GitHub

PyTorch Introduction

Workshop on Artificial Intelligence - 03/02/2026

Matthis Dallain & Alexandre Lainé - NeOpTo team
mattis.dallain@univ-amu.fr / alexandre.laine@univ-amu.fr

Aix Marseille Université

institut de neurosciences de la timone

Summary

1. PyTorch, What is that ?
2. Gradient descent
3. Few examples

1. PyTorch, What is that ?

1. PyTorch, What is that ?

Open source library (since 2017) for deep learning, initially developed by Meta and based on python programming language.

1. PyTorch, What is that ?

Open source library (since 2017) for deep learning, initially developed by Meta and based on python programming language.

1. PyTorch, What is that ?

Open source library (since 2017) for deep learning, initially developed by Meta and based on python programming language.

1. PyTorch, What is that ?

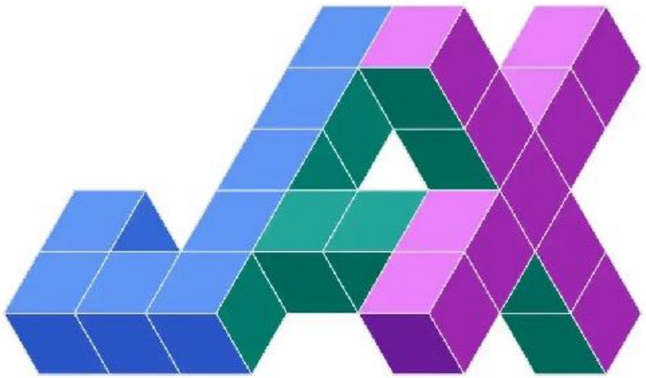
Meta

python™

PyTorch

TensorFlow

Keras



Open source library (since 2017) for deep learning, initially developed by Meta and based on python programming language.

1. PyTorch, What is that ?

Meta

python™

PyTorch

Open source library (since 2017) for deep learning, initially developed by Meta and based on python programming language.

Open Source Projet Evolution (based on commits, PR, Issues, Authors)

Rank	Label	Commits
1	Firebase	29,366
2	Linux	72,148
3	NixOS	121,407
4	React	62,173
5	Kubernetes	53,495
6	Hugging Face	45,612
7	LLVM	75,424
8	Datadog	195,792
9	pytorch	123,917
10	AMD ROCm	61,887
11	Arm Toolchain	51,094
12	Grafana Labs	124,391
13	Home Assistant	28,670
14	dotnet	97,705
15	ODOO Community Association	48,976
16	Rust	49,046

Source : Cloud Native Computing Foundation (CNCF). (2026). *Cncf/velocity* [Shell]. https://github.com/cncf/velocity (Original work published 2017)

1. PyTorch, What is that ?

Sources : *OpenAI standardizes on PyTorch*. (2024, March 13). / PyTorch (Director). (2019, November 6). *PyTorch at Tesla—Andrej Karpathy, Tesla*

1. PyTorch, What is that ?

Already used in lot of private companies / projects :

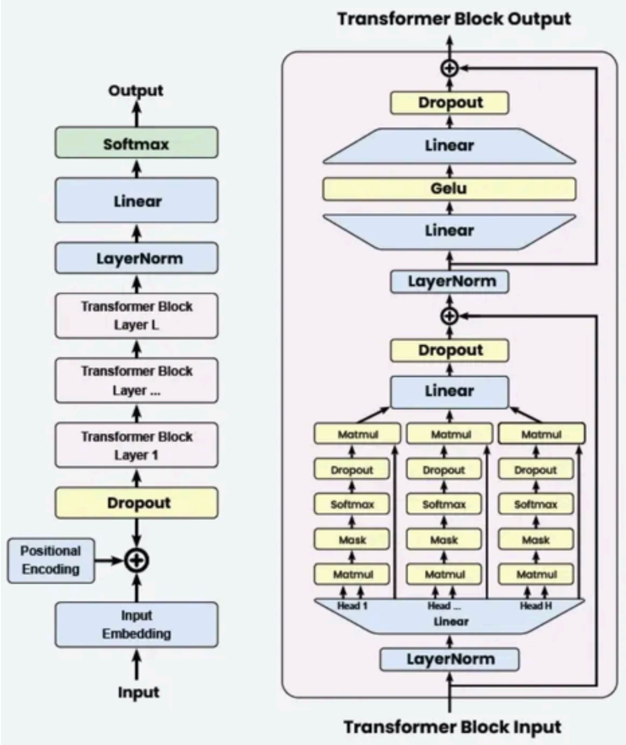
Sources : *OpenAI standardizes on PyTorch*. (2024, March 13). / PyTorch (Director). (2019, November 6). *PyTorch at Tesla—Andrej Karpathy, Tesla*

1. PyTorch, What is that ?

Already used in lot of private companies / projects :

OpenAI

Generative **P**re-trained
Transformers



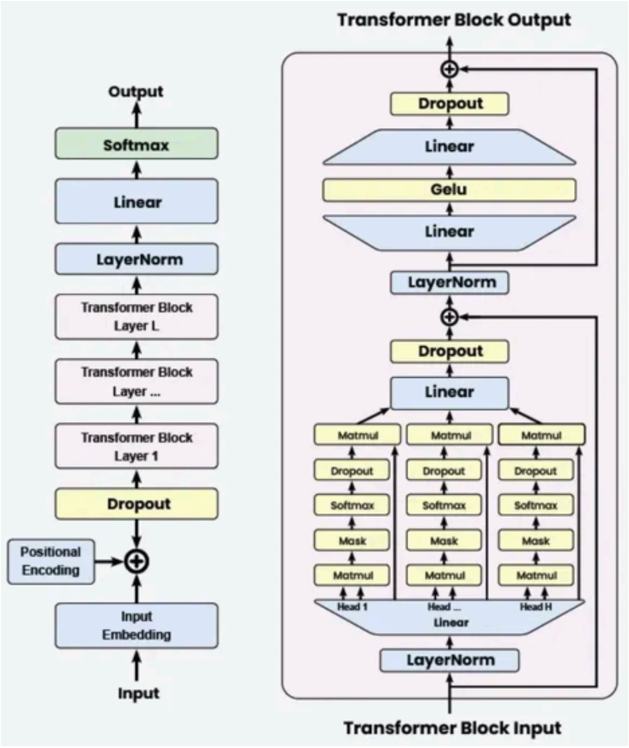
Sources : *OpenAI standardizes on PyTorch*. (2024, March 13). / PyTorch (Director). (2019, November 6). *PyTorch at Tesla—Andrej Karpathy, Tesla*

1. PyTorch, What is that ?

Already used in lot of private companies / projects :

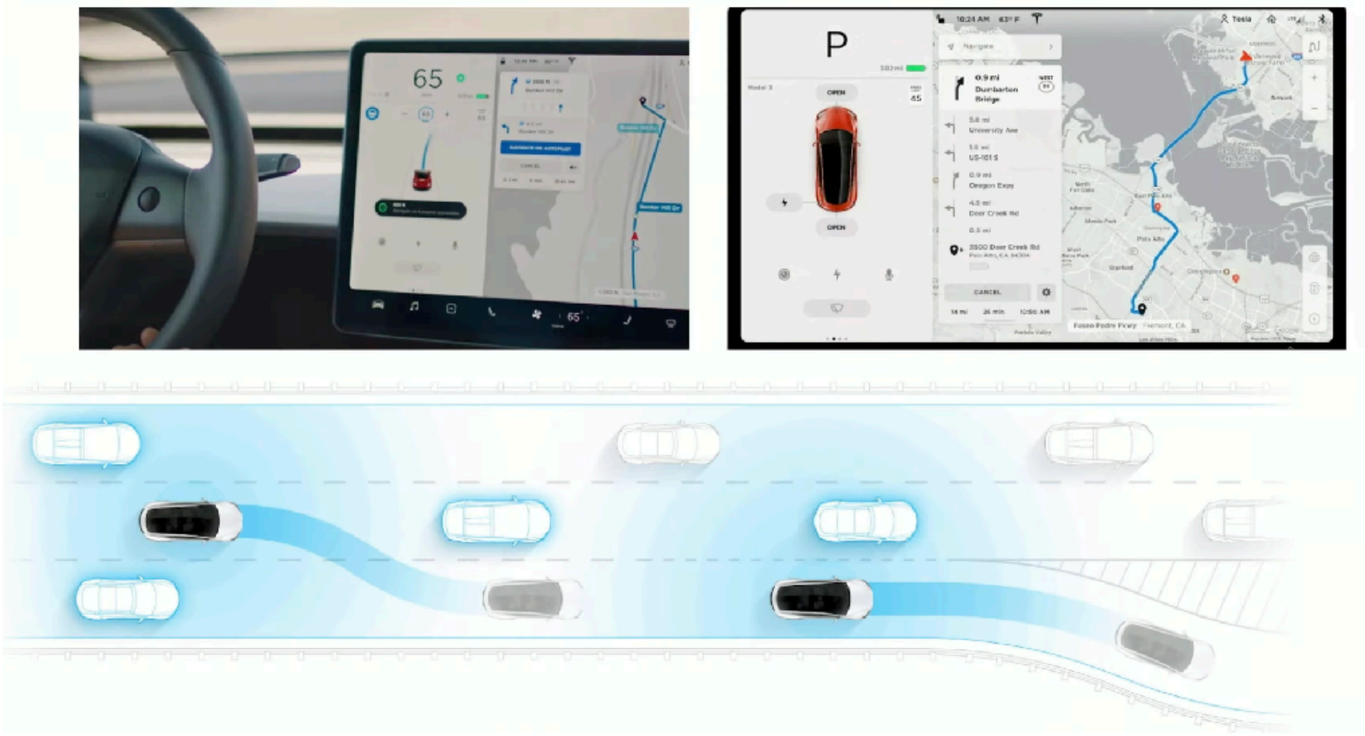
OpenAI

Generative **P**re-trained **T**ransformers



TESLA

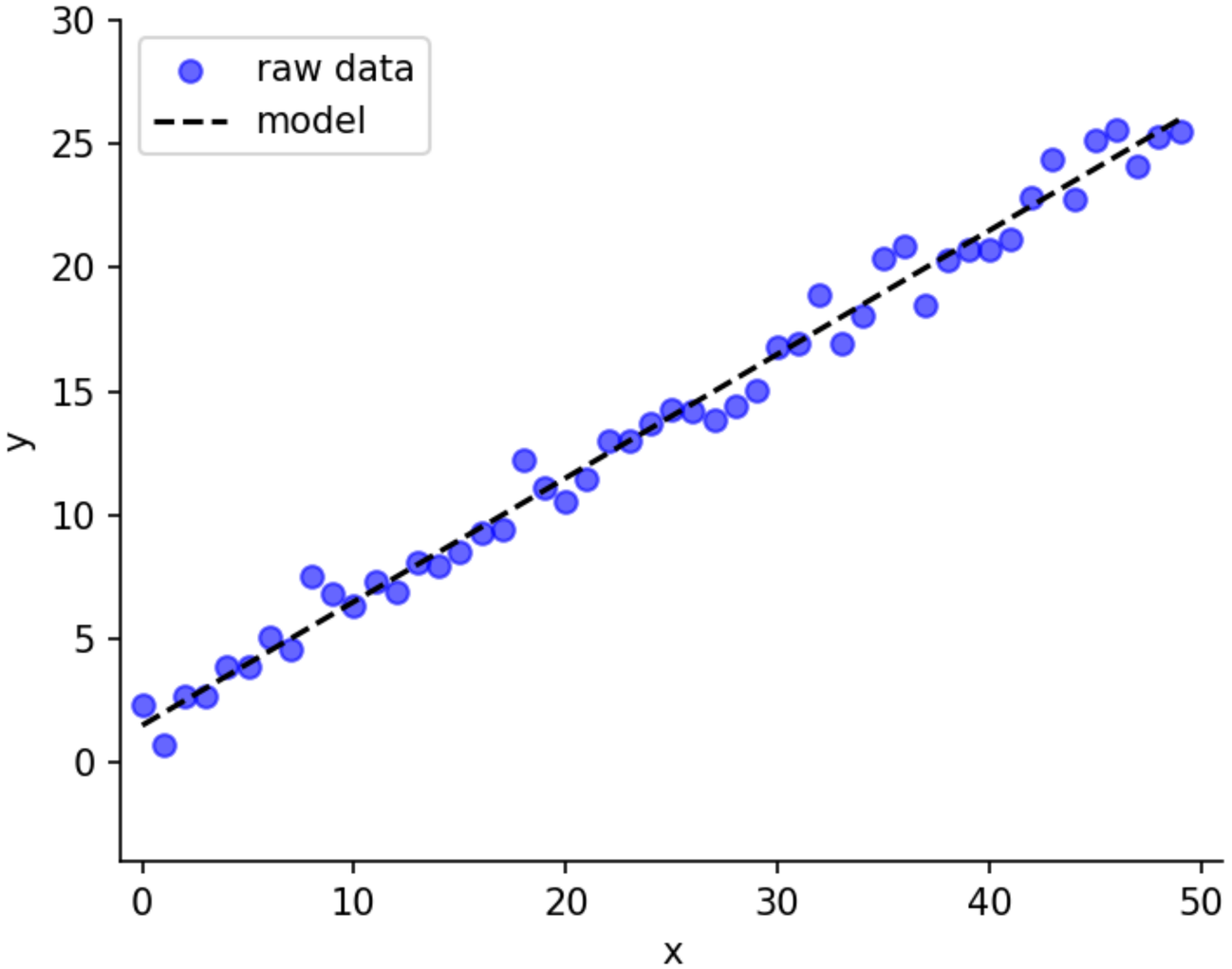
AutoPilot algorithm training



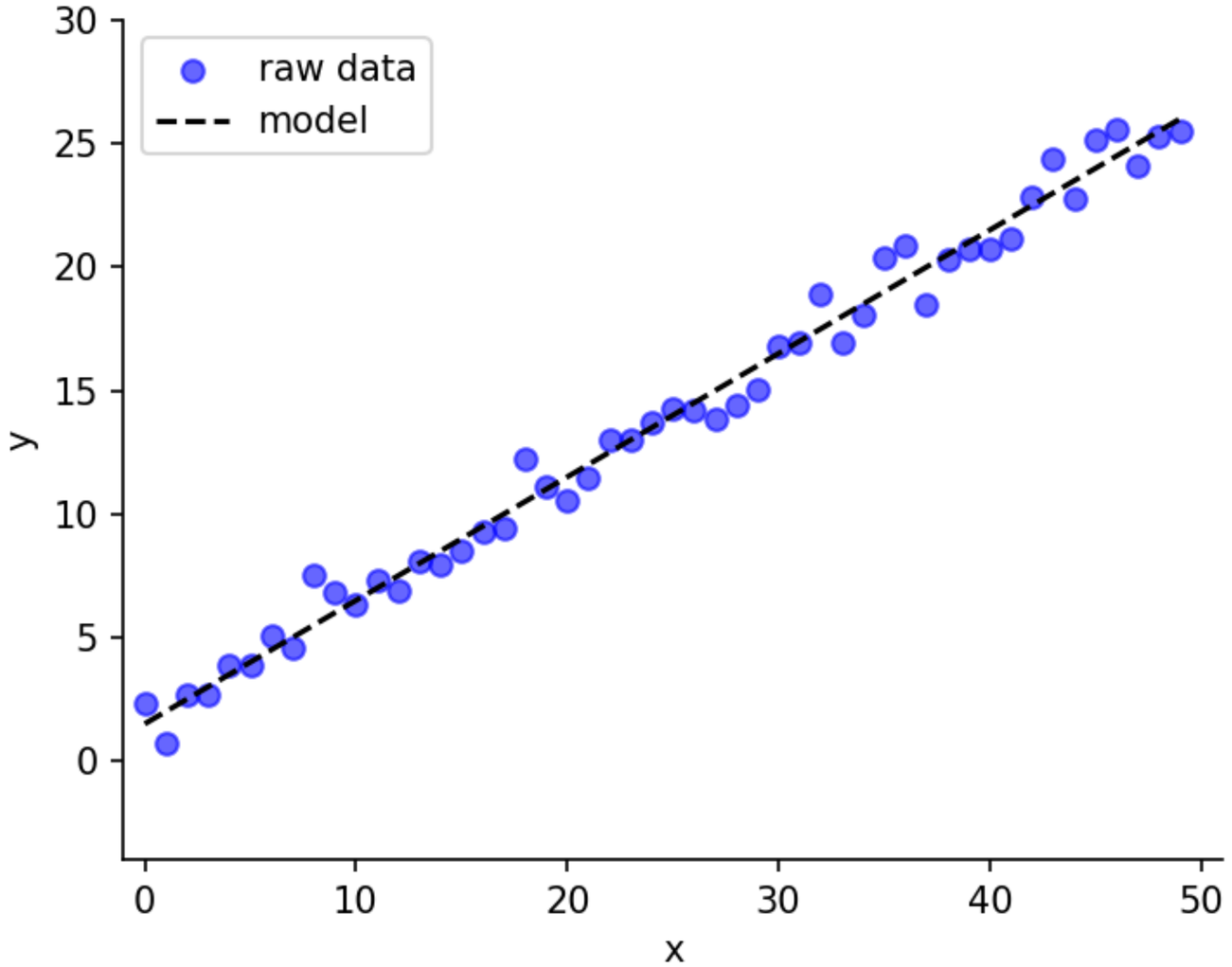
Sources : *OpenAI standardizes on PyTorch*. (2024, March 13). / PyTorch (Director). (2019, November 6). *PyTorch at Tesla—Andrej Karpathy, Tesla*

2. Gradient descent (an iterative process)

2. Gradient descent (an iterative process)



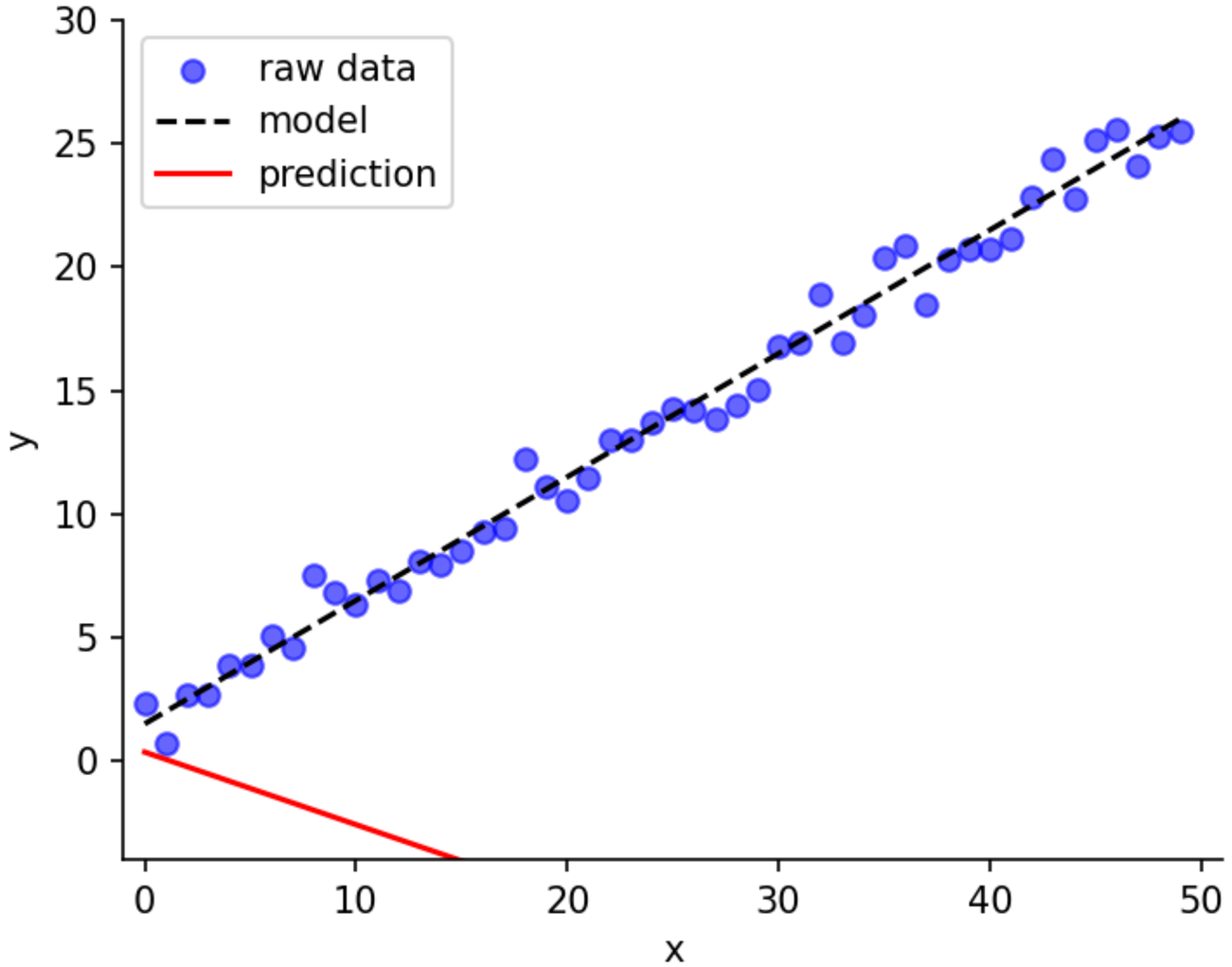
2. Gradient descent (an iterative process)



Model prediction

$$y' = a_t x + b_t$$

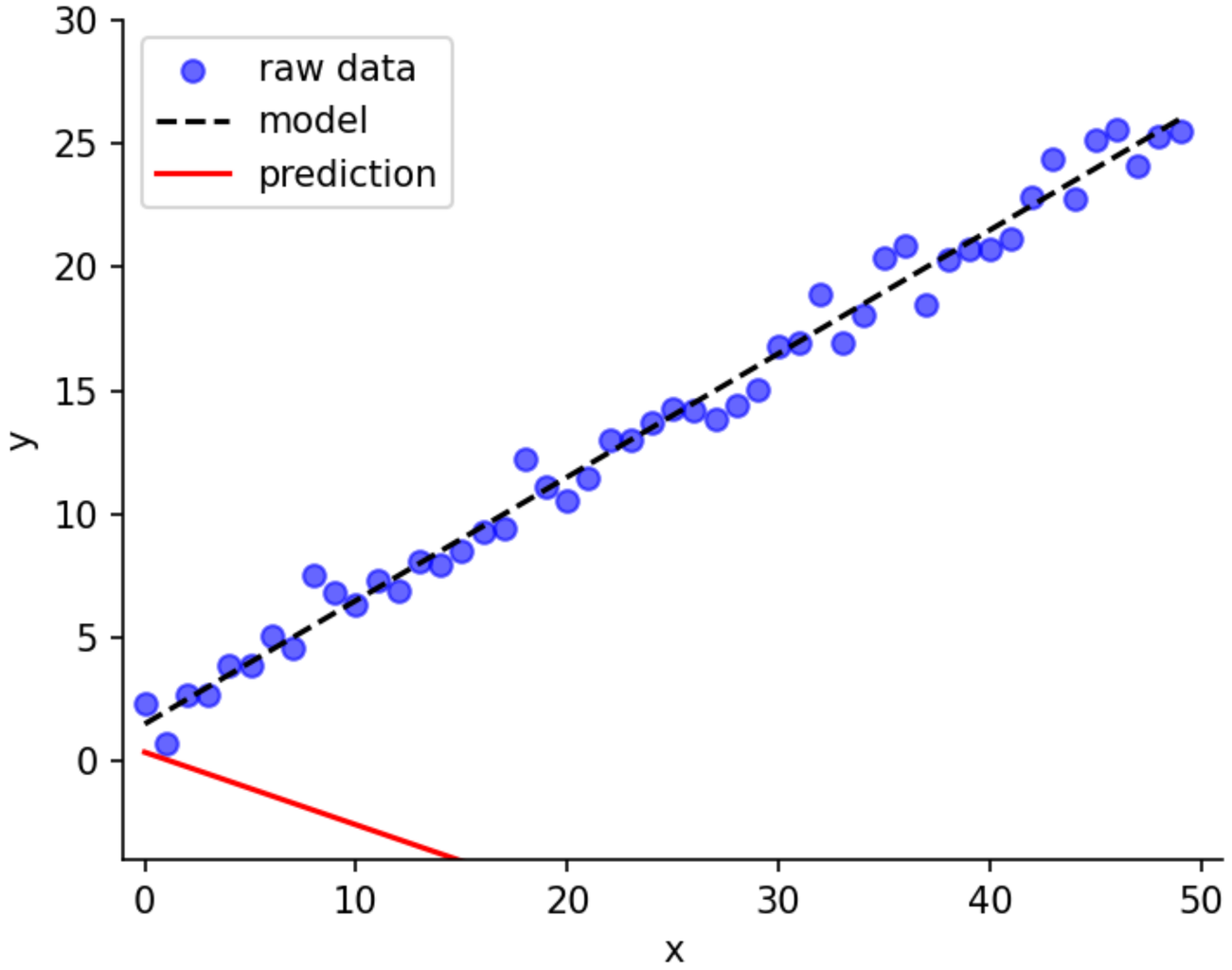
2. Gradient descent (an iterative process)



Model prediction

$$y' = a_t x + b_t$$

2. Gradient descent (an iterative process)

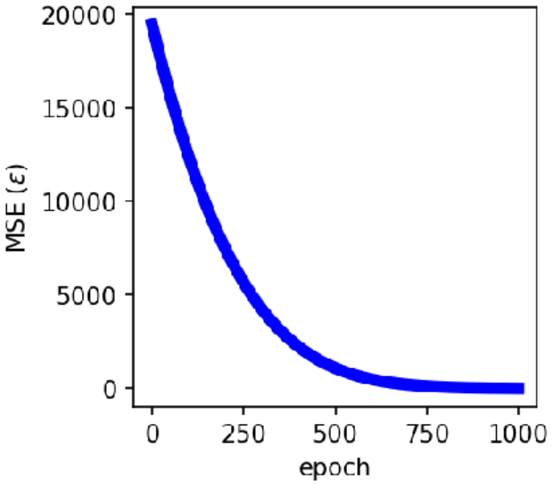


Model prediction

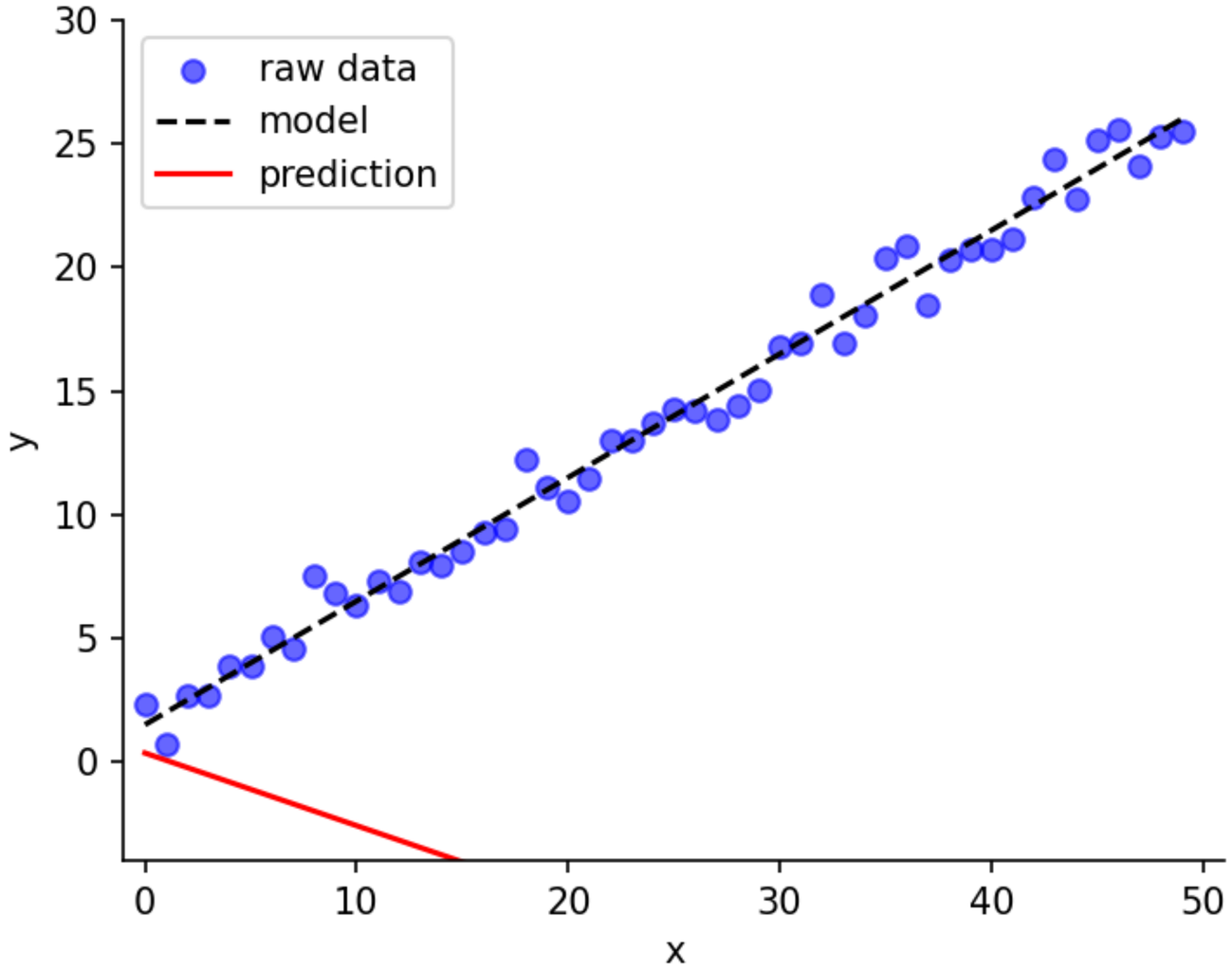
$$y' = a_t x + b_t$$

Loss function (MSE)

$$\epsilon = \frac{1}{n}\sum_{i=1}^{n}(y_i - y_i')^2$$



2. Gradient descent (an iterative process)

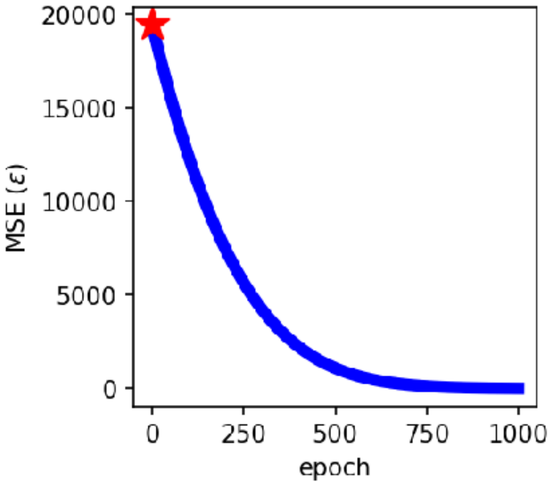


Model prediction

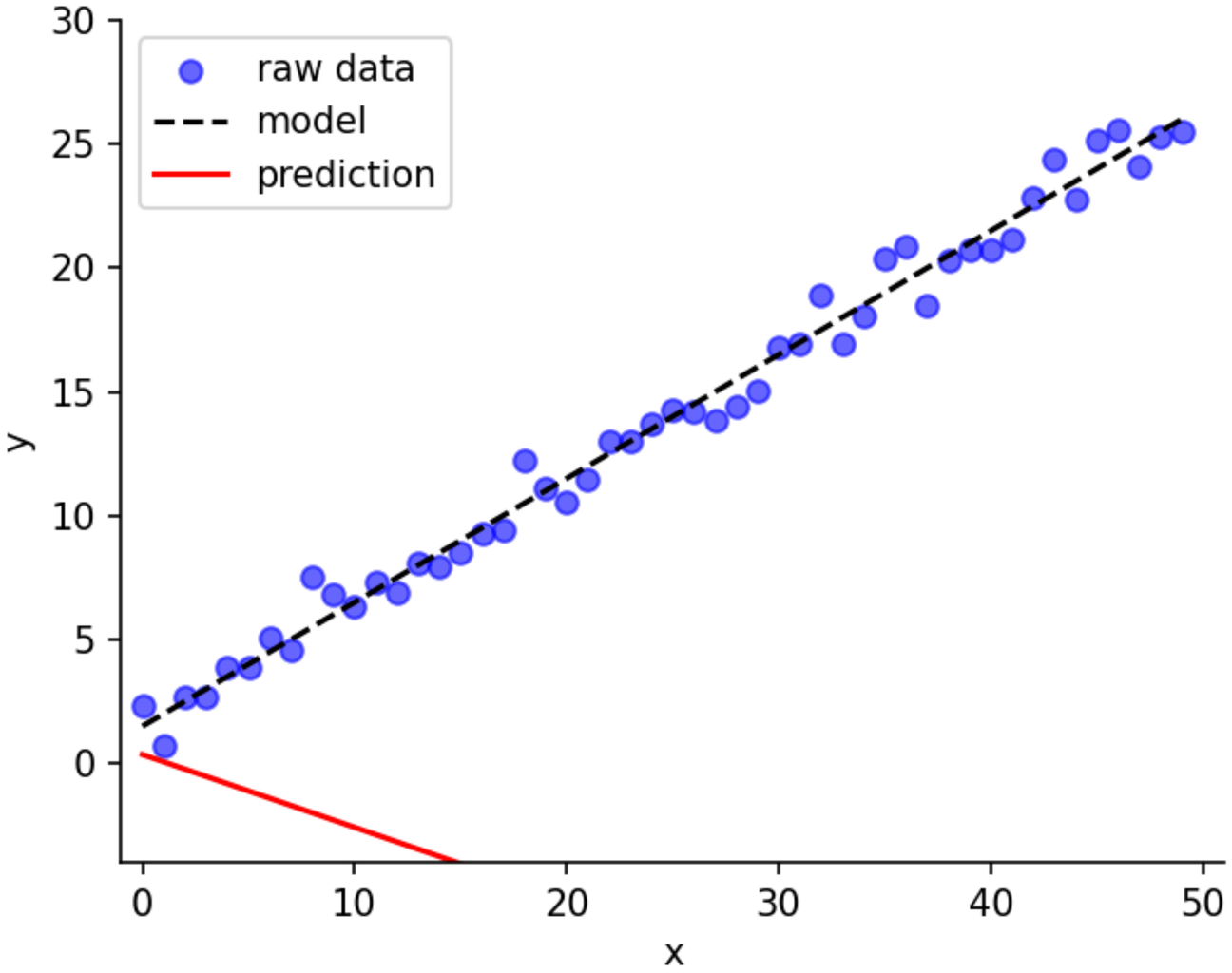
$$y' = a_t x + b_t$$

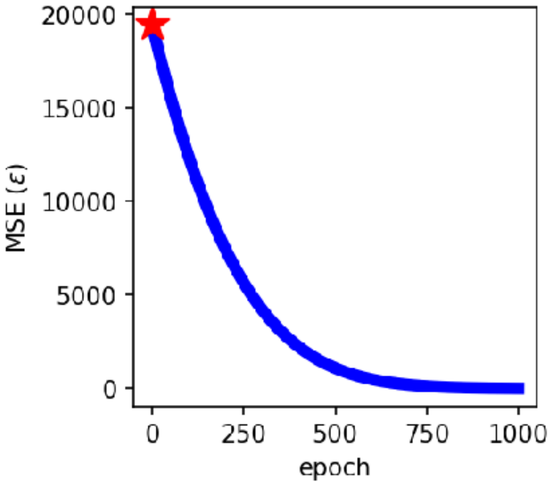
Loss function (MSE)

