Deep Learning for Developers

Julien Simon, AI Evangelist, EMEA
@julsimon

What to expect

- An introduction to Deep Learning
- An introduction to Apache MXNet
- Demos using Jupyter notebooks on Amazon SageMaker
- Resources

- 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

Amazon Alexa is based on Deep Learning

Amazon AI is based on Deep Learning

Vision Services

Amazon Rekognition Image
Deep learning-based image analysis

Learn more »

Amazon Rekognition Video
Deep learning-based video analysis

Learn more »

Conversational chatbots

Amazon Lex
Build chatbots to engage customers

Learn more »

Language Services

Amazon Comprehend
Discover insights and relationships in text

Learn more »

Amazon Translate
Fluent translation of text

Learn more »

Amazon Transcribe
Automatic speech recognition

Learn more »

Amazon Polly
Natural sounding text to speech

Learn more »

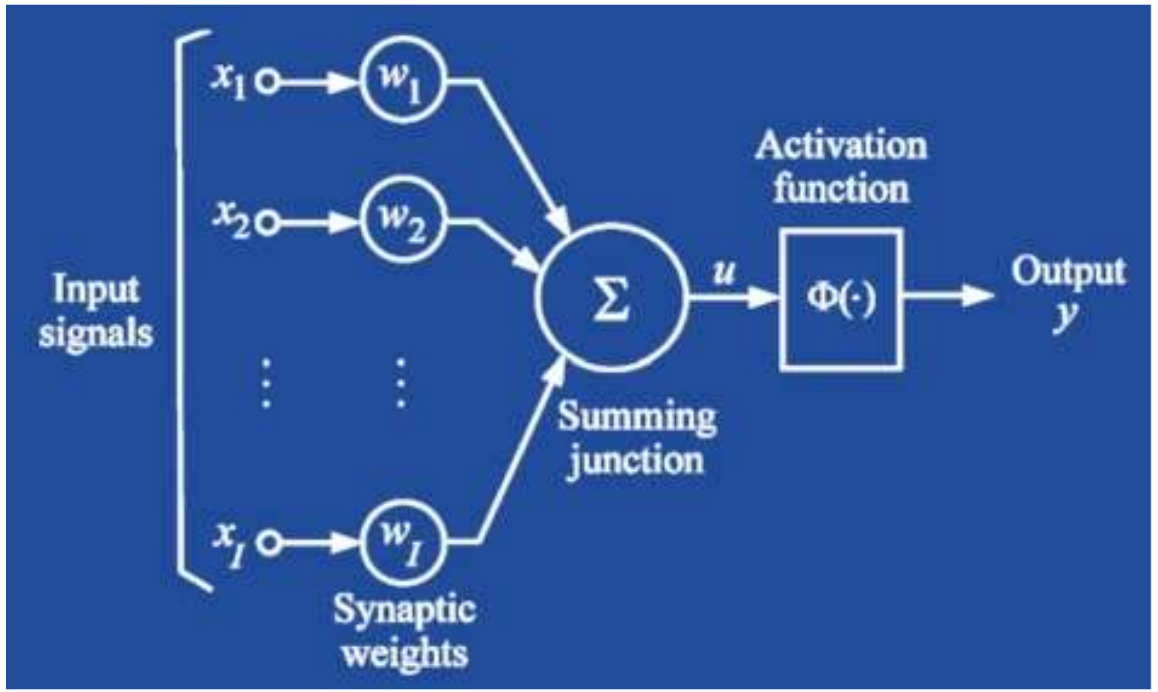
https://www.oreilly.com/ideas/self-driving-trucks-enter-the-fast-lane-using-deep-learn

