Covid-19 prediction using spatio temporal recurrent neural network

Project submitted
In partial fulfilments of the requirements for the degree of

MASTER OF COMPUTER APPLICATION

By

Abhirup Bandyopadhyay
Roll No: **001910503046**
Registration No: **149906 of 2019-20**

Examination Roll No MCA226045

Under the supervision of

DR. ANASUA SARKAR

Department of Computer Science & Engineering
Faculty of Engineering and Technology
Jadavpur University
Kolkata – 7000 032
India

Jadavpur University

Faculty of Engineering and Technology

Department of Computer Science & Engineering

CERTIFICATE

This is to clarify that the project entitled“**Covid-19 prediction using spatio temporal recurrent neural network**”has been completed by Abhirup Bandyopadhyay.This work is carried out under the supervision of Dr. Anasua Sarkar in partial fulfilment for the award of the degree of Master of Computer Application of the department of Computer Science and Engineering, Jadavpur University, during the session 2019-2022. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the said degree.

Dr. Anasua Sarkar
Project supervisor
Computer Science & Engineering
Jadavpur University

Countersigned:

Prof. Anupam Sinha
Head of the department
Computer Science & Engineering
Jadavpur University

Prof. Chandan Mazumdar
Dean
Faculty of Engineering & Technology
Jadavpur University

Jadavpur University

Faculty of Engineering and Technology

Department of Computer Science & Engineering

CERTIFICATE

This is to certify that the project entitled "**Covid-19 prediction using spatio temporal recurrent neural network**"has been submitted by Abhirup Bandyopadhyay in partial fulfilment of the requirements for the award of the degree of **Master of Computer Application** in the department of **Computer Science & Engineering**, Jadavpur University, during the period 2019-2022 has been carried out under my supervision and that this work has not been submitted elsewhere for obtaining a degree.

EXAMINER:

------------------------------------	------------------------------------
INTERNAL EXAMINER	EXTERNAL EXAMINER

DECLARATION OF ORIGINALITY
AND
COMPLIANCE OF ACADEMIC ETHICS

I hereby declare that this project contains original work by the undersigned candidate, as part of his Master of Computer Application (MCA) studies.

All information in this document have been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by these rules and conduct, I have fully cited and referenced all material results that are not original to this work.

Name : Abhirup bandyopadhyay
Roll No : 001910503046
Examination Roll No.MCA226045
Project Title : Covid-19 prediction using spatio temporal recurrent neural network

----

Signature with date

ACKNOWLEDGEMENTS

On the submission of “Covid-19 prediction using spatio temporal recurrent neural network.”, I would like to convey my sincere gratitude to Dr. Anasua Sarkar, Professor, Department of Computer Science & Engineering, Jadavpur University for her valuable suggestions throughout the duration of the thesis work. I am really grateful to her for constant support which immensely helped me to fully involve myself in this work and develop new approaches in the field of Machine learning.

Lastly, I would like to thank all my teachers, classmates, guardians and well-wishers for encouraging and co-operating me throughout the development of this thesis.

Regards,

Abhirup Bandyopadhyay
Roll Number: 001910503046
Examination Roll No.MCA226045
Registration Number: 149906 of 2019-20

Signature with date

Table of Contents :-

Abstract: Several studies of Graph Neural Networks has lead to a better prediction of the Covid-19 evolution than many other models .Out of Many such algorithms/models LSTM(Long Short Term Memory) a time series prediction tool ,is one of them.In our work ,we make an experiment by collaborating Graph neural network and Graph Convolutional Long Short Term memory.We achieve this combination of two modules by applying Spectral Graph Convolutional operator in place of linear transformation of Long-Short_Trerm Memory module gates.We hope to exploit spatial pattern in data by this module integration.Moreover we introduce the notion of skip connection to achieve a significant improvement in jointly capturing the spatio-temporal pattern in our raw input data.We select a timing window of nearly five hundread days days from WHO-COVID-19 DATA and choose thirty Countries to train test and validate our new-cases prediction on COVID-19 .Further we test our model based on multiple error metrics like RMSE,MASE,MAE ,R^2,MAPE and display their performance in tabular form.This research area has a potential in analyzing pandemic resource control ,spread forecasting and policy making application.

Introduction :Many researches have been conducted using GNN to model the pandemic COVID-19.Many papers have applied GNN using mobility data ,while some other have exploited some classic epidemiological models by concatenating outputs from the GNN model and feeding that to an epidemiological one and then into a distributional regression layer finally[29].Some papers have used the combined version of GNN and LSTM for predicting some SIRD(suspected infected recovered dead) model parameters[30].But our work here will not be explicitly dependent on any particular epidemiological model .our model will be able to parameterize its input features on its own. Some studies have leveraged the spatio-temporal dependencies in data and worked on modeling them jointly[31].This is done to take the advantage of The existing correlations that sustain on both timeshots and geographical regions but not separately. This they achieved in their work by means of a fourier transform.In our study the LSTM is modified by replacing the (matrix multiplication) liner transformation by an spectral Graph Convolutional operator(Chebyshev) .We have also applied the idea of a skip connection .This concatenates the spatial GNN output with above mentioned LSTM model.This solves and improves many common issues and challenges in jointly modeling the spatio-temporal dimensions .

Litarature Survey:-From the report of World Health OrganizationThe covid-19 pandemic has crossed nearly over 63 Lakh deathes and $2.96 trillion economic(GDP) Loss [24],[25] .It has devasted many countries [22],[23] .Two very important mesurements to tackle such damage in current and advance time is effective policy making (Intra National and Inter National) and prompt action(early response).Though The number of vaccination is steadily increasing, these two abilities will be very urgent to adopt due to the future repeation of this Pandemic is a possibility.More Often The Epidemological transmissions are exponential in nature[26].For this reason even if we mange to achieve a small improvement on early policy making and advanced intervention ,we receive a large positive impact on our end,proving premtiveness the most effective measure yet on tackling Pandemic.So the most crucial factor while fighting COVID-19 like pandemics is to stand out to be finding the information on future spread(regional and numerical) of these virus .Given the uregency of the pandemic ,many reserchs have been conducted to forecast the pandemic.The Machine Learning Community has achieved solutions from many directions and capacities,[40],[41],Where we can find an research conducted aiming early detection of covid from Computed Tomography Scan images[42],[43], where as another filed of research can be found aiming to find number of new hospitalizatins and new COVID-19 cases in future through various Time-series models[44].The LSTM network was realized to be very effective in early pandemic because of the inherent temporal patterns of the Covid-19 transmission [45],[46],[47].These reserched resulted in low error compared to the general Epidemological models like SIRD models[48].The two factors one that the epidemiological data has an inherent graph structure and the other is the spatio-temporal patterns and temporal corelations in data have been of special interest while developing Some combined temporal models such as Graph Neural Networks (with some recent algorithmic developments[28]) and LSTM(Long Short Term Memory) networks[27].In our paper we are building a time series input based model for predicting new COVID-19 cases. We have proposed a general method upon which This model is made on integration of GNN and Graph Convolutional LSTM and we are hoping to achieve an improved accuracy with this model.Our experiment with this model consists of thirty European provinces and a time series data of daily new cases ranging from 2020 to 2021 .In results and baseline section we have showed the model wise comparision of our model with severel other state of the art GNN and LSTM Machine Learning models.we show there that our model is able to forecast the new COVID-19 cases better accuracy than other models .We have achieved an 10% error while forecasting new COVID-19 cases after 7 days from any given day.Finally we have added that our study suggests of adding the notion of skip connection to control the bias and variance of the model.

Background :-

A.Graph Sage :

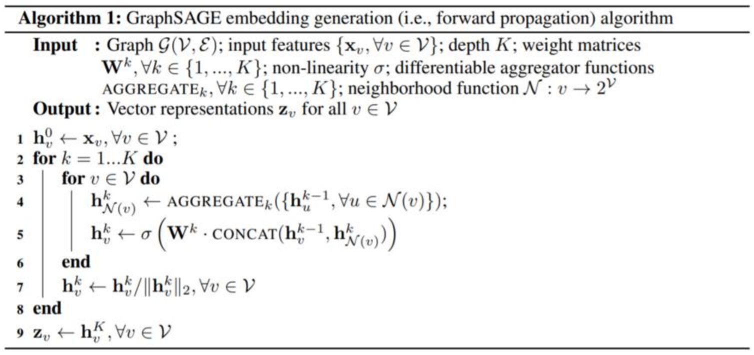
IN Graph structure every Node has some information in terms of feature vector,this is called feature information.For any particular node,its local neighbourhood all has some feture information.Once we create the graph by defining nodesAnd assigning edges bewteen two node enties ,we incorporate many Problem Specific relationships between entities(ie nodes) from the graph we formed.In our problem nodes are geographical regions and edges are nearest 3 regions with distances as edge features.Now the graph is formed , we need to integrate some meaning ful information(feature) about this structure(like local neighbourhood structure of a particular nodeor its global position in structure)into the machine learning model we propose to build in our paper.While extracting structural information we also want to train the model with a loss function And learn the features with the help of a loss function.To achieve this purpose we adopt a representation learning approach, GraphSage to encode the structural information that the model learn , to a embedding space (Euclidean space)We want to achieve a mapping function which embade our entire graph nodes from a non-euclidean (high-dimensional) space to a embedding spacewhich is low dimensional and is an Euclidean space.The mapping function we hope to learn should retain geometric relationships of our original graph structure inside our Euclidean space or embedding space. So that nearby nodes embedding stay close (high dot product) and unconnected (many hopes away) nodes are shoved apart. In the below diagram, we can see the mapping process.Encoder(enc) maps the nodes u and v from the graph from a high dimensional space to a low dimensional vector space Zu and Zv which is the embedding space.

So the main porpuse of GraphSAGE is to make these nodes learn to aggregate this neighbouring information.This learning method can be decomposed into two steps 1.Innitializing The model Parametres 2.Embedding generation :- Genrating Embedding for the Graph nodes through Forward Propagation.3.Learning Model-Parametres using gradient decent(stochastic) and back propagation.

A1.Embedding generation (i.e., forward propagation) algorithm :-

The Model Parameters in particular are 1.The Aggregator functions parametres .These functions are used to collect and aggregate feature information from the node's neighbourhood and denoted by (AGGREGATE$_K$, ∀k ∈ {1.....K}, total there are k functions).2Set of Matrices .These matrices are used to propagate through feature information through separate search depths of GraphSage model.These are called weight matrices and denoted by W^K , ∀k ∈ {1.....K}.

MAIN AlGORITHM[32]:-



```
Algorithm 1: GraphSAGE embedding generation (i.e., forward propagation) algorithm
  Input  : Graph $\mathcal{G}(\mathcal{V},\mathcal{E})$; input features $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$; depth $K$; weight matrices
           $\mathbf{W}^k, \forall k \in \{1,...,K\}$; non-linearity $\sigma$; differentiable aggregator functions
           $\text{AGGREGATE}_k, \forall k \in \{1,...,K\}$; neighborhood function $\mathcal{N}: v \to 2^{\mathcal{V}}$
  Output : Vector representations $\mathbf{z}_v$ for all $v \in \mathcal{V}$
1 $\mathbf{h}^0_v \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$ ;
2 for $k = 1...K$ do
3     for $v \in \mathcal{V}$ do
4         $\mathbf{h}^k_{\mathcal{N}(v)} \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}^{k-1}_u, \forall u \in \mathcal{N}(v)\})$;
5         $\mathbf{h}^k_v \leftarrow \sigma\left(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}^{k-1}_v, \mathbf{h}^k_{\mathcal{N}(v)})\right)$
6     end
7     $\mathbf{h}^k_v \leftarrow \mathbf{h}^k_v / \|\mathbf{h}^k_v\|_2, \forall v \in \mathcal{V}$
8 end
9 $\mathbf{z}_v \leftarrow \mathbf{h}^K_v, \forall v \in \mathcal{V}$
```

(1)

Figure 1. Graph SAGE Algorithm[53]

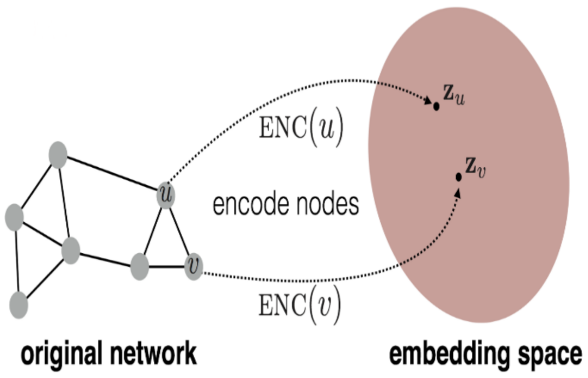


Figure 2. Node feature mapping [54]

A2.Node Embedding Generation : - performs node embedding by two major steps.First is from the input graph -> Neighbourhood Sampling and Second is at each srach deapth -> Learning aggregation function.This learning of aggregator function is the main element behind the inductiveness property of GraphSAGE.The GraphSAGE method , uses forward propagation at each layer(k) to generate node embeddings.Before moving to the working formulae ,Lets consider this variable and constant notations ,

X_v = node v feature vector

h^0_v = input node feature .

h^k_v = node embedding at search deapth k

z_v = final embedding of a node v after all search depth exhausted.

W^k = matrix of weights at kth search depth

N : v $\rightarrow 2^v$ = function of neighbourhood

The below diagram illustrates the process of how a node named node 0 aggregates information at search depth K=1 , from its sampled-neighbourhood.

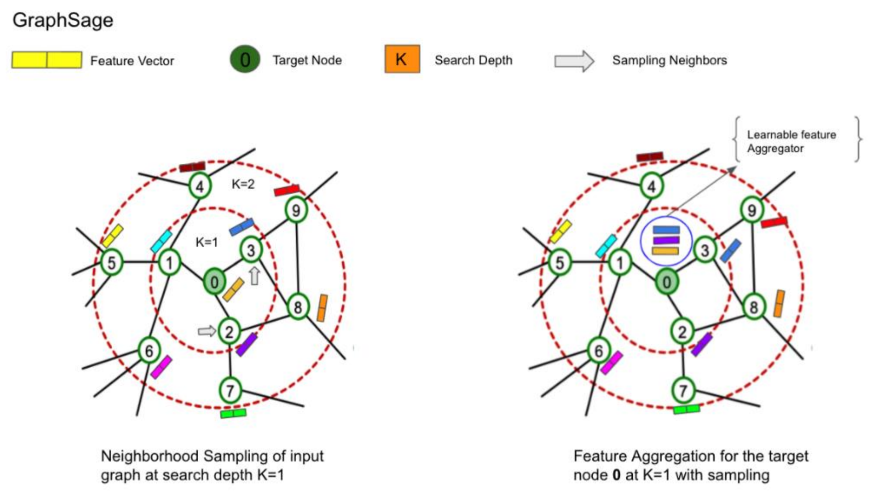

GraphSage
Feature Vector
Target Node
Search Depth
Sampling Neighbors
K=2
K=1
Learnable feature Aggregator
Neighborhood Sampling of input graph at search depth K=1
Feature Aggregation for the target node 0 at K=1 with sampling

Figure 3. Neighbourhood sampeling and feature aggregation.[54]

