COVID-19 Risk Neural Network

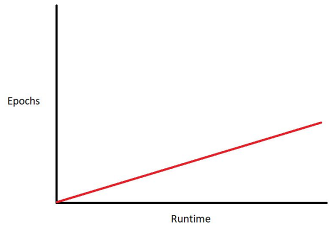
Hrishikesh Bagalkote
Gwinnett School of Mathematics, Science, and Technology

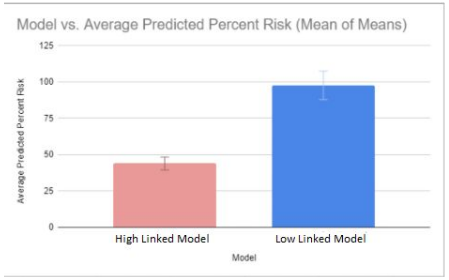
Quad Chart Overview

Research Question: How can we design a neural network to simplify disease forecasting on lower end hardware?

Data/Analysis & Results

- Algorithm optimization allows runtime to increase at a linear rate
- Low linked model produces statistically significant higher risk profiles than the high linked model.





Methodology/Design

- Brainstorm possible input features for training dataset.
- Create holistic training dataset and update data periodically.
- Refactor and test neuron using runtime data.
- Log runtime data and begin to analyze its relationship with epochs.
- Derive high/low sensitivity feature linked models from holistic dataset.
- Analyze risk profiles of each predictive model.

Interpretations/Conclusions

- the training algorithm turned out to be extremely lightweight, being able to complete 1 million epochs in under a tenth of a second.
- In addition, the time complexity of the algorithm was linear, having constant runtime gains as more epochs pass.
- The statistical difference in predicted risk between individual models show that one can change the training dataset in order to create a model that fits their task at hand.
- We believe the research can be used as a baseline attempt at accelerated disease forecasting using machine learning.

Research Question or Engineering Problem/Goal

Engineering Goal: To design a neural network that forecasts relative percent COVID-19 risk in a geographical area based on input factors. Such an algorithm will be useful in situations where there is not enough present data to conclusively determine COVID-19 risk.

Existing Accomplishments: Currently, machine learning and neural network algorithms are being used for tasks like COVID-19 diagnosis based on images of a patient's lungs. Machine learning models like thes have been created for forecasting with the rise of the UK COVID-19 variant, but do not make it easy to reproduce a model or create a new model based on new factors.

Advantages of the Software: the neural network, algorithm, and training process make it easy for people to construct their own predictive models based on new factors. In addition, the training algorithm is extremely lightweight. This allows millions of training cycles to be ran in just a few milliseconds.

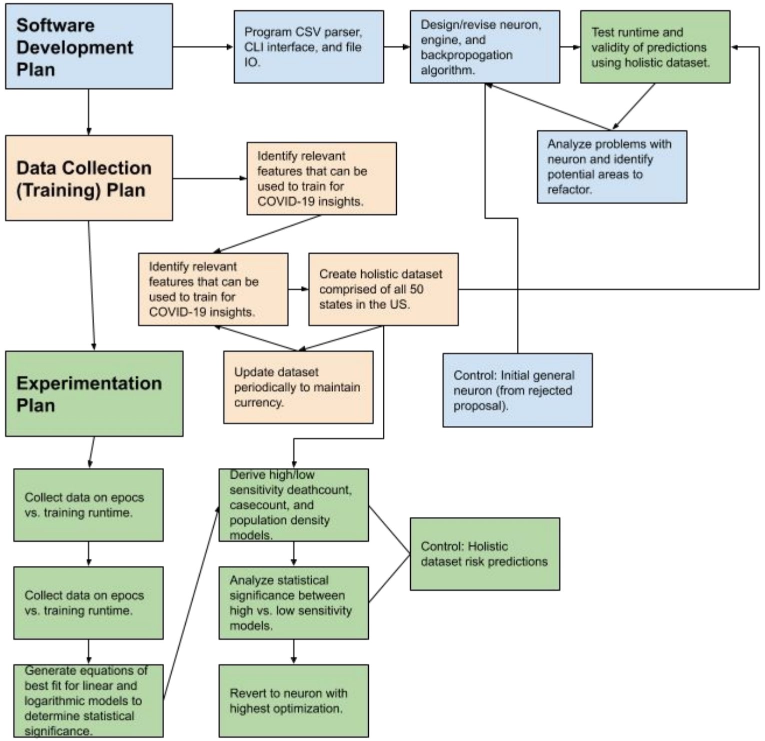
Specific Goals:

- Create an insight map for all 50 states in the US ranked by relative predicted COVID-19 risk.
- Create an insight map for Georgia counties ranked by relative predicted COVID-19 risk.
- Design a training algorithm that is fast enough to complete one million epochs in less than a second.

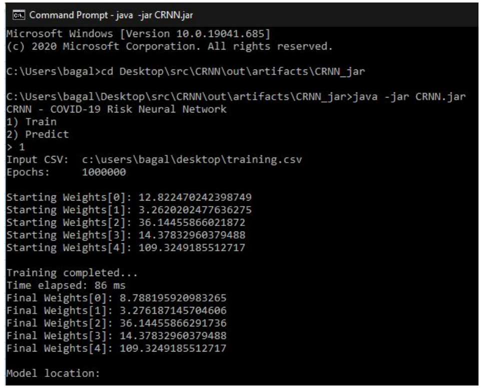
Future Research Prospects:

- Create a website or application that forecasts future COVID-19 risk of geographical areas through the automatic retraining of models.
- Optimize the training algorithm for general machine learning tasks, such as classification.

Methods

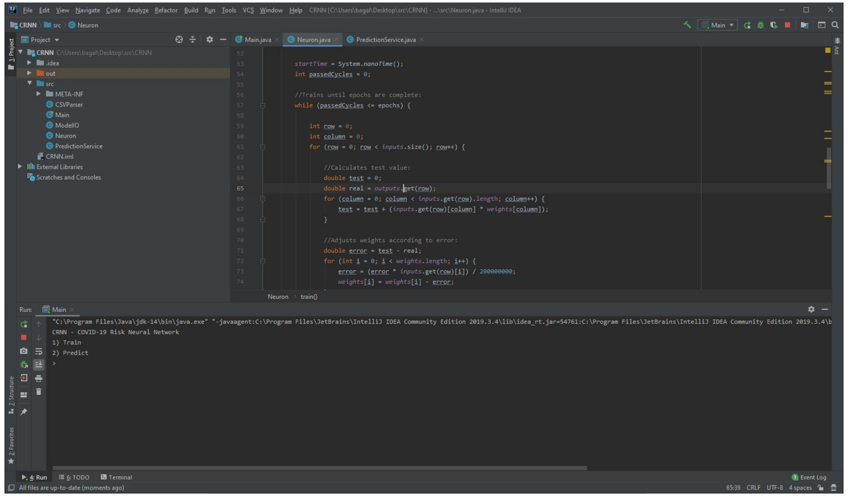


Software Development Plan
Program CSV parser, CLI interface, and file IO.
Design/revise neuron, engine, and backpropogation algorithm.
Test runtime and validity of predictions using holistic dataset.
Analyze problems with neuron and identify potential areas to refactor.
Data Collection (Training) Plan
Identify relevant features that can be used to train for COVID-19 insights.
Identify relevant features that can be used to train for COVID-19 insights.
Create holistic dataset comprised of all 50 states in the US.
Update dataset periodically to maintain currency.
Control: Initial general neuron (from rejected proposal).
Experimentation Plan
Collect data on epocs vs. training runtime.
Collect data on epocs vs. training runtime.
Generate equations of best fit for linear and logarithmic models to determine statistical significance.
Derive high/low sensitivity deathcount, casecount, and population density models.
Analyze statistical significance between high vs. low sensitivity models.
Revert to neuron with highest optimization.
Control: Holistic dataset risk predictions



```
Command Prompt - java  -jar CRNN.jar
Microsoft Windows [Version 10.0.19041.685]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\bagal>cd Desktop\src\CRNN\out\artifacts\CRNN_jar

C:\Users\bagal\Desktop\src\CRNN\out\artifacts\CRNN_jar>java -jar CRNN.jar
CRNN - COVID-19 Risk Neural Network
1) Train
2) Predict
> 1
Input CSV:  c:\users\bagal\desktop\training.csv
Epochs:     1000000

Starting Weights[0]: 12.822470242398749
Starting Weights[1]: 3.2620202477636275
Starting Weights[2]: 36.14455866021872
Starting Weights[3]: 14.37832960379488
Starting Weights[4]: 109.3249185512717

Training completed...
Time elapsed: 86 ms
Final Weights[0]: 8.788195920983265
Final Weights[1]: 3.276187145704606
Final Weights[2]: 36.14455866291736
Final Weights[3]: 14.37832960379488
Final Weights[4]: 109.3249185512717

Model location:
```

Training Sequence Running Locally Via Terminal (11/11/2020)



