lab_exercise

October 5, 2021

1 Continuous training with TFX and Google Cloud AI Platform

1.1 Learning Objectives

1. Use the TFX CLI to build a TFX pipeline.
2. Deploy a TFX pipeline version with tuning enabled to a hosted AI Platform Pipelines instance.
3. Create and monitor a TFX pipeline run using the TFX CLI and KFP UI.

In this lab, you use utilize the following tools and services to deploy and run a TFX pipeline on Google Cloud that automates the development and deployment of a TensorFlow 2.3 WideDeep Classifer to predict forest cover from cartographic data:

- The **TFX CLI** utility to build and deploy a TFX pipeline.
- A hosted **AI Platform Pipeline instance (Kubeflow Pipelines)** for TFX pipeline orchestration.
- **Dataflow** jobs for scalable, distributed data processing for TFX components.
- A **AI Platform Training** job for model training and flock management of tuning trials.
- **AI Platform Prediction**, a model server destination for blessed pipeline model versions.
- **CloudTuner** (KerasTuner implementation) and **AI Platform Vizier** for advanced model hyperparameter tuning.

You will then create and monitor pipeline runs using the TFX CLI as well as the KFP UI.

1.1.1 Setup

Update lab environment PATH to include TFX CLI and skaffold

[4]:
```
import yaml

# Set `PATH` to include the directory containing TFX CLI and skaffold.
PATH=%env PATH
%env PATH=/home/jupyter/.local/bin:{PATH}
```

```
env: PATH=/home/jupyter/.local/bin:/opt/conda/bin:/opt/conda/condabin:/usr/local
/bin:/usr/bin:/bin:/usr/local/games:/usr/games
```

Validate lab package version installation

[5]:
```
!python -c "import tensorflow; print('TF version: {}'.format(tensorflow.
 ↪__version__))"
!python -c "import tfx; print('TFX version: {}'.format(tfx.__version__))"
```

```
!python -c "import kfp; print('KFP version: {}'.format(kfp.__version__))"
```

```
2021-10-05 20:40:33.838456: W
tensorflow/stream_executor/platform/default/dso_loader.cc:59] Could not load
dynamic library 'libcudart.so.10.1'; dlerror: libcudart.so.10.1: cannot open
shared object file: No such file or directory
2021-10-05 20:40:33.838607: I tensorflow/stream_executor/cuda/cudart_stub.cc:29]
Ignore above cudart dlerror if you do not have a GPU set up on your machine.
TF version: 2.3.2
TFX version: 0.25.0
KFP version: 1.4.0
```

Note: this lab was built and tested with the following package versions:

```
TF version: 2.3.2
TFX version: 0.25.0
KFP version: 1.4.0
```

(Optional) If running the above command results in different package versions or you receive an import error, upgrade to the correct versions by running the cell below:

[ ]:
```
%pip install --upgrade --user tensorflow==2.3.2
%pip install --upgrade --user tfx==0.25.0
%pip install --upgrade --user kfp==1.4.0
```

Note: you may need to restart the kernel to pick up the correct package versions.

Validate creation of AI Platform Pipelines cluster Navigate to AI Platform Pipelines page in the Google Cloud Console.

Note you may have already deployed an AI Pipelines instance during the Setup for the lab series. If so, you can proceed using that instance. If not:

Create or select an existing Kubernetes cluster (GKE) and deploy AI Platform. Make sure to select `"Allow access to the following Cloud APIs https://www.googleapis.com/auth/cloud-platform"` to allow for programmatic access to your pipeline by the Kubeflow SDK for the rest of the lab. Also, provide an `App instance name` such as "tfx" or "mlops".

Validate the deployment of your AI Platform Pipelines instance in the console before proceeding.

1.2 Exercise: review the example TFX pipeline design pattern for Google Cloud

The pipeline source code can be found in the `pipeline` folder.

