C++ Capstone Project

Implement Neural Network

Christopher Gorzek

July 29, 2022

GitHub Repository Link:

https://github.com/cgorzek/CppND-Capstone-NN

Building and Running the Project:

In the project directory make a directory called `build`. Change directories to the new directory. Type `cmake ..` , followed by `make`. To run the project type: `./CapstoneNN <return>`.

Additional Libraries:

None. Note: Eigen library is used but is available in the current installation.

Project Description:

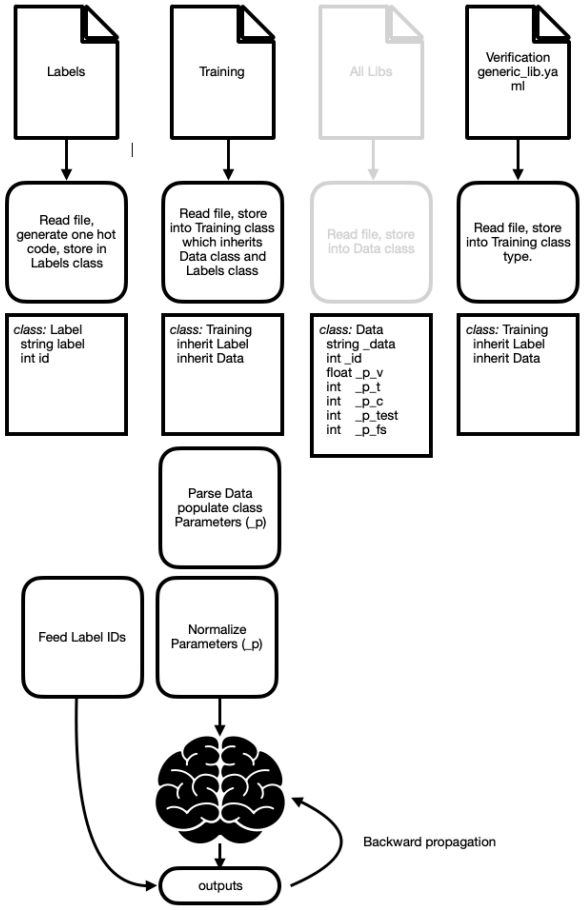
When doing ASIC Physical Design we have to deal with hundreds of library files from our vendors. These need to be grouped into labeled sets based on voltage, temperature, extraction corner and more. Each ip vendor has their own method for naming these files. It is tedious and error prone. I wanted to learn more about AI so I thought I would see if I could get a trained Neural Network to do this sorting for me on a generalized set of library files.

I used a neural network I found from the Udacity knowledge posts: https://www.geeksforgeeks.org/ml-neural-network-implementation-in-c-from-scratch Fixed some bugs and got it running on it's own test suite.

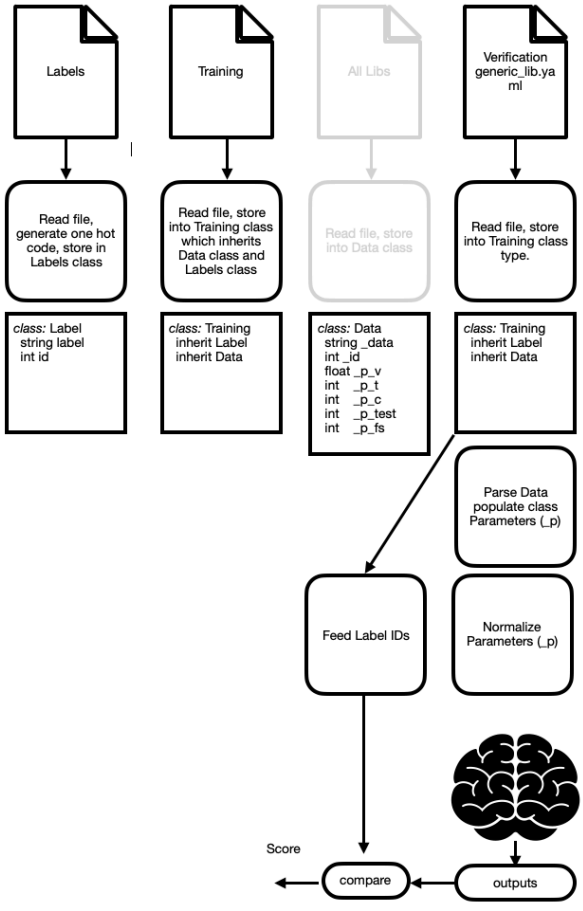
I picked 5 parameters to search for in the file names. I have one file with training data that is a subset of approximately 40% of the final test data that is grouped into the appropriate library sets. The 5 parameters are then encoded

where appropriate and normalized and fed into the Neural network. Label data was encoded and back propagated when training or compared with the output when running the verification step.