File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help CRNN [C:\Users\bagal\Desktop\src\CRNN] - ...\src\Neuron.java - IntelliJ IDEA
CRNN src Neuron
Project
Main.java
Neuron.java
PredictionService.java
CRNN C:\Users\bagal\Desktop\src\CRNN
.idea
out
src
META-INF
CSVParser
Main
ModelIO
Neuron
PredictionService
CRNN.iml
External Libraries
Scratches and Consoles
Neuron › train()
Run: Main
Run
TODO
Terminal
Event Log
All files are up-to-date (moments ago)

```java
        startTime = System.nanoTime();
        int passedCycles = 0;

        //Trains until epochs are complete:
        while (passedCycles <= epochs) {

            int row = 0;
            int column = 0;
            for (row = 0; row < inputs.size(); row++) {

                //Calculates test value:
                double test = 0;
                double real = outputs.get(row);
                for (column = 0; column < inputs.get(row).length; column++) {
                    test = test + (inputs.get(row)[column] * weights[column]);
                }

                //Adjusts weights according to error:
                double error = test - real;
                for (int i = 0; i < weights.length; i++) {
                    error = (error * inputs.get(row)[i]) / 200000000;
                    weights[i] = weights[i] - error;
```

```
"C:\Program Files\Java\jdk-14\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2019.3.4\lib\idea_rt.jar=54761:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2019.3.4\b
CRNN - COVID-19 Risk Neural Network
1) Train
2) Predict
>
```

Screenshot of Neuron Sthece Code With Network Running in IDE (12/22/2020)

Results (Tables)

Highest vs. Lowest Deathcount Model Statistics:

Trial	Highest 20 Deathcount Average Percent Risk in Trial	Lowest 20 Deathcount Average Percent Risk in Trial
1	50.8142238	153.2567767
2	57.1299558	126.5063073
3	37.61708443	89.47228216
4	45.87349575	63.83847371
5	35.2692335	98.54956186
6	41.96393818	60.99815868
7	48.3331354	121.2244072
8	45.65122952	74.26598997
9	45.84826521	131.2229748
10	30.78568241	57.58090632
Mean of Means	43.9286244	97.69158387
T Test (p) Value	6.38E-04	
Reject Null Hypothesis?	Yes	
df	9	
Critical Value (p value cutoff)	0.01	

Trial	Highest 20 Casecount Average Percent Risk in Trial	Lowest 20 Casecount Average Percent Risk in Trial
1	50.9505758	162.154531
2	66.1350466	155.0401611
3	40.33867067	103.2852094
4	43.11123286	81.64784681
5	53.48166754	120.1175272
6	47.32847571	85.70105183
7	51.51106743	132.3809187
8	56.5932893	109.3522194
9	41.35799654	157.5939969
10	35.20407815	67.7042858
Mean of Means	48.60121006	117.4977748
T Test (p) Value	8.68E-05	
Reject Null Hypothesis?	Yes	
df	9	

