AWS re:Invent

AIM410-R

Deep learning applications with TensorFlow, featuring Mobileye

Julien Simon
Global Evangelist, AI/ML
Amazon Web Services
@julsimon

Chaim Rand
ML Algorithm Developer
Mobileye

AWS re:Invent

aws

Agenda

TensorFlow on AWS

Customer case study: Mobileye

Demo: TensorFlow on Amazon SageMaker

Getting started

TensorFlow

https://www.tensorflow.org

- Main API in Python, with support for Javascript, Java, C++
- TensorFlow 1.x: symbolic execution
    - ‘Define then run’: build a graph, optimize it, feed data, and compute
    - Low-level API: variables, placeholders, tensor operations
    - High-level API: *tf.estimator.**
    - Keras library: *Sequential* and *Functional* API, predefined layers
- TensorFlow 2.0: imperative execution (aka eager execution)
    - ‘Define by run’: normal Python code, similar to numpy
    - Run it, inspect it, debug it
    - Keras is the preferred API

AWS: The platform of choice for TensorFlow

https://aws.amazon.com/tensorflow/

Adobe AUTODESK edmunds Expedia

SIEMENS grammarly intuit Pinterest

hudl COX AUTOMOTIVE THOMSON REUTERS Zocdoc

GoDaddy News Corp Australia zendesk

zalando babylon

89% of all deep learning workloads in the cloud run on AWS

85% of all TensorFlow workloads in the cloud run on AWS

Source: Nucleus Research, T147, October 2019

TensorFlow: a first-class citizen on Amazon SageMaker

- Built-in TensorFlow containers for training and prediction
  - Code available on Github: https://github.com/aws/sagemaker-tensorflow-containers
  - Build it, run it on your own machine, customize it, etc.
  - Versions : 1.4.1 → 1.15 (2.0 coming soon)

- Not just TensorFlow
  - Standard tools: TensorBoard, TensorFlow Serving
  - SageMaker features: Local Mode, Script Mode, Model Tuning, Spot Training, Pipe Mode, Amazon EFS & Amazon FSx for Lustre, Amazon Elastic Inference, etc.
  - Performance optimizations: GPUs and CPUs (AWS, Intel MKL-DNN library)
  - Distributed training: Parameter Server and Horovod

Amazon SageMaker

re:Invent 2019 announcements

First fully integrated development environment (IDE) for machine learning
SageMaker Studio

Enhanced notebook experience with quick-start & easy collaboration
SageMaker Notebooks
(preview)

Automatic debugging, analysis, and alerting
SageMaker Debugger

Experiment management system to organize, track, & compare thousands of experiments
SageMaker Experiments

Model monitoring to detect deviation in quality & take corrective actions
SageMaker Model Monitor

Automatic generation of ML models with full visibility & control
SageMaker Autopilot

Amazon SageMaker at Mobileye

Chaim Rand
ML Algorithm Developer
Mobileye

MOBILEYE®
An Intel Company

AWS re:Invent

aws

Making Amazon SageMaker work for you

- Story of how we adopted Amazon SageMaker
- Ways in which Amazon SageMaker accelerated our development process
- Challenges we faced and how we overcame them

Spoiler:

Adopting Amazon SageMaker enabled us to reduce our development by up to 10X (from several months to under a week)

Chapter 1 - Introduction

A bit about Mobileye

Founded in 1999 by Prof. Amnon Shashua and Mr. Ziv Aviram

Goal: use computer vision-based technologies to

- revolutionize the transportation industry
- make roads safer
- and save lives

Acquired by Intel in 2017 for $15.3 billion

MOBILEYE®
An Intel Company

A bit about Mobileye

We develop a range of software products, deployed on a proprietary family of computer chips named EyeQ

Leading supplier of software that enables advanced driver-assistance systems (ADAS)—deployed in over 40 million vehicles. (Includes adaptive cruise control, collision avoidance, lane departure warning, …)

Over 25 automaker partners, including most of the world's largest

Some of our technologies

Rely on monocular camera perception

- Reduces cost
- Other sensors can be used for redundancy when working on high level autonomous driving

Some of our Sensing Technologies

Use deep neural networks (DNNs) to detect:

- **Road users** – including vehicles and pedestrians
- **Road semantics** – including traffic lights, traffic signs, on-road arrows, stop-lines, and crosswalks
- **Road boundaries** – any delimiter of the drivable area, its 3D structure and semantics, including curbs, cones and debris
- **Road geometry** – driving paths and surface profile, including speed bumps and roadside ditches


egoSpeed: 8.86 yawRate: 0.00 expDTime: 0.03 egoAccel: -0.34 Day gfi: 1
distance
VPP

