主办方：Geekbang. 极客邦科技 InfoQ

QCon

全球软件开发大会【上海站】

Topic of your presentation

Subtitle / Your Name

AWS上的MXNet 深度学习框架
Getting start MXNet on AWS

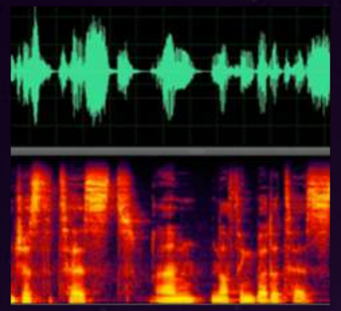
费良宏，AWS Technical Evangelist

lianghon@amazon.com

深度学习的发展

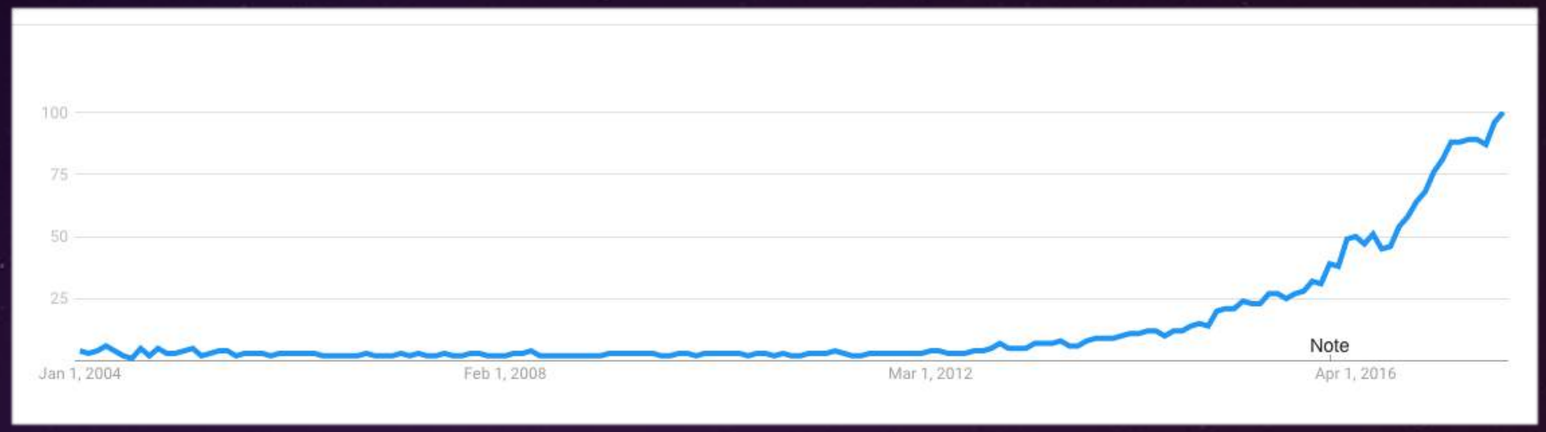
图像

声音



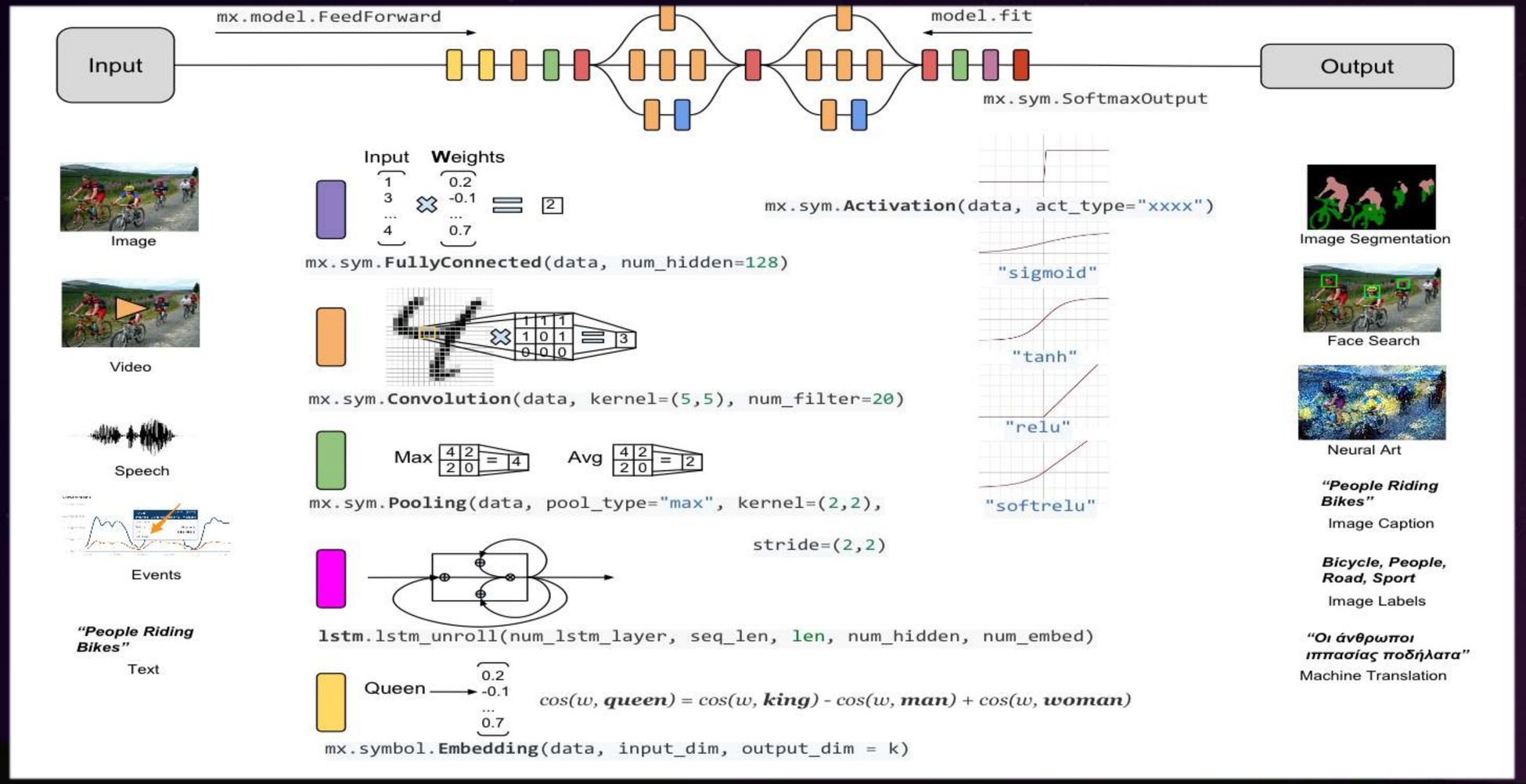
自然语言处理

机器自主



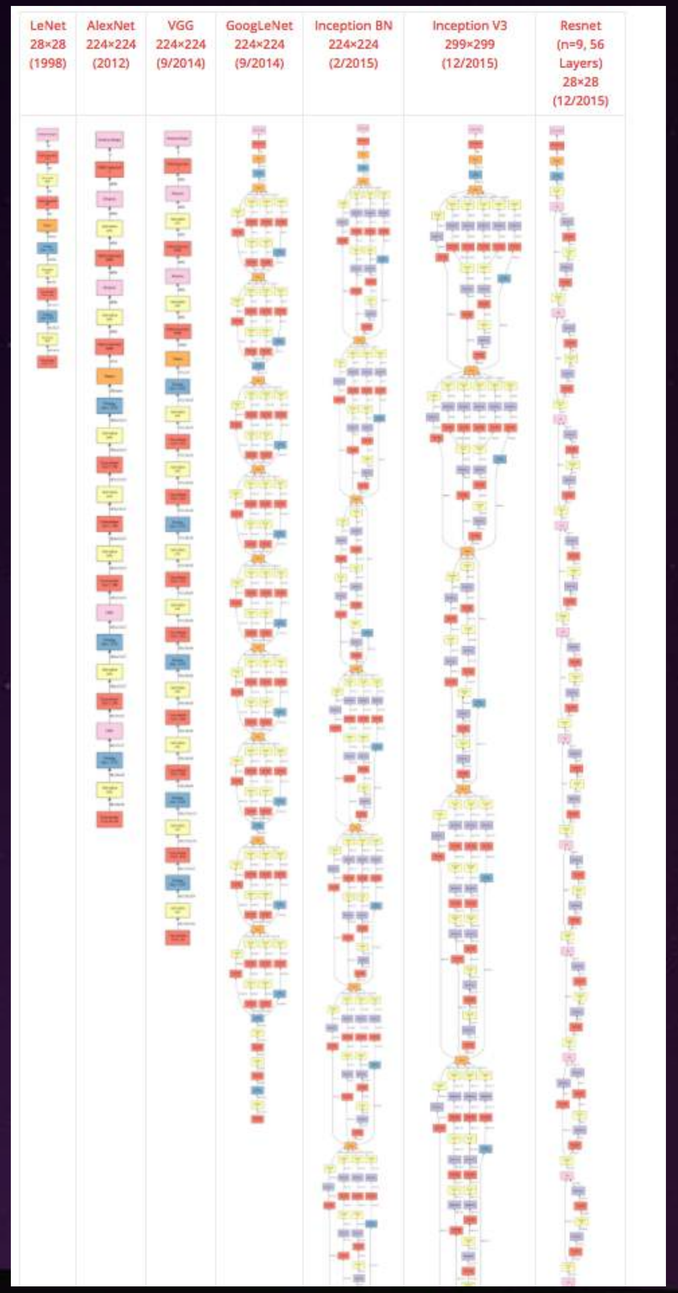
来源：https://trends.google.com/

深度学习的模型


mx.model.FeedForward
model.fit
Input
Output
mx.sym.SoftmaxOutput
Input
Weights
mx.sym.FullyConnected(data, num_hidden=128)
mx.sym.Activation(data, act_type="xxxx")
"sigmoid"
"tanh"
"relu"
"softrelu"
mx.sym.Convolution(data, kernel=(5,5), num_filter=20)
Max
Avg
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)
Queen