Highest vs. Lowest Population Density Model Statistics:

Trial	Highest 20 Population Density Average Percent Risk in Tria	Lowest 20 Population Density Average Percent Risk in Trial
1	52.07231586	219.4504697
2	53.59747298	488.399429
3	36.38223421	227.576314
4	36.05951393	254.5348118
5	44.84551887	351.3613694
6	35.50982698	359.233512
7	46.68497236	294.3919532
8	44.41208336	406.4147313
9	41.78861631	205.0260956
10	31.88354276	156.2678636
Mean of Means	42.32360976	296.265655
T Test (p) Value	2.71E-05	
Reject Null Hypothesis?	Yes	
df	9	
Critical Value (p value cutoff)	0.01	

The data to the left was collected to determine significance between the following: highest and lowest 20 deathcount models, highest and lowest 20 casecount models, and highest and lowest 20 population density models. The T-test (p values) are all below 0.01. This allows us to confirm significance between a high and low centered models of a respective feature. The models we tested were high/low (top/bottom 20) deathcount, casecounts, and population density data points. The T-test values shown suggest that the engineering goal was met. The statistical significance between high vs. low positive factor models show that one can easily create a predictive model using the algorithm to best fit their needs.

Linear vs. Logarithmic Line of Best Fit:

Linear (equation: 4.91e-5x + 10.2)

Test (Epochs in Millions)	Average Real Runtime (ms)	Modeled Runtime Through Equation(ms)	Percent Error
10	305.117	501.2	39.12270551
11	354.915	550.3	35.50517899
12	361.026	599.4	39.76876877
13	376.192	648.3	41.97254358
14	402.389	697.6	42.3180906
15	445.002	746.7	40.40417838
Average Percent Error:	39.84857764		
T Test (p) Value:	0.0004575532681		
Reject Null Hypothesis?	No		
df:	5		
Critical Value (p cutoff):	0.01		

Logarithmic (equation: -54.9 + 7.34lnx)

Test (Epochs in Millions)	Average Real Runtime (ms)	Modeled Runtime Through Equation(ms)	Percent Error
10	305.117	63.407	381.203968
11	354.915	64.106	453.637725
12	361.026	64.745	457.6121708
13	376.192	65.333	475.8070194
14	402.389	65.877	510.8186469
15	445.002	68.383	550.7494553
Average Percent Error:	471.6381642		
T Test (p) Value:	0.00001688502351		
Reject Null Hypothesis?	No		
df	5		
Critical Value (p cutoff):	0.01		

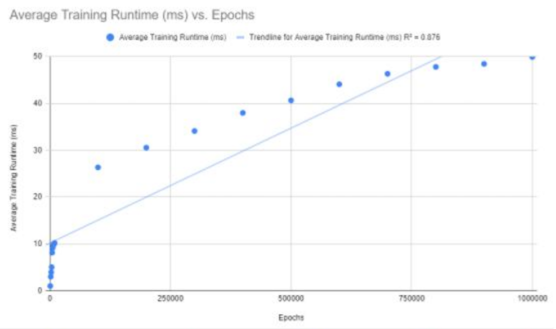
The above data was collected for verification purposes of the neural network's optimization. The conclusions show that the neural network's runtimes are shorter than the linear equation's expected values, but longer than the logarithmic equation's expected values. However, the p values and average percent error indicate that the linear equation of best fit is far more statistically similar to real runtime values.

