CSCE 315-100
Dr. Daugherity

Project 2 Team 6 Report

Introduction:

In project two, our team set out to create a program that could identify capital letters in a 5x7 matrix. In order to do this we needed to implement a neural network. A neural network is an artificial intelligence that handles very complex problems, and noise in input very well. This project is significant because it introduces us to important concepts in Artificial Intelligence, which is a very up and coming field of computer science, and has applications in pretty much every industry imaginable.

Restrictions and limitations:

The biggest restriction and limitation we ran into for this problem was being restricted to using java. Our team was mostly familiar with the language Python, and we had found several resources for neural networks written in Python.

The approach:

For the neural network, I wanted to make a deep learning network that would be very modular. I decided to write it using matrices, so that the network could be trained on one sample, or multiple samples at once. I decided to make the neural network have an input layer that would receive 35 input values, corresponding to each space on a 5x7 grid. The input would then be passed though two hidden layers, one that had 32 nodes, and one that had 16 nodes. The final output would then be passed into a 26-element array, which would show the confidence that our model believes each letter is. Initially we only had a single layer neural network, but we decided that we would need two layers so that our model could handle a problem with this complexity.

In order to work with our neural network, instead of taking one big matrix with the weights, we used the forward propagation we already had and calculated the errors on each layer and delta individually. We then updated the weights at each layer following the following the calculations provided in the pseudocode. So we didn't use the forward propagation pseudocode provided in the algorithm.

The approach for the GUI was to be simple, but usable. I accomplished this by looking at oracles javafx documentation and finding a way to make a GUI with several separate sections. I used these different sections to hold different parts such as the input grid, the instructions, the buttons to evaluate and reset, and the results.

Results and analysis:

We never really messed with different numbers of layers or nodes in each layer, as we saw much more potential in training the network, over changing the arbitrary layout of the network in achieving the best results possible. We were initially going to implement the Broyden-Fletcher-Goldfarb-Shanno algorithm in our back propagation code. With this quasi-newton algorithm our training model would not have been prone to falling into local min's, because the BFGS algorithm computes and deals with the 2nd derivative and would have been able to converge to the global min on a given input. We also initially wanted to add in a regularization factor so that our training would not be prone to over fitting, or falling into local min's when training. During the training process we changed the learning rate for a fixed amount of iterations, and saw that typically a smaller learning rate led to more accurate results, but that too small of a learning rate gave us garbage outputs. We also played with the iterations on the training, and for lower numbers of iterations, we did not really see much improvement in the model, but around 100,000 iterations, the model would produce pretty good results. We tested the trained model on the exact letter values, but also introduced some noise, and overall the model was able to handle the noise and still recognize what letter we were trying to show.

Conclusions:

We were able to implement a deep learning neural network, and a GUI that gave input to the neural network, and displayed the top 5 letters that the model thought the user had input. We also had written a back propagation algorithm and trained the neural network to a fair extent on the given input. We learned how neural networks operate, how to implement them, and design decisions to consider when putting everything together. It was also a very good lesson in team communication, and coordinating a project from start to finish.

Future research:

In order to make our model more accurate, more research would need to be done on different types of training, and more specifically how the learning algorithm works. We would look into different types of batch style learning or different methods of gradient descent in order to improve our model.

Instructions on how to run your program:

If you are on one of the Windows Lab computers, make sure you are running XMING, and have x-11 forwarding enabled in Putty.

COMPILE COMMAND:
Navigate to directory and type "javac -cp /usr/java/jre1.8.0_131/lib/ext/*:/usr/java/jre1.8.0_131/lib/* Project2GUI.java" and then type "java Project2GUI"

