Cloud Operated Open Source AI Robots

Adrian Cockcroft @adrianco

- Applications of Deep Learning
- Apache MXNet Overview
- Apache MXNet API
- Code and DIYrobocars
- Tools and Resources

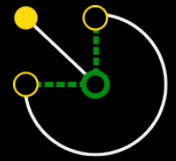
Applications of Deep Learning

The Challenge For AI: Scale

Data

Training

Inference



At the Edge

The Challenge For AI: Scale

Data

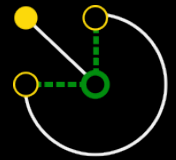
PBs of existing data
New data created on AWS

Training

Tons of GPUs
Elastic capacity
Pre-built images

Inference

Tons of GPUs and CPUs
Serverless



At the Edge

IoT and mobile deployment
Mobile optimization
IoT device optimization

AI On AWS Today

Thousands Of Amazon Engineers Focused On Machine Learning

Fulfilment & logistics

Search & discovery

Existing products

New products

At AWS

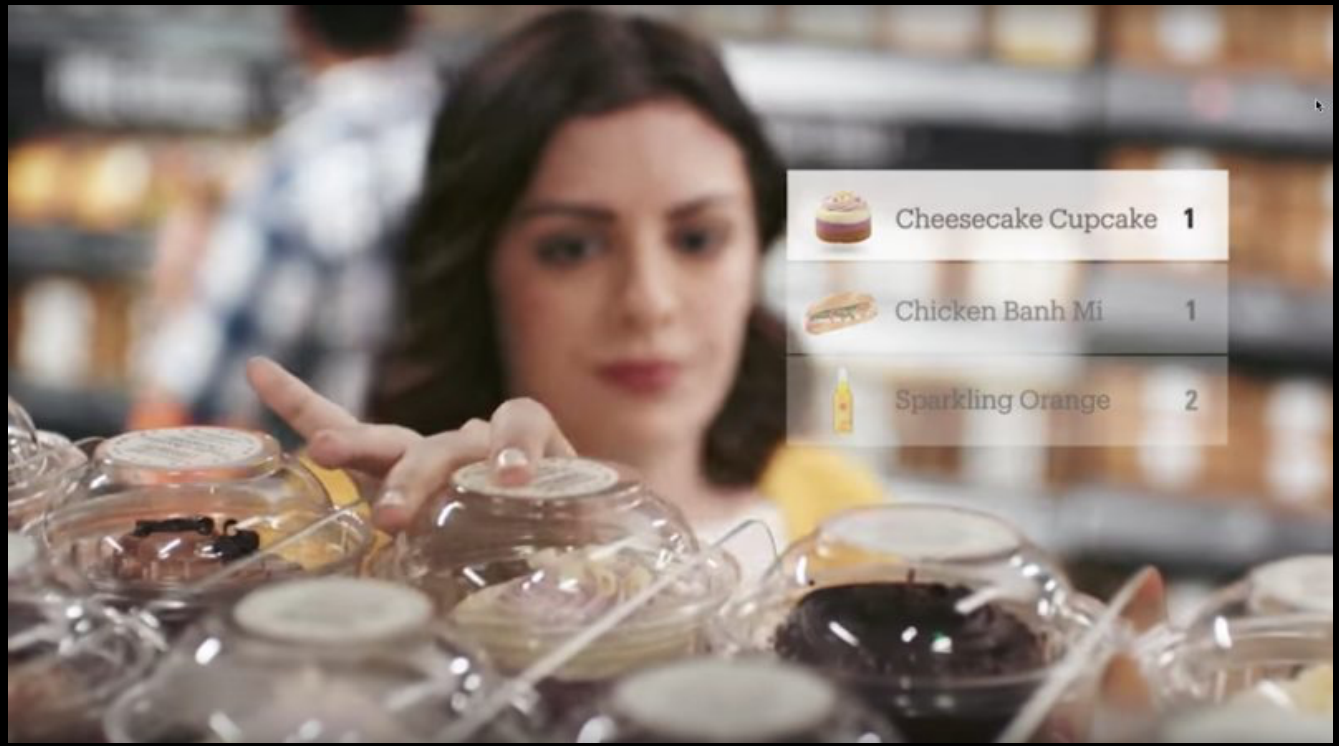
Deep Learning using MXNet @Amazon

- Applied Research
- Core Research
- Alexa
- Demand Forecasting
- Risk Analytics
- Search
- Recommendations
- AI Services | Rek, Lex, Polly
- Q&A Systems
- Supply Chain Optimization
- Advertising
- Machine Translation
- Video Content Analysis
- Robotics
- Lots of Computer Vision..
- Lots of NLP/U..

*Teams are either actively evaluating, in development, or transitioning to scale production

