aws SUMMIT ONLINE

INT01

AI/ML state of the nation

Denis V. Batalov

WW Tech Leader, AI/ML
Amazon Web Services

dbatalov

aws SUMMIT ONLINE

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

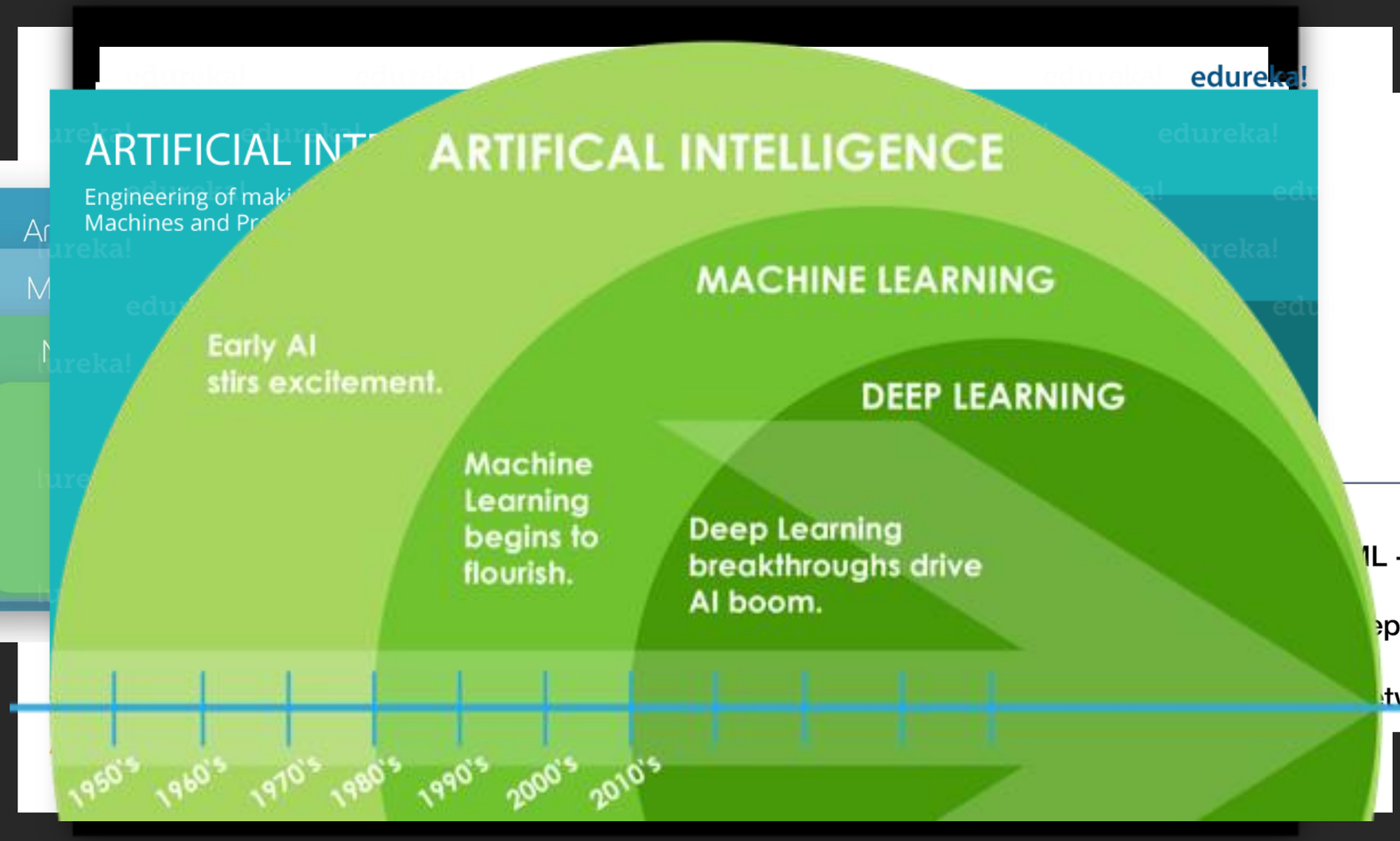
AI DEBATE : YOSHUA BENGIO | GARY MARCUS

Gary Marcus

——

Yoshua Bengio

AI, ML, Deep Learning?



AI - Artificial Inteligence

ML - Machine Learning

eep Learning

etworks

The AWS ML stack

Broadest and most complete set of machine learning capabilities

AI SERVICES

VISION	SPEECH		TEXT			SEARCH	CHATBOTS	PERSONALISATION	FORECASTING	FRAUD	DEVELOPMENT	CONTACT CENTERS
Amazon Rekognition *+Custom Labels*	Amazon Polly	Amazon Transcribe *+Medical*	Amazon Comprehend *+Medical*	Amazon Translate	Amazon Textract	Amazon Kendra	Amazon Lex	Amazon Personalize	Amazon Forecast	Amazon Fraud Detector	Amazon CodeGuru	Amazon Connect *with Contact Lens*

ML SERVICES

Amazon SageMaker

Ground Truth data labelling

ML Marketplace

SageMaker Studio IDE

Built-in algorithms	SageMaker Notebooks	SageMaker Experiments	Model tuning	SageMaker Autopilot	Model hosting	SageMaker Model Monitor

SageMaker Neo

ML FRAMEWORKS & INFRASTRUCTURE

TensorFlow

mxnet

PYTORCH

intel RL Coach

GLUON

Keras

HOROVOD

Deep Learning AMIs & Containers

GPUs & CPUs

Elastic Inference

Inferentia

FPGA

intel XEON PLATINUM inside

OpenVINO

intel inside

The AWS ML stack

Broadest and most complete set of machine learning capabilities

AI SERVICES

VISION
- Amazon Rekognition *+Custom Labels*

SPEECH
- Amazon Polly
- Amazon Transcribe *+Medical*

TEXT
- Amazon Comprehend *+Medical*
- Amazon Translate
- Amazon Textract

SEARCH
- Amazon Kendra

CHATBOTS
- Amazon Lex

PERSONALISATION
- Amazon Personalize

FORECASTING
- Amazon Forecast

FRAUD
- Amazon Fraud Detector

DEVELOPMENT
- Amazon CodeGuru

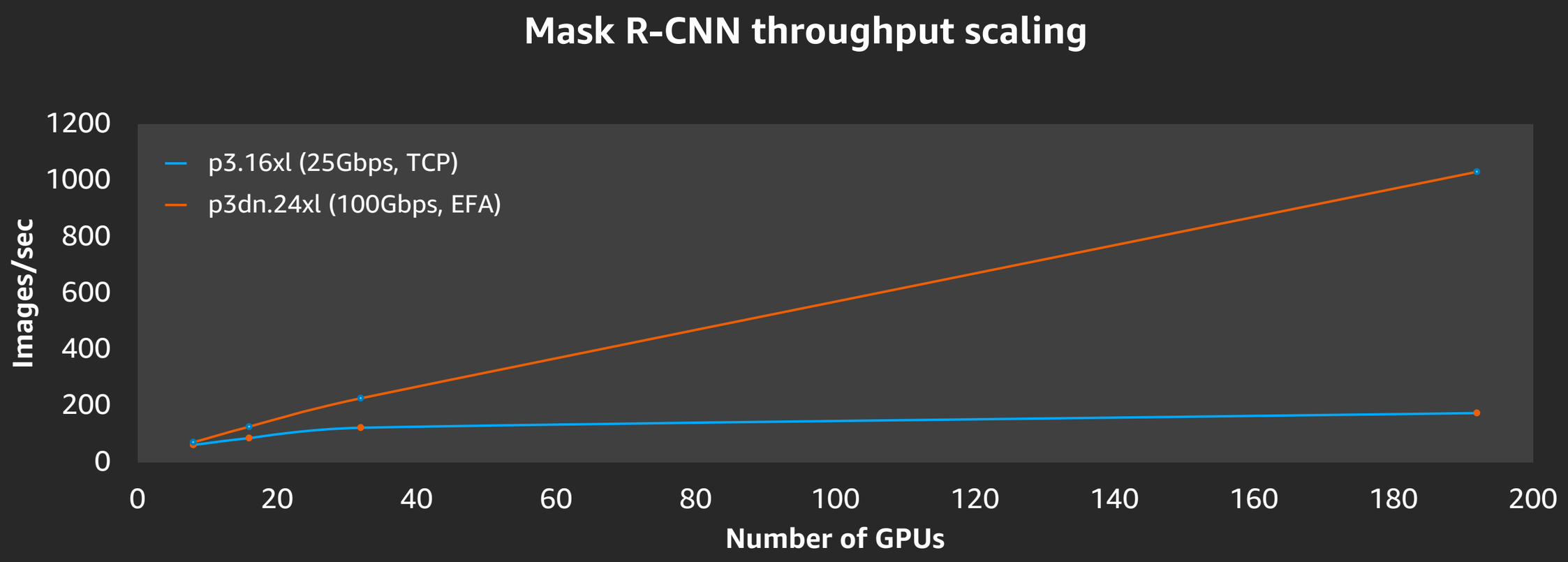
CONTACT CENTERS
- Amazon Connect *with Contact Lens*

