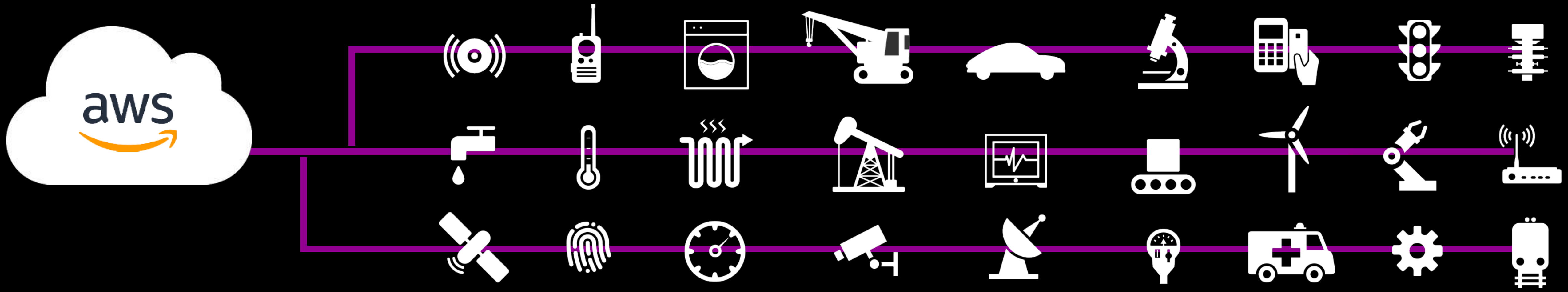
Get Started with Machine Learning and Computer Vision Using AWS DeepLens

Julien Simon
Global Evangelist, AI & Machine Learning
@julsimon

aws SUMMIT

Computing is increasingly available at the edge

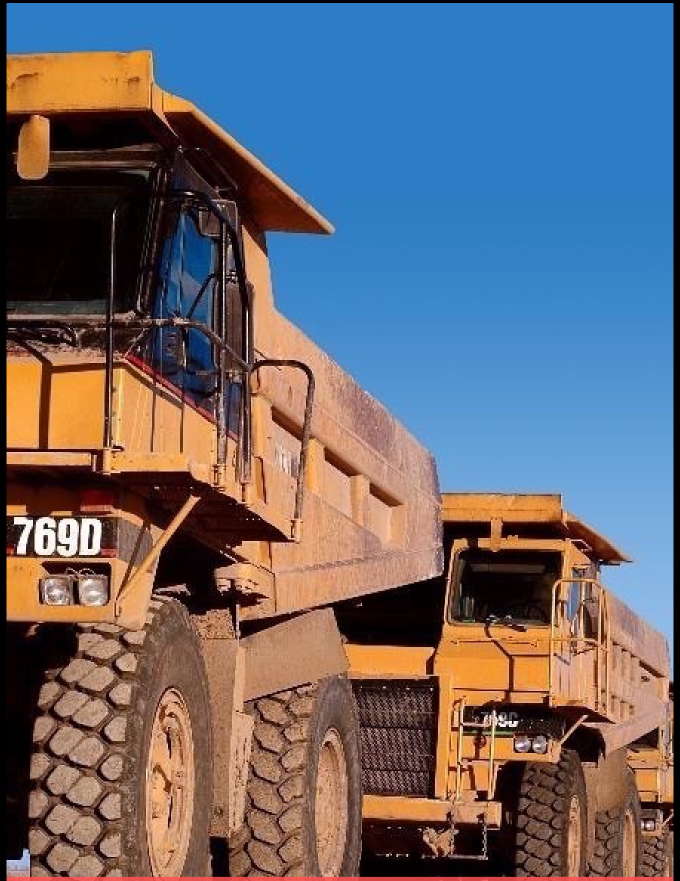


Machine Learning predictions at the edge would make devices smarter.

Could we simply invoke cloud-based models?