$$\epsilon = \frac{1}{n}\sum_{i=1}^{n}(y_i - y_i')^2$$



2. Gradient descent (an iterative process)





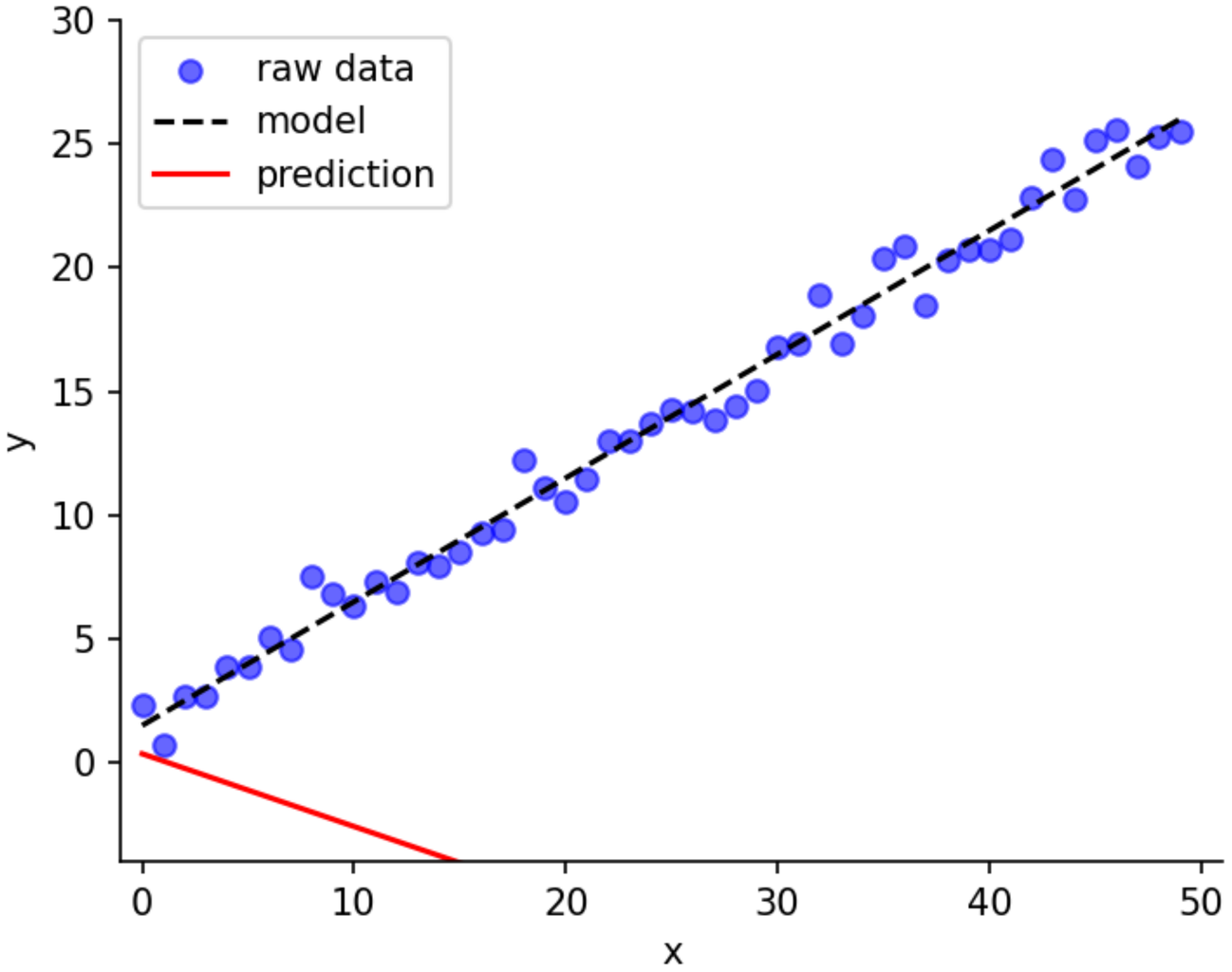
Model prediction

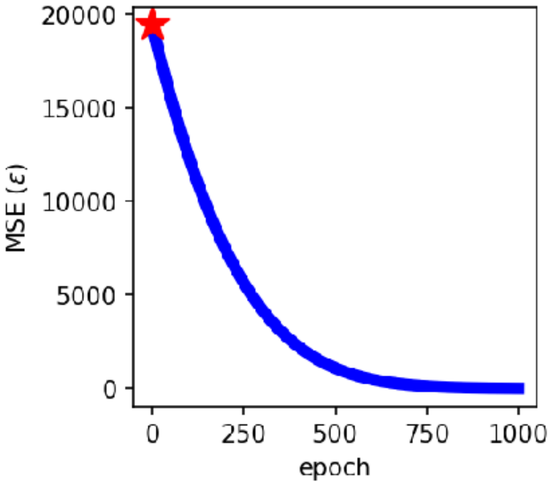
$$y' = a_t x + b_t$$

Loss function (MSE)

$$\epsilon = \frac{1}{n}\sum_{i=1}^{n}(y_i - y_i')^2 = \frac{1}{n}\sum_{i=1}^{n}(a_t x + b_t - y)^2$$

2. Gradient descent (an iterative process)





Model prediction

$$y' = a_t x + b_t$$

Loss function (MSE)

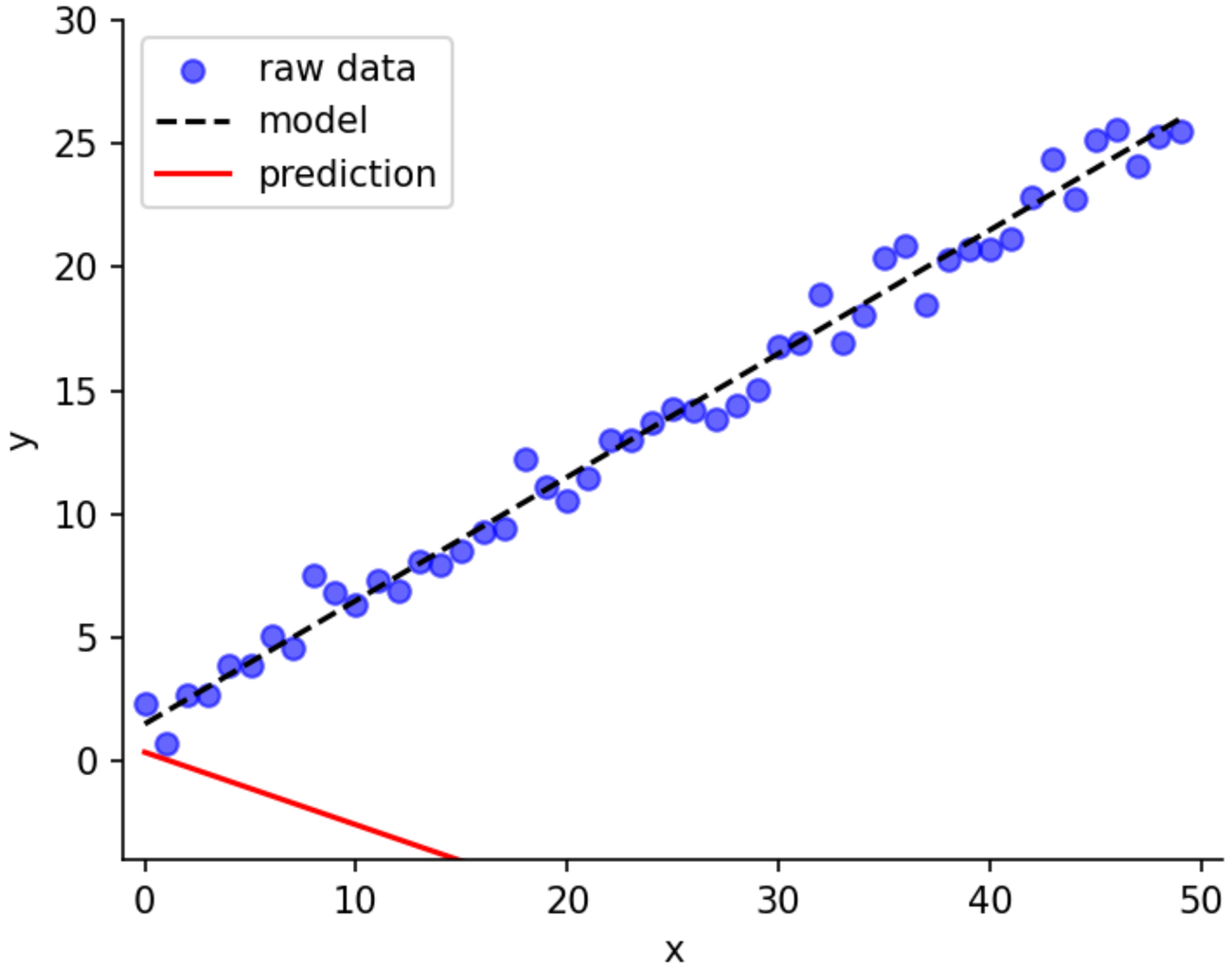
$$\epsilon = \frac{1}{n}\sum_{i=1}^{n}(y_i - y_i')^2 = \frac{1}{n}\sum_{i=1}^{n}(a_t x + b_t - y)^2$$

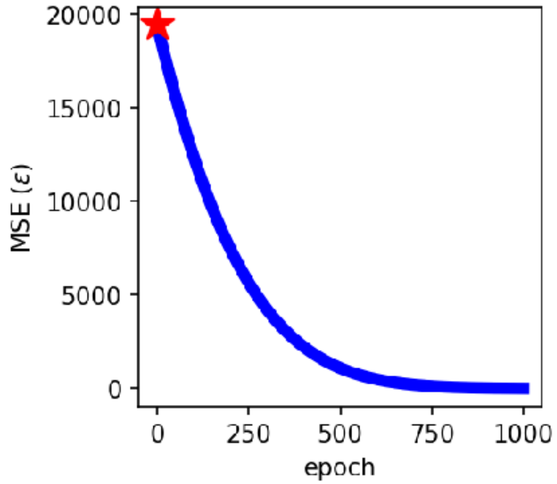
Compute the gradient

$$\frac{\partial \epsilon}{\partial a} = \frac{2}{n}\sum_{i=1}^{n}(a_t x_i + b_t - y_i)x_i$$

$$\frac{\partial \epsilon}{\partial b} = \frac{2}{n}\sum_{i=1}^{n}(a_t x_i + b_t - y_i)$$

2. Gradient descent (an iterative process)





Model prediction

$$y' = a_t x + b_t$$

Loss function (MSE)

$$\epsilon = \frac{1}{n}\sum_{i=1}^{n}(y_i - y_i')^2 = \frac{1}{n}\sum_{i=1}^{n}(a_t x + b_t - y)^2$$

Compute the gradient

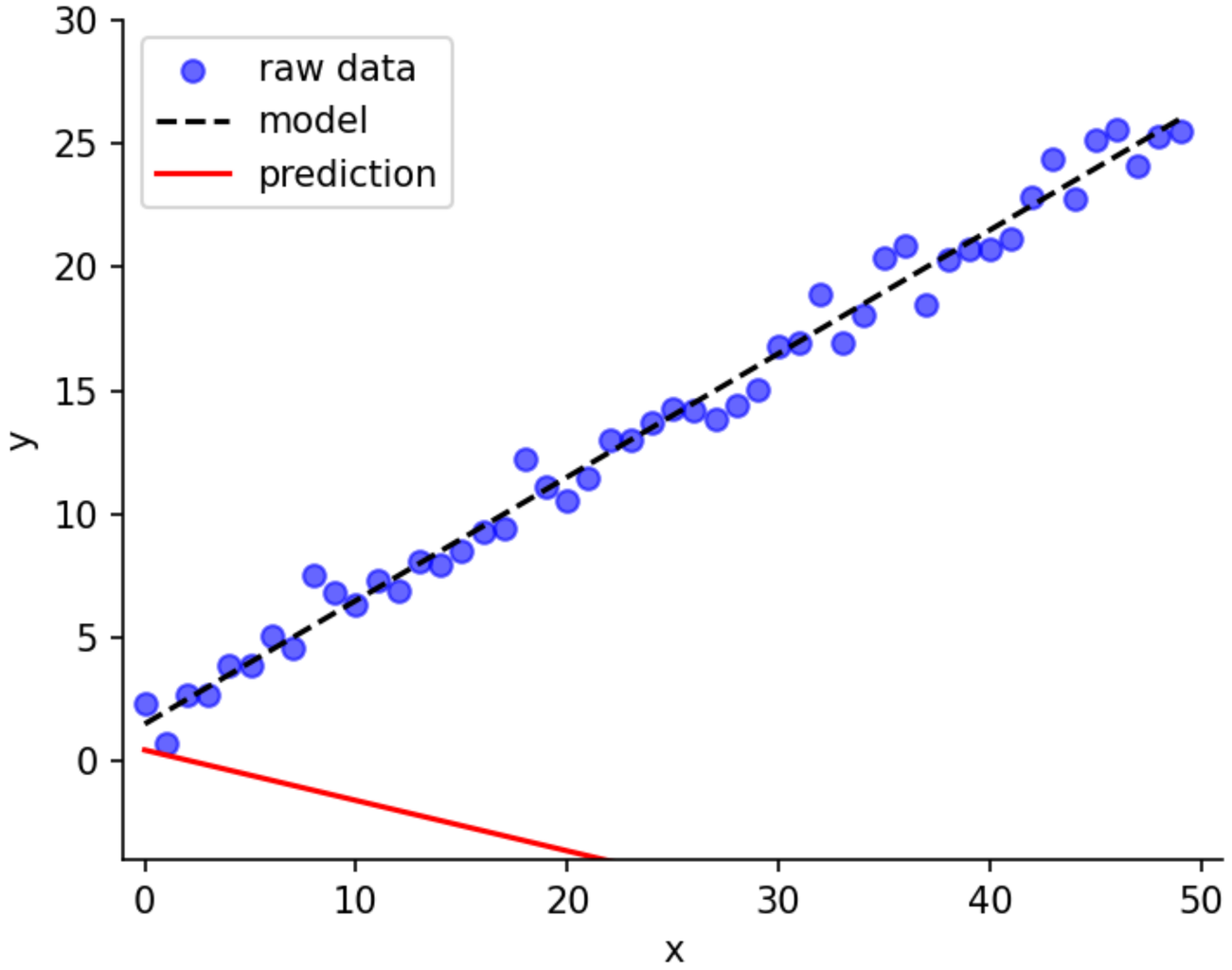
$$\frac{\partial \epsilon}{\partial a} = \frac{2}{n}\sum_{i=1}^{n}(a_t x_i + b_t - y_i)x_i$$

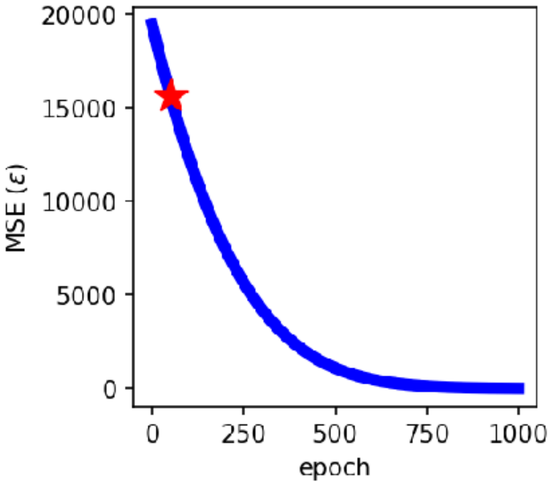
$$\rightarrow \quad a_{t+1} = a_t - \lambda\frac{\partial \epsilon}{\partial a}$$

$$\frac{\partial \epsilon}{\partial b} = \frac{2}{n}\sum_{i=1}^{n}(a_t x_i + b_t - y_i)$$

$$\rightarrow \quad b_{t+1} = b_t - \lambda\frac{\partial \epsilon}{\partial b}$$

2. Gradient descent (an iterative process)





Model prediction

$$y' = a_t x + b_t$$

Loss function (MSE)

$$\epsilon = \frac{1}{n}\sum_{i=1}^{n}(y_i - y_i')^2 = \frac{1}{n}\sum_{i=1}^{n}(a_t x + b_t - y)^2$$

Compute the gradient

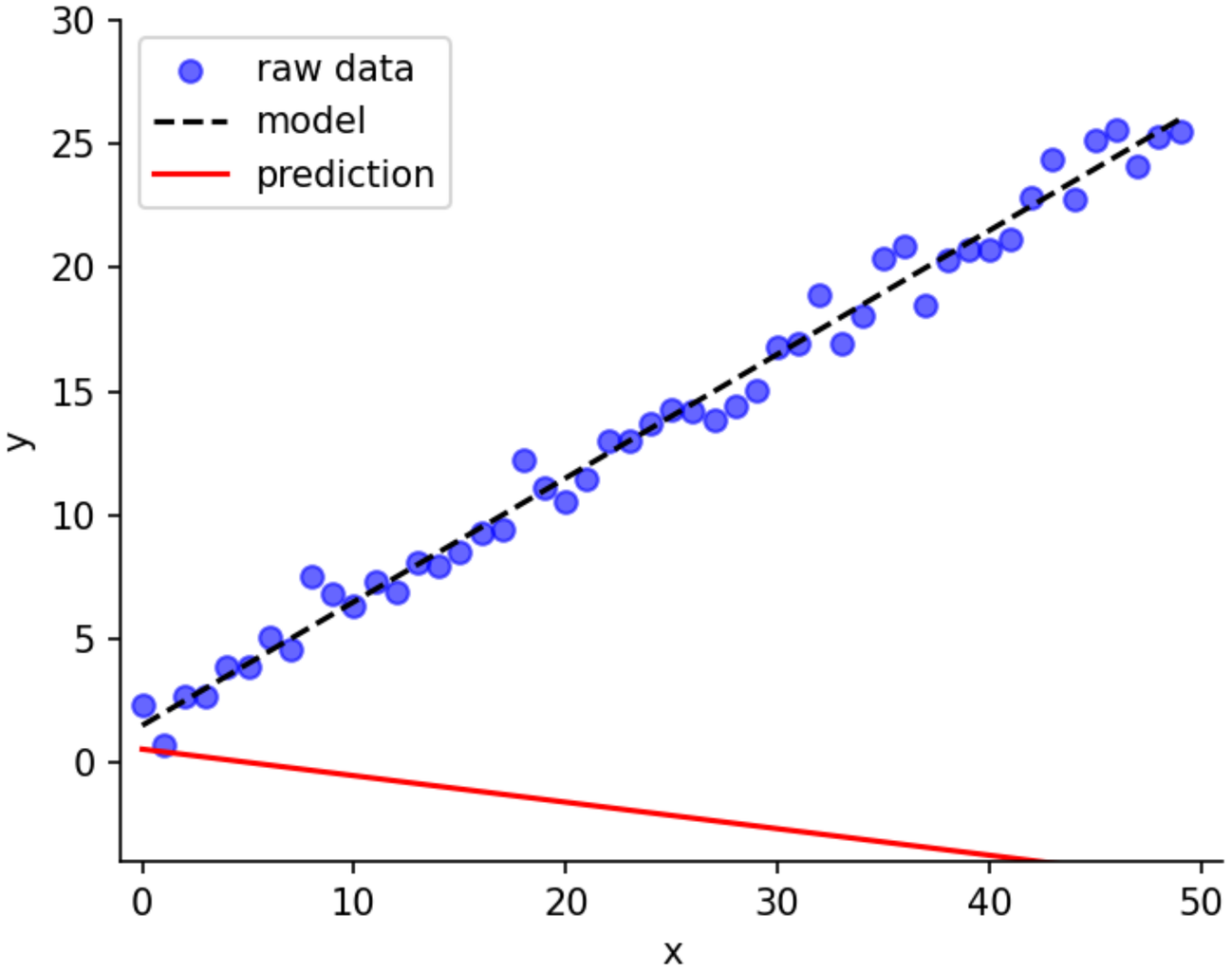
$$\frac{\partial \epsilon}{\partial a} = \frac{2}{n}\sum_{i=1}^{n}(a_t x_i + b_t - y_i)x_i$$

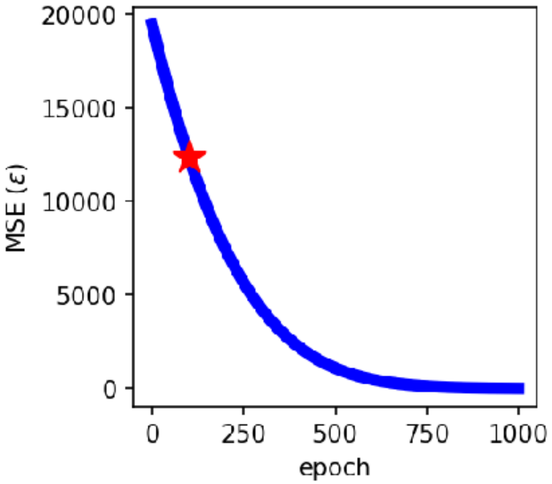
$$\rightarrow \quad a_{t+1} = a_t - \lambda\frac{\partial \epsilon}{\partial a}$$

$$\frac{\partial \epsilon}{\partial b} = \frac{2}{n}\sum_{i=1}^{n}(a_t x_i + b_t - y_i)$$

$$\rightarrow \quad b_{t+1} = b_t - \lambda\frac{\partial \epsilon}{\partial b}$$

2. Gradient descent (an iterative process)





Model prediction

$$y' = a_t x + b_t$$

Loss function (MSE)

$$\epsilon = \frac{1}{n}\sum_{i=1}^{n}(y_i - y_i')^2 = \frac{1}{n}\sum_{i=1}^{n}(a_t x + b_t - y)^2$$

Compute the gradient

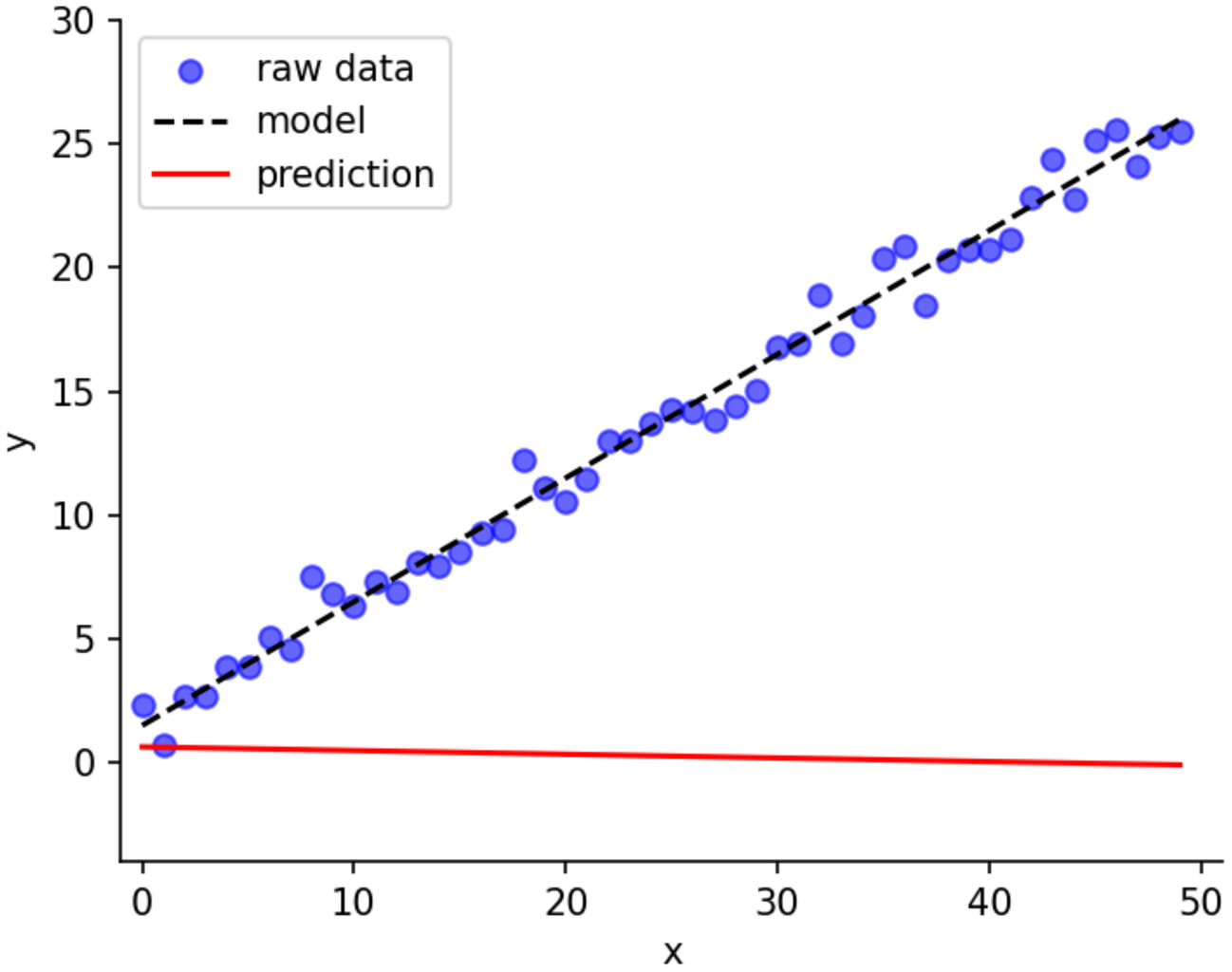
$$\frac{\partial \epsilon}{\partial a} = \frac{2}{n}\sum_{i=1}^{n}(a_t x_i + b_t - y_i)x_i$$

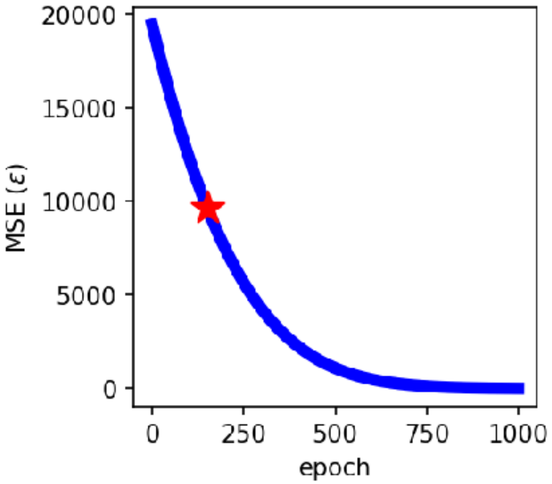
$$\rightarrow \quad a_{t+1} = a_t - \lambda\frac{\partial \epsilon}{\partial a}$$

$$\frac{\partial \epsilon}{\partial b} = \frac{2}{n}\sum_{i=1}^{n}(a_t x_i + b_t - y_i)$$

$$\rightarrow \quad b_{t+1} = b_t - \lambda\frac{\partial \epsilon}{\partial b}$$

2. Gradient descent (an iterative process)





Model prediction

$$y' = a_t x + b_t$$

Loss function (MSE)

$$\epsilon = \frac{1}{n}\sum_{i=1}^{n}(y_i - y_i')^2 = \frac{1}{n}\sum_{i=1}^{n}(a_t x + b_t - y)^2$$

Compute the gradient

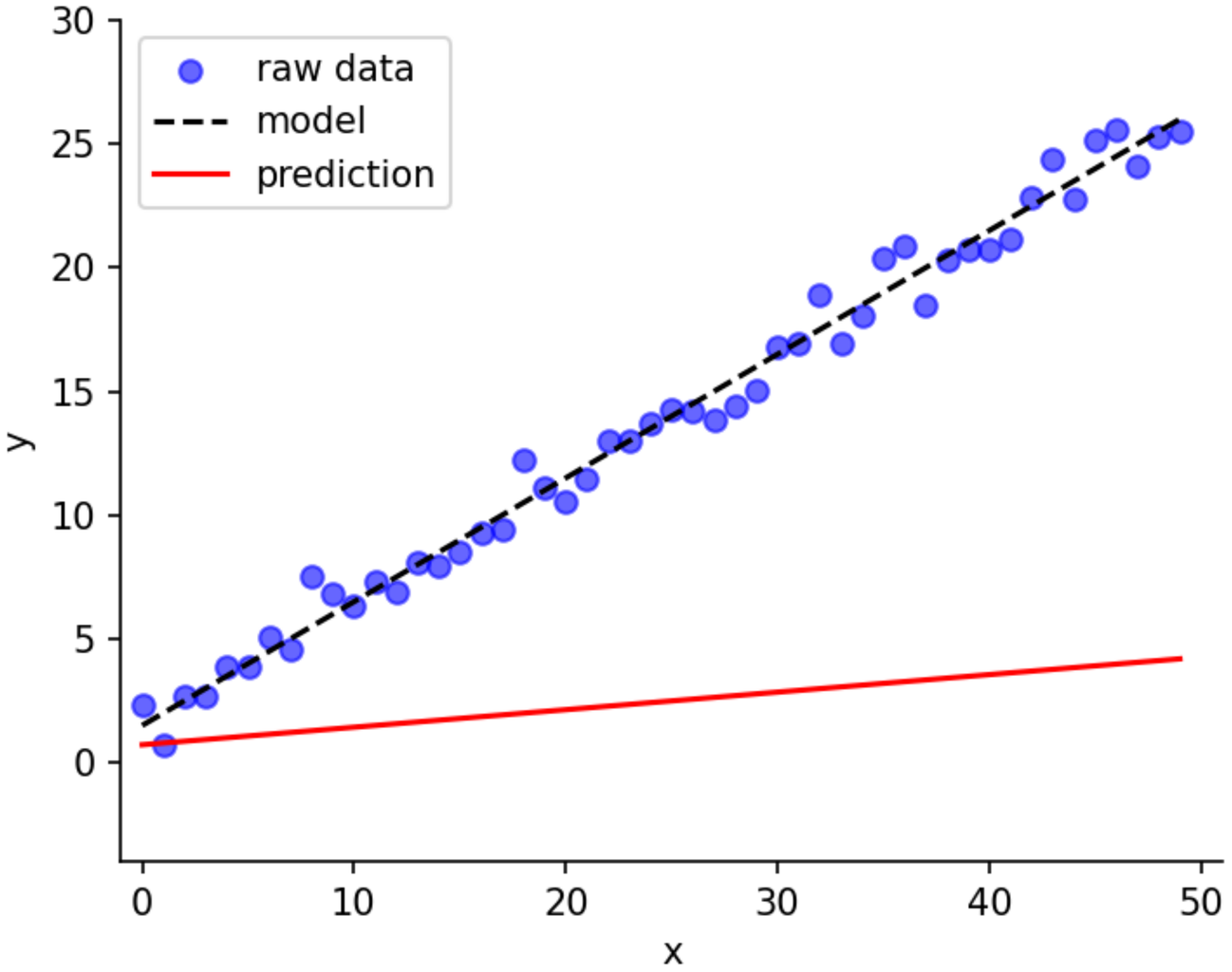
$$\frac{\partial \epsilon}{\partial a} = \frac{2}{n}\sum_{i=1}^{n}(a_t x_i + b_t - y_i)x_i$$