cos(w, queen) = cos(w, king) - cos(w, man) + cos(w, woman)
mx.symbol.Embedding(data, input_dim, output_dim = k)
Image
Video
Speech
Events
"People Riding Bikes"
Text
Image Segmentation
Face Search
Neural Art
"People Riding Bikes"
Image Caption
Bicycle, People, Road, Sport
Image Labels
"Οι άνθρωποι ιππασίας ποδήλατα"
Machine Translation

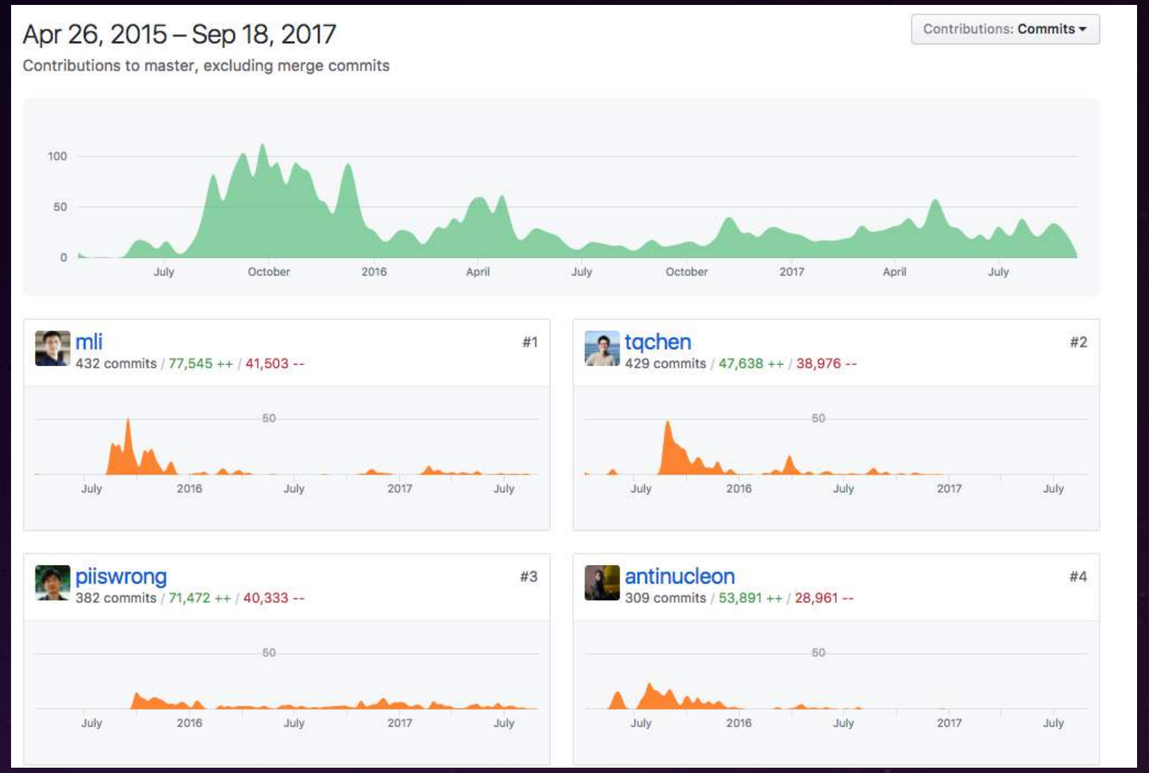
深度学习的模型

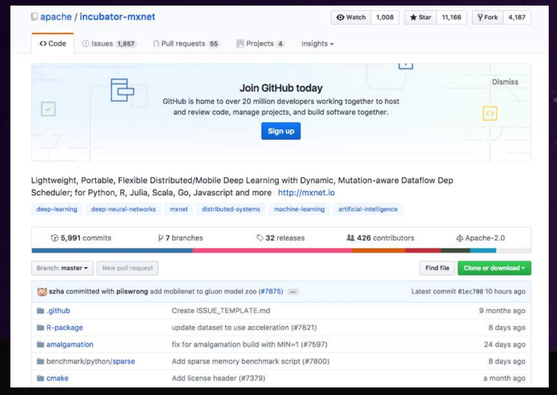
```python
#!/usr/bin/env python3
# -*- coding:utf-8 -*-
# Writen by Lianghong  2017-09-18 12:17:33

import find_mxnet
import mxnet as mx
import importlib

name = "inception-v3"
net = importlib.import_module("symbol_" + name).get_symbol(2)
a = mx.viz.plot_network(
    net, shape={"data":(1, 1, 299, 299)},
    node_attrs={"shape":'rect',"fixedsize":'false'})
a.render(name)
```

