Deep Learning: concepts and use cases

Julien Simon

Principal Technical Evangelist, AI and Machine Learning, AWS

@julsimon

November 2018

What to expect

- An introduction to Deep Learning theory
  - Neurons & Neural Networks
  - The Training Process
  - Backpropagation
  - Optimizers

- Common network architectures and use cases
  - Convolutional Neural Networks
  - Recurrent Neural Networks
  - Long Short Term Memory Networks
  - Generative Adversarial Networks

- Getting started

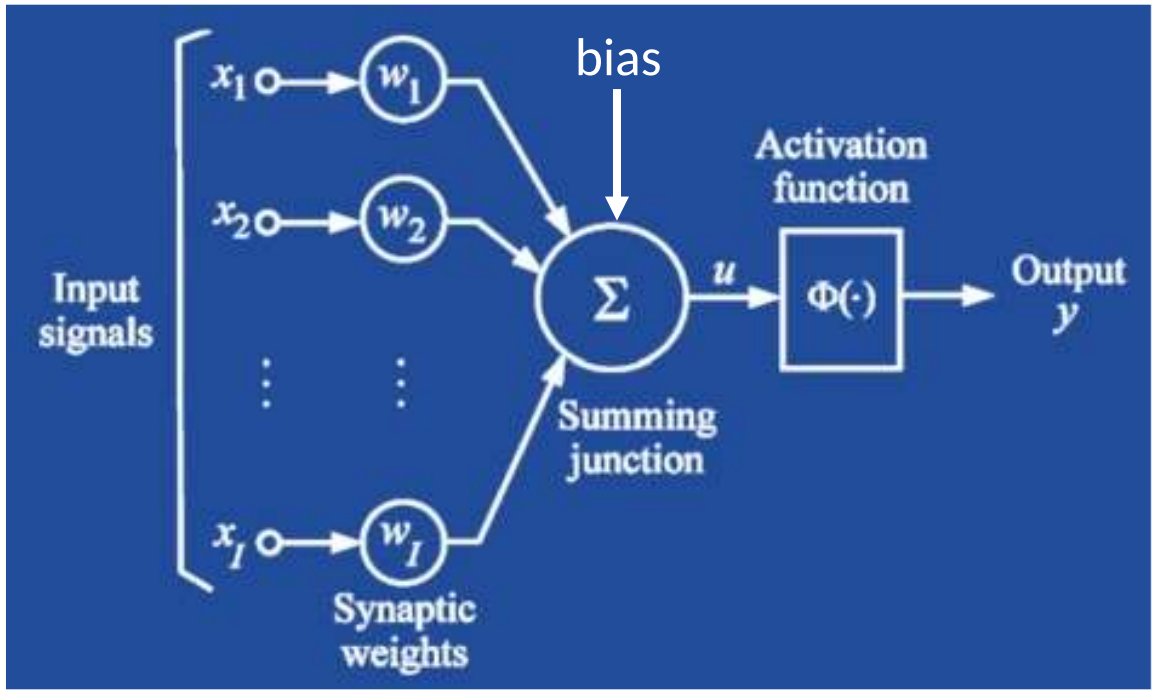
- Artificial Intelligence: design software applications which exhibit human-like behavior, e.g. speech, natural language processing, reasoning or intuition

- Machine Learning: using statistical algorithms, teach machines to learn from featurized data without being explicitly programmed

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

An introduction to Deep Learning theory

The neuron



$$\sum_{i=1}^{l} x_i * wi + b = u$$

”Multiply and Accumulate”

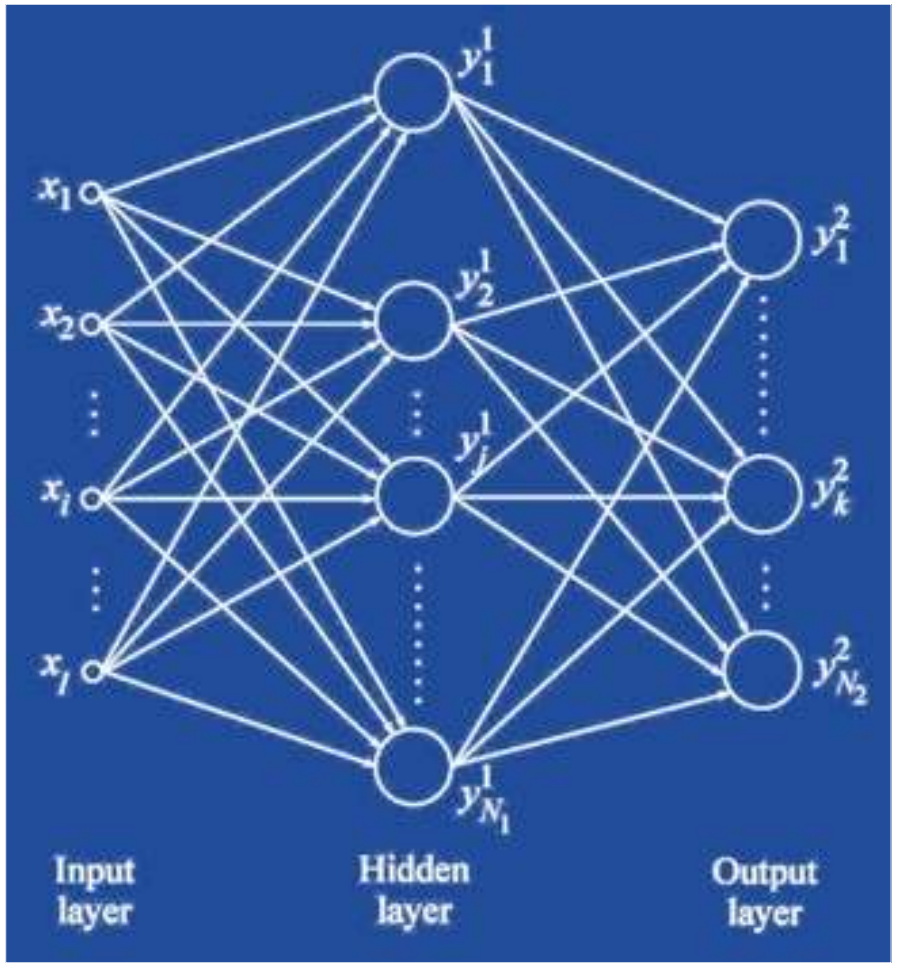
Activation functions

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

Building a simple classifier



Biases are ignored for the rest of this discussion

l features

$$x = \begin{bmatrix} x_{11}, x_{12}, \ldots x_{1l} \\ x_{21}, x_{22}, \ldots x_{2l} \\ \ldots \ldots \ldots \\ x_{m1}, x_{m2}, \ldots x_{ml} \end{bmatrix}$$

m samples

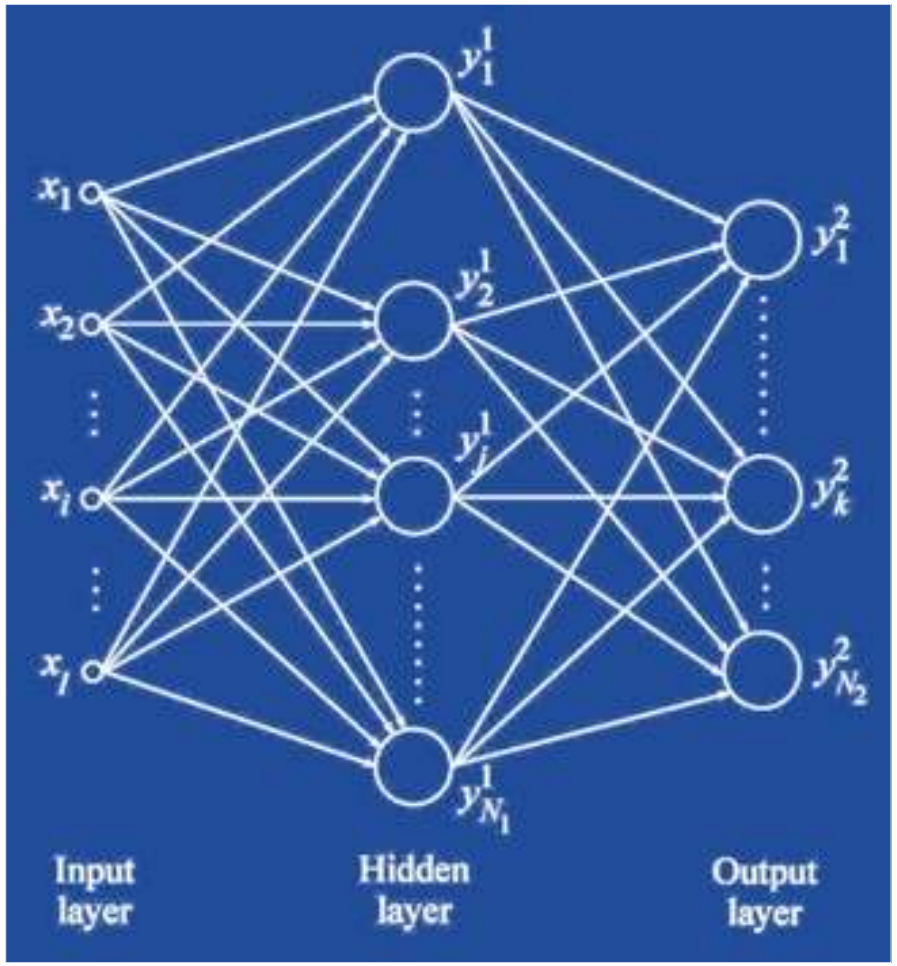
$$y = \begin{bmatrix} 2 \\ 0 \\ \ldots \\ 4 \end{bmatrix} \quad \begin{bmatrix} 0,0,1,0,0,\ldots,0 \\ 1,0,0,0,0,\ldots,0 \\ \ldots \\ 0,0,0,0,1,\ldots,0 \end{bmatrix}$$

One-hot encoding

m labels,
N_2 categories

Neural networks

Building a simple classifier



l features

$$x = \begin{bmatrix} x_{11}, x_{12}, \ldots x_{1l} \\ x_{21}, \ldots x_{2l} \\ \ldots \\ x_{m1}, \ldots x_{ml} \end{bmatrix}$$