ML SERVICES

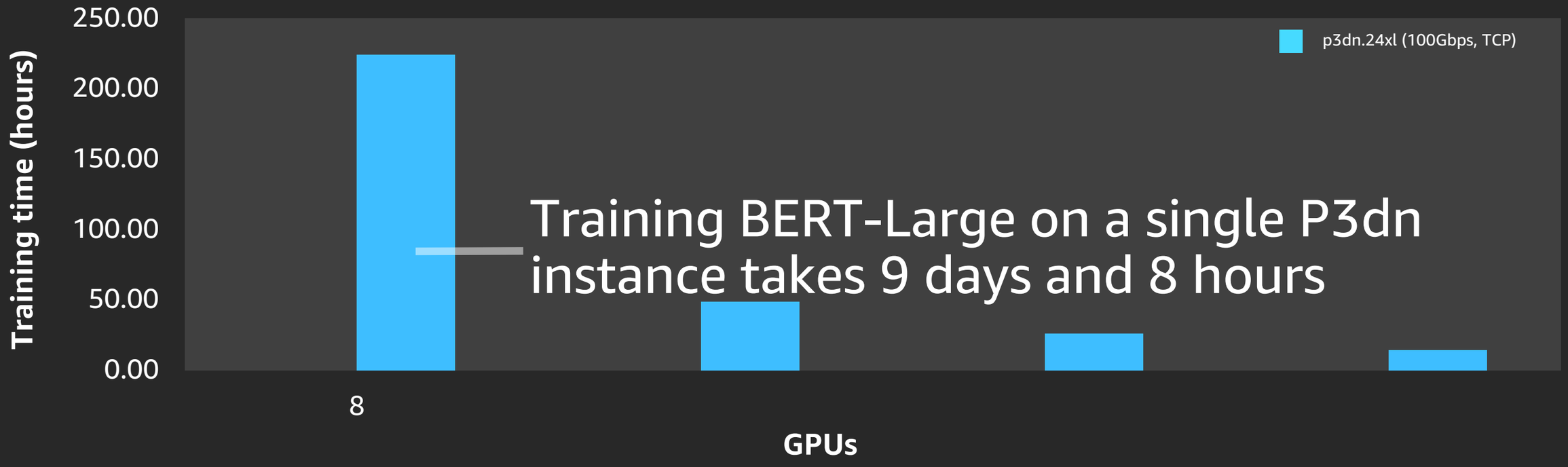
- Amazon SageMaker
- Ground Truth data labelling
- ML Marketplace
- SageMaker Studio IDE
  - Built-in algorithms
  - SageMaker Notebooks
  - SageMaker Experiments
  - Model tuning
  - SageMaker Autopilot
  - Model hosting
  - SageMaker Model Monitor
- SageMaker Neo

ML FRAMEWORKS & INFRASTRUCTURE

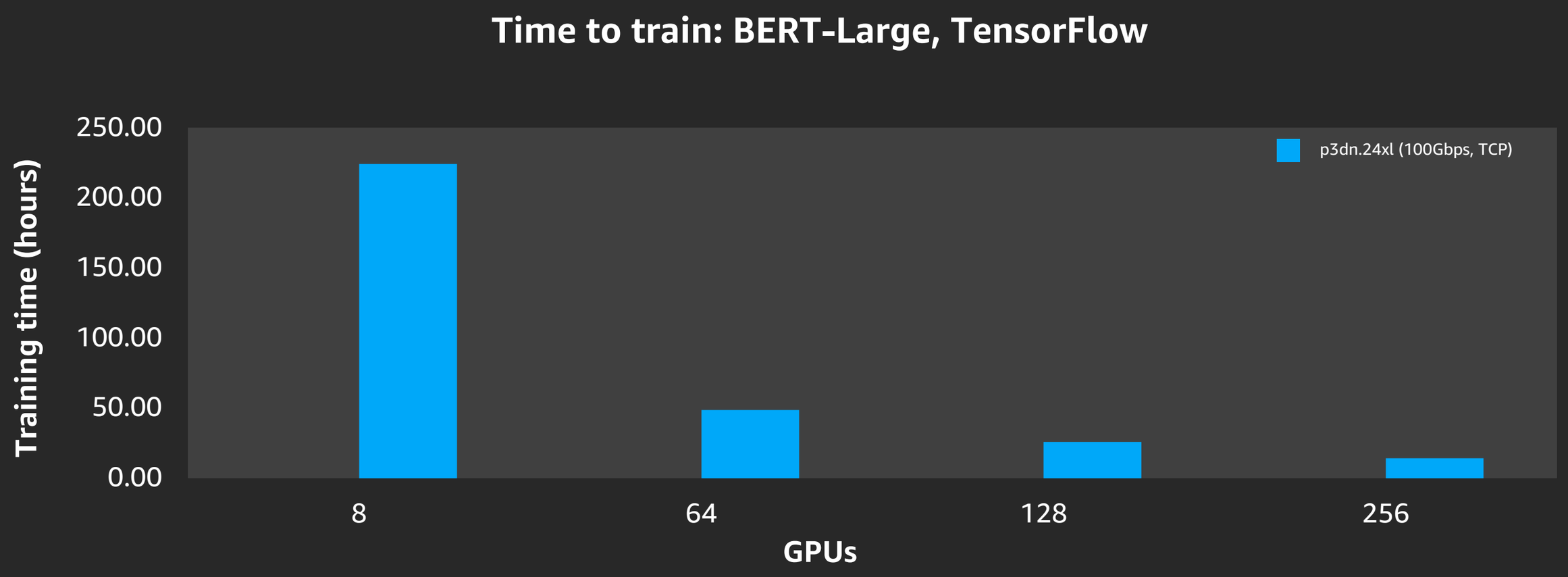
- TensorFlow
- mxnet
- PYTORCH
- intel RL Coach
- GLUON
- Keras
- HOROVOD
- Deep Learning AMIs & Containers
- GPUs & CPUs
- Elastic Inference
- Inferentia
- FPGA
- intel XEON PLATINUM inside
- OpenVINO
- intel inside

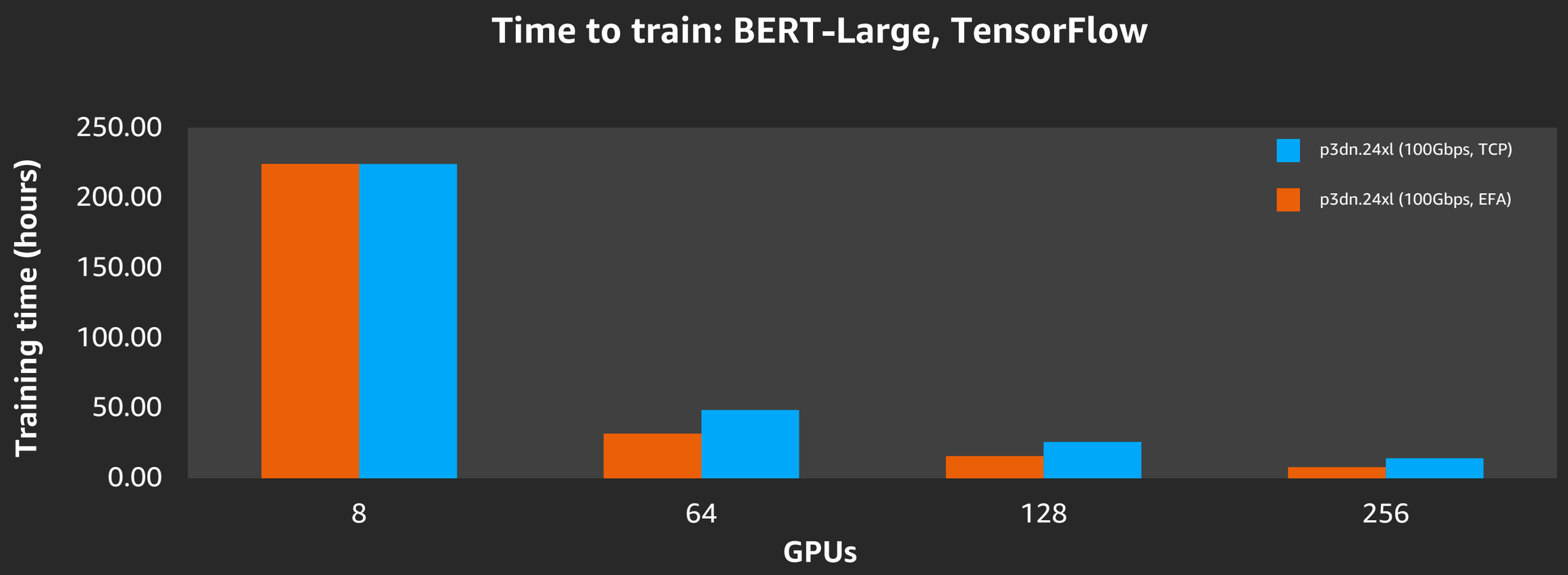

Mask R-CNN throughput scaling
p3.16xl (25Gbps, TCP)
p3dn.24xl (100Gbps, EFA)
Images/sec
1200
1000
800
600
400
200
0
0
20
40
60
80
100
120
140
160
180
200
Number of GPUs

