AWS INNOVATE 2018 ONLINE CONFERENCE

DEVELOPER EDITION

Introduction to Deep Learning Theory, Use Cases, and Tools

Julio Faerman @faermanj

Julien Simon @julsimon

@awscloud

#AWSInnovate

Myth: AI is dark magic
aka "you're not smart enough"

aws

What to expect

1. An introduction to Deep Learning
2. Common network architectures and use cases
3. Resources for diving deeper

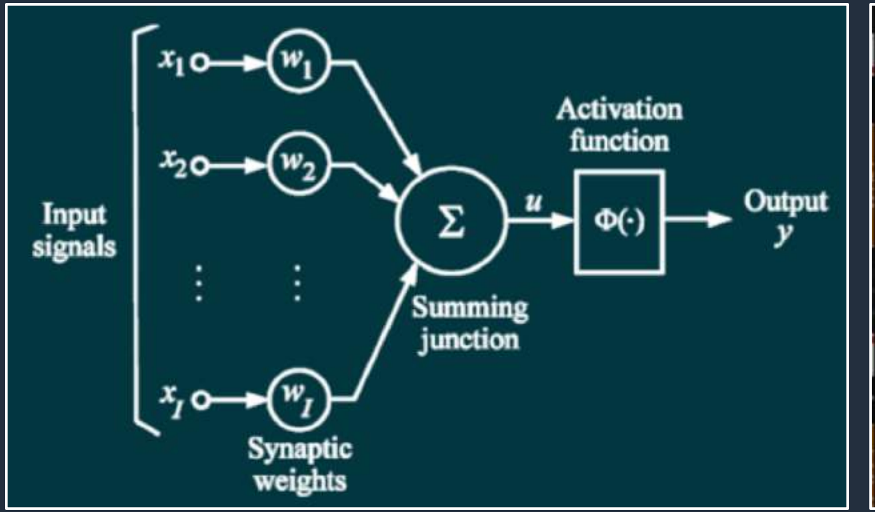
Artificial Intelligence: design software applications which exhibit human-like behavior, e.g. speech, natural language processing, reasoning or intuition

Machine Learning: teach machines to learn without being explicitly programmed

Deep Learning: using neural networks, teach machines to learn from complex data where features cannot be explicitly expressed

Fact: AI is math, code and chips

A bit of science, a lot of engineering




```
data = mx.symbol.Variable('data')
conv1 = mx.sym.Convolution(data=data, kernel=(5,5), num_filter=20)
relu1 = mx.sym.Activation(data=conv1, act_type="relu")
pool1 = mx.sym.Pooling(data=relu1, pool_type="max", kernel=(2,2), stride=(2,2))
conv2 = mx.sym.Convolution(data=pool1, kernel=(5,5), num_filter=50)
relu2 = mx.sym.Activation(data=conv2, act_type="relu")
pool2 = mx.sym.Pooling(data=relu2, pool_type="max", kernel=(2,2), stride=(2,2))
flatten = mx.sym.Flatten(data=pool2)
fc1 = mx.symbol.FullyConnected(data=flatten, num_hidden=500)
relu3 = mx.sym.Activation(data=fc1, act_type="relu")
fc2 = mx.sym.FullyConnected(data=relu3, num_hidden=10)
lenet = mx.sym.SoftmaxOutput(data=fc2, name='softmax')
```

ML / DL hardware ?

Google TPU, Intel Nervana, Xilinx Everest

Quantization: using integer or binary weights and activations

- Reduces power consumption
- Simplifies the logic needed to implement the model
- Reduces memory usage

Pruning: removing useless connections

- Increases computation speed
- Reduces memory usage

Compression: encoding weights

- Reduces model size

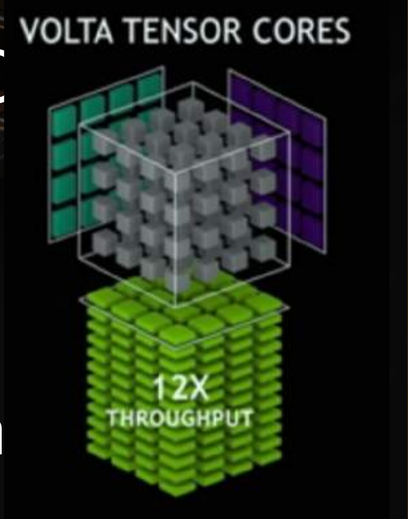
NVIDIA Tesla Volta V100 GPUs

"At the limit of photolithography" - Jen-Hsun Huang, Nvidia's CEC

5K FP32 GPU Cores | 15.7 Peak FP32 TFLOPS

640 Tensor Cores, 64 FMA/clock, up to 125 Tensor Core TFLOPS

16 GB GPU memory with 900 GB/sec peak 8xGPU memory bandwidth



Netflix Technology Blog Follow

Learn more about how Netflix designs, builds, and operates our systems and engineering organizations

Feb 10, 2014 · 11 min read

Distributed Neural Networks with GPUs in the AWS Cloud

by Alex Chen, Justin Basilico, and Xavier Amatriain

As we have described previously on this blog, at Netflix we are constantly innovating by looking for better ways to find the best movies and TV shows for our members. When a new algorithmic technique such as Deep Learning shows promising results in other domains (e.g. Image Recognition, Neuro-imaging, Language Models, and Speech Recognition), it should not come as a surprise that we would try to figure out how to apply such techniques to improve our product. In this post, we will focus on what we have learned

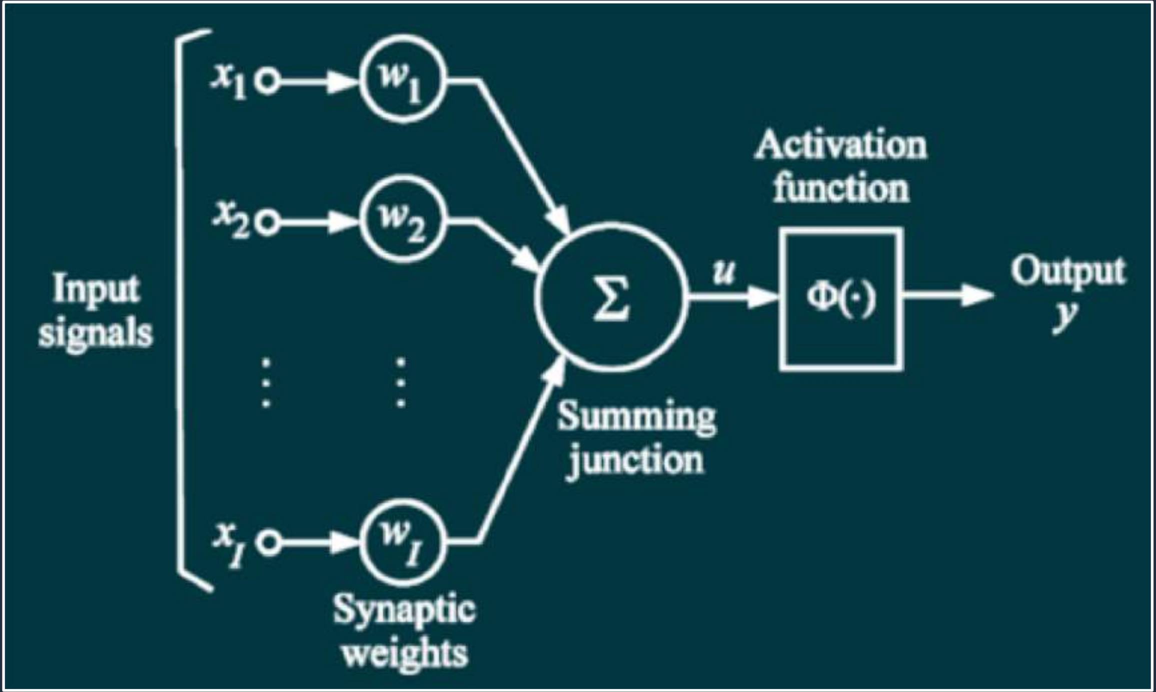
ML @ AWS

Put machine learning in the hands of every developer and data scientist

INNOVATE 2018
ONLINE CONFERENCE
DEVELOPER EDITION

An introduction to Deep Learning

The neuron



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

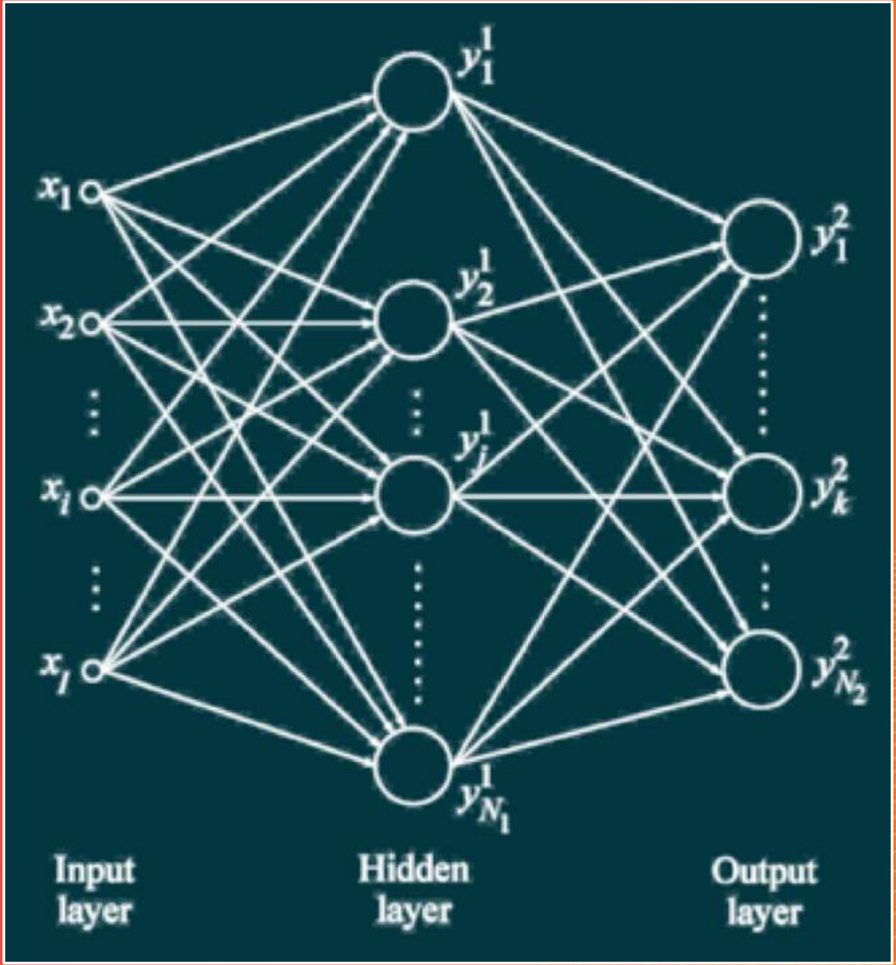
"Multiply and Accumulate"

