Neural Networks and Deep Learning

Convolutional Networks I - Foundations

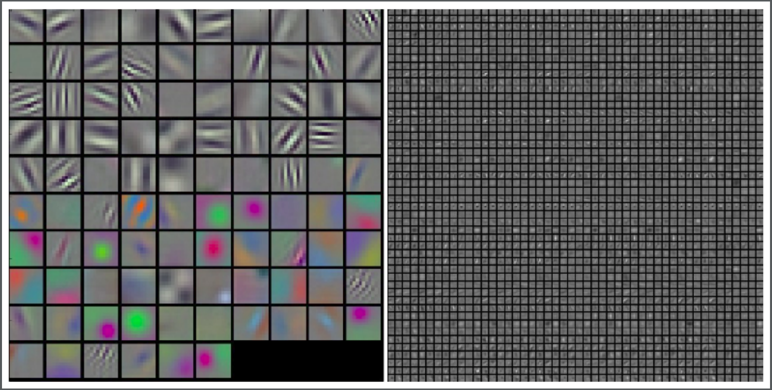
Nicholas Dronen

Department of Computer Science
dronen@colorado.edu

February 11, 2019

University of Colorado **Boulder**

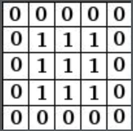
Why ConvNets?



Typical-looking filters on the first CONV layer (left), and the 2nd CONV layer (right) of a trained AlexNet. Notice that the first-layer weights are very nice and smooth, indicating nicely converged network. The color/grayscale features are clustered because the AlexNet contains two separate streams of processing, and an apparent consequence of this architecture is that one stream develops high-frequency grayscale features and the other low-frequency color features. The 2nd CONV layer weights are not as interpretable, but it is apparent that they are still smooth, well-formed, and absent of noisy patterns.
Image and Text Source: http://cs231n.github.io/understanding-cnn

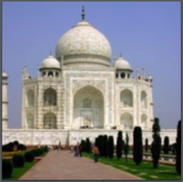
Why ConvNets?

original image

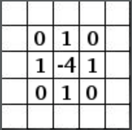


0	0	0	0	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	0	0	0	0

blur filter

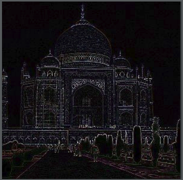

result

original image



	0	1	0	
	1	-4	1	
	0	1	0	

edge detect filter


result

Source:https://docs.gimp.org/2.6/en/plug-in-convmatrix.html

- Sparse interactions.
- Parameter sharing.
- Equivariant representation.

Discrete Convolution

- Discrete convolution is a linear transformation which preserves ordering of the data. It is sparse and reuses parameters.
- A kernel slides through the input feature map, At each location, the product between each element of the kernel and the input element it overlaps is computed and the results are summed up to obtain the output in the current location
- The procedure can be repeated using different kernels to form as many output feature maps as desired.

Discrete Convolution

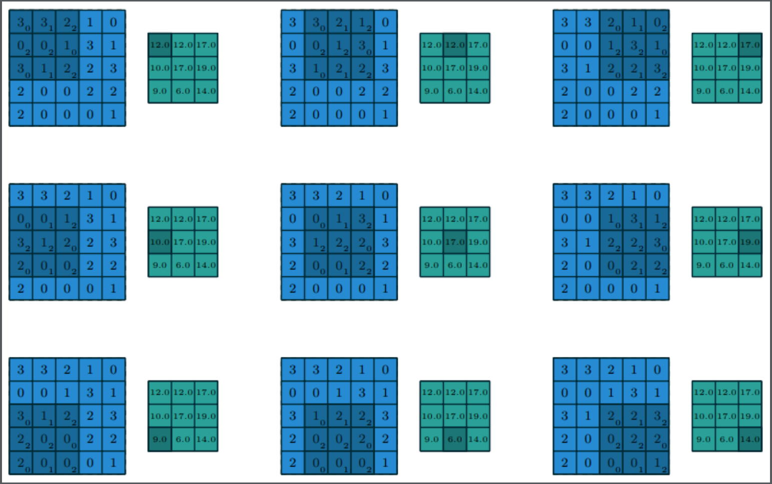


Image source: A guide to convolution arithmetic for deep learning.
blue: input feature map, *shaded blue*: kernel, *green*: output feature map.