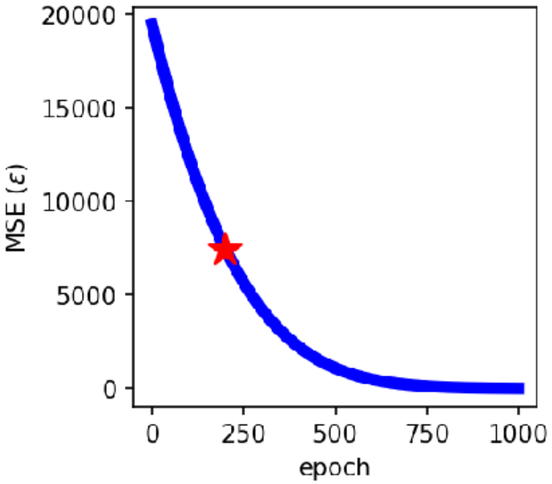
$$\rightarrow \quad a_{t+1} = a_t - \lambda\frac{\partial \epsilon}{\partial a}$$

$$\frac{\partial \epsilon}{\partial b} = \frac{2}{n}\sum_{i=1}^{n}(a_t x_i + b_t - y_i)$$

$$\rightarrow \quad b_{t+1} = b_t - \lambda\frac{\partial \epsilon}{\partial b}$$

2. Gradient descent (an iterative process)





Model prediction

$$y' = a_t x + b_t$$

Loss function (MSE)

$$\epsilon = \frac{1}{n}\sum_{i=1}^{n}(y_i - y_i')^2 = \frac{1}{n}\sum_{i=1}^{n}(a_t x + b_t - y)^2$$

Compute the gradient

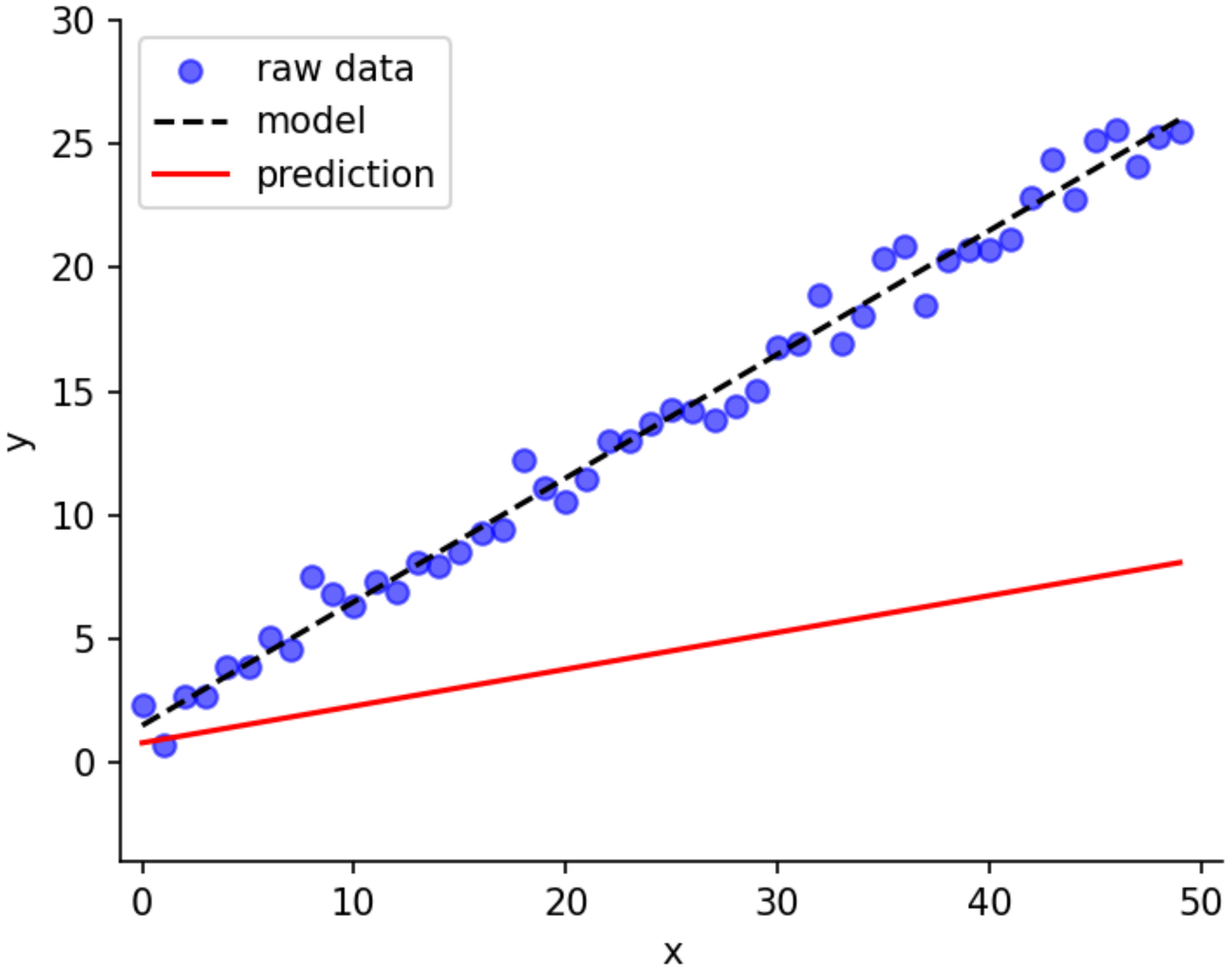
$$\frac{\partial \epsilon}{\partial a} = \frac{2}{n}\sum_{i=1}^{n}(a_t x_i + b_t - y_i)x_i$$

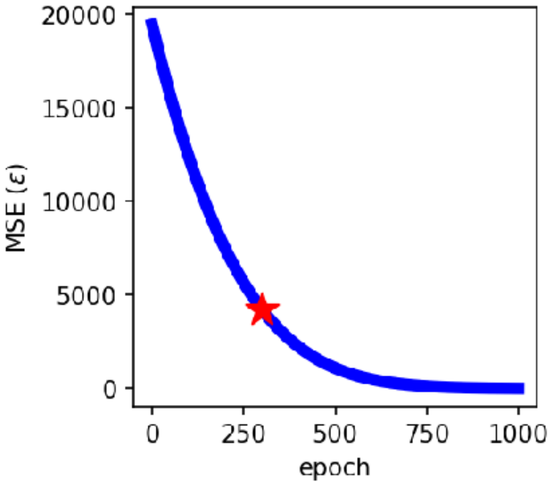
$$\rightarrow \quad a_{t+1} = a_t - \lambda\frac{\partial \epsilon}{\partial a}$$

$$\frac{\partial \epsilon}{\partial b} = \frac{2}{n}\sum_{i=1}^{n}(a_t x_i + b_t - y_i)$$

$$\rightarrow \quad b_{t+1} = b_t - \lambda\frac{\partial \epsilon}{\partial b}$$

2. Gradient descent (an iterative process)





Model prediction

$$y' = a_t x + b_t$$

Loss function (MSE)

$$\epsilon = \frac{1}{n}\sum_{i=1}^{n}(y_i - y_i')^2 = \frac{1}{n}\sum_{i=1}^{n}(a_t x + b_t - y)^2$$

Compute the gradient

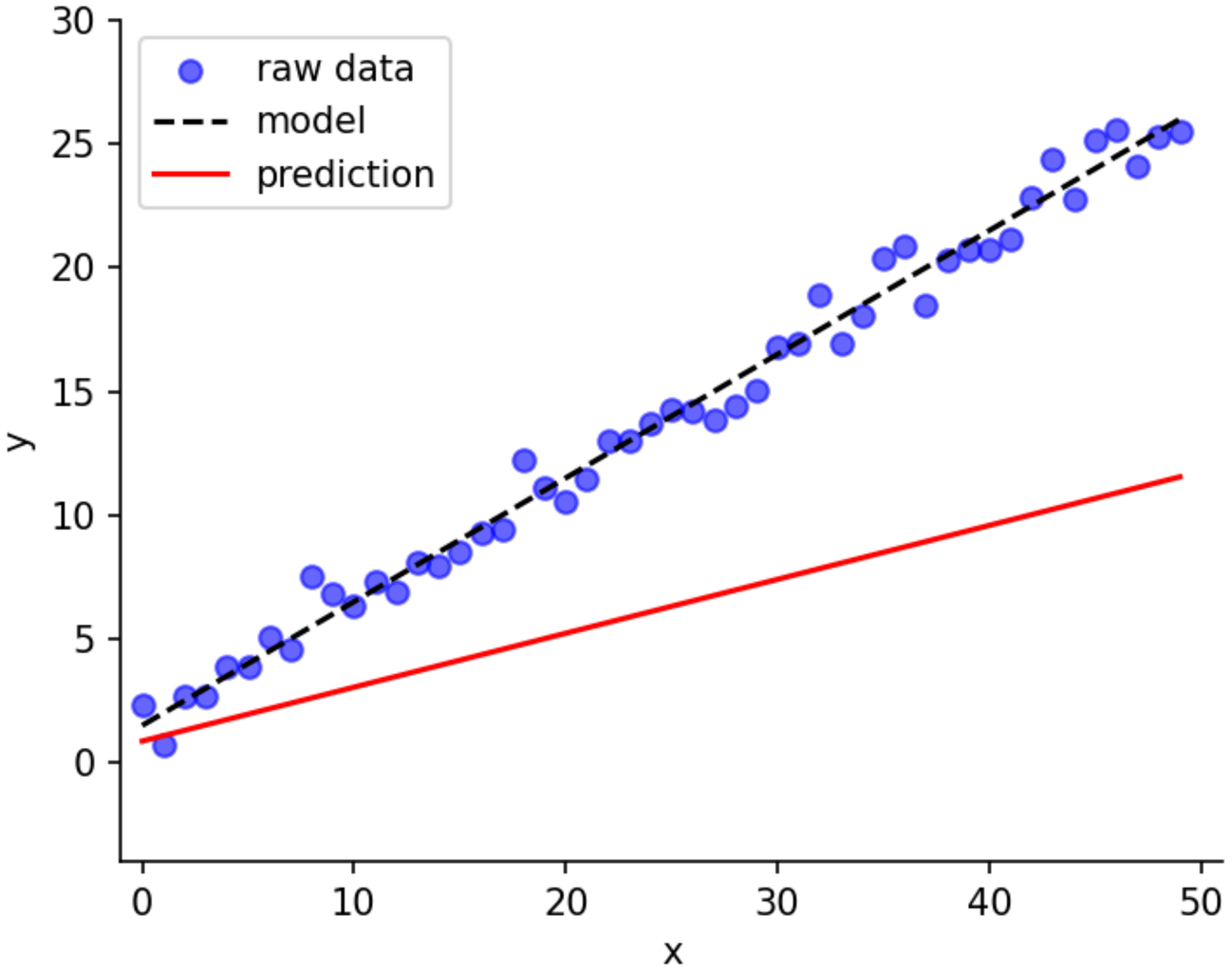
$$\frac{\partial \epsilon}{\partial a} = \frac{2}{n}\sum_{i=1}^{n}(a_t x_i + b_t - y_i)x_i$$

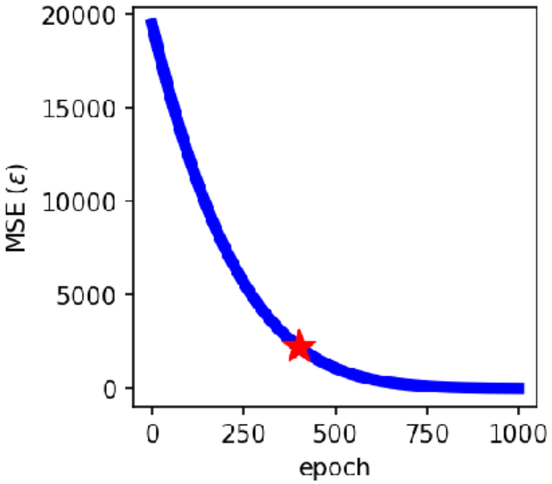
$$\rightarrow \quad a_{t+1} = a_t - \lambda\frac{\partial \epsilon}{\partial a}$$

$$\frac{\partial \epsilon}{\partial b} = \frac{2}{n}\sum_{i=1}^{n}(a_t x_i + b_t - y_i)$$

$$\rightarrow \quad b_{t+1} = b_t - \lambda\frac{\partial \epsilon}{\partial b}$$

2. Gradient descent (an iterative process)





Model prediction

$$y' = a_t x + b_t$$

Loss function (MSE)

$$\epsilon = \frac{1}{n}\sum_{i=1}^{n}(y_i - y_i')^2 = \frac{1}{n}\sum_{i=1}^{n}(a_t x + b_t - y)^2$$

Compute the gradient

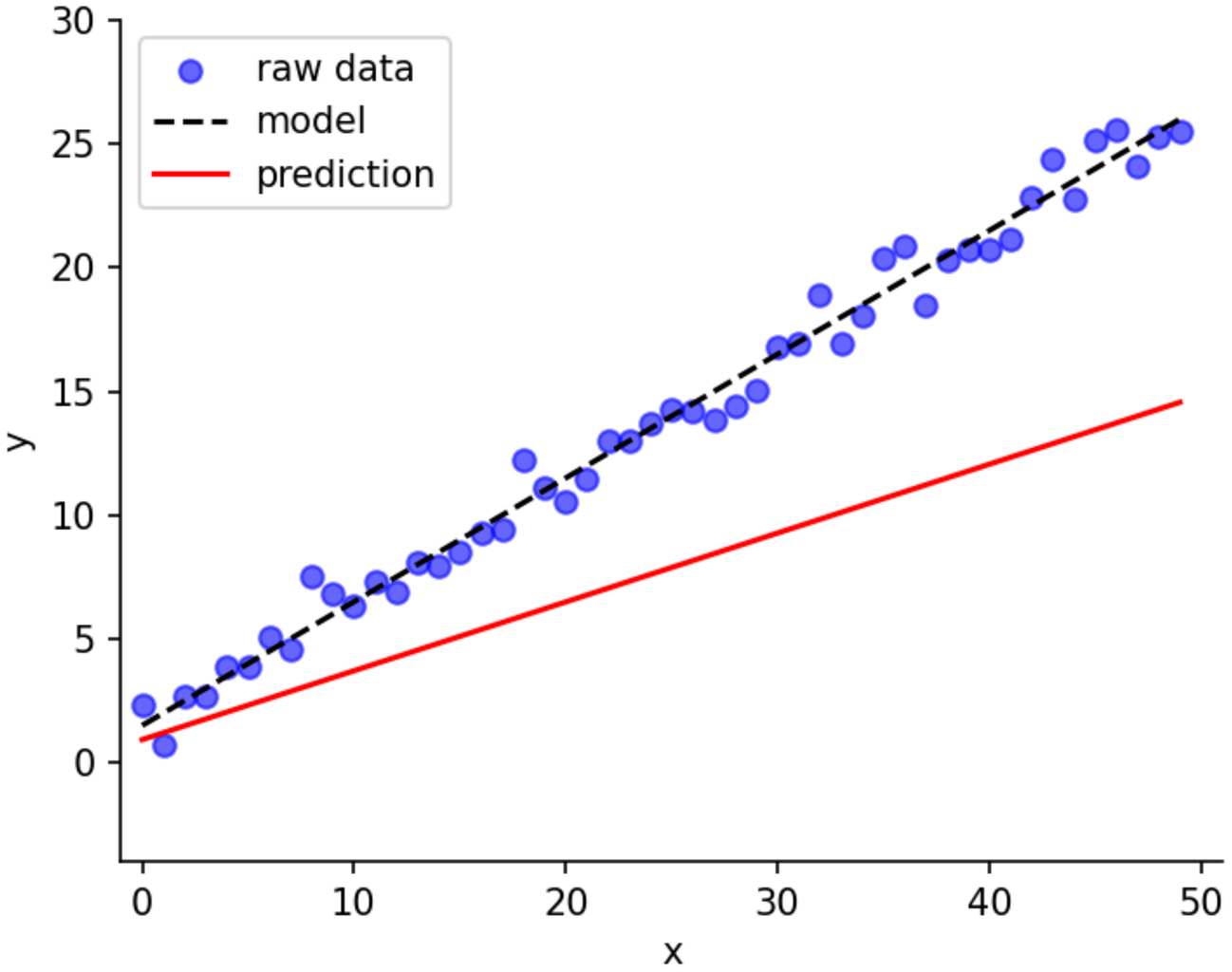
$$\frac{\partial \epsilon}{\partial a} = \frac{2}{n}\sum_{i=1}^{n}(a_t x_i + b_t - y_i)x_i$$

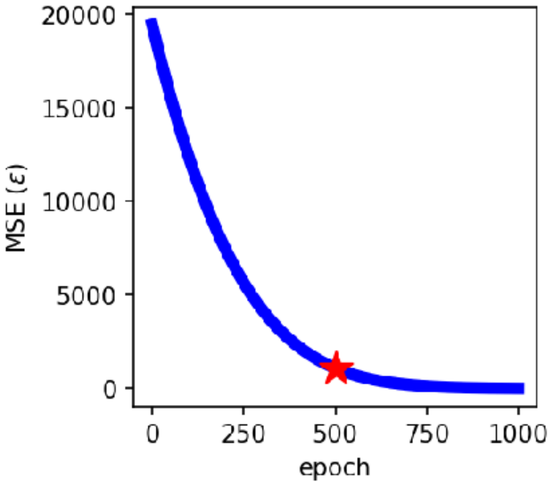
$$\rightarrow \quad a_{t+1} = a_t - \lambda\frac{\partial \epsilon}{\partial a}$$

$$\frac{\partial \epsilon}{\partial b} = \frac{2}{n}\sum_{i=1}^{n}(a_t x_i + b_t - y_i)$$

$$\rightarrow \quad b_{t+1} = b_t - \lambda\frac{\partial \epsilon}{\partial b}$$

2. Gradient descent (an iterative process)





Model prediction

$$y' = a_t x + b_t$$

Loss function (MSE)

$$\epsilon = \frac{1}{n}\sum_{i=1}^{n}(y_i - y_i')^2 = \frac{1}{n}\sum_{i=1}^{n}(a_t x + b_t - y)^2$$

Compute the gradient

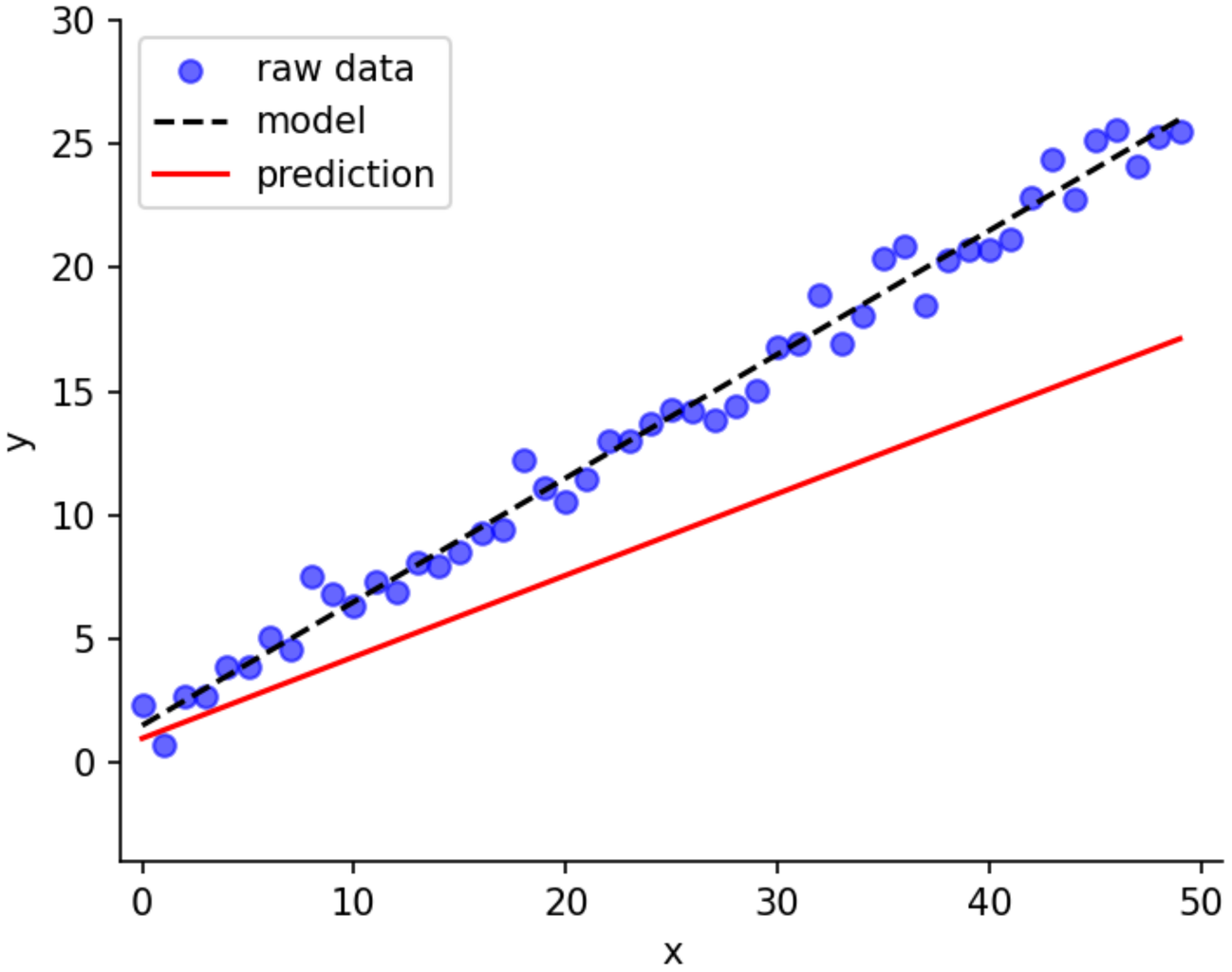
$$\frac{\partial \epsilon}{\partial a} = \frac{2}{n}\sum_{i=1}^{n}(a_t x_i + b_t - y_i)x_i$$

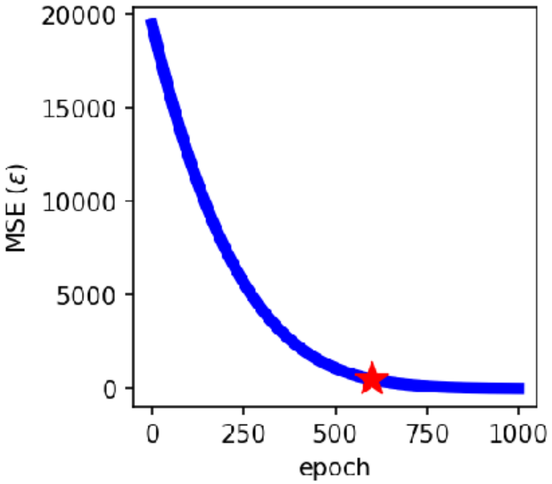
$$\rightarrow \quad a_{t+1} = a_t - \lambda\frac{\partial \epsilon}{\partial a}$$

$$\frac{\partial \epsilon}{\partial b} = \frac{2}{n}\sum_{i=1}^{n}(a_t x_i + b_t - y_i)$$

$$\rightarrow \quad b_{t+1} = b_t - \lambda\frac{\partial \epsilon}{\partial b}$$

2. Gradient descent (an iterative process)





Model prediction

$$y' = a_t x + b_t$$

Loss function (MSE)

$$\epsilon = \frac{1}{n}\sum_{i=1}^{n}(y_i - y_i')^2 = \frac{1}{n}\sum_{i=1}^{n}(a_t x + b_t - y)^2$$

Compute the gradient

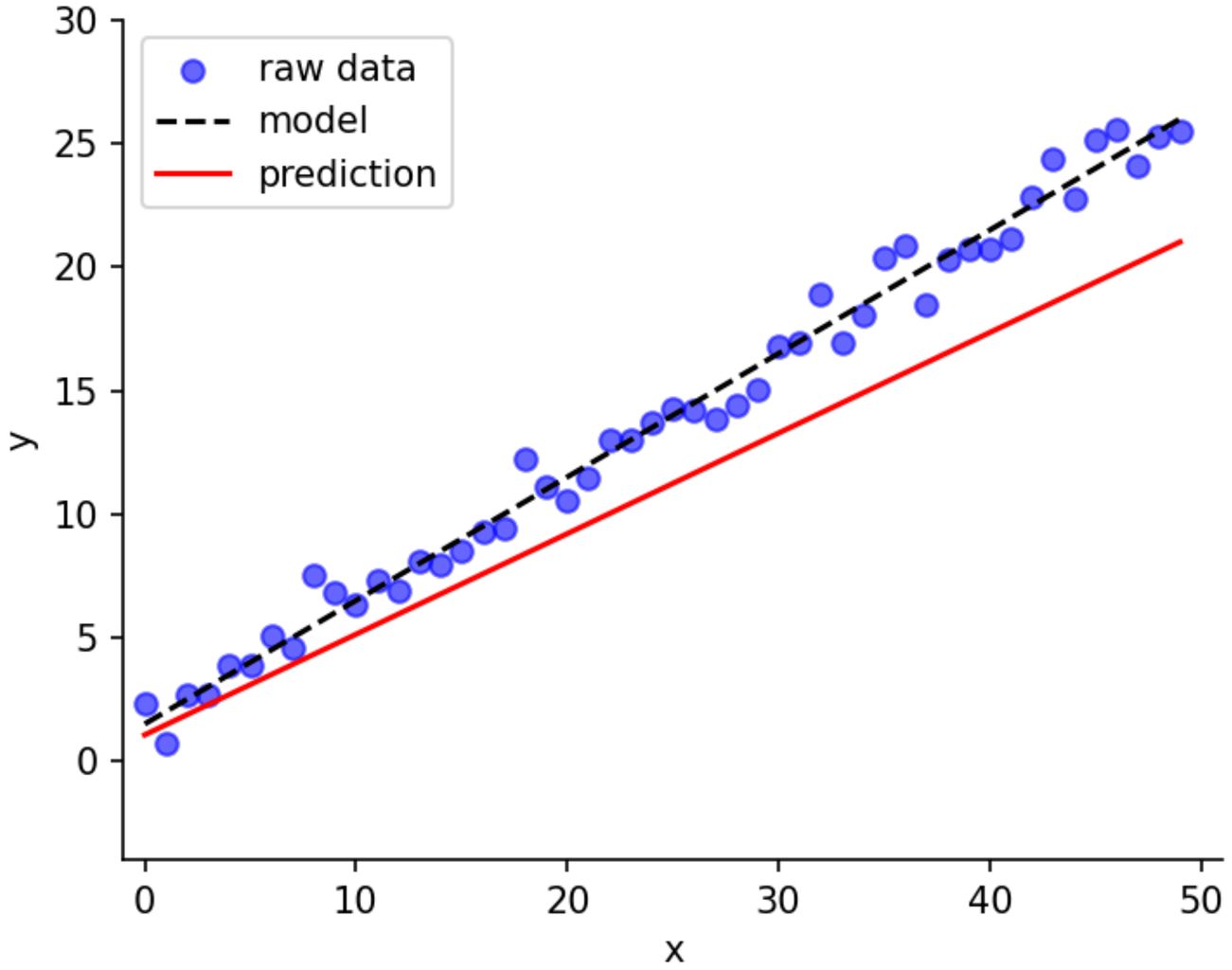
$$\frac{\partial \epsilon}{\partial a} = \frac{2}{n}\sum_{i=1}^{n}(a_t x_i + b_t - y_i)x_i$$

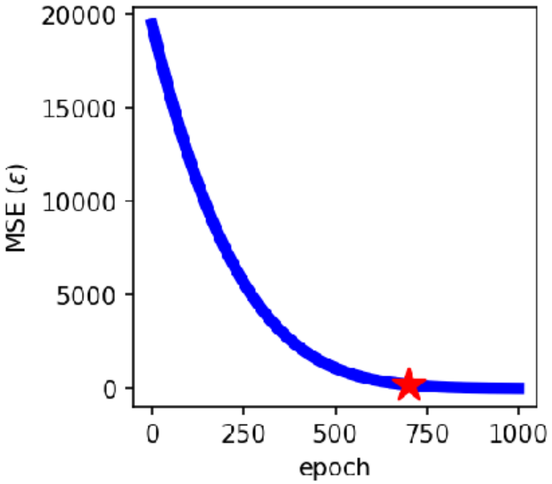
$$\rightarrow \quad a_{t+1} = a_t - \lambda\frac{\partial \epsilon}{\partial a}$$

$$\frac{\partial \epsilon}{\partial b} = \frac{2}{n}\sum_{i=1}^{n}(a_t x_i + b_t - y_i)$$

$$\rightarrow \quad b_{t+1} = b_t - \lambda\frac{\partial \epsilon}{\partial b}$$

2. Gradient descent (an iterative process)





Model prediction

$$y' = a_t x + b_t$$

Loss function (MSE)

$$\epsilon = \frac{1}{n}\sum_{i=1}^{n}(y_i - y_i')^2 = \frac{1}{n}\sum_{i=1}^{n}(a_t x + b_t - y)^2$$

Compute the gradient

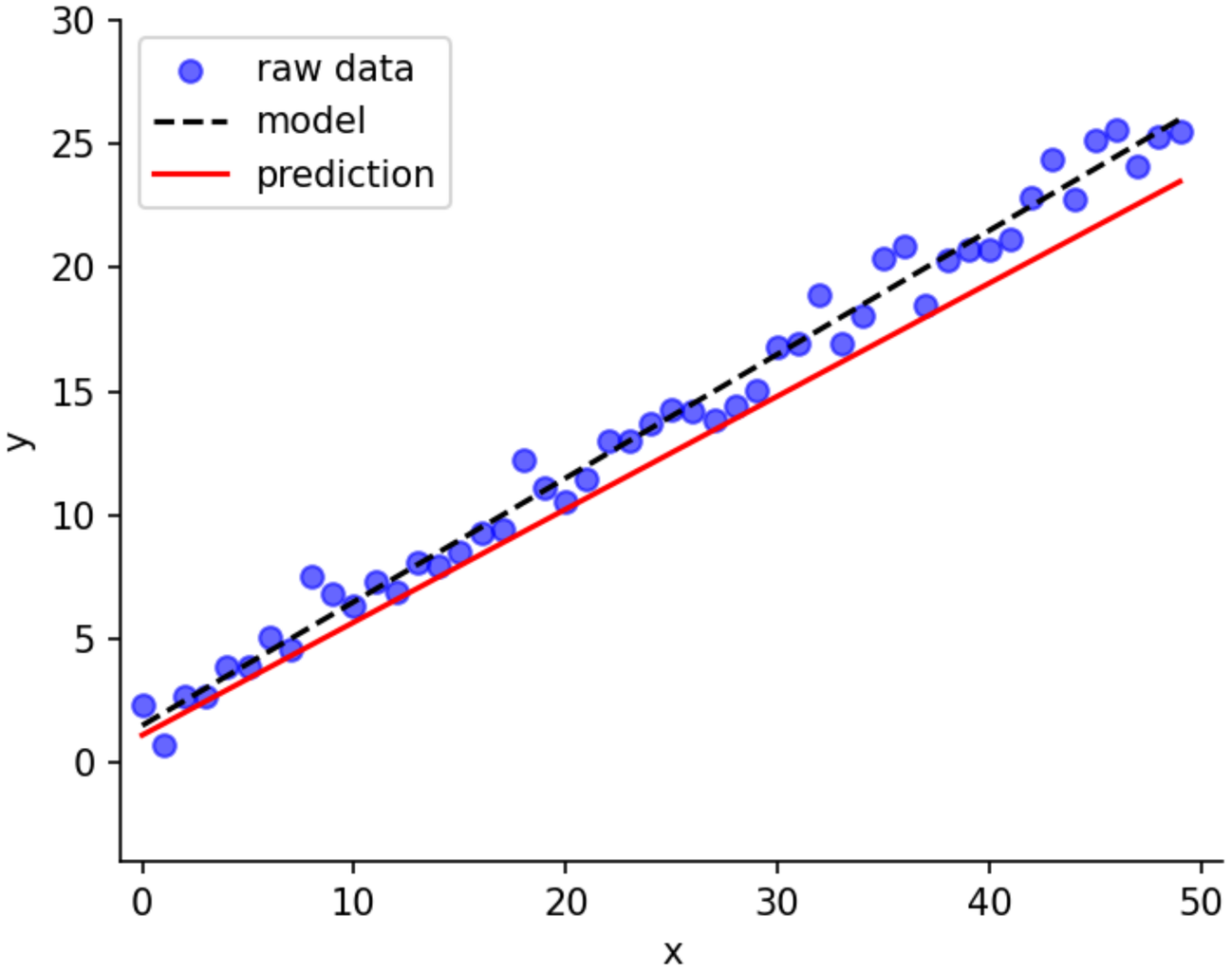
$$\frac{\partial \epsilon}{\partial a} = \frac{2}{n}\sum_{i=1}^{n}(a_t x_i + b_t - y_i)x_i$$

