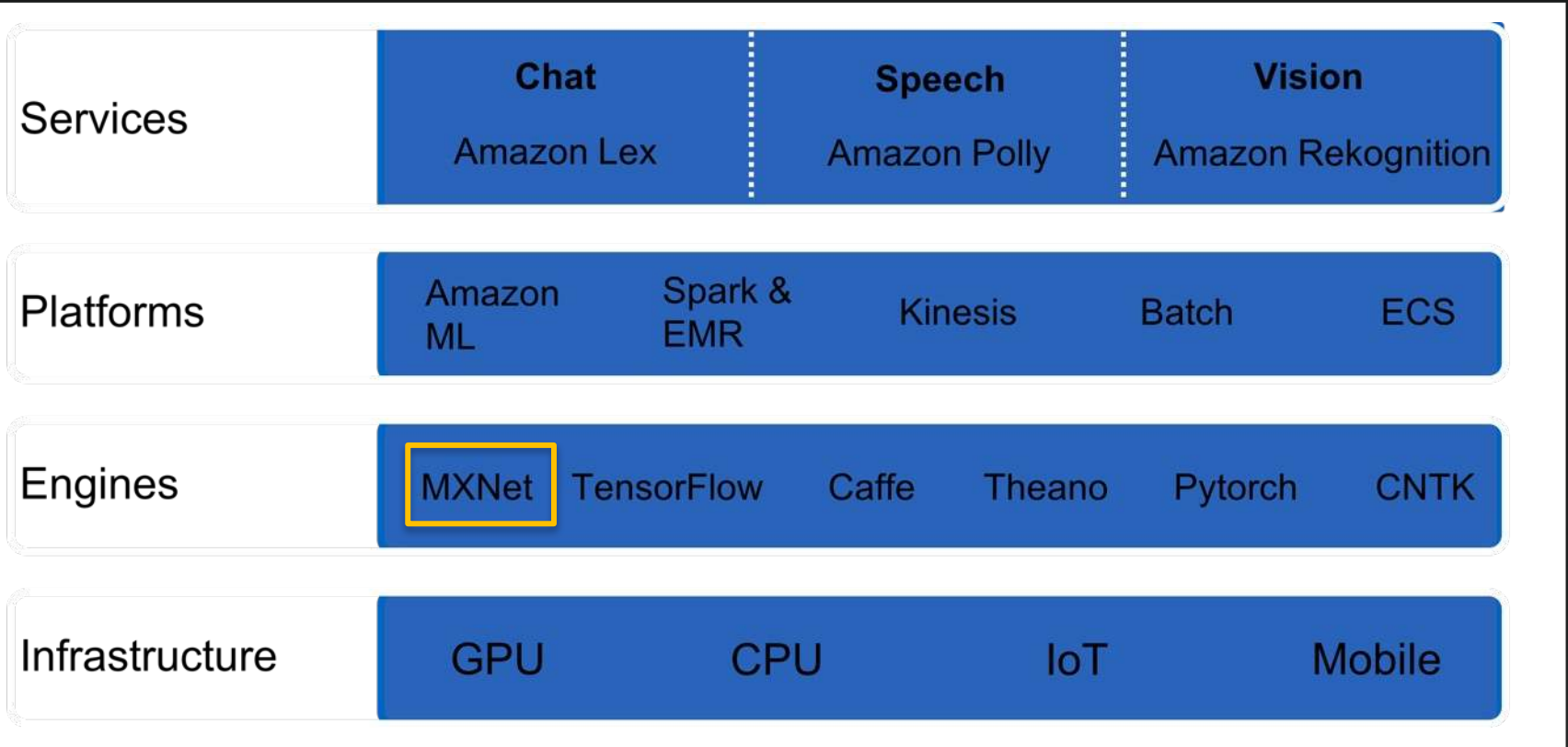
Deep Learning for Developers

Julien Simon, Principal Technical Evangelist
@julsimon

aws

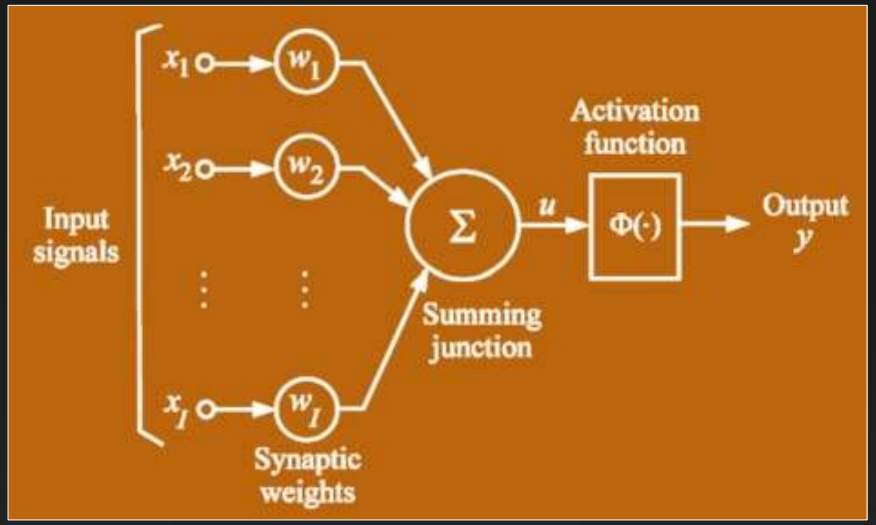
Amazon AI for every developer



More information at aws.amazon.com/ai

Neural Networks

The neuron



