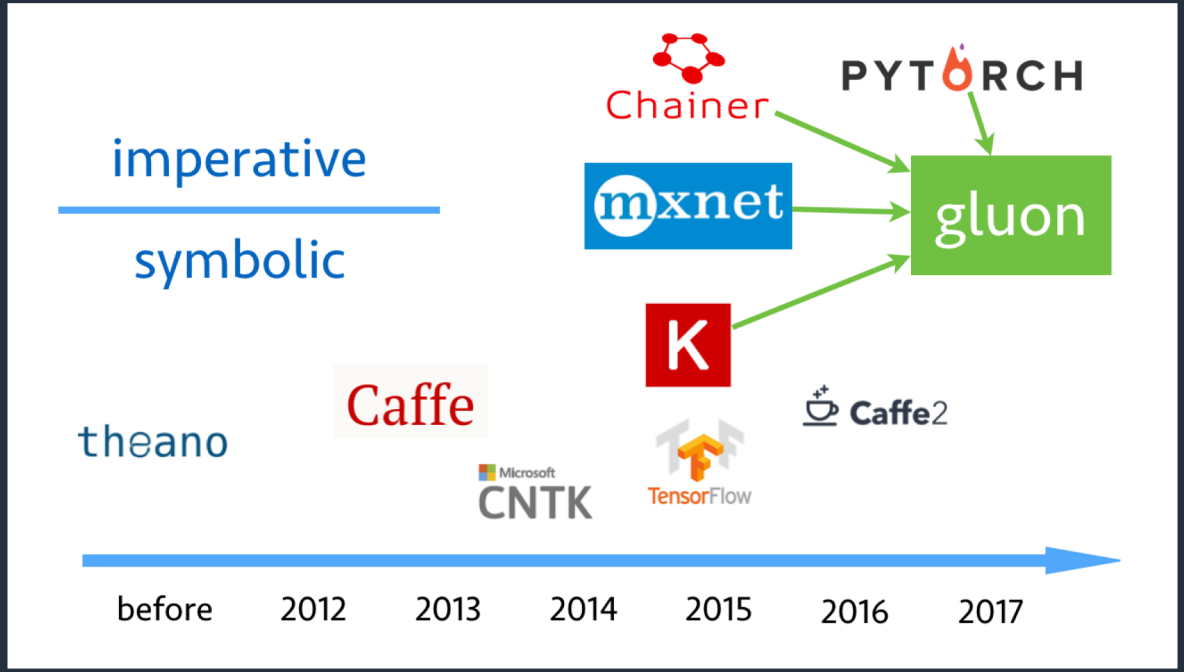
MXNet Ecosystem

Cyrus Vahid – cyrusmv@amazon.com

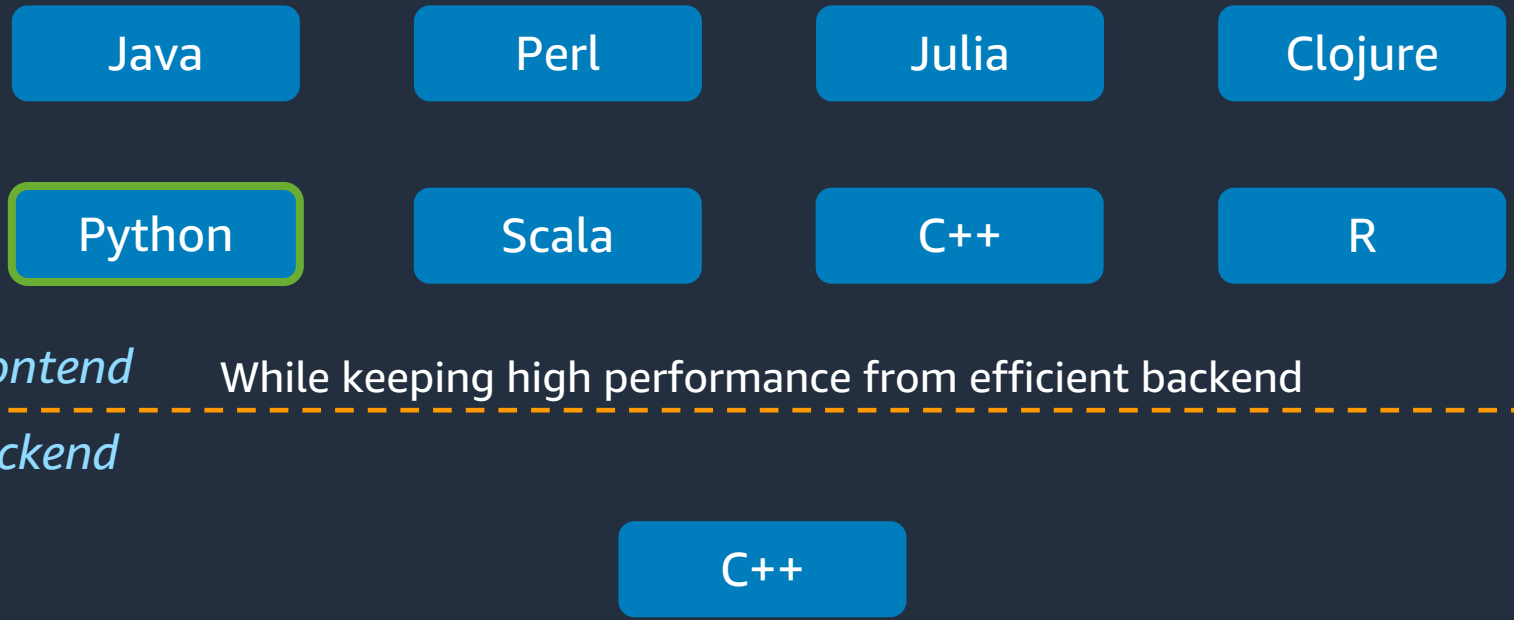
Principal Solutions Engineer– AWS Deep Engine

