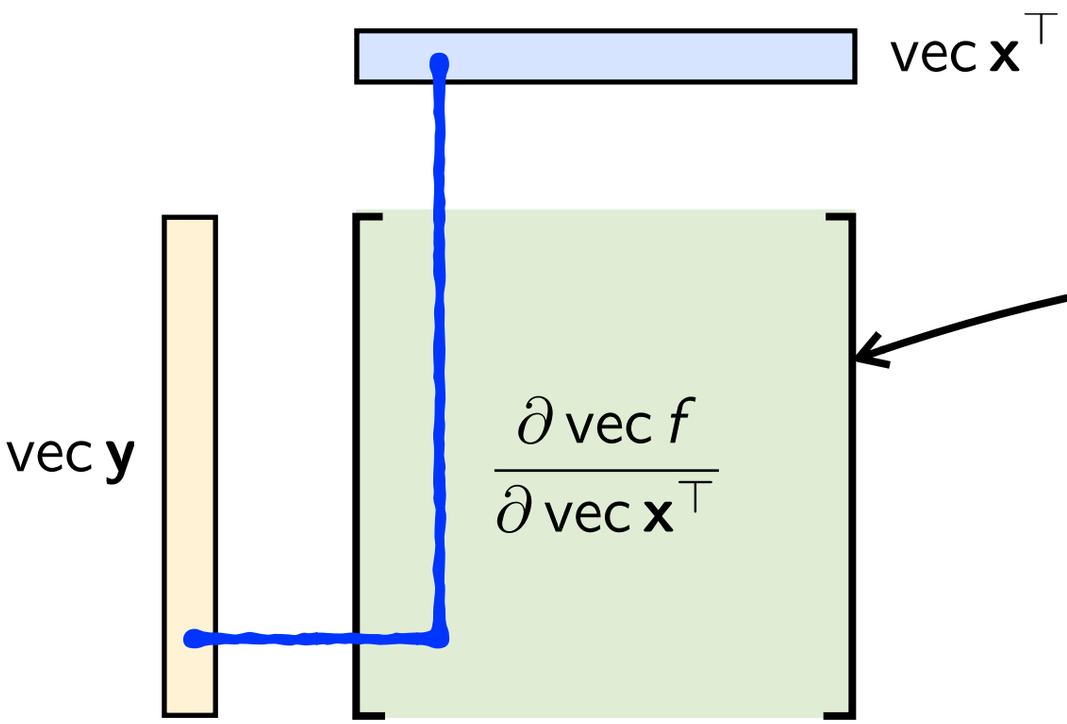


$$\frac{d \operatorname{vec} f}{d \operatorname{vec} \mathbf{x}_0^\top}$$

The (unbearable) size of tensor derivatives

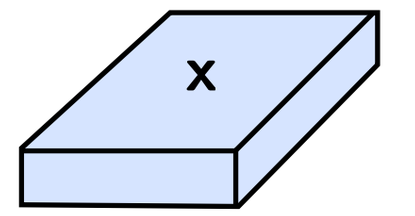


The size of these Jacobian matrices is **huge**. Example:

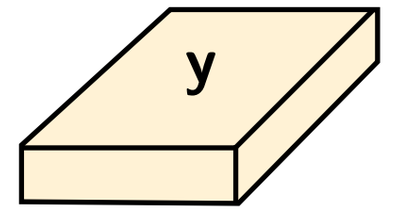


275 B elements

1 TB of memory required !!



32 × 32 × 512

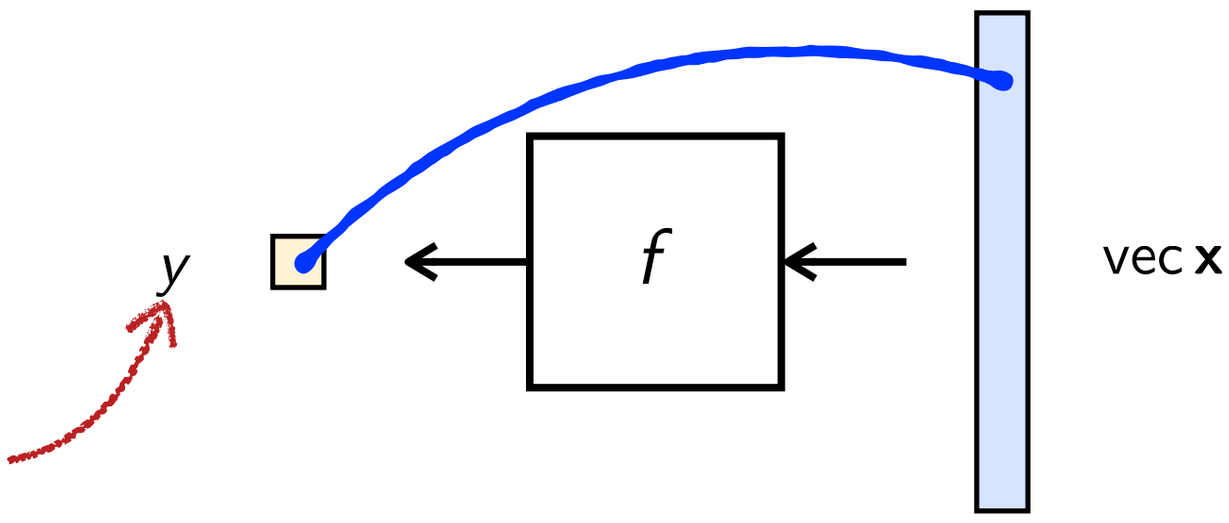


32 × 32 × 512

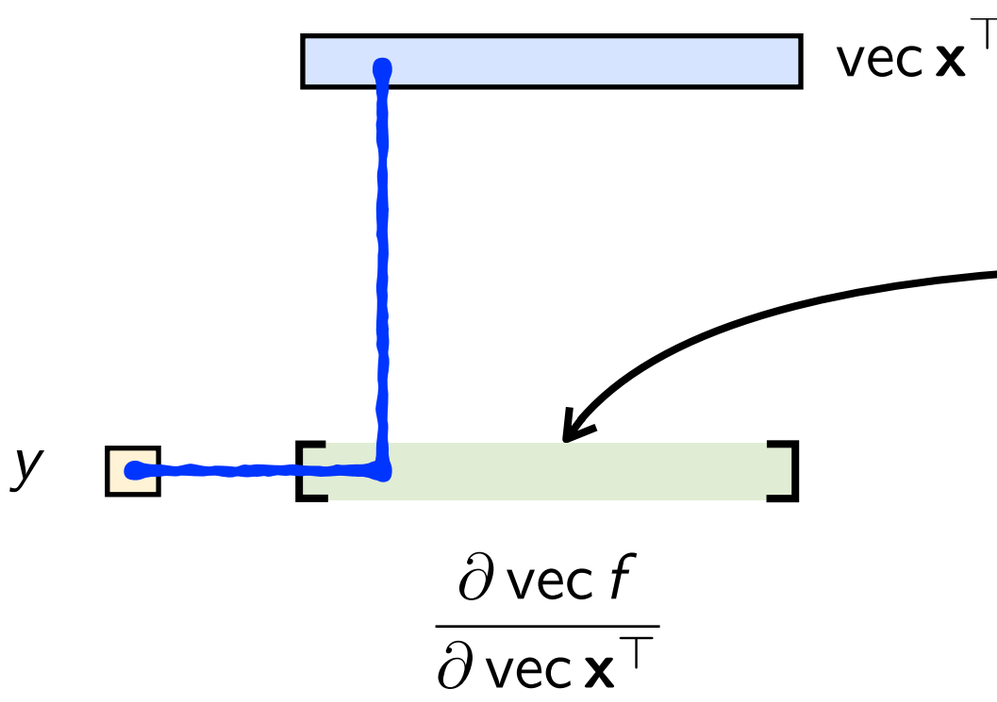
Unless the output is a scalar

Scalar

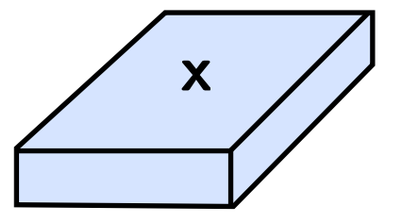
This is always the case
if the last layer
is the **loss function**



Now the Jacobian has the same size as **x**. Example:



Just 2MB of memory



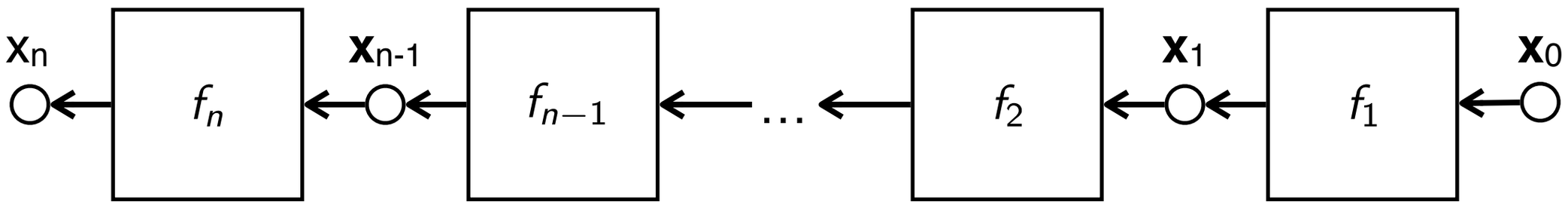
32 × 32 × 512

y

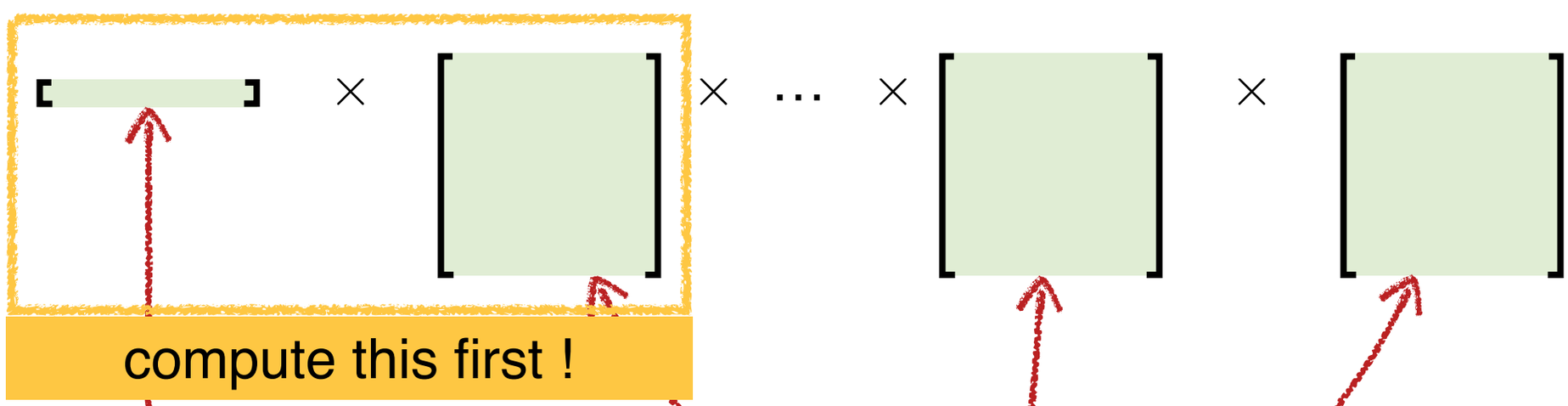
1 × 1 × 1

Backpropagation

Assume that x_n is a scalar (e.g. loss)

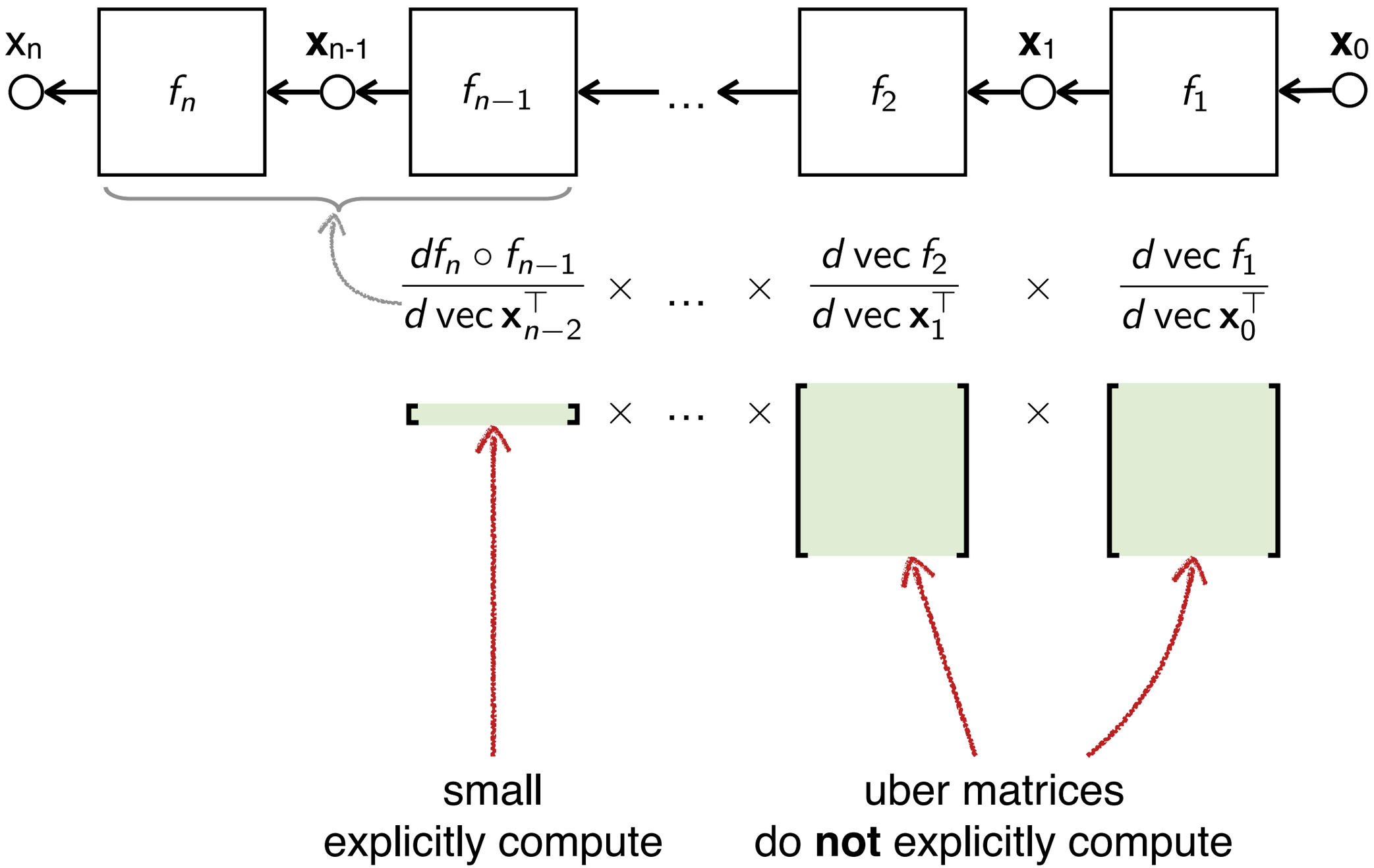


$$\frac{df_n}{d\,\mathrm{vec}\,\mathbf{x}_{n-1}^\top} \times \frac{d\,\mathrm{vec}\,f_{n-1}}{d\,\mathrm{vec}\,\mathbf{x}_{n-2}^\top} \times \cdots \times \frac{d\,\mathrm{vec}\,f_2}{d\,\mathrm{vec}\,\mathbf{x}_1^\top} \times \frac{d\,\mathrm{vec}\,f_1}{d\,\mathrm{vec}\,\mathbf{x}_0^\top}$$



