Convolutional Neural Networks

Bùi Tiến Lên

2023

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

Contents

Notation

symbol	meaning
$a, b, c, N \ldots$	scalar number
$\boldsymbol{w}, \boldsymbol{v}, \boldsymbol{x}, \boldsymbol{y} \ldots$	column vector
$\boldsymbol{X}, \boldsymbol{Y} \ldots$	matrix
$\mathbb{R}$	set of real numbers
$\mathbb{Z}$	set of integer numbers
$\mathbb{N}$	set of natural numbers
$\mathbb{R}^D$	set of vectors
$\mathcal{X}, \mathcal{Y}, \ldots$	set
$\mathcal{A}$	algorithm

operator	meaning
$\boldsymbol{w}^\mathsf{T}$	transpose
$\boldsymbol{X}\boldsymbol{Y}$	matrix multiplication
$\boldsymbol{X}^{-1}$	inverse
$\boldsymbol{X} \odot \boldsymbol{Y}$	an element-wise matrix-vector multiplication

Architecture

Convolutional neural network

Convolutional neural networks (CNNs) use three basic ideas:

- *local receptive fields*
- *shared weights*
- *pooling*

What are learnt in CNNs?

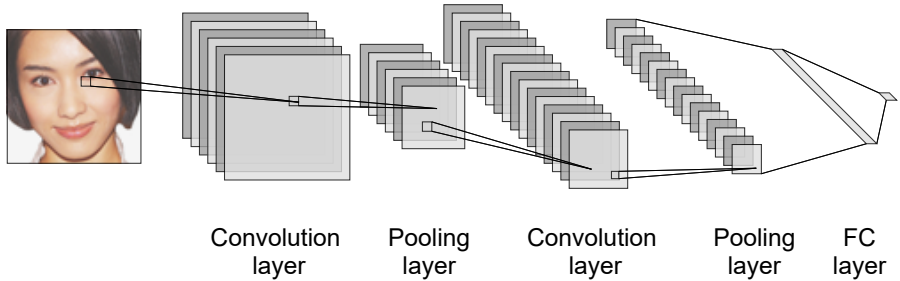
CNNs has two interesting properties

- The patterns they learn are translation invariant
- They can learn spatial hierarchies of patterns

A basic architecture of CNN

- A common pattern

$$\text{Convolution} + \text{ReLU} \rightarrow \text{Pool} \rightarrow \text{Fully connected} \rightarrow \text{Softmax loss}$$



Convolutional layer

- Single Input, Single Output
- Multiple Input, Multiple Output
- Tensor Operation

Convolutional layer

The fundamental difference between a densely connected layer and a convolution layer is this:

- Dense layers learn **global patterns** in their input feature space
- Convolutional layers learn **local patterns** in their input feature space

Convolutional types

There are three common kinds of convolution

- 1D convolution
- **2D convolution**
- 3D convolution

Image processing

In image processing, an input image is convolved with a kernel to create an output image

- **Input image** (*input feature map*) *In*
- **Output image** (*output feature map*) *Out*
- **Filter** (*kernel*) requires three parameters
  - The *spatial extent* (kernel size) F
  - The *stride* S (distance between two consecutive positions of the kernel)
  - The amount of *zero padding* P (number of zeros concatenated at the beginning and at the end of an axis)

Convolution vs. Cross-correlation

- Convolution operation (used in digital processing)

$$Out(x,y) = (K * In)(x,y) = \sum_m \sum_n In(x-m, y-n)K(m,n) \quad (1)$$

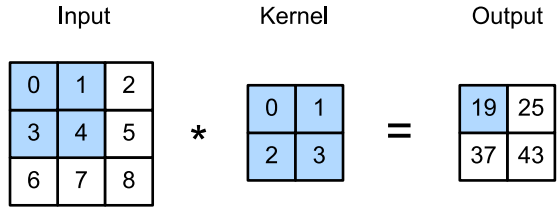
- Cross-correlation operation (used in CNN)

$$Out(x,y) = (K * In)(x,y) = \sum_m \sum_n In(x+m, y+n)K(m,n) \quad (2)$$

Convolution vs. Cross-correlation (cont.)

The following figure illustrates 2D cross-correlation operation.

- Input size is 3×3
- Output size is 2×2
- Kernel has $F = 2, S = 1, P = 0$

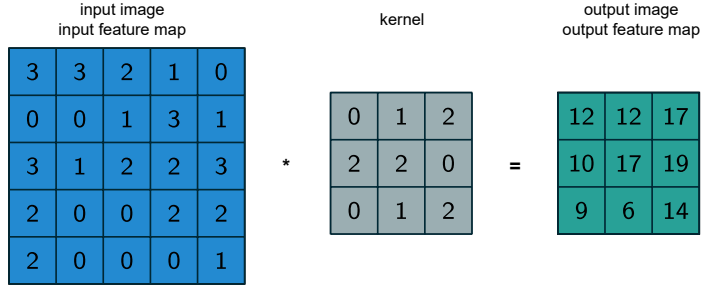


- The shaded portions are the first output element as well as the input and kernel tensor elements used for the output computation

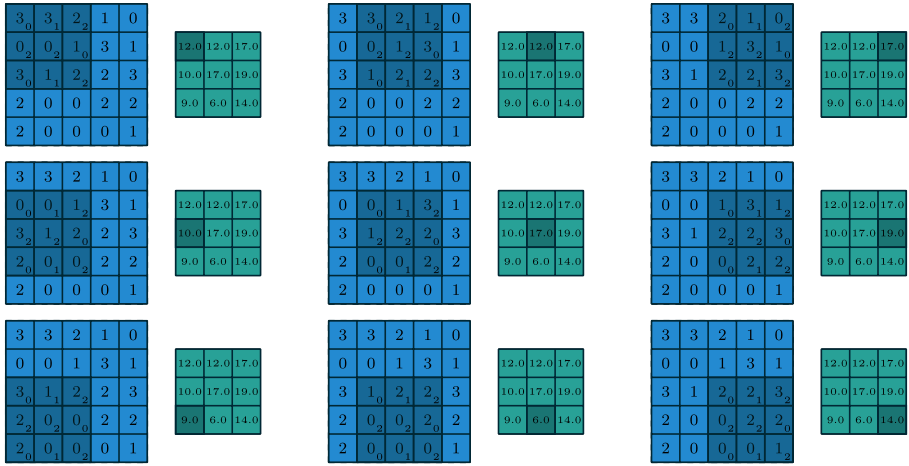
$$0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3 = 19$$

