Deep Learning for Developers

Julien Simon <@julsimon>
AI Evangelist, EMEA

Questions, questions…

What's the business problem my IT has failed to solve

Should I design and train my own Deep Learning model?

Should I use a pre-trained model?

Should I use a SaaS solution?

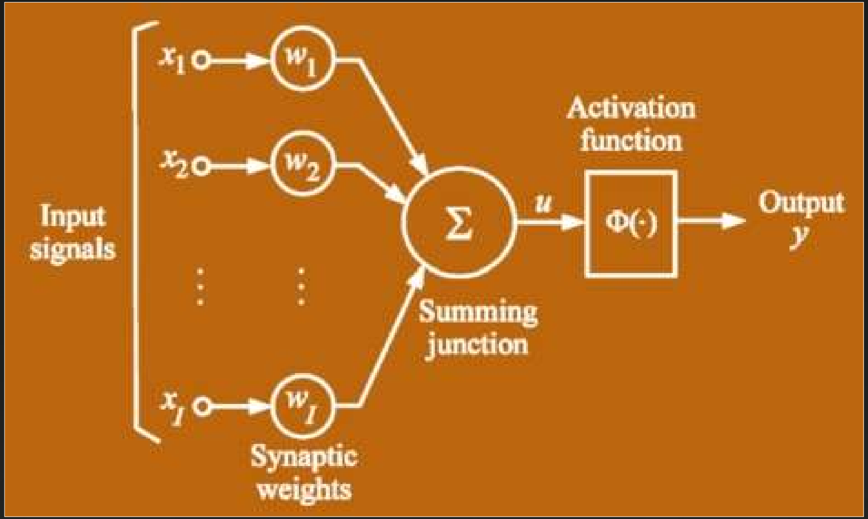
Same questions as "Big Data" years ago

Amazon AI for every developer

Services	**Chat** Amazon Lex	**Speech** Amazon Polly	**Vision** Amazon Rekognition			
Platforms	Amazon ML	Spark & EMR	Kinesis	Batch	ECS	
Engines	MXNet	TensorFlow	Caffe	Theano	Pytorch	CNTK
Infrastructure	GPU	CPU	IoT	Mobile		

Neural Networks

The neuron



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

$x = [x_{1,}\ x_{2,}\ \ldots.\ x_{l\,]}$

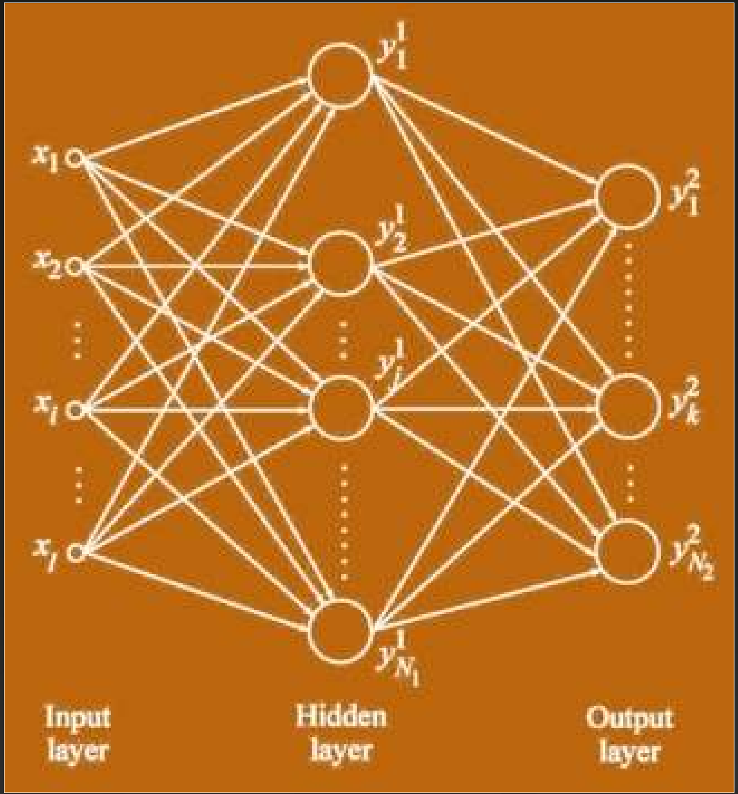
$w = [w_{1,}\ w_{2,}\ \ldots.\ w_{l\,]}$

u = x.w

Activation function

Name	Plot	Equation
Identity		$f(x) = x$
Binary step		$f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$
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]		$f(x) = \frac{x}{1+\|x\|}$
Rectified linear unit (ReLU)[9]		$f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases}$

The neural network



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

$$y = \left[\begin{array}{c} y_1 \\ y_2 \\ y_{\ldots} \\ y_m \end{array}\right] \left.\vphantom{\begin{array}{c} y \\ y \\ y \\ y \end{array}}\right\} \text{m labels}$$

Accuracy = $\dfrac{\textbf{Number of correct predictions}}{\textbf{Total number of predictions}}$

The training process

- The difference between the predicted output and the actual output (aka *ground truth*) is called the prediction loss.

- There are different ways to compute it (loss functions).

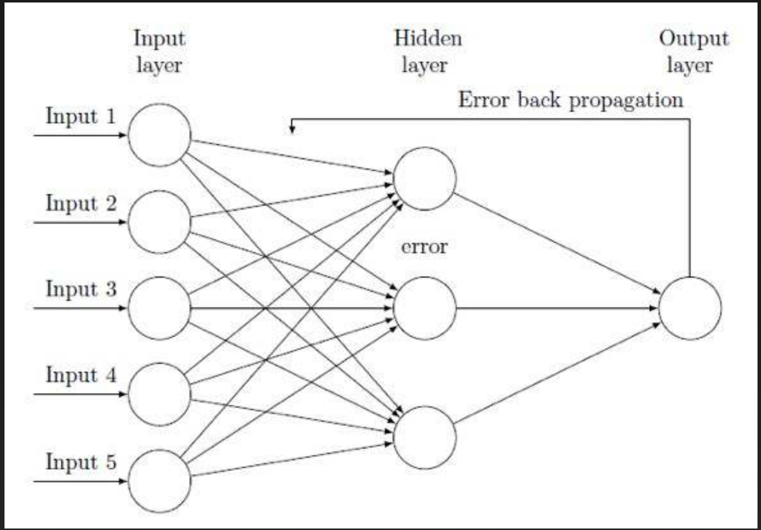
- The purpose of training is to iteratively minimize loss and maximize accuracy for a given data set.

