**A Tutorial on Caffe in Window 10**

Thuong Nguyen Canh

Digital Media Lab

Sungkyunkwan University

June 19, 2017

Contents

[1. Introduction 3](#_Toc485655110)

[2. Training Simple Neural Network with MINIS Data 3](#_Toc485655111)

[1.1. Step 1, Prepare MNIST data 3](#_Toc485655112)

[1.2. Step 2. Model definition 5](#_Toc485655113)

[A. Writing the data layer. 5](#_Toc485655114)

[B. Writing the convolutional layer. 5](#_Toc485655115)

[C. Writing the Pooling Layer 6](#_Toc485655116)

[1.3. Step 3, Solver Definition 6](#_Toc485655117)

[1.4. Step 4, Model Training 6](#_Toc485655118)

[Appendix 6](#_Toc485655119)

[I. Installing Bash 6](#_Toc485655120)

[II. Where the Ubuntu Bash Shell Files Are Stored in Windows [2] 7](#_Toc485655121)

[III. Where Your Windows System Drive Appears in Bash 8](#_Toc485655122)

[Appendix 2 – Google Protobuf 8](#_Toc485655123)

[3. Reference: 8](#_Toc485655124)

# **Introduction**

This work presents a detail tutorial on Caffe with Command line, MATLAB and python interface in Window 10.

# **(ComLine) Training Neural Network with MNIST CPU-Only**

There are 4 steps in training a CNN using Caffe [5]:

* Step 1 - Data preparation: In this step, we clean the images and store them in a format that can be used by Caffe. We will write a Python script that will handle both image pre-processing and storage.
* Step 2 - Model definition: In this step, we choose a CNN architecture and we define its parameters in a configuration file with extension. prototxt.
* Step 3 - Solver definition: The solver is responsible for model optimization. We define the solver parameters in a configuration file with extension. prototxt.
* Step 4 - Model training: We train the model by executing one Caffe command from the terminal. After training the model, we will get the trained model in a file with extension. caffemodel.

After the training phase, we will use the .caffemodel trained model to make predictions of new unseen data. We will write a Python script to this.

## **Step 1, Prepare MNIST data**

To prepare the MNIST example, Cafffe does support example with MNIST data. However, they use bash file written for Linux based system to download and generate data. So we cannot run the file directly on Window.

We could reproduce what written in bash files however it is difficult. One solution is about installing linux bash for window. You can follow this tutorial to enable ubuntu bash on Window 10 [1] in Appendix 1.

Then we follow the MNIST tutorial in [3]. I repost it here for the completed tutorial.

Prepare Datasets

You will first need to download and convert the data format from the MNIST website. To do this, simply run the following commands:

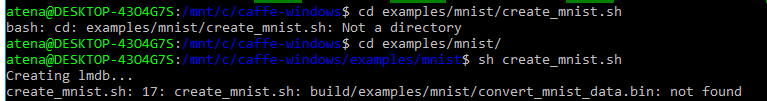
cd $CAFFE\_ROOT

./data/mnist/get\_mnist.sh

./examples/mnist/create\_mnist.sh

If it complains that wget or gunzip are not installed, you need to install them respectively. After running the script there should be two datasets, mnist\_train\_lmdb, and mnist\_test\_lmdb.

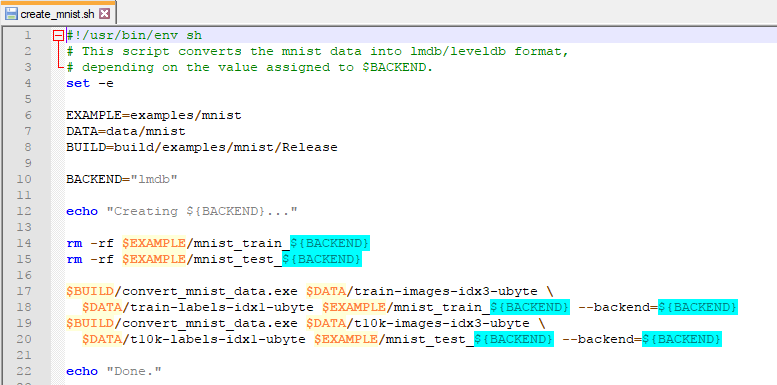
Again, it should be noted that the create\_mnist.sh file is written for linux based system. Since we running example on window, we will face with problem of cannot found “convert\_mnist\_data.bin” as shown below



