Компьютерное зрение

Лекция 1 – Сверточные нейронные сети

2021

План лекции

1. Основные направления CV
2. Сверточные нейронные сети (CNN)
3. Интерпретация результатов CNN

Основные направления в CV

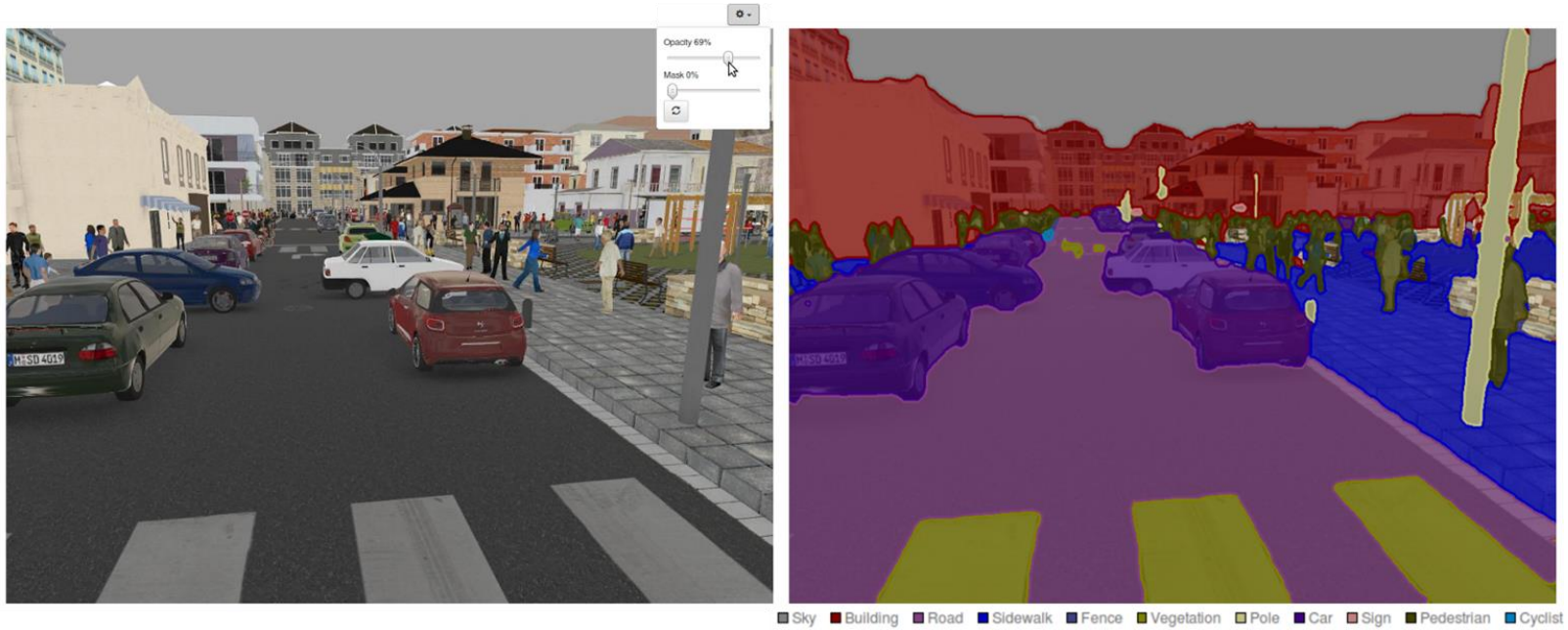
Классификация

Решается задача принадлежности изображения одному из классов

Основные направления в CV

Сегментация

Решается задача классификации, но попиксельной

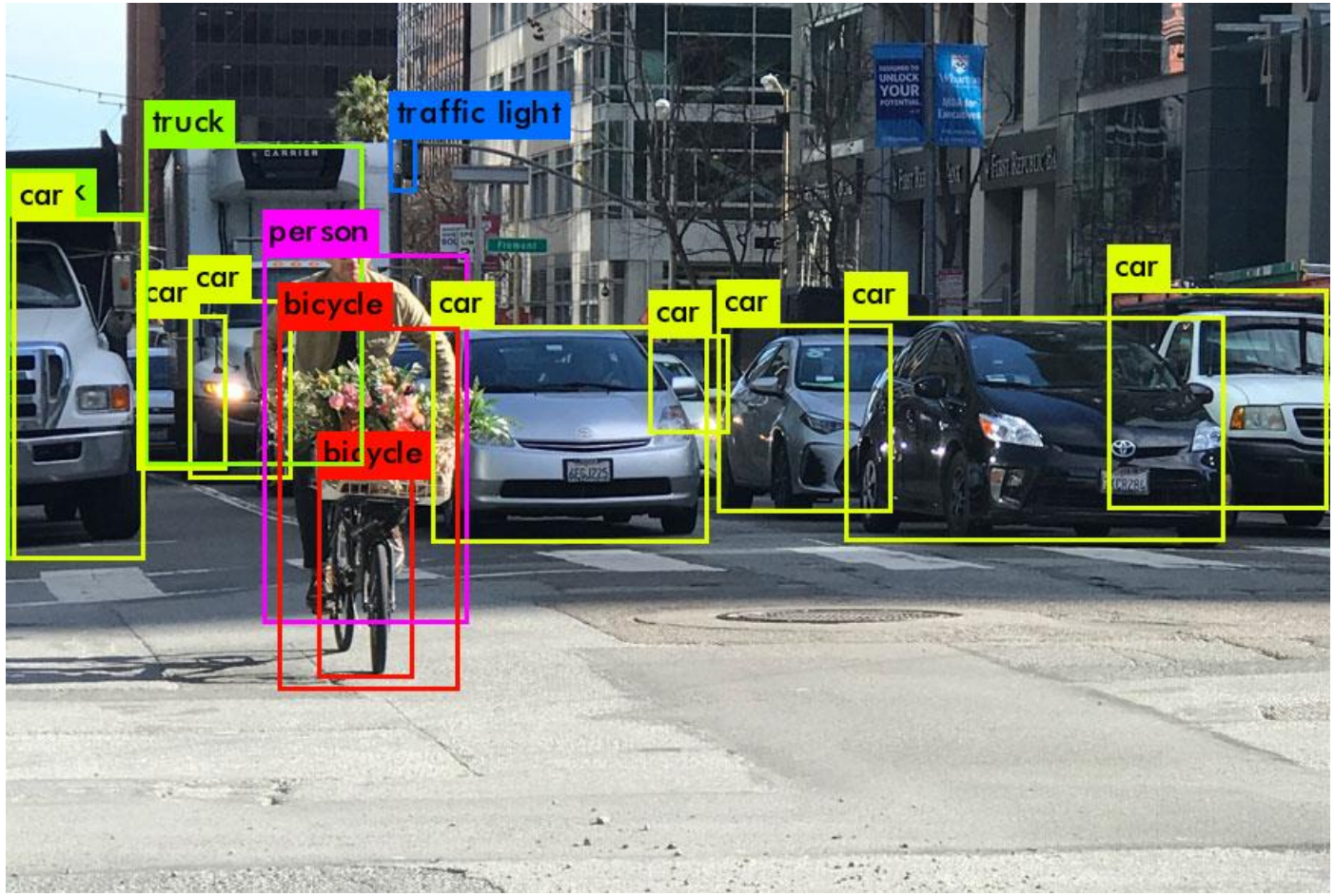


Изображение: ссылка на источник

Основные направления в CV

Детекция

Решается задача распознавания - найти и классифицировать



Изображение: ссылка на источник

Основные направления в CV

Трекинг и видео-аналитика

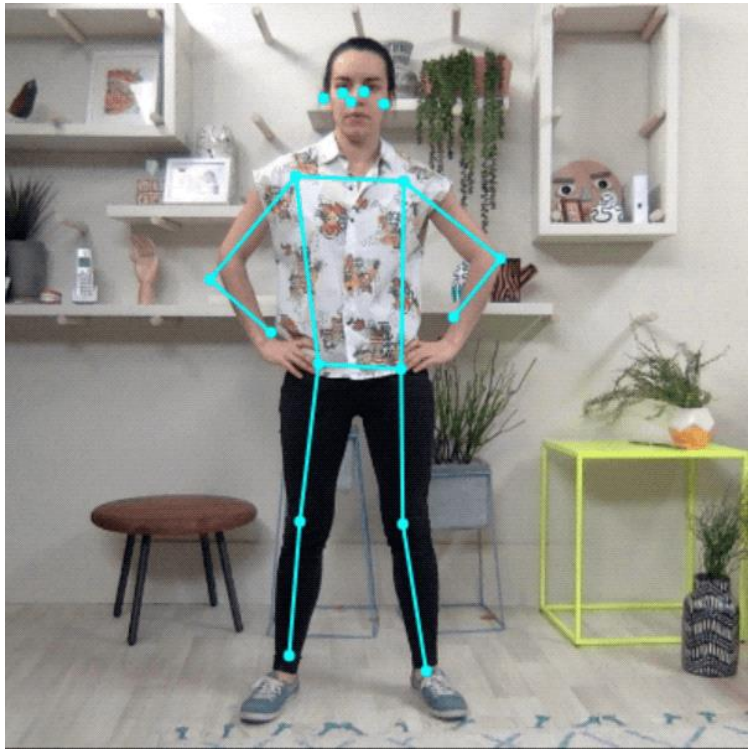
Распознавание объектов в видеопотоке

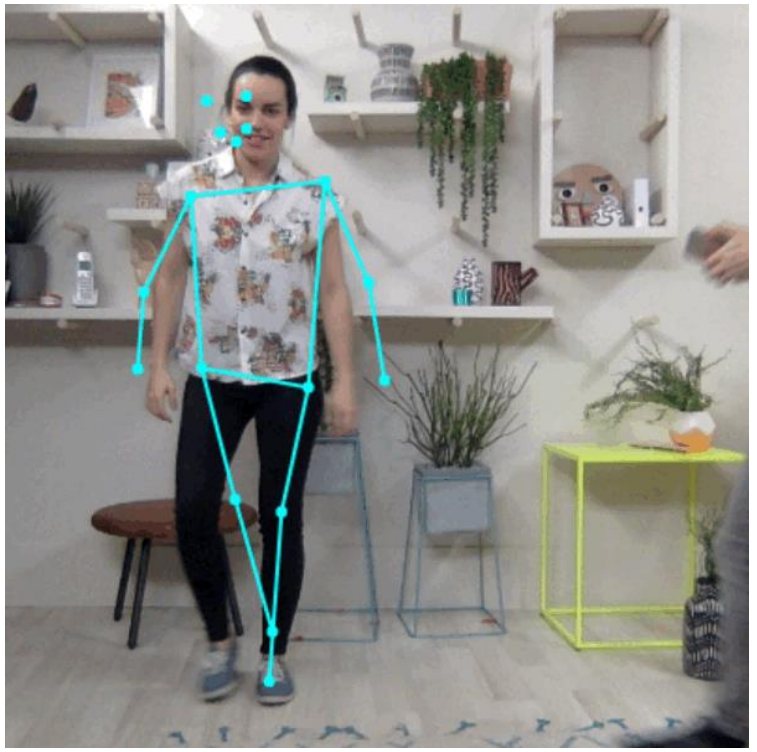
Гифка: ссылка на источник

Основные направления в CV

Оценка позы и взгляда человека

Решается задача распознавания ключевых точек, которые описывают позы человека, его положение, ориентацию в пространстве



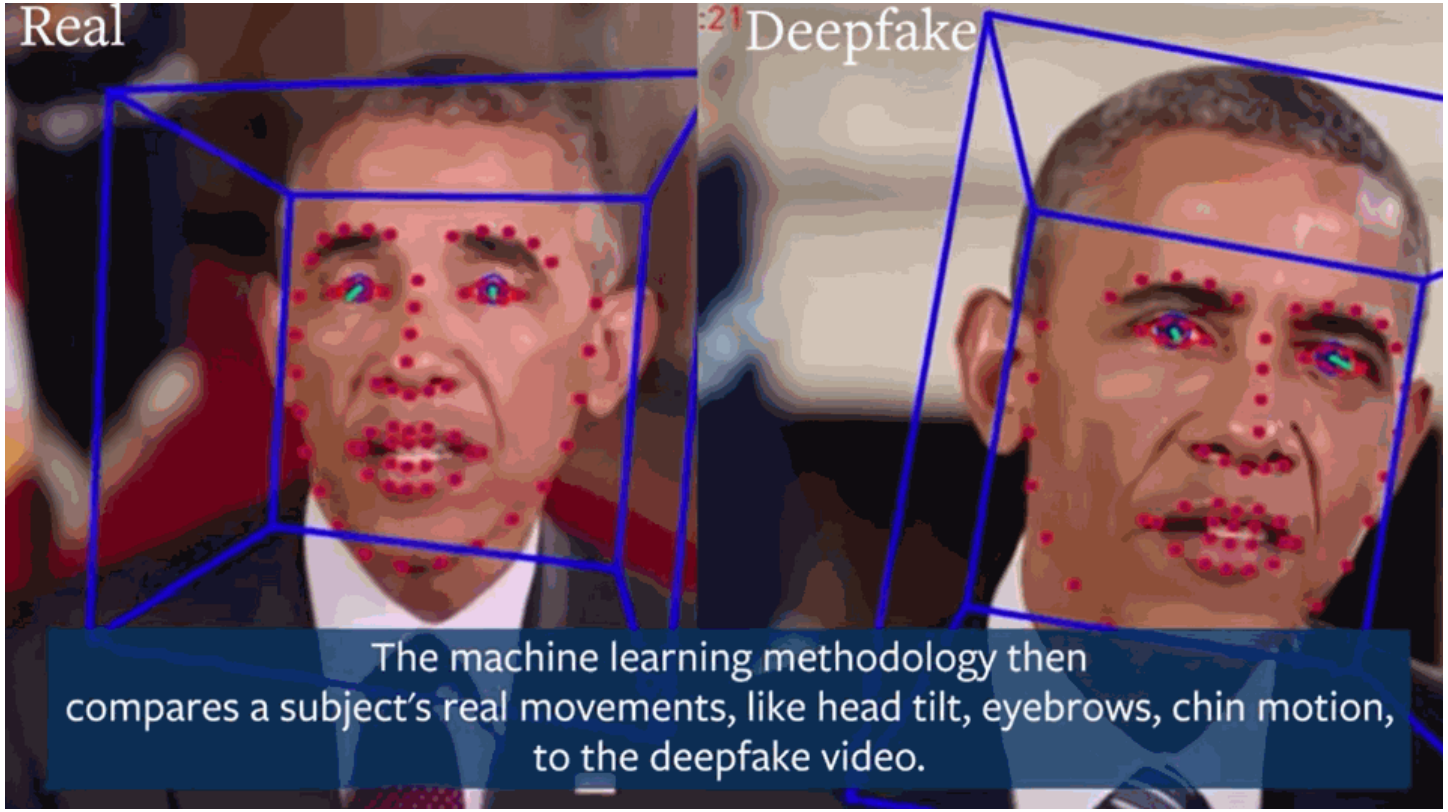


Гифки: ссылка на источник

Основные направления в CV

Биометрия

Решается задача классификации на изображении или видеопотоке: фейк или нет

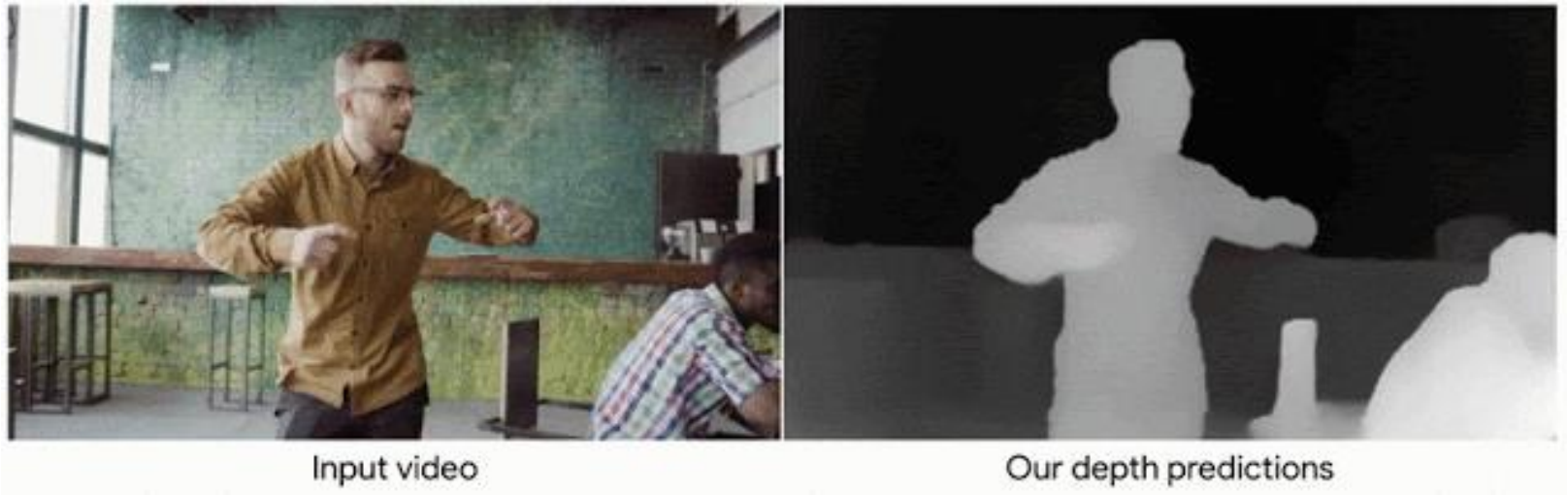


Видео: ссылка на источник

Основные направления в CV

Построение карты глубины

Решается задача построения карты расстояний до наблюдаемых объектов



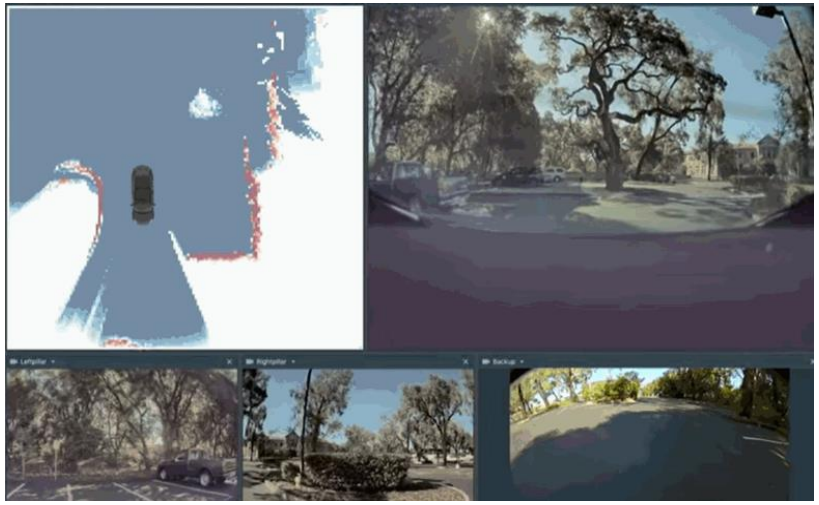
Гифка: ссылка на источник

Почему востребовано

Автономные системы

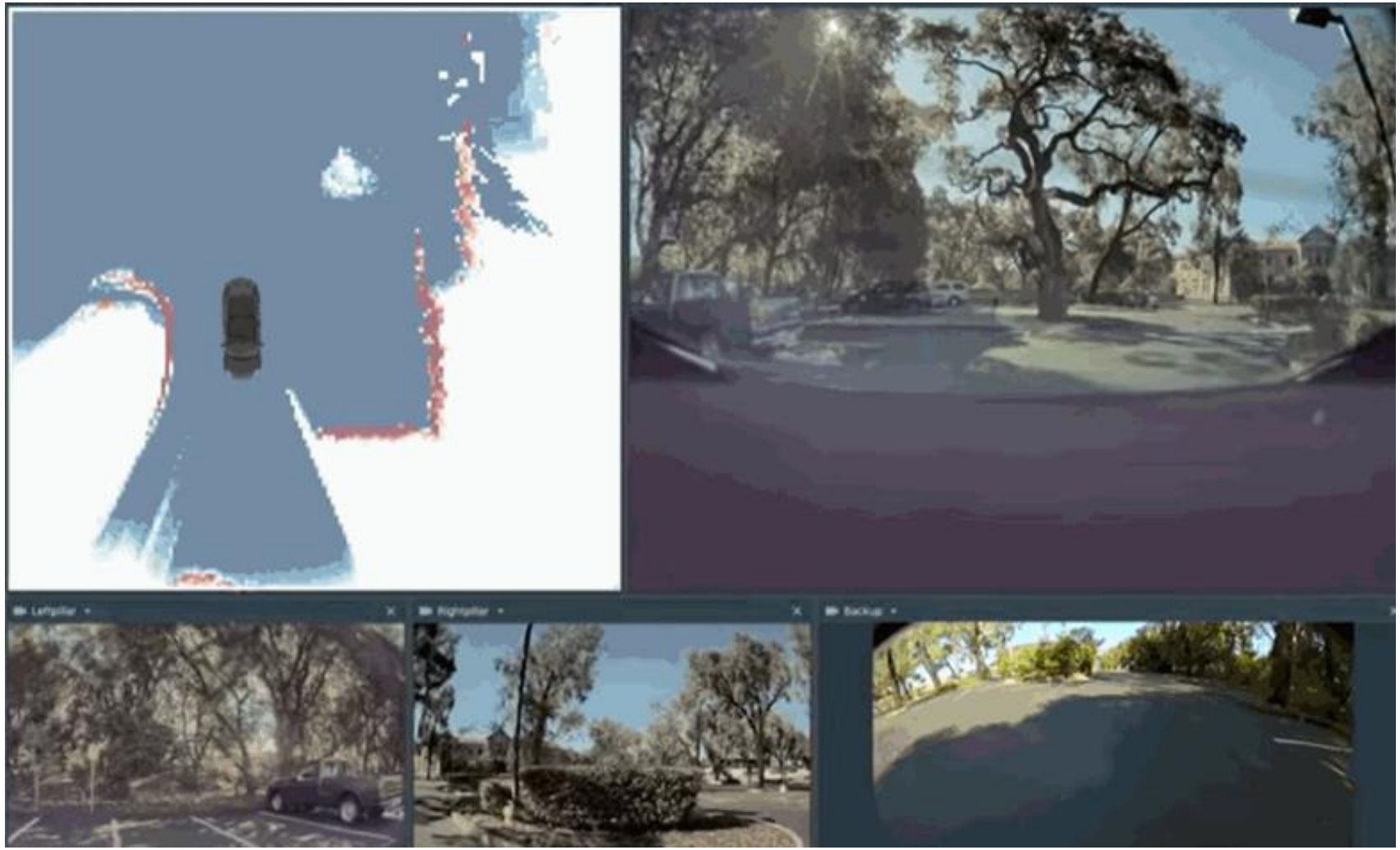
Автономные машины и роботы

Беспилотные летательные аппараты (БПЛА)



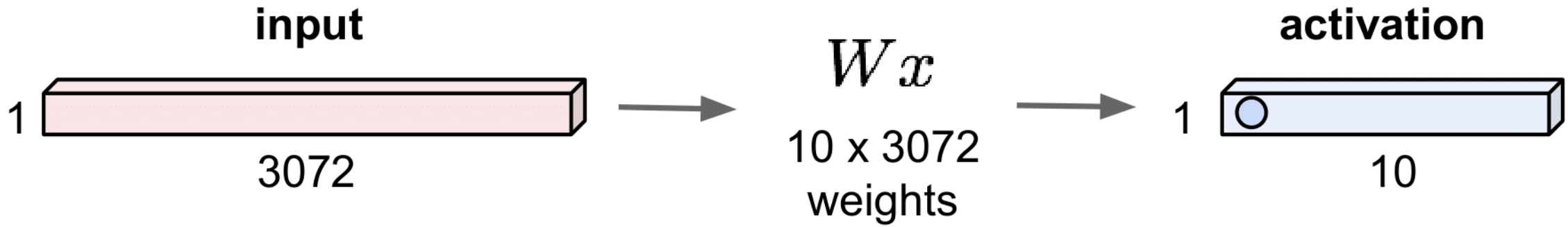
PyTorch at Tesla - Andrej Karpathy, Tesla | Spot robot patrolling Bishan-Ang Mo Kio Park | Полет группы БПЛА (ЛПСУ МФТИ)

