3a: Convolution

Week 3: Overview

This week we will focus on image processing, with particular emphasis on object classification. We will first describe convolutional neural networks, and then look at various methods which have been introduced to allow progressively deeper networks to be trained by backpropagation, including weight initialisation, batch normalisation, residual networks and dense networks. We will also discuss how the content of one image can be combined with the style of another, using neural style transfer.

Weekly learning outcomes

By the end of this week, you will be able to:

- describe convolutional networks, including convolution operator, stride, padding, max pooling
- describe how to train image classification networks
- identify the limitations of 2-layer neural networks
- explain the problem of vanishing or exploding gradients, and how it can be solved using techniques such as weight initialisation, batch normalization, skip connections, dense blocks
- describe the process of neural style transfer

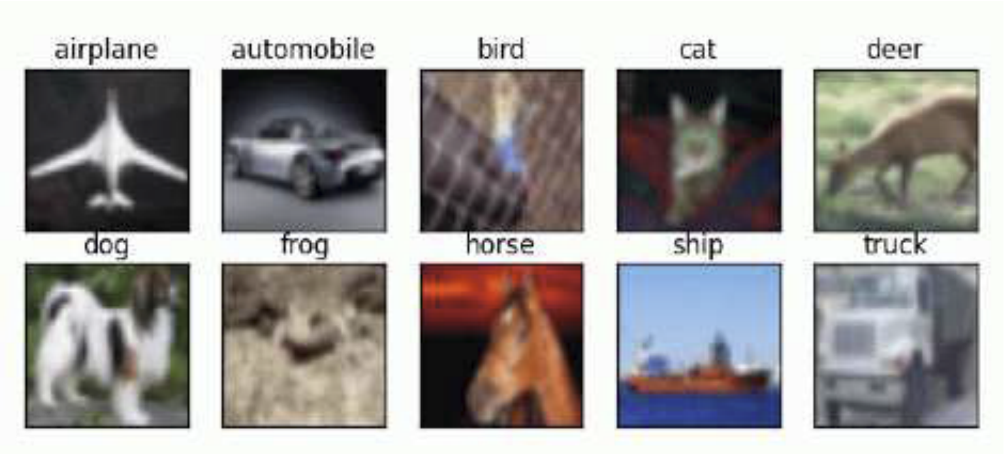
Image Classification

Ever since Rosenblatt trained his first Perceptron in 1957, image classification has played a pivotal role in driving technological advances in neural networks and deep learning. Three of the most commonly used image classification datasets are MNIST, CIFAR-10 and ImageNet.

MNIST Dataset

The MNIST dataset of handwritten digits was released in 1998, consisting of 50,000 training images and 10,000 test images. The images are black and white, at a resolution of 28×28 pixels.

CIFAR-10 Dataset



The CIFAR-10 dataset, introduced in 2009, consists of 60,000 colour images at resolution 32×32 in ten different categories.

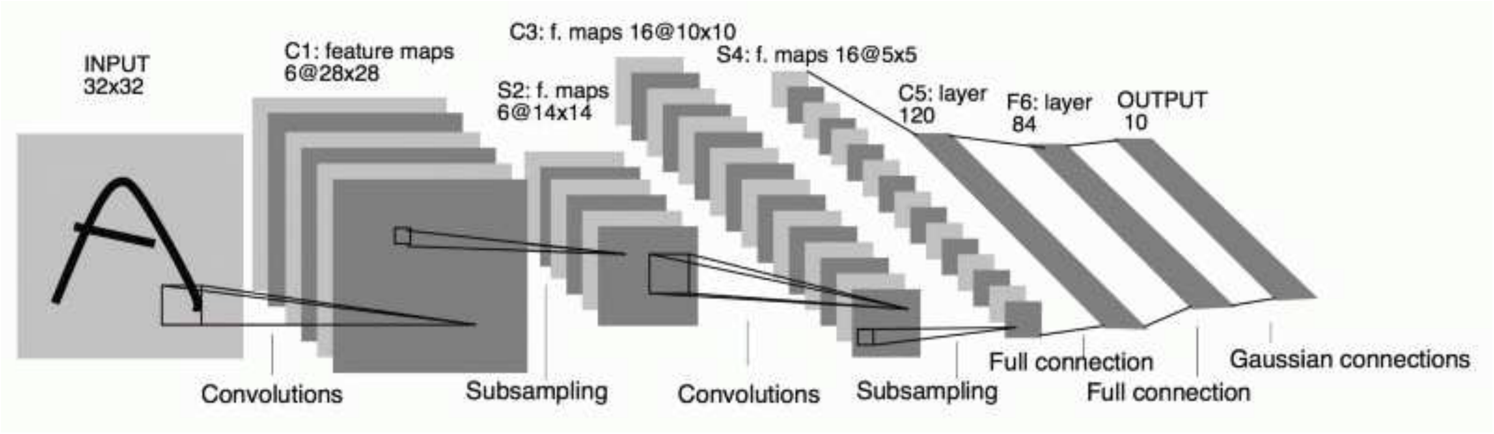
ImageNet Dataset

The ImageNet Dataset consists of 1.2 million images in 1000 different categories, and became the basis for the ImageNet LSVRC competition.

In 2012 a big advance was made in prediction accuracy on the ImageNet dataset, using an 8-layer convolutional neural network called AlexNet. Subsequent enhancements allowed even deeper networks to be trained and achieve even higher accuracy.

Convolutional Networks

Suppose we want to classify an image as a bird, sunset, dog, cat, etc. If we can identify features such as feather, eye, or beak which provide useful information in one part of the image, then those features are likely to also be relevant in another part of the image. We can exploit this regularity by using a **convolutional layer**, which applies the same weights to different parts of the image.



[image source: LeCun, 1998]