- We need a way to adjust weights (aka parameters) in order to gradually minimize loss

  → Backpropagation + optimization algorithm

1974 – Backpropagation

Paul Werbos
Artificial Intelligence pioneer
IEEE Neural Network Pioneer Award



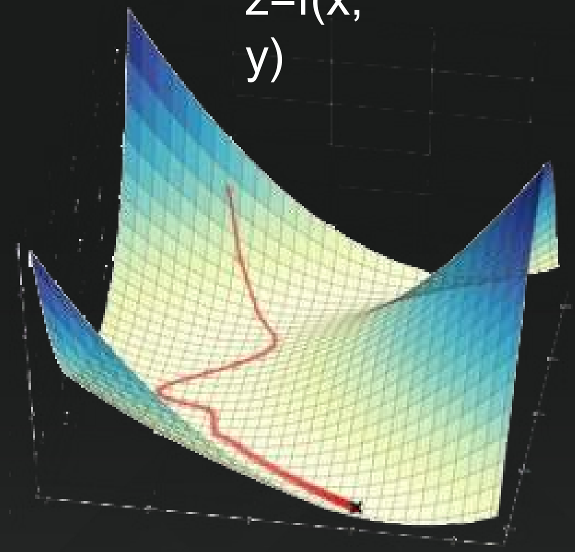
The back-propagation algorithm acts as an error correcting mechanism at each neuron level, thereby helping the network to learn effectively.

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

z=f(x, y)



The « step size » is called the learning rate

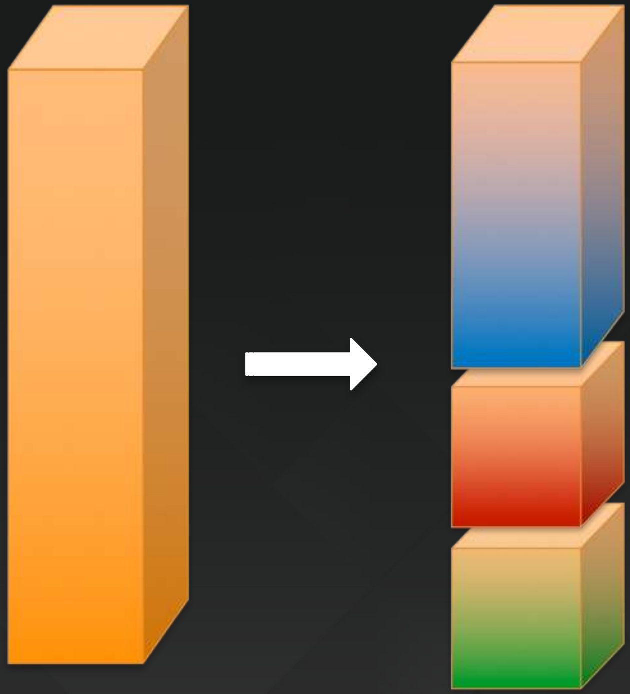
Source: https://devblogs.nvidia.com/parallelforall/deep-learning-nutshell-history-training/

aws

There is such a thing as « learning too well »

- If a network is large enough and given enough time, it will perfectly learn a data set (universal approximation theorem).
- But what about new samples? Can it also predict them correctly?
- In other words, does the network generalize well or not?
- To prevent overfitting, we need to know when to stop training.
- The training data set is not enough.

Data sets

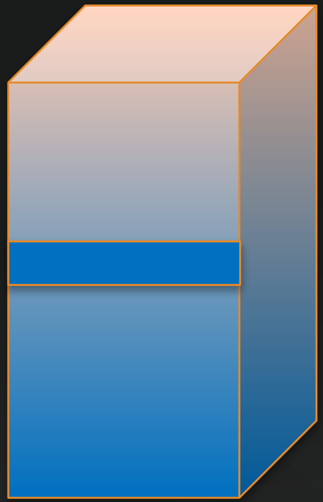


Training set: this data set is used to adjust the weights on the neural network.

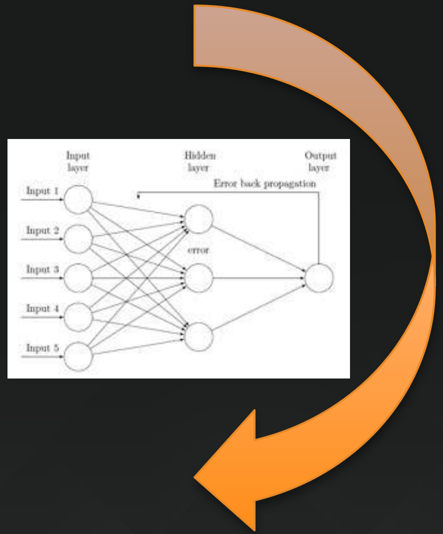
Validation set: this data set is used to minimize overfitting. You're not adjusting the weights of the network with this data set, you're just verifying that any increase in accuracy over the training data set actually yields an increase in accuracy over a data set that has not been shown to the network before.

Testing set: this data set is used only for testing the final weights in order to benchmark the actual predictive power of the network.

