Convolution Network

FDP on Artificial Intelligence
sponsored by AICTE Training And Learning (ATAL) Academy
organized by Department of Computer Science and Engineering,
Motilal Nehru National Institute of Technology, Allahabad, Prayagraj, U.P.
(1-5 October, 2020)

Mukesh A. Zaveri
Computer Engineering Department
Sardar Vallabhbhai National Institute of Technology, Surat
mazaveri@coed.svnit.ac.in

Convolutional Neural Network

- specialized kind of neural network, grid-like technology

Convolutional Neural Network

- specialized kind of neural network, grid-like technology
- for example, time-series data - 1D grid, image data - 2D grid

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

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 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

M. A. Zaveri, SVNIT, Surat Convolution Network 2 October 2020 2 / 40

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

M. A. Zaveri, SVNIT, Surat Convolution Network 2 October 2020 2 / 40

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 neuroscienctific principles influencing deep learning

M. A. Zaveri, SVNIT, Surat Convolution Network 2 October 2020 2 / 40

Convolution Example

- tracking location of spaceship with laser sensor
- single output $x(t)$ - position of the spachship 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 serveral 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

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

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

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 covolution 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 -

Machine Learning and Convolution

- convolution - improve machine learning - three ideas - sparse interactions, parameter sharing and equivariant representation

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

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

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

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

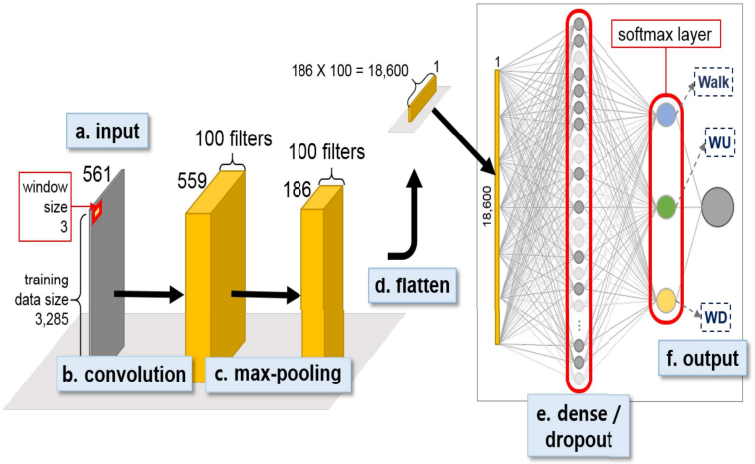
Convolutional Neural Network

- 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 $m \times n$

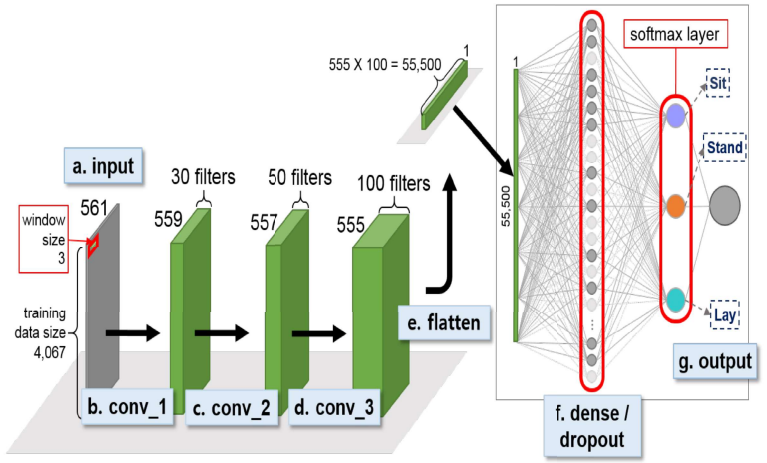
Convolutional Neural Network

- 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

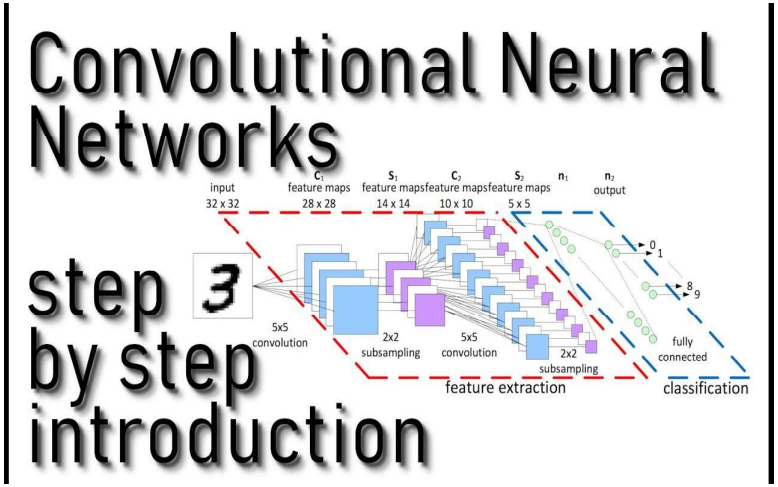
Convolutional Neural Network Example (image source:Google)