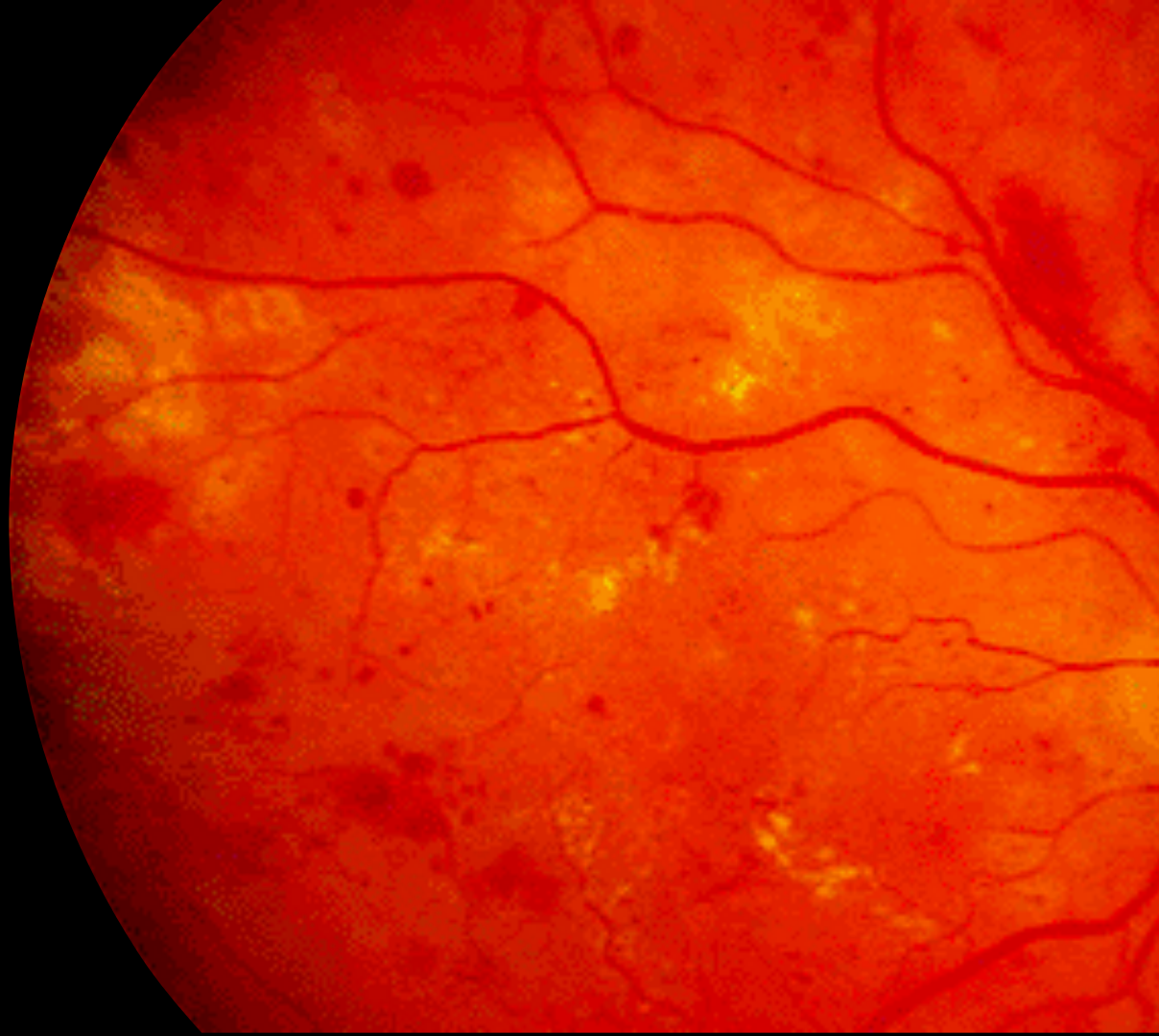

BRAUN

amazon go



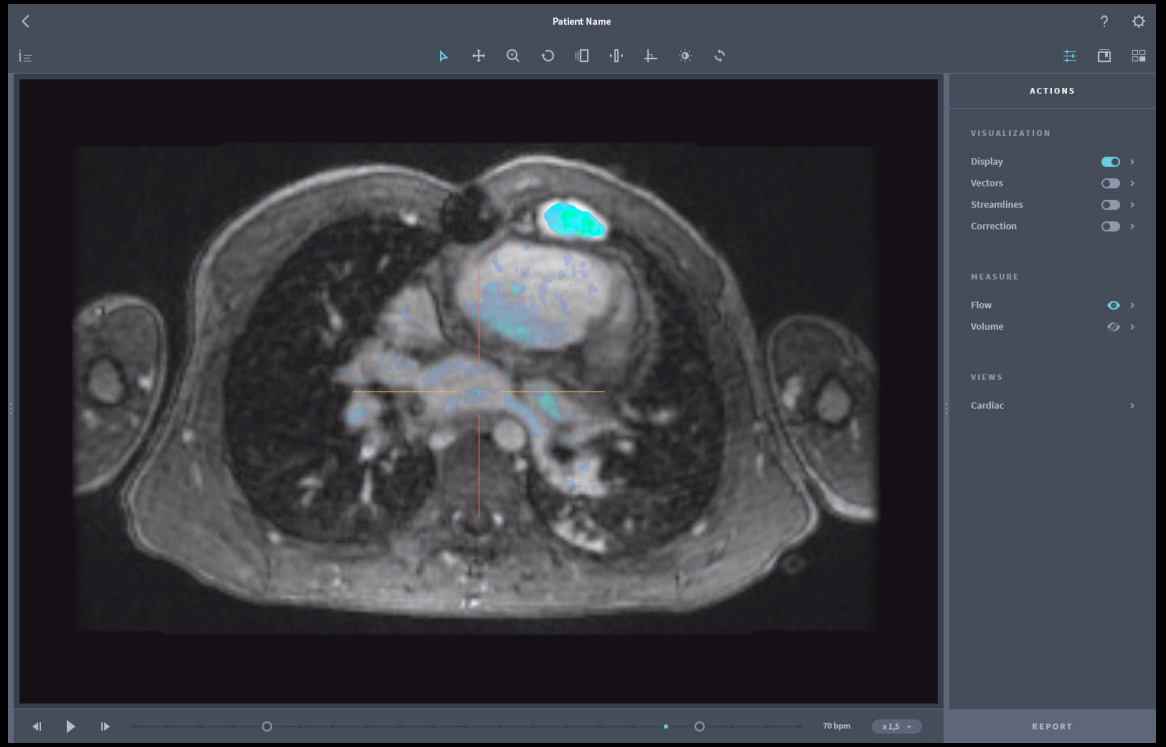
Stanford

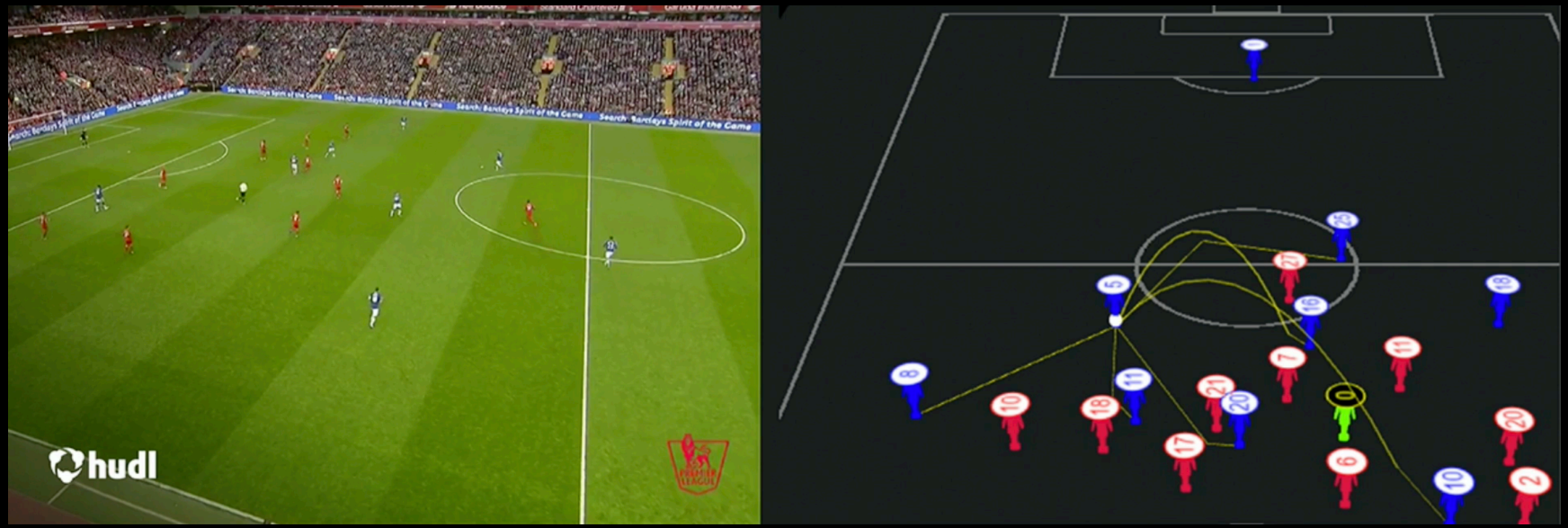
Early detection of diabetic complications



