[AWS Black Belt Online Seminar] Amazon SageMaker advanced

サービスカットシリーズ

Machine Learning Solutions Architect
宇都宮 聖子
2019/02/13

AWS 公式 Webinar
https://amzn.to/JPWebinar

過去資料
https://amzn.to/JPArchive

自己紹介

宇都宮 聖子

- 機械学習ソリューションアーキテクト
  - 機械学習サービスを担当
  - 前々職は量子情報の研究者
  - 前職は自動車OEMで自動運転開発

  - 好きなサービス
    - Amazon SageMaker

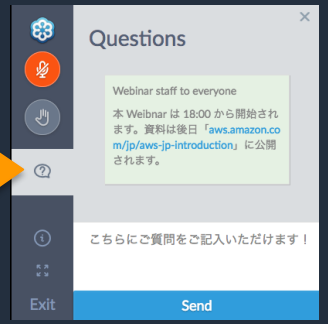
AWS Black Belt Online Seminar とは

「サービス別」「ソリューション別」「業種別」のそれぞれのテーマに分かれて、アマゾン ウェブ サービス ジャパン株式会社が主催するオンラインセミナーシリーズです。

質問を投げることができます！

- 書き込んだ質問は、主催者にしか見えません
- いただいたQ&Aをピックアップしてblogにご紹介させていただく場合がございます
- 今後のロードマップに関するご質問はお答えできませんのでご了承下さい

① 吹き出しをクリック
② 質問を入力
③ Sendをクリック



Twitter ハッシュタグは以下をご利用ください
#awsblackbelt

内容についての注意点

- 本資料では2019年2月13日時点のサービス内容および価格についてご説明しています。最新の情報はAWS公式ウェブサイト(http://aws.amazon.com)にてご確認ください。

- 資料作成には十分注意しておりますが、資料内の価格とAWS公式ウェブサイト記載の価格に相違があった場合、AWS公式ウェブサイトの価格を優先とさせていただきます。

- 価格は税抜表記となっています。日本居住者のお客様が東京リージョンを使用する場合、別途消費税をご請求させていただきます。

- AWS does not offer binding price quotes. AWS pricing is publicly available and is subject to change in accordance with the AWS Customer Agreement available at http://aws.amazon.com/agreement/. Any pricing information included in this document is provided only as an estimate of usage charges for AWS services based on certain information that you have provided. Monthly charges will be based on your actual use of AWS services, and may vary from the estimates provided.

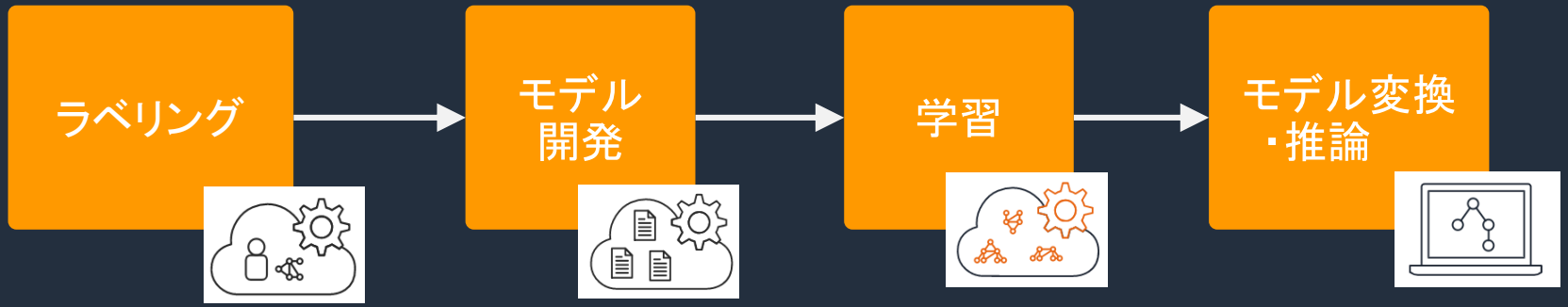
本日のアジェンダ

- 前提: Amazon SageMaker blackbelt の基本的な内容を習得されていること
-
- Amazon SageMaker の一歩進んだ使い方
    - データ準備：Amazon SageMaker Ground Truth
    - モデル開発：ビルトイン，学習スクリプト，ML AWS Marketplace
    - 学習：分散学習，Amazon SageMaker Neo
    - 推論：Elastic Inference
    - 強化学習
    - ワークフロー

Amazon SageMaker とは

- 機械学習システムでよくある問題を解消し，データサイエンティストやエンジニアが素早くプロセスを回せるようにするためのサービス
- 機械学習のインフラ構築・運用を自動化するだけでなく，そのほかのさまざまな機能も提供
- 東京リージョンを含む，13 リージョンにてサービスを展開

SageMakerで行う 機械学習の流れ



【ラベリング】機械学習のための教師データラベリングの支援ツール

【開発】学習するためのコードの記述や，入力データの加工整形を行うための環境

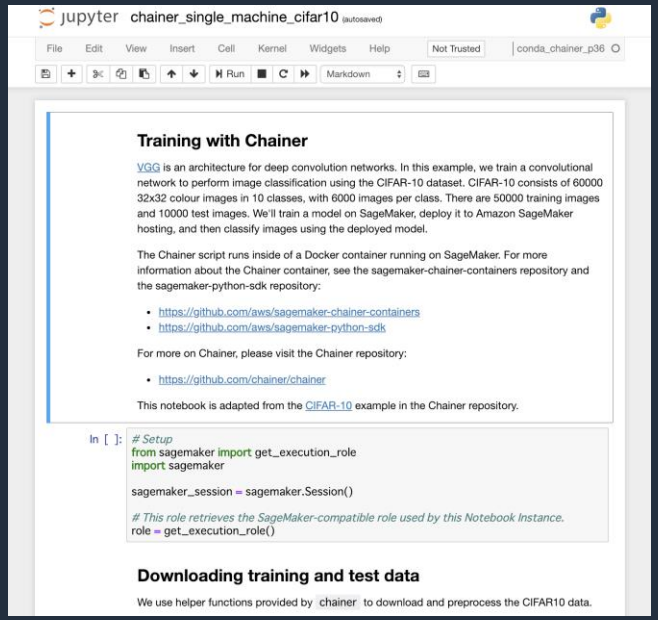
【学習】API を叩くと学習用インスタンスが立ち上がり，学習ジョブを実行．複数ジョブの同時実行や，複数インスタンスでの分散学習，ハイパーパラメーターチューニング

【モデル変換・推論】推論用のエンドポイントをデプロイ．推論におけるレイテンシを減らすため，学習済みモデルを実行環境に最適化された形へコンパイル

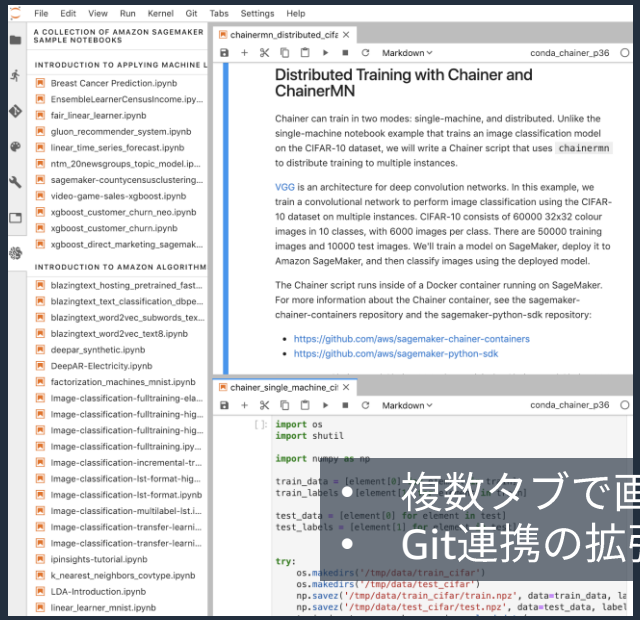
Amazon SageMaker Notebook instance

- SageMaker 上のワークフロー（環境・データのインポート，モデル定義，学習ジョブ，デプロイ，エンドポイント呼び出し）を SageMaker Python SDKで記述する

❖ Jupyter



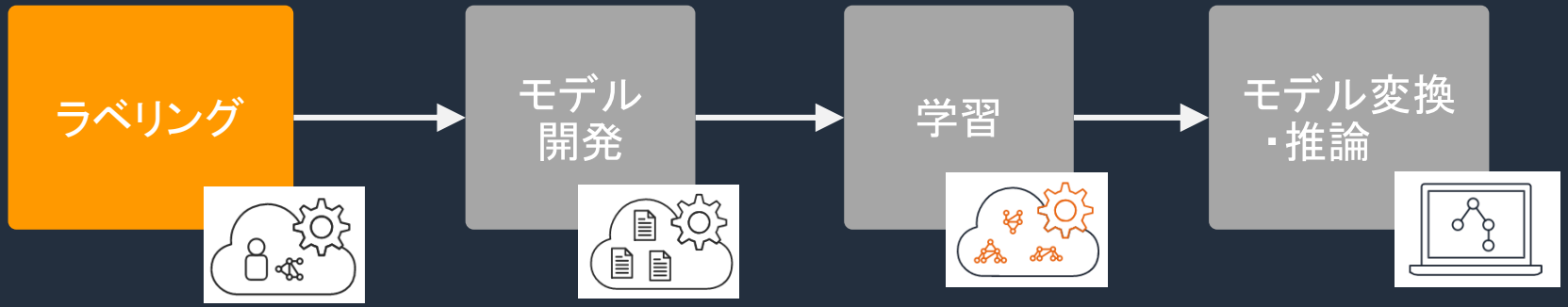
❖ JupyterLab



- 複数タブで画面管理
- Git連携の拡張機能

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

アノテーション



データへのラベル付け（アノテーション）にはコスト・時間がかかる

- 効率の良いアノテーションツールの作成
- 作業を割り当てるワーカーの募集
- 進捗管理・作業割り振り
- 大量データのラベル付け

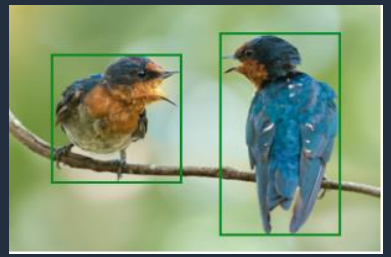
Amazon SageMaker Ground Truth

データにラベル（Ground Truth）を付与するアノテーション作業の支援サービス

- アノテーションにおける一般的なワークフローの管理ツール
- ラベルを付与するワーカーは，Amazon Mechanical Turk，外部ベンダ（AWS Marketplace），自社のプライベートチーム の3つから選択
- 以下の4種類のタスク向けアノテーションツールも提供（カスタムも可能）

