ResNet50 Keras Model Scaling

TODO

For those who need to count HPUs in use on a habana node, this hack works:

```
echo $(hl-smi | grep "Mib \/" | grep -v 512 | wc -l)
```

Login

On your development machine:

```
ssh -J CELSGCEUserID@homes.cels.anl.gov CELSGCEUserID@habana-
01.ai.alcf.anl.gov
CELS password
```

Clone ResNet50

```
cd /path/to/location/for/repo
git clone git@git.cels.anl.gov:ai-testbed-apps/habana/resnet50.git

cd resnet50
```

Create Venv

```
python3.8 -m venv --system-site-packages ~/venvs/habana/resnet50_TF_venv

source ~/venvs/habana/resnet50_TF_venv/bin/activate
```

Define PYTHON

Set the Python environment variable:

```
export PYTHON=$(which python)
```

Define PYTHONPATH

Set the Python path:

```
export PYTHONPATH=/path/to/Model-References/:$(which python)
```

Environment Variables

Set these:

```
export HABANA_LOGS=~/.habana_logs
export MPI_ROOT=/usr/local/openmpi
```

Change Directory

If necessary:

```
cd /path/to/ai-testbed-apps/habana/resnet50
```

Install Requirements

```
cd /path/to/ai-testbed-apps/habana/resnet50/Resnets/resnet_keras
```

Install required packages using pip:

```
python3 -m pip install -r requirements.txt
```

Training data

Skip this because it has already been done.

Also, the commands may not be quite right.

The ResNet50 Keras script operates on ImageNet 1k, a widely popular image classification dataset from the ILSVRC challenge. In order to obtain the dataset, follow these steps which can take over twelve hours:

1. Sign up with http://image-net.org/download-images and acquire the rights to download original images
2. Follow the link to the 2012 ILSVRC and download `ILSVRC2012_img_val.tar` and `ILSVRC2012_img_train.tar`.
3. Use the commands below - they will prepare the dataset under `/lambda_stor/data/imagenet/tf_records`. This is the default data_dir for the training script. In `/lambda_stor/data/imagenet/tf_records/train` and `/lambda_stor/data/imagenet/tf_records/val` directories original JPEG files will stay and can

be used for Media Loading Acceleration on Gaudi2. See examples with --data_dir and --jpeg_data_dir parameters for details.

```
export IMAGENET_HOME=/lambda_stor/data/imagenet/tf_records
export PYTHON=$(which python)
source ~/venvs/habana/resnet50_TF_venv/bin/activate
mkdir -p $IMAGENET_HOME/validation
mkdir -p $IMAGENET_HOME/train
tar xf ILSVRC2012_img_val.tar -C $IMAGENET_HOME/validation
tar xf ILSVRC2012_img_train.tar -C $IMAGENET_HOME/train
cd $IMAGENET_HOME/train
for f in *.tar; do
  d=`basename $f .tar`
  mkdir $d
  tar xf $f -C $d
done
cd $IMAGENET_HOME
rm $IMAGENET_HOME/train/*.tar # optional
wget -O synset_labels.txt
https://raw.githubusercontent.com/tensorflow/models/master/research/slim/da
tasets/imagenet_2012_validation_synset_labels.txt
cd /path/to/Model-References/TensorFlow/computer_vision/Resnets
python3 preprocess_imagenet.py \
  --raw_data_dir=$IMAGENET_HOME \
  --local_scratch_dir=$IMAGENET_HOME/tf_records
mkdir -p $IMAGENET_HOME/val
mv $IMAGENET_HOME/validation/* $IMAGENET_HOME/val
cd $IMAGENET_HOME/val
wget -qO-
https://raw.githubusercontent.com/soumith/imagenetloader.torch/master/valpr
ep.sh | bash
```

Single-Server Habana Resnet Training Steps

Using LARS Optimizer

Using LARS optimizer usually requires changing the default values of some hyperparameters and should be manually set for resnet_ctl_imagenet_main.py. The recommended parameters together with their default values are presented below:

Parameter	Value
optimizer	LARS
base_learning_rate	2.5 or 9.5*
warmup_epochs	3
lr_schedule	polynomial
label_smoothing	0.1

Parameter	Value
weight_decay	0.0001
single_l2_loss_op	True

*2.5 is the default value for single card (1 HPU) trainings, otherwise, the default is 9.5. These values have been determined experimentally.

