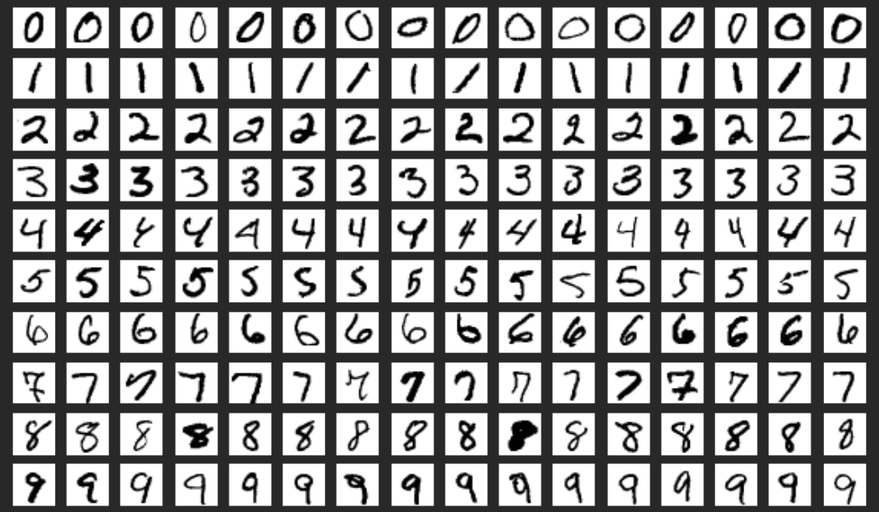
TensorFlow

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

TF1.x: 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: The platform of choice for TensorFlow

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

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

- 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

Training a TensorFlow model

- Script mode: simply add your own code.
  - Python 3 required
  - Hyperparameters are passed as command-line arguments
  - Location of training and validation sets are passed as environment variables
  - Location where model must be saved is passed as an environment variable

```
from sagemaker.tensorflow import TensorFlow

tf_estimator = TensorFlow(
       entry_point='my_script.py',
       role=role,
       train_instance_count=1, train_instance_type='ml.p3.2xlarge',
       framework_version='1.15', py_version='py3', script_mode=True,
       hyperparameters={'epochs': 10} )
)

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

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='my_script.py',
       role=role,
       train_instance_count=1, train_instance_type='local',   # or 'local_gpu'
       framework_version='1.15', py_version='py3', script_mode=True,
       hyperparameters={'epochs': 10} )
)

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

Training a TensorFlow model on multiple instances

- Aka Distributed Training
- Parameter Server (native mode), or Horovod
- Amazon SageMaker takes care of all infrastructure setup.

```
from sagemaker.tensorflow import TensorFlow

tf_estimator = TensorFlow(
       entry_point='my_script.py',
       role=role,
       train_instance_count=8, train_instance_type='ml.p3.2xlarge',
       framework_version='1.15', py_version='py3', script_mode=True,
       hyperparameters={'epochs': 10} )
)

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='my_script.py',
       role=role,
       train_instance_count=1, train_instance_type='ml.p3.2xlarge',
       framework_version='1.15', py_version='py3', script_mode=True,
       input_mode='Pipe'
)

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

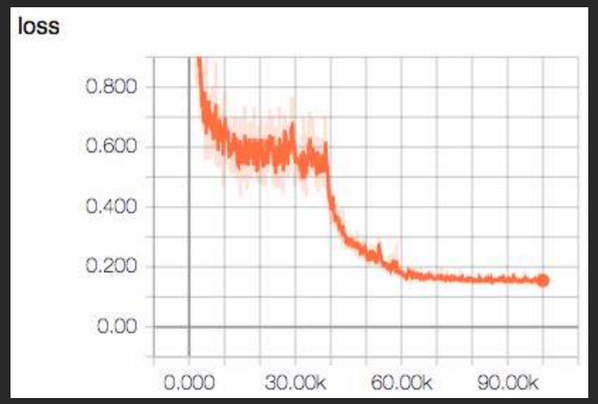
Streaming *TFRecord* files with Pipe Mode

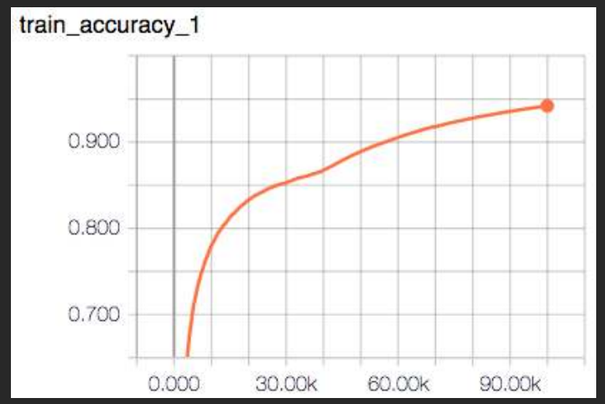
```
from sagemaker_tensorflow import PipeModeDataset

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
```

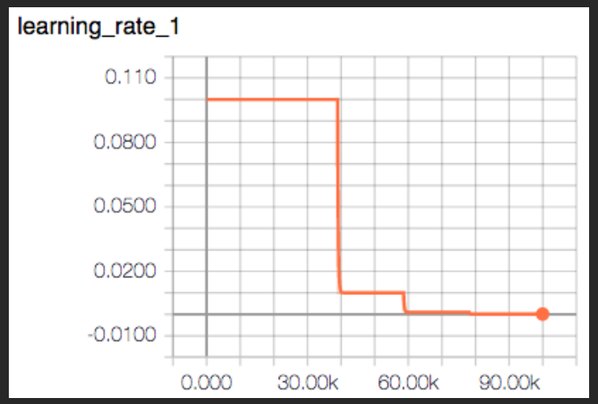
Visualizing training with TensorBoard

- 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-demand

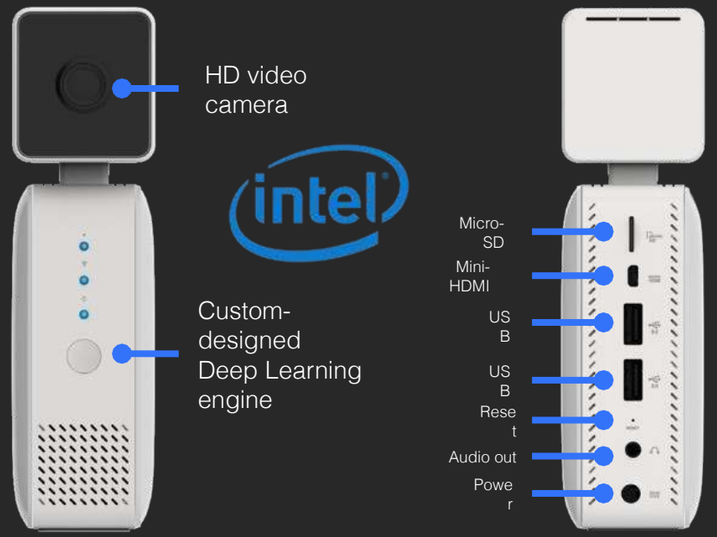
```
from sagemaker.tensorflow import TensorFlow

tf_estimator = TensorFlow(entry_point='tf-train.py', ...)
tf_estimator.fit(inputs)

predictor = tf_estimator.deploy(initial_instance_count=1, instance_type='ml.c4.xlarge')
```

Pretrained Model

```
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 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.



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/aim410 : End to end demo with Keras & SageMaker