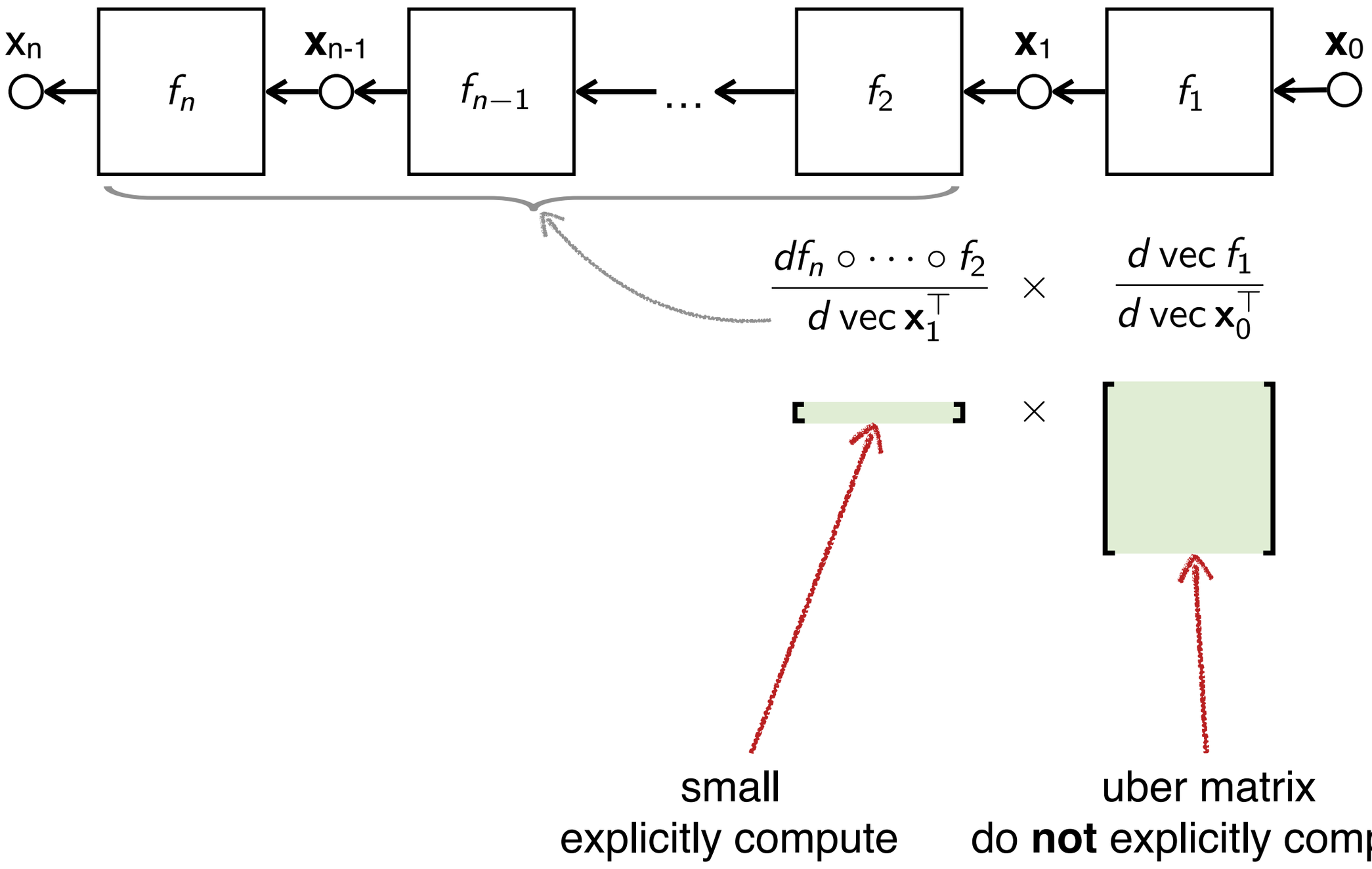
Backpropagation

Assume that x_n is a scalar (e.g. loss)



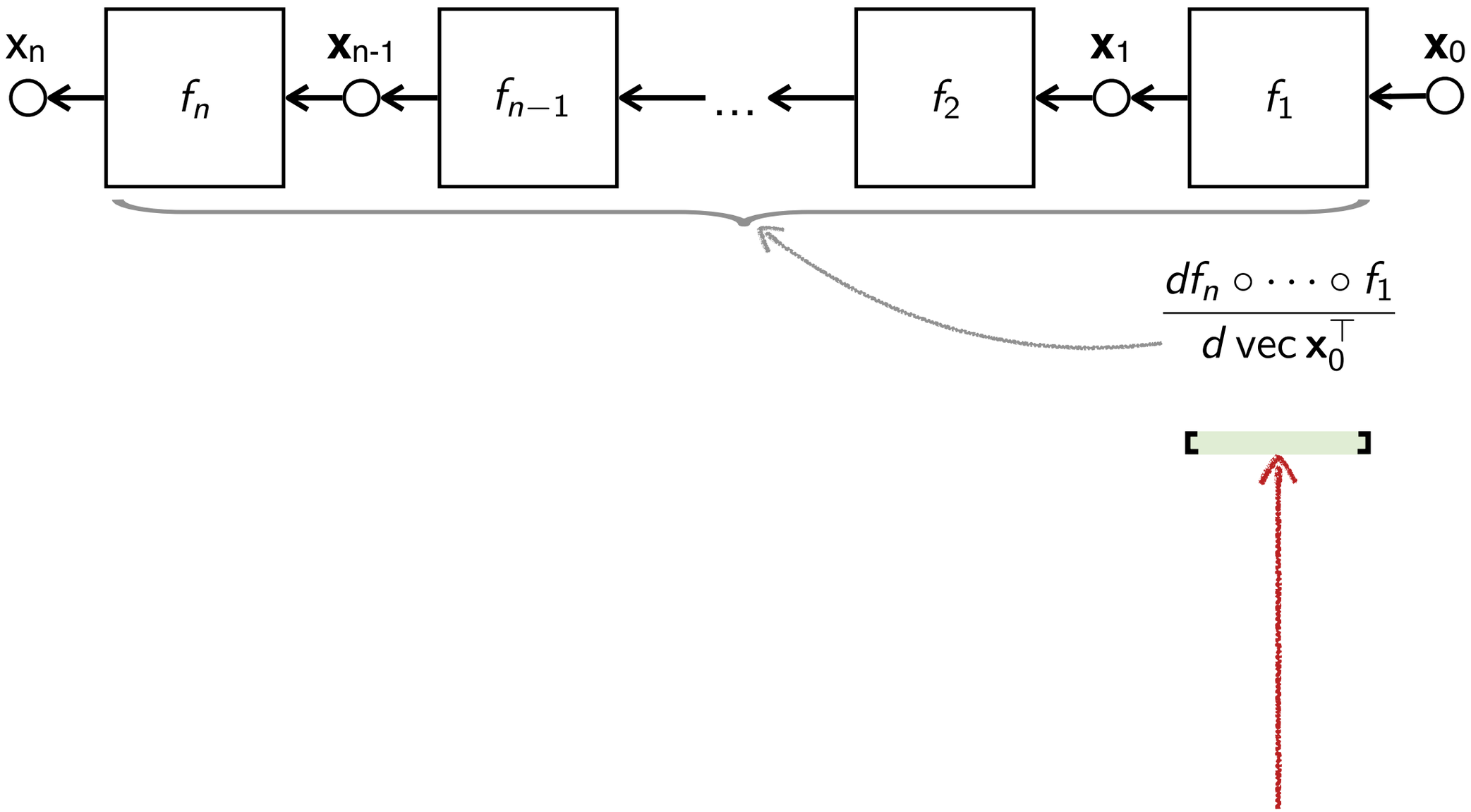
Backpropagation

Assume that x_n is a scalar (e.g. loss)



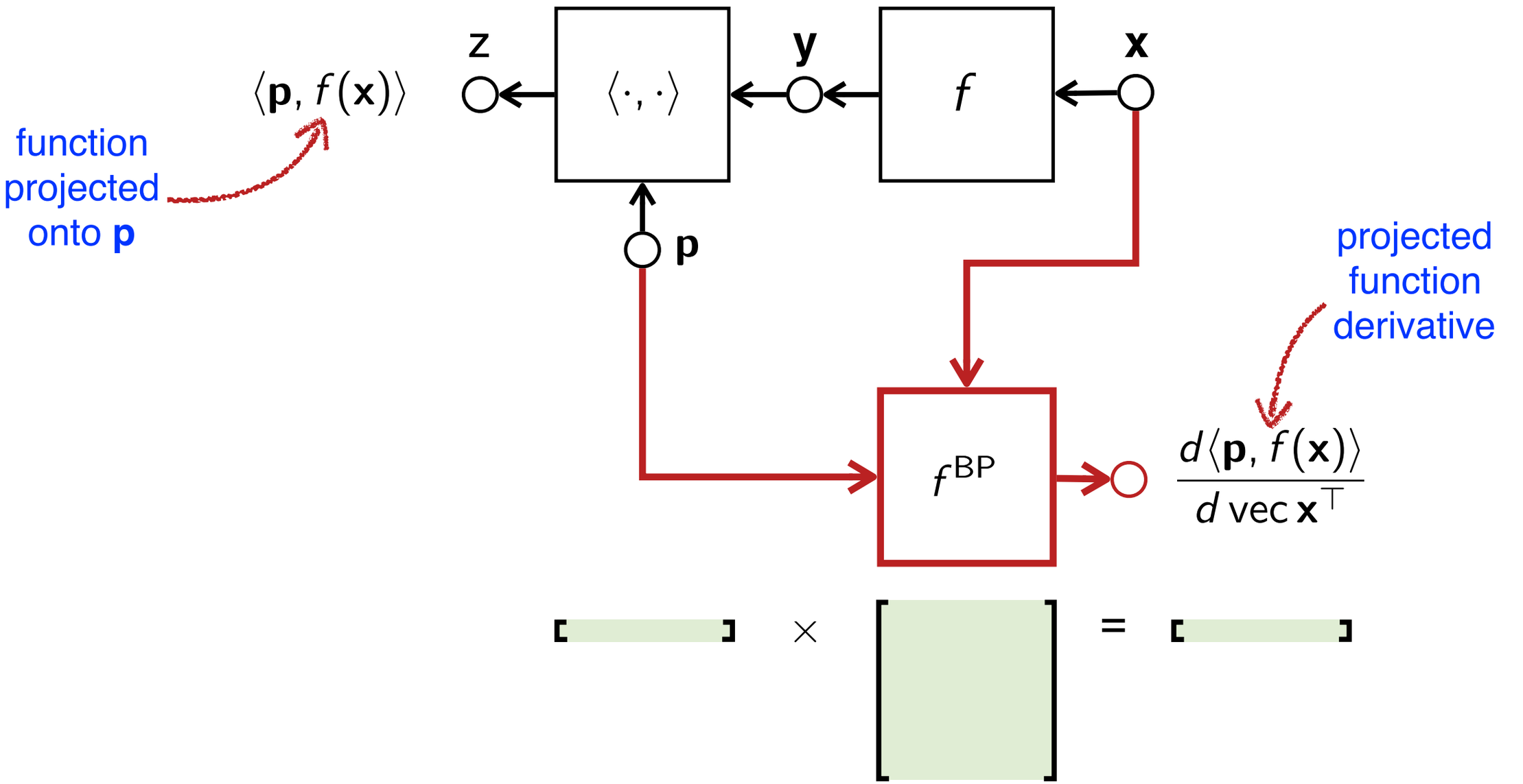
Backpropagation

Assume that x_n is a scalar (e.g. loss)



Projected function derivative

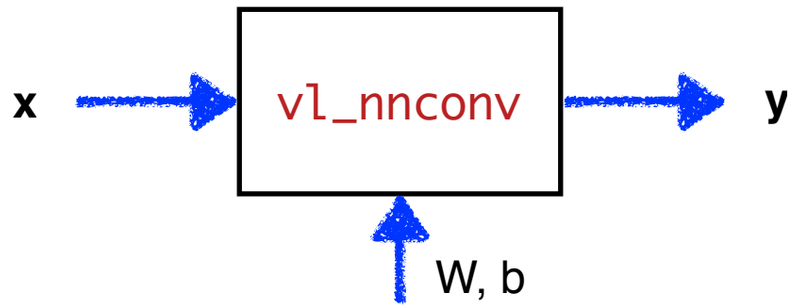
The “BP-reversed” layer



An “equivalent circuit” is obtained by introducing a transposed function f^T

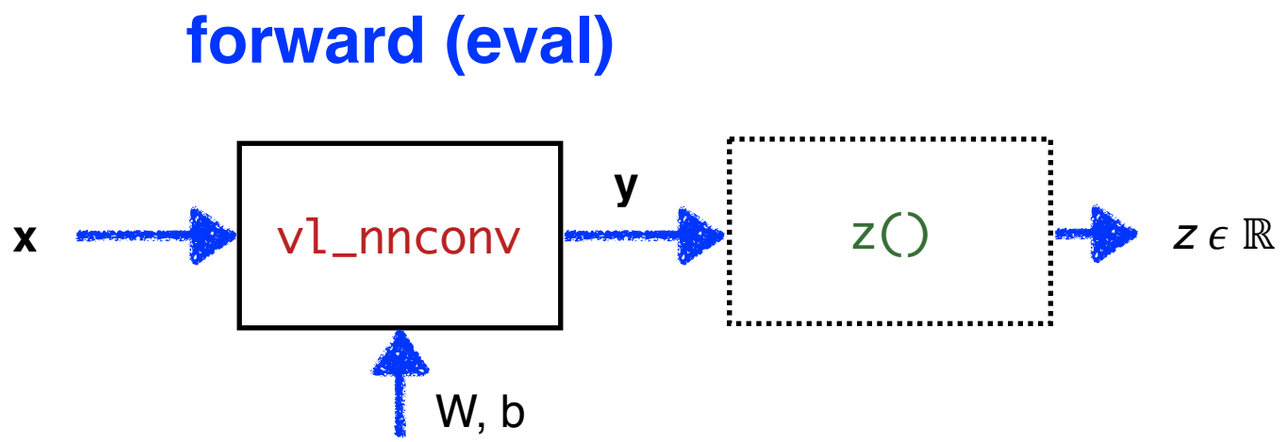
Anatomy of a building block

forward (eval)



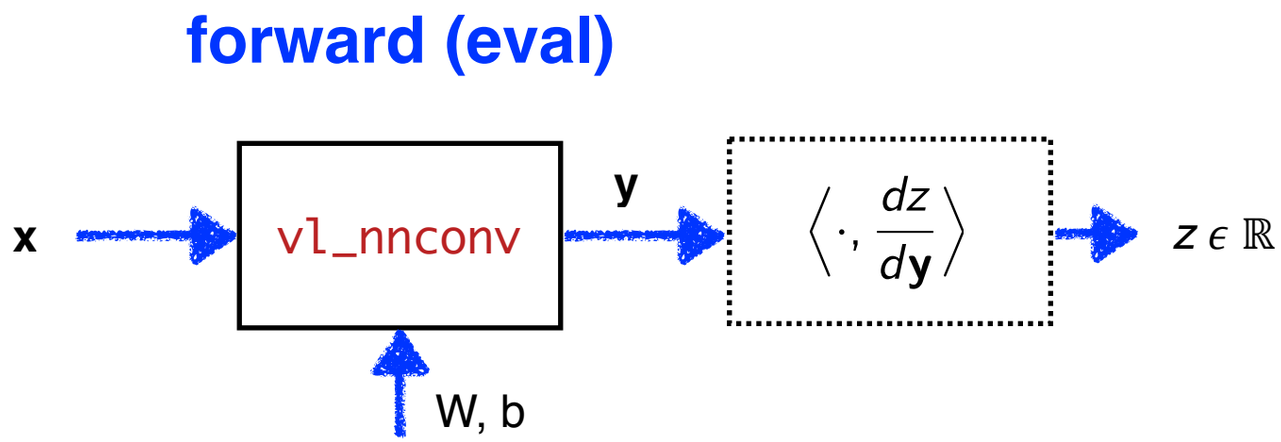

```
y = vl_nnconv(x, W, b)
```

Anatomy of a building block




```
y = vl_nnconv(x, W, b)
```

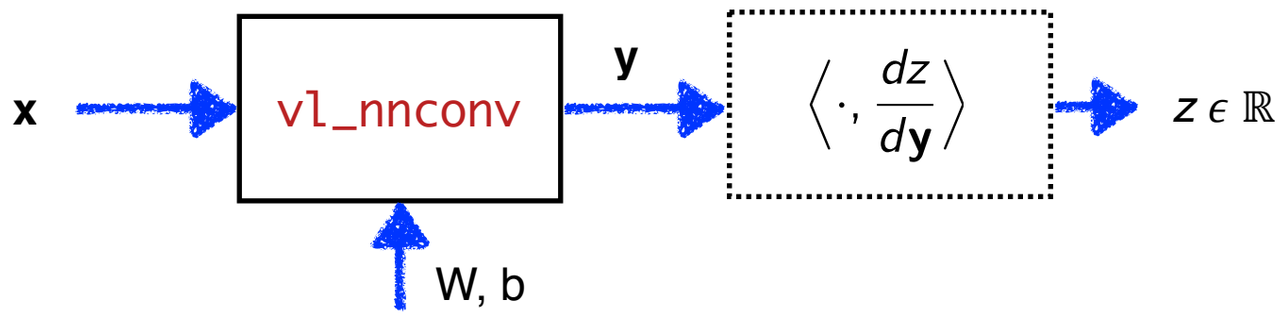
Anatomy of a building block




```
y = vl_nnconv(x, W, b)
```

