深度学习实践
from Tensorflow to AI-Hub
王顺 – Google Cloud
1 从hello world开始
以深度学习的第一个案例MNIST为例
学习Tensorflow框架的使用及代码编写风格
理解TF

开始使用 TensorFlow

TensorFlow 是一个用于研究和生产的开源机器学习库。TensorFlow 提供了各种 API，可供初学者和专家在桌面、移动、网络和云端环境下进行开发。请参阅以下几部分，了解如何开始使用。

学习和使用机器学习

1. 基本分类
2. 文本分类
3. 回归
4. 过拟合和欠拟合
5. 保存和加载

阅读 Keras 指南

```python
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(input_shape=(28, 28)),
                                     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)
```

在 Google 的交互式笔记本中试一试
Mac CPU运行结果

```
[shunwang-macbookpro:scripts shunwang$ python3 0921_tf2_01_mnist.py
uctions that this TensorFlow binary was not compiled to use: AVX2 FMA
WARNING:tensorflow:From /usr/local/lib/python3.7/site-packages/tensorflow/python/ops/math_grad.py:1250:
  add_dispatch_support.<locals>._wrapper (from tensorflow.python.ops.array_ops) is deprecated and will be
  removed in a future version.
  Instructions for updating:
  Use tf.where in 2.0, which has the same broadcast rule as np.where
Train on 60000 samples
Epoch 1/5
60000/60000 [============================] - 4s 72us/sample - loss: 0.2205 - accuracy: 0.9353
Epoch 2/5
60000/60000 [============================] - 4s 69us/sample - loss: 0.0975 - accuracy: 0.9701
Epoch 3/5
60000/60000 [============================] - 4s 67us/sample - loss: 0.0690 - accuracy: 0.9784
Epoch 4/5
60000/60000 [============================] - 4s 70us/sample - loss: 0.0524 - accuracy: 0.9831
Epoch 5/5
60000/60000 [============================] - 4s 69us/sample - loss: 0.0417 - accuracy: 0.9862
10000/10000 [============================] - 0s 34us/sample - loss: 0.0681 - accuracy: 0.9791
```
GPU运行结果

[2019-09-19 03:35:21.392195: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1326] Created TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:0 with 10805 MB memory) -> physical GPU (device: 0, name: Tesla K80, pci bus id: 0000:00:04.0, compute capability: 3.7)
WARNING:tensorflow:From /usr/local/lib/python3.6/site-packages/tensorflow/python/ops/math_grad.py:1250: add_dispatch_support.<locals>.wrapper (from tensorflow.python.ops.array_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
Train on 60000 samples
Epoch 1/5
2019-09-19 03:35:23.088163: I tensorflow/stream_executor/platform/default/dso_loader.cc:42] Successfully opened dynamic library libcublas.so.10.0
60000/60000 [==============================] - 9s 150us/sample - loss: 0.2215 - accuracy: 0.9348
Epoch 2/5
60000/60000 [==============================] - 9s 147us/sample - loss: 0.0972 - accuracy: 0.9703
Epoch 3/5
60000/60000 [==============================] - 9s 148us/sample - loss: 0.0690 - accuracy: 0.9784
Epoch 4/5
60000/60000 [==============================] - 9s 149us/sample - loss: 0.0527 - accuracy: 0.9830
Epoch 5/5
60000/60000 [==============================] - 9s 146us/sample - loss: 0.0437 - accuracy: 0.9858
10000/10000 [==============================] - 1s 79us/sample - loss: 0.0671 - accuracy: 0.9789
TPU运行结果