Most machine data can't reach the Cloud

Medical equipment

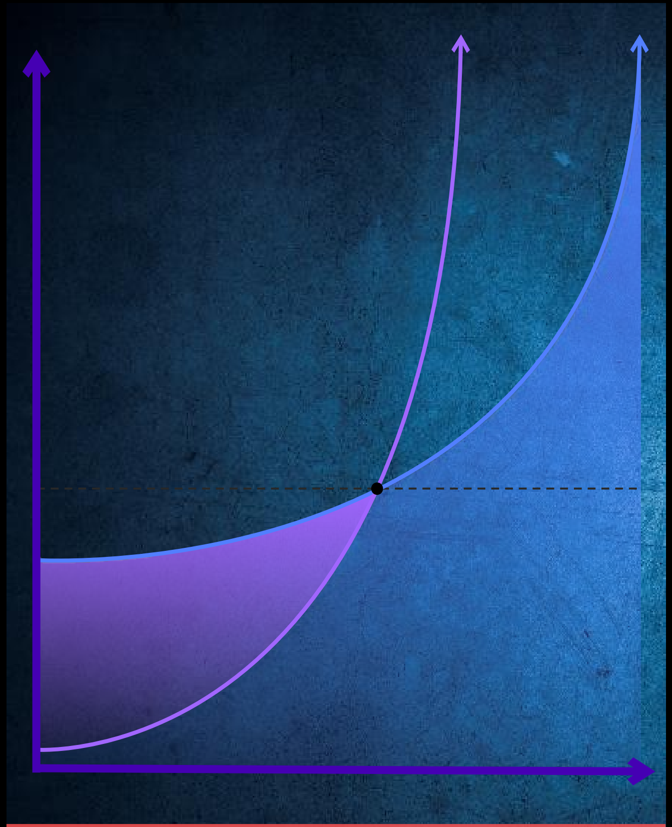


Industrial machinery

Extreme environments

Why this problem isn’t going away

Law of physics



Law of economics

Law of the land

To-do list

❑ Build a data set

❑ Experiment with state-of-the art algorithms for computer vision

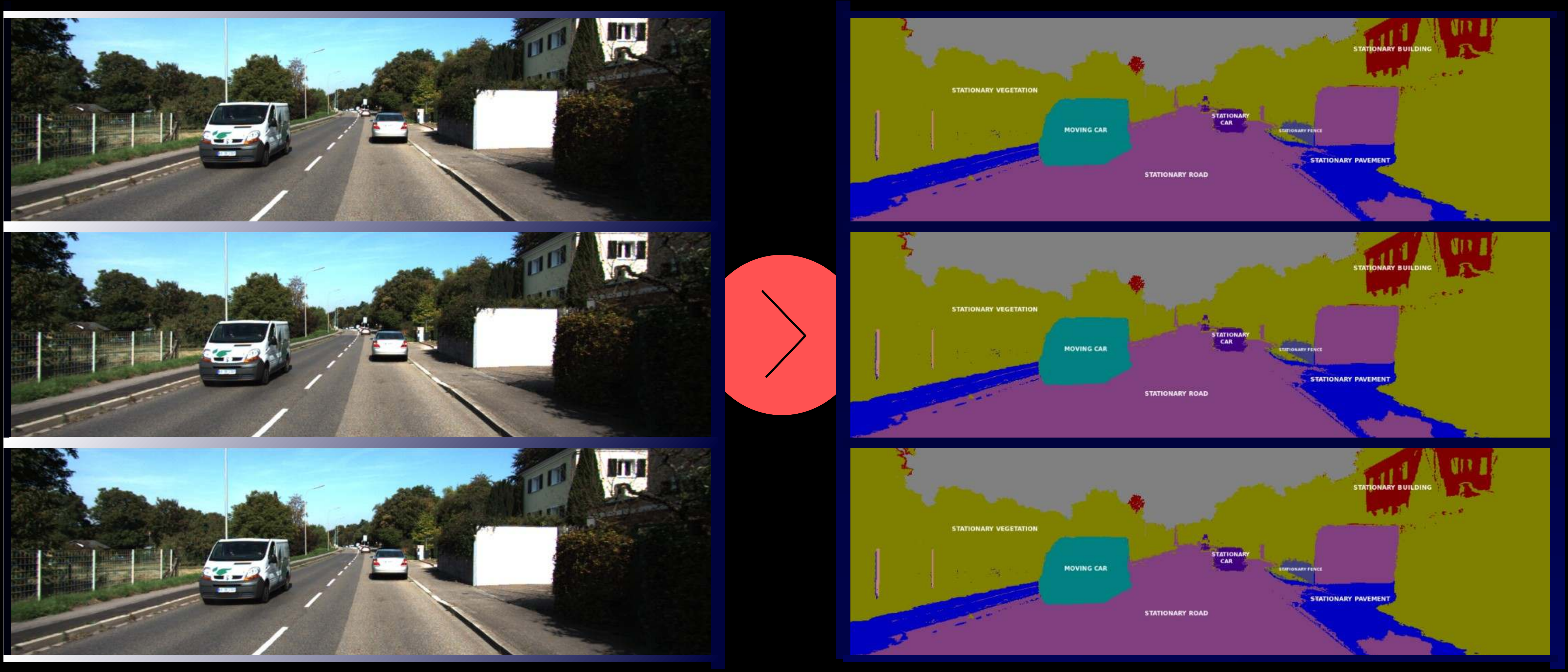
❑ Train in the Cloud at any scale

❑ Deploy inference code and model at the edge

Building an image dataset

aws SUMMIT

Annotating large datasets is time-consuming



Amazon SageMaker Ground Truth

https://aws.amazon.com/blogs/aws/amazon-sagemaker-ground-truth-build-highly-accurate-datasets-and-reduce-labeling-costs-by-up-to-70

Quickly label training data

Easily integrate human labelers

Get accurate results

KEY FEATURES

Automatic labeling via machine learning

Ready-made and custom workflows for images and text

Private and public human workforce

Label management

Experimenting and training at any scale

Do it yourself or fully-managed: you decide!

Amazon SageMaker

- Collect and prepare training data
- Choose and optimize your ML algorithm
- Set up and manage environments for training
- Train and Tune ML Models
- Deploy models in production
- Scale and manage the production environment

