An Introduction to Neural Networks II: Convolutional Neural Networks

Eric Gossett

Recap

- Machine learning's focus is developing an algorithm that learns a task from a set of data.

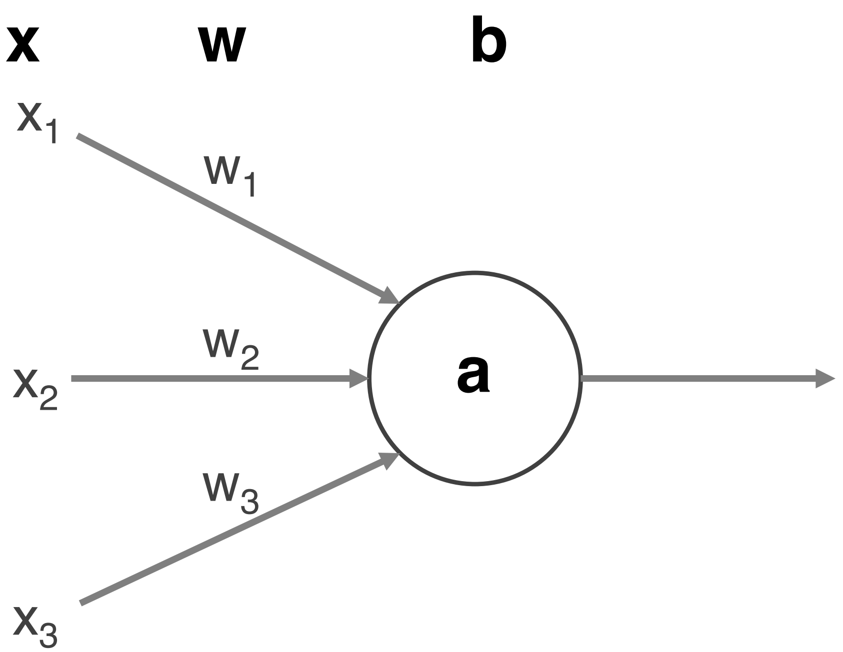
x, a set of **features**

y, associated **labels**

goal: predict **y** from **x**

Recap: Neural Networks

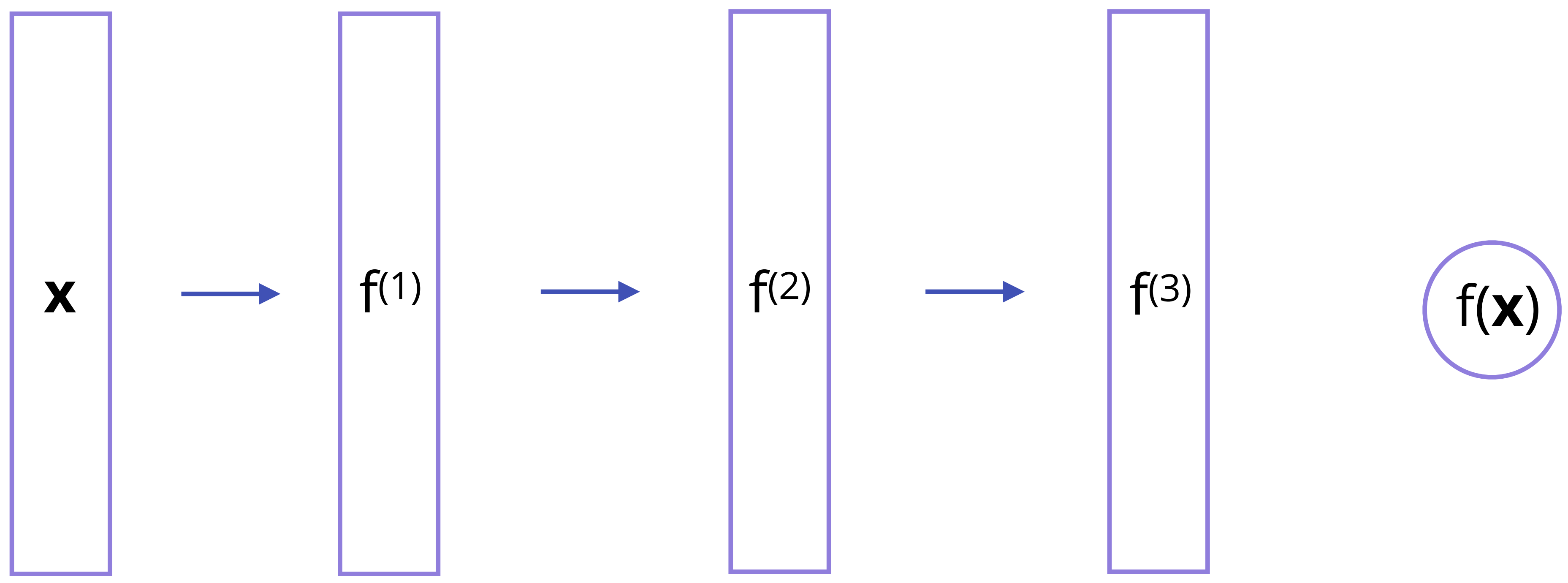
- Neural networks are comprised of layers. Each layer contains a number of **neurons.**
- **Neurons** consist of the following:



x: Input from previous layer
w: The weight (learned parameter)
b: The bias (learned parameter)
a: The activation function

$$\text{Output} = \mathbf{a}(\mathbf{w}^{(1)} \cdot \mathbf{x} + \mathbf{b}^{(1)})$$

Recap: Neural Networks



- neural network defines a mapping **y** = f(**x**; **w, b**) by learning the value of the parameters **w, b** that best approximate the function f* (the task function, e.g. classification).