2019-09-19 15:23:28.128919: W tensorflow/core/distributed_runtime/rpc/grpc_session.cc:370] GrpcSession::ListDevices will initialize the session with an empty graph and other defaults because the session has not yet been created.
WARNING:tensorflow:From /usr/local/lib/python3.5/dist-packages/tensorflow_core/python/ops/resource_variable_ops.py:1633: calling BaseResourceVariable.__init__ (from tensorflow.python.ops.resource_variable_ops) with constraint is deprecated and will be removed in a future version.
Instructions for updating:
If using Keras pass *constraint* arguments to layers.
WARNING:tensorflow:From /usr/local/lib/python3.5/dist-packages/tensorflow_core/python/distribute/values.py:882: BaseResourceVariable.constraint (from tensorflow.python.ops.resource_variable_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Apply a constraint manually following the optimizer update step.
Epoch 1/5
1875/1875 [==============================] - 6s 3ms/step - loss: 0.2196 - acc: 0.9346
Epoch 2/5
1875/1875 [==============================] - 6s 3ms/step - loss: 0.0976 - acc: 0.9700
Epoch 3/5
1875/1875 [==============================] - 6s 3ms/step - loss: 0.0687 - acc: 0.9784
Epoch 4/5
1875/1875 [==============================] - 6s 3ms/step - loss: 0.0536 - acc: 0.9825
Epoch 5/5
1875/1875 [==============================] - 6s 3ms/step - loss: 0.0416 - acc: 0.9861

shunwang@tpuvm:~/tfv15
TPU的创建和使用

```python
import os
import tensorflow as tf
from tensorflow.contrib import tpu
from tensorflow.contrib.cluster_resolver import TPUClusterResolver

def f(x, y):
    return x + y

inputs = {
    3.5,
    tf.ones((3, 3), tf.float32),
    tf.ones((3, 3), tf.float32),
}

tpu_compilation = tpu.rename(axy_computation, inputs)

tpu_grpc_url = TPUClusterResolver(
    tpu_cluster_resolver=tpu_cluster_resolver.get_master())

with tf.compat.v1.Session(tpu_grpc_url) as sess:
    sess.run([f(1, 2), f(3, 4)])
```

Please upgrade your code to TensorFlow 2.0:
* [https://www.tensorflow.org/guide/migration_guide](https://www.tensorflow.org/guide/migration_guide)

Or install the latest stable TensorFlow 1.X release:
* `pip install -U tensorflow==1.X`

Otherwise your code may be broken by the change.

Labels and description
```
[array([4., 4., 4.],
       [4., 4., 4.]),
       dtype=float32)]
```

You will be billed for this model.
TPU训练MNIST的改动

```
Traceback (most recent call last):
  File "03_mnist_tpu.py", line 32, in <module>
    model.fit(x_train, y_train, epochs=5)
File "~/.local/lib/python3.5/dist-packages/tensorflow_core/python/keras/engine/training.py", line 733, in fit
    use_multiprocessing=use_multiprocessing)
File "~/.local/lib/python3.5/dist-packages/tensorflow_core/python/keras/engine/training_distributed.py", line 666, in fit
    validation_freq=validation_freq)
File "~/.local/lib/python3.5/dist-packages/tensorflow_core/python/keras/engine/training_distributed.py", line 229, in experimental_tpu_fit_loop
    dist_utils.reset_metrics(model)
File "~/.local/lib/python3.5/dist-packages/tensorflow_core/python/keras/distribute/distributed_training_utils.py", line 103
  in reset_metrics
    first_model.reset_metrics()
File "~/.local/lib/python3.5/dist-packages/tensorflow_core/python/keras/engine/training.py", line 920, in reset_metrics
    m.reset_state()
File "~/.local/lib/python3.5/dist-packages/tensorflow_core/python/keras/metrics.py", line 210, in reset_state
    K.batch_set_value([(v, 0) for v in self.variables])
File "~/.local/lib/python3.5/dist-packages/tensorflow_core/python/keras/backend.py", line 3259, in batch_set_value
    get_session().run(assign_ops, feed_dict=feed_dict)
File "~/.local/lib/python3.5/dist-packages/tensorflow_core/python/keras/backend.py", line 486, in get_session
    initialize_variables(session)
File "~/.local/lib/python3.5/dist-packages/tensorflow_core/python/keras/backend.py", line 903, in _initialize_variables
    variables.module.is_variable_initialized(v) for v in candidate_vars)
File "~/.local/lib/python3.5/dist-packages/tensorflow_core/python/client/session.py", line 956, in run
    run_metadata_ptr)
File "~/.local/lib/python3.5/dist-packages/tensorflow_core/python/client/session.py", line 1180, in _run
    feed_dict_tensor, options, run_metadata)
File "~/.local/lib/python3.5/dist-packages/tensorflow_core/python/client/session.py", line 1359, in _do_run
    run_metadata)
File "~/.local/lib/python3.5/dist-packages/tensorflow_core/python/client/session.py", line 1384, in _do_call
raise type(e)(node_def, op, message)
tensorflow.python.framework.errors_impl.InvalidArgumentError: Unsupported data type for TPU: double
  caused by output IteratorGetNext:
```
TPU训练MNIST的改动

```python
import numpy as np
import os
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
y_train, y_test = y_train.astype(np.int64), y_test.astype(np.int64)

resolver = tf.distribute.cluster_resolver.TPUClusterResolver()
tf.tpu.experimental.initialize_tpu_system(resolver)
tpu_strategy = tf.distribute.experimental.TPUStrategy(resolver)

with strategy.scope():
    model = tf.keras.models.Sequential([tf.keras.layers.Flatten(input_shape=(28, 28))])
    model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
    model.fit(x_train, y_train, epochs=5)
    model.evaluate(x_test, y_test)
```