If we observe the above picture , we can see that when aggregating features (ie in the right side graph) at K =1 , our target node which is node 0 conatins the information(feature – information) Of nodes at one hop surroundings.As we described in above , that the target node 0 at K=1 aggregates feature-information from its 1-hop neighbourhood.similarly the target node 0 aggregates feature information from upto 2-hop neighbourhood at K=2.Thus as we iterate through K , the target node 0 inductively obtains more features from further nodes of the graph.This feature-information gathering process for each node in our input graph. $(\forall v \in V)$.The computation graph at layer K=0 of a target node , node 0 is described in below image.At this initial stage all graph nodes holds their input feature vectors (raw)as feature information.We propse to find the final embedding of node 0 (ie the final representation in low dimensional space = Z0) at layer 2(K=2).We learn this representation through an itarativeprocess og gathering local neighbourhood information..This is phrased as messege passing approach.We can mathematically represent this step as ,

$$h_v^0 \leftarrow x_v, \forall v \in V$$

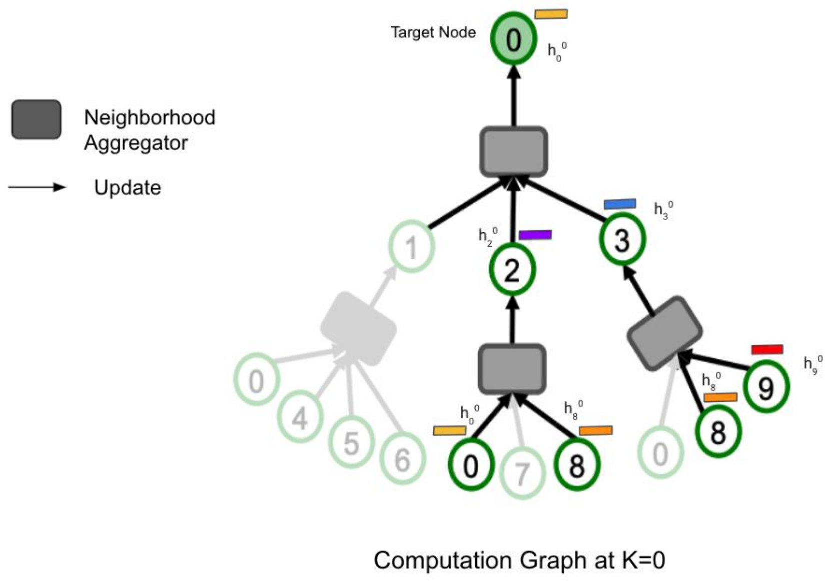


Figure 4 : Computaion graph at initial stage[54]

Herre We denote number of layer by superscript(kth) and Node id by subscript.Neighbourhood Aggregation at K = 1:We start our search depth iteration at K= 1 and run through 1……K, and nodes incremently gather Information about more nodes from its neighbourhood at deeper search depth of the graph.We aggregate this neighbourhood features of sourrounding nodes at previous layer for our target node (node0) into a vector.In picture h^1_0 is the vector representation of the aggregated features.The other nodes at the same time also aggregate feature information from their respective 1-hop neighbourhood.Thus each node at this point of time in a computation graph has the information of their respective immideate(1-hop) surroundings. We can represent this step mathematically as :-

$$\mathbf{h}^k_{\mathcal{N}(v)} \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}^{k-1}_u, \forall u \in \mathcal{N}(v)\});$$

Updation: Now once we get the neighbourhood aggregation at h^1_0 , we concatenateThe previous layer feature ie h^0_0 of target node 0 with the aggregated represntataion .Then we perform a linear transformation on this vector by a multiplication of weight matrix W_K.This weight matrix is learned by the model individually at each search depth k of 1to….K.Basically it learns to aggregateneighbourhood information for particular nodes at each search depth .After this process ,Now we pass This transformed output thru a non-linearity (such as sigmoid activation) for future learning and tasks.We Can represent above process mathematically as,

$$\mathbf{h}^k_v \leftarrow \sigma\left(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}^{k-1}_v, \mathbf{h}^k_{\mathcal{N}(v)})\right)$$

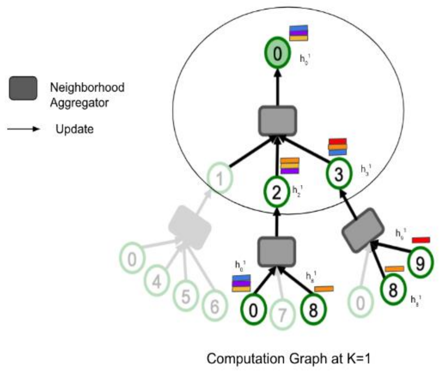


Figure 5. computation graph at 1st search depth.[54]

Normalization of node Embeddings:-

After we determine h^k_v we normalize this node representation. This is computed as ,

$$\mathbf{h}_v^k \leftarrow \mathbf{h}_v^k / \|\mathbf{h}_v^k\|_2, \forall v \in \mathcal{V}$$

This encourages a generelized distribution in graph node embedation.

A3. Node Embedding at second search depth :

At K=2 we explore further neighbourhood (2 hops) of the graph.When performing a node'sNeighbourhood aggregation, the target node 0, this time will aggregate information about further reaches of input graph ,ie it will cover upto 2nd hop distance. After this we process like previous itration First By updation and then normalization .Like this we perform our 2nd hop processing .At the end of 2nd hop iteration we stop and at this time each node has their respective low dimension representation in Euclidean space , and they are casted in the computation graph by their final node embeddings Z_V.The GraphSAGE workflow of our paper as referenced above is summerized in below image,

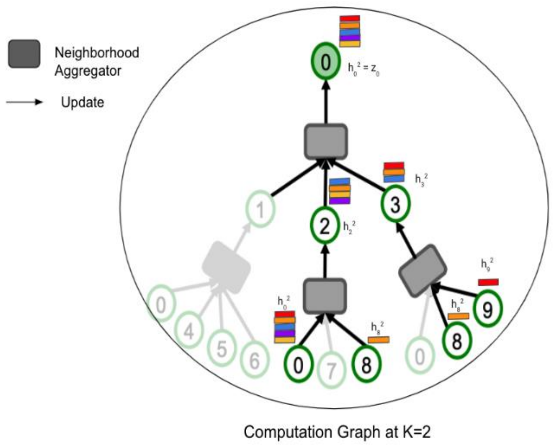


Figure 6. Computaion graph at 2nd search depth [54]

A4.Neighborhood definition :- Like said previously we we want some set of neighbourhoods into specific sized and consistently sampled , in-place of using all the nodes in a vertx neighbourhood like main algorithm.This procedure makes sure that the computational cost of each batch (in the mini-batch setting) remains nearly constant.Thus the notation N(v) denotes a fixedsize , consistent draw , and so at each search depth k the N(v) gathers different but consistent samples form the set {u ∈ V : (u,v) ∈ E}.We have taken this sampleing starategy because if we sample the full neighbourhood set then the memory and operation cost is of very high deviation And is of $O(|V|)$ in worst case in a single batch. By the minibatch adoption and neighbourhood sampling we have managed to reduce the memory and operation cost of graph sage to a fixed entity as $O(\prod S_i)$ here i runs from 1 to K . K and S are user given constants.

A5.Aggregator Arcitecture :-Mean aggregator is one of the permutation invariant aggregator we can use for aggregating nodefeature vectors.The mean operator , when performing neighbourhood aggregation for any particular node v (in the inner loop of the grph sage algorithm)it performs an per-element mean of all vectors in the nodes neighbourhood ie h^{k-1}_u , $\forall u \in N(v)$.

B.LONG SHORT-TERM MEMORY :- The Long short Term memory Cell is Derive from The General Recurrent Neural Network Class which very efficiently address the Vanishing gradient problem. Also LSTM s (introduced by [2]) are very efficietnt in capturing long term dependencies . Observing the architecture of LSTM in [2] ,we deducted that LSTM is a special class of Recurrent Neural Network.If we observe any typical RNN Architecture , we can follow that they contain a series of neural network component (e.g a tanh layer)in repated fashion. Science LSTMs are a derived class of RNN, they have similar kind of chain-like structures but The neural network component in each module or structure is not a single but four neural network layer.

B1.Structure overview :- The crucial idea on which the LSTM is centered is cell state , in the below figure.This cell state works like a conveyor in the entire chain structure of LSTM, where according linear Operations are performed in required places of this cell state flow line.Learned and Controlled by gates ,the LSTM add or removes Data to the cell state line.This gates of LSTM regulates the information flow by optionally letting them through.This gates are built with two components one is a neural network layer with sigmoid nonliearity activation in [0,1] controlling the portion of each component of information to pass through and the other is an elementwise multiplication operation.

B2.Gate Calculations:- As any vector input comes to LSTM , first it calculates the forget gate ie . decides How much of the input is redundant and needs to be thrown out from the cell state .This forget gate layer is composed of a sigmoid layer,which first reads the h_{t-1} and the x_t and calculates a number in [0,1] ,for each number of the C_{t-1} (cell state) by the equation

$f_t = \text{sigmoid}(W_f.[h_{t-1},x_t] + b_f)$.

The output 1 means keep all information and 0 means keep nothing.The next work of LSTM is to decide which of the new information received is to save in the cell state by input gate calculation.This calculation consists of two parts . In the first part , the sigmoid input gate layer the values to be updated, in the second part a new candidate vector is produced (by A tanh layer) which is to be added to the current state and this vector is denoted $C^{\sim}_t$.

$i_t = \text{sigmoid}(W_i.[h_{t-1},x_t] + b_i)$

$C^{\sim}_t = \tanh(W_c.[h_{t-1},x_t] + b_C)$

Now The LSTM updates the previous cell state C_{t-1} and creates new cell state C_t.For this ,the old state is multiplied by forget gate this makes the LSTM forget the information that was calculated to be,and then this result is added with the multiplication of input gate and candidate vector $\tilde{C}_t$.Thus we scale the new candidate values by the intensity of updation of each state value.

$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$

Finally the LSTM calculates the output gate .This will be basically a filtered version of cell state.First it decides which parts of the cell state to be outputted ,by a sigmoid layer.Then it squishes The cell state in [-1,1] through a tanh layer and finally multiplies the result with the sigmoid gate.Thus only intended part of output gate is outputted.

$o_t = \text{sigmoid}(W_o.[h_{t-1},x_t] + b_o)$

$h_t = o_t * \tanh(C_t)$

We have gone through numerous kind of sequence modelling tasks involving long term dependencies and we shaw the LSTM architecture have provided a very robust and stable model framework [1],[3],[4]. Now In our context of problem we want our input to the LSTM to be a d_x dimensional vector x_t ,whereas the output should be a vector in R^{d_h} as h_t and will exist in the interval of [-1,1] and finally we want states of this special RNN to be a vector in R^{d_h} that is a d_h dimensional vector which is denoted by c_t . Now let us discuss some notations that we will use in our model formulation , like We will denote the Hadamard product by @ ,we use here a sigmoid $\sigma\ (var) = 1/(1+ e^{-var})$ and finally LSTM gates as output gates forget gates and input gates are noted as o,f and i .Each of these gates we need to have as a d_h dimensional vector in the domain [0,1] . For this kind of model structure we use a multivariate(FC-LSTM) LSTM,for the formulation of which we follow [1] ,-

$i = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + w_{ci} @ c_{t-1} + b_i)$

$f = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + w_{cf} @ c_{t-1} + b_f)$

$c_t = f_t @ c_{t-1} + i_t @ \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c)$ (3)

$o = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + w_{co} @ c_t + b_o)$

$h_t = o @ \tanh(c_t)$

The parameters of the afore mentioned model Are The weights and biases where the weights are referenced as W_x , a matrix vector of the dimension $d_h * d_x$,second the W_h as a matrix vector of the dimension d_h*d_h And finally w_c as a vector of dimension d_h in R.Whereas b_o,b_i,b_c,b_f are the output , input , cell and forget gate biases of the dimension d_h in R respectively.Moreover we note the use of $w_c\ @c_t$, the peephole connectivity by [5] which improves the operation complexity and model robustness.We observe that all the vector component of the vectors x and h are linear combination of the two afore mentioned weight matrices namely W_x and W_h which are dense in nature.Thus we achieve a Fully connected LSTM model(FC-LSTM).

C . Graph signal filtering operator :- We want to generalize Convolutional Neural Network To arbitrary graph structures.This operation requires some fundamental proceedings to be delt with 1.The convolutional filters should be designed for them to localize On graph structures.2.Similer vertices Grouping should be captured by a coarsening technique.3.To manage the tradeoff between resoluation of high filters and resolutions of spatial structure of the Graph in consideration a pooling method should be determined.

C1 Spectral Filters:

Now let us take a look on what are the facts we should lookout when we define filter for Graph convolution .There are mainly two methods one is a spatial convolution and the other is spectral convolution.The spatial convolution , once the finite kernel size isKnown , it can perform a filter localization.Graph convolution Although is very relevant in spatial doain,The task of local neighbourhood matching is significantly complex and challenging[13].In fact from spatial perspective there is no definite and dedicated mathematical method of graph translation .Looking for an well defined operator which performs graph node localization,we found a spectral approach that accomplishes the task of localizing via a Kronecker delta associated convolution , implementation of which can be found in its spectral domain[14]. In the Convolution theorem we can see that the a Fourier basis is constituted from the Laplacian operator[15] .The basis vectors actually are the eigenvectors of the aforementioned operator.The Convolution operator mentioned here is linear in nature , which performs diagonalizaion pf the Fourier basis mentioned above.The main disadvantage of this translation is that the multiplication is of $O(n^2)$ when performed with Fourier basis.And in practise the spectral domain filter is not always naturally localized.Both this drawbacks and limmitations can be addressed with appropriate parameterization of Filter.

Fourier transformation of Graph :- we want to define signal processing in an connected and undirected graph $G = (V,E,W)$.V is the set of vertices And $|V| = n$. E is the undirected edge set.$W=R^{n*n}$ is the adjacency matrix denoting the edge index and edge weightes .Suppose x is a vector which represents the values of the graph nodes x_i.Then we define the Graph signal by $x \in R^n$ which is a mapping $x : V \dashrightarrow R$ .Graph Laplacian is the crucial operator of spectral graph analysis[16]. we define [16] graph Laplacian , by the notaion of combinatorial and normalized . The first of which is $L = D - W$, Here $D \in R^{n*n}$, which is the degree matrix(and essentially diagonal) defined by $D_{ii} = \Sigma_j W_{ij}$, and so $L \in R^{n*n}$ .And to define the normalized one we say $L = I_n - D^{-1/2} W D^{-1/2}$ in which I_n is an n*n identity matrix.We can identify the nature of the matrix L as an positive real symmetric And semidefinite one.The two features of this matrix are of our interest. 1. The eigen vector set of this matrix .It is a set of orthonormal vactors of cardinality $|V|(=n)$.So it's a complete set $\{u_l\}_{l=0}^{n-1} \in R^n$ and is the modes of the Graph Fourier.Consequently if we consider their eigen values(non-negetive real and associated) that is $= \{\lambda_l\}_{l=0}^{n-1}$ they are the Graph frequencies.Now we observe that the Laplacian is diagonalized as $L = U \Lambda U^T$ by the Fourier basis $U = [u_0, . . . , u_{n-1}] \in R^{n*n}$ .Here Λ is a diagonal matrix $\mathrm{diag}([\lambda_{0,....} \lambda_{n-1}]) \in R^{n*n}$. Thus we define the Fourier transformation Of a graph signal $x \in R^n$ as $x\hat{} = U^T x \in R^n$ the inverse of which is defined as $x= Ux\hat{}$ [14].This transforms emmarges Some fundamental operations of Graphs on Euclidean spaces, such as formulation of Filtering and etc.

C2 .filtering of graph signals :- We dfine the graph convolution operator in the fourier domain as $*G$.Where $x *G\, y = U((U^T x) \; @ \; (U^T y))$ .Here we use @ as an notation of an element wise product known as Hadamard product.The operator we denote as $g\theta$ filters the signal x as

$$Y = g\theta(L)x = g\theta\,(U \Lambda U^T)x = U\, g\theta\,(\Lambda)\, U^T x \qquad (9)$$

We introduce here the notion of a non- parametric filter , parameters of which are all free. This is formulated as $g\theta\,(\Lambda) = \mathrm{diag}(\theta)$, the θ here is a Fourier coefficient vector and $\theta \in R^n$

C3 Polynomial parametrization for localized filters : -

However we face two issues rather limmitations while usingThe parameter free filters,one is that these filters are not space-localized and two their complexity of learning is proportional To the dimention(n) of the data that is O(n).Therefore we address these issues with the notion of a polynomial filter ,

$g\theta(\Lambda) = \Sigma_{k=0}^{K-1} \theta_K \Lambda^K.$ (10)

Here θ is the polynomial coefficient vector of dimension k.When the filter is centered at ith vertex ,The value of the same at vertex j is formulated by :-

$(g\theta (L)\ \delta i)_j = (g\theta(L))_{i,j} = \Sigma_k \theta_K(L^K)_{i,j}$ (11)

Here the kernel localization is performed with a delta function Kronecker and finally achieved via a convolutionWith this delta function. $\delta_i \in R^n$. BY [17] we can see here that d_G denotes the shortest path(min number of edges) distance and $K< d_G(i,j)$ implies that value of $(L^K)_{i,j}=0$[Lemma 5.2]. Consequently the spectral filters which Are exactly K-localized are implied by K^{th} order Laplacian polynomials.By this we mange to reduce the learning complexity to O(K) like the classical CNNs , where K is basically the filter-support-size.