Scene Segmentation

From ADAS to autonomous

The key to full autonomy relies on three technological pillars

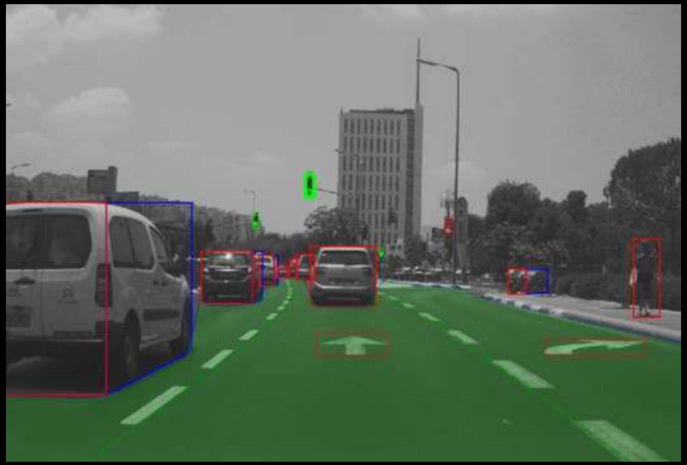
Sensing

Mapping

Driving policy

From ADAS to autonomous

The key to full autonomy relies on three technological pillars



Sensing

Mapping

Driving policy

Identify all artifacts in our surrounding environment

From ADAS to autonomous

The key to full autonomy relies on three technological pillars

Sensing

Mapping

Driving policy

Localize the vehicle on a map of the surrounding environment

- Up to <10cm precision

From ADAS to autonomous

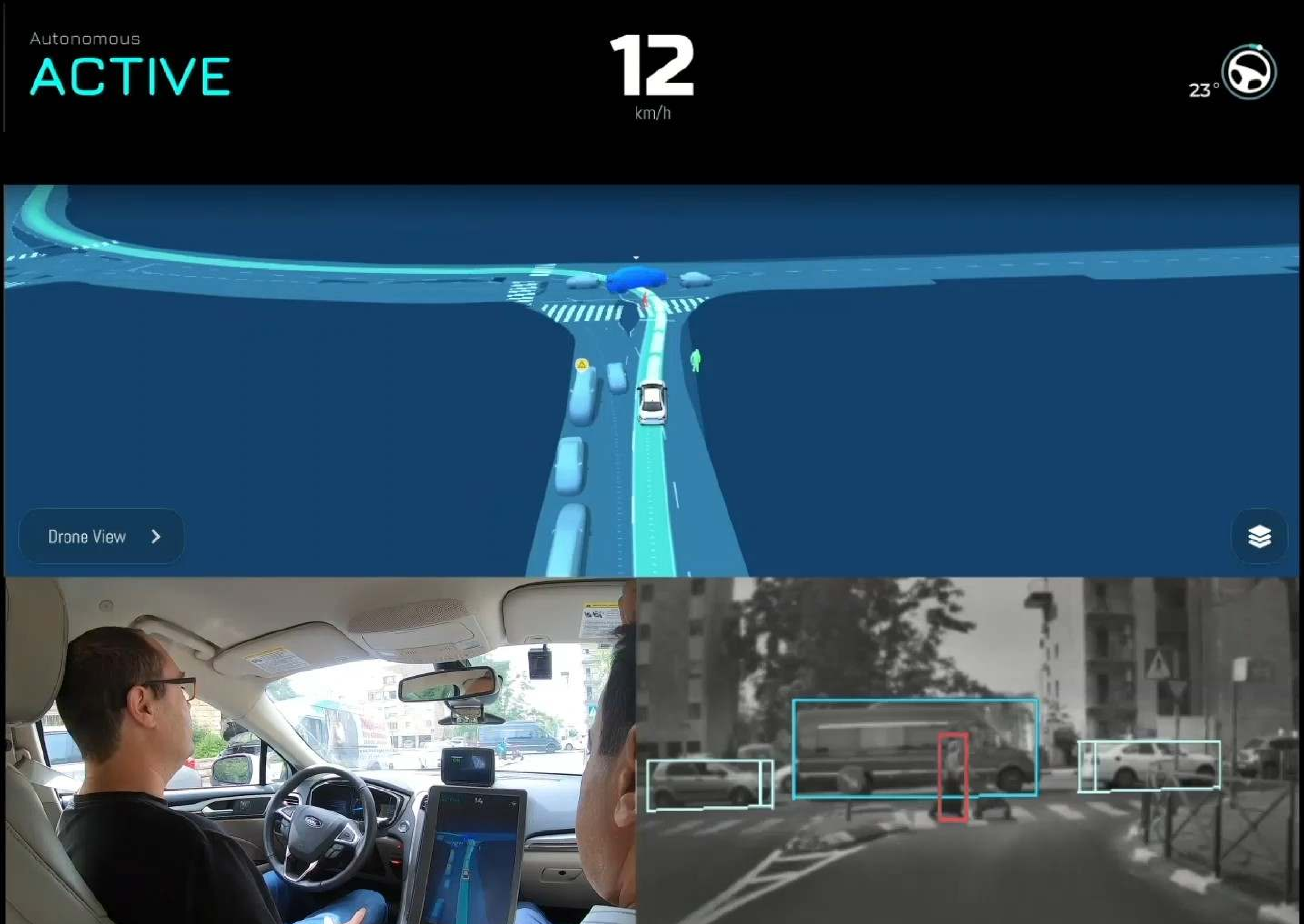
The key to full autonomy relies on three technological pillars