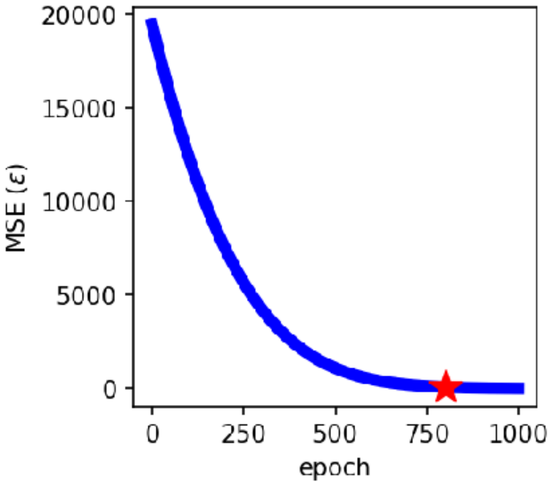
$$\rightarrow \quad a_{t+1} = a_t - \lambda\frac{\partial \epsilon}{\partial a}$$

$$\frac{\partial \epsilon}{\partial b} = \frac{2}{n}\sum_{i=1}^{n}(a_t x_i + b_t - y_i)$$

$$\rightarrow \quad b_{t+1} = b_t - \lambda\frac{\partial \epsilon}{\partial b}$$

2. Gradient descent (an iterative process)





Model prediction

$$y' = a_t x + b_t$$

Loss function (MSE)

$$\epsilon = \frac{1}{n}\sum_{i=1}^{n}(y_i - y_i')^2 = \frac{1}{n}\sum_{i=1}^{n}(a_t x + b_t - y)^2$$

Compute the gradient

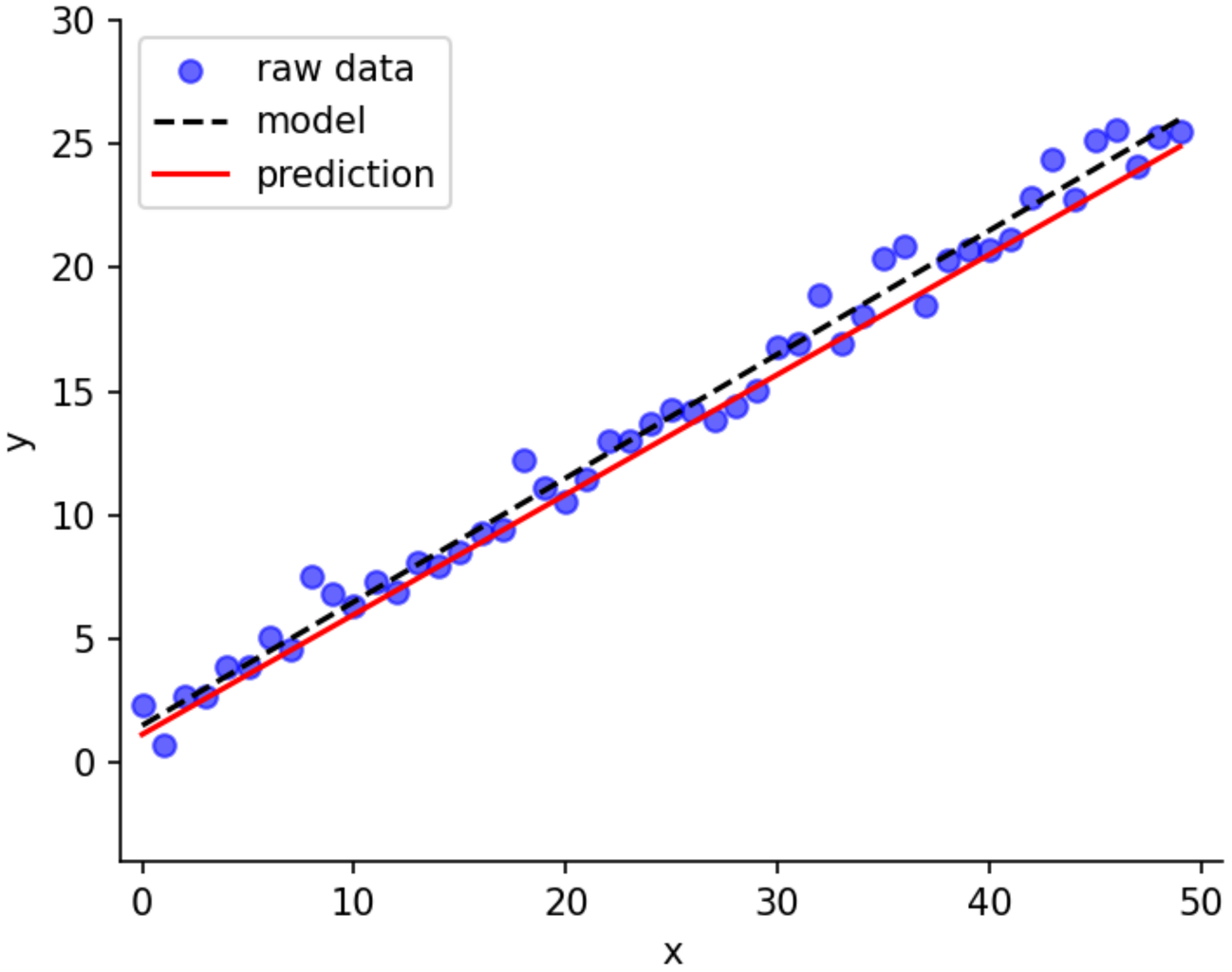
$$\frac{\partial \epsilon}{\partial a} = \frac{2}{n}\sum_{i=1}^{n}(a_t x_i + b_t - y_i)x_i$$

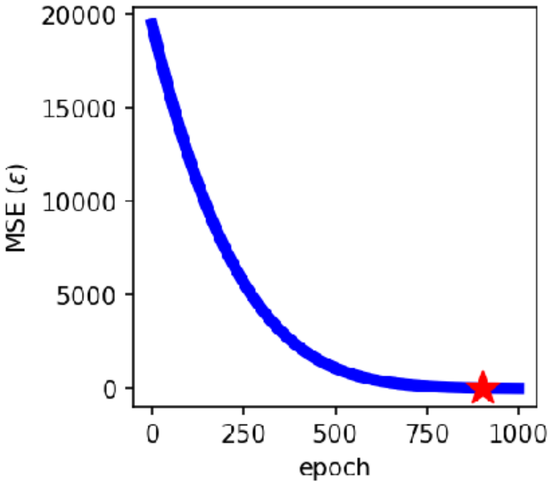
$$\rightarrow \quad a_{t+1} = a_t - \lambda\frac{\partial \epsilon}{\partial a}$$

$$\frac{\partial \epsilon}{\partial b} = \frac{2}{n}\sum_{i=1}^{n}(a_t x_i + b_t - y_i)$$

$$\rightarrow \quad b_{t+1} = b_t - \lambda\frac{\partial \epsilon}{\partial b}$$

2. Gradient descent (an iterative process)





Model prediction

$$y' = a_t x + b_t$$

Loss function (MSE)

$$\epsilon = \frac{1}{n}\sum_{i=1}^{n}(y_i - y_i')^2 = \frac{1}{n}\sum_{i=1}^{n}(a_t x + b_t - y)^2$$

Compute the gradient

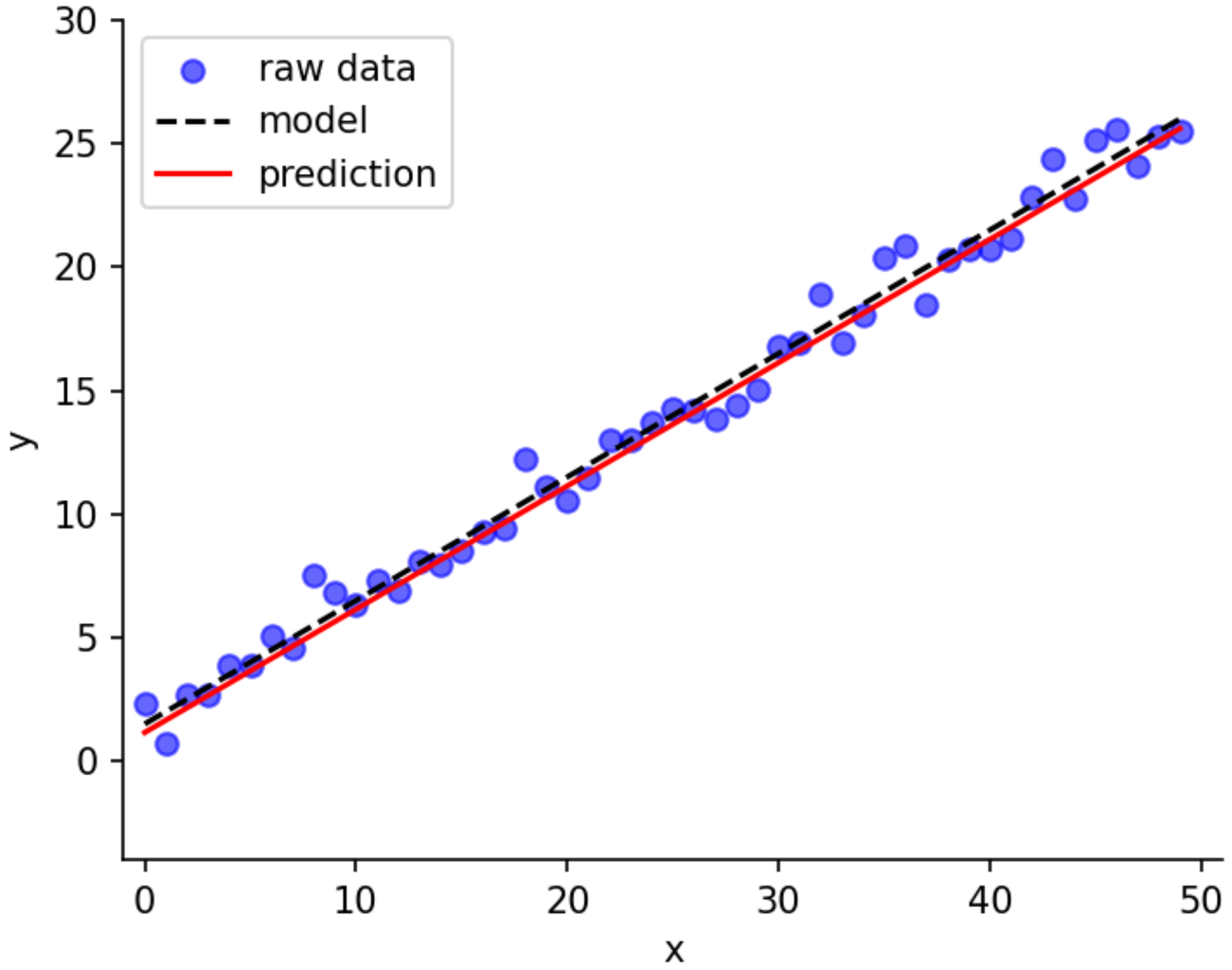
$$\frac{\partial \epsilon}{\partial a} = \frac{2}{n}\sum_{i=1}^{n}(a_t x_i + b_t - y_i)x_i$$

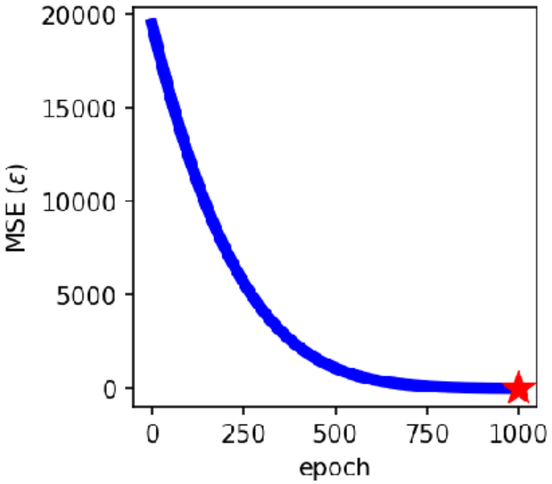
$$\rightarrow \quad a_{t+1} = a_t - \lambda\frac{\partial \epsilon}{\partial a}$$

$$\frac{\partial \epsilon}{\partial b} = \frac{2}{n}\sum_{i=1}^{n}(a_t x_i + b_t - y_i)$$

$$\rightarrow \quad b_{t+1} = b_t - \lambda\frac{\partial \epsilon}{\partial b}$$

2. Gradient descent (an iterative process)





Model prediction

$$y' = a_t x + b_t$$

Loss function (MSE)

$$\epsilon = \frac{1}{n}\sum_{i=1}^{n}(y_i - y_i')^2 = \frac{1}{n}\sum_{i=1}^{n}(a_t x + b_t - y)^2$$

Compute the gradient

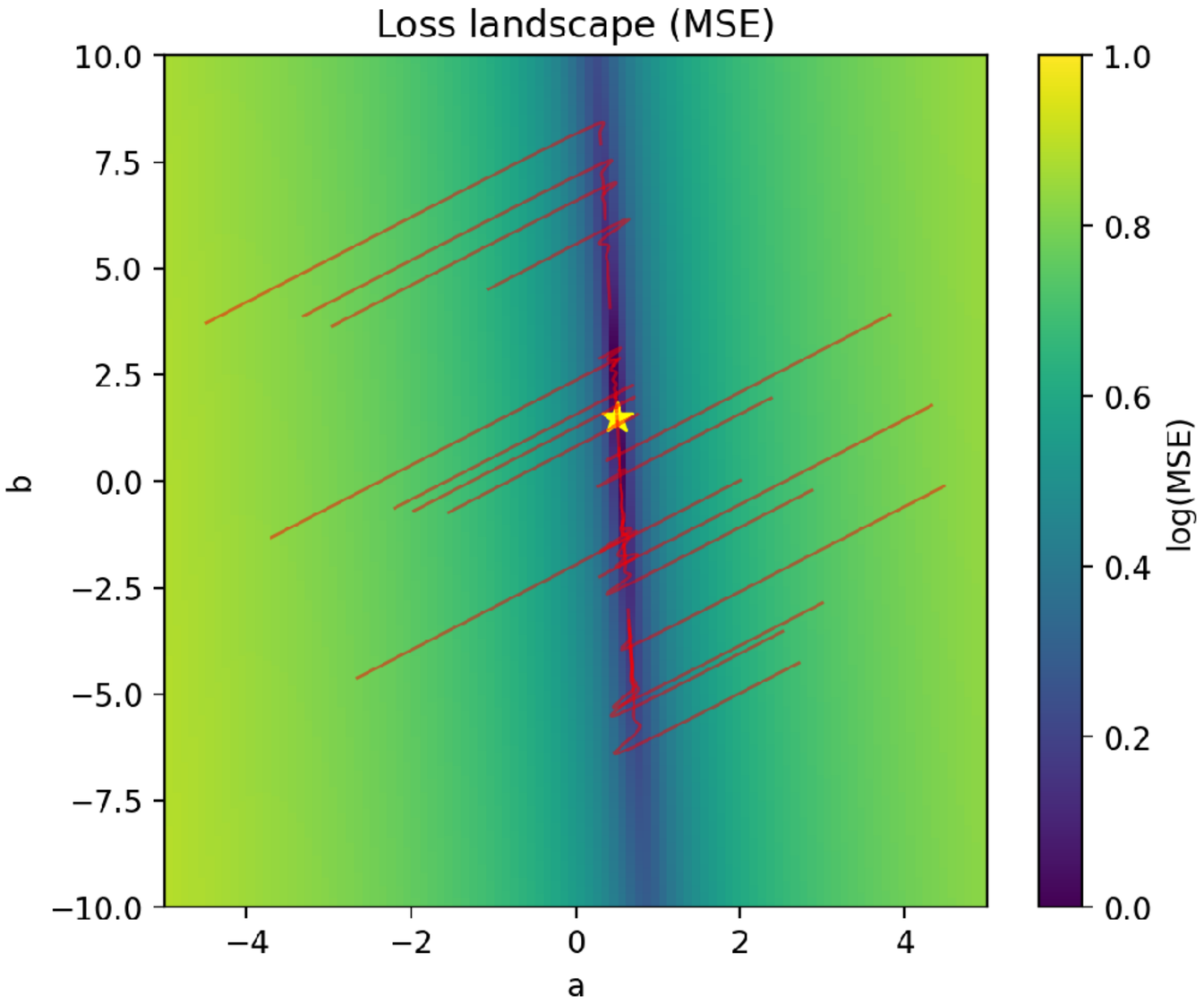
$$\frac{\partial \epsilon}{\partial a} = \frac{2}{n}\sum_{i=1}^{n}(a_t x_i + b_t - y_i)x_i$$

$$\rightarrow \quad a_{t+1} = a_t - \lambda\frac{\partial \epsilon}{\partial a}$$

$$\frac{\partial \epsilon}{\partial b} = \frac{2}{n}\sum_{i=1}^{n}(a_t x_i + b_t - y_i)$$

$$\rightarrow \quad b_{t+1} = b_t - \lambda\frac{\partial \epsilon}{\partial b}$$

2. Gradient descent (an iterative process)



Model prediction

$$y' = a_t x + b_t$$

Loss function (MSE)

$$\epsilon = \frac{1}{n}\sum_{i=1}^{n}(y_i - y_i')^2 = \frac{1}{n}\sum_{i=1}^{n}(a_t x + b_t - y)^2$$

Compute the gradient

$$\frac{\partial \epsilon}{\partial a} = \frac{2}{n}\sum_{i=1}^{n}(a_t x_i + b_t - y_i)x_i$$

$$\rightarrow \quad a_{t+1} = a_t - \lambda\frac{\partial \epsilon}{\partial a}$$

$$\frac{\partial \epsilon}{\partial b} = \frac{2}{n}\sum_{i=1}^{n}(a_t x_i + b_t - y_i)$$

$$\rightarrow \quad b_{t+1} = b_t - \lambda\frac{\partial \epsilon}{\partial b}$$

2. Gradient descent (an iterative process)

2. Gradient descent (an iterative process)

PyTorch

- Automatic differentiation (Autograd)
- More than 25 Loss already implemented (MSE, Kullback–Leibler, L1…)
- Around 15 optimizing algorithm (SGD, Adam…)
- Easy debugging
- CPU & GPU support

3. Few examples

3. Few examples

Also largely used in public research :

3. Few examples

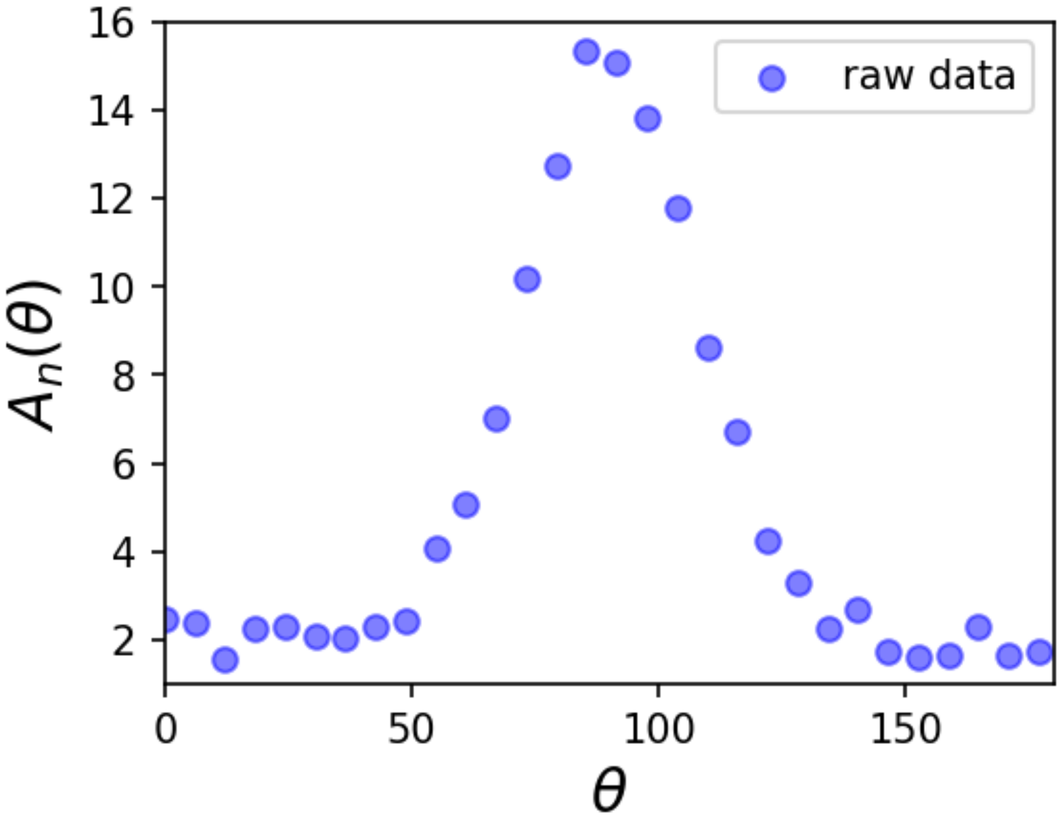
Also largely used in public research :

Fitting model parameters

3. Few examples

Also largely used in public research :

Fitting model parameters

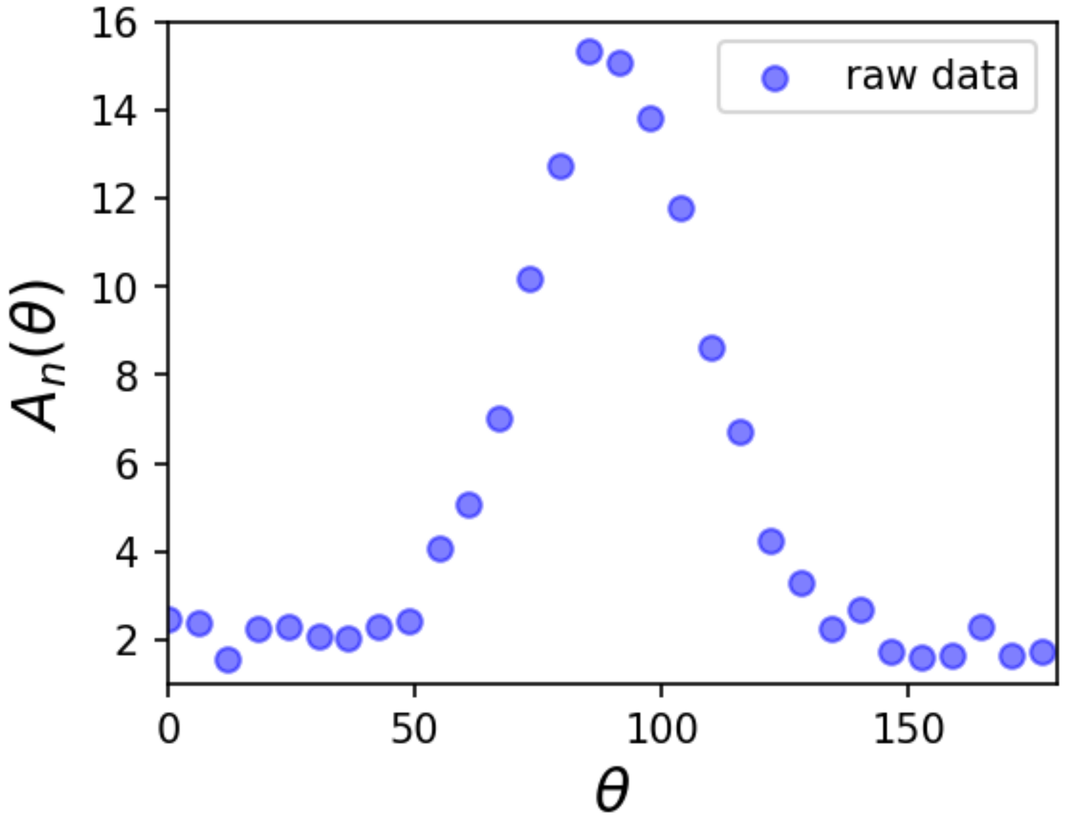


3. Few examples

Also largely used in public research :

$$A_n(\theta) = \frac{1}{\sigma\sqrt{2\pi}} \times \exp\left(\frac{-(\theta-\theta_0)^2}{2\sigma^2}\right)$$

Fitting model parameters

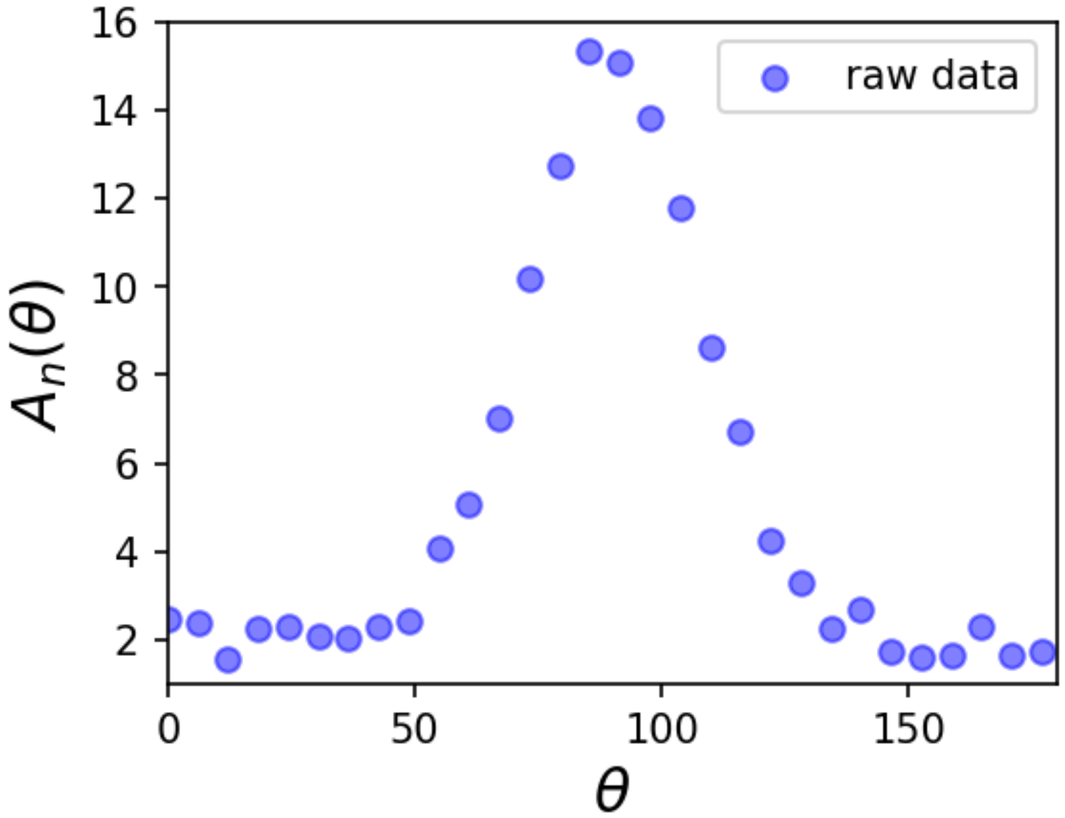


3. Few examples

Also largely used in public research :

$$A_n(\theta) = \frac{1}{\sigma\sqrt{2\pi}} \times \exp\left(\frac{-(\theta-\theta_0)^2}{2\sigma^2}\right)$$

Fitting model parameters

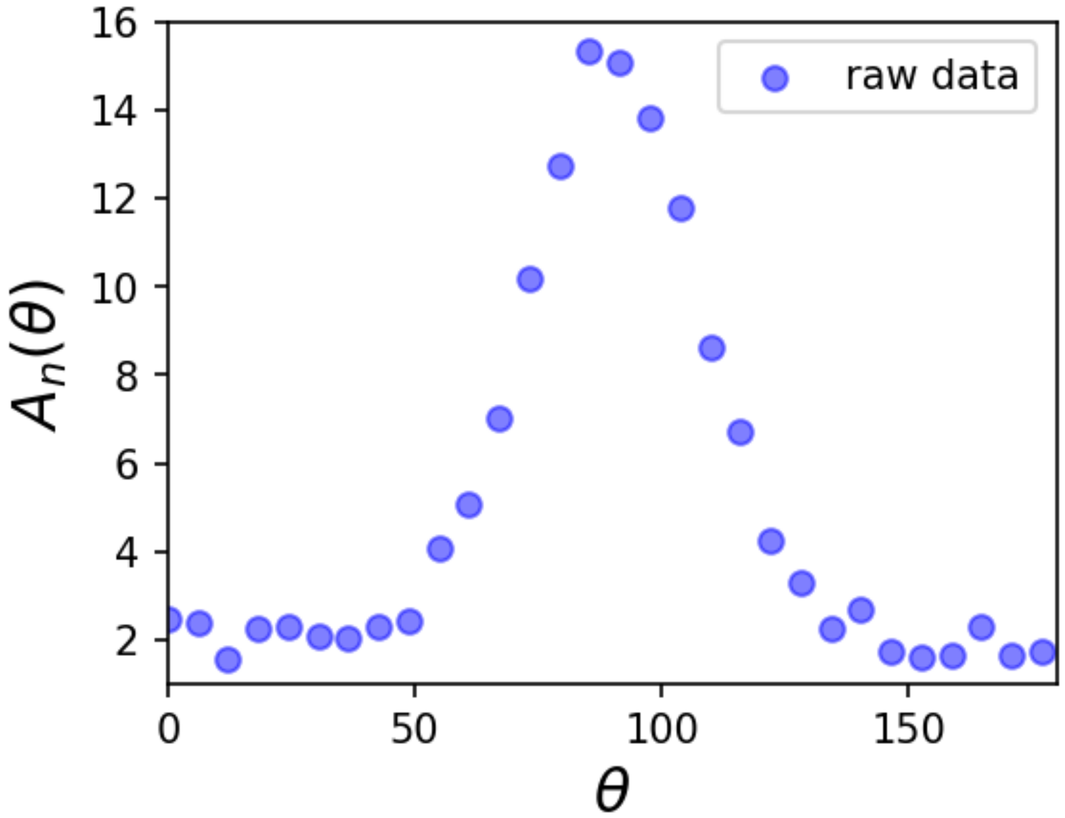


3. Few examples

Also largely used in public research :

$$A_n(\theta) = \frac{1}{\sigma\sqrt{2\pi}} \times \exp\left(\frac{-(\theta-\theta_0)^2}{2\sigma^2}\right)$$

<u>Fitting model parameters</u>



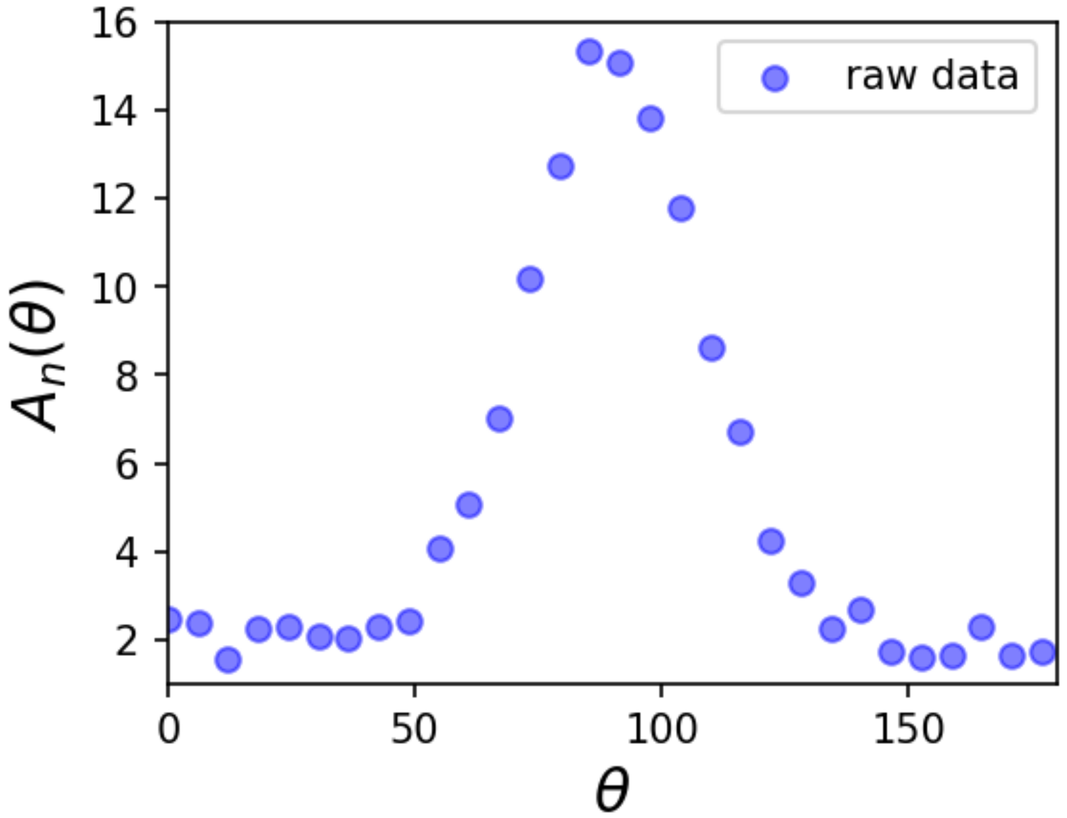

```python
class TorchModel(torch.nn.Module):
    def __init__(
            self,
            R_0,
            R_amp,
            sigma,
            theta_0
            ):
        super(TorchModel, self).__init__()

        self.R_0 = torch.nn.Parameter(torch.ones(1) * R_0)
        self.R_amp = torch.nn.Parameter(torch.ones(1) * R_amp)
        self.sigma = torch.nn.Parameter(torch.ones(1) * sigma)
        self.theta_0 = torch.nn.Parameter(torch.ones(1) * theta_0)

    def forward(self, theta):

        base_activity = torch.exp((- (theta - self.theta_0) ** 2) / (2 * self.sigma ** 2))
        modulation = (self.R_amp / (self.sigma * torch.sqrt(torch.Tensor([2 * torch.pi]))))

        return self.R_0 + modulation * base_activity
```

