ECBM E4040 Neural Networks and Deep Learning

Lecture: Convolutional Neural Networks (CNNs)

Instructor Zoran Kostic

Columbia University
Department of Electrical Engineering

October 22, 2016

Part I

Review of Previous Lecture - Optimization

Previous Lecture - Topics Covered

Optimization - Topics Covered:

- Optimization for Model Training
- Challenges in Neural Network Optimization
- Basic Learning Algorithms
- Algorithms for Adaptive Learning Rates

Previous Lecture - Learning Objectives

Optimization - Learning Objectives:

- Major challenges: ill-conditioning, local minima, saddle points, just to name a few.
- Stochastic gradient descent and variants are the algorithms of choice in neural networks.
- The choice of which algorithm to use depends as much on the users familiarity with the algorithm as it does on any established notion of superior performance.

Part II

Convolutional Networks

Convolutional Neural Networks (CNNs) - Topics

Convolutional Neural Networks - Topics:

- **The Convolution Operation**
- **Motivation**
- **Pooling**

Convolutional Networks - Learning Objectives

Convolutional Neural Networks - Learning Objectives:

- **Convolution leverages three important concepts that substantially improve a machine learning system: sparse connectivity, parameter sharing and equivariant representations.**
- **Convolution provides a means for working with inputs of variable size.**

The Convolution Operation

The convolution in a simplified form is defined by

$$s(t) = \int_{\mathbb{D}} x(a)w(t-a)da$$

and it is usually denoted by

$$s(t) = (x * w)(t).$$

Here x denotes the input, w denotes the kernel and s is the output, also referred to as the feature map. In discrete time notation

$$s[t] = (x * w)[t] = \sum_{a \in \mathbb{N}} x[a]w[t-a].$$

The Convolution Operation (cont'd)

The input is usually a multidimensional array of data and the kernel is usually a multidimensional array of learn-able parameters referred to as tensors. If the input is is a two-dimensional image I and K is a kernel is a two-dimensional kernel

$$s[i,j] = (I * K)[i,j] = \sum_m \sum_n I[m,n]K[i-m, j-n].$$

The convolution operation is commutative, that is,

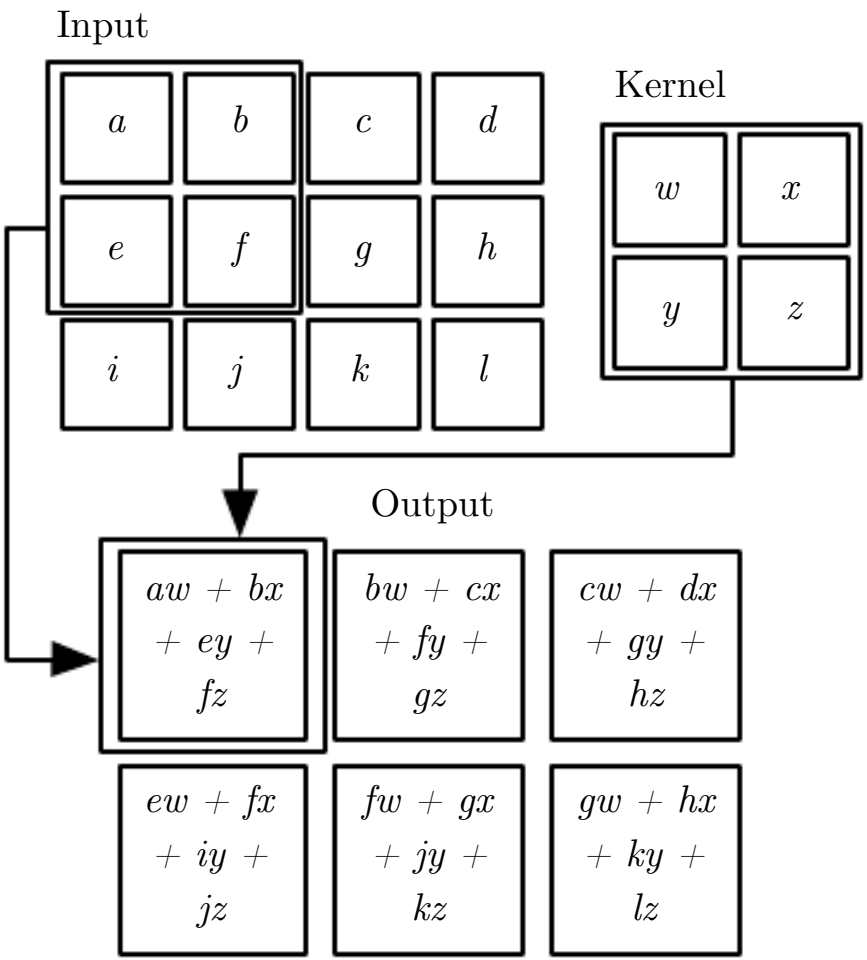
$$s[i,j] = (I * K)[i,j] = \sum_m \sum_n I[i-m, j-n]K[m,n].$$

Many neural network libraries implement the cross-correlation defined by (same as the convolution but without flipping the kernel)

$$s[i,j] = (I * K)[i,j] = \sum_m \sum_n I[i+m, j+n]K[m,n].$$

The Convolution Operation (cont'd)

An Example



2-D convolution without kernel flipping.

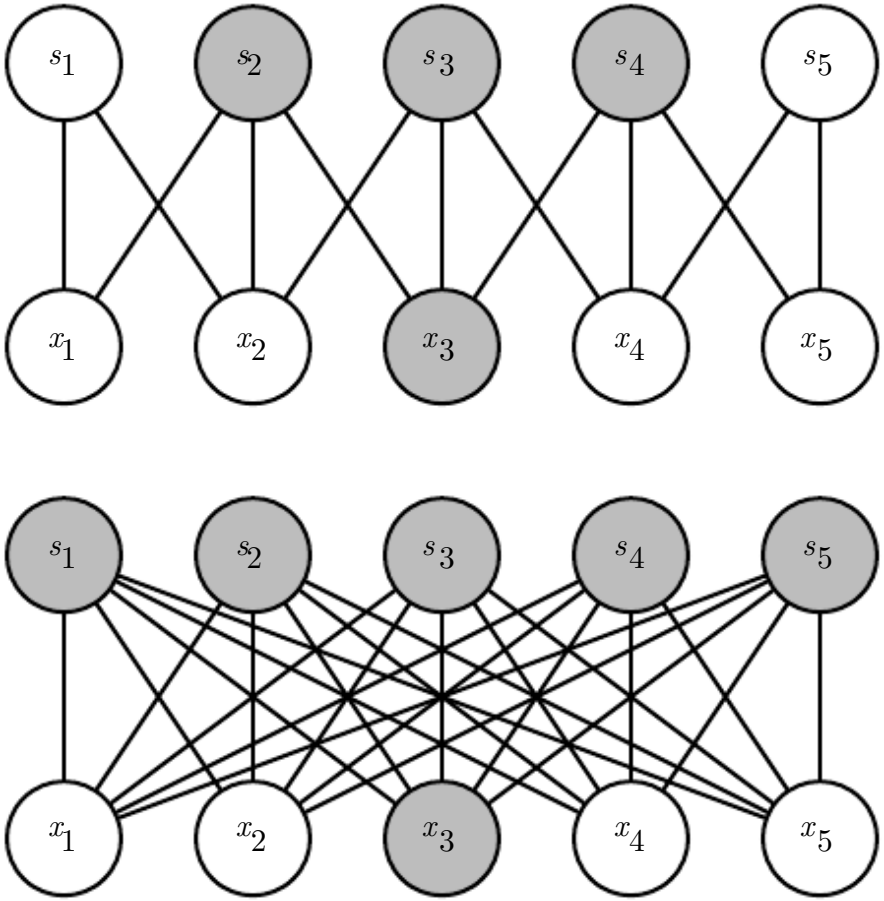
Motivation for Using Convolution

Convolution leverages three important concepts that can help improve a machine learning system:

- sparse connectivity,
- parameter sharing,
- equivariant representations.

Moreover, convolution provides a means for working with inputs of variable size.

