Objective: This tutorial shows you how to train the Tensorflow ResNet-50 model (https://github.com/tensorflow/tpu/tree/master/models/official/resnet) using a Cloud TPU device or Cloud TPU Pod slice (multiple TPU devices). You can apply the same pattern to other TPU-optimised image classification models that use TensorFlow and the ImageNet dataset.

The model in this tutorial is based on Deep Residual Learning for Image Recognition (https://arxiv.org/pdf/1512.03385.pdf), which first introduces the residual network (ResNet) architecture. The tutorial uses the 50-layer variant, ResNet-50, and demonstrates training the model using TPUEstimator (https://www.tensorflow.org/api_docs/python/tf/estimator/tpu/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.

- 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.

This tutorial uses billable components of Google Cloud, including:

- Compute Engine
- Cloud TPU
- Cloud Storage

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

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

1. Sign in (https://accounts.google.com/Login) to your Google Account.

   If you don't already have one, sign up for a new account (https://accounts.google.com/SignUp).

2. 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 (https://console.cloud.google.com/projectselector2/home/dashboard)

3. Make sure that billing is enabled for your Google Cloud project. Learn how to confirm billing is enabled for your project (/billing/docs/how-to/modify-project).

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

This section provides information on setting up Cloud Storage 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 (https://console.cloud.google.com/?cloudshell=true)

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

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

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

   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 (/tpu/docs/storage-buckets).

   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 (/tpu/docs/types-zones#types), which are subdivisions within a region.

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

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

   For more information on the CTPU utility, see CTPU Reference (/tpu/docs/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@project` to `username@tpuname`. This change shows that you are now logged into your Compute Engine VM. If you are not connected to the Compute Engine instance, you can do so by running the following command:

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

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

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.

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:

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 (/tpu/docs/imagenet-setup).

Caution: For this tutorial, make sure you **don't** 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 (/tpu/docs/tensorboard-setup).

1. Launch a Cloud TPU resource using the ctpu utility and set it up as an environment variable (`TPU_NAME`).

2. Add the top-level `/models` folder to the Python path with the command

The ResNet-50 model is pre-installed on your Compute Engine VM.

3. Navigate to the directory:

4. Run the training script.

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.
`config_file`	Specifies the YAML configuration file to use during training. The name of this file corresponds to the type of TPU used. For example, `v2-8.yaml`.

For a single Cloud TPU device, the procedure trains the ResNet-50 model for 90 epochs and evaluates every fixed number of steps. Using the specified flags, the model should train in about 10 hours.

Since the training and evaluation was done on the fake_imagenet dataset, the output results do not reflect actual output that would appear if the training and evaluation was performed on a real dataset.

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.

You can get results faster by scaling your model with Cloud TPU Pods. The fully supported ResNet-50 model can work with the following Pod slices:

- v2-32
- v2-128
- v2-256
- v2-512
- v3-32
- v3-128
- v3-256
- v3-512
- v3-1024
- v3-2048

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.

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

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

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.

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

4. Train the model, updating the `config_file` parameter to use the configuration file that corresponds with the Pod slice you want to use. For example, the following script uses the `v2-32.yaml` configuration file.

The procedure trains the model on the fake_imagnet dataset to 35 epochs. This training takes approximately 90 minutes on a v3-128 Cloud TPU.

In this step, 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.

2. Start a v2-8 Cloud TPU.

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

This generates output similar to the following:

Since the training and evaluation was done on the fake_imagenet dataset, the output results do not reflect actual output that would appear if the training and evaluation was performed on a real dataset.

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:

   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:

★ **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 `ctpu status` to make sure you have no instances allocated to avoid unnecessary charges for TPU usage. The deletion might take several minutes. A response like the one below indicates there are no more allocated instances:

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

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

- Learn more about `ctpu` (https://github.com/tensorflow/tpu/tree/master/tools/ctpu), including how to install it on a local machine.
- Explore the TPU tools in TensorBoard (/tpu/docs/cloud-tpu-tools).

- See how to train ResNet with Cloud TPU and GKE (/tpu/docs/tutorials/kubernetes-engine-resnet).
- Speed up your training by streaming the data from Cloud Bigtable (/tpu/docs/tutorials/bigtable-resnet).
- Walk through the tutorial for the RetinaNet object detection model (/tpu/docs/tutorials/retinanet).
- Run the TensorFlow SqueezeNet model (https://github.com/tensorflow/tpu/tree/master/models/official/squeezenet) on Cloud TPU, using the above instructions as your starting point. The model architectures for SqueezeNet and ResNet-50 are similar. You can use the same data and the same command-line flags to train the model.