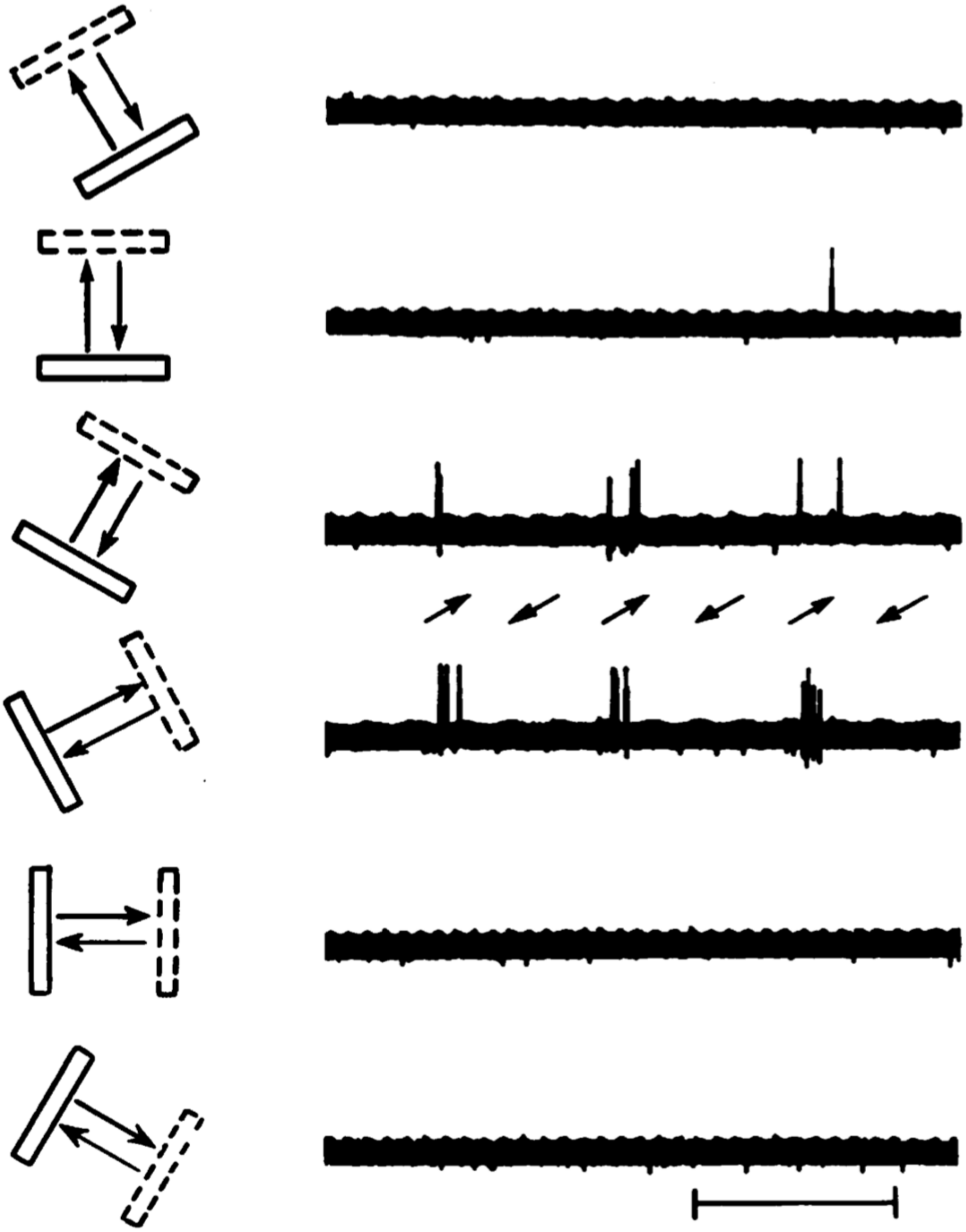
Convolutional Neural Networks generally consist of the following components:

- **convolutional layers:** extract shift-invariant features from the previous layer
- **subsampling or pooling layers:** combine the activations from multiple units in the previous layer into one unit
- **fully connected layers:** collect spatially diffuse information
- **output layer:** choose between classes

There can be multiple steps of convolution followed by subsampling, before reaching the fully connected layers. Note how subsampling reduces the size of the feature map (typically, by about half in each direction). For example, in the LeNet architecture shown above, the 5×5 window of the first convolution layer extracts from the original 32×32 image a 28×28 array of features. Subsampling then halves this size to 14×14. The second convolution layer uses another 5×5 window to extract a 10×10 array of features, which the second subsampling layer reduces to 5×5. These activations then pass through two fully connected layers into the 10 output units corresponding to the digits ’0’ to ’9’.

Visual cortex

The inspiration for convolutional neural networks comes in part from studies of the primary and secondary visual cortex by Hubel and Wiesel. They found that cells in the primary visual cortex detect low-level features such as a line at a specific angle, or a line at a specific angle moving in a particular direction, while cells in the secondary visual cortex respond to more sophisticated visual features.



[image source: Hubel & Wiesel, 1959]

The advent of massively parallel General Purpose Graphics Programming Units (GPGPU's) has made it possible to simulate convolutional networks on a large scale.

Optional video

References

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P.,1998. Gradient-based learning applied to document recognition, *Proceedings of the IEEE*, 86(11), 2278 - 2324.

Hubel, D.H. & Wiesel, T.N., 1959. Receptive fields of single neurones in the cat's striate cortex, *Journal of Physiology*, 148(3), 574-591.

Convolution in Detail

Convolution operator

Convolutional layers are an adaptation of the convolution operator commonly used in mathematical analysis as well as image and signal processing.

Continuous convolution

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

Discrete convolution

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

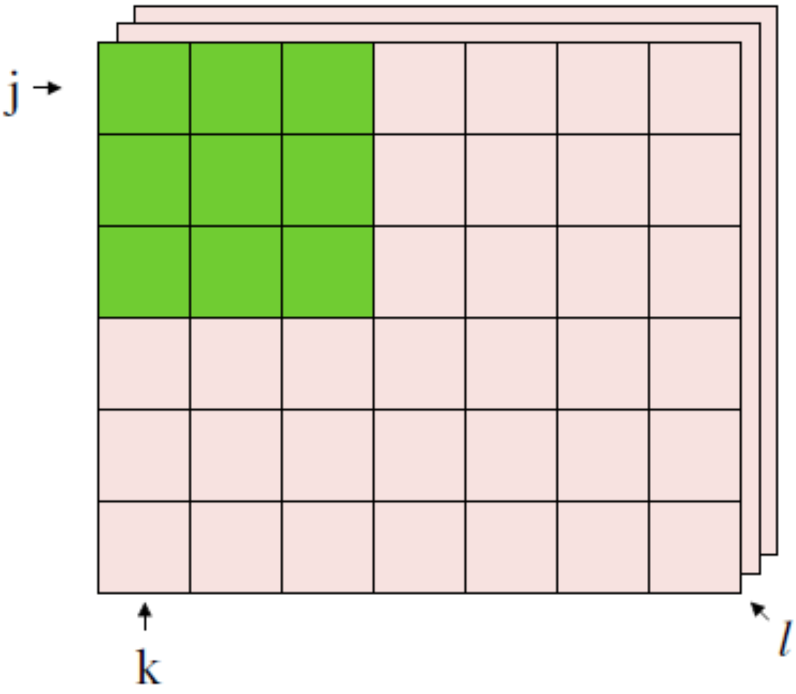
Two-dimensional convolution

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

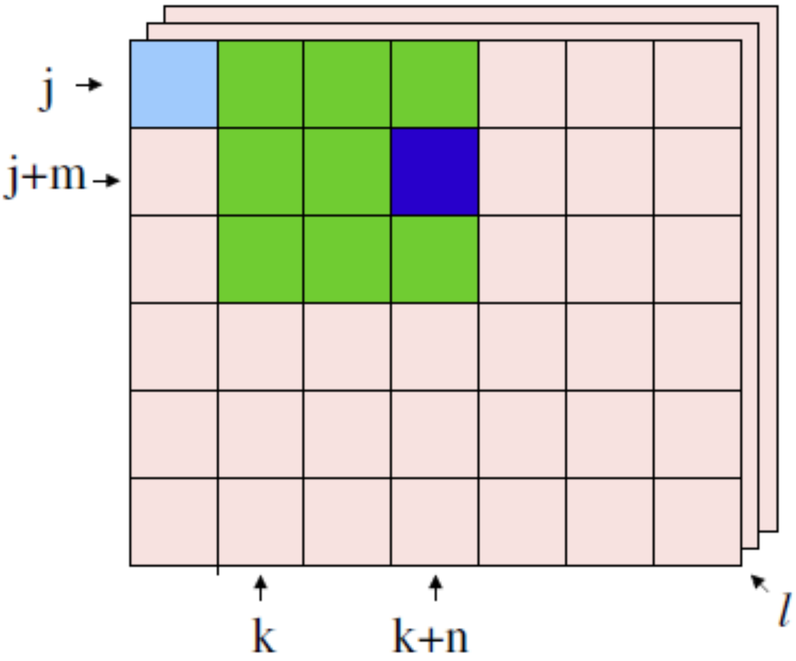
Note: Theoreticians sometimes write $I(j-m,k-n)$ in the above formula so that the operator is commutative. But, computationally, it is more convenient to write it with a plus sign.

