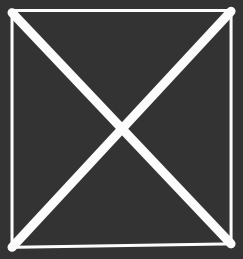
1st Convolutional Layer

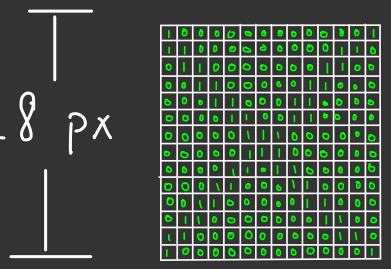
Convolution, Feature Map $\frac{1}{1}$ Activation

Network representation - Fully connected layer vs CNN.

Grayscale img: #1



Pixel representation: #2



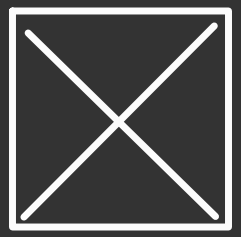
784 pixels

Fully connected network

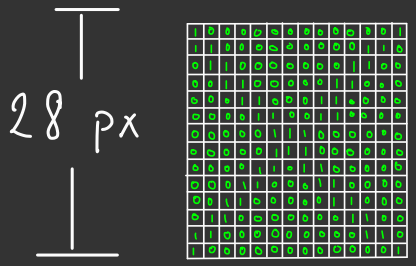
vs

CNN

Grayscale img: #1

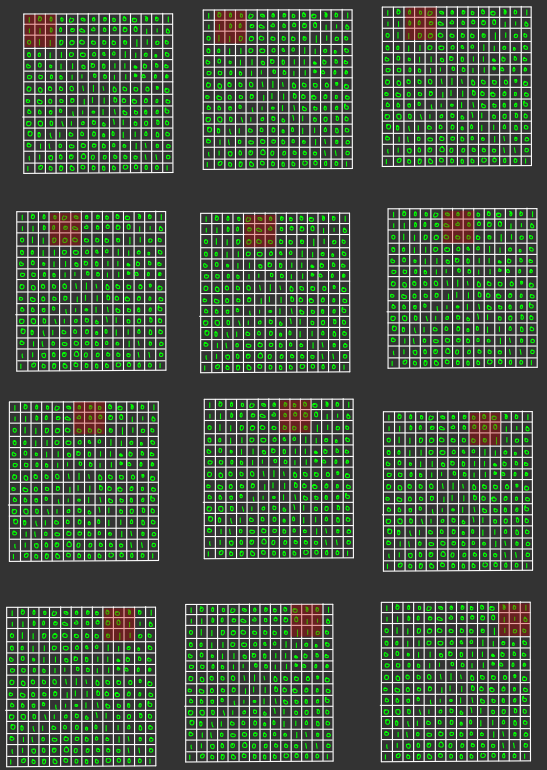


Pixel representation: #2



784 pixels

Also first input, img is treated as a 28x28 grid and passed directly to convolutional layer which processes img in its 2D form.

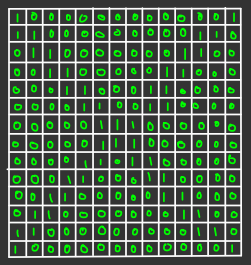


The amount the kernel moves each step depends on the stride size.

↳ The filter is initialized randomly, passing over every section of the input, at each point, the product point is calculated, producing a feature map with the results. The feature map is essentially a transformation of the input data, highlighting areas where certain features are detected, like edges. The filter values get refined through backpropagation.

Input img:

Pixel rep:



Filter (kernel):

0.3	1.1	-0.6
0.4	-0.6	-0.3
0.7	0.1	0 5

Weights = Raw value (Dot product)

Activation Function (Raw value) → Neuron output "0"

f(x)

Feature map (output):

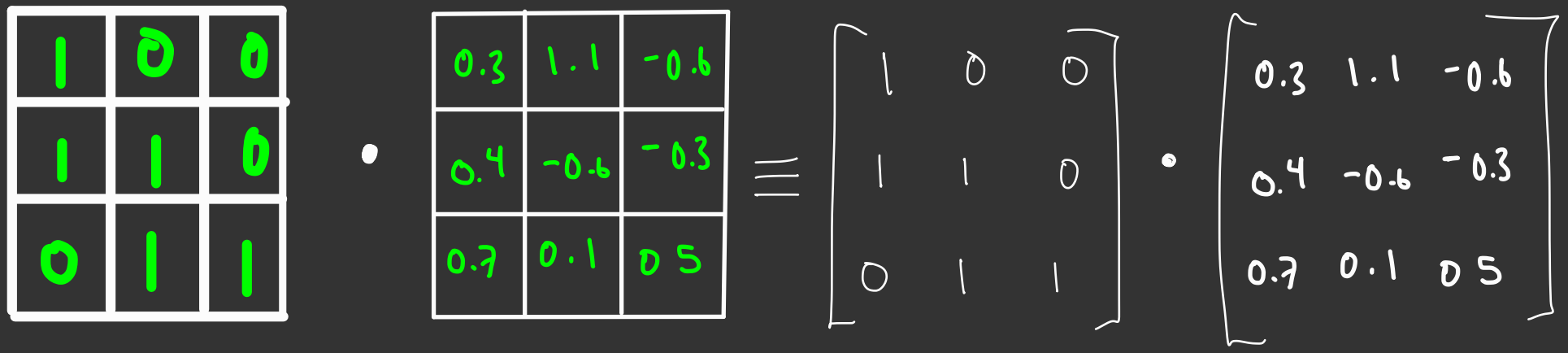
O_1	O_2	O_3
O_4	O_5	O_6
O_7	O_8	O_9

Feature map dimension formula:

$$\text{Output size} = \frac{\text{Input size} - \text{filter size} + 2 \times \text{Padding}}{\text{Stride}} + 1$$

*Different filters can detect different things like edges, textures or other features.

