AWS SUMMIT

Machine Learning Inference at the Edge

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

@julsimon

Simone Mangiante, Research and Standards Specialist, Vodafone

Simone.Mangiante@Vodafone.com

Agenda

- Deep Learning at the Edge?
- Apache MXNet
- Predicting in the Cloud or at the Edge?
- Case study: Driver Monitoring by Vodafone
- New services: AWS Greengrass ML and 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

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.

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.

2 - Scalable training in the Cloud

Amazon SageMaker

Build

Pre-built notebook instances

Highly-optimized machine learning algorithms

scikit learn

mxnet

GLUON

Train

One-click training for ML, DL, and custom algorithms

TensorFlow

PYTORCH

Easier training with hyperparameter optimization

Deploy

Deployment without engineering effort

Fully-managed hosting at scale

AWS Deep Learning AMI

TensorFlow

K

Microsoft Cognitive Toolkit

mxnet

GLUON

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/

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!

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).

Distributing ML at the edge

Dr Simone Mangiante

simone.mangiante@vodafone.com

27 March 2018

ML at the edge: challenges and benefits

Challenges from use cases, benefits from deployment at the telco network edge

Distributed, low latency machine learning

Increasing amount of in-vehicle data (40+ TB/h)

Desire for more mobile-driven services in cars harnessing advanced network functions

Needs to be developer friendly

Reduce in-vehicle bill-of-material by offloading computing resources

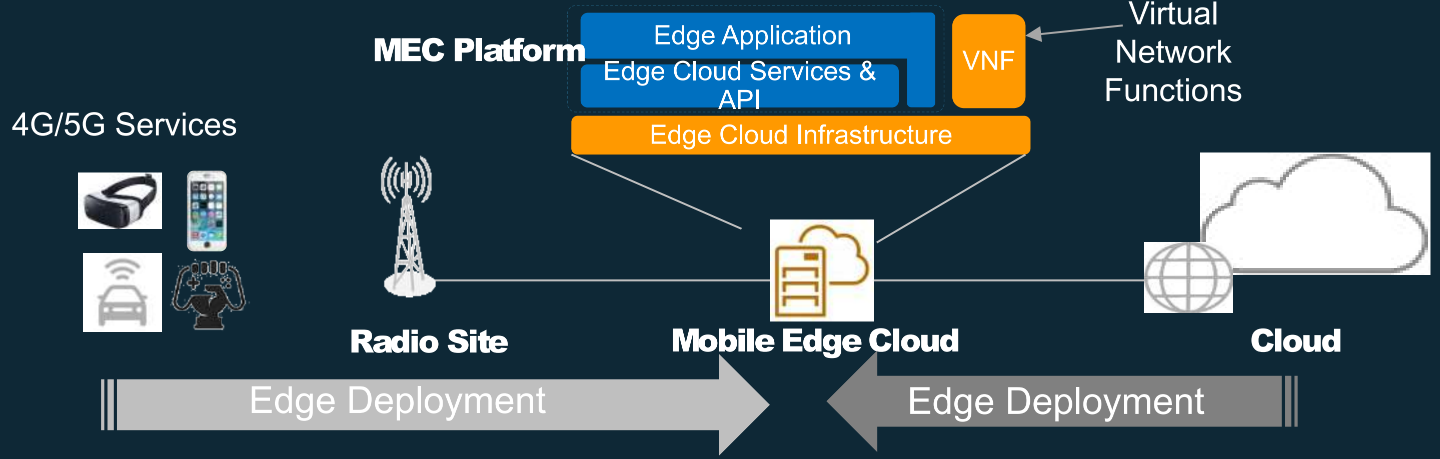
Optimise service cost with shared resources

Seamlessly upgrade in a cloud-based environment

Evolve at the pace of technology leveraging the latest advances

Real-time alerts that enhance safety

What is Mobile/Multi-access Edge Computing?



MEC is a network architectural concept that enables **cloud-computing** capabilities at the **edge of the network**