Задача анализа окружающей обстановки



Fully Connected Layer

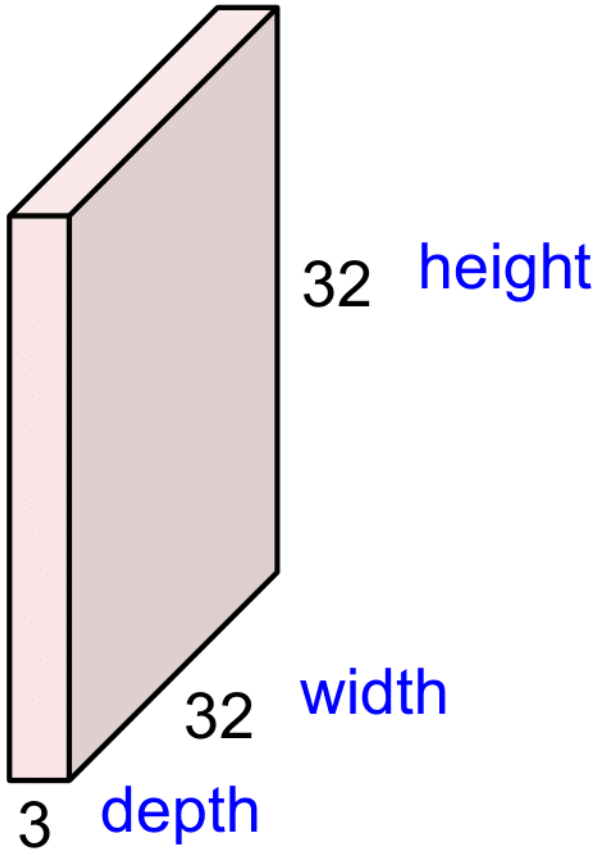
32x32x3 image -> stretch to 3072 x 1



Source: Stanford CS231n Lecture 5 2017 by Fei-Fei Li & Justin Johnson & Serena Yeung

Convolution Layer

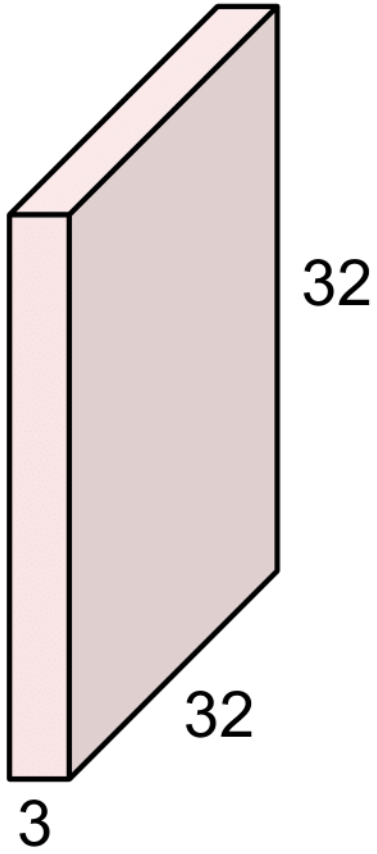
32x32x3 image -> preserve spatial structure



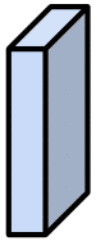
Source: Stanford CS231n Lecture 5 2017 by Fei-Fei Li & Justin Johnson & Serena Yeung

Convolution Layer

32x32x3 image



5x5x3 filter



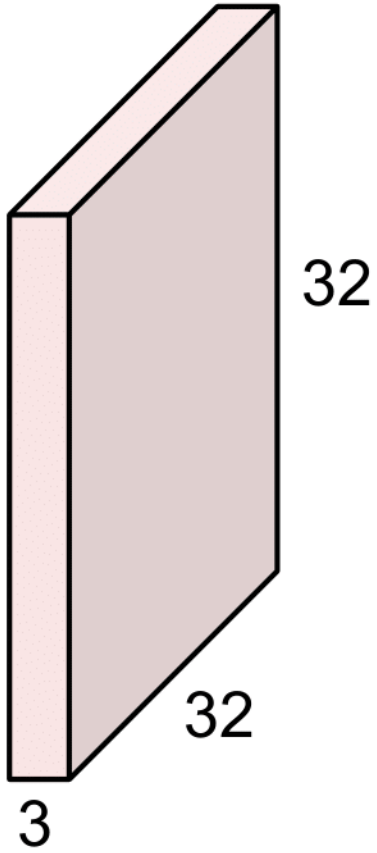
Convolve the filter with the image
i.e. “slide over the image spatially,
computing dot products”

Source: Stanford CS231n Lecture 5 2017 by Fei-Fei Li & Justin Johnson & Serena Yeung

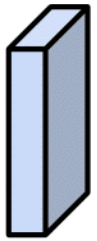
Convolution Layer

Filters always extend the full depth of the input volume

32x32x3 image



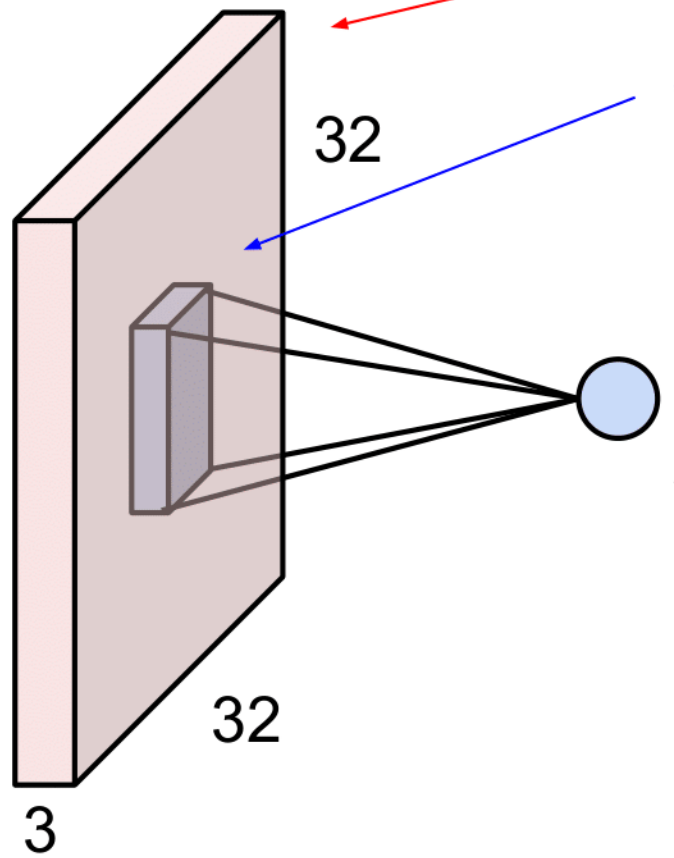
5x5x3 filter



Convolve the filter with the image
i.e. “slide over the image spatially, computing dot products”

Source: Stanford CS231n Lecture 5 2017 by Fei-Fei Li & Justin Johnson & Serena Yeung

Convolution Layer



32x32x3 image

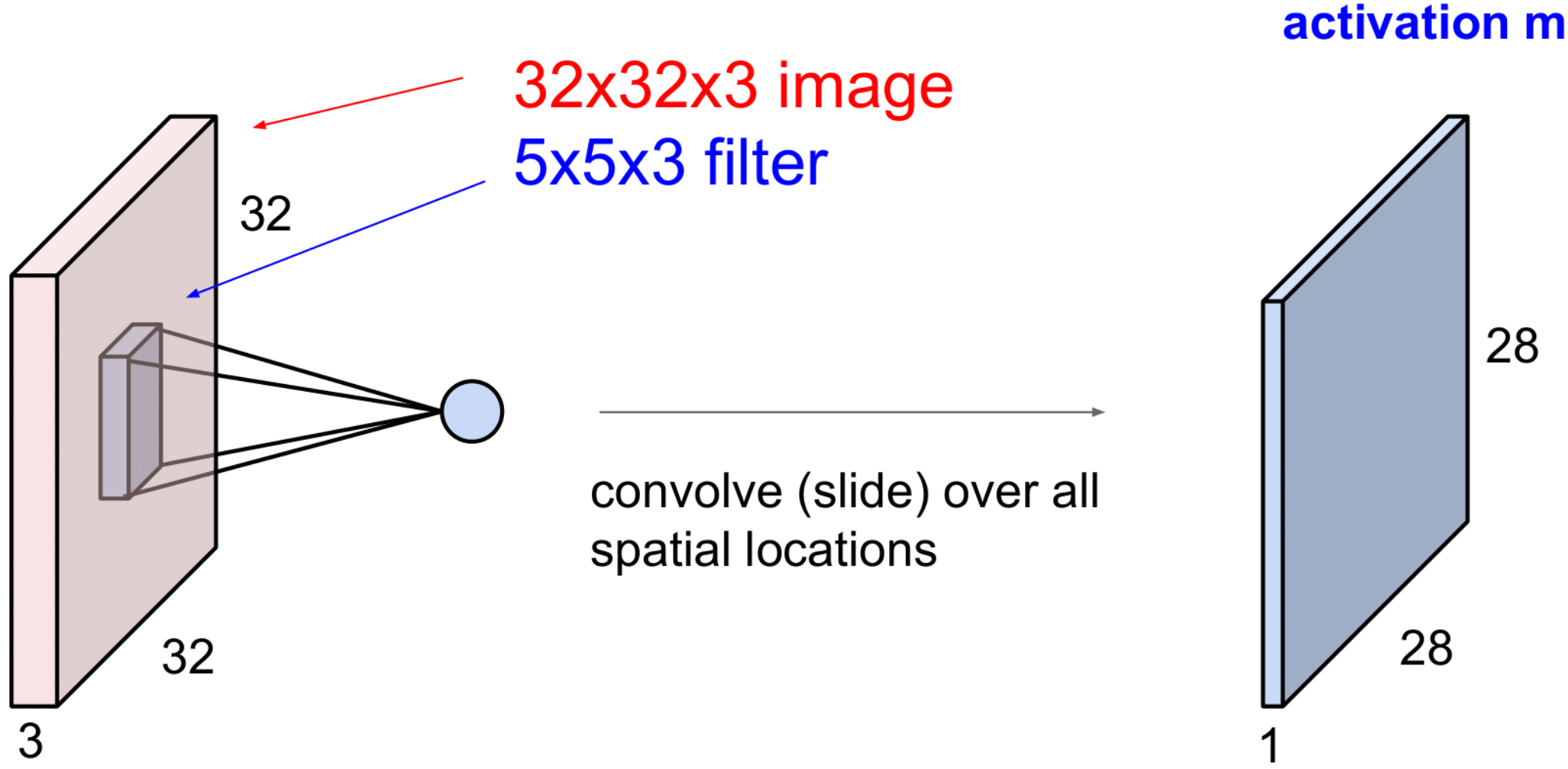
5x5x3 filter w

1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$

Source: Stanford CS231n Lecture 5 2017 by Fei-Fei Li & Justin Johnson & Serena Yeung

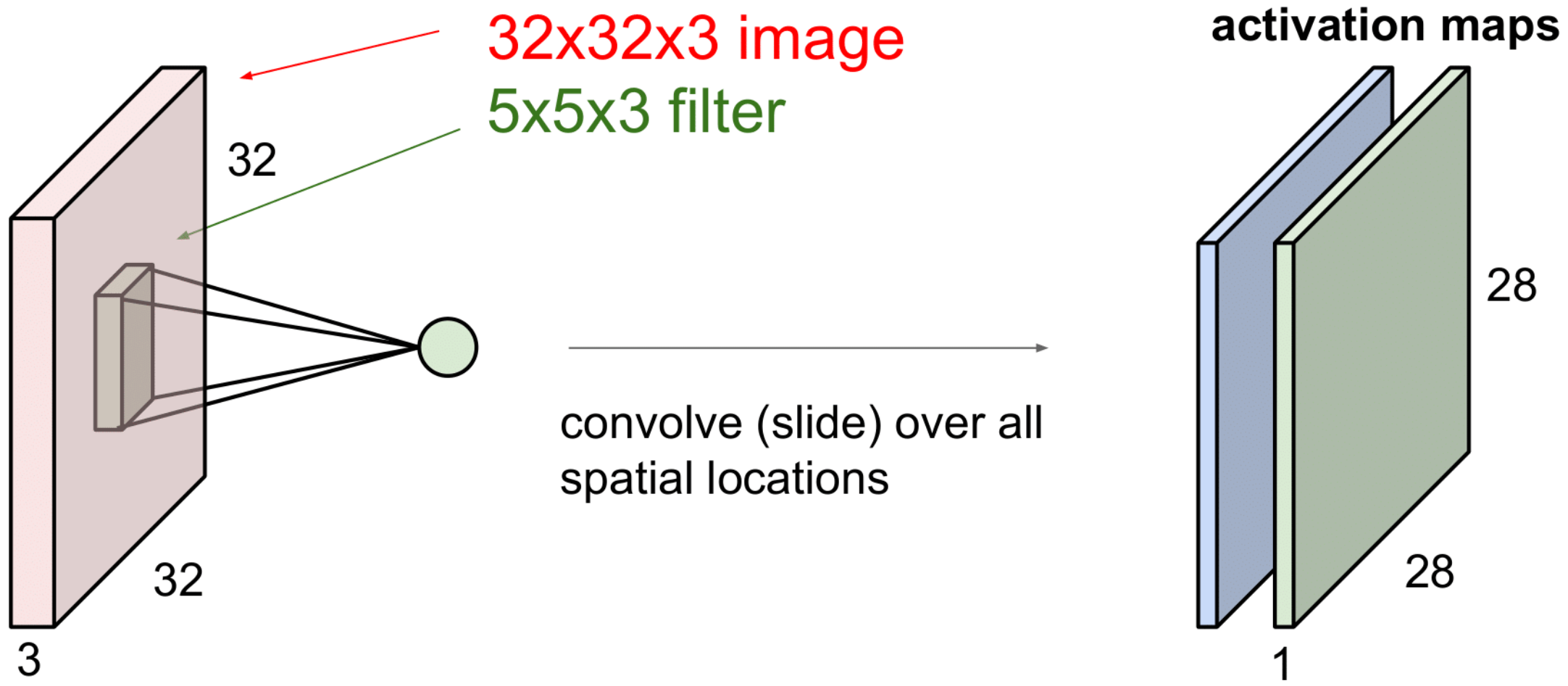
Convolution Layer



Source: Stanford CS231n Lecture 5 2017 by Fei-Fei Li & Justin Johnson & Serena Yeung

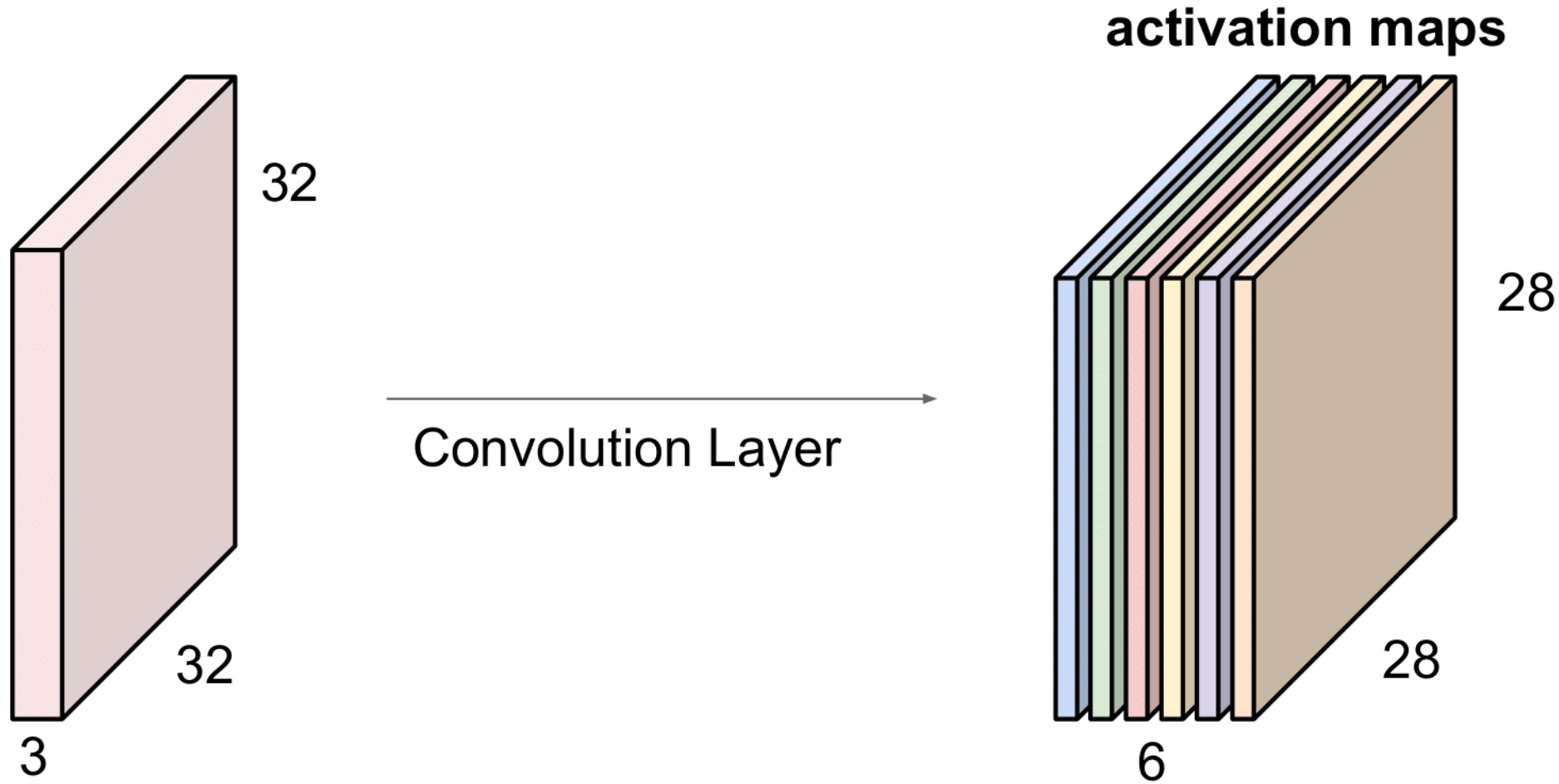
Convolution Layer

consider a second, green filter



Source: Stanford CS231n Lecture 5 2017 by Fei-Fei Li & Justin Johnson & Serena Yeung

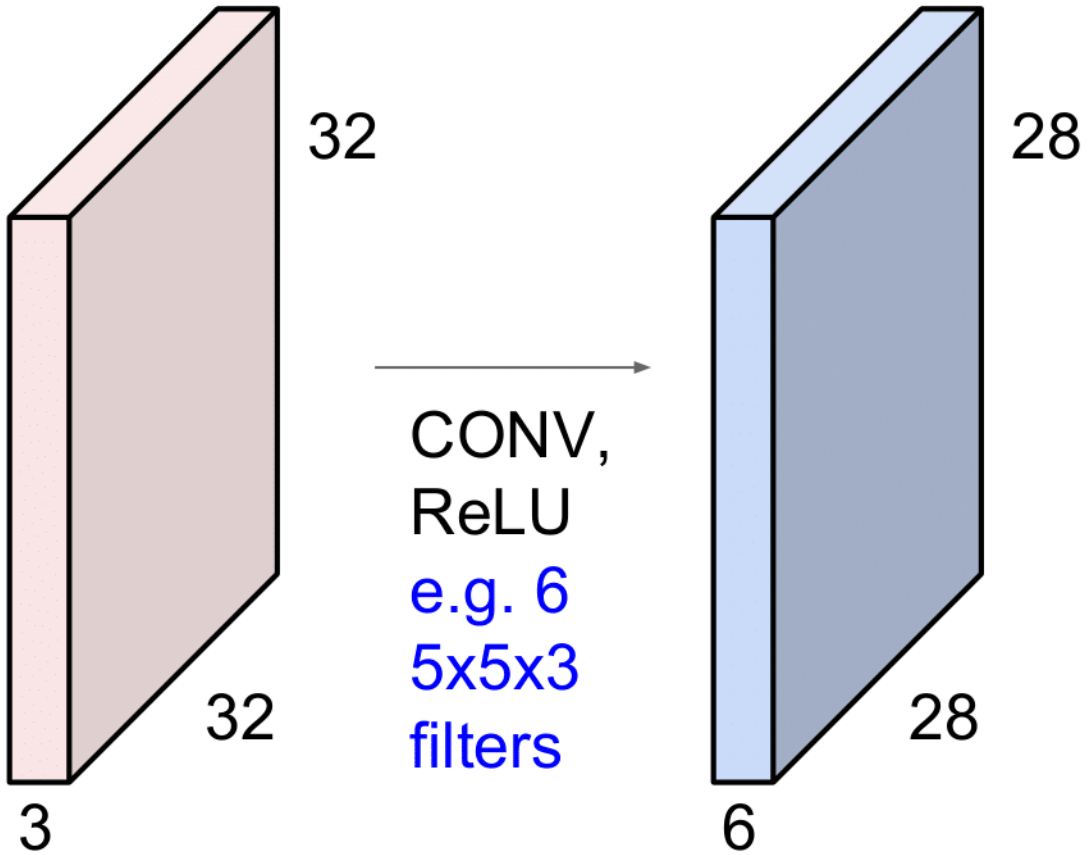
For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



We stack these up to get a "new image" of size 28x28x6!