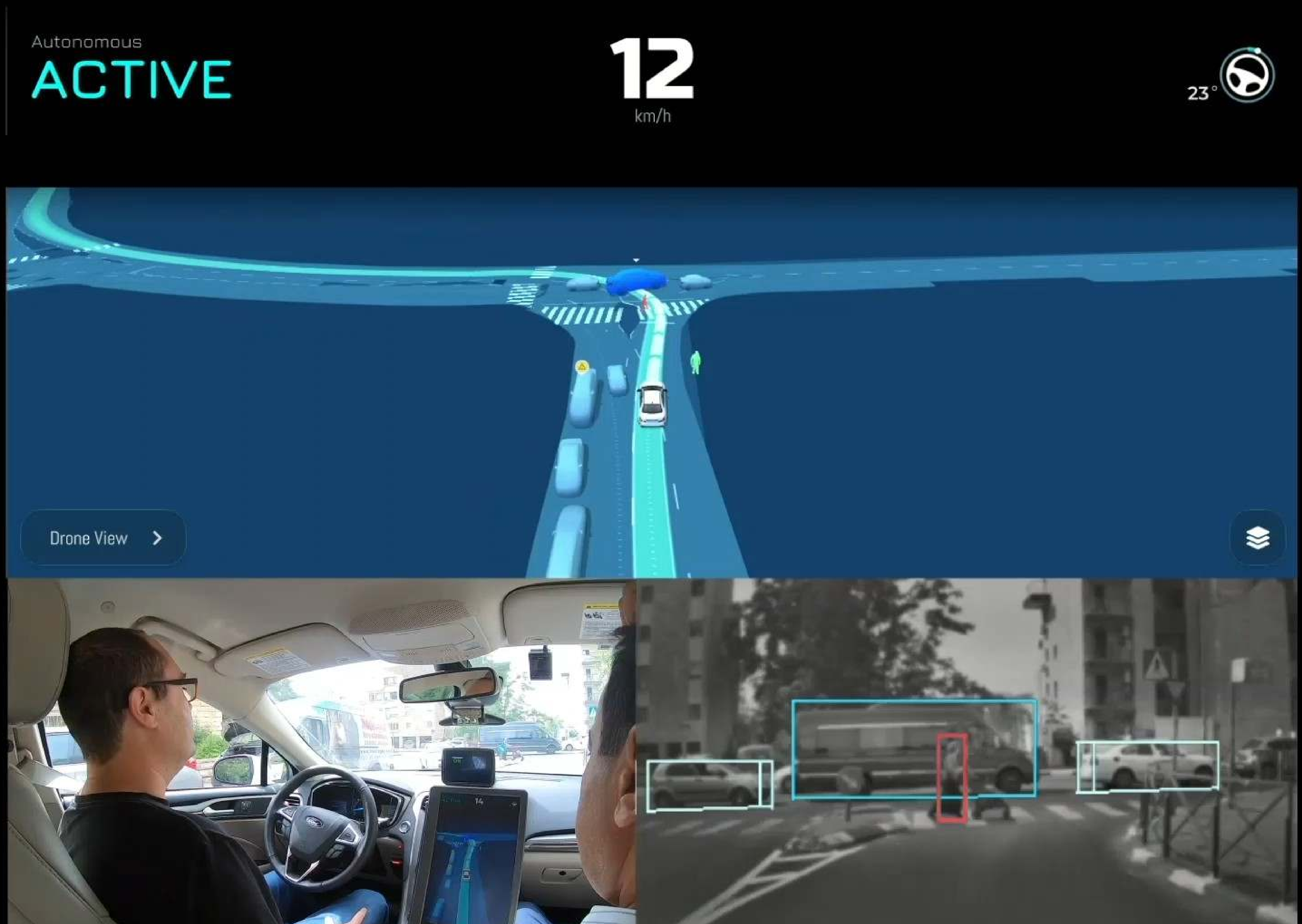
Sensing

Mapping

Driving policy

Decide what actions to take based on the input from Sensing and Mapping


Autonomous
ACTIVE
12
km/h
23°
Drone View


Autonomous
ACTIVE
12
km/h
23°
Drone View

From ADAS to autonomous

Learn more:

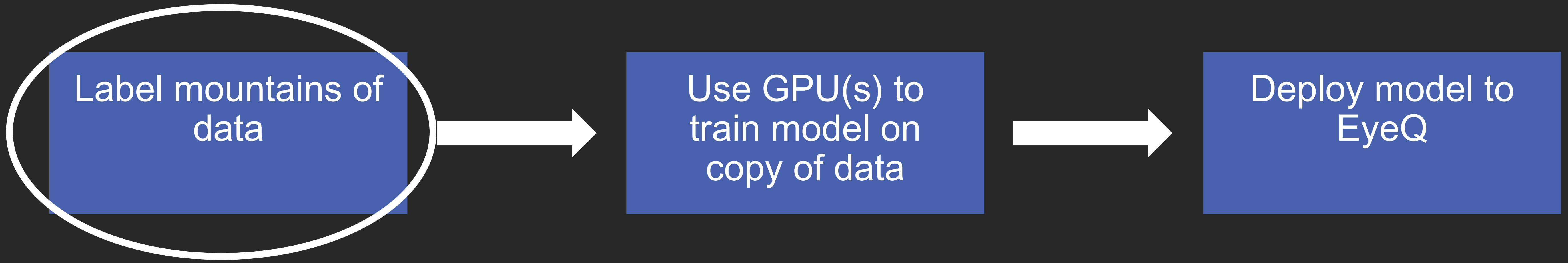
- From Mobileye VP Tal Babaioff
  - **AUT307 - Navigating the winding road toward driverless mobility**

- Online https://www.mobileye.com

Chapter 2 – Enter Amazon SageMaker

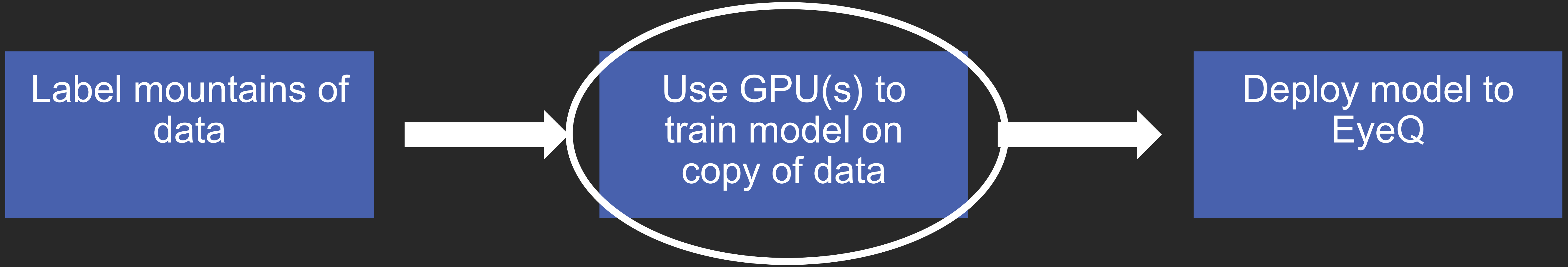
DNN training challenge

- Variety of different DNN architectures
- Mountains of data: a typical model may train on up to 200TB of data
- Simplified development cycle:



DNN training challenge

- Variety of different DNN architectures
- Mountains of data: a typical model may train on up to 200TB of data
- Simplified development cycle:



DNN training challenge

- Variety of different DNN architectures
- Mountains of data: a typical model may train on up to 200TB of data
- Simplified development cycle:

Label mountains of data

Use GPU(s) to train model on copy of data

Deploy model to EyeQ

DNN training challenge

- Variety of different DNN architectures
- Mountains of data: a typical model may train on up to 200TB of data
- Simplified development cycle:

Label mountains of data → Use GPU(s) to train model on copy of data → Deploy model to EyeQ

- Historically performed on premise

Training on premises

- Obvious drawbacks to training on premises
  - Limit to number of GPU instances
  - Limitations to data capacity
  - Challenge of staying up to date with latest HW and SW

- Any alternative must comply with our development pipeline, and must keep our IP safe

- Enter Amazon SageMaker …

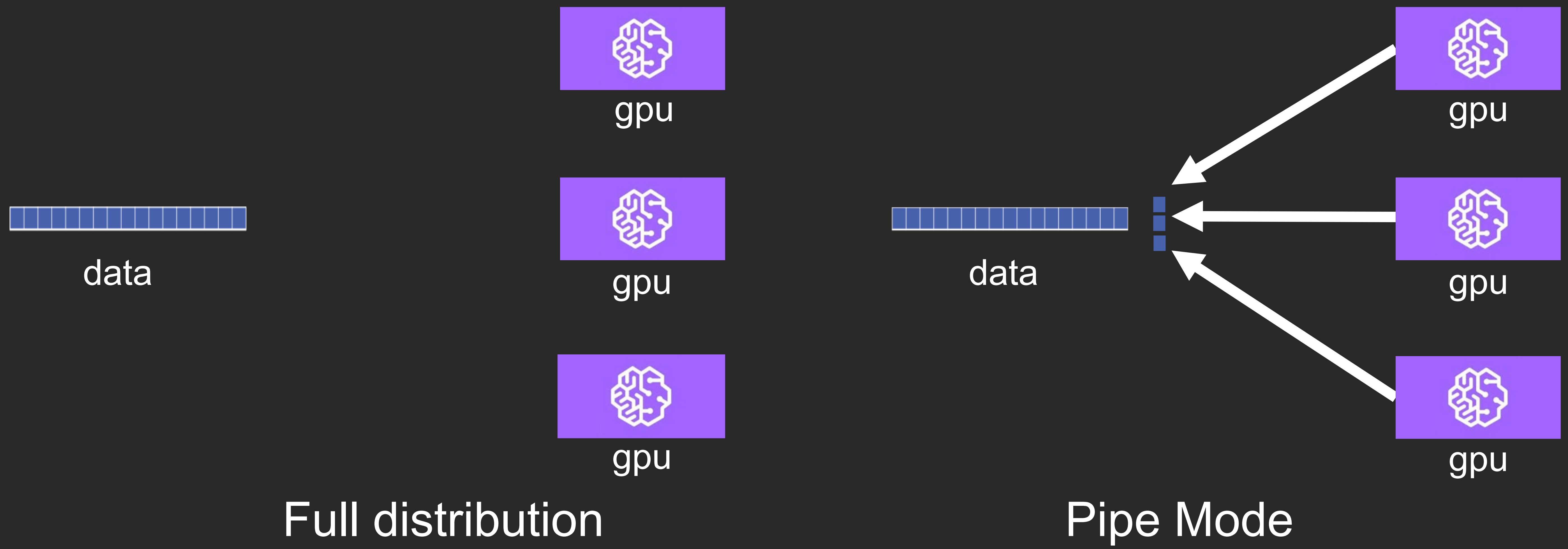
What I like about Amazon SageMaker

- Unconstrained capacity – means I can spin up as many training sessions as I need
- Instance type variety
  - Multi-generation CPU and GPU support
  - Up to date with latest HW and SW (tuned to maximize efficiency)
  - Distributed training support
- Learning curve
  - Well documented with many samples
- Secure – means data security team will let me use it
- Pipe Mode support

Chapter 3 – Amazon SageMaker Pipe Mode

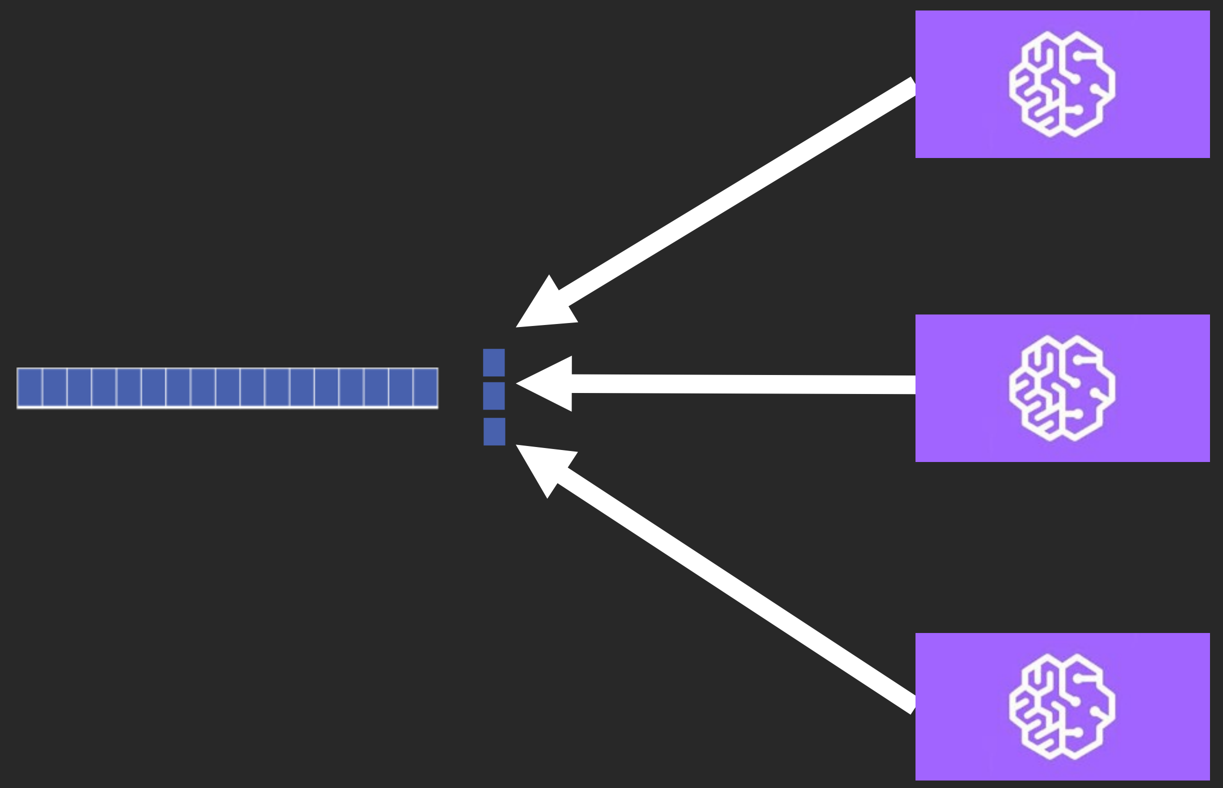
What is Amazon SageMaker Pipe Mode?

