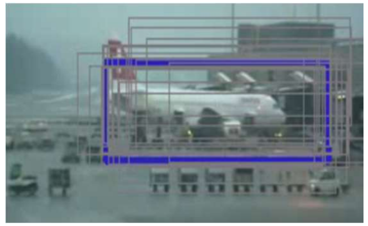
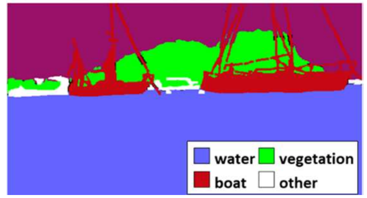
UNIVERSITÀ DEGLI STUDI DELLA BASILICATA

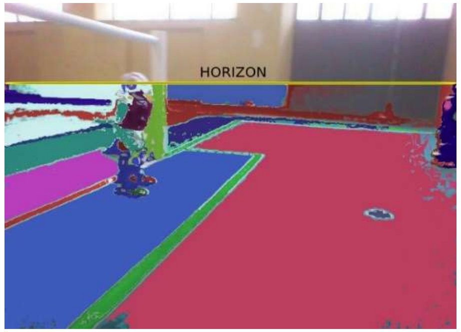
Corso di Visione e Percezione

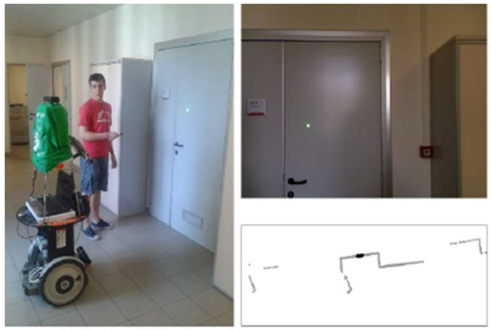
Docente
Domenico D. Bloisi



Introduzione al Deep Learning



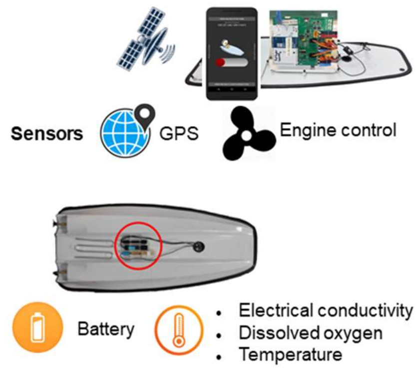


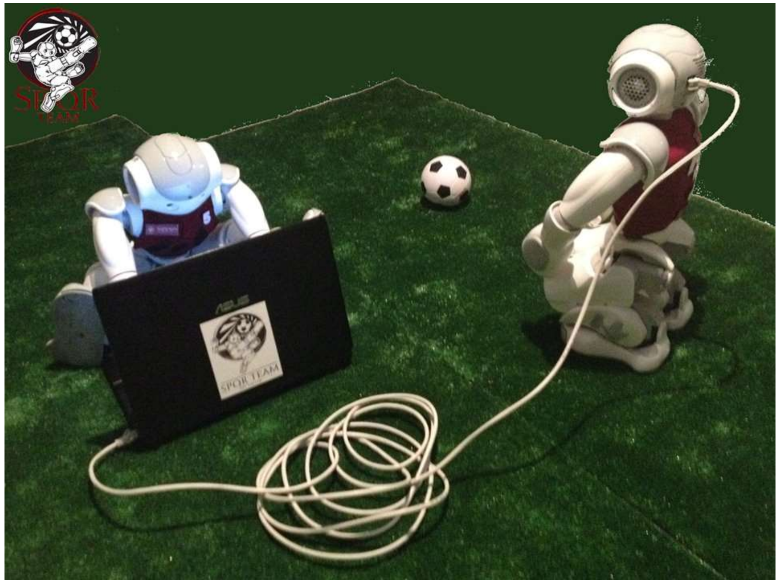


Domenico Daniele Bloisi

- Ricercatore RTD B
  Dipartimento di Matematica, Informatica ed Economia
  Università degli studi della Basilicata
  http://web.unibas.it/bloisi

- SPQR Robot Soccer Team
  Dipartimento di Informatica, Automatica e Gestionale Università degli studi di Roma "La Sapienza"
  http://spqr.diag.uniroma1.it





Informazioni sul corso

- Home page del corso
  http://web.unibas.it/bloisi/corsi/visione-e-percezione.html
- Docente: Domenico Daniele Bloisi
- Periodo: II semestre marzo 2021 – giugno 2021

  Martedì 17:00-19:00 (Aula COPERNICO)
  Mercoledì 8:30-10:30 (Aula COPERNICO)

Codice corso Google Classroom:
https://classroom.google.com/c/NjI2MjA4MzgzNDFa?cjc=xgolays

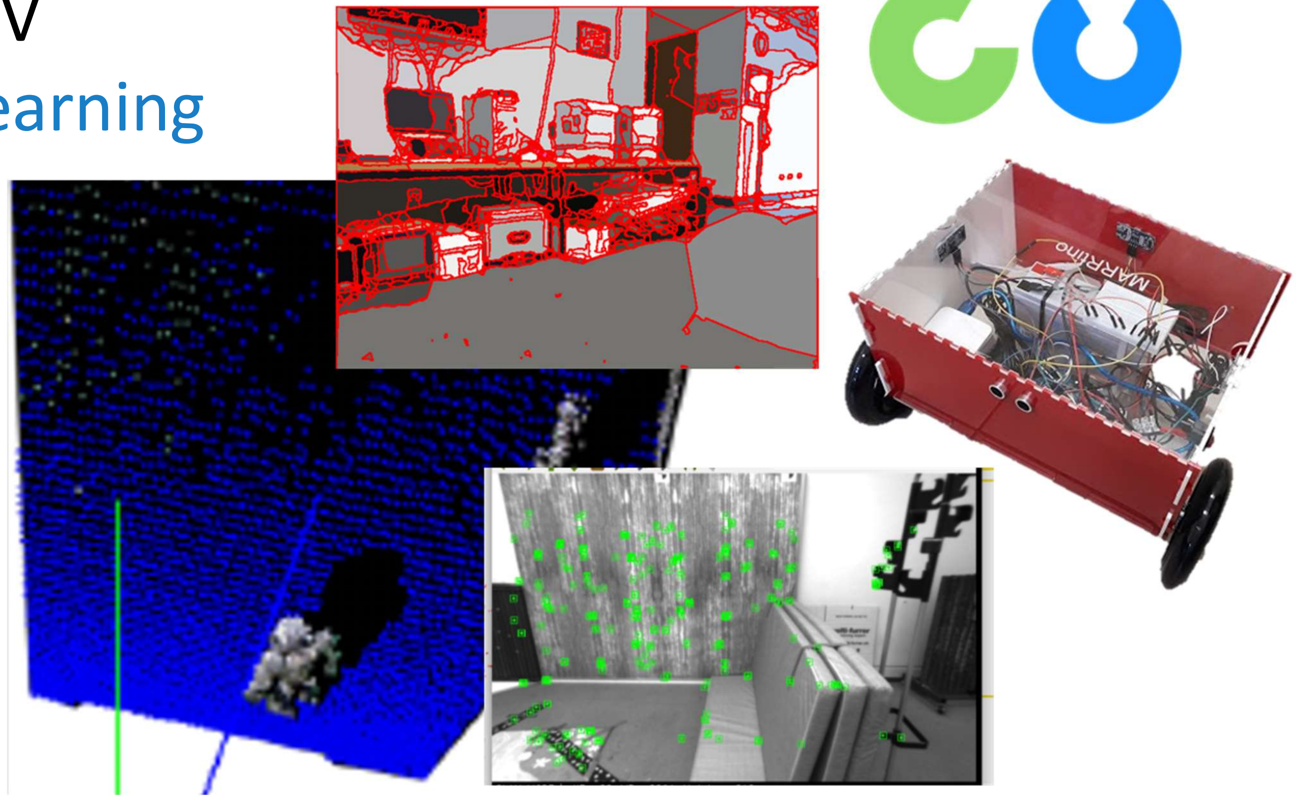
Ricevimento

- Su appuntamento tramite Google Meet

Per prenotare un appuntamento inviare una email a domenico.bloisi@unibas.it

Programma – Visione e Percezione

- Introduzione al linguaggio Python
- Elaborazione delle immagini con Python
- Percezione 2D – OpenCV
- Introduzione al Deep Learning
- ROS
- Il paradigma publisher and subscriber
- Simulatori
- Percezione 3D - PCL



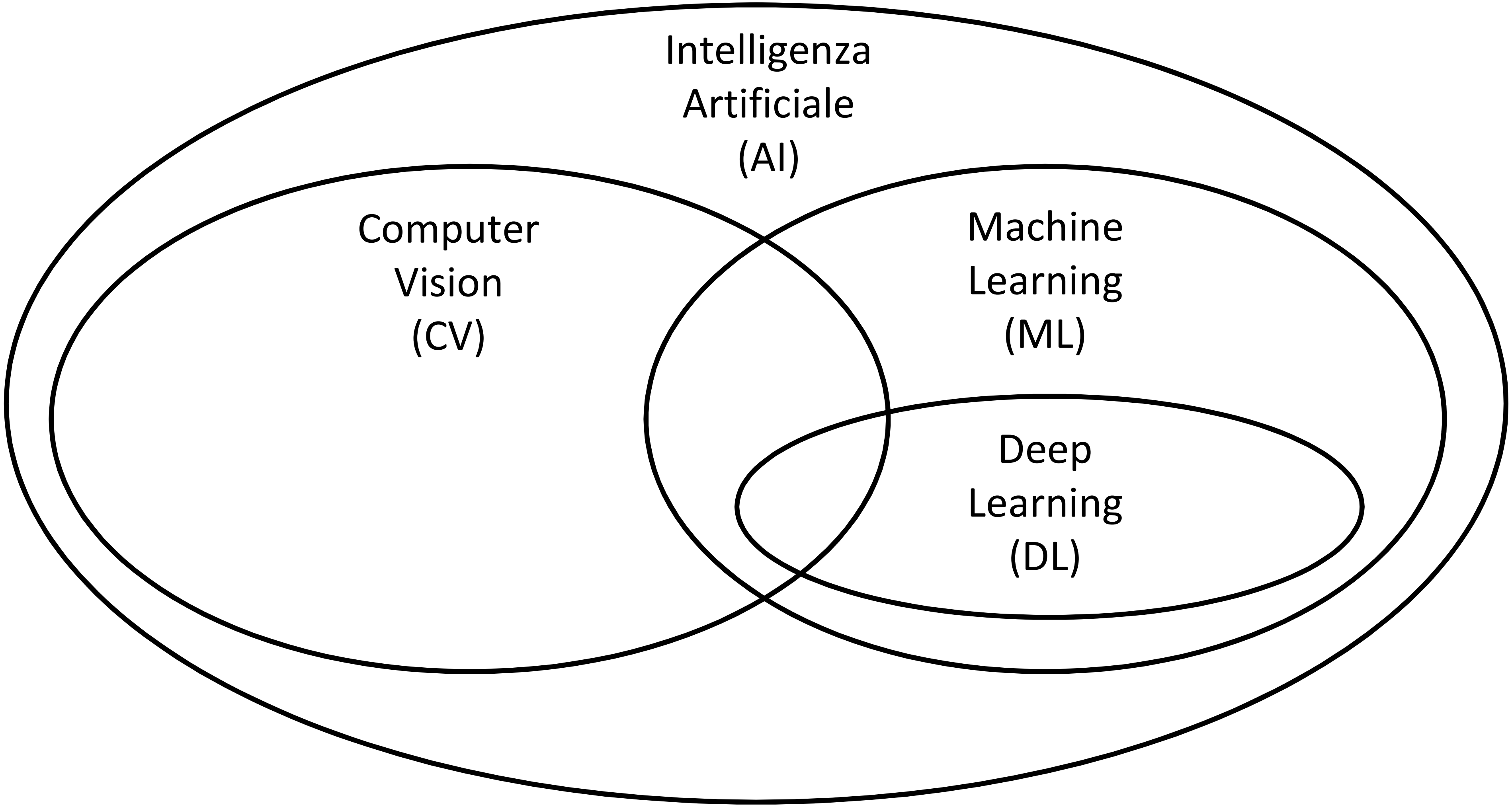
Riferimenti

Queste slide sono basate principalmente su:

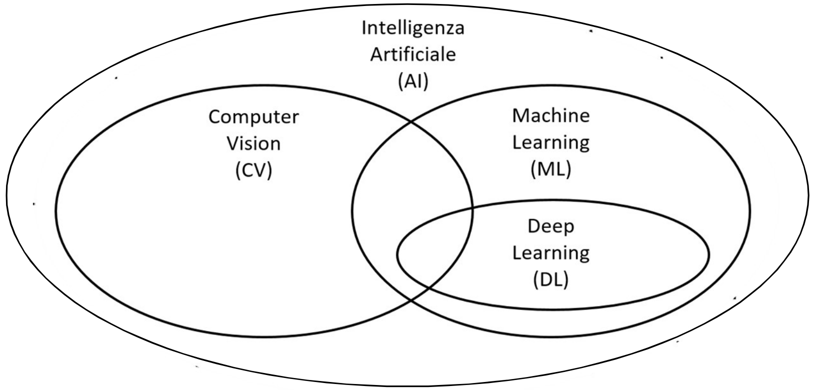
- Martin Görner
  Learn TensorFlow and deep learning, without a Ph.D.
  Video
  https://youtu.be/u4alGiomYP4

- Roberto Capobianco
  Introduction to Neural Networks

AI, CV, ML, and DL



AI

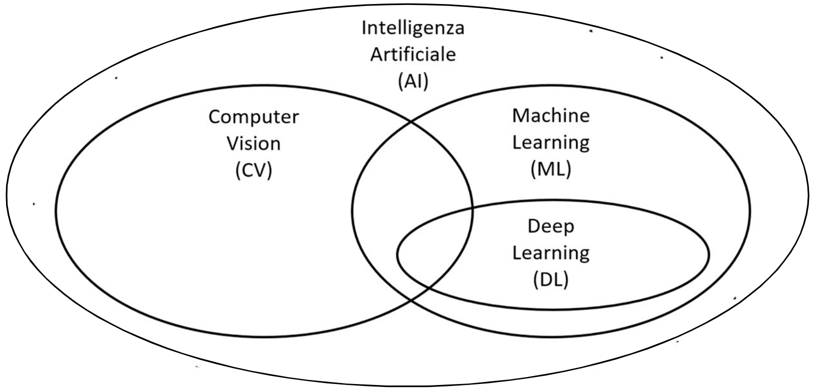


Qual è lo scopo dell'intelligenza artificiale?

"ragionare, prendere decisioni e compiere azioni in modo autonomo, cioè senza che vi sia l'intervento di un operatore umano"

- **Autonomia:** capacità di portare a termine un compito basandosi sullo stato e sulle percezioni correnti, senza intervento umano
- **Sistema autonomo:** un sistema che prende decisioni da solo, agendo senza la guida di un umano

CV



Qual è lo scopo della Computer Vision?

“creare sistemi artificiali che

- *processano*
- *percepiscono*
- *ragionano su*

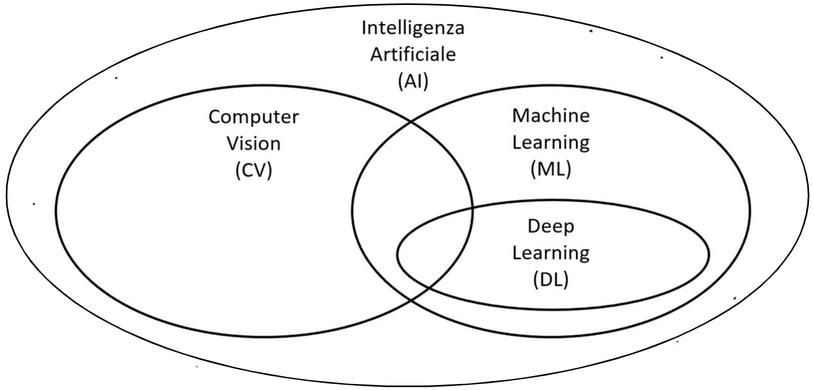
dati visuali”

- Immagini
- Video
- …

- **Instagram:** circa 100 milioni di foto e video caricati al giorno
- **Youtube:** più di 500 ore di video caricate ogni minuto

Source: Justin Johnson

ML



Qual è lo scopo del ML?

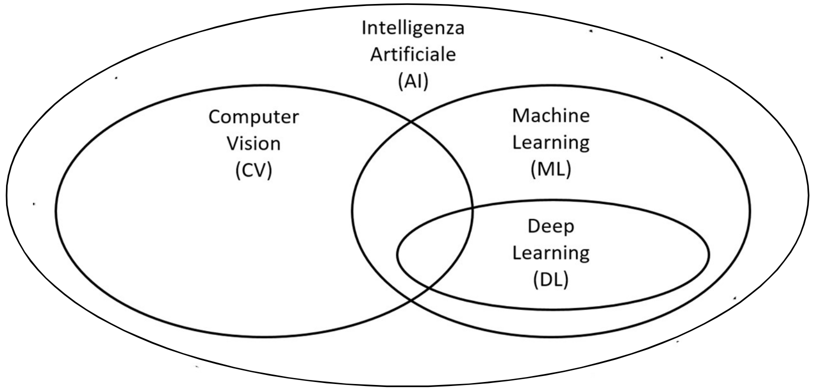
"creare sistemi artificiali che imparano da

- *dati*
- *esperienza"*

Lo scopo del ML è ortogonale rispetto al quello della CV, la quale è interessata a risolvere il problema di interpretare i dati visuali, ma non specifica come deve essere risolto tale problema

Source: Justin Johnson

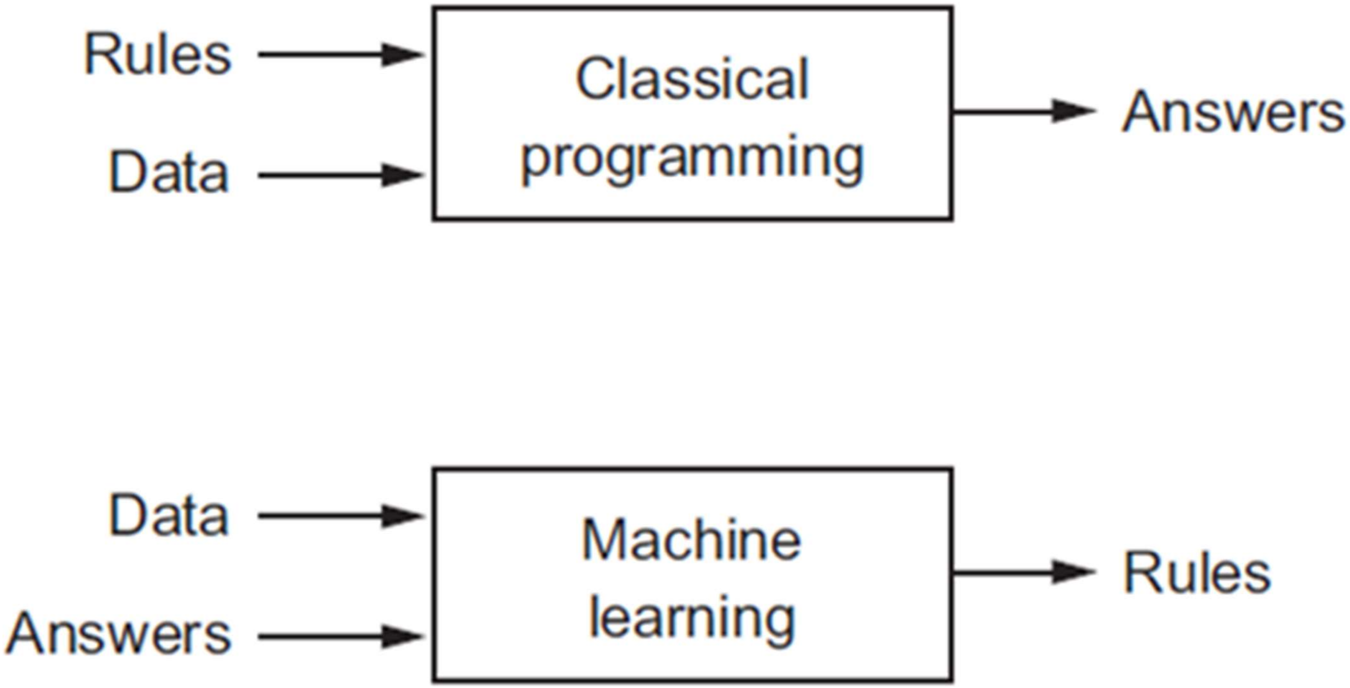
Domande del ML



Il ML nasce per rispondere alle seguenti domande:

- *“può una macchina andare oltre le istruzioni che un umano può fornirle su come svolgere un compito e imparare da sola nuove modalità per svolgere tale compito?”*
- *“può una macchina sorprenderci e risolvere un problema in un modo per noi difficile da immaginare?”*

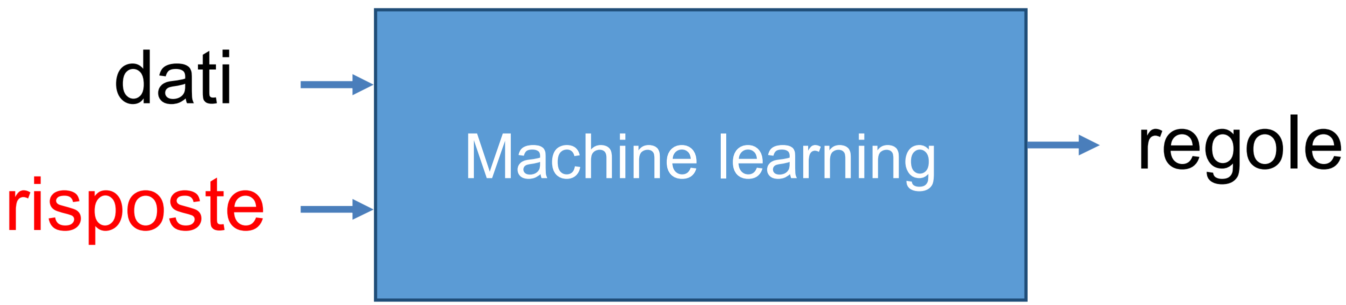
Paradigma del ML



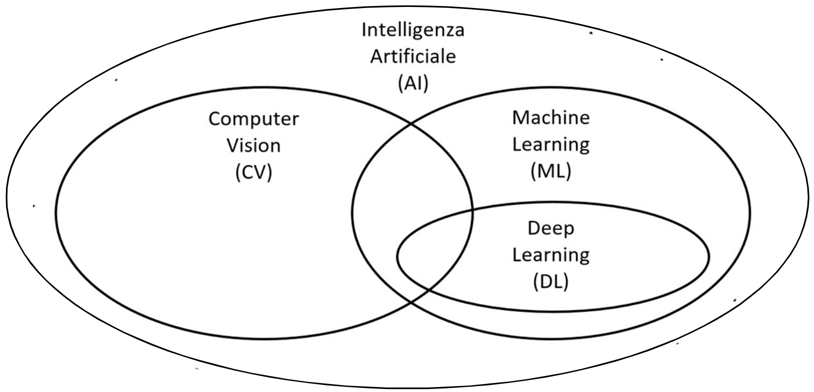
Francois Chollet "Deep Learning with Python"
Manning Publications Co.

Tipi di Apprendimento

- Un sistema di ML viene **addestrato a svolgere un compito** piuttosto che esplicitamente programmato a svolgerlo
- L'apprendimento può avvenire con diverse modalità:
  - **supervisionato** (supervised learning)
  - **semi-supervisionato** (semi-supervised learning)
  - **per rinforzo** (reinforcement learning)

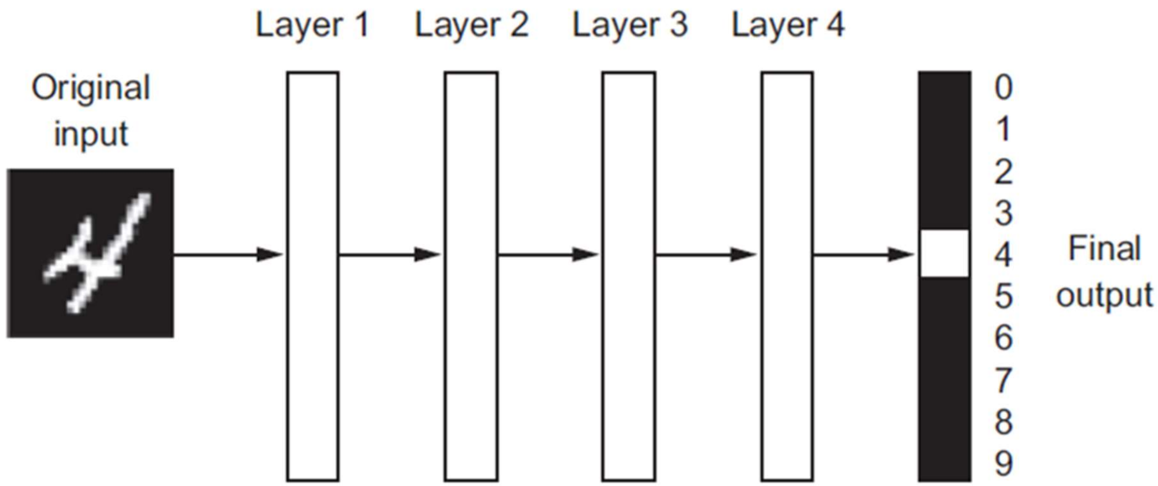


DL



Qual è lo scopo del DL?

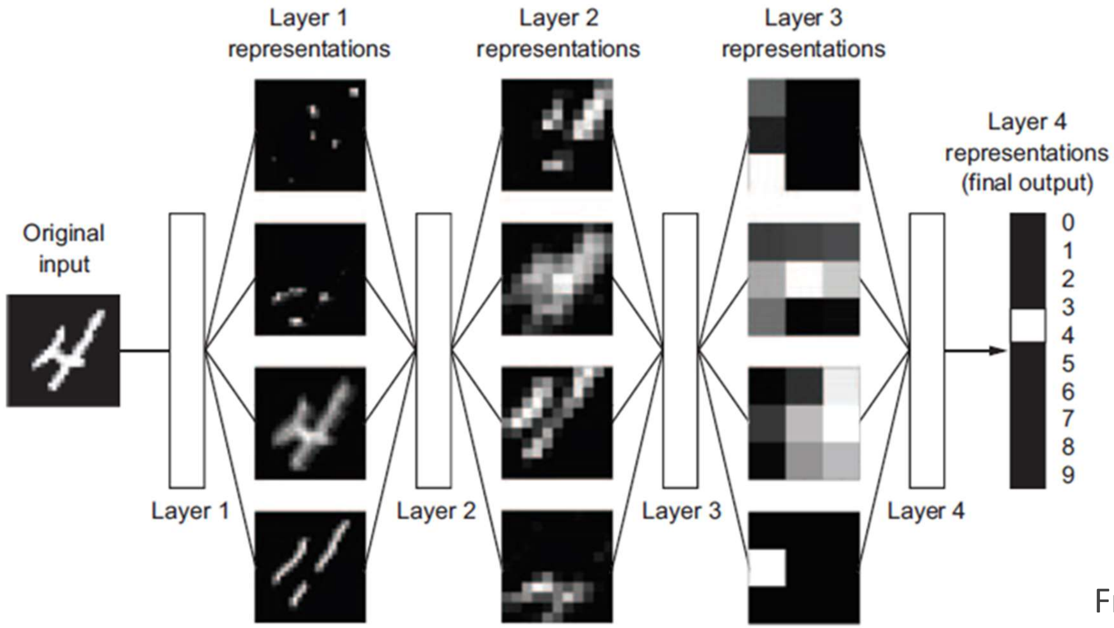
“creare sistemi di apprendimento automatico che abbiano una architettura gerarchica, composta da molti strati (layers), in modo da formare una lunga (deep) catena di rappresentazioni”



Francois Chollet "Deep Learning with Python"
Manning Publications Co.

Low and high level features

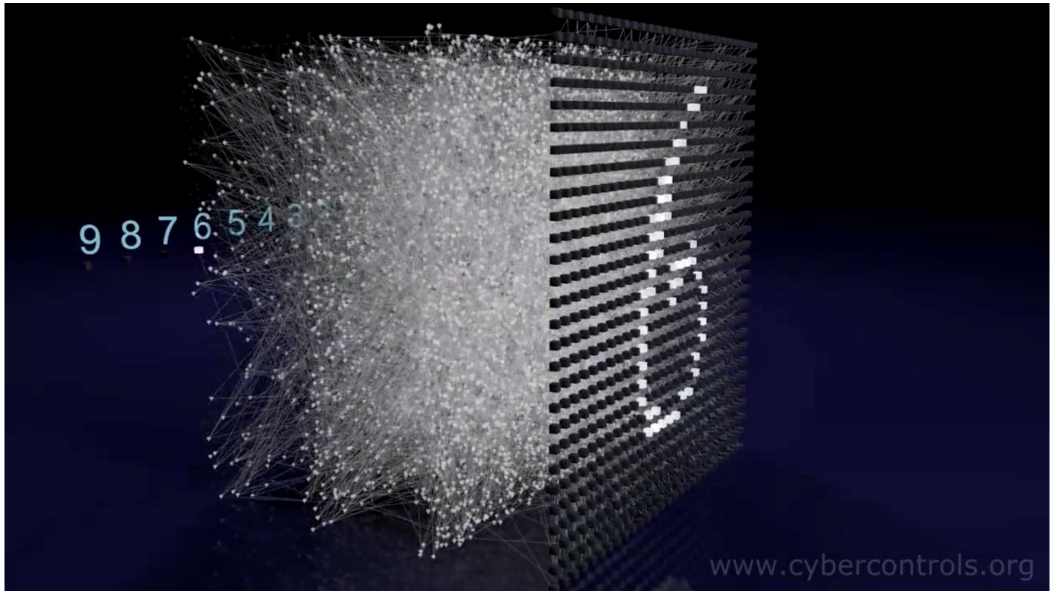
Deep learning methods aim at learning "feature" hierarchies with features from higher levels of the hierarchy formed by the composition of lower level features. [Glorot and Bengio]



Francois Chollet "Deep Learning with Python"
Manning Publications Co.

Deep Neural Networks (DNN)

Reti neurali artificiali organizzate in diversi strati (2 o più), dove ogni strato calcola i valori per quello successivo affinché l'informazione venga elaborata in maniera sempre più completa



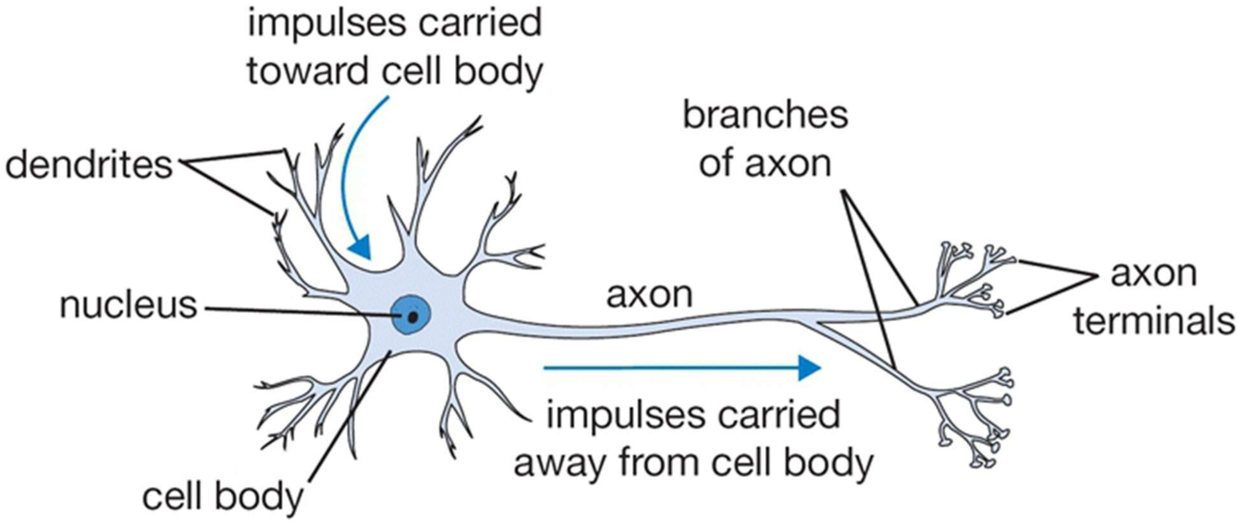
https://vimeo.com/154085950

DNN vs. Mente umana

- Sebbene alcuni concetti presenti nelle reti neurali siano stati sviluppati prendendo ispirazione dalle teorie sul funzionamento della mente umana, i modelli utilizzati nel deep learning non hanno nulla a che fare con il funzionamento del cervello umano
- **Non ci sono evidenze che possano accomunare il funzionamento delle reti neurali usate per la creazione tramite Deep Learning di modelli predittivi con i meccanismi cognitivi del cervello umano**

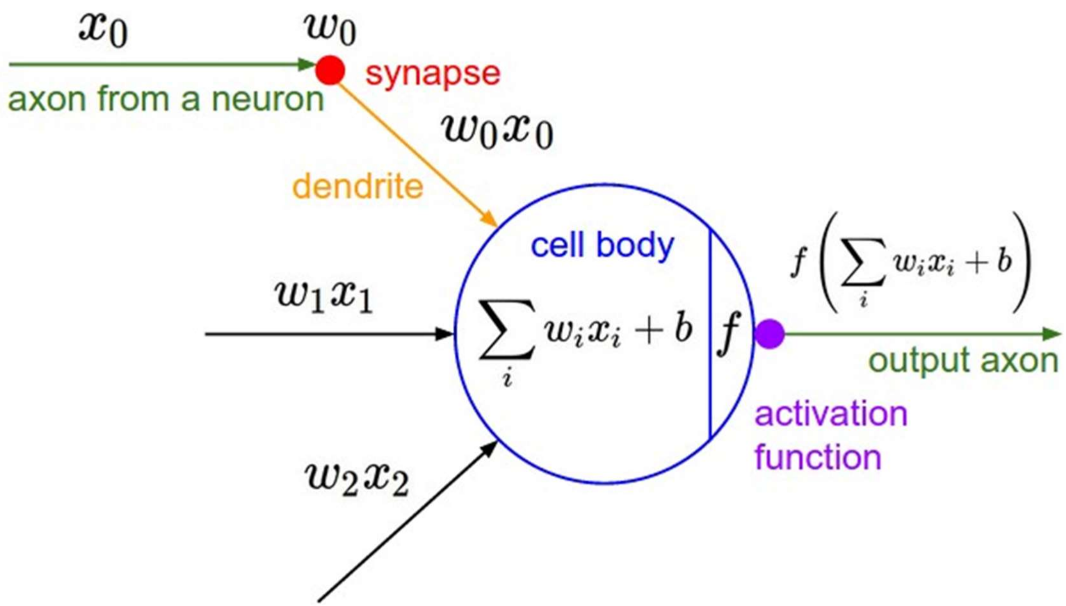
Neural Networks

- Initial goal: model biological neural systems
  - basic computational unit: neuron
  - ~86 billion neurons in the human nervous system
  - connected with ~10^{14k}-10^{15} synapses
  - signals on axons interact multiplicatively with dendrites of other neurons based on some synaptic strength



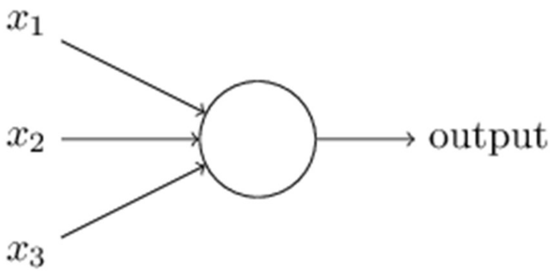
Neural Networks: Implementation

- Diverged from biological model
  - engineered to achieve good results in ML tasks (different from real neurons!)
  - idea: synaptic strengths can be learned
  - model: dendrites carry signals that get summed in the cell body; if sum is above some threshold neuron fires
  - neurons fire with a frequency that depends on the activation function



Perceptron

A perceptron takes several binary inputs, $x_1, x_2, \ldots$, computes a weighted sum of the inputs and produces a single binary output using a fixed threshold:

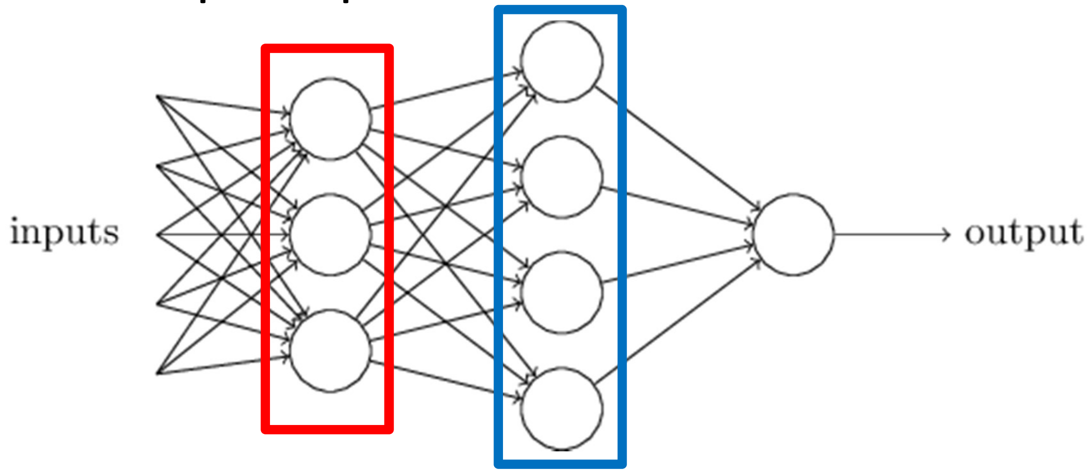


$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{ threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{ threshold} \end{cases}$$

We can use the perceptron to take decisions: by varying the weights and the threshold, we can get different models of decision-making.

