AWS re:Invent

AIM 401-R

Deep Learning Applications Using TensorFlow

Julien Simon
Principal Technical Evangelist, AI & Machine Learning
Amazon Web Services
@julsimon

AWS re:Invent

Agenda

- A quick overview of Amazon SageMaker
- A quick overview of TensorFlow
- TensorFlow on Amazon SageMaker
- Demo
- Resources

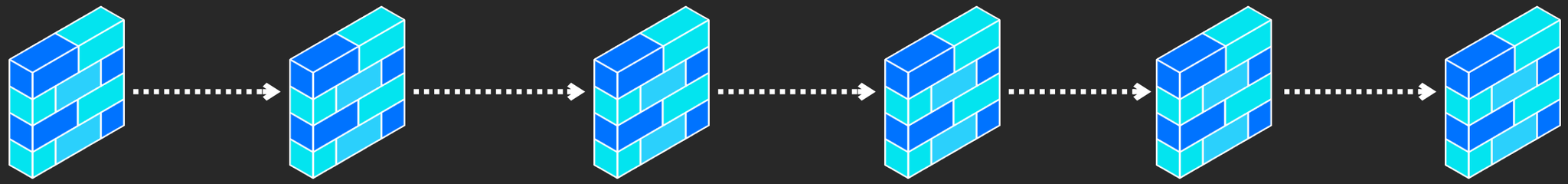
AWS re:Invent

Amazon SageMaker

AWS re:Invent

Amazon SageMaker

Easily build, train, and deploy Machine Learning models



Collect and prepare training data

Choose and optimize your ML algorithm

Set up and manage environments for training

Train and tune model (trial and error)

Deploy model in production

Scale and manage the production environment

FREE TIER

Amazon SageMaker

Notebook instances

Built-in, high-performance algorithms

ALGORITHMS	K-Means Clustering Principal Component Analysis Neural Topic Modelling Factorization Machines Linear Learner	XGBoost Latent Dirichlet Allocation Image Classification Seq2Seq, And more!
FRAMEWORKS	Apache MXNet, Chainer TensorFlow, PyTorch	Caffe2, CNTK, Torch

Set up and manage environments for training

Train and tune model (trial and error)

Deploy model in production

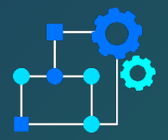
Scale and manage the production environment

