Deep Learning
Day 7: Convolutions

Thursday,
9/25/25

Snake River Canyon

Beras Tips

Use built in matrix operations in numpy (@ for matrix multiplication)

If you can't figure out a shape, play around in a REPL (i.e., run "python" from the terminal)

Many functions are "just a formula"

Forward function of linear layer: xW+b

MSE: mean((y-y_pred)**2)

get_input_gradient, returns a list of Tensors, one Tensor for each input. If one of those inputs is not a trainable parameter (i.e., the true labels in the loss function), return an array of 0's.

Others are conceptually harder…

Gradient Tape's gradient method

- Start with target (loss) gradient, backprop gradient to previous layers, push those gradients to grandparent layers, etc.

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!

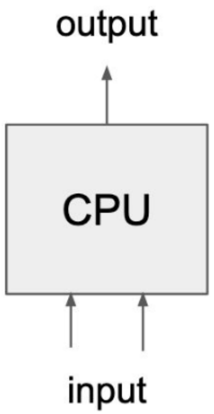
- Matrix multiplication == many **independent** multiply and add operations

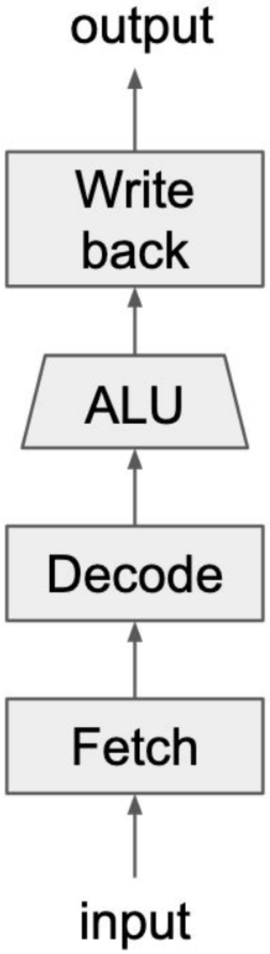
  Easily parallelizable

  GPUs are great for this!

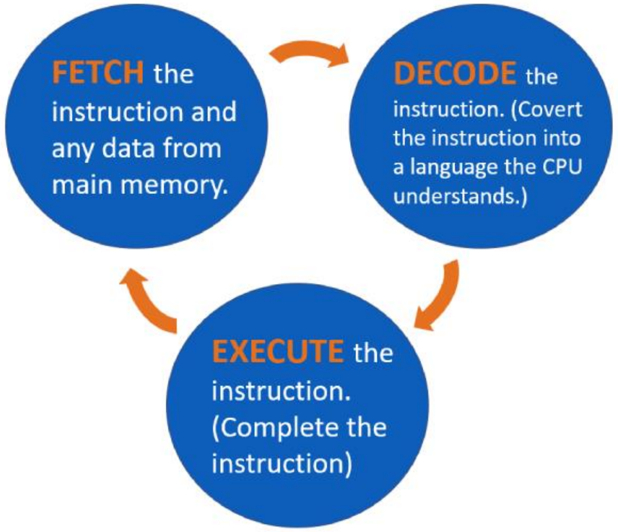
Image courtesy: https://global.aorus.com/blog-detail.php?i=878

CPU v/s GPU



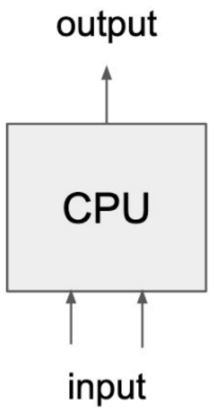


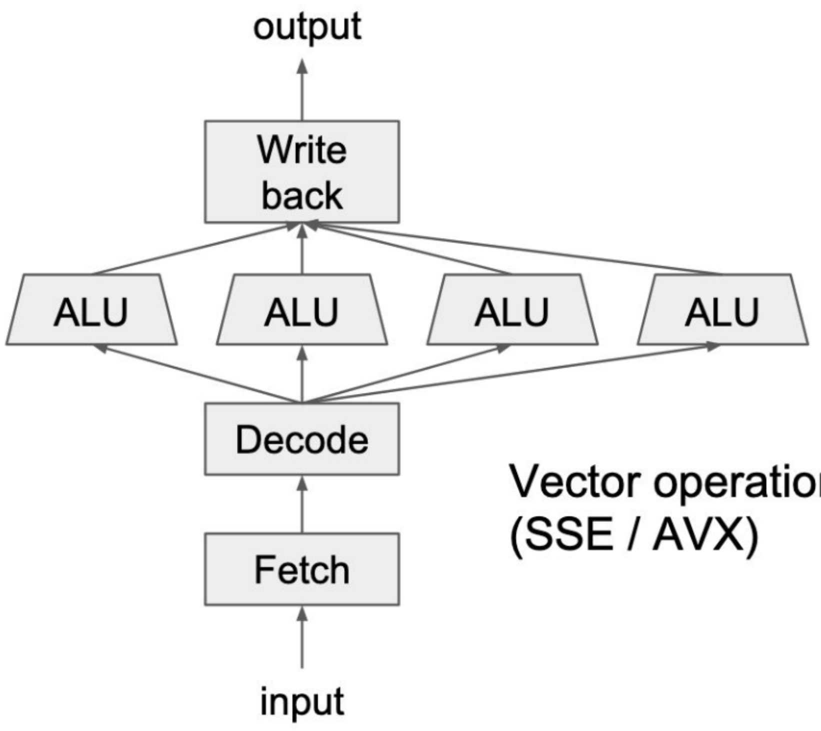
Arithmetic logic unit



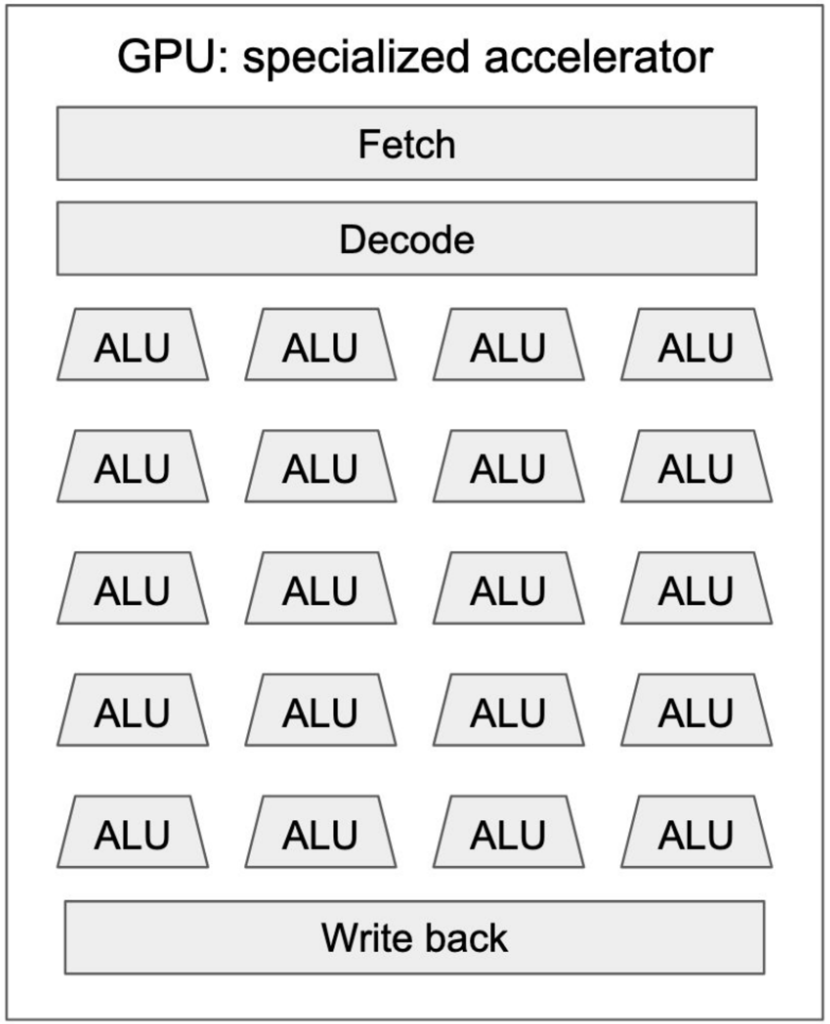
https://dlsys.cs.washington.edu/pdf/lecture5.pdf

CPU v/s GPU





Vector operations (SSE / AVX)

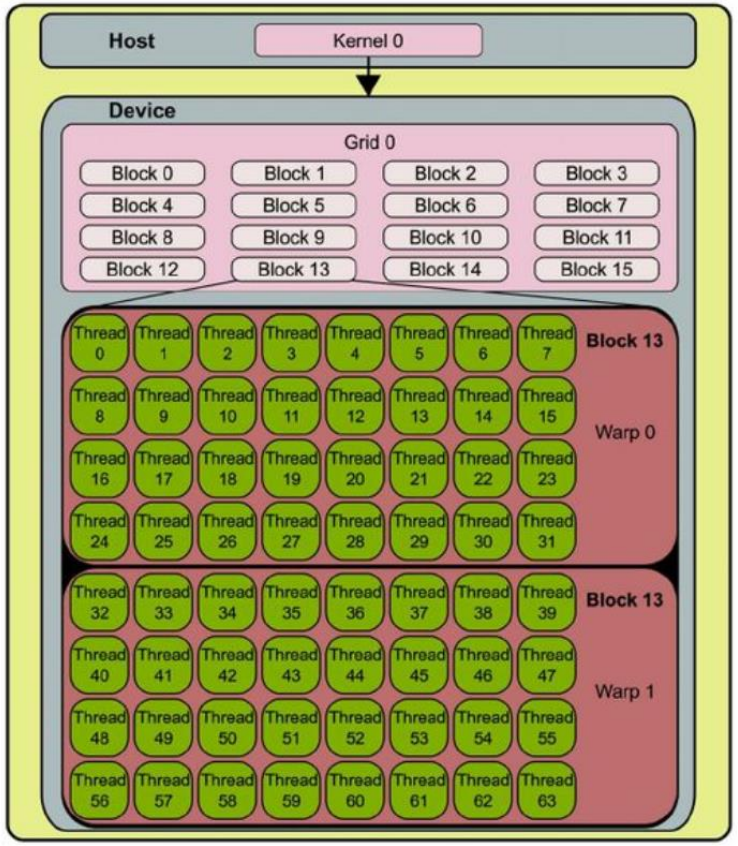


https://dlsys.cs.washington.edu/pdf/lecture5.pdf

GPU-Parallel Acceleration

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

CUDA compute model



- 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***

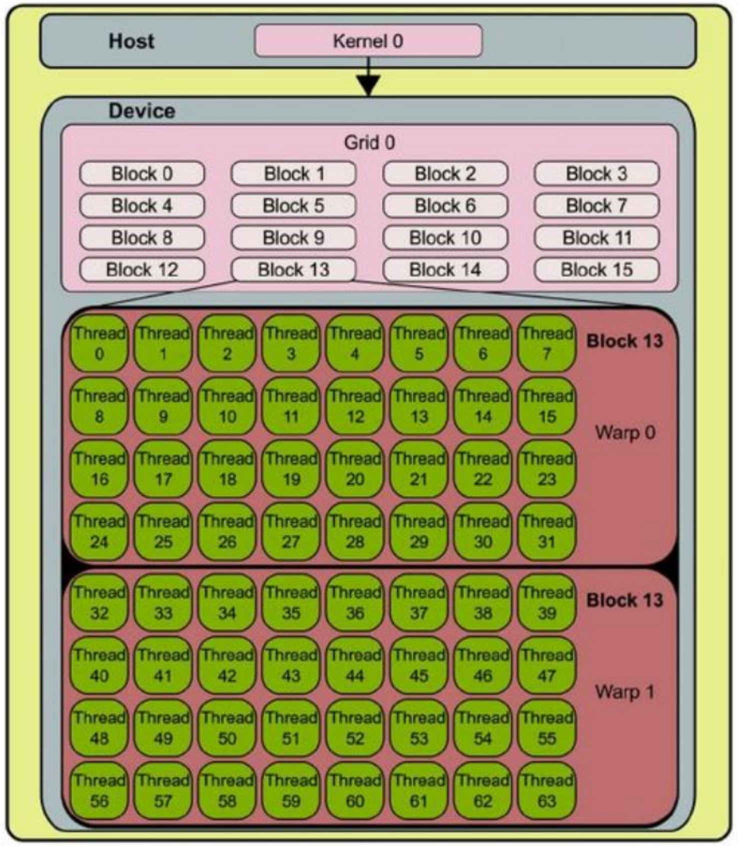
A warp is the basic schedule unit in kernel execution

A warp consists of 32 threads

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

GPU-Parallel Acceleration

CUDA compute model



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

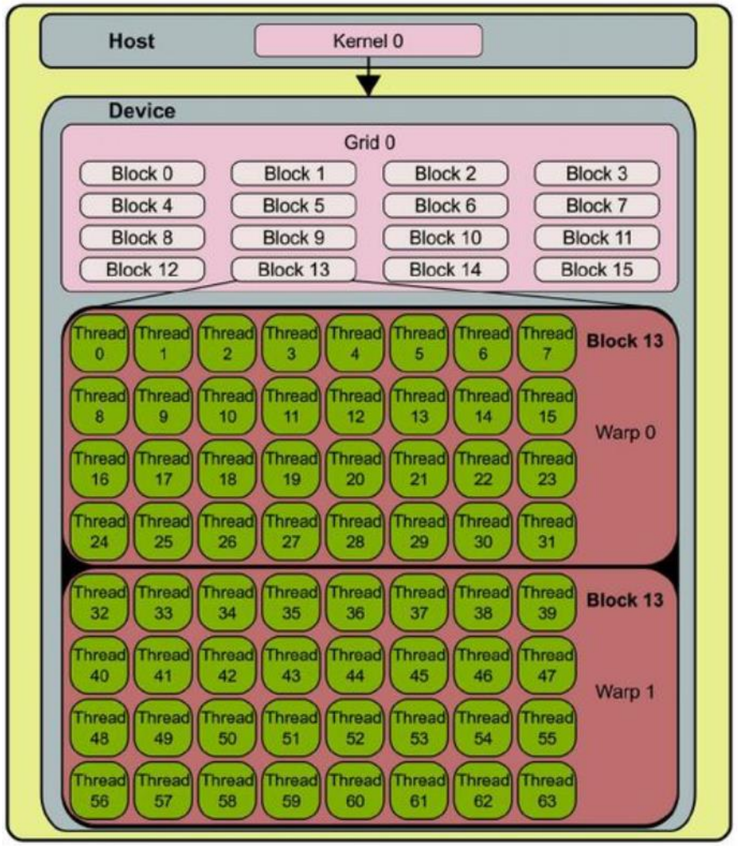
- CUDA compiler figures out how to schedule these units of computation on to the physical hardware

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



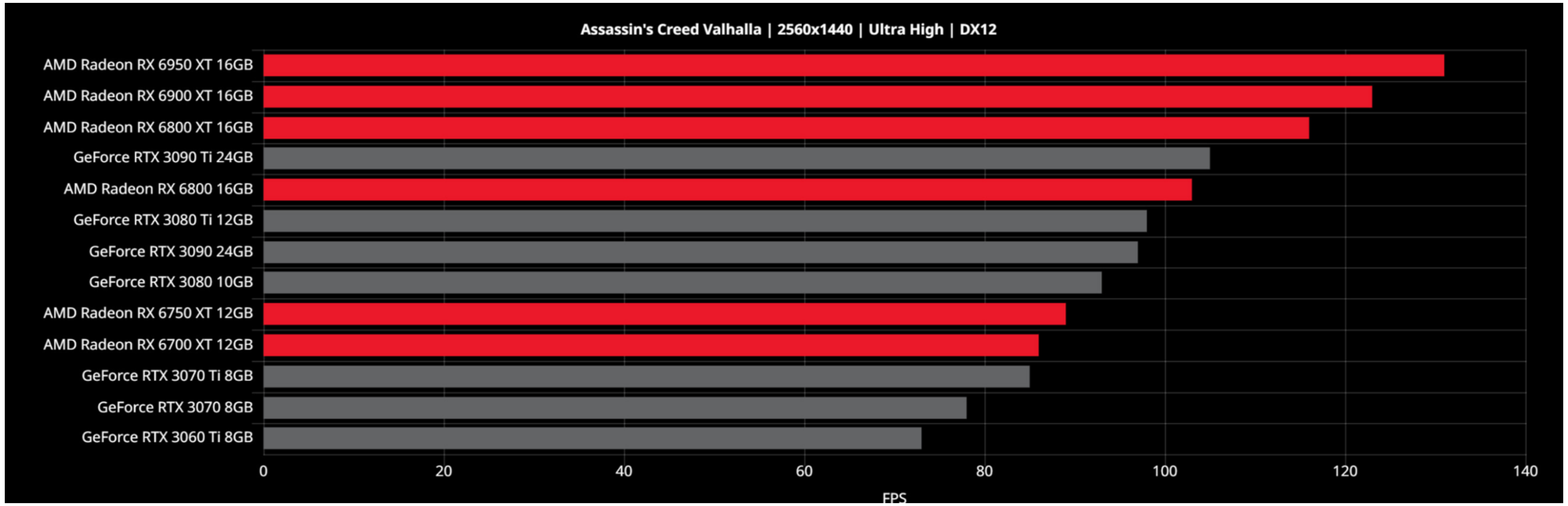
- Upshot: order of magnitude speedups!
- Example: training CNN on CIFAR-10 dataset

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

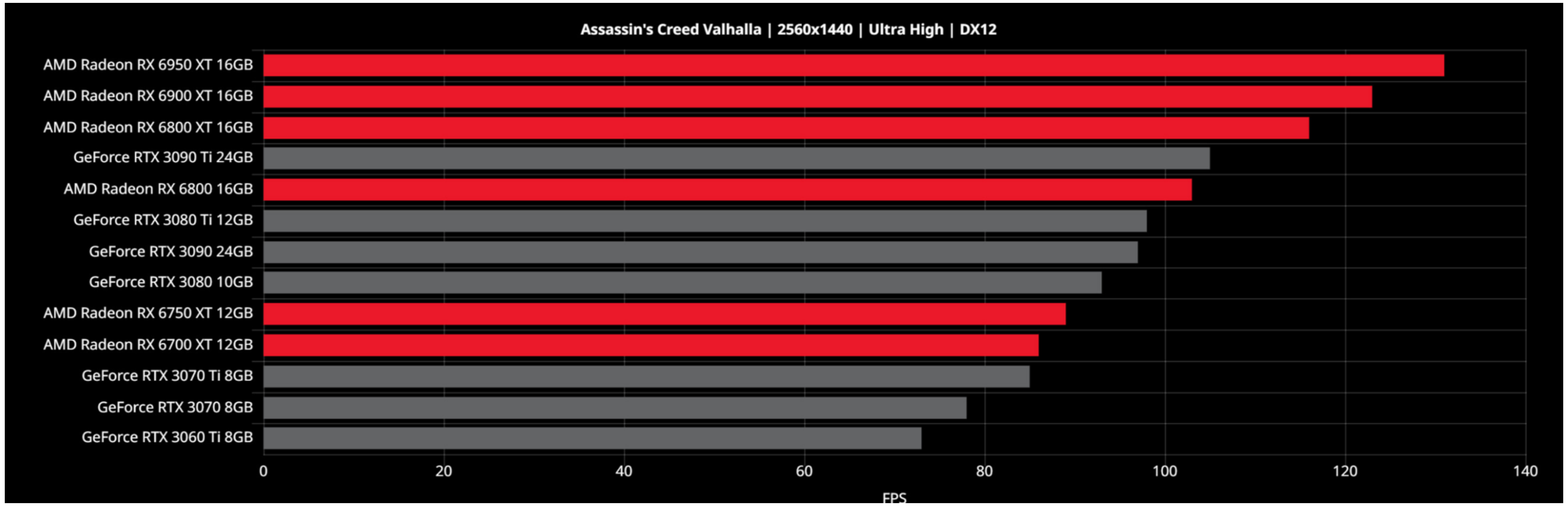
https://www.researchgate.net/publication/236666656_Accelerating_Fibre_Orientation_Estimation_from_Diffusion_Weighted_Magnetic_Resonance_Imaging_Using_GPUs

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



(With a benchmarking tool made by AMD)

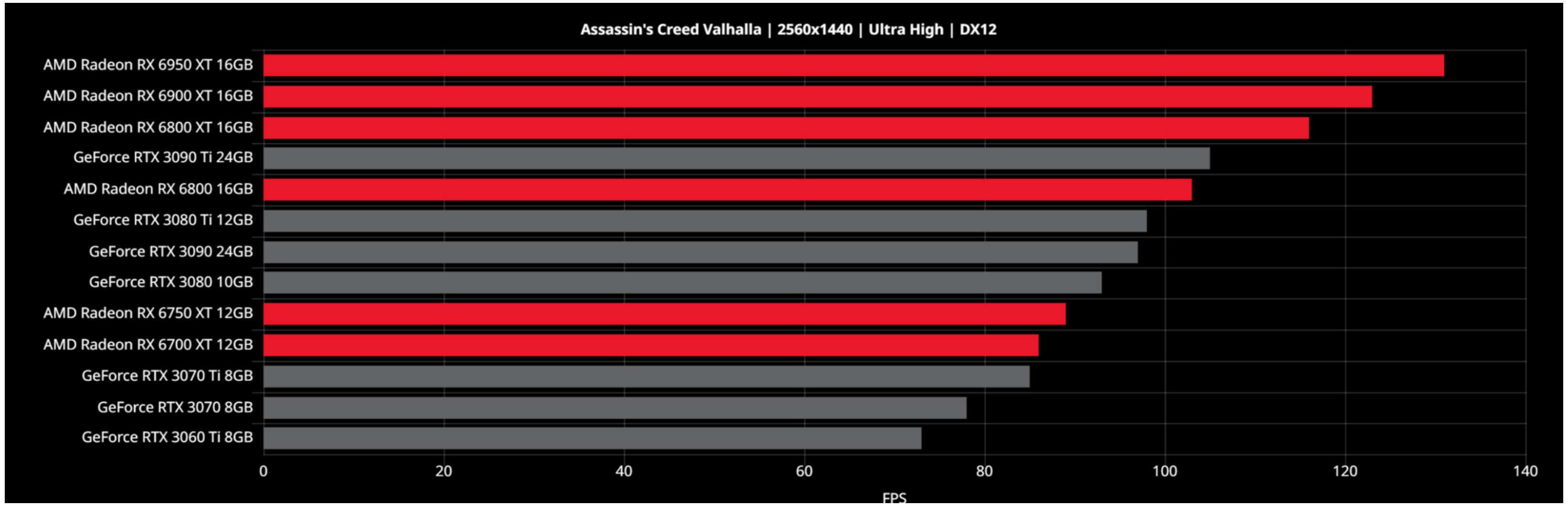
AMD GPUs are competitive for gaming and graphics, why not for AI?