AWS Deep Learning AMI

TensorFlow K Microsoft CNTK ONNX

mxnet GLUON Chainer HOROVOD

Caffe2 PYTORCH torch

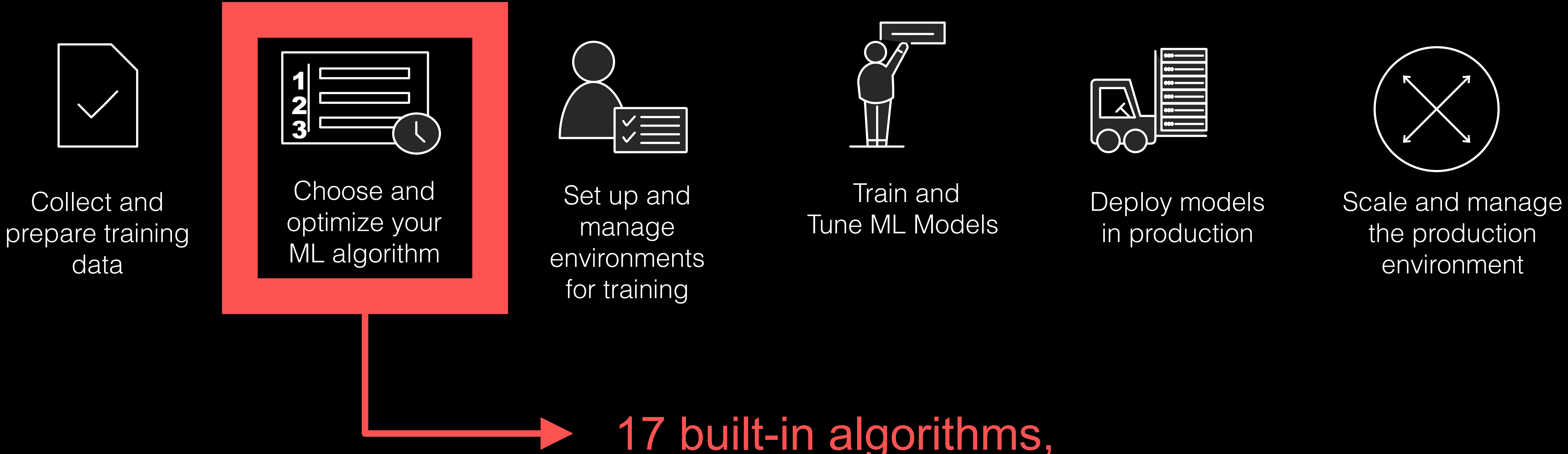
Amazon EC2

c
5

p3

Amazon SageMaker

Build, train, and deploy ML Models at any scale



Collect and prepare training data

Choose and optimize your ML algorithm

Set up and manage environments for training

Train and Tune ML Models

Deploy models in production

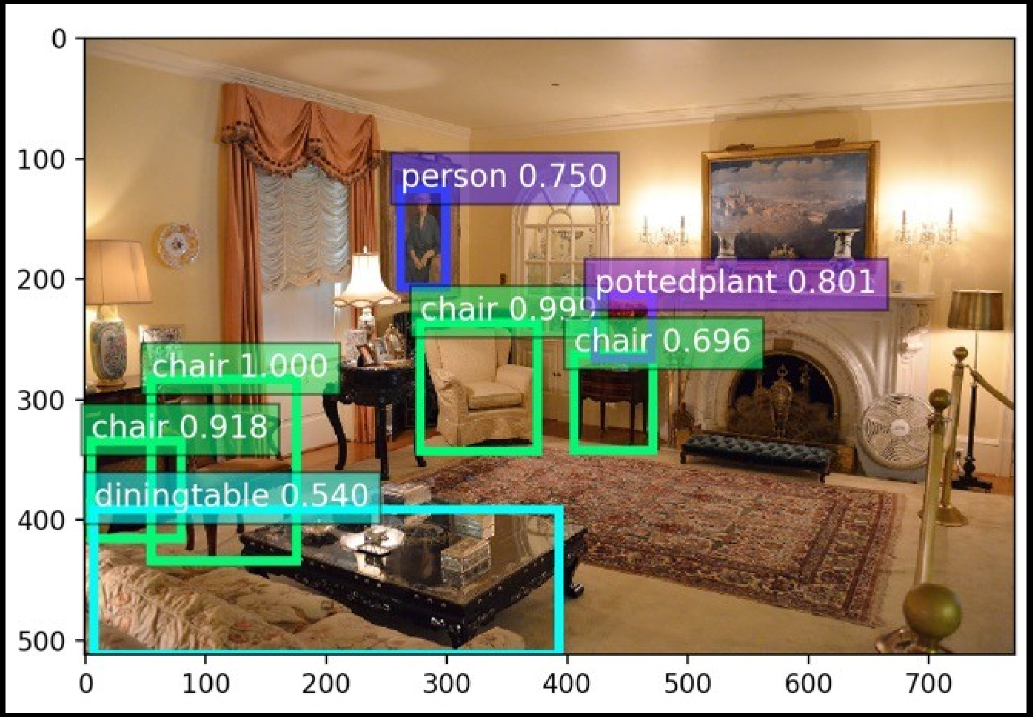
Scale and manage the production environment

