Convolutional Neural Networks

MICHAEL WOLLOWSKI

THIS PRESENTATION IS HEAVILY BASED ON THE BLOG ENTRY BY UJJWAL KARN ENTITLED AN INTUITIVE EXPLANATION OF CONVOLUTIONAL NEURAL NETWORKS

Introduction

Convolutional neural networks (CNNs) are more robust pattern matchers than feed-forward networks.

They were developed by Yann LeCun during the late 80s and 90s.

They are successfully used in:
- Image and video recognition
- Recommender systems
- Medical image analysis
- Natural language processing

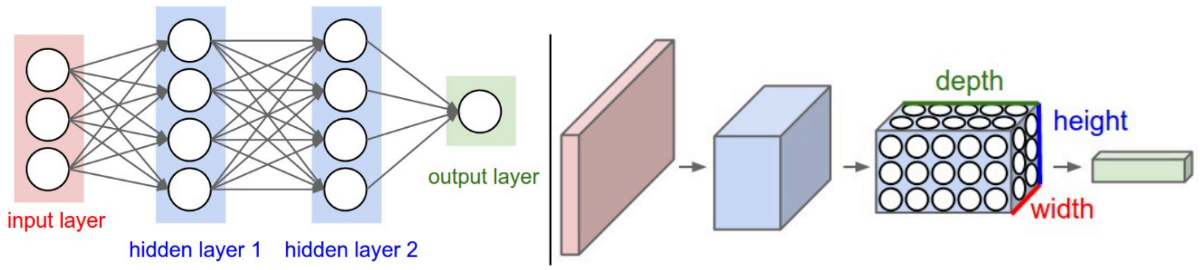
Shift-invariance

Feed-forward networks can be successfully used to classify images.

Furthermore, if we were to shift the images over by several pixels, the network will not be able to reliably classify images any longer.

CNNs are resistant to that effect and are called *shift-invariant.*

Conceptual Differences between FFN and CNN



On left an FFN

On right, CNN which transforms a 3D input into a 3D output.

The depth/height/width block gives a sense of the type of processing

Architecture

Below is an architecture of a CNN.

There are two portions, the ”pattern recognizier” on the left and ”classifier” on he right

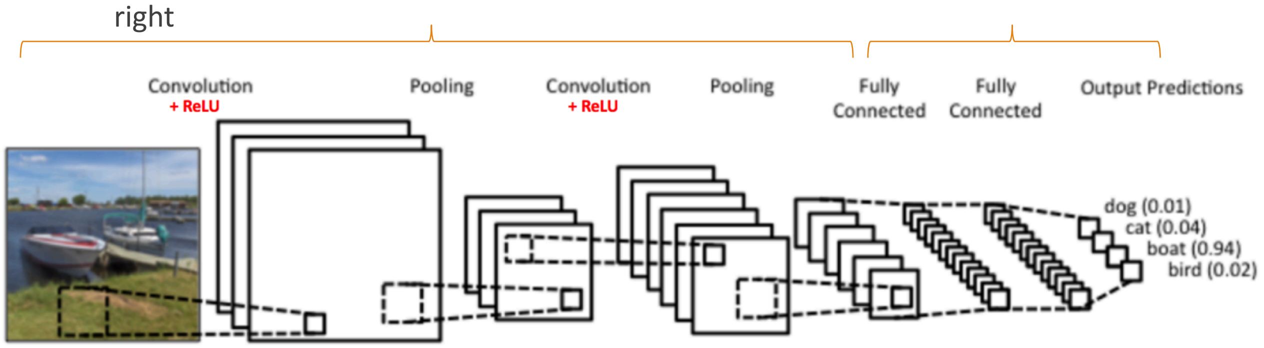


Image source: https://www.clarifai.com/technology

Architecture

The “pattern recognizer” consists of several iterations of a convolutional layer followed by a pooling layer

