# A Simple Tutorial for FastFlexANN

Wei (Wesley) Wen
Wew57@pitt.edu

## 1. What's FastFlexANN

FastFlexANN is a C implementation of sparse artificial neural networks (Multi-Layer Perceptron, MLP). Its advantages are:

- Compact and easy to be understood;
- Written by stand C to be portable to all C-compatible platforms;
- The connective topology (for both fully and sparsely connected MLP) between layers can be flexibly specified by *net file*;
- Connections can be pruned/reduced during training to sparsify nets;
- Samples can be easily formatted as *train file* and *test file.*

## 2. What's in repository

*./inc* - header files
*./src* - source files
*./matlab* – matlab scripts to generate MNIST samples and net files

## 3. How to run

### 3.1 Build

```
cd $FastFlexANNFolder/src
make clean #always run this
make
```

### 3.2 Run and parameter configuration

```
./main [database] [epochs] [learning rate] [pruning rate] [normalized] [net file] [weight file] [train file] [test file]
```

All parameters are required

- *database*: database for training and testing on neural networks. "*mnist*" and "*text*" are supported (in the first version, both work in the same way)
- *epochs*: epochs of training (ex. *100*)
- *learning rate*: learning rate of weights (ex. *0.0001*)
- *pruning rate*: the rate of connections to be pruned when conditions are satisfied (ex. *0* implying no pruning). The condition is if validating error is smaller than 5%
- *normalized*: reserved (always *0*)
- *net file*: the path of file for net topology
- *weight file*: the path of file of initial weights (ex. *default* – no initial file are specified and weights are randomly initialized between -0.05 and 0.05)
- *train file*: the path of file of training samples (ex. *MNIST_train.txt*)
- *test file*: the path of file of testing samples (ex. *MNIST_test.txt*)

### 3.3 Generate net, train and test files by Matlab

1) Download MNIST training and testing files from http://yann.lecun.com/exdb/mnist/
2) Unzip and place MNIST files in matlab/MNIST
3) Run MNIST_gen.m to load samples from binary MNIST files
4) Run MNIST_txt_gen.m to generate *train file* and *test file* named as "*MNIST_train.txt*" and "*MNIST_test.txt*" (pixels are normalized between -1 and 1)

5) Run full_connection_gen.m to generate *net file* for fully connected MLP
ex.
*>> full_connection_gen([784 300 100 10]));*
*full_nn_784_300_100_10.txt* will be generated, which is a MLP with 4 layers composed of 784 input neurons, 300+100 hidden neurons and 10 output neurons (Excluding these neurons, a *bias neuron* is also implicitly added at the end of every except output layers, and it is connected towards all neurons in the next layer, so as to implement all biases for neurons in the next layer.)

## 3.4 An running example

```
wesley$ ./main mnist 10 0.0001 0 0 ../matlab/full_nn_784_300_100_10.txt default ../matlab/MNIST_train.txt ../matlab/MNIST_test.txt
```

Output is:

```
loading files...
files loaded...
0
2000 #number of  samples are trained
4000
6000
8000
```

Information is logged in file *mnist.log*

```
/*************************************************************************\
Sun Jun 14 03:14:22 2015
**************************************************************************
[parameter config:
                          mnist weights [785 301 101 11] 266610 0 0.000100 0.000000 0
network topology is read from ../matlab/full_nn_784_300_100_10.txt
0 epochs are escaped since recovered weights...
[epoch     1]: test(validate) error is 0.134400(0.146000)
[epoch     2]: test(validate) error is 0.112400(0.122200)
[epoch     3]: test(validate) error is 0.106500(0.117500)
[epoch     4]: test(validate) error is 0.087700(0.098200)
[epoch     5]: test(validate) error is 0.081800(0.085700)
[epoch     6]: test(validate) error is 0.075500(0.078100)
[epoch     7]: test(validate) error is 0.072000(0.071300)
[epoch     8]: test(validate) error is 0.067500(0.072100)
[epoch     9]: test(validate) error is 0.065000(0.067400)
[epoch    10]: test(validate) error is 0.059500(0.062900)
Sun Jun 14 03:52:51 2015
\-------------------------------------------------------------------------/
```

# 4. File format

**[*net file*] – text file**

Format:

```
LayerNumber NeuronNumberInEveryLayer ConnectionNumber ReservedNormalizedParameter
OneConnectionBetweenTwoNeurons
OneConnectionBetweenTwoNeurons
OneConnectionBetweenTwoNeurons
...
```

Example:

```
4 784 300 100 10 266610 0 # MLP with 4 layers composed of 784 input neurons, 300+100 hidden neurons and 10 output neurons (excluding bias neurons). Totally 266610 connections (including those connected bias neurons)
0 785 # a connection exists between neuron 0 and 785, bias neurons are also indexed in this connection list
0 786
0 787
...
```

**[*train file/test file*] - text file**

Format:

```
FeatureDimension NumberOfClasses NumberOfSamples
FeaturesOfSample1 LabelOfSample1
FeaturesOfSample2 LabelOfSample2
...
```

Example:

```
784 10 60000 # 60000 training samples for MNIST dataset
```

-1.00 0.913 … 0.876 9 # a sample of digit 9
…

**[*weight file*] – binary file**

Filename Format:
*database*_weights_[*neuron#*]_*connection#_userdata_learningRate_pruningRate_ reserved#*
Ex.
mnist_weights_[785_301_101_11]_266610_0_0.000100_0.000000_0

Corresponding pruned *net file* may be saved as "pruned_*$weightfilename*.txt"
Ex.
pruned_mnist_weights_[785_301_101_11]_946_0_0.000100_0.050000_0.txt

Content Format:
1 integer: the epoch # of finished training when saving weights
1 integer: the number of weights/connections including those connected with bias neurons
fflex_msg_t(double) array: weight values

**[*.log]**

Logging info

## 5. Implementation details

- Referred to paper "Dan Claudiu Ciresan. *etc*. Deep Big Simple Neural Nets Excel on Handwritten Digit Recognition"
- 1/6 samples are utilized as validating samples
- Comments are given in codes