ARTERYS

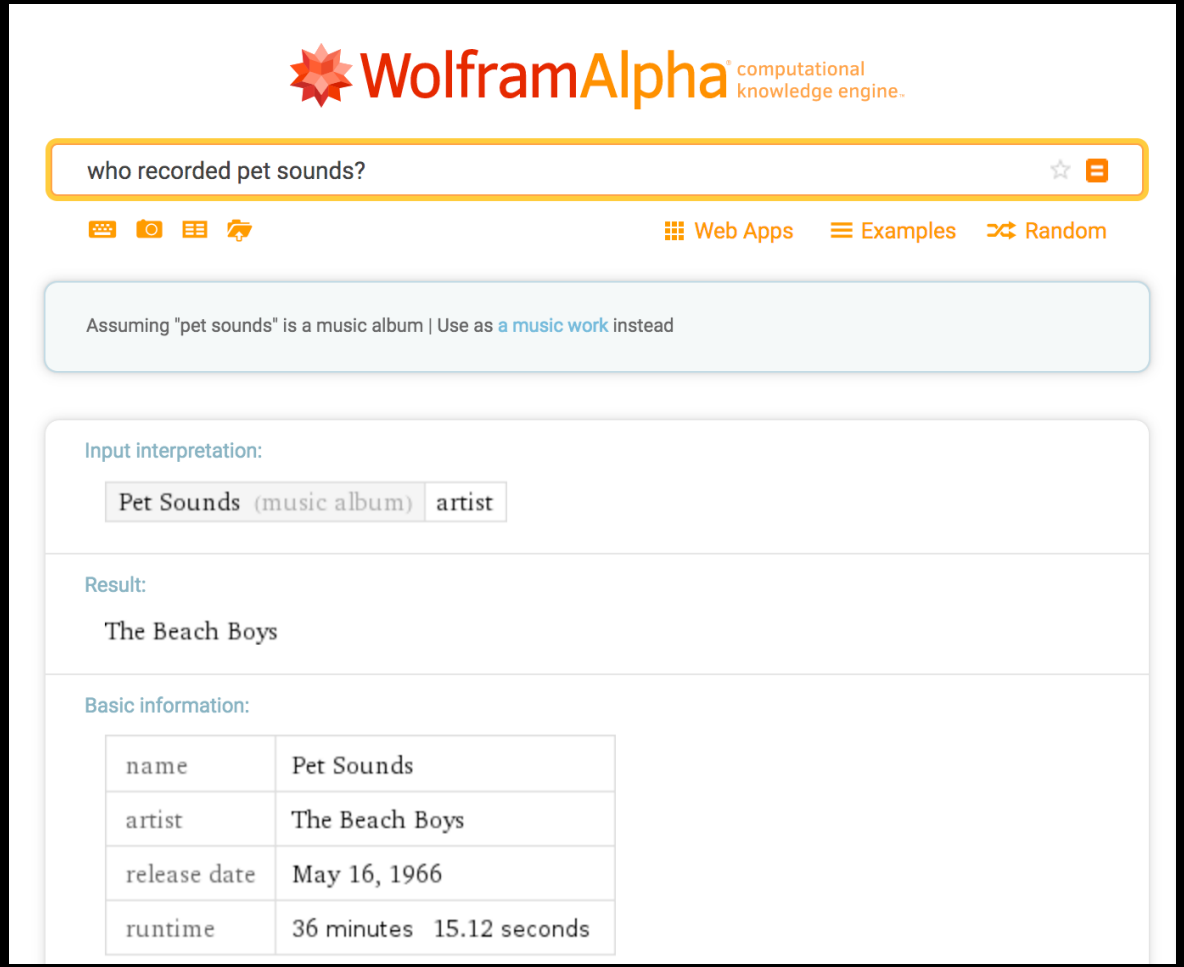
FDA-approved
medical imaging




hudl

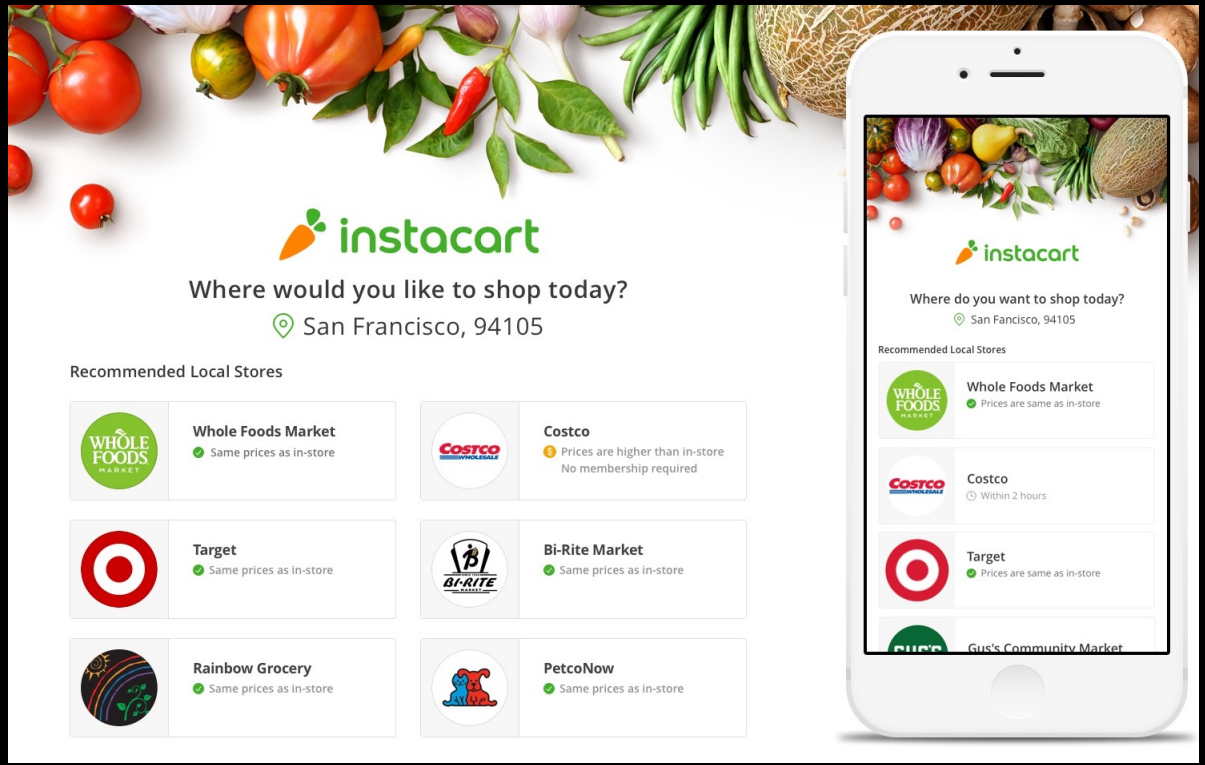
WolframAlpha

Computational knowledge engine



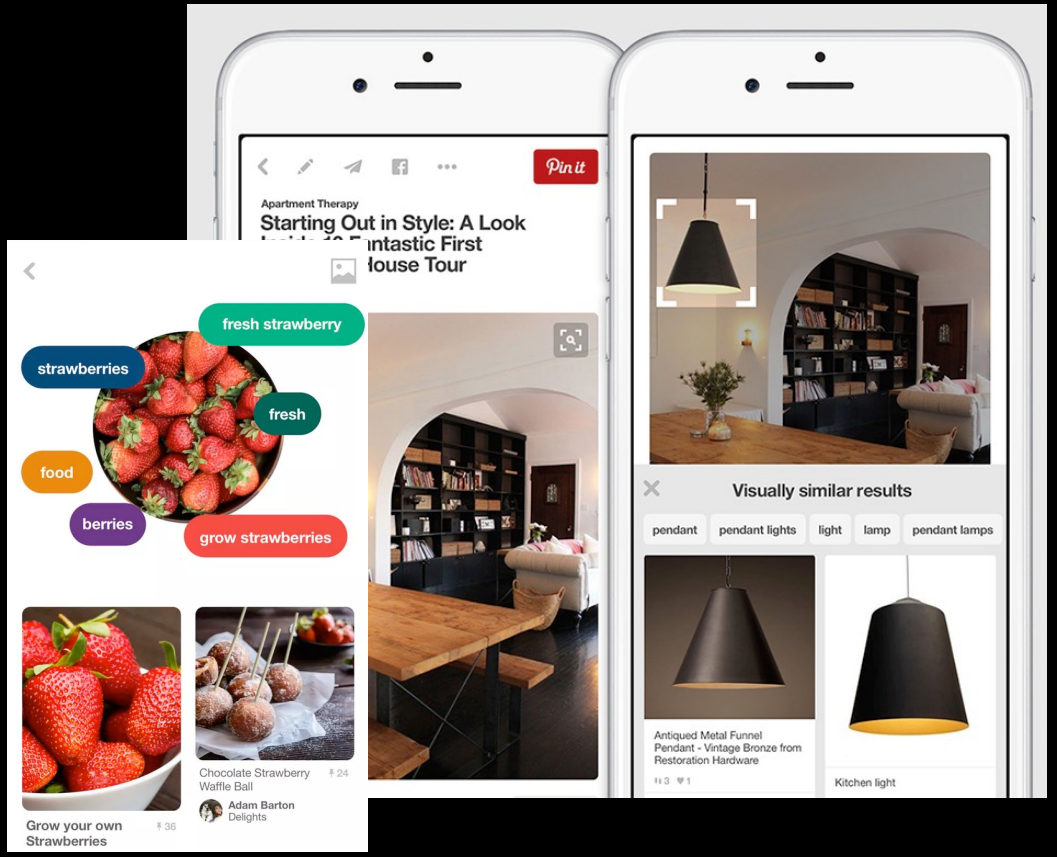
instacart

Online grocery delivery services

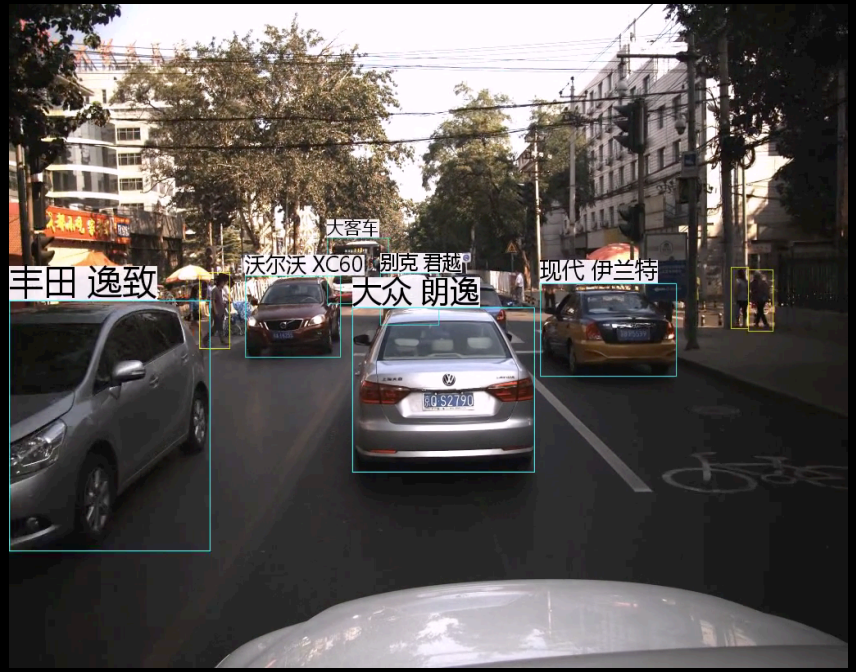


Pinterest

New Pinterest Lens mobile app uses TensorFlow to let users find pins based on the pictures they take

