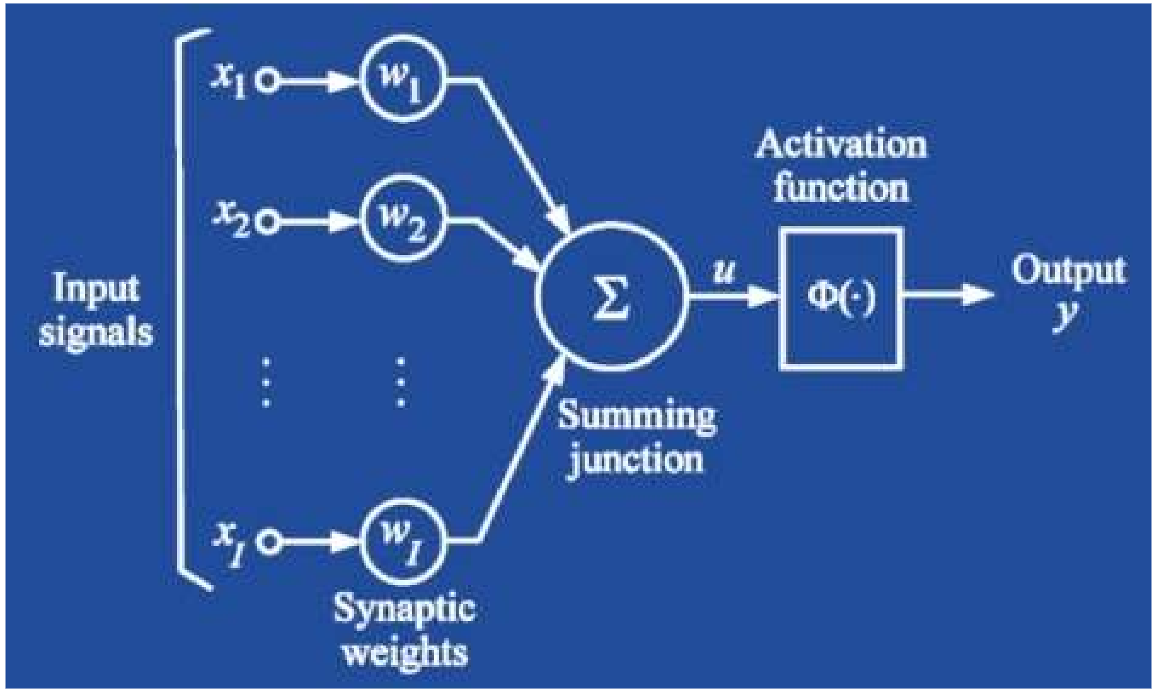
# Deep Learning for Developers

Julien Simon
Principal Technical Evangelist, AI & Machine Learning
@julsimon

July 2018

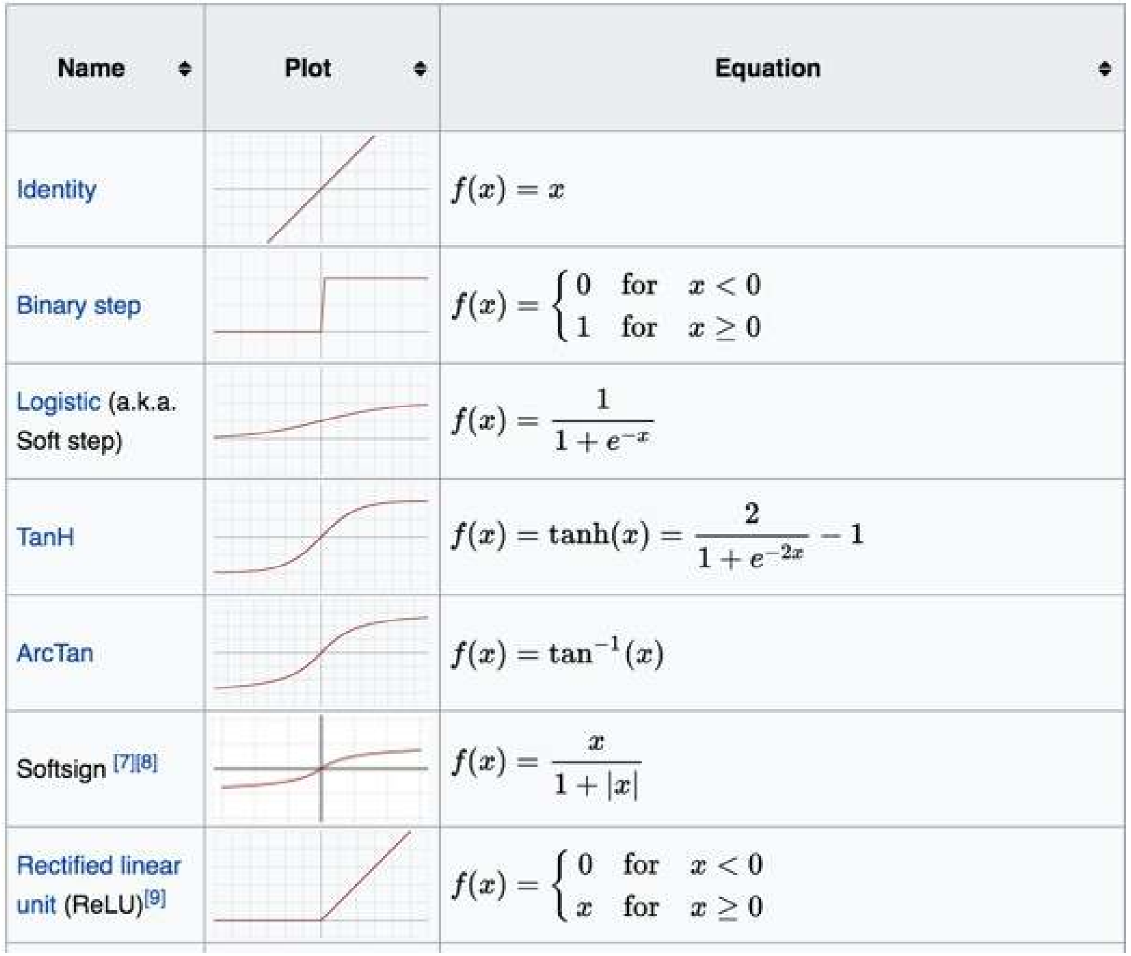
# The neuron



$$\sum_{i=1}^{l} x_i * w_i = u$$

”Multiply and Accumulate”