[6]:
```
%cd pipeline
```

```
/home/jupyter/training-data-analyst/self-paced-labs/tfx/tfx-ai-
platform/labs/pipeline
```

[7]:
```
!ls -la
```

```
total 80
drwxr-xr-x 5 jupyter jupyter  4096 Oct  5 20:33 .
drwxr-xr-x 4 jupyter jupyter  4096 Oct  5 20:39 ..
drwxr-xr-x 2 jupyter jupyter  4096 Oct  5 20:23 .ipynb_checkpoints
-rw-r--r-- 1 jupyter jupyter    97 Oct  5 19:48 Dockerfile
drwxr-xr-x 2 jupyter jupyter  4096 Oct  5 20:05 __pycache__
-rw-r--r-- 1 jupyter jupyter   300 Oct  5 20:33 build.yaml
-rw-r--r-- 1 jupyter jupyter  1666 Oct  5 19:48 config.py
-rw-r--r-- 1 jupyter jupyter  1222 Oct  5 19:48 features.py
-rw-r--r-- 1 jupyter jupyter 12979 Oct  5 19:48 model.py
-rw-r--r-- 1 jupyter jupyter 11063 Oct  5 19:48 pipeline.py
-rw-r--r-- 1 jupyter jupyter  2032 Oct  5 19:48 preprocessing.py
-rw-r--r-- 1 jupyter jupyter  3778 Oct  5 19:48 runner.py
drwxr-xr-x 2 jupyter jupyter  4096 Oct  5 19:48 schema
-rw-r--r-- 1 jupyter jupyter  4640 Oct  5 20:36
tfx_covertype_continuous_training.tar.gz
```

The `config.py` module configures the default values for the environment specific settings and the default values for the pipeline runtime parameters. The default values can be overwritten at compile time by providing the updated values in a set of environment variables updated in this lab notebook below.

The `pipeline.py` module contains the TFX DSL defining the workflow implemented by the pipeline.

The `preprocessing.py` module implements the data preprocessing logic the `Transform` component.

The `model.py` module implements the training, tuning, and model building logic for the `Trainer` and `Tuner` components.

The `runner.py` module configures and executes `KubeflowDagRunner`. At compile time, the `KubeflowDagRunner.run()` method converts the TFX DSL into the pipeline package in the argo format for execution on your AI Platform Pipelines instance.

The `features.py` module contains feature definitions common across `preprocessing.py` and `model.py`.

1.3 Exercise: build your pipeline package with the TFX CLI

You will use TFX CLI to compile and deploy the pipeline. As explained in the previous section, the environment specific settings can be provided through a set of environment variables and embedded into the pipeline package at compile time.

1.3.1 Configure your environment resource settings

Update the below constants with the settings reflecting your lab environment.

- `GCP_REGION` - the compute region for AI Platform Training, Vizier, and Prediction.
- `ARTIFACT_STORE` - An existing GCS bucket. You can use any bucket or use the GCS bucket created during installation of AI Platform Pipelines. The default bucket name will contain the `kubeflowpipelines-default` prefix.

- `CUSTOM_SERVICE_ACCOUNT` - In the gcp console Click on the Navigation Menu. Navigate to `IAM & Admin`, then to `Service Accounts` and use the service account starting with prefix - `'tfx-tuner-caip-service-account'`. This enables CloudTuner and the Google Cloud AI Platform extensions Tuner component to work together and allows for distributed and parallel tuning backed by AI Platform Vizier's hyperparameter search algorithm.
- `ENDPOINT` - set the `ENDPOINT` constant to the endpoint to your AI Platform Pipelines instance. The endpoint to the AI Platform Pipelines instance can be found on the AI Platform Pipelines page in the Google Cloud Console. Open the *SETTINGS* for your instance and use the value of the `host` variable in the *Connect to this Kubeflow Pipelines instance from a Python client via Kubeflow Pipelines SKD* section of the *SETTINGS* window. The format is `'...pipelines.googleusercontent.com'`.