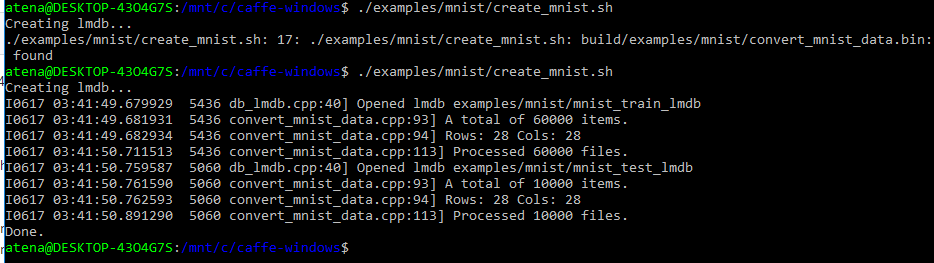
Firstly, we need to come back to the root folder (here they denoted as $CAFFE\_ROOT$) of Caffe as



However, we still face with same problem of cannot find *convert\_mnist\_data.bin*. It is because the execution file in window have “\*.exe” extension not “.bin” as in Linux. In addition, the file is in the “Release” or “Debug” in build folder, not as same as in linux system. Therefore, we need to change create\_mnist.sh to update the folder “BUILD” for the location of convert\_mnist\_data.exe, and change “.bin” to “.exe”. The changed file is shown following. In my case, I build Caffe in release mode, so that Release is added to BUILD.



Basically, this create\_mnist.sh download and extract the MNIST data. Then running the sh again to generate the lmdb data.



Note: should be updated how to make the training and label data.

## **Step 2. Model definition**

We use lenet defined in $CAFFE\_ROOT/src/caffe/proto/caffe.proto. The details are given as follows.

### ***Writing the data layer.***

name: “LeNet”

layer {

name: "mnist"

type: "Data"

transform\_param {

scale: 0.00390625

}

data\_param {

source: "mnist\_train\_lmdb"

backend: LMDB

batch\_size: 64

}

top: "data"

top: "label"

}

At first, the network name is named as “LeNet”, then we define data reading layer from lmdb file. The layer has *name* “minist” and *type* “Data”. The *transform\_param* define parameter *scale* = 0.00390625 which equals 1/255. That means we normalized the data in the range of [0, 1]. For *data\_pararm*, the *source* spefifies the input data with *backend* (or filetype) is LMDB and *batch\_size* of 64. There are two **blobs** for “data” and “label”.

### **Writing the convolutional layer.**

layer {

name: "conv1"

type: "Convolution"

param { lr\_mult: 1 }

param { lr\_mult: 2 }

convolution\_param {

num\_output: 20

kernel\_size: 5

stride: 1

weight\_filler {

type: "xavier"

}

bias\_filler {

type: "constant"

}

}

bottom: "data"

top: "conv1"

}

This layer names “conv1” with type of “Convolution”. It takes the data blob (which is provided by the data layer) as *bottom: “data”* and output the result of “conv1”. The outputs contain 20 channels with the convolutional kernel of size 5 and stride of 1.

The filter randomly initiate weights and bias. The weight filter uses *Xavier* algorithm (what is this?) which will automatically determine the scale of initialization based on the number of input and output neuron. The bias filter will be initialized with default 0.

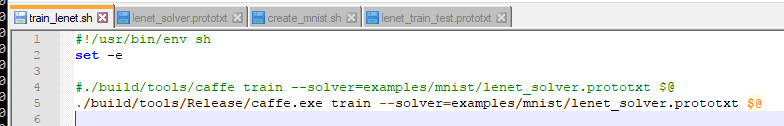
*lr\_mult* defines the learning rate adjustment for the layer’s learnable parameters. In this example, the learning rate for weight and the solver is the same, and the learning rate for bias will be twice times larger. In practice, this setting offers faster convergence rate.

