# Recurrent model for Graph Network: an application to the World Trade Network dataset

Lorenzo Jacopo Pileggi

September 2022

## 1 Introduction

In this project we extended the functionalities of the graph_nets library to cope with temporal graph data, and applied such framework to the World Trade Network dataset to make predictions on countries' GDP trends, comparing them to the ones of a simple RNN model on GDP data only.

## 2 Data preprocessing

The data relative to the World trade Network come from the UN comtrade database: https://comtrade.un.org/, whereas yearly GDP figures are taken from https://public.knoema.com/mhrzolg/historical-gdp-by-country-statistics-from-the-world-bank-1960-2019; both of them are measured in USD.

From the former, for each country we downloaded the csv files relative to their commercial exchanges for a given year, covering the period 1996-2018. From such files we extracted the quantities of interest, that is the yearly aggregated volumes of exchanges between couples of countries, and then built a weighted adjacency matrix by averaging the figures about the trades (as each country in general reports different figures regarding import/export from each of their counterparts) and discarding values below 1 million USD. Finally, to ignore effects linked to inflation and growth of total global trade, we normalised both the trade and GDP data to make their sum equal to 1 for each year considered.

The final adjacency matrix has been evaluated both through CPU and GPU computing to test the differences in performances: the former yielded the matrix in 22 s, the latter in 2.1 s.

## 3 Description of the framework

Our model is based on the GraphNetwork module of graph_nets, built in such a way to allow message-passing between the different blocks of the model (respectively, EdgeBlock, NodeBlock and GlobalBlock) through a reducer function, a

sort of pooling operation that takes the information coming from a different block (say the in- and out-going edges of a node) and reduces it to add some additional features to be used by the block in question. The edge and node blocks are provided with a recurrent model, made of a first MLP layer and a recurrent layer on top of it, using a GRU.

The nodes/edges tensors have to be of shape (n_nodes/edges, n_time_steps, n_features), with n_features given by the combination of features relative to the corresponding layer, plus the additional ones yielded by the pooling operation. For the edge block, the receiver and sender nodes' values are used as additional features, so that n_features = 3 for edges. For the node block, the reducer function consists in taking the list of in- or out-going edges of a node and, given a number $n$, taking the $n$ sums of all the elements of such list whose index $i$ satisfies $i \mod n = k, k \in \{0, n-1\}$; since this is done for both the in- and out-going edges, we end up with $2n + 1$ final features. The output of the last layer has then to be normalised to make the year-by-year sums equal 1, as our original data.

The competing model is analogous to the GraphIndependent module found in the graph_nets modules: it consists of a GRU applied to the node and edge feature lists, without any pooling.

# 4 Results and analyses

The simple GRU has been trained for 100 epochs, both for nodes and edges, whilst the Recurrent Graph Network, due to the high computational cost (around 6 minutes per epoch), only for 10 epochs; the rest of the parameters are found in the config files in the ./config folder in the GitHub repository. In particular for the latter, what we called the pool dimension, i.e. the number of "reduced features" added by the reducer function, has been set to 3, so that, for what we already said, n_features = 7 for nodes.

The training and validation set was fixed to the data from 1996 to 2011, and the test set to the remaining years. For the optimiser Adam has been chosen, and the accuracy was taken as the mean squared error of predicted vs real streaks.

The final loss on the test set yielded by the RecGraphNetwork model resulted in $2.8824 \cdot 10^{-6}$, outperforming the simple GRU, with a loss of $3.3797 \cdot 10^{-6}$, testifying the validity of the novel approach. In Figure (1) are showed the plots of real data vs prediction for both the GRU and RecGraphNetwork

# 5 Conclusions

We tested the newly developed Recurrent Graph Network framework on the World Trade Network and yearly GDP data, comparing the results with a simple recurrent model without message-passing. Though comparable in accuracy, the latter had nonetheless a clear edge on streaks of values below $10^{-3}$, calling for

further improvement of the former.

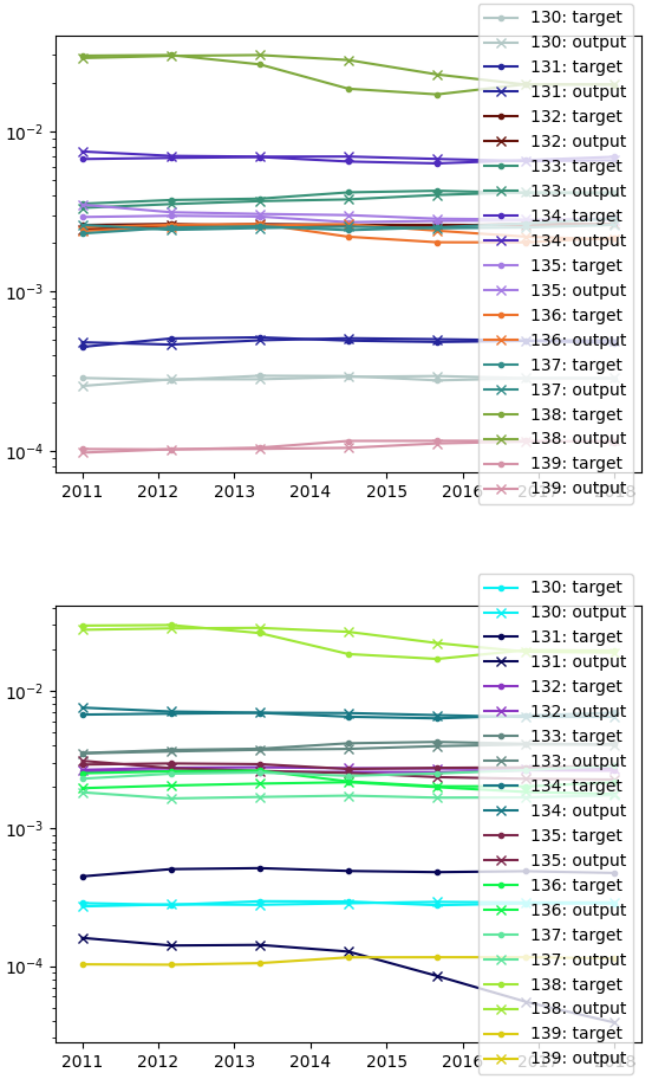


Figure 1: Plots of real (dotted points) vs predicted series (x crossed points) of simple GRU (upper plot) and RecGraphNetwork (lower plot), corresponding to the following countries (from 130 to 139): Papua New Guinea, Paraguay, Peru, Philippines, Poland, Portugal, Qatar, Romania, Russia, Rwanda. The discrepancy between the two model emerges below $10^{-3}$.