m samples

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

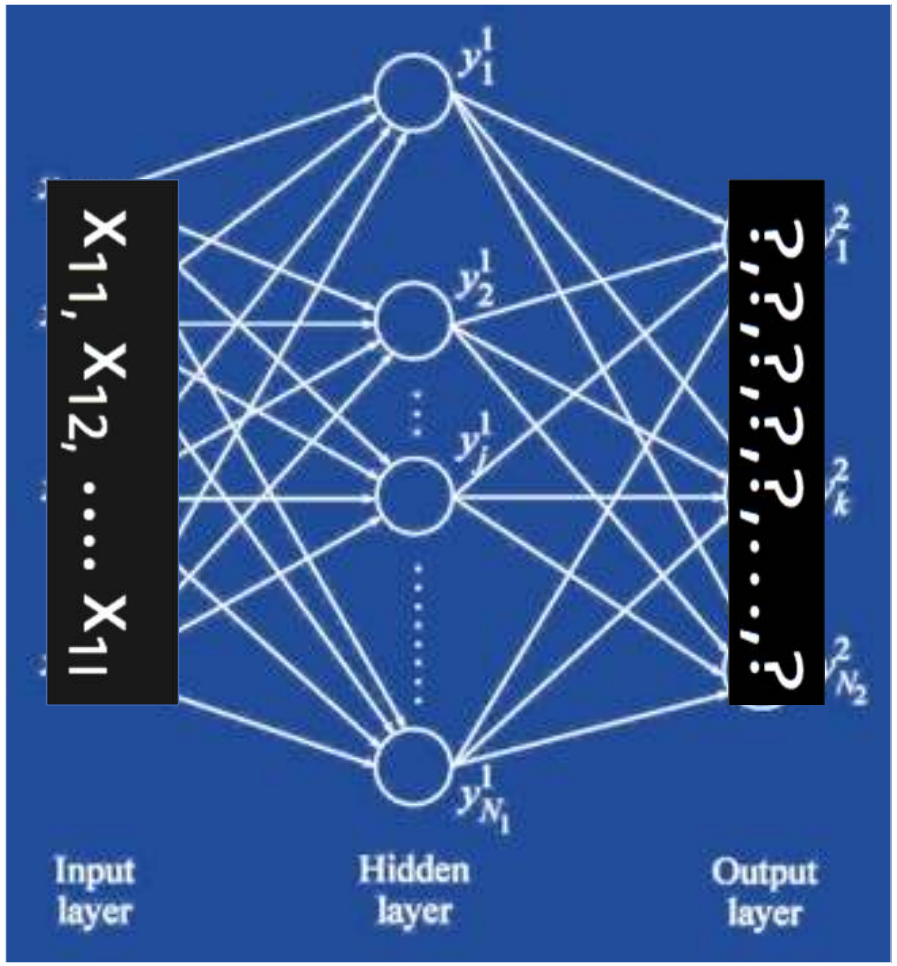
m labels,
N_2 categories

One-hot encoding

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

Neural networks

Building a simple classifier



Initially, the network will not predict correctly
$f(X_1) = Y'_1$

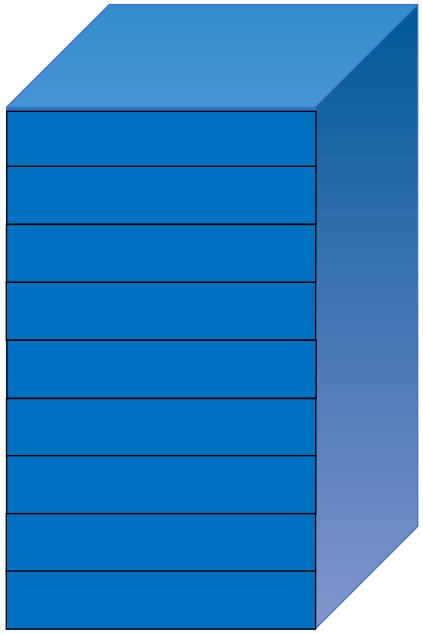
A loss function measures the difference between the real label Y_1 and the predicted label Y'_1
$error = loss(Y_1, Y'_1)$

For a batch of samples:

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

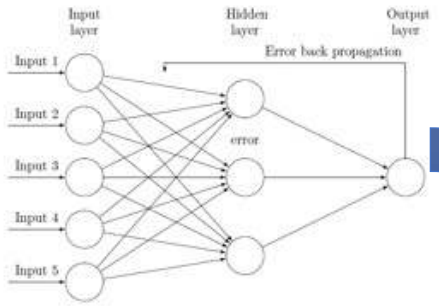
The purpose of the training process is to minimize error by gradually adjusting weights.

Mini-batch Training

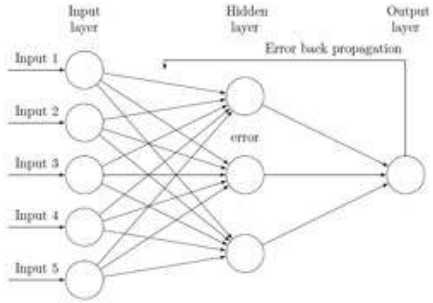


Training data set

Forward propagation



Backpropagation



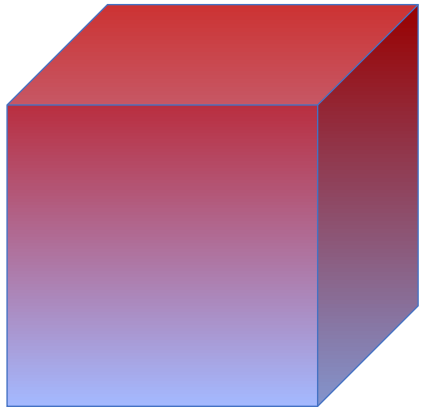
Trained
neural network

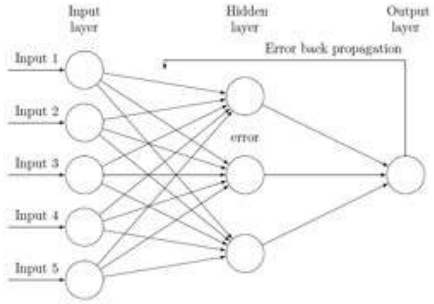
Batch size
Learning rate
Number of epochs

Hyper parameters

aws

Validation





Validation accuracy

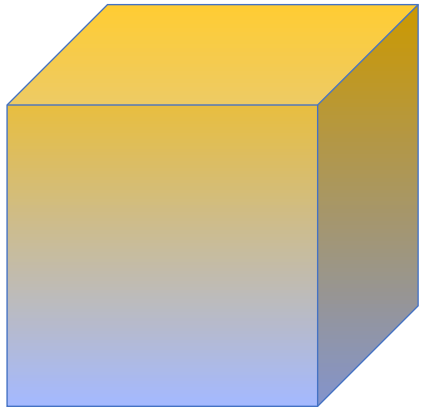
Validation data set (also called dev set)

Neural network in training

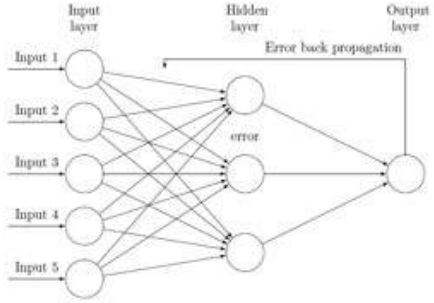
Prediction at the end of each epoch

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

Prediction at the end of experimentation

Test accuracy

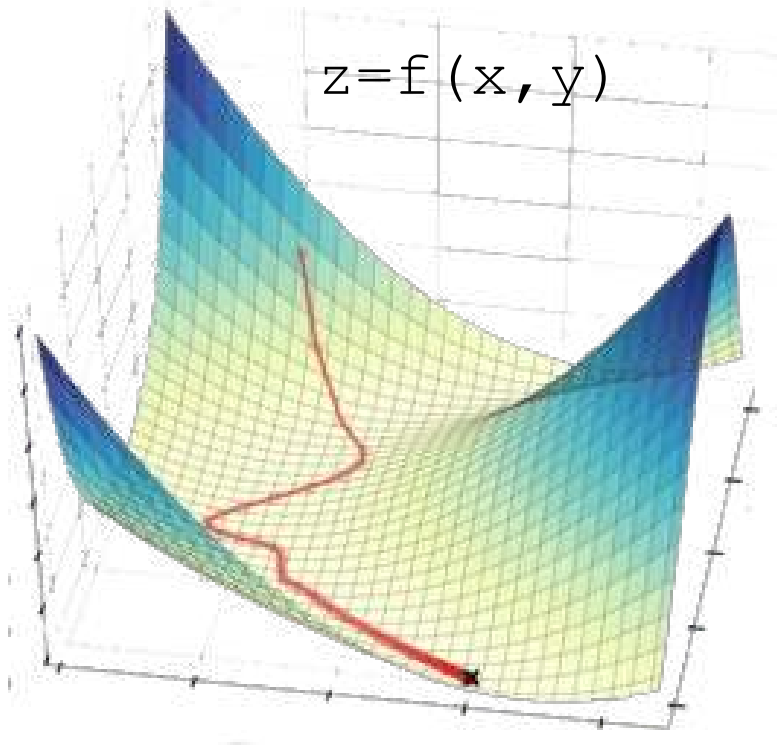
This data set must have the same distribution as real-life samples, or else test accuracy won't reflect real-life accuracy.

Stochastic Gradient Descent (1951)

Imagine you stand on top of a mountain (…). 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.

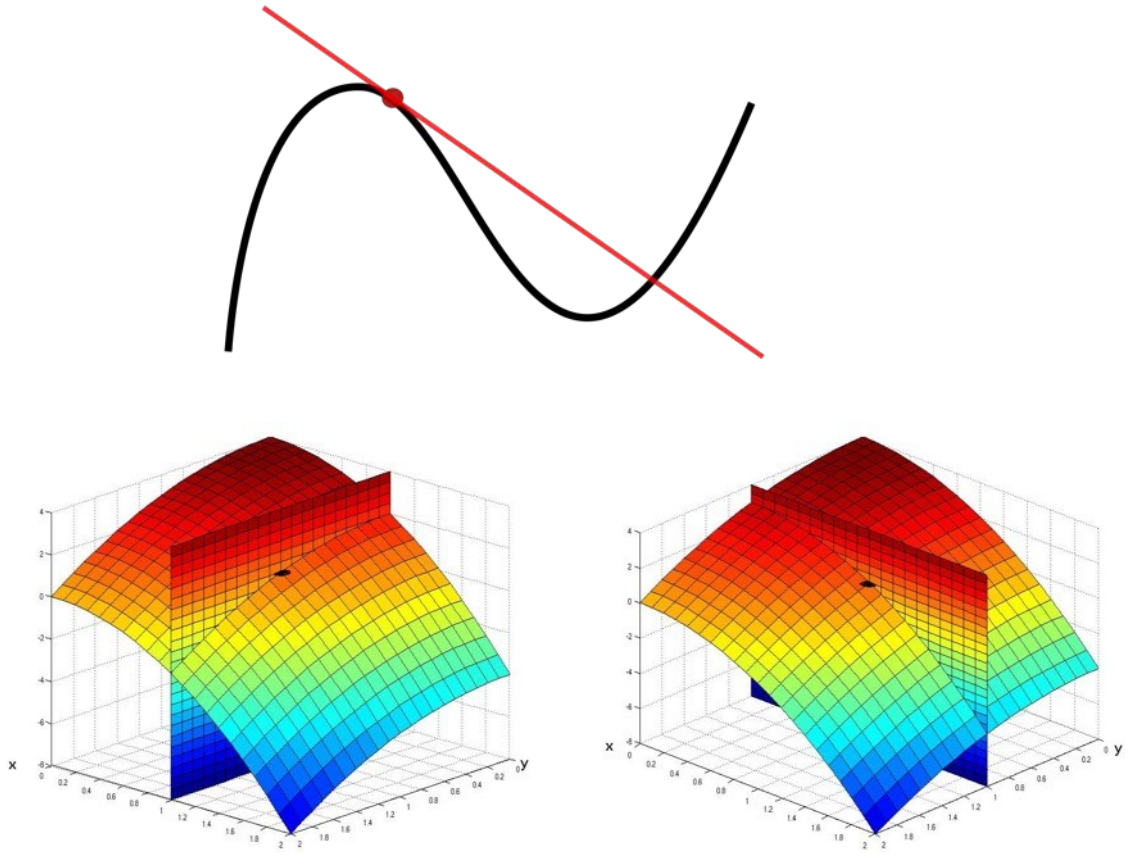
Tim Dettmers, University of Lugano, 2015



The « step size » depends on the learning rate

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

Finding the slope with Derivatives



Scalar-valued multivariable function

$$\nabla f(x_0, y_0, \dots) = \begin{bmatrix} \frac{\partial f}{\partial x}(x_0, y_0, \dots) \\ \frac{\partial f}{\partial y}(x_0, y_0, \dots) \\ \vdots \end{bmatrix}$$

∇f takes the same type of inputs as f

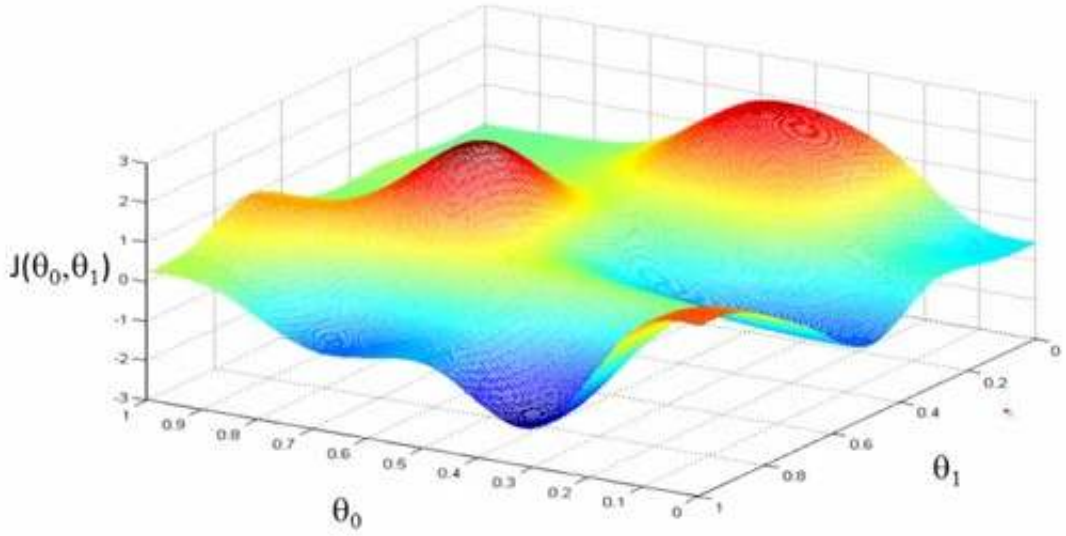
Notation for gradient, called "nabla".

