

System Requirements

- Computer capable of running a shell script and compiling a C++ program.
- Most likely this is going to be in a terminal of some kind, each computer has its own slight variations on its terminal, but there should not be issues with compiling provided instructions were followed carefully
- Internet Access
- Currently need to be a member of St Olaf

Audience

In terms of making the most out of the interface we have created, advanced c++ knowledge is important and possibly necessary. If a user is more interested in simply witnessing the magic, not many skills are required. We have created code that would allow for all sorts of machine learning, but a user can choose to just look at the rather mesmerizing pre-made programs. The suite of functions and classes is also more of an ode to what can be done, rather than being all encompassing. For example, if you want a computer to play connect 4 as best as possible, using a back-propagating neural network like the one we designed is not the optimal solution. However, our project is meant more as a source of experimentation, insight and entertainment for people who have an good amount of interest in neural networks. Given that people who share this interest are usually technically savvy or at least have the ability to quickly learn how to install and compile simple programs, we deemed it appropriate to base the execution of our project in the terminal and have it be accessed through github. As this is a school project as well, it is currently only available for other members of St Olaf. However, this could be expanded by making our gitlab repository public and allowing our project to become open source. Currently though, there is benefit to keeping the audience small, as we know that St. Olaf students will be familiar with stogit and the methods for using files on stogit.

What is a neural network, anyway?

A neural network can be thought of like a rudimentary brain. Similar to the way that our eyes take in tons of little bits of light and are able to deduce what we are looking at, be that a dog or a computer, a neural net, at its most fine-tuned state, can take in certain data and notice the valid information that emerges from it. In another parallel, we don't exactly know what goes on in our brains, we just know that it successfully interprets the environment. Likewise, even a neural net that accomplishes the most basic of tasks is stunningly complicated. Sitting down and hard-coding a function that performs as well as a self-trained network would be an utter nightmare. Instead, a neural network uses a certain function to slowly optimize itself to be "more right" the next time. The algorithm our suite uses is called "gradient descent."

Gradient Descent

The word “gradient” may be familiar to anyone who has taken multivariable calculus. It’s actually the exact same function! For those unfamiliar, the gradient of a given function $f(x,y)$ returns a vector of size two that represents the needed changes to x and y in order to make the function “ f ” increase the fastest. The gradient of a function $f(x_1 \dots x_n)$ returns a vector of size n that represents the changes in every single one of those n x values that would, again, increase f the fastest.

But what on earth does this have to do with making a neural net learn? The answer lies in the fact that we can make a function that takes the entire network as a parameter. This function will return a value that represents how successful the network was at doing its job. The function that we used is called a “cost” or “error” function, and it returns a lower value for a better network. If we want to make the “cost” as low as possible, why don’t we find the gradient of the cost function (will return a huge vector that is the size of the network) and then just walk the opposite way that the gradient points? Remember that the gradient points in the direction of greatest initial increase. We made a function that calculates the gradient, and then we move backwards along it. That’s all it takes to slowly but surely make the network better. Fascinatingly, even when the data is variant and patterns aren’t catchable to a human, something emerges in the robot. In terms of one bot that classifies a sentence as coming from Mary Shelley’s *Frankenstein* or Homer’s *The Odyssey*, success rates at one point hit 980 correct guesses out of 1000.

Access

To access our project, visit our repository at:
<https://stogit.cs.stolaf.edu/malek1/our-awesome-project>

From there, you will want to clone the repository to a directory of your choice, then make all files to compile all code into executables. To run the the finished version of our code, run the executable named presentation. Here is a more detailed explanation if you have questions still.