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

x = [x_1, x_2, …. x_I]

w = [w_1, w_2, …. w_I]

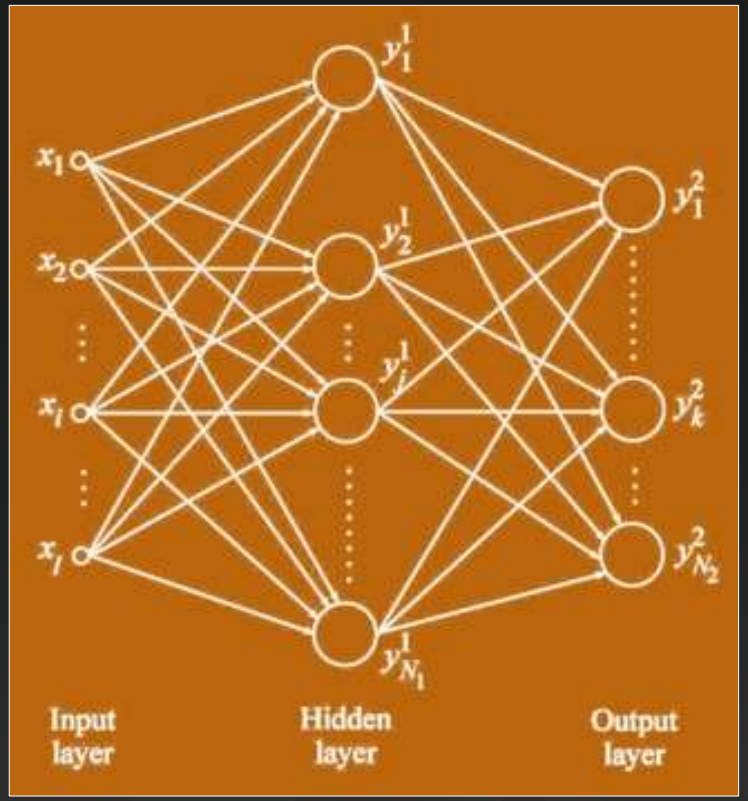
u =

x.w

Activation function

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]		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$