Epochs (Training Cycles) vs. Average Training Runtimes (ms)

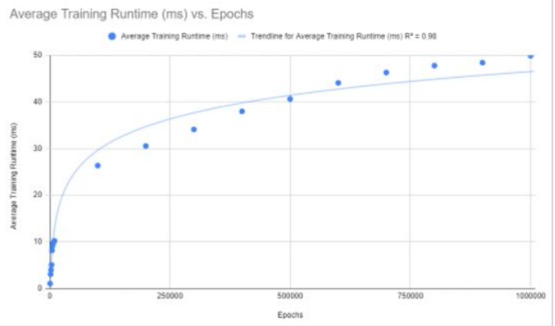
Epochs	Average Training Runtime (ms)
1000	0.998
2000	2.991
3000	3.952
4000	5.022
5000	8.119
6000	9.122
7000	9.671
8000	9.884
9000	9.959
10000	10.187
100000	26.34
200000	30.552
300000	34.102
400000	37.981
500000	40.649
600000	44.101
700000	46.334
800000	47.812
900000	48.445
1000000	49.918

The above table is processed data. Average training runtime values are average trial results, rather than singular trial points. The goal of this portion of data collection was to find the relationship between the number of epochs (training cycles) and the time it takes to complete training.

Results (Graphs)

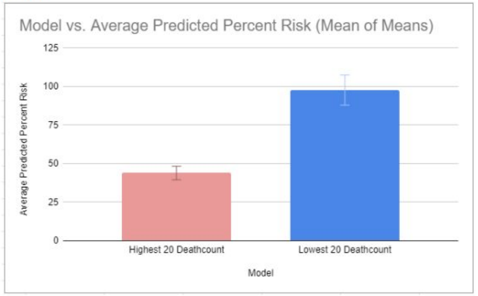


Although linear regression can be shown, a line of best fit does not best represent the data shown in the graph.

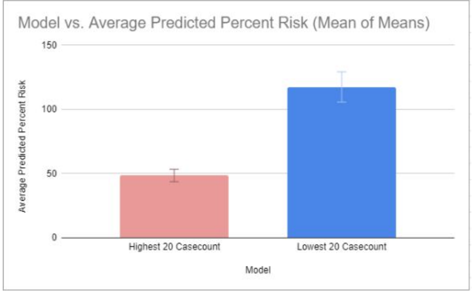


The above graphs show the comparison between the linear equation of best fit and logarithmic equation of best fit for the relationship between epochs and average training runtime. Although the logarithmic equation of best fit appears to fit the data more tightly, the linear equation is more representative of the real data, which was proved in the T-tests shown in the previous slide.

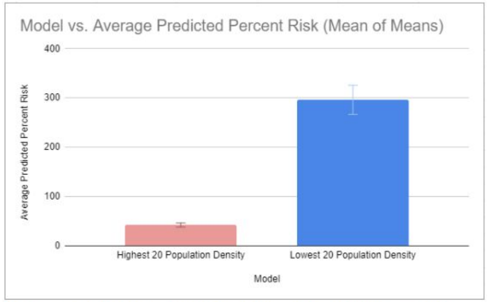
Average Predicted Percent Risk per Trial in Highest and Lowest 20 Deathcount Models:



The lowest 20 deathcount model produces significantly higher risk percentages than the highest 20 deathcount model.



The lowest 20 casecount model produces significantly higher risk percentages than the highest 20 casecount model.



The lowest 20 population density model produces significantly higher risk percentages than the highest 20 casecount model.

Interpretation of Results

Statistical Reasoning:
Epochs vs. Runtime: As stated before, the linear equation of best fit better represented the relationship between epochs and runtime than the logarithmic equation of best fit. Since the linear equation of best fit had a higher p value when being compared to real runtime values, it was less statistically different and varied than the logarithmic equation of best fit. In addition to this, percent error between expected and real runtime values was far lower in the linear equation of best fit than the logarithmic equation of best fit. The linear equation always modeled a higher training runtime than the real runtime value at a certain epoch x-value, but had lower error in comparison to the logarithmic function at higher epoch values.