First click on the big blue button that says clone on the top right of the webpage for our repository (the url starting with <https://>). Two pieces of copyable text should appear, one labeled Clone with SSH the other Clone with Https. Copy the text given for Clone with SSH. That text should read:

[git@stogit.cs.stolaf.edu](https://stogit.cs.stolaf.edu):malek1/our-awesome-project.git

Now you want to open the terminal on your computer, and navigate to the directory that you would like to put this project in. (use the command `cd` to navigate) Next, you need to clone the files. Ente the following text into your terminal:

```
git clone ssh:
```

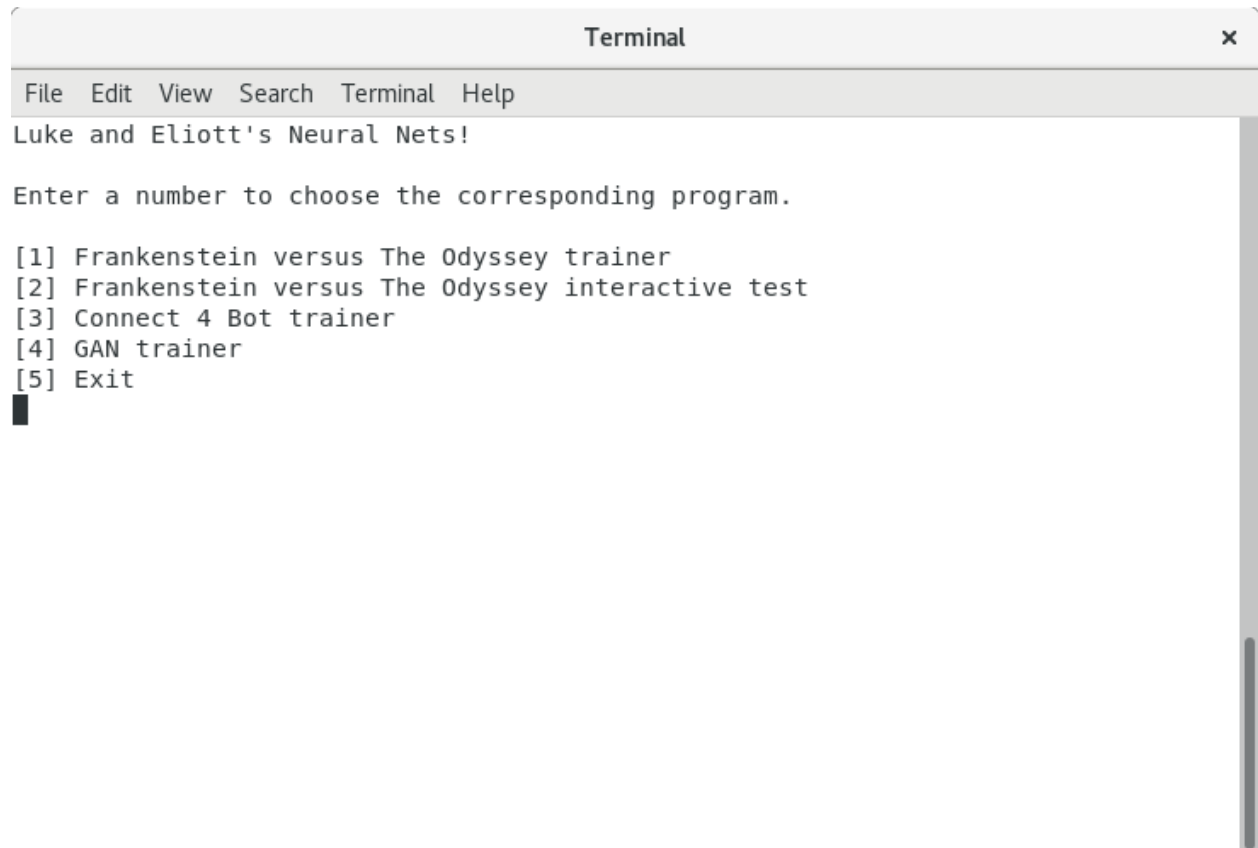
Now paste the text from Stogit (it begins with git@stogit.cs ...)

Now hit enter. You should see a bunch of things going on and see many new files in your directory.