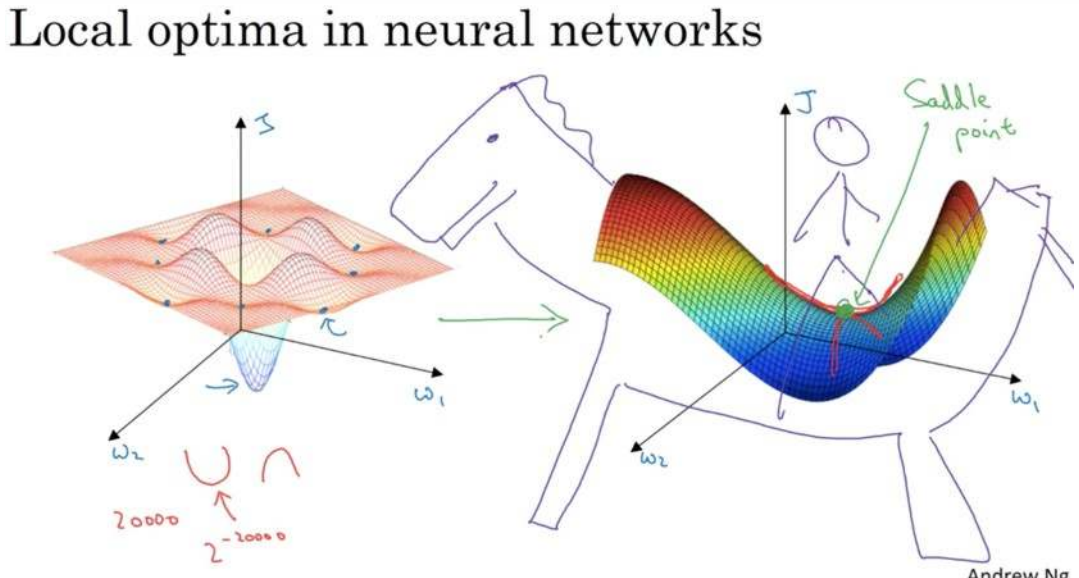
∇f outputs a vector with all possible partial derivatives of f.

End-to-end example of computing backpropagation with partial derivatives: https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example

Source: Wikipedia, Oklahoma State University, Khan

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

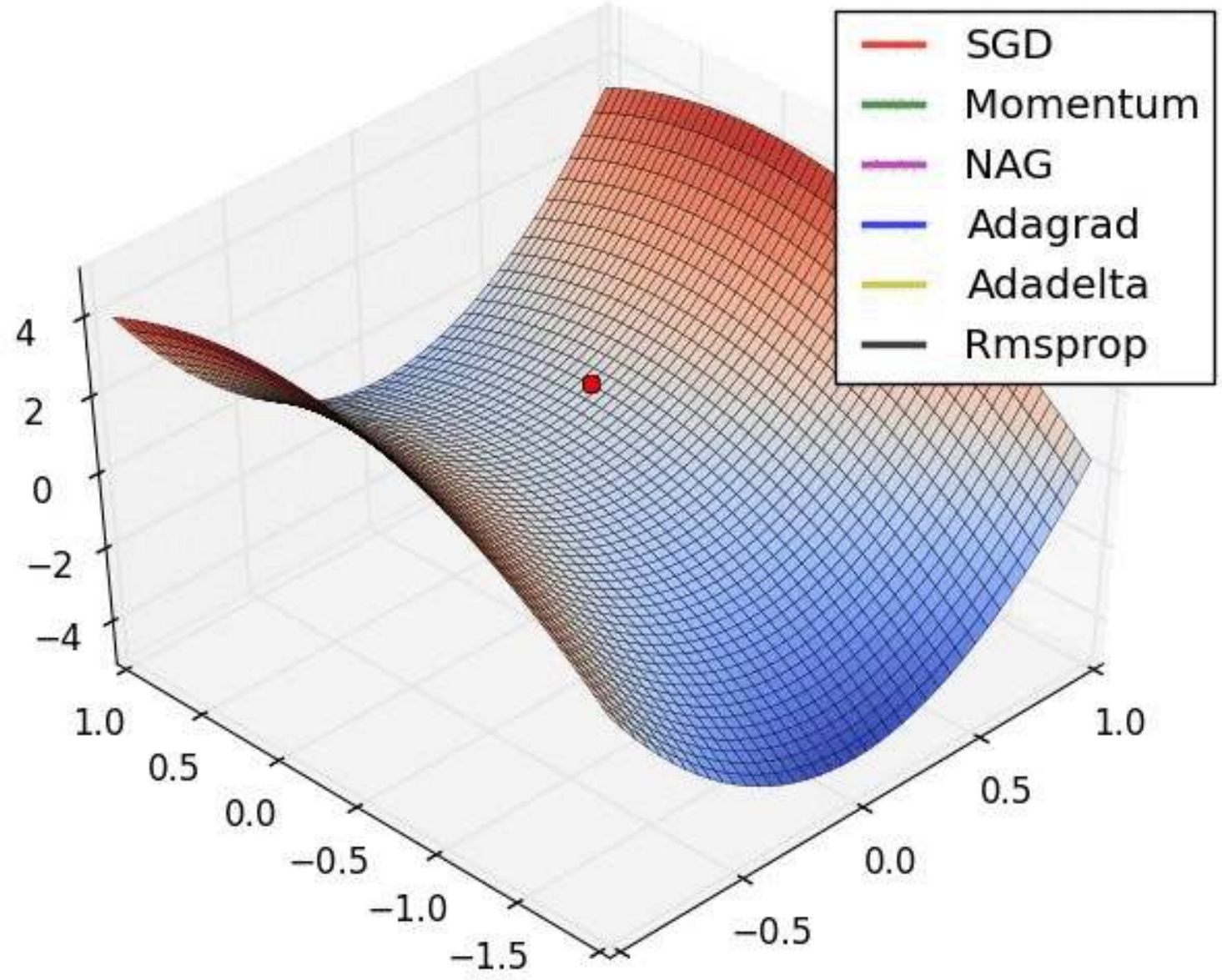
aws

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).

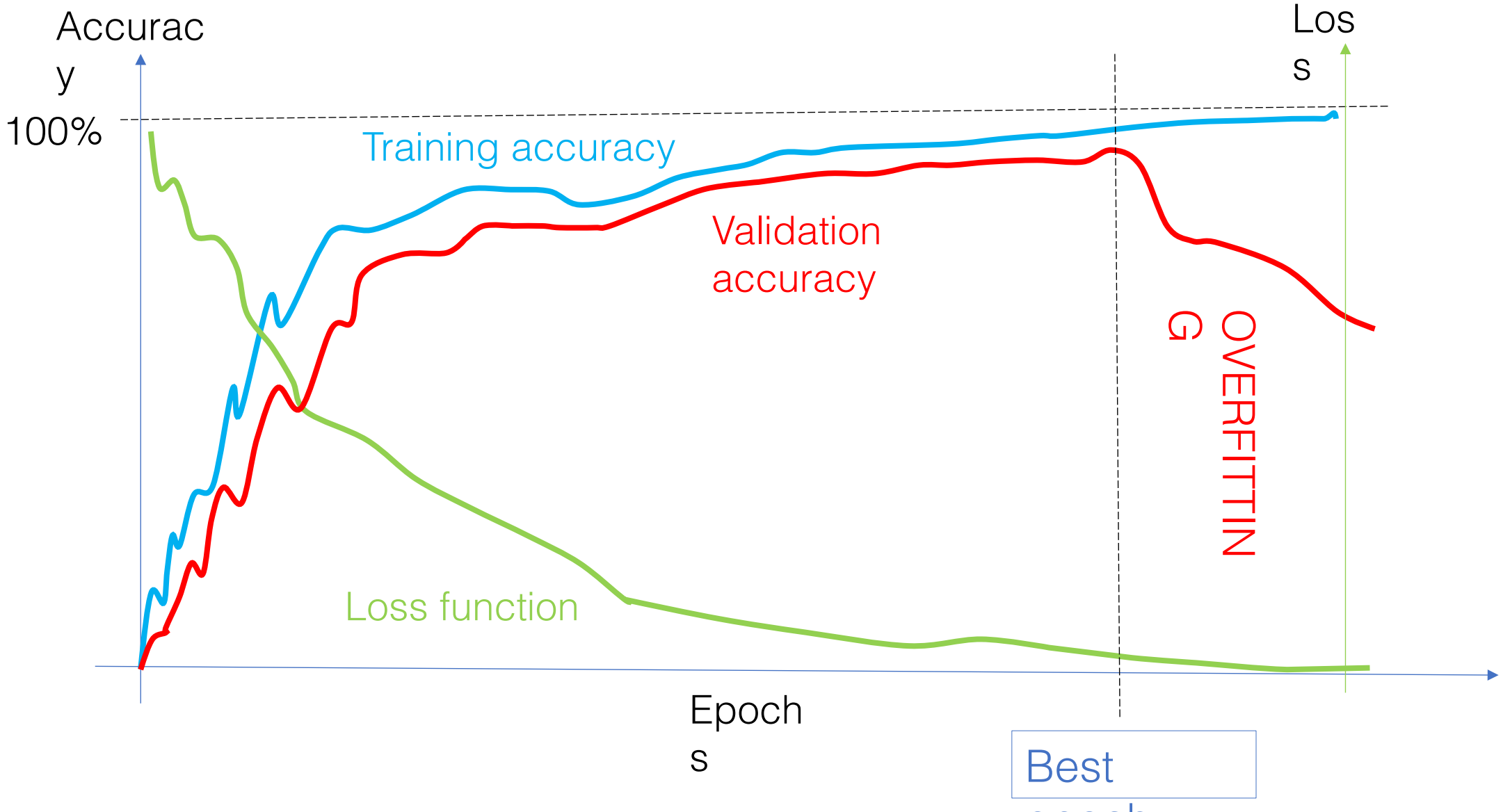


https://medium.com/@julsimon/tumbling-down-the-sgd-rabbit-hole-part-1-740fa402f0d7