The process defined below is performed every single time a filter moves.



$= (1 \times 0.3) + (0 \times 1.1) + (0 \times -0.6) + (1 \times 0.4) + (1 \times -0.6) +$

$(0 \times -0.3) + (0 \times 0.7) + (1 \times 0.1) + (1 \times 0.5) = 0.7$

$0.7 \rightarrow$ Dot product

Activation function (Dot product) $\rightarrow$ ReLU

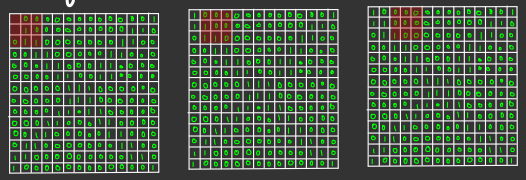
$$ReLU(x) \begin{cases} x, & \text{if } x > 0 \\ 0, & \text{if } x \leq 0 \end{cases}$$

$ReLU(0.7) \rightarrow 0.7$

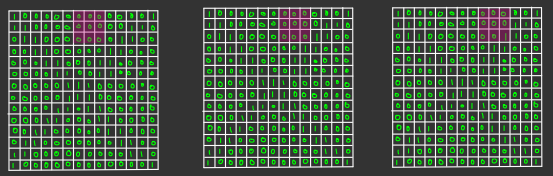
$ReLU(-0.7) \rightarrow 0$

Different filters

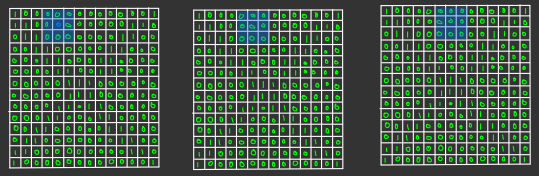
Edges



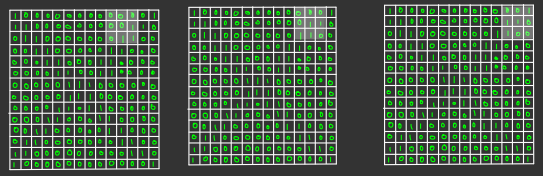
Textures



Corners



complex features



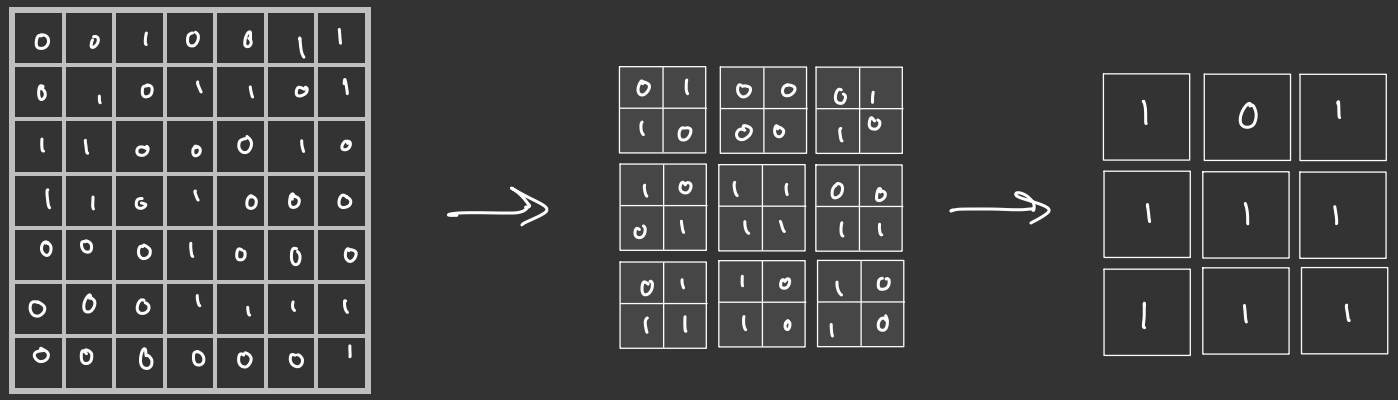
Summary of Filter Types:

- **Early layers** (close to the input): Focus on **low-level features** like edges and textures.
- **Mid layers**: Detect **combinations of these low-level features** to form shapes, like corners or curves.
- **Deeper layers**: Detect more **abstract and specific features**, such as object parts (e.g., eyes, or wheels of a car).

Pooling

MAX Pooling

For each feature map, we ' take a small patch (eg 2x2), and select the maximum value from the patch, reducing the map size while maintaining the most important information



This layer performs downsampling of the feature maps generated by the convolution.

2nd Convolutional Layer

Convolution, Feature Map $\frac{1}{1}^{3}$ Activation

Same Process as the First Convolution: The second convolutional layer applies filters to the pooled feature maps. However, the difference is that now the input feature maps contain higher-level features, so the second convolution detects more complex patterns (like shapes, textures, or combinations of edges).

Again, ReLU

Pooling