Processing large images with Cloud TPU

This document covers the use of spatial partitioning with Cloud TPUs and TPUEstimator to enable training for very large images and video.

The Cloud TPU spatial partitioning feature makes it possible to split up a single image across several TPU chips. Spatial partitioning allows you to easily scale models to run with input images too large to fit into the memory available to a single core on an accelerator chip.

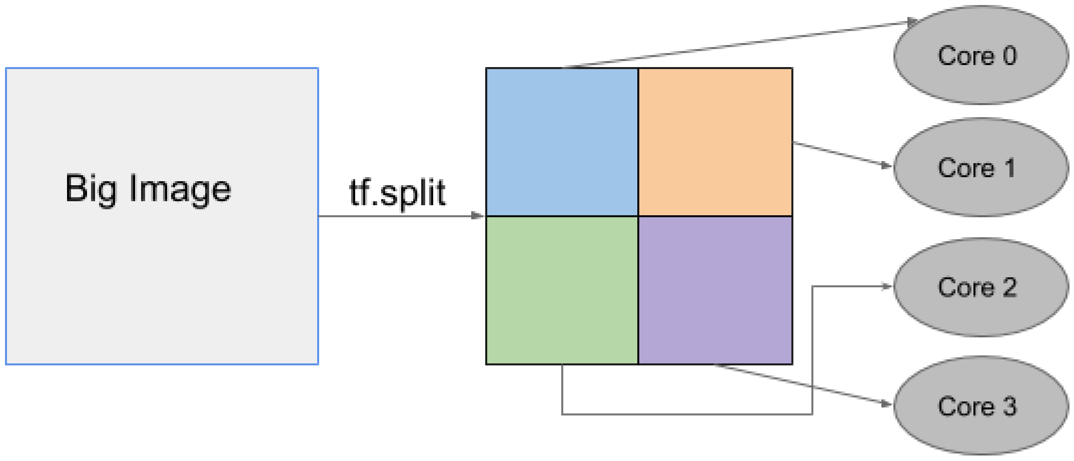
This feature distributes activations across multiple TPU cores, allowing you to scale your model to use 2, 4, 8 or even 16 cores for training. Scaling the number of processors available for training removes the need for downsampling image data which provides both better accuracy and more efficient training performance.

You can apply spatial partitioning to tasks such as:

- 3D Computer Tomography (CT) scan image segmentation
- video content analysis
- object detection for autonomous driving without downsampling your image data.

Enabling spatial partitioning with TPUEstimator

In Tensorflow, the XLA optimizing compiler for Tensorflow automatically handles the communications between all of the Cloud TPU cores. No changes to the code are required to enable spatial partitioning for a model. Because TPUEstimator supports the spatial partitioning API, all you need to do is to configure how to partition each input tensor in TPUConfig.



Example

The following code shows a TPUConfig example of four-way spatial partitioning for an image classification model. The tensor is split into four parts along the height dimension (assuming the tensor has the shape [batch, height, width, channel]).

```
tpu_config=tpu_config.TPUConfig(
    iterations_per_loop=100,
    num_cores_per_replica=4,
    per_host_input_for_training=tpu_config.InputPipelineConfig.PER_HOST_V2,
    input_partition_dims=[[1, 4, 1, 1], None]]
```

For spatial partitioning, the input pipeline must be in `tf.data` format and the `per_host_input_for_training` value (the `train_batch_size`) must be set to `PER_HOST_V2`. The `num_cores_per_replica` you specify determines the maximum number of partitions into which you can split an image.

The `input_partition_dims` values provide a list with two elements: `feature_partition_dims` and `label_partition_dims` that describe how to partition the input tensors. The structure of `feature_partition_dims` and `label_partition_dims` must match the structure of features and labels from `input_fn`. Specify "None" for the label partition so that labels are not split.

Running reference models with spatial partitioning

2D object detection

RetinaNet (https://github.com/tensorflow/tpu/tree/master/models/official/detection) is an object detection model that locates objects in images with a bounding box and classifies the identified objects. Training the model with very large images is a challenge since the largest image that can fit on a single Cloud TPU core (with per-device batch 8) is 1280x1280. You can train images that are 4x larger by spatially partitioning the model across 8 TPU cores, as shown.

Image size	TPU type	TPU cores	Global batch size	Step Time
1280x1280	v3-8	8	64	910 ms
2560x2560	v3-64	64	64	822 ms

3D image segmentation

3D UNet (https://github.com/tensorflow/tpu/tree/master/models/official/unet3d) is a popular dense 3D segmentation model which has been widely used in the medical image domain. The original resolution for CT images can be as large as 256x256x256, which do not fit into a single TPU core, so researchers must often downsample images. With TPU spatial partitioning, you can directly fit original resolution images using 16-way spatial partitioning.

Image size	TPU type	TPU cores	Global batch size	Step Time
128x128x128	v3-8	8	32	3.428s
2560x2560	v3-64	64	32	3.02s

Additional resources

- Spatial partitioning guide (https://github.com/tensorflow/estimator/blob/master/tensorflow_estimator/python/estimator/tpu/spatial_partitioning_api.md)
  - more instruction on how to configure spatial partitioning.
- Using the TPUEstimator API on Cloud TPU (https://cloud.google.com/tpu/docs/using-estimator-api)

Last updated December 4, 2019.