High vs. Low Positive Factor Models: Similarly to the statistical tests performed in the epochs vs. runtime data, T test values indicated that high and low positive factor models were extremely statistically significant (with p values less than 0.01). The high disparity in average percentage risks predicted show that low positive factor models consistently forecast much higher risk percentages for the same input data than high positive factor models.

Improvement of Prototype:

In retrospect, our group might have collected more data on epochs and their impact on training runtimes. Creating a graph that encompasses a greater window of data would allow us to calculate a logarithmic function of best fit that was more representative than the linear equation.
Another area we can focus on in the future is the development of a GUI application that shows the impact of each input feature in the model during the prediction sequence

Logical Reasoning (technicalities in the network, algorithms, and codebase):
Epochs vs. Runtime: When writing high performance algorithms, time complexity is a major factor a programmer must consider. In summary, time complexity is the amount of time in arbitrary units to a scale an algorithm takes to complete based on the operations that take place within the algorithm. In neural networks and machine learning, the neuron does not learn at the same rate as epochs pass. This means that the algorithm slowly does less and less math in each training cycle. Our neural network specifically adjusts weights corresponding to each input feature according to the error of the last epoch's prediction. The more epochs pass, the less the error in the network's predictions during training. Due to these factors, the learning rate of our neural network decreases over time as epochs pass. When the learning rate of the algorithm slows down, the amount of time spent per epoch decreases as well. Because of this, training runtime does not increase at a set linear rate as epochs pass. However, the linear equation of best fit was still more accurate in extrapolating our runtime data in comparison to the logarithmic equation of best fit solely because the logarithmic function plateaued off much more sharply than our runtimes did. In a perfect scenario, we would be able to measure runtime down to the nanosecond and observe runtime per epoch decreasing over millions of epochs.

High vs. Low Positive Factor Models: The way our neural network creates models is through weight correction over time and normalization around a baseline. When a network is trained on a low positive factor dataset, such as the lowest 20 deathcount data points, the model is normalized around a far lower relative risk. Whereas when a network is trained on a high positive factor dataset, such as the top 20 population density data points, the model's baseline risk is significantly higher. When a model's relative average risk perception is low, its risk predictions on new data points will be high, as the new data points far surpass the model's risk baseline. On the other hand, a model's risk predictions will be low for new data points if its existing risk baseline is significantly higher than that of the new data points.

Conclusions

Did ythe project turn out as expected?
Yes. The main goal of the project was to create a comprehensive system for predictive model creation and a training algorithm for COVID-19 forecasting. The result of the research was a easy-to-use, lightweight algorithm that can be used to forecast COVID-19 risk in any geographic area.

What do these results mean in the context of the literature review and other work being done in ythe research area?
We hope that the results of the research will be used in the future to further improve virus forecasting through the utilization of machine learning. We believe that intelligent forecasting models can be useful when used alongside existing machine learning diagnosis software. CRNN will be open-stheced so that other software developers, epidemiologists, data scientists, and people with knowledge in other disciplines will consider a new approach to disease forecasting.

How do the results address ythe research question? Do ythe results support ythe hypothesis?
The results of the project address the research question with two main pieces of supporting data. 1) The linear relationship between epochs and training runtime show that the training algorithm is refactored to a great degree and saves time complexity in its operations. In addition to this, extensive testing shows that the network can complete millions of epochs in less than a second without the use of multithreading or GPUs. 2) The statistical difference shown in the T-tests between opposing respective positive factor linked models show that a predictive model can easily be recreated and modified through the use of varying training data in order to conform to a specific use case.