(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

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

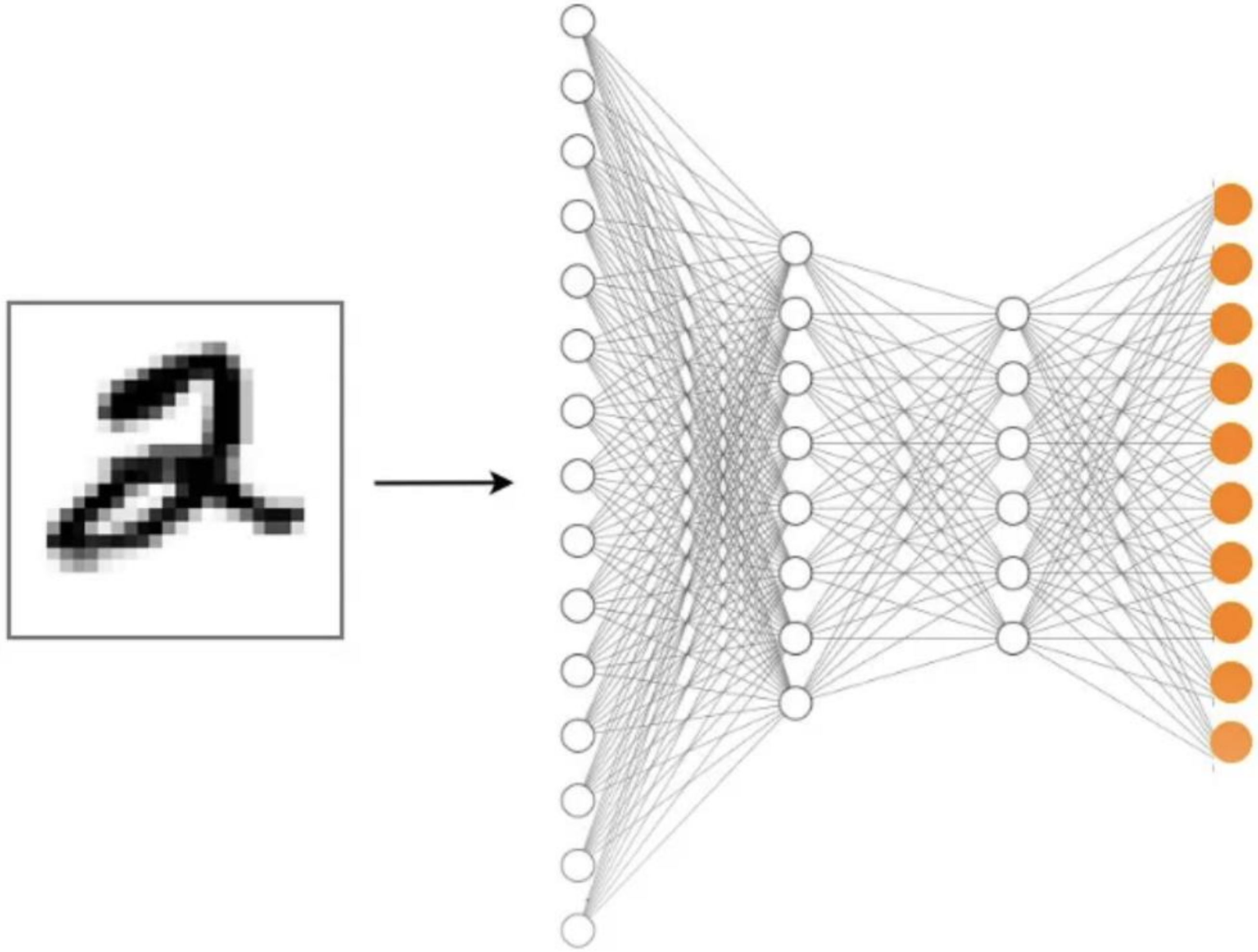
JAMES WANG

MAR 23, 2024

Issues with MLPs

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

MLPs and Spatial Reasoning



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

MLPs and Spatial Reasoning

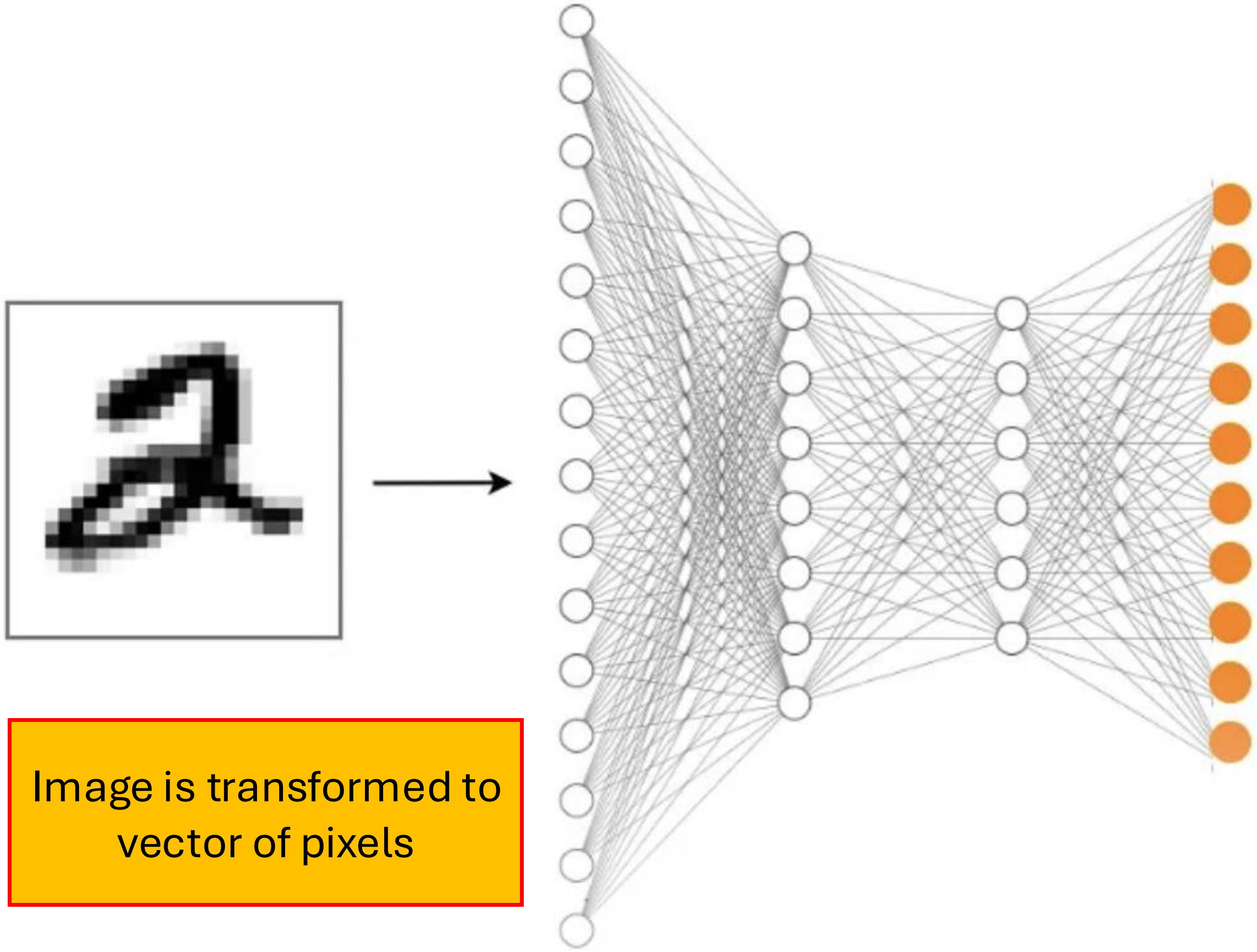


Image is transformed to vector of pixels

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

MLPs and Spatial Reasoning

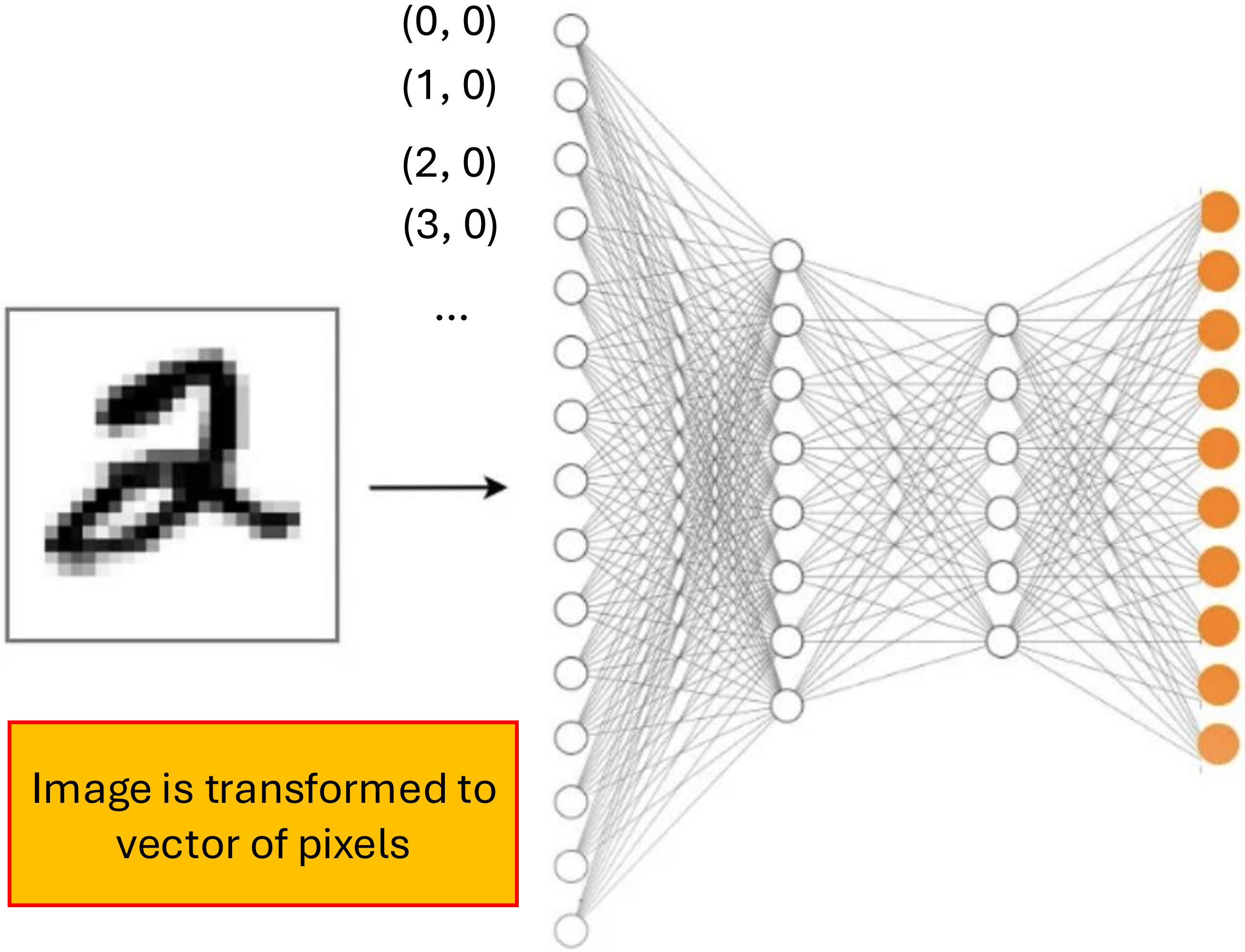


Image is transformed to vector of pixels

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?

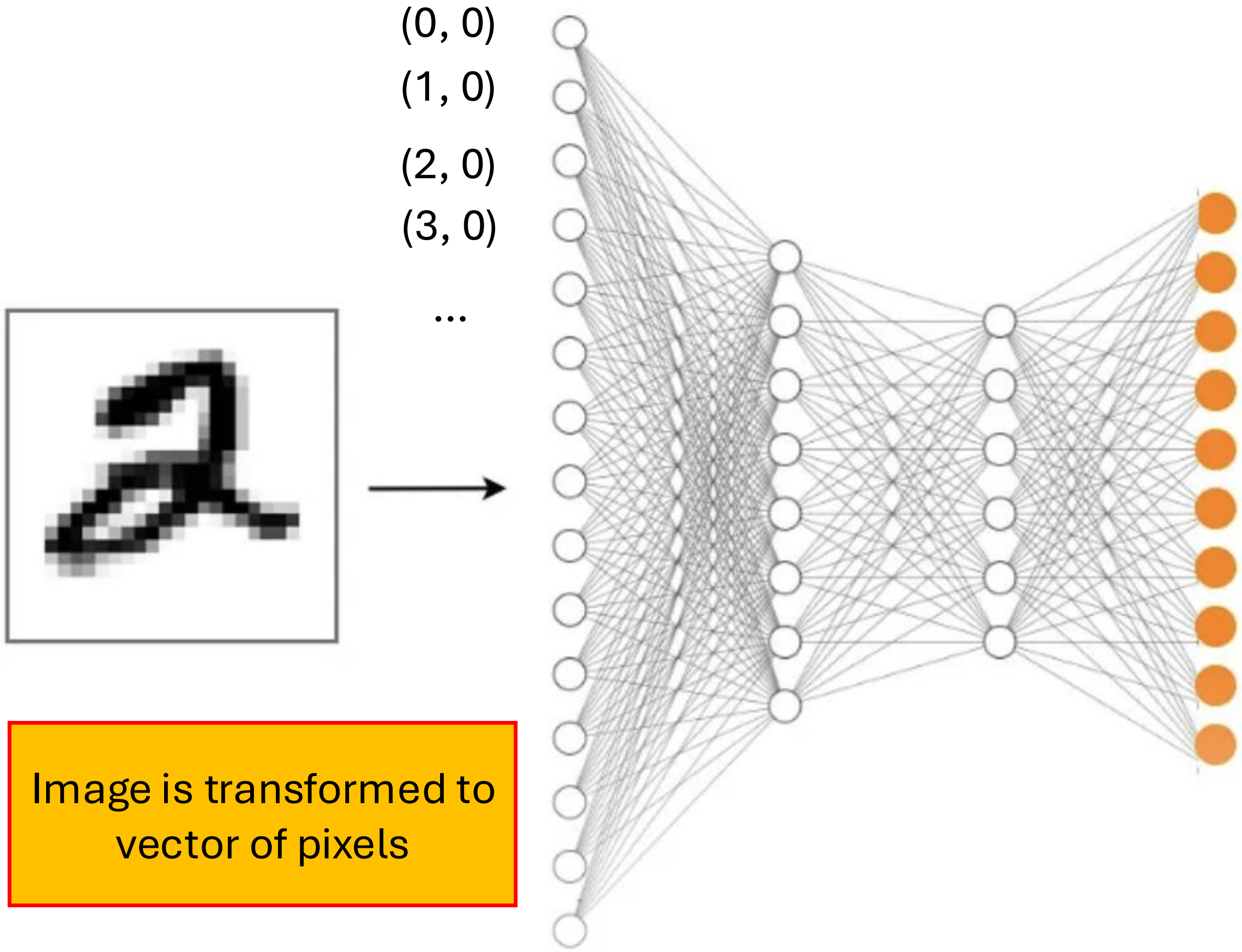


Image is transformed to vector of pixels

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?

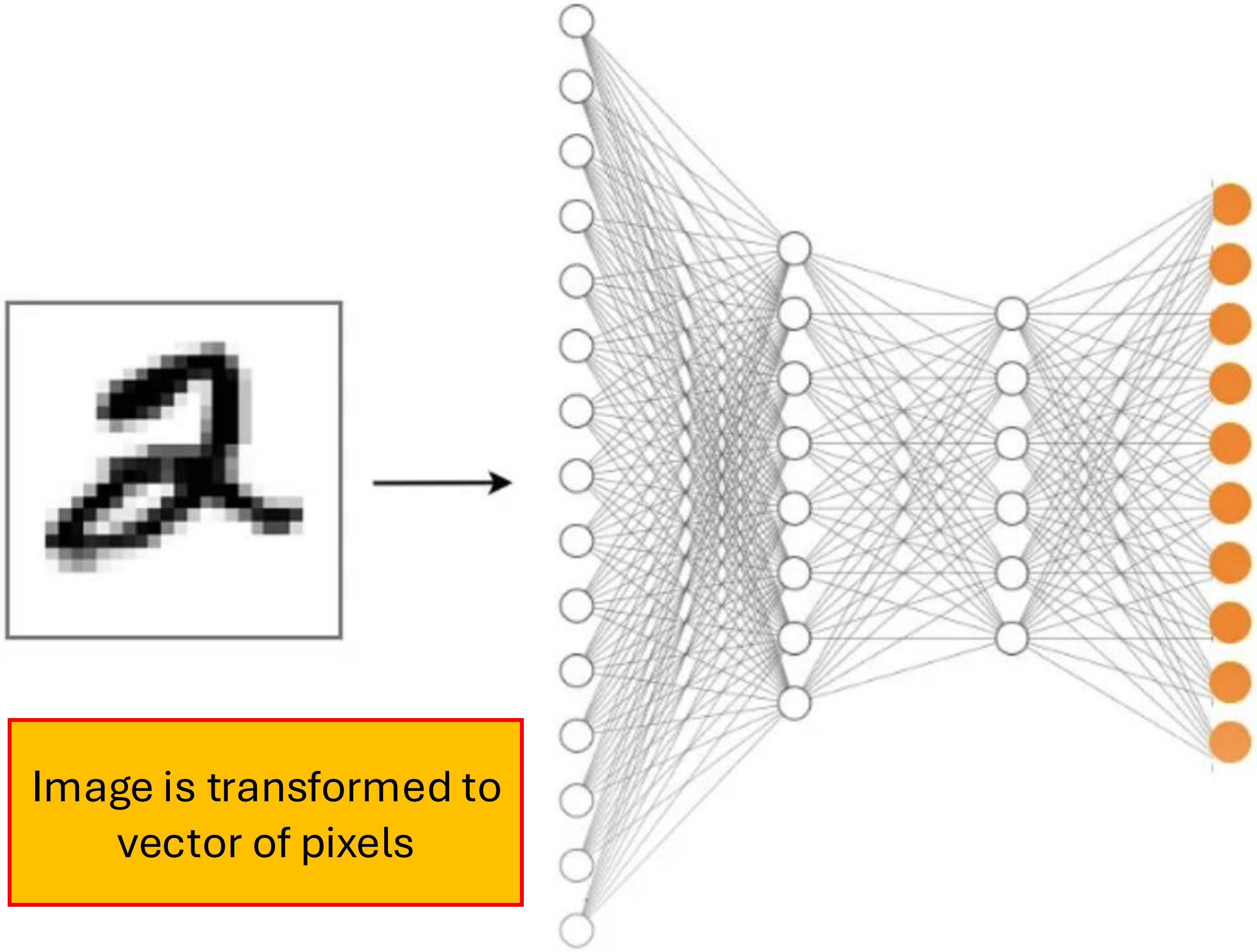
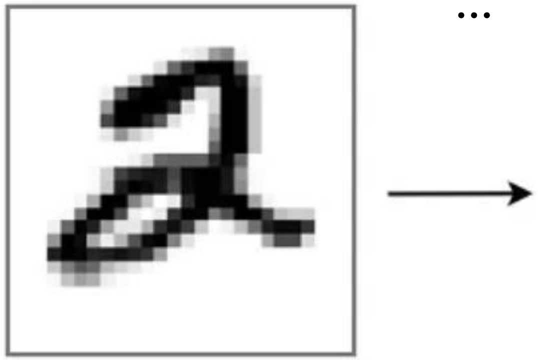


Image is transformed to vector of pixels

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?



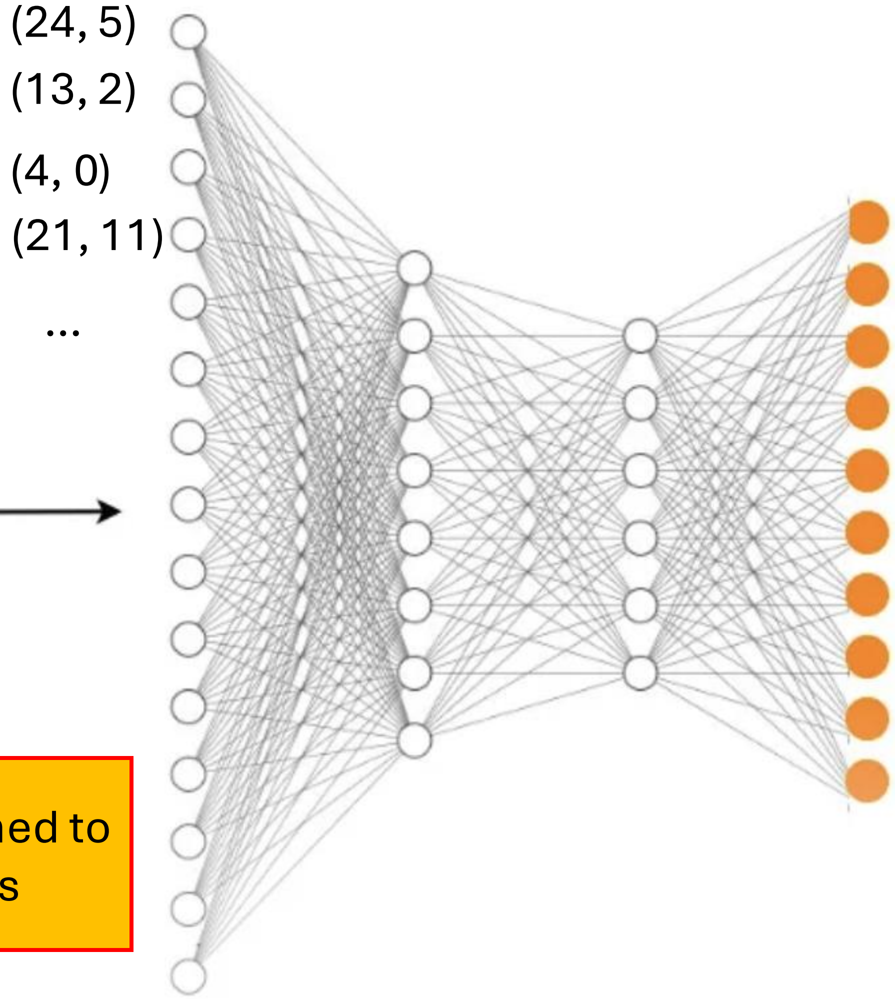


Image is transformed to vector of pixels

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

MLPs and Spatial Reasoning

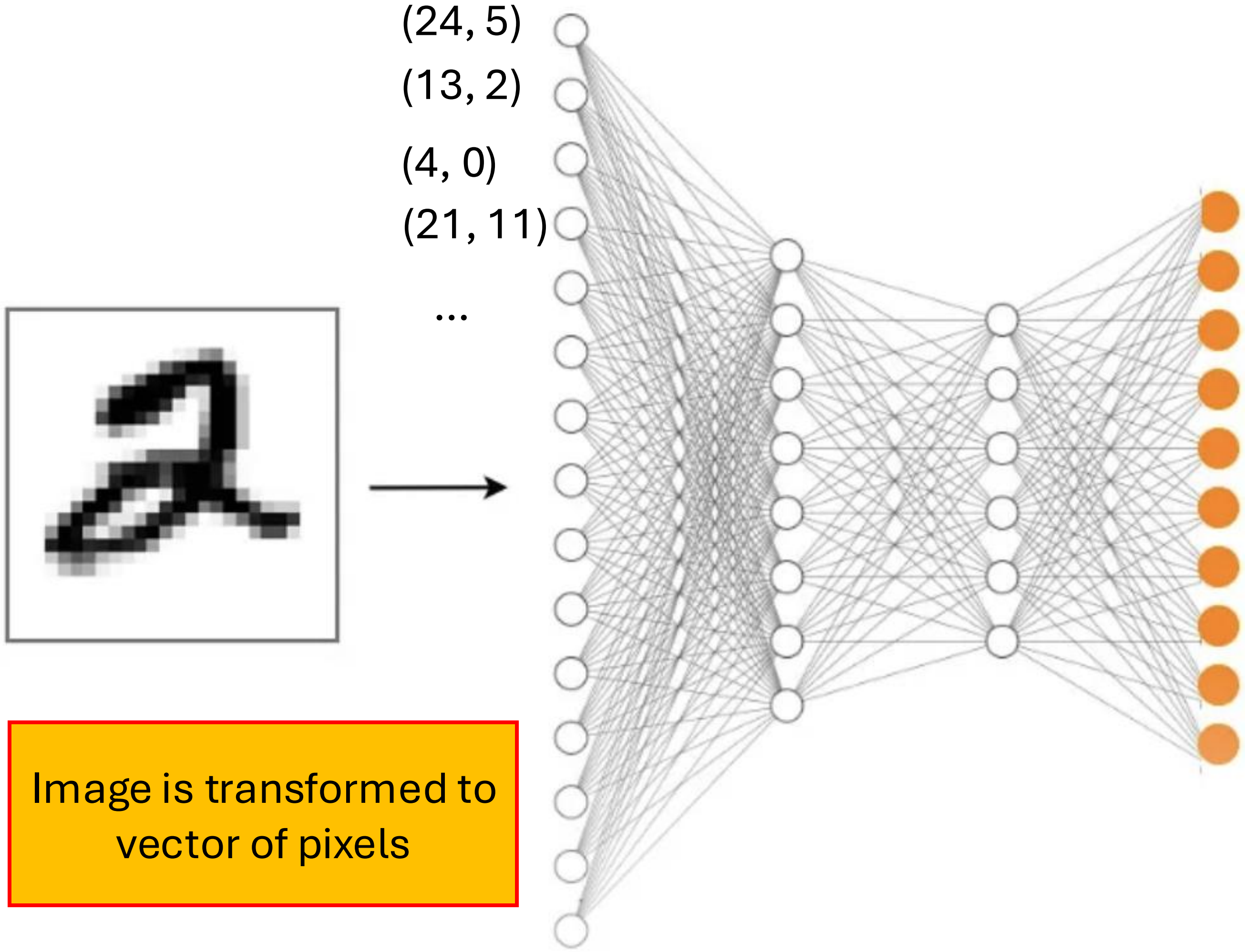


Image is transformed to vector of pixels

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

Will the training of the neural network differ?

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

MLPs and Spatial Reasoning

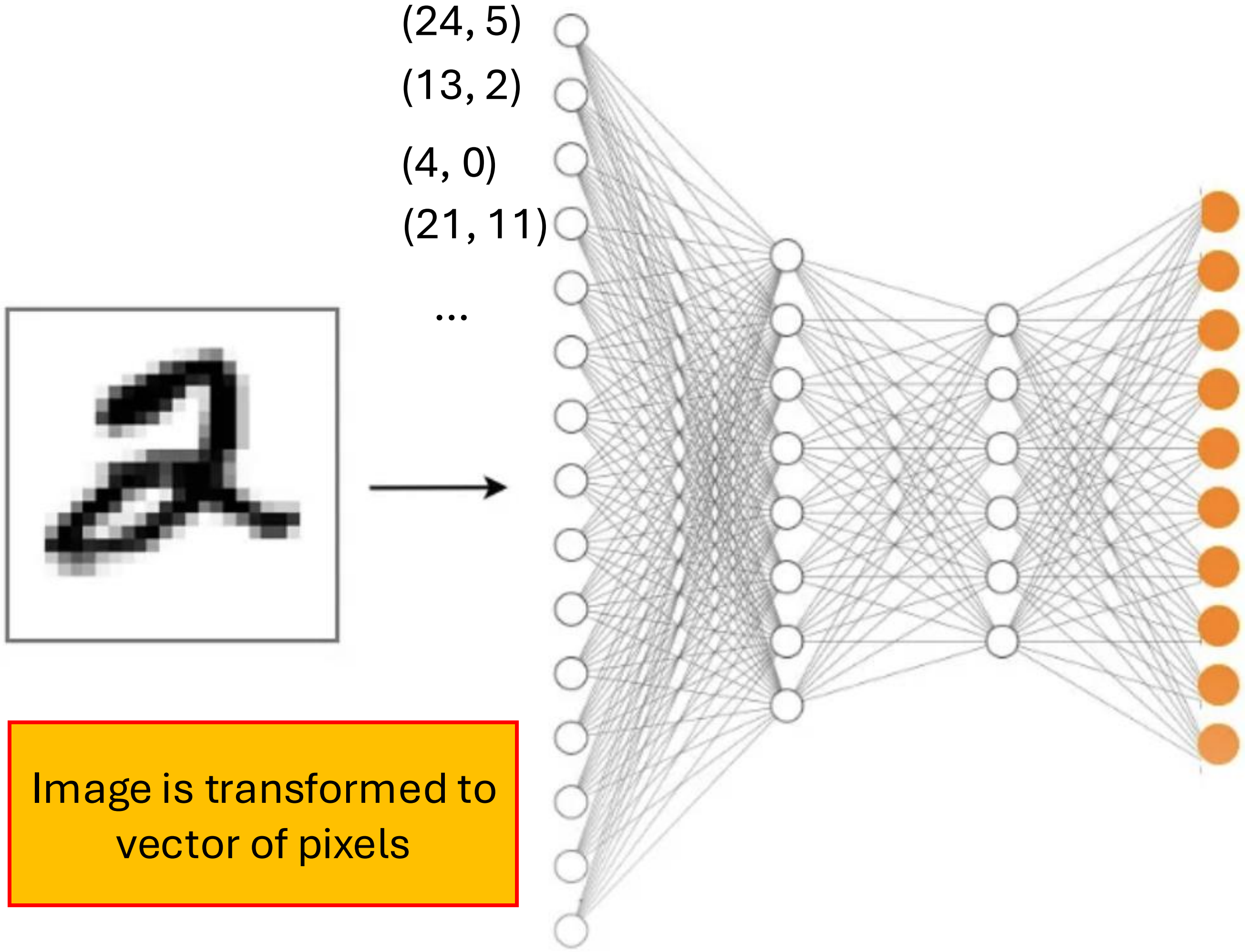


Image is transformed to vector of pixels

What would happen if we permuted the ordering of the 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

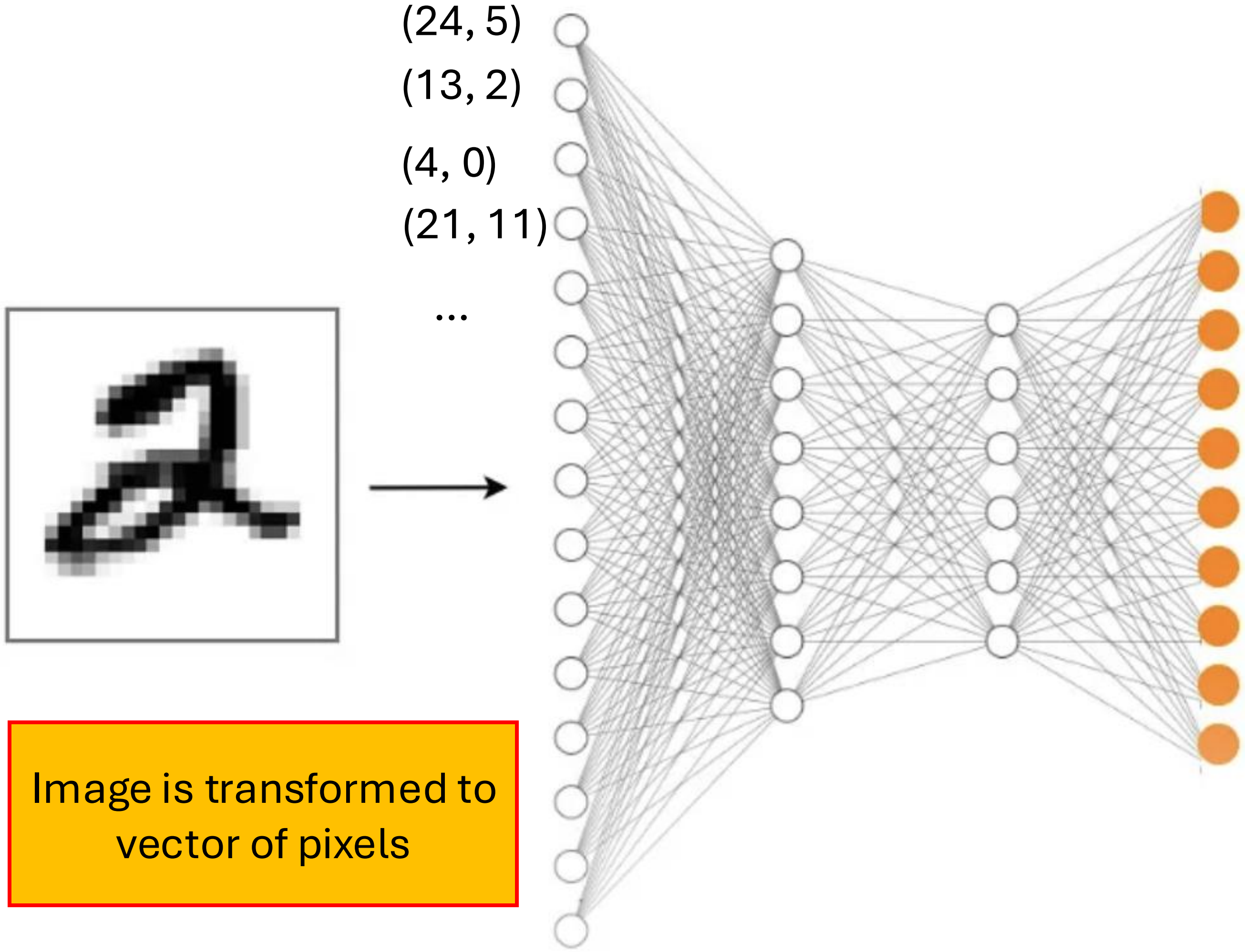


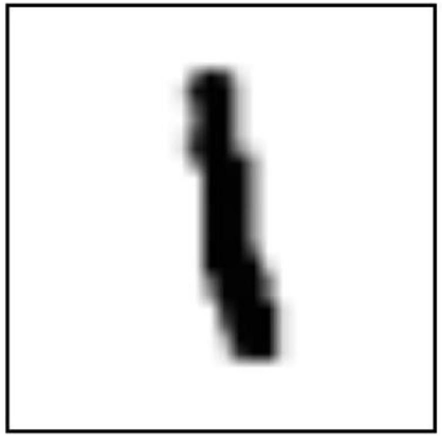
Image is transformed to vector of pixels

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...