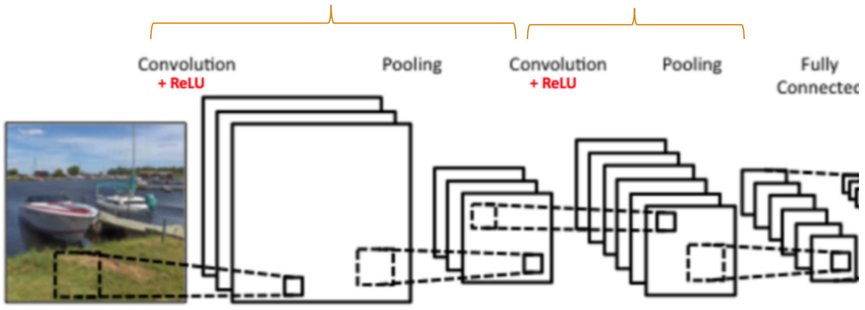


Image source: https://www.clarifai.com/technology

Architecture

The classifier consists of a feed-forward network.

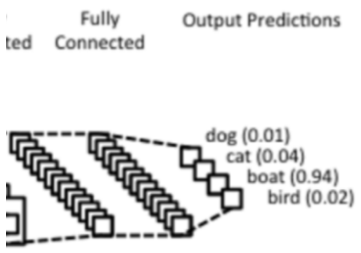
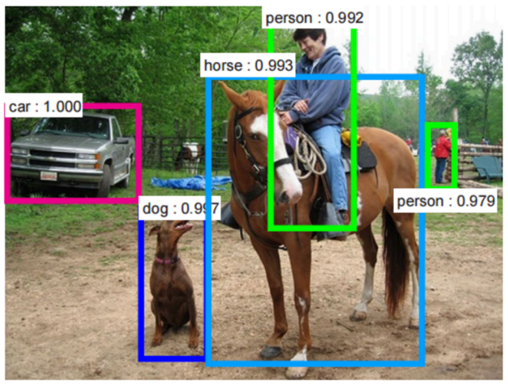


Image source: https://www.clarifai.com/technology

Example Image Recognition Task

Here are two examples of what CNNs can do.



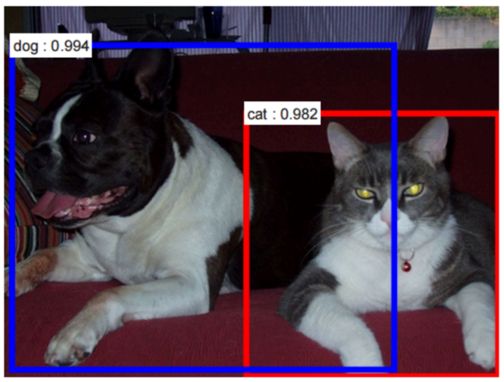


Image source: https://arxiv.org/pdf/1506.01497v3.pdf

Image Processing

We will now take a detailed look at how CNNs process an image.

Let's begin by looking at the data.

A color images comes in three colors and is of a certain dimension.

In this example, there are three colors and the image is 224x224 pixels large

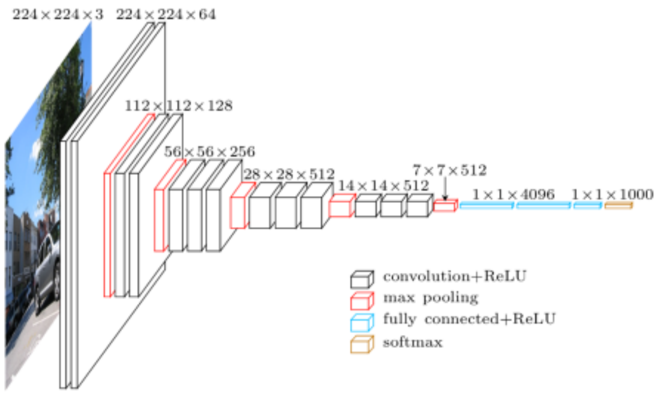


Image source: https://towardsdatascience.com/understand-the-architecture-of-cnn-90a25e244c7

Image Processing

The image is processed so that in this particular example, there are 64 layers of data.

This is called the *convolution* layer.

These layers detect patterns in the data.

In the next layer, the data matrix gets smaller.

In this example the size reduces to a ¼ of its original size.

This is called the *pooling* layer.

We are purposefully zooming-out, to see larger and larger patterns.