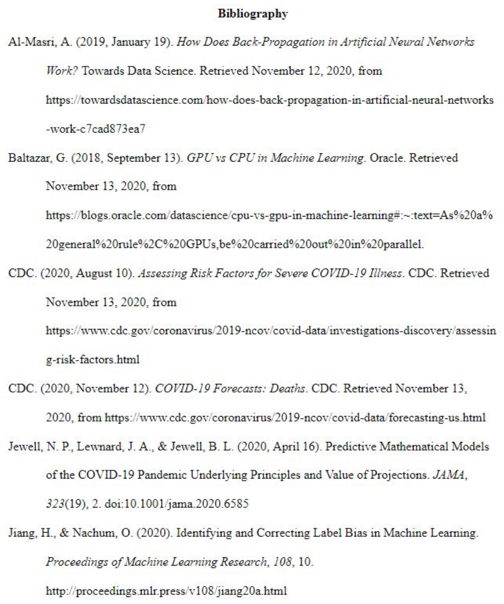
What application(s) do you see for ythe work?

Machine learning and artificial intelligence is becoming an increasingly utilized technology in all industries. This is because neural networks and trained models have the ability to display human-level intelligence in a given task, automating several processes across varying fields.

In a pandemic situation where there is a steady increase in publically available data, the utilization of a predictive model can be used as an alternate method of forecasting. Traditional mathematical forecasting models rely on large amounts of data and assume certain constants (such as spread and transmissibility rates). In contrast, a training algorithm and forecasting model is able to extrapolate existing data in areas such as school districts, parks, and neighborhoods. Differences in training data used can be taken advantage of in order to create specific models that fit a certain scenario. We strongly believe that the research can serve as a baseline for general pathogen forecasting using ML.

Aside from diseases and pandemics, the training algorithm is extremely efficient. the software prototype eliminates the need for expensive, high performance machine learning servers. In the future, we can optimize the algorithm for general ML tasks, such as classification problems. This would readily available hardware such as school laptops suitable for machine learning for trivial tasks.

References



Bibliography

Al-Masri, A. (2019, January 19). *How Does Back-Propagation in Artificial Neural Networks Work?* Towards Data Science. Retrieved November 12, 2020, from https://towardsdatascience.com/how-does-back-propagation-in-artificial-neural-networks-work-c7cad873ea7

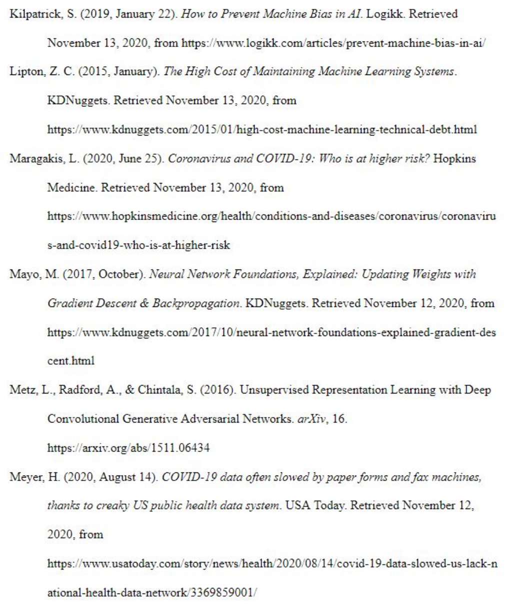
Baltazar, G. (2018, September 13). *GPU vs CPU in Machine Learning*. Oracle. Retrieved November 13, 2020, from https://blogs.oracle.com/datascience/cpu-vs-gpu-in-machine-learning#:~:text=As%20a%20general%20rule%2C%20GPUs,be%20carried%20out%20in%20parallel.

CDC. (2020, August 10). *Assessing Risk Factors for Severe COVID-19 Illness*. CDC. Retrieved November 13, 2020, from https://www.cdc.gov/coronavirus/2019-ncov/covid-data/investigations-discovery/assessing-risk-factors.html

CDC. (2020, November 12). *COVID-19 Forecasts: Deaths*. CDC. Retrieved November 13, 2020, from https://www.cdc.gov/coronavirus/2019-ncov/covid-data/forecasting-us.html

Jewell, N. P., Lewnard, J. A., & Jewell, B. L. (2020, April 16). Predictive Mathematical Models of the COVID-19 Pandemic Underlying Principles and Value of Projections. *JAMA*, *323*(19), 2. doi:10.1001/jama.2020.6585