Convolutional Network Layer

Assume the network is processing an image of size $J \times K$, with L **channels** (for example, there could be three channels corresponding to the colours Red, Green, Blue). The intensity in channel l of the pixel at location (j,k) is $V_{j,k}^{l}$.

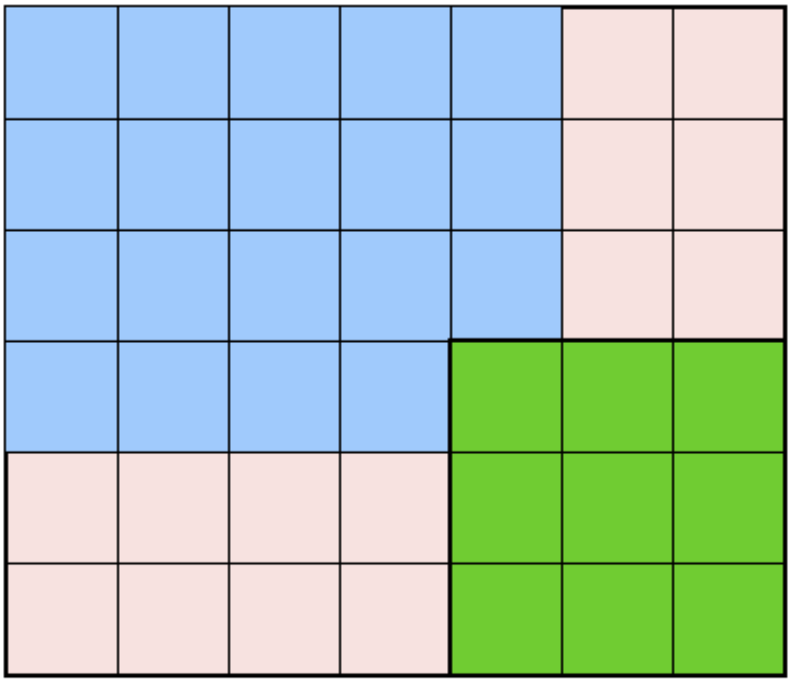


We apply an $M \times N$ **filter** to these inputs, in order to compute the activation of one hidden unit $Z^i_{j,k}$ in filter i of the convolution layer. In this example $J = 6, K = 7, L = 3, M = 3, N = 3$.



$$Z^i_{j,k} = g\left(b^i + \sum_l \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} K^i_{l,m,n} V^l_{j+m,k+n}\right)$$

The same weights $K^i_{l,m,n}$ and bias b^i are applied to the next $M \times N$ block of inputs, to compute the next hidden unit in the convolution layer, in a process known as **weight sharing**.



If the original image size is $J \times K$ and the filter size is $M \times N$, the size of the convolution layer will be $(J+1-M) \times (K+1-N)$

Transcript

Example: LeNet

For example, in the first convolutional layer of LeNet, $J = K = 32, M = N = 5$.

The width of the next layer is

$$J + 1 - M = 32 + 1 - 5 = 28$$

Question: If there are 6 filters in this layer, and the network is applied to black-and-white images, compute the number of:

- weights per filter
- neurons
- connections
- independent parameters

Answers:

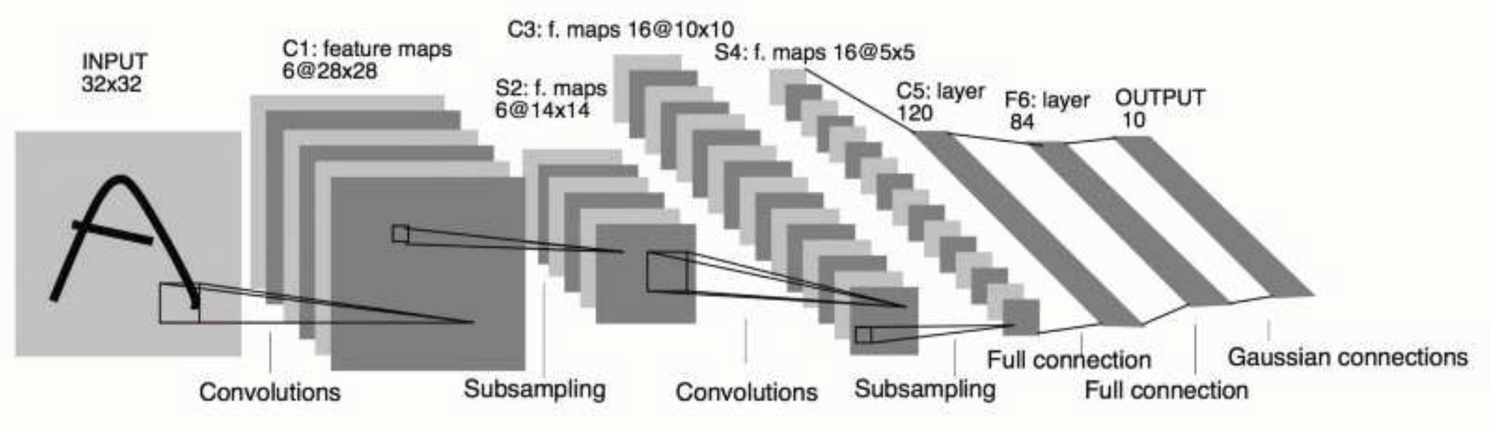
- weights per filter? 1 + 5 × 5 × 1 = 26
- neurons? 28 × 28 × 6 = 4704
- connections? 28 × 28 × 6 × 26 = 122304
- independent parameters? 6 × 26 = 156

Further Reading

Textbook Deep Learning (Goodfellow, Bengio, Courville, 2016):