Multi-level perceptrons

More complex networks of perceptrons can deal with more complex decision problems:



The first column (i.e., the **first layer**)of perceptrons is making simple, low level decisions, by directly weighing the inputs.

The perceptrons in the **second layer** is making a decision by weighing the results from the first layer:

The second layer **can make a decision at a more complex and more abstract level**.

From perceptrons to artificial neurons

1) Write the weighted sum as dot product

$$w \cdot x$$

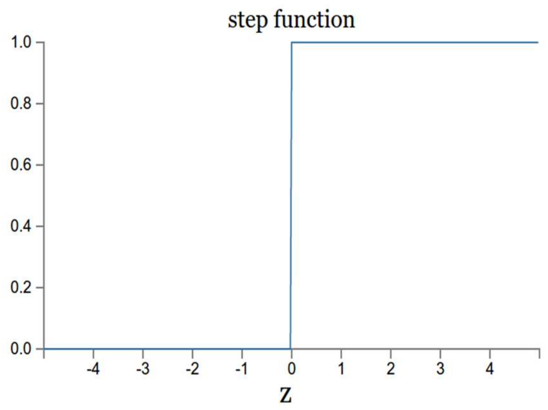
2) Replace the threshold with a bias *b*, where

b = -threshold

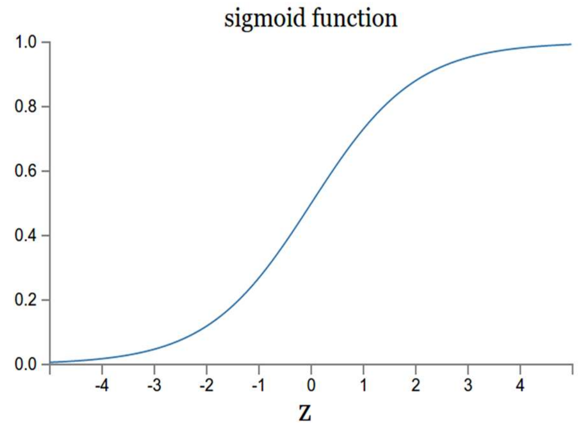
$$\text{output} = \begin{cases} 0 & \text{if } w \cdot x + b \leq 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases}$$

Sigmoid Neuron

3) "Smooth" the output using the sigmoid function as **activation function**



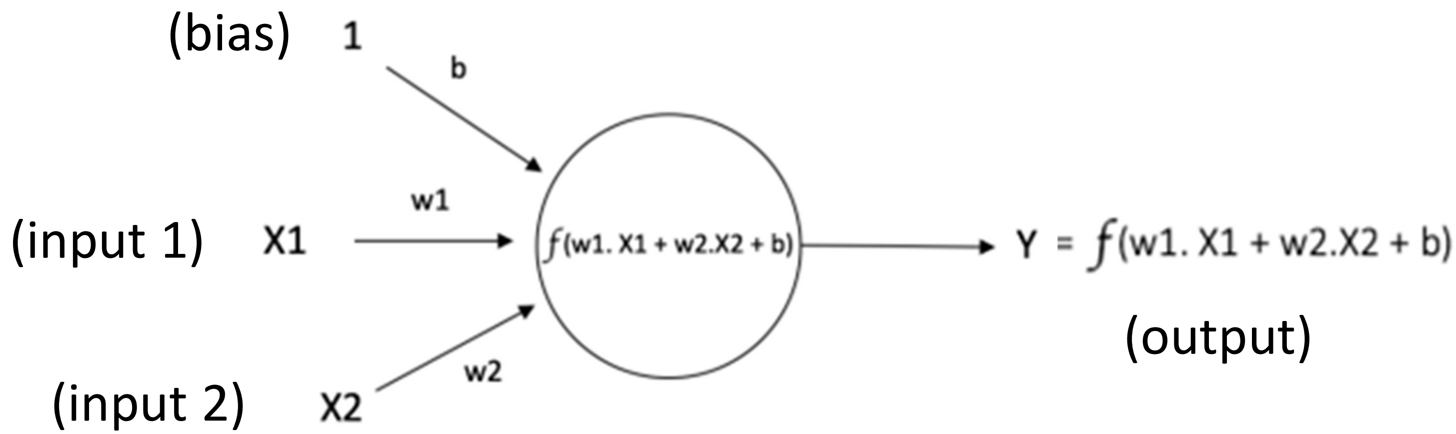
$$\sigma(z) \equiv \frac{1}{1+e^{-z}}$$



Now we have a **sigmoid neuron** → small changes in the weights and bias cause only a small change in the output → That is the crucial fact which will allow a network of sigmoid neurons to *learn*

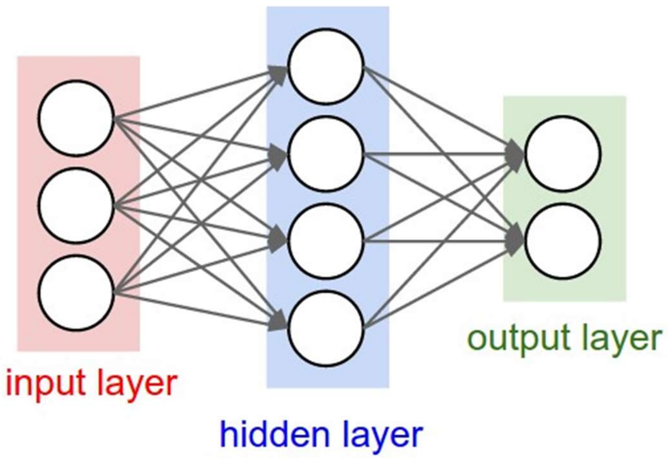
Artificial Neuron

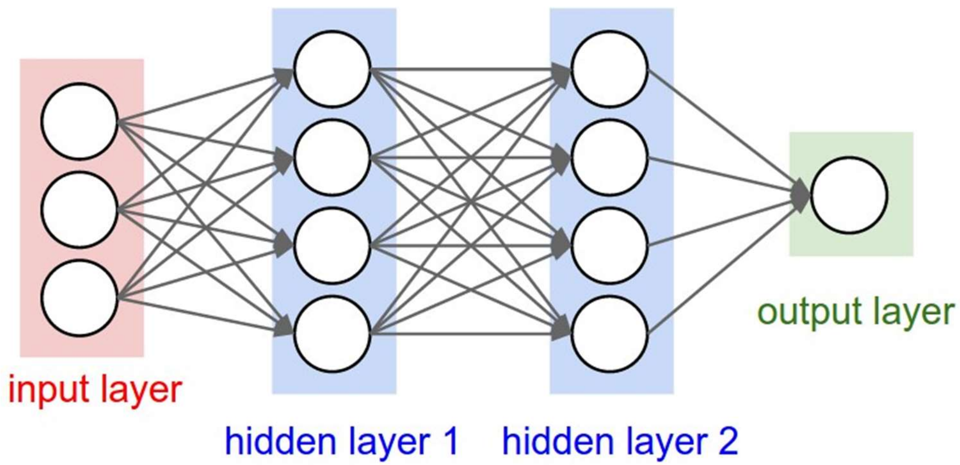
- takes numerical **inputs** (x)
- has a **weight** associated to each input (w)
- has a **bias** in the form of an additional input 1 with weight b
- applies an activation function (f) to the weighted sum of inputs



Network Architecture

- regular neural networks are neurons connected in an acyclic graph
- 1 or more layers
- typically fully connected layers (no connection inside the same layer)
- output layer typically without activation function
- naming convention: input layer is not counted
    - single-layer networks directly map input to output





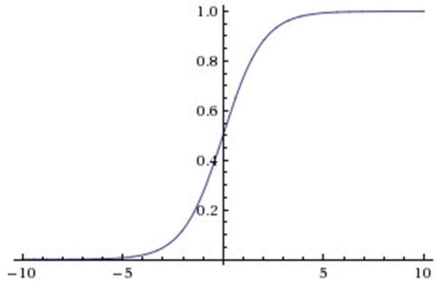
Activation Function

- It maps the resulting values into the desired range
- typically non-linear, aims at introducing non-linearity in the output of a neuron
- takes numbers as input
- performs a fixed mathematical operation on it

Activation Functions examples

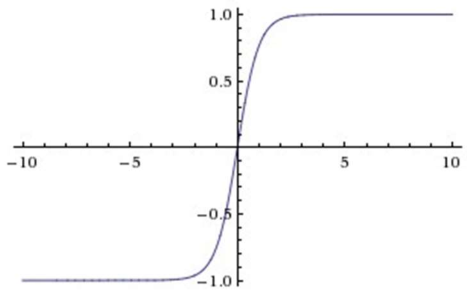
- **sigmoid** (bad!): takes a real-valued input and squashes it to range between 0 and 1

$$\sigma(x) = 1 / (1 + \exp(-x))$$



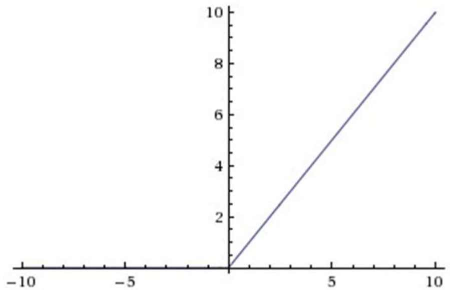
- **tanh**: takes a real-valued input and squashes it to the range [-1, 1]

$$\tanh(x) = 2\sigma(2x) - 1$$



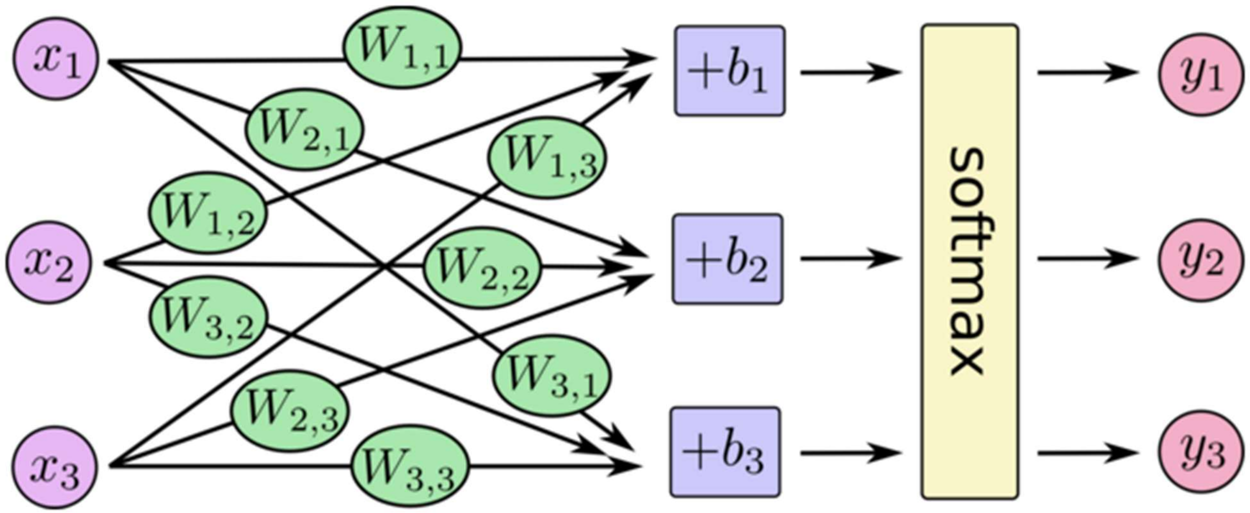
- **ReLU**: ReLU stands for Rectified Linear Unit. It takes a real-valued input and thresholds it at zero (replaces negative values with zero)

$$f(x) = \max(0, x)$$



Softmax function

The **softmax** function takes a vector of arbitrary real-valued scores (in Z) and squashes it to a vector of values between 0 and 1 that sums to 1

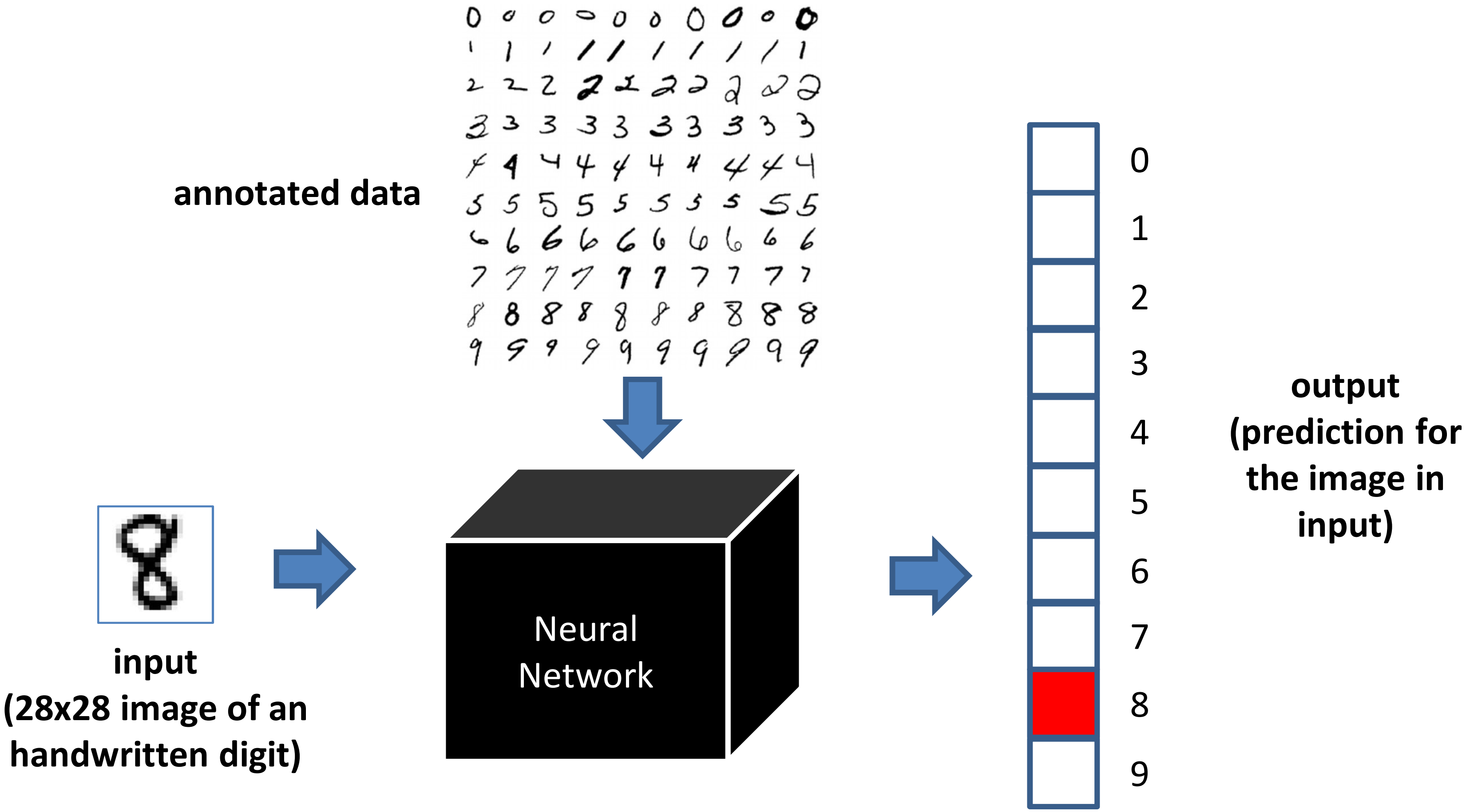


Softmax

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \text{softmax}\left(\begin{bmatrix} W_{1,1} & W_{1,2} & W_{1,3} \\ W_{2,1} & W_{2,2} & W_{2,3} \\ W_{3,1} & W_{3,2} & W_{3,3} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \right)$$

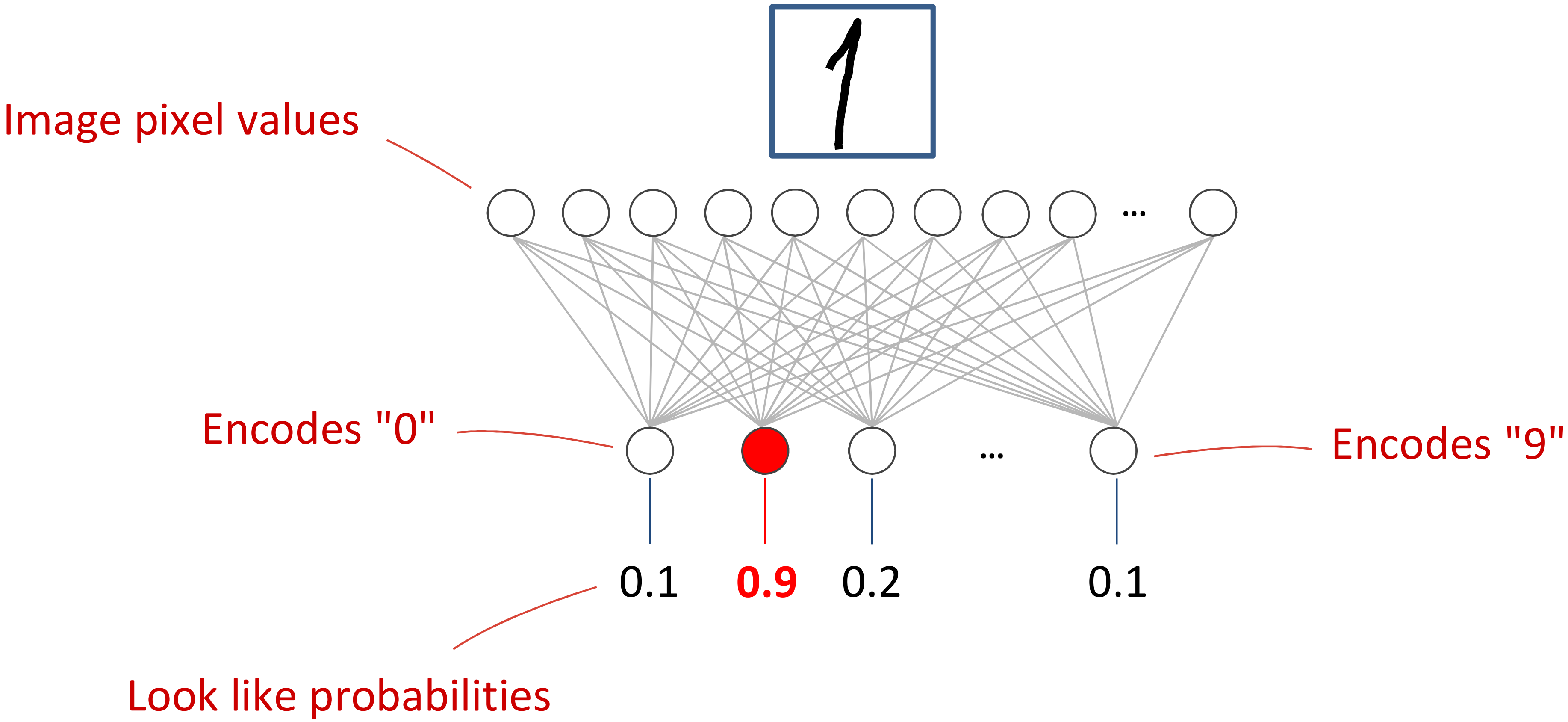
The output of a softmax layer depends on the outputs of **all** the other neurons in its layer

Problem: Handwritten digit classification

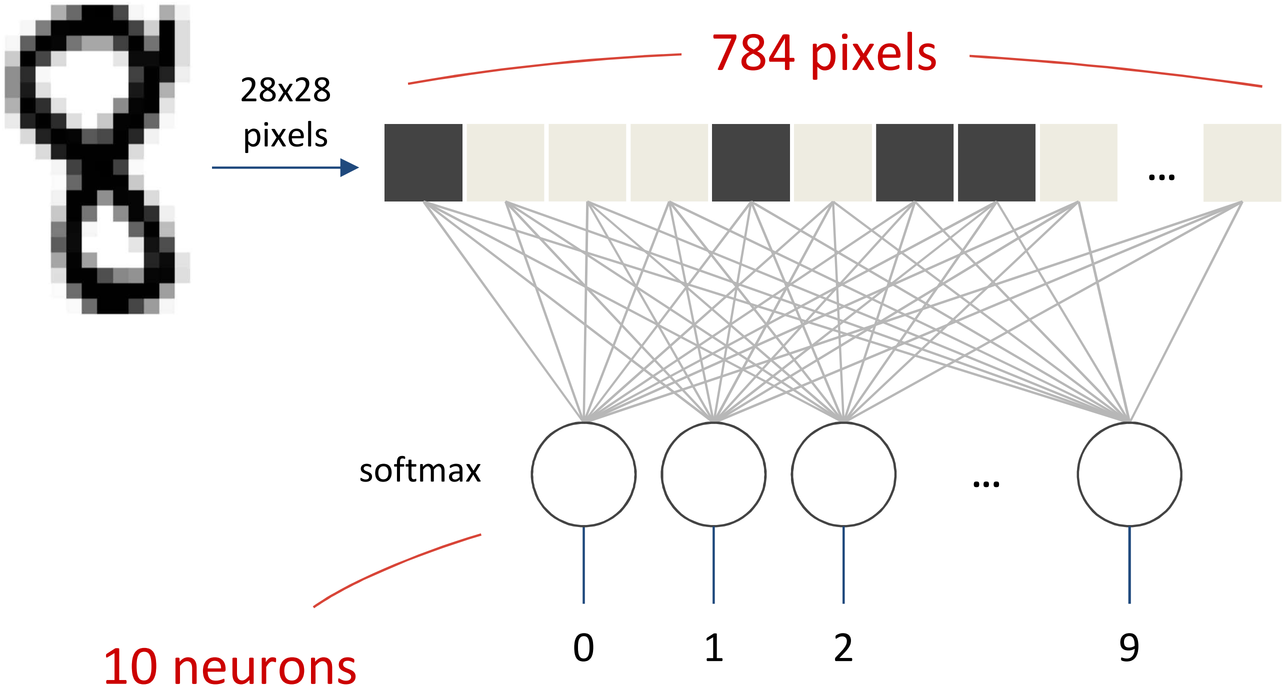


MNIST = Mixed National Institute of Standards and Technology - Download the dataset at http://yann.lecun.com/exdb/mnist/

Simple solution: Single layer network



Softmax classification



$$L = X.W + b$$

weighted sum of all pixels + bias

$$softmax(L_n) = \frac{e^{L_n}}{||e^L||}$$

neuron outputs

Martin Görner
Learn TensorFlow and deep learning, without a Ph.D.

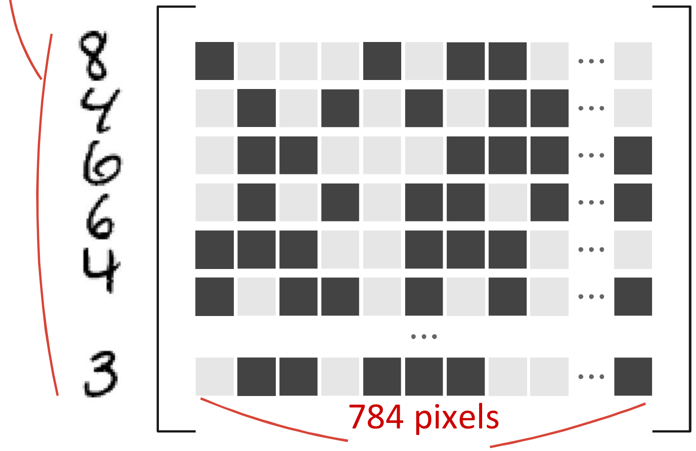
MiniBatch

number of samples to work through before updating the internal model parameters

$$
\begin{array}{cc}
\text{10 columns} & \\
\begin{bmatrix}
w_{0,0} & w_{0,1} & w_{0,2} & w_{0,3} & \cdots & w_{0,9} \\
w_{1,0} & w_{1,1} & w_{1,2} & w_{1,3} & \cdots & w_{1,9} \\
w_{2,0} & w_{2,1} & w_{2,2} & w_{2,3} & \cdots & w_{2,9} \\
w_{3,0} & w_{3,1} & w_{3,2} & w_{3,3} & \cdots & w_{3,9} \\
w_{4,0} & w_{4,1} & w_{4,2} & w_{4,3} & \cdots & w_{4,9} \\
w_{5,0} & w_{5,1} & w_{5,2} & w_{5,3} & \cdots & w_{5,9} \\
w_{6,0} & w_{6,1} & w_{6,2} & w_{6,3} & \cdots & w_{6,9} \\
w_{7,0} & w_{7,1} & w_{7,2} & w_{7,3} & \cdots & w_{7,9} \\
w_{8,0} & w_{8,1} & w_{8,2} & w_{8,3} & \cdots & w_{8,9} \\
 & \cdots & & & & \\
w_{783,0} & w_{783,1} & w_{783,2} & \cdots & w_{783,9} &
\end{bmatrix} & \text{784 lines}
\end{array}
$$

$$L = X.W + b$$

X → 100 images, one per line, flattened



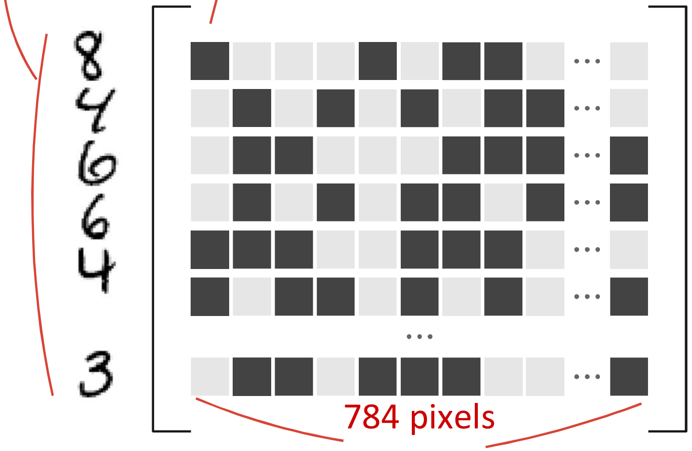
Martin Görner

[Learn TensorFlow and deep learning, without a Ph.D.]

MiniBatch

number of samples to work through before updating the internal model parameters

X → 100 images, one per line, flattened



x

$$\begin{array}{c} \text{10 columns} \\ \left[\begin{array}{cccccc} w_{0,0} & w_{0,1} & w_{0,2} & w_{0,3} & \cdots & w_{0,9} \\ w_{1,0} & w_{1,1} & w_{1,2} & w_{1,3} & \cdots & w_{1,9} \\ w_{2,0} & w_{2,1} & w_{2,2} & w_{2,3} & \cdots & w_{2,9} \\ w_{3,0} & w_{3,1} & w_{3,2} & w_{3,3} & \cdots & w_{3,9} \\ w_{4,0} & w_{4,1} & w_{4,2} & w_{4,3} & \cdots & w_{4,9} \\ w_{5,0} & w_{5,1} & w_{5,2} & w_{5,3} & \cdots & w_{5,9} \\ w_{6,0} & w_{6,1} & w_{6,2} & w_{6,3} & \cdots & w_{6,9} \\ w_{7,0} & w_{7,1} & w_{7,2} & w_{7,3} & \cdots & w_{7,9} \\ w_{8,0} & w_{8,1} & w_{8,2} & w_{8,3} & \cdots & w_{8,9} \\ & \cdots & & & & \\ w_{783,0} & w_{783,1} & w_{783,2} & \cdots & & w_{783,9} \end{array}\right] \text{784 lines} \end{array}$$

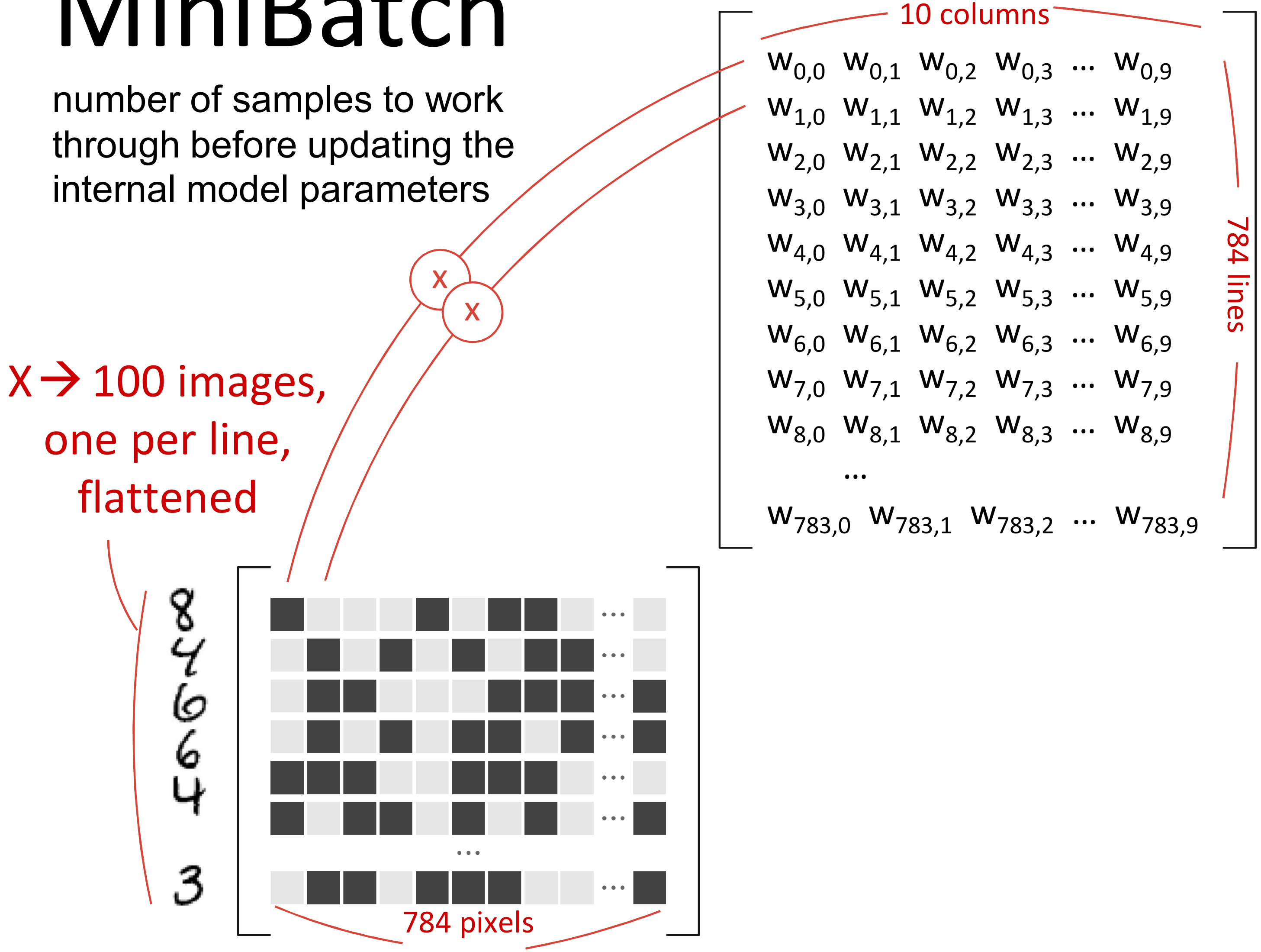
$$L = X.W + b$$

Martin Görner

[Learn TensorFlow and deep learning, without a Ph.D.]

MiniBatch

number of samples to work through before updating the internal model parameters



$$L = X.W + b$$

Martin Görner

[Learn TensorFlow and deep learning, without a Ph.D.](#)

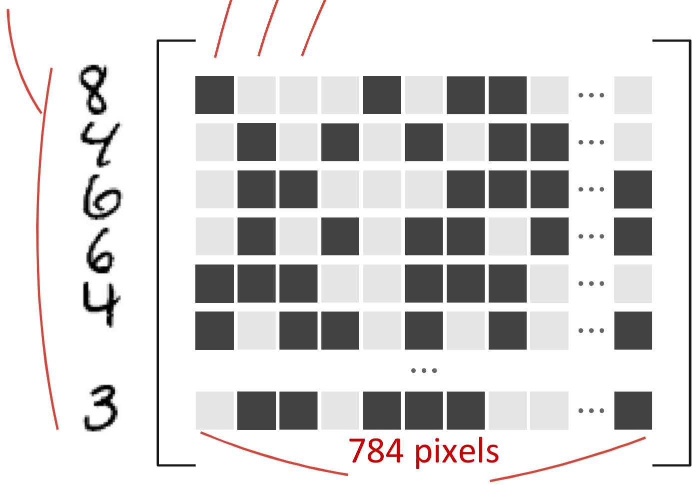
MiniBatch

number of samples to work through before updating the internal model parameters

$$\begin{array}{cc} \text{10 columns} & \\ \left[\begin{array}{cccccc} w_{0,0} & w_{0,1} & w_{0,2} & w_{0,3} & \cdots & w_{0,9} \\ w_{1,0} & w_{1,1} & w_{1,2} & w_{1,3} & \cdots & w_{1,9} \\ w_{2,0} & w_{2,1} & w_{2,2} & w_{2,3} & \cdots & w_{2,9} \\ w_{3,0} & w_{3,1} & w_{3,2} & w_{3,3} & \cdots & w_{3,9} \\ w_{4,0} & w_{4,1} & w_{4,2} & w_{4,3} & \cdots & w_{4,9} \\ w_{5,0} & w_{5,1} & w_{5,2} & w_{5,3} & \cdots & w_{5,9} \\ w_{6,0} & w_{6,1} & w_{6,2} & w_{6,3} & \cdots & w_{6,9} \\ w_{7,0} & w_{7,1} & w_{7,2} & w_{7,3} & \cdots & w_{7,9} \\ w_{8,0} & w_{8,1} & w_{8,2} & w_{8,3} & \cdots & w_{8,9} \\ & \cdots & & & & \\ w_{783,0} & w_{783,1} & w_{783,2} & & \cdots & w_{783,9} \end{array}\right] & \text{784 lines} \end{array}$$

$$L = X.W + b$$

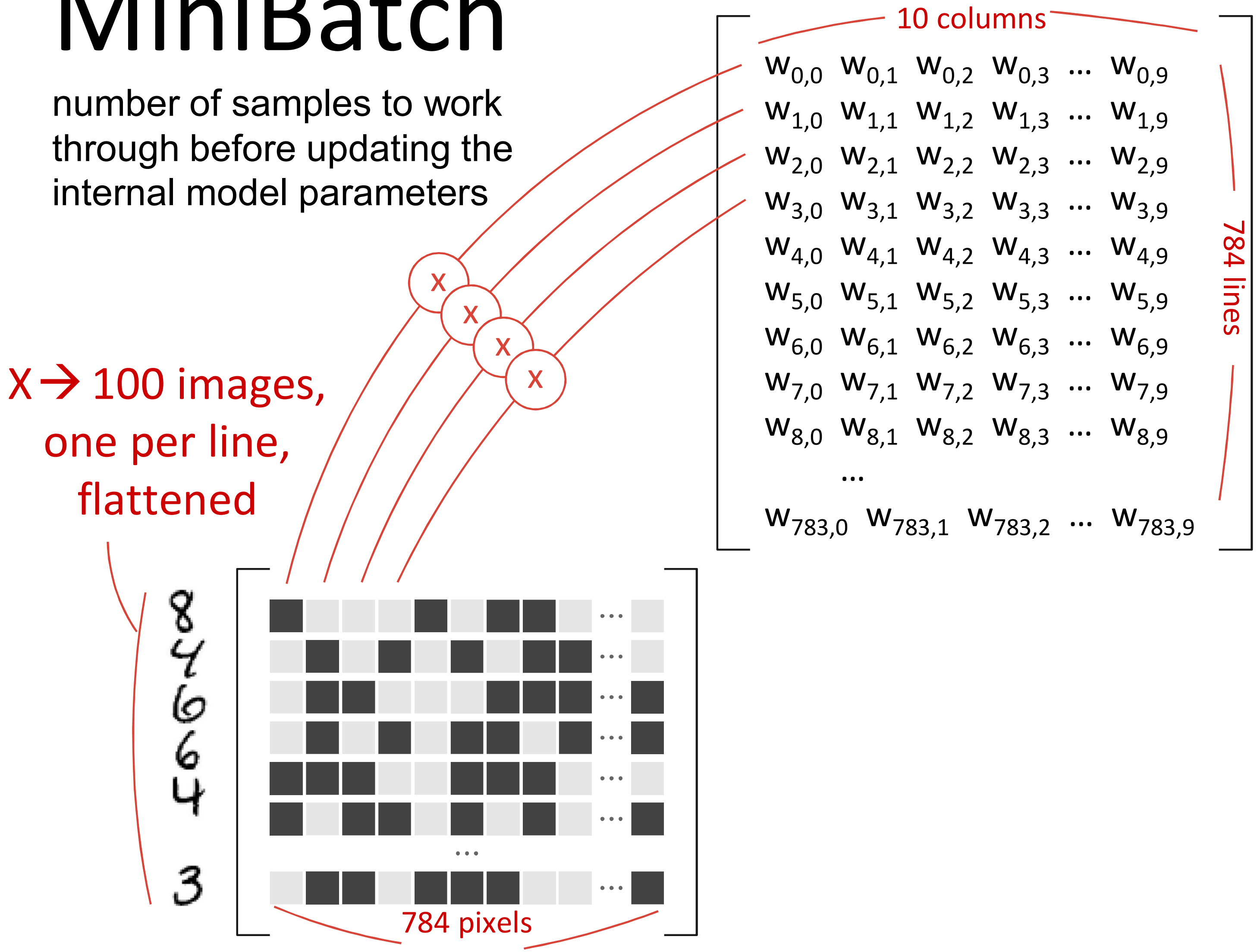
X → 100 images, one per line, flattened



Martin Görner
[Learn TensorFlow and deep learning, without a Ph.D.]

