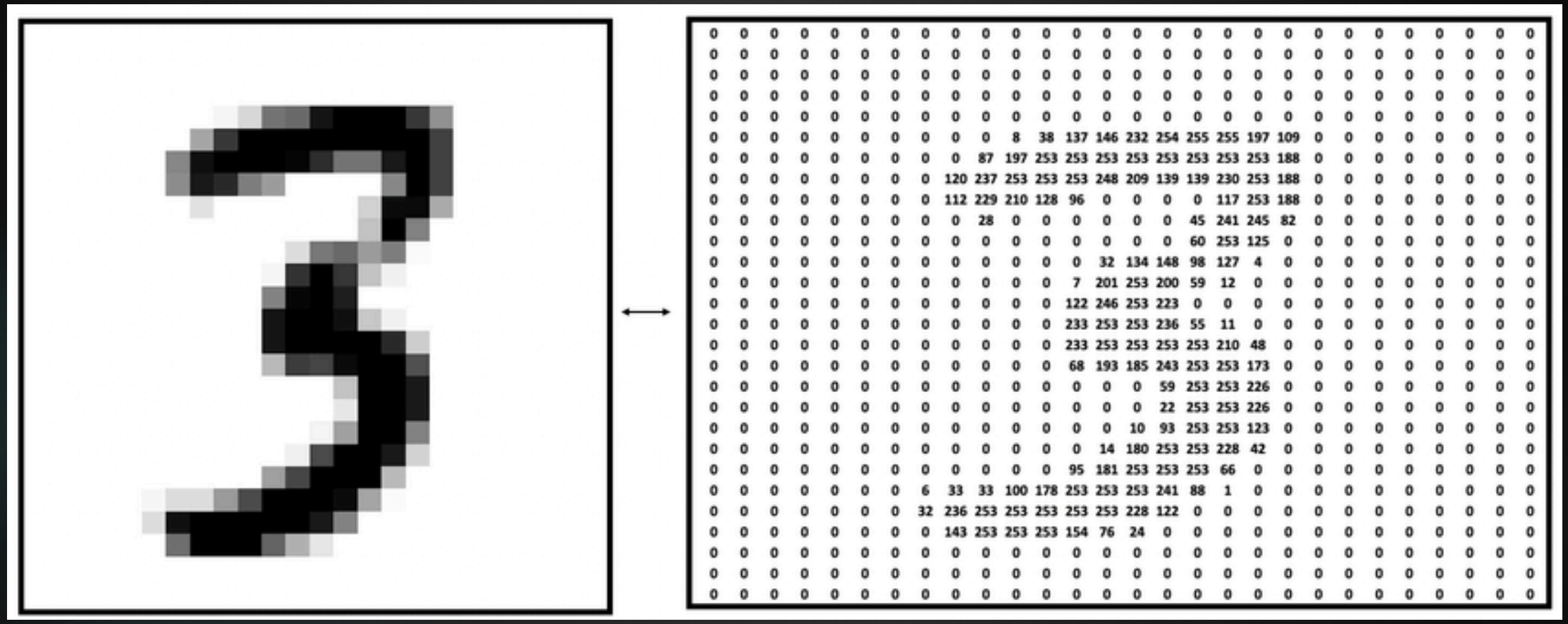
Convolutional neural networks

GUILLEM GUIGÓ I COROMINAS

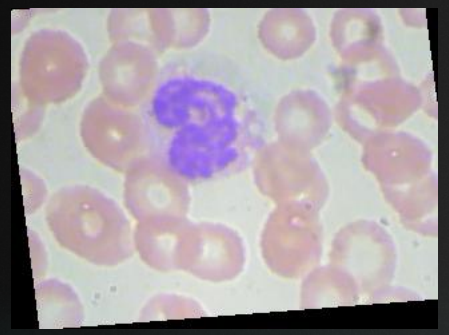
Digital images



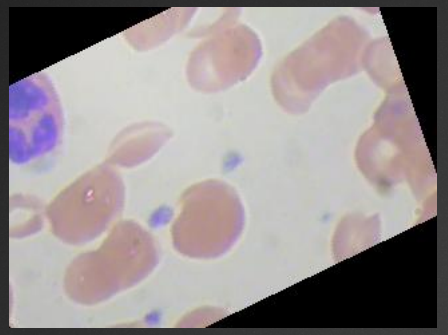
$1\times28\times28$ or $28\times28\times1$

https://www.researchgate.net/figure/Representation-of-value-three-in-the-MNIST-dataset-and-its-equivalent-matrix_fig1_361444345

