TASK 1

Signal Processing and Machine Learning

Machine Learning

PROBLEM STATEMENT – MULTI - LAYER PERCEPTRON FROM SCRATCH:

Given a dataset on the number of cases and deaths due to COVID-19, it's useful to know how the situation will progress in an area based on the current stats, so we can lock down, or loosen restrictions as needed. You are given the following data: continent and country, date, along with that day's cases and deaths. Build a multi-layer perceptron to do the following things:

- During the training stage, use the number of cases/deaths of the previous day as an input to the current day.
- Predict the number of deaths and cases for a given day in the future (modelled as a recurrent relation).

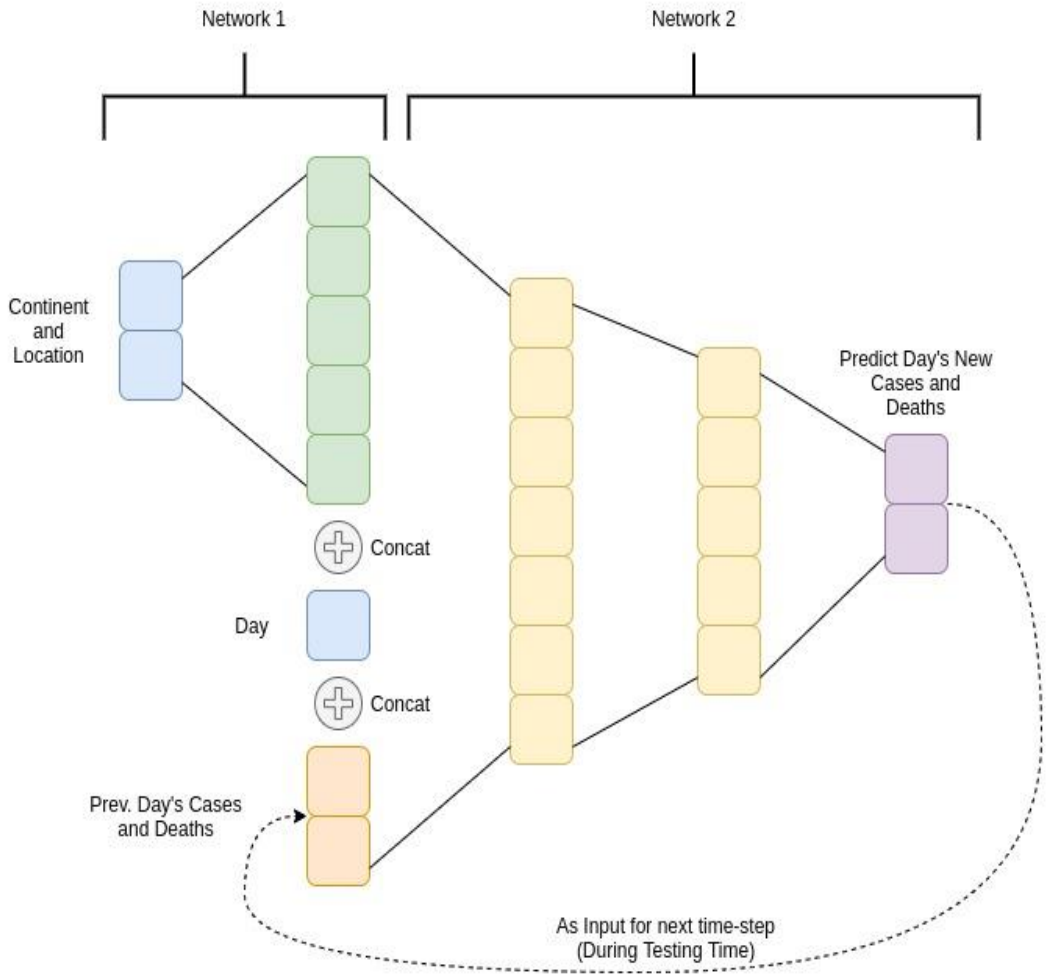
DATASET:

Link:

https://drive.google.com/drive/folders/19ENDyyza1N1VcSKdBpJZCSHoi7v03OPF?usp=sharing

GUIDELINES:

- To do this, use only numpy. DO NOT use libraries like Tensorflow, PyTorch etc. The layers in your model must be built from scratch. Visualisation libraries like pandas are allowed.
- Follow the diagram shown to create the network's architecture. All layers shown are Dense layers.
- Usage of other data given in the dataset is left to your choice. The graph is just representative of the barebone architecture.



Steps to Follow

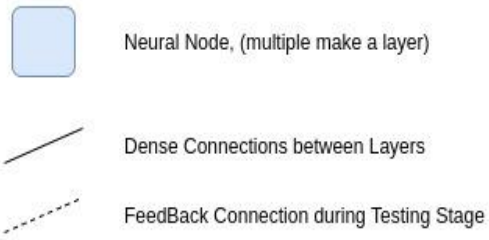
Let us say we are training for 2X epochs.

- Train Nets 1 and 2 **together** for **X** epochs with $\mathbf{lr_1 > lr_2}$
- **Freeze the weights of Net 1** and train Net 2 alone for the rest of the X epochs with $\mathbf{lr_2}$

lr - Learning Rate

Note:
The architecture must be strictly followed. The activation functions are not portrayed and are left to your choice.

Key To Graph



The prediction for a given day is built on the prediction of all the previous days recursively while making predictions.

EVALUATION METRICS:

- Accuracy of prediction
- Efficient implementation of layers
- Code quality – use proper classes and functions as necessary

SUBMISSION:

Submit a folder with the following files:

- train.py – containing the training and evaluation functions
- layers.py – containing classes of the layers used
- activations.py – with classes of the activations used
- weights.npy – the final weights of the model after training
- load_weights.py – loads the weights into the model
- main.ipynb – a Jupyter notebook where you import the necessary modules from the files and train and validate the data.
- After making the folder, compress the folder to .zip format and upload to your Google Drive, then get the sharing link of the folder from Google Drive and submit that link to induction portal.

RESOURCES:

- https://numpy.org/doc/1.19/
- https://www.coursera.org/learn/neural-networks-deep-learning?specialization=deep-learning
- https://datascience.stackexchange.com/questions/25606/gradient-flow-through-concatenation-operation