Pooling

- Pooling operations reduce the size of feature maps by using some function to summarize subregions, such as taking the average or the maximum value.
- Pooling works by sliding a window across the input and feeding the content of the window to a pooling function.
- In some sense, pooling works very much like a discrete convolution, but replaces the linear combination described by the kernel with some other function.
- Types -
  - Max pooling
  - Average pooling

Pooling

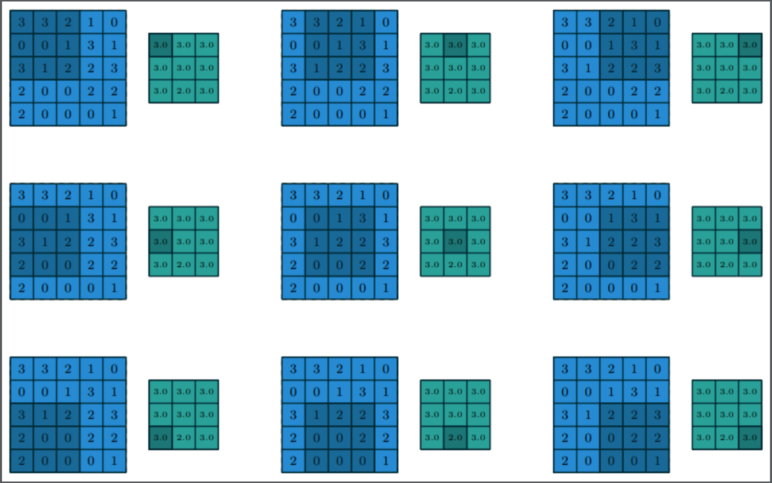


Image source: A guide to convolution arithmetic for deep learning.
blue: input feature map, *shaded blue*: max pooling kernel, *green*: output feature map.

Depth

- Depth of the output volume is a hyperparameter, it corresponds to the number of kernels/filters we would like to use, each learning to look for something different in the input.
- For example, if the first convolutional layer takes as input the raw image, then different neurons along the depth dimension may activate in presence of various oriented edges, or blobs of color

Stride

- Stride is the distance between two consecutive positions of the kernel
- Strides constitute a form of subsampling. As an alternative to being interpreted as a measure of how much the kernel is translated, strides can also be viewed as how much of the output is retained.
- For instance, moving the kernel by hops of two is equivalent to moving the kernel by hops of one but retaining only odd output elements.
- Increasing the stride will produce smaller output volumes spatially.

Padding

- Zero padding is concatenating zeros around the border for convenience.
- The nice feature of zero padding is that it will allow us to control the spatial size of the output volumes.
- Types
  - » Padding schemes: valid, half/same, full
  - » Padding Values
    - Zero
    - Reflection
    - Replication
    - Constant

Output Size

The following properties affect the output size o_j of a *conv* layer along axis j -

i_j : input size along axis j

k_j : kernel size along axis j

s_j : stride along axis j

p_j : zero-padding along along axis j

The following properties affect the output size o_j of a *pooling* layer along axis j -

i_j : input size along axis j

k_j : kernel size along axis j

s_j : stride along axis j

Output Size

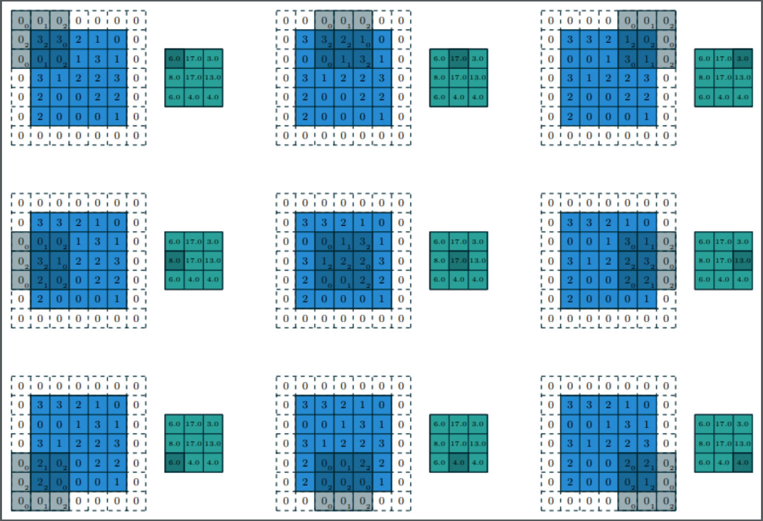


Image source: A guide to convolution arithmetic for deep learning.
$N = 2$, $i_1 = i_2 = 5$, $k_1 = k_2 = 3$, $s_1 = s_2 = 2$, $p_1 = p_2 = 1$

Setup

Simplified setting

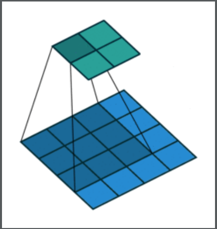
- 2D discrete convolutions ($N = 2$)
- Square inputs ($i_1 = i_2 = i$)
- Square kernel size ($k_1 = k_2 = k$)
- Same stride along both axes ($s_1 = s_2 = s$)
- Same zero padding along both axes ($p_1 = p_2 = p$)

No zero padding, unit strides

- In this case the kernel just slides across every position of the input (i.e. $s = 1$ and $p = 0$).