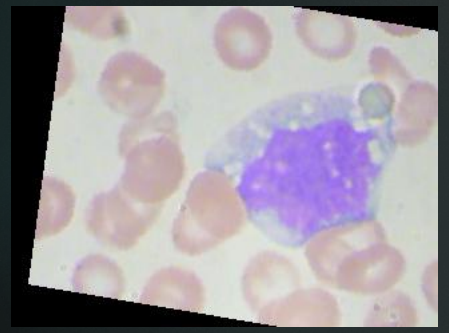
Digital images



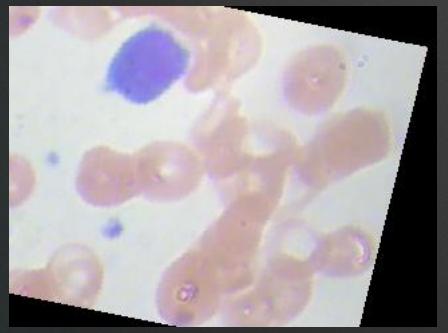
Neutrophil



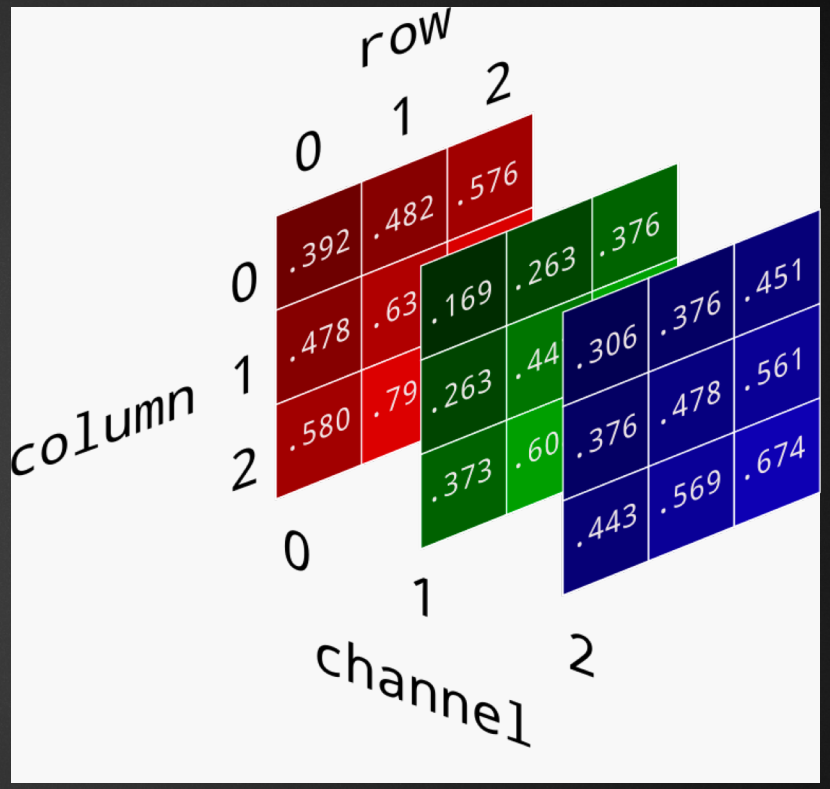
Eosinophil



Monocyte



Lymphocyte



3×240×320 or 240×320×3

https://github.com/Shenggan/BCCD_Dataset

https://e2eml.school/convert_rgb_to_grayscale.html

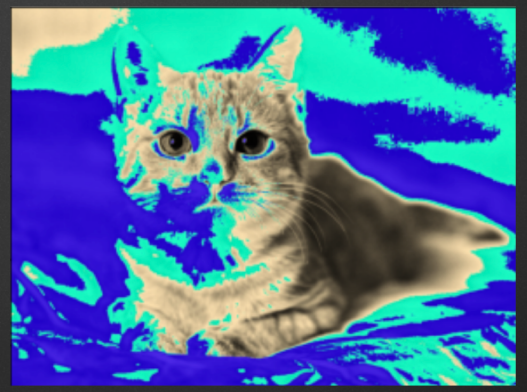
Digital images

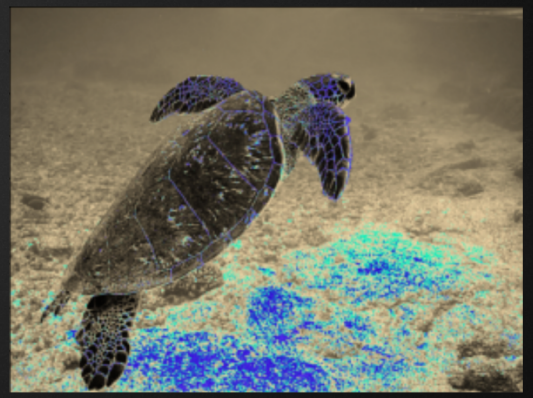
Sepia filter

```
[[.393, .769, .189],
 [.349, .686, .168],
 [.272, .534, .131]]
```

New R = (R * 0.393 + G * 0.769 + B * 0.189)
New G = (R * 0.349 + G * 0.686 + B * 0.168)
New B = (R * 0.272 + G * 0.534 + B * 0.131)

