Deep Dive on Deep Learning

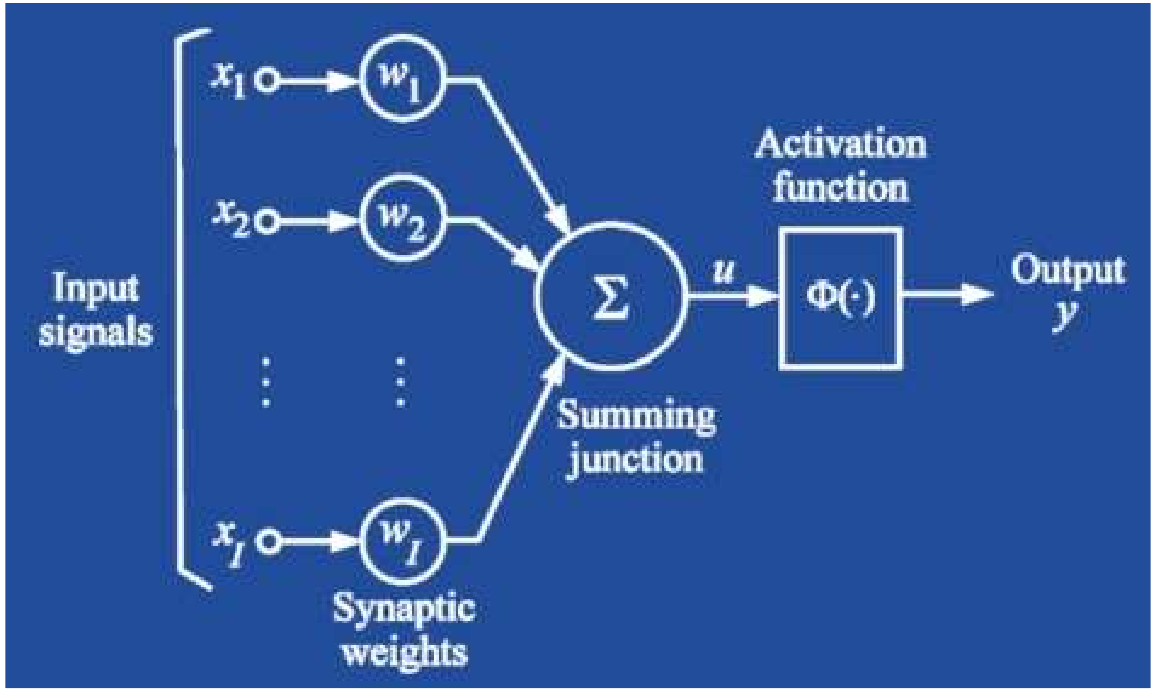
Julien Simon
Principal Technical Evangelist, AI and Machine Learning
@julsimon

Agenda

- Deep Learning concepts
- Common architectures and use cases
- Apache MXNet
- Infrastructure for Deep Learning
- Demos along the way: MXNet, Gluon, Keras, TensorFlow, PyTorch ☺

Deep Learning concepts

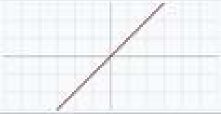
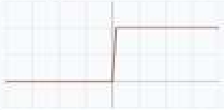
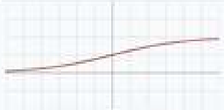
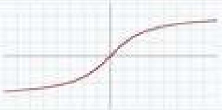
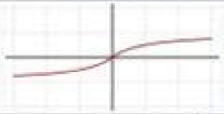
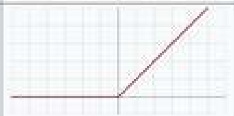
The neuron



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

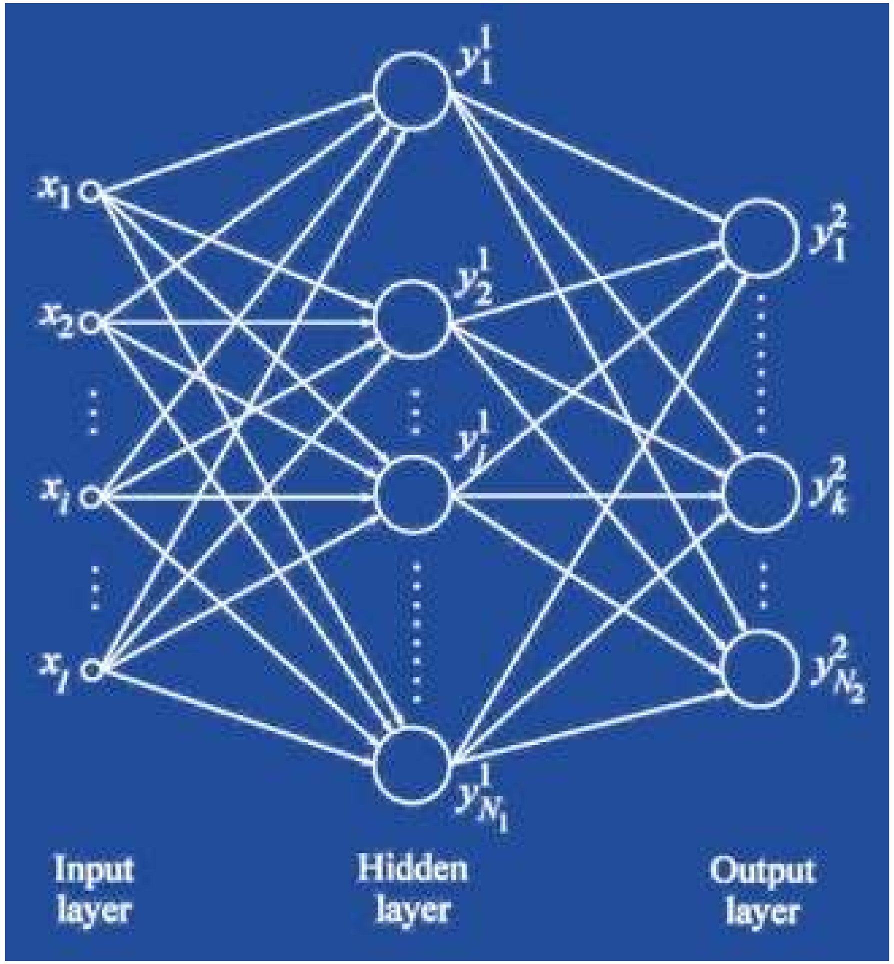
"Multiply and Accumulate"

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

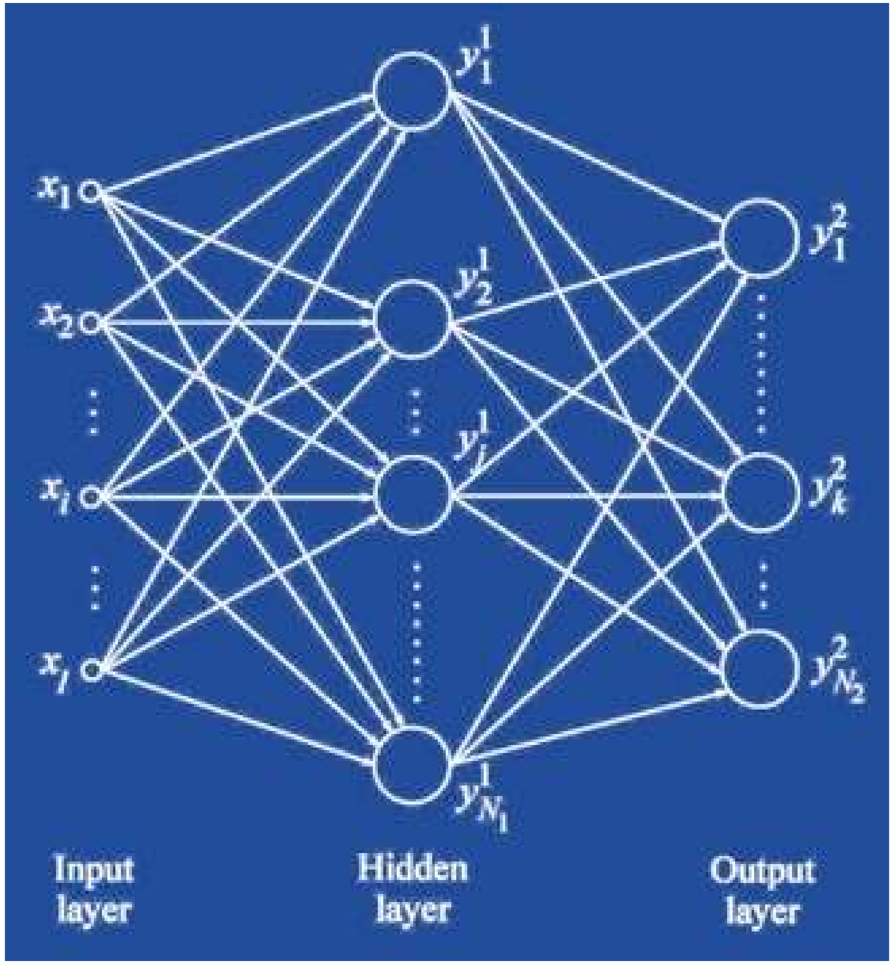


$$x = \overbrace{\begin{bmatrix} x_{11}, x_{12}, \dots x_{1l} \\ x_{21}, x_{22}, \dots x_{2l} \\ \dots \; \dots \; \dots \\ x_{m1}, x_{m2}, \dots x_{ml} \end{bmatrix}}^{\text{l features}} \quad \text{m samples}$$

$$y = \begin{bmatrix} 2 \\ 0 \\ \dots \\ 4 \end{bmatrix} \qquad \underbrace{\begin{bmatrix} 0,0,1,0,0,\dots,0 \\ 1,0,0,0,0,\dots,0 \\ \dots \\ 0,0,0,0,1,\dots,0 \end{bmatrix}}_{\text{One-hot encoding}} \quad \text{m labels, } N_2 \text{ categories}$$

aws

Neural networks



l features

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

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

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

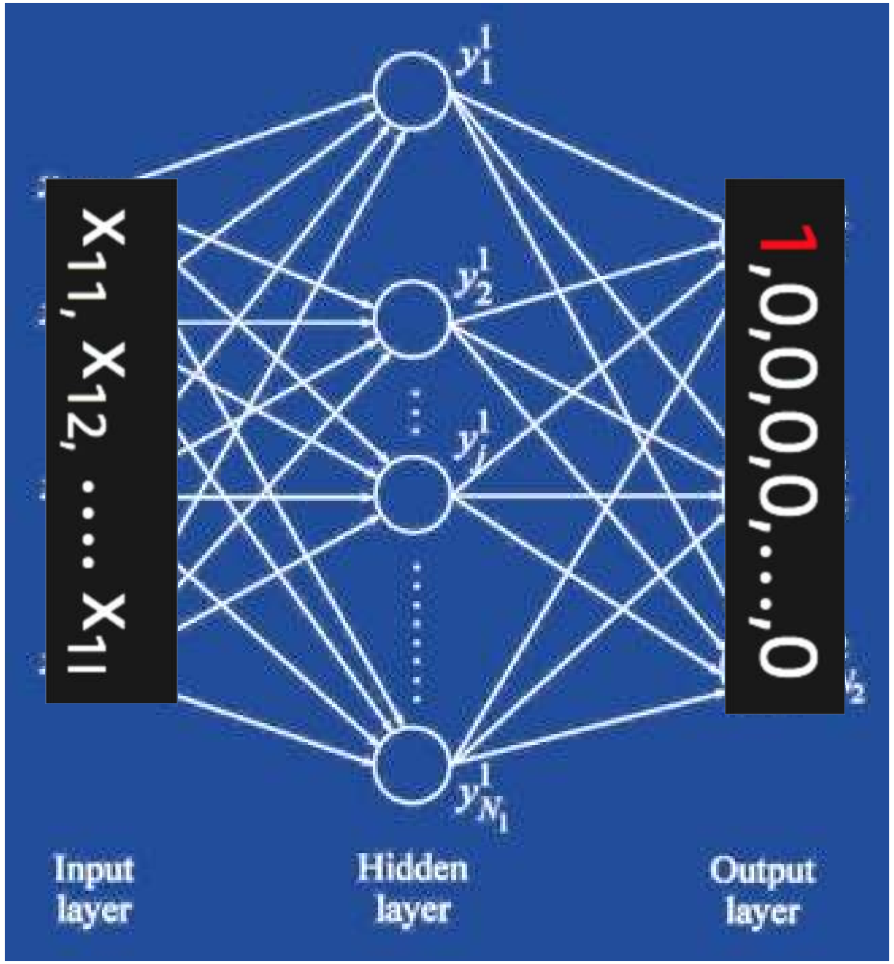
$$y = \begin{bmatrix} 2 \\ 0 \\ \ldots \\ 4 \end{bmatrix} \quad \begin{bmatrix} 0,0,1,\ \ldots,0 \\ 1,0,0,\ \ldots,0 \\ \ldots \\ 0,0,0,\ \ldots,0 \end{bmatrix}$$ m labels, N_2 categories

