Introduction to APACHE MXNET & Deep LEARNING

Part 1: Intro to Deep Learning

break;

Part 2: MXNet ecosystem

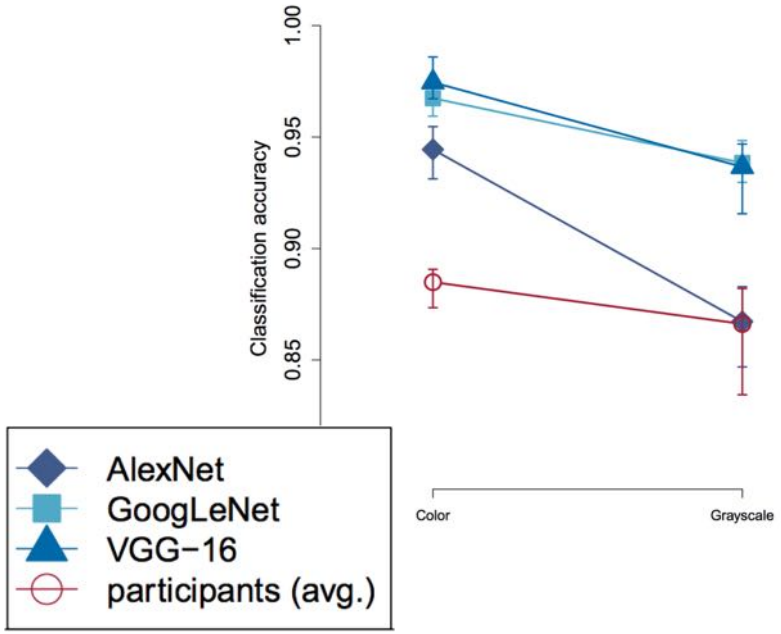
Why do machine learning?

- How many cats can you count?
- Complex tasks where you can’t code up explicit solutions

GPUs and CPUs

- GPUs have thousands of cores
- Can do matrix operations in parallel
- Ideal for Deep Learning: 100X faster than CPU
- NVIDIA GPUs with CUDA library commonly used
- Apache MXNet helps us manage
  - Moving data to/from GPU(s)
  - Doing GPU calculations
- CPU for data preprocessing, certain math functions
- Common: Train on GPU, predict on CPU

Deep Learning



When not to use machine learning?

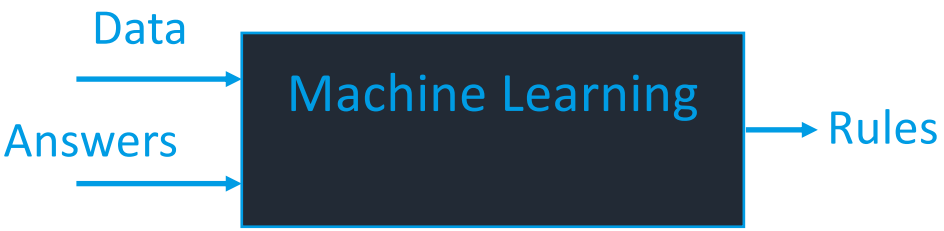
- USA Social Security numbers
- Stripping suffixes off English words
- Exact numerical solution required

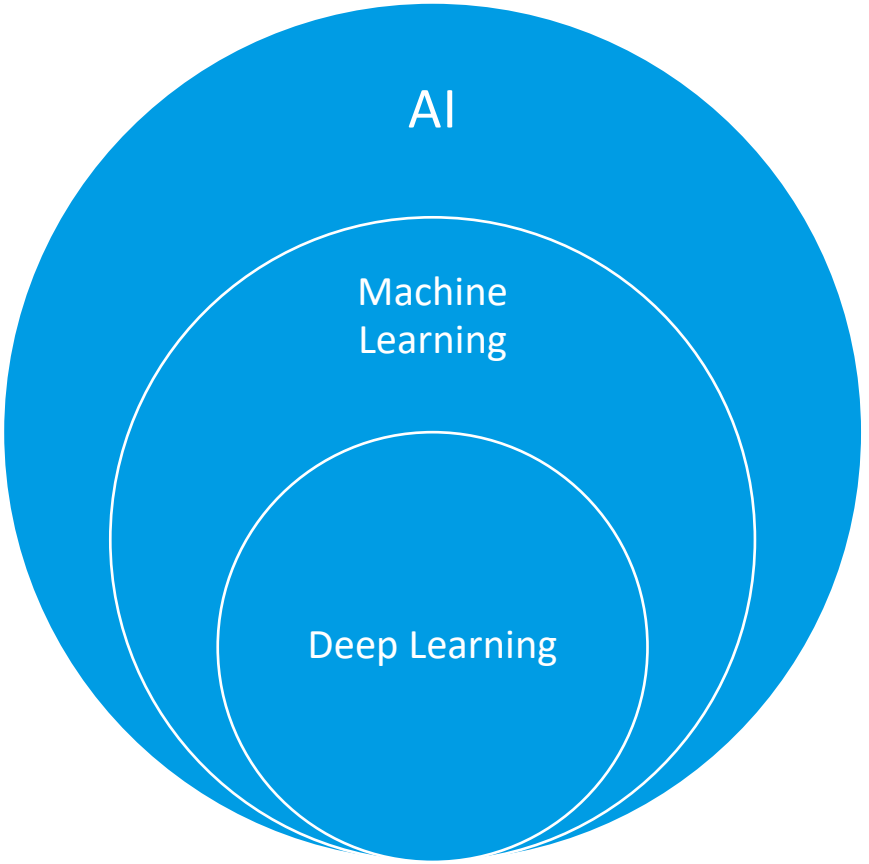
Situating Deep Learning

~~Can machines think?~~

Can machines do what we can?

(Turing, 1950)





Basic Terminology

Predict if a person earns > avg salary

Age	Education	Years of education	Marital status	Occupation	Sex	Label
39	Bachelors	16	Single	Adm-clerical	Male	-1
31	Masters	18	Married	Engineering	Female	+1

Training examples (rows)

Input features / x

Label / ground truth / y

Basic Terminology

Age	Education	Years of education	Marital status	Occupation	Sex	Label
39	Bachelors	16	Single	Adm-clerical	Male	-1
31	Masters	18	Married	Engineering	Female	+1

Age	Edu_Bachelors	Edu_Masters	Years of education	Marital_Single	…	Label
39	1	0	16	1	…	-1
31	0	1	18	0	…	+1

Types of Machine Learning

Supervised	• Data & labels
	  • Classification, Labeling
	  • Regression

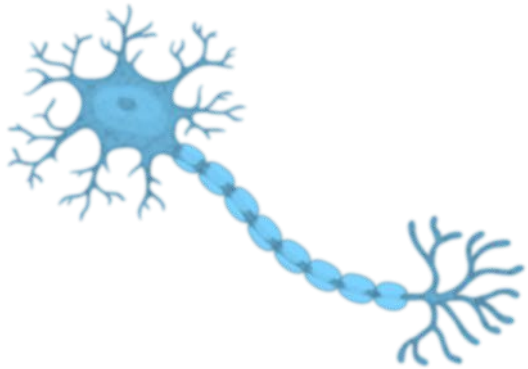
Unsupervised	• Data, no labels
	  • Clustering
	  • Dimensionality reduction

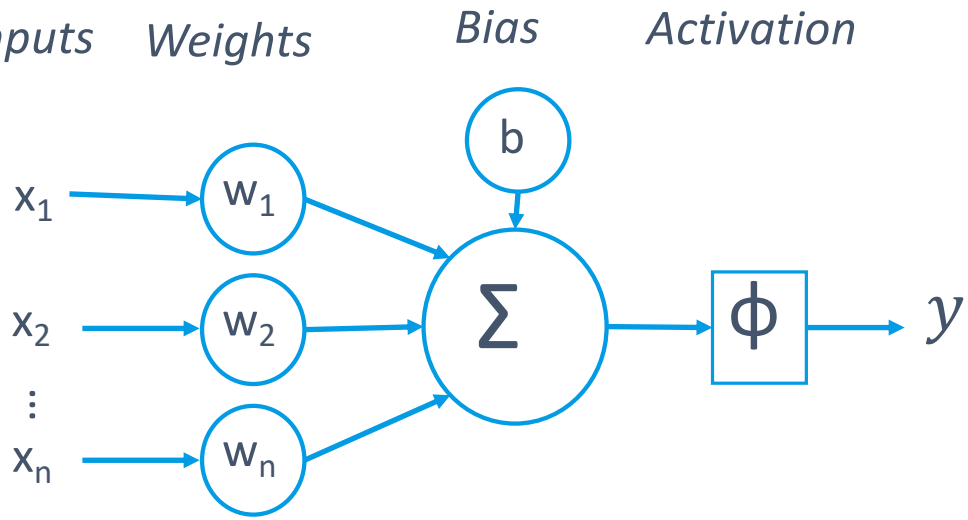
Semi-supervised	• Data, some labels • Active learning
Reinforcement learning	

(Artificial) Neural Networks (ANN)