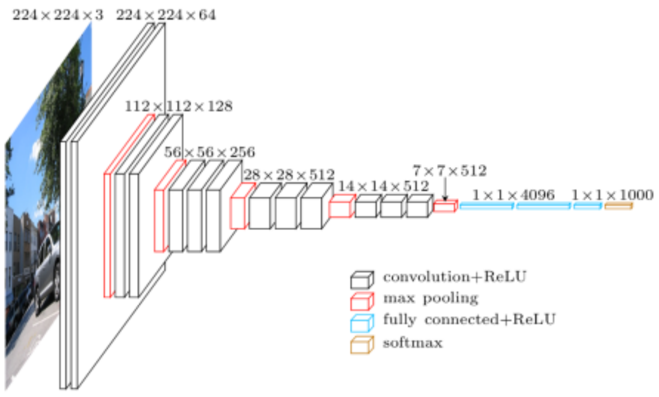


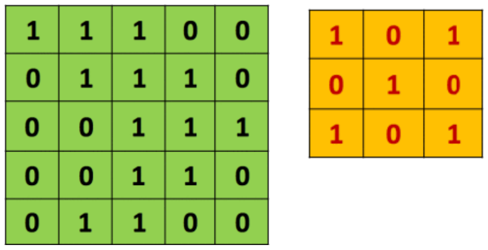
Image source: https://towardsdatascience.com/understand-the-architecture-of-cnn-90a25e244c7

Dissecting a CNN: Convolutions

To simplify, suppose we have the 5x5 image on the left.

In a convolutional layer, we apply another, smaller matrix to the image, called a filter.

A sample filter is shown on the right.



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

1	0	1
0	1	0
1	0	1

Image source: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

Dissecting a CNN: Convolutions

We multiply the filter (matrix) with the image (matrix).

We begin in the upper left corner and work towards the right, pixel by pixel

We continue in this fashion row, by row.

In this example, we move by one pixel.

The resulting matrix is called a *feature map*.

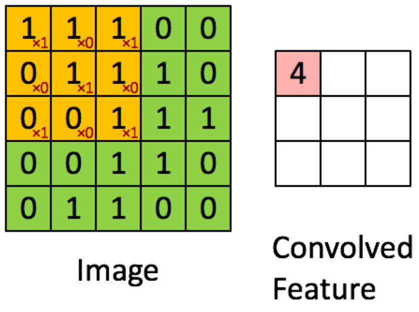


Image source: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

Dissecting a CNN: Convolutions

In our example, the *feature map* is smaller by 1 pixel in each dimension.

If the image is large, it does not matter that we loose a pixel along the border.

Most of the time, important things are not at the borders of images.

If you wish to keep the same size, then you can pad the source image with zeros along the borders.

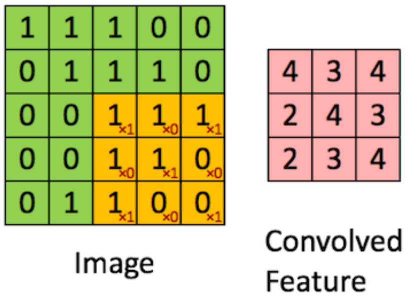


Image source: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

Dissecting a CNN: Convolutions

In our example, we moved one pixel at a time.

The number of pixels by which we over is called *stride.*

If we increase the stride then the feature map will be smaller.

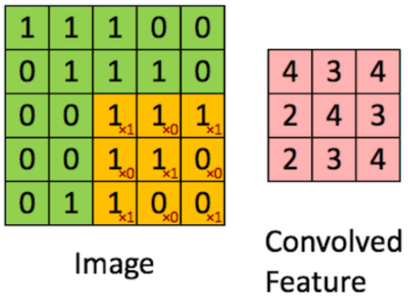


Image source: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

Dissecting a CNN: Filters

In the image on the right, there are 64 feature maps that were produced as part of the first convolutional layer.

Let’s have a look at some filters one may wish to apply.

