AWS SUMMIT

Machine Learning Inference at the Edge

Julien Simon

Principal Technical Evangelist, AI and Machine Learning

@julsimon

Agenda

- Deep Learning at the Edge?
- Apache MXNet
- Predicting in the Cloud or at the Edge?
- New services
    - AWS Greengrass ML
    - AWS DeepLens
- Resources

Deep Learning at the Edge?

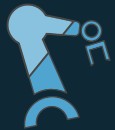
Use Cases


Self-driving cars


Smart Agriculture


Predictive maintenance


Video surveillance


Robotics

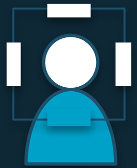
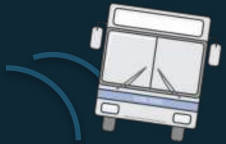


Image recognition


Voice/sound recognition

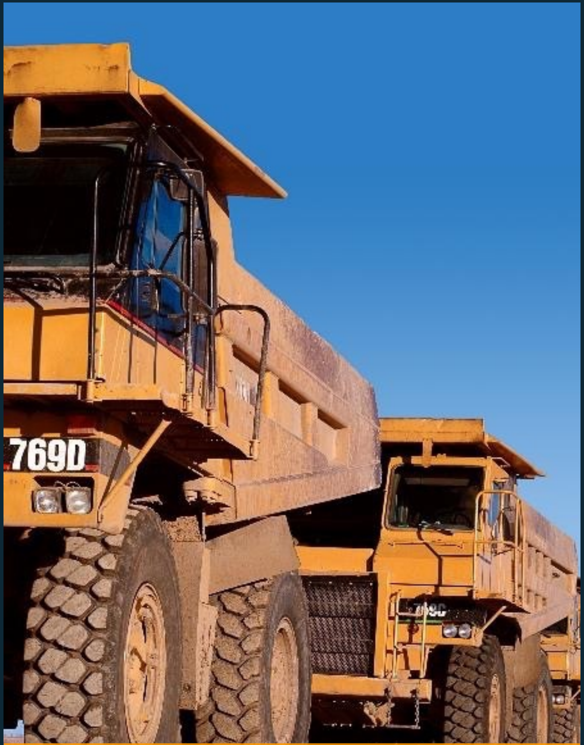

Collision avoidance


Anomaly detection

More

Most machine data never reaches the cloud

Medical equipment

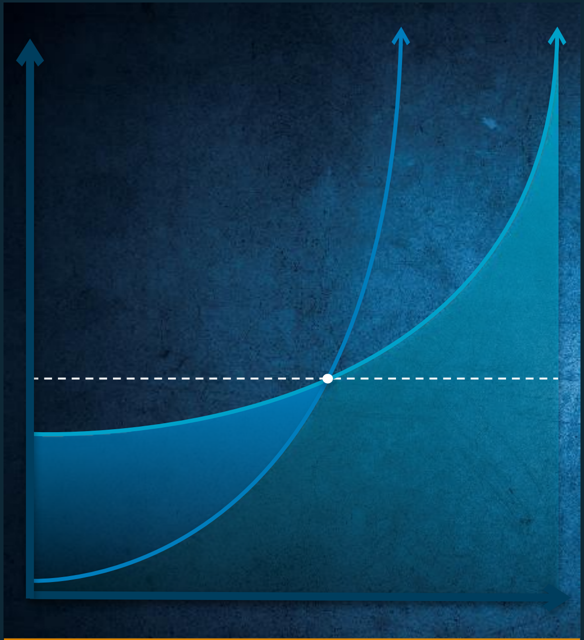


Industrial machinery

Extreme environments

Why this problem isn't going away

Law of physics



Law of economics

Law of the land

Deep Learning challenges at the Edge

- **Resource-constrained devices**
  - CPU, memory, storage, power consumption.
- **Network connectivity**
  - Availability, cost, bandwidth, latency.
  - On-device prediction may be the only option.
- **Deployment**
  - Updating code and models on a fleet of devices is not easy.

aws

Deep Learning wishlist at the Edge

- Rely on cloud-based services for seamless training and deployment.
- Have the option to use cloud-based prediction.
- Be able to run device-based prediction with good performance.
- Support different technical environments (CPUs, languages).