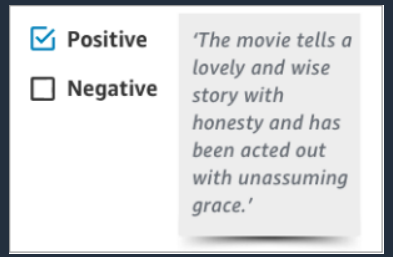
画像分類

物体検出



セマンティック セグメンテーション

文章分類



組み込みのアノテーションツール紹介

アノテーションの基本プロセス

1. アノテーション対象の
データをアップロード

Amazon S3

SageMaker
ユーザ

2. ラベリング
ジョブの作成

Amazon
SageMaker

3. タスクはワーカーに
自動で割り振られる

ワーカー

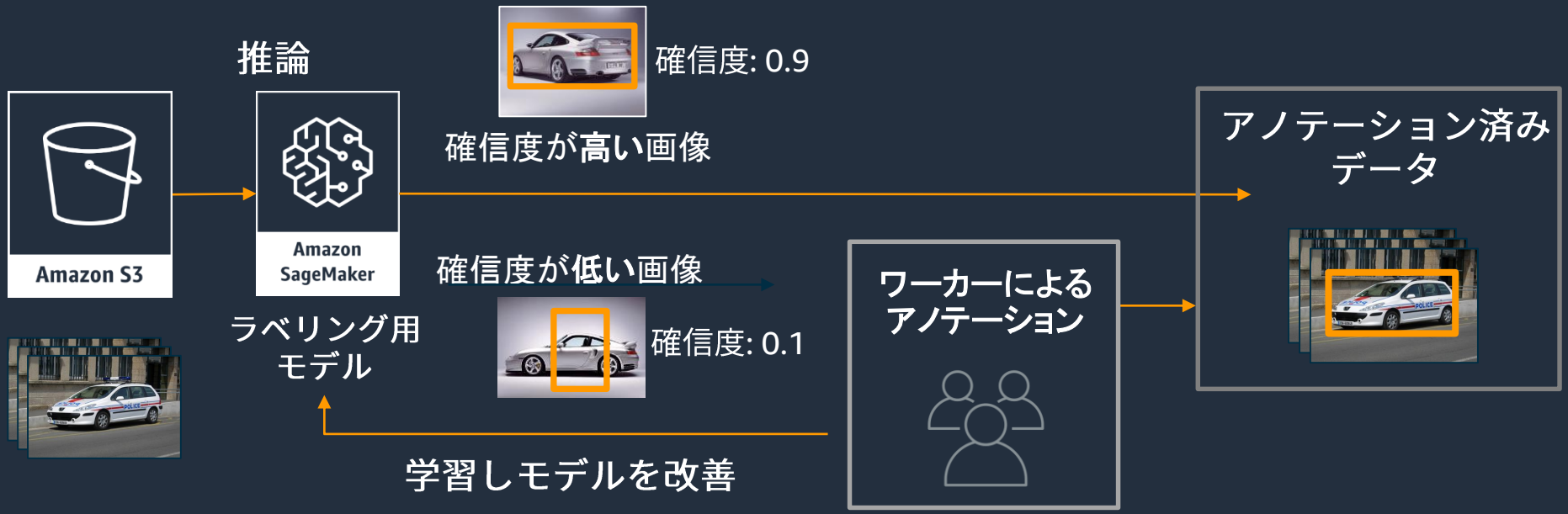
5. アノテーション
結果がS3集計される

4. アノテーションツールで
ワーカーがアノテーションを行う

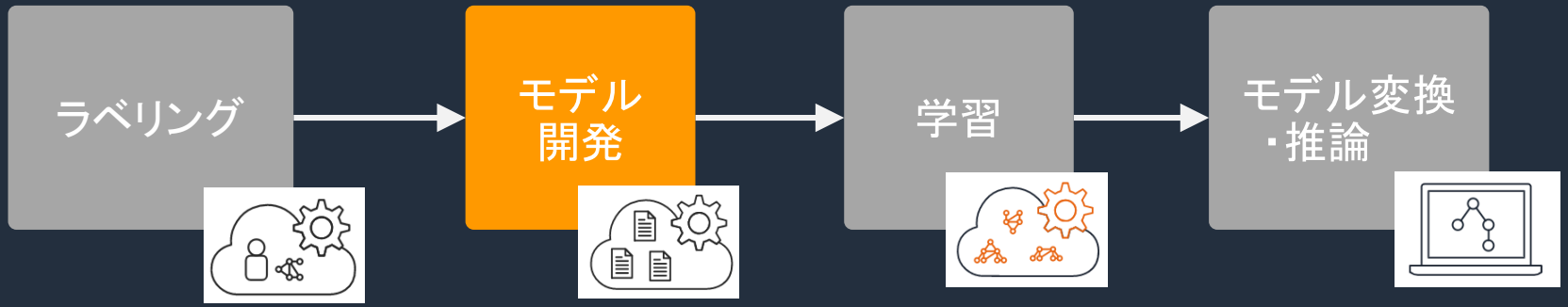
複数人の結果を
マージできる

自動ラベリング（オプション機能）

大規模データセット（5000データ以上）にラベリングする際、数割を
ワーカーでラベル付けし、残りを自動化することで、時間とコストを削減



機械学習のモデルを開発する



モデル開発

- モデル生成の方法
  - ビルトインアルゴリズムを使う
  - SageMaker コンテナ対応のフレームワークを使ったモデル開発
  - AWS Marketplace Machine Learning でモデル購入

モデル開発の選択肢

- Amazon SageMaker ビルトインアルゴリズムを使う
- 自前の学習用スクリプトを使う
- AWS Marketplace で機械学習モデルを購入する

SageMaker ビルトインアルゴリズムを使う

- SageMaker上での実装を最適化した，機械学習アルゴリズム
- アルゴリズムごとにコンテナが準備されており，分散学習などが簡単に使える

Semantic segmentation: ピクセル単位のラベリング

- Fully-Convolutional Network (FCN) algorithm , Pyramid Scene Parsing (PSP) algorithm, DeepLabV3 のいずれかからアルゴリズムを選択

SageMaker ビルトインアルゴリズム ~ 機械学習モデル ~

モデル名	教師データ	アルゴリズム説明	利用用途の例
Linear Learner	あり	線形回帰	分類・回帰などの分析
XGBoost	あり	XGBoost, 勾配ブーストツリー (eXtreme Gradient Boosting)	分類・回帰などの分析
PCA	なし	主成分分析 (Principal Component Analysis)	次元削減
k-means	なし	K平均法	クラスタリング
k-NN	あり	K近傍法	クラスタリング
Factorization Machines	あり	行列分解	レコメンド，回帰，分類
Random Cut Forest	なし	robust random cut tree	時系列データの異常検知
LDA (Latent Dirichlet Allocation)	あり	生成的統計モデル	トピックモデル

※ LDAのオリジナルは教師なし

https://docs.aws.amazon.com/sagemaker/latest/dg/algos.html

aws

SageMaker ビルトインアルゴリズム ~ ディープラーニング モデル ~

	モデル名	教師データ	アルゴリズム	利用用途の例
画像処理	Image classification	あり	ResNet	画像の多値分類
	Object Detection	あり	SSD (Single Shot multibox Detector)	物体の画像内領域をバウンディングボックスで検出
	Semantic Segmentation	あり	FCN, PSP, DeepLabV3 (ResNet50, ResNet101)	ピクセル単位の画像内の物体領域検出
自然言語処理	seq2seq	あり	Deep LSTM	テキスト要約，音声認識
	Neural Topic Model	なし	NTM, LDA	テキストデータの構造化
	Blazing text	なし	Word2Vec	センチメント分析
		あり	Text Classification	単語のマイニング
	Object2Vec	あり	Word2Vec 一般ベクトル化	分類，レコメンド
時系列	DeepAR Forecasting	あり	Autoregressive RNN	確率的な時系列予測
異常検知	IP Insights	なし	NN (IPとentityの関連付け)	悪意あるIPアドレスの検出

https://docs.aws.amazon.com/sagemaker/latest/dg/algos.html

aws

自前の学習用スクリプトを使う

- SageMaker でサポートしているフレームワーク一覧　※ 2019年2月13日時点の情報です

	フレームワーク	SageMaker container サポートバージョン
Deep learning	TensorFlow	**Legacy mode:** 1.4.1, 1.5.0, 1.6.0, 1.7.0, 1.8.0, 1.9.0, 1.10.0 **Script mode:** 1.11.0, 1.12.0
	Chainer	4.0.0, 4.1.0, 5.0.0
	PyTorch	0.4.0, 1.0.0
	MXNet	1.3.0, 1.2.1, 1.1.0, 0.12.1
ML	scikit-learn	0.20.0

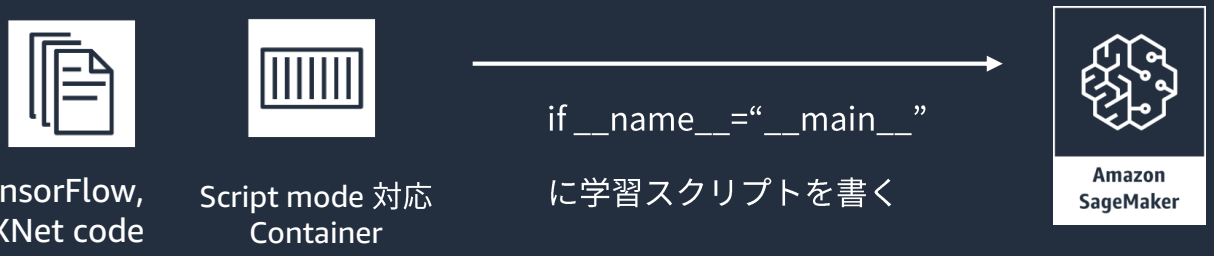
https://github.com/aws/sagemaker-python-sdk/tree/master/src/sagemaker
TensorFlow: https://github.com/aws/sagemaker-python-sdk/tree/master/src/sagemaker/tensorflow
Chainer: https://github.com/aws/sagemaker-python-sdk/tree/master/src/sagemaker/chainer
PyTorch: https://github.com/aws/sagemaker-python-sdk/tree/master/src/sagemaker/pytorch
MXNet: https://github.com/aws/sagemaker-python-sdk/tree/master/src/sagemaker/mxnet
Sklearn: https://github.com/aws/sagemaker-python-sdk/tree/master/src/sagemaker/sklearn

TensorFlowとMXNet が Script Modeに対応

- TensorFlow
  - Ver.1.11以降はscript mode対応，Legacy modeサポートは1.12まで
  - これまでpython2系のみだったが，ver.1.11からpython3系もサポート
  - Elastic Inferenceは1.11.0, 1.12.0 のみ
- MXNet
  - Ver. 1.3.0, 1.2.1, 1.1.0, 0.12.1
- PyTorch, Chainerは元々Script mode

https://github.com/aws/sagemaker-python-sdk/blob/master/src/sagemaker/tensorflow/README.rst

https://github.com/aws/sagemaker-python-sdk/blob/master/src/sagemaker/mxnet/README.rst



TensorFlow が Script Modeに対応

Script Mode では，スクリプトの変更が少なくて済むように

```python
import argparse
import os

if __name__ =='__main__':

    parser = argparse.ArgumentParser()

    # hyperparameters sent by the client are passed as command-line arguments to the script.
    parser.add_argument('--epochs', type=int, default=10)
    parser.add_argument('--batch_size', type=int, default=100)
    parser.add_argument('--learning_rate', type=float, default=0.1)

    # input data and model directories
    parser.add_argument('--model_dir', type=str)
    parser.add_argument('--train', type=str, default=os.environ.get('SM_CHANNEL_TRAIN'))
    parser.add_argument('--test', type=str, default=os.environ.get('SM_CHANNEL_TEST'))

    args, _ = parser.parse_known_args()

    # ... load from args.train and args.test, train a model, write model to args.model_dir.
```