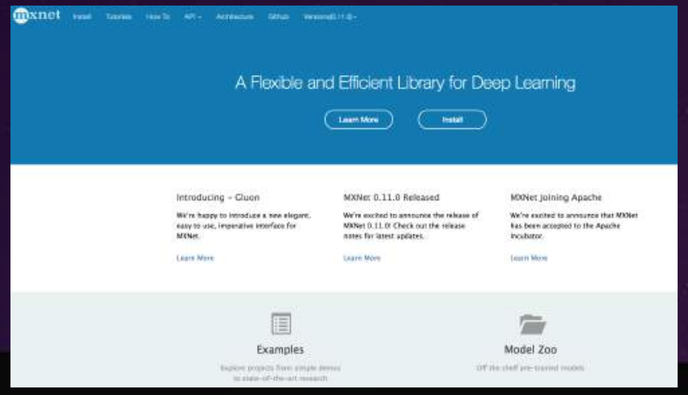


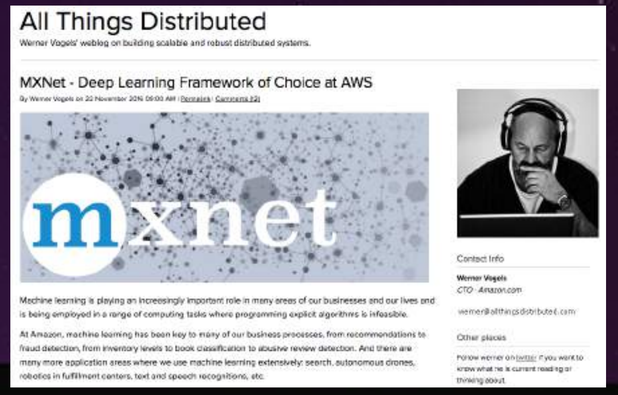
关于 Apache MXNet

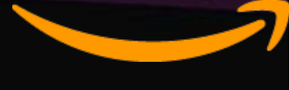
- 2015 年项目启动
- 2016 年11月 被 Amazon 选中
- 2017 年 1 月 成为 Apache 孵化项目
- 当前版本：**0.11**
- 代码贡献者 ：428 人
- Star 数量：11,262
- 许可证 ： Apache 2.0











Apache MXNet 的新版本 0.11

MXNet 0.11.0

nswamy released this 20 days ago · **200 commits** to master since this release

0.11.0

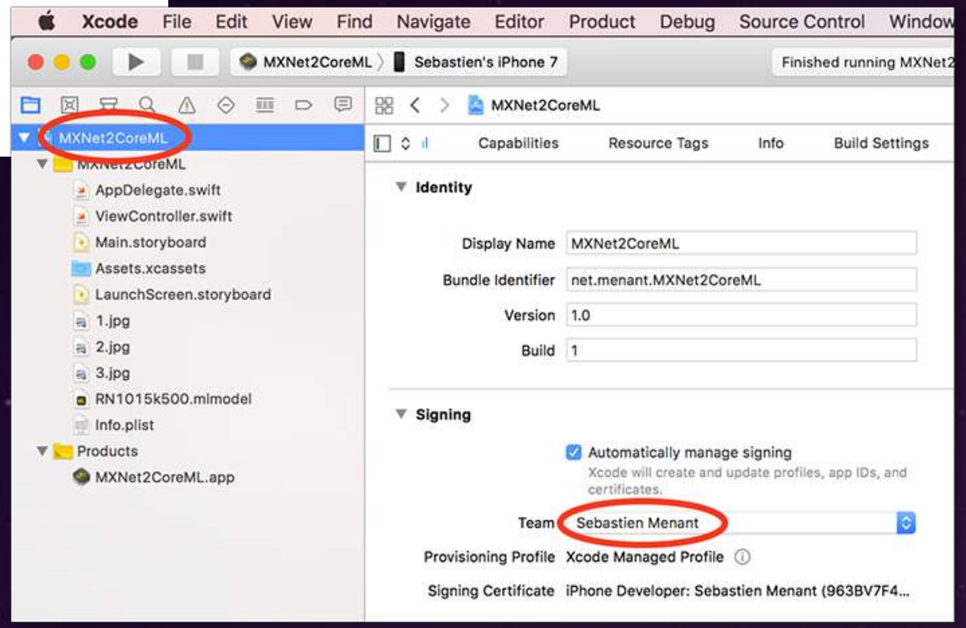
Major Features

- Apple Core ML model converter
- Support for Keras v1.2.2
- Gluon Interface (experimental)
- Updated LICENSE and NOTICE files.
- For more information see full release notes

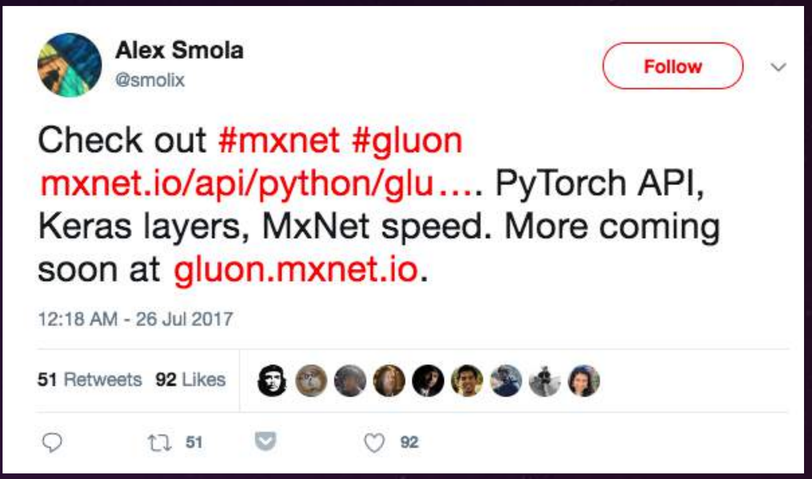
Holy moly! MXNet is **60% faster**: 25 seconds per epoch instead of 61. Very nice. In the same time frame, this would definitely allow us to try more things, like different model architectures or different hyper parameters. Definitely an advantage when you're experimenting.