C4.Recursive formulation for fast filtering : We have minimized the complexity of learning localized filters to K parameters , the cost of filtering a signal x as $y = U\ g\theta\ (\ \Lambda\ )\ U^Tx$ is still involved with the multiplication by the Fourier basis U and thereby sustaining with the High complexity of $O(n^2)$.To address this issue we make gθ(L) recursively computable by parameterizing it as a polynomial function because we realize that multiplicating K times by L(sperse) will cost O(K|E|) which is much lesser than $O(n^2)$.We here use a state of the art method for approximating kernels is called Chebyshev expansion[17].Naturally we denote a Chebysheb polynomial by $T_K(x)$ which denotes its order as k and it is calculated by the reccurence relation as follows $T_K(x) = 2xT_{K-1}(x) - T_{K-2}(x)$ and the starting values of recurrence are provided as …$T_1 = x$ and $T_0 = 1$.As we know $L^2([-1,1],dy/sqrt(1-y^2))$ Is the Hilbert space . The Cheb Polynomials form an orthogonal basis For this space of square integrable functions we mentioned above measured wrt $dy/sqrt(1-y^2)$.With the help of above information we parameterize a filter of order K-1 where the parameter(θ= Chebyshev coefficient vector) $\in R^K$ [49] As $g\theta(\Lambda) = \Sigma_{k=0}^{K-1} \theta_K T_K(\Lambda^\sim)$, (12)

Here $T_K(\Lambda^\sim) \in R^{n*n}$ is the Chebyshev Polynomial (order = K) which is evaluated at $\Lambda^\sim = 2\Lambda/\ \lambda_{max} - I_n$

We realise that $\Lambda^\sim$ is a matrix of scaled eigenvalues .Also This matrix is diagonal and the eigen values lie in [-1,1].Now that the formulaes are built and notations are defined , we define the filter formulae as[49] :

$y = g\theta(L)x = \Sigma_{k=0}^{K-1} \theta_K T_K(L^\sim)x$ 13)

Here we define the notation $T_K(L^\sim)$ as the order K chebyshev polynomial and its $\in R^{n*n}$ .This is evaluated at$L^\sim = 2L/\ \lambda_{max} - I_n$, the scaled Laplacian.Also this formulation denots that $x^-_K = T_K(L^\sim)x$ is a vector in R^n.So we can compute with the help of reccurence relation that $x^-_K = 2L^\sim x^-_{k-1} - x^-_{k-2},$ once the starting values $x^-_1 = L^\sim x$ and $x_0^- = x$ are provided.This whole operation of filtering we discussed which in a nutshell is $y = g\theta(L)x = [x_0^-,……x^-_{k-1}]\theta$ .The operation cost of which we thus managed to reduce to O(K|E|).

C5 . Learning Filters :- Let us consider a sample s , and denote $x_{s,i}$ as the feature maps of the input ,also consider $F_{in} * F_{out}$ vectors as Chebyshev coefficient vectors and $\theta_{i,j}$ of K dimension in R ($\theta_{i,j} \in \mathbb{R}^n$) as trainable parameters of the layer , Now the feature map of the j th output of s is given by[49] ;

$$Y_{s,j} = \Sigma_{i=1}^{Fin} \quad g\theta_{i,j} (L)x_{s,i} \in \mathbb{R}^n \tag{14}$$

Now at the time when we come in a situation of training multiple layers of convolution, the backpropagation calculates two gradiants[49] ,

One is , $\partial E/\partial\theta_{i,j} = \Sigma_{s=1}^{S}[x_{s,i,o}^{-},......x^{-}_{s,i,K-1}]^{T}\partial E/\partial y_{s,j}$ (15)

and the other one is,

$$\partial E/\partial x_{s,i} = \Sigma_{j=1}^{Fout} g\theta_{i,j} (L)(\partial E/\partial y_{s,j}) \tag{16}$$

Here we partial derivate the Loss Energy denoted by E which is measured over a S sampel mini batch.To discuss about the operation complexity of the above three operation we observe that they each involve in sparse vector matrix multiplications (K times) and a single dense vector matrix multiplication and the total operation cost comes out to be $O(K|E|F_{in}F_{out}S)$.Though $[x_{s,i,o}^{-},......x^{-}_{s,i,K-1}]$ needs only one time to be computed , it is important to note that exploiting the tensor features and operations , we can create and run these on parallel architecture , ie perform efficient computation simultaneously.

Top level view of the problem and the proposed method: We consider N different geographical regions (may be different states different contries).We collect and observe the new COVID-19 cases data and curves separately for each of these regions .Our task now is to forecast The number of new COVID-19 cases in each of the considered regions in M Timeunit advance.We can use month or days or weeks in place of M .In our problem we take M = 7 days.We have created a graph structure with number of nodes equal to the number of Geographical areas we considered and the structure of the graph will preserve the geographical orientation of the regions .With each node representing the unique regions , we have associated a feature vector with each of the nodes in graph.Thus we create sliding window time series snapshot each repsrsented by graph structure.These time snaps We feed to our model which consists a total of six layers of graphSAGE convolution and there exists an Graph Convolutional LSTM module within the GraphSage layers ,with the liner transformation replaced By gates of spectral Graph convolution operator .Finally we pass the final output of the model through some fully connected layers.There is an skip connection integration between Graph Convolutional LSTM and first three GraphSAGE Layers .

Formulating The Problem :

In our paper we want to predict new cases of a number of different regions For some number of days(continuous) in future ,Provided a time series sequence of previous daily cases of those regions.In our Paper the Regional Time Series Sequence of Previous daily COVID -19 cases is represented as Graph data Structure Where , Each Graph Represents a particular day in input Time series sequence And Graph Nodes represents unique regions in consideration,with node featuresDenoting new cases at a particular region at a particular day.To Denote This problem Mathematically we take graph G as (X, A, W).Here $X =\{x_{ijt}\} \in R^{K_X \times N \times T}$, given K_X is the input features N is number of different Nodes(regions) and T is the Number of Timestpes in our input Time Series sequence of Graph Timesnaps. $X_t \in R^{K_X \times N}$ Denotes the state at t th timestep.The adjacency matrix A Is unchanging(constant) over different time steps ,lastly $W \in R^{K_W \times N \times N}$ denotes the static edge features(each of total K_W)(here we use only one, ie. distance between different regions) that we implement between each pair of regions(nodes).So our problem is basically a Sequence Modelling problem , where let us suppose A set D denotes some set containg observed features ,making D the domain of same ,An observation at t th time is given by x_t consequently which is a subset of D , and we are to be provided with J number of ovservations in previous ,and we need to predict A K length sequence in future which is most likely.ie

$$\hat{x}_{t+1},\ldots,\hat{x}_{t+K} = \underset{x_{t+1},\ldots,x_{t+K}}{\arg\max}\, P(x_{t+1},\ldots,x_{t+K} \mid x_{t-J+1},\ldots,x_t) \qquad (2)$$

The probability of appearing of the the word x_{t+1} Which is further conditioned on past sequence of J words is modelled by $P(x_{t+1} \mid x_{t-J+1},\ldots,x_t)$.The Quintessential application stands out to be a language model of n(where n=J+1)-gram where such conditional probability is modelled by $P(x_{t+1} \mid x_{t-J+1},\ldots,x_t)$ in the given sentence [1] .IN our paper we are dealing with spatio -temporal sequences of structures where given any observation x_t (this is basically the graph signal at time stamp t)the features of which are dependent in terms of pairwise relation and these kind of linking(relations) are modelled with weighted graphs generally.To define a graph first Let us denote the components .We denote the set of graph nodes As a finite set V where $|V|= n$ .The edge set as E and finally we describe the weighted node connectivity of the graph by a adjacency matrix ,Entries of which shows the existence and weight of the connection between any two nodes in the graph.And science there are n nodes in graph $A \in R^{n*n}$.Now we define x_t as a signal from an weighted and undirected graph $G = (V,E,A)$.The signal is basically a mapping from The domain set V to a d_x dimensional space and the mathematical structure of the signal defined on the graph vertices is indeed a matrix existing in R^{n*d_x} where we can find the signal x_t at i^{th} node existing in the i^{th} column of the matrix as a vector of d_x dimension.We are lavaraging the fact that in a K length structured sequence the free variable number are bounded to $O(n^K d_x^K)$ to utilize probable prediction space structure and there by bring down the dimensionality and consequently turn those problems more controllable in large scale .

Objective :- In form of graph Structures we would be given L number of timesteps denoting L different timeshots of daily COVID-19 cases of the selected regions , namely $\{X_{t-L+1},\ldots,X_t\}$ Our task is to Build a model That will predict number of new COVID-19 cases in future ,that is M days after , $\hat{X}_{t+M} \in R^N$.In input timeseries , each Graph represents timesnap for a particular day,helping us to understand and predict New COVID-19 cases after M days.

D.Proposed Method : We want to explore the spatio-temporal relationships in the time series data of COVID-19 Cases,for that We have used GraphSage[32] where we have used edge-features in the layers in Combination with Graph Convolutional LSTM [50] where we have replaced linear operations of the layer(s) by spectral graph convolutions.

D1. CONVOLUTION ON GRAPHS :-

Our focus is to generelize CNN(convolutional Neural Network) to random graph structures. We go through two discovered methodes 1.Define the spatial convolution in a generalized way for adopting with arbitrary graph structures [6],[7]. 2.Adopting the convolutional theorem in Fourier domain of graph to apply multiplication [8],[9].[7] generalize the CNNs to three dimensional meshes in spatial domain. The authors have described the method of convolutional procedure in three dimensional mesh patches by means of polar co-ordinates of geodesic,which in turn contributed in formulating deep learning infrastructures that permits comparison through different meshes.Where as the spatial approach showed by [6] may be characterized in three stapes

1. Selection of a Graph Network vertex.
2. Compose the structure of neighbourhood of the selected vertex.
3. Finally perform ordering of the neighbouring node constructed in previous step in order to normalize the thus formed subgraph structure.

Now the patches are drawn out and given to a one dimensional Euclidean Convolutional Neural Network as input.Its important to note that Graph structures in general, Does not contain any spatio-temporal ordering for that we should introduce some labelling mechanism to the graph nodes.The spectral modelling framework was introduced by [9],We have discussed some relevant points below with reference to Graph Convolutional Neural Network[9]. We notice that this method has $O(n^2)$ operation cost ,which we manage to bypass with the technique of [8] where we note the rigorous use of localized filters which results in a reduced $O(|E|)$ operational cost(which is linear).Our use case is centric to the [8] framework .Our proposed method will be skeptic to the selection of the Graph Convolutional operator that is $*G$. We observed It is quiet a high complexity operation to formulate a consequential translation operation in the domain of the graph Nodes[6],[9].We refer to [8] where we notice a spectral expression for the Graph Convolution operation .From this we come to realise the idea of a graph signal that it is a $n*dx$ dimensional matrix vector in R .This signal is filtered by a Parametre-free kernel as follows,

$$y = g\theta *Gx=g\theta(L)x= g\theta(U\Lambda U^T)x=Ug\theta(\Lambda)U^Tx = R^{n * d}{}_x \qquad (4)$$

Here θ is a n dimensional vector in R , And $g\theta(\Lambda) = diag(\theta)$ is the parameter free kernel which filters the Graph Signal, U is a eigen vectors matrix of $n*n$ in R . Consequently The graph laplacian is normalized as $L = In - D^{(-1/2)}AD^{(-1/2)}$ and is diagonalized as $U\Lambda UT$ which is a $n*n$ matrix vector(In = Identity,D = degree of $n*n$) , The eigen values of which is denoted by the diagonal matrix $\Lambda \in R^{n*n}$. Here $D_{ii} = \Sigma_j A_{ij}$ [10].Here evaluating this formulation we face A high operational cost, First we can observe that Large graphs will have excessive operation expensive computation while eigendecompositioning L and second because we have to perform an per- element multiplication (to filter the signal x by operator $g\theta$) of the Fourier transformation of the graph ie U^Tx with the operator $g\theta$ [11] . This multiplication has an operation cost of $O(n^2)$.To avoid this high operational cost We refer [8] where we can observe the trimmed parametrization of $g\theta$ as a order K-1 truncated expansion of Chebyshev polynomial such that ,

$$g\theta(\Lambda) = \Sigma^{K-1}{}_{K=0}\ \theta_k T_k(\tilde{\Lambda}) \qquad (5)$$

Here T_k is the notation for Chebyshev Polynomial , θ is a K dimensional chebyshev coefficient vector in R and Finally $T_k(\tilde{\Lambda})$ is a order K Chebyshev polynomial in R^{n*n} which is evaluated at $\tilde{\Lambda}$ (which equals $(2\Lambda/ \lambda_{max}) - I_n$). Thus we can write the filtering operation of graph as :-

$$y = g\theta * Gx = g\theta(L)x = \Sigma^{K-1}{}_{K=0}\ \theta_k T_k(\tilde{L})x \qquad (6)$$

$T_k(\tilde{L})$ is a order K Chebyshev polynomial in R^{n*n} which is evaluated at $\tilde{L}$ (the scaled laplacian which equals $(2L/ \lambda_{max}) - I_n$). This operation above we mentioned one can realise is a strictly localized (K - localized) filtering operation which is an order K polynomial of the laplacian .Because of this the localized filtering Relies on only the nodes at K hops away range in max from the central node. Consequently The last equation can run in linear complexity with the edge numbers of a graph (ie $O(K|E|)$)once the starting values $T_1 = x$ and $T_0= 1$ are provided with the recurrence

$T_K(x)=2xT_{K-1}(x) - T_{K-2}(x)$.

D2. RNN : - Now we refer to ,at [12] who introduced how to build models to learn dependencies in general sequences that are grid structures.The graph here is treated as an image grid of well ordered vertices.We relaise this model is basically the classical Fully connected Long Short term MemoryCell(3) where the matrix multiplication involving dense matrix W is substituted with two dimensional kernel convolutions with W(kernel-set).(denoted by *)

$i = \sigma(W_{xi} * x_t + W_{hi} * h_{t-1} + w_{ci} @ c_{t-1} + b_i)$

$f = \sigma(W_{xf} * x_t + W_{hf} * h_{t-1} + w_{cf} @ c_{t-1} + b_f)$

$c_t = f_t @ c_{t-1} + i_t @ \tanh(W_{xc} * x_t + W_{hc} * h_{t-1} + b_c)$ (7)

$o = \sigma(W_{xo} * x_t + W_{ho} * h_{t-1} + w_{co} @ c_t + b_o)$

$h_t = o @ \tanh(c_t)$

Observing their experiment setting [12] , we realize that The spatial region here is represented by a $n_r * n_c$ matrix, over this region A Total of d_x measurements of a dynamical system are observed at t^{th} time stamp And this observations are expressed in terms of x_t which is the input tensor to the model Of dimension $n_r * n_c * d_x$ in R.The cell and hidden states of this model is distributed spatially and are of size d_h and they reside as tensors of dimensions $n_r * n_c * d_h$ in R denoted as c_t and h_t respectively.The kernels of convolution are denoted by W_h and W_x and they have m number of parameters which are not dependant(free) of the size $n_c * n_r$ of the grid denoting their dimension as $m*m*d_h*d_x$.To generelize the aforementioned convolutional LSTM model on graphs , The * operator ie the Euclidean two dimensional convolution is substituted by the graph convolutional operator ie *G mentioned earlier(6).