aws

Early stopping

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



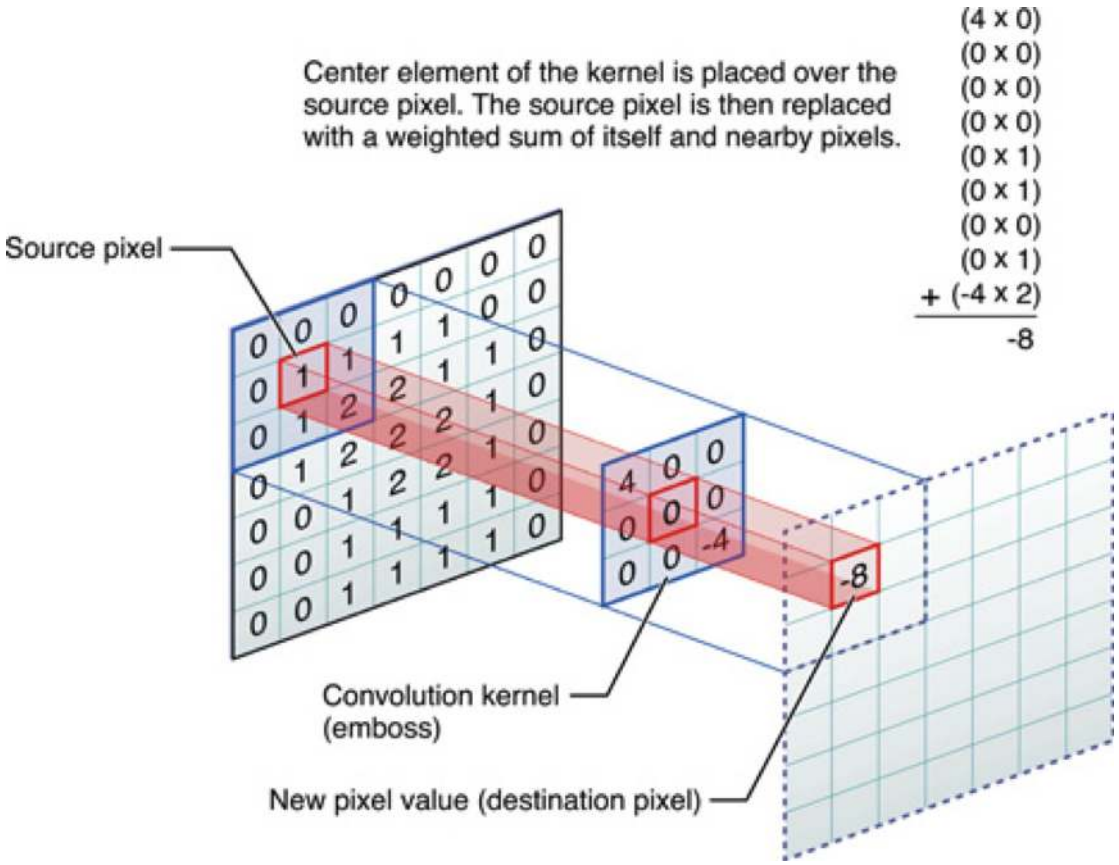
Common network architectures and use cases

Fully Connected Networks are nice, but…

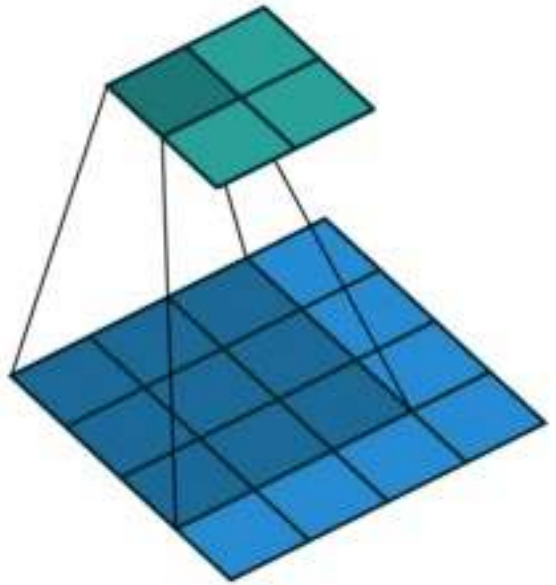
- What if we need lots of layers in order to extract complex features?
  - The number of parameters increases very quickly with the number of layers
  - Overfitting is a constant problem
- What about large data?
  - 256x256 images = 65,535 input neurons ?
- What about 2D/3D data ? Won't we lose lots of info by flattening it?
  - Images, videos, etc.
- What about sequential data, where the order of samples is important?
  - Translating text
  - Predicting time series

Convolutional Neural Networks

The convolution operation



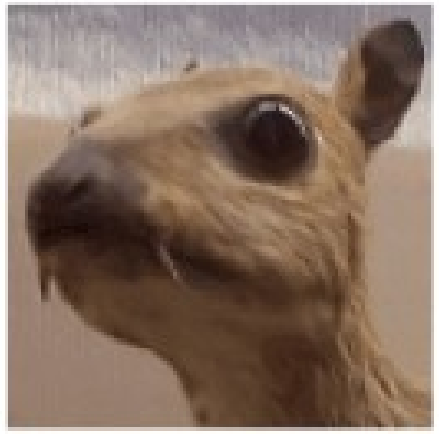
Source: https://ikhlestov.github.io/pages/machine-learning/convolutions-types/



Source: Theano documentation

Extracting features with convolution

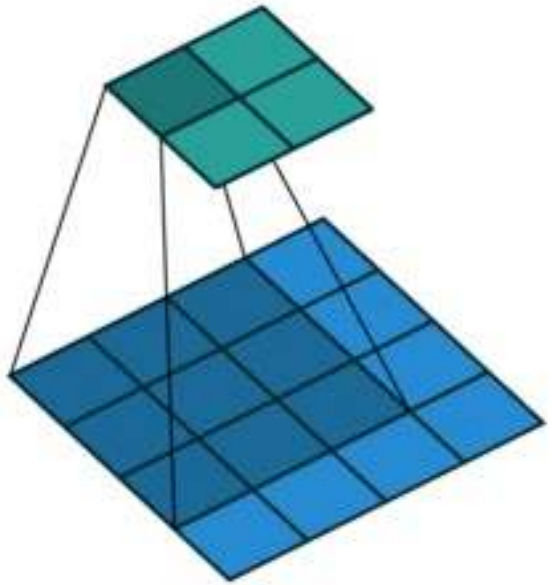
Input image



Convolution Kernel

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

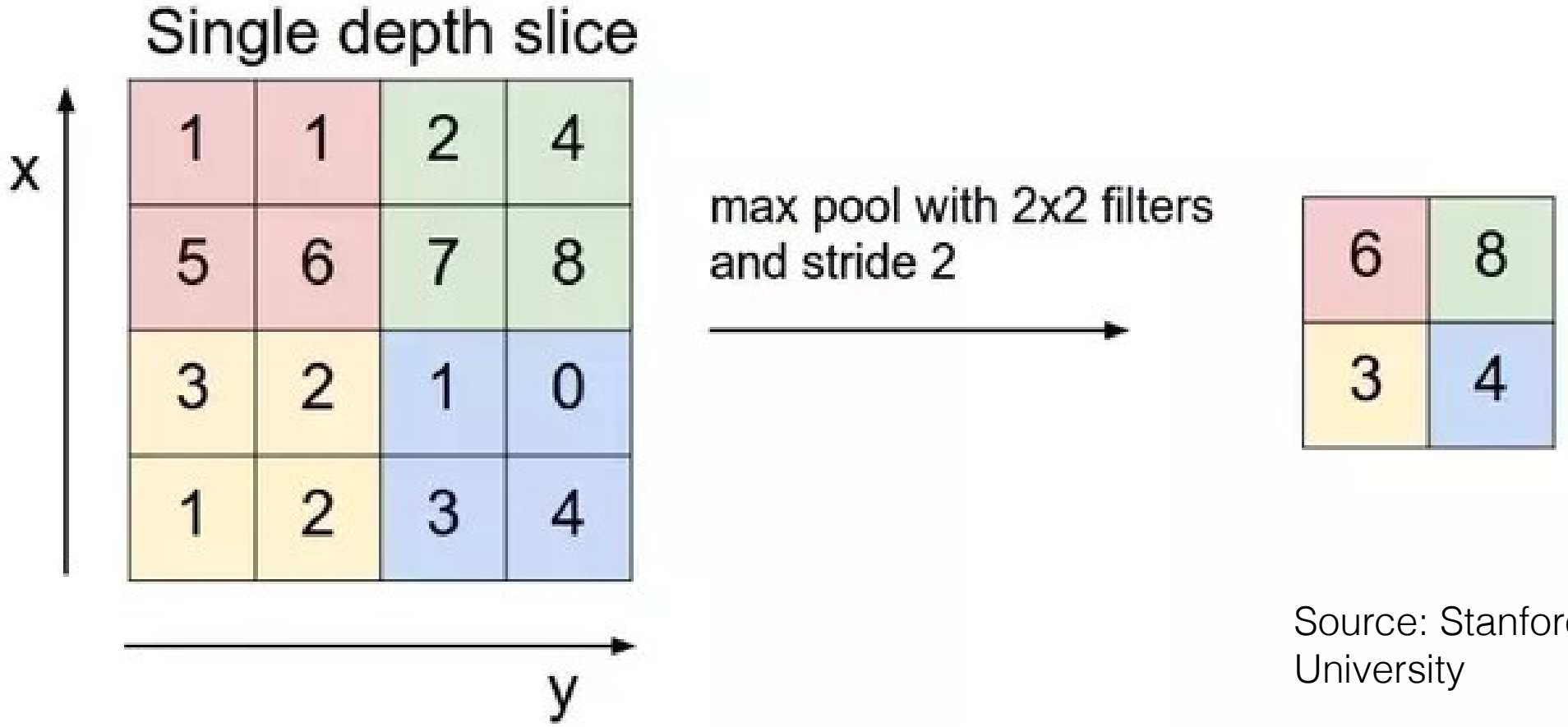
Feature map



Source: http://timdettmers.com

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

Downsampling images with pooling

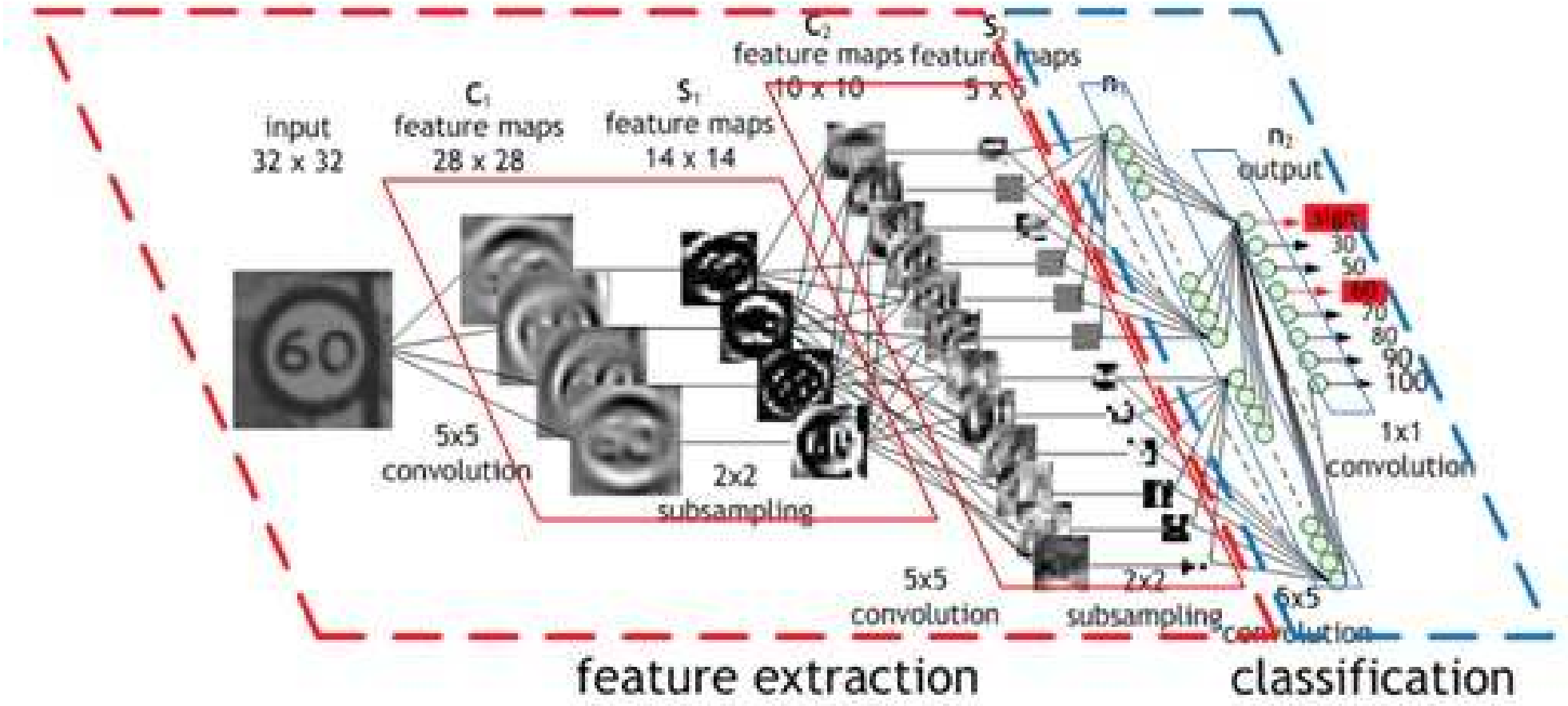


Source: Stanford University

Pooling shrinks images while preserving significant information.