★ Note: When using TPUStrategy with TPU pods with Keras, currently the user will have to explicitly shard or shuffle the data for different workers, but we will change this in the future to automatically shard the input data intelligently.

https://www.tensorflow.org/guide/distribute_strategy
2 初步修改
针对第一个python代码执行、思考和改进
如何能做的更好？
TPU Pod

TPU v3-32
(32 cores, 4x4 slice)

TPU v3-128
(128 cores, 8x8 slice)

TPU v3-512
(512 cores, 16x16 slice)

TPU v3-1024
(1024 cores, 16x32 slice)
BERT 训练时间短

Data and TPU/GPU Runtime

BERT was trained using 3.3 Billion words total with 2.5B from Wikipedia and 0.8B from BooksCorpus. The training was done using TPU, while GPU estimates are shown below.

<table>
<thead>
<tr>
<th>Model</th>
<th>TPU Pod</th>
<th>TPU Chips</th>
<th>TPU Cores¹</th>
<th>PFLOPS²</th>
<th>GPU³</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT_BASE</td>
<td>4 x 4 days</td>
<td>16 x 4 days</td>
<td>32 x 4 days</td>
<td>0.7 x 4 days</td>
<td>2 x 50-70 days</td>
</tr>
<tr>
<td>BERT_LARGE</td>
<td>16 x 4 days</td>
<td>64 x 4 days</td>
<td>128 x 4 days</td>
<td>2.9 x 4 days</td>
<td>8 x 50-70 days</td>
</tr>
</tbody>
</table>

Training devices and times for BERT; used TPU and estimated for GPU.

Fine-tuning was done using 2.5K to 392K labeled samples. Importantly, datasets above 100K training samples showed robust performance over various hyper-parameters. Each fine-tuning experiment runs within 1 hour on a single cloud TPU and few hours on GPU.

https://github.com/google-research/bert
Cloud TPU v3 Pod Performance: Two Examples

Train ResNet-50 ImageNet image classification model from scratch in <2 minutes on full v3 Pod.

Process more than 1.05M images / second along the way!

Train BERT language representation from scratch in just 76 minutes on full v3 Pod.

Training BERT takes days on smaller systems.
What about these data-driven organizations enables them to use data to gain a competitive advantage? In *Building Data Science Teams*, we said that a data-driven organization requires high-quality data in order to build high-quality models. The data needs to be clean, well-organized, and error-free. **Cleaning the data is often the most taxing part of data science, and is frequently 80% of the work.** Setting up the process to clean data at scale adds further complexity. Successful organizations invest heavily in tooling, processes, and regular audits. They have developed a culture that understands the importance of data quality; otherwise, as the adage goes, *garbage in, garbage out.*
• Why are my tip predictions bad in the morning hours?