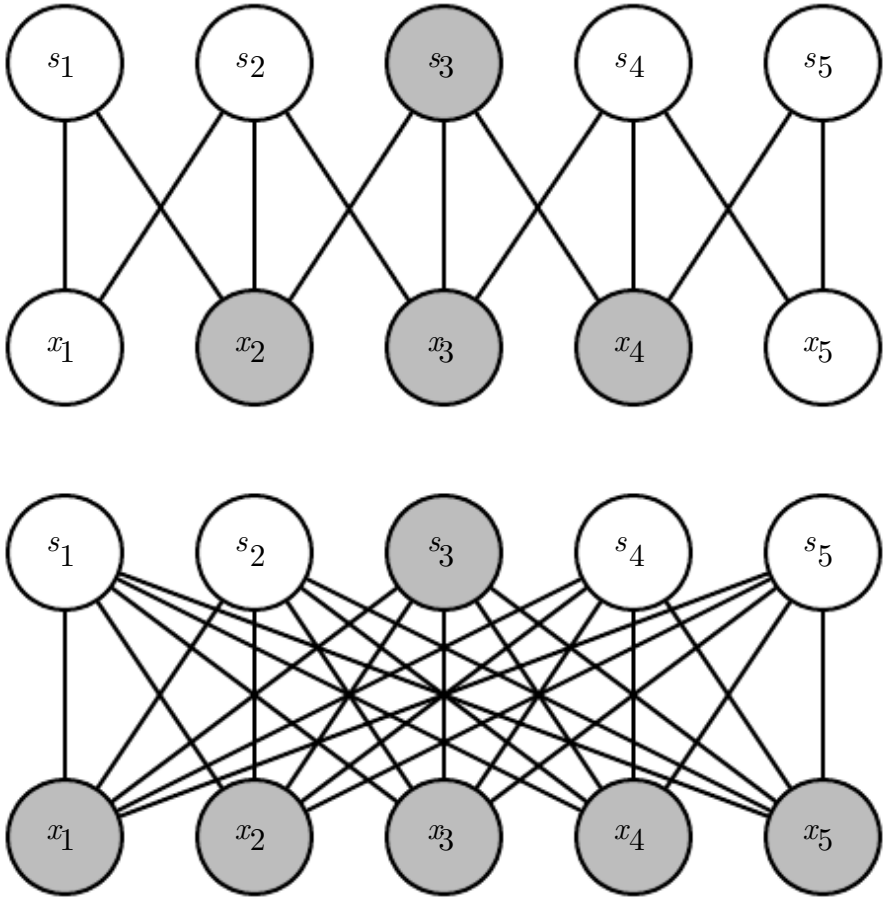
Convolutional Networks Have Sparse Connectivity



Matrix multiplication (bottom) with full connectivity. Convolution with a kernel highlights sparse connectivity.

Convolutional Networks Have Sparse Connectivity (cont'd)

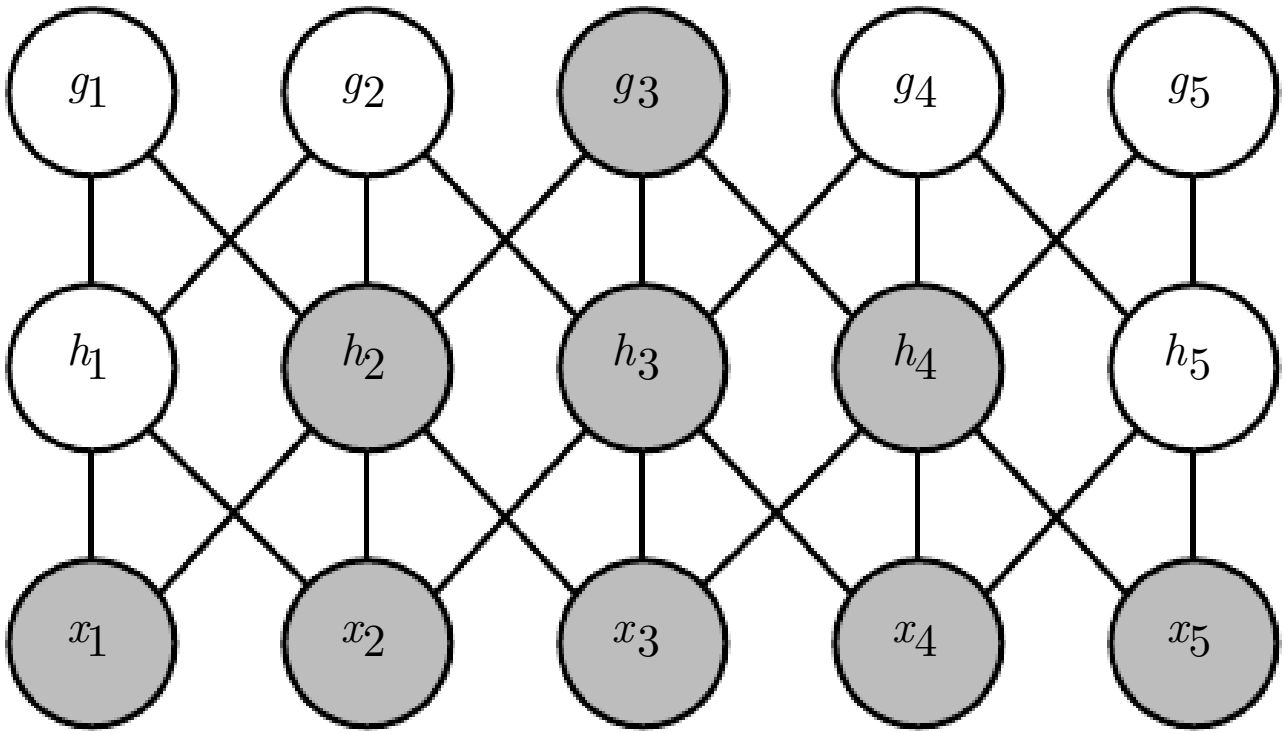
Characterization via Receptive Fields



Top: s_3 receives 3 inputs. Bottom: s_3 receives full input.

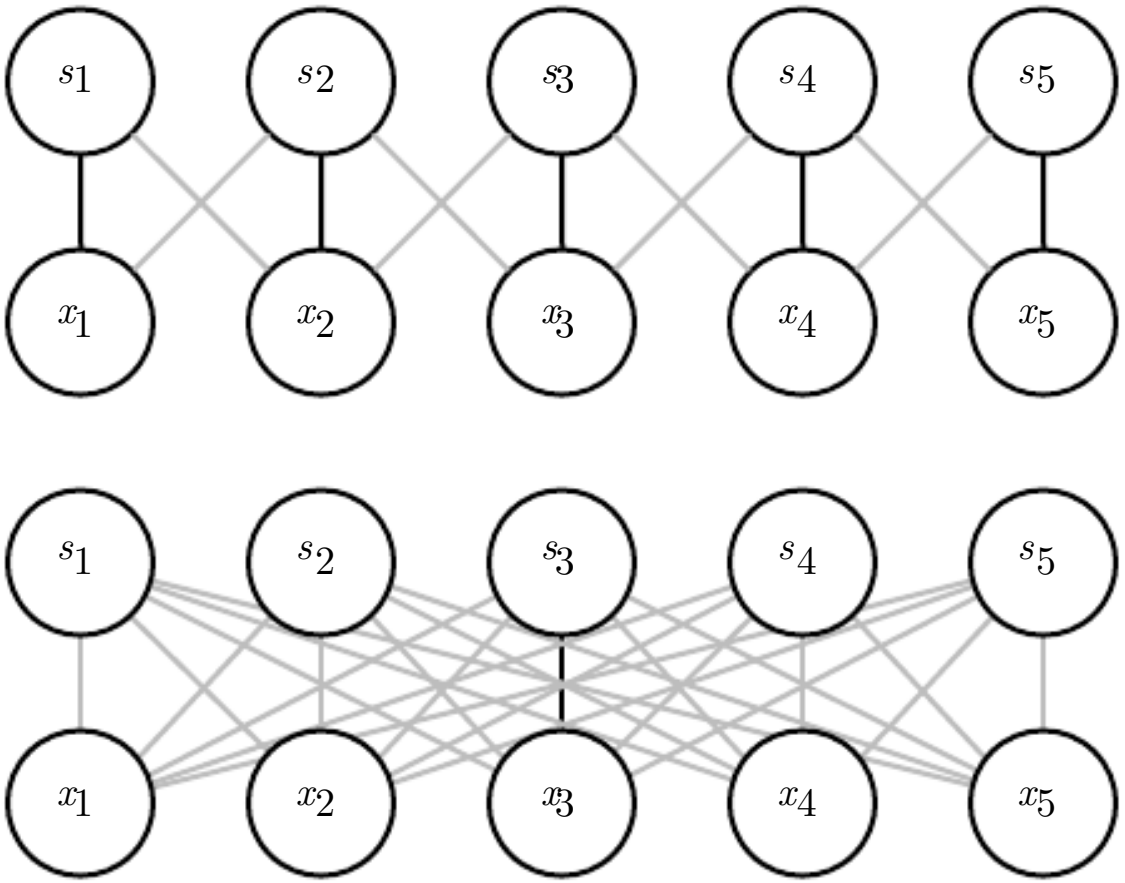
Convolutional Networks Have Sparse Connectivity (cont'd)

Complex Interactions from Simple Building Blocks



The receptive field of the units in the deeper layers of a convnet is larger than the receptive field of the units in the shallow layers.