$i = \sigma(W_{xi} *G\, x_t + W_{hi} *G\, h_{t-1} + w_{ci} @ c_{t-1} + b_i)$

$f = \sigma(W_{xf} *G\, x_t + W_{hf} *G\, h_{t-1} + w_{cf} @ c_{t-1} + b_f)$

$c_t = f_t @ c_{t-1} + i_t @ \tanh(W_{xc} *G\, x_t + W_{hc} *G\, h_{t-1} + b_c)$ (8)

$o = \sigma(W_{xo} *G\, x_t + W_{ho} *G\, h_{t-1} + w_{co} @ c_t + b_o)$

$h_t = o @ \tanh(c_t)$

Here W_h and W_x are the Chebyshev coefficients residing in $K* d_h*d_h$ and $K* d_h*d_x$ dimension of R respectively where K is the support which decides how many parametres (K) of the Graph Convolutional kernel will be independent of the |V| ie number of vertices in the graph .In the formulation the graph convolution of signal x_t with the filters(these filters are graph Laplacian(L) functions which are as stated in (5) and (6) , parameterized by coefficients of Chebyshev polynomial(Total K co-efficient)) $d_h d_x$ is expressed as $W_{xi} *G\, x_t$ .K controls for any given specific vertex lets say i , how many exchange should happen to The node i to compute its local states.

D3.Spatio -Temporal RNN : we have used a special class of RNN(recurrent neural networks)Architecture,ie LSTM.The main reason behind this is to maintain the spatial structure we got in our input data and with that , we want to model the spatio temporal dependencies jointly.LSTM keeps tracks of long and short term dependencies in data By using a sequence of gates[27].The weights in these gates are implemented by applying matrix multiplication,But in our problem we have slightly altered this linear operation scheme by spectral Graph convolution (replaces matrix multiplication),this is called chebyshev spectral graph convolution.If a graph snapshots with its the node features at time t we denote by x then x is a graph signal $\in R^{n*d_x}$ where $n = |V|$ and d_x is the length of the embedded feature – vector(here $d_x = 1$) , then the filtering operation of this graph signal by Chebyshev spectral graph convolution[49] is expressed as(we can refer (13)) :-

$y = g\theta * Gx = g\theta(L)x = \Sigma^{K-1}_{K=0} \theta_k T_k(L^{\sim})x$ (18)

Where $T_k(L^{\sim})$ is a order K Chebyshev polynomial in R^{n*n} which is evaluated at $L^{\sim}$(the scaled laplacian which equals $(2L/\lambda_{max}) - I_n$).Here we can see the trimmed parametrization of gθ as a order K-1 truncated expansion of Chebyshev polynomial such that , $g\theta(\Lambda) = \Sigma^{K-1}_{K=0} \theta_k T_k(\Lambda^{\sim})$ (19) Here T_k is the notation for Chebyshev Polynomial , θ is a K dimensional chebyshev coefficient vector in R and Finally $T_k(\Lambda^{\sim})$ is a order K Chebyshev polynomial in R^{n*n} which is evaluated at $\Lambda^{\sim}$(which equals $(2\Lambda/\lambda_{max}) - I_n$).In above equation Λ is replaced by L.This operation above we mentioned one can realise is a strictly localized (K -localized) filtering operation which is an order K polynomial of the laplacian .Because of this the localized filtering Relies on only the nodes at K hops away range in max from the central node. Consequently The last equation can run in linear complexity with the edge numbers of a graph (ie O(K|E|))once the starting values $T_1 = x$ and $T_0= 1$ are provided with the recurrence $T_K(x)=2xT_{K-1}(x) - T_{K-2}(x)$. Thus the Graph Convolution is defined,now we implement the Graph convolution in the gates of LSTM[12],[51] replacing the dense matrix multiplications as ,

$i = \sigma(W_{xi} *G\ x_t + W_{hi}*G\ h_{t-1} + w_{ci} @ c_{t-1} + b_i)$ …..input gate

$f = \sigma(W_{xf} *G\ x_t + W_{hf} *G\ h_{t-1} + w_{cf} @ c_{t-1} + b_f)$ ……..forget gate (20)

$c_t = f_t @ c_{t-1} + i_t @ \tanh(W_{xc}*G\ x_t + W_{hc}*G\ h_{t-1} + b_c)$ …….cell state

$o = \sigma(W_{xo}*G\ x_t + W_{ho}*G\ h_{t-1} + w_{co} @ c_t + b_o)$……..output gte

$h_t = o @ \tanh(c_t)$ ……hidden state

In the above equations x_t is the input graph signal, The * operator ie the Euclidean two dimensional convolution used in [12] is substituted by the graph convolutional operator ie *G mentioned earlier(18),(19). Here @ is the element wise Hadamard product, W_h and W_x are the Chebyshev coefficients residing in $K* d_h*d_h$ and $K* d_h*d_x$ dimension of R respectively where K is the support which decides how many parametres (K) of the Graph Convolutional kernel will be independent of the |V| ie number of vertices in the graph .In the formulation the graph convolution of signal x_t with the filters(these filters are graph Laplacian(L) functions which are as stated in (5) and (6) , parameterized by coefficients of Chebyshev polynomial(Total K co-efficient)) $d_h d_x$ is expressed as $W_{xi} *G\ x_t$ .This modification of the LSTM architecture we reffered earlier (20) filters the input signal x_t(In our Scenario its new cases graph snapshot)

Training Workflow :-

The graph structure is defined as G = (V,E,W) ,where V is the set of graph nodes and |V| = n , E is the edge set of the structure, and W is the adjacency matrix which stores the edge connection-weight information and W is n*n in R.Let us denote the input signal by x at a particular time stamp t , here x is an n dimensional vector in R where each x_i represents value at any particular graph node(new cases in our problem).So we can say the graph signal at time t is a mapping from graph vertices set to Real vector of dimension |V|

While training we send a particular graph snapshotie. The graph signal x to a three layer graphsage convolution batch where the signal is proceed like below…

E.Model Framework :- We have used a spatial GNN namely GraphSAGE[32] which generates an embedding For each node in input graph signal x of graph G = (V, E) by aggregating (we are using mean aggregation)features of local nodes at a specified distance(hop).Where Innitial embedding for each Node is the input node feature itself.let the the ferature vector of a Particular node (namely j) be x_j, then the embedding of x_j at 0th iteration will be h^0_j .Each node v ∈V aggregates the embeddings of all 1 hop adjacent nodes (previous layer embedding of these nodes ,h^{k-1}_u , ∀u ∈ N (v)) by using a predefined aggregation method(TopKPooling,by default uses mean aggregation method).To compute the final embedding of this node at any iteration (k ∈ {1, ..., n}) the aggregated vector($h^{k-1}_{N(v)}$) is concatenated to the embedding of the centre vector(h^{k-1}_v) at previous (k-1th)itaration.Thus we generate current that is kth embedding (at kth iteration) and we pass this to a neural network layer(single) and a activation function(sigmoid) to compute the final representation value of h^k_v. We can formulate the embedding generation process as ,-

$$h^{k-1}_{N(v)} = (1 / |N(v)|) * (\Sigma u \in N(v) (e_{vu}.h^{k-1}_u)) \qquad (17)$$

This process is repeted for n number of itrations (k ∈ {1, ..., n}) for each node(from total nodes).we introduce a edge feature e_{vu}(=distance) between any two different regions(nodes)in above mentioned formulae for taking a weighted mean,when aggregating.

Thus the input graph signal is proceed in graph sage layer as ,

```
$h^0_v \leftarrow x_v , v \in V$
for k = 1 .....k do:
 for v ∈ V do :
      $h^{k-1}_{N(v)} \leftarrow (1 / |N(v)|) * (\Sigma u \in N(v) (e_{vu}.h^{k-1}_u ))$
      $h^k_v \leftarrow \sigma(W^k .CONCAT(h^{k-1}_v, h^{k-1}_{N(v)}))$
 end
$h^k_v \leftarrow h^k_v / ( ||h^k_v ||_2 , \forall v \in V$
end
$z_v \leftarrow h^k_v , \forall v \in V$
```

Now after this we send this processed input graph signal at time stamp t , ie x_t to a LSTM RNN cell where the convolution takes place like this :-

The gates of the LSTM RNN are (we can refer 8)

$i = \sigma(W_{xi} *G\ x_t + W_{hi}*G\ h_{t-1} + w_{ci} @ c_{t-1} + b_i)$

$f = \sigma(W_{xf} *G\ x_t + W_{hf} *G\ h_{t-1} + w_{cf} @ c_{t-1} + b_f)$

$c_t = f_t @ c_{t-1} + i_t @ \tanh(W_{xc}*G\ x_t + W_{hc}*G\ h_{t-1} + b_c)$

$o = \sigma(W_{xo}*G\ x_t + W_{ho}*G\ h_{t-1} + w_{co} @ c_t +b_o)$

$h_t = o @ \tanh(c_t)$

x_t is the input graph signal at time stamp t which is a tensor of dimension $n*d_x$ where d_x is the number of node features .The hidden and cell states are tensors of dimension $n * d_h$ and denoted by c_t and

h_t.Here W_h and W_x are the kernels of graph convolution and has the dimensions K * d_h * d_h and k*d_h*d_x where K is the order of Chebhyshev polynomial. In the formulation the graph convolution of signal x_t with the filters(these filters are graph Laplacian(L) functions which are as stated in (5) and (6) , parameterized by coefficients of Chebyshev polynomial(Total K co-efficient)) $d_h d_x$ is expressed as $W_{xi} *G\ x_t$, the process of which is mentioned below.

Now the essential component of graph spectral filtering ie $W_{xi} *G\ x_t$ operation is The Laplacian of the graph structure, which we need to process x.(the graph signal).The Laplacian of the graph is denoted by L and computed as $L = I_n - D^{-1/2}WD^{-1/2}$ .Here D is a n*n diagonal matrix in R , the degree matrix of the Graph.So $D_{ii} = \Sigma_j W_{ij}$ .The set of eigen vectors $U = [u_{0,\dots\dots}u_{n-1}]$ of L is of cardinality n they are n dimensional vectors in R ie u_l l is from 0 to n-1 .This eigen vector set has a corresponding ordered eigen value set $\Lambda[\lambda_{0\dots\dots}\ \lambda_{n-1}]$ where elements are λ_l(real and positive) where l is from 0 to n-1.The Laplacian is diagonalized by U and Λ such that $L = U\Lambda U^T$ .Here U and Λ(diagonal matrix) is n*n . The input graph signal x is now filtered as $y = g\theta\ (L)x = \Sigma_{k=0}^{K-1} \theta_K T_K\ (L^\sim)x$ ....(A).Here T_k denotes and order k Chebyshev polynomial , which is calculated via a recurrence relation ie $T_k(x) = 2xT_{k-1}x - T_{k-2}x$,where we initialize T_0=1 and T_1 = x. θ is a K dimensional vector of Chebyshev coeficients in R. ## Here $T_k\ (L^\sim)$ is a matrix vector of R^{n*n} which is an order k Chebsyhev polynomial .The polynomial is evaluated at $L^\sim = (2L/\ \lambda_{max}) - I_n$ which is a scaled version of graph Laplacian.The vector $T_K\ (L^\sim)x(=x^-_k$ say) is n dimensional in R and we calculate kth value using recurrence,:- $x^-_k = 2L^\sim x_{K-1} - x^-_{k-2}$; where $x^-_0 = x$ and $x^-_1 = L^\sim x$.Finally we get the filtered signal y which is again a n dimensional in R.

After receiving this proceed signal we pass this signal through Multi Layer perceptorns to receive the final output.(the filtered graph signal with dimention |V|)

E1.MultiLayer Perceptorn :- The 0th (first) layer of a Multi Layer Perceptorn [53] receives the D dimensional input vector (D=|V|) say x(this is the convoluted graph signal at time step t in our case ie x_t)And performs liner combinations ,say M numbers then we can represent resultant vector as ,

$$b_j = \Sigma^D_{i=0}\ W^{(1)}_{ji}x_i \quad j= 1,2\dots.M \tag{21}$$

the components b_j s are known as activations,here in the input signal x_0 is the bias and initialized to 1 and w_{ji}^1(here the superscript denotes which number of the layer is) s are corresponding weights.Next a non linearity is applied to these activations which transfoms the signal as,

$z_j = h(b_j) = 1/(1+\exp(-b_j))$

The outputs that is the z_j components are the hidden unit outputs and they are again linearly combined which results in the output unit activations.so if there are K output units Then the output layer activations parameterized by $w^{(2)}_{kj}$ (weight matrix)are ,

$$a_k = \Sigma^M_{i=0}\ W^{(2)}_{kj}z_j \quad k = 1,2\dots.K \tag{22}$$

where z_0 is the additive bias ,and the a_k components are the activations of second layer which in turn is transformed by sigmoid non linearity as,

$y_k = g(a_k) = 1/(1+\exp(-a_k))$

Thus we can describe the forward propagation of the Multi Layer perceptorn(we are using 2 layer in our case) and express the transformation of input graph signal of dimension D From input layer through hidden layers and to the out put layer as

$y_k = g(\Sigma^{M}_{i=0} W^{(2)}_{kj} h(\Sigma^{D}_{i=0} W^{(1)}_{ji} x_i))$ (23)

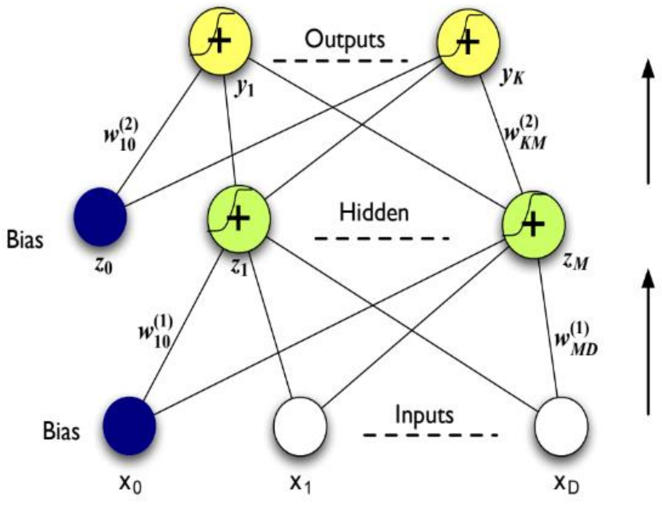


Figure 7. MLP network feed forward diagram with 1 hidden layer .[55]

E2.Skip Connection :- We have used the notion of Skip-Connection in our model .To discuss overall structure of our model ,we have two GraphSage layers (each with num_layers =3).The output of the first GraphSAGE layer is fed to a GraphLSTM(Graph convolution embedded),which is followd by the final GraphSAGE layer(num_layer=3).The output of the final GraphSAGE layer is passed through two mlp(multi-layer-perceptorn) to generate the final prediction.Because of this Skip-Connection feature we used between these GraphSAGE layers , the final GraphSage layer is feeded(or receives) a concatenated vector of the outputs of the first GraphSAGE(num_layers=3)And the GraphLSTM.This feature helps in model speed stabilization and reduces bias as well as increases variance(underfitting).

E3.Additional Arcitecture :- To the proposed arcitecture above we have added some Machine Learning TechniquesTo both of our GraphSage Layers ,(initial and final) they are ,

I.**Droupout**:- droupout helps in increasing bias and reducing variance by disabling Graph Nodes in a random fashion,there by increases robustness of the model[33].If we denote a neural network hidden layers indexed by $l \in \{0,....L-1\}$,and the input vector And output vector of any layer l as z^l and y^l respectively,Then the feed forward equation of a neural network layer(y^0=x) as we know it is ,

$Z_i^{(l+1)} = W_i^{(l+1)} y^l + b_i^{(l+1)}$

$Y_i^{(l+1)} = f(z_i^{(l+1)})$

Where f is activation ,$W^{(l)}$ and $b^{(l)}$ at any layer l is the respective weights and biases. Now after applying dropout this feed forward is modified to ,

$Z_i^{(l+1)} = W_i^{(l+1)} y^{\sim l} + b_i^{(l+1)}$

$Y_i^{(l+1)} = f(z_i^{(l+1)})$

where $r^{(l)}_j$ ~ Bernoulii(p). From this probability distribution of Bernoulli the independent random variables generates the vactor $r^{(l)}$ at layer l where each component has the probability of existing 1(p=1) .This probability distribution resulted vector is used to thin the output of layer l ie $y^{\sim(l)}$ by an element wise multiplication with $r^{(l)}$,which is used as next layer input and so on.