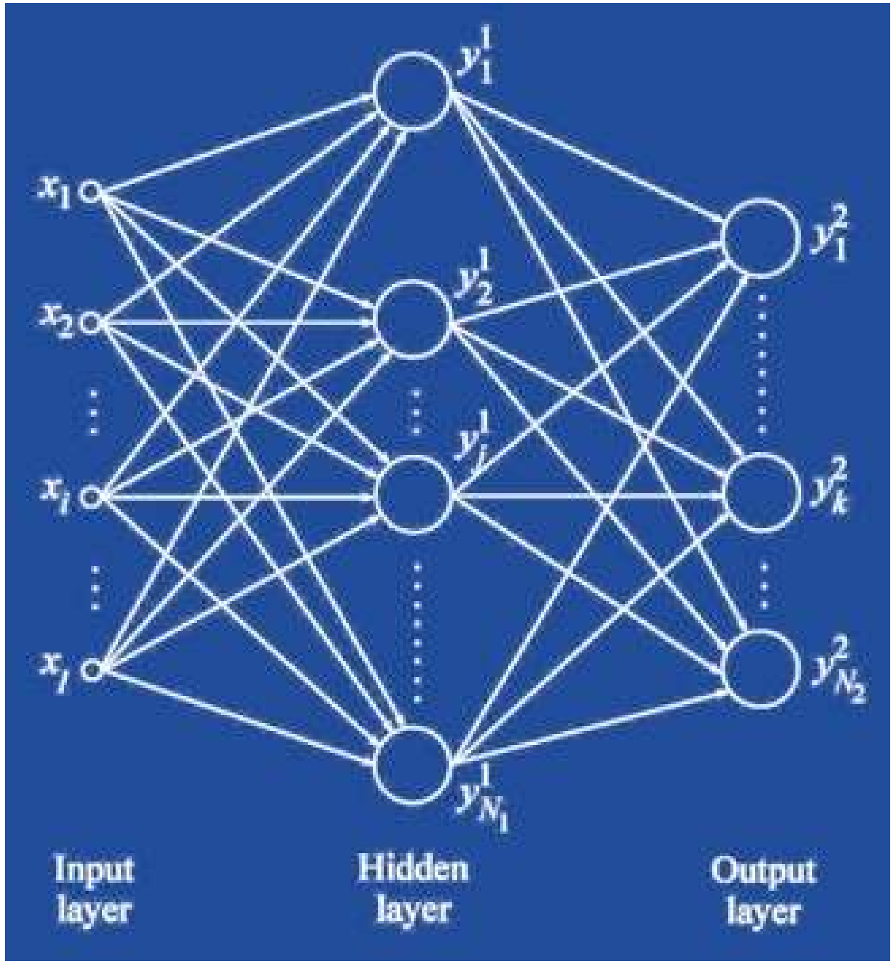
## Activation functions



| Name | Plot | Equation |
|---|---|---|
| Identity | | $f(x) = x$ |
| Binary step | | $f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$ |
| Logistic (a.k.a. Soft step) | | $f(x) = \frac{1}{1+e^{-x}}$ |
| TanH | | $f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$ |
| ArcTan | | $f(x) = \tan^{-1}(x)$ |
| Softsign [7][8] | | $f(x) = \frac{x}{1+\lvert x\rvert}$ |
| Rectified linear unit (ReLU)[9] | | $f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$ |

Source: Wikipedia

# Neural networks



l features

$$x = \left[\begin{array}{l} x_{11}, x_{12}, \dots x_{1l} \\ x_{21}, x_{22}, \dots x_{2l} \\ \dots \dots \dots \\ x_{m1}, x_{m2}, \dots x_{ml} \end{array}\right]$$

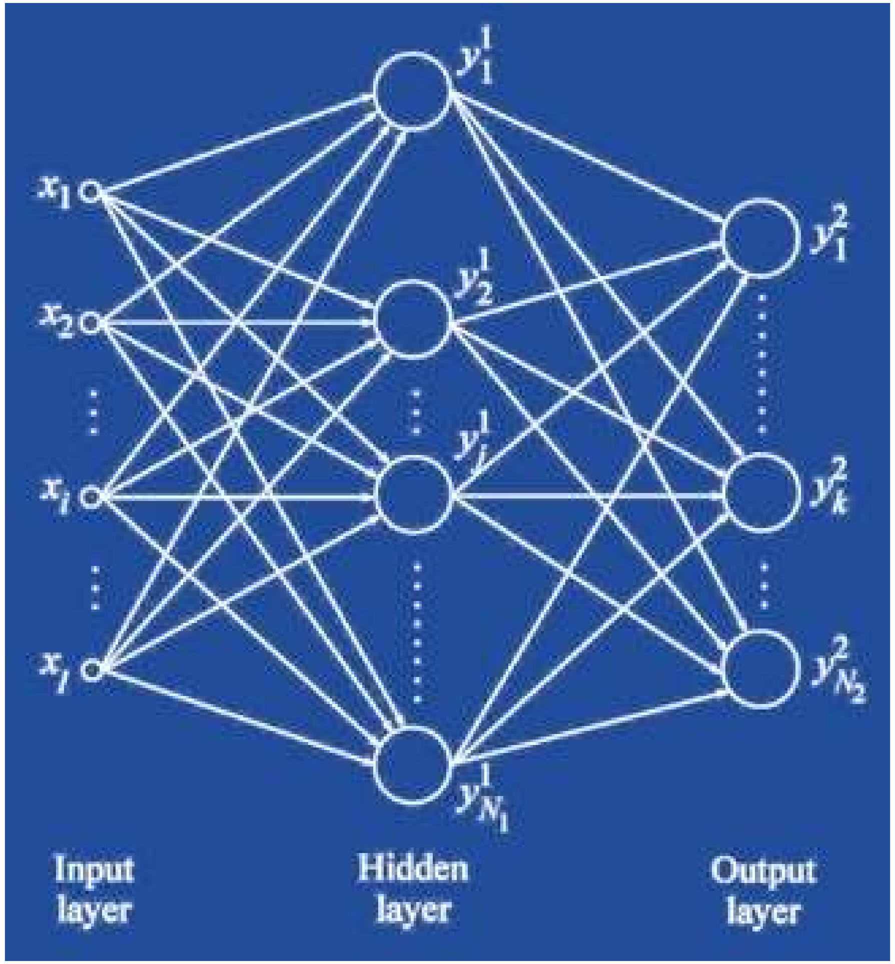
m samples

$$y = \left[\begin{array}{c} 2 \\ 0 \\ \dots \\ 4 \end{array}\right] \quad \left[\begin{array}{l} 0,0,1,0,0,\dots,0 \\ 1,0,0,0,0,\dots,0 \\ \dots \\ 0,0,0,0,1,\dots,0 \end{array}\right]$$

One-hot encoding

m labels,
$N_2$ categories

# Neural networks



l features

$$x = \begin{bmatrix} x_{11}, x \ldots x_{1l} \\ x_{21}, x \ldots x_{2l} \\ \ldots \ldots \\ x_{m1}, x \ldots \end{bmatrix}$$

$x_{11}, x_{12}, \ldots x_{1l}$

m samples

$$y = \begin{bmatrix} 2 \\ 0 \\ \ldots \\ 4 \end{bmatrix} \quad \begin{bmatrix} 0,0,1, \ldots,0 \\ 1,0,0, \ldots,0 \\ \ldots \\ 0,0,0, \ldots,0 \end{bmatrix}$$