The neural network



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

l features

m samples

$$y = \begin{bmatrix} y_1 \\ y_2 \\ y_{\ldots} \\ y_m \end{bmatrix}$$

m labels

Accurac
y =

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

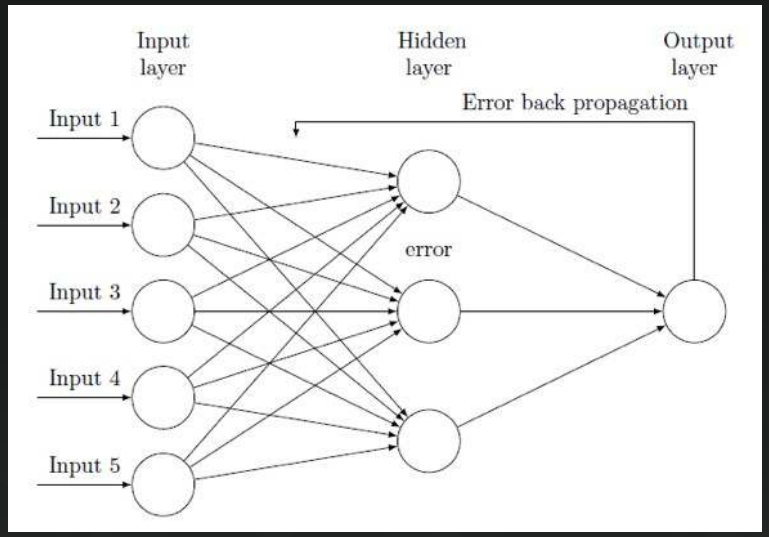
The training process

- 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

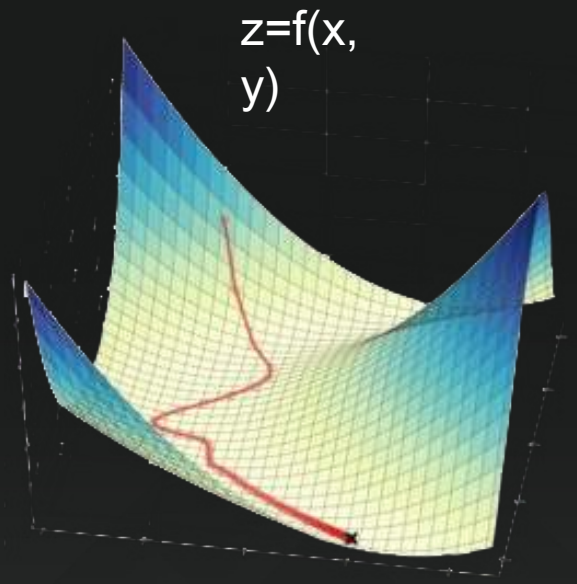


The back-propagation algorithm acts as an error correcting mechanism at each neuron level, thereby helping the network to learn effectively.

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



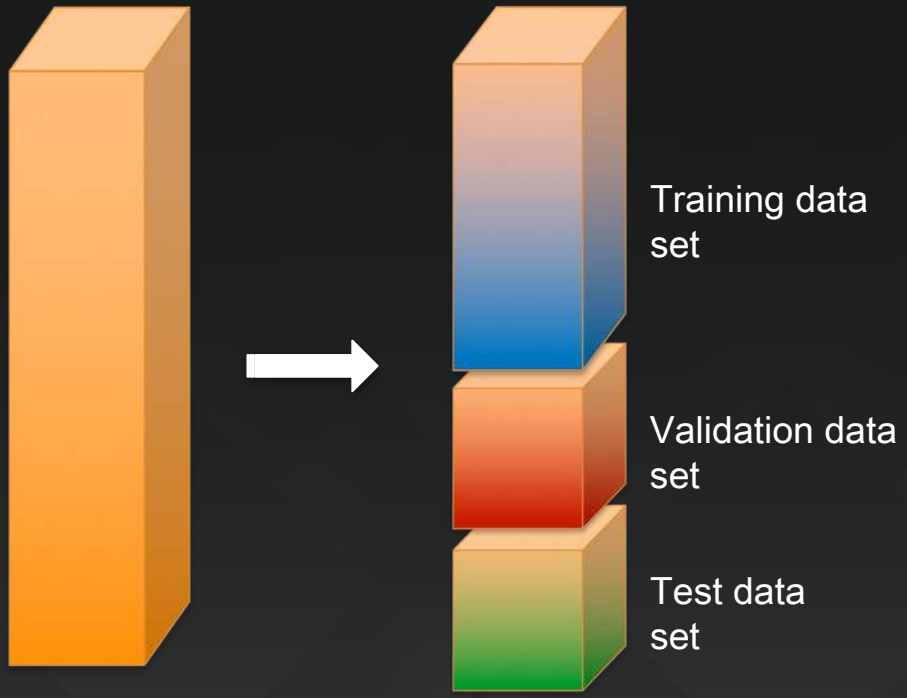
The « step size » is called the learning rate

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

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

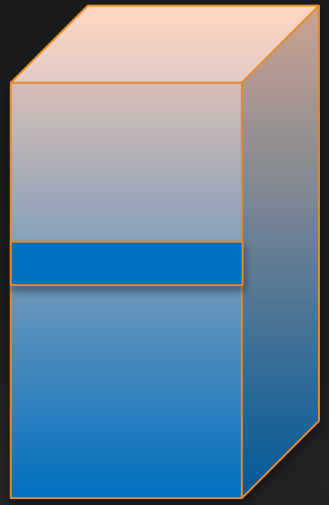


Training set: this data set is used to adjust the weights on the neural network.