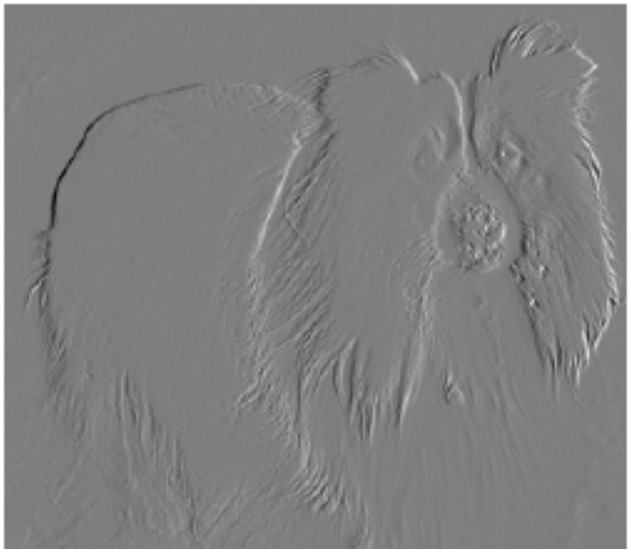
Parameter Sharing



In a convnet, each member of the kernel is used at every position of the input.

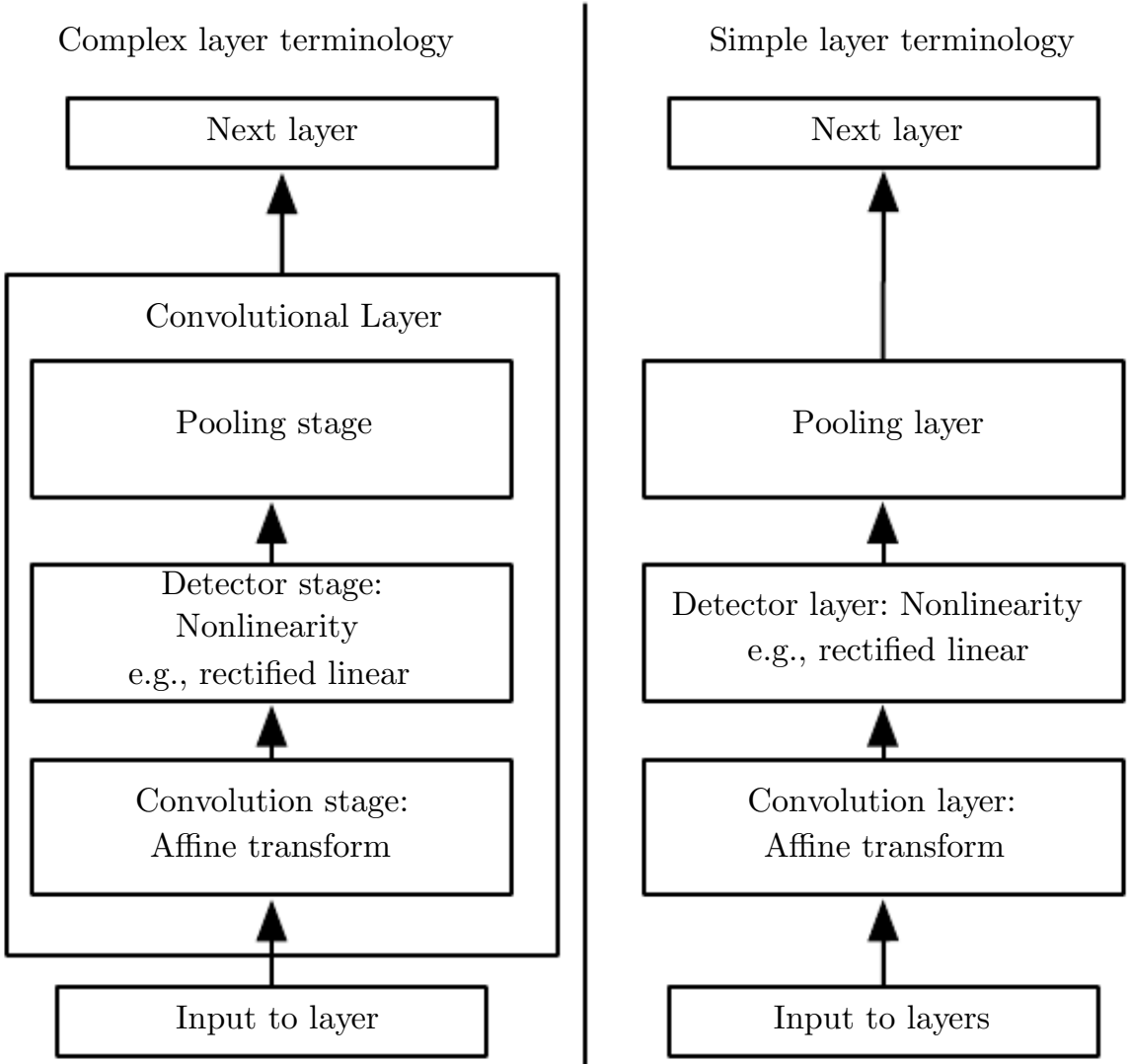
Parameter Sharing (cont'd)

Efficient Edge Detection



The input image has 280×320 pixels. The convolution kernel has 2 (two) elements and requires $319 \times 280 \times 3 = 267,960$ floating point operations (two multiplications and one addition per output pixel) to compute. Matrix multiplication of the same operation of edge detection requires $320 \times 280 \times 319 \times 280$, or over 8 billion matrix entries, making convolution 4 billion times more efficient.

The Convolutional Network Layer



Pooling

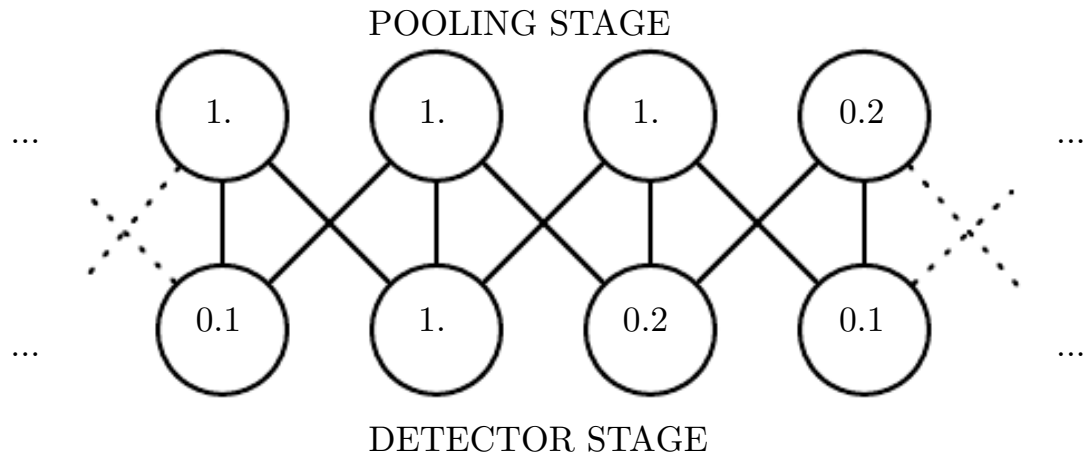
A pooling function replaces the output of the net at a certain location with a summary statistic of the nearby outputs. For example,