One-hot encoding

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

aws

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

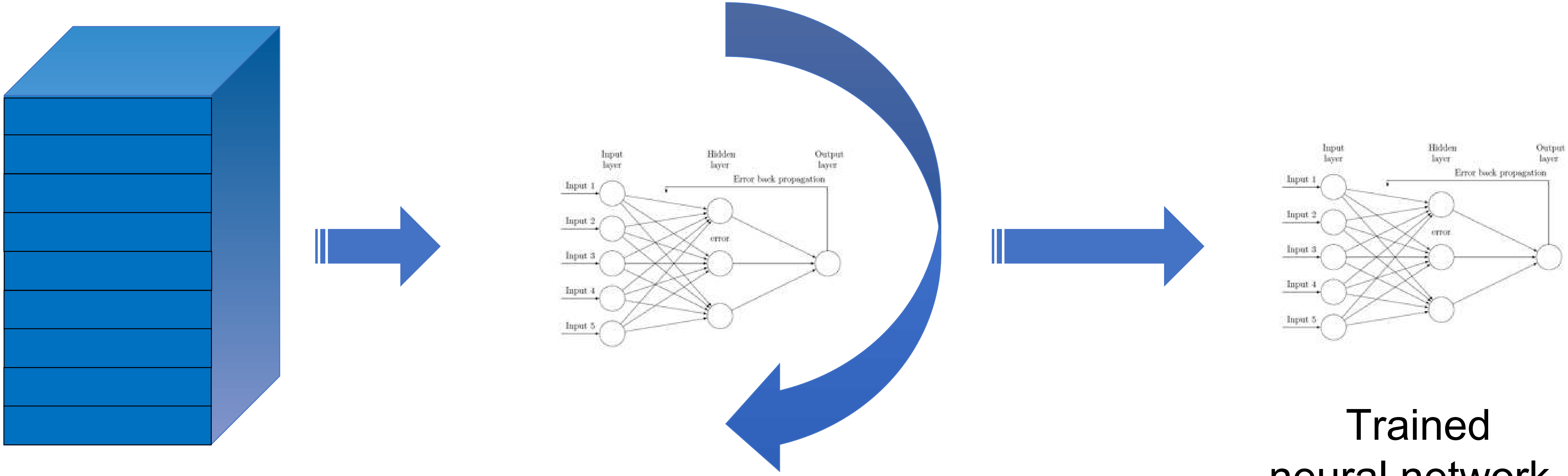
$error = loss(Y_1, Y'_1)$

For a batch of samples:

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

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

Training



Training data set

Backpropagation

Trained
neural network

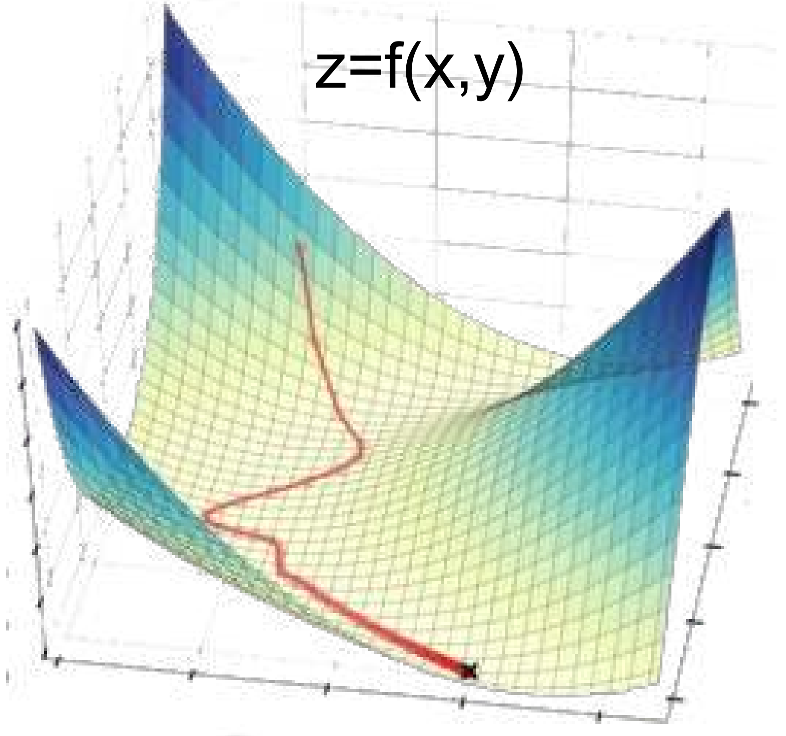
Batch size
Learning rate
Number of epochs

Hyper parameters

Stochastic Gradient Descent

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.