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

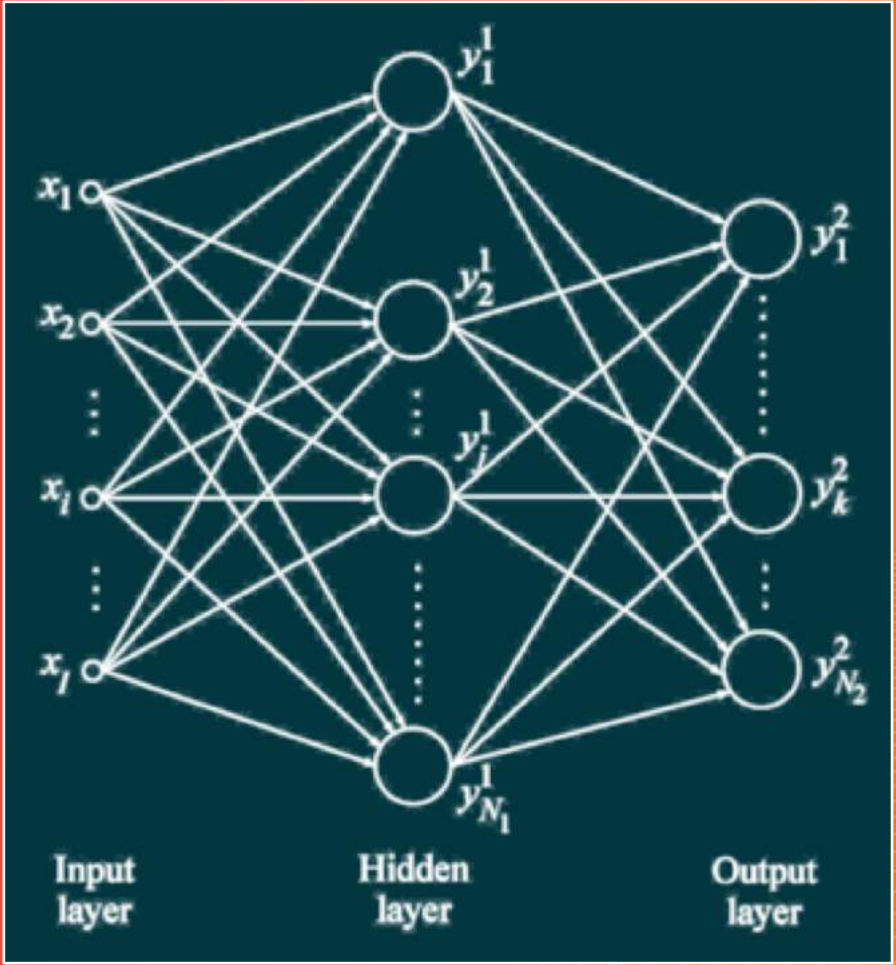


$$x = \left.\begin{bmatrix} \overbrace{x_{11}, x_{12}, \ldots x_{1l}}^{l \text{ features}} \\ x_{21}, x_{22}, \ldots x_{2l} \\ \ldots \ldots \ldots \\ x_{m1}, x_{m2}, \ldots x_{ml} \end{bmatrix}\right\} m \text{ samples}$$

$$y = \begin{bmatrix} 2 \\ 0 \\ \ldots \\ 4 \end{bmatrix} \quad \left.\begin{bmatrix} 0,0,1,0,0,\ldots,0 \\ 1,0,0,0,0,\ldots,0 \\ \ldots \\ 0,0,0,0,1,\ldots,0 \end{bmatrix}\right\} m \text{ labels, } N_2 \text{ categories}$$

One-hot encoding

Neural networks



I features

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

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

m samples

$0,0,1,0,0,\ldots,0$

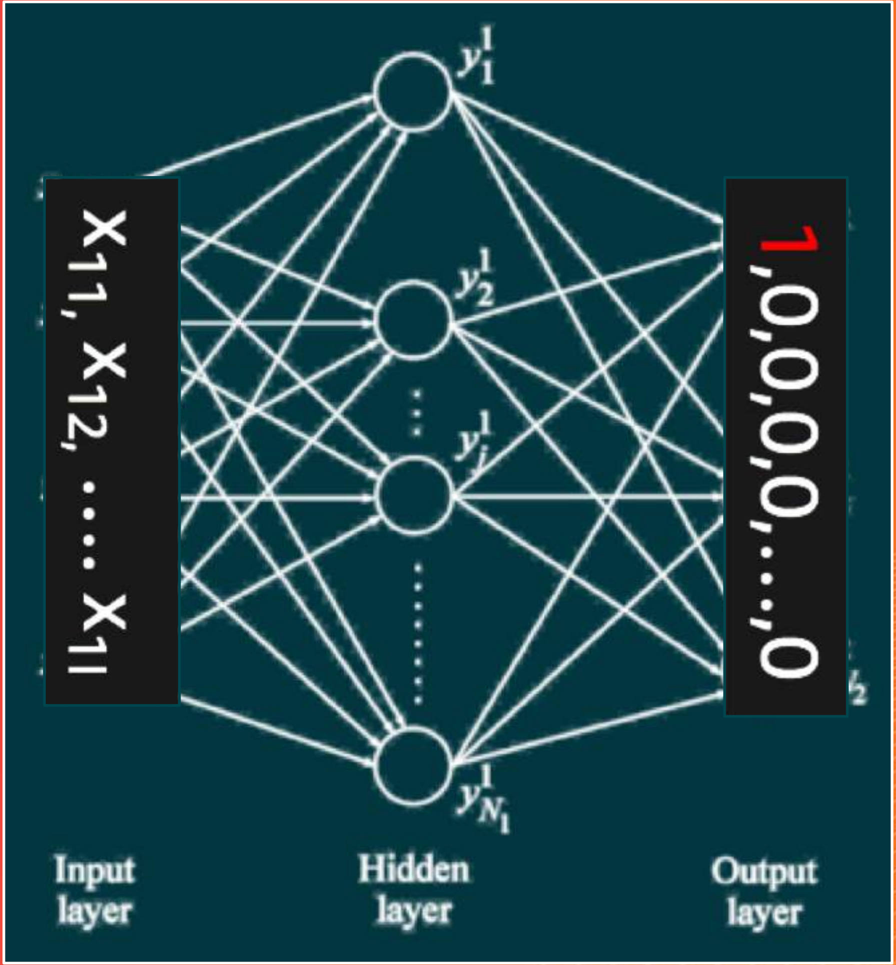
$$y = \begin{bmatrix} 2 \\ 0 \\ \ldots \\ 4 \end{bmatrix} \qquad \begin{bmatrix} 0,0,1, & \ldots,0 \\ 1,0,0, & \ldots,0 \\ \ldots & \\ 0,0,0, & \ldots,0 \end{bmatrix}$$

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

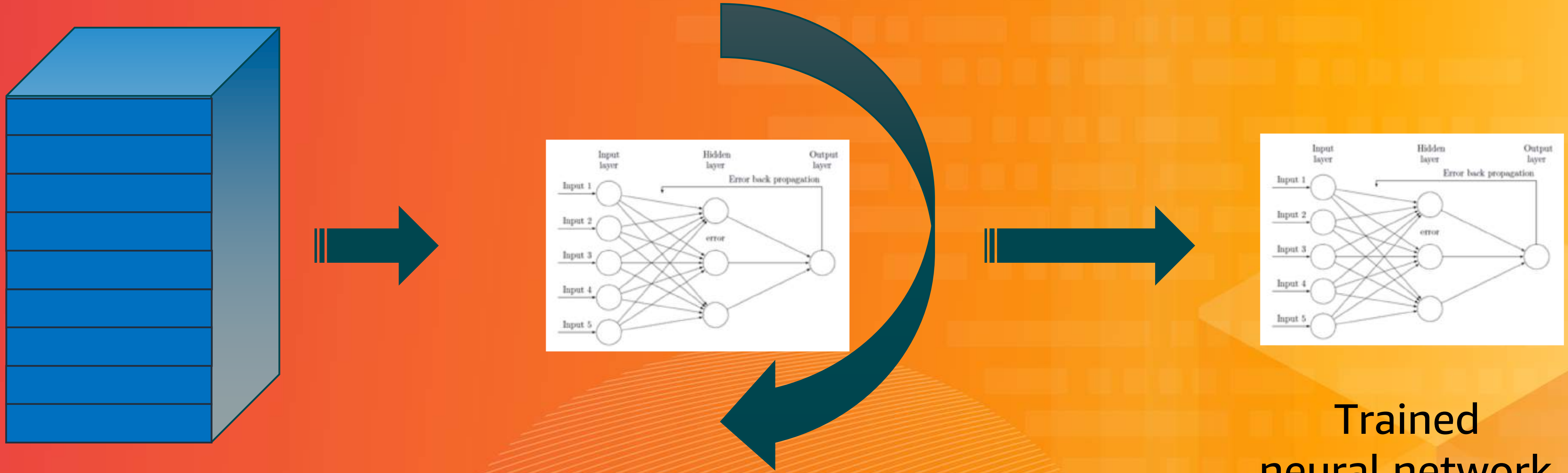
$error = loss(Y_1, Y'_1)$

For a batch of samples:

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

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

Training



Training data set

Backpropagation

Trained
neural network

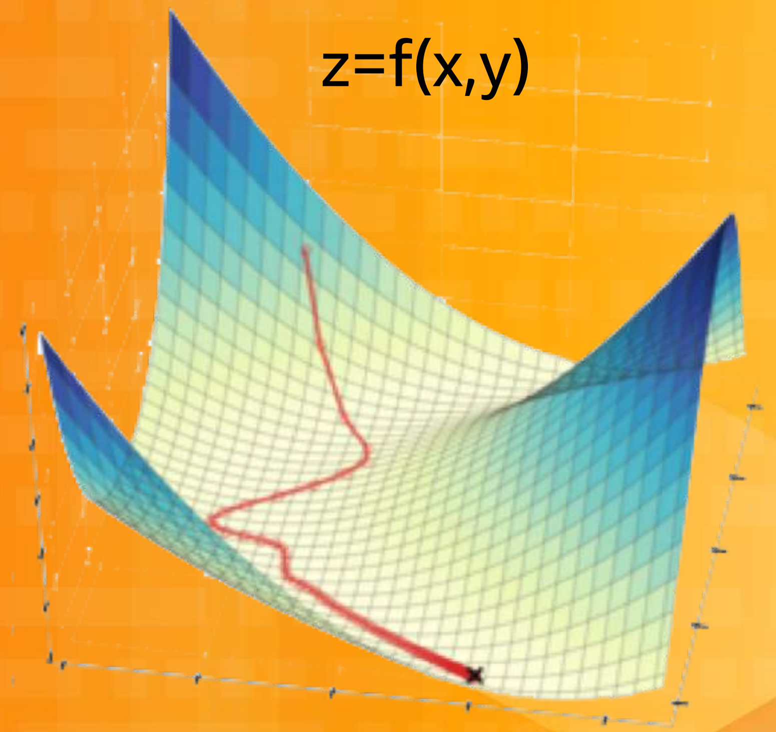
Batch size
Learning rate
Number of epochs

