Getting started with WMLCE (former PowerAI)

WMLCE has reached End-Of-Life and is now out of date.

See Getting started with Open Cognitive Environment (OpenCE, former WMLCE) for the latest software stack.

- IBM Watson Machine Learning Community Edition (WMLCE-1.7.0, WMLCE-1.6.2)
- Simple Example with Caffe
  - Interactive mode
  - Batch mode
- Simple Example with Caffe2
  - Interactive mode
  - Batch mode
- Simple Example with TensorFlow
  - Interactive mode
  - Batch mode
  - Visualization with TensorBoard
    - Interactive mode
    - Batch mode
    - Start the TensorBorad session
- Simple Example with Pytorch
  - Interactive mode
  - Batch mode
- Major Installed PowerAI Related Anaconda Modules

IBM Watson Machine Learning Community Edition (WMLCE-1.7.0, WMLCE-1.6.2)

WMLCE is an enterprise software distribution that combines popular open-source deep learning frameworks, efficient AI development tools, and accelerated IBM Power Systems servers. It includes the following frameworks:

Framework	Version	Description
Caffe	1.0	Caffe is a deep learning framework made with expression, speed, and modularity in mind. It is developed by Berkeley AI Research and by community contributors.
Pytorch	1.3.1	Pytorch is an open-source deep learning platform that provides a seamless path from research prototyping to production deployment. It is developed by Facebook and by community contributors.
TensorFlow	2.1.0	TensorFlow is an end-to-end open-source platform for machine learning. It is developed by Google and by community contributors.

For complete WMLCE documentation, see https://developer.ibm.com/linuxonpower/deep-learning-powerai/releases/. Here we only show simple examples with system-specific instructions.

Simple Example with Caffe

Interactive mode

Get one compute node for interactive use:

```
swrun -p gpux1
```

Once on the compute node, load PowerAI module using one of these:

```
module load wmlce/1.6.2
module load wmlce/1.7.0
```

Install samples for Caffe:

```
caffe-install-samples ~/caffe-samples
cd ~/caffe-samples
```

Download data for MNIST model:

```
./data/mnist/get_mnist.sh
```

Convert data and create MNIST model:

```
./examples/mnist/create_mnist.sh
```

Train LeNet on MNIST:

```
./examples/mnist/train_lenet.sh
```

Batch mode

The same can be accomplished in batch mode using the following caffe_sample.swb script:

```
wget https://wiki.ncsa.illinois.edu/download/attachments/82510352/caffe_sample.swb
swbatch caffe_sample.swb
squeue
```

Simple Example with Caffe2

Interactive mode

Get a node for interactive use:

```
swrun -p gpux1
```

Once on the compute node, load PowerAI module using one of these:

```
module load wmlce/1.6.2
module load wmlce/1.7.0
```

Install samples for Caffe2:

```
caffe2-install-samples ~/caffe2-samples
cd ~/caffe2-samples
```

Download data with LMDB:

```
python ./examples/lmdb_create_example.py --output_file lmdb
```

Train ResNet50 with Caffe2:

```
python ./examples/resnet50_trainer.py --train_data ./lmdb
```

Batch mode

The same can be accomplished in batch mode using the following caffe2_sample.swb script:

```
wget https://wiki.ncsa.illinois.edu/download/attachments/82510352/caffe2_sample.swb
sbatch caffe2_sample.swb
squeue
```

Simple Example with TensorFlow

Interactive mode

Get a node for interactive use:

```
swrun -p gpux1
```

Once on the compute node, load PowerAI module using one of these:

```
module load wmlce/1.6.2
module load wmlce/1.7.0
```

Copy the following code into file "mnist-demo.py":

```
import tensorflow as tf
mnist = tf.keras.datasets.mnist

(x_train, y_train),(x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(512, activation=tf.nn.relu),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

Train on MNIST with keras API:

```
python ./mnist-demo.py
```

Batch mode

The same can be accomplished in batch mode using the following tf_sample.swb script:

```
wget https://wiki.ncsa.illinois.edu/download/attachments/82510352/tf_sample.swb
sbatch tf_sample.swb
squeue
```

Visualization with TensorBoard

Interactive mode

Get a node for interactive use:

```
swrun -p gpux1
```

Once on the compute node, load PowerAI module using one of these:

```
module load wmlce/1.6.2
module load wmlce/1.7.0
```

Download the code mnist-with-summaries.py to $HOME folder:

```
cd ~
wget https://wiki.ncsa.illinois.edu/download/attachments/82510352/mnist-with-summaries.py
```

Train on MNIST with TensorFlow summary:

```
python ./mnist-with-summaries.py
```

Batch mode

The same can be accomplished in batch mode using the following tfbd_sample.swb script:

```
wget https://wiki.ncsa.illinois.edu/download/attachments/82510352/tfbd_sample.swb
sbatch tfbd_sample.swb
squeue
```

Start the TensorBorad session

After job completed the TensorFlow log files can be found in "~/tensorflow/mnist/logs", start the TensorBoard server on hal-ondemand, detail refers Getting started with HAL OnDemand.

Simple Example with Pytorch

Interactive mode

Get a node for interactive use:

```
swrun -p gpux1
```

Once on the compute node, load PowerAI module using one of these:

```
module load wmlce/1.6.2
module load wmlce/1.7.0
```

Install samples for Pytorch:

```
pytorch-install-samples ~/pytorch-samples
cd ~/pytorch-samples
```

Train on MNIST with Pytorch:

```
python ./examples/mnist/main.py
```

Batch mode

The same can be accomplished in batch mode using the following pytorch_sample.swb script:

```
wget https://wiki.ncsa.illinois.edu/download/attachments/82510352/pytorch_sample.swb
sbatch pytorch_sample.swb
squeue
```

Major Installed PowerAI Related Anaconda Modules

Name	Version	Description

caffe	1.0	Caffe is a deep learning framework made with expression, speed, and modularity in mind.
cudatoolkit	10.2.89	The NVIDIA® CUDA® Toolkit provides a development environment for creating high-performance GPU-accelerated applications.
cudnn	7.6.5+10.2	The NVIDIA CUDA® Deep Neural Network library (cuDNN) is a GPU-accelerated library of primitives for deep neural networks.
nccl	2.5.6	The NVIDIA Collective Communications Library (NCCL) implements multi-GPU and multi-node collective communication primitives that are performance-optimized for NVIDIA GPUs.
opencv	3.4.8	OpenCV was designed for computational efficiency and with a strong focus on real-time applications.
pytorch	1.3.1	PyTorch enables fast, flexible experimentation and efficient production through a hybrid front-end, distributed training, and ecosystem of tools and libraries.
tensorboard	2.1.0	To make it easier to understand, debug, and optimize TensorFlow programs, we've included a suite of visualization tools called TensorBoard.
tensorflow-gpu	2.1.0	The core open-source library to help you develop and train ML models.