- Given a set of training data our objective is to learn the best set of **weights** **(w)** and **biases** **(b)** that give the best prediction of **y**

-

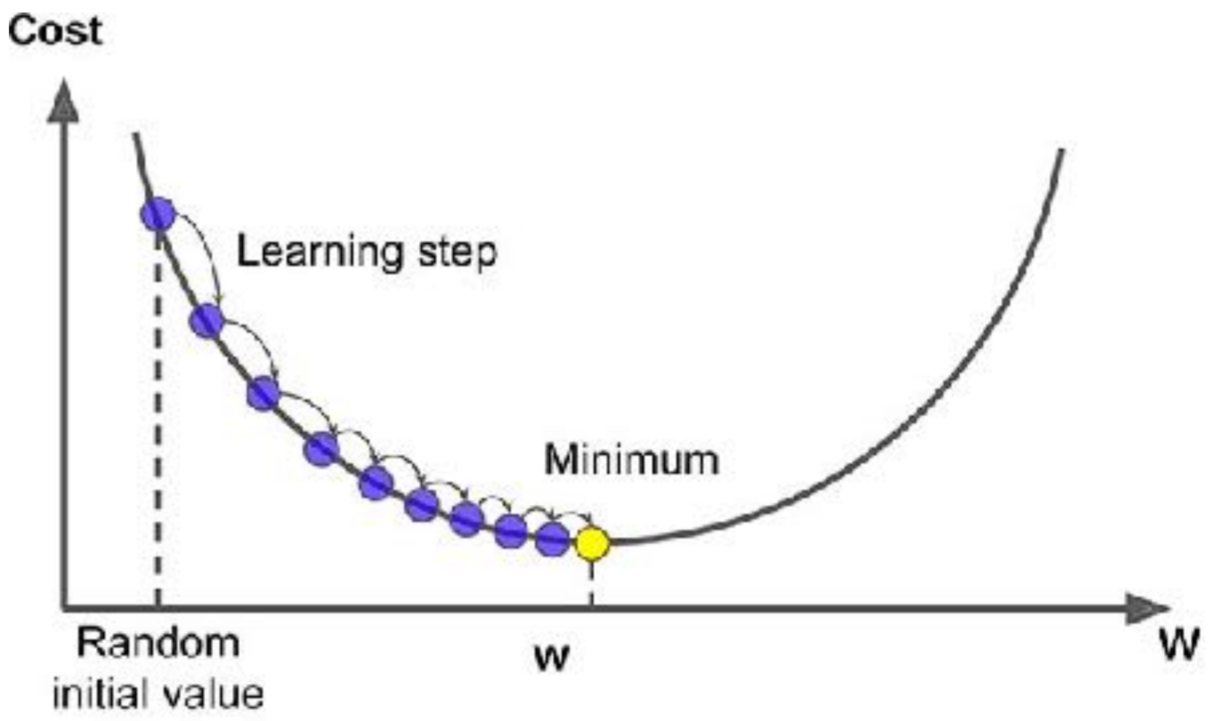
Recap: Neural Networks

- The network learns by determine the best parameters (**w** and **b**) that **minimize the error** (e.g. find the most accurate prediction)

- The error is determined by the cost function **C**

- Minimization of error is done using stochastic gradient descent.

- The weights and biases are updated via back propagation



Fully Connected NN and Image Classification

- Last time, a fully connected neural network (FCNN) was trained to classify images of handwritten digits (MNIST)

- However, for image classification a FCNN fails to take into account the spatial structure of an image.

- The network fails to account for how close or far each pixel is.

- **Goal:** Modify the network to take the spatial structure into account.

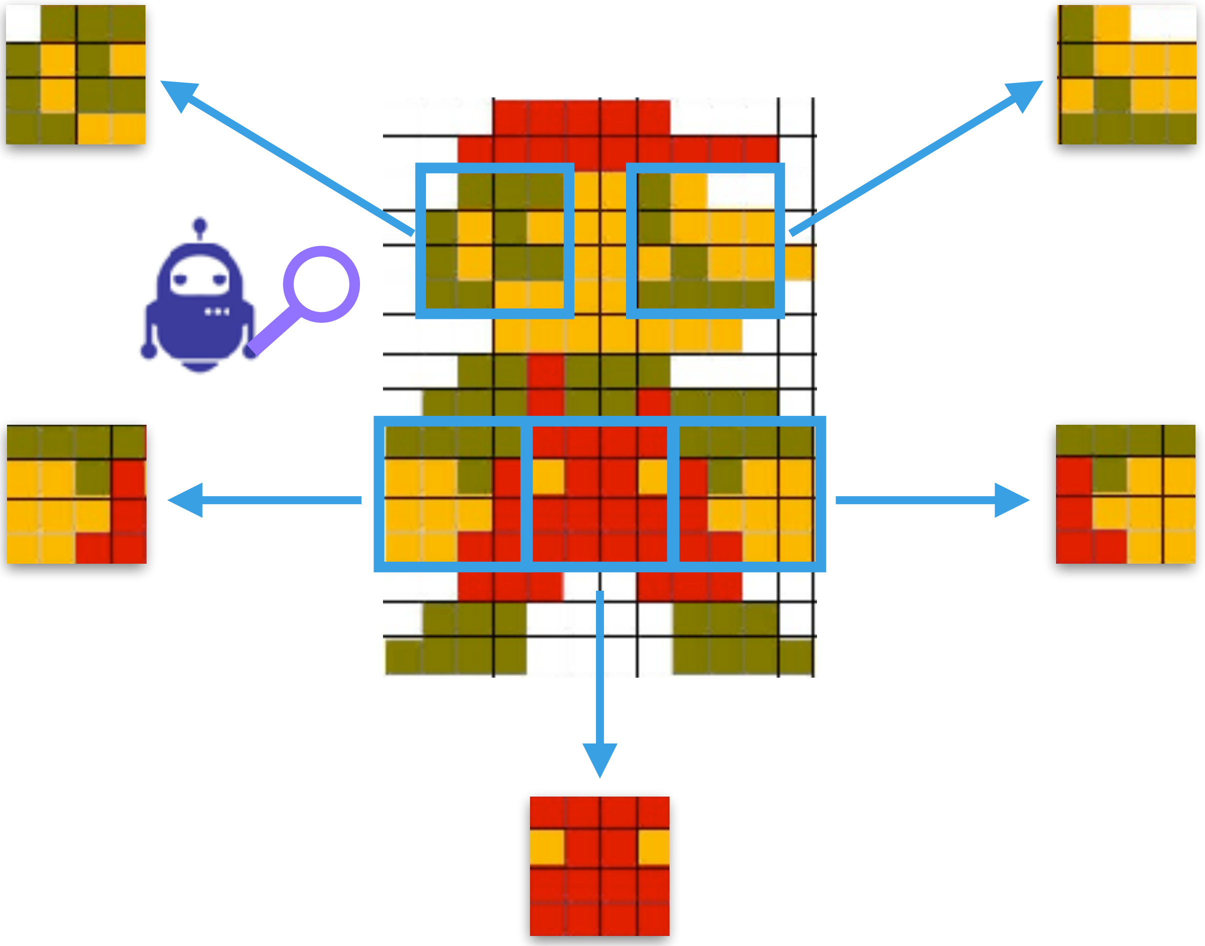
Convolutional Neural Networks (CNN)