*

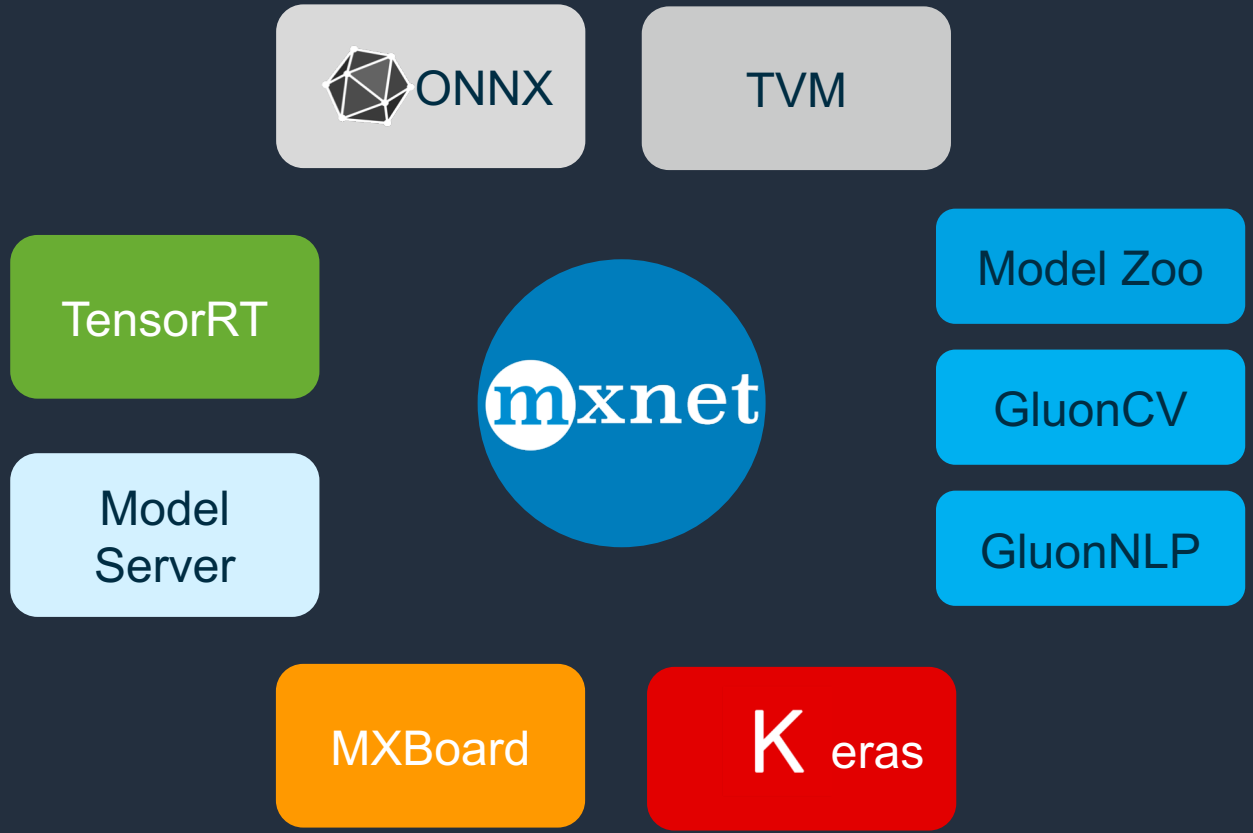
Apache MXNet - History



Multi-language Support

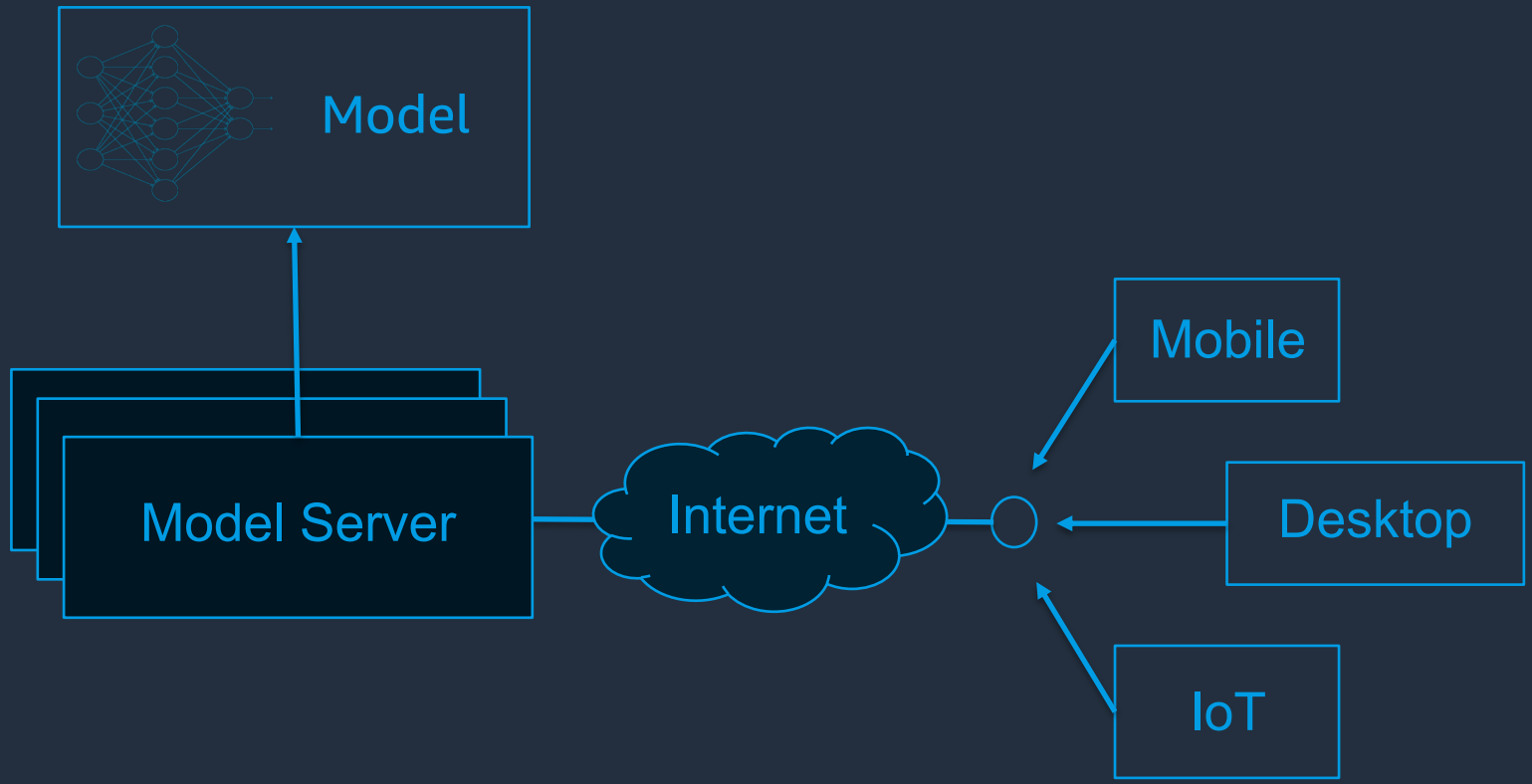


Apache MXNet Ecosystem

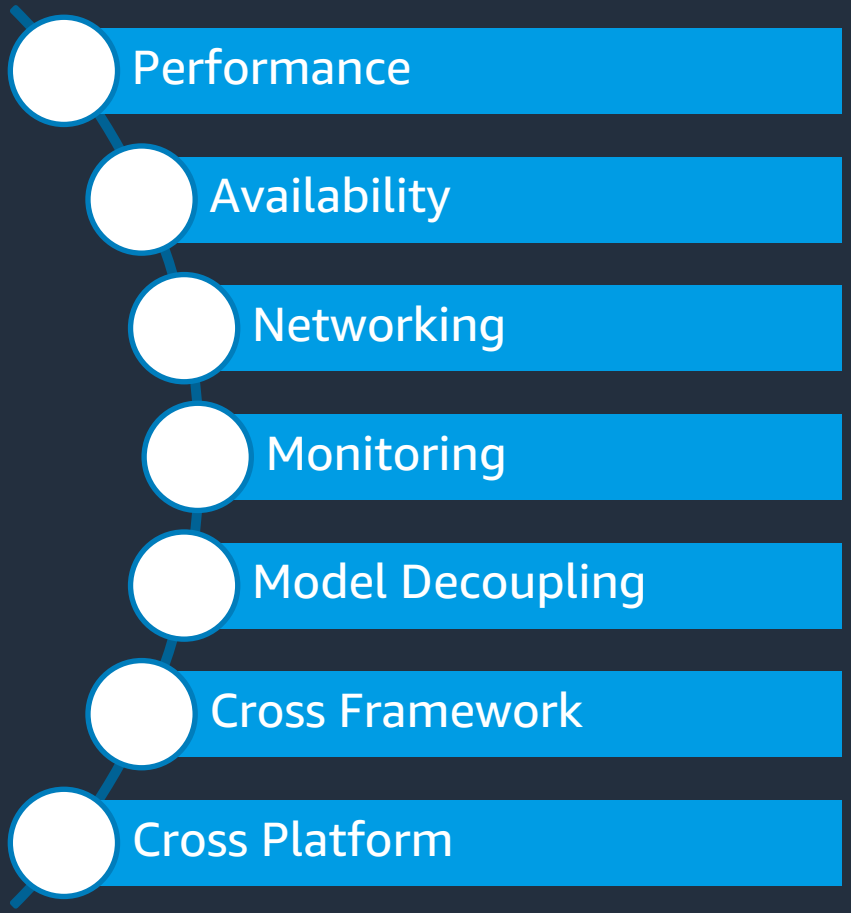


Model Server

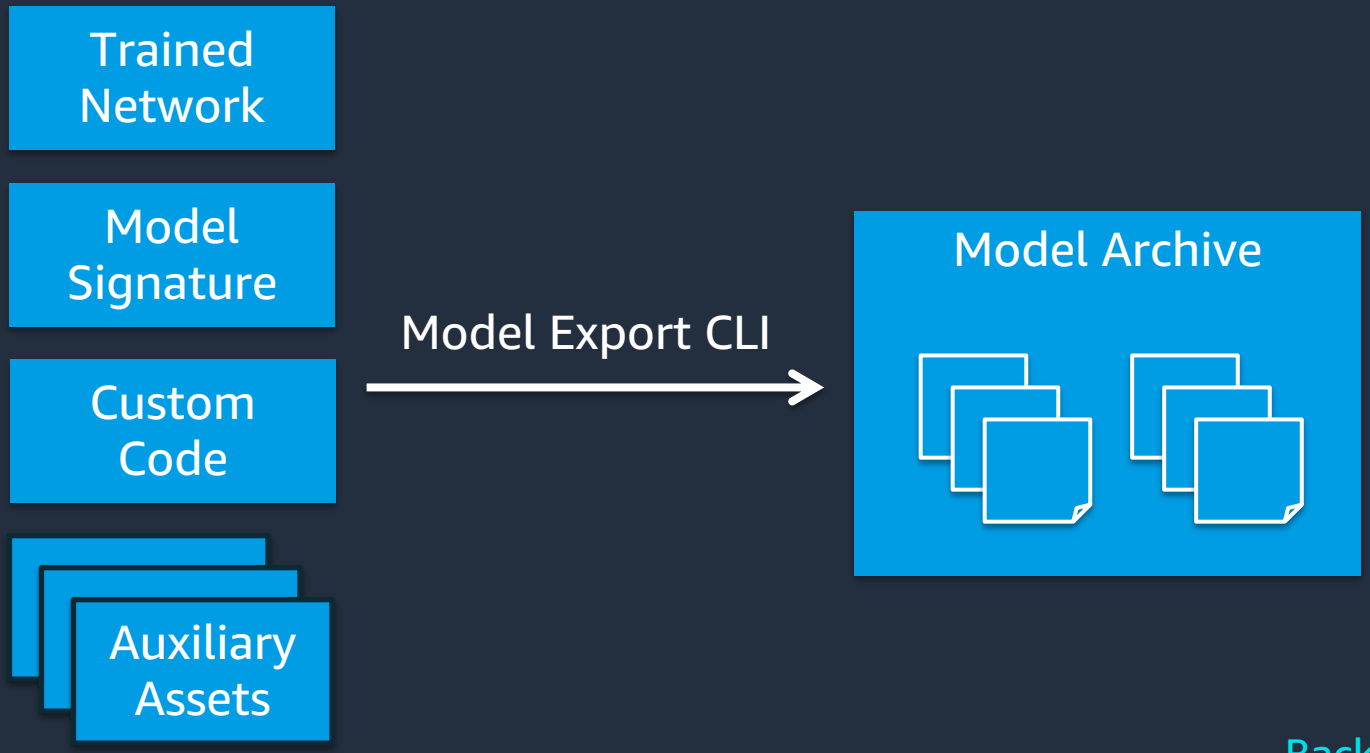
So what does a deployed model looks like?



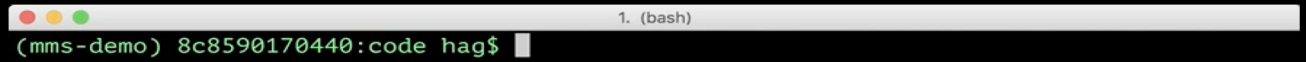
The Undifferentiated Heavy Lifting of Model Serving