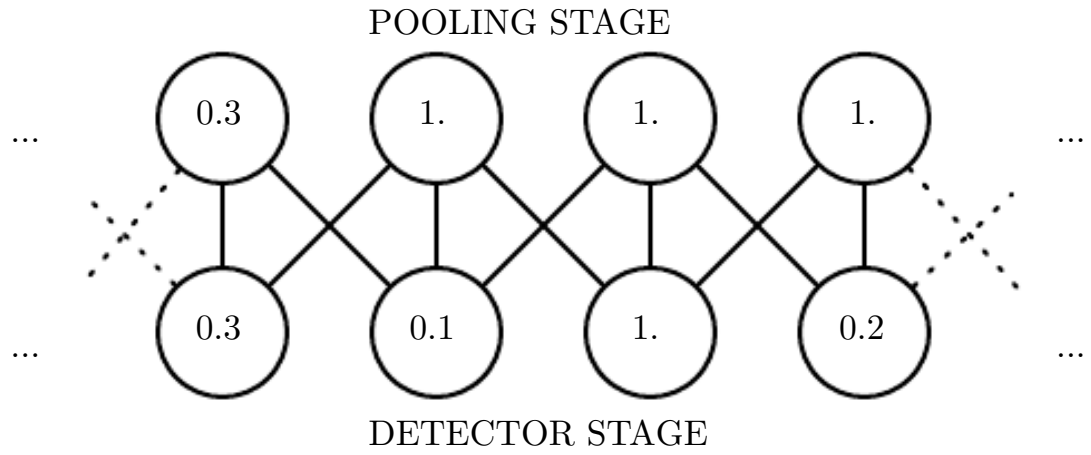
- max pooling reports the maximum output within a rectangular neighborhood;
- the average of a rectangular neighborhood;
- the L2 norm of a rectangular neighborhood;
- a weighted average based on the distance from the central pixel.

Pooling helps to make the representation become invariant to small translations of the input.
Invariance to local translation can be a very useful property if we care more about whether some feature is present than exactly where it is.

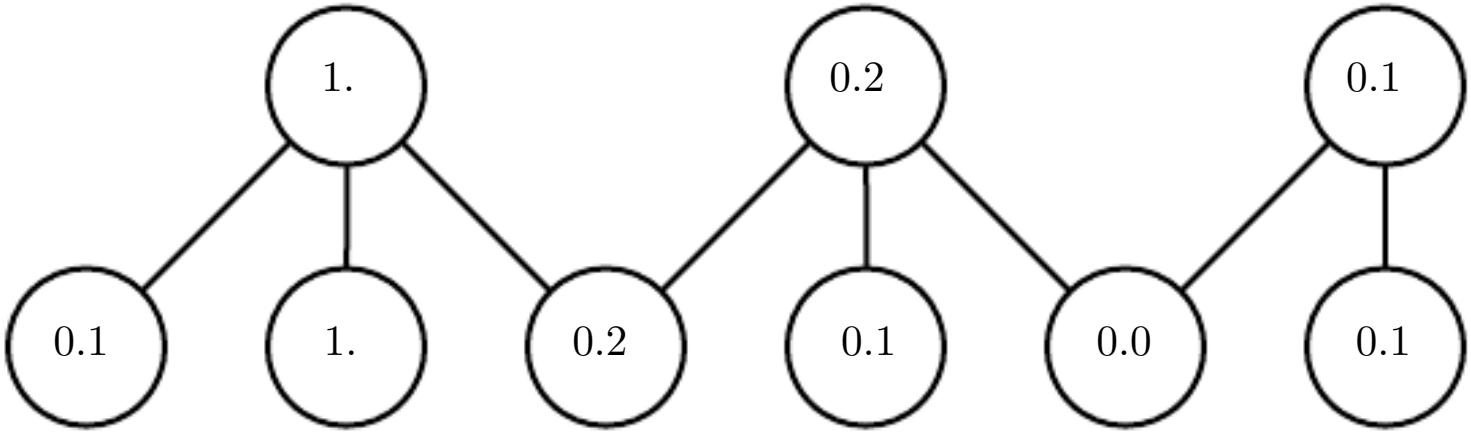
Pooling (cont'd)

Max Pooling Introduces Invariance with Respect to Small Translations





Pooling with Downsampling



max-pooling with a pool width of 3 and a stride between pools of 2 reduces the representation size by a factor of 2. The computational and statistical burden on the next layer is considerably reduced.

Part III

Convolutional Networks (cont'd)

Convolutional Neural Networks (CNNs) - Topics

Convolutional Neural Networks - Topics:

- **Basic Convolution Operation**
- **Variants of the Basic Convolution Operation**
- **The Brain as an Information Processor**

Basic Convolution Function

Convolution neural networks consist of many applications of convolution in parallel. Convolution with a single kernel extracts a single feature at many locations. The input is typically a grid of vector-valued observations.

In a multilayer convolutional network, the input to the second layer is the output of the first layer, which usually has the output of many different convolutions at each position.

When working with images, we usually think of the input and output of the convolution as being 3-D tensors, with one index for the different channels and two indices for the spatial coordinates of each channel.

Basic Convolution Function (cont'd)

Assume we have a 4-D kernel tensor K with element K_{ijkl} giving the connection strength between a unit in channel i of the output and a unit in channel j of the input, with an offset of k rows and l columns between the output unit and the input unit.

Assume our input consists of observed data V with element V_{ijk} giving the value of the input unit within channel i at row j and column k. Assume our output consists of Z with the same format as V. If Z is produced by convolving K across V without flippping K, then

$$Z_{ijk} = \sum_{l,m,n} V_{l,j+m-1,k+n-1} K_{ilmn},$$

where the summation over l, m and n is over all values for which the tensor indexing operations inside the summation is valid.

Variants of the Basic Convolution Function

Downsampling

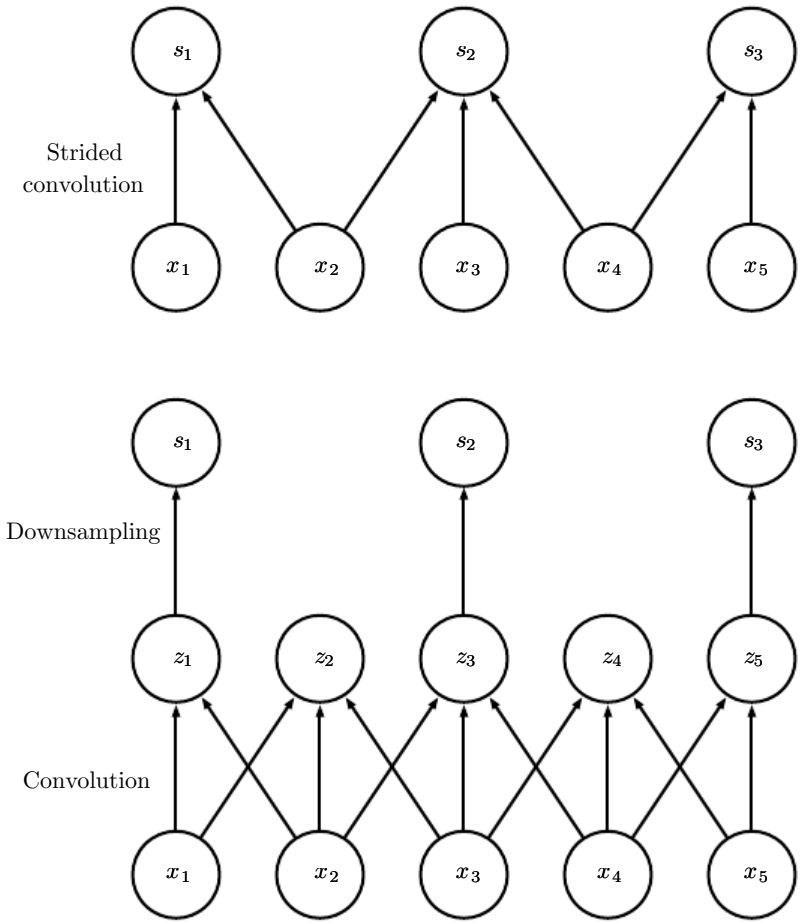
We may want to skip over some positions of the kernel in order to reduce the computational cost (at the expense of not extracting our features as finely). We can think of this as downsampling the output of the full convolution function. If we want to sample only every s pixels in each direction in the output, then we can define a downsampled convolution function c such that

$$Z_{ijk} = c(K, V, s)_{ijk} = \sum_{l,m,n} [V_{l,(j-1)s+m,(k-1)s+n} K_{ilmn}].$$

We refer to s as the stride of this downsampled convolution. It is also possible to define a separate stride for each direction of motion.

Variants of the Basic Convolution Function

Example: Convolution with a Stride



(Top) Convolution with a stride length of two implemented in a single operation.
(Bottom) Convolution with a stride greater than one pixel is mathematically equivalent to convolution with unit stride followed by downsampling.
The two-step approach involving downsampling is computationally wasteful, because it computes many values that are then simply discarded.