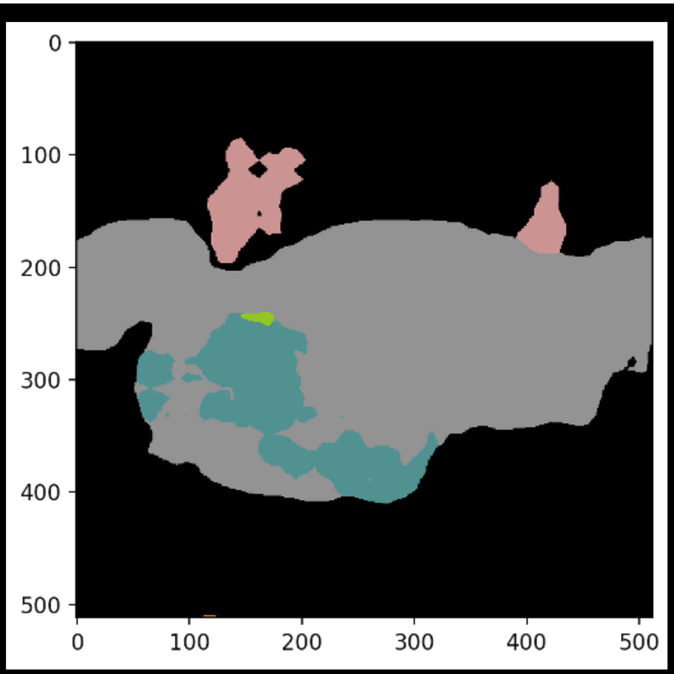
17 built-in algorithms, including 3 for computer vision

Deep Learning-based algorithms and pre-trained models

Classification, detection, segmentation

[electric_guitar],
with probability 0.671





Amazon SageMaker

Build, train, and deploy ML Models at any scale

Collect and prepare training data

Choose and optimize your ML algorithm

Set up and manage environments for training

Train and Tune ML Models

Deploy models in production

Scale and manage the production environment

Deploying inference code and model at the edge

Deploying at the edge with AWS Greengrass

- Install the Greengrass runtime
- Deploy and run Lambda functions
- Manage from AWS Console
- Same programming model as in the Cloud
- Local communication and orchestration

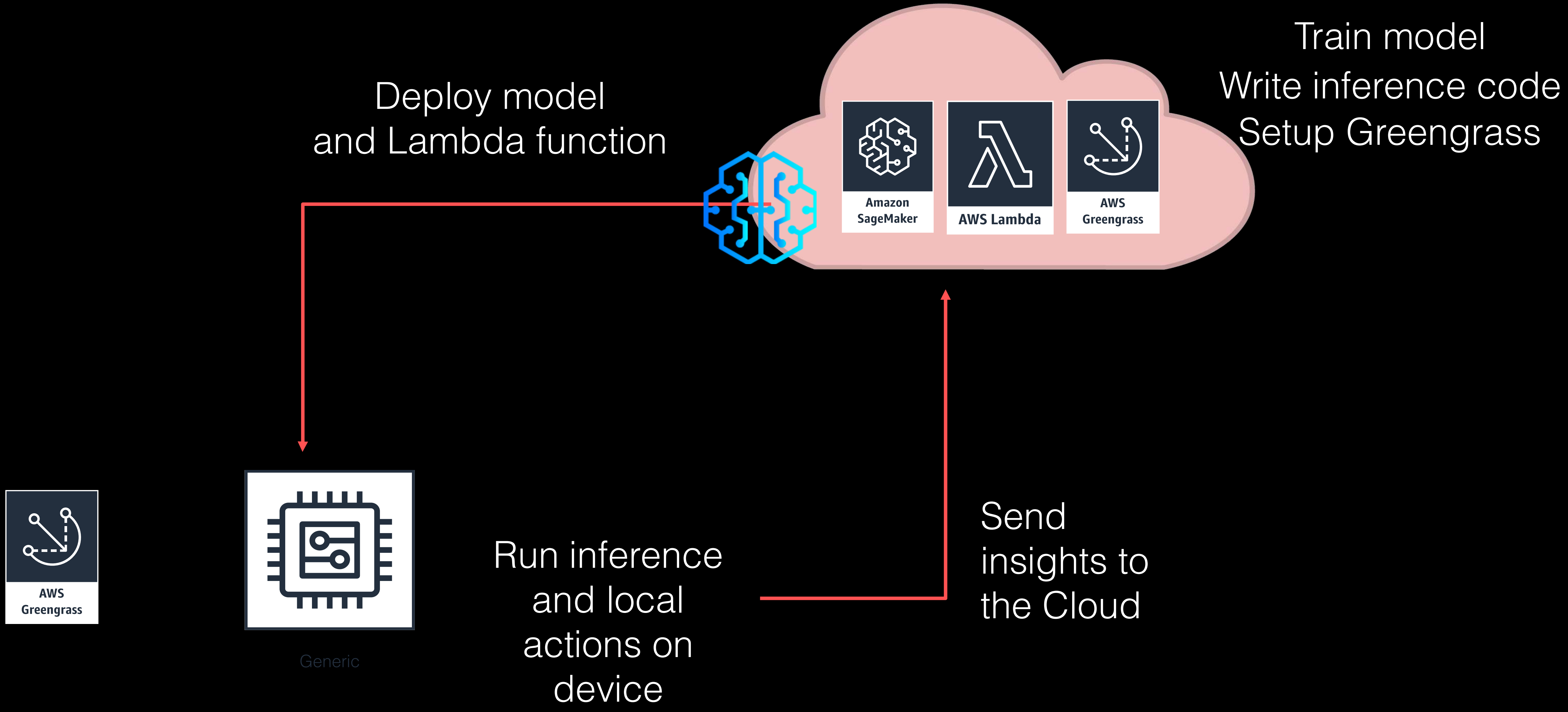
Deploying models with AWS Greengrass ML Inference