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

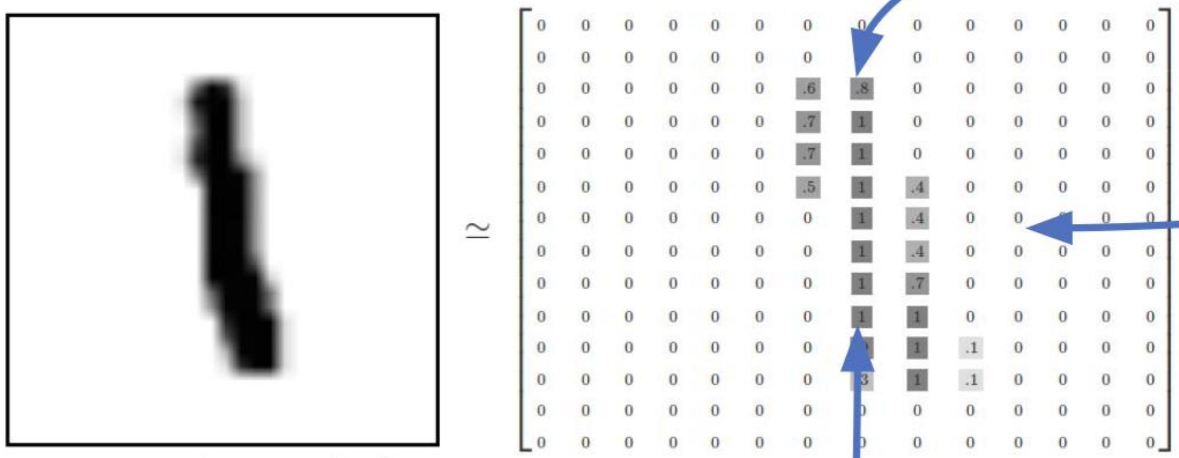
#1 encoded as □

Limitations of Full Connections for MNIST

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

this pixel gets weight 0.6

this pixel gets weight 0.1

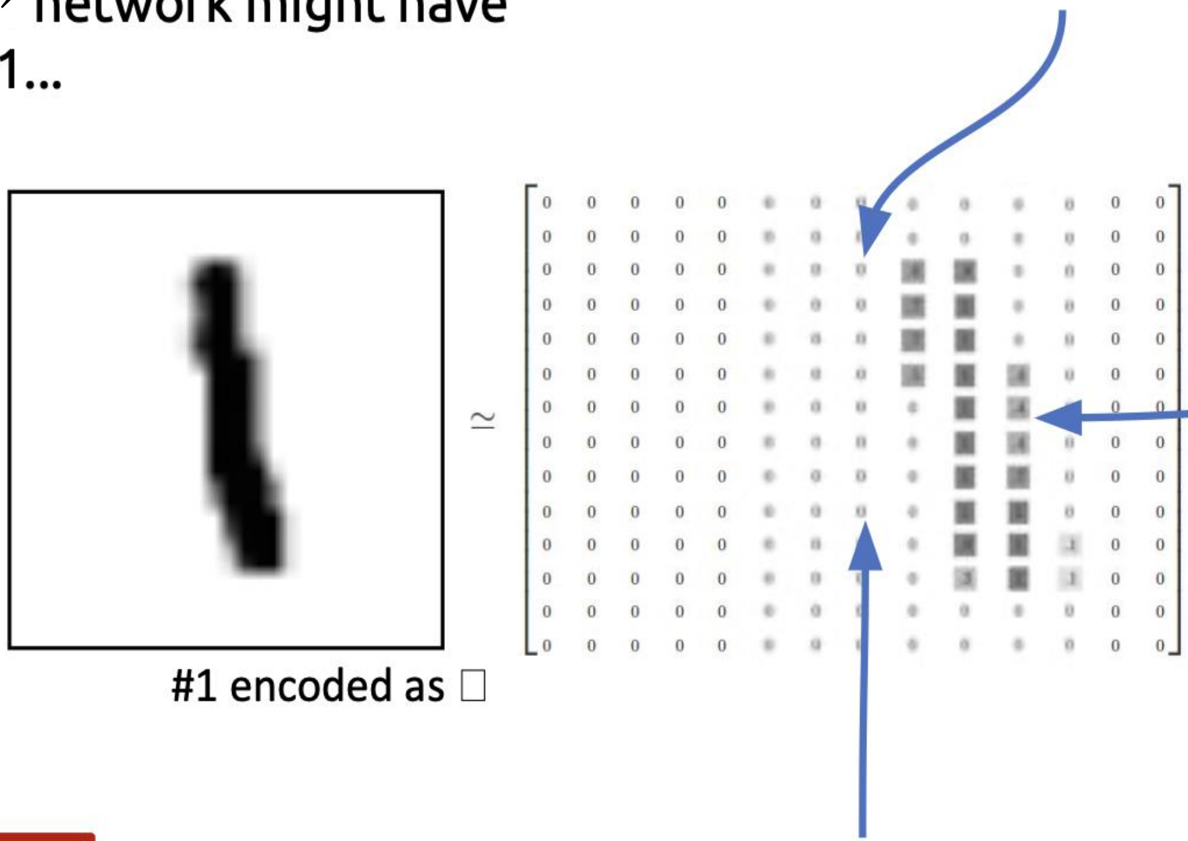


#1 encoded as □

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 → network might have trouble classifying this as a 1…

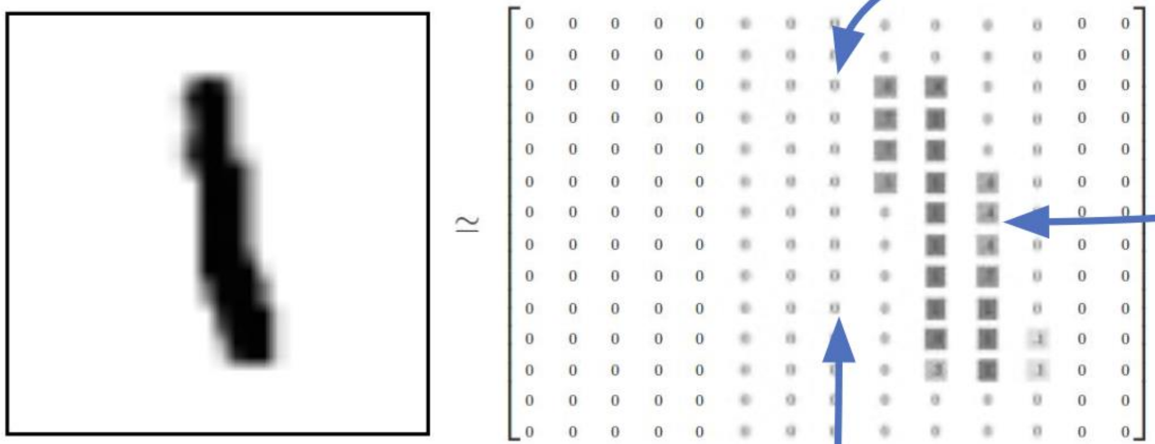


#1 encoded as □

Can you tell this is a 1?

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

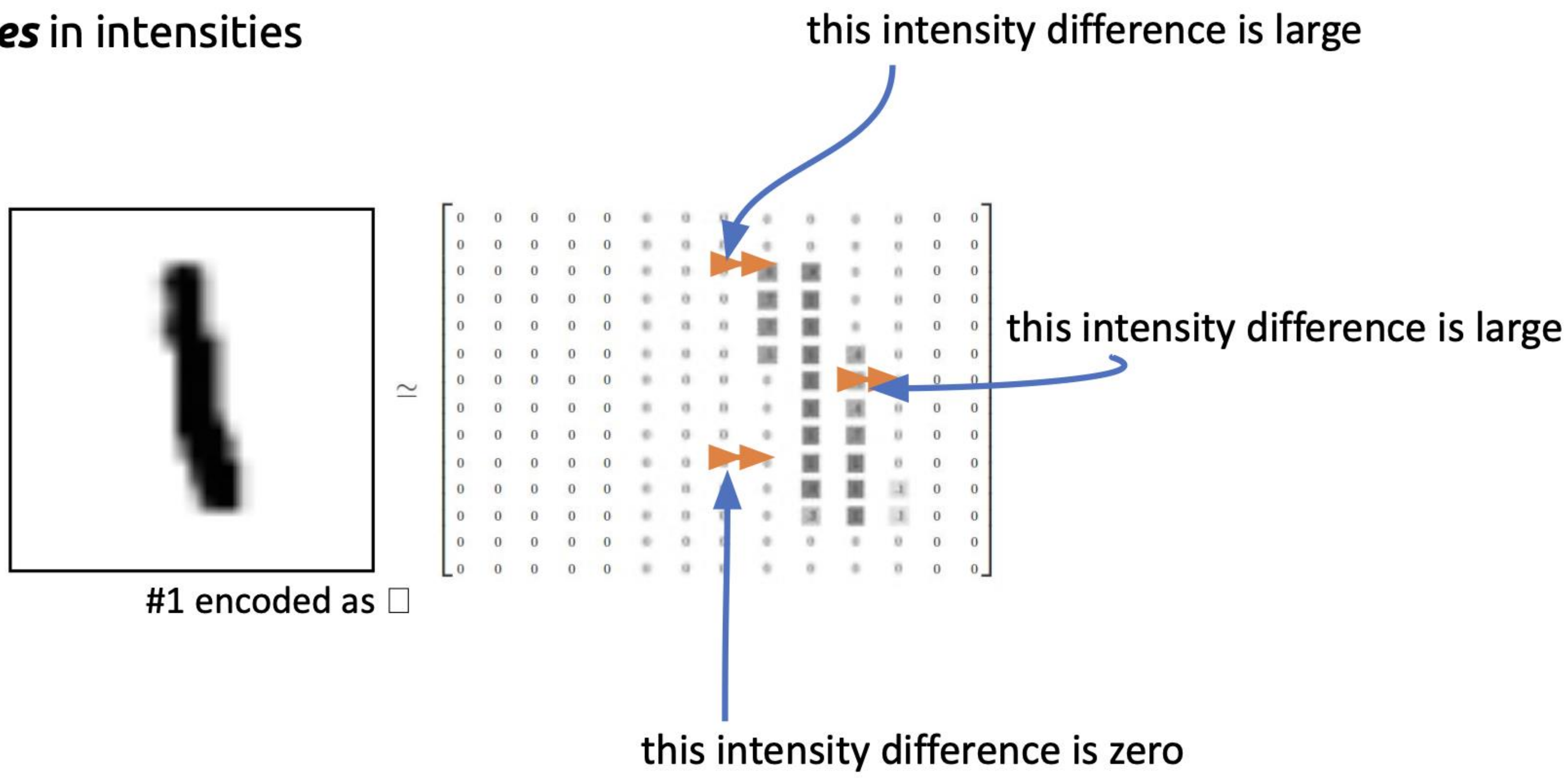
Our eyes don't look at absolute intensity values...



#1 encoded as

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

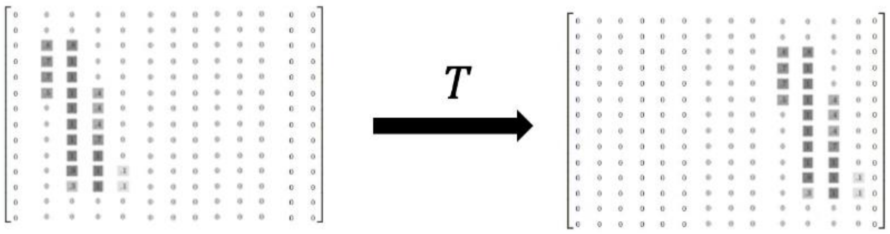
…but rather ***local differences*** in intensities

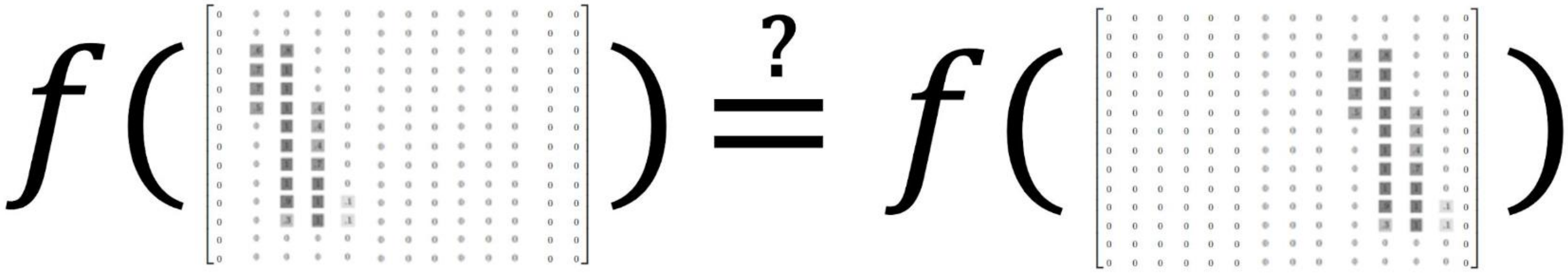


#1 encoded as □

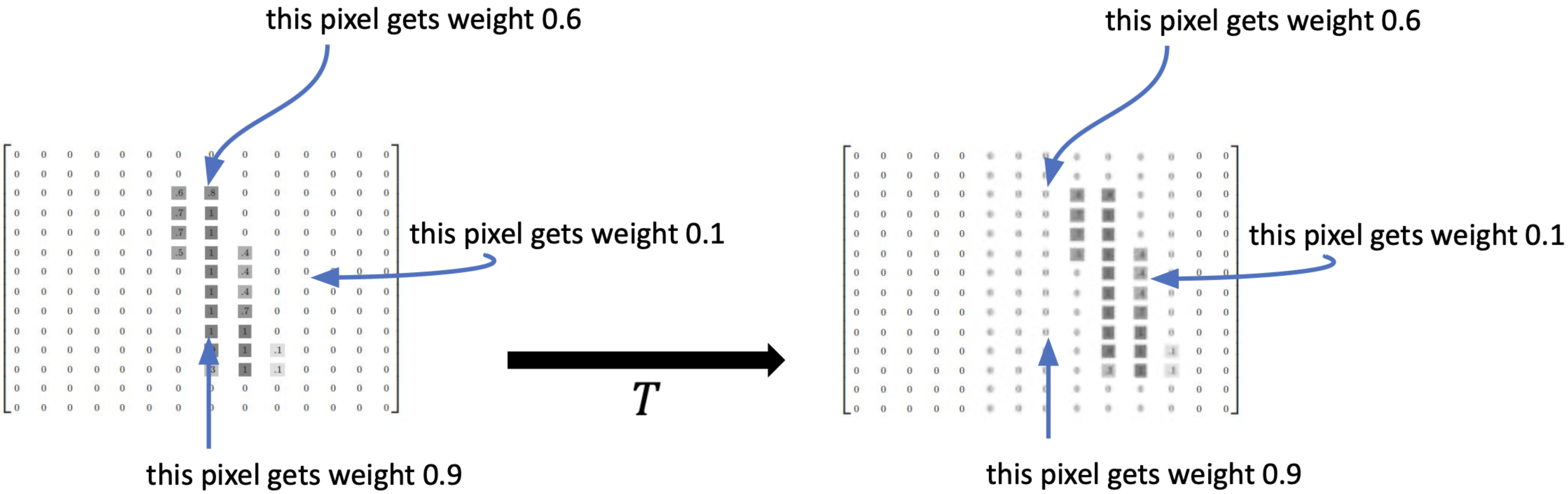
Translational Invariance

- To make a neural net f robust in this same way, it should ideally satisfy ***translational invariance***: $f\left(T(x)\right) = f(x)$, 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

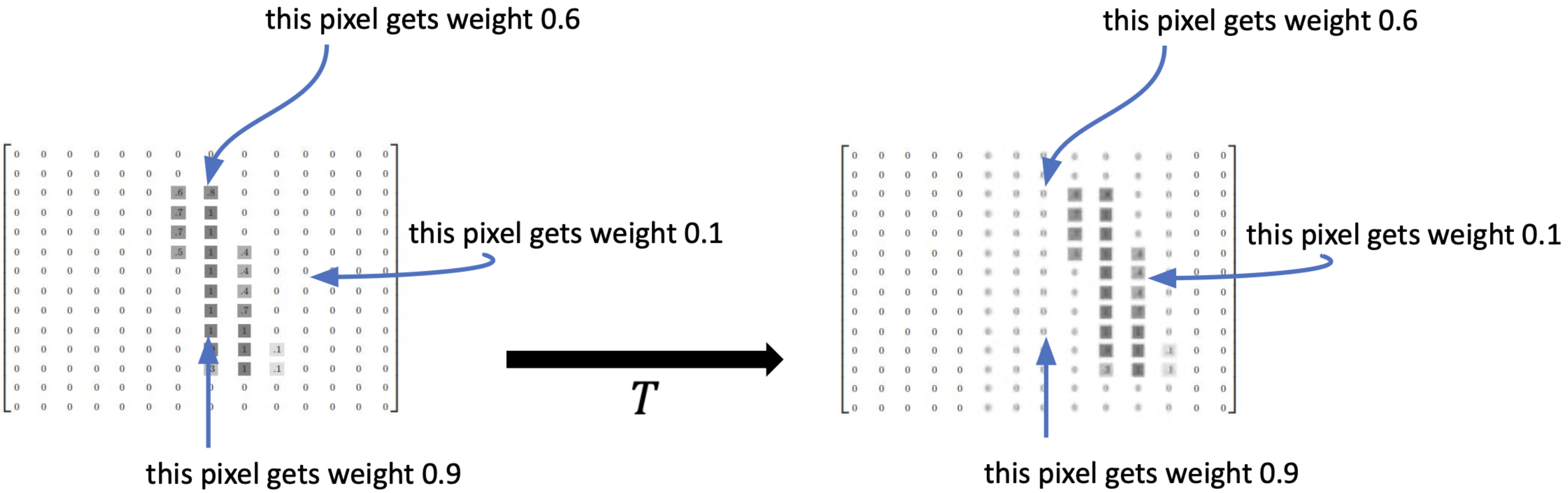


Sum of these three: $0.6 \cdot 0.8 + 0.1 \cdot 0 + 0.9 \cdot 1 = 1.38$

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

How to make the network translationally invariant?

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



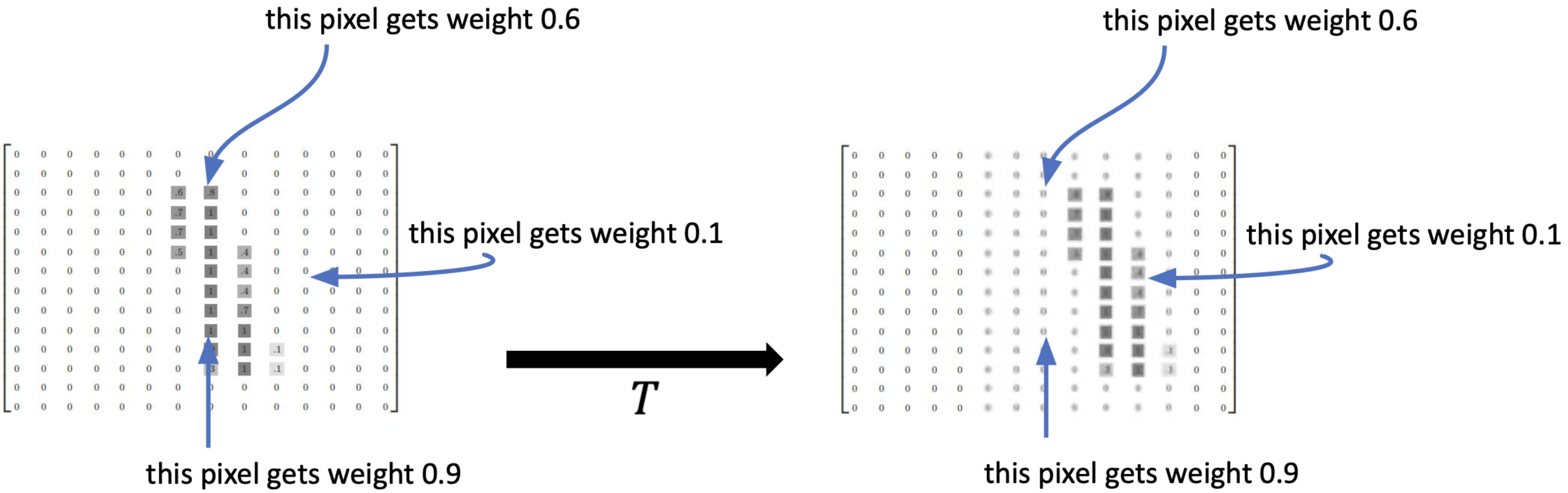
Sum of these three: $0.6 \cdot 0.8 + 0.1 \cdot 0 + 0.9 \cdot 1 = 1.38$

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

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

How to make the network translationally invariant?

Focus on local differences/patterns

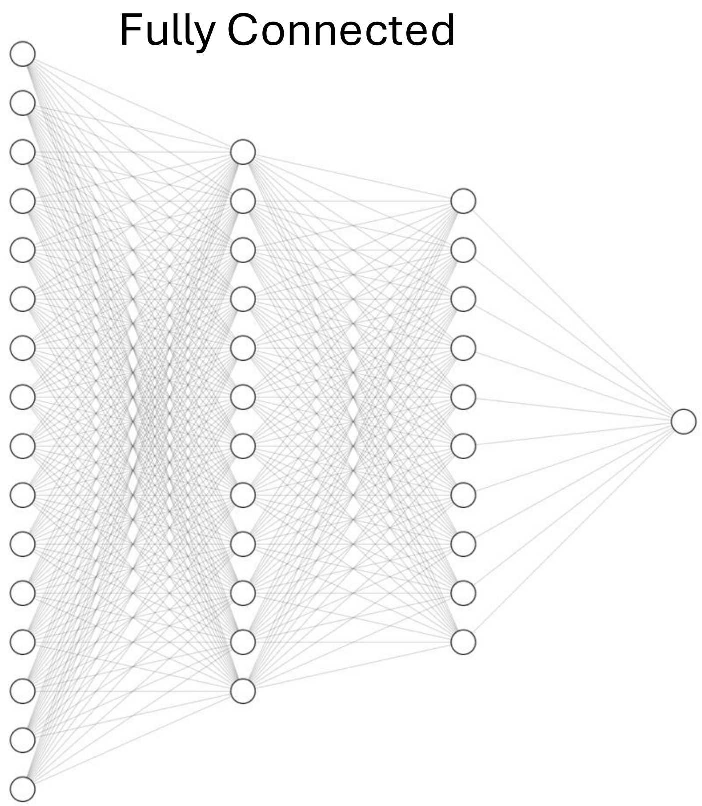


Sum of these three: $0.6 \cdot 0.8 + 0.1 \cdot 0 + 0.9 \cdot 1 = 1.38$

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

MLPs and Spatial Reasoning

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

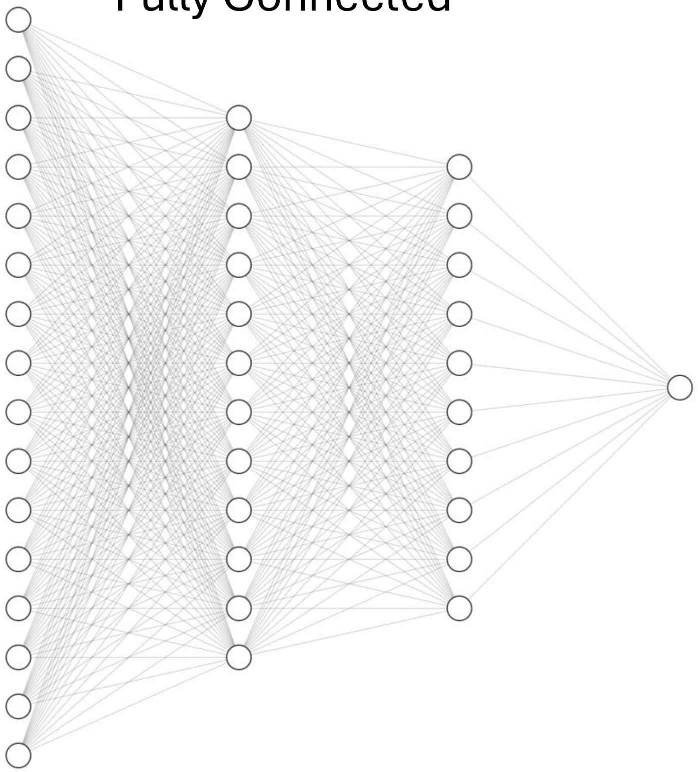


MLPs and Spatial Reasoning

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

Fully Connected

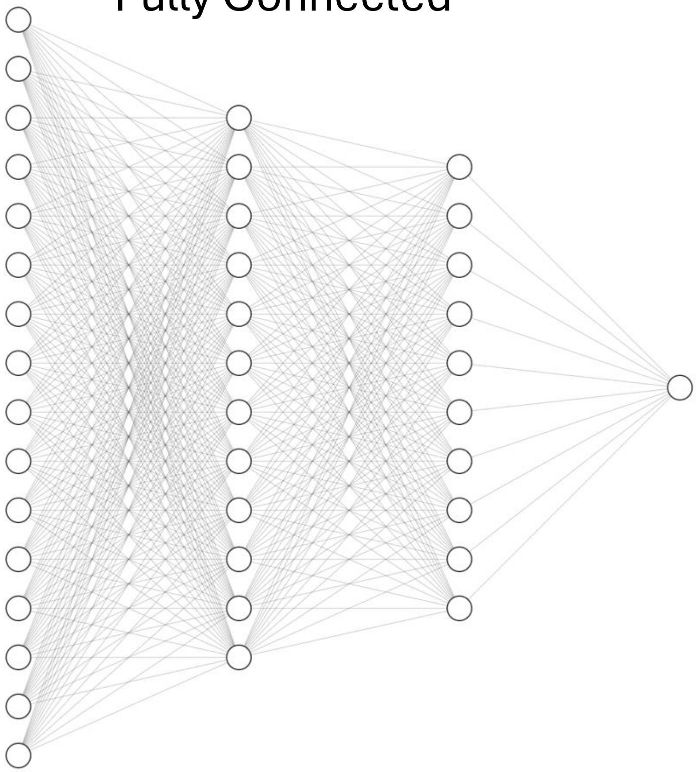


MLPs and Spatial Reasoning

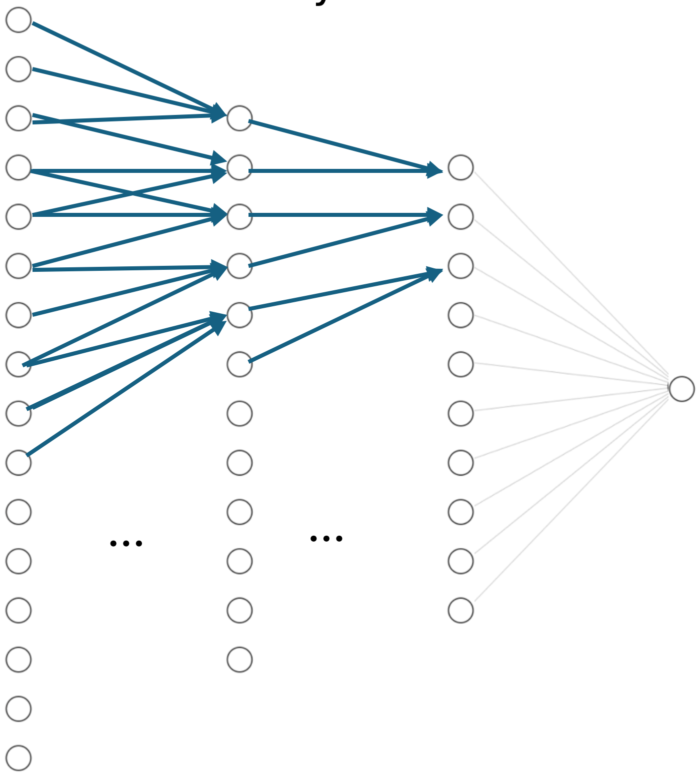
How can we change a fully-connected network to account for spatial information?