Objective

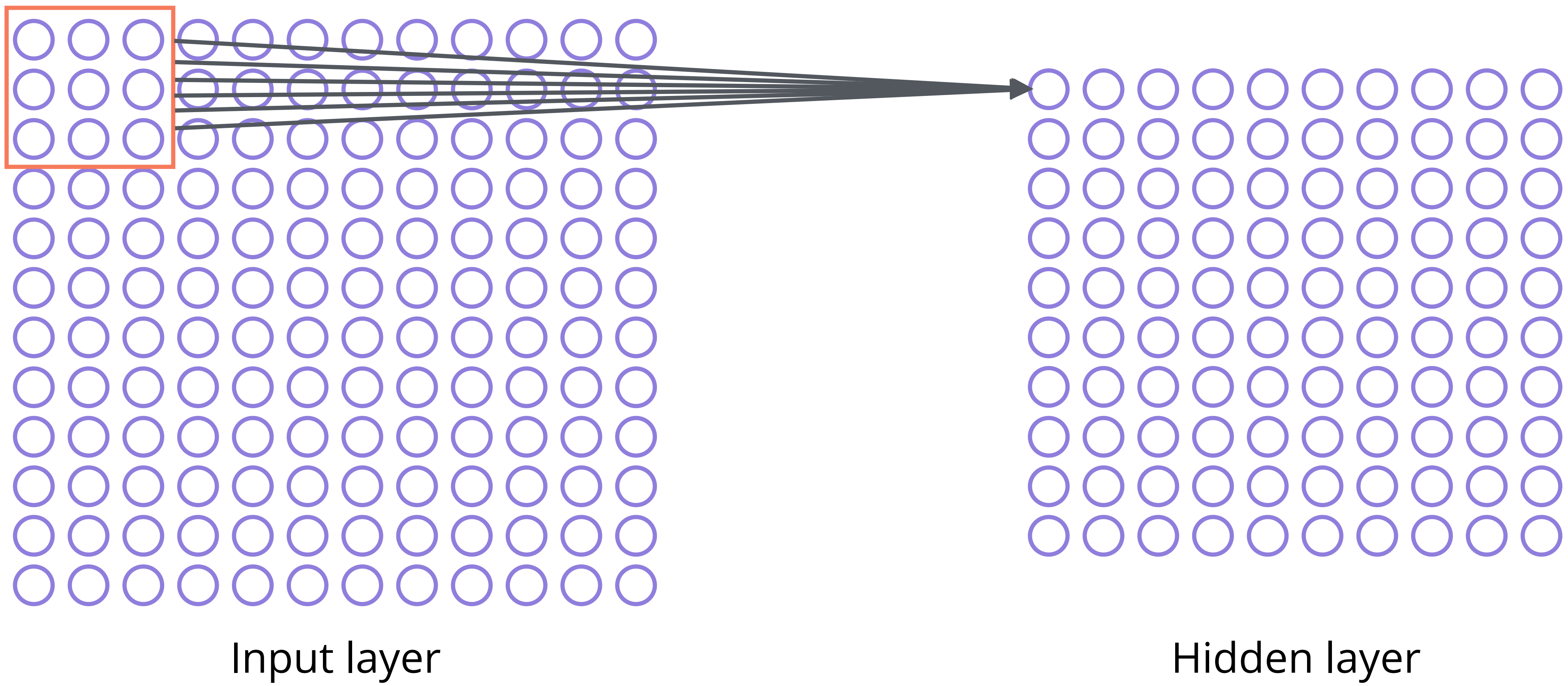
Learn filters that identify an object in an image.

CNN concepts

- The local receptive field
- Convolutions and shared weights
- Pooling



Local receptive field

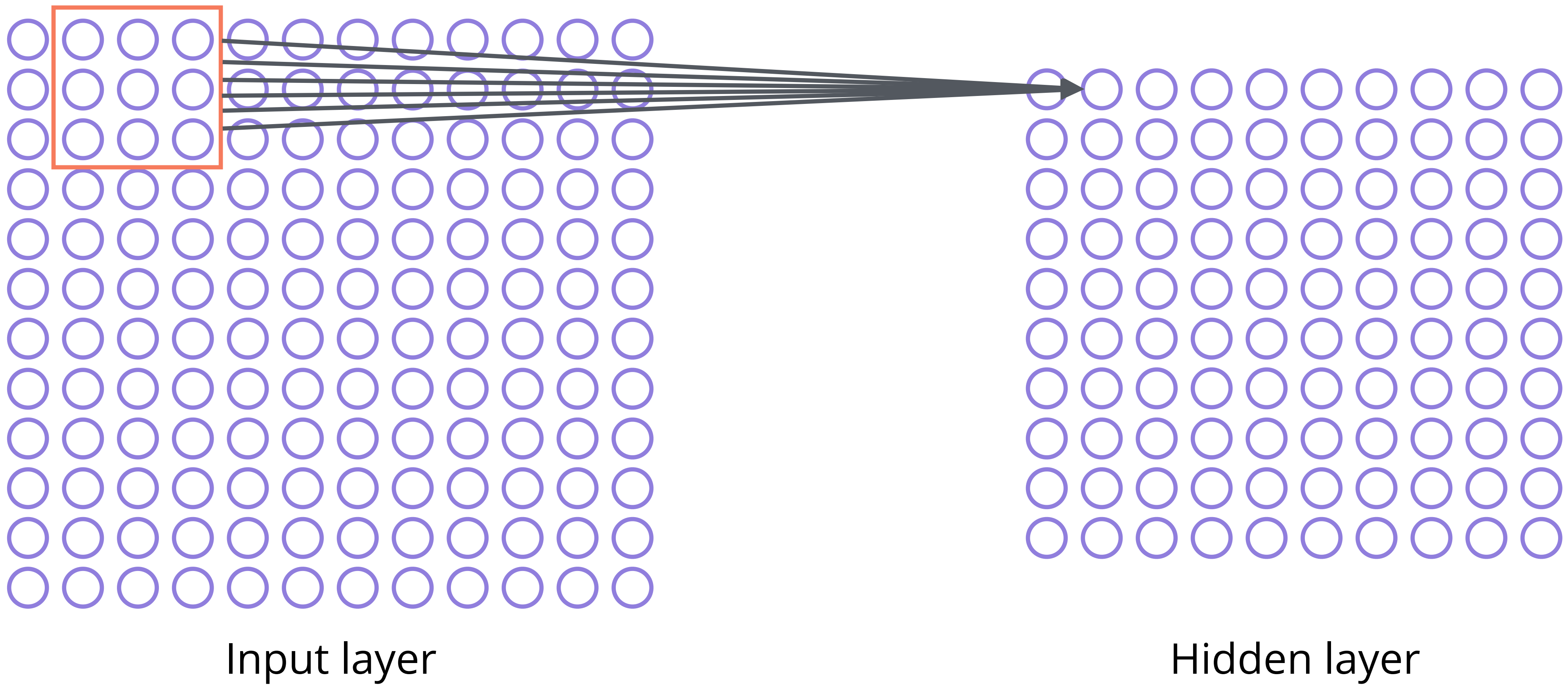


- In a FCNN, all neurons in a layer where connected to the adjacent layer.