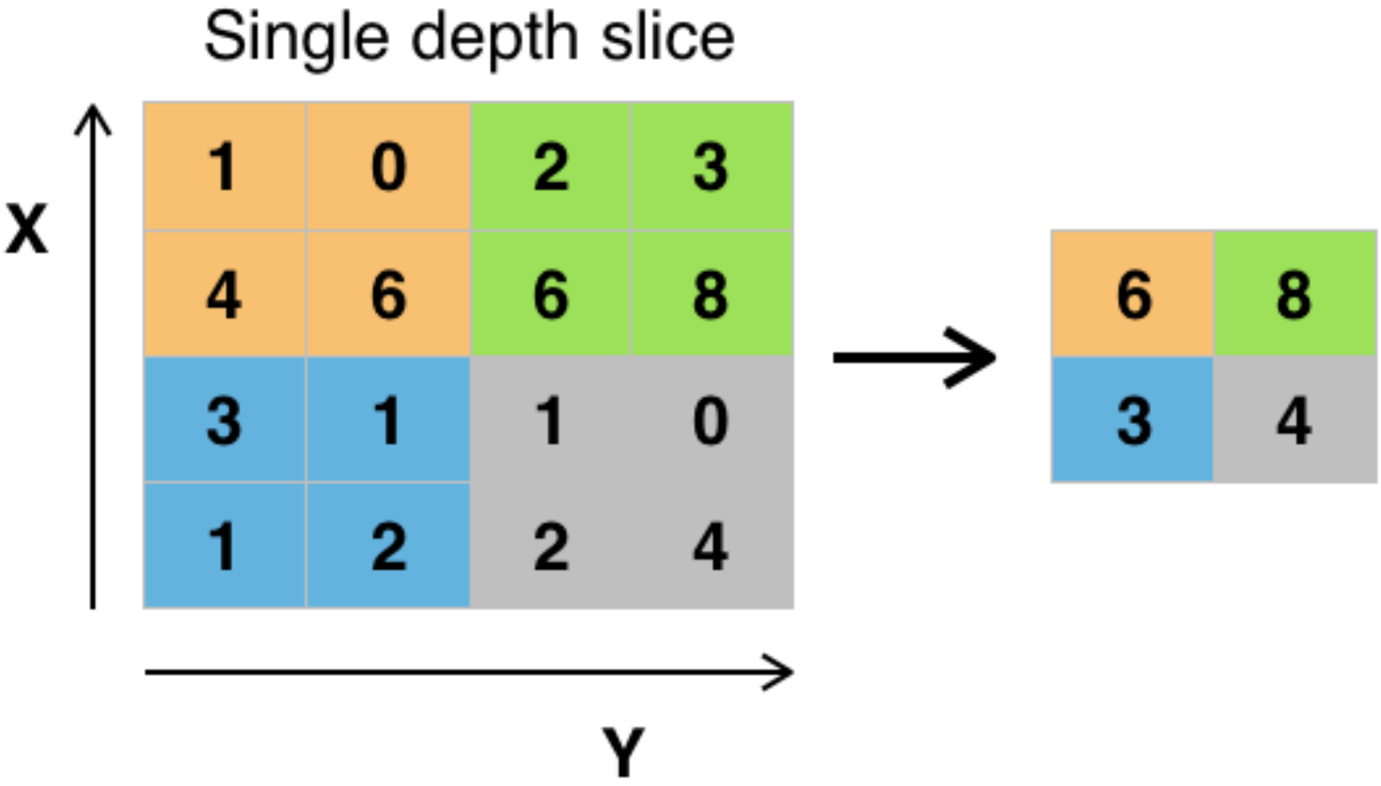
- Convolutional Networks (7.9)
- Convolution Operator (9.1-9.2)

Pooling and Padding



Max Pooling

One common form of subsampling is **max pooling**. The previous layer is divided into small groups of units (typically, in a 2×2 grid) and the maximum activation among each small group of units is copied to a single unit in the subsequent layer.

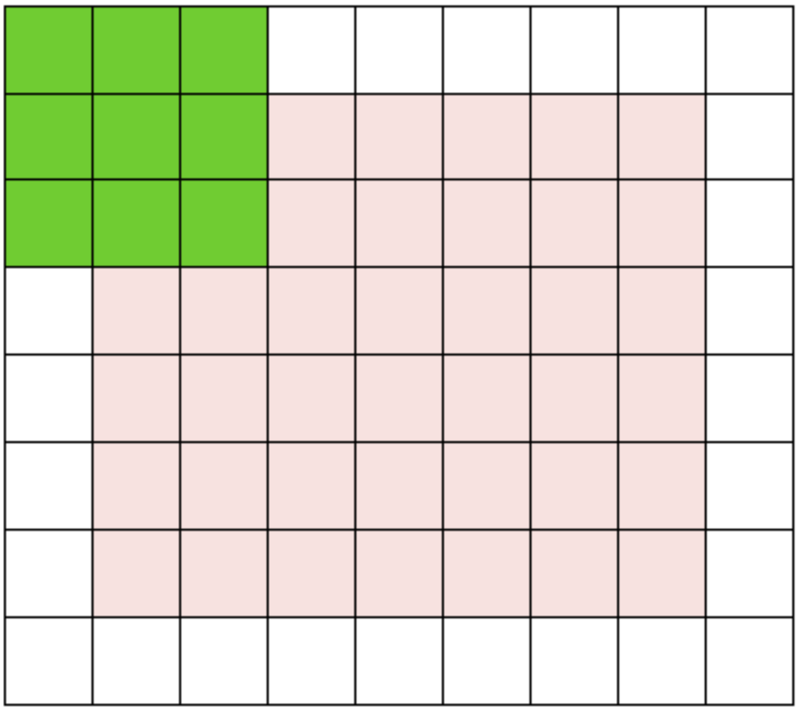


1	0	2	3
4	6	6	8
3	1	1	0
1	2	2	4

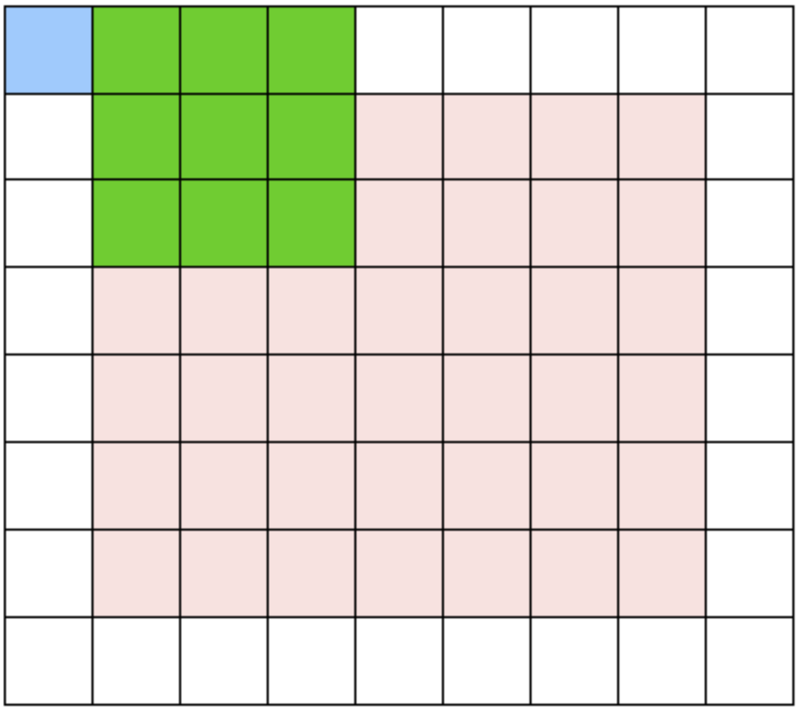
6	8
3	4

[image source]

Zero Padding



Sometimes, we expand the size of the input (or the previous layer) but treat the newly added units as having a fixed value (typically, zero).