Model Archive

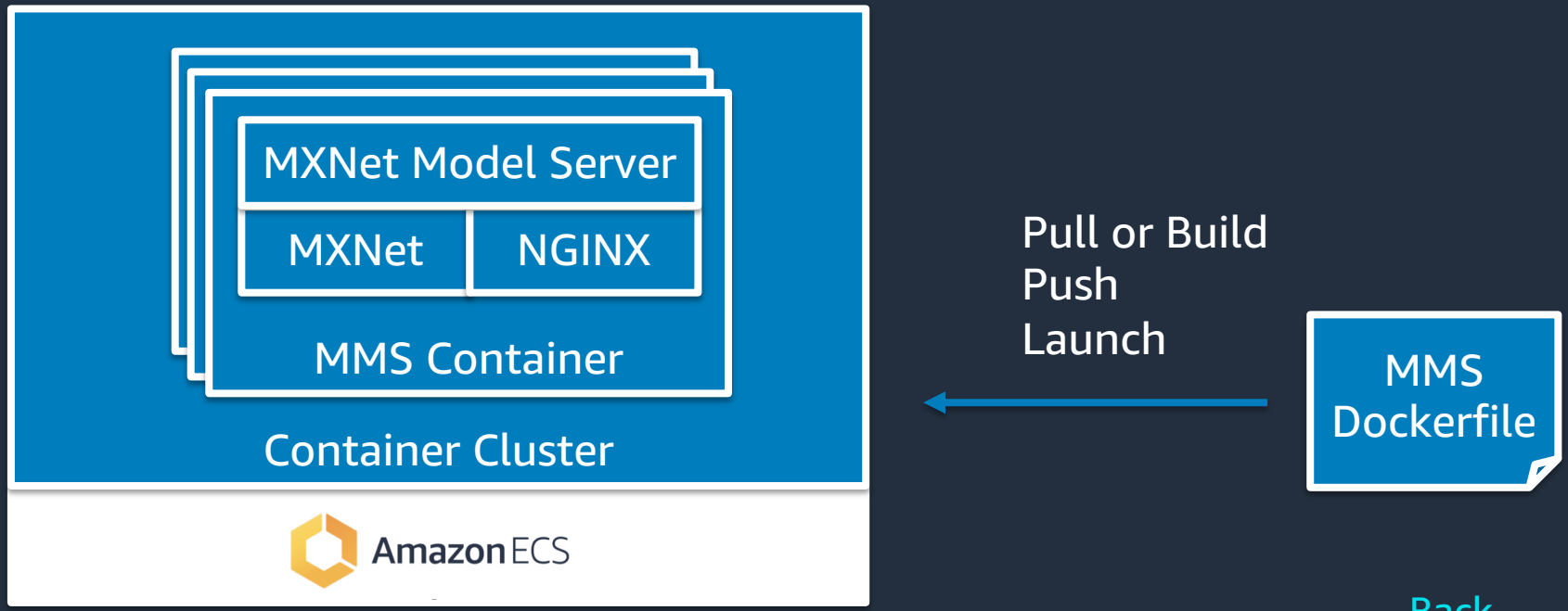


Back


1. (bash)
(mms-demo) 8c8590170440:code hag$

Containerization

Lightweight virtualization, isolation, runs anywhere



Back

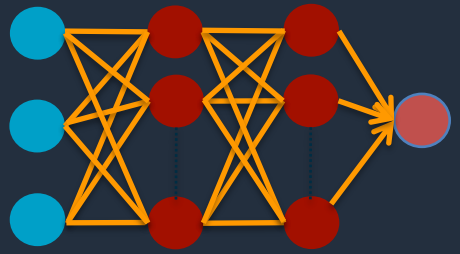
Demo: https://bit.ly/2vgj4A7

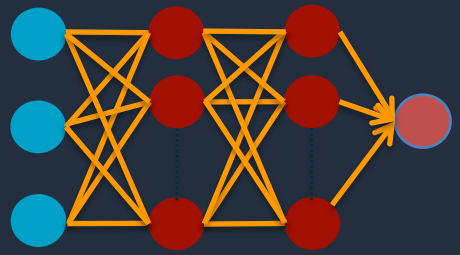
ONNX

Portability - ONNX

Model Parameters

Hyper Parameters





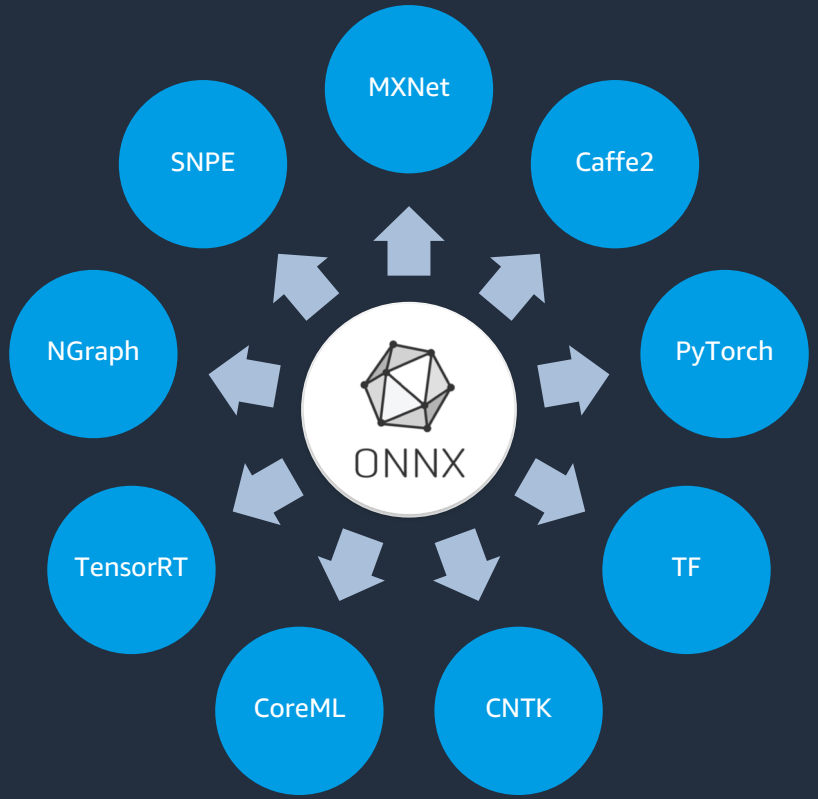
mxnet

Open Neural Network eXchange - Overview

Many Frameworks
Many Platforms

ONNX: Common IR

- Open source
- Community driven
- Simple



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)
```

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

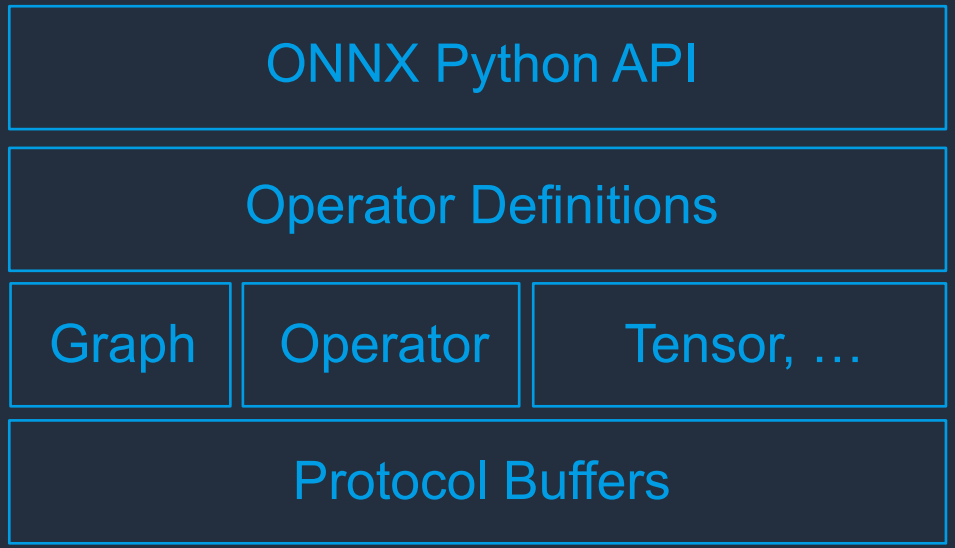
```
# Export MXNet model to ONNX format(from MXNet 1.3)
onnx_file_path = onnx_mxnet.export_model(sym, params, [input_shape],
input_data_type, onnx_file_path)
```

Back

ONNX – Internals

Protocol Buffers:

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

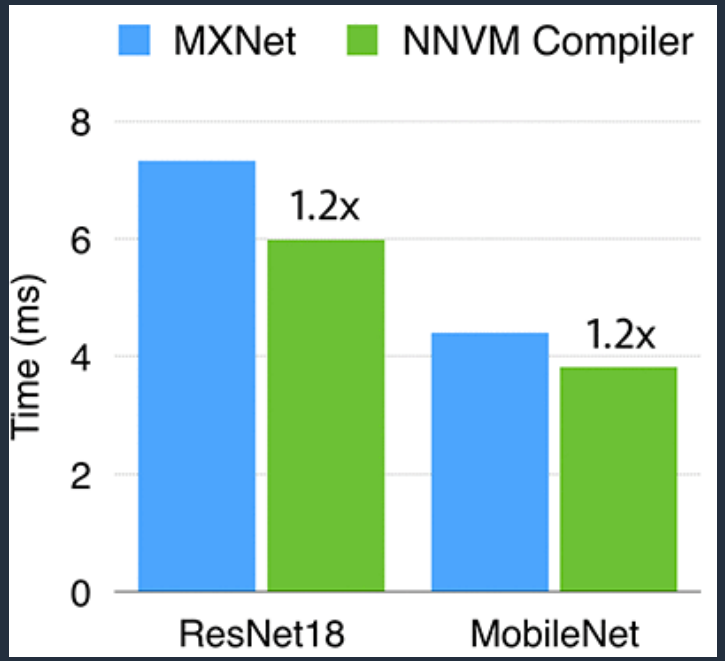