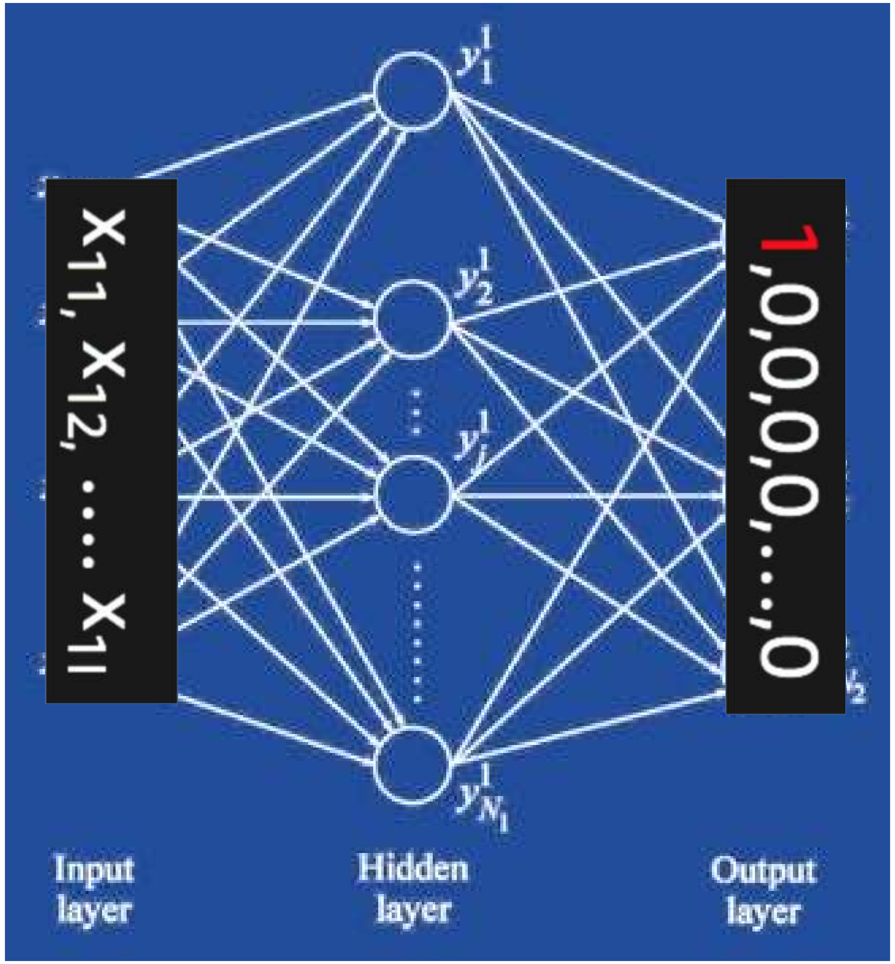
0,0,1,0,0,…,0

One-hot encoding

m labels,
$N_2$ categories

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

# Neural networks



Initially, the network will not predict correctly

$f(X_1) = Y'_1$

A loss function measures the difference between the real label $Y_1$ and the predicted label $Y'_1$

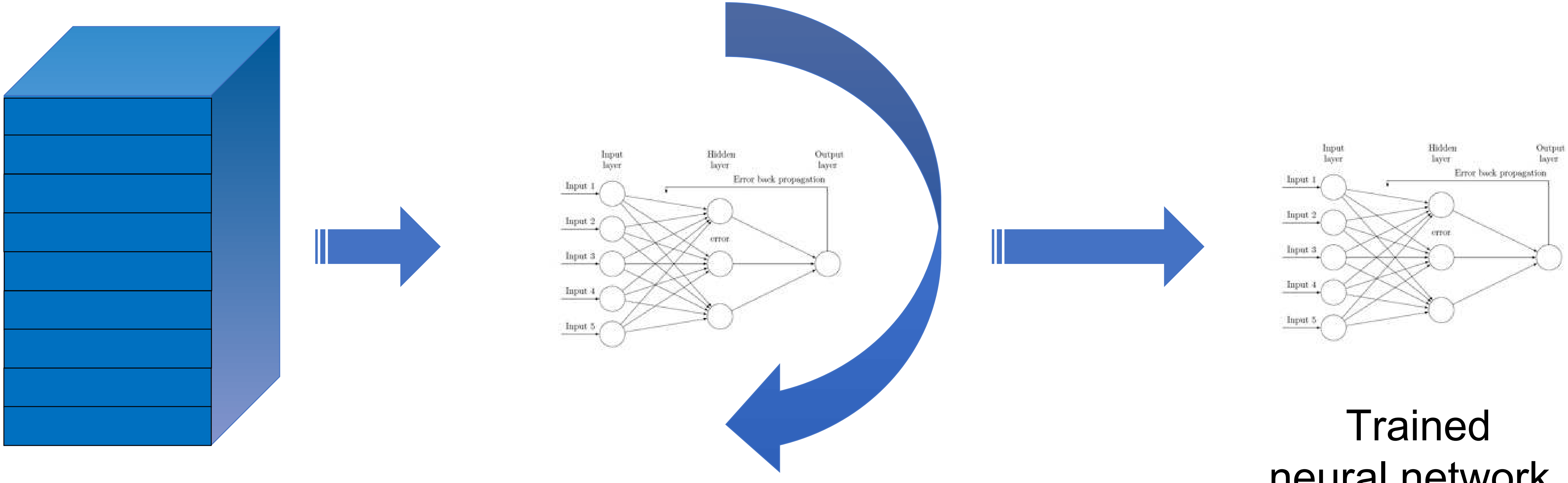
$\text{error} = \text{loss}(Y_1, Y'_1)$

For a batch of samples:

$$\sum_{i=1}^{batch\ size} \text{loss}(Y_{i,}\ Y'_i) = \text{batch error}$$

The purpose of the training process is to minimize loss by gradually adjusting weights

# Training



Trained
neural network

Training data set

Backpropagation

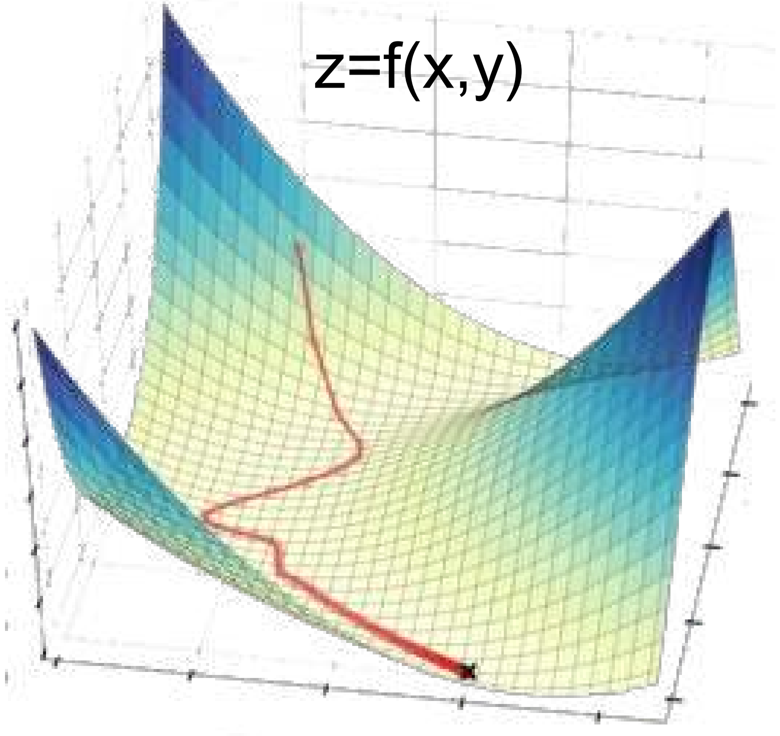
Batch size
Learning rate
Number of epochs

Hyper parameters

# Stochastic Gradient Descent (SGD)

*Imagine you stand on top of a mountain with skis strapped to your feet. You want to get down to the valley as quickly as possible, but there is fog and you can only see your immediate surroundings. How can you get down the mountain as quickly as possible? You look around and identify the steepest path down, go down that path for a bit, again look around and find the new steepest path, go down that path, and repeat—this is exactly what gradient descent does.*

