# 7. More Convolutional Neural Networks

CS 519 Deep Learning, Winter 2017

Fuxin Li

*With materials from Zsolt Kira, Roger Grosse, Nitish Srivastava*

# Backpropagation for the convolution operator

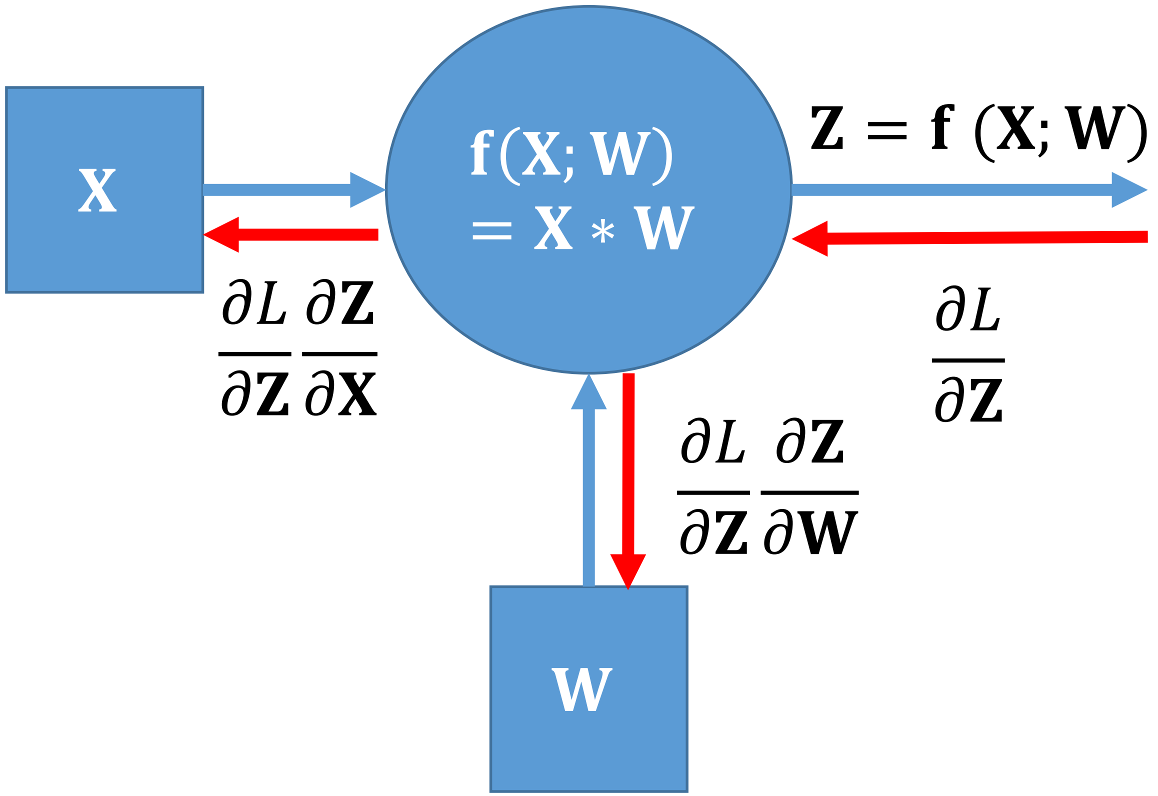
Forward pass:

Compute $\mathbf{f}(\mathbf{X};\mathbf{W}) = \mathbf{X} * \mathbf{W}$

Backward pass:

Compute $\frac{\partial Z}{\partial X} = ?$

$$\frac{\partial Z}{\partial W} = ?$$



# Convolution as Matrix Multiplication

For each filter,
$\mathbf{Z}$ is image with same size
as $\mathbf{X}$ (assuming padding)

$$\mathbf{Z} = \mathbf{f}(\mathbf{X}; \mathbf{W}) = \mathbf{X} * \mathbf{W}$$

Gradient $\dfrac{\partial \mathbf{Z}}{\partial \mathbf{X}} = ?$

Toeplitz Matrix

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $w_{11}$ | $w_{21}$ | $w_{31}$ | … | $w_{12}$ | $w_{22}$ | $w_{32}$ | … | $w_{13}$ | … |
| 0 | $w_{11}$ | $w_{21}$ | … | 0 | $w_{12}$ | $w_{22}$ | … | 0 | … |
| 0 | 0 | $w_{11}$ | … | 0 | 0 | $w_{12}$ | … | 0 | … |
| … | … | … | … | … | … | … | … | … | … |
| 0 | 0 | 0 | 0 | $w_{11}$ | $w_{21}$ | $w_{31}$ | … | $w_{12}$ | … |
| 0 | 0 | 0 | 0 | 0 | $w_{11}$ | $w_{21}$ | … | 0 | … |
| 0 | 0 | 0 | 0 | 0 | 0 | $w_{11}$ | … | 0 | … |
| … | … | … | … | … | … | … | … | … | … |