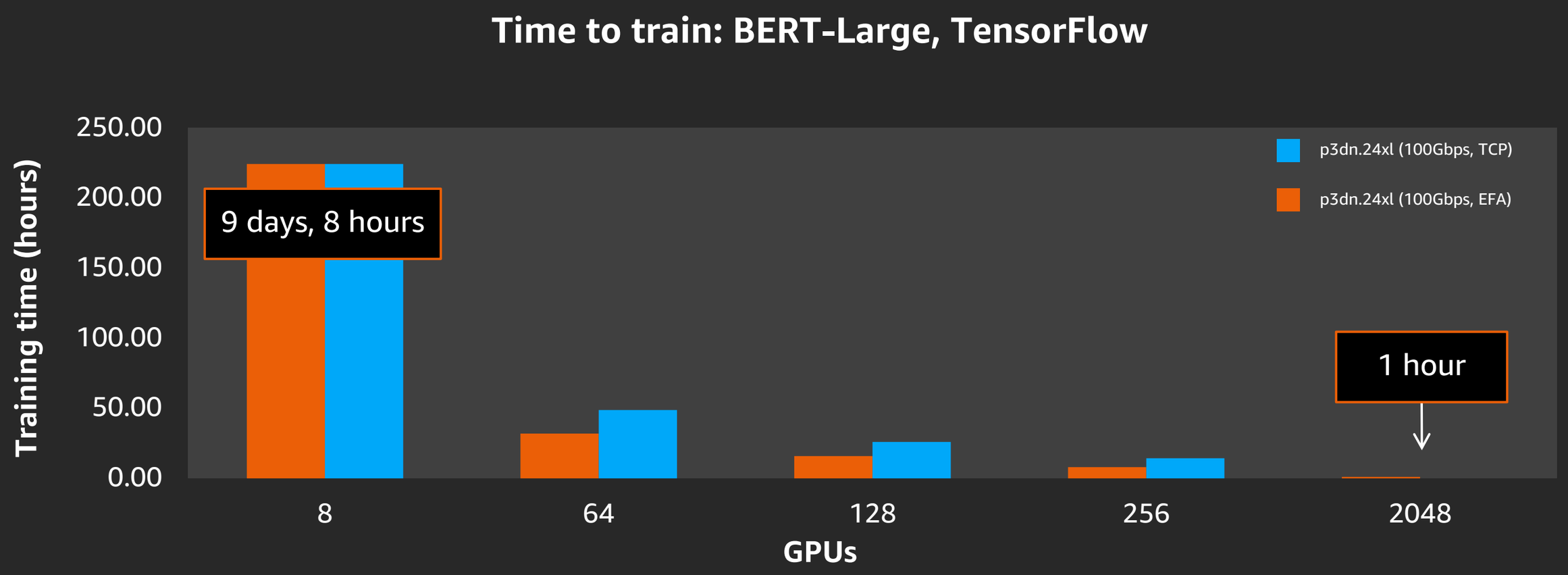
Time to train: BERT-Large, TensorFlow

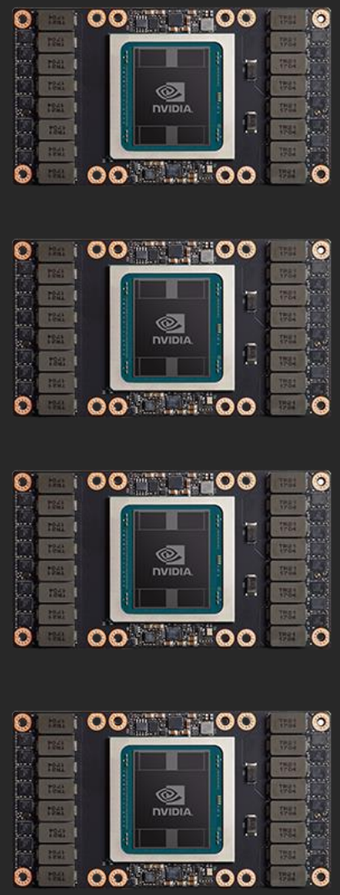


Training BERT-Large on a single P3dn instance takes 9 days and 8 hours


Time to train: BERT-Large, TensorFlow
Training time (hours)
250.00
200.00
150.00
100.00
50.00
0.00
p3dn.24xl (100Gbps, TCP)
8
64
128
256
GPUs


Time to train: BERT-Large, TensorFlow
Training time (hours)
250.00
200.00
150.00
100.00
50.00
0.00
p3dn.24xl (100Gbps, TCP)
p3dn.24xl (100Gbps, EFA)
8
64
128
256
GPUs


Time to train: BERT-Large, TensorFlow
Training time (hours)
250.00
200.00
150.00
100.00
50.00
0.00
9 days, 8 hours
1 hour
p3dn.24xl (100Gbps, TCP)
p3dn.24xl (100Gbps, EFA)
8
64
128
256
2048
GPUs



Highest performance for compute and graphics workstations

EC2 P3

NVIDIA® Tesla® V100

The AWS ML stack

Broadest and most complete set of machine learning capabilities

AI SERVICES

VISION	SPEECH		TEXT			SEARCH	CHATBOTS	PERSONALISATION	FORECASTING	FRAUD	DEVELOPMENT	CONTACT CENTERS
Amazon Rekognition *+Custom Labels*	Amazon Polly	Amazon Transcribe *+Medical*	Amazon Comprehend *+Medical*	Amazon Translate	Amazon Textract	Amazon Kendra	Amazon Lex	Amazon Personalize	Amazon Forecast	Amazon Fraud Detector	Amazon CodeGuru	Amazon Connect *with Contact Lens*

ML SERVICES

Amazon SageMaker

Ground Truth data labelling

ML Marketplace

SageMaker Studio IDE

Built-in algorithms | SageMaker Notebooks | SageMaker Experiments | Model tuning | SageMaker Autopilot | Model hosting | SageMaker Model Monitor

SageMaker Neo

new

ML FRAMEWORKS & INFRASTRUCTURE

TensorFlow | mxnet | PYTORCH

GLUON | Keras | HOROVOD

Deep Learning AMIs & Containers | GPUs & CPUs | Elastic Inference | Inferentia | FPGA

NEW

Introducing Amazon SageMaker Studio

The first fully integrated development environment (IDE) for machine learning

Collaboration at scale

Share notebooks without tracking code dependencies

Easy experiment management

Organise, track, and compare thousands of experiments

Automatic model generation

Get accurate models with full visibility & control without writing code

Higher quality ML models

Automatically debug errors, monitor models, & maintain high quality

Increased productivity

Code, build, train, deploy, & monitor in a unified visual interface

Amazon SageMaker Studio File Edit View Run Kernel Git Tabs Settings Help

xgboost_customer_churn.ipyr

Markdown conda_amazonei_mxnet_p27

- Have the predictor variable in the first column
- Not have a header row

But first, let's convert our categorical features into numeric features.

[ ]:
```
model_data = pd.get_dummies(churn)
model_data = pd.concat([model_data['Churn?_True.'], model_data.drop(['Chur
```

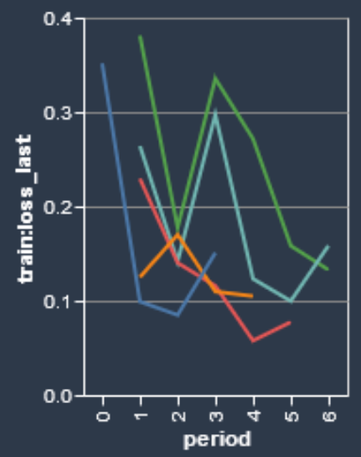
And now let's split the data into training, validation, and test sets. This will help prevent us from overfitting the model, and allow us to test the models accuracy on data it hasn't already seen.

[ ]:
```
train_data, validation_data, test_data = np.split(model_data.sample(frac=1
train_data.to_csv('train.csv', header=False, index=False)
validation_data.to_csv('validation.csv', header=False, index=False)
```

Now we'll upload these files to S3.

[ ]:
```
boto3.Session().resource('s3').Bucket(bucket).Object(os.path.join(prefix,
boto3.Session().resource('s3').Bucket(bucket).Object(os.path.join(prefix,
```

Trial Component Chart



Trial Component List

TRIAL COMPONENTS

10 rows selected

Add chart Deploy model

Status	Experiment	Type	Trial	Trial c
Completed	customer-churn-predi...	Training job	Trial-3	Tra
Completed	customer-churn-predi...	Training job	Trial-2	Tra
Completed	customer-churn-predi...	Training job	Trial-1	Tra
Completed	customer-churn-predi...	Training job	Trial-0	Tra

0 2 conda_amazonei_mxnet_p27 | Idle Mode: Command Ln 1, Col 1 xgboost_customer_churn.ipynb

Data science and collaboration needs to be easy

Setup and manage resources

+

Collaboration across multiple data scientists

+

Different data science projects have different resource needs

=

Managing notebooks and collaborating across multiple data scientists is highly complicated

NEW

Introducing Amazon SageMaker Notebooks

(Available in Preview)

Fast-start shareable notebooks

Easy access with Single Sign-On (SSO)

Access your notebooks in seconds with your corporate credentials

Fully managed and secure

Administrators manage access and permissions

No explicit setup

Start your notebooks without spinning up compute resources

Easy collaboration

Share your notebooks as a URL with a single click

Flexibility

Dial up or down compute resources (Coming soon)

SageMaker Studio

sagemaker.aws.amazon.com

File Edit View Run Kernel Tabs Settings Help

Launcher | my_notebook.ipynb

Markdown git conda_pytorch_p36

The following shell code shows how to build the container image using `docker build` and push the container image to ECR using `docker push`. This code is also available as the shell script `container/build-and-push.sh`, which you can run as `build-and-push.sh pytorch-extending-our-containers-cifar10-example` to build the image `pytorch-extend`

This code looks for an ECR repository in the acco a SageMaker notebook instance, this is the region where the notebook instance was created). If the reposi are using the SageMaker PyTorch image as the base, we will need to retrieve ECR credentials to pull this p

Create shareable snapshot

Share as

my_notebook.ipynb

Include Git repo information

Include notebook output

Cancel Create snapshot

Sharing notebook: MyNotebook.ipynb

[ ]:
```
%%sh

# The name of our algorithm
algorithm_name=pytorch-extending-our-co

cd container

account=$(aws sts get-caller-identity --query Account --output text)

# Get the region defined in the current configuration (default to us-west-2 if none defined)
region=$(aws configure get region)
region=${region:-us-west-2}

fullname="${account}.dkr.ecr.${region}.amazonaws.com/${algorithm_name}:latest"

# If the repository doesn't exist in ECR, create it.

aws ecr describe-repositories --repository-names "${algorithm_name}" > /dev/null 2>&1

if [ $? -ne 0 ]
then
    aws ecr create-repository --repository-name "${algorithm_name}" > /dev/null
fi

# Get the login command from ECR and execute it directly
$(aws ecr get-login --region ${region} --no-include-email)

# Get the login command from ECR in order to pull down the SageMaker PyTorch image
$(aws ecr get-login --registry-ids 520713654638 --region ${region} --no-include-email)
```

Managing trials and experiments is cumbersome

Thousands of experiments

+

Hundreds of parameters per experiment

+

Compare and evaluate

=

Very cumbersome and error prone

NEW

Introducing Amazon SageMaker Experiments

Organise, track, and compare training experiments

Tracking at scale

Track parameters & metrics across experiments & users

Custom organisation

Organise experiments by teams, goals, & hypotheses

Visualisation

Easily visualise experiments and compare

Metrics and logging

Log custom metrics using the Python SDK & APIs

Fast Iteration

Quickly go back & forth & maintain high-quality

Amazon SageMaker Studio File Edit View Run Kernel Git Tabs Settings Help

Trial Component List | Trial Component Chart

TRIAL COMPONENTS 5 rows selected. Select rows to toggle chart visibility.