For example, with a 3×3 filter and one extra row of zero padding on each side, the subsequent layer will be the same size as the previous layer (or the input).

Optional video

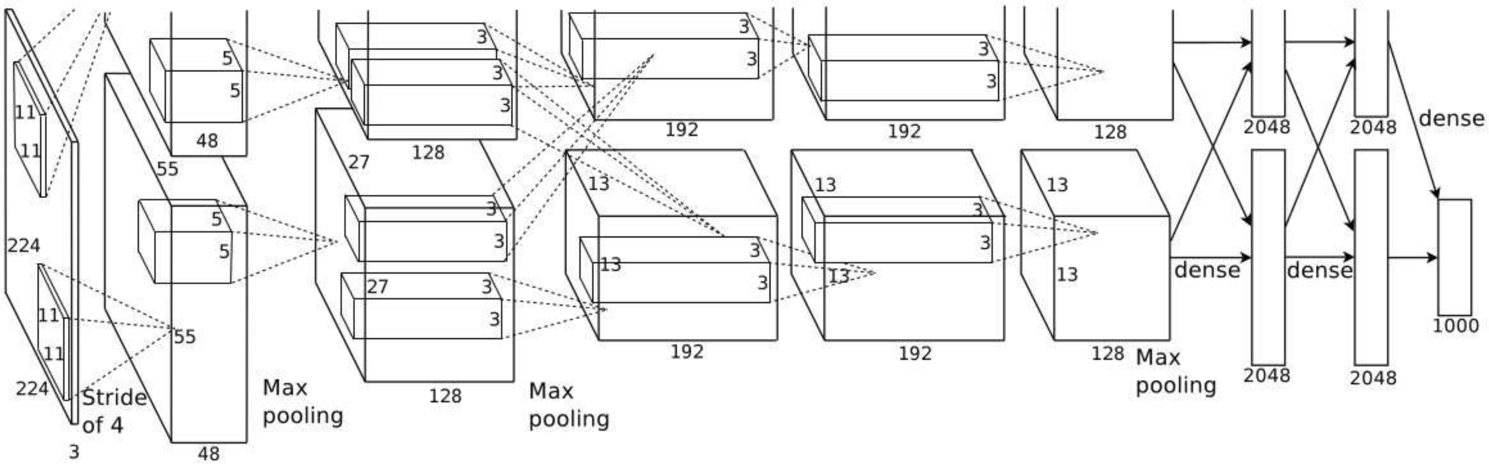
Further reading

Textbook Deep Learning (Goodfellow, Bengio, Courville, 2016):

- Max Pooling (9.3-9.4) and Stride (9.5)

Stride and Data Augmentation

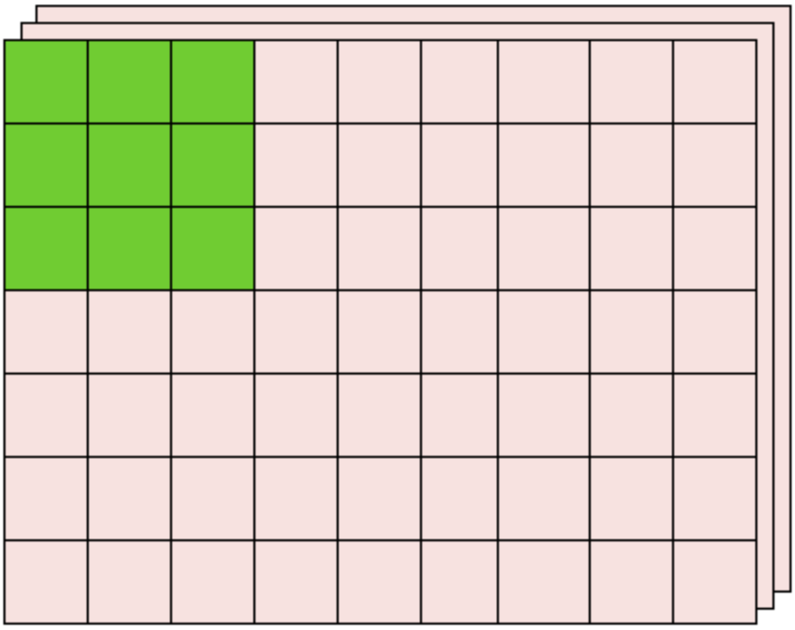
Example: AlexNet



Here is a summary of AlexNet (Krizhevsky, 2012)

- 5 convolutional layers plus 3 fully connected layers.
- Softmax with 1000 classes applied at the output layer.
- Two GPUs, which interact only at certain layers.
- Max pooling of 3×3 neighbourhoods with stride 2.
- 50% dropout at the fully connected layers.
- Data augmentation by randomly cropping as well as transforms on RGB space.

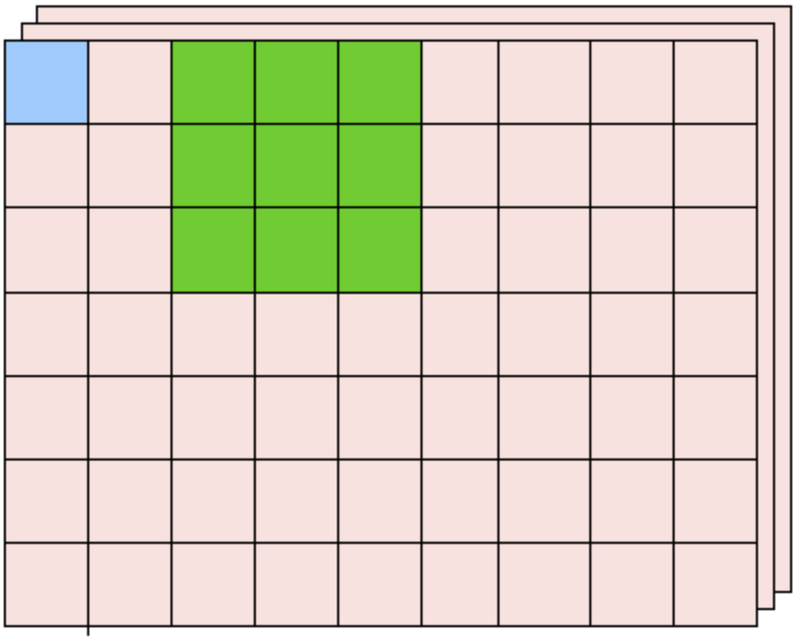
Stride



Assume the original image is $J \times K$, with L channels.

We again apply an $M \times N$ filter, but this time with a **stride** of $s > 1$.

