TensorFlow 2.13.0 Testing

one, Automatic testing:

Will 2.script 、 3.yaml 4. Copy all the data to the current path.

Execute the following command to give all script files execution permissions

|  |
| --- |
| chmod +x 2.script/\* |

Execute the following command to build the test file.

|  |
| --- |
| ./2.script/build.sh |

Start the test and execute the commands one by one.

**examples**

|  |
| --- |
| ./examples.sh |

**models**

|  |
| --- |
| ./recommendation.sh  ./vision.sh  ./projects.sh |

**Benchmark**

|  |
| --- |
| ./benchmark.sh |

**others**

|  |
| --- |
| ./others.sh |

2. Manual testing:

Note: The path should be switched to your own path, and cannot be directly copied and pasted.

**examples**

The warehouse is There are in the tensorflow\_examples/models/ path dcgan 、 densenet 、​

nmt \_ with \_ attention and There are four models in pix 2 pix , which are tested below.

First clone the repository and set the environment variables `PYTHONPATH` , as follows :

Clone the repository

|  |
| --- |
| git clone [https://github.com/tensorflow/e examples](https://github.com/tensorflow/examples) |

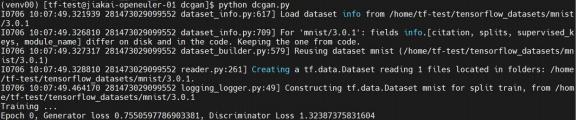
Bundle The examples repository is added to PYTHONPATH Environment variables

|  |
| --- |
| export PYTHONPATH="$PYTHONPATH:/home/tf-test/file /examples" |

**dcgan**

Here train .

|  |
| --- |
| python dcgan.py |



**densenet**

Here train .

|  |
| --- |
| python train.py |



**nmt\_with\_attention**

Here train .

|  |
| --- |
| python train.py |



**pix2pix**

Here train .

Download the dataset [facades] to tensorflow\_examples/ models/pix2pix/ and unzip it facades.tar .

|  |
| --- |
| Python pix 2 pix .py - path / home / tf-​​  test/file/examples/tensorflow\_examples/models/pix2pix/faca des/ |



**models**

This time we mainly tested the warehouse recommendation 、 nlp , vision and Projects The model in Clone the repository and set the environment variables:

Clone the repository

|  |
| --- |
| git clone [https://github.com/tensorfl ow/models](https://github.com/tensorflow/models) |

Bundle The models repository is added to PYTHONPATH environment variable

|  |
| --- |
| export PYTHONPATH="$PYTHONPATH:/home/tf-test/ file/models" |

**recommendation**

recommendation The model in the path Under models/official/recommendation .

***NCF***

NCF to download and preprocess the dataset and train and evaluate the model are as follows. The working path is models/official/recommendation / :

Download and preprocess the dataset, select ml- 1m dataset

|  |
| --- |
| python movielens.py --data\_dir /home/tf-test/models/dataset/ncf --dataset ml- 1m |

Training and evaluating the model

|  |
| --- |
| python ncf\_keras\_main.py --model\_dir /home/tf-test/models/model\_dir/ncf --data\_dir |

|  |
| --- |
| /home/tf-test/models/dataset/ncf/ --data set ml- 1m --num\_gpus 0 |



***DCN v 2***

DCN v2 has the following steps for training with synthetic data:

|  |
| --- |
| python3 models/official/recommendation/ranking/train.py --mode=eval \ --model\_dir=/home/tf-test/models/model\_dir/dcn\_v2 --pa rams\_override="  runtime:  distribution\_strategy: ' one\_device'  task:  use\_synthetic\_data: true  model:  num\_dense\_features: 13  bottom\_mlp: [512,256,2]  embedding\_dim: 2  top\_mlp: [1024, 1024,512,256, 1]  interaction: 'cross'  vocab\_sizes: [39884406, 39043, 17289, 7420, 20263, 3, 7120, 1543, 63,  38532951, 2953546, 403346, 10, 2208, 11938, 155 , 4, 976, 14,  39979771, 25641295, 39664984, 58593 5, 12972, 108, 36]  trainer:  validation\_steps: 2  " |