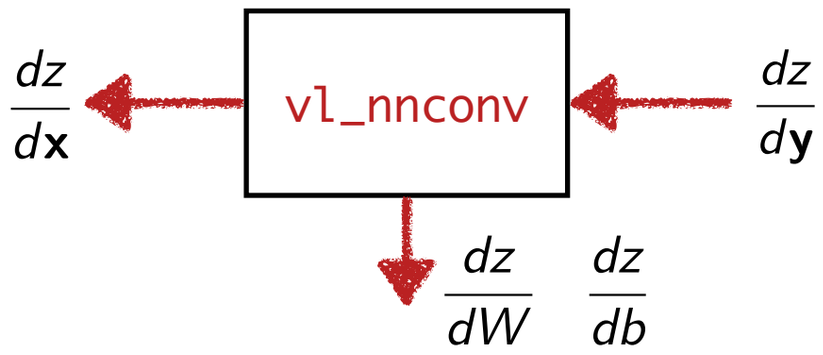
Anatomy of a building block

forward (eval)




```
y = vl_nnconv(x, W, b)
```

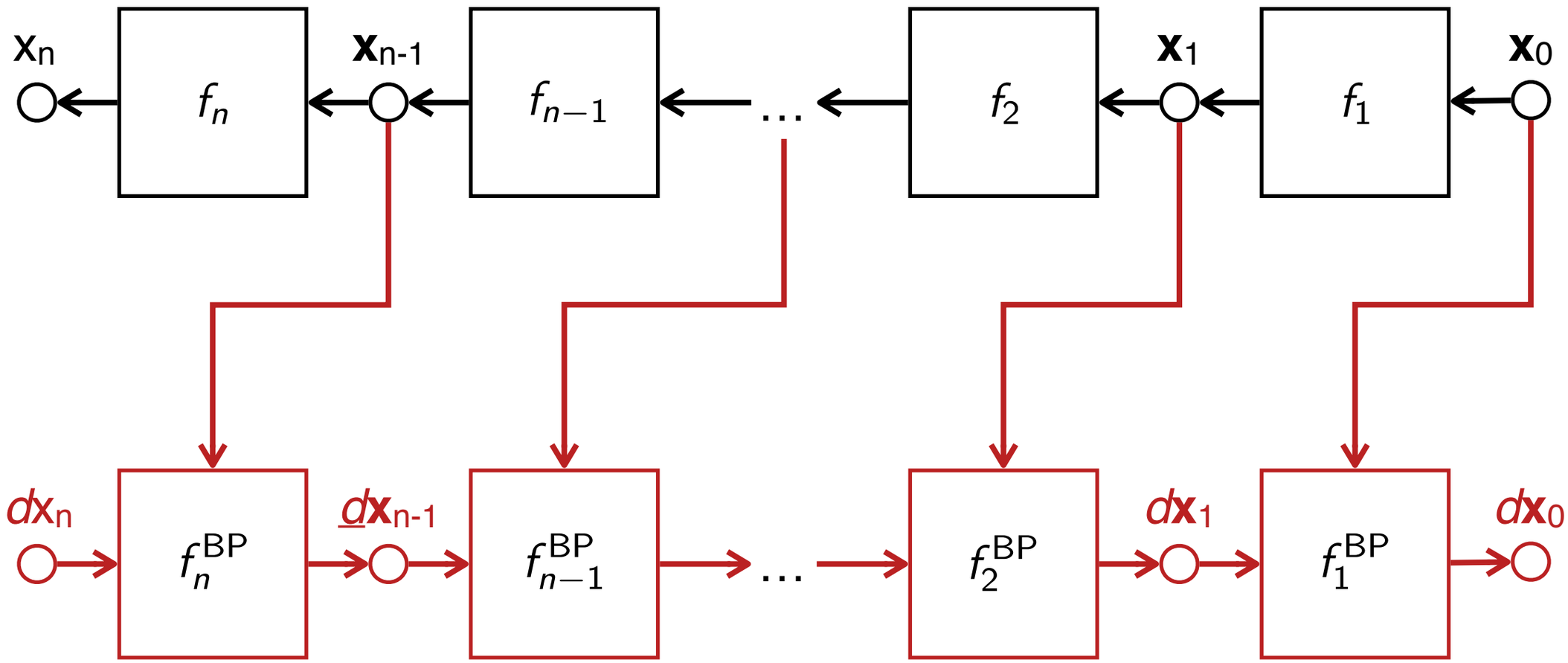
backward (backprop)




```
dzdx = vl_nnconv(x, W, b, dzdy)
```

Backpropagation network

BP induces a “reversed” network

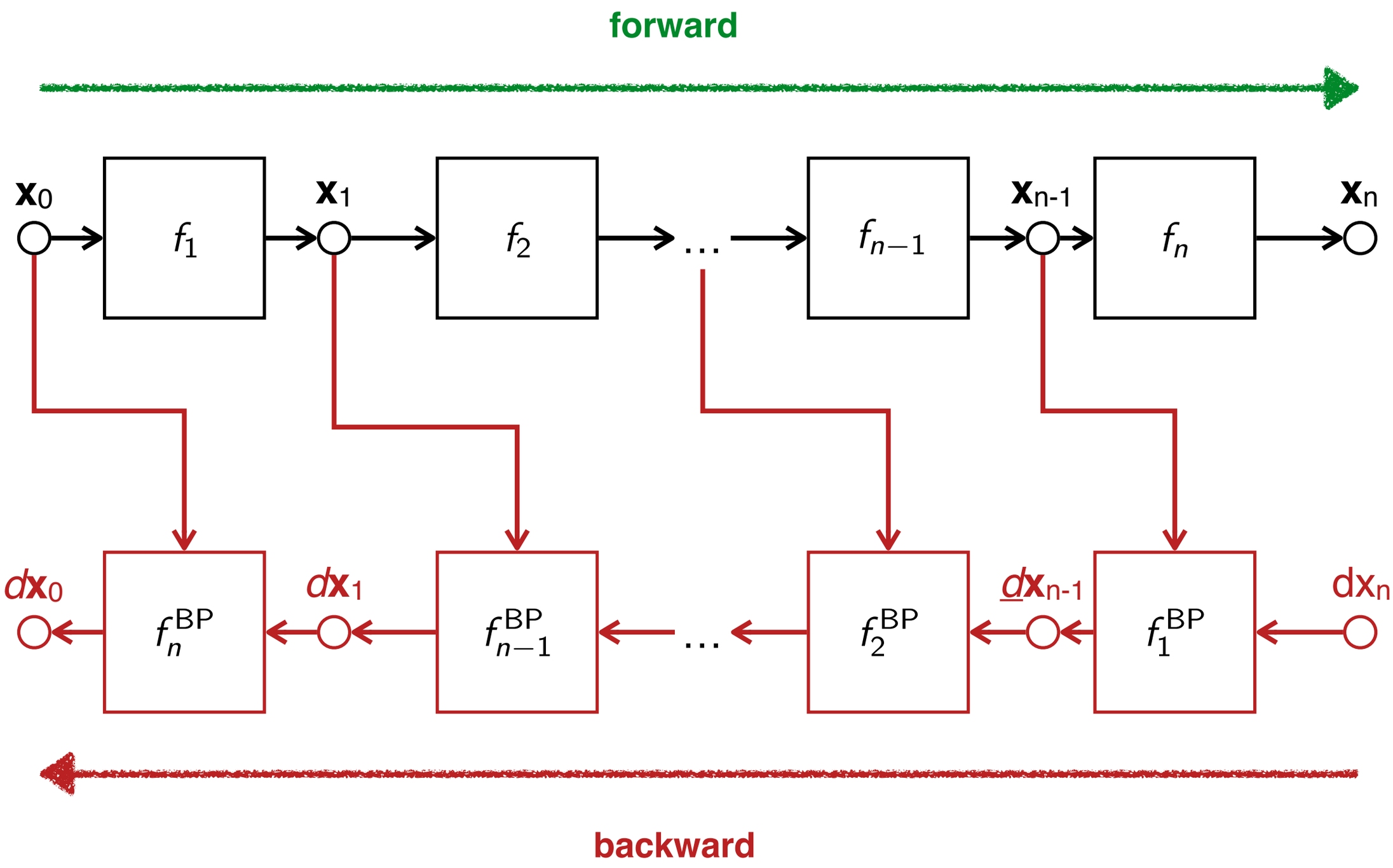


where $d\mathbf{x}_i = \dfrac{df_n \circ \cdots \circ f_{i+1}}{d \operatorname{vec} \mathbf{x}_i}$

Note: the BP network is linear in $d\mathbf{x}_1, \ldots, d\mathbf{x}_{n-1}, d\mathbf{x}_n$. Why?

