AI & Machine Learning Products (https://cloud.google.com/products/machine-learning/)
Cloud TPU (https://cloud.google.com/tpu/) Guides

Training RetinaNet on Cloud TPU

This document describes an implementation of the RetinaNet object detection model. The code is available on GitHub (https://github.com/tensorflow/tpu/tree/master/models/official/detection).

The instructions below assume you are already familiar with running a model on Cloud TPU. If you are new to Cloud TPU, you can refer to the Quickstart (https://cloud.google.com/tpu/docs/quickstart) for a basic introduction.

If you plan to train on a TPU Pod slice, review Training on TPU Pods (https://cloud.google.com/tpu/docs/training-on-tpu-pods) to understand parameter changes required for Pod slices.

Warning: This tutorial uses a third-party dataset. Google provides no representation, warranty, or other guarantees about the validity, or any other aspects of, this dataset.

Objectives

- Create a Cloud Storage bucket to hold your dataset and model output
- Prepare the COCO dataset
- Set up a Compute Engine VM and Cloud TPU node for training and evaluation
- Run training and evaluation on a single Cloud TPU or a Cloud TPU Pod

Costs

This tutorial uses billable components of Google Cloud, including:

- Compute Engine
- Cloud TPU
- Cloud Storage

Use the pricing calculator (https://cloud.google.com/products/calculator/) to generate a cost estimate based on your projected usage. New Google Cloud users might be eligible for a free trial (https://cloud.google.com/free/).

Before you begin

This section provides information on setting up Cloud Storage storage and a Compute Engine VM.

Important: Set up your Compute Engine VM, your Cloud TPU node and your Cloud Storage bucket in the same region/zone to reduce network latency and network costs.

1. Open a Cloud Shell window.

   OPEN CLOUD SHELL (HTTPS://CONSOLE.CLOUD.GOOGLE.COM/?CLOUDSHELL=TRUE)

2. Create a variable for your project's name.

   ```
   export PROJECT_NAME=your-project_name
   ```

3. Configure `gcloud` command-line tool to use the project where you want to create Cloud TPU.

   ```
   gcloud config set project ${PROJECT_NAME}
   ```

4. Create a Cloud Storage bucket using the following command:

   Note: In the following command, replace ***your-bucket-name*** with the name you want to assign to your bucket.

   ```
   gsutil mb -p ${PROJECT_NAME} -c standard -l europe-west4 -b on gs://your-bucket
   ```

   This Cloud Storage bucket stores the data you use to train your model and the training results.

5. Launch a Compute Engine VM using the `ctpu up` command.

   ```
   $ ctpu up  --vm-only --disk-size-gb=300 --machine-type=n1-standard-8 --zone=eur
   ```

Note: If you have more than one project, you must specify the project name. If `--name` is not specified, it defaults to your username.

6. The configuration you specified appears. Enter `y to approve or n to cancel.`
7. When the `ctpu up` command has finished executing, verify that your shell prompt has changed from `username@project` to `username@tpuname`. This change shows that you are now logged into your Compute Engine VM.

   If you are not logged into the Compute Engine VM, you can do so by running the following command:

```
gcloud compute ssh username --zone=europe-west4-a
```

Note: The first time you run `ctpu up` on a project it takes several minutes to perform startup tasks such as SSH key propagation and API turnup. Occasionally, during the SSH key propagation, an error message is returned. If that happens, rerun the `ctpu up` command to see if that clears the problem.

As you continue these instructions, run each command that begins with `(vm)$` in your VM session window.

When the `ctpu` command launches a Compute Engine virtual machine (VM), automatically places the RetinaNet model files from TensorFlow branch (https://github.com/tensorflow/tpu/tree/r1.14/models/official/detection) in the following location:

```
(vm)$ ls /usr/share/tpu/models/official/detection/
```

Prepare the COCO dataset

1. Run the `download_and_preprocess_coco.sh` script to convert the COCO dataset into a set of TFRecords (`*.tfrecord`) that the training application expects.

```
(vm)$ bash /usr/share/tpu/tools/datasets/download_and_preprocess_coco.sh ./data
```

   This installs the required libraries and then runs the preprocessing script. It outputs a number of `*.tfrecord` files in your local data directory. The COCO download and

conversion script takes approximately 1 hour to complete.