```
im = np.array(Image.open('cat.jpeg'))
sep_im = np.dot(im, sepia_filter.T).astype(np.uint8)
plt.imshow(sep_im)
```





Convolutions in images

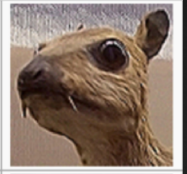
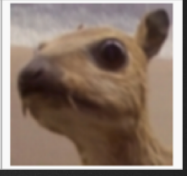
0	0	0	0	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	0	0	0	0

	0	1	0	
	1	-4	1	
	0	1	0	

https://dennybritz.com/posts/wildml/understanding-convolutional-neural-networks-for-nlp/

Convolutions in images

Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	

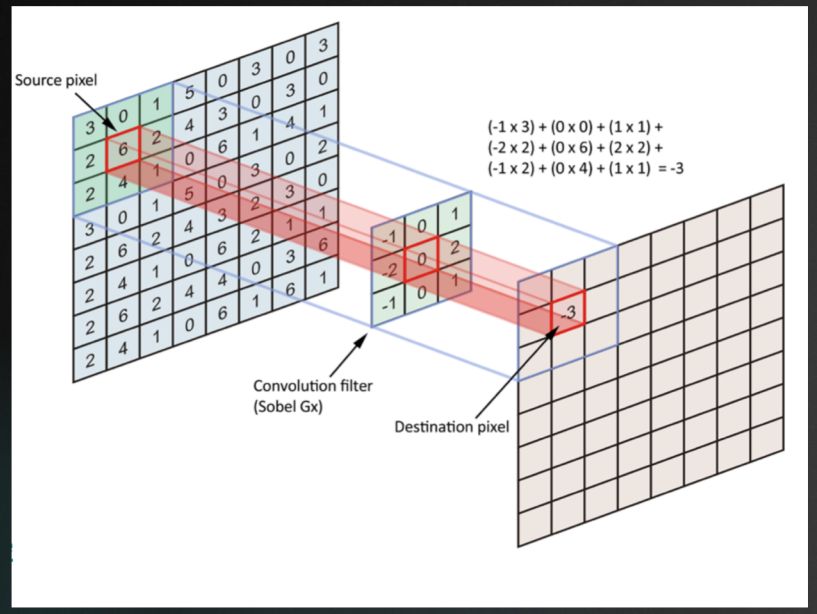
Operation	Filter	Convolved Image
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