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

Fully Connected

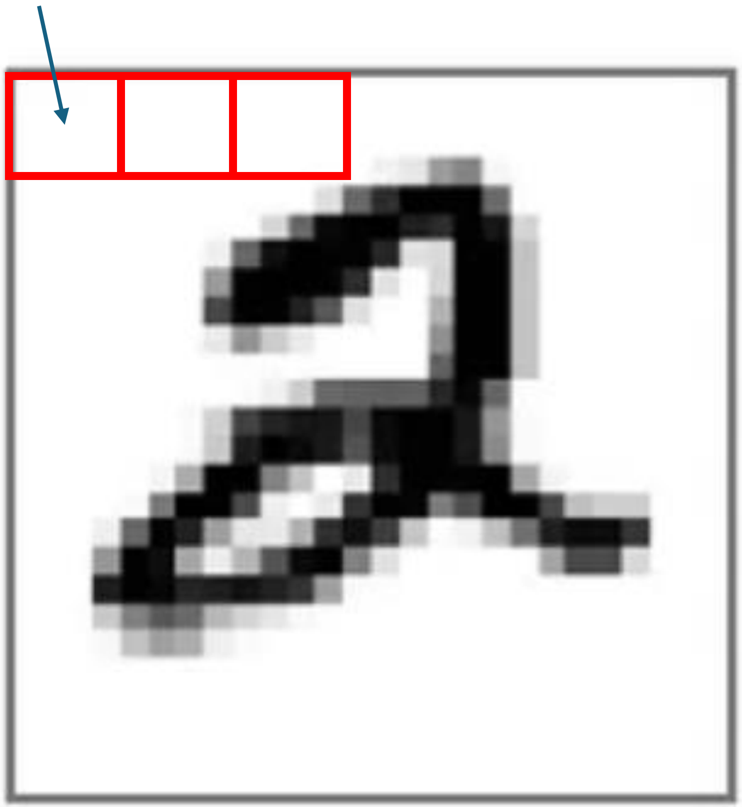


Not Fully-Connected



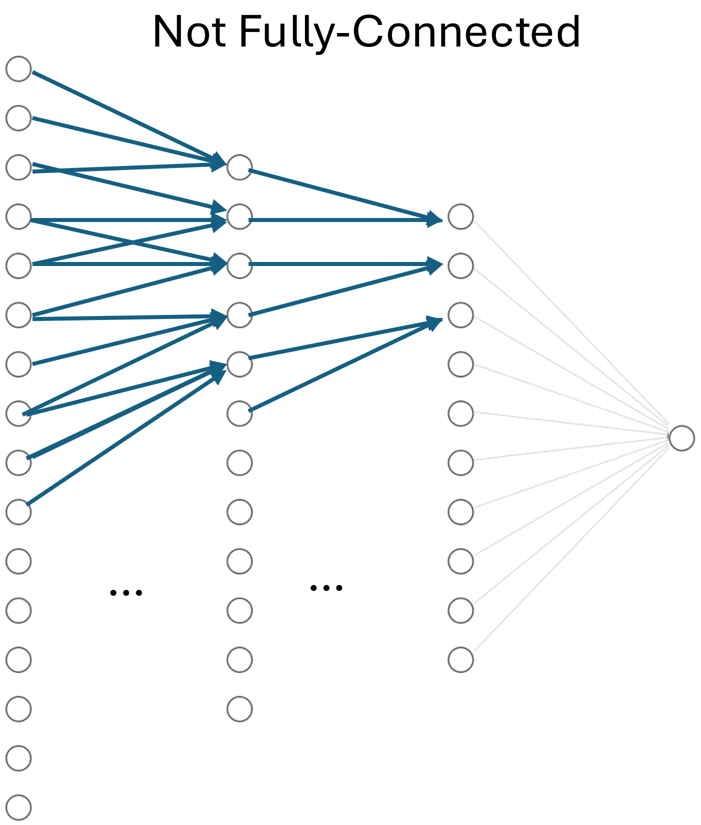
MLPs and Spatial Reasoning

Patches: Pixels close to each other



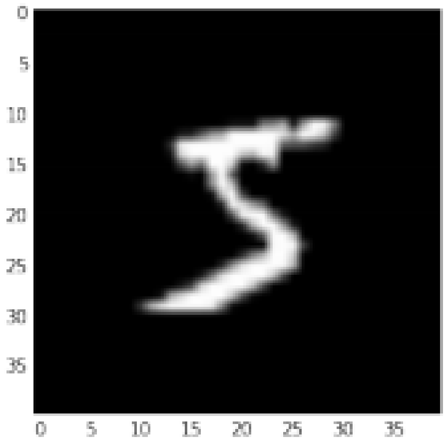
Advantages of Not Fully Connected Layers

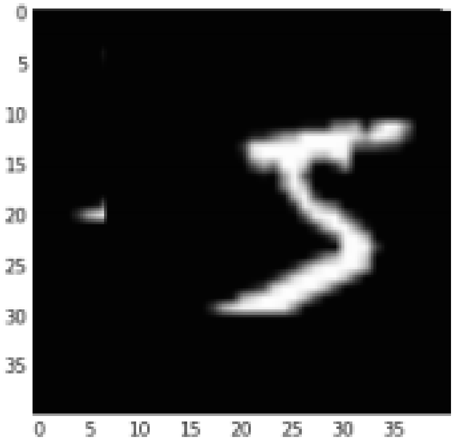
- Fewer weights → Faster?
- The outputs of neurons are "features" for local "patches"
- Incorporates spatial information (pixels that are close together matter)



Disadvantages of Not Fully Connected Layers

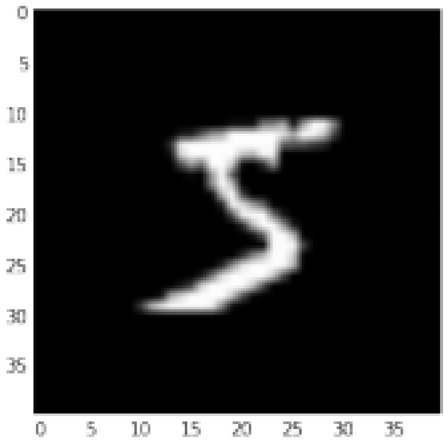
- What happens if the image is Translated?
- The patches on the right side were never trained with 5's in that side.

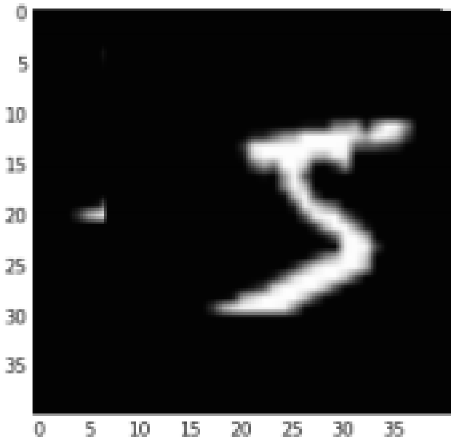




Disadvantages of Not Fully Connected Layers

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

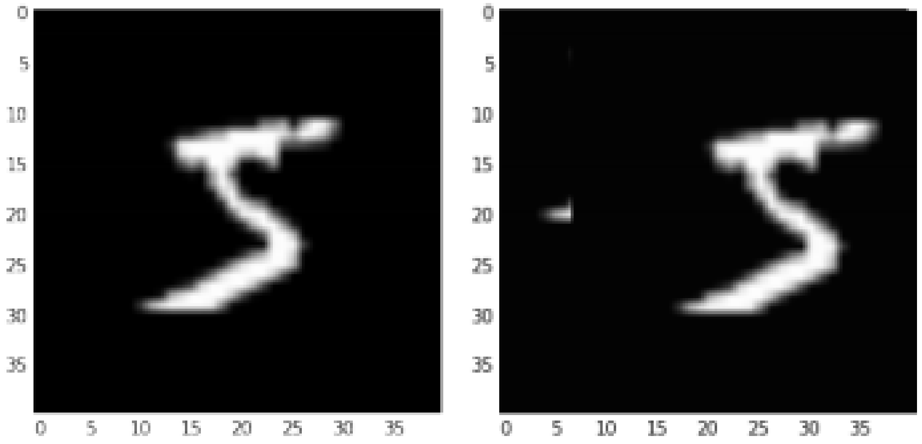




Even though we include spatial ***information***, we still don't have spatial ***reasoning***. *(Can't recognize a shifted 5 is still a 5)*

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

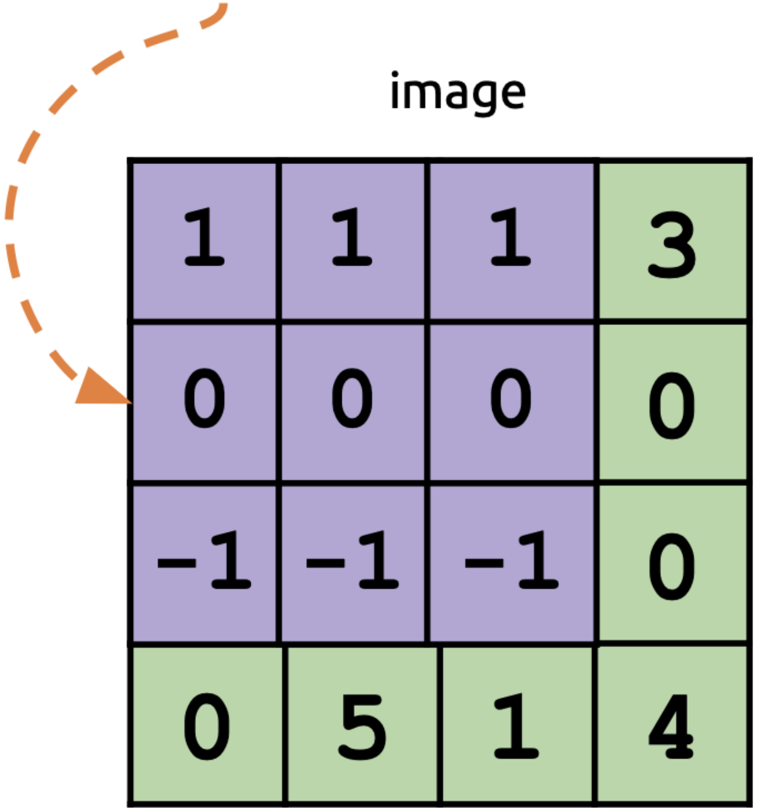
1	1	1
0	0	0
−1	−1	−1

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

What Convolution Does (Visually)

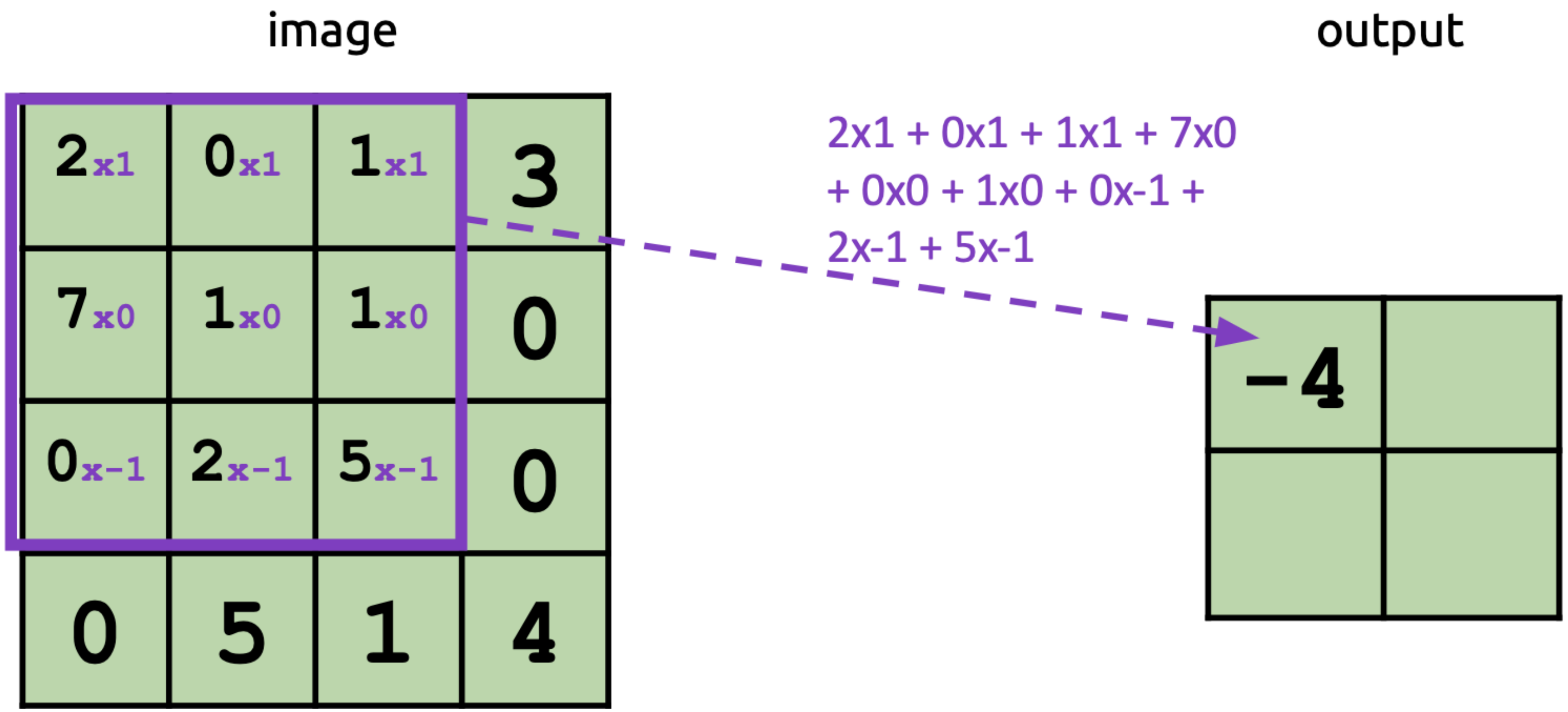
Overlay the filter on the image

image



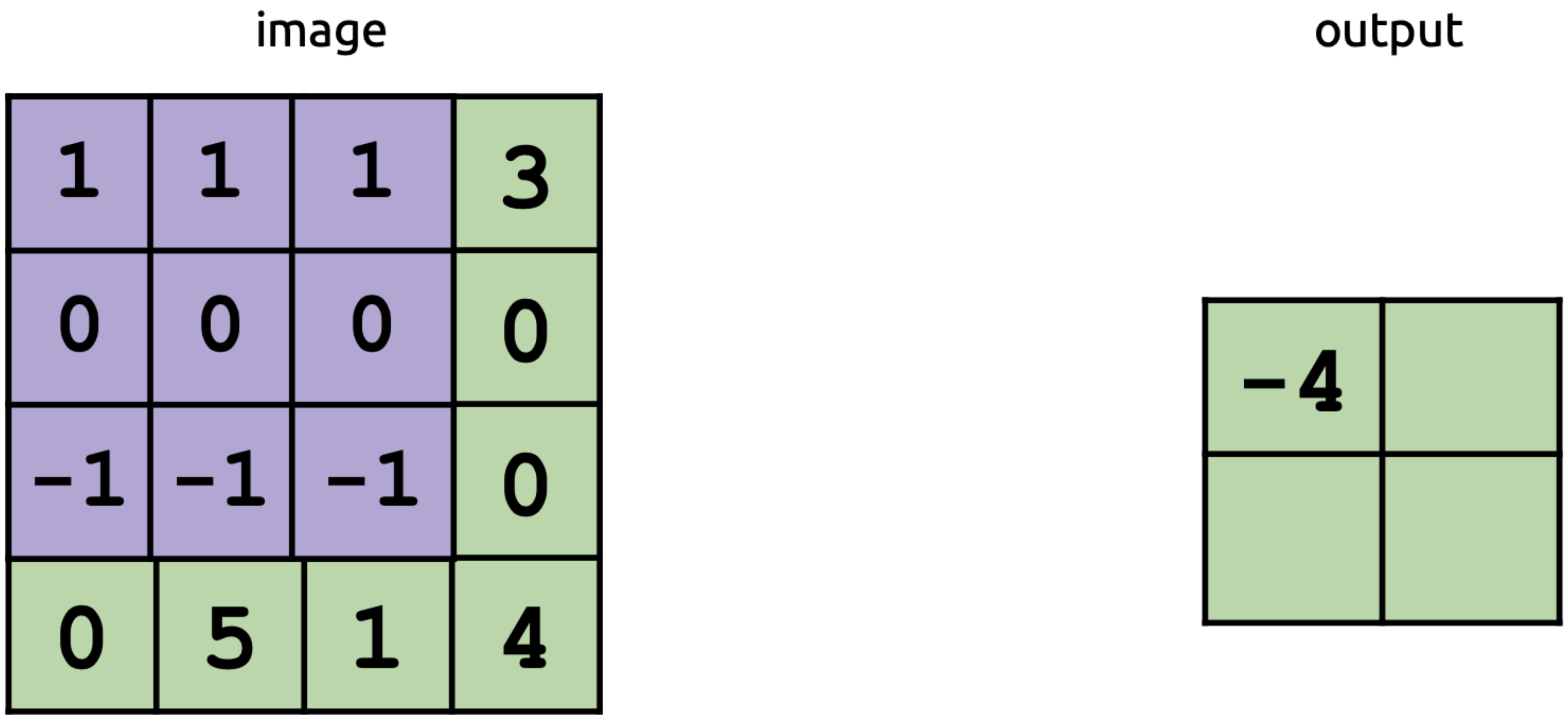
What Convolution Does (Visually)

Sum up multiplied values to produce output value



What Convolution Does (Visually)

Move the filter over by one pixel



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

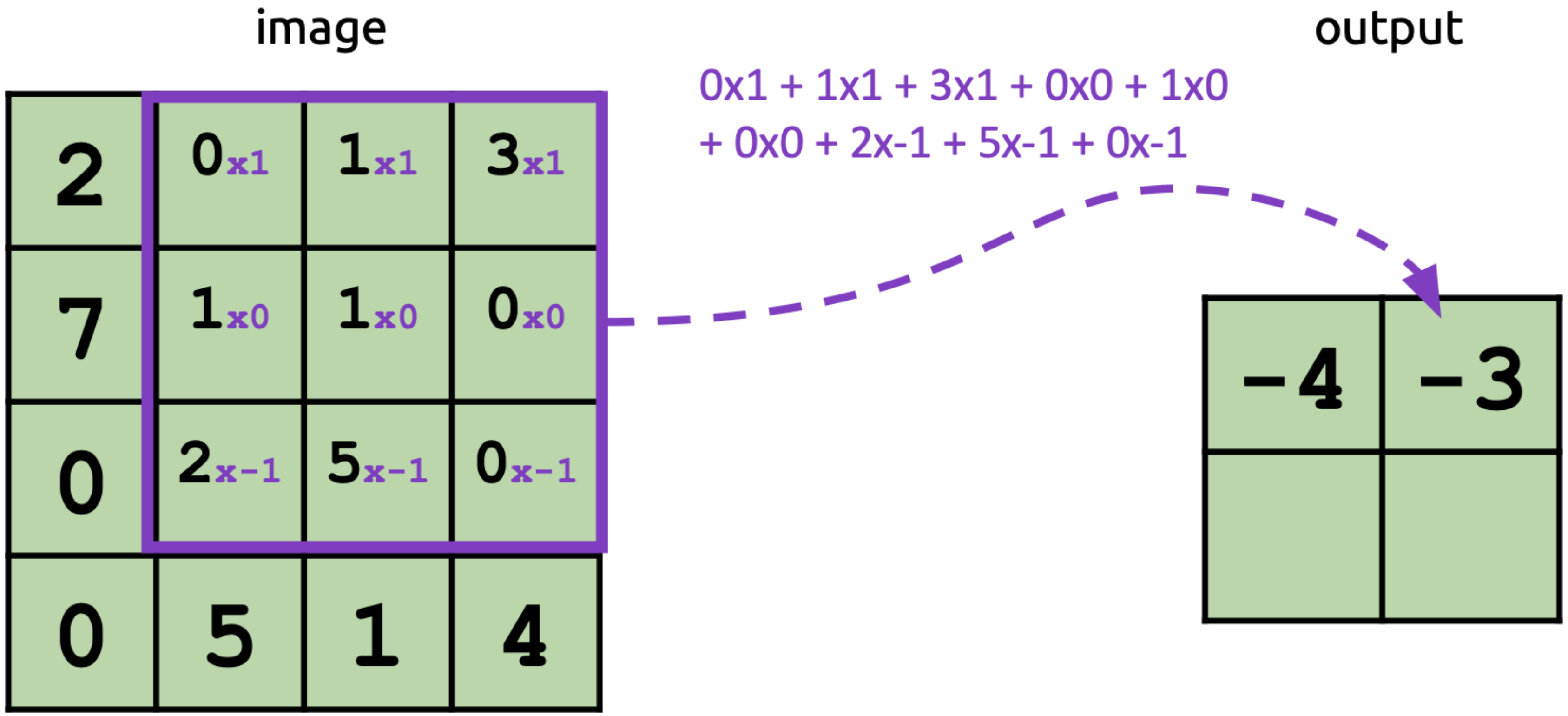
2	0_{x1}	1_{x1}	3_{x1}
7	1_{x0}	1_{x0}	0_{x0}
0	2_{x-1}	5_{x-1}	0_{x-1}
0	5	1	4

output

-4	

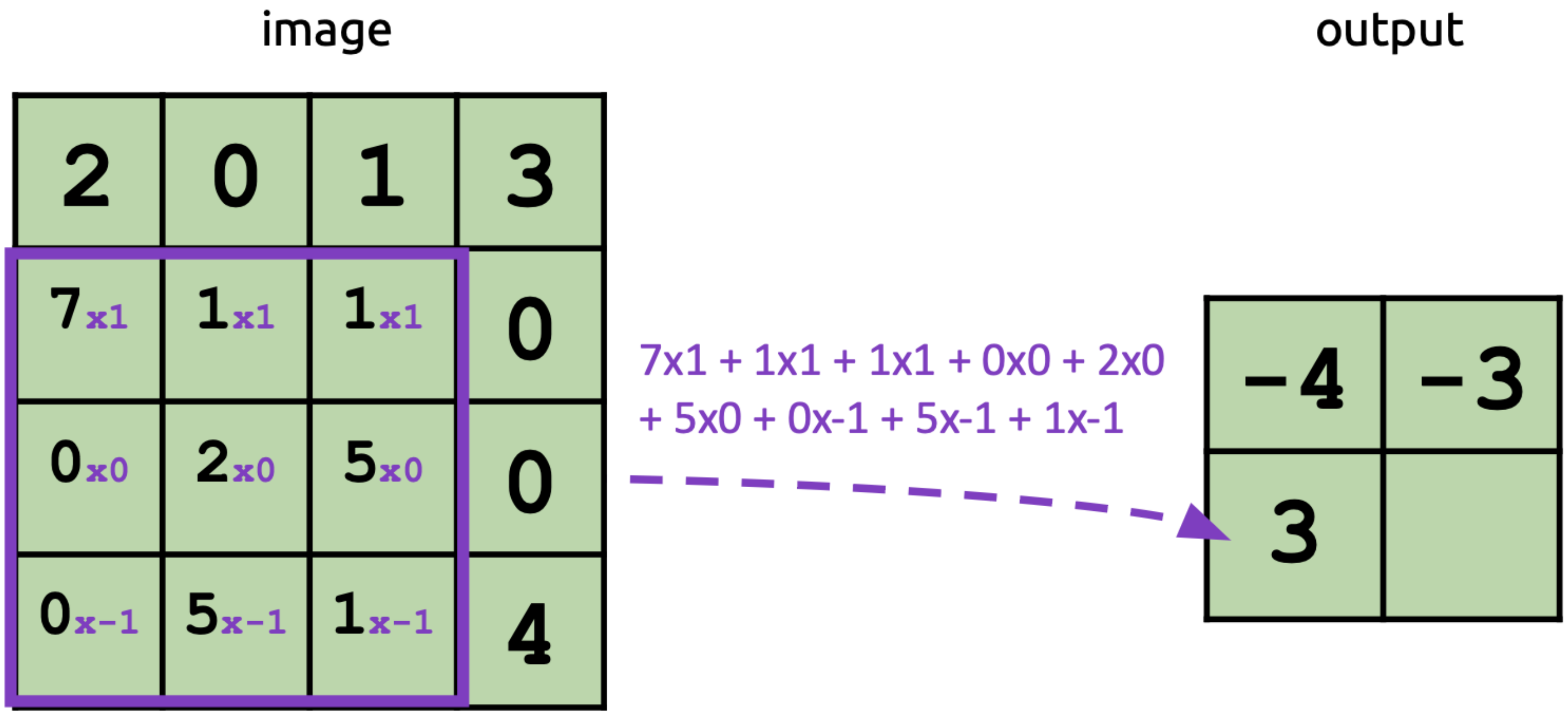
What Convolution Does (Visually)