***DLRM***

DLRM has the following steps for training with synthetic data:

|  |
| --- |
| python3 models/official/recommendation/ranking/train.py --mode=eval \ --model\_dir=/home/tf-test/models/model\_dir/dlrm --para ms\_override="  runtime:  distribution\_strategy: ' one\_device'  task:  use\_synthetic\_data: true  model:  num\_dense\_features: 13  bottom\_mlp: [512,256,2]  embedding\_dim: 2  top\_mlp: [1024, 1024,512,256, 1]  interaction: 'dot'  vocab\_sizes: [39884406, 39043, 17289, 7420, 20263, 3, 7120, 1543, 63,  38532951, 2953546, 403346, 10, 2208, 11938, 155 , 4, 976, 14,  39979771, 25641295, 39664984, 58593 5, 12972, 108, 36]  trainer:  validation\_steps: 2  " |



**nlp**

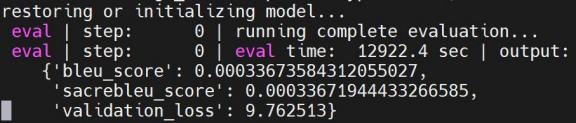
nlp The model in the path Run under models/official/nlp/ .

***Transformer***

Transformer is as follows:

|  |
| --- |
| Python train.py --experiment​​​ wmt\_transformer / large -- mode​​ eval --model\_dir / home / tf-​​​​  test/models/model\_dir/transformer --params\_ove rride  task.sentencepiece\_model\_path='gs://tf\_model\_garden/nlp/transformer\_wmt/ende\_b pe\_3 |

|  |
| --- |
| 2k.model' |



**vision**

vision The model in the path Run under models/official/vision/ .

In operation Before models , you need to download and preprocess ILSVRC2012 and co co2017 dataset, and finally Include `train\*` and `val\* ` The ILSVRC2012 dataset folder is renamed `imagenet-2012-

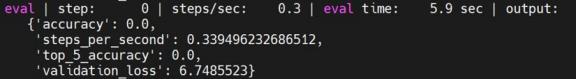
tfrecord` and put it in vision Directory; include `val\*` and ` ins tances\_val2017.json` coco2017 Rename the dataset folder to Coco and put it vision directory.

Then run the following model.

***vit\_imagenet\_finetune***

The specific test method is as follows:

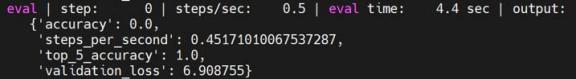
|  |
| --- |
| python train.py -experiment vit\_imagenet\_finetune -mode eval -model\_dir /home/tf- test/models/model\_dir/vit\_imagenet\_finetune --params\_overr ide="  runtime:  distribution\_strategy: ' one\_device'  task:  validation\_data:  global\_batch\_size: 2  trainer:  validation\_steps: 2  validation\_interval: 2  " |



***vit\_imagenet\_pretrain***

The specific test method is as follows:

|  |
| --- |
| python train.py -experiment vit\_imagenet\_pretrain -mode eval -model\_dir /home/t f- test/models/model\_dir/vit\_imagenet\_pretrain --params\_over ride="  runtime:  distribution\_strategy: ' one\_device'  task:  validation\_data:  global\_batch\_size: 2  trainer:  validation\_steps: 2  validation\_interval: 2  " |

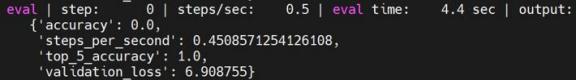


***deit\_imagenet\_pretrain***

The specific test method is as follows:

|  |
| --- |
| python train.py -experiment deit\_imagenet\_pretrain -mode eval -model\_dir /home/ tf- test/models/model\_dir/deit\_imagenet\_pretrain --params\_ove rride="  runtime:  distribution\_strategy: ' one\_device'  task: |

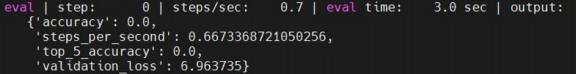
|  |
| --- |
| validation\_data:  global\_batch\_size: 2  trainer:  validation\_steps: 2  validation\_interval: 2  " |



***mobilenet\_imagenet​​***

The specific test method is as follows:

|  |
| --- |
| python train.py -experiment mobilenet\_imagenet -mode eval -model\_dir /home/tf- test/models/model\_dir/mobilenet\_imagenet --params\_ove rride="  runtime:  distribution\_strategy: ' one\_device'  task:  validation\_data:  global\_batch\_size: 2  trainer:  validation\_steps: 2  validation\_interval: 2  " |