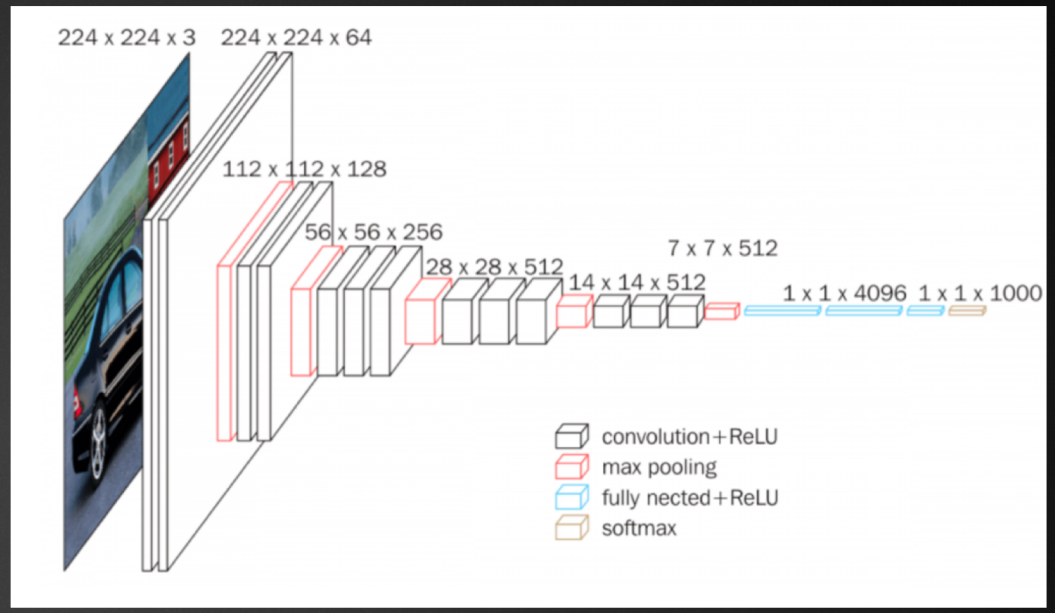
https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

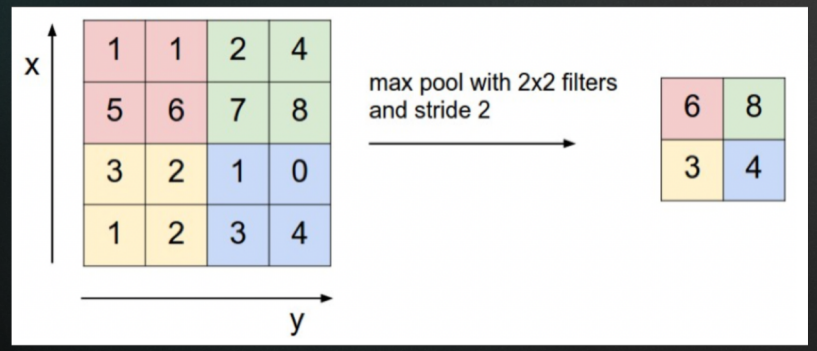
Convolutional neural networks (CNNs)

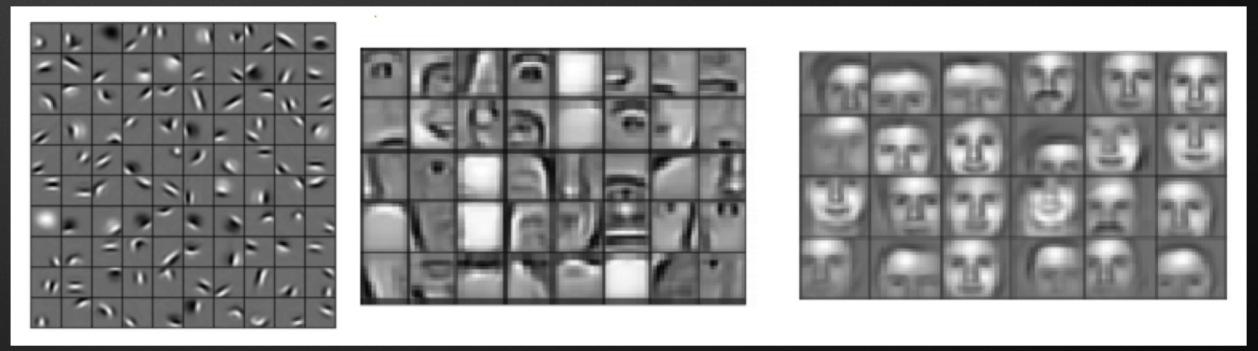
- We can parametrize the kernels and the parameters can be learnt
- **Convolutional layers** in a convolutional neural network systematically apply learned filters to input images in order to create feature maps that summarize the presence of those features in the input.
- A lower resolution version of an input signal is created that still contains the large or important structural elements
- Down sampling can be achieved with convolutional layers by changing the stride of the convolution across the image.
- **Pooling layers** provide an approach to down sampling feature maps by summarizing the presence of features in patches of the feature map.