Repeat (multiply, sum up)

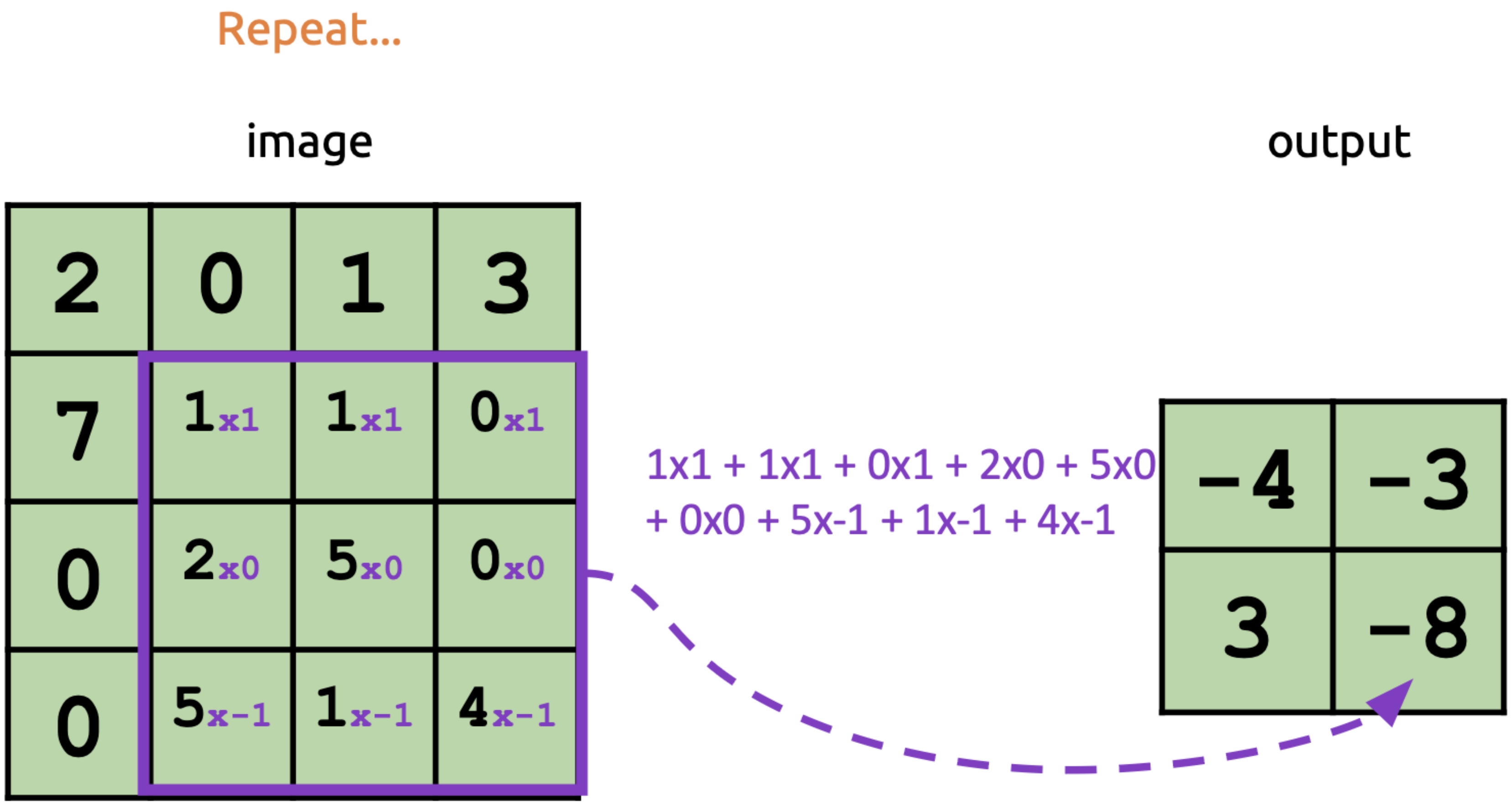


What Convolution Does (Visually)

Repeat...



What Convolution Does (Visually)



What Convolution Does (Visually)

In summary:

$$\underset{\text{image}}{\begin{bmatrix} 2 & 0 & 1 & 3 \\ 7 & 1 & 1 & 0 \\ 0 & 2 & 5 & 0 \\ 0 & 5 & 1 & 4 \end{bmatrix}} \otimes \underset{\text{filter/kernel}}{\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}} = \underset{\text{output}}{\begin{bmatrix} -4 & -3 \\ 3 & -8 \end{bmatrix}}$$

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)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Source: Wikipedia

What Convolution Does (Mathematically)

$$V(x,y) = (\mathrm{I} \otimes K)(x,y) = \sum_{m}\sum_{n} I(x+m, y+n)K(m,n)$$

The output at pixel (x,y)

"Image I convolved with kernel K"

Sum over kernel columns

Sum over kernel rows

Multiply kernel value with corresponding image pixel value

What Convolution Does (Mathematically)

$$\underbrace{V(x,y)}_{\text{The output at pixel } (x,y)} = \underbrace{(\mathrm{I}\otimes K)}_{\text{“Image } I \text{ convolved with kernel } K\text{”}}(x,y) = \underbrace{\sum_{m}}_{\text{Sum over kernel columns}} \underbrace{\sum_{n}}_{\text{Sum over kernel rows}} \underbrace{I(x+m, y+n)K(m,n)}_{\text{Multiply kernel value with corresponding image pixel value}}$$

Is this a matrix multiplication?

What Convolution Does (Mathematically)

$$V(x,y) = (I \otimes K)(x,y) = \sum_m \sum_n \underbrace{I(x+m, y+n)K(m,n)}_{\text{Multiply kernel value with corresponding image pixel value}}$$

The output at pixel (x, y)

"Image I convolved with kernel K"

Sum over kernel columns

Sum over kernel rows

Is this a matrix multiplication?

No, output for each pixel is a single value, not a matrix.

What Convolution Does (Mathematically)

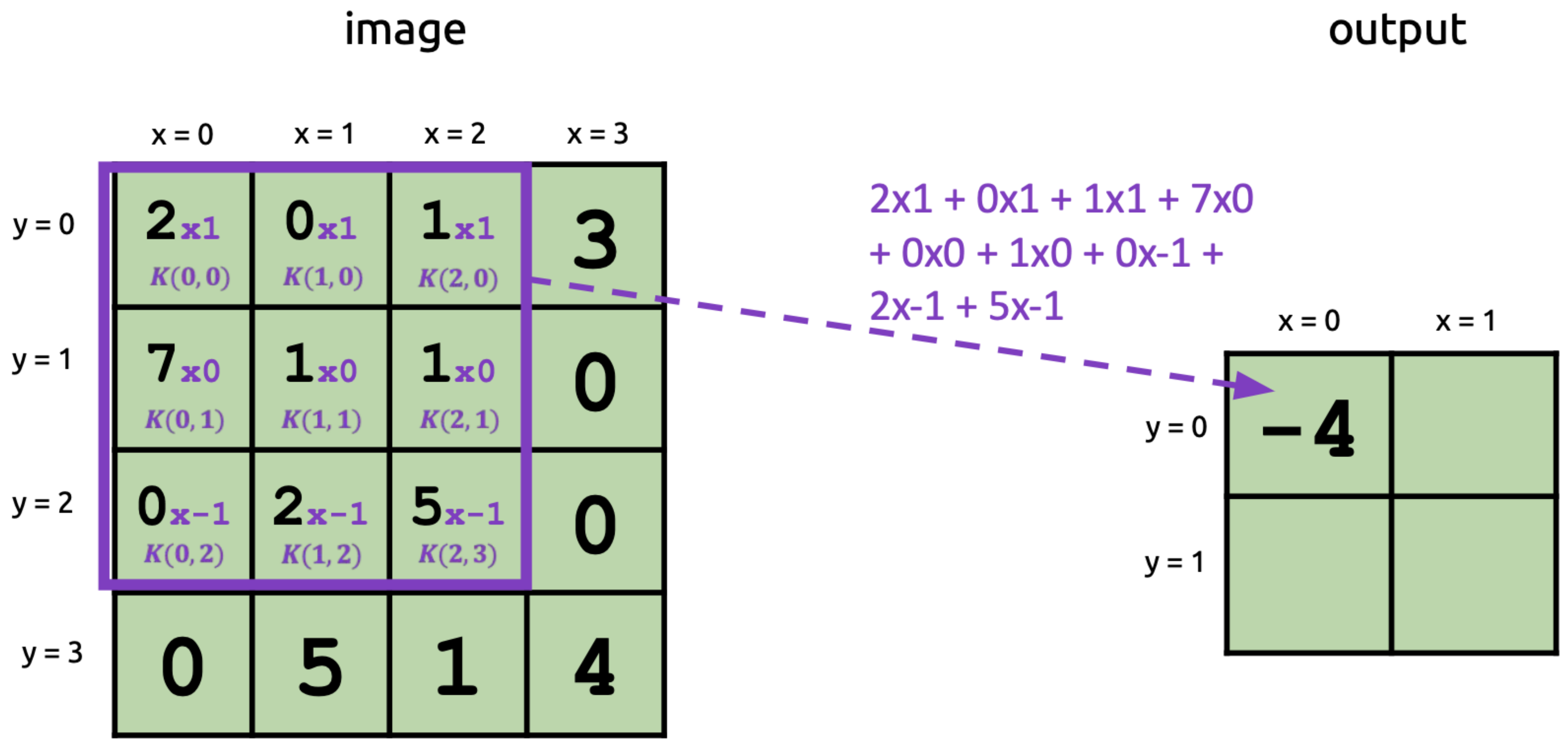
image

filter/kernel

output

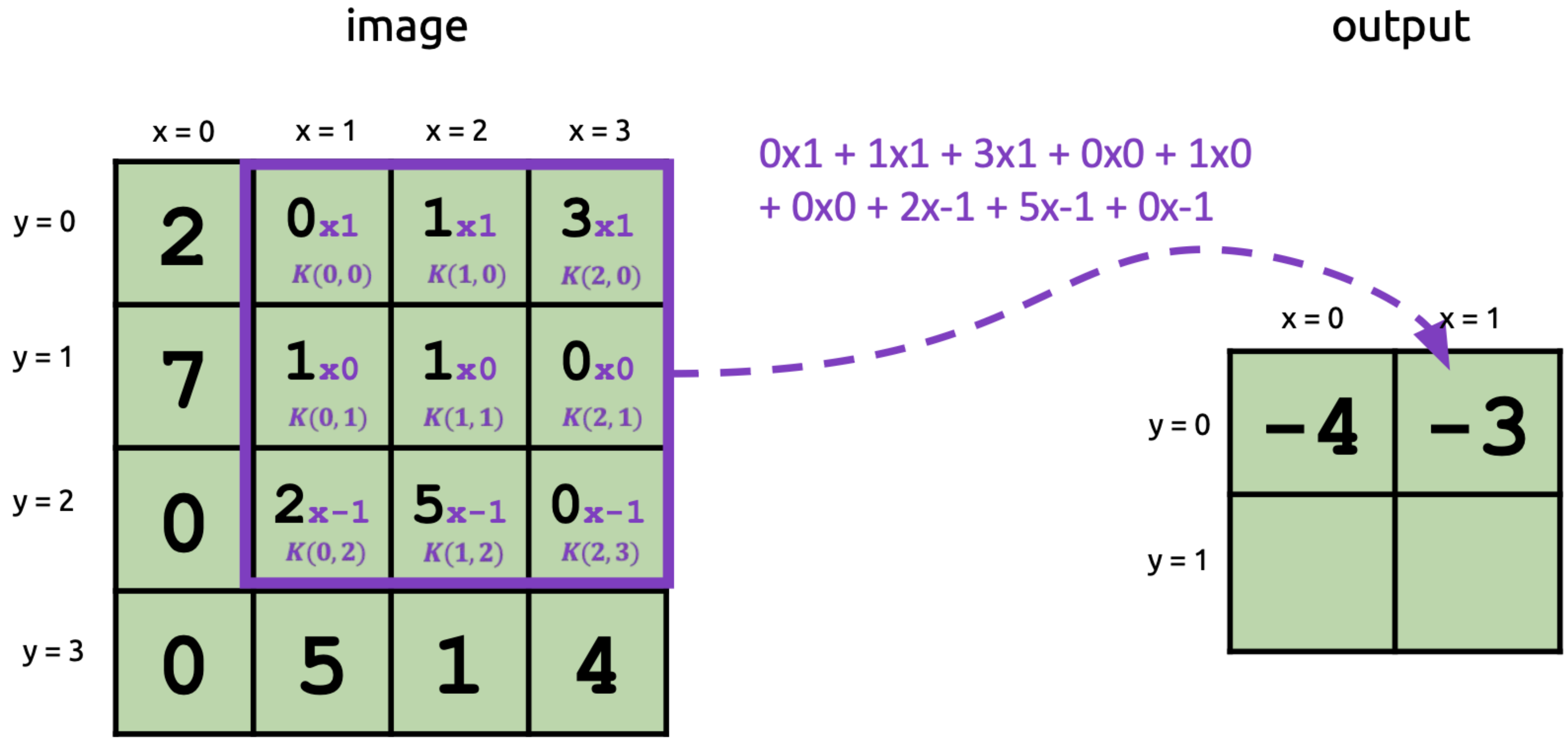
$$
\begin{array}{c|cccc}
 & x=0 & x=1 & x=2 & x=3 \\
\hline
y=0 & 2 & 0 & 1 & 3 \\
y=1 & 7 & 1 & 1 & 0 \\
y=2 & 0 & 2 & 5 & 0 \\
y=3 & 0 & 5 & 1 & 4
\end{array}
\otimes
\begin{array}{c|ccc}
 & m=0 & m=1 & m=2 \\
\hline
n=0 & 1 & 1 & 1 \\
n=1 & 0 & 0 & 0 \\
n=2 & -1 & -1 & -1
\end{array}
=
\begin{array}{c|cc}
 & x=0 & x=1 \\
\hline
y=0 & -4 & -3 \\
y=1 & 3 & -8
\end{array}
$$

What Convolution Does (Mathematically)



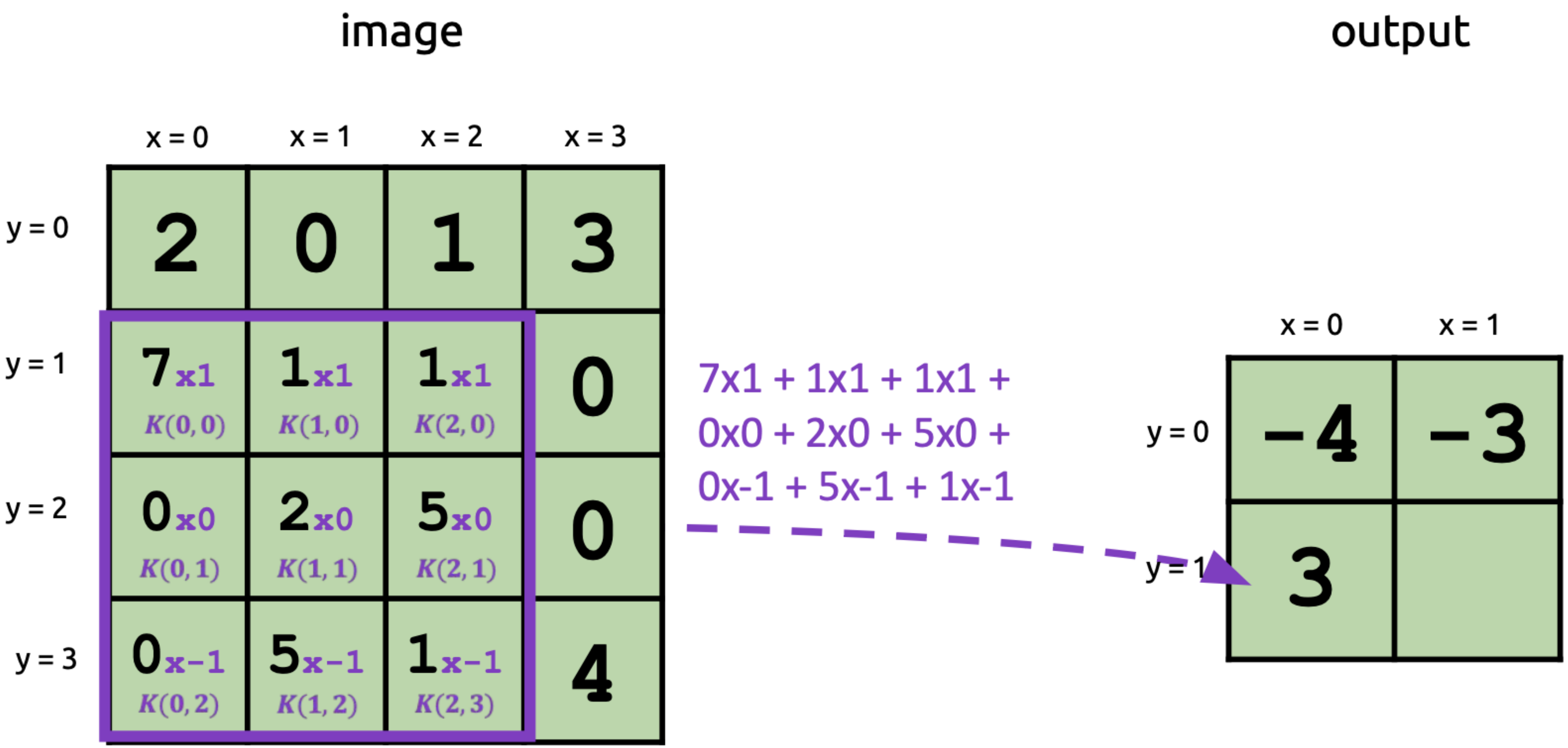
$$V(0,0) = (\mathrm{I} \otimes K)(0,0) = \sum_{m=0}^{2} \sum_{n=0}^{2} I(0+m, 0+n) K(m,n)$$

What Convolution Does (Mathematically)



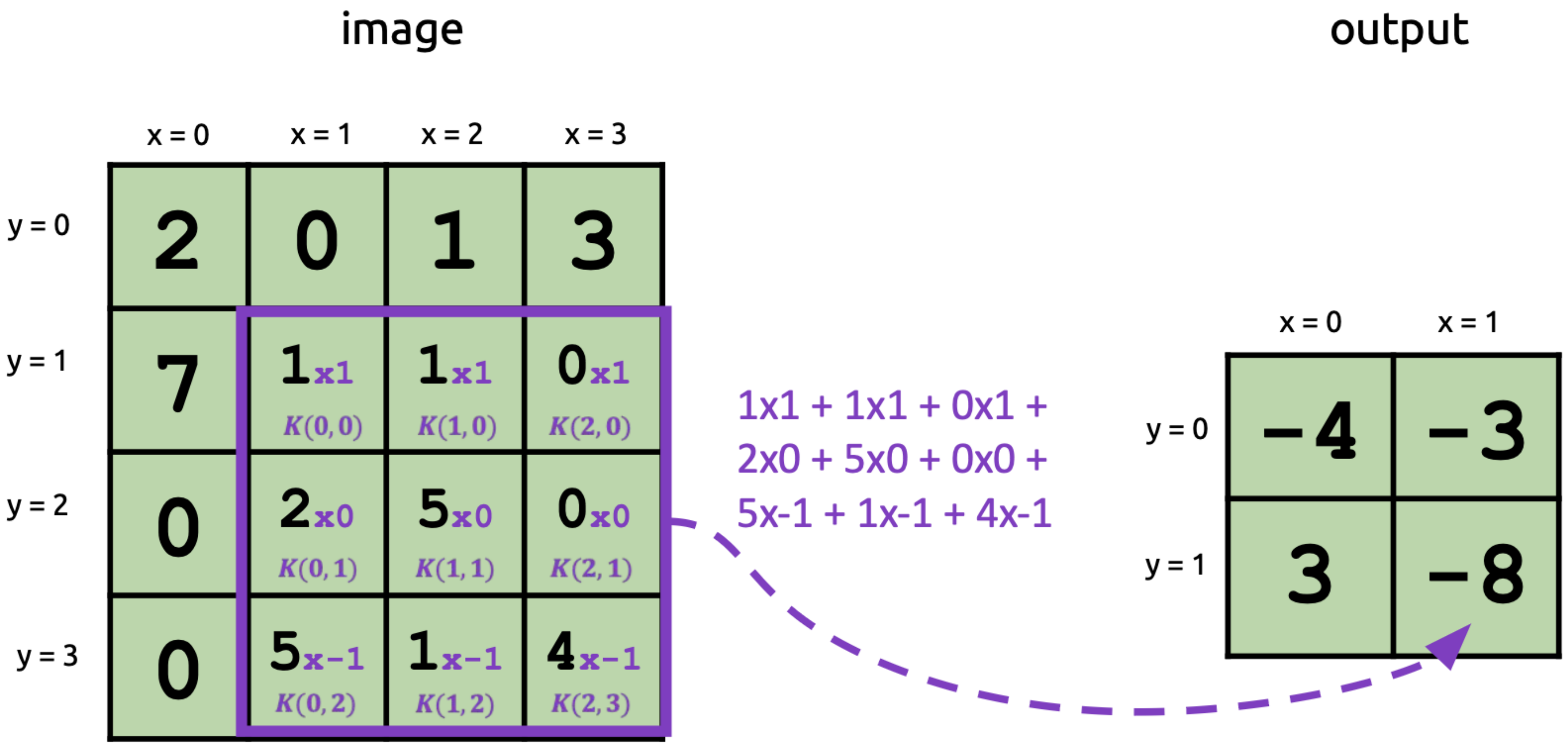
$$V(1,0) = (\mathrm{I} \otimes K)(1,0) = \sum_{m=0}^{2} \sum_{n=0}^{2} I(1+m, 0+n) K(m,n)$$

What Convolution Does (Mathematically)



$$V(0,1) = (\mathrm{I}\otimes K)(0,1) = \sum_{m=0}^{2}\sum_{n=0}^{2} I(0+m,1+n)K(m,n)$$

What Convolution Does (Mathematically)



$$V(1,1) = (\mathrm{I} \otimes K)(1,1) = \sum_{m=0}^{2} \sum_{n=0}^{2} I(1+m, 1+n) K(m,n)$$

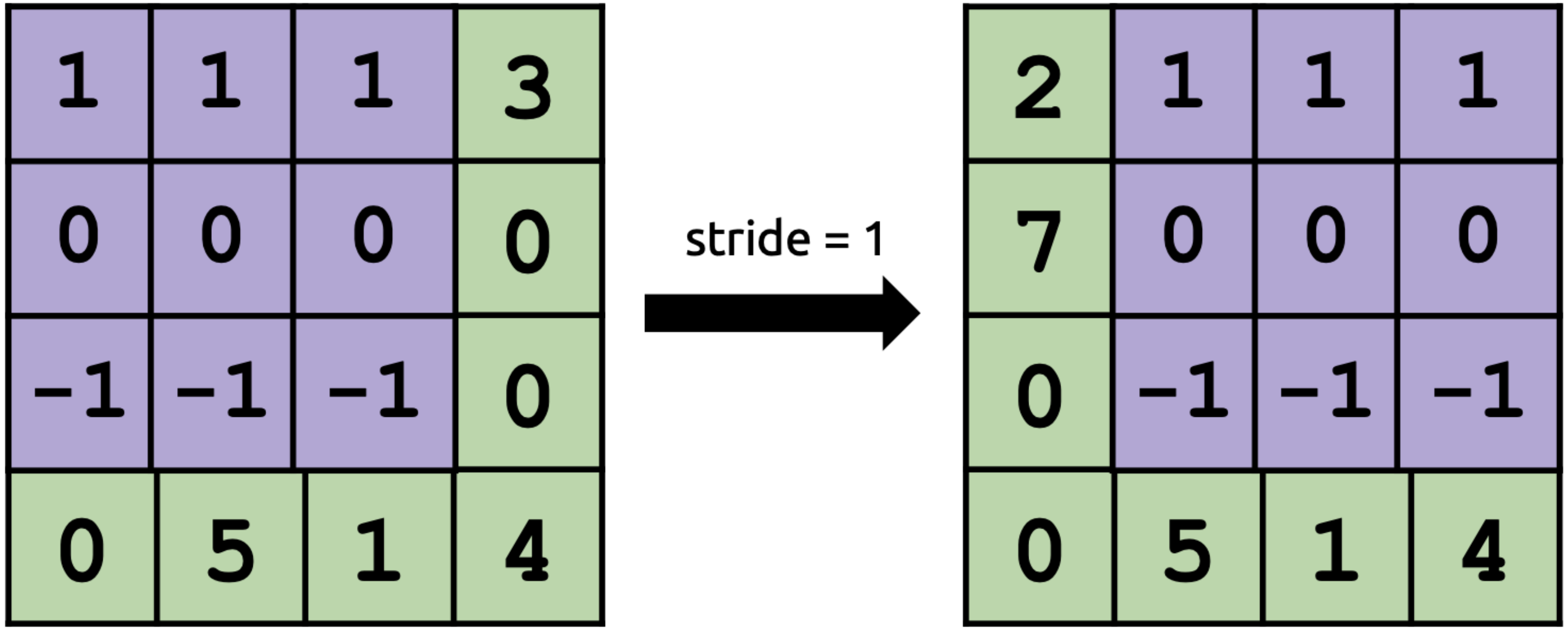
What Convolution Does (In Code)

```
// Input: Image I, Kernel K, Output V, pixel index x,y
// Assumes K is 3x3
function apply_kernel(I, K, V, x, y)
   for m = 0 to 2:
      for n = 0 to 2:
         V(x,y) += K(m,n) * I(m+x, n+y)
```

$$Equation: V(x,y) = (\mathrm{I} \otimes K)(x,y) = \sum_m \sum_n I(x+m, y+n) K(m,n)$$

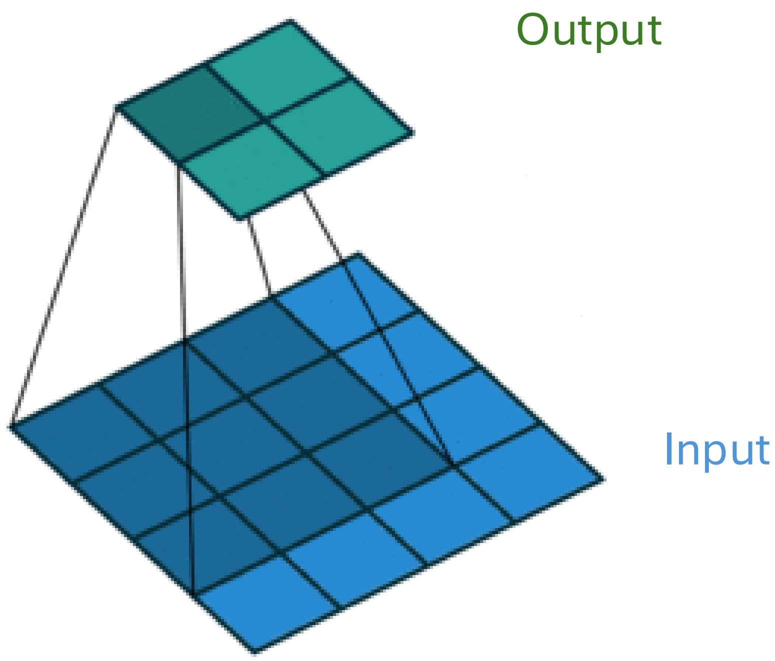
Stride

- We don’t just have to slide the filter by one pixel every time
- The distance we slide a filter by is called ***stride***
    - All the examples we’ve seen thus far have been stride = 1