Example 1

- Input image size is 5×5
- Output image size is 3×3
- Kernel has $F = 3, S = 1, P = 0$

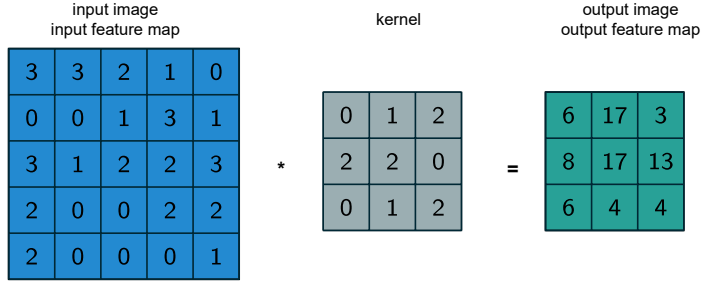


Example 1 (cont.)

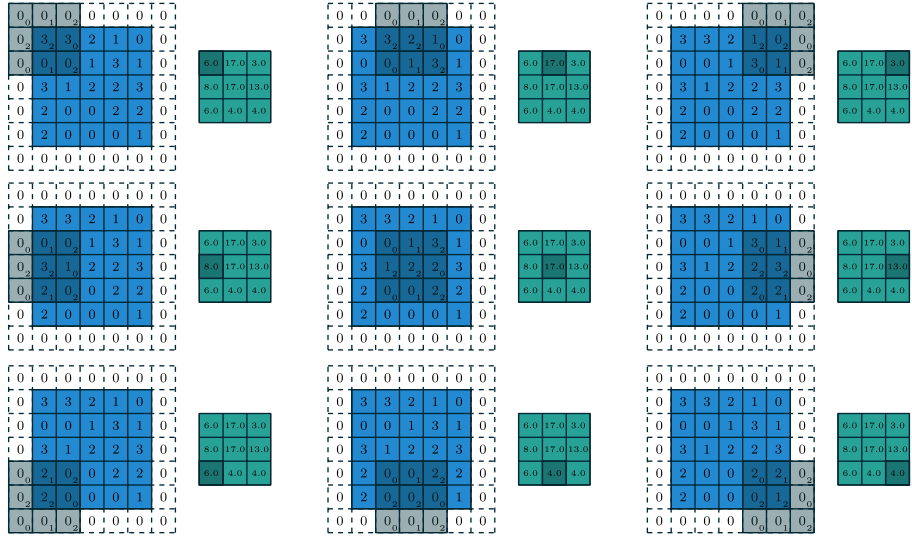


Example 2

- Input image size is 5×5
- Output image size is 3×3
- Kernel has $F = 3, S = 2, P = 1$

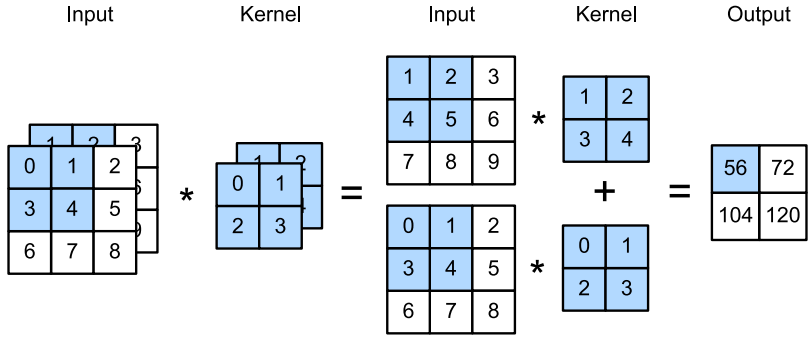


Example 2 (cont.)



Multiple Input Channels

- Convolution computation with 2 input channels.

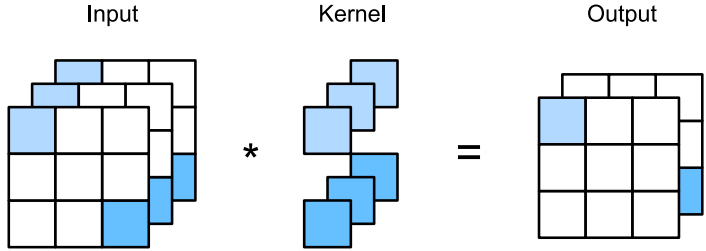


- The shaded portions are the first output element as well as the input and kernel tensor elements used for the output computation

$$(1 \times 1 + 2 \times 2 + 4 \times 3 + 5 \times 4) + (0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3) = 56$$

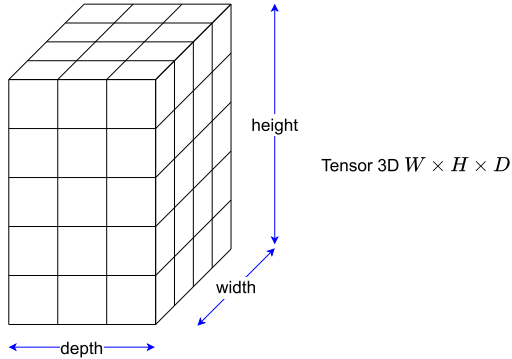
Multiple Input, Output Channels

- Convolution computation for 3 input channels and 2 output channels.



