Artificial Neural Networks: Lecture 8

Convolutional Neural Networks

Johanni Brea

EPFL, Lausanne, Switzerland

Objectives for today:

- Review: No free-lunch theorem and inductive biases
- Convolution Layers
- Max Pooling Layers
- Inductive bias of ConvNets
- ImageNet competition and modern ConvNets
- Training ConvNets with AutoDiff
- Applications beyond object recognition

Miniproject 1 is due on April 29 at 23:59
End of next week: handout of miniprojects 2 & 3

Reading for this lecture:

Goodfellow et al.,2016 *Deep Learning*

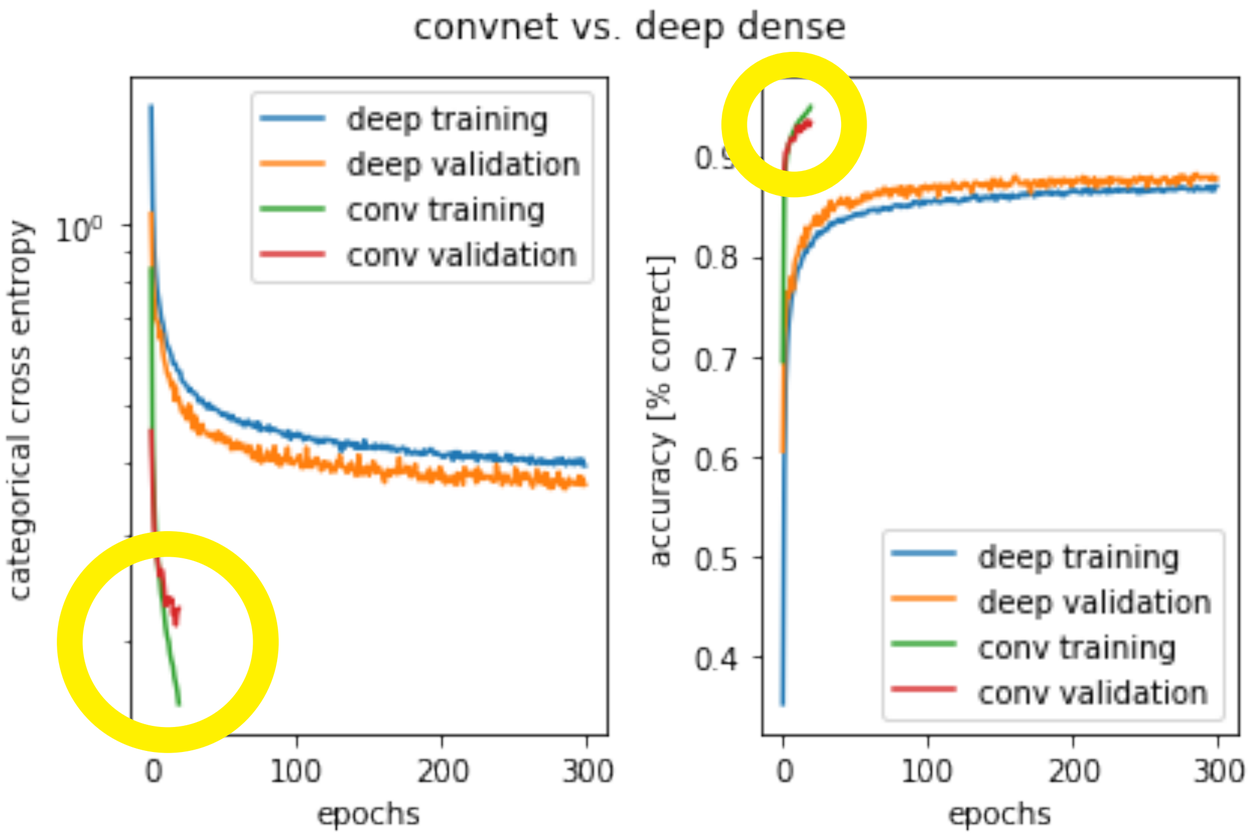
- Ch **9**

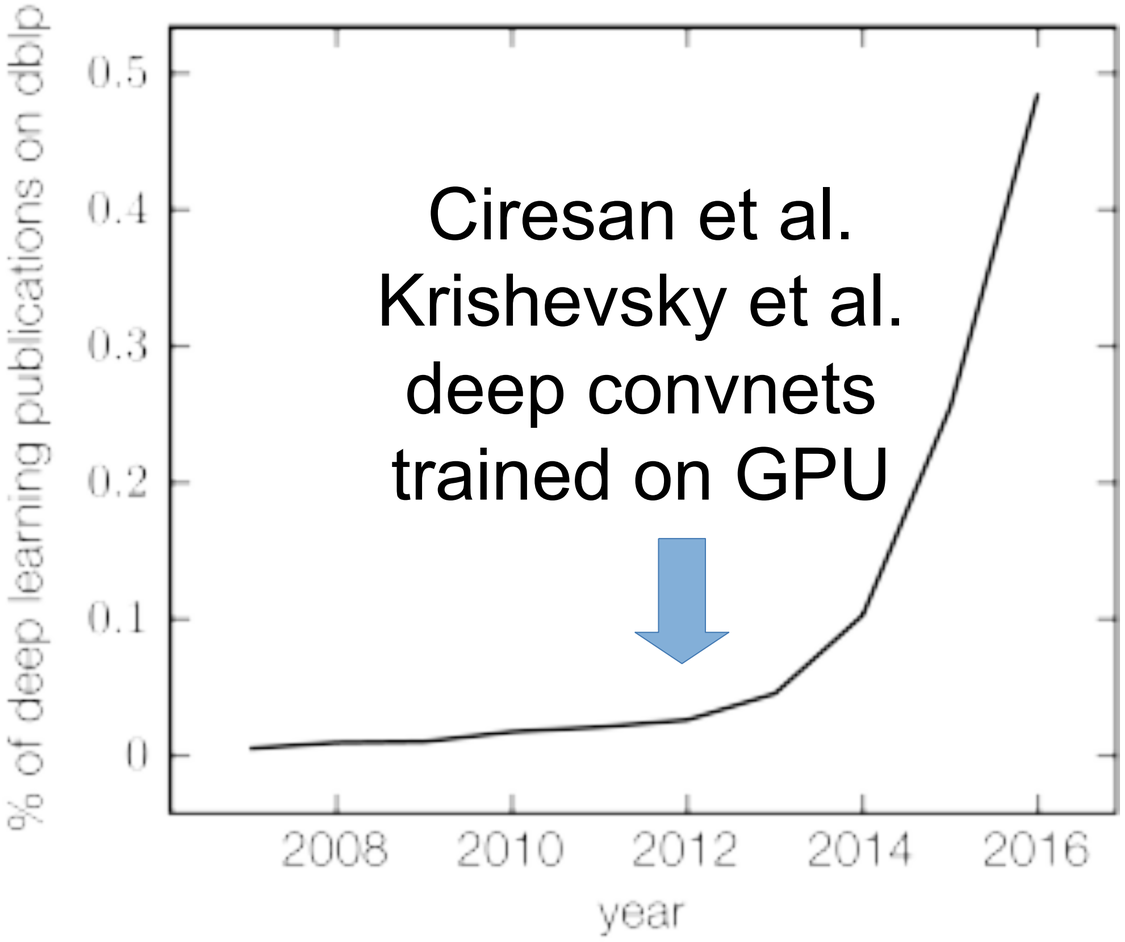
Further (optional) reading/watching for this lecture:

See references in the slides.
Lectures by Andrew Ng

Motivation

Miniproject





Previous slide.

Left: In the first miniproject we see that networks with convolutional layers reach significantly better training and test perfomances than those without.

Right: Deep learning and convolutional networks have been around for 2-3 decades. But the “deep learning revolution” started only around 2012, after the publication of several really deep convolutional networks that out-performed other approaches on e.g. difficult object recognition tasks. Afterwards the number of deep learning publications increased dramatically.

No free lunch theorem revisited

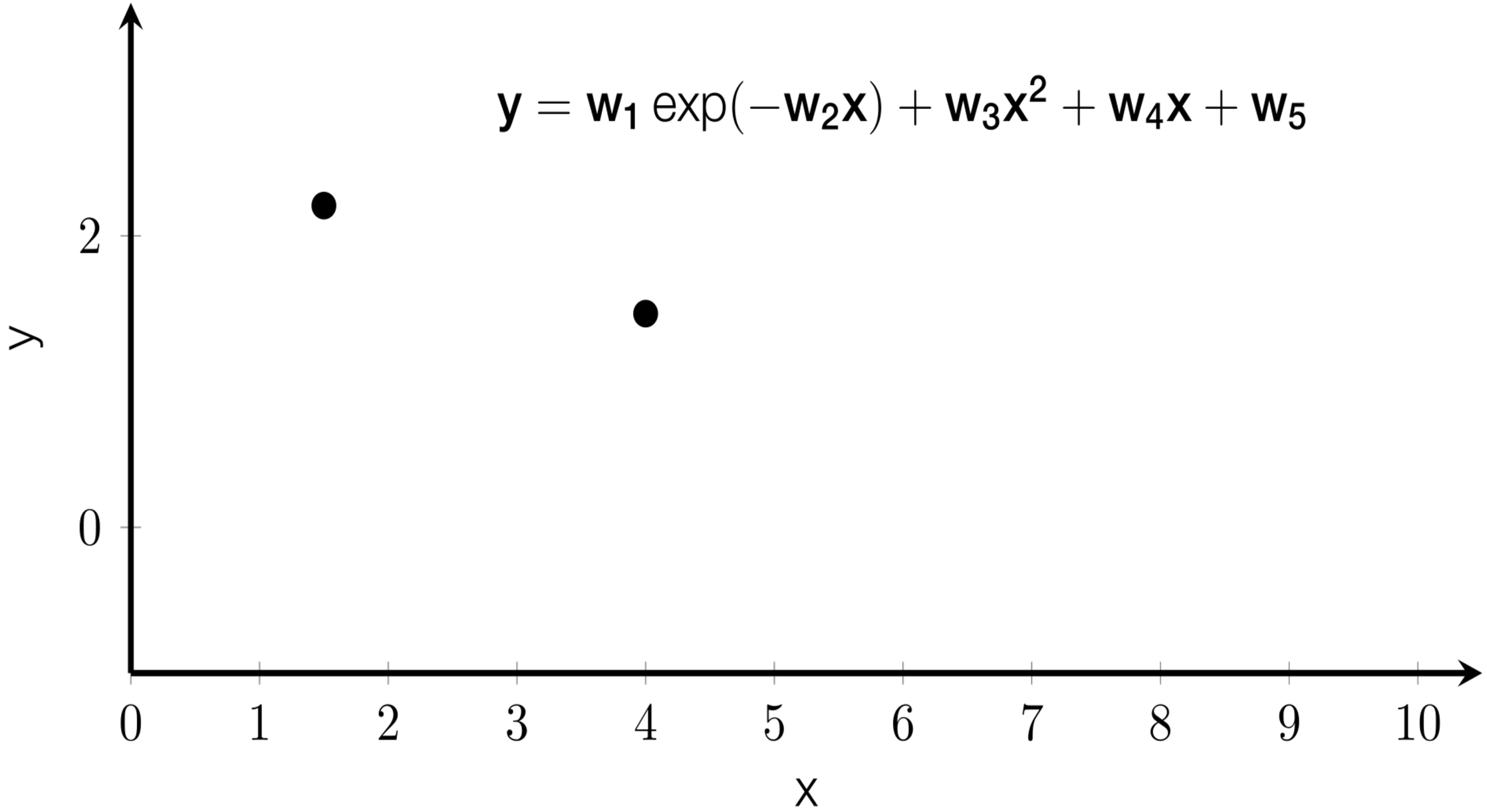
Explicit inductive biases, data augmentation and transfer learning

Previous slide.

Why are convolutional networks much better than other approaches on these tasks? Because they have the right “inductive bias”.

Before we actually start looking at convolutional networks, we will illustrate the term “inductive bias” with a toy example and look at the different ways to find good inductive biases.

review: No Free-lunch Theorem (weak inductive bias)

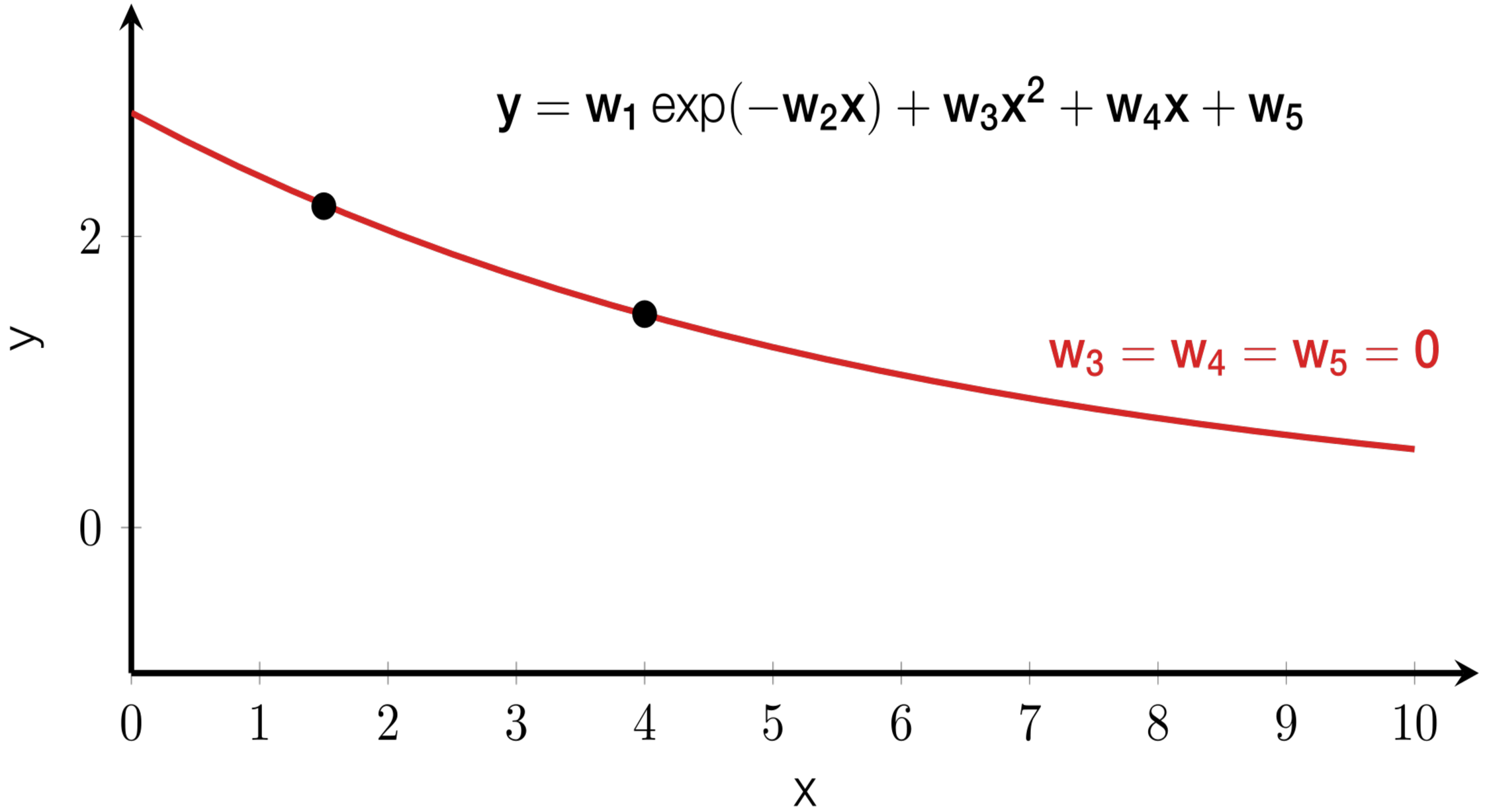


Previous slide.

Let us suppose we have two data points in our training set. It is a regression task with input x and real-valued target y; if it were a classification task, y would be discrete.

There are infinitely many possibilities to fit these two data points, but let us assume that our initial inductive bias is that x and y can be linked through the displayed formula.
The fitting problem still is under-constraint, since we have more parameters than data points; we still have have infinitely many possibilities to fit these two data points with the displayed formula.

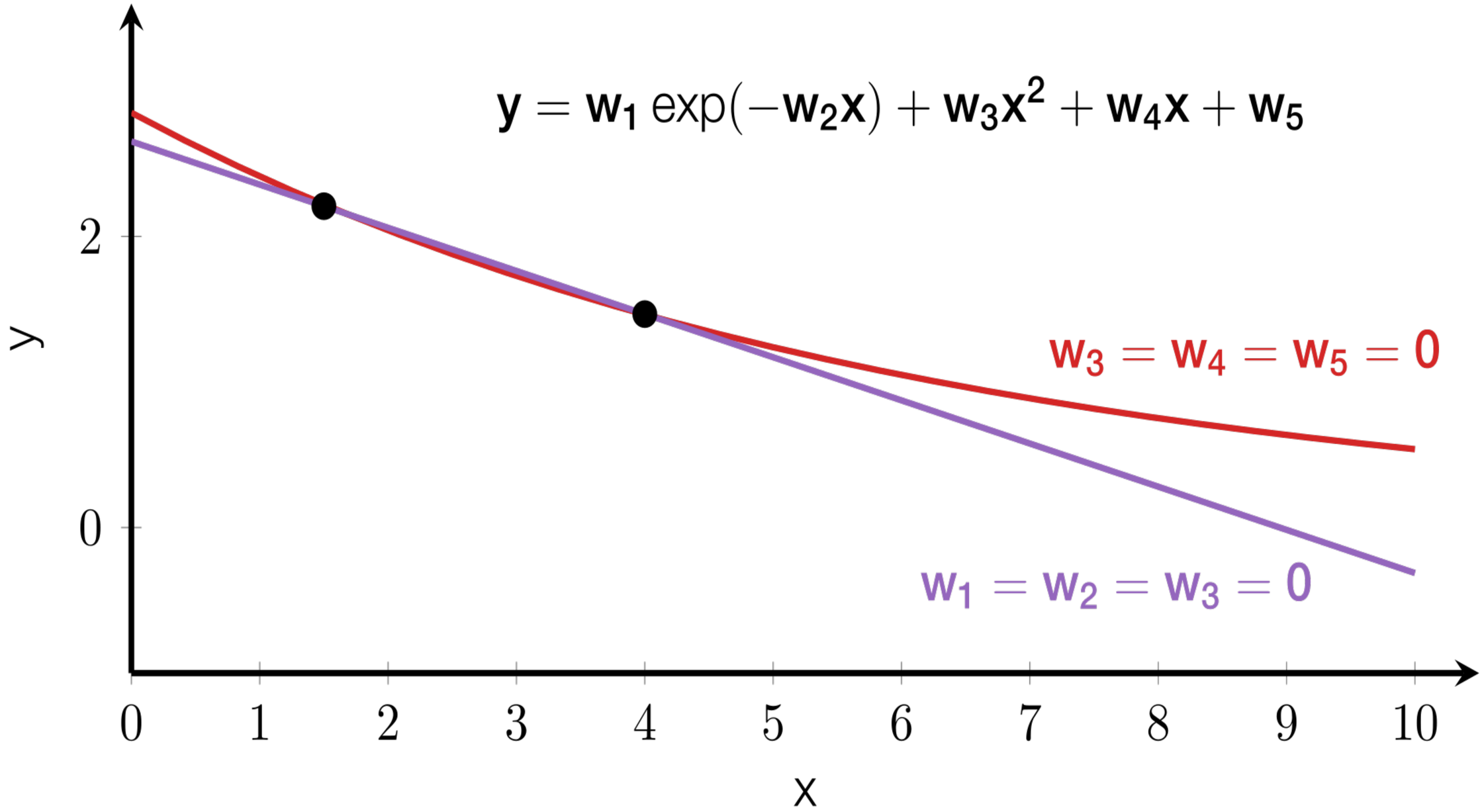
review: No Free-lunch Theorem (strong inductive bias 1)



Previous slide.