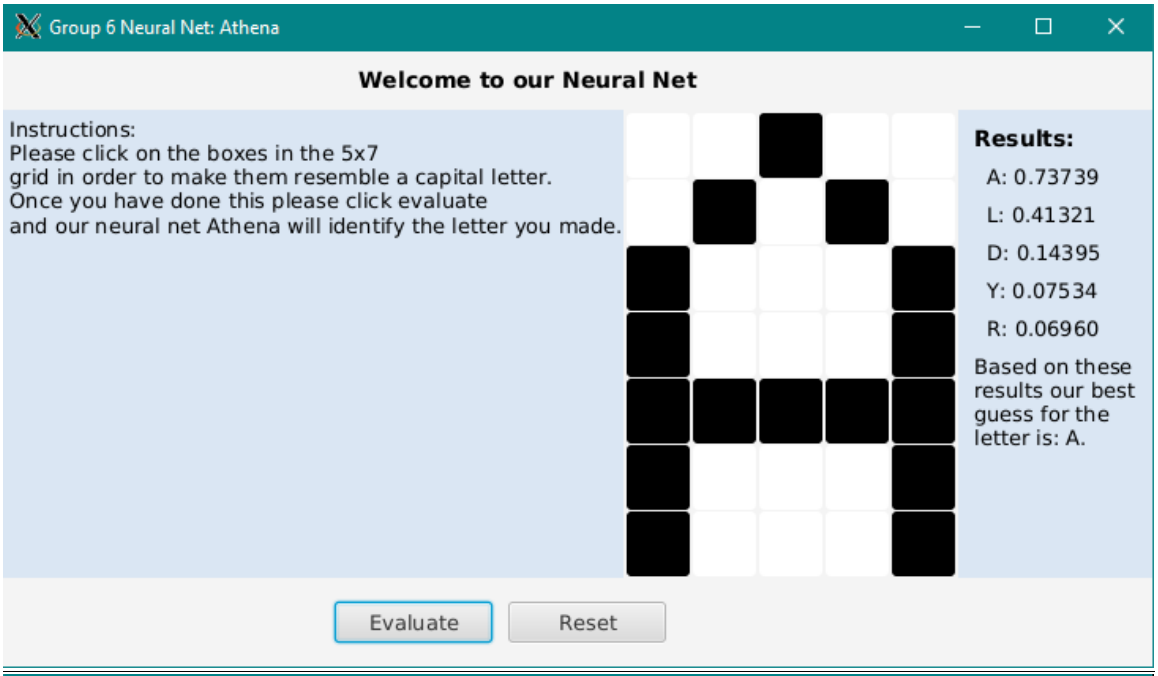
Sources:

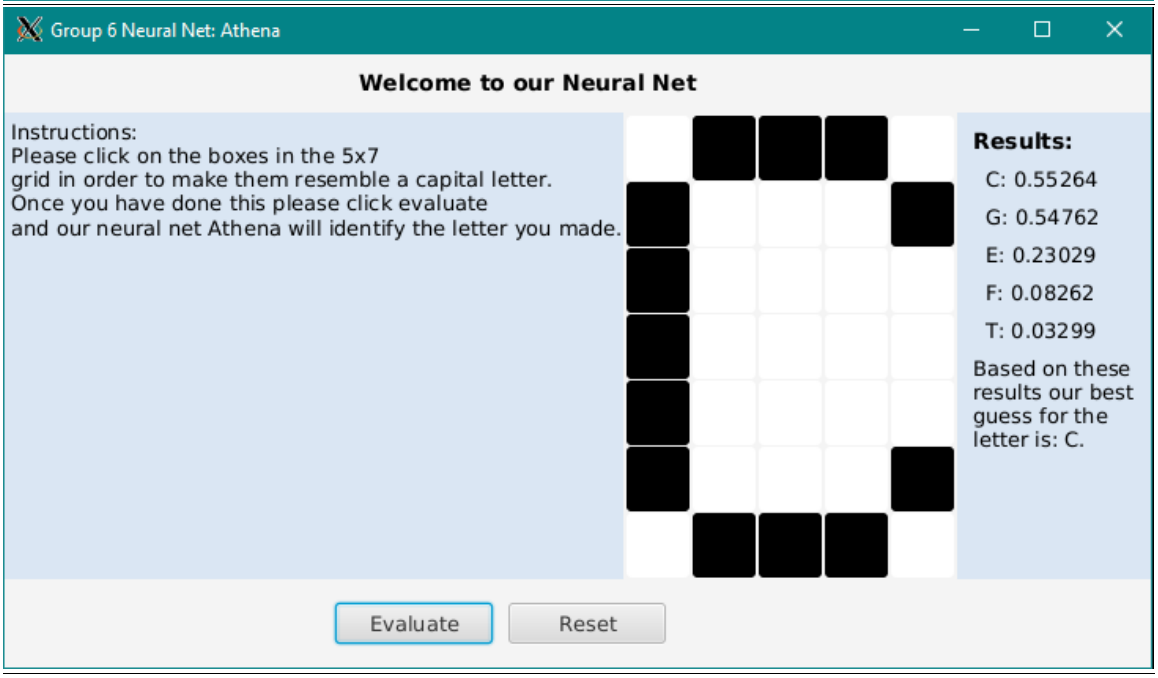
-Stackoverflow.com

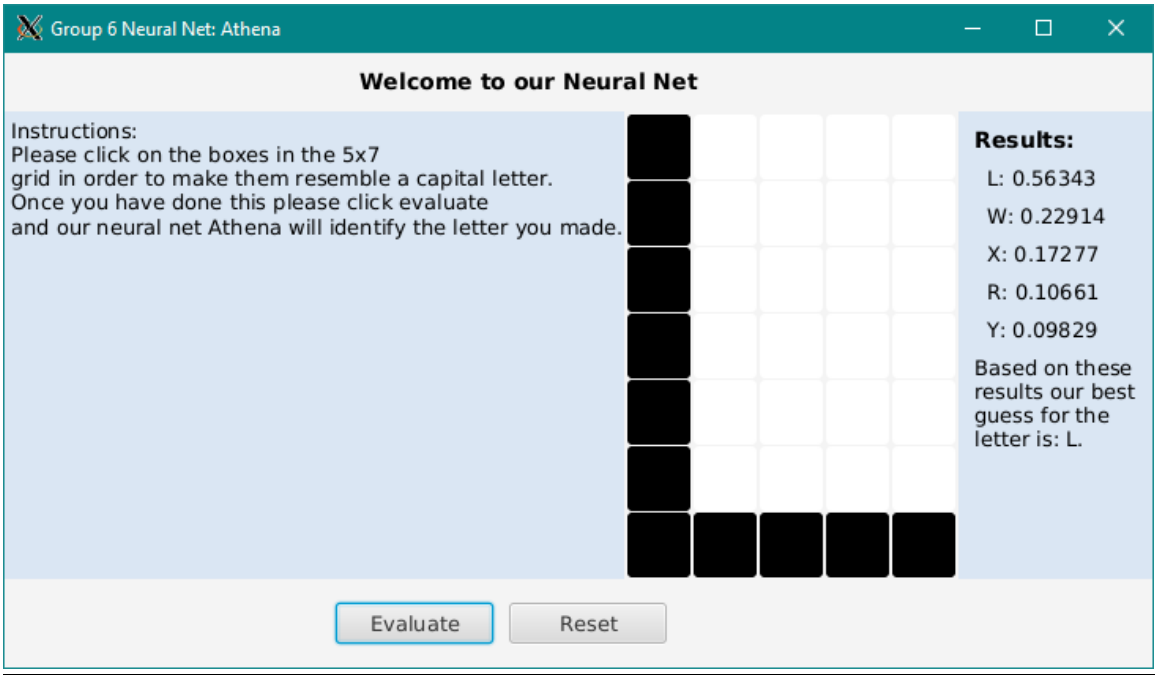
-Neural Networks Demystified by Welch Labs from Youtube.com
-https://docs.oracle.com/javase/tutorial/getStarted/cupojava/netbeans.html
-http://download.java.net/jdk8/jfxdocs/javafx/scene/doc-files/cssref.html
-http://docs.oracle.com/javase/8/javase-clienttechnologies.htm

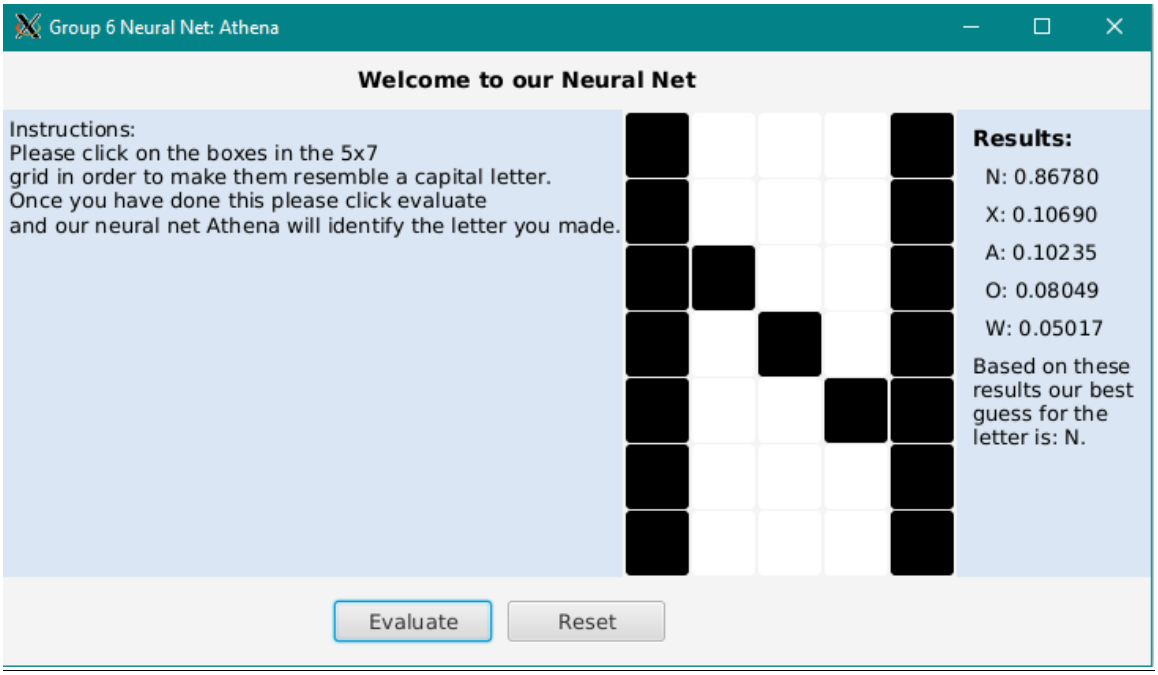
Screenshots of Testing:

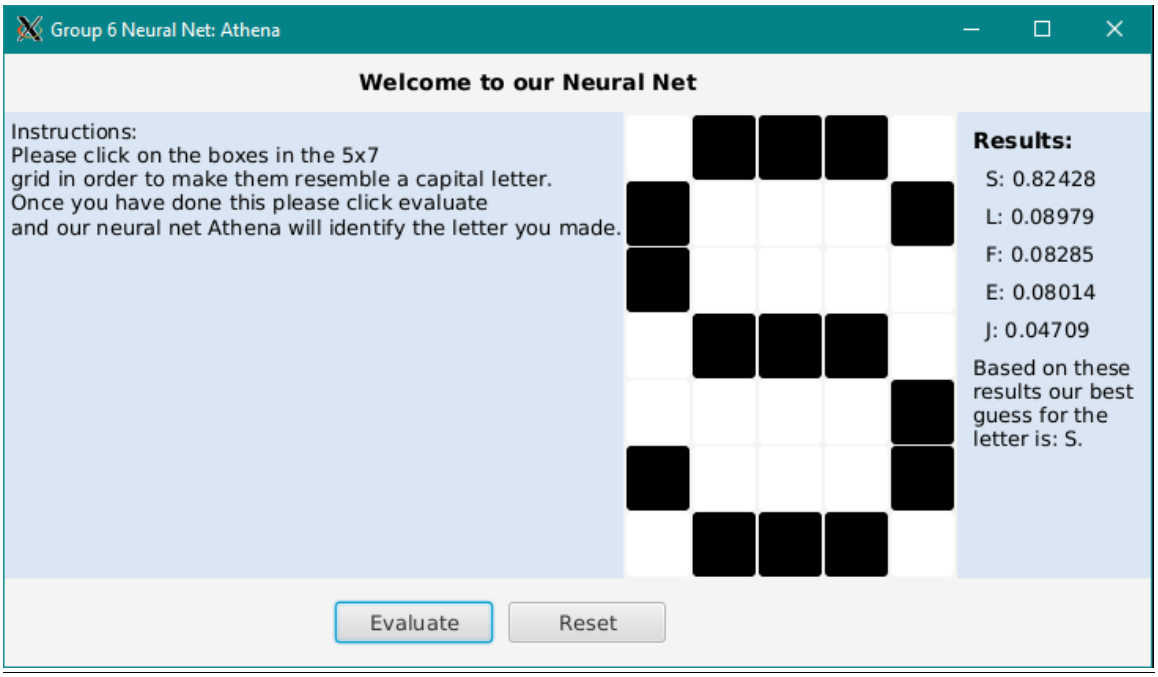
Included below are some sample screenshots of the trained neural network running some tests.