MiniBatch

number of samples to work through before updating the internal model parameters



$$L = X.W + b$$

Martin Görner

Learn TensorFlow and deep learning, without a Ph.D.

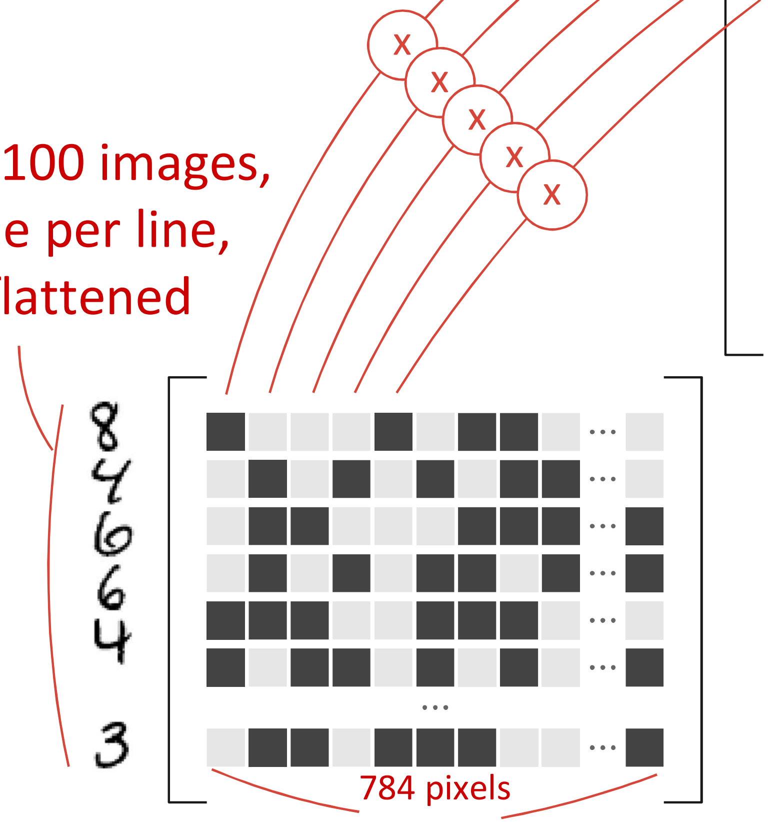
MiniBatch

number of samples to work through before updating the internal model parameters

$$
\begin{array}{c}
\text{10 columns} \\
\left[\begin{array}{cccccc}
w_{0,0} & w_{0,1} & w_{0,2} & w_{0,3} & \cdots & w_{0,9} \\
w_{1,0} & w_{1,1} & w_{1,2} & w_{1,3} & \cdots & w_{1,9} \\
w_{2,0} & w_{2,1} & w_{2,2} & w_{2,3} & \cdots & w_{2,9} \\
w_{3,0} & w_{3,1} & w_{3,2} & w_{3,3} & \cdots & w_{3,9} \\
w_{4,0} & w_{4,1} & w_{4,2} & w_{4,3} & \cdots & w_{4,9} \\
w_{5,0} & w_{5,1} & w_{5,2} & w_{5,3} & \cdots & w_{5,9} \\
w_{6,0} & w_{6,1} & w_{6,2} & w_{6,3} & \cdots & w_{6,9} \\
w_{7,0} & w_{7,1} & w_{7,2} & w_{7,3} & \cdots & w_{7,9} \\
w_{8,0} & w_{8,1} & w_{8,2} & w_{8,3} & \cdots & w_{8,9} \\
 & \cdots & & & & \\
w_{783,0} & w_{783,1} & w_{783,2} & \cdots & w_{783,9} &
\end{array}\right] \text{784 lines}
\end{array}
$$

$$L = X.W + b$$

X → 100 images, one per line, flattened

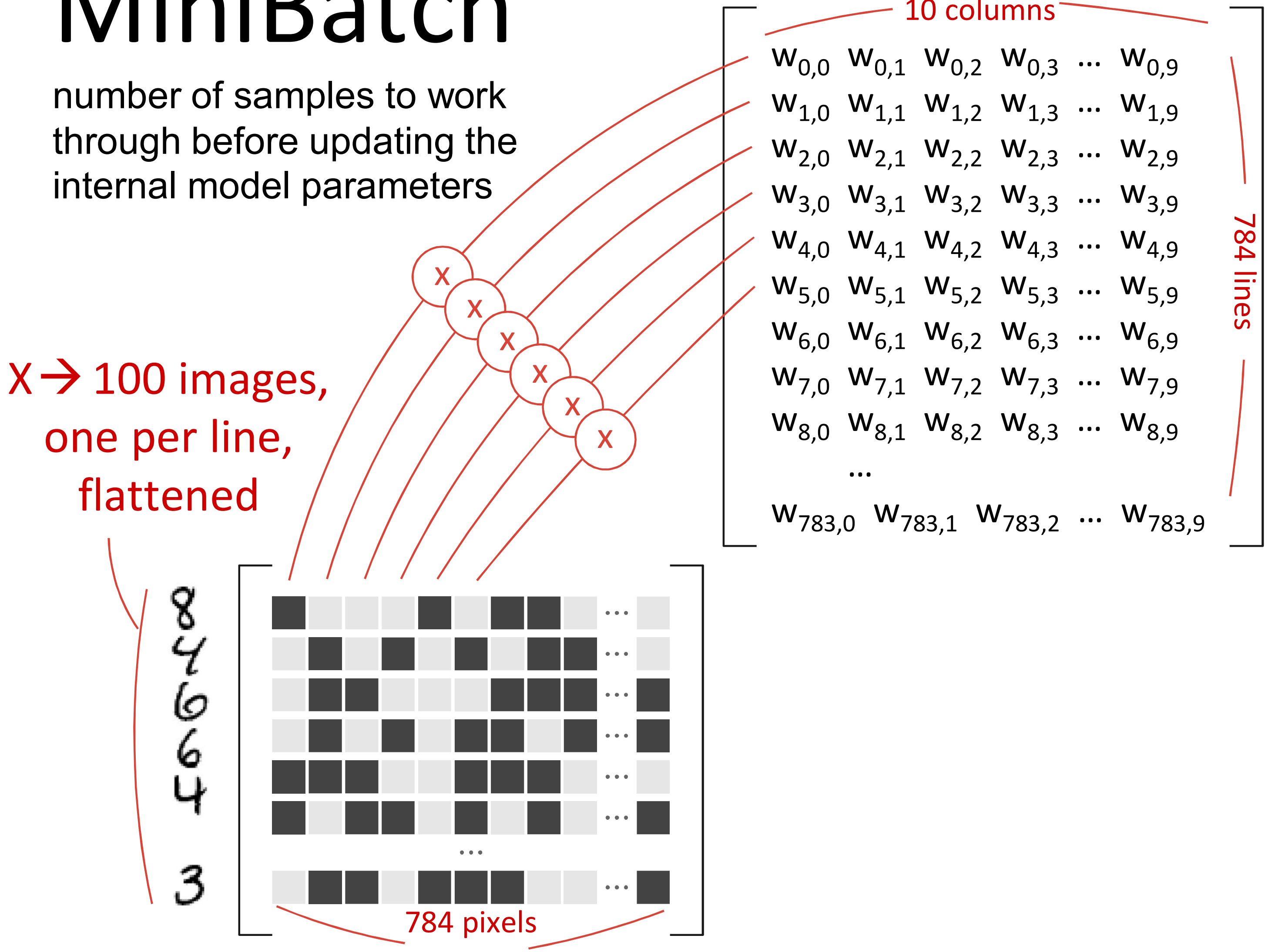


Martin Görner

[Learn TensorFlow and deep learning, without a Ph.D.]

MiniBatch

number of samples to work through before updating the internal model parameters



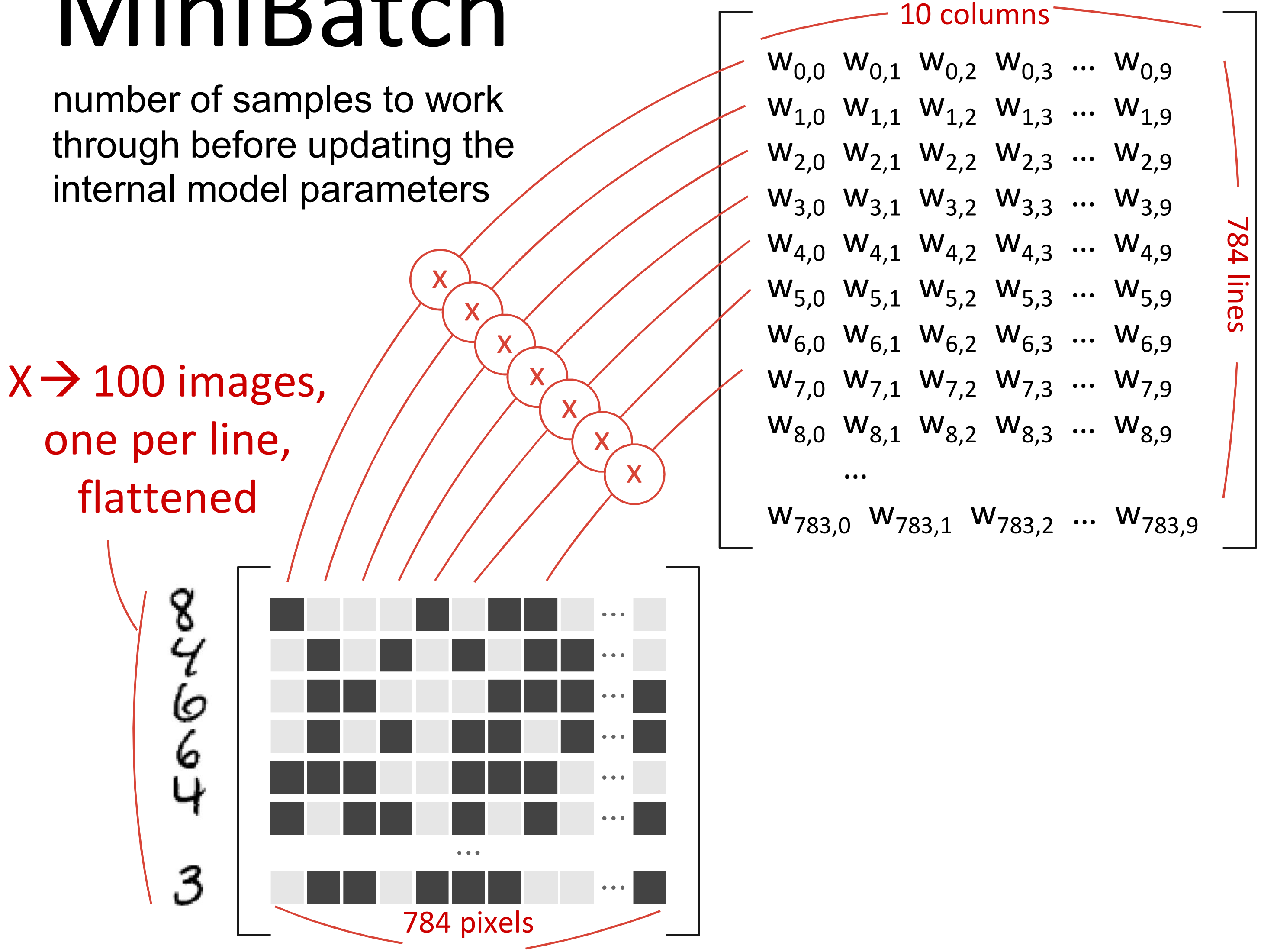
$$L = X.W + b$$

Martin Görner

[Learn TensorFlow and deep learning, without a Ph.D.]

MiniBatch

number of samples to work through before updating the internal model parameters



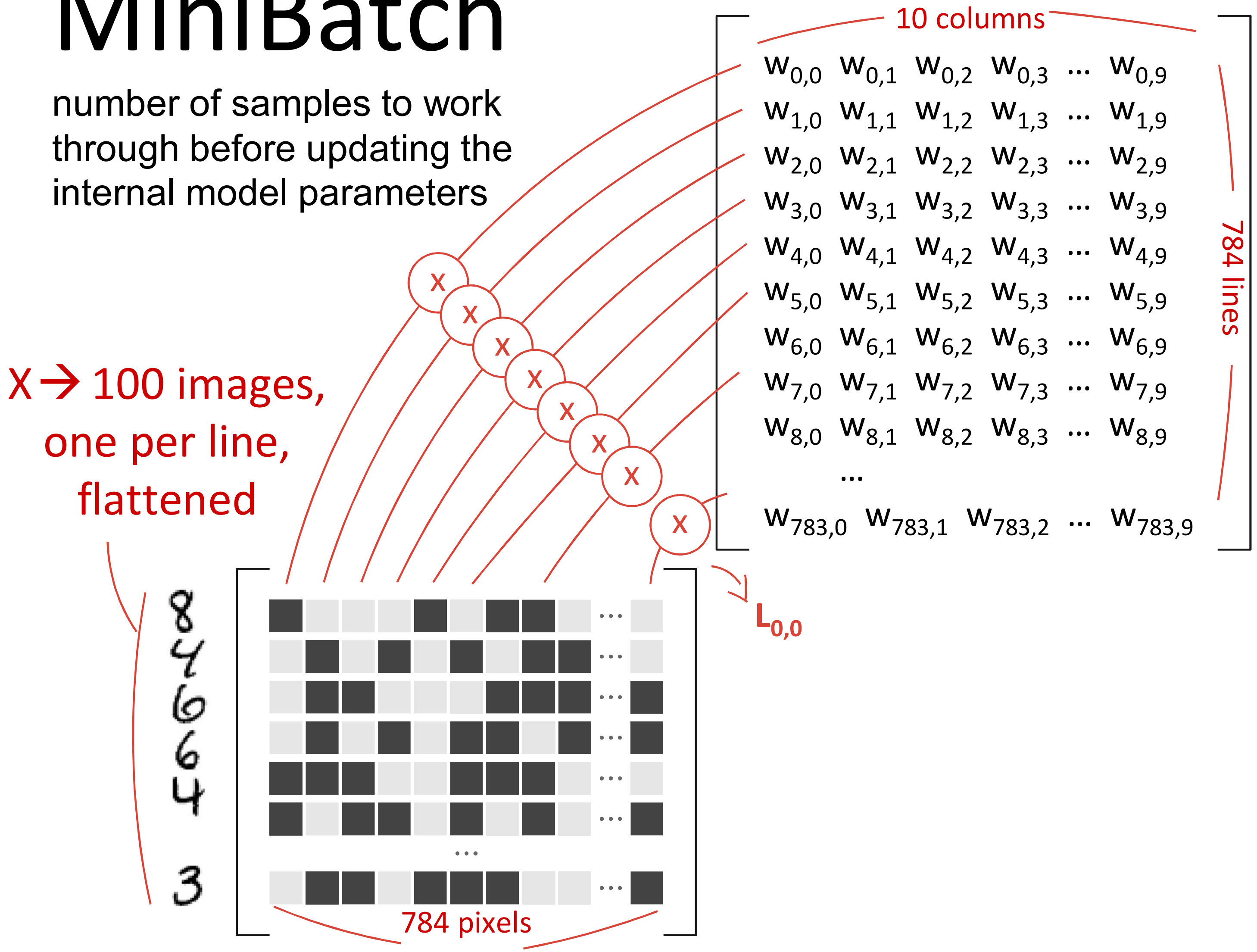
$$L = X.W + b$$

Martin Görner

[Learn TensorFlow and deep learning, without a Ph.D.]

MiniBatch

number of samples to work through before updating the internal model parameters



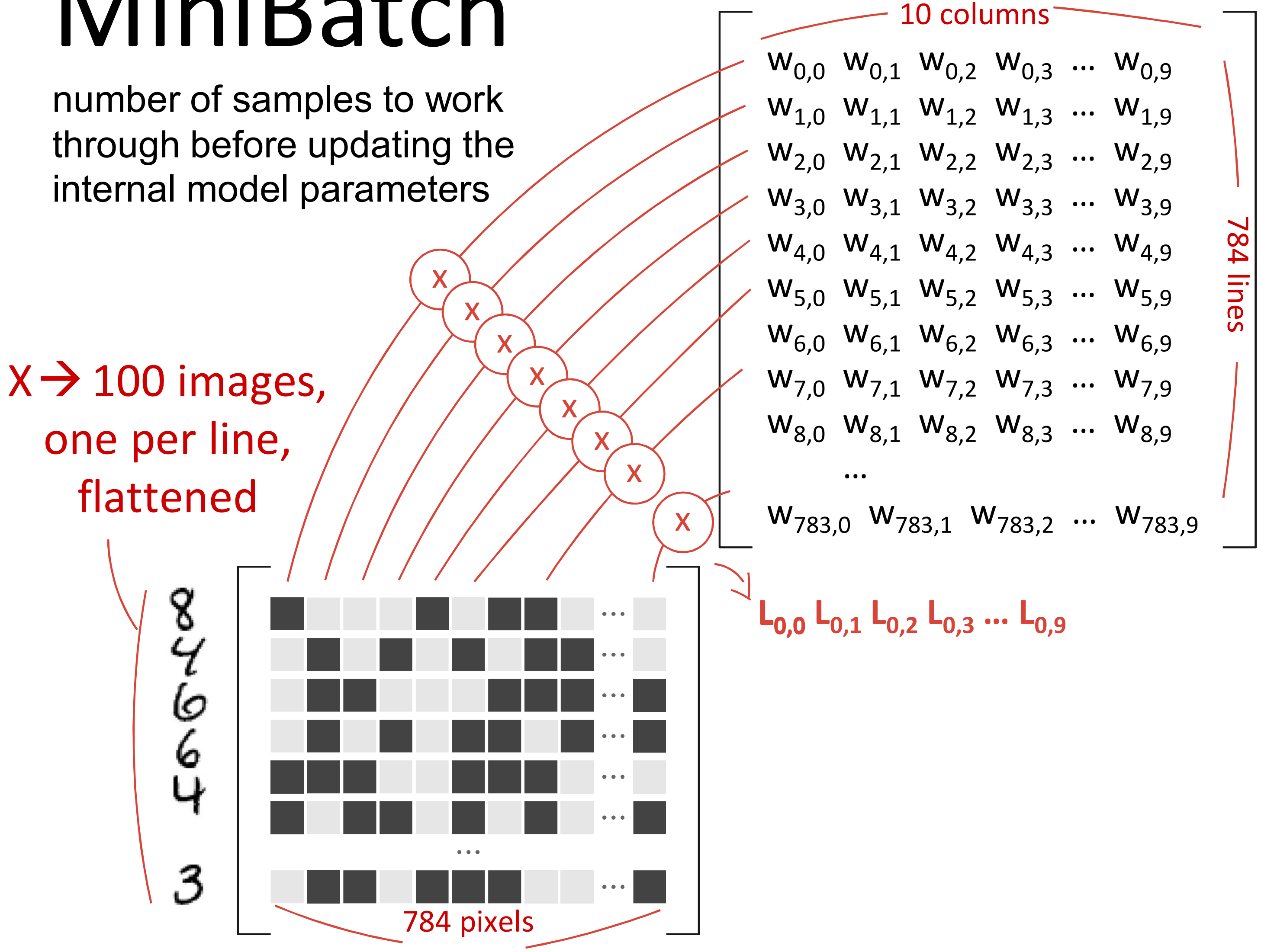
$$L = X.W + b$$

Martin Görner

[Learn TensorFlow and deep learning, without a Ph.D.](#)

MiniBatch

number of samples to work through before updating the internal model parameters



$$L = X.W + b$$

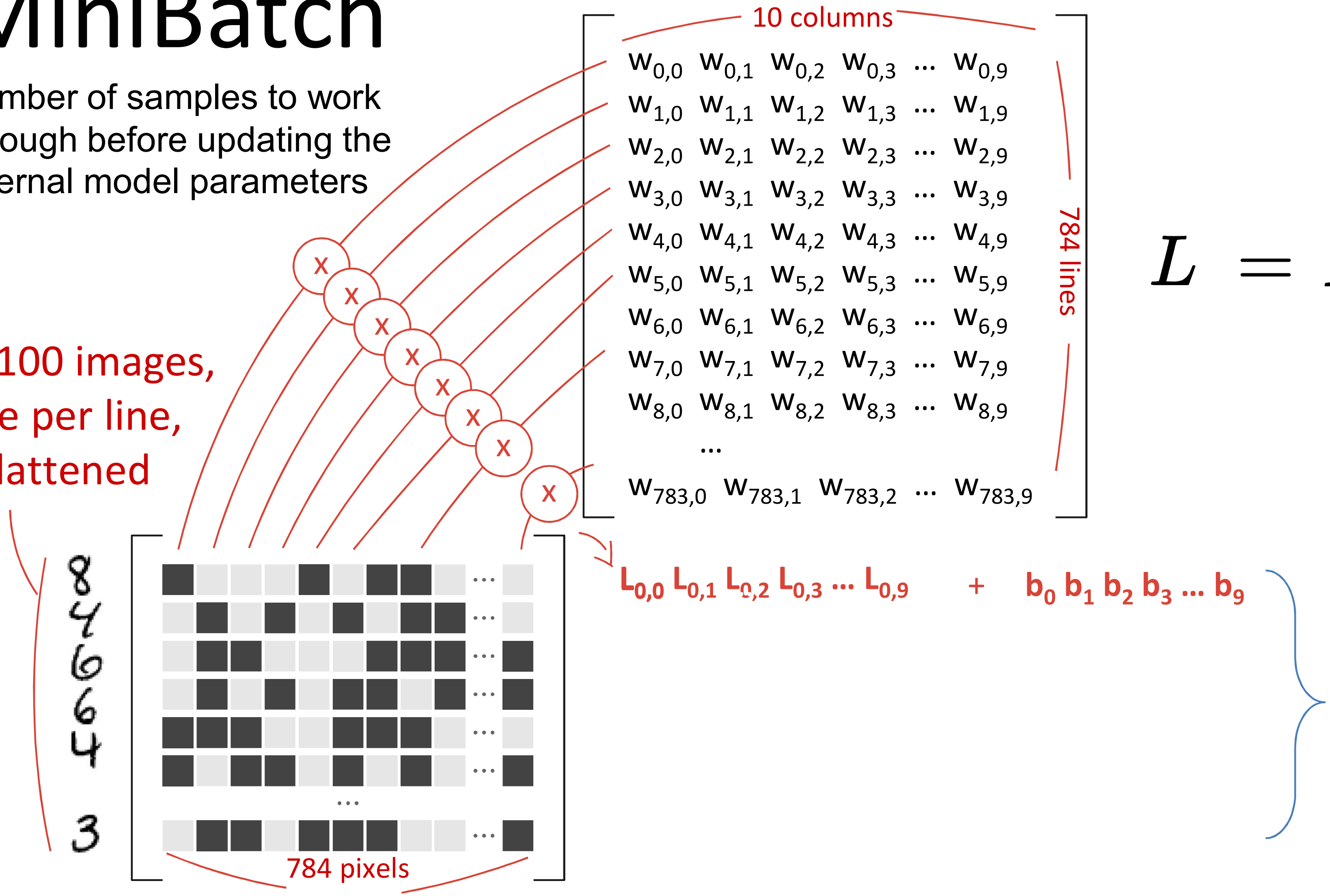
Martin Görner

[Learn TensorFlow and deep learning, without a Ph.D.]

MiniBatch

number of samples to work through before updating the internal model parameters

X → 100 images, one per line, flattened



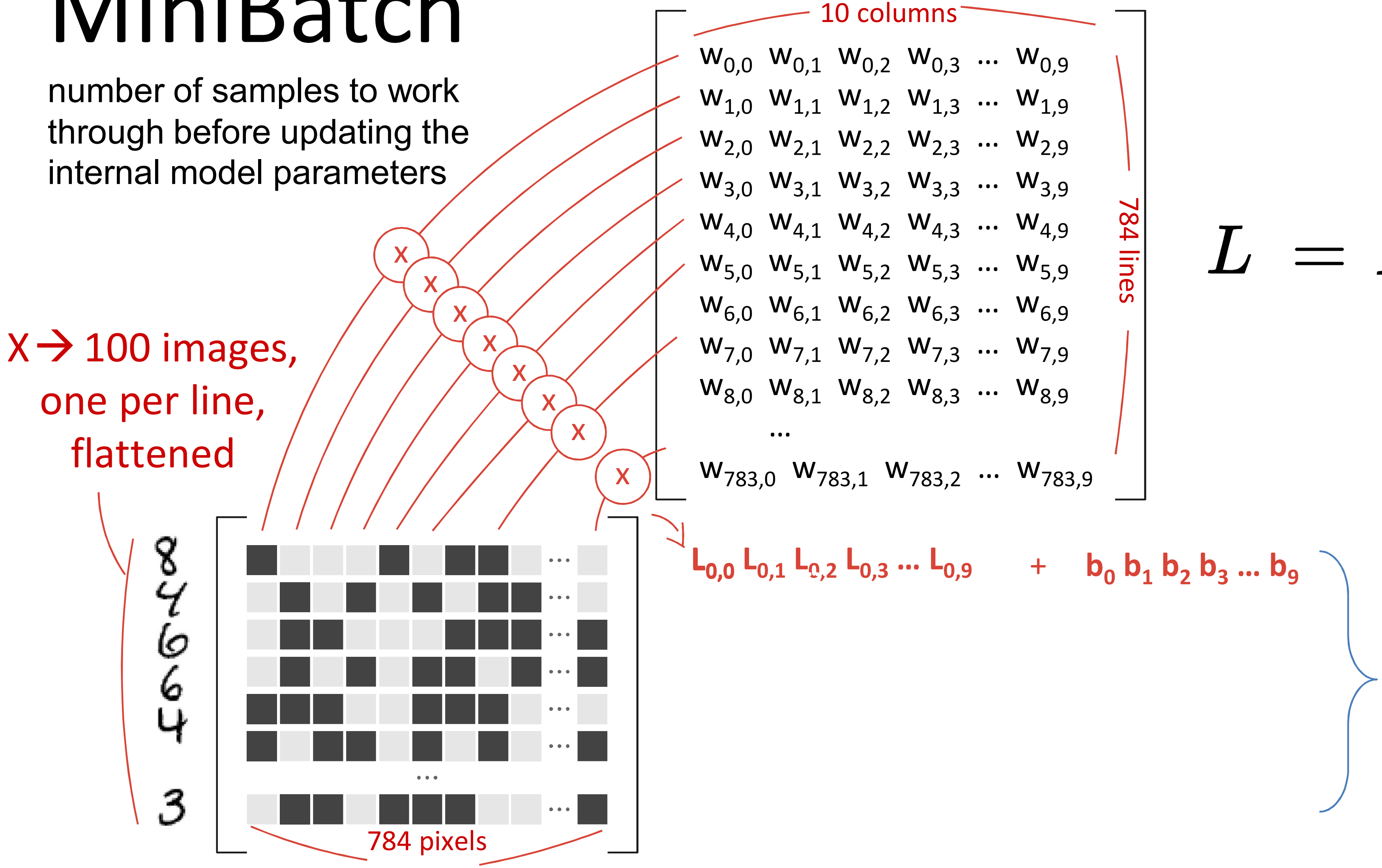
$$L = X.W + b$$

Martin Görner

[Learn TensorFlow and deep learning, without a Ph.D.]

MiniBatch

number of samples to work through before updating the internal model parameters



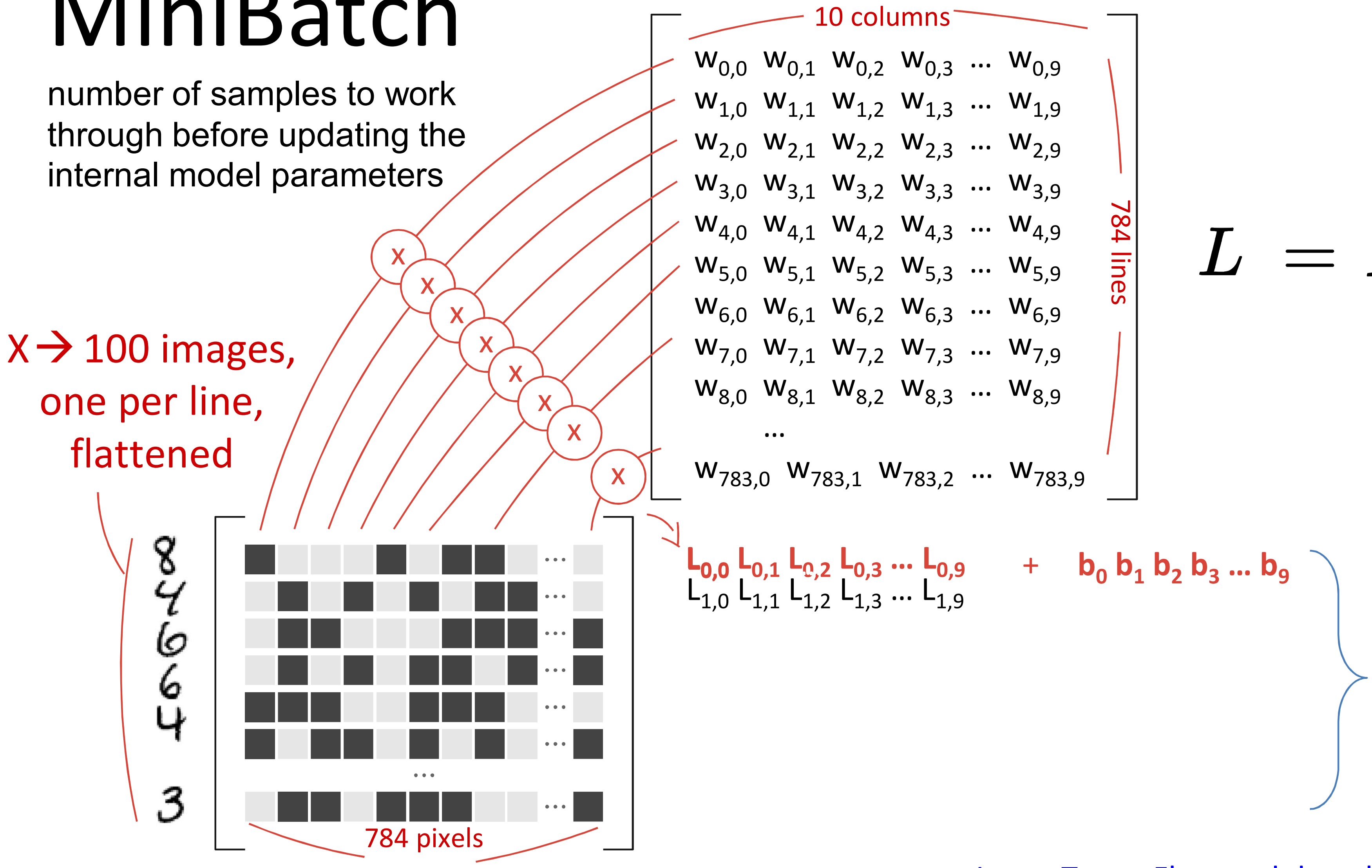
$$L = X.W + b$$

Martin Görner

[Learn TensorFlow and deep learning, without a Ph.D.]

MiniBatch

number of samples to work through before updating the internal model parameters



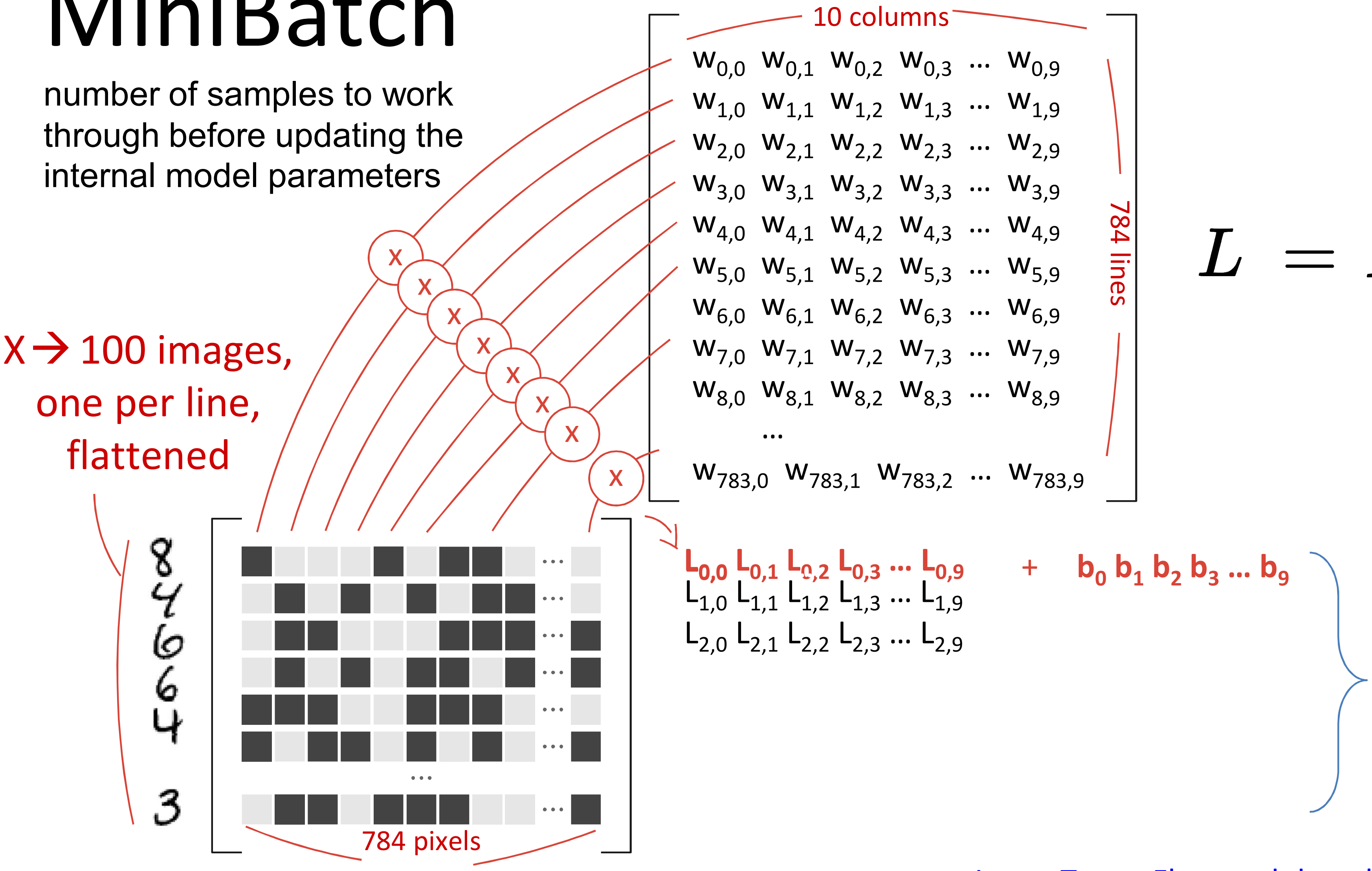
$$L = X.W + b$$

Martin Görner

[Learn TensorFlow and deep learning, without a Ph.D.]