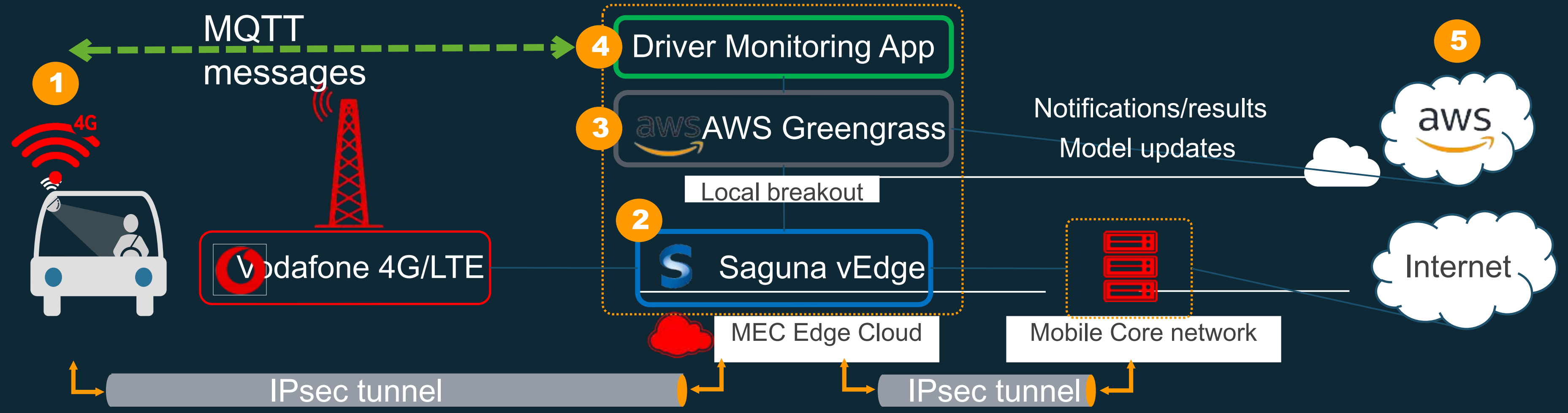
MEC offers **applications and content providers** cloud-computing capabilities at the edge of the network

- MEC software (i.e. MEC Platform) runs as a VNF on a cloud-based edge infrastructure
- 3rd party applications can be deployed on MEC platforms
- MEC platform can expose network APIs to applications

The rationale for edge computing goes beyond latency

Latency argument is sometimes ‘overhyped’

MEC driver monitoring proof of concept



1
- Camera device (Raspberry Pi)
- Connected via cellular radio
- Video streamed to Driver Monitoring App

2
- Filters traffic based on traffic rules: e.g. pass-through to Internet, mirror, redirect to local application
- Can 'chain' applications

3
- Camera device is in Greengrass Group
- Runs in a VM
- It's a 'MEC application'
- Receives traffic from radio access as per configured traffic rules

4
- Edge app
- Business logic
- Includes neural network

5
- Training

Driver monitoring application

In-Vehicle

Stream video

Receive alerts

AWS IoT SDK

MEC Edge Cloud

Receive video
Send alerts

Perform inference using ML model

AWS Greengrass core

AWS Cloud

Create and train the driver monitoring model

AWS SageMaker

Developed by TensorIoT

Convolutional Neural Network
- Two architectures explored: Inception, MobileNet
- MobileNet chosen for its speed advantage and ability to run on a light platform (as for a PoC)

Business logic
- Classifies a series of sequential video frames
- Then makes decision on whether “distracted driving” detected
- For PoC, sends message to Raspberry Pi for local web server and displaying via browser on monitor

Training data
- Source: Kaggle.com
- Dataset: State Farm Distracted Driver Detection

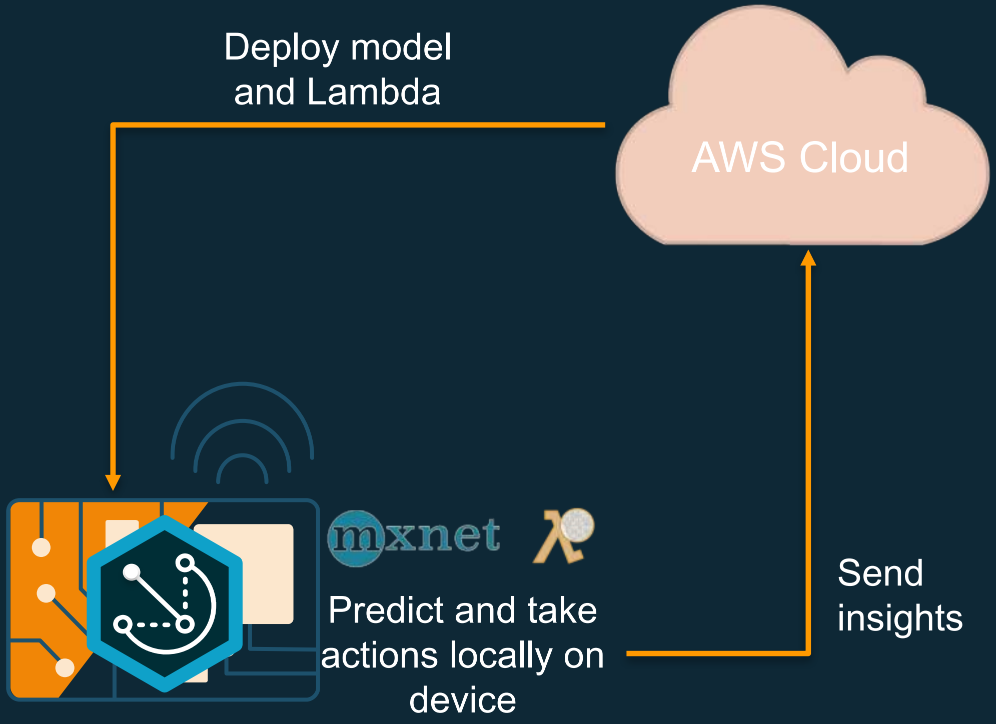
aws

The future is exciting.