Last June, tuSimple drove an autonomous truck

for 200 miles from Yuma, AZ to San Diego,

An introduction to Deep Learning

The neuron



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

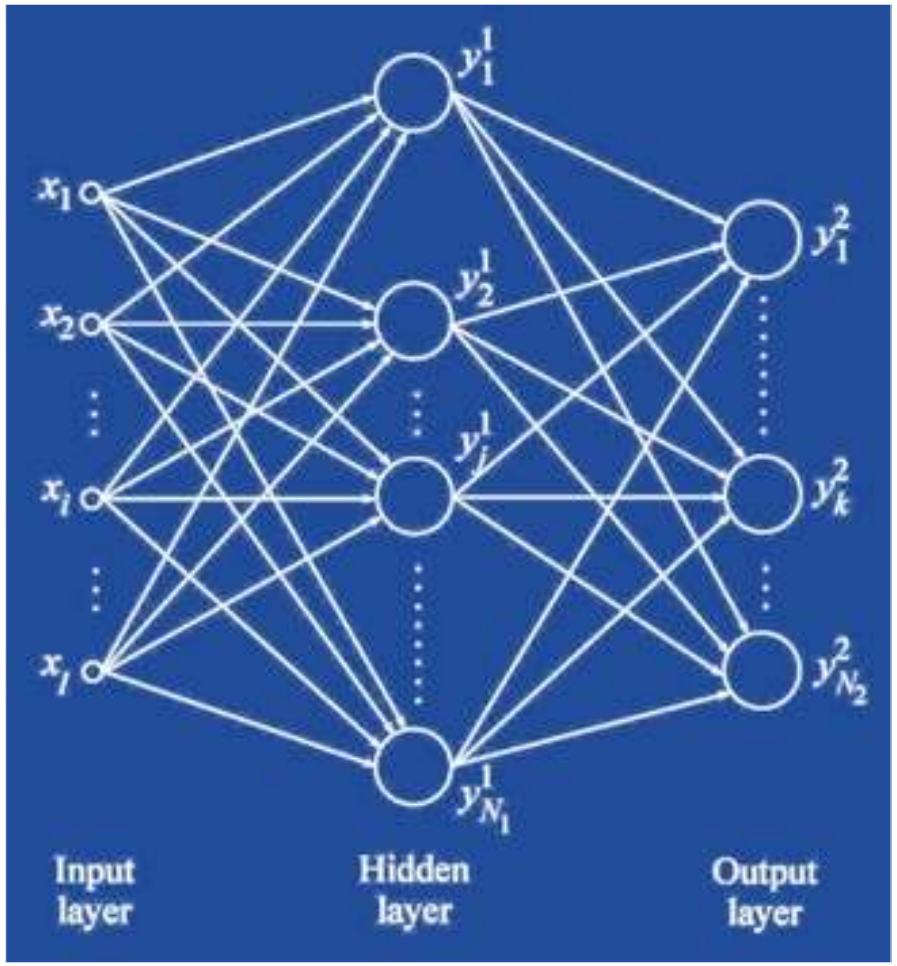
”Multiply and Accumulate”

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 =

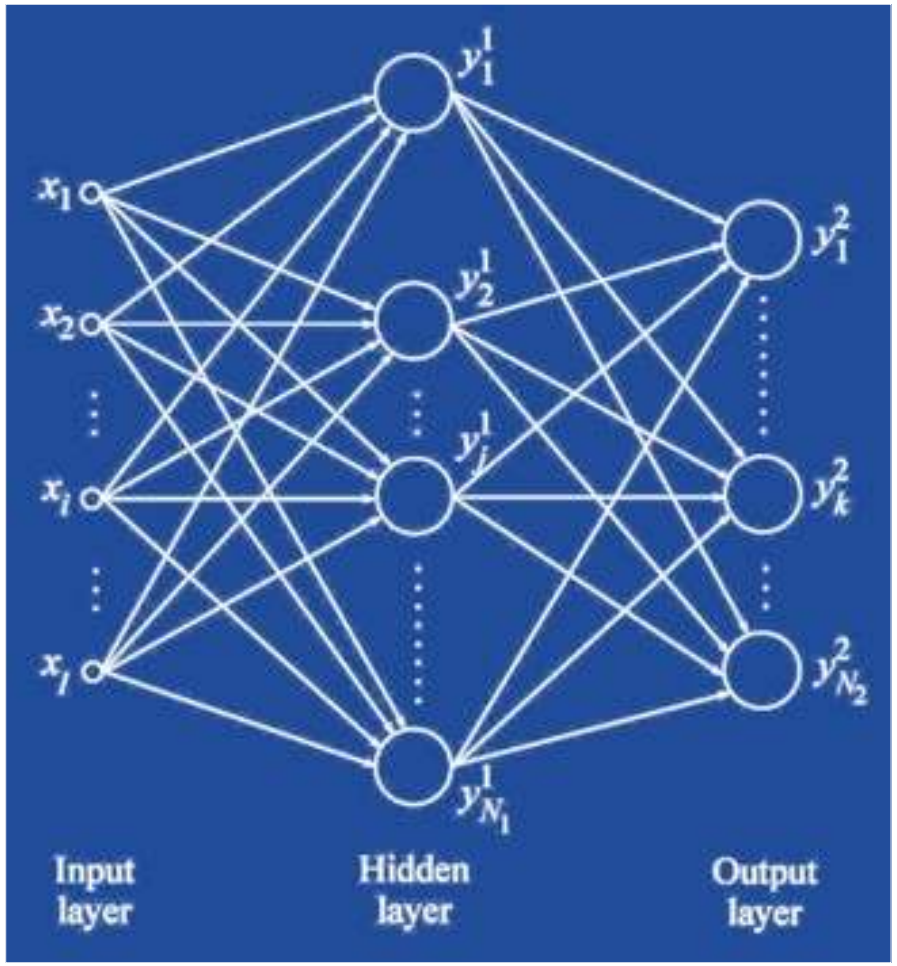
I features	
$x_{11}, x_{12}, \ldots x_{1l}$	
$x_{21}, x_{22}, \ldots x_{2l}$	
… … …	
$x_{m1}, x_{m2}, \ldots x_{ml}$	m samples