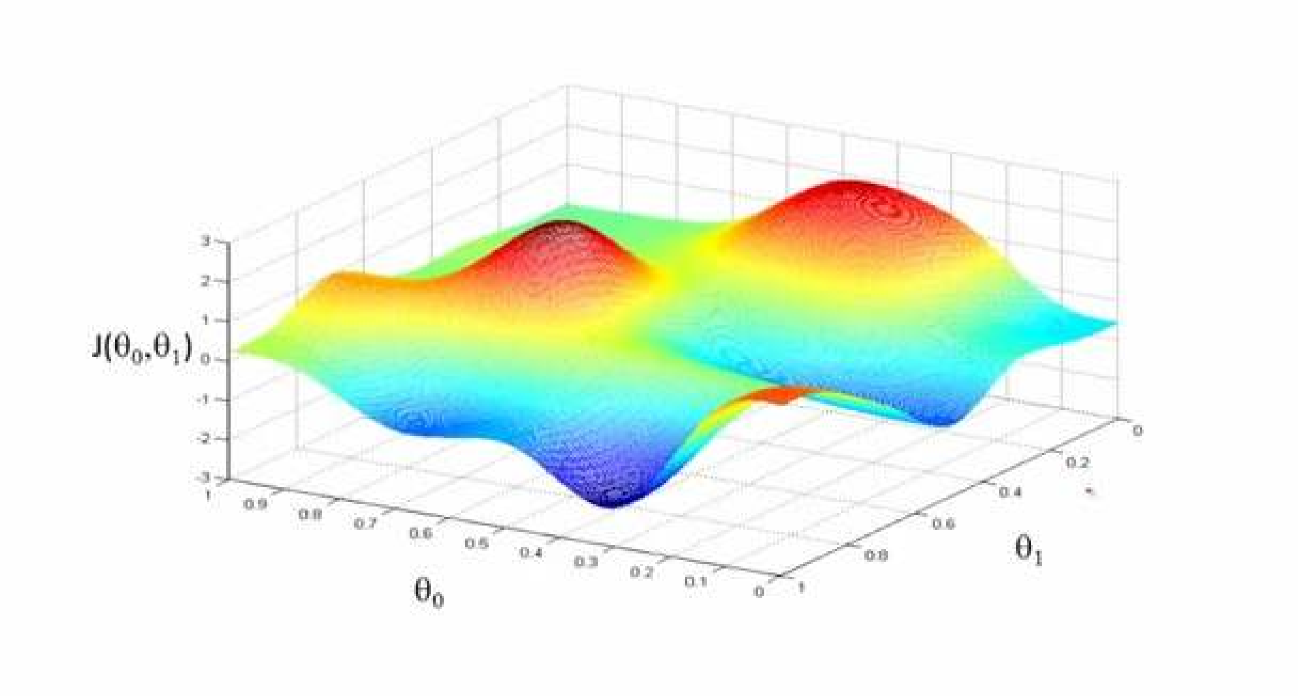
Tim Dettmers
University of Lugano
2015

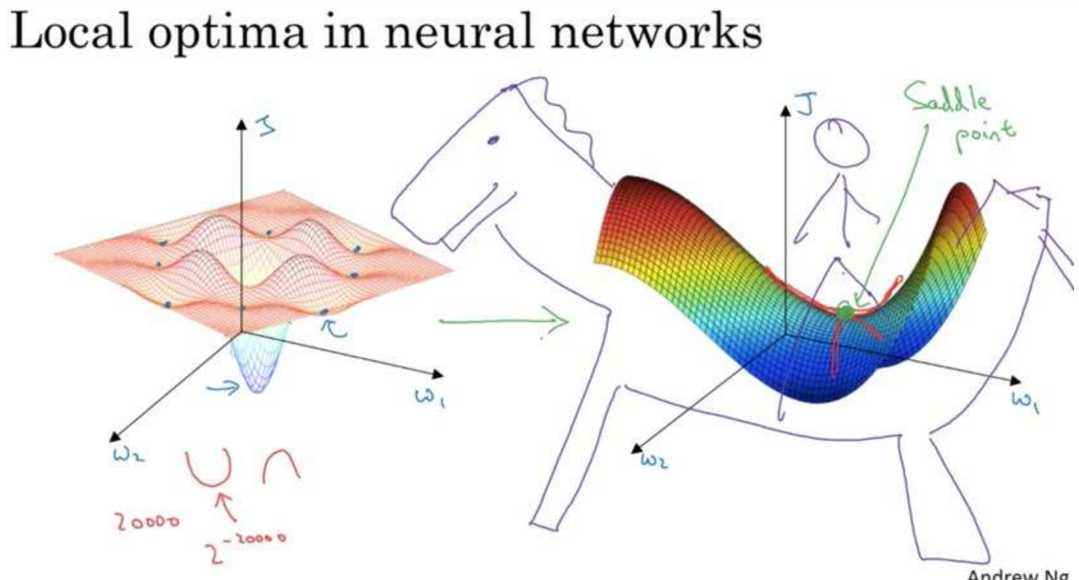


The « step size » depends on the learning rate

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

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. »*

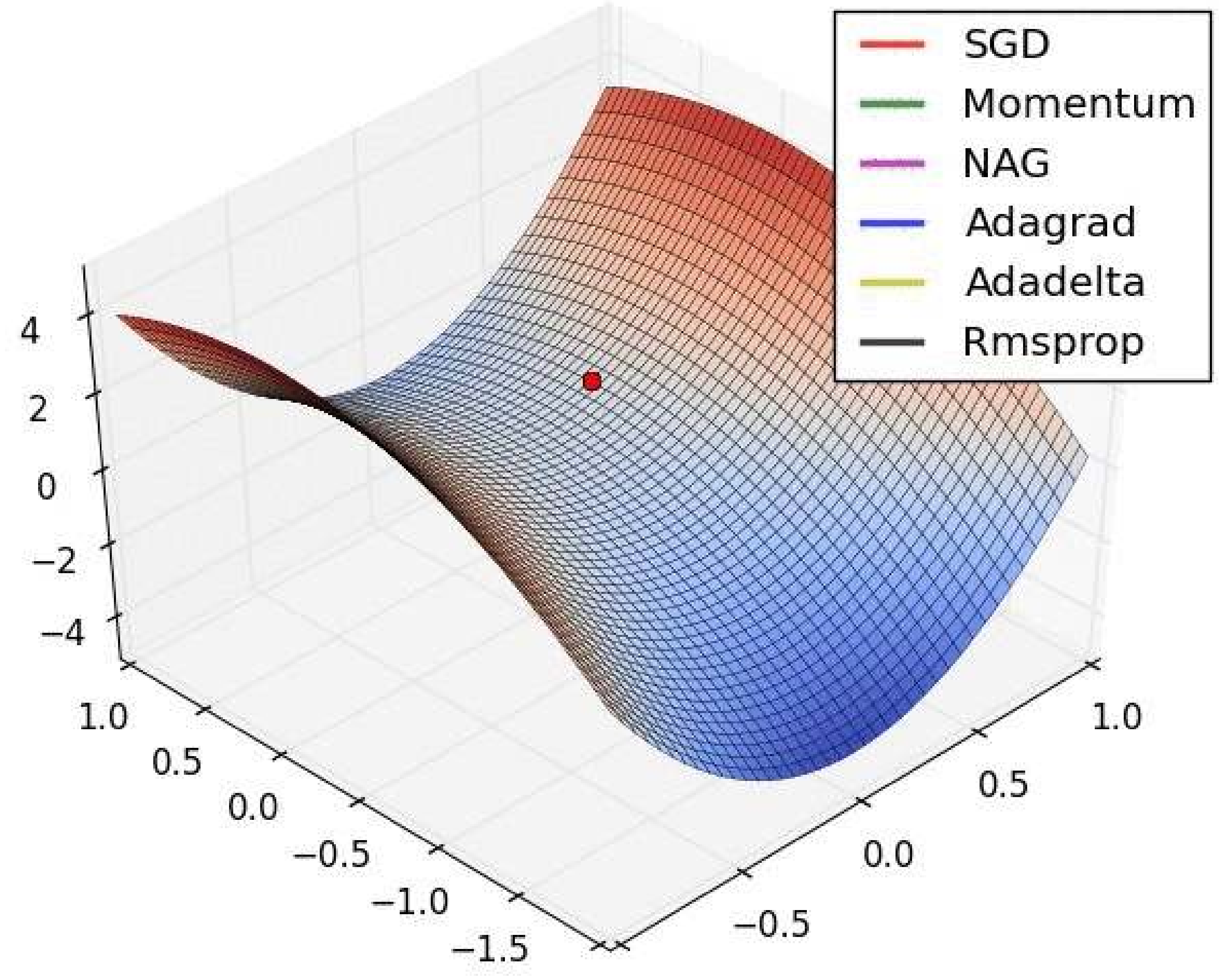
« Qualitatively characterizing neural network optimization problems », Goodfellow et al, 2015
https://arxiv.org/abs/1412.6544

Optimizers

SGD works remarkably well and is still widely used.

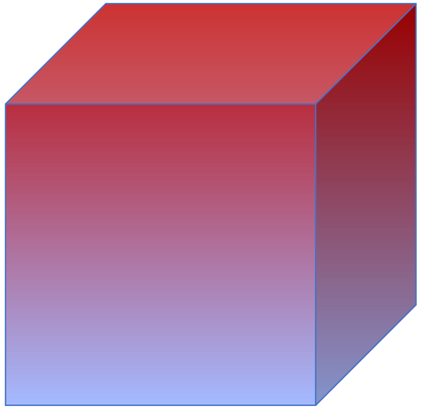
Adaptative optimizers use a variable learning rate.

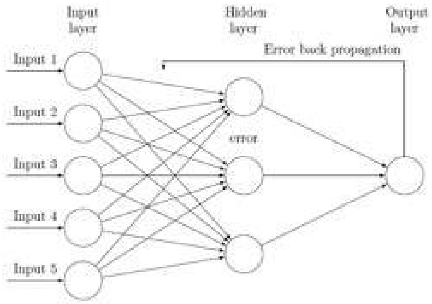
Some even use a learning rate per dimension (Adam).



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

Validation





Validation accuracy

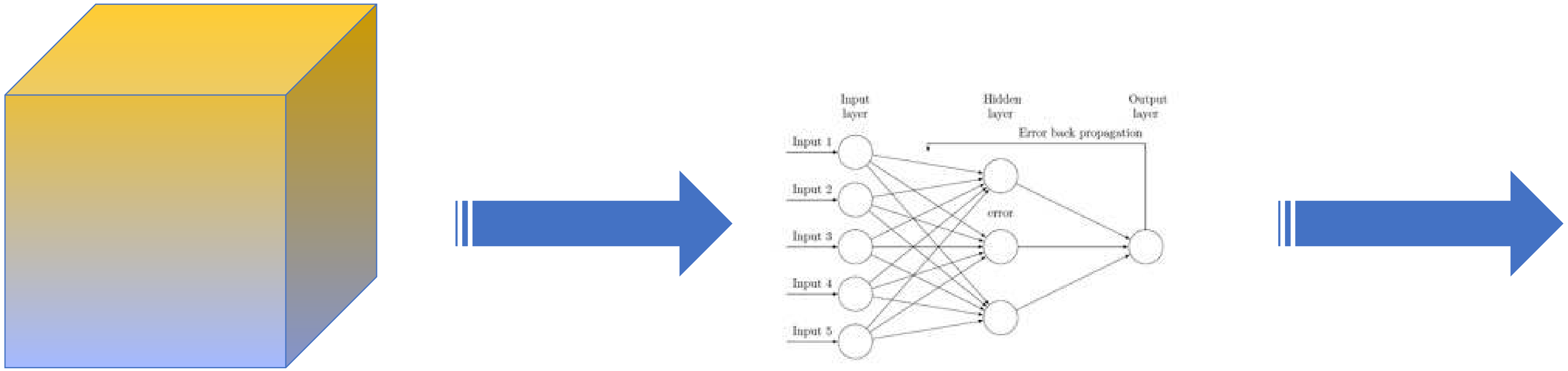
Validation data set
(also called dev set)