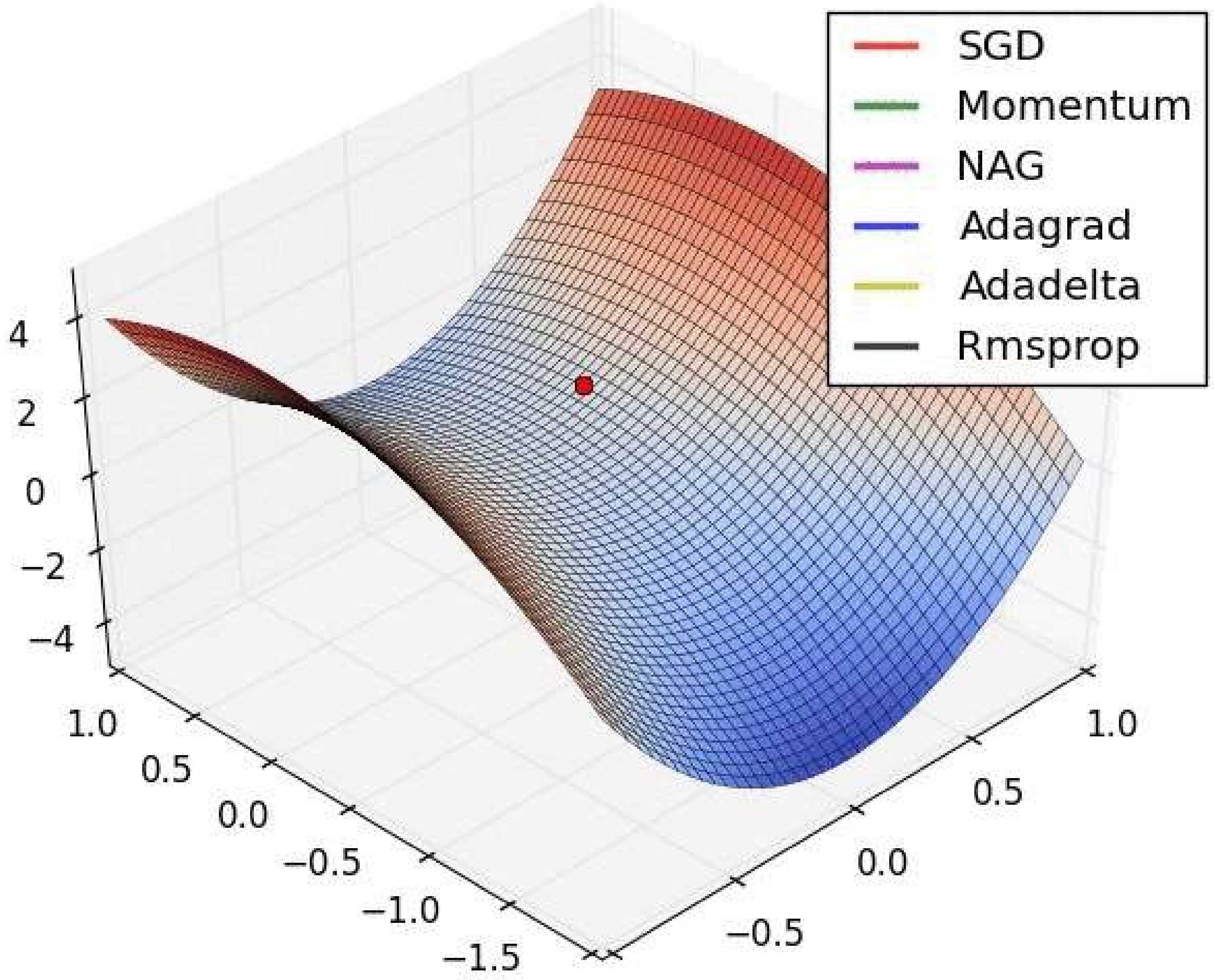
**Tim Dettmers**
University of Lugano
2015



The « step size » is called the learning rate

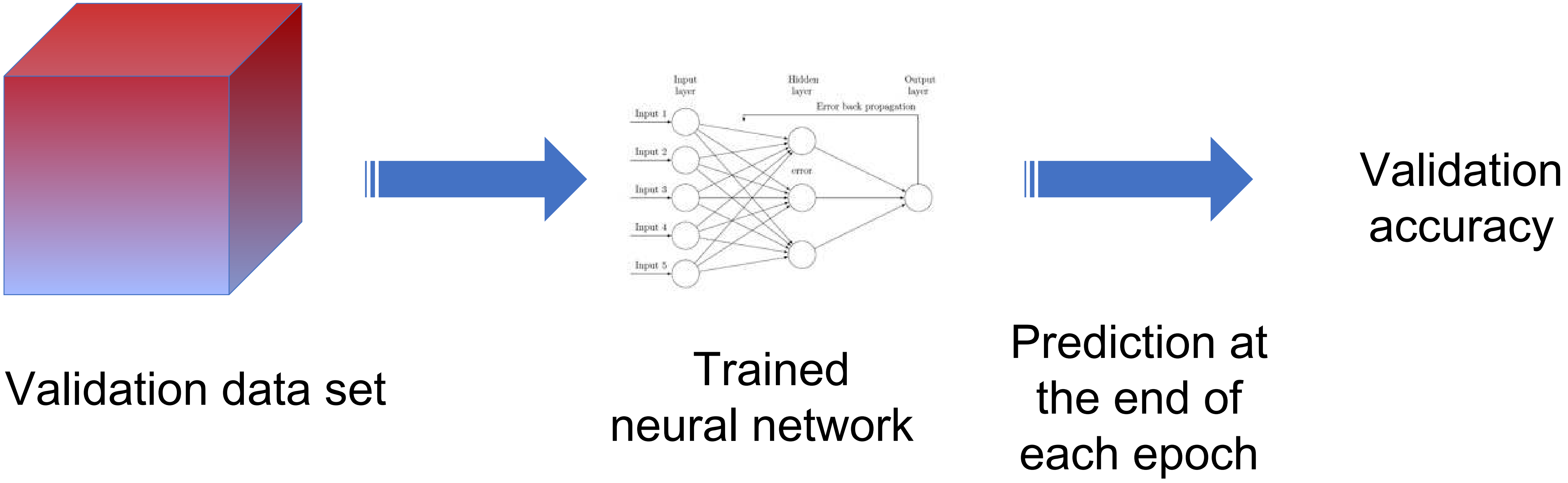
https://devblogs.nvidia.com/parallelforall/deep-learning-nutshell-history-training/

# Optimizers



https://medium.cim/@julsimon/tumbling-down-the-sgd-rabbit-hole-part-1-740fa402f0d7