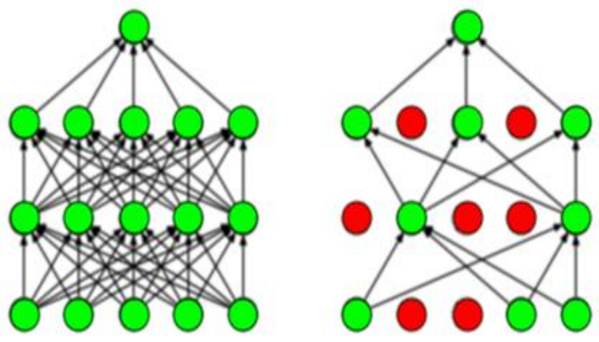


Figure 8 . neural network without dropout and with droupout.[56]

Along with overfittng prevention ..this dropout layers(temporarily removes selected neural network layer) in neural network,efficiently combines various different neural networks as in our case.Each Neural network layer can be retained with a different probability which are not dependant on other's.By default it is 0.5

II.**residual connections** :- The out put of previous layer is input of next in standard neural networks.Now this residual connection technology[34] provides an alternate path for the data to reach any layer skipping some previous. In traditional neural network feed forward method layer any data x at any input layer say i will propagate through all the layers upto nth layer i+n, and let the outcome thus be F(x).The residual connection on the other way first it maps the input at layer I with identity mapping or linear transformation depending on dimension of F(x) and x.Then it adds F(x) with x(or Wx if liner transformation is performed) elementwise .

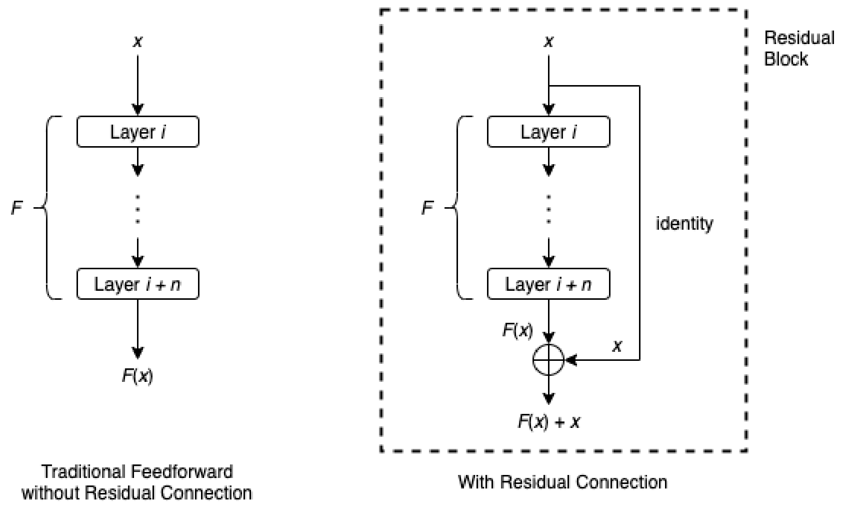
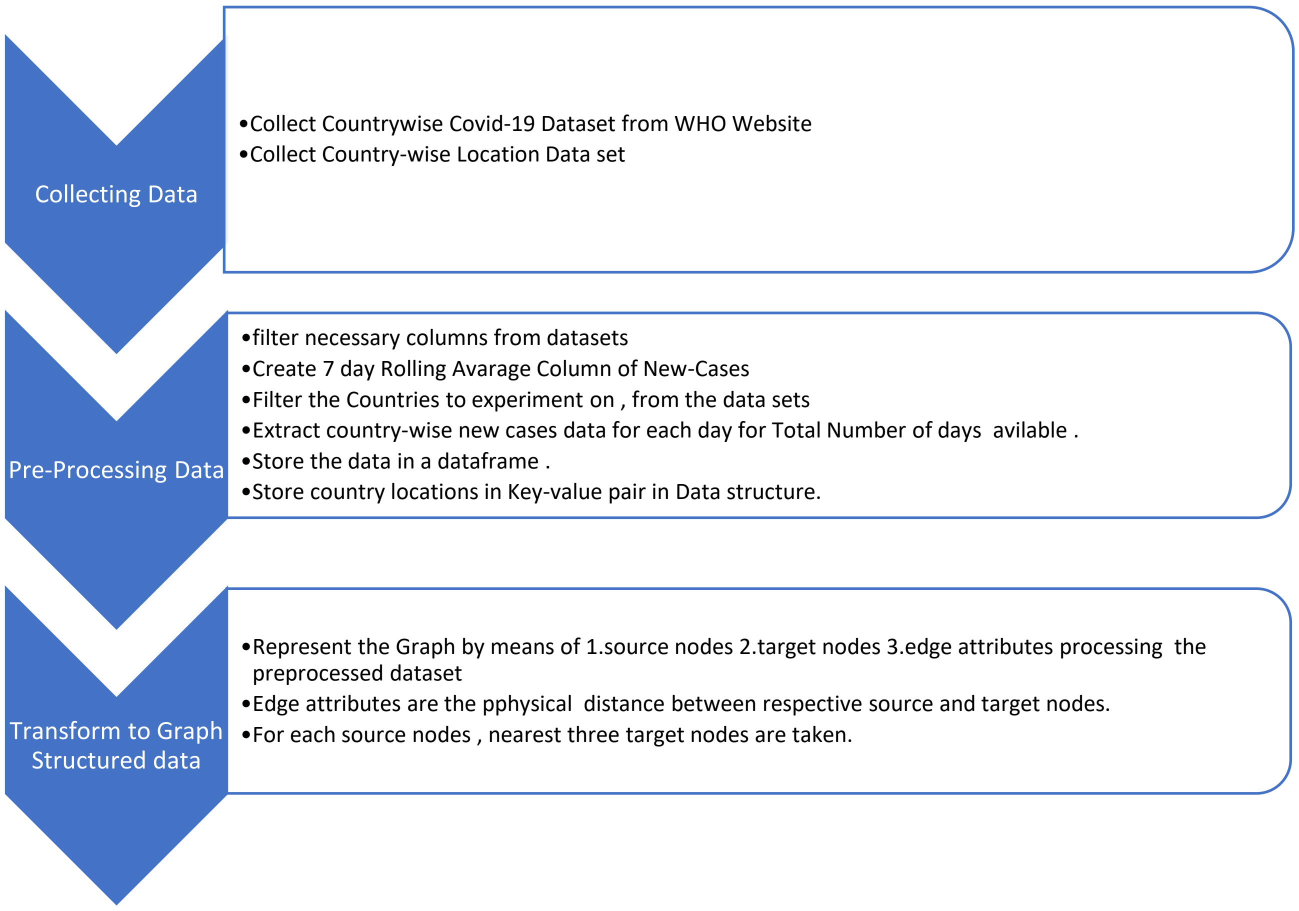


Figure 9. Classical neural network and neural network with residual connection.[57]

After this we apply a nonlinearity to F(x) + x.Residual connections helps neural networks converge quickly by adrresing the exploding and vanishing gradient problems.The neural networks with with residual connections works like network ensembales[52] because the layers are not independent that much like in the standard one.Because of this during gradient decent in an residual connection neural network layer maximum of the gradients flows through the residual paths thus addresses the gradient problems by means of shallowing the architecture.Thus We solve the issue of vanishing gradient by the concatenation of Raw inputs to the Posterior layers, with the help of this feature.

Data processing flow and model Innitialization:-


Collecting Data
•Collect Countrywise Covid-19 Dataset from WHO Website
•Collect Country-wise Location Data set
Pre-Processing Data
•filter necessary columns from datasets
•Create 7 day Rolling Avarage Column of New-Cases
•Filter the Countries to experiment on , from the data sets
•Extract country-wise new cases data for each day for Total Number of days avilable .
•Store the data in a dataframe .
•Store country locations in Key-value pair in Data structure.
Transform to Graph Structured data
•Represent the Graph by means of 1.source nodes 2.target nodes 3.edge attributes processing the preprocessed dataset
•Edge attributes are the pphysical distance between respective source and target nodes.
•For each source nodes , nearest three target nodes are taken.

Creating Time Series data

- We take a window of country-wise 0th day to 20th day new-cases data as lookback day , and 27th days of same prediction day.
- Now we slide the window one day and repeat the same.
- With each of this temoral informations we add the spatial information (ie the graph structure fromed earlier) and treat each sliding window as a particular timesnap of country-wise Covid-19 cases .
- Finally te graph timeseries is formed where the countries are the nodes ,new-cases is node feature and distances are edge attributes.
- we save this graph structured time-series as the final processed dataset.

Split Dataset into train,validation,test

- We take 80% of the dataset for traing our model
- 10% to validate our model
- 10% to test our model.

Building the Model GraphSageBatch1

- Our model Consists of,
- First a batch of Graph Sage convolutional Layers .
  - Construction of the first batch of three Graph Sage layers(num_layers= 3) :-
  - We innitialize the first graph sage batch by adding a graph sage convolution layer to the hidden module list as
  - input channels= number of input node features = 1 (new_cases) , output channels = length of hidden embedding vactors = 16
  - we add two more graph sage convolution layers with input channels = output channels = length of hidden embedding vactors
  - (input out put dimensions are considersd per node)

Building the Model
LSTM RNN

- Now attach an RNN(LSTM) componant with depth=1 and convolution operator = Graph Convolution(Chebyshev) as,
- in channels = number of input features = output channel of last Module = 16
- out_channels =number of output features = in channels = 16., and Chebyshev Filter size(K) = 3.,normalization = symmetric, bias = True
- Thus the layers formed are(x: layer for input node features,h: layer for hidden state node features), :-
- input gate layers, forget gate layers ,
- (conv_x_i) : ChebConv(16,16,K=3,normalization = $I-D^{-1/2}AD^{-1/2}$) (conv_x_f) : ChebConv(16,16,K=3,normalization = $I-D^{-1/2}AD^{-1/2}$)
- (conv_h_i) : ChebConv(16,16,K=3,normalization = $I-D^{-1/2}AD^{-1/2}$) (conv_h_f) : ChebConv(16,16,K=3,normalization = $I-D^{-1/2}AD^{-1/2}$)
- Cell state layers, Output gate layers
- (conv_x_c) : ChebConv(16,16,K=3,normalization = $I-D^{-1/2}AD^{-1/2}$) (conv_x_o) : ChebConv(16,16,K=3,normalization = $I-D^{-1/2}AD^{-1/2}$)
- (conv_h_c) : ChebConv(16,16,K=3,normalization = $I-D^{-1/2}AD^{-1/2}$) (conv_h_o) : ChebConv(16,16,K=3,normalization = $I-D^{-1/2}AD^{-1/2}$)
- Input output channels are considered per node.ChebConv(size of input sample, size of output sample)(D :degree,A : adjacency of input graphsignal x_t)
- dimension of hidden state matrix and cell state matrix are = |V| * out_channels.
-

Building the Model
GraphSageBatch2

- Finally we add Second batch of Graph sage convolution with three layers with each layer as ,
- in channels = 2*number of outpput features of LSTM module =32, because here we concatenate the
- output of the first graph sage layer through skip connection.
- out channels = in_channels = legnth of hidden embedding vector = 32.
- (input out put dimensions are considersd per node)

Bulding The model
Multi Layer Perceptorn

- Finally we add Two Multi Layer Perceptorns ,
- First with in_features = out channels of last Module = 32
- out features = 16
- Second with in_features = out channels of last Module = 16
- outfeatures = length of the output vector for each node = 1

Model : Over view of our model structure is

```
GNN(
  (graphSAGEbatch1): GNNModule(
    (hidden): ModuleList(
      (0): EdgeWeightedSAGEConv(1, 16)
      (1): EdgeWeightedSAGEConv(16, 16)
      (2): EdgeWeightedSAGEConv(16, 16)
    )

  )
  (recurrent): Long Sort Term Memory(
    (conv_x_i): GraphConvolution(16, 16, K=3, normalization=sym)#inp
ut gate for node features
    (conv_h_i): GraphConvolution (16, 16, K=3, normalization=sym)# i
nput gate for hidden state
```

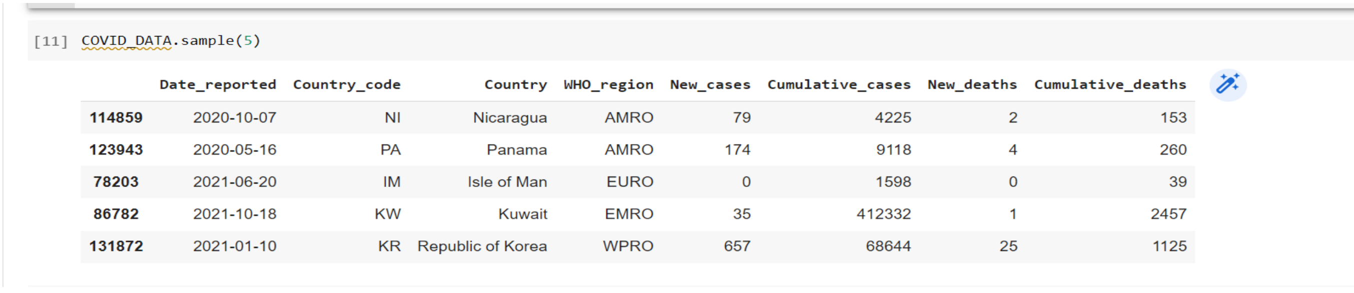
```
    (conv_x_f): GraphConvolution (16, 16, K=3, normalization=sym)#fo
rget gate for node features
    (conv_h_f): GraphConvolution (16, 16, K=3, normalization=sym)# f
orgate gate for hidden state
    (conv_x_c): GraphConvolution (16, 16, K=3, normalization=sym)# c
ell state for node features
    (conv_h_c): GraphConvolution (16, 16, K=3, normalization=sym)#ce
ll state for hidden state
    (conv_x_o): GraphConvolution (16, 16, K=3, normalization=sym)#ou
tput gate for node features
    (conv_h_o): GraphConvolution (16, 16, K=3, normalization=sym)#ou
tput gate for hidden state
  )
  (graphSAGEbatch2): GNNModule(
    (hidden): ModuleList(
      (0): EdgeWeightedSAGEConv(32, 32)
      (1): EdgeWeightedSAGEConv(32, 32)
      (2): EdgeWeightedSAGEConv(32, 32)
    )

  )
  (lin1): Linear(in_features=32, out_features=16, bias=True)# multi
Layer perceptorn 1
  (lin2): Linear(in_features=16, out_features=1, bias=True) # multi
Layer perceptorn 2

)
```

Figure 10 . proposed Model

DataSet and Training :- We predict COVID-19 cases in advance now by applying our model(with the help of [35]) to **WHO COVID-19 Data** [58] .First we Process The dataset, Then Train The model and finally explore the results.We have selected the countries of Europe as regions in consideration .WE have observed the Covid-19 dynamics in Europe [36],[37] and We have considered a total of thirty Europe contries (N=30) with a number of inhabitants more than 100 Thousands.We have used the as timeline a total Of nearly Five Hundread days window (28th jan 2020 – 31st may 2021) in our COVID-19 dataset.We use first 80% days to train our model ,for validation data we use the 10% days afterwords and the following 10% days are used to test the model.Our primary objective is to forecast the number of new COVID-19 cases (smoothed by taking a Seven days average (while processing)) with a M value of 7 , ie one week after any particular given day.To train our model , we first create snapshots of Twenty one continuous days with given covid-19 cases region wise(here L=21) and set the region-wise new COVID-19 cases of Twenty Eighth day as target.The output of Twenty First day is our Twenty one days time series prediction,where the target is Twenty Eighth day.Thus we form the timeseries data from our raw dataset by sliding window technique.We have choosen the node adjacency based on geographical distance ie , geodisc between landmass centroids[59](assending,up to number of edges we want)between two nations. We have assigned Three nations in particular to each geographical regions(Graph Nodes)based on this geo-proximity feature.This is the edge feature we have used for GraphSAGE ,so if we refer to the problem formulation ,we can see that here K_w = 1(e_{uv} = distance between two countries).