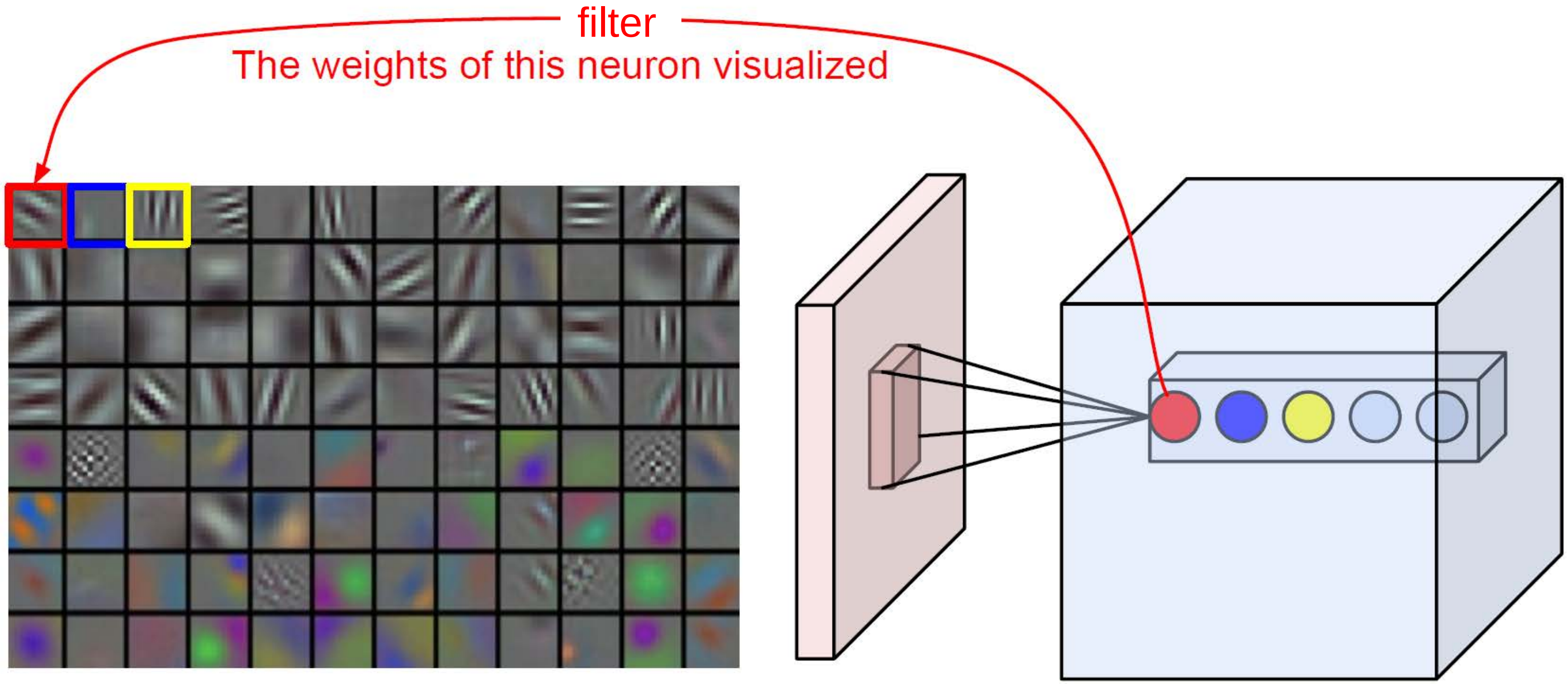
| |
|---|
| $x_{11}$ |
| $x_{21}$ |
| $x_{31}$ |
| … |
| $x_{12}$ |
| $x_{22}$ |
| $x_{32}$ |
| … |
| $x_{13}$ |

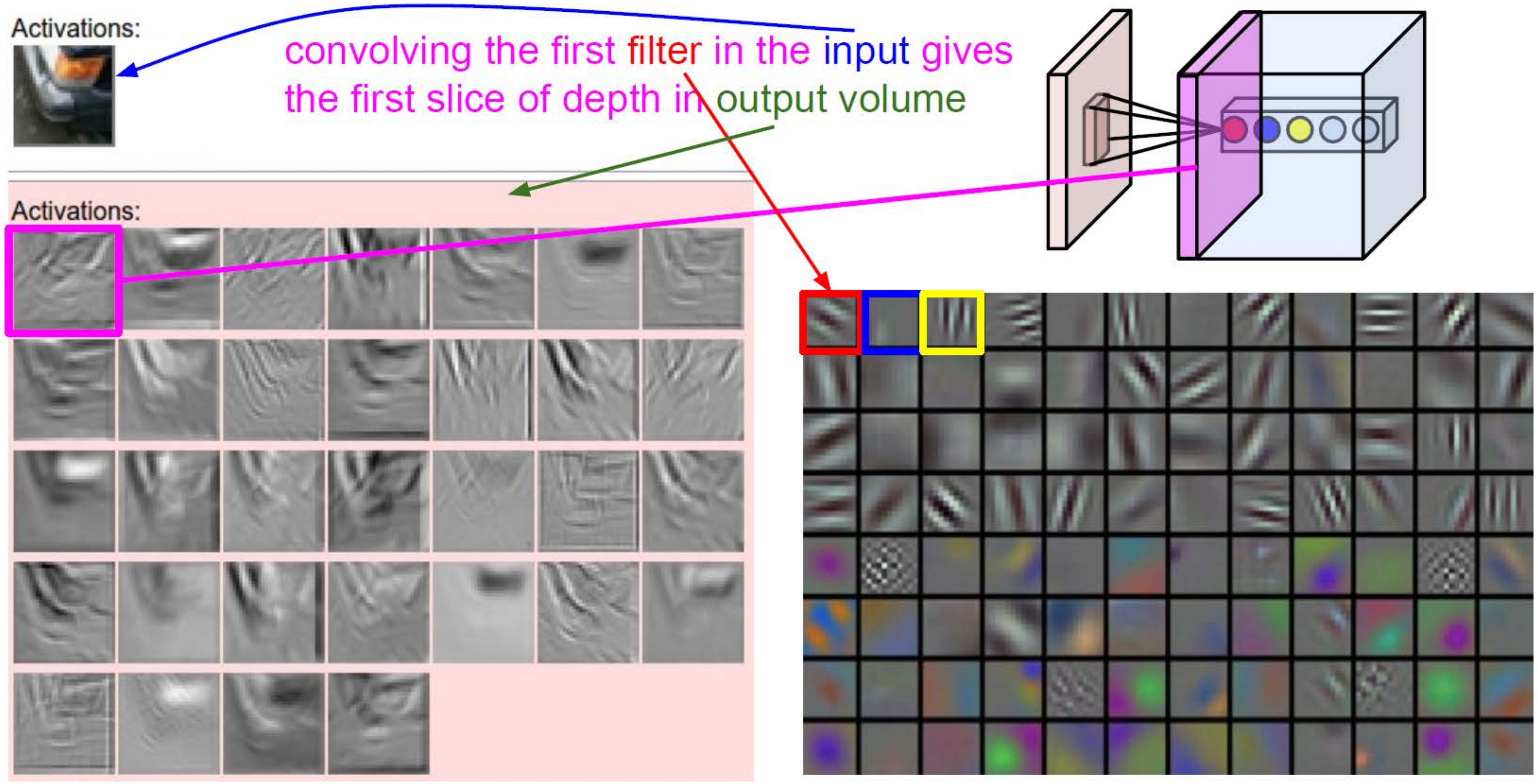
| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $x_{11}$ | $x_{12}$ | $x_{13}$ | $x_{21}$ | $x_{22}$ | $x_{23}$ | $x_{31}$ | $x_{32}$ | $x_{33}$ |
| $x_{12}$ | $x_{13}$ | $x_{14}$ | $x_{22}$ | $x_{23}$ | $x_{24}$ | $x_{32}$ | $x_{33}$ | $x_{34}$ |
| $x_{13}$ | $x_{14}$ | $x_{15}$ | $x_{23}$ | $x_{24}$ | $x_{25}$ | $x_{33}$ | $x_{34}$ | $x_{35}$ |
| … | … | … | … | … | … | … | … | … |

| |
|---|
| $w_{11}$ |
| $w_{12}$ |
| $w_{13}$ |
| $w_{21}$ |
| $w_{22}$ |
| $w_{23}$ |
| $w_{31}$ |
| $w_{32}$ |
| $w_{33}$ |