Training



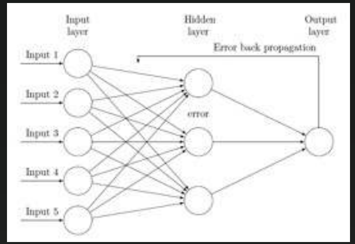
Training data set



Training

Number of epochs
Batch size
Learning rate

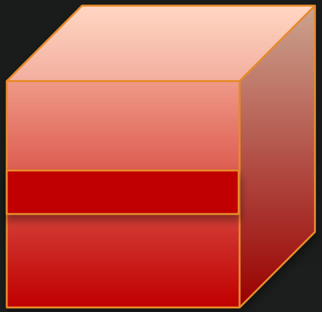
Hyper parameters



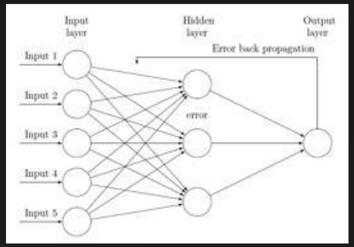
Trained neural network

aws

Validation



Validation Data set



Trained neural network

Prediction at the end of each epoch

Validation accuracy

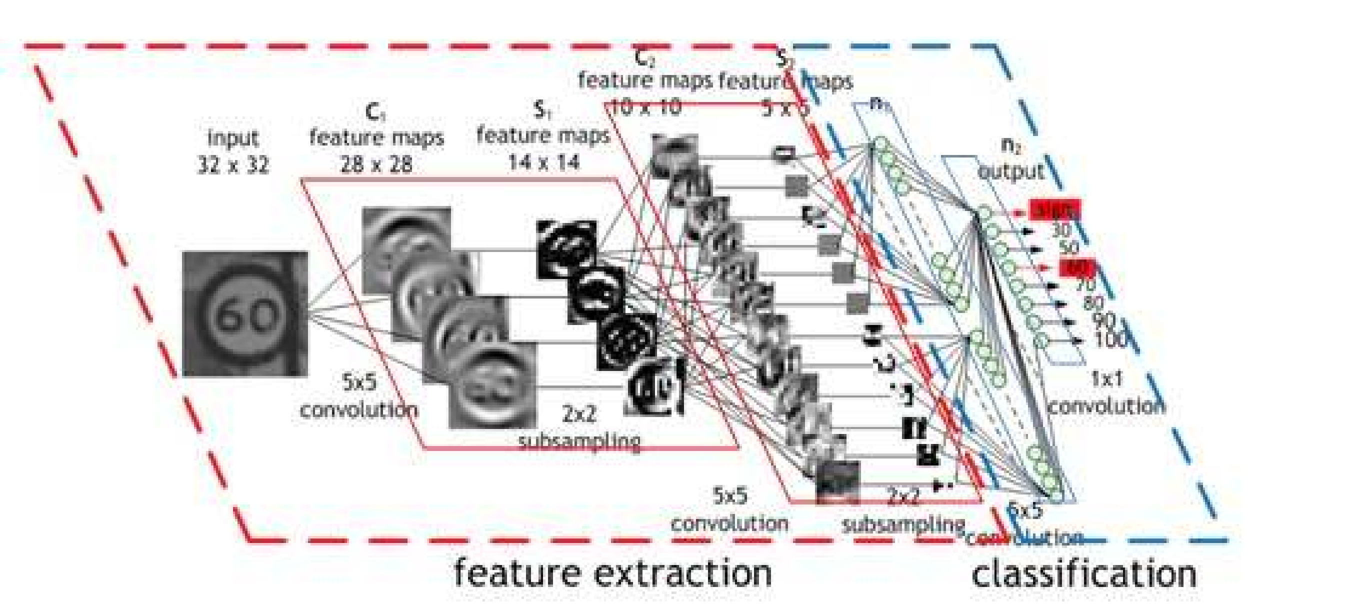
Stop training when validation accuracy stops increasing

Saving parameters at the end of each epoch

Convolutional Neural Networks

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

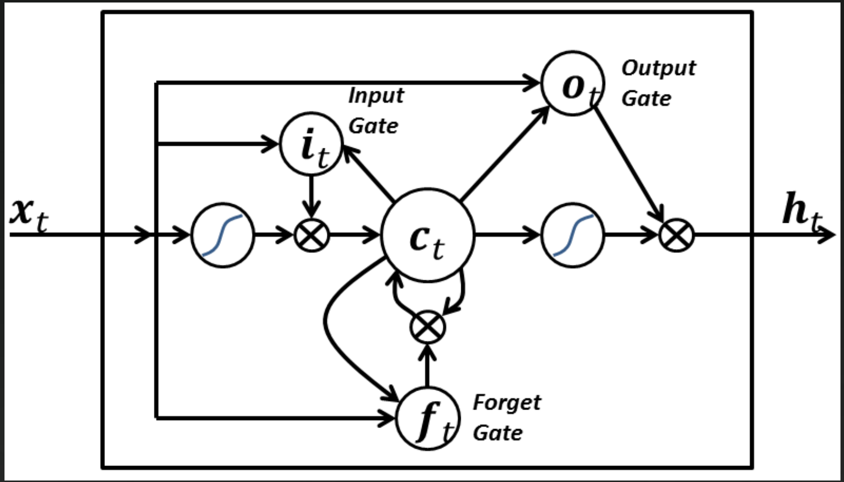
Feature extraction and downsampling allow smaller networks



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

Long Short Term Memory (LSTM) Networks

- 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



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

https://mxnet.io

aws

Last June, tuSimple drove an autonomous truck
for 200 miles from Yuma, AZ to San Diego, CA

https://www.oreilly.com/ideas/self-driving-trucks-enter-the-fast-lane-using-deep-learning mx.model.FeedForward

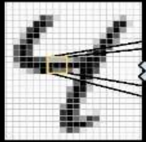
model.fit

Input

Output

mx.sym.SoftmaxOutput

mx.sym.FullyConnected(data, num_hidden=128)



mx.sym.Convolution(data, kernel=(5,5), num_filter=20)

