# Table of Contents

Table of Contents ................................................................. i
0.1 About This Manual ........................................................... iii
0.2 About The Manuals In General ........................................... iii
0.3 Getting Administrator-Level Support .................................... iv
0.4 Getting Professional Services ............................................ iv

1 Introduction And Installing the Machine Learning RPMs 1
1.1 Introduction ................................................................. 1
1.2 Packages Available ........................................................ 1
1.3 Requirements ............................................................... 2
  1.3.1 Software Installation .................................................. 2
  1.3.2 Module Installation .................................................... 3
  1.3.3 Further reading ....................................................... 4

2 Running Caffe 5
2.1 Downloading The MNIST Data .......................................... 5
2.2 Setting The Default Matplotlib Backend ............................... 5
2.3 Creating A Working Directory ............................................ 6
2.4 Using The Web App ........................................................ 6
2.5 Logging In ................................................................. 7
2.6 Creating Training And Validation Datasets ............................ 8
2.7 Training A Model .......................................................... 14
2.8 Testing A Model ........................................................... 17

3 Running TensorFlow 21
3.1 Hello World ............................................................... 21
3.2 Training A Convolutional Neural Network ............................. 21
3.3 Image Recognition ........................................................ 22
Preface

Welcome to the *Machine Learning Manual* for Bright Cluster Manager 7.3.

**0.1 About This Manual**

This manual is aimed at helping cluster administrators install, understand, configure, and manage basic machine learning capabilities easily using Bright Cluster Manager. The administrator is expected to be reasonably familiar with the *Administrator Manual*.

**0.2 About The Manuals In General**

Regularly updated versions of the Bright Cluster Manager 7.3 manuals are available on updated clusters by default at /cm/shared/docs/cm. The latest updates are always online at [http://support.brightcomputing.com/manuals](http://support.brightcomputing.com/manuals).

- The *Installation Manual* describes installation procedures for a basic cluster.
- The *Administrator Manual* describes the general management of the cluster.
- The *User Manual* describes the user environment and how to submit jobs for the end user.
- The *Cloudbursting Manual* describes how to deploy the cloud capabilities of the cluster.
- The *Developer Manual* has useful information for developers who would like to program with Bright Cluster Manager.
- The *OpenStack Deployment Manual* describes how to deploy OpenStack with Bright Cluster Manager.
- The *Big Data Deployment Manual* describes how to deploy Big Data with Bright Cluster Manager.
- The *UCS Deployment Manual* describes how to deploy the Cisco UCS server with Bright Cluster Manager.
- The *Machine Learning Manual*—this manual—describes how to install and configure machine learning capabilities with Bright Cluster Manager.

If the manuals are downloaded and kept in one local directory, then in most pdf viewers, clicking on a cross-reference in one manual that refers to a section in another manual opens and displays that section in the second manual. Navigating back and forth between documents is usually possible with keystrokes or mouse clicks.

For example: &lt;Alt&gt;&lt;Backarrow&gt; in Acrobat Reader, or clicking on the bottom leftmost navigation button of xpdf, both navigate back to the previous document.

The manuals constantly evolve to keep up with the development of the Bright Cluster Manager environment and the addition of new hardware and/or applications. The manuals also regularly incorporate customer feedback. Administrator and user input is greatly valued at Bright Computing. So any comments, suggestions or corrections will be very gratefully accepted at manuals@brightcomputing.com.
0.3 Getting Administrator-Level Support

If the reseller from whom Bright Cluster Manager was bought offers direct support, then the reseller should be contacted.

Otherwise the primary means of support is via the website https://support.brightcomputing.com. This allows the administrator to submit a support request via a web form, and opens up a trouble ticket. It is a good idea to try to use a clear subject header, since that is used as part of a reference tag as the ticket progresses. Also helpful is a good description of the issue. The followup communication for this ticket goes via standard e-mail. Section 11.2 of the Administrator Manual has more details on working with support.