MiniBatch

number of samples to work through before updating the internal model parameters



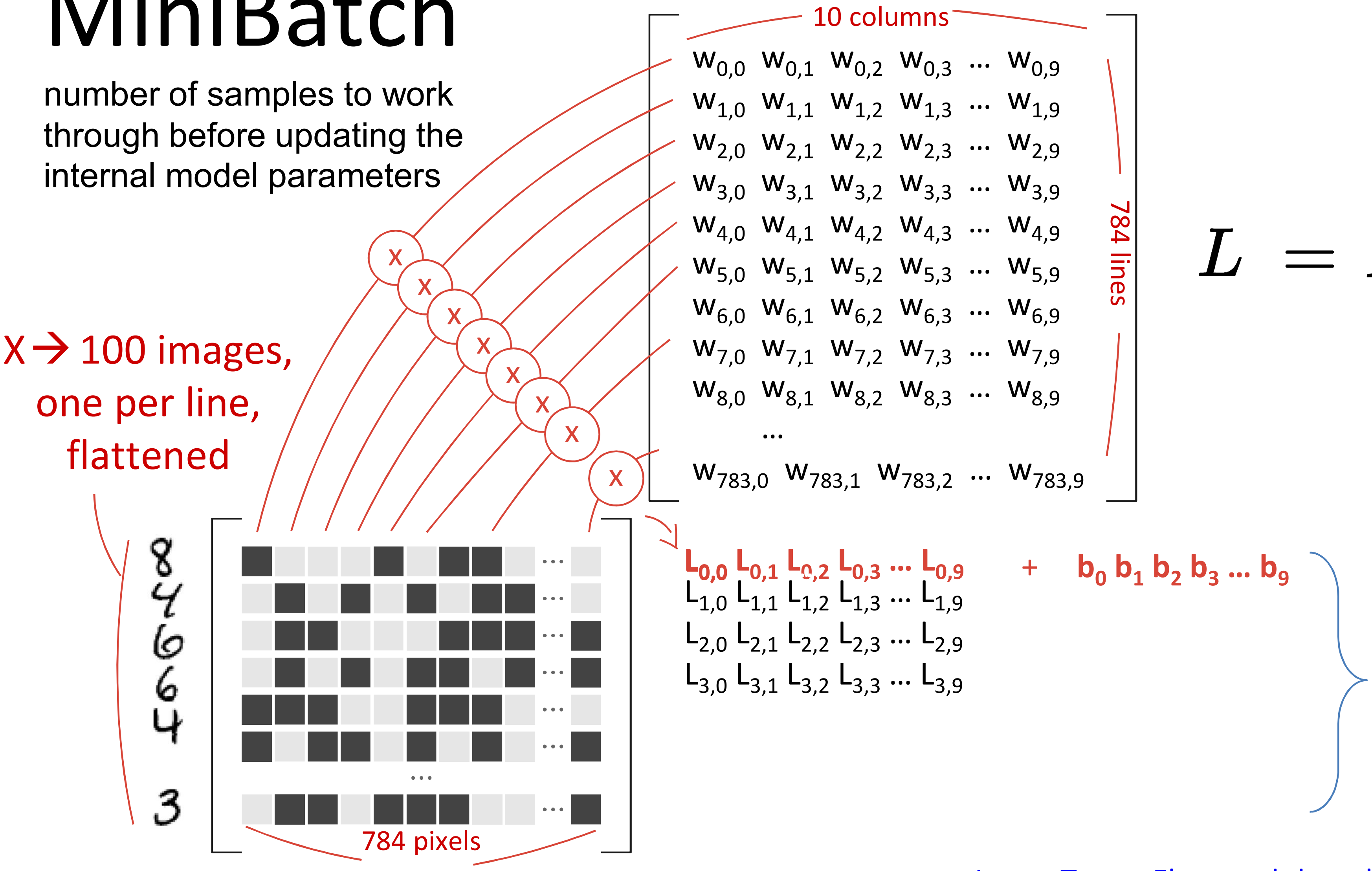
$$L = X.W + b$$

Martin Görner

[Learn TensorFlow and deep learning, without a Ph.D.]

MiniBatch

number of samples to work through before updating the internal model parameters



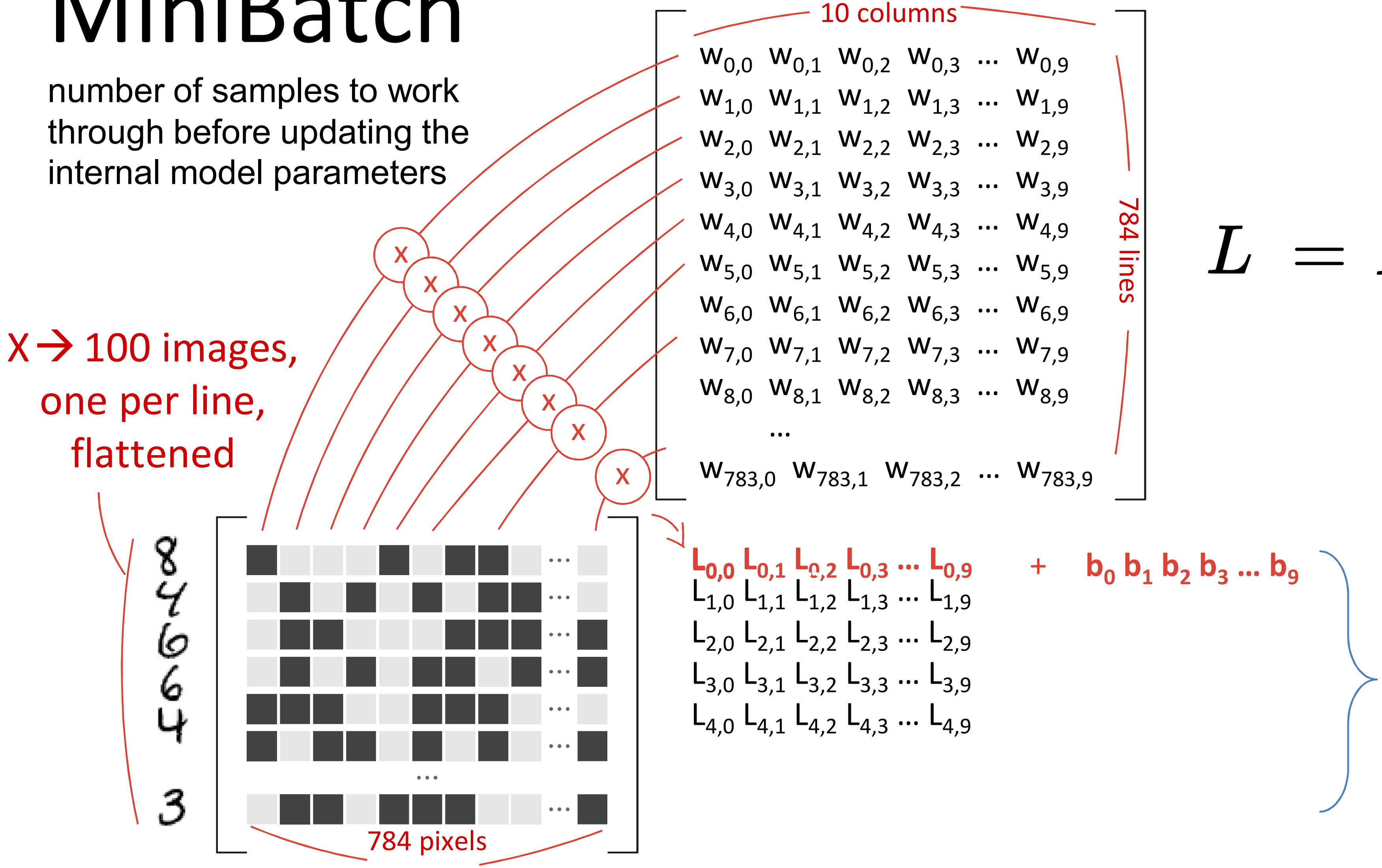
$$L = X.W + b$$

Martin Görner

[Learn TensorFlow and deep learning, without a Ph.D.]

MiniBatch

number of samples to work through before updating the internal model parameters



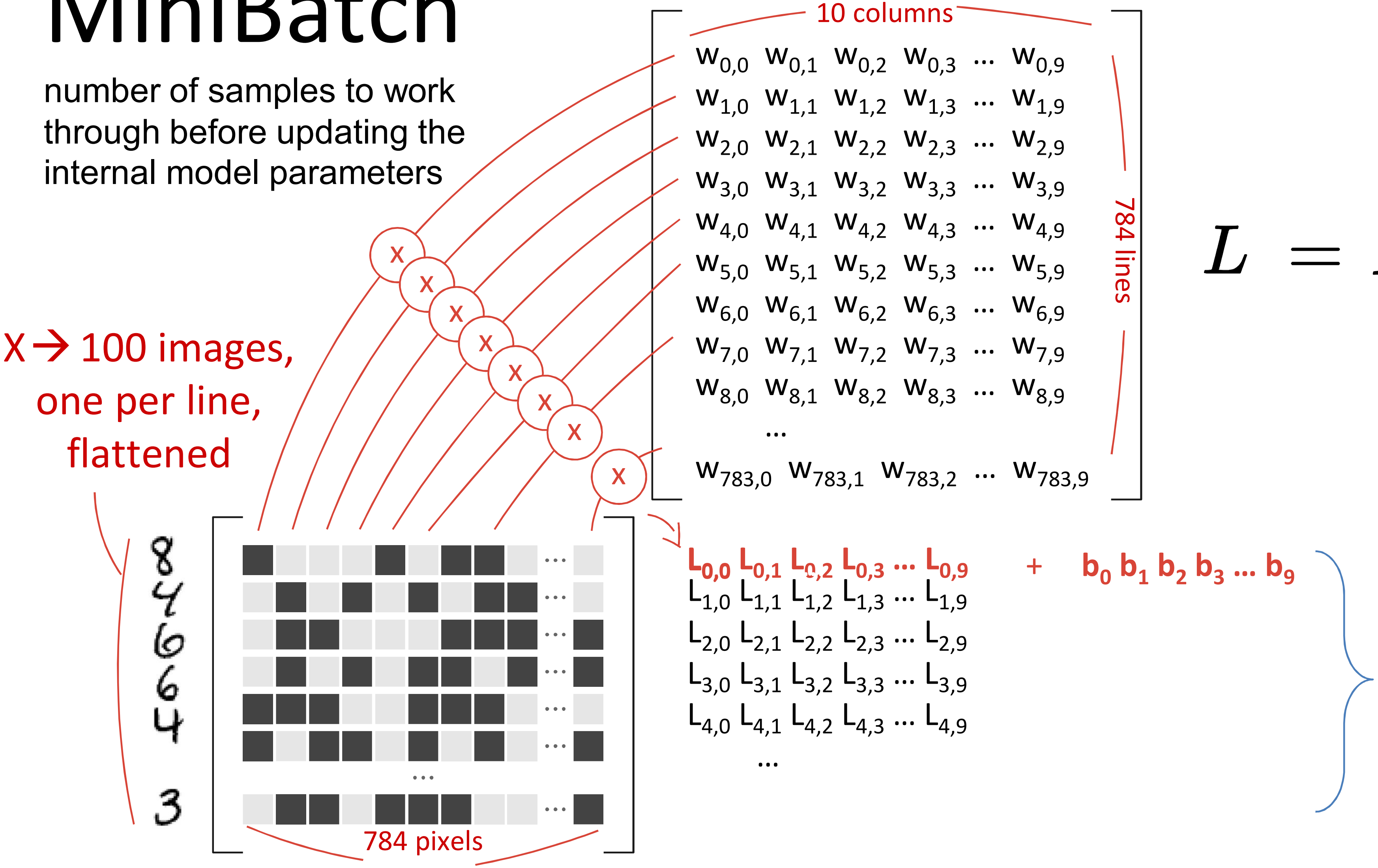
$$L = X.W + b$$

Martin Görner

[Learn TensorFlow and deep learning, without a Ph.D.]

MiniBatch

number of samples to work through before updating the internal model parameters



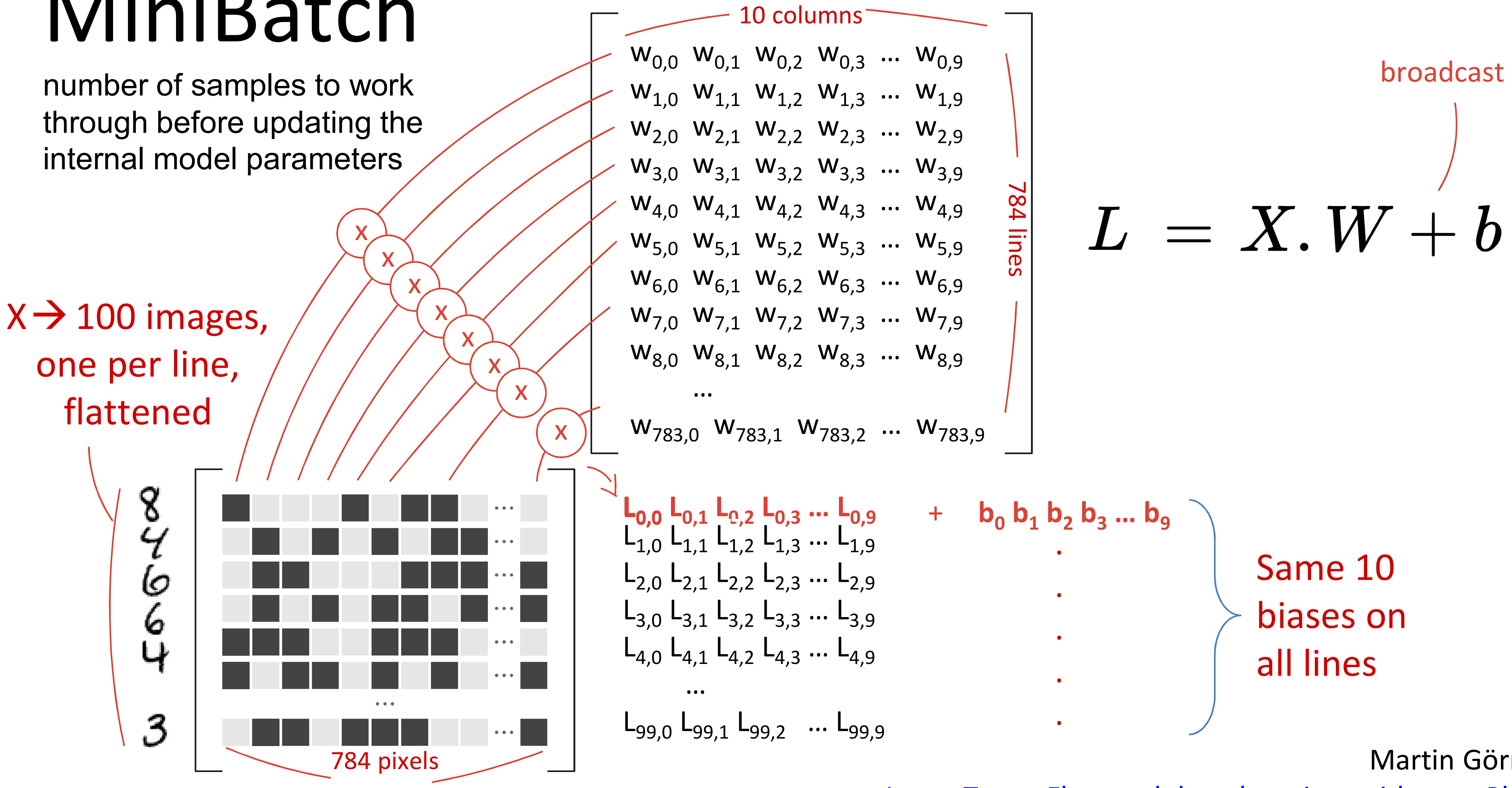
$$L = X.W + b$$

Martin Görner

[Learn TensorFlow and deep learning, without a Ph.D.]

MiniBatch

number of samples to work through before updating the internal model parameters



$$L = X.W + b$$

Martin Görner

[Learn TensorFlow and deep learning, without a Ph.D.](#)

Softmax on a batch of images

Predictions

Y[100, 10]

Images

X[100, 784]

Weights

W[784,10]

Biases

b[10]

$$Y = softmax(X.W + b)$$

applied line by line

matrix multiply

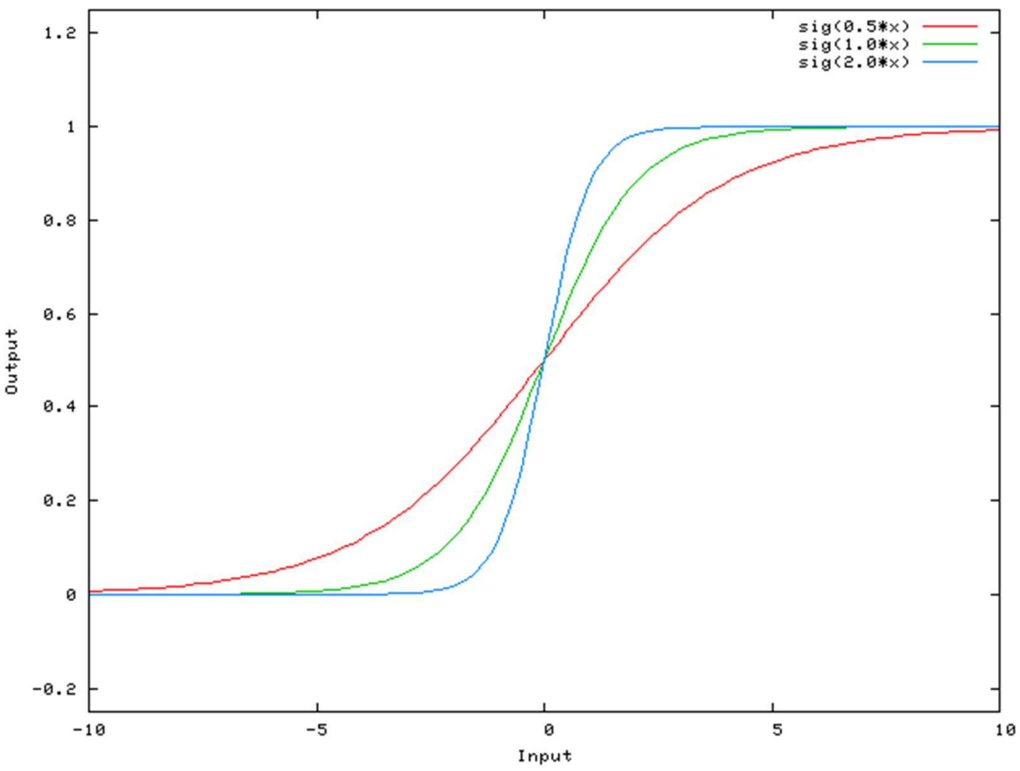
broadcast on all lines

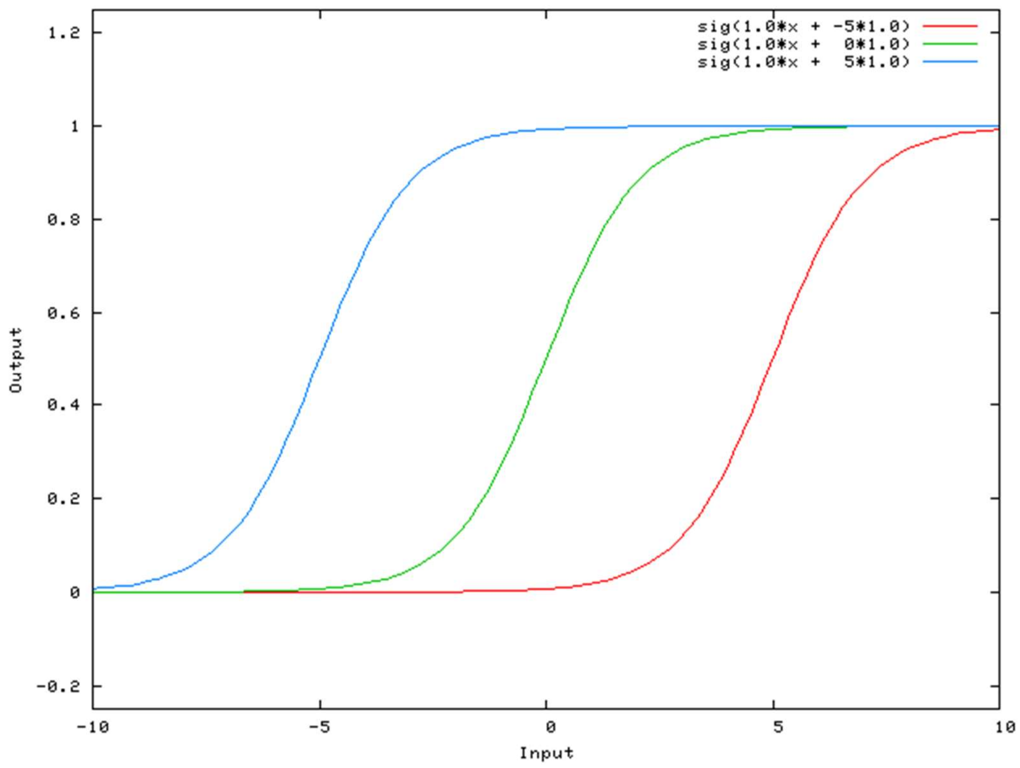
tensor shapes in [ ]

Martin Görner

Learn TensorFlow and deep learning, without a Ph.D.

Role of Bias


sig(0.5*x)
sig(1.0*x)
sig(2.0*x)
Output
Input
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
-10
-5
0
5
10


sig(1.0*x + -5*1.0)
sig(1.0*x + 0*1.0)
sig(1.0*x + 5*1.0)
Output
Input
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
-10
-5
0
5
10

Possible Softmax Output

Y computed probabilities

this is a “6”

0.1	0.2	0.1	0.3	0.2	0.1	**0.9**	0.2	0.1	0.1
0	1	2	3	4	5	6	7	8	9

Y' actual probabilities, “one-hot” encoded

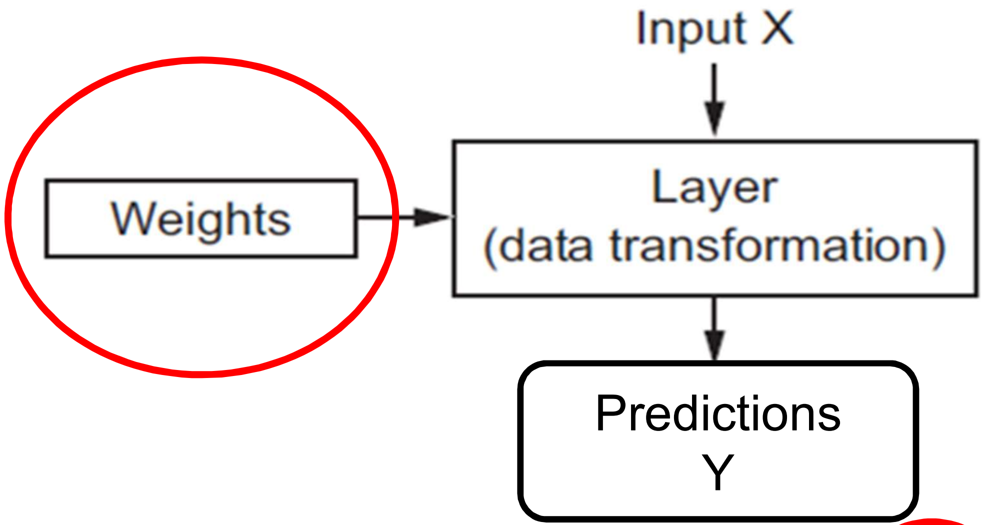
this is a “8”

0	0	0	0	0	0	0	0	**1**	0
0	1	2	3	4	5	6	7	8	9

We want a “8”
not a “6”

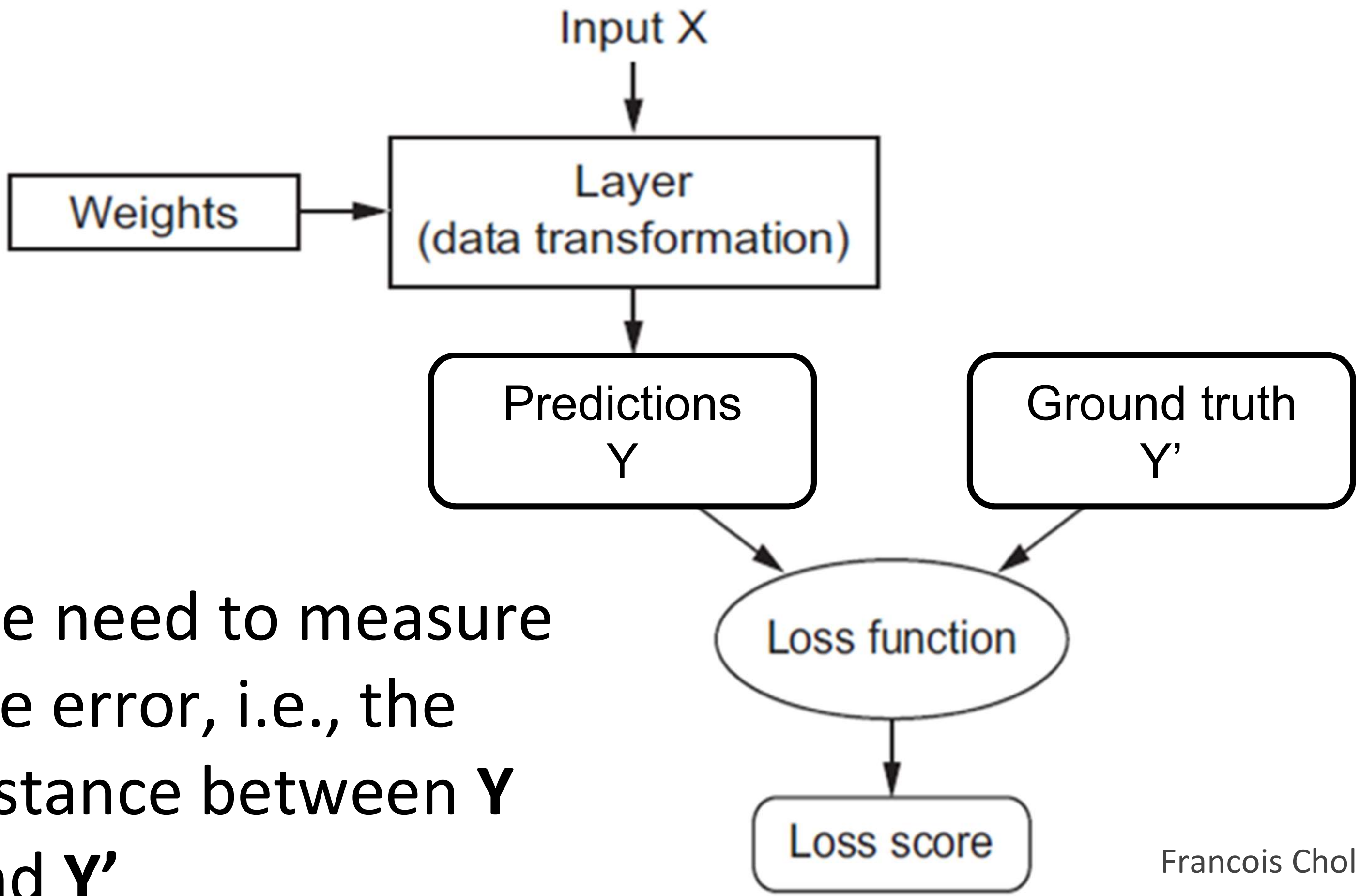
Finding the right weights (parameters)

Let's modify the weights **W** to change **Y** in order to represent a "8"



$$Y = softmax(X.W + b)$$

Loss Function



We need to measure the error, i.e., the distance between **Y** and **Y'**

Francois Chollet "Deep Learning with Python"
Manning Publications Co.

Cross Entropy as Loss function

computed probabilities

0.1	0.2	0.1	0.3	0.2	0.1	**0.9**	0.2	0.1	0.1
0	1	2	3	4	5	6	7	8	9

Cross entropy: $$-\sum Y'_i \, . \, log(Y_i)$$

actual probabilities, "one-hot" encoded

0	0	0	0	0	0	0	0	**1**	0
0	1	2	3	4	5	6	7	8	9

Martin Görner

[Learn TensorFlow and deep learning, without a Ph.D.]()

Learning the network parameters

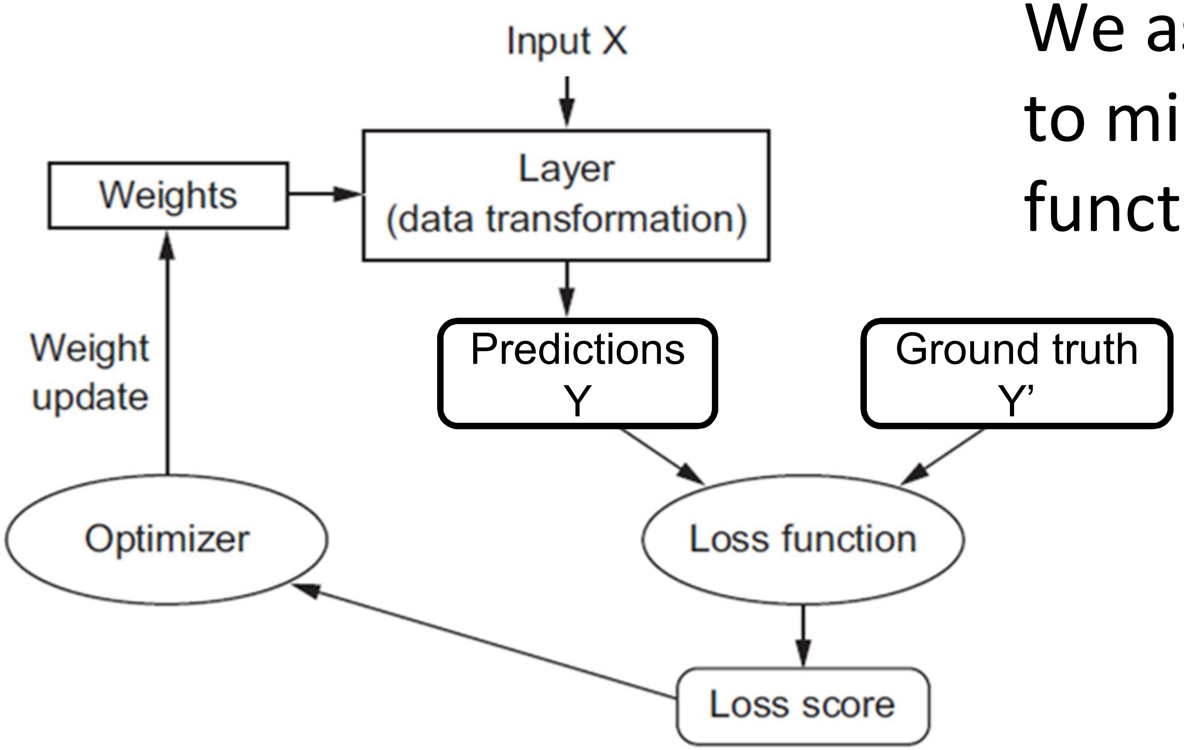
<u>Problem</u>

Given

- a labeled dataset **X** = {x1, x2, …} of inputs
- the associated outputs **Y**(x) = {y(x1), y(x2), …}

Find the weights *w* and biases *b* that minimize the **loss function** (i.e., the error)

Adjust the weights



We ask an optimizer to minimize our error function

Which is a good optimizer for our problem?

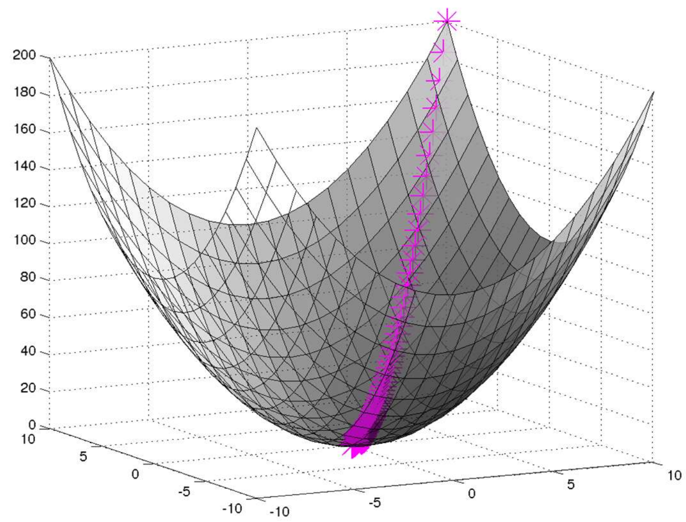
Francois Chollet "Deep Learning with Python"
Manning Publications Co.

Gradient descent optimizer

Easy answer! Gradient descent!

Correct but … very difficult implementation in practice, due to:

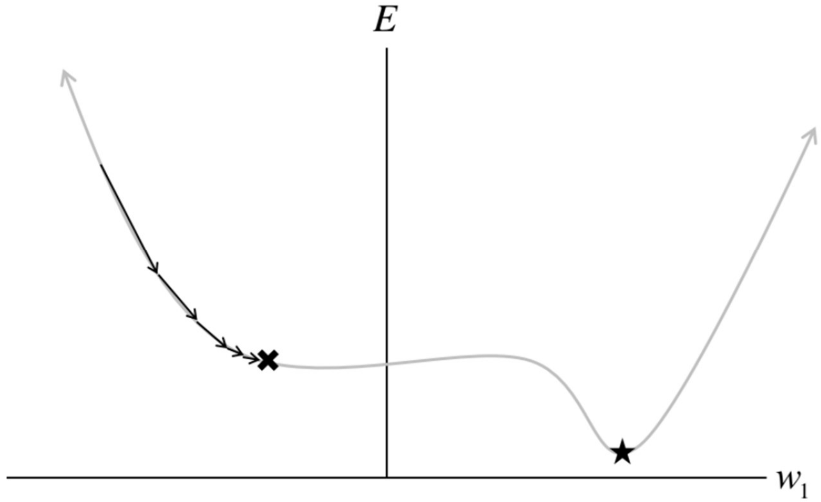
- Very large parameters set
- Very slow convergence rate
- Huge amount of data
- Weight saturation
- ….