***revnet\_imagenet​​***

The specific test method is as follows:

|  |
| --- |
| python train.py -experiment revnet\_imagenet -mode eval -model\_dir /home/tf- test/models/model\_dir/revnet\_imagenet --params\_ove rride="  runtime:  distribution\_strategy: ' one\_device'  task:  validation\_data:  global\_batch\_size: 2  trainer:  validation\_steps: 2  validation\_interval: 2  " |

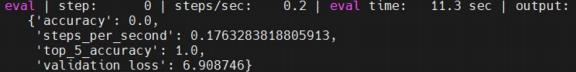


***resnet\_rs\_imagenet​​​​***

The specific test method is as follows:

|  |
| --- |
| python train.py -experiment resnet\_rs\_imagenet -mode eval -model\_dir /home/tf- test/models/model\_dir/resnet rs imagenet --params\_override ="  runtime:  distribution\_strategy: ' one\_device'  task:  validation\_data:  global\_batch\_size: 2  trainer:  validation\_steps: 2  validation\_interval: 2 |

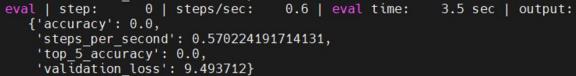
|  |
| --- |
| " |



***resnet\_imagenet​​***

The specific test method is as follows:

|  |
| --- |
| python train.py -experiment resnet\_imagenet -mode eval -model\_dir /home/ tf- test/models/model\_dir/resnet\_imagenet --params\_over ride="  runtime:  distribution\_strategy: ' one\_device'  task:  validation\_data:  global\_batch\_size: 2  trainer:  validation\_steps: 2  validation\_interval: 2  " |

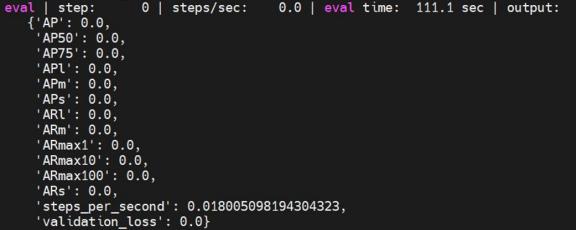


***fasterrcnn\_resnetfpn\_coco​​​​***

The specific test method is as follows:

|  |
| --- |
| python train.py -experiment fasterrcnn\_resnetfpn\_coco -mode eval -model \_dir /home/tf- test/models/model\_dir/fasterrcnn\_resnetfpn\_coco --params\_ove rride="  runtime:  distribution\_strategy: ' one\_device'  task: |

|  |
| --- |
| validation\_data:  global\_batch\_size: 2  trainer:  validation\_steps: 2  validation\_interval: 2  " |

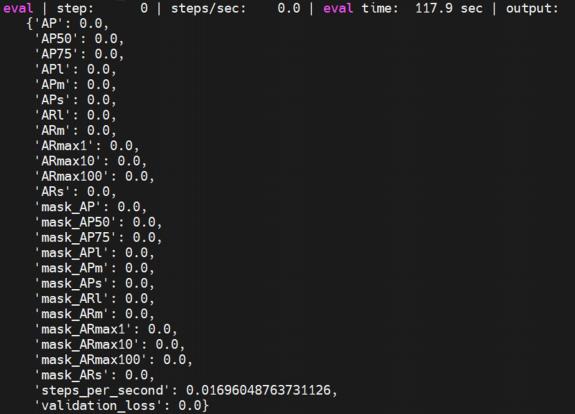


***maskrcnn\_resnetfpn\_coco​***

The specific test method is as follows:

|  |
| --- |
| python train.py -experiment maskrcnn\_resnetfpn\_coco -mode eval -model\_dir /home/tf- test/models/model\_dir/maskrcnn\_resnetfpn\_coco --params\_overr ide="  runtime:  distribution\_strategy: ' one\_device'  task:  validation\_data:  global\_batch\_size: 2  trainer:  validation\_steps: 2  validation\_interval: 2 |