What about memory usage? As we can see, MXNet uses over **90% less RAM** and there is plenty left for other jobs.

- Julien Simon, https://medium.com/@julsimon/keras-shoot-out-tensorflow-vs-mxnet-51ae2b30a9c0



https://aws.amazon.com/cn/blogs/ai/bring-machine-learning-to-ios-apps-using-apache-mxnet-and-apple-core-ml/

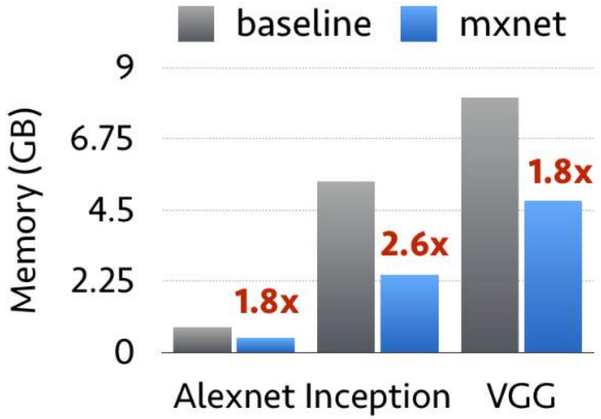


aws

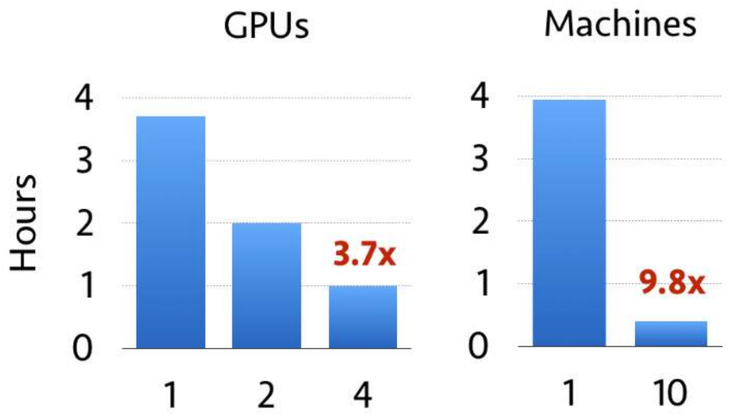
Apache MXNet 框架特点

Portable

Efficient



Scalable



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

Apache MXNet 框架特点 – 广泛的支持

```python
import picamera
from picamera.array import PiRGBArray
import inception_predict
import pollyApi as pol
import os

filename = '/tmp/capture.jpg'

def init():
    polly = pol.pollyInit()
    camera = picamera.PiCamera()
    camera.resolution=(640,480)
    return polly,camera

def capture(polly,camera):
    camera.capture(filename)
    camera.close()

    topn = inception_predict.predict_from_local_file(filename, N=5)
    print(topn)
    top1 = topn[0]
    # Convert probability to integer percentage
    prob = (str)((int)(top1[0]*100))
    # Remove category number
    item = top1[1].split(' ')
    item = ' '.join(item[1:])
    message = "I'm "+prob+"% sure that this is a "+item+". "
    os.remove(filename)
    return message

polly,cam=init()
message=capture(polly,cam)
print(message)
pol.pollySpeak(polly,message)
```

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import os
import boto3
import pygame

pollyRegion = 'us-east-1'
pollyUrl = 'https://polly.us-east-1.amazonaws.com'

def pollyInit():
  pygame.mixer.init()
  polly = connectToPolly()
  return polly

def connectToPolly(regionName=pollyRegion, endpointUrl=pollyUrl):
    return boto3.client('polly', region_name=regionName, endpoint_url=endpointUrl)

def pollySpeak(polly, text, format='mp3', voice='Amy'):
    resp = polly.synthesize_speech(OutputFormat=format, Text=text, VoiceId=voice)
    soundfile = open('/tmp/sound.mp3', 'wb')
    soundBytes = resp['AudioStream'].read()
    soundfile.write(soundBytes)
    soundfile.close()
    pygame.mixer.music.load('/tmp/sound.mp3')
    pygame.mixer.music.play()
    while pygame.mixer.music.get_busy():
        pygame.time.Clock().tick(10)
    os.remove('/tmp/sound.mp3')
```

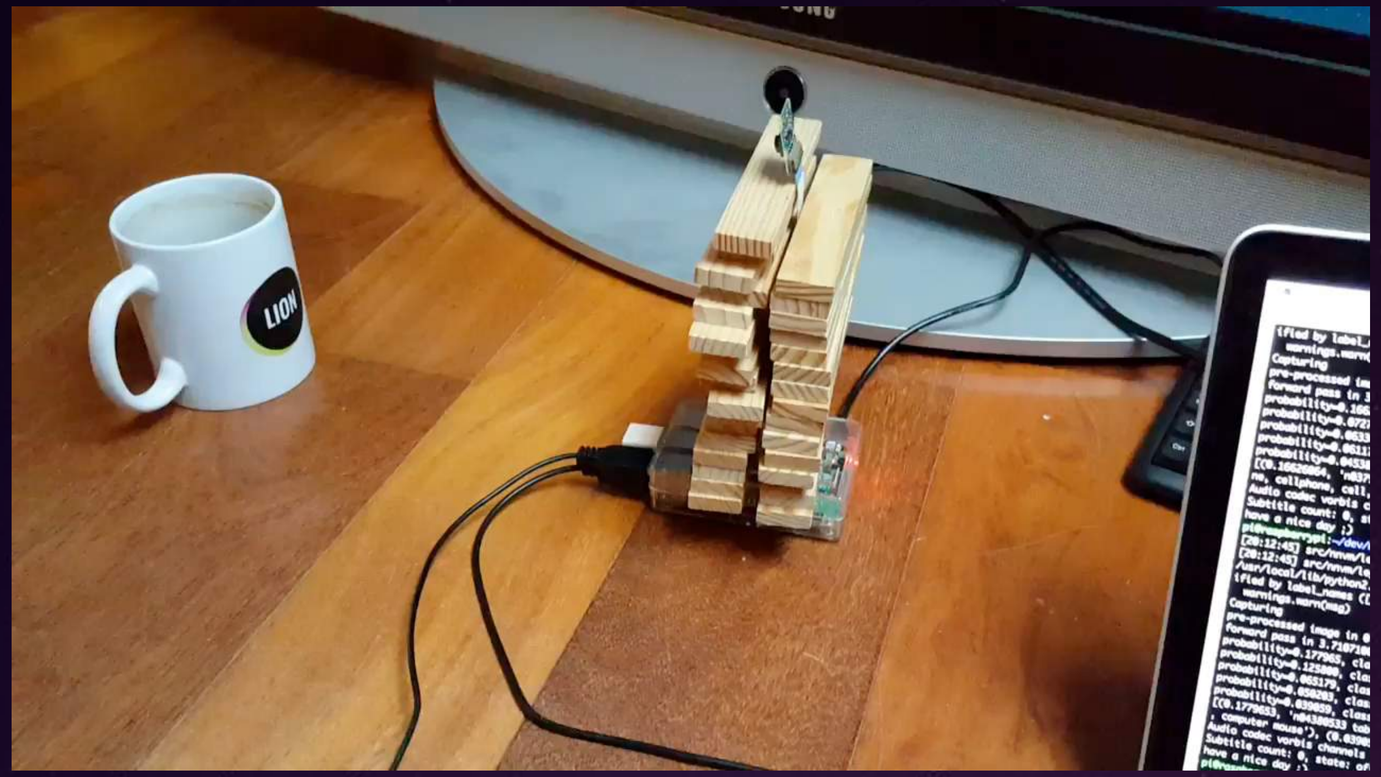
mxnet

Amazon AI

Intelligent Services Powered By Deep Learning