Variants of the Basic Convolution Function

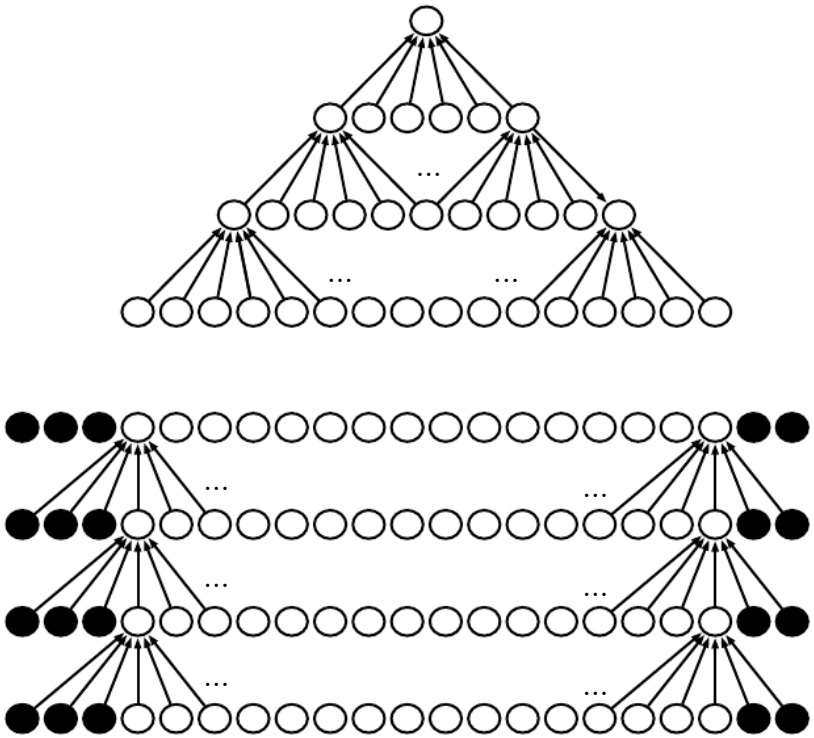
Zero-Padding

One essential feature of any convolutional network implementation is the ability to implicitly zero-pad the input V in order to make it wider. Zero padding the input allows us to control the kernel width and the size of the output independently.
Three special cases of the zero-padding setting are worth mentioning:

- the extreme case in which no zero-padding is used whatsoever, and the convolution kernel is only allowed to visit positions where the kernel is contained entirely within the image.
- the case when just enough zero-padding is added to keep the size of the output equal to the size of the input.
- the case in which enough zeroes are added for every pixel to be visited k times in each direction, resulting in an output image of width $m + k - 1$.

Variants of the Basic Convolution Function

Example: The Effect of Zero Padding on Network Size



(Top) No implicit zero padding. This causes the representation to shrink by five pixels at each layer. Starting from an input of sixteen pixels, there are only three convolutional layers available. (Bottom) By adding five implicit zeroes to each layer, we prevent the representation from shrinking with depth. This allows us to make an arbitrarily deep convolutional network.

Variants of the Basic Convolution Function

Unshared Convolution

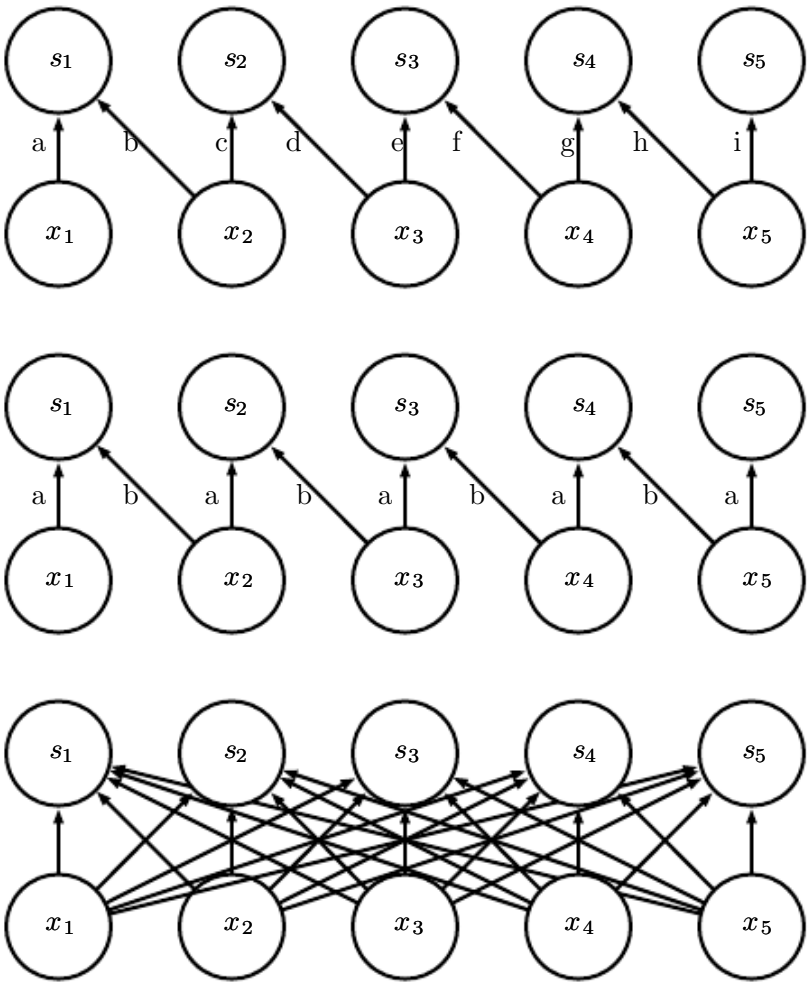
In some cases, we do not actually want to use convolution, but rather locally connected layers. In this case, the adjacency matrix in the graph of our MLP is the same, but every connection has its own weight, specified by a 6-D tensor W.
The indices into W are respectively: i, the output channel, j, the output row, k, the output column, l, the input channel, m, the row offset within the input, and n, the column offset within the input. The linear part of a locally connected layer is then given by

$$Z_{ijk} = \sum_{l,m,n} [V_{l,j+m-1,k+n-1} w_{ijklmn}].$$

This is sometimes also called unshared convolution, because it is a similar operation to discrete convolution with a small kernel, but without sharing parameters across locations.

Variants of the Basic Convolution Function

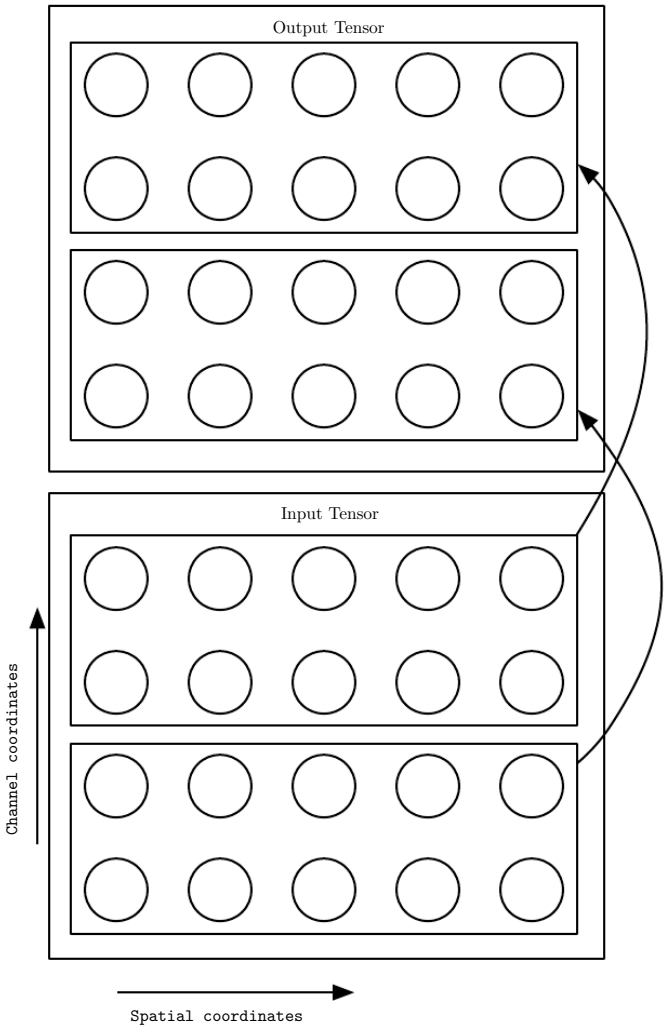
Example: Unshared Convolution



(Top) A locally connected layer with a patch size of two pixels. (Center) A convolutional layer with a kernel width of two pixels. The locally connected layer has no parameter sharing. The conv layer uses the same two weights repeatedly across the entire input, as indicated by the repetition of the letters labeling each edge. (Bottom) A fully connected layer resembles a locally connected layer in the sense that each edge has its own parameter. However, it does not have the restricted connectivity of the locally connected layer