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 accuracy

Test data set

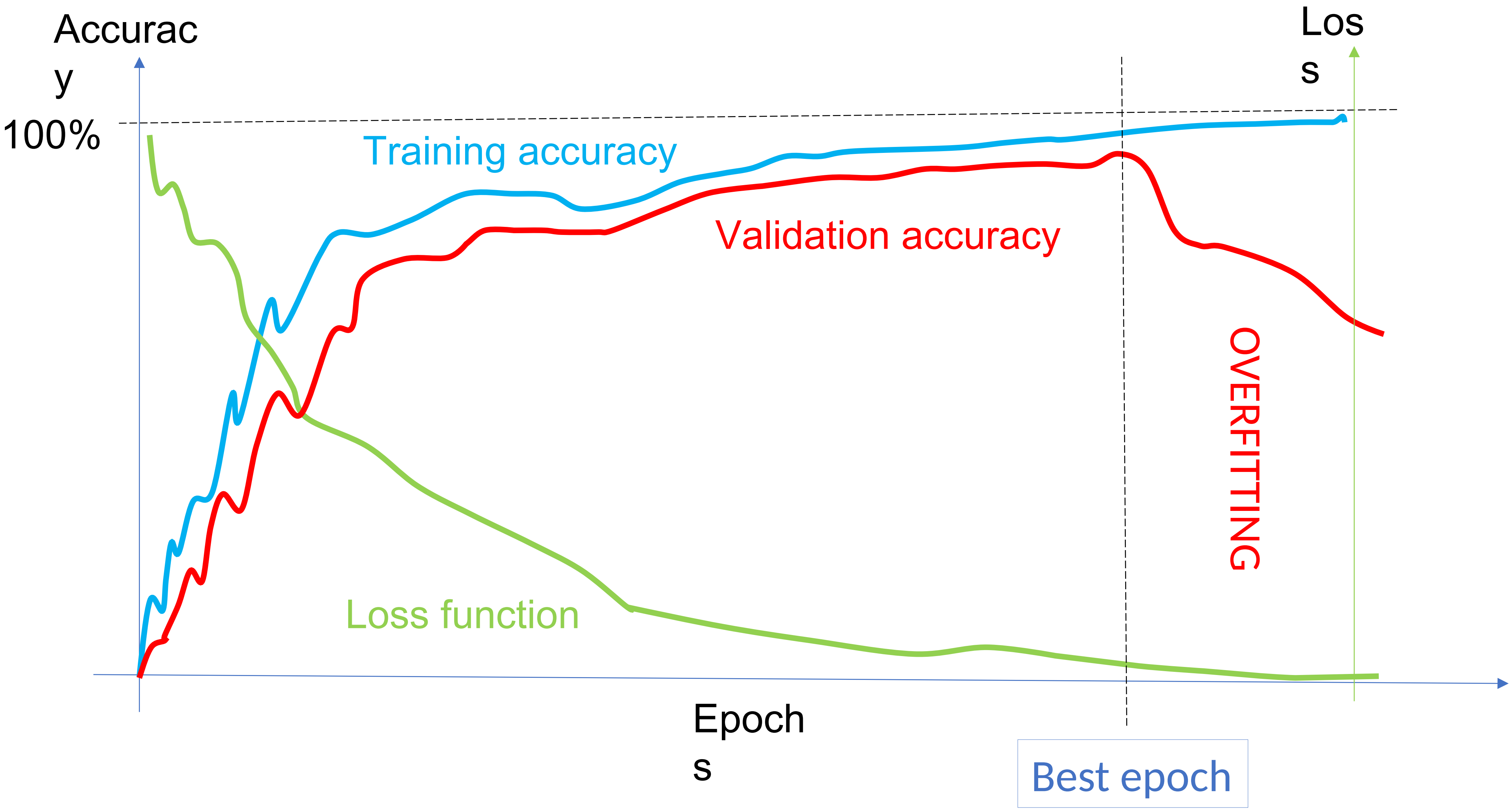
Fully trained neural network

Prediction at the end of experimentation

<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

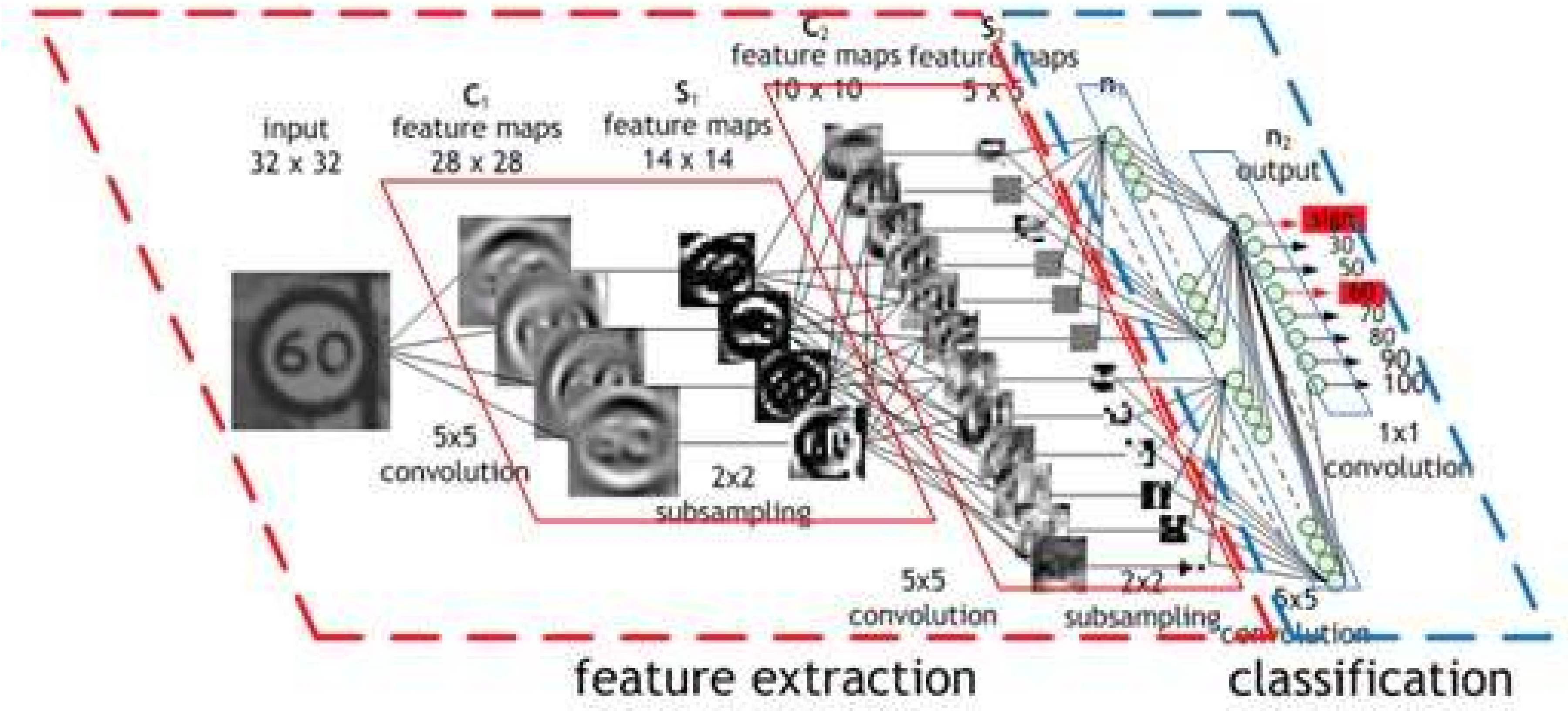
« *Deep Learning ultimately is about finding a minimum that generalizes well, with bonus points for finding one fast and reliably* », Sebastian Ruder



Common architectures and use cases

Convolutional Neural Networks (CNN)

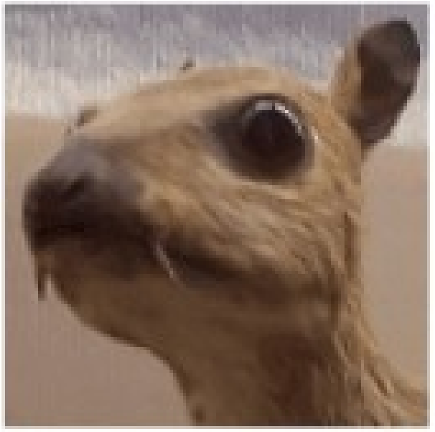
Le Cun, 1998: handwritten digit recognition, 32x32 pixels



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

Extracting features with convolution

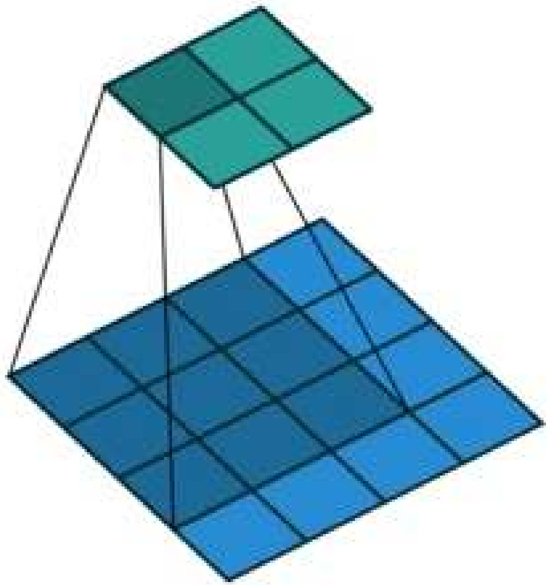
Input image



Convolution Kernel

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

Feature map

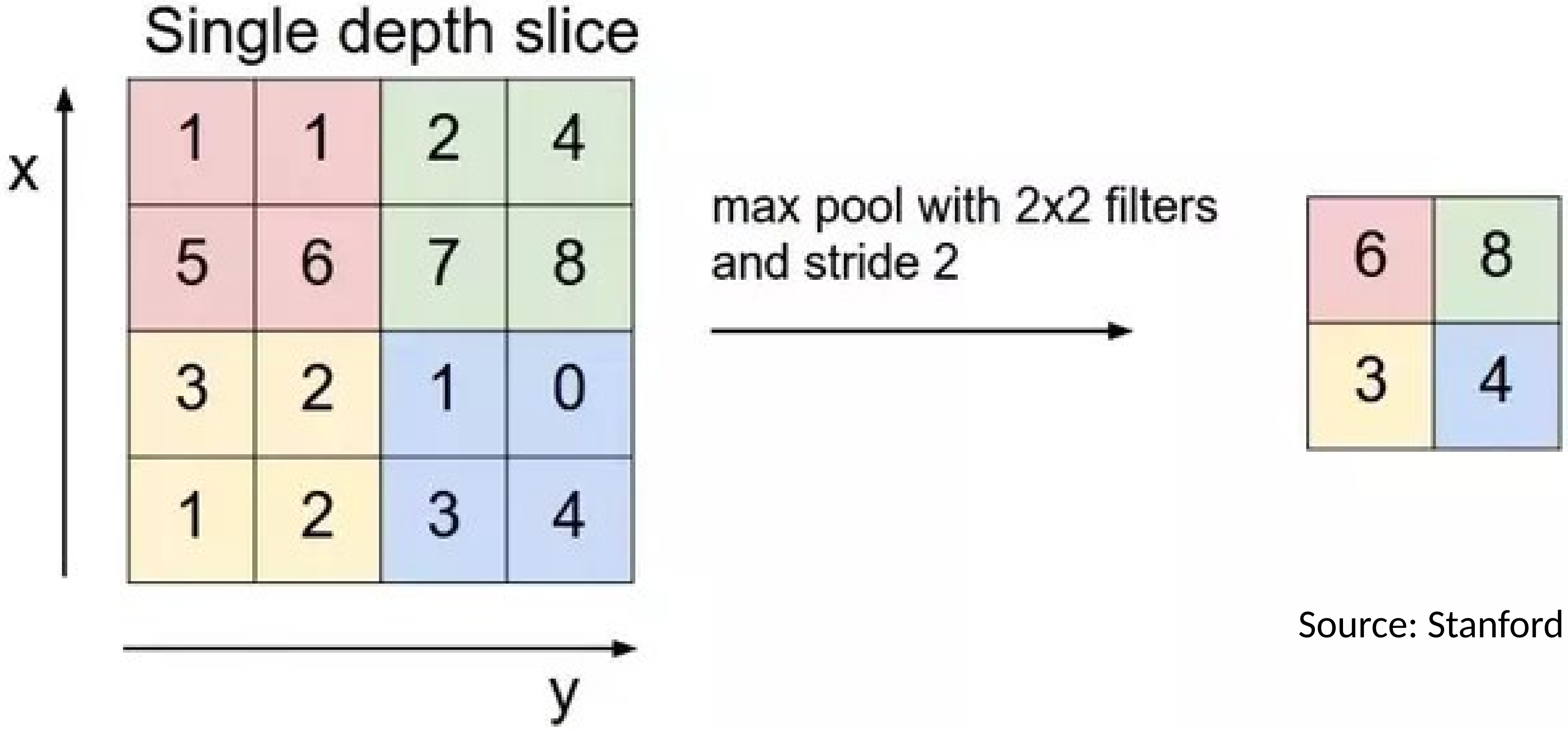


Source: http://timdettmers.com

Convolution extracts features automatically.

Kernel parameters are learned during the training process.

Downsampling images with pooling

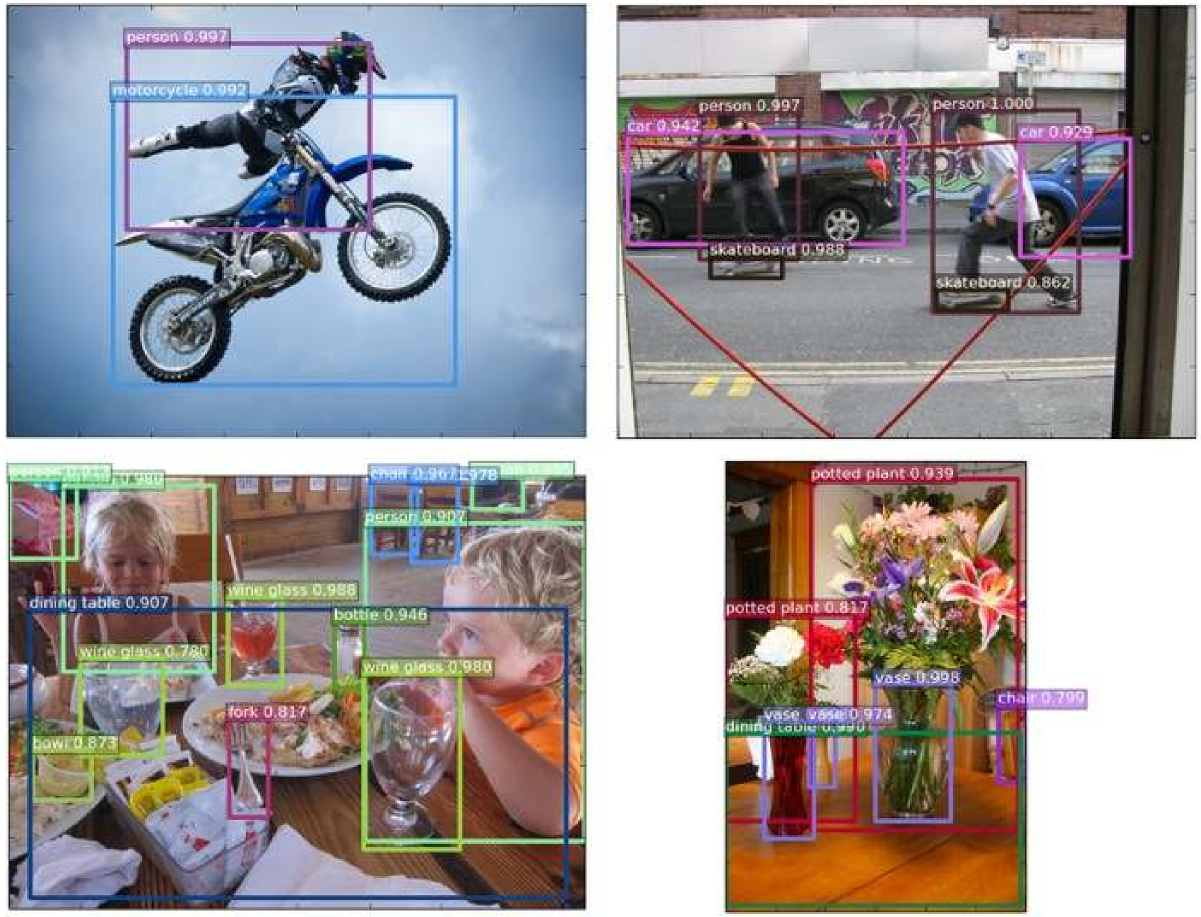


Source: Stanford University

Pooling shrinks images while preserving significant information.

