LAB ACTIVITY 10:
XOR EXAMPLE: INTRODUCTION TO FAST ARTIFICIAL NEURAL NETWORK (FANN) LIBRARY

CARANDO, Paolo Creas
GOMENDOZA, Frank
LIM, Theo Paolo
LORENA, Neil John
ZABAT, Juan Paolo

I.INTRODUCTION

The Fast Artificial Neural Network (FANN) Library enables the implementation of multilayer artificial neural networks in C language. Cross-platform implementation for fixed and floating point are included in the library. It also contains frameworks to easily handle training sample sets. Well-documented library that includes manuals and implementation reports. Faster than other libraries. Provides easier implementation of ANN because of its three function calls (create, train and run). And has language bindings to different programming languages.

II.PROCEDURE

Fast Artificial Neural Network (FANN) Library:

1. Download the FANN repository from: https://github.com/DeLaSalleUniversity-Manila/fann
2. Clone the repository and go to the root directory.

```
git clone https://github.com/libfann/fann.git
cd ./fann
Code 1.
```

3. Run CMake and install.

```
cmake .
sudo make install
Code 2.
```

4. Run XOR sample train file:

```
g++ xor_train.c -o xor_train -lfann -lm
cat xor.data
./xor_train xor.data
Code 3.
```

5. After running the train file, test the XOR network.

```
g++ xor_test.c -o xor_test -lfann -lm
./xor_test
Code 4.
```

ANN with FANN on MNIST:

1. Download the repository from: https://github.com/DeLaSalleUniversity-Manila/ArtificialNeuralNetworkWithFANNonMNIST
2. Clone the repository and go to the root directory.

```
git clone https://github.com/DeLaSalleUniversity-Manila/ArtificialNeuralNetworkWithFANNonMNIST.git
cd ArtificialNeuralNetworkWithFANNonMNIST/
Code 5.
```

3. Perform these commands to compile.

```
make g++ src/train.cpp -lfann -o train
```

```
g++ src/test.cpp -lfann -o test
g++ src/convert.cpp -o convert
g++ src/bulk-test.cpp -lfann -o bulk-test
```
Code 6.

4. After compiling, convert MNIST format into FANN training file.

```
./convert
```
Code 7.

5. Perform training of the network with these commands. Set the training parameters according to your choice.

```
Usage: ./train [options]
Options:
 --mse <error>              Mean square error to reach
 --epoch <max>              Maximum epochs number
 --network <network-file>   Network file
 --train <train-file>       The training file
 --backprop-batch           Learning algorithm
 --backprop-incr            Learning algorithm
 --quickprop                Learning algorithm
 --help                     Display this information

./train --mse 0.001 --epoch 500 --train training.data --backprop-batch
```
Code 8.

6. After training, perform testing of the network

```
Usage: ./test [options]
Options:
 --network <network-file>  Network filename
 --lbl <lbl-file>          Labels filename
 --img <img-file>          Images filename
 --help

./test --network network.nn

Bulk Testing:
Usage: ./bulk-test [options]
Options:
 --network <network-file>  Network filename
 --lbl <lbl-file>          Labels filename
 --img <img-file>          Images filename
 --tot <tests-no>          Number of test to do
 --skip <starting-from>    Number of image to skip from begin
 --help                    Display this information
./bulk-test --network network.nn --tot 1000 --skip 1000
```
Code 9.

III.RESULTS AND DISCUSSION

Fast Artificial Neural Network (FANN) Library:

XOR sample train:
4 2 1
-1 -1
-1
-1 1
1
1 -1
1
1 1
-1

Max epochs 1000. Desired error: 0.0000000000.
Epochs 1. Current error: 0.2763172984. Bit fail 4.
Epochs 10. Current error: 0.1761302352. Bit fail 4.
Epochs 20. Current error: 0.0331001319. Bit fail 4.
Epochs 30. Current error: 0.0004698806. Bit fail 4.
Epochs 34. Current error: 0.0000754816. Bit fail 0.
Testing network. 0.000056

XOR test (-1.000000,-1.000000) -> -

0.990666, should be -1.000000, difference=0.009334
XOR test (-1.000000,1.000000) -> 0.977528, should be 1.000000, difference=0.022472
XOR test (1.000000,-1.000000) -> 0.989648, should be 1.000000, difference=0.010352
XOR test (1.000000,1.000000) -> -0.986067, should be -1.000000, difference=0.013933

Testing XOR:
XOR test (-1.000000, -1.000000) -> -0.990666, should be -1.000000, difference=0.009334
XOR test (-1.000000, 1.000000) -> 0.977528, should be 1.000000, difference=0.022472
XOR test (1.000000, -1.000000) -> 0.989648, should be 1.000000, difference=0.010352
XOR test (1.000000, 1.000000) -> -0.986067, should be -1.000000, difference=0.013933

ANN with FANN on MNIST:

FANN using MNIST:
Test Done: 1000
Test Passed: 827
82% of passed test

IV.CONCLUSION

For this experiment we were able to apply the fast artificial neural network for testing and training the data. After running the train file, we test the XOR network. We applied FANN on MNIST by compiling it and training and testing it. For testing the iteration was set to 1000 and the best desired error should be 0.00%. As the testing of the network goes up to 1000 errors were encountered on certain epochs. The test done 1000 and there are 827 that passed. The training parameters that was set for training the data is important for the efficiency of the network. The number of epochs may have an effect for the accuracy of the network. This is just a simple function so this can be done without the validation process. The network can be improved by simply training the data.

V.BIBLIOGRAPHY

[1] S. Nissen, 'FANN', FANN, 2015. [Online]. Available: http://leenissen.dk/fann/wp/.