

# ~~Sudoku~~ and Optical Character Recognition in Python

IDA Østjylland Seminar, 23/10/2023

Luminita C. Totu  
Control Engineer  
luminita.totu@gmail.com



5	3		7			
6		1	9	5		
	9	8			6	
8			6			3
4		8	3			1
7		2			6	
	6			2	8	
		4	1	9		5
			8		7	9

# OCR in Python: Pipeline for Today

- ***Image Processing***
- Text Detection (bounding boxing)
- Text Recognition

5	3		7			
6		1	9	5		
	9	8			6	
8			6			3
4		8	3			1
7			2			6
	6			2	8	
		4	1	9		5
			8		7	9

# Image Processing

Opens source tools for image processing:

- Pillow (basic image processing tasks)
- Scikit-image (python based, diverse set of algorithms, good for research, not fast / real-time)
- OpenCV (comprehensive library aimed at real-time computer vision, multi-language support)

Basic Image Processing functionalities:

- Reading through the file encoding format to extract the color matrix or the images “bands of data” ( and conversions between color systems)
- (0,0) in the upper left corner; coordinates refer to the implied pixel corners
- Geometric transformations: rotation, **resizing**, cropping, flip, mirror
- Enhancements: color, contrast, sharpness, brightness adjustments functions, **grayscale**
- Draw, write on existing images or as images
- **Filtering & Convolution**
- **Morphology Operations**
- **Thresholds**

5	3		7			
6			1	9	5	
	9	8				6
8			8	6		3
4				3		1
7			2			6
	6				2	8
			4	1	9	5
			8		7	9

# Image Processing

- Resizing
- Grayscaleing
- Filtering & Convolution
- Morphology Operations
- Thresholding

5	3		7			
6			1	9	5	
	9	8				6
8			8	6		3
4				3		1
7			2			6
	6				2	8
			4	1	9	5
			8		7	9

# Image Processing - Resizing

- Refers to reducing its size or resolution (the width and height of the matrix)
- Multiple input pixels will be mapped to a single output pixel
- This operation can lead to a loss of information, which is why the choice of **resampling technique** is crucial to maintain as much of the original content as possible.

Different options:

- Nearest-neighbor Interpolation
- Bilinear Interpolation
- Bicubic Interpolation
- Area-based (or Pixel-averaging)
- Hamming
- Lanczos Resampling

5	3		7			
6			1	9	5	
	9	8				6
8			6			3
4		8		3		1
7			2			6
	6				2	8
		4	1	9		5
			8		7	9

# Image Processing - Grayscale

Converting an RGB image to grayscale, some techniques:

- **Averaging Method:** takes the average of the red, green, and blue color channels.  

$$\text{Gray} = (R+G+B) / 3$$
- **Luminosity (or Weighted) Method:** a widely used technique as it takes into account the perceived intensity of colors to the human eye. The human eye is most sensitive to green  

$$\text{Gray} = \text{Gray} = 0.299 \times R + 0.587 \times G + 0.114 \times B$$
- **Desaturation Method:** It involves taking the average of the maximum and minimum of the three color channels.

$$\text{Gray} = 0.5 * ( \max(R,G,B) + \min(R,G,B) )$$

5	3		7			
6		1	9	5		
	9	8			6	
8			6			3
4		8	3			1
7		2			6	
	6			2	8	
		4	1	9		5
			8		7	9

# Image Processing - Thresholding/Binarization

The idea is to convert a grayscale image into a binary image where the pixels are either 0 (black) or 1 (white) based on a threshold value:

- **Simple, global technique:** a single threshold value is user-chosen, and all pixels with intensities above (or below) this value are set to one level (either black or white), and all other pixels to the opposite level
- **Otsu's Thresholding:** Assumes that the image contains two classes of pixels and calculates the optimum threshold that minimizes the intraclass variance. Particularly effective for bimodal histograms.
- **Triangle Thresholding:** Draws a line from the peak of the histogram (highest frequency) to the farthest end of the histogram and then finds the point on the histogram that is farthest from this line. This point's intensity is taken as the threshold
- **Adaptive (or Local) Thresholding:** Instead of a single threshold value, different values are used for different regions of an image. Useful for images with varying light conditions. Common methods include "Mean Adaptive Thresholding" where the threshold is the mean of the neighboring area and "Gaussian Adaptive Thresholding" where the threshold is a weighted sum of the neighboring values, with weights given by a Gaussian window.

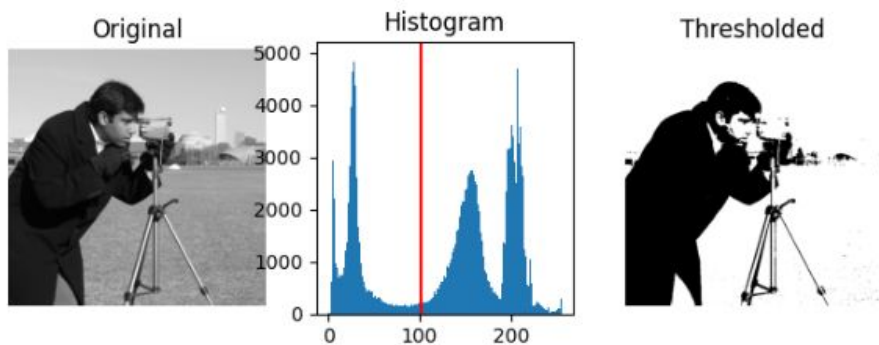


5	3		7			
6		1	9	5		
	9	8			6	
8			6			3
4		8	3			11
7		2				6
	6			2	8	
		4	1	9		5
			8		7	9

# Image Processing - Thresholding

- for pictures with a bimodal histogram, more specific algorithms can be used. For instance, the minimum algorithm takes a histogram of the image and smooths it repeatedly until there are only two peaks in the histogram.

- Otsu-Thresholding:



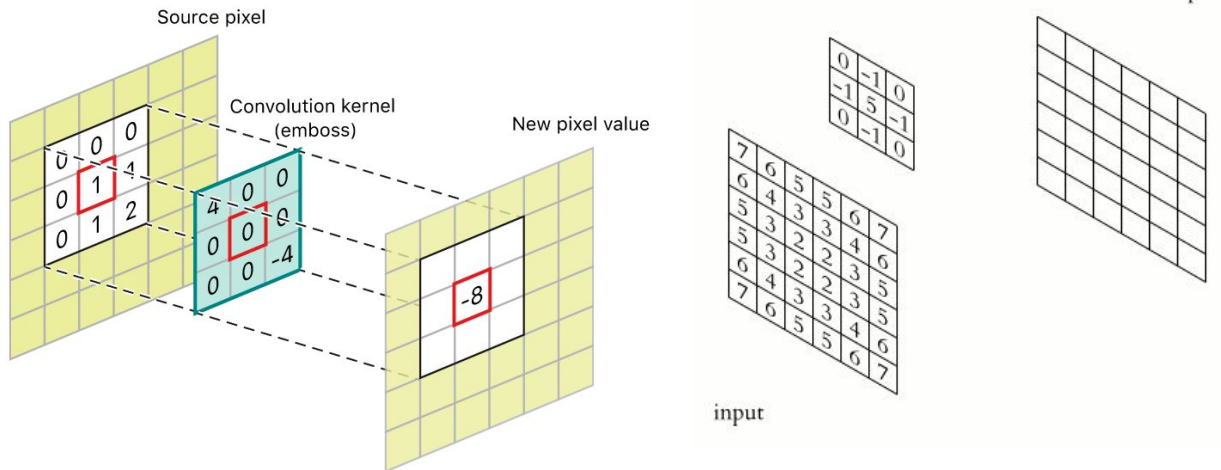
*scikit-images*

- Otsu's threshold method can be applied locally. For each pixel, an "optimal" threshold is determined by maximizing the variance between two classes of pixels of the local neighborhood (defined by `block_size` or a by a structuring element)

5	3		7			
6			1	9	5	
	9	8				6
8			6			3
4		8	3			1
7			2			6
	6				2	8
		4	1	9		5
			8		7	9

# Image Processing - Convolution Kernels

**Spatial Filtering with a Convolution Kernel:** a small matrix used for blurring, sharpening, embossing, edge detection, and more. This is accomplished by doing a “convolution” between the kernel and an image. Each pixel in the output image is a *function* of the nearby pixels (including itself) in the input image. The kernel is that function.