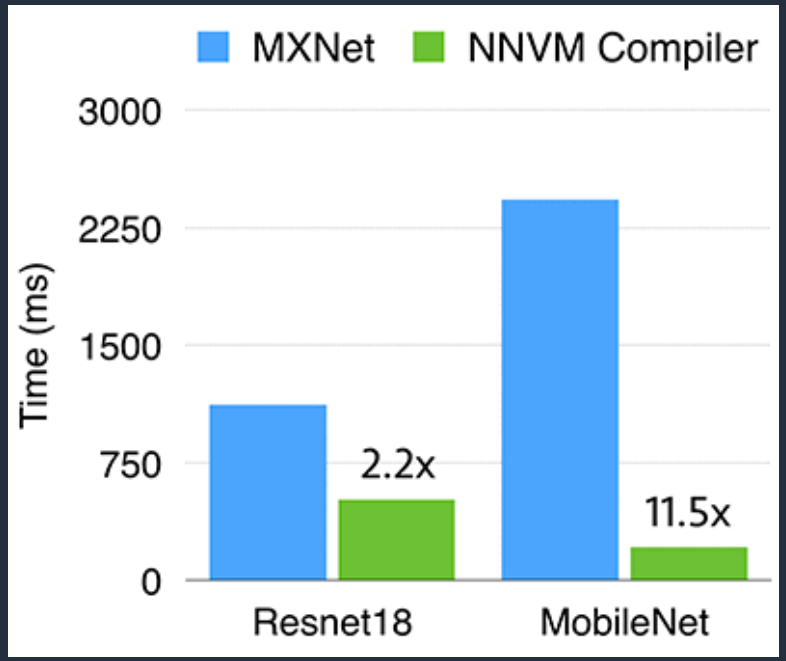
ONNX – Coverage

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

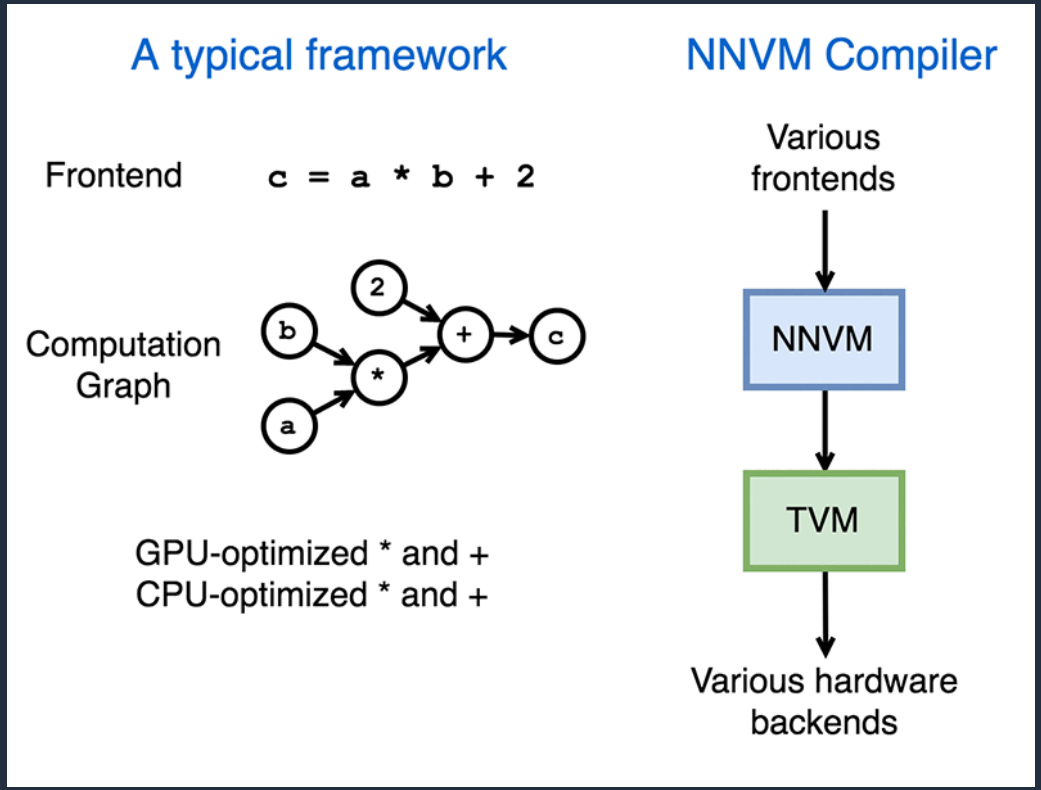
TVM

Inference Efficiency - TVM





Portability - TVM



TensorRT

Inference Efficiency - TensorRT

Model Name	Relative TRT Sppedup	HArdware
Resnet 101	1.99x	Titan V
Resnet 50	1.76x	Titan V
Resnet 18	1.54x	Jetson TX1
cifar_resnext29_16x64d	1.26x	Titan V
cifar_resnet20_v2	1.21x	Titan V
Resnet 18	1.8x	Titan V
Alexnet	1.4x	Titan V

https://cwiki.apache.org/confluence/display/MXNET/How+to+use+MXNet-TensorRT+integration

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

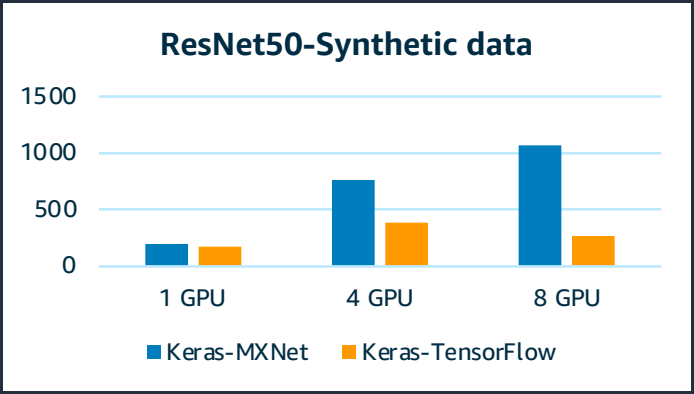
```
from keras.models import Sequential
model = Sequential()
from keras.layers import Dense
model.add(Dense(units=64, activation='relu', input_dim=100))
model.add(Dense(units=10, activation='softmax'))
model.compile(loss='categorical_crossentropy',
              optimizer='sgd',
              metrics=['accuracy'])
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