Chicago Taxi Cab Dataset
Tensorflow/Keras中的网络

Module: tf.keras.applications

Keras Applications are canned architectures with pre-trained weights.

- **densenet** module: DenseNet models for Keras.
- **imagenet_utils** module: Utilities for ImageNet data preprocessing & prediction decc
- **inception_resnet_v2** module: Inception-ResNet V2 model for Keras.
- **inception_v3** module: Inception V3 model for Keras.
- **mobilenet** module: MobileNet v1 models for Keras.
- **mobilenet_v2** module: MobileNet v2 models for Keras.
- **nasnet** module: NASNet-A models for Keras.
- **resnet** module: ResNet models for Keras.
- **resnet50** module: Public API for tf.keras.applications.resnet50 namespace.
- **resnet_v2** module: ResNet v2 models for Keras.
- **vgg16** module: VGG16 model for Keras.
- **vgg19** module: VGG19 model for Keras.
- **xception** module: Xception V1 model for Keras.
Custom training with TPUs

- https://www.tensorflow.org/tutorials/distribute/tpu_custom_training

This tutorial will take you through using tf.distribute.experimental.TPUStrategy. This is a new strategy, a part of tf.distribute.Strategy, that allows users to easily switch their model to using TPUs. As part of this tutorial, you will create a Keras model and take it through a custom training loop (instead of calling fit method).

You should be able to understand what is a strategy and why it’s necessary in Tensorflow. This will help you switch between CPU, GPUs, and other device configurations more easily once you understand the strategy framework. To make the introduction easier, you will also make a Keras model that produces a simple convolutional neural network. A Keras model usually is trained in one line of code (by calling its fit method), but because some users require additional customization, we showcase how to use custom training loops. Distribution Strategy was originally written by DeepMind -- you can read the story here.
3 业务升级

以上已经针对MNIST做了一些深入学习
接下来思考如何满足实际业务上的需要
Integrated Frontend for Job Management, Monitoring, Debugging, Data/Model/Evaluation Visualization

Shared Configuration Framework and Job Orchestration

Data Ingestion | Data Analysis + Validation | Data Transformation | Trainer | Model Evaluation and Validation | Serving | Logging

Shared Utilities for Garbage Collection, Data Access Controls

Pipeline Storage

Figure 1: High-level component overview of a machine learning platform.
Component: ExampleGen

Inputs and Outputs

Raw Data

CSV
TF Record

Example Gen

Split TF Record Data

Training
Eval

Configuration

```python
examples = csv_input(os.path.join(data_root, 'simple'))
example_gen = CsvExampleGen(input_base=examples)
```
Component: StatisticsGen

Inputs and Outputs

ExampleGen -> StatisticsGen -> Statistics

Data

StatisticsGen

Configuration

\[
\text{statistics\_gen} = \text{StatisticsGen}(\text{input\_data} = \text{example\_gen}.\text{outputs}.\text{examples})
\]

Visualization

- **Numeric Features (15)**
  - **fare**: 10,000, 0%, 11.74, 12.13, 0.17%, 0, 7.85, 700.07
  - **trip\_start\_hour**: 10,000, 0%, 13.63, 6.61, 4.14%, 0, 15, 23
  - **dropoff\_census\_tract**: 7,148, 28.52%, 17.08, 331k, 0%, 17.08, 17.08
  - **trip\_start\_timestamp**: 10,000, 0%, 1.41B, 29.2M, 0%, 1.36B, 1.41B, 1.48B