3D tensor

- Convolutions operate over 3D tensors, called *feature maps*, with two spatial axes (*width* W and *height* H) as well as a *depth* axis D (also called the *channels* axis)



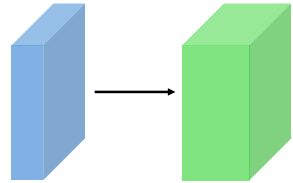
Convolution operator

- Accepts a volume of size $W_{in} \times H_{in} \times D_{in}$ and a group of filters with four hyperparameters:
  - Number of filters K
  - Their spatial extent F
  - The stride S
  - The amount of zero padding P

Convolution operator (cont.)

- Produces a volume of size $W_{out} \times H_{out} \times D_{out}$ where:

$$
\begin{aligned}
W_{out} &= \frac{W_{in} - F + 2P}{S} + 1 \\
H_{out} &= \frac{H_{in} - F + 2P}{S} + 1 \\
D_{out} &= K
\end{aligned}
\tag{3}
$$



Convolution operator (cont.)

In summary,

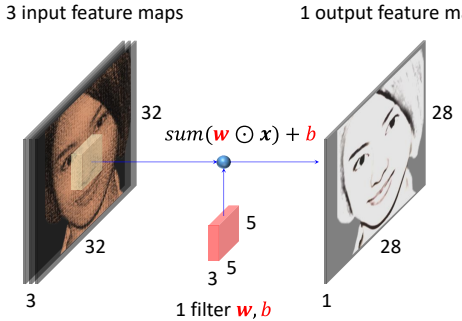
- With parameter sharing, it introduces $F \times F \times D_{in}$ weights per filter, for a total of $F \times F \times D_{in} \times K$ **weights** and K **biases**.
- The filters depth is always equal to the input volume depth.
- The output volume depth is equal to the number of filters
- In the output volume, the d-th depth slice (of size $W_{out} \times H_{out}$ is the result of performing a valid convolution of the d-th filter over the input volume with a stride of S, and then offset by d-th bias.
- Normally, in the same conv layer, all filters have the same dimensions so that special optimized routines can be invoked.

Explanation

- We can imagine a filter as a neuron.
- In a convolution layer we have filters.
  - Each filter is a weight matrix which is convoluted over the input volume.
  - Each time the filter is applied it outputs a single value.
  - The result of convolving the filter over all the input volume (*input feature maps*) is another volume called **activation map** (*output feature map*).
  - A filter's depth is always equal to the input volume depth.
- Instead of having only one filter, we have a stack of filter which produce a stack of activation maps (one for each filter)

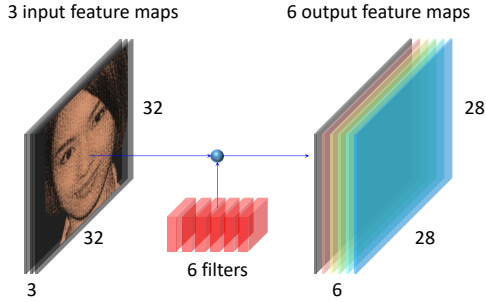
Explanation (cont.)

- The input volume $32 \times 32 \times 3$ is convolved with the filter $5 \times 5 \times 3$ ($S = 1$ and $P = 0$) and the result is the activation map $28 \times 28 \times 1$



Explanation (cont.)

- The input volume $32 \times 32 \times 3$ is convolved with the stack of 6 filter $5 \times 5 \times 3$ and the result is the stack of 6 activation map $28 \times 28 \times 1$ ($28 \times 28 \times 6$)



Pooling layer

Pooling layer

- The reason to use pooling layer (downsampling) is to reduce the number of feature-map coefficients to process
- Pooling layers also provide invariance to small translations of the input
- The most common kind of pooling is *max pooling*
- Another kind of pooling is average pooling; however, it's more informative to look at the *maximal presence* of different features than at their *average presence*

Pooling operator

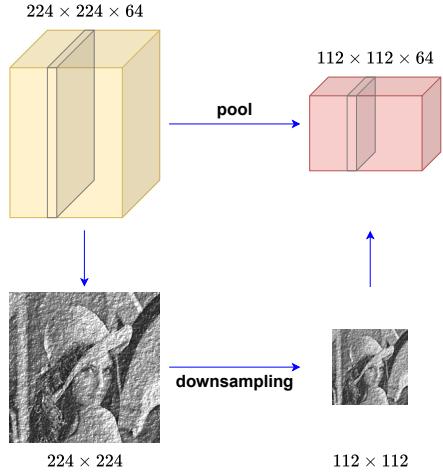
- Accepts a volume of size $W_{in} \times H_{in} \times D_{in}$ and one filter with two hyperparameters:
  - Their spatial extent F
  - The stride S
- Produces a volume of size $W_{out} \times H_{out} \times D_{out}$ where:

$$\begin{aligned} W_{out} &= \frac{W_{in} - F}{S} + 1 \\ H_{out} &= \frac{H_{in} - F}{S} + 1 \\ D_{out} &= D_{in} \end{aligned} \tag{4}$$

- The most common form is the filter of size 2×2 applied with a stride of 2.

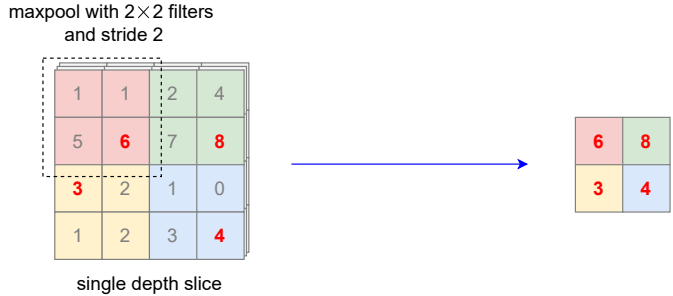
Example

- The input volume of size $[224 \times 224 \times 64]$ is pooled with filter size 2, stride 2 into output volume of size $[112 \times 112 \times 64]$



Example (cont.)

