ResNet-50 Training

Kim-Khanh Tran and Ling Cheng

ResNet-50: Metrics and State of the Art Performance

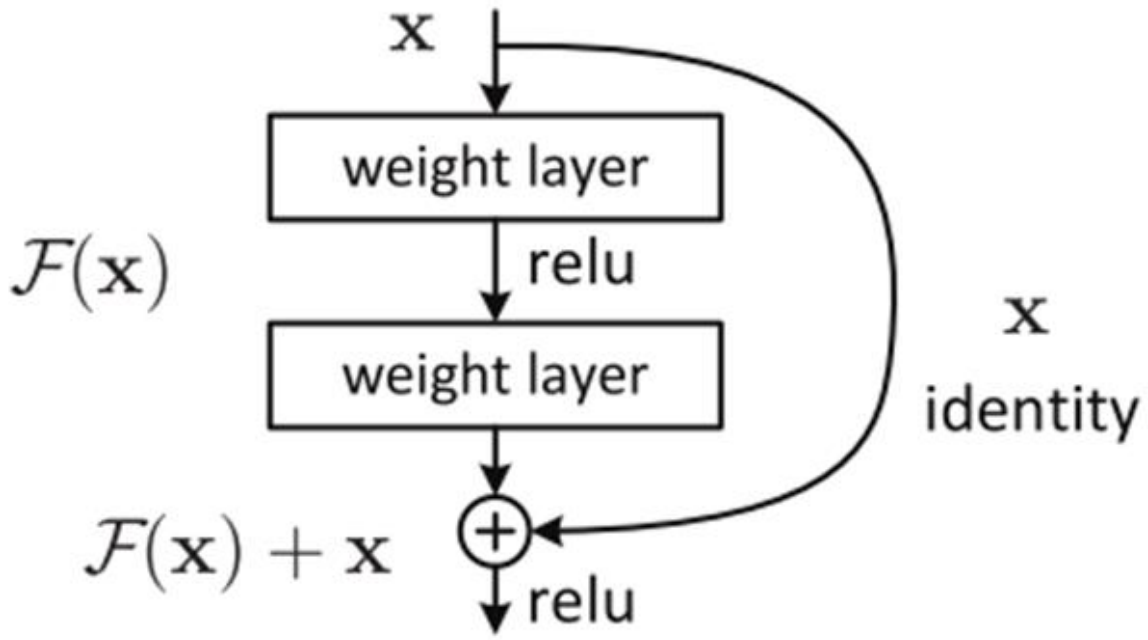
- **ResNet-50** is variation of ResNet which is a convolutional neural network. ResNet is popular for image classification. The main feature of ResNet is the skip connection to address the vanishing gradients issue as the model backpropagates. The “50” is the number of layers in the network.
- Reference paper:

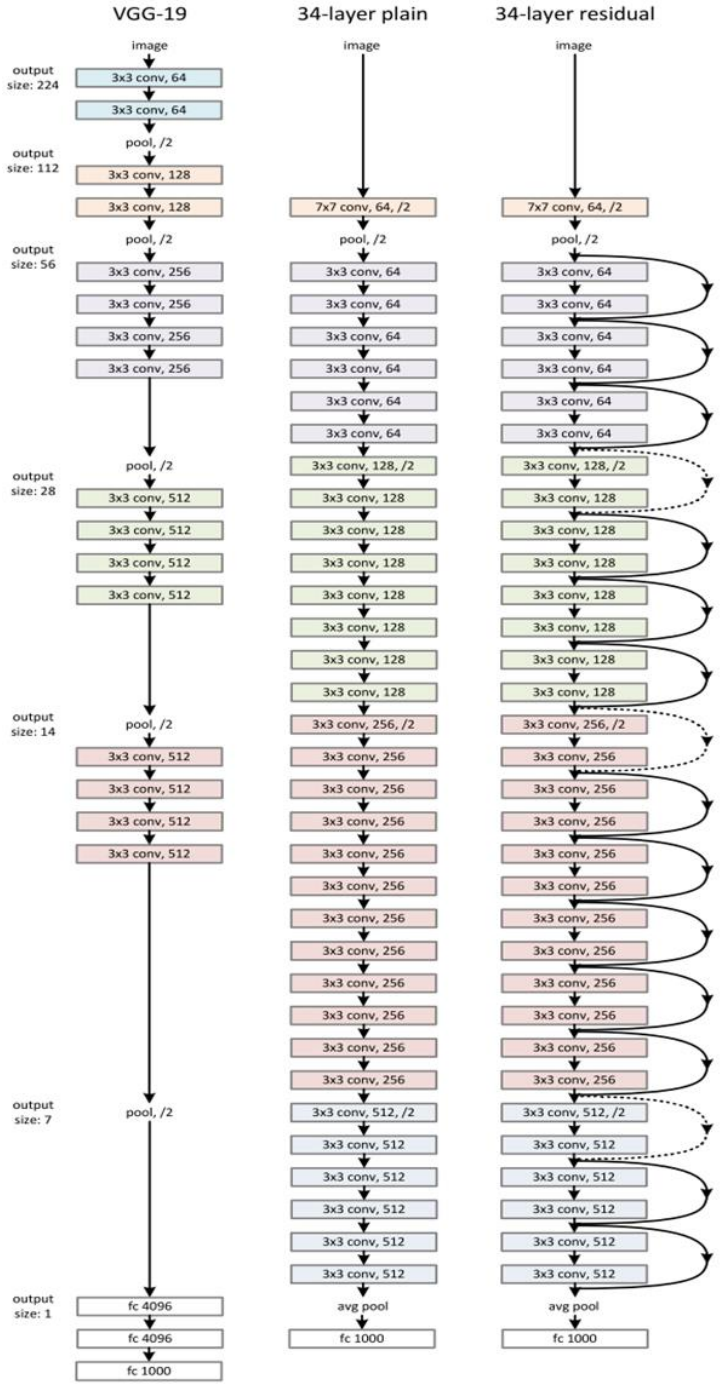
  https://arxiv.org/abs/1611.05431
- Source:

  https://github.com/horovod/horovod/tree/master/examples

	ResNet-50
Xie [43]	76.1
Mixup [45]	76.7
LabelRefinery [2]	76.5
Autoaugment [7]	77.6
Weakly supervised [27]	78.2

ResNet-50 Architecture





Experiment Setup

- ImageNet dataset
- Horovod implementation
  - https://github.com/horovod/horovod/blob/master/examples/pytorch/pytorch_imagenet_resnet50.py
- Parameters
  - Weight decay: 0.0005
  - Initial learning rate: 0.0125
  - Epochs: 62
  - Batch size: 32
  - Nodes: 1, 2, 3, 8, 16, 32, 64, 128 (6, 12, 24, 48, 96, 129, 384, 768 gpus)

run-resnet.sh

```
# Parameters for Slurm
#!/bin/bash -x
#SBATCH -J pytorch_resnet  # job name
#SBATCH -o pytorch_resnet_%j.out  # output file
#SBATCH -e pytorch_resnet_%j.err  # error file
#SBATCH --gres=gpu:6  # gpus per node
#SBATCH --nodes=16 #  number  of  nodes
#SBATCH --time=06:00:00  # time limit (max 6hr)

# Parameters for ResNet
codedir=~/scratch/horovod/examples
codepath=$codedir/pytorch_imagenet_resnet50.py
traindir=/gpfs/u/locker/200/CADS/datasets/ImageNet/train
valdir=/gpfs/u/locker/200/CADS/datasets/ImageNet/val
logdir=~/scratch/horovod/examples/logs
hostdir=~/scratch/horovod/examples/hosts
epochs=62
Batchsize=32

# Calculate the number of tasks
NP=`expr $SLURM_JOB_NUM_NODES \*
$SLURM_NTASKS_PER_NODE`

# Activate the correct conda environment
. ~/scratch/miniconda3/etc/profile.d/conda.sh
conda activate wmlce-1.7.0

# save the hostname of allocated nodes for distributed run
srun hostname -s | sort -u > $hostdir/hosts.$SLURM_JOBID
awk "{ print \$0 \"-ib\"; }" $hostdir/hosts.$SLURM_JOBID
>$hostdir/tmp.$SLURM_JOBID
mv $hostdir/tmp.$SLURM_JOBID $hostdir/hosts.$SLURM_JOBID

# run the code with its parameters

horovodrun --verbose -np $NP   -hostfile $hostdir/hosts.$SLURM_JOBID
python $codepath --train-dir $traindir --val-dir $valdir --log-dir $logdir --
fp16-allreduce --batch-size=$batchsize --epochs=$epochs

# Clean up the generated host list
rm  $hostdir/hosts.$SLURM_JOBID
```