Define models as Greengrass resources and transfer them to your devices

Inference takes place on devices

Devices take action quickly – even when disconnected

Architecture



Train model

Write inference code

Setup Greengrass

Deploy model
and Lambda function

Run inference
and local
actions on
device

Send
insights to
the Cloud

aws SUMMIT

To-do list

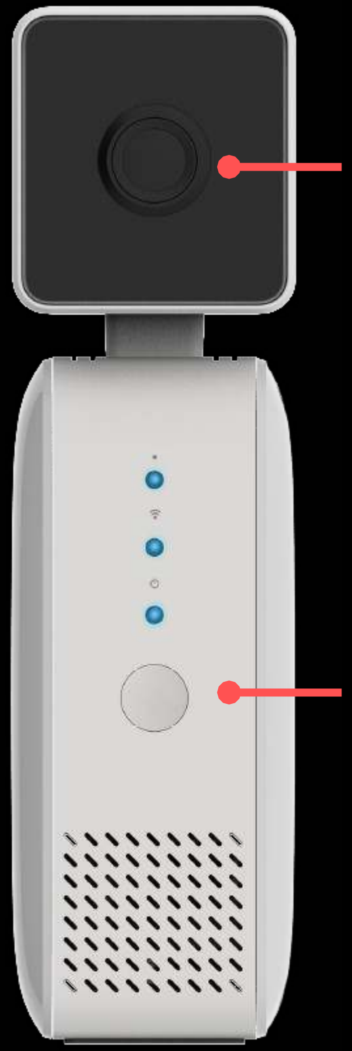
- ✓ Build a data set
- ✓ Experiment with state-of-the art algorithms for computer vision
- ✓ Train in the Cloud at any scale
- ✓ Deploy inference code and model at the edge
- ❑ Put it all in practice with a fun device

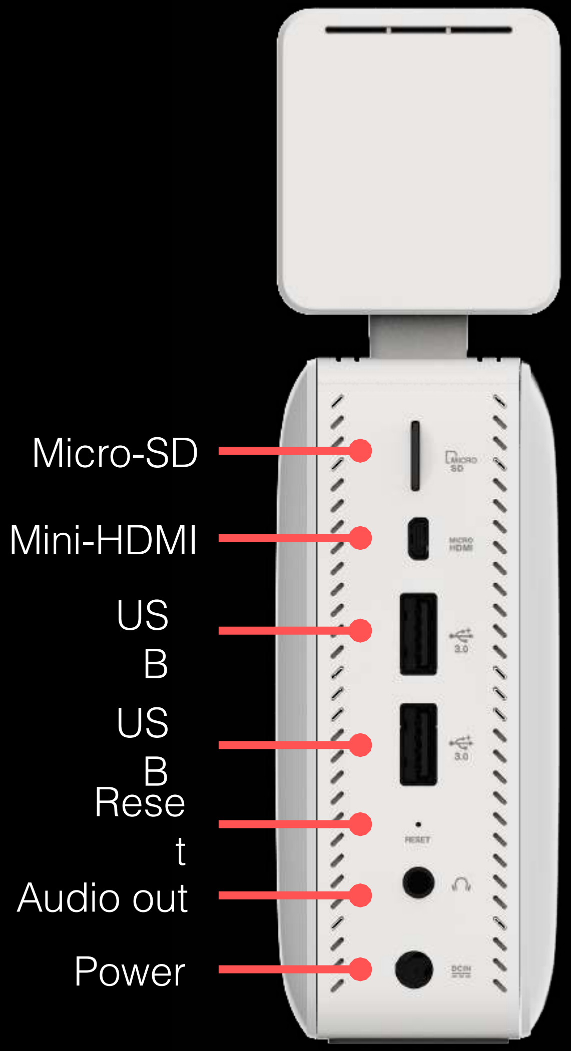
AWS DeepLens

aws SUMMIT

aws
intel
inside

AWS DeepLens





HD video camera with on-board compute optimised for deep learning

Integrates with Amazon SageMaker and AWS Lambda

10 MIN

From unboxing to first inference in <10 minutes

Tutorials, examples, demos, and pre-built models

aws SUMMIT

Get started in minutes with sample projects

OBJECT DETECTION

Detect and recognise objects.

HOT DOG NOT HOT DOG

Classify your food.

CAT AND DOG

Detect a cat or dog.

ARTISTIC STYLE TRANSFER

Transfer a style onto video.

ACTIVITY RECOGNITION

Recognise common activities.

FACE DETECTION

Detect faces of people.