All that is needed to run the program in to type in

```
./presentation
```

Into the terminal. Hit enter. There should appear some text on the screen that looks like this:



```
Terminal
File Edit View Search Terminal Help
Luke and Elliott's Neural Nets!

Enter a number to choose the corresponding program.

[1] Frankenstein versus The Odyssey trainer
[2] Frankenstein versus The Odyssey interactive test
[3] Connect 4 Bot trainer
[4] GAN trainer
[5] Exit
█
```

The text will guide you as to what to enter for the various different parts. If it asks you to enter a number or a phrase, type the number or phrase as it is written and then hit enter to make that selection. Enter numbers 1 through 5 to select the different programs. Programs will look like

```
Terminal
File Edit View Search Terminal Help
Luke and Elliott's Neural Nets!

Enter a number to choose the corresponding program.

[1] Frankenstein versus The Odyssey trainer
[2] Frankenstein versus The Odyssey interactive test
[3] Connect 4 Bot trainer
[4] GAN trainer
[5] Exit
1

The cruel wretch vouchsafed me not one word of answer, but with a sudden clutch
he gripped up two of my men at once and dashed them down upon the ground as thou
gh they had been puppies.
0.0857915 Number correct out of 1000: 967
0.0971275 Number correct out of 1000: 965
0.108288 Number correct out of 1000: 957
0.0865366 Number correct out of 1000: 971
0.103696 Number correct out of 1000: 956
0.0945261 Number correct out of 1000: 965
0.0966052 Number correct out of 1000: 965
0.095012 Number correct out of 1000: 963
█
```

this:

```
Terminal
File Edit View Search Terminal Help
Luke and Elliott's Neural Nets!

Enter a number to choose the corresponding program.

[1] Frankenstein versus The Odyssey trainer
[2] Frankenstein versus The Odyssey interactive test
[3] Connect 4 Bot trainer
[4] GAN trainer
[5] Exit
2
HomerVShelly2 net tester! I work best when inputting sentences with a bit of bul
k to them (25+ chars)
Enter a phrase : hey will this work?
I think that is frankenstein-y! Was I right? (y/n)
0.978032 0.0216987
█
```

```

Terminal
File Edit View Search Terminal Help
|      X      X      |
| 0      0      0      |
| 0      0      X      |
| 0 X    X      0      X |
| 0 X    0 0  X      X |
|-----|
| X          |
| 0 0 0 0    |
| 0 X 0 X 0   |
| 0 X X 0 X 0  |
| X X 0 X 0 X X |
|-----|
|          0   |
| 0      0      0   |
| 0      X      0   X |
| X      0  X  X  X  X |
|-----|

```

```

Terminal
File Edit View Search Terminal Help
[2] Frankenstein versus The Odyssey interactive test
[3] Connect 4 Bot trainer
[4] GAN trainer
[5] Exit
4
tsoil detmieh edation wsbh soatet eoi lrmenwt (0.7 discriminated)
tsoil detmieh edation wsbh soatet eoi lrmenwt (0.7 discriminated)
sahilodetmiehc edation wsbh soatet eoi lrmenwt (0.733333 discriminated)
sahil detmieec edation wsbh soatet eoi oemenwt (0.75 discriminated)
tsoilodetmieec edation wsbh soatet eoi lrmenwt (0.73 discriminated)
tahil betmieh edation wsbh soatet eoi lemenwt (0.733333 discriminated)
tahil detmieec enation wsbh soatet eoi oemenwt (0.75 discriminated)
tshil detmiehc enation wsbh soatet eoi lemenwt (0.74375 discriminated)
tlhil detmiehc enation wsbh soatet eoi oemensh (0.75 discriminated)
tahil bttmieh enation wsbh soatets eoi lemenwh (0.765 discriminated)
tlhil betmiehc enation wsbh soatets eoi oemenwh (0.763636 discriminated)
tlhilobetmieec endtion wsbh soatet eoi oemenwh (0.7625 discriminated)
tlhil betmiehc endtion wsbh soatet eoi lemenwt (0.757692 discriminated)
tshilo etmiee enation wsbh soatet eoi oemenwt (0.764286 discriminated)
tshilobttmiehc enation wsbh soatet eoi oemensh (0.76 discriminated)
tshel betmieec endtion wsbh soatets eoi oemenwh (0.753125 discriminated)
tlhelobttmiehc enation wsbh soatets eoi oemensh (0.755882 discriminated)
ilhil bttmiehc endtion wsbh soatets eoi oemensh (0.755556 discriminated)
t,hel ttmiehc endtion wsbh soatets eoi oemenwh (0.755263 discriminated)