Inspired by the brain's neurons

We have ~100B of them, and ~1Q Synapses

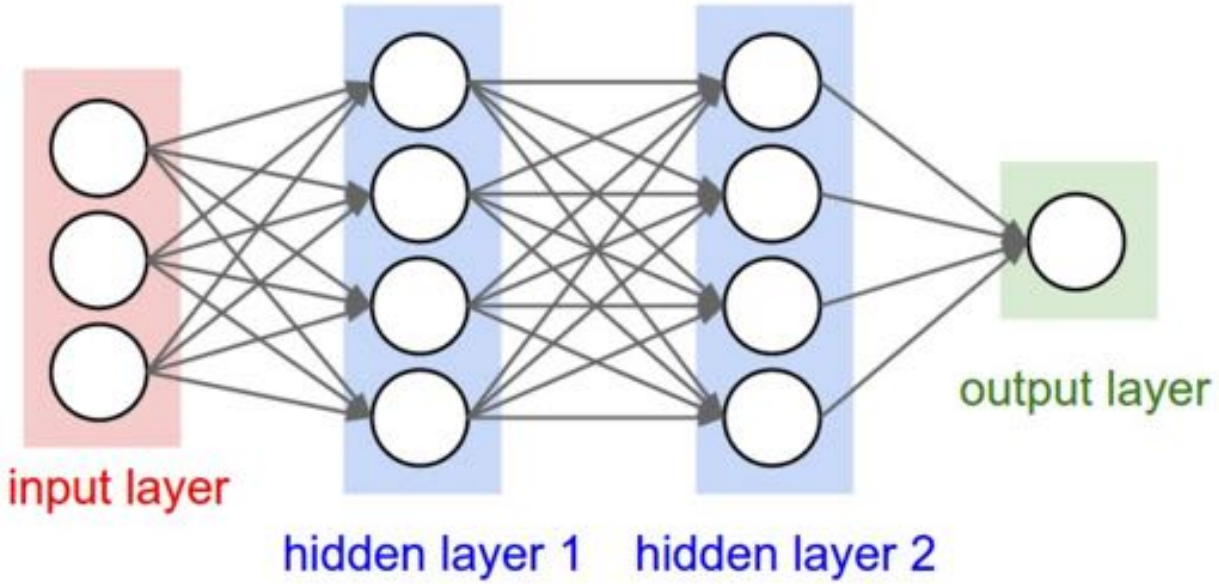


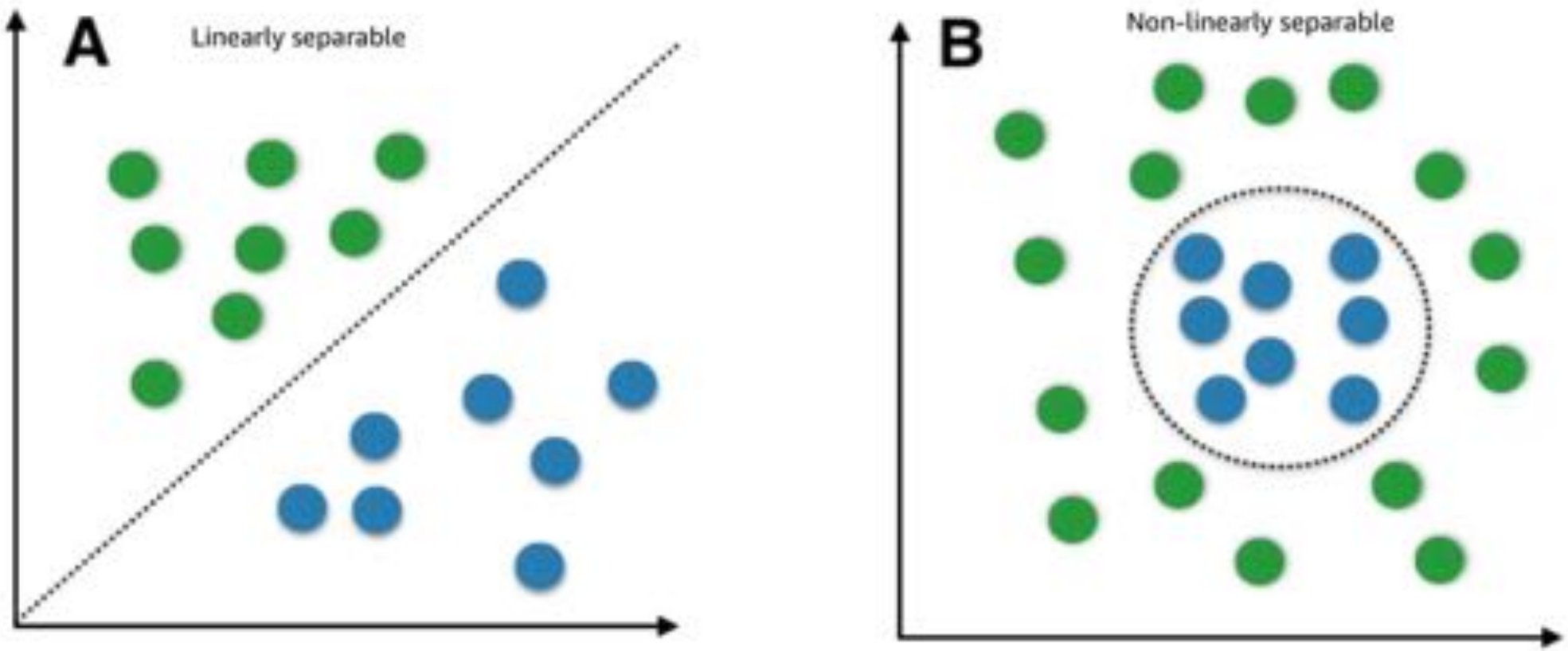


$$y = \varphi\left(\sum_{j=1}^{n} w_j x_j + b\right)$$

Deep Neural Networks

- Non-linear
- Hierarchical feature learning
- Scalable architecture

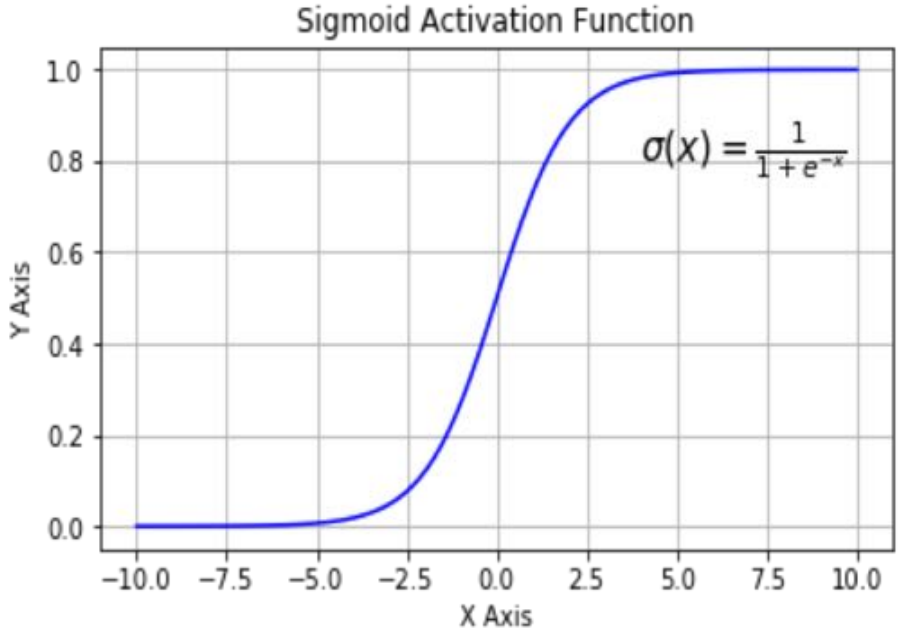


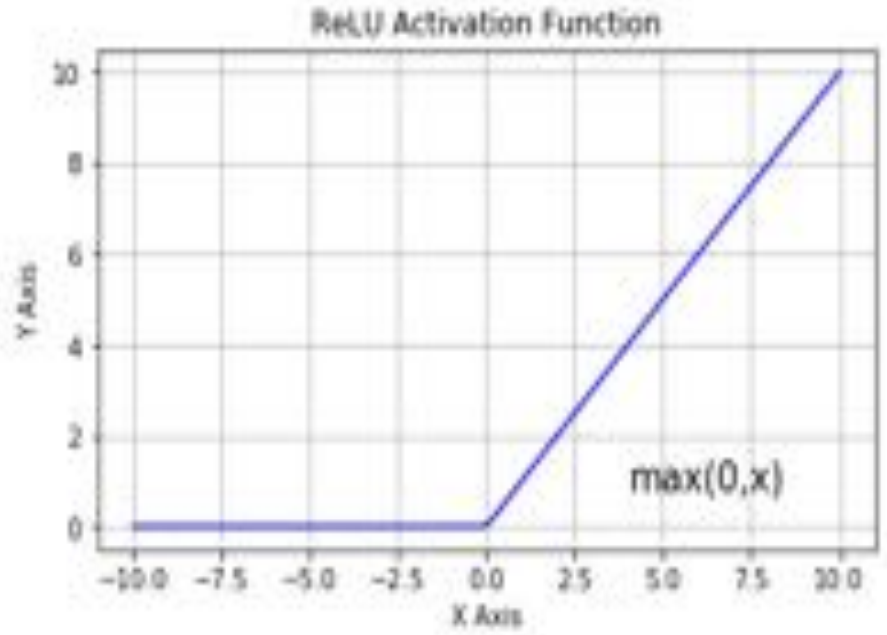

A
Linearly separable
B
Non-linearly separable

Activation Functions

- Determine how the neuron fires
- Represent non-linearity

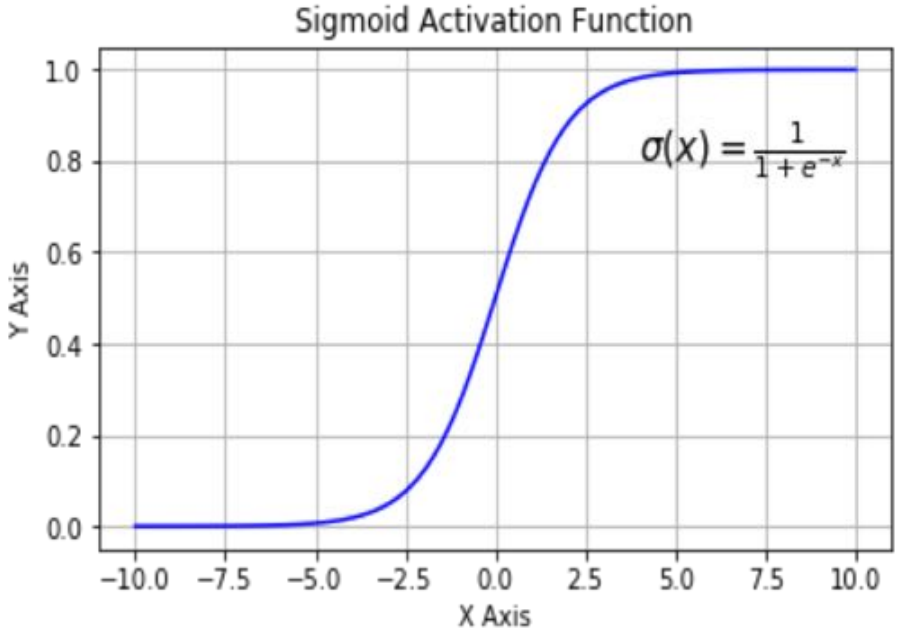
$$y = \varphi\left(\sum_{j=1}^{n} w_j x_j + b\right)$$





Bias term

- Each neuron has a bias associated with it
- Moves the activation left or right on x-axis

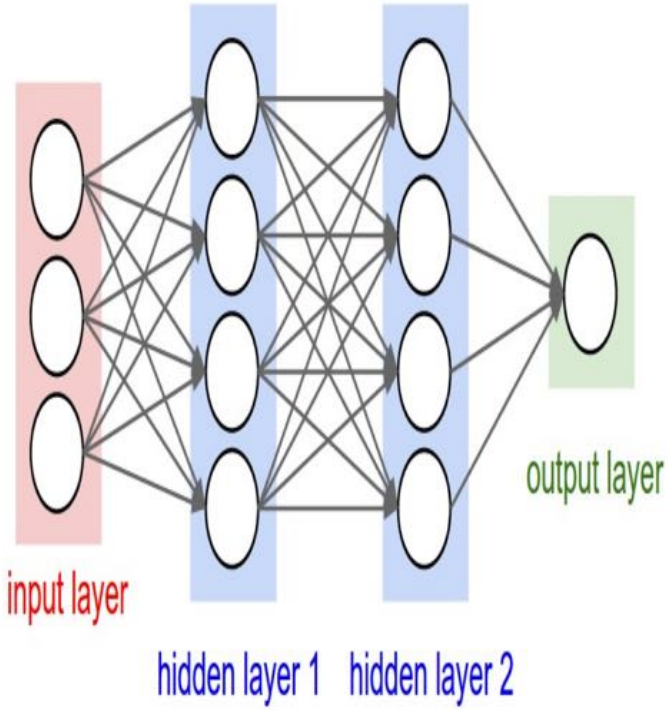


$$y = \varphi\left(\sum_{j=1}^{n} w_j x_j + b\right)$$

Parameters and Hyperparameters

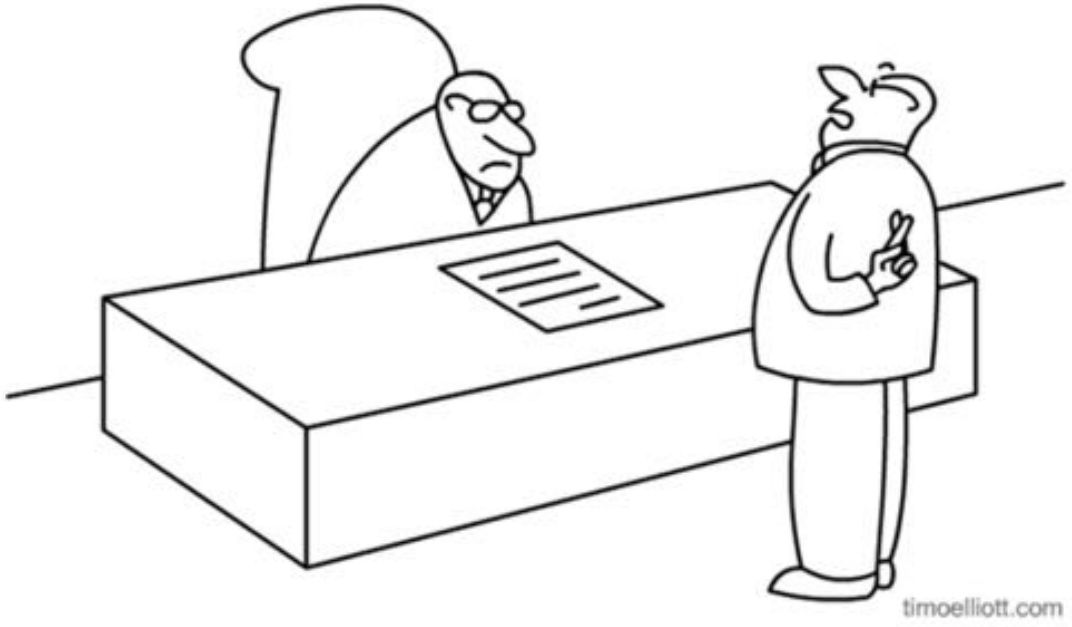
- Parameters
  - Numeric values in the model: weights and biases
  - Learned during training
- Hyperparameters
  - Values set for the training session
  - Numeric e.g. mini-batch size
  - Non-numeric e.g. which algorithm to use for optimization
- Hyperparameter optimization
  - Outer layer of

Training to estimate parameters



- One parameter per connection
- One bias per neuron
- Find the values that minimize the loss on training data

Trainig and optimization

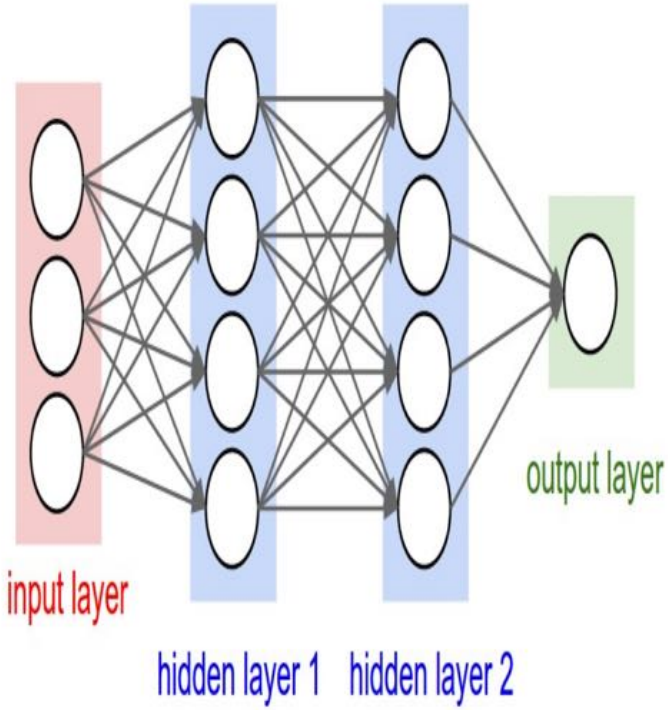


"Yes sir, you can absolutely trust those numbers"