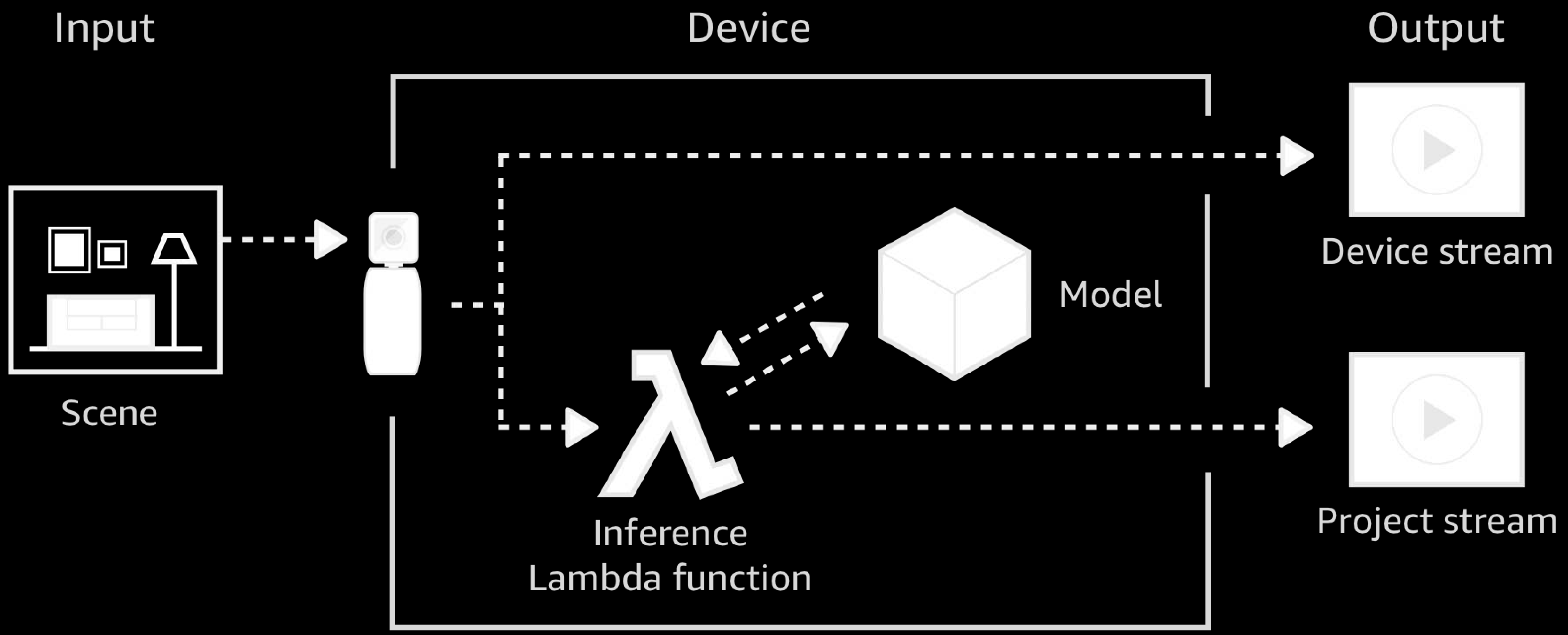
Use your own models with AWS DeepLens

- AWS DeepLens can run TensorFlow, Caffe and Apache MXNet models
  - Inception
  - MobileNet
  - NasNet
  - ResNet
  - Etc.

- Train or fine-tune your model on Amazon SageMaker

- Deploy to AWS DeepLens with AWS Greengrass

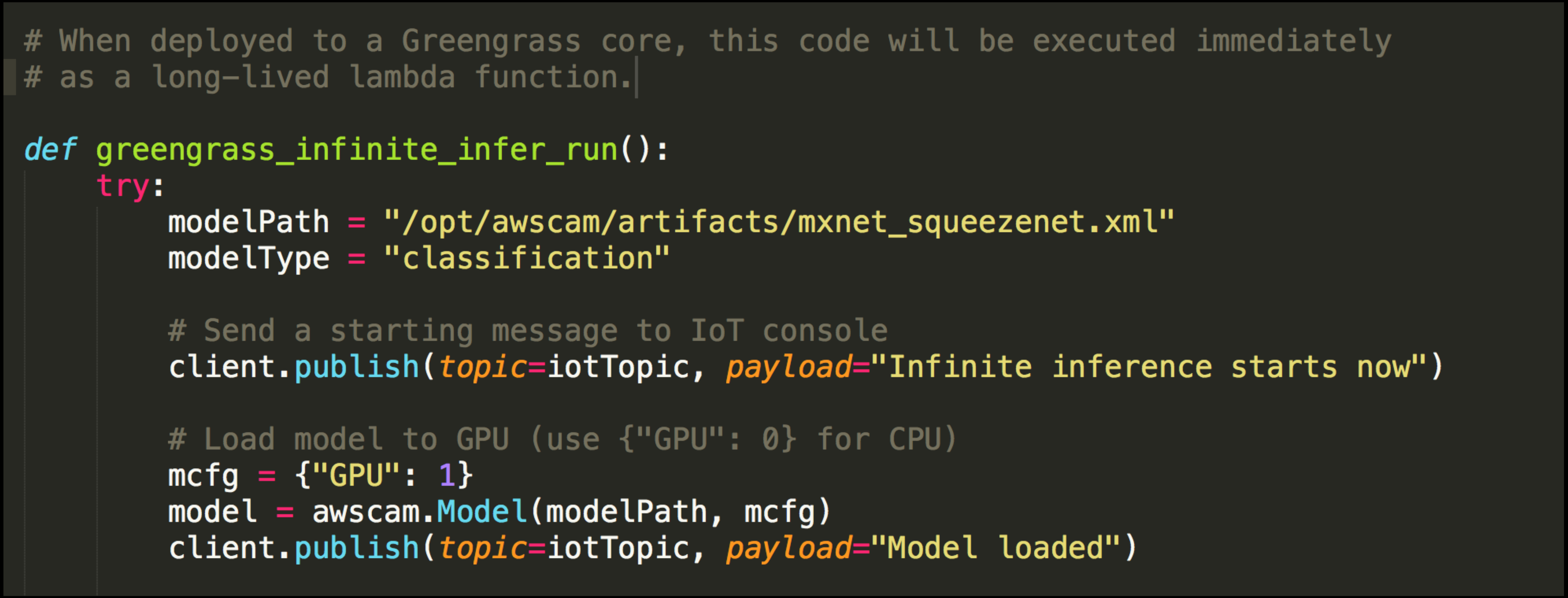
AWS DeepLens



Writing the Lambda function

not scary: it’s mostly cut and paste ;)