# Validation



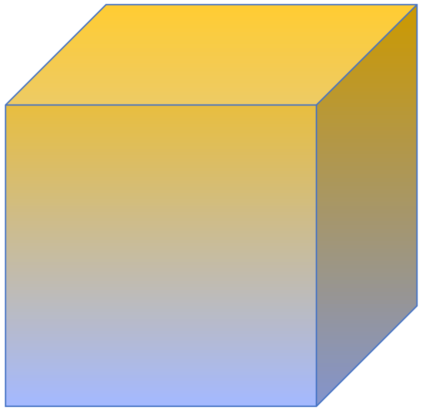
Validation data set

Trained neural network

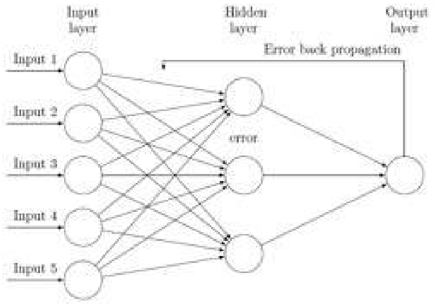
Prediction at the end of each epoch

Validation accuracy

# Test



Test data set



Fully trained neural network

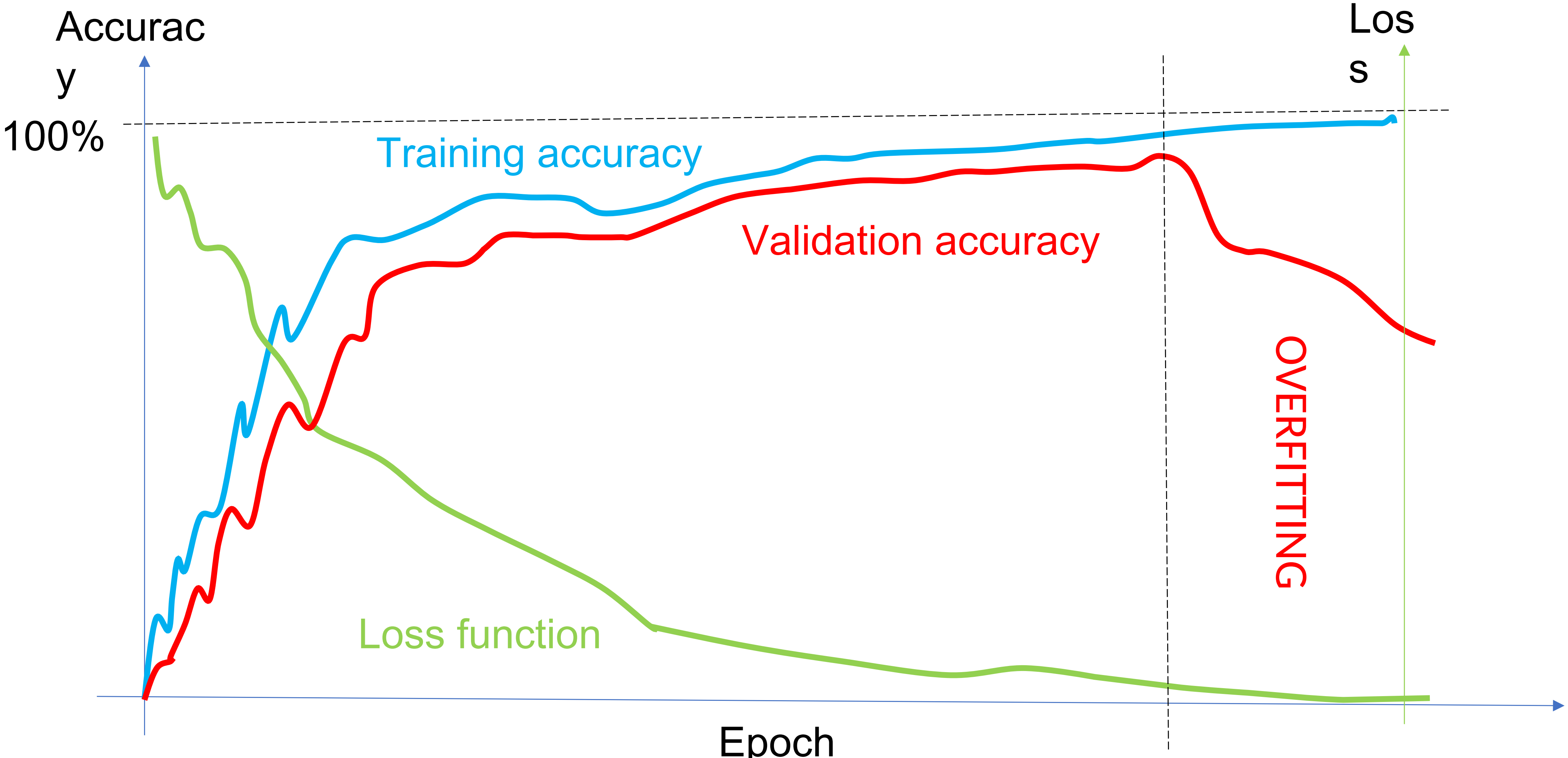
Prediction at the end of experimentation

Test accuracy

<u>This data set must have the same distribution as real-life samples, or else test accuracy won’t reflect real-life accuracy.</u>

# Early stopping

« *Deep Learning ultimately is about finding a minimum that generalizes well, with bonus points for finding one fast and reliably* », Sebastian Ruder