Convolutional Neural Networks (CNN)

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

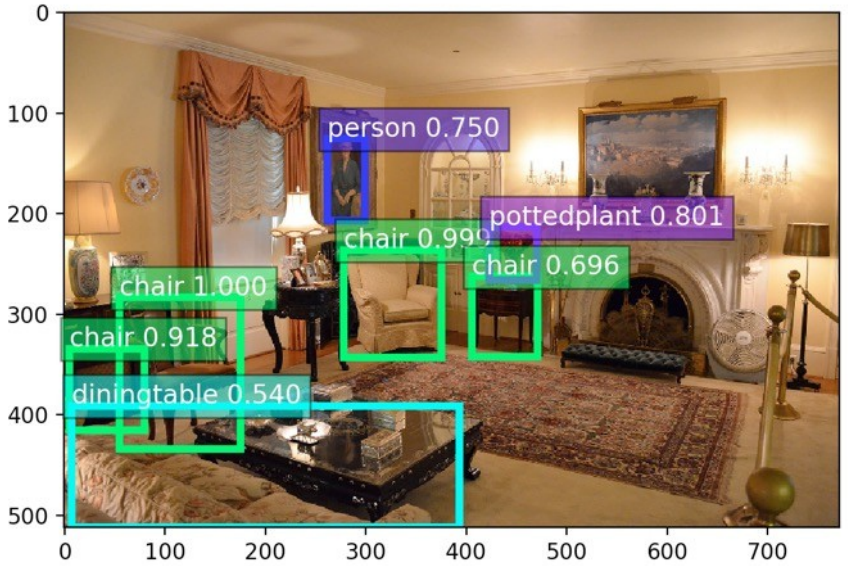


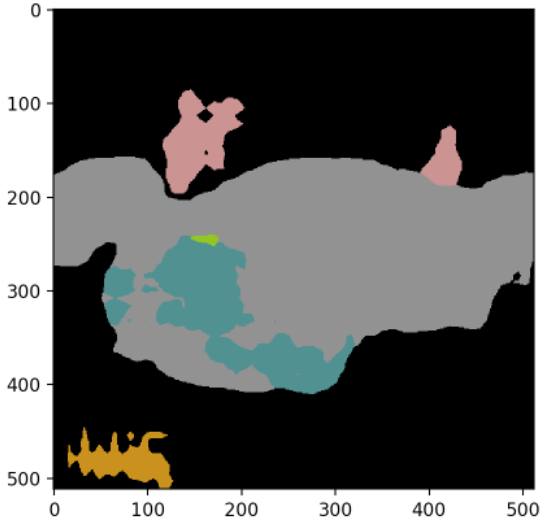
https://devblogs.nvidia.com/parallelforall/deep-learning-nutshell-core-concepts/

aws

Classification, detection, segmentation

[electric_guitar],
with probability 0.671





https://github.com/dmlc/gluon-cv
Based on models published in 2015-2017

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
Based on models published 2015-2016

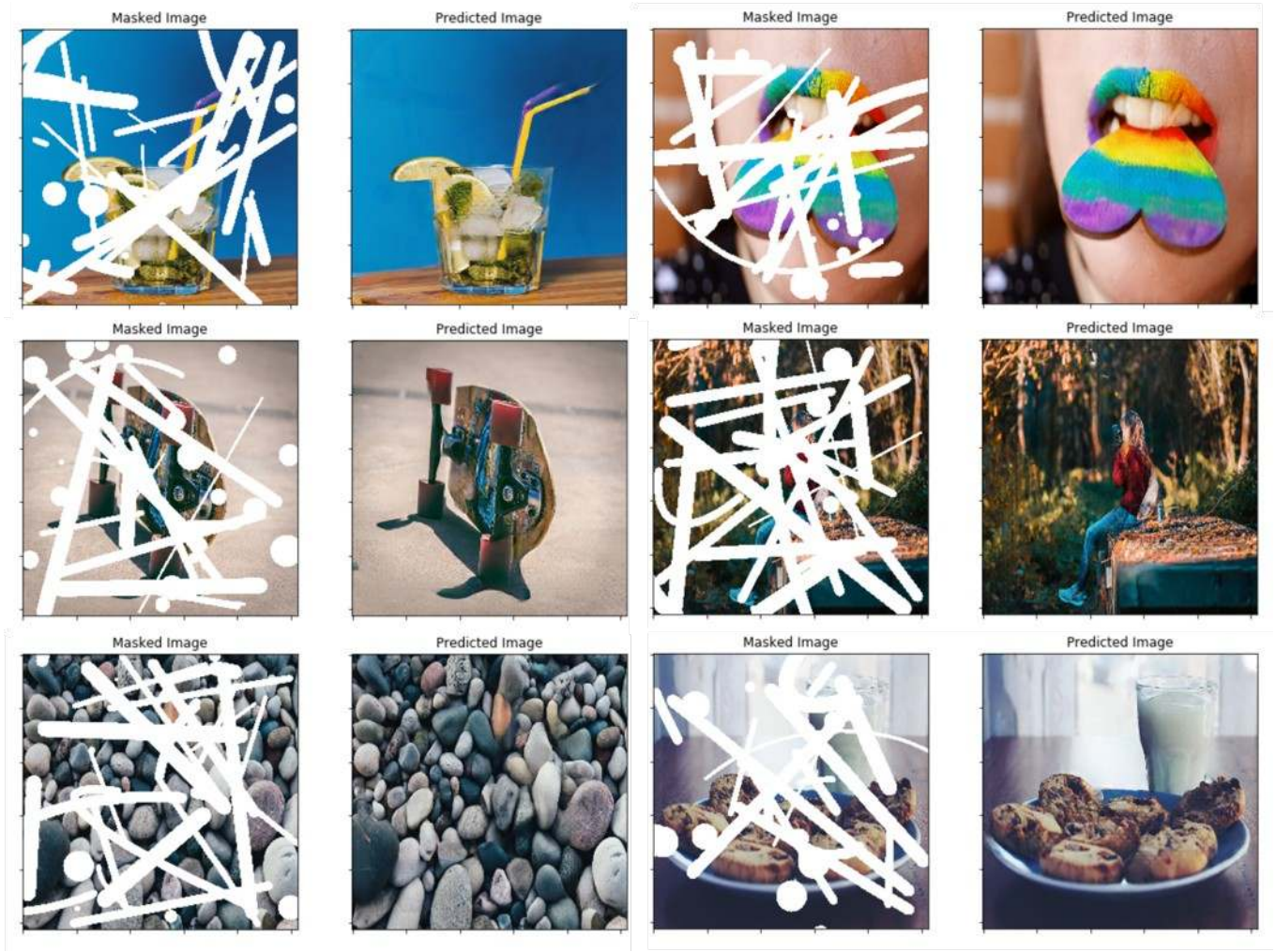
Face Recognition

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

https://github.com/deepinsight/insightface
https://arxiv.org/abs/1801.07698
January 2018

Keras

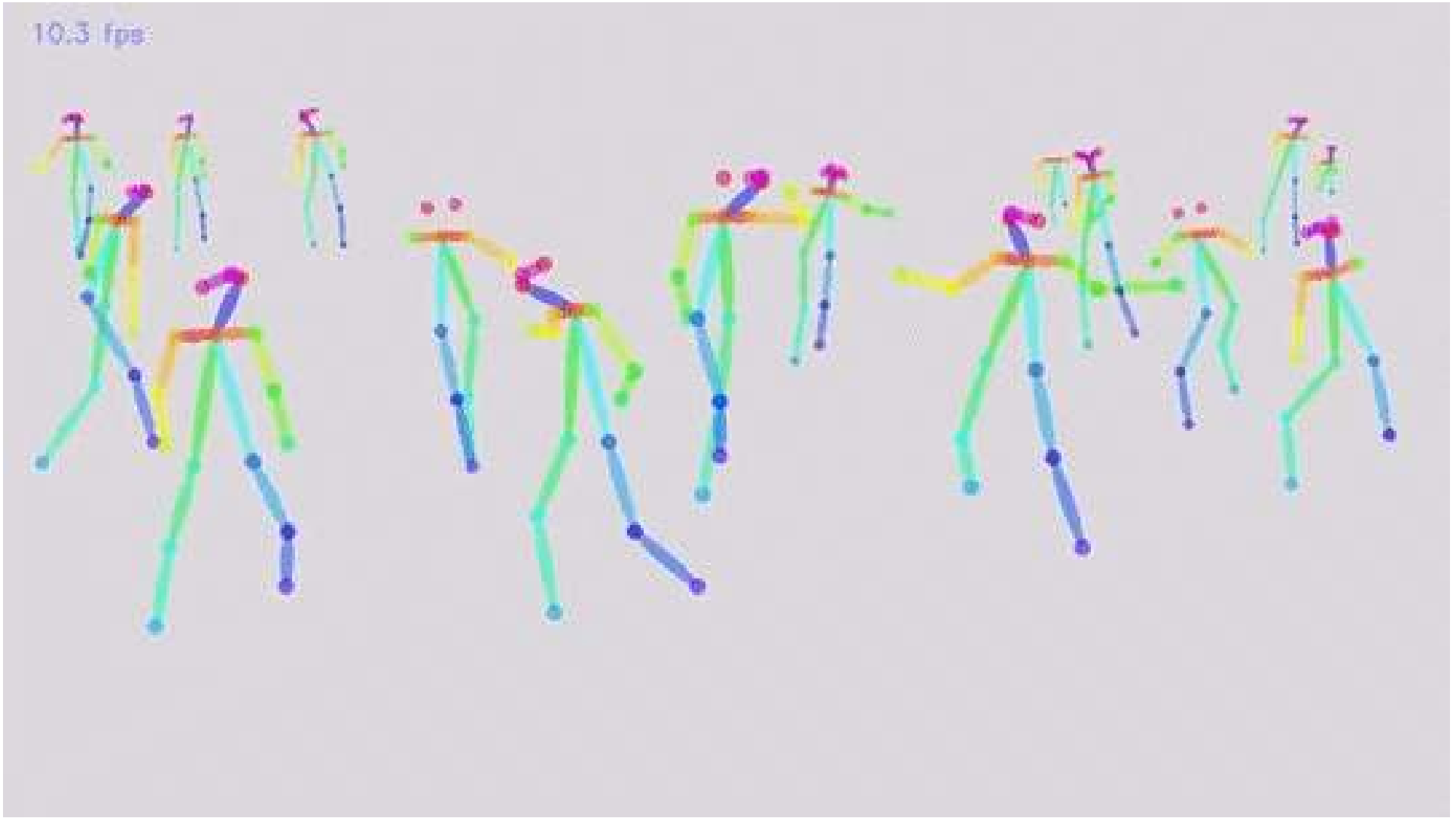
Image Inpainting



https://github.com/MathiasGruber/PConv-Keras
https://arxiv.org/abs/1804.07723
April 2018