```python
import mxnet as mx
import numpy as np
import cv2, os, urllib, time
from collections import namedtuple
Batch = namedtuple('Batch', ['data'])

# Load the symbols for the networks
with open('synset.txt', 'r') as f:
    synsets = [l.rstrip() for l in f]

# Load the network parameters
sym, arg_params, aux_params = mx.model.load_checkpoint('Inception-BN', 0)

# Load the network into an MXNet module and bind the corresponding parameters
mod = mx.mod.Module(symbol=sym, context=mx.cpu())
mod.bind(for_training=False, data_shapes=[('data', (1,3,224,224))])
mod.set_params(arg_params, aux_params)

'''
Function to predict objects by giving the model a pointer to an image file and running a forward pass through the model.

inputs:
filename = jpeg file of image to classify objects in
mod = the module object representing the loaded model
synsets = the list of symbols representing the model
N = Optional parameter denoting how many predictions to return (default is top 5)

outputs:
python list of top N predicted objects and corresponding probabilities
'''
def predict(filename, mod, synsets, N=5):
    tic = time.time()
    img = cv2.cvtColor(cv2.imread(filename), cv2.COLOR_BGR2RGB)
    if img is None:
        return None
    img = cv2.resize(img, (224, 224))
    img = np.swapaxes(img, 0, 2)
    img = np.swapaxes(img, 1, 2)
    img = img[np.newaxis, :]
    print("pre-processed image in "+str(time.time()-tic))

    toc = time.time()
    mod.forward(Batch([mx.nd.array(img)]))
    prob = mod.get_outputs()[0].asnumpy()
    prob = np.squeeze(prob)
    print("forward pass in "+str(time.time()-toc))


    topN = []
    a = np.argsort(prob)[::-1]
    for i in a[0:N]:
        print('probability=%f, class=%s' %(prob[i], synsets[i]))
        topN.append((prob[i], synsets[i]))
    return topN


# Code to download an image from the internet and run a prediction on it
def predict_from_url(url, N=5):
    filename = url.split("/")[-1]
    urllib.urlretrieve(url, filename)
    img = cv2.imread(filename)
    if img is None:
        printi("Failed to download")
    else:
        return predict(filename, mod, synsets, N)

# Code to predict on a local file
def predict_from_local_file(filename, N=5):
    return predict(filename, mod, synsets, N)
```

深度学习的模型



mxnet

Amazon AI

Intelligent Services Powered By Deep Learning

aws

Apache MXNet 框架特点 – 融合（Amalgamation）

- 将所需的库合并到单独的一个C++文件中
- 可在任何平台编译

BlindTool by Joseph Paul Cohen, demo on Nexus 4

在浏览器中运行的JAVASCRIPT 代码

Apache MXNet 框架特点 – 高效

High-level MXNet Example

In the interest of comparison; a common (custom) data-generator (called yield_mb(X, y, batchsize=64, shuffle=False)) was originally used for all other frameworks - but not for MXNet. I have reproduced the MXNet example using this same generator (wrapping the results in the mx.io.DataBatch class) to test if MXNet is faster than other frameworks just because I was using its own data-generator. This does not appear to be the case.

[1]:
```
import os
import sys
import numpy as np
import mxnet as mx
from common.params import *
from common.utils import *
```

```
/anaconda/envs/py35/lib/python3.5/site-packages/urllib3/contrib/pyopenssl.py:46: DeprecationWarnin
g: OpenSSL.rand is deprecated - you should use os.urandom instead
  import OpenSSL.SSL
```

[2]:
```
print("OS: ", sys.platform)
print("Python: ", sys.version)
print("Numpy: ", np.__version__)
print("MXNet: ", mx.__version__)
print("GPU: ", get_gpu_name())
```

```
OS:  linux
Python:  3.5.2 |Anaconda custom (64-bit)| (default, Jul  2 2016, 17:53:06)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-1)]
Numpy:  1.13.1
MXNet:  0.11.0
GPU:  ['Tesla K80', 'Tesla K80']
```

[3]:
```
def create_symbol():
    data = mx.symbol.Variable('data')
    # size = [(old-size - kernel + 2*padding)/stride]+1
    # if kernel = 3, pad with 1 either side
    conv1 = mx.symbol.Convolution(data=data, num_filter=50, pad=(1,1), kernel=(3,3))
    relu1 = mx.symbol.Activation(data=conv1, act_type="relu")
    conv2 = mx.symbol.Convolution(data=relu1, num_filter=50, pad=(1,1), kernel=(3,3))
    pool1 = mx.symbol.Pooling(data=conv2, pool_type="max", kernel=(2,2), stride=(2,2))
    relu2 = mx.symbol.Activation(data=pool1, act_type="relu")
    drop1 = mx.symbol.Dropout(data=relu2, p=0.25)