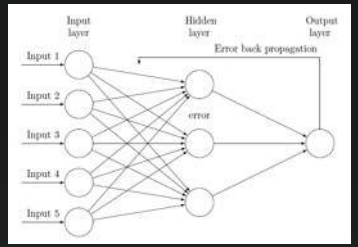
Validation set: this data set is used to minimize overfitting. You're not adjusting the weights of the network with this data set, you're just verifying that any increase in accuracy over the training data set actually yields an increase in accuracy over a data set that has not been shown to the network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



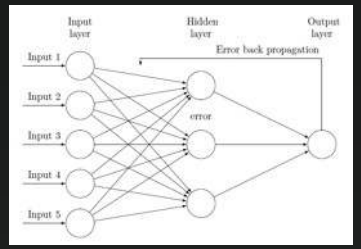
Training data set



Training

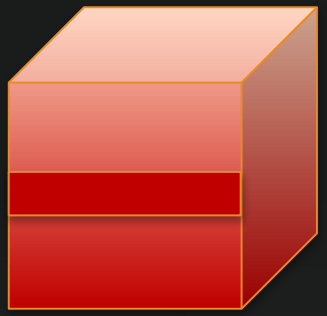
Number of epochs
Batch size
Learning rate

Hyper parameters

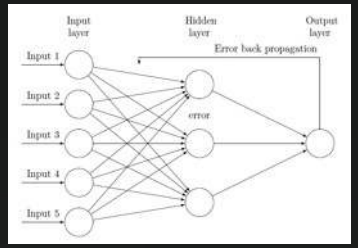


Trained neural network

Validation



Validation Data set



Trained neural network

Prediction at the end of each epoch

Validation accuracy

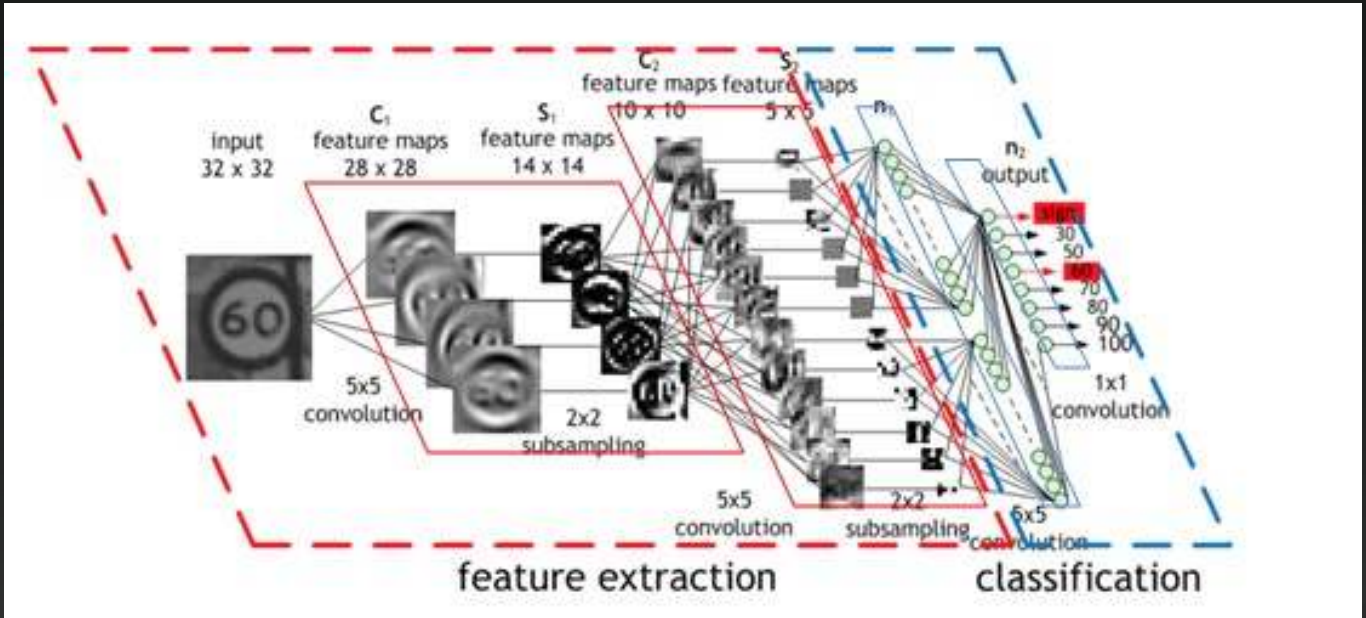
Stop training when validation accuracy stops increasing