Add Chart

	Experiment	Trial	Trial Component	Type	Training time	Actions
	customer-churn-predi...	Trial-9	Training-Run-9-aws-training-job	Training job	~6 minutes	Remove
	customer-churn-predi...	Trial-8	Training-Run-8-aws-training-job	Training job	~5 minutes	Remove
	customer-churn-predi...	Trial-7	Training-Run-7-aws-training-job	Training job	~6 minutes	Remove
	customer-churn-predi...	Trial-6	Training-Run-6-aws-training-job	Training job	~4 minutes	Remove
	customer-churn-predi...	Trial-5	Training-Run-5-aws-training-job	Training job	~4 minutes	Remove

1 CHART

train:loss_last with 1-minute aggregation

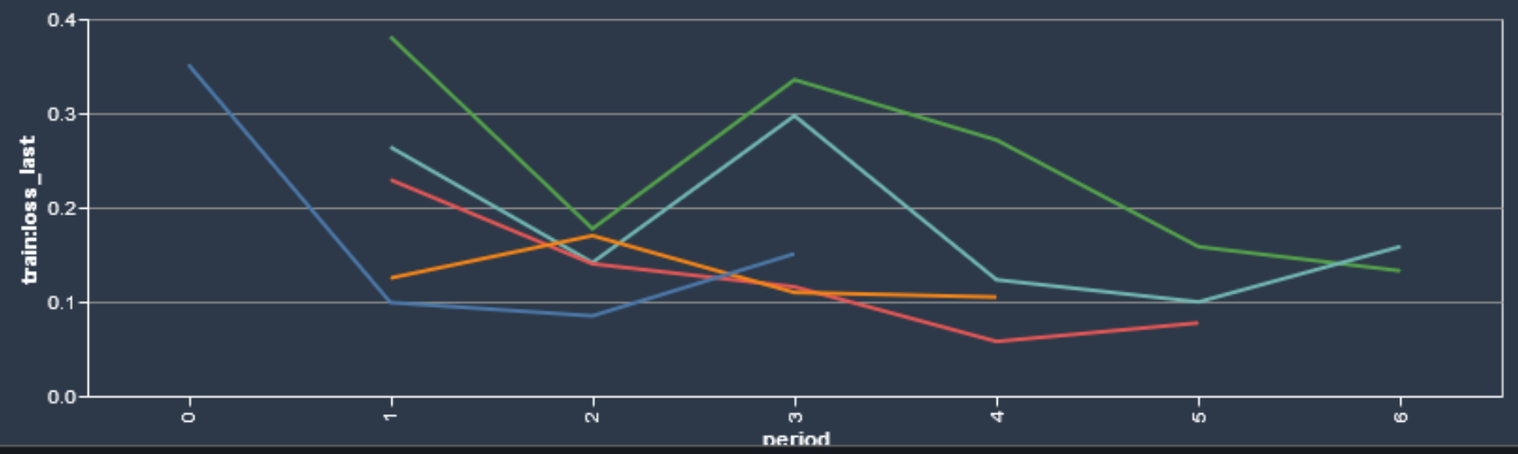


CHART PROPERTIES

Data type
- Time series
- Summary statistics

Chart type
- Histogram
- Line

X-axis dimension
- Epoch
- Time
- Periods from start

X-axis aggregation
- 1-minute
- 5-minute
- 60-minute

Y-axis

train:loss_last ▼

2 $_ 0

Trial Component Chart

Debugging and profiling deep learning is painful

Large neural networks
with many layers

+

Data capture with many
connections

+

Additional tooling for analysis
and debug

=

Extraordinarily difficult
to inspect, debug, and profile
the ‘black box’

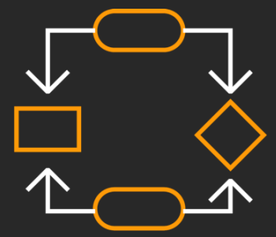
NEW

Introducing Amazon SageMaker Debugger

Analysis and debugging, explainability, and alert generation

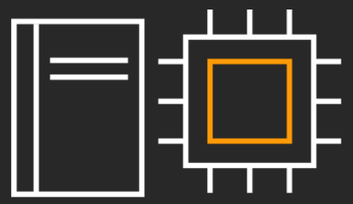
Relevant data capture

Data is automatically captured for analysis



Data analysis & debugging

Analyse & debug data with no code changes



Automatic error detection

Errors are automatically detected based on rules

Improved productivity with alerts

Take corrective action based on alerts

Visual analysis and debugging

Visually analyse & debug from SageMaker Studio

Amazon SageMaker Studio File Edit View Run Kernel Git Tabs Settings Help

SMDEbugger-CloudWatch-Lo

Markdown git conda_tensorflow_p36

Using SageMaker Rules

In this example we'll demonstrate how to use SageMaker rules to be evaluated against your training. You can find the list of SageMaker rules and the configurations best suited for using them here.

We specify a few rules that check for overfitting, descrease in loss across epochs and for saturated activations.

[8]:
```
estimator = TensorFlow(
    role=sagemaker.get_execution_role(),
    base_job_name='mnist-tensorflow-example',
    train_instance_count=1,
    train_instance_type='ml.p3.2xlarge',
    image_name=cpu_training_image,
    entry_point=entrypoint_script,
    framework_version='1.15',
    py_version='py3',
    train_max_run=3600,
    script_mode=True,
    sagemaker_session=sess,
    ## New parameter
    rules = [ Rule.sagemaker(rule_configs.vanishing_gradient()),
              Rule.custom(name='Overfitting', # used to identify the rule
                          image_uri='759209512951.dkr.ecr.us-west-2.amazonaws.com/
                          instance_type='ml.c4.xlarge', # instance type to run the
                          source='my_custom_rule.py', # path to the rule source f
                          rule_to_invoke='CustomGradientRule', # name of the clas
                          volume_size_in_gb=400, # EBS volume size required to be
                          collections_to_save=[CollectionConfig(name='losses')],
                          rule_parameters={
                              "threshold": "20.0" # this will be used to intializ
                          }) ],
    hyperparameters = {'num_epochs' : 100 }
)
```

Note that Sagemaker-Debugger is only supported for py_version='py3' currently.

Let's start the training by calling `fit()` on the MXNet estimator

[9]:
```
# After calling fit, SageMaker will spin off 1 training job and 1 rule job for y
# The rule evaluation status(es) will be visible in the training logs
# at regular intervals

estimator.fit(wait=False)
```

Result

Describe Trial Component

Experiment: Unassigned

Trial: Unassigned

Trial stages

mnist-tensorflow-example-2019-12-02-09-52-13-126-aws-training-job
Created
15 minutes ago

Charts | Metrics | Parameters | Artifacts | AWS Settings | Debugger

Status	Last modified	Rule name	Job ARN
Issues Found	4 minutes ago	VanishingGradient	arn:aws:sagemaker:us-west-2:3
Issues Found	4 minutes ago	Overfitting	arn:aws:sagemaker:us-west-2:3

Trial Component Chart

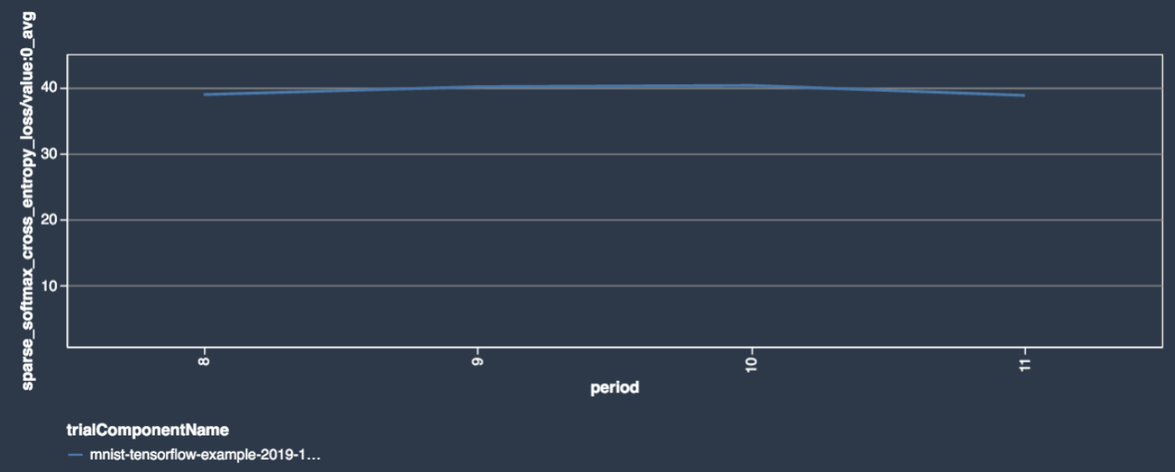
TRIAL COMPONENTS 1 rows selected. Select rows to toggle chart visibility.

Add Chart

Experiment	Trial	Trial Component	Type	Train
N/A	N/A	mnist-tensorflow-example-2019-12-02-09-52-13-126-aws-trainin...	Training job	~10

2 CHARTS

sparse_softmax_cross_entropy_loss/value:0_avg with 1-minute aggregation



0 4 Describe Trial Component

Deploying a model is not the end. You need to continuously monitor models in production and iterate

Concept drift due to divergence of data

+

Model performance can change due to unknown factors

+

Continuous monitoring involves a lot of tooling and expense

=

Model monitoring is cumbersome but critical

NEW

Introducing Amazon SageMaker Model Monitor
Continuous monitoring of models in production

Automatic data collection

Data is automatically collected from your endpoints

Continuous Monitoring

Define a monitoring schedule and detect changes in quality against a pre-defined baseline

Flexibility with rules

Use built-in rules to detect data drift or write your own rules for custom analysis

Visual Data analysis

See monitoring results, data statistics, and violation reports in SageMaker Studio

CloudWatch Integration

Automate corrective actions based on Amazon CloudWatch alerts

Amazon SageMaker Studio File Edit View Run Kernel Git Tabs Settings Help

ENDPOINTS

Name	Endpoint status
AutoML-toms-super-notebo...	InService
UC-DEMO-xgb-churn-pred-...	InService
demo-xgboost-customer-ch...	InService
mengyw-test-form-1	InService
mengyw-test-capture	InService
my-sagemaker-ap1	InService
mengyw-test-vpc-3	Failed
DEMO-xgb-churn-pred-mod...	InService
mengyw-test-vpc-2	InService
mengyw-test-vpc	Failed

Amazon S | UC-DEMO | UC-DEMO | model-mo | SageMake | model-mo | DEMO-xgt | model-mo

Monitoring results | Monitoring job history | AWS settings

AMAZON SAGEMAKER MODEL MONITOR

Amazon SageMaker Model Monitor detects data drift and other issues that can affect models in production and alerts you so you can take corrective action.