2. After you convert the data into TFRecords, copy them from local storage to your Cloud Storage bucket using the `gsutil` command. You must also copy the annotation files. These files help validate the model's performance:

```
(vm)$ gsutil -m cp ./data/dir/coco/*.tfrecord ${STORAGE_BUCKET}/coco
(vm)$ gsutil cp ./data/dir/coco/raw-data/annotations/*.json ${STORAGE_BUCKET}/c
```

Set up the training environment

1. Run the following command to create your Cloud TPU.

```
(vm)$ ctpu up --tpu-only --tpu-size=v3-8 --zone=europe-west4-a
```

Parameter	Description
tpu-size	Specifies the type of Cloud TPU. Specify a type that is available in the zone you are using to create your Cloud TPU.
zone	The zone where you plan to create your Cloud TPU. This should be the same zone you used for the Compute Engine VM. For example, **europe-west4-a**.

2. The configuration you specified appears. Enter `y to approve or n to cancel.`

   You will see a message: **`Operation success; not ssh-ing to Compute Engine VM due to --tpu-only flag`**. Since you previously completed SSH key propagation, you can ignore this message.

3. Install extra packages

   The RetinaNet training application requires several extra packages. Install them now:

```
(vm)$ sudo apt-get install -y python-tk && \
      pip install --user Cython matplotlib opencv-python-headless pyyaml Pillow
      pip install --user 'git+https://github.com/cocodataset/cocoapi#egg=pycoco
      pip install --user -U gast==0.2.2
```

4. Update the keepalive values for your VM connection

This tutorial requires a long-lived connection to the Compute Engine instance. To ensure you aren't disconnected from the instance, run the following command:

```
(vm)$ sudo /sbin/sysctl \
      -w net.ipv4.tcp_keepalive_time=120 \
      net.ipv4.tcp_keepalive_intvl=120 \
      net.ipv4.tcp_keepalive_probes=5
```

5. Define parameter values

   Next, you need to define several parameter values. You use these parameters to train and evaluate your model.

   The variables you need to set are described in the following table:

Parameter	Description
STORAGE_BUCKET	This is the name of the Cloud Storage bucket that you created in the **Before you begin** section.
TPU_NAME	This is the name of the Compute Engine VM and the the Cloud TPU. The Compute Engine VM and the Cloud TPU name must be the same. Since the Compute Engine VM was set to the default value, your username, set the Cloud TPU to the same value.

6. Use the `export` command to set these variables.

★ Note: In the following code example, replace ***your-bucket-name*** with the name of your bucket and set the TPU_NAME variable to your username.

```
(vm)$ export STORAGE_BUCKET=gs://your-bucket-name
(vm)$ export TPU_NAME=username
```

7. You are now ready to run the model on the preprocessed COCO data. First, add the top-level `/models` folder to the Python path with the command:

```
(vm)$ export PYTHONPATH="$PYTHONPATH:/usr/share/tpu/models"
```

Training and evaluation require TensorFlow 1.13 or a later version.

Note: If you want to monitor the model's output and performance, follow the guide to [setting up TensorBoard] [tensorboard-setup].

Single Cloud TPU device training

The following training scripts were run on a Cloud TPU v3-8. It will take more time, but you can also run them on a Cloud TPU v2-8.

This script trains for 22,500 steps and takes approximately 3 hours to train on a Cloud TPU v2-8 and approximately 1 1/2 hours to train on a Cloud TPU v3-8.

1. Set up the following environment variables:

```
(vm)$ TPU_NAME=$TPU_NAME  \
  export MODEL_DIR=${STORAGE_BUCKET}/retinanet-model-train; \
  export RESNET_CHECKPOINT=gs://cloud-tpu-artifacts/resnet/resnet-nhwc-2018-10-
  export TRAIN_FILE_PATTERN=${STORAGE_BUCKET}/coco/train-*; \
  export EVAL_FILE_PATTERN=${STORAGE_BUCKET}/coco/val-*; \
  export VAL_JSON_FILE=${STORAGE_BUCKET}/coco/instances_val2017.json
```

2. Run the training script:

```
(vm)$ python /usr/share/tpu/models/official/detection/main.py \
     --use_tpu=True \
     --tpu=${TPU_NAME:?} \
     --num_cores=8 \
     --model_dir="${MODEL_DIR?}" \
     --mode="train" \
     --eval_after_training=True \
     --params_override="{ type: retinanet, train: { checkpoint: { path: ${RESNE
```