Apache MXNet

Apache MXNet: Open Source library for Deep Learning

Programmable

Simple syntax, multiple languages

Portable

Highly efficient models for mobile and IoT

High Performance

Near linear scaling across hundreds of GPUs

Most Open

Accepted into the Apache Incubator

Best On AWS

Optimized for Deep Learning on AWS

Apache MXNet for IoT

1. Flexible experimentation in the Cloud.
2. Scalable training in the Cloud.
3. Good prediction performance at the Edge.
4. Prediction in the Cloud or at the Edge.

1 - Flexible experimentation in the Cloud

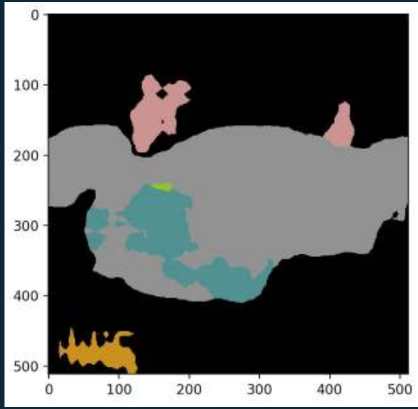
- API for Python, R, Perl, Matlab, Scala, C++.

- Gluon
  - Imperative programming aka 'define-by-run'.
  - Inspect, debug and modify models during training.

- Extensive model zoo
  - Pre-trained computer vision models
  - DenseNet, SqueezeNet for resource-constrained devices.

Gluon CV: classification, detection, segmentation

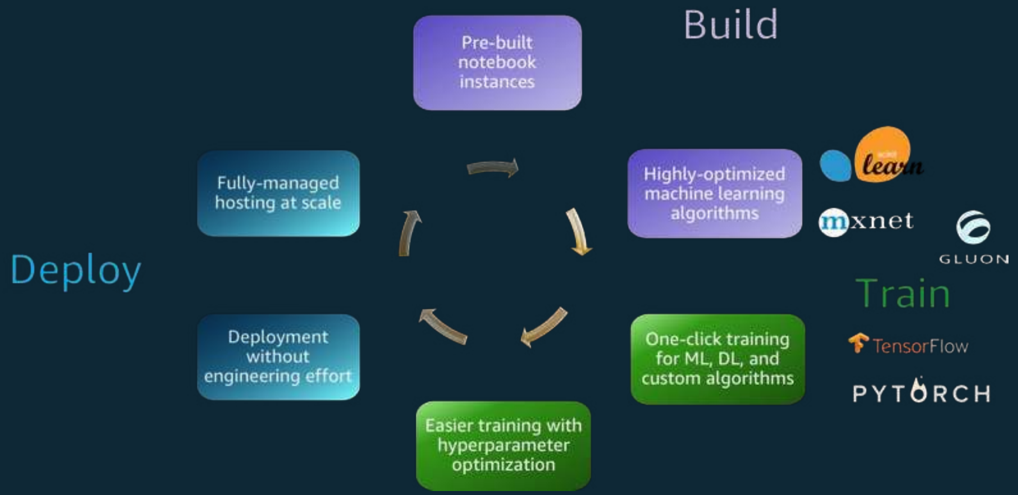
[electric_guitar],
with probability 0.671



https://github.com/dmlc/gluon-cv

2 - Scalable training in the Cloud

Amazon SageMaker



AWS Deep Learning AMI

TensorFlow
K
Microsoft Cognitive Toolkit
mxnet
GLUON
Chainer
Caffe2
PYTORCH

Amazon EC2

c5



p3

aws

3 - Good prediction performance at the Edge

- MXNet is written in C++.
- Gluon networks can be 'hybridized' for additional speed.
- Two libraries boost performance on CPU-only devices
  - Fast implementation of math primitives
  - Hardware-specific instructions, e.g. Intel AVX or ARM NEON
  - Intel Math Kernel Library https://software.intel.com/en-us/mkl
  - NNPACK https://github.com/Maratyszcza/NNPACK