- A local receptive field defines a window that is connected a single neuron in the hidden layer.

I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning (MIT Press, 2016). http://www.deeplearningbook.org.

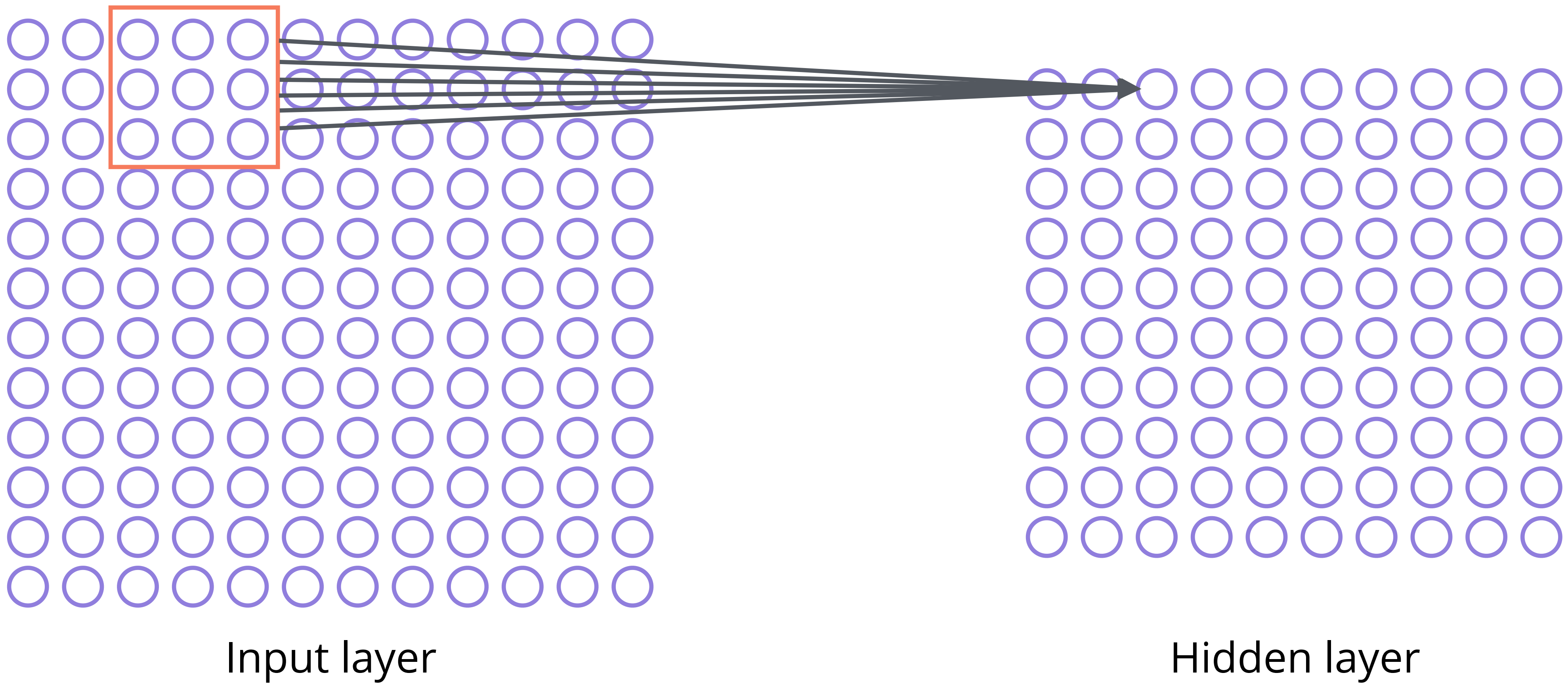
Local receptive field



- This window is then moved across the input, in this case sliding to the right by one pixel.

I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning (MIT Press, 2016). http://www.deeplearningbook.org.

Local receptive field



- Each time the window is moved, the weights of the input layers local receptive field are connected to a different hidden layer.
- The amount the window is moved is known as the stride length.

I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning (MIT Press, 2016). http://www.deeplearningbook.org.

Shared weights and biases

- In this example a 3 x 3 local receptive field is moved across the image where at each stride, its output is connected to a single hidden neuron.

- However, the 3 x 3 weights and biases that comprise the local receptive field will be shared for each neuron in the hidden layer.

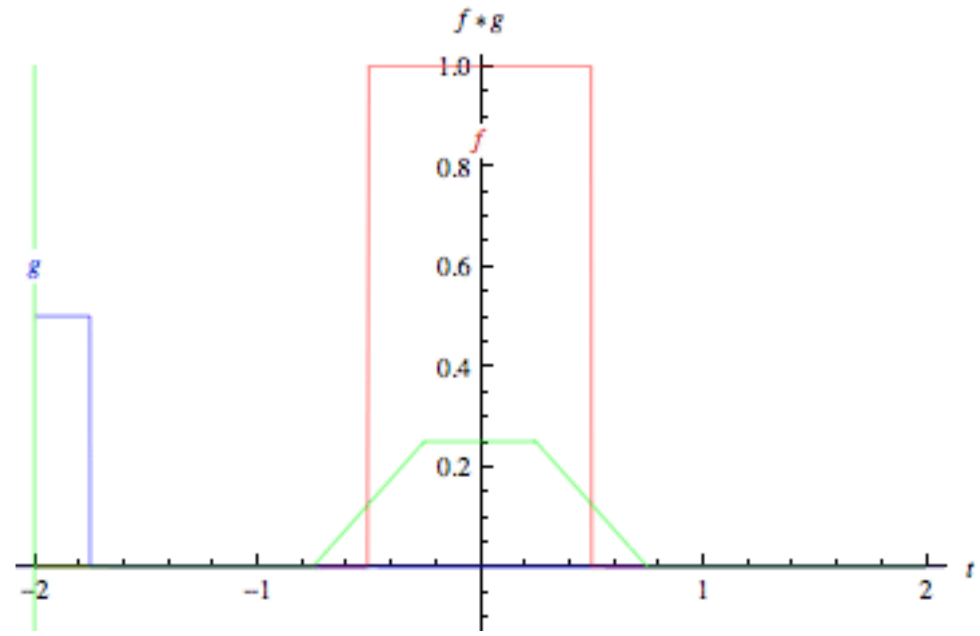
- What this means is that the local receptive field acts as a filter (aka kernel) while the input to hidden layer defines a feature map.