Parameter	Description
tpu	Specifies the name of the Cloud TPU. This is set by specifying the environment variable (**TPU_NAME**).
model_dir	Specifies the directory where checkpoints and summaries are stored during model training. If the folder is missing, the program creates one. When using a Cloud TPU, the **model_dir** must be a Cloud Storage path (`gs://...`). You can reuse an existing folder to load current checkpoint data and to store additional

	checkpoints as long as the previous checkpoints were created using TPU of the same size and TensorFlow version.
RESNET_CHECKPOINT	Specifies a pretrained checkpoint. RetinaNet requires a pre-trained image classification model (like ResNet) as a backbone network. This example uses a pretrained checkpoint created with the ResNet demonstration model. You can instead train your own ResNet model if desired, and specify a checkpoint from your ResNet model directory.

Single Cloud TPU device evaluation

The following procedure uses the COCO evaluation data. It takes about 10 minutes to run through the evaluation steps.

1. Set up the following environment variables:

```
(vm)$ TPU_NAME=$TPU_NAME  \
  export MODEL_DIR=${STORAGE_BUCKET}/retinanet-model-train; \
  export EVAL_FILE_PATTERN=${STORAGE_BUCKET}/coco/val-*; \
  export VAL_JSON_FILE=${STORAGE_BUCKET}/coco/instances_val2017.json \
  export EVAL_SAMPLES=5000
```

2. Run the evaluation script:

```
(vm)$ python /usr/share/tpu/models/official/detection/main.py \
      --use_tpu=True \
      --tpu=${TPU_NAME:?} \
      --num_cores=8 \
      --model_dir="${MODEL_DIR?}" \
      --mode="eval" \
      --params_override="{ type: retinanet, eval: { val_json_file: ${VAL_JSON_F
```

Parameter	Description
tpu	Specifies the name of the Cloud TPU. This is set by specifying the environment variable (**TPU_NAME**).
model_dir	Specifies the directory where checkpoints and summaries are stored during model training. If the folder is missing, the program creates one. When using a Cloud TPU, the **model_dir** must be a Cloud Storage path (`gs://...`). You can reuse an existing folder to load current checkpoint data and to store additional checkpoints as long as the previous checkpoints were created using TPU of the same size and TensorFlow version.

At this point, you can either conclude this tutorial and clean up (#clean-up) your GCP resources, or you can further explore running the model on Cloud TPU Pods.

Scaling your model with Cloud TPU Pods

You can get results faster by scaling your model with Cloud TPU Pods. The fully supported RetinaNet model can work with the following Pod slices:

- v2-32
- v3-32

Caution: Scaling to larger Pod slices are experimental with this model. The training scripts in this tutorial were optimized for a v3-32 TPU node. They will not run as-is on a v2 configuration because they require additional memory.

The script trains for 2109 steps. It takes approximately 30 minutes to train on a v3-32 TPU type and 10 minutes to train on a v3-128 TPU type.

1. Delete the Cloud TPU resource you created for training the model on a single device.

```
(vm)$ ctpu delete --tpu-only --zone=europe-west4-a
```

2. Run the `ctpu up` command, using the `tpu-size` parameter to specify the Pod slice you want to use. For example, the following command uses a v3-32 Pod slice.

```
(vm)$ ctpu up --tpu-only --tpu-size=v3-32 --zone=europe-west4-a
```

3. The configuration you specified appears. Enter `y to approve or n to cancel.`

   You will see a message: **`Operation success; not ssh-ing to Compute Engine VM due to --tpu-only flag`**. Since you previously completed SSH key propagation, you can ignore this message.

4. Install the extra packages needed by RetinaNet.

```
(vm)$ sudo apt-get install -y python-tk && \
      pip install --user Cython matplotlib opencv-python-headless pyyaml Pillow
      pip install --user 'git+https://github.com/cocodataset/cocoapi#egg=pycoco
      pip install --user -U gast==0.2.2
```

5. Update the keepalive values of your VM connection

   This tutorial requires a long-lived connection to the Compute Engine instance. To ensure you aren't disconnected from the instance, run the following command:

```
(vm)$ sudo /sbin/sysctl \
   -w net.ipv4.tcp_keepalive_time=120 \
   net.ipv4.tcp_keepalive_intvl=120 \
   net.ipv4.tcp_keepalive_probes=5
```

6. Define the variables you need for training on a Pod. Use the `export` command to create multiple bash variables and use them in a configuration string.