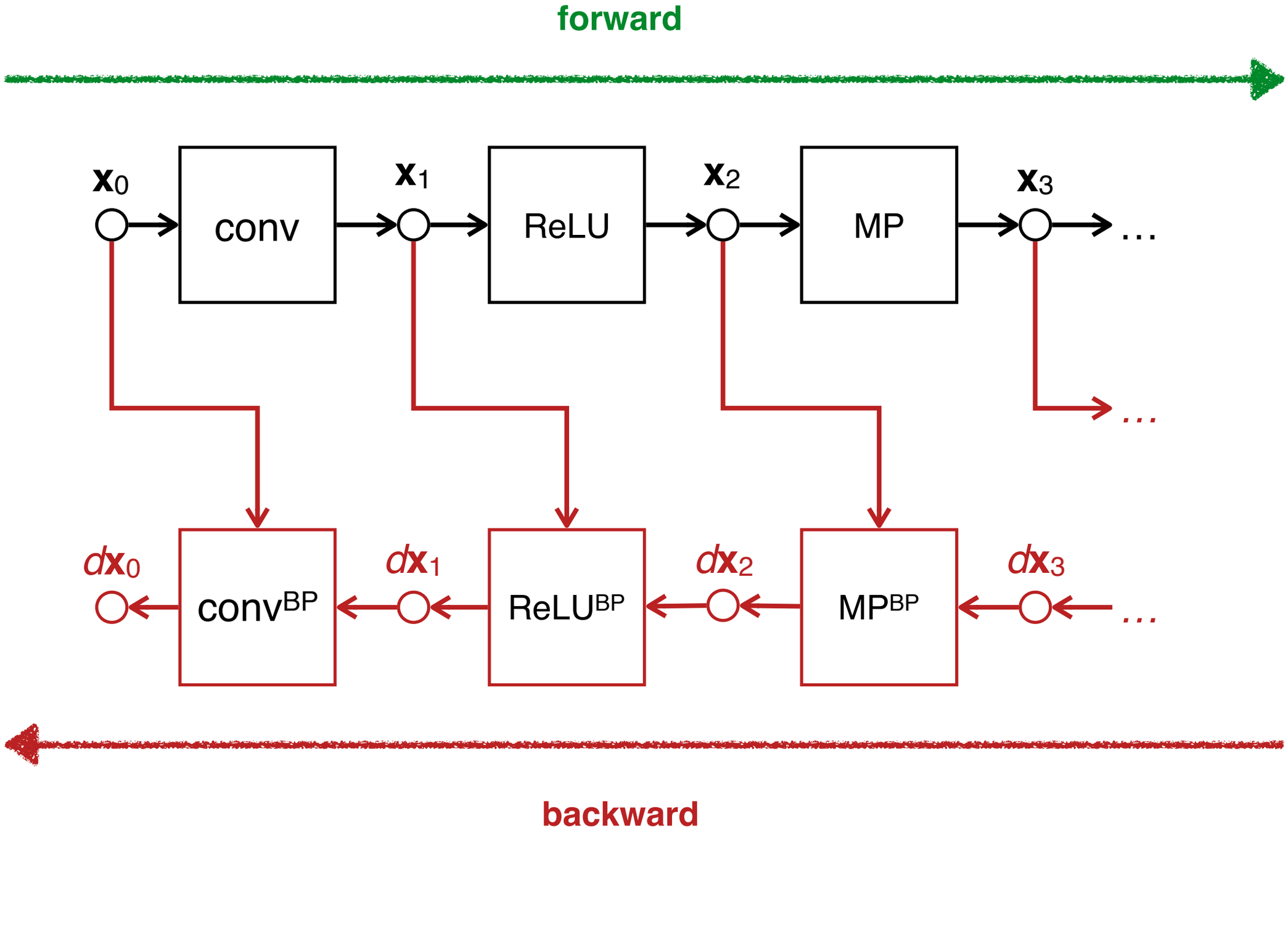
Backpropagation network

BP induces a “transposed” network



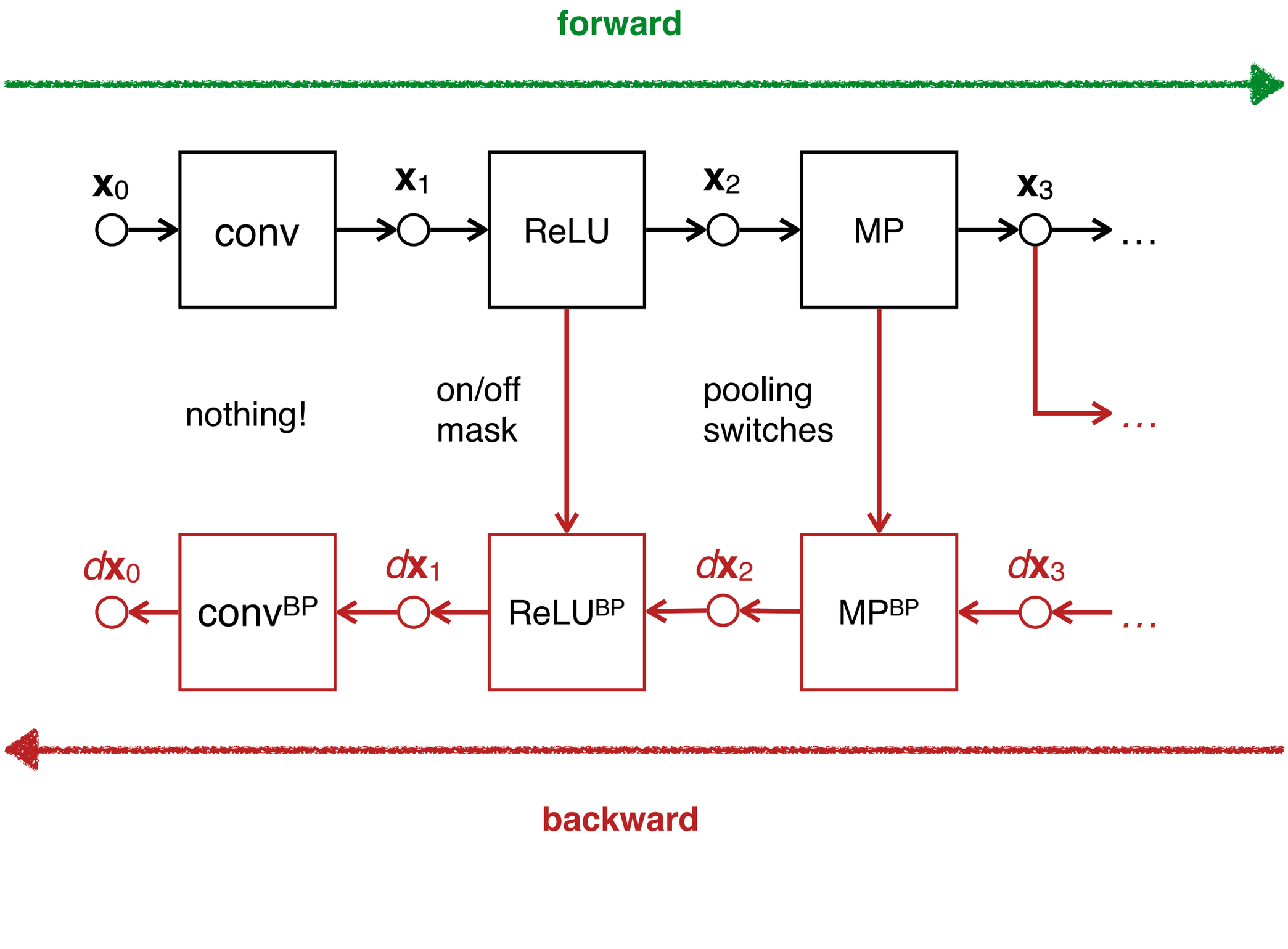
Backpropagation network

Conv, ReLU, MP and their transposed blocks



Sufficient statistics and bottlenecks

Usually much less information is needed



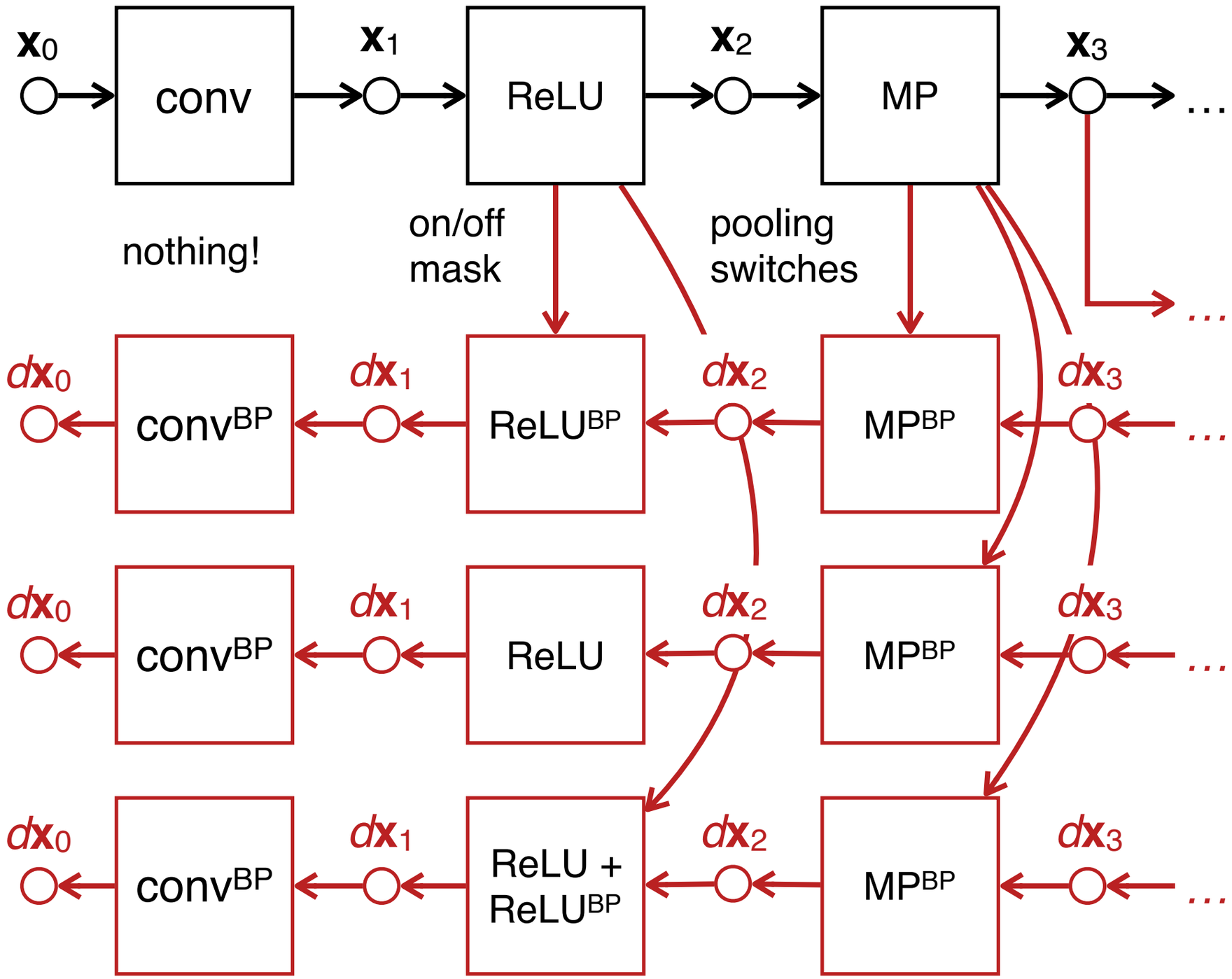
Three visualisation techniques

Modified backpropagation networks

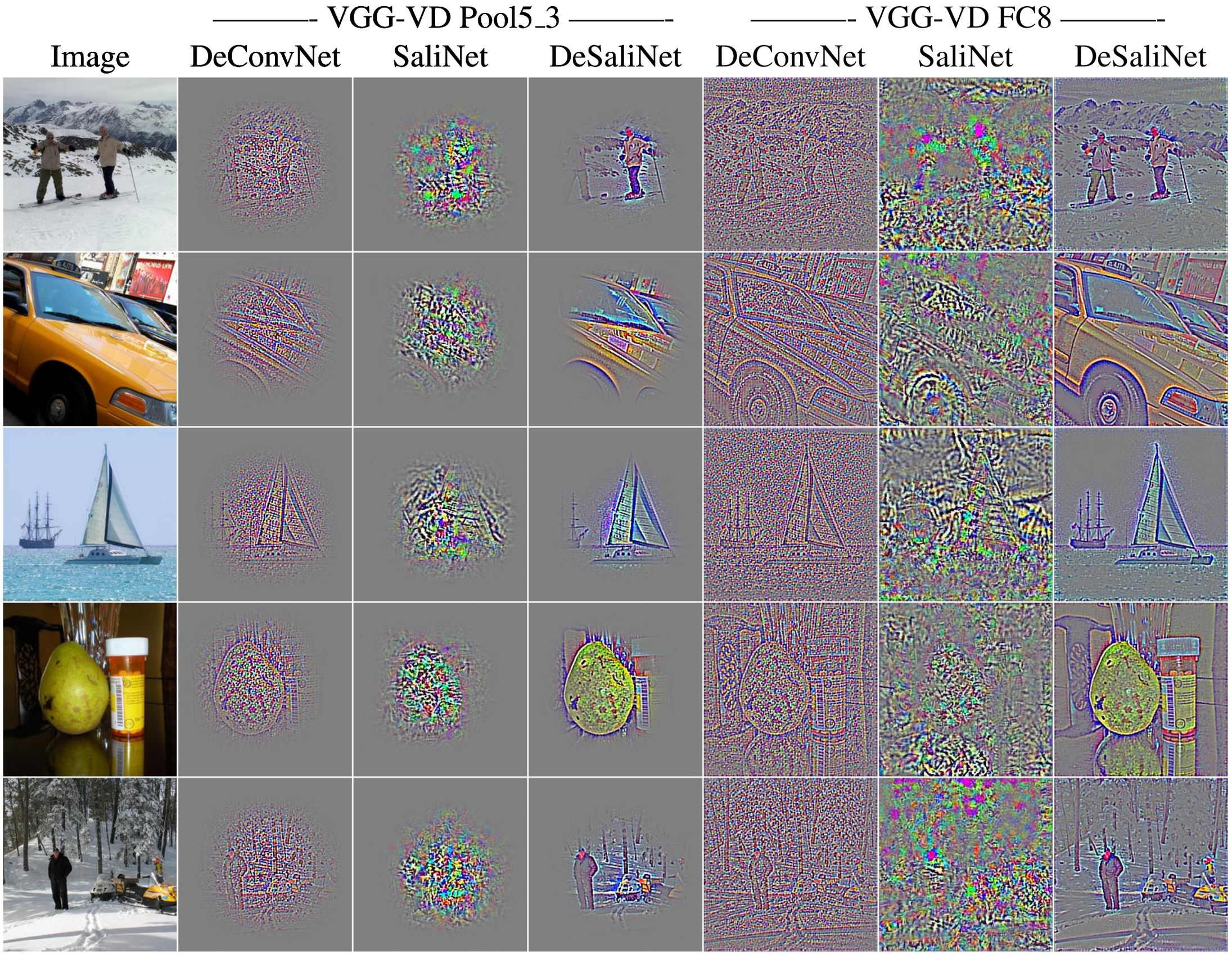
SaliNet
Equiv. to
[Simonyian et al. 14]

DeConvNet
Same as
[Zeiler Fergus 14]

DeSaliNet
[Mahendran Vedaldi 16,
Springenberg et al. 15]

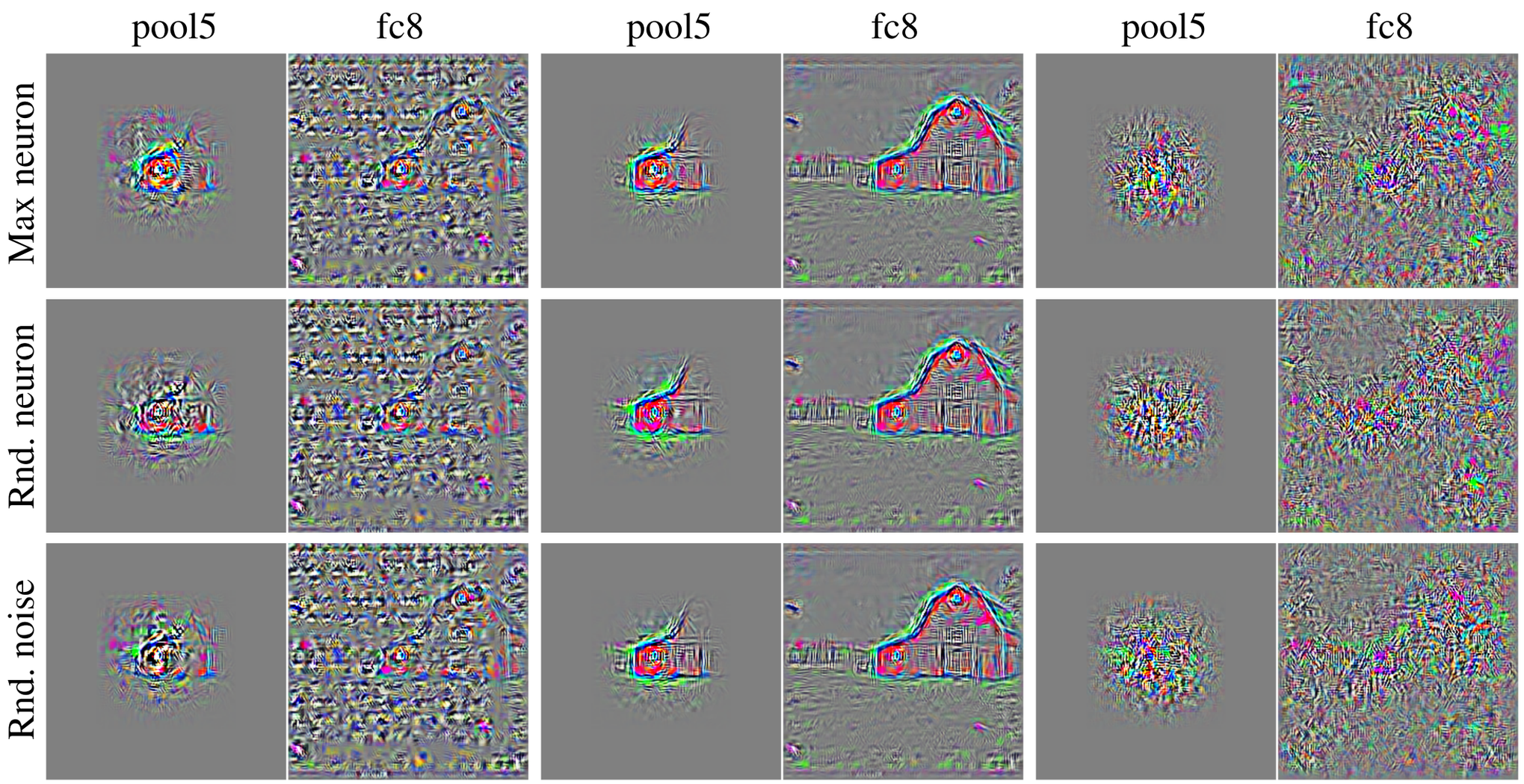


Results


VGG-VD Pool5_3
VGG-VD FC8
Image
DeConvNet
SaliNet
DeSaliNet
DeConvNet
SaliNet
DeSaliNet

Key limitation of DeConvNets and similar

They are largely *not* neuron selective



Good for saliency

Bad for studying individual neurons
(use inversion, act. max., etc. instead)

Understanding visual representations

Intro

Visualizing representations

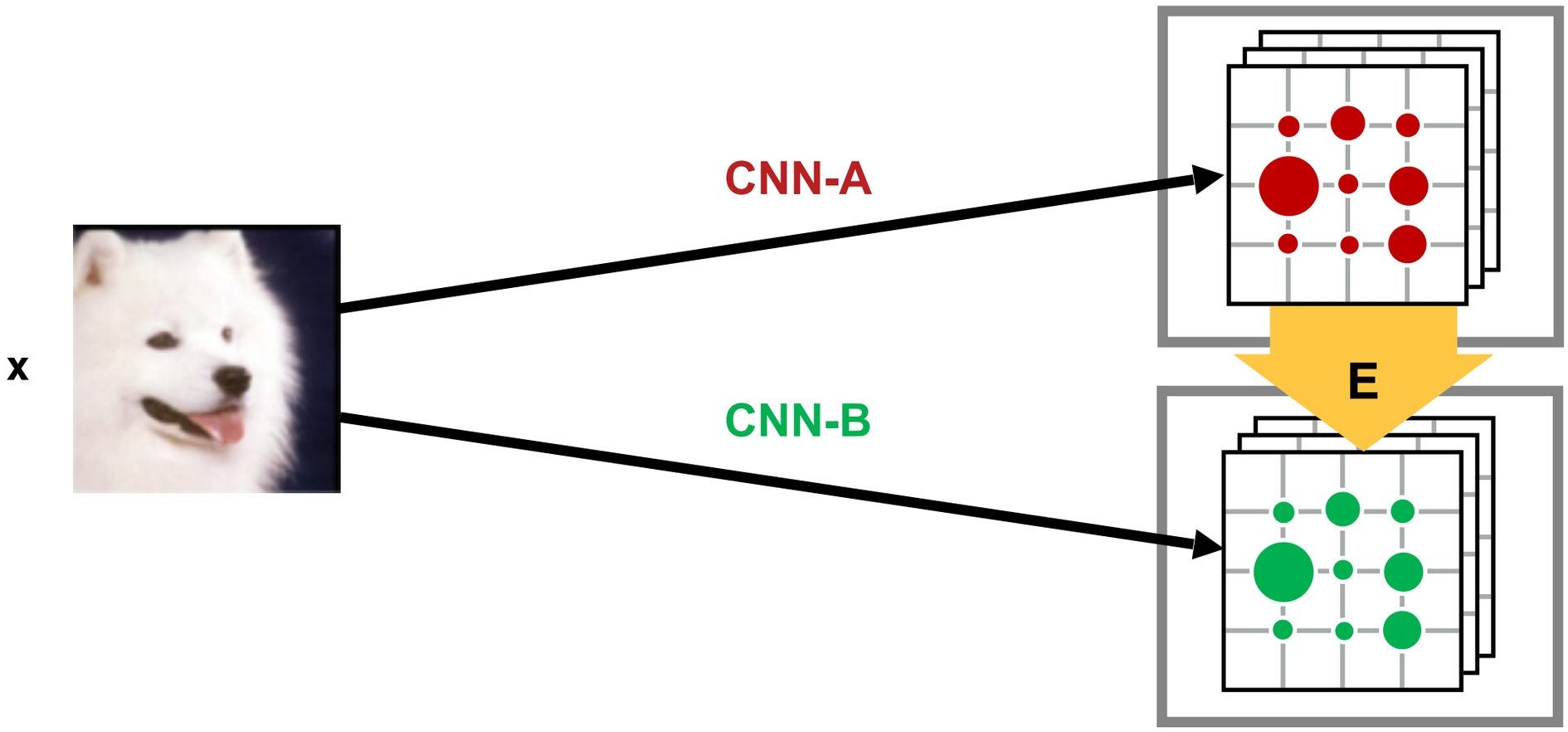
Backpropagation networks and “deconvolution”

Representations: equivalence & transformations

When are two representations the same?