Build

AWS re:Invent

Amazon SageMaker

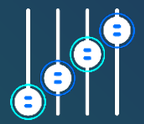
Notebook instances



Built-in, high-performance algorithms

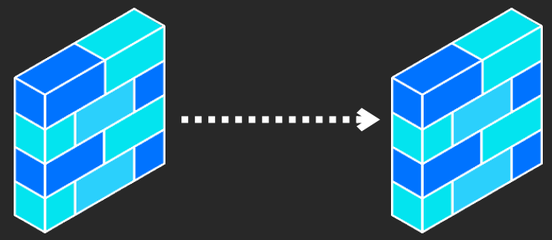
Build

One-click training



Automatic Model Tuning

Train



Deploy model in production

Scale and manage the production environment

AWS re:Invent

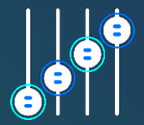
Amazon SageMaker

Notebook instances

Built-in, high-performance algorithms

Build

One-click training



Automatic Model Tuning

Train

One-click deployment

Fully managed hosting with auto-scaling

Deploy

Selected Amazon SageMaker customers

GE Healthcare

intuit

Hotels.com

DOW JONES

THOMSON REUTERS

edmunds.com

DigitalGlobe

Formula 1

tinder

zendesk

grammarly

realtor.com®

AWS re:Invent

TensorFlow

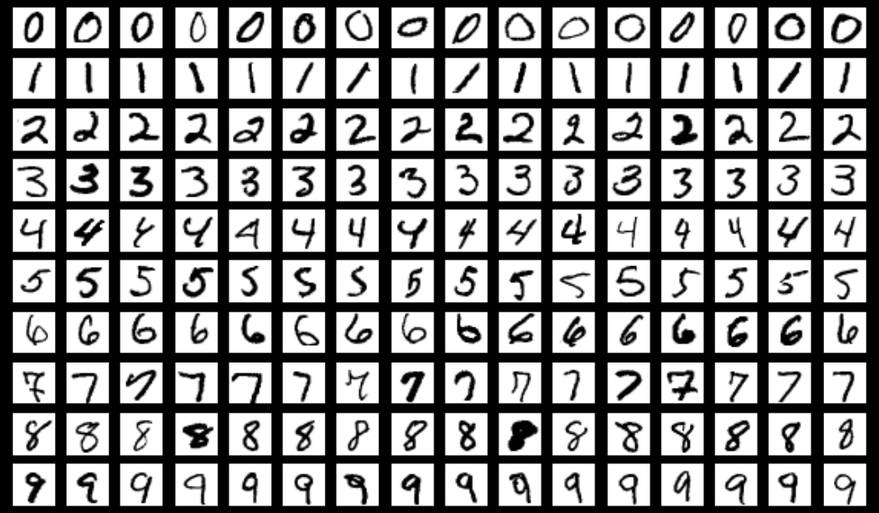
AWS re:Invent

TensorFlow

- Open source software library for Machine Learning
- Main API in Python, experimental support for other languages
- Built-in support for many network architectures: FC, CNN, LSTM, etc.
- Support for symbolic execution, as well as imperative execution since v1.7
  (aka eager execution)
- Complemented by the Keras high-level API

Example: MNIST with a Fully Connected network

```
import tensorflow as tf
mnist = tf.keras.datasets.mnist

(x_train, y_train),(x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(512, activation=tf.nn.relu),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```



AWS is the place of choice for TensorFlow workloads

"Of 388 projects, 80 percent using TensorFlow and other frameworks are running exclusively on AWS.

88% using only TensorFlow are running exclusively on AWS."

Nucleus Research report,
December 2017
https://aws.amazon.com/tensorflow

Selected customers running TensorFlow on AWS

Matrix Analytics

- The Colorado-based startup uses the Deep Learning AMI, TensorFlow and GPU instances to track disease progression for patients diagnosed with pulmonary nodules.
- Their tools are able to outperform previous methods in their ability to diagnose cancer from a CT scan.
- The software automates follow-up care in order to monitor changes to the patient's condition.

« Using the convenience of the [Deep Learning] AMI on AWS gives us the opportunity to offer up different business models, which allows us to become excellent technology partners as the market evolves at an ever-increasing pace. »

Dr. Aki Alzubaidi, Founder, Matrix Analytics

TensorFlow on Amazon SageMaker

AWS re:Invent

TensorFlow: a first-class citizen

- 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.
  - Supported versions: 1.4.1, 1.5.0, 1.6.0, 1.7.0, 1.8.0, 1.9.0, 1.10.0, 1.11.0

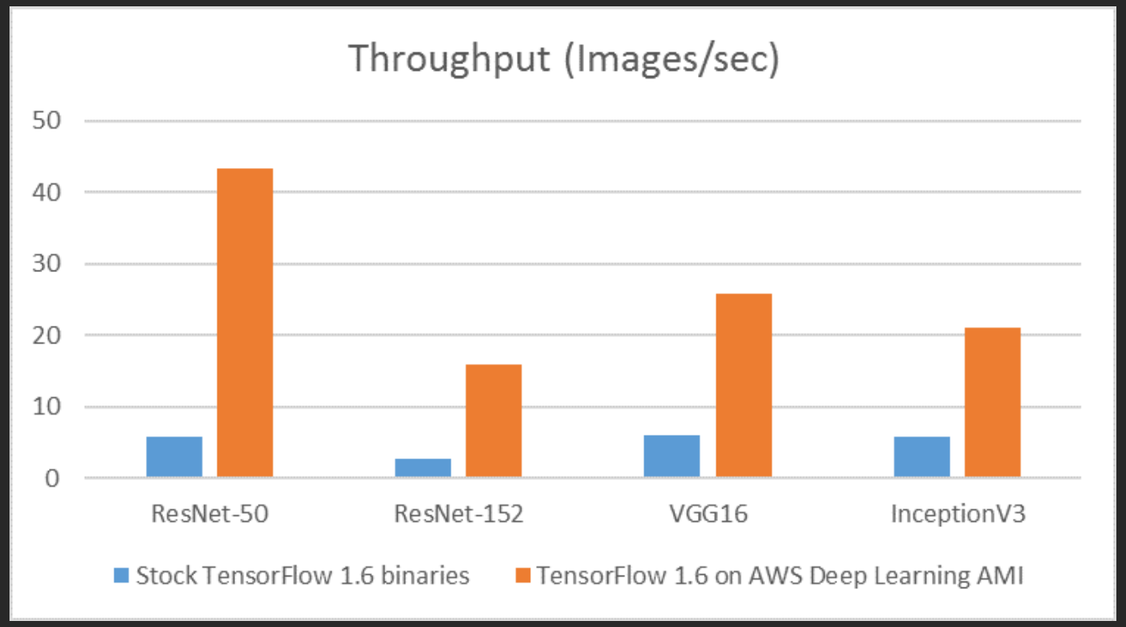
- Advanced features
  - Optimized both for GPUs and CPUs (Intel MKL-DNN library).
  - Distributed training.
  - Pipe mode.
  - TensorBoard.
  - Keras
  - Automatic Model Tuning

Training a TensorFlow model

- Simply add your own code.
  - Python 2.7 and Python 3.
  - Must implement *model_fn()* or *keras_model_fn()* or *estimator_fn()* to select a model
  - Must implement *train_input_fn()* to preprocess and load training data.
  - Must implement *eval_input_fn()* to preprocess and load evaluation data.
  - Optional: implement *serving_input_fn()* if the model will be deployed