Hyper parameters

Stochastic Gradient Descent (1951)

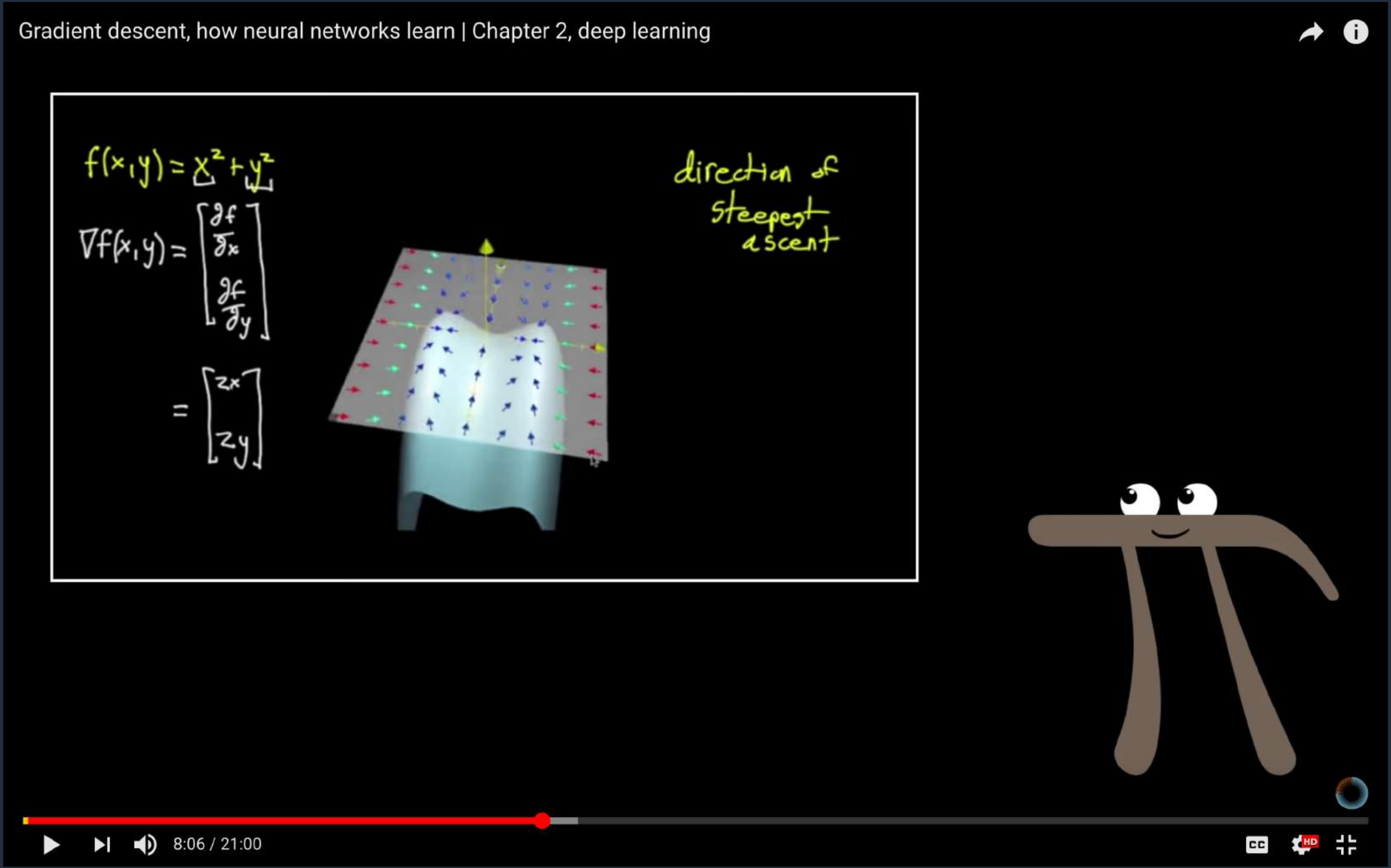
Imagine you stand on top of a mountain with skis strapped to your feet. You want to get down to the valley as quickly as possible, but there is fog and you can only see your immediate surroundings. How can you get down the mountain as quickly as possible? You look around and identify the steepest path 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 » depends on the learning rate

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


Gradient descent, how neural networks learn | Chapter 2, deep learning
$f(x,y) = x^2 + y^2$
$\nabla f(x,y) = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \begin{bmatrix} 2x \\ 2y \end{bmatrix}$
direction of steepest ascent
8:06 / 21:00

Error derivative

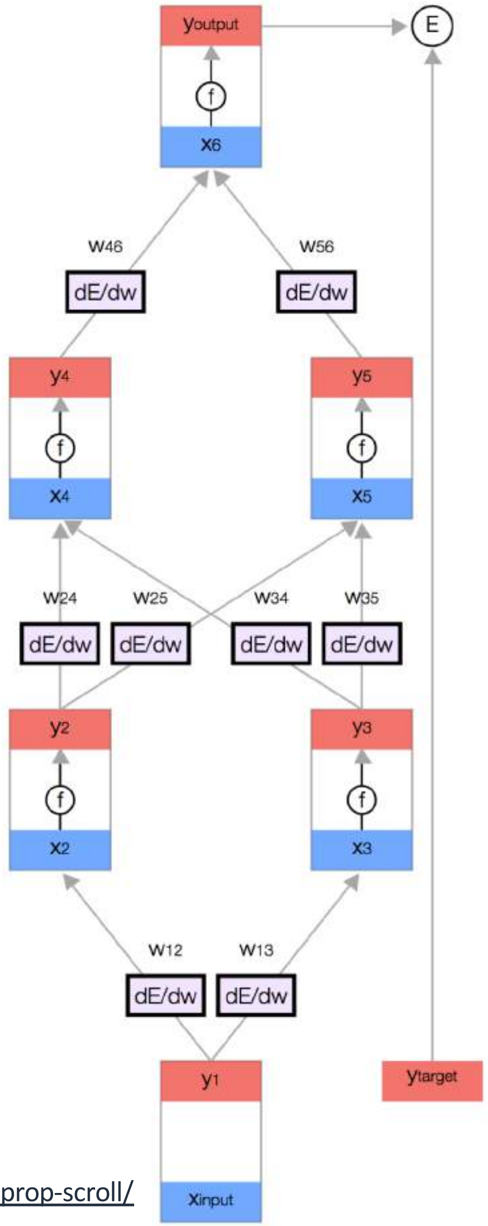
The backpropagation algorithm decides how much to update each weight of the network after comparing the predicted output with the desired output for a particular example. For this, we need to compute how the error changes with respect to each weight $\frac{dE}{dw_{ij}}$.

Once we have the error derivatives, we can update the weights using a simple update rule:

$$w_{ij} = w_{ij} - \alpha \frac{dE}{dw_{ij}}$$

where α is a positive constant, referred to as the learning rate, which we need to fine-tune empirically.

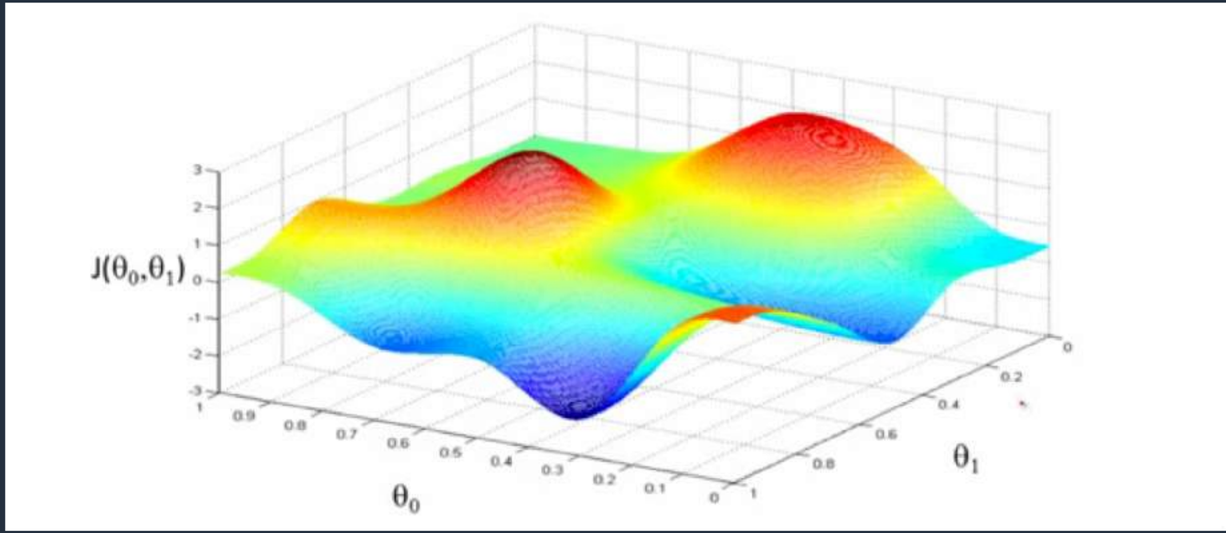
[Note] The update rule is very simple: if the error goes down when the weight increases ($\frac{dE}{dw_{ij}} < 0$), then increase the weight, otherwise if the error goes up when the weight increases ($\frac{dE}{dw_{ij}} > 0$), then decrease the weight.