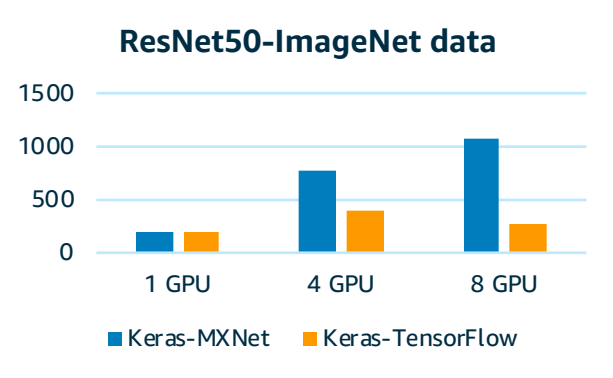
Setup: https://github.com/awslabs/keras-apache-mxnet/tree/master/benchmark

	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

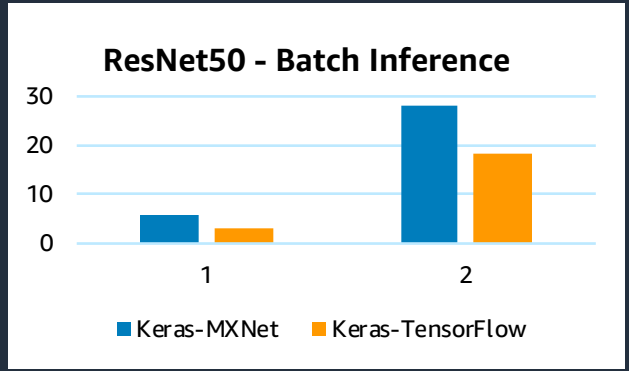


GPUs	Keras-MXNet [Image/sec]	Keras-TensorFlow [Image/sec]	Speed Up
1	194	184	1.05
4	764	393	1.94
8	1068	261	4.09

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

Debuggable

Flexible

Scalable

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))
```

e.g. Inception Stage

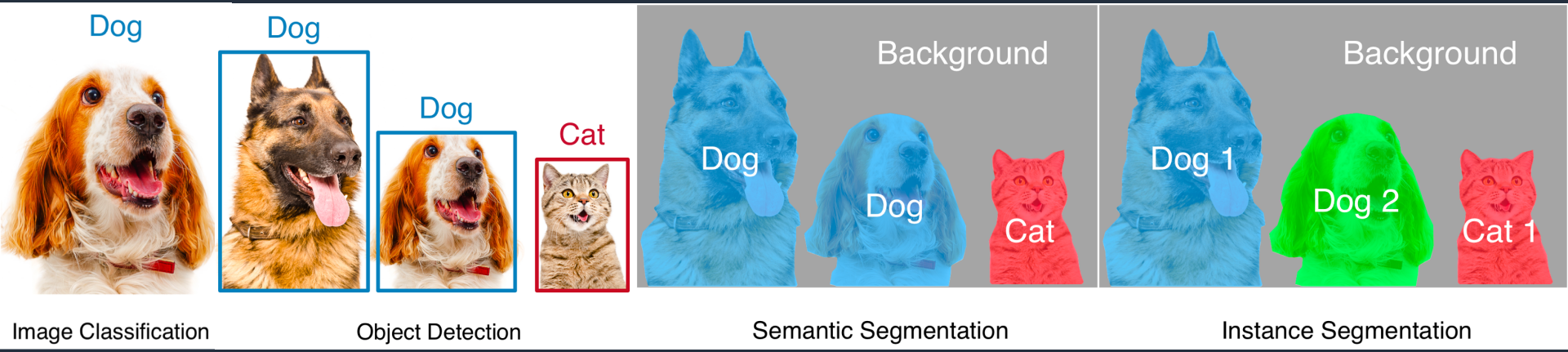
Imperative is “define-by-run”