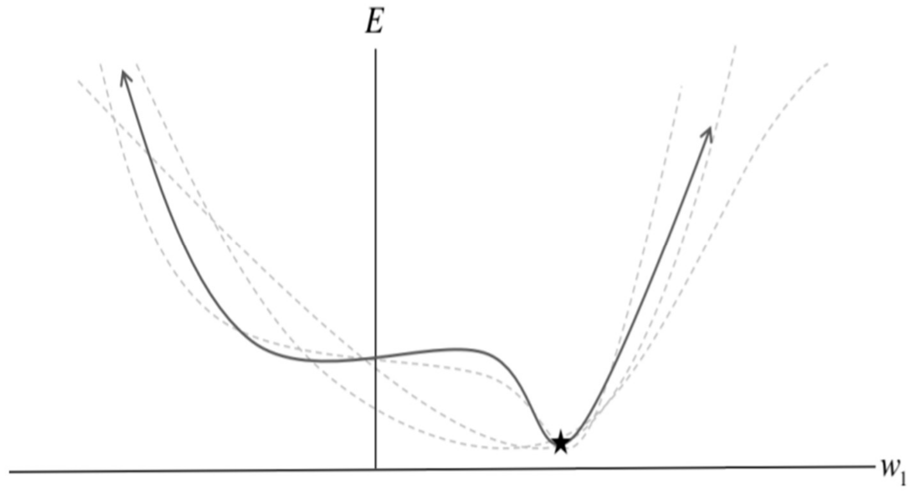


Stochastic Gradient Descent

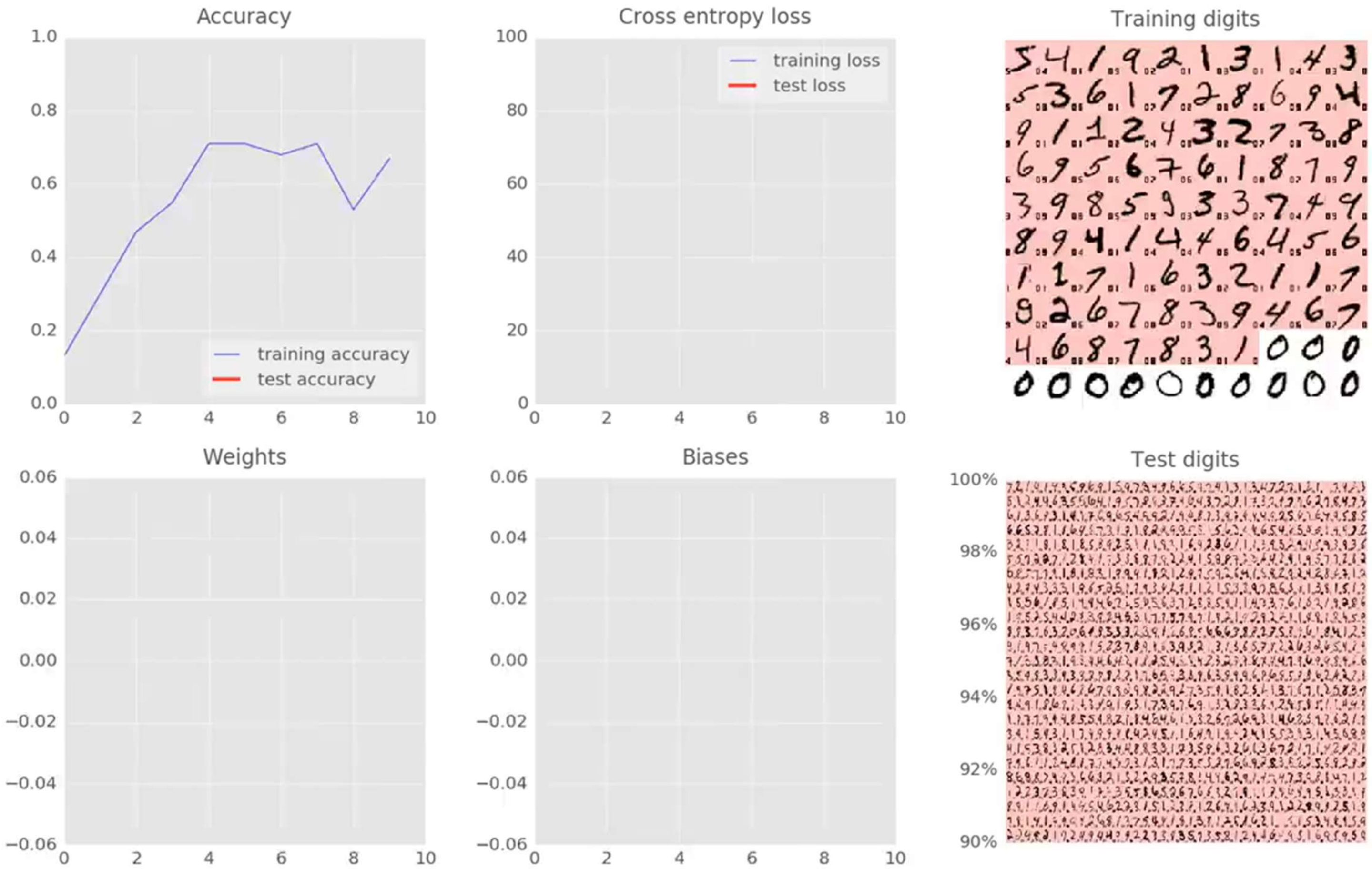
Batch gradient descent is sensitive to saddle points, which can lead to premature convergence

In stochastic gradient descent, at every iteration, we compute the error surface with respect to some subset (minibatch) of the total dataset





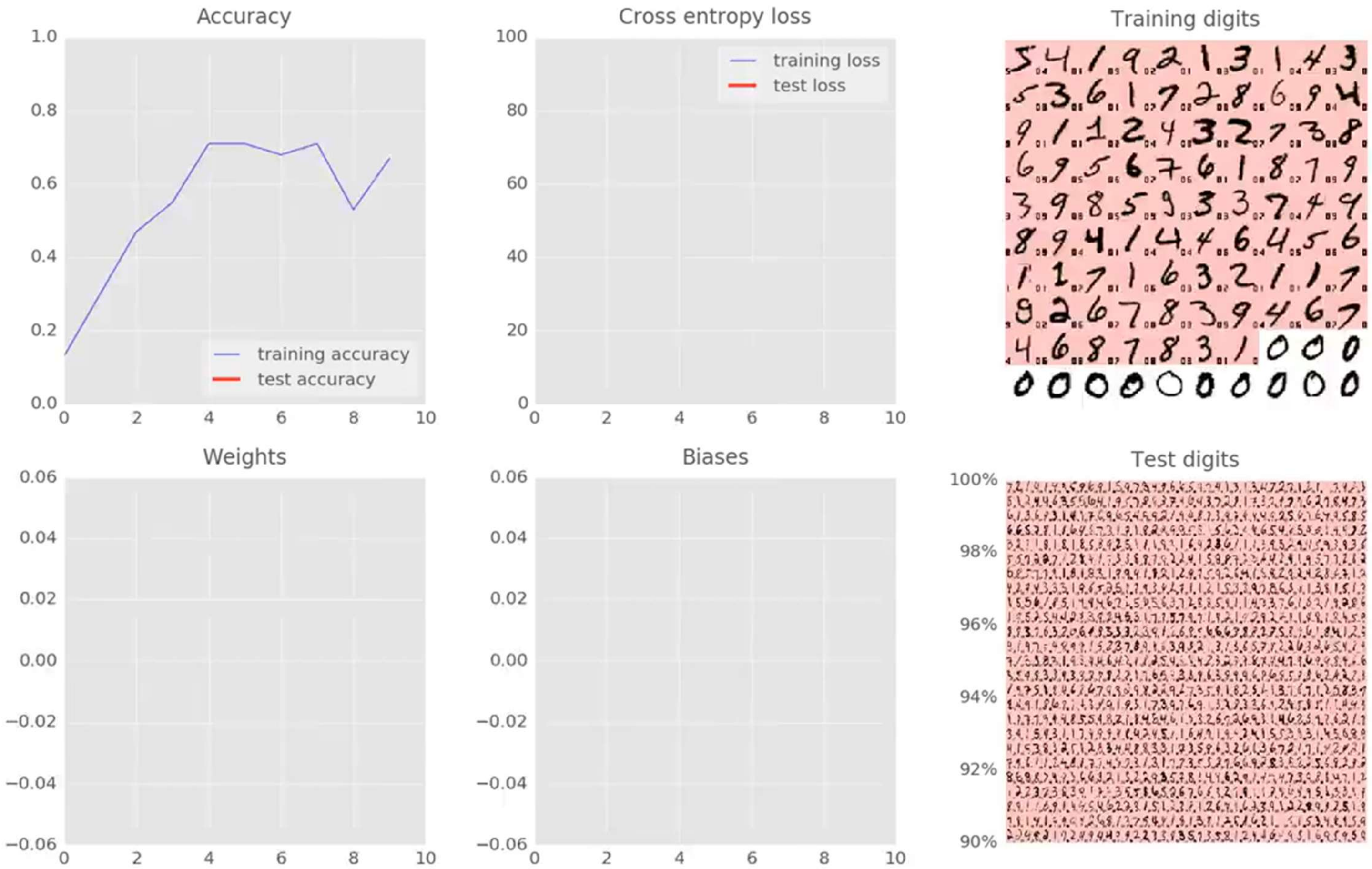
Training - Single layer network



Martin Görner
[Learn TensorFlow and deep learning, without a Ph.D.]()

Training - Single layer network

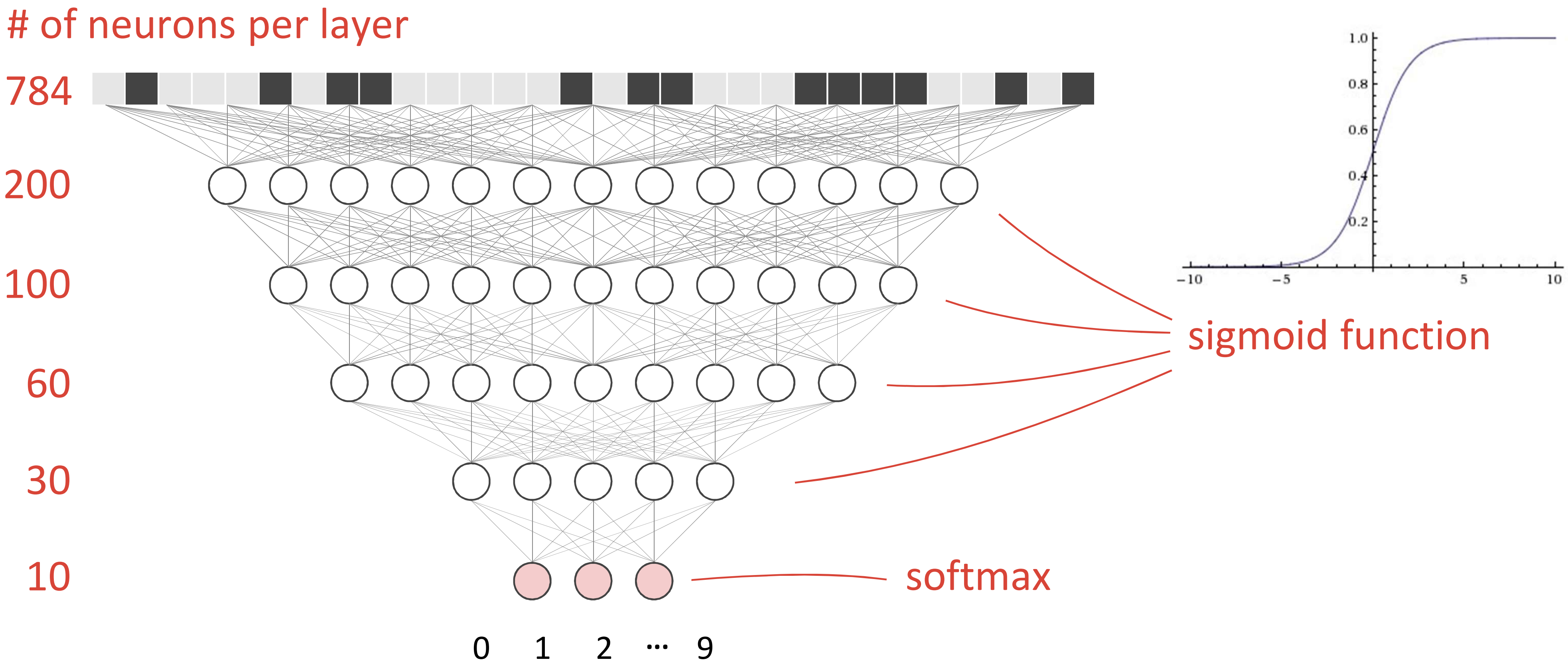
92%



Martin Görner

[Learn TensorFlow and deep learning, without a Ph.D.]

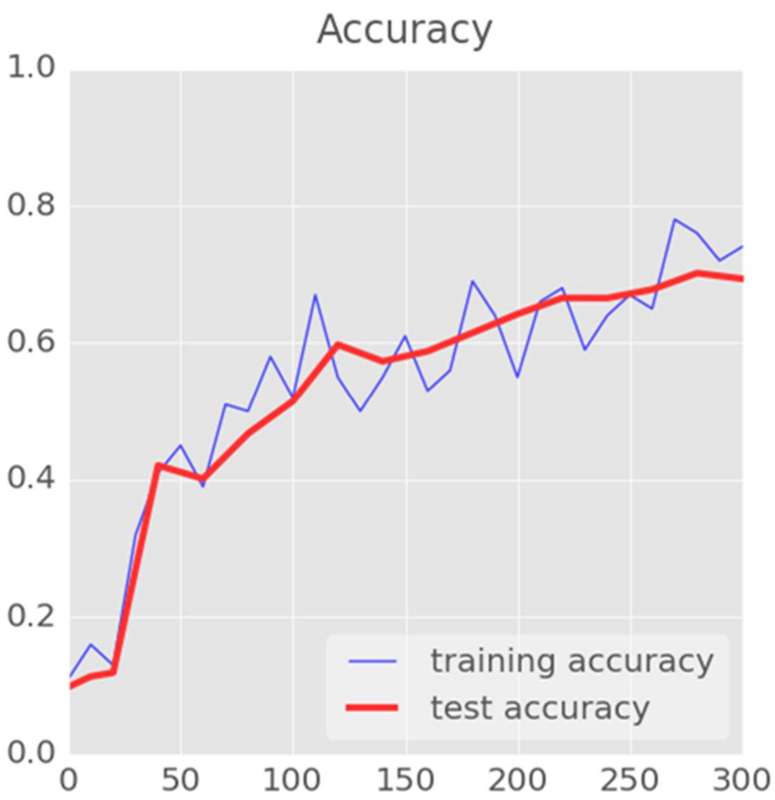
Let's go deep → from 1 to 5 layers

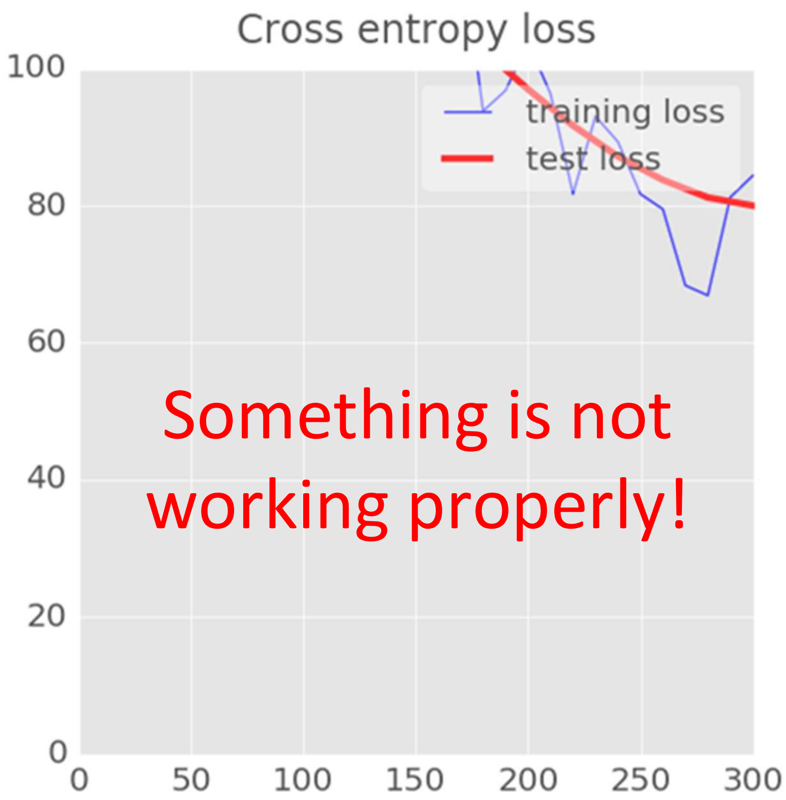


Martin Görner

[Learn TensorFlow and deep learning, without a Ph.D.]

Training - Five layer network (sigmoid)





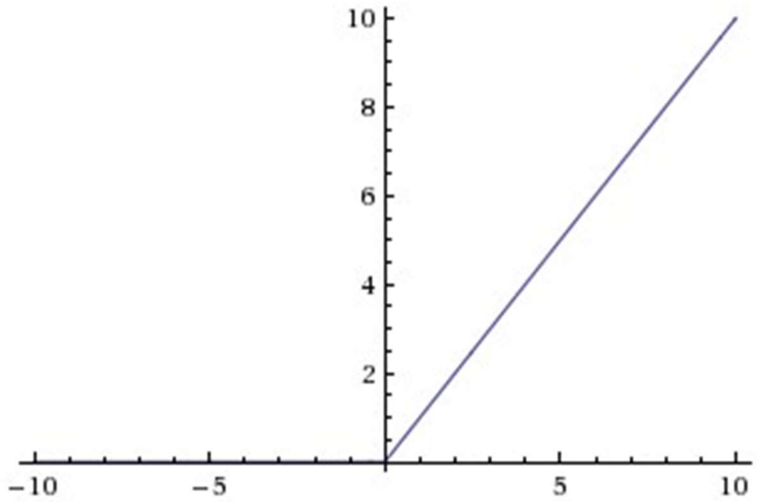
- 300 iterations
- learning rate 0.003

Martin Görner
[Learn TensorFlow and deep learning, without a Ph.D.]

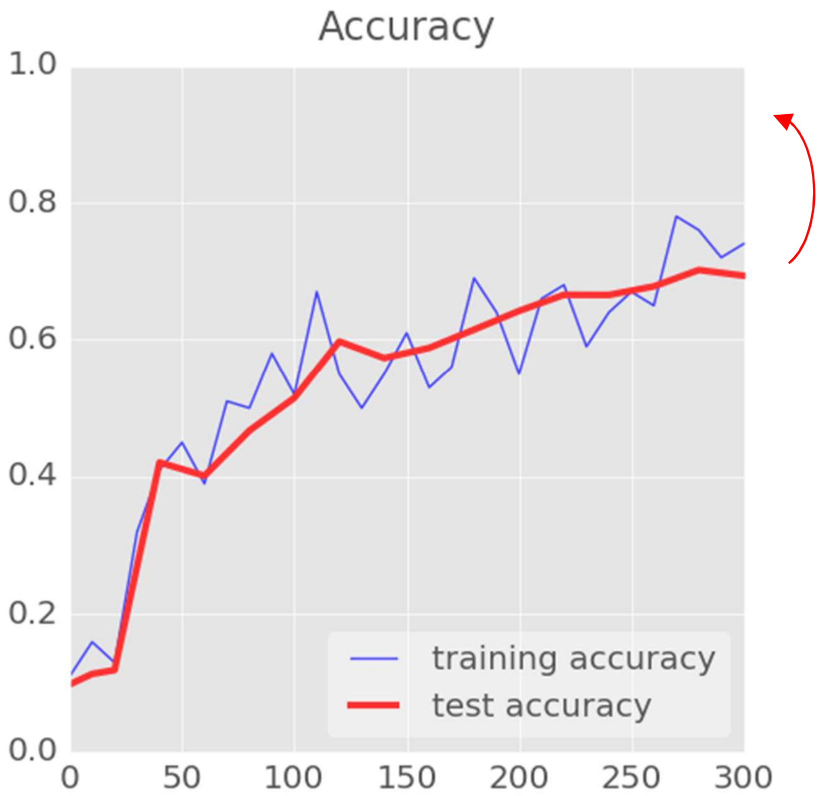
ReLU

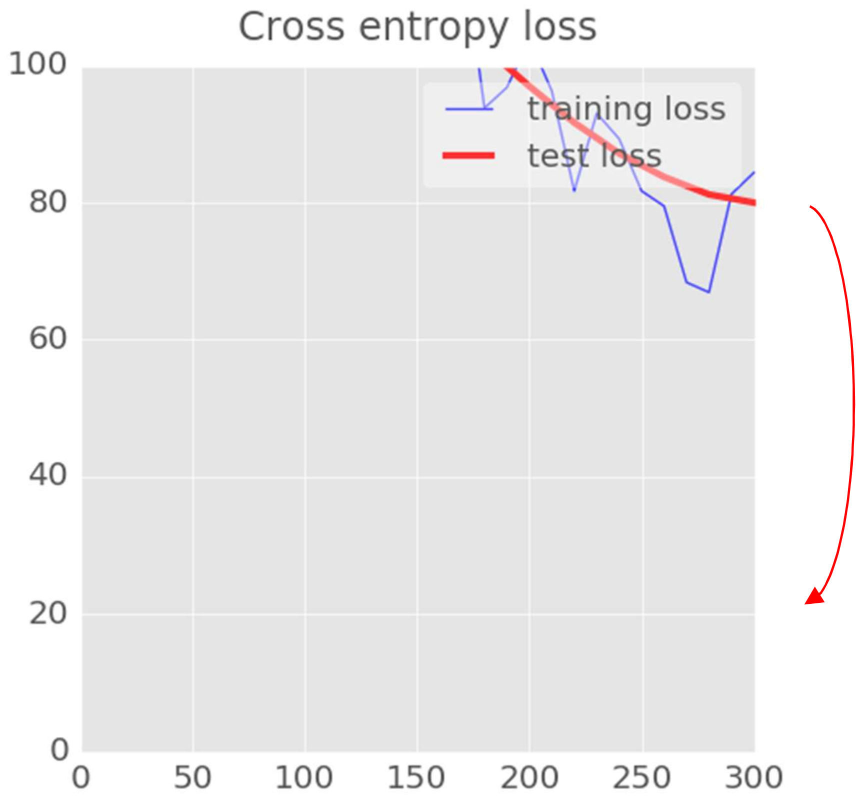
- ReLU stands for Rectified Linear Unit
- It takes a real-valued input and thresholds it at zero (replaces negative values with zero)

$$f(x) = \max(0, x)$$



Training with ReLU



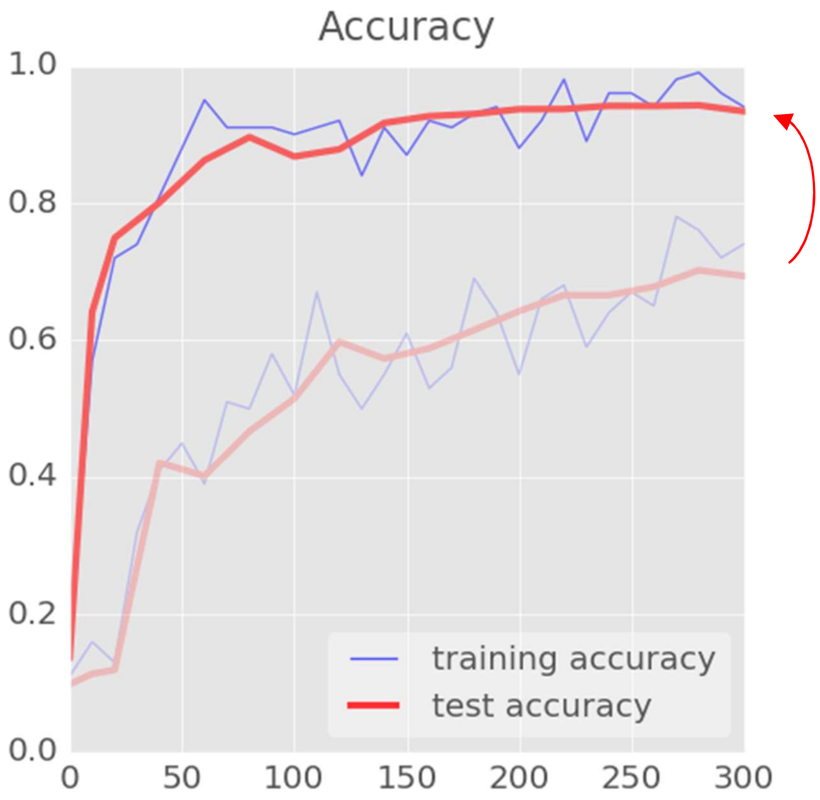


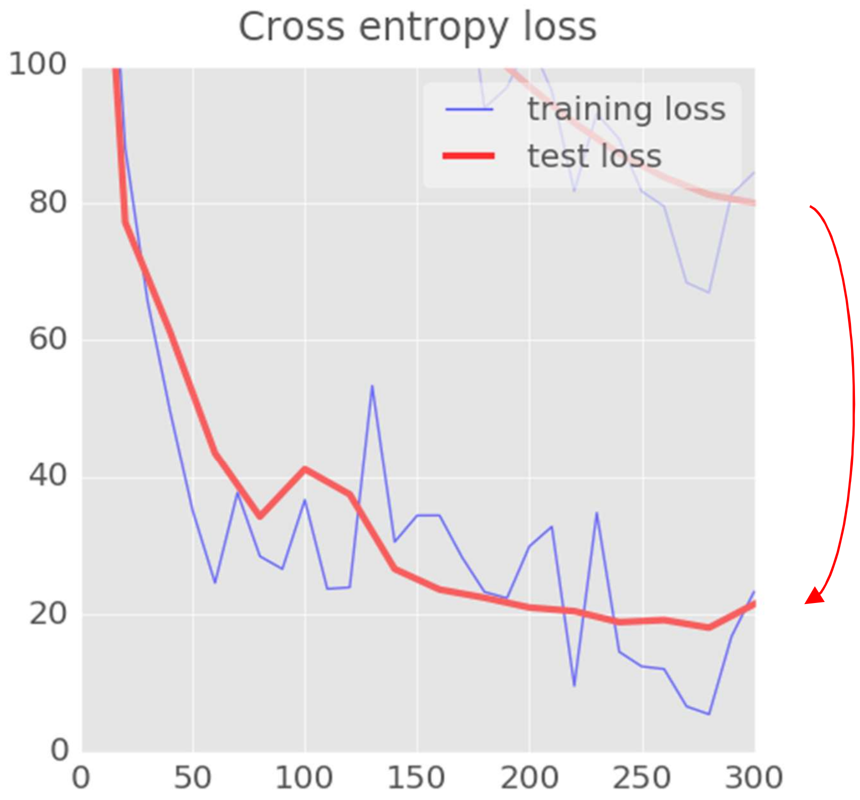
- 300 iterations
- learning rate 0.003

Martin Görner

[Learn TensorFlow and deep learning, without a Ph.D.]

Training with ReLU



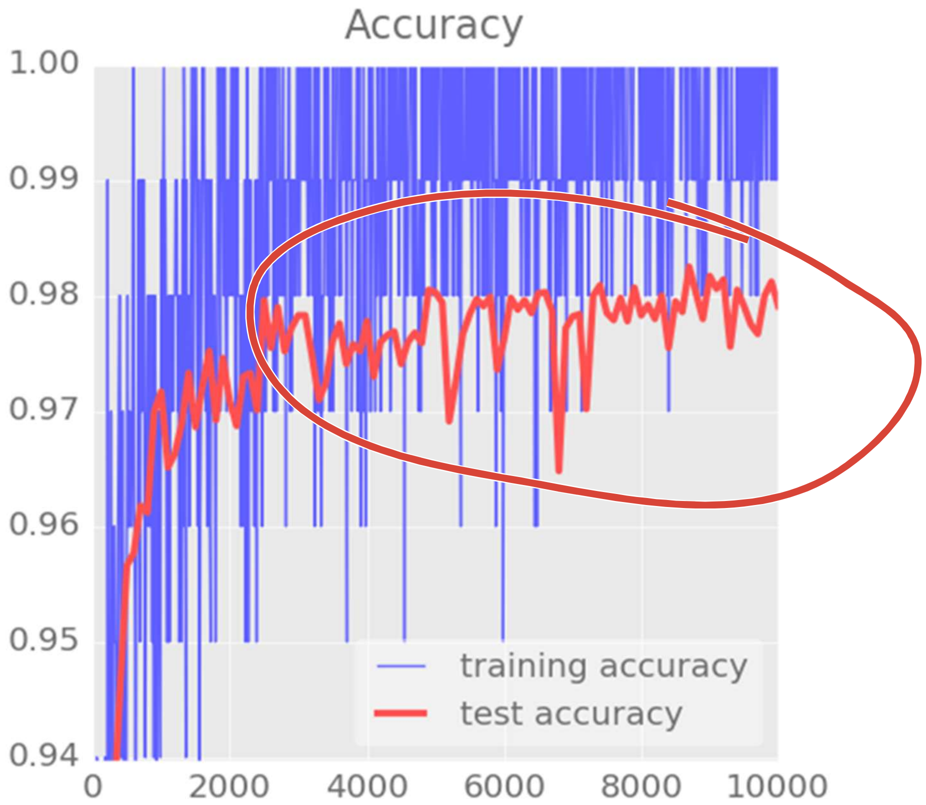


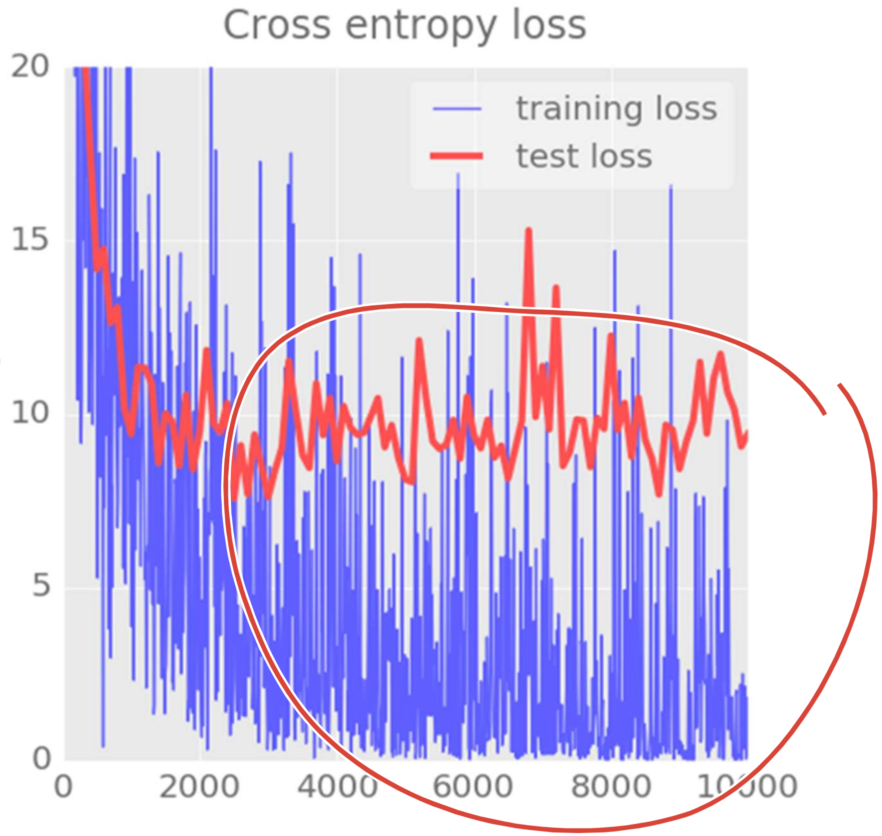
- 300 iterations
- learning rate 0.003

Martin Görner
[Learn TensorFlow and deep learning, without a Ph.D.](#)

Training with ReLU

98%





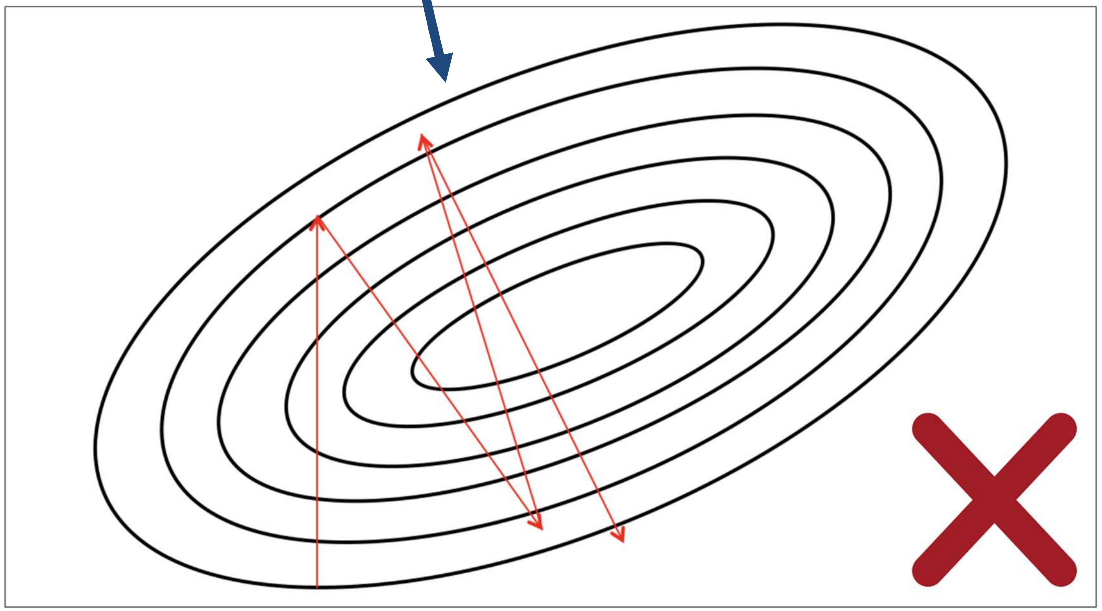
- 10000 iterations
- learning rate 0.003

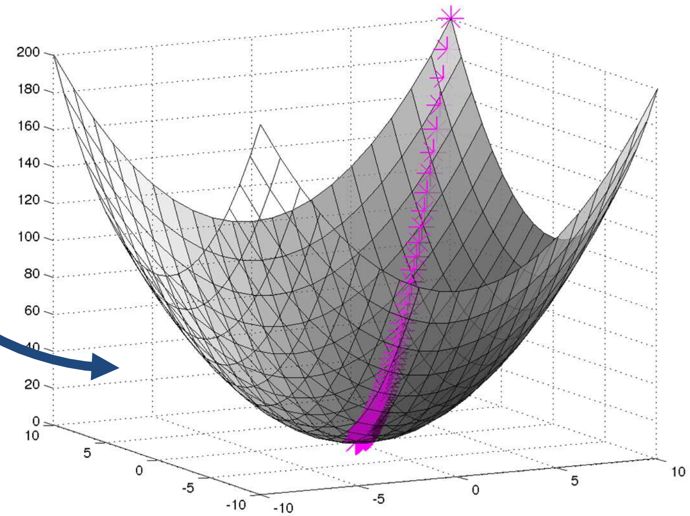
Martin Görner

[Learn TensorFlow and deep learning, without a Ph.D.](#)

We are going too fast!