0.4 Getting Professional Services

Bright Computing normally differentiates between professional services (customer asks Bright Computing to do something or asks Bright Computing to provide some service) and support (customer has a question or problem that requires an answer or resolution). Professional services can be provided after consulting with the reseller, or the Bright account manager.
Introduction And Installing the Machine Learning RPMs

1.1 Introduction
As of Bright Cluster Manager 7.3 a number of Machine/Deep Learning library and framework RPM packages have been included. Bright makes it faster and easier for organizations to gain actionable insights from rich, complex data using the latest, state-of-the-art libraries with minimal effort during installation.

1.2 Packages Available
Currently the following RPMs are available:

<table>
<thead>
<tr>
<th>Package name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cm-ml-distdeps</td>
<td>Meta-package containing library dependencies available from the base Linux distribution repositories as well as the EPEL repository</td>
</tr>
<tr>
<td>cm-ml-pythondeps</td>
<td>Pre-packaged Python dependencies for Bright’s RPM packages</td>
</tr>
<tr>
<td>caffe</td>
<td>A deep learning framework made with expression, speed, and modularity in mind.</td>
</tr>
<tr>
<td>torch7</td>
<td>A library in C++ for developing Open Source speech and machine learning applications.</td>
</tr>
<tr>
<td>theano</td>
<td>Theano is a Python library that allows you to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently.</td>
</tr>
</tbody>
</table>

...continues
Introduction And Installing the Machine Learning RPMs

<table>
<thead>
<tr>
<th>Package name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tensorflow</td>
<td>TensorFlow is an Open Source Software Library for Machine Intelligence</td>
</tr>
<tr>
<td>cub</td>
<td>Reusable software components for CUDA</td>
</tr>
<tr>
<td>nccl</td>
<td>(pronounced &quot;Nickel&quot;) A standalone library of standard collective communication routines, such as all-gather, reduce, broadcast, etc., that have been optimized to achieve high bandwidth over PCIe</td>
</tr>
<tr>
<td>mlpython</td>
<td>MLPython is a library for organizing machine learning research.</td>
</tr>
<tr>
<td>DIGITS</td>
<td>A web frontend to Caffe and Torch, developed by NVIDIA</td>
</tr>
</tbody>
</table>

The following packages are planned for the future:

<table>
<thead>
<tr>
<th>Package name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>caffeeonspark</td>
<td>CaffeOnSpark brings deep learning to Hadoop and Spark clusters. By combining salient features from deep learning framework Caffe and big-data frameworks Apache Spark and Apache Hadoop, CaffeOnSpark enables distributed deep learning on a cluster of GPU and CPU servers.</td>
</tr>
<tr>
<td>cntk</td>
<td>The Computational Network Toolkit by Microsoft Research, is a unified deep-learning toolkit</td>
</tr>
<tr>
<td>bidmach</td>
<td>BIDMach is a very fast machine learning library.</td>
</tr>
</tbody>
</table>

1.3 Requirements

The following requirements must be met before installing the preceding machine learning packages:

- The base distribution used must be RHEL, Centos or Scientific Linux 7.x
- There must be access to the Linux distribution’s online YUM repositories as well the EPEL repository
- There must be 2 GB of free space for the RPMs that are installed on the head node, and an additional 400 MB for each software image that will be used for running machine learning pipelines

It is recommended, though not required, that the NVIDIA GPUs be Maxwell or more recent, with compute capability 3.5 or later.

1.3.1 Software Installation

Compute Nodes Installation

The cm-ml-distdeps RPM package must be installed onto all compute nodes that are to run machine learning applications. The cm-ml-distdeps meta-package instructs YUM to install the necessary system libraries as well as the development packages, e.g. blas-devel.

For example, if the name of the software image is gpu-image, then the administrator can install the RPM as follows:

Example

[root@bright73 ~]# yum install --installroot=/cm/images/gpu-image cm-ml-distdeps

The preceding command must be applied to all software images that are used to run machine learning applications.
1.3 Requirements

Head Node Installation
The head node must also have the cm-ml-distdeps RPM package installed on it. If it is not already installed, then it should be installed:

Example

[root@bright73 ~]# yum install cm-ml-distdeps

The Bright Cluster Manager machine learning packages have proper RPM dependencies defined. This means that the cluster administrator does not need to spend time figuring out what needs to be installed.