Variants of the Basic Convolution Function

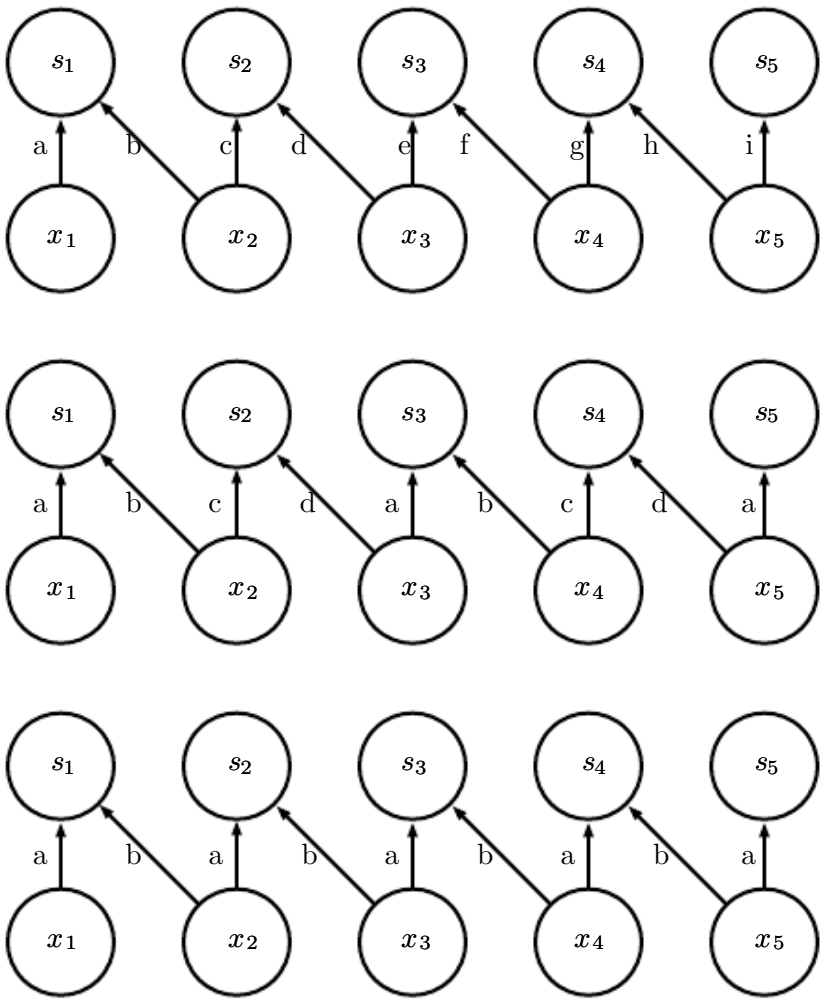
Example: Restricting Connectivity



A convolutional network with the first two output channels connected to only the first two input channels, and the second two output channels connected to only the second two input channels.

Variants of the Basic Convolution Function

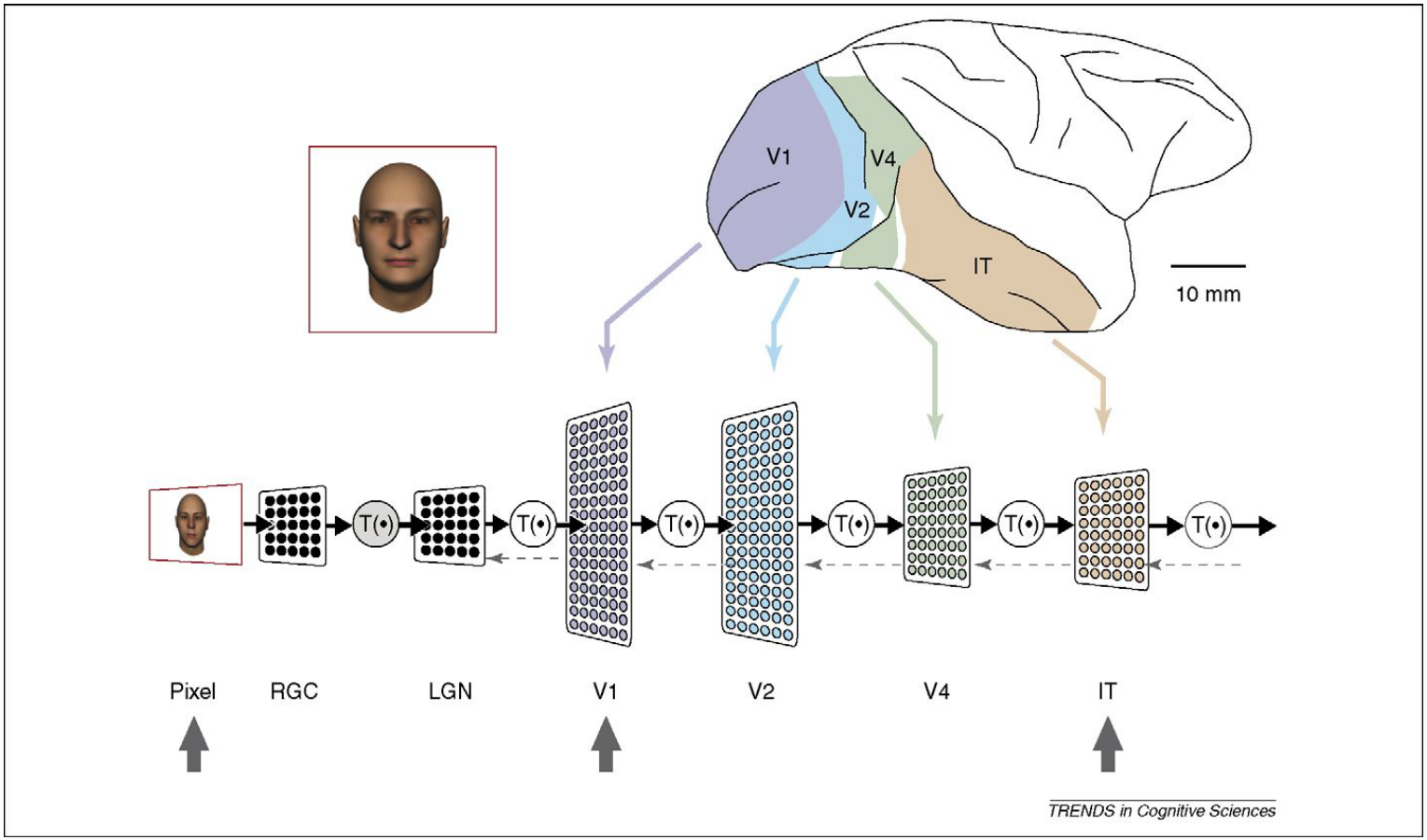
Example: Tiled Convolution



(Top) A locally connected layer has no sharing at all. Each connection has its own weight by labeling each connection with a unique letter. (Center) Tiled convolution with a set of $t = 2$ different kernels. One of these kernels has edges labeled a and b, while the other has edges labeled c and d. If two output units are separated by a multiple of t steps, then they share parameters. (Bottom) Traditional convolution is equivalent to tiled convolution with $t = 1$. There is only one kernel and it is applied everywhere with weights labeled a and b everywhere.

The Brain as an Information Processor

The Basic Building Blocks of Visual Information Processing



RGC: Retinal Ganglion Cells; LGN: Lateral Geniculate Nucleus; Primary Visual Cortex: V1; Visual Areas V2 and V4; IT: Inferior Temporal

Basics of the Early Visual System

Neurophysiologists David Hubel and Torsten Wiesel collaborated for several years in the late 50s and early 60s to determine many of the most basic facts about how the mammalian early visual system works.

Images are formed by light arriving in the eye and stimulating the retina, the light-sensitive tissue in the back of the eye.
The neurons in the retina perform preprocessing of the image and the RGCs represent the pre-processed information in the spike domain.
The image then passes through the optic nerve and a brain region called the lateral geniculate nucleus (LGN).
LGN neurons project their axons into the primary visual cortex V1 located at the back of the head.