2 CHARTS

Add chart

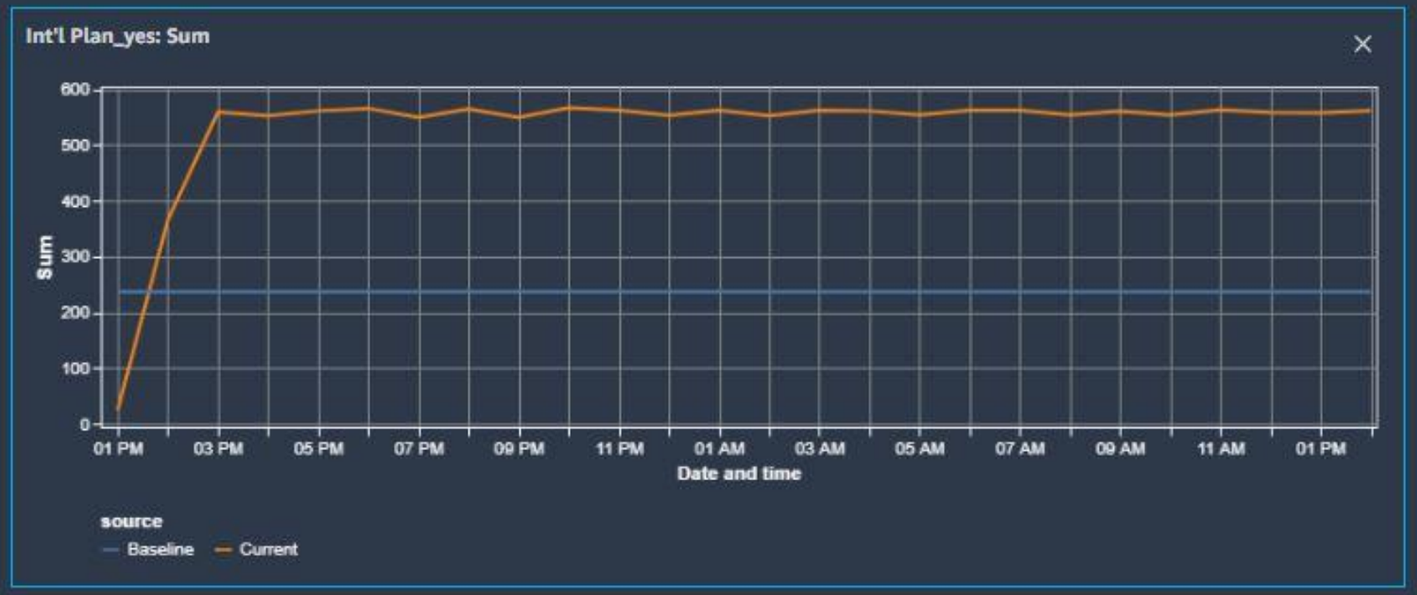


CHART PROPERTIES

Timeline
- 1 week
- 1 day
- 12 hours
- 6 hours
- 1 hour

Statistic
- Average
- SampleCount
- Sum
- Minimum
- Maximum

Feature

feature_data_Int'l Plan_yes

Successful ML requires complex, hard to discover combinations

of algorithms, data, parameters

Largely explorative & iterative

+

Requires broad and complete knowledge of ML domain

+

Lack of visibility

=

Time consuming, error prone process, even for ML experts

NEW

Introducing Amazon SageMaker Autopilot
Automatic model creation with full visibility and control

Quick to start

Provide your data in a tabular form & specify target prediction

Automatic model creation

Get ML models with feature engineering & automatic model tuning automatically done

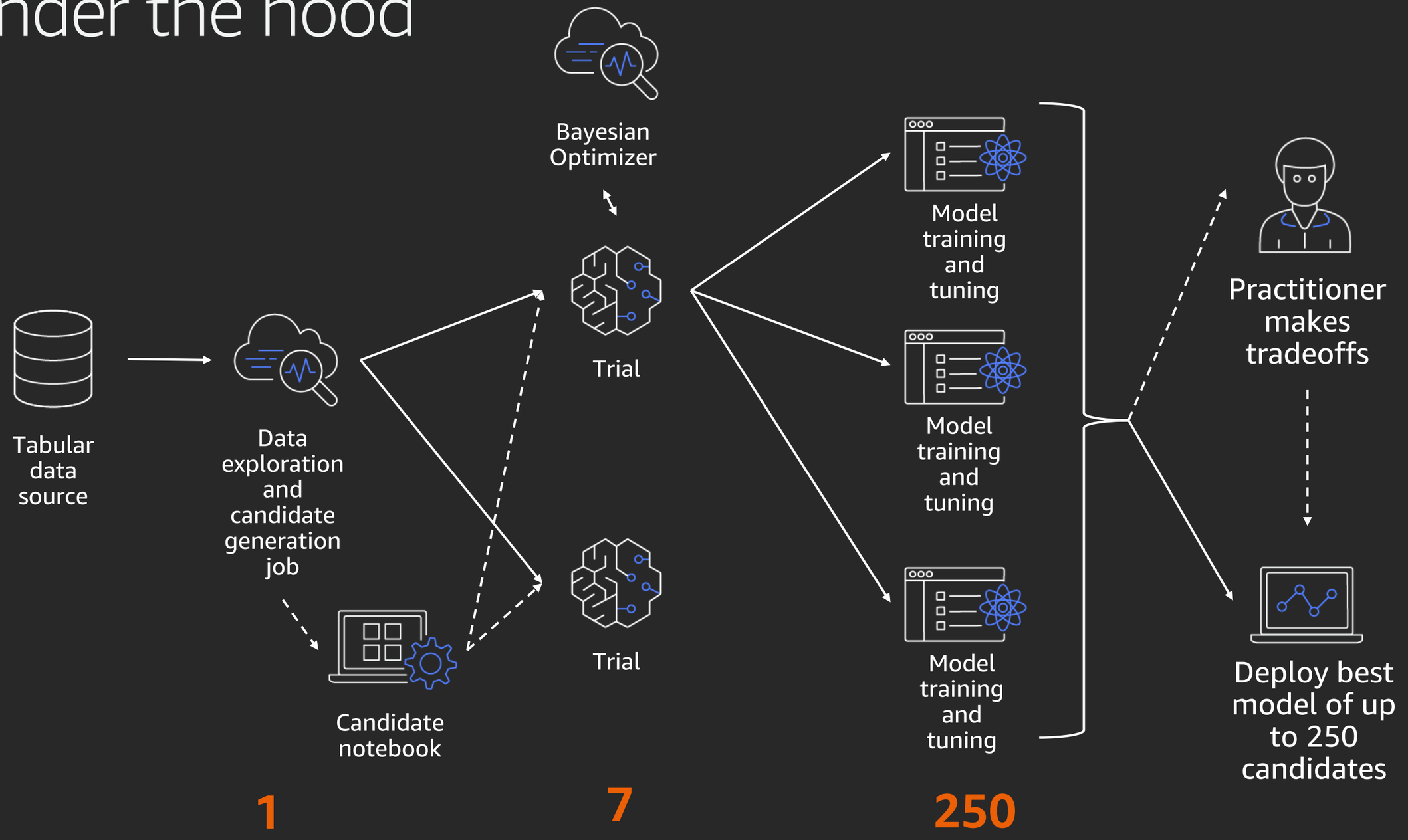
Visibility & control

Get notebooks for your models with source code

Recommendations & Optimisation

Get a leaderboard & continue to improve your model

Under the hood



Model training involves tradeoffs

#	Model	Accuracy	Latency	Model Size
1	churn-xgboost-1756-013-33398f0	95%	450ms	9.1MB
2	churn-xgboost-1756-014-53facc2	93%	200ms	4.8MB
3	churn-xgboost-1756-015-58bc692	92%	200ms	4.5MB
4	churn-linear-1756-016-db54598	91%	50ms	1.3MB
5	churn-xgboost-1756-017-af8d756	91%	190ms	4.2MB

The AWS ML stack

Broadest and most complete set of machine learning capabilities

AI SERVICES

VISION	SPEECH		TEXT			SEARCH	CHATBOTS	PERSONALISATION	FORECASTING	FRAUD	DEVELOPMENT	CONTACT CENTERS
Amazon Rekognition *+Custom Labels*	Amazon Polly	Amazon Transcribe *+Medical*	Amazon Comprehend *+Medical*	Amazon Translate	Amazon Textract	Amazon Kendra	Amazon Lex	Amazon Personalize	Amazon Forecast	Amazon Fraud Detector	Amazon CodeGuru	Amazon Connect *with Contact Lens*

ML SERVICES

Amazon SageMaker

Ground Truth data labelling

ML Marketplace

SageMaker Studio IDE

Built-in algorithms	SageMaker Notebooks	SageMaker Experiments	Model tuning	SageMaker Autopilot	Model hosting	SageMaker Model Monitor

SageMaker Neo

ML FRAMEWORKS & INFRASTRUCTURE

TensorFlow

mxnet

PYTORCH

GLUON

Keras

Deep Learning AMIs & Containers | GPUs & CPUs | Elastic Inference | Inferentia | FPGA

Amazon Polly: Brand voice

Colonel Sanders

Customer service

The AWS ML stack

Broadest and most complete set of machine learning capabilities

AI SERVICES

VISION	SPEECH		TEXT			SEARCH	CHATBOTS	PERSONALISATION	FORECASTING	FRAUD	DEVELOPMENT	CONTACT CENTERS
Amazon Rekognition *+Custom Labels*	Amazon Polly	Amazon Transcribe *+Medical*	Amazon Comprehend *+Medical*	Amazon Translate	Amazon Textract	Amazon Kendra	Amazon Lex	Amazon Personalize	Amazon Forecast	Amazon Fraud Detector	Amazon CodeGuru	Amazon Connect *with Contact Lens*

ML SERVICES

Amazon SageMaker

Ground Truth data labelling

ML Marketplace

SageMaker Studio IDE

Built-in algorithms	SageMaker Notebooks	SageMaker Experiments	Model tuning	SageMaker Autopilot	Model hosting	SageMaker Model Monitor