|  |
| --- |
| " |

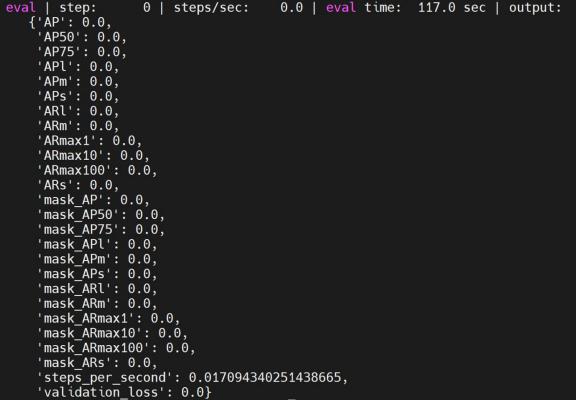


***maskrcnn\_spinenet\_coco***

The specific test method is as follows:

|  |
| --- |
| python train.py -experiment maskrcnn\_spinenet\_coco -mode eval -model\_dir /home/tf- test/models/model\_dir/maskrcnn\_spinenet\_coco --params\_ov erride="  runtime:  distribution\_strategy: ' one\_device'  task:  validation\_data:  global\_batch\_size: 2  trainer:  validation\_steps: 2  validation\_interval: 2 |

|  |
| --- |
| " |

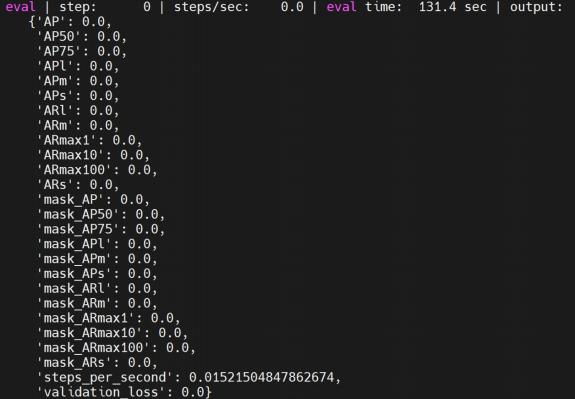


***cascadercnn\_spinenet\_coco***

The specific test method is as follows:

|  |
| --- |
| python train.py -experiment cascadercnn\_spinenet\_coco -mode eval -model\_dir /home/tf- test/models/model\_dir/cascadercnn\_spinenet\_coco --params\_ove rride="  runtime:  distribution\_strategy: ' one\_device'  task:  validation\_data:  global\_batch\_size: 2  trainer:  validation\_steps: 2 |

|  |
| --- |
| validation\_interval: 2  " |

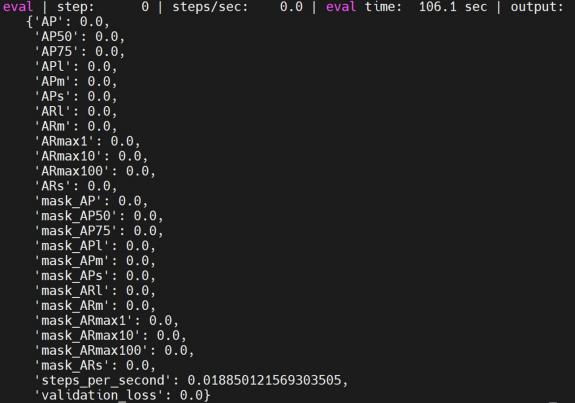


***maskrcnn\_mobilenet\_coco***

The specific test method is as follows:

|  |
| --- |
| python train.py -experiment maskrcnn\_mobilenet\_coco -mode eval -model\_dir /home/tf- test/models/model\_dir/maskrcnn\_mobilenet\_coco --params\_ove rride="  runtime:  distribution\_strategy: ' one\_device'  task:  validation\_data:  global\_batch\_size: 2  trainer:  validation\_steps: 2 |

|  |
| --- |
| validation\_interval: 2  " |

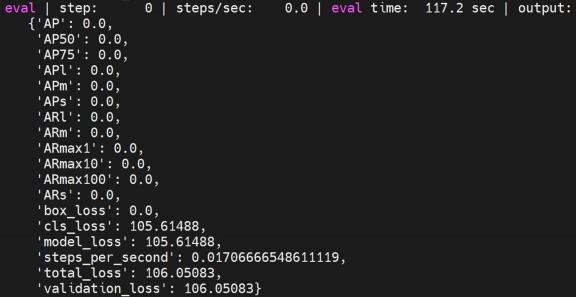


***retinanet\_resnetfpn \_coco***

The specific test method is as follows:

|  |
| --- |
| python train.py -experiment retinanet\_resnetfpn\_coco -mode eval -model\_dir /home/tf- test/models/model\_dir/retinanet\_resnetfpn\_coco --params\_ove rride="  runtime:  distribution\_strategy: ' one\_device'  task:  validation\_data:  global\_batch\_size: 2  trainer:  validation\_steps: 2 |