What is a convolution?

- A convolution expresses the amount of overlap a function **g** has as it is shifted over **f**.

$$(f^*g)(t) \triangleq \int_{-\infty}^{\infty} f(\tau)g(t-\tau)\tau'$$

$$(f * g)(x, y) = \sum_m \sum_n f(m, n)g(x - m, y - n)$$

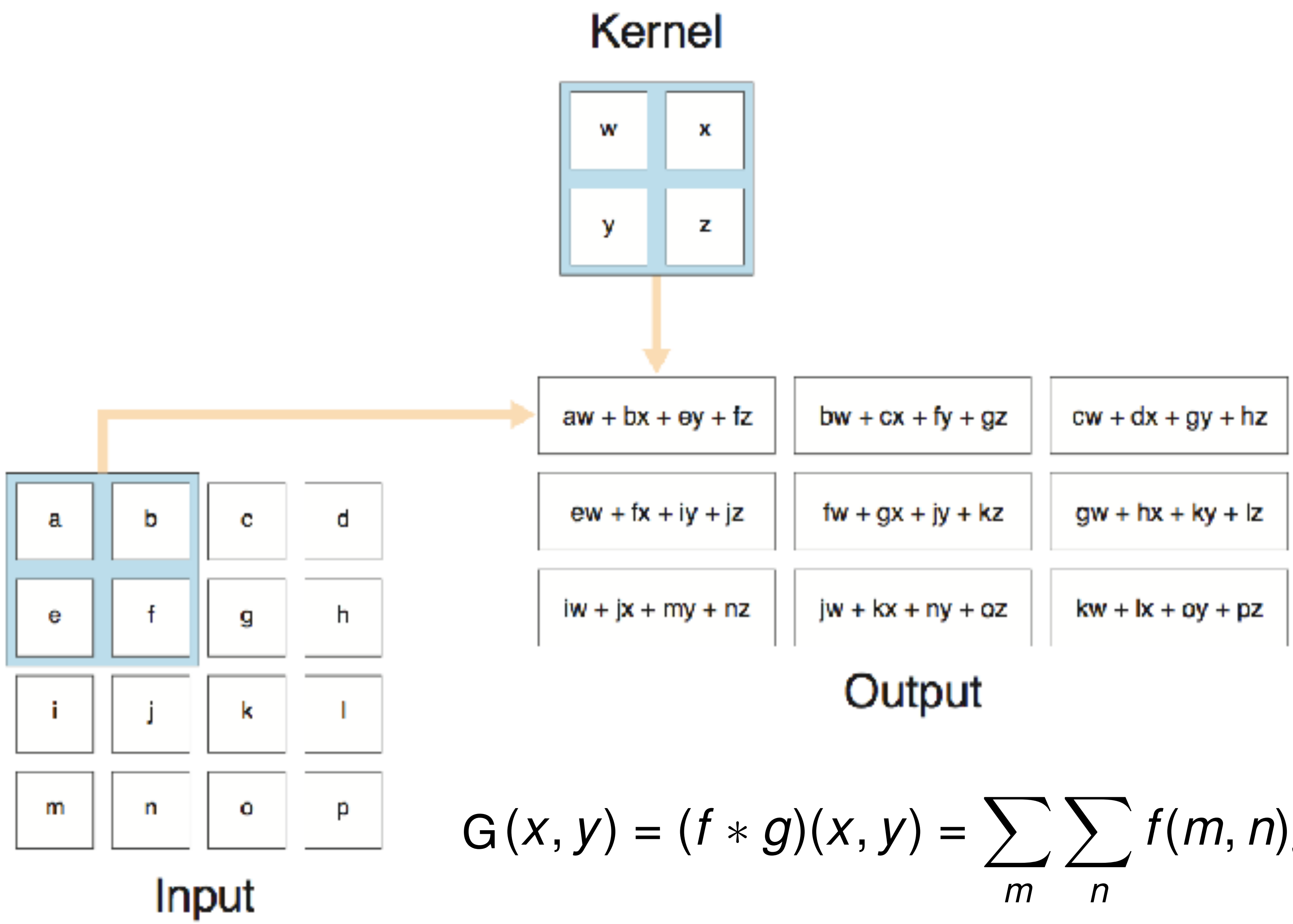


https://mathworld.wolfram.com/Convolution.html

- For a CNN **g** defines our filter (kernel) where **f** is the input neurons.

- Therefore, **f * g** defines the feature maps of the hidden layer!