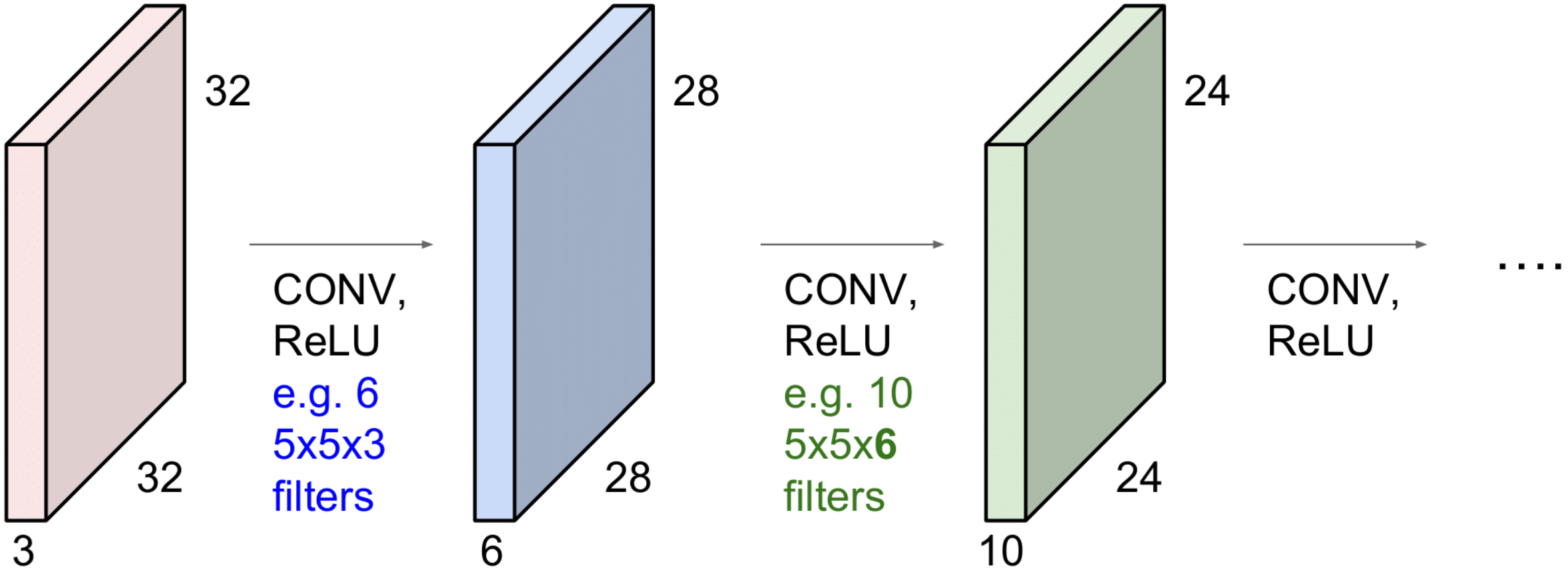
Source: Stanford CS231n Lecture 5 2017 by Fei-Fei Li & Justin Johnson & Serena Yeung

Preview: ConvNet is a sequence of Convolution Layers, interspersed with activation functions



Source: Stanford CS231n Lecture 5 2017 by Fei-Fei Li & Justin Johnson & Serena Yeung

Preview: ConvNet is a sequence of Convolutional Layers, interspersed with activation functions



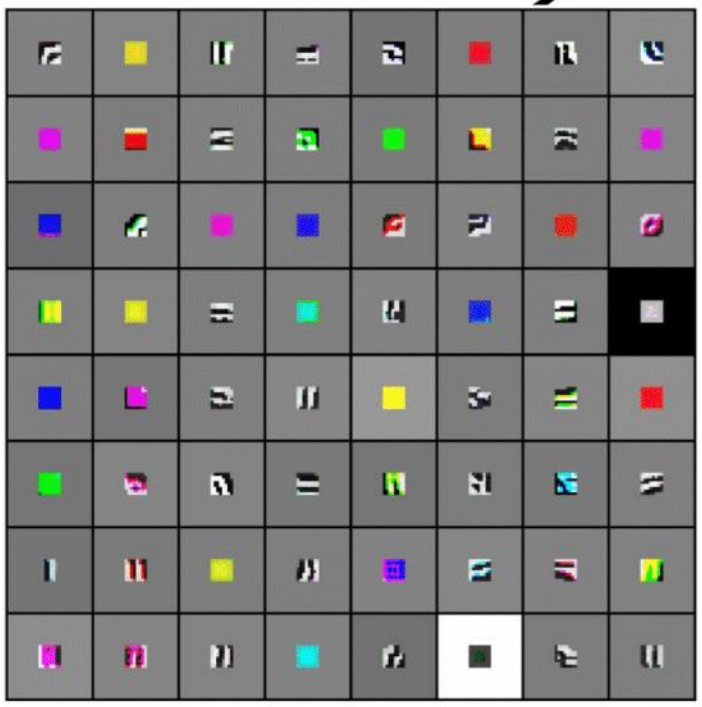
Source: Stanford CS231n Lecture 5 2017 by Fei-Fei Li & Justin Johnson & Serena Yeung

Preview

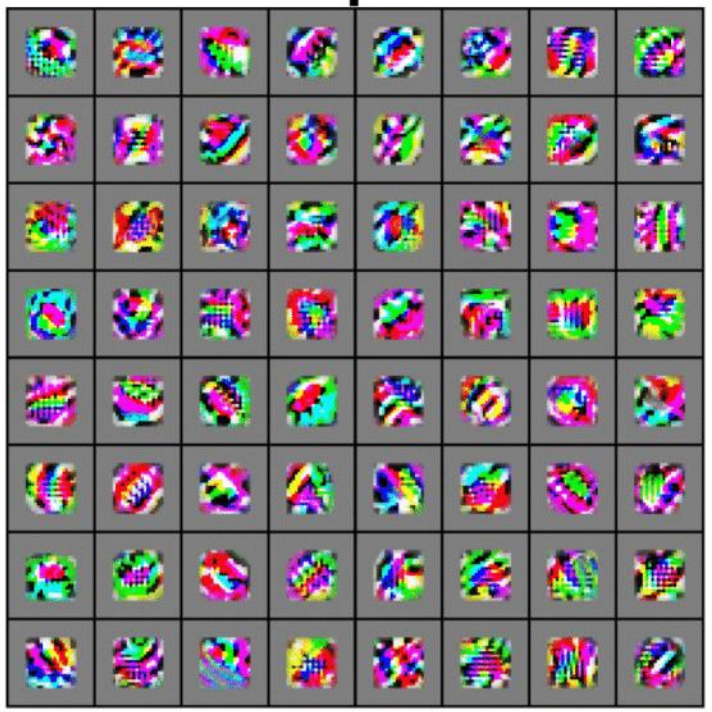
[Zeiler and Fergus 2013]

Visualization of VGG-16 by Lane McIntosh. VGG-16 architecture from [Simonyan and Zisserman 2014].

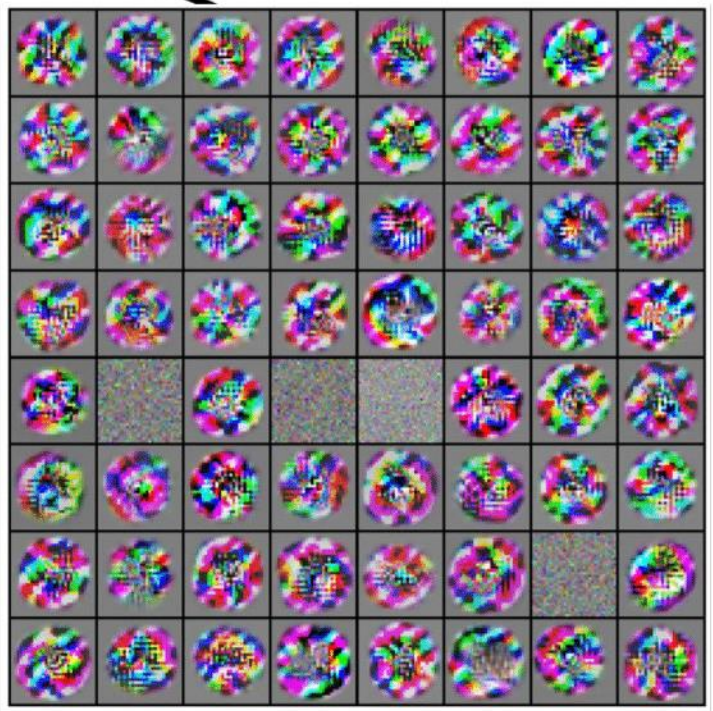
Low-level features → Mid-level features → High-level features → Linearly separable classifier



VGG-16 Conv1_1

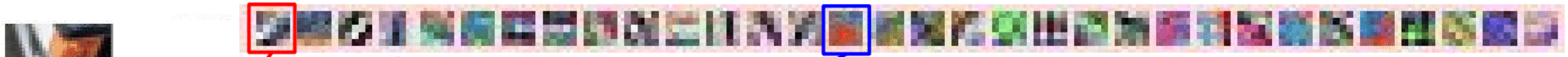


VGG-16 Conv3_2



VGG-16 Conv5_3

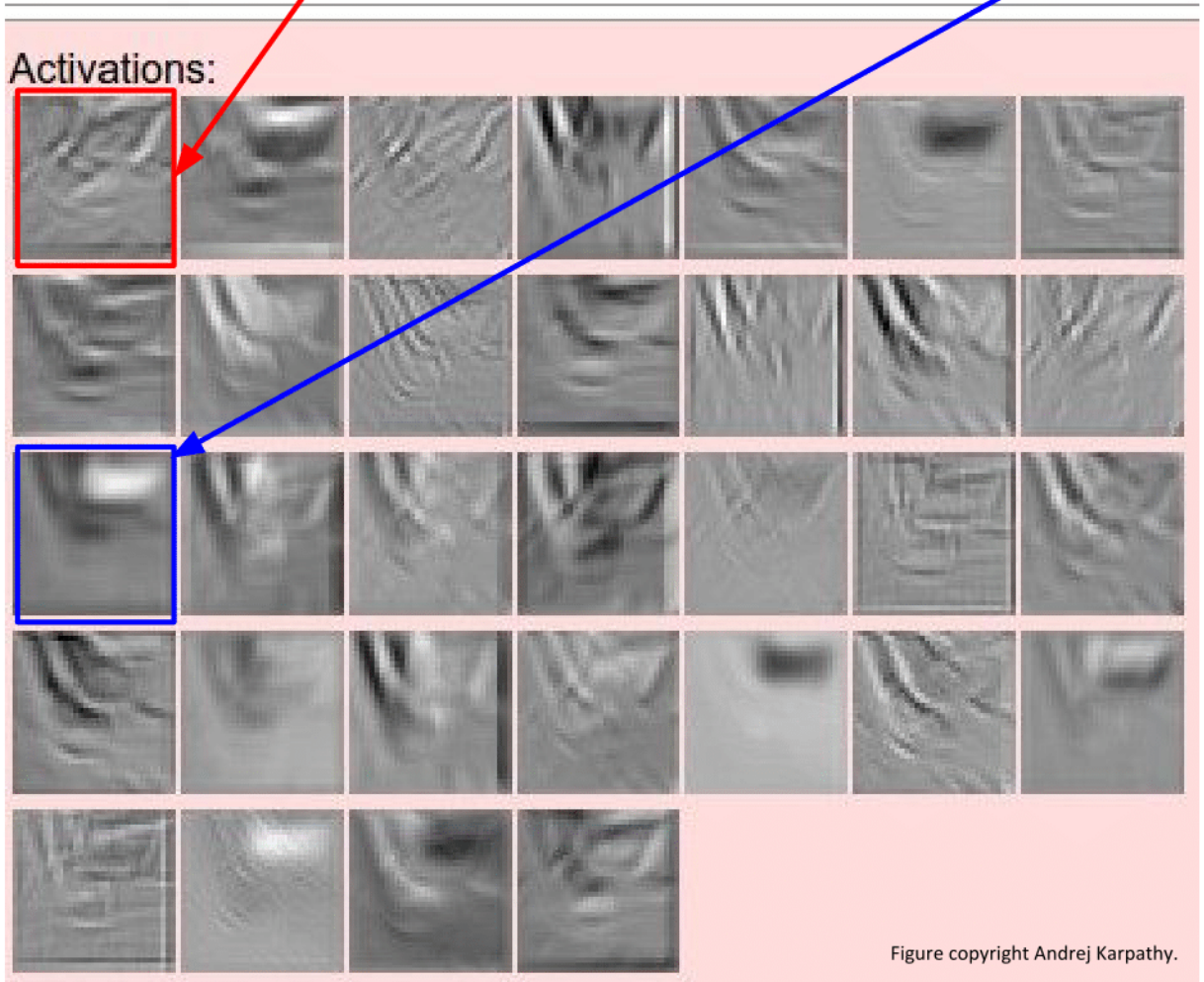
Source: Stanford CS231n Lecture 5 2017 by Fei-Fei Li & Justin Johnson & Serena Yeung



one filter =>

one activation map

example 5x5 filters
(32 total)



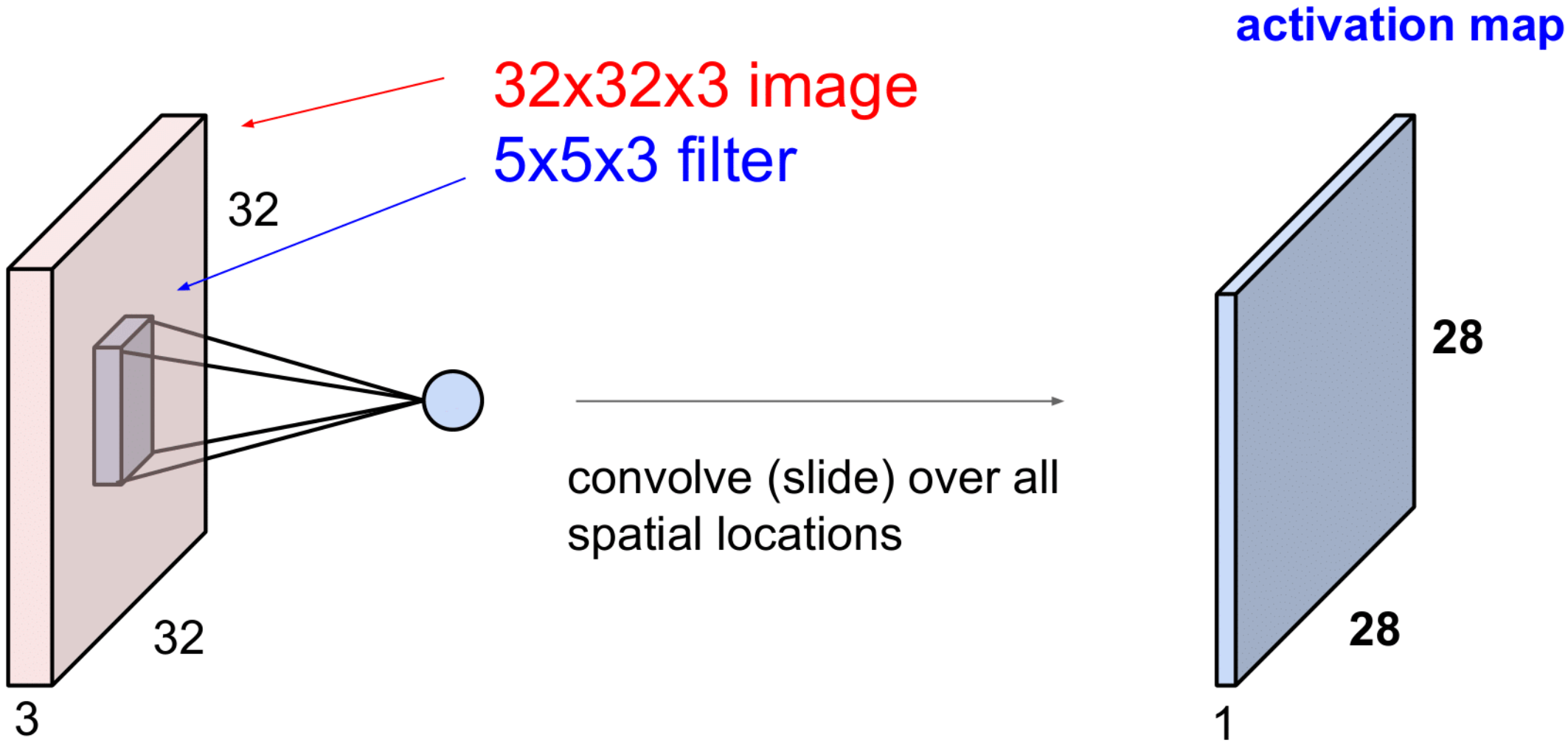
We call the layer convolutional because it is related to convolution of two signals:

$$f[x,y] * g[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1,n_2] \cdot g[x-n_1, y-n_2]$$

elementwise multiplication and sum of a filter and the signal (image)

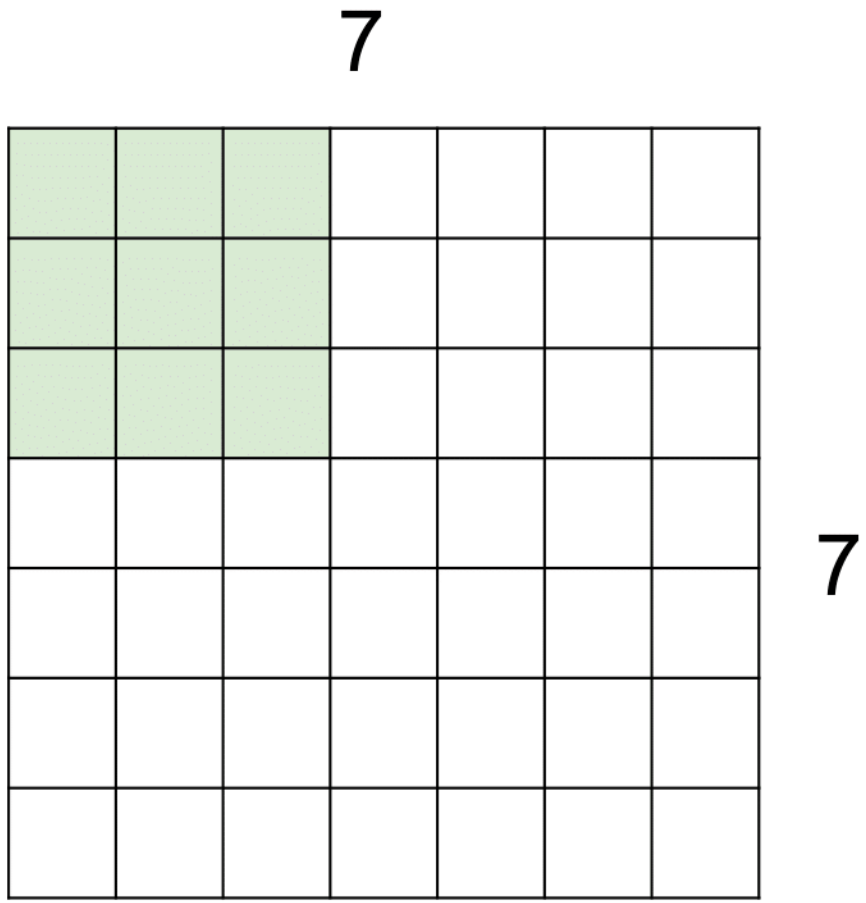
Source: Stanford CS231n Lecture 5 2017 by Fei-Fei Li & Justin Johnson & Serena Yeung

A closer look at spatial dimensions:



Source: Stanford CS231n Lecture 5 2017 by Fei-Fei Li & Justin Johnson & Serena Yeung

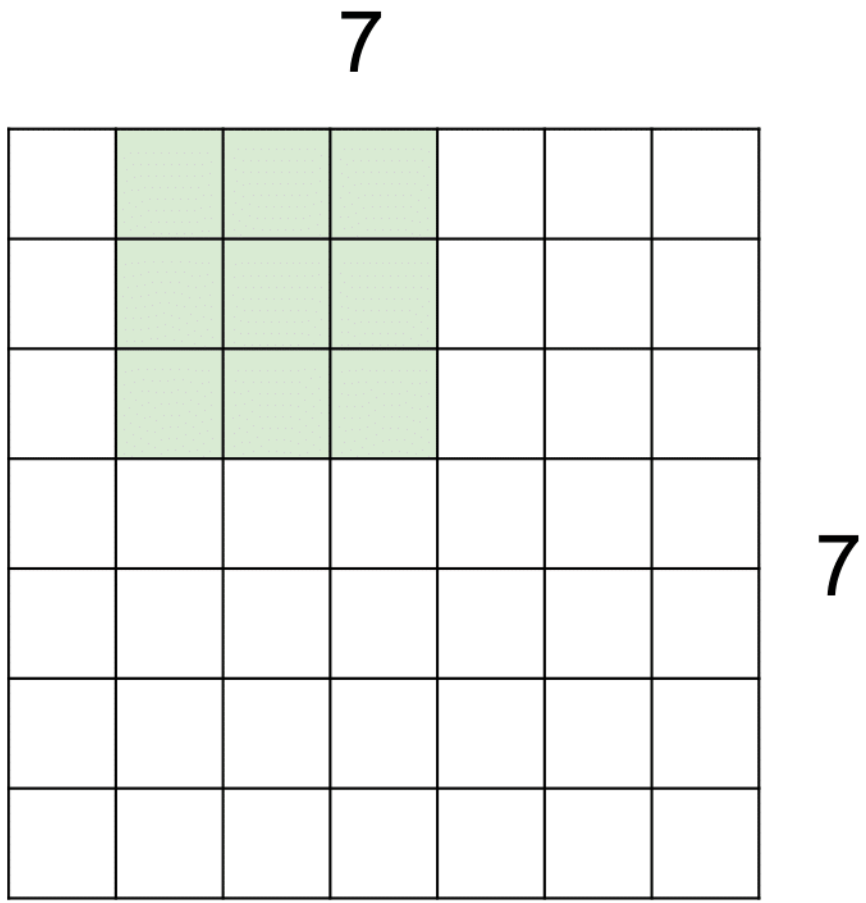
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

Source: Stanford CS231n Lecture 5 2017 by Fei-Fei Li & Justin Johnson & Serena Yeung

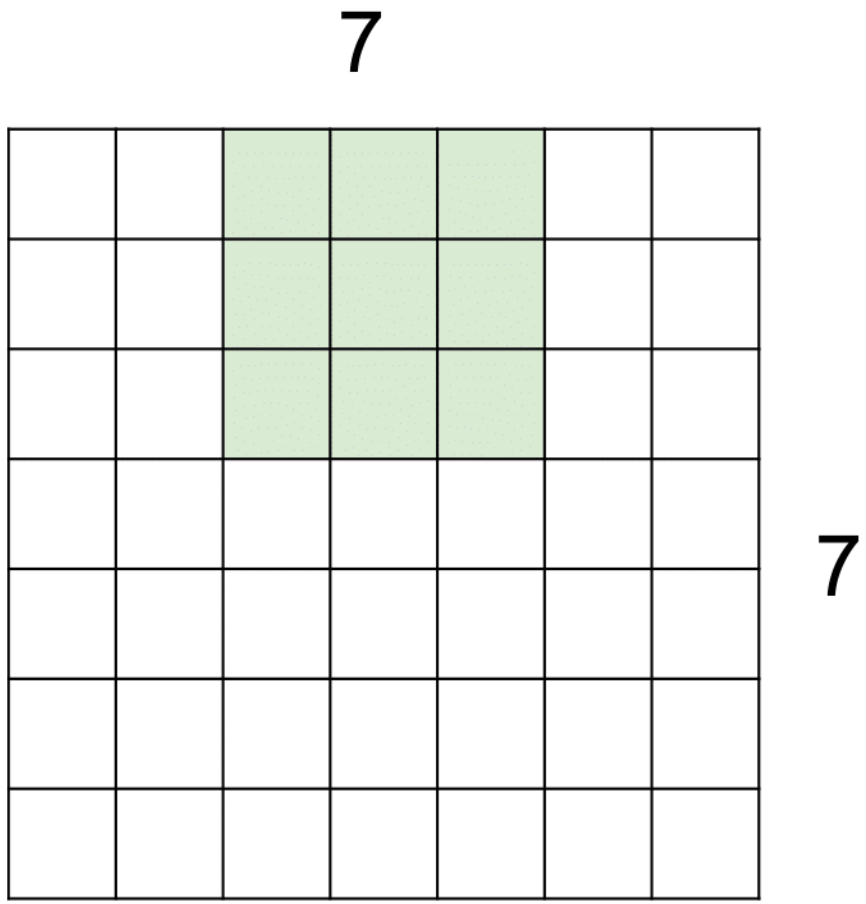
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

Source: Stanford CS231n Lecture 5 2017 by Fei-Fei Li & Justin Johnson & Serena Yeung

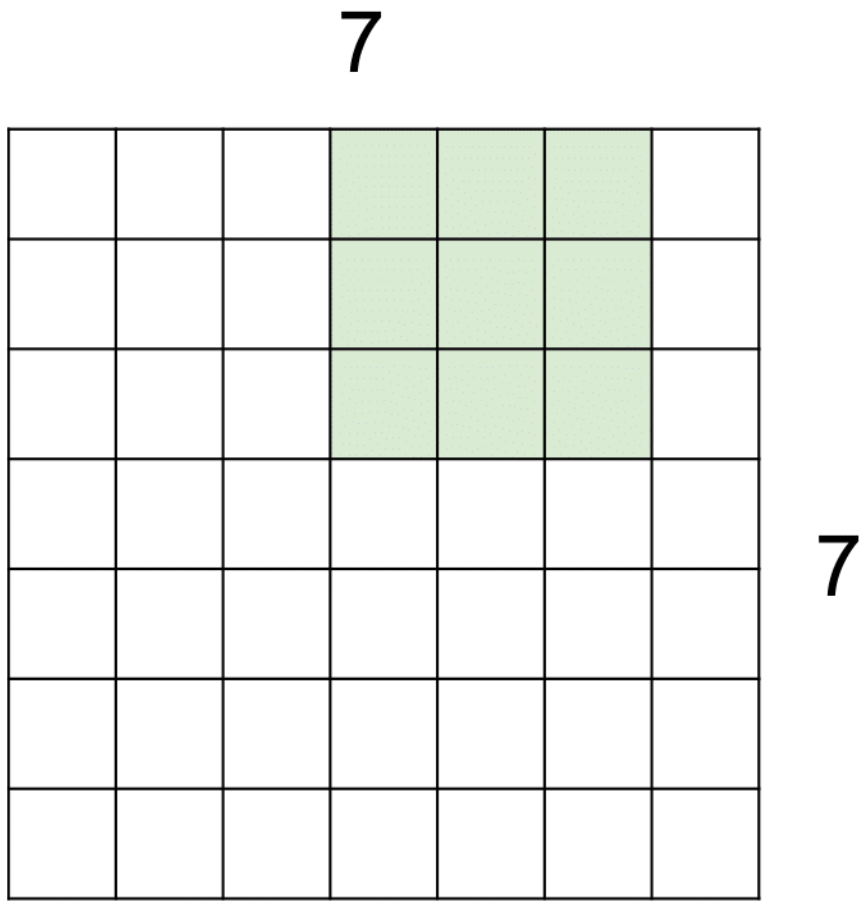
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

