Lecture 8 Convolutional Neural Networks

Shiwei Lan[1]

[1]School of Mathematical and Statistical Sciences
Arizona State University

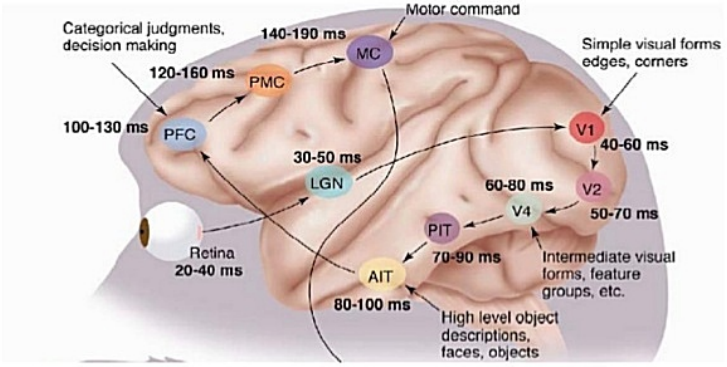
STP598 Machine Learning and Deep Learning
Fall 2021

Table of Contents

Convolutional Neural Networks

- Convolutional Neural Networks (CNN) are a specialized kind of neural network for processing data that has a known, grid-like topology, e.g. time-series, images, etc.
- CNNs are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers.
- Instead of using all input features to create the linear combination, a "convolutional layer" builds neurons that each takes a subset (a local region) of the input features.
- This is motivated by the fact that biologically, the neurons only take signals from neighboring neurons.

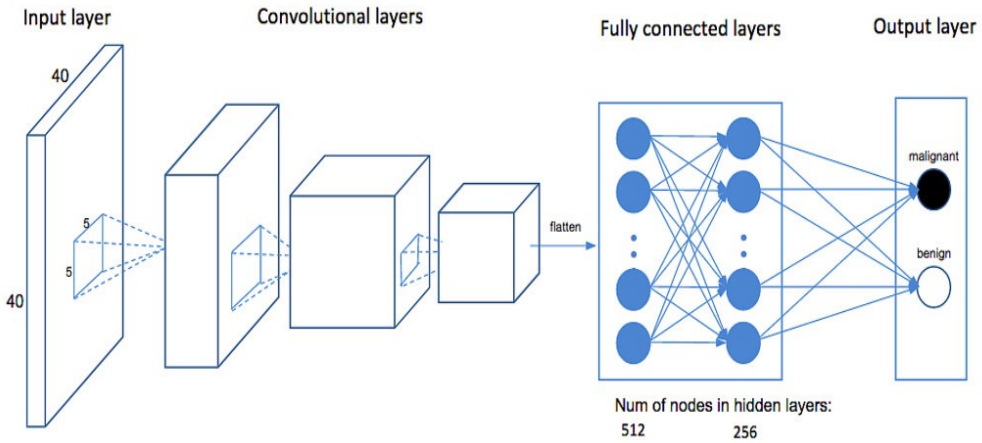
Deep Neural Networks: Feature Hierarchy



Hierarchical information processing in the brain

(Source: Simon Thorpe)

Convolutional Neural Networks



See this hand digit writing recognition example, and this interesting application by Tesla.

Table of Contents

Convolution

- Suppose we track the location of a spaceship in real time $x(t)$.
- The sensor is noisy and we need to estimate the position based on noisy measurements.
- More recent measurements are more relevant so we use a weighting function $w(a)$ to emphasize that.
- A smoothed estimate of the position is given by the following *convolution*

$$s(t) = (x * w)(t) := \int x(a)w(t-a)da \tag{1}$$

  where we require $w(a) = 0$ for $a < 0$.
- In the CNN terminology, we have:
  - **input** x
  - **kernel** w
  - **feature map** s

Convolution

- In practice, we observe noisy measure at discrete time points so we have the *discrete convolution*

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a) \tag{2}$$

- In two-dimensional case, e.g. with image I as input, we define the convolution with a 2-d kernel K:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \tag{3}$$