Kernel		Image		Output
0 -1 0		2 2 2		
-1 5 -1	×	2 3 2	=	7
0 -1 0		2 2 2		
Kernel		Image		Output
0 -1 0		2 2 2		
-1 5 -1	×	2 1 2	=	-3
0 -1 0		2 2 2		

[https://developer.apple.com/documentation/accelerate/blurring\\_an\\_image](https://developer.apple.com/documentation/accelerate/blurring_an_image)

Prakhar Ganesh, <https://towardsdatascience.com/types-of-convolution-kernels-simplified-f040cb307c37>

5	3		7			
6			1	9	5	
	9	8				6
8			6			3
4		8	3			1
7		2			6	
	6			2	8	
		4	1	9		5
			8		7	9

# Image Processing - Convolution Kernels

- To be useful, however, a kernel should generally be small enough that the majority of the pixel accesses fall within the bounds of the target image. Performance is also a factor with large kernels.

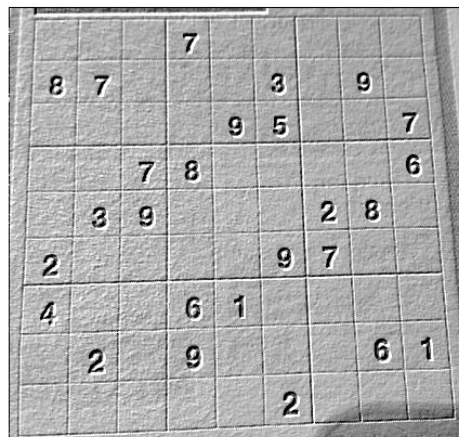
*Activity !* <https://setosa.io/ev/image-kernels/>

Choose File S1.jpg

Live video

-2	-1	0
-1	1	1
0	1	2

emboss ▼



5	3		7			
6			1	9	5	
	9	8				6
8			6			3
4		8	3			11
7			2			6
	6			2	8	
		4	1	9		5
			8		7	9

# Image Processing - Convolution Kernels

- **Box Blurring Filter** is shader friendly

$$\begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix}$$

// A shader is a piece of code that is executed on the Graphics Processing Unit (GPU), usually found on a graphics card, to manipulate an image before it is drawn to the screen

*Example of a 3x3 Kernel*

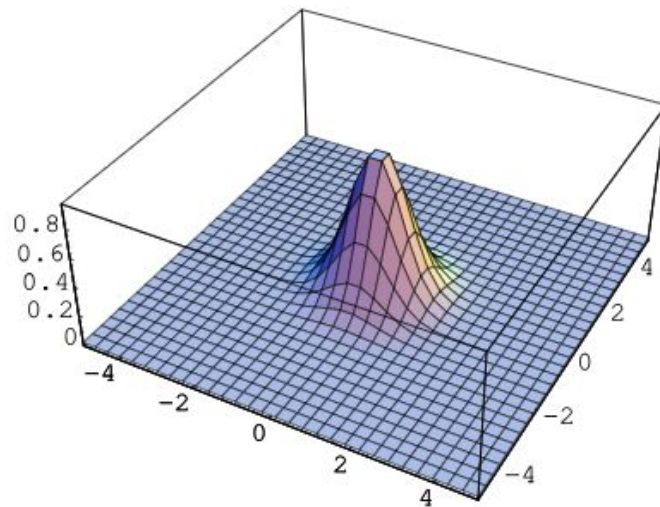
5	3		7			
6			1	9	5	
	9	8				6
8			6			3
4		8		3		11
7			2			6
	6				2	8
		4	1	9		5
			8		7	9

# Image Processing - Convolution Kernels

## - Gaussian blur

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

$\frac{1}{273}$



*Example of 2D Gaussian function with sigma = 1  
Normalized; Kernel Size 5x5*

5	3		7			
6		1	9	5		
	9	8			6	
8			6			3
4		8	3			11
7		2				6
	6			2	8	
		4	1	9		5
			8		7	9

# Image Processing - Convolution Kernels

**Edge Detection:** Basic, Sobel, Prewitt and Kirsch operators.

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

*A basic 3x3 edge detector kernel*

$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} \quad \mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

*The Sobel detector kernels*

- Sobel, Prewitt and Kirsch operators are composed of multiple convolutions
- Sobel and Prewitt are similar: They perform a pair of horizontal and vertical convolutions, which are then used to produce a final edge value at the target pixel.
- The Kirsch edge detector performs a convolution for each of the 8 compass directions at the target pixel

$$\mathbf{g}^{(1)} = \begin{bmatrix} +5 & +5 & +5 \\ -3 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix}, \quad \mathbf{g}^{(2)} = \begin{bmatrix} +5 & +5 & -3 \\ +5 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix}$$

$$\mathbf{g}^{(3)} = \begin{bmatrix} +5 & -3 & -3 \\ +5 & 0 & -3 \\ +5 & -3 & -3 \end{bmatrix}, \quad \dots$$

*The Kirsch detector kernels*

5	3		7		
6		1	9	5	
	9	8			6
8			6		3
4		8		3	1
7		2			6
	6			2	8
		4	1	9	5
			8		7
					9

# Image Processing - Convolution Kernels

## Edge Detection: Explanation of Basic Kernel



### 1D Black/White case:

- We need to find a way to “detect” the “01” and “10” adjacent pixels (0 black, 1 white). The edge pixel is the white one
- We consider a 3x1 kernel
- Total number of possibilities in the image are: 000, 001, 010, 011, 100, 101, 110 and 111
- The current pixel (middle one in the kernel) is an edge if the pattern is one of 010, 011, 110

$$edge(x) = -pixels[x - 1] + 2 * pixels[x] - pixels[x + 1] \quad + \quad \text{Clamping}$$

<i>pattern</i>	<i>result</i>	<i>clamped</i>
000	0	0
001	-1	0
010	2	1
011	1	1

100	-1	0
101	-2	0
110	1	1
111	0	0

Kernel is:  $[-1 \ 2 \ -1]$  + clamping

- Different options can to be considered for the boundary pixels

5	3		7		
6		1	9	5	
	9	8			6
8			6		3
4			8	3	
7		2			6
	6			2	8
		4	1	9	
			8		7
				5	9