Source: Stanford CS231n Lecture 5 2017 by Fei-Fei Li & Justin Johnson & Serena Yeung

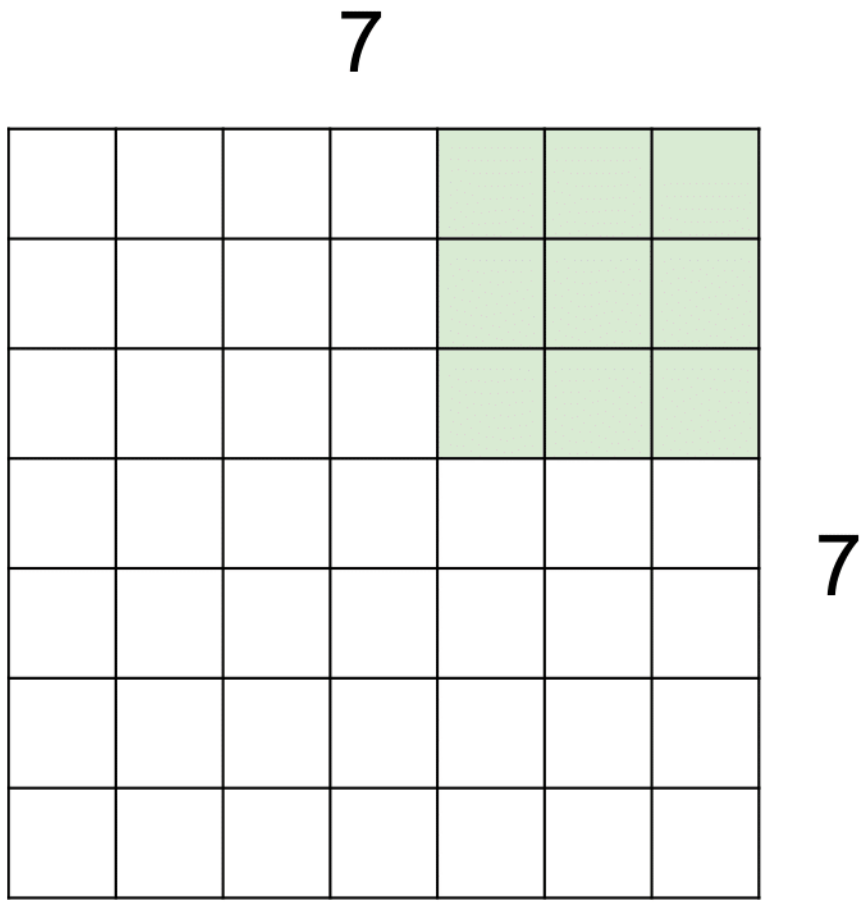
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

Source: Stanford CS231n Lecture 5 2017 by Fei-Fei Li & Justin Johnson & Serena Yeung

A closer look at spatial dimensions:

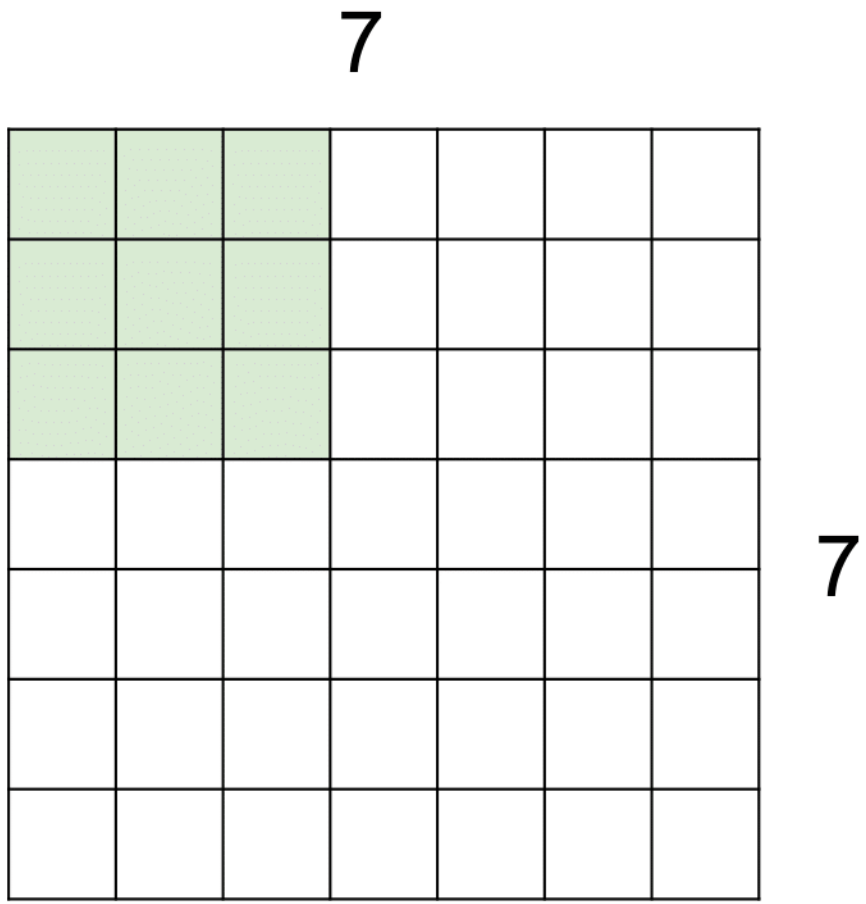


7x7 input (spatially)
assume 3x3 filter

=> 5x5 output

Source: Stanford CS231n Lecture 5 2017 by Fei-Fei Li & Justin Johnson & Serena Yeung

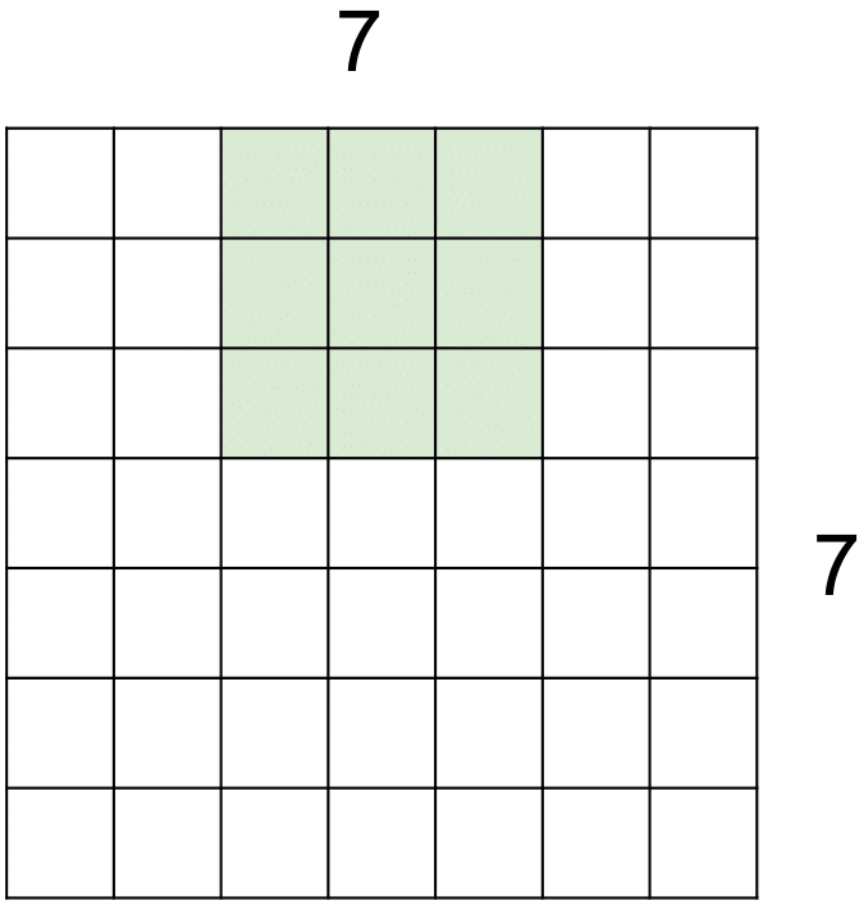
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

Source: Stanford CS231n Lecture 5 2017 by Fei-Fei Li & Justin Johnson & Serena Yeung

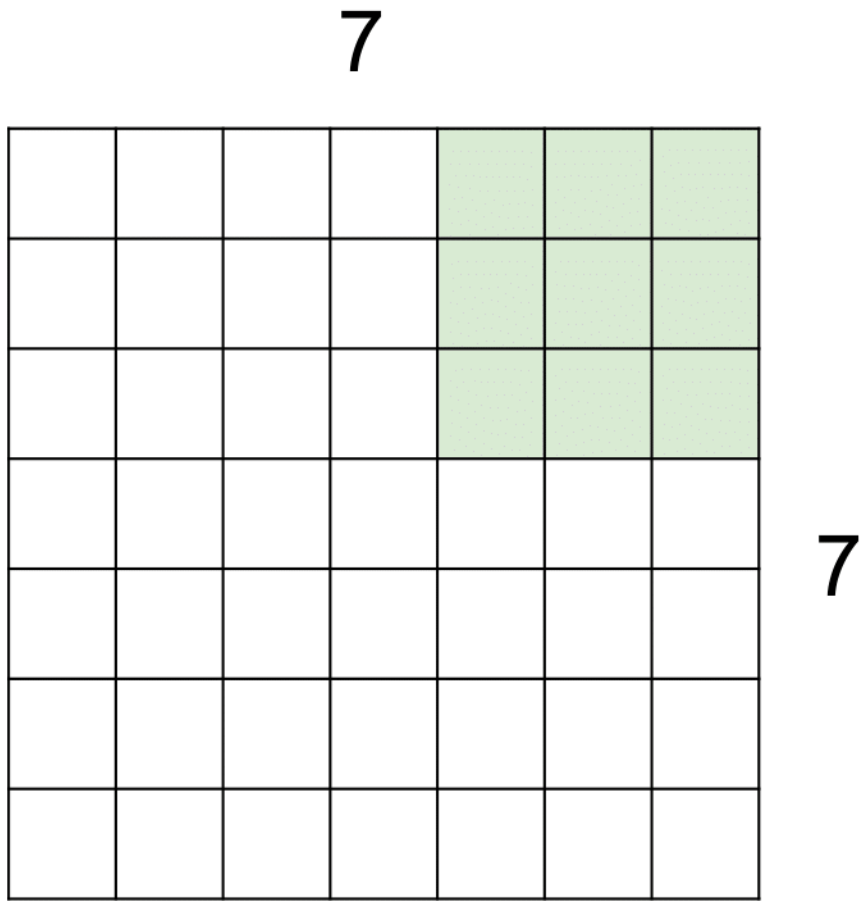
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

Source: Stanford CS231n Lecture 5 2017 by Fei-Fei Li & Justin Johnson & Serena Yeung

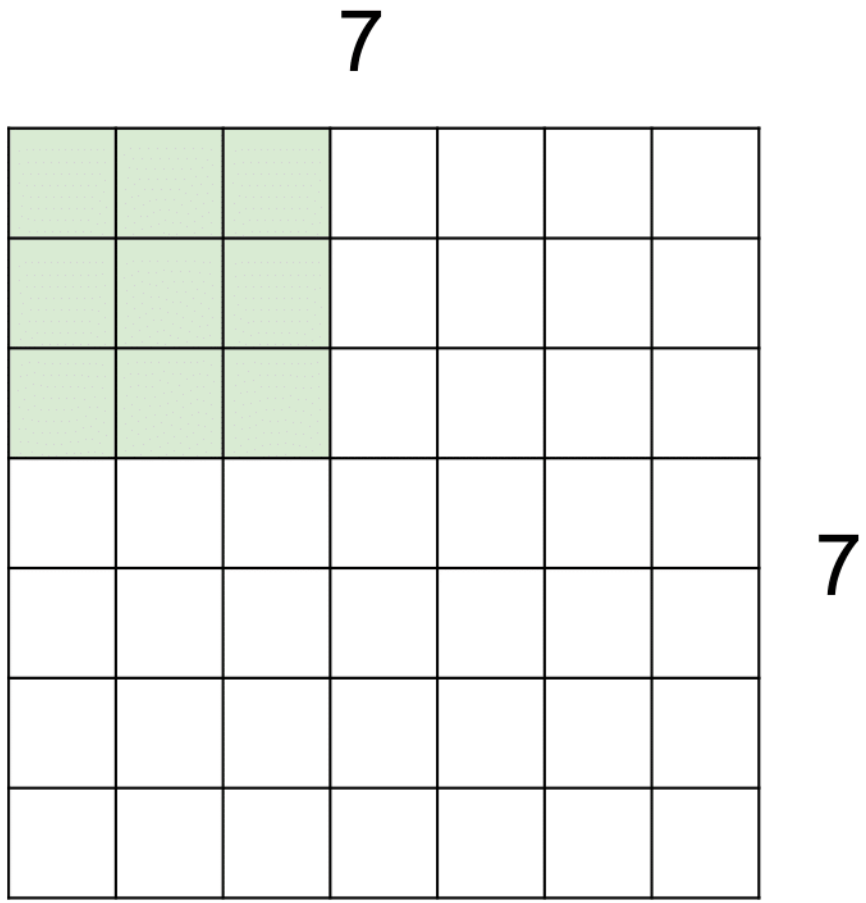
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**
=> 3x3 output!

Source: Stanford CS231n Lecture 5 2017 by Fei-Fei Li & Justin Johnson & Serena Yeung

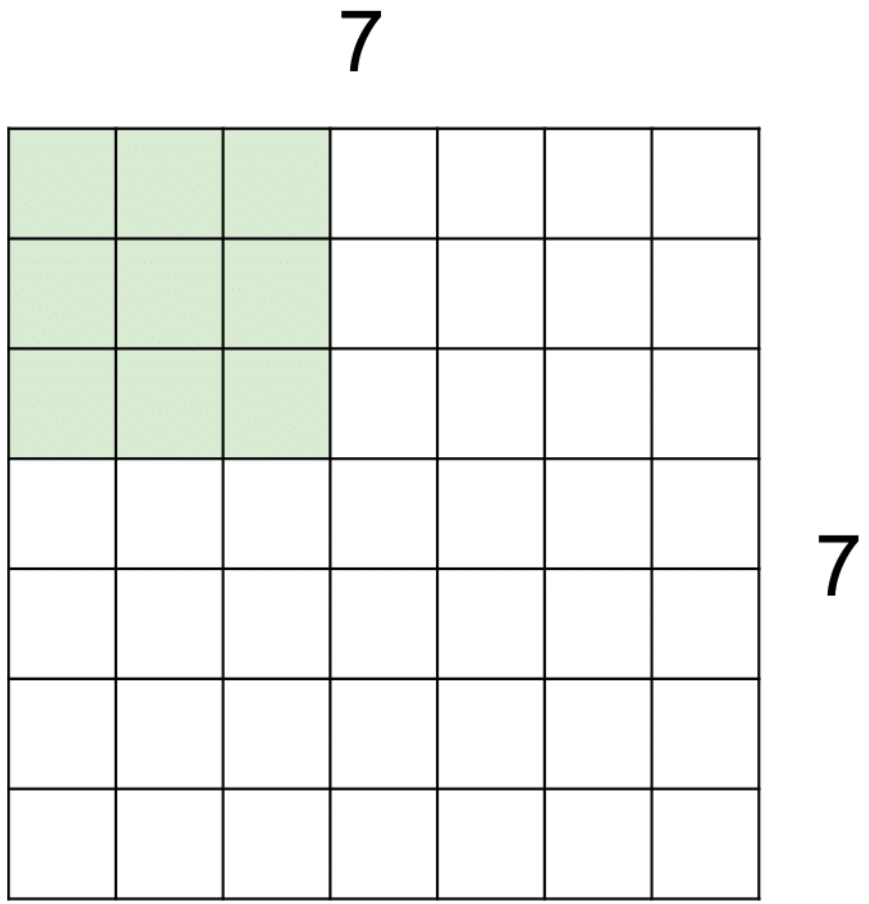
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter
applied **with stride 3?**

Source: Stanford CS231n Lecture 5 2017 by Fei-Fei Li & Justin Johnson & Serena Yeung

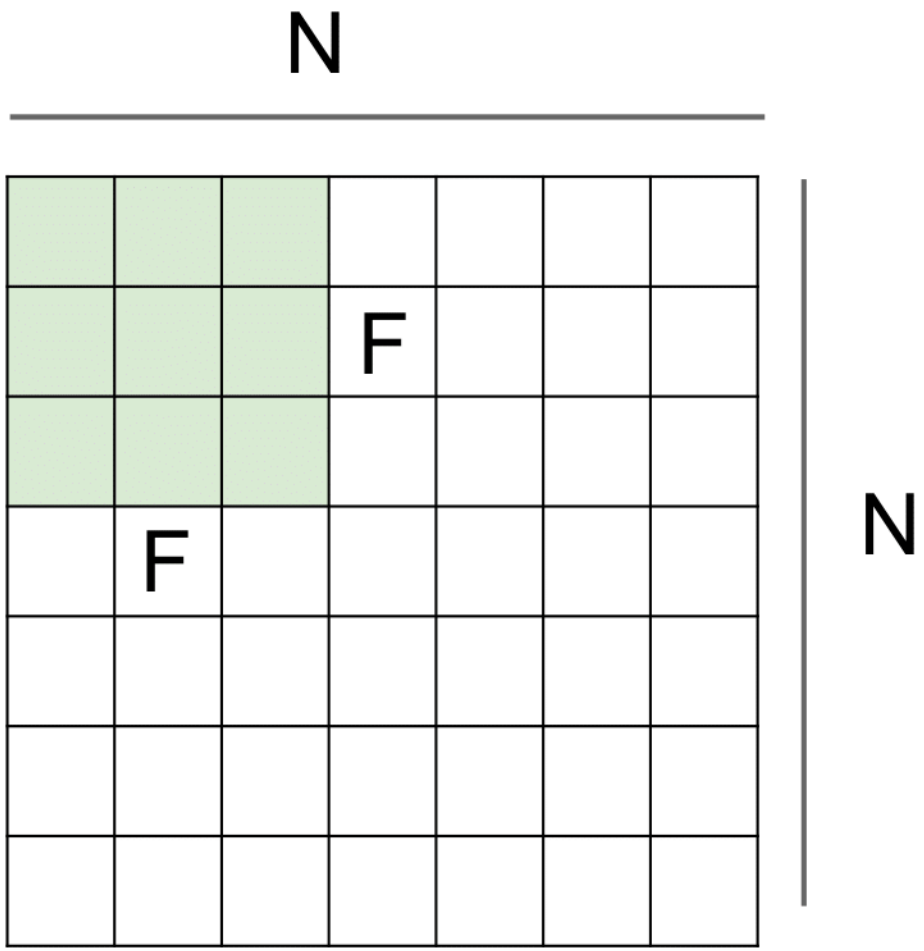
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter
applied **with stride 3?**

doesn't fit!
cannot apply 3x3 filter on
7x7 input with stride 3.

Source: Stanford CS231n Lecture 5 2017 by Fei-Fei Li & Justin Johnson & Serena Yeung

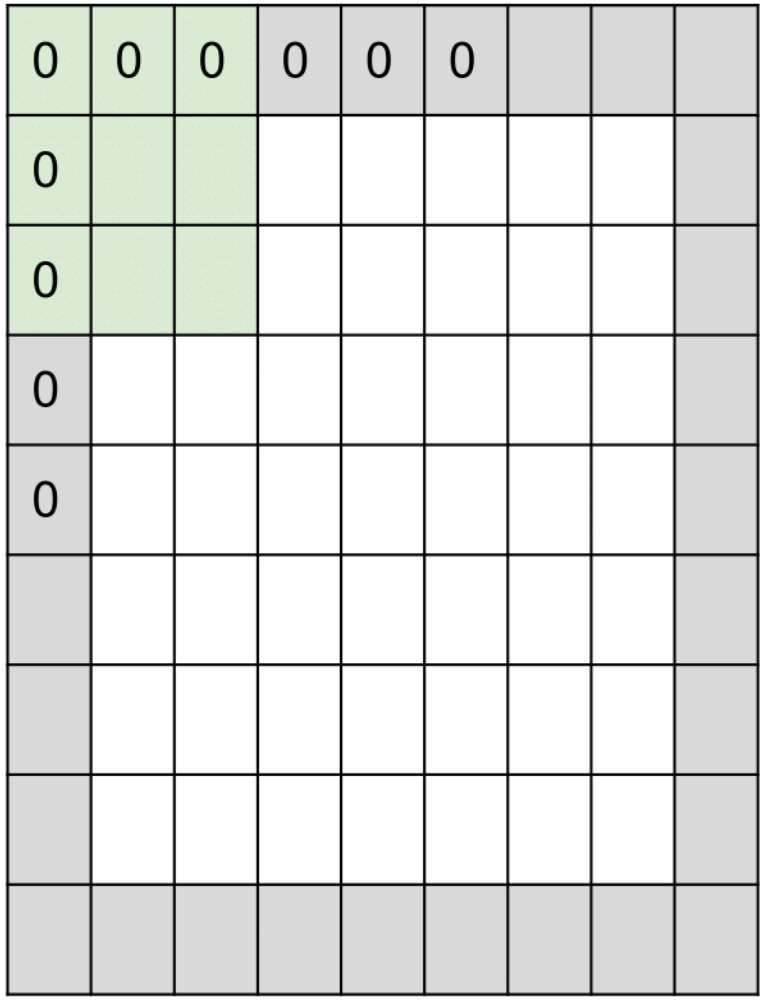


Output size:
(N - F) / stride + 1

e.g. N = 7, F = 3:
stride 1 => (7 - 3)/1 + 1 = 5
stride 2 => (7 - 3)/2 + 1 = 3
stride 3 => (7 - 3)/3 + 1 = 2.33 :\

Source: Stanford CS231n Lecture 5 2017 by Fei-Fei Li & Justin Johnson & Serena Yeung

In practice: Common to zero pad the border

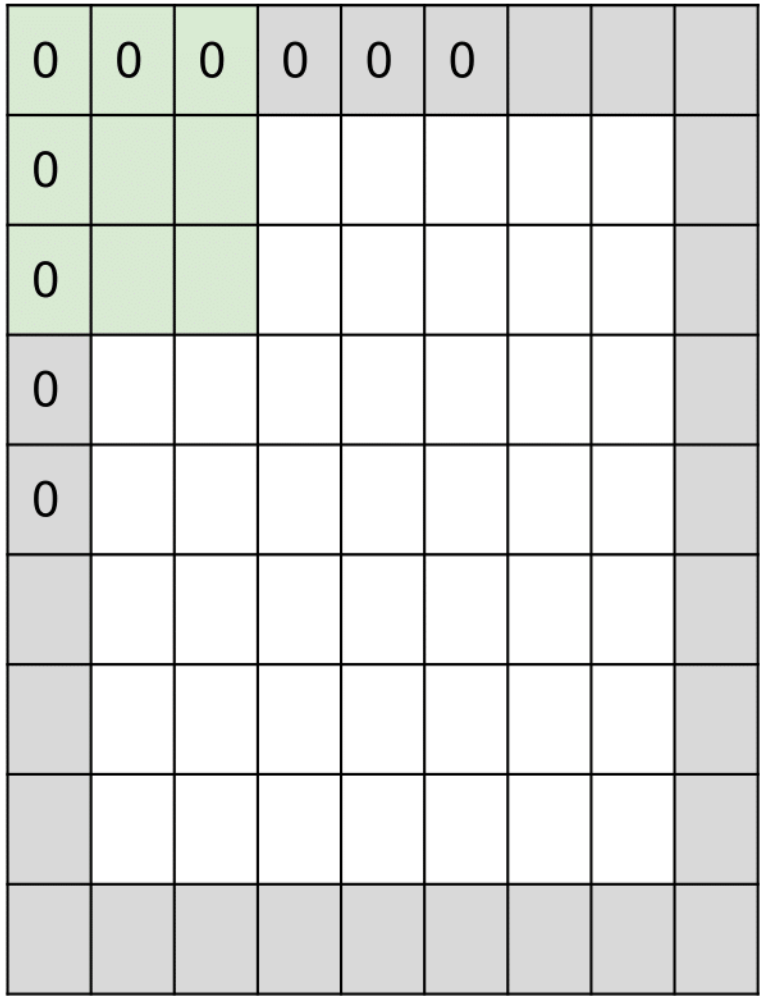


e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

(recall:)
(N - F) / stride + 1

Source: Stanford CS231n Lecture 5 2017 by Fei-Fei Li & Justin Johnson & Serena Yeung

In practice: Common to zero pad the border

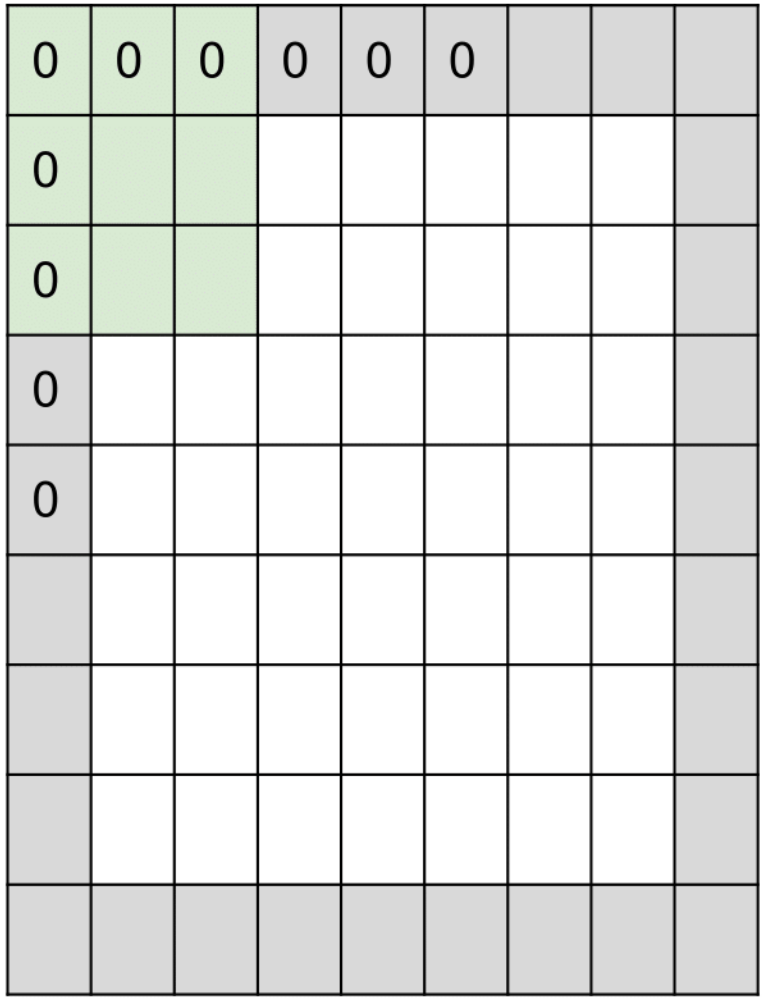


e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!