But let us assume we had reasons to consider even stronger inductive biases.
As a first example, exponential decay with $w_3 = w_4 = w_5 = 0$.

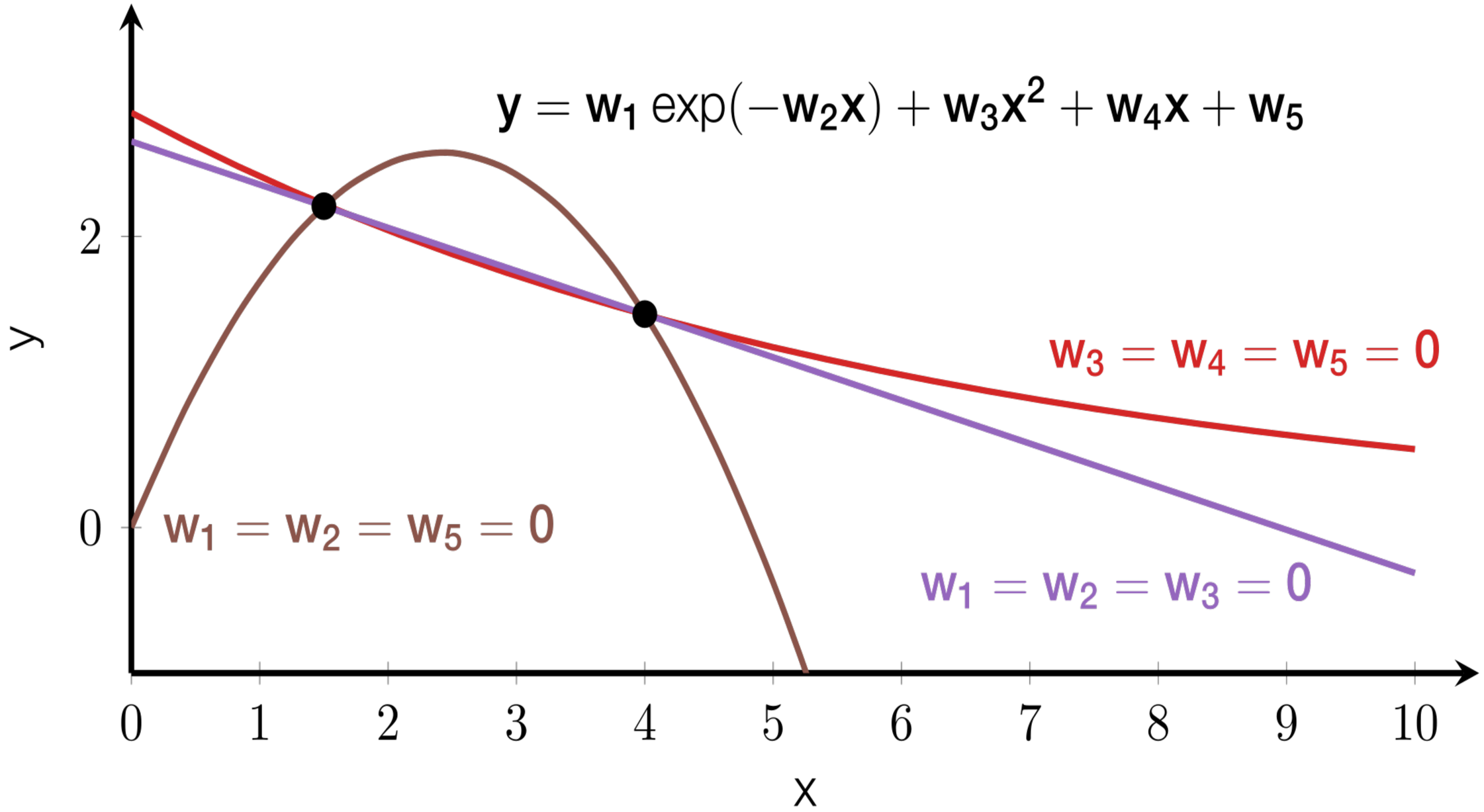
review: No Free-lunch Theorem (strong inductive bias 2)



Previous slide.

Second a straight line with $w_1 = w_2 = w_3 = 0$.

review: No Free-lunch Theorem (strong inductive bias 3)

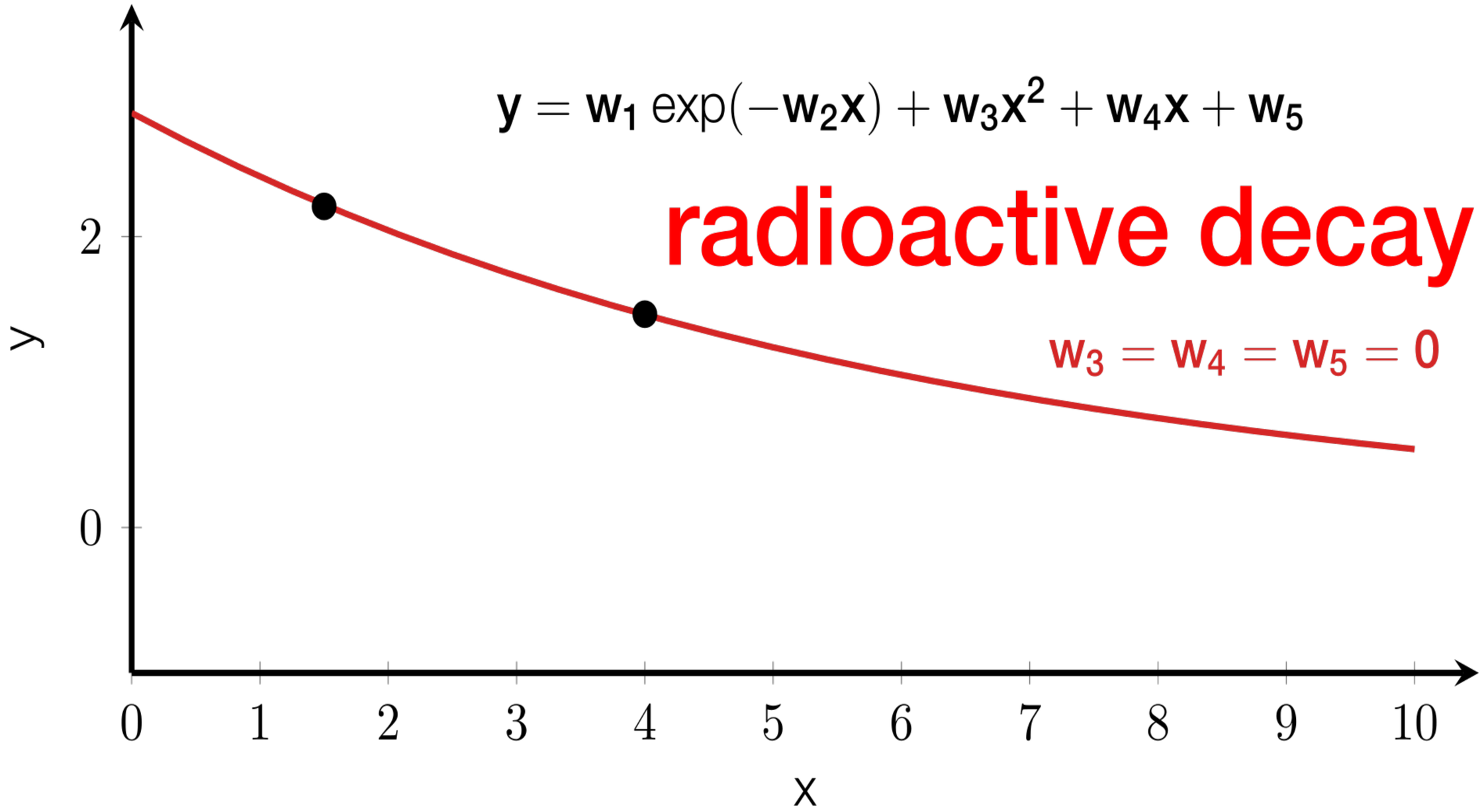


Previous slide.

And third, a parabola through the origin with $w_1 = w_2 = w_5 = 0$.
Any other way of fixing at least three parameters would be an valid alternative.

How should we choose between these inductive biases?

review: No Free-lunch Theorem (transfered inductive bias)



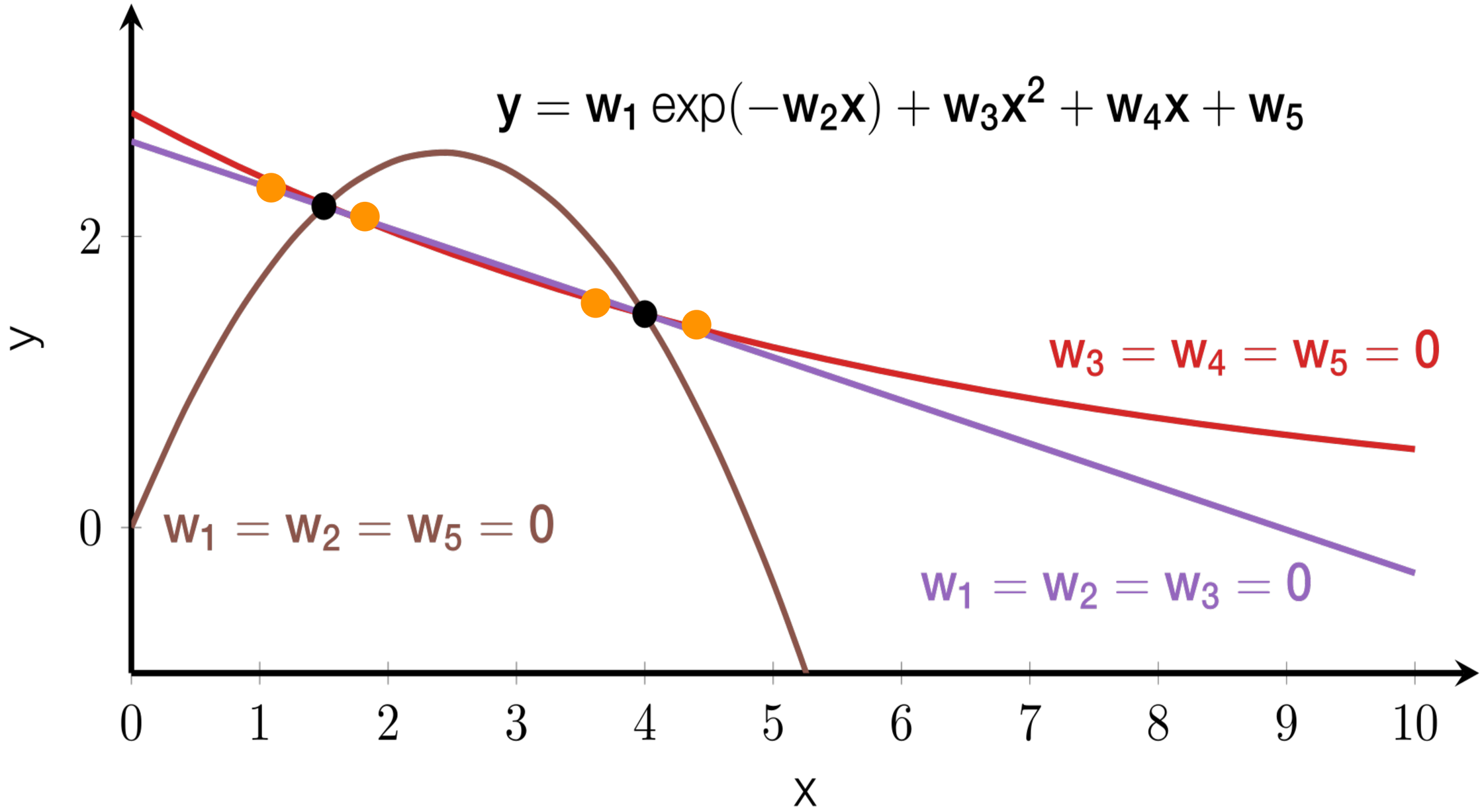
Previous slide.

If we have more information about the data, e.g. it comes from measurements of radioactive decay, we can transfer our knowledge about this type of problem, select a strong inductive bias and fit the other parameters.

This transfer can happen in different ways.

1. (The data scientist approach) We have already several times fitted all parameters w_1 to w_5 to similar data and we have always observed that w_3 to w_5 were close to 0.
2. (The physicist approach) There is some law that dictates a certain form of the function.

review: No Free-lunch Theorem (data augmentation)



Previous slide.

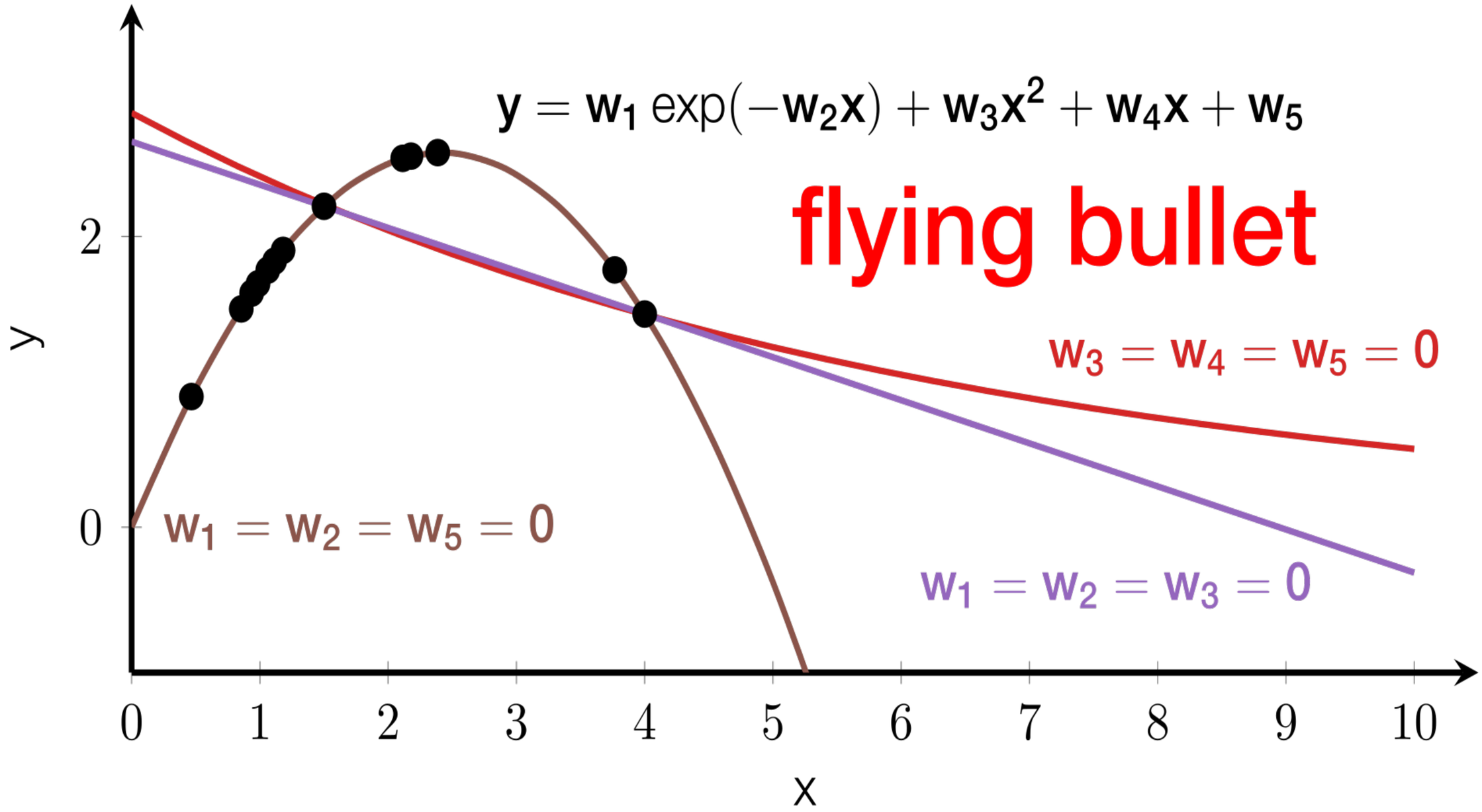
We may not know much about the data or find it difficult to define an explicit inductive bias but we have the intuition that the outcome should not change much if we transform the input in a certain way, e.g. if we would move the data points in the training set a bit to the left or to the right the outcome y should not be very different, as indicated with the orange data points.