[8]:
```
PROJECT_ID = !(gcloud config get-value core/project)
PROJECT_ID = PROJECT_ID[0]
GCP_REGION = 'us-central1'
ARTIFACT_STORE_URI = f'gs://{PROJECT_ID}-kubeflowpipelines-default'
CUSTOM_SERVICE_ACCOUNT = f'tfx-tuner-caip-service-account@{PROJECT_ID}.iam.
 ↪gserviceaccount.com'

#TODO: Set your environment resource settings here for ENDPOINT.
ENDPOINT = 'https://679c0f31be458a15-dot-us-central1.pipelines.
 ↪googleusercontent.com'
```

[9]:
```
# Set your resource settings as Python environment variables. These override␣
 ↪the default values in pipeline/config.py.
%env GCP_REGION={GCP_REGION}
%env ARTIFACT_STORE_URI={ARTIFACT_STORE_URI}
%env CUSTOM_SERVICE_ACCOUNT={CUSTOM_SERVICE_ACCOUNT}
%env PROJECT_ID={PROJECT_ID}
```

```
env: GCP_REGION=us-central1
env: ARTIFACT_STORE_URI=gs://qwiklabs-gcp-01-1c499afec2b6-kubeflowpipelines-
default
env: CUSTOM_SERVICE_ACCOUNT=tfx-tuner-caip-service-account@qwiklabs-
gcp-01-1c499afec2b6.iam.gserviceaccount.com
env: PROJECT_ID=qwiklabs-gcp-01-1c499afec2b6
```

1.3.2 Set the pipeline compile time settings

Default pipeline runtime environment values are configured in the pipeline folder `config.py`. You will set their values directly below:

- `PIPELINE_NAME` - the pipeline's globally unique name. For each subsequent pipeline update, each pipeline version uploaded to KFP will be reflected on the `Pipelines` tab in the `Pipeline name > Version name` dropdown in the format `PIPELINE_NAME_datetime.now()`.
- `MODEL_NAME` - the pipeline's unique model output name for AI Platform Prediction. For multiple pipeline runs, each pushed blessed model will create a new version with the format `'v{}'.format(int(time.time()))`.

- `DATA_ROOT_URI` - the URI for the raw lab dataset `gs://workshop-datasets/covertype/small`.
- `CUSTOM_TFX_IMAGE` - the image name of your pipeline container build by skaffold and published by `Cloud Build` to `Cloud Container Registry` in the format `'gcr.io/{}/{}'.format(PROJECT_ID, PIPELINE_NAME)`.
- `RUNTIME_VERSION` - the TensorFlow runtime version. This lab was built and tested using TensorFlow `2.3`.
- `PYTHON_VERSION` - the Python runtime version. This lab was built and tested using Python `3.7`.
- `USE_KFP_SA` - The pipeline can run using a security context of the GKE default node pool's service account or the service account defined in the `user-gcp-sa` secret of the Kubernetes namespace hosting Kubeflow Pipelines. If you want to use the `user-gcp-sa` service account you change the value of `USE_KFP_SA` to `True`. Note that the default AI Platform Pipelines configuration does not define the `user-gcp-sa` secret.
- `ENABLE_TUNING` - boolean value indicating whether to add the `Tuner` component to the pipeline or use hyperparameter defaults. See the `model.py` and `pipeline.py` files for details on how this changes the pipeline topology across pipeline versions.

[10]:
```
PIPELINE_NAME = 'tfx_covertype_continuous_training'
MODEL_NAME = 'tfx_covertype_classifier'
DATA_ROOT_URI = 'gs://workshop-datasets/covertype/small'
CUSTOM_TFX_IMAGE = 'gcr.io/{}/{}'.format(PROJECT_ID, PIPELINE_NAME)
RUNTIME_VERSION = '2.3'
PYTHON_VERSION = '3.7'
USE_KFP_SA=False
ENABLE_TUNING=False
```

