

Convolutional Neural Network

- specialized kind of neural network, grid-like technology
- for example, time-series data - 1D grid, image data - 2D grid
- successful in practical applications
- convolution - linear operation
- convolutional networks - neural networks that use convolution in place of genral matrix multiplication in at least one of their layers
- pooling is employed by all CNN
- convolution networks - example of neuroscientific principles influencing deep learning



Convolution Example

- tracking location of spaceship with laser sensor
- single output $x(t)$ - position of the spaceship at time t
- x and t real values, different reading from laser sensor at any instant in time
- laser sensor noisy, less noisy estimate of the spaceship's position - average several measurements
- more recent measurements are more relevant, weighted average that give more weight to recent measurements - through weighted function $w(a)$, a is age of a measurement



Convolution

- new function s smoothed estimate of the position of the spaceship

$$s(t) = \int x(a)w(t-a)da$$

- this operation is called convolution $s(t) = (x * w)(t)$
- w needs to be valid probability density function
- w sets to 0 for all negative arguments, not look into the future
- x input, w as kernel and s feature map
- for discrete convolution

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a)$$



Convolution

- input multidimensional array of data
- kernel is multidimensional array of parameters - adapted by learning algorithms
- these multidimensional arrays are referred as tensors
- x and w - finite set of points
- in practice s a summation over a finite number of array elements

$$S(i, j) = (I * K)(i, j) = \sum_m \sum_n I(m, n) K(i - m, j - n)$$

- convolution is commutative

$$S(i, j) = (K * I)(i, j) = \sum_m \sum_n I(i - m, j - n) K(m, n)$$



Convolution

- flipped the kernel relative to input, m increases, index into the input increases index into the kernel decreases
- reason to flip the kernel to obtain the commutative property
- neural network libraries implement a related function called cross-correlation
- it is same as convolution but without flipping the kernel

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m, n)$$

- many machine learning libraries implement cross-correlation but call it convolution



Convolution as Matrix Multiplication

- machine learning - algorithm will learn the appropriate values of the kernel in the appropriate place
- algorithm based on convolution with kernel flipping will learn a kernel that is flipped relative to the kernel learned by an algorithm without the flipping
- discrete convolution can be viewed as multiplication by a matrix
- matrix has several entries constrained to be equal to other entries
- for example, for univariate discrete convolution, each row of the matrix is constrained to be equal to the row above shifted by one element
- this is known as a Toeplitz matrix
- in two dimensions, a doubly block circulant matrix corresponds to convolution



Machine Learning and Convolution

- convolution - improve machine learning - three ideas - sparse interactions, parameter sharing and equivariant representation
- parameter describing the interaction between each input unit and each output unit
- kernel smaller than the input
- if there are m inputs and n outputs, matrix multiplication requires $m \times n$ parameters $O(m \times n)$ runtime
- limiting the number of connections each output may have to k then $k \times n$ parameters required and runtime $O(k \times n)$
- for many practical applications, it is possible to obtain good performance on the machine learning task while keeping
- k several orders of magnitude smaller than m



Convolutional Neural Network

- in a deep convolutional network, units in the deeper layers may indirectly interact with a larger portion of the input
- this allows the network to efficiently describe complicated interactions between many variables by constructing such interactions from simple building blocks that each describe only sparse interactions
- parameter sharing refers to using the same parameter for more than one function in a model
- in a traditional neural net, each element of the weight matrix is used exactly once when computing the output of a layer
- it is multiplied by one element of the input and then never revisited
- as a synonym for parameter sharing, one can say that a network has tied weights, because the value of the weight applied to one input is tied to the value of a weight applied elsewhere



Convolutional Neural Network

- in a convolutional neural net, each member of the kernel is used at every position of the input
- except perhaps some of the boundary pixels, depending on the design decisions regarding the boundary
- the parameter sharing used by the convolution operation means that rather than learning a separate set of parameters for every location,
- learn only one set
- this does not affect the runtime of forward propagation — it is still $O(k \times n)$
- reduce the storage requirements of the model to k parameters
- k is usually several orders of magnitude less than m
- m and n are usually roughly the same size, k is practically insignificant compared to $m \times n$