# Image Processing - Convolution Kernels

## Sharpening:

- A simple technique is to add the edge detector output to the initial image. The edges be more apparent. The sharpening effect can be controlled by introducing an amount parameter that scales the edge detector contribution, see the 3x3 example

$$K_{sharp} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} * amount$$

$$K_{sharp} = \begin{bmatrix} -0.00391 & -0.01563 & -0.02344 & -0.01563 & -0.00391 \\ -0.01563 & -0.06250 & -0.09375 & -0.06250 & -0.01563 \\ -0.02344 & -0.09375 & 1.85980 & -0.09375 & -0.02344 \\ -0.01563 & -0.06250 & -0.09375 & -0.06250 & -0.01563 \\ -0.00391 & -0.01563 & -0.02344 & -0.01563 & -0.00391 \end{bmatrix}$$

- “Unsharp” technique - combining an edge detector and blur filter, results in a more refined sharpening effect, see the 5x5 example

<https://www.taylorpetrick.com/blog/post/convolution-part3>

<https://www.imagemagick.org/Usage/convolve/>



5	3		7			
6		1	9	5		
	9	8			6	
8			6			3
4		8	3			11
7		2			6	
	6			2	8	
		4	1	9		5
			8		7	9

# Image Processing - Morphological Operations

- Similar with the spatial filtering with a convolution kernel: it has a structuring element and a predefined calculation rule
- different from the process of convolution, the morphological operation is logical (e.g. AND, OR, NOT) rather than arithmetic
- It is applied on binary images (black & white images). Some of the techniques can be extended to grayscale images as well
- Affects the “shape” of the binary image
- See Mathematical Morphology

**Example of Operations:** Dilation (D), Erosion (E), Closing (D+E), Opening (E+D), Hit-or-Miss

**Other examples:** thinning, pruning, top hat

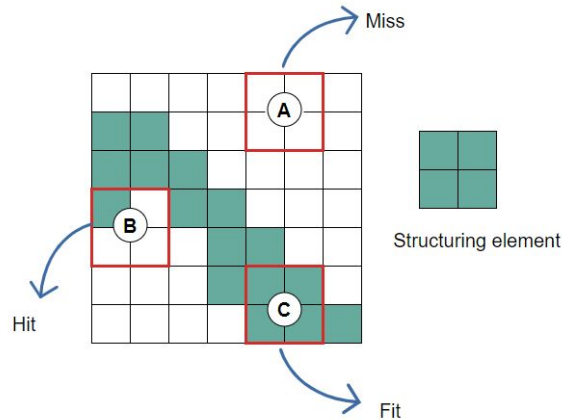
First, a **structuring element** (which is similar to the kernel in convolution, moves point-to-point across each pixel of the input image) it is a matrix or a small-sized template that is used to traverse an image. **Its pixels have values zero or one.** It also has an origin, the middle of the matrix (therefore odd size)

5	3		7			
6		1	9	5		
	9	8			6	
8			8	6		3
4				3		1
7		2				6
	6				2	8
		4	1	9		5
			8		7	9

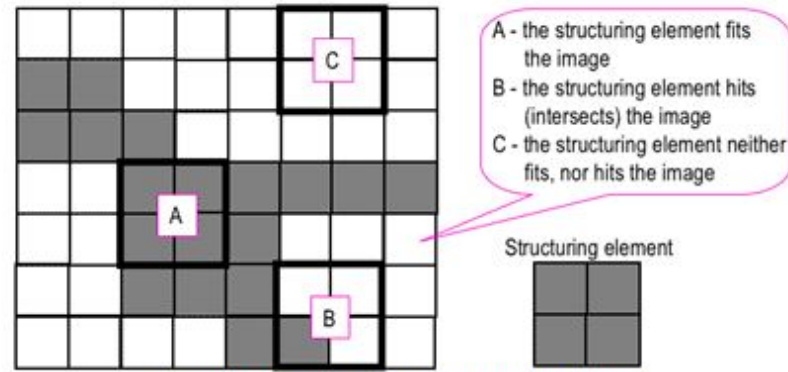
# Image Processing - Morphological Operations

The structuring element is positioned at all possible locations in the image, and it is **compared with the overlaid pixels**.

- **Fit**: When all the pixels in the *structuring element* cover the pixels of the object,
- **Hit**: When at least one of the pixels in the *structuring element* cover the pixels of the object
- **Miss**: When no pixel in the structuring element cover the pixels of the object



Probing an image with a structuring element



Probing of an image with a structuring element

(white and grey pixels have zero and non-zero values, respectively).

5	3		7			
6		1	9	5		
	9	8			6	
8			8	6		3
4				3		1
7			2			6
	6				2	8
		4	1	9		5
			8		7	9

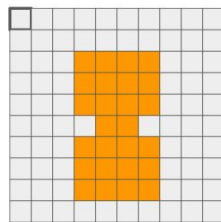
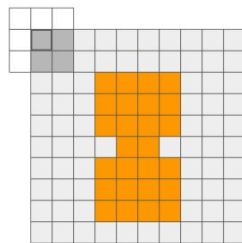
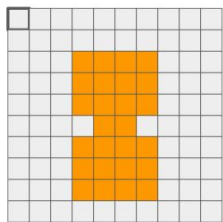
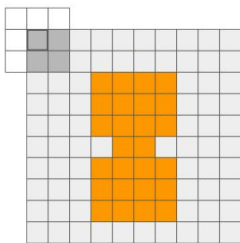
# Image Processing - Morphological Operations

**Morphological Dilation:** expands the image pixels, or it adds pixels on object boundaries. Can repair breaks or intrusions. Computational rule:

- **Pixel (output) = 1 {if HIT or FIT}**
- **Pixel (output) = 0 {otherwise}**

**Morphological Erosion:** expands the image pixels, or it adds pixels on object boundaries. Computational rule:

- **Pixel (output) = 1 {if FIT}**
- **Pixel (output) = 0 {otherwise}**



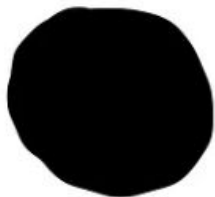
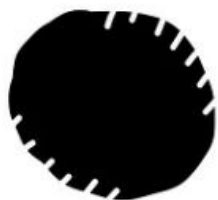
5	3		7			
6		1	9	5		
	9	8			6	
8			6			3
4		8	3			11
7		2				6
	6			2	8	
		4	1	9		5
			8		7	9

# Image Processing - Morphological Operations

Examples of “purposeful” Dilation:



can repair breaks



can repair intrusions



1



2



3

5	3		7			
6		1	9	5		
	9	8			6	
8			6			3
4		8	3			11
7		2				6
	6			2	8	
		4	1	9		5
			8		7	9

# Image Processing - Morphological Operations

Examples of “purposeful” Erosion:



can split apart  
joint objects



can strip away  
extrusions



1



2



3



```
kernel = np.ones((10, 1),  
np.uint8)
```



5	3		7			
6		1	9	5		
	9	8			6	
8			6			3
4		8	3			1
7			2			6
	6				2	8
		4	1	9		5
			8		7	9