[11]:
```
%env PIPELINE_NAME={PIPELINE_NAME}
%env MODEL_NAME={MODEL_NAME}
%env DATA_ROOT_URI={DATA_ROOT_URI}
%env KUBEFLOW_TFX_IMAGE={CUSTOM_TFX_IMAGE}
%env RUNTIME_VERSION={RUNTIME_VERSION}
%env PYTHON_VERIONS={PYTHON_VERSION}
%env USE_KFP_SA={USE_KFP_SA}
%env ENABLE_TUNING={ENABLE_TUNING}
```

```
env: PIPELINE_NAME=tfx_covertype_continuous_training
env: MODEL_NAME=tfx_covertype_classifier
env: DATA_ROOT_URI=gs://workshop-datasets/covertype/small
env: KUBEFLOW_TFX_IMAGE=gcr.io/qwiklabs-
gcp-01-1c499afec2b6/tfx_covertype_continuous_training
env: RUNTIME_VERSION=2.3
env: PYTHON_VERIONS=3.7
env: USE_KFP_SA=False
env: ENABLE_TUNING=False
```

1.3.3 Compile your pipeline code

You can build and upload the pipeline to the AI Platform Pipelines instance in one step, using the `tfx pipeline create` command. The `tfx pipeline create` goes through the following steps: - (Optional) Builds the custom image to that provides a runtime environment for TFX components or uses the latest image of the installed TFX version. - Compiles the pipeline code into a pipeline package. - Uploads the pipeline package via the `ENDPOINT` to the hosted AI Platform instance.

When prototyping with the TFX SDK, you may prefer to first use the `tfx pipeline compile` command, which only executes the compilation step. After the pipeline compiles successfully you can use `tfx pipeline create` to go through all steps.

[12]:
```
!tfx pipeline compile --engine kubeflow --pipeline_path runner.py
```

```
2021-10-05 20:41:41.317755: W
tensorflow/stream_executor/platform/default/dso_loader.cc:59] Could not load
dynamic library 'libcudart.so.10.1'; dlerror: libcudart.so.10.1: cannot open
shared object file: No such file or directory
2021-10-05 20:41:41.317888: I tensorflow/stream_executor/cuda/cudart_stub.cc:29]
Ignore above cudart dlerror if you do not have a GPU set up on your machine.
CLI
Compiling pipeline
2021-10-05 20:41:44.886759: W
tensorflow/stream_executor/platform/default/dso_loader.cc:59] Could not load
dynamic library 'libcudart.so.10.1'; dlerror: libcudart.so.10.1: cannot open
shared object file: No such file or directory
2021-10-05 20:41:44.886906: I tensorflow/stream_executor/cuda/cudart_stub.cc:29]
Ignore above cudart dlerror if you do not have a GPU set up on your machine.
WARNING:absl:RuntimeParameter is only supported on Cloud-based DAG runner
currently.
WARNING:absl:RuntimeParameter is only supported on Cloud-based DAG runner
currently.
WARNING:absl:RuntimeParameter is only supported on Cloud-based DAG runner
currently.
WARNING:absl:RuntimeParameter is only supported on Cloud-based DAG runner
currently.
WARNING:absl:`instance_name` is deprecated, please set node id directly
using`with_id()` or `.id` setter.
WARNING:absl:`instance_name` is deprecated, please set node id directly
using`with_id()` or `.id` setter.
Pipeline compiled successfully.
Pipeline package path: /home/jupyter/training-data-analyst/self-paced-
labs/tfx/tfx-ai-platform/labs/pipeline/tfx_covertype_continuous_training.tar.gz
```

Note: you should see a `{PIPELINE_NAME}.tar.gz` file appear in your `/pipeline` directory.

1.4 Exercise: deploy your pipeline container to AI Platform Pipelines with TFX CLI

After the pipeline code compiles without any errors you can use the `tfx pipeline create` command to perform the full build and deploy the pipeline. You will deploy your compiled pipeline container hosted on Google Container Registry e.g. `gcr.io/[PROJECT_ID]/[PIPELINE_NAME]` to run on AI Platform Pipelines with the TFX CLI. To learn more about the command below, you can review the TFX CLI documentation.