Accuracy vs. Loss

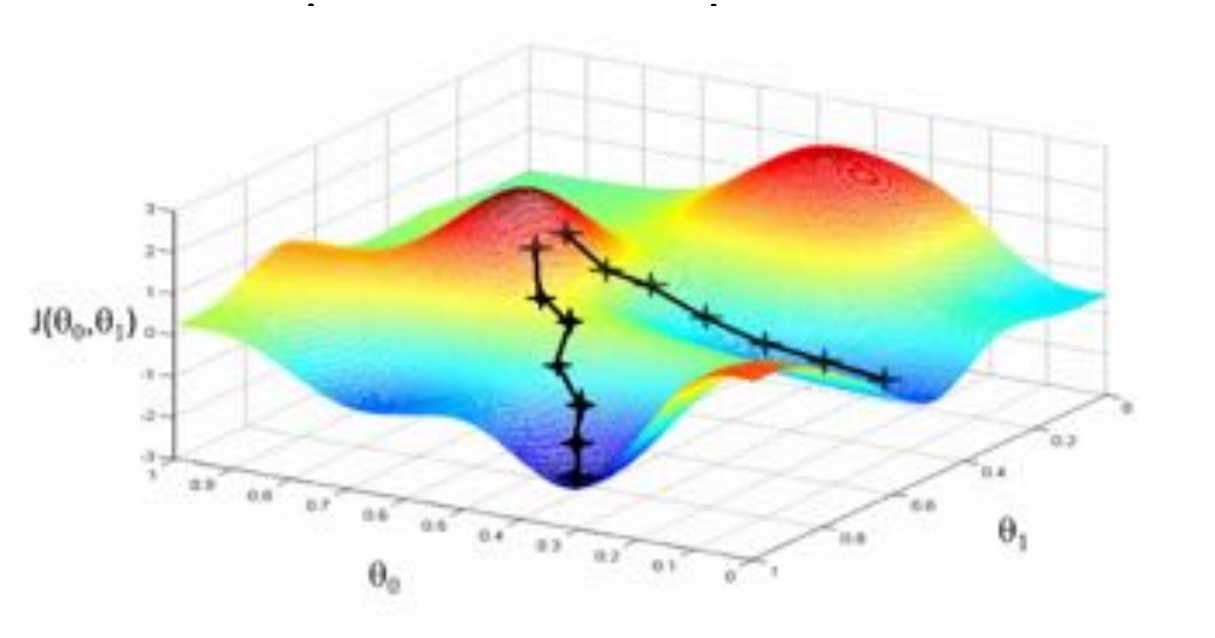
- Accuracy: A percentage
  - Correct or not per example; Not differentiable
- Loss: calculated during training
  - How far off is the current model?
  - Continuous value. Differentiable
- Common loss functions
  - Mean squared error (regression)
  - Cross entropy: log of difference in probability
- During training, minimize loss with an optimizer

Basic Optimizer: Stochastic Gradient Descent



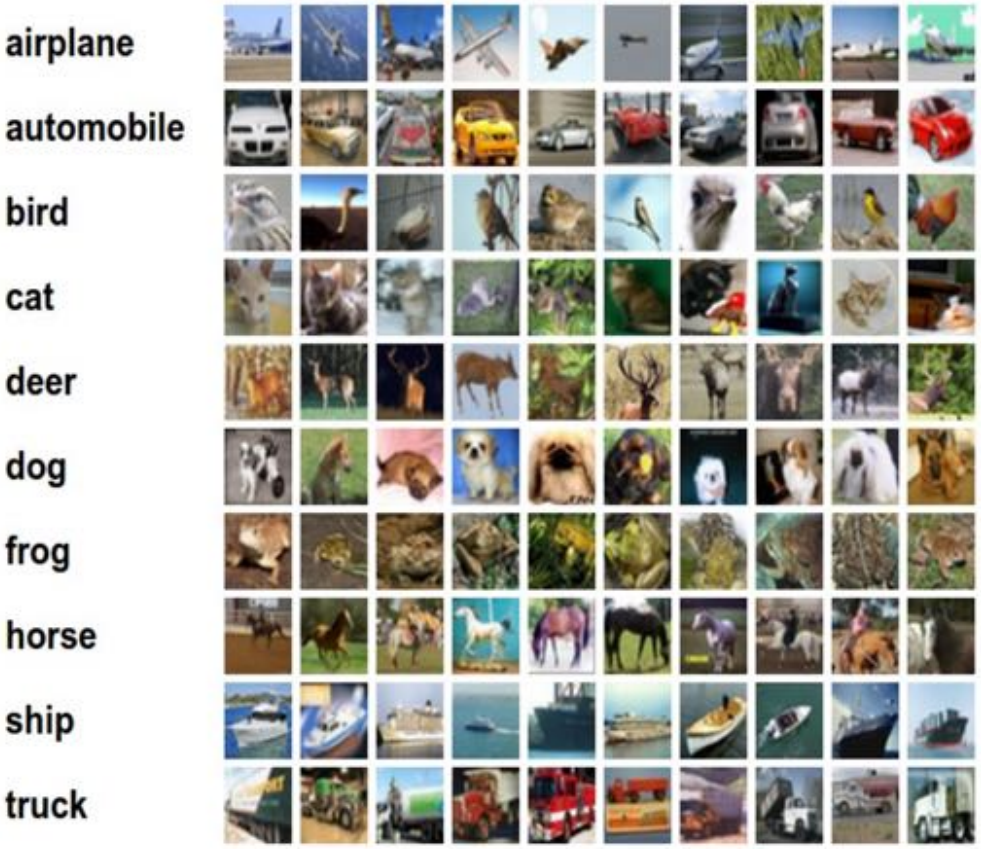
- Init params. Choose learning rate
- Divide data: train, validate, test. Shuffle training
- For each training example:
  - Feed forward, calculate output
  - Compare to truth, calculate loss
  - Calc. gradient for loss for each parameter

param = param – learning_rate *gradient



- Take a series of steps
- Specify a learning rate

Image Classification



CIFAR-10
10 labels
50,000 training images
each image is **32x32x3**
10,000 test images.

Softmax Classifier (Multinomial Logistic Regression)

cat	**3.2**
car	5.1
frog	-1.7

Softmax Classifier (Multinomial Logistic Regression)

Scores = unnormalized log prob. of the classes

$$s = f(x, W)$$

cat	**3.2**
car	5.1
frog	-1.7

Softmax Classifier (Multinomial Logistic Regression)

Scores = unnormalized log prob. of the classes

$$P(Y = k|X = x_i) = \frac{e^{s_{y_i}}}{\sum_j e^{s_j}} \quad \text{where} \quad s = f(x, W)$$

cat	**3.2**
car	5.1
frog	-1.7

Softmax Classifier (Multinomial Logistic Regression)

Scores = unnormalized log prob. of the classes

$$P(Y = k|X = x_i) = \frac{e^{s_{y_i}}}{\sum_j e^{s_j}} \quad \text{where} \quad s = f(x, W)$$

Softmax function

cat	**3.2**
car	5.1
frog	-1.7

Softmax Classifier (Multinomial Logistic Regression)

cat **3.2**

car 5.1

frog -1.7

Scores = unnormalized log prob. of the classes

$$P(Y = k|X = x_i) = \frac{e^{s_{y_i}}}{\sum_j e^{s_j}} \quad \text{where} \quad s = f(x, W)$$

Want to maximize the log likelihood, or (for a loss function) to minimize the negative log likelihood of the correct class:

$$L_i = -\log P(Y = y_i|X = x_i)$$

Softmax Classifier (Multinomial Logistic Regression)

cat **3.2**

car 5.1

frog -1.7

Scores = unnormalized log prob. of the classes

$$P(Y = k|X = x_i) = \frac{e^{s_{y_i}}}{\sum_j e^{s_j}} \quad \text{where} \quad s = f(x, W)$$

Want to maximize the log likelihood, or (for a loss function) to minimize the negative log likelihood of the correct class:

$$L_i = -\log P(Y = y_i|X = x_i)$$

In summary:

$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

Softmax Classifier (Multinomial Logistic Regression)

$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

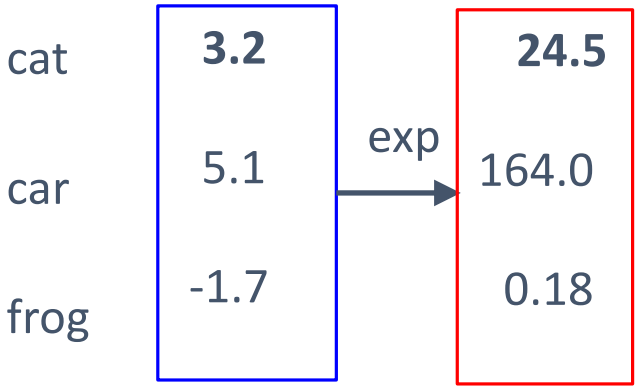
cat	**3.2**
car	5.1
frog	-1.7

unnormalized log probabilities

Softmax Classifier (Multinomial Logistic Regression)

$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

unnormalized probabilities

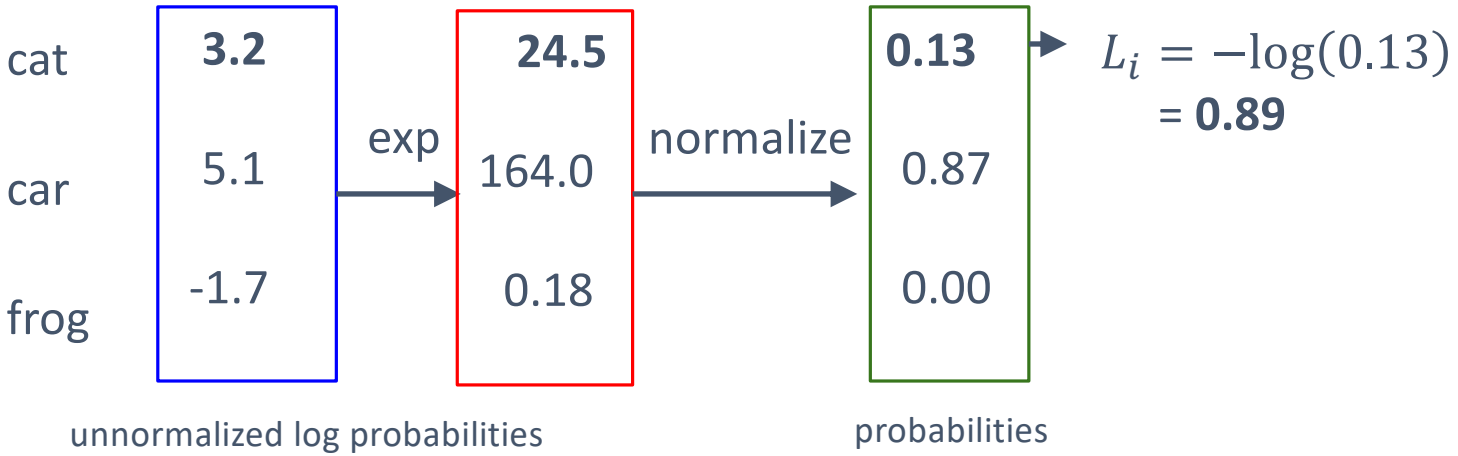


unnormalized log probabilities

Softmax Classifier (Multinomial Logistic Regression)

$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

unnormalized probabilities

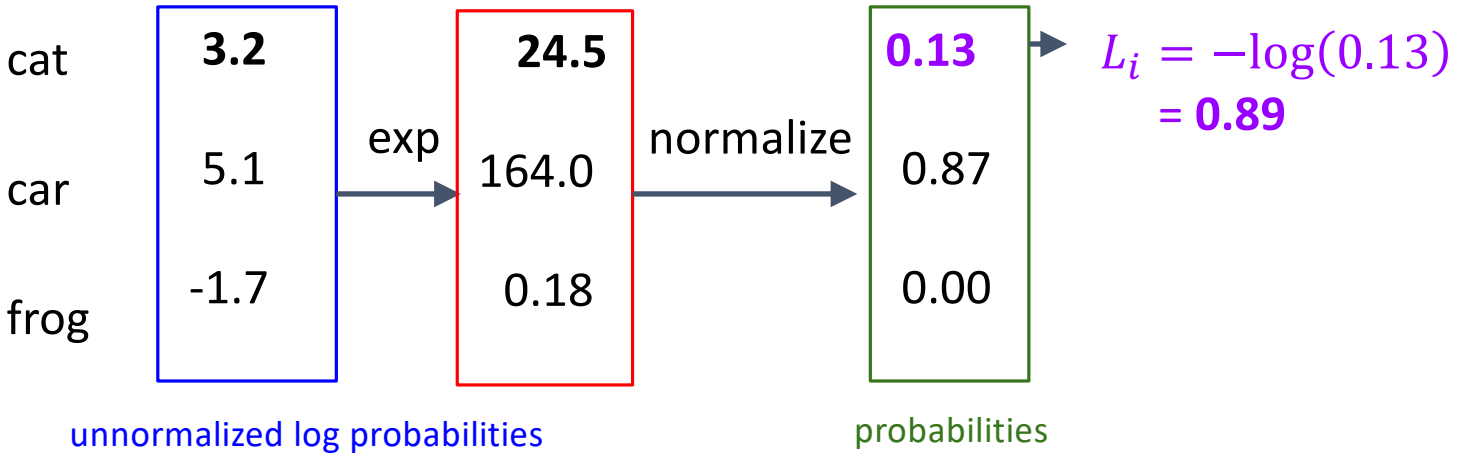


Softmax Classifier (Multinomial Logistic Regression)

$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

Q: What is the min/max possible L_i ?