Running ResNet-50 on AiMOS

Setup your environment

1. Set up passwordless: https://ibm-ai-hardware-center.github.io/AiMOS/aimos-env-basics.html#set-up-passwordless
2. Set up your proxy: https://ibm-ai-hardware-center.github.io/AiMOS/aimos-env-basics.html#set-up-proxy
3. Install conda: https://ibm-ai-hardware-center.github.io/AiMOS/aimos-workload.html#install-anaconda
4. Install WML-CE (PowerAI): https://ibm-ai-hardware-center.github.io/AiMOS/aimos-workload.html#install-wml-ce-a-k-a-powerai

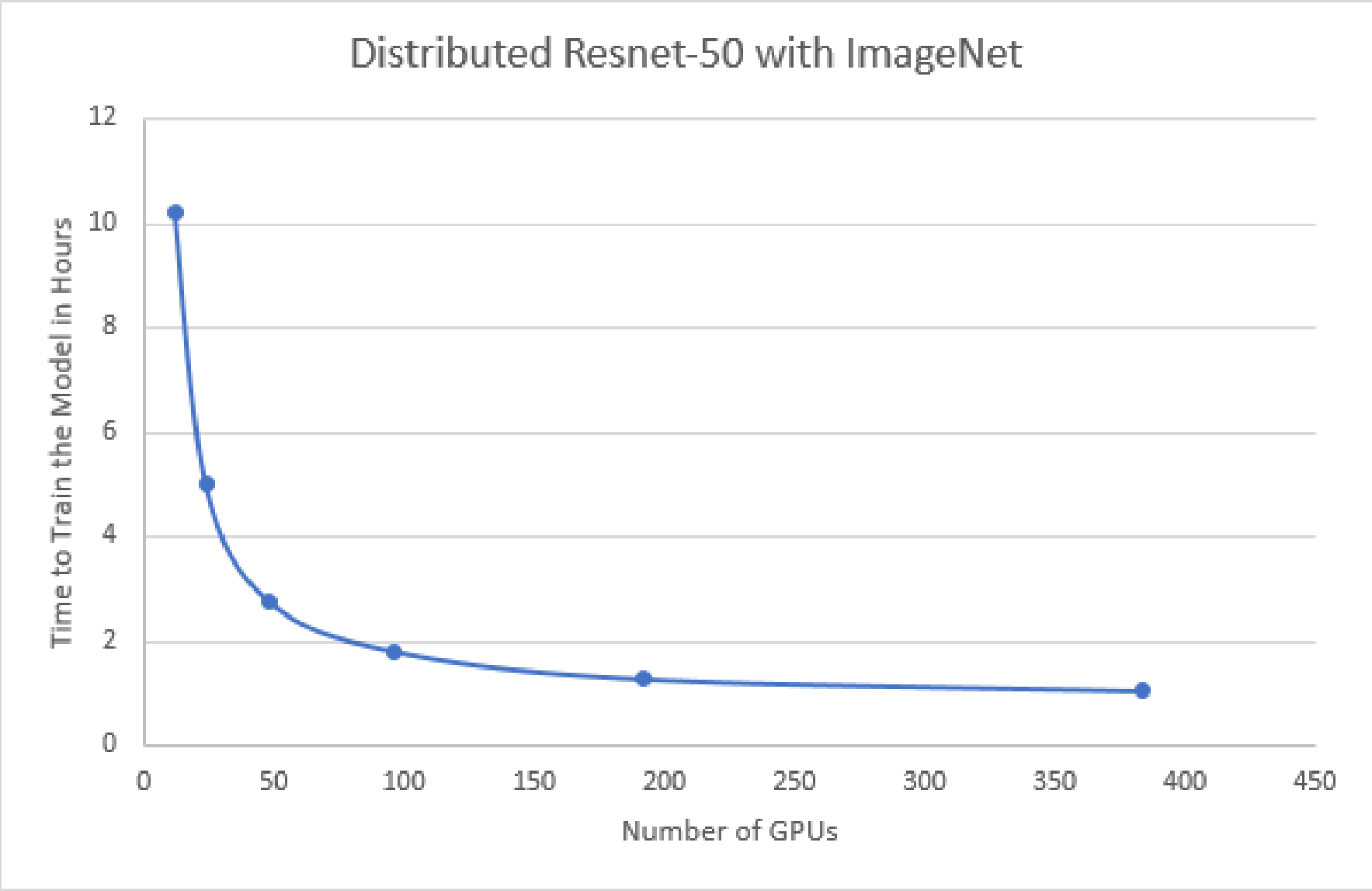
Setup and Run ResNet-50

1. Copy ResNet files into your scratch directory
   ```
   cp –r /gpfs/u/locker/201/CABS/resnet-50 ~/scratch
   ```
   Note: If you don't have permissions to this directory, contact Lorraine Herger (herger@us.ibm.com).
2. Go into the directory you just copied and add executable permission
   ```
   cd ~/scratch/resnet-50
   chmod +x *
   ```
3. Make sure you have access to the ImageNet dataset in the storage locker
   ```
   ls /gpfs/u/locker/200/CADS/datasets/ImageNet
   ```
   You should see a directory named `train` and `val`.
   Note: If you don't have permissions to this directory, contact Lorraine Herger (herger@us.ibm.com).
4. Edit `run-resnet.sh` to set the parameters
5. Start the job with
   ```
   sbatch run-resnet.sh
   ```

Sample Job Output

```
++ expr 2 '*' 6
+ NP=12
+ echo 12
+ srun hostname -s
+ sort -u
+ awk '{ print $0 "-ib slots=6"; }'
/gpfs/u/home/BMHR/BMHRkmkh/scratch/horovod/examples/hosts/hosts.184973
+ mv /gpfs/u/home/BMHR/BMHRkmkh/scratch/horovod/examples/hosts/tmp.184973
/gpfs/u/home/BMHR/BMHRkmkh/scratch/horovod/examples/hosts/hosts.184973