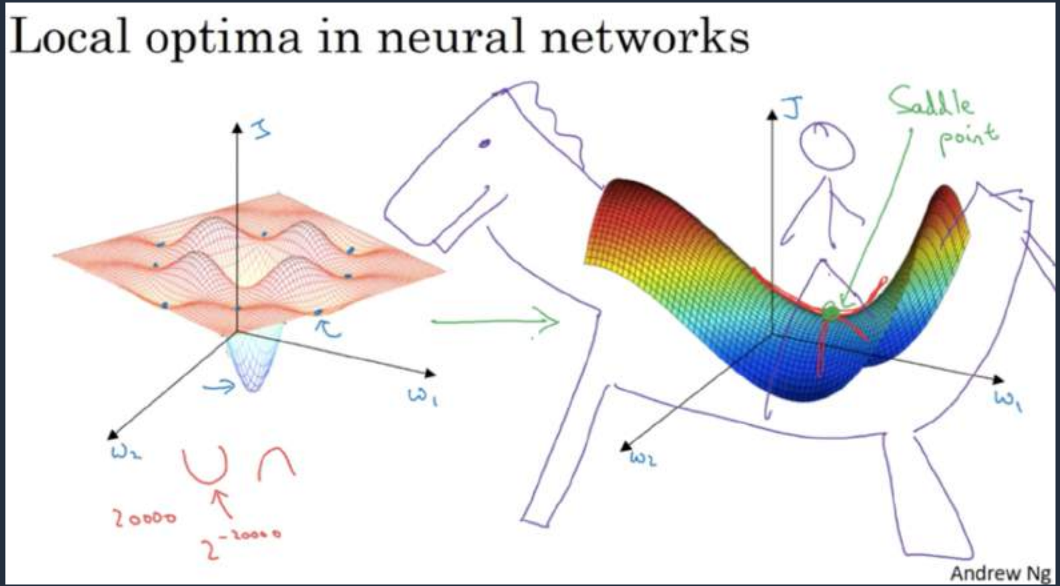


https://google-developers.appspot.com/machine-learning/crash-course/backprop-scroll/

$$\vec{\mathbf{W}} = \begin{bmatrix} 2.25 \\ -1.57 \\ 1.98 \\ \vdots \\ -1.16 \\ 3.82 \\ 1.21 \end{bmatrix} \quad 0.18 \qquad -\nabla C(\vec{\mathbf{W}}) = \begin{bmatrix} 0.18 \\ 0.45 \\ -0.51 \\ \vdots \\ 0.40 \\ -0.32 \\ 0.82 \end{bmatrix}$$

Local minima and saddle points





« Do neural networks enter and escape a series of local minima? Do they move at varying speed as they approach and then pass a variety of saddle points? Answering these questions definitively is difficult, but we present evidence strongly suggesting that the answer to all of these questions is no. »

« Qualitatively characterizing neural network optimization problems », Goodfellow et al, 2015
https://arxiv.org/abs/1412.6544

Cross Validated

QUESTIONS TAGS USERS BADGES UNANSWERED ASK QUESTION

In neural nets, why use gradient methods rather than other metaheuristics?

For large-size networks, most local minima are equivalent and yield similar performance on a test set.

The probability of finding a "bad" local minimum decreases quickly with networks size.

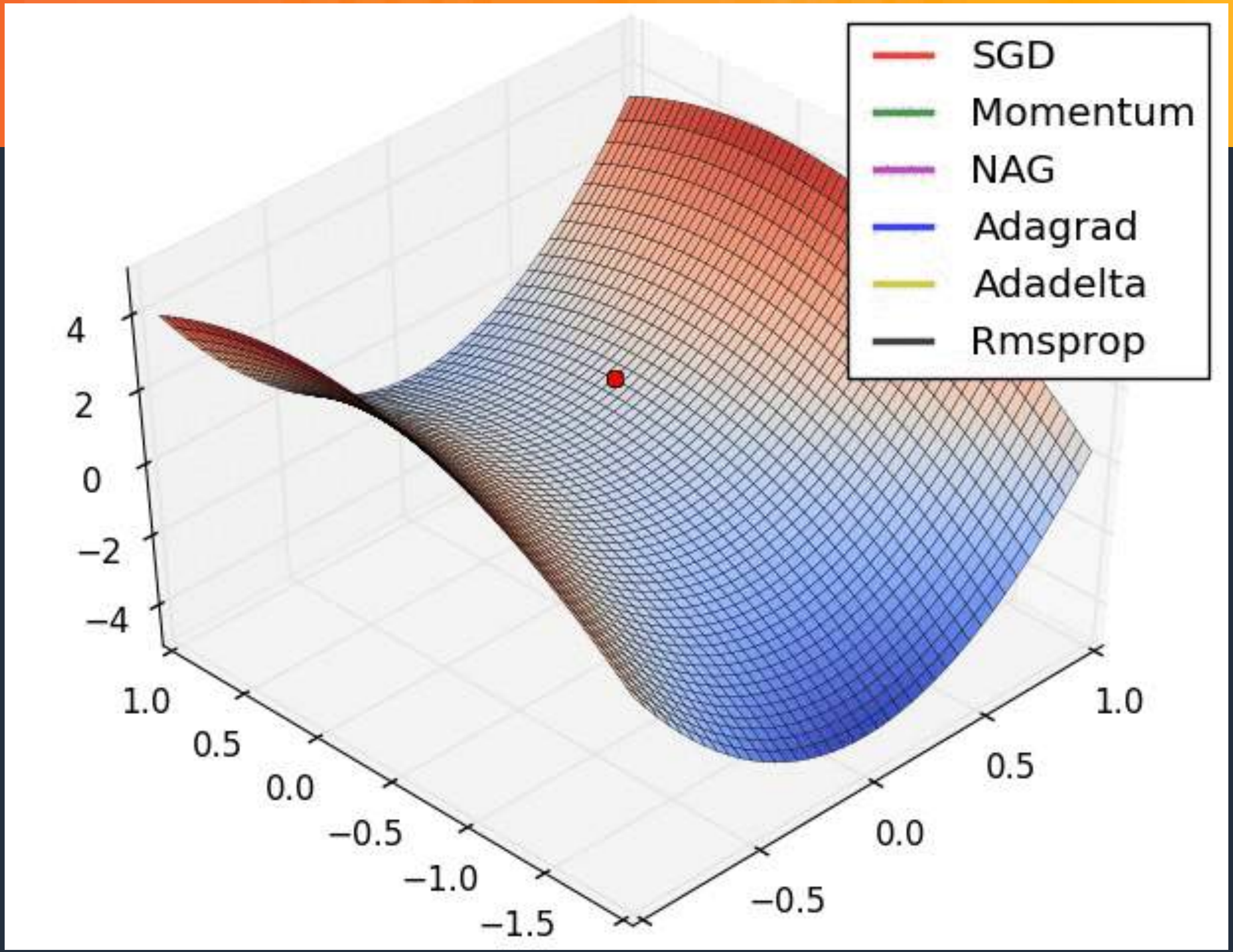
Struggling to find the global minimum is not so useful in practice and may lead to overfitting.

Optimizers

SGD works remarkably well and is still widely used.

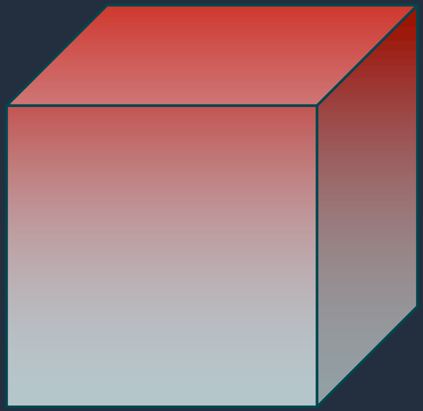
Adaptative optimizers use a variable learning rate.

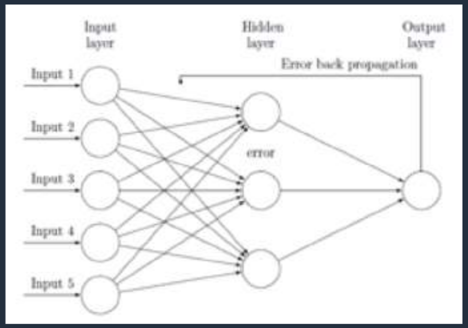
Some even use a learning rate per dimension (Adam).



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

Validation





Validation accuracy

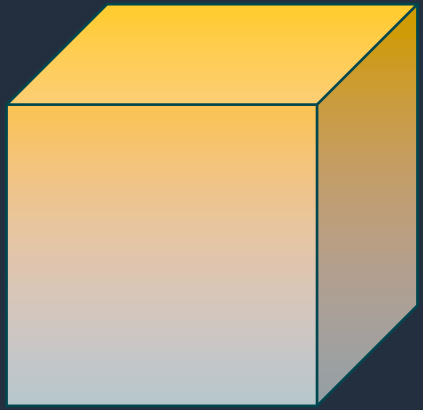
Validation data set
(also called dev set)

Neural network
in training

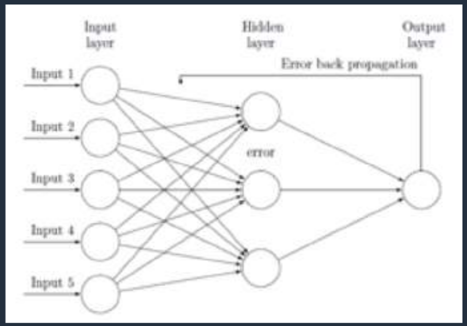
Prediction at
the end of
each epoch

This data set must have the same distribution as real-life samples,
or else validation accuracy won't reflect real-life accuracy.