Jiang, H., & Nachum, O. (2020). Identifying and Correcting Label Bias in Machine Learning. *Proceedings of Machine Learning Research*, *108*, 10. http://proceedings.mlr.press/v108/jiang20a.html



Kilpatrick, S. (2019, January 22). *How to Prevent Machine Bias in AI*. Logikk. Retrieved November 13, 2020, from https://www.logikk.com/articles/prevent-machine-bias-in-ai/

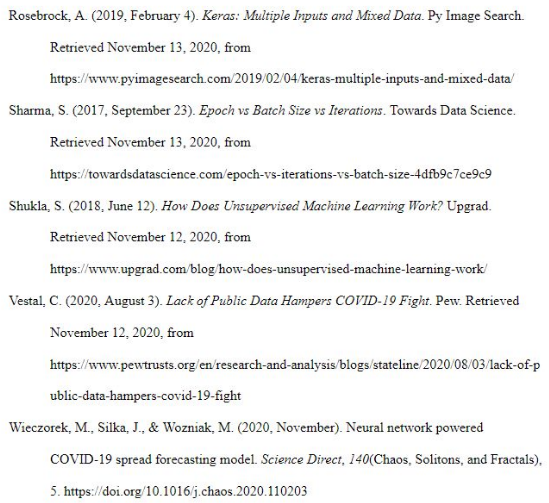
Lipton, Z. C. (2015, January). *The High Cost of Maintaining Machine Learning Systems*. KDNuggets. Retrieved November 13, 2020, from https://www.kdnuggets.com/2015/01/high-cost-machine-learning-technical-debt.html

Maragakis, L. (2020, June 25). *Coronavirus and COVID-19: Who is at higher risk?* Hopkins Medicine. Retrieved November 13, 2020, from https://www.hopkinsmedicine.org/health/conditions-and-diseases/coronavirus/coronavirus-and-covid19-who-is-at-higher-risk

Mayo, M. (2017, October). *Neural Network Foundations, Explained: Updating Weights with Gradient Descent & Backpropagation*. KDNuggets. Retrieved November 12, 2020, from https://www.kdnuggets.com/2017/10/neural-network-foundations-explained-gradient-descent.html

Metz, L., Radford, A., & Chintala, S. (2016). Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *arXiv*, 16. https://arxiv.org/abs/1511.06434

Meyer, H. (2020, August 14). *COVID-19 data often slowed by paper forms and fax machines, thanks to creaky US public health data system*. USA Today. Retrieved November 12, 2020, from https://www.usatoday.com/story/news/health/2020/08/14/covid-19-data-slowed-us-lack-national-health-data-network/3369859001/



Rosebrock, A. (2019, February 4). *Keras: Multiple Inputs and Mixed Data*. Py Image Search. Retrieved November 13, 2020, from https://www.pyimagesearch.com/2019/02/04/keras-multiple-inputs-and-mixed-data/

Sharma, S. (2017, September 23). *Epoch vs Batch Size vs Iterations*. Towards Data Science. Retrieved November 13, 2020, from https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9

Shukla, S. (2018, June 12). *How Does Unsupervised Machine Learning Work?* Upgrad. Retrieved November 12, 2020, from https://www.upgrad.com/blog/how-does-unsupervised-machine-learning-work/

Vestal, C. (2020, August 3). *Lack of Public Data Hampers COVID-19 Fight*. Pew. Retrieved November 12, 2020, from https://www.pewtrusts.org/en/research-and-analysis/blogs/stateline/2020/08/03/lack-of-public-data-hampers-covid-19-fight

Wieczorek, M., Silka, J., & Wozniak, M. (2020, November). Neural network powered COVID-19 spread forecasting model. *Science Direct*, *140*(Chaos, Solitons, and Fractals), 5. https://doi.org/10.1016/j.chaos.2020.110203