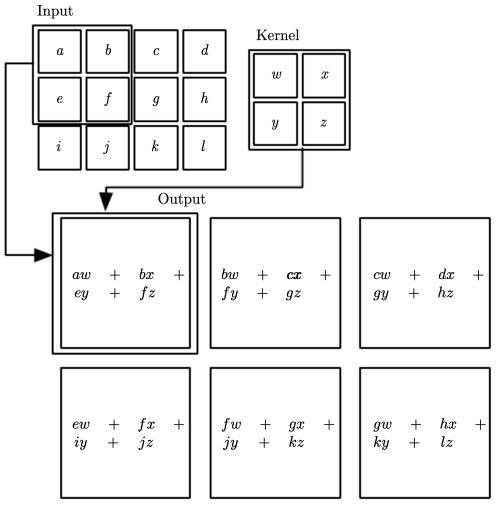
- Or equivalently (convolution is commutative),

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n) \tag{4}$$

- Many neural network libraries implement *cross-correlation*

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n) \tag{5}$$

Convolution

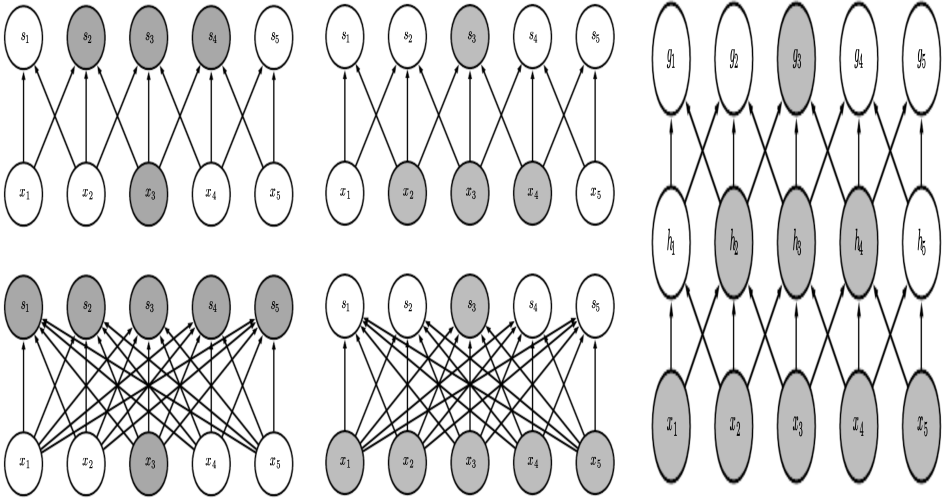


$$T = \begin{bmatrix} w_0 & 0 & 0 & 0 & \cdot & 0 \\ w_1 & w_0 & 0 & 0 & \cdot & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ w_s & w_{s-1} & \cdots & w_0 & 0\cdots & 0 \\ 0 & w_s & \cdots & w_1 & w_0\cdots & 0 \\ \vdots & \ddots & \ddots\ddots & \ddots & \ddots\ddots & \vdots \\ \cdots & \cdots & 0 & w_s & \cdots & w_0 \\ \cdots & \cdots & \cdots & 0 & w_s\cdots & w_1 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots\cdots & \cdots 0 & w_s & w_{s-1} \\ 0 & \cdots & \cdots & \cdots & \cdots 0 & w_s \end{bmatrix}.$$

Why convolution?

- Convolution leverages three important ideas that can help improve a machine learning system:
  - **sparse interactions**,
  - **parameter sharing** and
  - **equivariant representations**.
- Sparse connectivity (weights): if there are m inputs and n outputs, then matrix multiplication based algorithms have $\mathcal{O}(mn)$ runtime (per sample). If we limit the number of connections each output may have to k, the runtime can be reduced to $\mathcal{O}(kn)$ with $k \ll m$.
- A function $f(x)$ is *equivariant* to a a function g if $f(g(x)) = g(f(x))$. Convolution with particular parameter sharing makes the layer equivariant to translation.