- Enables streaming training data from Amazon S3 to training instances
  - Instead of copying data to each training device



Benefits of Pipe Mode

- Enables streaming training data from Amazon S3 to training instances
  - Removes limitation on dataset size
  - No need to have dedicated (and costly) storage for each training instance or to download data to each instance, and no delay to training start
- Enables decoupling of data from training instances
  - Multiple training instances can all pull from the same dataset in S3



Pipe Mode with TensorFlow

- Lest you should fear the need to have to manage the data stream on your own…
- Amazon SageMaker wraps the pipe with an implementation of the *tf.data.Datasets* API
  - Complete with all the standard dataset functions
  - Ready to be fed directly into your model

Setting up Pipe Mode

```
from sagemaker.tensorflow import TensorFlow

tensorflow = TensorFlow(
        entry_point='pipemode.py',
        input_mode='Pipe’, …)

pipes = {'train':'s3://sagemaker-path-to-data’}

tensorflow.fit(pipes)
```

PipeModeDataset initialization

```
def parse(record):
    feature = {'label': tf.FixedLenSequenceFeature([], tf.int64, allow_missing=True),
               'image_raw': tf.FixedLenFeature([], tf.string)}
    features = tf.parse_single_example(record, feature)
    image = tf.decode_raw(features['image_raw'], tf.uint8)
    label = features['label']
    return {"image": image}, label # This is what will be fed into your model
ds = PipeModeDataset("train", record_format='TFRecord')
ds = ds.apply(map_and_batch(parse, batch_size=32, num_parallel_batches=2))
return ds
```

Chapter 4 – Pipe Mode Challenges

Pipe Mode challenges

- Converting data to supported format
- Sequential nature of pipe mode
- Pipe number limitation
  - Currently stands at 20

Pipe Mode challenges

- Converting data to supported format
  - Amazon SageMaker’s *PipeModeDataset* supports a limited number of formats
  - Format conversion of “mountains” of data to *TFRecord* format seemed daunting

Generating data in *TFRecord* format

- Turned out to be a blessing in disguise
  - Task was performed using AWS Batch
  - Used up to hundreds of thousands of vCPUs in parallel. (Each created a100MB TFRecord file.)
    - Tip: split generated *TFRecord* files into 100MB chunks
  - Accelerated data preparation time significantly (from days to hours)

Generating data in *TFRecord* format

- One more step to moving end-to-end development to cloud:

Raw data and labels stored in S3 and managed with Amazon DynamoDB

→

TFRecord files generated in S3 using AWS Batch

→

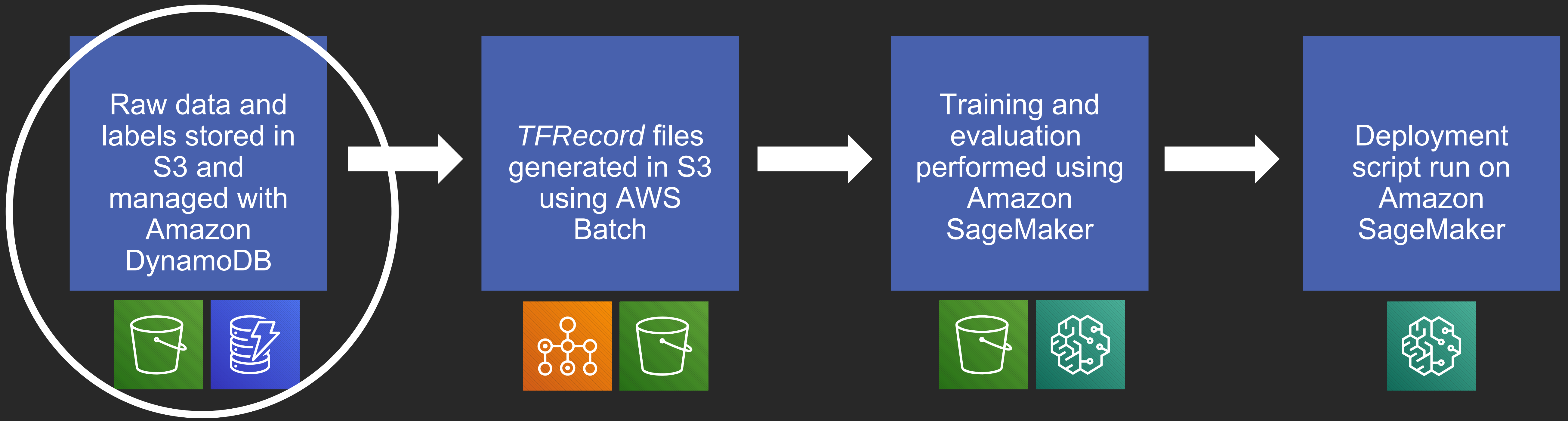
Training and evaluation performed using Amazon SageMaker