Test



Test data set



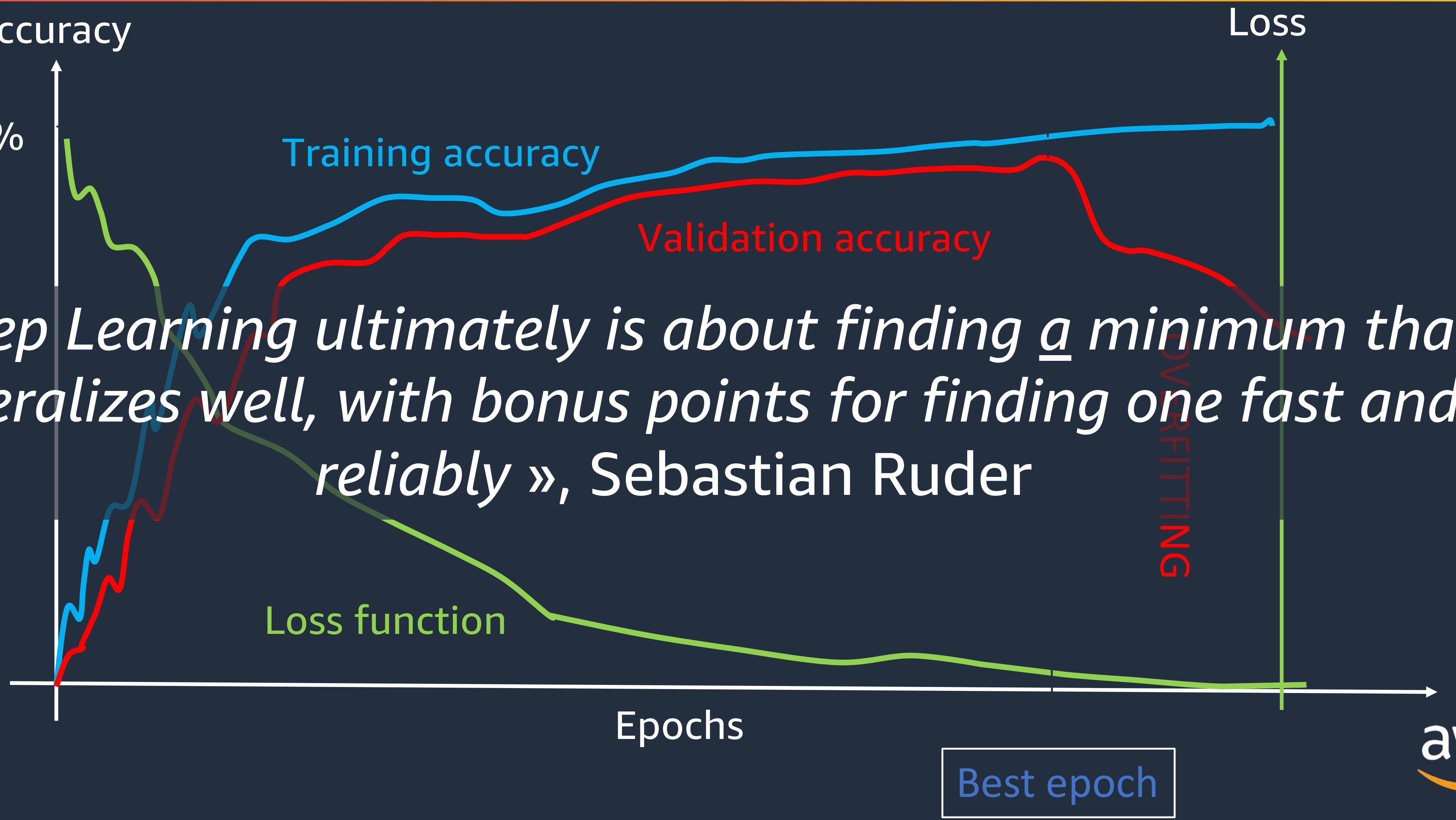
Fully trained neural network

Test accuracy

Prediction at the end of experimentation

This data set must have the same distribution as real-life samples, or else test accuracy won't reflect real-life accuracy.

Early stopping

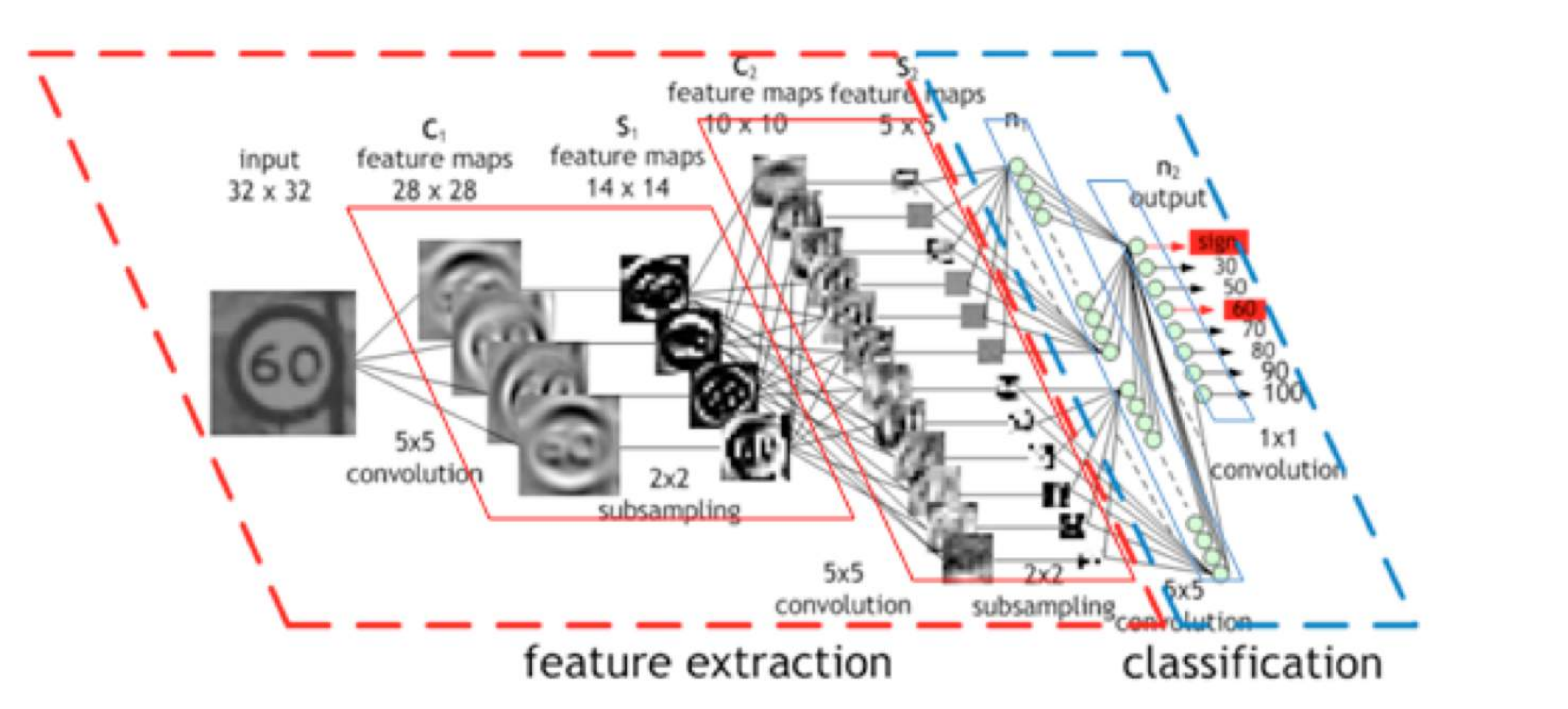


« Deep Learning ultimately is about finding a minimum that generalizes well, with bonus points for finding one fast and reliably », Sebastian Ruder

Common network architectures and use cases

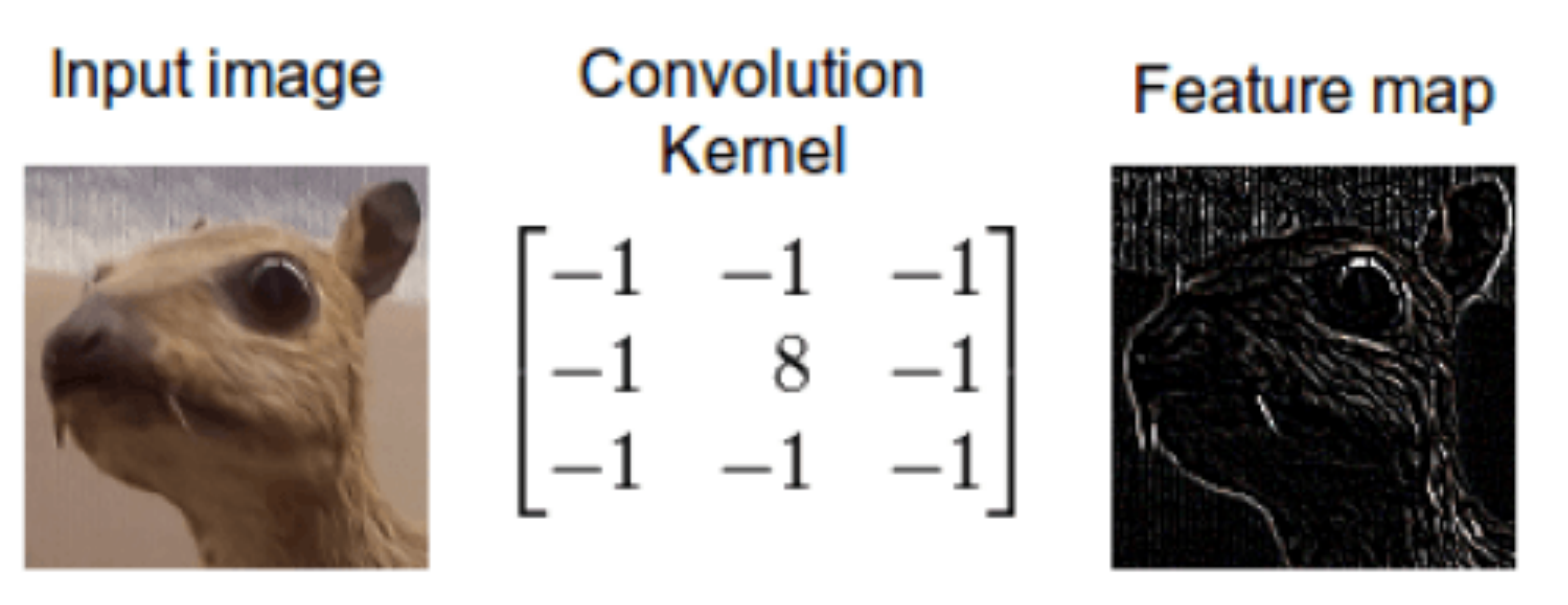
Convolutional Neural Networks (CNN)

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



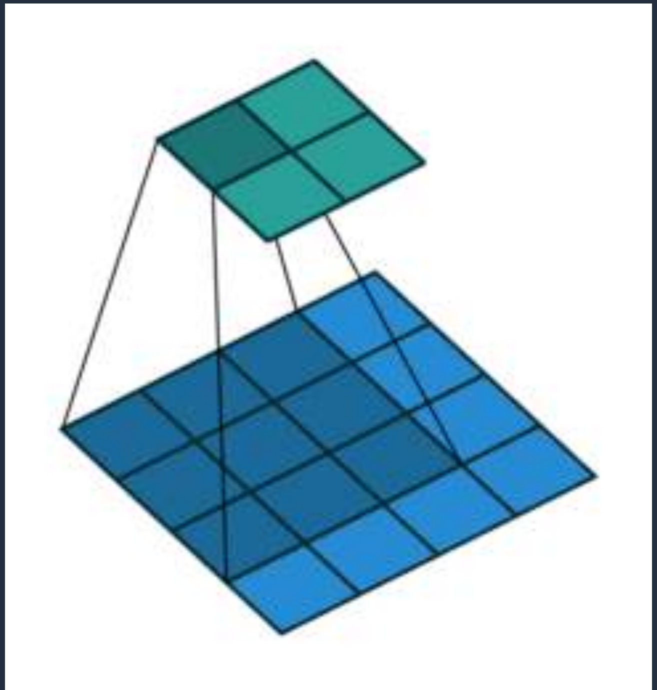
https://devblogs.nvidia.com/parallelforall/deep-learning-nutshell-core-concepts/