# Image Processing - Morphological Operation

- **Closing** (by first performing dilation and then erosion)
- **Opening** (by first performing erosion and then dilation)



image

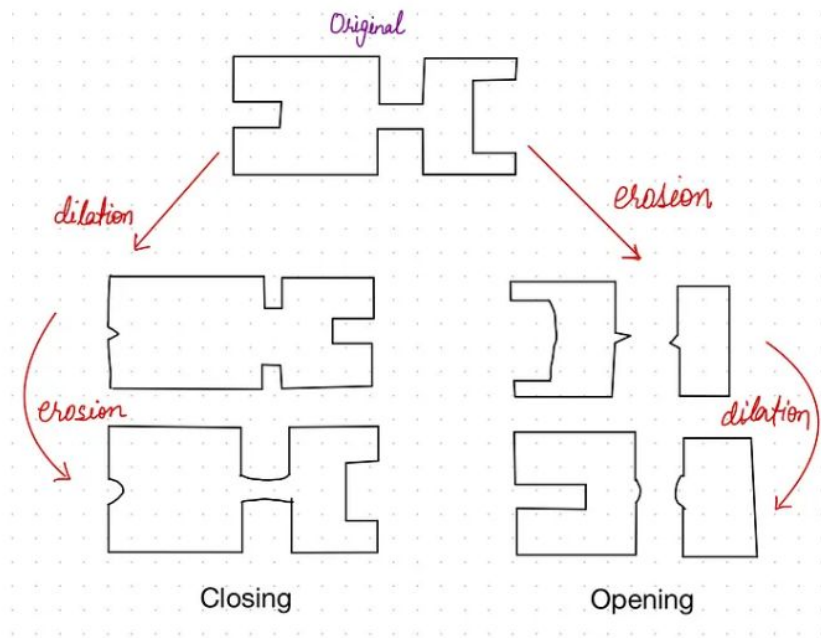


image



QR code after Closing

<https://www.dynamsoft.com/blog/image-processing/morphological-operations/>



5	3		7			
6		1	9	5		
	9	8			6	
8			6			3
4		8	3			11
7		2				6
	6			2	8	
		4	1	9		5
			8		7	9

# Image Processing - Morphological Operation

<https://planetcalc.com/9325/>



Activity!

5	3		7			
6			1	9	5	
	9	8				6
8			6			3
4		8		3		11
7			2			6
	6				2	8
		4	1	9		5
			8		7	9

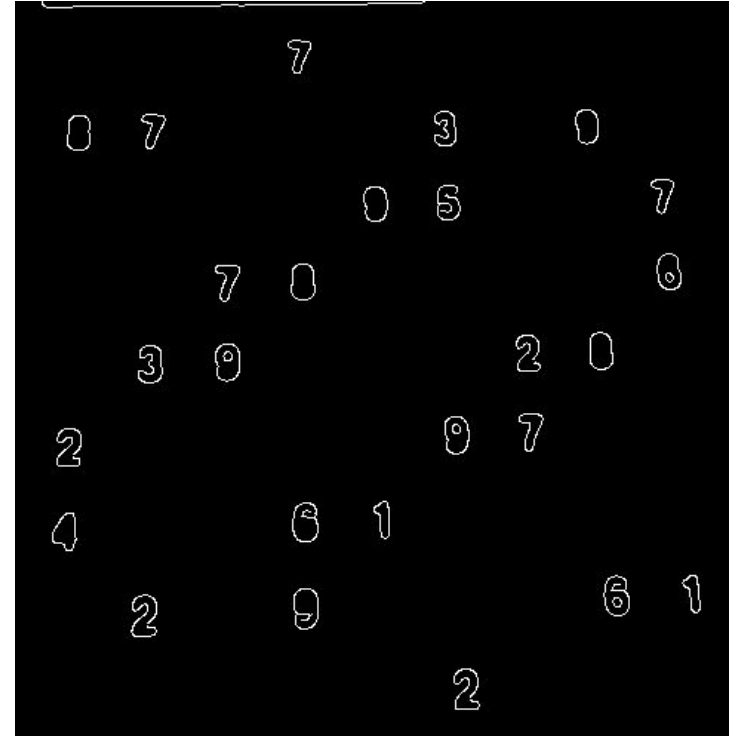
# Image Processing: Other Alg

- Canny Edge Detection (John F. Canny 1986), multi-stage filter:

Noise reduction ( gaussian filter 5x5 ), a Sobel kernel for gradients, removing everything that is not an edge based on gradient ( non-maximum suppression), and an extra step that requires two thresholds.

- Contours extraction (1985, Satoshi Suzuki and others)

(opencv) Finding white object from black background. A curve joining all the continuous points (along the boundary), having same color or intensity.

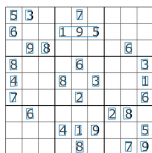




5	3		7			
6			1	9	5	
	9	8				6
8			8	6		3
4				3		1
7			2			6
	6				2	8
			4	1	9	5
			8		7	9

# OCR in Python: Pipeline for Today

- Image Processing
- ***Text Detection (bounding boxing)***
- Text Recognition



# Bounding Box Detection for Text

The objective is to identify and outline regions in an image that contain text. An incomplete overview of methods:

Classical Methods	NN Methods/Deep Learning
<b>Morphological Operations</b> are most effective for images with clear contrast between text and background (thresholding + dilation e.g)	<b>EAST</b> (Efficient and Accurate Scene Text Detector)
<b>Connected Component Analysis</b> (thresholding first). Suitable for simple scenarios where text regions are distinctly separate	<b>CRAFT</b> (Character-Region Awareness For Text detection)
<b>Maximally Stable Extremal Regions:</b> detects areas in images where pixel intensities are consistent. Suitable for scene text detection.	<b>SSD</b> (Single Shot MultiBox Detector)
	<b>Faster R-CNN</b> (Region-CNN, Fast R-CNN, Faster R-CNN)

5	3		7			
6		1	9	5		
	9	8			6	
8			6			3
4		8	3			1
7		2			6	
	6			2	8	
		4	1	9		5
			8		7	9

# Bounding Box Detection for Text

## CRAFT- pytorch:

clovaai / CRAFT-pytorch

Issues 86 Pull requests 12 Actions Projects Security Insights

CRAFT-pytorch Public

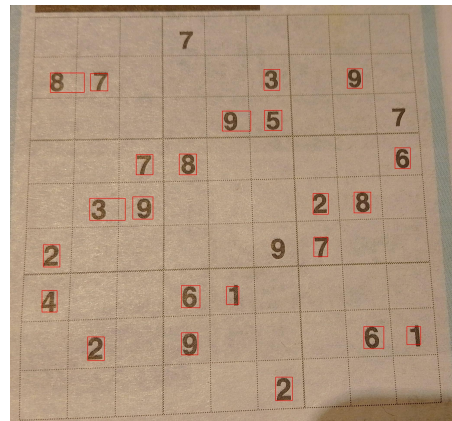
master 2 branches 0 tags

Youngmin Baek add torch.no\_grad()

basenet initial commit

figures add license


- Manually download the trained NN file
- Can run on a CPU
- **python test.py --trained\_model=dwnlds\craft\_mlt\_25k.pth --test\_folder=...newspaper\_lokalavisenfavrskov --cuda=False**



5	3		7			
6		1	9	5		
	9	8			6	
8			6			3
4	8		3			1
7		2				6
	6			2	8	
		4	1	9		5
			8		7	9

# Bounding Box Detection for Text

**EAST** (Efficient accurate scene text detector) is supported by OpenCV, but does not work well on my Sudoku images


Open Source Computer Vision

[Main Page](#)
[Related Pages](#)
[Modules](#)
[Namespaces ▾](#)
[Classes ▾](#)
[Files ▾](#)
[Examples](#)
[Java documentation](#)