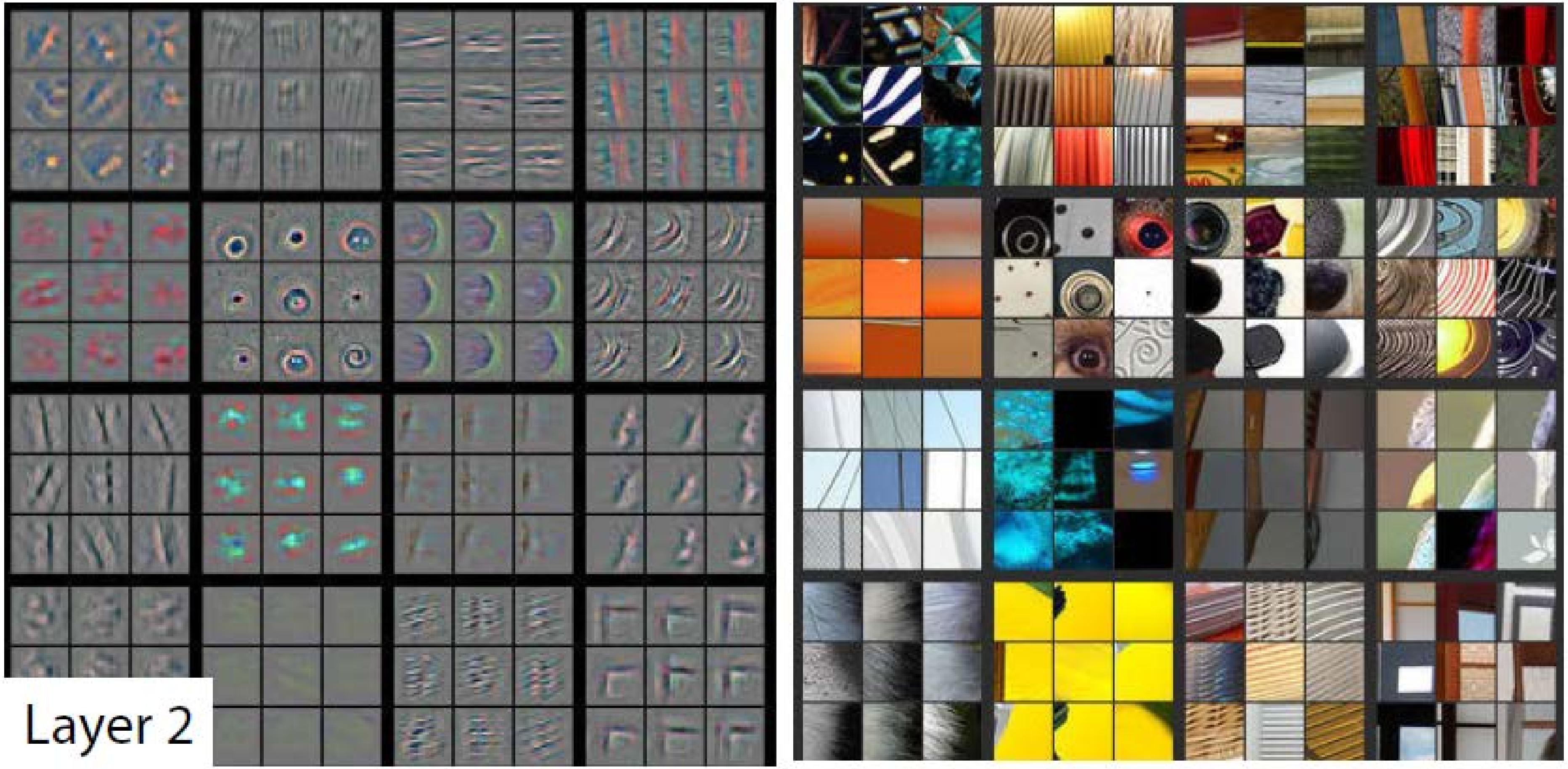
# Visualization of the filters (1st layer)