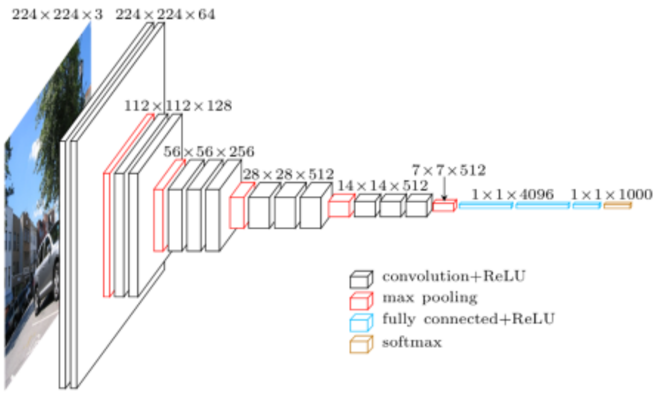


Image source: https://towardsdatascience.com/understand-the-architecture-of-cnn-90a25e244c7

Dissecting a CNN: Filters

Consider the image on the right.

Below is the identity filter and the result of processing the image with it.

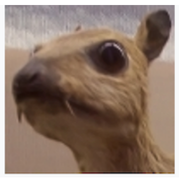
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	

Image source: https://en.wikipedia.org/wiki/Kernel_(image_processing)

Dissecting a CNN: Filters

Here are some filters for edge detection.

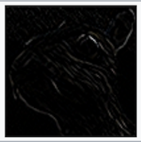
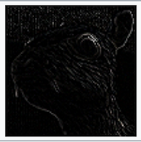
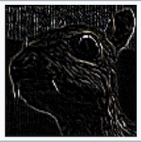
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	

Image source: https://en.wikipedia.org/wiki/Kernel_(image_processing)

Dissecting a CNN: Filters

Here are some filters for sharpening and blurring.

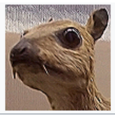
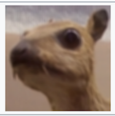
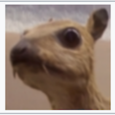
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur 3 × 3 (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	
Gaussian blur 5 × 5 (approximation)	$\frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$	

Image source: https://en.wikipedia.org/wiki/Kernel_(image_processing)

Dissecting a CNN: Pooling

The pooling layer is designed to zoom-out so that the net can focus on a larger pattern.

When reducing an image with pooling, there are several options on how to do that.

In the image on the right, we apply a 2x2 filter, with a stride of 2.

Hence the resulting image is a ¼ of the original images size.

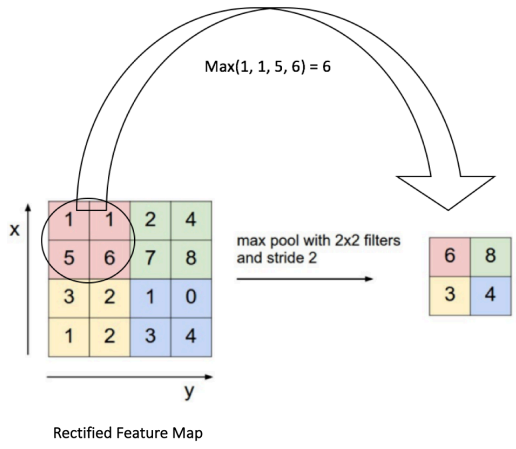


Image source:http://cs231n.github.io/convolutional-networks/

Dissecting a CNN: Pooling

The filter applied is a "max" filter.

It picks the maximum value of the sub-matrix to which it is applied.

There are other filter.

We could have applied a "sum" filter.

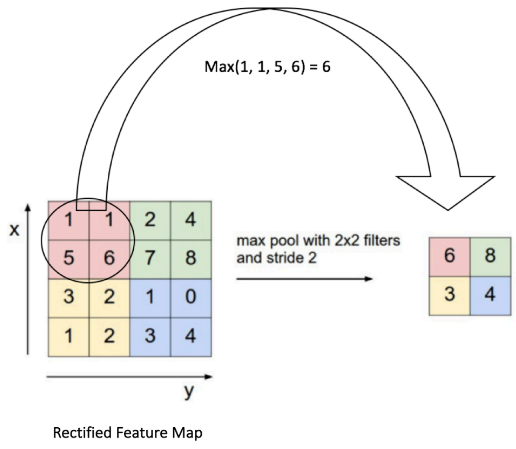


Image source:http://cs231n.github.io/convolutional-networks/

Dissecting a CNN: Pooling

Here is a visualization of applying a max and a sum filter to an image.