[OpenCV Tutorials](#)
[Deep Neural Networks \(dnn module\)](#)

## High Level API: TextDetectionModel and TextRecognitionModel

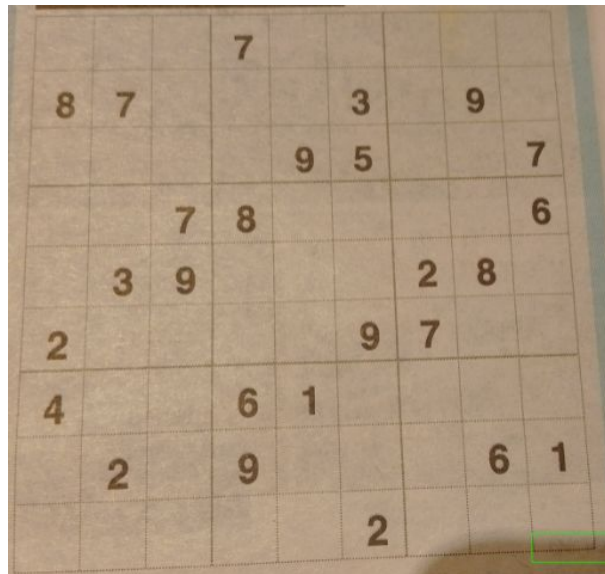
Prev Tutorial: [How to run custom OCR model](#)

Next Tutorial: [DNN-based Face Detection And Recognition](#)

Original author	Wenqing Zhang
Compatibility	OpenCV >= 4.5

### Introduction

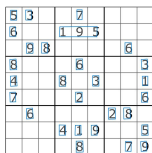
In this tutorial, we will introduce the APIs for TextRecognitionModel and TextDetectionModel in detail.



5	3		7			
6			1	9	5	
	9	8				6
8			8	6		3
4				3		1
7			2			6
	6				2	8
			4	1	9	5
			8		7	9

# OCR in Python: Pipeline for Today

- Image Processing
- Text Detection (bounding boxing)
- ***Text Recognition***



# Text Recognition

Incomplete overview of methods for text recognition, with focus on single digit recognition:

Rule based matching	Deep Learning
<b>Template Matching</b> - uses comparison of the image with a predefined template of each character. Works well when font and size are known and consistent, doesn't not handle well variability in fonts, sizes, and distortions.	<b>CNN</b> - LeNet-5, proposed by Yann LeCun in the late 1990s, is one of the early CNNs used for digit recognition. Modern architectures like AlexNet, VGG, and ResNet have also been adapted for OCR tasks.
<b>Feature Extraction</b> of like Zoning, Histograms, Profiles, and Geometrical moments, from characters and uses these features for classification. Can handle some variability in fonts and sizes, but it sensitive to distortions, noise, and requires good segmentation.	<b>Tesseract OCR</b> : An open-source OCR engine originally developed by HP and now supported by Google. Over time, its architecture has evolved, and the latest versions use LSTM-based methods. It is still continuously improved. Although powerful, it might not be the best for all specific use cases, and fine-tuning might be required
<b>Boundary Analysis</b> - analysis of the boundary or contour of the digit and extraction of features such as curvature, inflection points, and corners can provide clues about the digit's identity.	<b>.OpenCV Text Recognition</b> supports loading of neural networks and has a couple of options available documented <a href="https://docs.opencv.org/4.x/d4/d43/tutorial_dnn_text_spotting.html">https://docs.opencv.org/4.x/d4/d43/tutorial_dnn_text_spotting.html</a> <a href="https://docs.opencv.org/4.x/d9/d1e/tutorial_dnn_OCR.html">https://docs.opencv.org/4.x/d9/d1e/tutorial_dnn_OCR.html</a>

5	3		7			
6		1	9	5		
	9	8			6	
8			6			3
4		8	3			1
7		2				6
	6			2	8	
		4	1	9		5
			8		7	9

# Neural Networks

```
+-----+
| Training Dataset |
+-----+
```

|  
v

```
+-----+
| Neural Network |
| (Architecture) |
+-----+
```

=>

```
+-----+
| Training      |
| Procedure/    |
| Algorithm     |
| (e.g., SGD)   |
+-----+
```

=>

```
+-----+
| Input (specific |
| format e.g., image) |
+-----+
```

|  
v

```
+-----+
| Trained Model   |
| (Stored in various |
| formats: pb,    |
| onnx, h5, etc.) |
+-----+
```

|  
v

```
+-----+
| Output (Prediction/ |
| Classification, etc.) |
+-----+
```

When using a Trained Model it is first important for the final performance to know:

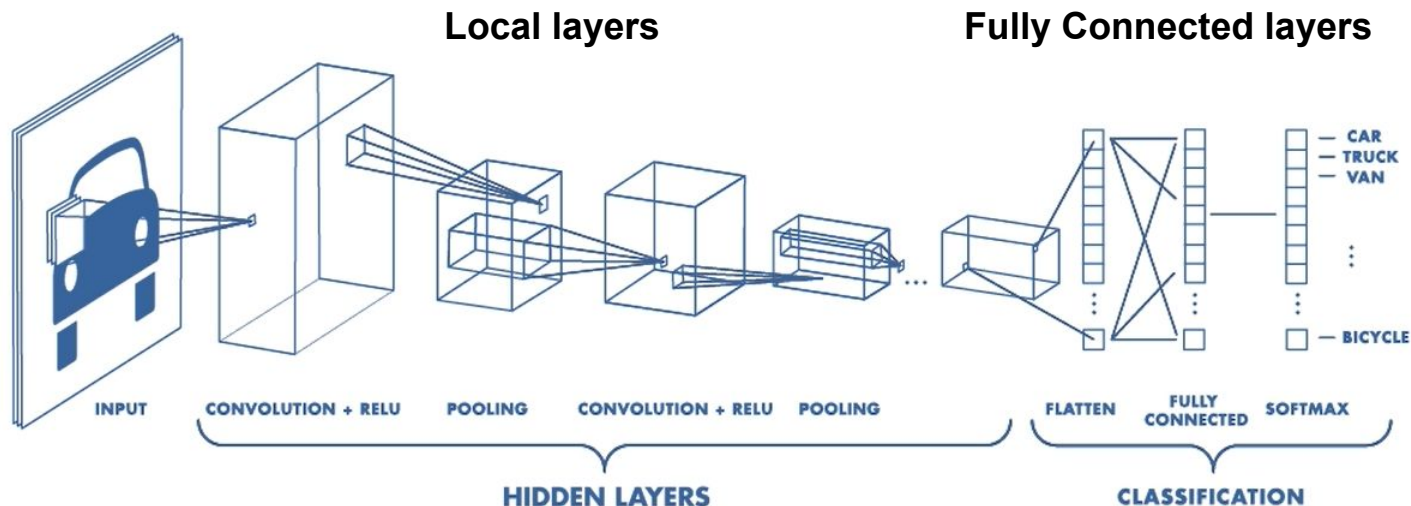
- input format and recommended requirements
- full output format (includes scores and details about the result)
- its own format and available run-time options (e.g. CPU/CUDA)
- know as much as possible about its pedigree (training dataset, architecture, etc )

5	3		7			
6			1	9	5	
	9	8				6
8				6		3
4			8	3		1
7			2			6
	6				2	8
		4	1	9		5
			8		7	9

# Convolutional Neural Networks

CNN are the most popular type for image processing tasks. CNNs utilize layers of convolutional filters that can learn spatial hierarchies of features. CNNs have been used extensively for tasks such as image classification, object detection, and segmentation. Examples of CNN architectures: LeNet, VGG-16, ResNet, Inception, and MobileNet.