    conv3 = mx.symbol.Convolution(data=drop1, num_filter=100, pad=(1,1), kernel=(3,3))
    relu3 = mx.symbol.Activation(data=conv3, act_type="relu")
    conv4 = mx.symbol.Convolution(data=relu3, num_filter=100, pad=(1,1), kernel=(3,3))
    pool2 = mx.symbol.Pooling(data=conv4, pool_type="max", kernel=(2,2), stride=(2,2))
    relu4 = mx.symbol.Activation(data=pool2, act_type="relu")
    drop2 = mx.symbol.Dropout(data=relu4, p=0.25)
```

来源：https://github.com/Ilkarman/DeepLearningFrameworks

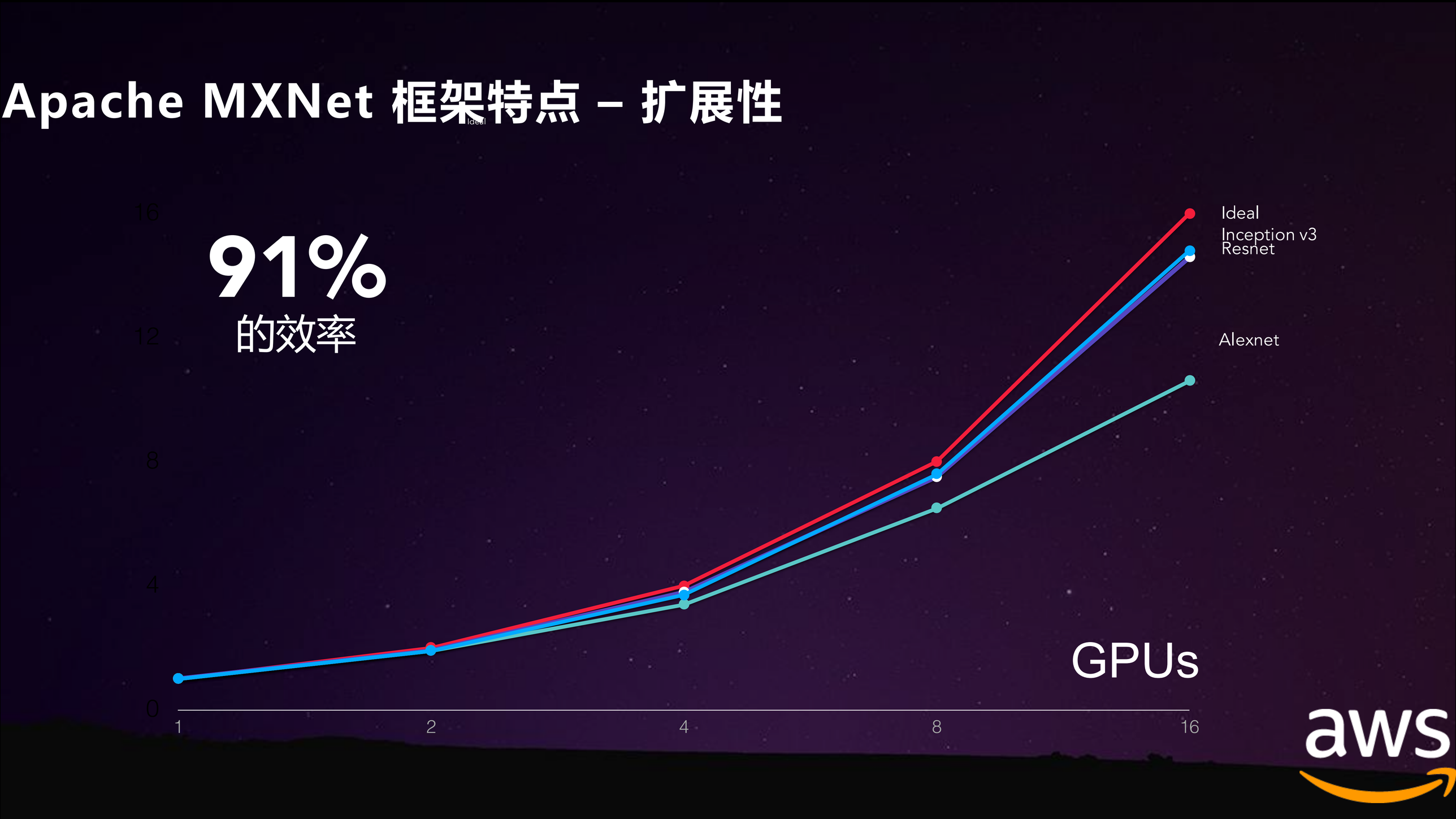
VGG-style CNN on CIFAR-10

DL Library	Test Accuracy (%)	Training Time (s)
Caffe2	79	149
MXNet	77	149
Gluon	77	157
CNTK	78	166
PyTorch	78	168
Tensorflow	78	173
Keras(CNTK)	78	200
Chainer	79	240
Keras(TF)	77	252
Lasagne(Theano)	77	253
Keras(Theano)	78	269

LSTM(GRU) on IMDB

DL Library	Test Accuracy (%)	Training Time (s)
Keras(CNTK)	86	223
Tensorflow	86	79
Pytorch	87	36
MXNet	88	12

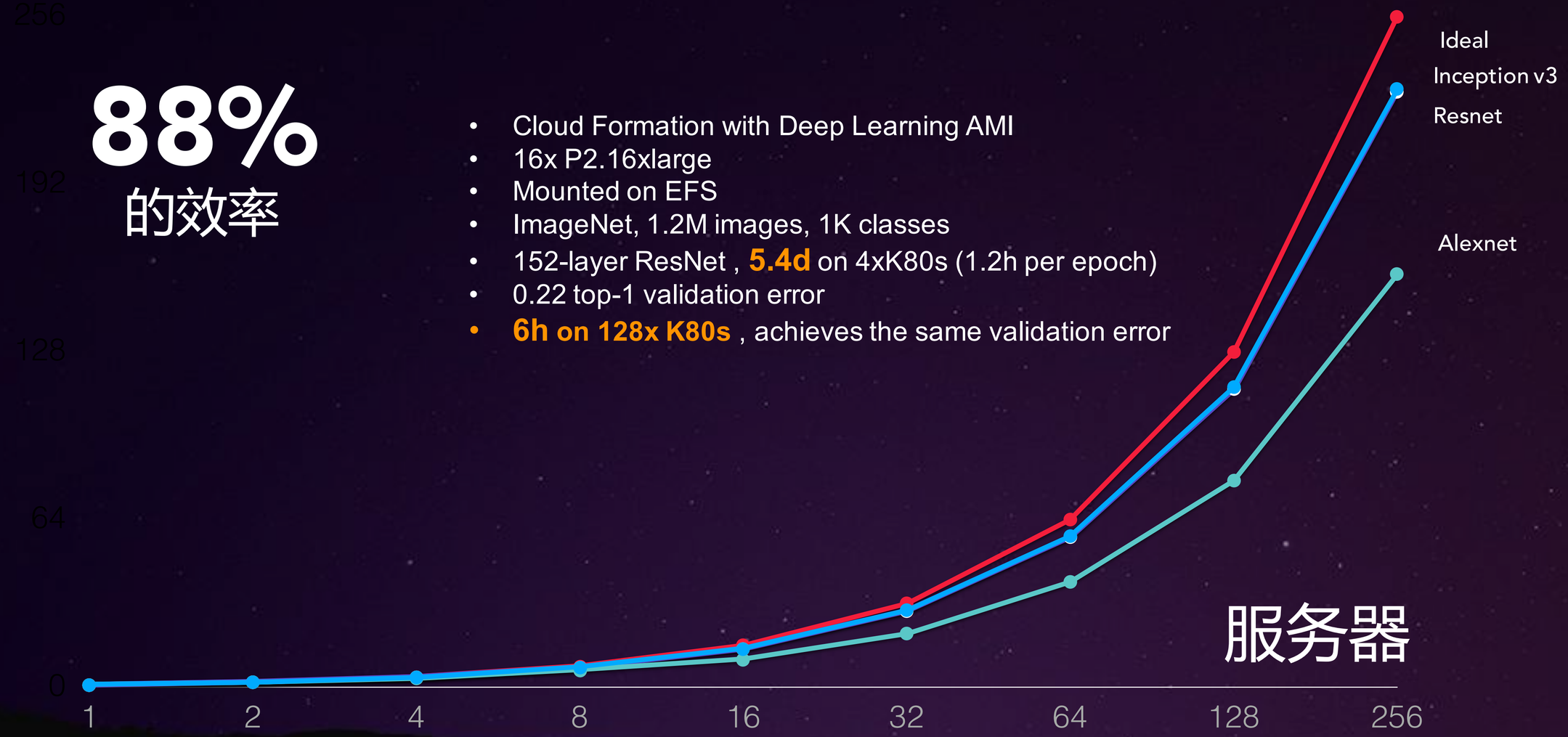