3. Few examples

Also largely used in public research :

$$A_n(\theta) = \frac{1}{\sigma\sqrt{2\pi}} \times \exp\left(\frac{-(\theta-\theta_0)^2}{2\sigma^2}\right)$$

Fitting model parameters




```python
class TorchModel(torch.nn.Module):
    def __init__(
            self,
            R_0,
            R_amp,
            sigma,
            theta_0
            ):
        super(TorchModel, self).__init__()

        self.R_0 = torch.nn.Parameter(torch.ones(1) * R_0)
        self.R_amp = torch.nn.Parameter(torch.ones(1) * R_amp)
        self.sigma = torch.nn.Parameter(torch.ones(1) * sigma)
        self.theta_0 = torch.nn.Parameter(torch.ones(1) * theta_0)

    def forward(self, theta):

        base_activity = torch.exp((- (theta - self.theta_0) ** 2) / (2 * self.sigma ** 2))
        modulation = (self.R_amp / (self.sigma * torch.sqrt(torch.Tensor([2 * torch.pi]))))

        return self.R_0 + modulation * base_activity
```

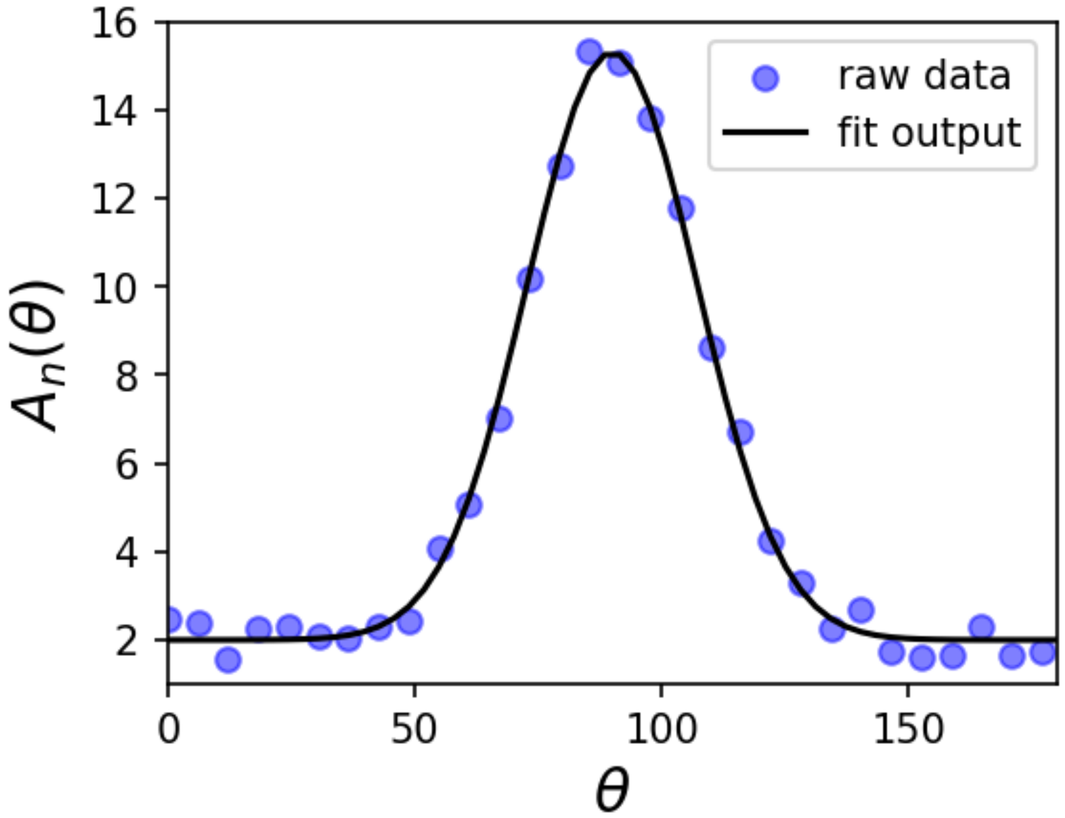
Training loop /
Loss minimization

3. Few examples

Also largely used in public research :

$$A_n(\theta) = \frac{1}{\sigma\sqrt{2\pi}} \times \exp\left(\frac{-(\theta-\theta_0)^2}{2\sigma^2}\right)$$

Fitting model parameters




```python
class TorchModel(torch.nn.Module):
    def __init__(
            self,
            R_0,
            R_amp,
            sigma,
            theta_0
            ):
        super(TorchModel, self).__init__()

        self.R_0 = torch.nn.Parameter(torch.ones(1) * R_0)
        self.R_amp = torch.nn.Parameter(torch.ones(1) * R_amp)
        self.sigma = torch.nn.Parameter(torch.ones(1) * sigma)
        self.theta_0 = torch.nn.Parameter(torch.ones(1) * theta_0)

    def forward(self, theta):

        base_activity = torch.exp((- (theta - self.theta_0) ** 2) / (2 * self.sigma ** 2))
        modulation = (self.R_amp / (self.sigma * torch.sqrt(torch.Tensor([2 * torch.pi]))))

        return self.R_0 + modulation * base_activity
```

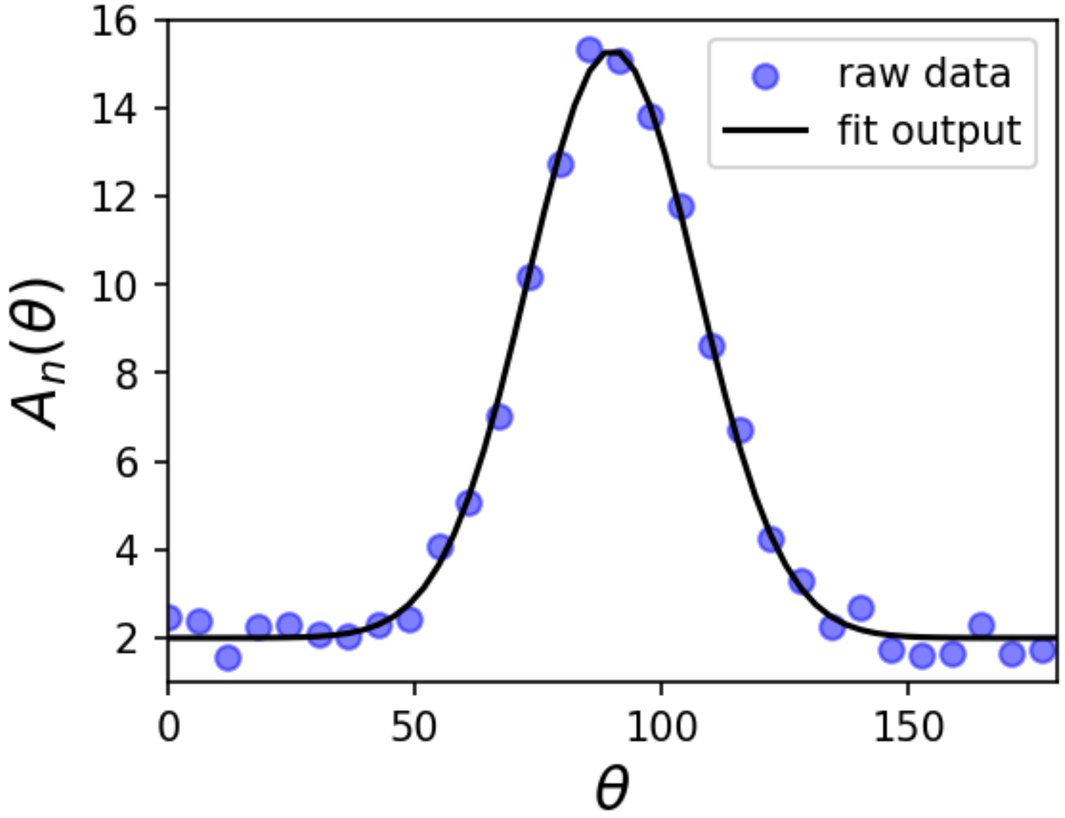
Training loop /
Loss minimization

3. Few examples

Also largely used in public research :

$$A_n(\theta) = \frac{1}{\sigma\sqrt{2\pi}} \times \exp\left(\frac{-(\theta-\theta_0)^2}{2\sigma^2}\right)$$

Fitting model parameters

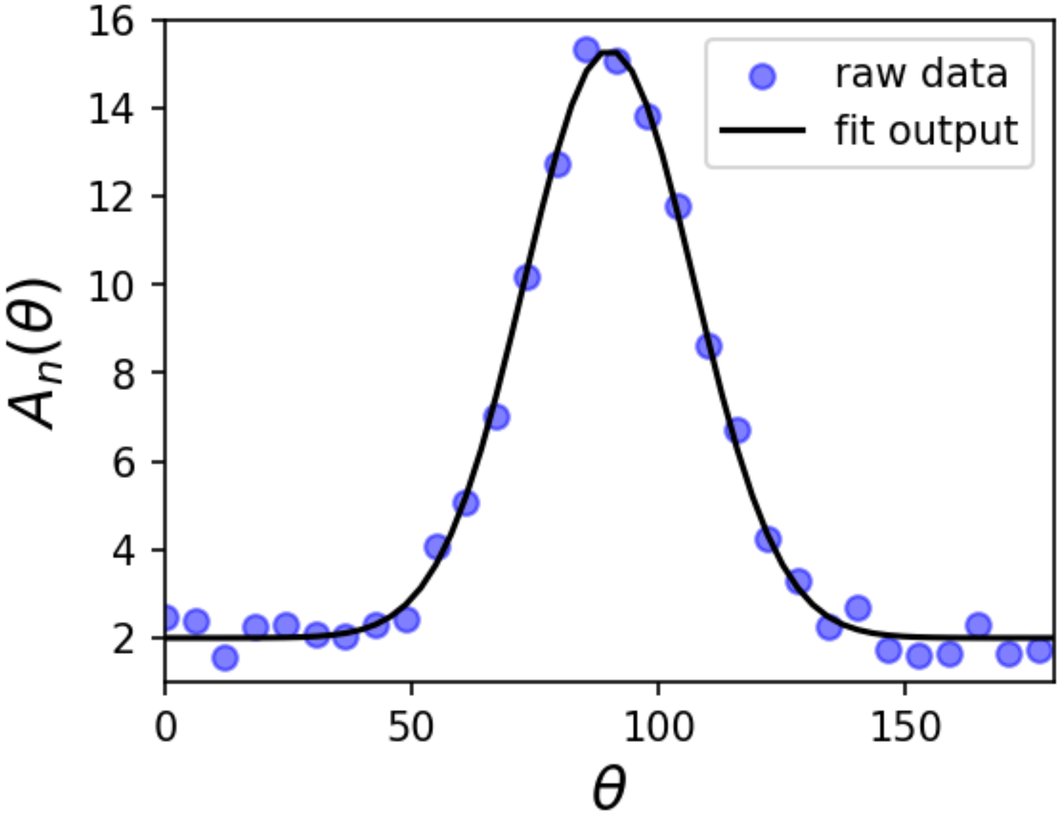


3. Few examples

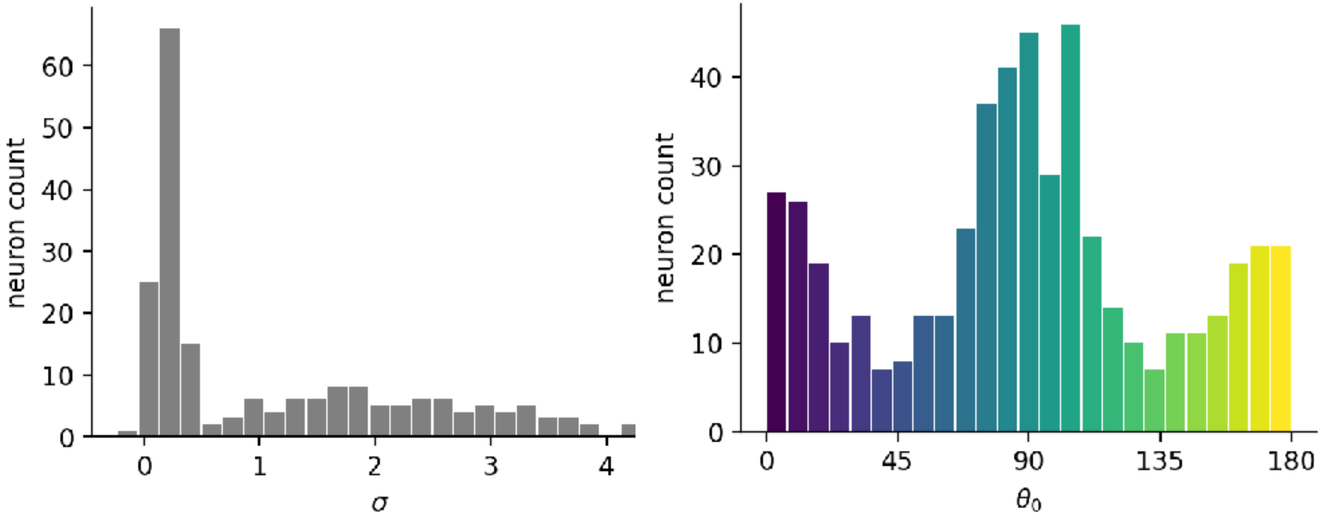
Also largely used in public research :

$$A_n(\theta) = \frac{1}{\sigma\sqrt{2\pi}} \times \exp\left(\frac{-(\theta-\theta_0)^2}{2\sigma^2}\right)$$

Fitting model parameters



Parameter extraction

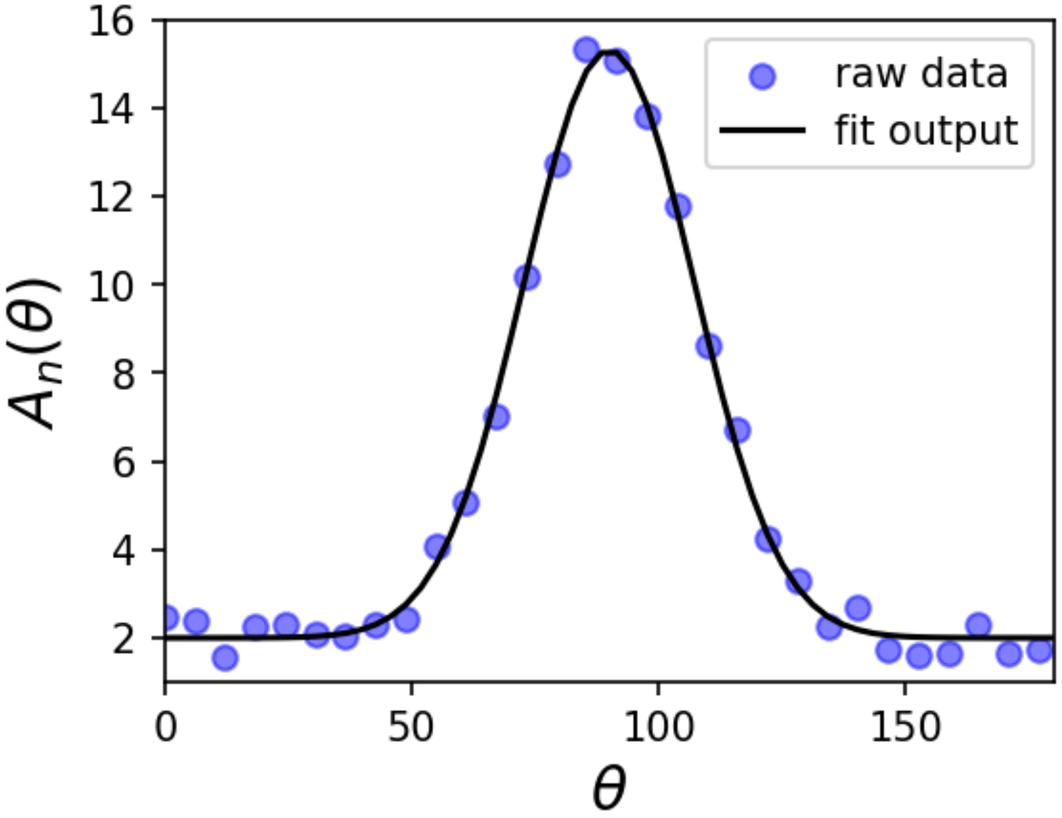


3. Few examples

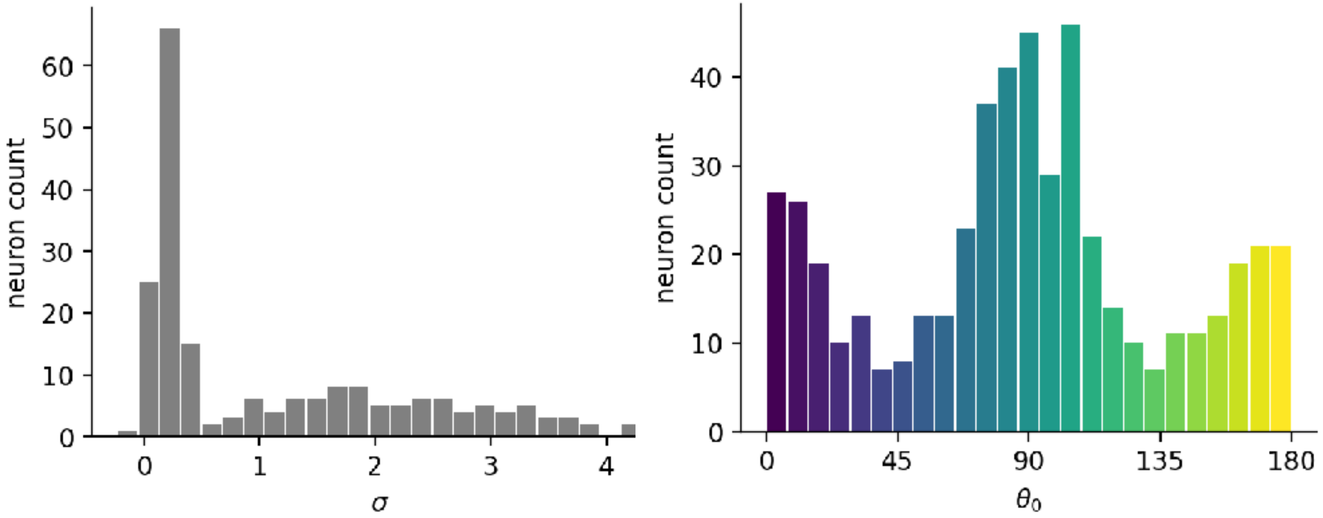
Also largely used in public research :

$$A_n(\theta) = \frac{1}{\sigma\sqrt{2\pi}} \times \exp\left(\frac{-(\theta-\theta_0)^2}{2\sigma^2}\right)$$

Fitting model parameters



Parameter extraction



- Simplify the analysis of neuronal properties
- Compare neuronal populations (V1 vs V2)
- Identify the best tuning model (Gaussian…)

3. Few examples

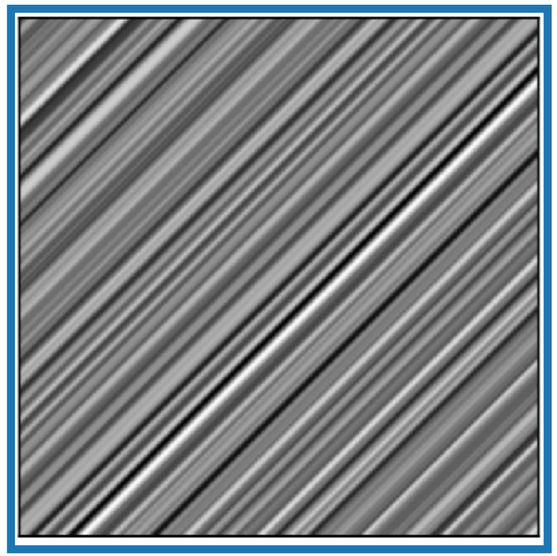
Also largely used in public research :

Classification for decoding

3. Few examples

Also largely used in public research :

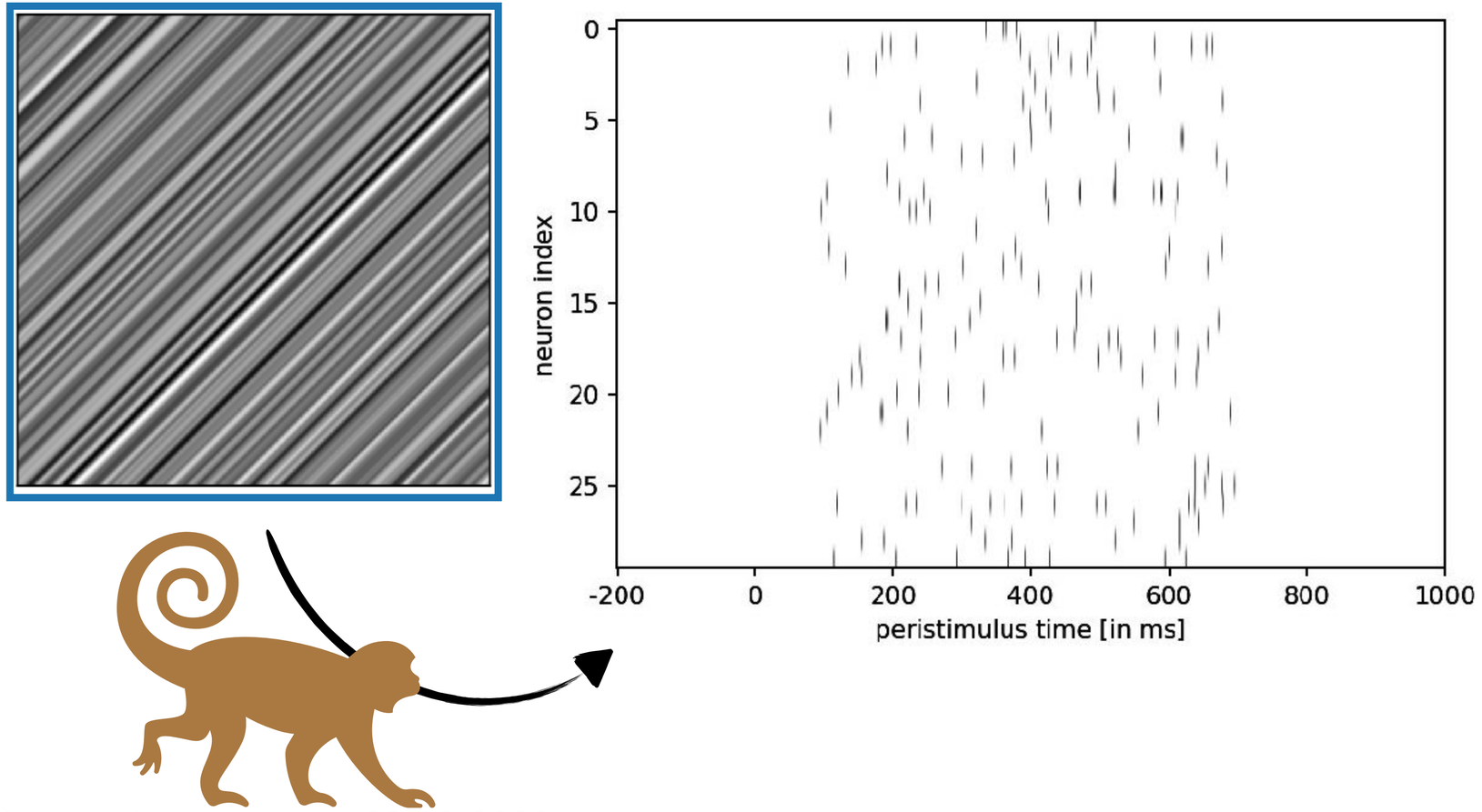
Classification for decoding



3. Few examples

Also largely used in public research :

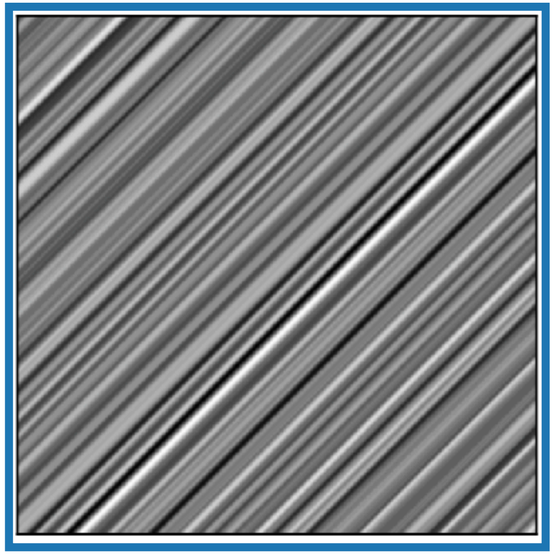
Classification for decoding

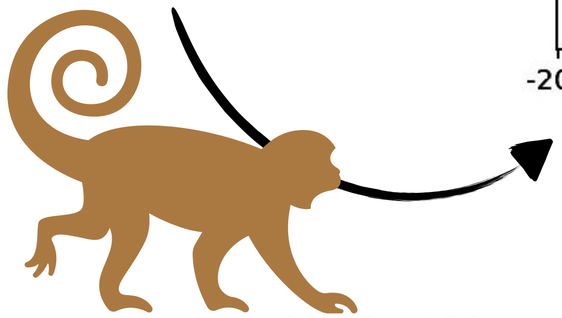


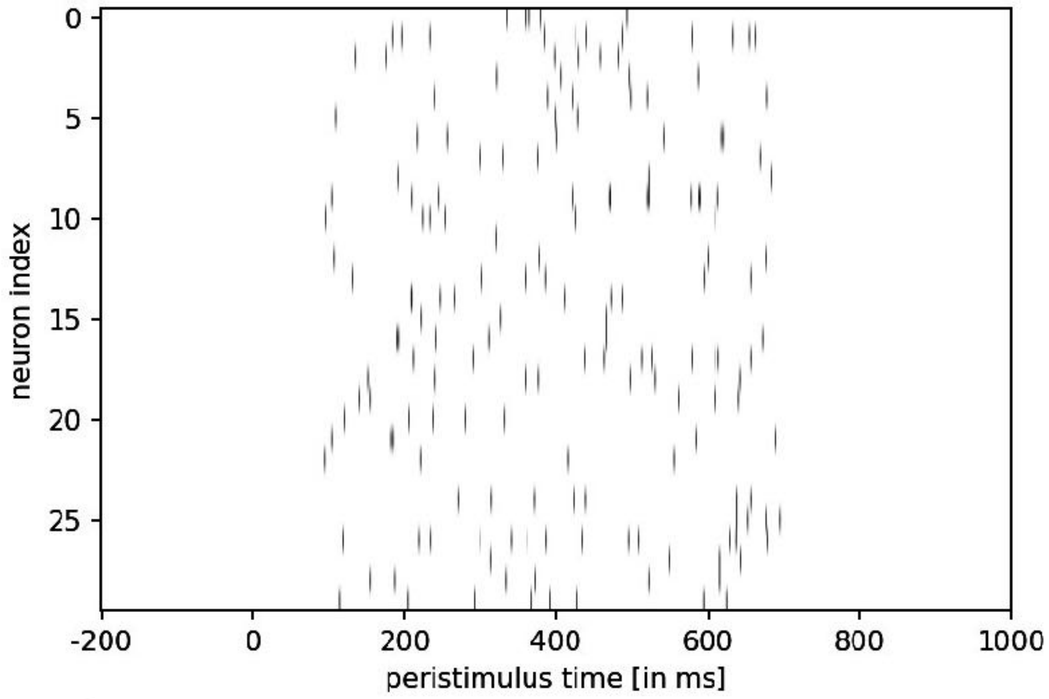
3. Few examples

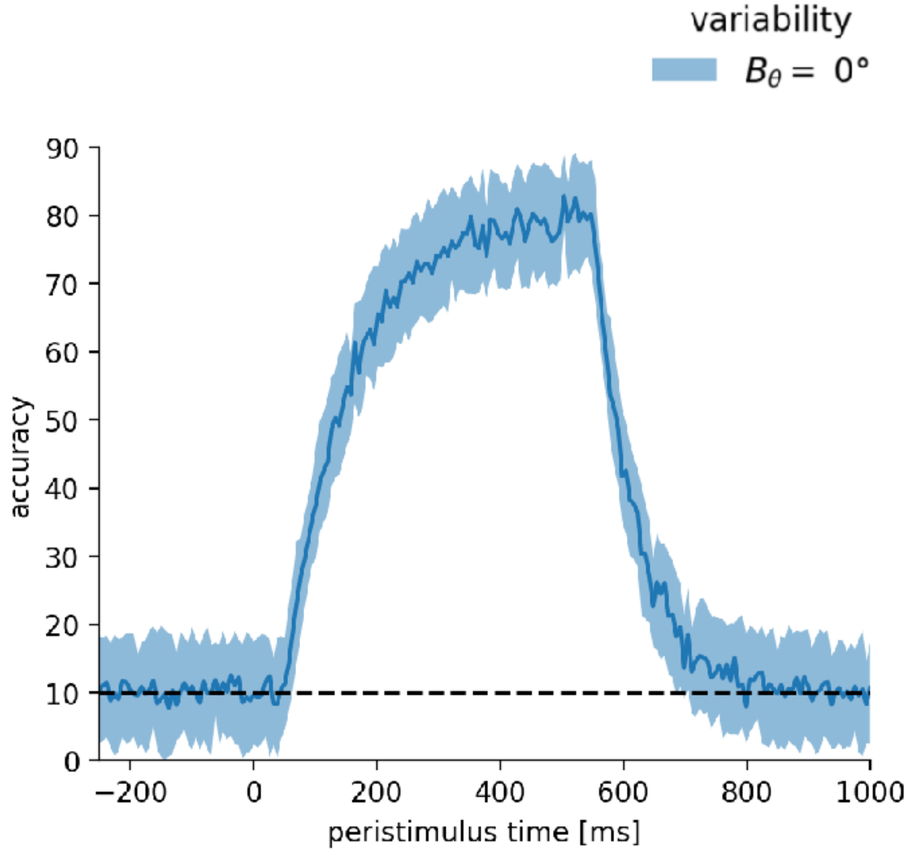
Also largely used in public research :