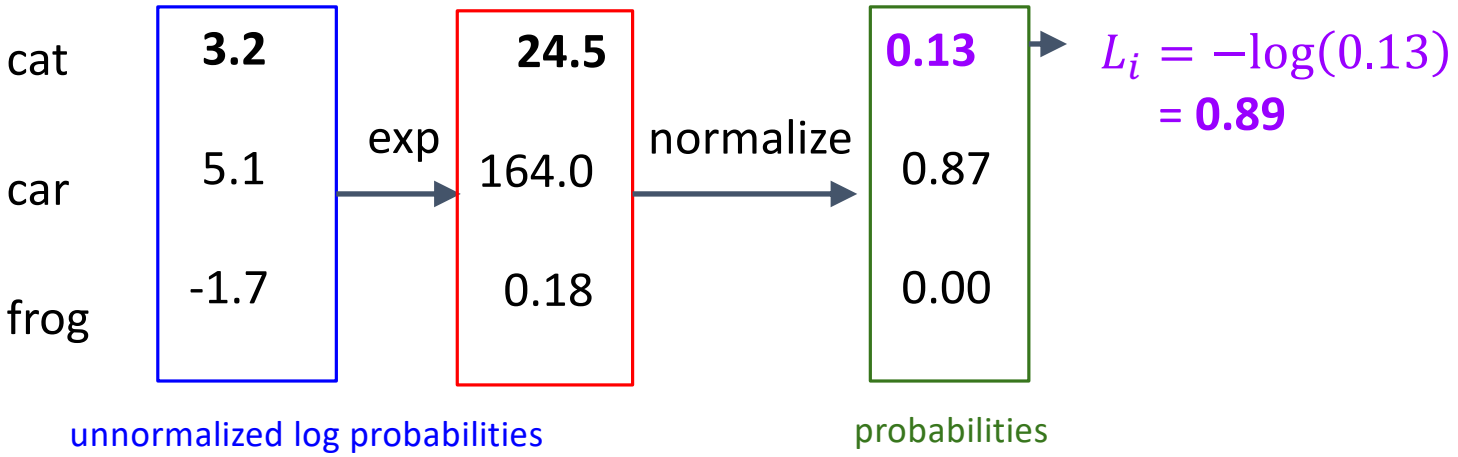
unnormalized probabilities



Softmax Classifier (Multinomial Logistic Regression)

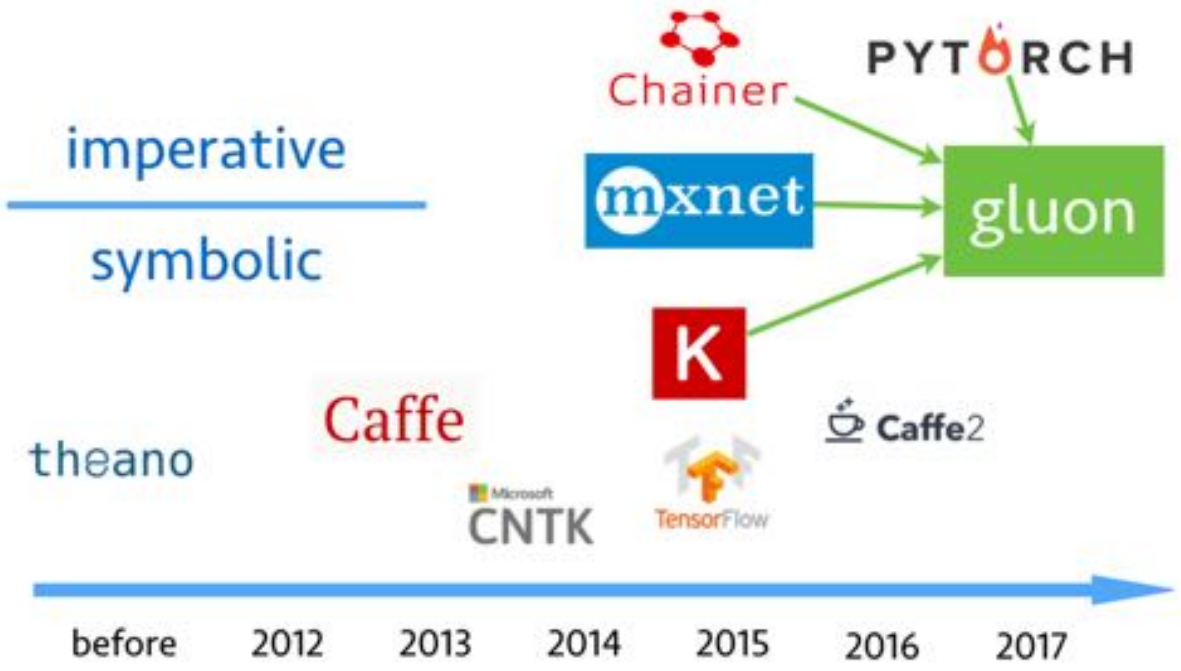
$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

unnormalized probabilities



Break

Evolution of deep learning frameworks



The Apache MXNet ecosystem

Apache MXNet - History

Created by academia
(CMU and UW)

Amazon's deep-learning
framework of choice
since November

Accepted into
Apache Incubator
in January 2017.

The Apache Software foundation (ASF)

- Decentralized
- Non profit
- Open source
- Community driven

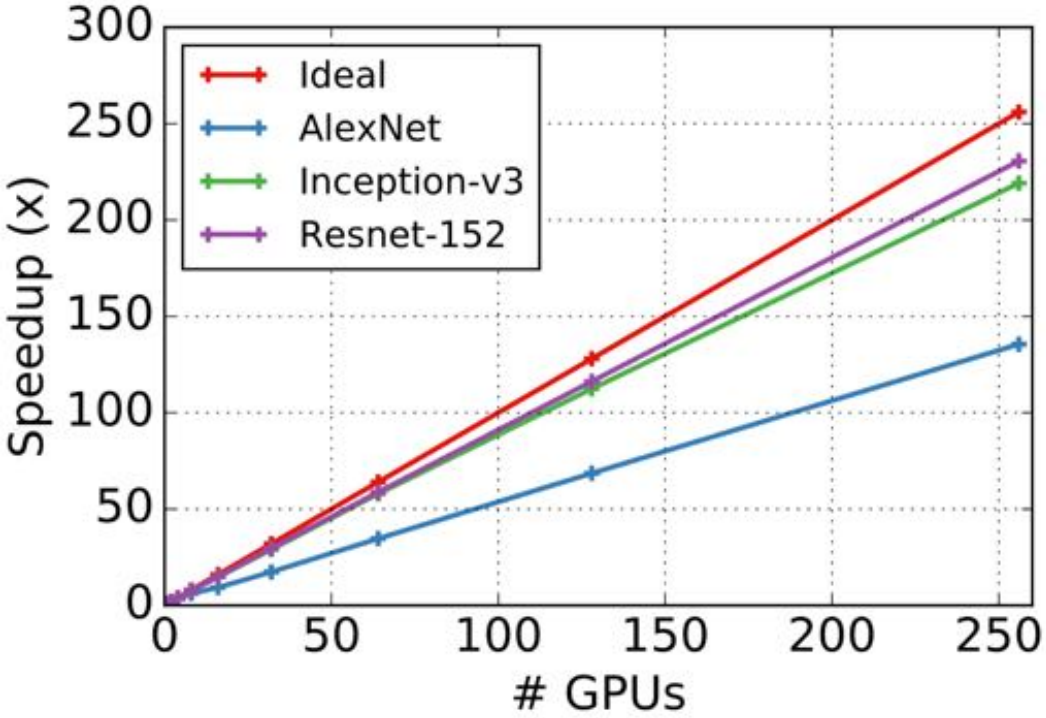
MXNet

- Open source
- Apache incubating
- Community driven (not controlled by a single company or entity)
- Intel, NVidia, Amazon, Microsoft, Wolfram Research, Carnegie Mellon, MIT, The University of Washington, Hong Kong university of Science and Technology.
- Scalable, designed to be distributed using a parameter server, based on CMU, Baidu and Google research. Scales close to linearly with multiple CPUs and GPUs.

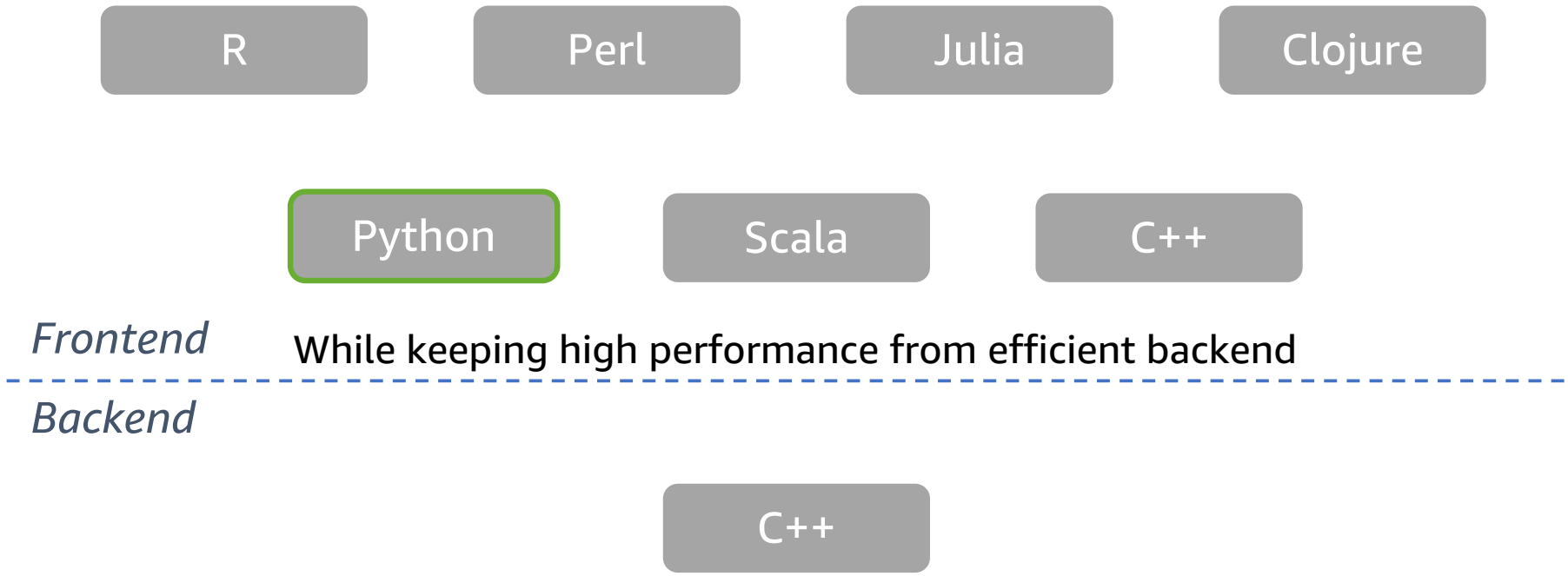
MXNet

- Flexible, supports both imperative and symbolic programming. Imperative makes it easy to use breakpoints and debug, check weights, modify hyperparameters on the fly.
- Multiple language bindings, Core is in C++, bindings for Python (most popular), Scala, R, C++, Perl, Matlab, and JS.
- Portable, ARM support, Raspberry PI etc.

MXNet Scalability



Multi-language Support



Deep Learning acceleration

CUDA & CuDNN

`pip install mxnet-cu92`

TensorRT

`pip install mxnet-tensorrt-cu92`

MKL, MKLML & MKLDNN

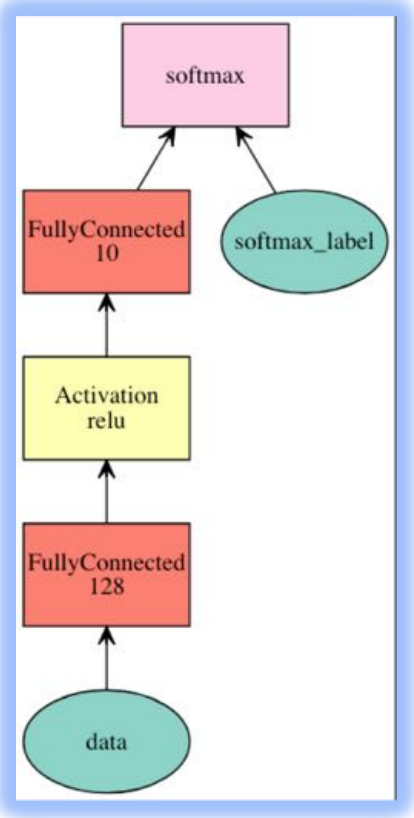
`e.g. pip install mxnet-mkl`

Computational graph

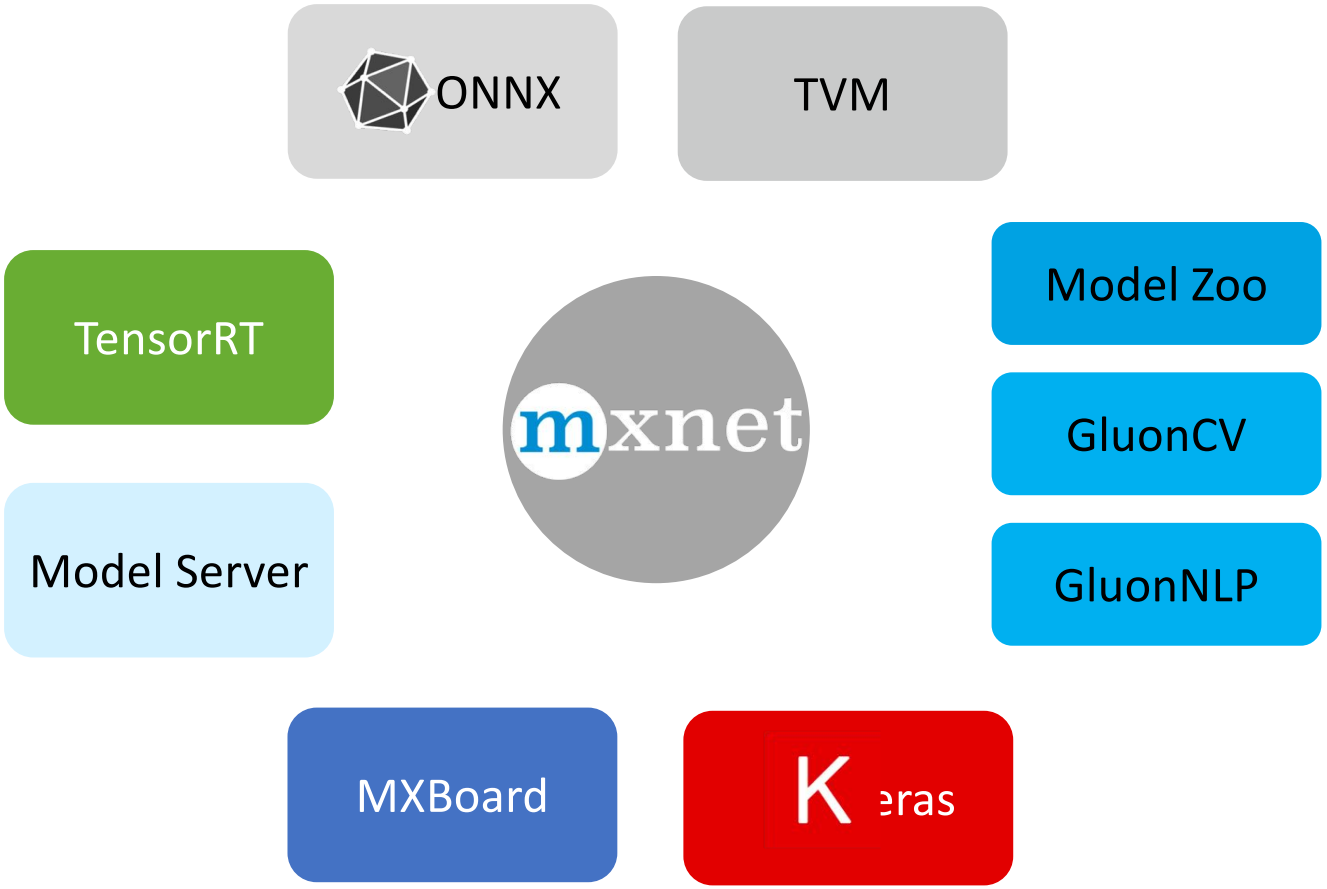
- Neural network operations can be modeled as a special form of computational graphs.
- A computational graph is a directed graph where the nodes correspond to operations or variables. Variables can feed their values into operations, and those can be scheduled in parallel or potentially across different compute devices.
- With a CG we give a functional description of the computation, we can run forward computation and backward.
- A node knows how to compute its value and the value of the gradient