Extracting features with convolution



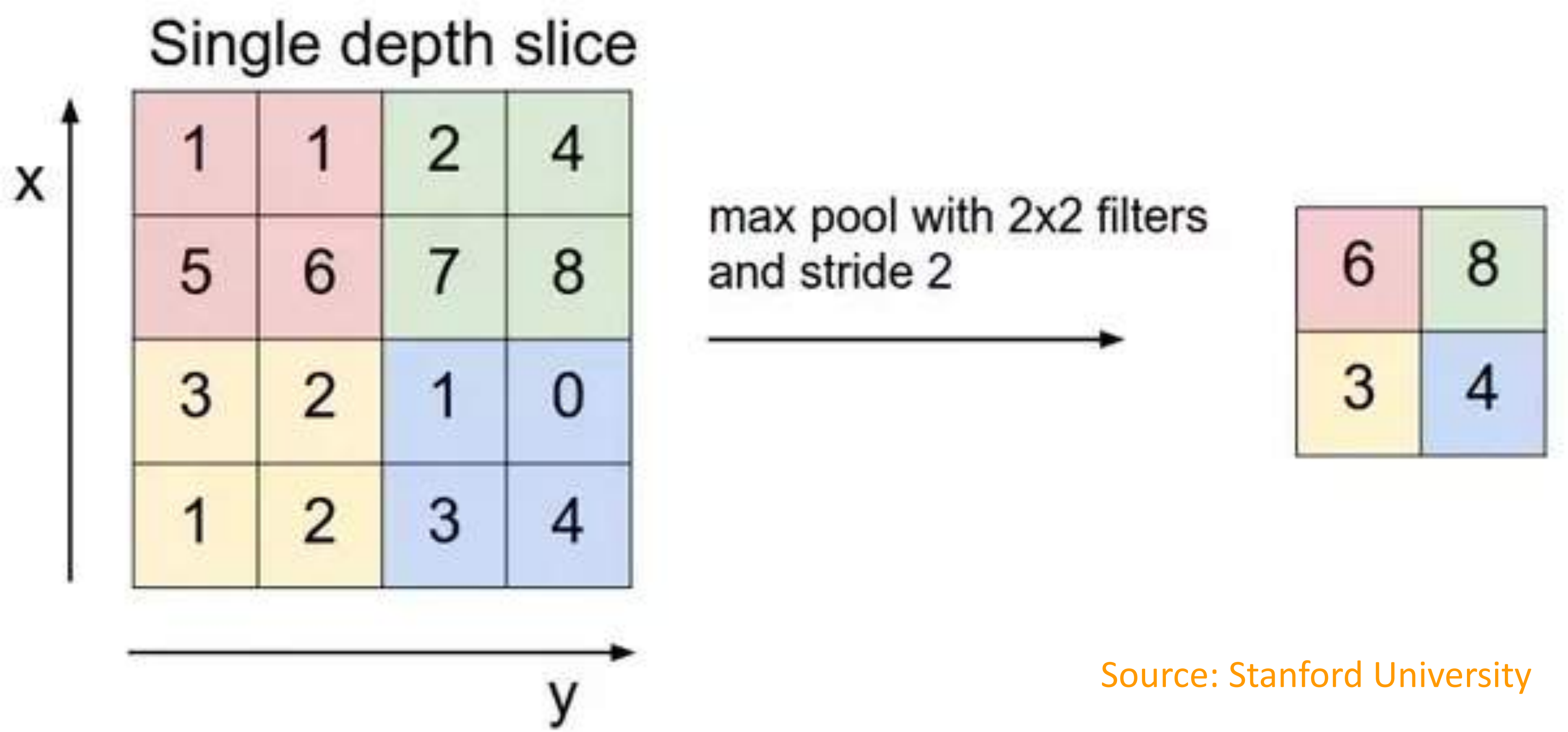
$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Source: http://timdettmers.com



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

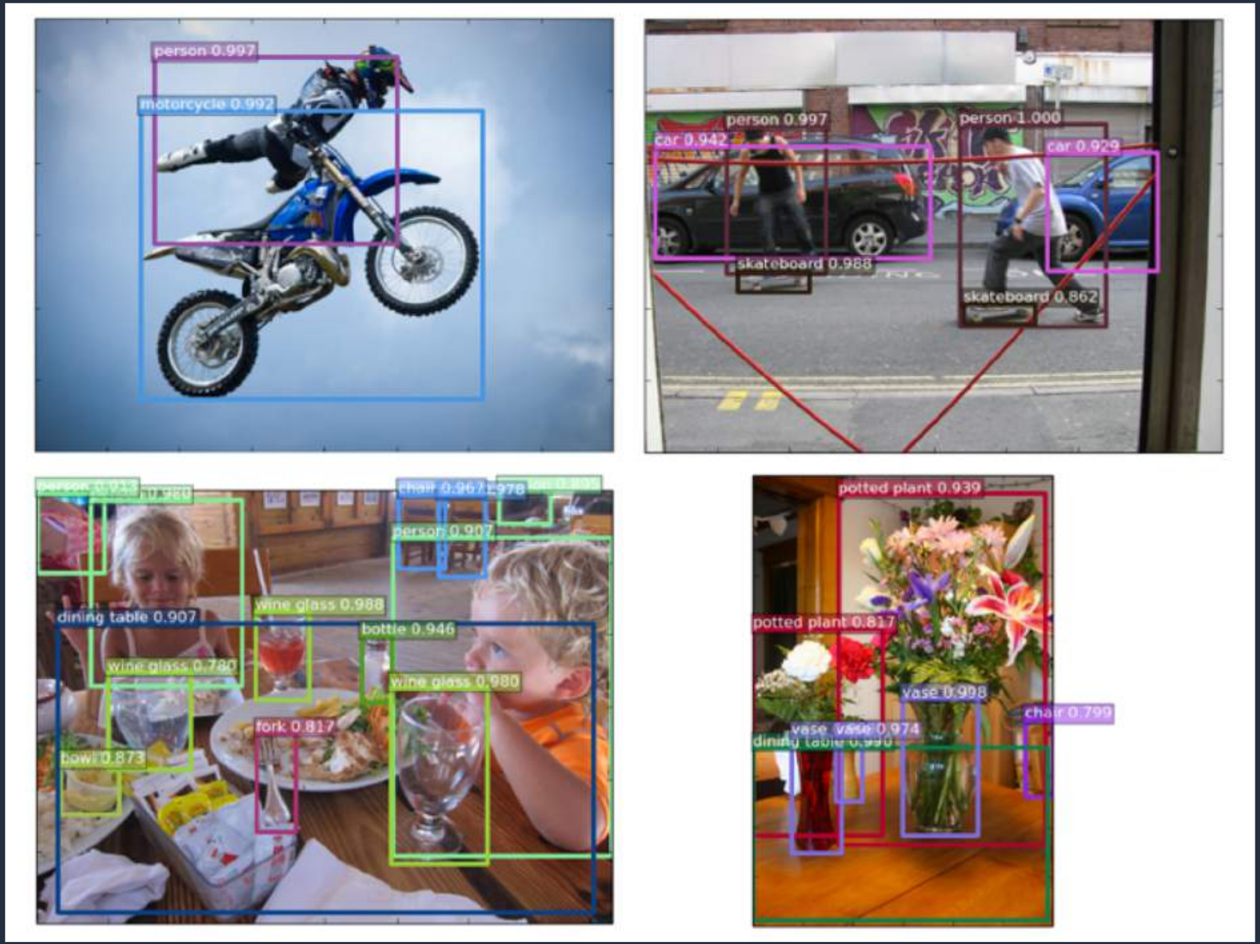
Downsampling images with pooling



Pooling shrinks images while preserving significant information.

Object Detection

MXNet



https://github.com/precedenceguo/mx-rcnn

https://github.com/zhreshold/mxnet-yolo

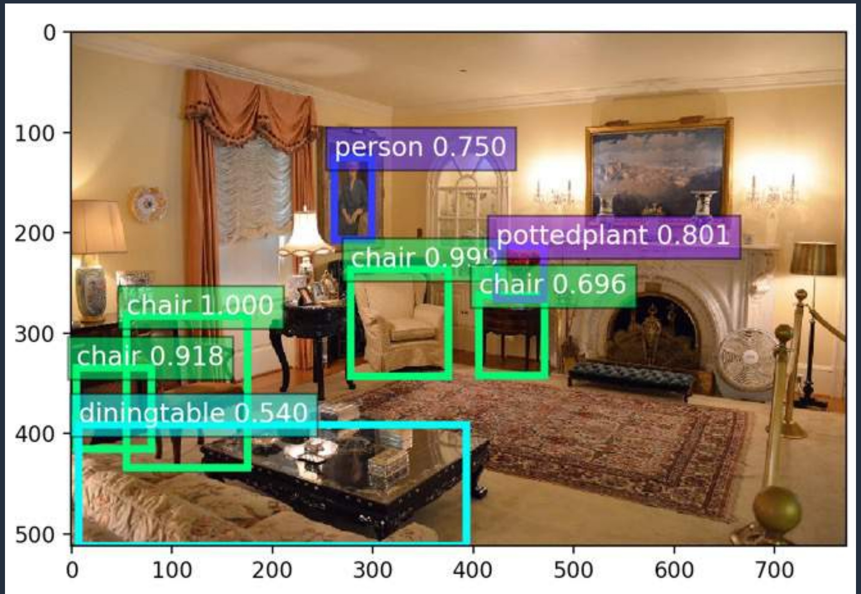
aws

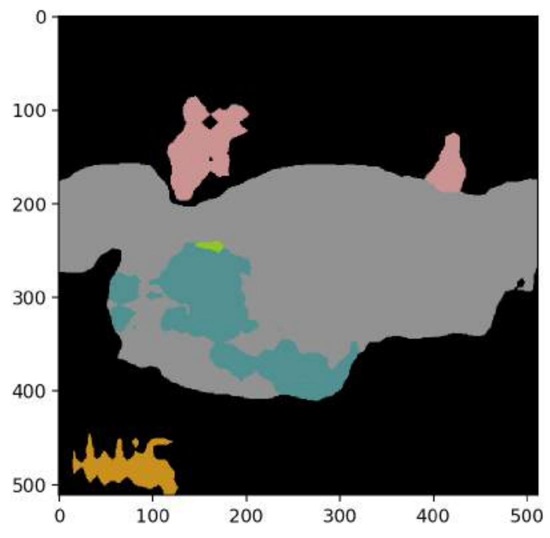
Object Segmentation

https://github.com/TuSimple/mx-maskrcnn

Classification, detection, segmentation

[electric_guitar],
with probability 0.671





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

Face Detection

```
attribution is:
   5_o_Clock_Shadow :  No
    Arched_Eyebrows :  No
         Attractive :  No
     Bags_Under_Eyes :  No
               Bald :  No
              Bangs :  No
           Big_Lips :  No
           Big_Nose :  No
         Black_Hair :  No
         Blond_Hair :  No
             Blurry :  Yes
         Brown_Hair :  No
     Bushy_Eyebrows :  No
             Chubby :  No
        Double_Chin :  No
         Eyeglasses :  No
             Goatee :  No
          Gray_Hair :  No
       Heavy_Makeup :  No
    High_Cheekbones :  No
               Male :  Yes
 Mouth_Slightly_Open :  No
           Mustache :  No
        Narrow_Eyes :  Yes
           No_Beard :  Yes
          Oval_Face :  No
          Pale_Skin :  No
        Pointy_Nose :  No
   Receding_Hairline :  No
        Rosy_Cheeks :  No
          Sideburns :  No
            Smiling :  No
      Straight_Hair :  No
          Wavy_Hair :  No
    Wearing_Earrings :  No
        Wearing_Hat :  No
    Wearing_Lipstick :  No
    Wearing_Necklace :  No
     Wearing_Necktie :  No
              Young :  Yes
```