For any i, k and $s = 1$,

$$o = (i - k) + 1$$



$i = 4, k = 3, s = 1, p = 0$

Zero padding, unit strides

- Padding with p zeros changes the effective input size from i to $i + 2p$.

For any i, k, p, $s = 1$,

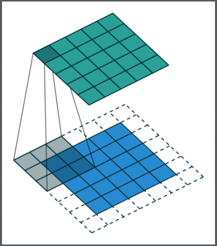
$$o = (i - k) + 2p + 1$$

Half (or "same") padding

- Having the output size be the same as the input size (i.e., o = i) can be a desirable property.

For any i and odd k ($k = 2n + 1,\ n \in N$), $s = 1$ and $p = \lfloor k/2 \rfloor = n$,

$$o = i + 2\lfloor k/2 \rfloor - (k - 1) = i$$



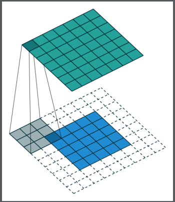
$i = 5, k = 3, s = 1, p = 1$

Full padding

- While convolving a kernel generally decreases the output size with respect to the input size, sometimes the opposite is required. This can be achieved with proper zero padding.

For any i and k, $s = 1$ and for $p = k - 1$, $s = 1$,

$$o = i + 2(k-1) - (k-1) = i + (k-1)$$



$i = 5, k = 3, s = 1, p = 2$

No zero padding, non-unit strides

- In this case, the output size can be defined in terms of the number of possible placements of the kernel on the input.
- The kernel starts as usual on the leftmost part of the input, but this time it slides by steps of size s until it touches the right side of the input.
- The size of the output is equal to the number of steps made, plus one, accounting for the initial position of the kernel.
- The floor function accounts for the fact that sometimes the last possible step does not coincide with the kernel reaching the end of the input, i.e., some input units are left out.

For any i, k, s and for $p = 0$,

$$o = \left\lfloor \frac{i-k}{s} \right\rfloor + 1$$

Zero padding, non-unit strides

- The most general case, can be derived from the previous cases.

For any i, k, s and p,

$$o = \left\lfloor \frac{i + 2p - k}{s} \right\rfloor + 1$$

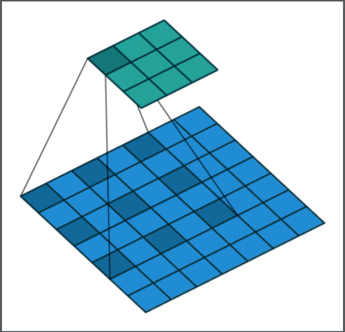
Transposed Convolution

- Transposed convolutions – also called fractionally strided convolutions or deconvolutions – work by swapping the forward and backward passes of a convolution.
- The simplest way to think about a transposed convolution on a given input is to imagine such an input as being the result of a direct convolution applied on some initial feature map. The transposed convolution can be then considered as the operation that allows to recover the shape of this initial feature map.

Dilated Convolution

- Dilated convolutions "inflate" the kernel by inserting spaces between the kernel elements. The dilation "rate" is controlled by an additional hyper-parameter d.
- Implementations may vary, but there are usually d—1 spaces inserted between kernel elements such that $d = 1$ corresponds to a regular convolution.
- Dilated convolutions cheaply increase the receptive field of output units without increasing the kernel size