y =

y_1	0,0,1,0,0,…,0
y_2	1,0,0,0,0,…,0
…	…
y_m	0,0,0,0,1,…,0
	One-hot encoding

m labels, N_2 outputs

Neural networks



l features

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

m samples

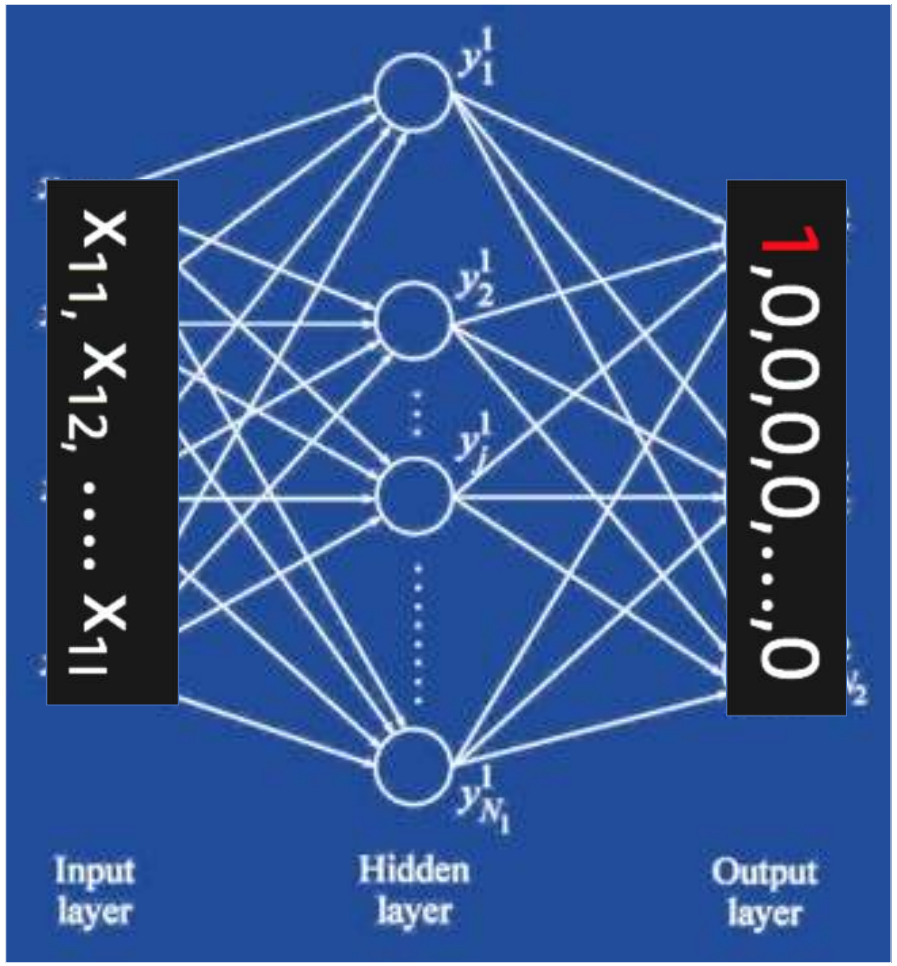
$$y = \begin{bmatrix} y_1 \\ y_2 \\ \ldots \\ y_m \end{bmatrix} \quad \begin{bmatrix} 0,0,1,\ldots,0 \\ 1,0,0,\ldots,0 \\ \ldots \\ 0,0,0,\ldots,0 \end{bmatrix}$$