CNNs Inspired by Basic Elements of the Visual System

A convolutional network layer is designed to capture three properties of the primary visual cortex V1:

- V1 is arranged in a spatial map (retinotopy); its 2-D structure mirrors the structure of the image on the retina.

  Convolutional networks capture this property by having their features defined in terms of 2-D maps.

- V1 contains many simple cells; a simple cell's activity can be characterized by a linear function of the image in a small, spatially localized receptive field.

  The detector units of a convolutional network are designed to emulate the local properties of simple cells.

- V1 also contains many complex cells; complex cells are invariant to small shifts in the position of the feature.

  CNN pooling units attempt to address small shift invariance. Complex cells are also invariant to some changes in lighting that have inspired some of the cross-channel pooling strategies in CNNs,

CNNs Inspired by Basic Elements ... (dreaming cont'd)

Grandmother Cells

As we repeatedly apply basic strategy of detection followed by pooling and move deeper into the brain, we eventually find cells that respond to some specific concept and are invariant to many transformations of the input. These cells have been nicknamed grandmother cells.

The hypothesis is that a person could have a neuron that activates when seeing an image of their grandmother, regardless of whether she appears on the left or right side of the image, whether the image is a close-up of her face or zoomed out shot of her entire body, whether she is brightly lit, or in shadow, etc.

These grandmother cells have been shown to actually exist in the human brain, in a region called the medial temporal lobe. These medial temporal lobe neurons/circuits are somewhat more general than modern convolutional networks, which would not automatically generalize to identifying a person or object when reading its name.

Differences between CNNs and Mammalian Visual Systems

There are many differences between convolutional networks and the mammalian vision system (too many to enumerate):

- The retina is mostly very low resolution except for a tiny patch called the fovea. Though we feel as if we can see an entire scene in high resolution, this is an illusion created by our visual cortex, as it stitches together several glimpses of small areas.

  Most CNNs receive large full resolution photographs as input.

- The human visual system is integrated with many other senses, such as hearing, and factors like our moods and thoughts.

  Convolutional networks are purely visual.

- The human visual system does much more than just recognize objects. It is able to understand entire scenes including many objects and relationships between objects, and processes rich 3-D geometric information needed for our bodies to interface with the world.

  CNNs have been applied to some of these problems but these applications are in their infancy.

Differences between CNNs and MVSs (cont'd)

Moreover

- V1 is heavily impacted by feedback from higher brain centers. Feedback has been explored extensively in neural network models but has not yet been shown to offer a compelling improvement.
- While feed-forward IT firing rates capture much of the same information as convolutional network features, it's not clear how similar the intermediate computations are. The brain probably uses very different activation and pooling functions.
- An individual neuron's activation probably is not well characterized by a single linear filter response. Our cartoon picture of simple cells and complex cells might create a nonexistent distinction; simple cells and complex cells might both be the same kind of cell but with their "parameters" enabling a continuum of behaviors ranging from what we call "simple" to what we call "complex".

Neuroscience and Learning in CNNs

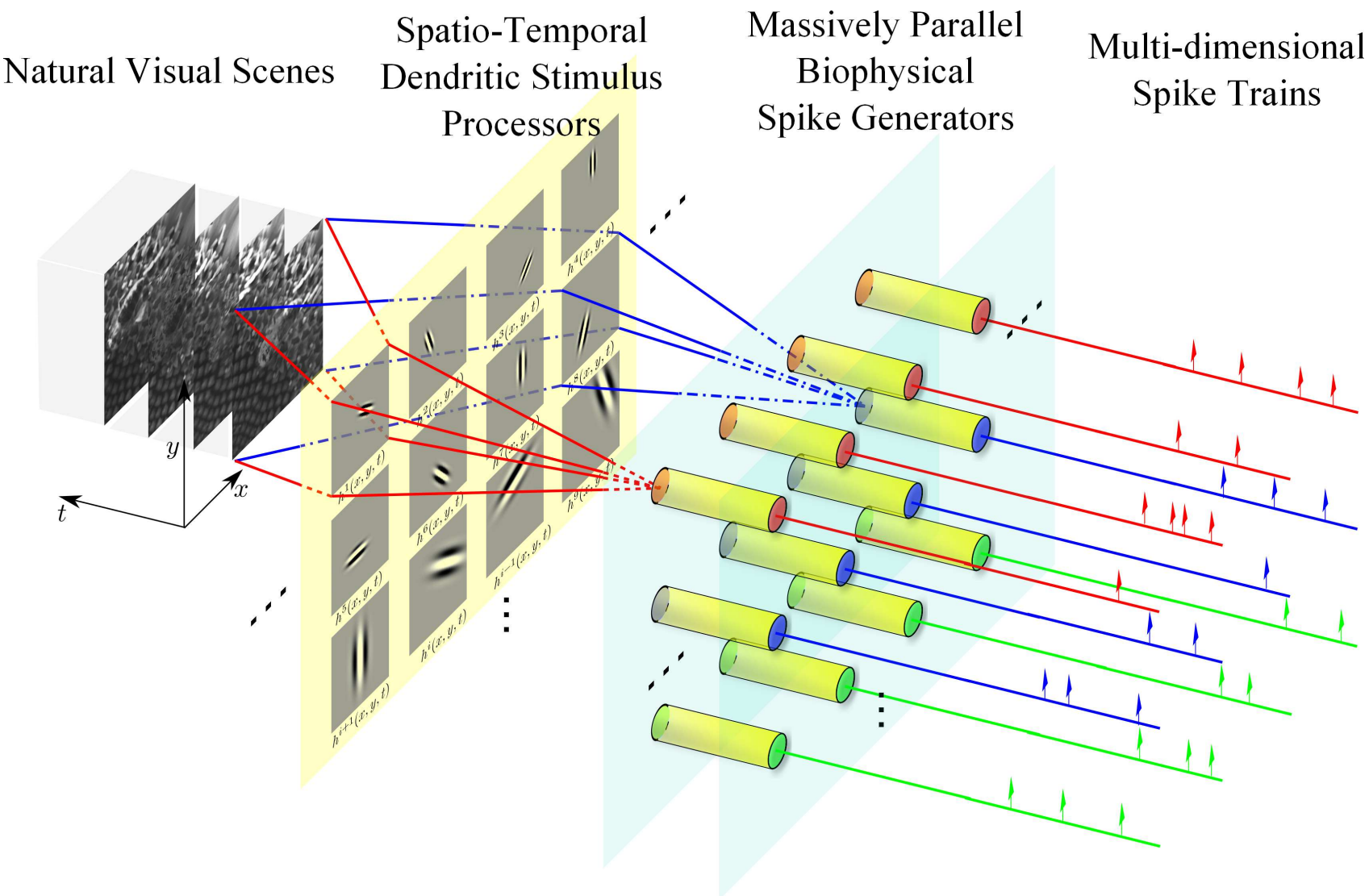
Neuroscience has told us relatively little about how to train convolutional networks.

Back-propagation to train 1-D convolutional networks applied to time series was first introduced in 1988 (Hinton).

Back-propagation applied to these models was not inspired by any neuroscientific observation and is considered to be biologically implausible.

In 1989 (LeCun) developed the modern convolutional network by applying the same training algorithm to 2-D convolution applied to images.