### **Writing the Pooling Layer**

## **Step 3, Solver Definition**

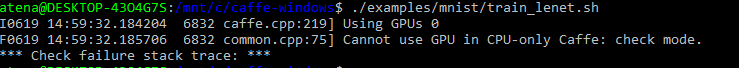
## **Step 4, Model Training**

In order to run the model training, we should execute the *\*.sh* file “train\_lenet.sh”. Again, this file is written for Linux based system, we should change it to window based.

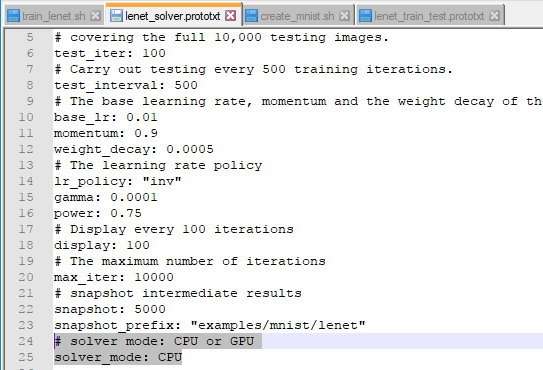


Basically this function will execute caffe (which is built in release mode in \*.exe file in window) at *train* mode with solver defined in “lenet\_solver.protxt”.

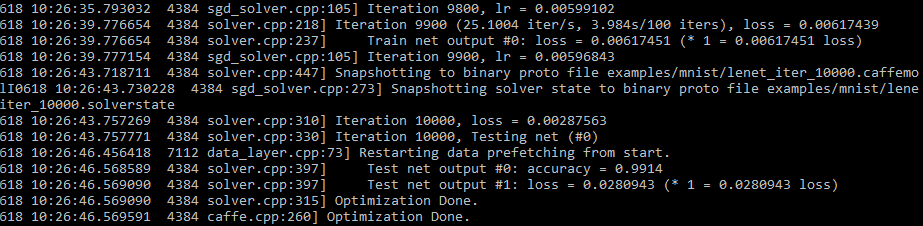
At this step, I face with problem of cannot run for GPU configuration. It is because I did not build GPU enable Caffe.



We can switch it to CPU mode by changing the *lenet\_solver.prototxt*  file.



Then we can run the training Lenet network without any problem.



# **(ComLine) Training Simple Neural Network with MINIS Data - GPU**

To train the neural network with GPU seting, make sure that we compile Caffe with GPU-enabled by setting “CPU Only” to 0 in “build\_win.cmd” in /script folder.

I already rebuild Caffe with CPU Only = 0, but the problem still remain. I notice that matlab interface also not responding with GPU configuration. There are two possible actions could be checked for this issue.

Firstly, we did not process the last step of copy caffe\_ file in installation. I checked it, but the problem still remain. I install matlab 2016b but no different.

Secondly, the build in my own computer is working with matlab interface, I will check it again. The matlab intereface in my PC can run GPU enable. It also can run in command line in my PC. Where is the source of this problem? The only thing is the caffe build.

Answer. I have found one mistake in my installation. The cudatoolkit is not installed successfully. But I still got the same error.

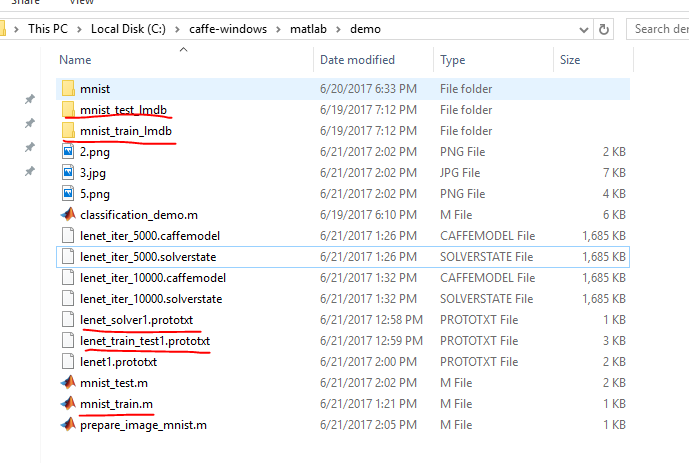
Approach 2: I will delete all build folder and build it again.

Solution. Building caffe again with correct installation of CUDA 8.0 and cudnn5.1 (copy to corresponding folder). I have successfully train Lenet under GPU with command line.