One HPU on 1 server, batch 256, 5 epochs, BF16 precision, LARS

Scaling Runs

```
time $PYTHON resnet_ctl_imagenet_main.py \
    -dt bf16 \
    --data_loader_image_type bf16 \
    -te 5 \
    -bs 256 \
    --optimizer LARS \
    --base_learning_rate 9.5 \
    --warmup_epochs 3 \
    --lr_schedule polynomial \
    --label_smoothing 0.1 \
    --weight_decay 0.0001 \
    --single_l2_loss_op \
    --data_dir /lambda_stor/data/imagenet/tf_records
```

Statistics:

```
real    67m37.184s
'avg_exp_per_second': 1594.6571942068174
```

Two HPUs on 1 server, batch 256, 5 epochs, BF16 precision, LARS

```
time mpirun \
    --allow-run-as-root \
    --bind-to core \
    -np 2 \
    --map-by socket:PE=7 \
    --merge-stderr-to-stdout \
       $PYTHON resnet_ctl_imagenet_main.py \
            --dtype bf16 \
            --data_loader_image_type bf16 \
            --use_horovod \
            -te 5 \
            -bs 256 \
            --optimizer LARS \
            --base_learning_rate 9.5 \
            --warmup_epochs 3 \
            --lr_schedule polynomial \
            --label_smoothing 0.1 \
            --weight_decay 0.0001 \
            --single_l2_loss_op \
            --data_dir /lambda_stor/data/imagenet/tf_records
```

Statistics:

```
real    36m48.760s
'avg_exp_per_second': 2990.6616862467968
```

Four HPUs on 1 server, batch 256, 5 epochs, BF16 precision, LARS

```
time mpirun \
    --allow-run-as-root \
    --bind-to core \
    -np 4 \
    --map-by socket:PE=7 \
    --merge-stderr-to-stdout \
       $PYTHON resnet_ctl_imagenet_main.py \
            --dtype bf16 \
            --data_loader_image_type bf16 \
            --use_horovod \
            -te 5 \
            -bs 256 \
            --optimizer LARS \
            --base_learning_rate 9.5 \
            --warmup_epochs 3 \
            --lr_schedule polynomial \
            --label_smoothing 0.1 \
            --weight_decay 0.0001 \
            --single_l2_loss_op \
            --data_dir /lambda_stor/data/imagenet/tf_records
```

Statistics:

```
real    19m34.556s
'avg_exp_per_second': 5708.747706392027
```

Eight HPUs on 1 server, batch 256, 5 epochs, BF16 precision, LARS

```
time mpirun \
    --allow-run-as-root \
    --bind-to core \
    -np 8 \
    --map-by socket:PE=7 \
    --merge-stderr-to-stdout \
       $PYTHON resnet_ctl_imagenet_main.py \
            --dtype bf16 \
            --data_loader_image_type bf16 \
            --use_horovod \
            -te 5 \
            -bs 256 \
            --optimizer LARS \
            --base_learning_rate 9.5 \
            --warmup_epochs 3 \
            --lr_schedule polynomial \
            --label_smoothing 0.1 \
            --weight_decay 0.0001 \
            --single_l2_loss_op \
            --data_dir /lambda_stor/data/imagenet/tf_records
```

Statistics:

```
real    11m11.386s
'avg_exp_per_second': 10484.725513908621
```

Multi-Server Habana Resnet Training Steps

One-time per user ssh key set up

On both Habana machines set up the ssh key

Habana-01

On **Habana-01**

```
mkdir ~/.ssh
cd ~/.ssh
ssh-keygen -t rsa -b 4096
#Accecpt default filename of id_rsa
#Enter passphrase (empty for no passphrase):
#Enter same passphrase again:
cat id_rsa.pub >> authorized_keys
```

```
ssh-keyscan -H 140.221.77.101 >> ~/.ssh/known_hosts
```

You should see:

```
# 140.221.77.101:22 SSH-2.0-OpenSSH_8.2p1 Ubuntu-4ubuntu0.5
# 140.221.77.101:22 SSH-2.0-OpenSSH_8.2p1 Ubuntu-4ubuntu0.5
# 140.221.77.101:22 SSH-2.0-OpenSSH_8.2p1 Ubuntu-4ubuntu0.5
# 140.221.77.101:22 SSH-2.0-OpenSSH_8.2p1 Ubuntu-4ubuntu0.5
# 140.221.77.101:22 SSH-2.0-OpenSSH_8.2p1 Ubuntu-4ubuntu0.5
```

```
ssh-keyscan -H 140.221.77.102 >> ~/.ssh/known_hosts
```

You should see:

```
# 140.221.77.102:22 SSH-2.0-OpenSSH_8.2p1 Ubuntu-4ubuntu0.5
# 140.221.77.102:22 SSH-2.0-OpenSSH_8.2p1 Ubuntu-4ubuntu0.5
# 140.221.77.102:22 SSH-2.0-OpenSSH_8.2p1 Ubuntu-4ubuntu0.5
# 140.221.77.102:22 SSH-2.0-OpenSSH_8.2p1 Ubuntu-4ubuntu0.5
# 140.221.77.102:22 SSH-2.0-OpenSSH_8.2p1 Ubuntu-4ubuntu0.5
```

Verify you can ssh from habana-01 to habana-02 without a password.

