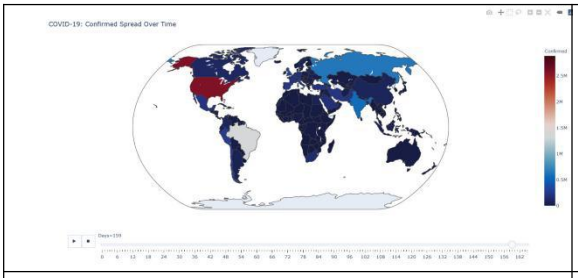
Merlin Project

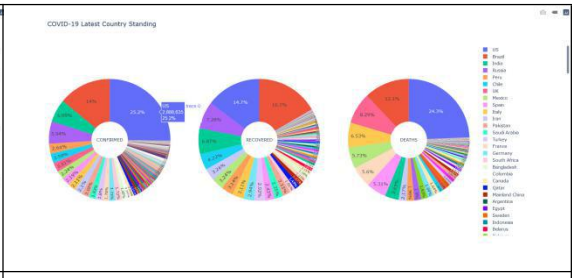
By: Amal Jogy

Abstract: Merlin project presents the COVID-19 analysis of countries across the world in different formats such as World Map, Bubble format, histogram ans sunburst etc. It also tries to interpolate and predict the future tendency (5 to 10% data points) ahead the COVID-19 data using neural network concept implemented on GPU and also for CPU. For training, the percentage of dataset may vary from 90 to 95%, depending on the dataset. Best interpolation accuracy was around 99.9% (Pearson R parameter) Typical runtime for GPU model was around 3 to 5 min and for CPU was 5-6 min. As of today, the project is 3/4 completed.

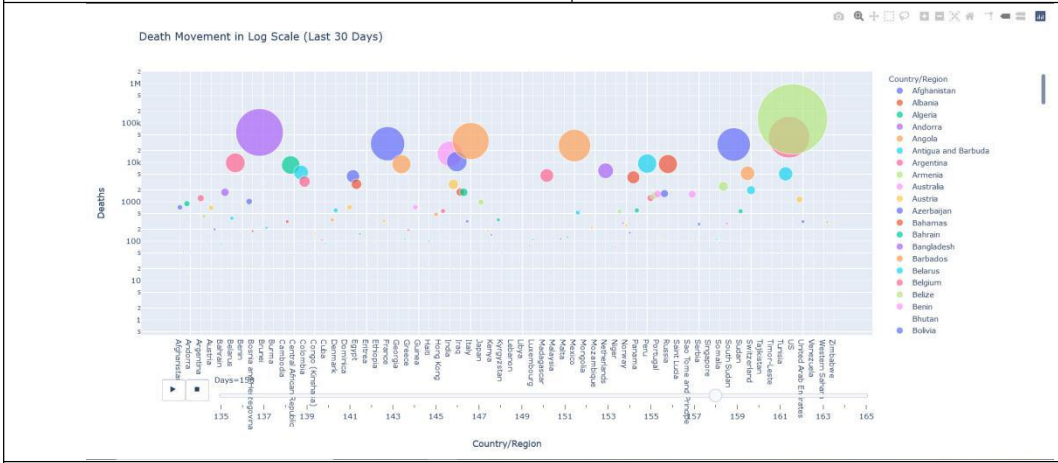
Graphs: Data Visualization examples: Kaggle Dataset on 04/09/2020