With this augmented dataset we can fit all parameters w_1 to w_5.

Caveat: It may be difficult to find transformations that really leave the outcome invariant; getting more actual training data may be more worthwhile than data augmentation.

Thanks to data augmentation, or transfer from related problems we may be able to find a fit to only two data points that generalizes well if the data actually came from a measurement of radioactive decay.

review: No Free-lunch Theorem (the wrong inductive bias)



Previous slide.

Even though the training error is zero for all three strong inductive biases considered here and we have the same degrees of freedom in each case (namely two parameters to be fitted) the performance on the test set can be terrible, if we pick the wrong inductive bias for the data at hand. If the data came from the measurement of the trajectory of a flying bullet the fit with the exponential decay or the straight line would not generalize well.
In other words, we don’t get good generalization for free; there is no free lunch here. Only if we choose the inductive bias that matches the data we get good generalization.

Inductive bias

Induction = finding a rule (function) from specific examples
Inductive bias = prior preference of rules (functions)

1) Explicit inductive bias (transfer reasoning)

"For radioactive decay I know that $w_3 = w_4 = w_5 = 0$"

2) Inductive bias through transfer learning

3)"I train first different models on data from other radioactive elements and choose the best for my current case"

4) Inductive bias through data augmentation

5)"For radioactive decay neighboring points have similar values"

Quiz: Inductive bias

1)[ ] With a strong inductive bias one can reach a low test error with very little training data.

2)[ ] With a strong inductive bias the test error will always be low.

3)[ ] Data augmentation is a heuristic method to get more training data.

4)[ ] In data augmentation there is an inductive bias in the form of our assumptions
5) about reasonable transformations.

6)[ ] Choosing a specific neural network architecture is choosing an explicit inductive bias.

Convolutional Layers

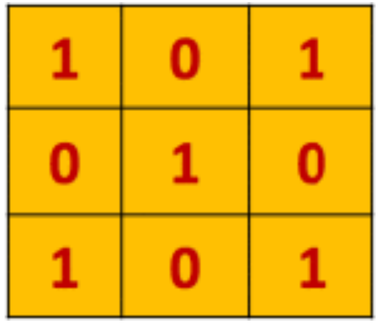
An explicit inductive bias for natural images

Previous slide.

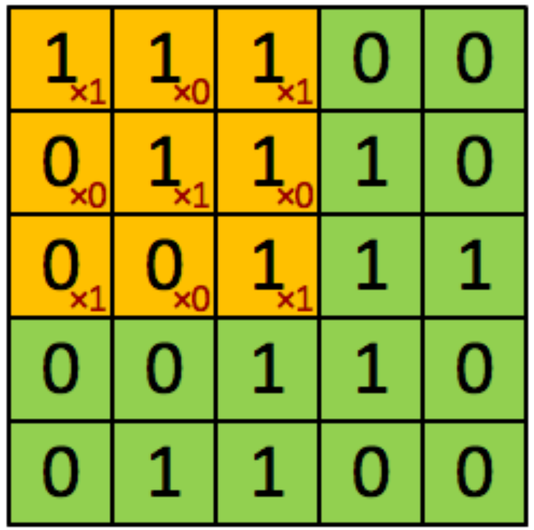
For tasks involving natural images, like object recognition, we have now strong empirical evidence, that neural networks with convolutional layers have the better explicit inductive bias than networks with only dense layers (all-to-all connectivity).

We will start now with a recap of convolution.

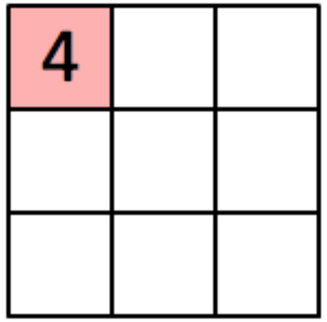
Convolution: one feature



Filter (or Feature)



Image



Convolved Feature

w_{xy}

I_{xy}

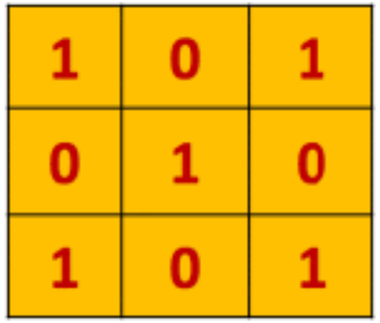
$$a_{ij} = b + \sum_{x=1}^{3} \sum_{y=1}^{3} I_{i+x-1,j+y-1} w_{xy}$$

http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution

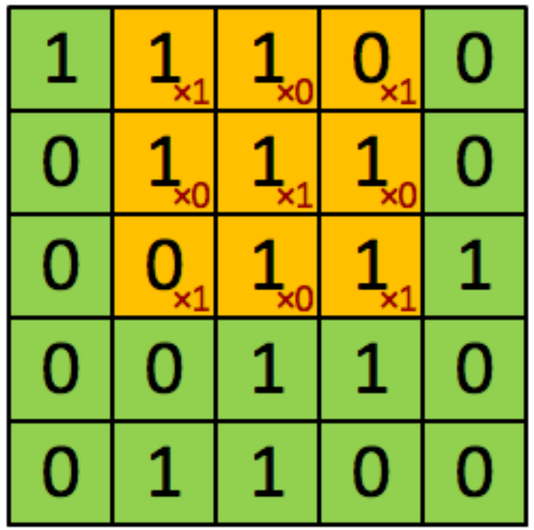
Previous slide.

On the left we have a 3x3 filter. Applying it to the top-left corner of the 5x5 image in the middle means multiplying each filter weight with the corresponding pixel value and summing the results to get the activation $a_{11} = 4$ for bias $b = 0$.

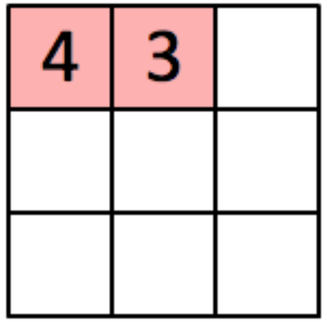
Convolution: one feature



Filter (or Feature)



Image



Convolved Feature

w_{xy}

I_{xy}

$$a_{ij} = b + \sum_{x=1}^{3} \sum_{y=1}^{3} I_{i+x-1,j+y-1} w_{xy}$$

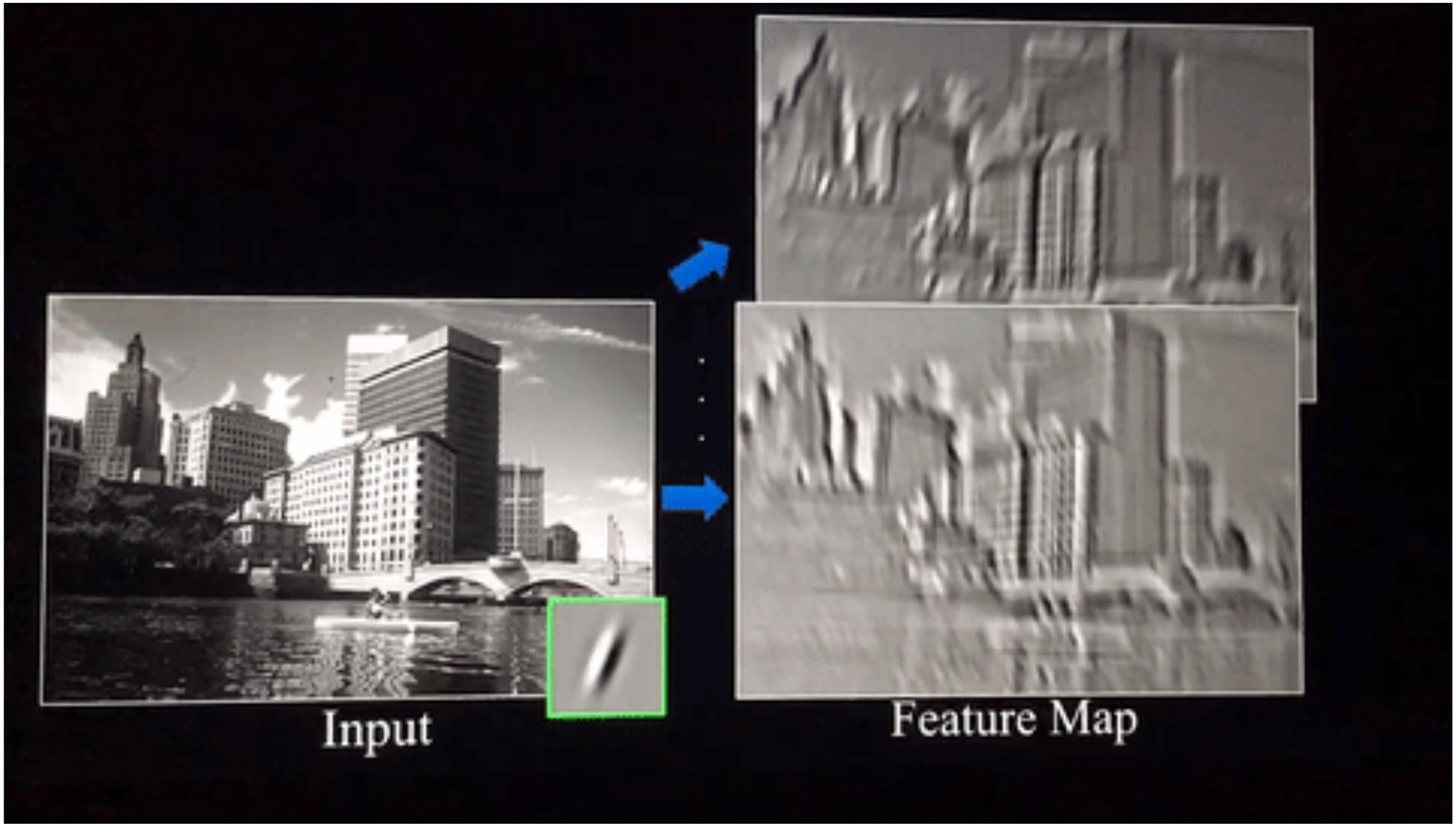
http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution

Previous slide.

Next we move the filter over the image and repeat the multiplication and summation at each position to get all values of the convolved feature matrix on the right.

Note that in contrast to a “standard neuron” that would take the full image as input and compute one activation value, we get multiple activation values with a “convolutional neuron” (filter): there is one value for each position at which the filter (neuron) is applied.

Convolution: multiple features



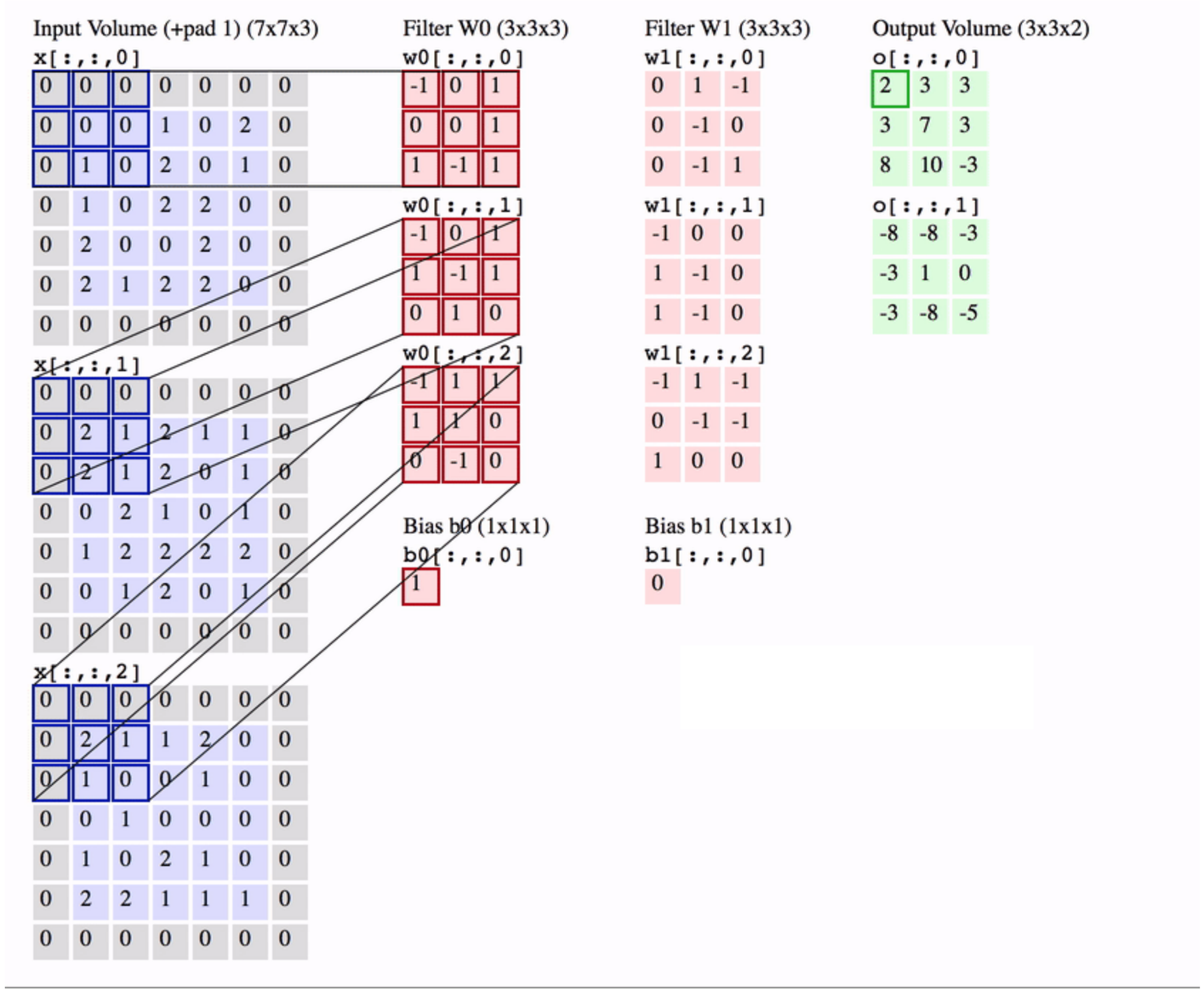
https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

Previous slide.

For multiple filters we get multiple convolved features, also called feature maps. Note how the different simple filters in this example extract edges of different orientation.

One convolutional layer is usually composed of multiple filters. Since each feature map is two-dimensional we can think of the output of a convolutional layer as a three-dimensional object. We will discuss this in more details on the next-but-one slide.

Convolution: stride and padding



- Stride 2

filters jump 2 pixels at a time as we convolve

- Images are sometimes padded with 0 at the borders

http://cs231n.github.io/convolutional-networks/

Previous slide.

So far we applied each filter starting at the top left-corner and moving it row by row over the image. This is known as convolution with padding 0 and stride 1.
Here we consider two generalizations:

First, if filters are applied only at every n’th position, we speak of stride n.

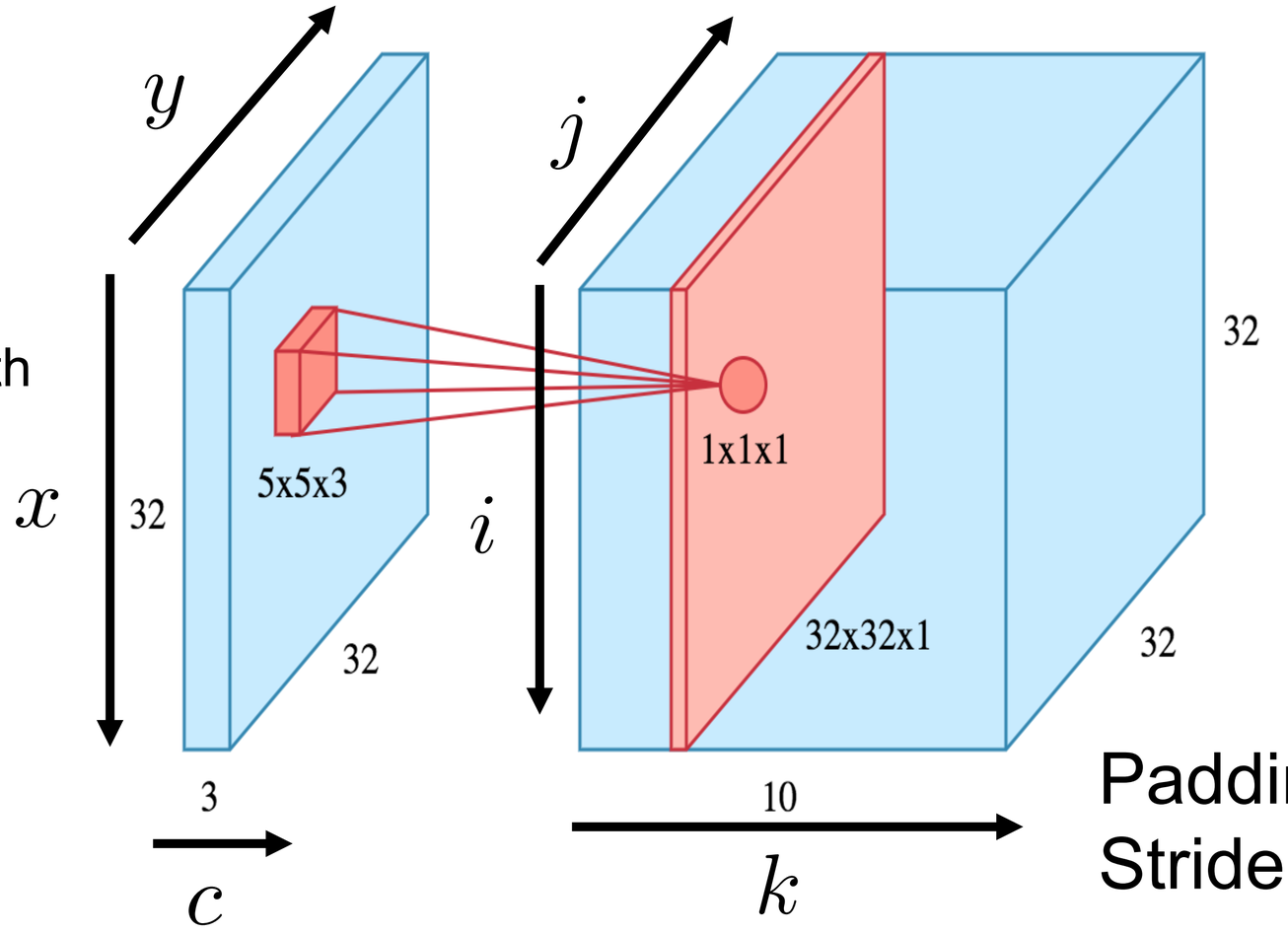
Second, if one wants to position the center of a filter to the top-left pixel of an image, this can easily be achieved with padding the original image, i.e. appending rows and columns of zeros to the original image.