Learning representations means that there is an endless number of them

Variants obtained by learning on different datasets, or different local optima

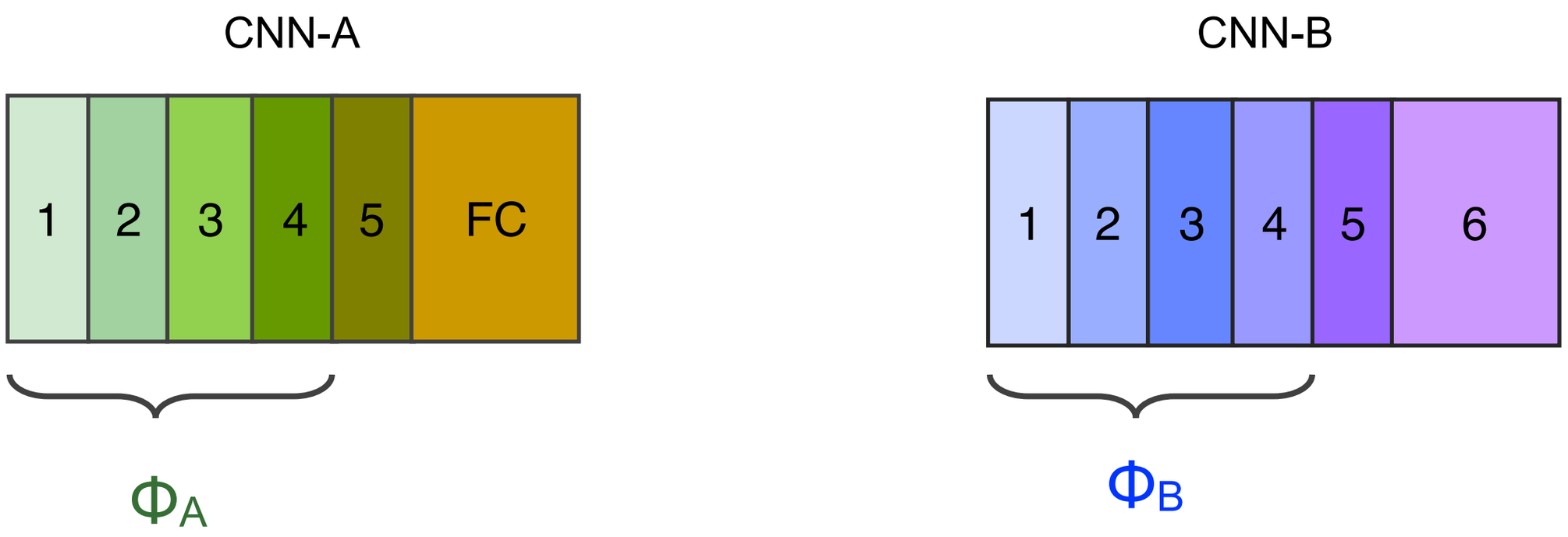


Equivalence

$\Phi_B(\mathbf{x}) = E\ \Phi_A(\mathbf{x})$

Equivalence

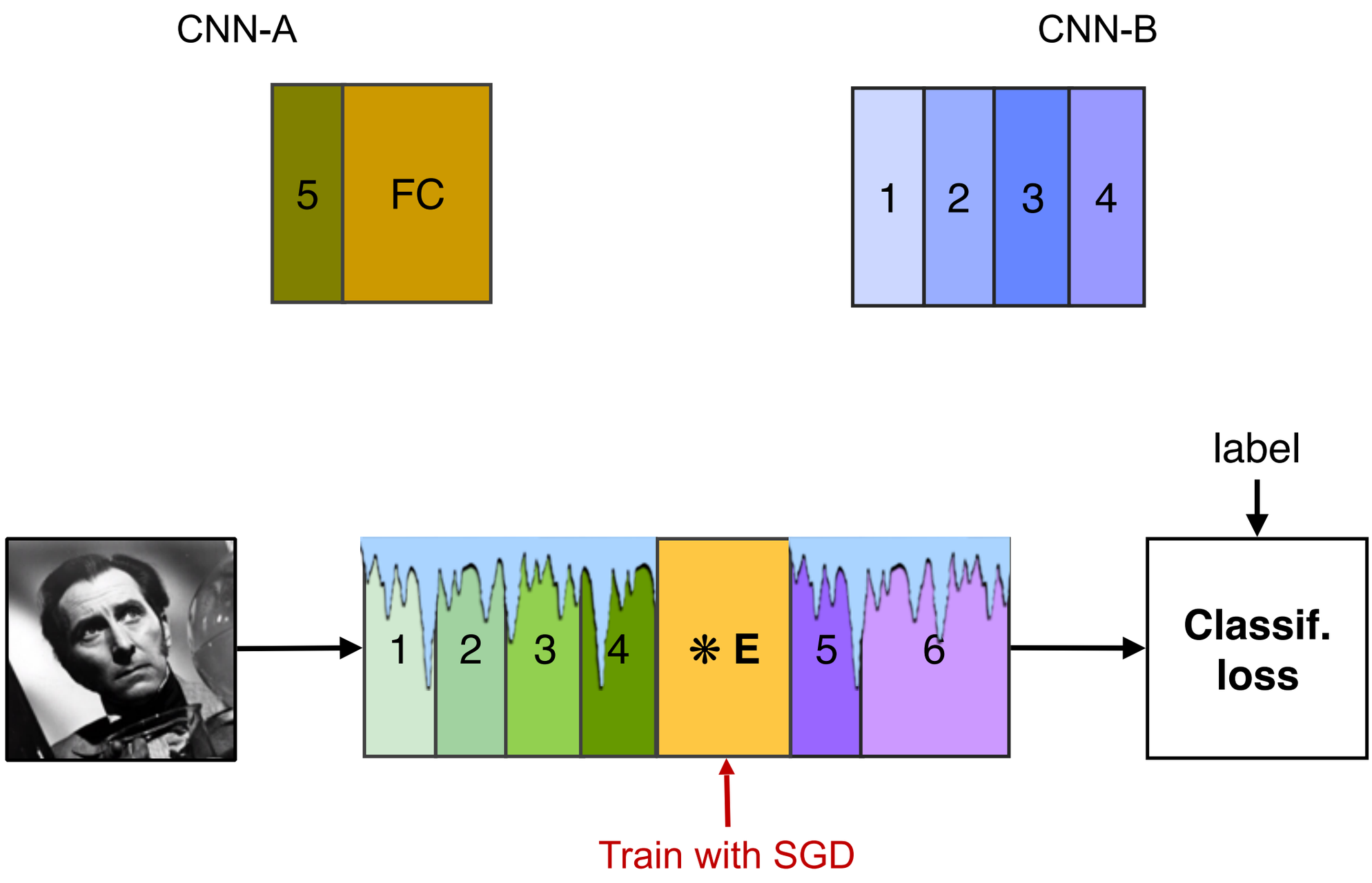
AlexNet, same training data, different parametrization:



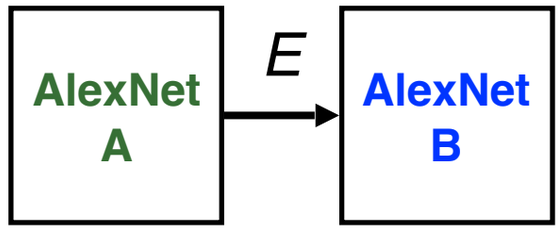
Are Φ_A and Φ_B equivalent ?

Equivalence

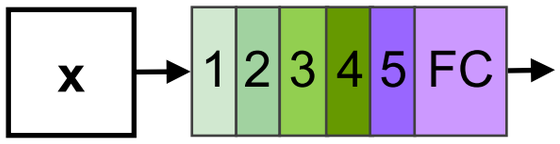
AlexNet, same training data, different parametrization:



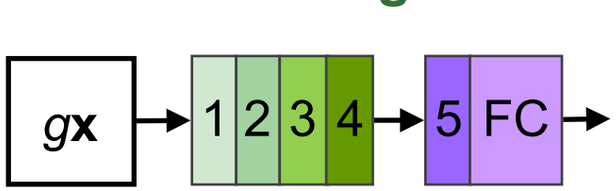
Equivalence with different random seeds



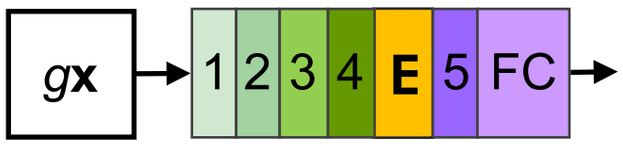
Baseline

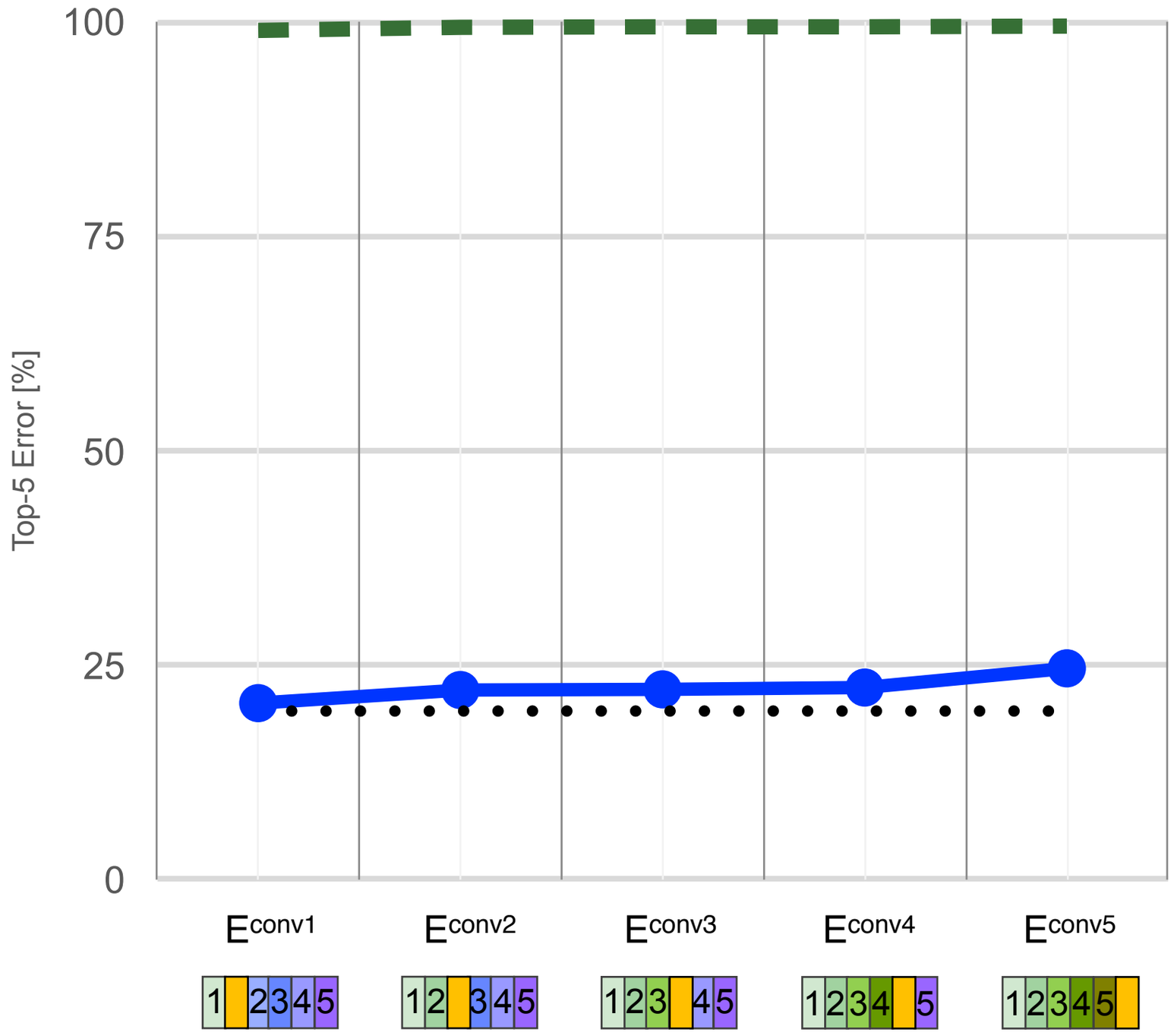


Before training



After training

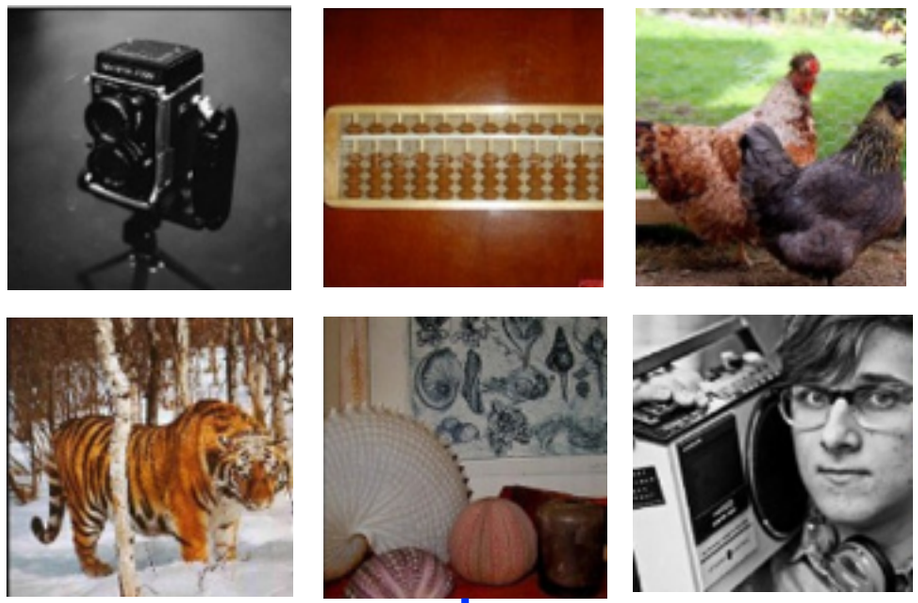




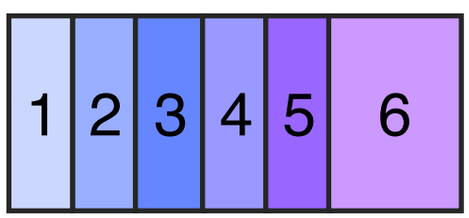
Equivalence of similar architecture

Train on two different datasets

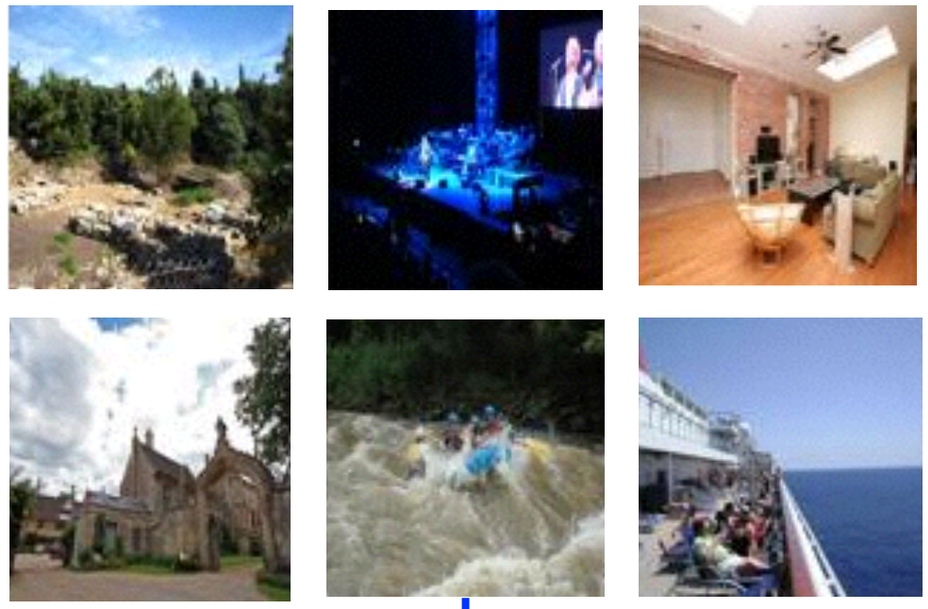
ILSVRC12 dataset



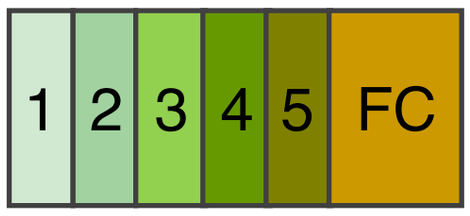
CNN-IMNET



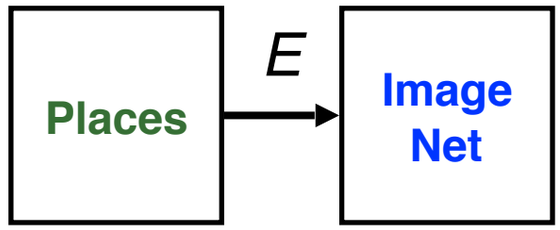
Places dataset



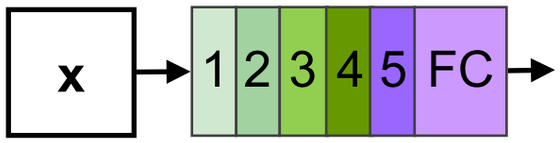
CNN-PLACES



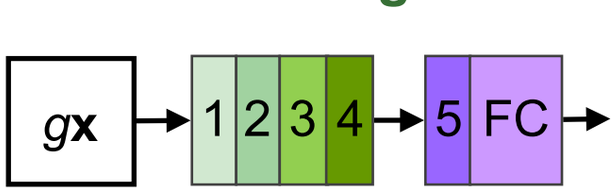
Equivalence with different training data