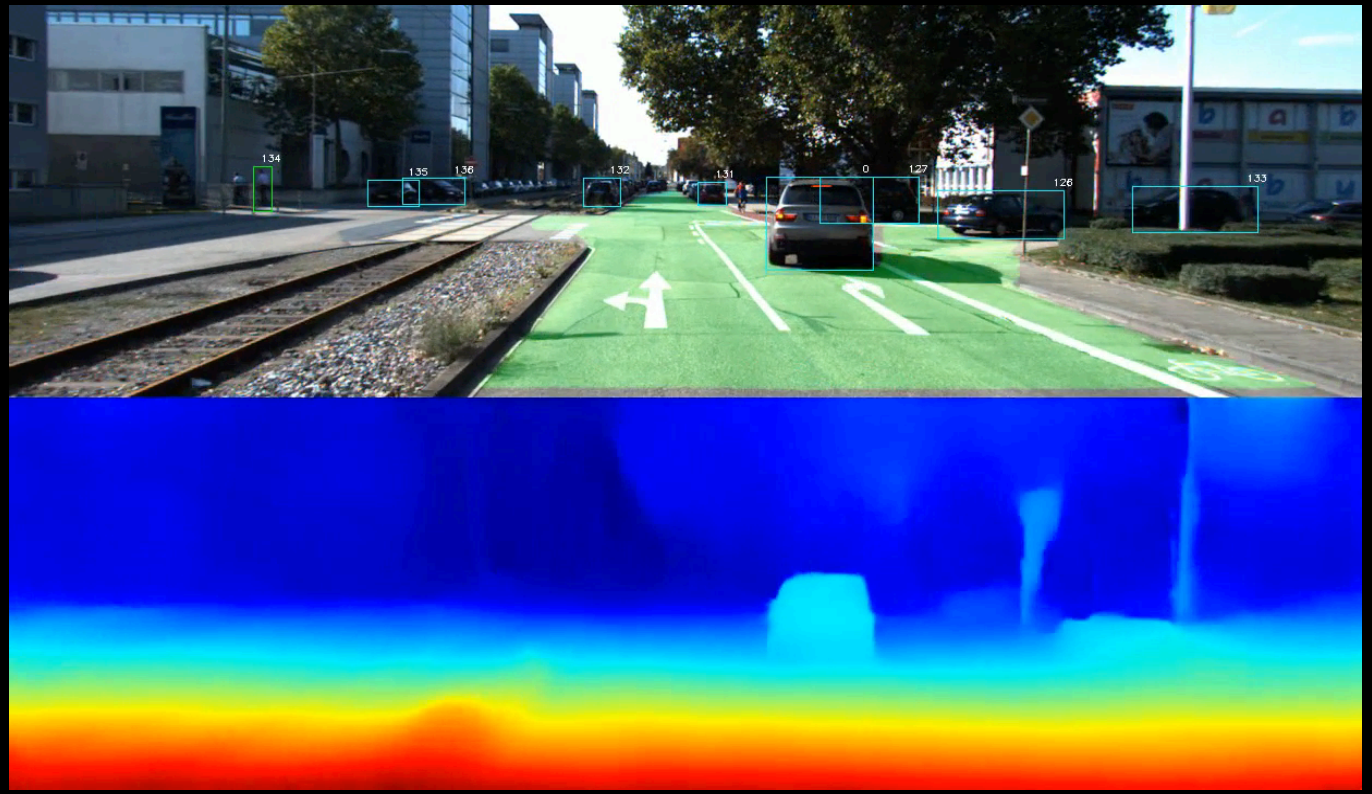


图森 tu Simple

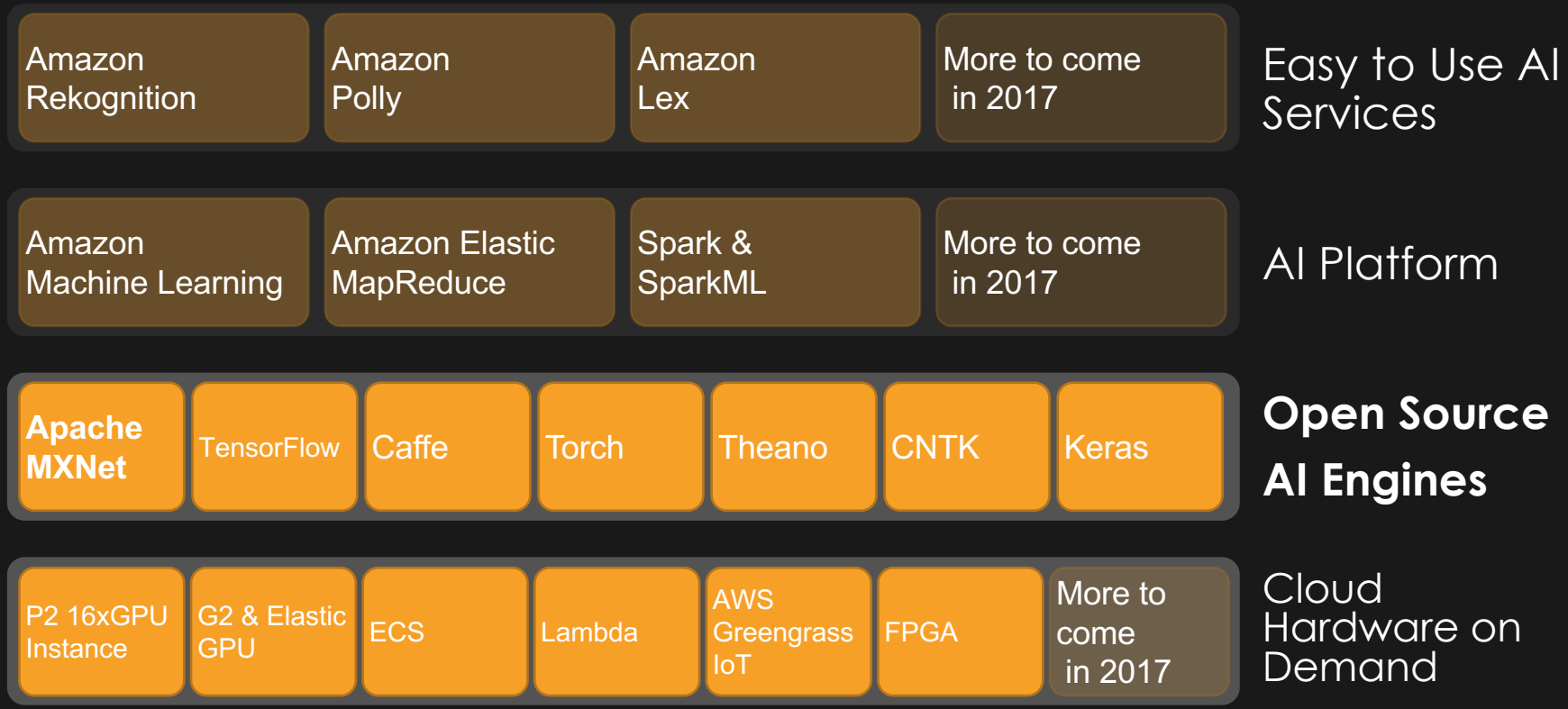


图森 tu Simple

图森 tu Simple



Amazon AI: Democratized Artificial Intelligence



Amazon Rekognition: Object And Scene Detection

Boat	99.3%
Plant	95.1%
Harbor	94.8%
Waterfront	94.8%
Yacht	78.1%
Dock	75.7%
City	72.4%
Architecture	71.8%
Urban	63.9%
Building	62.3%
Marina	60.3%
Plaza	51.1%
Spire	50.8%
Neighborhood	50.7%
Flower	50.6%

Amazon Rekognition: Facial Analysis

Amazon Rekognition: Facial Verification

Similarity: 98%

Amazon Rekognition: Facial Recognition

With Amazon Polly

"The temperature in Chicago, IL is 75˚F"

Amazon Polly

"The temperature in Chicago, Illinois is 75 degrees Fahrenheit"

47 voices

24 languages

Including: Danish, Swedish, Norwegian

Amazon Polly Customers

Amazon Lex

Speech recognition and natural language understanding

Amazon Lex

Automatic speech recognition
Natural language understanding

Weather forecast

Amazon Lex

Speech recognition and natural language understanding

“What’s the weather forecast?”

Amazon Lex

Automatic speech recognition
Natural language understanding

“It will be sunny and 75F”

Weather forecast

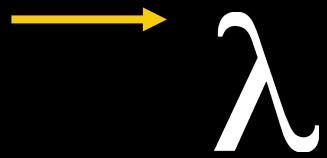
Amazon Lex

Speech recognition and natural language understanding

“What’s the weather forecast?”

Amazon Lex

Automatic speech recognition
Natural language understanding



Weather forecast

“It will be sunny and 75F”

Amazon Polly

“It will be sunny and 75 degrees Fahrenheit”

Amazon Lex Customers

Amazon Connect

Simple to use, cloud-based contact center

Easy to set up and manage

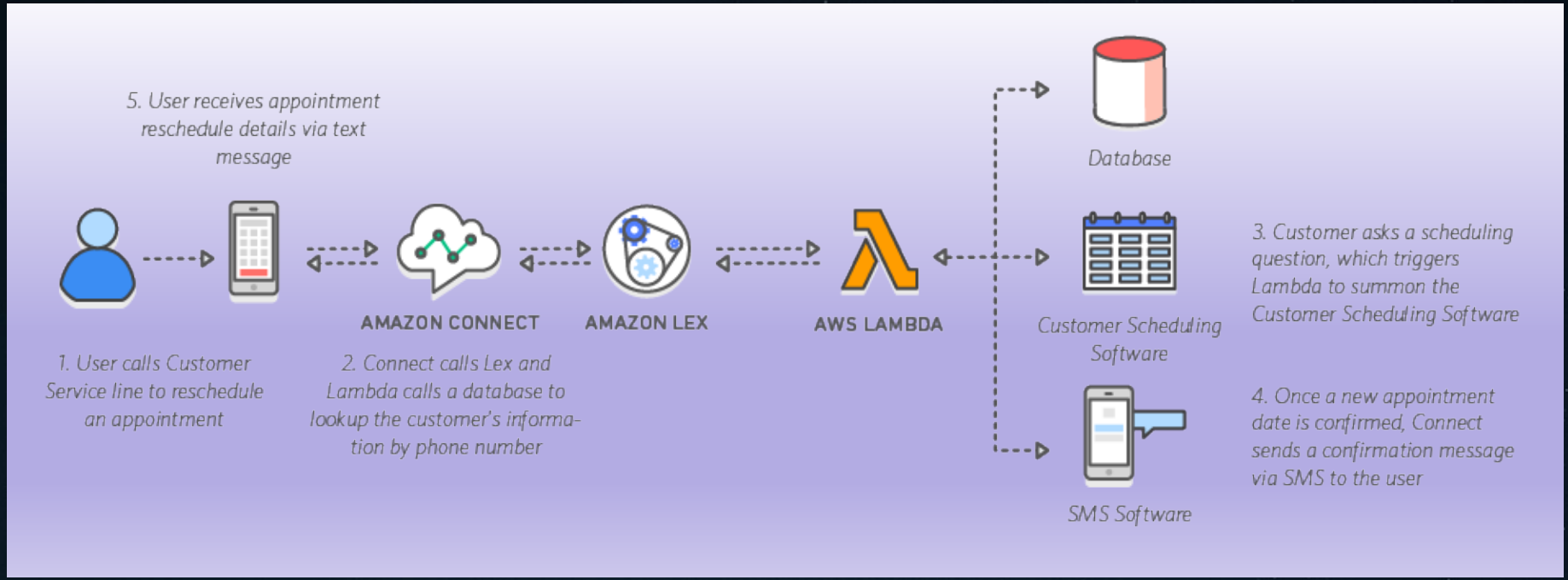
Scalable and elastic

Pay as you go

Reliable

Open platform

Amazon Connect Contact Center Uses Amazon Lex For Natural Conversations



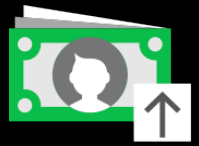
Securing Sensitive Data Is **Job Zero**

Identifying And Protecting Sensitive Data Can Be Challenging

Manual process

Time-consuming

Inaccurate



Expensive

Growing volumes of data


NEW!
AVAILABLE TODAY
Amazon Macie
Automatically discover, classify, and protect sensitive data in AWS using Machine Learning