m labels,
N_2 categories

One-hot encoding

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

Neural networks



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

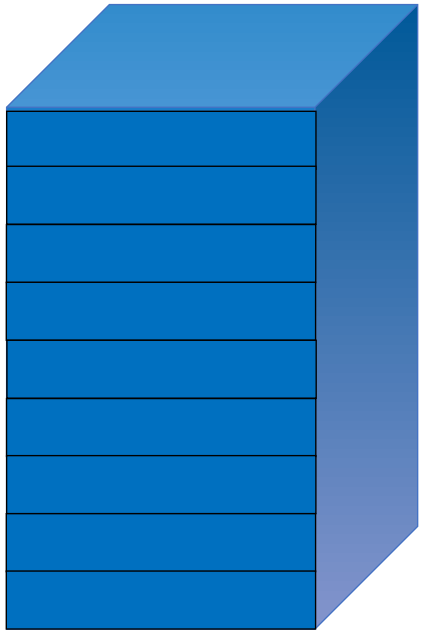
$error = loss(Y_1, Y'_1)$

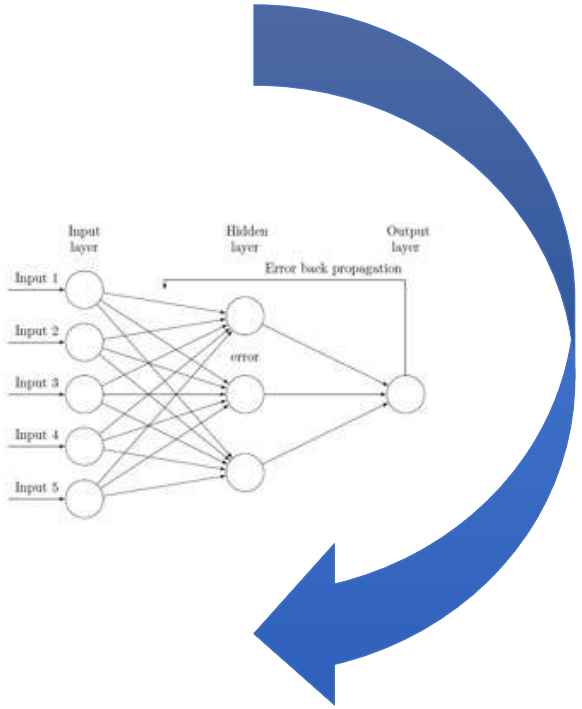
For a batch of samples:

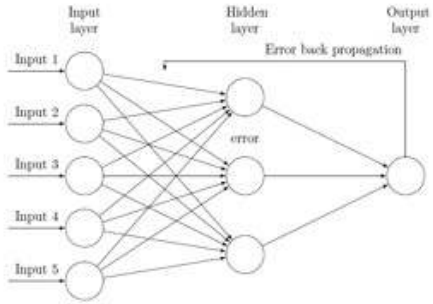
$$\sum_{i=1}^{batch\ size} loss(Y_i, Y'_i) = \text{batch error}$$