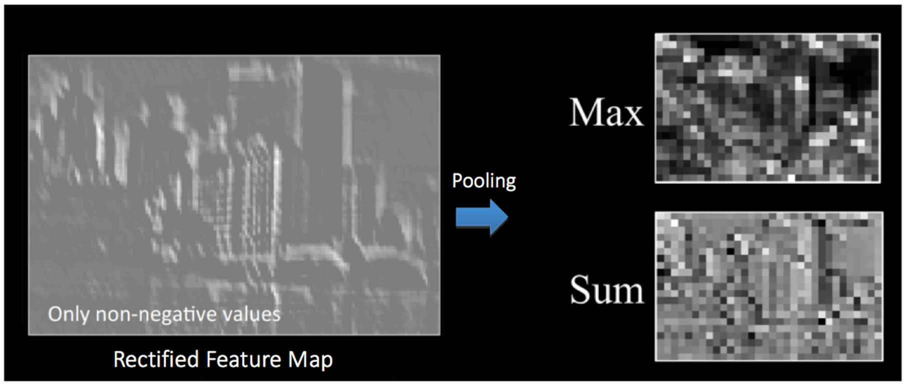


Image source: http://mlss.tuebingen.mpg.de/2015/slides/fergus/Fergus_1.pdf

Architecture

We keep on performing convolutions and pooling

This increases the number of layers or matrices.

See below.

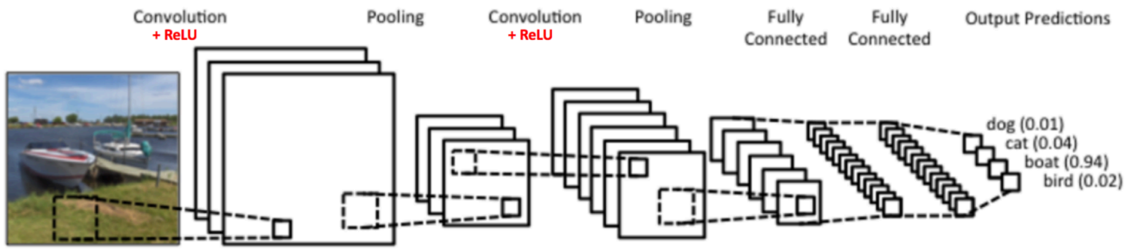


Image source: https://www.clarifai.com/technology

Dissecting a CNN: Classifier

The last stage of a CNN is a feed-forward classifier.

The activation function of the output layer is the *softmax* function.

It takes a vector of arbitrary real-valued scores and squashes it to a vector of values between zero and one that sum to one.

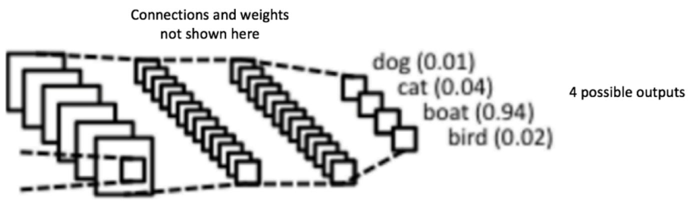
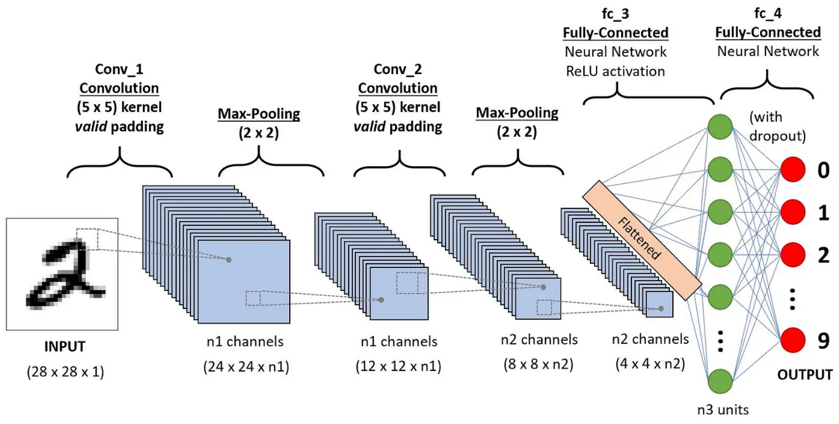