Group 6 Neural Net: Athena
Welcome to our Neural Net
Instructions:
Please click on the boxes in the 5x7
grid in order to make them resemble a capital letter.
Once you have done this please click evaluate
and our neural net Athena will identify the letter you made.
Results:
L: 0.56343
W: 0.22914
X: 0.17277
R: 0.10661
Y: 0.09829
Based on these results our best guess for the letter is: L.
Evaluate
Reset


Group 6 Neural Net: Athena
Welcome to our Neural Net
Instructions:
Please click on the boxes in the 5x7
grid in order to make them resemble a capital letter.
Once you have done this please click evaluate
and our neural net Athena will identify the letter you made.
Results:
N: 0.86780
X: 0.10690
A: 0.10235
O: 0.08049
W: 0.05017
Based on these results our best guess for the letter is: N.
Evaluate
Reset



Post Production Notes:

Changes we had to make to the design and why:

Neural Network:
I needed to change my initial design from ArrayLists to matrices to handle the learning algorithm and the GUI. I had to redesign how the weights and layers interacted with each other in order to be more modular to account for any design changes elsewhere in the project.

GUI:
Overall, the GUI that we were left with at the end of the project is pretty much how I envisioned it from the start. I made some small changes such as instead of having two screens, one for the input and one for the results, I kept the input and results on one screen. I also had to tweak some things to make the input and output work with the other parts of the project my partners were creating.

Back Propagation:

Originally we had thought of implementing the neural network using arrays but this was changed to vectors since they were a little easier to work with logically. We originally had everything computing once in one for loop but we found it was easier to work on individual errors and deltas for debugging purposes and to understand the concept better.

Difficulties

Neural Network: The most difficult thing I had to consider when writing the neural network was to consider how the layers would interact with each other to get the input data to the output layer, and back again for the learning algorithm.

GUI: The biggest difficulty I had to overcome was due to the fact I had no experience with java. I picked up java fairly quickly though due to its similarity to c++ in syntax. After this, the real difficulty came with learning the ins and outs of javafx and familiarizing myself with CSS. After I learned how to properly design with javafx the whole GUI came together fairly quickly.

Back Propagation: Since this was a new concept it was difficult to understand the concept behind back propagation and neural networks being that we only had one lecture in class about this concept. It was also also quit challenges picking up a new language since I had no experience working with java.

Solutions

Neural Network: I overcame the difficulties by starting over and designing the model to fit better to the specifications of the rest of the project. I had to sit down and come up with a way to account for different sized matrices, different amounts of layers, and different amounts of sample sizes so that our neural network would be able to handle a wide variety of design choices made elsewhere in the project.

GUI: I overcame all of the difficulties in working with a new and unfamiliar language by reading the documentation and trial and error. Oracle actually had surprisingly good documentation for javafx which made everything easier. The biggest solution I had to come up with was to make the toggle buttons switch colors when pressed. I accomplished this using CSS.

Back Propagation: To fix this problem, we watched several videos about neural networks and and read some stuff about it online which was helpful in given us a general understanding of what was required. For the back propagation part I reached out to people who had understood the concept to help understand that. I then wrote the back propagation algorithm in a language that I was familiar with and met with my partner Rick and I then translated it to Java with the correct syntax so that it works with our project since he was better with the language than I was.

Lessons Learned

Neural Network: I learned to better communicate and plan before writing out my part of the project to be too specific for the rest of the project to be successful.

GUI: I learned throughout this project many very important lessons. First and foremost I learned how to design a decent looking and functional GUI using javafx. Along with that I learned how to program in java. Other lessons I have learned are how to function with a programming team and some of the principles about neural networks and artificial intelligence.

Back Propagation: In the processing of find out out how Neural networks work, this introduced me to Artificial Intelligence and its capabilities in the future with computers being able to learn complex tasks making this something I would look into for the future.

Approach

Neural Network: For the neural network, I wanted to make a deep learning network that would be very modular. I decided to write it using matrices, so that the network could be trained on one sample, or multiple samples at once. I decided to make the neural network have an input layer that would receive 35 input values, corresponding to each space on a 5x7 grid. The input would then be passed though two hidden layers, one that had 32 nodes, and one that had 16 nodes. The final output would then be passed into a 26-element array, which would show the confidence that our model believes each letter is.

GUI: The approach for the GUI was to be simple, but usable. I accomplished this by looking at oracles javafx documentation and finding a way to make a GUI with several separate sections. I used these different sections to hold different parts such as the input grid, the instructions, the buttons to evaluate and reset, and the results.

Back Propagation: In order to work with our neural network, instead of taking one big matrix with the weights, we used the we the forward propagation we already had and calculated the errors on each layer and delta individually. We then updated the weights at each layer following the following the calculations provided in the pseudocode. So we didn't use the forward propagation pseudocode provided in the algorithm.

Consensus Individual workload distribution:

We did not reach a consensus on the workload distribution. We plan to meet with Dr. Daugherity to discuss this further.