For example, if the administrator wants to install NVIDIA’s DIGITS, then all that needs to be done is to run the following command on the head node:

[root@bright73 ~]# yum install digits

In general, a `yum install <name of desired package>` should do installation for supported machine learning packages. That is, YUM then automatically installs cm-ml-pythondeps, cm-ml-distdeps, cudnn, caffe, torch and cuda75-toolkit as dependencies if needed.

The RPMs get installed in the /cm/shared directory, which is exported over NFS. The RPMS are therefore available to all the compute nodes. The installation of the machine learning libraries is therefore done within minutes, rather than the days that it typically takes to build and install all the necessary dependencies.

Software Libraries Useful For Developers
Developers that work on extending the machine learning libraries typically will not want to use the pre-packaged RPMs. For this use case, Bright can help minimize the time spent to get started.

By installing the cm-ml-pythondeps, cm-ml-distdeps (on the head node and compute nodes), and cudnn RPM packages a developer can get ready for machine learning development within minutes and spend time on the interesting application at hand, rather than waste time dealing with dependency hell.

1.3.2 Module Installation
Bright provides environment module definitions for all the machine learning packages. The environment module files are also compatible with the Lmod software introduced in 7.3.

The machine learning environment modules automatically load additional environment modules as dependencies.

For example, loading the DIGITS module with:

```
module load shared digits
```

automatically loads additional modules such cudnn, openblas, hdf5_18, and so on. Those modules are the dependencies needed to use DIGITS.

Example

[root@bright73 ~]# module list
Currently Loaded Modulefiles:
1) shared 3) cmsh 5) cluster-tools/trunk
2) cmgui/trunk 4) cmd 6) slurm/16.05.2
[root@bright73 ~]# module load shared digits
[root@bright73 ~]# module list
Currently Loaded Modulefiles:
1) shared 6) slurm/16.05.2 11) hdf5_18/1.8.17
2) cmgui/trunk 7) cm-ml-pythondeps/1.10.0 12) caffe/0.16

© Bright Computing, Inc.
4 Introduction And Installing the Machine Learning RPMs

3) cmsh 8) cudnn/5.0 13) torch7/7.0
4) cmd 9) openblas/dynamic/0.2.18 14) digits/4.0
5) cluster-tools/trunk 10) cuda75/toolkit/7.5.18

[root@bright73 ~]#

The module dependencies are achieved via the module definition files:

[root@bright73 ~]# module show digits

/cm/shared/modulefiles/digits/4.0:

module-whatis adds nVidia Deep Neural Network Library to your environment variables
module    load caffe
module    load torch7
module    load cm-ml-pythondeps
module    load cudnn
module    load openblas
module    load cuda75/toolkit
module    load hdf5_18
prepend-path PATH /cm/shared/apps/digits/4.0/

[root@bright73 ~]#

1.3.3 Further reading
Additional information about the usage of each individual framework may be found in the user manual.
This chapter goes through an example workflow in order to train a Caffe model to recognize handwritten digits. It closely follows the sample run by NVIDIA's Luke Yeager at https://github.com/NVIDIA/DIGITS/blob/master/docs/GettingStarted.md.

The example uses the MNIST handwritten digit database (http://yann.lecun.com/exdb/mnist) as the input to provide a training and validation dataset, and LeNet-5 (http://yann.lecun.com/exdb/lenet/) for the neural network model that is to be trained to classify the dataset. Both are generously made available by Yann LeCun from his website at http://yann.lecun.com/.