- Max pooling with filter size 2, stride 2



Fully connected layer

Fully connected layer

- Fully connected layers (FC layers), also called *affine layers*, produce the *high-level reasoning* in the DNN.
- Neurons in a fully connected layer have full connections to all activations in the previous layer, as in regular Neural Networks.
- Their activations can hence be computed with a matrix multiplication followed by a bias offset.

VGGNet

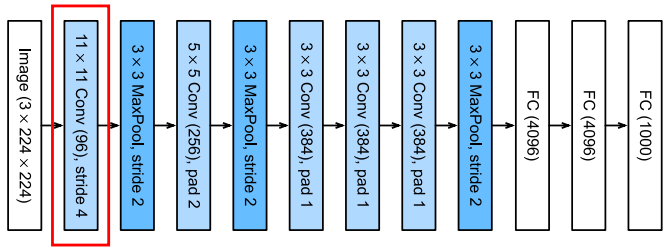
Layer name	Variables		Parameters	
	Layout	Size	Weights	Biases
INPUT	[224x224x3]	224*224*3=150K		
CONV3-64	[224x224x64]	224*224*64=3.2M	(3*3*3)*64=1,728	64
CONV3-64	[224x224x64]	224*224*64=3.2M	(3*3*64)*64=36,864	64
POOL2	[112x112x64]	112*112*64=800K		
CONV3-128	[112x112x128]	112*112*128=1.6M	(3*3*64)*128=73,728	128
CONV3-128	[112x112x128]	112*112*128=1.6M	(3*3*128)*128=147,456	128
POOL2	[56x56x128]	56*56*128=400K		
CONV3-256	[56x56x256]	56*56*256=800K	(3*3*128)*256=294,912	256
CONV3-256	[56x56x256]	56*56*256=800K	(3*3*256)*256=589,824	256
CONV3-256	[56x56x256]	56*56*256=800K	(3*3*256)*256=589,824	256
POOL2	[28x28x256]	28*28*256=200K		
CONV3-512	[28x28x512]	28*28*512=400K	(3*3*256)*512=1,179,648	512
CONV3-512	[28x28x512]	28*28*512=400K	(3*3*512)*512=2,359,296	512
CONV3-512	[28x28x512]	28*28*512=400K	(3*3*512)*512=2,359,296	512
POOL2	[14x14x512]	14*14*512=100K		
CONV3-512	[14x14x512]	14*14*512=100K	(3*3*512)*512=2,359,296	512
CONV3-512	[14x14x512]	14*14*512=100K	(3*3*512)*512=2,359,296	512
CONV3-512	[14x14x512]	14*14*512=100K	(3*3*512)*512=2,359,296	512
POOL2	[7x7x512]	7*7*512=25K		
FC	[1x1x4096]	4096	7*7*512*4096=102,760,448	4096
FC	[1x1x4096]	4096	4096*4096=16,777,216	4096
FC	[1x1x1000]	1000	4096*1000=4,096,000	1000

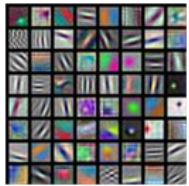
- Total variables (memory): 24M $\times$ 4 bytes $\simeq$ 93MB/image (only forward! $\sim \times 2$ for backward)
- Total parameters: 138M parameters
- The most of the network parameters are in the FC layer and most of the variables (memory) required by the network is used in the first 2 Conv Layers.

Visualizing and Understanding

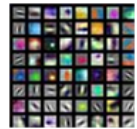
First Layer

- Visualize first-layer filters/kernels (raw weights) directly

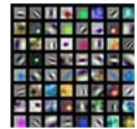




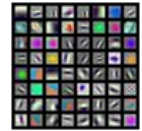
AlexNet:
64 x 3 x 11 x 11



ResNet-18:
64 x 3 x 7 x 7



ResNet-101:
64 x 3 x 7 x 7



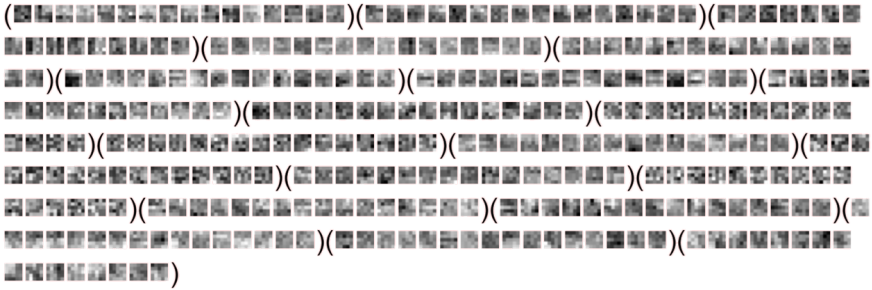
DenseNet-121:
64 x 3 x 7 x 7

First Layer (cont.)

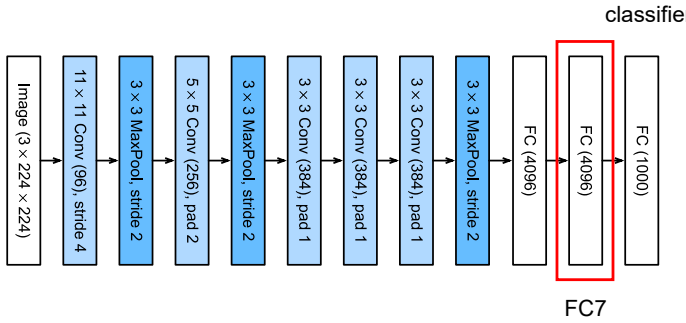
- We can visualize filters at higher layers, but not that interesting (these pictures are taken from ConvNetJS CIFAR-10 demo)