- **Chart to show**: Standard
Component: SchemaGen

**Inputs and Outputs**

- StatisticsGen
- Statistics
- SchemaGen
- Schema

**Configuration**

```python
infer_schema = SchemaGen(stats=statistics_gen.outputs.output)
```

**Visualization**

<table>
<thead>
<tr>
<th>Feature name</th>
<th>Type</th>
<th>Presence</th>
<th>Valency</th>
<th>Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>'fare'</td>
<td>FLOAT</td>
<td>required</td>
<td>single</td>
<td>-</td>
</tr>
<tr>
<td>'trip_start_hour'</td>
<td>INT</td>
<td>required</td>
<td>single</td>
<td>-</td>
</tr>
<tr>
<td>'pickup_census_tract'</td>
<td>BYTES</td>
<td>optional</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>'dropoff_census_tract'</td>
<td>FLOAT</td>
<td>optional</td>
<td>single</td>
<td>-</td>
</tr>
<tr>
<td>'company'</td>
<td>STRING</td>
<td>optional</td>
<td>single</td>
<td>'company'</td>
</tr>
<tr>
<td>'trip_start_timestamp'</td>
<td>INT</td>
<td>required</td>
<td>single</td>
<td>-</td>
</tr>
</tbody>
</table>
Component: ExampleValidator

**Inputs and Outputs**

- StatisticsGen
- SchemaGen

**Example Validator**

- Statistics
- Schema

**Anomalies Report**

**Configuration**

```python
validate_stats = ExampleValidator(
    stats=statistics_gen.outputs.output,
    schema=infer_schema.outputs.output)
```

**Visualization**

<table>
<thead>
<tr>
<th>Feature name</th>
<th>Anomaly short description</th>
<th>Anomaly long description</th>
</tr>
</thead>
<tbody>
<tr>
<td>'payment_type'</td>
<td>Unexpected string values</td>
<td>Examples contain values missing from the schema: Prasad (&lt;1%).</td>
</tr>
<tr>
<td>'company'</td>
<td>Unexpected string values</td>
<td>Examples contain values missing from the schema: Prasad (&lt;1%).</td>
</tr>
</tbody>
</table>
Component: Transform

```
transform = Transform(
    input_data=example_gen.outputs.examples,
    schema=infer_schema.outputs.output,
    module_file=taxi_module_file)
```

```
for key in _DENSE_FLOAT_FEATURE_KEYS:
    outputs[_transformed_name(key)] = transform.scale_to_z_score(
        _fill_in_missing(inputs[key]))
# ...
```

```
outputs[_transformed_name(_LABEL_KEY)] = tf.where(
    tf.is_nan(taxi_fare),
    tf.cast(tf.zeros_like(taxi_fare), tf.int64),
    # Test if the tip was > 20% of the fare.
    tf.cast(
        tf.greater(tips, tf.multiply(taxi_fare, tf.constant(0.2))), tf.int64))
# ...
Component: Trainer

**Configuration**

```python
trainer = Trainer(
    module_file=taxi_module_file,
    transformed_examples=transform.outputs.transformed_examples,
    schema=infer_schema.outputs.output,
    transform_output=transform.outputs.transform_output,
    train_steps=10000,
    eval_steps=5000,
    warm_starting=True)
```

**Code: Just TensorFlow :)**
Component: Evaluator

Inputs and Outputs

- ExampleGen
- Trainer

Data → Model → Evaluator → Evaluation Metrics

Configuration

```
model_analyzer = Evaluator(
    examples=examples_gen.outputs.output,
    eval_spec=taxi_eval_spec,
    model_exports=trainer.outputs.output)
```

Visualization

![Visualization of model analyzer outputs and metrics](image)
Component: ModelValidator

Inputs and Outputs

ExampleGen

Data

Trainer

Model (x2)

Model Validator

Validation Outcome

Configuration

```python
model_validator = ModelValidator(
    examples=examples_gen.outputs.output,
    model=trainer.outputs.output,
    eval_spec=taxi_mv_spec)
```

- **Configuration options**
  - Validate using current eval data
  - “Next-day eval”, validate using unseen data
Component: Pusher

Inputs and Outputs

- Model Validator
- Validation Outcome
- Pusher
- Deployment Options

Configuration

