Example Programs

SambaNova provides examples of some well-known AI applications under the path: `/software/sambanova/apps/1.10.3-11/starters`, on the SambaNova compute node sm-01. Make a copy of this to your home directory:

```
cd ~/
mkdir apps
cp -r /software/sambanova/apps/1.10.3-11/starters apps/starters
```

LeNet

Change directory

```
cd ~/apps/starters/
```

Common Arguments

Below are some of the common arguments used across most of the models in the example code.

Argument	Default	Help
-b	1	Batch size for training
-n, --num-iterations	100	Number of iterations to run the pef for
-e, --num-epochs	1	Number epochs for training
--log-path	'check points'	Log path
--num-workers	0	Number of workers
--measure-train-performance	None	Measure training performance

LeNet Arguments

Argument	Default	Help
--lr	0.01	Learning rate for training
--momentum	0.0	Momentum value for training
--weight-decay	0.01	Weight decay for training
--data-path	'./data'	Data path
--data-folder	'mnist_data'	Folder containing mnist data

Run these commands:

```
srun python lenet.py compile -b=1 --pef-name="lenet" --output-folder="pef"
srun python lenet.py test --pef="pef/lenet/lenet.pef"
srun python lenet.py run --pef="pef/lenet/lenet.pef"
```

To use Slurm sbatch, create submit-lenet-job.sh with the following contents:

```
#!/bin/sh

python lenet.py compile -b=1 --pef-name="lenet" --output-folder="pef"
python lenet.py test --pef="pef/lenet/lenet.pef"
python lenet.py run --pef="pef/lenet/lenet.pef"
```

Then

```
sbatch --output=pef/lenet/output.log submit-lenet-job.sh
```

Squeue will give you the queue status.

```
squeue
```

The output file, pef/lenet/output.log, will look something like this:

```
[Info][SAMBA][Default] # Placing log files in
pef/lenet/lenet.samba.log

[Info][MAC][Default] # Placing log files in
```

```
pef/lenet/lenet.mac.log

[Warning][SAMBA][Default] #

--------------------------------------------------

Using patched version of torch.cat and torch.stack

--------------------------------------------------

[Warning][SAMBA][Default] # The dtype of "targets" to
CrossEntropyLoss is torch.int64, however only int16 is currently
supported, implicit conversion will happen

[Warning][MAC][GraphLoweringPass] # lenet__reshape skip
set_loop_to_air

[Warning][MAC][GraphLoweringPass] # lenet__reshape_bwd skip
set_loop_to_air

...

Epoch [1/1], Step [59994/60000], Loss: 0.1712

Epoch [1/1], Step [59995/60000], Loss: 0.1712

Epoch [1/1], Step [59996/60000], Loss: 0.1712

Epoch [1/1], Step [59997/60000], Loss: 0.1712

Epoch [1/1], Step [59998/60000], Loss: 0.1712

Epoch [1/1], Step [59999/60000], Loss: 0.1712

Epoch [1/1], Step [60000/60000], Loss: 0.1712

Test Accuracy: 98.06 Loss: 0.0628

2021-6-10 10:52:28 : [INFO][SC][53607]: SambaConnector: PEF File:
pef/lenet/lenet.pef

Log ID initialized to: [ALCFUserID][python][53607] at
/var/log/sambaflow/runtime/sn.log
```

MNIST - Feed Forward Network

Change directory (if necessary)

```
cd ~/apps/starters/
```

commands to run MNIST example:

```
srun python ffn_mnist.py compile --pef-name="ffn_mnist" --output-
folder="pef"
srun python ffn_mnist.py test --pef="pef/ffn_mnist/ffn_mnist.pef"
srun python ffn_mnist.py run --pef="pef/ffn_mnist/ffn_mnist.pef" --data-
path mnist_data
```

To the run the same using Slurm sbatch, create and run the submit-ffn_mnist-job.sh with the following contents.

```
#!/bin/sh
srun python ffn_mnist.py compile --pef-name="ffn_mnist" --output-
folder="pef"
srun python ffn_mnist.py test --pef="pef/ffn_mnist/ffn_mnist.pef"
srun python ffn_mnist.py run --pef="pef/ffn_mnist/ffn_mnist.pef" --data-
path mnist_data
```

```
sbatch --output=pef/ffn_mnist/output.log submit-ffn_mnist-job.sh
```

Logistic Regression

Change directory (if necessary)

```
cd ~/apps/starters/
```

Logistic Regression Arguments

This is not an exhaustive list of arguments.