Layer 1 filters

Layer 2 filters

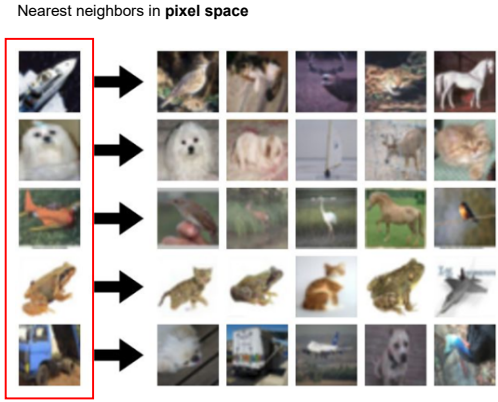


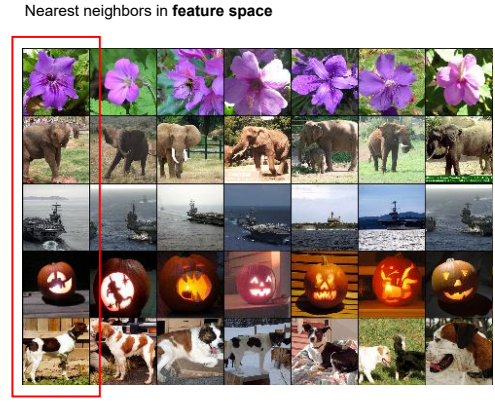
Last Layer



- FC7: 4096-dimensional feature vector for an image (layer immediately before the classifier)
- Run the network on many images, collect the feature vectors

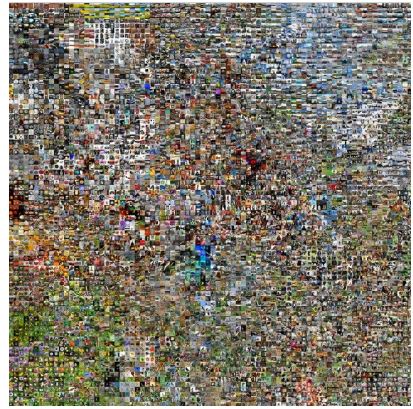
Last Layer: Nearest Neighbors





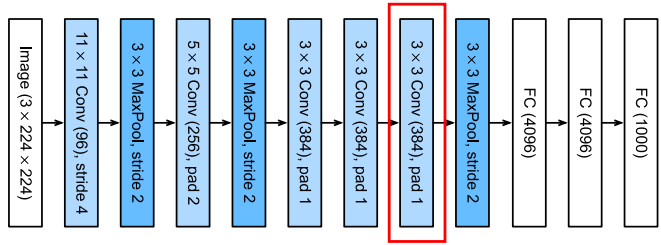
Last Layer: Dimensionality Reduction

- Visualize the "space" of FC7 feature vectors by reducing dimensionality of vectors from 4096 to 2 dimensions
  - Simple algorithm: Principal Component Analysis (PCA)
  - More complex: t-SNE



Visualizing Activations

- A view of the 13×13 activation of the 151^{st} channel on the conv5 layer



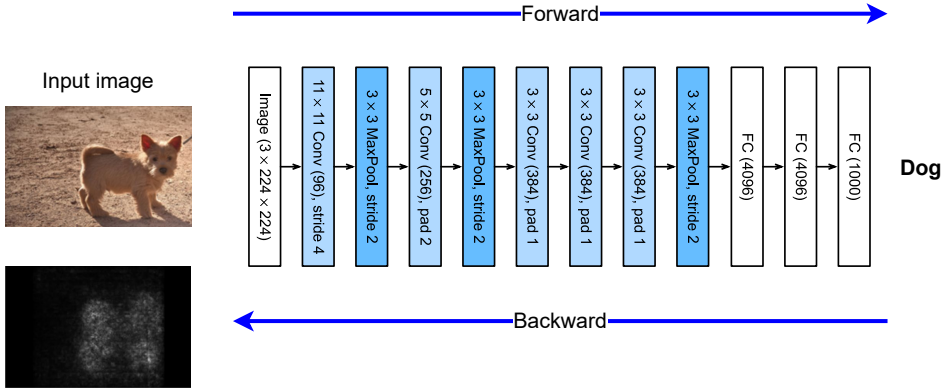
Mapping activations back to pixels

(Guided) backpropagation method

- Forward an image through the network
- Choose a feature map and an activation
- Zero out all values except for the one of interest
- Propagate that value back to the image

Which pixels matter: Saliency

- Compute gradient of (unnormalized) class score with respect to image pixels,
- Take absolute value and max over RGB channels



Which pixels matter: Saliency (cont.)

- Superimposing the class activation heatmap on the original picture

Generate Images: Gradient Ascent

(Guided) backpropagation
Find the part of an image that a neuron responds to

Gradient ascent
Generate a synthetic image that maximally activates a neuron

$$I_{syn} = \arg\max_I S_c(I) - \lambda \|I\|_2^2 \tag{5}$$

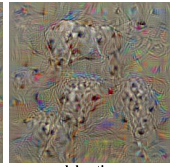
where S_c score for class c (before Softmax)

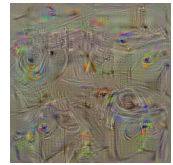
1. Initialize image I to zeros
2. Repeat:
   - 2.1 Forward image to compute current scores
   - 2.2 Backprop to get gradient of neuron value with respect to image pixels
   - 2.3 Make a small update to the image