Of course, stride and padding can in general be different in x and y direction.

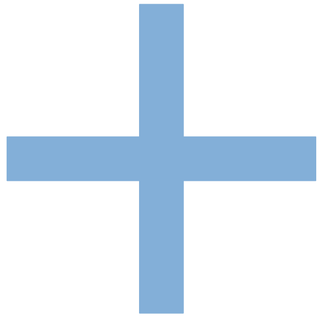
Note, that using a stride > 1 results in a reduction of the x-y dimensions of the feature map, but if we would use stride 1 in this example, the input without padding and the output would have the same x-y dimensions (5x5) thanks to padding.

Convolution: one layer

color image
width x height x depth
32 x 32 x 3



filters, e.g



w_{xyc1}

w_{xyc2}

Padding 2
Stride 1

I_{xyc}

$$a_{ijk} = b_k + \sum_{x=1}^{5}\sum_{y=1}^{5}\sum_{c=1}^{3} I_{i+x-1,j+y-1,c} w_{xyck}$$

https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

Previous slide.

On the left we have a color image of 32x32 pixels with 3 color channels. We represent it as a volume of dimensions 32x32x3. If we convolve it with 10 filters of size 5x5x3 with padding 2 and stride 1 we get a volume of dimensions 32x32x10. One of those 10 filters could be a detector for blue crosses or for vertical yellow bars.

What would be the output volume if we used padding 0 instead of padding 2? 28x28x10
What would it be if we used stride 2 instead of stride 1 for padding 2? 16x16x10
What would it be if we used 24 filters instead of 10? 32x32x24

Convolution: computing the volumes

n x n x c input volume (e.g. image) k filters of size f x f x c

stride s
padding p

$$\left(\frac{n+2p-f}{s}+1\right)\times\left(\frac{n+2p-f}{s}+1\right)\times k$$

Proof: exercise

Previous slide.

In general the size of the output volume can be computed with the above formula.

Inductive bias of a convolutional layer

1) Equivariance to translation
2)f(x) is equivariant to g if f(g(x)) = g(f(x))
3)“A feature detector (such as a vertical edge detector) that’s useful in one part of the image is probably useful in another part of
4) the image.”

5) Only local interactions matter

A conv layer is like a standard dense layer with
1) many neurons having the same weights
2) many weights zero (except in a small neighborhood)

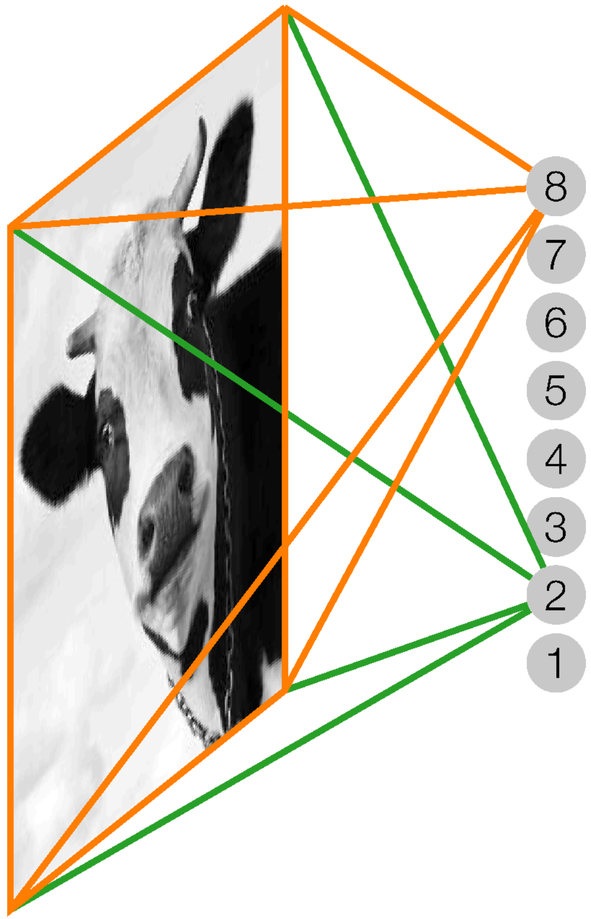
Previous slide.

With a convolutional layer, even if the training set had a certain feature only in one part of the images – say, always in the upper part of the images – it will be detected in a test image also when it appears in another region.

Since the filters are typically much smaller than the full image, they are sensitive only to the configuration of pixels in a small neighborhood.

With dense layers one could achieve something similar to a convolutional layer with setting for each neuron all weights to zero except those in a small region of the input space and using data augmentation. We will look at this in more details on the next slides.

From dense to convolutional layers

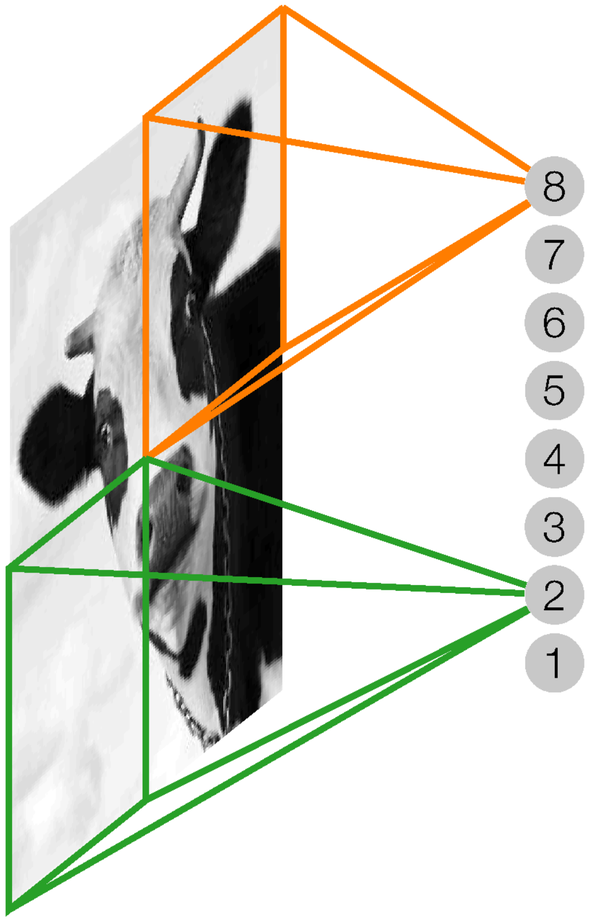


dense (all-to-all) connectivity

Previous slide.

Let us take the image on the left as input and a dense layer with 8 neurons that all connect to all pixels of the input image.

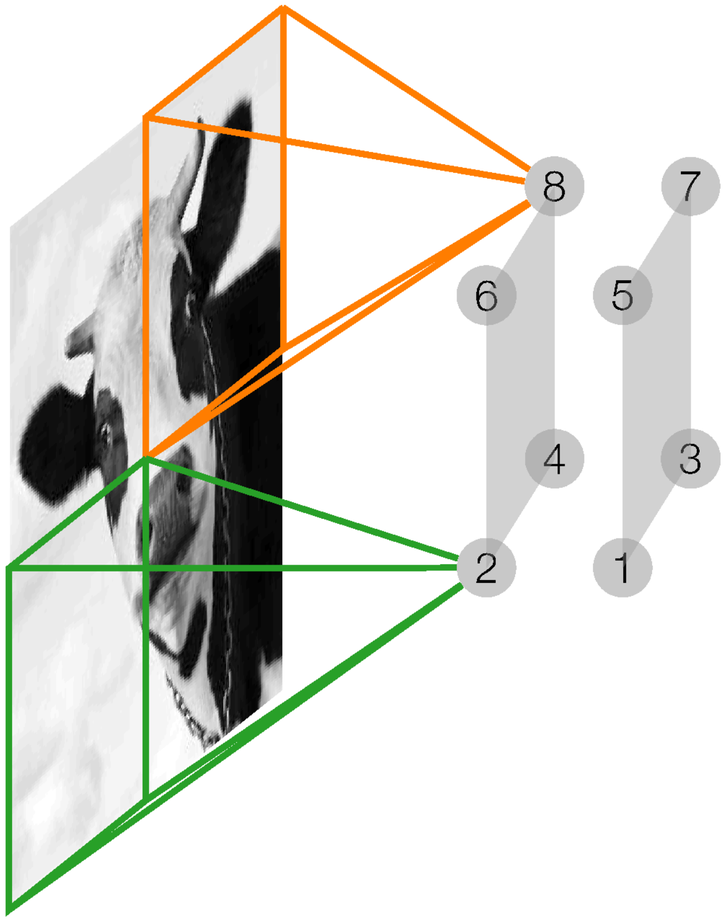
From dense to convolutional layers



Previous slide.

Now let us set most weights of these neurons to zero, such that neurons 1 and 2 only have non-zero weights to pixels in the bottom left quadrant of the image, neurons 7 and 8 only have non-zero weights to pixels in the top right quadrant etc.

From dense to convolutional layers



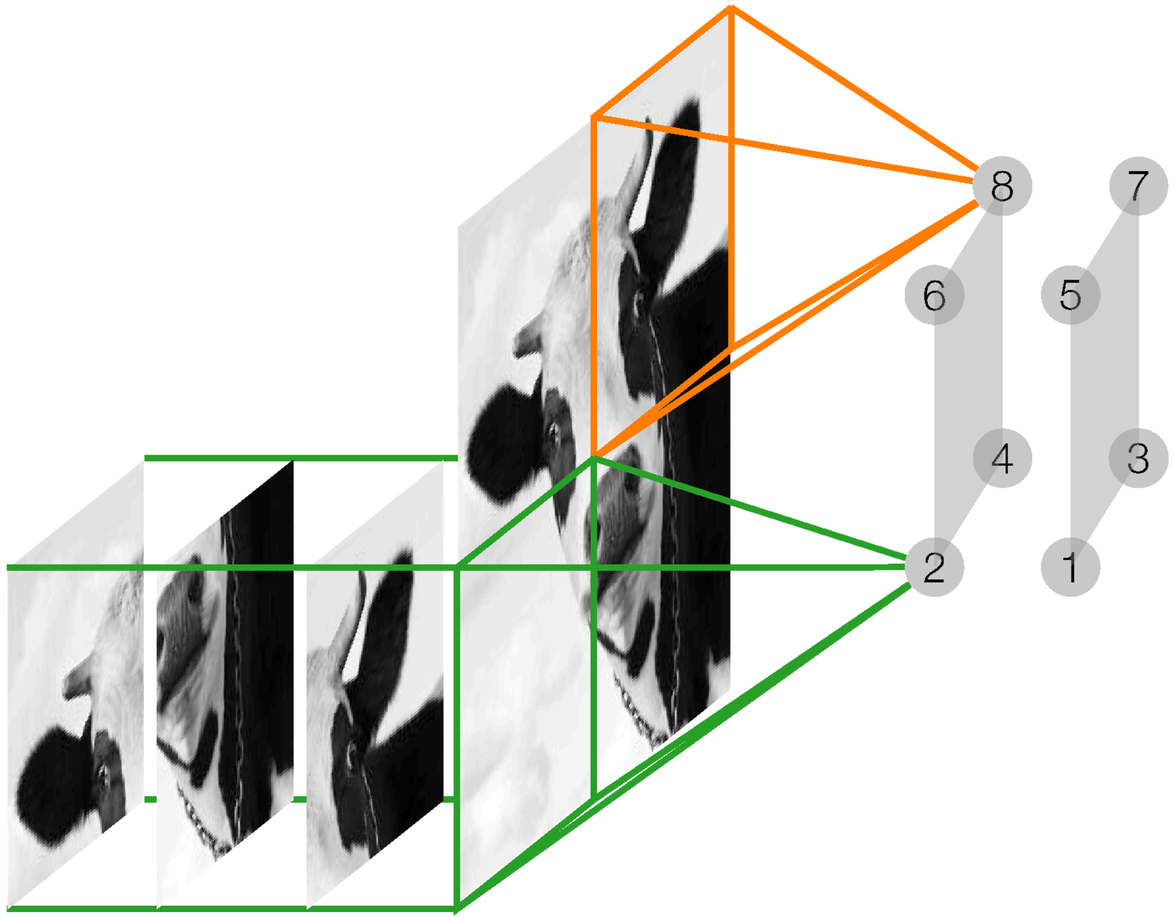
patchy connectivity

Previous slide.

Now we rearrange the neurons according to the region where they have non-zero input weights. This already resembles a lot the volumes we saw for convolutional networks.
We will call the left vertical plane the “even feature map” and the other one the “odd feature map”.

Let now all neurons in each feature map have exactly the same weight vector, e.g. the orange weight vector and the green weight vector are the same. Think of the green weight vector as the concatenation of the weights from each row of the patch.
But neurons in different feature maps shall have different weight vectors.

From dense to convolutional layers



patchy connectivity & data augmentation

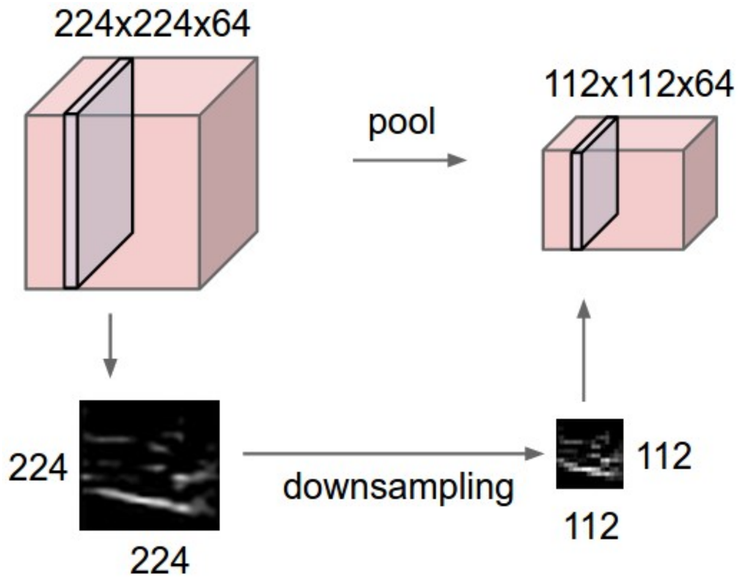
Previous slide.

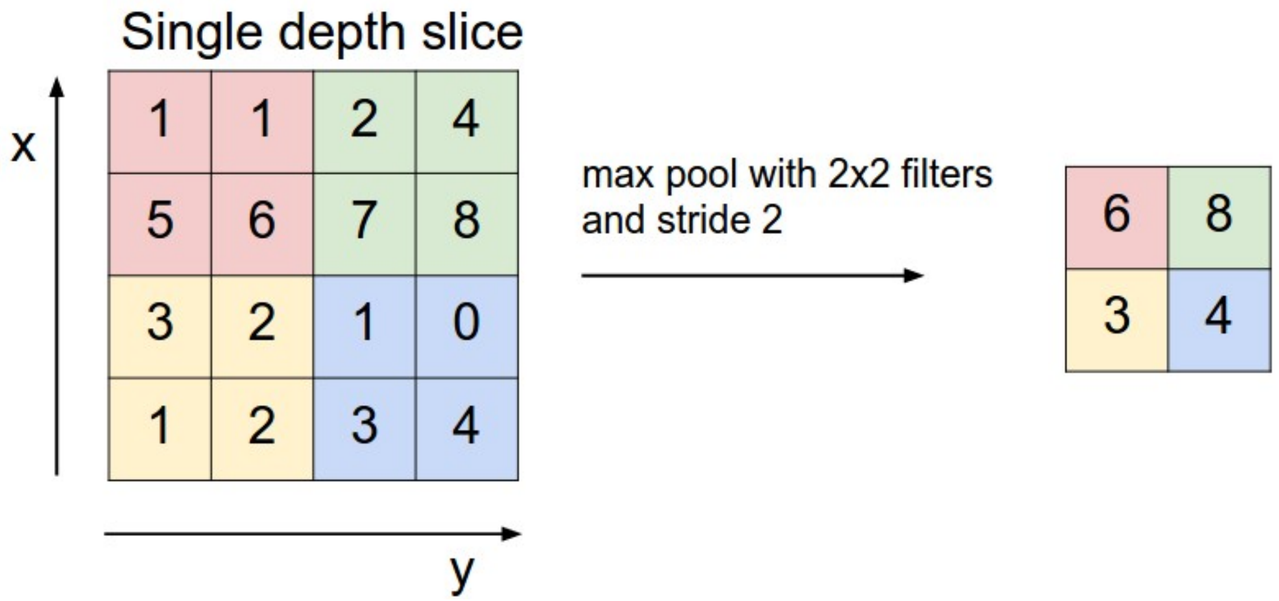
During training we now do data augmentation, such that each neuron sees each patch of the image equally often. Since the weight vectors of the neurons in each feature map were initialized to the same values and all neurons see the same training data, the weight vectors of all neurons in the same feature map will always remain the same.

You should realize now that we have total equivalence to a convolutional layer. But instead of keeping different neurons with the same weight vectors we replace all the neurons in one feature map with a single filter and instead of data augmentation, we move this filter over all patches of the image.

In the introductory example we discussed, how setting some weights to zero and doing data augmentation installs an inductive bias. Here we see how such an inductive bias can be absorbed in the architecture of the network.

Max Pooling





Inductive bias of max pooling

Invariance to small translations with some probability

f(x) is invariant to g if f(g(x)) = f(x)

“quite a few activations remain the same when wiggling the image”

Previous slide.