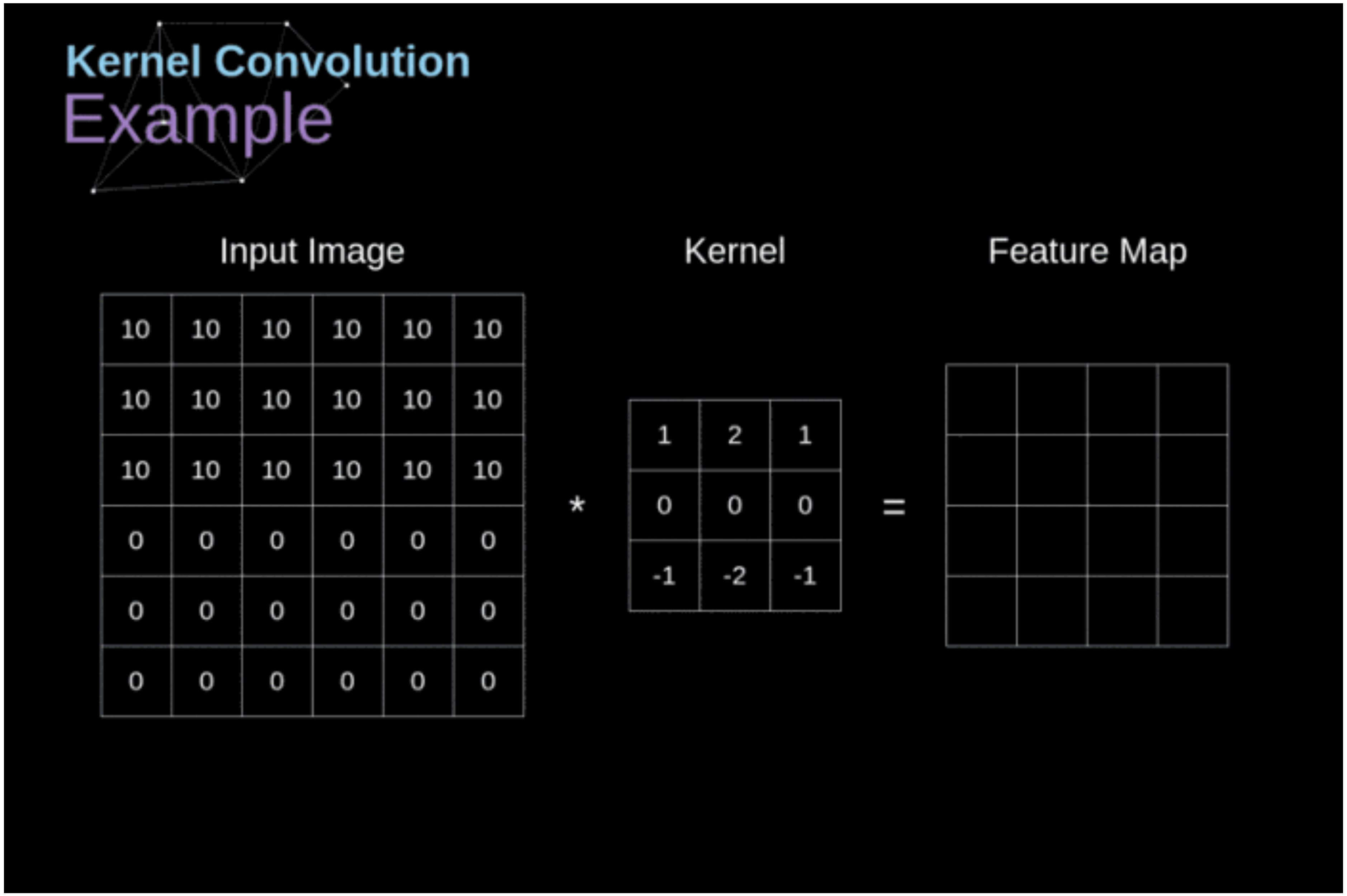
Convolution layer



$$G(x, y) = (f * g)(x, y) = \sum_m \sum_n f(m, n) g(x - m, y - n)$$

I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning (MIT Press, 2016).

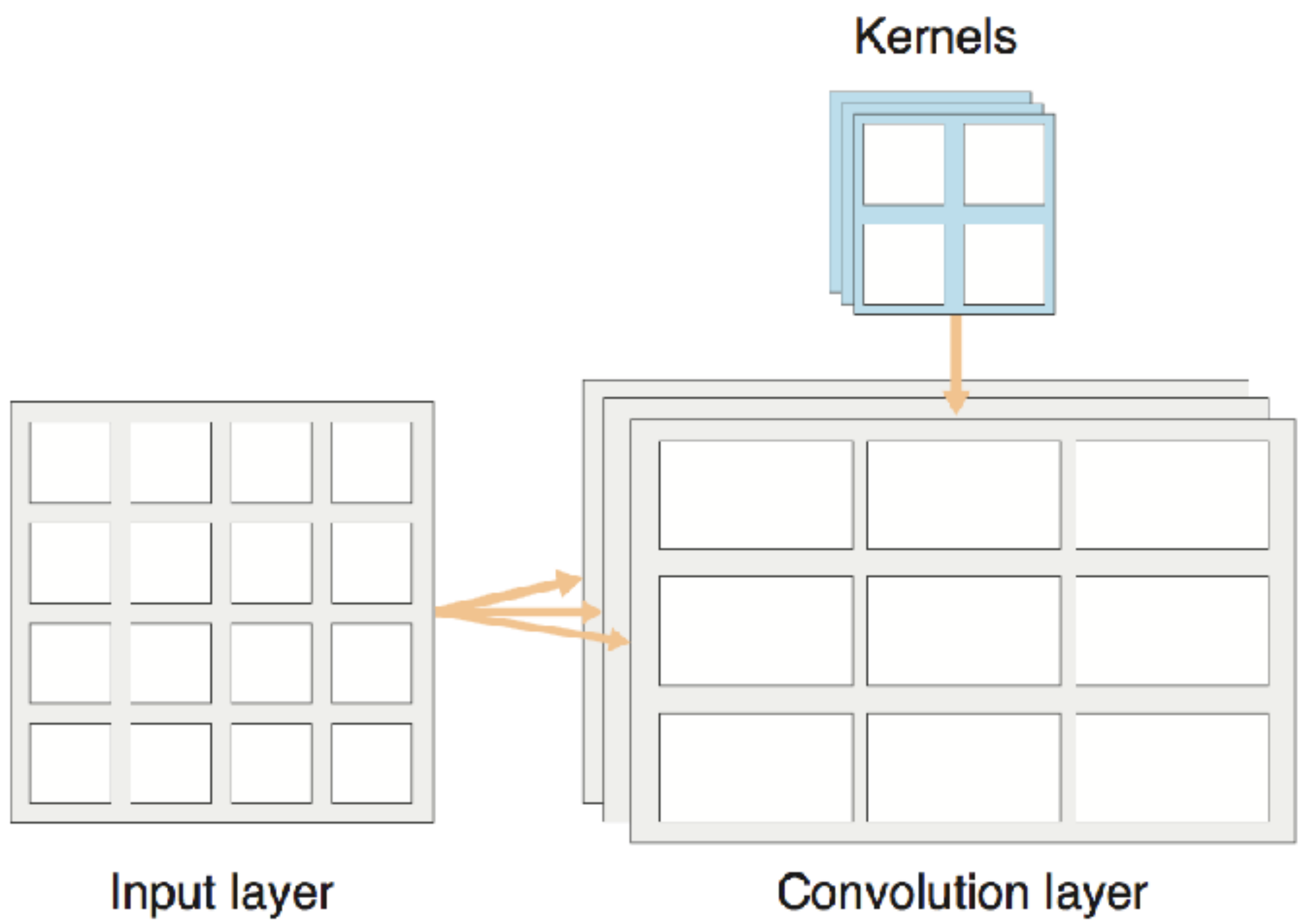
Convolution layer



$$G(x,y) = (f * g)(x,y) = \sum_{m}\sum_{n} f(m,n)g(x-m, y-n)$$

https://towardsdatascience.com/gentle-dive-into-math-behind-convolutional-neural-networks-79a07dd44cf9

Convolution layer



Convolution layer cont.