Source: Stanford CS231n Lecture 5 2017 by Fei-Fei Li & Justin Johnson & Serena Yeung

In practice: Common to zero pad the border

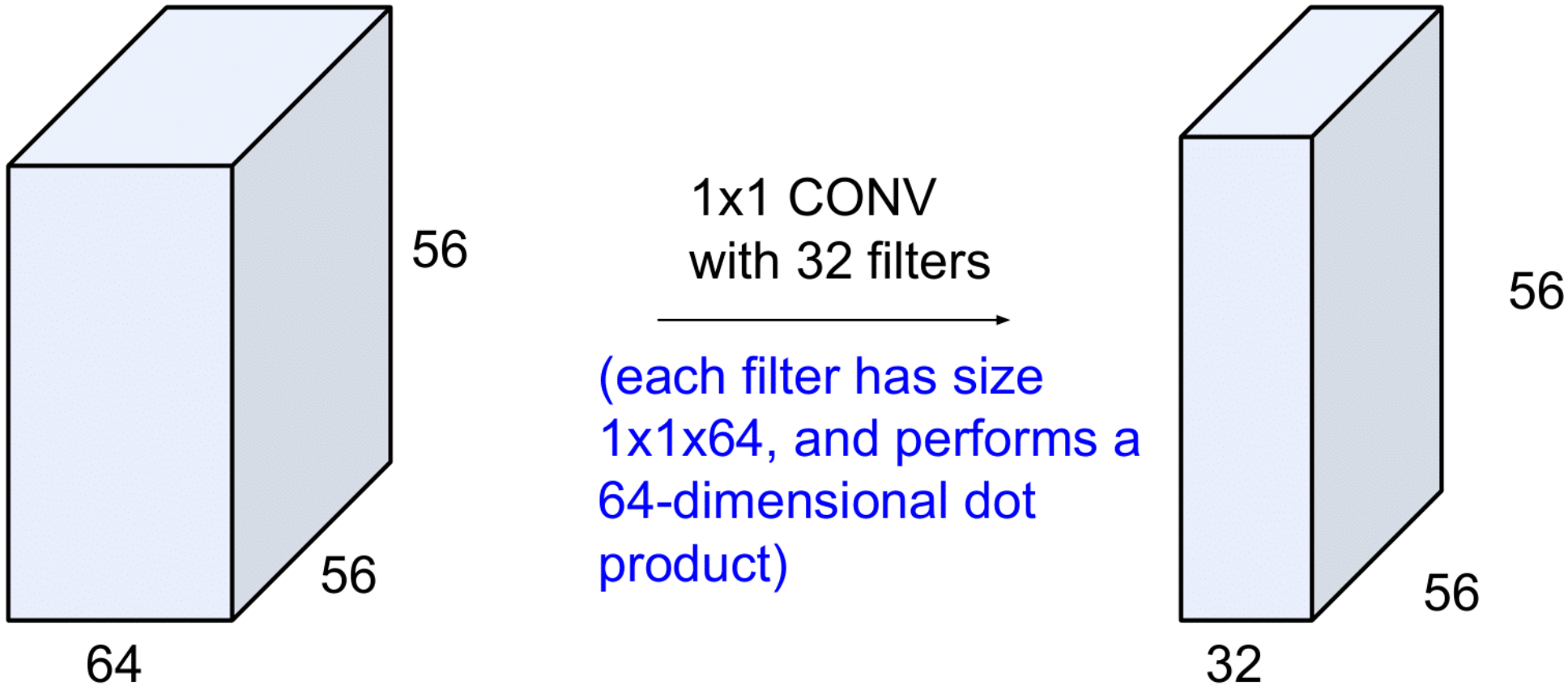


e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!
in general, common to see CONV layers with stride 1, filters of size FxF, and zero-padding with (F-1)/2. (will preserve size spatially)
e.g. F = 3 => zero pad with 1
F = 5 => zero pad with 2
F = 7 => zero pad with 3

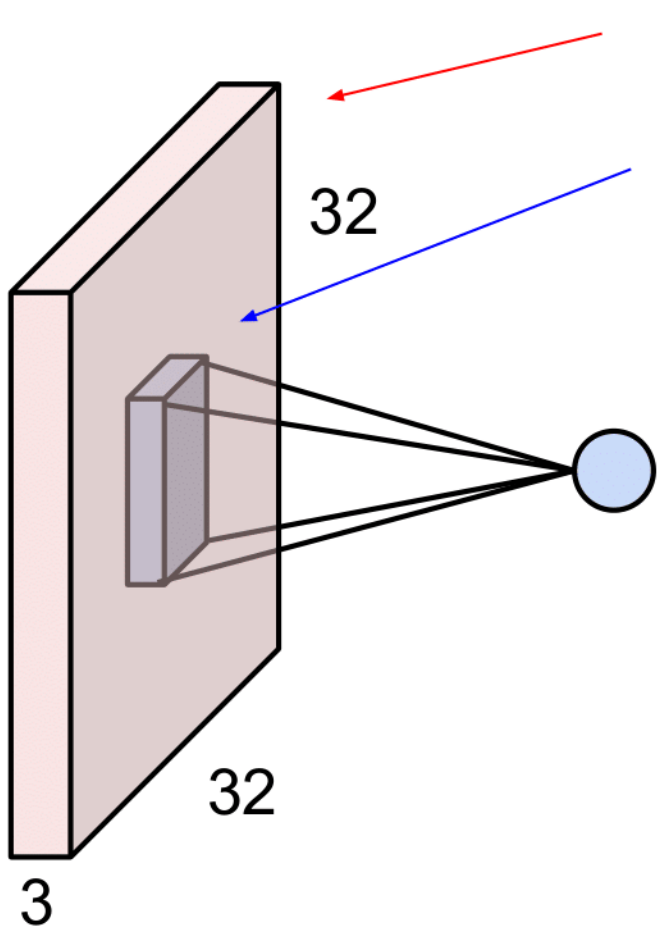
Source: Stanford CS231n Lecture 5 2017 by Fei-Fei Li & Justin Johnson & Serena Yeung

(btw, 1x1 convolution layers make perfect sense)



Source: Stanford CS231n Lecture 5 2017 by Fei-Fei Li & Justin Johnson & Serena Yeung

The brain/neuron view of CONV Layer



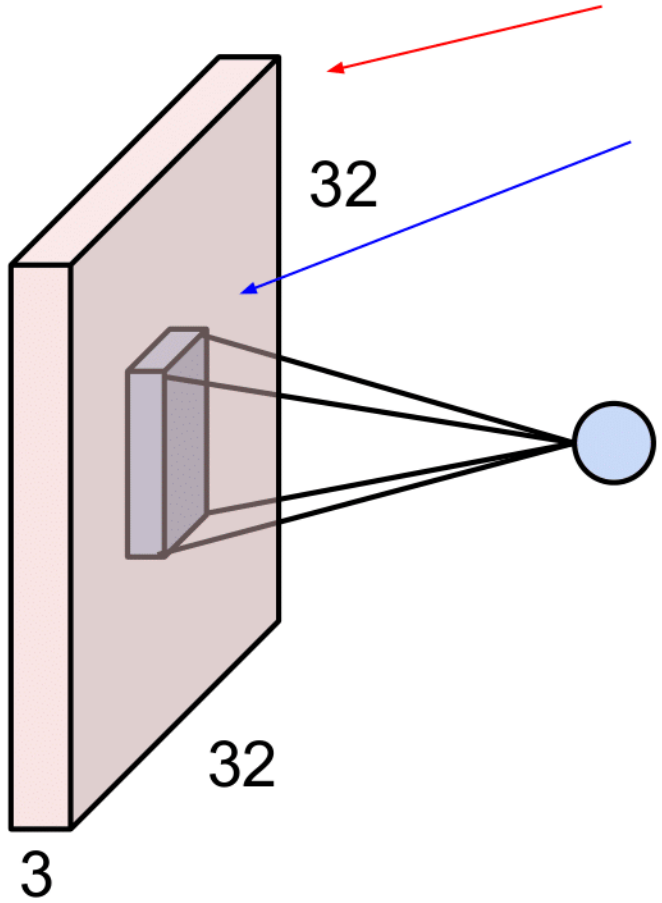
32x32x3 image

5x5x3 filter

1 number:
the result of taking a dot product between the filter and this part of the image
(i.e. 5*5*3 = 75-dimensional dot product)

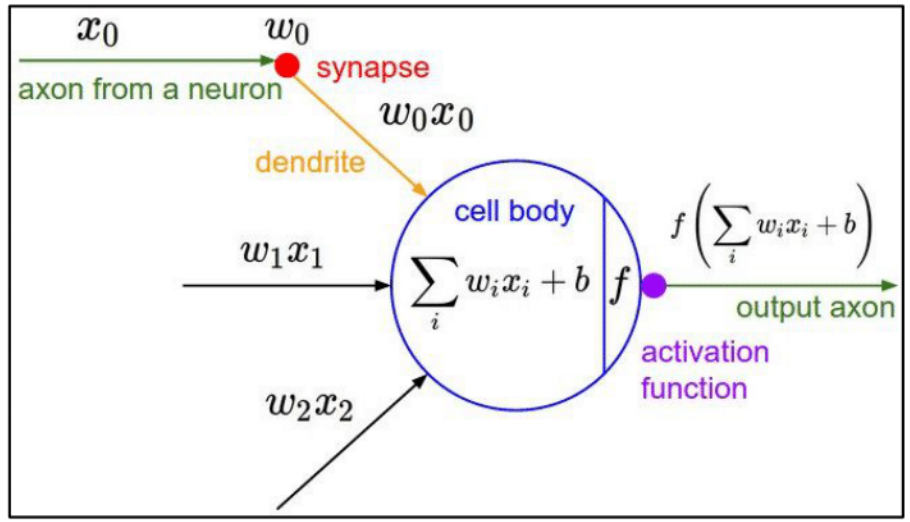
Source: Stanford CS231n Lecture 5 2017 by Fei-Fei Li & Justin Johnson & Serena Yeung

The brain/neuron view of CONV Layer



32x32x3 image

5x5x3 filter

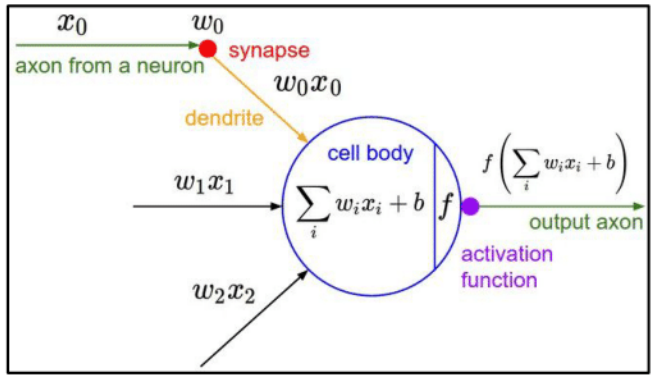


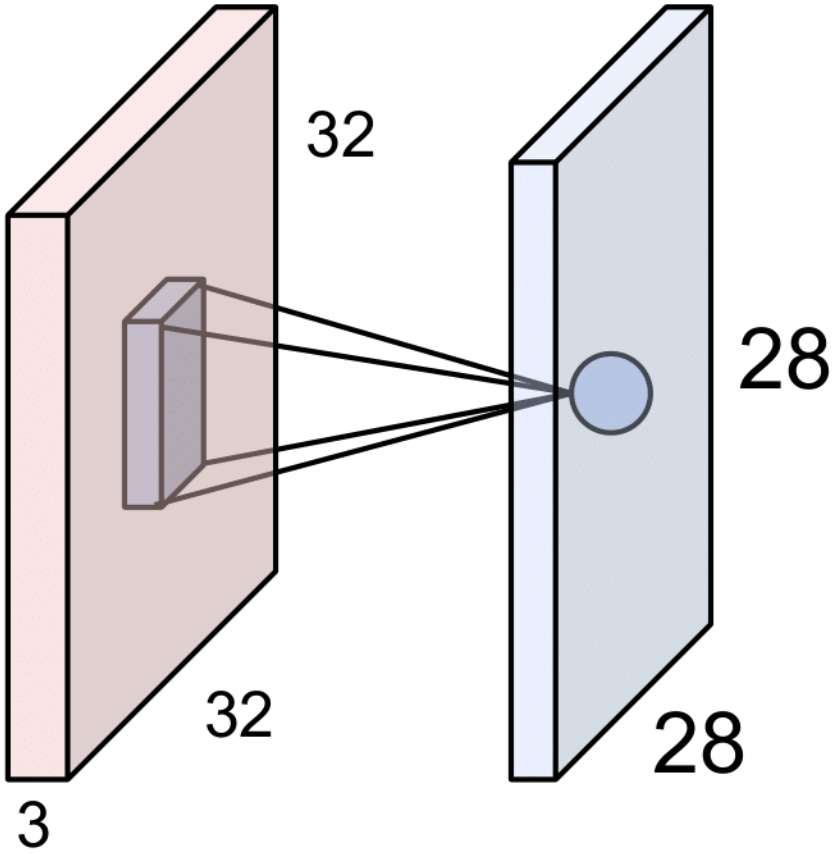
It's just a neuron with local connectivity...

1 number:
the result of taking a dot product between the filter and this part of the image
(i.e. 5*5*3 = 75-dimensional dot product)

Source: Stanford CS231n Lecture 5 2017 by Fei-Fei Li & Justin Johnson & Serena Yeung

The brain/neuron view of CONV Layer





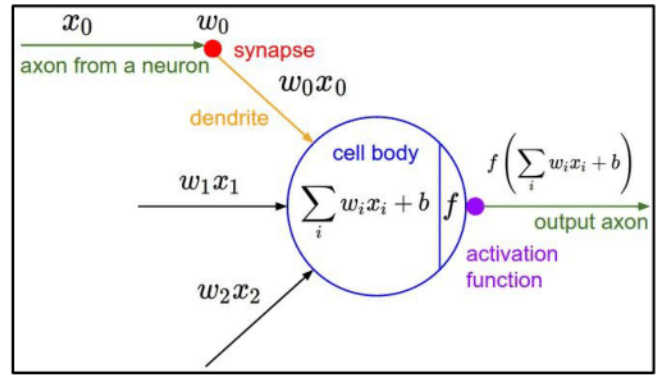
An activation map is a 28x28 sheet of neuron outputs:

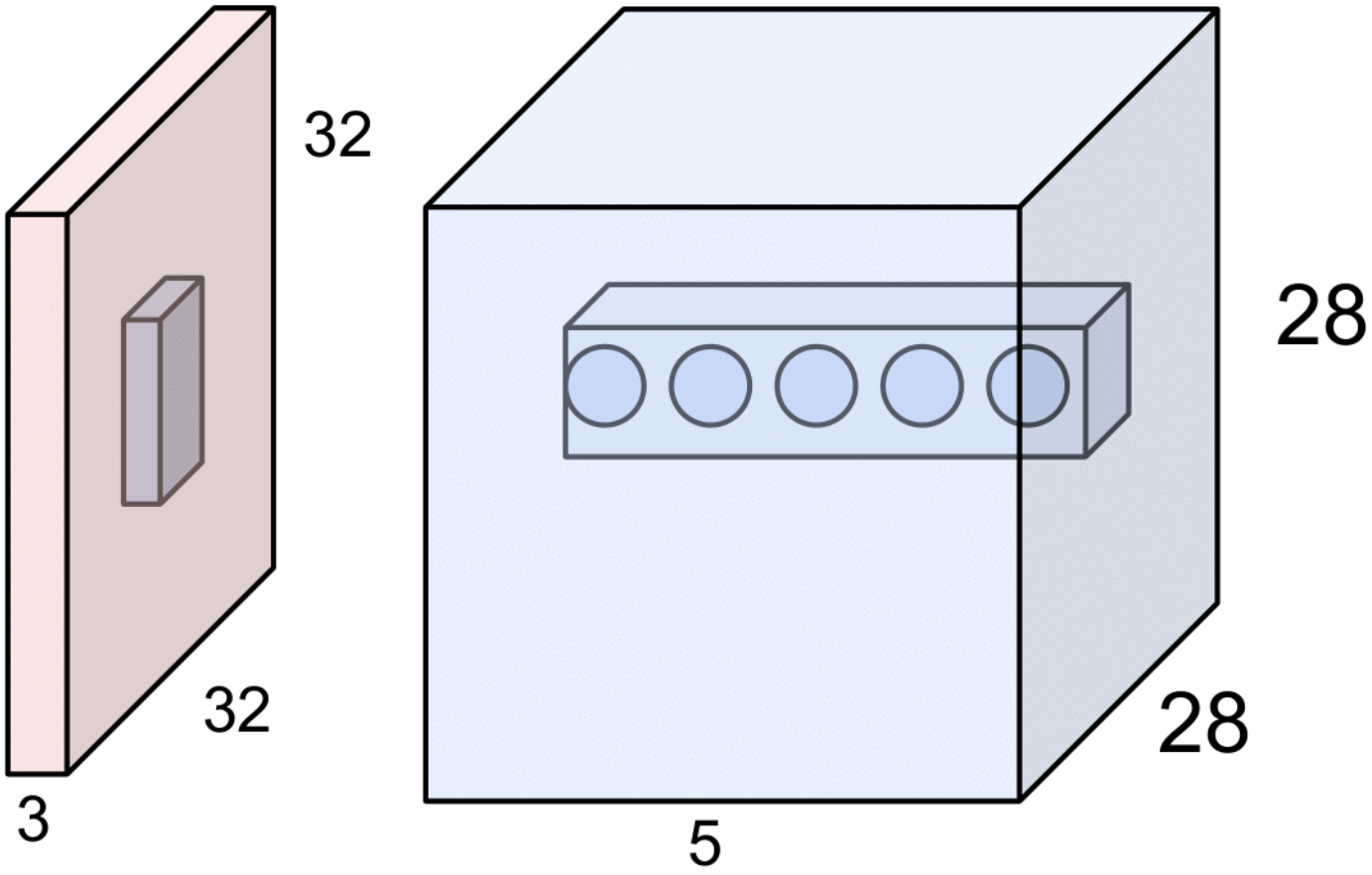
1. Each is connected to a small region in the input
2. All of them share parameters

"5x5 filter" -> "5x5 receptive field for each neuron"

Source: Stanford CS231n Lecture 5 2017 by Fei-Fei Li & Justin Johnson & Serena Yeung

The brain/neuron view of CONV Layer





E.g. with 5 filters,
CONV layer consists of neurons arranged in a 3D grid (28x28x5)

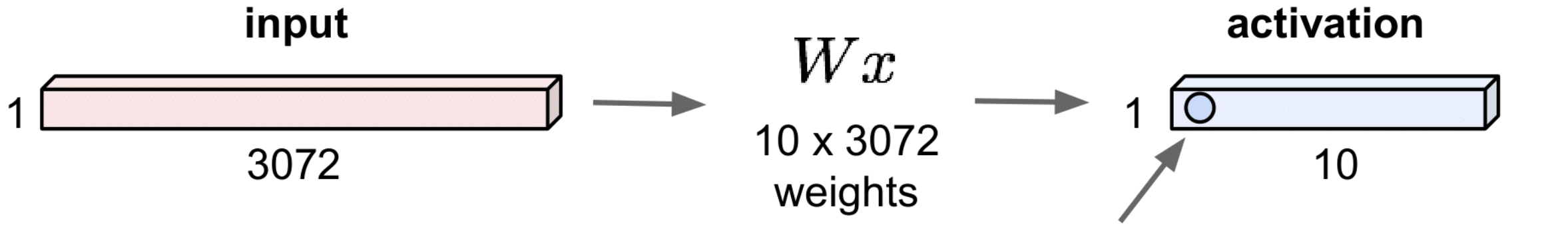
There will be 5 different neurons all looking at the same region in the input volume

Source: Stanford CS231n Lecture 5 2017 by Fei-Fei Li & Justin Johnson & Serena Yeung