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

Training







Training data set

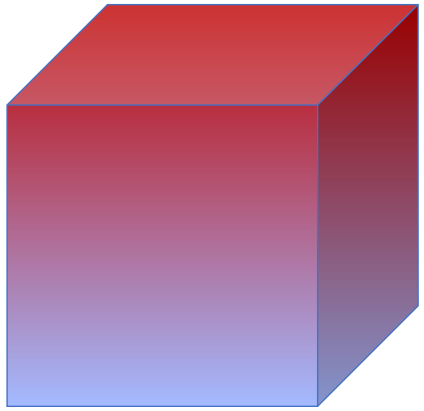
Backpropagation

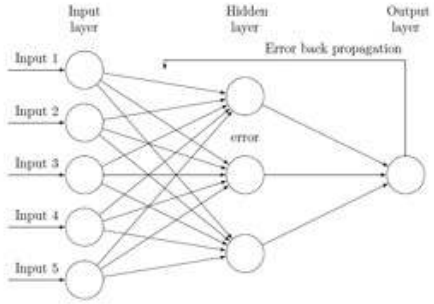
Trained neural network

Batch size
Learning rate
Number of epochs

Hyper parameters

Validation





Validation accuracy

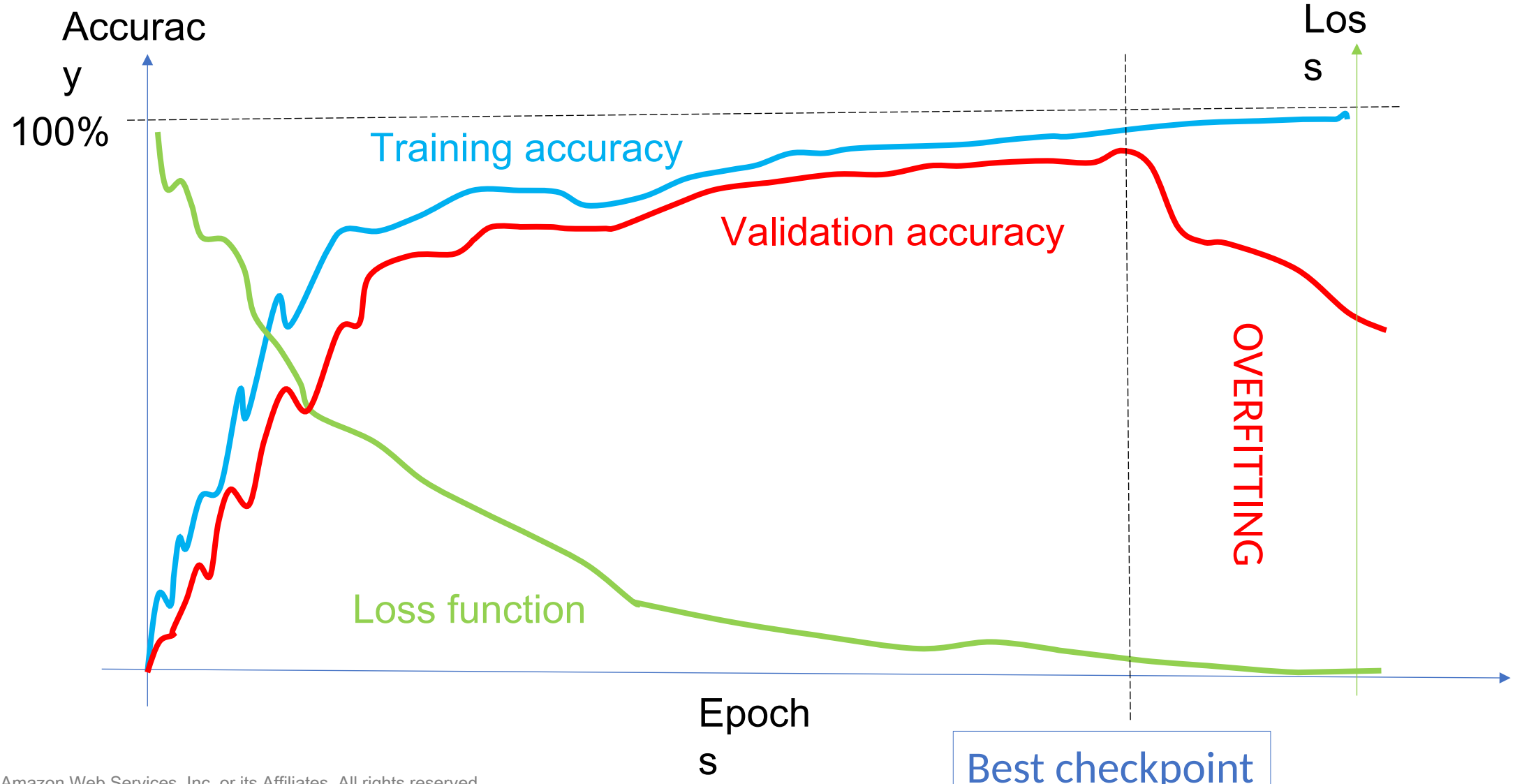
Validation data set

Trained neural network

Prediction at the end of each epoch

Save the model at the end of each epoch

Early stopping

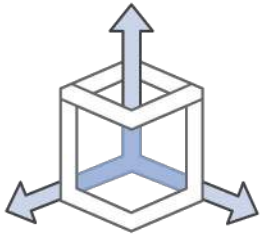


Apache MXNet

Apache MXNet: Open Source library for Deep Learning

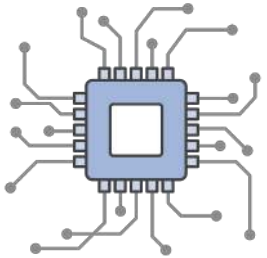
Programmable

Simple syntax, multiple languages