MXNet

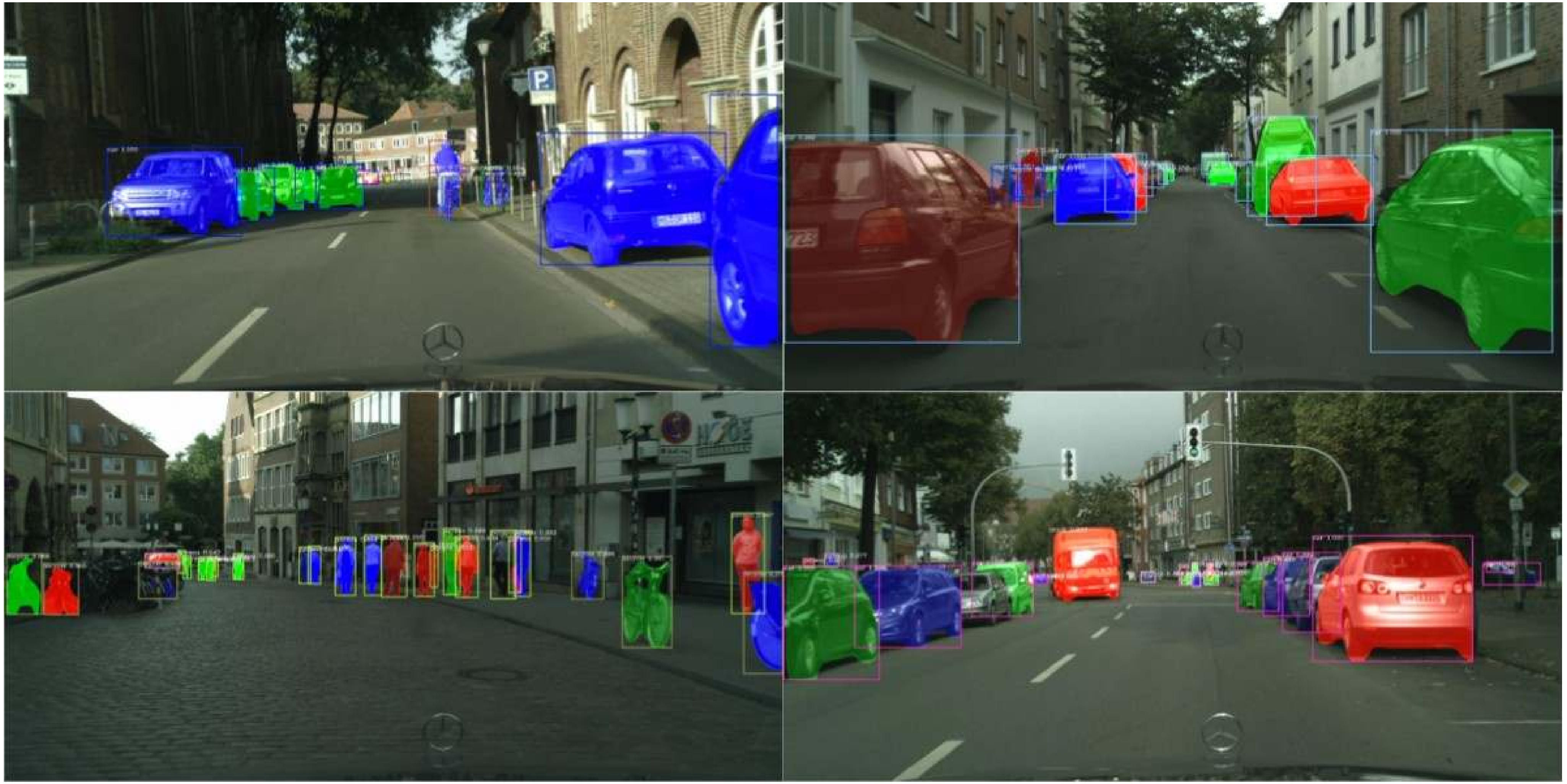
Object Detection



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

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

Object Segmentation



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

Text Detection and Recognition

https://github.com/Bartzi/stn-ocr

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
```

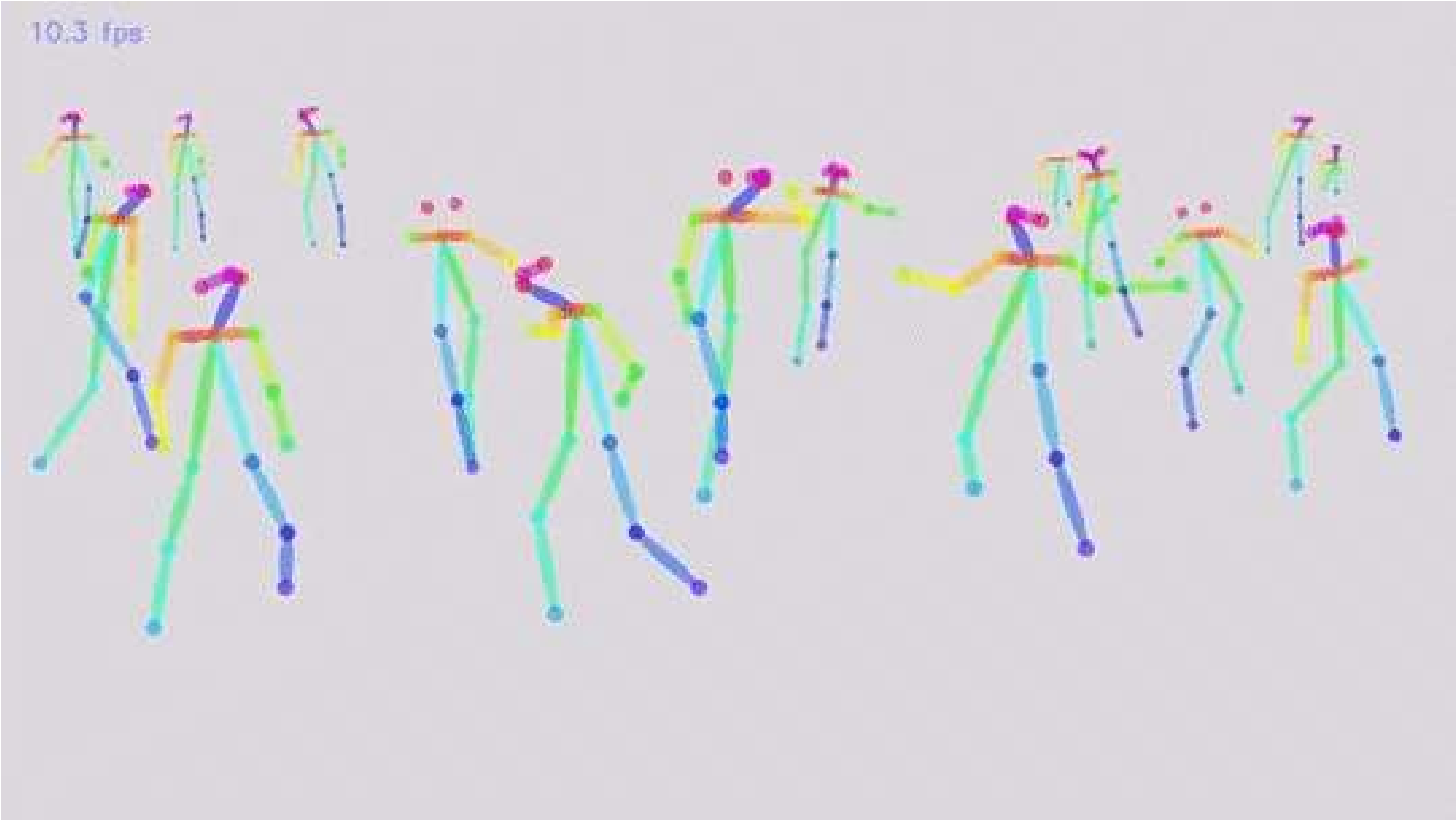
https://github.com/tornadomeet/mxnet-face

Face Recognition

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

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

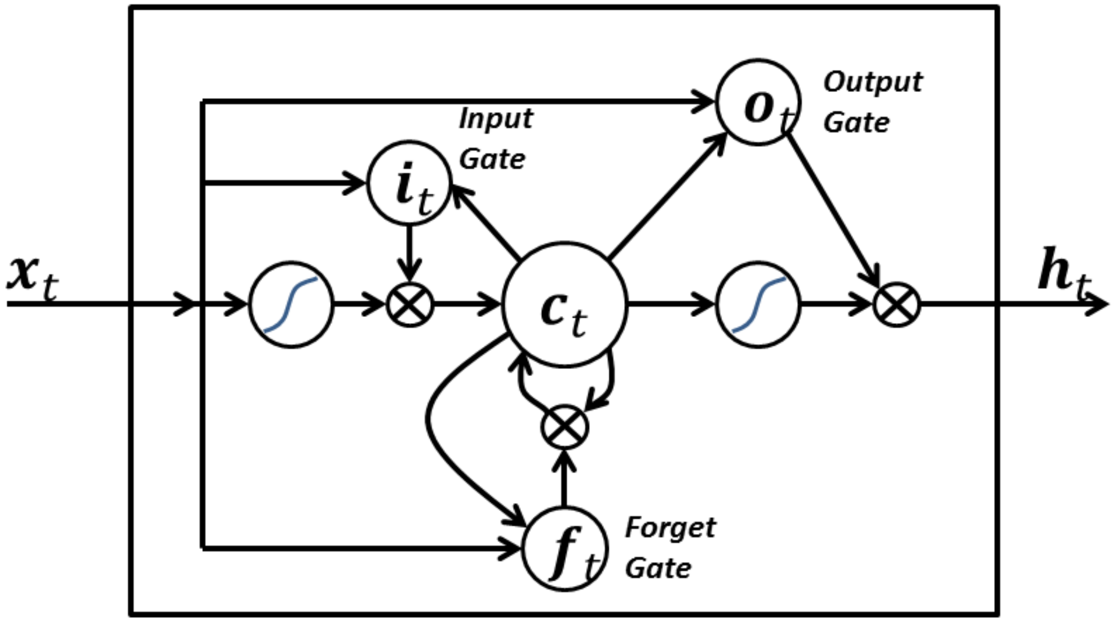
Real-Time Pose Estimation



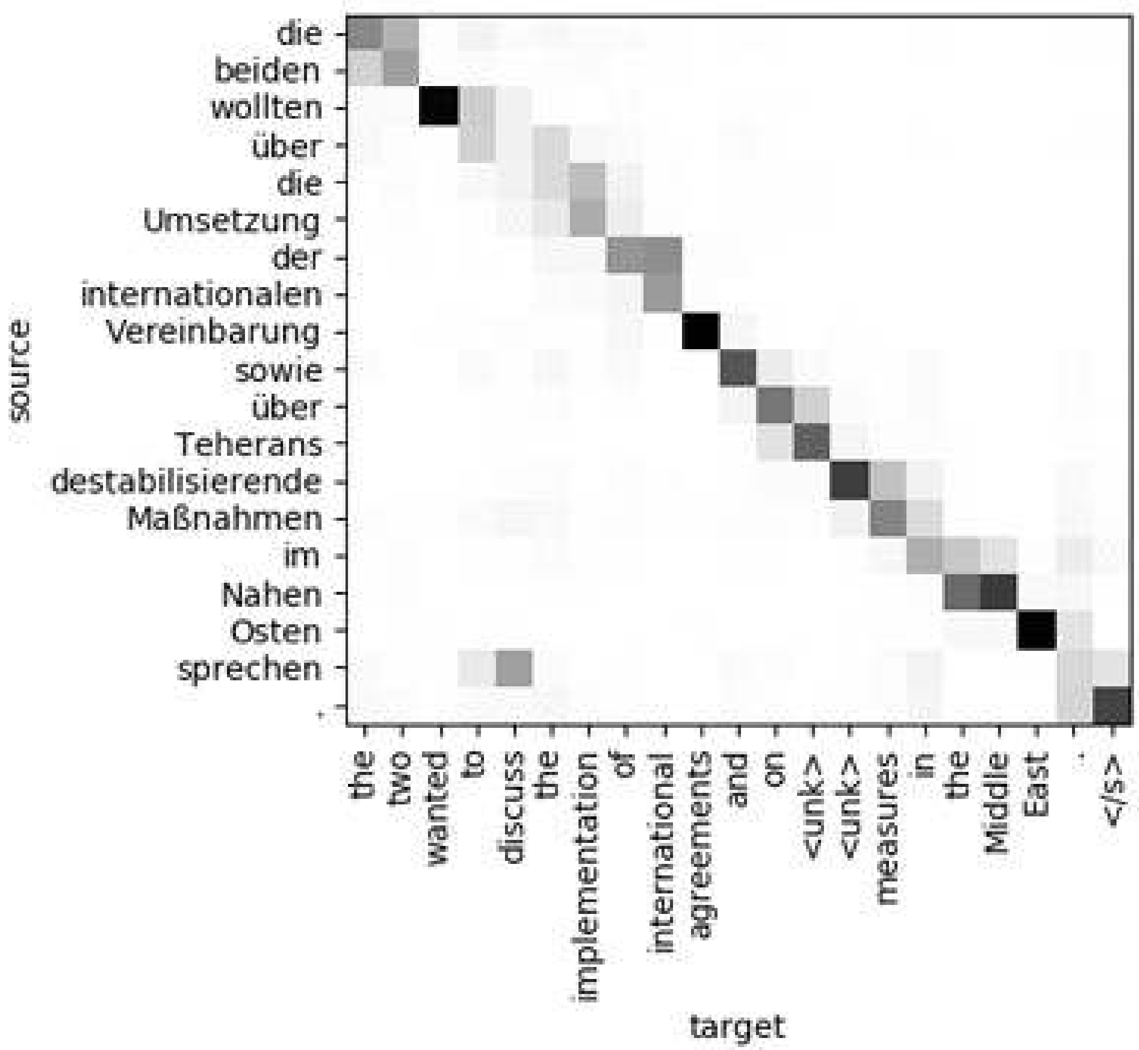
https://github.com/dragonfly90/mxnet_Realtime_Multi-Person_Pose_Estimation

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

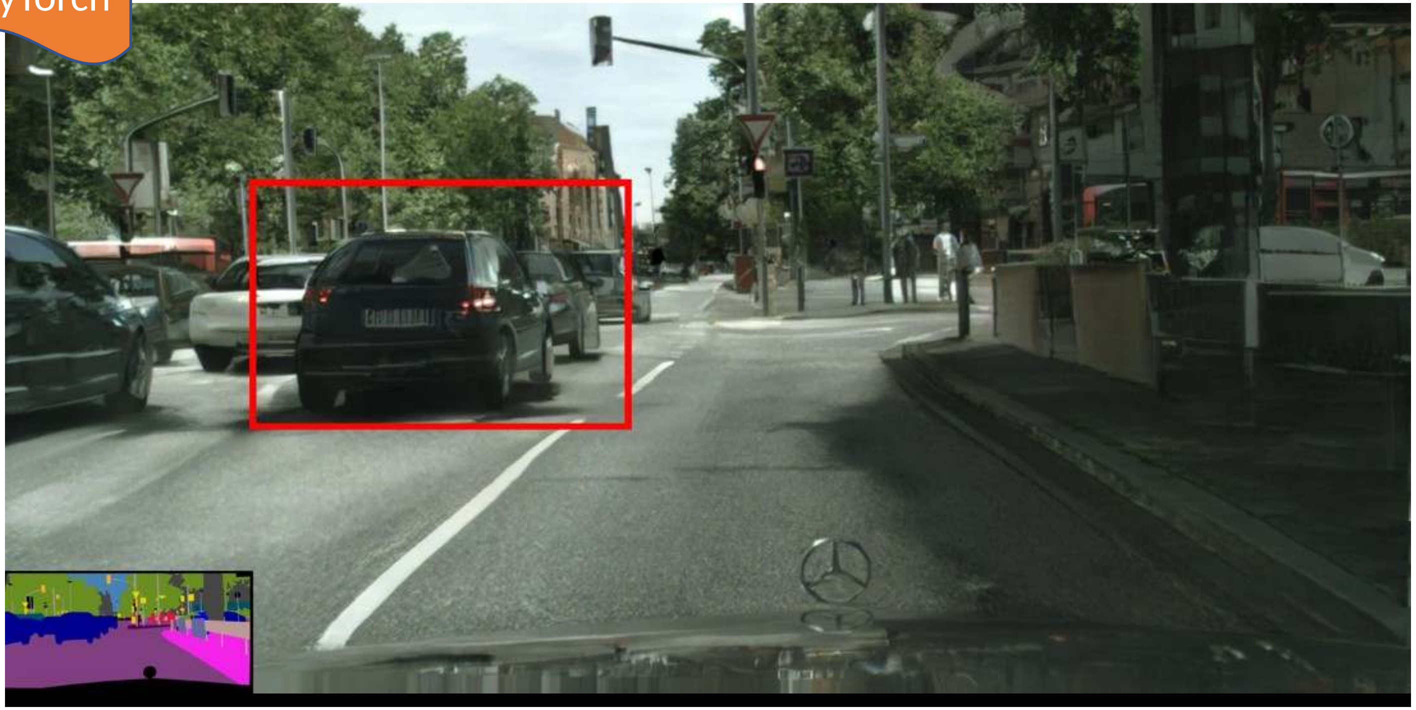


https://github.com/awslabs/sockeye

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

TF

PyTorch

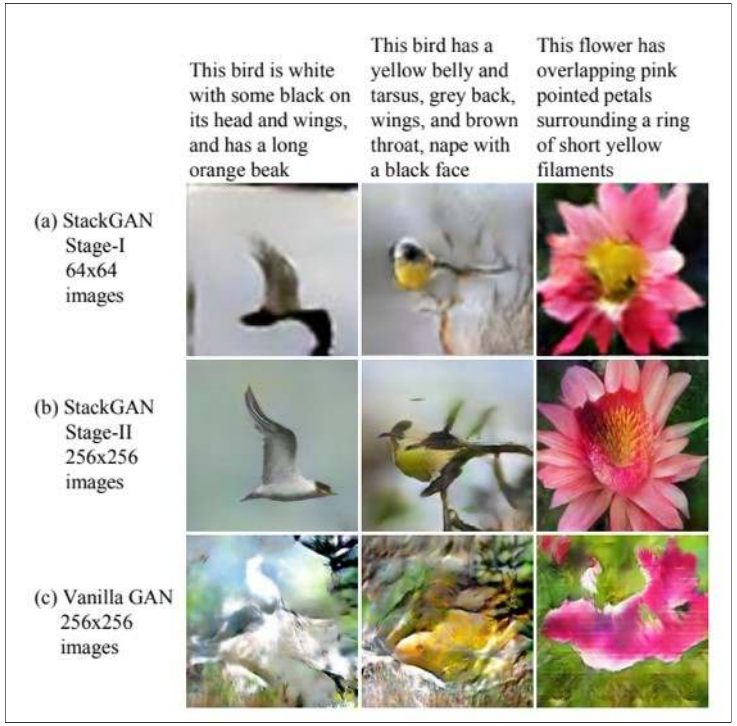


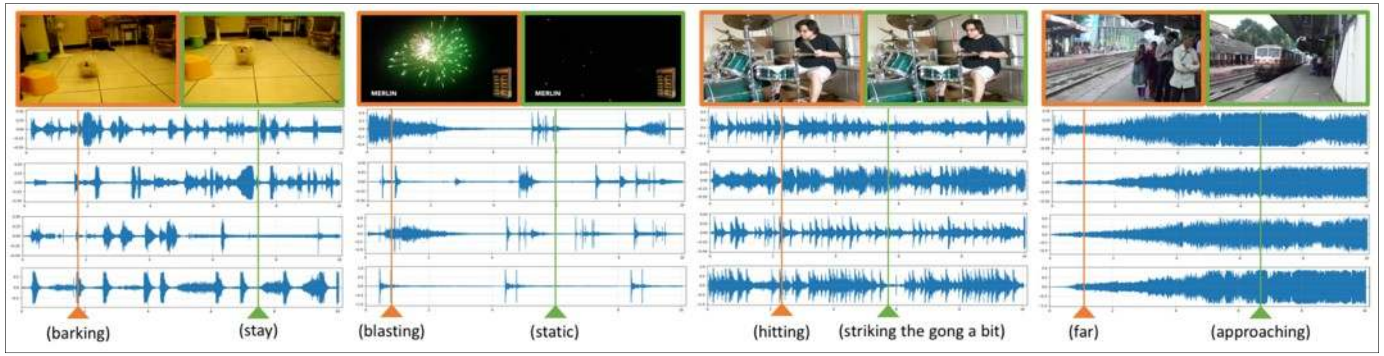
Generating new "celebrity" faces https://github.com/tkarras/progressive_growing_of_gans

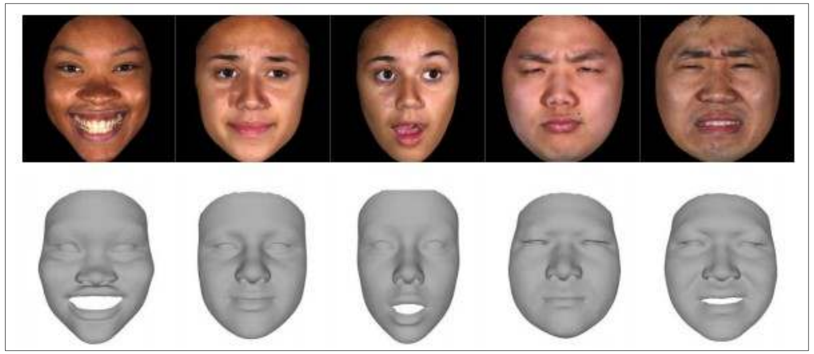
From semantic map to 2048x1024 picture
https://tcwang0509.github.io/pix2pixHD/

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.