- Mixed precision training
  - Use float16 instead of float32 for weights and activations
  - Almost 2x reduction in model size, no loss of accuracy, faster inference
  - https://devblogs.nvidia.com/parallelforall/mixed-precision-training-deep-neural-networks/

4 - Predicting in the Cloud or at the Edge

- Cloud-based: invoke a Lambda function with AWS IoT.
- Cloud-based: invoke a SageMaker endpoint with HTTP.
- Device-based: bring your own code and model.
- Device-based: deploy your code and model with AWS Greengrass.

Invoking a Lambda function with AWS IoT

- Train a model in SageMaker (or bring your own).
- Host it in S3 (or embed it in a Lambda function).
- Write a Lambda function performing prediction.
- Invoke it through AWS IoT.

Best when
Devices can support neither HTTP nor local inference (e.g. Arduino).
Costs must be kept as low as possible.

Requirements
Network is available and reliable (MQTT is less demanding than HTTP).
Devices are provisioned in AWS IoT (certificate, keys).

https://aws.amazon.com/blogs/compute/seamlessly-scale-predictions-with-aws-lambda-and-mxnet/

aws

Invoking a SageMaker endpoint with HTTP

- Train a model in SageMaker (or bring your own).
- Deploy it to a prediction endpoint.
- Invoke the HTTP endpoint from your devices.

Best when
Devices are not powerful enough for local inference.
Models can't be easily deployed to devices.
Additional cloud-based data is required for prediction.
Prediction activity must be centralized.

Requirements
Network is available and reliable.
Devices support HTTP.

Bring your own code and model

- Train a model in SageMaker (or bring your own).
- Bring your own application code.
- Provision devices at manufacturing time (or use your own update mechanism).

Best when
You don’t want to or can’t rely on cloud services (no network connectivity?)

Requirements
Devices are powerful enough for local inference.
Models don’t need to be updated, if ever.
DIY!

aws

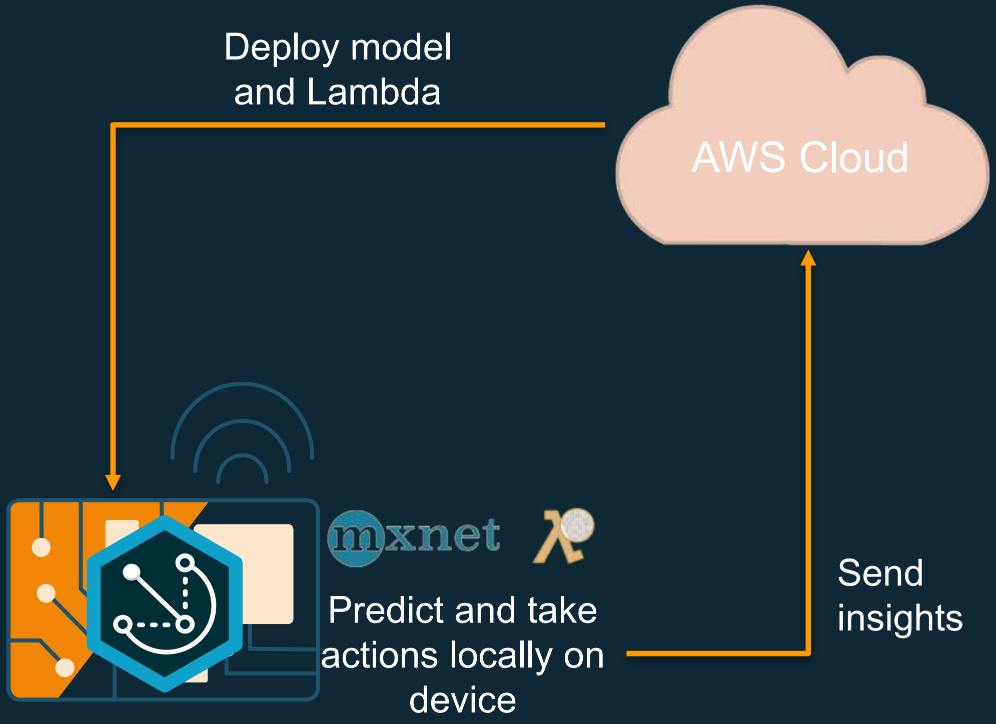
Deploy your code and model with AWS Greengrass