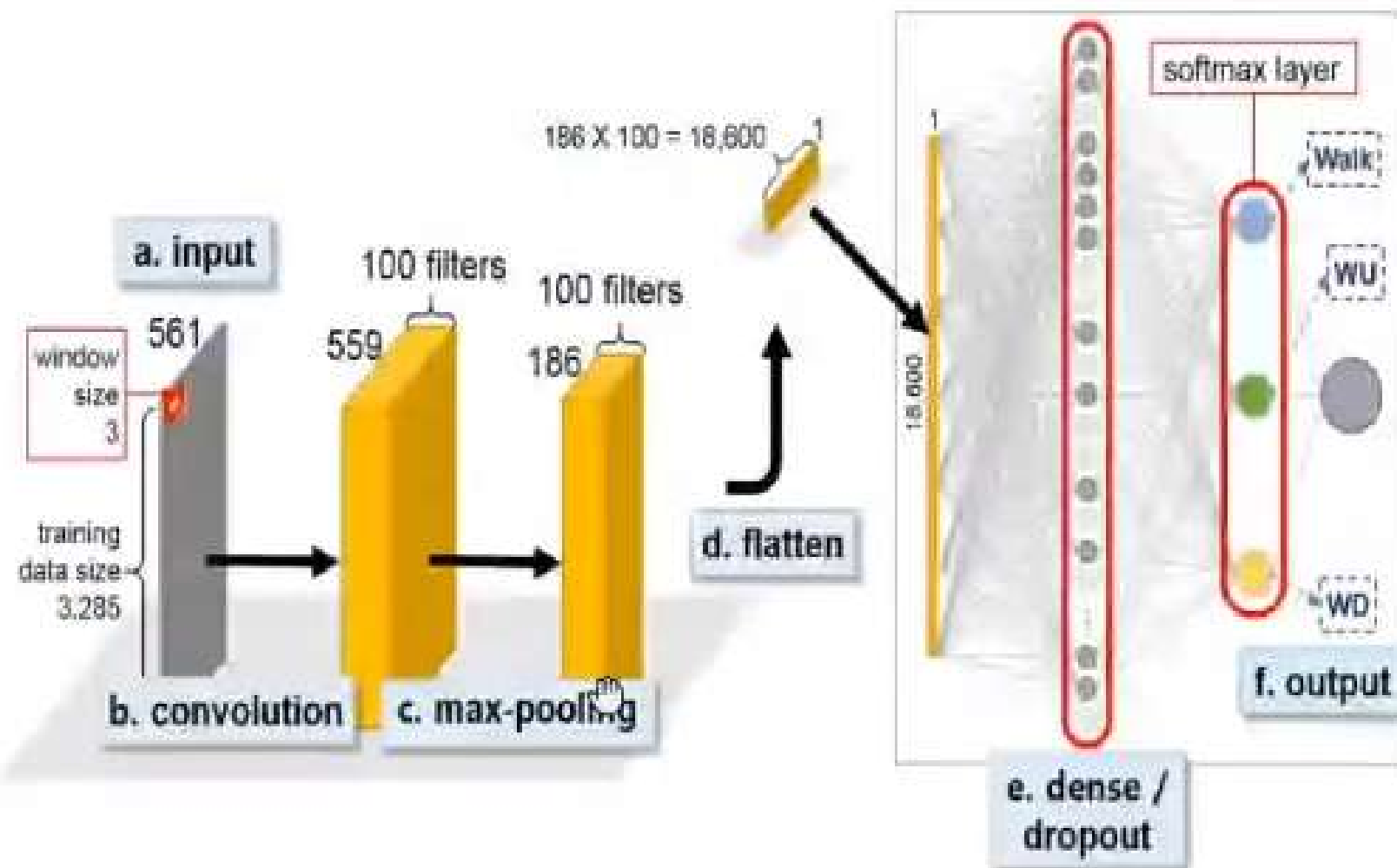


Convolutional Neural Network

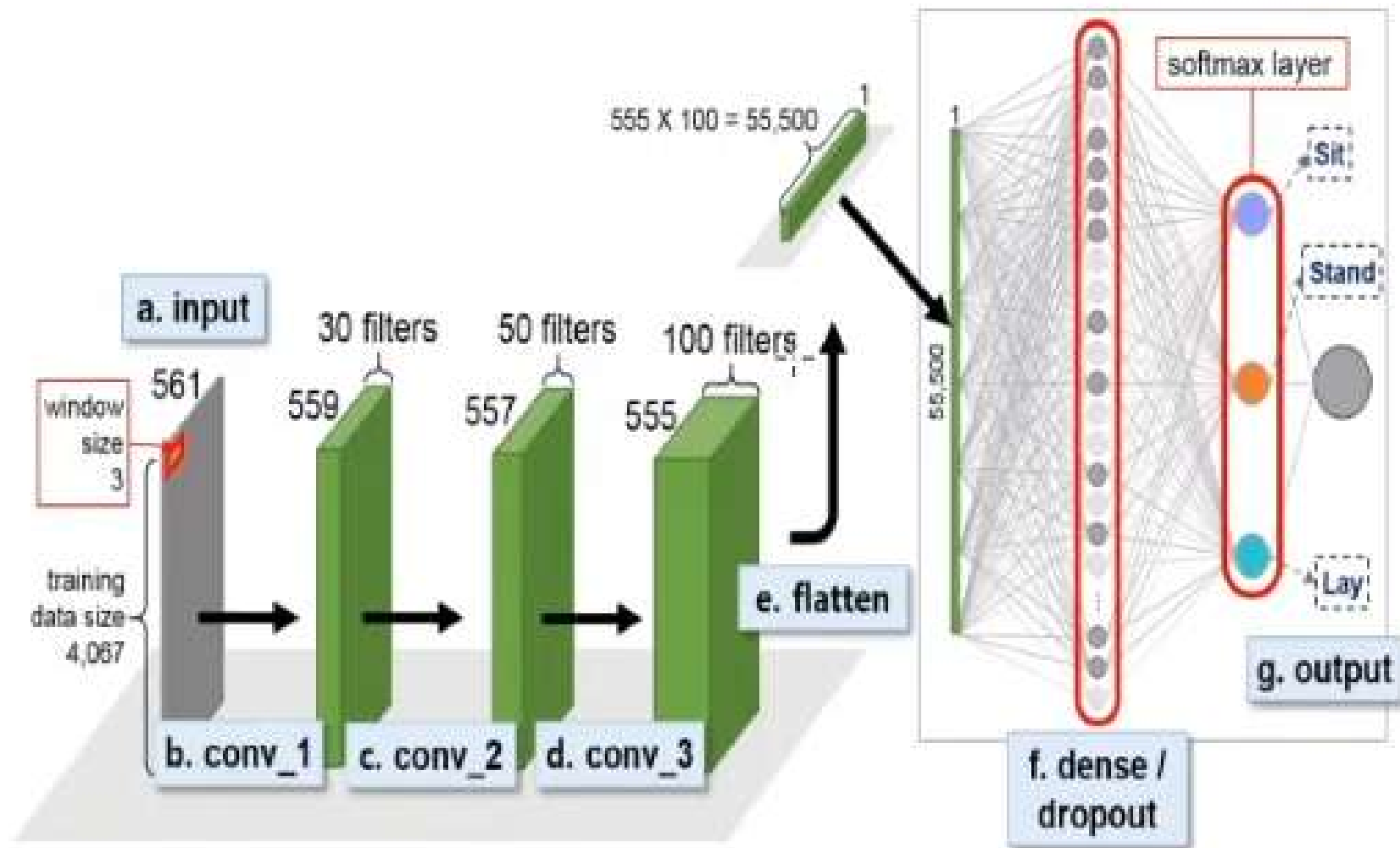
- input image, convolution layer, subsampling (pooling),
- fully connected layer, classification layer
- convolution filter, say, 3×3 , windowing, filtering, masking
- low pass filter, high pass filter etc.,
- sliding overlapping window or without overlapping results into different size output of convolution
- multiple filters applied at same location
- pooling or subsampling reduces size
- followed by classical neural network based fully connected layer and classification layer



Convolutional Neural Network Example (image source:Google)