Apache MXNet

Apache MXNet: Open Source library for Deep Learning

Programmable

Simple syntax,
multiple
languages

Portable

Highly efficient
models for
mobile
and IoT

High Performance

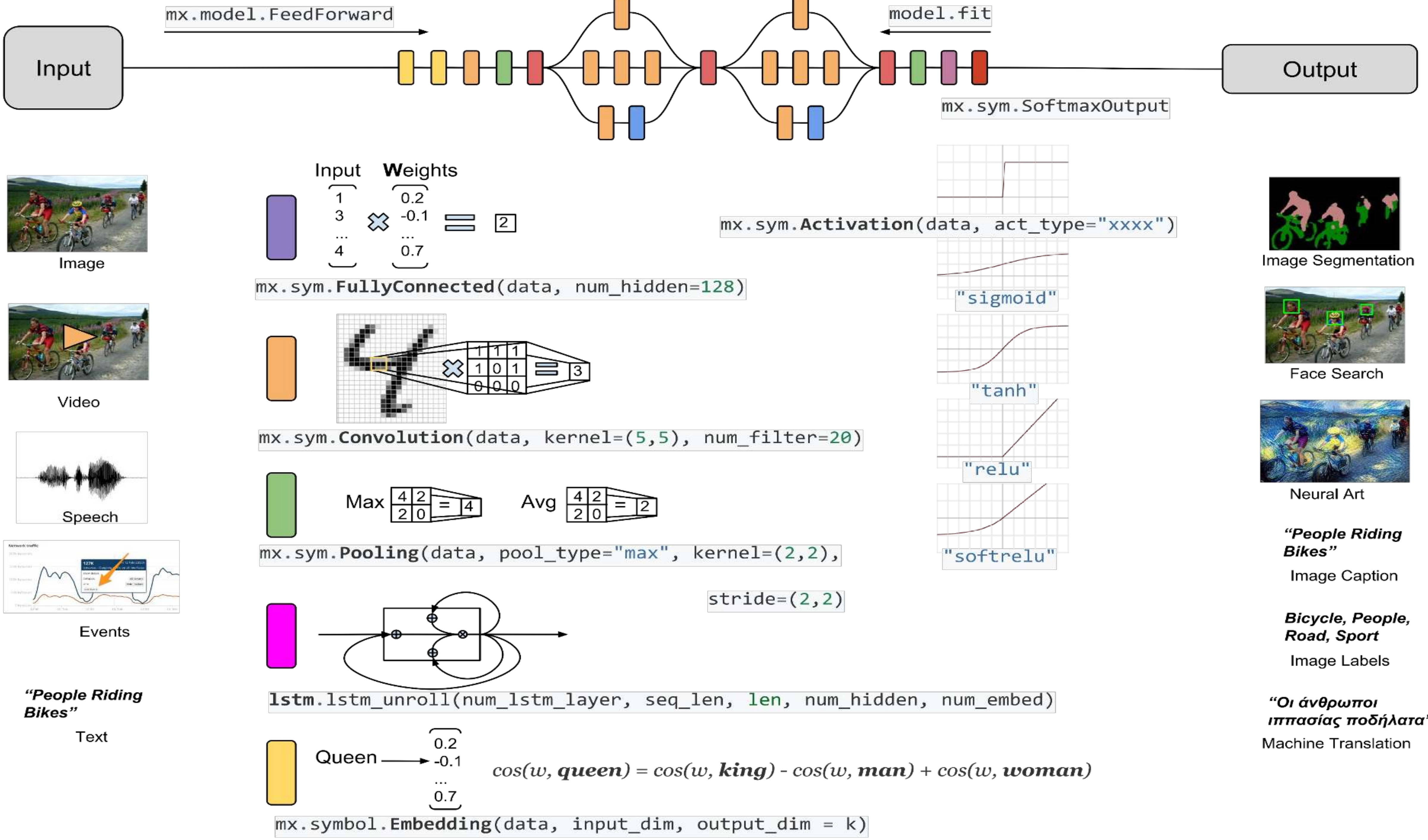
Near linear scaling
across hundreds of
GPUs

Most Open

Accepted into the
Apache Incubator

Best On AWS

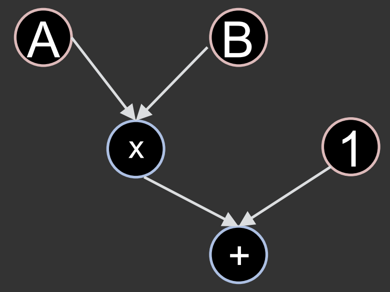
Optimized for
Deep Learning on AWS


mx.model.FeedForward
model.fit
Input
Output
mx.sym.SoftmaxOutput
Image
Video
Speech
Events
“People Riding Bikes”
Text
Input
Weights
1
3
...
4
0.2
-0.1
...
0.7
2
mx.sym.FullyConnected(data, num_hidden=128)
1 1 1
1 0 1
0 0 0
3
mx.sym.Convolution(data, kernel=(5,5), num_filter=20)
Max
4 2
2 0
= 4
Avg
4 2
2 0
= 2
mx.sym.Pooling(data, pool_type="max", kernel=(2,2),
stride=(2,2)
lstm.lstm_unroll(num_lstm_layer, seq_len, len, num_hidden, num_embed)
Queen
0.2
-0.1
...
0.7
cos(w, queen) = cos(w, king) - cos(w, man) + cos(w, woman)
mx.symbol.Embedding(data, input_dim, output_dim = k)
mx.sym.Activation(data, act_type="xxxx")
"sigmoid"
"tanh"
"relu"
"softrelu"
Image Segmentation
Face Search
Neural Art
“People Riding Bikes”
Image Caption
Bicycle, People, Road, Sport
Image Labels
“Οι άνθρωποι ιππασίας ποδήλατα”
Machine Translation

Declarative Programming

‘define then run’

