Training EfficientNet on Cloud TPU

> **Objective:** Train the Tensorflow EfficientNet model using a Cloud TPU device or Cloud TPU Pod slice (multiple TPU devices).

The EfficientNet models are a family of image classification models, which achieve state-of-the-art accuracy, while also being smaller and faster than other models. EfficientNet-EdgeTpu are models customized to run efficiently on the Google EdgeTPU devices.

The model in this tutorial is based on EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. Researchers developed a new technique to improve model performance: carefully balancing network depth, width, and resolution, using a simple yet highly effective compound coefficient.

The family of models from `efficientnet-b0` to `efficientnet-b7`, can achieve decent image classification accuracy given the resource constrained Google EdgeTPU devices.

`efficientnet-b0`, the model used in this tutorial, corresponds to the smallest base model, whereas `efficientnet-b7` corresponds to the most power but computation-expensive model. The tutorial demonstrates training the model using TPUEstimator.

> **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 a test version of the ImageNet dataset, referred to as the *fake_imagenet* dataset.
- Run the training job.
- Verify the output results.

Costs

This tutorial uses billable components of Google Cloud, including:

- Compute Engine
- Cloud TPU
- Cloud Storage

Use the pricing calculator to generate a cost estimate based on your projected usage.

Before you begin

Before starting this tutorial, check that your Google Cloud project is correctly set up.

1. In the Cloud Console, on the project selector page, select or create a Cloud project.

   > **Note**: If you don't plan to keep the resources that you create in this procedure, create a project instead of selecting an existing project. After you finish these steps, you can delete the project, removing all resources associated with the project.

   Go to the project selector page

2. Make sure that billing is enabled for your Google Cloud project. Learn how to confirm billing is enabled for your project.

   This walkthrough uses billable components of Google Cloud. Check the Cloud TPU pricing page to estimate your costs. Be sure to clean up resources you create when you've finished with them to avoid unnecessary charges.

Set up your resources

This section provides information on setting up Cloud Storage, VM, and Cloud TPU resources for tutorials.

> **Important:** Set up all resources in the same region/zone to reduce network latency and network costs.

1. Open a Cloud Shell window.

   Open Cloud Shell

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

```
export PROJECT_ID=project-id
```

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

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

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

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

   This Cloud Storage bucket stores the data you use to train your model and the training results. The `ctpu up` tool used in this tutorial sets up default permissions for the Cloud TPU service account.

If you want finer-grain permissions, review the access level permissions.

The bucket location must be in the same region as your virtual machine (VM) and your TPU node. VMs and TPU nodes are located in specific zones, which are subdivisions within a region.

5. Launch the Compute Engine and Cloud TPU resources required for this using the `ctpu up`command.

```
ctpu up --zone=europe-west4-a \
  --vm-only \
  --disk-size-gb=300 \
  --machine-type=n1-standard-8 \
  --tf-version=1.15.3 \
  --name=efficientnet-tutorial
```

> **Note:** If you have more than one project, you must specify the project name.

For more information on the CTPU utility, see CTPU Reference.

6. When prompted, press `y` to create your Cloud TPU resources.

> **Note:** The first time you run **`ctpu up`** on a project it takes about 5 minutes to perform startup tasks such as SSH key propagation and API turnup.

When the `ctpu up` command has finished executing, verify that your shell prompt has changed from `username@projectname` to `username@vm-name`. This change shows that you are now logged into your Compute Engine VM.

```
gcloud compute ssh efficientnet-tutorial --zone=europe-west4-a
```

From this point on, a prefix of `(vm)$` means you should run the command on the Compute Engine VM instance.

Prepare the data

Set up the following environment variables, replacing ***bucket-name*** with the name of your Cloud Storage bucket:

1. Create an environment variable for your bucket name. Replace ***bucket-name*** with your bucket name.

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

2. Create some additional environment variables.

```
(vm)$ export MODEL_DIR=${STORAGE_BUCKET}/efficientnet
(vm)$ export DATA_DIR=gs://cloud-tpu-test-datasets/fake_imagenet
(vm)$ export TPU_NAME=efficientnet-tutorial
(vm)$ export PYTHONPATH=$PYTHONPATH:/usr/share/tpu/models
```

The training application expects your training data to be accessible in Cloud Storage. The training application also uses your Cloud Storage bucket to store checkpoints during training.

Train and evaluate the EfficientNet model with fake_imagenet

ImageNet is an image database. The images in the database are organized into a hierarchy, with each node of the hierarchy depicted by hundreds and thousands of images.

This tutorial uses a demonstration version of the full ImageNet dataset, referred to as *fake_imagenet*. This demonstration version allows you to test the tutorial, while reducing the storage and time requirements typically associated with running a model against the full ImageNet database.

The fake_imagenet dataset is at this location on Cloud Storage:

`gs://cloud-tpu-test-datasets/fake_imagenet`

The fake_imagenet dataset is only useful for understanding how to use a Cloud TPU and validating end-to-end performance. The accuracy numbers and saved model will not be meaningful.

For information on how to download and process the full ImageNet dataset, see Downloading, preprocessing, and uploading the ImageNet dataset.

> **Caution:** For this tutorial, **do not** set the **`STORAGE_BUCKET`** environment variable to the path of the fake_imagenet dataset. You can read from **`gs://cloud-tpu-test-datasets`** but you can't write to it. As a result, you can't use it to write out training logs. Make sure the **`STORAGE_BUCKET`** environment variable is set to your own Cloud Storage bucket, as shown above.