クライアントから送信されたハイパーパラメータは，
コマンドライン引数としてスクリプトに渡される

入力データとモデルのディレクトリ

main ガードの中に学習処理をそのまま書く

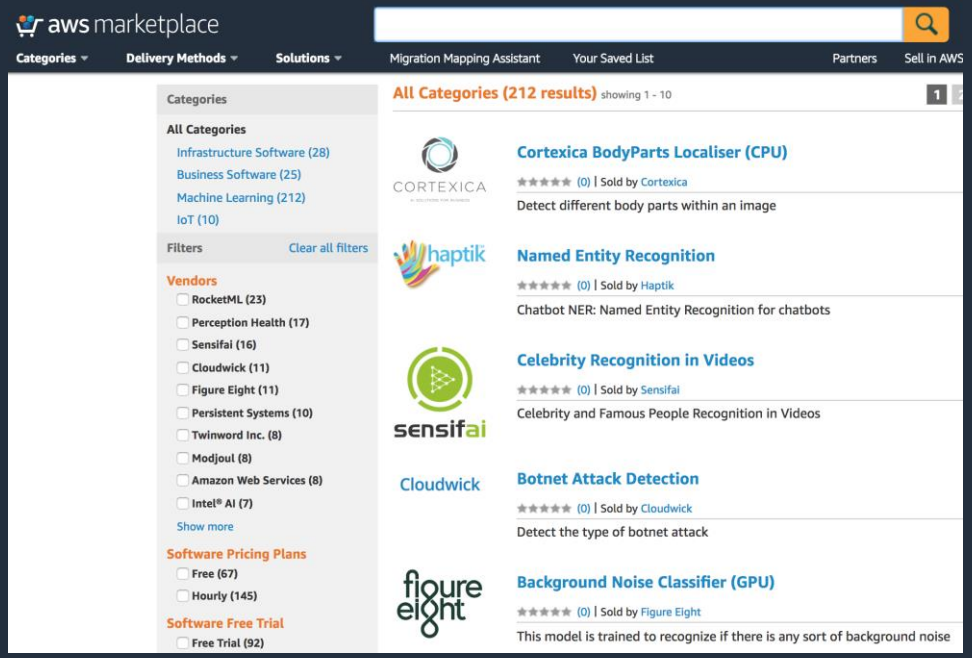
SageMaker 側で引数として渡される hyperparameter や 入力データのパス などを argparseでパースして読み取る

https://github.com/aws/sagemaker-python-sdk/blob/master/src/sagemaker/tensorflow/README.rst

aws

AWS Marketplace から機械学習のモデルを購入する

- AWSマーケットプレイス経由で，SageMaker上で使う機械学習モデルの売買が可能．小売、メディア向けなど200 以上のアルゴリズムがすでに公開済み



アルゴリズム購入者：

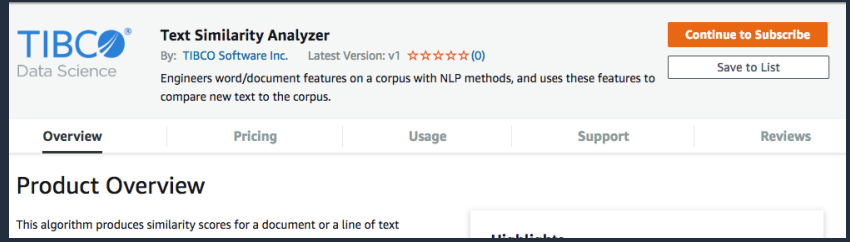
Amazon SageMaker で学習ジョブおよび，推論エンドポイント（バッチ推論ジョブもok）

アルゴリズム購買者：

モデルの中身を秘匿してモデルの出品が可能

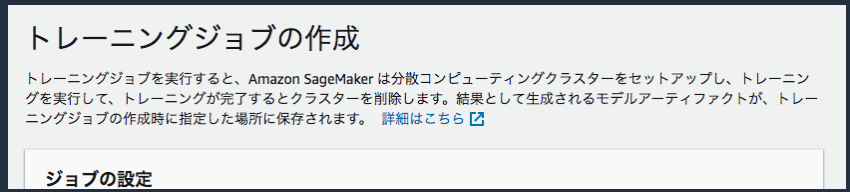
MLマーケットプレイスアルゴリズムの利用（学習）

Marketplace上で
アルゴリズムの選択



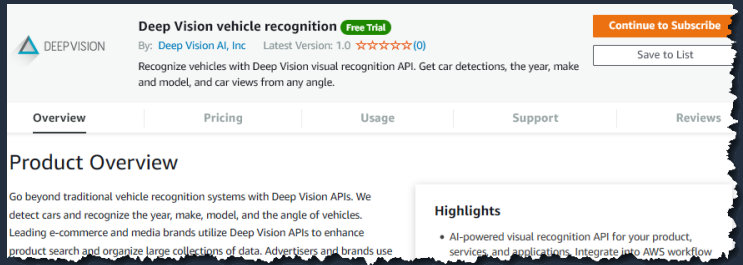
SageMaker上で
アルゴリズム登録

トレーニングジョブ
の作成・実行

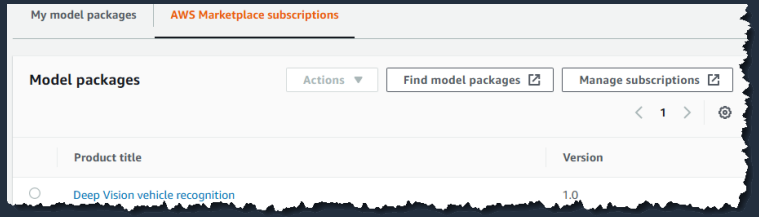


MLマーケットプレイスモデルの利用（推論）

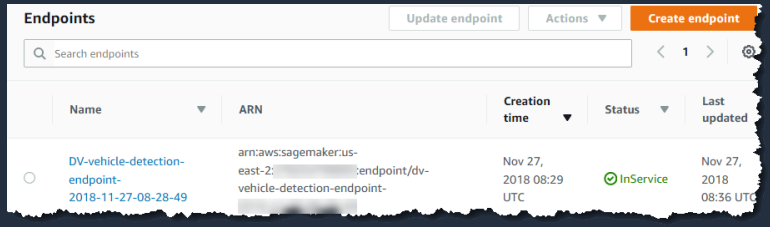
Marketplace 上で
モデルの選択



SageMaker 上での
モデルパッケージ登録



エンドポイントの作成

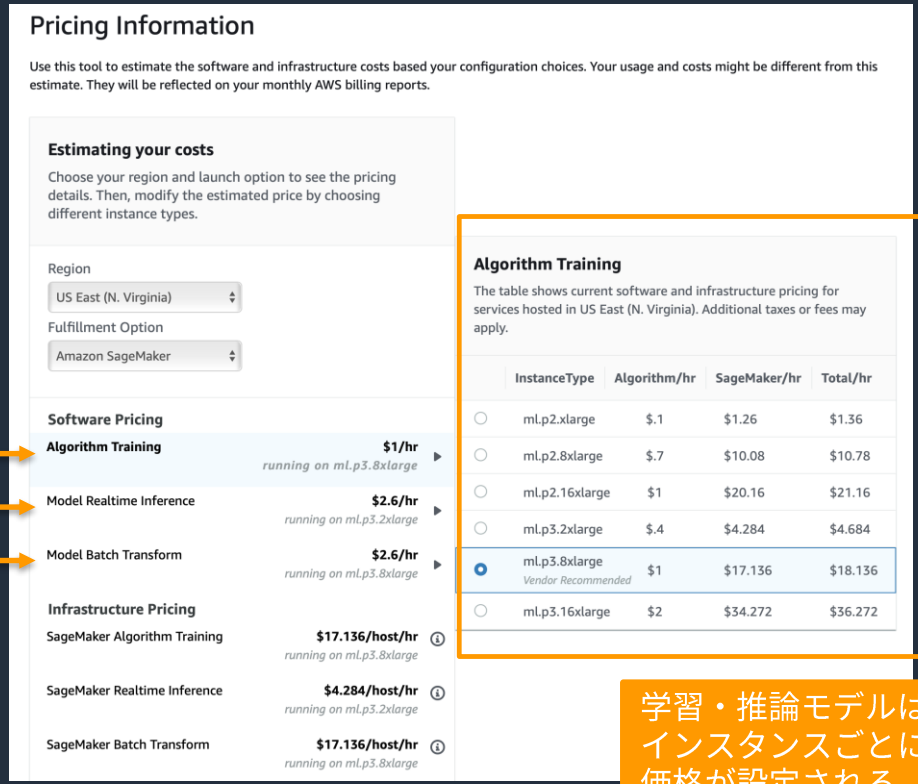


MLマーケットプレイス 価格

SageMakerコンソール，SDK，
AWS CLIからアクセス可能

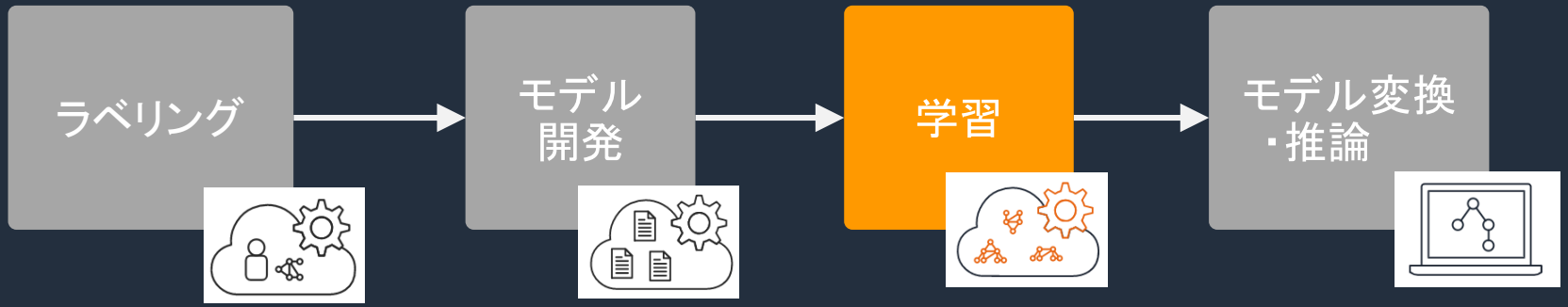
価格：モデルパッケージの利用に対し，
ソフトウエア使用料と AWS のリソース
使用料を利用時間に応じて課金

- アルゴリズム学習
- モデルリアルタイム推論
- モデルバッチ推論



学習・推論モデルは
インスタンスごとに
価格が設定される

機械学習のモデルを開発する



- TensorFlow (Horovodにも対応), MXNet, PyTorch, Chainer それぞれのフレームワークに適した分散学習を提供. training instance 数の指定ですぐに分散学習が利用できる
- ハイパーパラメータ最適化

TensorFlow で分散学習をする場合

- Parameter server を使った分散学習

```
from sagemaker.tensorflow import TensorFlow

tf_estimator = TensorFlow(entry_point='tf-train.py', role='SageMakerRole',
                          train_instance_count=2, train_instance_type='ml.p2.xlarge',
                          framework_version='1.11', py_version='py3',
                          distributions={'parameter_server': {'enabled': True}})
tf_estimator.fit('s3://bucket/path/to/training/data')
```

- Horovod を使った分散学習　　TF v1.12以降で利用可能

```
from sagemaker.tensorflow import TensorFlow

tf_estimator = TensorFlow(entry_point='tf-train.py', role='SageMakerRole',
                          train_instance_count=1, train_instance_type='ml.p2.xlarge',
                          framework_version='1.12', py_version='py3',
                          distributions={
                              'mpi': {
                                  'enabled': True,
                                  'processes_per_host': 2,
                                  'custom_mpi_options': '--NCCL_DEBUG INFO'
                              }
                          })
tf_estimator.fit('s3://bucket/path/to/training/data')
```