Let us move on to a second component that is often used in convolutional networks: max pooling. Any kind of pooling reduces the x-y dimension of a volume; by a factor two in each dimension in the example on the left.

A max-pooling layer with a filter of size 2x2 and stride 2 picks the maximal number in groups of 4 neighboring pixels, as illustrated on the right.

If we move the input image on the right one pixel to the left and pad with zeros, the red, green and blue output would remain the same and the yellow output would switch from 3 to 2. A max-pooling layer thus implements an inductive bias that small translations should not have a large effect on the output of the neural network. Note, however, that the outcome changes quite a bit, if instead one moves the input one pixel to the right.

Instead of using the maximum, people sometimes use also the median or the mean.

Reducing the size of the output volume may be desirable from the perspective of computational resources. Pooling is however only one way to achieve this: increasing the stride in a convolutional layer achieves the same.

Critic of max-pooling

"The pooling operation used in convolutional neural networks is a big mistake and the fact that it works so well is a disaster." Geoff Hinton

https://www.reddit.com/r/MachineLearning/comments/2lmo0l/ama_geoffrey_hinton/clyj4jv/
https://openreview.net/pdf?id=HJWLfGWRb

https://www.youtube.com/watch?v=pPN8d0E3900

"We find that max-pooling can simply be replaced by a convolutional layer with increased stride without loss in accuracy on several image recognition benchmarks."

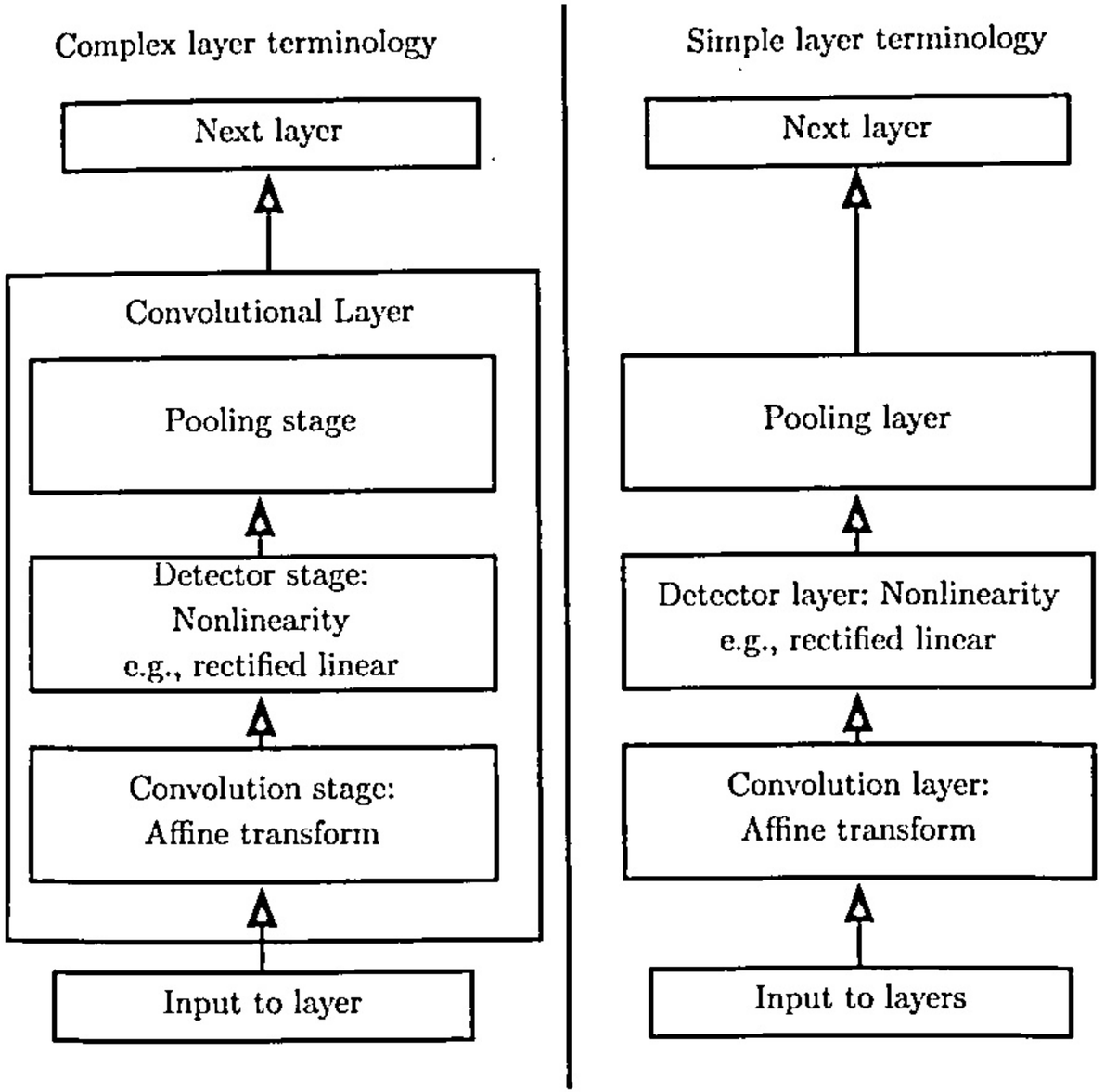
https://arxiv.org/abs/1412.6806

See also https://openreview.net/forum?id=HJeuOiRqKQ

Previous slide.

The popularity of pooling layers has decreased in recent years.

Components of a typical conv-net layer

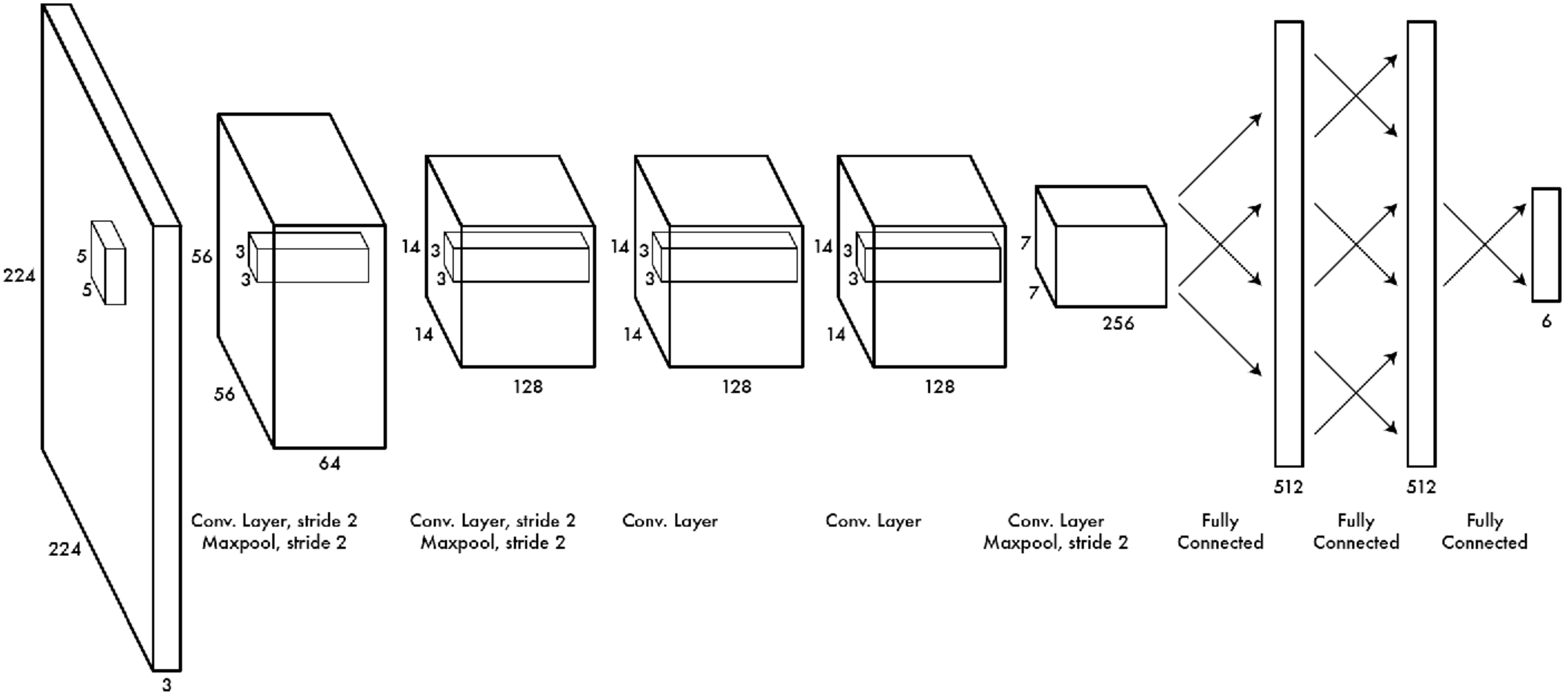


Goodfellow et al. Chapter 9.3

Previous slide.

Nevertheless, pooling was so popular that even in textbooks the definition of convolutional layers sometimes includes a pooling stage, as in the complex layer terminology on the left, where one layer consists of a convolution, nonlinearity and a pooling stage.

Convolutional network



http://doi.org/10.1109/ICRA.2015.7139361

Previous slide.

Putting everything together we can draw nice images like these to describe the network architecture. You should now be able to understand such graphs.

E.g.
How many filters are there in the first convolutional layer? 64
What padding was used in the first convolutional layer? 2. Two times stride of 2 reduces each dimension by a factor 4. Since 224/56 = 4, we conclude that the center of the filters was placed also at all edges of the image, which for a filter of size 5x5 means that a padding of 2 is needed.

Quiz:

1)[ ] The number of weights in a convolutional layer with 10 filters of size 5x5x3 is 750 (excluding biases).

2)[ ] The number of weights in a convolutional layer does not depend on the size of
3) the x-y dimension of input layer.

4)[ ] The number of outputs of a convolutional layer with 10 filters of size 5x5x3 is 250.

5)[ ] Convolutional layers are also equivariant under rotations of the image.

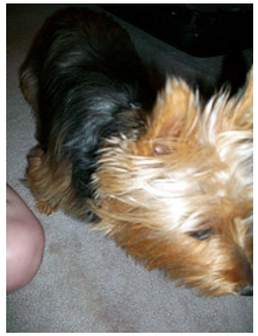
6)[ ] Max pooling layers are also likely to be invariant under rotations of the image.

7)[ ] You have a dataset of centered portraits (passport photos) on white background.
8) The equivariance property of ConvNets is a good inductive bias for this dataset.

Modern ConvNets and ImageNet Competitions

Imagenet challenge 2012

example images of australian terriers

- 1.2 million images training set
- 1000 categories
- 50 thousand validation set
- 100 thousand test set

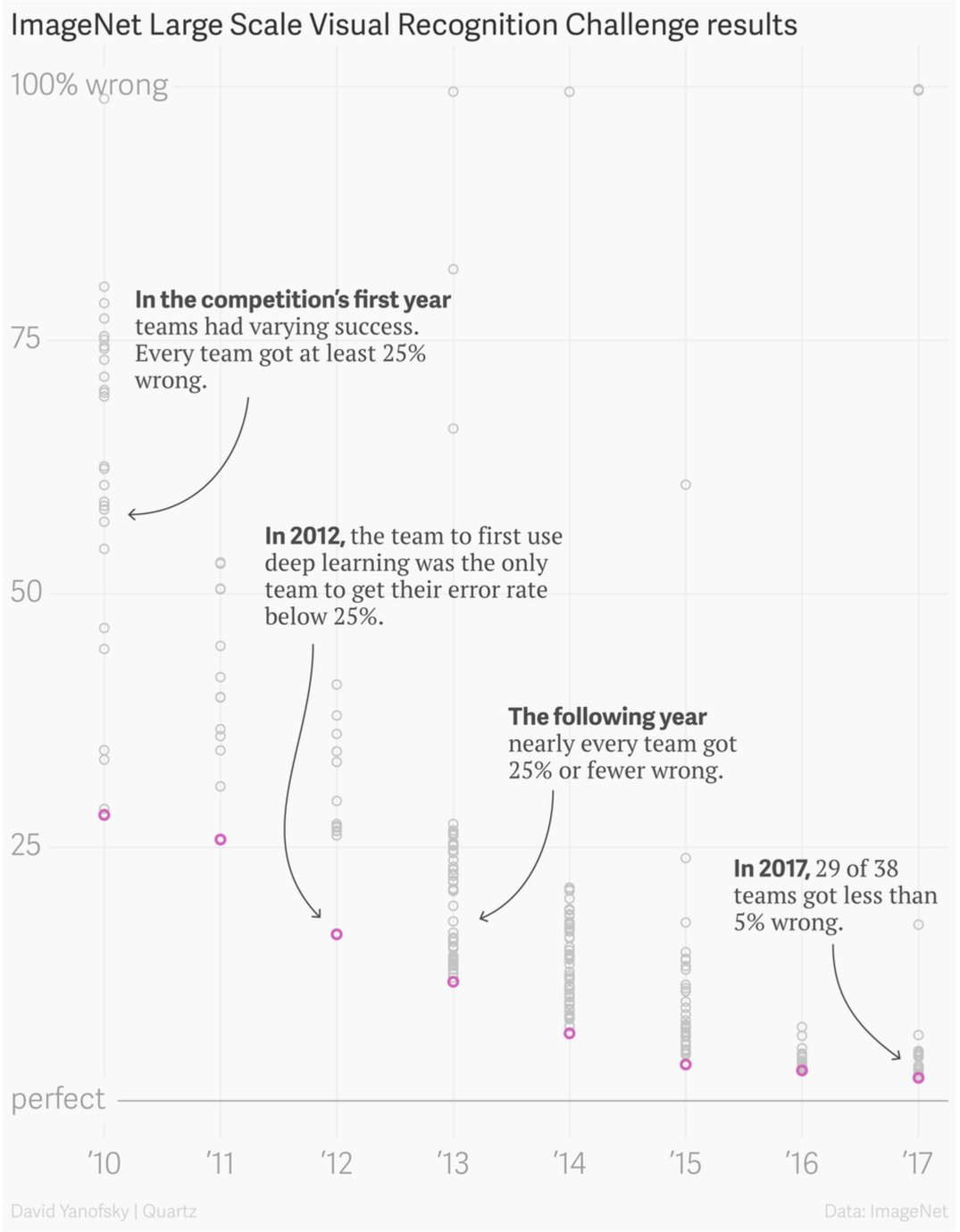
http://www.image-net.org/

Previous slide.

The imagenet competitions have extensively been used to demonstrate progress in deep learning. The competition in 2012 contained 1.2 hand-labeled images containing objects in 1000 classes; these numbers have increased in the meantime by at least an order of magnitude.

Imagenet challenges

http://www.image-net.org/
https://www.kaggle.com/c/imagenet-object-detection-challenge

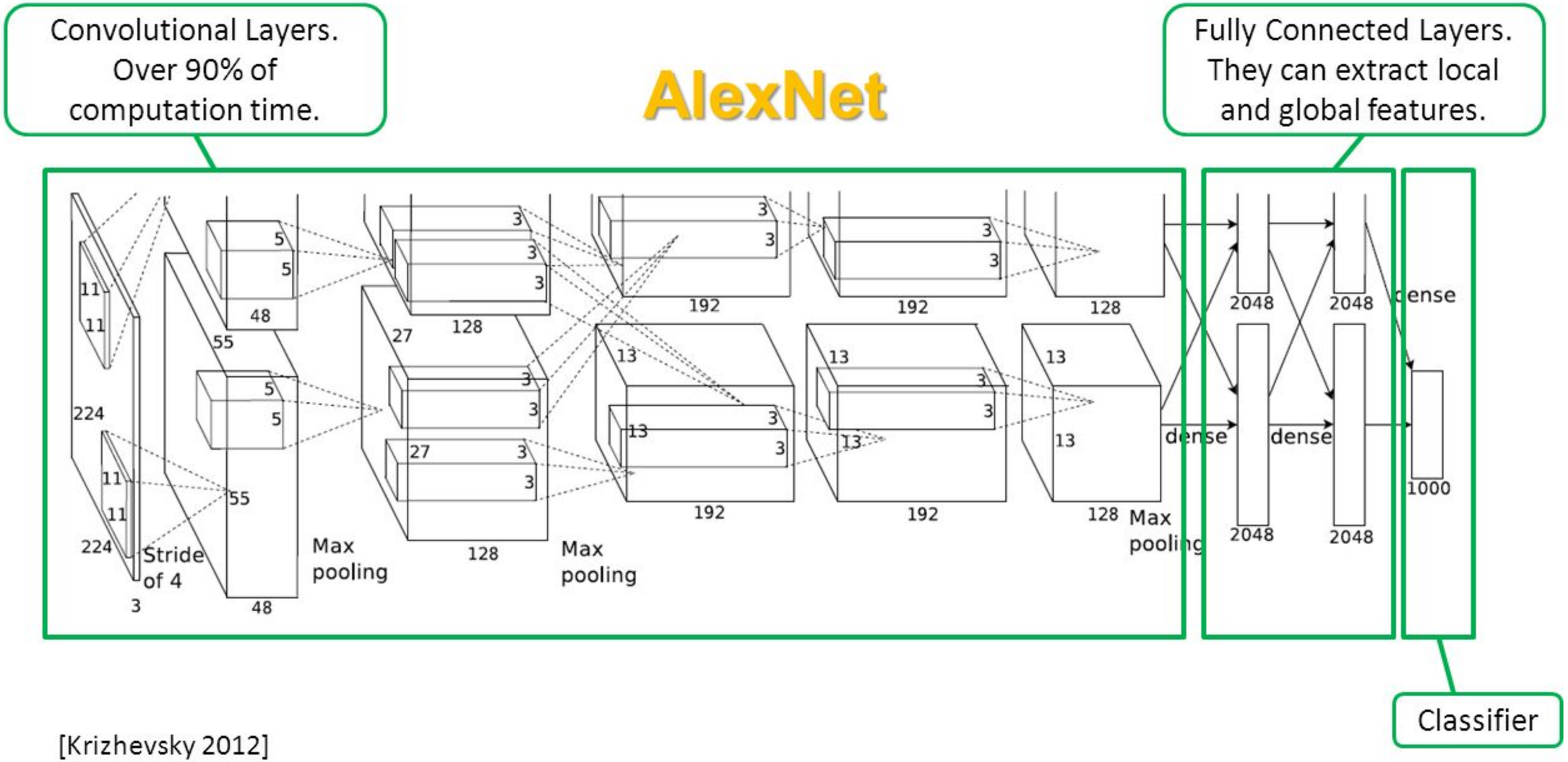


https://qz.com/1046350/the-quartz-guide-to-artificial-intelligence-what-is-it-why-is-it-important-and-should-we-be-afraid/

Previous slide.