Results were not great but based on some final debugging where I went from 9% matching to 21% matching by fixing some issues with input data there is a lot more to learn about effectively setting up using neural networks.



Training (nn.train( ))



Verification (nn.verify( ))

Project File and Class Structure:

Code is contained in CppND-Capstone-NN/src:

Data.cpp	Data class, holds majority of the data
Data.h	
Labels.h	Label class, holds group labels and id code
NeuralNetwork.cpp	Neural Network from www.geeksforgeeks.org
NeuralNetwork.hpp	
TrainingData.h	TrainingData Class, inherits both Data and Labels
main.cpp	

Data is contained in CppND-Capstone-NN/data:

all_lib_files.txt	List of library files. Used only if running nn.work(). (currently not used in this implementation)
generic.lib.yaml	Full list of labeled library files, used for nn.verify()
tags.txt	List of labels, read into labelData and encoded.
training_lib.yaml	Training Data. All generic.lib.yaml labels 40% of library file data.

Project Expected Output:

The end of the output to the screen looks like this:

```
Input to neural network is :  1  1  1 -1 -1
Expected output is : 12
Output produced is : 12
  correct
Input to neural network is :  1  1  1 -1 -1
Expected output is : 12
Output produced is : 12
  correct
Score: 21%  (199 of 913)
Finished Analysis with Verify........
root@e47703e693c3:/home/workspace/CppND-Capstone-NN/build#
```

The first pass through the neural net is training. The second is a full data set with verification. For verification I added checking that rounds the neural net output and checks it against the expected integer. The “Score” is the number that produced the correct result.

Rubric Points addressed (files, line numbers):

“Review the rubric. Ensure that the application you build will satisfy all criteria for the “README” and “Compiling and Testing” sections, and that the application will satisfy at least 5 total criteria from the rest of the rubric. Not all rubric items need to be satisfied in order to pass the project. Scope the features of your project to meet the rubric criteria you want to target.”

Loops, Functions, IO

- The project demonstrates an understanding of C++ functions and control structures. File: main.cpp starting at line 58 function uses various control structures.
- The project reads data from a file and process the data, or the program writes data to a file. File: main.cpp starting at line 26 including several functions that follow read data from several file types.

Object Oriented Programming

- The project uses Object Oriented Programming techniques. (The project code is organized into classes with class attributes to hold data, and class methods to perform tasks. Files: Data.cpp, Data.h, Labels.h, TrainingData.h
- Classes use appropriate access specifiers for class members. Files: Data.cpp, Data.h, Labels.h, TrainingData.h
- Classes abstract implementation details from their interfaces. File: Data.h, Data.cpp, Data.cpp starting at line 51 which automatically formats and populates parameters _p and encodes certain parameters where needed so that

the getter functions can return data that can be formatted for entry to the neural network.

- Classes encapsulate behavior. (Appropriate data and functions are grouped into classes. Member data that is subject to an invariant is hidden from the user. State is accessed via member functions.) Files: Data.cpp,Data.h, Labels.h, TrainingData.h. In addition to the information provided in the previous bullet TrainingData needs both data and label data thus inherits both classes.
- Classes follow an appropriate inheritance hierarchy. TrainingData.h. In addition to the information provided in the previous bullet TrainingData needs both data and label data thus inherits both classes.
- Overloaded functions allow the same function to operate on different parameters. File: main.cpp starting at line 90. PrepInputNNData can take 2 input data types TrainingData or just Data.

Memory Management

- The project makes use of references in function declarations. (At least two variables are defined as references, or two functions use pass-by-reference in the project code.) File: main.cpp starting at line 26 several functions pass data by reference.)
- The project uses smart pointers instead of raw pointers. (The project uses at least one smart pointer: `unique_ptr`, `shared_ptr`, or `weak_ptr`. The project does not use raw pointers.) File: main.cpp starting at line 44 uses shared pointers for some of the large data types. The project does not only use smart pointers for two reasons. The neural network in the project came from another project that did not use smart pointers and I did not what to change this code. In addition the Labels data structure labelData (main.cpp, 26) is a relatively small data structure so I thought the best place for it was on the stack rather than the heap.