Apache MXNet 框架特点 – 扩展性



Apache MXNet 框架特点 – 扩展性

88%
的效率

- Cloud Formation with Deep Learning AMI
- 16x P2.16xlarge
- Mounted on EFS
- ImageNet, 1.2M images, 1K classes
- 152-layer ResNet , **5.4d** on 4xK80s (1.2h per epoch)
- 0.22 top-1 validation error
- **6h on 128x K80s** , achieves the same validation error



Apache MXNet 框架基础

- **NDArray: 多维数组运算（imperative）**
- **Symbol: 神经网络符号表达式（declarative）**

- **Module**: 神经网络模型训练及接口
- **Loading Data**: 提供数据给训练 / 推理过程

- **混合编程**: 使用NDArray 和 Symbolic 进行模型训练

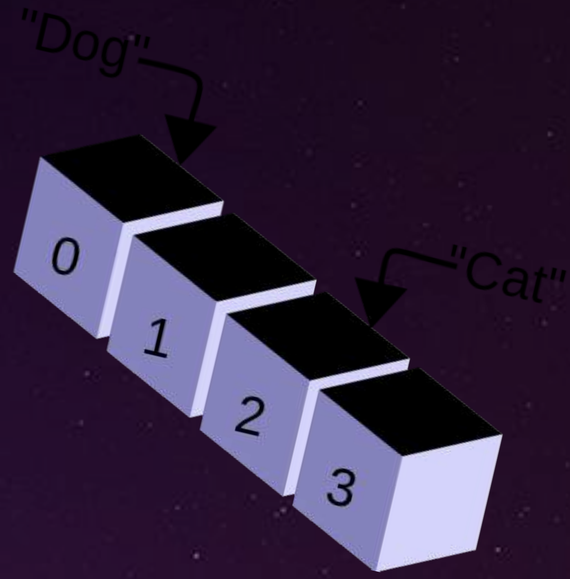
Apache MXNet 框架基础 – NDArry

NumPy

- 仅限于使用 CPU
- 没有自动微分（ automatic differentiation ）
- **阻塞式（计算完成返回结果）**

NDArry

- 通过设备上下文的多个CPUs/GPUs
- 云计算环境下的分布式环境
- 自动微分（ automatic differentiation ）
- **非阻塞式的Python 前端（无GIL）& 懒惰评估（lazy evaluation）**



Apache MXNet 框架基础 – NDArry

context = mx.gpu(0)
context = mx.gpu(1)

context = mx.cpu()

注意！
NDArry 在GPUs 上是不可变的！

g = copyto(c)
g = c.as_in_context(mx.gpu(0))

aws

命令式编程（Imperative）