Left: Challenges include now also finding the bounding box of different objects.
Right: Within only a few years the error rate decreased from 25 to 5 percent thanks to advances in the design and training of convolutional networks.

AlexNet: winner of Imagenet challenge 2012



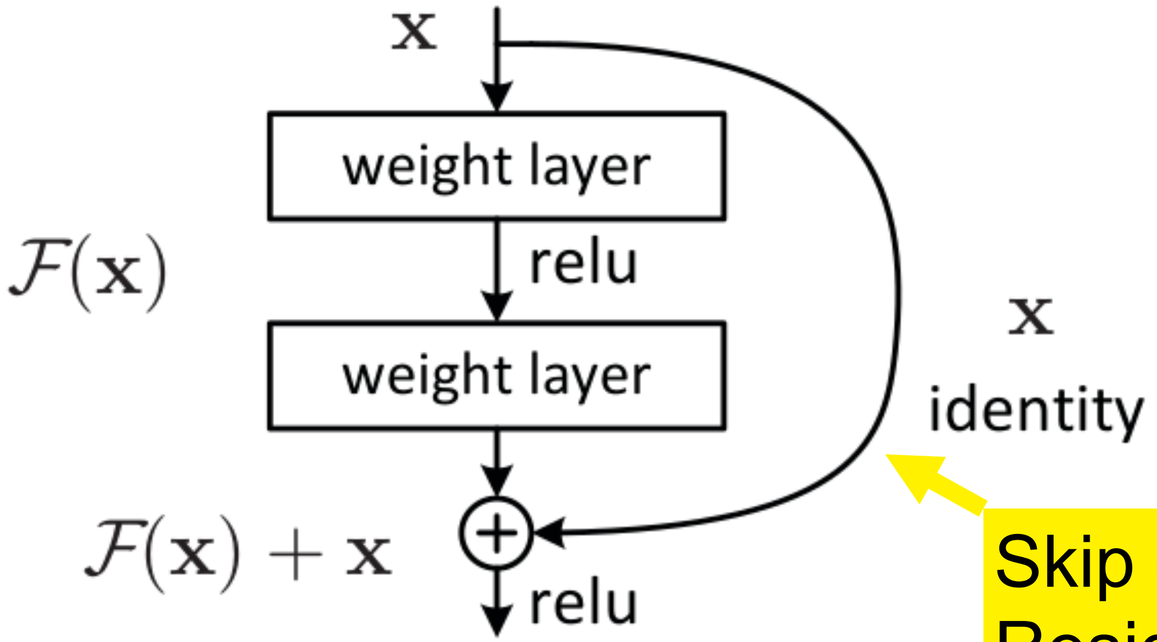
[Krizhevsky 2012]

http://slideplayer.com/slide/10259573/34/images/5/AlexNet+Convolutional+Layers.+Over+90%25+of+computation+time..jpg

Previous slide.

The best method of the 2012 competition, now called AlexNet, still had a pretty conventional architecture with elements that you have seen up to now in this lecture. We will now start to look at newer features that helped to further improve convolutional networks.

ResNet



Better training in very deep (>100 layers) networks.

Skip connections
Residual connections

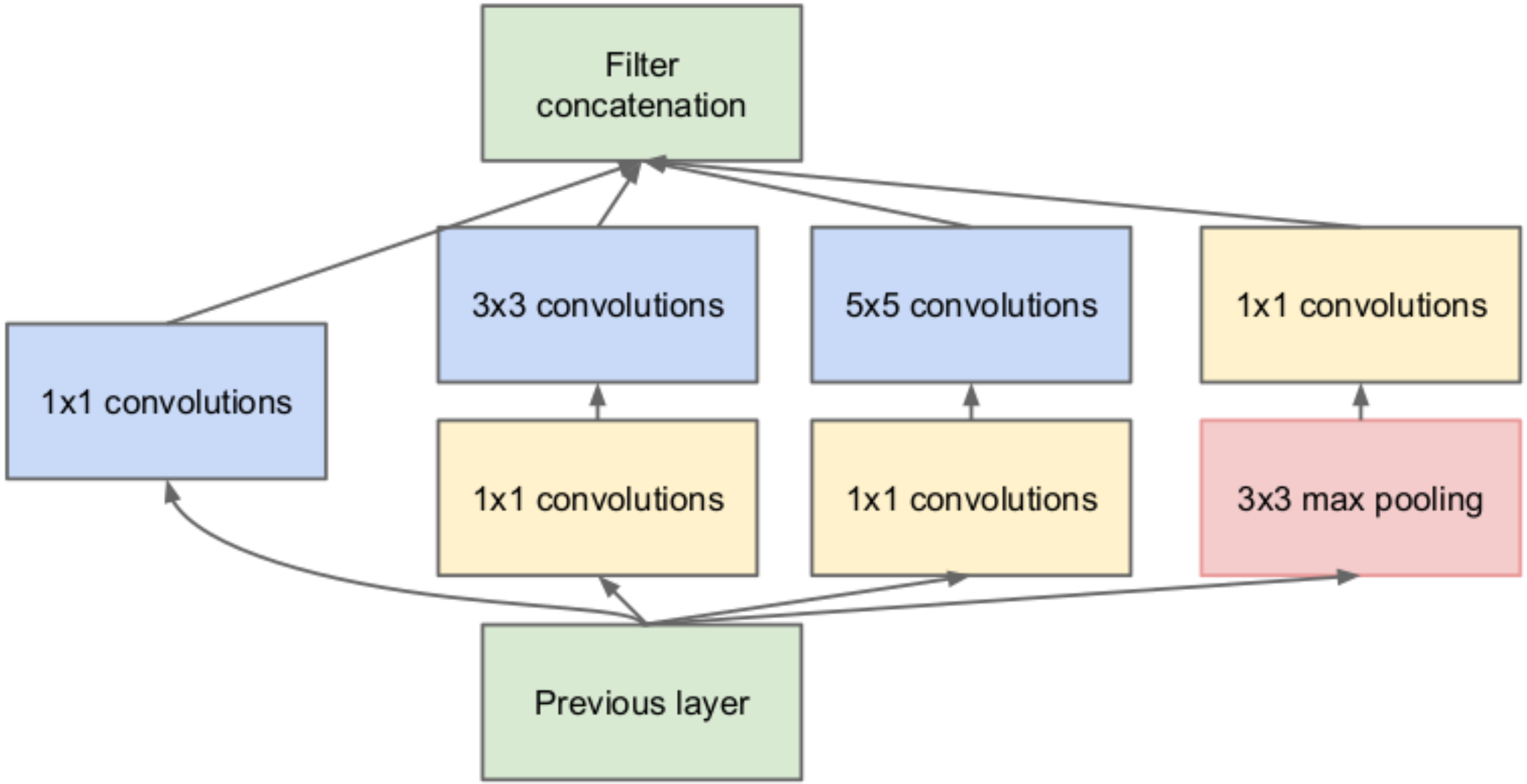
https://arxiv.org/pdf/1512.03385.pdf

Previous slide.

In theory, the training loss should decrease with deeper and deeper networks, simply because they have more parameters. In practice, however, it was observed that successfully training very deep convolutional networks is difficult. To my knowledge, the reasons are not yet well understood, but vanishing of gradients does not seem to be the problem.

The idea of skip connections is to allow the network to “dynamically choose the number of layers” (see blackboard). The layers within the skip-connections learn the residual F(x), the part that is not yet learned by the smaller network that would just output x.

Inception module



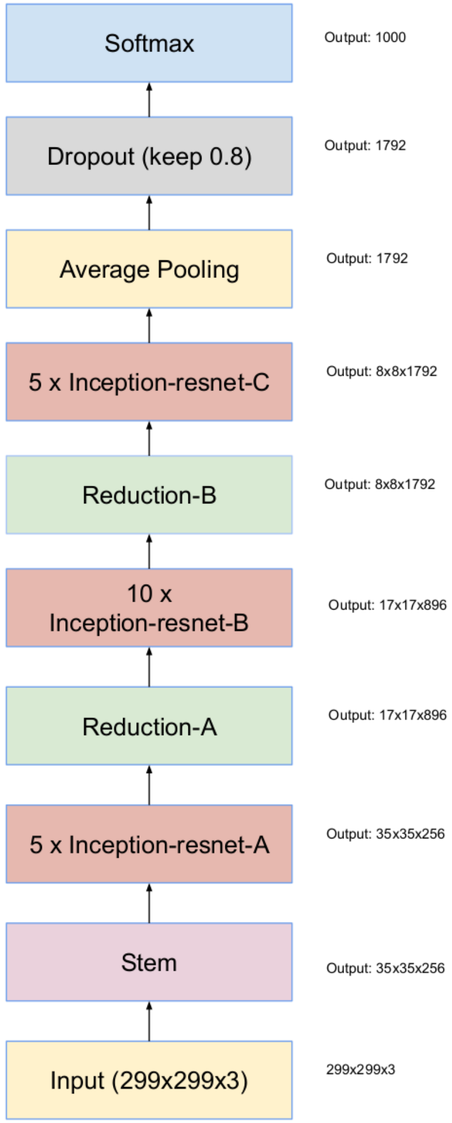
https://arxiv.org/pdf/1409.4842.pdf

Previous slide.

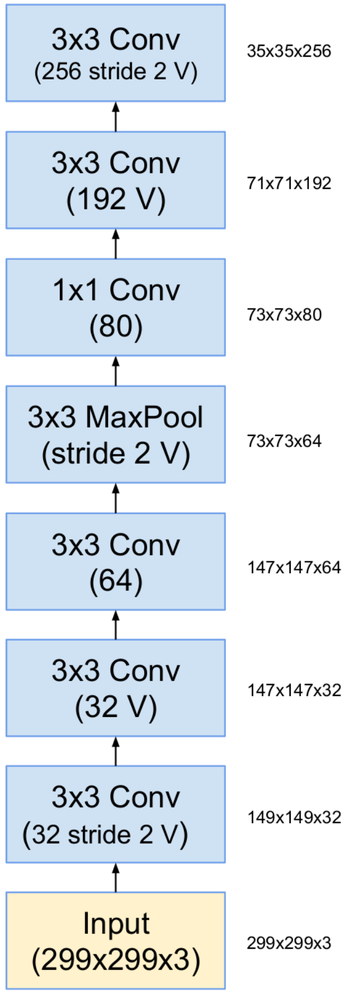
For AlexNet the designers decided where to place convolutional layers with filters of certain sizes and where to place pooling layers. But how should we know what the best architecture is? The inception module leaves this choice to the network itself: each layer contains convolutions with different filter sizes and pooling operations in parallel, not sequentially as e.g. in the complex layer terminology, and the 1x1 convolutions may then learn to select the relevant features or mixture of features from the concatenation of the outputs. The 1x1 convolutions also help to reduce the computational load and the number of parameters. If the green boxes contain a large number k filters (size of the third dimension of the volume), $k_2 < k$ 1x1 filters can reduce this again to a reasonable number. (see also blackboard). Note that the convolution and max-pooling layers have to use the same stride, i.e. if stride 1 is used for the 3x3 and 5x5 filters, than also stride 1 should be used for the max-pooling layer, in contrast to the example shown in the first part of the lecture, where stride matched filter size in the case of max-pooling.

Cp

Inception-ResNet-v2



Full model



Stem

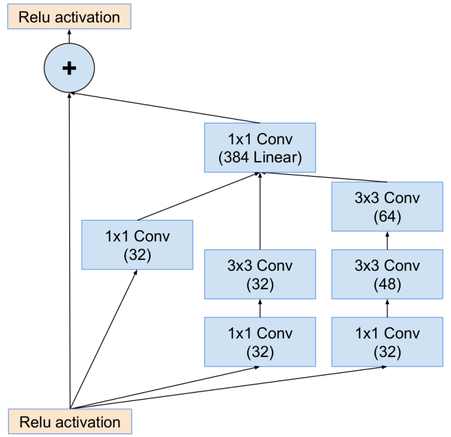
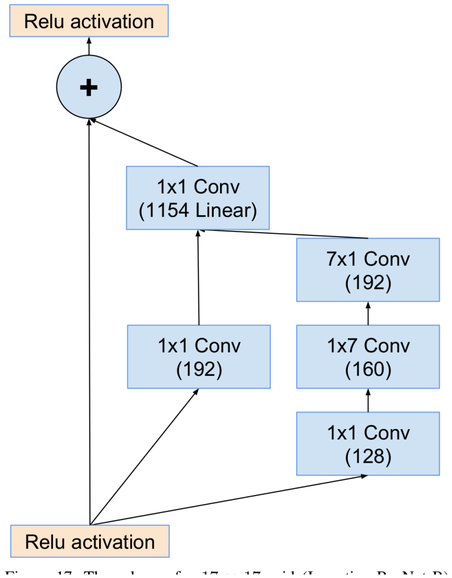


Figure 16. The schema for 35×35 grid (Inception-ResNet-A) module of the Inception-ResNet-v2 network.



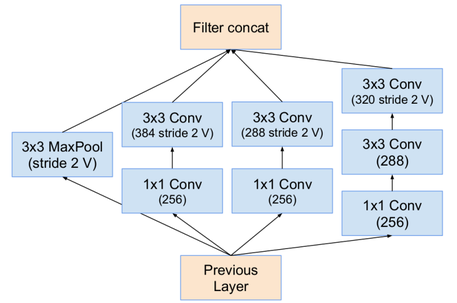


Figure 18. The schema for 17×17 to 8×8 grid-reduction module. Reduction-B module used by the wider Inception-ResNet-v1 network in Figure 15

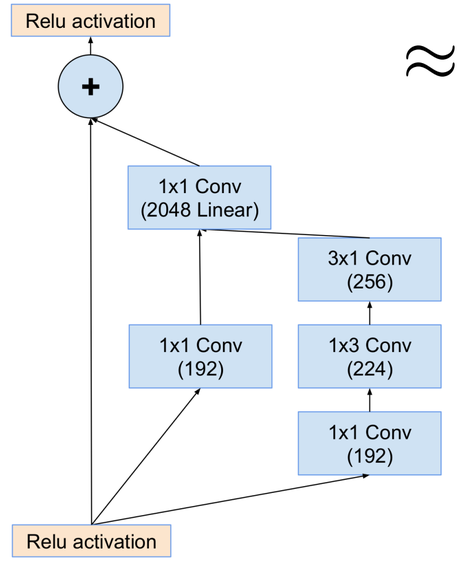


Figure 19. The schema for 8×8 grid (Inception-ResNet-C) module of the Inception-ResNet-v2 network.

Network	k	l	m	n
Inception-v4	192	224	256	384
Inception-ResNet-v1	192	192	256	384
Inception-ResNet-v2	256	256	384	384