- Train a model in SageMaker (or bring your own).
- Write a Lambda function performing prediction.
- Add both as resources in your Greengrass group.
- Let Greengrass handle deployment and updates.

Best when
You want the same programming model in the Cloud and at the Edge.
Code and models need to be updated, even if network connectivity is infrequent or unreliable.
One device in the group should be able to perform prediction on behalf on other devices.

Requirements
Devices are powerful enough to run Greengrass (XXX HW requirements)
Devices are provisioned in AWS IoT (certificate, keys).

ML Inference using AWS Greengrass



AWS Greengrass ML

- Deployments
- Subscriptions
- Cores
- Devices
- Lambdas
- **Resources**
- Settings

Local resources

Add

Name	Resource Type	Status	Local path	
videoCoreInterface	Device	Affiliated	/dev/vchiq	•••
videoCoreShareMemory	Device	Affiliated	/dev/vcsm	•••

Machine learning resources

Add

Name	Resource Type	Status	Local path	
squeezenet_model	Model	Affiliated	https://jsimon-greengras…	•••

AWS DeepLens

AWS DeepLens

The world’s first Deep Learning-enabled video camera for developers

HD video camera with on-board compute optimized for Deep Learning

Integrates with Amazon SageMaker and AWS Lambda

10 MIN

From unboxing to prediction in <10 minutes

Tutorials, examples, demos, and pre-built models

Get Started with Deep Learning

It takes less than 10 minutes with AWS DeepLens


FACIAL DETECTION
Detect faces of people.


ACTIVITY RECOGNITION
Recognize more than 30 kinds of actions such as brushing teeth, applying lipstick, and playing guitar.


HOT DOG NOT HOT DOG
A hot dog or not a hot dog, that is the question.


OBJECT DETECTION
Accurately detect and recognize objects.


ARTISTIC STYLE TRANSFER
Transfer the style from one image, such as a painting, to an entire video sequence captured by your AWS DeepLens in real-time.


CAT AND DOG RECOGNITION
Detect whether it's a cat or dog.

Build custom Deep Learning models in the cloud using Amazon SageMaker, or use the collection of pretrained models included with AWS DeepLens