Sparse connectivity



Parameter sharing

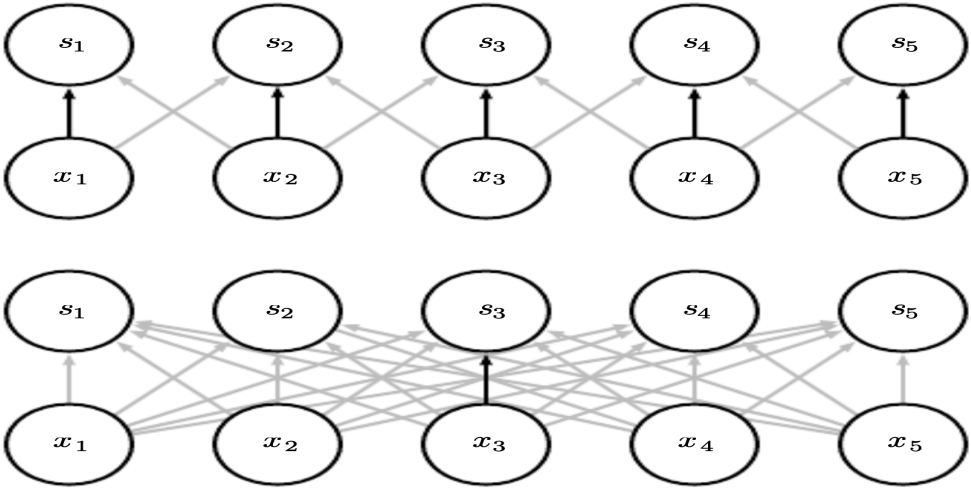
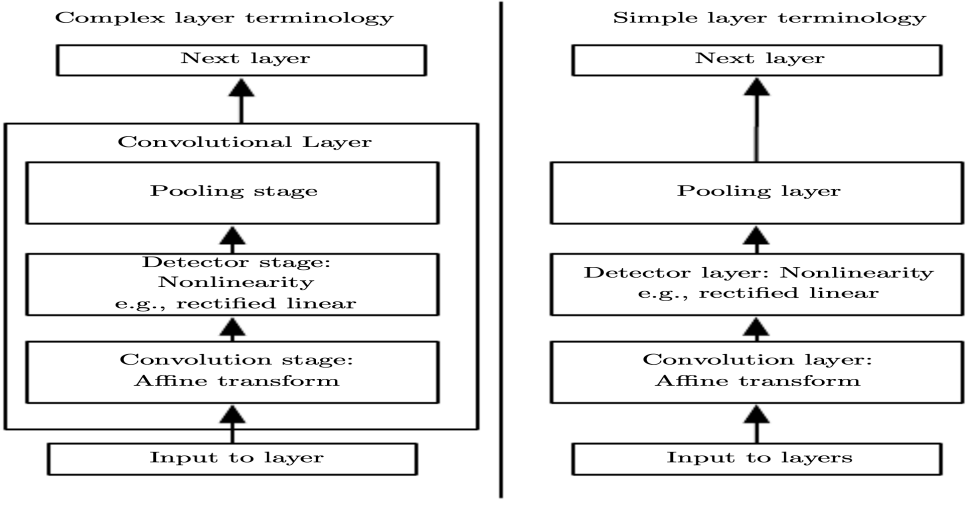


Table of Contents

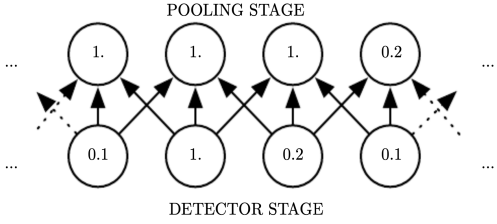
Components of CNN

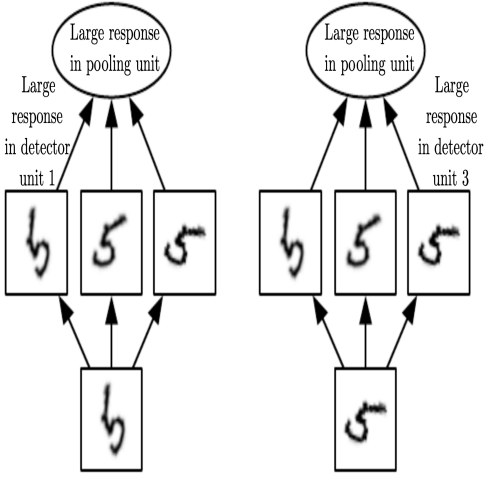


Pooling

- A 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 (Zhou and Chellappa, 1988) operation reports the maximum output within a rectangular neighborhood. Others include the average, or L^2 norm of a rectangular neighborhood.
- Pooling helps to make the representation become approximately **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