`mpi`: trueにすると mpirun コマンドが使えるようになる

https://github.com/aws/sagemaker-python-sdk/blob/master/src/sagemaker/tensorflow/README.rst

ChainerMN を使った分散学習

- ChainerMN によるMPI(Message Passing Interface)を用いた分散学習
- 2019年2月12日時点では，ChainerMN の対応は ver. Chainer 4.1 まで
- フレームワークバージョンが指定されない場合は，デフォルトで 4.1.0 が選択される
- train_instance_count が 2 より大きい場合に，mpirun で学習スクリプトを実行

```
from sagemaker.chainer.estimator import Chainer

chainer_estimator = Chainer(entry_point='chainer_cifar_vgg_distributed.py',
                            source_dir="src",
                            role=role,
                            sagemaker_session=sagemaker_session,
                            use_mpi=True,
                            train_instance_count=2,
                            train_instance_type='ml.p2.xlarge',
                            hyperparameters={'epochs': 30, 'batch-size': 256})

chainer_estimator.fit({'train': train_input, 'test': test_input})
```

```
WARNING:sagemaker:No framework_version specified, defaulting to version 4.1.0. This is not the latest supporte
d version. If you would like to use version 5.0.0, please add framework_version=5.0.0 to your constructor.
INFO:sagemaker:Creating training-job with name: sagemaker-chainer-2019-01-20-05-30-01-031
```

https://github.com/aws/sagemaker-python-sdk/blob/master/src/sagemaker/tensorflow/README.rst

ハイパーパラメータ最適化 (HPO) の書き方

1. Chainer estimator 初期化の際に hyperparameter を指定
2. HyperparameterTuner に hyperparameter_ranges を渡す．連続・離散の範囲指定ジョブ内でベイズ最適化によるHPO
3. HyperparameterTunerにチューニング対象の (2) や，ターゲットメトリクスを指定，CloudWatch Logs に抽出される

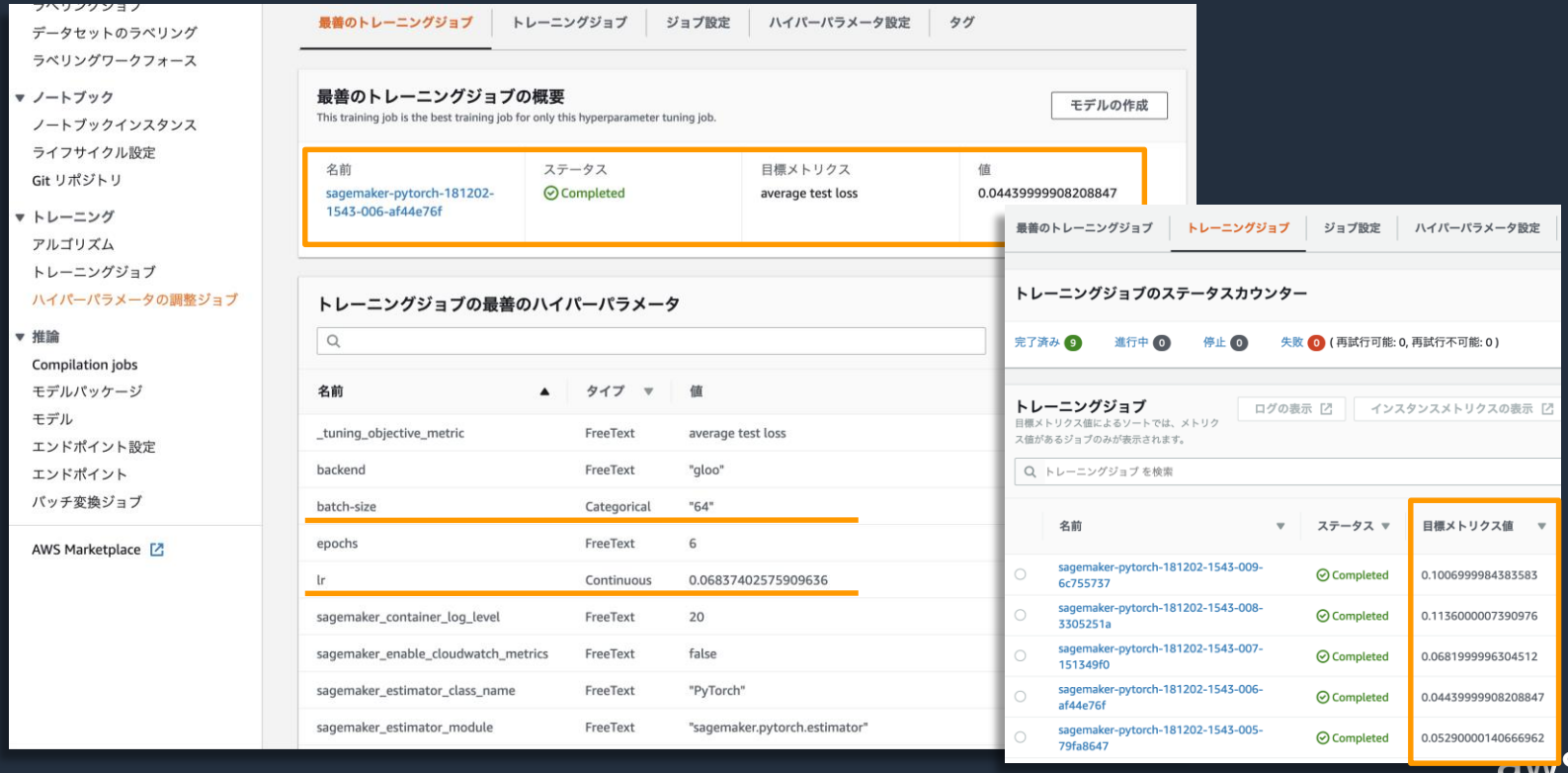
- HPO ウォームスタート機能も利用可能

```
from sagemaker.chainer import Chainer

estimator = Chainer(entry_point="mnist.py",
                    role=role,
                    framework_version='5.0.0',
                    train_instance_count=1,
                    train_instance_type='ml.m4.xlarge',
①                   hyperparameters={'epochs': 30, 'batch-size': 256})

hyperparameter_ranges = {'lr': ContinuousParameter(0.001, 0.1),
②                        'batch-size': CategoricalParameter([32,64,128,256,512])}

tuner = HyperparameterTuner(estimator,
                            objective_metric_name,
③                           hyperparameter_ranges,
                            metric_definitions,
                            max_jobs=9,
                            max_parallel_jobs=3,
                            objective_type=objective_type)

tuner.fit({'training': inputs})
```

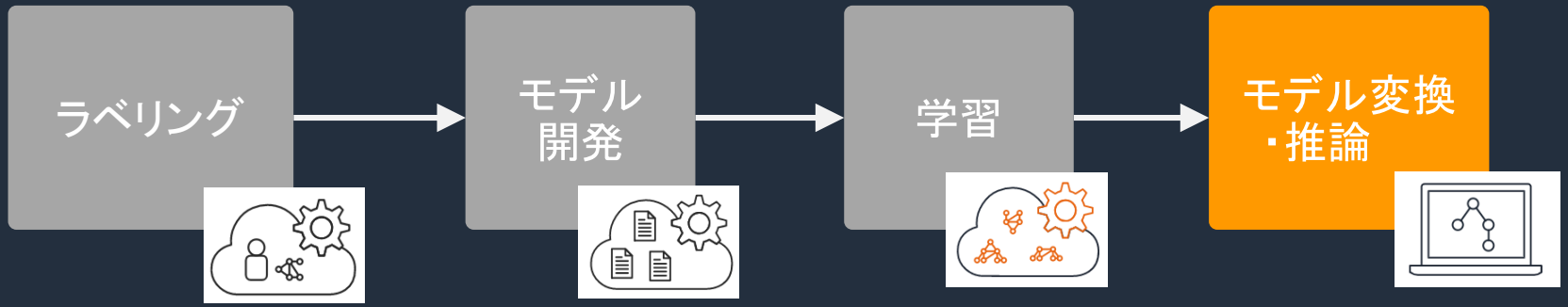
https://github.com/aws/sagemaker-python-sdk#sagemaker-automatic-model-tuning
https://aws.amazon.com/jp/blogs/news/amazon-sagemaker-automatic-model-tuning-becomes-more-efficient-with-warm-start-of-hyperparameter-tuning-jobs/

ハイパーパラメータ探索の結果をコンソールで確認

目標メトリクス値に基づいて探索した結果を



学習済みモデルの変換・推論

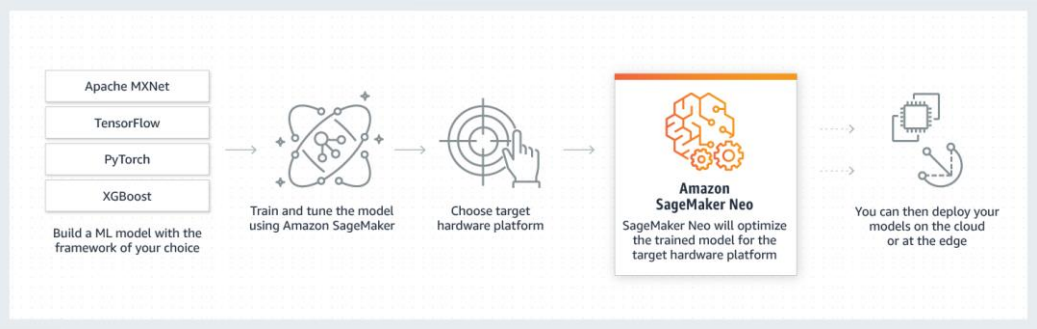


モデル変換・推論

- SageMaker Neo: モデル変換による推論の高速化
- Elastic Inference: GPUアクセラレータ追加による推論のコスト削減と高速化

Amazon SageMaker Neo

- SageMaker で学習したモデルを，推論環境に最適化された形にコンパイル
- デプロイメントプラットフォーム上のリソース使用料を1/10程度に削減，推論の高速化
- MLフレームワーク固有の機能を，どこでも実行できる単一のコンパイル済み環境に変換
- EC2 インスタンスや Greengrass デバイス上で高速に動作するように変換する



https://aws.amazon.com/sagemaker/neo/

SageMaker Neo によるモデルのコンパイル

- 各種MLフレームワークやビルトインアルゴリズムに対応
- 対応リージョンはバージニア北部，オレゴン，オハイオ，アイルランド
- Neoの利用料金は無料（インスタンスでのジョブ実行料金のみ）

対応フレームワーク	対応プラットフォーム
• TensorFlow • MXNet • PyTorch • ONNX • XGBoost	• EC2 インスタンス（ml.c4, ml.c5, ml.m4, ml.m5, ml.p2, ml.p3） • Jetson TX1/2 • DeepLens • Raspberry Pi 3 Model

https://aws.amazon.com/jp/blogs/news/amazon-sagemaker-neo-train-your-machine-learning-models-once-run-them-anywhere/
https://docs.aws.amazon.com/sagemaker/latest/dg/neo.html