[13]:
```
!tfx pipeline create  \
--pipeline_path=runner.py \
--endpoint={ENDPOINT} \
--build_target_image={CUSTOM_TFX_IMAGE}
```

```
2021-10-05 20:41:56.570747: W
tensorflow/stream_executor/platform/default/dso_loader.cc:59] Could not load
dynamic library 'libcudart.so.10.1'; dlerror: libcudart.so.10.1: cannot open
shared object file: No such file or directory
2021-10-05 20:41:56.570891: I tensorflow/stream_executor/cuda/cudart_stub.cc:29]
Ignore above cudart dlerror if you do not have a GPU set up on your machine.
CLI
Creating pipeline
Detected Kubeflow.
Use --engine flag if you intend to use a different orchestrator.
Reading build spec from build.yaml
Target image gcr.io/qwiklabs-
gcp-01-1c499afec2b6/tfx_covertype_continuous_training is not used. If the build
spec is provided, update the target image in the build spec file build.yaml.
[Skaffold] Generating tags…
[Skaffold]  - gcr.io/qwiklabs-
gcp-01-1c499afec2b6/tfx_covertype_continuous_training -> gcr.io/qwiklabs-
gcp-01-1c499afec2b6/tfx_covertype_continuous_training:latest
[Skaffold] Checking cache…
[Skaffold]  - gcr.io/qwiklabs-
gcp-01-1c499afec2b6/tfx_covertype_continuous_training: Not found. Building
[Skaffold] Starting build…
[Skaffold] Building [gcr.io/qwiklabs-
gcp-01-1c499afec2b6/tfx_covertype_continuous_training]…
[Skaffold] Sending build context to Docker daemon  60.93kB
[Skaffold] Step 1/4 : FROM tensorflow/tfx:0.25.0
[Skaffold]  ---> 05d9b228cf63
[Skaffold] Step 2/4 : WORKDIR ./pipeline
[Skaffold]  ---> Using cache
[Skaffold]  ---> c30ad679fe36
[Skaffold] Step 3/4 : COPY ./ ./
[Skaffold]  ---> 45f0c26c7efb
[Skaffold] Step 4/4 : ENV PYTHONPATH="/pipeline:${PYTHONPATH}"
[Skaffold]  ---> Running in bc43bdac99c0
```

```
[Skaffold] Removing intermediate container bc43bdac99c0
[Skaffold]  ---> c2719964f0ad
[Skaffold] Successfully built c2719964f0ad
[Skaffold] Successfully tagged gcr.io/qwiklabs-
gcp-01-1c499afec2b6/tfx_covertype_continuous_training:latest
[Skaffold] The push refers to repository [gcr.io/qwiklabs-
gcp-01-1c499afec2b6/tfx_covertype_continuous_training]
[Skaffold] d7b05e7db468: Preparing
[Skaffold] 0e82e24a5fb7: Preparing
[Skaffold] 5dadc0a09248: Preparing
[Skaffold] 8fb12d3bda49: Preparing
[Skaffold] 2471eac28ba8: Preparing
[Skaffold] 674ba689ae71: Preparing
[Skaffold] 4058ae03fa32: Preparing
[Skaffold] e3437c61d457: Preparing
[Skaffold] 84ff92691f90: Preparing
[Skaffold] 54b00d861a7a: Preparing
[Skaffold] c547358928ab: Preparing
[Skaffold] 84ff92691f90: Preparing
[Skaffold] c4e66be694ce: Preparing
[Skaffold] 47cc65c6dd57: Preparing
[Skaffold] 674ba689ae71: Waiting
[Skaffold] 4058ae03fa32: Waiting
[Skaffold] e3437c61d457: Waiting
[Skaffold] 84ff92691f90: Waiting
[Skaffold] 54b00d861a7a: Waiting
[Skaffold] c547358928ab: Waiting
[Skaffold] c4e66be694ce: Waiting
[Skaffold] 47cc65c6dd57: Waiting
[Skaffold] 0e82e24a5fb7: Layer already exists
[Skaffold] 2471eac28ba8: Layer already exists
[Skaffold] 8fb12d3bda49: Layer already exists
[Skaffold] 5dadc0a09248: Layer already exists
[Skaffold] 674ba689ae71: Layer already exists
[Skaffold] e3437c61d457: Layer already exists
[Skaffold] 4058ae03fa32: Layer already exists
[Skaffold] 84ff92691f90: Layer already exists
[Skaffold] 54b00d861a7a: Layer already exists
[Skaffold] c547358928ab: Layer already exists
[Skaffold] c4e66be694ce: Layer already exists
[Skaffold] 47cc65c6dd57: Layer already exists
[Skaffold] d7b05e7db468: Pushed
[Skaffold] latest: digest:
sha256:cb0ea4dab945a4eaba264c892eb861fa2d80bea876b676948126cd2e0b8e70be size:
3267
[Skaffold]
New container image is built. Target image is available in the build spec file.
2021-10-05 20:42:15.158556: W
```

```
tensorflow/stream_executor/platform/default/dso_loader.cc:59] Could not load
dynamic library 'libcudart.so.10.1'; dlerror: libcudart.so.10.1: cannot open
shared object file: No such file or directory
2021-10-05 20:42:15.158678: I tensorflow/stream_executor/cuda/cudart_stub.cc:29]
Ignore above cudart dlerror if you do not have a GPU set up on your machine.
WARNING:absl:RuntimeParameter is only supported on Cloud-based DAG runner
currently.
WARNING:absl:RuntimeParameter is only supported on Cloud-based DAG runner
currently.
WARNING:absl:RuntimeParameter is only supported on Cloud-based DAG runner
currently.
WARNING:absl:RuntimeParameter is only supported on Cloud-based DAG runner
currently.
WARNING:absl:`instance_name` is deprecated, please set node id directly
using`with_id()` or `.id` setter.
WARNING:absl:`instance_name` is deprecated, please set node id directly
using`with_id()` or `.id` setter.
Pipeline compiled successfully.
Pipeline package path: /home/jupyter/training-data-analyst/self-paced-
labs/tfx/tfx-ai-platform/labs/pipeline/tfx_covertype_continuous_training.tar.gz
Pipeline "tfx_covertype_continuous_training" already exists.
```