```
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)
```

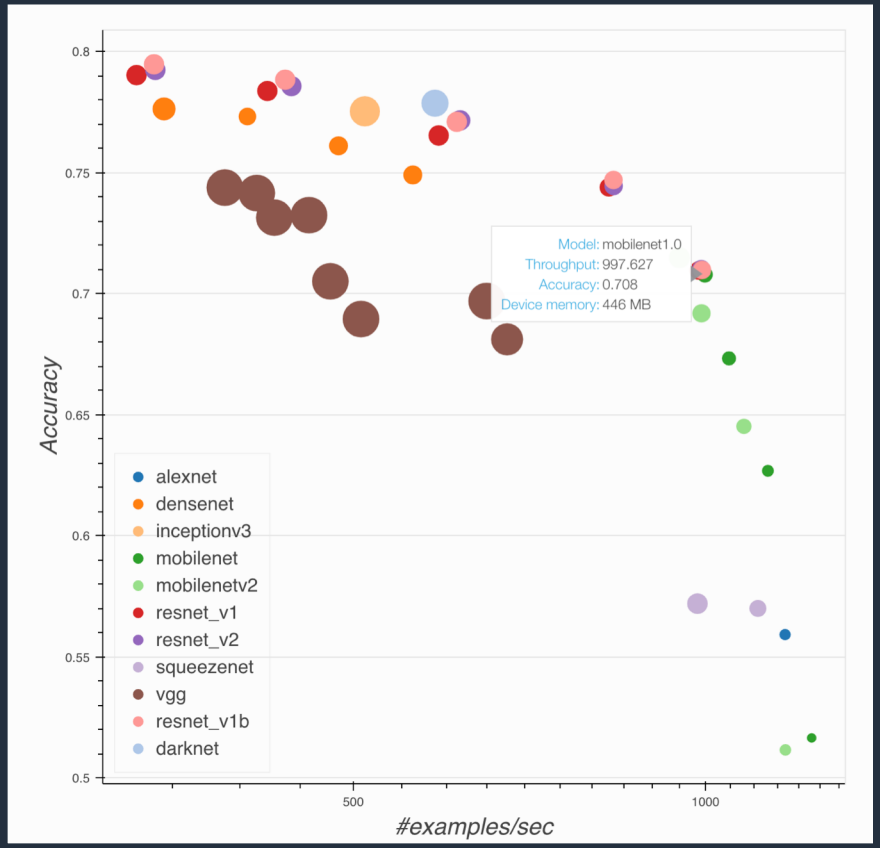
GluonCV: a Deep Learning Toolkit for Computer Vision

https://gluon-cv.mxnet.io

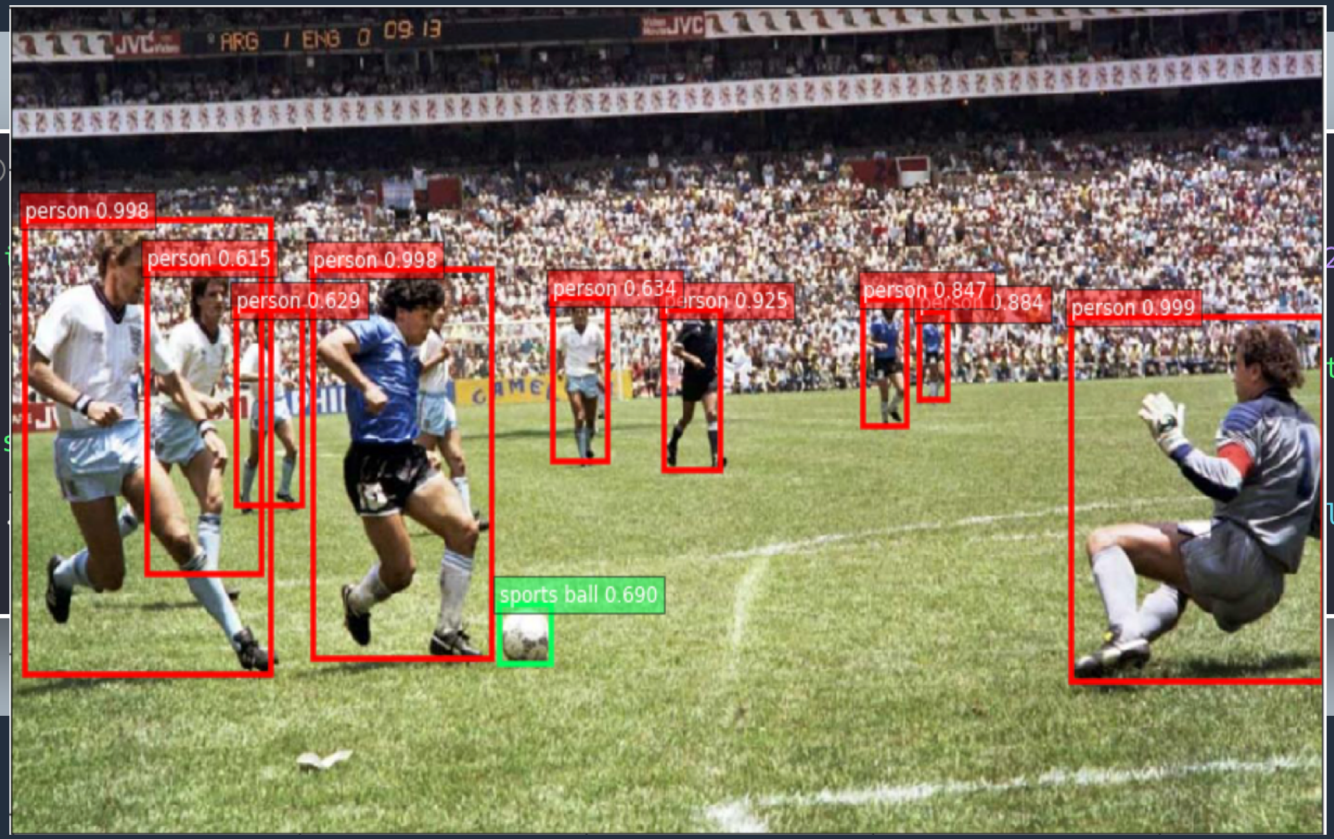


50+ Pre-trained models, with training scripts, datasets, tutorials

GluonCV: pre-trained models, help to choose



GluonCV: example code



GluonNLP: a Deep Learning Toolkit for Natural Language Processing

https://gluon-nlp.mxnet.io

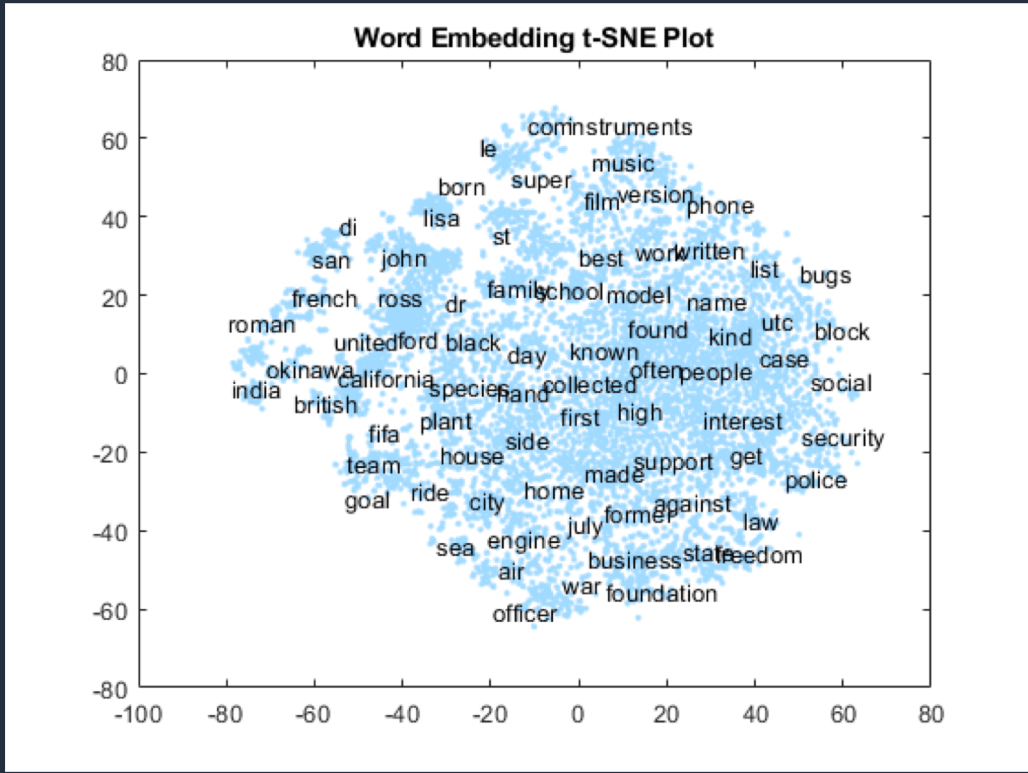
Features (as of 0.3.2)

- 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.

Word embedding:

	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)])
```