```python
pusher = Pusher(
    model_export=trainer.outputs.output,
    model_blessing=model_validator.outputs.blessing,
    serving_model_dir=serving_model_dir)
```

- Block push on validation outcome
- Push destinations supported today
  - Filesystem
  - TF Serving model server
4 实践指南

实践经验小结
1. Data

Data Preparation & Data Augmentation

Improving Deep Learning Performance with AutoAugment
Monday, June 4, 2018

Posted by
SpecAugment: A New Data Augmentation Method for Automatic Speech Recognition
Monday, April 22, 2019

Posted by Daniel S. Park, AI Resident and William Chan, Research Scientist
2. Tensorboard
3. Fine tune

DAWNBench
An End-to-End Deep Learning Benchmark and Competition

DAWNBench is a benchmark suite for end-to-end deep learning training and inference. Computation time and cost are critical resources in building deep models, yet many existing benchmarks focus solely on model accuracy. DAWNBench provides a reference set of common deep learning workloads for quantifying training time, training cost, inference latency, and inference cost across different optimization strategies, model architectures, software frameworks, cloud, and hardware.

The first iteration of DAWNBench is over, and the competition results and key takeaways have been finalized. However, we are still curious to see how well people can do on this benchmark and are now accepting rolling submissions. The original results before the April 20, 2018 deadline are archived for reference. For a more comprehensive benchmark, please consider submitting to the updated MLPerf benchmark.

Image Classification on ImageNet

Training Time

Objective: Time taken to train an image classification model to a top-5 validation accuracy of 93% or greater on ImageNet.
4. checkpoint
5. Pipeline

TFX Config

Training + Eval Data -> ExampleGen -> StatisticsGen -> SchemaGen -> Transform -> Trainer

Evaluator -> Model Validator -> Pusher

Airflow Runtime
Kubeflow Runtime

Metadata Store

TensorFlow Serving
TensorFlow Hub
TensorFlow Lite
TensorFlow JS

Kubeflow Runtime

- ExampleGen
- StatisticsGen
- SchemaGen
- Transform
- Trainer
- Evaluator
- Model Validator
- Pusher

Airflow Runtime

- ExampleGen
- StatisticsGen
- SchemaGen
- Transform
- Trainer
- Evaluator
- Model Validator
- Pusher

Metadata Store

- TensorFlow Serving
- TensorFlow Hub
- TensorFlow Lite
- TensorFlow JS
6. 协作

Explore production-ready AI services & solutions from Google

- **AutoML Natural Language**
  By Google
  Build custom machine learning models to classify, extract, and
detect sentiment in text

- **Tensorflow Deep Learning VM Images**
  By Google
  Preconfigured VMs for deep learning applications

- **Training a Faster RCNN object detection model using**
  By Google
  The Faster RCNN component provides an easy way for users to
train their own Faster RCNN object detection model on GPUs or TPUs

- **AutoML Tables**
  By Google
  Automated feature engineering for tabular data e.g. for Energy Forecasting

- **Big Query XGBoost Pipeline**
  By Google
  A template Kubeflow pipeline for using XGBoost model training and prediction.

Cutting-edge AI research from Google & DeepMind

- **BERT**
  By Google
  Bidirectional Encoder Representations from

- **Big GAN**
  By DeepMind
  BigGAN image generator trained on S12xS12 ImageNet.

- **Universal Sentence Encoder**
  By Google
  Learn how to access the Universal Sentence Encoder in TF-Hub and use it for sentence similarity and

- **Inception V3**
  By Google
  Feature vectors of images with Inception V3 trained on ImageNet (ILSVRC-2015-CLS)

- **ResNet V2**
  By Google
  Feature vectors of images with ResNet V2 152 trained on Imagenet (ILSVRC-2012-CLS)
Takeaways

• 在不同设备上执行训练
• 基于AI产品的全流程
• 深度学习实践：
  • 质量
  • 效率
  • 专注
  • 稳定
• 参与和行动！！！
THANK YOU

希望对大家有所帮助和启发