Arguments

Argument	**Default**	**Help**	**Step**
--lr	0.001	Learning rate for training	Compile
--momentum	0.0	Momentum value for training	Compile
--weight-decay	1e-4	Weight decay for training	Compile
--num-features	784	Number features for training	Compile
--num-classes	10	Number classes for training	Compile

Argument	Default	Help	Step
--weight-norm	na	Enable weight normalization	Compile

Run these commands:

```
srun python logreg.py compile --pef-name="logreg" --output-folder="pef"
srun python logreg.py test --pef="pef/logreg/logreg.pef"
srun python logreg.py run --pef="pef/logreg/logreg.pef"
```

To use Slurm, create submit-logreg-job.sh with the following contents:

```
#!/bin/sh

python logreg.py compile --pef-name="logreg" --output-folder="pef"
python logreg.py test --pef="pef/logreg/logreg.pef"
python logreg.py run --pef="pef/logreg/logreg.pef"
```

Then

```
sbatch --output=pef/logreg/output.log submit-logreg-job.sh
```

The output file, pef/logreg/output.log, will look something like:

```
pef/logreg/output.log

[Info][SAMBA][Default] # Placing log files in
pef/logreg/logreg.samba.log
[Info][MAC][Default] # Placing log files in
pef/logreg/logreg.mac.log
[Warning][SAMBA][Default] #
--------------------------------------------------
Using patched version of torch.cat and torch.stack
--------------------------------------------------

[Warning][SAMBA][Default] # The dtype of "targets" to
CrossEntropyLoss is torch.int64, however only int16 is currently
supported, implicit conversion will happen
[Warning][MAC][MemoryOpTransformPass] # Backward graph is trimmed
according to requires_grad to save computation.
[Warning][MAC][WeightShareNodeMergePass] # Backward graph is
trimmed according to requires_grad to save computation.
[Warning][MAC][ReduceCatFaninPass] # Backward graph is trimmed
according to requires_grad to save computation.
[info ] [PLASMA] Launching plasma compilation! See log file:
```

```
/home/ALCFUserID/apps/starters/pytorch/pef/logreg//logreg.plasma_compile.lo
g
...

[Warning][SAMBA][Default] # The dtype of "targets" to
CrossEntropyLoss is torch.int64, however only int16 is currently
supported, implicit conversion will happen
Epoch [1/1], Step [10000/60000], Loss: 0.4763
Epoch [1/1], Step [20000/60000], Loss: 0.4185
Epoch [1/1], Step [30000/60000], Loss: 0.3888
Epoch [1/1], Step [40000/60000], Loss: 0.3721
Epoch [1/1], Step [50000/60000], Loss: 0.3590
Epoch [1/1], Step [60000/60000], Loss: 0.3524
Test Accuracy: 90.07 Loss: 0.3361
2021-6-11 8:38:49 : [INFO][SC][99185]: SambaConnector: PEF File:
pef/logreg/logreg.pef
Log ID initialized to: [ALCFUserID][python][99185] at
/var/log/sambaflow/runtime/sn.log
```

UNet

Change directory

```
cp -r /software/sambanova/apps/1.10.3-11/image apps/image
cd ~/apps/image/pytorch/unet
export OUTDIR=~/apps/image/pytorch/unet
export DATADIR=/software/sambanova/dataset/kaggle_3m
```

Export the path to the dataset which is required for the training.

Run these commands for training (compile + train):

```
srun python unet.py compile --in-channels=3 --in-width=32 --in-height=32 --
init-features 32 --batch-size 1 --pef-name="unet_train" --output-
folder=${OUTDIR}
srun python unet.py run --do-train --in-channels=3 --in-width=32 --in-
height=32 --init-features 32 --batch-size 1 --data-dir $DATADIR --log-dir
${OUTDIR} --epochs 5 --pef=${OUTDIR}/unet_train/unet_train.pef
```

Using Slurm: To use Slurm, create submit-unet-job.sh with the following contents:

```
#!/bin/sh
export OUTDIR=~/apps/image/pytorch/unet
export DATADIR=/software/sambanova/dataset/kaggle_3m
python unet.py compile --in-channels=3 --in-width=32 --in-height=32 --init-
features 32 --batch-size 1 --pef-name="unet_train" --output-
folder=${OUTDIR}
```

```
python unet.py run --do-train  --in-channels=3  --in-width=32  --in-
height=32 --init-features 32 --batch-size=1 --data-dir $DATADIR --log-dir
${OUTDIR}/log_dir_unet32_train --epochs 5 --
pef=${OUTDIR}/unet_train/unet_train.pef
```

Then

```
sbatch submit-unet-job.sh
```