2.1 Downloading The MNIST Data

The MNIST dataset can be downloaded using the DIGITS downloader. For example, a user tim, could unpack the datasets into a directory /mnt as follows):

```
[tim@bright73 ~]$ python /cm/shared/apps/digits/current/digits/download_data/__main__.py
mnist /home/tim/
```

```
Downloading url=http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz ...
Downloaded
```

```
Downloading url=http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz ...
Downloaded
```

```
Downloading url=http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz ...
Downloaded
```

```
Downloading url=http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz ...
Downloaded
```

```
Uncompressing file=train-images-idx3-ubyte.gz ...
Uncompressed
```

```
Uncompressing file=train-labels-idx1-ubyte.gz ...
Uncompressed
```

```
Uncompressing file=t10k-images-idx3-ubyte.gz ...
Uncompressed
```

```
Uncompressing file=t10k-labels-idx1-ubyte.gz ...
Uncompressed
```

```
Reading labels from /home/tim/train-labels.bin ...
```

```
Reading images from /home/tim/train-images.bin ...
```

```
Reading labels from /home/tim/test-labels.bin ...
```

```
Reading images from /home/tim/test-images.bin ...
```

Dataset directory is created successfully at '/home/tim/'

Done after 37.8421010971 seconds.

2.2 Setting The Default Matplotlib Backend

The administrator can set the default matplotlib GUI backend supported by editing the file:

```
/cm/shared/apps/cm-ml-pythondeps/lib64/python2.7/site-packages/matplotlib/mpl-data/matplotlibrc
```

and, for example, changing the line:

```
backend: agg
```
Running Caffe

backend: gtk3agg

The preceding change allows GTK3 GUI support, if, for example, just basic X11 GUI support is not wanted and GTK3 is installed.

2.3 Creating A Working Directory
A working directory can be created with, for example:

```
[tim@bright73 ~]$ mkdir -p /tmp/digits/jobs
```

The actual path, which is set by the parameter jobs_dir, can be changed by the administrator in DIGITS version 6.0 in:

```
/cm/shared/apps/digits/6.0.0/digits/digits.cfg
```

2.4 Using The Web App
The DIGITS server can be started with:

```
[root@bright73 ~]# module load shared digits
[root@bright73 ~]# cd /cm/shared/apps/digits/current
[root@bright73 current]# ./digits-devserver
```

* Running on http://0.0.0.0:5000/

The server must be run with GPU. If run without a GPU some GPU or CUDA-related errors should show up in the output. If there is a GPU on the node, and the error #35 response is still displayed, then a possible cause is that the NVIDIA driver has not been loaded, or has loaded incorrectly.

Once the DIGITS server is up and running, a web browser can be used to navigate to the home screen of DIGITS. If all is well, then the server location should be indicated by the last line. The locations are typically at:

- `http://localhost/`
- `http://localhost:5000/` if using digits-devserver
- `http://localhost:34448/` if using digits-server

Instead of using localhost in the preceding URLs, the external IP address of the system can be used, if the Shorewall firewall is opened for the appropriate port. Typically the administrator adds lines to the rules file, similar to the output of the following grep:

```
[root@bright73 ~]# grep 5000 /etc/shorewall/rules
# -- Allow port 5000 (digits) traffic from elsewhere to master
ACCEPT:info net     fw       tcp     5000     #DIGITS
```
2.5 Logging In

After adding the lines, Shorewall should be restarted, with, for example, `service shorewall restart`.

The DIGITS home screen should then show up in the browser (figure 2.1).

![DIGITS Home Screen](image.png)

Figure 2.1: DIGITS Home Screen

2.5 Logging In

Clicking on the Datasets tab, then on the Images menu button under the New Dataset label, opens up a menu. Choosing the Classification menu item leads to the login page (figure 2.2). The login is carried out without authentication, for convenience. It is not a security feature.
2.6 Creating Training And Validation Datasets

After login the New Image Classification Dataset page appears (figure 2.3). The page shows several panes which group the input values for the training dataset that is about to be imported.
2.6 Creating Training And Validation Datasets

Figure 2.3: New Image Classification Dataset Dialog

For the pane starting with the Image Type field:
- The Image Type should be set to Grayscale
- The Image size should be changed to 28 x 28

For the pane starting with the Use Image Folder tab label:

© Bright Computing, Inc.
• For **Training Images**, the path to the MNIST training images should be entered. For the download described earlier in section 2.1 the path would be `/home/tim/mnist/train`.