Now we Import to data set to our workspace one is with the land mass centroids of all countries in the world ,country centroids and the other is COVID-19 data set,WHO-COVID-19 .From the country centroids dataset, now we select the geographical regions to consider for our problem.Scince we are intending to experiment on European nations first , we collect landmass centroids of all European countries from country centroids file.Now we select only three features from WHO-COVID-19 file that is location date and new_cases .Now we smooth the new cases column by taking an moving average of 7 days.Now we process the WHO-COVID-19 data set by one by one Nations (the European countries we filtered) and create a data frame with first column as the dates under the time range we selected andrest of the collumns are the Geographical regions with date wise new cases(smoothed) entries.After this we store the centroid of our considered countries in a data structure for easy retrival.

Our data set before processing :-

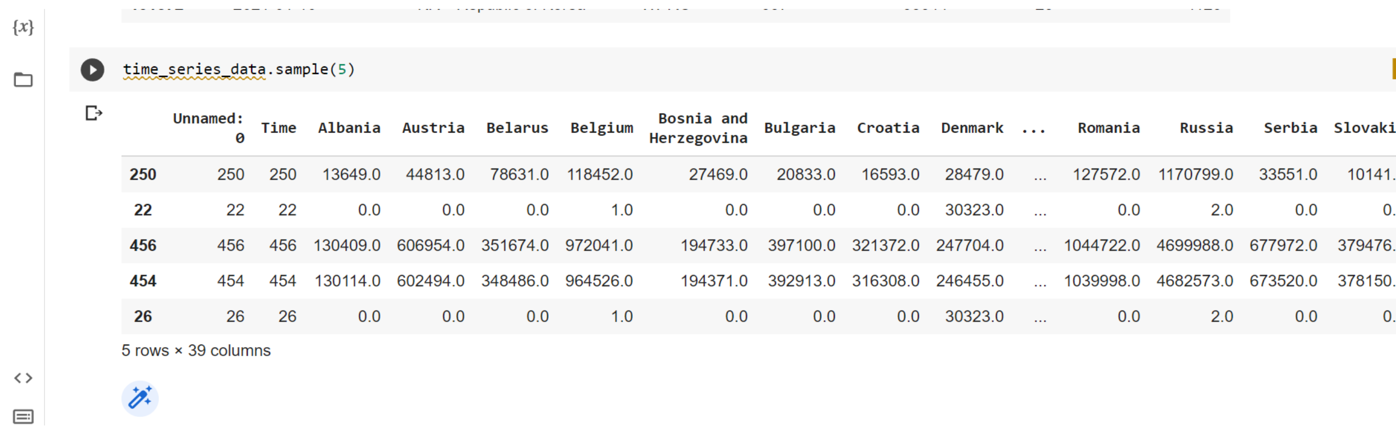


```
[11] COVID_DATA.sample(5)
```

	Date_reported	Country_code	Country	WHO_region	New_cases	Cumulative_cases	New_deaths	Cumulative_deaths
114859	2020-10-07	NI	Nicaragua	AMRO	79	4225	2	153
123943	2020-05-16	PA	Panama	AMRO	174	9118	4	260
78203	2021-06-20	IM	Isle of Man	EURO	0	1598	0	39
86782	2021-10-18	KW	Kuwait	EMRO	35	412332	1	2457
131872	2021-01-10	KR	Republic of Korea	WPRO	657	68644	25	1125

Figure 11 : Covid-19 raw dataset

After processing ..the collums and sample rows of the dataset:-



```
time_series_data.sample(5)
```

	Unnamed: 0	Time	Albania	Austria	Belarus	Belgium	Bosnia and Herzegovina	Bulgaria	Croatia	Denmark	...	Romania	Russia	Serbia	Slovaki
250	250	250	13649.0	44813.0	78631.0	118452.0	27469.0	20833.0	16593.0	28479.0	...	127572.0	1170799.0	33551.0	10141.
22	22	22	0.0	0.0	0.0	1.0	0.0	0.0	0.0	30323.0	...	0.0	2.0	0.0	0.
456	456	456	130409.0	606954.0	351674.0	972041.0	194733.0	397100.0	321372.0	247704.0	...	1044722.0	4699988.0	677972.0	379476.
454	454	454	130114.0	602494.0	348486.0	964526.0	194371.0	392913.0	316308.0	246455.0	...	1039998.0	4682573.0	673520.0	378150.
26	26	26	0.0	0.0	0.0	1.0	0.0	0.0	0.0	30323.0	...	0.0	2.0	0.0	0.

5 rows × 39 columns

Figure 12 . covid 19 processed dataset for spatio-temporal modelling.

After this we proceed towards Graph structure building .

We define A Graph data structure where the Nodes are geographical regions we considered, the node features are new cases for that day in that region and we assign three nearest region(by means of landmass centroid distance)to every unique regions .The edge features is the distance between corresponding nodes(countries)

Like Pytorch accepts[35] a Graph data structure we build three list data structure source nodes target nodes and edge attributes.In edge attributes list we put the edge feture vector of the corresponding edge , ie reffered by the corresponding entries in source and target list, (the distance between them).

source = [source , , , , , , , , , , , , ,]

target = [target, , , , , , , , , , , , ,]

edge_feature = [dist, , , , , , , , , ,]

Now we add node features.To create the node features data structure , we take a sliding window of size first 21 days and catch the new cases amount of those days fro every region in consideration.And for target feature , we take 28 th day's (as we are predicting new cases 7 days ahed (M=7)) number of new cases across all regions.By this We create One window of our time series where we have multiple graph snapshots , starting from 1st day to 21th day. Each day is represented by a particular graph snapshot Where nodes are unique geographical regions in consideration , they are connected by an edge to nearest three nodes and edge features are distances between corresponding nodes.We push this window into a list and slide our window .Like this we push all windows into our list and this is our final dataset to work with.Now that the data set in fromed we split the data set into train (we take the first 80% data and shuffle) validation(we take the next 10% data) and test (we take the rest 10% data).Now feed these datasets to our model.

When training our dataset with our model we execute this for total number of epochs innitiated,

1.we loop through the train dataset where in each iteration we retrive an enumerated tuple of timestampseries and corresponding graphsnapshot series.

2.we again loop through this touple we retrived and retrive a single timestamp and corresponding graph-snapshot.

3.we pass this graph-snapshot(ie a graph signal observation of a particular time stamp) to our model.

4 our model is named GNN .which will serve as a base class for our Graph Sage and Graph LSTM Recurrent Neural Networks. This class consists 6 GraphSAGE convolution layers with a GraphLSTM network in between.we have initialized this model as 1 feature per node ,length of output vector also 1 for each node we have selected 16 features for the embedding vector for for each node.Finally we have initialized a module as Spectral graph convolution, which will replace the linear transformation in LSTM by a Graph convolution and will be used in each gate of LSTM to calculate various states.Also we have two parametrs graphSAGEbatch1 and graphSAGEbatch2 which represents the first three and last three GraphSAGE layers , we have initialized them as DeepWeightedSageConvolution which is an object of Graph Neural Network Module from torch.nn.module with number of layers initialized to 3.

5.after receiving the graph snapshot in our Model we retrive the nodefeatures(new cases of a particular day across regions), and edge information of the graph(edge index and edge attribute)

6.Now we pass this graph snapshot to the first gnn layer by means of the retrived data at step 5.This layer is a 3 layer GraphSAGE layer(DeepWeightedSageConvolution layer)Here we first initialize residual as input node features, then we calculate residual by First padding the residual(to match matrix multiplication dimension) and then performing a matrix multiplication by first res_factor.we foroward this result to the first layer by means of an rectified linear unit added with calculated residual .and repeat these process for rest of the layers with performing dropouts first each time. The hidden layers of the afore mentioned graphsage batch is a single edgeweighted graphsage convolution layer.Thus After 3 layers of convolution we return our output .

7. after getting the out put from first batch of GraphSAGE layer we pass it through a nonlinearity .

8.Now we pass this output to a single stack LSTM where we have changed the way the gates apply weights by replacing Matrix multiplication with a single layer spectral Graph convolution.(we can refer eqn 8).Finally the LSTM cell returns the hidden and cell state matrix .

9.we catch the cell and hidden state output of the LSTM.Now Science we use skip connection , we concatenate the output of first batch of GraphSAGE layers with this hidden state and pass that to second batch of GraphSAGE layer.

10 .Finally we pass our output to first a fully connected readout layer (32*16) and then an output layer(16*1) .we return this output along with hidden state and cell state matrix of Graph LSTM to our main function .

11.we store the output of the graphsnapshot we passed at step 3 in a variable and resend next snapshot in current time window along with previous hidden and cell state matrix.

12.we use the out put of the final snapshot in the current time window as our prediction or output, and calculate severel of our metrics accordingly.

13.Finally when we iterate through all timewindow in training dataset , a single epoch completes We perform 100 such epochs , and store the observed metrics (error a or goodness report through all traing set)each time.We have used ADAM(torch.optim.adam)optimizer to update gradient.

Hyper Parametres:-

1 . We use the following Pattern of previous timesteps to predict the future day new case M(=7) day ahed.(int,int,python list object)

```
pred_timestep = 7
lookback_range = 27
lookback_timesteps = [timestep for timestep in range(lookback_range,pre
d_timestep-1,-1)]
"time_window":lookback_timesteps ,
```

2.Edges per node , that is number of countries every country is connected to. (int)

```
"EDGES_PER_NODE": 3,
```

3. Total num epochs number of passes . The training data set will be feed forward through the network this many times.(int)

```
"num_epochs": 100,
```

4. Node feature to predict and edge feature between any two connected nodes for weighted GraphSage convolution.(string)

```
Graph_structure_node_features": ["new_cases"] , [‘new_deaths’]
Graph_structure_mobility_edge_features: ["distance"]
```

5. Portion of the total raw data to extract after processing the dataset,and portion distribution into train , test and validation.(float,int)

```
"sample": .48,
"train": .80,
"val":.20,
"test" : 20
```

6.Learning rate of the model while gradient decent .(float)

```
"learning_rate": 0.05,
```

7. Optimizer of the cost function while training(optimizer object of Pytorch module)

```
"optimizer": torch.optim.Adam,
```

8. Number of folds to use in K-fold cross validation of training data.(int)

```
"cross_val_k": 5,
```

9. Wheather to use skip connection(to concatenate outputs resulted by skipping the RNN component) (bool) :

```
"skip_connection": True
```

Model Performance , and Result Visualization for new cases :-

We have monitored and plotted the Mean Absolute Error curve of our Model’s forecasting performance .The Country wise Mean Absolute Error on training evaluation is shown below,In these below picture each separate colour represents individual countries. x- axis is Mean Absolute Error(ie difference between actual and predicted node feature value(=new cases)) of any particular day(x-axis)The Country wise Mean Absolute Error on validation data is shown below, (x- axis is Mean Absolute Error)

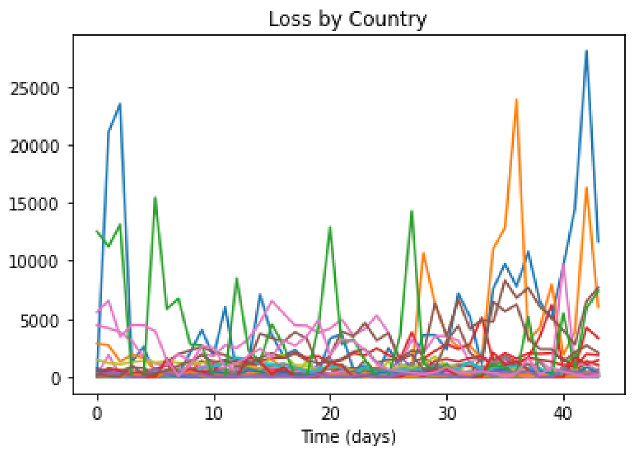


Figure 13. MAE loss by countries on new case prediction on validation data

The Country wise Mean Absolute Error on test data is shown below,(x- axis is Mean Absolute Error)

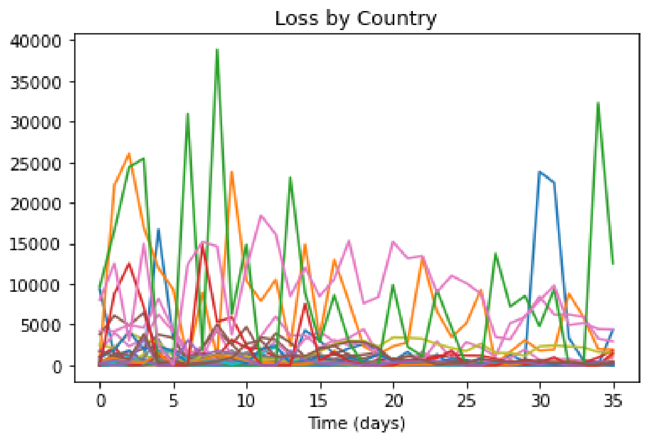


Figure 14. MAE loss by countries on new case prediction on test data

From these Plots we can realise that leaving only few countries rest of the countries all have Mean Absolute Error in around 5000 of total new cases prediction on validation and test data.Now we plot the total cases vs total prediction on the whole geographical region ie talking all subregions(countries)

On validation data:-

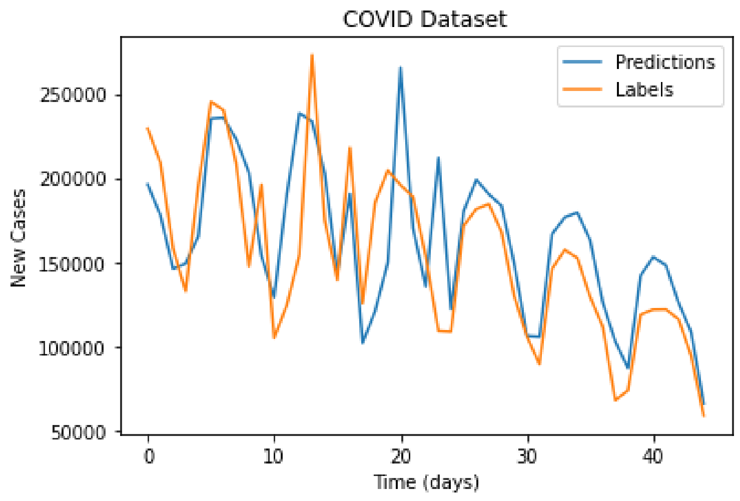


Figure 15:- Prediction vs actual plot of new cases on valdation data on new casess

On Test data:-

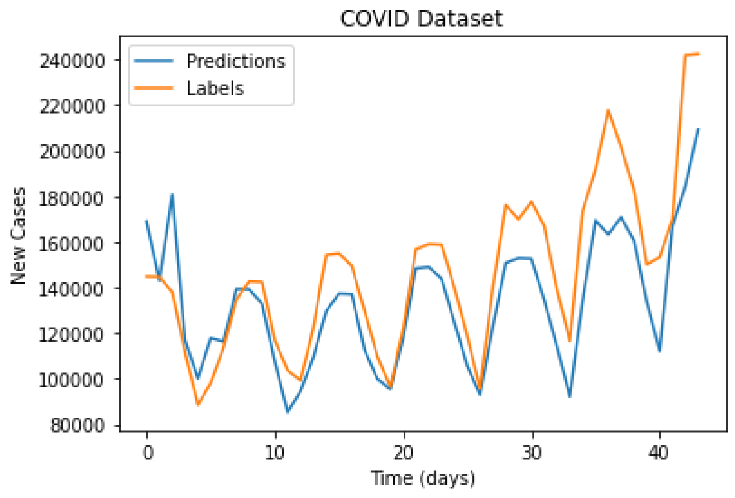


Figure 16:- Prediction vs actual plot of new cases on test data on new cases

We have used six widely used metrics for epidemiological time series regression to evaluate our models performance.They are Reporting Metric(vide this title to see result of this metric in json file):-

1.Mean Absolute Scalled Error(per Individual):-

MASE is a specific types of Mean Absolute Scaled error to monitor and report loss.In error monitoring , ie when training the model, we have used the per-individual error,we get this value by taking the mean of the difference vector between output and label ,and scaled by mean of cases over all regions(label_mean).where N stands for the number of   geographical areas (regions) in consideration.And this We can formulate as ,

MASE1 = mean(abs(predicted - actual)) / label_mean

We have used this metric as loss function while training our method . The reading of this error metric at last epoch on different datasets is shown below:-

```
"Train": 0.2651436924934387,
"Train Evaluation": 0.2845142674446106,
"Validation": 0.2499507796764374,
"Test": 0.21184120893478394
```

Loss Function (vide this title to see result of this metric in json file) :-

2. Mean Absolute Scaled Error(Per country):-In this Measurment we aggregate the actual and predicted cases separately and take the absolute difference between them .Then we scale the absolute difference vector by aggregated actual cases .

otptsum = torch.sum(predicted)

labelsum = torch.sum(actual)

abs(otptsum - labelsum) / labelsum

The reading of this error metric at last epoch on different datasets is shown below:-

```
"Train": 0.23676176130771637,
"Train Evaluation": 0.2064223277568817,
"Validation": 0.17772770762443542,
"Test": 0.24208979576826096
```

The MASE(Per-Country) performance of the model in validation data through epochs are shown below,

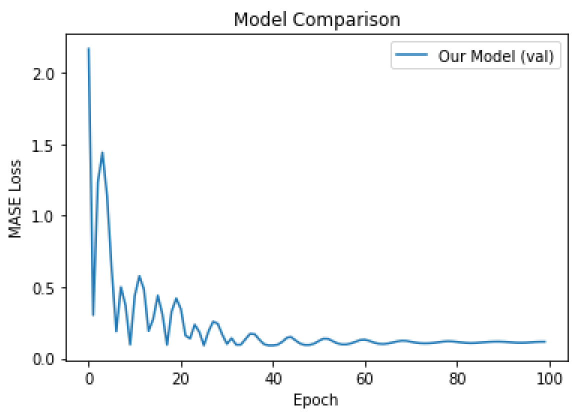


Figure 17. MASE(per country) Loss by epoch on validation data

LossFunction(vide this title to see result of this metric in json file):-

3.Mean Absolute Error :- This is the average absolute difference between Actual New cases and predicted new cases for a particular day across regions.