+ horovodrun --verbose -np 12 -hostfile
/gpfs/u/home/BMHR/BMHRkmkh/scratch/horovod/examples/hosts/hosts.184973 python
/gpfs/u/home/BMHR/BMHRkmkh/scratch/horovod/examples/resnet50-lms.py --train-dir
/gpfs/u/locker/200/CADS/datasets/ImageNet/train --val-dir
/gpfs/u/locker/200/CADS/datasets/ImageNet/val --log-dir
/gpfs/u/home/BMHR/BMHRkmkh/scratch/horovod/ examples /logs --fp16-allreduce --batch-size=32 --
epochs=10
```

Performance of Distributed Training of ResNet in AiMOS



How to Enable and Tune LMS in Pytorch

- When a limit is defined via *torch.cuda.set_limit_lms*, the LMS algorithm favors allocation of GPU memory up to the limit prior to swapping any tensors out to host memory. This allows the user to control the amount of GPU memory consumed when using LMS.

- Tuning this option to optimize GPU memory utilization can reduce data transfers and improve performance.

- The ideal tuning for any given scenario may differ, so it is best practice to determine the value of limit_lms empirically

```
# Enable LMS
torch.cuda.set_enabled_lms(True)

# limit = 15G
torch.cuda.set_limit_lms(16106119113)

# limit = 20G
torch.cuda.set_limit_lms(21474836480)
```

Sample Job Output Without LMS enabled

[1,7]<stderr>:RuntimeError: CUDA out of memory. Tried to allocate 1.15 GiB (GPU 1; 31.75 GiB total capacity; 30.01 GiB already allocated; 484.50 MiB free; 13.82 MiB cached; 0 bytes inactive)

Primary job terminated normally, but 1 process returned
a non-zero exit code. Per user-direction, the job has been aborted.

---

mpirun detected that one or more processes exited with non-zero status, thus causing
the job to be terminated. The first process to do so was:

  Process name: [[733,1],4]
  Exit code: 1

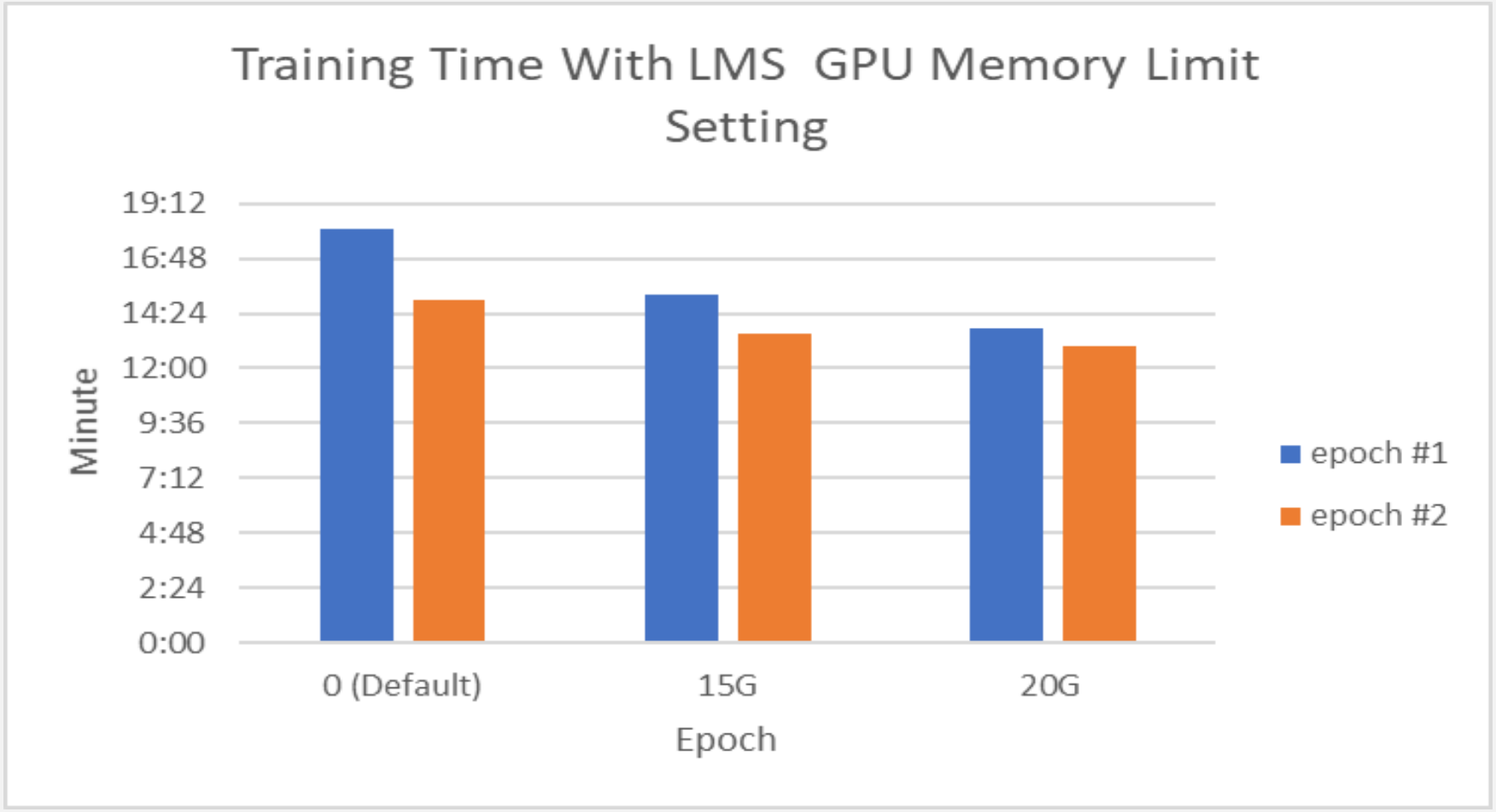
Unable to run the training job because the required memory is not available in existing GPUs

Enable LMS

You now can run the training without the error.

Enabling LMS allows training to run without error and further improves the training time.

```
Train Epoch     #1: 100%|████████████████| 140/140 [18:06<00:00,  7.76s/it, loss=6.78,
accuracy=0.475][1,0]<stderr>:
Validate Epoch  #1: 100%|████████████████| 131/131 [00:44<00:00,  2.98it/s, loss=6.58,
accuracy=0.64][1,0]<stderr>::
Train Epoch     #2: 100%|████████████████| 140/140 [14:58<00:00,  6.42s/it, loss=6.12,
accuracy=2.55][1,0]<stderr>:
Validate Epoch  #2: 100%|████████████████| 131/131 [00:39<00:00,  3.33it/s, loss=7.34,
accuracy=1.06][1,0]<stderr>:
Train Epoch     #3:  25%|███       | 35/140 [03:46<09:49,  5.61s/it, loss=5.82, accuracy=3.95]
slurmstepd: error: *** JOB 184629 ON dcs054 CANCELLED AT 2020-05-21T16:19:19 DUE TO TIME LIMIT ***
```



You can control how GPU memory used for tensors under LMS. Tuning this limit to optimize GPU memory utilization, therefore, can reduce data transfers and improve performance.