Reminder: Fully Connected Layer

32x32x3 image -> stretch to 3072 x 1

Each neuron looks at the full input volume

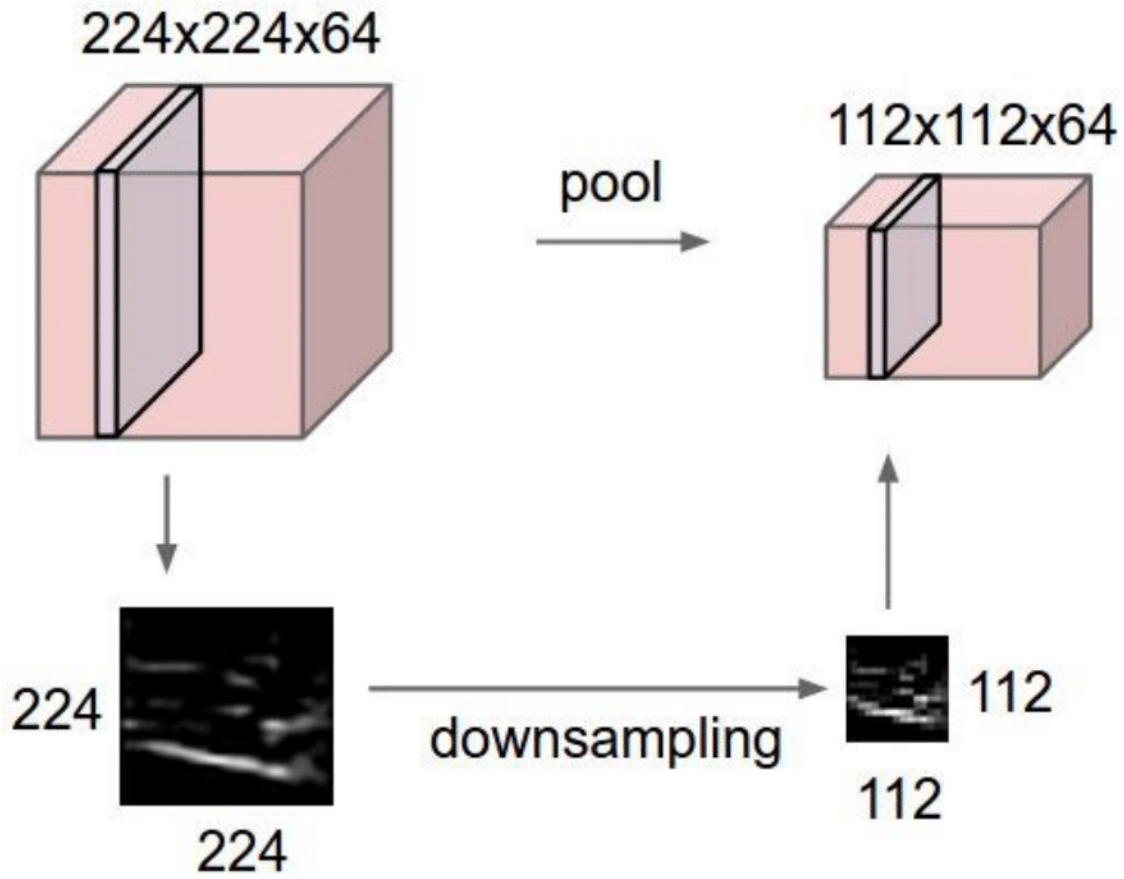


1 number:
the result of taking a dot product between a row of W and the input (a 3072-dimensional dot product)

Source: Stanford CS231n Lecture 5 2017 by Fei-Fei Li & Justin Johnson & Serena Yeung

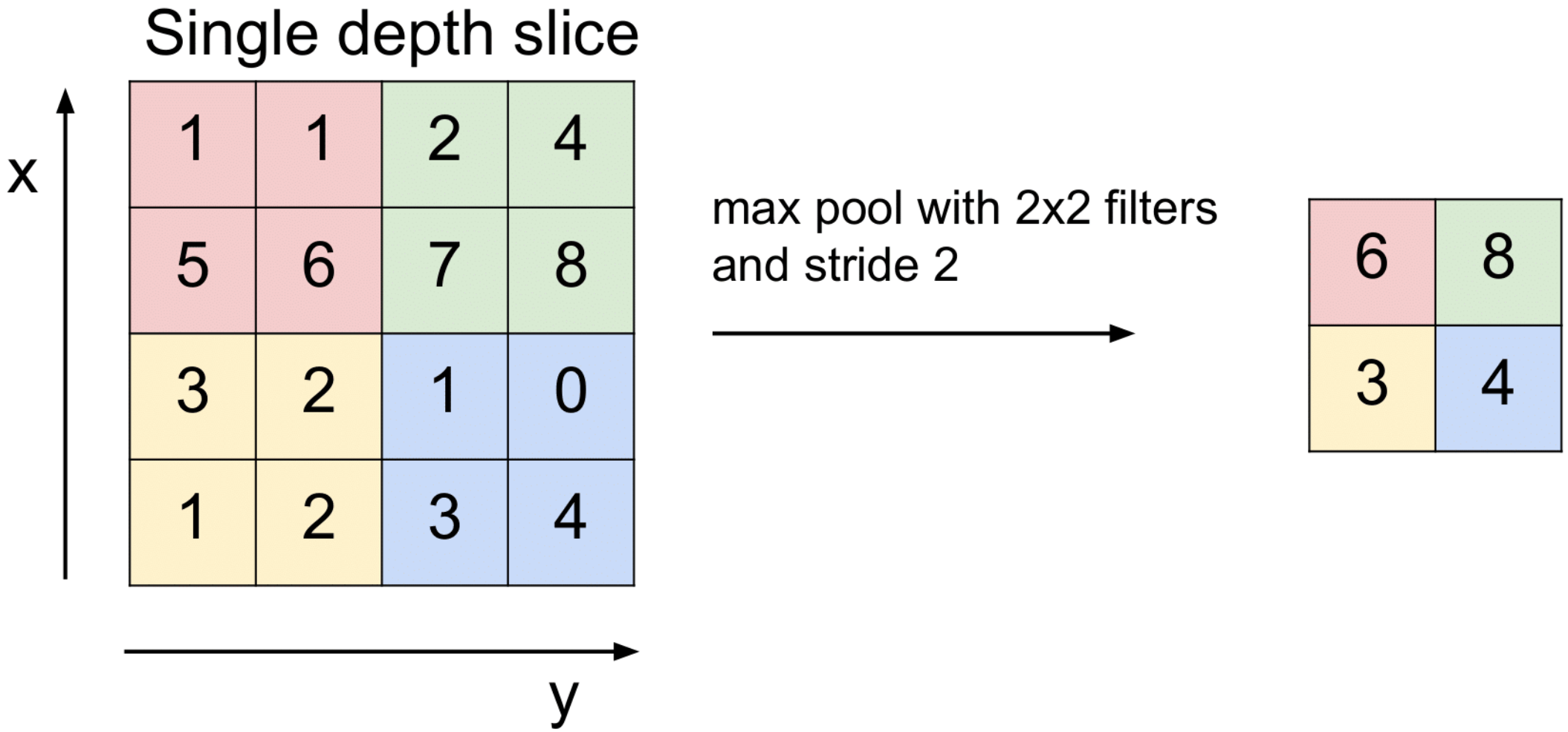
Pooling layer

- makes the representations smaller and more manageable
- operates over each activation map independently:



Source: Stanford CS231n Lecture 5 2017 by Fei-Fei Li & Justin Johnson & Serena Yeung

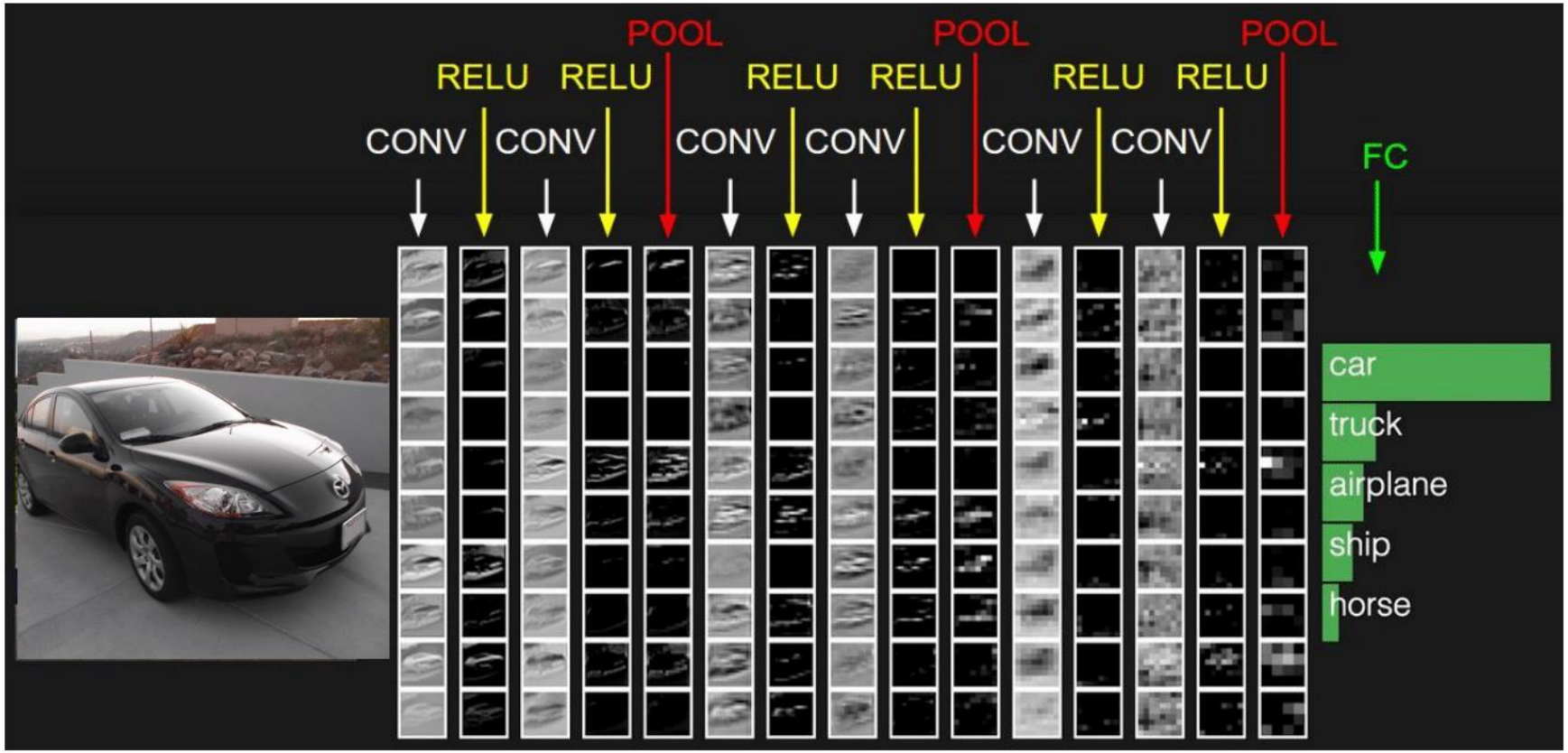
MAX POOLING



Source: Stanford CS231n Lecture 5 2017 by Fei-Fei Li & Justin Johnson & Serena Yeung

Fully Connected Layer (FC layer)

- Contains neurons that connect to the entire input volume, as in ordinary Neural Networks



Source: Stanford CS231n Lecture 5 2017 by Fei-Fei Li & Justin Johnson & Serena Yeung

Understanding ConvNets

- Visualize patches that maximally activate neurons
- Visualize the weights
- Visualize the representation space (e.g. with t-SNE)
- Occlusion experiments
- Human experiment comparisons
- Deconv approaches (single backward pass)
- Optimization over image approaches (optimization)

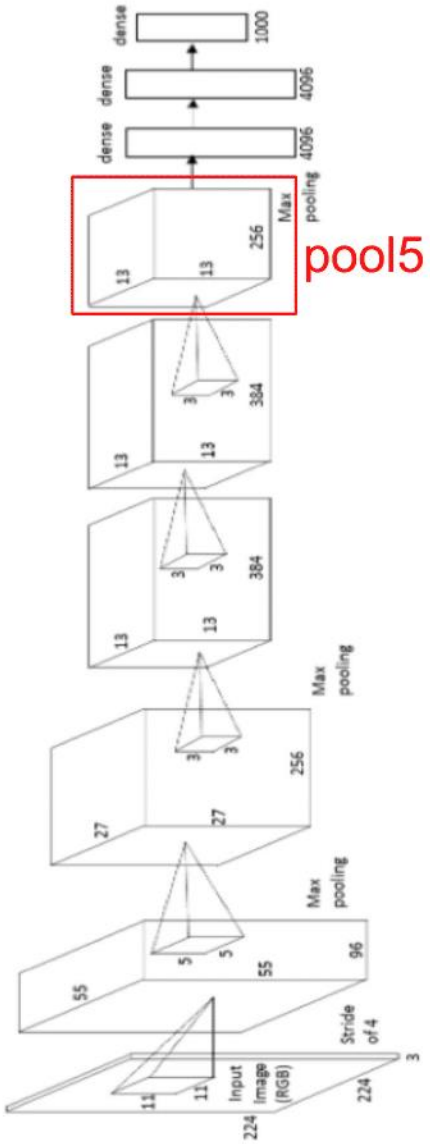
Source: Stanford CS231n Lecture 9 2016 by Fei-Fei Li & Andrej Karpathy & Justin Johnson

Visualize patches that maximally activate neurons

Figure 4: Top regions for six pool$_5$ units. Receptive fields and activation values are drawn in white. Some units are aligned to concepts, such as people (row 1) or text (4). Other units capture texture and material properties, such as dot arrays (2) and specular reflections (6).

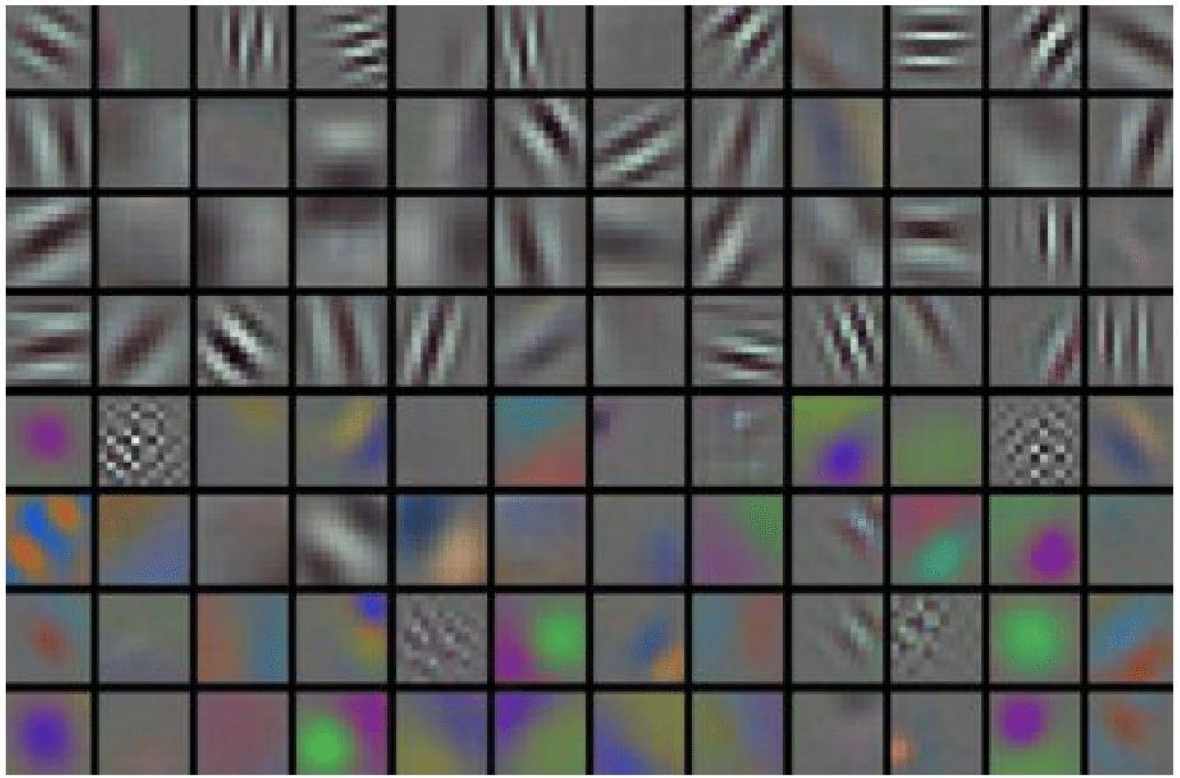
Rich feature hierarchies for accurate object detection and semantic segmentation
[Girshick, Donahue, Darrell, Malik]

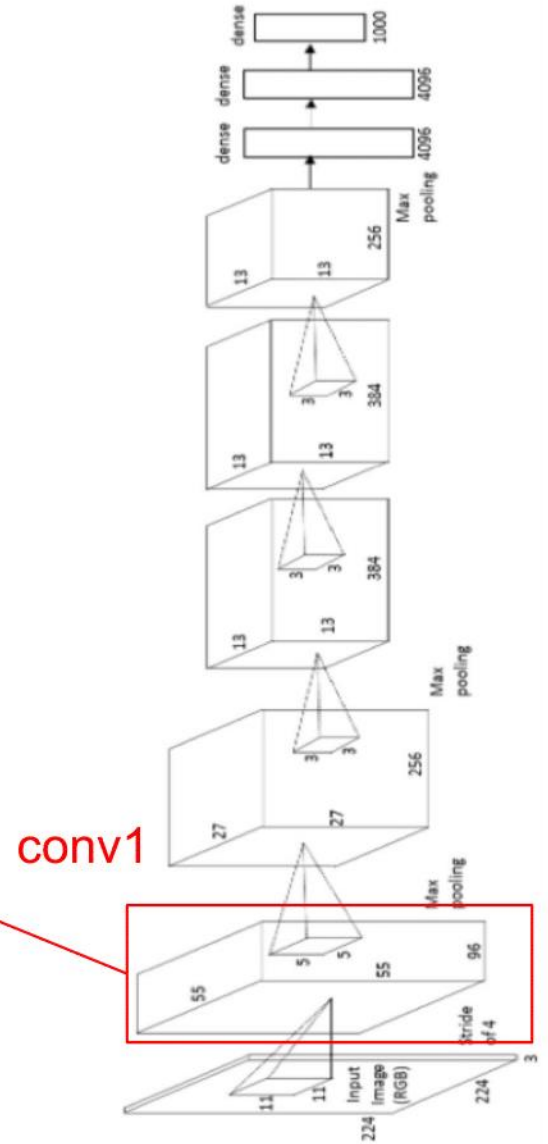
one-stream AlexNet



Source: Stanford CS231n Lecture 9 2016 by Fei-Fei Li & Andrej Karpathy & Justin Johnson

Visualize the filters/kernels (raw weights)





only interpretable on the first layer :(

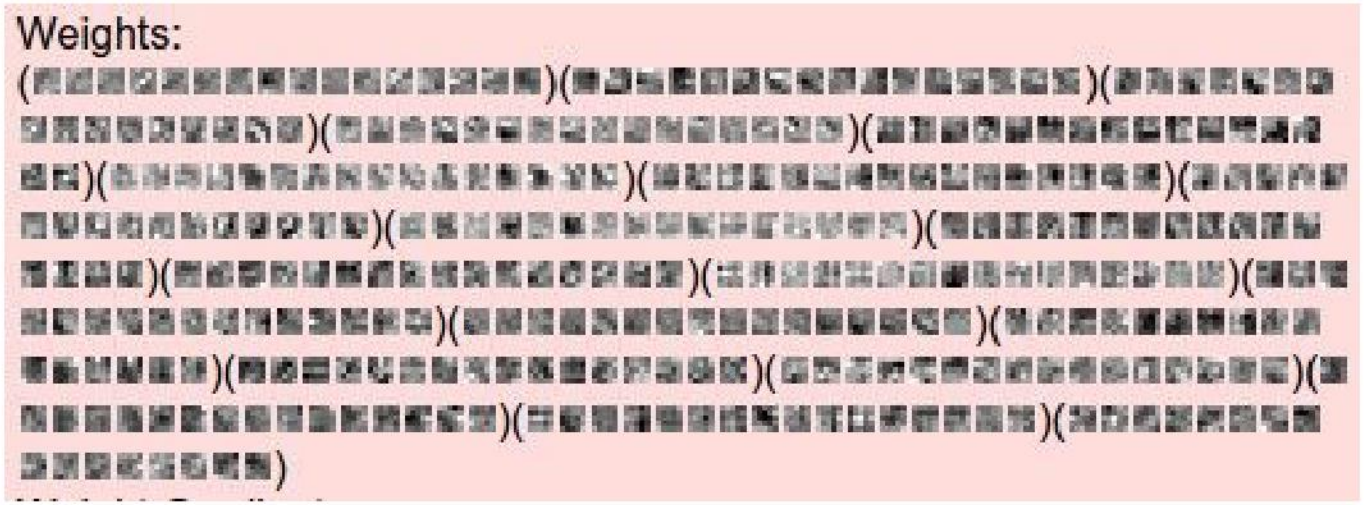
Source: Stanford CS231n Lecture 9 2016 by Fei-Fei Li & Andrej Karpathy & Justin Johnson

Visualize the filters/kernels (raw weights)

you can still do it for higher layers, it's just not that interesting

(these are taken from ConvNetJS CIFAR-10 demo)

layer 1 weights

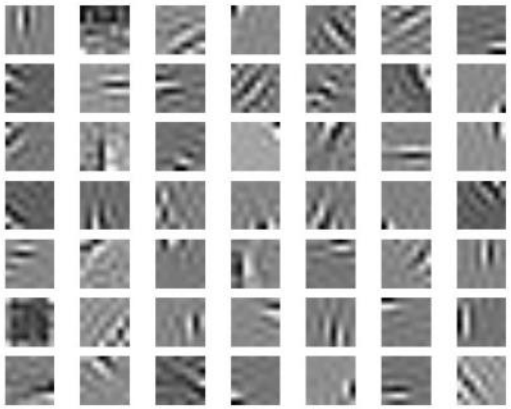


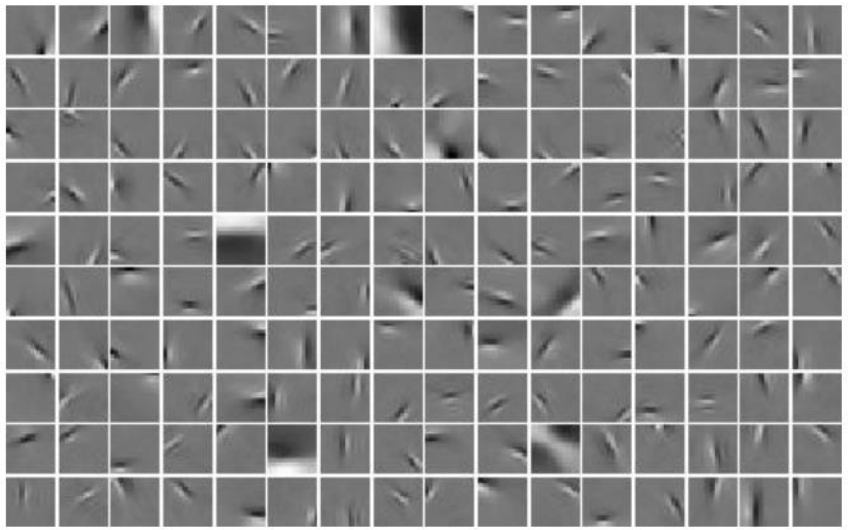
layer 2 weights

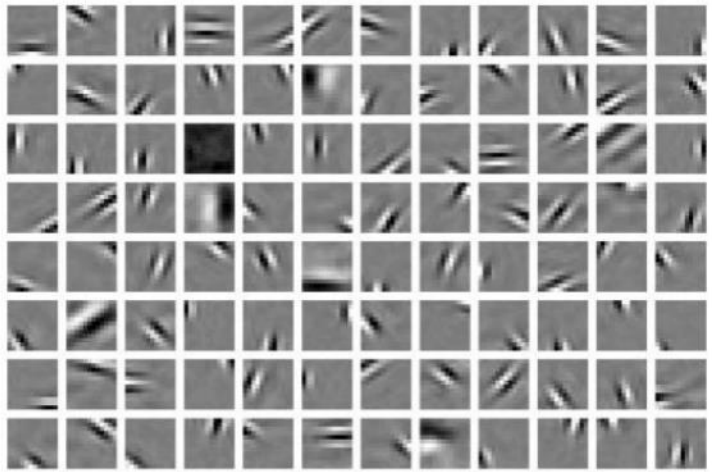
layer 3 weights

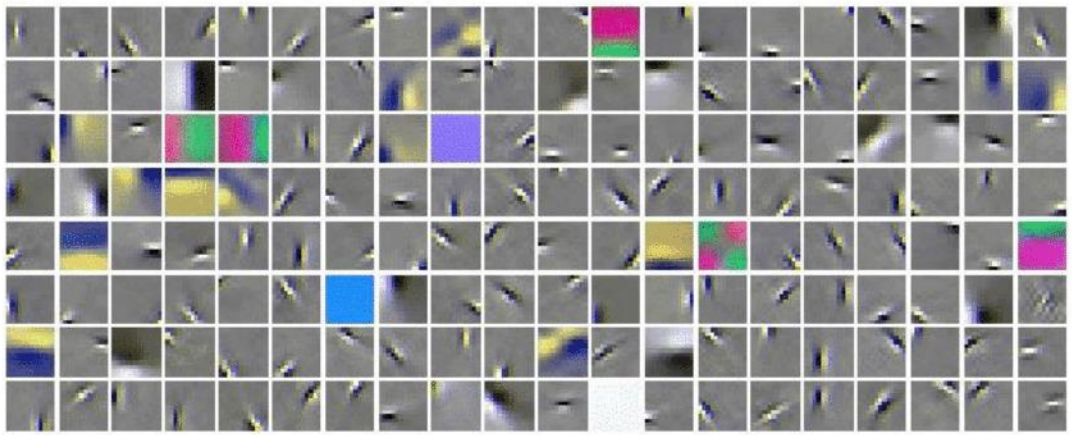
Source: Stanford CS231n Lecture 9 2016 by Fei-Fei Li & Andrej Karpathy & Justin Johnson