SageMaker Neo

ML FRAMEWORKS & INFRASTRUCTURE

TensorFlow

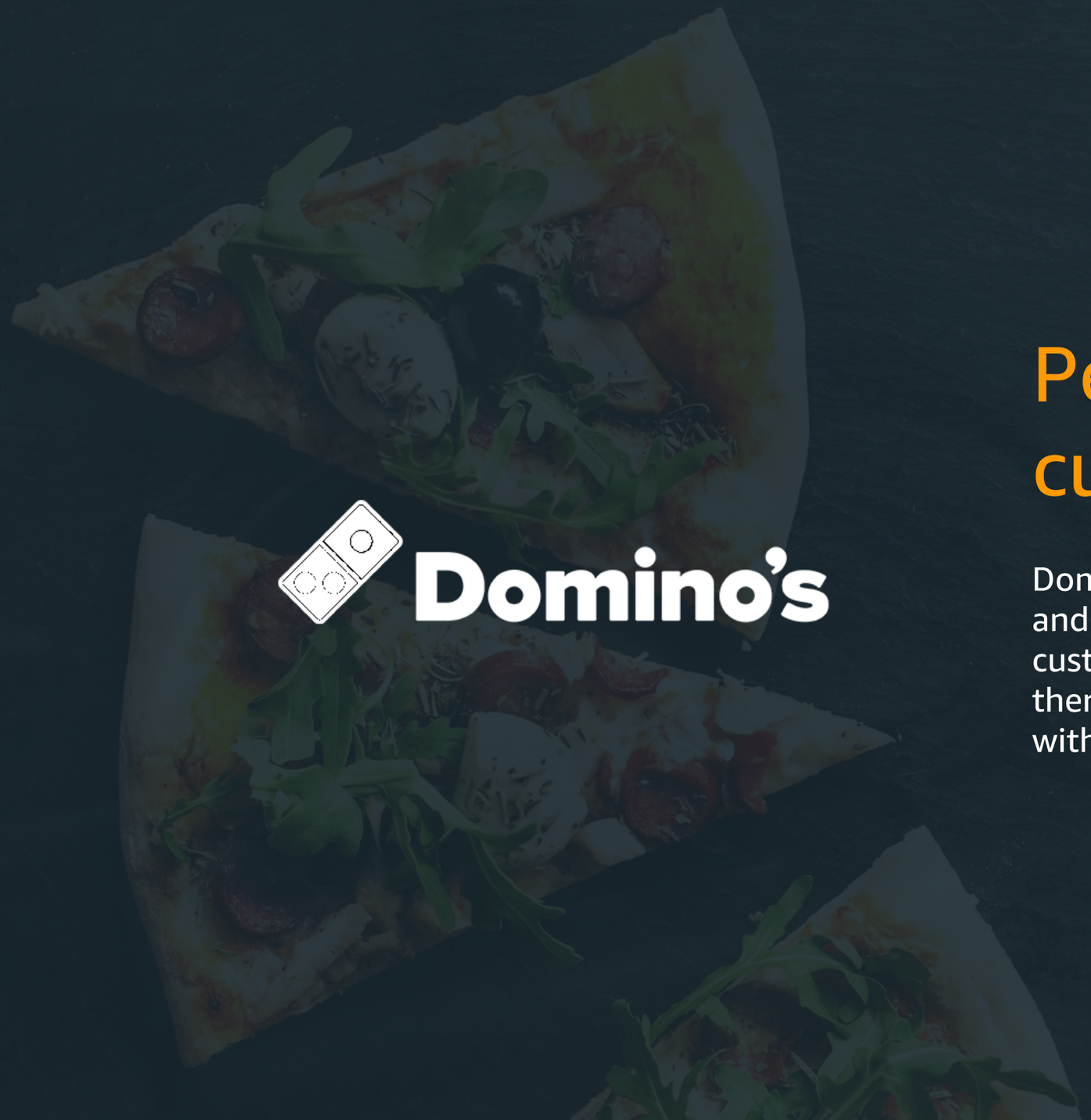
mxnet

PYTORCH

GLUON

Keras

Deep Learning AMIs & Containers | GPUs & CPUs | Elastic Inference | Inferentia | FPGA



Personalising customer experiences

Domino's uses Amazon Personalize to customise and scale relevant marketing communications to customers based on time, context, and content, thereby improving and enhancing their experience with the Domino's brand.

The AWS ML stack

Broadest and most complete set of machine learning capabilities

AI SERVICES

VISION
- Amazon Rekognition *+Custom Labels* *new*

SPEECH
- Amazon Polly
- Amazon Transcribe *+Medical* *new*

TEXT
- Amazon Comprehend *+Medical*
- Amazon Translate
- Amazon Textract

SEARCH
- Amazon Kendra *new*

CHATBOTS
- Amazon Lex

PERSONALISATION
- Amazon Personalize

FORECASTING
- Amazon Forecast

FRAUD
- Amazon Fraud Detector *new*

DEVELOPMENT
- Amazon CodeGuru *new*

CONTACT CENTERS
- Amazon Connect *with Contact Lens* *new*

ML SERVICES

- Amazon SageMaker
- Ground Truth data labelling
- ML Marketplace
- SageMaker Studio IDE
  - Built-in algorithms
  - SageMaker Notebooks
  - SageMaker Experiments
  - Model tuning
  - SageMaker Autopilot
  - Model hosting
  - SageMaker Model Monitor
- SageMaker Neo

ML FRAMEWORKS & INFRASTRUCTURE

- TensorFlow
- mxnet
- PYTORCH
- GLUON
- Keras
- Deep Learning AMIs & Containers
- GPUs & CPUs
- Elastic Inference
- Inferentia
- FPGA

Cerner®

Amazon Transcribe Medical

An automatic speech recognition (ASR) service that enables developers to add medical speech-to-text capabilities to their voice-enabled applications.

Accurate

US English

Primary Care

Dictation Transcription

Conversational Transcription

Easy-to-Use

Real-time Public API

Automatic Punctuation

Word-level Time Stamps

Word-level Confidence Scores

Affordable

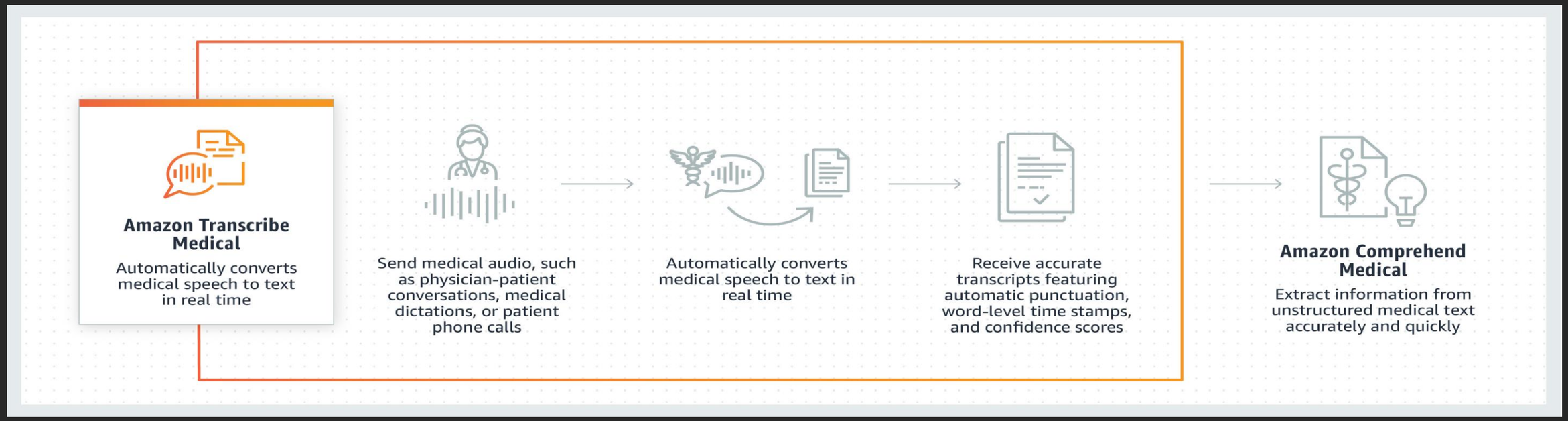
Pay-as-you-go Model

Charge by Transcription Usage

7.5 cents / minute

60 Minute Free Tier

AI/ML in Healthcare



The AWS ML stack

Broadest and most complete set of machine learning capabilities

AI SERVICES

VISION	SPEECH		TEXT			SEARCH	CHATBOTS	PERSONALISATION	FORECASTING	FRAUD	DEVELOPMENT	CONTACT CENTERS
Amazon Rekognition *+Custom Labels*	Amazon Polly	Amazon Transcribe *+Medical*	Amazon Comprehend *+Medical*	Amazon Translate	Amazon Textract	Amazon Kendra	Amazon Lex	Amazon Personalize	Amazon Forecast	Amazon Fraud Detector	Amazon CodeGuru	Amazon Connect *with Contact Lens*

ML SERVICES

Amazon SageMaker

Ground Truth data labelling

ML Marketplace

SageMaker Studio IDE

Built-in algorithms	SageMaker Notebooks	SageMaker Experiments	Model tuning	SageMaker Autopilot	Model hosting	SageMaker Model Monitor

SageMaker Neo

ML FRAMEWORKS & INFRASTRUCTURE

TensorFlow

mxnet

PYTORCH

GLUON

Keras

HOROVOD

Deep Learning AMIs & Containers | GPUs & CPUs | Elastic Inference | Inferentia | FPGA