aws

Real-Time Pose Estimation

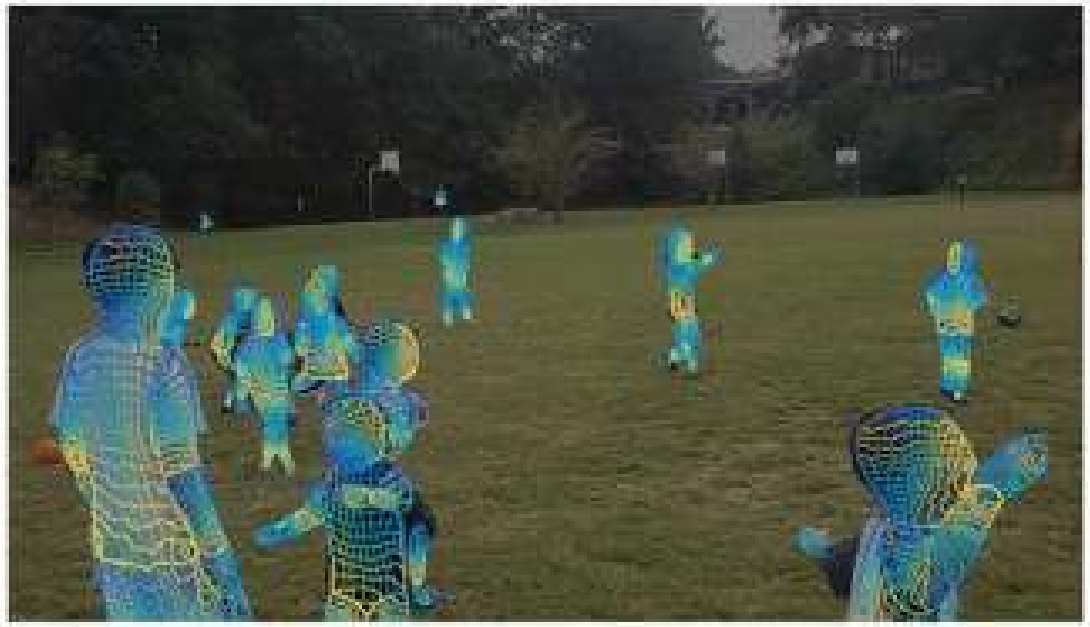


https://github.com/dragonfly90/mxnet_Realtime_Multi-Person_Pose_Estimation

November 2016

Caffe 2

Real-Time Pose Estimation: DensePose



https://github.com/facebookresearch/DensePose

February 2018

aws

Recurrent Neural Networks

Recurrent Neural Networks (RNN)

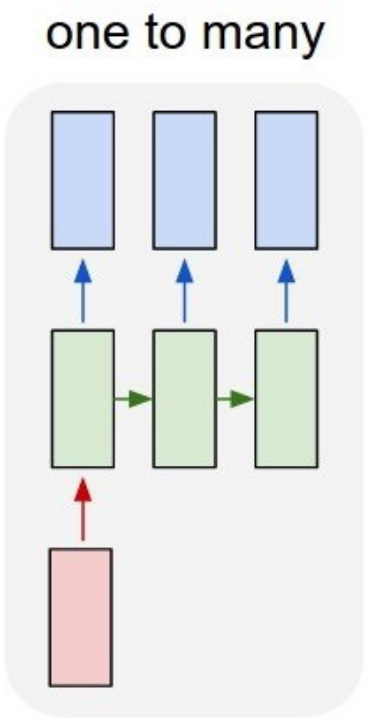
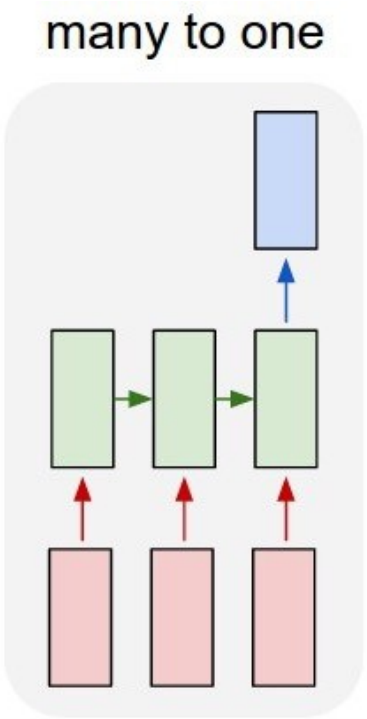
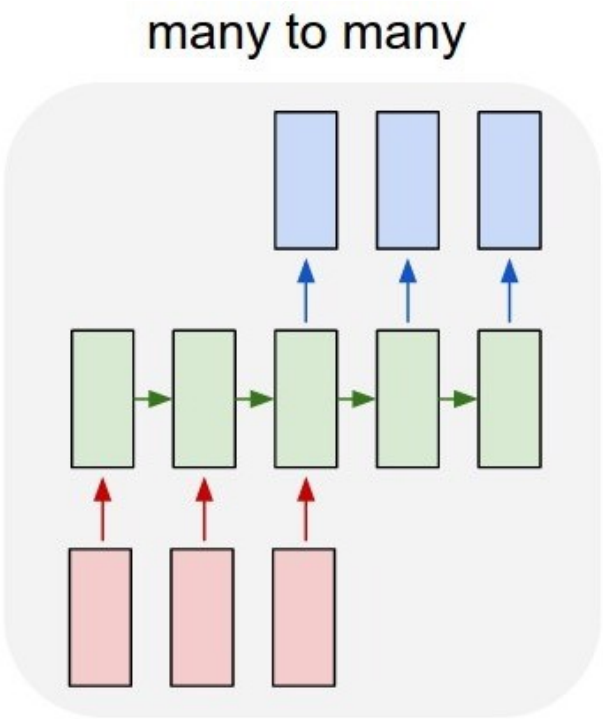