- 11x11 filters

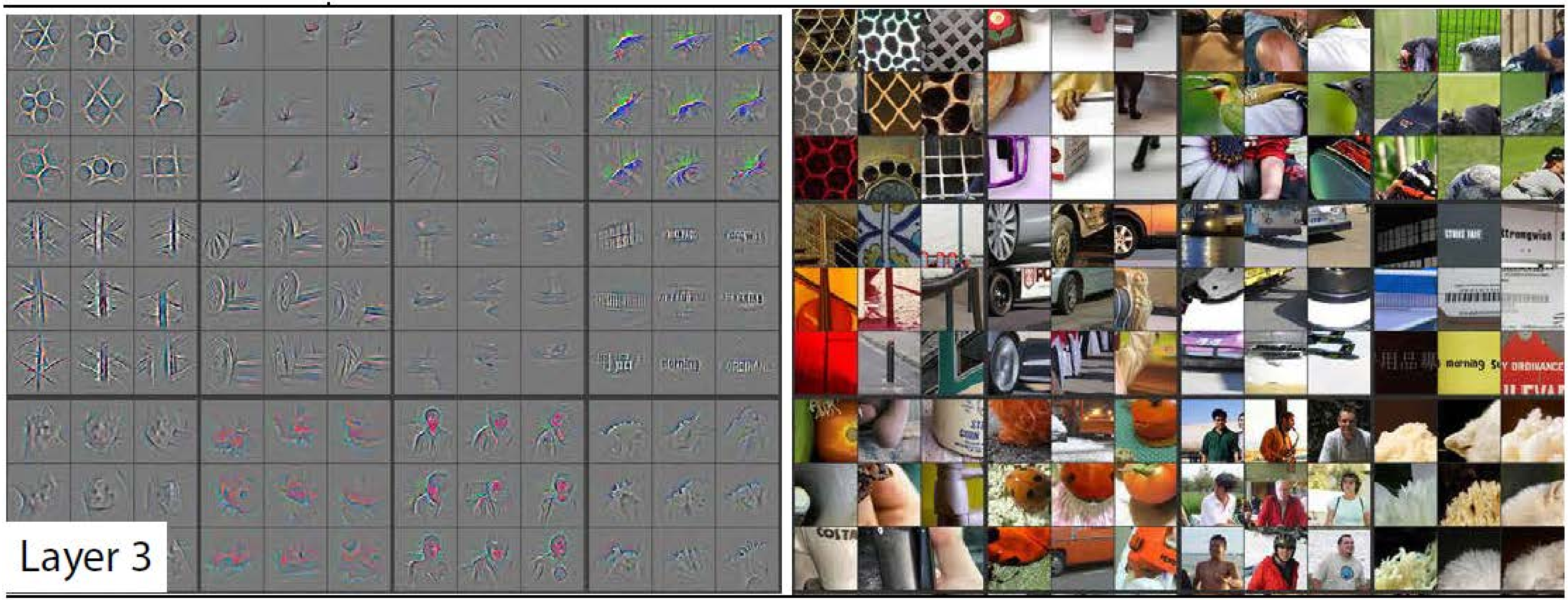



Activations:
convolving the first filter in the input gives
the first slice of depth in output volume
Activations:

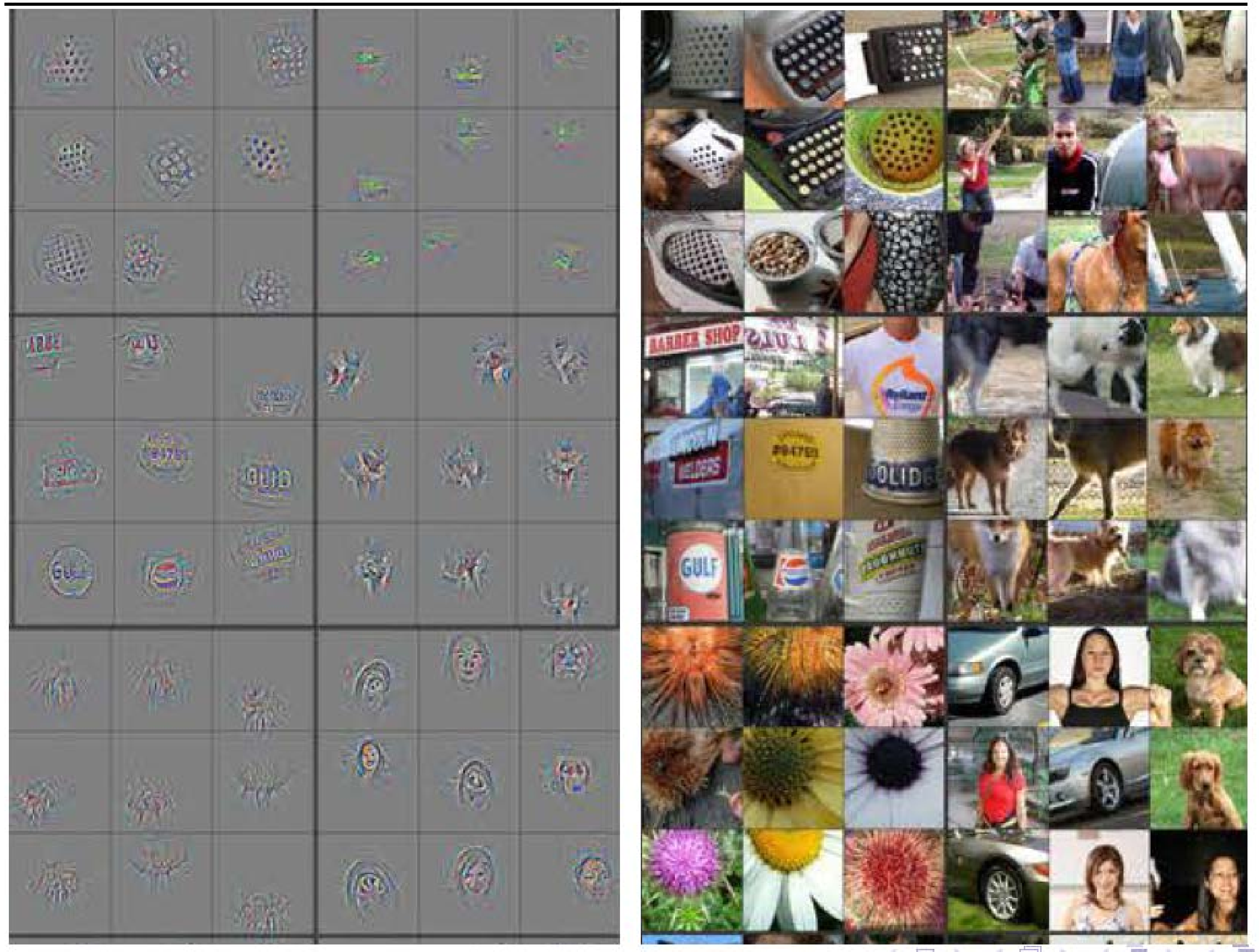
# Visualization of second-level filters



# Visualization of third-level filters



# Visualization of layer 5



# Another visualization: Maximizing class score

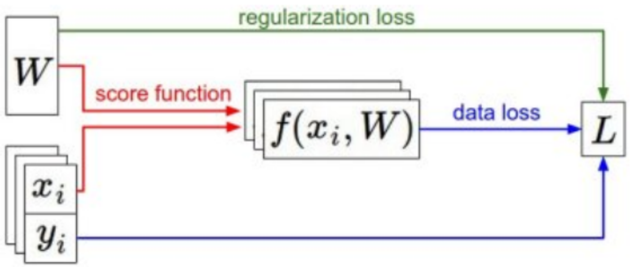
*Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps*
*Karen Simonyan, Andrea Vedaldi, Andrew Zisserman, 2014*

1. Find images that maximize some class score:

$$\arg\max_I \boxed{S_c(I)} - \lambda \|I\|_2^2$$

Score for class c
(before Softmax)