Ready?

ML Inference using AWS Greengrass

PREVIEW AVAILABLE



AWS Greengrass ML

PREVIEW AVAILABLE

- 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

AWS DeepLens

AWS DeepLens

World's first Deep Learning enabled video camera for developers

- A new way to learn
- Custom built for Deep Learning
- Broad Framework Support
- Deploy models from Amazon SageMaker
- Integrated with AWS
- Fully programmable with AWS Lambda

aws

AWS DeepLens

Object-detection

Delete | Deploy to device

Project

Copy | Edit

Name	Description	Version
Object-detection	Detect 20 popular objects	-

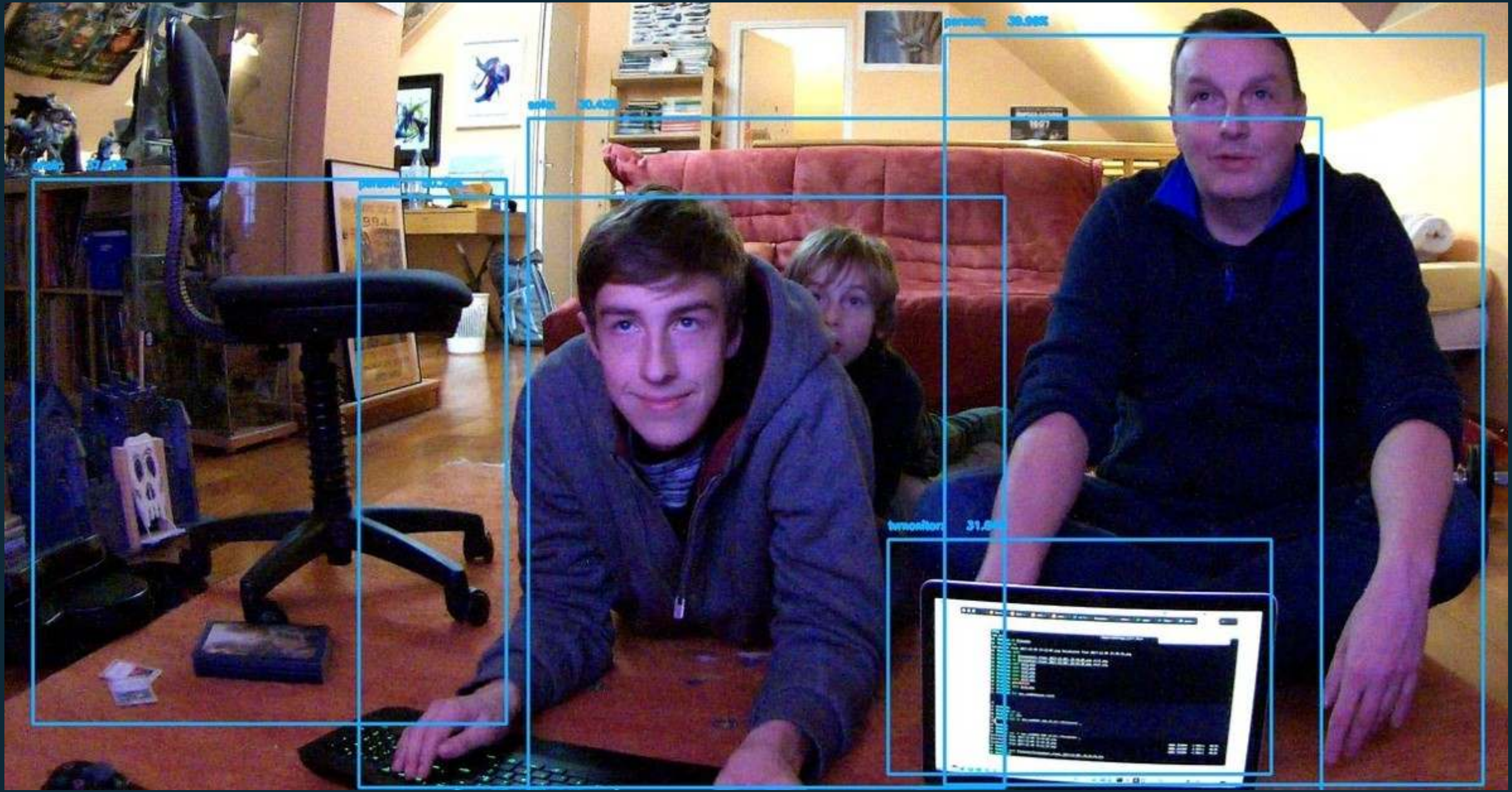
ARN

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://mxnet.incubator.apache.org

http://gluon.mxnet.io

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

An overview of Amazon SageMaker: https://www.youtube.com/watch?v=ym7NEYEx9x4

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

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

https://aws.amazon.com/deeplens

https://medium.com/@julsimon

Thank you!

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

@julsimon

Simone Mangiante, Research and Standards Specialist, Vodafone

Simone.Mangiante@Vodafone.com