POOLING STAGE
1.
1.
1.
0.2
0.1
1.
0.2
0.1
DETECTOR STAGE
POOLING STAGE
0.3
1.
1.
1.
0.3
0.1
1.
0.2
DETECTOR STAGE


Large response in pooling unit
Large response in detector unit 1
Large response in pooling unit
Large response in detector unit 3

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.
- Because pooling summarizes the responses over a whole neighborhood, it can be used with *downsampling* by reporting every k pixels apart.
- For many tasks, pooling is essential for handling inputs of varying size. The final pooling layer may be defined to output fixed sets of summary statistics, regardless of the input (image) size.
- Theoretical work gives guidance as to which kinds of pooling one should use in various situations (Boureau et al., 2010)

Table of Contents

Variants of Basic Convolution

- CNN consists of many applications of convolution in parallel.
- The input is a grid of vector-valued observations. For example, a color image has a red, green and blue intensity at each pixel, represented as a 3-D tensor.
- Assume we have:
  - a 4-D kernel tensor $\mathbf{K}$ 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 the output and the input;
  - input data $\mathbf{V}$ with element $V_{i,j,k}$ giving the value of the input unit within channel i at row j and column k;
  - output data $\mathbf{Z}$ with the same format of $\mathbf{V}$.
- Then we have $\mathbf{Z}$ produced by convolving $\mathbf{K}$ across $\mathbf{V}$:

$$Z_{i,j,k} = \sum_{l,m,n} V_{l,j+m-1,k+n-1} K_{i,l,m,n} \tag{6}$$

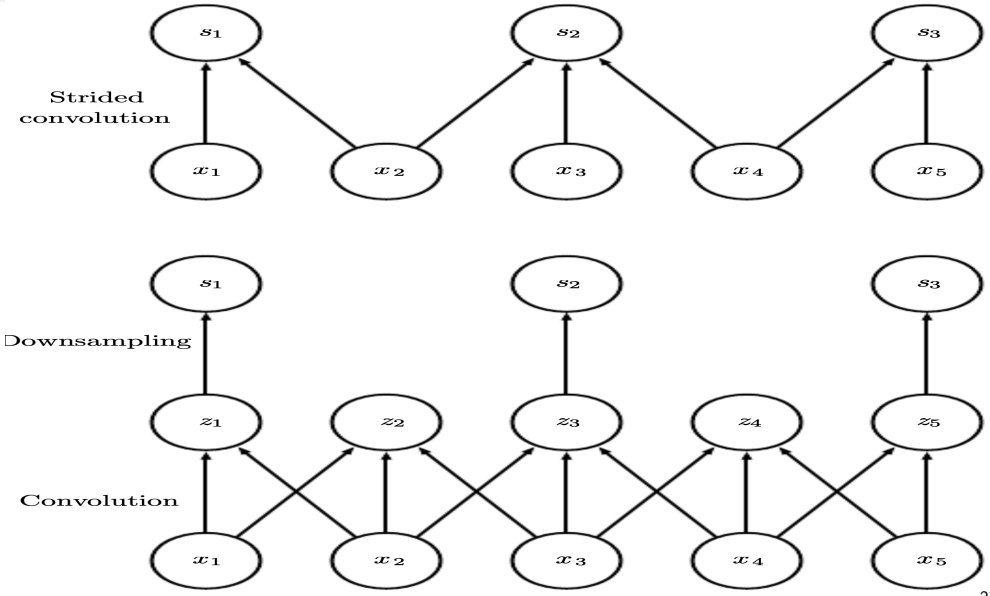
Stride

- We may want to skip over some positions of the kernel in order to reduce the computational cost.
- 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

$$Z_{i,j,k} = c(\mathbf{K}, \mathbf{V}, s)_{i,j,k} = \sum_{l,m,n} [V_{l,(j-1)\times s+m,(k-1)\times s+n} K_{i,l,m,n}] \tag{7}$$

- We refer to s as the **stride** of this downsampled convolution.