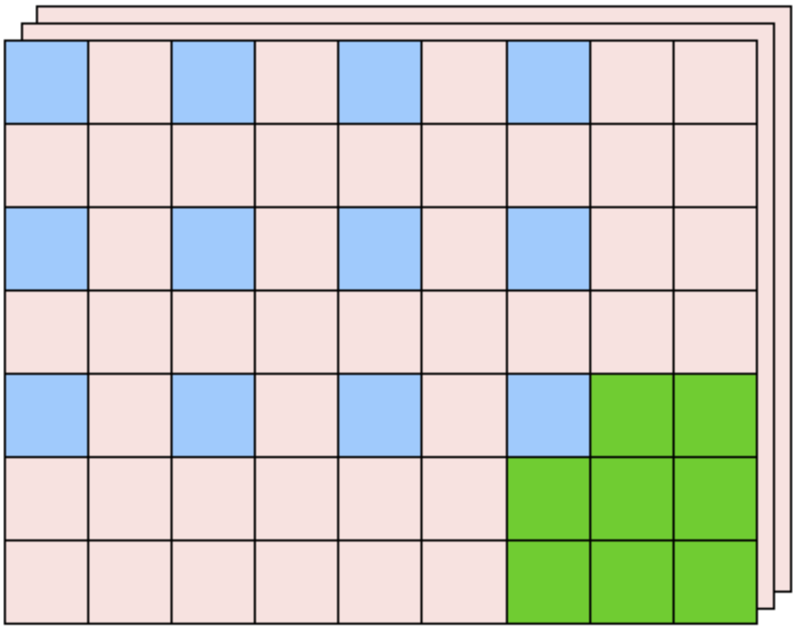
In this example $J = 7, K = 9, L = 3, M = 3, N = 3, s = 2$.



$$Z^i_{j,k} = g\left(b^i + \sum_l \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} K^i_{l,m,n} V^l_{j+m,k+n}\right)$$

The same formula is used, but j and k are now incremented by s each time, so activations are only computed for values of j, k with $j \bmod s = 0$ and $k \bmod s = 0$. The number of free parameters for each filter is $1 + L \times M \times N$.

Stride Dimensions

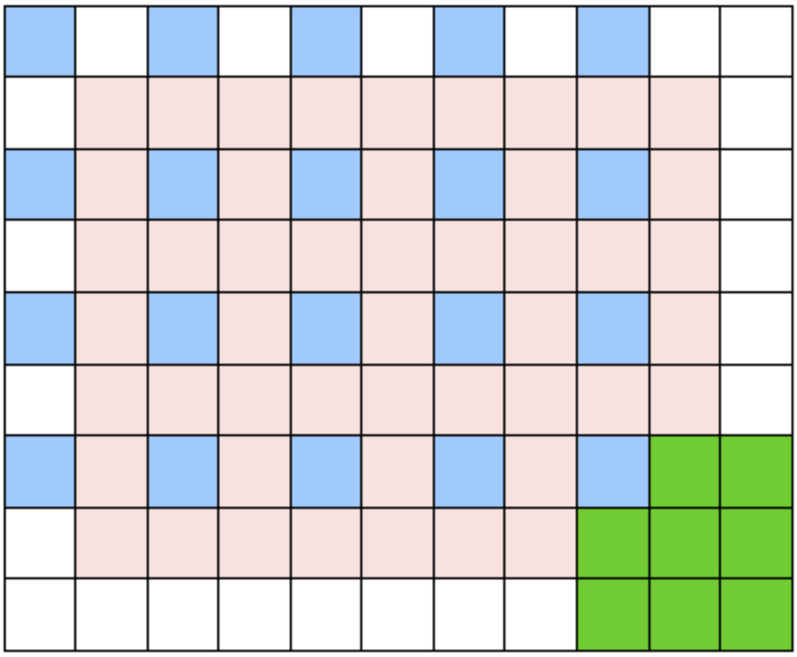


j takes on the values $0, s, 2s, ..., (J - M)$.

k takes on the values $0, s, 2s, ..., (K - N)$.

The next layer is $(1 + (J - M)/s)$ by $(1 + (K - N)/s)$.

Stride with Zero Padding



When combined with zero padding of width P,

j takes on the values $0, s, 2s, ..., (J + 2P - M)$.

k takes on the values $0, s, 2s, ..., (K + 2P - N)$. The size of the next layer is $(1 + (J + 2P -$

$M)/s)$ by $(1+(K+2P-N)/s)$.

Example: AlexNet Conv Layer 1

In the first convolutional layer of AlexNet, $J=K=224, P=2, M=N=11, s=4$. The width of the next layer is:

$$1+(J+2P-M)/s = 1+(224+2\times 2-11)/4 = 55.$$

Question: If there are 96 filters in this layer, compute the number of:

- weights per neuron
- neurons
- connections
- independent parameters

Answers:

- weights per neuron? 1 + 11 × 11 × 3 = 364
- neurons? 55 × 55 × 96 = 290,400
- connections? 55 × 55 × 96 × 364 = 105,705,600
- independent parameters? 96 × 364 = 34,944

Overlapping Pooling

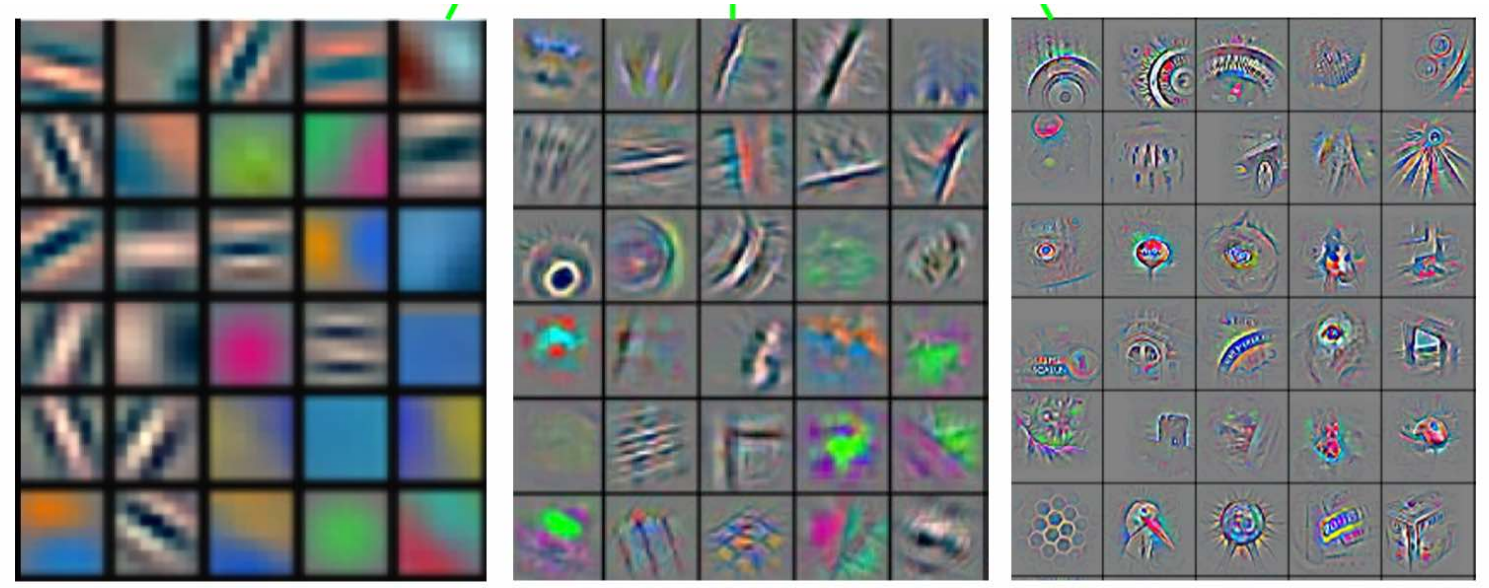
If the previous layer is $J \times K$, and max pooling is applied with width F and stride s, the size of the next layer will be:

$$(1+(J-F)/s) \times (1+(K-F)/s).$$

For example, in the first convolutional layer of AlexNet $F=3$ and $s=2$, thus reducing the size of the layer from 55×55 to 27×27. Note that in this case the 3×3 neighbourhoods overlap, so that the activation of a single unit in the previous layer could get copied to two distinct units in the following layer.

Convolutional Filters

Each unit in a convolutional layer can be visualised by constructing the sub-image that would cause that unit to be most strongly activated. Generally, the units in the first convolutional layer respond to lines and boundaries in a similar way to Gabor filters, or to the neurons in the primary visual cortex studied by Hubel and Wiesel. As we move to deeper convolutional layers, the units extract progressively more abstracted and high-level features.



First Layer Second Layer Third Layer

[image source]

Optional video

Data Augmentation

- patches of size 224×224 are randomly cropped from the original images
- images can be reflected horizontally
- also include changes in intensity of RGB channels
- at test time, average the prediction of 10 different crops of each test image

Optional video

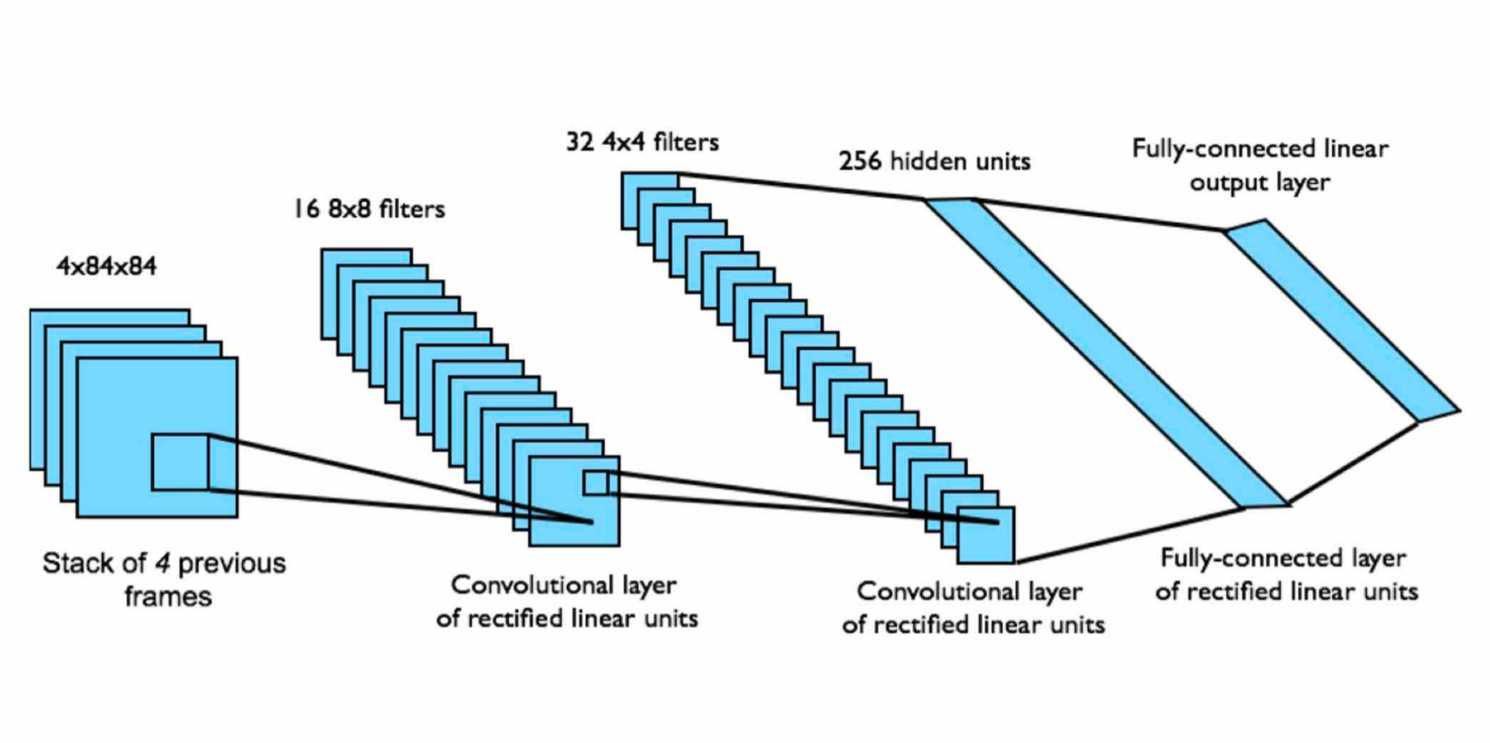
References

Krizhevsky, A., Sutskever, I., & Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, *25*, 1097-1105.

Exercise: Convolution

Exercise - Convolution

One of the early papers on Deep Q-Learning for Atari games (Mnih et al, 2013) contains this description of its Convolutional Neural Network:


4x84x84
16 8x8 filters
32 4x4 filters
256 hidden units
Fully-connected linear output layer
Stack of 4 previous frames
Convolutional layer of rectified linear units
Convolutional layer of rectified linear units
Fully-connected layer of rectified linear units

"The input to the neural network consists of an 84 × 84 × 4 image. The first hidden layer convolves 16 8 × 8 filters with stride 4 with the input image and applies a rectifier nonlinearity. The second hidden layer convolves 32 4 × 4 filters with stride 2, again followed by a rectifier nonlinearity. The final hidden layer is fully-connected and consists of 256 rectifier units. The output layer is a fully-connected linear layer with a single output for each valid action. The number of valid actions varied between 4 and 18 on the games we considered."

For each layer in this network, compute the number of:

1. weights per neutron (or filter) in this layer (including bias)
2. width and height of layer (only for convolutional layers)
3. neurons in this layer
4. connections into the neurons in this layer
5. independent parameters in this layer

You should assume the input images are grayscale, there is no padding, and there are 18 valid actions (outputs).

Question 1 *Submitted Sep 17th 2022 at 2:37:07 pm*

Compute each of the above for the first convolutional layer.