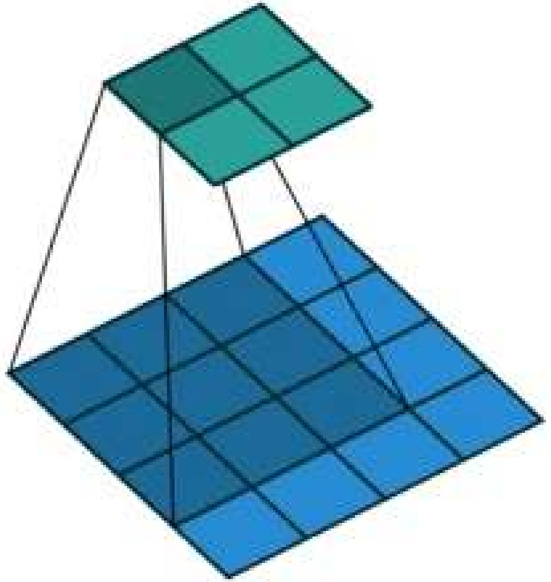


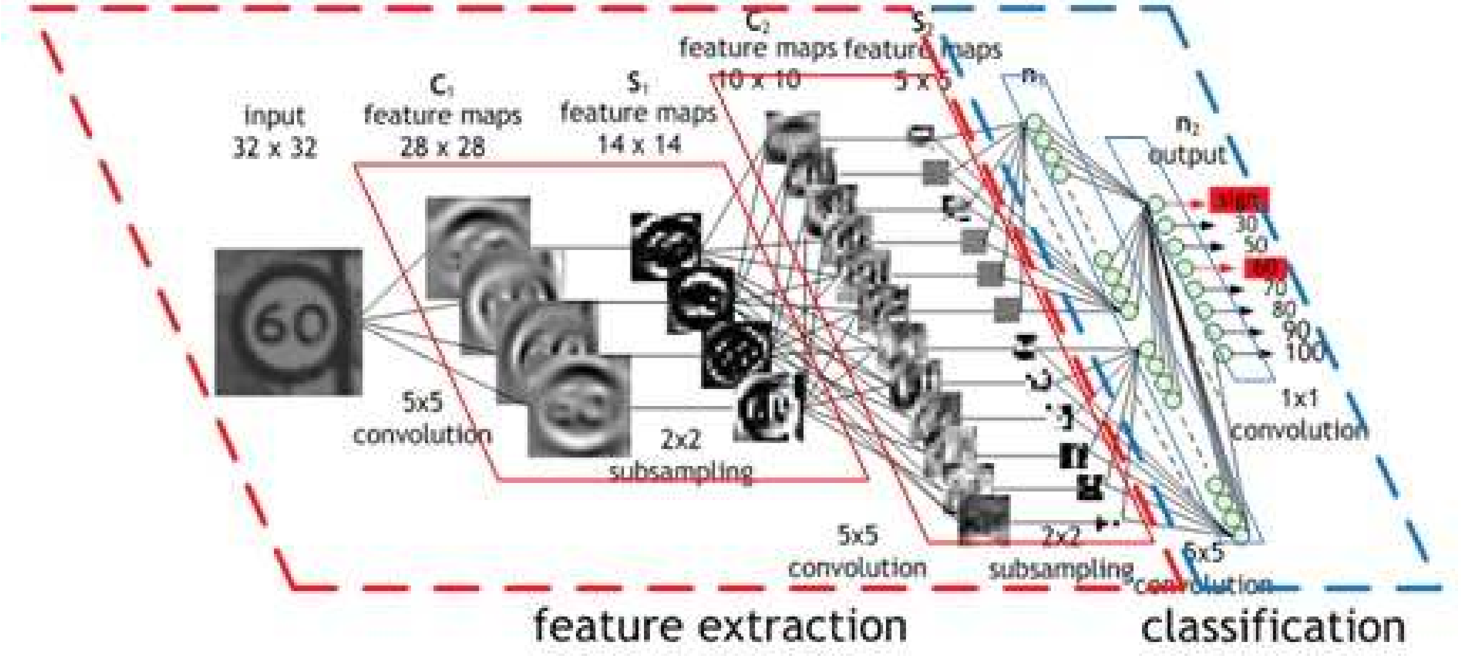
Best epoch

# Demo: fully connected network

# Convolutional Neural Networks (CNN)

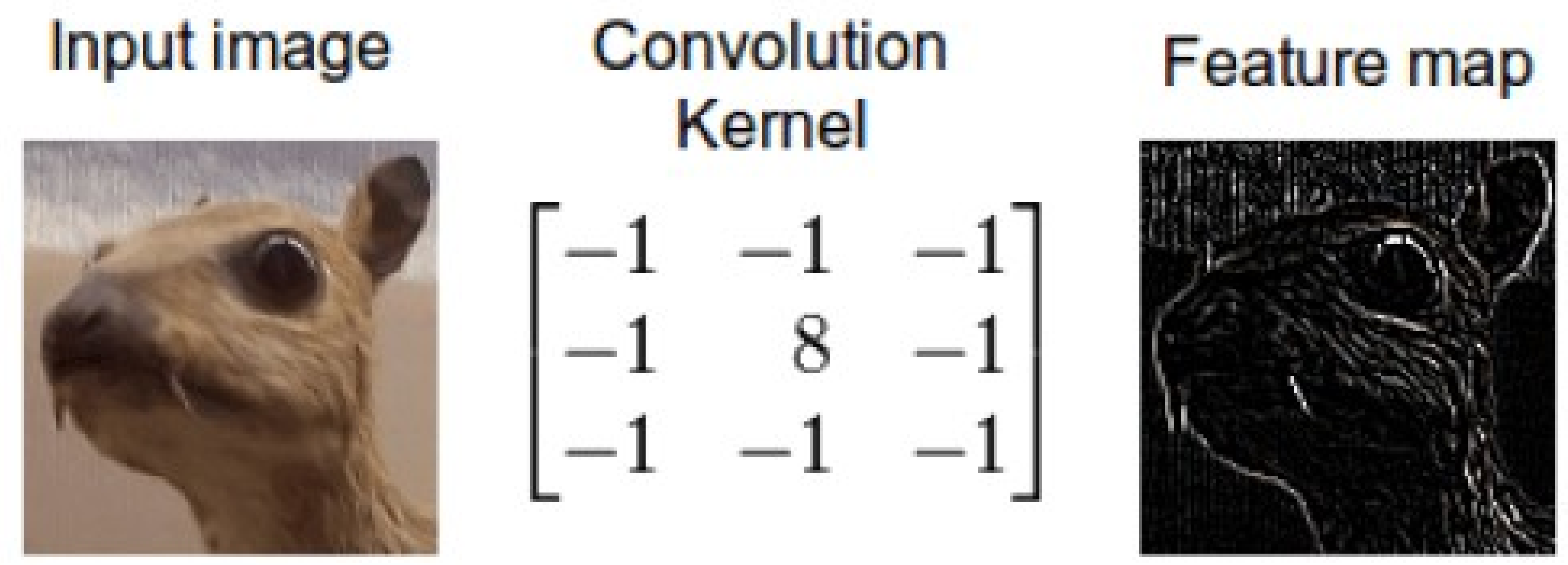
Le Cun, 1998: handwritten digit recognition, 32x32 pixels





https://devblogs.nvidia.com/parallelforall/deep-learning-nutshell-core-concepts/

# Extracting features with convolution

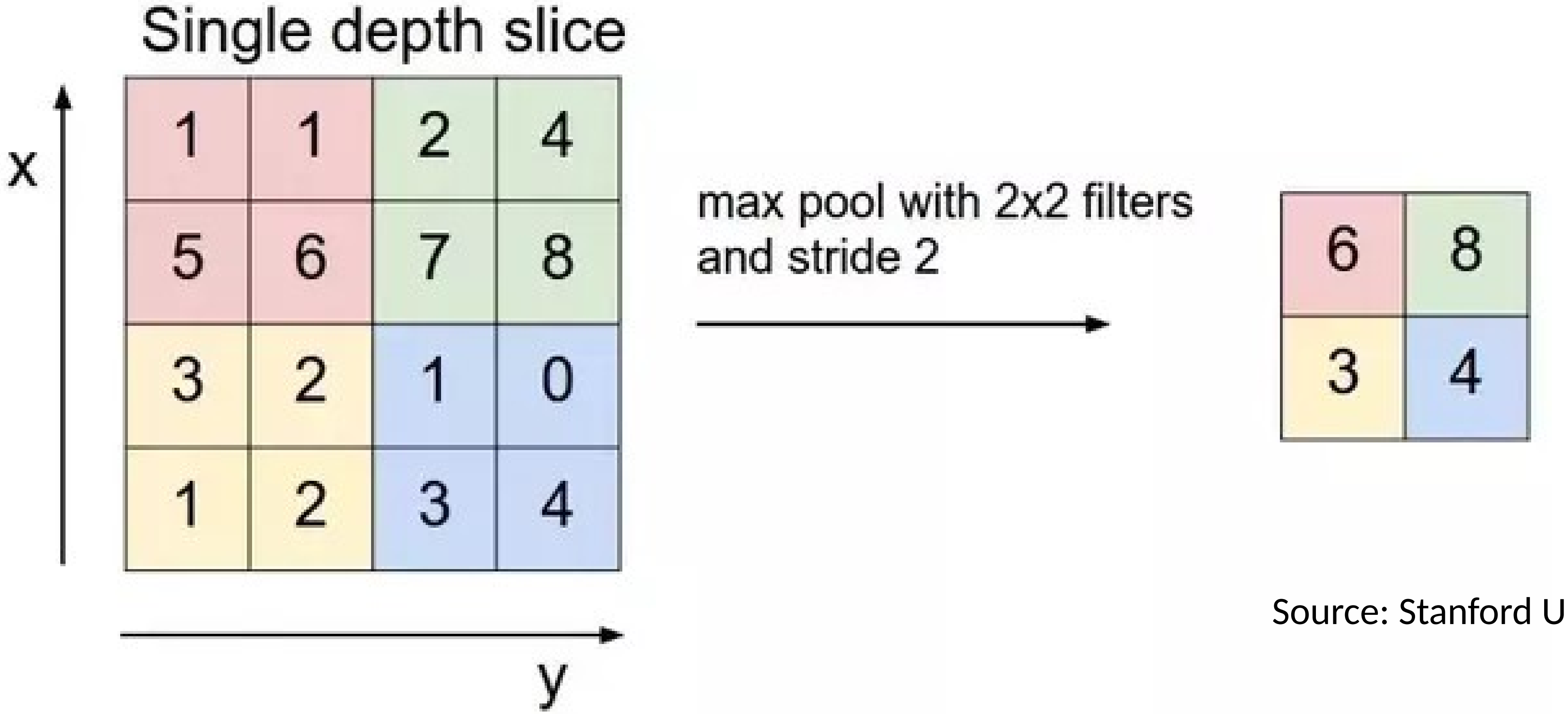


$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Source: http://timdettmers.com

Convolution extracts features automatically.
Kernel parameters are learned during the training process.

