Variants of the Basic Convolution Function

Sargur Srihari srihari@buffalo.edu

Topics in Convolutional Networks

- Overview
- 1. The Convolution Operation
- 2. Motivation
- 3. Pooling
- 4. Convolution and Pooling as an Infinitely Strong Prior
- 5. Variants of the Basic Convolution Function
- 6. Structured Outputs
- 7. Data Types
- 8. Efficient Convolution Algorithms
- 9. Random or Unsupervised Features
- 10. The Neuroscientific Basis for Convolutional Networks
- 11. Convolutional Networks and the History of Deep Learning

Topics in Variants of Convolution Functions

- Neural net convolution is not same as mathematical convolution
- How convolution in neural networks is different
- Multichannel convolution due to image color and batches
- Convolution with a stride
- Locally connected layers (unshared convolution)
- Tiled convolution
- Implementation of a convolutional network

Neural Net Convolution is Different

- Convolution in the context of neural networks does not refer exactly to the standard convolution operation in mathematics
- The functions used differ slightly
- Here we describe the differences in detail and highlight their useful propertied

Convolution Operation in Neural Networks

1. It refers to an operation that consists of many applications of convolution in parallel

- This is because convolution with a single kernel can only extract one kind of feature, albeit at many locations
- Usually we want to extract many kinds of features at many locations

2. Input is usually not a grid of real values

- Rather it is a vector of observations
 - E.g., a color image has R ,G, B values at each pixel
- Input to the next layer is the output of the first layer which has many different convolutions at each position
- When working with images, input and output are 3-D tensors

Four indices with image software

- 1. One index for Channel
- 2. Two indices for spatial coordinates of each channel
- 3. Fourth index for different samples in a batch
 - We omit the batch axis for simplicity of discussion

Multichannel Convolution

- Because we are dealing with multichannel convolution, linear operations are not usually commutative, even of kernel flipping is used
- These multi-channel operations are only commutative if each operation has the same number of output channels as input channels.

Definition of 4-D kernel tensor

- Assume we have a 4-D kernel tensor ${\bf K}$ with element $K_{i,j,k,l}$ 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 output and input units
- Assume our input consists of observed data ${\bf V}$ with element $V_{i,i,k}$ giving the value of the input unit
 - within channel i at row j and column k.
- Assume our output consists of \mathbf{Z} with the same format as \mathbf{V} .
- If ${\bf Z}$ is produced by convolving ${\bf K}$ across ${\bf V}$ without flipping ${\bf K}$, then

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

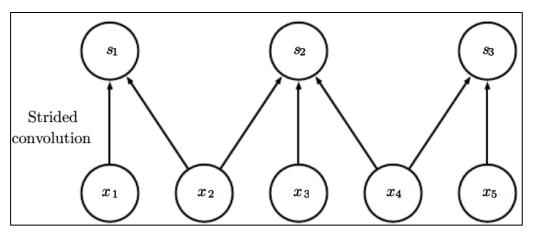
Convolution with a stride: Definition

- We may want to skip over some positions in the kernel to reduce computational cost
 - At the cost of not extracting fine features
- We can think of this as down-sampling the output of the full convolution function
- If we want to sample only every s pixels in each direction of output, then we can define a down-sampled convolution function c such that

$$Z_{i,j,k} = c(\mathbf{K}, \mathbf{V}, s)_{i,j,k} = \sum_{l,m,n} \left[V_{l,(j-1)\times s+m,(k-1)\times s+n} K_{i,l,m,n} \right]$$

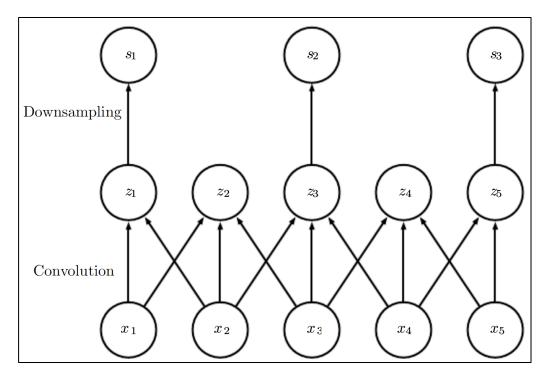
 We refer to s as the stride. It is possible to define a different stride for each direction

Convolution with a stride: Implementation



Here we use a stride of 2

Convolution with a stride of length two implemented in a single operation

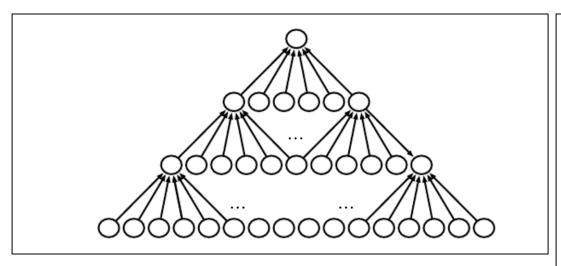


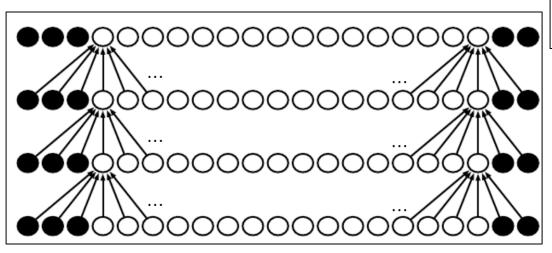
Convolution with a stride greater than one pixel is mathematically equivalent to convolution with a unit stride followed by down-sampling.