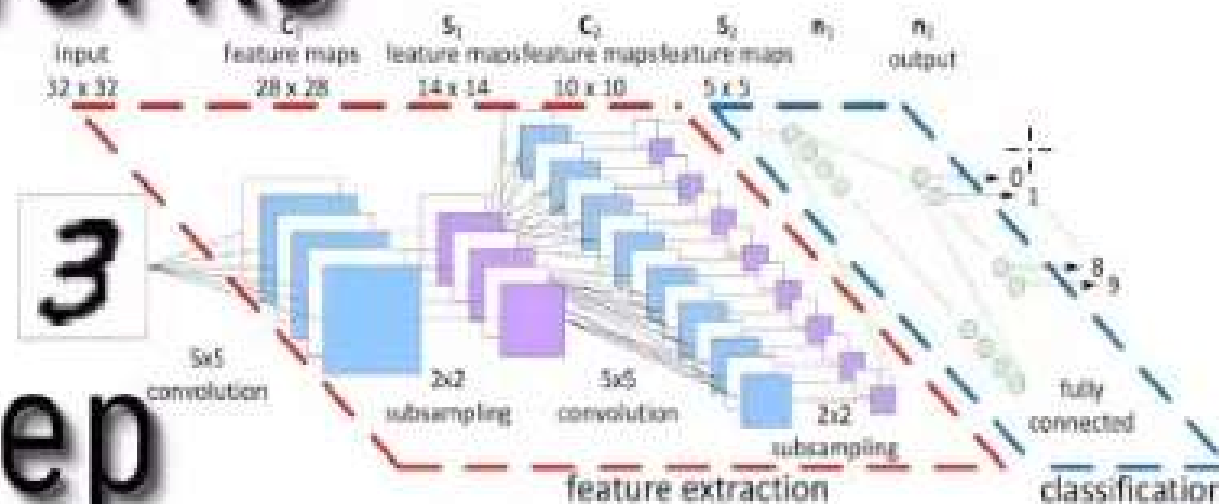


Convolutional Neural Network Example (image source: Google)

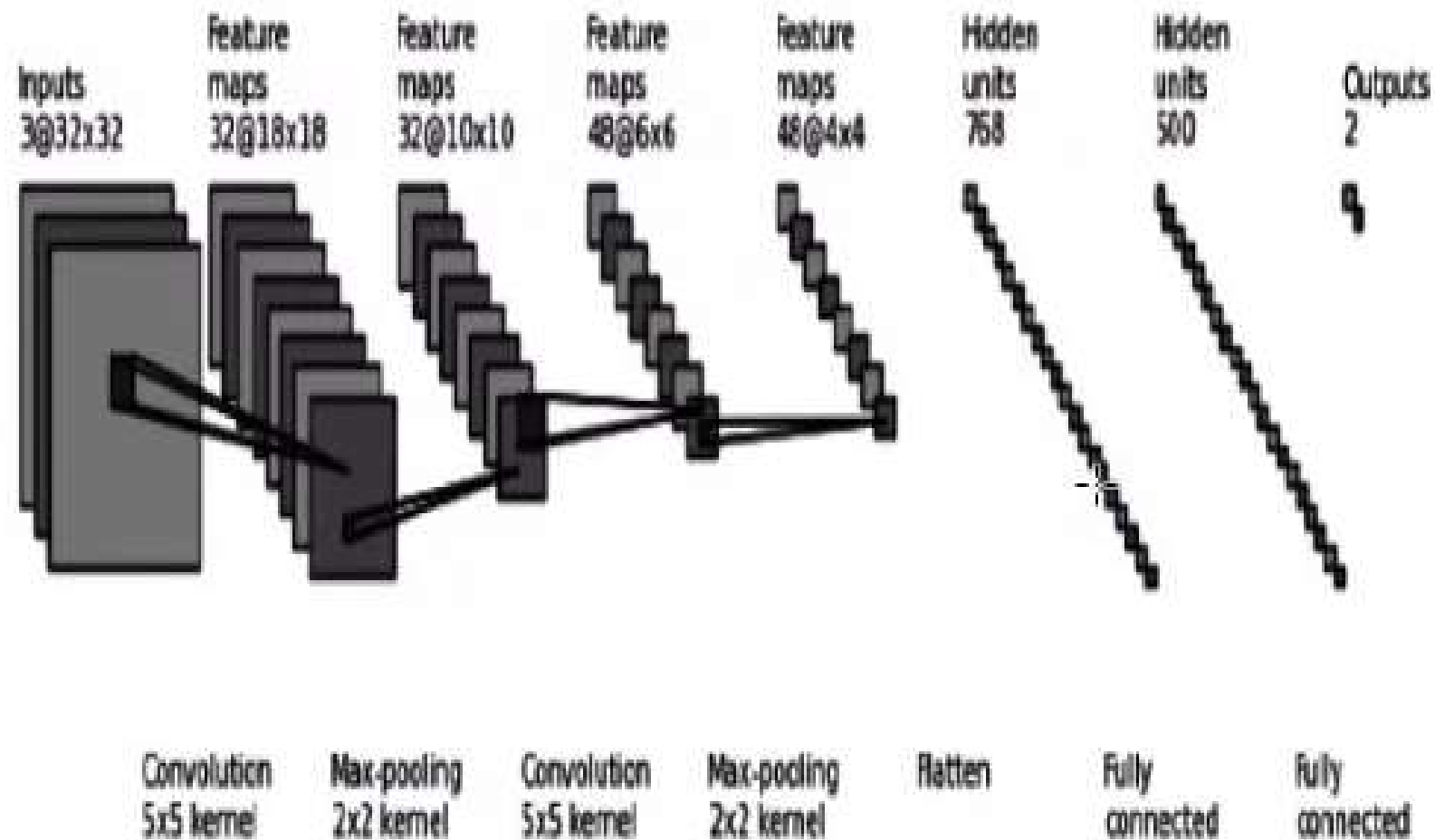


Convolutional Neural Networks

step 3
by step
introduction



Convolutional Neural Network Example (image source: Google)