SageMaker Neo の Python SDK による利用の流れ

- TensorFlowEstimator.compile_model メソッドで学習モデルを，ターゲットデバイスに最適なコンパイル

学習

C5向けに最適化

C5にデプロイ

```
mnist_estimator = TensorFlow(
                    entry_point='mnist.py', role=role, framework_version='1.11.0’,
                    training_steps=1000, evaluation_steps=100,
                    train_instance_count=2, train_instance_type='ml.c4.xlarge’)

mnist_estimator.fit(inputs)

optimized_estimator = mnist_estimator.compile_model(
                    target_instance_family='ml_c5', input_shape={'data':[1, 784]},
                    output_path=output_path,
                    framework='tensorflow’, framework_version='1.11.0’)

optimized_predictor = optimized_estimator.deploy(
                    initial_instance_count = 1, instance_type = 'ml.c5.4xlarge')
```

https://github.com/awslabs/amazon-sagemaker-examples/blob/master/sagemaker-python-sdk/tensorflow_distributed_mnist/tensorflow_distributed_mnist_neo.ipynb

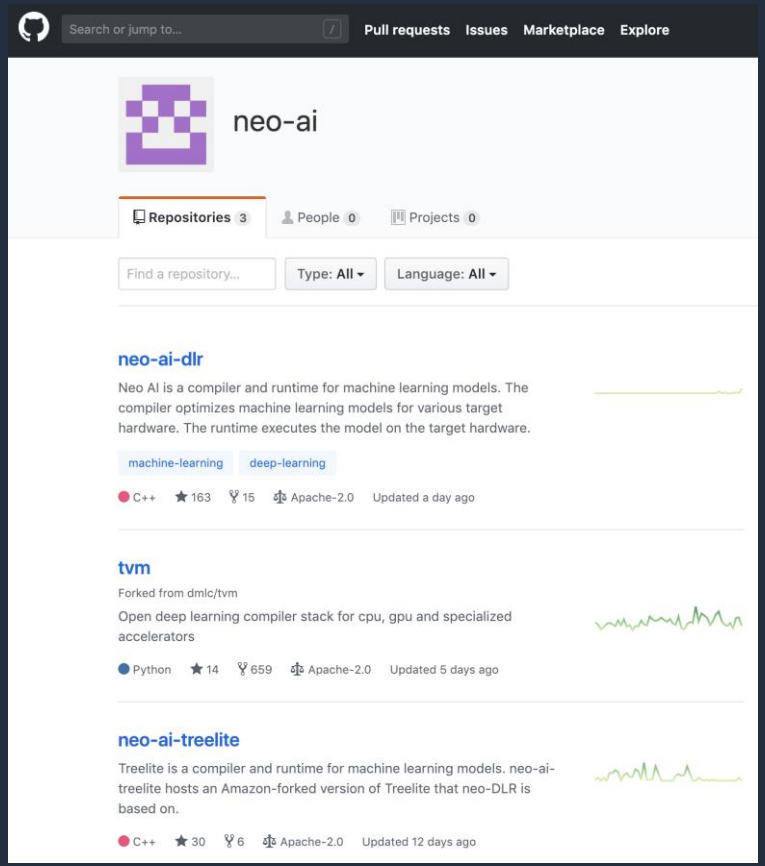
aws

OSS Neo-AI プロジェクト

- Apache Software License に基づく Neo-AI プロジェクトを2019年1月発表
- ワシントン大学の OSS 研究プロジェクト DMLC (Distributed Machine Learning Common) のTVMとTreeliteを拡張
    - TVM: Deep learning stack compiler for CPU, GPU.
    - Treelite: Decision tree compiler

- LLVMやHalideなどの従来のコンパイラテクノロジに関する数十年にわたる研究に基づいて構築されたランタイム

- 現在は AWS, ARM, Intel, NVIDIAをサポート
- Cadence, Qualcomm, Xilinx にも対応予定



https://github.com/neo-ai/

CPU インスタンスの計算を GPU でアクセラレート

- スタンドアロンのGPU は主に学習に最適化されており、推論には大きすぎる
- リアルタイムの推論では 少量のGPUしか消費せず高コスト
- 適切なGPU リソースをCPUに関連づけることで，高速な推論を低コストで実行
- 画像分類の推論処理をする場合、コストを下げながら十分なパフォーマンスを得ることが可能

インスタンス	EIA	推論速度 (msec)	価格 ($/hour)
c5.large	なし	230 msec	$0.085
c5.large	eia1.medium	46 msec	$0.22
p2.xlarge	なし	42 msec	$0.90

価格は 2018/11 時点のバージニア北部のもの

https://aws.amazon.com/jp/blogs/news/amazon-elastic-inference-gpu-powered-deep-learning-inference-acceleration/

Amazon Elastic Inference

- 推論に適した低コストのGPU駆動のアクセラレーションを，CPU の EC2 および SageMaker インスタンスに適用．DL 実行コストを最大75%削減
- 最大 32TFLOPSの混合精度演算を提供できる

- 利用方法
    - エンドポイントのインスタンスにEIAアタッチして利用
    - ローカルモード推論時は SageMaker Notebook インスタンスにEIAをアタッチ

Accelerator type	TFLOPS FP32 throughput	TFLOPS FP16 throughput	Memory in GB
ml.eia1.medium	1	8	1
ml.eia1.large	2	16	2
ml.eia1.xlarge	4	32	4

https://docs.aws.amazon.com/sagemaker/latest/dg/ei.html

aws

Amazon Elastic Inference の利用

- SageMaker Python SDK のTensorFlowまたは MXNet と，SageMaker 用コンテナを使用
- 他のフレームワークはONNXを使ってエクスポートし，MXNetにインポートして利用
- Image Classification, Object Detection のいずれかの組み込みアルゴリズムを使用
- TensorFlow または MXNet の EIバージョン対応の独自コンテナを構築

- 例）TensorFlowの場合：
  - Estimator または Model オブジェクトの deploy メソッドで EIA を指定

```
# Deploy an estimator using EI (using the accelerator_type input argument)
predictor = estimator.deploy(initial_instance_count=1,
                             instance_type='ml.m4.xlarge',
                             accelerator_type='ml.eia1.medium')
```

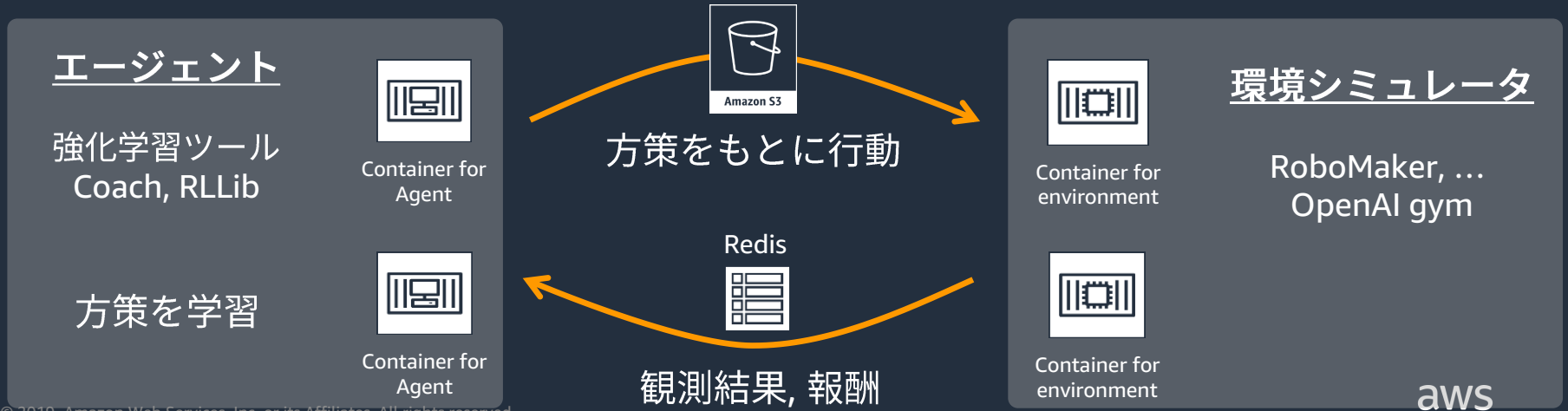
```
# Deploy a model using EI (using the accelerator_type input argument)
predictor = model.deploy(initial_instance_count=1,
                             instance_type='ml.m4.xlarge',
                             accelerator_type='ml.eia1.medium')
```

https://docs.aws.amazon.com/ja_jp/sagemaker/latest/dg/ei.html

aws

Amazon SageMaker Reinforcement Learning

- ゲームやロボットのシミュレーション環境と統合した SageMaker 上の強化学習
- 強化学習ツールキット として Coach とRL-Ray をサポート
- AWS RoboMaker，Roboschoolなど OSS の OpenAI Gym interface 経由で利用可能
- 強化学習ライブラリを利用し，分散学習環境（学習の分散と環境の分散が可能）