Classification for decoding





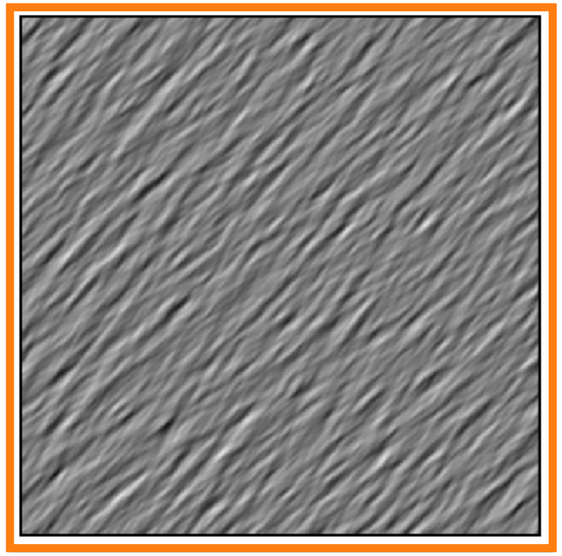


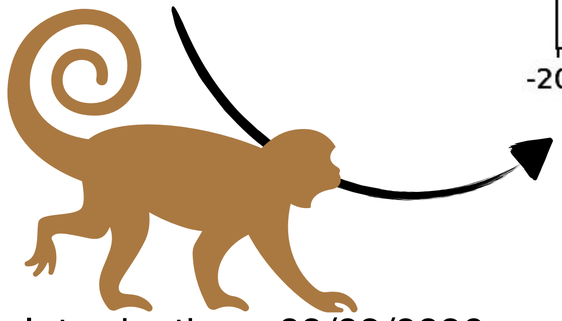


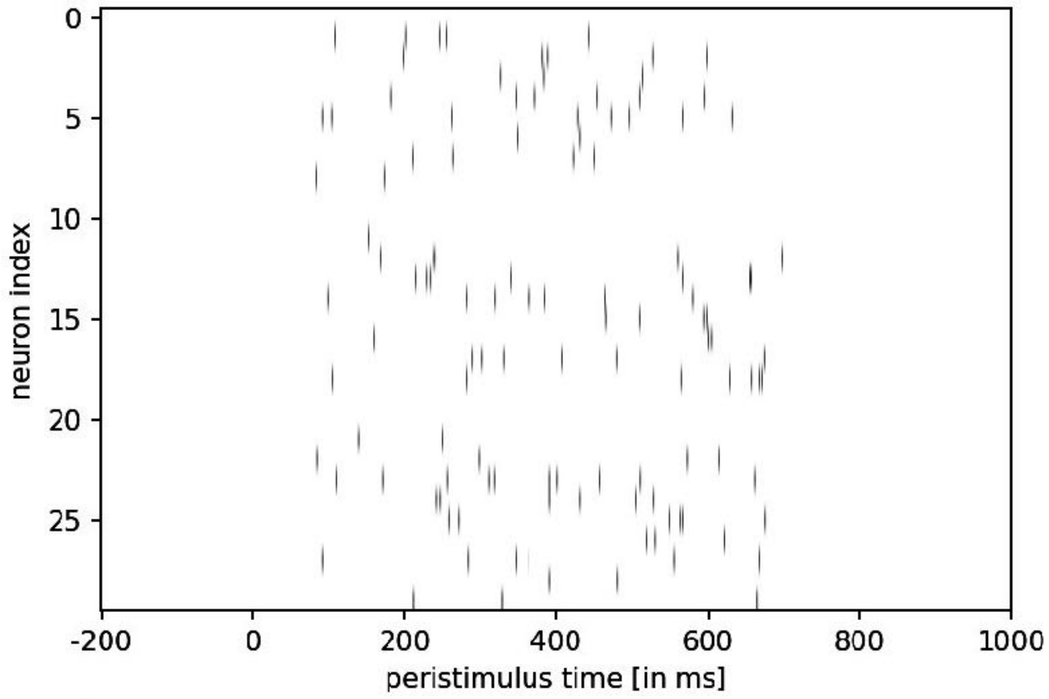
3. Few examples

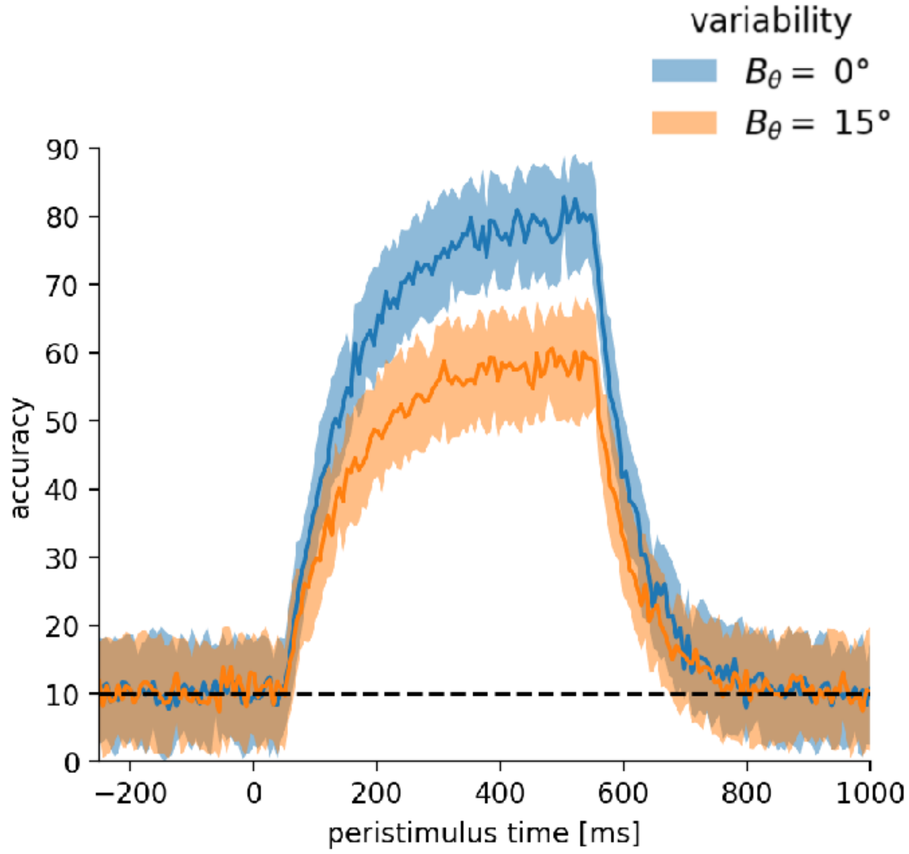
Also largely used in public research :

Classification for decoding





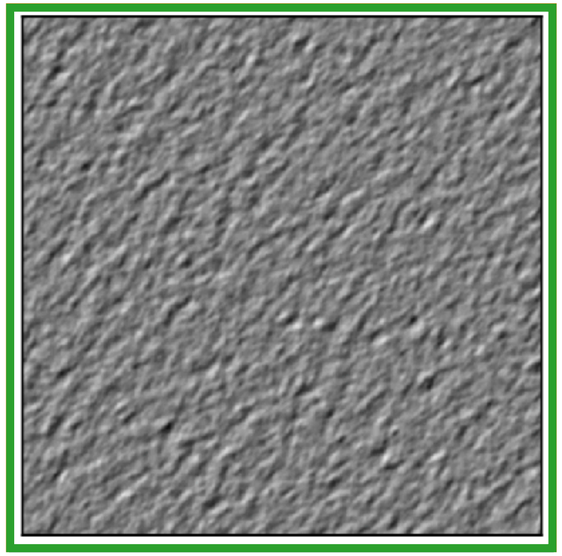


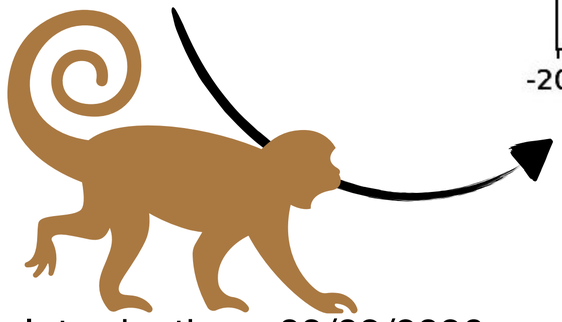


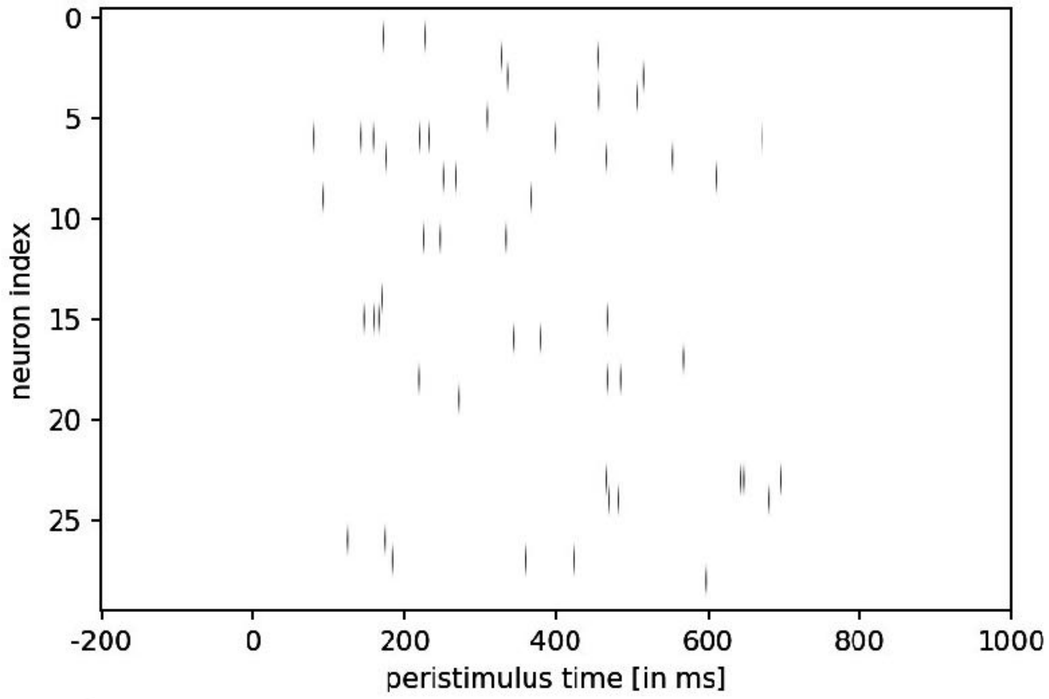
3. Few examples

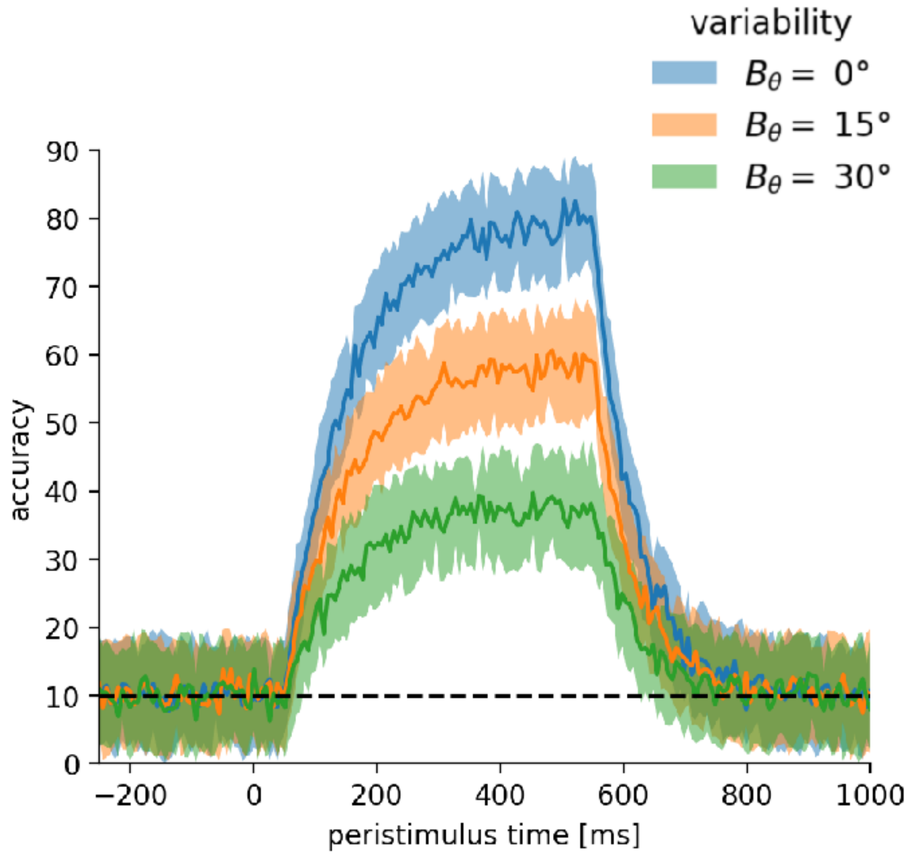
Also largely used in public research :

Classification for decoding





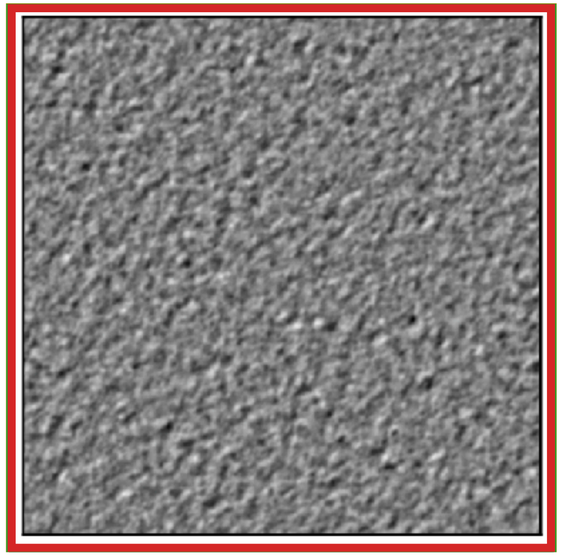


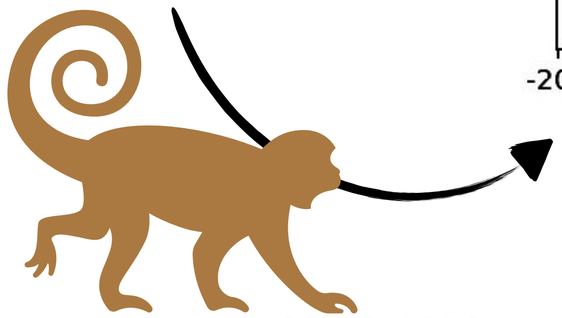


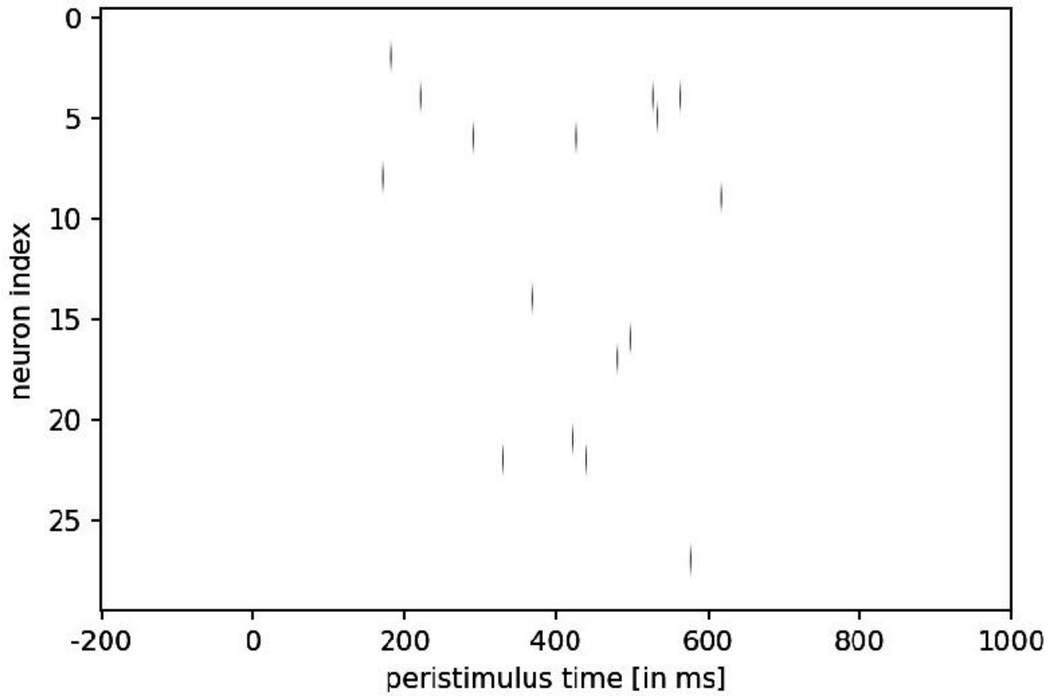
3. Few examples

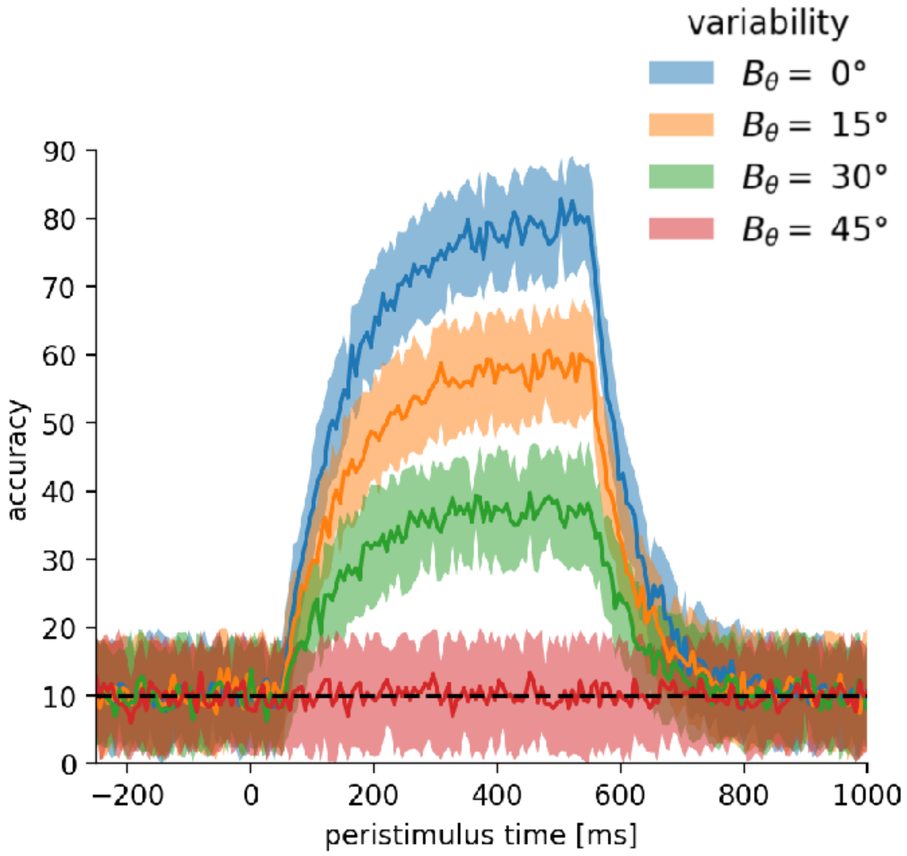
Also largely used in public research :

Classification for decoding







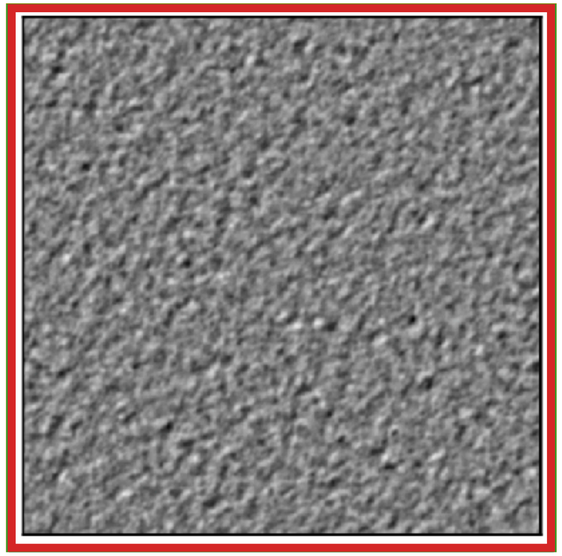


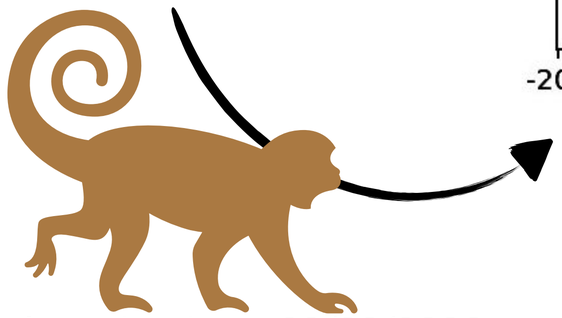
3. Few examples

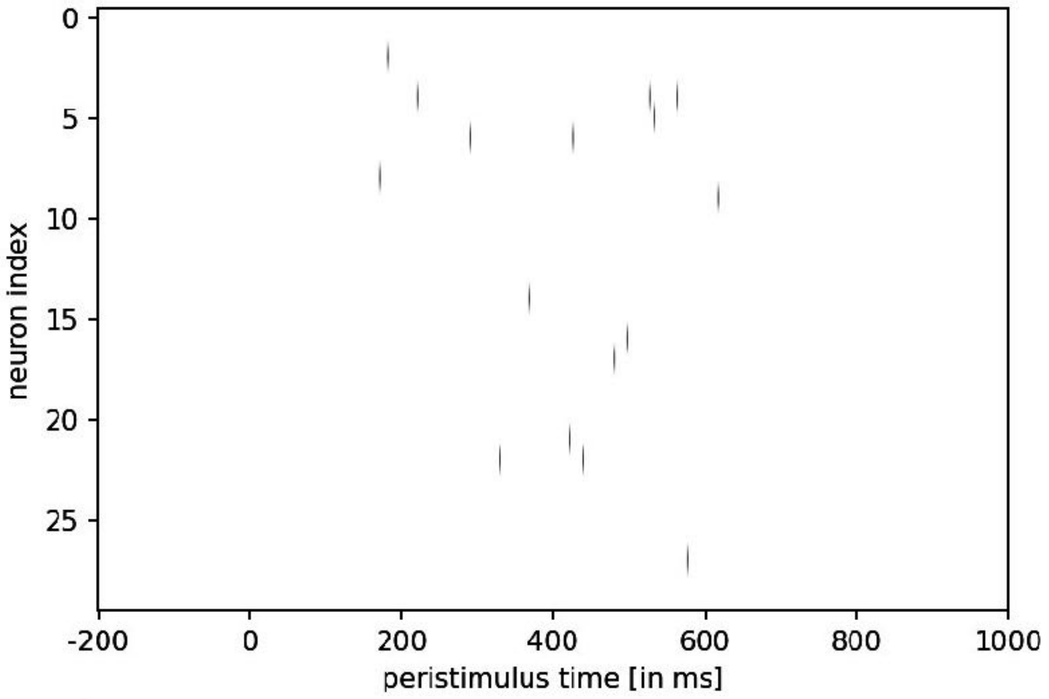
Also largely used in public research :

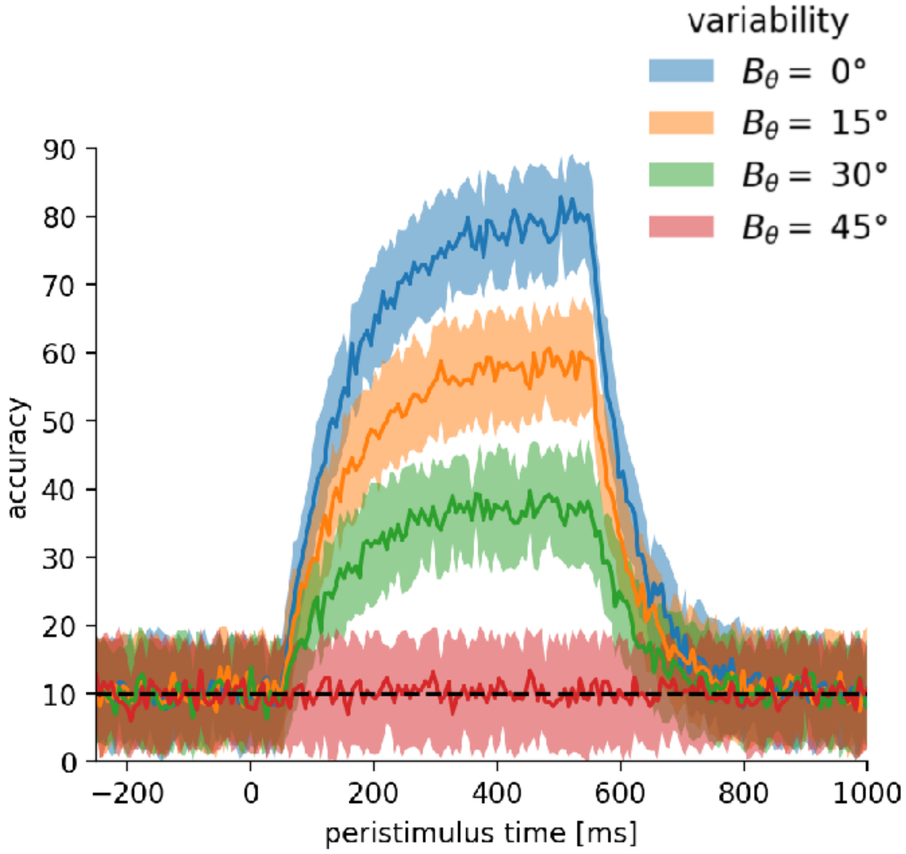
- Predict stimulus from neuronal activity
- Study how information is modulated
- Data-driven approach

Classification for decoding









3. Few examples

Also largely used in public research :

3. Few examples

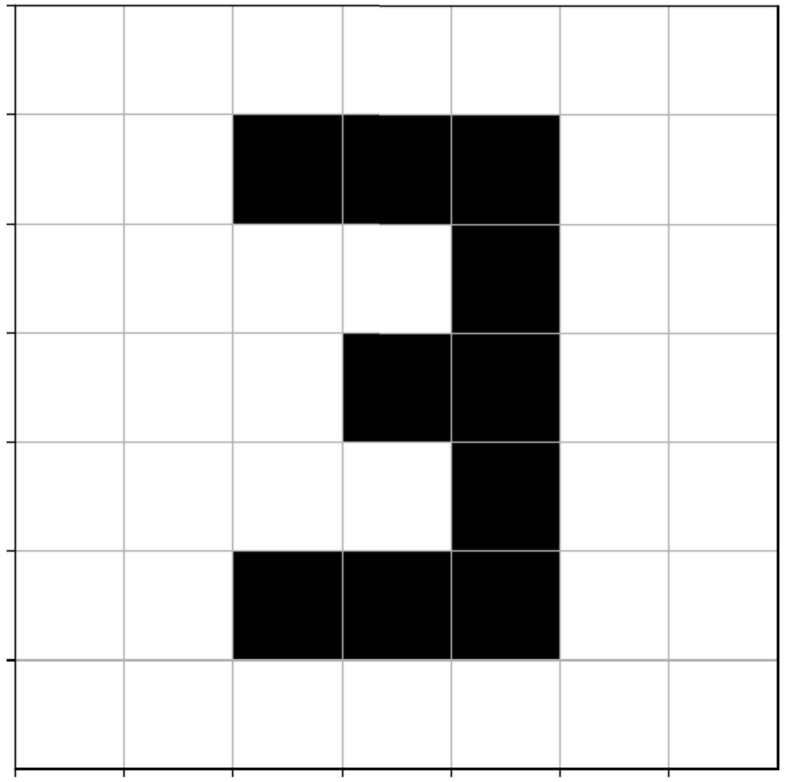
Also largely used in public research :

Convolutional Neuronal Network

3. Few examples

Also largely used in public research :

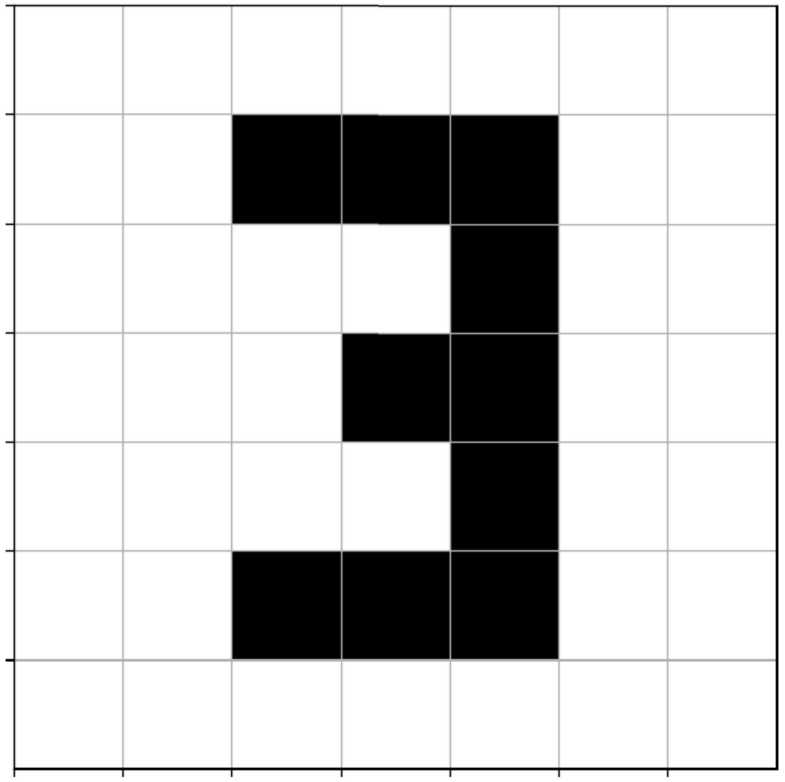
Convolutional Neuronal Network

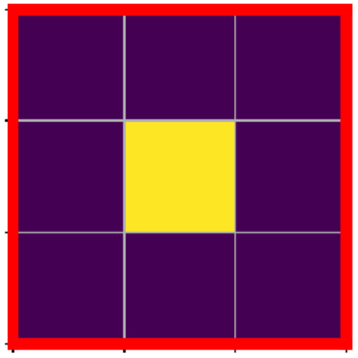


3. Few examples

Also largely used in public research :

Convolutional Neuronal Network

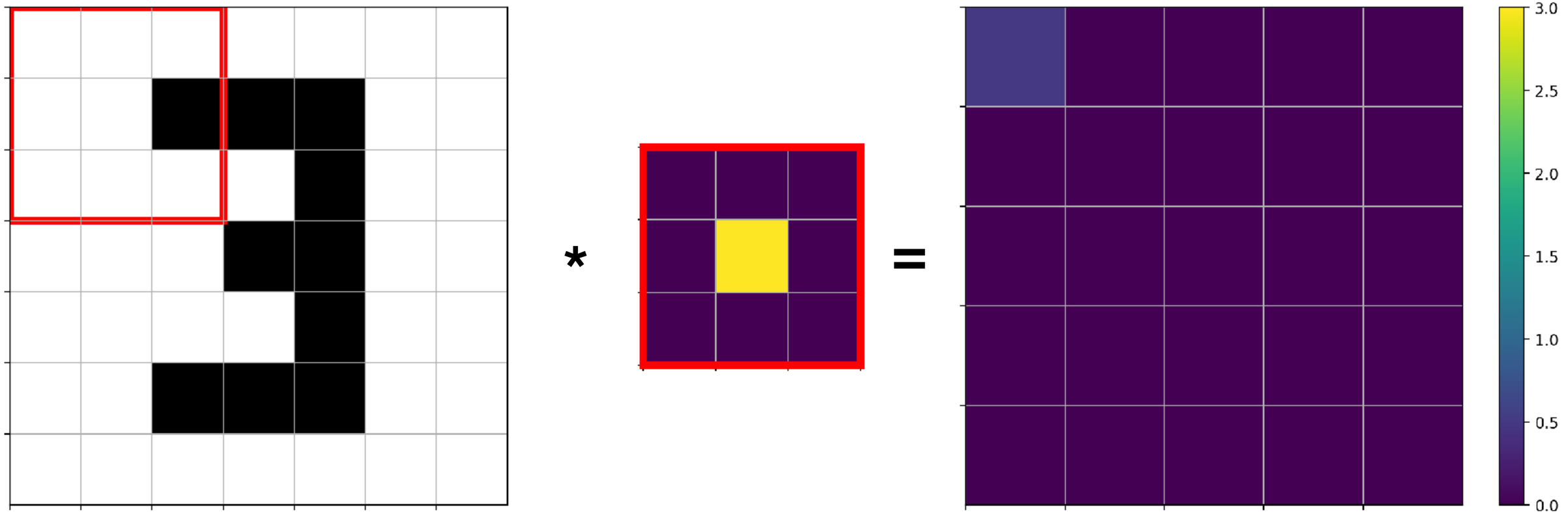




3. Few examples

Also largely used in public research :

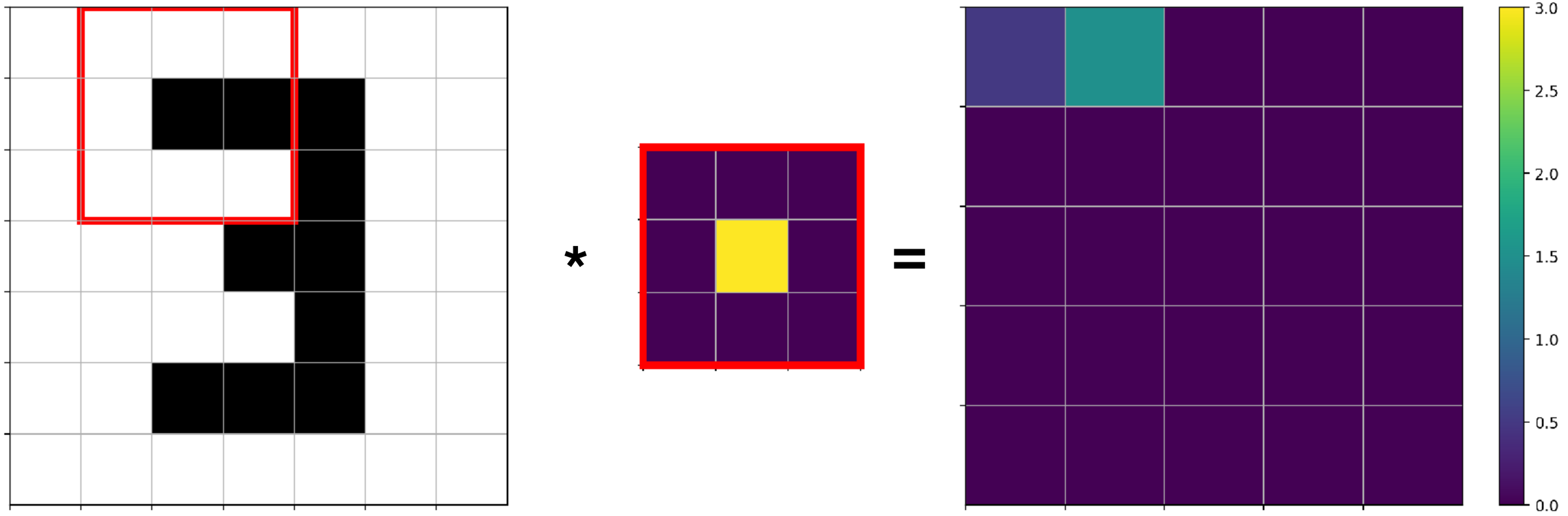
Convolutional Neuronal Network



3. Few examples

Also largely used in public research :