Introducing Amazon Macie

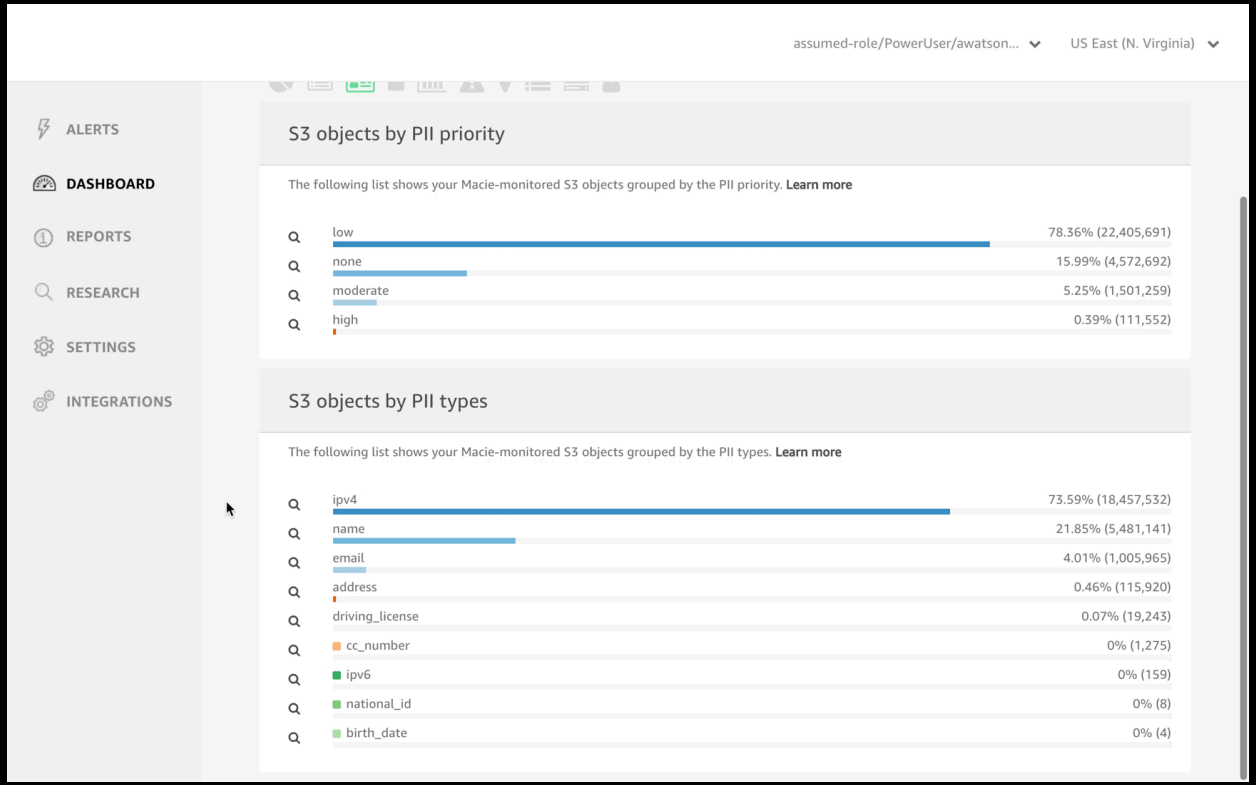
Automatically discover and classify your data

