

# Image Classification

**Warning:** the features in the `image_classification/` directory have been fully integrated into the [new code base](#).

This folder contains TF 2 model examples for image classification:

- [MNIST](#)
- [Classifier Trainer](#), a framework that uses the Keras compile/fit methods for image classification models, including:
  - ResNet
  - EfficientNet<sup>[^1]</sup>

[^1]: Currently a work in progress. We cannot match "AutoAugment (AA)" in [the original version](#). For more information about other types of models, please refer to this [README file](#).

## Before you begin

Please make sure that you have the latest version of TensorFlow installed and add the models folder to your Python path.

## ImageNet preparation

### Using TFDS

`classifier_trainer.py` supports ImageNet with [TensorFlow Datasets \(TFDS\)](#).

Please see the following [example snippet](#) for more information on how to use TFDS to download and prepare datasets, and specifically the [TFDS ImageNet readme](#) for manual download instructions.

### Legacy TFRecords

Download the ImageNet dataset and convert it to TFRecord format. The following [script](#) and [README](#) provide a few options.

Note that the legacy ResNet runners, e.g. [resnet/resnet\\_ctl\\_imagenet\\_main.py](#), require TFRecords whereas

`classifier_trainer.py` can use both by setting the builder to 'records' or 'tfds' in the configurations.

## Running on Cloud TPUs

Note: These models will **not** work with TPUs on Colab.

You can train image classification models on Cloud TPUs using [tf.distribute.TPUStrategy](#). If you are not familiar with Cloud TPUs, it is strongly recommended that you go through the [quickstart](#) to learn how to create a TPU and GCE VM.

## Running on multiple GPU hosts

You can also train these models on multiple hosts, each with GPUs, using [tf.distribute.experimental.MultiWorkerMirroredStrategy](#).

The easiest way to run multi-host benchmarks is to set the [TF\\_CONFIG](#) appropriately at each host. e.g., to run using `MultiWorkerMirroredStrategy` on 2 hosts, the `cluster` in `TF_CONFIG` should have 2 `host:port` entries, and host `i` should have the `task` in `TF_CONFIG` set to `{"type": "worker", "index": i}`. `MultiWorkerMirroredStrategy` will automatically use all the available GPUs at each host.

## MNIST

To download the data and run the MNIST sample model locally for the first time, run one of the following command:

► Details

To train the model on a Cloud TPU, run the following command:

► Details

Note: the `--download` flag is only required the first time you run the model.

## Classifier Trainer

The classifier trainer is a unified framework for running image classification models using Keras's compile/fit methods. Experiments should be provided in the form of YAML files, some examples are included within the `configs/examples` folder. Please see [configs/examples](#) for more example configurations.

The provided configuration files use a per replica batch size and is scaled by the number of devices. For instance, if `batch_size` = 64, then for 1 GPU the global batch size would be  $64 * 1 = 64$ . For 8 GPUs, the global batch size would be  $64 * 8 = 512$ . Similarly, for a v3-8 TPU, the global batch size would be  $64 * 8 = 512$ , and for a v3-32, the global batch size is  $64 * 32 = 2048$ .

### ResNet50

#### On GPU:

► Details

To train on multiple hosts, each with GPUs attached using [MultiWorkerMirroredStrategy](#), please update `runtime` section in `gpu.yaml` (or override using `--params_override` ) with:

► Details

By having `task_index: 0` on the first host and `task_index: 1` on the second and so on. `$HOST1` and `$HOST2` are the IP addresses of the hosts, and `port` can be chosen any free port on the hosts. Only the first host will write TensorBoard Summaries and save checkpoints.

#### On TPU:

► Details

### VGG-16

#### On GPU:

► Details

### EfficientNet

**Note: EfficientNet development is a work in progress.**

#### On GPU:

► Details

#### On TPU:

► Details

Note that the number of GPU devices can be overridden in the command line using `--params_overrides` . The TPU does not need this override as the device is fixed by providing the TPU address or name with the `--tpu` flag.