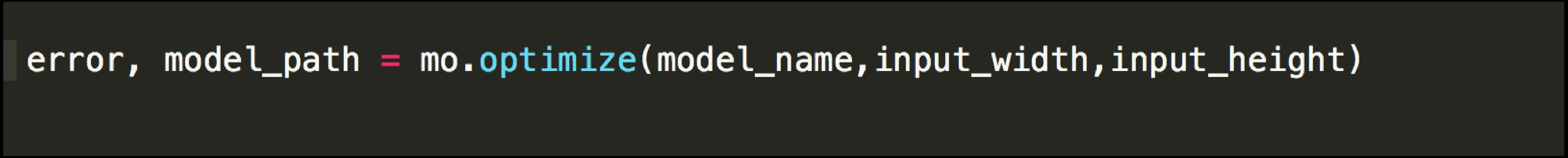
Lambda function: load the model



```
# When deployed to a Greengrass core, this code will be executed immediately
# as a long-lived lambda function.

def greengrass_infinite_infer_run():
    try:
        modelPath = "/opt/awscam/artifacts/mxnet_squeezenet.xml"
        modelType = "classification"

        # Send a starting message to IoT console
        client.publish(topic=iotTopic, payload="Infinite inference starts now")

        # Load model to GPU (use {"GPU": 0} for CPU)
        mcfg = {"GPU": 1}
        model = awscam.Model(modelPath, mcfg)
        client.publish(topic=iotTopic, payload="Model loaded")
```

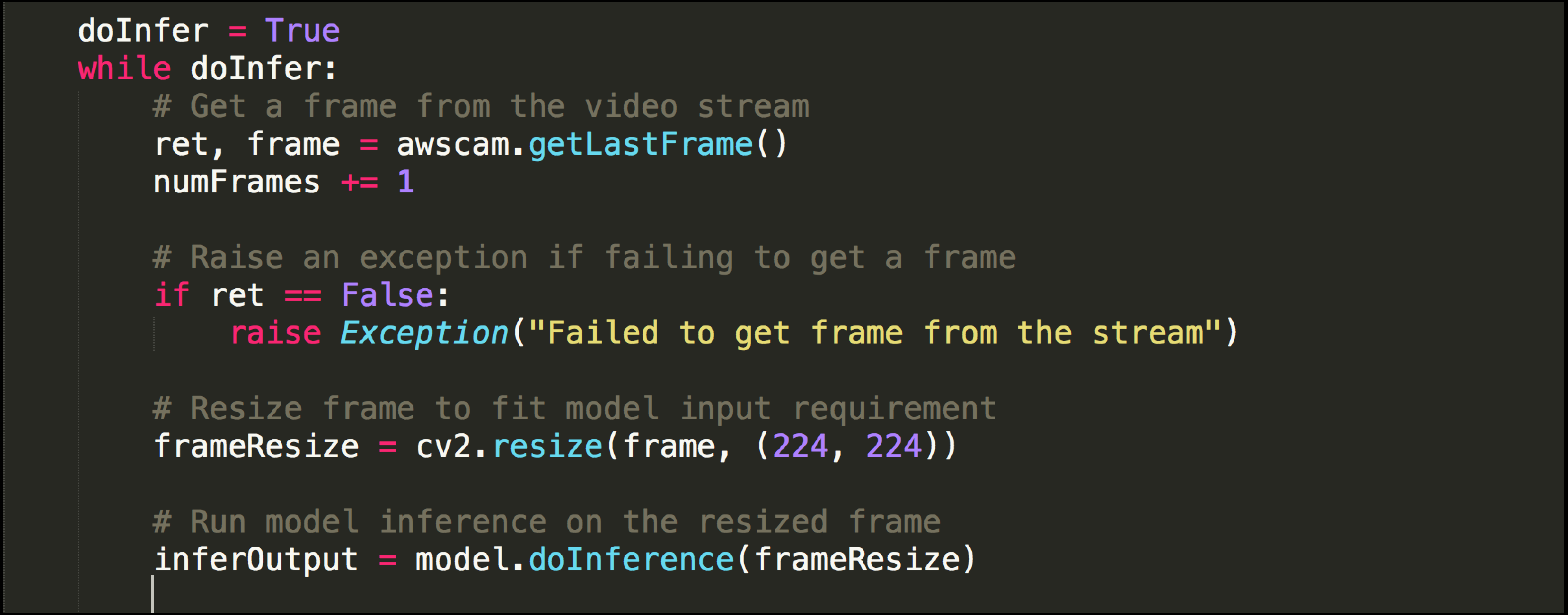
Lambda function: optimize a custom model

- Custom models need to be optimized for the on-board GPU.
- The first call optimizes the model, further calls do nothing.