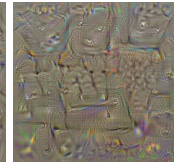
Generate Images: Gradient Ascent

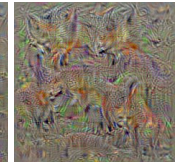

dumbbell


cup

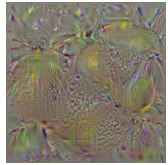

dalmatian

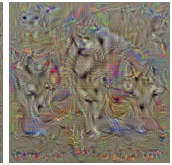

washing machine


computer keyboard


kit fox

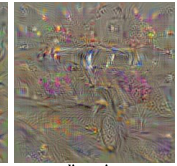

bell pepper


lemon


husky


goose


ostrich


limousine

Network Design

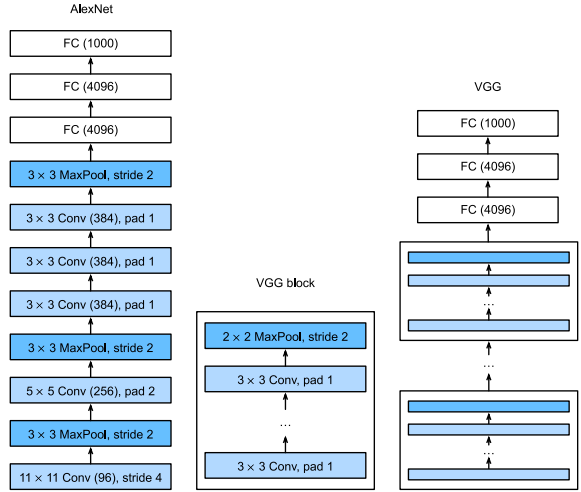
- Networks Using Blocks
- Networks with Parallel Concatenations
- Residual Networks

Blocks

- The design of neural network architectures had grown progressively more abstract, moving from thinking in terms of individual neurons to whole layers, and now to blocks, repeating patterns of layers.
- The basic building block of classic CNNs is a sequence of the following:
  - a convolutional layer with padding to maintain the resolution
  - a nonlinearity such as a ReLU
  - a pooling layer such as a max pooling layer

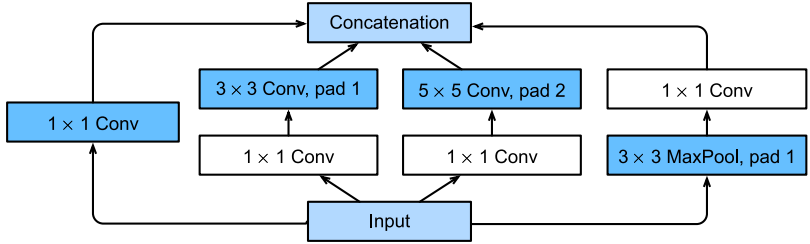
VGG Network

- From AlexNet to VGG that is designed from building blocks.



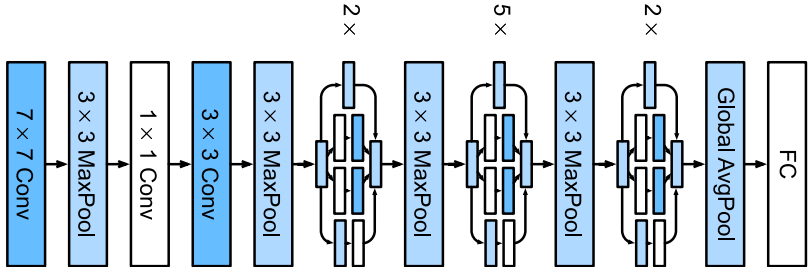
Inception Blocks

- Inception block employs a combination of variously-sized kernels



GoogleNet

- GoogLeNet uses a stack of a total of 9 inception blocks and global average pooling to generate its estimates.



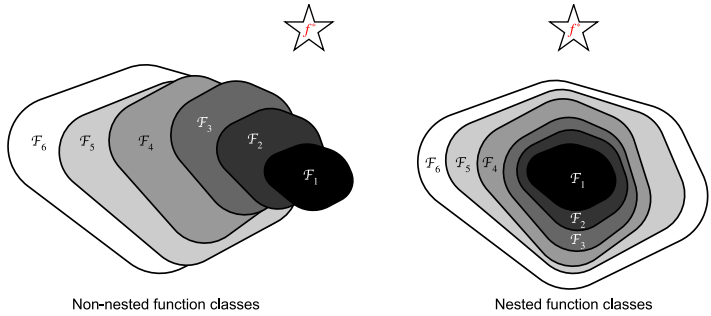
Learning Model

- Consider $\mathcal{F}$, **the class of functions** (hypothesis set) that a specific network architecture can reach.
- Given a data set $\mathcal{D}$, algorithm A finds the best estimated $\hat{f}$ of the "truth" function f^*

$$\hat{f} = \arg\min_f L(f \mid \mathcal{D}) \text{ subject to } f \in \mathcal{F}. \tag{6}$$

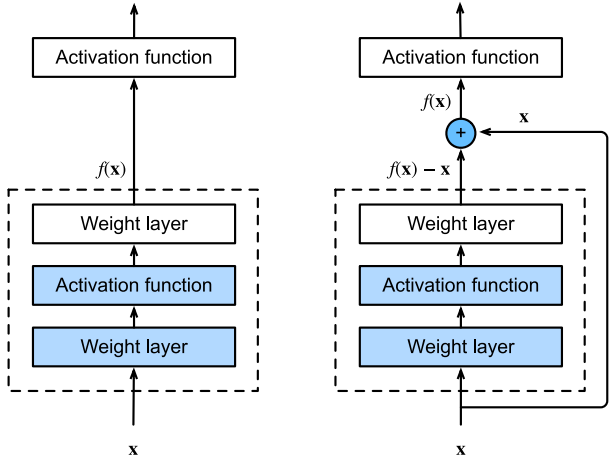
Learning Model (cont.)

- For non-nested function classes, a larger (indicated by area) function class does not guarantee to get closer to the "truth" function f^*. This does not happen in nested function classes.



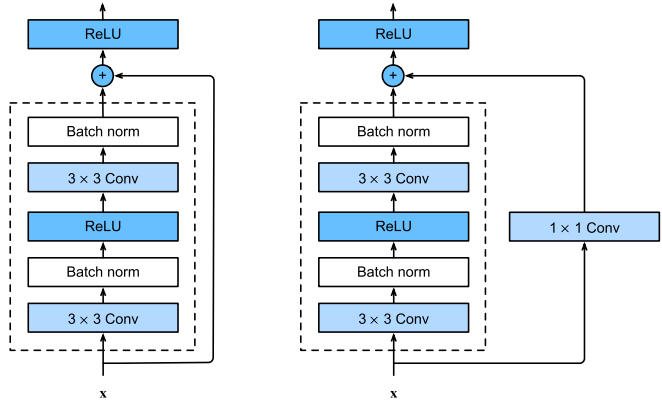
Residual Blocks

- A regular block (left) and a residual block (right).