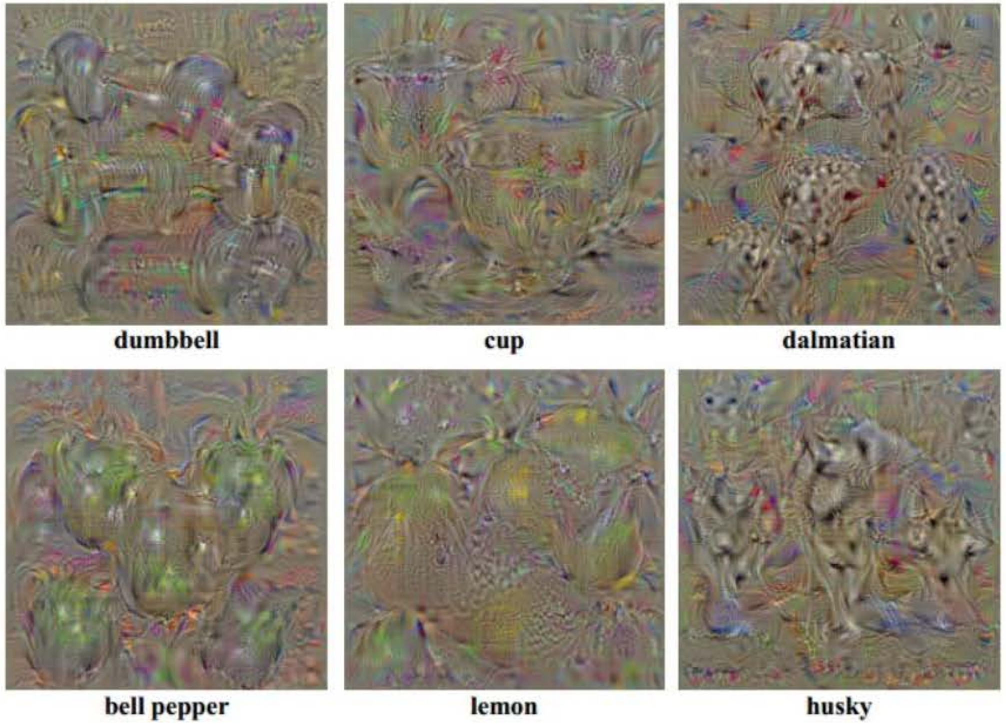
Remember:




- **Start with zero image**
- **Keep weights fixed, perform backpropagation for a class!**

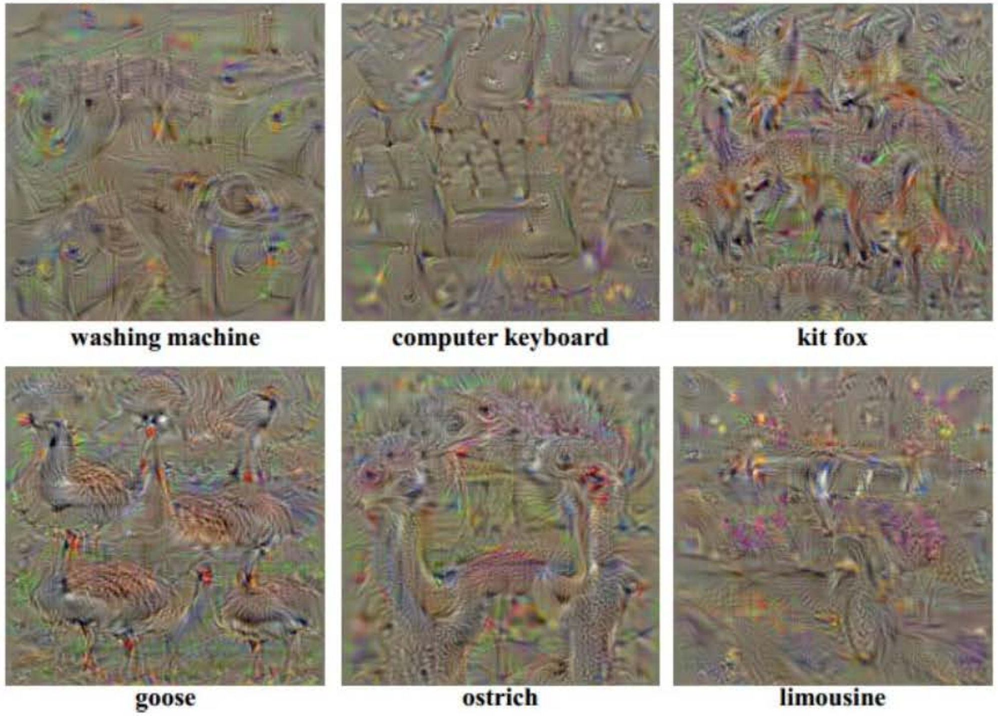
*Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps*
*Karen Simonyan, Andrea Vedaldi, Andrew Zisserman, 2014*