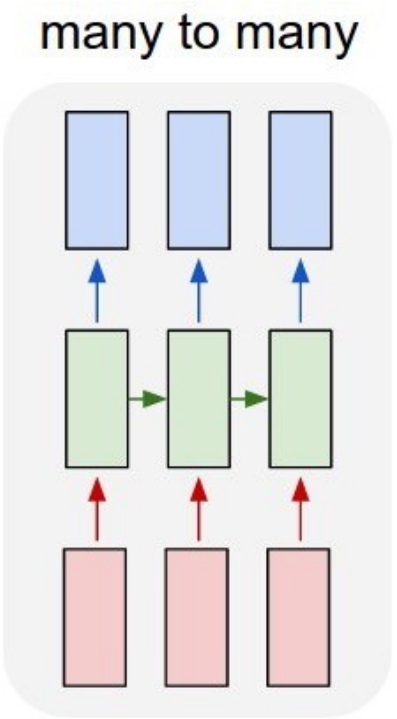
Image captioning



Sentiment analysis



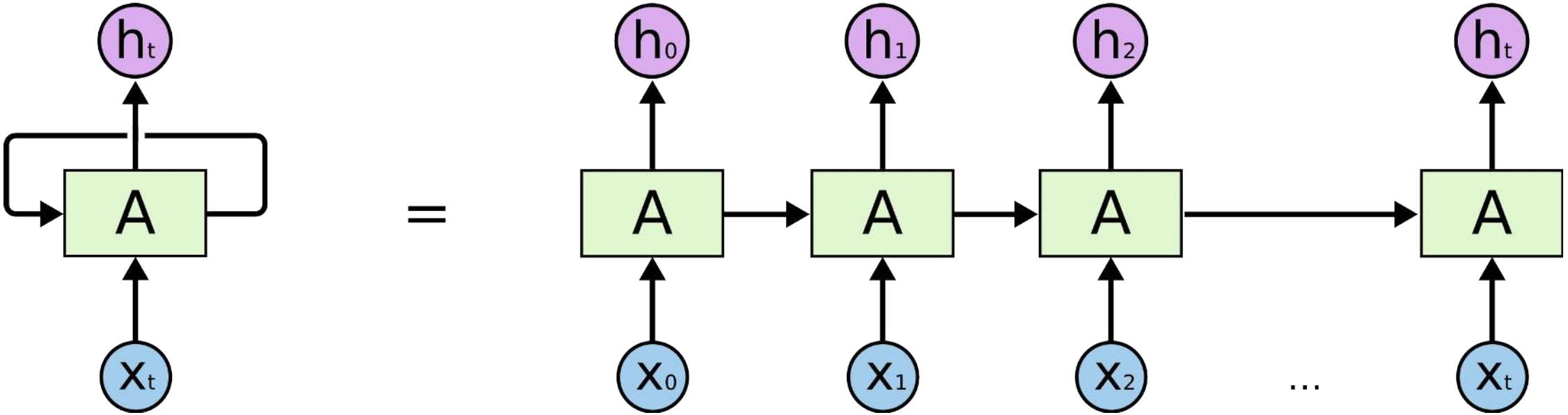
Machine translation



Video frame labeling

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Recurrent Neural Networks

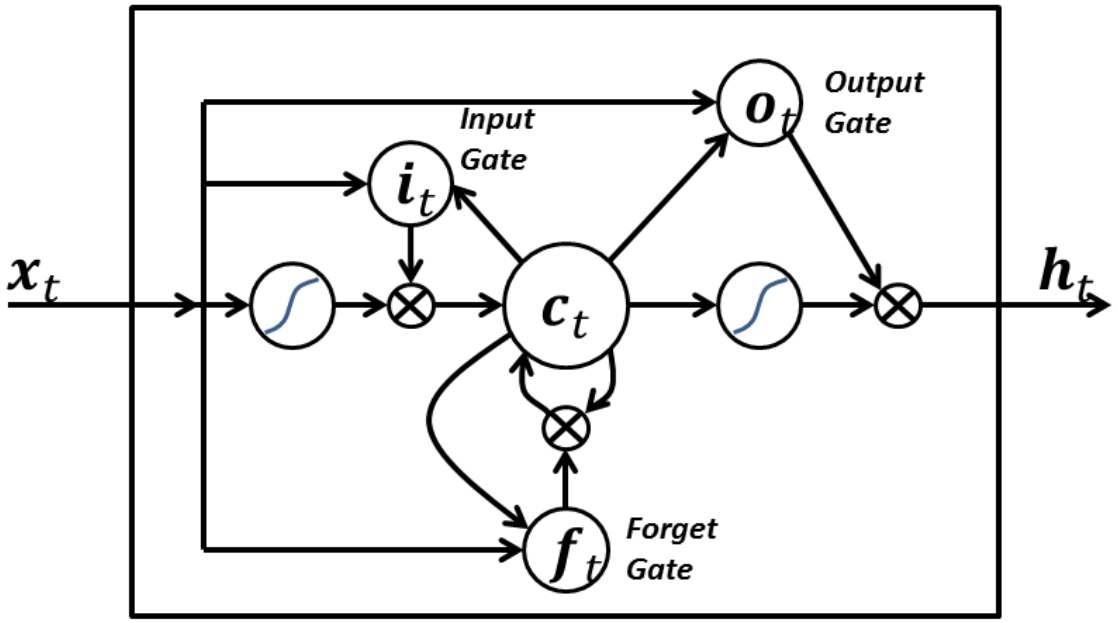


http://colah.github.io/posts/2015-08-Understanding-LSTMs/

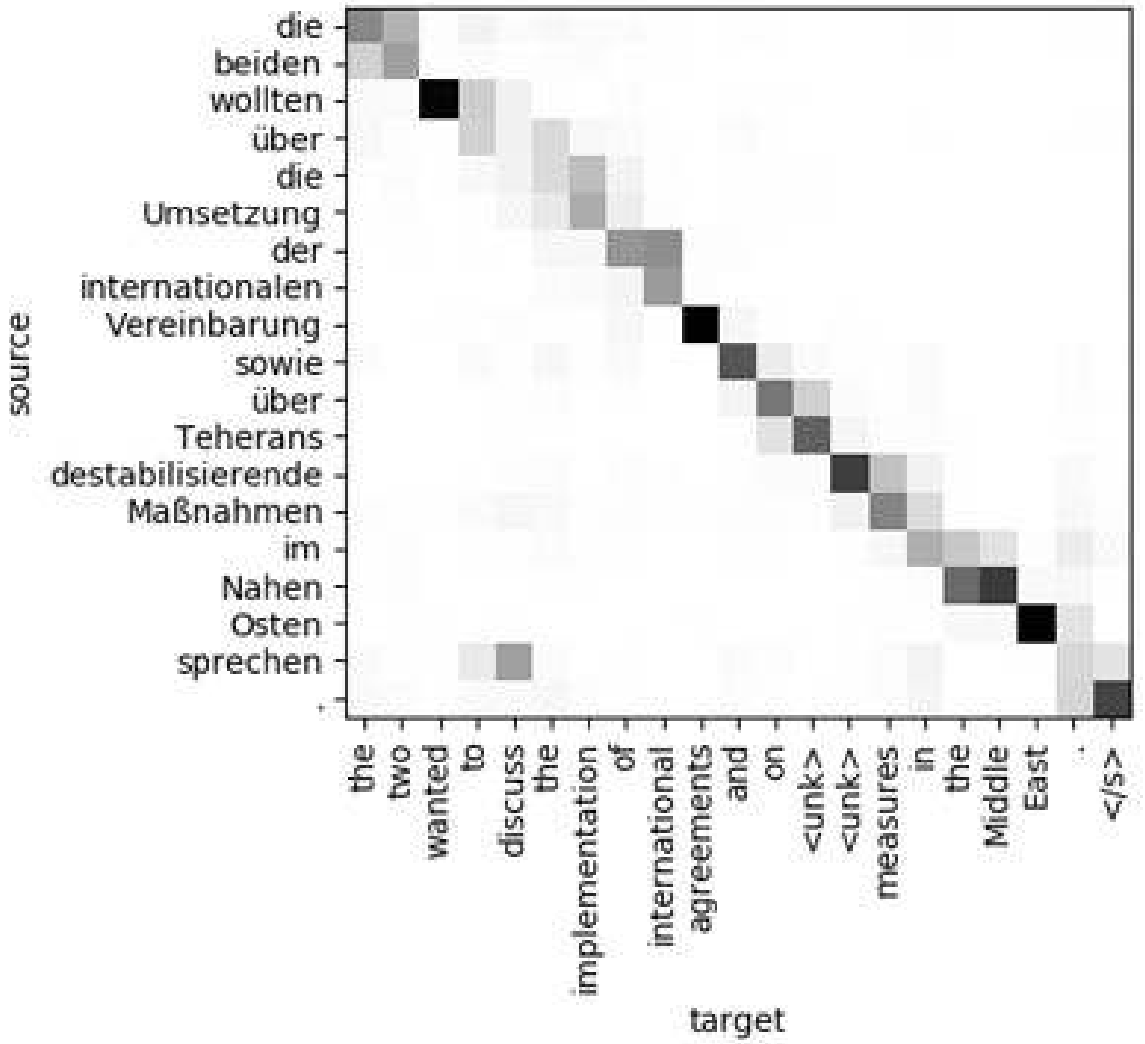
Long Short Term Memory Networks (LSTM)

Hochreiter and Schmidhuber,1997

- A LSTM neuron computes the output based on the input and a previous state
- LSTM neurons have « short-term memory »
- They do a better job than RNN at predicting longer sequences of data (i.e. hundreds of steps)

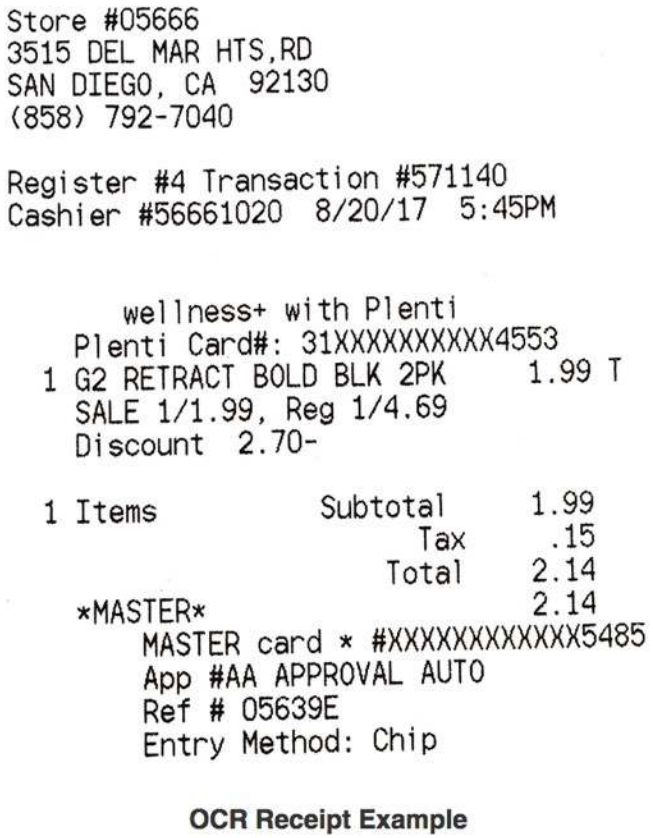


Machine Translation – AWS Sockeye



https://github.com/awslabs/sockeye

OCR – Tesseract 4.0 (beta)



```
Store #05666
3515 DEL MAR HTS,RD
SAN DIEGO, CA  92130
(858) 792-7040

Register #4 Transaction #571140
Cashier #56661020  8/20/17  5:45PM


     wellness+ with Plenti
  Plenti Card#: 31XXXXXXXXXX4553
1 G2 RETRACT BOLD BLK 2PK      1.99 T
  SALE 1/1.99, Reg 1/4.69
  Discount  2.70-

1 Items          Subtotal      1.99
                     Tax        .15
                   Total       2.14
  *MASTER*                     2.14
      MASTER card * #XXXXXXXXXXXX5485
      App #AA APPROVAL AUTO
      Ref # 05639E
      Entry Method: Chip
```

OCR Receipt Example

Output

Store #056663515
DEL MAR HTS,RD
SAN DIEGO, CA 92130
(858) 792-7040Register #4 Transaction
#571140
Cashier #56661020 8/20/17
5:45PMwellnesst+ with Plenti
Plenti Card#: 31XXXXXXXXXX4553
1 G2 RETRACT BOLD BLK 2PK 1.99 T
SALE 1/1.99, Reg 1/4.69
Discount 2.70-

1 Items Subtotal 1.99
Tax .15

Total 2.14
xMASTER 2.14
MASTER card * #XXXXXXXXXXXX548S
Apo #AA APPROVAL AUTO
Ref # 05639E
Entry Method: Chip

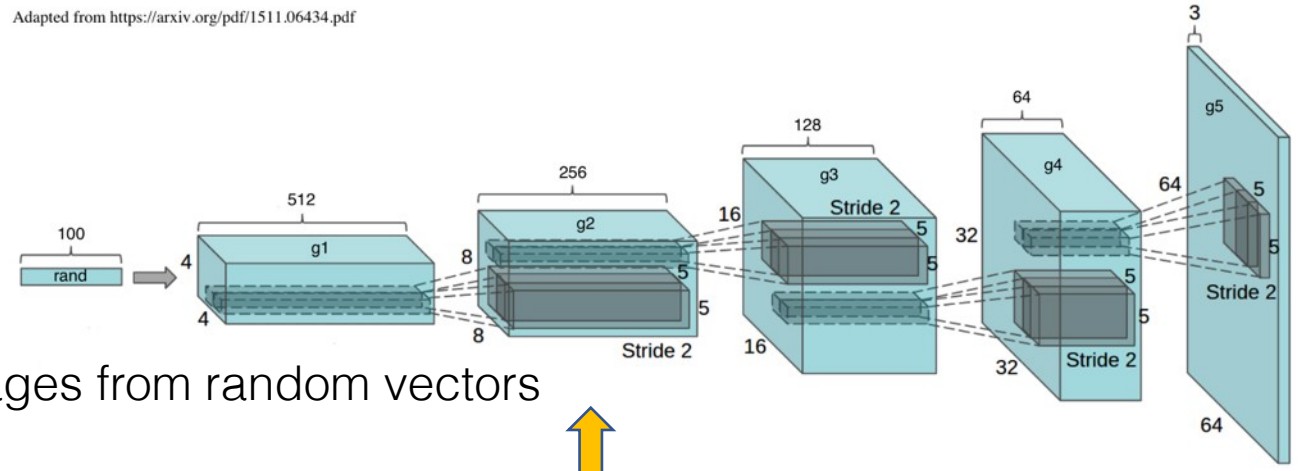
https://github.com/tesseract-ocr/tesseract/wiki/NeuralNetsInTesseract4.00https://www.learnopencv.com/deep-learning-based-text-recognition-ocr-using-tesseract-and-opencv/

aws

Generative Adversarial Networks

Generative Adversarial Networks

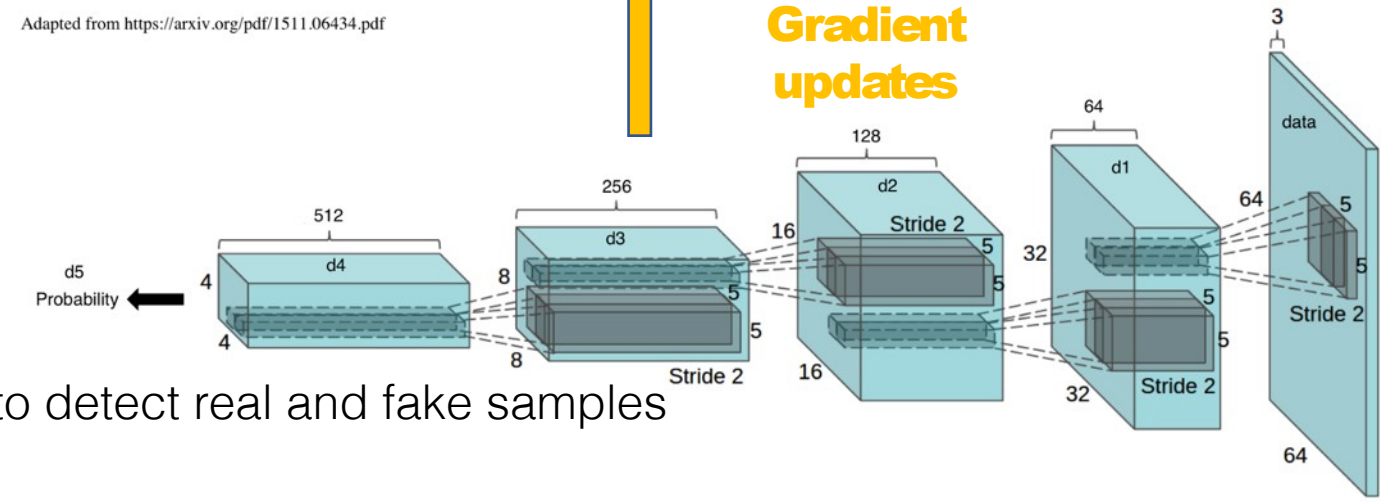
Goodfellow, 2014 https://arxiv.org/abs/1406.2661



Generator

Building images from random vectors

Gradient updates



Detector

Learning to detect real and fake samples

Fake images

Real images

https://medium.com/@julsimon/generative-adversarial-networks-on-apache-mxnet-part-1-b6d39e6b5df1