Stride

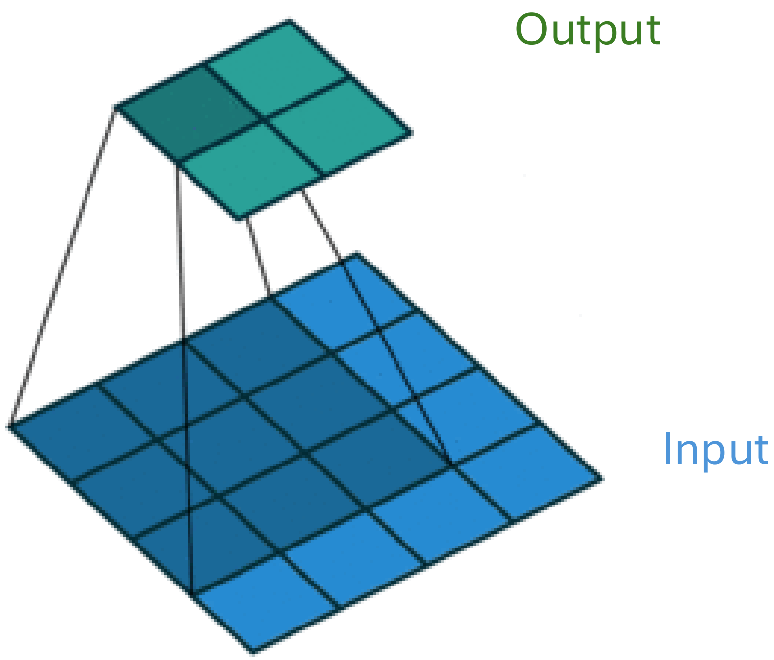
Stride=1



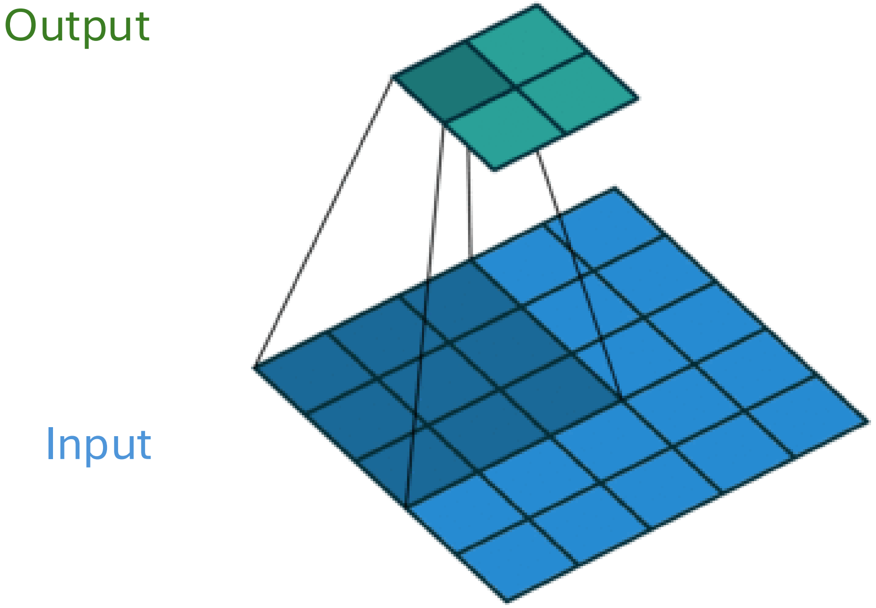
Accessed: https://hannibunny.github.io/mlbook/neuralnetworks/convolutionDemos.html

Stride

Stride=1



Stride=2

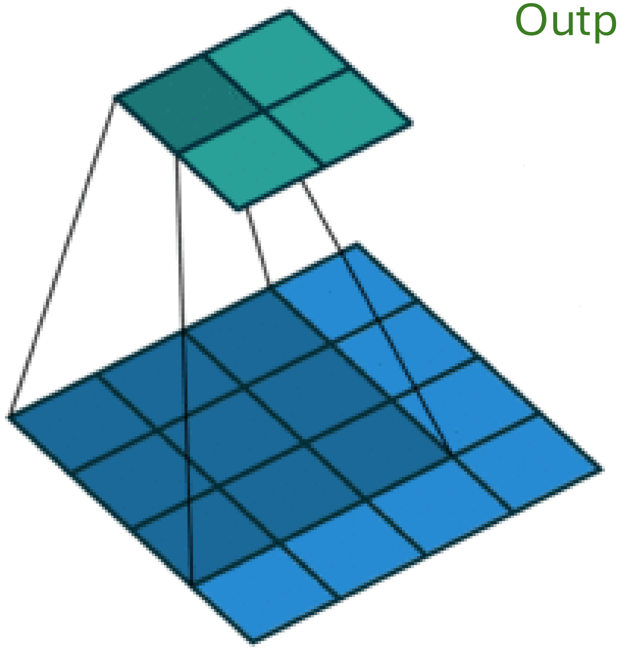


Accessed: https://hannibunny.github.io/mlbook/neuralnetworks/convolutionDemos.html

Stride

Stride=1

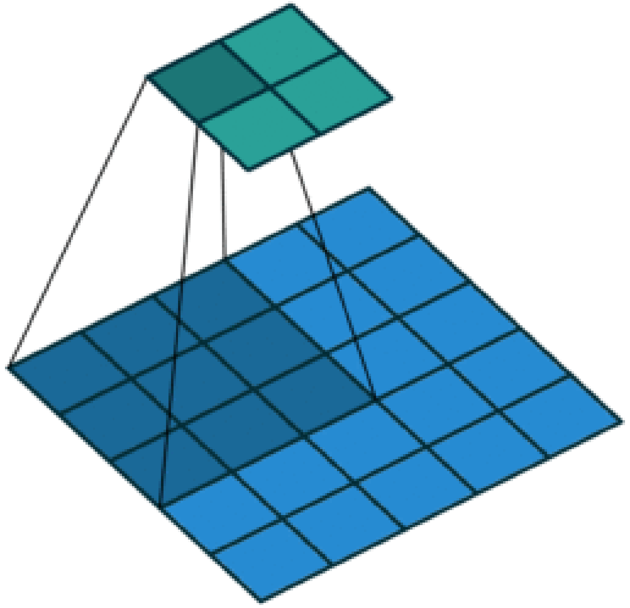
Output



Input

Stride=2

Output



Input

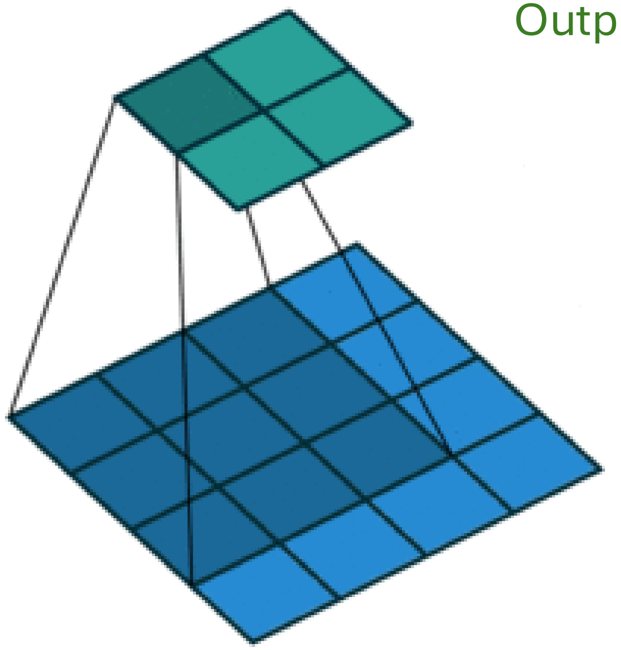
Any connection between input and output size?

Accessed: https://hannibunny.github.io/mlbook/neuralnetworks/convolutionDemos.html

Stride

Stride=1

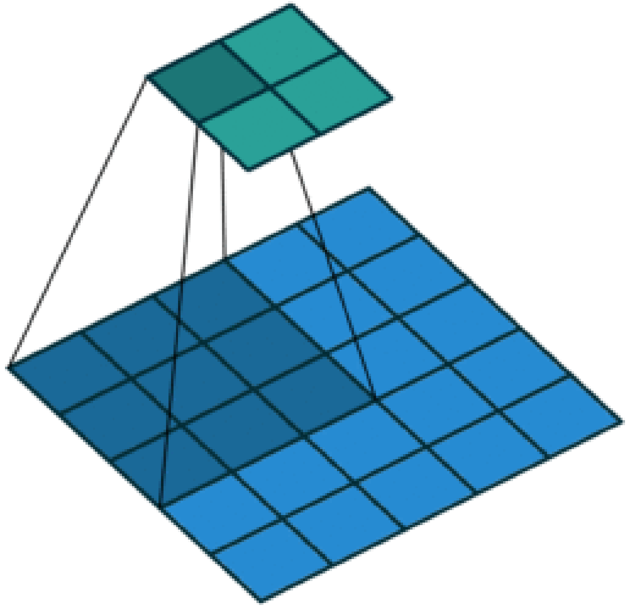
Output



Input

Stride=2

Output



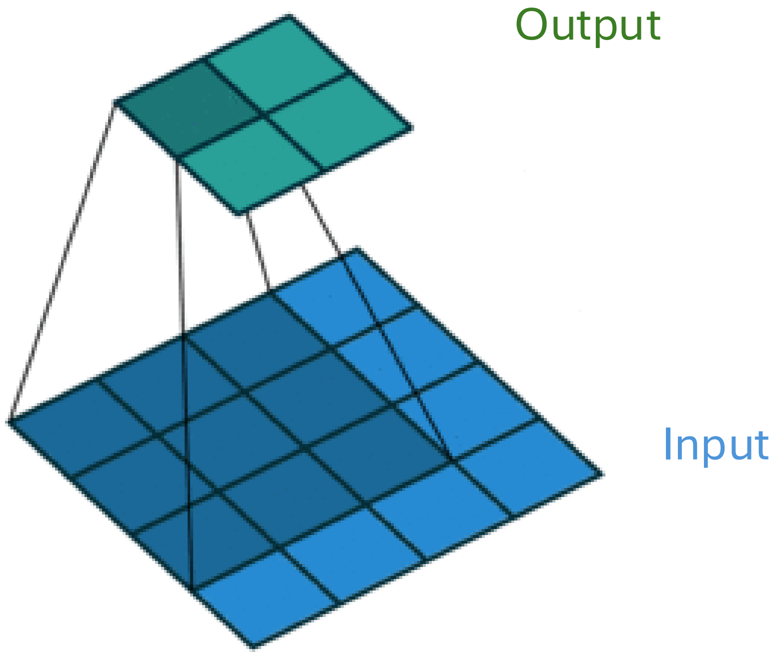
Input

Larger stride turns **larger** input into **same** sized output

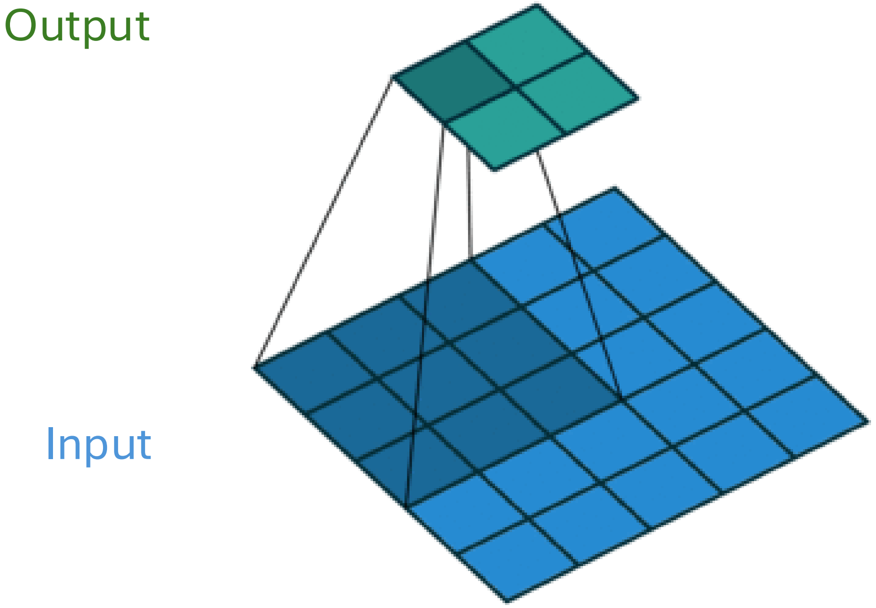
Accessed: https://hannibunny.github.io/mlbook/neuralnetworks/convolutionDemos.html

Stride

Stride=1



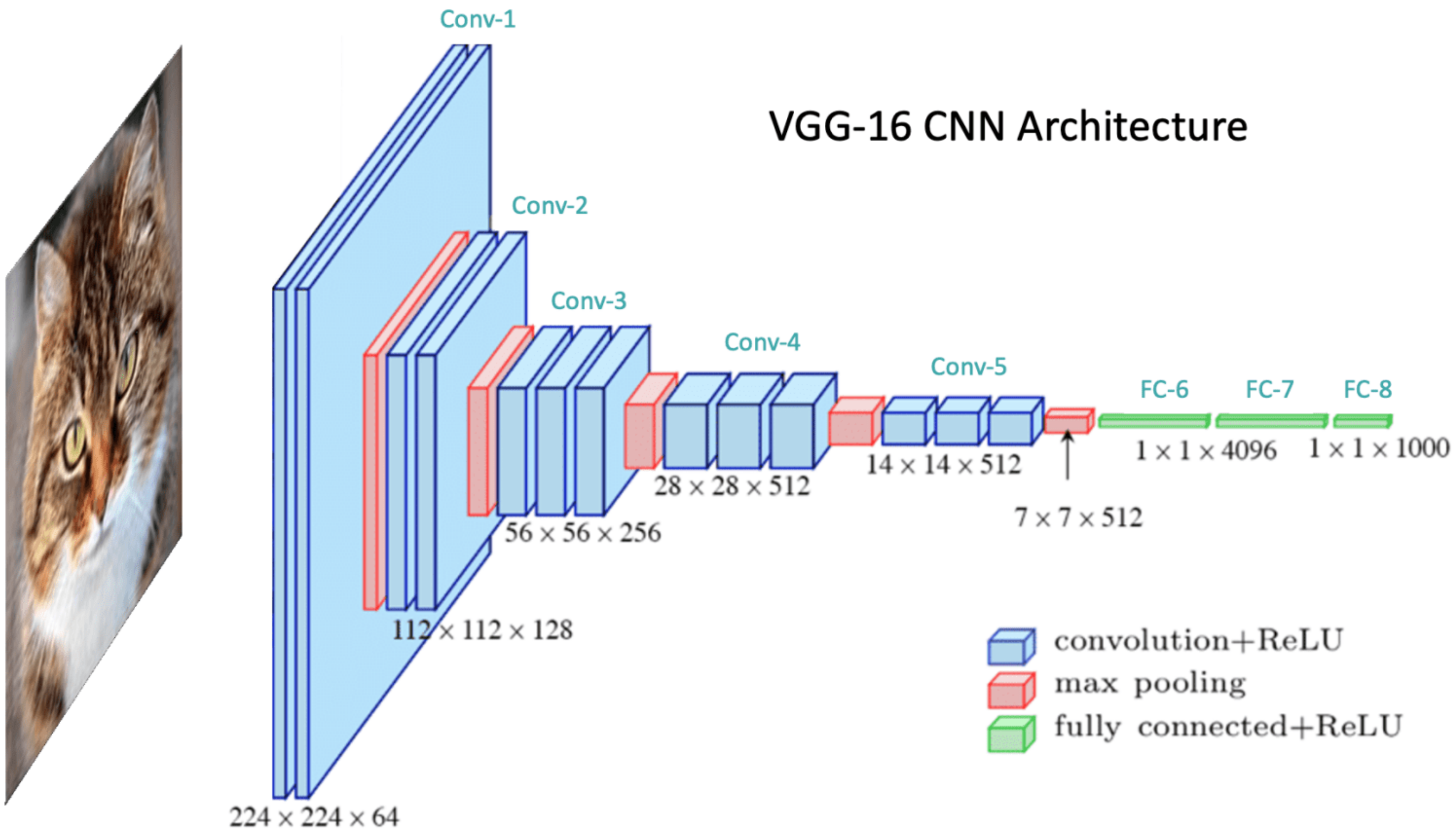
Stride=2



Larger stride turns **same** sized input into **smaller** sized output

Accessed: https://hannibunny.github.io/mlbook/neuralnetworks/convolutionDemos.html

Convolutional Neural Networks

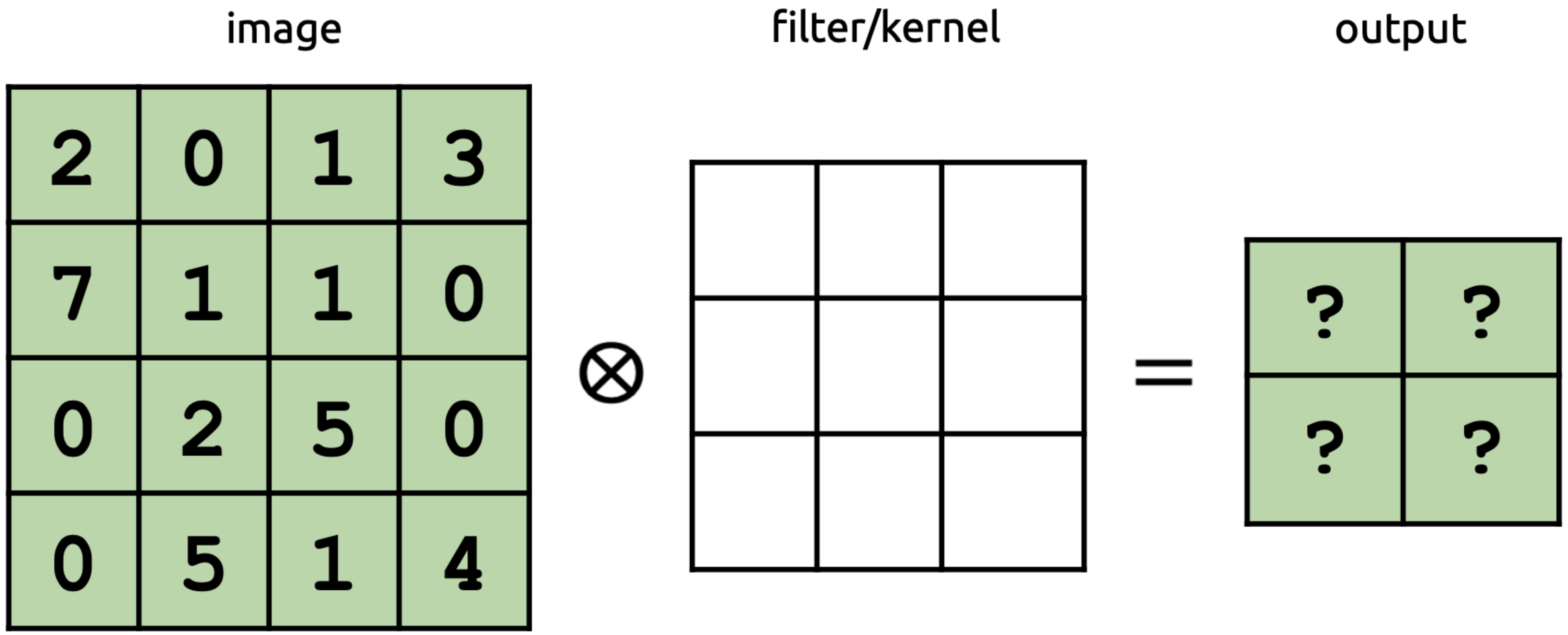


https://learnopencv.com/understanding-convolutional-neural-networks-cnn/

Key Idea 1: Filters are ***Learnable***

$$\underset{\text{image}}{\begin{bmatrix} 2 & 0 & 1 & 3 \\ 7 & 1 & 1 & 0 \\ 0 & 2 & 5 & 0 \\ 0 & 5 & 1 & 4 \end{bmatrix}} \otimes \underset{\text{filter/kernel}}{\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}} = \underset{\text{output}}{\begin{bmatrix} -4 & -3 \\ 2 & -9 \end{bmatrix}}$$

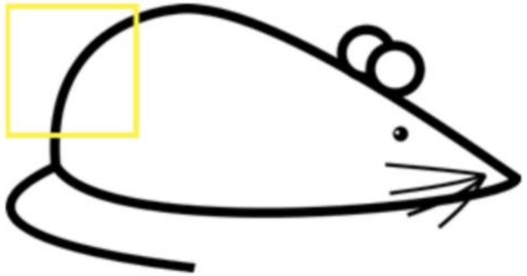
Key Idea 1: Filters are ***Learnable***



$k_{i,j}$ are learnable parameters

Key Idea 1: Filters are ***Learnable***

Original image

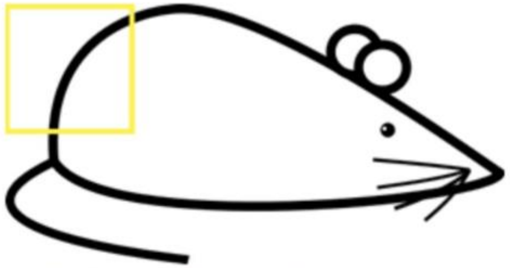


Visualization of the filter on the image

Label="Mouse"

Detecting patterns using learned filters

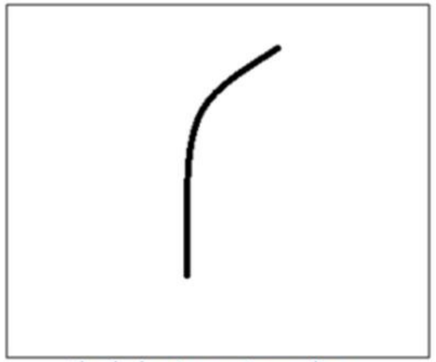
Original image



Visualization of the filter on the image

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter



Visualization of a curve detector filter

Visualization of the receptive field

0	0	0	0	0	0	30
0	0	0	0	50	50	50
0	0	0	20	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0

Pixel representation of the receptive field

*

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Multiplication and Summation = (50*30)+(50*30)+(50*30)+(20*30)+(50*30) = 6600 (A large number!)

Image courtesy: https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/

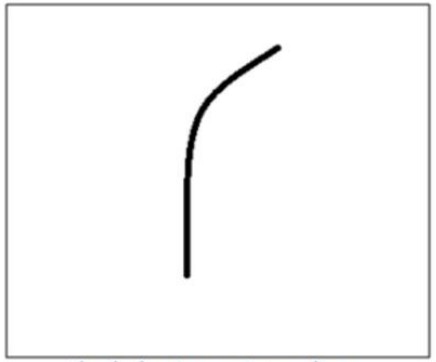
Detecting patterns using learned filters

Original image

How to detect other patterns?