1. Find images that maximize some class score:



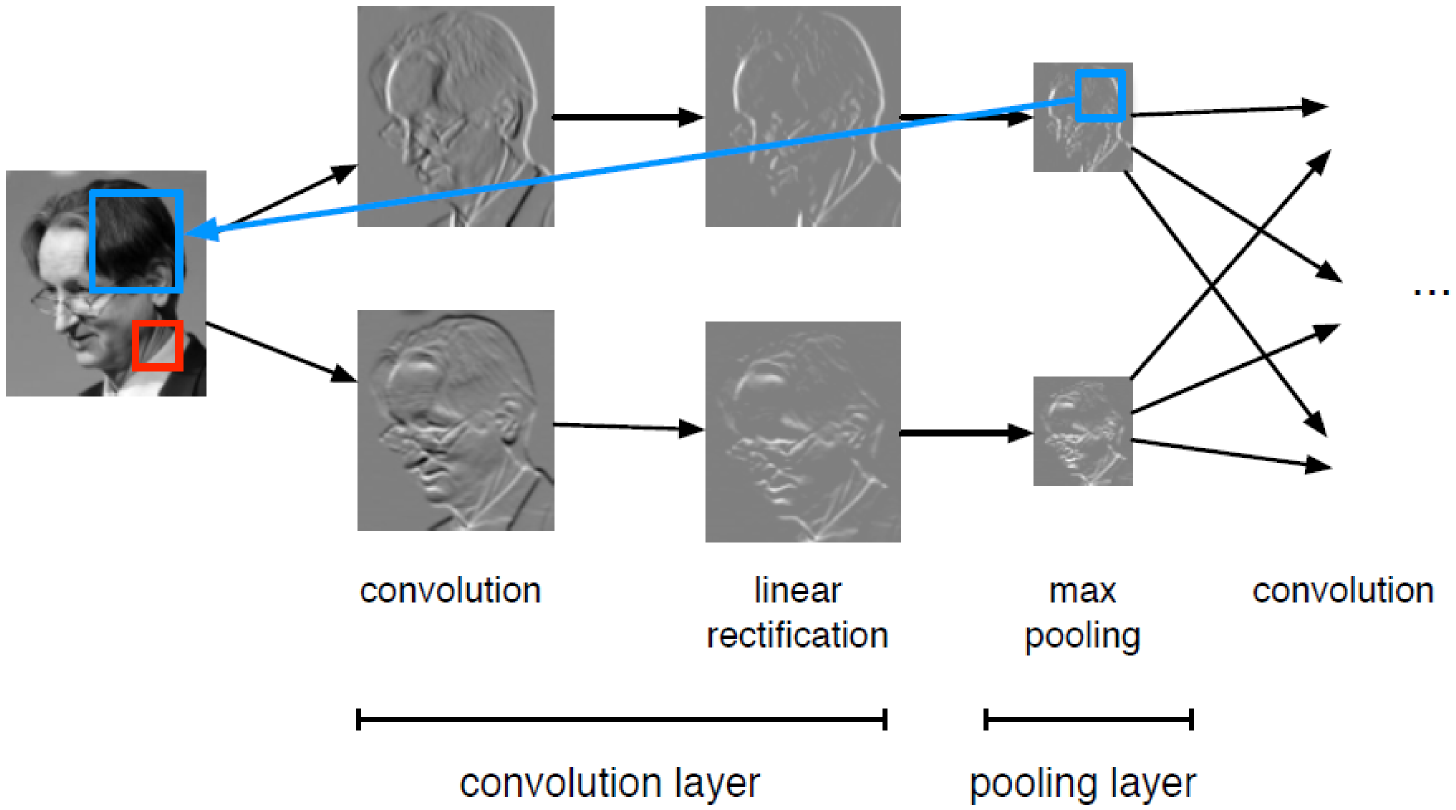
*Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps*
*Karen Simonyan, Andrea Vedaldi, Andrew Zisserman, 2014*

# 1. Find images that maximize some class score:



# Convolution/ReLU/Pooling

Because of pooling, higher-layer filters can cover a larger region of the input than equal-sized filters in the lower layers.



Modern CNN trend toward:

- Small filter sizes (3x3 and less)
- Small pooling sizes (2x2 and less)
- Small strides (stride = 1, ideally)
- Deep

- Conv Layers should **pad with zeros** to not reduce spatial size
- Pool Layers should reduce size once in a while
- Eventually Fully-Connected Layers take over

Graham. Fractional Max-pooling. arXiv:1412.6071v4

# arXiv!

- Because how fast the field evolves, most deep learning papers are on arXiv first
- http://arxiv.org/list/cs.CV/recent
- http://arxiv.org/list/cs.CL/recent

- Check that for newest papers/ideas!

*Understanding Deep Image Representations by Inverting Them*
*[Mahendran and Vedaldi, 2014]*

original image

reconstructions from the 1000 log probabilities for ImageNet (ILSVRC) classes

Find an image such that:

- Its code is similar to a given code
- It "looks natural" (image prior regularization)

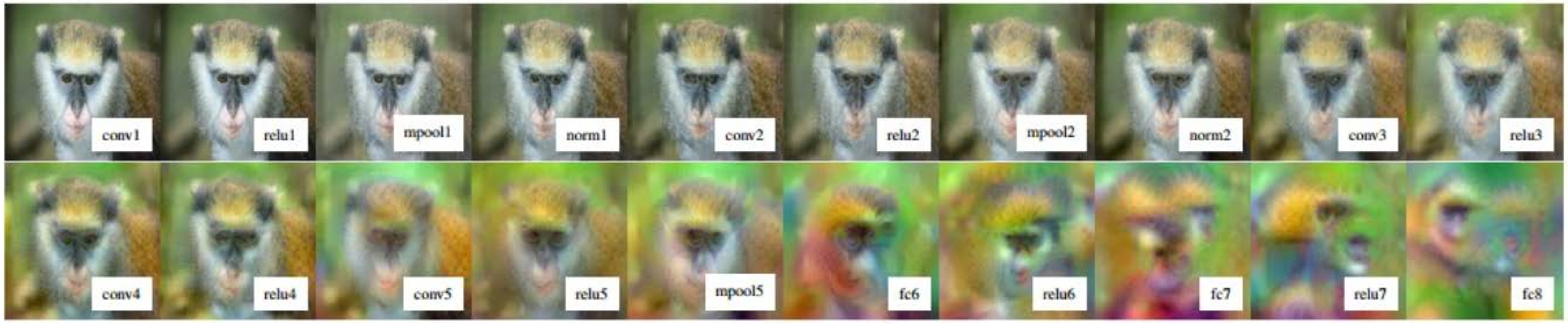
$$\mathbf{x}^* = \underset{\mathbf{x}\in\mathbb{R}^{H\times W\times C}}{\operatorname{argmin}} \ \ell(\Phi(\mathbf{x}), \Phi_0) + \lambda\mathcal{R}(\mathbf{x})$$

