Introduction to Convolutional Neural Networks

Komal Teru
Mila, McGill University

Topics to cover today

- Convolution operation: kernel size, stride, padding
- Pooling Operation: kernel size, type of pooling
- Implementations in PyTorch

Dealing with images

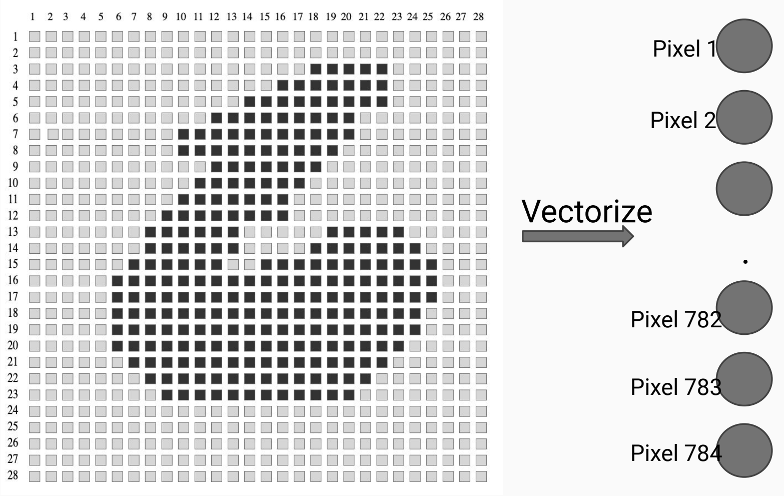
MNIST Dataset

MNIST Dataset: Dataset of handwritten digits. Each image represented as 28 x 28 binary pixels. Thus, dimension: (1 x 28 x 28)

Flatten the image to get a 784 dimensional vector representing each image.

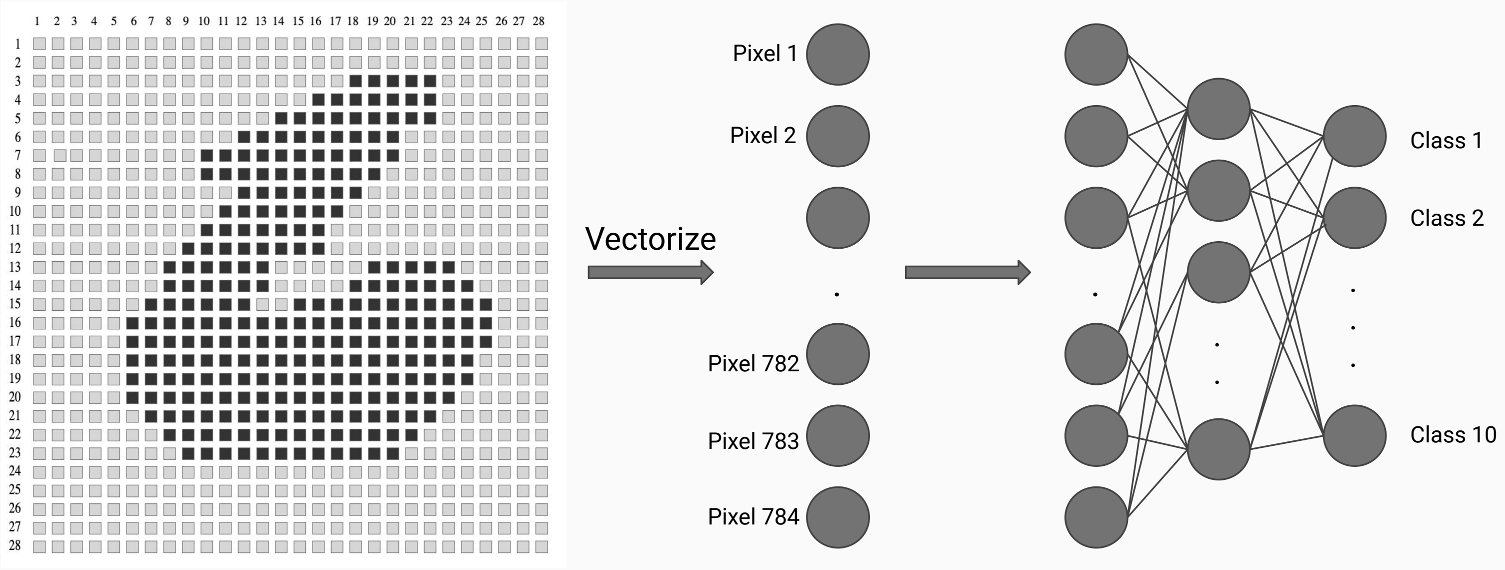
In general, an image has dimensions (C x H x W), where C, H, and W are for channels, height and width.

For color images, C = 3 (R, G, and B)



Dealing with images

Feeding the data to a model



Dealing with images

Feeding the data to a model

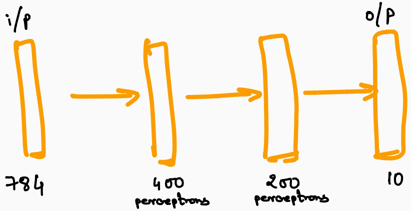
Demo: https://colab.research.google.com/drive/1PYUjTKM1SidodJNyIRJepVsy1L5L1OiM

Disadvantages of vectorizing images

- Loses spatial information inherent to the image
- High resolution images lead to very high number of parameters

Exercise:

How many trainable parameters does the following model have?



Convolution operation

Discrete 1-d convolution

Given two signals, *i*(t) and *f*(t), their convolution is mathematically defined as

$$(i * f)(T) = \sum_{t=0}^{T} i(t) f(T - t)$$

Convolution operation

Discrete 1-d convolution

i	0	0	0	0	0	1	1	1	1	1
f	-1	1								
i*f	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	

Convolution operation

Discrete 1-d convolution

i	0	0	0	0	0	1	1	1	1	1
f		-1	1							
i*f	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	

Convolution operation

Discrete 1-d convolution

i	0	0	0	0	0	1	1	1	1	1
f					-1	1				
i*f	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	

Convolution operation

Discrete 1-d convolution

i	0	0	0	0	0	1	1	1	1	1
f									-1	1
i*f	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	

Convolution operation

Discrete 2-d convolution

$$i = \begin{array}{|c|c|c|} \hline 0 & 80 & 40 \\ \hline 20 & 40 & 0 \\ \hline 0 & 0 & 40 \\ \hline \end{array} \quad * \quad f = \begin{array}{|c|c|} \hline 0 & 0.25 \\ \hline 0.5 & 1 \\ \hline \end{array}$$

Convolution operation

Discrete 2-d convolution

$1\times 0 + 0.5\times 80 + 0.25\times 20 + 0\times 40$

$$\overset{i}{\begin{bmatrix} 0 & 80 & 40 \\ 20 & 40 & 0 \\ 0 & 0 & 40 \end{bmatrix}} * \overset{f}{\begin{bmatrix} 1 & 0.5 \\ 0.25 & 0 \end{bmatrix}} = \begin{bmatrix} 45 & \\ & \end{bmatrix}$$

Convolution operation

Discrete 2-d convolution

$$\overset{i}{\begin{bmatrix} 0 & 80 & 40 \\ 20 & 40 & 0 \\ 0 & 0 & 40 \end{bmatrix}} * \overset{f}{\begin{bmatrix} 1 & 0.5 \\ 0.25 & 0 \end{bmatrix}} = \begin{bmatrix} 45 & 110 \\ & \end{bmatrix}$$

Convolution operation

Discrete 2-d convolution

$$\overset{i}{\begin{bmatrix} 0 & 80 & 40 \\ 20 & 40 & 0 \\ 0 & 0 & 40 \end{bmatrix}} * \overset{f}{\begin{bmatrix} 1 & 0.5 \\ 0.25 & 0 \end{bmatrix}} = \begin{bmatrix} 45 & 110 \\ 40 & \end{bmatrix}$$

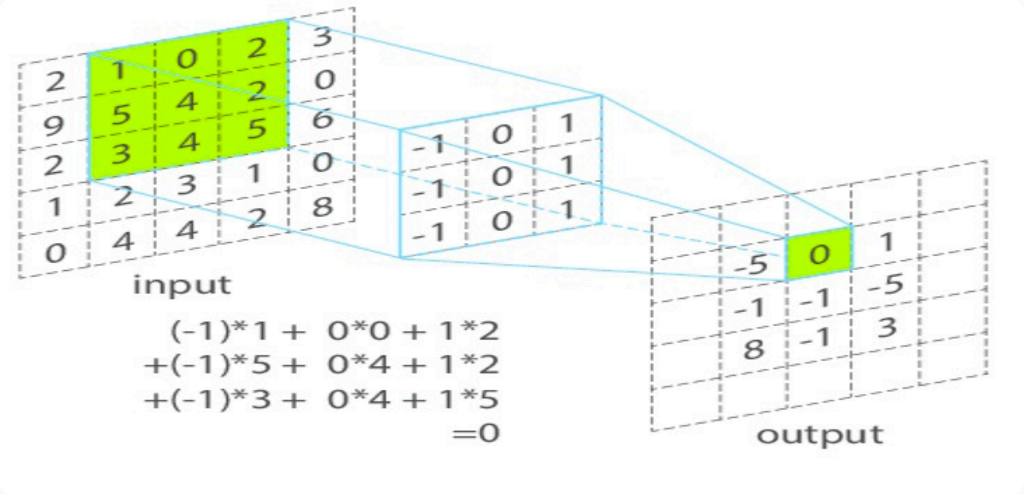
Convolution operation

Discrete 2-d convolution

$$\underset{3\times 3}{\overset{i}{\begin{bmatrix} 0 & 80 & 40 \\ 20 & 40 & 0 \\ 0 & 0 & 40 \end{bmatrix}}} * \underset{2\times 2}{\overset{f}{\begin{bmatrix} 1 & 0.5 \\ 0.25 & 0 \end{bmatrix}}} = \underset{2\times 2}{\overset{i * f}{\begin{bmatrix} 45 & 110 \\ 40 & 40 \end{bmatrix}}}$$

Convolution operation

Discrete 2-d convolution : **another example**



Convolution operation

Discrete 2-d convolution : **Kernel size**

$$\overset{i}{\begin{bmatrix} 0 & 80 & 40 \\ 20 & \boxed{40} & \boxed{0} \\ 0 & \boxed{0} & \boxed{40} \end{bmatrix}} * \overset{f}{\begin{bmatrix} 1 & 0.5 \\ 0.25 & 0 \end{bmatrix}} = \overset{i * f}{\begin{bmatrix} 45 & 110 \\ 40 & 40 \end{bmatrix}}$$

f

2 x 2

$i * f$

Kernel size

Convolution operation

Discrete 2-d convolution : **Stride**

i

$$\begin{bmatrix} 0 & 80 & 40 \\ 20 & 40 & 0 \\ 0 & 0 & 40 \end{bmatrix} * \underbrace{\begin{bmatrix} 1 & 0.5 \\ 0.25 & 0 \end{bmatrix}}_{2 \times 2} = \begin{bmatrix} 45 & 110 \\ 40 & 40 \end{bmatrix}$$

Stride = 1

Convolution operation

Discrete 2-d convolution : **Padding**

- What if we want the output size to be the same as the input size?

- Also, note that the filter isn't able to reach the corners of our input.

Padding allows us to do these things.

Convolution operation

Discrete 2-d convolution : **Padding**

$$
i = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 80 & 40 & 0 \\ 0 & 20 & 40 & 0 & 0 \\ 0 & 0 & 0 & 40 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} * f = \begin{bmatrix} 0 & 0.25 \\ 0.5 & 1 \end{bmatrix} =
$$

What will be the output??

Convolution operation

Discrete 2-d convolution : **Padding**

$$
\underset{i}{\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 80 & 40 & 0 \\ 0 & 20 & 40 & 0 & 0 \\ 0 & 0 & 0 & 40 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}}
*
\underset{f}{\begin{bmatrix} 0 & 0.25 & 0 \\ 0.25 & 0.5 & 1 \\ 0 & 0 & 0.25 \end{bmatrix}}
=
\begin{bmatrix} 90 & 80 & 40 \\ 50 & 55 & ? \\ ? & ? & ? \end{bmatrix}
$$

(this assumes stride = ?)

Convolution operation

Discrete 2-d convolution : **Padding**

$$\underbrace{\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 80 & 40 & 0 \\ 0 & 20 & 40 & 0 & 0 \\ 0 & 0 & 0 & 40 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}}_{i} * \underbrace{\begin{bmatrix} 0 & 0.25 & 0 \\ 0.25 & 0.5 & 1 \\ 0 & 0 & 0.25 \end{bmatrix}}_{f} = ?$$

What if the stride is 2 ?

Convolution operation

Discrete 2-d convolution

Output size formula:

$$\boxed{O = \left\lfloor \frac{i + 2p - k}{s} \right\rfloor + 1}$$

floor function

i : square input size

p : square padding

k : square kernel

s : square stride

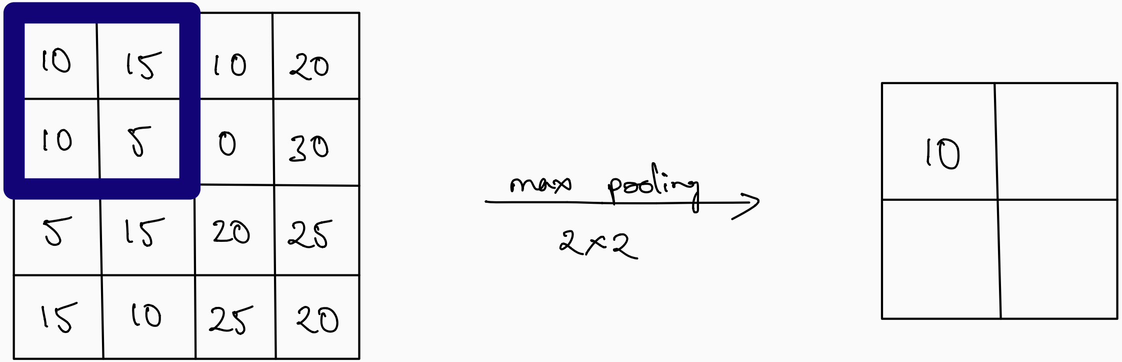
Pooling operation

- Pooling reduces the input size

- Pooling is performed on non-overlapping neighborhoods

Pooling operation

Max pooling



Pooling operation

Max pooling

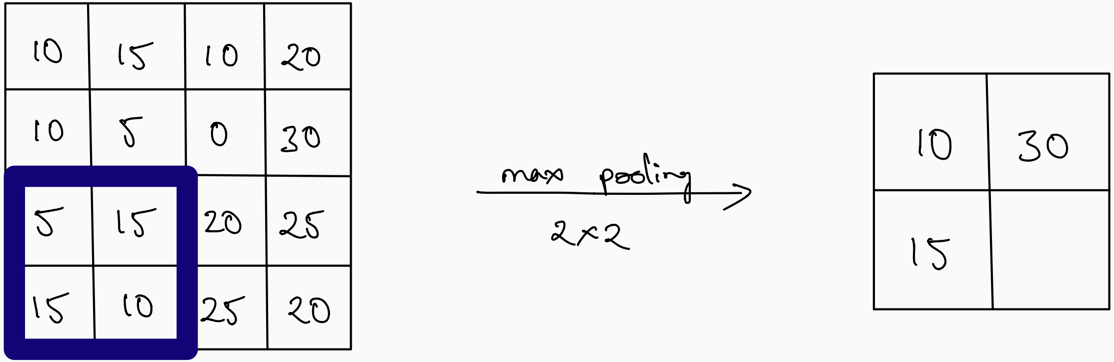
10	15	10	20
10	5	0	30
5	15	20	25
15	10	25	20

$\xrightarrow[2\times2]{\text{max pooling}}$

10	30

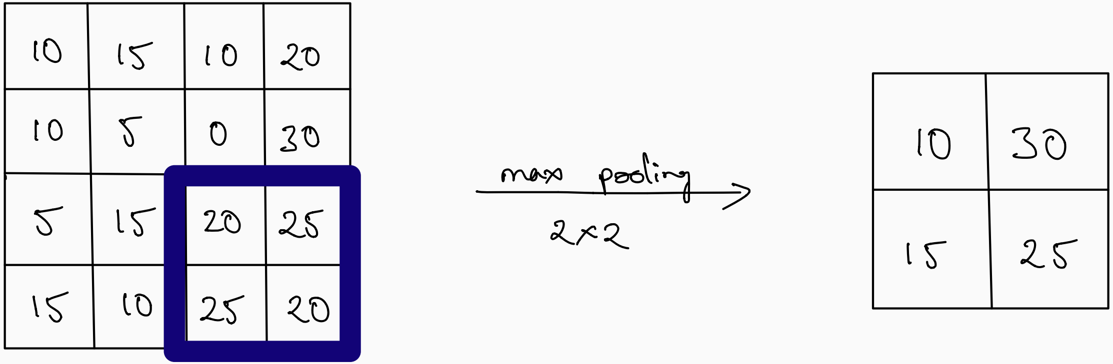
Pooling operation

Max pooling



Pooling operation

Max pooling



Pooling operation

Average pooling

10	15	10	20
10	5	0	30
5	15	20	25
15	10	25	20

$\xrightarrow[2\times 2]{\text{avg pooling}}$

10	15
13.75	22.5

Advantages of convolution and pooling operations

- Local connectivity
- Parameter sharing
- Translation invariance (due to pooling)