> **Note:** If you want to monitor the model's output and performance, follow the guide to setting up TensorBoard.

1. Launch a Cloud TPU resource using the ctpu utility.

```
(vm)$ ctpu up --tpu-only \
 --tf-version=1.15.3
 --name=${TPU_NAME}
```

2. Navigate to the model directory:

```
(vm)$ cd /usr/share/tpu/models/official/efficientnet/
```

3. Run the training script.

```
(vm)$ python3 main.py \
  --tpu=${TPU_NAME} \
  --data_dir=${DATA_DIR} \
  --model_dir=${MODEL_DIR} \
  --model_name='efficientnet-b0' \
  --skip_host_call=true \
  --train_batch_size=2048 \
  --train_steps=218948
```

Parameter	Description
`tpu`	Specifies the name of the Cloud TPU. Note that `ctpu` passes this name to the Compute Engine VM as an environment variable (`TPU_NAME`).
`data_dir`	Specifies the Cloud Storage path for training input. It is set to the fake_imagenet dataset in this example.
`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.

This procedure trains the EfficientNet model (`efficientnet-b0` variant) for 350 epochs and evaluates every fixed number of steps. Using the specified flags, the model should train in about 23 hours. With real imagenet data, the settings should obtain ~76.5% top-1 accuracy on ImageNet validation dataset. The best model checkpoint and corresponding evaluation result is inside the `archive` folder in the model directory: `${STORAGE_BUCKET}/efficientnet/archive`.

Scaling your model with Cloud TPU Pods

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

- v2-32
- v3-32

When working with Cloud TPU Pods, you first train the model using a Pod, then use a single Cloud TPU device to evaluate the model.

Training with Cloud TPU Pods

> **Note:** If you have already deleted your Compute Engine instance, create a new one following the steps in Set up your resources.

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

```
(vm)$ ctpu delete --tpu-only --name=efficientnet-tutorial
```

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 v2-32 Pod slice.

```
(vm)$ ctpu up --tpu-only \
  --tf-version=1.15.3 \
  --tpu-size=v2-32 \
  --name=efficientnet-tutorial-pod
```

3. Create an environment variable for your TPU name.

```
(vm)$ export TPU_NAME=efficientnet-tutorial-pod
```

4. Update the `MODEL_DIR` directory to store the training data.

```
(vm)$ export MODEL_DIR=${STORAGE_BUCKET}/efficientnet-tutorial-pod
```

5. Train the model.

```
(vm)$ python3 main.py \
  --tpu=${TPU_NAME} \
  --data_dir=${DATA_DIR} \
  --model_dir=${MODEL_DIR} \
  --model_name='efficientnet-b3' \
  --skip_host_call=true \
  --mode=train \
  --train_steps=109474 \
  --train_batch_size=4096 \
  --iterations_per_loop=100
```

Parameter	Description
`tpu`	Specifies the name of the Cloud TPU. Note that `ctpu` passes this name to the Compute Engine VM as an environment variable (`TPU_NAME`).
`data_dir`	Specifies the Cloud Storage path for training input. It is set to the fake_imagenet dataset in this example.
`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.

The procedure trains the EfficientNet model (`efficientnet-b3` variant) for 350 epochs. The model should get 81.1% accuracy on ImageNet dev set, which should finish in around 20 hours. The best model checkpoint and corresponding evaluation result is inside the `archive` folder in the model directory: `${STORAGE_BUCKET}/efficientnet/archive`.

> **Note:** The edge TPU efficientnets comes in three sizes: S, M, and L. For training the edgetpu versions of efficientnet, simply provide the model_name as **`efficientnet-edgetpu-{S,M,L}`**

Evaluating the model

In this set of steps, you use Cloud TPU to evaluate the above trained model against the fake_imagenet validation data.

1. Delete the Cloud TPU resource you created to train the model on a Pod.

```
(vm)$ ctpu delete --tpu-only --name=efficientnet-tutorial-pod
```

2. Start a v2-8 Cloud TPU. Use the same name that you used for the Compute Engine VM, which should still be running.

```
(vm)$ ctpu up --tpu-only \
 --tf-version=1.15.3 \
 --name=efficientnet-tutorial-eval
```

3. Create an environment variable for your TPU name.

```
(vm)$ export TPU_NAME=efficientnet-tutorial-eval
```

4. Run the model evaluation. This time, add the `mode` flag and set it to `eval`.

```
(vm)$ python3 main.py \
   --tpu=${TPU_NAME} \
   --data_dir=${DATA_DIR} \
   --model_dir=${MODEL_DIR} \
   --model_name='efficientnet-b3' \
   --skip_host_call=true \
   --mode=eval
```

This generates output similar to the following:

```
Eval results: {'loss': 7.532023,
'top_1_accuracy': 0.0010172526,
'global_step': 100,
'top_5_accuracy': 0.005065918}.
Elapsed seconds: 88
```

Cleaning up

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

1. Delete your Cloud TPU

```
$ ctpu delete --tpu-only \
  --name=efficientnet-tutorial-eval \
  --zone=europe-west4-a
```

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

```
(vm)$ exit
```

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

3. 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 --name=efficientnet-tutorial \
  --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.

4. Verify that you have no instances allocated to avoid unnecessary charges for TPU usage.

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

Deleting resources can 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:             --
```

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

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

Note: For free storage limits and other pricing information, see the Cloud Storage pricing guide.

What's next

- Learn more about `ctpu`, including how to install it on a local machine.
- Explore the TPU tools in TensorBoard.