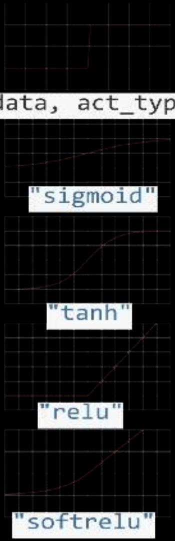
mx.sym.Pooling(data, pool_type="max", kernel=(2,2), stride=(2,2)

lstm.lstm_unroll(num_lstm_layer, seq_len, len, num_hidden, num_embed)

$cos(w, \textbf{queen}) = cos(w, \textbf{king}) - cos(w, \textbf{man}) + cos(w, \textbf{woman})$

mx.symbol.Embedding(data, input_dim, output_dim = k)

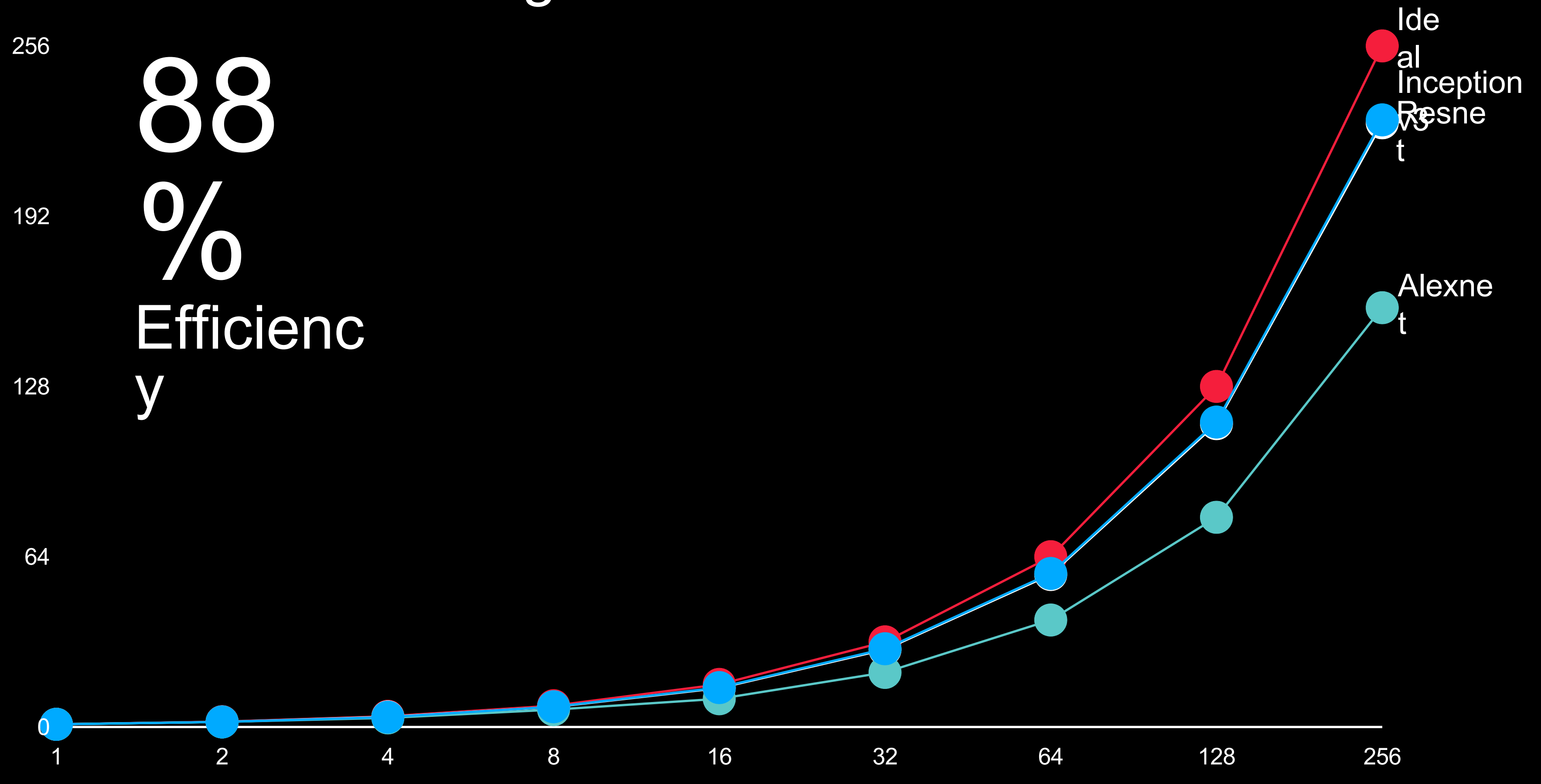
mx.sym.Activation(data, act_type="xxxx")



CPU or GPU: your choice

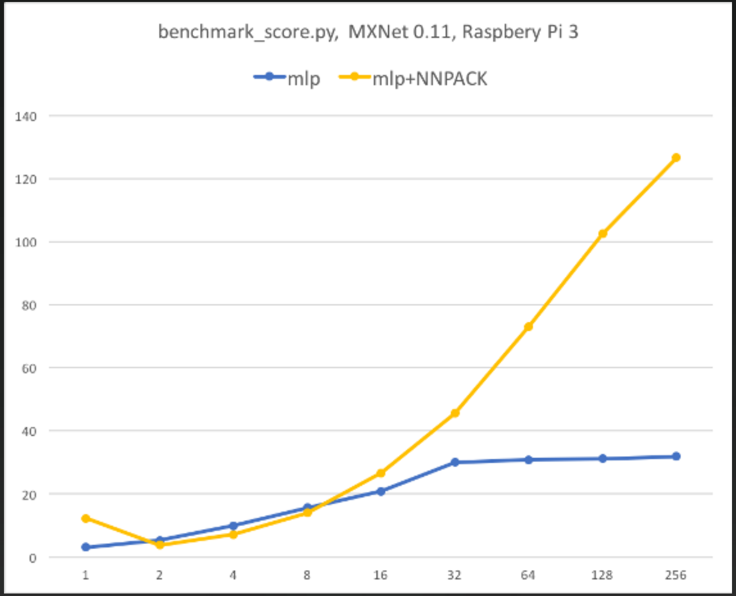
```
mod = mx.mod.Module(lenet)

mod = mx.mod.Module(lenet, context=mx.gpu(0))

mod = mx.mod.Module(lenet,
context=(mx.gpu(7), mx.gpu(8), mx.gpu(9)))
```