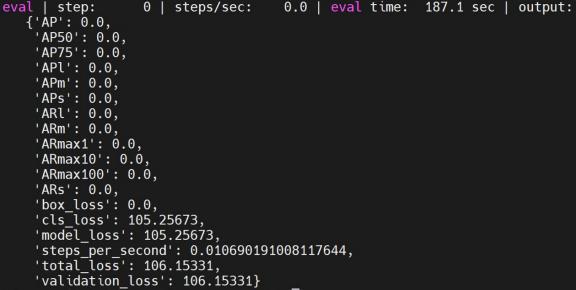
|  |
| --- |
| validation\_interval: 2  " |



***retinanet\_spinenet\_coco***

The specific test method is as follows:

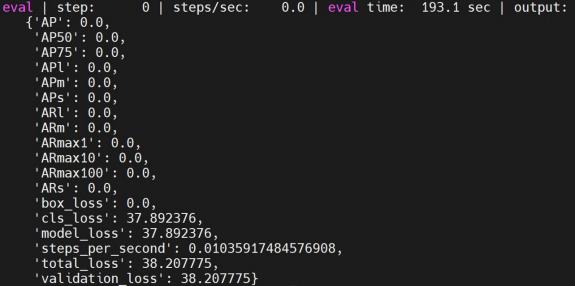
|  |
| --- |
| python train.py -experiment retinanet\_spinenet\_coco -mode eval -model\_dir /home/tf- test/models/model\_dir/retinanet\_spinenet\_coco --params\_overr ide="  runtime:  distribution\_strategy: ' one\_device'  task:  validation\_data:  global\_batch\_size: 2  trainer:  validation\_steps: 2  validation\_interval: 2  " |



***retinanet\_mobile\_coco***

The specific test method is as follows:

|  |
| --- |
| python train.py -experiment retinanet\_mobile\_coco -mode eval -model\_dir /home/tf- test/models/model\_dir/retinanet\_mobile\_coco --params\_ove rride="  runtime:  distribution\_strategy: ' one\_device'  task:  validation\_data:  global\_batch\_size: 2  trainer:  validation\_steps: 2  validation\_interval: 2  " |



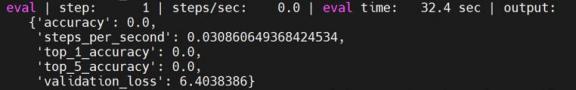
**Projects**

Projects The model in the path models/official/projects/ under.

***\*\*assemblenet\*\****

specific The eval is as follows:

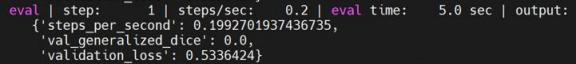
|  |
| --- |
| python train\_test.py |



***\*\*volumetric\_models \*\****

specific The eval is as follows:

|  |
| --- |
| python train\_test.py |



***\*\*nhnet\*\****

specific The eval is as follows:

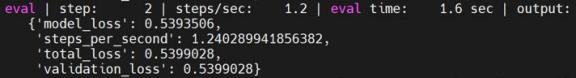
|  |
| --- |
| python trainer\_test.py |



***\*\*yt8m \*\****

specific The eval is as follows:

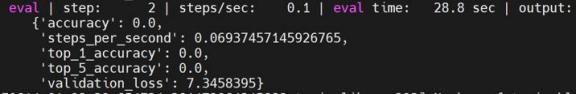
|  |
| --- |
| python train\_test.py |



***\*\*MOVENT\*\****

specific The eval is as follows:

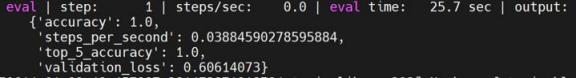
|  |
| --- |
| python train\_test.py |



***\*\*maxvit\*\****

specific The eval is as follows:

|  |
| --- |
| python train\_test.py |



**benchmark**

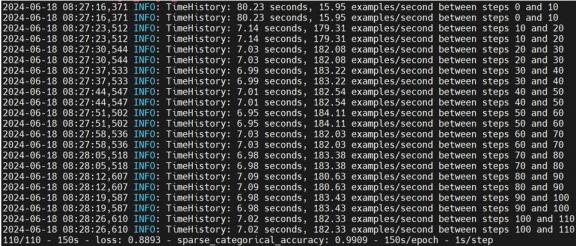
The benchmark model needs to be cloned benchmark repository, and clone the above The models repository is switched to have Benchmark module v2.15.0 tags , as follows:

|  |
| --- |
| git [clone https://github.com/tensorflow/benchmarks](https://github.com/tensorflow/benchmarks)  cd models  git checkout v2.15.0 |