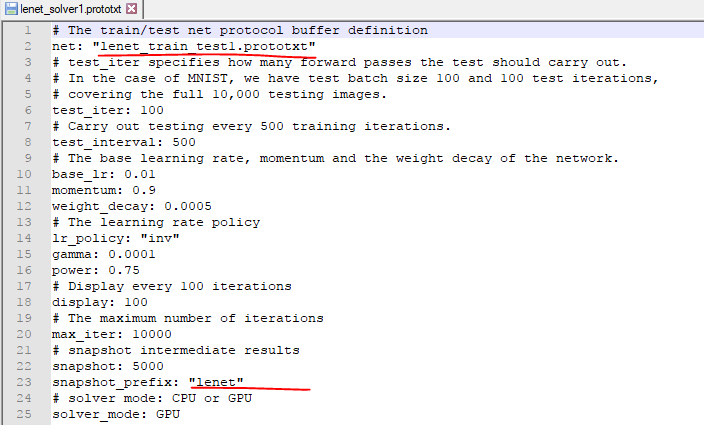
# **(Matlab) Training Neural Network with MNIST**

Even though Caffe supports MATLAB interface but there is limited support and tutorial for Matcaffe. In order to perform training on Matlab with MNIST data, we need to make sure all location ( about the file and data is corrected for window application).

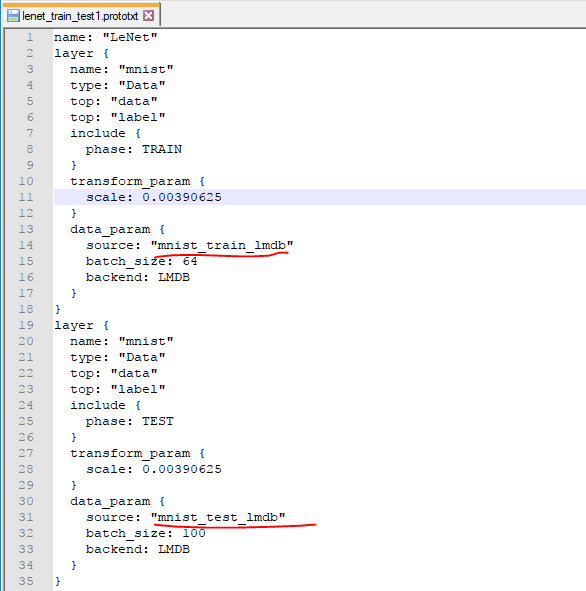
The simplest solution is to put everything required into one folder like



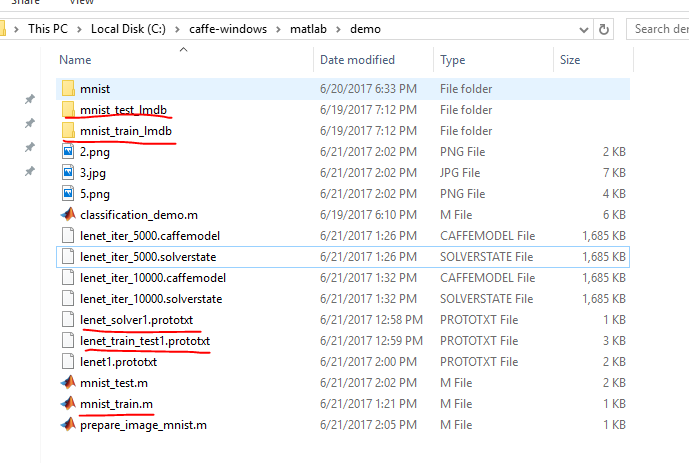
Where the lenet\_solver1 and lenet\_train\_test1 is modified from its original by changing the location index in the file as



And



Then we will get the results of trained data as



## Testing with MNIST in Matlab interface

Testing custom hand writing image is required to follow classification\_demo.m. Firstly, we need to define the net\_model described in “lenet.prototxt”. However, this model often go with fixed number of test image. For example, 64 test images are used as default. We should change this number depend on our testing.

Secondly, pre-processing image is required which corresponding to resize image and normalized to [0, 1] range.