If you make a mistake above and need to redeploy the pipeline you can first delete the previous version using `tfx pipeline delete` or you can update the pipeline in-place using `tfx pipeline update`.

To delete the pipeline:

```
tfx pipeline delete --pipeline_name {PIPELINE_NAME} --endpoint {ENDPOINT}
```

To update the pipeline:

```
tfx pipeline update --pipeline_path runner.py --endpoint {ENDPOINT}
```

1.5 Exercise: create a pipeline run with the TFX CLI

After the pipeline has been deployed, you can trigger and monitor pipeline runs using TFX CLI. For more information on the step below, review the TFX CLI documentation on the "run group".

[14]:
```
!tfx run create --pipeline_name={PIPELINE_NAME} --endpoint={ENDPOINT}
```

```
2021-10-05 20:42:39.121109: W
tensorflow/stream_executor/platform/default/dso_loader.cc:59] Could not load
dynamic library 'libcudart.so.10.1'; dlerror: libcudart.so.10.1: cannot open
shared object file: No such file or directory
2021-10-05 20:42:39.121244: I tensorflow/stream_executor/cuda/cudart_stub.cc:29]
Ignore above cudart dlerror if you do not have a GPU set up on your machine.
CLI
Creating a run for pipeline: tfx_covertype_continuous_training
Detected Kubeflow.
Use --engine flag if you intend to use a different orchestrator.
```

```
Run created for pipeline: tfx_covertype_continuous_training
+----------------------------------+--------------------------------------+----
------+--------------------------+-------------------------------------------
-------------------------------------------------------------------------------
-+
| pipeline_name                    | run_id                               |
status   | created_at               | link
|
+==================================+======================================+====
======+==========================+===========================================
===============================================================================
=+
| tfx_covertype_continuous_training | 7e22638f-4410-4954-a0e9-2f998033c13a |
| 2021-10-05T20:42:43+00:00 | https://679c0f31be458a15-dot-us-central1.pipelines
.googleusercontent.com/#/runs/details/7e22638f-4410-4954-a0e9-2f998033c13a |
+----------------------------------+--------------------------------------+----
------+--------------------------+-------------------------------------------
-------------------------------------------------------------------------------
-+
```