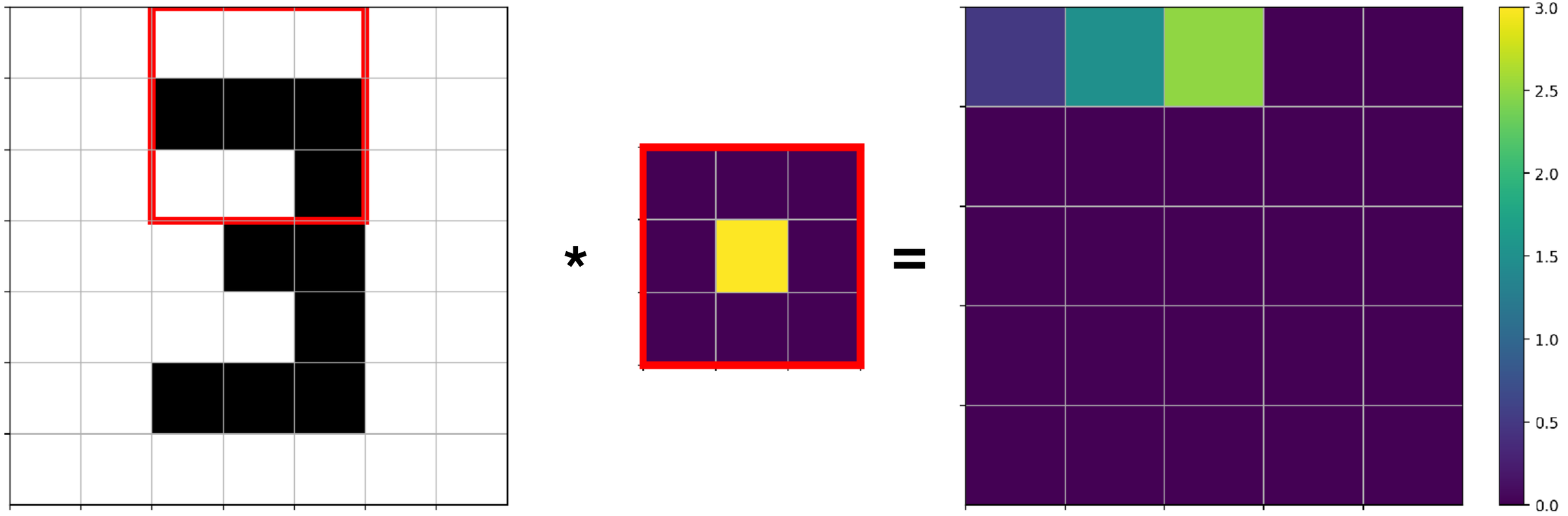
Convolutional Neuronal Network



3. Few examples

Also largely used in public research :

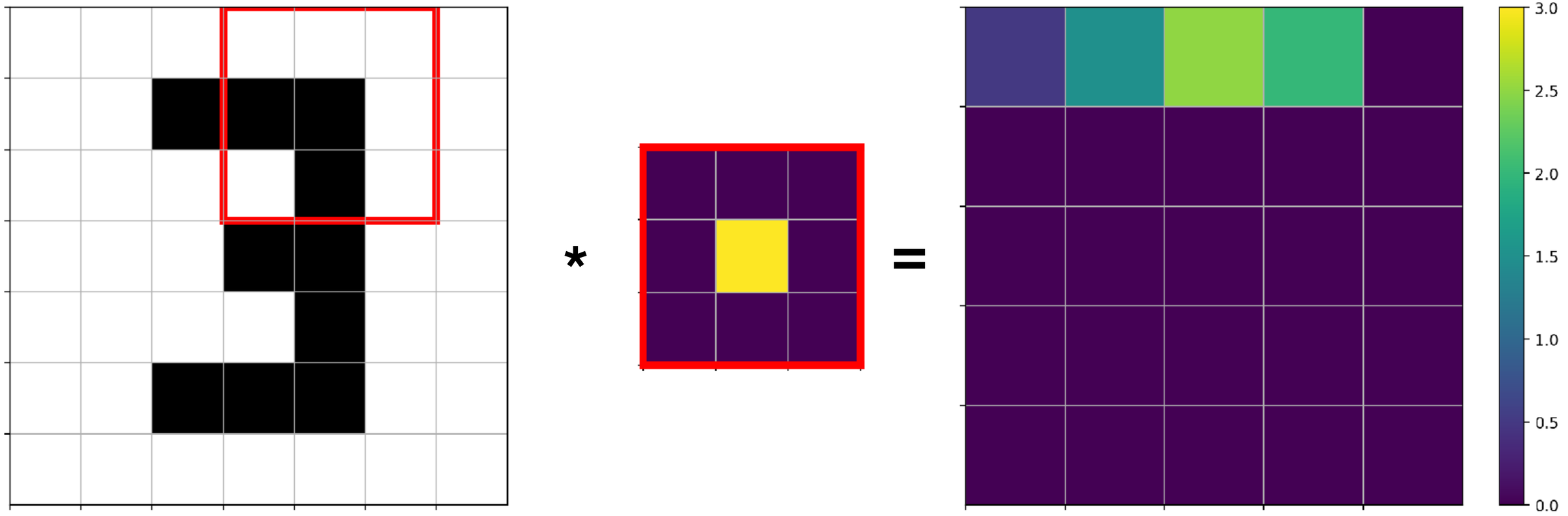
Convolutional Neuronal Network



3. Few examples

Also largely used in public research :

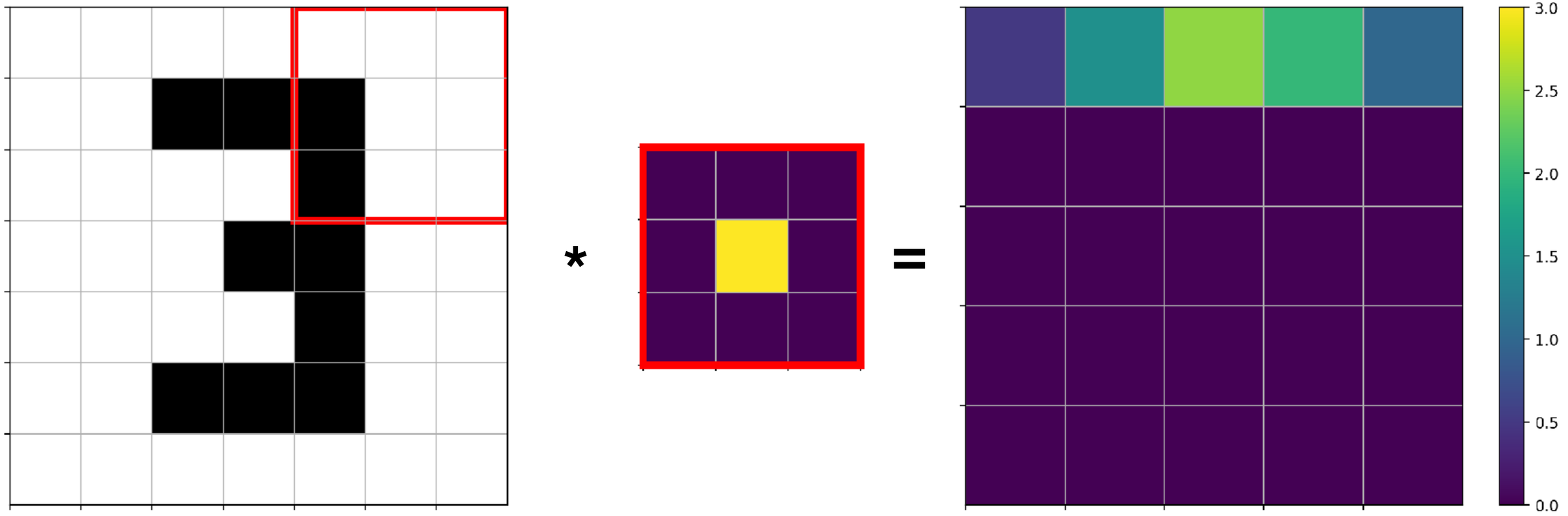
Convolutional Neuronal Network



3. Few examples

Also largely used in public research :

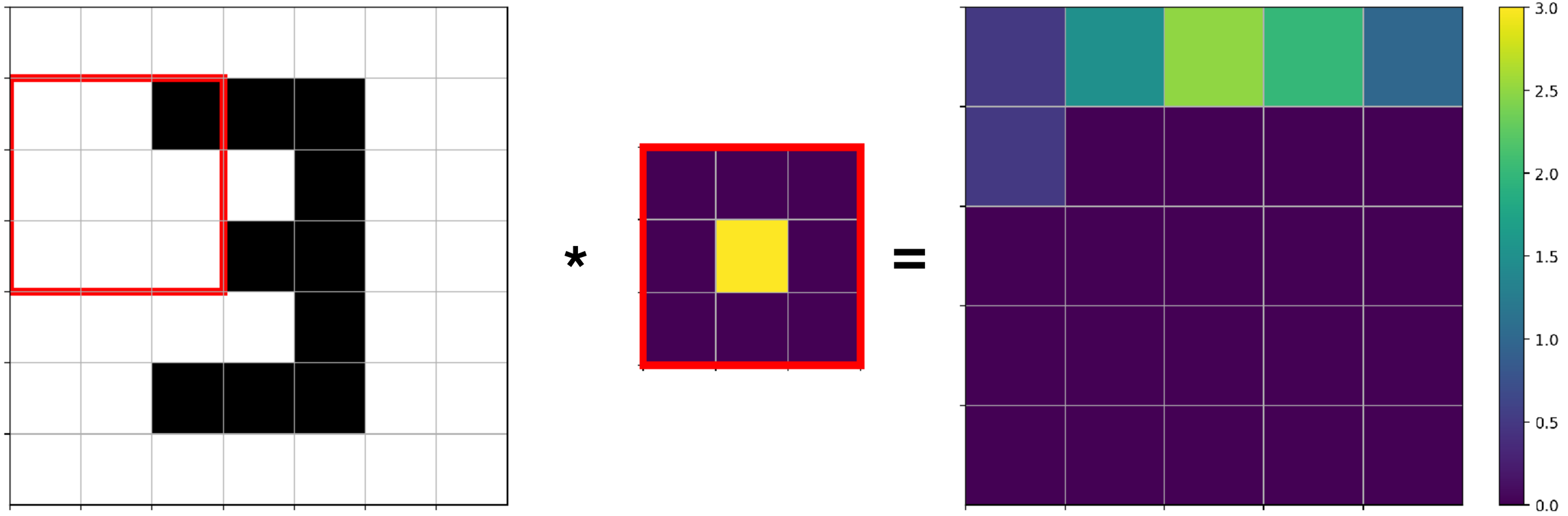
Convolutional Neuronal Network



3. Few examples

Also largely used in public research :

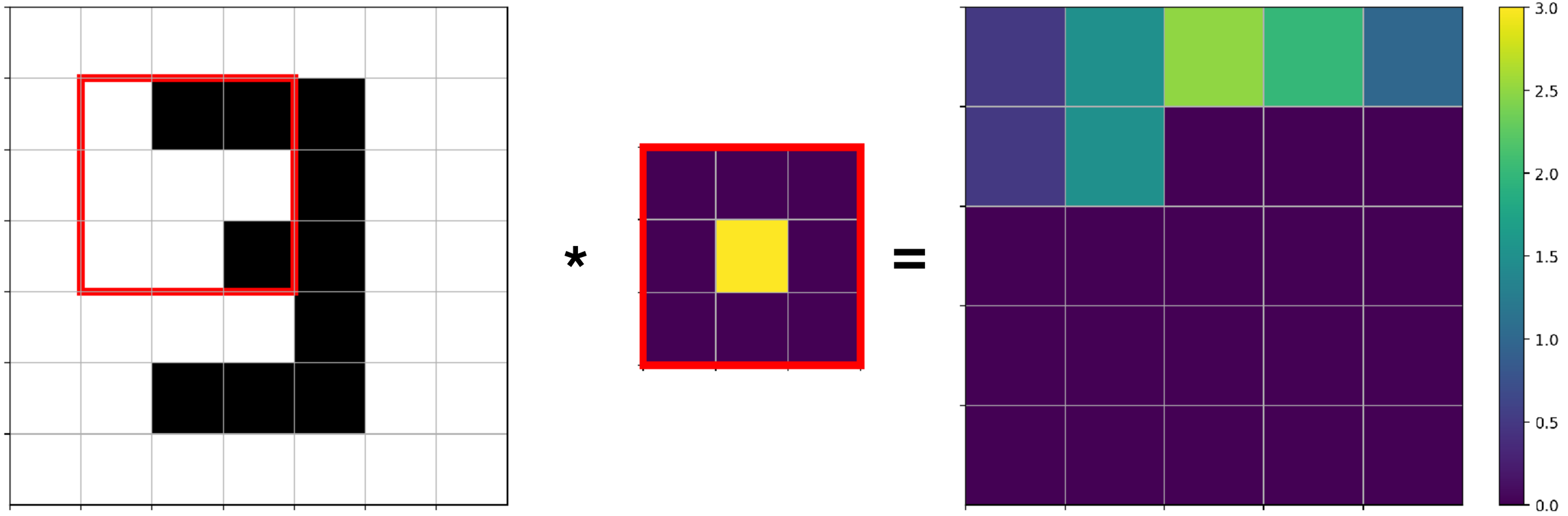
Convolutional Neuronal Network



3. Few examples

Also largely used in public research :

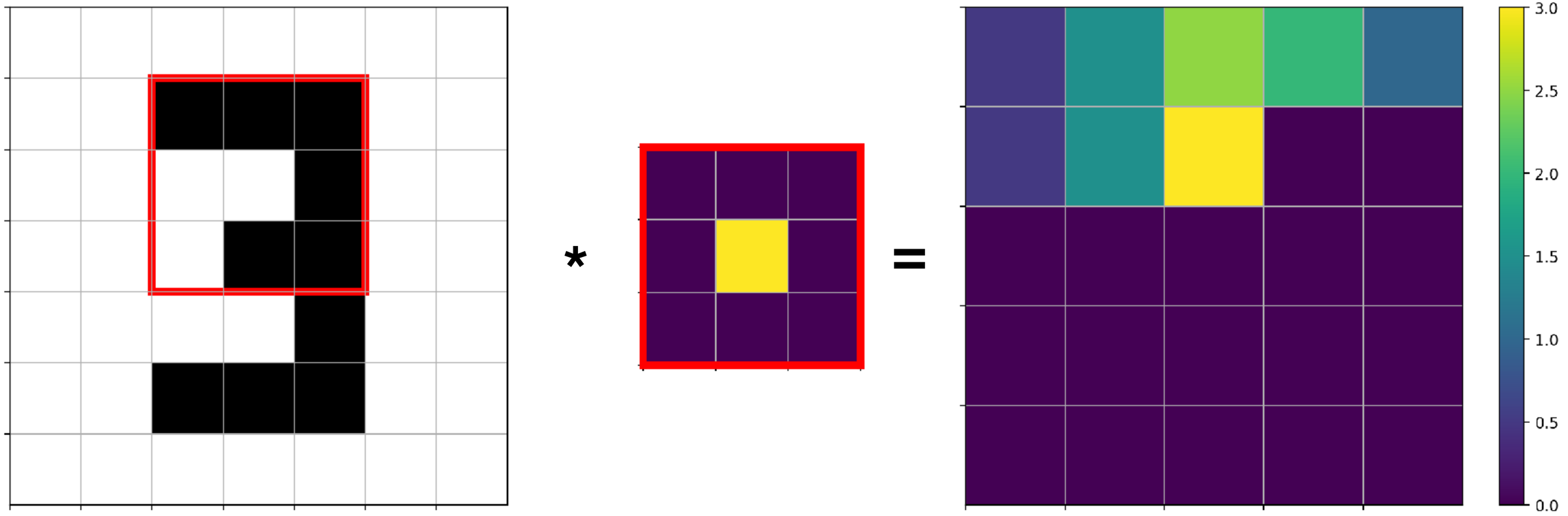
Convolutional Neuronal Network



3. Few examples

Also largely used in public research :

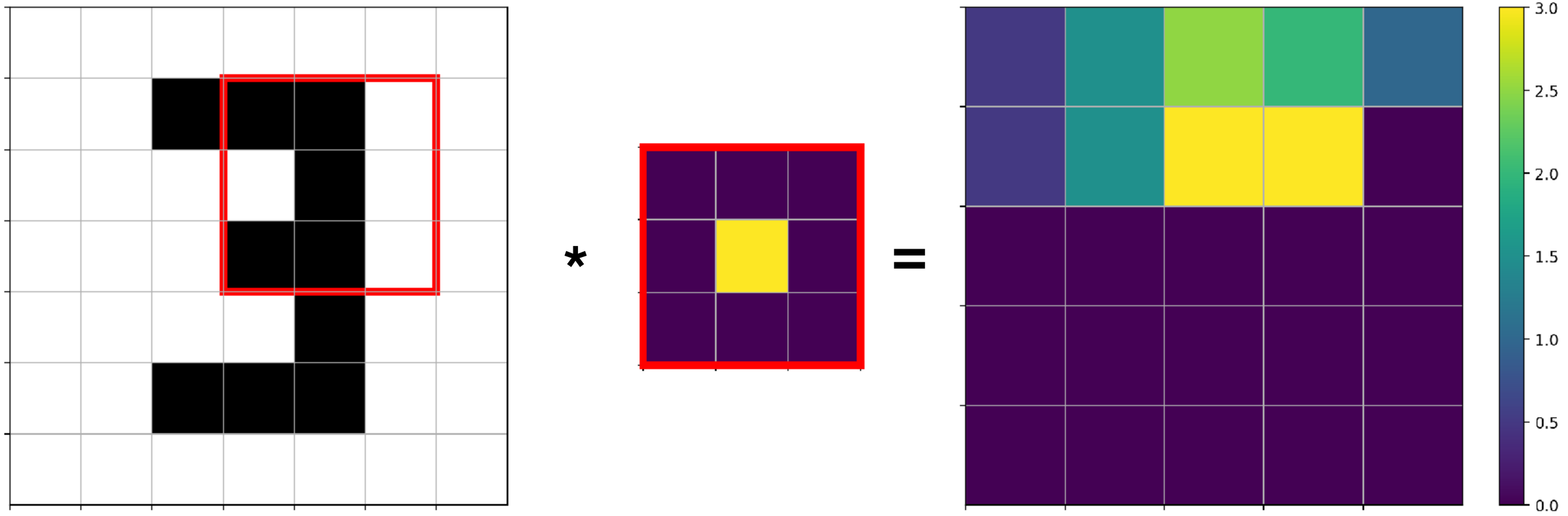
Convolutional Neuronal Network



3. Few examples

Also largely used in public research :

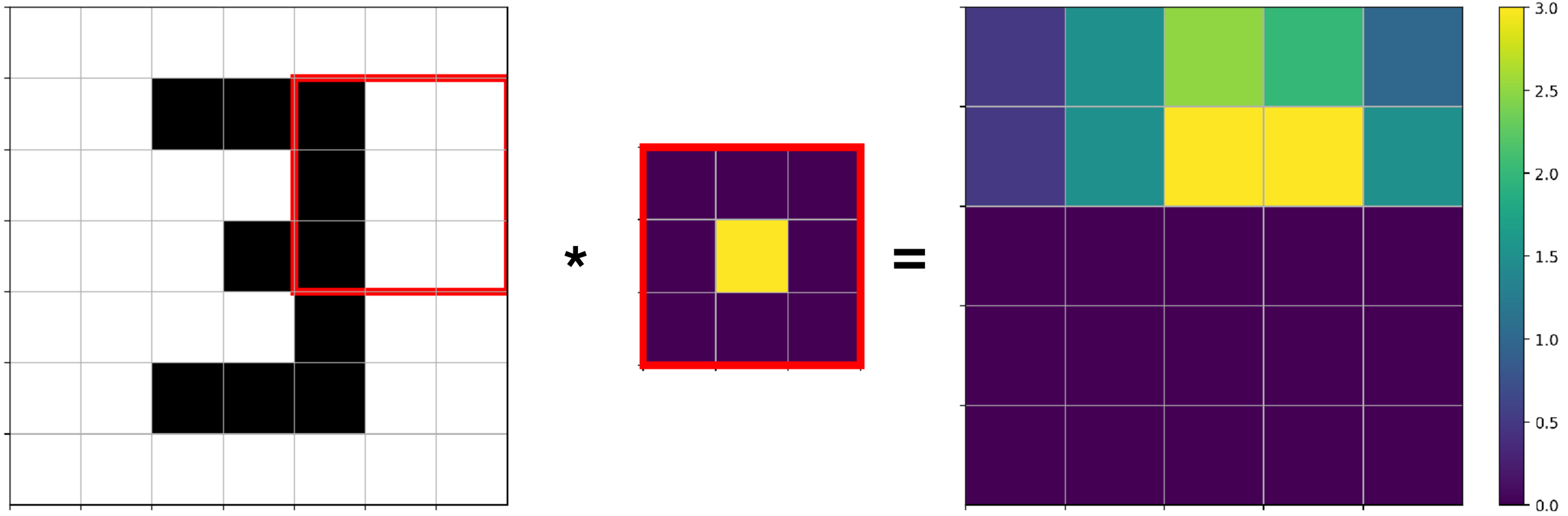
Convolutional Neuronal Network



3. Few examples

Also largely used in public research :

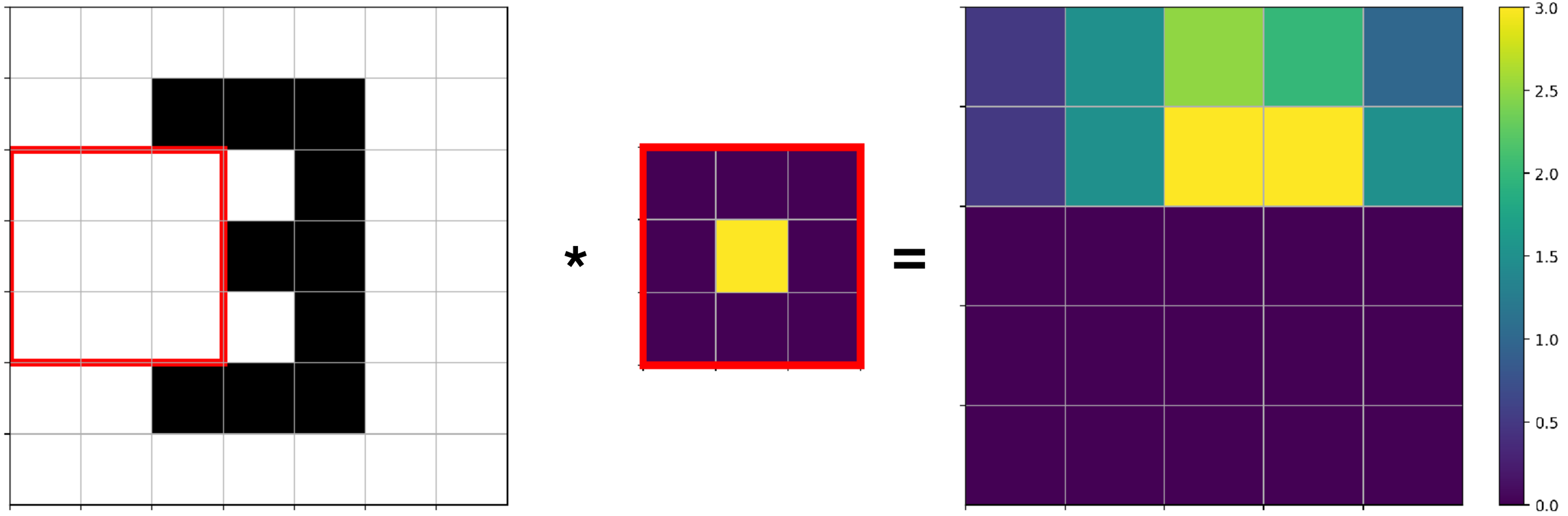
Convolutional Neuronal Network



3. Few examples

Also largely used in public research :

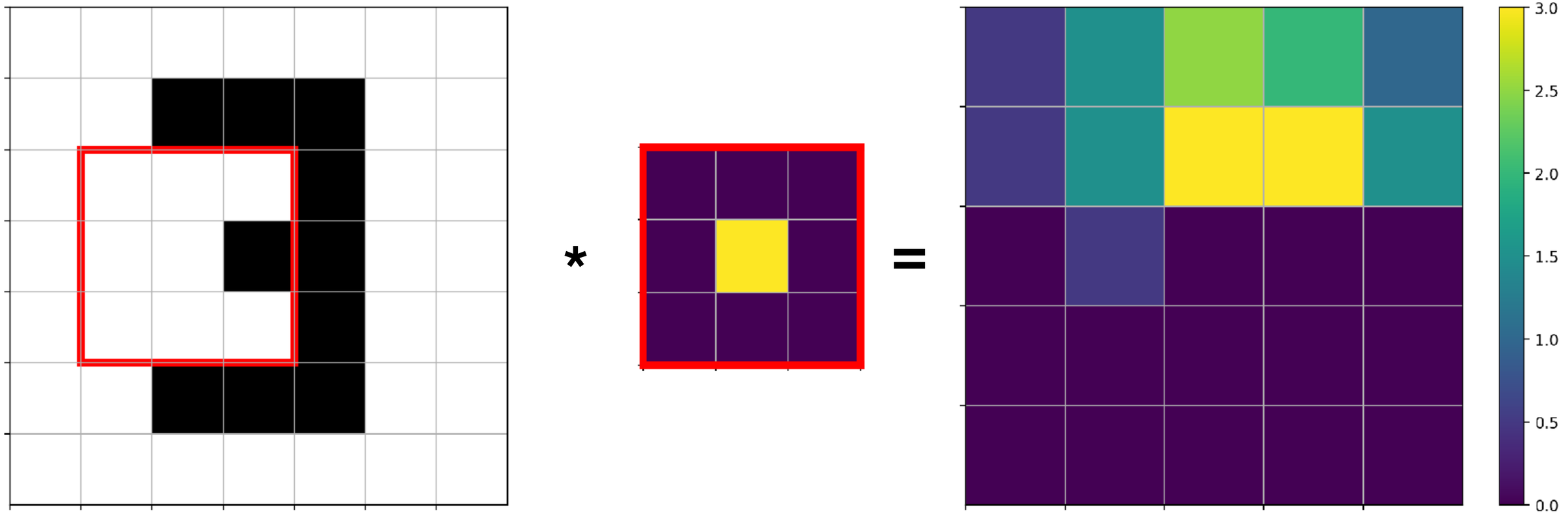
Convolutional Neuronal Network



3. Few examples

Also largely used in public research :

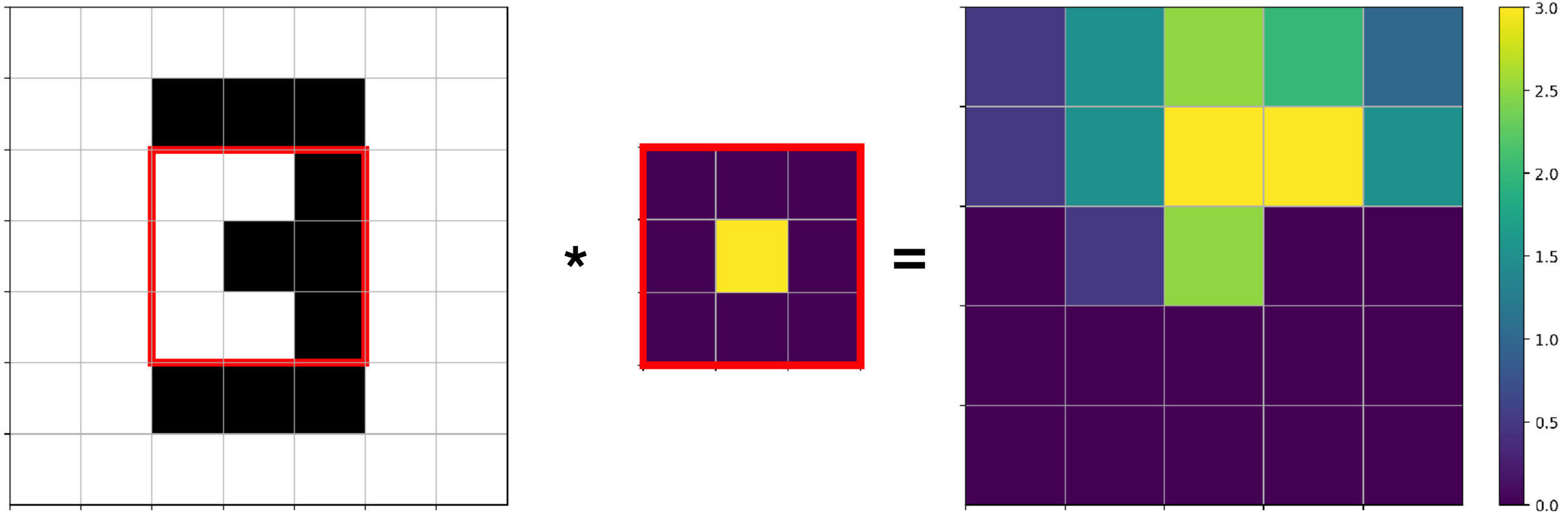
Convolutional Neuronal Network



3. Few examples

Also largely used in public research :

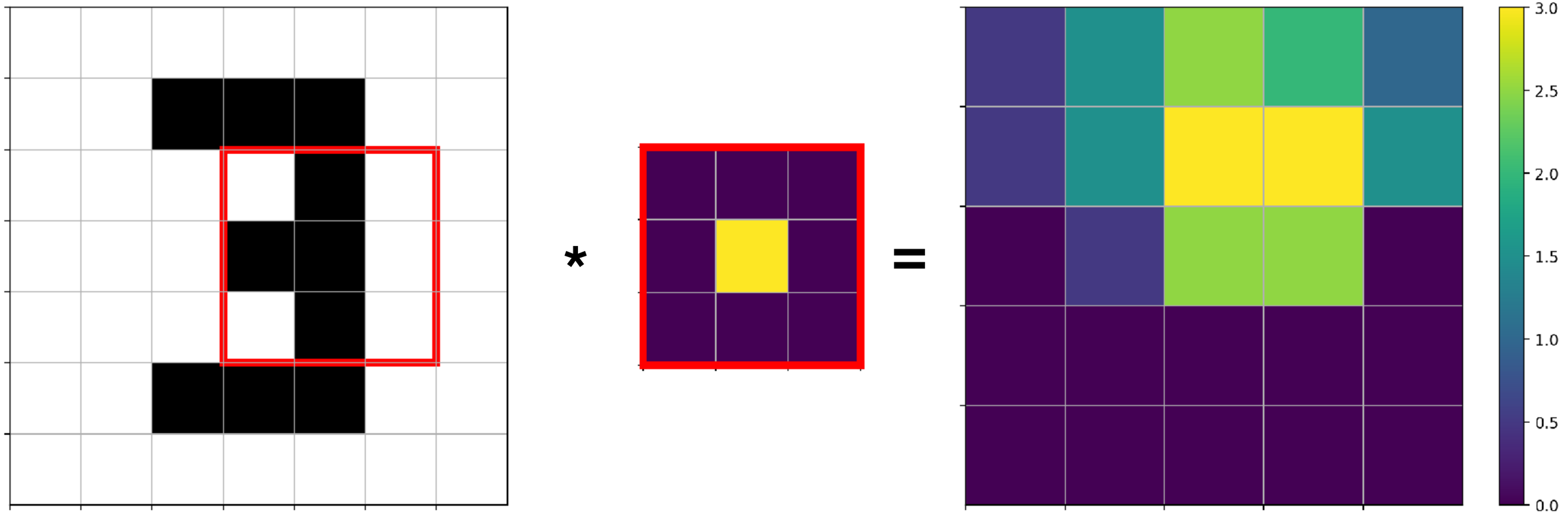
Convolutional Neuronal Network



3. Few examples

Also largely used in public research :