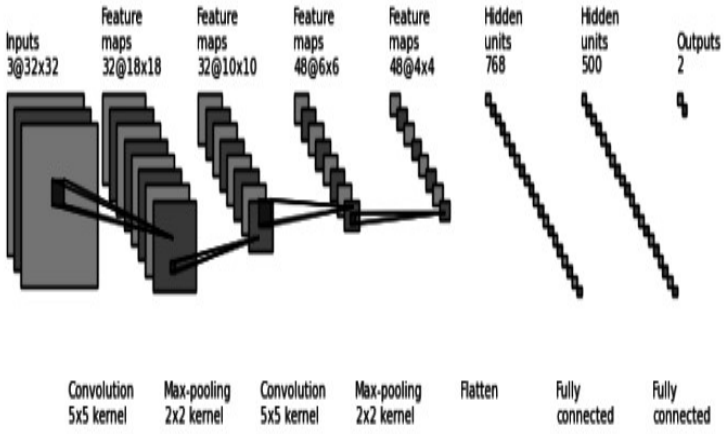
Convolutional Neural Network Example (image source:Google)



Convolutional Neural Network Example (image source:Google)



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

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

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 $f(g(x)) = g(f(x))$

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

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

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

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

M. A. Zaveri, SVNIT, Surat Convolution Network 2 October 2020 18 / 40

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

M. A. Zaveri, SVNIT, Surat Convolution Network 2 October 2020 19 / 40

Convolutional Neural Network: Pooling

- pooling function replaces the output of the net at a certain location with a summary statistic of the nearby outputs

M. A. Zaveri, SVNIT, Surat Convolution Network 2 October 2020 20 / 40

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 wighted averge 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

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 wighted averge 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

M. A. Zaveri, SVNIT, Surat | Convolution Network | 2 October 2020 | 20 / 40

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,

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

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

Convolutional Neural Network

- this is usually accomplished by varying the size of an offset between pooling regions so that the classification layer always receives the same number of summary statistics regardless of the input size
- for example, the final pooling layer of the network may be defined to output four sets of summary statistics, one for each quadrant of an image, regardless of the image size
- some theoretical work gives guidance as to which kinds of pooling one should use in various situations, say, dynamically pool features
- for example, by running a clustering algorithm on the locations of interesting features
- this approach yields a different set of pooling regions for each image
- another approach is to learn a single pooling structure that is then applied to all images

Convolutional Neural Network

- Convolution and Pooling as an Infinitely Strong Prior
- recall the concept of a prior probability distribution
- this is a probability distribution over the parameters of a model that encodes our belief about what models are reasonable, before any data have been seen
- priors can be considered weak or strong depending on how concentrated the probability density in the prior is
- a weak prior is a prior distribution with high entropy, such as a Gaussian distribution with high variance
- such a prior allows the data to move the parameters more or less freely

Convolutional Neural Network

- a strong prior has very low entropy, such as a Gaussian distribution with low variance
- such a prior plays a more active role in determining where the parameters end up
- an infinitely strong prior places zero probability on some parameters and says that
- these parameter values are completely forbidden, regardless of how much support the data gives to those values
- imagine a convolutional net as being similar to a fully connected net,
- but with an infinitely strong prior over its weights
- this infinitely strong prior says that the weights for one hidden unit must be identical to the weights of its neighbor, but shifted in space

Convolutional Neural Network

- the prior also says that the weights must be zero, except for in the small, spatially contiguous receptive field assigned to that hidden unit
- the use of convolution as introducing an infinitely strong prior probability distribution over the parameters of a layer
- this prior says that the function the layer should learn contains only local interactions and is equivariant to translation
- the use of pooling is an infinitely strong prior that each unit should be invariant to small translations
- implementing a convolutional net as a fully connected net with an infinitely strong prior would be extremely computationally wasteful but it can give us some insights into how convolutional nets work

Convolutional Neural Network

- convolution and pooling can cause underfitting
- convolution and pooling are only useful when the assumptions made by the prior are reasonably accurate
- if a task relies on preserving precise spatial information, then using pooling on all features can increase the training error
- some convolutional network architectures are designed to use pooling on some channels but not on other channels,
- in order to get both highly invariant features and features that will not underfit when the translation invariance prior is incorrect
- when a task involves incorporating information from very distant locations in the input, then the prior imposed by convolution may be inappropriate

Convolutional Neural Network

- for example, a color image has a red, green and blue intensity at each pixel
- 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, the input and output of the convolution as being 3-D tensors,
- with one index into the different channels and two indices into the spatial coordinates of each channel