$J = K = 84, L = 4, M = N = 8, P = 0, s = 4$

1. weights per filter $= 1 + M \times N \times L = 1 + 8 \times 8 \times 4 = 257$
2. width and height $= 1 + (J - M)/s = 1 + (84 - 8)/4 = 20$
3. number of neurons $= 20 \times 20 \times 16 = 6400$
4. connections $= 20 \times 20 \times 16 \times 257 = 1644800$
5. independent parameters $= 16 \times 257 = 4112$

Question 2 *Submitted Sep 17th 2022 at 2:38:13 pm*

Compute each of the above for the second convolutional Layer.

$J = K = 20, L = 16, M = N = 4, P = 0, s = 2$

1. weights per filter $= 1 + M \times N \times L = 1 + 4 \times 4 \times 16 = 257$
2. width and height $= 1 + (J - M)/s = 1 + (20 - 4)/2 = 9$
3. number of neurons $= 9 \times 9 \times 32 = 2592$
4. connections $= 9 \times 9 \times 32 \times 257 = 666144$
5. independent parameters $= 32 \times 257 = 8224$

Question 3 *Submitted Sep 17th 2022 at 2:39:15 pm*

Compute each of the above for the fully connected layer.

1. weights per neuron $= 1 + 2592 = 2593$
2. width and height (not applicable)
3. number of neurons 256
4. connections $= 256 \times 2593 = 663808$
5. independent parameters $= 663808$

Question 4 *Submitted Sep 17th 2022 at 2:40:11 pm*

Compute each of the above, for the output layer.

1. weights per neuron $= 1 + 256 = 257$
2. width and height (not applicable)
3. number of neurons 18
4. connections $= 18 \times 257 = 4626$
5. independent parameters $= 4626$

Revision 4: Convolutional Networks

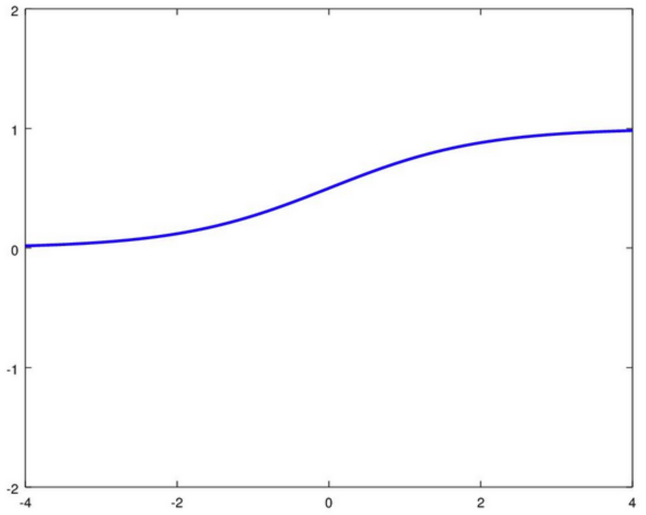
This is a revision quiz to test your understanding of the material from Week 3 on Convolutional Networks.

You must attempt to answer each question yourself, before looking at the sample answer.

Question 1 *Submitted Sep 17th 2022 at 2:43:49 pm*

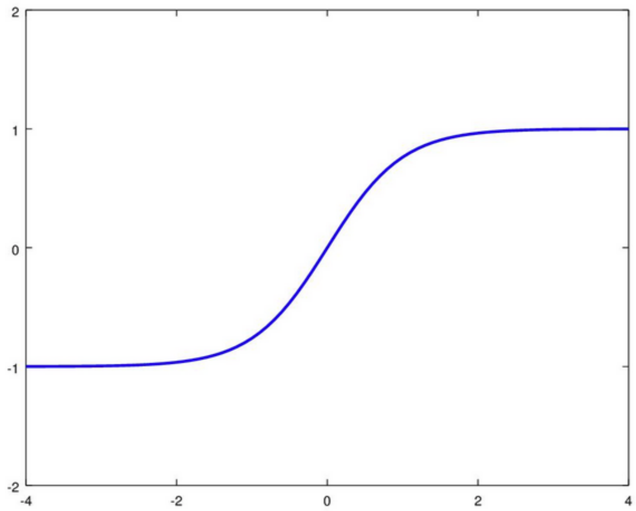
Sketch the following activation functions, and write their formula: Sigmoid, Tanh, ReLU.

Sigmoid:



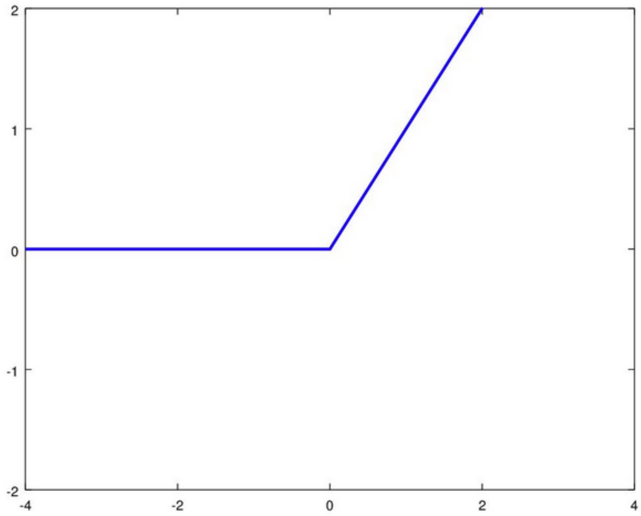
$$y = \frac{1}{1+e^{-x}}$$

Tanh:



$$y \;=\; \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

ReLU:



$$y = 0,\;\; \text{if}\, x \;<\; 0;\;\; y = x,\; \text{if}\, x \geq 0.$$

Question 2 *Submitted Sep 17th 2022 at 2:45:06 pm*

Write the formula for activation $Z^i_{j,k}$ of the node at location (j,k) in the ith filter of a Convolutional Neural Network which is connected by weights $K^i_{l,m,n}$ to all nodes in an $M \times N$ window from the L filters (or channels) in the previous layer, assuming bias weights are included and the activation function is $g()$. How many free parameters would there be in this layer?

$$Z^i_{j,k} = g\left(b^i + \sum_l \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} K^i_{l,m,n} V^l_{j+m,\,k+n}\right)$$

The number of free parameters is: $F \times (1 + L \times M \times N)$ where F is the number of filters in this layer.

Question 3 *Submitted Sep 17th 2022 at 2:46:07 pm*

If the previous layer has size $J \times K$, and a filter of size $M \times N$ is applied with stride s and zero-padding of width P, what will be the size of the resulting convolutional layer?

$(1 + (J + 2P - M)/s) \times (1 + (K + 2P - N)/s)$

Question 4 *Submitted Sep 17th 2022 at 2:47:04 pm*

If max pooling with filter size F and stride s is applied to a layer of size $J \times K$, what will be the size of the resulting (downsampled) layer?

$(1 + (J - F)/s) \times (1 + (K - F)/s)$

Question 5 *Submitted Sep 17th 2022 at 2:48:08 pm*

Explain the concept of Data Augmentation, and how it was used in AlexNet.

Data Augmentation is when additional training items are generated from those originally provided, using domain knowledge. In AlexNet, each original image was randomly cropped in different ways to create images of size 224×224. Images can also be reflected left-to-right, and changes can be made to the RGB channels of the images.