This tutorial contains a high-level description of the MNIST model, instructions on downloading the MNIST TensorFlow TPU code sample, and a guide to running the code on Cloud TPU.

This information is also covered in the Cloud TPU quickstart (/tpu/docs/quickstart).

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.

The MNIST dataset (http://yann.lecun.com/exdb/mnist/) contains a large number of images of hand-written digits in the range 0 to 9, as well as the labels identifying the digit in each image.

This tutorial trains a machine learning model to classify images based on the MNIST dataset. After training, the model classifies incoming images into 10 categories (0 to 9) based on what it's learned about handwritten images from the MNIST dataset. You can then send the model an image that it hasn't seen before, and the model identifies the digit in the image based on what the model has learned during training.

The MNIST dataset has been split into three parts:

- 55,000 examples of training data
- 10,000 examples of test data
- 5,000 examples of validation data

You can find more information about the dataset at the MNIST database site (http://yann.lecun.com/exdb/mnist/).

The model has a mixture of seven layers:

- 2 x convolution
- 2 x max pooling

- 2 x dense (fully connected)
- 1 x dropout

Loss is computed via softmax.

This version of the MNIST model uses tf.estimator (https://www.tensorflow.org/get_started/estimator) —a high-level TensorFlow API—which is the recommended way to build and run a machine learning model on a Cloud TPU.

The Tensorflow Estimator API simplifies the model development process by hiding most of the low-level implementation, which also makes it easy to switch between TPU and other test platforms such as GPUs or CPUs.

This model performs training only.

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.

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

You need a Cloud Storage bucket to store 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.

1. Go to the Cloud Storage page on the Cloud Console.

   Go to the Cloud Storage page (https://console.cloud.google.com/storage/browser)

2. Create a new bucket, specifying the following options:

   - A unique name of your choosing.
   - Select `Region` for Location type and `us-central1` for the Location (zone)
   - Default storage class: `Standard`
   - Location: Specify a bucket location in the same region where you plan to create your TPU node. See TPU types and zones (/tpu/docs/types-zones#types) to learn where various TPU types are available.

This section demonstrates using the Cloud TPU provisioning tool (https://github.com/tensorflow/tpu/tree/master/tools/ctpu) (`ctpu`) for creating and managing Cloud TPU project resources. The resources are comprised of a virtual machine (VM) and a Cloud TPU resource that have the same name. **These resources must reside in the same region/zone as the bucket you just created.**

You can also set up your VM and TPU resources using `gcloud` commands or through the Cloud Console (https://console.cloud.google.com/). See the creating and deleting TPUs (/tpu/docs/creating-deleting-tpus) page to learn all the ways you can set up and manage your Compute Engine VM and Cloud TPU resources.

1. Open a Cloud Shell window.

   Open Cloud Shell (https://console.cloud.google.com/?cloudshell=true)

2. Run `gcloud config set project <var>your-project</var>` to set the project where you want to create Cloud TPU.

3. Run `ctpu up` specifying the flags shown for either a Cloud TPU device or Pod slice. If you do not specify `tpu-size`, the default is a v2-8 Cloud TPU. Refer to CTPU Reference (/tpu/docs/ctpu-reference) for flag options and descriptions.

4. Set up a Cloud TPU device:

★ **Note:** If you have more than one project, you must specify the project name. If `--name` is not specified, it defaults to your username. If `--zone` is not specified, it defaults to `us-central1-b`. Make sure the zone matches the zone where you set up the storage bucket.

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

6. 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.

★ **Note:** If you are not connected to the Compute Engine instance, you can connect by running the following commands, replacing ***vm-name*** with the name of your VM:

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

The MNIST dataset is hosted on the MNIST database site (http://yann.lecun.com/exdb/mnist/). Follow the instructions below to download and convert the data to the required format, and to upload the converted data to Cloud Storage.

The convert_to_records.py (https://raw.githubusercontent.com/tensorflow/tensorflow/r1.15/tensorflow/examples/how_tos/reading_data/convert_to_records.py) script downloads the data and converts it to the TFRecord format expected by the example MNIST model.

Use the following commands to run the script and decompress the files:

Upload the data to your Cloud Storage bucket so that the TPU server can access the data. When setting the variable in the commands below, replace `YOUR-BUCKET-NAME` with the name of your Cloud Storage bucket:

The MNIST TPU model is pre-installed on your Compute Engine VM in the following directory:

The source code for the MNIST TPU model is also available on GitHub (https://github.com/tensorflow/models/blob/master/official/r1/mnist/mnist_tpu.py). You can run the model on a Cloud TPU. Alternatively, see how to run the model on a local machine (#run-local).

In the following steps, a prefix of `(vm)$` means you should run the command on your Compute Engine VM:

1. Run the MNIST model:

Parameter	Description
`tpu`	The name of the Cloud TPU. Note that `ctpu` passes this name to the Compute Engine VM as an environment variable (`TPU_NAME`). If you fail to connect to the VM, or lose your connection, you can connect by running `ctpu up` again. `TPU_NAME` *is not* set if you connect to the VM by running `gcloud compute ssh`.
`data_dir`	The directory that contains files used for training.
`model_dir`	The directory that contains the model files. This tutorial uses a folder within the Cloud Storage bucket. The script creates the folder if it does not already exist. 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.
`iterations`	The number of training steps to run on the TPU before returning control to Python. If this number is too small (for example, less than 100) then this can result in excessive communication overhead, which negatively impacts performance.
`train_steps`	The total number of steps (batches) for training to run.

To run the model on a non-TPU machine, omit `--tpu`, and set the following flag:

This causes the computation to land on a GPU if one is present. If no GPU is present, the computation falls back to the CPU.

By default, the tf.estimator model reports loss value and step time in the following format:

This model is extremely small and is not useful for benchmarking TPU performance.

To avoid incurring charges to your GCP account for the resources used in this topic:

1. Disconnect from the Compute Engine VM:

   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:

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.
- Verify performance on a large-scale model by running the ResNet sample (/tpu/docs/tutorials/resnet).
- Explore the TPU tools in TensorBoard (/tpu/docs/cloud-tpu-tools).