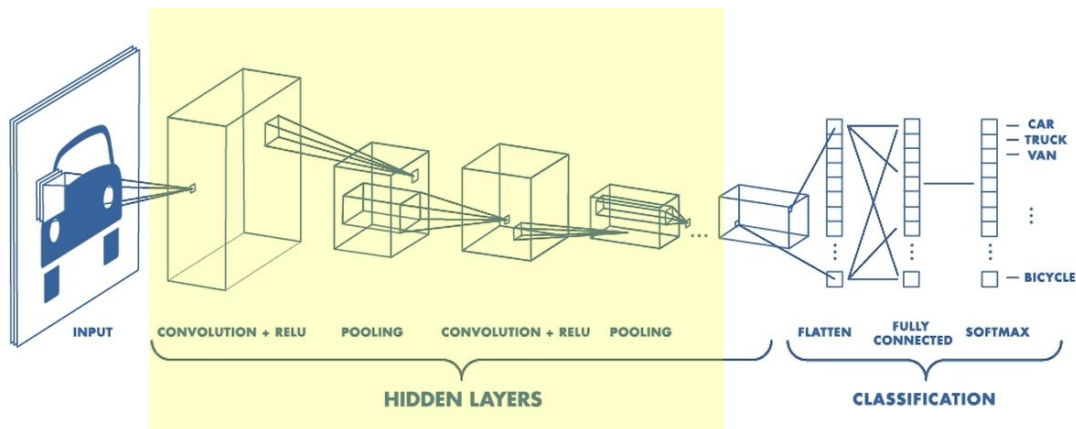
<https://se.mathworks.com/videos/introduction-to-deep-learning-what-are-convolutional-neural-networks--1489512765771.html>





5	3		7			
6			1	9	5	
	9	8				6
8			6			3
4		8	3			11
7		2				6
	6			2	8	
		4	1	9		5
			8		7	9

# Convolutional Neural Networks



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

Image

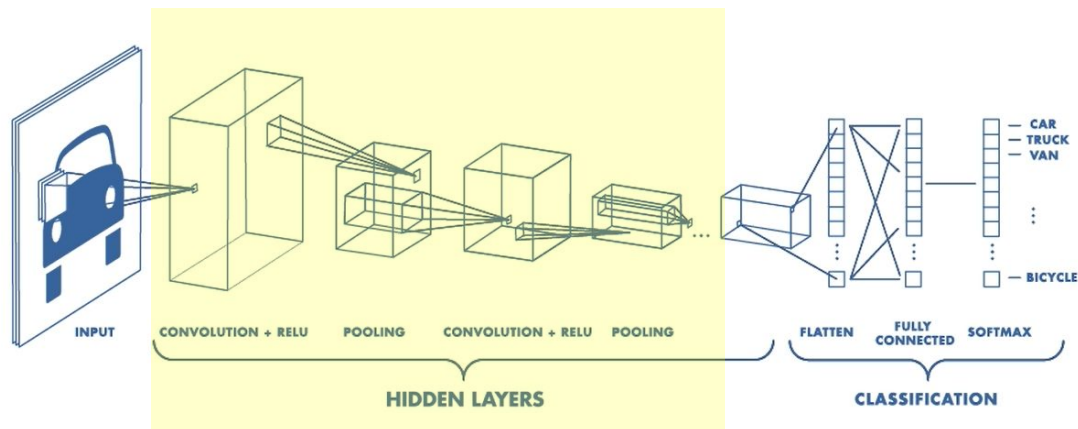
4		

Convolved  
Feature

- Learnable filter/kernels are applied to each data band, just like a typical convolution operation, resulting in a stack of feature /convolution maps.
- Filters/Kernels are again typically small in width and height, and typically square
- The training will optimize the filter/kernel values
- The image is multiplied by the number of kernels/filters, the filter sliding step can reduce the size of the image slice, padding parameters, band of data depth multiplies again the number of slices

5	3		7			
6			1	9	5	
	9	8				6
8			6			3
4		8	3			11
7			2			6
	6			2	8	
			4	1	9	5
			8		7	9

# Convolutional Neural Networks



**Input**

**ReLU**

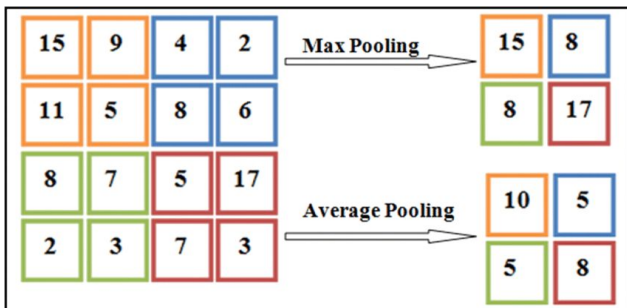
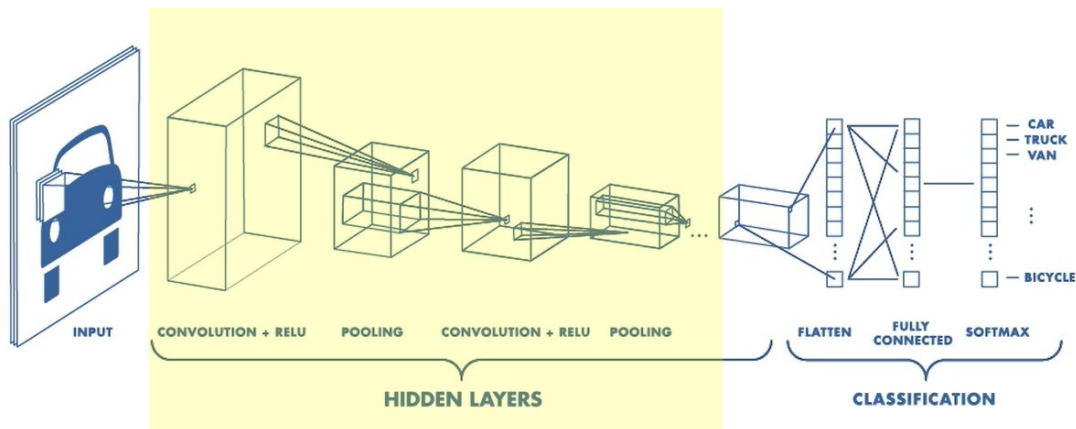
-249	-91	-37
250	-134	101
27	61	-153

0	0	0
250	0	101
27	61	0

- The filters/kernels weights can be both positive and negative. Depending on the region of the image and the values of the filter, the convolution can produce negative. This is perfectly normal and expected.
- By applying the ReLU activation, we effectively discard these negative activations, setting them to zero, which can simplify the learning process and lead to more robust features.
- Output is also called Activation Maps

5	3		7			
6			1	9	5	
	9	8				6
8			6			3
4		8	3			1
7			2			6
	6			2	8	
		4	1	9		5
			8		7	9