Verify vice-versa.

ResNet50

Ensure $MPI_ROOT is set

```
echo $MPI_ROOT
```

If not,

```
export MPI_ROOT=/usr/local/openmpi
```

Tensorflow

Test with:

```
cd /path/to/Model-
References/TensorFlow/computer_vision/Resnets/resnet_keras
```

Verify a connection with

```
mpirun -v --allow-run-as-root  --mca btl_tcp_if_include enp75s0f0,ens1f0 --
tag-output --merge-stderr-to-stdout --prefix /usr/local/openmpi --host
140.221.77.101:8,140.221.77.102:8 hostname
```

Almost immediately, you should see

```
[1,0]<stdout>:habana-01
[1,1]<stdout>:habana-01
[1,2]<stdout>:habana-01
[1,3]<stdout>:habana-01
[1,4]<stdout>:habana-01
[1,5]<stdout>:habana-01
[1,6]<stdout>:habana-01
[1,7]<stdout>:habana-01
[1,8]<stdout>:habana-02
[1,9]<stdout>:habana-02
[1,10]<stdout>:habana-02
[1,11]<stdout>:habana-02
[1,12]<stdout>:habana-02
[1,13]<stdout>:habana-02
[1,14]<stdout>:habana-02
[1,15]<stdout>:habana-02
```

If the above command hangs, there is a communication problem.

Sixteen HPUs on 2 servers, batch 256, 5 epochs, BF16 precision, LARS

```
mpirun -v \
    --allow-run-as-root \
    -np 16 \
    --mca btl_tcp_if_include 192.168.201.0/24 \
    --tag-output \
    --merge-stderr-to-stdout \
    --prefix /usr/local/openmpi \
    -H 140.221.77.101:8,140.221.77.102:8 \
    -x GC_KERNEL_PATH \
    -x HABANA_LOGS \
    -x PYTHONPATH \
    -x HCCL_SOCKET_IFNAME=enp75s0f0,ens1f0 \
    -x HCCL_OVER_TCP=1 \
    $PYTHON resnet_ctl_imagenet_main.py \
        -dt bf16 \
        -dlit bf16 \
        -bs 256 \
        -te 5 \
        --use_horovod \
        --data_dir /lambda_stor/data/imagenet/tf_records/ \
        --optimizer LARS \
        --base_learning_rate 9.5 \
        --warmup_epochs 3 \
        --lr_schedule polynomial \
        --label_smoothing 0.1 \
        --weight_decay 0.0001 \
        --single_l2_loss_op \
        --model_dir=`mktemp -d`
```

Statistics:

~~~
2022-09-27 18:23:10.461103
2022-09-27 18:31:34.638972
real    8:24
'avg_exp_per_second': 15481.107839604083

### PyTorch

This section has not been tested.

```bash
cd Model-References/PyTorch/computer_vision/classification/torchvision/
export MASTER_PORT=12355
export MASTER_ADDR=192.168.201.101
mpirun --allow-run-as-root --mca --bind-to core --map-by ppr:4:socket:PE=7
-np 16 --mca btl_tcp_if_include 192.168.201.0/24 --merge-stderr-to-stdout -
-prefix $MPI_ROOT -H 140.221.77.101:8,140.221.77.102:8 -x GC_KERNEL_PATH -x
PYTHONPATH -x MASTER_ADDR -x MASTER_PORT -x
HCCL_SOCKET_IFNAME=enp75s0f0,ens1f0 -x HCCL_OVER_TCP=1 $PYTHON -u train.py
--batch-size=256 --model=resnet50 --device=hpu --workers=8 --print-freq=1 -
-deterministic --data-path=/lambda_stor/habana/data/tensorflow/imagenet --
epochs=2 --hmp --hmp-bf16 ./ops_bf16_Resnet.txt --hmp-fp32
./ops_fp32_Resnet.txt --custom-lr-values 0.475
~~~