Saving parameters at the end of each epoch

Convolutional Neural Networks

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

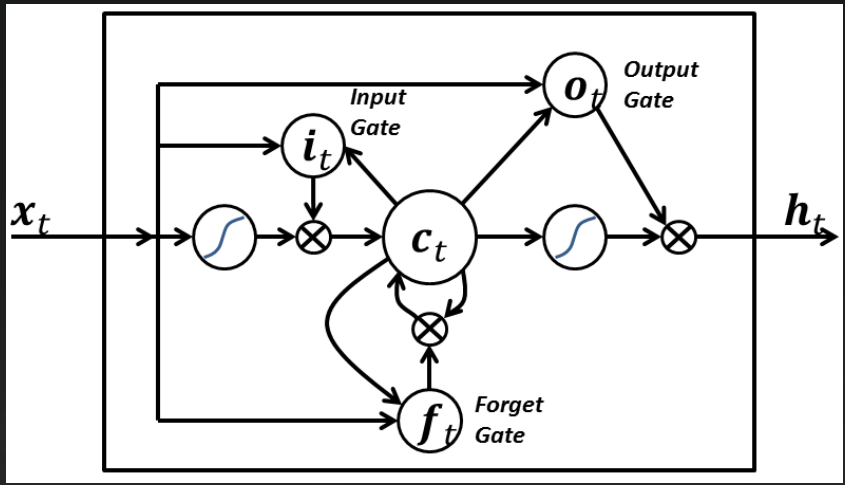
Feature extraction and downsampling allow smaller networks



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

mx.model.FeedForward

model.fit

Input

Output

mx.sym.SoftmaxOutput

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

"sigmoid"

"tanh"

"relu"

"softrelu"