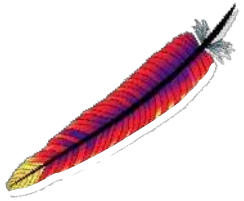
Portable

Highly efficient models for mobile and IoT



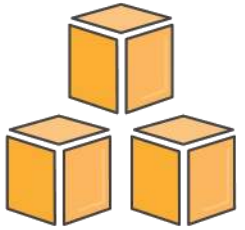
High Performance

Near linear scaling across hundreds of GPUs



Most Open

Accepted into the Apache Incubator

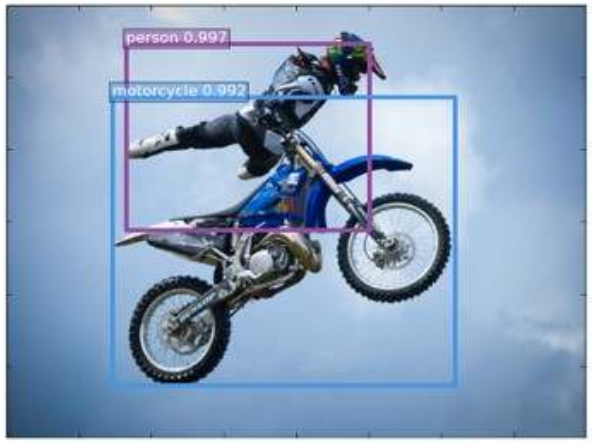


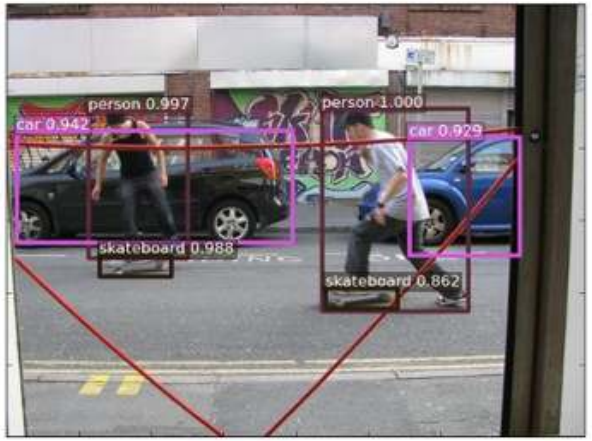
Best On AWS

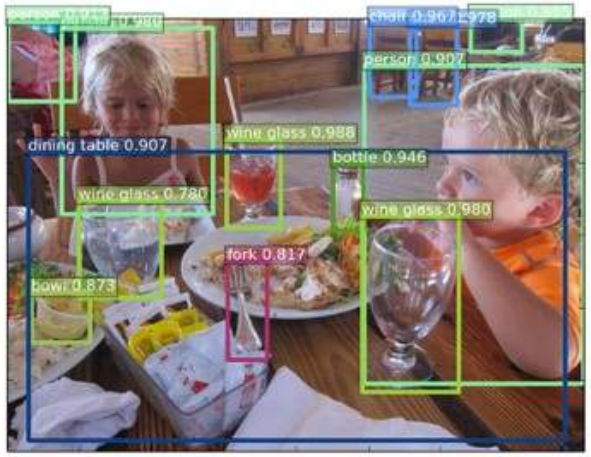
Optimized for Deep Learning on AWS

MXNet 1.0 released on December 4th

Object Detection



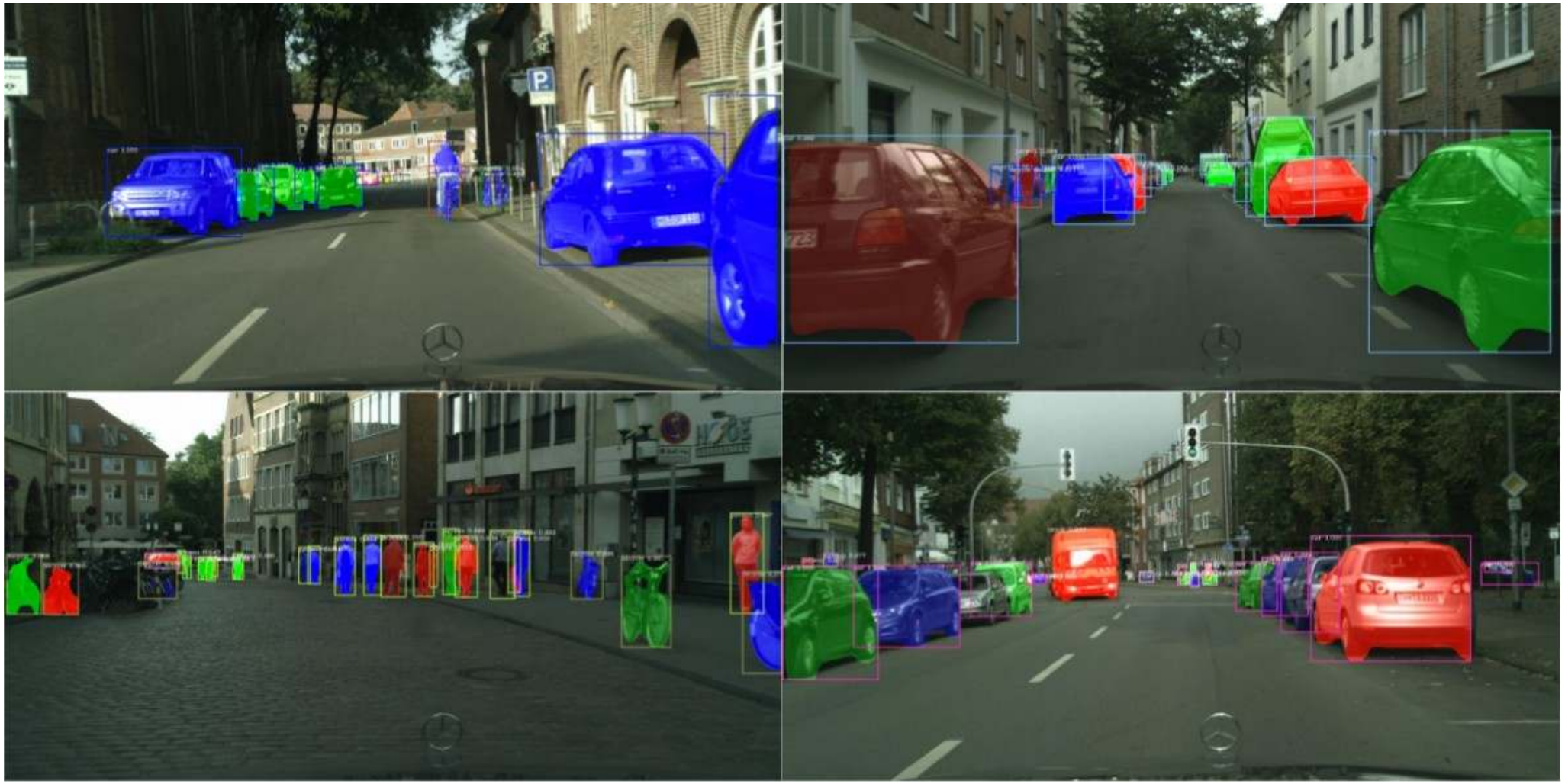




https://github.com/precedenceguo/mx-rcnn

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

Object Segmentation



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