$$\ell(\Phi(\mathbf{x}), \Phi_0) = \|\Phi(\mathbf{x}) - \Phi_0\|^2$$

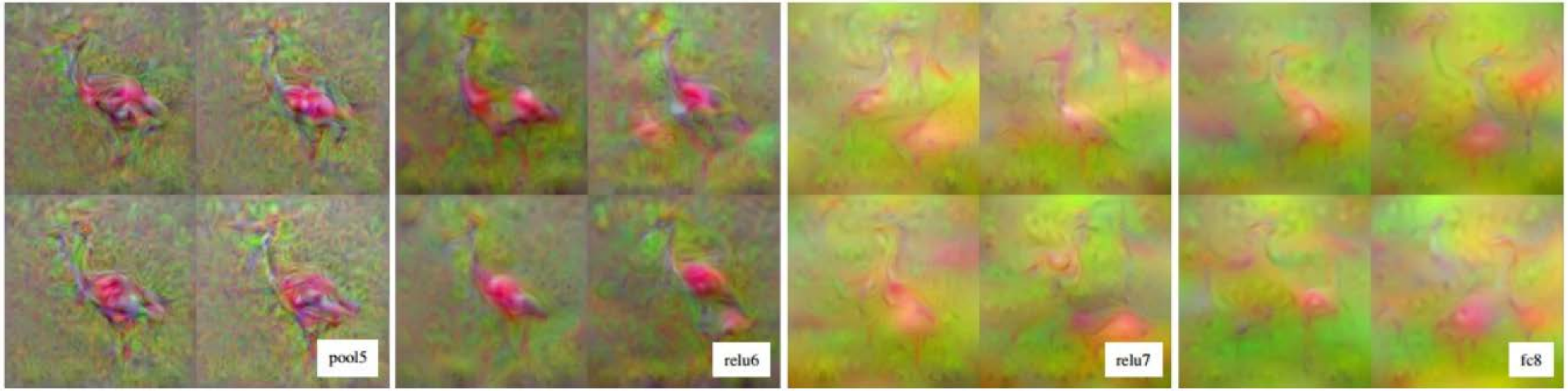
Solve using SGD + Momentum

Reconstructions from the representation after last last pooling layer (immediately before the first Fully Connected layer)

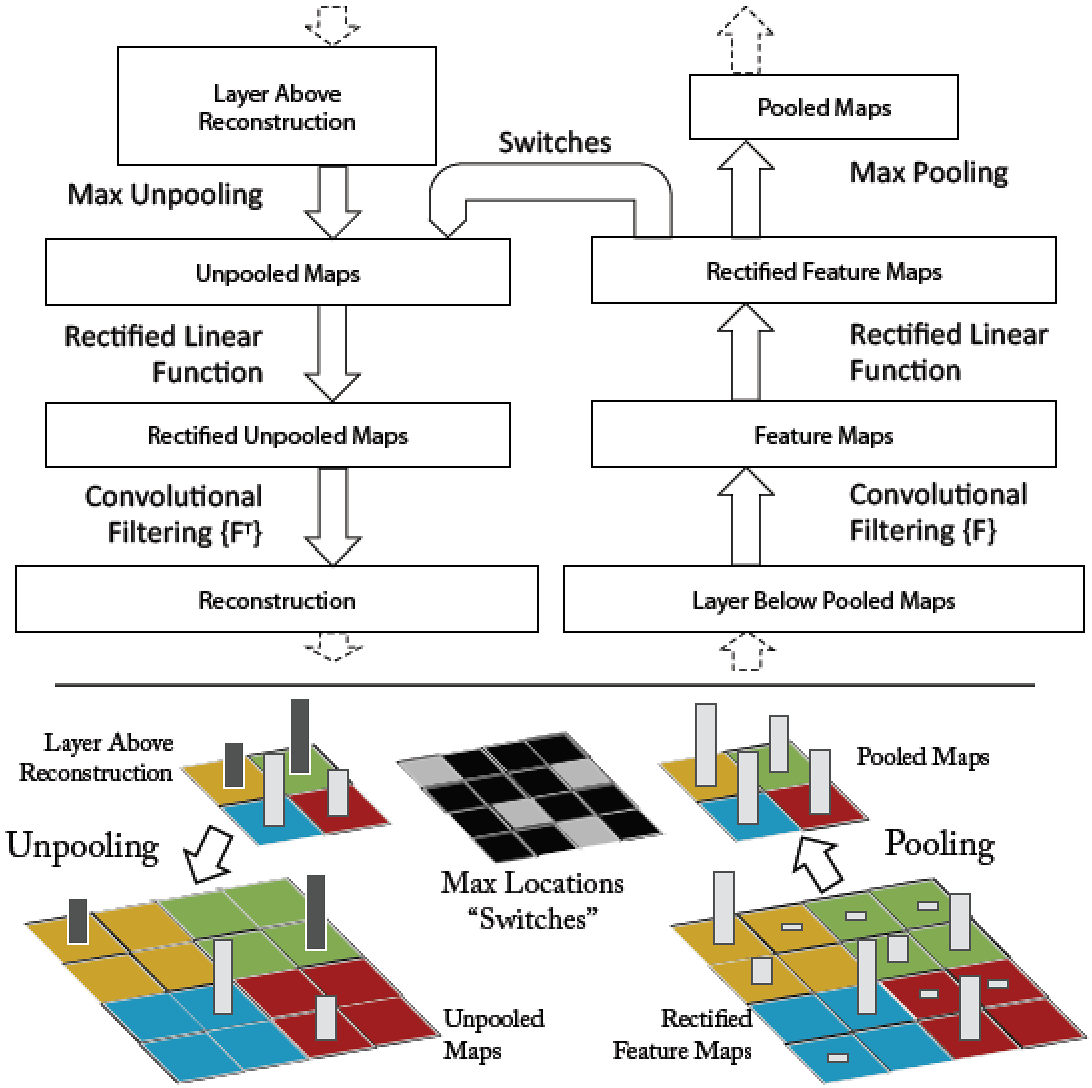
# Reconstructions from intermediate layers


Multiple reconstructions. Images in quadrants all “look” the same to the CNN (same code)