A Model of Neural Encoding in the Primary Visual Cortex



Gabor Filters as Convolution Operators

We can think of an image as being a function of 2-D coordinates, $I(x, y)$. Likewise, we can think of a simple cell as sampling the image at a set of locations, defined by a set of x coordinates $\mathbb{X}$ and a set of y coordinates, $\mathbb{Y}$, and applying weights that are also a function of the location, $w(x, y)$. From this point of view, the response of a simple cell to an image is given by

$$s(I) = \sum_{x \in \mathbb{X}} \sum_{y \in \mathbb{Y}} w(x, y) I(x, y).$$

Here $w(x, y)$ takes the form of a Gabor function:

$$w(x, y; \alpha, \beta_x, \beta_y, f, \phi, x_0, y_0, \tau) = \alpha \exp(-\beta_x x^{'2} - \beta_y y^{'2}) \cos(f x^{'} + \phi),$$

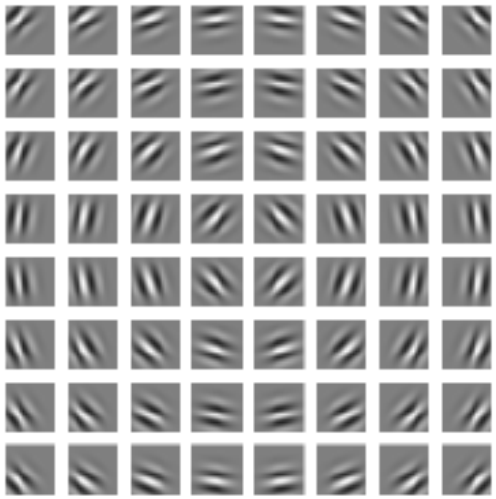
where

$$x^{'} = (x - x_0) \cos(\tau) + (y - y_0) \sin(\tau)$$

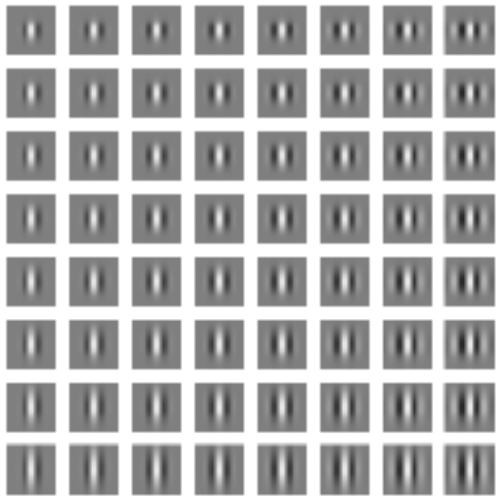
and

$$y^{'} = -(x - y_0) \sin(\tau) + (y - y_0) \cos(\tau).$$

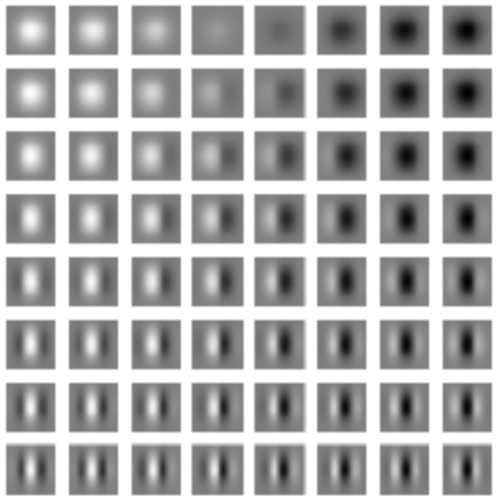
Gabor Filters/Kernels/Functions



Gabor functions with different parameter values (x_0, y_0) and τ.



Gabor functions with different Gaussian scale parameters β_x and β_y.



Gabor functions with different sinusoid parameters f and ϕ.

Part IV

Convolutional Networks - Applications and Historical Breakthroughs

Object Recognition

Object Recognition in Images is HARD

- Segmentation - objects clutter the scene
  - which pieces go together as a part of an object, some parts are hidden behind others
- Lighting - pixel intensity changes with lighting exposure
- Deformation - affine or non-affine ways
  - hand-written number 2 can have a loop or a cusp
- Affordances - which class the object belongs to depends on ... how it is used
  - chairs are for sitting, and a barrel could serve as a chair

Viewpoint Towards an Object

Viewpoint Matters

- Example: images, databases
- Different viewpoints make it hard for traditional learning methods to deal with
  - Information moves around: a face of a child can be in the middle of an image, or in a corner
- Example of dimension-hopping in databases
  - the first field is sometimes a name, and sometimes a title in a company hierarchy

Invariance in Viewpoint

Humans are good at viewpoint invariance

- But it is very hard for computers
- No definite computing solution to the most general problem
- Approaches to achieve invariance
  - use redundant invariant features
  - draw a box around an object and normalize pixels
  - convolutional neural nets which use replicated features with pooling
  - hierachy of parts

Invariance by Convolutional Neural Networks

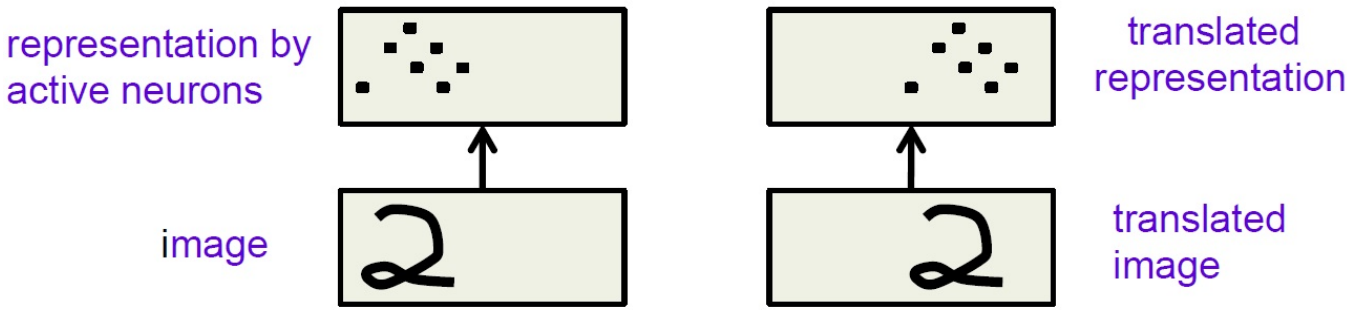
CNNs implement replicated features

- Use many copies of the same feature detector
  - in different positions
  - dramatically reduces the number of free parameters to be learned
  - not easy to replicate accross scale and orientation
- Use several different feature types
  - each with its own map of replicated detectors
  - allows each patch of an image to be represented in several ways

Invariance by Convolutional Neural Networks

Replicated Feature Detectors - What do they achieve?

- Equivariant activities of neurons
  - The activities of individual neuron(al) activities are not invariant to translation
  - The activities are equivariant
- Invariant knowledge
  - in training, a feature may occur in several locations only
  - in testing, the detector for that feature will be available in all location



from Hinton, course on deep learning

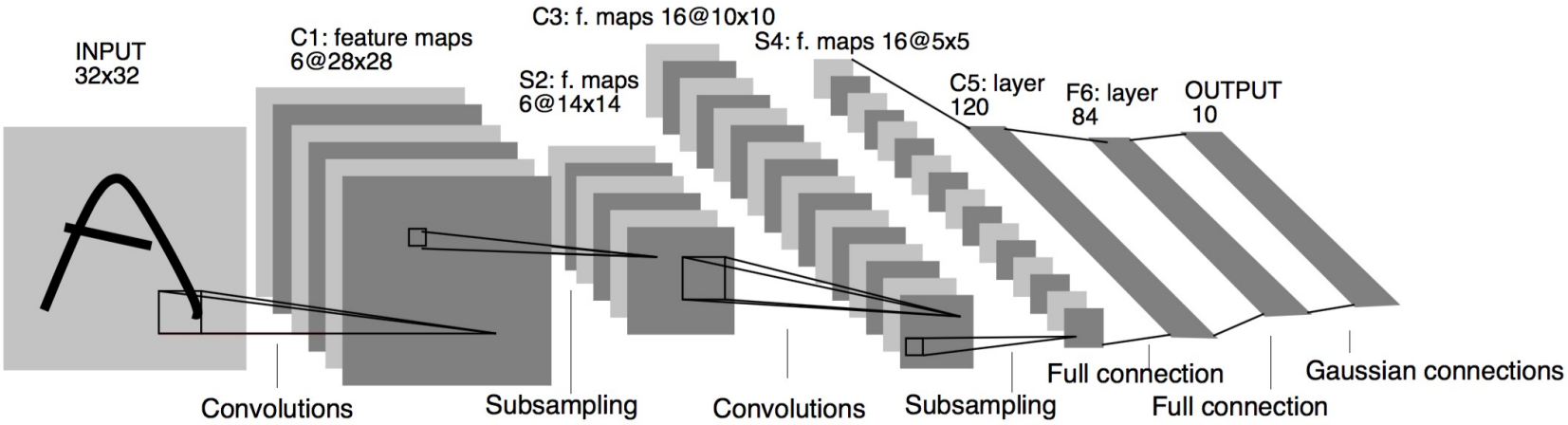
LeNet: an early successful CNN

Digit Recognition

- High quality recognizer of handwritten digits with feedforward NN
  - using backpropagation
  - multiple hidden layers
  - pooling of outputs of nearby replicated units (neurons)
  - a wider net can deal with several characters at once even if they overlap
  - new way of training a complete system
- Was used by 10 percent of bank check-reading applications
- See: http://yann.lecun.com

LeNet: an early successful CNN

LeNet5



LeNet: an early successful CNN

LeNet5