→

Deployment script run on Amazon SageMaker

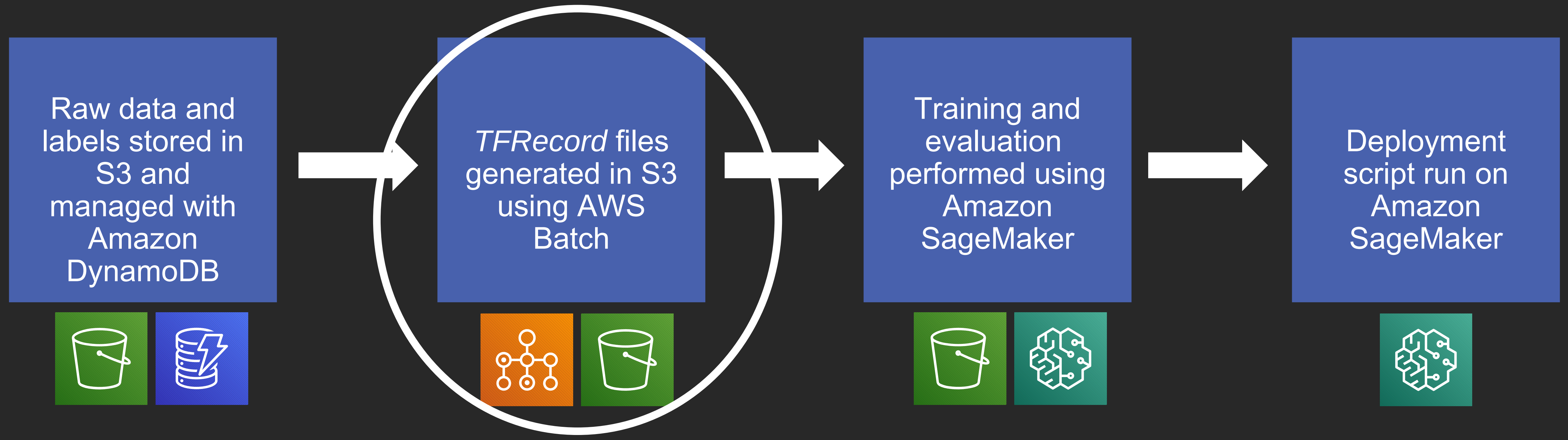
Generating data in *TFRecord* format

- One more step to moving end-to-end development to cloud:



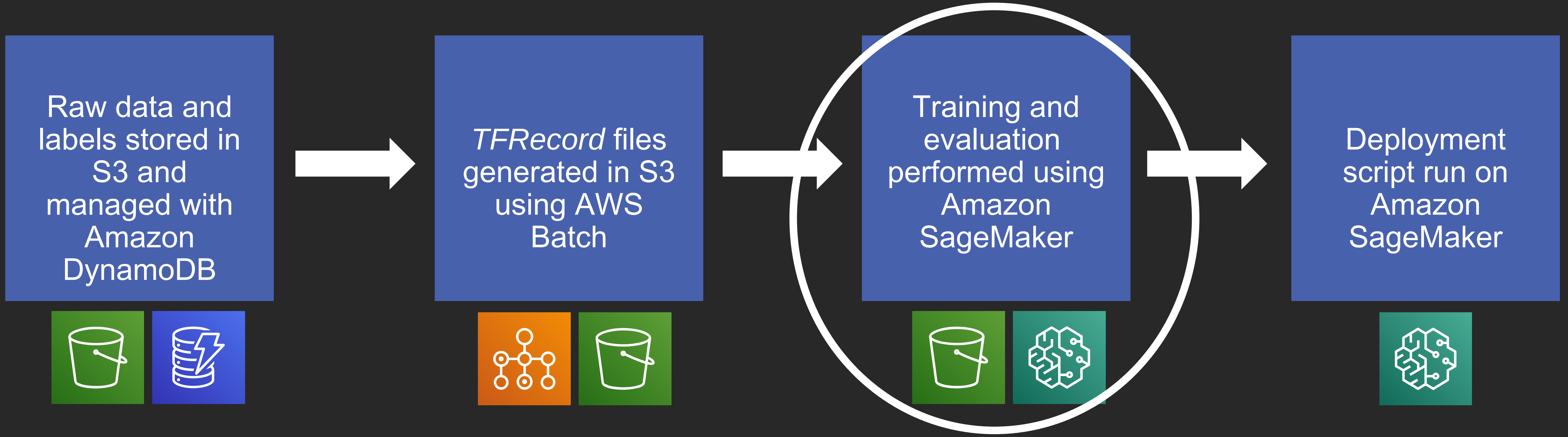
Generating data in *TFRecord* format

- One more step to moving end-to-end development to cloud:



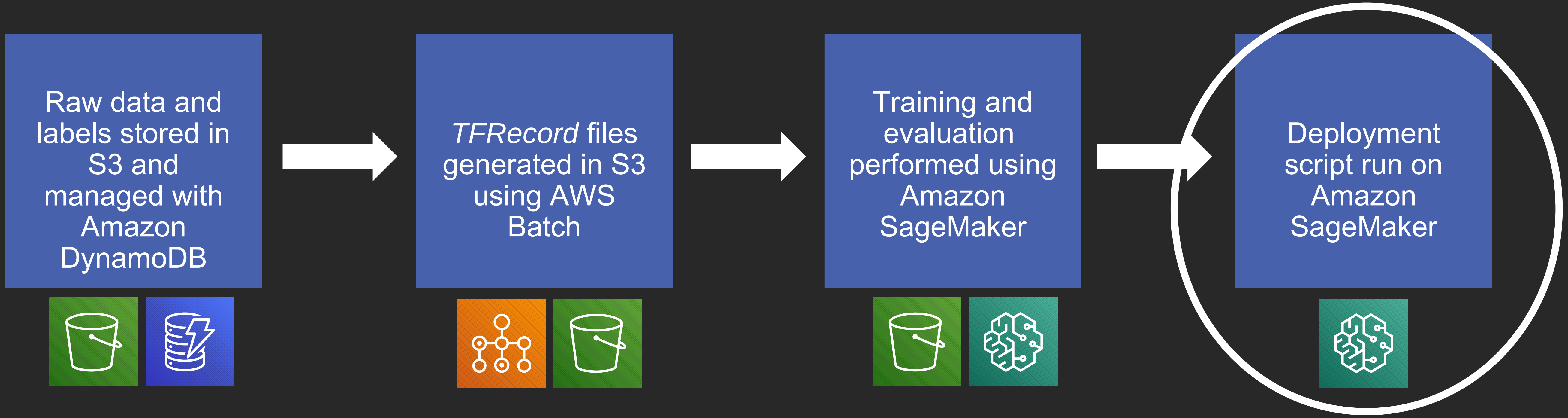
Generating data in *TFRecord* format

- One more step to moving end-to-end development to cloud:



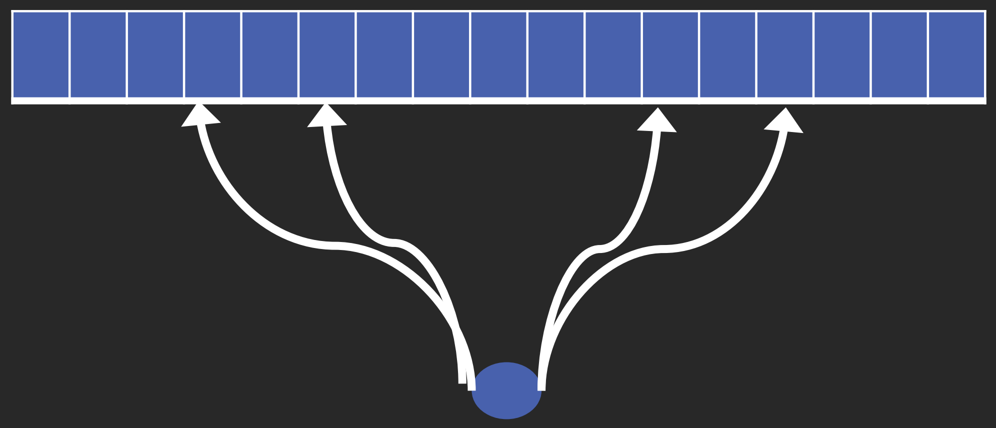
Generating data in *TFRecord* format

- One more step to moving end-to-end development to cloud:



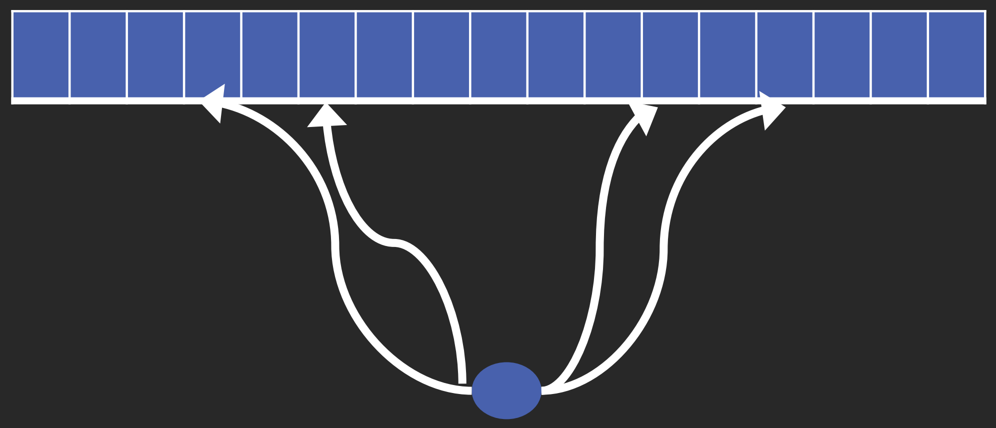
Pipe mode challenges

- Sequential nature of pipe mode
  - Overcoming the lack of random access to full dataset
  - Relied on for shuffling and boosting



Overcoming the lack of random access to data

- Challenge: shuffling
  - In the pre-Amazon SageMaker era, we relied on random access to the data to ensure shuffling



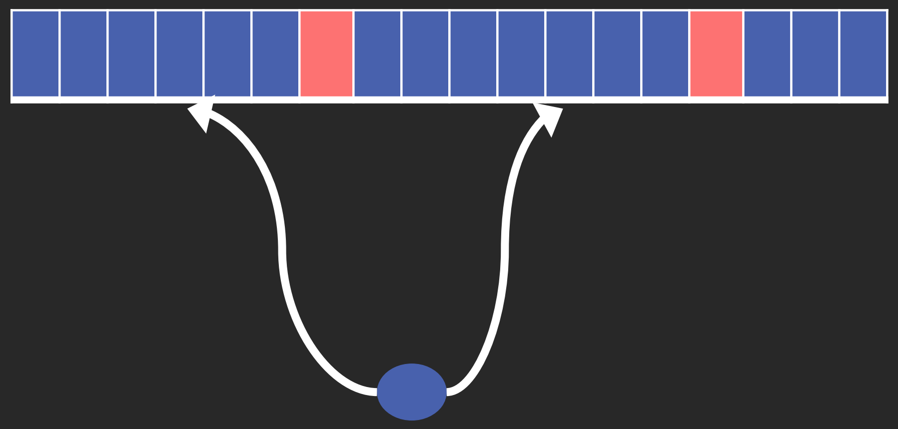
- Solution: introduce shuffling on a number of levels
  - Using Amazon Sagemaker *ShuffleConfig* class to shuffle *TFRecord* files for each epoch

```
train_data = s3_input('s3://sagemaker-path-to-train-data‘,
                      shuffle_config=ShuffleConfig(seed))
pipes = {'train':train_data}
```