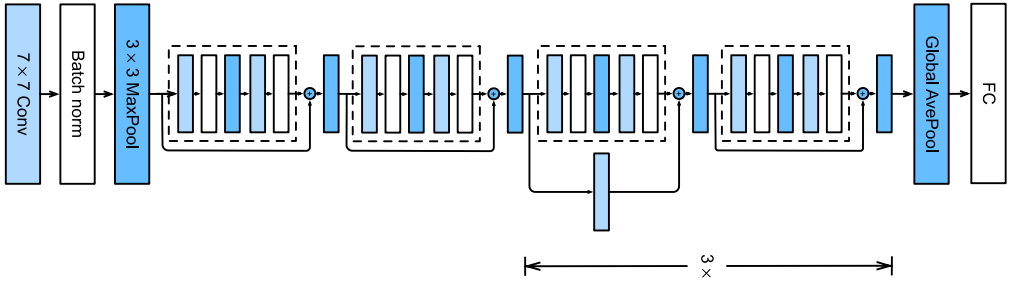
Residual Blocks (cont.)

- ResNet block with and without 1×1 convolution.



ResNet Model

- The ResNet-18 architecture



Miscellaneous convolutions

- Transposed convolution
- Dilated convolutions

Transposed convolution

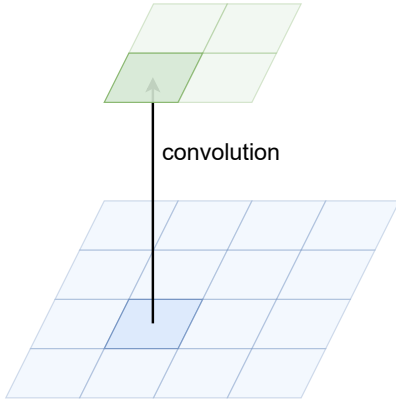
- Convolution can be interpreted as a matrix multiplication

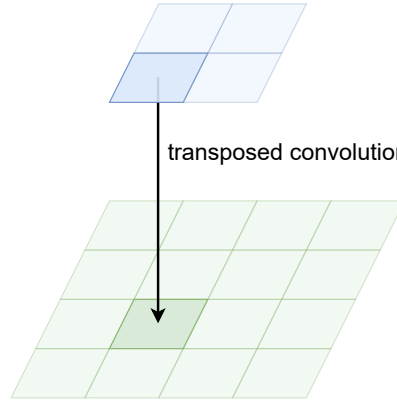
$$\text{kernel } w \rightarrow \text{matrix } C$$

- The need for **transposed convolutions** (often called **deconvolution**) generally arises from the desire to use a transformation going in the opposite direction of a normal convolution

$$\text{convolution } C \longleftrightarrow \text{transposed convolution } C^{\mathsf{T}}$$

Transposed convolution (cont.)

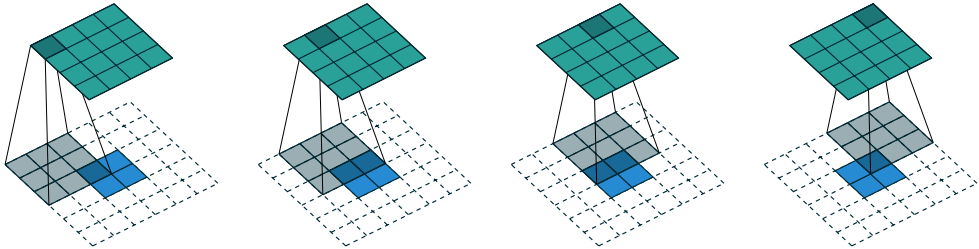




Transposed Convolution (cont.)

- No zero padding, unit strides, transposed

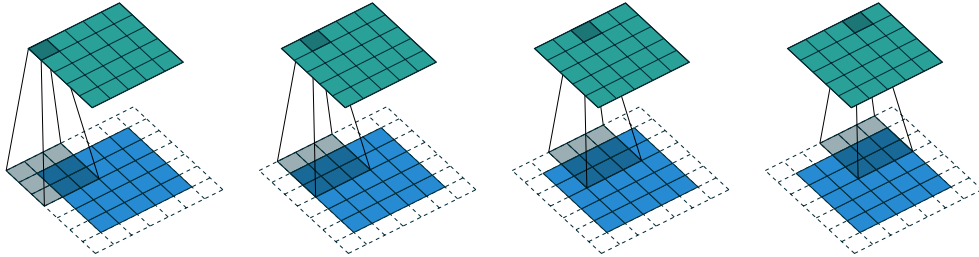
	convolution	transposed convolution
kernel size	k	$k' = k$
stride	$s = 1$	$s' = 1$
padding	$p = 0$	$p' = k - 1$
output size		$o' = i' + (k - 1)$



Transposed Convolution (cont.)

- Zero padding, unit strides, transposed

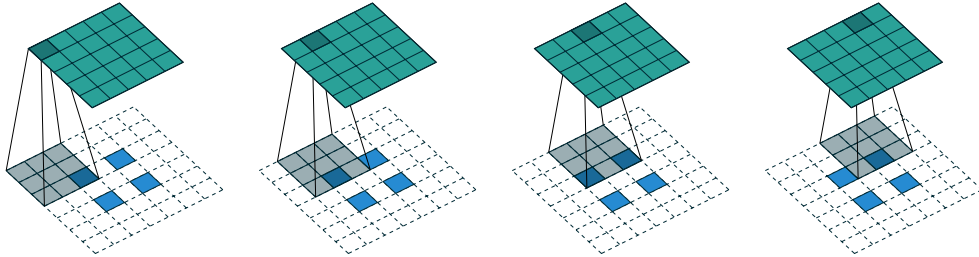
	convolution	transposed convolution
kernel size	k	$k' = k$
stride	$s = 1$	$s' = 1$
padding	p	$p' = k - p - 1$
output size		$o' = i' + (k - 1) - 2p$



Transposed Convolution (cont.)