```
from sagemaker.tensorflow import TensorFlow

tf_estimator = TensorFlow(entry_point='tf-train.py', role='SageMakerRole',
                          training_steps=10000, evaluation_steps=100,
                          train_instance_count=1, train_instance_type='ml.p3.2xlarge')

tf_estimator.fit('s3://bucket/path/to/training/data')
```

Optimizing TensorFlow training on C5



https://aws.amazon.com/blogs/machine-learning/faster-training-with-optimized-tensorflow-1-6-on-amazon-ec2-c5-and-p3-instances/ (March 2018)

Training a ResNet-50 benchmark with the synthetic ImageNet dataset using our optimized build of TensorFlow 1.11 on a c5.18xlarge instance type is 11X faster than training on the stock binaries.

https://aws.amazon.com/about-aws/whats-new/2018/10/chainer4-4_theano_1-0-2_launch_deep_learning_ami/ (October 2018)

Training a TensorFlow model in local mode

- You can train on the notebook instance itself, aka local mode.
- This is particularly useful while experimenting:
  you can save time and money by not firing up training instances.

```
from sagemaker.tensorflow import TensorFlow

tf_estimator = TensorFlow(entry_point='tf-train.py', role='SageMakerRole',
                          training_steps=10000, evaluation_steps=100,
                          train_instance_type='local')

tf_estimator.fit('s3://bucket/path/to/training/data')
```

Training a TensorFlow model on multiple instances

- Aka Distributed Training
- Amazon SageMaker takes care of all infrastructure setup.
- Only change required in your script: use global steps in model_fn().

```
from sagemaker.tensorflow import TensorFlow

tf_estimator = TensorFlow(entry_point='tf-train.py', role='SageMakerRole',
                          training_steps=10000, evaluation_steps=100,
                          train_instance_count=4, train_instance_type='ml.p3.2xlarge')

tf_estimator.fit('s3://bucket/path/to/training/data')
```

Training on infinitely large data sets with Pipe Mode

- By default, Amazon SageMaker copies the data set to all training instances.
  - This is the best option when the data set fits in memory.
- For larger data sets, Pipe Mode lets you stream data from Amazon S3.
  - Training starts faster.
  - You can train on infinitely large data sets.