Two-step approach is computationally wasteful, because it discard many values that are discarded

Effect of Zero-padding on network size

Convolutional net with a kernel of $width\ 6$ at every layer No pooling, so only convolution shrinks network size





We do not use any implicit zero padding

Causes representation to shrink by five pixels at each layer
Starting from an input of 16 pixels we are only able to have 3 convolutional layers and the last layer does not ever move the kernel, so only two layers are convolutional

By adding 5 implicit zeroes to Each layer, we prevent the Representation from shrinking with depth

This allows us to make an arbitrarily deep convolutional network

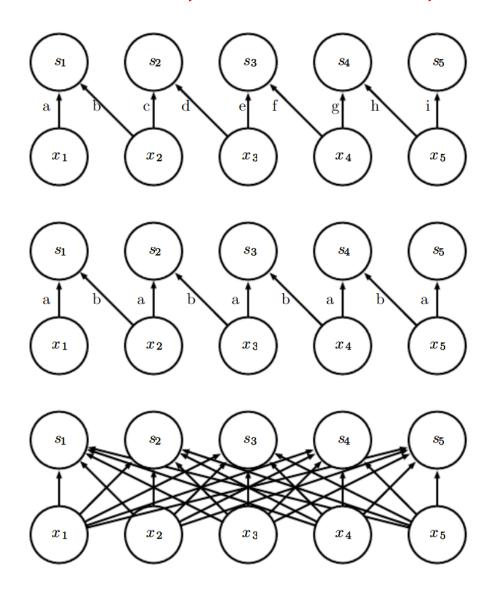
Locally connected layer

- In some cases, we do not actually want to use convolution, but rather locally connected layers
 - 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

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

Also called unshared convolution

Local connections, convolution, full connections



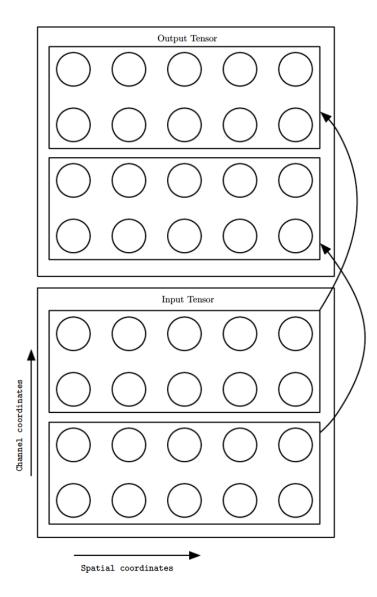
Use of locally connected layers

- Locally connected layers are useful when
 - we know that each feature should be 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
- Ex: if we want to tell if an image is a picture of a face, we only need to look for the mouth in the bottom half of the image

Constraining Outputs

- Constrain each output channel i to be a function of only a subset of the input channels l
 - 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, etc
- Modeling interactions between few channels allows fewer parameters to:
 - Reduce memory, increase statistical efficiency, reduce computation for forward/back-propagation.
 - It accomplishes these goals without reducing ng.of hidden units.

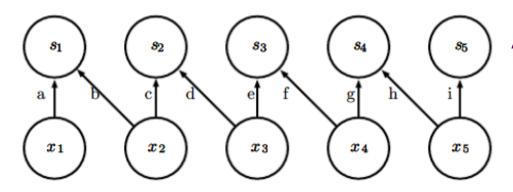
Network with further restricted connectivity



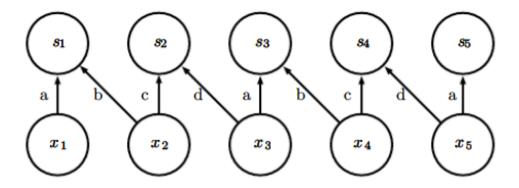
Tiled Convolution

- Compromise between a convolutional layer and a locally connected layer.
 - Rather than learning a separate set of weights at every spatial location, we learn a set of kernels that we rotate through as we 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.

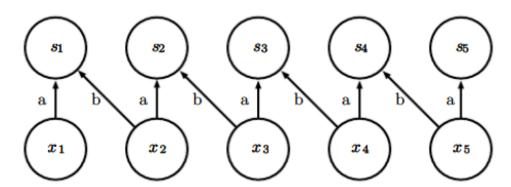
Comparison of locally connected layers, tiled convolution and standard convolution



A locally connected layer
Has no sharing at all
Each connection has its own weight



Tiled convolution Has a set of different kernels With t=2



Traditional convolution Equivalent to tiled convolution with t=1 There is only one kernel and it is applied everywhere

Defining 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

$$Z_{i,j,k} = \sum_{l,m,n} V_{l,j+m-1,k+n-1} K_{i,l,m,n,j\%t+1,k\%t+1}$$

• where % is the modulo operation, with t%t = 0, (t+1)%t = 1, etc

Operations to implement convolutional nets

- Besides convolution, other operations are necessary to implement a convolutional network.
- To perform learning, need to compute gradient wrt the kernel, given the gradient with respect to the outputs.
- In some simple cases, this operation can be performed using the convolution operation, but when stride greater than 1, do not have this property.

Implementation of Convolution

- Convolution is a linear operation and can thus be described as a matrix multiplication
 - if we first reshape the input tensor into a flat vector
- Matrix involved is a function of the convolution kernel
 - Matrix is sparse and each element of the kernel is copied to several elements of the matrix.
- This view helps us to derive some of the other operations needed to implement a convolutional network