Convolutional neural networks (CNNs)







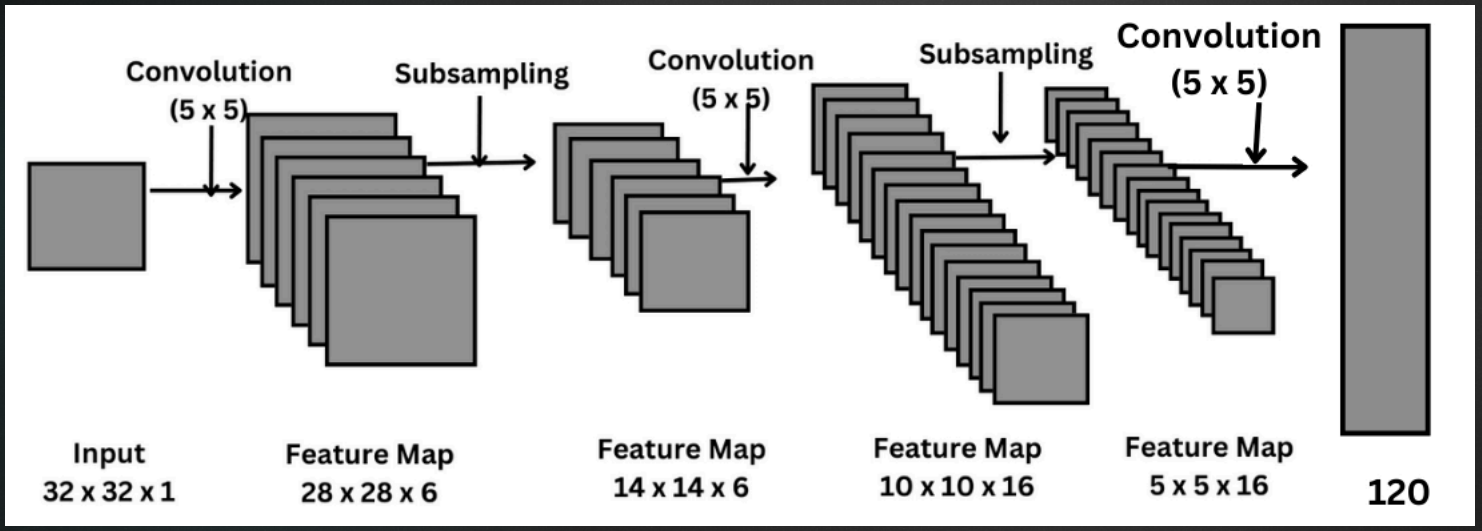


Hachilif, Radja, Riyadh Baghdadi, and Fatima Benhamida. *Graduation Thesis Implementing and Optimizing Neural Networks using Tiramisu*. Diss. PhD thesis. Ecole Nationale Supérieure d'Informatique, 2019.

Güngör, Cengiz, and Kenan Zengin. "GE-International Journal of Engineering Research."