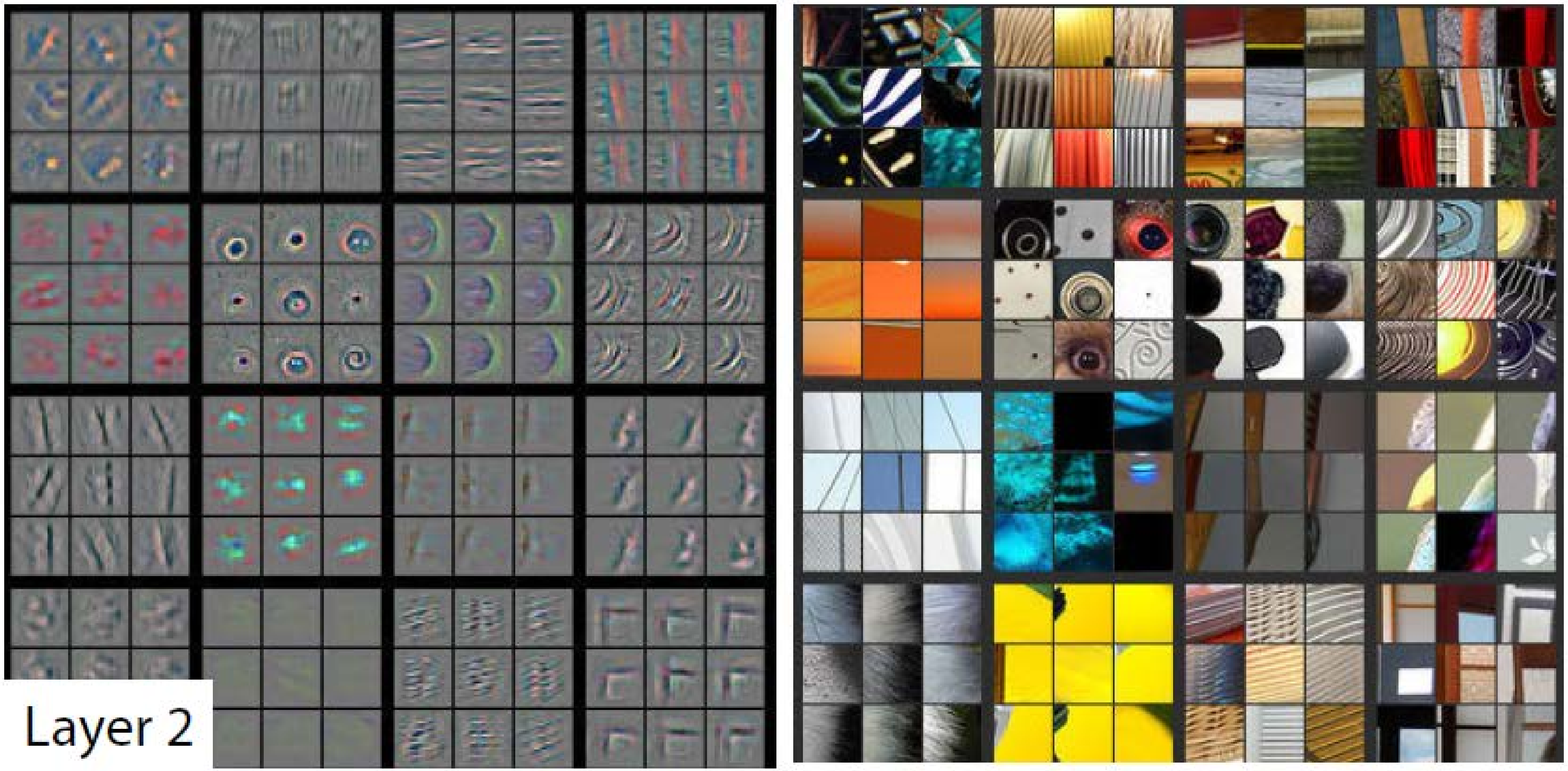


# Deconvolutional Network

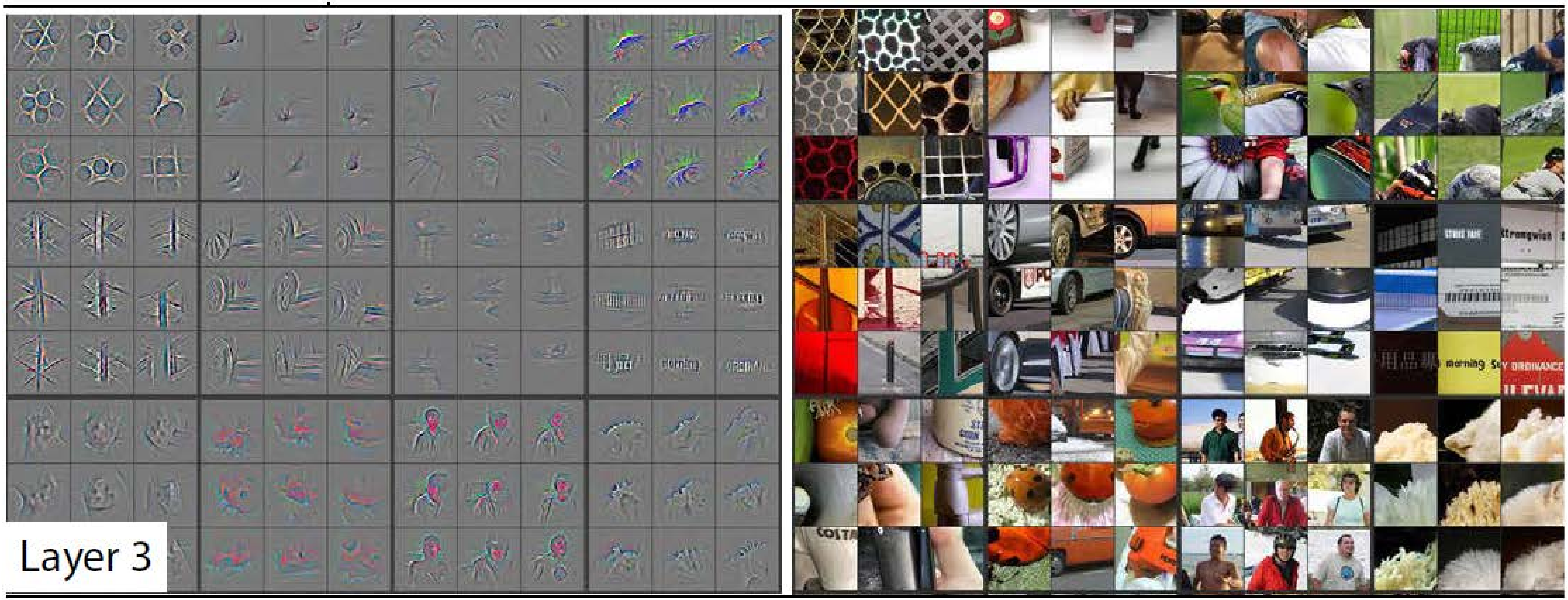



- **Instead of mapping pixels to features, map the other way around**
- **Can be used to learned unsupervised features**
- **Here, attached to trained convnet**

# Visualization of second-level filters



# Visualization of third-level filters



# Visualization of layer 5