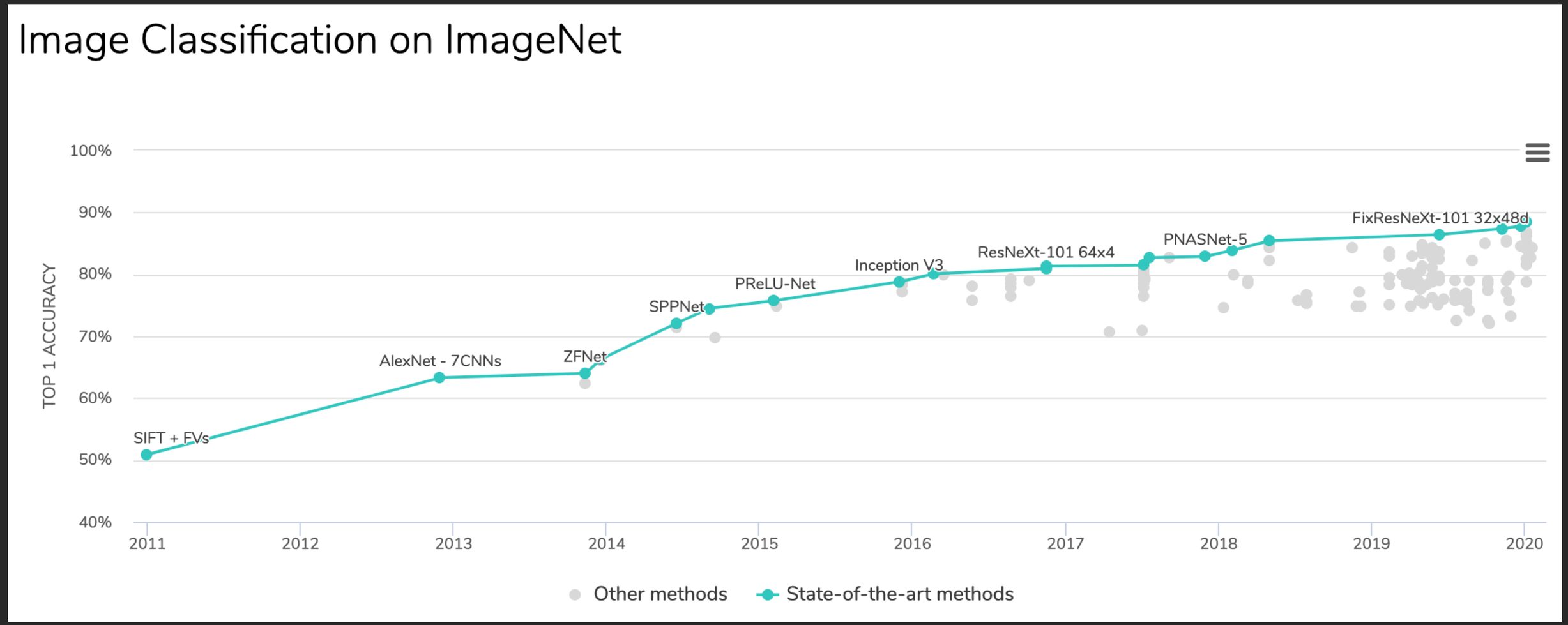
AI/ML Trends

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

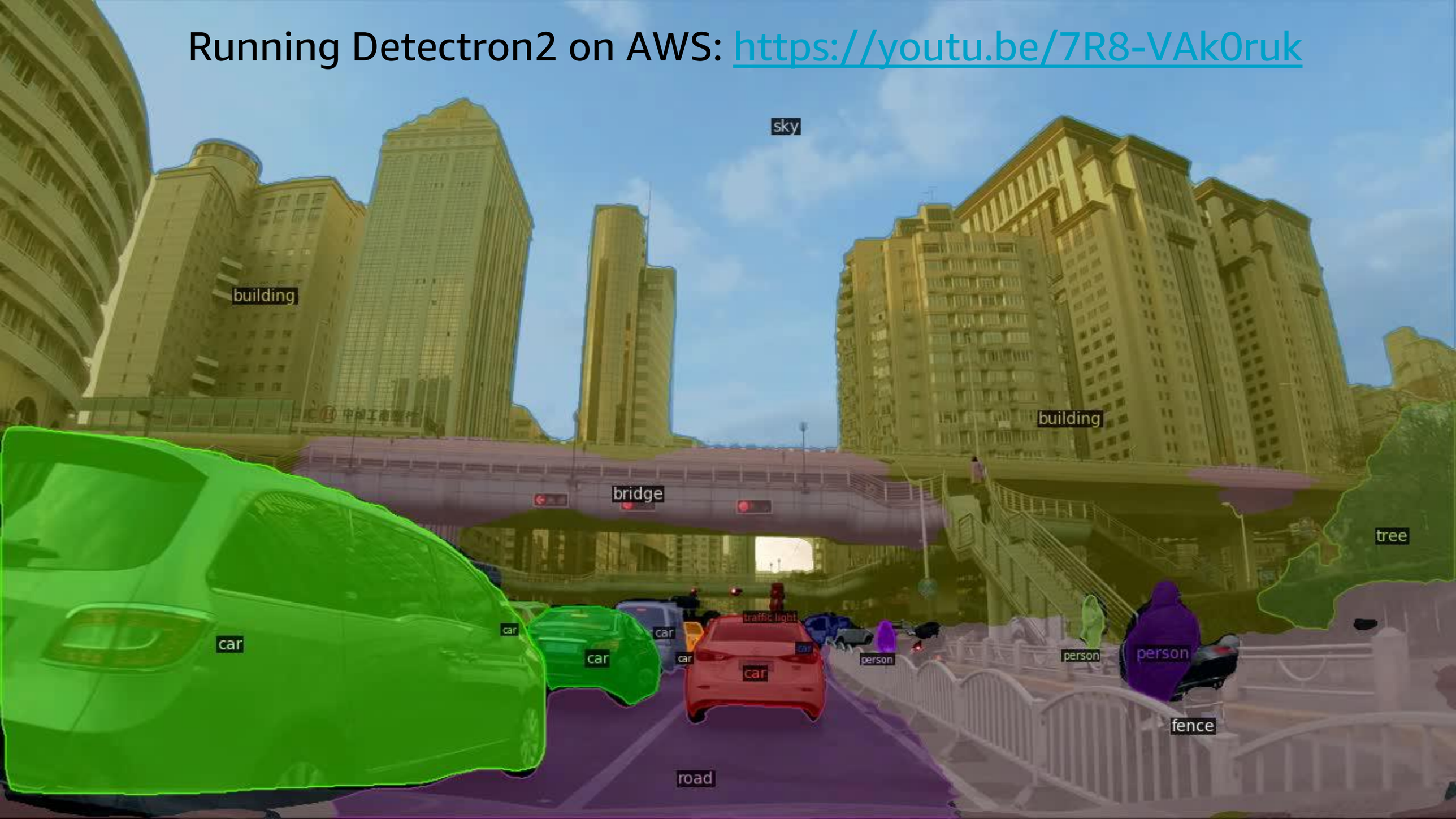
Better and faster models

Object detection: Mask R-CNN, YOLOv2

NLP: ULMFiT, BERT, GPT-2



Running Detectron2 on AWS: https://youtu.be/7R8-VAk0ruk



Reinforcement learning

Reinforcement learning is based on the reward hypothesis:

All goals can be described by the maximisation of an expected cumulative reward

AWS DeepRacer Evo

- 1:18 4WD scale car
- Intel Atom processor
- Intel distribution of OpenVINO toolkit
- Stereo Camera (4MP)
- 360 Degree 12 Meters Scanning Radius LIDAR Sensor
- System memory: 4 GB RAM
- 802.11ac Wi-Fi
- Ubuntu 16.04.3 LTS
- ROS Kinetic

Customise your agent's sensors in the Garage

AWS DeepRacer > Reinforcement learning > Garage

Garage

Create model | Build new vehicle

The garage shows the DeepRacer vehicles that you can train models for. You can add vehicles by using the "build new vehicle button"

Evo

Mod vehicle

Sensor

Lidar
Stereo cameras

Neural network topology

DCN Shallow

Action space

Speed: 4 m/s Steering Angle: 30°

Mod your own vehicle

Mod specifications

The garage shows the DeepRacer vehicles that you can train models for. You can add vehicles by using the "build new vehicle button"

Sensor modification

Swap sensors to improve your DeepRacer's racing performance

- Front-facing camera

  Single camera that captures the images with sizes of 160 x 120 in front of the agent at 15 fps. The camera has 120 wide angle lens. The images are converted into grey scale before being fed to the neural network

  ▶ Benefits of the front-facing camera

- Stereo cameras (right/left) sensor

  Composed of two front-facing cameras, stereo cameras can generate depth information of the objects in front of the agent and thus be used to detect and avoid obstacles on the track. The cameras capture images with the same resolution and frequency. Images from both cameras are converted into grey scale, stacked and then fed into the neural network.

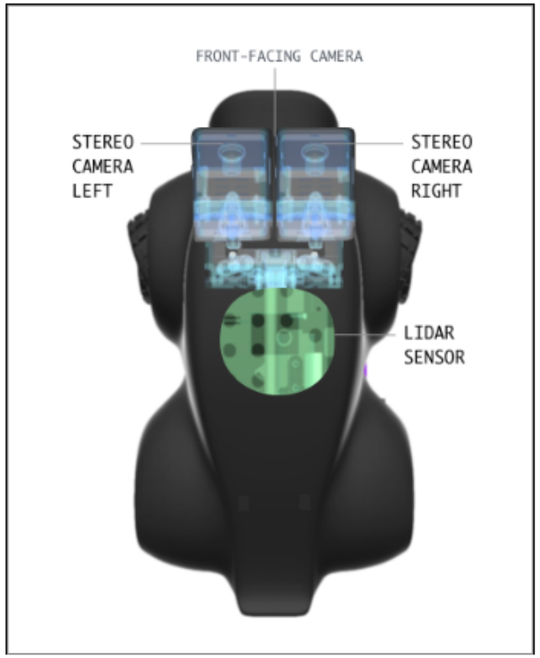
  ▶ Benefits of the stereo camera

Add-on sensors

- [x] LIDAR sensor

  LIDAR is a surveying method that measures a distance to a target by illuminating the target with laser light and measuring the reflected light with a sensor.

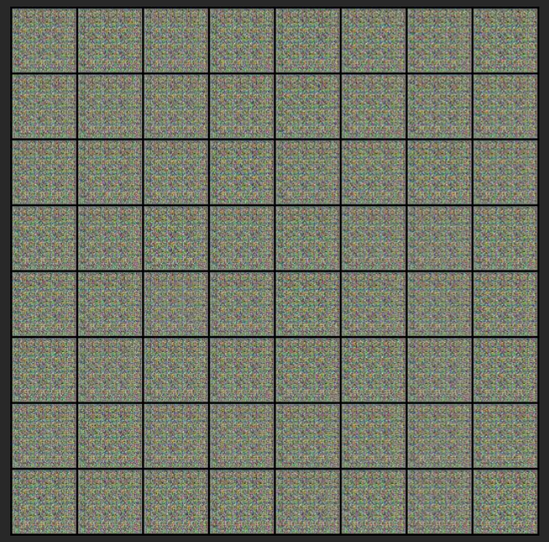
  ▶ How LIDAR works with autonomous driving