Dilated Convolution



$$i = 7, k = 3, d = 2, s = 1, p = 0$$

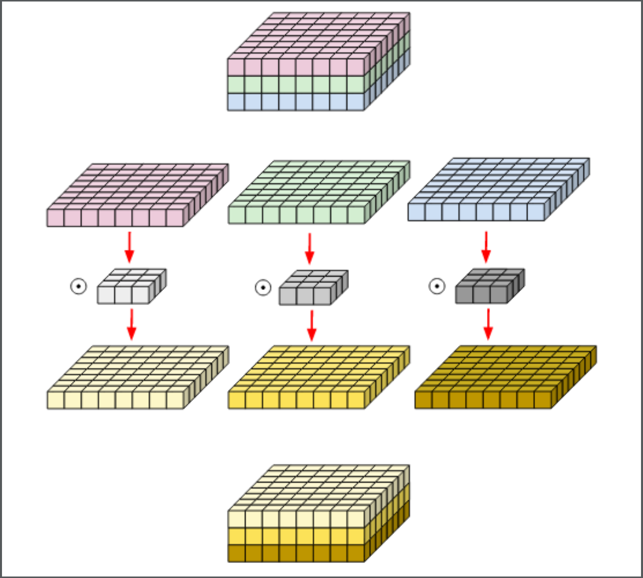
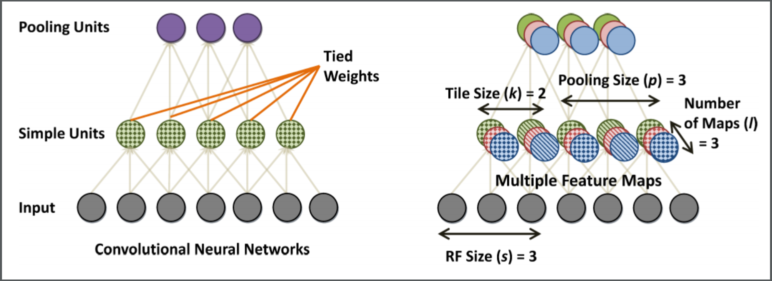


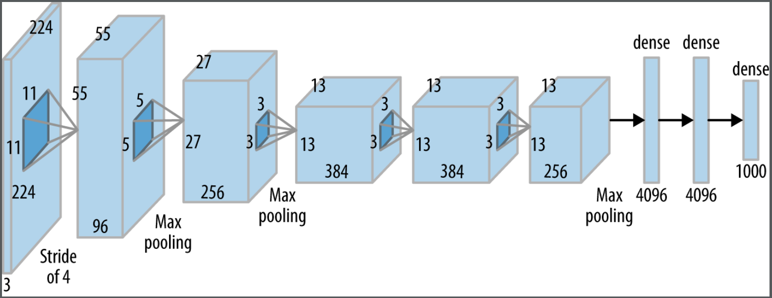
Image Source: https://eli.thegreenplace.net

What?



Left: Convolutional Neural Networks with local receptive fields and tied weights.
Right: Partially untied local receptive field networks – Tiled CNNs.
Image Source: Tiled convolutional neural networks.

Encoder



AlexNet CNN Architecture
Image source: https://www.oreilly.com/library/view/tensorflow-for-deep/9781491980446/ch01.html.

GAN

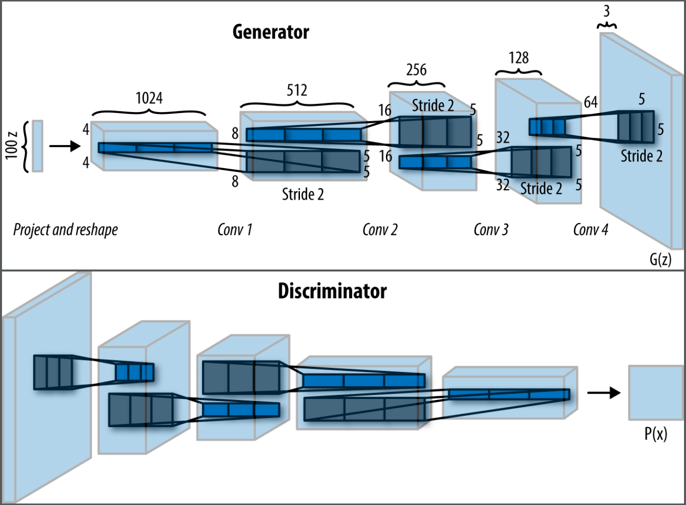


Image source:
https://www.oreilly.com/ideas/deep-convolutional-generative-adversarial-networks-with-tensorflow

Encoder-Decoder

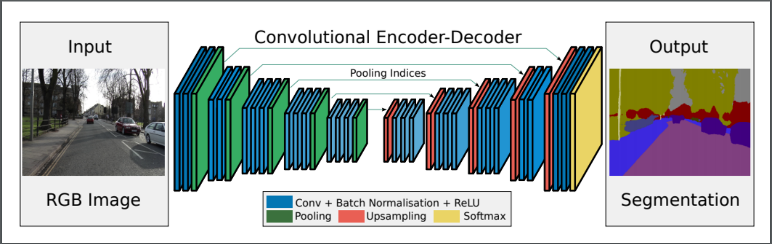


Image source: SegNet - A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation.