CSCI 1470

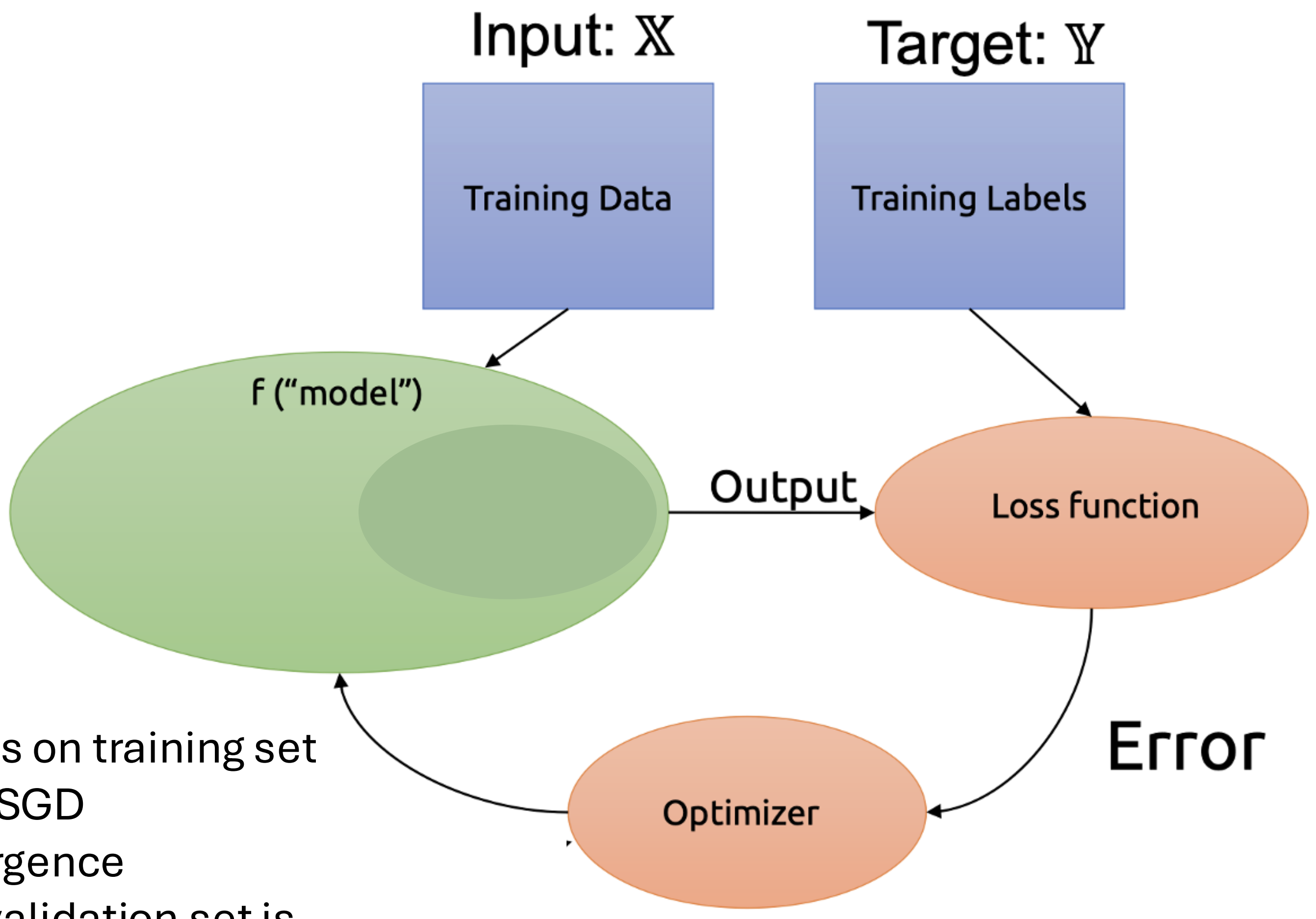
Eric Ewing

Wednesday, 2/12/25

Deep Learning

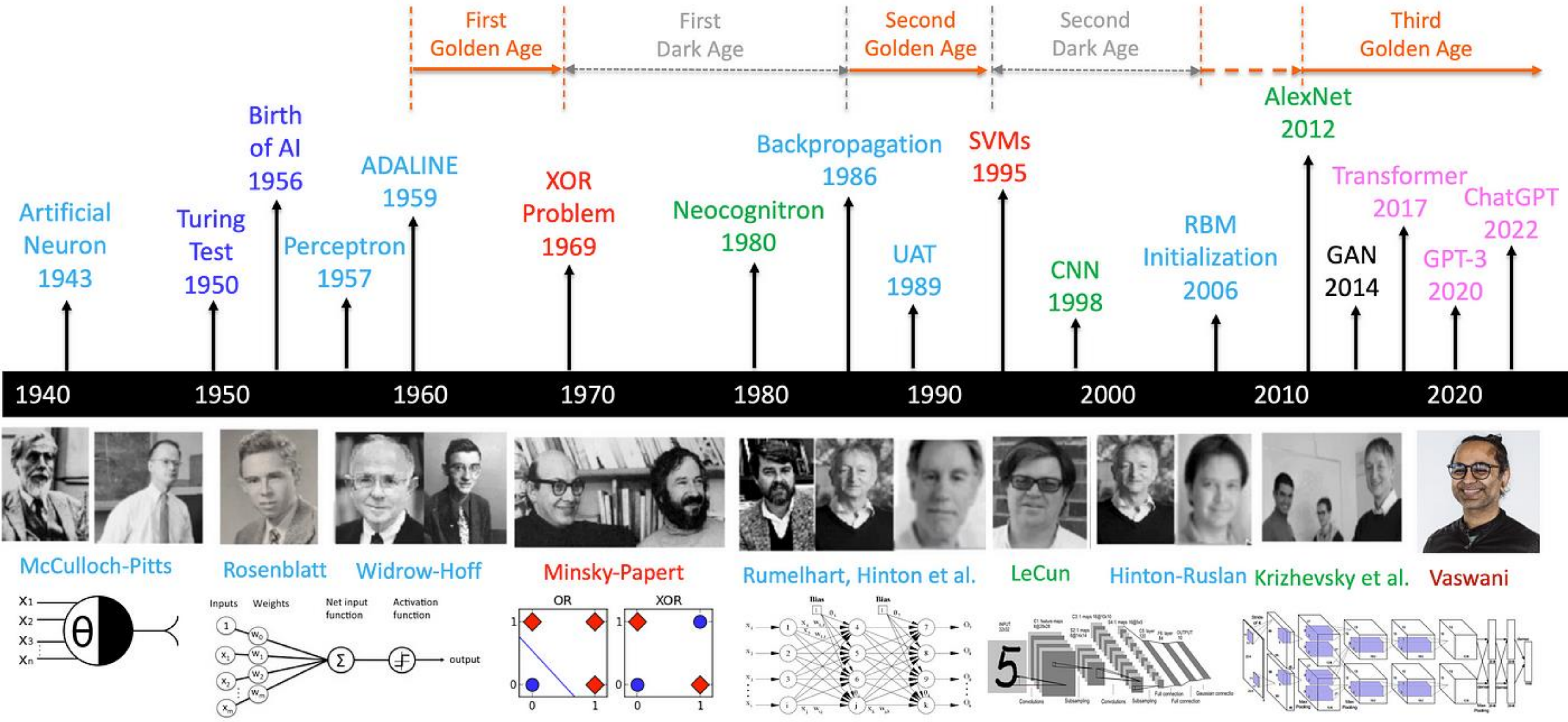
Day 10: Introduction to Convolutions

Recap: MLPs



1. Compute Error/Loss on training set
2. Run Backprop and SGD
3. Repeat until convergence
4. If performance on validation set is acceptable, terminate, else try new hyperparameters

A Brief History of AI with Deep Learning



First Golden Age

First Dark Age

Second Golden Age

Second Dark Age

Third Golden Age

Artificial Neuron 1943

Turing Test 1950

Birth of AI 1956

Perceptron 1957

ADALINE 1959

XOR Problem 1969

Neocognitron 1980

Backpropagation 1986

UAT 1989

SVMs 1995

CNN 1998

RBM Initialization 2006

AlexNet 2012

GAN 2014

Transformer 2017

GPT-3 2020

ChatGPT 2022

1940 1950 1960 1970 1980 1990 2000 2010 2020

McCulloch-Pitts

Rosenblatt

Widrow-Hoff

Minsky-Papert

Rumelhart, Hinton et al.

LeCun

Hinton-Ruslan

Krizhevsky et al.

Vaswani

https://medium.com/@lmpo/a-brief-history-of-ai-with-deep-learning-26f7948bc87b

What has happened in the last 15 years?

What has changed?

1. Power and efficiency of compute (GPUs)
2. Availability of data (the internet)
3. New Architectures (e.g., CNNs, Transformers)

Issues with MLPs

1. Resource Intensive
2. Difficult to incorporate certain types of information
3. (and more)

Issues with MLPs

1. Resource Intensive
2. Difficult to incorporate certain types of information
3. (and more)

GPUs to the rescue!

- *Graphics* Processing Units

- GPUs are really good at computing mathematical operations in parallel!

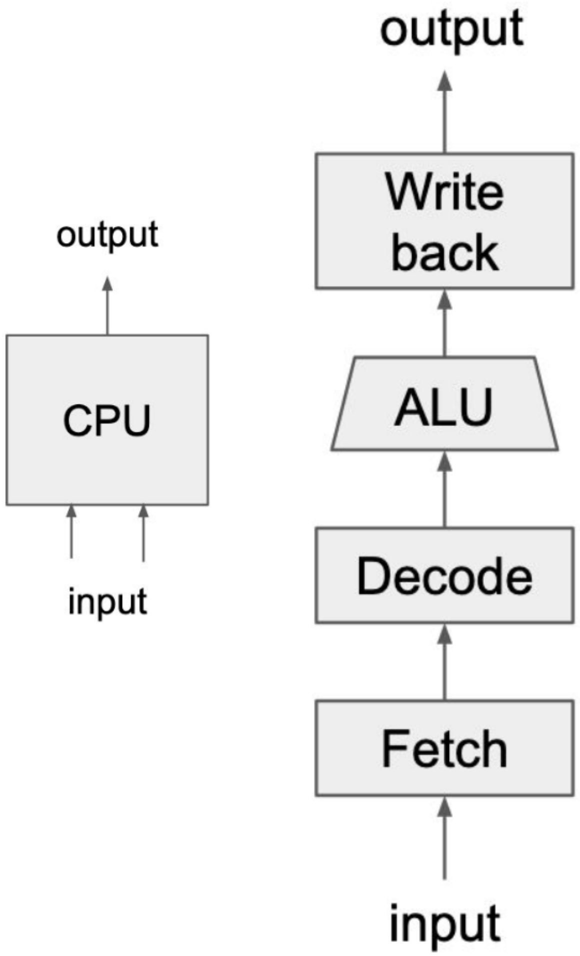
- Matrix multiplication == many **independent** multiply and add operations

  Easily parallelizable

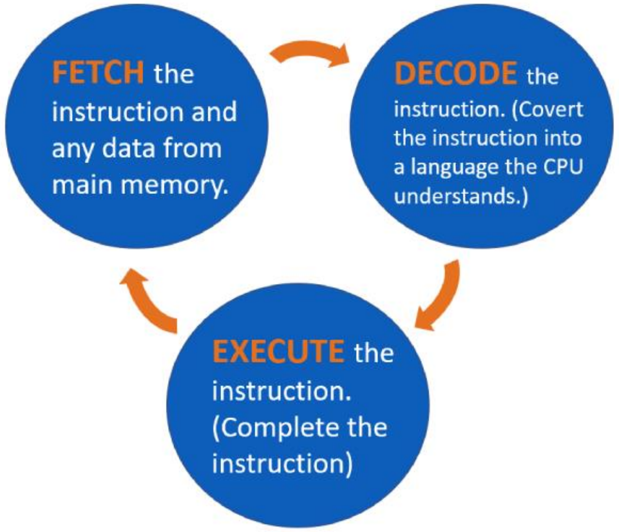
  GPUs are great for this!

Image courtesy: https://global.aorus.com/blog-detail.php?i=878

CPU v/s GPU



Arithmetic logic unit



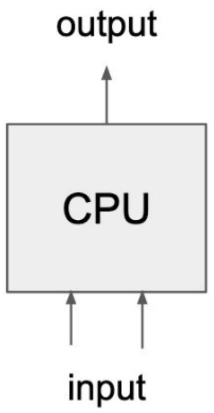
FETCH the instruction and any data from main memory.

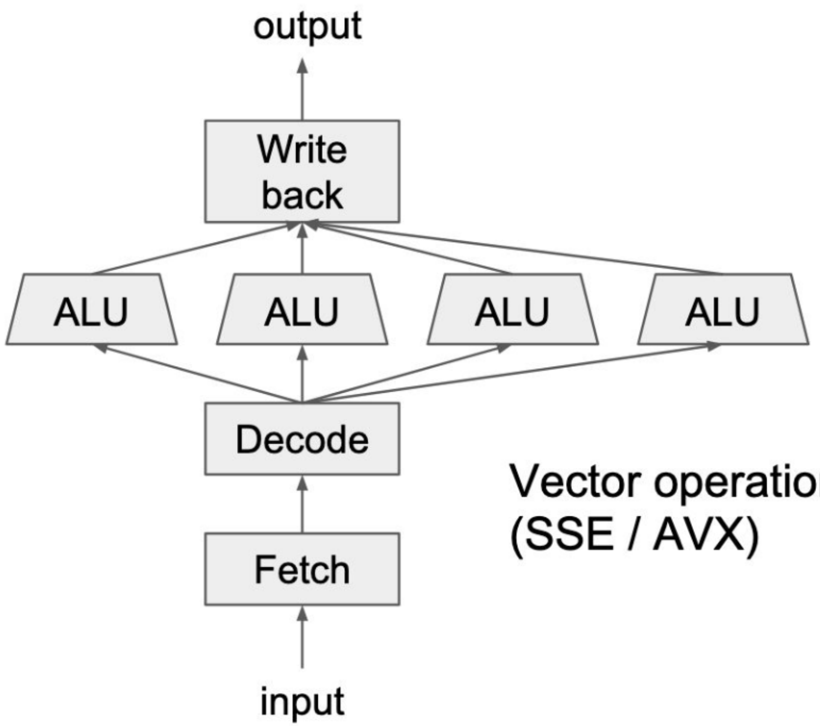
DECODE the instruction. (Covert the instruction into a language the CPU understands.)

EXECUTE the instruction. (Complete the instruction)

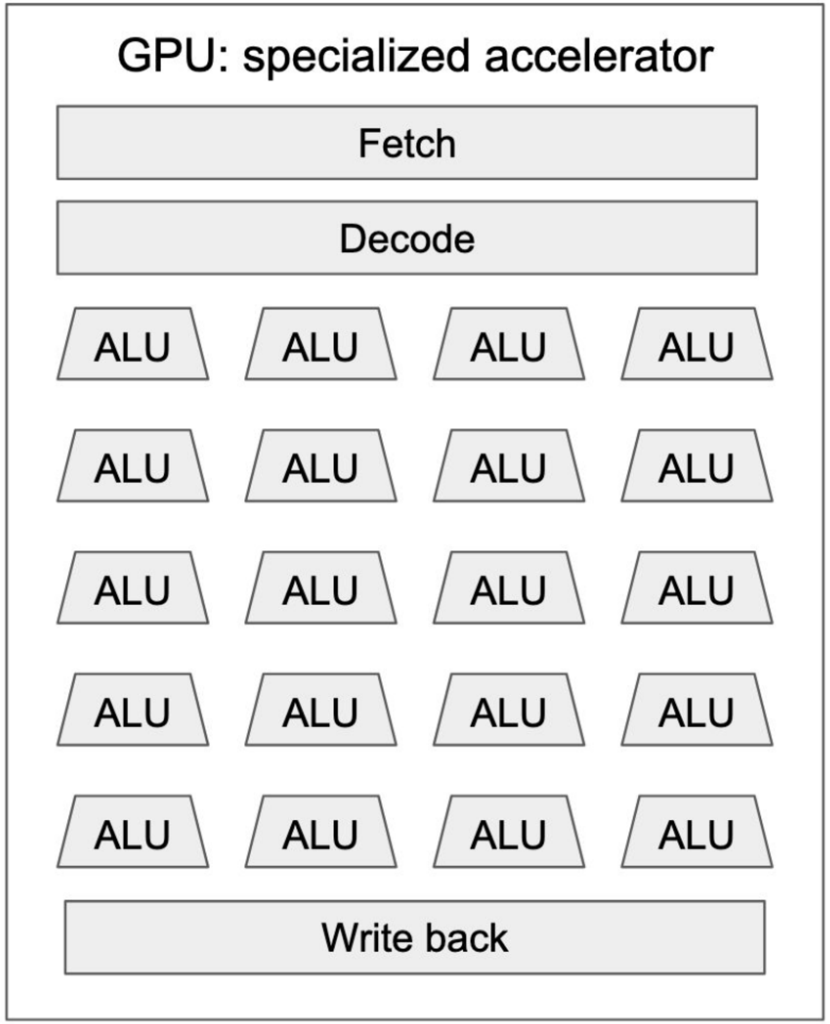
https://dlsys.cs.washington.edu/pdf/lecture5.pdf

CPU v/s GPU





Vector operations (SSE / AVX)



https://dlsys.cs.washington.edu/pdf/lecture5.pdf

GPU-Parallel Acceleration

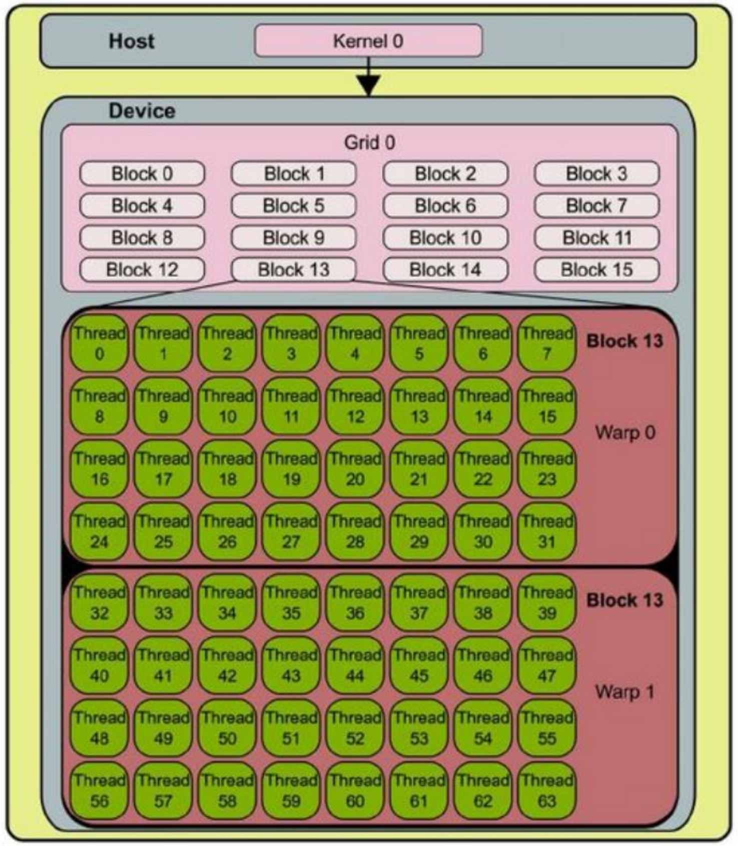
Compute Unified Device Architecture is a parallel computing platform and application programming interface (API)

- User code (***kernels***) is compiled on the ***host*** (the CPU) and then transferred to the ***device*** (the GPU)
- Kernel is executed as a ***grid***
- Each grid has multiple ***thread blocks***
- Each thread block has multiple ***warps***

A warp is the basic schedule unit in kernel execution

A warp consists of 32 threads

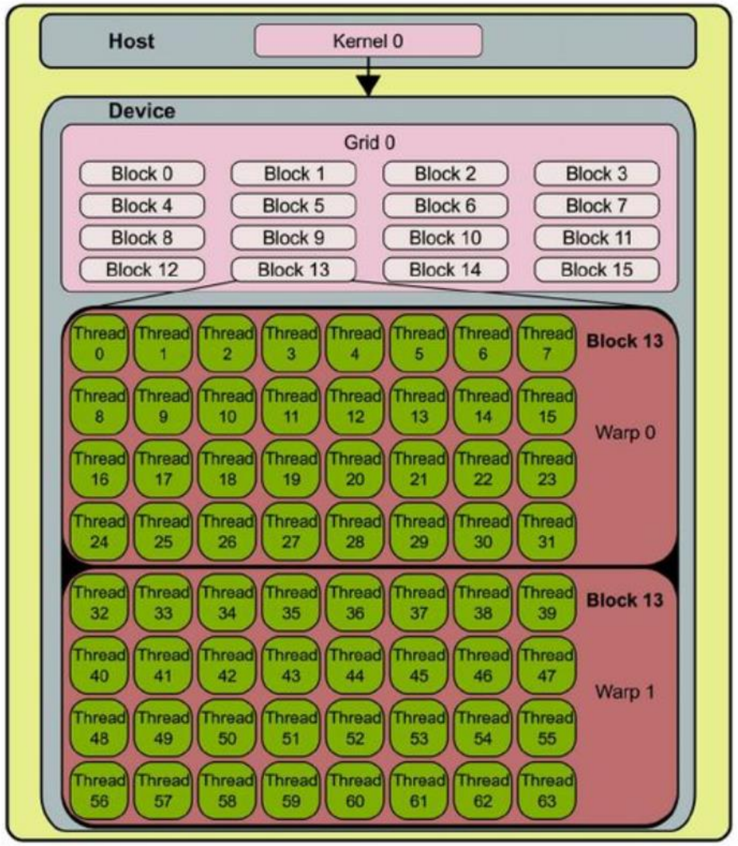
CUDA compute model



https://www.researchgate.net/publication/236666656_Accelerating_Fibre_Orientation_Estimation_from_Diffusion_Weighted_Magnetic_Resonance_Imaging_Using_GPUs

GPU-Parallel Acceleration

CUDA compute model


Host
Kernel 0
Device
Grid 0
Block 0 Block 1 Block 2 Block 3
Block 4 Block 5 Block 6 Block 7
Block 8 Block 9 Block 10 Block 11
Block 12 Block 13 Block 14 Block 15
Thread 0 Thread 1 Thread 2 Thread 3 Thread 4 Thread 5 Thread 6 Thread 7
Block 13
Thread 8 Thread 9 Thread 10 Thread 11 Thread 12 Thread 13 Thread 14 Thread 15
Warp 0
Thread 16 Thread 17 Thread 18 Thread 19 Thread 20 Thread 21 Thread 22 Thread 23
Thread 24 Thread 25 Thread 26 Thread 27 Thread 28 Thread 29 Thread 30 Thread 31
Thread 32 Thread 33 Thread 34 Thread 35 Thread 36 Thread 37 Thread 38 Thread 39
Block 13
Thread 40 Thread 41 Thread 42 Thread 43 Thread 44 Thread 45 Thread 46 Thread 47
Warp 1
Thread 48 Thread 49 Thread 50 Thread 51 Thread 52 Thread 53 Thread 54 Thread 55
Thread 56 Thread 57 Thread 58 Thread 59 Thread 60 Thread 61 Thread 62 Thread 63

- Programmer decides how they want to parallelize the computation across grids and blocks
  - Modern deep learning frameworks take care of this for you

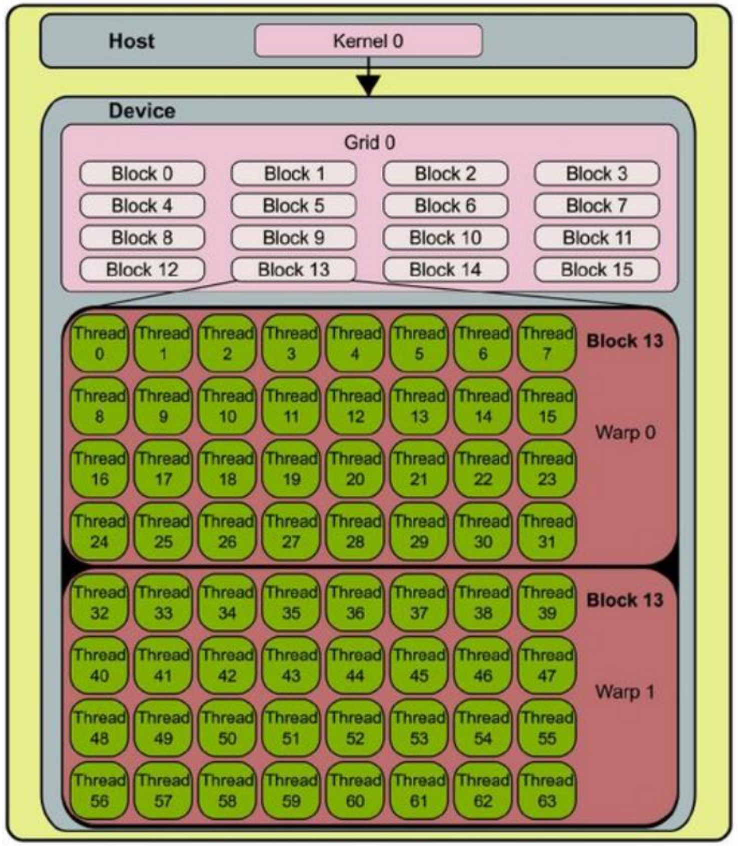
- CUDA compiler figures out how to schedule these units of computation on to the physical hardware

https://www.researchgate.net/publication/236666656_Accelerating_Fibre_Orientation_Estimation_from_Diffusion_Weighted_Magnetic_Resonance_Imaging_Using_GPUs

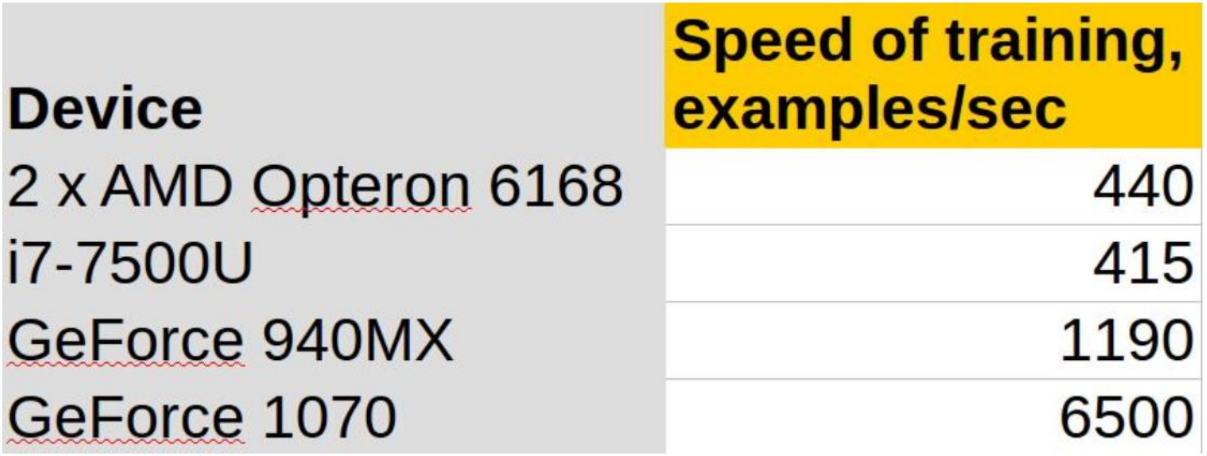
Any questions?

GPU-Parallel Acceleration

CUDA compute model



- Upshot: order of magnitude speedups!
- Example: training CNN on CIFAR-10 dataset

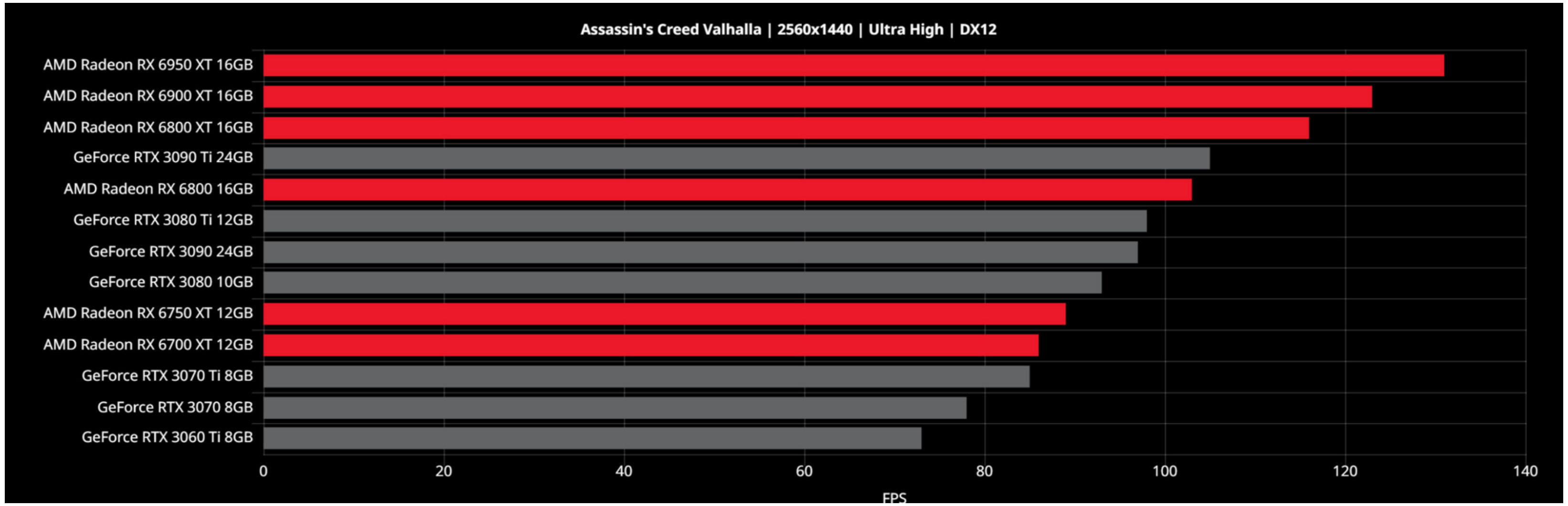


Device	Speed of training, examples/sec
2 x AMD Opteron 6168	440
i7-7500U	415
GeForce 940MX	1190
GeForce 1070	6500

From: https://medium.com/@andriylazorenko/tensorflow-performance-test-cpu-vs-gpu-79fcd39170c

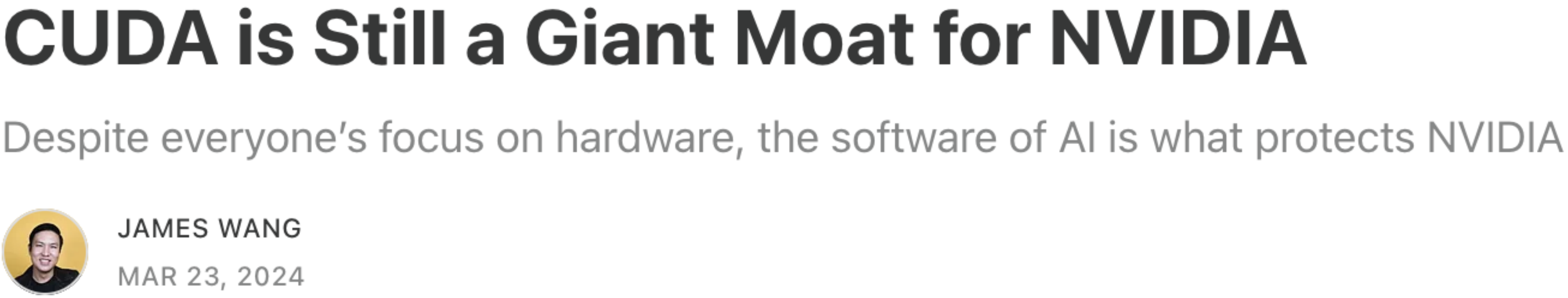
https://www.researchgate.net/publication/236666656_Accelerating_Fibre_Orientation_Estimation_from_Diffusion_Weighted_Magnetic_Resonance_Imaging_Using_GPUs

AMD GPUs are competitive for gaming and graphics, why not for AI?



(With a benchmarking tool made by AMD)

- CUDA is far better than competitors (AMD)
    - Easier to use
    - Better optimization
- AMD makes GPUs for graphics, NVIDIA makes GPUs for AI



CUDA is Still a Giant Moat for NVIDIA

Despite everyone's focus on hardware, the software of AI is what protects NVIDIA

JAMES WANG

MAR 23, 2024

Issues with MLPs

1. Resource Intensive
2. Difficult to incorporate certain types of information

MLPs and Spatial Reasoning

What would happen if we permuted the ordering of the pixels?

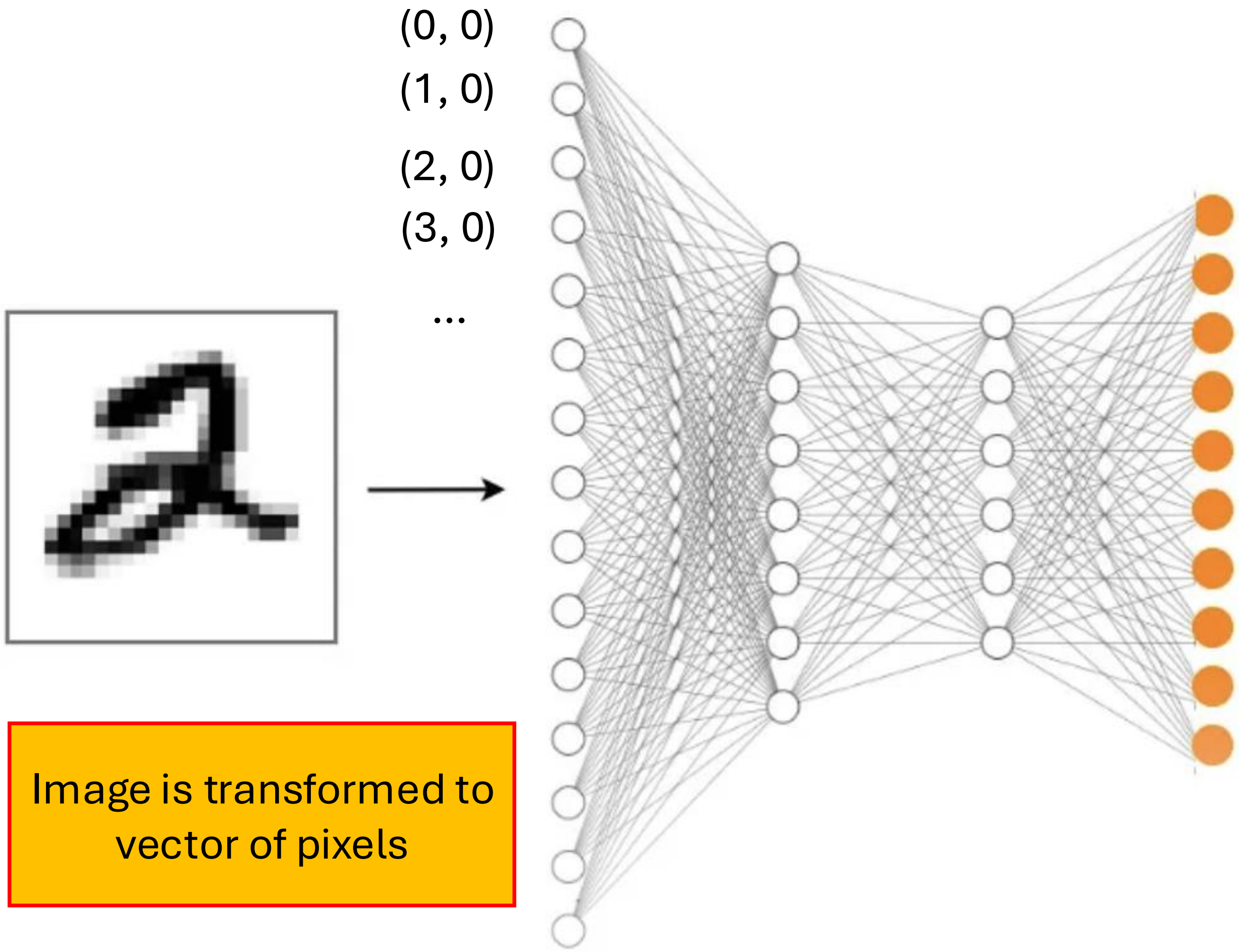


Image is transformed to vector of pixels

https://medium.com/towards-data-science/creating-a-multilayer-perceptron-mlp-classifier-model-to-identify-handwritten-digits-9bac1b16fe10

MLPs and Spatial Reasoning

What would happen if we permuted the ordering of the pixels?

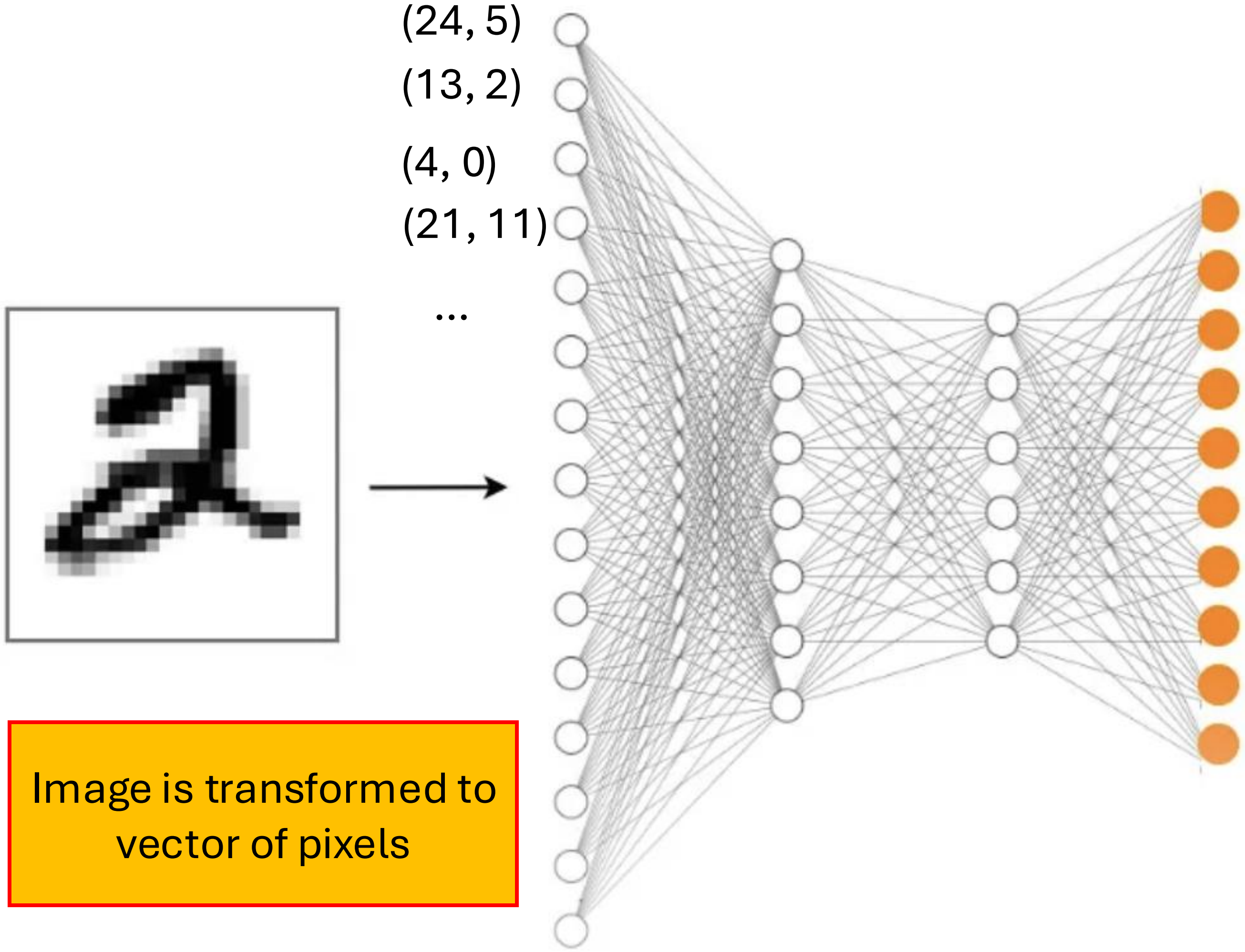


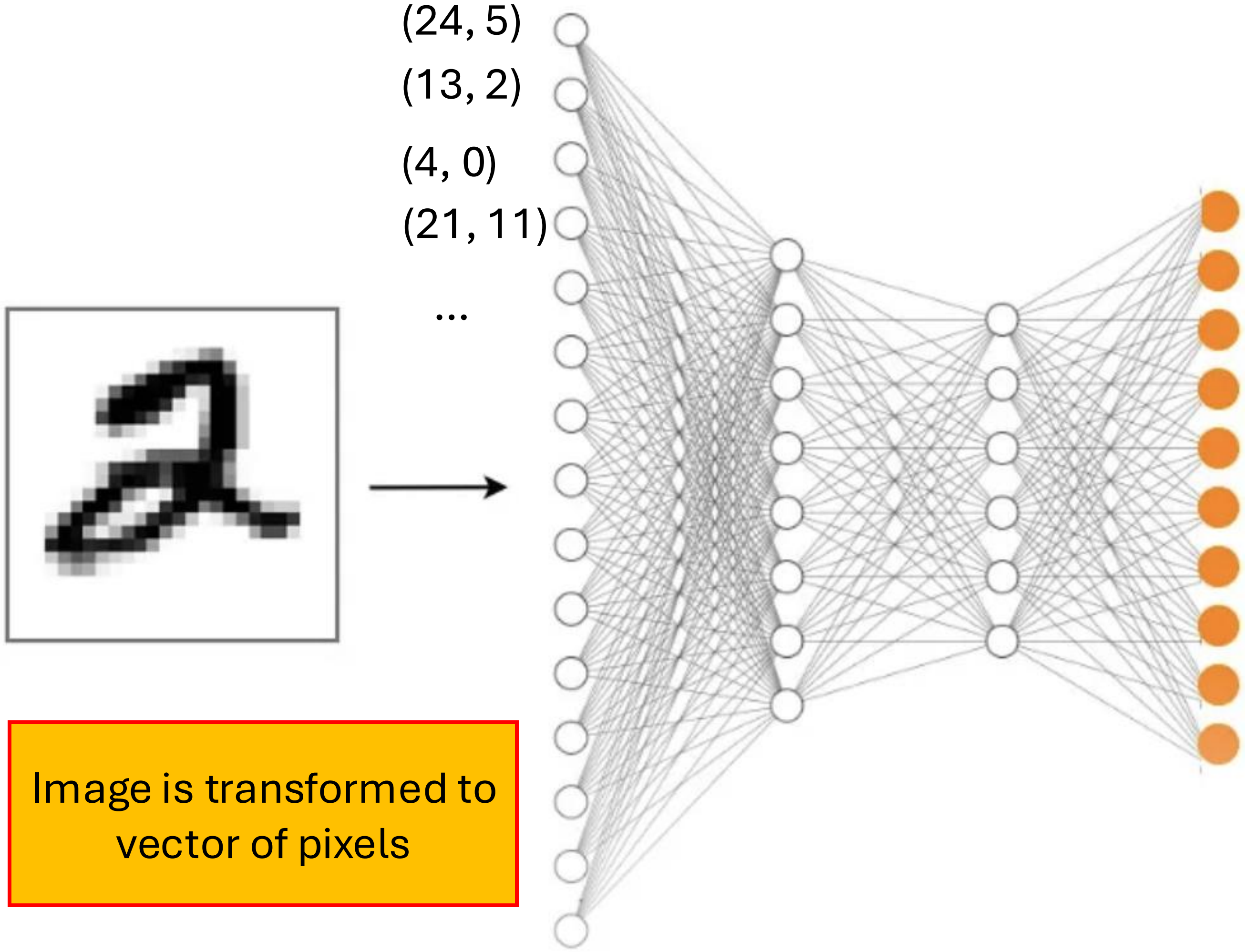
Image is transformed to vector of pixels

Will the training of the neural network differ?

No! MLPs do not use spatial information, it does not matter which order the pixels are fed in so long as it is the same ordering for every input

https://medium.com/towards-data-science/creating-a-multilayer-perceptron-mlp-classifier-model-to-identify-handwritten-digits-9bac1b16fe10

MLPs and Spatial Reasoning



(24, 5)
(13, 2)
(4, 0)
(21, 11)
…

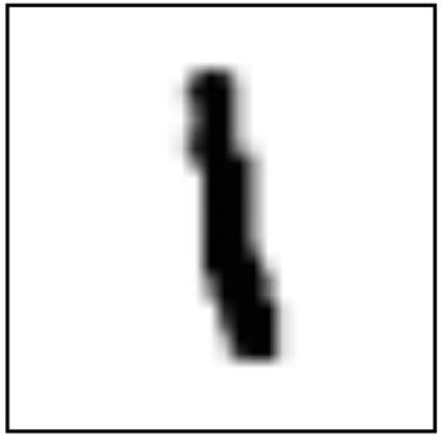
Image is transformed to vector of pixels

Isn’t this actually a hard problem that we are trying to learn?

https://medium.com/towards-data-science/creating-a-multilayer-perceptron-mlp-classifier-model-to-identify-handwritten-digits-9bac1b16fe10

Limitations of Full Connections for MNIST

Suppose we've got a well-trained MNIST classifier...

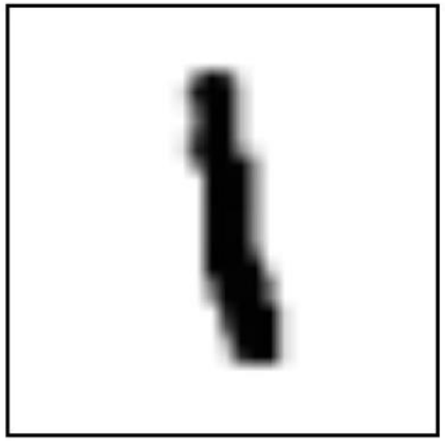


$$\simeq \begin{bmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&.6&.8&0&0&0&0&0&0\\
0&0&0&0&0&0&.7&1&0&0&0&0&0&0\\
0&0&0&0&0&0&.7&1&0&0&0&0&0&0\\
0&0&0&0&0&0&.5&1&.4&0&0&0&0&0\\
0&0&0&0&0&0&0&1&.4&0&0&0&0&0\\
0&0&0&0&0&0&0&1&.4&0&0&0&0&0\\
0&0&0&0&0&0&0&1&.7&0&0&0&0&0\\
0&0&0&0&0&0&0&1&1&0&0&0&0&0\\
0&0&0&0&0&0&0&.9&1&.1&0&0&0&0\\
0&0&0&0&0&0&0&.3&1&.1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{bmatrix}$$

#1 encoded as □

Limitations of Full Connections for MNIST

Suppose we’ve got a well-trained MNIST classifier...



#1 encoded as □

$$\approx \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \text{[illegible]} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .6 & .8 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .7 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .7 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .5 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .7 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \text{[illegible]} & 1 & .1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & .3 & 1 & .1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

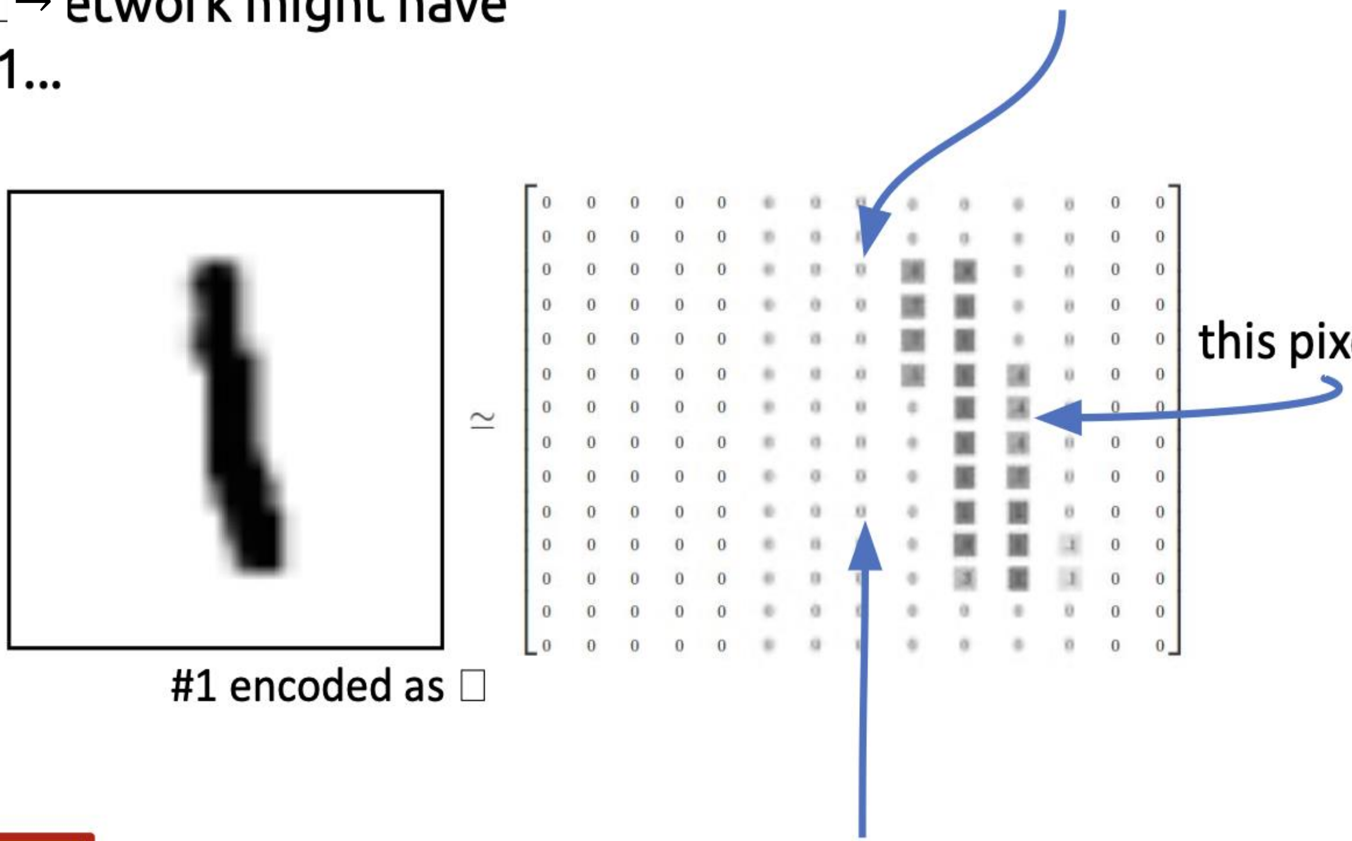
this pixel gets weight 0.6

this pixel gets weight 0.1

this pixel gets weight 0.9

Limitations of Full Connections for MNIST

If we shift the digit to the right, then a different set of weights becomes relevant □→ etwork might have trouble classifying this as a 1…



this pixel gets weight 0.6

this pixel gets weight 0.1

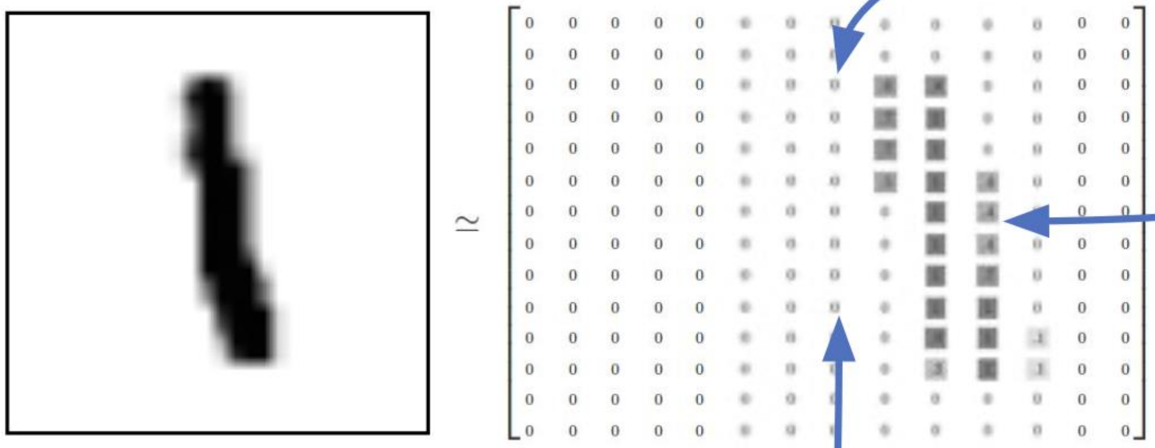
#1 encoded as □

this pixel gets weight 0.9

Can you tell this is a 1?

This would ***not*** be a problem for the human visual system

Our eyes don't look at absolute intensity values…



#1 encoded as

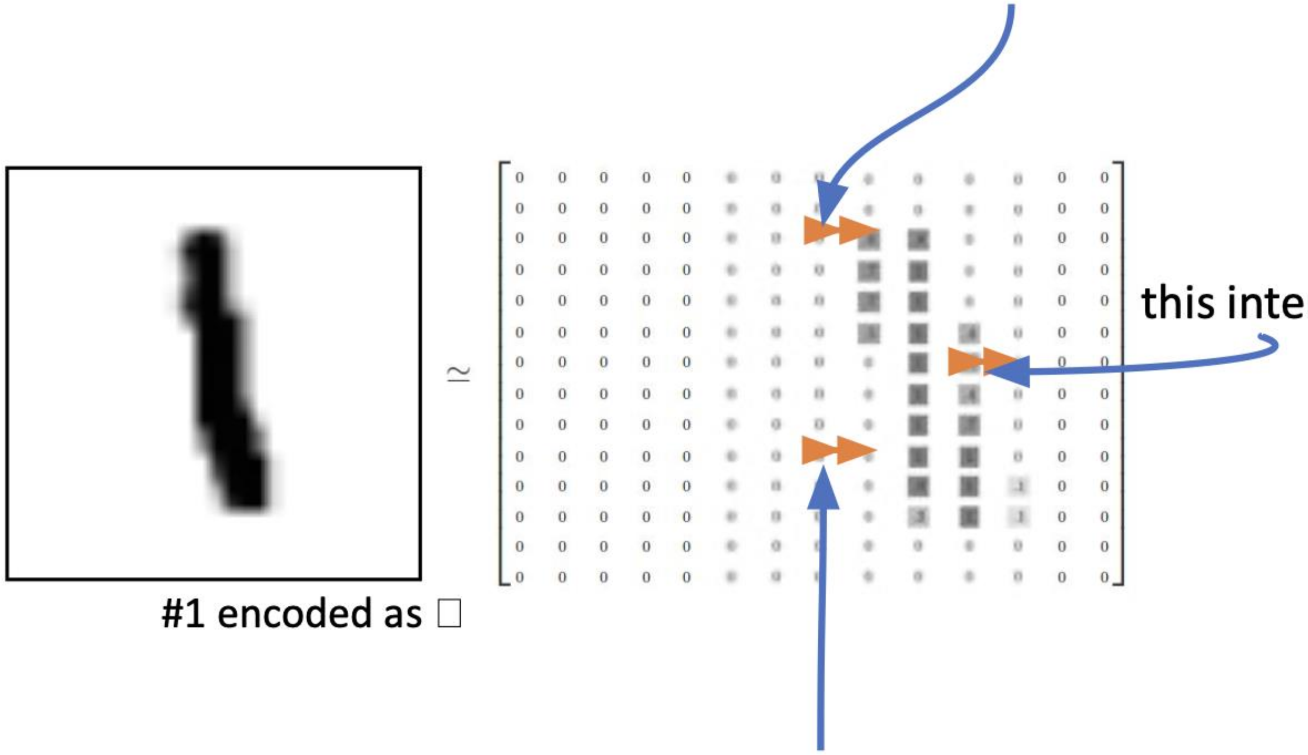
this pixel has a low intensity

this pixel has a high intensity

this pixel has a low intensity

This would *not* be a problem for the human visual system

…but rather ***local differences*** in intensities



this intensity difference is large

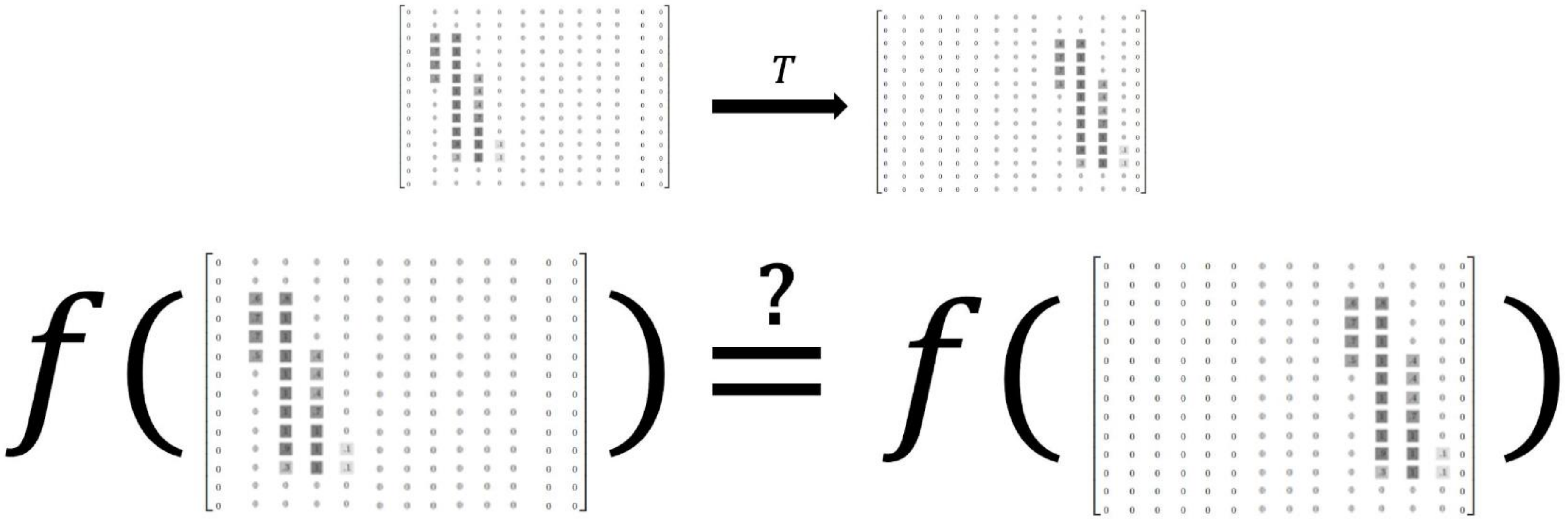
this intensity difference is large

#1 encoded as □

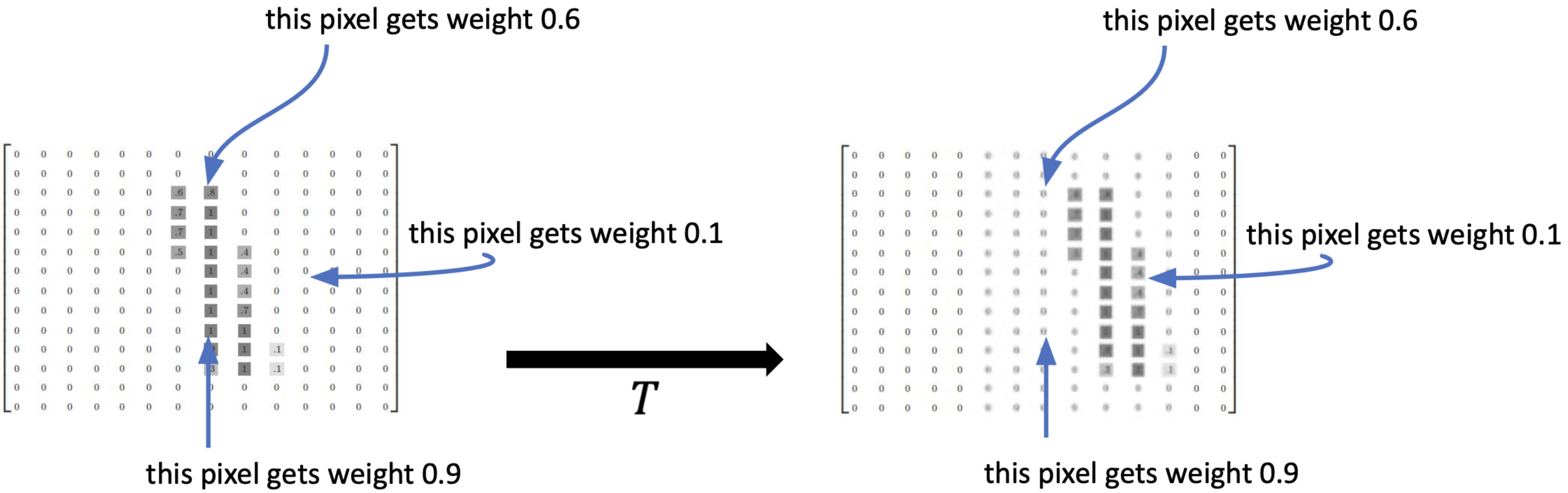
this intensity difference is zero

Translational Invariance

- To make a neural net f robust in this same way, it should ideally satisfy ***translational invariance***: $f\big(T(x)\big) = f(x)$, where
  - x is the input image
  - T is a translation (i.e. a horizonal and/or vertical shift)



Fully Connected Nets are ***not*** Translationally Invariant



this pixel gets weight 0.6

this pixel gets weight 0.1

this pixel gets weight 0.9

$$\xrightarrow{T}$$

this pixel gets weight 0.6

this pixel gets weight 0.1

this pixel gets weight 0.9

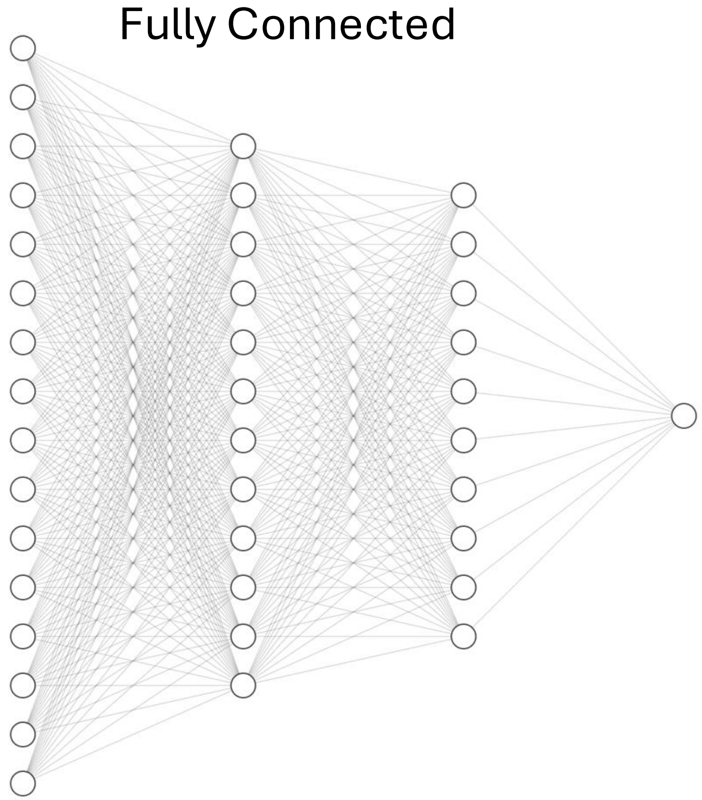
Sum of these three: $0.6 \cdot 0.8 + 0.1 \cdot 0 + 0.9 \cdot 1 = 1.38$

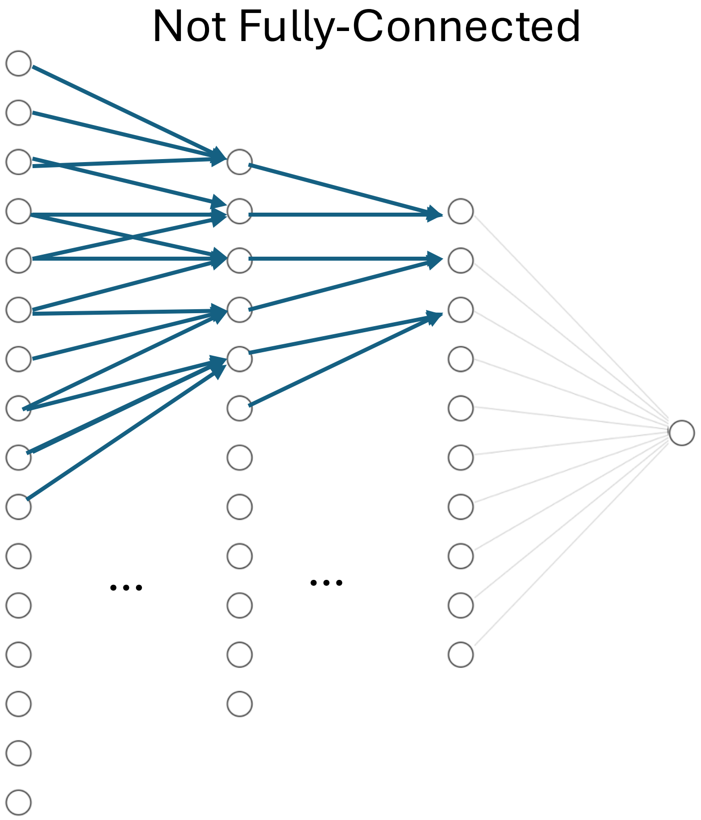
Sum of these three: $0.6 \cdot 0 + 0.1 \cdot 0.4 + 0.9 \cdot 0 = 0.4$

MLPs and Spatial Reasoning

How can we change a fully-connected network to account for spatial information?

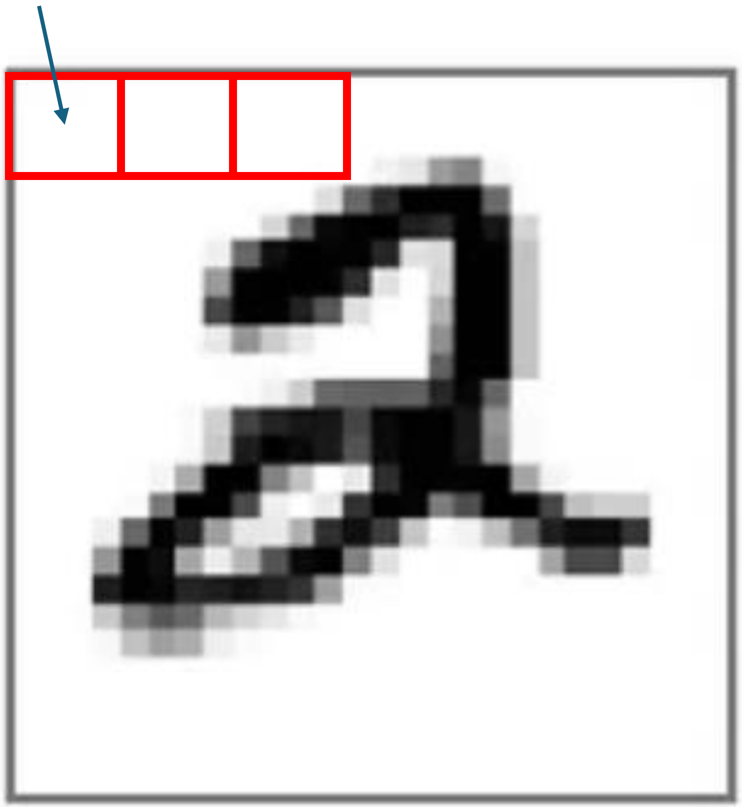
MLPs (also called fully-connected networks) have weights from every pixel to every neuron





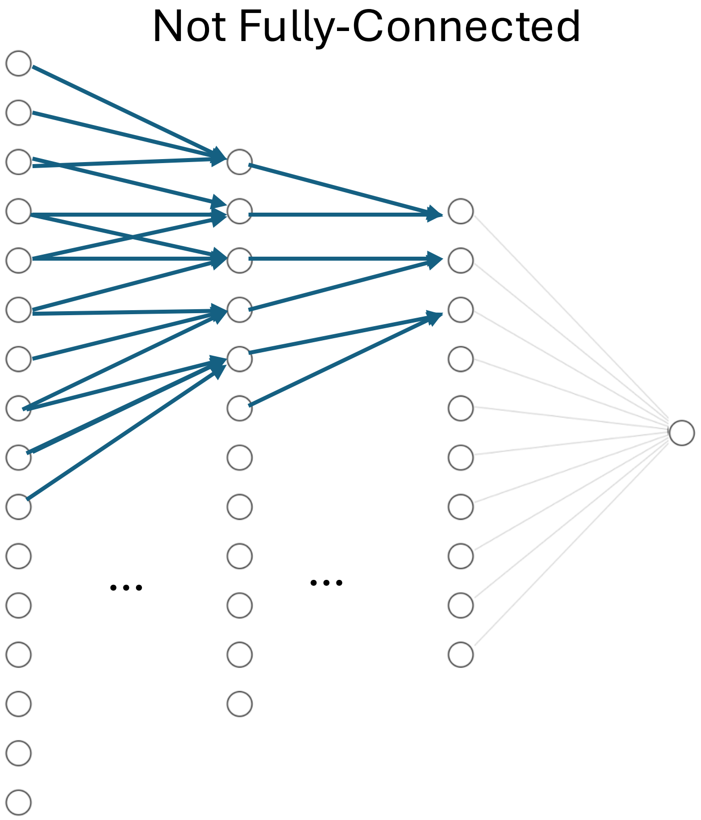
MLPs and Spatial Reasoning

Patches: Pixels close to each other



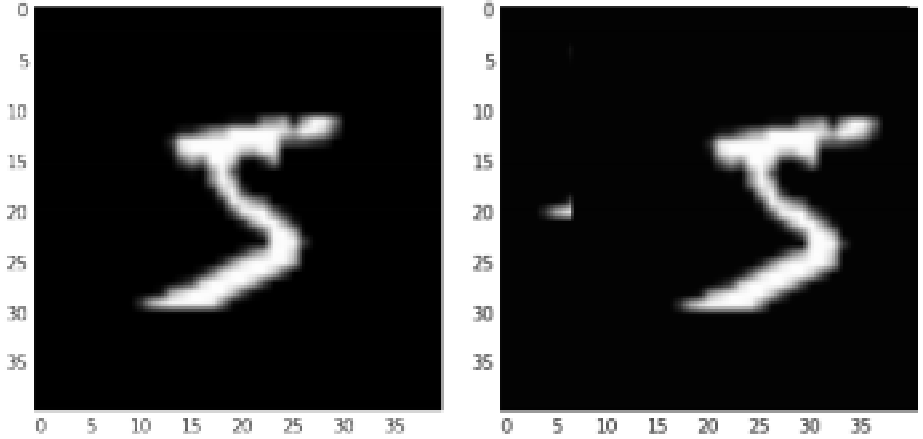
Advantages of Not Fully Connected Layers

- Fewer weights → Faster?
- The outputs of neurons are "features" for local "patches"
- Incorporates spatial information (pixels that are close together matter)



Disadvantages of Not Fully Connected Layers

- What happens if the image is Translated?
- The patches on the right side were never trained with 5's in that side.



Even though we include spatial ***information***, we still don't have spatial ***reasoning***. *(Can't recognize a shifted 5 is still a 5)*

What if we used the same weights for each patch? (Weight Sharing)

The Main Building Block: Convolution

Convolution is an operation that takes two inputs:

(1) An image (2D – B/W)

(2) A filter (also called a kernel)

1	1	1
0	0	0
-1	-1	-1

2D array of numbers; could be any values

What Convolution Does (Visually)

image

2	0	1	3
7	1	1	0
0	2	5	0
0	5	1	4

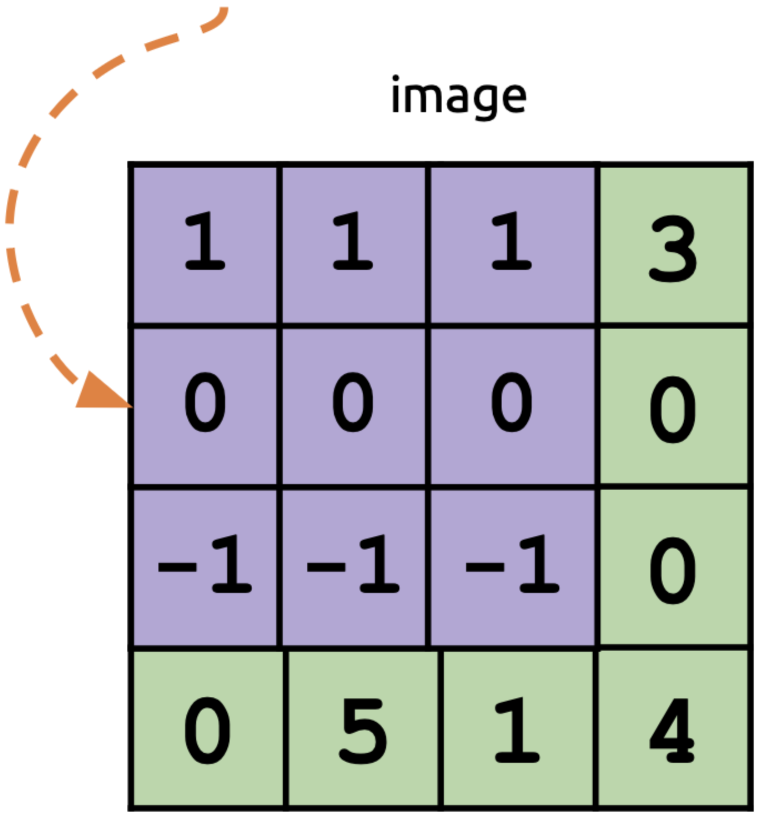
$\otimes$

filter/kernel

1	1	1
0	0	0
−1	−1	−1

(We use this symbol for convolution)
(The verb form is "convolve")

What Convolution Does (Visually)

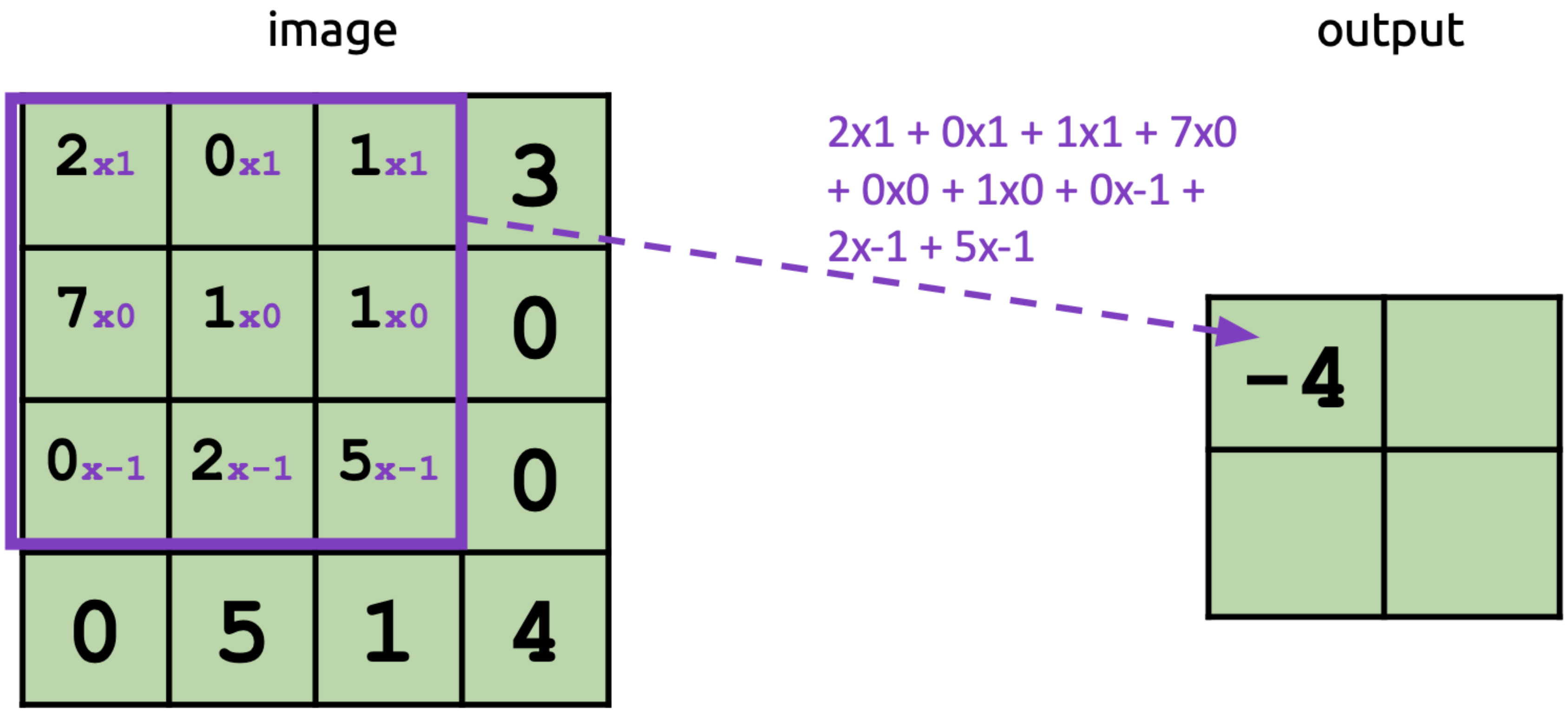


Overlay the filter on the image

1	1	1	3
0	0	0	0
-1	-1	-1	0
0	5	1	4

What Convolution Does (Visually)

Sum up multiplied values to produce output value



image

2 x1	0 x1	1 x1	3
7 x0	1 x0	1 x0	0
0 x-1	2 x-1	5 x-1	0
0	5	1	4

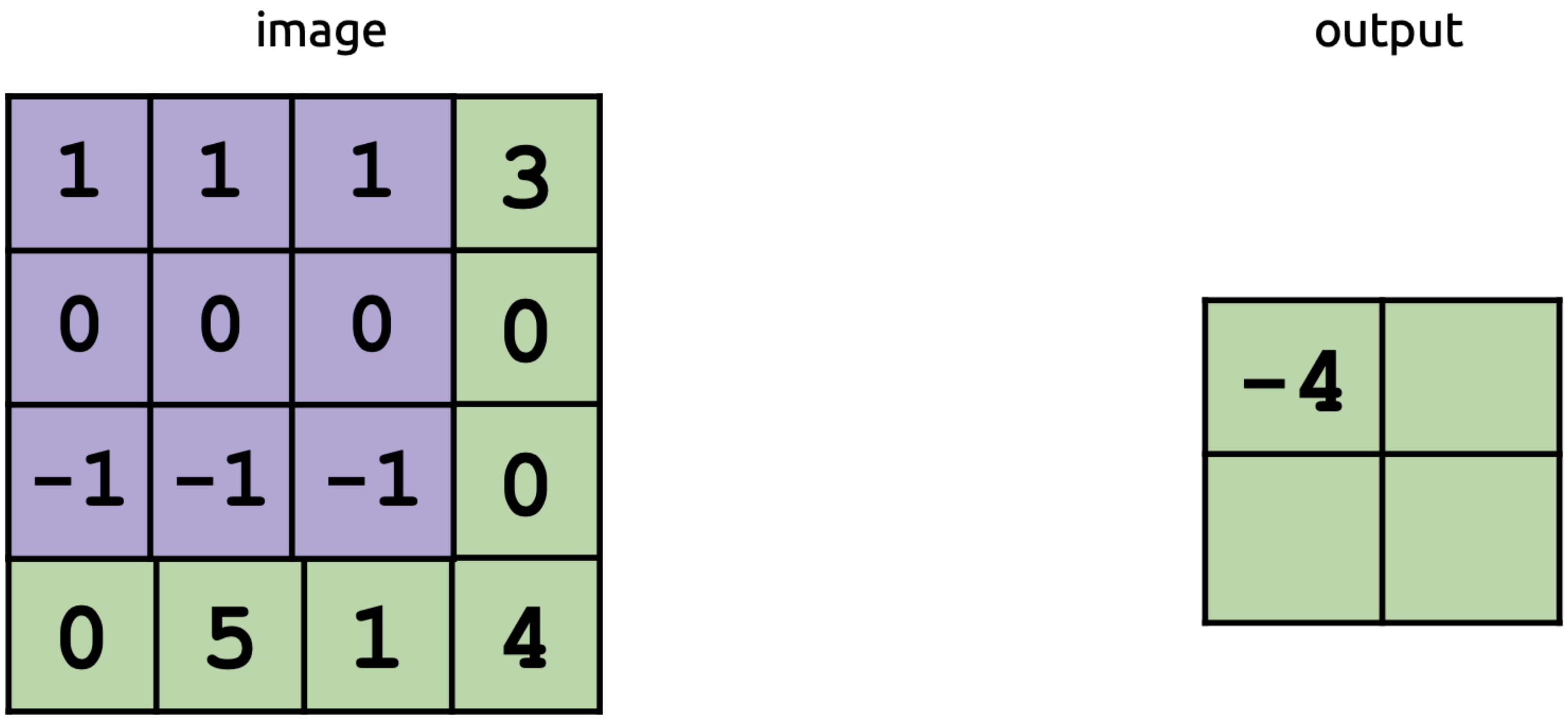
2x1 + 0x1 + 1x1 + 7x0 + 0x0 + 1x0 + 0x-1 + 2x-1 + 5x-1

output

-4	

What Convolution Does (Visually)

Move the filter over by one pixel



image

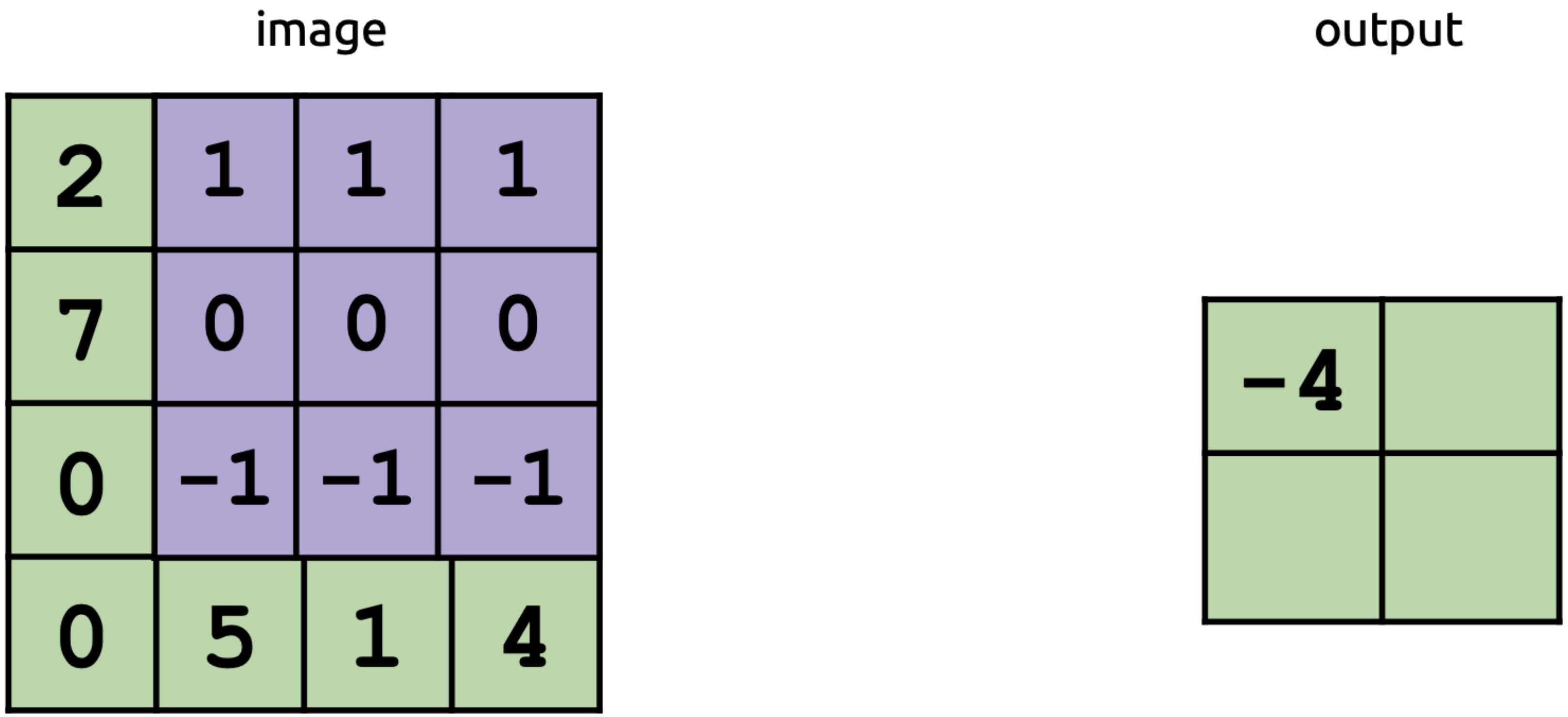
1	1	1	3
0	0	0	0
-1	-1	-1	0
0	5	1	4

output

-4	

What Convolution Does (Visually)

Move the filter over by one pixel



image

2	1	1	1
7	0	0	0
0	-1	-1	-1
0	5	1	4

output

-4	

What Convolution Does (Visually)

Repeat (multiply, sum up)

image

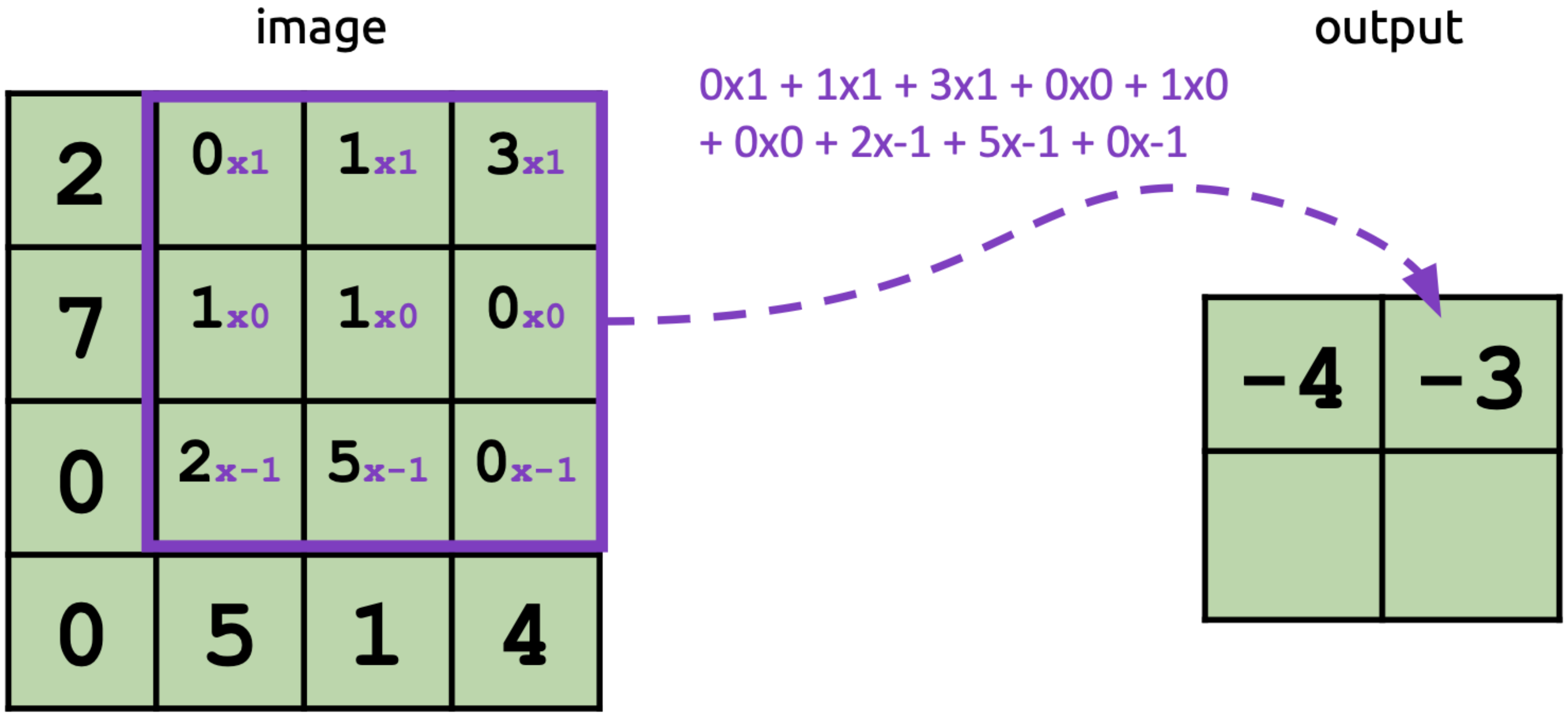
2	0_{x1}	1_{x1}	3_{x1}
7	1_{x0}	1_{x0}	0_{x0}
0	2_{x-1}	5_{x-1}	0_{x-1}
0	5	1	4

output

-4	

What Convolution Does (Visually)

Repeat (multiply, sum up)



image

2	0_{x1}	1_{x1}	3_{x1}
7	1_{x0}	1_{x0}	0_{x0}
0	2_{x-1}	5_{x-1}	0_{x-1}
0	5	1	4

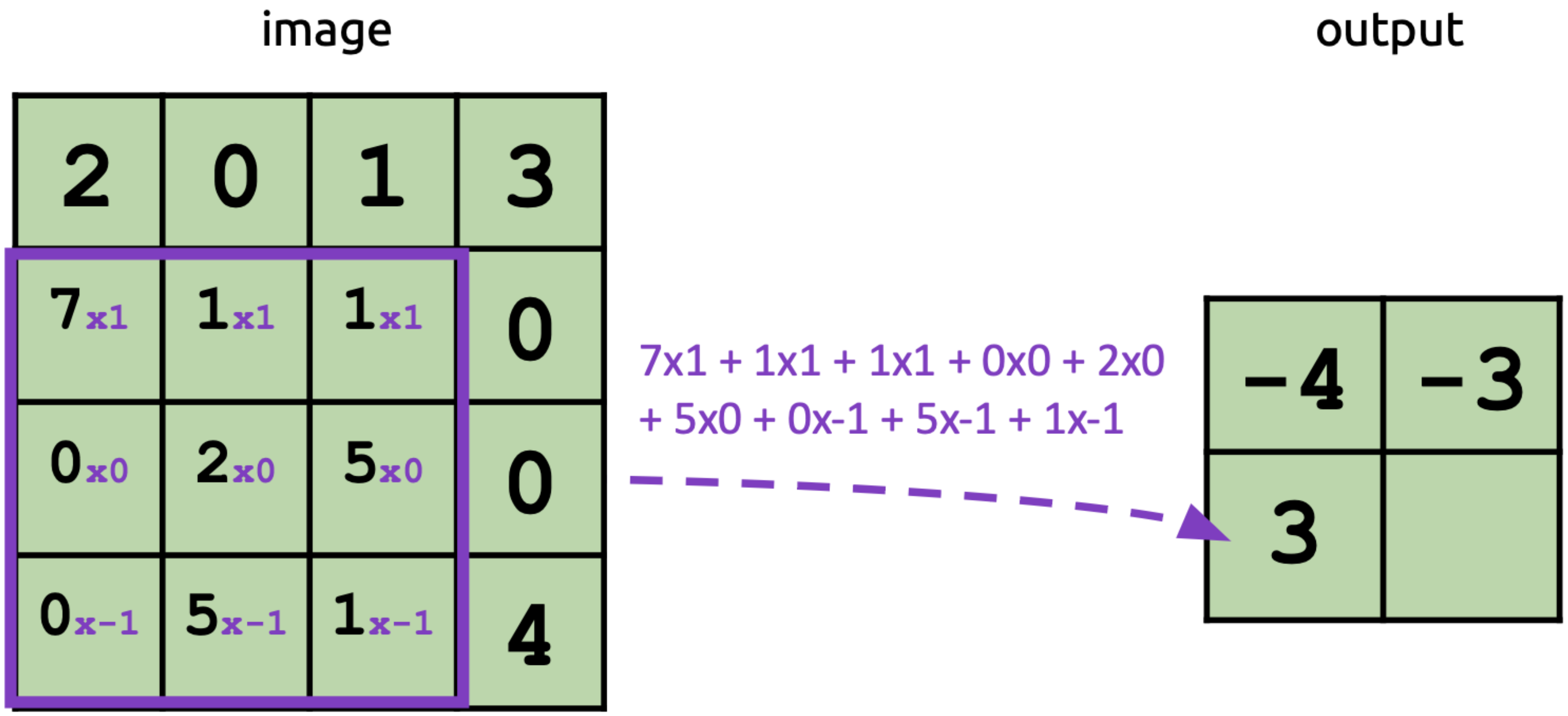
0x1 + 1x1 + 3x1 + 0x0 + 1x0
+ 0x0 + 2x-1 + 5x-1 + 0x-1

output

-4	-3

What Convolution Does (Visually)

Repeat…



image

2	0	1	3
7_{x1}	1_{x1}	1_{x1}	0
0_{x0}	2_{x0}	5_{x0}	0
0_{x-1}	5_{x-1}	1_{x-1}	4

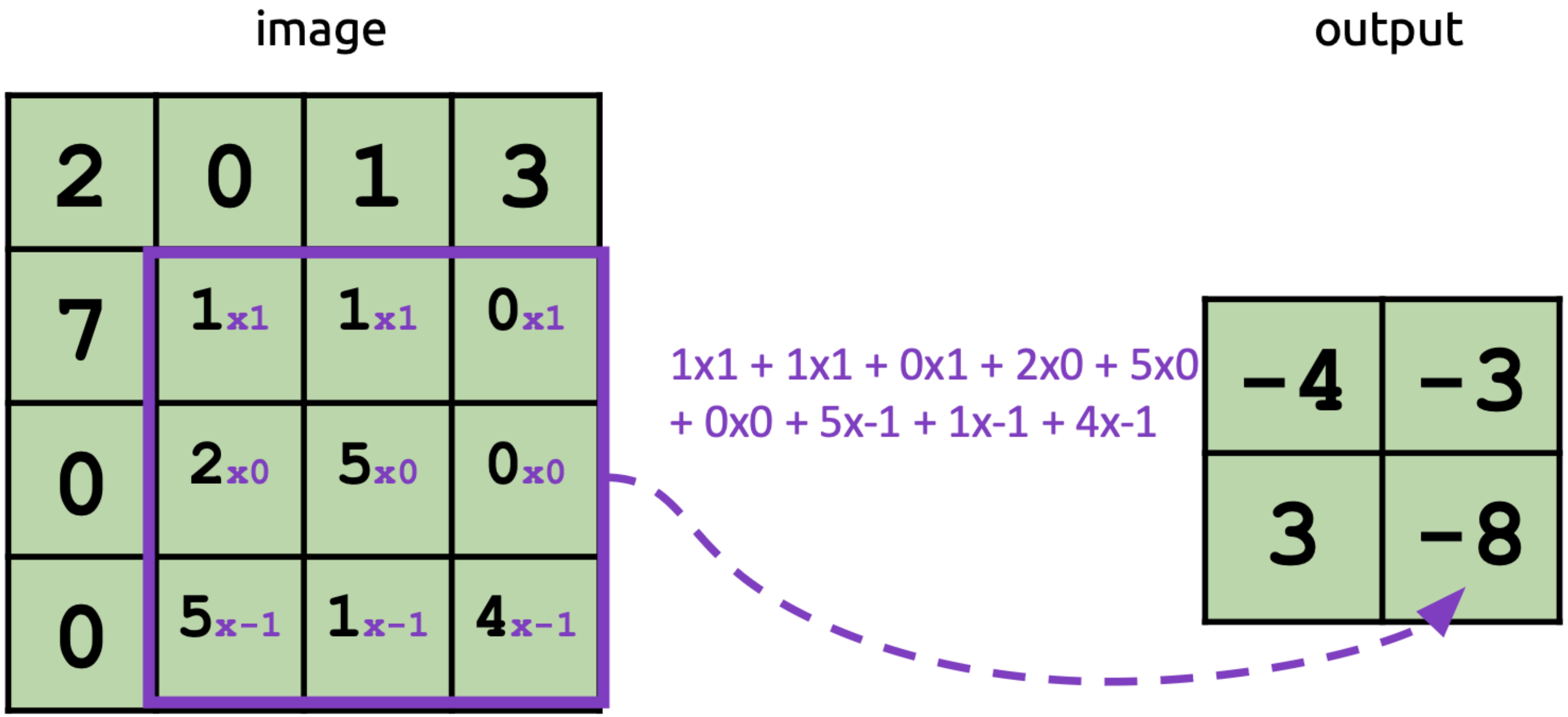
7x1 + 1x1 + 1x1 + 0x0 + 2x0
+ 5x0 + 0x-1 + 5x-1 + 1x-1

output

-4	-3
3	

What Convolution Does (Visually)

Repeat...



image

2	0	1	3
7	1x1	1x1	0x1
0	2x0	5x0	0x0
0	5x-1	1x-1	4x-1

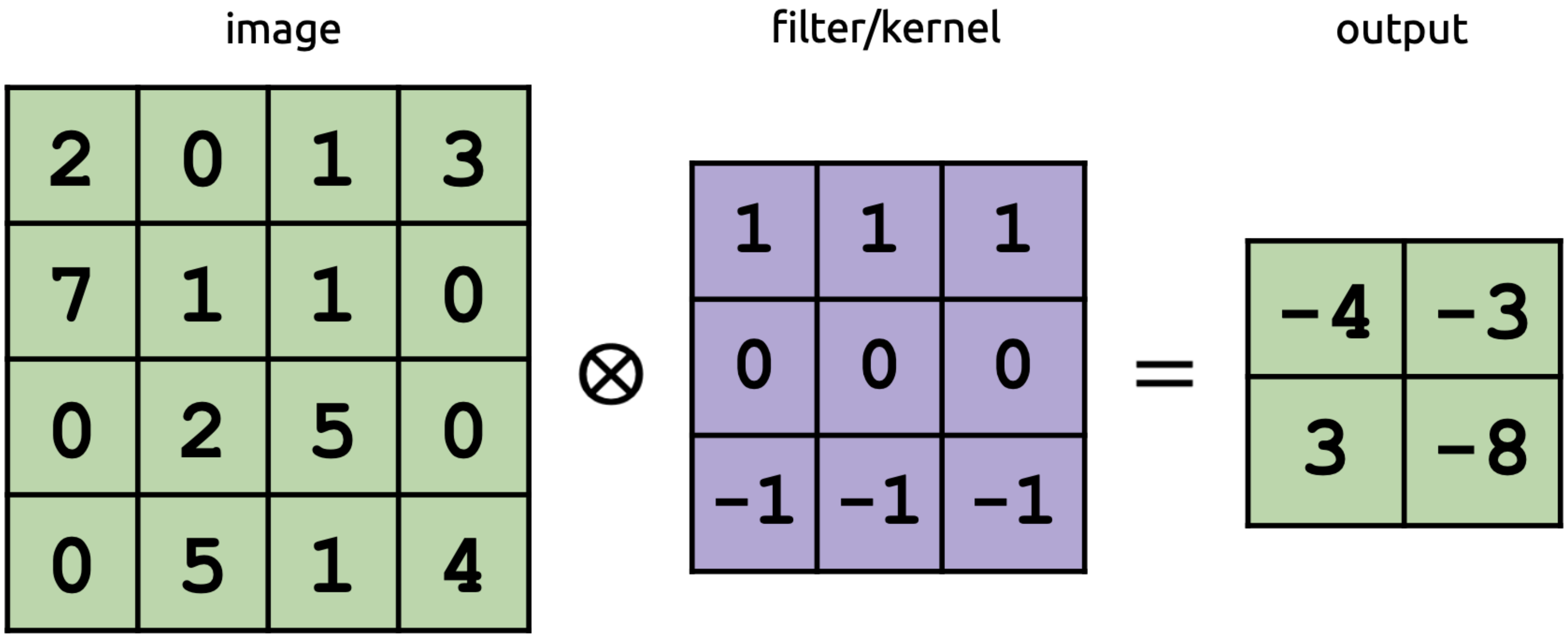
1x1 + 1x1 + 0x1 + 2x0 + 5x0 + 0x0 + 5x-1 + 1x-1 + 4x-1

output

-4	-3
3	-8

What Convolution Does (Visually)

In summary:



image

filter/kernel

output

$$\begin{bmatrix} 2 & 0 & 1 & 3 \\ 7 & 1 & 1 & 0 \\ 0 & 2 & 5 & 0 \\ 0 & 5 & 1 & 4 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} = \begin{bmatrix} -4 & -3 \\ 3 & -8 \end{bmatrix}$$

Handmade Kernels and Filters

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

Identity kernel

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Edge detection

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

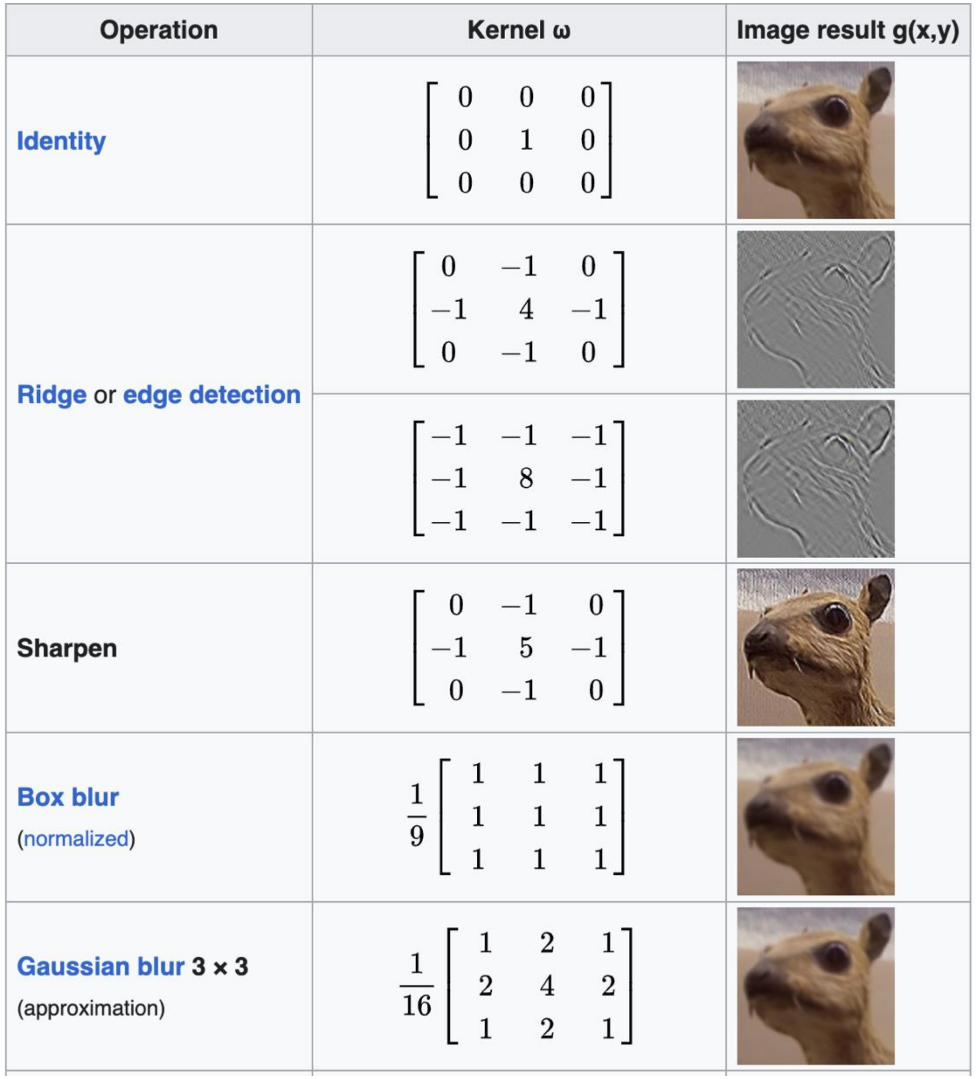
Sharpen kernel

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

Box blur

$$\frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

Gaussian blurr kernel



Operation	Kernel ω	Image result g(x,y)
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Ridge or edge detection	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
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)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

What Comes Next?

Can we learn a filter for our images rather than “hand crafting” one?

Recap

Fully Connected Neural Networks for images lack spatial information

Participation Quiz #3 is up!

We can add spatial reasoning by connecting pixels that are “spatially” close

Convolutions/Filters/Kernels are a technique from image processing that combine close pixels with a linear transformation