Convolutional Neural Network

- software implementations work in batch mode, so they use 4-D tensors
- with the fourth axis indexing different examples in the batch
- 4-D kernel tensor with element i, j, k, l 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 input consists of observed data with element i, j, k giving the value of the input unit within channel i at row j and column k

Convolutional Neural Network

- want to skip over some positions of the kernel in order to reduce the computational cost
- at the expense of not extracting features as finely

Convolutional Neural Network

- want to skip over some positions of the kernel in order to reduce the computational cost
- at the expense of not extracting features as finely
- think of this as downsampling the output of the full convolution function
- want to sample only every s pixels in each direction in the output, then a downsampled convolution function can be defined as c
- refer s as the stride of this downsampled convolution

Convolutional Neural Network

- want to skip over some positions of the kernel in order to reduce the computational cost
- at the expense of not extracting features as finely
- think of this as downsampling the output of the full convolution function
- want to sample only every s pixels in each direction in the output, then a downsampled convolution function can be defined as c
- refer s as the stride of this downsampled convolution
- it is also possible to define a separate stride for each direction of motion

Convolutional Neural Network

- essential feature of any convolutional network implementation is the ability to implicitly zero-pad the input in order to make it wider
- without this feature, the width of the representation shrinks by one pixel less than the kernel width at each layer
- zero padding the input allows us to control the kernel width and the size of the output independently
- without zero padding, forced to choose between shrinking the spatial extent of the network rapidly and using small kernels
- three special cases of the zero-padding setting
- the extreme case in which no zero-padding is used whatsoever, and the convolution kernel is only allowed to visit positions where the entire kernel is contained entirely within the image

Convolutional Neural Network

- all pixels in the output are a function of the same number of pixels in the input, so the behavior of an output pixel is somewhat more regular
- the size of the output shrinks at each layer
- if the input image has width m and the kernel has width k, the output will be of width $m - k + 1$
- the rate of this shrinkage can be dramatic if the kernels used are large
- the shrinkage is greater than 0, it limits the number of convolutional layers that can be included in the network
- as layers are added, the spatial dimension of the network will eventually drop to 1×1, at which point additional layers cannot meaningfully be considered convolutional

Convolutional Neural Network

- another special case of the zero-padding setting is when just enough zero-padding is added to keep the size of the output equal to the size of the input
- the network can contain as many convolutional layers as the available hardware can support
- the operation of convolution does not modify the architectural possibilities available to the next layer
- the input pixels near the border influence fewer output pixels than the input pixels near the center
- this can make the border pixels somewhat underrepresented in the model

Convolutional Neural Network

- full convolution, in which enough zeroes are added for every pixel to be visited k times in each direction, resulting in an output image of width $m + k - 1$
- the output pixels near the border are a function of fewer pixels than the output pixels near the center
- this can make it difficult to learn a single kernel that performs well at all positions in the convolutional feature map
- unshared convolution, because it is a similar operation to discrete convolution with a small kernel, but without sharing parameters across locations

Convolutional Neural Network

- locally connected layers are useful when each feature is a function of a small part of space, but
- there is no reason to think that the same feature should occur across all of space
- for example, for recognizing that an image is a picture of a face, need to look for the mouth in the bottom half of the image
- it can also be useful to make versions of convolution or locally connected layers in which the connectivity is further restricted,
- for example to constrain that each output channel i be a function of only a subset of the input channels l
- a common way to do this is to make the first m output channels connect to only the first n input channels, the second m output channels connect to only the second n input channels, and so on

Convolutional Neural Network

- modeling interactions between few channels allows the network to have fewer parameters in order to reduce memory consumption and increase statistical efficiency, and also
- reduces the amount of computation needed to perform forward and back-propagation
- it accomplishes these goals without reducing the number of hidden units
- Tiled convolution (Gregor and LeCun, 2010a; Le et al., 2010) offers a compromise between a convolutional layer and a locally connected layer

Convolutional Neural Network

- rather than learning a separate set of weights at every spatial location, learn a set of kernels that rotate through as move through space
- this means that immediately neighboring locations will have different filters, like in a locally connected layer, but
- the memory requirements for storing the parameters will increase only by a factor of the size of this set of kernels, rather than the size of the entire output feature map
- to define tiled convolution algebraically, let k be a 6-D tensor, where
- two of the dimensions correspond to different locations in the output map
- rather than having a separate index for each location in the output map, output locations cycle through a set of t different choices of kernel stack in each direction

- if t is equal to the output width, this is the same as a locally connected layer
- both locally connected layers and tiled convolutional layers have an interesting interaction with max-pooling:
- the detector units of these layers are driven by different filters
- if these filters learn to detect different transformed versions of the same underlying features, then
- the max-pooled units become invariant to the learned transformation
- convolutional layers are hard-coded to be invariant specifically to translation

Thank You