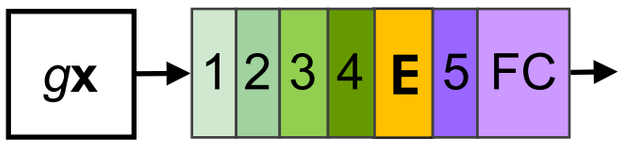
Baseline

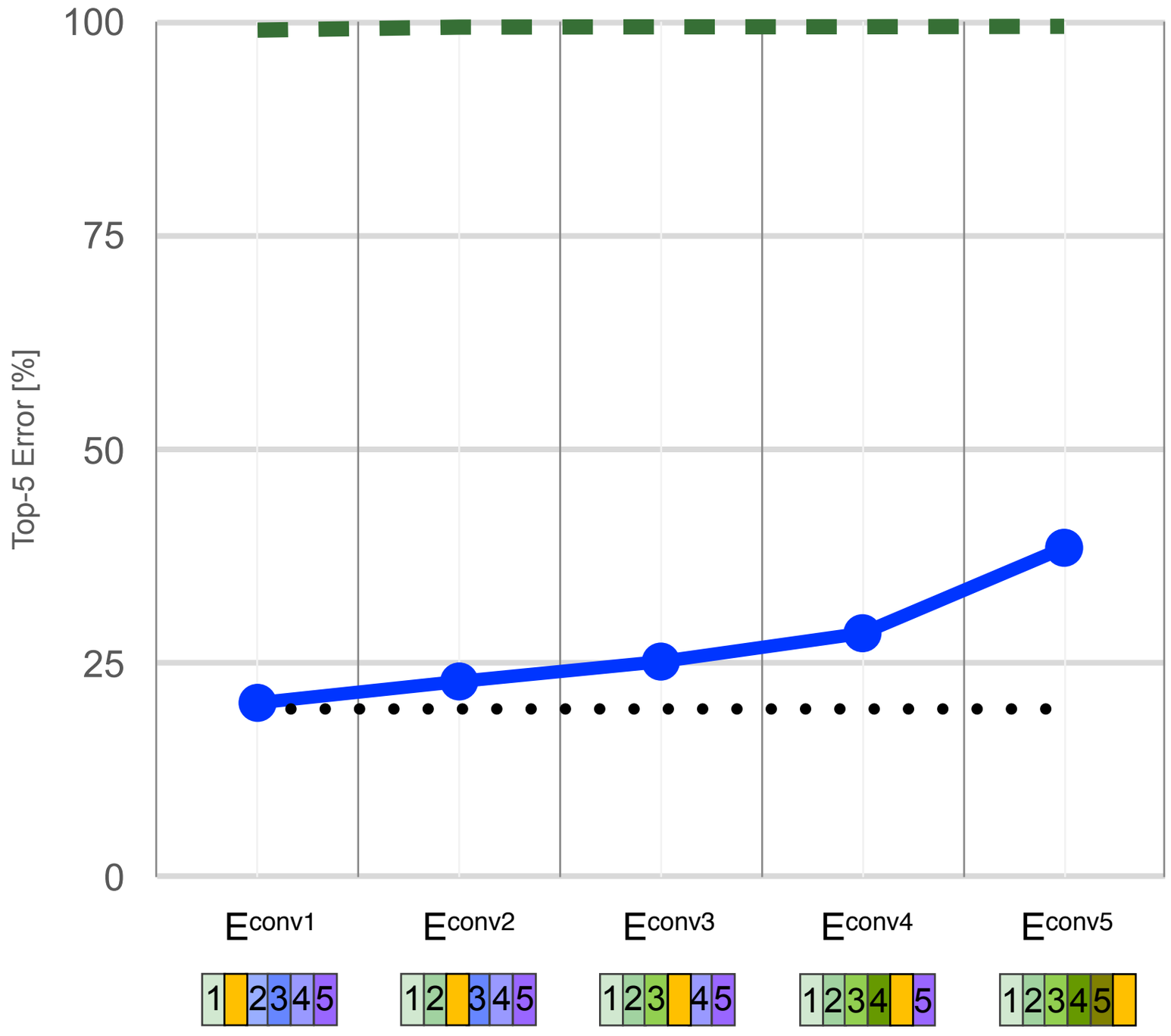


Naive stitching



Learned stitches



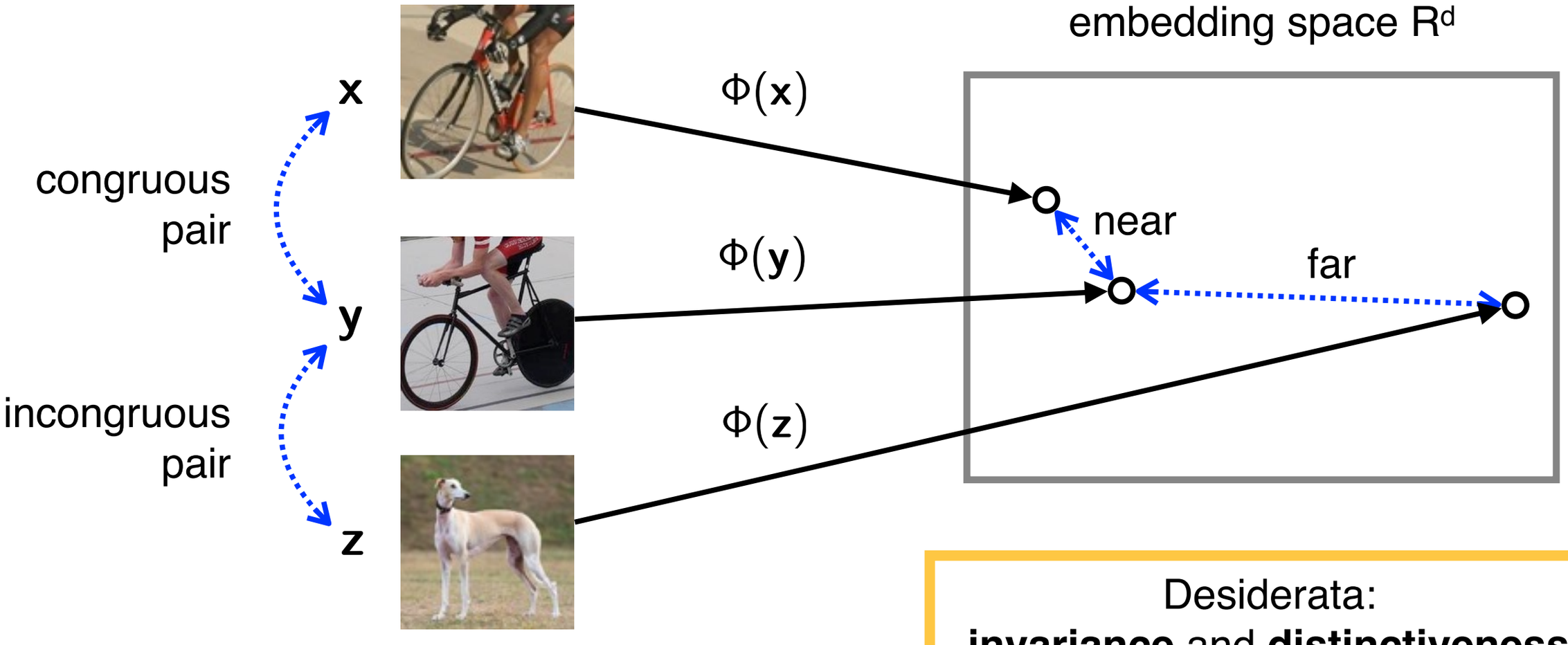


Meaningful representations

representation

Semantic similarity

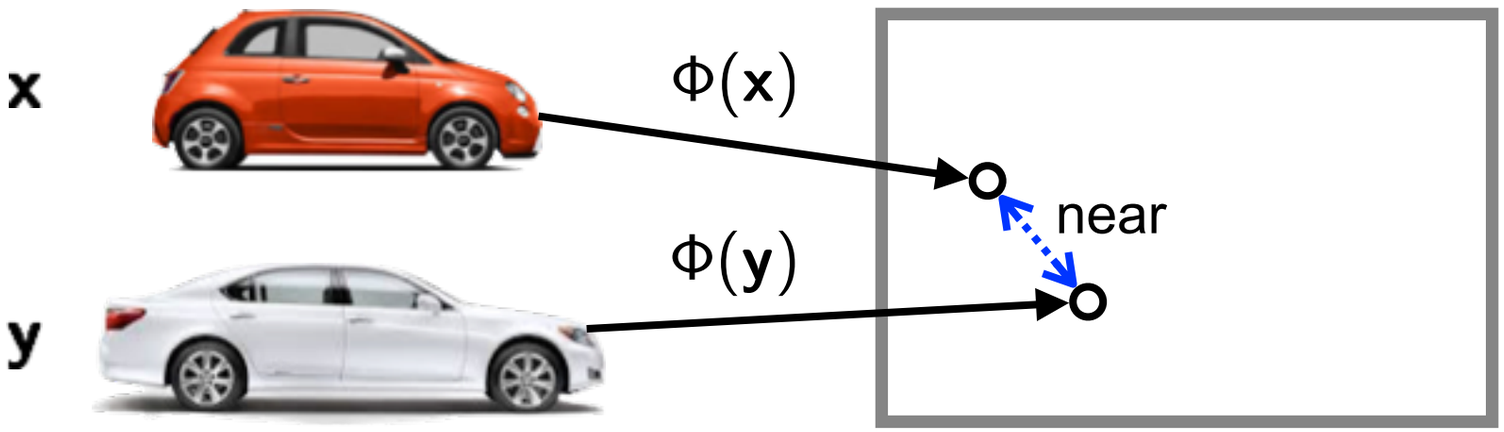
Vector similarity
(distance)



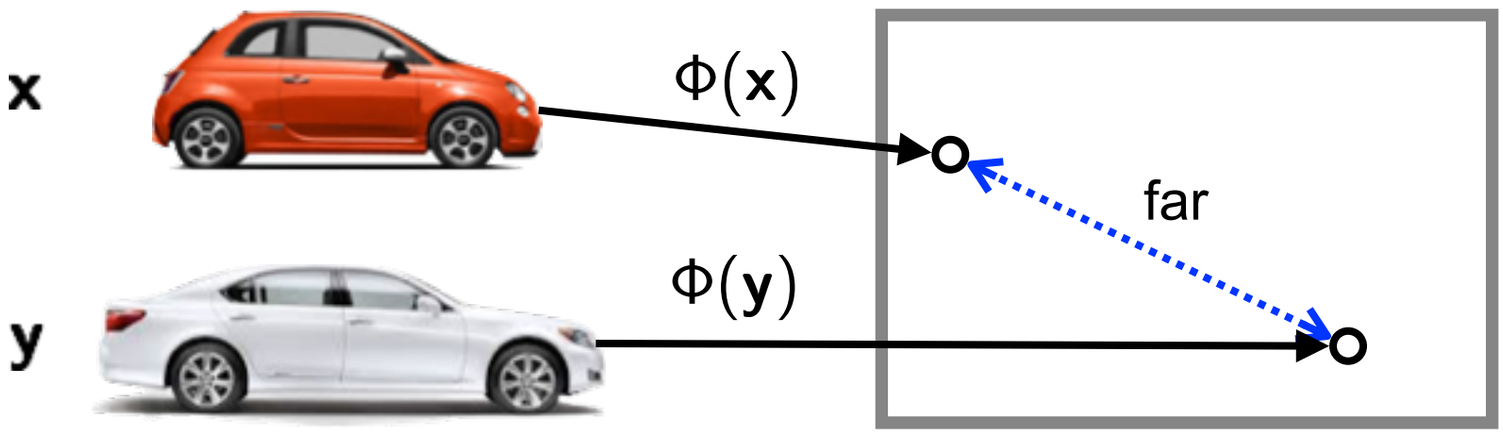
Desiderata:
invariance and **distinctiveness**

Invariance is task dependent

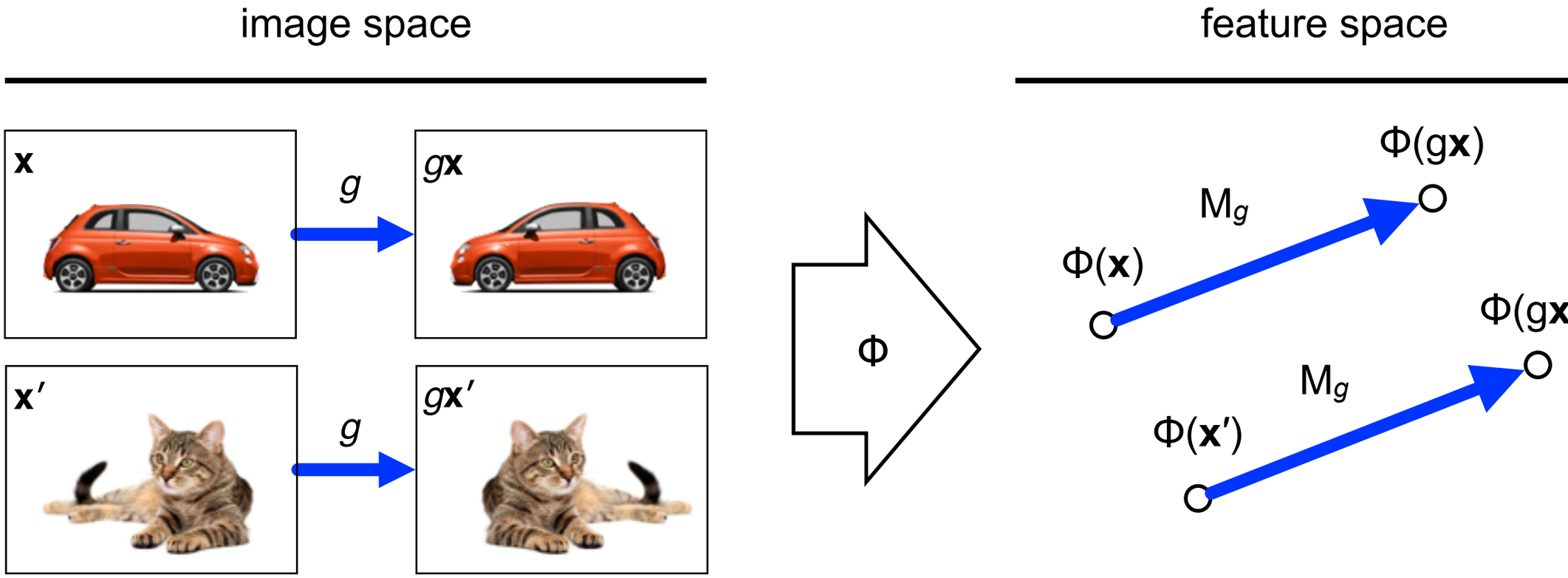
Are **x** and **y** the same category?



Are **x** and **y** the same colour?