World Map: Global confirmed cases



Pie chart: Global confirmed

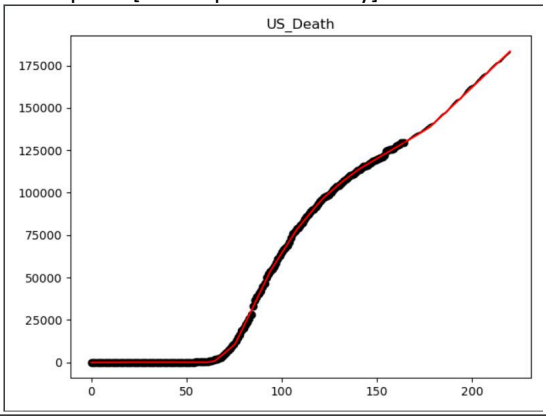


Bubble Graph: Global confirmed

Database dated:4th September 2020

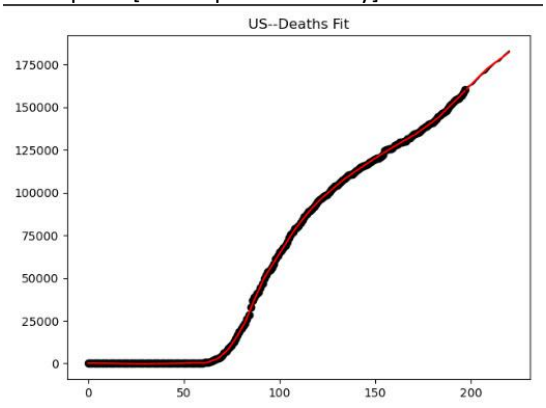
Graphs: Data Interpolation Examples for CPU and GPU Models: US Death a parameter

Incomplete [data + plot+ accuracy]



CPU Model 99% Accuracy Fitting (US Death)

Incomplete [data + plot+ accuracy]



GPU Model 98% Accuracy Fitting (US Death)

The thick solid dots in the above figures shows the data uses for training, and the fine red color data shows the predicted model results as well as the testing data used.

More Details:

CPU Model:

The CPU model is written in Python language using PyCharm 2020.1.3. The code has nearly about 800 lines. The CPU model has two parts: graphical representation of DATA in different formats such as World Map, Pie Chart, Bubble Graph etc for global and for a user selected nation, for Confirmed (C), Recovered (R) and Death (D). The second part of the CPU model uses a normalized dataset for interpolation using a Multi Layer Perceptron (MLP) type of Neural Network for training on a subset of the dataset (typically 80 to 95%) and test the accuracy of the model for the rest of the dataset. The CPU model has no facility currently for predicting the future trend, though it can be implemented, if necessary. However, this facility has been implemented in the next part, GPU model. The CPU Model uses MLP Regressor from Scikit-Learn and uses a custom made Training Module which finds the best model with Number of Hidden layers , Activation Function, Alpha Value as the iteration values for the model trainer. The testing of the model is done for 5 to 15% of the later data points to check its model accuracy. The CPU model takes around 5-10 min to complete the training and testing. This model stores and track the best model in a CSV file.

Scenario-1: (First Run Detected): If the CSV file is empty or if there is no history of the model for user selected parameter, then the model is trained with 70% data points, and testing is done on the remaining 30% data points. The best model parameters such as neural size, activation function, solver, model name, error matrix and accuracy etc are stored in a CSV file for future use.

Scenario-2: (Pre-Trained Model Detected): For a user selected parameter, if the CSV file has a model with more than 90% accuracy, then the training part is skipped and the best model is directly to the testing phase and prediction of the data.

*Scenario-3: (Training):*For a user selected parameter, if the CSV file has a model with less than 90% accuracy, then the model is trained again and the best model parameters are saved in the CSV file. The is model is then applied to the testing (30%) and prediction of the remaining data.

The best accuracy obtained during the testing phase was around 99.8%.

GPU Model:

The GPU model imports the plotting facility from CPU model, and the rest of the code is entirely different as it optimized for implementation on multiple core GPU. The model uses a custom made architecture which loosely resembles Recurrent Neural Network (RNN). The model is made from PyTorch Library and uses OPTUNA library for hyper-parameter optimization. For self learning and faster prediction, the model retains two types of data in the hard-disk:

A) Pickle Format Model Data: The trained models with all its parameters is stored in pickle format. Sometimes 10 to 25 entries are stored for each model, which vary in neural size, learning rate, activation weight, neural architecture. At the end of the training session, the best model with respect to accuracy is identified and this information is stored in CSV format mentioned below.

B) CSV Format for BEST model: The most accurate model from training with only its important parameters such as Neural Size, Learning Rate and Activation Function along with its R^2 parameter are stored in CSV file.

Model Improvement/ Selection Logic: When a user inputs the country and targeted parameter such as Confirmed, Recovered or Death of CVOID-19 cases, it immediately checks the CSV file for any past history.

Scenario-1: (First Run Detected): In the absence of any history or empty CSV file, it detects First Run scenario, and trains the Nerural Network for 10 epochs. The 10 iterations are stored separately in Pickle Format and the best model from there is stored in the CSV and is used for prediction.

Scenario-2: (High Accuracy Detected): If a model with more than 89% accuracy is detected, then the model is directly used for prediction, while skipping any training. This saves time.