For visualization purposes consider edge detection using a Sobel filter:

Horizontal

$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * \mathbf{A}$$

Filter (Kernel)

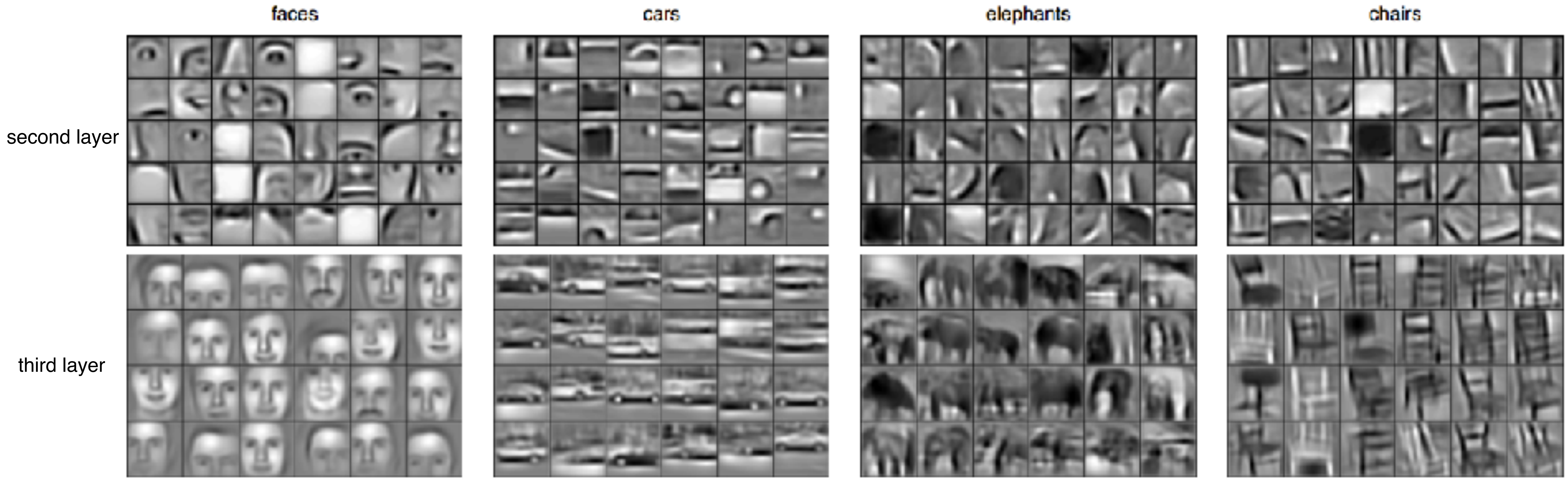
Vertical

$$\mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A}$$

Convolution layer cont.

- Unlike the Sobel example, for CNN we do not know the filter a priori
- However, because the filter shares weights, as the network trains it **learns the best filter(s) for a given task!**

Filters in a CNN



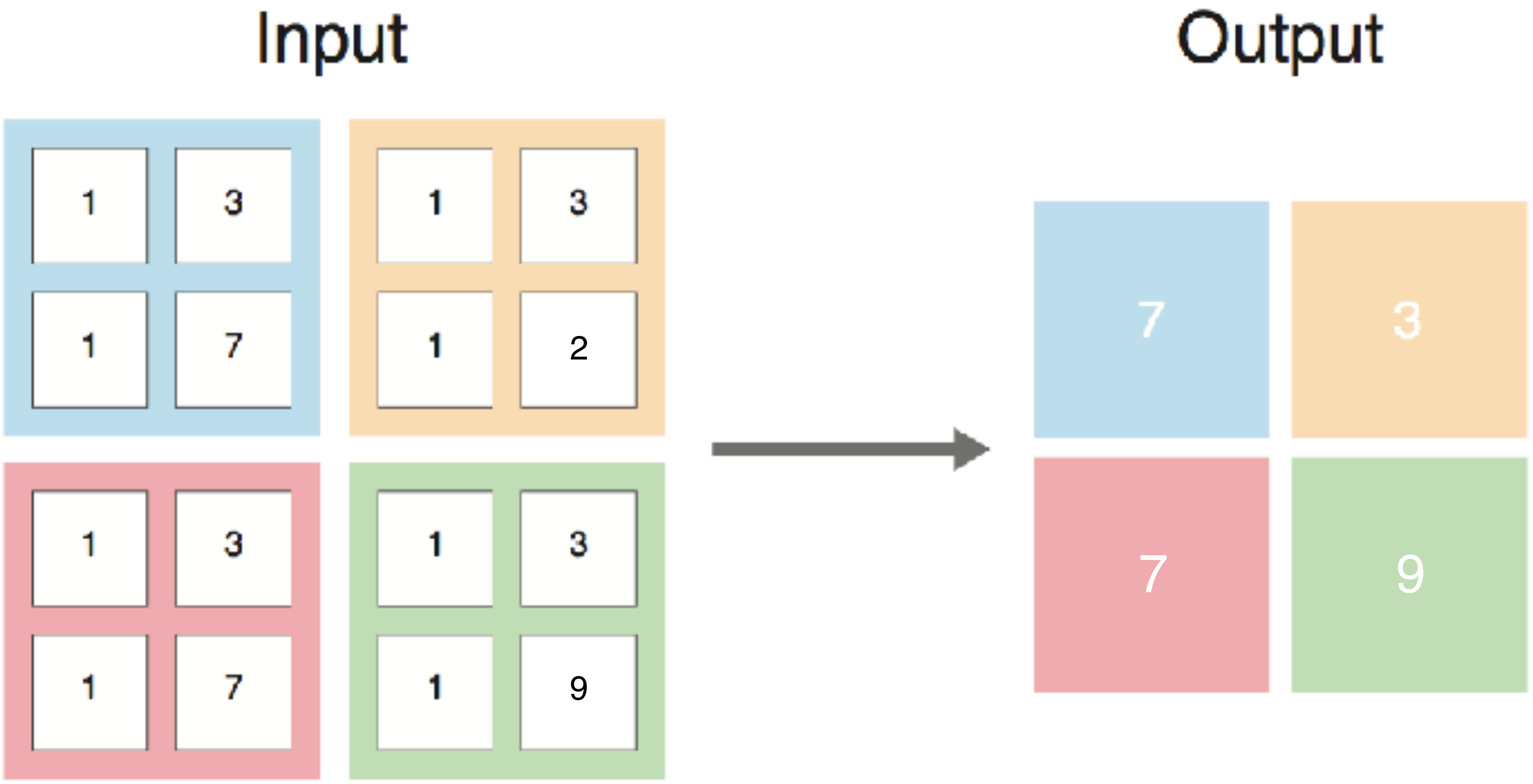
Lee, H., Grosse, R., Ranganath, R. and Ng, A.Y., 2009, June. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In Proceedings of the 26th annual international conference on machine learning (pp. 609-616). ACM.