1.6 Exercise: monitor your pipeline runs with the TFX CLI

To view the status of existing pipeline runs:

[15]:
```
!tfx run list --pipeline_name {PIPELINE_NAME} --endpoint {ENDPOINT}
```

```
2021-10-05 20:43:02.884194: W
tensorflow/stream_executor/platform/default/dso_loader.cc:59] Could not load
dynamic library 'libcudart.so.10.1'; dlerror: libcudart.so.10.1: cannot open
shared object file: No such file or directory
2021-10-05 20:43:02.884349: I tensorflow/stream_executor/cuda/cudart_stub.cc:29]
Ignore above cudart dlerror if you do not have a GPU set up on your machine.
CLI
Listing all runs of pipeline: tfx_covertype_continuous_training
Detected Kubeflow.
Use --engine flag if you intend to use a different orchestrator.
+----------------------------------+--------------------------------------+----
------+--------------------------+-------------------------------------------
-------------------------------------------------------------------------------
-+
| pipeline_name                    | run_id                               |
status   | created_at               | link
|
+==================================+======================================+====
======+==========================+===========================================
===============================================================================
=+
| tfx_covertype_continuous_training | d3f91144-0960-4a37-8980-a1fe7282b0a6 |
```

```
Running  | 2021-10-05T20:36:17+00:00 | https://679c0f31be458a15-dot-us-central1.
pipelines.googleusercontent.com/#/runs/details/d3f91144-0960-4a37-8980-a1fe7282b
0a6 |
+----------------------------------+--------------------------------------+----
------+--------------------------+-------------------------------------------
-----------------------------------------------------------------------------
-+
| tfx_covertype_continuous_training | 7e22638f-4410-4954-a0e9-2f998033c13a |
Running  | 2021-10-05T20:42:43+00:00 | https://679c0f31be458a15-dot-us-central1.
pipelines.googleusercontent.com/#/runs/details/7e22638f-4410-4954-a0e9-2f998033c
13a |
+----------------------------------+--------------------------------------+----
------+--------------------------+-------------------------------------------
-----------------------------------------------------------------------------
-+
```

To retrieve the status of a given run retrieved from the command above:

[18]:
```
RUN_ID='7e22638f-4410-4954-a0e9-2f998033c13a'

!tfx run status --pipeline_name {PIPELINE_NAME} --run_id {RUN_ID} --endpoint␣
 ↪{ENDPOINT}
```

```
2021-10-05 20:44:34.716228: W
tensorflow/stream_executor/platform/default/dso_loader.cc:59] Could not load
dynamic library 'libcudart.so.10.1'; dlerror: libcudart.so.10.1: cannot open
shared object file: No such file or directory
2021-10-05 20:44:34.716365: I tensorflow/stream_executor/cuda/cudart_stub.cc:29]
Ignore above cudart dlerror if you do not have a GPU set up on your machine.
CLI
Retrieving run status.
Detected Kubeflow.
Use --engine flag if you intend to use a different orchestrator.
+----------------------------------+--------------------------------------+----
------+--------------------------+-------------------------------------------
-----------------------------------------------------------------------------
-+
| pipeline_name                    | run_id                               |
status   | created_at               | link
|
+==================================+======================================+====
======+==========================+===========================================
=============================================================================
=+
| tfx_covertype_continuous_training | 7e22638f-4410-4954-a0e9-2f998033c13a |
Running  | 2021-10-05T20:42:43+00:00 | https://679c0f31be458a15-dot-us-central1.
pipelines.googleusercontent.com/#/runs/details/7e22638f-4410-4954-a0e9-2f998033c
13a |
```

```
+----------------------------------+--------------------------------------+----
------+--------------------------+-------------------------------------------
-----------------------------------------------------------------------------
-+
```

1.7 Exercise: monitor your pipeline runs with the Kubeflow Pipelines UI

On the AI Platform Pipelines page, click `OPEN PIPELINES DASHBOARD`. A new browser tab will open. Select the `Pipelines` tab to the left where you see the `PIPELINE_NAME` pipeline you deployed previously.