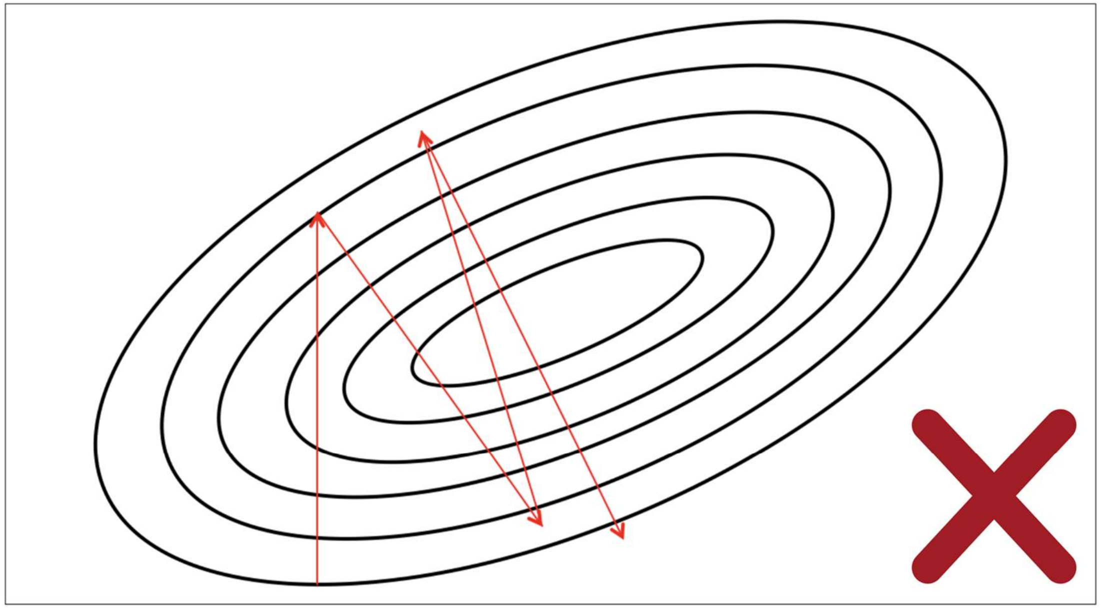
- Our training curves are too noisy
- We are jumping from one side of the valley to the other without reaching the bottom of our error function

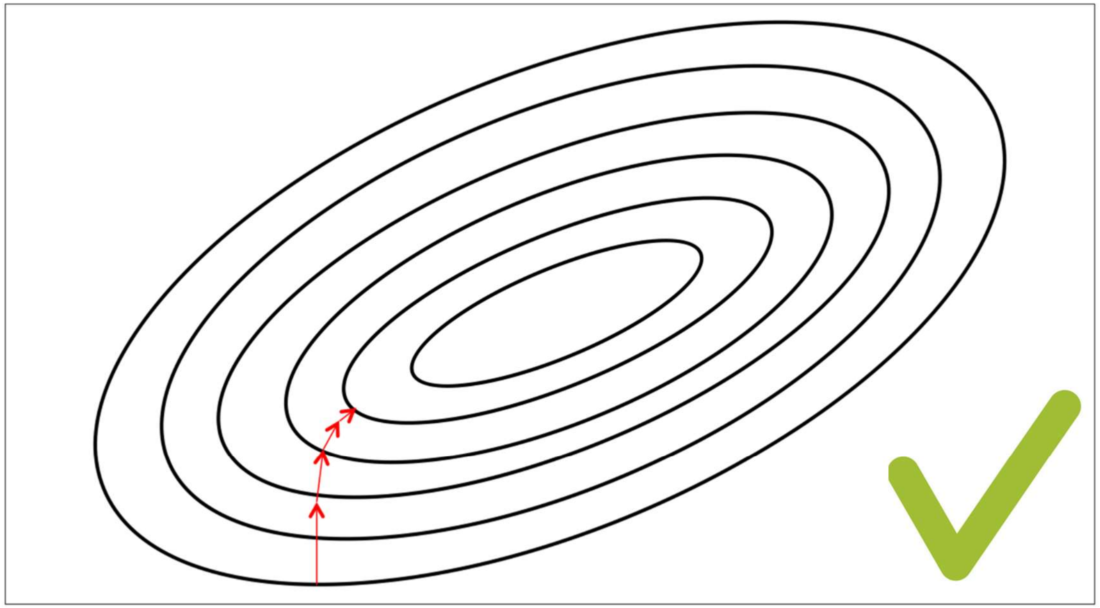




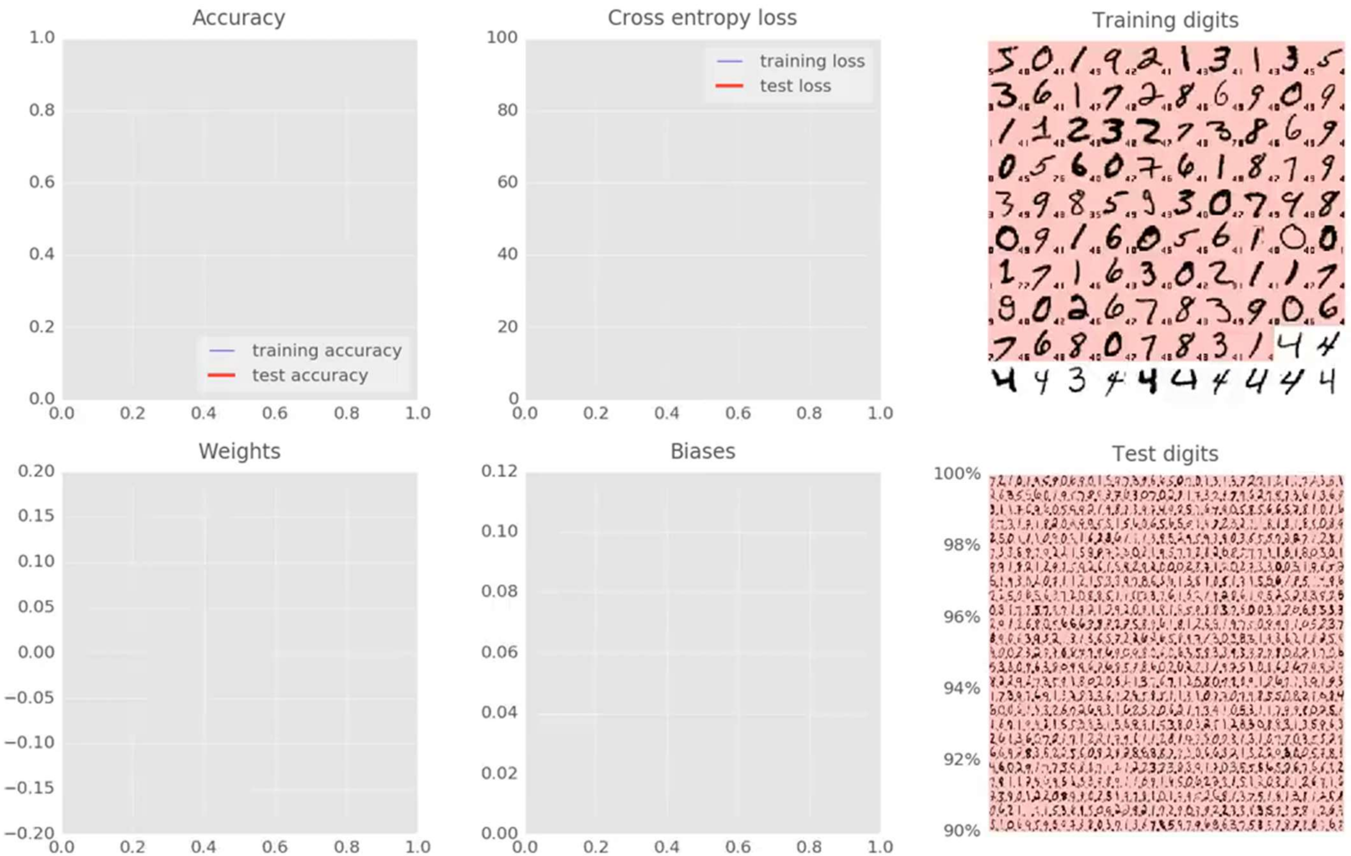
Solution: Adaptive Learning rate

- We start fast and than slow down
- The closer we are to the minimum, the shorter we want to step forward





Training with Adaptive Learning rate

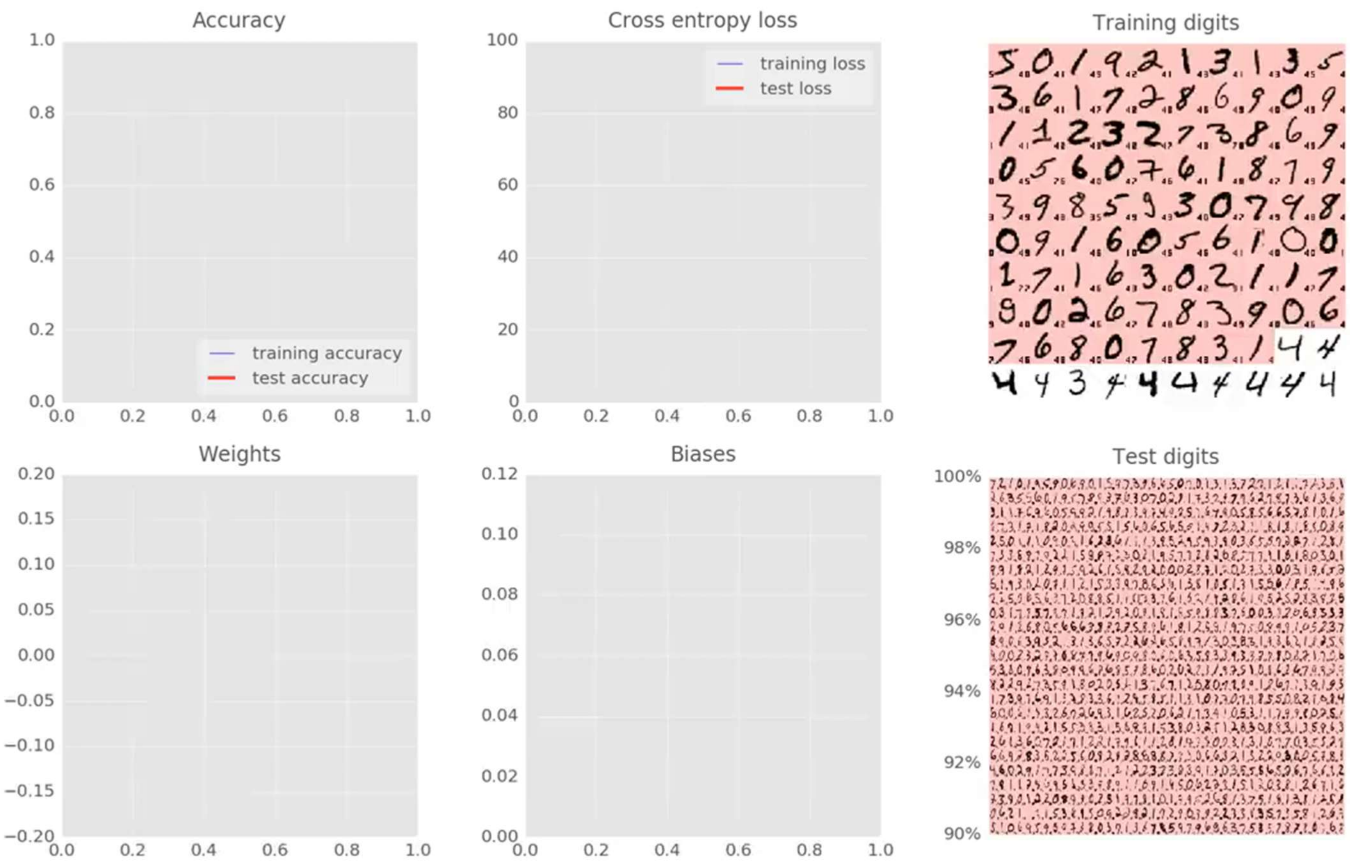


Martin Görner

[Learn TensorFlow and deep learning, without a Ph.D.]

Training with Adaptive Learning rate

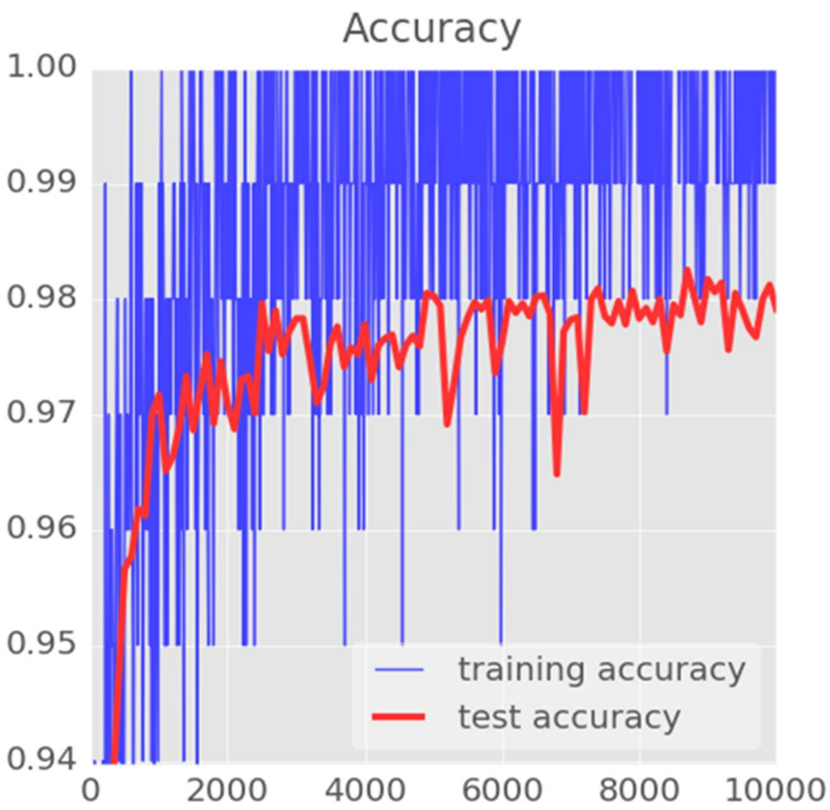
98%

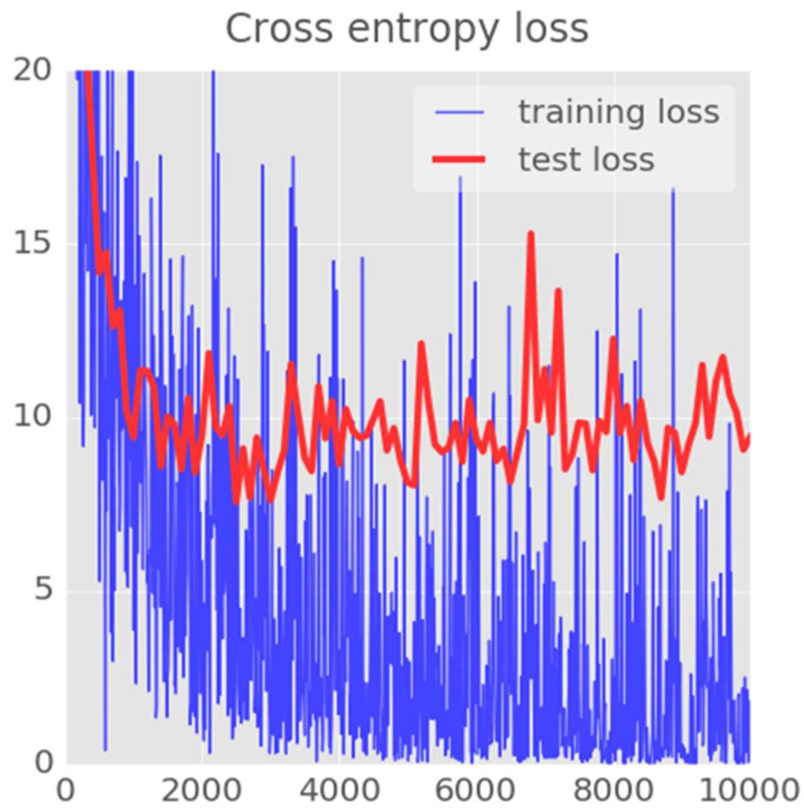


Martin Görner

[Learn TensorFlow and deep learning, without a Ph.D.]

Learning rate decay



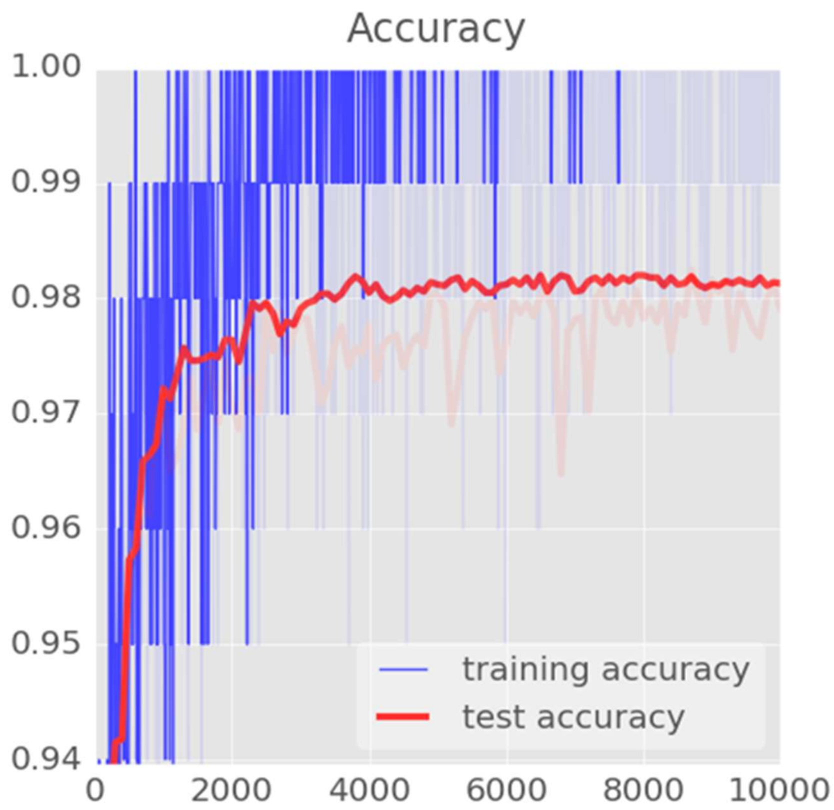


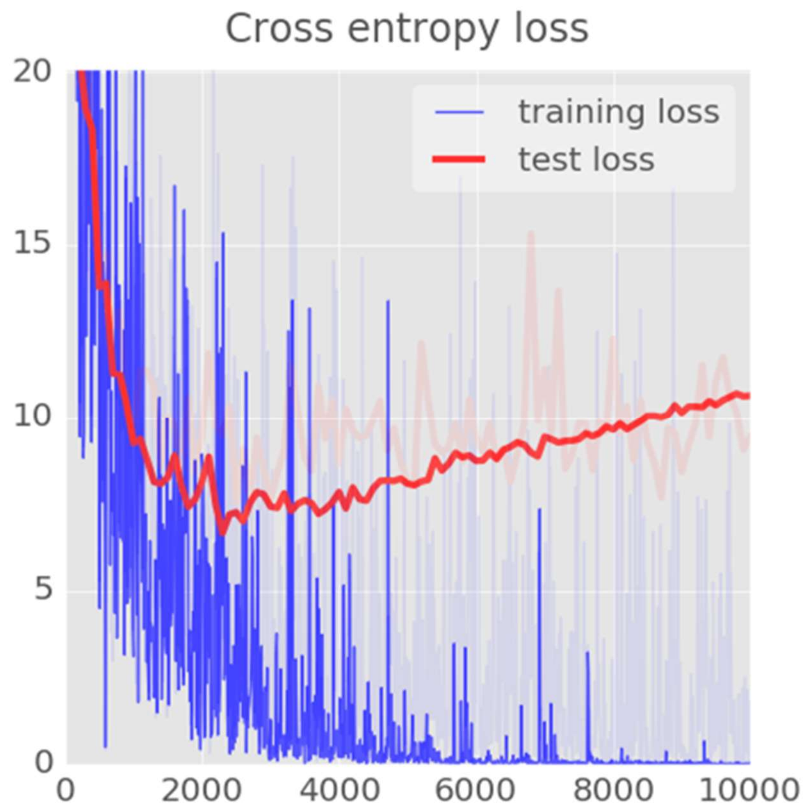
- 10000 iterations
- learning rate 0.003

Martin Görner

[Learn TensorFlow and deep learning, without a Ph.D.](#)

Learning rate decay

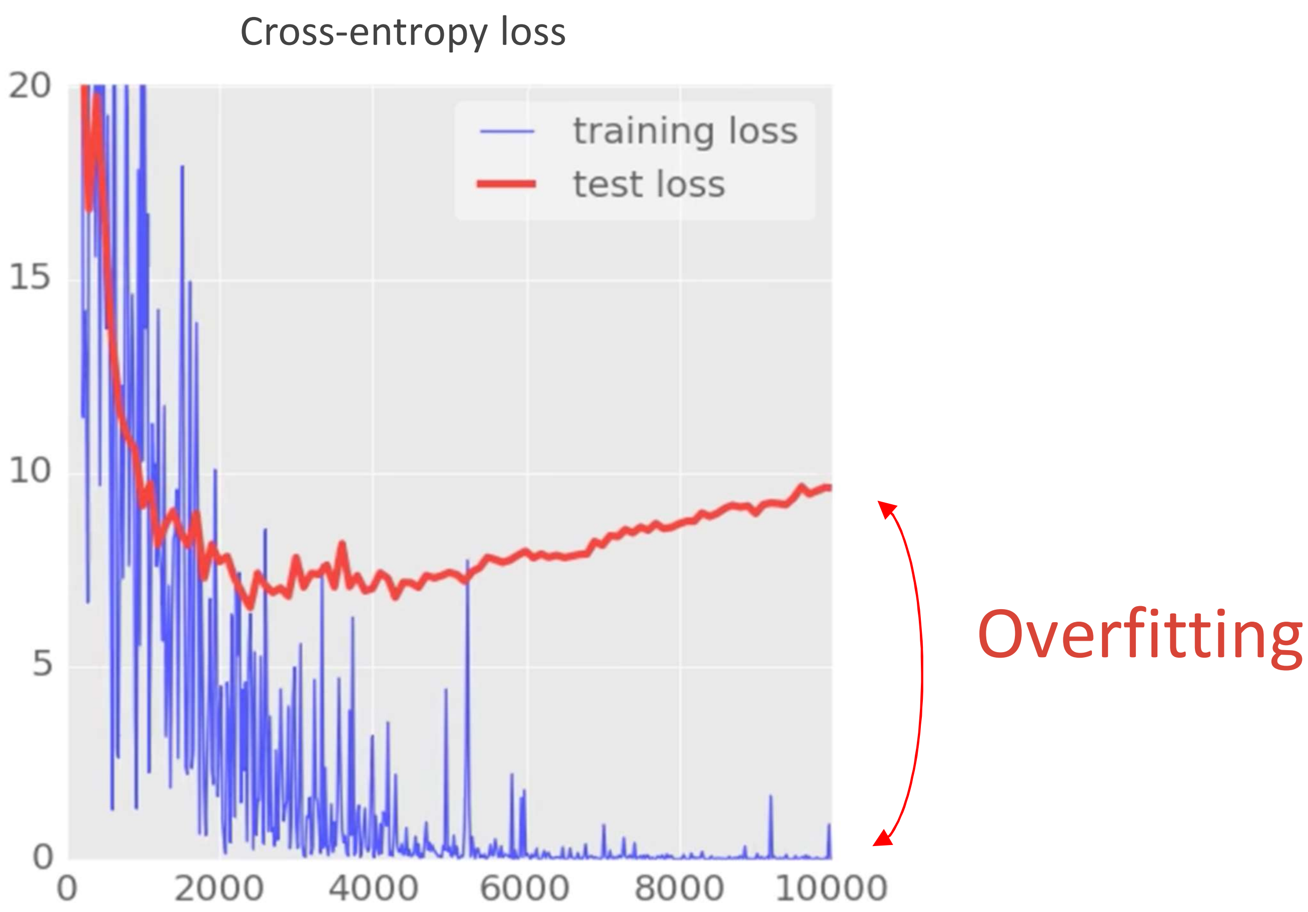




- 10000 iterations
- learning rate 0.003 at start then dropping exponentially to 0.0001

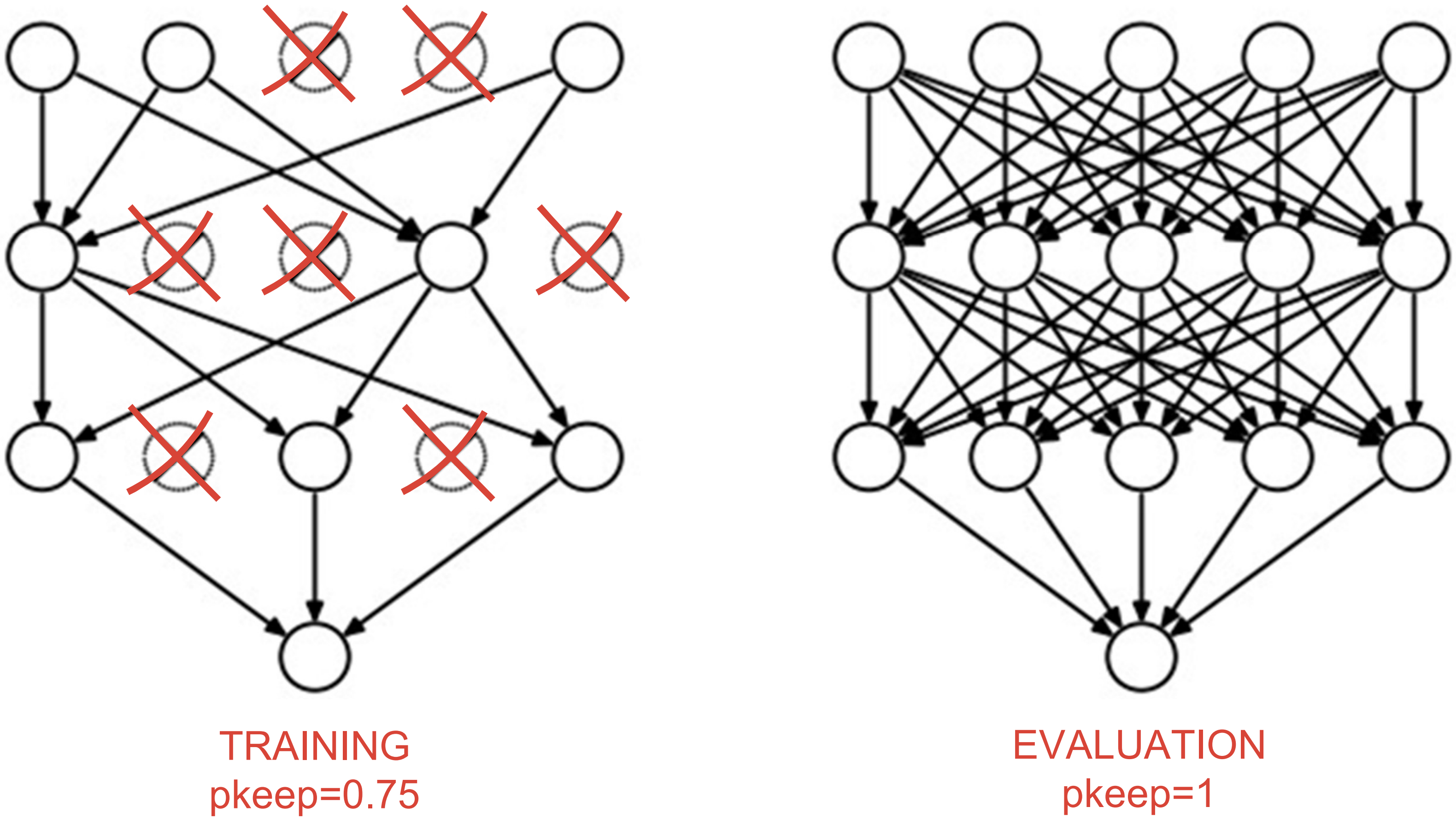
Martin Görner
[Learn TensorFlow and deep learning, without a Ph.D.]

Overfit



Martin Görner
[Learn TensorFlow and deep learning, without a Ph.D.](#)

Dropout

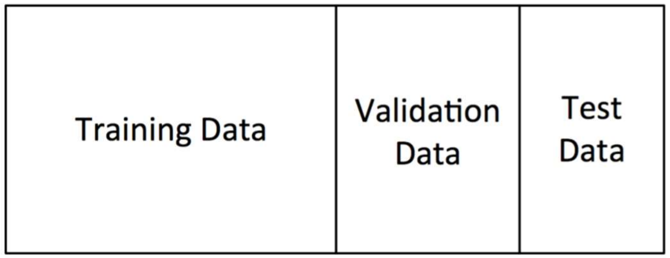


Martin Görner
Learn TensorFlow and deep learning, without a Ph.D.

Validation Set

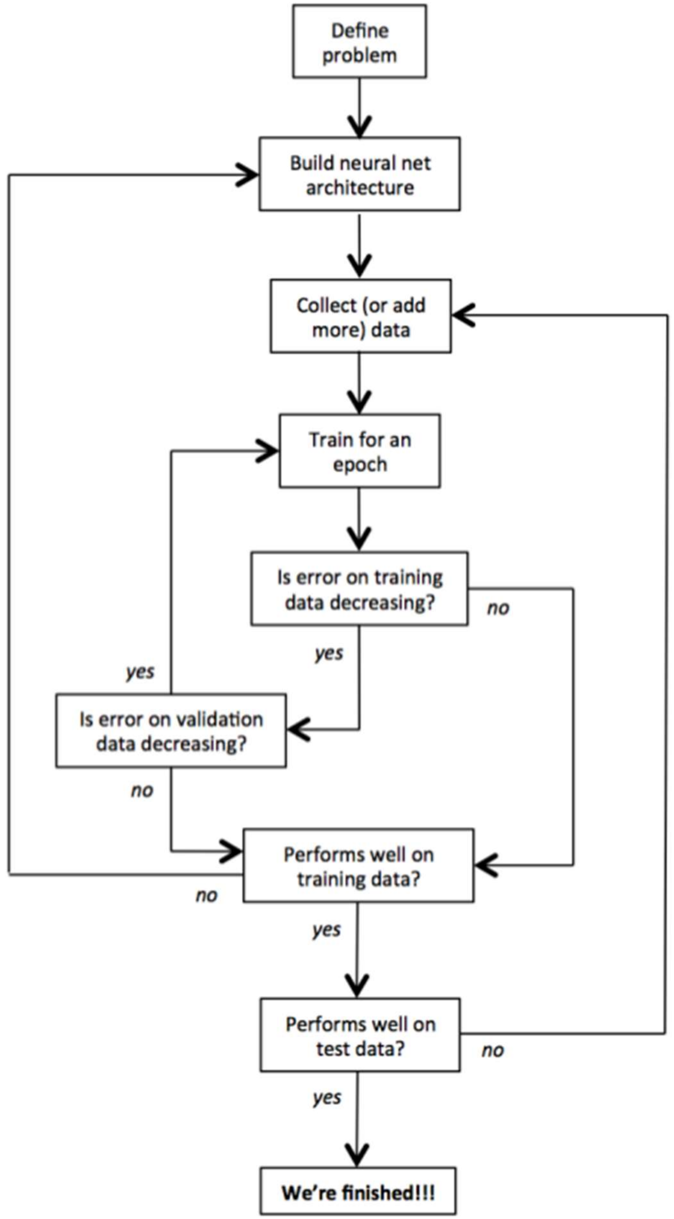
A validation set is used to prevent overfitting during the training process

Full Dataset:

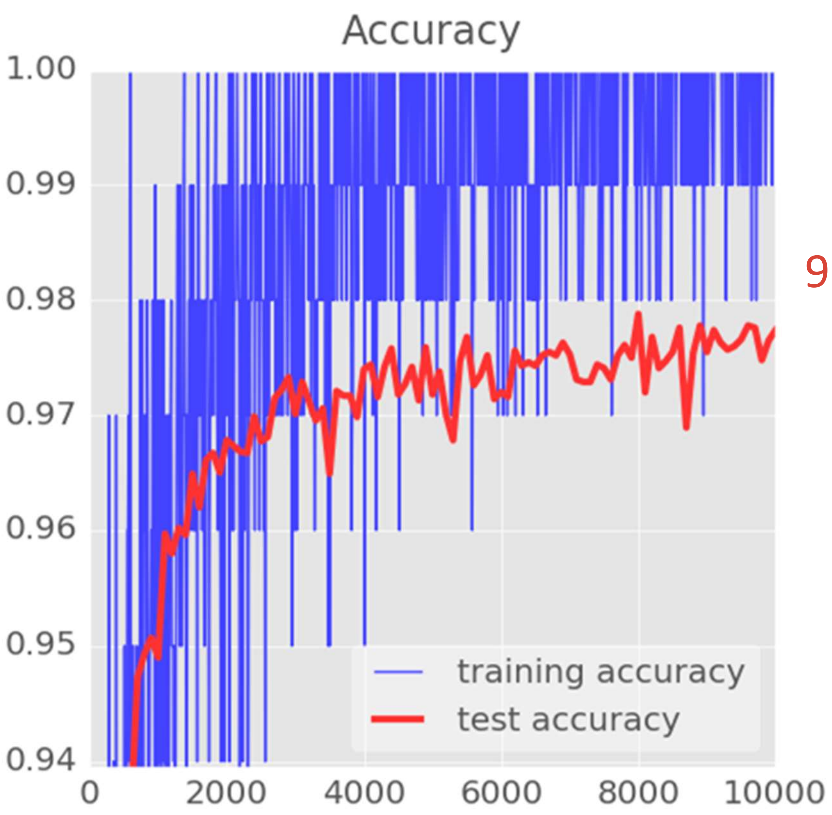


We divide our training process into **epochs** i.e., a single iteration is performed over the <u>entire</u> training set.

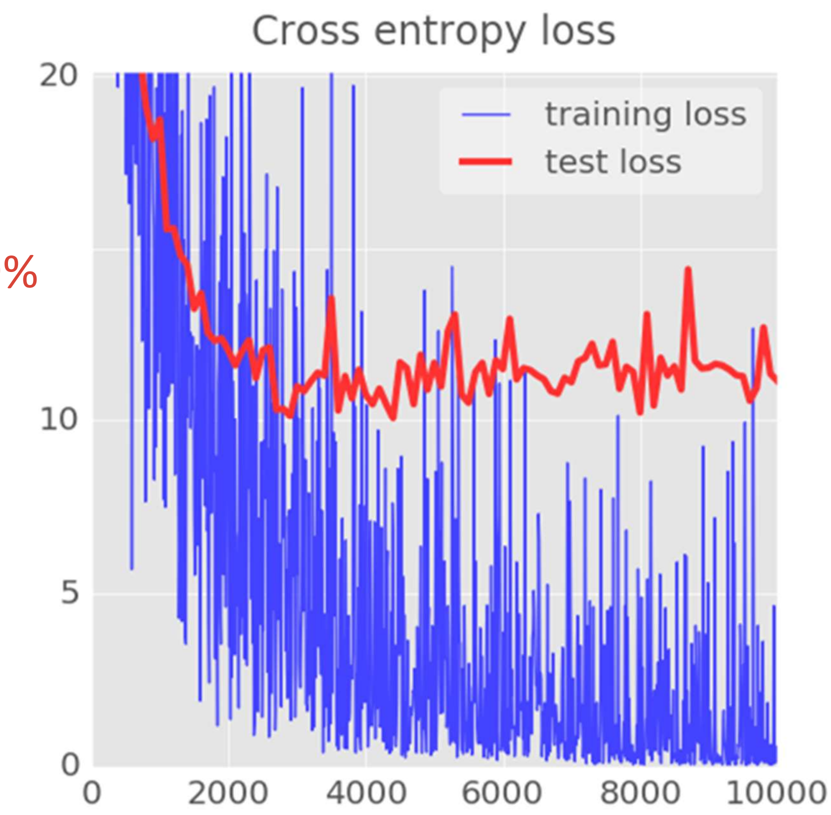
If the accuracy on the training set continues to increase while the accuracy on the validation set stays the same (or decreases)→ **overfit!**



Putting it all together



97.9%

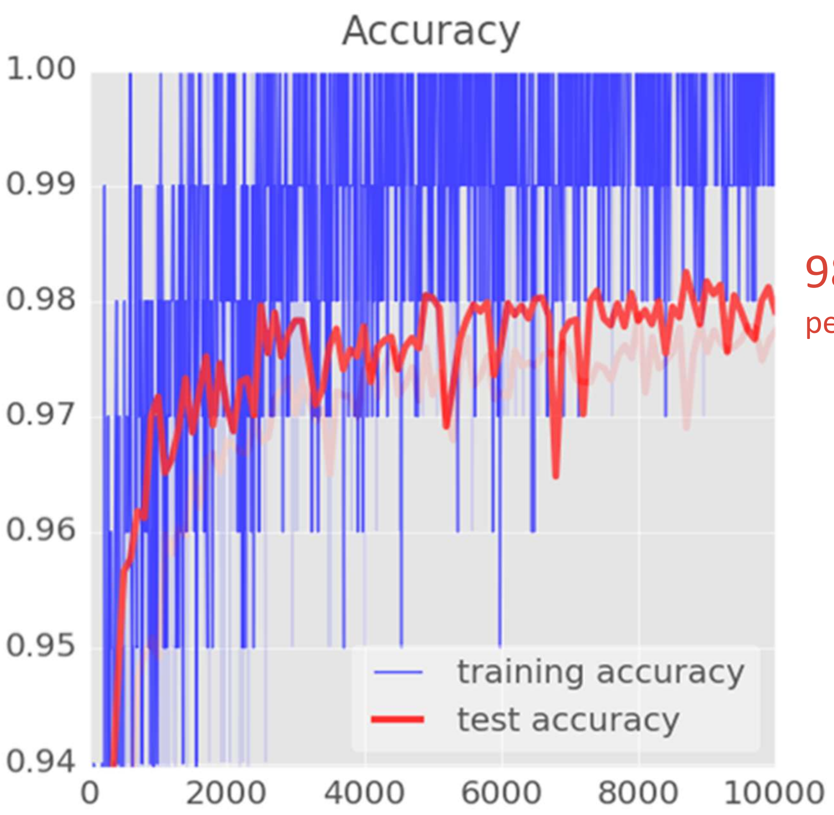


Sigmoid, learning rate = 0.003

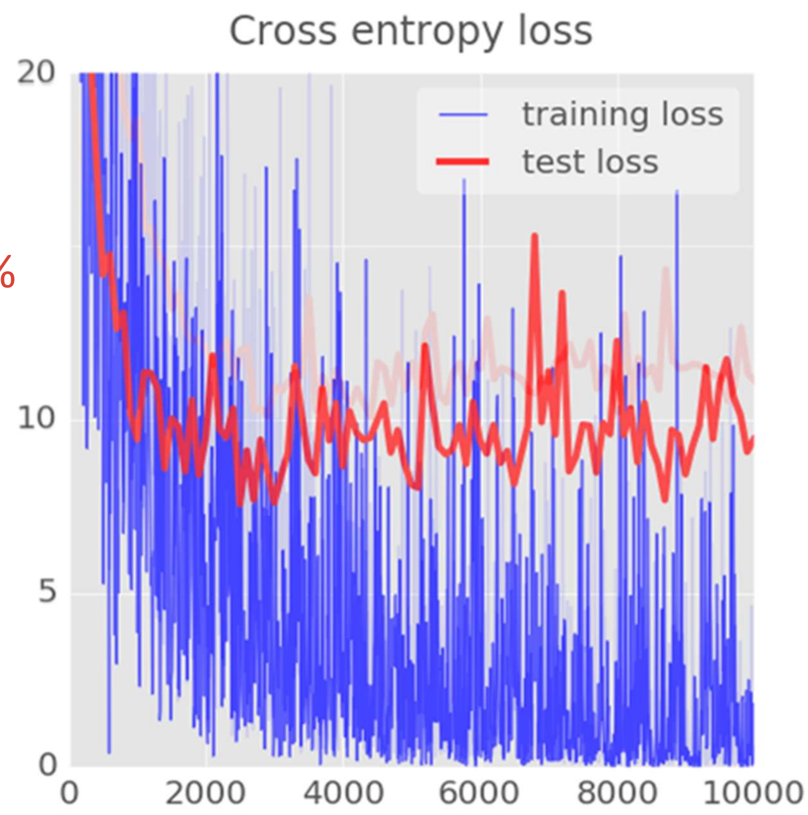
Martin Görner

Learn TensorFlow and deep learning, without a Ph.D.