Computational graph (symbolic)

```
net = mx.sym.Variable('data')
net = mx.sym.FullyConnected(net, name='fc1', num_hidden=128)
net = mx.sym.Activation(net, name='relu1', act_type="relu")
net = mx.sym.FullyConnected(net, name='fc2', num_hidden=10)
net = mx.sym.SoftmaxOutput(net, name='softmax')
```



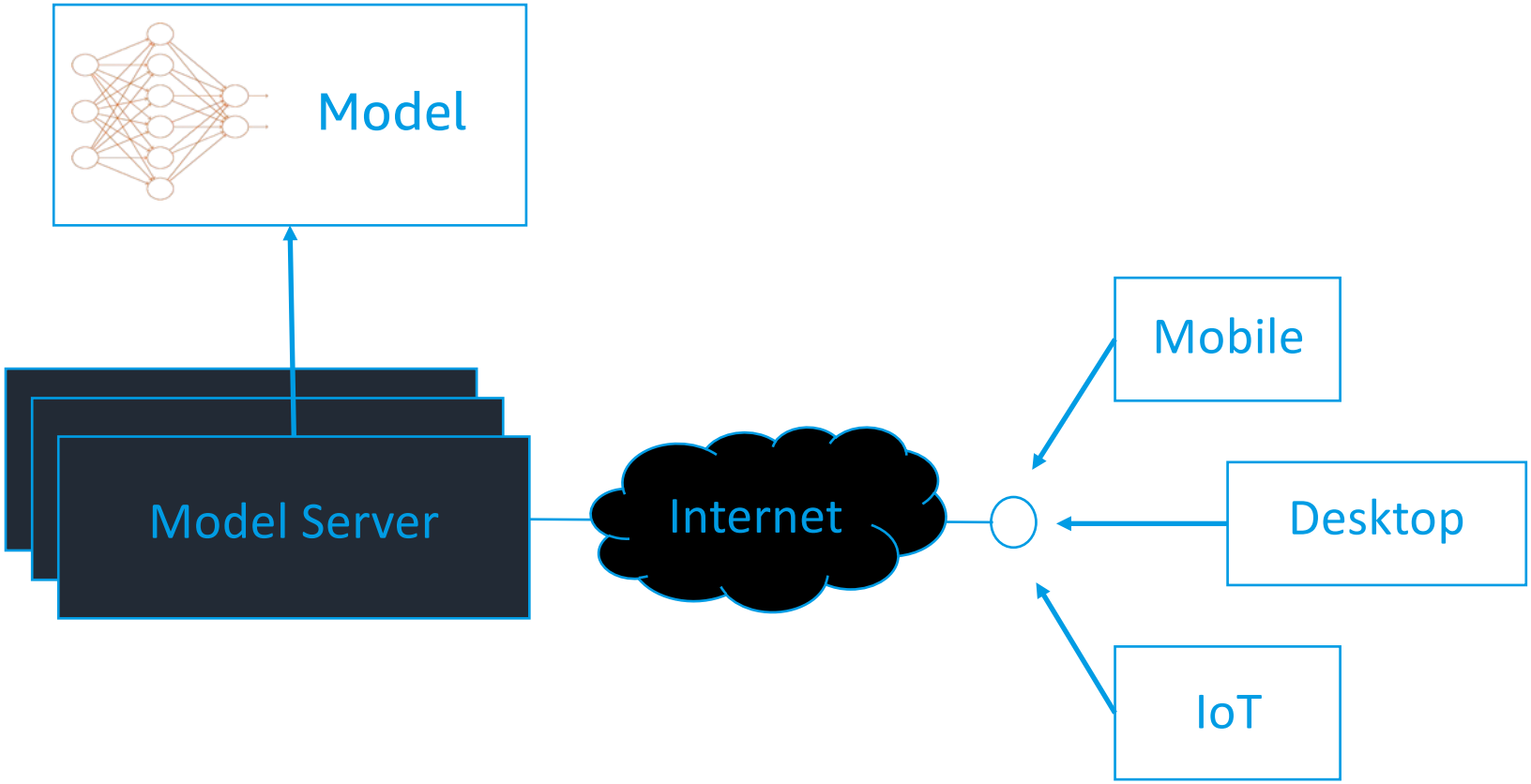
Apache MXNet Ecosystem



Model Server

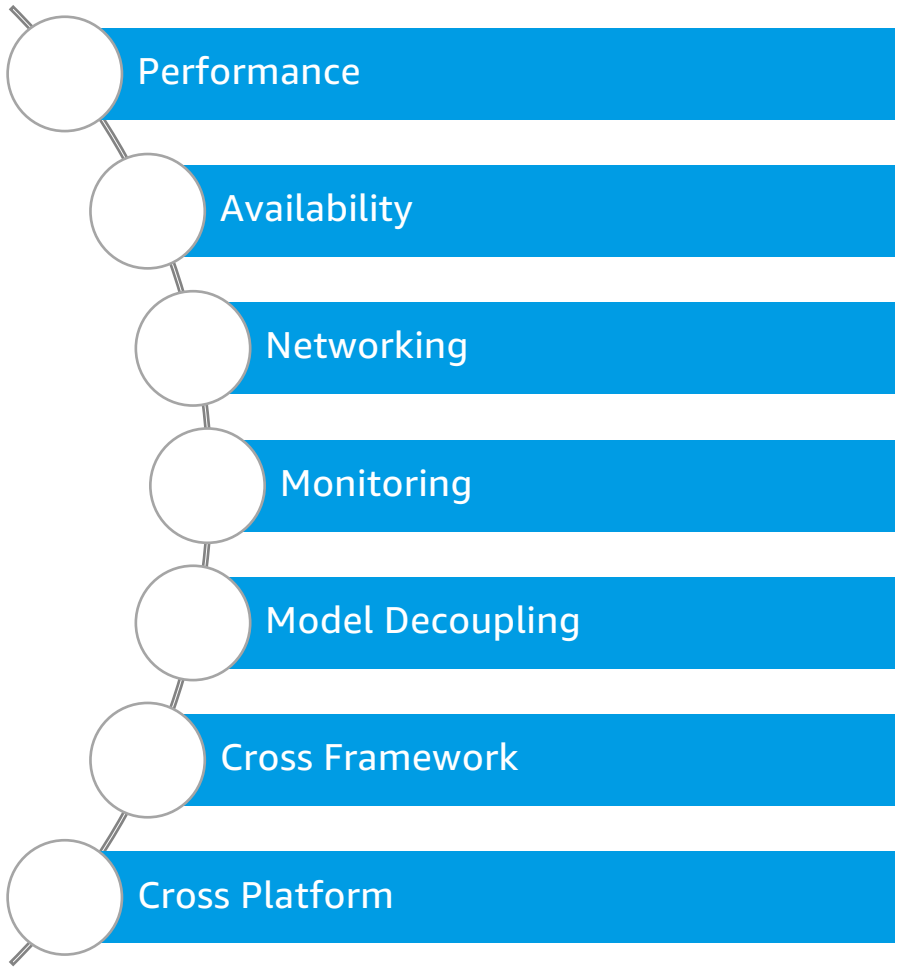
Credits: Hagay Lupesko

So what does a deployed model looks like?