```
from sagemaker.tensorflow import TensorFlow

tf_estimator = TensorFlow(entry_point='tf-train.py', role='SageMakerRole',
                          training_steps=10000, evaluation_steps=100,
                          train_instance_count=1, train_instance_type='ml.p3.2xlarge',
                          input_mode='Pipe')

tf_estimator.fit('s3://bucket/path/to/training/data')
```

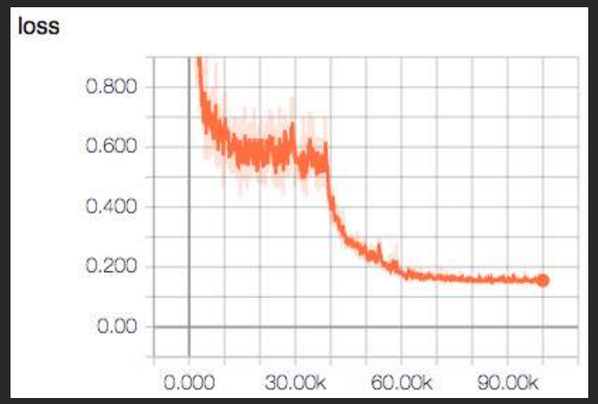
Streaming *TFRecord* files with Pipe Mode

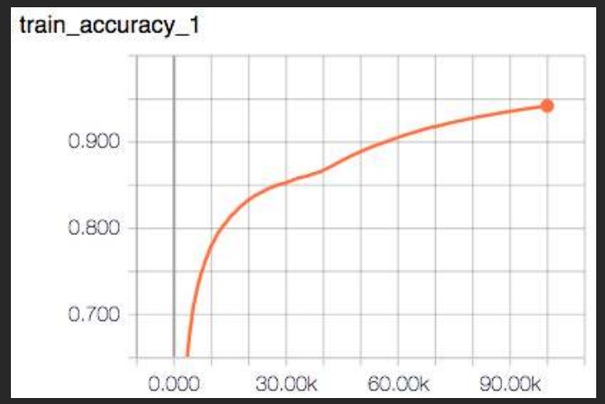
```
from sagemaker_tensorflow import PipeModeDataset

features = { 'data': tf.FixedLenFeature([], tf.string),
             'labels': tf.FixedLenFeature([], tf.int64), }

def parse(record):
       parsed = tf.parse_single_example(record, features)
       return ({ 'data': tf.decode_raw(parsed['data'], tf.float64) }, parsed['labels’])

def train_input_fn(training_dir, hyperparameters):
       ds = PipeModeDataset(channel='training', record_format='TFRecord’)
       ds = ds.repeat(20)
       ds = ds.prefetch(10)
       ds = ds.map(parse, num_parallel_calls=10)
       ds = ds.batch(64)
       return ds
```

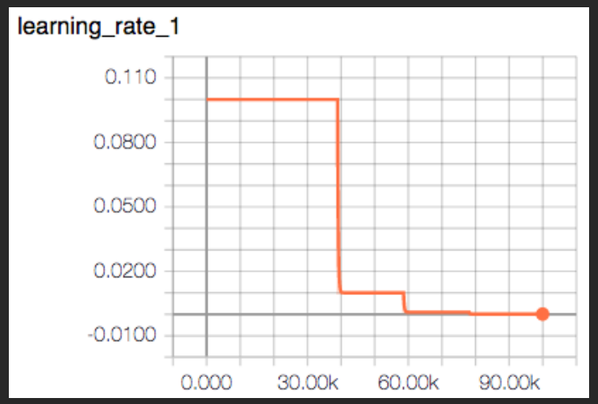
Visualizing training with TensorBoard

- TensorBoard is a suite of visualization tools: graph, metrics, etc.
- When enabled, it will run on the notebook instance.
- You can access it at https://NOTEBOOK_INSTANCE/proxy/6006/

```
tf_estimator.fit(inputs, run_tensorboard_locally=True)
```







Deploying a TensorFlow model to an HTTPS endpoint

Model trained on Amazon SageMaker