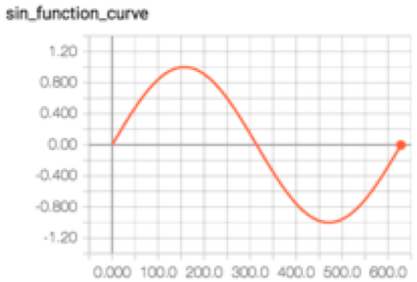
This movie is considered the <?>

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

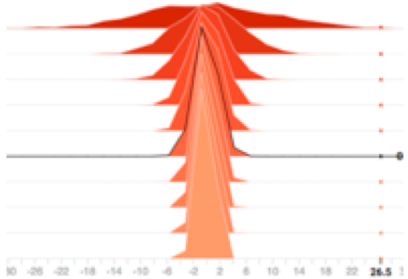
MXBoard: MXNet plugin to TensorBoard

Graph

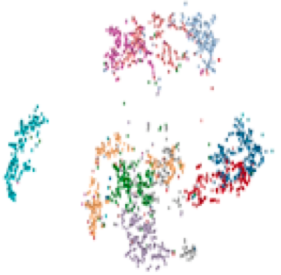
Scalar



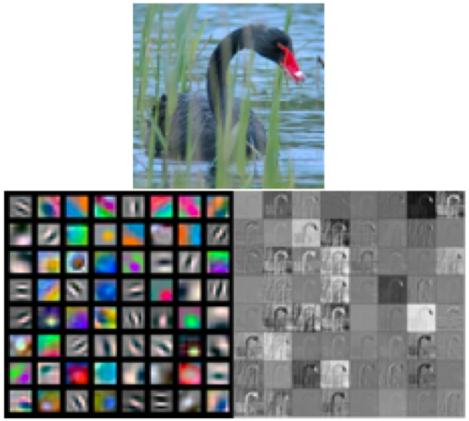
Histogram



Embedding

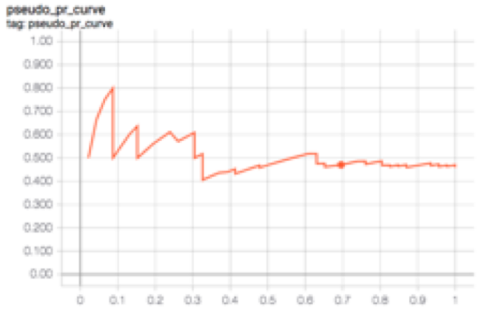


Image

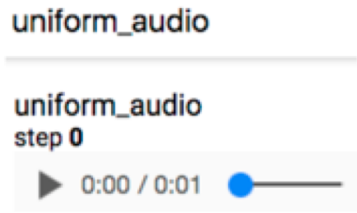


Text

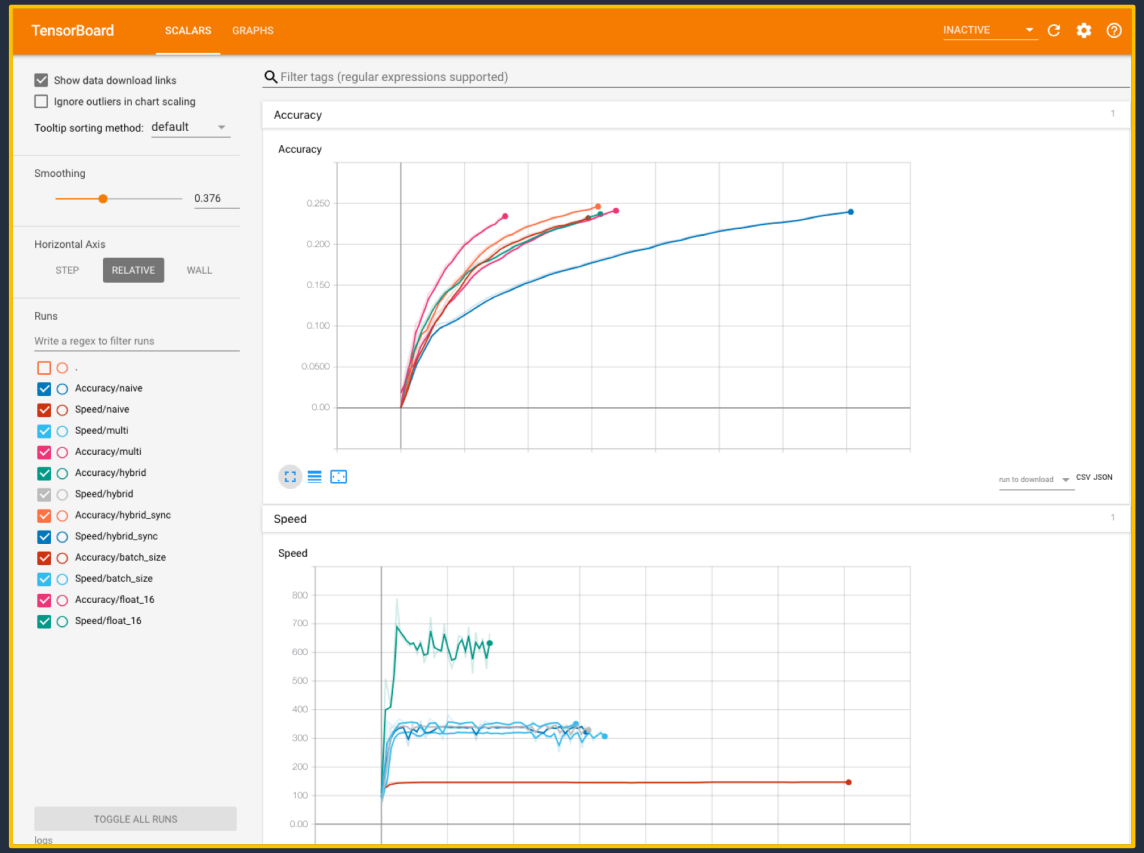
Precision-Recall Curve



Audio



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
Accuracy/naive
Speed/naive
Speed/multi
Accuracy/multi
Accuracy/hybrid
Speed/hybrid
Accuracy/hybrid_sync
Speed/hybrid_sync
Accuracy/batch_size
Speed/batch_size
Accuracy/float_16
Speed/float_16
TOGGLE ALL RUNS
logs
Filter tags (regular expressions supported)
Accuracy
Accuracy
0.250
0.200
0.150
0.100
0.0500
0.00
CSV JSON
Speed
Speed
800
700
600
500
400
300
200
100
0.00

Deep Learning acceleration

CUDA & CuDNN

```
pip install mxnet-cu92
```

TensorRT

```
pip install mxnet-tensorrt-cu92
```

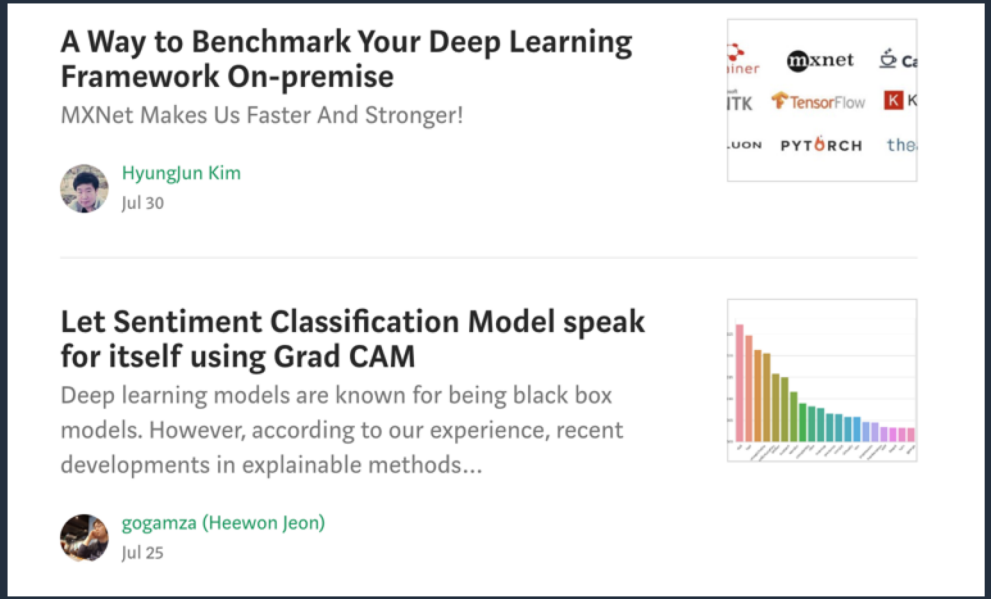
MKL, MKLML & MKLDNN

```
e.g. pip install mxnet-mkl
```