**resnet56**

resnet56​ In the models/official/benchmark/ keras\_cifar\_benchmark.py file, the combined Test with data benchmark\_cpu , so you can test it directly as follows:

|  |
| --- |
| python 3 benchmarks / perfzero / lib / benchmark.py --​​  git\_repos=" <https://github.com/tensorflow/models.git>;benchmark" --  gcloud\_key\_file\_url="" --  benchmark\_methods=official.benchmark.keras\_cifar\_benchmark.Resnet56KerasBen chm arkSynth.benchmark\_cpu |



**resnet50**

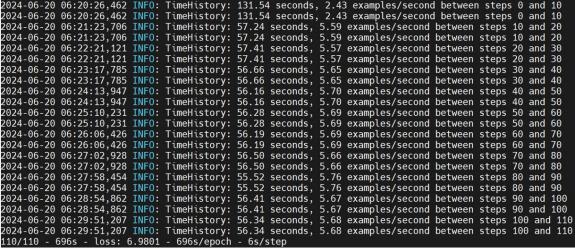
resnet50​ In the models/official/benchmark/keras\_imagenet\_benchmark.py file , Tested with synthetic data, it comes with benchmark 1 GPU , when the program is running, it will detect whether there is a Used GPU If it does not exist, it will be used The cpu is tested, so it can be tested directly as follows:

Note: You can modify In the keras\_imagenet\_benchmark.py file

KerasClassifierBenchmark Base class benchmark 1 GPU method

per\_replica\_batch\_size value to reduce batch\_size To shorten the testing time.

|  |
| --- |
| python 3 benchmarks / perfzero / lib / benchmark.py --​​  git\_repos=" <https://github.com/tensorflow/models.git>;benchmark" --  gcloud\_key\_file\_url="" --  benchmark\_methods=official.benchmark.keras\_imagenet\_benchmark.Resnet50Keras Ben chmarkSynth.benchmark 1 GPU |



**mobilenetv1**

MobileNetV1​ In the models/official/benchmark/ keras\_imagenet\_benchmark.py file, In order to facilitate testing, we choose to use synthetic data here, so you need to write the corresponding

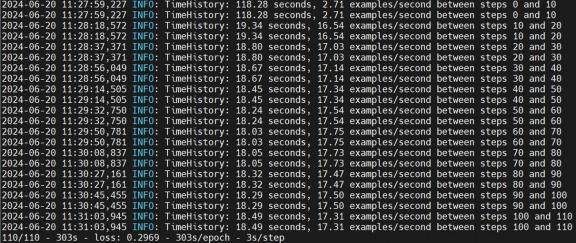
KerasPruningBenchmarkSynthBase and MobilenetV1KerasPrun ingBenchmarkSynth class, The details are as follows:

|  |
| --- |
| class KerasPruningBenchmarkSynthBase ( Resnet 50 KerasBenchmarkBase ):  """Pruning method benchmarks."""  def \_\_init \_\_ ( self , root \_ data \_ dir = None , default \_ flags = None , \*\* kwargs ):  if default\_flags​​ is None :  default\_flags = {}  default\_flags.update({  'skip\_eval': True,  'report\_accuracy\_metri cs': False,  'use\_synthetic\_data': True,  'train\_steps': 110,  'log\_steps': 10,  'pruning\_method': 'pol ynomial\_decay', |

|  |
| --- |
| 'pruning\_begin\_step': 0,  'pruning\_end\_step': 50000,  'pruning\_initial\_sparsity': 0,  'pruning\_final\_sparsity': 0.5,  'pruning\_frequency': 100,  })  super(KerasPruningBenchmarkSynthBase, self).\_\_in it\_\_(  default\_flags=default\_flags, \*\*kwargs)  class MobilenetV 1 KerasPruningBenchmarkSynth ( KerasPruningBenchmarkSynthBase ):  """Pruning method benchmarks for MobilenetV1."""  def \_\_init \_\_ ( self , \*\* kwargs ):  default\_flags = {  'model': 'mobilenet',  'optimizer': 'mobile net\_default',  }  super(MobilenetV1KerasPruningBenchmarkSynth, self).\_\_in it\_\_(  default\_flags=default\_flags, \*\*kwargs) |