https://github.com/tornadomeet/mxnet-face

Face Recognition

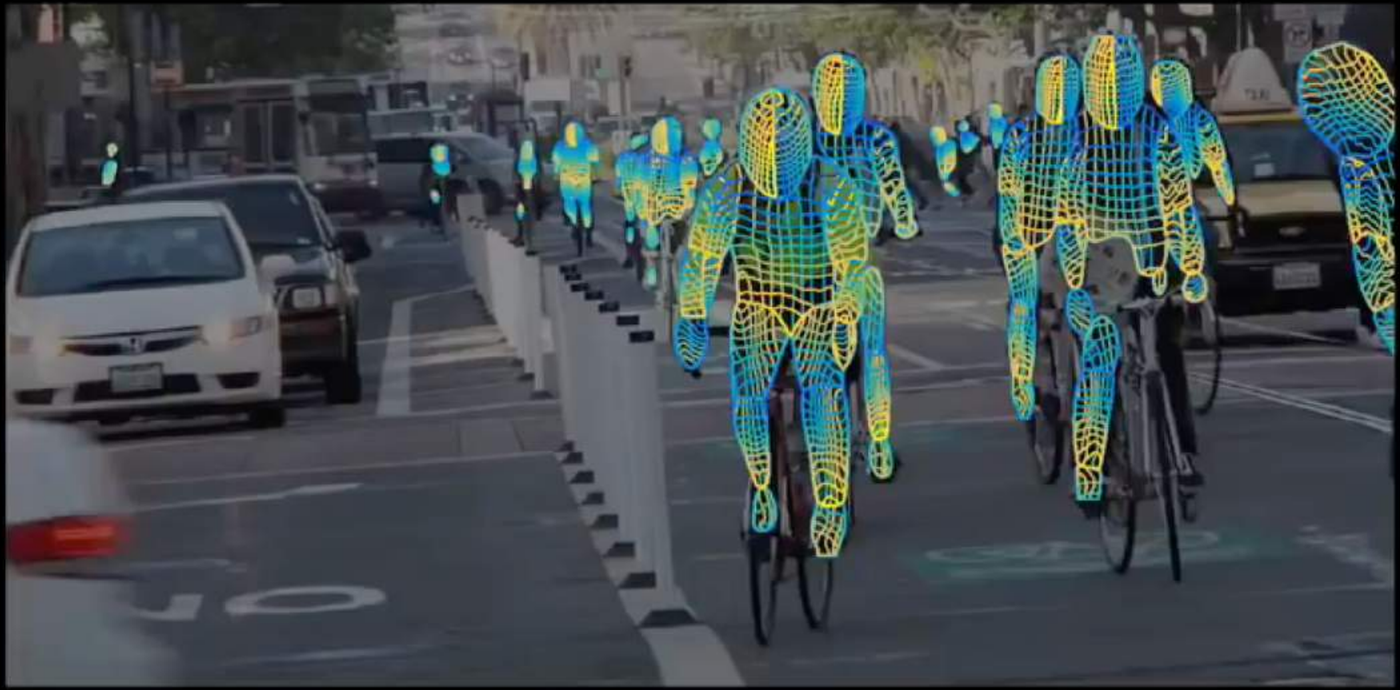
LFW 99.80%+
Megaface 98%+
with a single model

https://github.com/deepinsight/insightface
https://arxiv.org/abs/1801.07698

Real-Time Pose Estimation

https://github.com/dragonfly90/mxnet_Realtime_Multi-Person_Pose_Estimation

DensePose:
Dense Human Pose Estimation In The Wild



Rıza Alp Güler *
INRIA, CentraleSupélec

Natalia Neverova
Facebook AI Research

Iasonas Kokkinos
Facebook AI Research

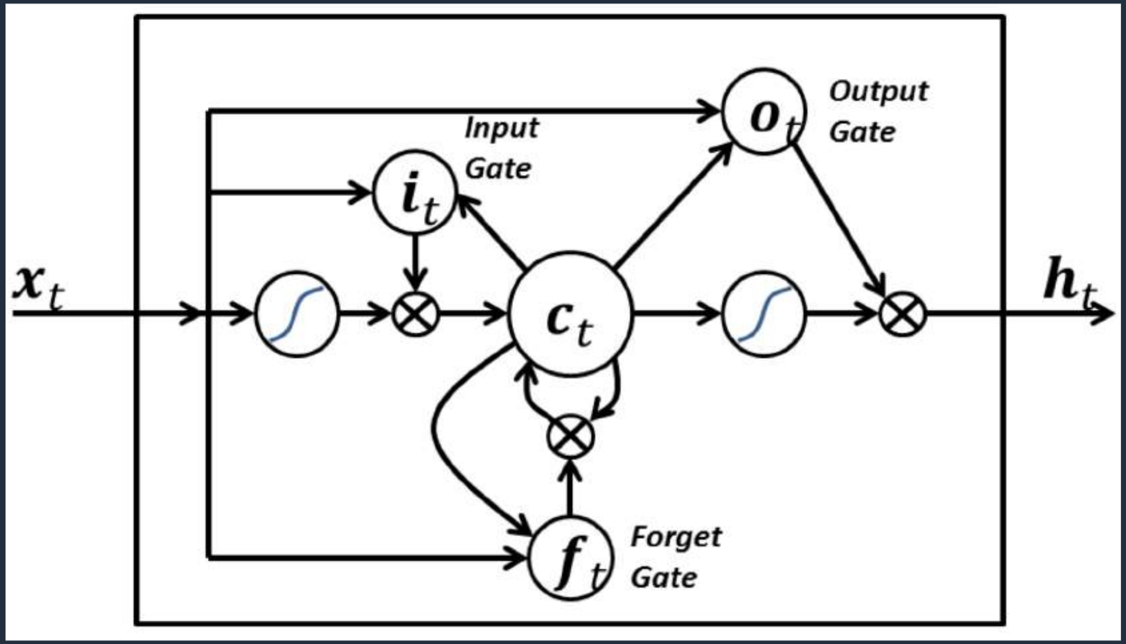
* Rıza Alp Güler was with Facebook AI Research during this work.

Long Short Term Memory Networks (LSTM)