Mean of (|prediction vector – actual cases vector|)

The reading of this error metric at last epoch on different datasets is shown below:-

```
"Train": 963.46240234375,
"Train Evaluation": 907.4509887695312,
"Validation": 1147.6856079101562,
"Test": 1262.640380859375
```

Reporting Metric(vide this title to see result of this metric in json file):-

4.Root Mean square Error : First we take the square of the distance vector Between prediction and actual cases ,Then take its component mean and Finally take square root of it.

Squareroot(Mean(Square(prediction vector – actual cases vector)))

The reading of this error metric at last epoch on different datasets is shown below:-

```
"Train": 1612.1820068359375,
"Train Evaluation": 1613.2989501953125,
"Validation": 1922.38232421875,
"Test": 3659.93359375
```

Loss Function (vide this title to see result of this metric in json file) :-

5.Mean Absolute Percentage Error :- This is a time series regression Metric Based on summation of Scaled Absolute errors.Suppose at any particular time the actual and forecasted vector are Act and Fore. Then the MAPE error of that time is

MAPE = mean($\Sigma^{n}_{i=1}$ |(Act_i -$Fore_i$)/Act_i|) *100

Where Act_i and $Fore_i$ are components of Actual and Forecasted vector

The MAPE performance of the model in validation data through epochs are shown below,

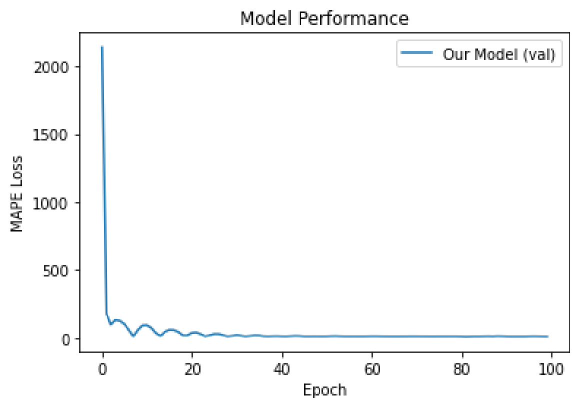


Figure 18 .MAPE loss by epoch on validation data

The reading of this error metric at last epoch on different datasets is shown below:-

```
"Train": 17.81756019592285,
"Train Evaluation": 17.865793228149414,
"Validation": 10.01746940612793,
"Test": 11.368473052978516
```

Reporting Metric(vide this title to see result of this metric in json file):-

6. R^2 Score :- We use R^2 score to determine how much portion of depemdemt variable variance is forecastble from the independent variables .Suppose Act is our actual vector and Pred is our predicted vector and act_mean is the mean value of actual vector components .

Then sum square of residuals $= 1 - (\Sigma^n_{i=1} (Act_i - Pred_i)^2 / \Sigma^n_{i=1} (Act_i - act\_mean_i)^2)$

Where Act_i , $pred_i$, act_mean_i are respectively components of those vectors.The R2 performance of the model in validation data through epochs are shown below,

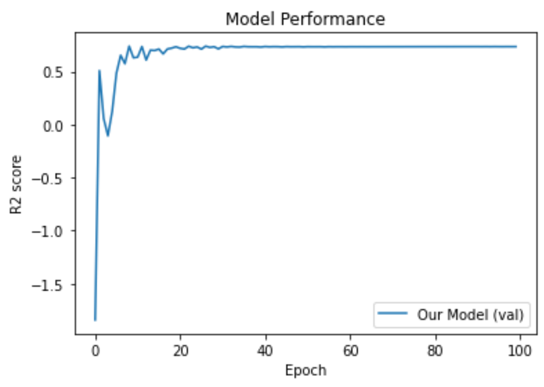


Figure 19. R^2 score per epoch on validation data.

The reading of this error metric at last epoch on different datasets is shown below:-

```
"Train": 0.7386975288391113,
"Train Evaluation": 0.7578183603286743,
"Validation": 0.7863057112693787,
"Test": 0.7204577922821045
```

Model Performance , and Result Visualization for new deaths :-

In these below picture each separate colour represents individual countries. x- axis is Mean Absolute Error(ie difference between actual and predicted node feature value(=new cases)) of any particular day(x-axis)

The Country wise Mean Absolute Error on validation data is shown below, (x- axis is Mean Absolute Error)

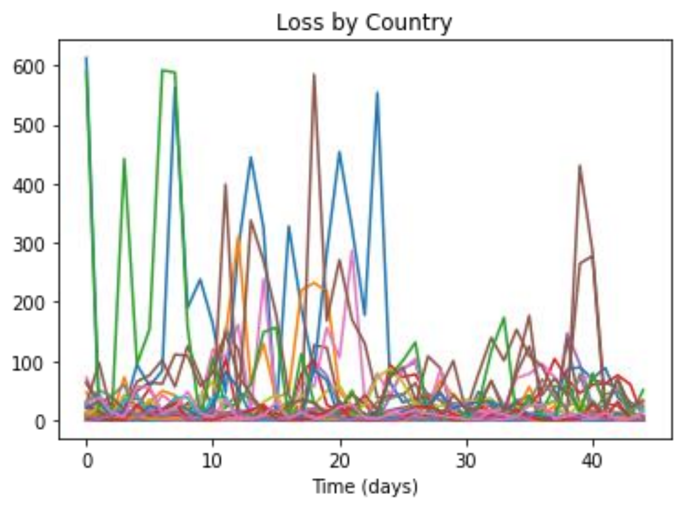


Figure 20 . loss by country on new deaths prediction on validation data

The Country wise Mean Absolute Error on Test data is shown below, (x- axis is Mean Absolute Error)

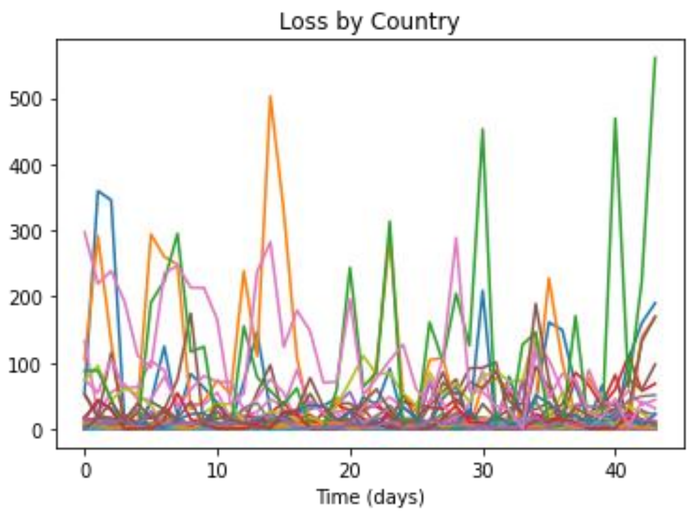


Figure 21 . loss by country on new deaths prediction on test data

From these Plots we can realise that leaving only few countries rest of the countries all have Mean Absolute Error in around 50 of total new deathes prediction on validation and test data. Now we plot the total deathes vs total prediction on the whole geographical region ie talking all subregions(countries)

On validation data:-

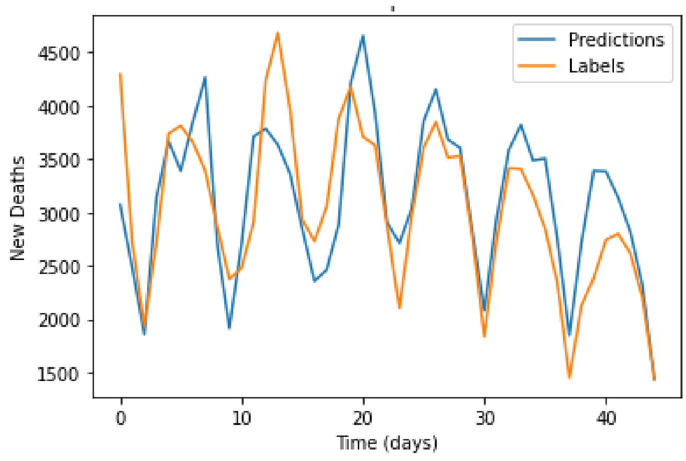


Figure 22. predicted vs actual new deaths on validation data

On validation data:-

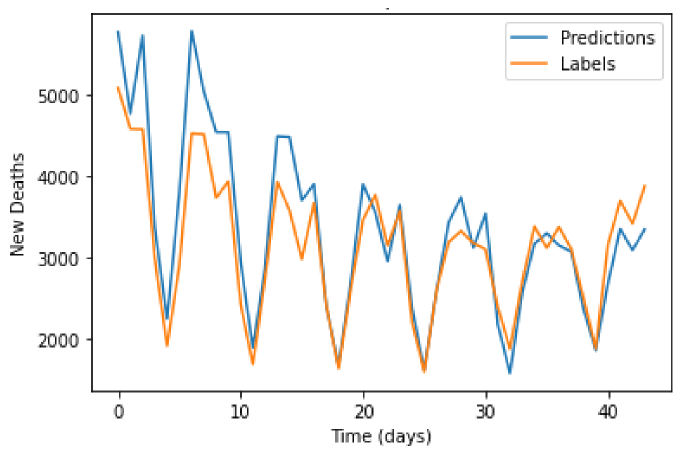


Figure 23. predicted vs actual new deaths on test data

Mean Absolute Scalled Error(per Individual):-

```
"Train": 0.29496802031993866,
"Train Evaluation": 0.2549143761396408,
"Validation": 0.34720532596111298,
"Test": 0.31055675685405731
```

Mean Absolute Scalled Error(per country):-

```
"Train": 0.2056809961795807,
"Train Evaluation": 0.20625975728034973,
"Validation": 0.10702469199895859,
"Test": 0.1330944448709488
```

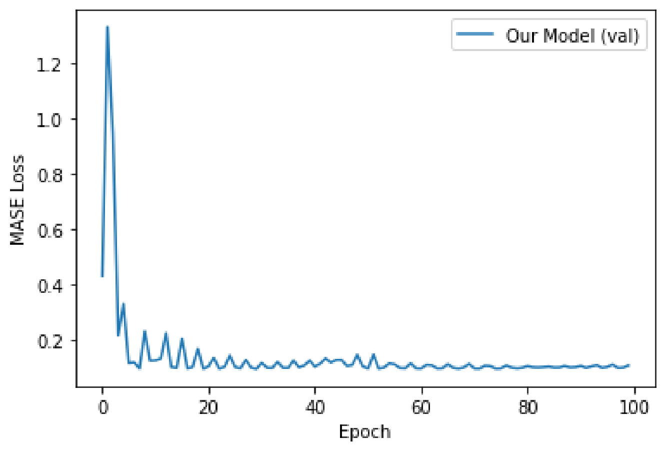


Figure 24. MASE Loss by epoch on validation data on new deaths

Mean Absolute Error :-

```
"Train": 18.317209243774414,
"Train Evaluation": 16.315427780151367,
"Validation": 22.783531188964844,
"Test": 21.691511154174805
```

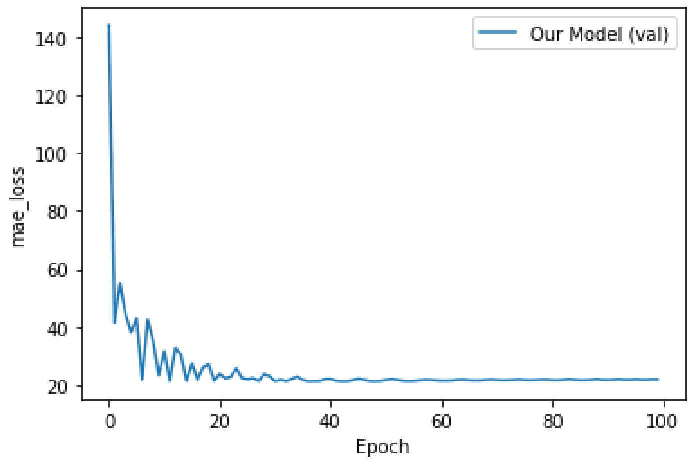


Figure 25. MAE loss by epoch on validation data on new deaths

Root Mean square Error :-

```
"Train": 46.32280731201172,
"Train Evaluation": 46.289554595947266,
"Validation": 70.5757942199707,
"Test": 65.34444808959961
```

R^2 score :-

```
"Train": 0.7953214049339294,
"Train Evaluation": 0.8151505780220032,
"Validation": 0.7984634280204773,
"Test": 0.8242509484291077
```

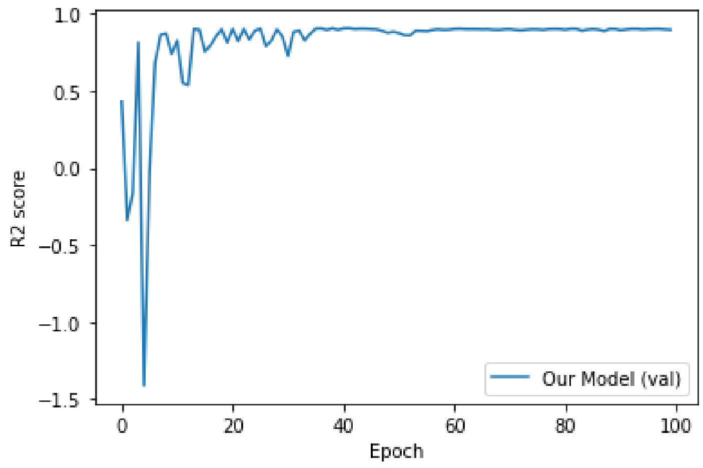


Figure 26.R^2 loss per epoch on validation data on new deaths

Mean Absolute Percentage Error:-

```
"Train": 23.483774185180664,
"Train Evaluation": 24.49663734436035,
"Validation": 29.710880279541016,
"Test": 27.037327766418457
```

We have observed and tabulerized the performance of our model through the 100 Epochs of iteration under aforementioned Reporting Metrics by Epoch and Loss by Epoch as -> 1.while training:Train 2.after training on train dataset :Train Evaluation 3.Validation dataset : Validation 4.Test dataset:Test and Saved it in a json file.We have also saved the Best Epoch amoungst them.Below is a tabulerized view of the results on different datasets.