Scenario-3: (Potentially High Accuracy Detected): If a model wit more than 71% but less than 89% accuracy is detected, the model is trained for 15 epochs instead of (10 default value). The previous 10 epoch results for the model in Pickle format are overwritten with the new 15 epoch results. At the end of the session, the best model is identified again and updated in the CSV file (append operation).
Scenario-4: (Low Accuracy Detected): If a model with less than 71% accuracy is detected, then the model is trained with 15 epochs. As earlier, the Pickle Format data and CSV data are updated.

The models are benchmarked for its accuracy based on the *mean squared error* as the criterion and after detecting the best model, its R2 parameter (which is the square of the pearson R parameter) is estimated and this information is updated as accuracy in the CSV file.

The best accuracy obtained was 99.9% for GPU model.

Difference between Sci-kit R2 parameter and Pearson R parameter.

In statistics, there are two types of unit-less accuracy detection techniques. One is the R2 parameter implemented in the numerical library of Python known as Sci-kit Learn Library. The value of this R2 parameter can mostly vary from 0 to 1 with 1 for the best match between observed data and predicted data and 0 for no correlation. However, at times, the parameter can even be negative when arbitrary model not matching with the data is forced to fit on the observed data. The mathematical model of sci-kit learn R2 parameter **R(y,ŷ)** is given as

$$R^2(y,\hat{y}) = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

where $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ and $\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{n}\epsilon_i^2$.

Where y_i are the observed values of y parameter, $\bar{\mathbf{y}}$ is the average value of y parameter, and $\hat{\mathbf{y}}_i$ is the predicted value of the y parameter. Thus, if the error is much larger as compared to the deviation of each value from the average value, the R2 parameter can be negative!

However, the Pearson R parameter is also a standard feature to check the quality of fit against the observed data. The Xi are the observed values and Yi is the predicted values and $\bar{x}$ and $\bar{y}$ are their averages, then the Pearson R parameter is given by,

$$r = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2}\sqrt{\sum_i (y_i - \bar{y})^2}}$$

The value of the R parameter vary from -1 to 1, with 0 for uncorrelated data of (X,Y) and either 1 or -1 for perfect correlation. Also, -1 for perfect negative correlation) X increases, Y decreases) and 1 for positive correlation (X increases and Y also increases).

Sometimes, the square of the Pearson R parameter is also termed as he R2 parameter, but then it strictly varies from 0 to 1 only, with no possibility of negatives values unlike the previous Sci-kit R2 parameter.

During the benchmark of various models, we use the Sci-kit library mean squared error to find the best model. Later, its Sci-kit R2 parameter is stored in the CSV file.

For testing the accuracy of the model, the software also provides CSV data in table format (X,Y) where X is the observed data and Y is the predicted data. This can be manually applied to the Pearson R value calculator available at: https://www.socscistatistics.com/tests/pearson/default.aspx

Requirements for running the software:

1. Standard Picharm IDE with additional libraries such as PyTorch (torch), Sci-Kit Learn, Optuna, Plotly, datetime, Pickle, Joblib etc
2. COVID-19 database as on 4 Sep 2020 (downloaded from Kaggle: https://www.kaggle.com/sudalairajkumar/novel-corona-virus-2019-dataset)

3. Tracker files databases:
    a) CSV for CPU Model
    b) Pickle database and CSV database for GPU model.
4. CPU Model software: MERLIN_CPU.py
5. GPU model software: GPU_MODEL_v7.py
6. Nvidia Graphics Card MX-150 or greater
7. CUDA (Nvidia software to run code on GPU): cuda_10.1.105_418.96_win10.exe

Future Plans:

Some of the future plans include

1. Pearson R parameter estimation either by using a standard Library or by developing the code myself
2. Implement more sophisticated algorithms to improve accuracy of the models and faster execution
3. Bug fixing, if any

About me:

I am, Amal Jogy, a 3rd semester CSE student at SRM Institute, Chennai. This was part of my home project during my COVID-19 break.

Address: E 77, VIndhya Tower, RRCAT Colony, Indore - 452013. Madhya Pradesh
Mobile/WhatsApp:+91 81032 72992
Gmial: amaljogy08@gmail.com
Linkedin: https://www.linkedin.com/in/amal-jogy-3b650b18b/