# Convolutional Neural Networks

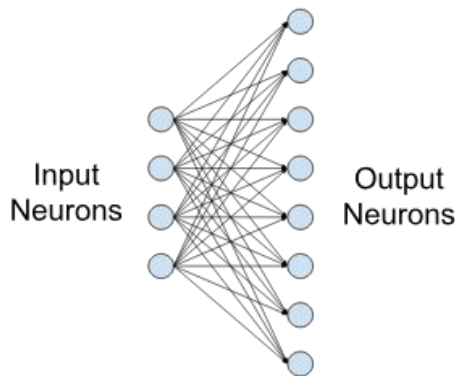
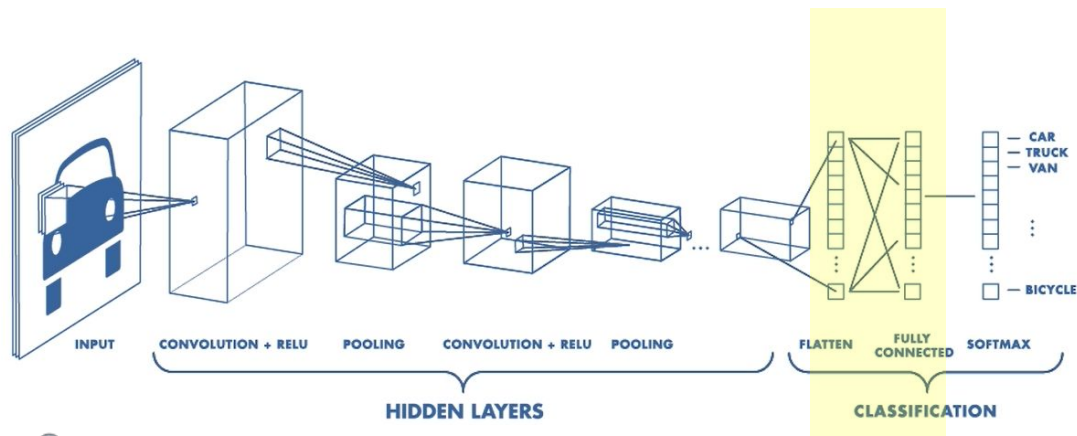


- The pooling operation involves a fixed 2D filter over each channel of feature map and summarising the features lying within the region covered by the filter in a single number

- Pooling layers are used to reduce the dimensions of the feature maps. Thus, it reduces the number of parameters to learn and the amount of computation performed in the network. The pooling layer summarises the features present in a region of the feature map generated by a convolution layer

5	3		7			
6			1	9	5	
	9	8				6
8			6			3
4		8	3			11
7			2			6
	6				2	8
		4	1	9		5
			8		7	9

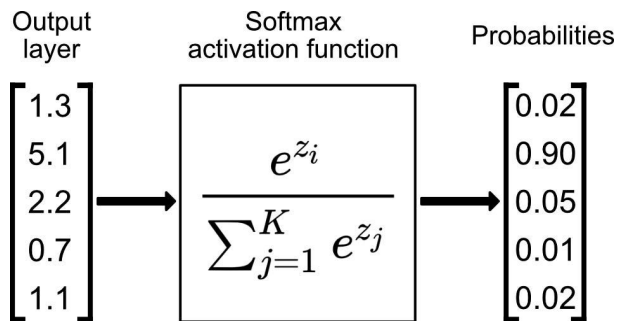
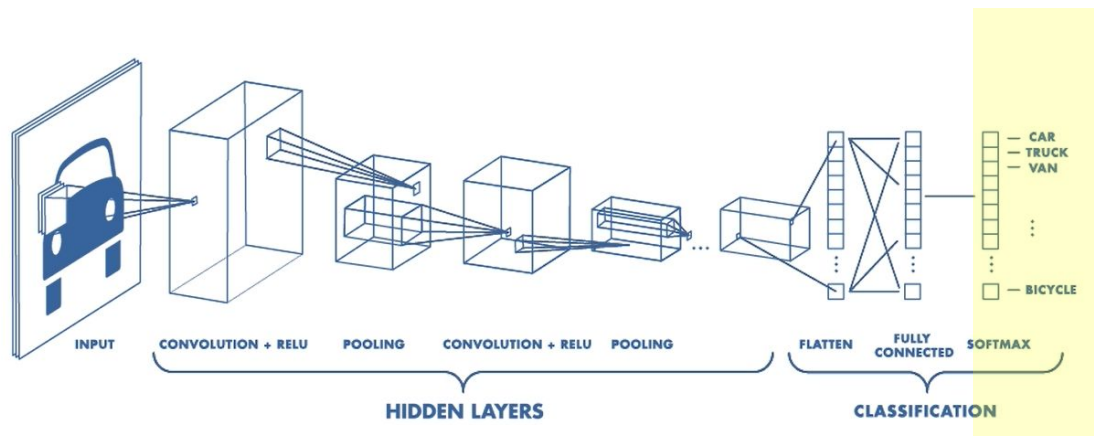
# Convolutional Neural Networks



- CNN contains also at least a couple of fully connected layers, therefore “flattening” the stacking geometry of the convolutional layers
- Used to predict the class of the image based on the features extracted in previous stages

5	3		7			
6			1	9	5	
	9	8				6
8			6			3
4		8	3			1
7			2			6
	6				2	8
		4	1	9		5
			8		7	9

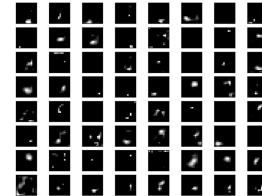
# Convolutional Neural Networks



- Final classification output with a softmax function
- The softmax function is a mathematical function that maps a vector of K real numbers to a vector of K real numbers in the range (0, 1) that add up to 1. It is often used in the output layer of a neural network to produce a probability distribution over multiple classes.

5	3		7			
6		1	9	5		
	9	8			6	
8			6			3
4		8	3			11
7		2			6	
	6			2	8	
		4	1	9		5
			8		7	9

# Feature/ Activation Maps



<https://machinelearningmastery.com/how-to-visualize-filters-and-feature-maps-in-convolutional-neural-networks/>

- The obtained feature maps and activation maps of the convolution layers are not usually/easily directly interpretable for humans

5	3		7			
6			1	9	5	
	9	8				6
8			6			3
4		8	3			1
7			2			6
	6				2	8
		4	1	9		5
			8		7	9

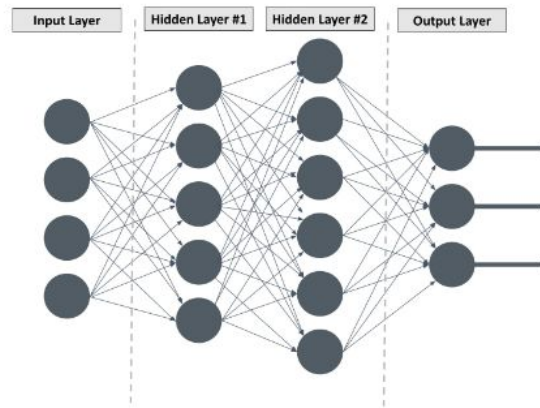
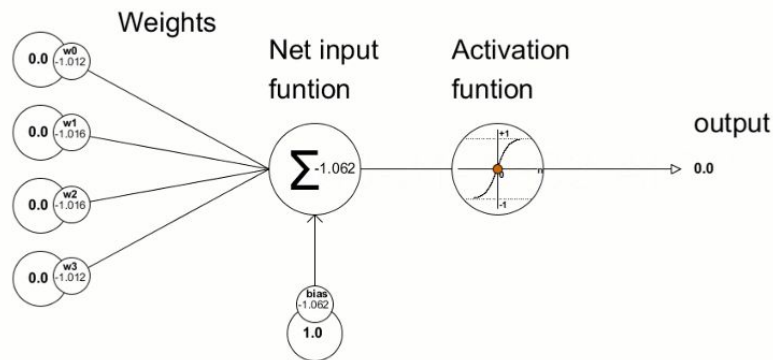
# Convolutional Neural Networks