• For the checkbox options:
  - The folder of MNIST test images can optionally also be added as a **Separate validation images folder**.
  - The **Separate test images folder** should not be used—test images are not used for anything in DIGITS yet.

For the pane starting with the **DB backend** field:

• The dataset should be given a name, for example `mnistdataset`.

The **Create** button can then be clicked to start the job run to create the classification databases. A new screen showing the classification job status is then displayed. The screen is scrollable in the browser. The screen is shown in this manual, for printing reasons, as a top part (figure 2.4) and a bottom part (figure 2.5).

A **Job Status** pane at the top right hand side (figure 2.4) shows the expected job completion time if it is refreshed before the job completes, and shows the time the job took after it has completed.
2.6 Creating Training And Validation Datasets

Figure 2.4: DIGITS MNIST Job Run Screen, Top Part

The bottom section (figure 2.5) allows access to the input and log files.
After the training and validation database import run is complete, the classification spread can be visualized in the refreshed updated bottom section (figure 2.6).
2.6 Creating Training And Validation Datasets

Figure 2.6: DIGITS MNIST Job Run Screen, Bottom Part, After Run

© Bright Computing, Inc.
The corresponding training (train) and validation (val) database entries can be explored by clicking the Explore the db button.

Clicking DIGITS in the top left hand part of the web page brings up the home page again. Within the Datasets tab (figure 2.7), the name mnistdataset that was set earlier is now visible. Clicking on the name brings a screen where like in figure 2.6 the classification spread can be visualized, and the corresponding training (train) and validation (val) database entries can be explored by clicking the Explore the db button.

2.7 Training A Model

The datasets can now be used to train the LeNet model.

Clicking on the Images button that is associated with the New Model label (figure 2.7), and then on the Classification option, opens up the New Image Classification Model page. The top part of the page is shown in figure 2.8:
2.7 Training A Model

Figure 2.8: New Image Classification Model Top Part

The bottom part of the page, with network and GPU options (figure 2.9) shows network and GPU options.
In the New Image Classification Model page the following steps can be followed to classify the data:

- **mnistdataset** is selected in the Select Dataset field
- The **LeNet** network can be selected from the Standard Networks tab
- The model is given a name, for example **Len**
- The **Create** button is clicked

If there is no GPU on the system, then the **Torch** (experimental) tab should be selected instead of Caffe.

While training the model, the expected completion time is seen on the right side (figure 2.10):
2.8 Testing A Model

The bottom of the page (figure 2.11) allows the model to be tested.
Figure 2.11: New Image Classification Of One Image

- The **Browse button** in the **Upload Image field**, in the **Test a single image** can be clicked upon and a file chosen.
  - There are plenty to choose from under the `/home/username/mnist/test/` directory.
  - Alternatively an image of a digit can be created by hand in an image manipulation software such as Gimp.
  - Alternatively an image from the web can have its URL pasted into the **Image URL** field
- The **Show visualizations and statistics** box can be checked for extra information on how the algorithm was applied.

Clicking on the **Classify One button** then displays a classified image (figure 2.12):
At the top of the page, DIGITS displays the top five classifications and corresponding confidence values. DIGITS also provides further visualizations and statistics about the weights and activations of each layer of the network.
This chapter goes through some example runs with TensorFlow. The INFO output messages have been removed in the runs for readability.