```

You can always quit the program at any time by pressing ctrl and c at the same time, and you can always start the program again after it has stopped by typing `./presentation` and hitting

enter, provided that the executables have been compiled correctly once before, and you are in the correct directory. (you can check your directory using the command `pwd`)

Feedback

While these instructions give you access to a way to see the final implementation of our neural network, there is much more to our project. The benefit of sharing our project via git is that anyone can take a look into both how we built our network and how we implemented it, and learn from it, provide feedback to us, or improve it themselves. We encourage anyone who uses our project to do at least a little digging! The network is contained in `Neural_net.cpp` and `Neural_net.h`, with supporting files `net_fun.h` and `net_fun.cpp`. Then there are a variety of implementation programs within our repository, which typically follow a pattern of having three files, a `.cpp` version, a `.h` version and a `driver.cpp` version. The actual values of the various networks used by the implementation programs are stored in files with the suffix `.net` and are usually named something related to what they are trying to accomplish. Each implementation program will have at least one network, and if you would like to check which belongs to which, the best way is to look in the implementation files. Feel free to create more `.net` files if you would like to try different networks but still store the ones that we had built. Any feedback is greatly appreciated and it would be awesome to see this project become open source! You can suggest changes or post information in the issues tab of the stogit webpage.

Ethical Considerations and Terms of Use

With our project in its current state, there are few real dangers of serious harm that it could cause. The closest thing would be the addiction of watching the program learn for hours and taking up a large part of the user's time. However, with the possibility for our project to grow and the fact that we have made the base level code accessible to anyone, there is a real possibility that it could theoretically be expanded many times and for any purpose, and some of those could be nefarious. Just like we wrote a program that used the base structure of `Neural_net.cpp` to do harmless things like connect or identify sentences from books, someone else could write an implementation program using it to make facial recognition software that could invade the privacy of countless people. Also, for our training data, we used only data that had been available already on the internet in the form of text documents, but those text documents were books and the intellectual property of their authors, and should have been credited or paid for. When training AI, a big question is finding datasets. For example, many networks take in pictures and many pictures from the internet are not available for anyone to use without permission. Future users of our network must agree to only use training data that has been cleared for their use.

Furthermore, a more relevant question given our current implementation programs is the issue of the property rights of a machine. We created a program that actually creates its own

words, and thus the question arises of who those words belong to. One could argue that the program creators own them, while others could argue that the machine made them and thus it owns them. You could even argue that the people who wrote the training data have at least partial ownership of anything the computer produces. This isn't a huge issue currently as our program produces close to gibberish, but if it were expanded and in the future wrote a best selling book, the question of who gets the profit would be important to consider.

There is a question of access with our program given that there is fair bit of technical knowledge needed to use it, but we have done our best to make the instruction manual simple and explicit enough for almost all people to understand. Since a neural network has such a broad range of possible applications, there could be many issues of honesty and deception where our network could be used to create fake identities or trick people into believing they were talking to a real person, or risk and reliability, if our program were used to make decisions about the trajectory of a spaceship. There are even questions about our program negatively impacting the quality of life for all humans, it could one day be used to kill all humans or enslave them if it went rogue. Giving computers the power to change themselves with only an end goal in mind can have some very dangerous implications, but in reality this thing can barely play a simple board game and has neither access to the internet nor the physical world at this point, so we very much doubt that it will be the AI that becomes Skynet.