Click on the most recent pipeline version which will open up a window with a visualization of your TFX pipeline directed graph. Pipeline components are represented as named boxes with direct arrows representing artifact dependencies and the execution order of your ML workflow.

Next, click the `Experiments` tab. You will see your pipeline name under `Experiment name` with an downward arrow that allows you to view all active and previous runs. Click on the pipeline run that you trigger with the step above. You can follow your pipeline's run progress by viewing your pipeline graph get built on the screen and drill into individual components to view artifacts, ML Metadata, and logs.

Click on the `Trainer` component in the KFP UI once it is running and navigate to the `Visualizations` tab. Scroll down to the Tensorboard widget and hit the `Open Tensorboard` button to monitor your training performance. Individual model performance will vary run-to-run but your model will achieve about 70-72% accuracy. On the AI Platform Jobs page, you can also further inspect the Training job logs and graphs on training resource utilization. Tracking your model's performance and resource footprint across runs gives you a systemic way to measure and prioritize actions to improve your model's performance such as hyperparameter tuning, adding additional data, increasing model complexity, or focusing on additional feature engineering.

1.7.1 Important

A full pipeline run with tuning enabled will take about 50 minutes to complete. You can view the run's progress using the TFX CLI commands above and in the KFP UI. While the pipeline run is in progress, there are also optional exercises below to explore your pipeline's artifacts and Google Cloud integrations while the pipeline run is in progress.

1.7.2 Exercise (optional): review the pipeline code to explore hyperparameter tuning workflow

Incorporating automatic model hyperparameter tuning into a continuous training TFX pipeline workflow enables faster experimentation, development, and deployment of a top performing model. However, you might not want to tune the hyperparameters every time you retrain your model due to the computational cost, amount of time to tune, and diminishing performance returns over time. Typically you would want to tune for a large number of trials over a wide search space which is beyond the duration of this lab.

The pipeline run you triggered above did not include hyperparameter tuning based on the `ENABLE_TUNING=False` environment variable set above and the model instead used default hyperparameters. Default hyperparameter values in the search space are defined in

`_get_hyperparameters()` in the pipeline's `model.py` and these values are used to build a TensorFlow WideDeep Classifier model.

Review the pipeline design pattern for conditional model tuning in `pipeline.py`. When `ENABLE_TUNING=True`, the pipeline typology changes to include the `Tuner` component that calls out to the AI Platform Vizier service for hyperparameter tuning. The `Tuner` component `"best_hyperparameters"` artifact will be passed directly to your `Trainer` component to deploy the top performing model. Also, review the tuning function in `model.py` for configuring `CloudTuner`.

Once you have used `Tuner` determine a good set of hyperparameters, you can remove `Tuner` from your pipeline and use model hyperparameters defined in your model code or use a `ImporterNode` to import the `Tuner "best_hyperparameters"`artifact from a previous `Tuner` run to your model `Trainer`.

1.7.3 Exercise (optional): review your pipeline's Dataflow jobs for data processing

On the Dataflow page, click on the most recent job and inspect the computation graph for parallelized data processing. It will include details about your job's status, type, SDK version, any errors or warnings, and additional diagnostic graphs. As your pipeline run progresses, you will see jobs kick off for the ExampleGen, StatisticsGen, Transform, and Evaluator pipeline components.

Take a look at the job monitoring charts that display metrics over the duration of the pipeline job. They provide I/O metrics to identify bottlenecks, statistical information to surface anomalies, and step-level visibility for debugging pipeline lag or errors.

1.7.4 Exercise (optional): review your pipeline's artifacts on Cloud Storage

On the Cloud Storage page, review how TFX standardizes the organization of your pipeline run artifacts. You will find them organized by component and versioned in your `gs://{PROJECT_ID}-kubeflowpipelines-default` artifact storage bucket. This standardization brings reproducibility and traceability to your ML workflows and allows for easier reuse of pipeline components and artifacts across use cases.

1.8 License