  - Using TF dataset shuffle

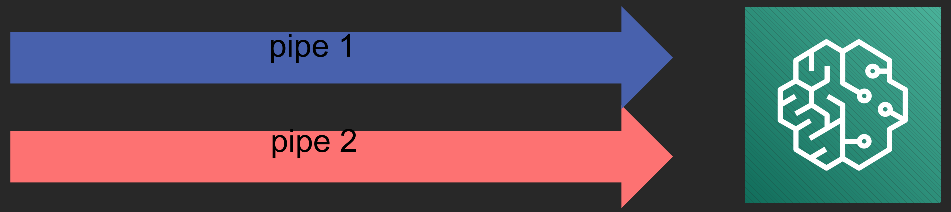
Overcoming the lack of random access to data

- Challenge: boosting
  - Solution for increasing representation of underrepresented data
    - E.g., pink cars in a vehicle detection DNN
  - In the pre-Amazon SageMaker era, we relied on random access to the data to perform boosting



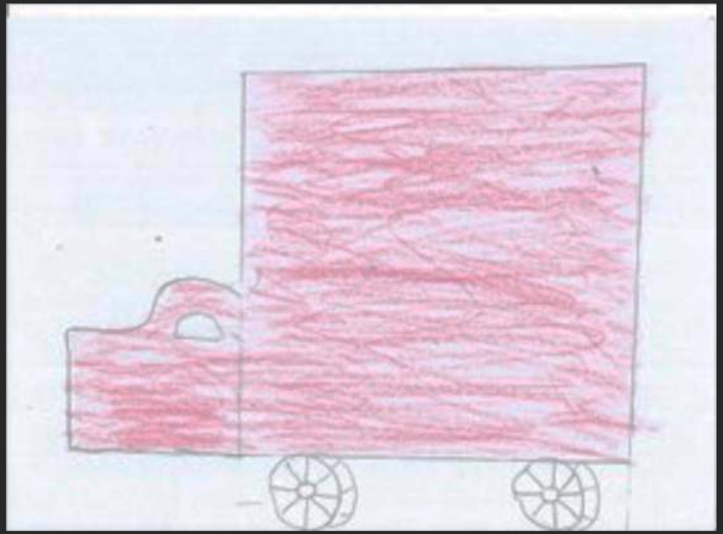
- Solution:
  - Underrepresented data can be separated and fed using a dedicated pipe
  - *Dataset* APIs can be used to associate a weight with each pipe and interleave the pipes:

```
ds = tf.data.sample_from_datasets(datasets, weights)
```



Pipe Mode challenges

- Overcoming the limitation on number of pipes
- Why might I need more than 20 pipes?
  - Boost underrepresented data



Pipe Mode challenges

- Overcoming the limitation on number of pipes
- Why might I need more than 20 pipes?
  - Boost underrepresented data
  - Distributed training with Horovod, multiply by number of GPUs

Overcoming the limitation on number of pipes

- Solution: use Pipe Mode manifest files
  - An alternative way to configure an Amazon SageMaker pipe
  - Replace path prefix with a list of files
  - Include the same file multiple times to increase its weight
  - Have finer control over the data used for training

```
data = s3_input('s3://path-to-manifest-file',
                s3_data_type='ManifestFile',
                shuffle_config=ShuffleConfig(seed))
```

Chapter 5 – Closing Remarks

Other considerations when moving to Amazon SageMaker

- Blog post: https://bit.ly/2sLRJb5
- Distributed training
  - How to choose the instance type with the optimal number of GPUs
  - Should one use TensorFlow distributed training or Horovod?
  - How does one maximize resource utilization?
- How to take advantage of Spot Instances to reduce cost
  - How to ensure that mid-train termination won’t have negative impact
- Using TensorBoard and other benchmarking tools
- Debugging on a remote environment
  - Tip: make sure your model compiles and runs locally before running on cloud

Summary

- Adopting Amazon SageMaker has boosted our DNN development capabilities
- Pipe Mode offers a compelling solution for training with large datasets
- Reduce development time significantly (~10X)
  - Scalability enables you to test multiple models in parallel
  - Integrating with other AWS services can provide further acceleration
- Challenges were overcome using advanced Amazon SageMaker features and TF dataset APIs
- You can make Amazon SageMaker work for you too

Amazon SageMaker at Mobileye

Chaim Rand

ML Algorithm Developer
Mobileye

AWS re:Invent

Demo

https://gitlab.com/juliensimon/aim410

Amazon SageMaker

Build, train, deploy machine learning models quickly at scale

NEW!
SageMaker Studio
IDE

Amazon SageMaker

- Ground Truth
- ML Marketplace
- Algorithms & Frameworks
- NEW! Quick-start notebooks
- Training & Tuning
- Reinforcement Learning
- NEW! Experiments
- NEW! Debugger
- NEW! Autopilot
- Neo
- Deployment & Hosting
- NEW! Monitoring

Getting started

http://aws.amazon.com/free

https://aws.amazon.com/tensorflow/

https://aws.amazon.com/sagemaker

https://github.com/aws/sagemaker-python-sdk

https://sagemaker.readthedocs.io/en/stable/using_tf.html

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

https://gitlab.com/juliensimon/dlnotebooks

Related breakouts

[AUT307] [Navigating the winding road toward driverless mobility, with Mobileye]
Dec 4, 4:00 PM – Aria, Level 1 West, Bristlecone 9 Red

[AIM410R1] [Deep learning applications with TensorFlow, with Fannie Mae]
Dec 5, 1:00 PM – Venetian, Level 3, Lido 3002

[AIM307] [Amazon SageMaker deep dive: A modular solution for machine learning]
Dec 4, 1:45 PM – Venetian, Level 3, Lido 3005

Learn ML with AWS Training and Certification

The same training that our own developers use, now available on demand

Role-based ML learning paths for developers, data scientists, data platform engineers, and business decision makers

70+ free digital ML courses from AWS experts let you learn from real-world challenges tackled at AWS

Validate expertise with the
AWS Certified Machine Learning - Specialty exam

Visit https://aws.training/machinelearning

aws training and certification

Thank you!

AWS re:Invent

aws

Please complete the session survey in the mobile app.