```
from sagemaker.tensorflow import TensorFlow
tf_estimator = TensorFlow(entry_point='tf-train.py', ..., train_instance_count=1,
                          train_instance_type='ml.c4.xlarge’)
tf_estimator.fit(inputs)
predictor = tf_estimator.deploy(initial_instance_count=1, instance_type='ml.c4.xlarge')
```

Model trained elsewhere

```
from sagemaker.tensorflow import TensorFlowModel
tf_model = TensorFlowModel(model_data='s3://mybucket/model.tar.gz’, …,
                                 entry_point='entry.py', name='model_name’)
predictor = tf_model.deploy(initial_instance_count=1, instance_type='ml.c4.xlarge')
```

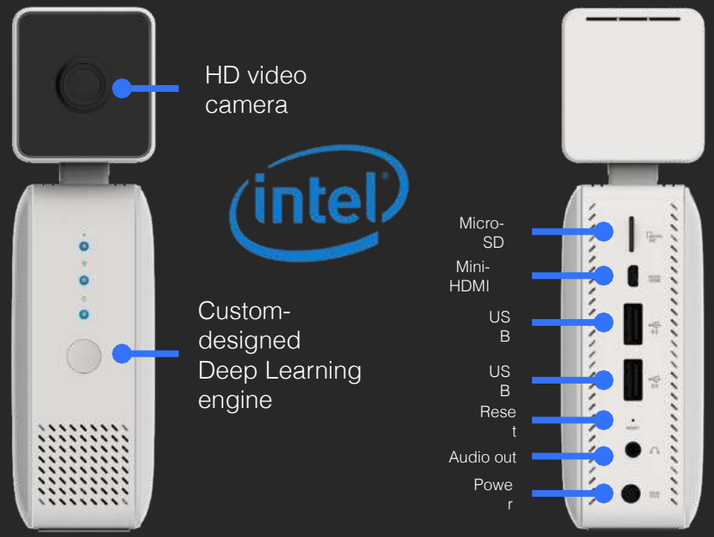
Using Keras on Amazon SageMaker

- Keras is a popular API running on top of TF, Theano and Apache MXNet.
- The *tf.keras* API is natively supported in Amazon SageMaker
- To use Keras itself (*keras.**), you need to build a custom container.
- This is not difficult!
  - Write a Dockerfile.
  - Build the container.
  - Push it to Amazon ECR.
  - Use it with *sagemaker.estimator.Estimator*.

- Full instructions and demo in this AWS Innovate talk: https://www.youtube.com/watch?v=c8Nhwr9VmfM

Using TensorFlow with AWS DeepLens

- AWS DeepLens can run TensorFlow models.
  - Inception
  - MobileNet
  - NasNet
  - ResNet
  - VGG
- Train or fine-tune your model on Amazon SageMaker.
- Deploy to DeepLens through AWS Greengrass.



Demo: custom Keras container

Training
- Passing hyper parameters from Amazon SageMaker
- Setting up Keras image augmentation
- Defining Keras callbacks for early stopping and checkpointing
- Logging training metrics in Amazon CloudWatch

Tuning
- Optimizing hyper parameters with Automatic Model Tuning
- Defining a custom tuning metric
- Defining a custom Keras callback to log the tuning metric
- Plotting tuning results

Resources

AWS re:Invent

Resources

https://ml.aws

https://tensorflow.org/
https://keras.io/

https://aws.amazon.com/sagemaker
https://github.com/awslabs/amazon-sagemaker-examples
https://github.com/aws/sagemaker-python-sdk

https://medium.com/@julsimon

Breakout repeats

Thursday, November 29th

Deep Learning Applications using Tensorflow , featuring Siemens Financial Services

3:15 PM - 4:15 PM | Venetian, Level 2, Venetian Theatre

Friday, November 30th

Deep Learning Applications using Tensorflow, featuring Advanced Microgrid Systems

10:45AM – 11:45AM | Venetian, Level 5, Palazzo O

AWS re:Invent

Please complete the session survey in the mobile app.

Thank you!

Julien Simon
Principal Technical Evangelist, AI & Machine Learning
Amazon Web Services
@julsimon

AWS re:Invent