### 3.1 Hello World

A “Hello World” example that just shows that the software is in place for TensorFlow 0.10 can be run as follows:

```python
[root@bright73 ~]# python
Python 2.7.5 (default, Nov 20 2015, 02:00:19)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-4)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow as tf

hello = tf.constant('Hello, TensorFlow!')

>>> sess = tf.Session()

name: Tesla K40c
major: 3 minor: 5 memoryClockRate (GHz) 0.745
pciBusID 0000:05:00.0
Total memory: 11.92GiB
Free memory: 11.78GiB

> (device: 0, name: Tesla K40c, pci bus id: 0000:05:00.0)
>>> sess.run(hello)
'Hello, TensorFlow!'

>>> a = tf.constant(10)
>>> b = tf.constant(32)

>>> sess.run(a+b)
42

```

### 3.2 Training A Convolutional Neural Network

The following trains a convolutional neural network similar to LeNet-5, as explained in [https://www.tensorflow.org/versions/r0.11/tutorials/mnist/beginners/index.html](https://www.tensorflow.org/versions/r0.11/tutorials/mnist/beginners/index.html).

The code uses the TensorFlow convolutional module at `/cm/shared/apps/tensorflow/0.10/lib/python2.7/site-packages/tensorflow/models/image/mnist/convolutional.py`. It picks up training images and labels from the MNIST site, and places them in a directory data if it needs to. The images are then used to train and validate the model.

© Bright Computing, Inc.
Example

[root@bright73 ~]# python -m tensorflow.models.image.mnist.convolutional
Successfully downloaded train-images-idx3-ubyte.gz 9912422 bytes.
Successfully downloaded train-labels-idx1-ubyte.gz 28881 bytes.
Successfully downloaded t10k-images-idx3-ubyte.gz 1648877 bytes.
Successfully downloaded t10k-labels-idx1-ubyte.gz 4542 bytes.
Extracting data/train-images-idx3-ubyte.gz
Extracting data/train-labels-idx1-ubyte.gz
Extracting data/t10k-images-idx3-ubyte.gz
Extracting data/t10k-labels-idx1-ubyte.gz
name: Tesla K40c
major: 3 minor: 5 memoryClockRate (GHz) 0.745
pciBusID 0000:05:00.0
Total memory: 11.92GiB
Free memory: 11.78GiB
Initialized!
Step 0 (epoch 0.00), 36.5 ms
Minibatch loss: 12.054, learning rate: 0.010000
Minibatch error: 90.6%
Validation error: 84.6%
Step 100 (epoch 0.12), 14.2 ms
Minibatch loss: 3.301, learning rate: 0.010000
Minibatch error: 4.7%
Validation error: 7.4%
Step 200 (epoch 0.23), 14.2 ms
Minibatch loss: 3.458, learning rate: 0.010000
Minibatch error: 14.1%
...
Step 8400 (epoch 9.77), 14.1 ms
Minibatch loss: 1.596, learning rate: 0.006302
Minibatch error: 0.0%
Validation error: 0.8%
Step 8500 (epoch 9.89), 14.1 ms
Minibatch loss: 1.626, learning rate: 0.006302
Minibatch error: 1.6%
Validation error: 0.9%
Test error: 0.8%
[root@bright73 ~]#

3.3 Image Recognition

The following session shows a pre-trained model recognizing a test image of a tiger:

Example

[root@bright73 ~]# module load tensorflow/0.10
[root@bright73 ~]# cd /cm/shared/apps/tensorflow/0.10/lib/python2.7/site-packages/tensorflow\models/image/imagenet
[root@bright73 imagenet]# python classify_image.py --image_file=Indochinese-Tiger-Zoo.jpg
name: Tesla K40c
major: 3 minor: 5 memoryClockRate (GHz) 0.745
pciBusID 0000:05:00.0
Total memory: 11.92GiB
Free memory: 11.78GiB
W tensorflow/core/framework/op_def_util.cc:332] Op BatchNormWithGlobalNormalization is depre
3.3 Image Recognition

cated. It will cease to work in GraphDef version 9. Use tf.nn.batch_normalization().
tiger, Panthera tigris (score = 0.71628)
tiger cat (score = 0.11725)
lynx, catamount (score = 0.00376)
jaguar, panther, Panthera onca, Felis onca (score = 0.00371)
cougar, puma, catamount, mountain lion, painter, panther, Felis concolor (score = 0.00218)
[root@bright73 imagenet]#