```
import numpy as np
a = np.ones(10)
b = np.ones(10) * 2
c = b * a
d = c + 1
```

易于针对Python代码进行调整

优点

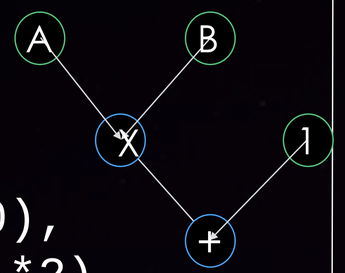
- 简单直接并且灵活
- 可利用程序语言自身的特点（循环、条件判断、调试器）
- 例如： Numpy、Matlab,以及Torch

缺点

- 难于优化

声明式编程（Declarative）

```
A = Variable('A')
B = Variable('B')
C = B * A
D = C + 1
f = compile(D)
d = f(A=np.ones(10),
      B=np.ones(10)*2)
```



C可以与D共享内存，因为C稍后会被删除

优点

- 更多优化的机会
- 可以适用多种程序语言
- 例如： TensorFlow、Theano 以及 Caffe

缺点

- 灵活性不足
- 难于调试
- 无法使用原生的程序代码
- 无动态图

混合编程的范式

命令式

NDARRAY API

```
>>> import mxnet as mx
>>> a = mx.nd.zeros((100, 50))
>>> b = mx.nd.ones((100, 50))
>>> c = a + b
>>> c += 1
>>> print(c)
```

声明式

SYMBOLIC EXECUTOR

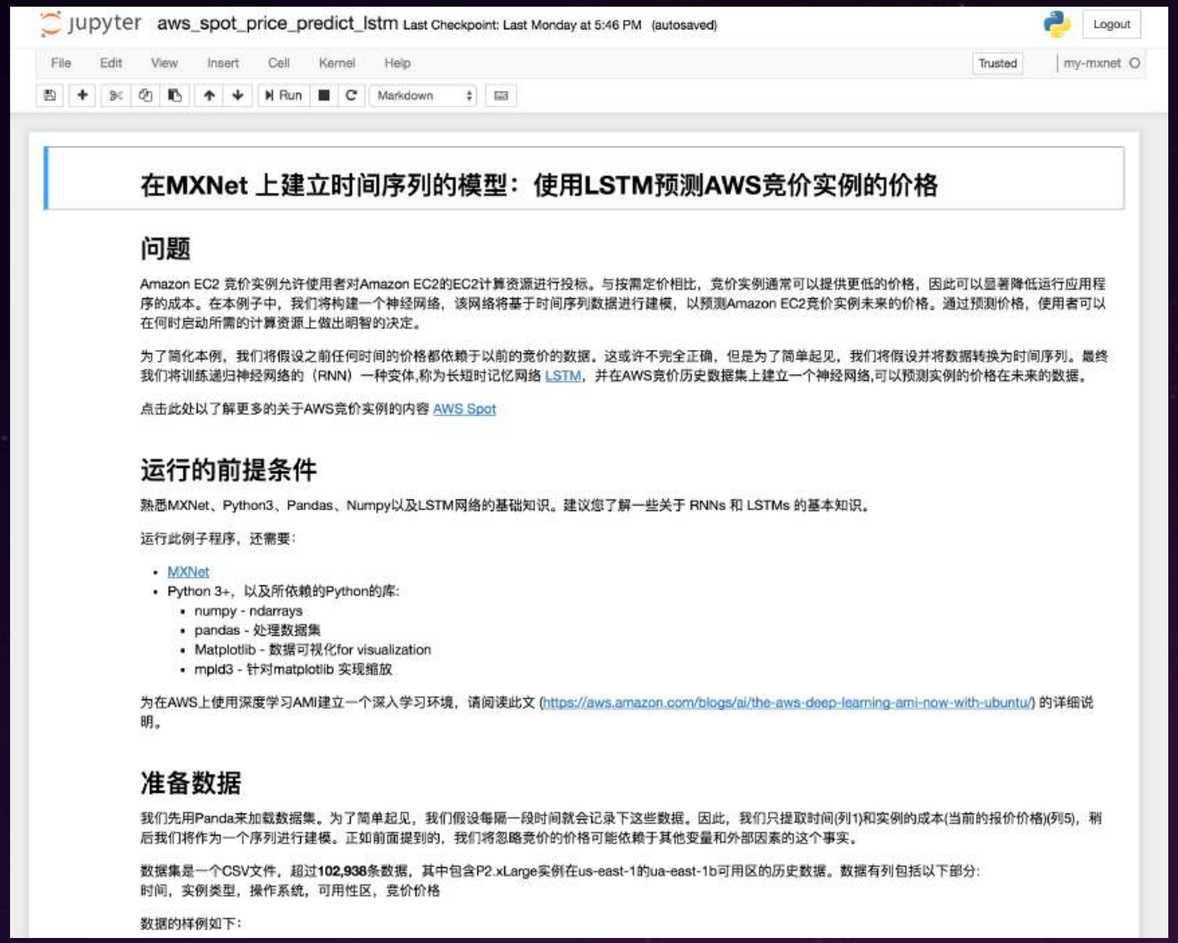
```
>>> import mxnet as mx
>>> net = mx.symbol.Variable('data')
>>> net = mx.symbol.FullyConnected(data=net, num_hidden=128)
>>> net = mx.symbol.SoftmaxOutput(data=net)
>>> texec = mx.module.Module(net)
>>> texec.forward(data=c)
>>> texec.backward()
```

NDArray 可以用来作为 Graph 的输入

Apache MXNet 应用演示 – AWS EC2 竞价实例价格预测

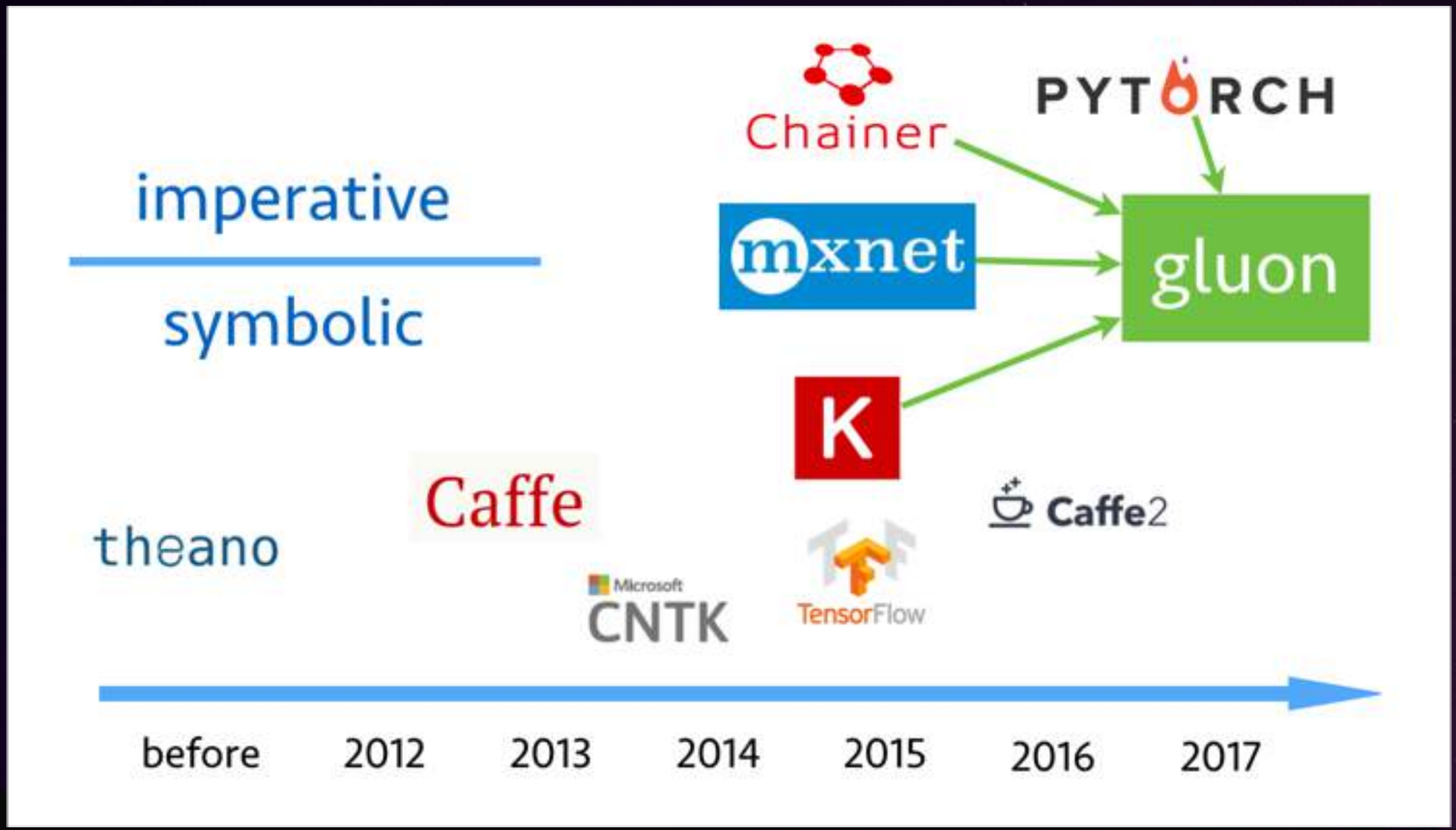
AWS EC2竞价型实例 - 客户可以利用竞价型实例对未使用的 EC2 容量进行竞价，并可在竞价高于当前现货价格的期间内运行此类实例。现货价格根据供需情况定期变化，出价达到或超过竞价的客户可获得相应的竞价型实例。

Apache MXNet 应用演示 – AWS EC2 竞价实例价格预测



链接：https://lianghong.info/ipython/notebooks/spotprice_predict/aws_spot_price_predict_lstm.ipynb

Apache MXNet 框架的新接口 – Gluon



Apache MXNet 框架的新接口 – Gluon

```
net = gluon.nn.Sequential()
with net.name_scope():
    net.add(gluon.nn.Dense(128, activation='relu'))
    net.add(gluon.nn.Dense(64, activation='relu'))
    net.add(gluon.nn.Dense(10))

loss = gluon.loss.SoftmaxCrossEntropyLoss()

for data, label in get_batch():
    with autograd.record():
        l = loss(net(data), label)
    l.backward()
    trainer.step(batch_size=data.shape[0])
```

net.hybridize()

从声明式转换到符号执行