Putting it all together



98.2%
peak

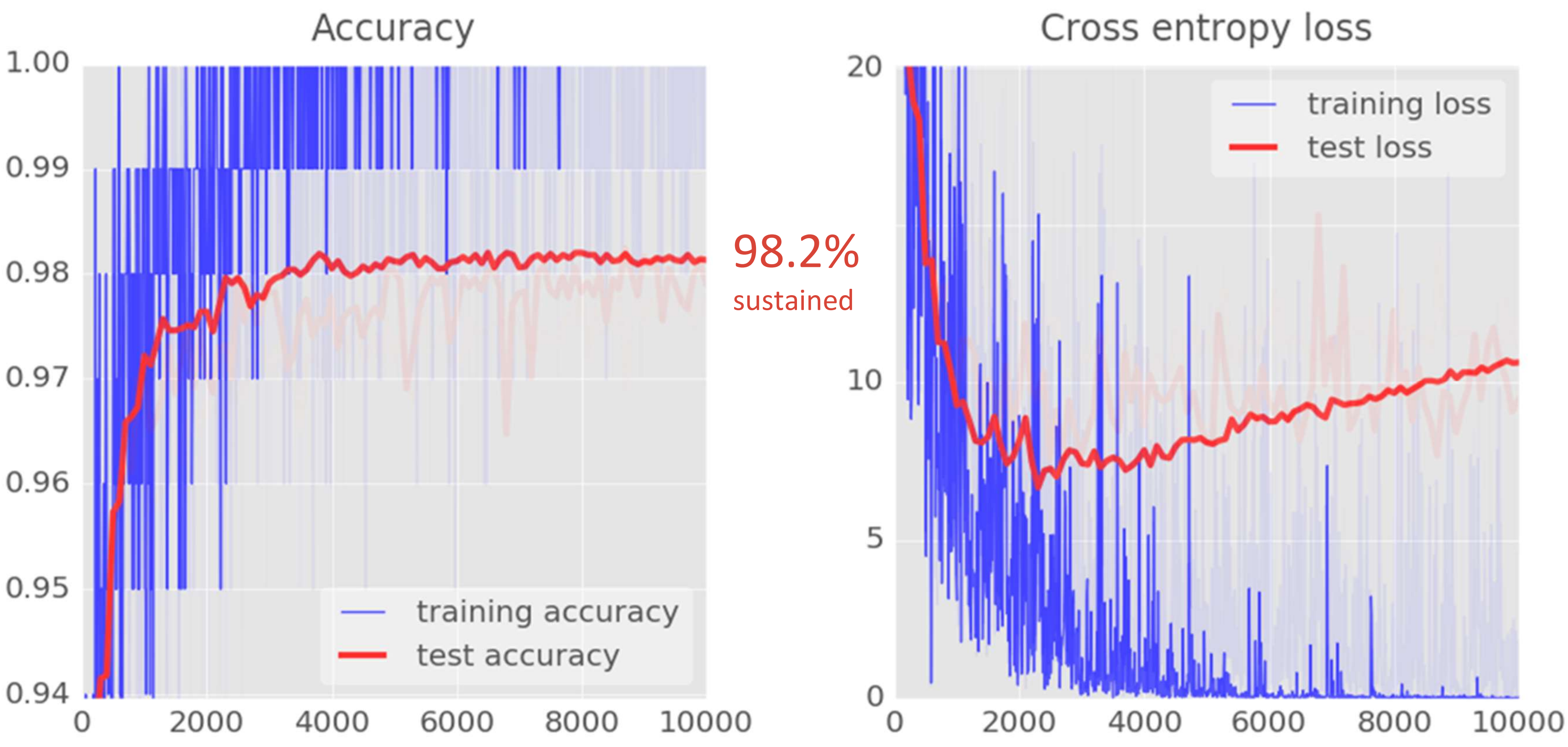


RELU, learning rate = 0.003

Martin Görner

[Learn TensorFlow and deep learning, without a Ph.D.]

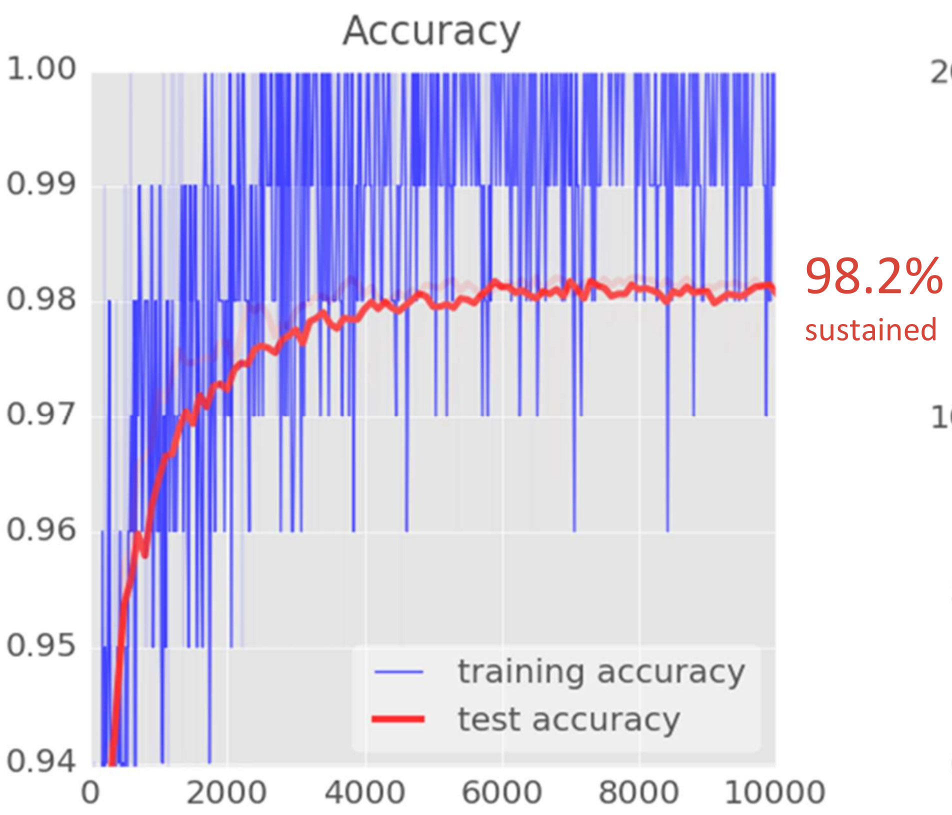
Putting it all together

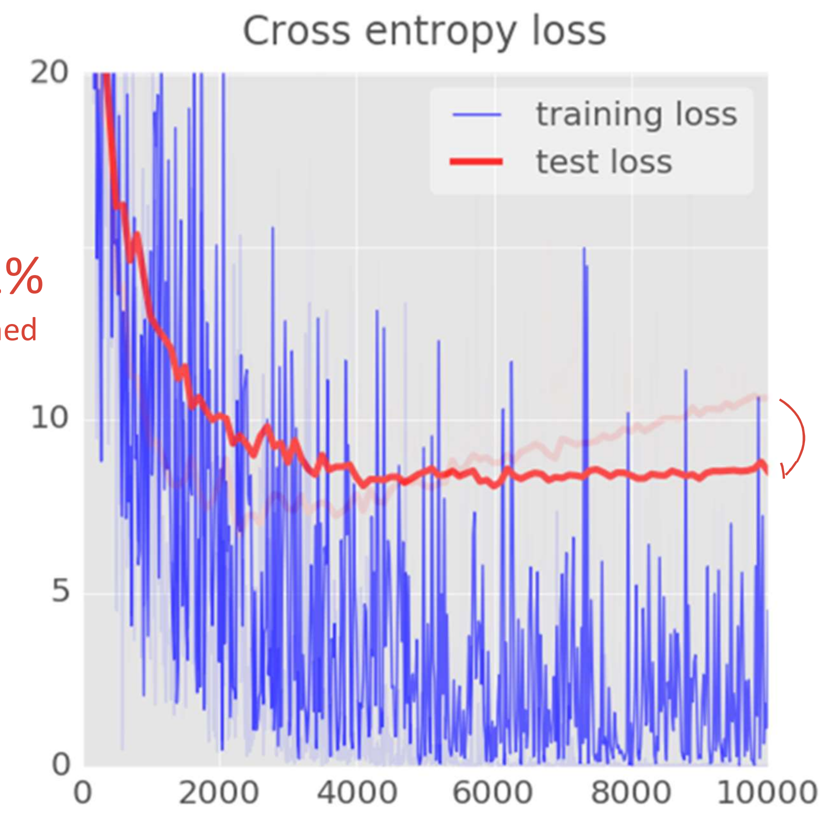


RELU, decaying learning rate 0.003 -> 0.0001

Martin Görner
[Learn TensorFlow and deep learning, without a Ph.D.]

Putting it all together





RELU, decaying learning rate 0.003 -> 0.0001 and dropout 0.75

Martin Görner

[Learn TensorFlow and deep learning, without a Ph.D.]

How to improve the accuracy

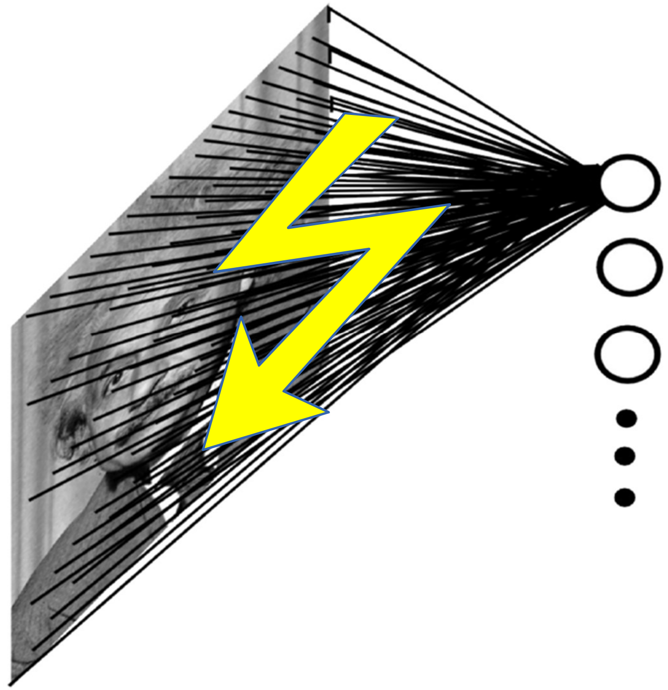
- 98.2% accuracy is not enough! We want to increase the accuracy
- A multilayer perceptron network requires a vector in input, so we have flattened our input image
- However, images contains information at pixel level and also at local (neighborhood-) level
- By flattening the image, we lose this information

From Neural Network to CNNs

Applying DNN to images to perform classification, detection, etc… by using fully connected layers is infeasible

200x200 RGB image in input

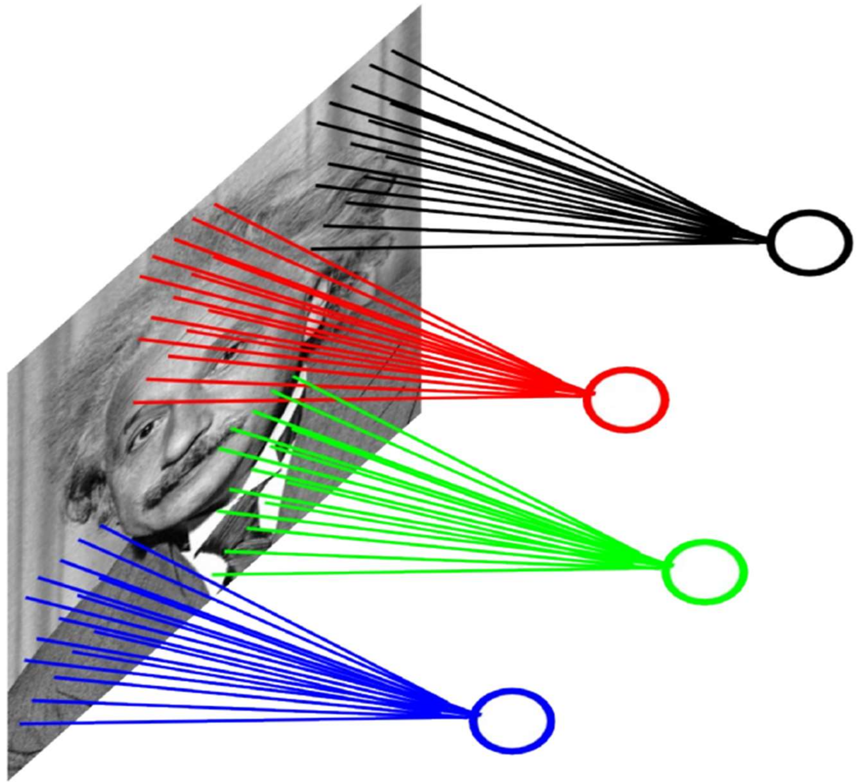
120000 parameters for each node!!



Convolutional Neural Networks

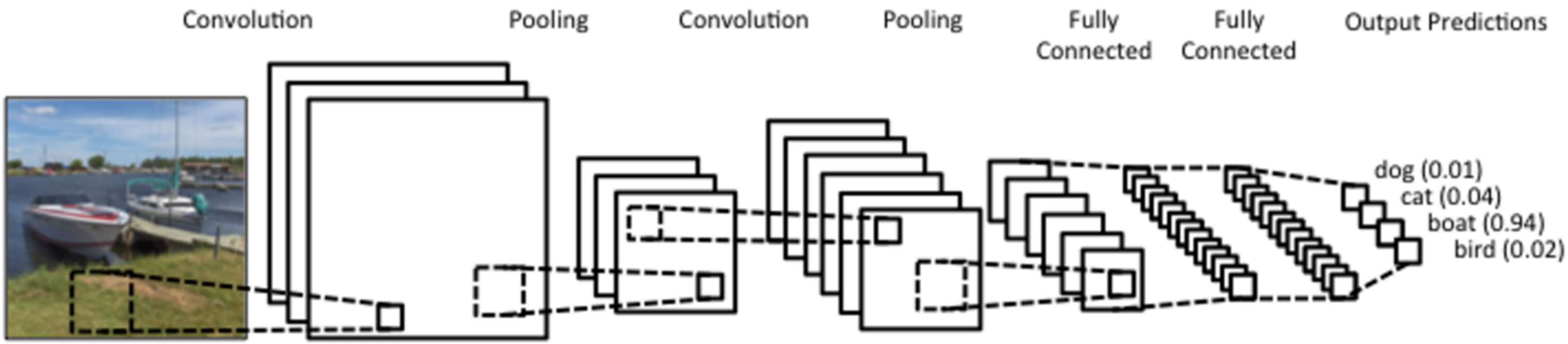
Convolutional Neural Networks use three basic ideas:

1. **local receptive fields**
2. **shared weights**
3. **pooling**

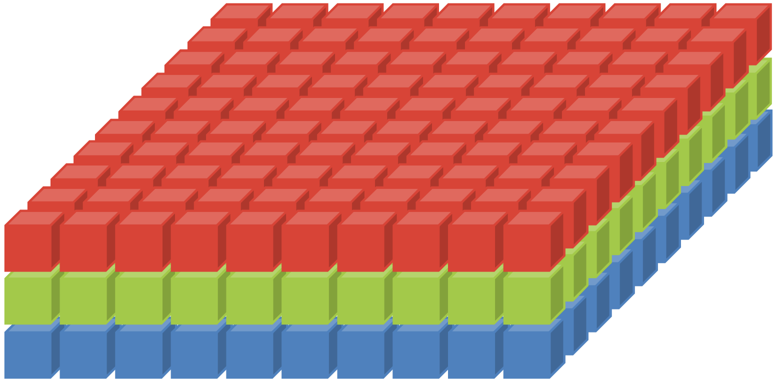


CNNs

- unit connectivity pattern inspired by the organization of the visual cortex
- units respond to stimuli in a restricted region of space known as the receptive field
- receptive fields partially overlap, over-covering the entire visual field

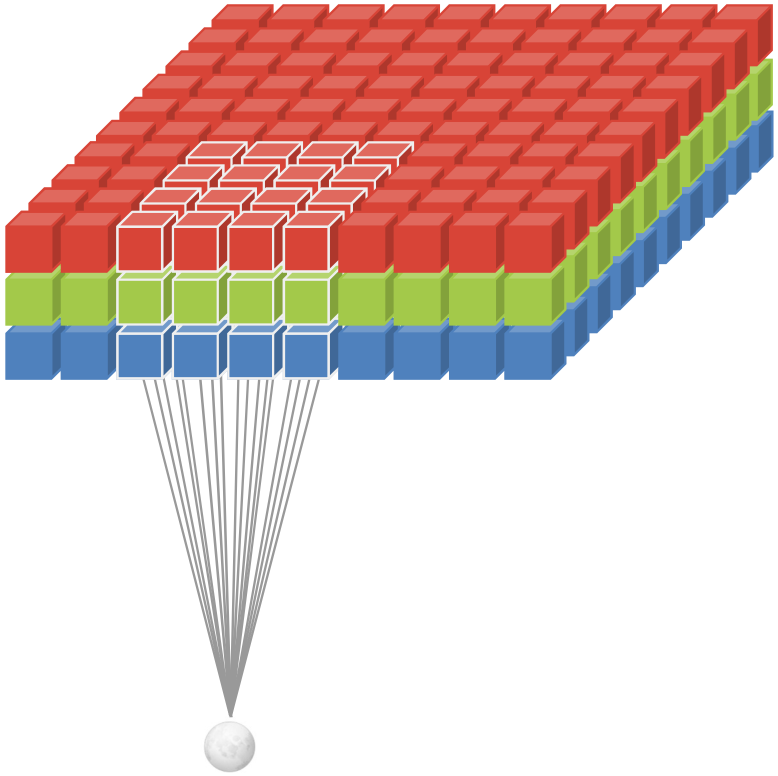


Convolutional layer



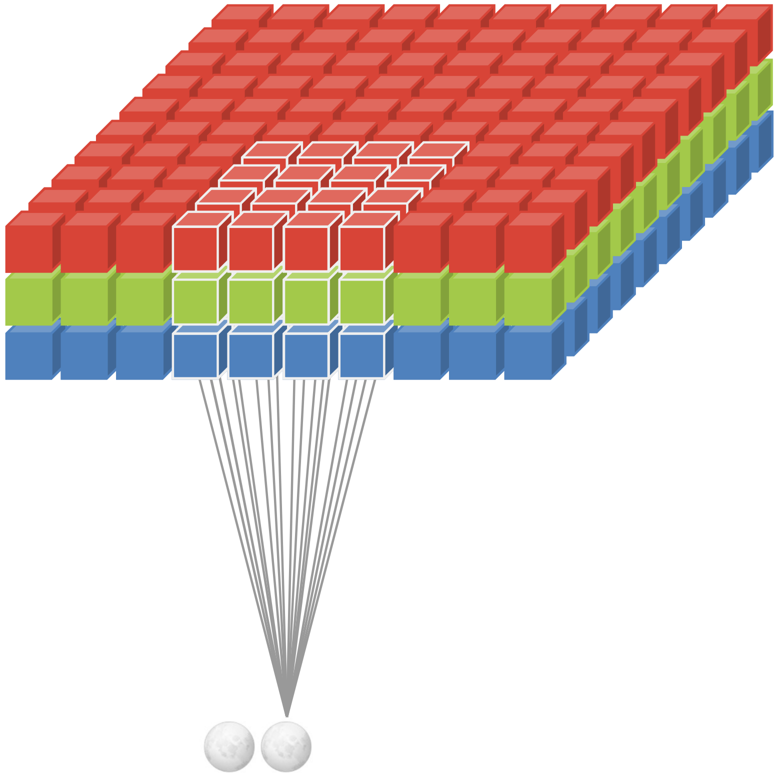
Martin Görner
[Learn TensorFlow and deep learning, without a Ph.D.]

Convolutional layer



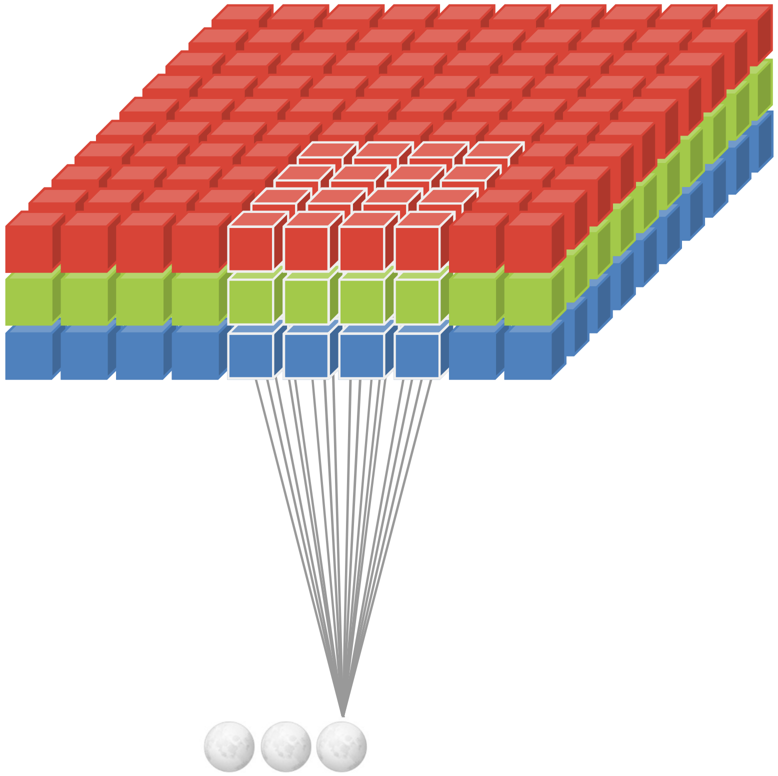
Martin Görner
Learn TensorFlow and deep learning, without a Ph.D.

Convolutional layer



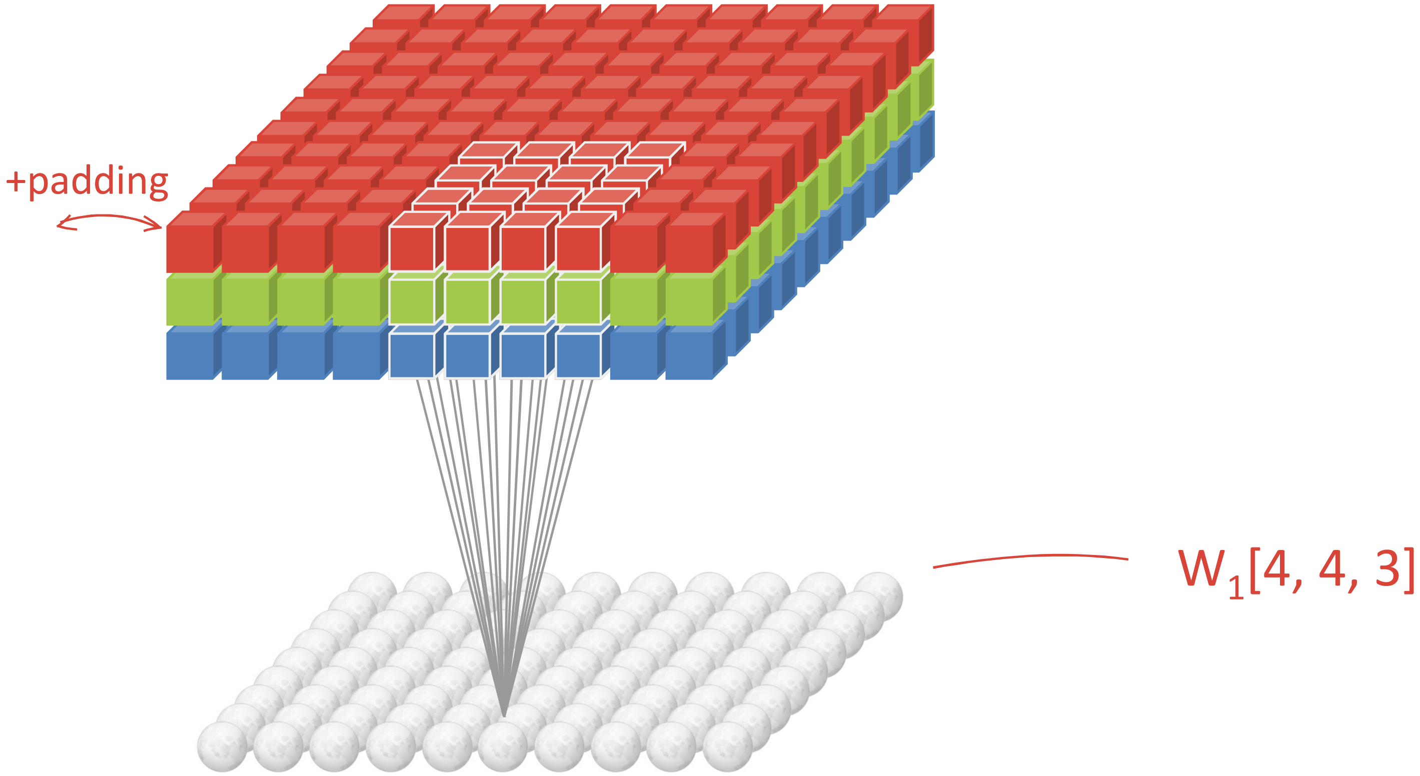
Martin Görner
[Learn TensorFlow and deep learning, without a Ph.D.](#)

Convolutional layer



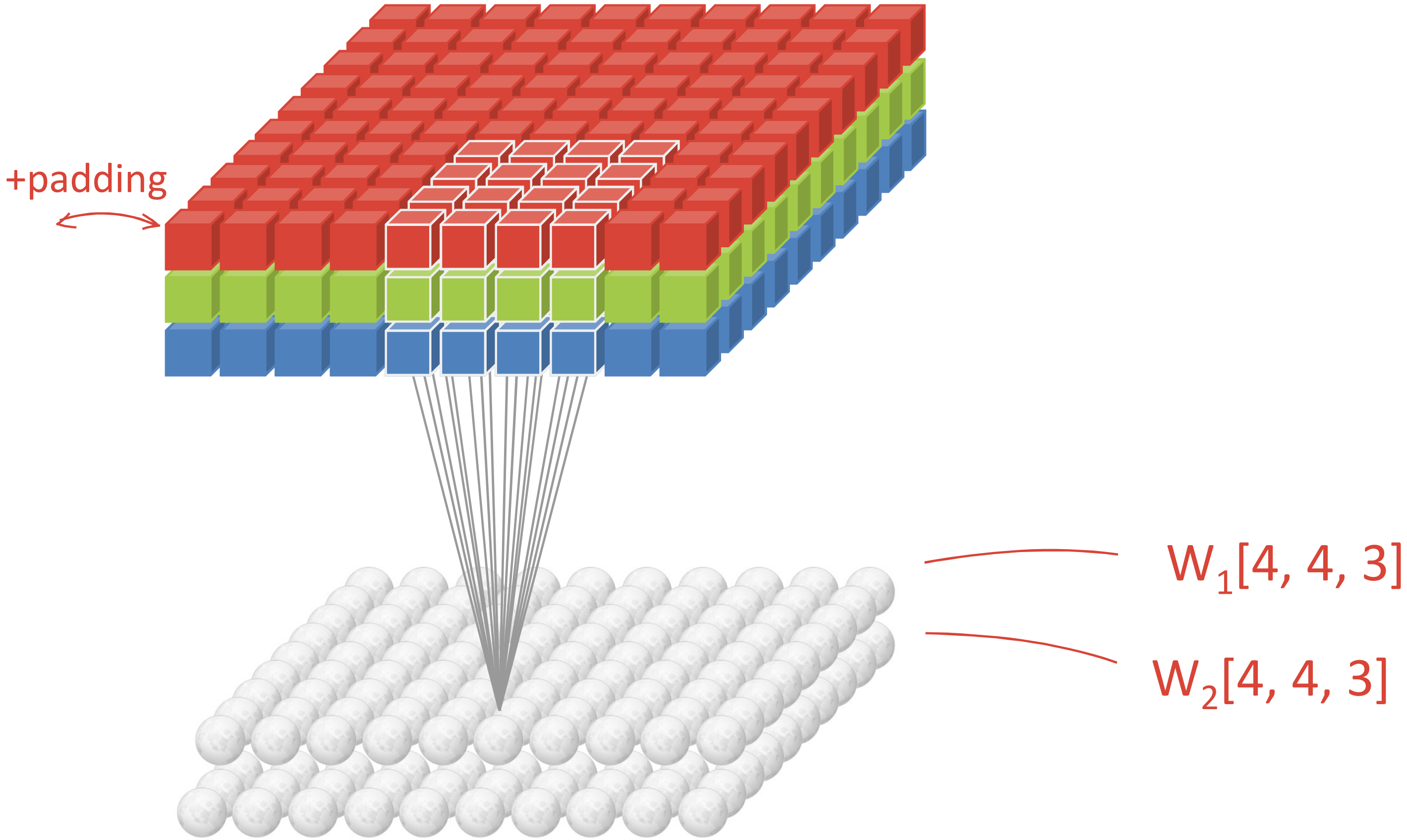
Martin Görner
Learn TensorFlow and deep learning, without a Ph.D.

Convolutional layer



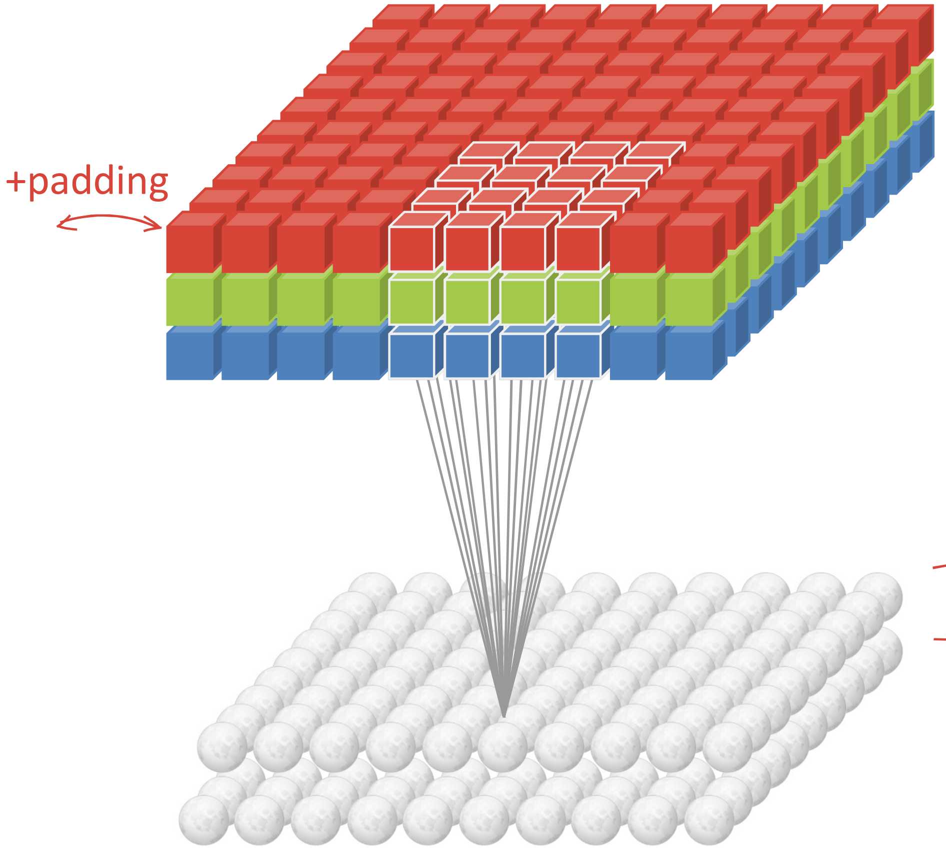
Martin Görner
[Learn TensorFlow and deep learning, without a Ph.D.]

Convolutional layer



Martin Görner
[Learn TensorFlow and deep learning, without a Ph.D.]()

Convolutional layer



$W_1[4, 4, 3]$

$W_2[4, 4, 3]$

$W[4, 4, 3, 2]$

filter size

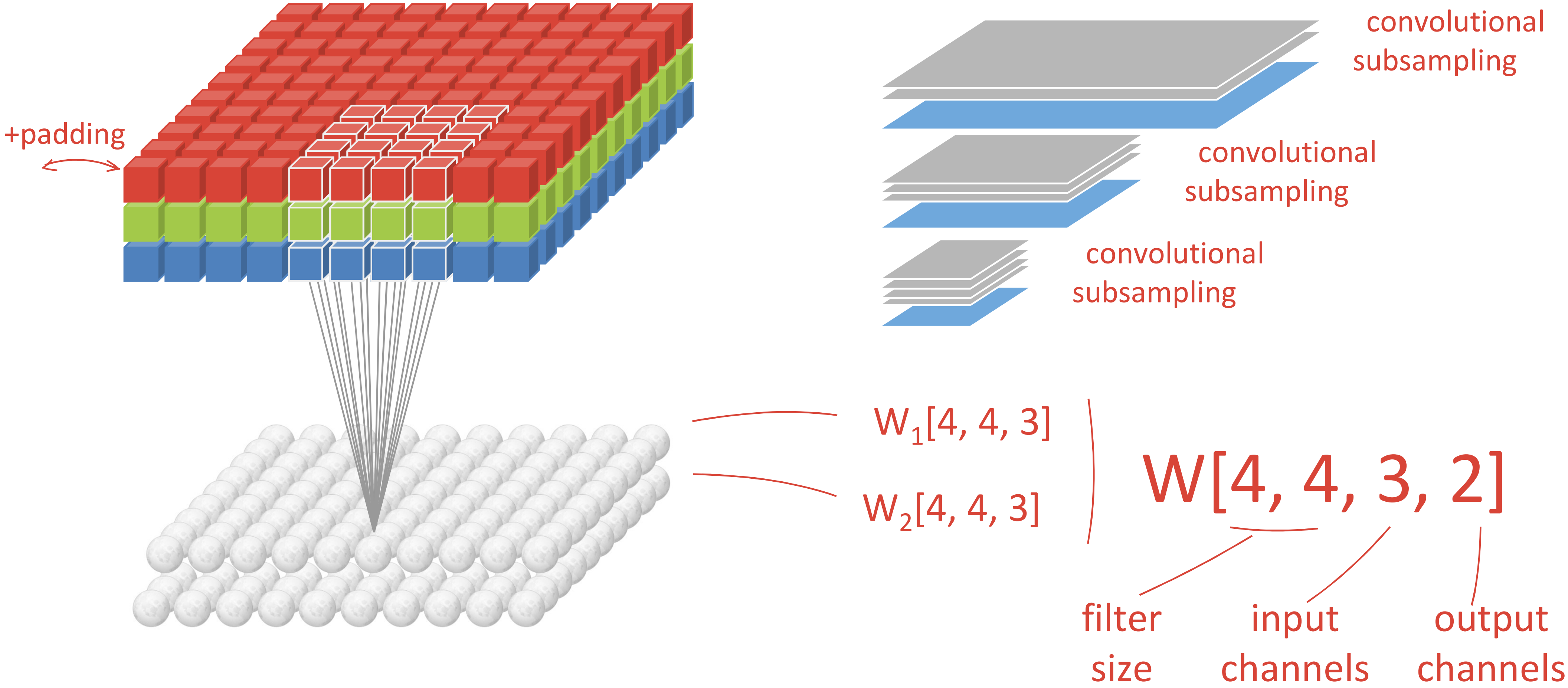
input channels

output channels

Martin Görner

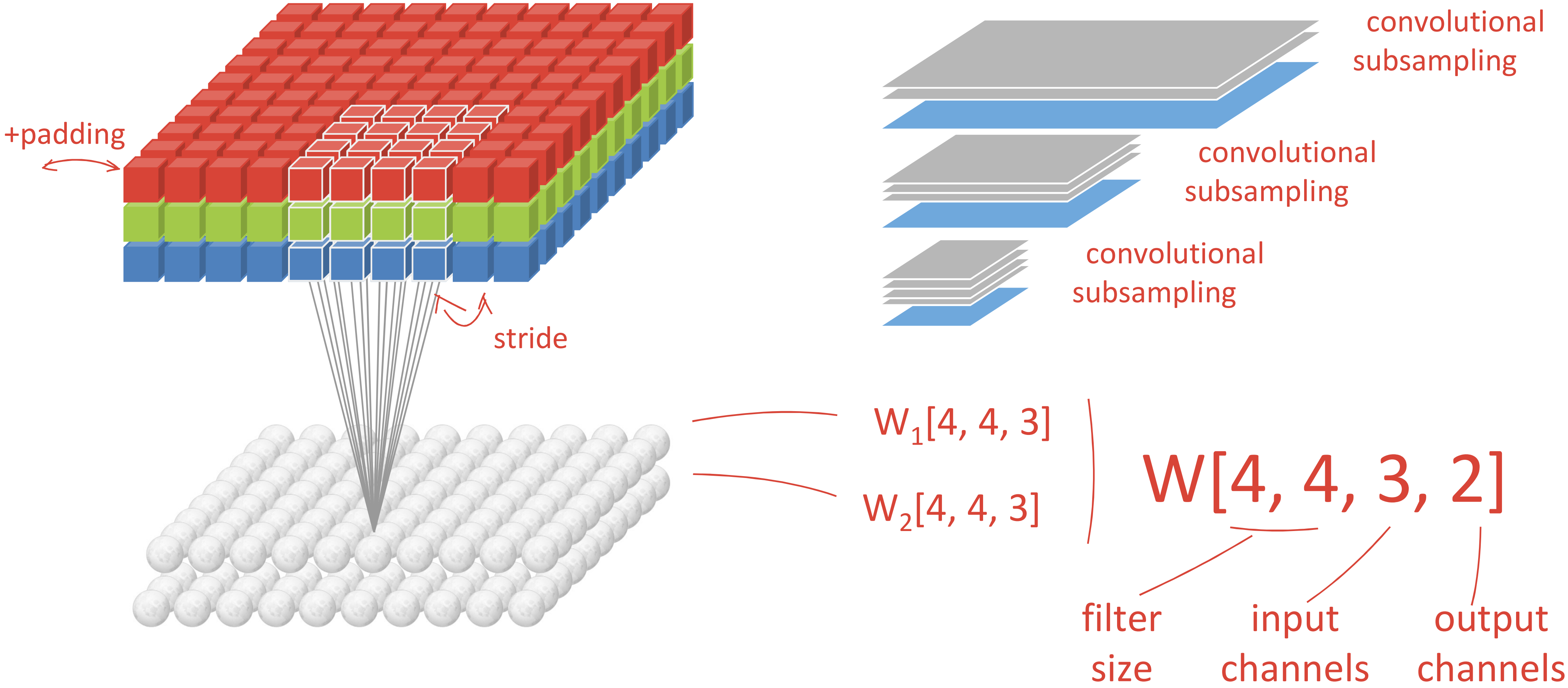
[Learn TensorFlow and deep learning, without a Ph.D.]