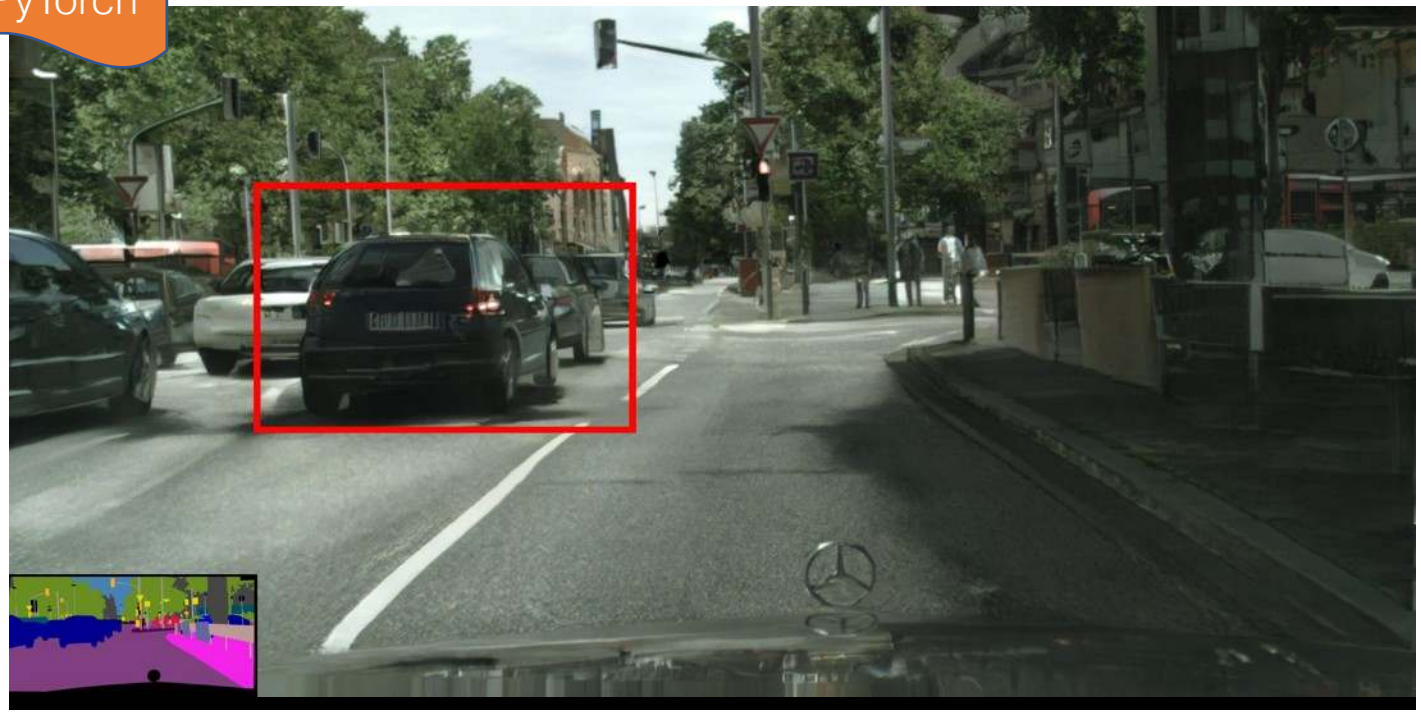
aws

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

TF

Generating new ”celebrity” faceshttps://github.com/tkarras/progressive_growing_of_gans
April 2018

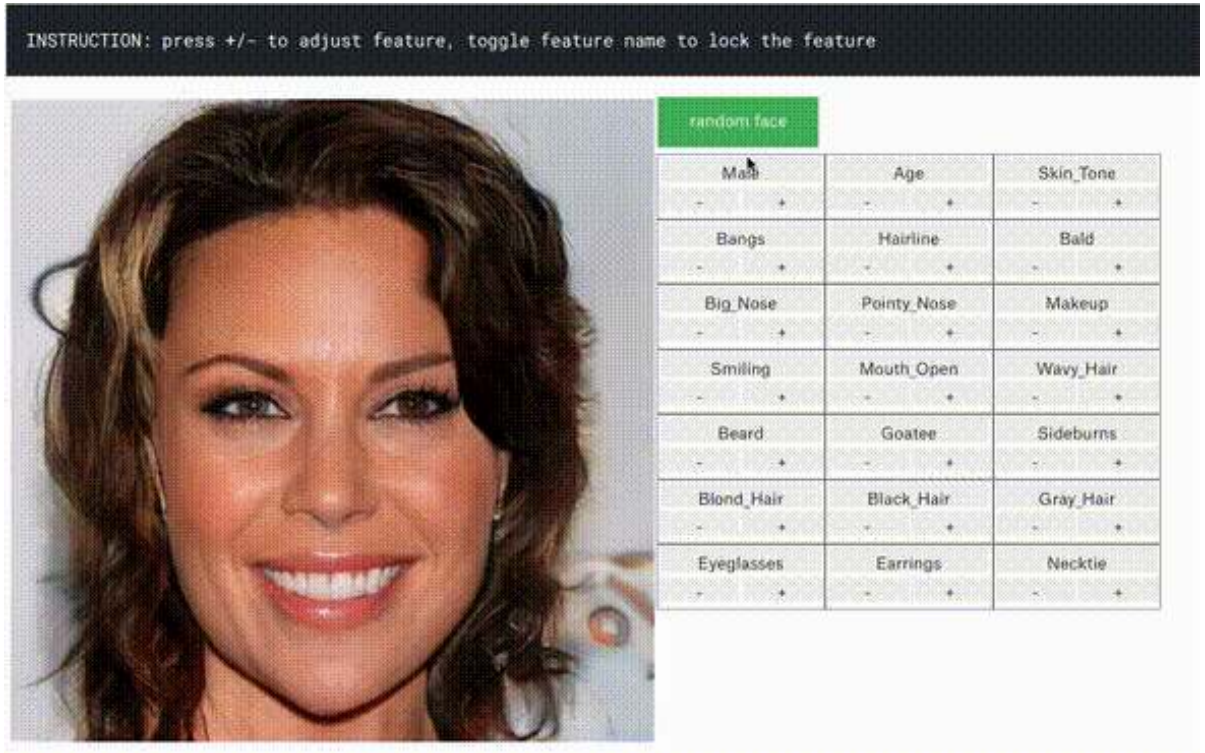
PyTorch

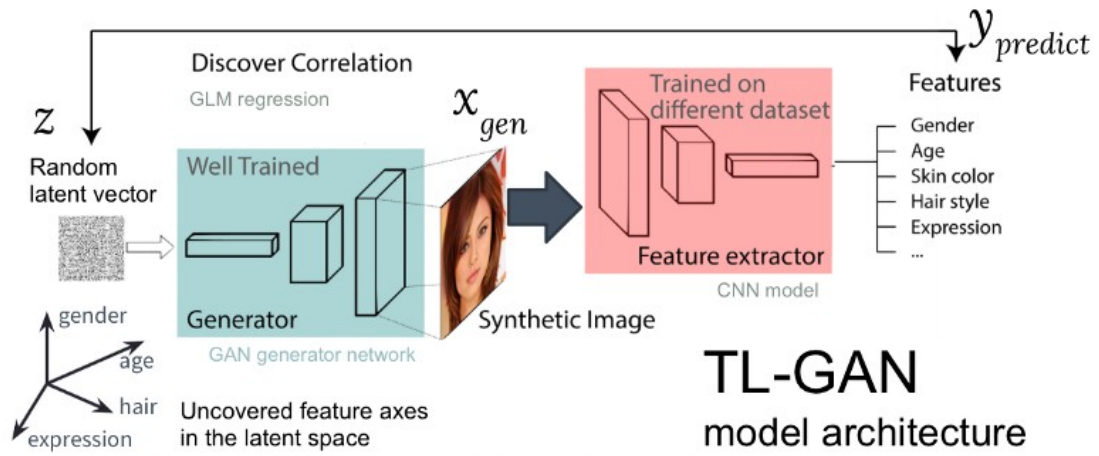


From semantic map to 2048x1024 picture
https://tcwang0509.github.io/pix2pixHD/
November 2017

aws

Controlled Image Generation with TL-GAN

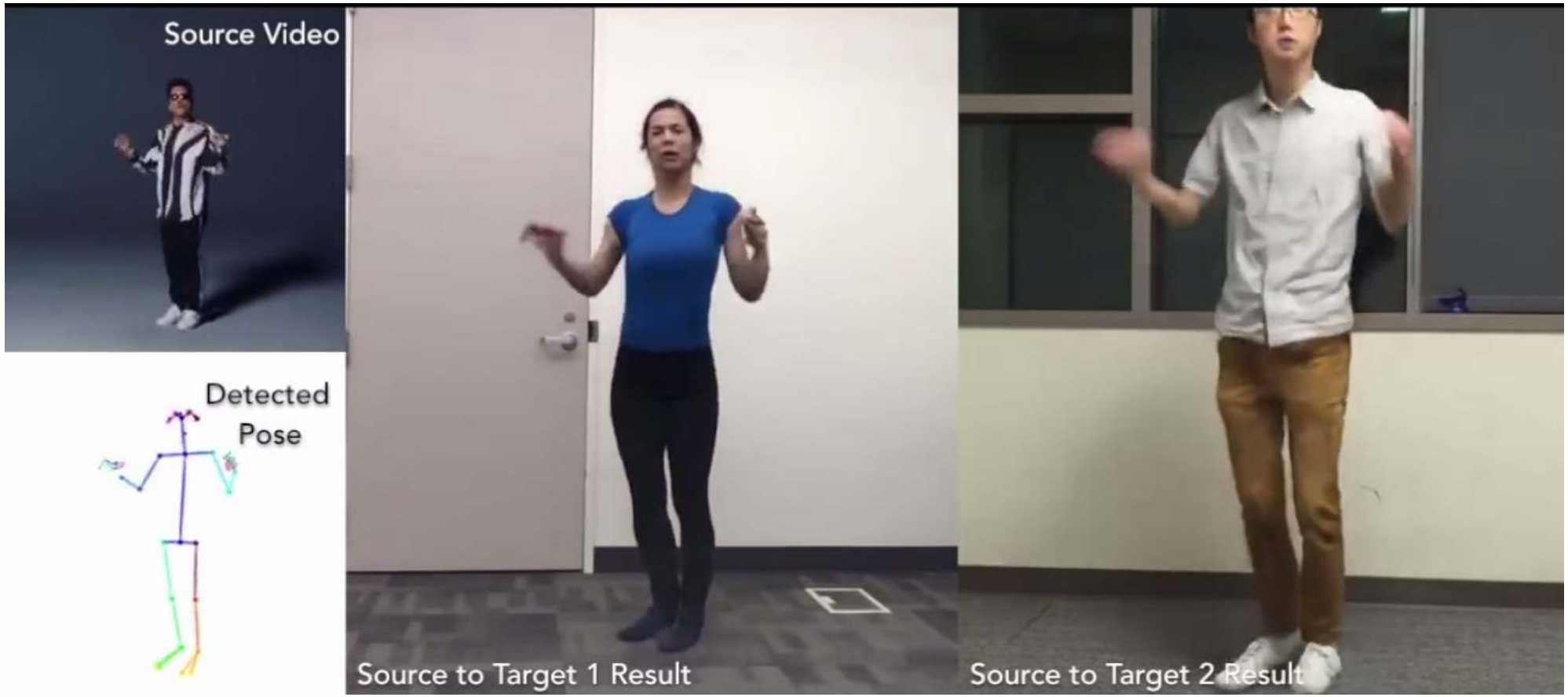




https://github.com/SummitKwan/transparent_latent_gan

October 2018

GAN: Everybody dance now



https://arxiv.org/abs/1808.07371
https://www.youtube.com/watch?v=PCBTZh41Ris
August 2018

Getting started

Resources

https://deeplearning.ai
https://fast.ai

http://www.deeplearningbook.org/
https://gluon.mxnet.io
https://keras.io

https://medium.com/@julsimon
https://gitlab.com/juliensimon/{aws,dlnotebooks}

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.

Thank you!

Julien Simon

Principal Technical Evangelist, AI and Machine Learning, AWS

@julsimon