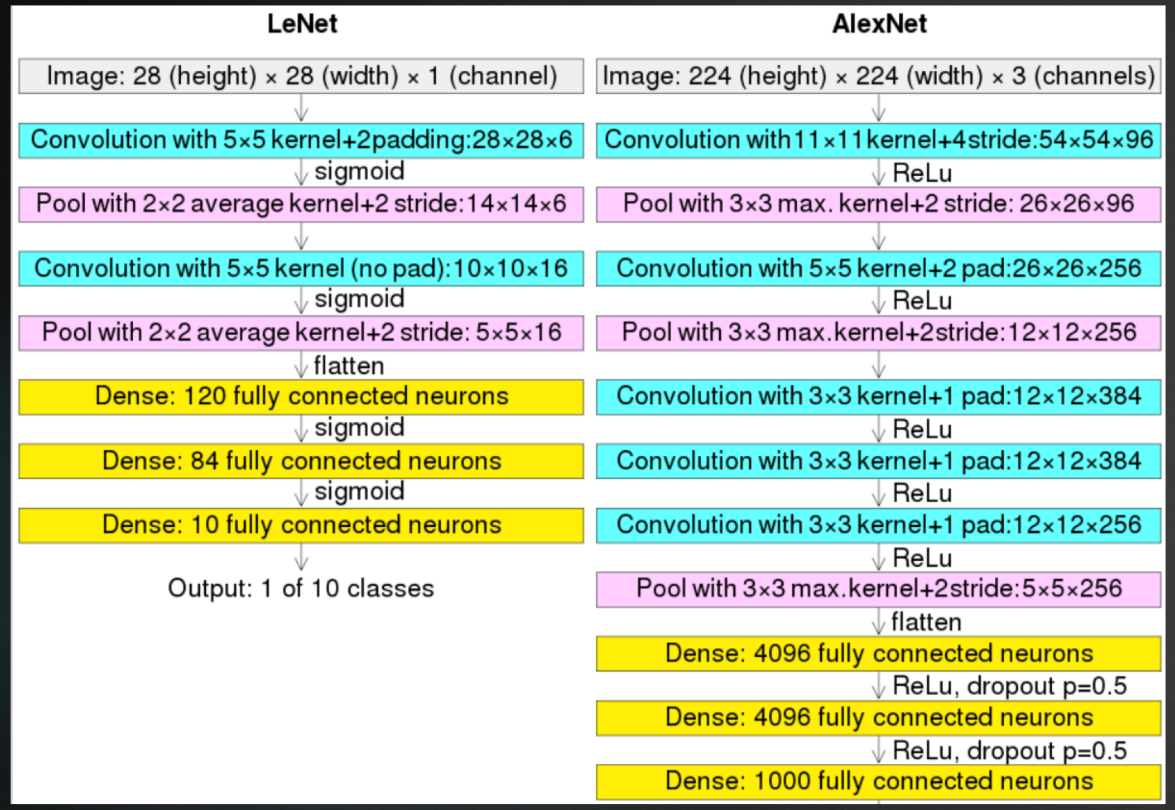
LeNet-5

- CNN structure proposed by Yann Lecun in 1998
- LeNet-5 network has 5 layers (hence its name)
- 3 sets of convolution layers with a combination of average pooling, followed by 2 fully connected layers with a Softmax classifier



Samat, Nurul Ashikin, Mohd Najib Mohd Salleh, and Haseeb Ali. "The comparison of pooling functions in convolutional neural network for sentiment analysis task." *Recent Advances on Soft Computing and Data Mining: Proceedings of the Fourth International Conference on Soft Computing and Data Mining (SCDM 2020), Melaka, Malaysia, January 22– 23, 2020*. Springer International Publishing, 2020.

LeNet-5



Layer (type)	Output Shape	Param #
conv2d_15 (Conv2D)	(None, 28, 28, 6)	456
max_pooling2d_8 (MaxPooling 2D)	(None, 14, 14, 6)	0
conv2d_16 (Conv2D)	(None, 10, 10, 16)	2416
max_pooling2d_9 (MaxPooling 2D)	(None, 5, 5, 16)	0
conv2d_17 (Conv2D)	(None, 1, 1, 120)	48120
flatten_2 (Flatten)	(None, 120)	0
dense_3 (Dense)	(None, 84)	10164
dense_4 (Dense)	(None, 10)	850

Total params: 62,006
Trainable params: 62,006
Non-trainable params: 0

Why use CNN?

- Local connectivity: CNNs exploit the spatial structure of data through the local connectivity pattern
- Translational invariance: CNNs can recognize and detect patterns independently of their location in the input
- Parameter efficient: parameters are shared across the input
- Hierarchical feature learning: The initial layers of the network learn simple and low-level features such as edges and textures, while subsequent layers learn more complex and high-level features

Images sources

- Cat: https://www.goodhousekeeping.com/life/pets/a43276342/cat-instagram-captions/
- Turtle: https://en.wikipedia.org/wiki/Sea_turtle
- VGG-16 achitecture: https://pytorch.org/TensorRT/_notebooks/vgg-qat.html#1
- LeNet/Alexnet architectures: https://upload.wikimedia.org/wikipedia/commons/c/cc/Comparison_image_neural_networks.svg