SageMaker 強化学習の構成　~ RoboMakerを使ったDeepRacer~

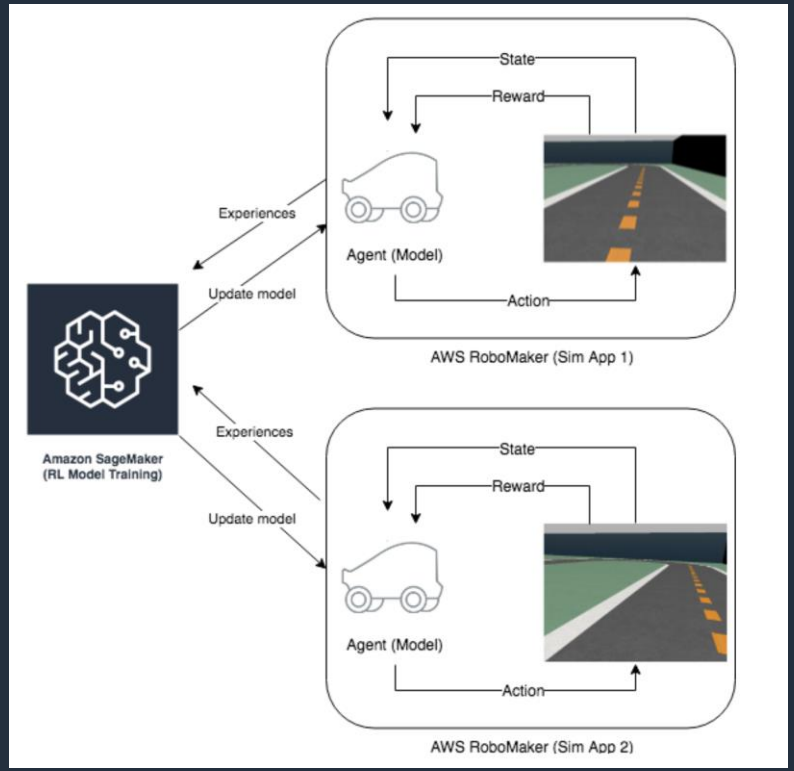
1. 目的関数：レーンの中心に沿って自律走行する

2. 環境：AWS RoboMaker でホストされている3Dドライビングシミュレータ

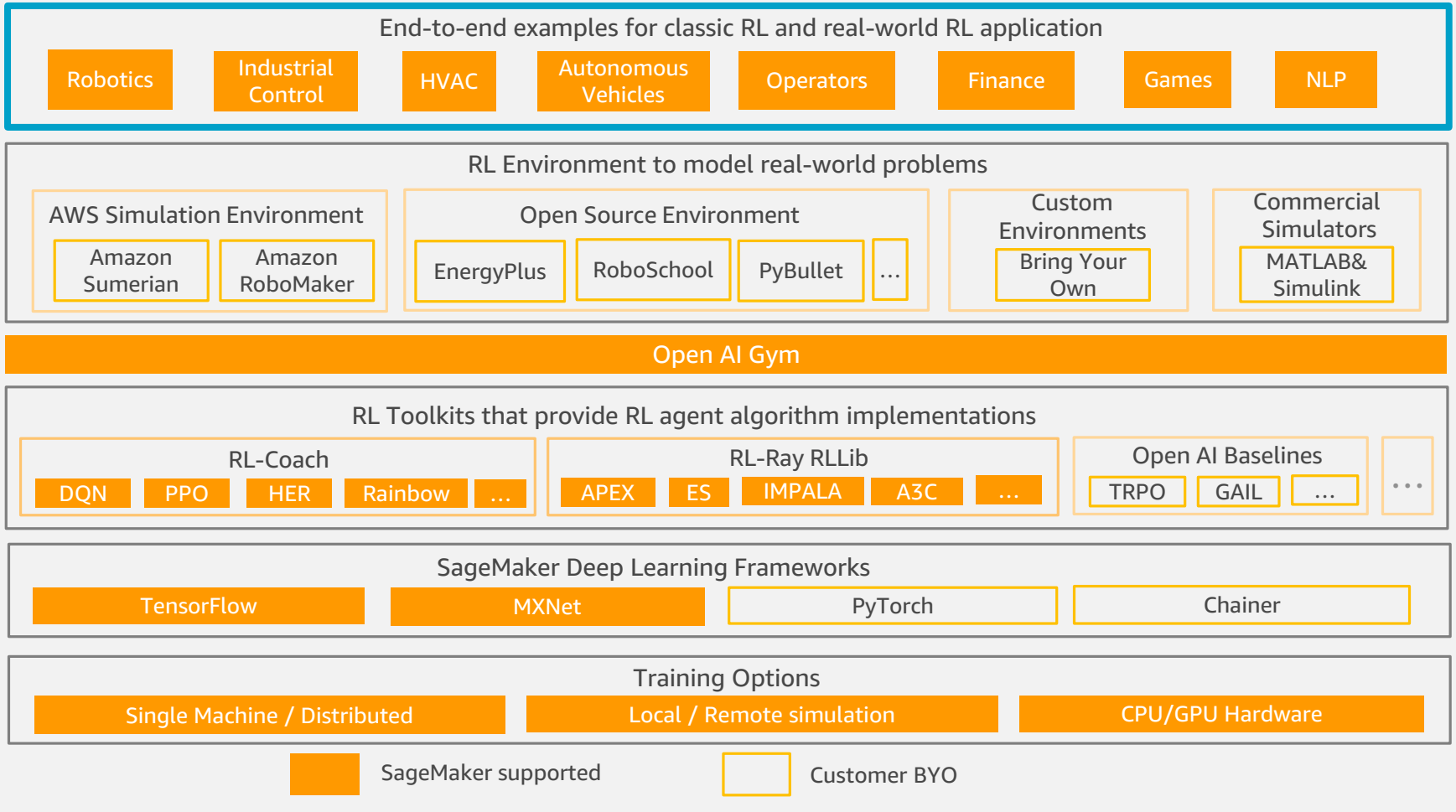
3. 状態：右図のような，車のヘッドカメラで撮影した運転中の POV 画像

4. アクション：異なる角度で6つの別々のステアリングホイール位置（設定可能）

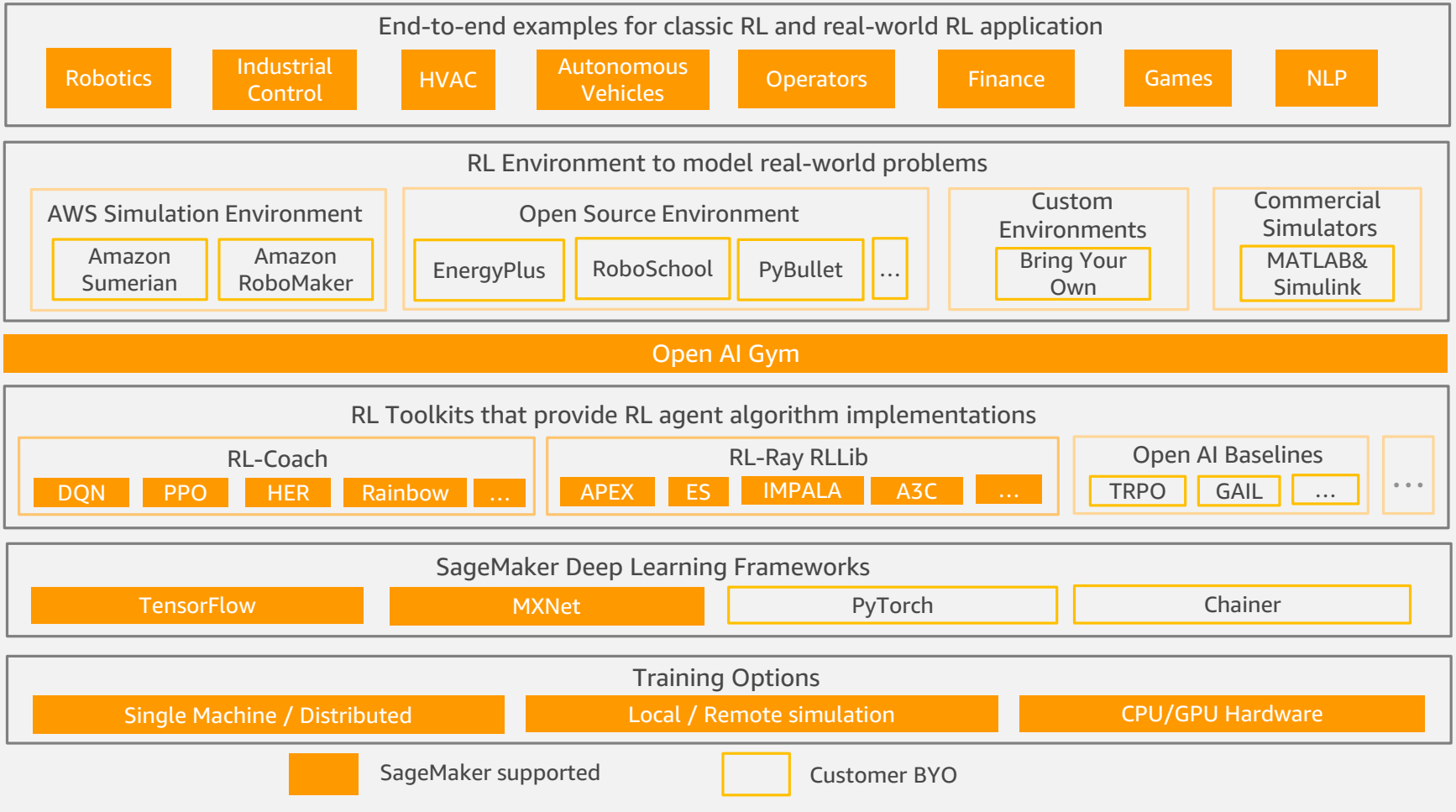
5. 報酬：[設定設定の例] 中心線に近づくことでプラスに，トラックから外れることにより大きなペナルティが課される．例えばステアリングペナルティを追加することなど，詳細の設定も可能