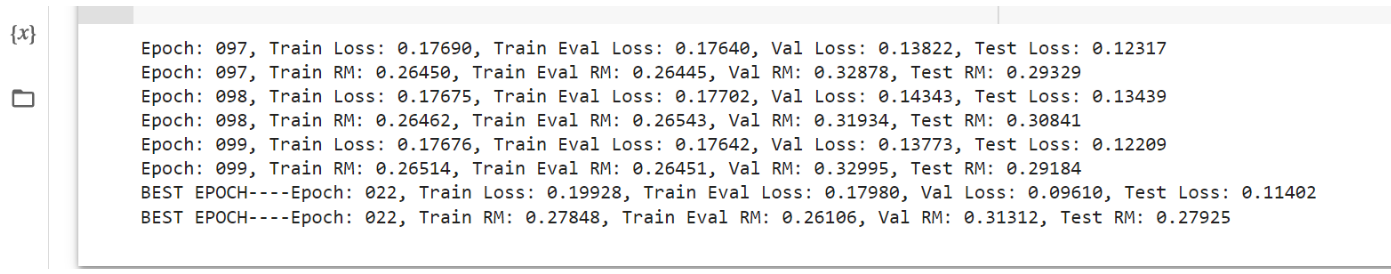
	MASE **Per Individual**	**MASE** **Per Country**	**MAPE**	**MAE**	**RMSE**	**R^2**

	New Case ***New Death***	***New Case*** ***New Death***	***New Case*** ***New Death***	***New Caes*** ***New Death***	***New Caes*** ***New Death***	***New Caes*** ***New Death***

train	0.26	0.29	0.23	0.20	17	23	963	18	1612	46	0.73	79
Train eva	0.28	0.25	0.20	0.20	17	24	907	16	1613	46	0.75	81
val	0.24	0.34	0.17	0.10	10	29	1147	22	1922	70	0.78	79
Test	0.21	0.31	0.24	0.13	11	27	1262	21	3659	65	0.72	82

Table 1. Tabularized view of new case and new death prediction readings on different error metrics at last (100th epoch) across different datasets.

The Run Log of our model , out of 100 epochs the last three epochs and best epoch is shown below:



```
Epoch: 097, Train Loss: 0.17690, Train Eval Loss: 0.17640, Val Loss: 0.13822, Test Loss: 0.12317
Epoch: 097, Train RM: 0.26450, Train Eval RM: 0.26445, Val RM: 0.32878, Test RM: 0.29329
Epoch: 098, Train Loss: 0.17675, Train Eval Loss: 0.17702, Val Loss: 0.14343, Test Loss: 0.13439
Epoch: 098, Train RM: 0.26462, Train Eval RM: 0.26543, Val RM: 0.31934, Test RM: 0.30841
Epoch: 099, Train Loss: 0.17676, Train Eval Loss: 0.17642, Val Loss: 0.13773, Test Loss: 0.12209
Epoch: 099, Train RM: 0.26514, Train Eval RM: 0.26451, Val RM: 0.32995, Test RM: 0.29184
BEST EPOCH----Epoch: 022, Train Loss: 0.19928, Train Eval Loss: 0.17980, Val Loss: 0.09610, Test Loss: 0.11402
BEST EPOCH----Epoch: 022, Train RM: 0.27848, Train Eval RM: 0.26106, Val RM: 0.31312, Test RM: 0.27925
```

Figure 27. Console output of model epochs and best epoch

Experiment Setting : - Anaoonda jupyter Notebook,Version 3.5.5. Proessor Intel i5 5th generation.

The overview of the json file(results file) structure is ,

1.Loss report(using loss function) for each dataset(train,validation ,set) per epoch.

2. reporting metric (using reporting metric) for each dataset(train,validation ,set) per epoch.

3.Loss report by each separate geographical regions(country) per epoch.

4.Predictions in test data per epoch

5.Actual in test data Per Epoch.

Conclusion And Future Work :-

We can robustly use the above mentioned metrics In effective policy making .The per-Individual MASE is to be used in nation -wide(international) policy making , The per-region MASE is used in making national polices.In our paper we represented a model by using Graph convolution and Recurrent Neural networks (LSTMs) with modification upon exsisting approaches of COVID-19 prediction by graphsage layer , skip connection , residual connection and droupout layerto reduce overfitting and increasing roboustness.The output of our model gives an improved knowledge about future pandemic evolution which can prove usefull for policy makers in case of taking preemptive decession .In our model We modify the LSTM algorithm by embedding Graph convolution operator from Spectral Graph Convolution within the LSTM gates which replaces the linear tranformations of LSTM by Graph convolution.This LSTM module can capture spatio-temporal patterns jointly in data.We also added the notion of skip connection .We performed an ablation test To validate the efficiency of the skip connection we proposed .We found that the model without skip connection (altered) shows an Less accurate result of .51 per -individual MASE and .58 per region MASE.This skip connection helps the model significantly to learn spatio -temporal patterens in the data.Thus our work can produce results and sollutions that are usefull beyond the epidemiological prediction application. Many problems thus depends on a generic graph structure.Like our case ,the node features of which will change according to the time .Our Future Work will improve the model by broadening the domain of input data by using more node feature as well as edge feature,and considering more other relevant parametres like rescued ,hospital capacity,death ,hospitalized, mobility age demographics etc.This we plan to include in our future work.

References:-

1. Alex Graves. Generating sequences with recurrent neural networks. arXiv:1308.0850, 2013
2. Sepp Hochreiter and Jurgen Schmidhuber. Long short-term memory. ¨ Neural computation, 1997.
3. Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. Unsupervised learning of video representations using lstms. In International Conference on Machine Learning (ICML), 2015.
4. Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In Advances in neural information processing systems (NIPS), 2014.
5. Felix A Gers and Jurgen Schmidhuber. Recurrent nets that time and count. In ¨ IEEE-INNS-ENNS International Joint Conference on Neural Networks, 2000
6. Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning Convolutional Neural Networks for Graphs. In International Conference on Machine Learning (ICML), 2016.
7. Jonathan Masci, Davide Boscaini, Michael M. Bronstein, and Pierre Vandergheynst. Geodesic convolutional neural networks on riemannian manifolds. In IEEE International Conference on Computer Vision (ICCV) Workshops, 2015.
8. Michael Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with ¨ fast localized spectral filtering. In Advances in Neural Information Processing Systems (NIPS), 2016.
9. Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral Networks and Locally Connected Networks on Graphs. In International Conference on Learning Representations (ICML), 2014.
10. F. R. K. Chung. Spectral Graph Theory. American Mathematical Society, 1997.
11. D. Shuman, S. Narang, P. Frossard, A. Ortega, and P. Vandergheynst. The Emerging Field of Signal Processing on Graphs: Extending High-Dimensional Data Analysis to Networks and other Irregular Domains. IEEE Signal Processing Magazine, 2013.
12. Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun Woo. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. In Advances in Neural Information Processing Systems (NIPS), 2015.
13. J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. Spectral Networks and Deep Locally Connected Networks on Graphs. arXiv:1312.6203, 2013.
14. D. Shuman, S. Narang, P. Frossard, A. Ortega, and P. Vandergheynst. The Emerging Field of Signal Processing on Graphs: Extending High-Dimensional Data Analysis to Networks and other Irregular Domains. IEEE Signal Processing Magazine, 30(3):83–98, 2013.
15. S. Mallat. A Wavelet Tour of Signal Processing. Academic press, 1999.
16. F. R. K. Chung. Spectral Graph Theory, volume 92. American Mathematical Society, 1997.
17. D. Hammond, P. Vandergheynst, and R. Gribonval. Wavelets on Graphs via Spectral Graph Theory. Applied and Computational Harmonic Analysis, 30(2):129–150, 2011.
18. T.N. Bui and C. Jones. Finding Good Approximate Vertex and Edge Partitions is NP-hard. Information Processing Letters, 42(3):153–159, 1992.
19. I. Dhillon, Y. Guan, and B. Kulis. Weighted Graph Cuts Without Eigenvectors: A Multilevel Approach. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 29(11):1944–1957, 2007.
20. G. Karypis and V. Kumar. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. SIAM Journal on Scientific Computing (SISC), 20(1):359–392, 1998.
21. J. Shi and J. Malik. Normalized Cuts and Image Segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 22(8):888–905, 2000.

22. Fauci, A. S., Lane, H. C., and Redfield, R. R. Covid-19 — navigating the uncharted. New England Journal of Medicine, 382(13):1268–1269, 2020
23. Velavan, T. P. and Meyer, C. G. The covid-19 epidemic. Tropical medicine & international health, 25(3):278, 2020
24. https://en.wikipedia.org/wiki/Template:COVID-19_pandemic_data
25. https://www.statista.com/topics/6139/covid-19-impact-on-the-global-economy/#dossierKeyfigures
26. Li, Q., Guan, X., Wu, P., Wang, X., Zhou, L., Tong, Y., Ren, R., Leung, K. S., Lau, E. H., Wong, J. Y., Xing, X., Xiang, N., Wu, Y., Li, C., Chen, Q., Li, D., Liu, T., Zhao, J., Liu, M., Tu, W., Chen, C., Jin, L., Yang, R., Wang, Q., Zhou, S., Wang, R., Liu, H., Luo, Y., Liu, Y., Shao, G., Li, H., Tao, Z., Yang, Y., Deng, Z., Liu, B., Ma, Z., Zhang, Y., Shi, G., Lam, T. T., Wu, J. T., Gao, G. F., Cowling, B. J., Yang, B., Leung, G. M., and Feng, Z. Early transmission dynamics in wuhan, china, of novel coronavirus–infected pneumonia. New England Journal of Medicine, 382(13): 1199–1207, 2020.
27. Hochreiter, S. and Schmidhuber, J. Long short-term memory. Neural computation, 9(8):1735–1780, 1997.
28. Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. The graph neural network model. IEEE transactions on neural networks, 20(1):61–80, 2008.
29. La Gatta, V., Moscato, V., Postiglione, M., and Sperl´ı, G. An epidemiological neural network exploiting dynamic graph structured data applied to the covid-19 outbreak. IEEE Transactions on Big Data, 7(1):45–55, 2021.
30. Fritz, C., Dorigatti, E., and Rugamer, D. Combining ¨ graph neural networks and spatio-temporal disease models to predict covid-19 cases in germany. arXiv preprint arXiv:2101.00661, 2021.
31. Cao, D., Wang, Y., Duan, J., Zhang, C., Zhu, X., Huang, C., Tong, Y., Xu, B., Bai, J., Tong, J., et al. Spectral temporal graph neural network for multivariate time-series forecasting. arXiv preprint arXiv:2103.07719, 2021
32. Hamilton, W. L., Ying, R., and Leskovec, J. Inductive representation learning on large graphs. arXiv preprint arXiv:1706.02216, 2017
33. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1):1929–1958, 2014.
34. He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778, 2016.
35. Rozemberczki, B., Scherer, P., He, Y., Panagopoulos, G., Riedel, A., Astefanoaei, M., Kiss, O., Beres, F., Lopez, G., Collignon, N., and Sarkar, R. PyTorch Geometric Temporal: Spatiotemporal Signal Processing with Neural Machine Learning Models, 2021.
36. Dye, C., Cheng, R. C., Dagpunar, J. S., and Williams, B. G. The scale and dynamics of covid-19 epidemics across europe. Royal Society open science, 7(11):201726, 2020 (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7735356/)
37. Saglietto, A., D'Ascenzo, F., Zoccai, G. B., and De Ferrari, G. M. Covid-19 in europe: the italian lesson. Lancet, 395 (10230):1110–1111, 2020
38. Seo, Y., Defferrard, M., Vandergheynst, P., and Bresson, X. Structured sequence modeling with graph convolutional recurrent networks. In International Conference on Neural Information Processing, pp. 362–373. Springer, 2018.
39. Li, Y., Yu, R., Shahabi, C., and Liu, Y. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. arXiv preprint arXiv:1707.01926, 2017.#

40. Alimadadi, A., Aryal, S., Manandhar, I., Munroe, P. B., Joe, B., and Cheng, X. Artificial intelligence and machine learning to fight covid-19. Physiological Genomics, 52 (4):200–202, 2020.
41. Buckee, C. O., Balsari, S., Chan, J., Crosas, M., Dominici, F., Gasser, U., Grad, Y. H., Grenfell, B., Halloran, M. E., Kraemer, M. U. G., Lipsitch, M., Metcalf, C. J. E., Meyers, L. A., Perkins, T. A., Santillana, M., Scarpino, S. V., Viboud, C., Wesolowski, A., and Schroeder, A. Aggregated mobility data could help fight covid-19. Science, 368(6487):145–146, 2020. ISSN 0036-8075.
42. Barstugan, M., Ozkaya, U., and Ozturk, S. Coronavirus (COVID-19) Classification using CT Images by Machine Learning Methods. arXiv preprint arXiv:2003.09424, mar 2020
43. Wang, S.-H., Govindaraj, V. V., Gorriz, J. M., Zhang, X., ´ and Zhang, Y.-D. Covid-19 classification by fgcnet with deep feature fusion from graph convolutional network and convolutional neural network. Information Fusion, 67:208–229, 2021. ISSN 1566-2535.
44. Alazab, M., Awajan, A., Mesleh, A., Abraham, A., Jatana, V., and Alhyari, S. Covid-19 prediction and detection using deep learning. International Journal of Computer Information Systems and Industrial Management Applications, 12:168–181, 2020.
45. Chimmula, V. K. R. and Zhang, L. Time series forecasting of covid-19 transmission in canada using lstm networks. Chaos, Solitons & Fractals, 135:109864, 2020. ISSN 0960-0779.
46. Shahid, F., Zameer, A., and Muneeb, M. Predictions for covid-19 with deep learning models of lstm, gru and bilstm. Chaos, Solitons & Fractals, 140:110212, 2020. ISSN 0960-0779
47. Arora, P., Kumar, H., and Panigrahi, B. K. Prediction and analysis of covid-19 positive cases using deep learning models: A descriptive case study of india. Chaos, Solitons & Fractals, 139:110017, 2020. ISSN 0960-0779.
48. Bailey, N. T. et al. The mathematical theory of infectious diseases and its applications. Charles Griffin & Company Ltd, 5a Crendon Street, High Wycombe, Bucks HP13 6LE., 1975.
49. Michaël Defferrard, Xavier Bresson ,Pierre Vandergheynst. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering
50. Aggregated mobility data could help fight covid-19Buckee, C. O., Balsari, S., Chan, J., Crosas, M., Dominici, F., Gasser, U., Grad, Y. H., Grenfell, B., Halloran, M. E., Kraemer, M. U. G., Lipsitch, M., Metcalf, C. J. E., Meyers, L. A., Perkins, T. A., Santillana, M., Scarpino, S. V., Viboud, C., Wesolowski, A., and Schroeder, A. Aggregated mobility data could help fight covid-19. Science, 368(6487):145–146, 2020. ISSN 0036-8075.
51. Youngjoo Seo,Michel Defferrad,Pierre vanderghenyst,Xavier bersson.Structured sequence modelling with Graph convolutional Recurrent Networks
52. Residual Connection Behave Like Ensembales of Relatively Shallow Networks(2016),NIPS 2016 Fritz, A.Veit,M.Wilber, and S Belongie
53. Multilayer Perceptron and Neural Networks, MARIUS-CONSTANTIN POPESCU VALENTINA E. BALAS, LILIANA PERESCU-POPESCU , NIKOS MASTORAKIS, WSEAS Transactions on Circuits and Systems · July 2009
54. https://medium.com/analytics-vidhya/ohmygraphs-graphsage-and-inductive-representation-learning-ea26d2835331
55. https://towardsdatascience.com/multi-layer-neural-networks-with-sigmoid-function-deep-learning-for-rookies-2-bf464f09eb7f
56. https://towardsdatascience.com/multi-layer-neural-networks-with-sigmoid-function-deep-learning-for-rookies-2-bf464f09eb7f
57. https://towardsdatascience.com/residual-blocks-building-blocks-of-resnet-fd90ca15d6ec
58. https://coronavirus.jhu.edu/about/how-to-use-our-data
59. https://developers.google.com/public-data/docs/canonical/countries_csv