A LSTM neuron computes the output based on the input and a previous state

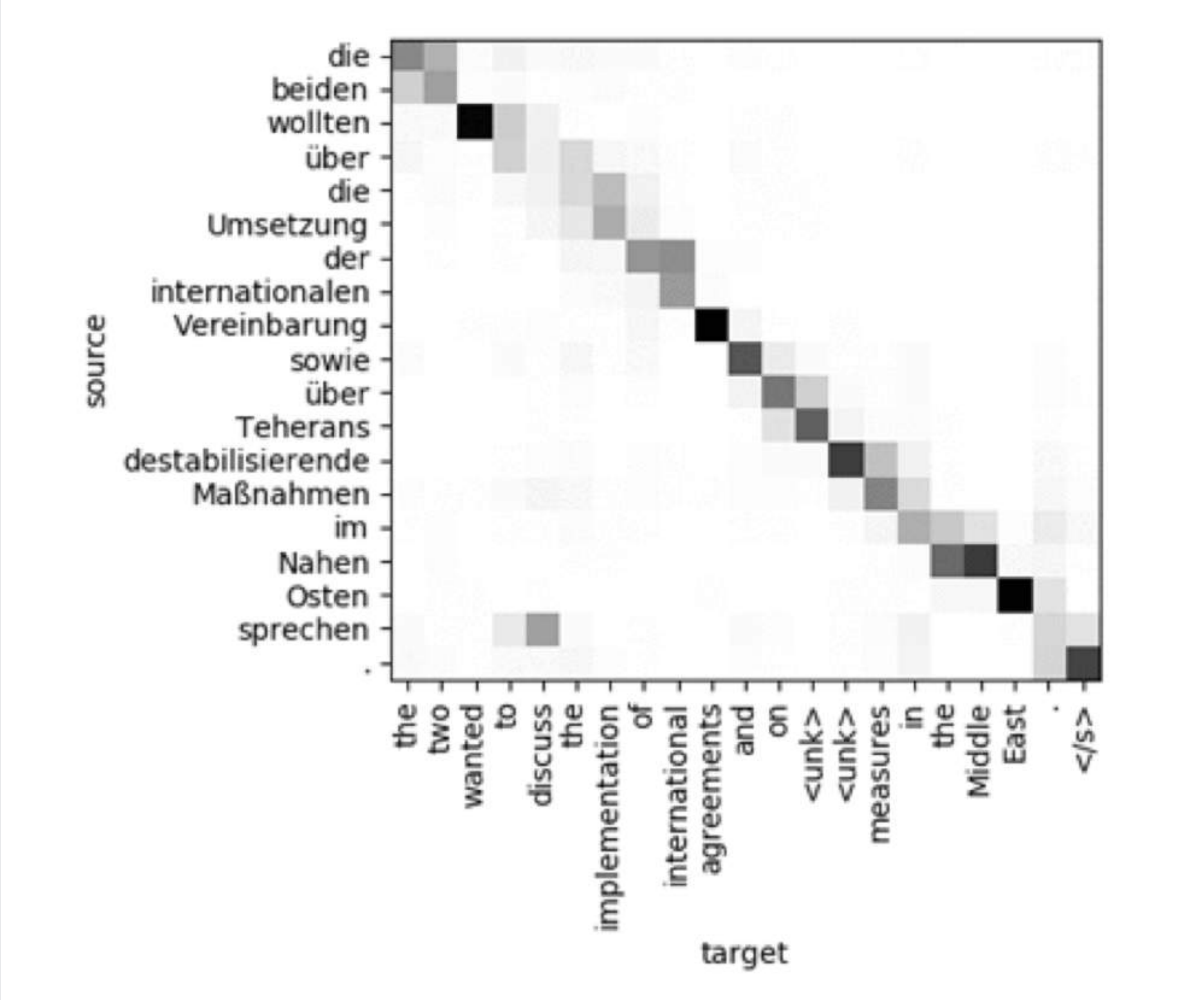
LSTM networks have memory

They're great at predicting sequences of data



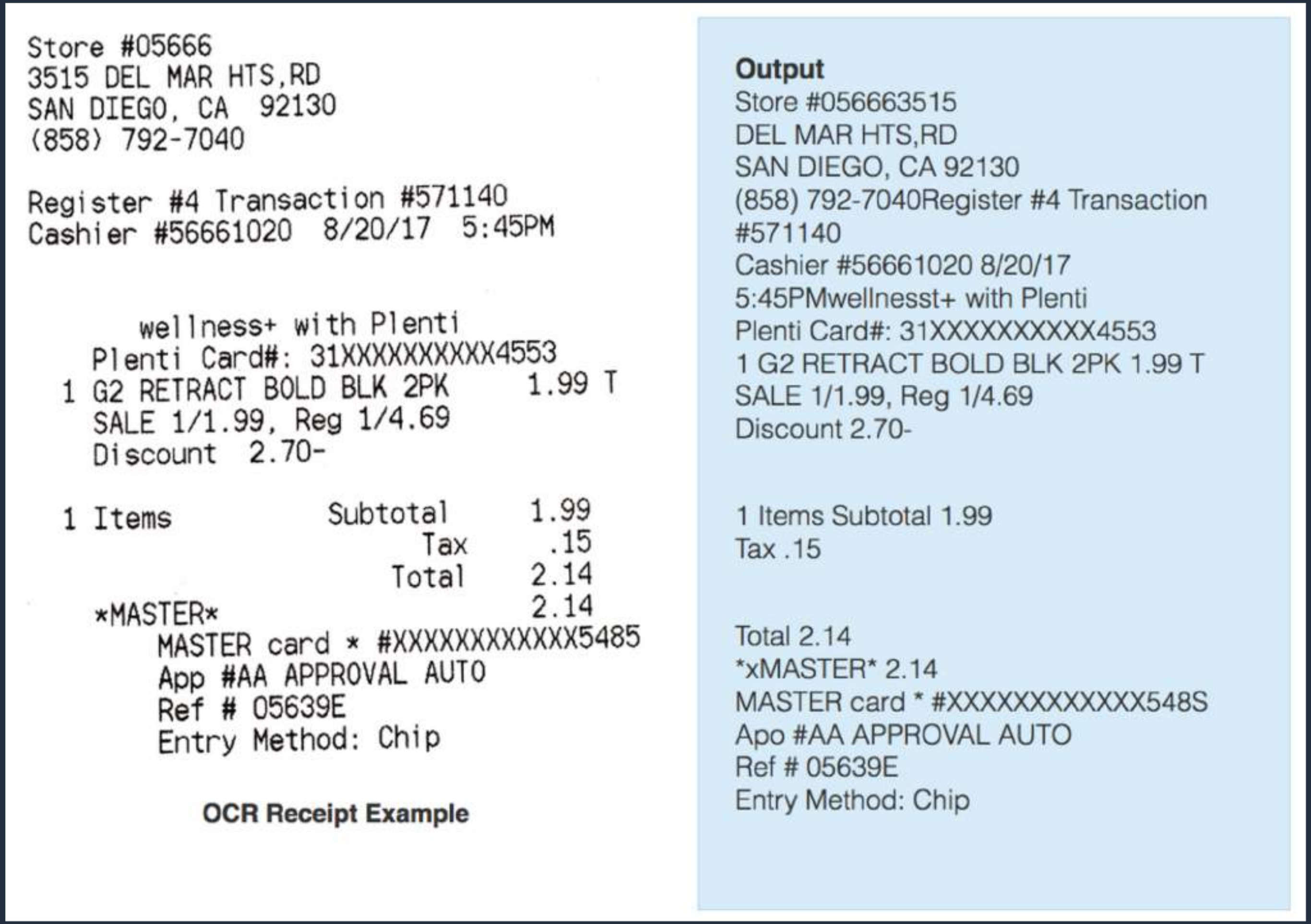
aws

Machine Translation – AWS Sockeye



https://github.com/awslabs/sockeye

OCR – Tesseract 4.0 (beta)



https://github.com/tesseract-ocr/tesseract/wiki/4.0-with-LSTM
https://www.learnopencv.com/deep-learning-based-text-recognition-ocr-using-tesseract-and-opencv/

Generative Adversarial Networks

TF

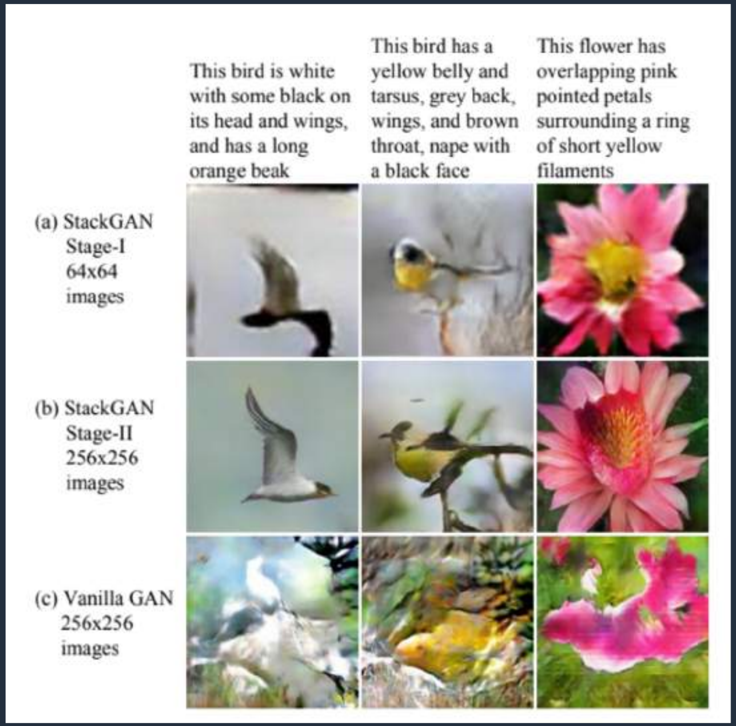
Generating new "celebrity" faces
https://github.com/tkarras/progressive_growing_of_gans

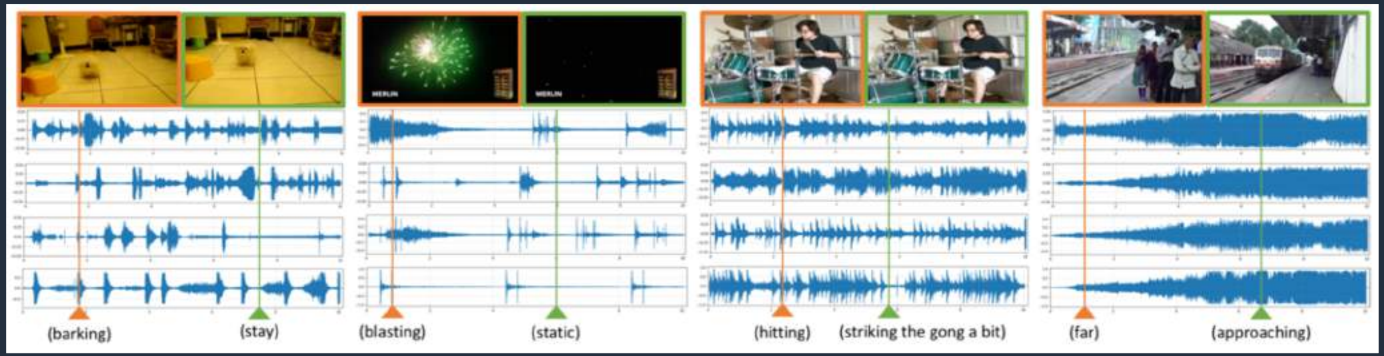
PyTorch

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

Wait! There's more!

Models can also generate text from text, text from images, text from video, images from text, sound from video, 3D models from 2D images, etc.





https://aws.amazon.com/solutions/case-studies/irobot/ @pitbullsharky

Medium

Applause from Adrian Hornsby and 33 others

Julien Simon
Hacker. Headbanger. Harley rider. Hunter. https://aws.amazon.com/evangelists/julien-simon/
Jan 12 · 5 min read

10 steps on the road to Deep Learning (part 1)

One of the questions I often get after my talks is: “*I’m a developer. How can I get started with this stuff?*”. Here’s how I worked my way into Deep Learning. By no means am I claiming that this is “The Way”, if such a thing even exists.

In this post and the next, I’ll go through **10 steps** that will hopefully help you learn in the right order and at your own pace.

Deep Learning - The Straight Dope

This repo contains an incremental sequence of notebooks designed to teach deep learning, Apache MXNet (incubating), and the gluon interface. Our goal is to leverage the strengths of Jupyter notebooks to present prose, graphics, equations, and code together in one place. If we're successful, the result will be a resource that could be simultaneously a book, course material, a prop for live tutorials, and a resource for plagiarising (with our blessing) useful code. To our knowledge there's no source out there that teaches either (1) the full breadth of concepts in modern deep learning or (2) interleaves an engaging textbook with runnable code. We'll find out by the end of this venture whether or not that void exists for a good reason.

Another unique aspect of this book is its authorship process. We are developing this resource fully in the public view and are making it available for free in its entirety. While the book has a few primary authors to set the tone and shape the content, we welcome contributions from the community and hope to coauthor chapters and entire sections with experts and community members. Already we've received contributions spanning typo corrections through full working examples.

How to contribute

To clone or contribute, visit Deep Learning - The Straight Dope on Github

Getting started

https://aws.amazon.com/machine-learning

https://aws.amazon.com/blogs/ai

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

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

https://medium.com/@julsimon

https://youtube.com/juliensimonfr

https://gitlab.com/juliensimon/dlnotebooks

AWS DeepLens

INNOVATE 2018
ONLINE CONFERENCE

DEVELOPER EDITION

Thank you!

Julien Simon @julsimon

Julio Faerman @faermanj

@awscloud

#AWSInnovate