Equivariance

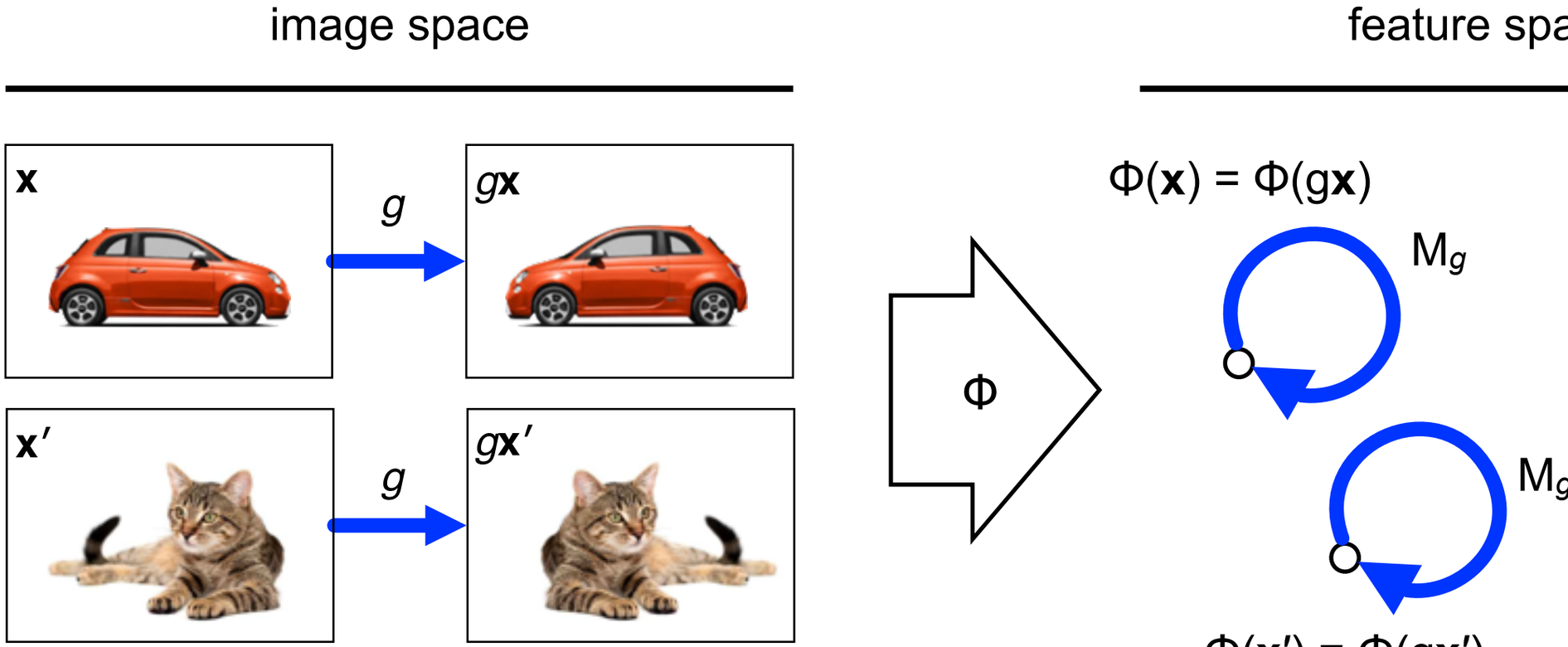


g is an image transformation

g is the corresponding feature transformation

$$\Phi(g\mathbf{x}) = M_g \, \Phi(\mathbf{x})$$

Invariance

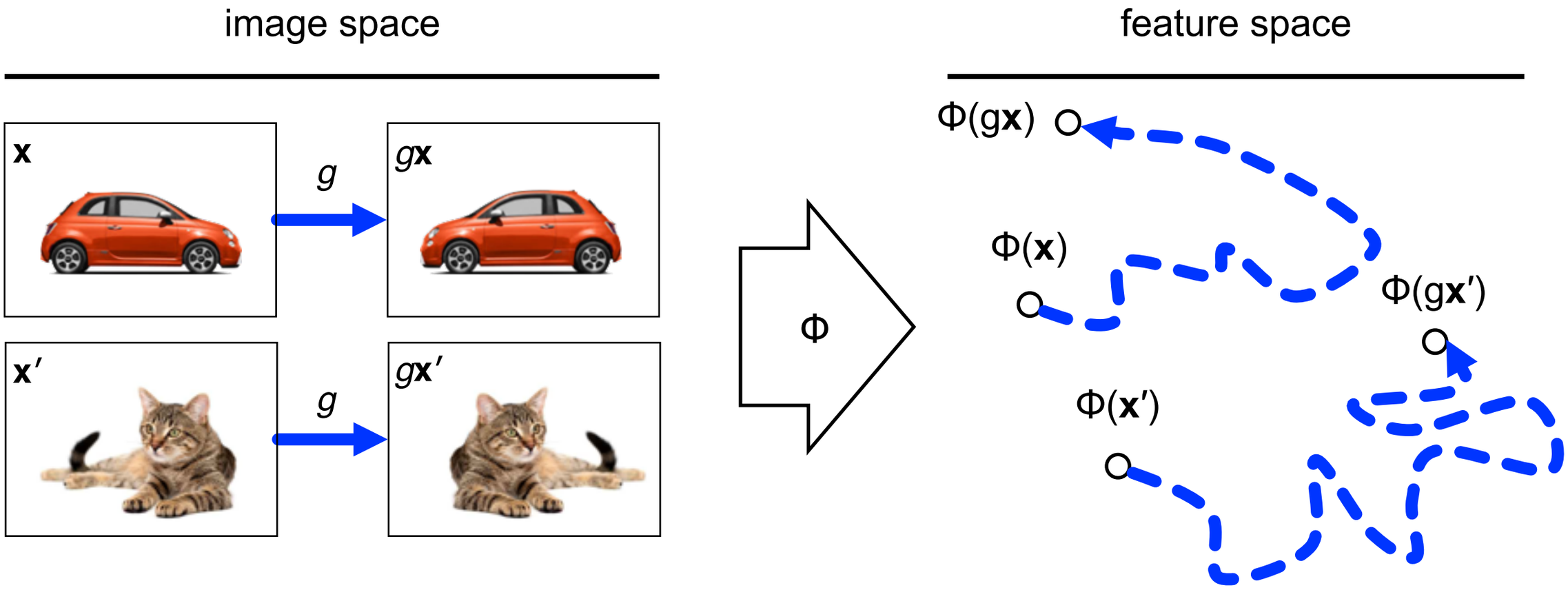


g is an image transformation

g is the corresponding feature transformation

the map M_g is the identity

No equivariance



g is an image transformation

g is the corresponding feature transformation

the map M_g does not exists (or is intractable)