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),

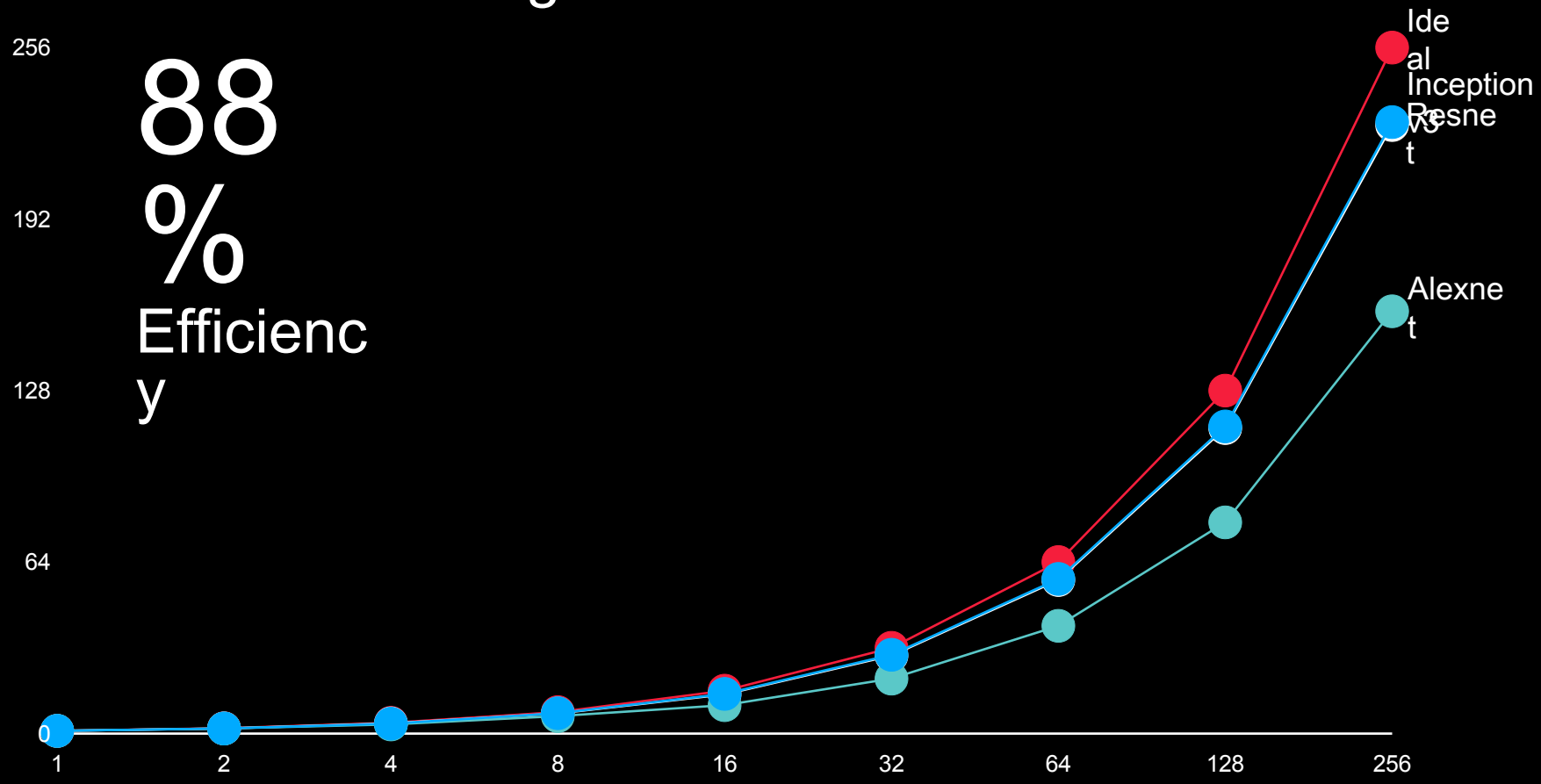
stride=(2,2)

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

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

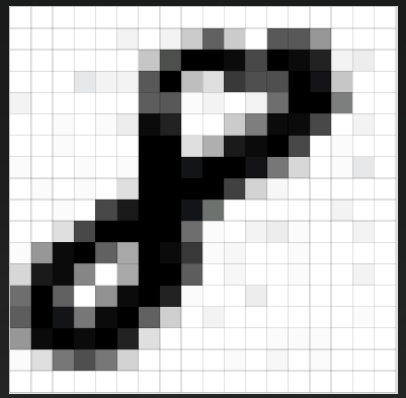
mx.symbol.**Embedding**(data, input_dim, output_dim = k)