There is a full equivalence between a Classical NN layer and Conv layer:

(<https://www.arxiv-vanity.com/papers/1712.01252/>)

- A classical NN layer consists of (fully connected) neurons ( which are a inputs, a set of weights, and an activation function)
- The convolutional layers have **sparse connectivity** for the filter/kernel (which is a set of weights) - there is a **shared weights aspect** - and an activation function (e.g. RELU) and intrinsic 2D/3D geometric structure
- Convolutional operation can be converted to matrix multiplication, which has the same calculation way with fully connected layer.

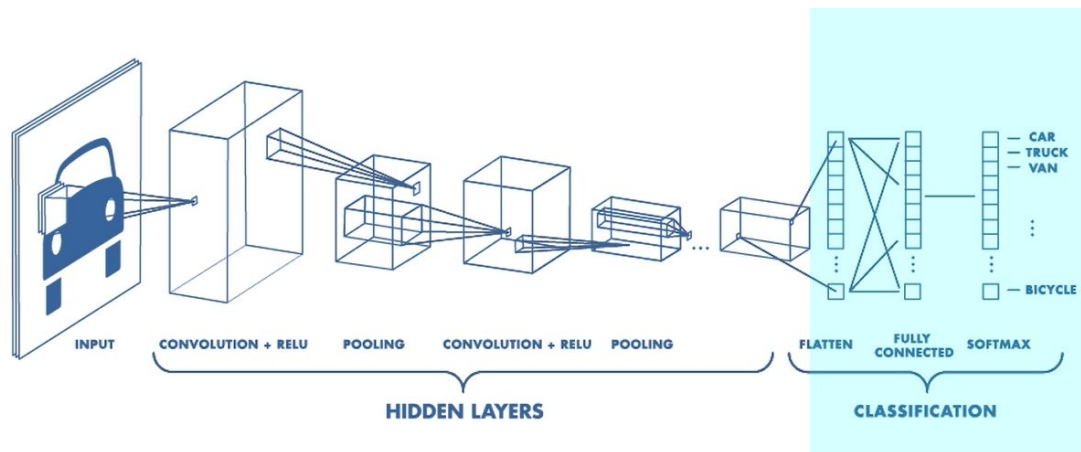
Inputs



The skeleton of a neural network

5	3		7			
6			1	9	5	
	9	8				6
8			6			3
4		8	3			1
7			2			6
	6				2	8
		4	1	9		5
			8		7	9

# Convolutional Neural Networks - Transfer Learning



- use the Convolutional Layers that extract features, re-train the “front-end” on the smaller/specific dataset
- For e.g. Keras API (with TensorFlow backend) comes with pre-trained models for image processing: <https://keras.io/api/applications/#available-models> “Keras Applications are deep learning models (architectures) that are made available alongside pre-trained weights. These models can be used for prediction, feature extraction, and fine-tuning”



5	3		7			
6		1	9	5		
	9	8			6	
8			6			3
4		8	3			1
7		2				6
	6			2	8	
		4	1	9		5
			8		7	9

# Convolutional Neural Networks

<https://www.ibm.com/topics/convolutional-neural-networks>

- Kunihiko Fukushima and Yann LeCun laid the foundation of research around convolutional neural networks in their work in “Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position” in 1980 and "Backpropagation Applied to Handwritten Zip Code Recognition" in 1989, respectively
- Yann LeCun successfully applied backpropagation to train neural networks to identify and recognize patterns within a series of handwritten zip codes. He would continue his research with his team throughout the 1990s, culminating with “LeNet-5”, which applied the same principles of prior research to document recognition
- Since then, a number of variant CNN architectures have emerged with the **introduction of new datasets**, such as MNIST and CIFAR-10, and competitions, like ImageNet Large Scale Visual Recognition Challenge (ILSVRC). Popular architectures include: AlexNet, VGGNet, GoogLeNet, ResNet, ZFNet

5	3		7			
6		1	9	5		
	9	8			6	
8			6			3
4		8		3		1
7			2			6
	6				2	8
		4	1	9		5
			8		7	9

# Convolutional Neural Networks

<https://www.ibm.com/topics/convolutional-neural-networks>

- Kunihiko Fukushima and Yann LeCun laid the foundation of research around convolutional neural networks in their work in “Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position” in 1980 and "Backpropagation Applied to Handwritten Zip Code Recognition" in 1989, respectively
- Yann LeCun successfully applied backpropagation to train neural networks to identify and recognize patterns within a series of handwritten zip codes. He would continue his research with his team throughout the 1990s, culminating with “LeNet-5”, which applied the same principles of prior research to document recognition
- Since then, a number of variant CNN architectures have emerged with the **introduction of new datasets**, such as MNIST and CIFAR-10, and competitions, like ImageNet Large Scale Visual Recognition Challenge (ILSVRC). Popular architectures include: AlexNet, VGGNet, GoogLeNet (aka Inception), ResNet, ZFNet

5	3		7			
6		1	9	5		
	9	8			6	
8			6			3
4		8	3			1
7		2				6
	6			2	8	
		4	1	9		5
		8		7	9	

# Tesseract OCR

- The architectures used by Tesseract OCR can vary depending on the version and the underlying implementation and configuration options
- In the newer versions ( $\geq 4.0.0$ ) Tesseract does not use a traditional CNN architecture like one might find in VGG, ResNet, etc. Instead, a recurrent neural network (RNN) based approach, specifically the Long Short-Term Memory (LSTM) architecture, for recognizing line-level text in images. This LSTM-based model enables Tesseract to recognize more complex scripts and improve overall accuracy compared to its older versions.
- The use of LSTM is particularly suitable for sequence prediction problems, which include handwriting recognition and other tasks where the context (i.e., the sequence of characters) is essential.
- It's worth noting that while LSTMs have been very effective in Tesseract's OCR tasks, in other domains, such as image classification, CNNs have shown to be more effective. The choice of neural network architecture often depends on the nature of the problem being addressed.
- For single digit recognition like Sudoku Tesseract might not be most suitable choice

5	3		7			
6			1	9	5	
	9	8				6
8			6			3
4		8	3			1
7		2				6
	6			2	8	
		4	1	9		5
			8		7	9

# OpenCV uses a CRNN architecture

- [https://docs.opencv.org/4.x/d4/d43/tutorial\\_dnn\\_text\\_spotting.html](https://docs.opencv.org/4.x/d4/d43/tutorial_dnn_text_spotting.html)
- Convolutional Recurrent Neural Network architecture is used for text recognition in OpenCV's text recognition module. The CRNN architecture combines convolutional layers (CNN) and recurrent layers (RNN), making it well-suited for recognizing text in images with varying lengths.