Multi-GPU Scaling With MXNet



Speeding up Apache MXNet inference on CPU

- Intel MKL https://software.intel.com/en-us/mkl
- NNPACK https://github.com/Maratyszcza/NNPACK



https://medium.com/@julsimon/speeding-up-apache-mxnet-with-the-nnpack-library-7427f367490f
https://medium.com/@julsimon/speeding-up-apache-mxnet-with-the-nnpack-library-raspberry-pi-edition-e444b446a180

Optimizing model size

- Complex neural networks are too large for resource-constrained environments (memory, power)

- Networks can shrink without losing accuracy
  - Song Han et al, "Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding", 2016
  - Zhu et al, "To prune, or not to prune: exploring the efficacy of pruning for model compression", 2017

Optimizing models with Mixed Precision Training

Almost 2x reduction in memory consumption

No loss of accuracy

Model	Baseline	Mixed Precision
AlexNet	56.77%	56.93%
VGG-D	65.40%	65.43%
GoogleNet	68.33%	68.43%
Inception v1	70.03%	70.02%
Resnet50	73.61%	73.75%

Micikevicius et al, "Mixed Precision Training", 2017

MXNet supports Mixed Precision Training

- https://devblogs.nvidia.com/parallelforall/mixed-precision-training-deep-neural-networks/
- http://docs.nvidia.com/deeplearning/sdk/mixed-precision-training/index.html#mxnet

Gluon: Deep Learning gets even easier

https://github.com/gluon-api/

- Announced October 11th (yes, that's yesterday)
- Available now in MXNet, soon in Microsoft Cognitive Toolkit

- Developer-friendly high-level API
- Dynamic networks can be modified during training
- No compromise on performance
- Extensive model zoo

https://aws.amazon.com/blogs/aws/introducing-gluon-a-new-library-for-machine-learning-from-aws-and-microsoft/

Gluon Model Zoo

vgg11	VGG-11 model from the "Very Deep Convolutional Networks for Large-Scale Image Recognition" paper.
vgg13	VGG-13 model from the "Very Deep Convolutional Networks for Large-Scale Image Recognition" paper.
vgg16	VGG-16 model from the "Very Deep Convolutional Networks for Large-Scale Image Recognition" paper.
vgg19	VGG-19 model from the "Very Deep Convolutional Networks for Large-Scale Image Recognition" paper.
vgg11_bn	VGG-11 model with batch normalization from the "Very Deep Convolutional Networks for Large-Scale Image Recognition" paper.
vgg13_bn	VGG-13 model with batch normalization from the "Very Deep Convolutional Networks for Large-Scale Image Recognition" paper.
vgg16_bn	VGG-16 model with batch normalization from the "Very Deep Convolutional Networks for Large-Scale Image Recognition" paper.
vgg19_bn	VGG-19 model with batch normalization from the "Very Deep Convolutional Networks for Large-Scale Image Recognition" paper.

VGG	VGG model from the "Very Deep Convolutional Networks for Large-Scale Image Recognition" paper.
get_vgg	VGG model from the "Very Deep Convolutional Networks for Large-Scale Image Recognition" paper.

resnet18_v1	ResNet-18 V1 model from "Deep Residual Learning for Image Recognition" paper.
resnet34_v1	ResNet-34 V1 model from "Deep Residual Learning for Image Recognition" paper.
resnet50_v1	ResNet-50 V1 model from "Deep Residual Learning for Image Recognition" paper.
resnet101_v1	ResNet-101 V1 model from "Deep Residual Learning for Image Recognition" paper.
resnet152_v1	ResNet-152 V1 model from "Deep Residual Learning for Image Recognition" paper.
resnet18_v2	ResNet-18 V2 model from "Identity Mappings in Deep Residual Networks" paper.
resnet34_v2	ResNet-34 V2 model from "Identity Mappings in Deep Residual Networks" paper.
resnet50_v2	ResNet-50 V2 model from "Identity Mappings in Deep Residual Networks" paper.
resnet101_v2	ResNet-101 V2 model from "Identity Mappings in Deep Residual Networks" paper.
resnet152_v2	ResNet-152 V2 model from "Identity Mappings in Deep Residual Networks" paper.

ResNetV1	ResNet V1 model from "Deep Residual Learning for Image Recognition" paper.
ResNetV2	ResNet V2 model from "Identity Mappings in Deep Residual Networks" paper.
BasicBlockV1	BasicBlock V1 from "Deep Residual Learning for Image Recognition" paper.
BasicBlockV2	BasicBlock V2 from "Identity Mappings in Deep Residual Networks" paper.
BottleneckV1	Bottleneck V1 from "Deep Residual Learning for Image Recognition" paper.
BottleneckV2	Bottleneck V2 from "Identity Mappings in Deep Residual Networks" paper.
get_resnet	ResNet V1 model from "Deep Residual Learning for Image Recognition" paper.

mobilenet1_0	MobileNet model from the "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications" paper, with width multiplier 1.0.
mobilenet0_75	MobileNet model from the "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications" paper, with width multiplier 0.75.
mobilenet0_5	MobileNet model from the "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications" paper, with width multiplier 0.5.
mobilenet0_25	MobileNet model from the "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications" paper, with width multiplier 0.25.