   Note: In the following code example, replace ***your-bucket-name*** with the name of your bucket and set the TPU_NAME variable to your username.

```
(vm)$ export STORAGE_BUCKET=gs://your-bucket-name
(vm)$ export TPU_NAME=username
```

7. Add the top-level `/models` folder to the Python path.

```
(vm)$ export PYTHONPATH="$PYTHONPATH:/usr/share/tpu/models"
```

8. Set up the following environment variables:

```
(vm)$ TPU_NAME=$TPU_NAME  \
  export MODEL_DIR=${STORAGE_BUCKET}/retinanet-model-pod; \
  export RESNET_CHECKPOINT=gs://cloud-tpu-artifacts/resnet/resnet-nhwc-2018-10-
  export TRAIN_FILE_PATTERN=${STORAGE_BUCKET}/coco/train-*;
```

9. Run the Pod training script on a v3-32 TPU node:

```
(vm)$ python /usr/share/tpu/models/official/detection/main.py \
      --use_tpu=True \
      --tpu=${TPU_NAME:?} \
      --num_cores=32 \
      --model_dir="${MODEL_DIR?}" \
      --mode="train" \
      --eval_after_training=False \
      --params_override="{ type: retinanet, train: { train_batch_size: 1024, to
```

Parameter	Description
tpu	Specifies the name of the Cloud TPU. This is set by specifying the environment variable (**TPU_NAME**).
model_dir	Specifies the directory where checkpoints and summaries are stored during model training. If the folder is missing, the program creates one. When using a Cloud TPU, the **model_dir** must be a Cloud Storage path (`gs://...`). You can reuse an existing folder to load current checkpoint data and to store additional checkpoints as long as the previous checkpoints were created using TPU of the same size and TensorFlow version.
RESNET_CHECKPOINT	Specifies a pretrained checkpoint. RetinaNet requires a pre-trained image classification model (like ResNet) as a backbone network. This example uses a pretrained checkpoint created with the ResNet demonstration model. You can instead train your own ResNet model if desired, and specify a checkpoint from your ResNet model directory.

Cleaning up

To avoid incurring charges to your Google Cloud Platform account for the resources used in this tutorial:

1. Disconnect from the Compute Engine instance, if you have not already done so:

```
(vm)$ exit
```

   Your prompt should now be `user@projectname`, showing you are in the Cloud Shell.

2. In your Cloud Shell, run `ctpu delete` with the –zone flag you used when you set up the Cloud TPU to delete your Compute Engine VM and your Cloud TPU:

```
$ ctpu delete --zone=europe-west4-a
```

Important: If you set the TPU resources name when you ran `ctpu up`, you must specify that name with the `--name` flag when you run `ctpu delete` in order to shut down your TPU resources.

3. Run the following command to verify the Compute Engine VM and Cloud TPU have been shut down:

```
$ ctpu status --zone=europe-west4-a
```

The deletion might take several minutes. A response like the one below indicates there are no more allocated instances:

```
2018/04/28 16:16:23 WARNING: Setting zone to "europe-west4-a"
No instances currently exist.
        Compute Engine VM:     --
        Cloud TPU:             --
```

Caution: All training data will be lost when you delete your bucket, so only do this step when you are finished running the tutorial.

4. Run `gsutil` as shown, replacing ***your-bucket-name*** with the name of the Cloud Storage bucket you created for this tutorial:

```
$ gsutil rm -r gs://your-bucket-name
```

What's next

Train with different image sizes

You can explore using a larger backbone network (for example, ResNet-101 instead of ResNet-50). A larger input image and a more powerful backbone will yield a slower but more precise model.

Use a different basis

Alternatively, you can explore pre-training a ResNet model on your own dataset and using it as a basis for your RetinaNet model. With some more work, you can also swap in an alternative *backbone* network in place of ResNet. Finally, if you are interested in implementing your own object detection models, this network may be a good basis for further experimentation.

Except as otherwise noted, the content of this page is licensed under the Creative Commons Attribution 4.0 License (https://creativecommons.org/licenses/by/4.0/), and code samples are licensed under the Apache 2.0 License (https://www.apache.org/licenses/LICENSE-2.0). For details, see our Site Policies (https://developers.google.com/terms/site-policies). Java is a registered trademark of Oracle and/or its affiliates.

Last updated December 2, 2019.