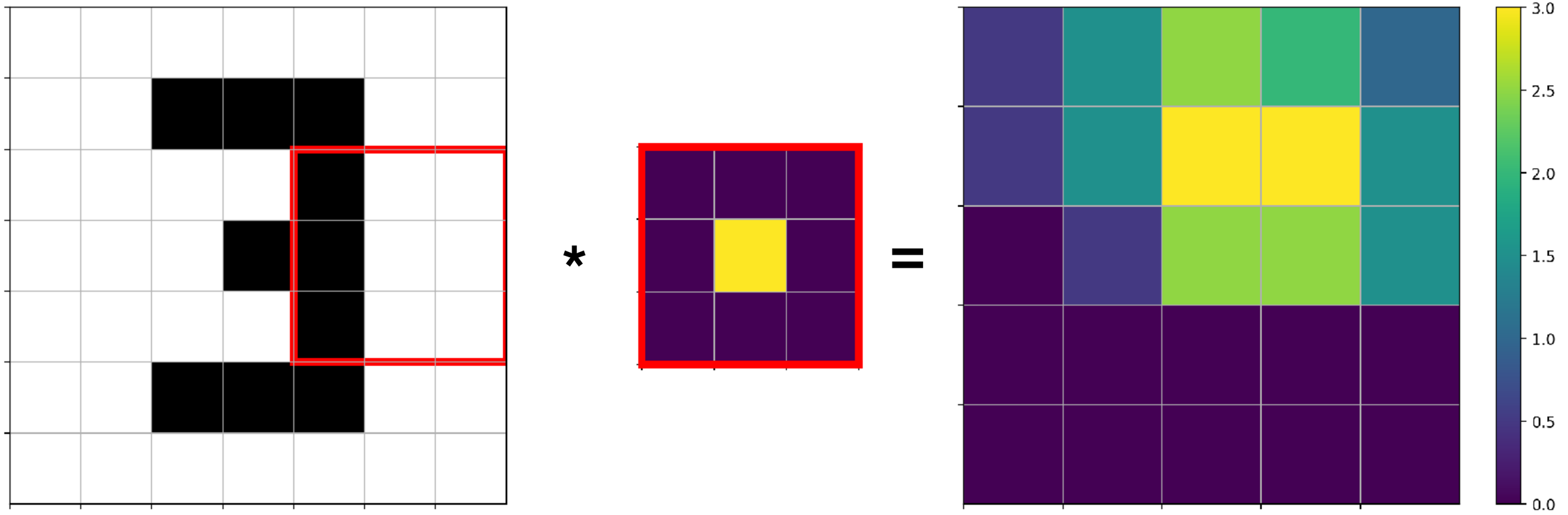
Convolutional Neuronal Network



3. Few examples

Also largely used in public research :

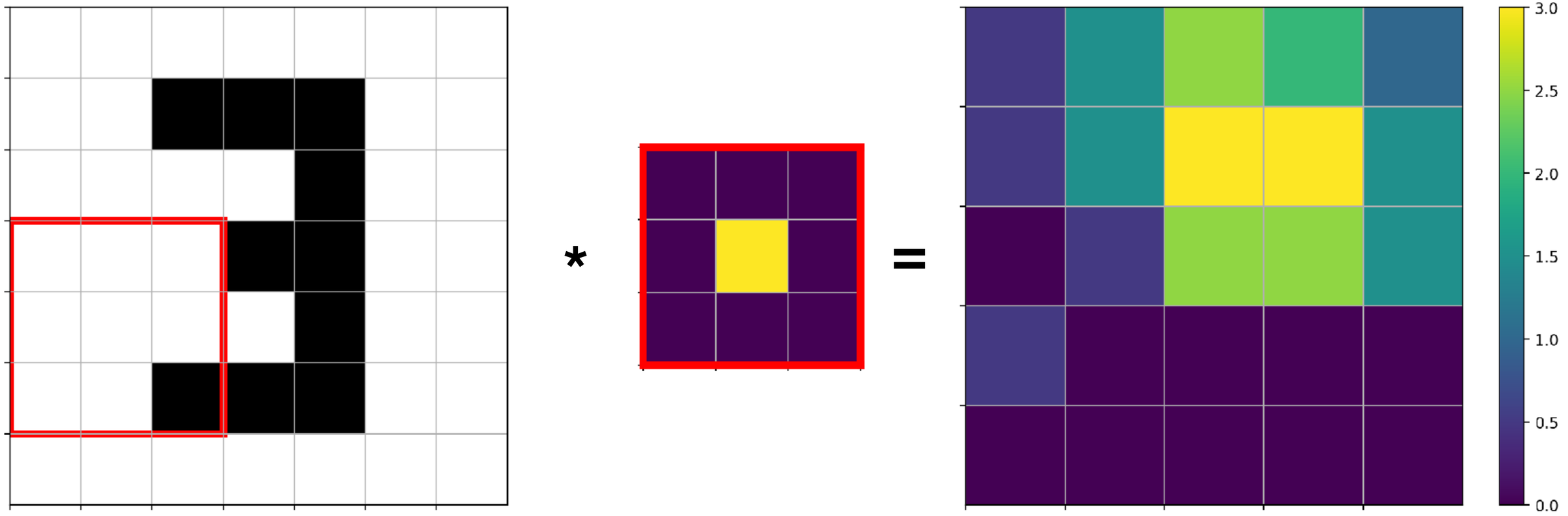
Convolutional Neuronal Network



3. Few examples

Also largely used in public research :

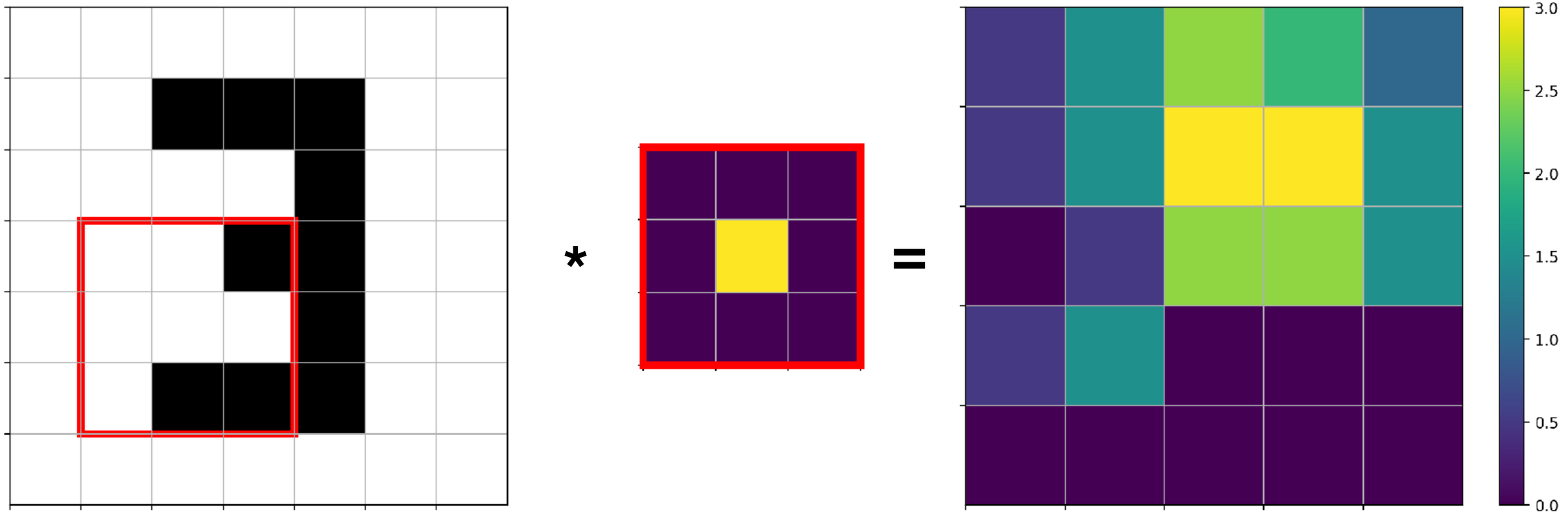
Convolutional Neuronal Network



3. Few examples

Also largely used in public research :

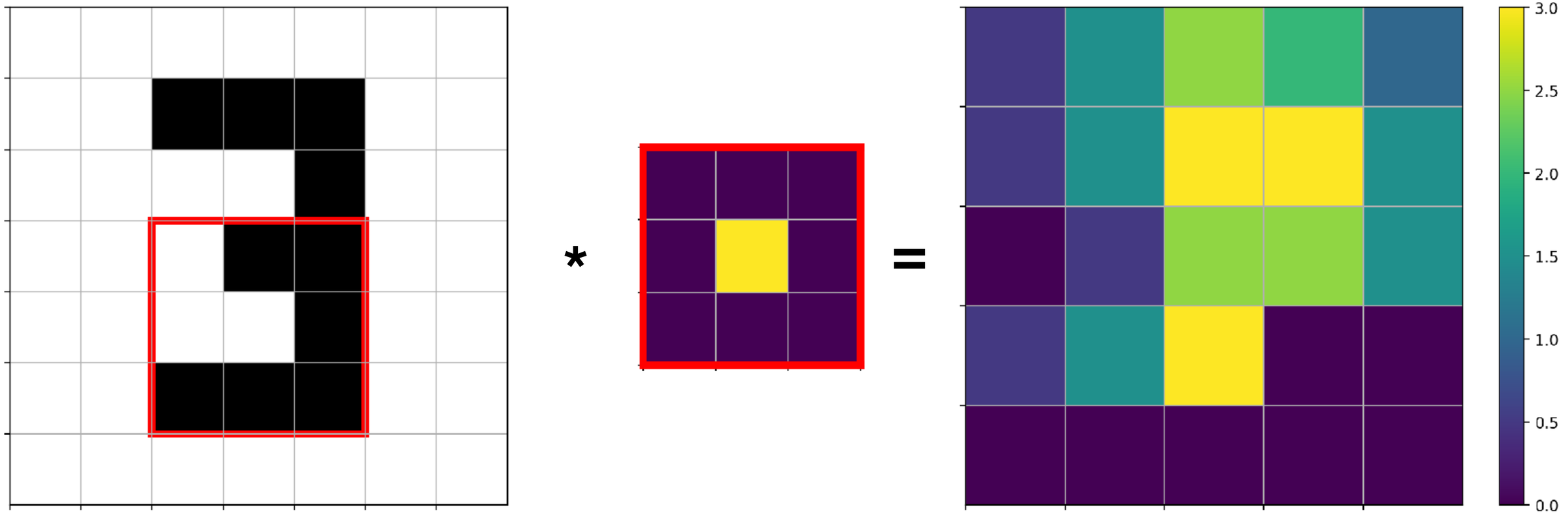
Convolutional Neuronal Network



3. Few examples

Also largely used in public research :

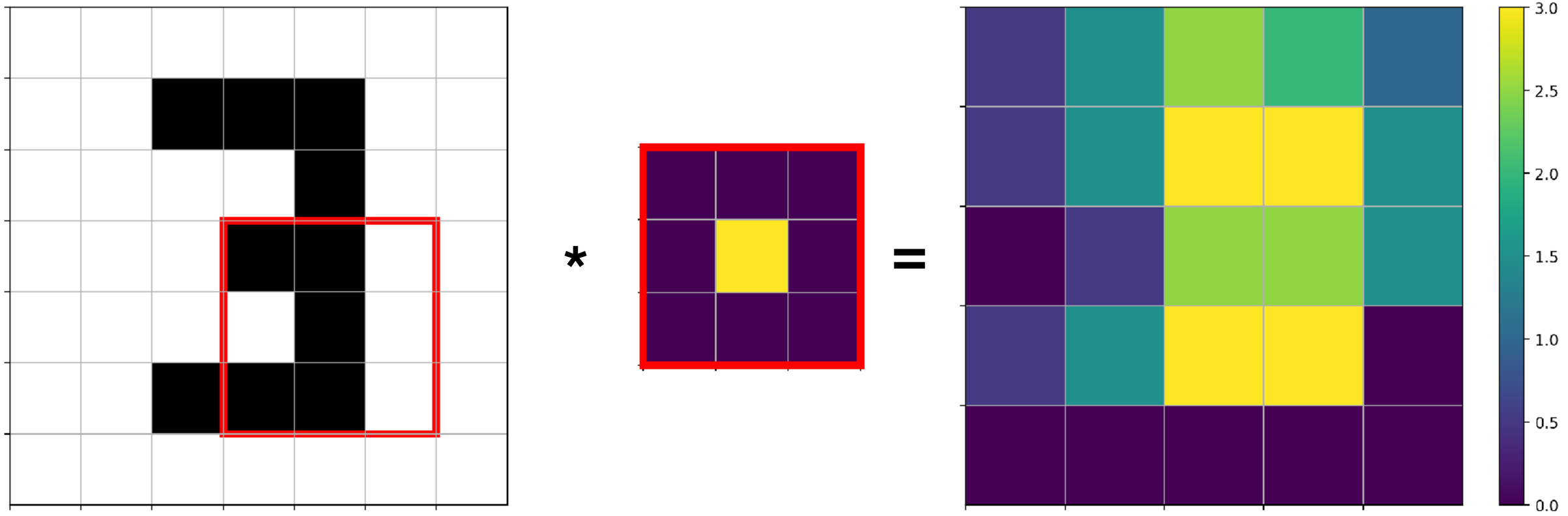
Convolutional Neuronal Network



3. Few examples

Also largely used in public research :

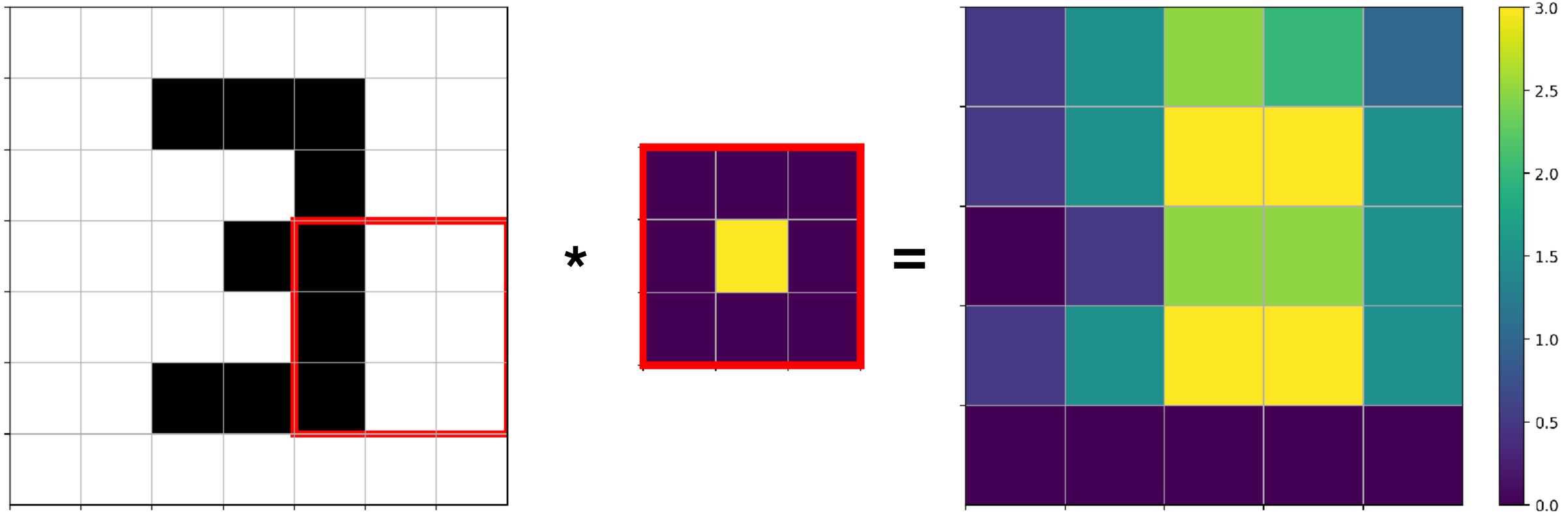
Convolutional Neuronal Network



3. Few examples

Also largely used in public research :

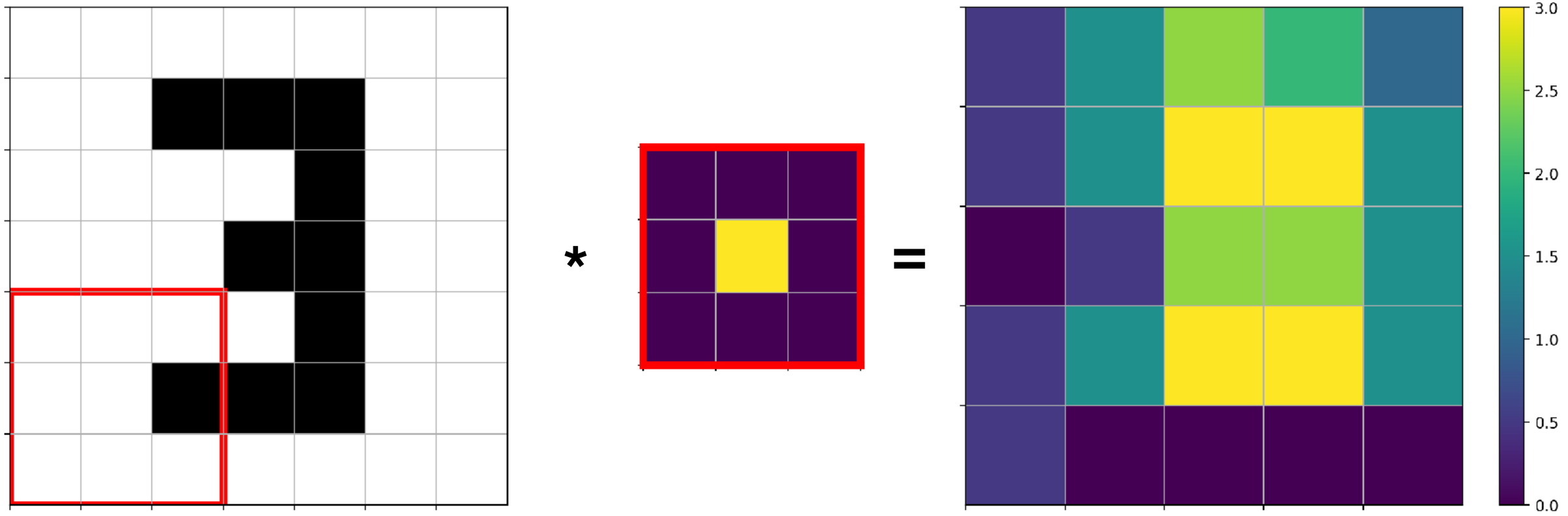
Convolutional Neuronal Network



3. Few examples

Also largely used in public research :

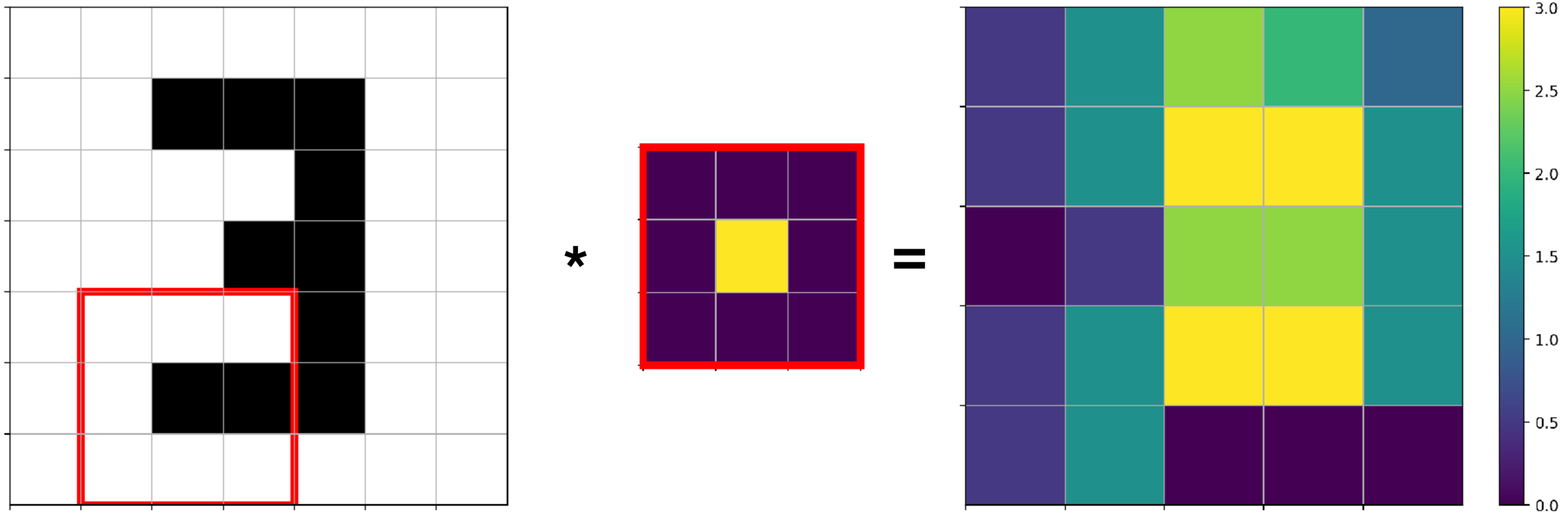
Convolutional Neuronal Network



3. Few examples

Also largely used in public research :

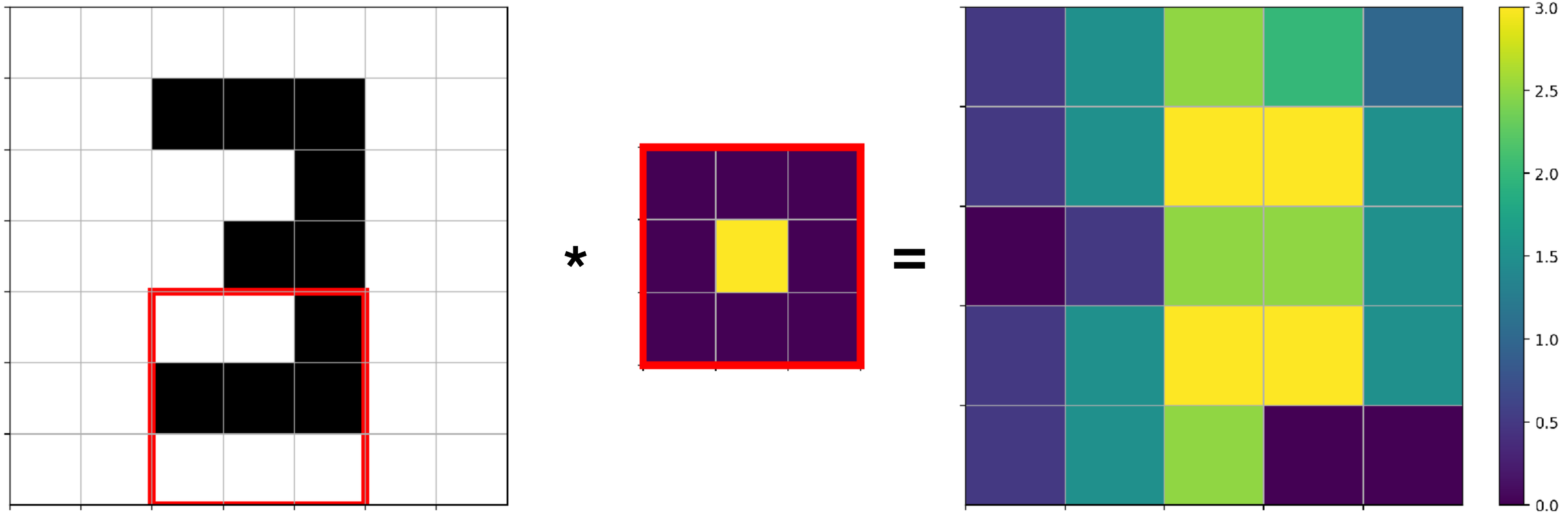
Convolutional Neuronal Network



3. Few examples

Also largely used in public research :

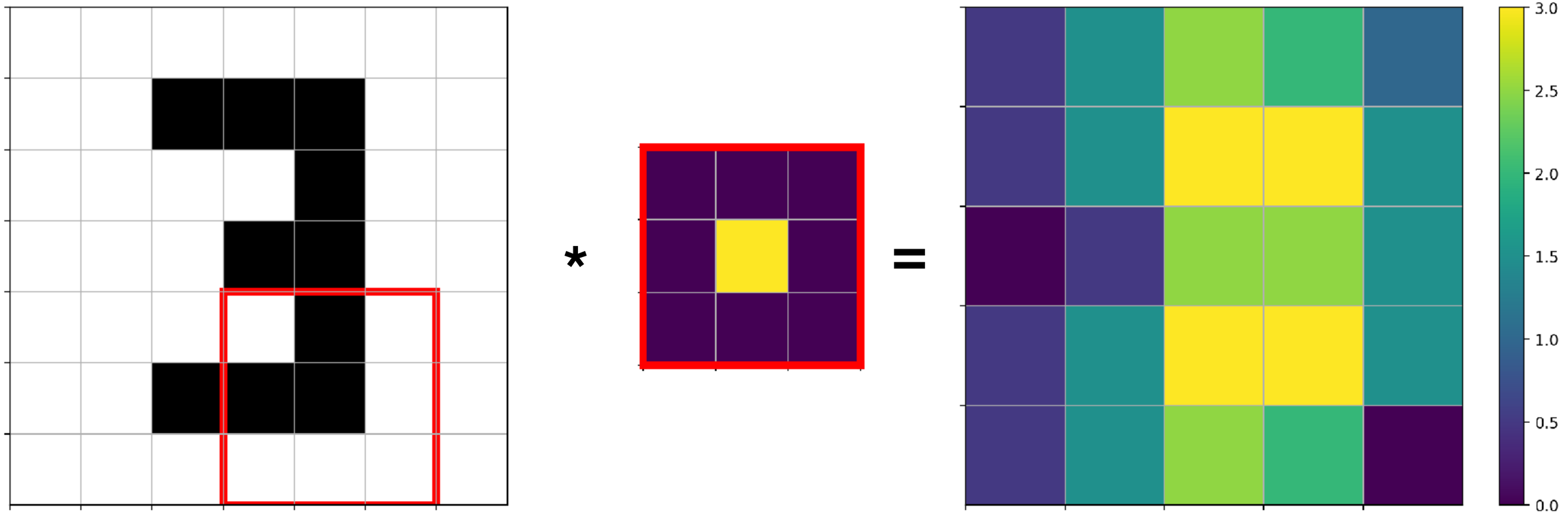
Convolutional Neuronal Network



3. Few examples

Also largely used in public research :

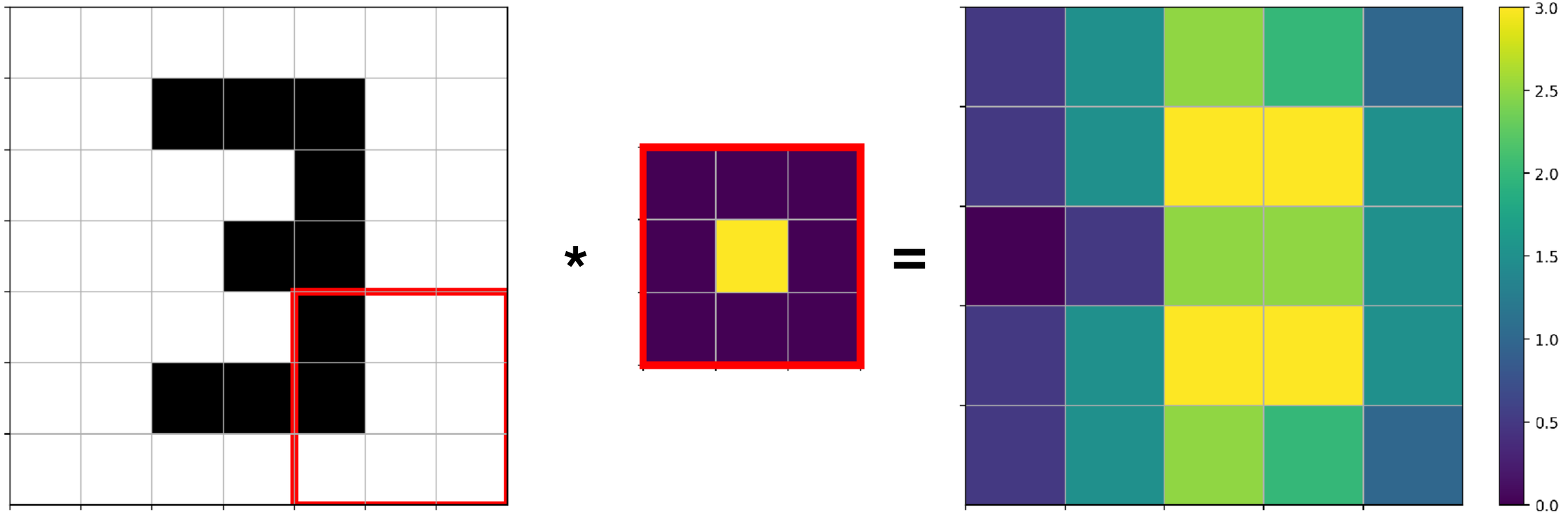
Convolutional Neuronal Network



3. Few examples

Also largely used in public research :

Convolutional Neuronal Network



3. Few examples

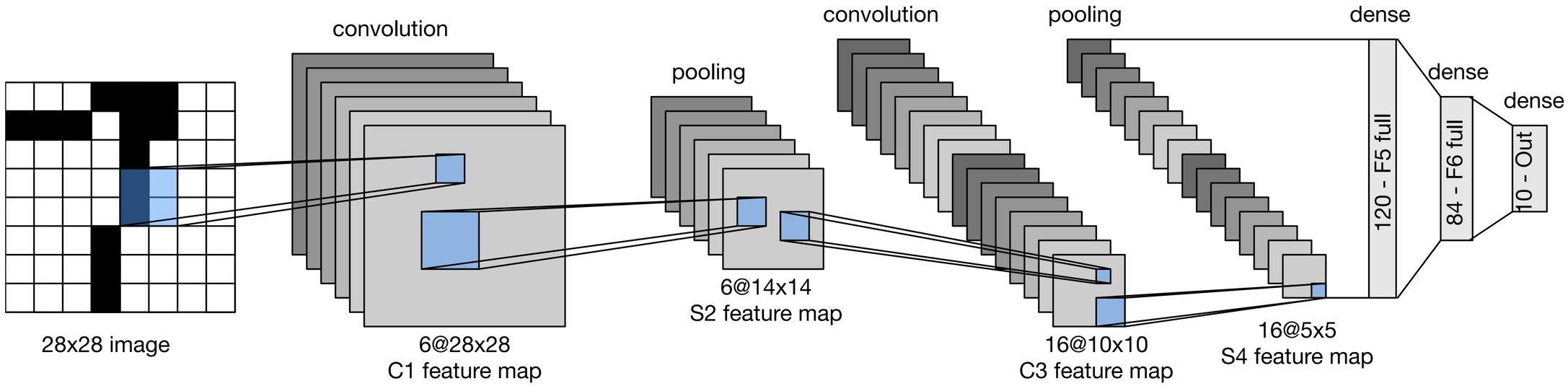
Also largely used in public research :

Convolutional Neuronal Network

3. Few examples

Also largely used in public research :

Convolutional Neuronal Network



GitHub

PyTorch Introduction

Workshop on Artificial Intelligence - 03/02/2026

Matthis Dallain & Alexandre Lainé - NeOpTo team
mattis.dallain@univ-amu.fr / alexandre.laine@univ-amu.fr

Aix Marseille Université

institut de neurosciences de la timone

Additional information

PyTorch vs Keras vs TensorFlow

Software	Creator	Initial release	Software license[a]	Open source	Platform	Written in	Interface	OpenMP support	OpenCL support	CUDA support	ROCm support[1]	Automatic differentiation[2]	Has pretrained models	Recurrent nets	Convolutional nets	RBM/DBNs	Parallel execution (multi node)	Actively developed
Keras	François Chollet	2015	MIT license	Yes	Linux, macOS, Windows	Python	Python, R	Only if using Theano as backend	Can use Theano, Tensorflow or PlaidML as backends	Yes	No	Yes	Yes[20]	Yes	Yes	No[21]	Yes[22]	Yes
PyTorch	Meta AI	2016	BSD	Yes	Linux, macOS, Windows, Android[47]	Python, C, C++, CUDA	Python, C++, Julia, R[48]	Yes	Via separately maintained package[49][50][51]	Yes	Yes	Yes	Yes	Yes	Yes	Yes[52]	Yes	Yes
TensorFlow	Google Brain	2015	Apache 2.0	Yes	Linux, macOS, Windows,[55] [56] Android	C++, Python, CUDA	Python (Keras), C/C++, Java, Go, JavaScript, R,[57] Julia, Swift	No	On roadmap[58] but already with SYCL[59] support	Yes	Yes	Yes[60]	Yes[61]	Yes	Yes	Yes	Yes	Yes

DBN = Deep Belief Network / RBM = Restricted Boltzmann Machine / ROCm = CUDA for AMD GPU
OpenMP = CPU (GPU) parallelization / OpenCL portability between