Understand where sensitive data is located and how it is accessed

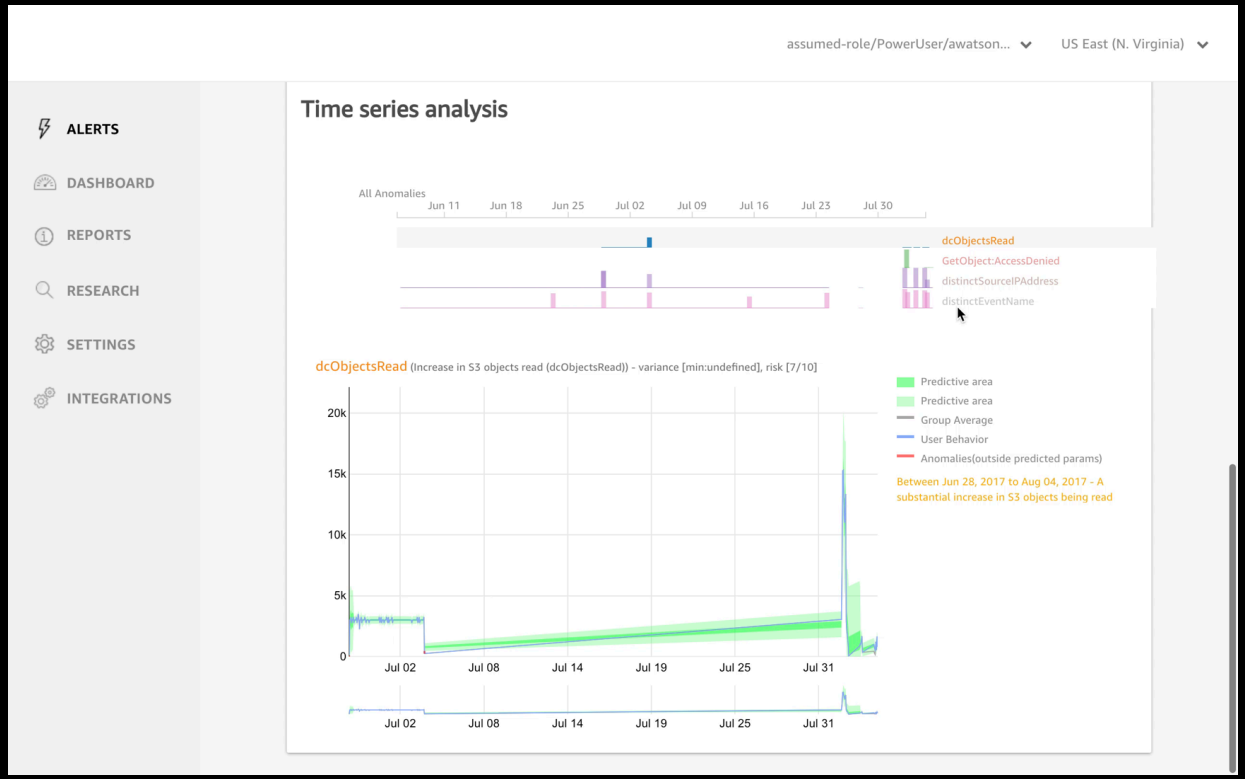
Automatically monitor for anomalies

Alert your security team when anomalies are detected

Gain Visibility Into Globally Shared Content

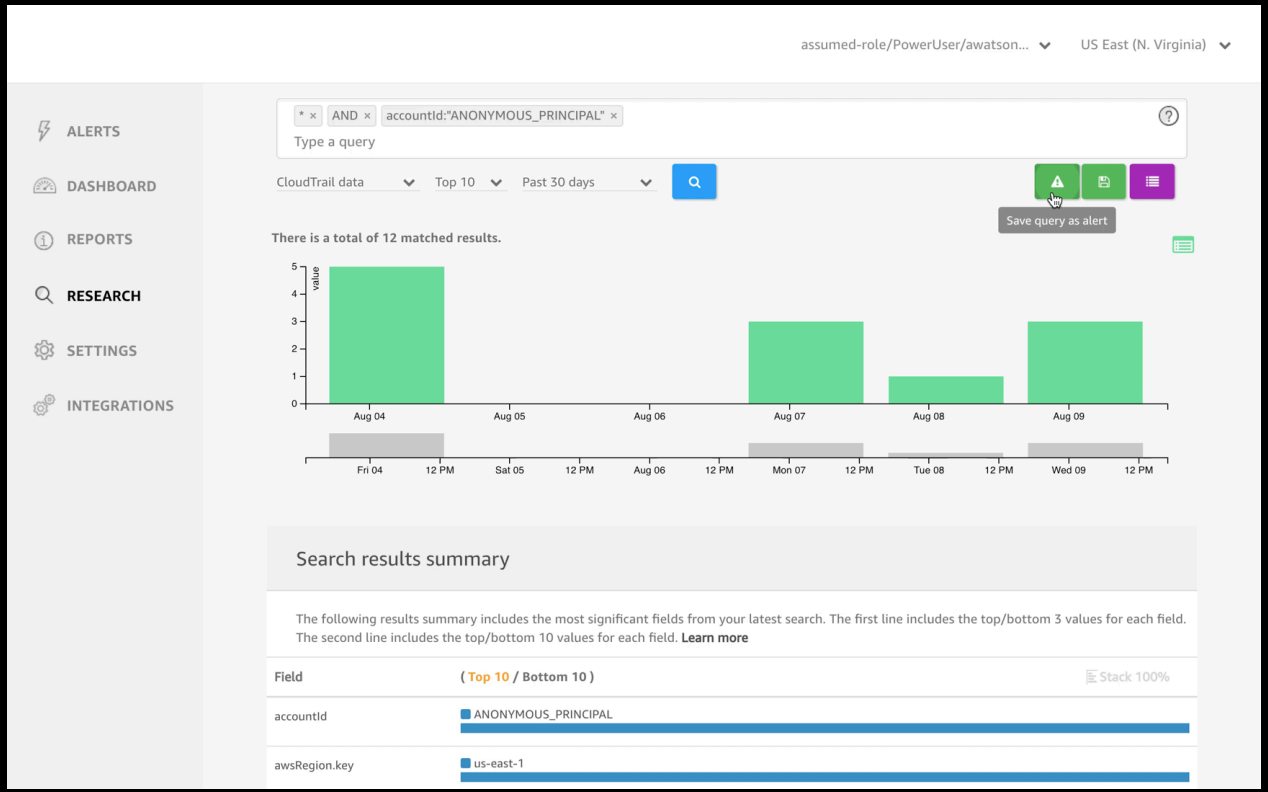


Identify anomalous accesses



Implementing Continuous Compliance

GDPR, PCI, PII

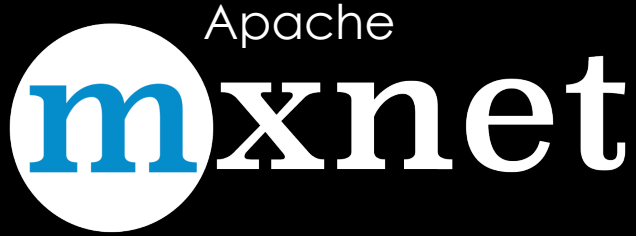


Customers Using Amazon Macie

Continuous insights into cloud infrastructure and practices

Securing PII and alerting to access anomalies

Delivering instant information and detail in the dashboard



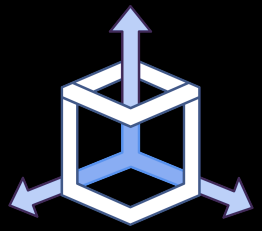
We support, use and tune Keras, Tensorflow and Caffe, but Apache MXNet is the deep learning framework of choice for AWS

Tutorial introduction:
http://gluon.mxnet.io/

Apache MXNet

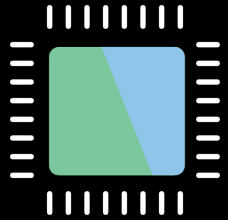
Programmable

Simple syntax, multiple languages



Portable

Highly efficient models for mobile and IoT



High Performance

Near linear scaling across hundreds of GPUs

Why Apache MXNet?

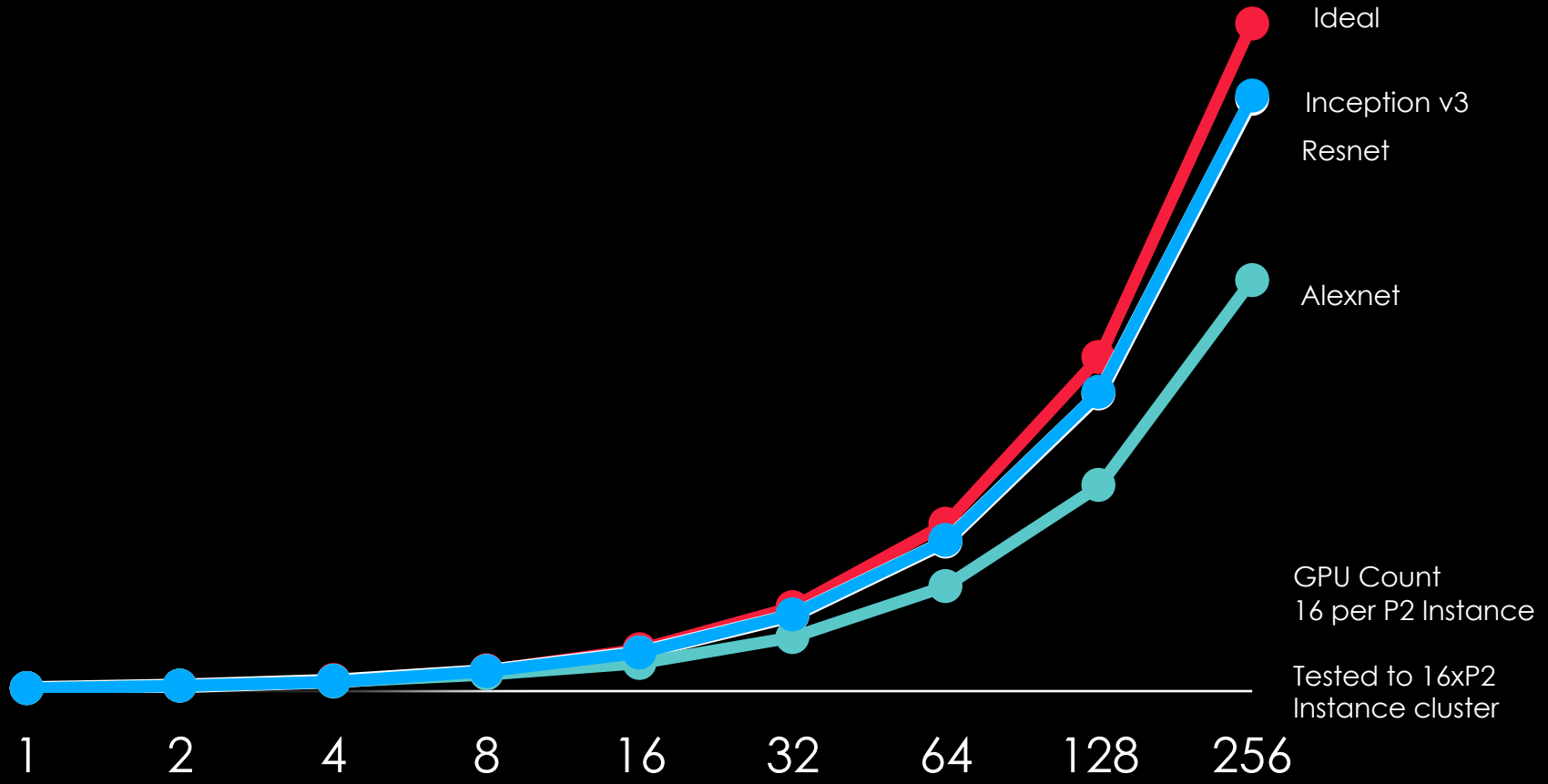
Open Model

Accepted into the Apache Incubator
Broad contributions
e.g. Apple Core ML

Better Performance

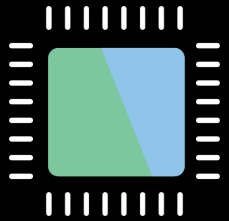
Excellent scalability
Optimized stack for deep learning on AWS

Scaling With Apache MXNet: 88% Efficiency



Apache MXNet Introduction

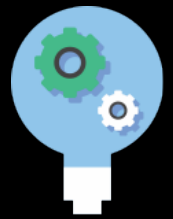
AWS Deep Learning AMIs: One-Click Deep Learning



Kepler, Volta
& Skylake

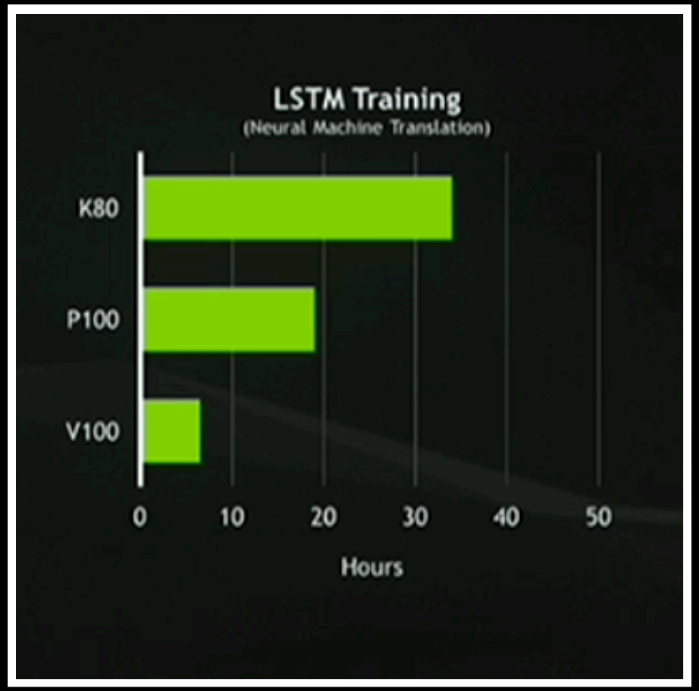
Apache MXNet
TensorFlow Caffe2,
CNTK Keras,
Theano, Torch