Pooling

- Feature maps are sensitive to the location of the features in an input.
- Therefore, small translations in a features position will result in different feature maps.
- Pooling downsamples a feature map, which create a lower resolution version that contains all the important information (think of it as a summary)
- This adds invariance to local translation, making the network robust to changes.

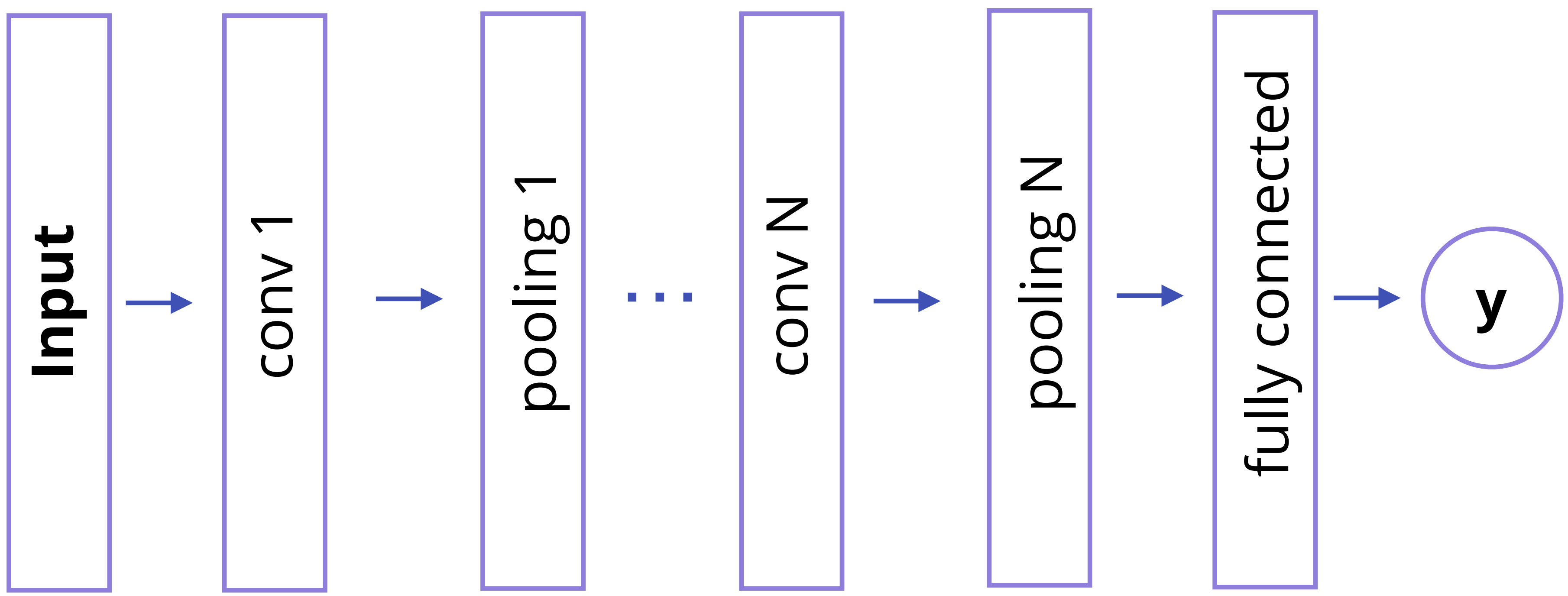
I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning (MIT Press, 2016).

Max pooling



I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning (MIT Press, 2016).

Putting it all together



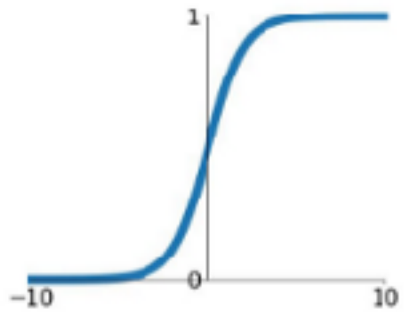
Building a CNN

- Returning to MNIST, we are now going to build a CNN using Tensorflow (TF)
- At it's core TF is a symbolic math library that provides ways to build computational graphs and preform auto-differentiation.
- This makes is a fantastic tool for building machine learning models.
- For this example we will use Keras, which is a high level API for building NN included in TF.

Activation functions

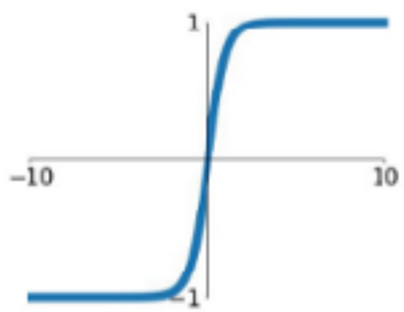
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



tanh

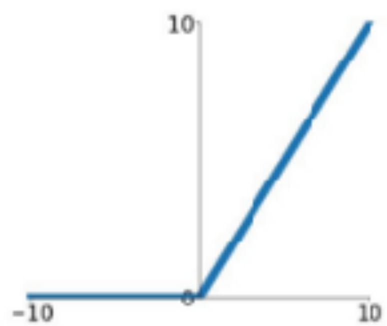
$\tanh(x)$



Rectified Linear Unit

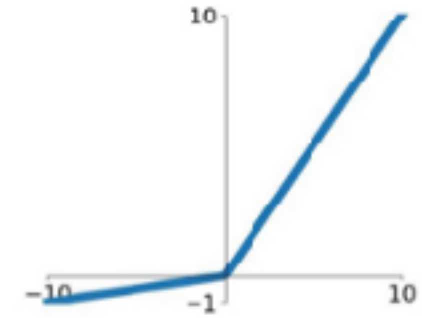
ReLU

$\max(0, x)$



Leaky ReLU

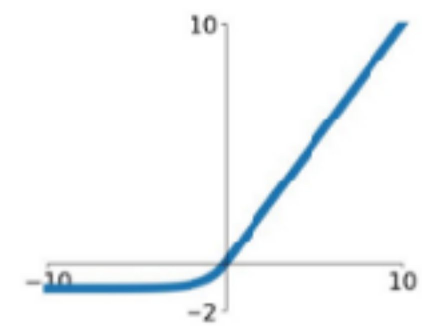
$\max(0.1x, x)$



Exponential Linear Unit

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



https://towardsdatascience.com/complete-guide-of-activation-functions-34076e95d044

Git Repo

https://github.com/ericgossett/Intro-to-Neural-Networks-Tech-Talk