Special Layers

$\approx$ 60 million parameters

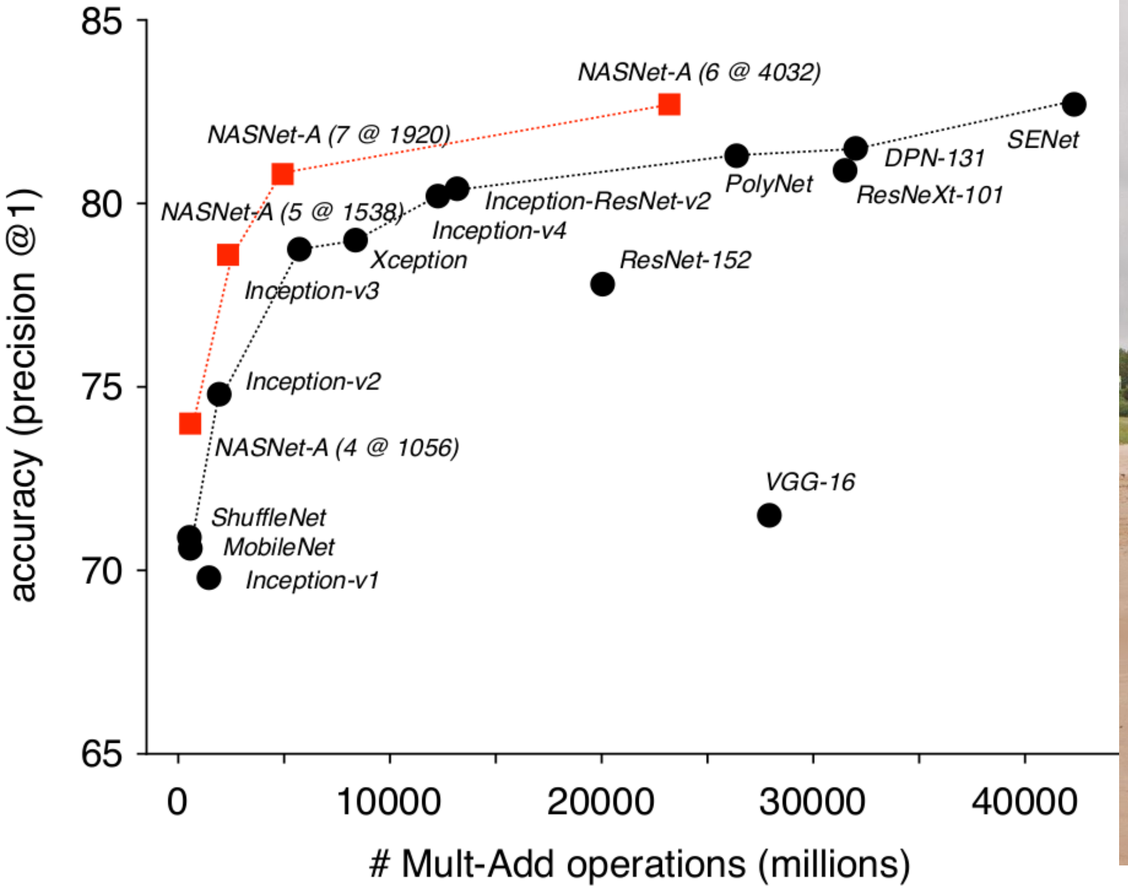
https://arxiv.org/pdf/1602.07261.pdf

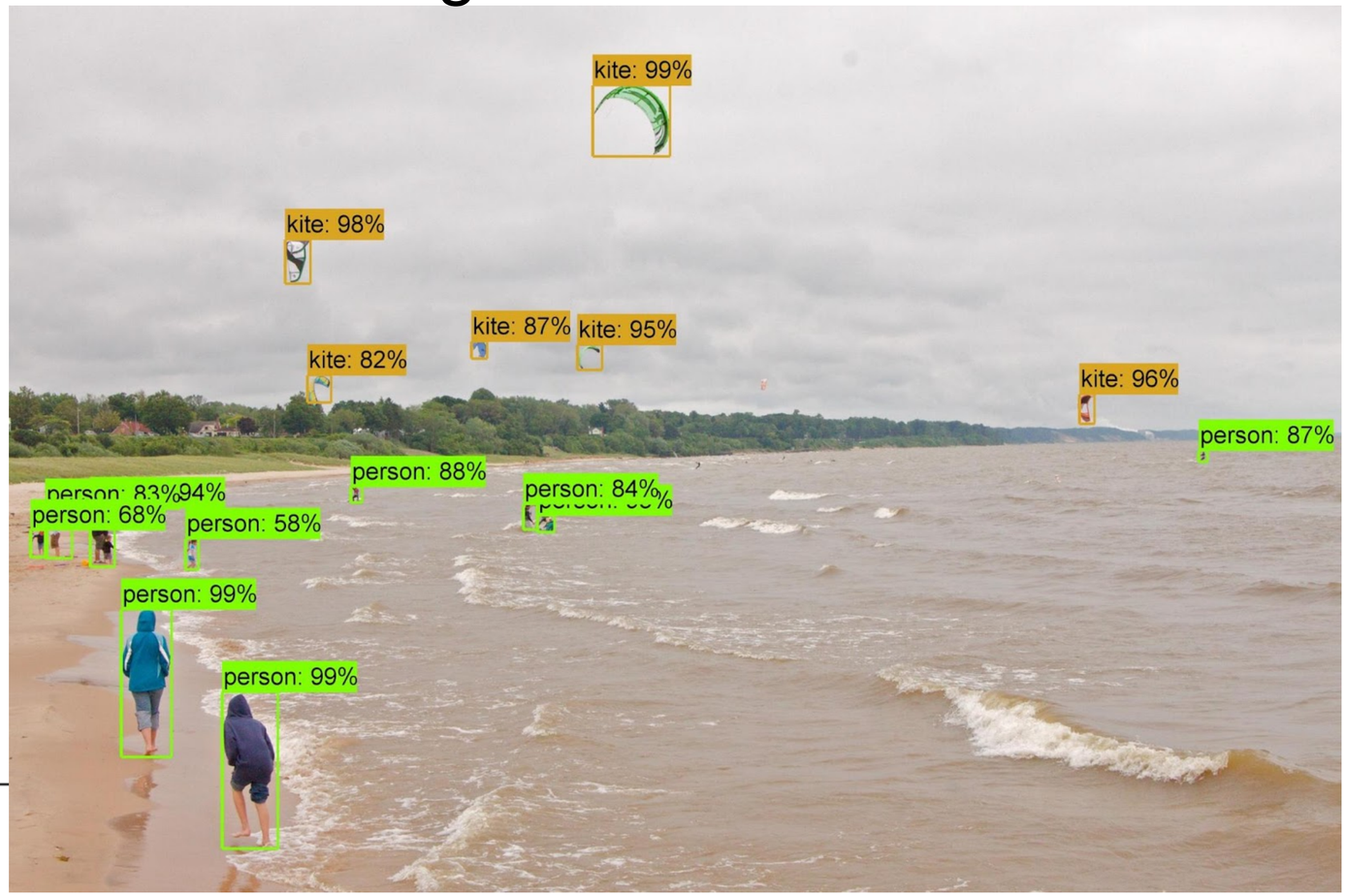
Previous slide.

A full network may look quite scary and use these inception modules several times.

Transfer Learning for Image Recognition

- Find network architecture with reinforcement learning or evolutionary
- programming on a small dataset
- Use the same architectural elements on Imagenet





https://arxiv.org/pdf/1707.07012.pdf

https://arxiv.org/abs/1802.01548

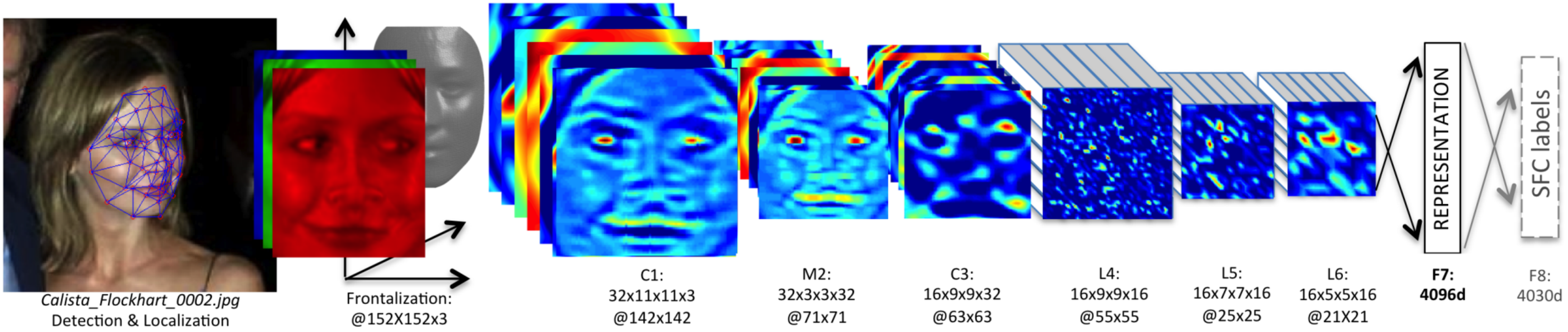
Previous slide.

With skip connections and Inception modules, allowed significant improvements over AlexNet. Yet, the search for the optimal network architecture, i.e. the best inductive bias, continues. Can we automatize this process? Yes, with sufficient computing power one may search for architectures, e.g. with reinforcement learning methods or evolutionary programming. State-of-the-art architectures, like NASNet and AmoebaNet were found like this.

On the left we see that NASNet reaches higher performance on imagenet than hand-crafted variants. In particular the NASNet variants reach consistently higher performance than alternatives with the same number of basic operations. VGG-16 is a famous hand-crafted network with a simple stucture of 13 convolutional layers with 3x3 filters and 3 fully connected layers.

On the right we see an impressive example of what can nowadays be achieved with these networks.

DeepFace



Closing the Gap to Human-Level Performance in Face Verification (2014)

- 4 million user - labeled faces on FaceBook images
- 4000 individuals
- Retrain fully-connected layers at the top on Labeled Faces in the Wild (LFW) dataset reaching (human level) accuracy of 97.35%

Previous slide.

Even though good inductive biases help us reduce the amount of training data needed, it is still crucial to have a lot of labeled data. I find the example of Facebooks face recognition network interesting, because some years ago, millions of users worked for Facebook by labeling their images, without ever getting paid for it and probably without knowing that they helped Facebook to create a state-of-the-art face recognition tool.

Also interesting about this work is the demonstration of transfer: although the network was trained on the faces of Facebook users, it was sufficient to just retrain the fully connected layers at the top to beat all benchmarks on a dataset of images of celebrities.

Automatic Differentation

Shall we manually compute the gradients? (blackboard)

No. Use automatic (reverse mode) differentiation.

- Record computation tree on forward path to
- apply chain rule in reverse order.
- (Same idea as backprop in simple feedforward net)

- An implementation requires just a Tracker (to record the computation tree) and analytical expressions of primitive operations

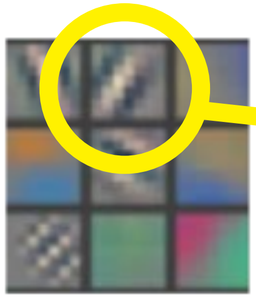
$$\frac{\partial f}{\partial x} \text{ e.g. } \frac{\partial \sin(x)}{\partial x} = cos(x)$$

Applications beyond object recognition

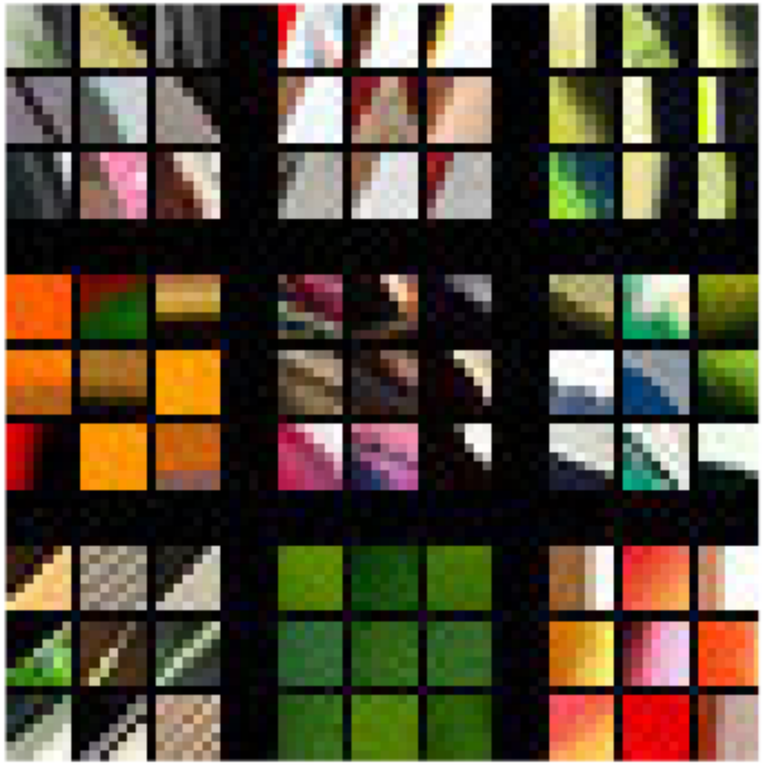
Next 4 slides

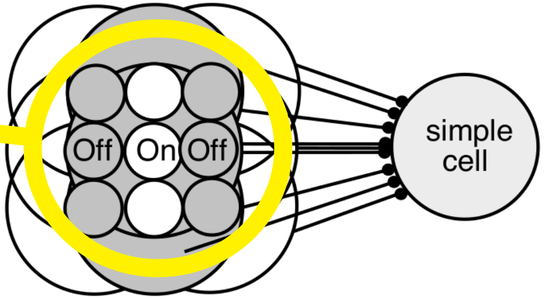
Convolutional neural networks share some features with biological neural networks.
In particular neurons in higher layers tend to respond to more abstract features than neurons in lower layers. In layer 1 we find features like those of complex cells, described in the famous work of Hubbel and Wiesel that studied experimentally the visual system of cats. For layers 2 to 5 on the following slides, we find the top 9 image patches that maximally activated a trained neural network together with a reconstruction of the feature within each patch that was responsible for the high activation. We see that neurons in the second layer respond to basic features like colors or edges of different orientation. In layer 5 for example, we see neurons that respond to the face of a dog with a high degree of invariance, i.e. the feature is not “distracted” by different backgrounds of kinds of dogs.

Neuroscience

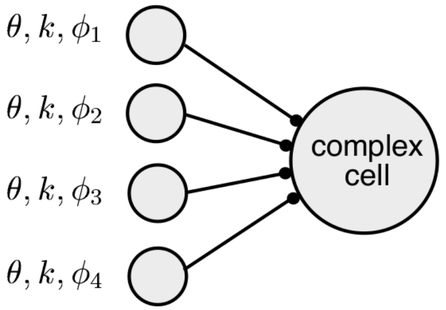


Layer 1





LGN receptive fields



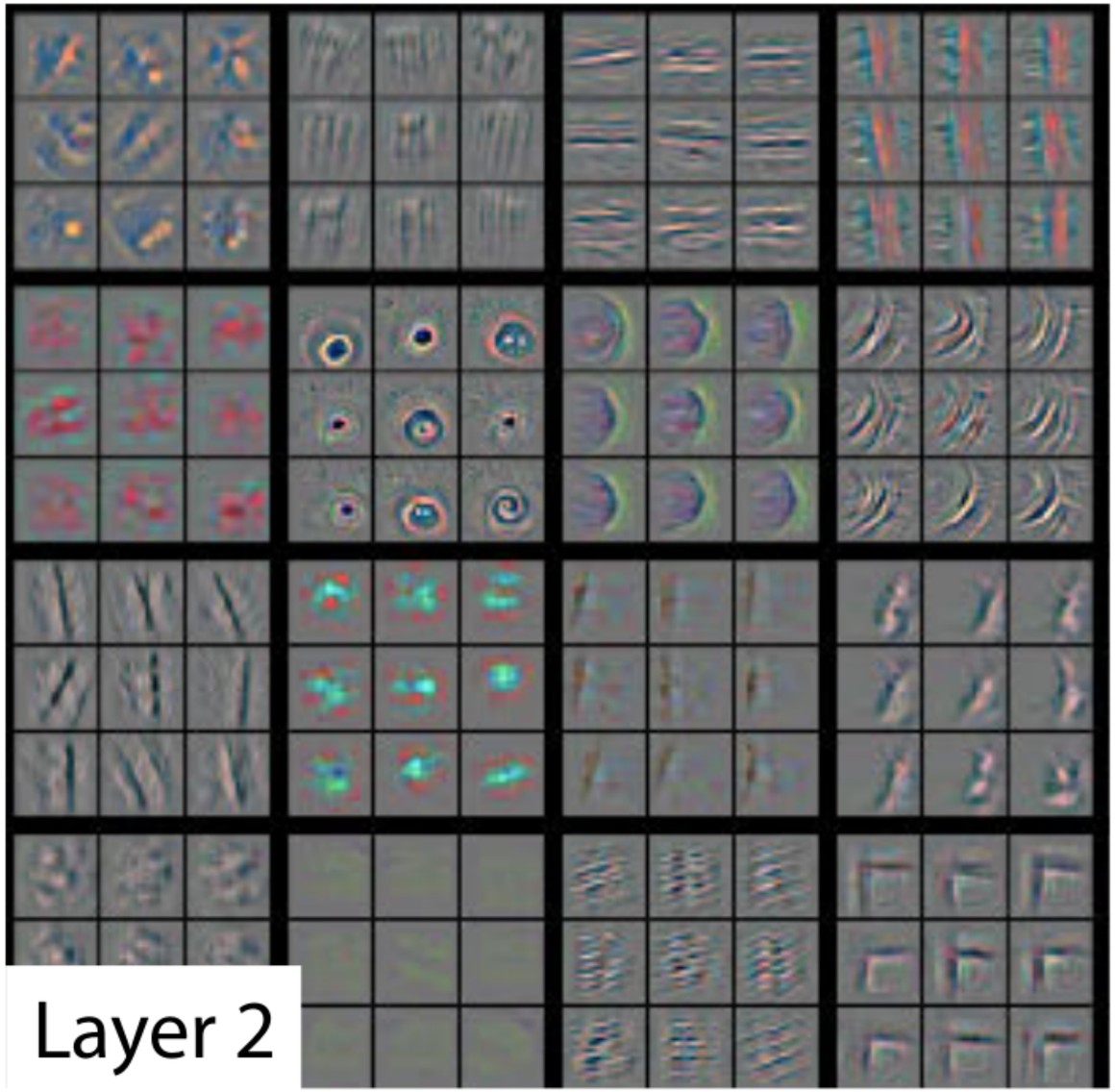
simple cells

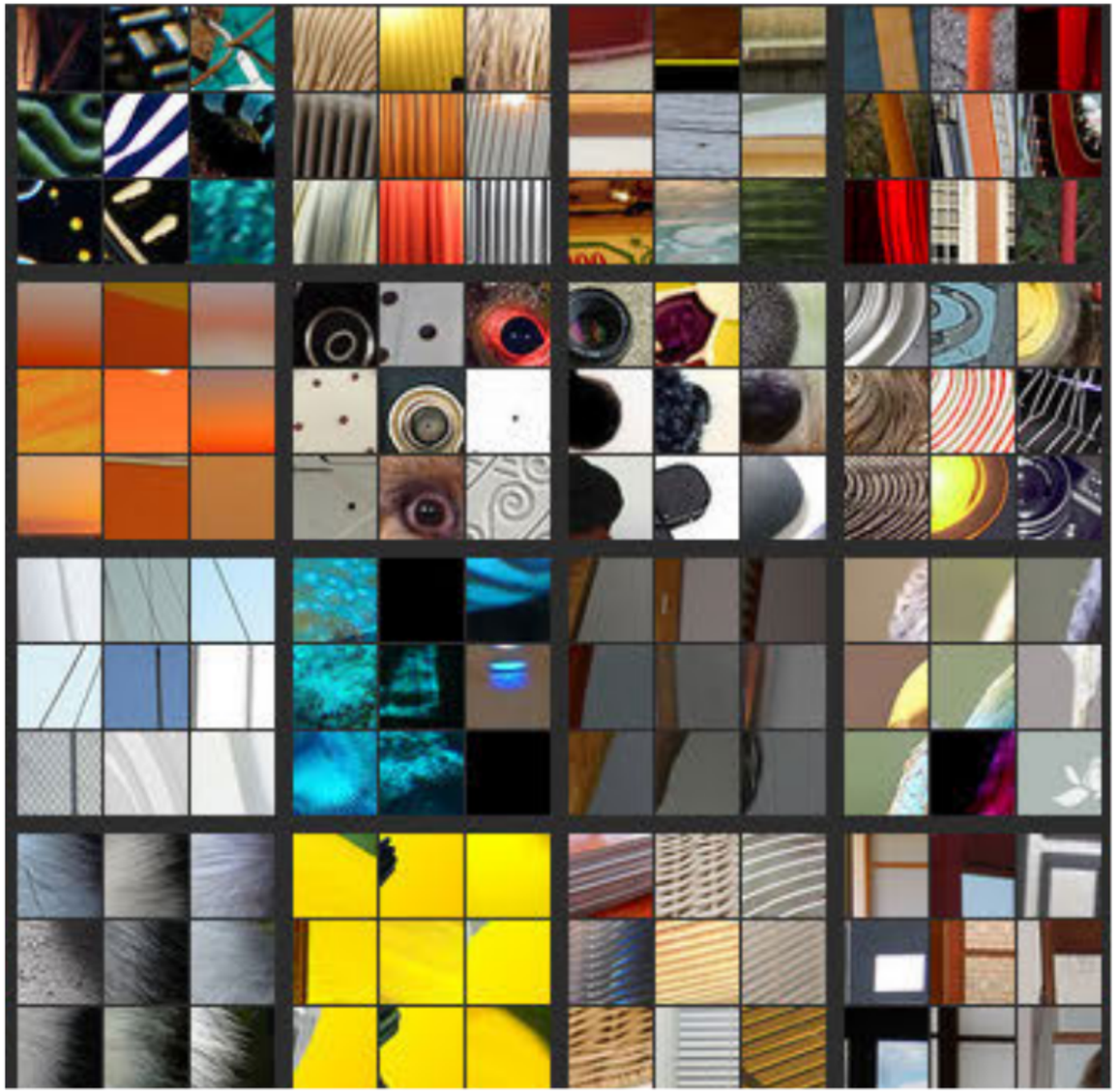
- ConvNets have similar architecture and similar features as the visual system of animals and humans