Python 3



Jupyter
Notebooks
& Examples

Training Artificial Intelligence With GPUs



NVIDIA Volta

Train in hours, not days

Custom built for artificial intelligence

Core of the next AWS GPU instance family

MXNet is already optimized for Volta

AWS Deep Learning AMI

Up to~40k CUDA cores on P2

Apache MXNet

TensorFlow

Theano

Caffe & Caffe 2

Torch

Keras

Pre-configured CUDA drivers,

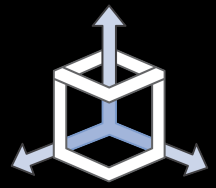
MKL

Anaconda, Python3

Ubuntu or Amazon Linux

+ CloudFormation template

+ Container Image

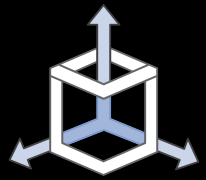


One-Click GPU & CPU
Open Source
Deep Learning
Installed, Tested, Tuned
Bootable Machine Image

Apache MXNet

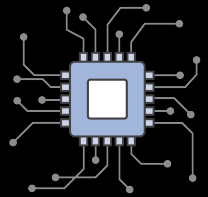
Programmable

Simple syntax,
multiple languages



Portable

Highly efficient
models for mobile
and IoT



High Performance

Near linear scaling
across hundreds of GPUs

Open Governance

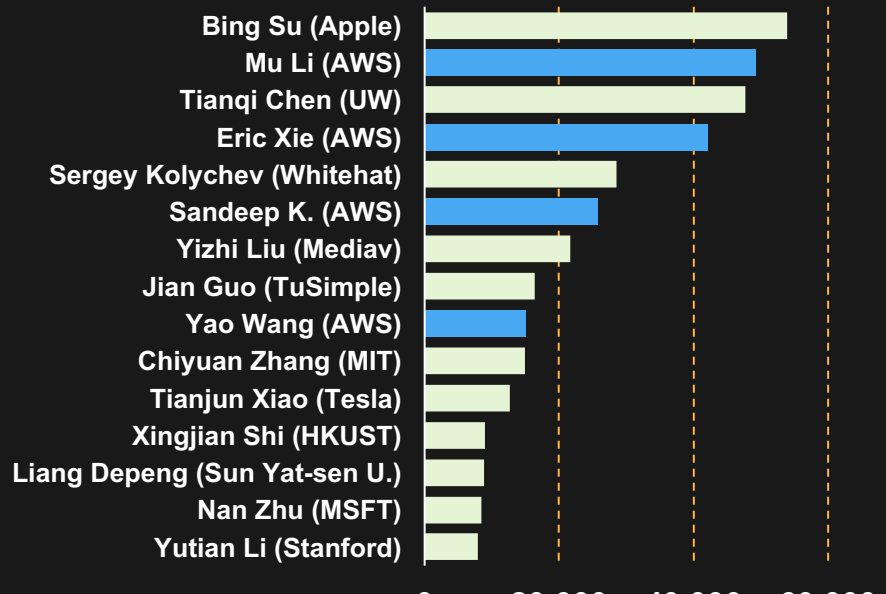
Accepted into the
Apache Incubator

Tuned On AWS

Optimized performance and
scalability on AWS GPUs

Apache MXNet | Collaborations and Community

Diverse Community



*As of 3/30/17
**Amazon @35% of Contributions

Try out Keras models on MXNet instead of Tensorflow

Also see https://medium.com/@julsimon/keras-shoot-out-tensorflow-vs-mxnet-51ae2b30a9c0
Faster, smaller and more accurate…

All it takes is one line in the ~/.keras/keras.json *file.*

```
# ~/.keras/keras.json for Tensorflow
{
    "image_dim_ordering": "tf",
    "epsilon": 1e-07,
    "floatx": "float32",
    "backend": "tensorflow"
}

# ~/.keras/keras.json for MXNet
{
    "image_dim_ordering": "tf",
    "epsilon": 1e-07,
    "floatx": "float32",
    "backend": "mxnet"
}
```

Deep Learning Framework Comparison

	Apache MXNet	**TensorFlow**	**Cognitive Toolkit**
Industry Owner	N/A – Apache Community	Google	Microsoft
Programmability	Imperative and Declarative	Declarative only	Declarative only
Language Support	R, Python, Scala, Julia, C++. Javascript, Go, Matlab, Perl...	Python, C++. Experimental Go and Java	Python, C++, Brainscript.
Code Length | AlexNet (Python)	44 sloc	107 sloc using TF.Slim	214 sloc
Memory Footprint (LSTM)	2.6GB	7.2GB	N/A

Apache MXNet | Amazon Strategy

Integrate with AWS Services

Bring Scalable Deep Learning to EMR, Lambda, ECS and many more..

Foundation for AI Services

Higher Velocity for AI Services, Research and Core AI Development

Leverage the Community

Community brings velocity and innovation with no industry ownership
Safest for long term investment

Apache MXNet API

Apache MXNet | The Basics

- ***NDArray*:** Manipulate multi-dimensional arrays (tensors) in a command line paradigm (imperative).
- ***Symbol*:** Symbolic expression for neural network flows (declarative).
- ***Module***: Intermediate-level and high-level interface for neural network training and inference.
- **Loading Data**: Feeding data into training/inference programs.
- **Mixed Programming**: Training algorithms developed using *NDArrays* in concert with *Symbols*.

https://medium.com/@julsimon/an-introduction-to-the-mxnet-api-part-1-848febdcf8ab

Imperative Programming

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

Easy to tweak
in Python, R, Perl etc.

PROS

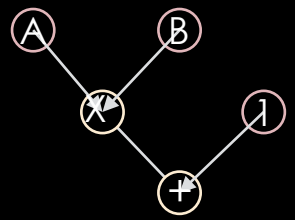
- Straightforward and flexible.
- Take advantage of language native features (loop, condition, debugger).
- E.g. Numpy, Matlab, Torch, …