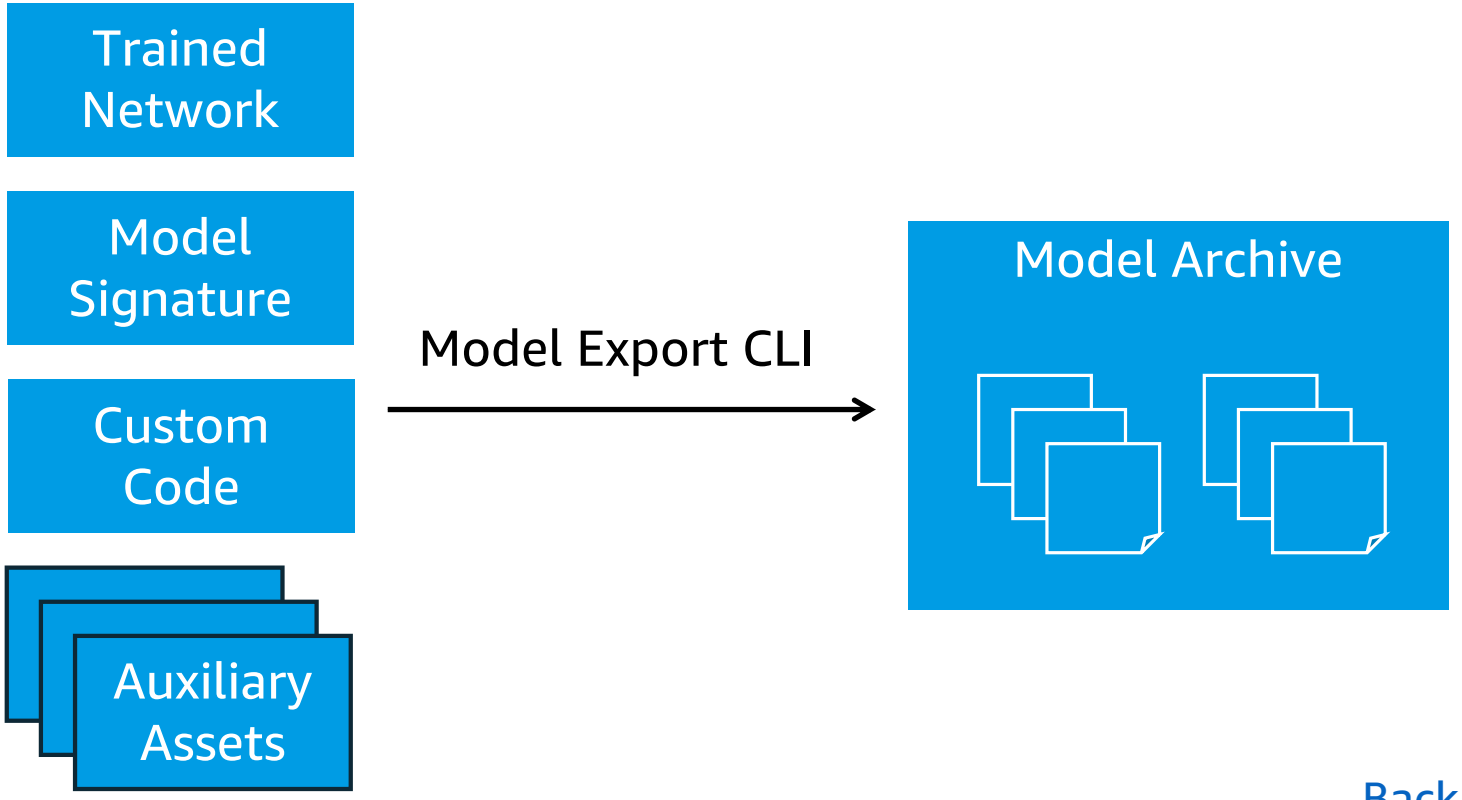
Credits: Hagay Lupesko

The Undifferentiated Heavy Lifting of Model Serving



Credits: Hagay Lupesko

Model Archive

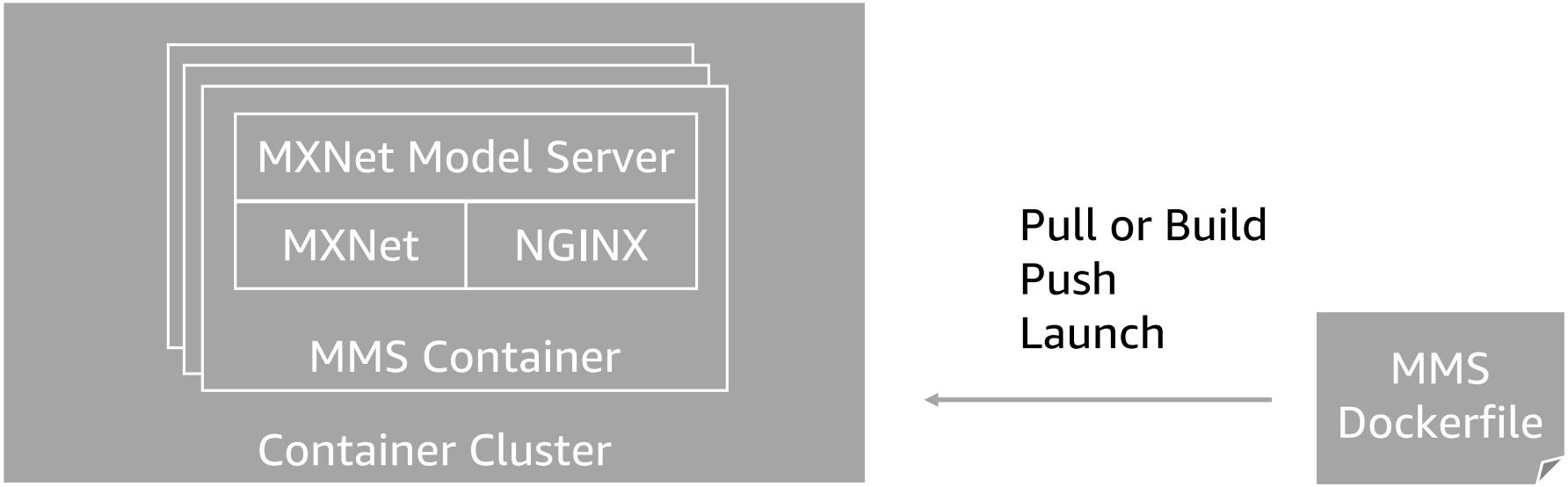


Back

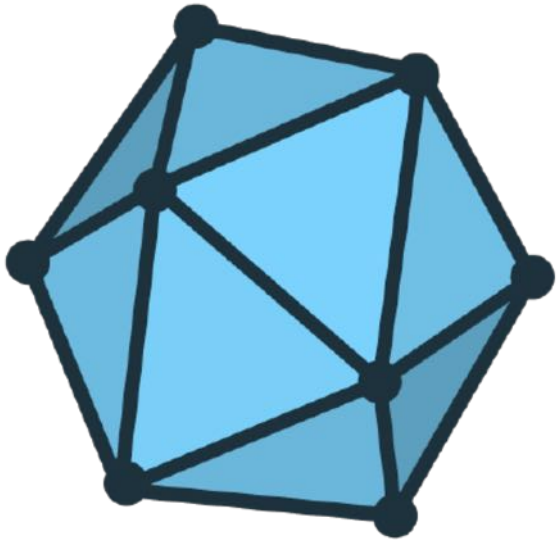
Credits: Hagay Lupesko

Containerization

Lightweight virtualization, isolation, runs anywhere



Back



ONNX

Credits: Hagay Lupesko

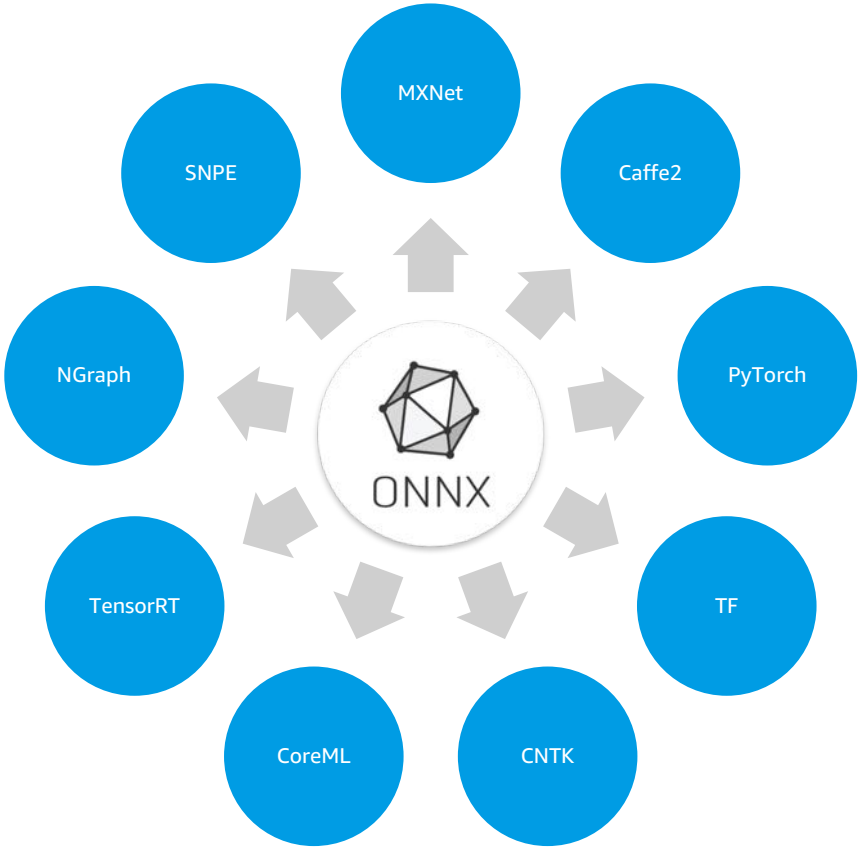
Open Neural Network eXchange - Overview

Many Frameworks

Many Platforms

ONNX: Common IR

- Open source
- Community driven
- Simple



Credits: Hagay Lupesko

Import ONNX model in MXNet– Usage Example

Build and train your model with PyTorch

Load your ONNX model with MXNet

Run inference, fine tune or save as MXNet model.

```
# Import into MXNet (from MXNet 1.2)
sym, arg_params, aux_params = onnx_mxnet.import_model('model.onnx’)

# create module
mod = mx.mod.Module(symbol=sym, data_names=['input_0'], label_names=None)
mod.bind(for_training=False, data_shapes=[('input_0', input_img.shape)])
mod.set_params(arg_params=arg_params, aux_params=aux_params)
```

Credits: Hagay Lupesko

Export MXNet model to ONNX – Usage

Build and train your model in MXNet

Export trained MXNet model to ONNX format

Import in other framework like cntk, caffe2 for inference

```
# Export MXNet model to ONNX format(from MXNet 1.3)
onnx_file_path = onnx_mxnet.export_model(sym, params, [input_shape],
input_data_type, onnx_file_path)
```

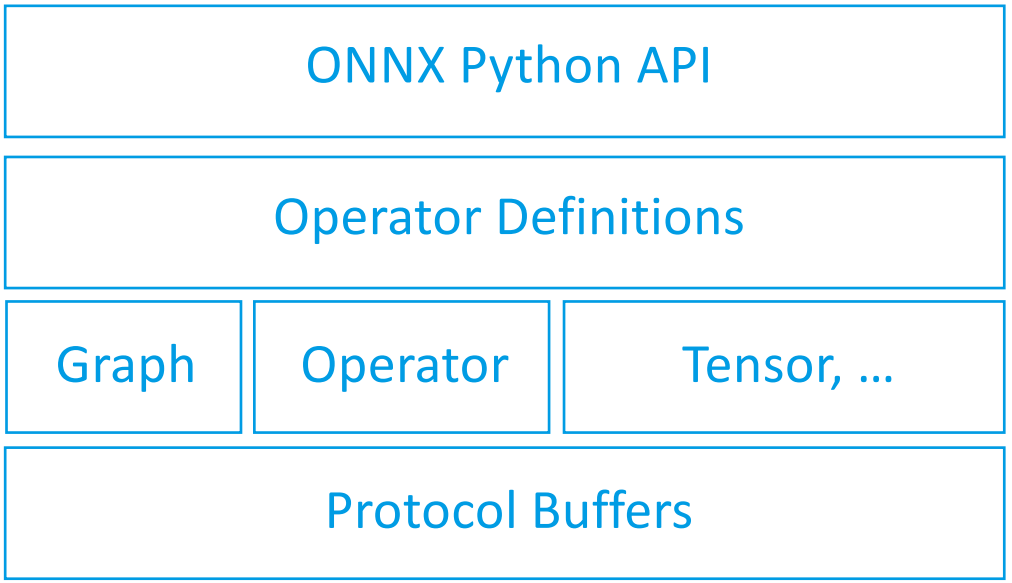
Back

Credits: Hagay Lupesko

ONNX – Internals

Protocol Buffers:

- Binary compact format
- Statically defined
- APIs for de/serialization
- Cross platform



Credits: Hagay Lupesko

ONNX – Coverage

Framework	Export	Import
MXNet	Supported	Supported
Caffe2	Supported	Supported
PyTorch	Supported	Coming…
CNTK	Supported	Supported
Chainer	Supported (external)	N/A
TensorFlow	Supported (external)	Supported (external)
CoreML	Supported (external)	Supported (external)
SciKit-Learn	Supported (external)	N/A

Credits: Hagay Lupesko

Keras-MXNet

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

Keras – Apache MXNet

- Deep Learning for Humans
- 2nd most popular Deep Learning framework
- Keras users leverage MXNet's great performance

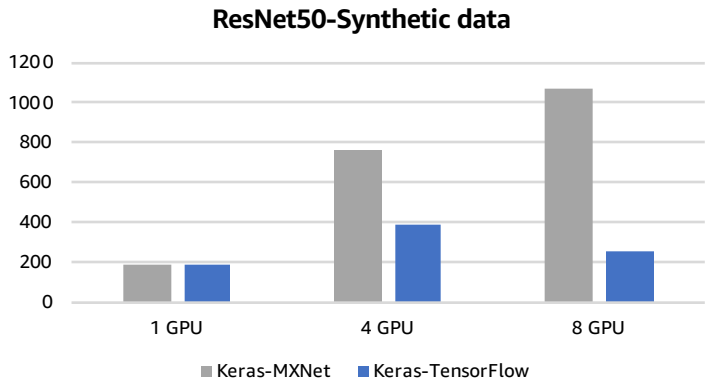
```
from keras.models import Sequential
model = Sequential()
from keras.layers import Dense
model.add(Dense(units=64, activation='relu', input_dim=100))
model.add(Dense(units=10, activation='softmax'))
model.compile(loss='categorical_crossentropy',
              optimizer='sgd',
              metrics=['accuracy'])
model.fit(x_train, y_train, epochs=5, batch_size=32)
model.train_on_batch(x_batch, y_batch)
loss_and_metrics = model.evaluate(x_test, y_test,
batch_size=128)
classes = model.predict(x_test, batch_size=128)
```

```
pip install mxnet-(mkl|cu92)
pip install keras-mxnet
---
~/.keras/keras.json
backend: mxnet
image_data_format: channels_first
---
```

Keras Benchmarks

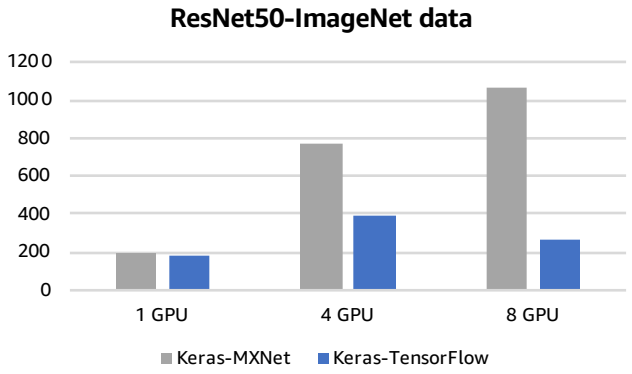
Setup: https://github.com/awslabs/keras-apache-mxnet/tree/master/benchmark

	Training	Inference
Instance	P3.8x Large, P3.16x Large	C5.xLarge, C4.8xLarge
Network	ResNet50v1	ResNet50v1
Batch size	32 * Num of GPUs	32
Image size	3*256*256	3*256*256

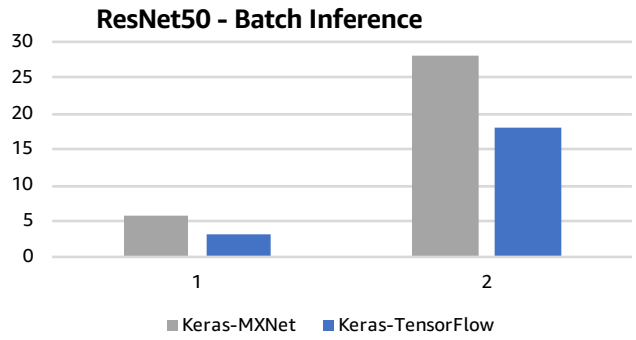


GPUs	Keras-MXNet [Image/sec]	Keras-TensorFlow [Image/sec]	Speed Up
1	194	184	1.05
4	764	393	1.94
8	1068	261	4.09

Keras Benchmarks



GPUs	Keras-MXNet	Keras-TensorFlow	Speed Up
1	135	52	2.59
4	536	162	3.30
8	722	211	3.42



Instance	Keras-MXNet	Keras-TensorFlow	Speed Up
C5.X Large	5.79	3.27	1.782
C5.8X Large	27.9	18.2	1.53

Imperative API

GLUON

Debuggable

Flexible

Scalable

Why Gluon

Simple, Easy-to-Understand Code

Flexible, Imperative Structure

Dynamic Graphs

High Performance

Gluon example, network definition

```
net = gluon.nn.HybridSequential()
with net.name_scope():
    net.add(gluon.nn.Dense(units=64, activation='relu'))
    net.add(gluon.nn.Dense(units=10))


softmax_cross_entropy =
gluon.loss.SoftmaxCrossEntropyLoss()


net.initialize(mx.init.Xavier(magnitude=2.24), ctx=ctx,
force_reinit=True)


trainer = gluon.Trainer(net.collect_params(), 'sgd',
{'learning_rate': 0.02})
```

Gluon example, training

```
smoothing_constant = .01
for e in range(10):
    cumulative_loss = 0
    for i, (data, label) in enumerate(train_data):
        data = data.as_in_context(model_ctx).reshape((-1, 784))
        label = label.as_in_context(model_ctx)
        with autograd.record():
            output = net(data)
            loss = softmax_cross_entropy(output, label)
        loss.backward()
        trainer.step(data.shape[0])
test_accuracy = evaluate_accuracy(test_data, net)
train_accuracy = evaluate_accuracy(train_data, net)
```

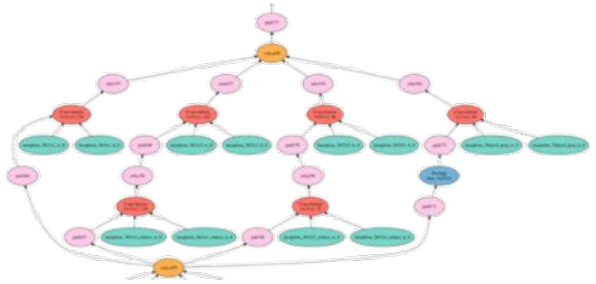
Symbolic vs Imperative

Symbolic is “define, compile, run”