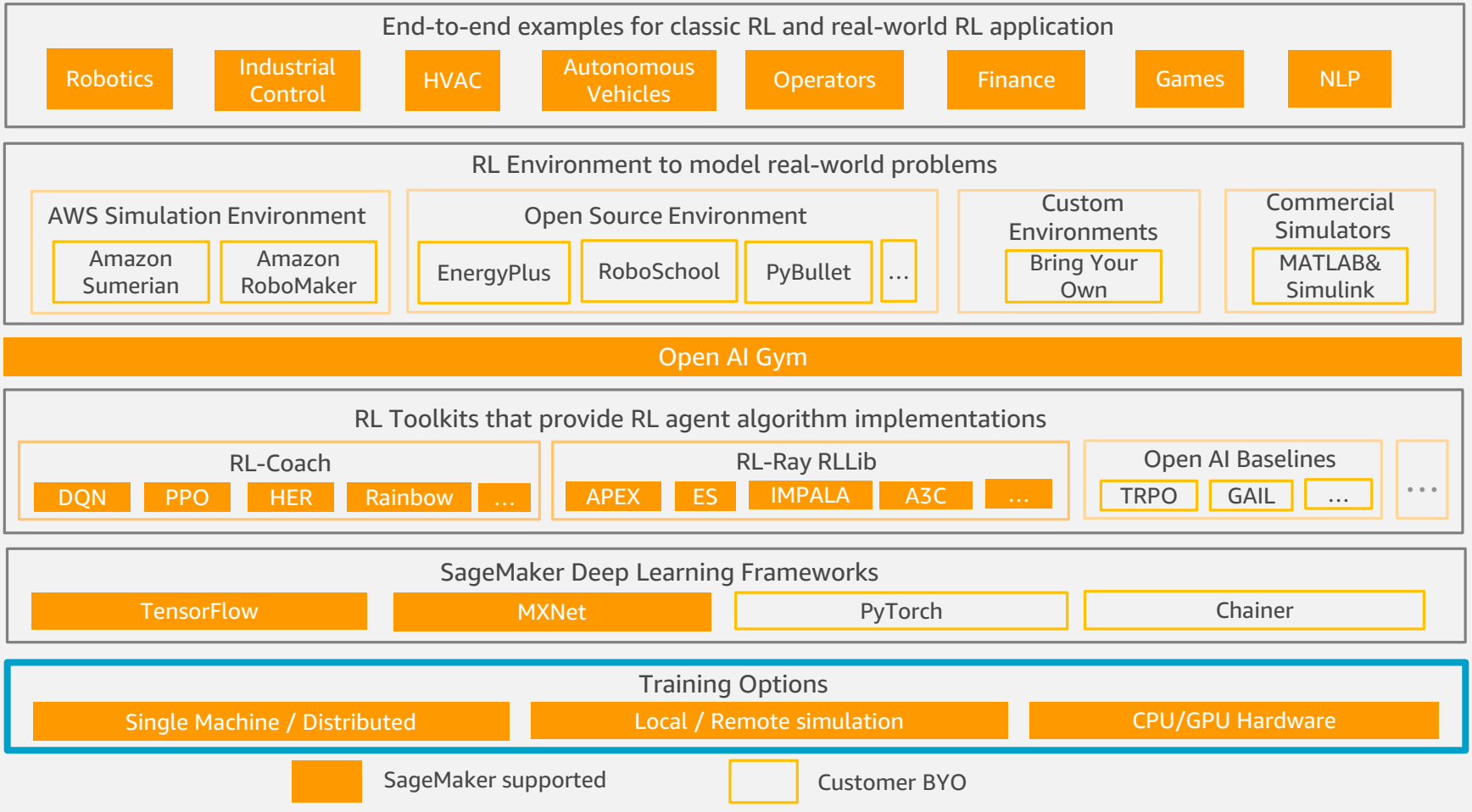
Amazon SageMaker Reinforcement Learning



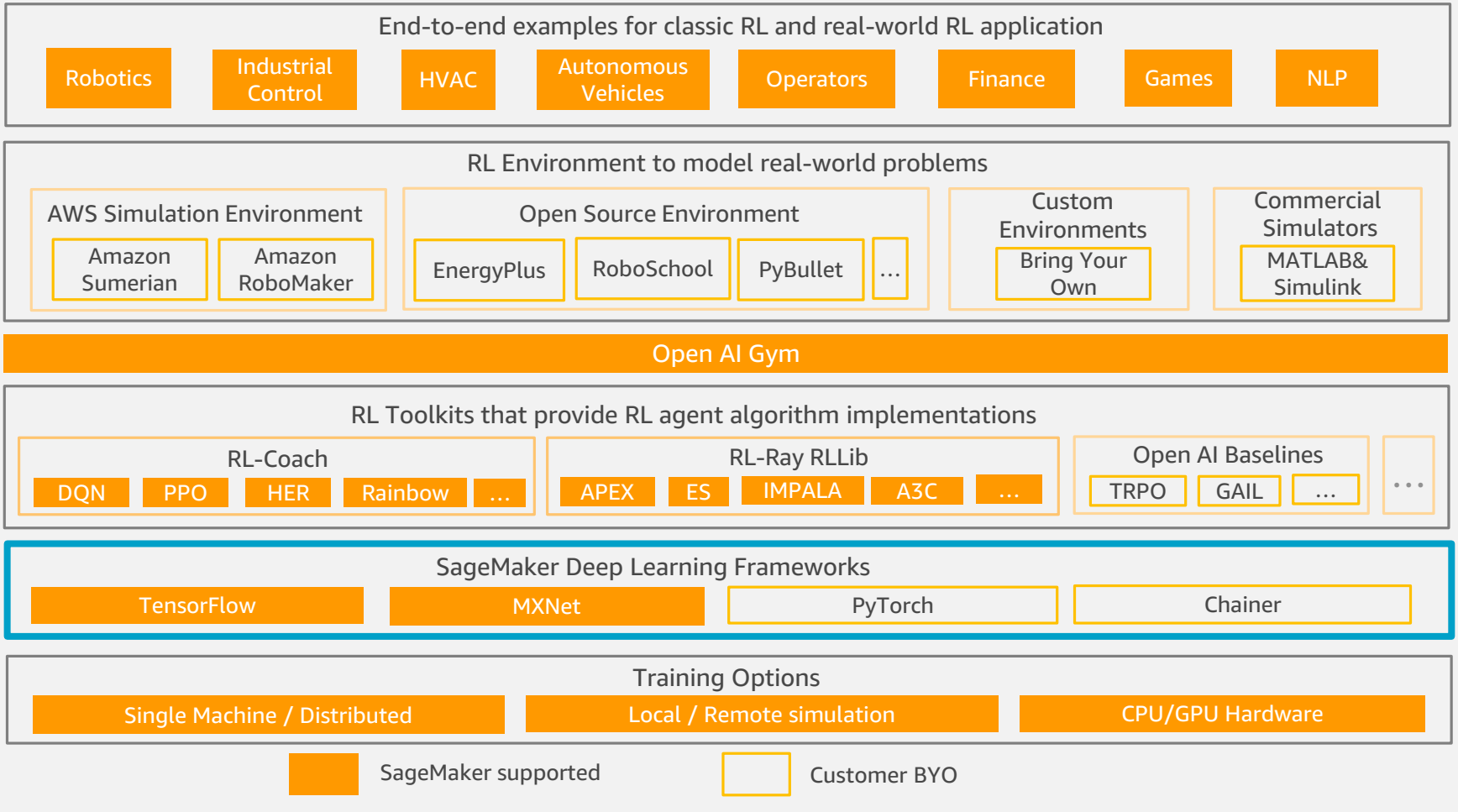
Amazon SageMaker Reinforcement Learning



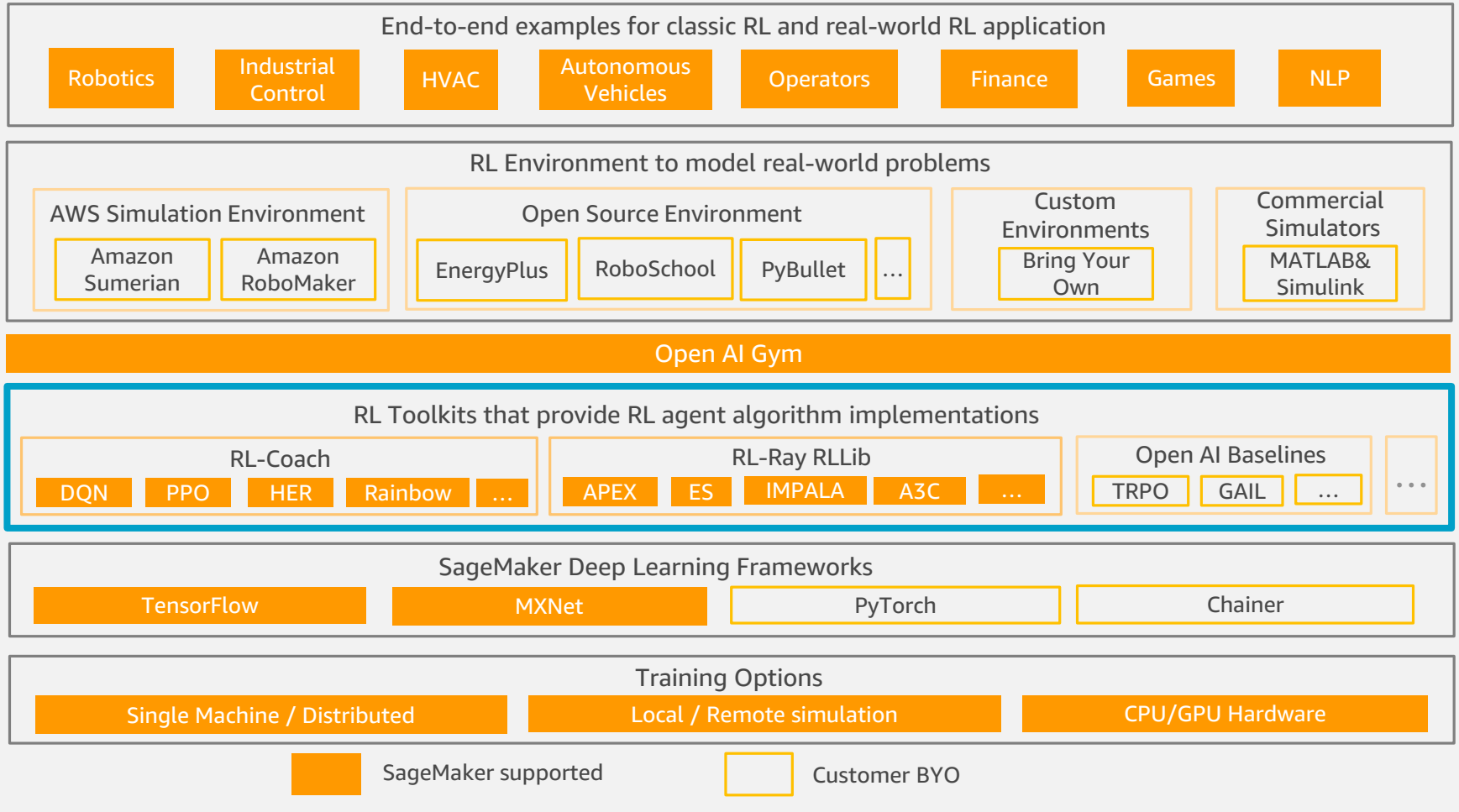
Amazon SageMaker Reinforcement Learning



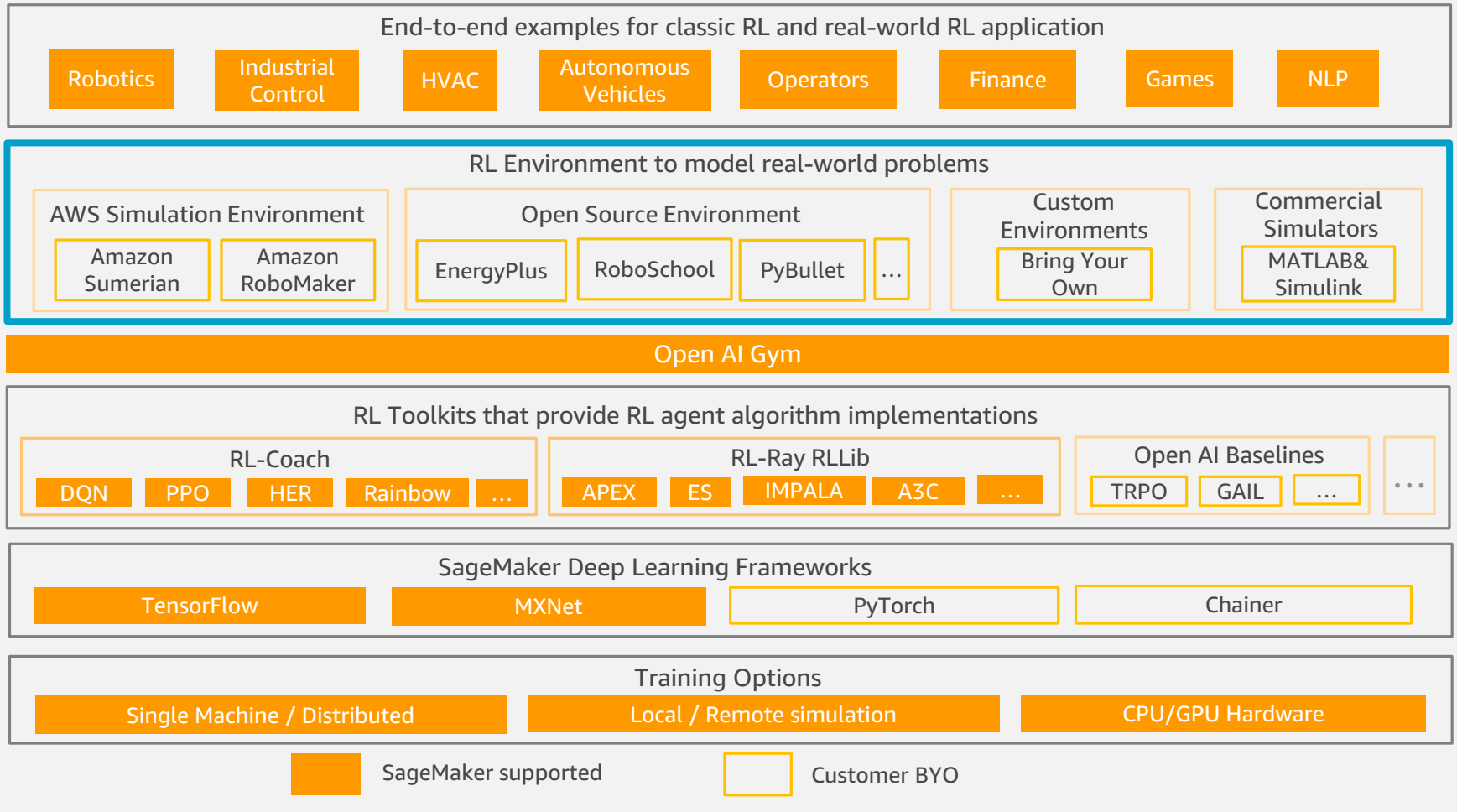
Amazon SageMaker Reinforcement Learning



Amazon SageMaker Reinforcement Learning



Amazon SageMaker Reinforcement Learning



強化学習モデルの学習・デプロイ

```
from sagemaker.rl import RLEstimator, RLToolkit, RLFramework

# Train my estimator
rl_estimator = RLEstimator(entry_point='coach-train.py',
                           toolkit=RLToolkit.COACH,
                           toolkit_version='0.11.0',
                           framework=RLFramework.MXNET,
                           role='SageMakerRole',
                           train_instance_type='ml.c4.2xlarge',
                           train_instance_count=1)

rl_estimator.fit()

# Deploy my estimator to a SageMaker Endpoint and get a MXNetPredictor
predictor = rl_estimator.deploy(instance_type='ml.m4.xlarge',
                                initial_instance_count=1)

response = predictor.predict(data)
```