Convolutional layer



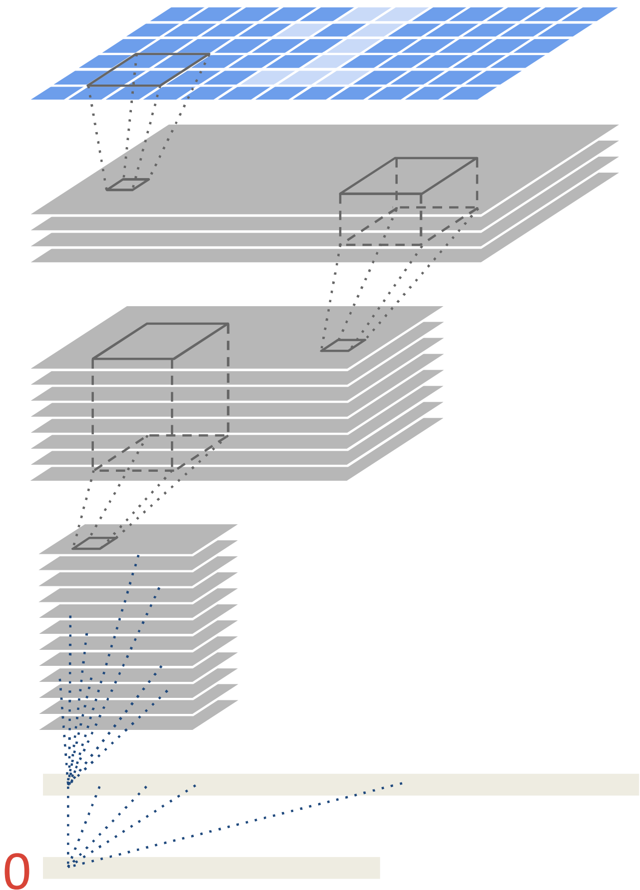
Martin Görner
[Learn TensorFlow and deep learning, without a Ph.D.](#)

Convolutional layer



Martin Görner
[Learn TensorFlow and deep learning, without a Ph.D.]()

Convolutional neural network



+ biases on all layers

28x28x1

convolutional layer, 4 channels

W1[5, 5, 1, 4] stride 1

28x28x4

convolutional layer, 8 channels

W2[4, 4, 4, 8] stride 2

14x14x8

convolutional layer, 12 channels

W3[4, 4, 8, 12] stride 2

7x7x12

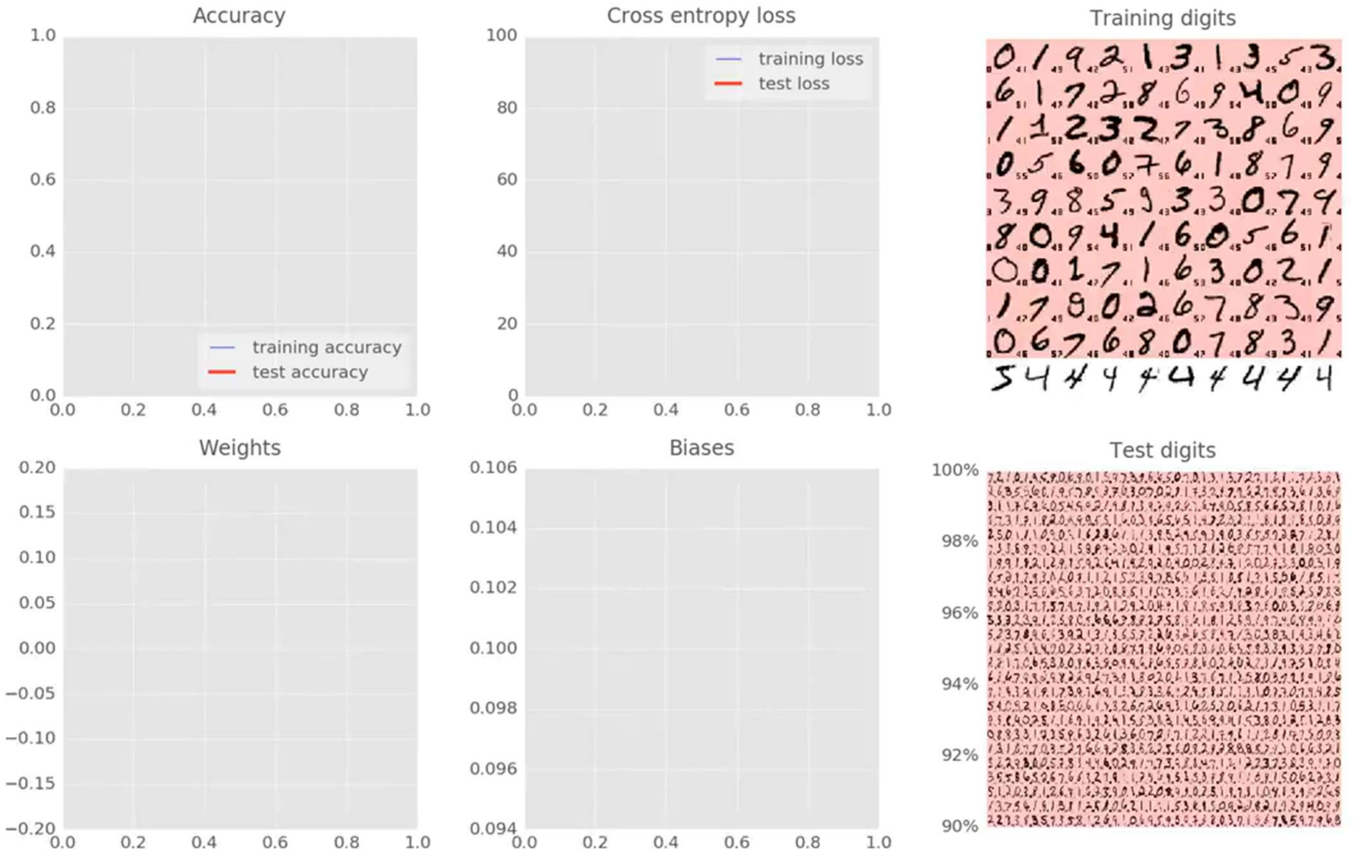
200 fully connected layer W4[7x7x12, 200]

10 softmax readout layer W5[200, 10]

Martin Görner

[Learn TensorFlow and deep learning, without a Ph.D.]

Training CNN

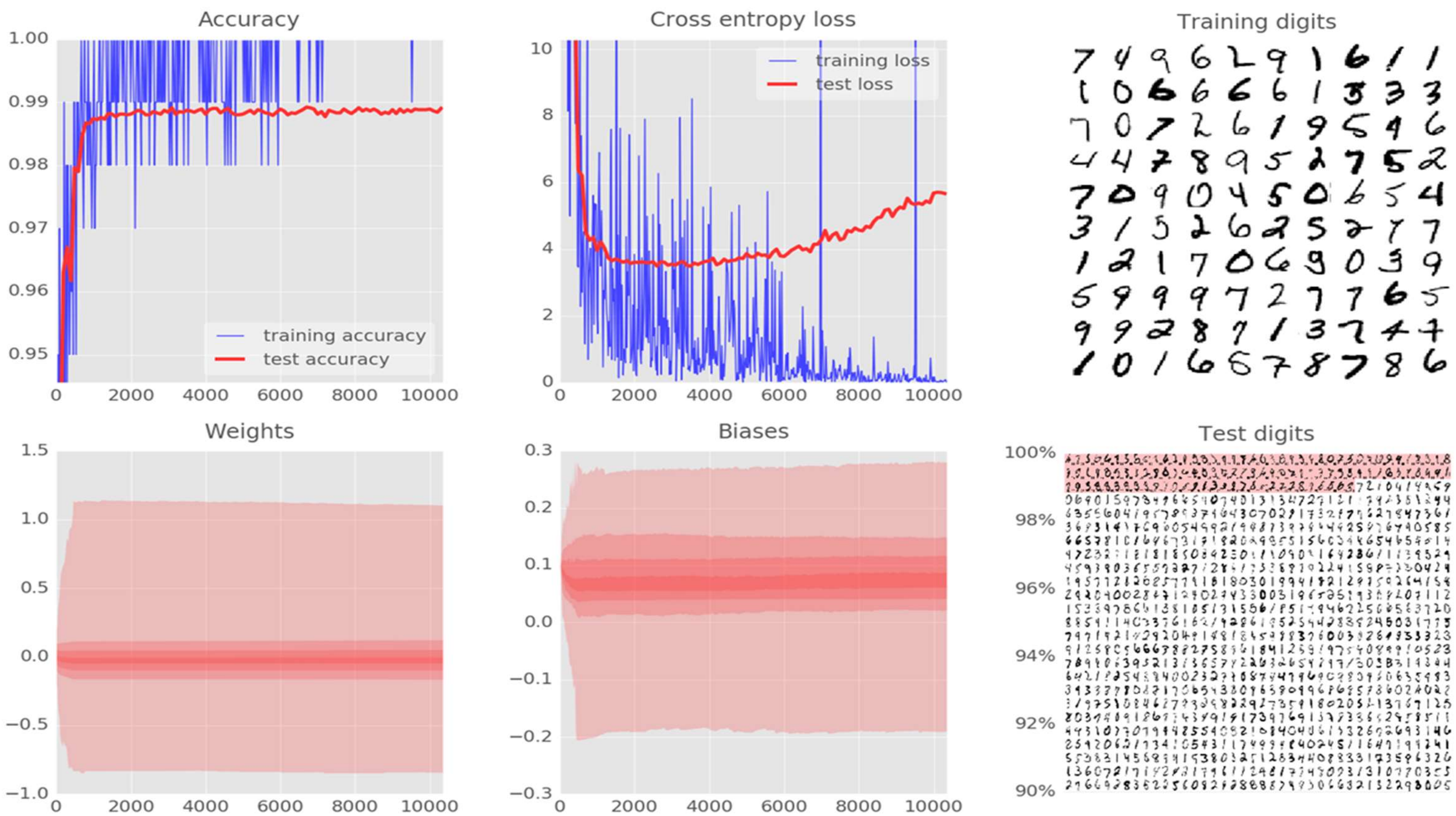


Martin Görner

[Learn TensorFlow and deep learning, without a Ph.D.]

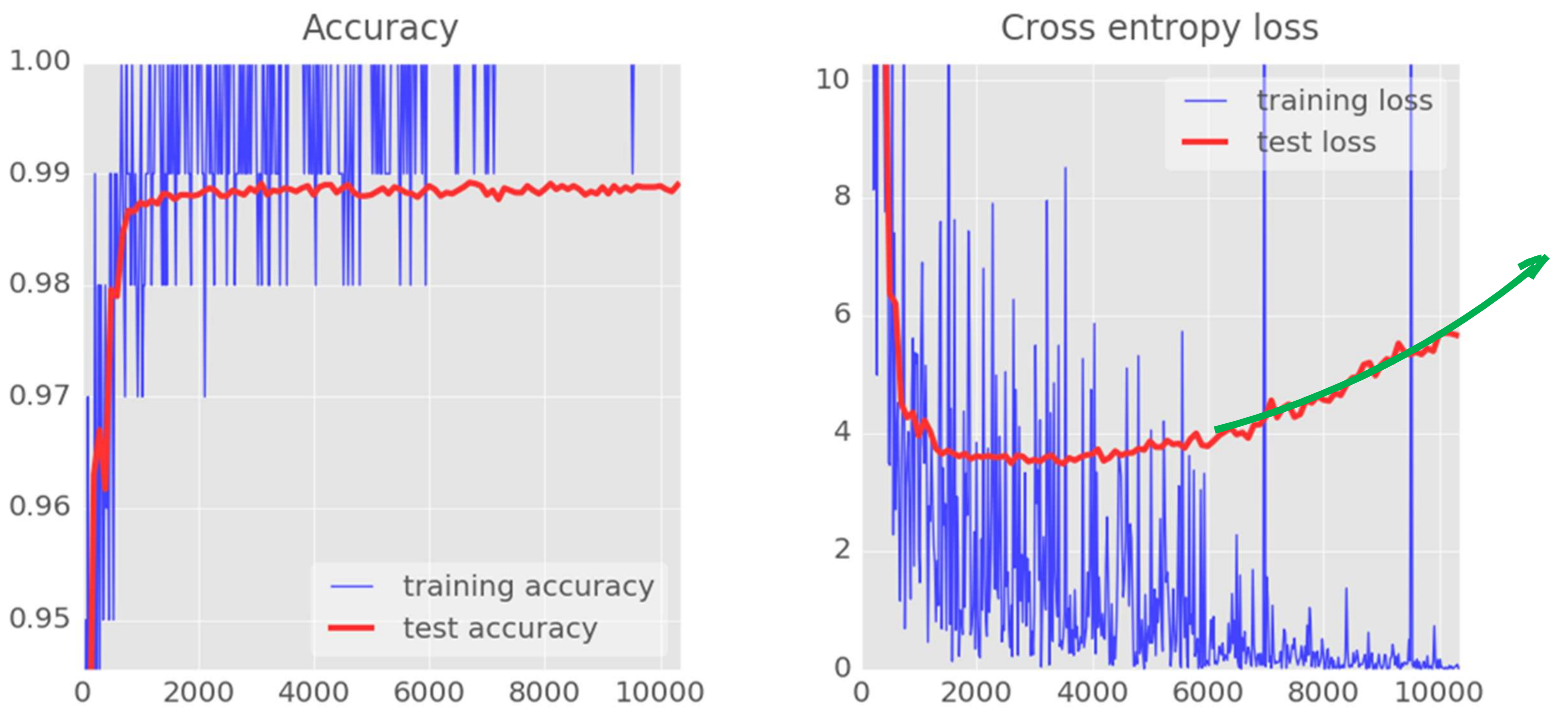
Accuracy CNN

98.9%



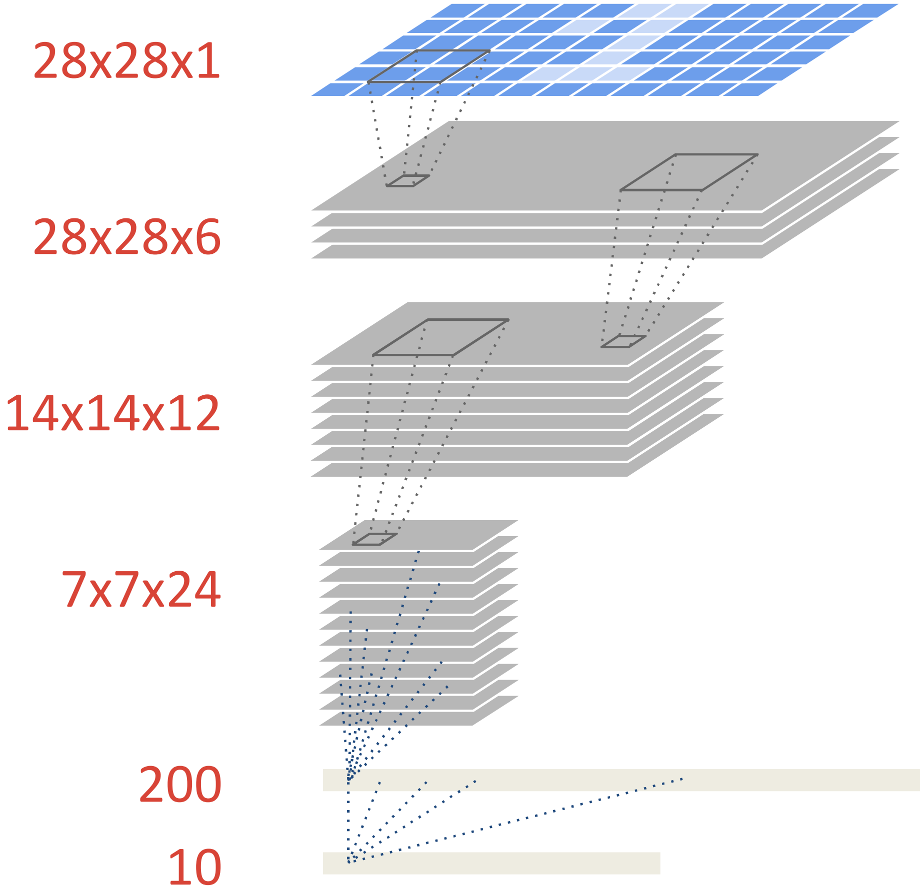
Martin Görner
[Learn TensorFlow and deep learning, without a Ph.D.]

Overfitting?



Martin Görner
Learn TensorFlow and deep learning, without a Ph.D.

Overfitting? Bigger Network + dropout



+ biases on all layers

convolutional layer, 6 channels
W1[6, 6, 1, 6] stride 1

convolutional layer, 12 channels
W2[5, 5, 6, 12] stride 2

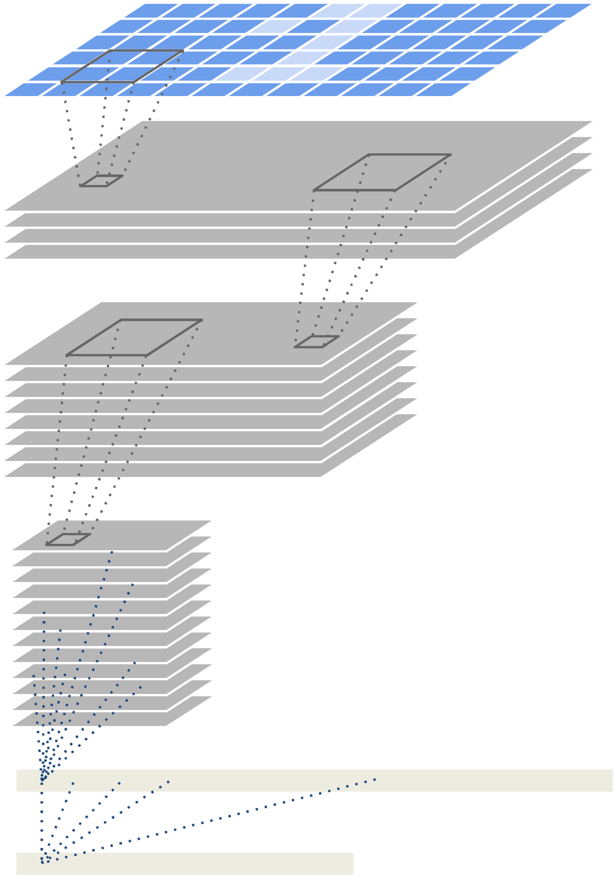
convolutional layer, 24 channels
W3[4, 4, 12, 24] stride 2

fully connected layer W4[7x7x24, 200]
softmax readout layer W5[200, 10]

Martin Görner
Learn TensorFlow and deep learning, without a Ph.D.

Overfitting? Bigger Network + dropout

+ biases on all layers



28x28x1

28x28x6

14x14x12

7x7x24

200

10

convolutional layer, **6 channels**

W1[**6, 6**, 1, 6] stride 1

convolutional layer, **12 channels**

W2[**5, 5**, 6, 12] stride 2

convolutional layer, **24 channels**

W3[**4, 4**, 12, 24] stride 2

fully connected layer W4[7x7x24, 200]

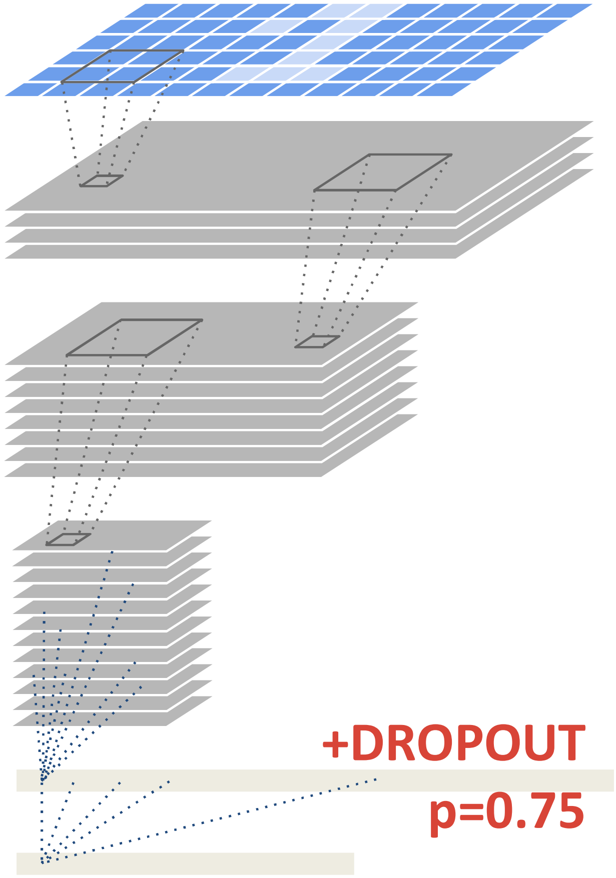
softmax readout layer W5[200, 10]

Martin Görner

[Learn TensorFlow and deep learning, without a Ph.D.](#)

Overfitting? Bigger Network + dropout

+ biases on all layers



convolutional layer, **6 channels**
W1[**6, 6**, 1, 6] stride 1

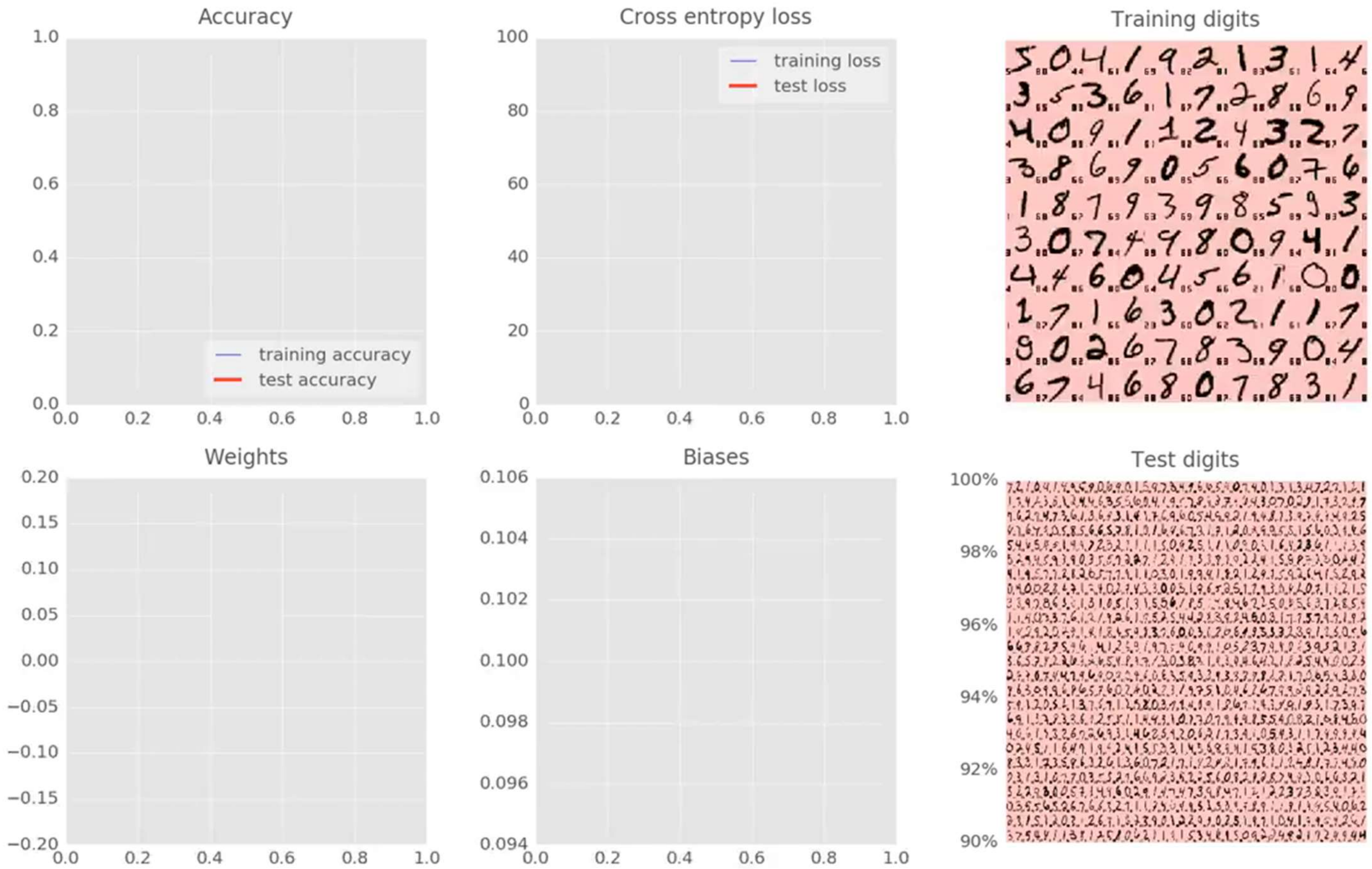
convolutional layer, **12 channels**
W2[**5, 5**, 6, 12] stride 2

convolutional layer, **24 channels**
W3[**4, 4**, 12, 24] stride 2

fully connected layer W4[7x7x24, 200]
softmax readout layer W5[200, 10]

Martin Görner
[Learn TensorFlow and deep learning, without a Ph.D.]

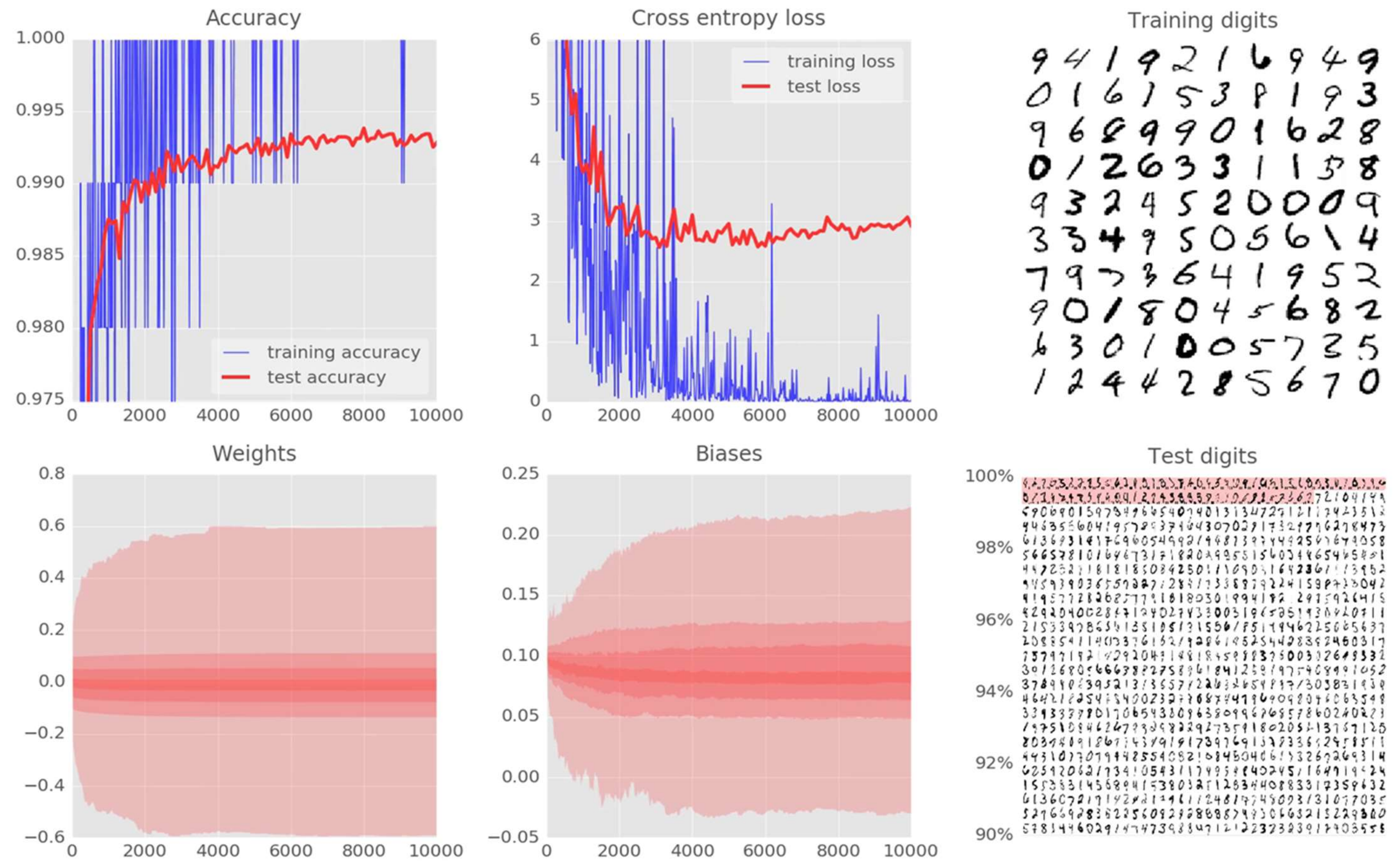
Training Bigger Network + dropout



Martin Görner
[Learn TensorFlow and deep learning, without a Ph.D.]

Accuracy Bigger Network + dropout

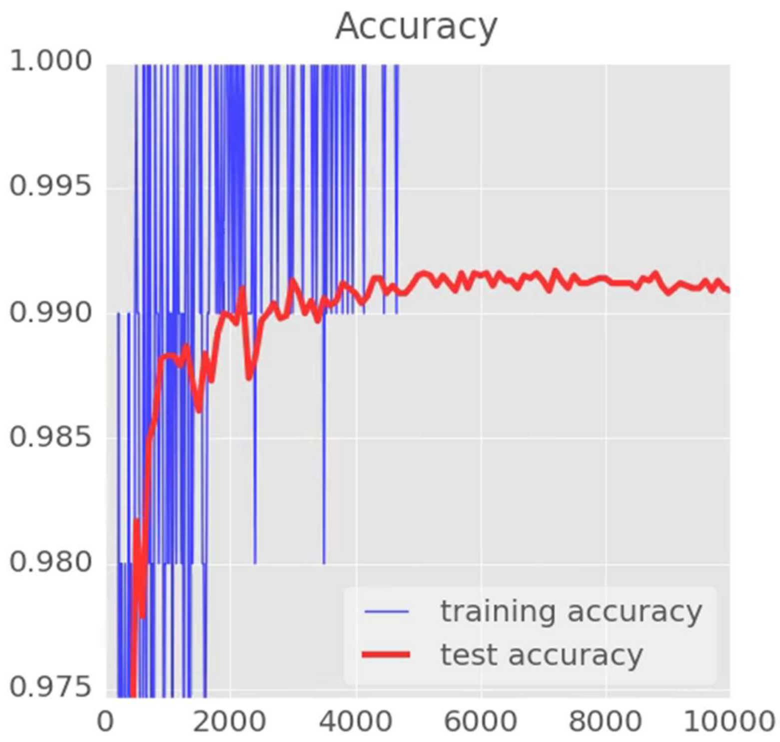
99.3%

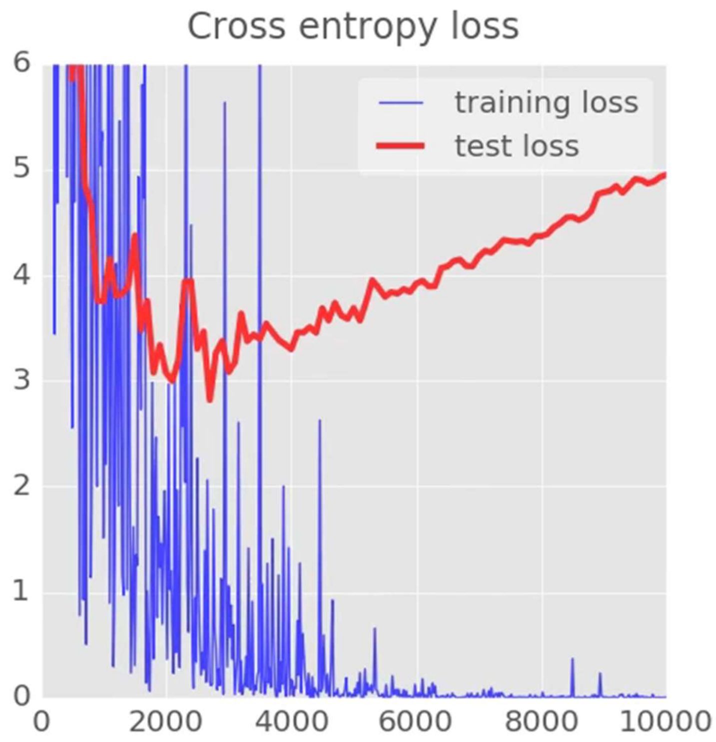


Martin Görner

[Learn TensorFlow and deep learning, without a Ph.D.]

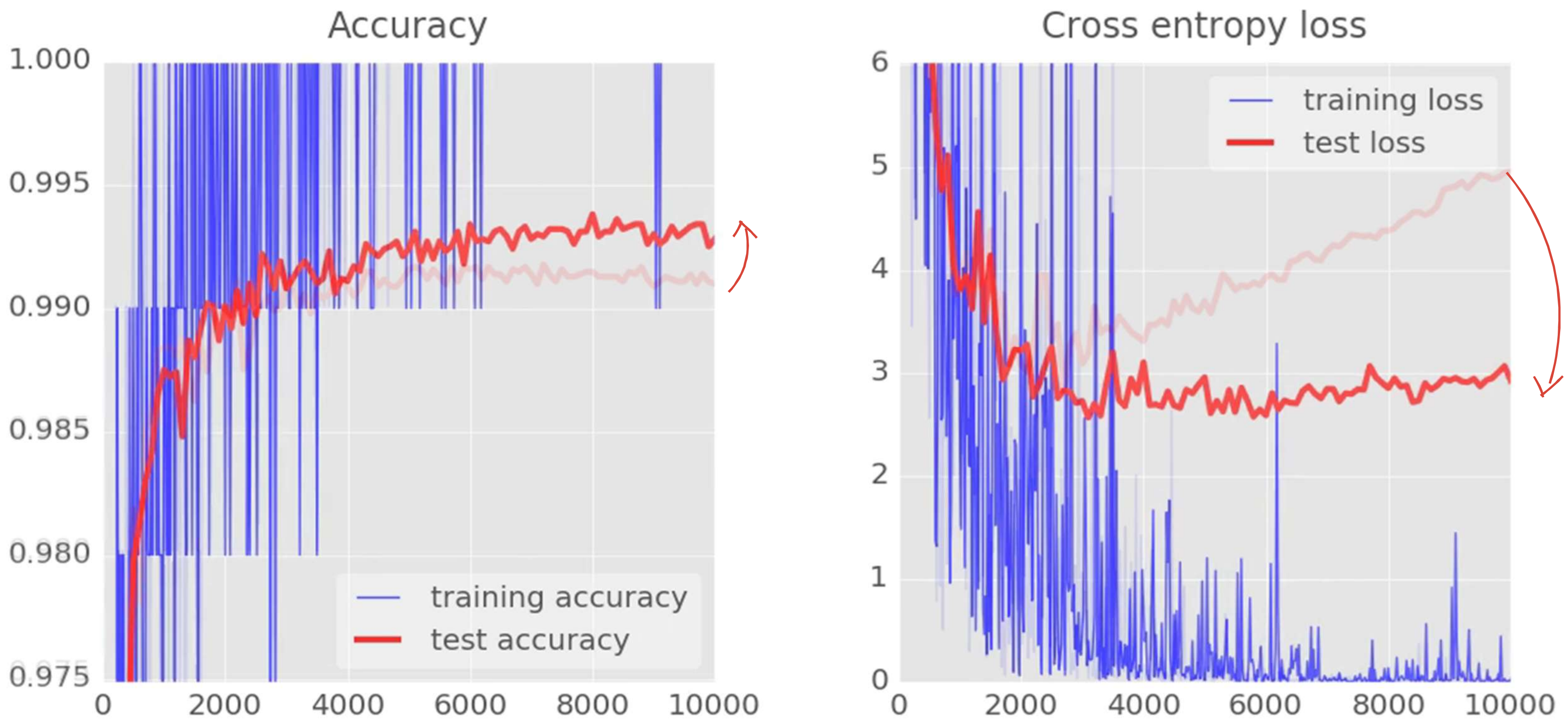
Mission accomplished!





Martin Görner
[Learn TensorFlow and deep learning, without a Ph.D.]

Mission accomplished!



with dropout

Martin Görner
[Learn TensorFlow and deep learning, without a Ph.D.](#)

UNIVERSITÀ DEGLI STUDI DELLA BASILICATA



Corso di Visione e Percezione

Docente

Domenico D. Bloisi

Introduzione al Deep Learning