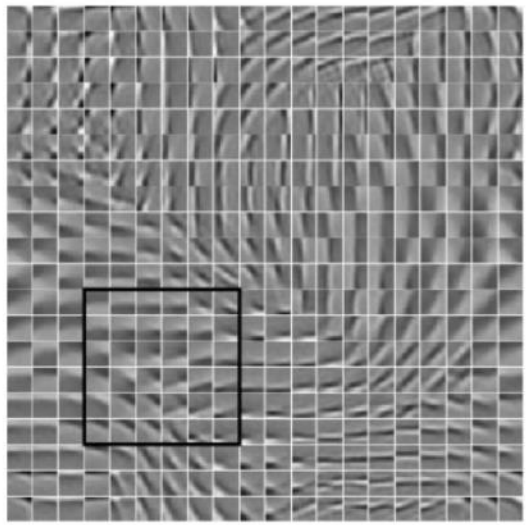
The gabor-like filters fatigue

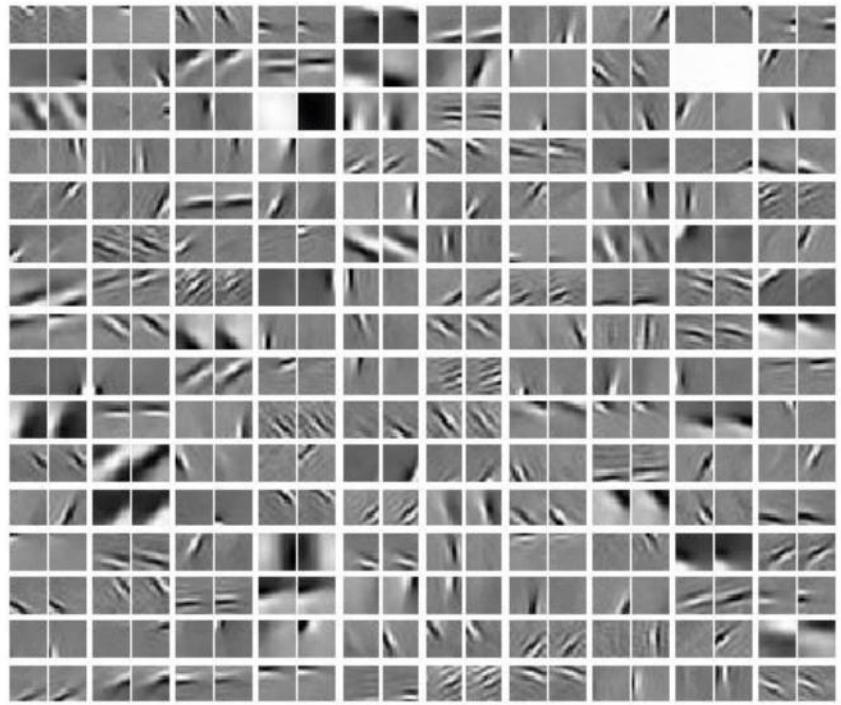






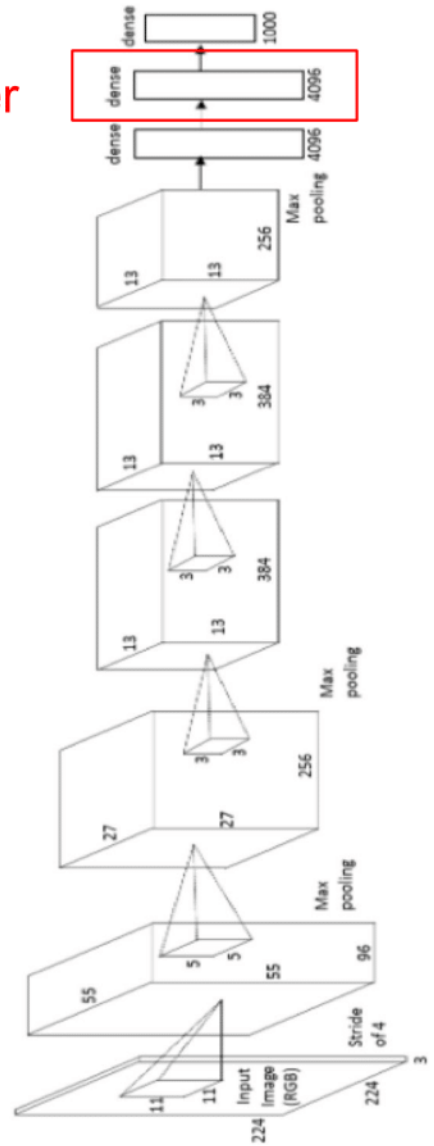






Source: Stanford CS231n Lecture 9 2016 by Fei-Fei Li & Andrej Karpathy & Justin Johnson

Visualizing the representation



4096-dimensional “code” for an image
(layer immediately before the classifier)

can collect the code for many images

Source: Stanford CS231n Lecture 9 2016 by Fei-Fei Li & Andrej Karpathy & Justin Johnson

Visualizing the representation

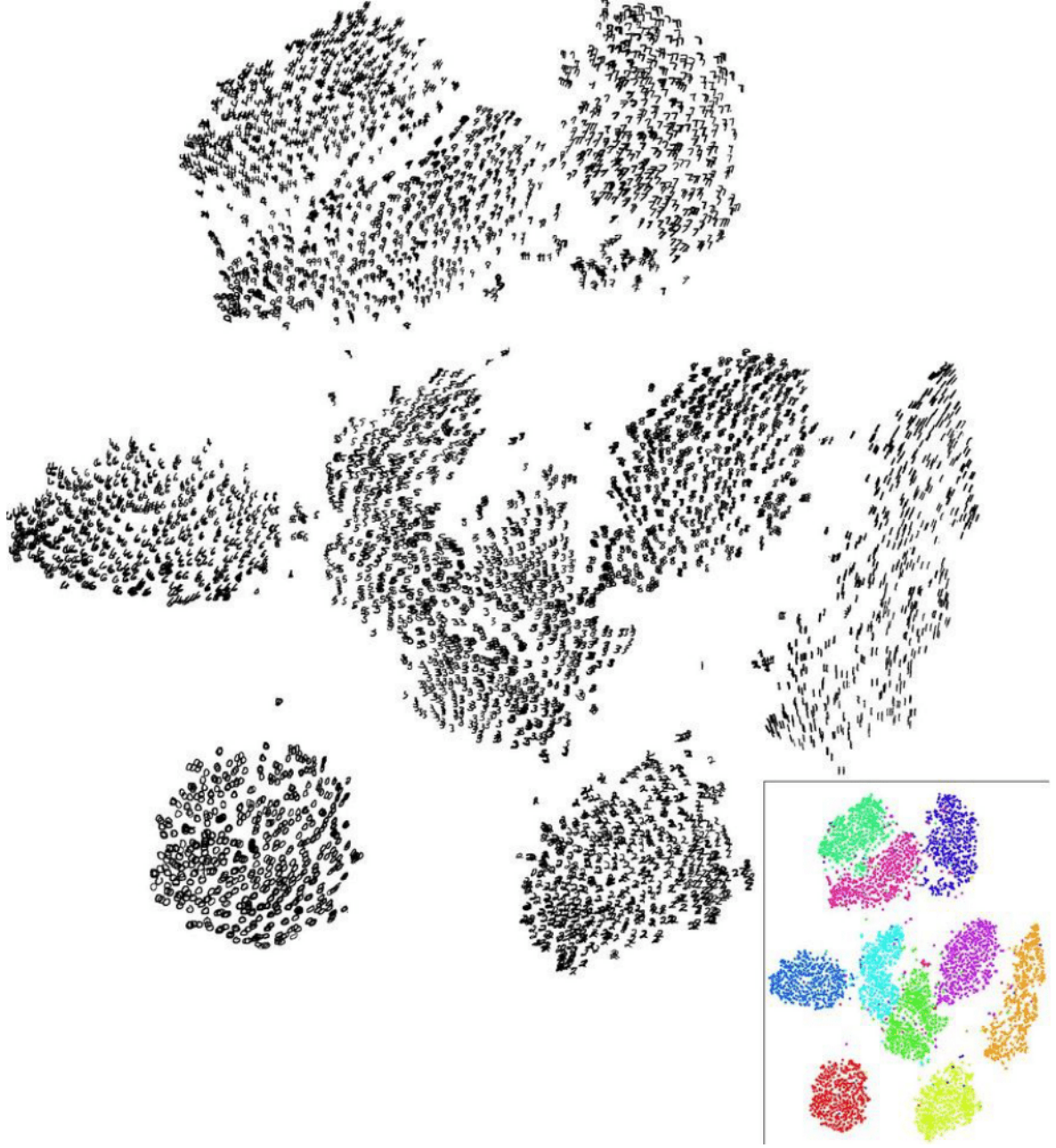
t-SNE visualization

[van der Maaten & Hinton]

Embed high-dimensional points so that locally, pairwise distances are conserved

i.e. similar things end up in similar places. dissimilar things end up wherever

Right: Example embedding of MNIST digits (0-9) in 2D



Source: Stanford CS231n Lecture 9 2016 by Fei-Fei Li & Andrej Karpathy & Justin Johnson

Occlusion experiments

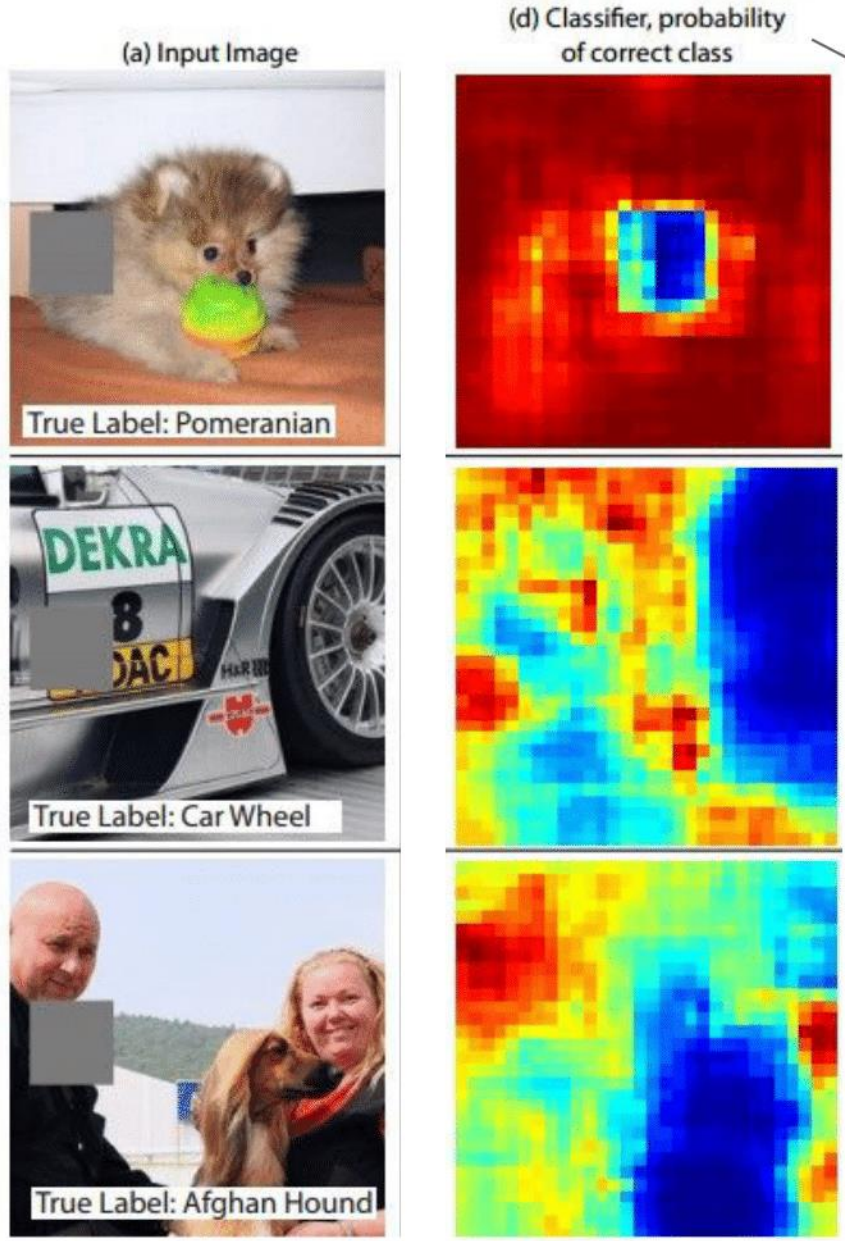
[Zeiler & Fergus 2013]

(as a function of the position of the square of zeros in the original image)

Source: Stanford CS231n Lecture 9 2016 by Fei-Fei Li & Andrej Karpathy & Justin Johnson

Occlusion experiments

[Zeiler & Fergus 2013]

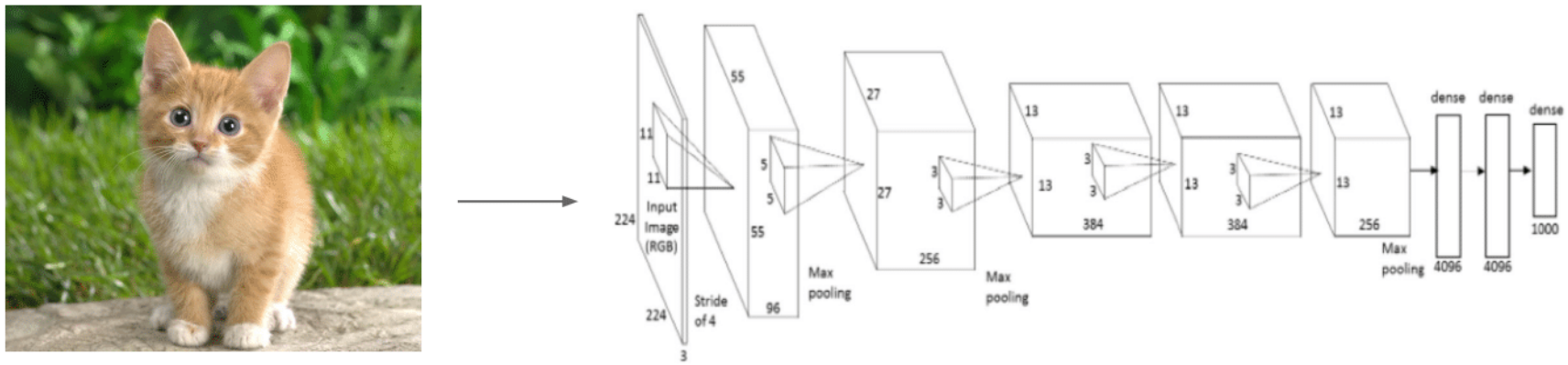


(as a function of the position of the square of zeros in the original image)

Source: Stanford CS231n Lecture 9 2016 by Fei-Fei Li & Andrej Karpathy & Justin Johnson

Deconv approaches

1. Feed image into net

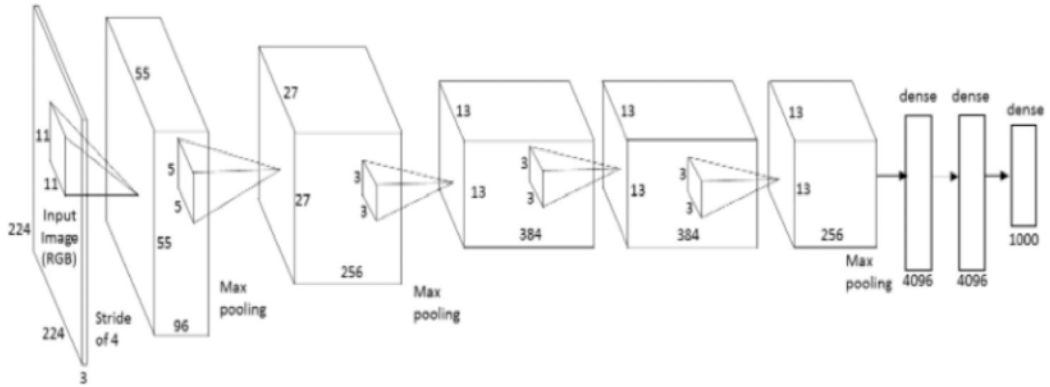


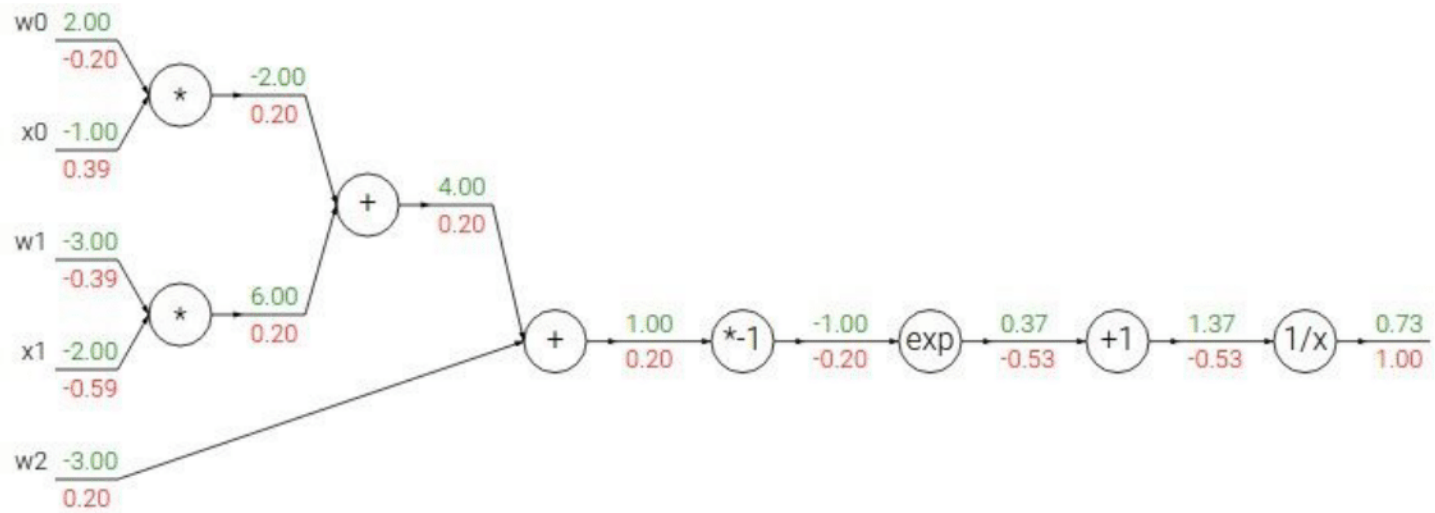
Q: how can we compute the gradient of any arbitrary neuron in the network w.r.t. the image?

Source: Stanford CS231n Lecture 9 2016 by Fei-Fei Li & Andrej Karpathy & Justin Johnson

Deconv approaches

1. Feed image into net

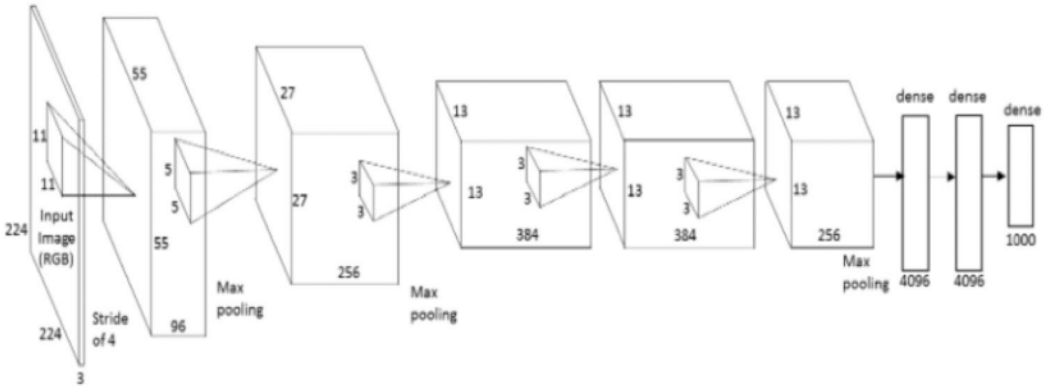




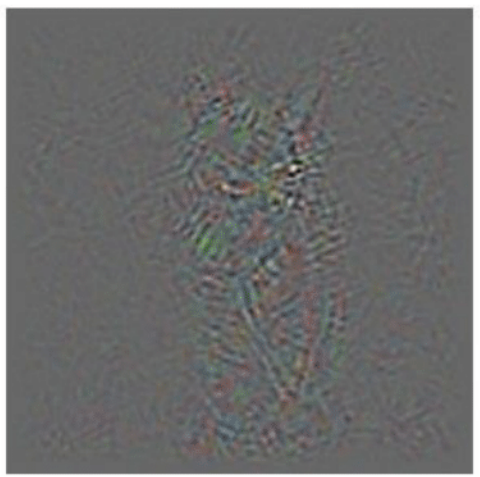
Source: Stanford CS231n Lecture 9 2016 by Fei-Fei Li & Andrej Karpathy & Justin Johnson

Deconv approaches

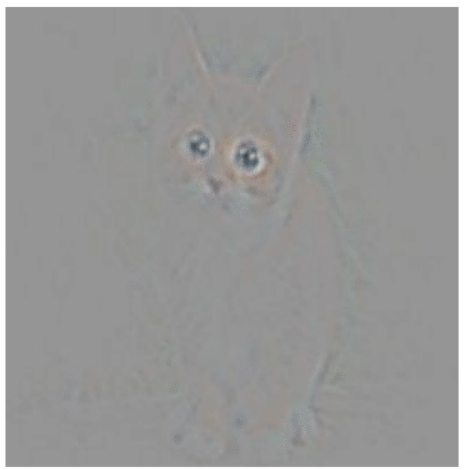
1. Feed image into net



2. Pick a layer, set the gradient there to be all zero except for one 1 for some neuron of interest
3. Backprop to image:



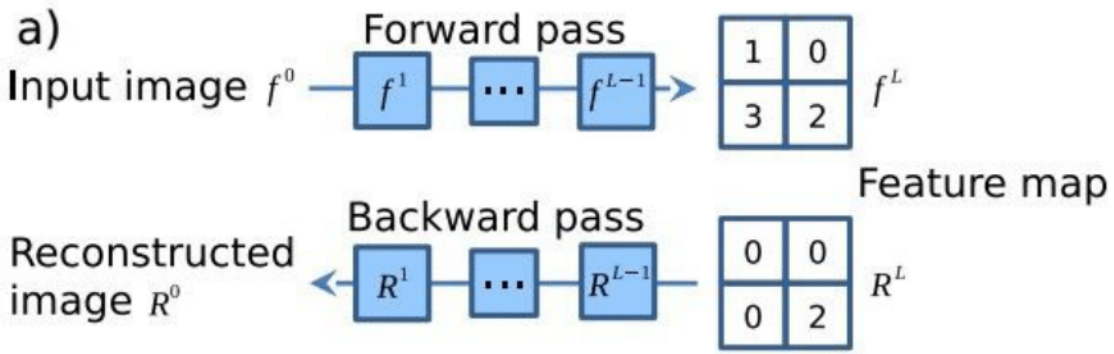
"Guided backpropagation:" instead



Source: Stanford CS231n Lecture 9 2016 by Fei-Fei Li & Andrej Karpathy & Justin Johnson