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
[illegible]	[illegible]	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter



Visualization of a curve detector filter

Visualization of the filter on the image

0	0	0	0	0	0	0
0	40	0	0	0	0	0
40	0	40	0	0	0	0
40	20	0	0	0	0	0
0	50	0	0	0	0	0
0	0	50	0	0	0	0
25	25	0	50	0	0	0

Pixel representation of receptive field

*

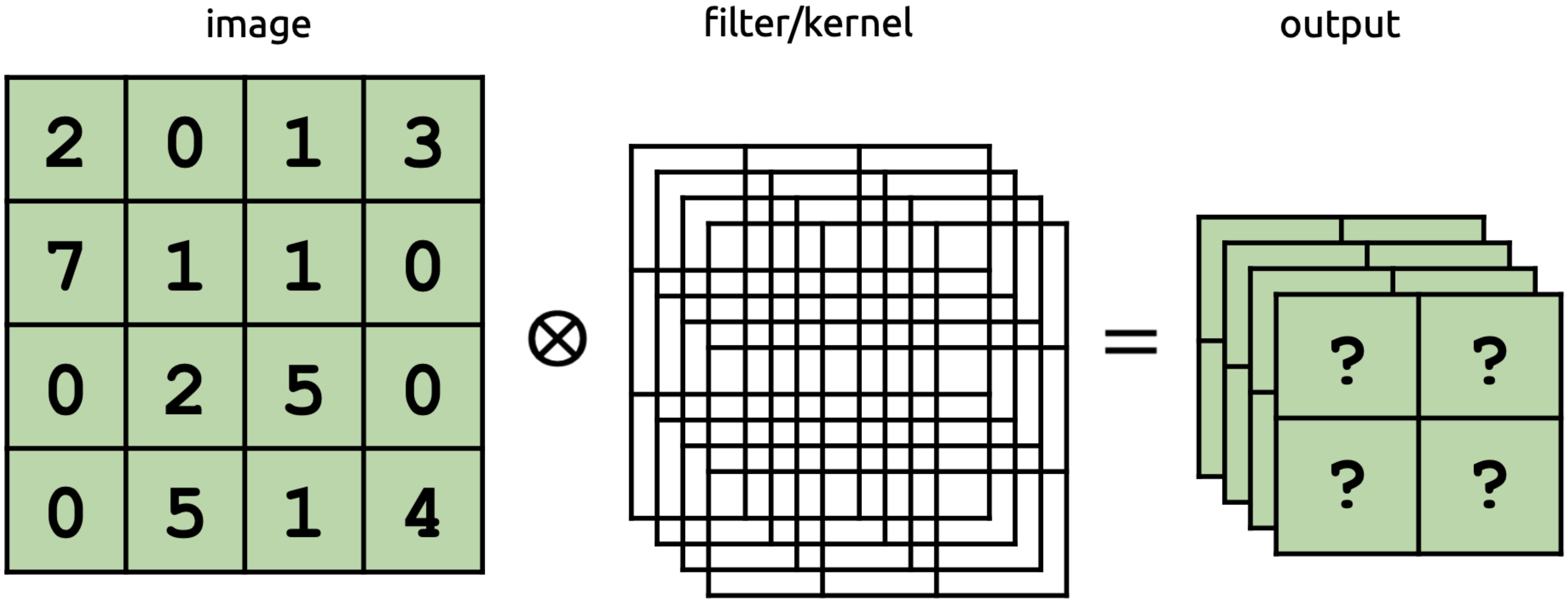
0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Multiplication and Summation = 0

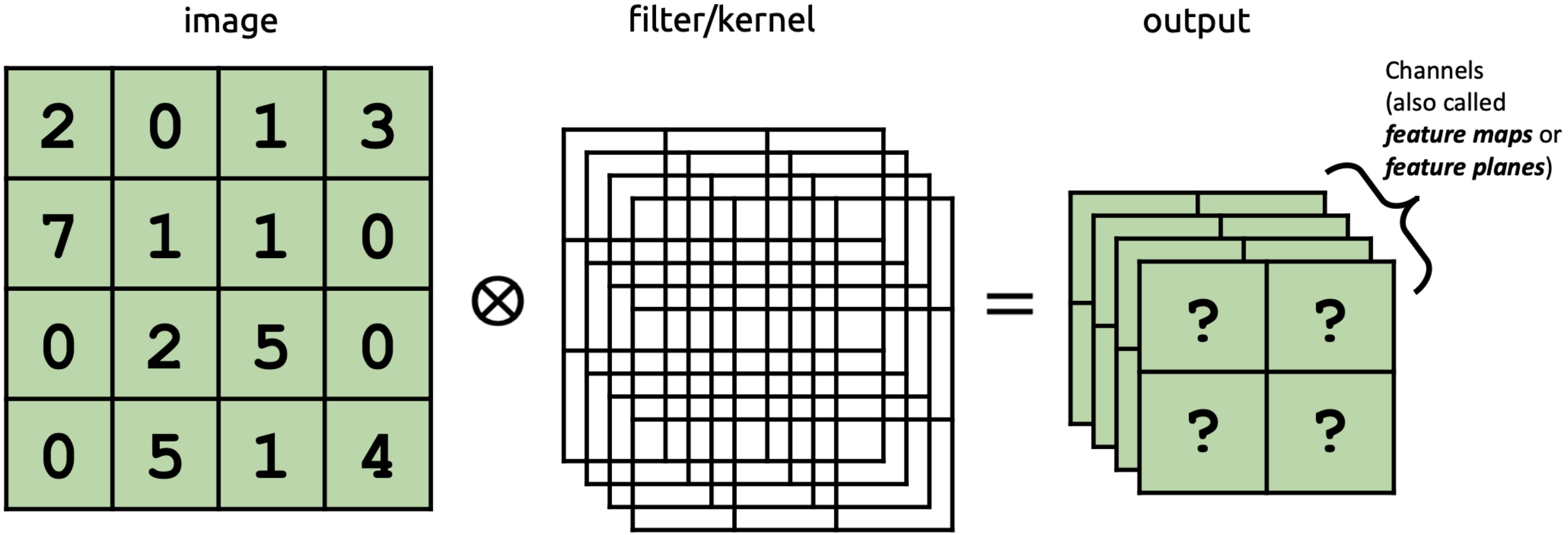
Image courtesy: https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/

Key Idea 2: Learn *many* filters



This block of filters is called a ***filter bank***

Key Idea 2: Learn ***many*** filters

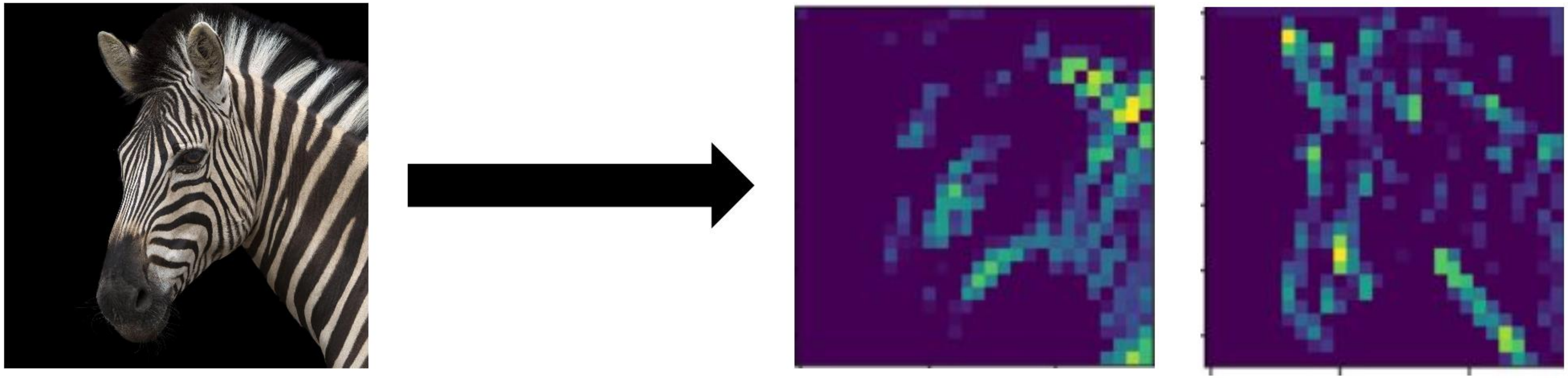


The output is now a <u>multi-channel</u> image

Any questions?

Key Idea 2: Learn ***many*** filters

- Why are multiple filters a good idea?
  - Can learn to extract different ***features*** of the image

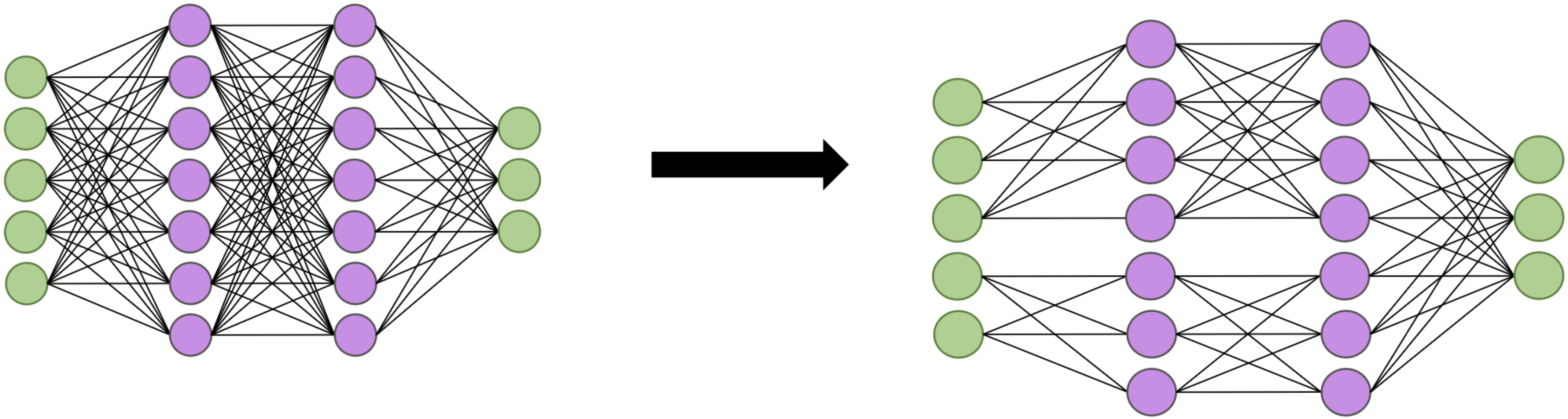


Input image | Output of filter 1 | Output of filter 2

How is convolution “partially connected?”

Fully Connected

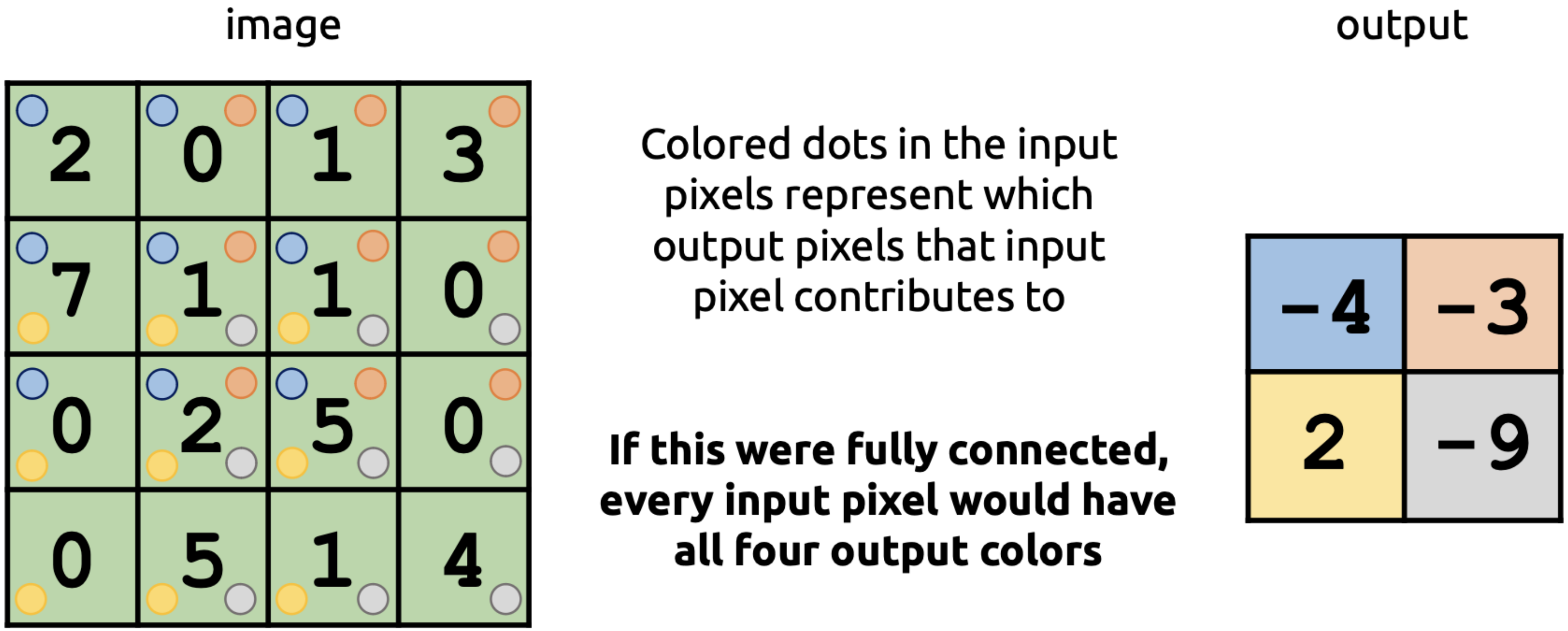
Partially Connected



Only certain input pixels are “connected” to certain output pixels

$$\underset{\text{image}}{\begin{bmatrix} 2 & 0 & 1 & 3 \\ 7 & 1 & 1 & 0 \\ 0 & 2 & 5 & 0 \\ 0 & 5 & 1 & 4 \end{bmatrix}} \otimes \underset{\text{filter/kernel}}{\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}} = \underset{\text{output}}{\begin{bmatrix} -4 & -3 \\ 2 & -9 \end{bmatrix}}$$

Only certain input pixels are "connected" to certain output pixels

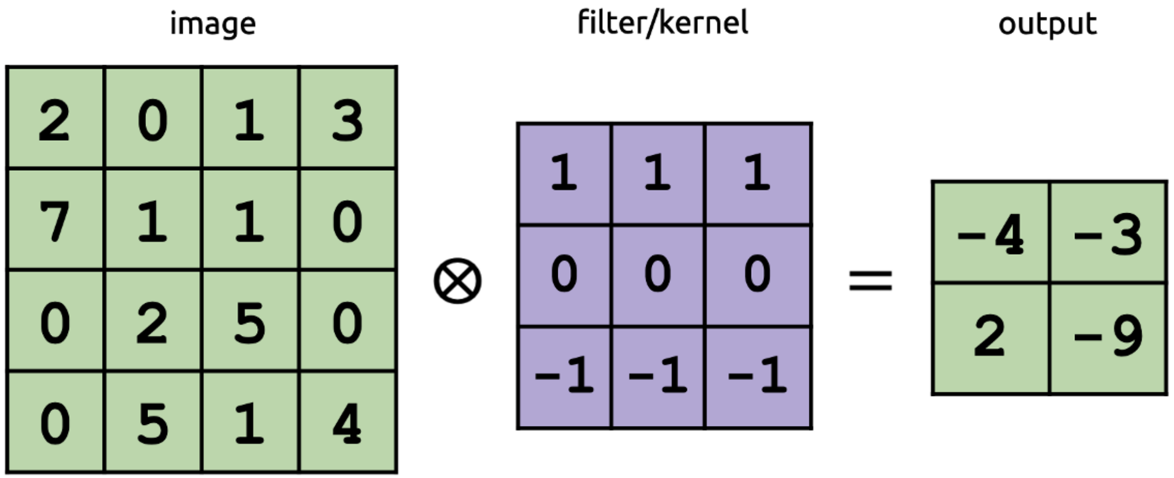


Colored dots in the input pixels represent which output pixels that input pixel contributes to

If this were fully connected, every input pixel would have all four output colors

Convolutions in Tensorflow

tf.nn.conv2d(input, filter, stride, padding)

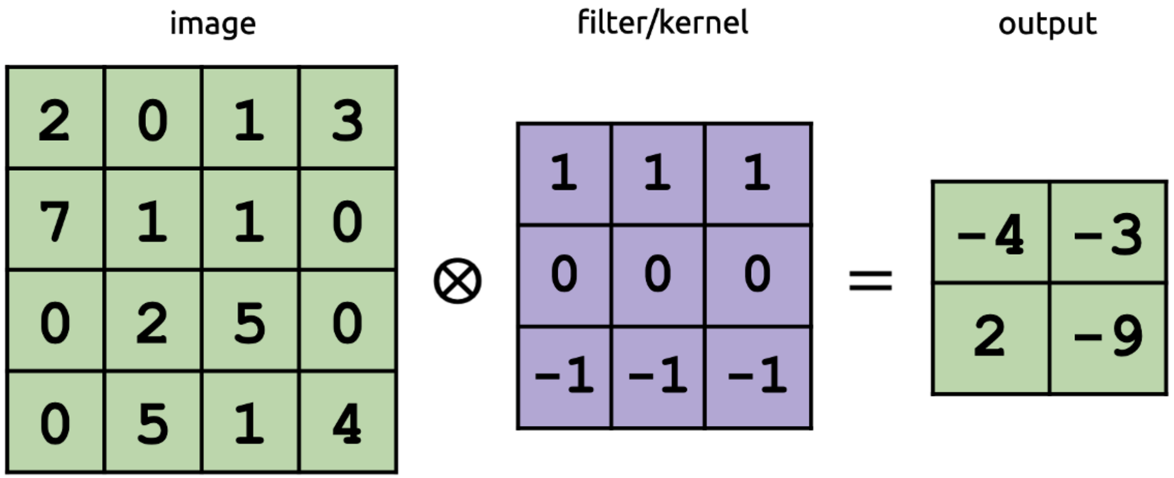


Documentation: https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/conv2d

Convolutions in Tensorflow

tf.nn.conv2d(input, filter, stride, padding)

Input "Image"   Kernel/filter



Documentation: https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/conv2d

Convolutions in Tensorflow

tf.nn.conv2d(input, filter, stride, padding)

Input “Image”

Kernel/filter

What is the shape of the input?
Tensor of:[# items in batch, width, height, # channels]

$$\overset{\text{image}}{\begin{bmatrix} 2 & 0 & 1 & 3 \\ 7 & 1 & 1 & 0 \\ 0 & 2 & 5 & 0 \\ 0 & 5 & 1 & 4 \end{bmatrix}} \otimes \overset{\text{filter/kernel}}{\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}} = \overset{\text{output}}{\begin{bmatrix} -4 & -3 \\ 2 & -9 \end{bmatrix}}$$

Documentation: https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/conv2d

Convolutions in Tensorflow

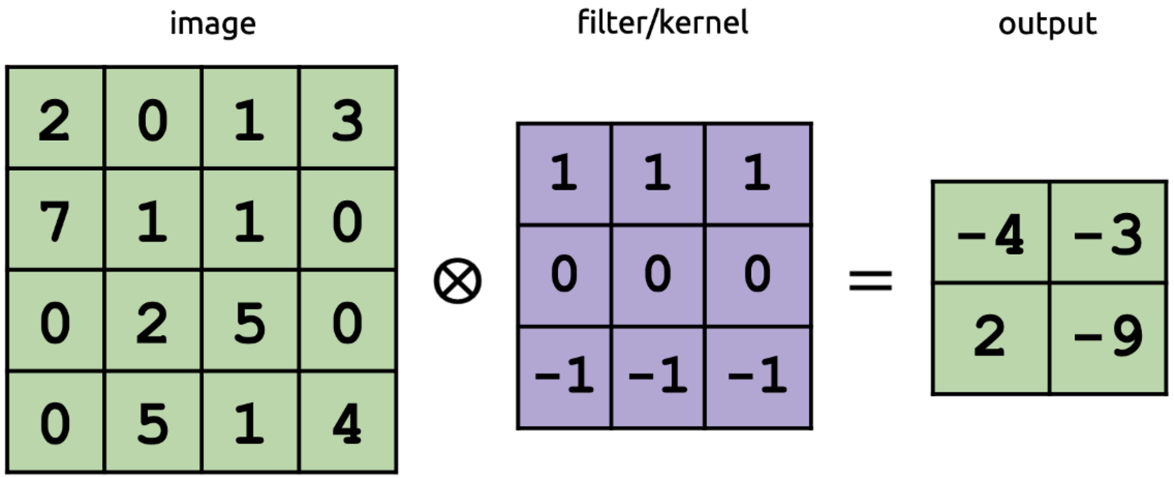
tf.nn.conv2d(input, filter, stride, padding)

Input "Image"

Kernel/filter

What is the shape of the input?
Tensor of:[# items in batch, width, height, # channels]

Channels: Number of input "colors"



Documentation: https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/conv2d

Convolutions in Tensorflow

tf.nn.conv2d(input, filter, stride, padding)

Input “Image”

Kernel/filter

What is the shape of the input?
Tensor of:[# items in batch, width, height, # channels]

Channels: Number of input “colors”

How can we determine the output size of a convolution?

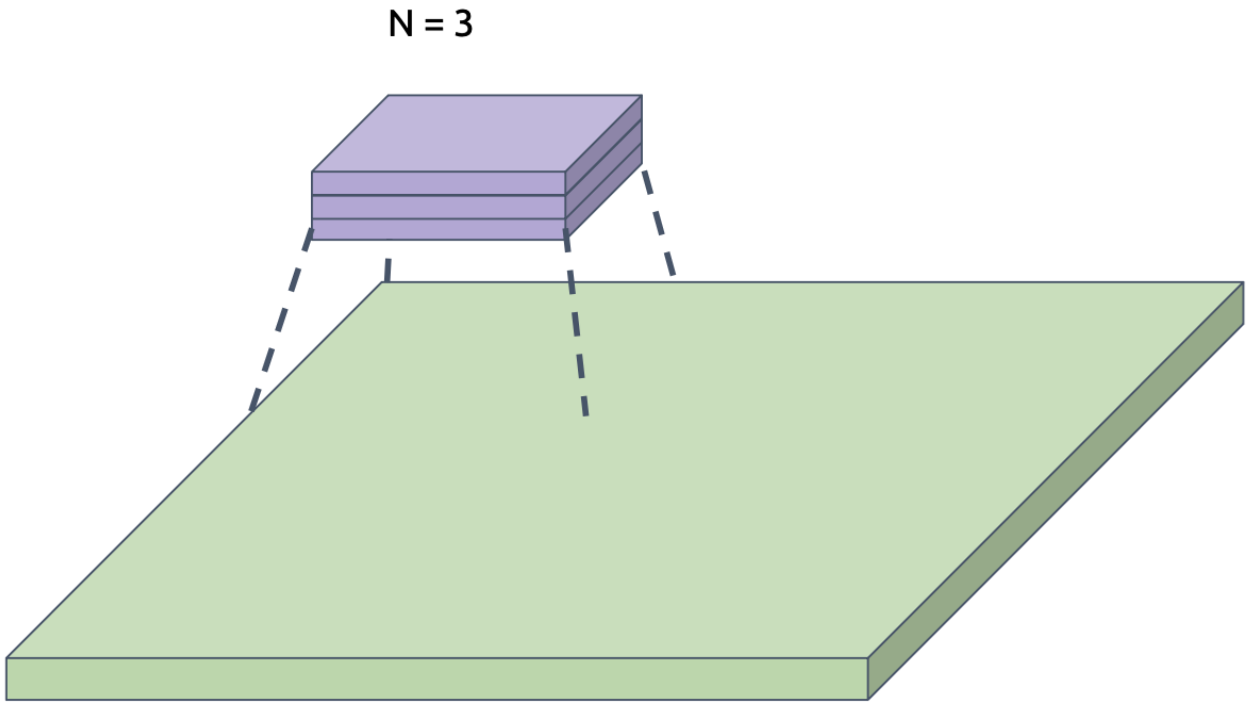
$$\underset{\text{image}}{\begin{bmatrix} 2 & 0 & 1 & 3 \\ 7 & 1 & 1 & 0 \\ 0 & 2 & 5 & 0 \\ 0 & 5 & 1 & 4 \end{bmatrix}} \otimes \underset{\text{filter/kernel}}{\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}} = \underset{\text{output}}{\begin{bmatrix} -4 & -3 \\ 2 & -9 \end{bmatrix}}$$

Documentation: https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/conv2d

Output Size of a Convolution Layer

The output size of a convolution layer depends on 4 Hyperparameters:

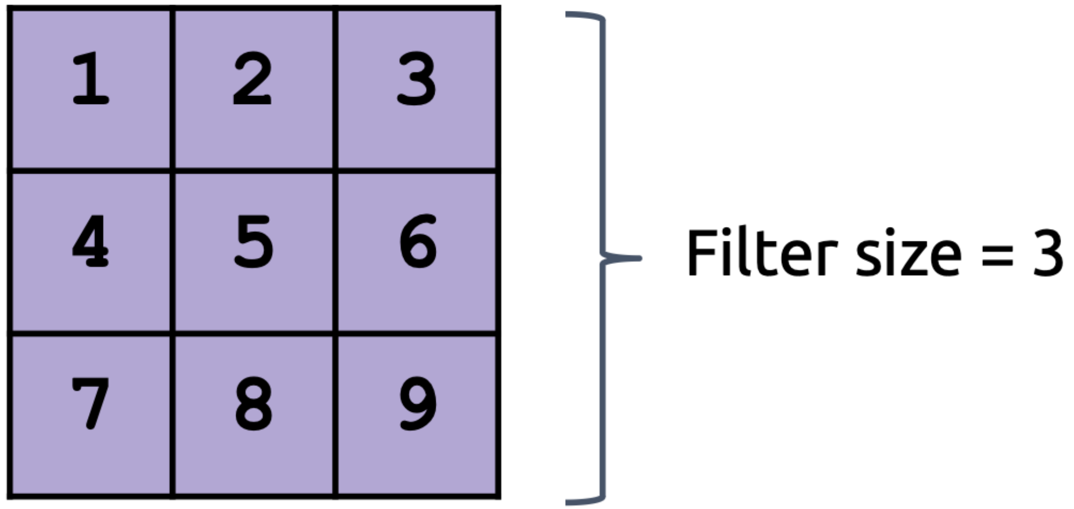
- Number of filters, **N**



Output Size of a Convolution Layer

The output size of a convolution layer depends on 4 Hyperparameters:

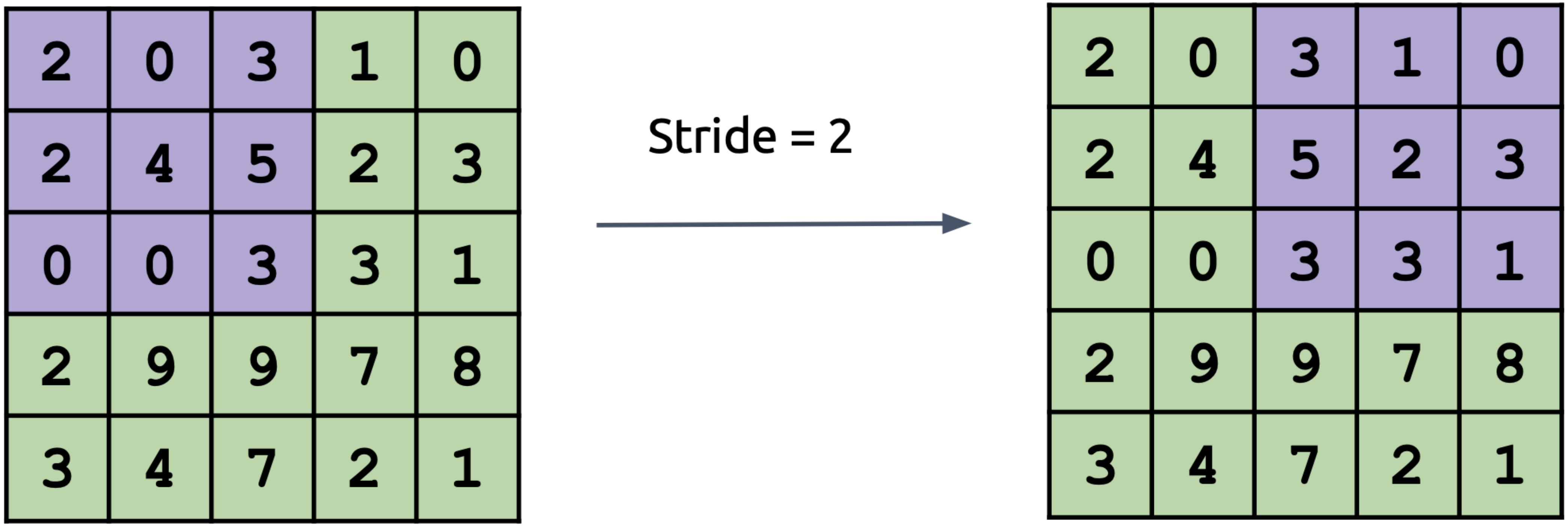
- Number of filters, **N**
- The size of these filters, **F**



Output Size of a Convolution Layer

The output size of a convolution layer depends on 4 Hyperparameters:

- Number of filters, **N**
- The size of these filters, **F**
- The stride, **S**



“Problem” With Convolution

$$\begin{bmatrix} 2 & 0 & 1 & 3 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 0 & -1 \end{bmatrix}$$

- Output of convolution is always smaller than the input
- Why might we want the output size to be the same?
  - To avoid the filter “eating at the border” of the image when applying multiple conv layers

Solution: Padding

Apply the kernel to ‘imaginary’ pixels surrounding the image

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

Solution: Padding

Apply the kernel to ‘imaginary’ pixels surrounding the image

?	?	?	?	?	?	?
?	2	0	3	1	1	?
?	1	1	0	0	2	?
?	4	3	2	0	1	?
?	1	0	5	2	0	?
?	0	1	0	3	0	?
?	?	?	?	?	?	?

What Values to Use For These Pixels?

?	?	?	?	?	?	?
?	2	0	3	1	1	?
?	1	1	0	0	2	?
?	4	3	2	0	1	?
?	1	0	5	2	0	?
?	0	1	0	3	0	?
?	?	?	?	?	?	?

What Values to Use For These Pixels?

Standard practice: fill with zeroes

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

What Values to Use For These Pixels?