```
A = Variable('A')
B = Variable('B')
C = B * A
D = C + 1
f = compile(D)
d = f(A=np.ones(10),
      B=np.ones(10)*2)
```



C can share memory with D because C is deleted later

PROS

- More chances for optimization
- Language independent
- E.g. TensorFlow, Theano, Caffe, MXNet Symbol API

CONS

- Less flexible
- ’Black box’ training

DEMO: Symbol API

1 – Fully Connected Neural Network (MNIST)

2 – Convolution Neural Network (MNIST)

Imperative Programming

‘define by run’

```
import numpy as np
a = np.ones(10)
b = np.ones(10) * 2
c = b * a
d = c + 1
```

PROS

- Straightforward and flexible.
- Take advantage of language native features (loop, condition, debugger).
- E.g. Numpy, PyTorch, MXNet Gluon API

CONS

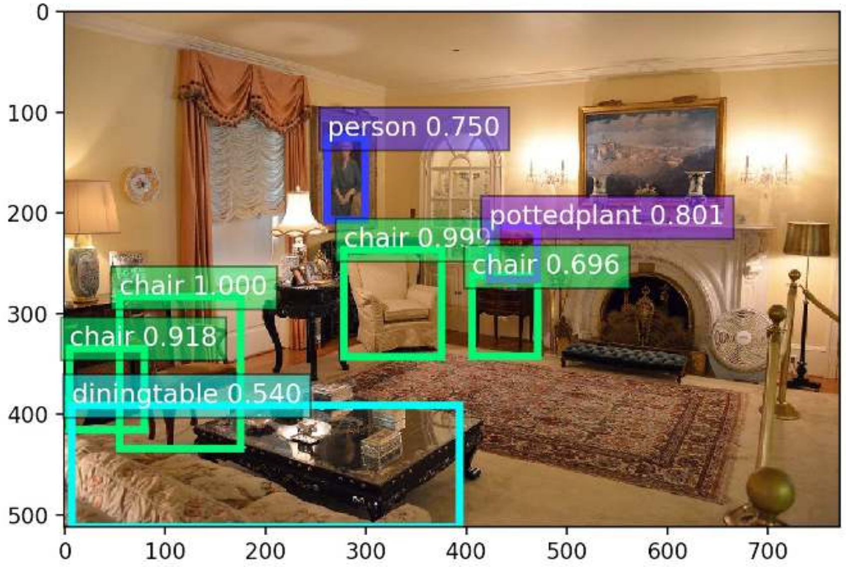
- Harder to optimize

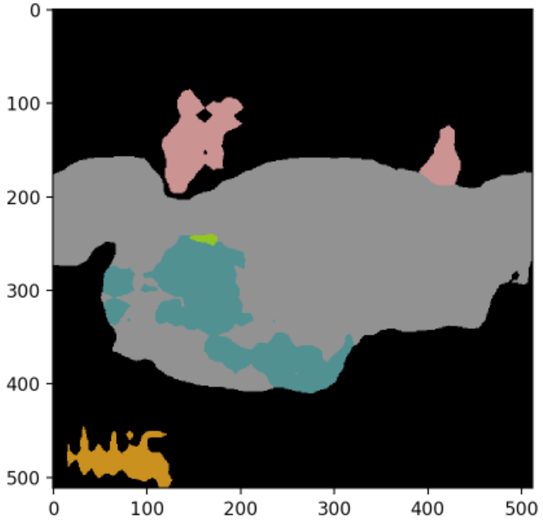
DEMO: Gluon API

Fully Connected Network (MNIST)

Gluon CV: classification, detection, segmentation

[electric_guitar],
with probability 0.671





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

DEMO: Gluon CV

Model Server for Apache MXNet

ubuntu/python-2.7	ubuntu/python-3.5
AWS CodeBuild Passing	AWS CodeBuild Passing

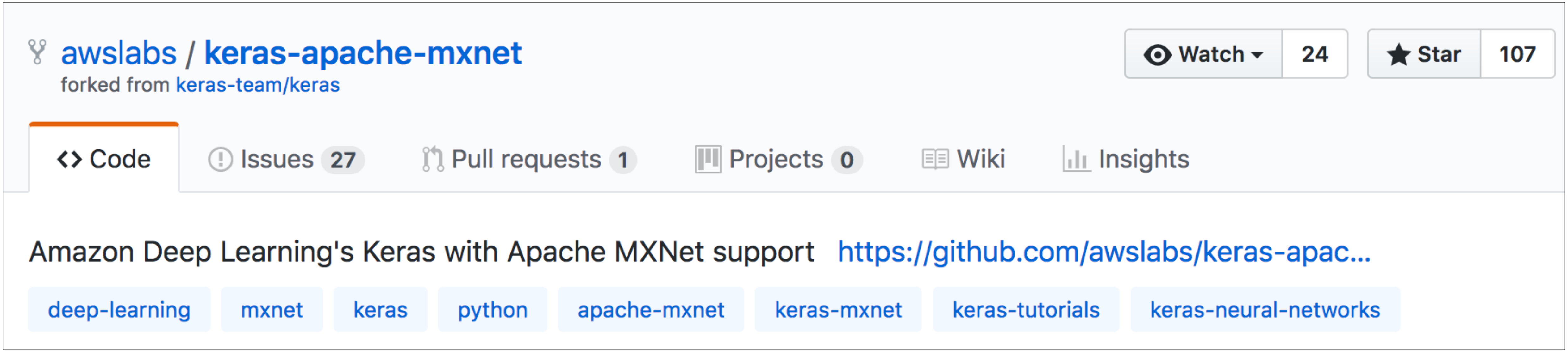
Model Server for Apache MXNet (MMS) is a flexible and easy to use tool for serving Deep Learning models.

Use MMS Server CLI, or the pre-configured Docker images, to start a service that sets up HTTP endpoints to handle model inference requests.

https://github.com/awslabs/mxnet-model-server/

https://aws.amazon.com/blogs/ai/announcing-onnx-support-for-apache-mxnet/

Keras-MXNet



https://github.com/awslabs/keras-apache-mxnet

DEMO: Keras-MXNet

Convolutional Neural Network (MNIST)

Infrastructure for Deep Learning

Amazon EC2 C5 instances

C5: Next Generation
Compute-Optimized
Instances with
Intel® Xeon® Scalable Processor

AWS Compute optimized instances support the new Intel® AVX-512 advanced instruction set, enabling you to more efficiently run vector processing workloads with single and double floating point precision, such as AI/machine learning or video processing.

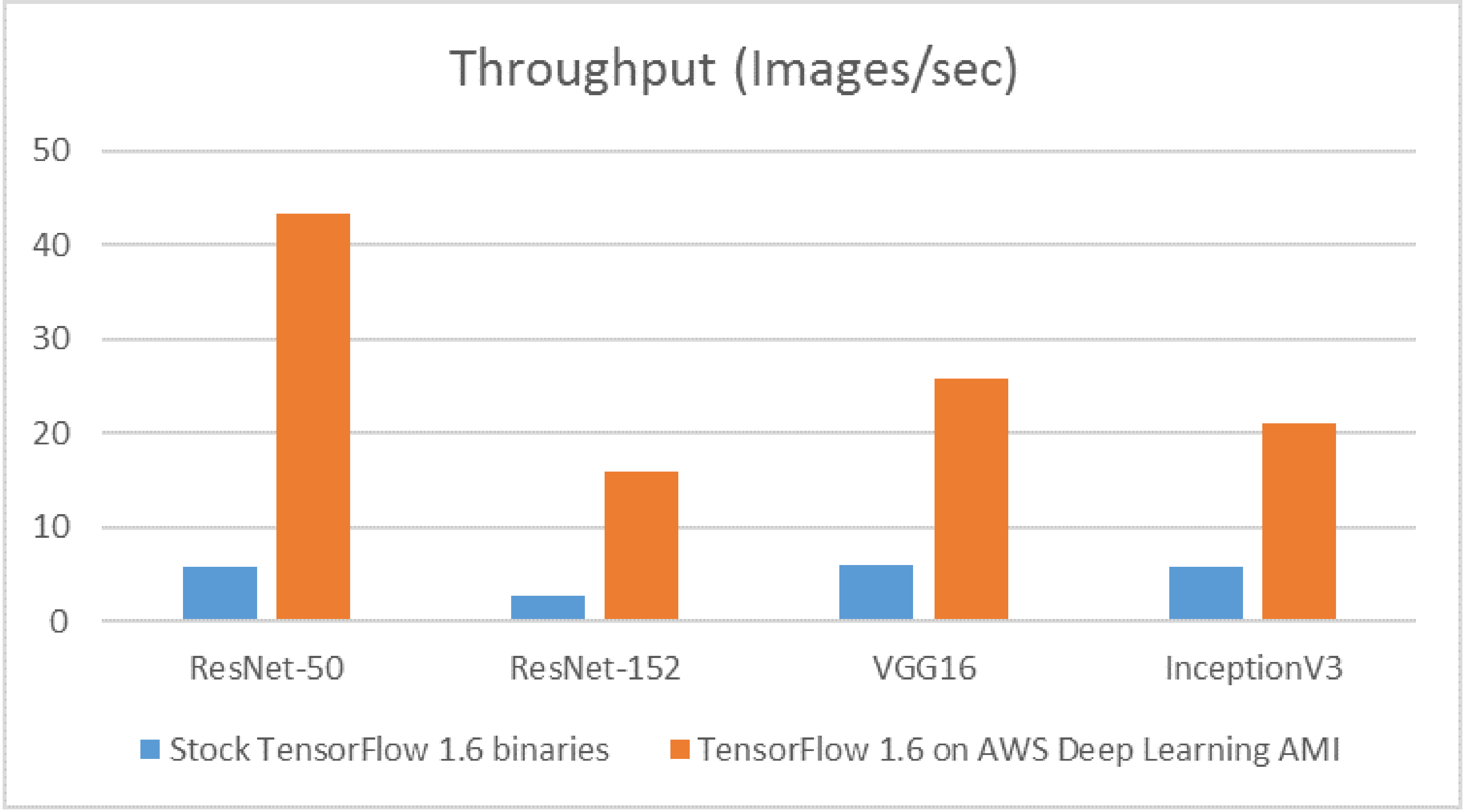
25% improvement in price/performance over C4

C4	C5
36 vCPUs	72 vCPUs **2X vCPUs**
“Haswell”	“Skylake”
4 Gbps to EBS	12 Gbps to EBS **3X throughput**
60 GiB memory	144 GiB memory **2.4X memory**

AVX 512

2X performance

Faster TensorFlow training on C5



https://aws.amazon.com/blogs/machine-learning/faster-training-with-optimized-tensorflow-1-6-on-amazon-ec2-c5-and-p3-instances/

Amazon EC2 P3 Instances

The fastest, most powerful GPU instances in the cloud

- P3.2xlarge, P3.8xlarge, P3.16xlarge
- Up to eight NVIDIA Tesla V100 GPUs in a single instance
    - 40,960 CUDA cores, 5120 Tensor cores
    - 128GB of GPU memory
- 1 PetaFLOPs of computational performance *– 14x better than P2*
- 300 GB/s GPU-to-GPU communication (NVLink) *– 9x better than P2*