MobileNet	MobileNet model from the "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications" paper.

densenet121	Densenet-BC 121-layer model from the "Densely Connected Convolutional Networks" paper.
densenet161	Densenet-BC 161-layer model from the "Densely Connected Convolutional Networks" paper.
densenet169	Densenet-BC 169-layer model from the "Densely Connected Convolutional Networks" paper.
densenet201	Densenet-BC 201-layer model from the "Densely Connected Convolutional Networks" paper.

DenseNet	Densenet-BC model from the "Densely Connected Convolutional Networks" paper.

inception_v3	Inception v3 model from "Rethinking the Inception Architecture for Computer Vision" paper.

Inception3	Inception v3 model from "Rethinking the Inception Architecture for Computer Vision" paper.

alexnet	AlexNet model from the "One weird trick..." paper.

AlexNet	AlexNet model from the "One weird trick..." paper.

squeezenet1_0	SqueezeNet 1.0 model from the "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size" paper.
squeezenet1_1	SqueezeNet 1.1 model from the official SqueezeNet repo.

SqueezeNet	SqueezeNet model from the "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size" paper.

VGG
ResNet
AlexNet
DenseNet
SqueezeNet
Inception
MobileNet

https://mxnet.incubator.apache.org/versions/master/api/python/gluon/model_zoo.html

AWS Deep Learning AMI

- Deep Learning Frameworks – 5 popular Deep Learning Frameworks (mxnet, Caffe, Tensorflow, Theano, and Torch) all prebuilt and pre-installed
- Pre-installed components – Nvidia drivers, cuDNN, Anaconda, Python2 and Python3
- AWS Integration – Packages and configurations that provide tight integration with Amazon Web Services like Amazon EFS (Elastic File System)
- Amazon Linux & Ubuntu

https://aws.amazon.com/about-aws/whats-new/2017/03/deep-learning-ami-release-v2-0-now-available-for-amazon-linux/

Apache MXNet demos

1. Image classification: using pre-trained models
   Imagenet, multiple CNNs, MXNet
2. Image classification: fine-tuning a pre-trained model
   CIFAR-10, ResNet-50, Keras + MXNet
3. Image classification: learning from scratch
   MNIST, MLP & LeNet, MXNet
4. Machine Translation: translating German to English
   News, LSTM, Sockeye + MXNet
5. AI! IoT! Robots!

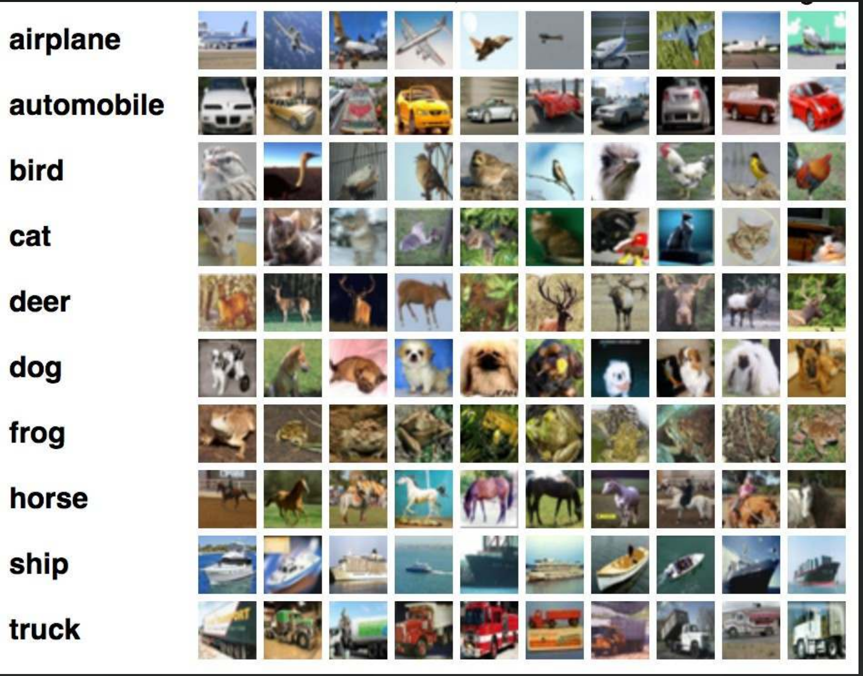
Demo #1 – Image classification: using a pre-trained model

```
*** VGG16
[(0.46811387, 'n04296562 stage'), (0.24333163,
'n03272010 electric guitar'), (0.045918692, 'n02231487
walking stick, walkingstick, stick insect'),
(0.03316205, 'n04286575 spotlight, spot'),
(0.021694135, 'n03691459 loudspeaker, speaker, speaker
unit, loudspeaker system, speaker system')]
*** ResNet-152
[(0.8726753, 'n04296562 stage'), (0.046159592,
'n03272010 electric guitar'), (0.041658506, 'n03759954
microphone, mike'), (0.018624334, 'n04286575 spotlight,
spot'), (0.0058045341, 'n02676566 acoustic guitar')]
*** Inception v3
[(0.44991142, 'n04296562 stage'), (0.43065304,
'n03272010 electric guitar'), (0.067580454, 'n04456115
torch'), (0.012423956, 'n02676566 acoustic guitar'),
(0.0093934005, 'n03250847 drumstick')]
```

https://medium.com/@julsimon/an-introduction-to-the-mxnet-api-part-5-9e78534096db

Demo #2 – Image classification: fine-tuning a model

- CIFAR-10 data set
  - 60,000 images in 10 classes
  - 32x32 color images
- Initial training
  - Resnet-50 CNN
  - 200 epochs
  - 82.12% validation
- Cars vs. horses
  - 88.8% validation accuracy



https://medium.com/@julsimon/keras-shoot-out-part-3-fine-tuning-7d1548c51a41

aws

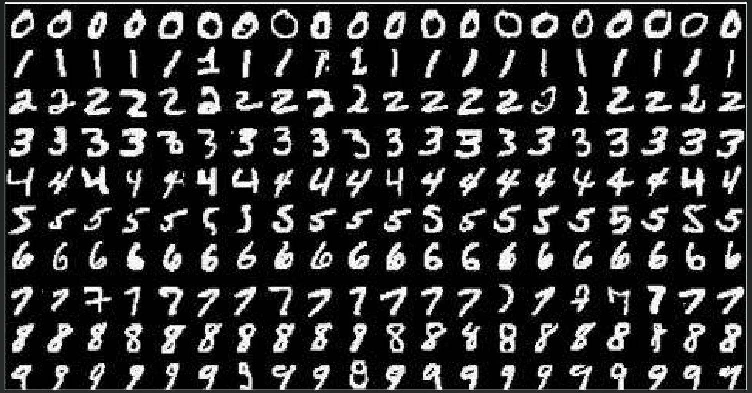
Demo #2 – Image classification: fine-tuning a model