Stride


Strided convolution
s_1
s_2
s_3
x_1
x_2
x_3
x_4
x_5
Downsampling
Convolution
s_1
s_2
s_3
z_1
z_2
z_3
z_4
z_5
x_1
x_2
x_3
x_4
x_5

Zero padding

- One essential feature of CNN implementation is the ability to implicitly zero-pad the input $\mathbf{V}$ 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.
- If the input image has width m and the kernel has width k, we have (in MATLAB terminology)
  - **valid** convolution: output will be of width $m - k + 1$
  - **same** convolution: output has the same size of input
  - **full** convolution: enough zeroes are added for every pixel to be visited k times in each direction, resulting in an output image of width $m + k - 1$.
- Usually the optimal amount of zero padding (in terms of test set classification accuracy) lies somewhere between "valid" and "same" convolution.

Zero padding

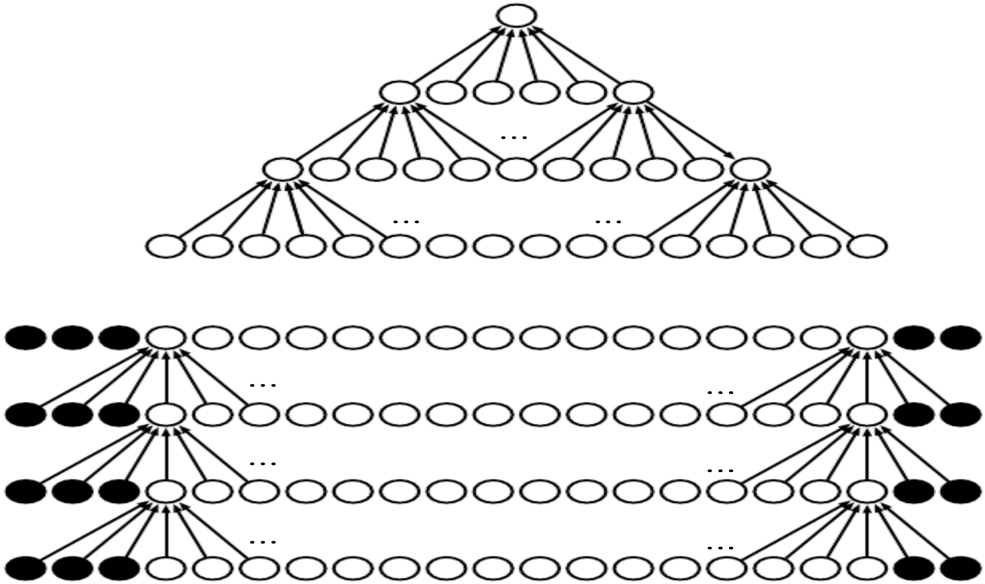


Table of Contents

TensorFlow

- In TensorFlow, we build CNN with tf.keras.layers.Conv2D for convolution and tf.keras.layers.MaxPool2D for pooling.
- tf.keras.layers.Conv2D has several arguments:
  - filters: the dimensionality of the output space
  - kernel_size: specifying the height and width of the 2D convolution window
  - strides: specifying the strides of the convolution along the height and width
  - padding: one of "valid" or "same"
  - activation: activation function to use
  - use_bias: whether the layer uses a bias vector
- tf.keras.layers.MaxPool2D has several arguments:
  - pool_size: window size over which to take the maximum
  - strides: how far the pooling window moves for each pooling step
  - padding: one of "valid" or "same"

PyTorch

- In PyTorch, we build CNN with torch.nn.Conv2d for convolution and torch.nn.MaxPool2d for pooling.
- torch.nn.Conv2d has several arguments:
  - in_channels,: number of channels in the input image
  - out_channels,: number of channels produced by the convolution
  - kernel_size: size of the convolving kernel
  - stride: stride of the convolution
  - padding: padding added to all four sides of the input
  - padding_mode: 'zeros', 'reflect', 'replicate' or 'circular'
  - bias: whether to add a learnable bias to the output
- torch.nn.MaxPool2d has several arguments:
  - kernel_size: the size of the window to take a max over
  - stride: the stride of the window
  - padding: implicit zero padding to be added on both sides

More Reading

- TensorFlow
  - https://www.datacamp.com/community/tutorials/cnn-tensorflow-python
  - https://cnvrg.io/cnn-tensorflow/
  - https://www.tensorflow.org/tutorials/images/cnn
- PyTorch
  - https://www.pyimagesearch.com/2021/07/19/pytorch-training-your-first-convolutional-neural-network-cnn/
  - https://towardsdatascience.com/pytorch-basics-how-to-train-your-neural-net-intro-to-cnn-26a14c2ea29
  - https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html