Text Detection and Recognition

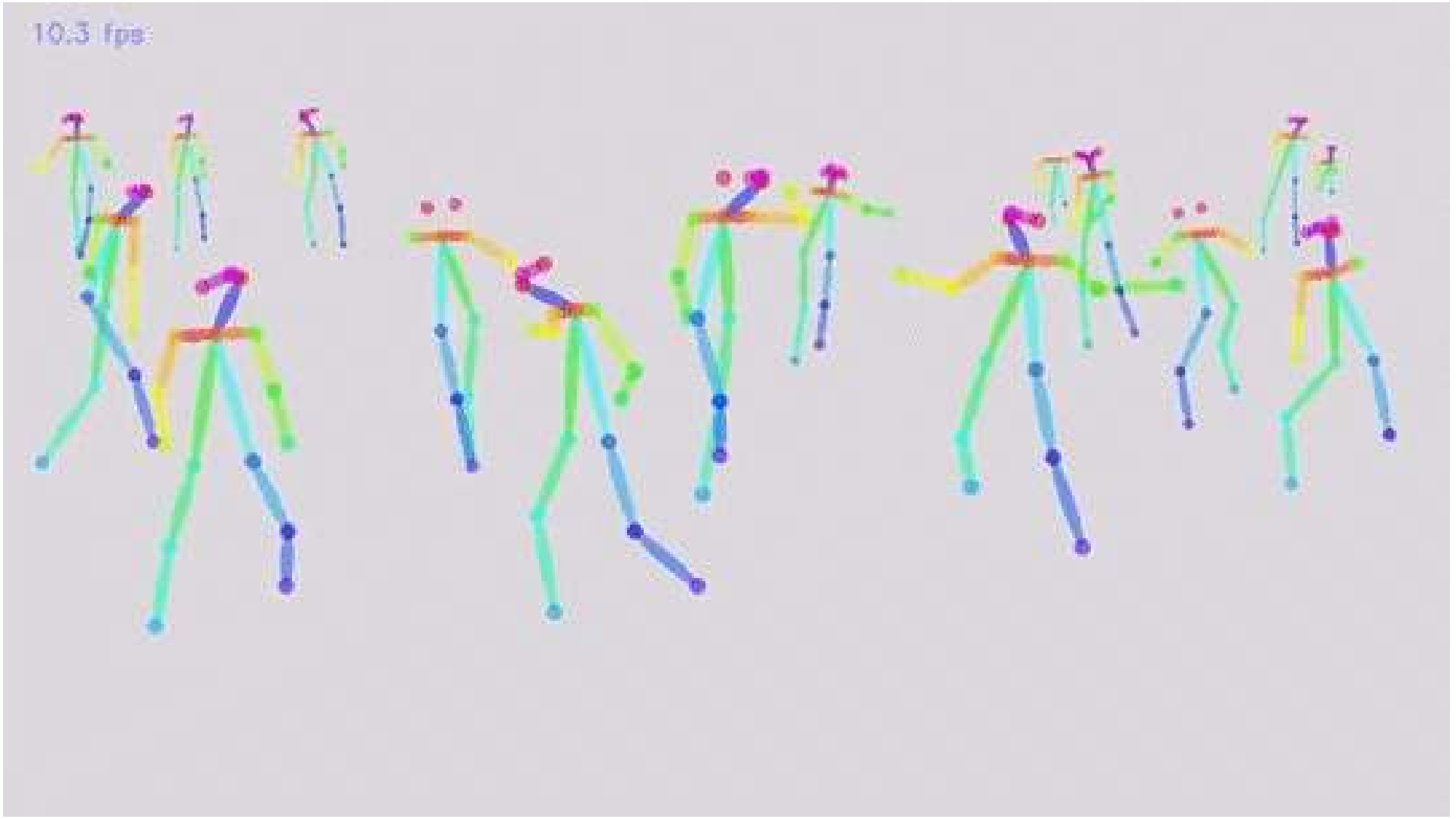
https://github.com/Bartzi/stn-ocr

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
```

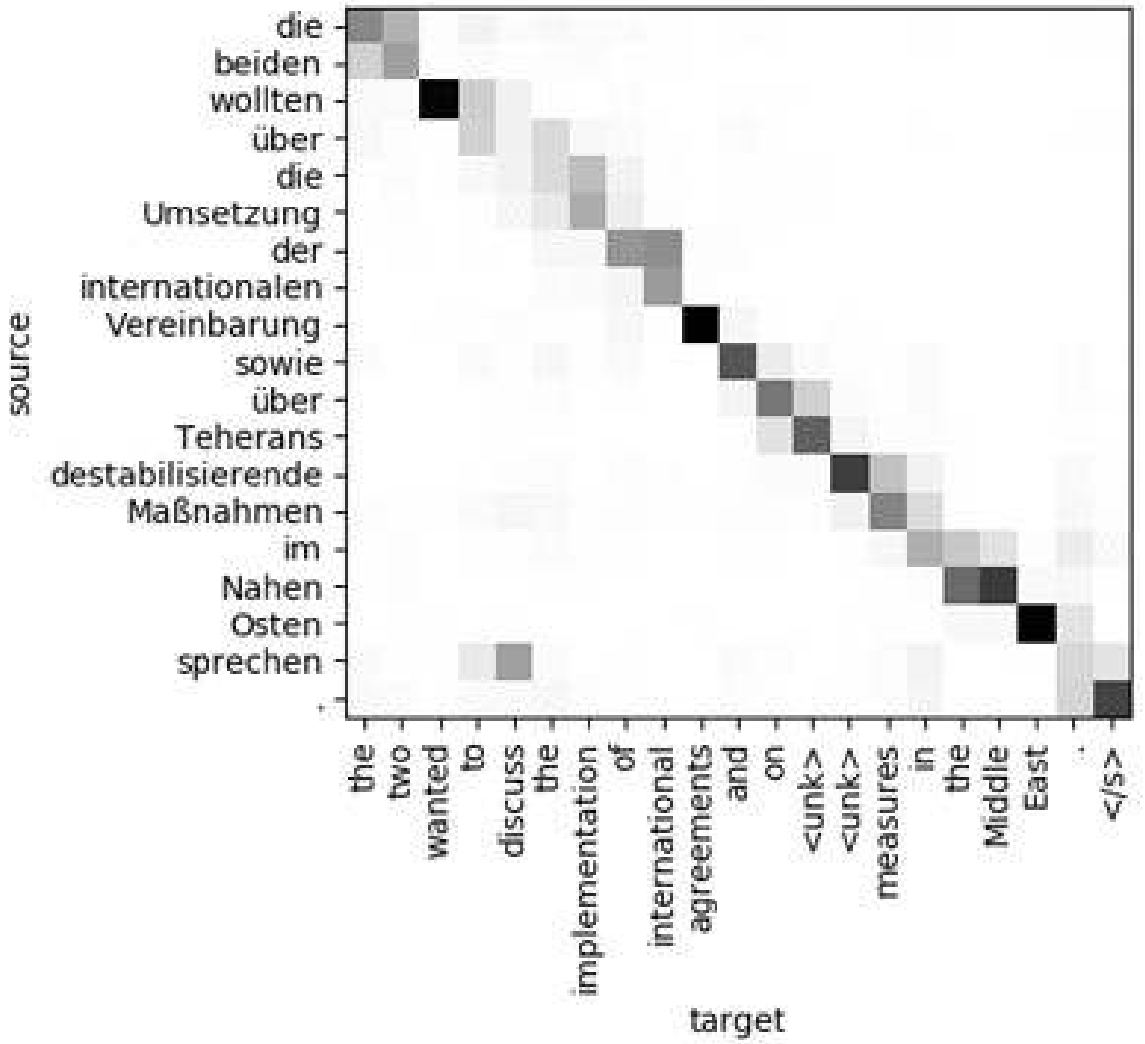
https://github.com/tornadomeet/mxnet-face

Real-Time Pose Estimation



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

Machine Translation



https://github.com/awslabs/sockeye

The Apache MXNet API

- Storing and accessing data in multi-dimensional arrays
  → *NDArray* API
- Building models (layers, weights, activation functions)
  → *Symbol* API
- Serving data during training and validation
  → Iterators
- Training and using models
  → *Module* API

Demos

https://github.com/juliensimon/dlnotebooks

1) Synthetic data set
2) Classify images with pre-trained models
3) Learn MNIST with a Multi-Layer Perceptron
4) Learn MNIST with the LeNet CNN
5) Predict handmade MNIST samples
6) Train and host a MNIST model with Amazon SageMaker

Resources

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

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

https://mxnet.incubator.apache.org

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

https://github.com/gluon-api

https://github.com/awslabs/sockeye

An overview of Amazon SageMaker https://www.youtube.com/watch?v=ym7NEYEx9x4

https://medium.com/@julsimon

Thank you!

Julien Simon, AI Evangelist, EMEA
@julsimon