# **Appendix I: Bash on Window**

1. Installing Bash

There are several things should be noticed for this bash using. Firstly, few common commands on linux are given

* Change Directory: **cd** in Bash, **cd** or **chdir** in DOS
* List Contents of Directory: **ls** in Bash, **dir** in DOS
* Move or Rename a File: **mv** in Bash, **move** and **rename** in DOS
* Copy a File: **cp** in Bash, **copy** in DOS
* Delete a File: **rm** in Bash, **del** or **erase** in DOS
* Create a Directory: **mkdir** in Bash, **mkdir** in DOS
* Use a Text Editor: **vi** or **nano** in Bash, **edit** in DOS

You’ll need to [use the apt-get command](https://www.howtogeek.com/63997/how-to-install-programs-in-ubuntu-in-the-command-line/) to install and update the Ubuntu environment’s software. Be sure to prefix these commands with “sudo”, which makes them run as root–the Linux equivalent of Administrator. Here are the apt-get commands you’ll need to know:

* Download Updated Information About Available Packages: **sudo apt-get update**
* Install an Application Package: **sudo apt-get install packagename** (Replace “packagename” with the package’s name.)
* Uninstall an Application Package: **sudo apt-get remove packagename** (Replace “packagename” with the package’s name.)
* Search for Available Packages: **sudo apt-cache search word** (Replace “word” with a word you want to search package names and descriptions for.)
* Download and Install the Latest Versions of Your Installed Packages: **sudo apt-get upgrade**

Once you’ve downloaded and installed an application, you can type its name at the prompt and press Enter to run it. Check that application’s documentation for more details.

## **Where the Ubuntu Bash Shell Files Are Stored in Windows [2]**

The Ubuntu system files are stored at:

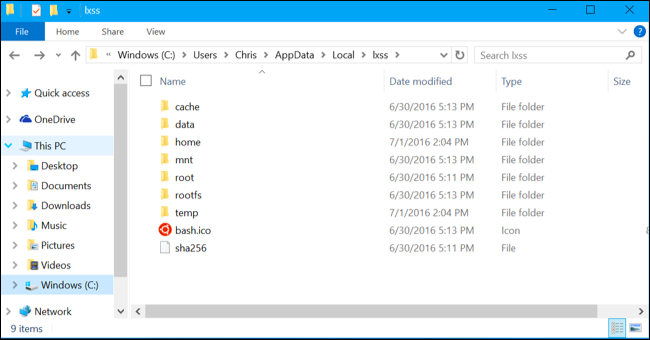
C:\Users\USERNAME\AppData\Local\Lxss\rootfs

Your Ubuntu user account’s home folder is stored at:

C:\Users\USERNAME\AppData\Local\Lxss\home\USERNAME

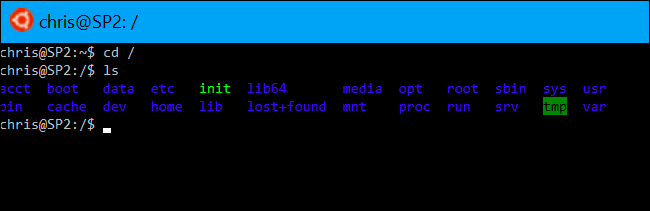
The root account’s home folder is stored at:

C:\Users\USERNAME\AppData\Local\Lxss\root



1. **Where Your Windows System Drive Appears in Bash**

The Bash environment doesn’t just dump you in your C:\ drive. Instead, it places you in /, or the root directory you’d have on Linux.



To change to this directory with the “cd” command, run the following command:

cd /mnt/c

# Appendix 2 – Google Protobuf

1, Definition.

The google protocol buffer method for serializing structured data [4] with smaller and faster than XML. Protocol Buffers is widely used for storing and interchanging structured information.

# **Reference:**

1. <https://www.howtogeek.com/249966/how-to-install-and-use-the-linux-bash-shell-on-windows-10/>
2. <https://www.howtogeek.com/261383/how-to-access-your-ubuntu-bash-files-in-windows-and-your-windows-system-drive-in-bash/>
3. <http://caffe.berkeleyvision.org/gathered/examples/mnist.html>
4. <https://en.wikipedia.org/wiki/Protocol_Buffers>
5. A. Moujahid, Introduction deep learning with python Caffe, 2016. Available at adilmoujahid.com/posts/2016/06/introduction-deep-learning-python-caffe/