Representations and transformations

image space

feature space

g is an image transformation

Equivariance
There is a (simple) M_g

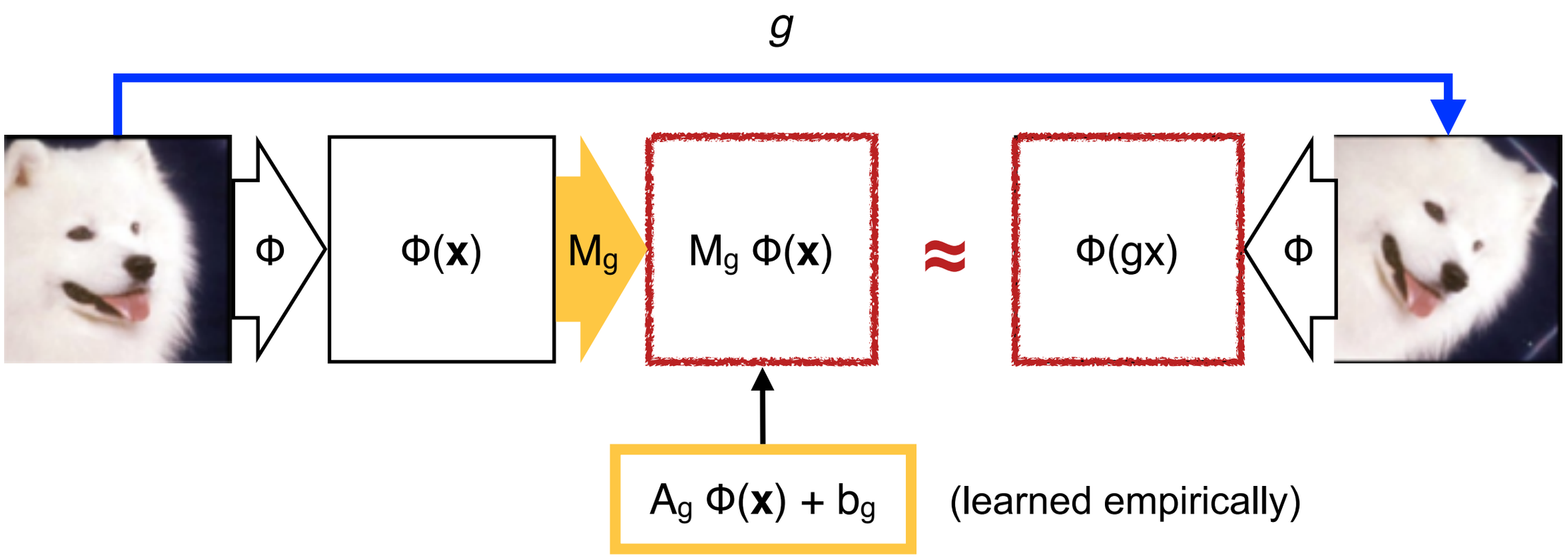
Invariance
M_g is the identity

Neither
There is no (simple) M_g

$$\Phi(g\mathbf{x}) = M_g\,\Phi(\mathbf{x})$$

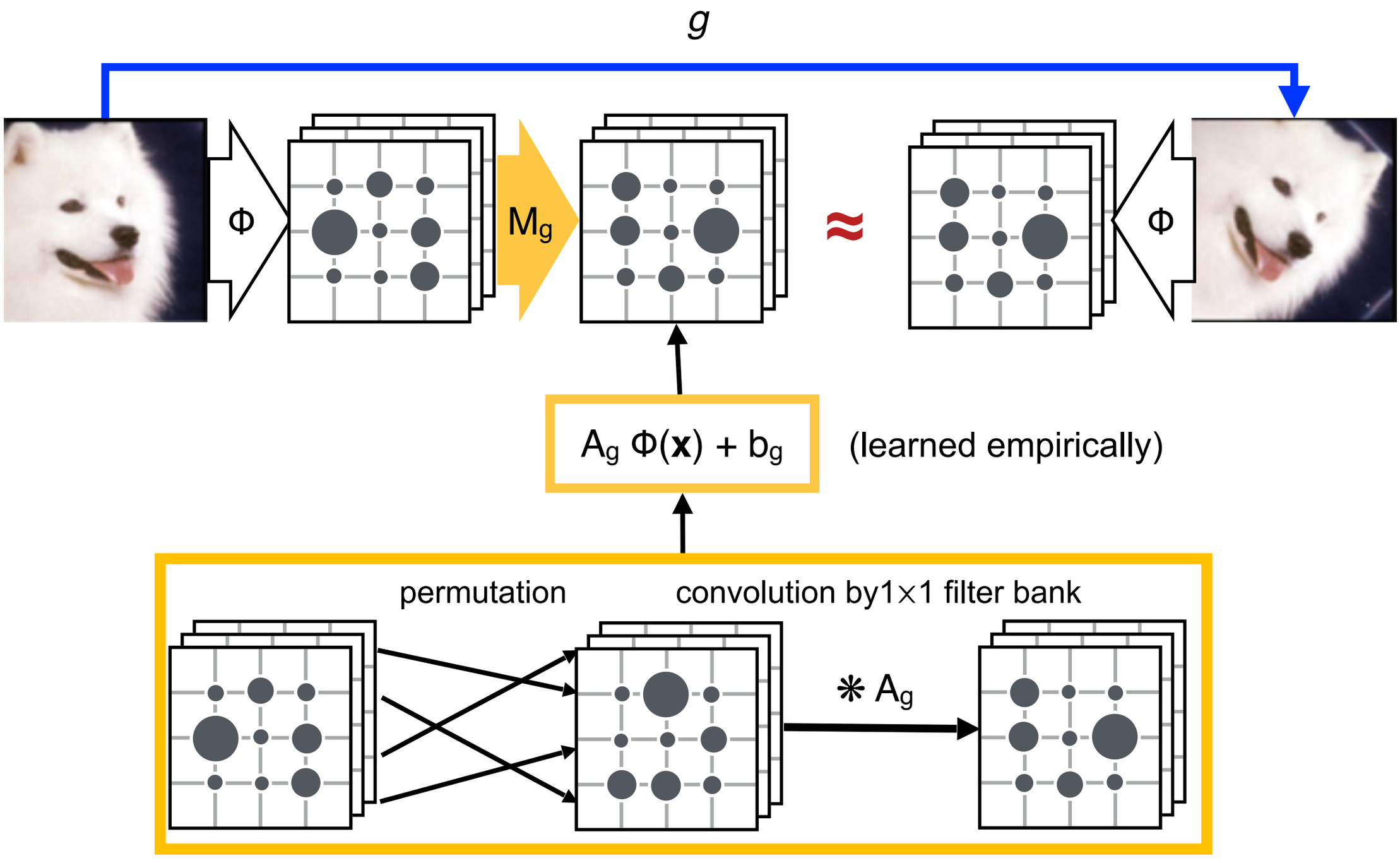
An empirical test of equivariance

Regularized linear regression



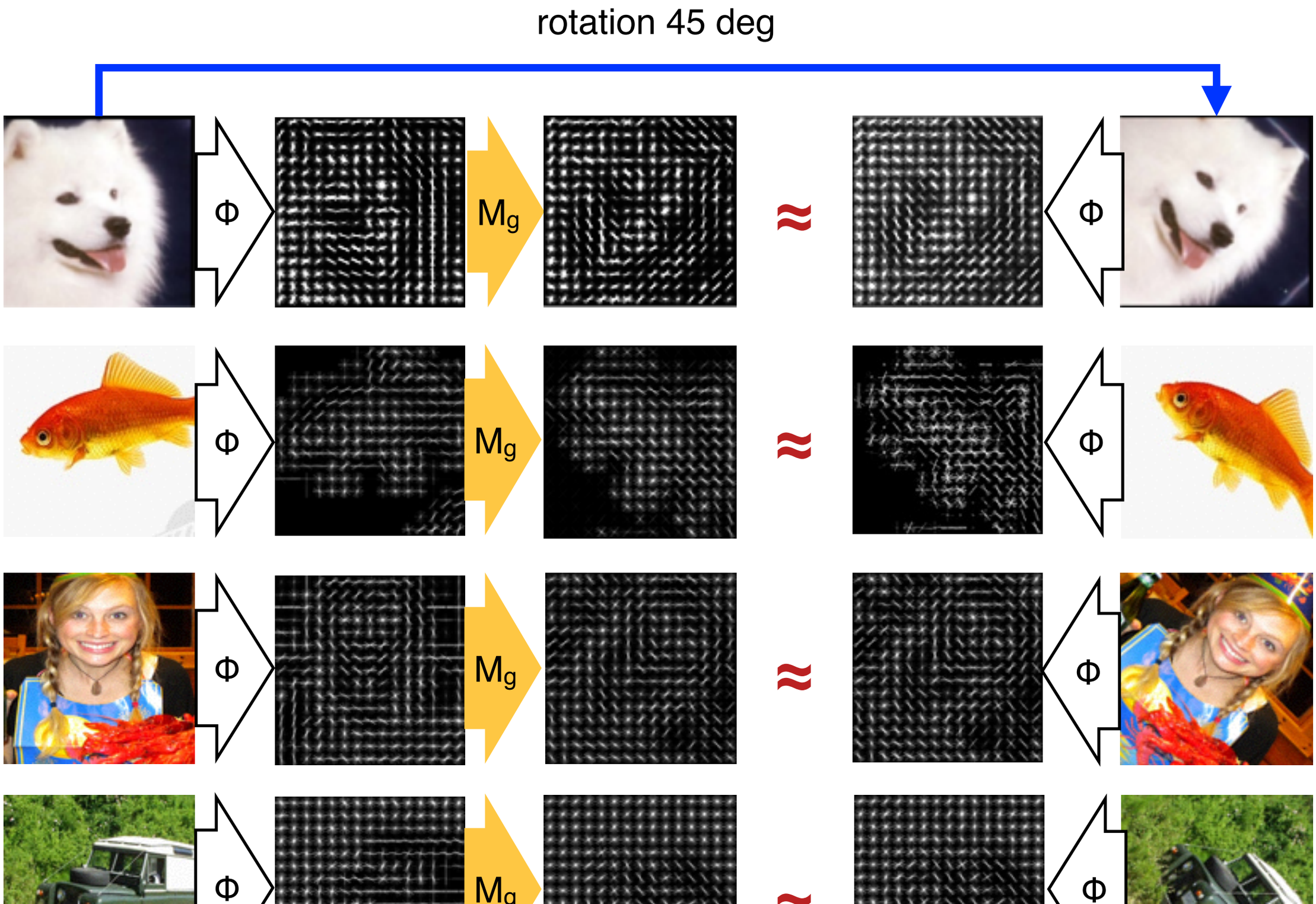
An empirical test of equivariance

Regularized linear regression



An empirical test of equivariance

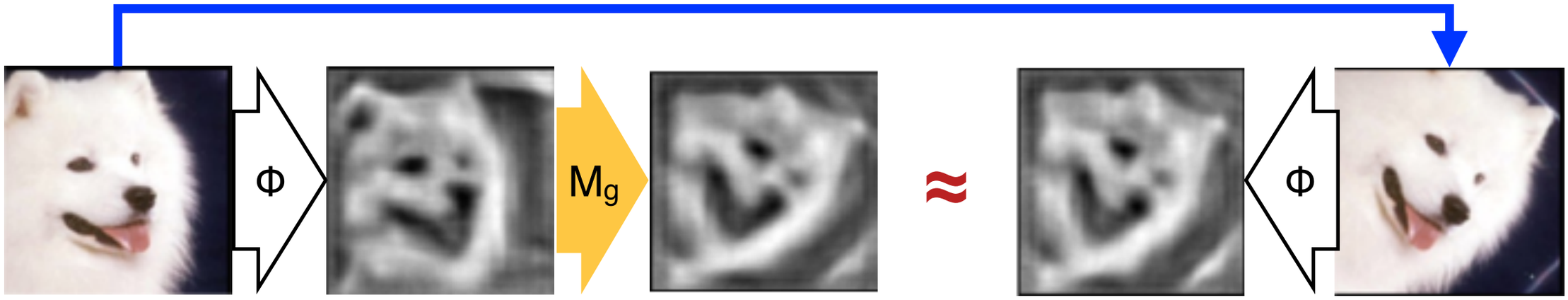
HOG features

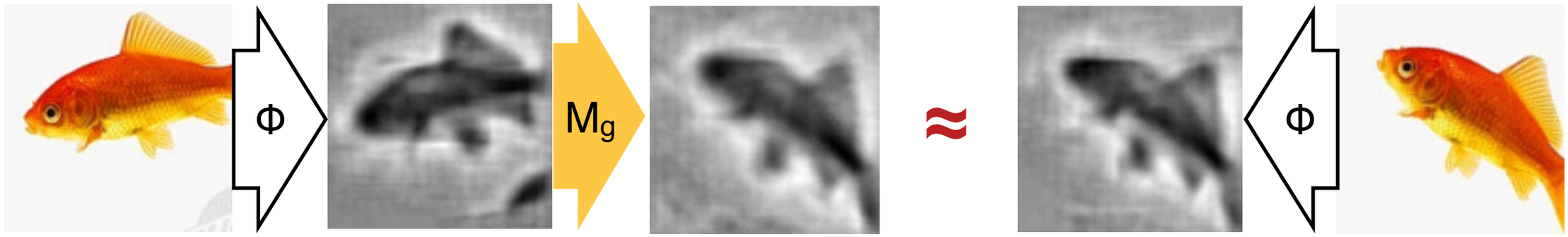


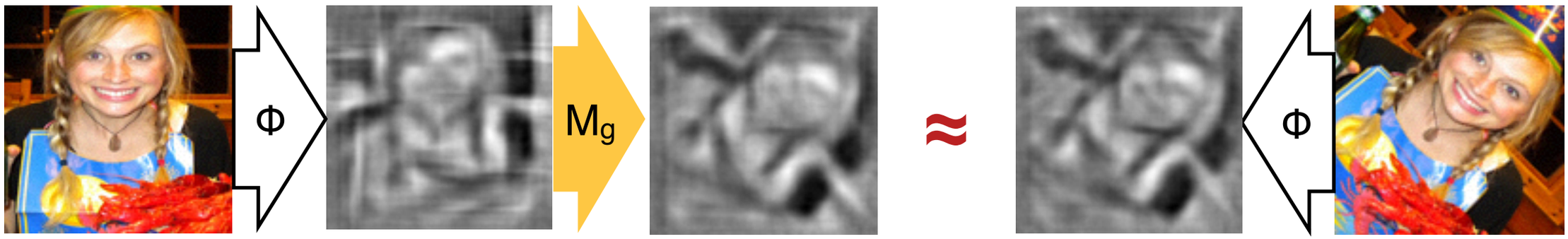
An empirical test of equivariance

HOG features

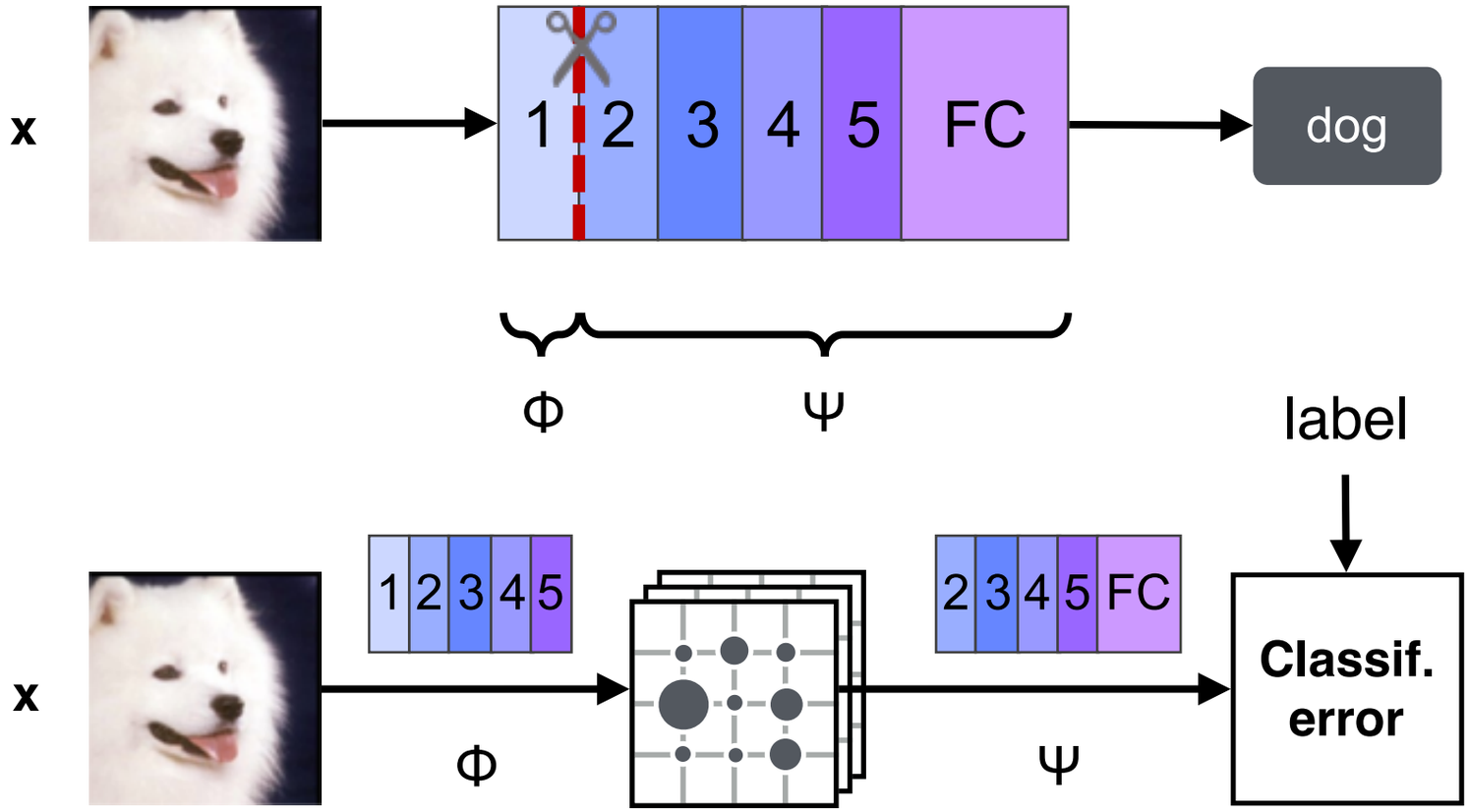
rotation 45 deg







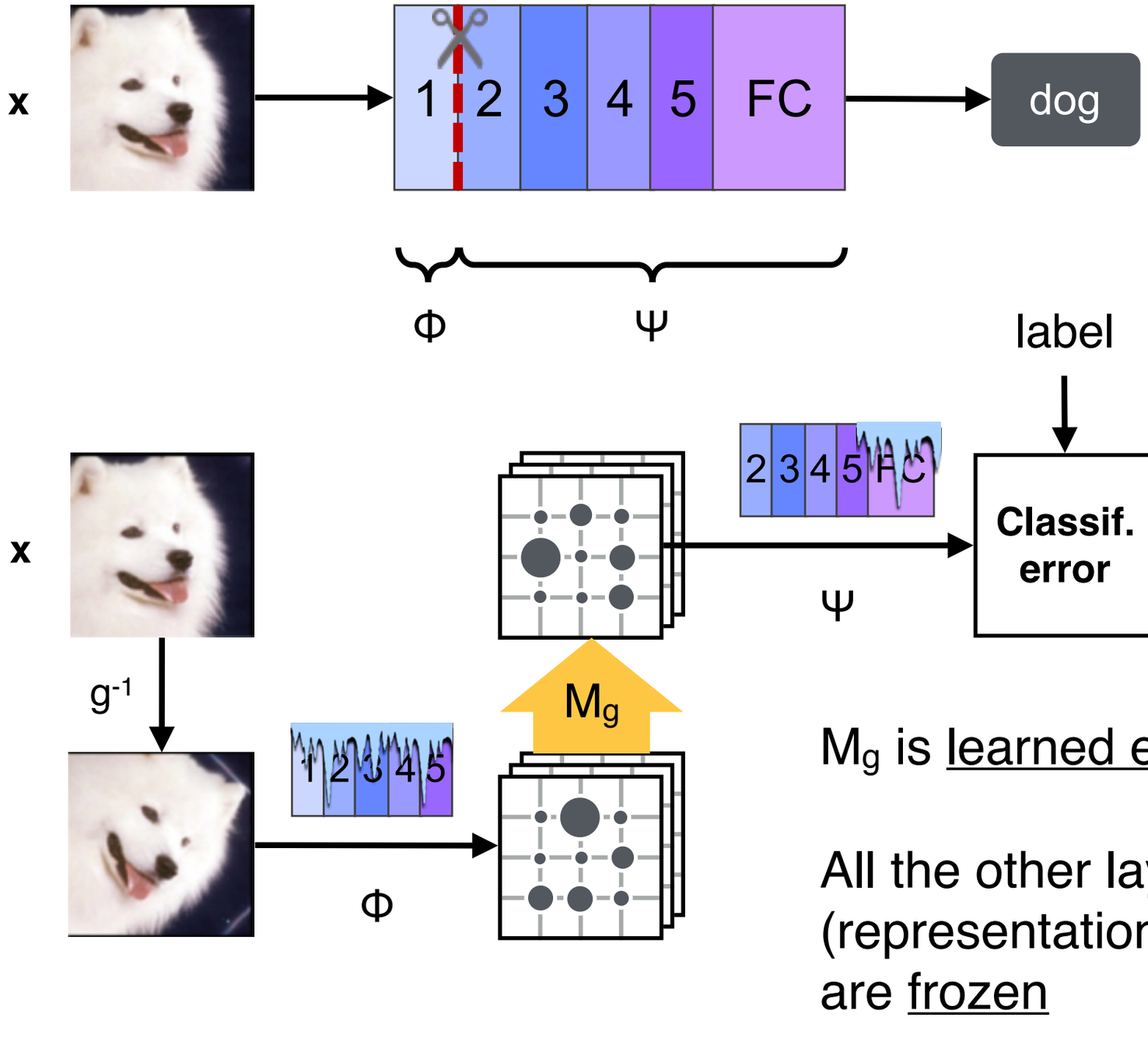
CNN: a sequence of representations



We run the same analysis on a typical CNN architecture

- AlexNet [Krizevsky et al. 12]
- 5 convolutional layers + fully-connected layers
- Trained on ImageNet ILSVRC

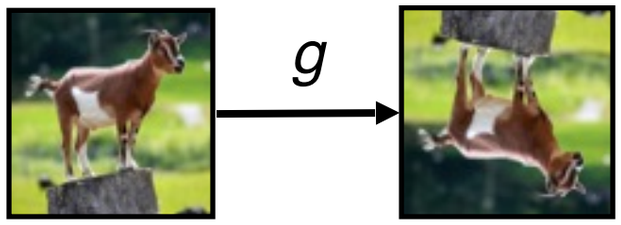
A discriminative goal to learn equivariance



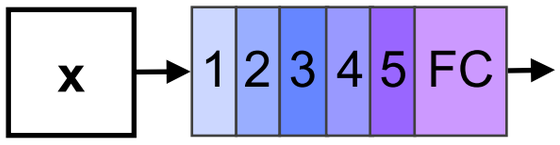
M_g is learned empirically

All the other layers (representation and classifier) are frozen

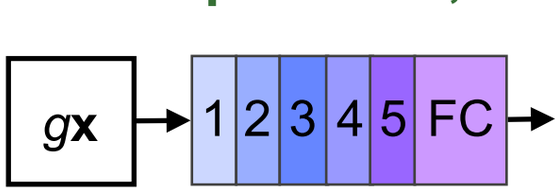
Vertical flips



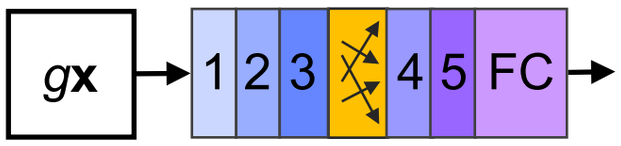
Uncompensated, no TF



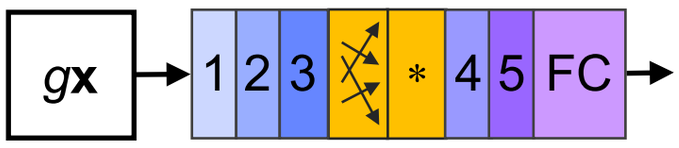
Uncompensated, TF

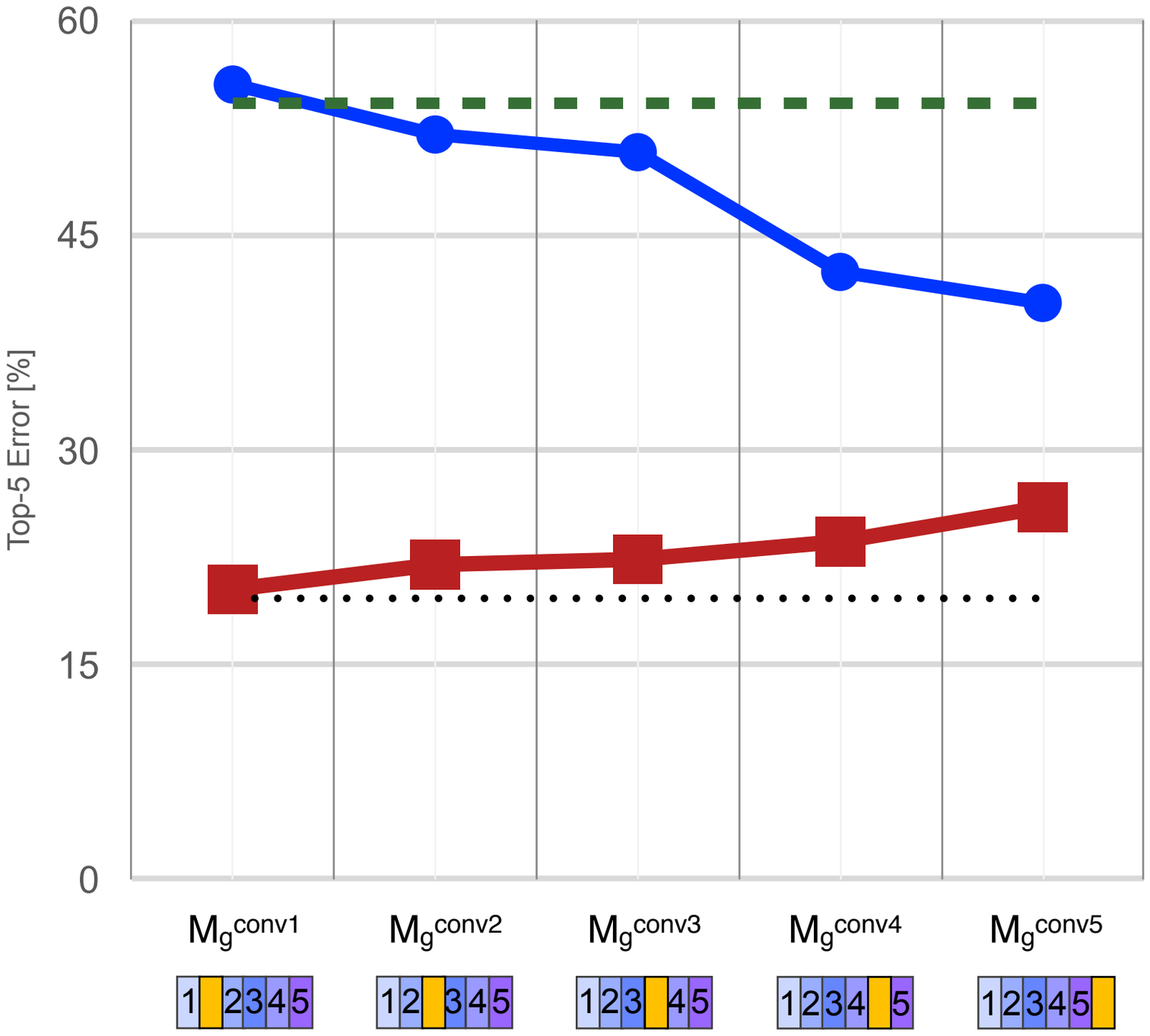


Compensated TF w perm.



Compensated TF, learned





Summary

Modern CNNs are still “new” technology

- Expect bigger and meaner CNNs to further improve performance
- CNNs are more than big balls of parameters
- We do not really understand what they do

Understanding deep nets

- What do deep networks do?
- Are there computational / statistical principles to be learned?
- How much do deep nets learn about the visual world?

Possible angles

- Visualisations
- Probing statistical properties

Are we there yet?

No!

Integrated computer vision

- One network to rule them all

Cognition

- Integrate perception and “the rest”
- Planning, memory, background knowledge, …

No labels required

- Unsupervised
- Alternatively supervised (reinforcement, pseudo-tasks)

Spatial reasoning

- From image-centric to object-centric and scene-centric understanding
- Representing deformable 3D shape
- Dynamic environments, physics

Understanding visual representations

Learnable representations, and deep convolutional neural networks (CNNs) in particular, have become the preferred way of extracting visual features for image understanding tasks, from object recognition to semantic segmentation. Part of the reason for the power of these models is that they contain millions of parameters learned from millions of images. However, this also means that deep networks are black boxes of difficult interpretation.

In this talk, I will focus on the problem of understanding deep networks. First, I will discuss a number of techniques to visualise the feature learned in a CNN: inversion, exploring which visual information is retained in a deep representation, activation maximisation, which finds patterns that maximally excite representation components, and caricaturisation, which emphasise which visual properties are captured by different representation layers. I will also briefly discuss methods to study how image transformations are encoded by representations, and whether different representations are equivalent.

Bio

Andrea Vedaldi is Assistant Professor of Engineering Science at the University of Oxford. His main research interests include large scale machine learning and optimization applied to semantic image understanding. He is author of more than 50 peer-reviewed publications in the top machine vision and artificial intelligence conferences and journals (CVPR, ECCV, ICCV, NIPS; h-index 34). He is project leader of VLFeat (http://www.vlfeat.org), a popular computer vision software toolkit (ACM Open Source Award 2010), and of MatConvNet (http://www.vlfeat.org/matconvnet), a MATLAB-based toolbox for convolutional neural networks. He is a recipient of the ERC Starting Grant (similar to the NSF CAREER award), of the Mark Everingham Prize (ICCV 2015) for selfless contributions to the computer vision community, a Glasstone Research Fellow, a J. J. Spooner Fellow, a winner of the PASCAL Visual Object Classification competition, and a recipient of the Best Paper Award at the British Machine Vision Conference (BMVC) and of the Open Source Software Award by the ACM. He is supported by the ERC, EPSRC, BP, and XRCE.