aws

AWS DeepLens

Object-detection

Delete | Deploy to device

Project

Copy | Edit

Name	Description	Version
Object-detection	Detect 20 popular objects	-

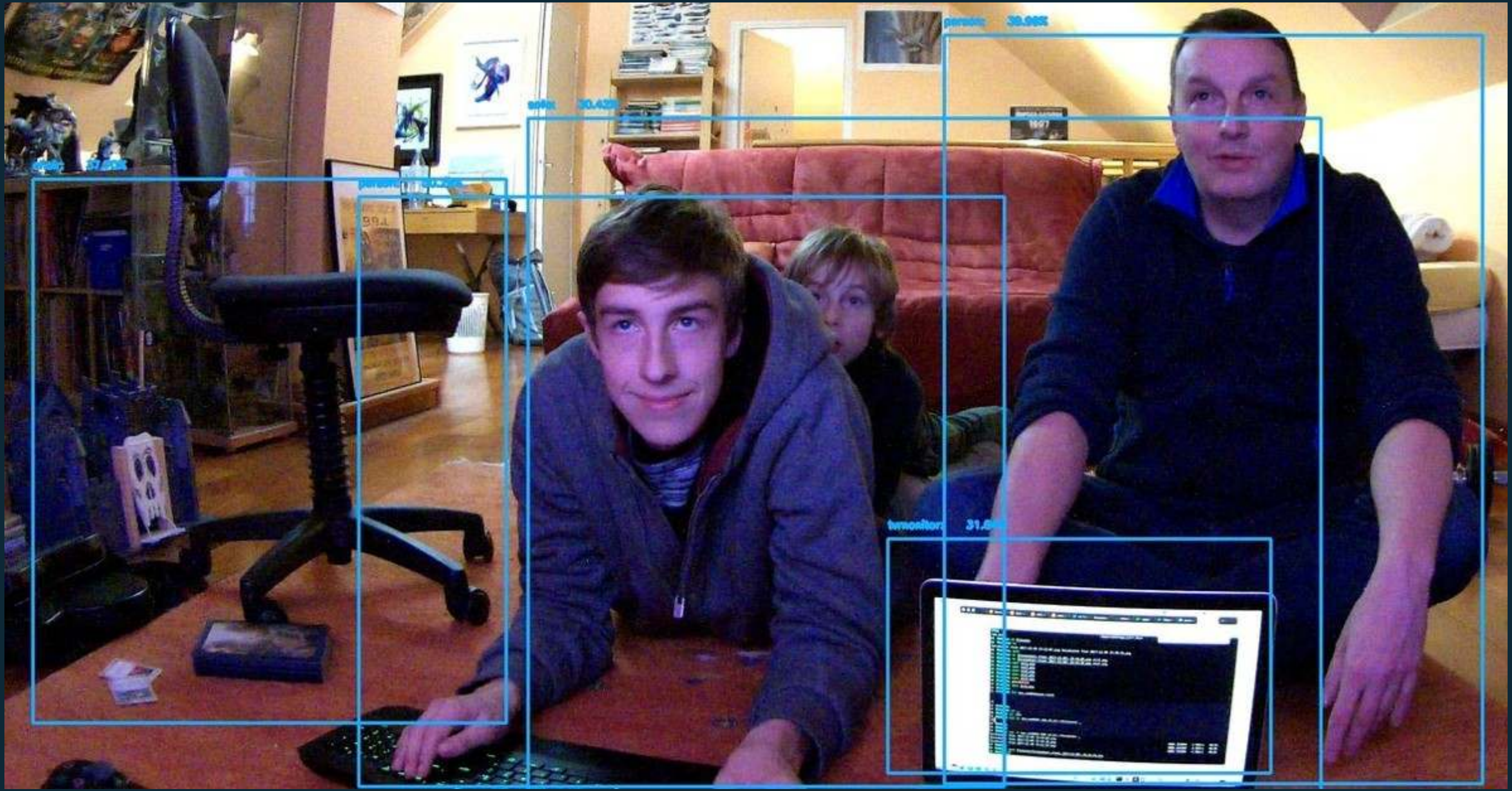
ARN

arn:aws:deeplens:us-east-1[redacted]:project/Object-detection

Project content

Type	Name
Function	arn:aws:lambda:us-east-1:[redacted]:function:deeplens-object-detection:1
Model	deeplens-object-detection

Object detection with AWS DeepLens



Resources

Resources

https://aws.amazon.com/machine-learning/amis/

https://mxnet.incubator.apache.org

http://gluon.mxnet.io

https://aws.amazon.com/sagemaker (free tier available)

https://github.com/awslabs/amazon-sagemaker-examples

https://aws.amazon.com/greengrass (free tier available)

https://aws.amazon.com/greengrass/ml/

https://aws.amazon.com/deeplens

https://medium.com/@julsimon

https://youtube.com/juliensimonfr

Thank you!

Julien Simon, Principal AI/ML Evangelist, Amazon Web Services

@julsimon