CONS

- Hard to optimize

Declarative Programming

```
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 can share memory with D because C is deleted later

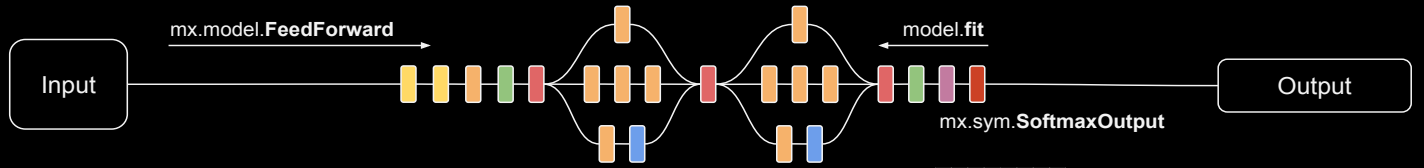
PROS

- More chances for optimization
- Separates flow structure
- E.g. TensorFlow, Theano, Caffe

CONS

- Less flexible, hard to debug

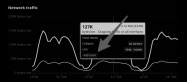
Deep Learning Models




Image


Video

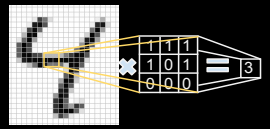

Speech


Events

"People Riding Bikes"

Text

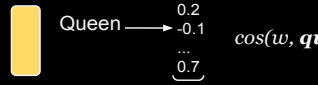
mx.sym.**FullyConnected**(data, num_hidden=128)



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

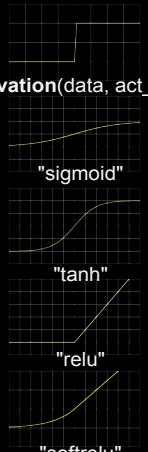
mx.sym.**Pooling**(data, pool_type="max", kernel=(2,2), stride=(2,2)

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")


Image Segmentation


Face Search


Neural Art

"People Riding Bikes"

Image Caption

Bicycle, People, Road, Sport

Image Labels

"Οι άνθρωποι ιππασίας ποδήλατα"

Machine Translation

Do It Yourself - Robot Racing Cars

DIYrobocars Meetup

https://www.meetup.com/diyrobocars/

Follow @DIYrobocars for updates

Common Implementation ~$200

1:16 scale RC Truck + Raspberry Pi3 + Camera + EC2

https://github.com/wroscoe/donkey software

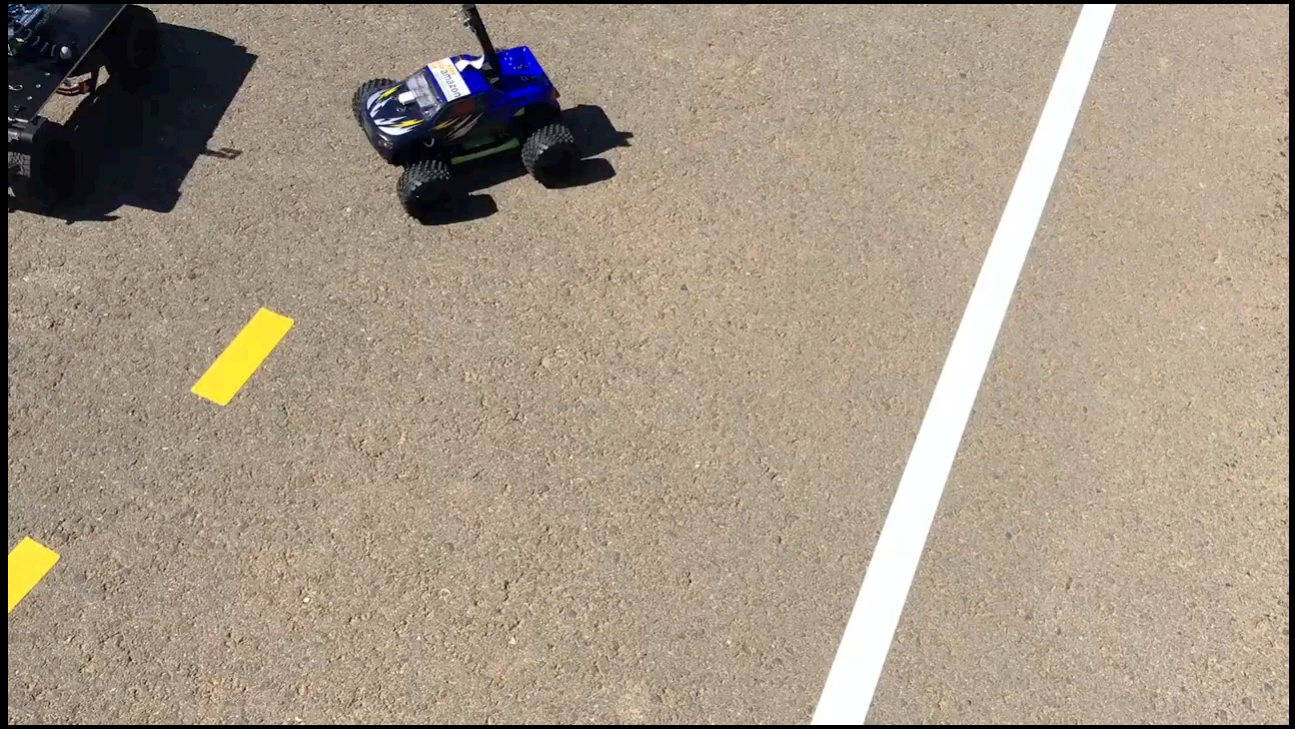
Camera feeds Donkey software with Keras or MXNet model, trained (slowly) on laptop or (quickly) on EC2 GPU instance

Will Roscoe wrote Donkey

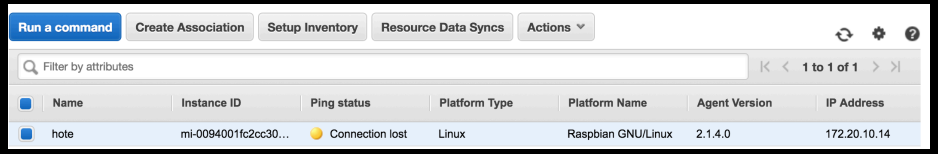
Adrian built Donkey Hoté and instigated…

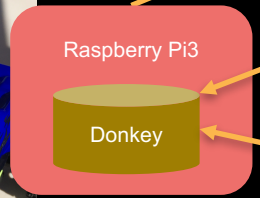
Sunil Mallya added MXNet model and got AWS officially involved

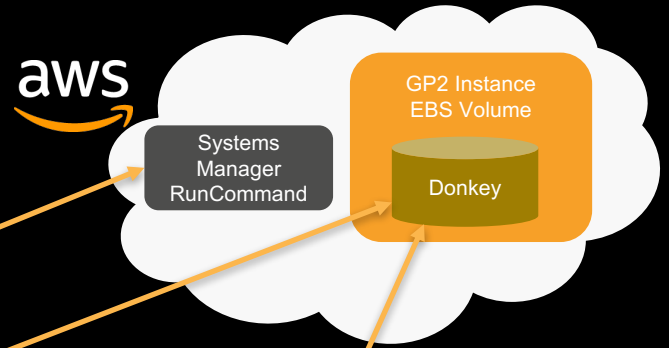
Donkey Hoté's first action…



How The Donkey Software Works







Register IP and
Run commands

Copy training data to EC2 and
Model to Raspberry Pi with rsync

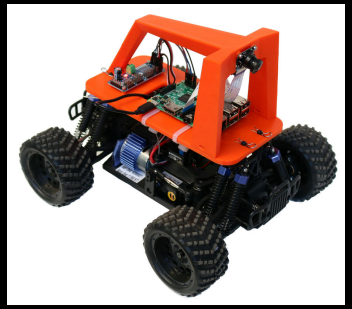
Phone WiFi hotspot and browser
based UI to control donkey

Laptop to run commands
browse training data
and train model on EC2

One of the Keras model options for Donkey

```
def default_categorical():
            from keras.layers import Input, Dense, merge
            from keras.models import Model
            from keras.layers import Convolution2D, MaxPooling2D, Reshape, BatchNormalization
            from keras.layers import Activation, Dropout, Flatten, Dense

            img_in = Input(shape=(120, 160, 3), name='img_in')                      # First layer, input layer, Shape comes from camera.py resolution, RGB
            x = img_in
            x = Convolution2D(24, (5,5), strides=(2,2), activation='relu')(x)       # 24 features, 5px5p kernel (convolution, feature) window, 2wx2h stride, relu
            x = Convolution2D(32, (5,5), strides=(2,2), activation='relu')(x)       # 32 features, 5px5p kernel window, 2wx2h stride, relu activatiion
            x = Convolution2D(64, (5,5), strides=(2,2), activation='relu')(x)       # 64 features, 5px5p kernal window, 2wx2h stride, relu
            x = Convolution2D(64, (3,3), strides=(2,2), activation='relu')(x)       # 64 features, 3px3p kernal window, 2wx2h stride, relu
            x = Convolution2D(64, (3,3), strides=(1,1), activation='relu')(x)       # 64 features, 3px3p kernal window, 1wx1h stride, relu
            x = Flatten(name='flattened')(x)                                       # Flatten to 1D (Fully connected)
            x = Dense(100, activation='relu')(x)                                   # Classify the data into 100 features, make all negatives 0
            x = Dropout(.1)(x)                                                     # Randomly drop out (turn off) 10% of the neurons (Prevent overfitting)
            x = Dense(50, activation='relu')(x)                                    # Classify the data into 50 features, make all negatives 0
            x = Dropout(.1)(x)                                                     # Randomly drop out 10% of the neurons (Prevent overfitting)
            # categorical output of the angle
            # Connect every input with every output, output 15 hidden units. Use Softmax to get percentage. 15 categories, find best one based off percentage 0.0-1.0
            angle_out = Dense(15, activation='softmax', name='angle_out')(x)
            # continous output of throttle
            throttle_out = Dense(1, activation='relu', name='throttle_out')(x)      # Reduce to 1 number, Positive number only
            model = Model(inputs=[img_in], outputs=[angle_out, throttle_out])
            model.compile(optimizer='rmsprop',
                          loss={'angle_out': 'categorical_crossentropy',
                                'throttle_out': 'mean_absolute_error'},
                          loss_weights={'angle_out': 0.9, 'throttle_out': .001})
            return model
```



Robocar Rally 2017

at AWS re:Invent

SUNDAY, NOV. 26 | 6:00PM – 10:00PM
MONDAY, NOV. 27 | 7:00AM – 12:00AM

Registration for Robocar Rally 2017 will launch on Oct. 19 as part of the Reserved Seating launch
https://reinvent.awsevents.com/learn/robocar-rally/

https://aws.amazon.com/blogs/ai/build-an-autonomous-vehicle-on-aws-and-race-it-at-the-reinvent-robocar-rally/

Additional Resources

MXNet Resources

- MXNet Blog Post | AWS Endorsement
  - http://www.allthingsdistributed.com/2016/11/mxnet-default-framework-deep-learning-aws.html
- Read up on MXNet and Learn More:
  - See gluon.mxnet.io https://github.com/dmlc/mxnet/
- Re:Invent MXNet Recommender Systems Talk by Leo Dirac
  - https://www.portal.reinvent.awsevents.com/connect/sessionDetail.ww?SESSION_ID=8591

AWS Resources: follow Julien Simon @julsimon, Sunil Mallya @sunilmallya

- Deep Learning AMI
  - https://aws.amazon.com/marketplace/pp/B01M0AXXQB | Amazon Linux
  - https://aws.amazon.com/marketplace/pp/B06VSPXKDX | Ubuntu
- CloudFormation Template Instructions
  - https://github.com/dmlc/mxnet/tree/master/tools/cfn
- Deep Learning Benchmark
  - https://github.com/awslabs/deeplearning-benchmark
- MXNet on Lambda
  - https://github.com/awslabs/mxnet-lambda
- MXNet on ECS/Docker
  - https://github.com/awslabs/ecs-deep-learning-workshop

THANK YOU!

@adrianco