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])

model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
```

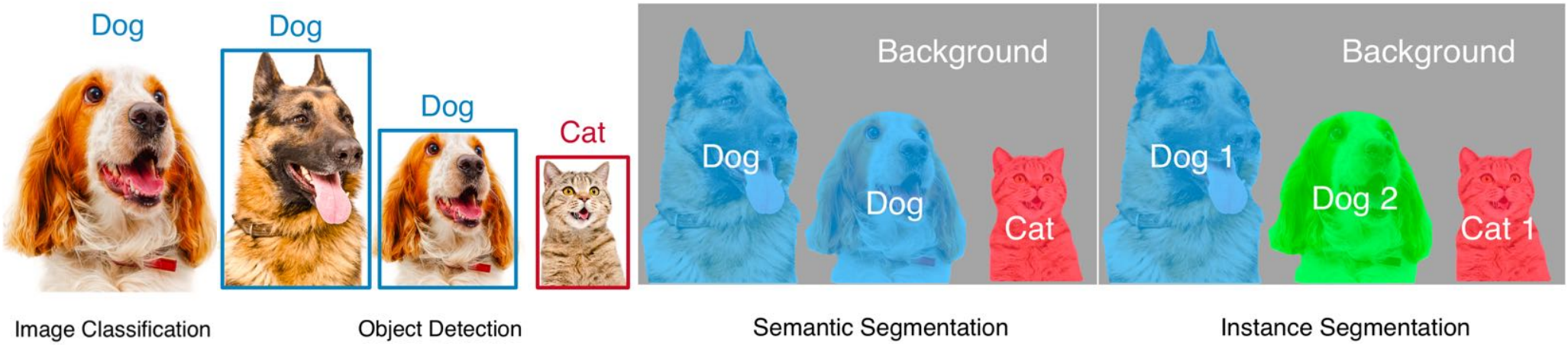


```
net = nn.Sequential()
with net.name_scope():
    net.add(
        nn.Conv2D(channels=6, kernel_size=5, activation='relu'),
        nn.MaxPool2D(pool_size=2, strides=2),
        nn.Conv2D(channels=16, kernel_size=3, activation='relu'),
        nn.MaxPool2D(pool_size=2, strides=2),
        nn.Flatten(),
        nn.Dense(120, activation="relu"),
        nn.Dense(84, activation="relu"),
        nn.Dense(10)
    )
net.initialize(init=init.Xavier())
```

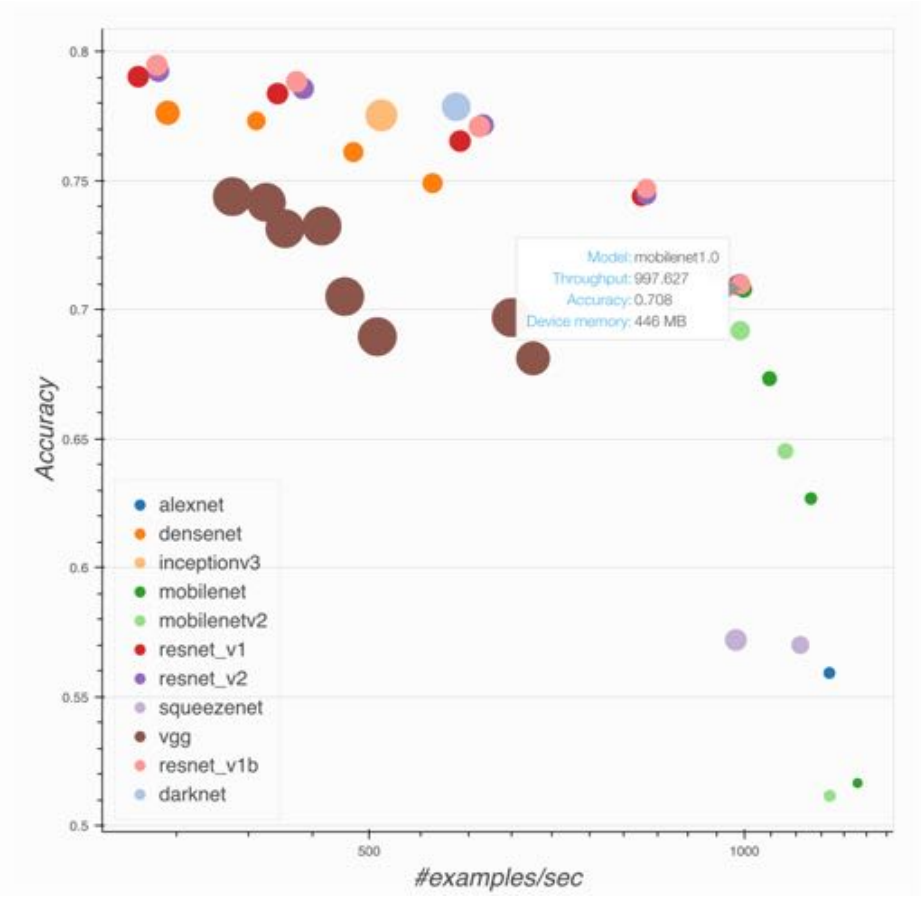
```
for epoch in range(10):
    for data, label in train_data:
        with autograd.record():
            output = net(data)
            loss = softmax_cross_entropy(output, label)
        loss.backward()
        trainer.step(batch_size)
```

GluonCV: a Deep Learning Toolkit for Computer Vision

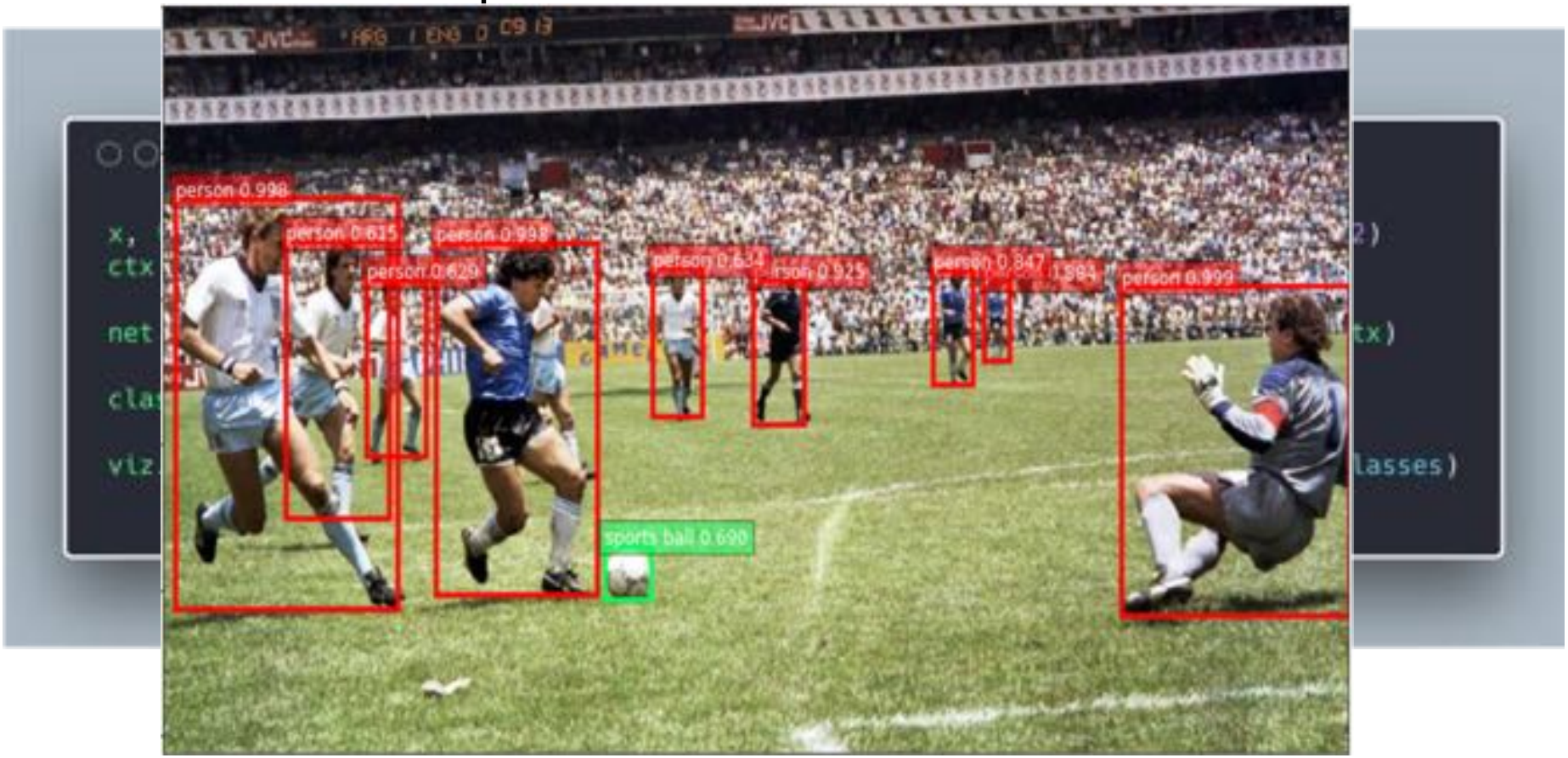
https://gluon-cv.mxnet.io



GluonCV: pre-trained models, help to choose



GluonCV: example code



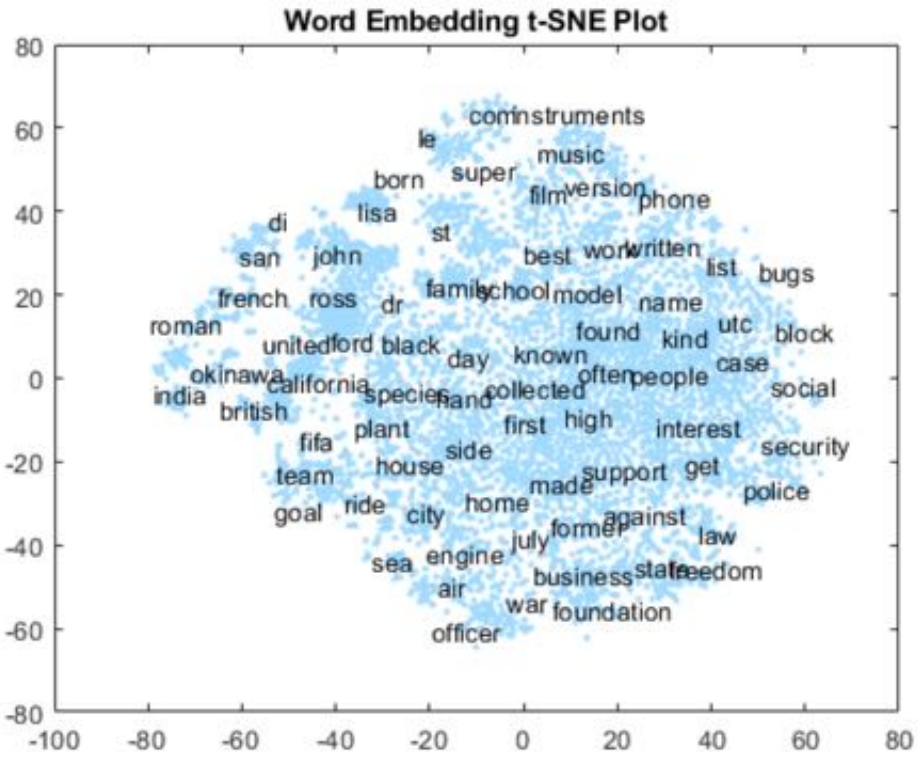
GluonNLP: a Deep Learning Toolkit for Natural Language Processing
https://gluon-nlp.mxnet.io

- Features (as of 0.3.2)

- Pre-trained models: over 300 word-embedding
- 5 language models
- Neural Machine Translation (Google NMT, Transformer)
- Flexible data pipeline tools and many public datasets.
- NLP examples such as sentiment analysis.

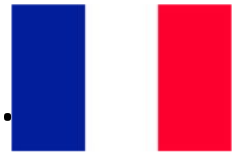
	car
dog	0
apple	0
eat	0
car	1
van	0
leaf	0
the	0
at	0

car	van
0.1	0.2
0.5	0.3
-1.1	-1.0
0.4	0.5



Language modeling

Trained to predict the next word ($P(w_n|w_i \dots w_{n-1})$):

- The winner of the 2018 FIFA world cup is ……. 