Standard practice: fill with zeroes

- Zero-valued padding pixels just result in some terms in the convolution sum being zero

$$V(x,y) = (I \otimes K)(x,y) = \sum_m \sum_n I(x+m, y+n) K(m,n)$$

This is zero for a padding pixel

- End result: equivalent to a applying a 'masked' version of the filter that only covers the valid pixels

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

Padding Modes in Tensorflow

2 available options: ‘VALID’ and ‘SAME’:

Valid

Filter only slides over “Valid” regions of the data

2	0	1	3
0	1	1	0
0	0	2	0
0	1	1	1

Same

Filter slides over the bounds of the data, ensuring output size is the “Same” as input size (when stride = 1)

0	0	0	0	0	0
0	2	0	1	3	0
0	1	1	2	3	0
0	4	3	2	1	0
0	8	3	1	3	0
0	0	0	0	0	0

VALID Padding in Tensorflow

```
tf.nn.conv2d(input, filter, strides,
padding='VALID')
```

2	0	1	3
0	1	1	0
0	0	2	0
0	1	1	1

VALID Padding in Tensorflow

```
tf.nn.conv2d(input, filter, strides,
padding='VALID')
```

2	0	1	3
0	1	1	0
0	0	2	0
0	1	1	1

VALID Padding in Tensorflow

```
tf.nn.conv2d(input, filter, strides,
padding='VALID')
```

2	0	1	3
0	1	1	0
0	0	2	0
0	1	1	1

VALID Padding in Tensorflow

```
tf.nn.conv2d(input, filter, strides,
             padding='VALID')
```

2	0	1	3
0	1	1	0
0	0	2	0
0	1	1	1

We already tried this! (reduced output size)

$$\begin{bmatrix} 2 & 0 & 3 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 2 & 0 \\ 1 & 0 & 1 & 2 \end{bmatrix} \underset{\substack{\text{“VALID”} \\ \text{Stride = 1}}}{\otimes} \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & -1 \end{bmatrix}$$

SAME Padding in Tensorflow

```
tf.nn.conv2d(input, filter, strides,
             padding='SAME')
```

0	0	0	0	0	0
0	2	0	1	3	0
0	1	1	2	3	0
0	4	3	2	1	0
0	8	3	1	3	0
0	0	0	0	0	0

SAME Padding in Tensorflow

```
tf.nn.conv2d(input, filter, strides,
         padding='SAME')
```

0	0	0	0	0	0
0	2	0	1	3	0
0	1	1	2	3	0
0	4	3	2	1	0
0	8	3	1	3	0
0	0	0	0	0	0

SAME Padding in Tensorflow

```
tf.nn.conv2d(input, filter, strides,
padding='SAME')
```

0	0	0	0	0	0
0	2	0	1	3	0
0	1	1	2	3	0
0	4	3	2	1	0
0	8	3	1	3	0
0	0	0	0	0	0

SAME Padding in Tensorflow

```
tf.nn.conv2d(input, filter, strides,
        padding='SAME')
```

0	0	0	0	0	0
0	2	0	1	3	0
0	1	1	2	3	0
0	4	3	2	1	0
0	8	3	1	3	0
0	0	0	0	0	0

Output Size of a Convolution Layer

The output size of a convolution layer depends on 4 Hyperparameters:

- Number of filters, **N**
- The size of these filters, **F**
- The stride, **S**
- The amount of padding, **P**

0	0	0	0	0	0
0	0	0	0	0	0
0	0	2	3	0	0
0	0	9	2	0	0
0	0	0	0	0	0
0	0	0	0	0	0

Padding = 2

Output Size of a Convolution Layer

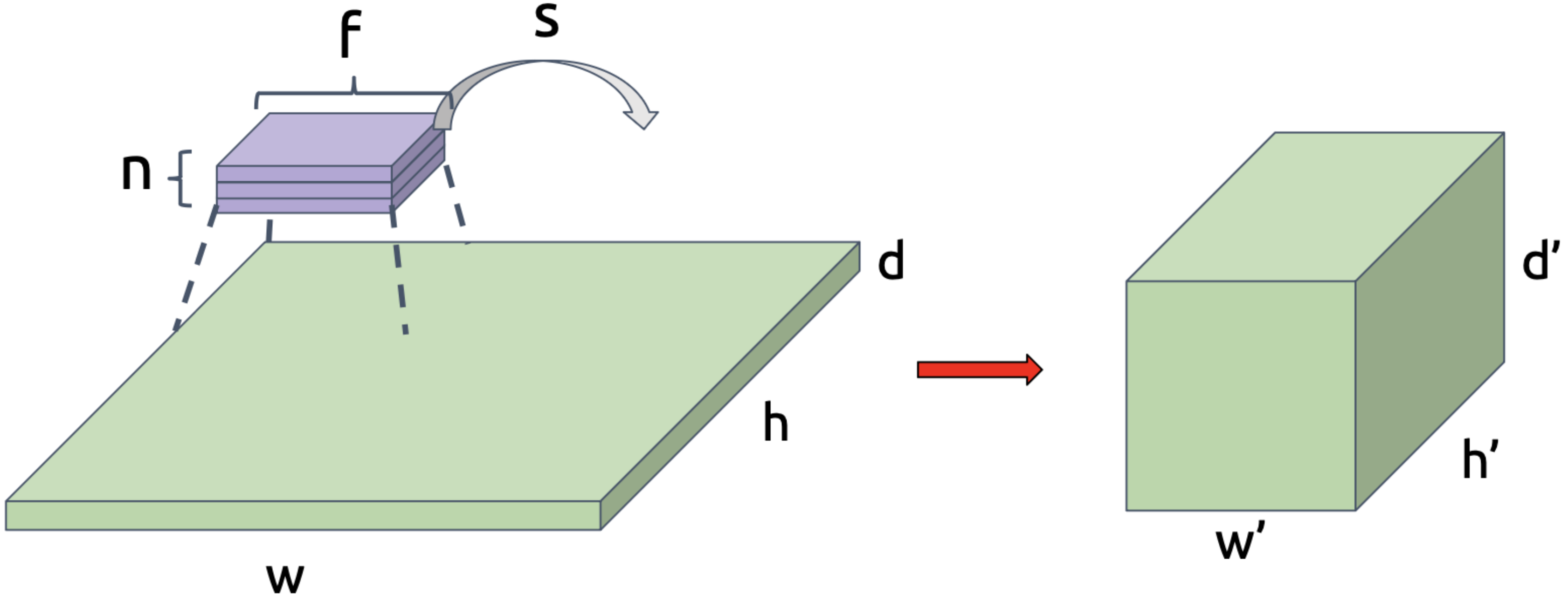
Suppose we know the number of filters, their size, the stride, and padding (**n,f,s,p**).

Then for a convolution layer with input dimension **w x h x d**, the output dimensions **w’ x h’ x d’** are:

$$w' = \frac{w\ - f + 2p}{s} + 1$$

$$h' = \frac{h\ - f + 2p}{s} + 1$$

$$d' = n$$



Output Size for “VALID” Padding

$$w' = \frac{w \ - f + 2p}{s} + 1$$

Let $w = 4$

num filters n = 1
filter size f = 3
stride s = 1
padding p = 0

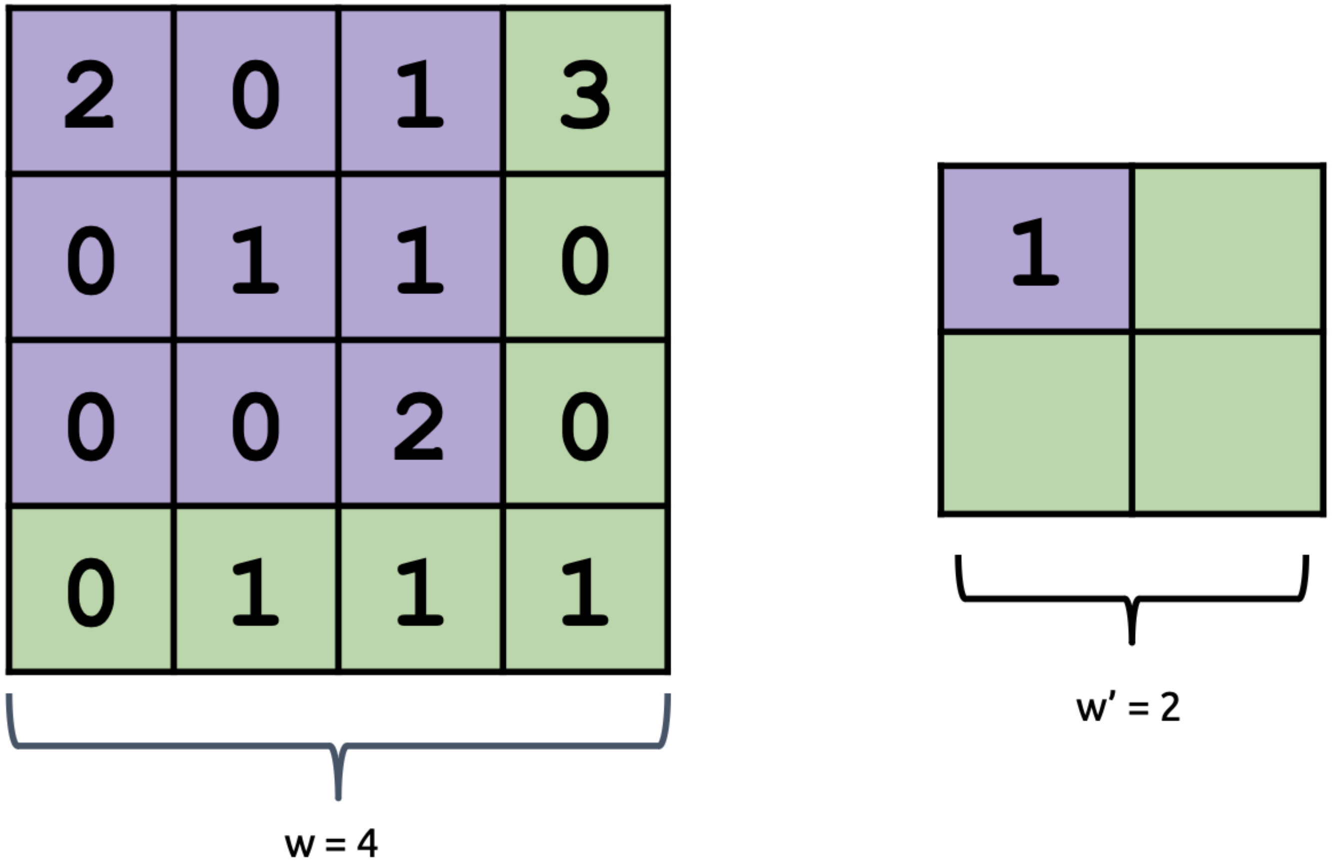
$$w' = \frac{4 \ - 3 + 2 \cdot 0}{1} + 1$$

$$= 1 + 1 = 2$$

Output Size for “VALID” Padding

$$w' = \frac{w - f + 2p}{s} + 1$$

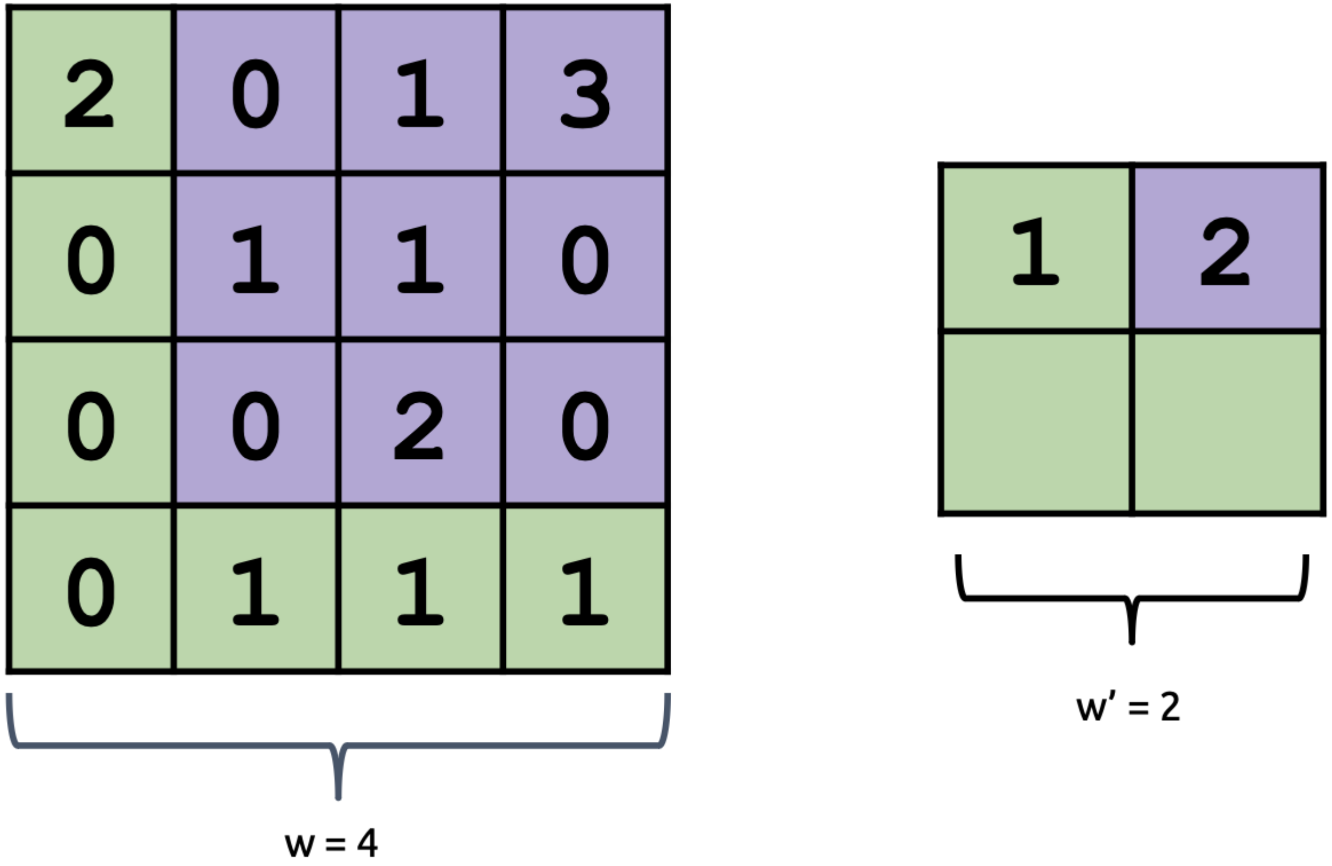
num filters n = 1
filter size f = 3
stride s = 1
padding p = 0



Output Size for “VALID” Padding

$$w' = \frac{w\ - f + 2p}{s} + 1$$

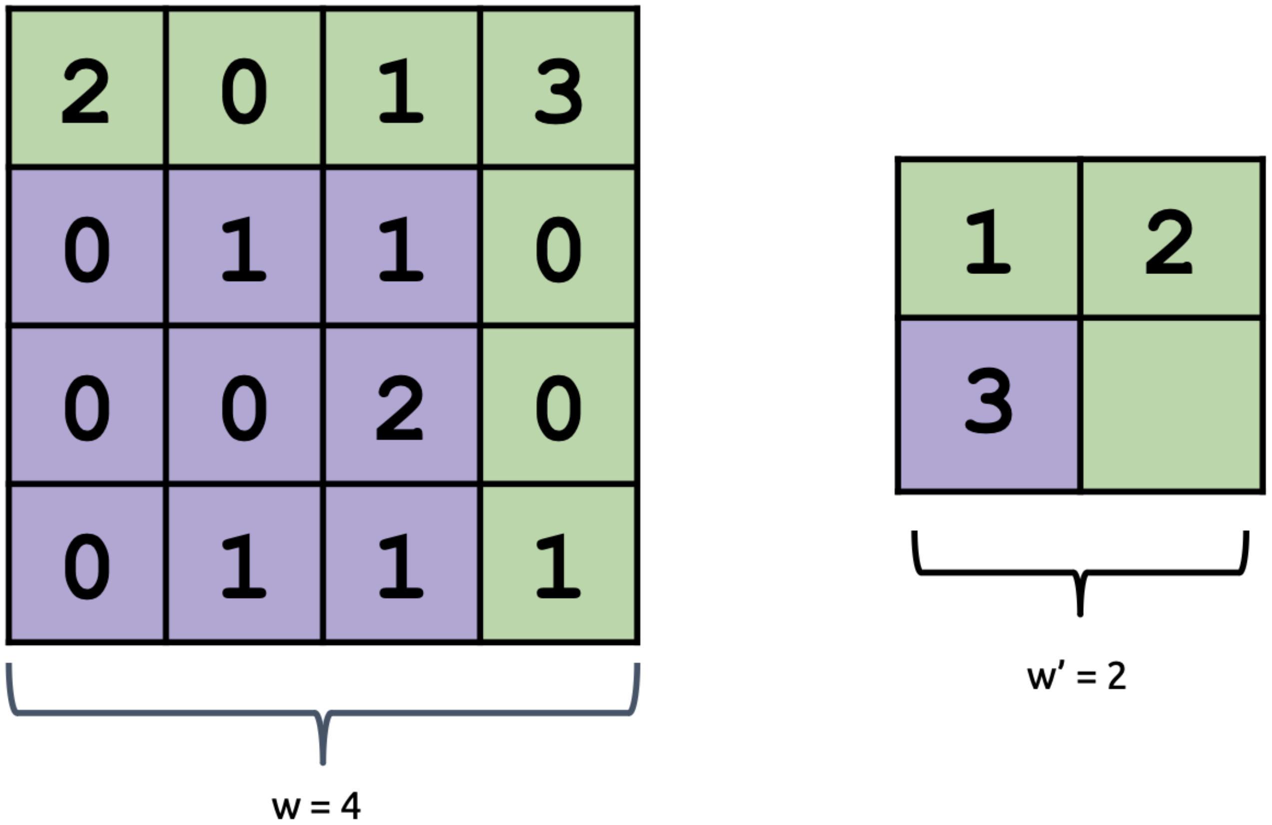
num filters n = 1
filter size f = 3
stride s = 1
padding p = 0



Output Size for "VALID" Padding

$$w' = \frac{w - f + 2p}{s} + 1$$

num filters n = 1
filter size f = 3
stride s = 1
padding p = 0



Output Size for “VALID” Padding

$$w' = \frac{w \ - f + 2p}{s} + 1$$

num filters n = 1
filter size f = 3
stride s = 1
padding p = 0

2	0	1	3
0	1	1	0
0	0	2	0
0	1	1	1

w = 4

1	2
3	4

w’ = 2

Output Size for "SAME" Padding

$$w' = \frac{w \; - f + 2p}{s} + 1$$

num filters n = 1
filter size f = 3
stride s = 1
padding p = 1*

Padding size needs to be determined

Let $w = 4$

$$w' = \frac{4 \; - 3 + 2 \cdot 1}{1} + 1$$

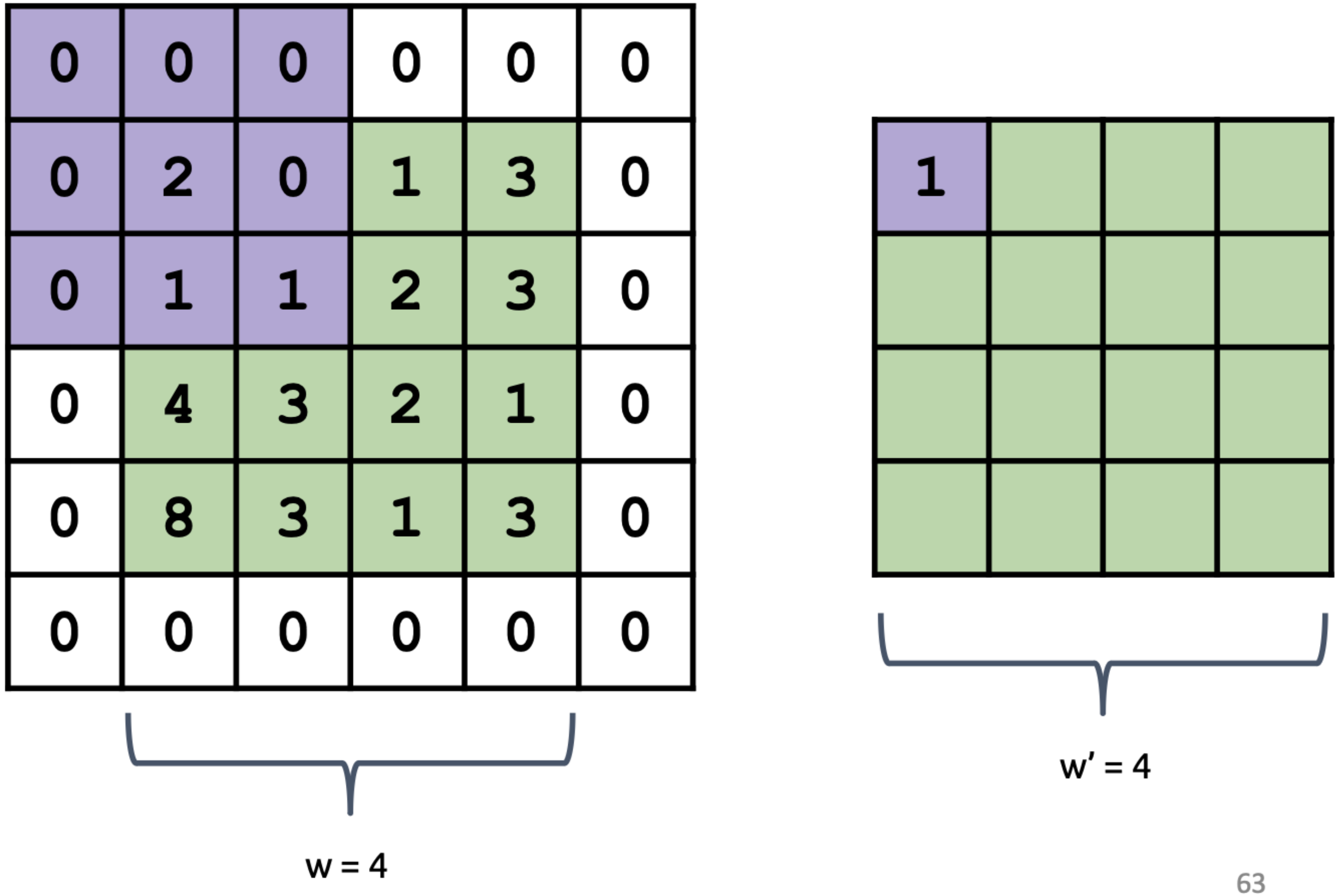
$$= 3 + 1 = 4$$

Output Size for "SAME" Padding

$$w' = \frac{w \; - f + 2p}{s} + 1$$

num filters n = 1
filter size f = 3
stride s = 1
padding p = 1*

Padding size needs to be determined



Output Size for "SAME" Padding

$$w' = \frac{w \; - f + 2p}{s} + 1$$

num filters n = 1
filter size f = 3
stride s = 1
padding p = 1*

Padding size needs to be determined

0	0	0	0	0	0
0	2	0	1	3	0
0	1	1	2	3	0
0	4	3	2	1	0
0	8	3	1	3	0
0	0	0	0	0	0

w = 4

1	2		

w′ = 4

Output Size for "SAME" Padding

$$w' = \frac{w \; - f + 2p}{s} + 1$$

num filters n = 1
filter size f = 3
stride s = 1
padding p = 1*

Padding size needs to be determined

0	0	0	0	0	0
0	2	0	1	3	0
0	1	1	2	3	0
0	4	3	2	1	0
0	8	3	1	3	0
0	0	0	0	0	0

w = 4

1	2	3	

w' = 4

Any questions?

Output Size for “SAME” Padding

$$w' = \frac{w \;- f + 2p}{s} + 1$$

num filters n = 1
filter size f = 3
stride s = 1
padding p = 1*

Padding size needs to be determined

0	0	0	0	0	0
0	2	0	1	3	0
0	1	1	2	3	0
0	4	3	2	1	0
0	8	3	1	3	0
0	0	0	0	0	0

$$\underbrace{\hspace{6em}}_{w\,=\,4}$$

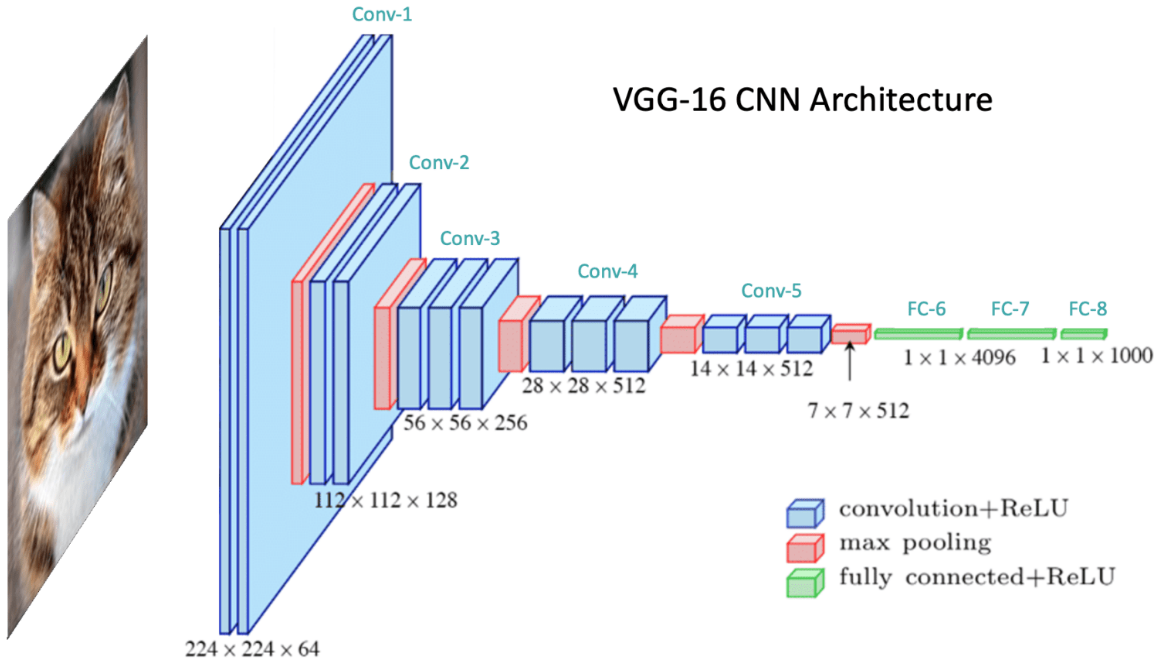
1	2	3	4

$$\underbrace{\hspace{6em}}_{w'\,=\,4}$$

Getting network output

Remaining Question: If the convolution creates another [h x w x d] tensor, how do we actually get an output?

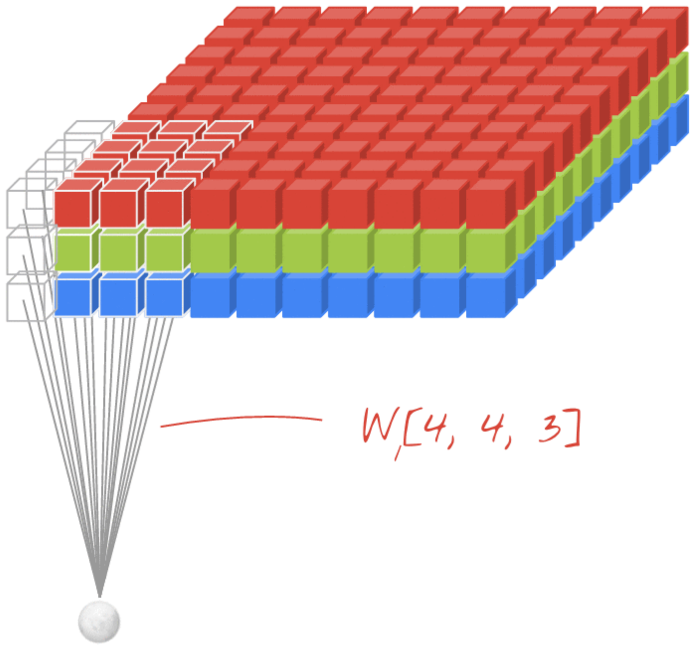
How can we turn use convolutions for classification?



https://learnopencv.com/understanding-convolutional-neural-networks-cnn/

Color images…

Remaining Question: What if our input has multiple channels (colors)? Do we apply filters to each individual color matrix? Or in some other way?



Source: Martin Görner