It can also be adjusted batch\_size to shorten the test time.

|  |
| --- |
| python 3 benchmarks / perfzero / lib / benchmark.py --​​  git\_repos=" <https://github.com/tensorflow/models.git>;benchmark" --  gcloud\_key\_file\_url="" --  benchmark\_methods=official.benchmark.keras\_imagenet\_benchmark.MobilenetV1Ke ras PruningBenchmarkSynth.benchmark 1 GPU |



**trivial**

trivial​ In the models/official/benchmark/ keras\_imagenet\_benchmark.py file, for convenience

Test, here we choose to use synthetic data, so you need to write the corresponding

TrivialKerasBenchmarkSynth class is as follows:

|  |
| --- |
| class TrivialKerasBenchmarkSynth(keras\_benchmark.Kera sBenchmark):  """Trivial model with synth data b enchmark tests."""  def \_\_ init \_\_( self , output \_ dir = None , root \_ data \_ dir = None , \*\* kwargs ):  flag\_methods = [resnet\_imagenet\_m ain.define\_imagenet\_keras\_flags]  def\_flags = {}  def\_flags['use\_trivial\_mod el'] = True  def\_flags['skip\_eval'] = True  def\_flags['report\_accuracy\_metrics '] = False  def\_flags['dtype'] = 'fp16'  def\_flags['use\_synthetic\_dat a'] = True  def\_flags['train\_ste ps'] = 600  def\_flags['log\_steps'] = 100 |

|  |
| --- |
| def\_flags['distribution\_strategy'] = 'mirrored'  super(TrivialKerasBenchmarkSynth, self).\_\_in it\_\_(  output\_dir=output\_dir,  flag\_methods=flag\_methods,  default\_flags=def\_flags)  @benchmark\_wrappers.enable \_runtime\_flags  def \_ run \_ and \_ report \_ benchmark ( self ):  start\_time\_sec = time.time()  stats = resnet\_imagenet\_ main.run(FLAGS)  wall\_time\_sec = time.time() - start\_time \_sec  super(TrivialKerasBenchmarkSynth, self).\_re port\_benchmark(  stats,  wall\_time\_sec,  total\_batch\_size=FLAGS.batch\_size ,  log\_steps=FLAGS.log\_steps)  def benchmark\_cpu ( self ) :​  self.\_setup()  FLAGS.num\_gpus = 0  FLAGS.enable\_eager = True  FLAGS.model\_dir = self.\_get\_model\_dir('benchmar k\_cpu') |

|  |
| --- |
| FLAGS.batch\_size = 32  FLAGS.train\_steps = 10  self.\_run\_and\_report\_benchmark()  def fill\_report\_object ( self , stats ) :​​​  super(TrivialKerasBenchmarkSynth, self).fil l\_report\_object(  stats,  total\_batch\_size=FLAGS.batch\_size ,  log\_steps=FLAGS.log\_steps) |

|  |
| --- |
| python 3 benchmarks / perfzero / lib / benchmark.py --​​  git\_repos=" <https://github.com/tensorflow/models.git>;benchmark" --  gcloud\_key\_file\_url="" --  benchmark\_methods=official.benchmark.keras\_imagenet\_benchmark.TrivialKerasB ench markSynth.benchmark\_cpu |



**efficientnet**

efficientnet​ In the models/official/ benchmark/keras\_imagenet\_benchmark.py file, in order to

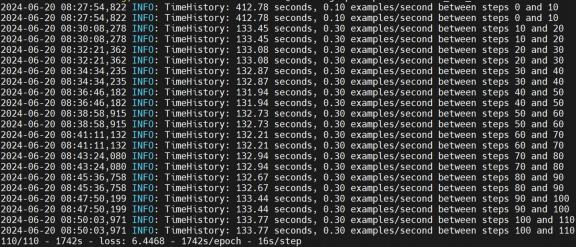
To facilitate testing, we choose to use synthetic data here, so you need to write the corresponding