Dayan and Abbott, Theoretical Neuroscience, 2001

https://arxiv.org/pdf/1311.2901.pdf

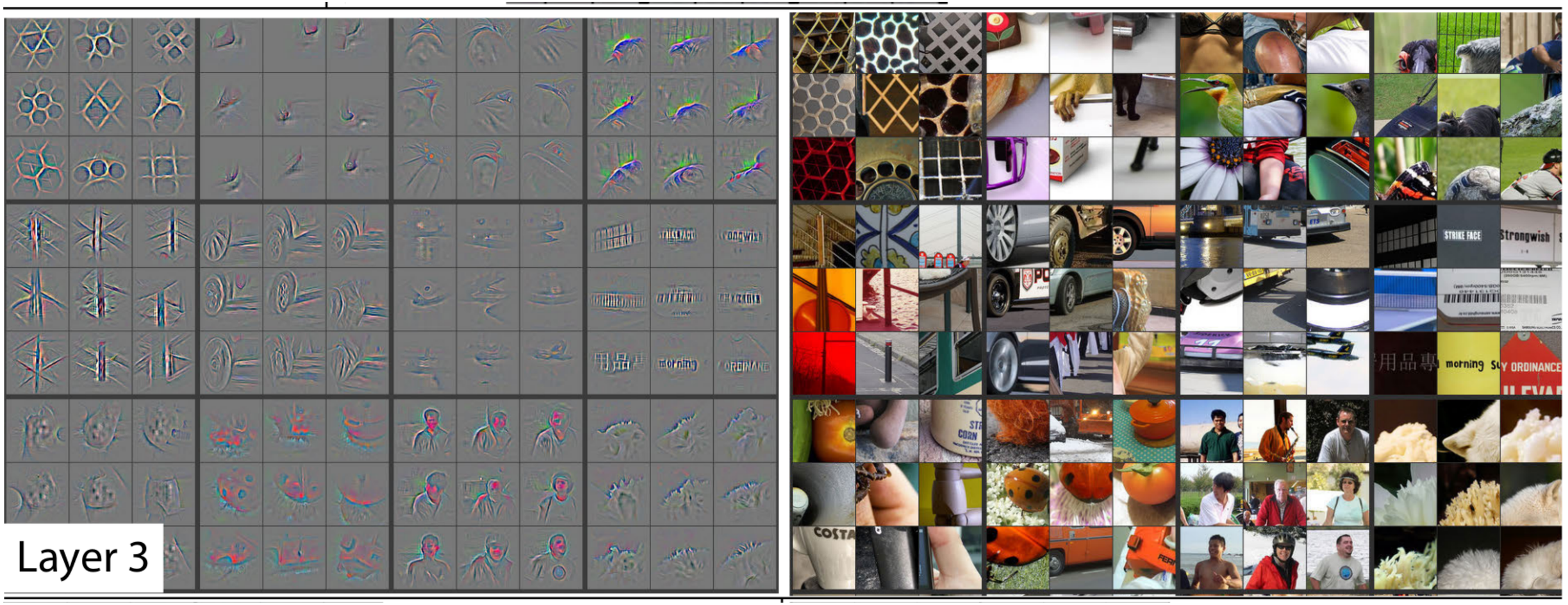
Visualizing convolutional networks





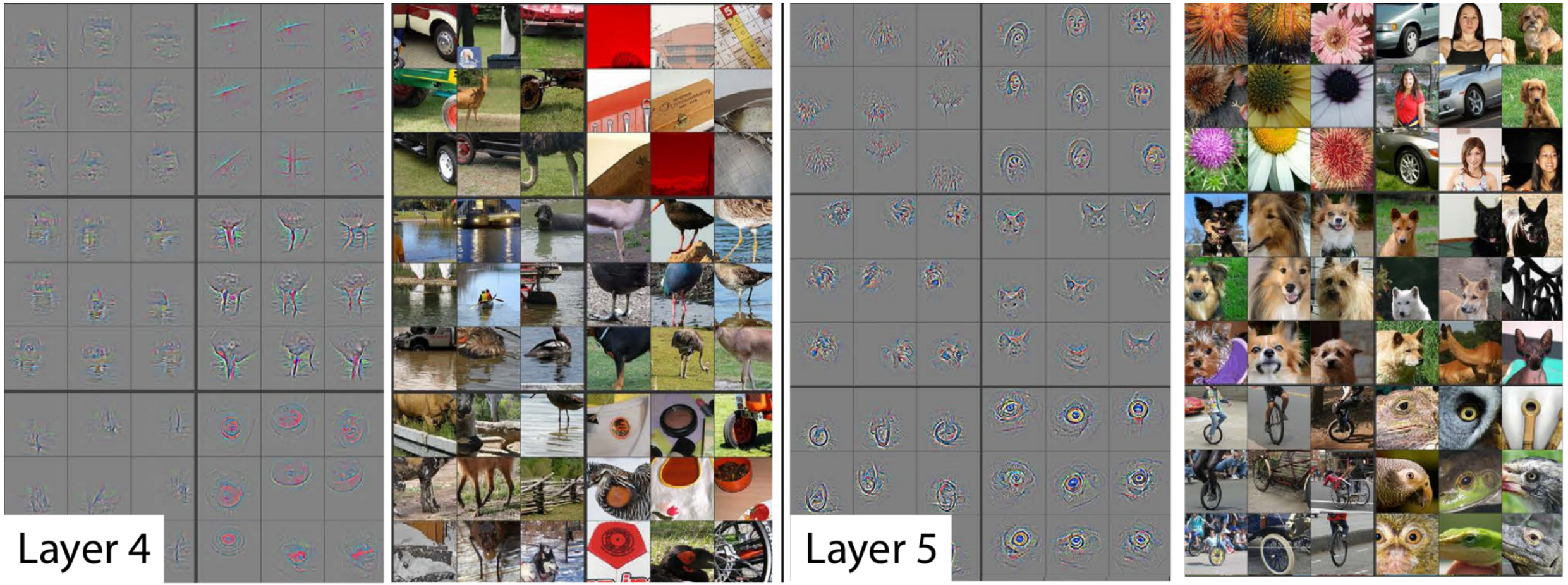
https://arxiv.org/pdf/1311.2901.pdf

Visualizing convolutional networks



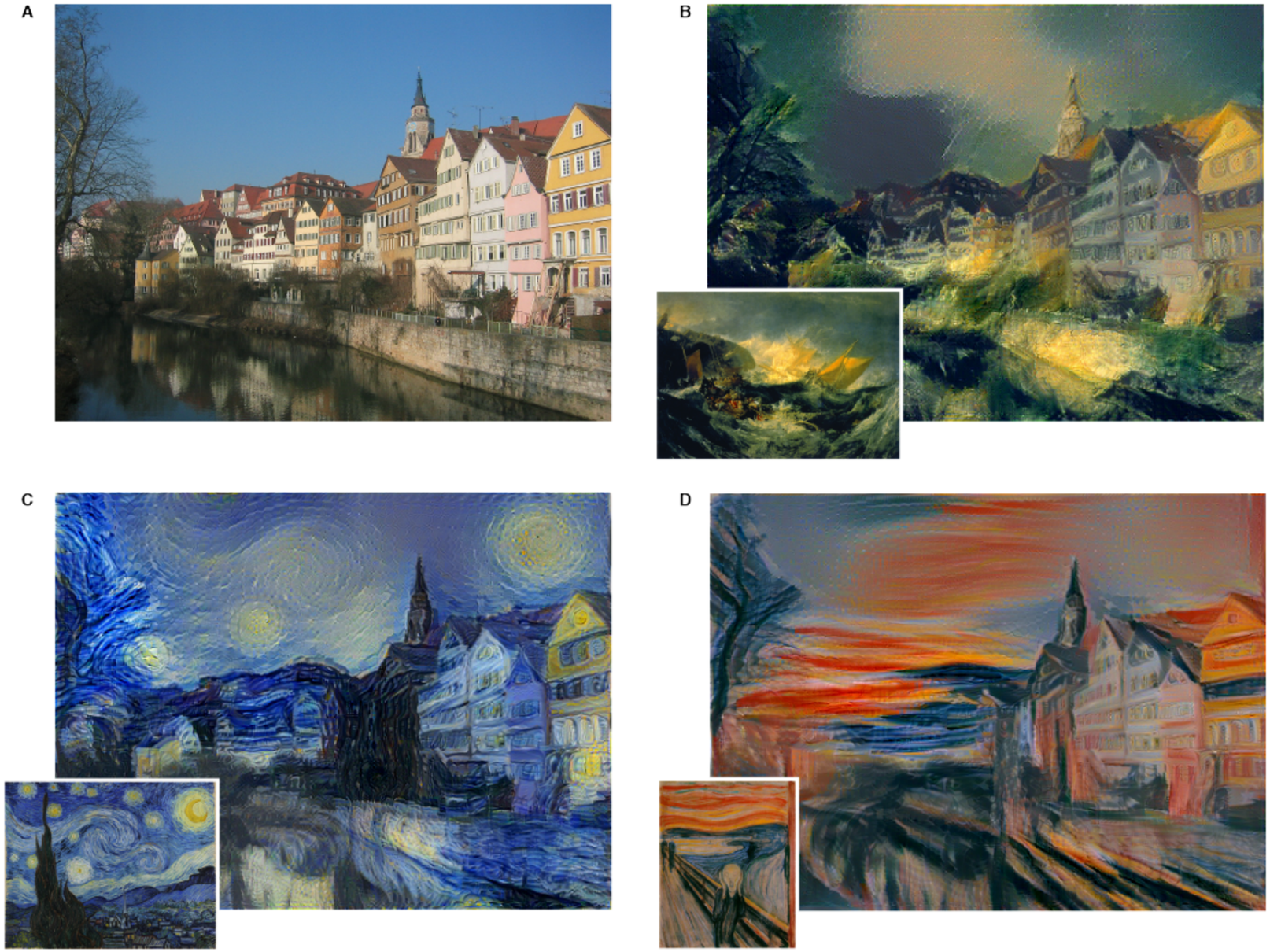
https://arxiv.org/pdf/1311.2901.pdf

Visualizing convolutional networks



https://arxiv.org/pdf/1311.2901.pdf

Neural style



https://arxiv.org/pdf/1508.06576.pdf

Previous slide.

Neural style transfer is a now famous application of convolutional neural networks. Shown is a photo of Tübingen, regenerated by the network in 3 different styles, where the style was extracted from the small images in the bottom left corner of each panel.

Neural style

$$E_L = \sum \left(G^L - A^L\right)^2$$

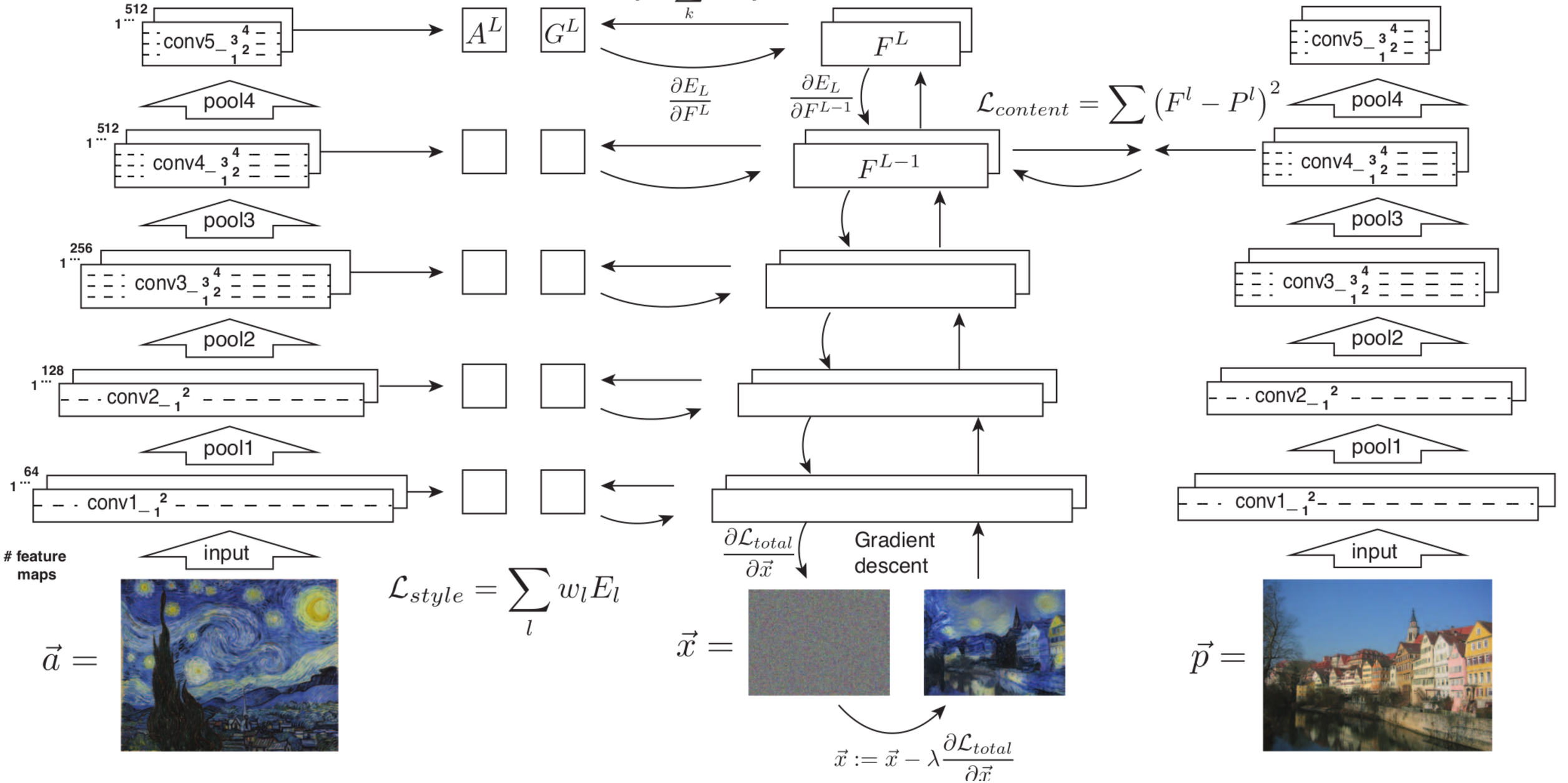
$$\mathcal{L}_{total} = \alpha \mathcal{L}_{content} + \beta \mathcal{L}_{style}$$

$$G^L_{ij} = \sum_k F^L_{ik} F^L_{jk}.$$

$$\mathcal{L}_{content} = \sum \left(F^l - P^l\right)^2$$

$$\mathcal{L}_{style} = \sum_l w_l E_l$$

$$\vec{x} := \vec{x} - \lambda \frac{\partial \mathcal{L}_{total}}{\partial \vec{x}}$$



Previous slide.

This works the following way:
A trained convolutional neural network is activated with an input image p. The activity P^l in each feature layer l in response to this input can than be used to reconstruct the image.

This is done in the following way: start with a new input of random pixel values x and compare the feature response F^l to P^l. Now we minimize the difference between F^l and P^l with gradient descent to find back a reconstruction of p. When using the features in the lower layers the reconstruction is almost perfect.

The style of an image can be reconstructed by following a very similar procedure but instead of comparing the responses, the difference between feature correlations G^l and A^l are minimized.

If the losses of style reconstruction and content reconstructions are mixed, the resulting image will have the content of input image p with the style of input image a.

Caption generation and generative models of images

a cow is standing in the middle of a street

https://cs.stanford.edu/people/karpathy/deepimagesent/

ConvNet + bidirectional RNN

https://thispersondoesnotexist.com/

ConvNets in generative adversarial networks (GANs)

Previous slide.

Convolutional networks together with recurrent neural networks have also enabled better automatic caption generators. On the left the image is given as input and the caption is generated as the output from recurrent neural network that received as input abstract features extracted with a convolutional neural network.

You have seen already in the previous lecture the images of persons that don’t exist that where generated with generative adversarial networks that use convolutional neural networks.

Summary

1) Use good explicit inductive biases together with transfer learning and data augmentation.

2) Convolutions (and max-pooling) are reasonable inductive biases for natural images.

3) Residual connections help to learn in very deep networks.

4) Modern architectures like Inception-Resnet-v2, DeepFace and NASNet reach human level performance on recognition tasks.

5) AutoDiff is a flexible tool to train different architectures.

6) ConvNets and neurons in the visual systems have similar receptive fields.