Convolutional Neural Network

- convolution is more efficient than dense matrix multiplication in terms of the memory requirements and statistical efficiency
- in the case of convolution, the particular form of parameter sharing causes the layer to have a property called equivariance to translation
- to say a function is equivariant means that if the input changes, the output changes in the same way
- function $f(x)$ is equivariant to a function g if $f(g(x)) = g(f(x))$
- in the case of convolution, let g be any function that translates the input, i.e., shifts it, then the convolution function is equivariant to g
- for example $I' = g(I)$ image function with $I'(x, y) = I(x - 1, y)$
- if this transformation is applied to I , then apply convolution, the result will be the same as if convolution is applied to I , then applied the transformation g to the output



Convolutional Neural Network

- time series data - convolution produces a sort of timeline that shows when different features appear in the input
- if an event is moved later in time in the input, the exact same representation of it will appear in the output, just later in time
- similarly with images, convolution creates a 2-D map of where certain features appear in the input
- if the object is moved in the input, its representation will move the same amount in the output 
- this is useful for when it is known that some function of a small number of neighboring pixels is useful when applied to multiple input locations



Convolutional Neural Network

- for example, when processing images, it is useful to detect edges in the first layer of a convolutional network
- the same edges appear more or less everywhere in the image, so it is practical to share parameters across the entire image
- in some cases, it is not required to share parameters across the entire image
- for example, if the images are processed, that are cropped to be centered on an individual's face,
- probably want to extract different features at different locations
- the part of the network processing the top of the face needs to look for eyebrows, while
- the part of the network processing the bottom of the face needs to look for a chin



Convolutional Neural Network

- convolution is not naturally equivariant to some other transformations, such as
- changes in the **scale or rotation** of an image, other mechanisms are necessary for handling these kinds of transformations
- some kinds of data cannot be processed by **neural networks** defined by matrix multiplication with a fixed-shape matrix
- convolution enables processing of some of these kinds of data
- in the **first stage**, the layer performs several convolutions in parallel to produce a set of linear activations
- in the **second stage**, each linear activation is run through a **nonlinear activation function**, such as the rectified linear activation function
- this stage is sometimes called the **detector stage**
- in the **third stage**, a **pooling function** is used to modify the output of the layer further



Convolutional Neural Network: Pooling

- pooling function replaces the output of the net at a certain location with a summary statistic of the nearby outputs
- for example, the max pooling operation reports the maximum output within a rectangular neighborhood
- other popular pooling functions include the average of a rectangular neighborhood
- the L^2 norm of a rectangular neighborhood or a weighted average based on the distance from the central pixel
- in all cases, pooling helps to make the representation become approximately invariant to small translations of the input
- invariance to translation means that if the input is translated by a small amount, the values of most of the pooled outputs do not change



Convolutional Neural Network: Pooling

- invariance to local translation can be a very useful property if it takes care more about whether some feature is present than exactly where it is
- for example, when determining whether an image contains a face, need not to know the location of the eyes with pixel-perfect accuracy,
- just need to know that there is an eye on the left side of the face and an eye on the right side of the face
- in other contexts, it is more important to preserve the location of a feature
- for example, say, wants to find a corner defined by two edges meeting at a specific orientation, needs to preserve the location of the edges well enough to test whether they meet



Convolutional Neural Network: Pooling

- the use of pooling can be viewed as adding an infinitely strong prior that the function the layer learns must be invariant to small translations
- when this assumption is correct, it can greatly improve the statistical efficiency of the network
- pooling over spatial regions produces invariance to translation, but
- if it is pooled over the outputs of separately parametrized convolutions, the features can learn which transformations to become invariant to
- because pooling summarizes the responses over a whole neighborhood, it is possible to use fewer pooling units than detector units, by reporting summary statistics for pooling regions spaced k pixels apart rather than 1 pixel apart



Convolutional Neural Network: Pooling

- this improves the computational efficiency of the network because the next layer has roughly k times fewer inputs to process
- when the number of parameters in the next layer is a function of its input size (such as when the next layer is fully connected and based on matrix multiplication)
- this reduction in the input size can also result in improved statistical efficiency and reduced memory requirements for storing the parameters
- for many tasks, pooling is essential for handling inputs of varying size
- for example, if wants to classify images of variable size, the input to the classification layer must have a fixed size