# Downsampling images with pooling



| | | | |
|---|---|---|---|
| 1 | 1 | 2 | 4 |
| 5 | 6 | 7 | 8 |
| 3 | 2 | 1 | 0 |
| 1 | 2 | 3 | 4 |

| | |
|---|---|
| 6 | 8 |
| 3 | 4 |

Source: Stanford University

Pooling shrinks images while preserving significant information.

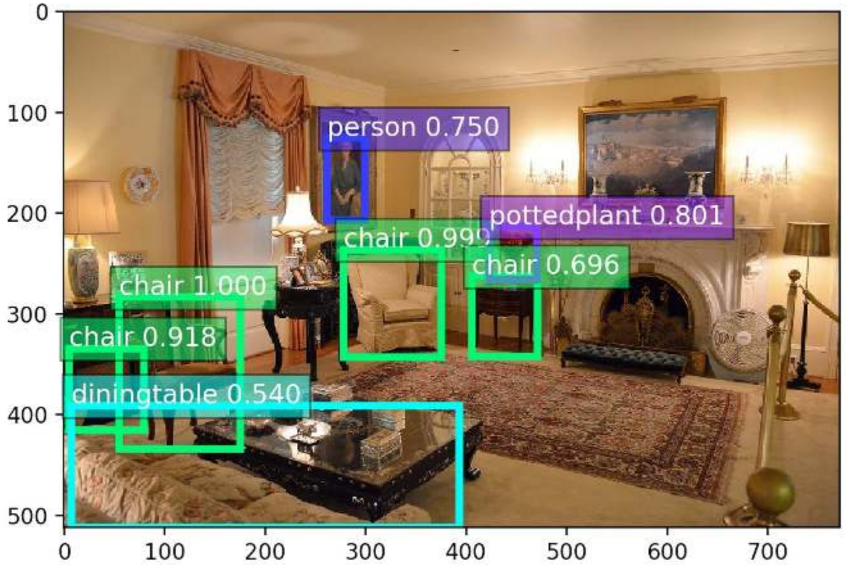
# Demo: convolutional network

# Gluon CV: state of the art pre-trained models

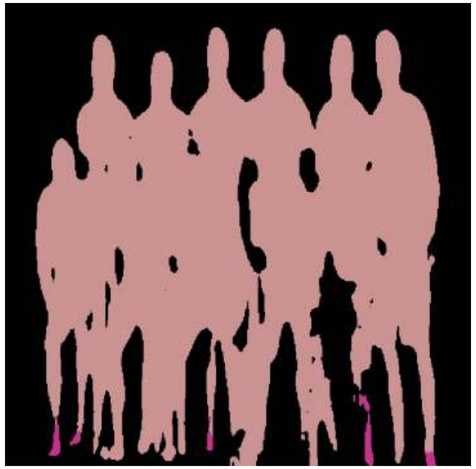
## Classification

[electric_guitar],
with probability 0.671

## Detection



## Segmentation

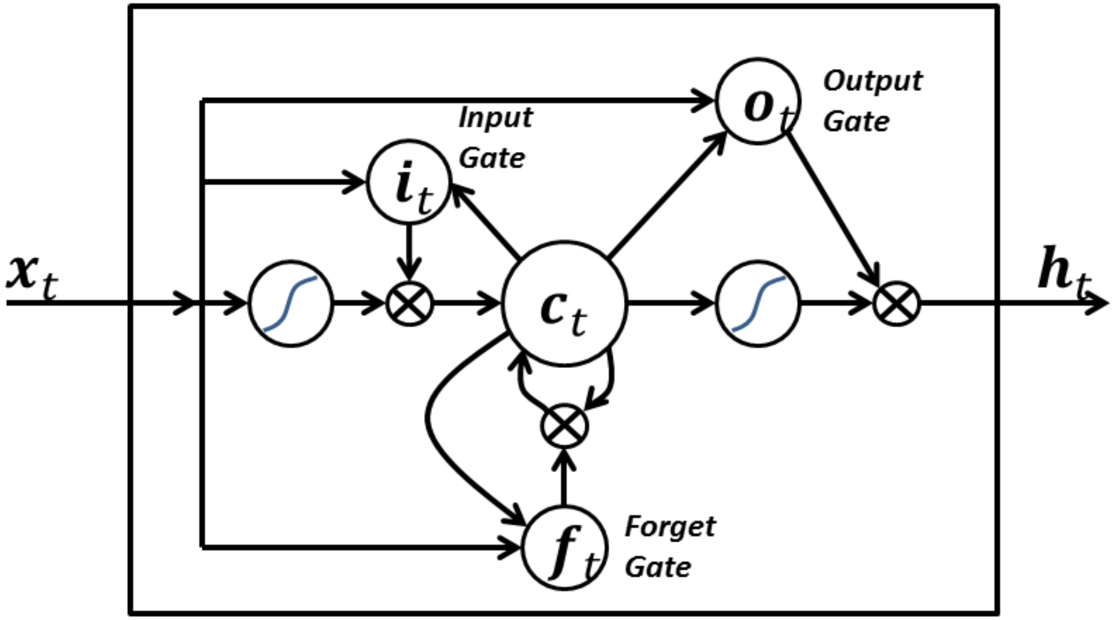


https://github.com/dmlc/gluon-cv

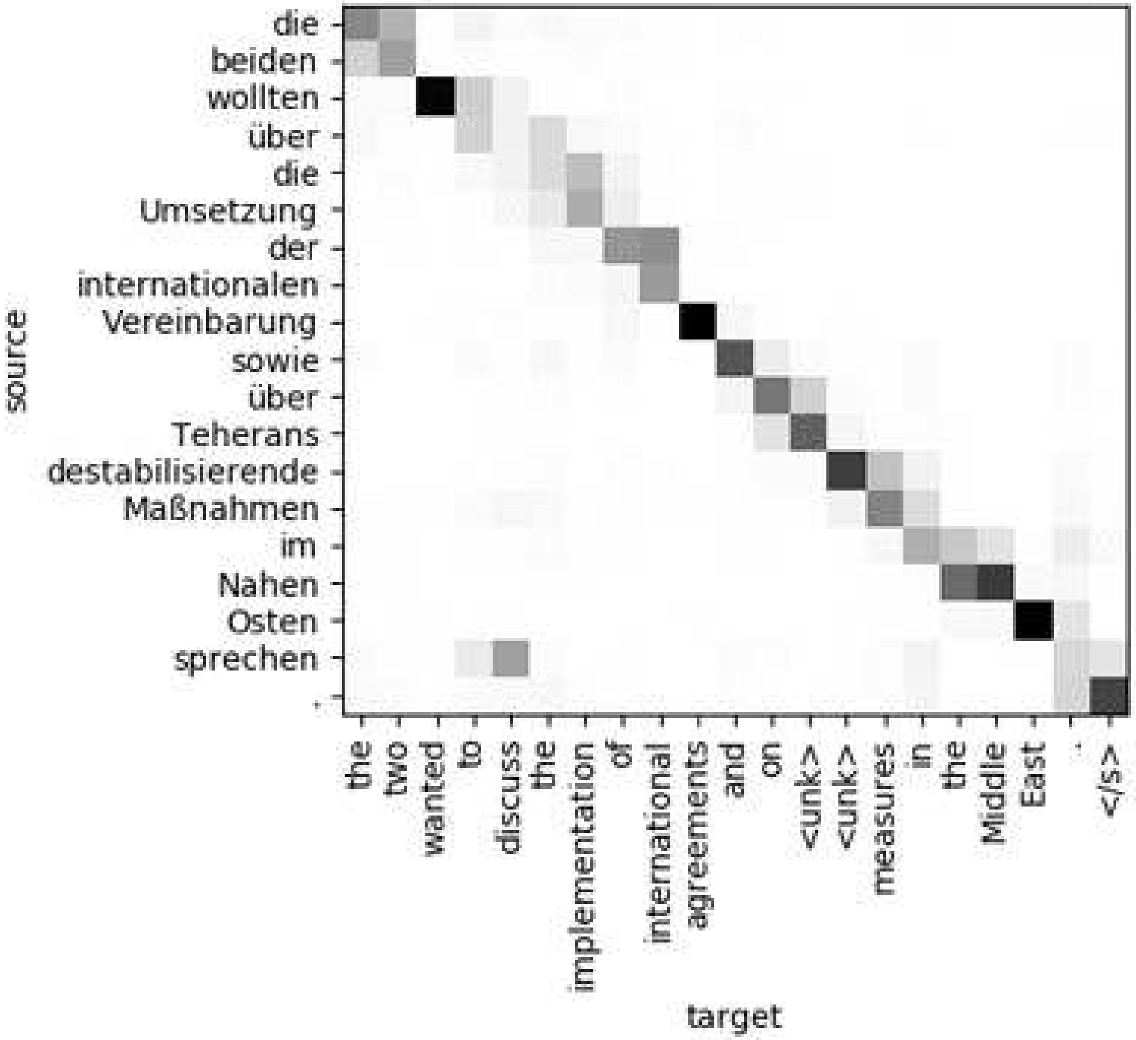
# Demo: Gluon CV

# Long Short Term Memory Networks (LSTM)

- A LSTM neuron computes the output based on the input and a previous state
- LSTM networks have memory
- They’re great at predicting sequences, e.g. machine translation



# Machine Translation

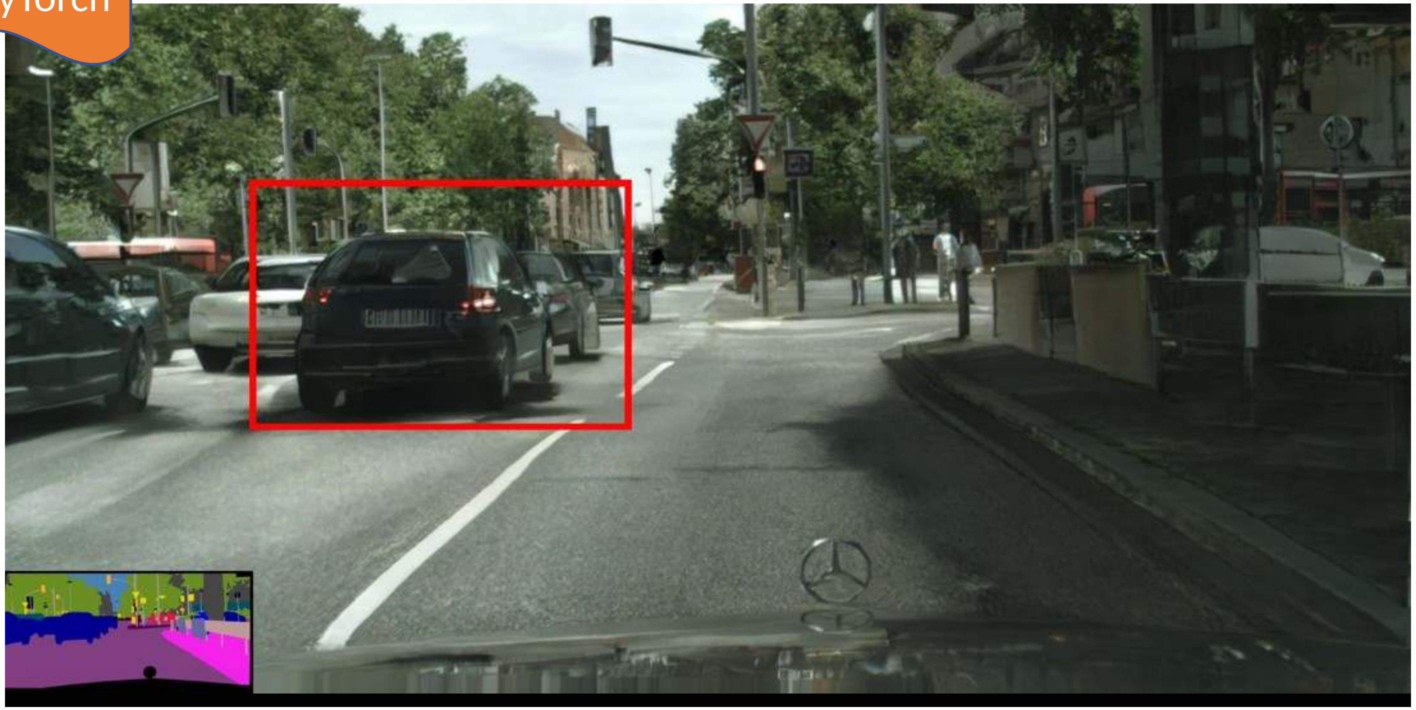


https://github.com/awslabs/sockeye

# GAN: Welcome to the (un)real world, Neo

TF

Generating new ”celebrity” faceshttps://github.com/tkarras/progressive_growing_of_gans



PyTorch

From semantic map to 2048x1024 picture
https://tcwang0509.github.io/pix2pixHD/

# Scalable training on AWS

## Amazon SageMaker

Pre-built notebooks for common problems

Built-in, high-performance algorithms

Build

One-click training

Hyperparameter optimization

Train

One-click deployment

Fully managed hosting with auto-scaling

Deploy

## AWS Deep Learning AMI

TensorFlow

K

Microsoft CNTK

mxnet

GLUON

PYTORCH

Chainer

Caffe2

torch

## Amazon EC2

c5

p3

# Getting started

https://ml.aws | https://aws.amazon.com/blogs/machine-learning/

https://mxnet.incubator.apache.org | https://github.com/apache/incubator-mxnet

https://gluon.mxnet.io | https://github.com/gluon-api | https://github.com/dmlc/gluon-cv

https://aws.amazon.com/sagemaker

https://github.com/awslabs/amazon-sagemaker-examples

https://github.com/aws/sagemaker-python-sdk | https://github.com/aws/sagemaker-spark

https://medium.com/@julsimon

https://youtube.com/juliensimonfr

https://gitlab.com/juliensimon/dlnotebooks

# Thank you!

Julien Simon
Principal Technical Evangelist, AI & Machine Learning
@julsimon