Apache MXNet community

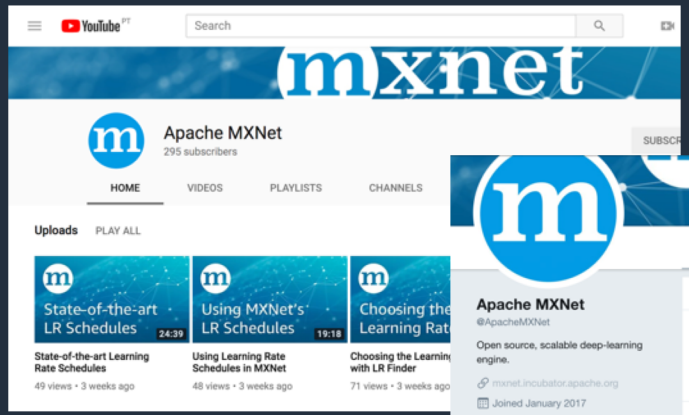
Keeping Up to Date

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



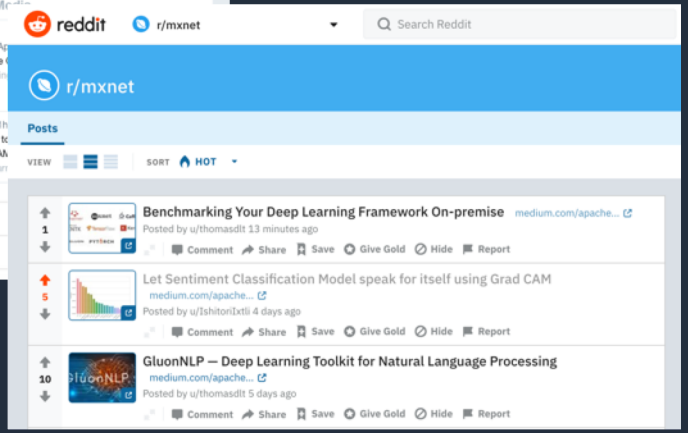
Keeping Up to Date: Social

YouTube: /apachemxnet



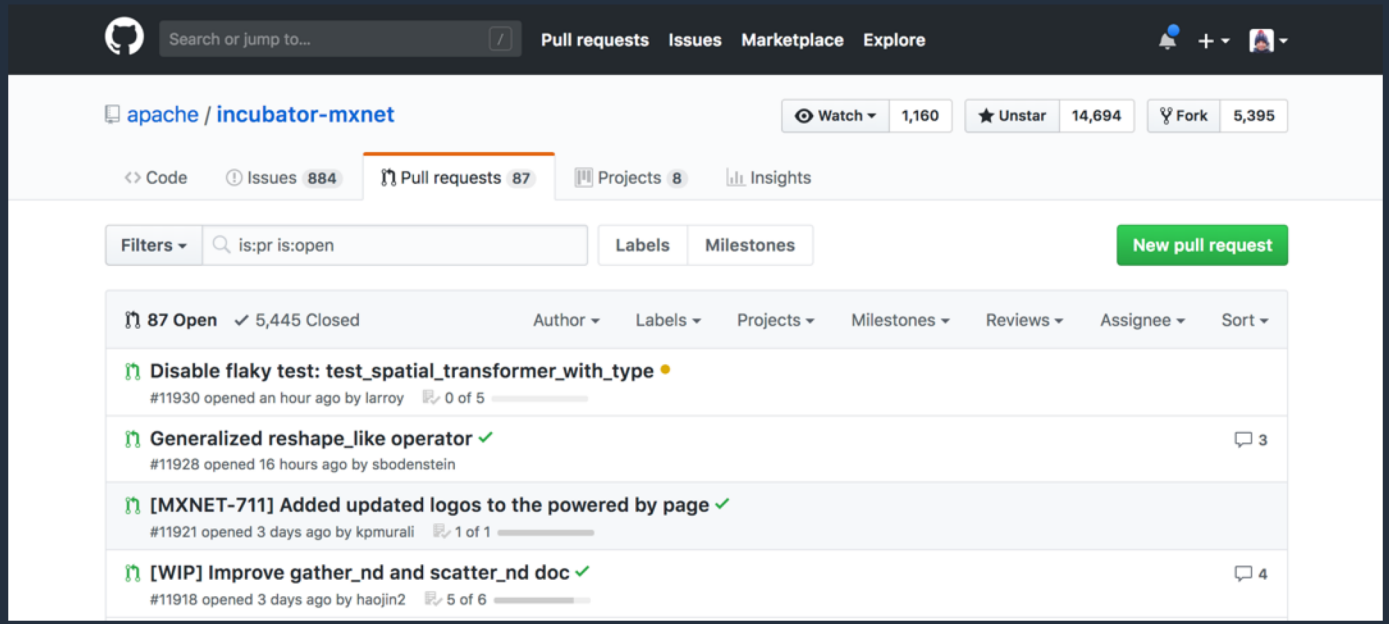
Twitter: @apachemxnet

Reddit: r/mxnet



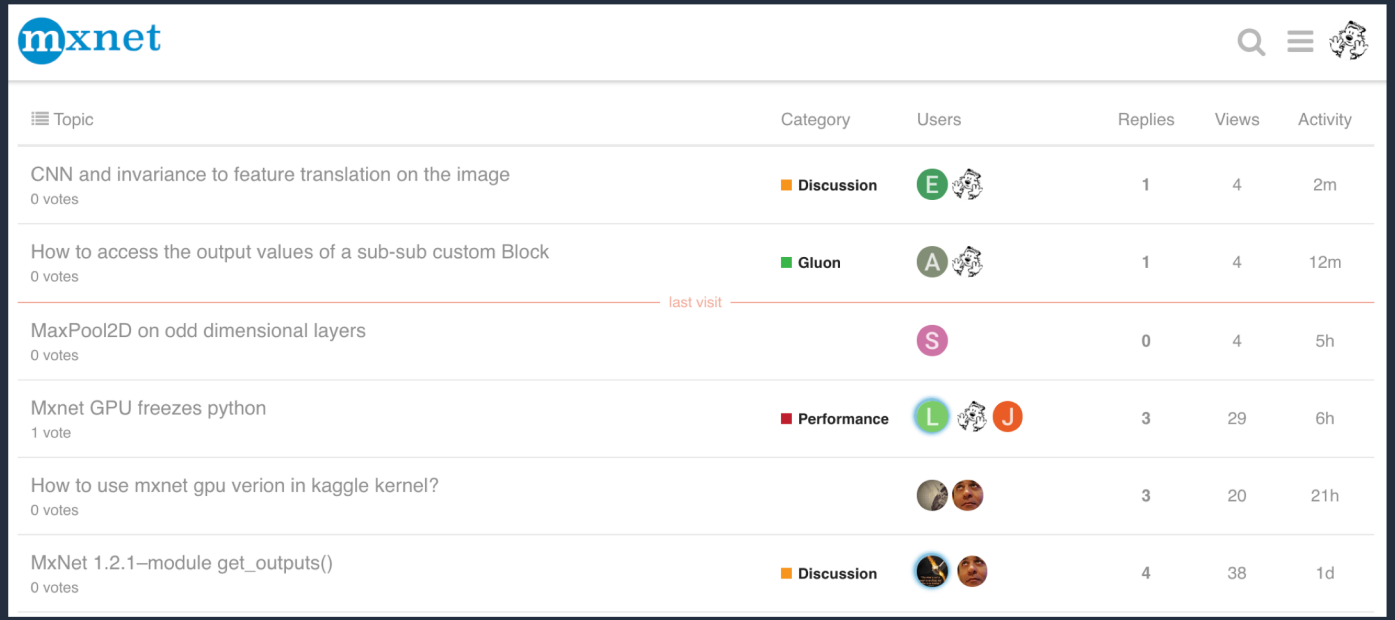
Community

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



Community

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



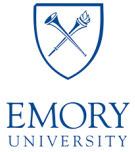
Community

Mailing list:

dev@mxnet.apache.org

user@mxnet.apache.org

MXNet Customer Momentum



Thank you!