Deconv approaches

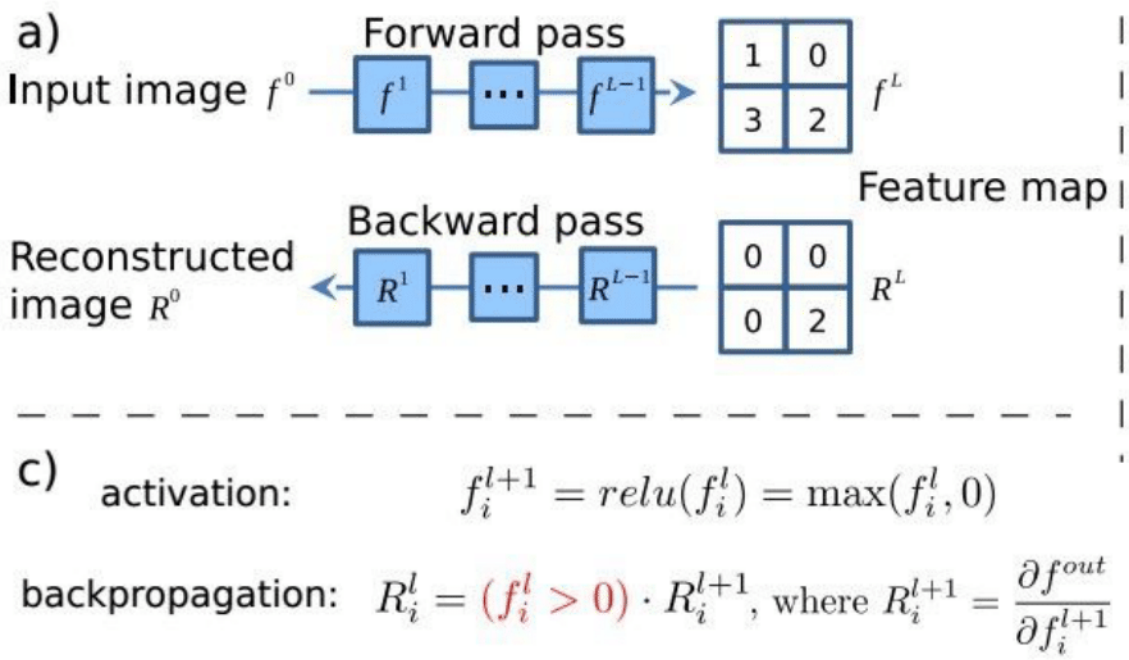
[Visualizing and Understanding Convolutional Networks, Zeiler and Fergus 2013]
[Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps, Simonyan et al., 2014]
[Striving for Simplicity: The all convolutional net, Springenberg, Dosovitskiy, et al., 2015]

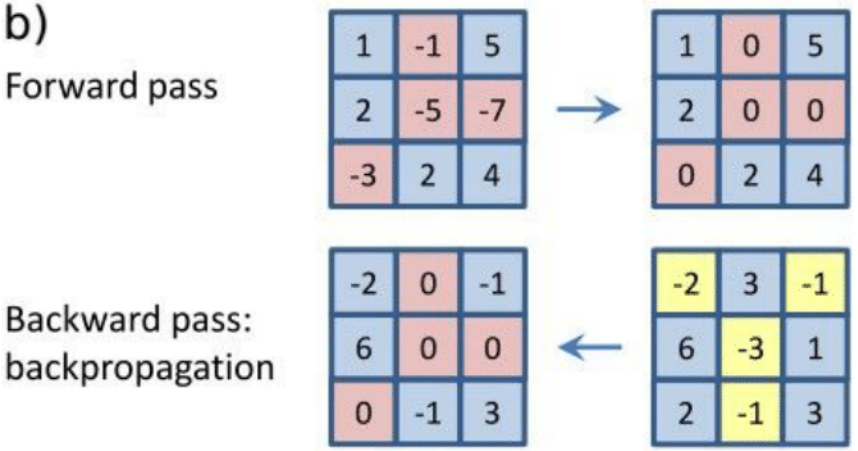


Source: Stanford CS231n Lecture 9 2016 by Fei-Fei Li & Andrej Karpathy & Justin Johnson

Deconv approaches

[Visualizing and Understanding Convolutional Networks, Zeiler and Fergus 2013]
[Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps, Simonyan et al., 2014]
[Striving for Simplicity: The all convolutional net, Springenberg, Dosovitskiy, et al., 2015]



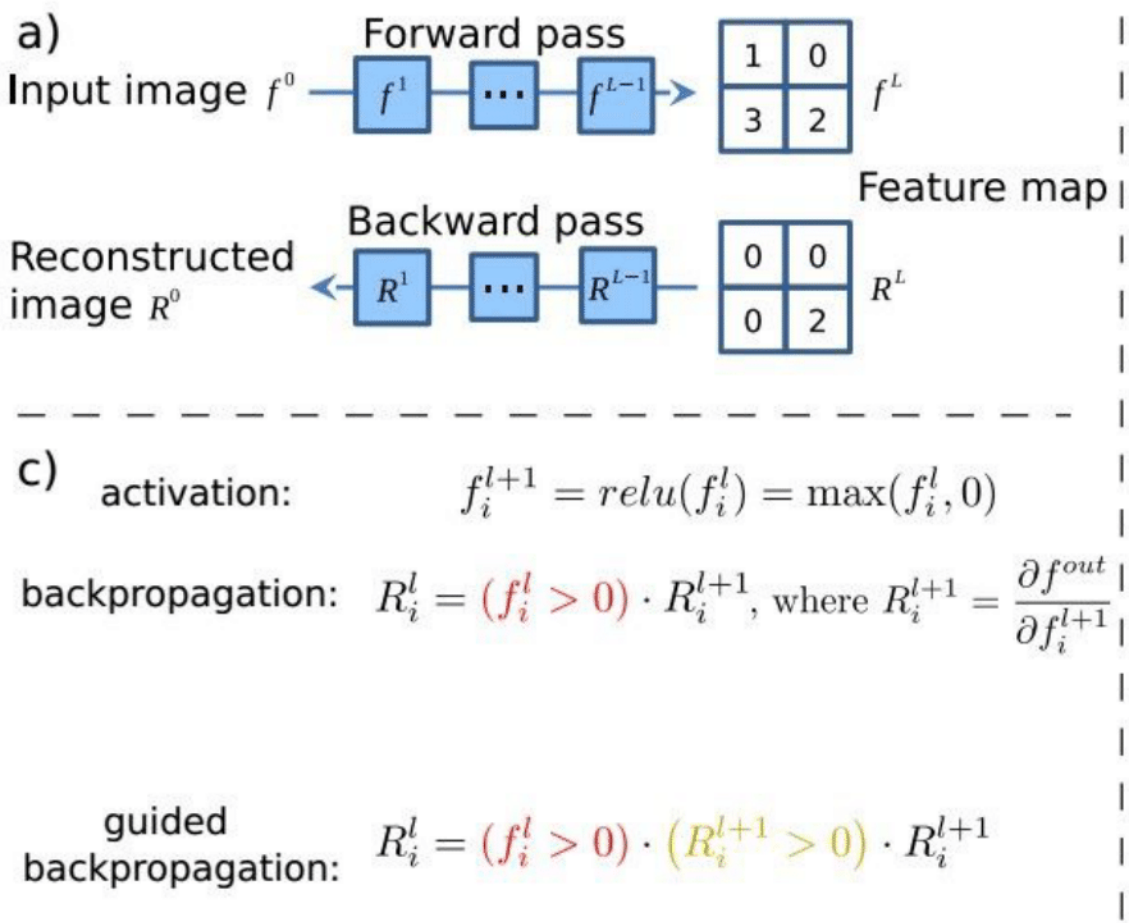


Backward pass for a ReLU (will be changed in Guided Backprop)

Source: Stanford CS231n Lecture 9 2016 by Fei-Fei Li & Andrej Karpathy & Justin Johnson

Deconv approaches

[Visualizing and Understanding Convolutional Networks, Zeiler and Fergus 2013]
[Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps, Simonyan et al., 2014]
[Striving for Simplicity: The all convolutional net, Springenberg, Dosovitskiy, et al., 2015]



Source: Stanford CS231n Lecture 9 2016 by Fei-Fei Li & Andrej Karpathy & Justin Johnson

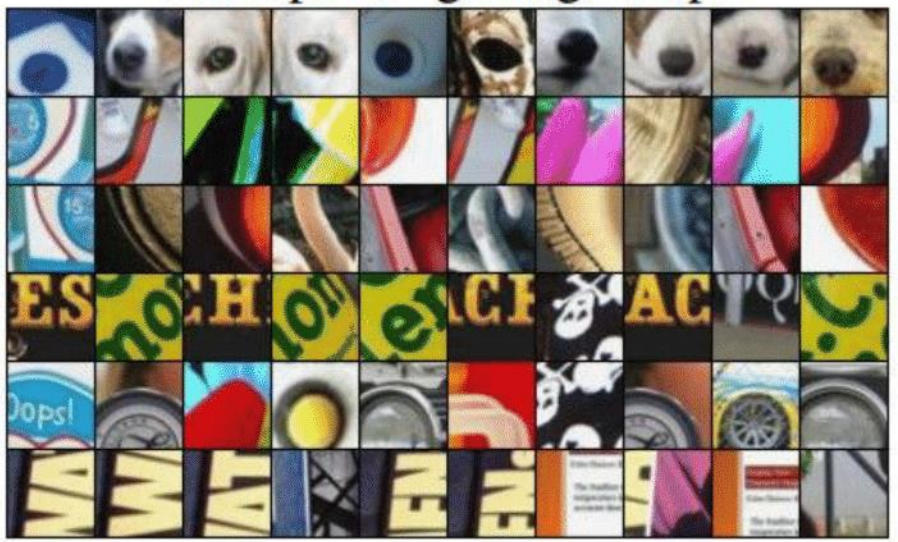
Visualization of patterns learned by the layer **conv6** (top) and layer **conv9** (bottom) of the network trained on ImageNet.

Each row corresponds to one filter.

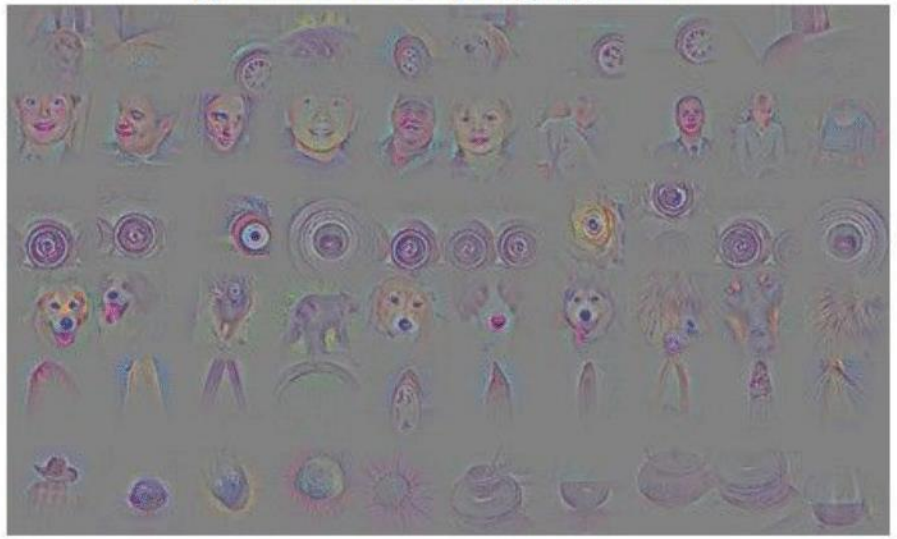
The visualization using "guided backpropagation" is based on the top 10 image patches activating this filter taken from the ImageNet dataset.

guided backpropagation

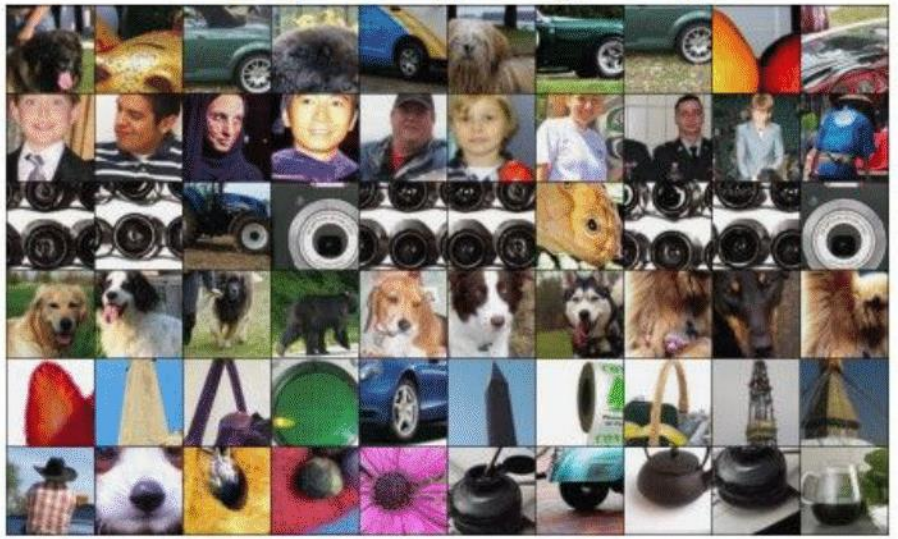
corresponding image crops



guided backpropagation



corresponding image crops

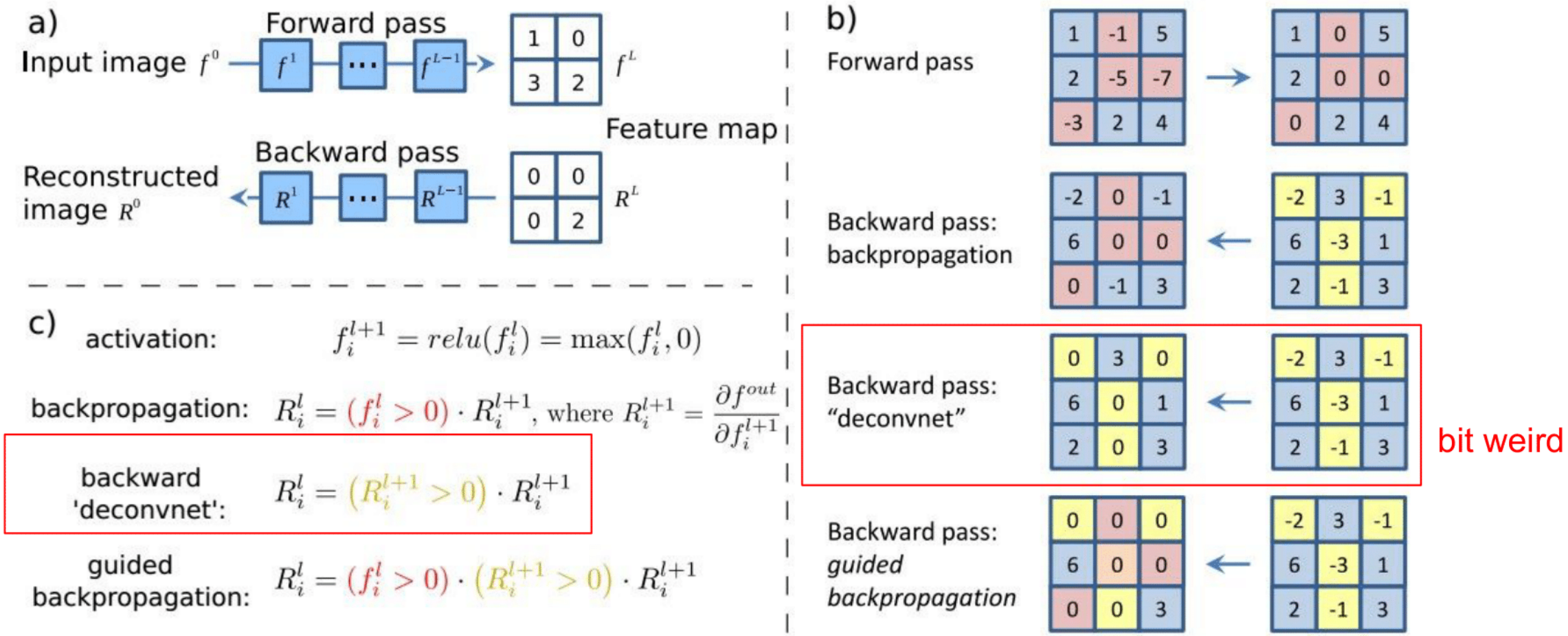


[Striving for Simplicity: The all convolutional net, Springenberg, Dosovitskiy, et al., 2015]

Source: Stanford CS231n Lecture 9 2016 by Fei-Fei Li & Andrej Karpathy & Justin Johnson

Deconv approaches

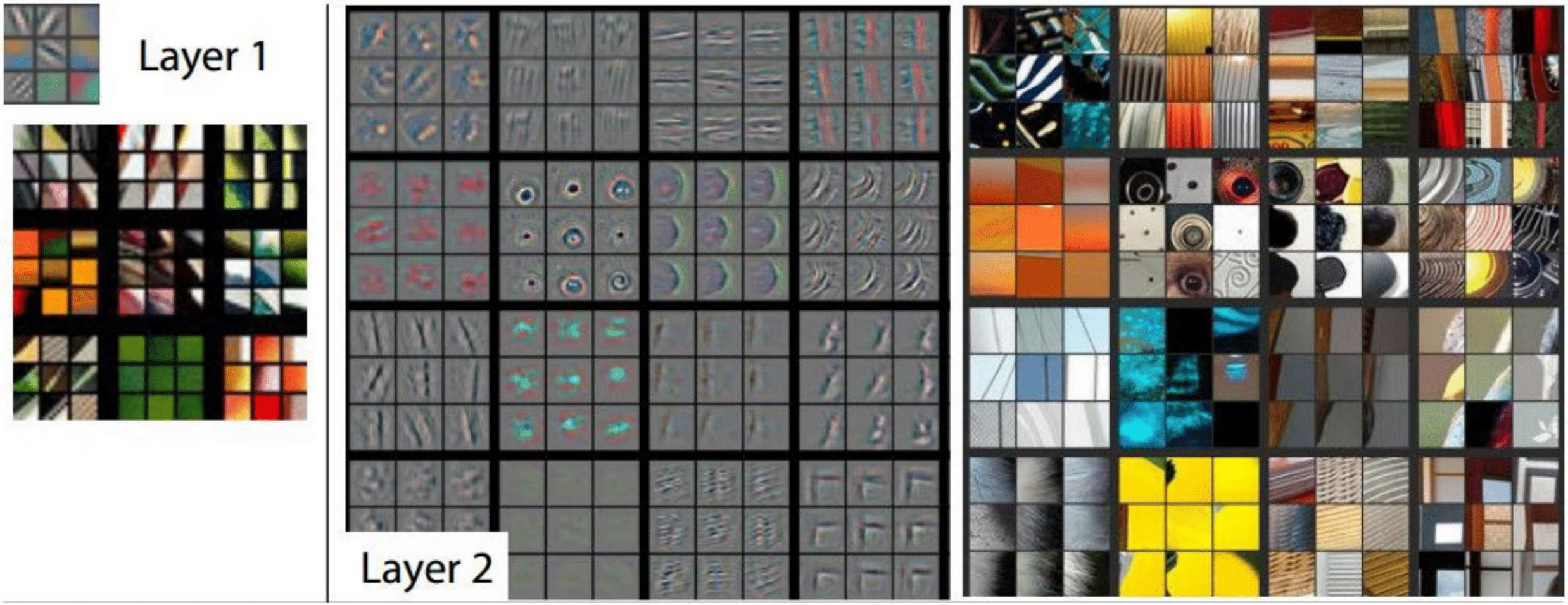
[Visualizing and Understanding Convolutional Networks, Zeiler and Fergus 2013]
[Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps, Simonyan et al., 2014]
[Striving for Simplicity: The all convolutional net, Springenberg, Dosovitskiy, et al., 2015]



Source: Stanford CS231n Lecture 9 2016 by Fei-Fei Li & Andrej Karpathy & Justin Johnson

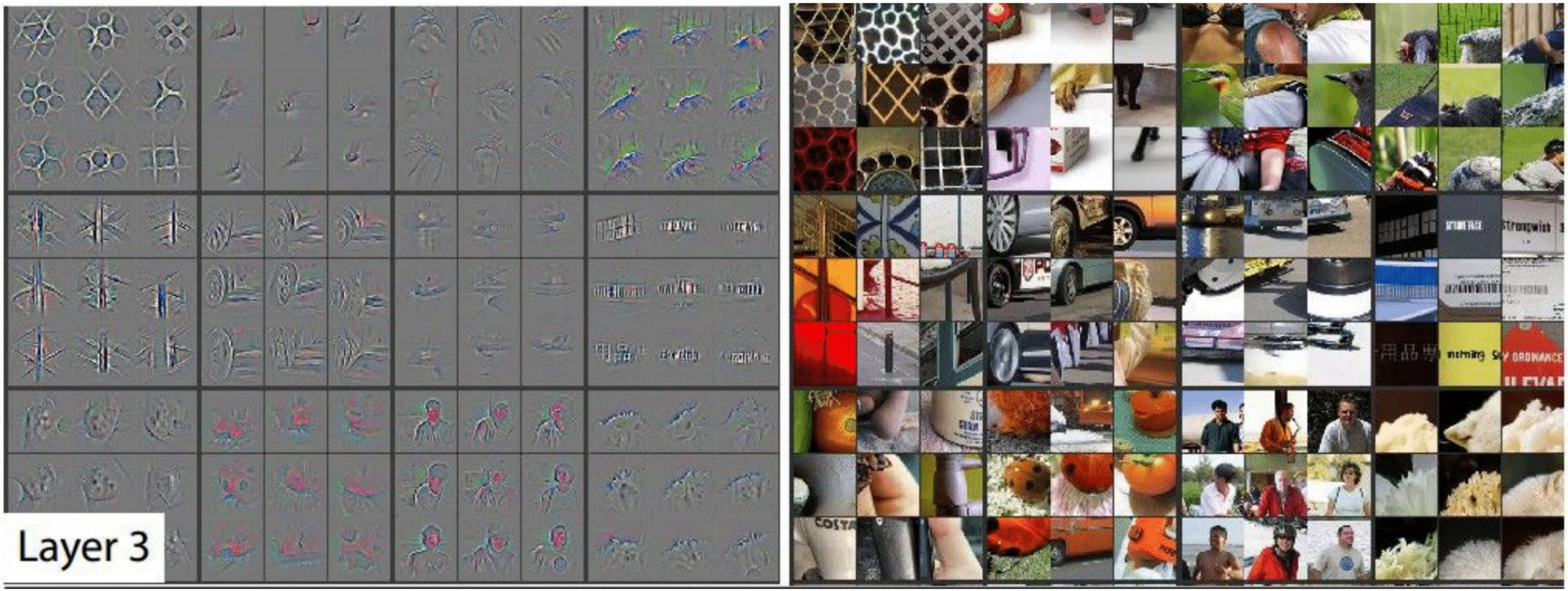
Visualizing and Understanding Convolutional Networks
Zeiler & Fergus, 2013

Visualizing arbitrary neurons along the way to the top...



Source: Stanford CS231n Lecture 9 2016 by Fei-Fei Li & Andrej Karpathy & Justin Johnson

Visualizing arbitrary neurons along the way to the top...



Source: Stanford CS231n Lecture 9 2016 by Fei-Fei Li & Andrej Karpathy & Justin Johnson

Заключение

1. Основные направления CV
2. Примеры решаемых задач
3. CV Pipeline
4. Заключение
5. Q&A