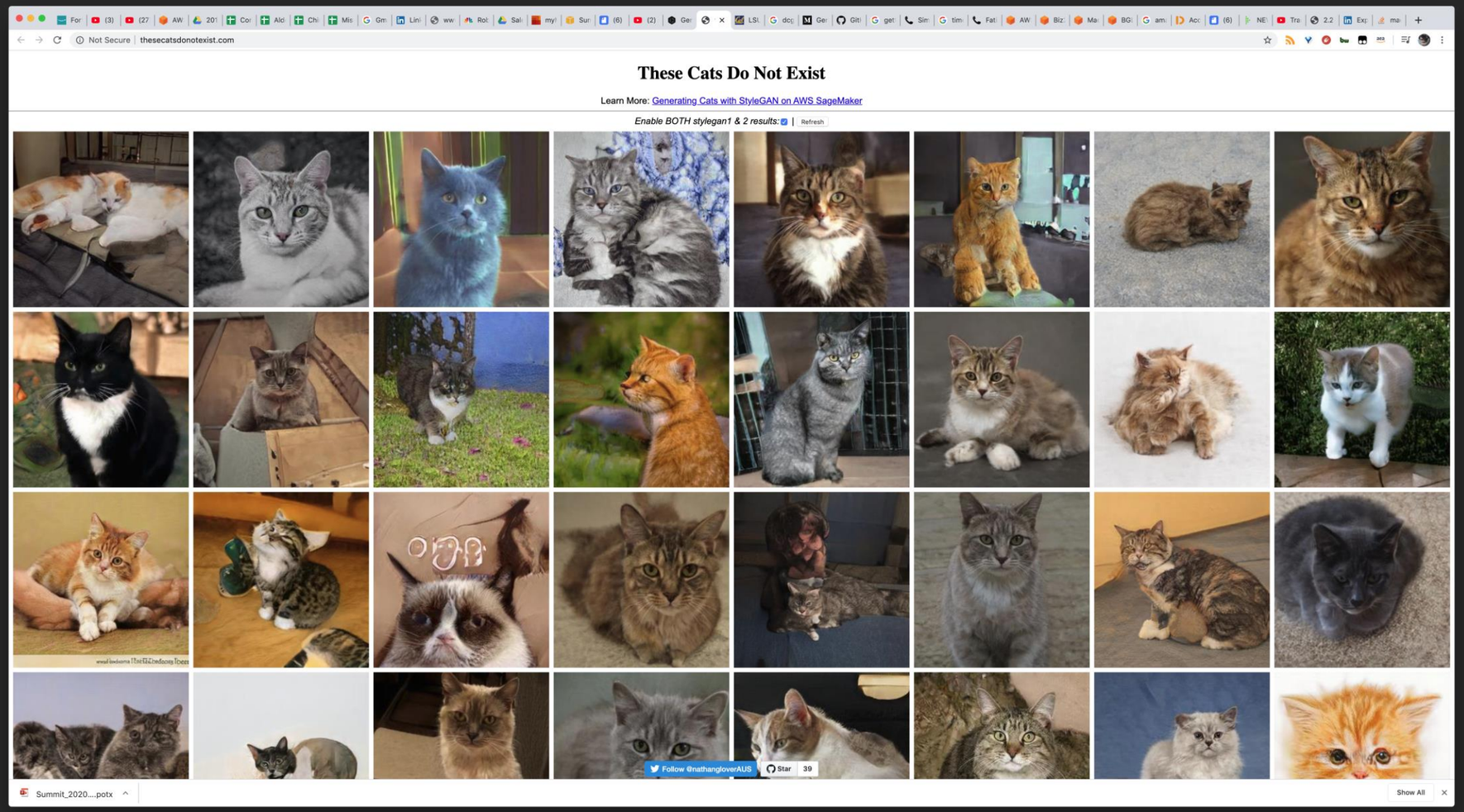


Unsupervised Learning

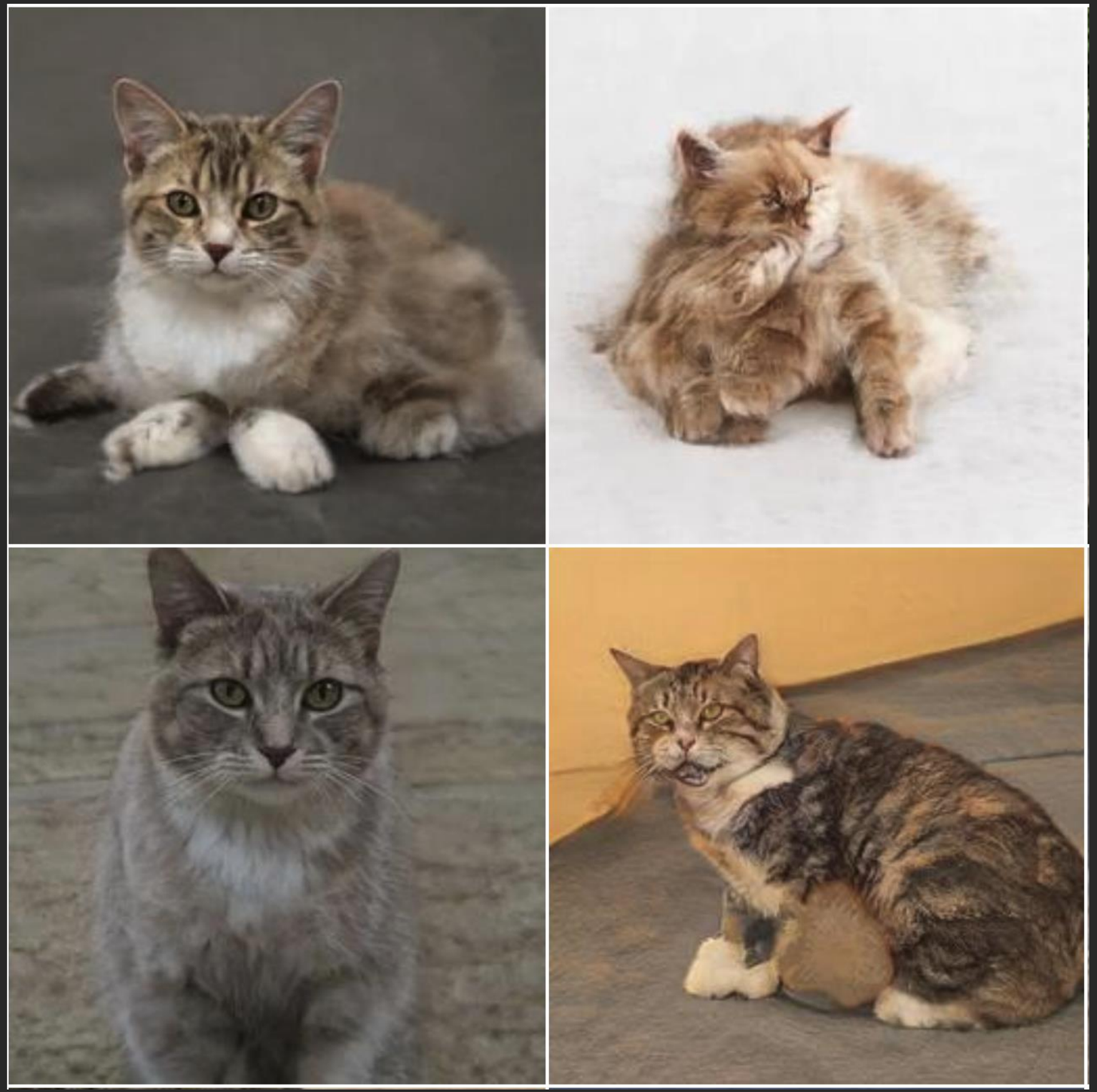
Generative Adversarial Networks

ajolicoeur.wordpress.com/cats/



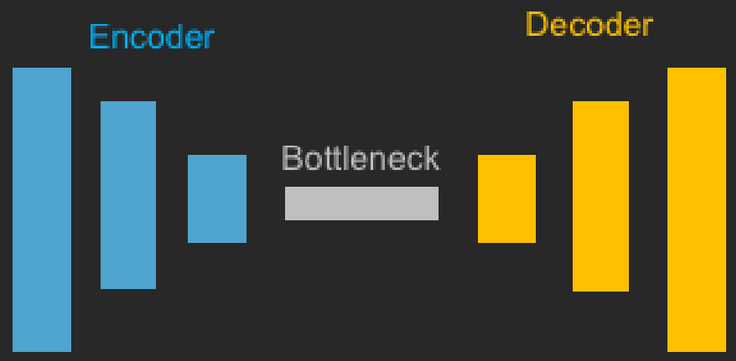


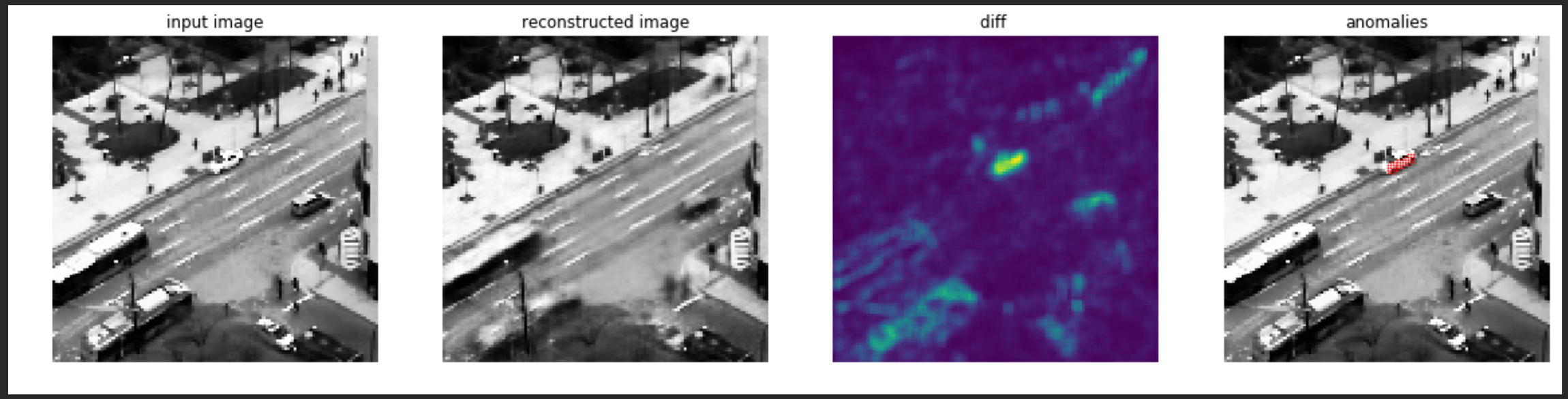
devopstar.com/2019/02/25/generating-cats-with-stylegan-on-aws-sagemaker



Unsupervised learning

Visual Anomaly Detection





Thank you!

Denis V. Batalov

dbatalov

aws SUMMIT ONLINE

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.