EfficientNetKerasBenchmarkSynth class , as follows:

|  |
| --- |
| class EfficientNetKerasBenchmarkSynth ( KerasClassifierBenchmarkBase ):  """EfficientNet synth data benchmark tests."""  def \_\_init\_\_(self, output\_dir=None, root\_data\_dir=None, tpu=None, \*\*kwargs):  def\_flags = {}  def\_flags['log\_steps'] = 10 |

|  |
| --- |
| super(EfficientNetKerasBenchmarkSynth, self).\_\_in it\_\_(  model='efficientnet', output\_dir=output\_dir, default\_flags=def\_ flags,  tpu=tpu, dataset\_builder='synthetic', train\_epochs= 1, train\_steps= 110)  def benchmark\_cpu ( self ) :​  self.\_setup()  self.\_run\_and\_repor t\_benchmark(  experiment\_name='ben chmark\_cpu',  model\_variant='efficientnet -b7',  dtype = 'bfloat16',  num\_tpus=0,  distribution\_strategy='o ne\_device',  per\_replica\_batch\_size=4) |

|  |
| --- |
| python 3 benchmarks / perfzero / lib / benchmark.py --​​  git\_repos=" <https://github.com/tensorflow/models.git>;benchmark" --  gcloud\_key\_file\_url="" --  benchmark\_methods=official.benchmark.keras\_imagenet\_benchmark.EfficientNetKe ras BenchmarkSynth.bench mark\_cpu |



**resnet50ctl**

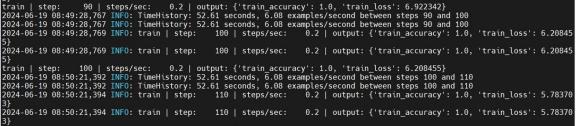
resnet50ctl​ In the models/official/benchmark/resnet\_ctl\_imagenet\_benchmark.py file ,

Here we use synthetic data for testing. benchmark 1 GPU , when the program is running, it will detect whether There is available GPU If it does not exist, it will be used cpu To test, you can directly test, such as

Down:

In addition, Resnet50CtlBenchmarkBase class​ benchmark 1 GPU method batch\_size Change to 32 Otherwise, it will appear The problem of insufficient RAM .

|  |
| --- |
| python 3 benchmarks / perfzero / lib / benchmark.py --​​  git\_repos=" <https://github.com/tensorflow/models.git>;benchmark" --  gcloud\_key\_file\_url="" --  benchmark\_methods=official.benchmark.resnet\_ctl\_imagenet\_benchmark.Resnet50CtlB enchmarkSynth.benchmark 1 GPU |



**TfScanE2E**

Testing and Evaluation Benchmark of the overall end-to-end performance of TensorFlow models.

TFScanE2E​ In the models/official/ben chamrk/tf\_scan\_benchmark.py file, there is a built-in benchmark\_cpu , as follows:

|  |
| --- |
| python 3 benchmarks / perfzero / lib / benchmark.py --​​  git\_repos=" <https://github.com/tensorflow/models.git>;benchmark" --  gcloud\_key\_file\_url="" --  = official.benchmark.tf\_scan\_benchmark.TfScanE2EBenchmark.ben chmark\_cpu |



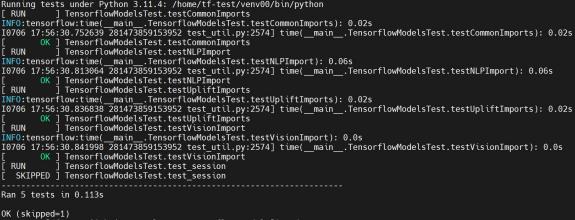
**others**

Here are some small test .

**\*\*tensorflow\_models\*\***

Working path: /home/tf-test/file/models/ tensorflow\_models

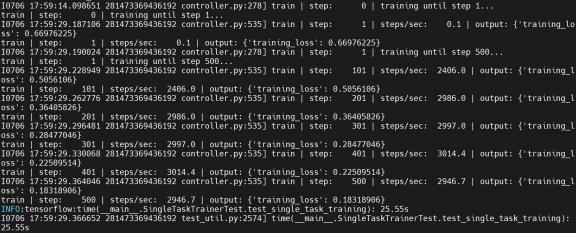
|  |
| --- |
| python tensorflow\_models\_tes t.py |



**ORBIT**

Working path: /home/tf-test/file/models/orbit/ examples/single\_task

|  |
| --- |
| python single\_task\_trainer\_test.py |



|  |
| --- |
| python single\_task\_evaluator\_test .py |