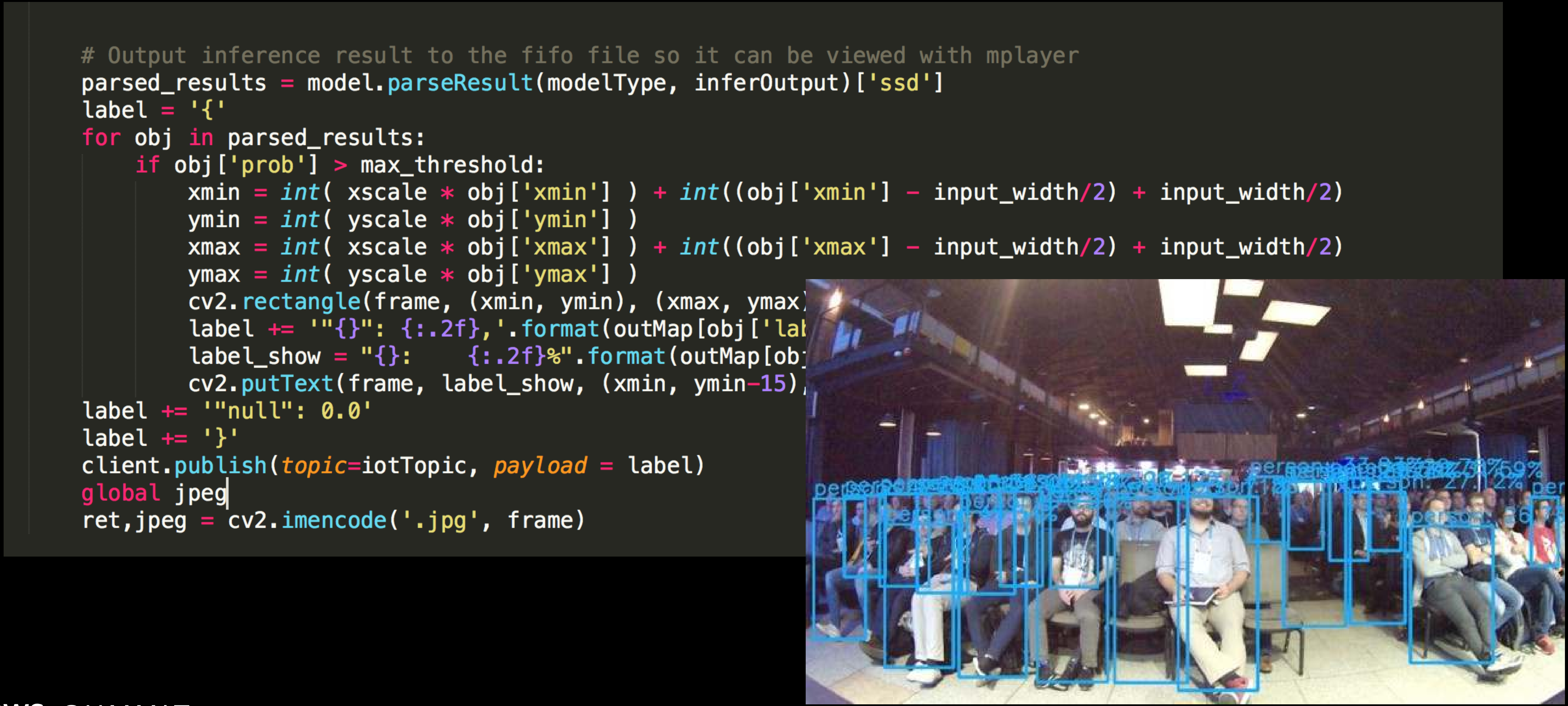
```
error, model_path = mo.optimize(model_name,input_width,input_height)
```

Lambda function: get a video frame and predict



```
doInfer = True
while doInfer:
    # Get a frame from the video stream
    ret, frame = awscam.getLastFrame()
    numFrames += 1

    # Raise an exception if failing to get a frame
    if ret == False:
        raise Exception("Failed to get frame from the stream")

    # Resize frame to fit model input requirement
    frameResize = cv2.resize(frame, (224, 224))

    # Run model inference on the resized frame
    inferOutput = model.doInference(frameResize)
```

Lambda function: annotate live stream



```
# Output inference result to the fifo file so it can be viewed with mplayer
parsed_results = model.parseResult(modelType, inferOutput)['ssd']
label = '{'
for obj in parsed_results:
    if obj['prob'] > max_threshold:
        xmin = int( xscale * obj['xmin'] ) + int((obj['xmin'] - input_width/2) + input_width/2)
        ymin = int( yscale * obj['ymin'] )
        xmax = int( xscale * obj['xmax'] ) + int((obj['xmax'] - input_width/2) + input_width/2)
        ymax = int( yscale * obj['ymax'] )
        cv2.rectangle(frame, (xmin, ymin), (xmax, ymax
        label += '"{}": {:.2f},'.format(outMap[obj['lab
        label_show = "{}:    {:.2f}%".format(outMap[obj
        cv2.putText(frame, label_show, (xmin, ymin-15),
label += '"null": 0.0'
label += '}'
client.publish(topic=iotTopic, payload = label)
global jpeg
ret,jpeg = cv2.imencode('.jpg', frame)
```

Demo

- Use the built-in algorithm for image classification in Amazon SageMager
- Fine tune a pre-trained model on the CIFAR-256 image dataset
- Write a simple Lambda function for inference
- Deploy function and model to AWS DeepLens

https://gitlab.com/juliensimon/dlnotebooks/sagemaker/

ml.aws

aws.amazon.com/deeplens

aws.training/machinelearning

Thank you!

Julien Simon
Global Evangelist, AI and Machine Learning

@julsimon
https://medium.com/julsimon

!

Please complete the session survey.

aws SUMMIT