https://github.com/aws/sagemaker-python-sdk/tree/master/src/sagemaker/rl

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

学習用コンテナとシミュレーション用コンテナの分離

- 学習にはGPU，環境シミュレータにはCPU
- Estimatorを独立に作り，インスタンスを指定

```
primary_cluster_instance_type = "ml.p3.2xlarge"
primary_cluster_instance_count = 1

secondary_cluster_instance_type = "ml.c5.4xlarge"
secondary_cluster_instance_count = 2

total_cpus = 40 - 1 # Leave one for ray scheduler
total_gpus = 1
```

```
primary_cluster_estimator = RLEstimator(entry_point="train-%s.py" % roboschool_problem,
                        source_dir='src',
                        dependencies=["common/sagemaker_rl"],
                        image_name=gpu_image_name,
                        role=role,
                        train_instance_type=primary_cluster_instance_type,
                        train_instance_count=primary_cluster_instance_count,
                        output_path=s3_output_path,
                        base_job_name=job_name_prefix,
                        metric_definitions=metric_definitions,
                        train_max_run=int(3600 * .5), # Maximum runtime in seconds
                        hyperparameters={
                          "s3_prefix": s3_prefix, # Important for syncing
                          "s3_bucket": s3_bucket, # Important for syncing
                          "aws_region": boto3.Session().region_name, # Important for S3 connection
                          "rl_cluster_type": "primary", # Important for syncing
                          "rl_num_instances_secondary": secondary_cluster_instance_count, # Important for syncing
                          "rl.training.config.num_workers": total_cpus,
                          "rl.training.config.train_batch_size": 20000,
                          "rl.training.config.num_gpus": total_gpus,
                        },
                        subnets=default_subnets, # Required for VPC mode
                        security_group_ids=default_security_groups # Required for VPC mode
                    )

primary_cluster_estimator.fit(wait=False)
primary_job_name = primary_cluster_estimator.latest_training_job.job_name
print("Primary Training job: %s" % primary_job_name)
```

Primary cluster with GPU

```
secondary_cluster_estimator = RLEstimator(entry_point="train-%s.py" % roboschool_problem,
                        source_dir='src',
                        dependencies=["common/sagemaker_rl"],
                        image_name=cpu_image_name,
                        role=role,
                        train_instance_type=secondary_cluster_instance_type,
                        train_instance_count=secondary_cluster_instance_count,
                        output_path=s3_output_path,
                        base_job_name=job_name_prefix,
                        metric_definitions=metric_definitions,
                        train_max_run=3600, # Maximum runtime in seconds
                        hyperparameters={
                          "s3_prefix": s3_prefix, # Important for syncing
                          "s3_bucket": s3_bucket, # Important for syncing
                          "aws_region": boto3.Session().region_name, # Important for S3 connection
                          "rl_cluster_type": "secondary", # Important for syncing
                        },
                        subnets=default_subnets, # Required for VPC mode
                        security_group_ids=default_security_groups # Required for VPC mode
                    )

secondary_cluster_estimator.fit(wait=False)
secondary_job_name = secondary_cluster_estimator.latest_training_job.job_name
print("Secondary Training job: %s" % secondary_job_name)
```

Secondary cluster with CPU

https://github.com/awslabs/amazon-sagemaker-examples/blob/master/reinforcement_learning/rl_roboschool_ray/rl_roboschool_ray_distributed.ipynb

aws

SageMaker Reinforcement Learning 対応フレームワーク

Dependencies	Coach 0.10.1	Coach 0.11.0	Ray 0.5.3
Python	3.6	3.5 (MXNet) 3.6 (TensorFlow)	3.6
CUDA (GPU image only)	9.0	9.0	9.0
DL Framework	TensorFlow-1.11.0	MXNet-1.3.0 TensorFlow-1.11.0	TensorFlow-1.11.0
gym	0.10.5	0.10.5	0.10.5

https://github.com/aws/sagemaker-python-sdk/tree/master/src/sagemaker/rl#distributed-rl-training

※ 2019年2月13日時点の情報です

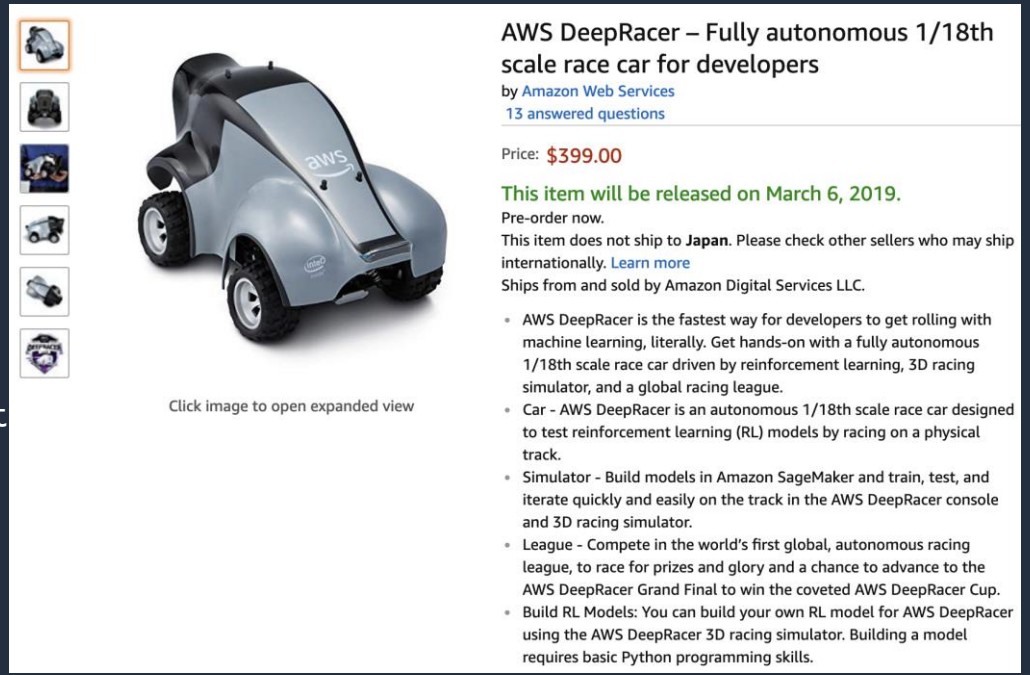
aws

AWS DeepRacer

Amazon.com にて$399.00, 2019/3/6 発売開始予定

仕様

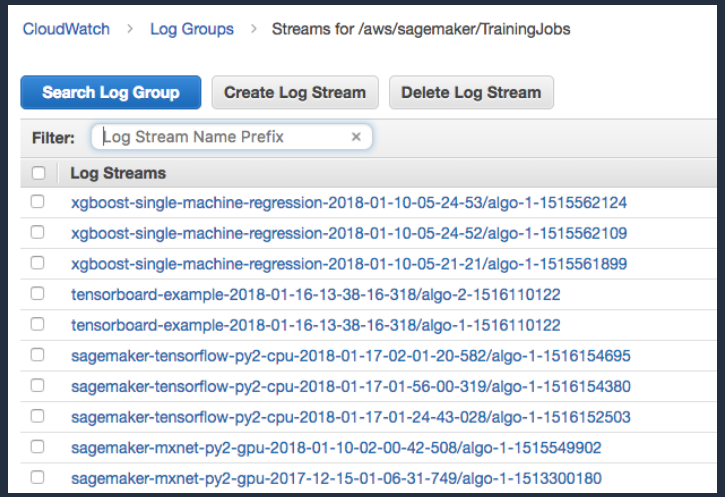
- 1/18 スケールラジコンカー
- Intel Atom プロセッサ
- 4Mピクセル, 1080p カメラ
- WiFi (802.11ac)
- Compute用バッテリー (>2h)と、モーター用バッテリー(>15m)
- Ubuntu 16.04LTS, ROS (Robot Operat System), OpenVino

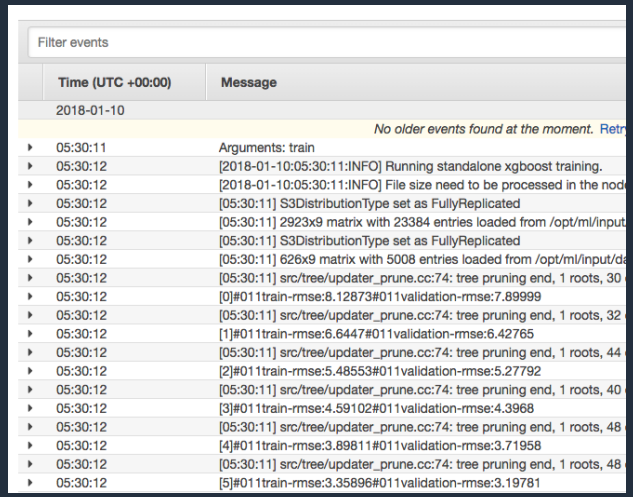


https://aws.amazon.com/deepracer/

学習ジョブの評価

- 学習ジョブの実行時のログは，すべて CloudWatch Logs に出力される
- 自作コンテナを使う場合は，標準出力に出したものがそのまま CloudWatch Logs に送られる
- モデル評価等は，すべてログに出力して，あとから集計





メトリクスのモニタリング

- 学習時のモデルの精度をグラフ化できる
- 学習が上手く行っているかどうかを確認し、必要に応じて学習を止める判断に利用できる
- CloudWatch のメトリックスダッシュボードで，グラフの可視化

学習スクリプトのメトリクス定義

- 自前のスクリプトについては，メトリクス正規表現で定義

```
from sagemaker.tuner import HyperparameterTuner, ContinuousParameter

# Configure HyperparameterTuner
my_tuner = HyperparameterTuner(estimator=my_estimator,  # previously-configured Estimator object
                               objective_metric_name='validation-accuracy',
                               hyperparameter_ranges={'learning-rate': ContinuousParameter(0.05, 0.06)},
                               metric_definitions=[{'Name': 'validation-accuracy', 'Regex':
                                                    'Regex': 'validation-accuracy=(\d\.\d+)'}],
                               max_jobs=100,
                               max_parallel_jobs=10)

# Start hyperparameter tuning job
my_tuner.fit({'train': 's3://my_bucket/my_training_data', 'test': 's3://my_bucket_my_testing_data'})

# Deploy best model
my_predictor = my_tuner.deploy(initial_instance_count=1, instance_type='ml.m4.xlarge')

# Make a prediction against the SageMaker endpoint
response = my_predictor.predict(my_prediction_data)
```

Q&A

ご質問については，AWS Japan Blog
「https://aws.amazon.com/jp/blogs/news/」
にて後日掲載します

公式Twitter/Facebook
AWSの最新情報をお届けします

@awscloud_jp

もしくは
http://on.fb.me/1vR8yWm

最新技術情報、イベント情報、お役立ち情報、
お得なキャンペーン情報などを日々更新しています！

AWS Well-Architected 個別技術相談会

毎週”W-A個別技術相談会”を実施中

- AWSのソリューションアーキテクト(SA)に対策などを相談することも可能
- 申込みはイベント告知サイトから

(https://aws.amazon.com/jp/about-aws/events/)

AWS イベント で[検索]