AWS Deep Learning AMI

Preconfigured environments to quickly build Deep Learning applications

Conda AMI
For developers who want pre-installed pip packages of DL frameworks in separate virtual environments.

Base AMI
For developers who want a clean slate to set up private DL engine repositories or custom builds of DL engines.

AMI with source code
For developers who want preinstalled DL frameworks and their source code in a shared Python environment.

TensorFlow mxnet torch Microsoft CNTK Caffe2 Chainer K GLUON PYTORCH

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

Amazon SageMaker

Pre-built notebooks for common problems

Built-in, high-performance algorithms

ALGORITHMS

K-Means Clustering
Principal Component Analysis
Neural Topic Modelling
Factorization Machines
Linear Learner

XGBoost
Latent Dirichlet Allocation
Image Classification
Seq2Seq,
And more!

FRAMEWORKS

Apache MXNet
TensorFlow

Caffe2, CNTK,
PyTorch, Torch

Set up and manage environments for training

Train and tune model (trial and error)

Deploy model in production

Scale and manage the production environment

Build

Amazon SageMaker

Pre-built notebooks for common problems

Built-in, high-performance algorithms

One-click training

Hyperparameter optimization

Deploy model in production

Scale and manage the production environment

Build

Train

Amazon SageMaker

Pre-built notebooks for common problems

Built-in, high-performance algorithms

One-click training

Hyperparameter optimization

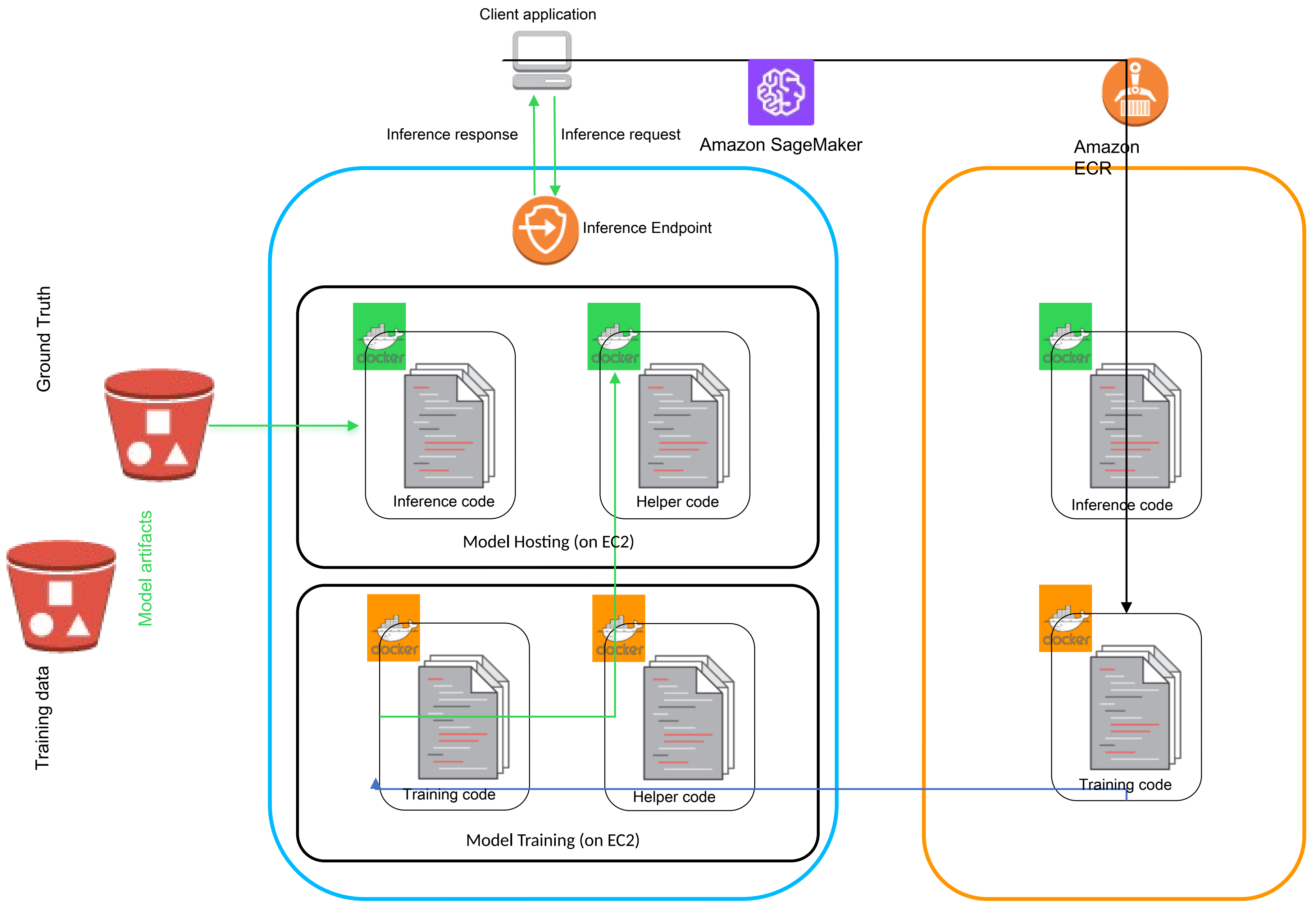
One-click deployment

Fully managed hosting with auto-scaling

Build

Train

Deploy


Client application
Inference response
Inference request
Amazon SageMaker
Amazon ECR
Inference Endpoint
Ground Truth
docker
Inference code
docker
Helper code
Model Hosting (on EC2)
Model artifacts
docker
Training code
docker
Helper code
Model Training (on EC2)
Training data
docker
Inference code
docker
Training code

Open Source Containers for TF and MXNet

https://github.com/aws/sagemaker-tensorflow-containers

https://github.com/aws/sagemaker-mxnet-containers

- Build them and run them on your own machine
- Run them directly on a notebook instance (aka local mode)
- Customize them and push them to ECR
- Run them on SageMaker for training and prediction at scale

DEMO: SageMaker

1 – Use the built-in algorithm for image classification (CIFAR-10)

2– Bring your own Tensorflow script for image classification (MNIST)

3– Bring your own Gluon script for sentiment analysis (Stanford Sentiment Tree Bank 2)

4 – Build your own Keras-MXNet container (CNN + MNIST)

5 – Build your own PyTorch container (CNN + MNIST)

Danke schön!

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://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

Julien Simon

Principal Technical Evangelist, AI and Machine Learning

@ julsimon