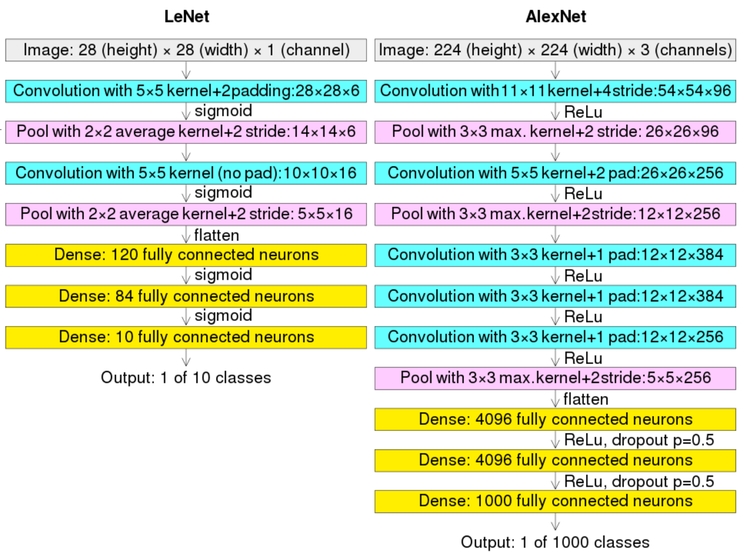
Image source: https://www.clarifai.com/technology

Connecting Image Processor with Classifier



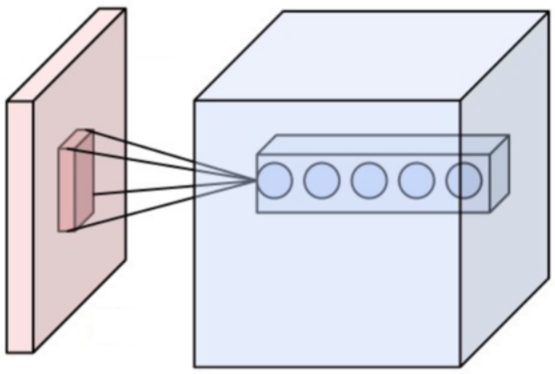
Source: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Sample CNNs for Digit Recognition


LeNet
Image: 28 (height) × 28 (width) × 1 (channel)
Convolution with 5×5 kernel+2padding:28×28×6
sigmoid
Pool with 2×2 average kernel+2 stride:14×14×6
Convolution with 5×5 kernel (no pad):10×10×16
sigmoid
Pool with 2×2 average kernel+2 stride: 5×5×16
flatten
Dense: 120 fully connected neurons
sigmoid
Dense: 84 fully connected neurons
sigmoid
Dense: 10 fully connected neurons
Output: 1 of 10 classes
AlexNet
Image: 224 (height) × 224 (width) × 3 (channels)
Convolution with 11×11 kernel+4 stride:54×54×96
ReLu
Pool with 3×3 max. kernel+2 stride: 26×26×96
Convolution with 5×5 kernel+2 pad:26×26×256
ReLu
Pool with 3×3 max.kernel+2stride:12×12×256
Convolution with 3×3 kernel+1 pad:12×12×384
ReLu
Convolution with 3×3 kernel+1 pad:12×12×384
ReLu
Convolution with 3×3 kernel+1 pad:12×12×256
ReLu
Pool with 3×3 max.kernel+2stride:5×5×256
flatten
Dense: 4096 fully connected neurons
ReLu, dropout p=0.5
Dense: 4096 fully connected neurons
ReLu, dropout p=0.5
Dense: 1000 fully connected neurons
Output: 1 of 1000 classes

Source: https://en.wikipedia.org/wiki/Convolutional_neural_network

Convolutional Layer and Receptive Field



Source: https://en.wikipedia.org/wiki/Convolutional_neural_network