```
lm_model, vocab = nlp.model.get_model(name='standard_lstm_lm_200', dataset_name='wikitext-2',
pretrained=True)
data = vocab[tokenizer("This movie is considered the")]

pred, _ = lm_model(mx.nd.array(data, dtype='float32').expand_dims(axis=1))

for p in pred[-1].squeeze().topk(k=20).asnumpy():
    print(vocab.idx_to_token[int(p)])
```

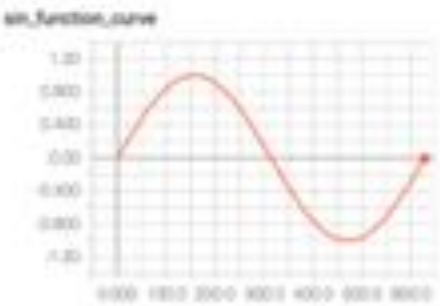
This movie is considered the <?>

first	greatest	highest
most	main	final
<unk>	second	worst
only	last	sixth
same	name	third
best	largest	way
		primary

MXBoard: MXNet plugin to TensorBoard

Graph

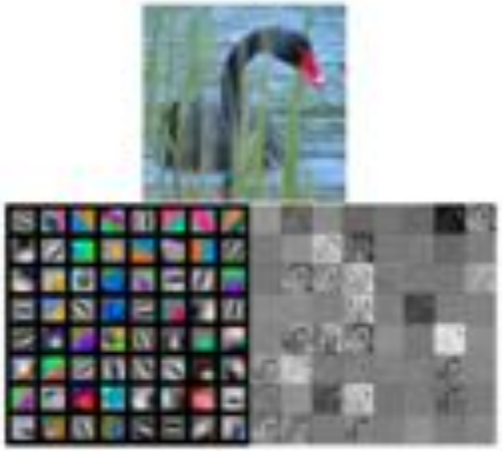
Scalar



Histogram

Embedding

Image



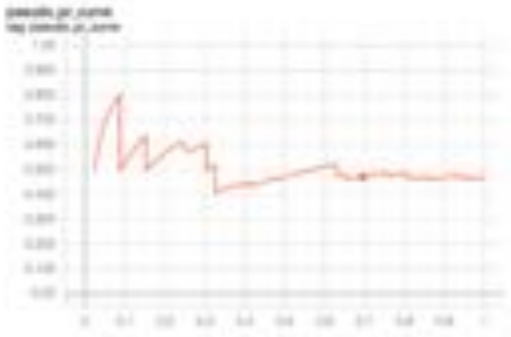
Text

markdown_table
tag: markdown_table

step 0

Hello	**MXNet,**
This	is
so	awesome!

Precision-Recall Curve



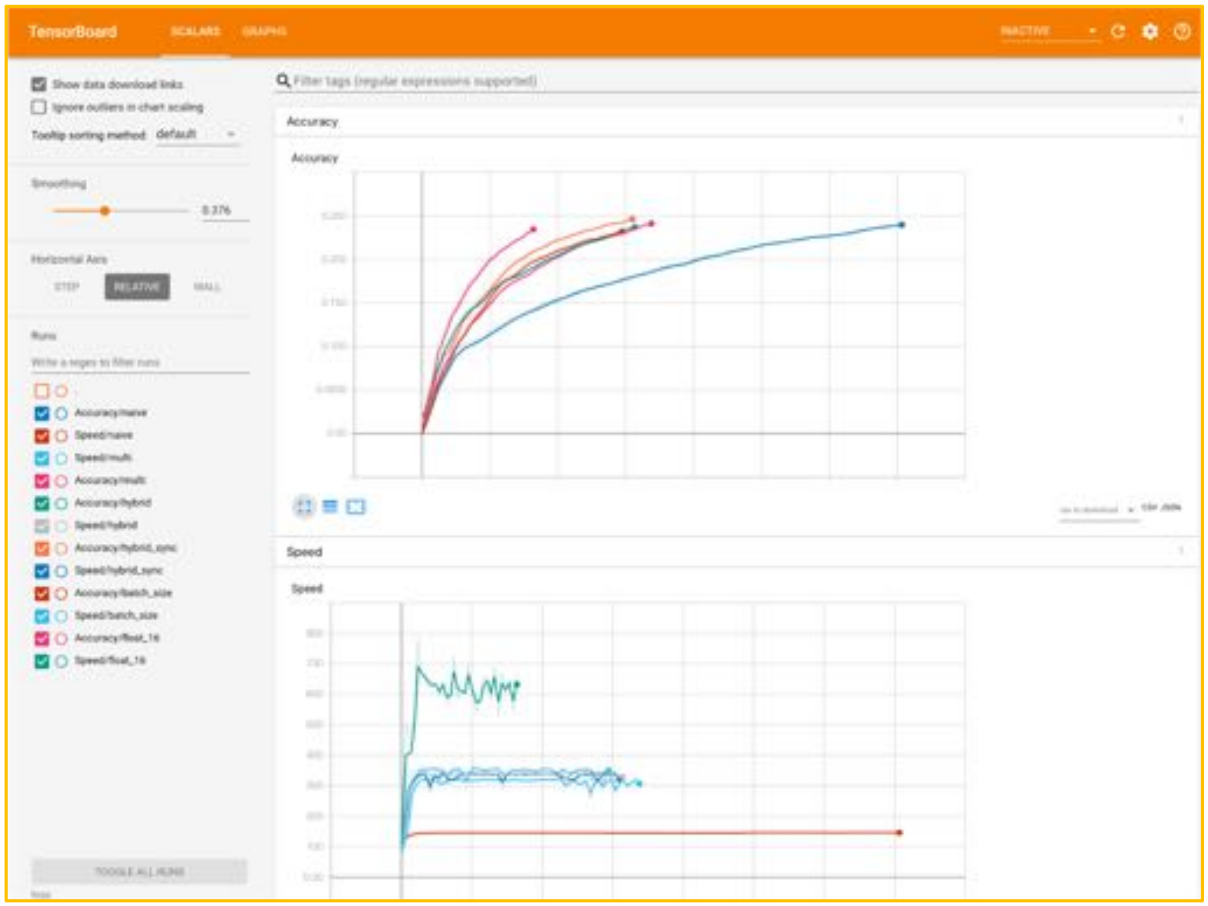
Audio

uniform_audio

uniform_audio
step 0

0:00 / 0:01

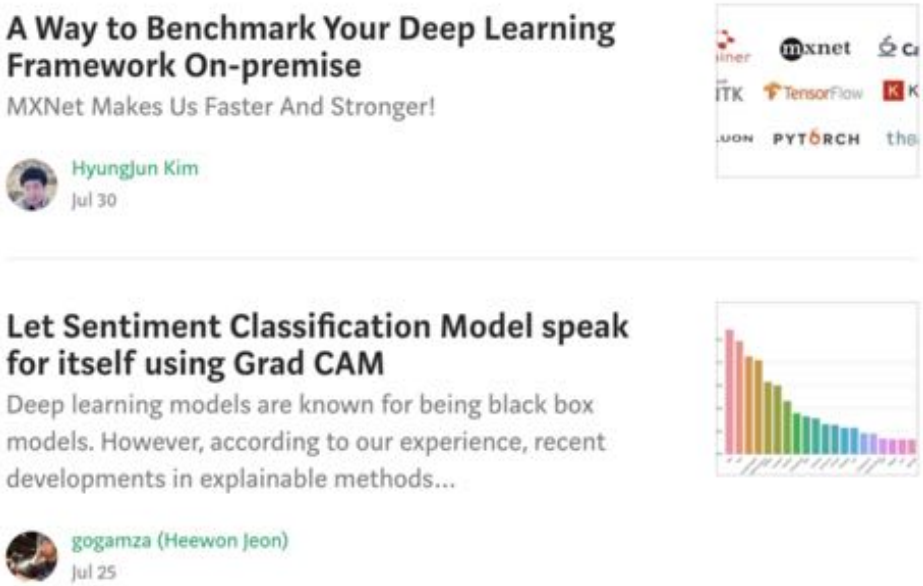
MXBoard


TensorBoard
SCALARS
GRAPHS
INACTIVE
Show data download links
Ignore outliers in chart scaling
Tooltip sorting method: default
Smoothing
0.376
Horizontal Axis
STEP
RELATIVE
WALL
Runs
Write a regex to filter runs
Filter tags (regular expressions supported)
Accuracy
Accuracy
Speed
Speed

Apache MXNet community

Keeping Up to Date

- Medium: https://medium.com/apache-mxnet

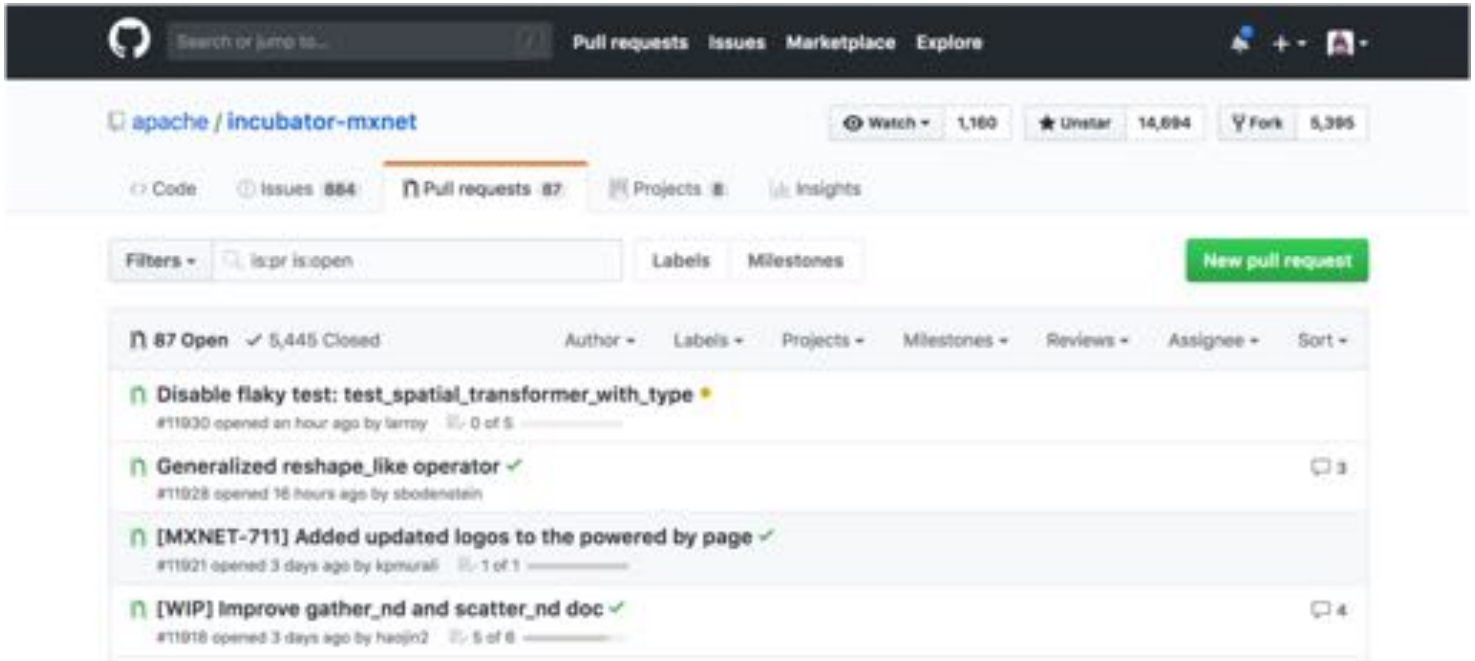


Keeping Up to Date: Social

- YouTube: /apachemxnet

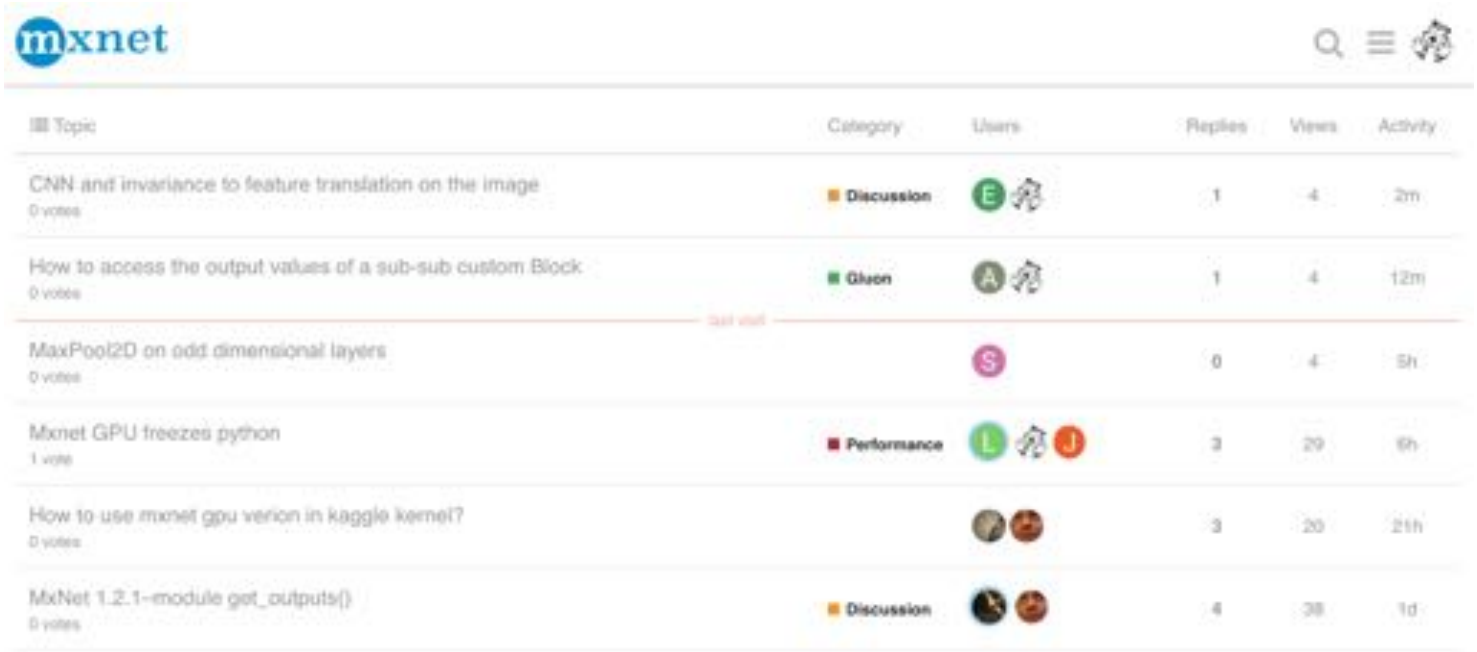
Community

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



Community

- Discuss Forum: https://discuss.mxnet.io/



Community

- Mailing list:

- dev@mxnet.apache.org
- user@mxnet.apache.org

MXNet Customer Momentum

Thank you!

http://mxnet.io

http://discuss.mxnet.io

/apachemxnet

@ApacheMXNet

@ApacheMXNet

/r/mxnet