Multi-GPU Scaling With MXNet



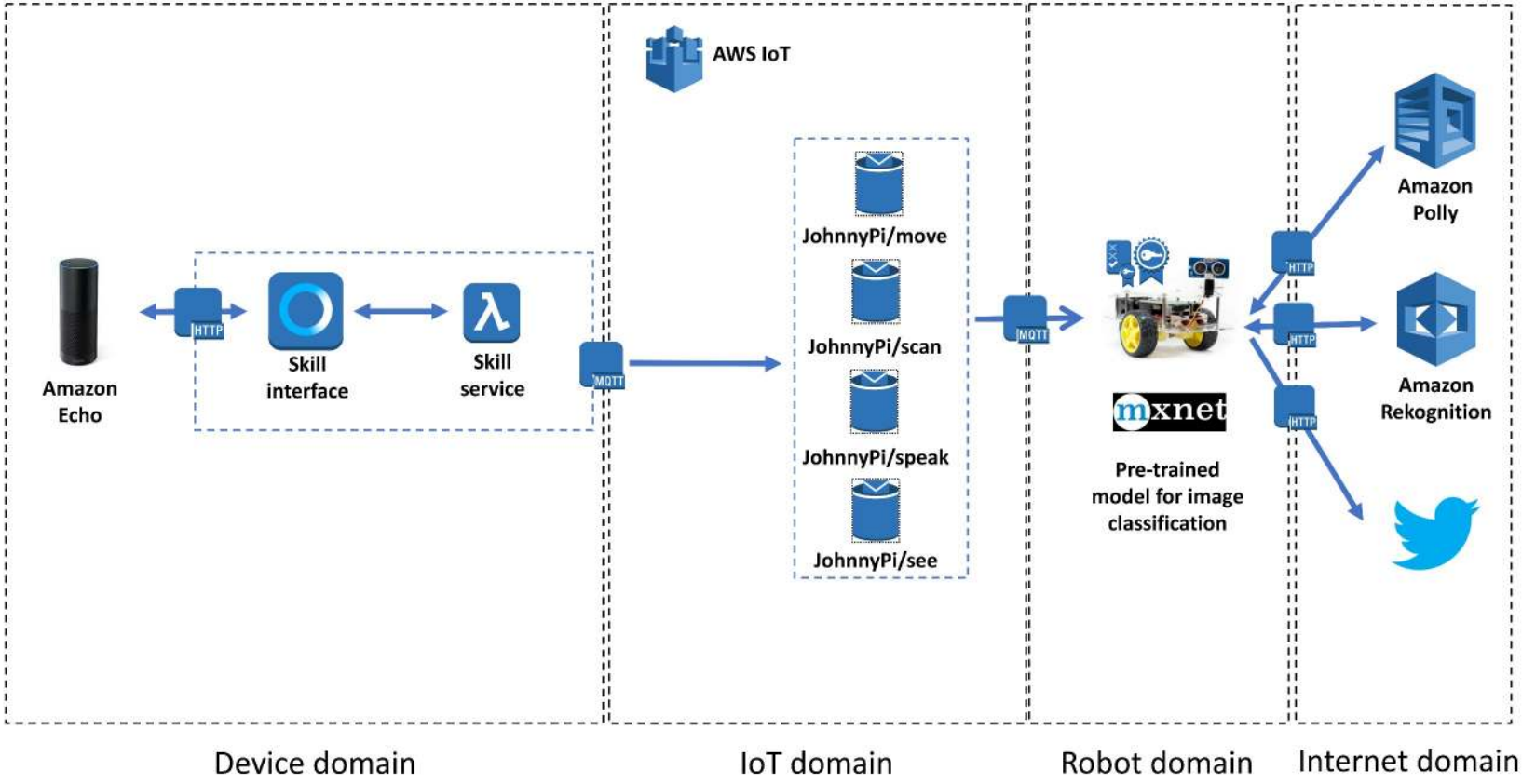
Demos

1. Image classification with pre-trained models (CNN)
2. Image classification from scratch (MLP, CNN)
3. Machine Translation with Sockeye (LSTM)
4. AI, IoT and robots



0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	12	0	11	39	137	37	0	152	147	84	0	0	0		
0	0	1	0	0	0	41	160	250	255	235	162	255	238	206	11	13	0		
0	0	0	16	9	9	158	251	45	21	184	158	154	255	233	40	0	0		
10	0	0	0	0	0	145	146	3	10	0	11	124	253	255	107	0	0		
0	0	3	0	4	15	236	216	0	0	38	109	247	248	169	0	11	0		
1	0	2	0	0	0	253	253	23	62	224	241	255	164	0	5	0	0		
6	0	0	4	0	3	252	250	228	255	255	234	112	28	0	2	17	0		
0	2	1	4	0	21	255	253	251	255	172	31	8	0	1	0	0	0		
0	0	4	0	163	225	251	255	229	120	0	0	0	0	0	11	0	0		
0	0	21	162	255	255	254	255	126	6	0	10	14	6	0	0	9	0		
3	79	242	255	141	66	255	245	189	7	8	0	0	5	0	0	0	0		
26	221	237	98	0	67	251	255	144	0	8	0	0	7	0	0	11	0		
125	255	141	0	87	244	255	208	3	0	0	13	0	1	0	1	0	0		
145	248	228	116	235	255	141	34	0	11	0	1	0	0	0	1	3	0		
85	237	253	246	255	210	21	1	0	1	0	0	6	2	4	0	0	0		
6	23	112	157	114	32	0	0	0	0	2	0	0	0	7	0	0	0		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

AI, IoT and robots



Resources

http://mxnet.io

https://github.com/awslabs/sockeye

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

http://medium.com/@julsimon

https://github.com/juliensimon/aws/tree/master/mxnet

Thank you!

http://aws.amazon.com/evangelists/julien-simon
@julsimon