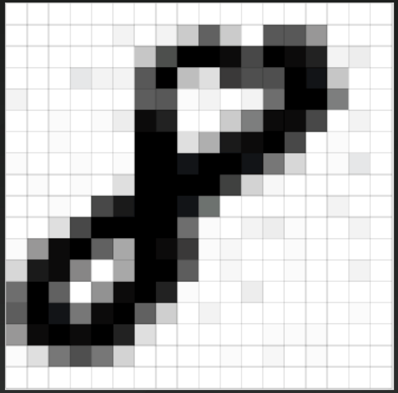
- Freezing all layers but the last one
- Fine-tuning on « cars vs. horses » for 10 epochs
- 2 minutes on 1 GPU
- 98.8% validation accuracy

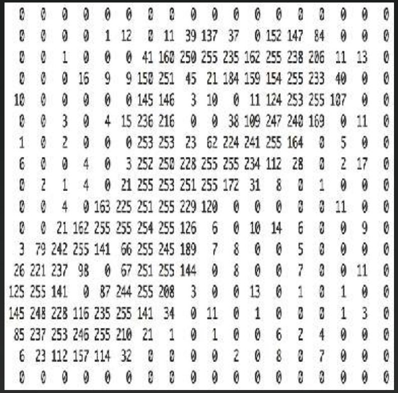
```
Epoch 10/10
10000/10000 [==============================] - 12s
loss: 1.6989 - acc: 0.9994 - val_loss: 1.7490 - val_acc: 0.9880

2000/2000 [==============================] - 2s
[1.7490020694732666, 0.98799999999999999]
```

Demo #3 – Image classification: learning from scratch

- MNIST data set
- 70,000 hand-written digits
- 28x28 grayscale images



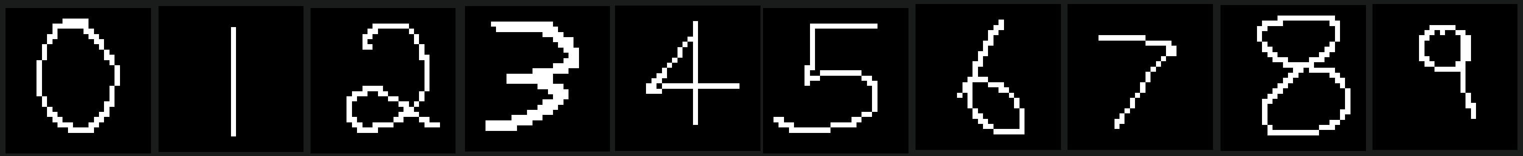




https://medium.com/@julsimon/training-mxnet-part-1-mnist-6f0dc4210c62

Multi-Layer Perceptron vs. Handmade-Digits-From-Hell™

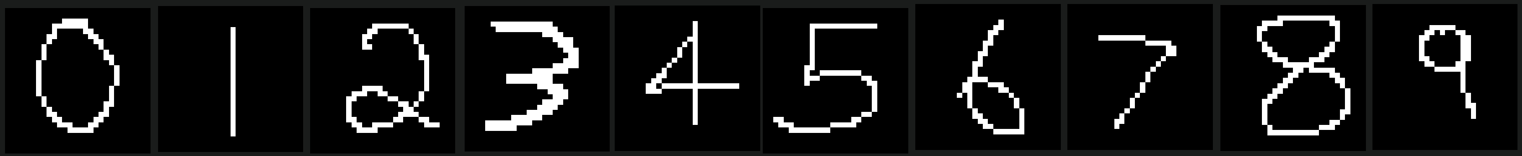
784/128/64/10, Relu, AdaGrad, 100 epochs → 97.51% validation accuracy



```
[[ 0.839  0.034  0.039  0.009  0.     0.008  0.066  0.002  0.     0.004]]
[[ 0.     0.988  0.001  0.003  0.001  0.001  0.002  0.003  0.001  0.002]]
[[ 0.006  0.01   0.95   0.029  0.     0.001  0.004  0.     0.     0.]]
[[ 0.     0.     0.     1.     0.     0.     0.     0.     0.     0.]]
[[ 0.     0.001  0.005  0.001  0.982  0.001  0.     0.007  0.     0.002]]
[[ 0.001  0.001  0.     0.078  0.     0.911  0.01   0.     0.     0.]]
[[ 0.003  0.     0.019  0.     0.005  0.004  0.863  0.     0.105  0.001]]
[[ 0.001  0.008  0.098  0.033  0.     0.     0.     0.852  0.004  0.004]]
[[ 0.001  0.     0.006  0.     0.     0.001  0.002  0.     0.991  0.]]
[[ 0.002  0.158  0.007  0.117  0.082  0.001  0.     0.239  0.17   0.224]]
```