- No zero padding, non-unit strides, transposed:
  - adding $s - 1$ zeros between each input unit

	convolution	transposed convolution
kernel size	k	$k' = k$
stride	s	$s' = 1$
padding	$p = 0$	$p' = k - 1$
output size		$o' = s(i' - 1) + k$



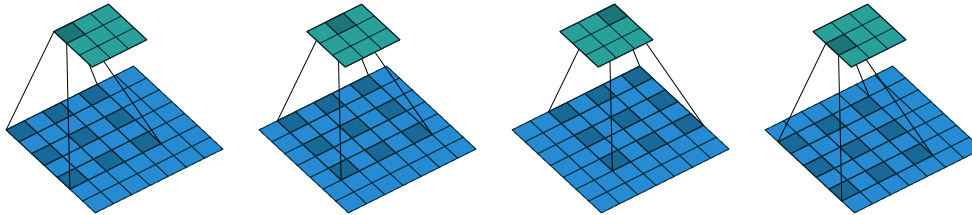
Dilated convolutions

Concept 1

Dilated convolutions "inflate" the kernel by inserting spaces between the kernel elements. The dilation "rate" is controlled by an additional hyperparameter d.

- For any i, k, s and p, and for a dilation rate d,

$$o = \left\lfloor \frac{i + 2p - k - (k-1)(d-1)}{s} \right\rfloor + 1. \tag{7}$$

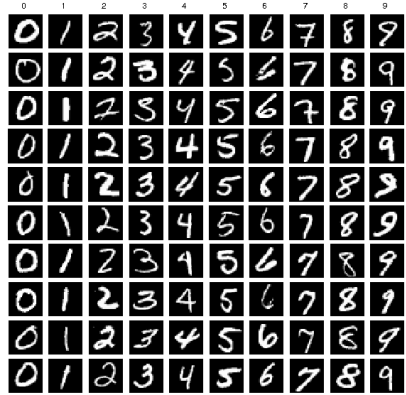


Applications

- Datasets
- Applications

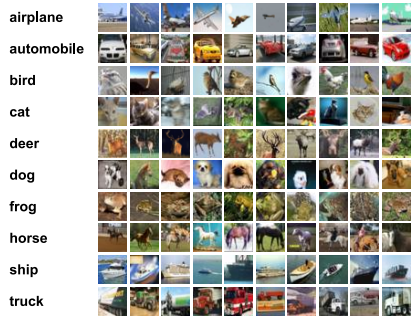
MNIST

- The MNIST database is a large database of handwritten digits that is commonly used for training various image processing systems
- The MNIST database contains 60,000 training images and 10,000 testing images



CIFAR

- The CIFAR-10 and CIFAR-100 are labeled subsets of the 80 million tiny images dataset. They were collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton.
- The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.



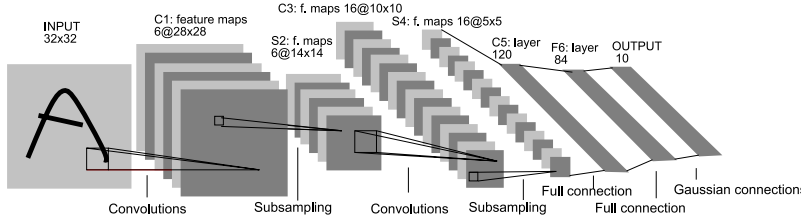
ImageNet

- The ImageNet project is a large visual database designed for use in visual object recognition software research

Image classifcation

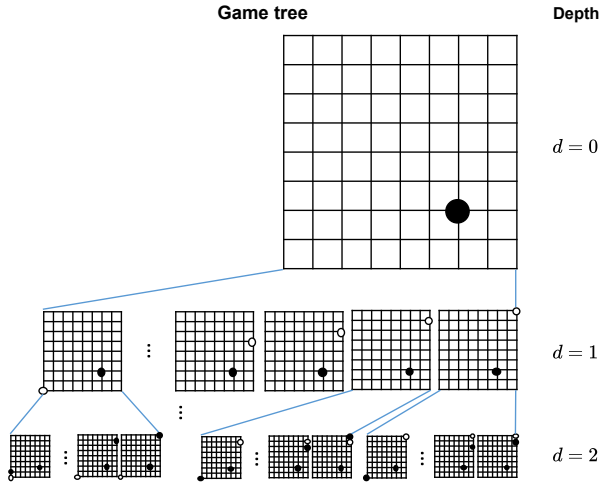
- Architecture of LeNet-5 a Convolutional Neural Network, here for digit recognition, but is extended to character recognition.

:

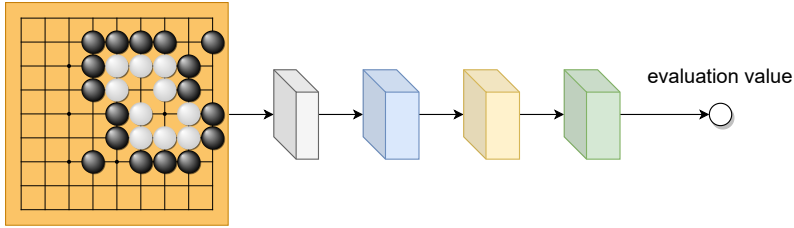


Games

- Go game



Games (cont.)



References

Goodfellow, I., Bengio, Y., and Courville, A. (2016).
Deep learning.
MIT press.

Lê, B. and Tô, V. (2014).
Cở sở trí tuệ nhân tạo.
Nhà xuất bản Khoa học và Kỹ thuật.

Russell, S. and Norvig, P. (2021).
Artificial intelligence: a modern approach.
Pearson Education Limited.