LeNet vs. Handmade-Digits-From-Hell™

ReLu instead of tanh, 20 epochs, AdaGrad → 99.20% validation accuracy



```
[[ 1.     0.     0.     0.     0.     0.     0.     0.     0.     0.]]
[[ 0.     1.     0.     0.     0.     0.     0.     0.     0.     0.]]
[[ 0.     0.     1.     0.     0.     0.     0.     0.     0.     0.]]
[[ 0.     0.     0.     1.     0.     0.     0.     0.     0.     0.]]
[[ 0.     0.     0.001  0.     0.998  0.     0.     0.001  0.     0.]]
[[ 0.     0.     0.     0.     0.     1.     0.     0.     0.     0.]]
[[ 0.     0.     0.     0.     0.     0.     1.     0.     0.     0.]]
[[ 0.     0.     0.     0.001  0.     0.     0.     0.999  0.     0.]]
[[ 0.     0.     0.006  0.     0.     0.     0.     0.     0.994  0.]]
[[ 0.     0.     0.     0.001  0.001  0.     0.     0.001  0.001  0.996]]
```

Demo #4 – Machine Translation: German to English

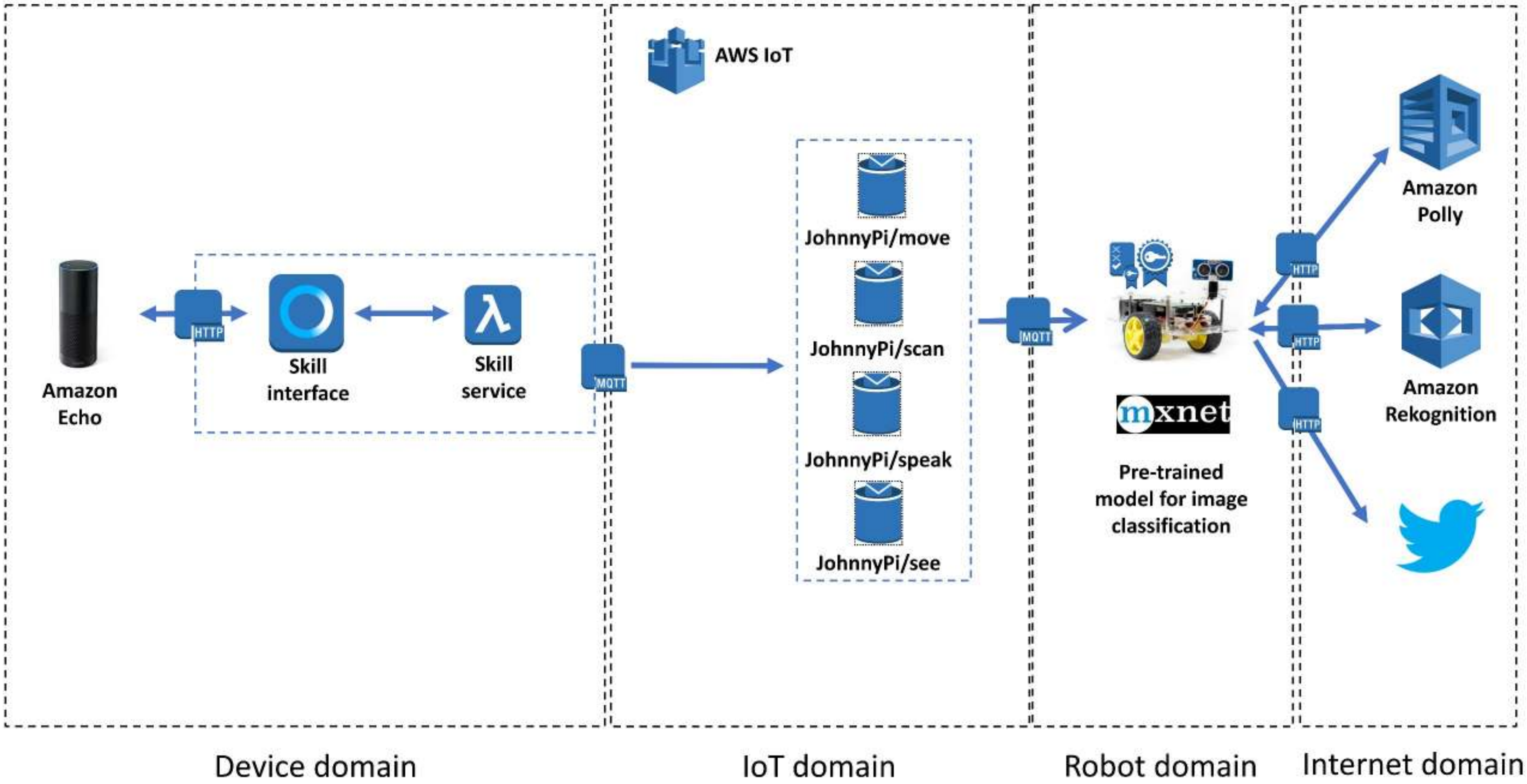
- AWS Open Source project https://github.com/awslabs/sockeye
- Sequence-to-sequence models with Apache MXNet
- 5.8M sentences (news headlines), 5 hours of training on 8 GPUs

```
./translate.sh "Chopin zählt zu den bedeutendsten Persönlichkeiten der
Musikgeschichte Polens .”

Chopin is one of the most important personalities of Poland’s history

./translate.sh "Hotelbetreiber müssen künftig nur den Rundfunkbeitrag bezahlen,
wenn ihre Zimmer auch eine Empfangsmöglichkeit bieten .”

in the future , hotel operators must pay only the broadcasting fee if their rooms
also offer a reception facility .
```

https://aws.amazon.com/blogs/ai/train-neural-machine-translation-models-with-sockeye/

Demo #5 – AI! IoT! Robots!

https://medium.com/@julsimon/johnny-pi-i-am-your-father-part-0-1eb537e5a36



Anything you dream is fiction, and anything you accomplish is science, the whole history of mankind is nothing but science fiction.

Ray Bradbury

Resources

https://aws.amazon.com/ai/
https://aws.amazon.com/blogs/ai/
https://mxnet.io
https://github.com/gluon-api/
https://github.com/awslabs/sockeye

https://medium.com/@julsimon/getting-started-with-deep-learning-and-apache-mxnet-34a978a854b4

Thank you!

https://aws.amazon.com/evangelists/julien-simon

@julsimon