Bilateral Trade Modelling with Graph Neural Networks

Kobby Panford-Quainoo*
African Masters in Machine Intelligence
African Institute for Mathematical Sciences
Kigali, Rwanda
kpanford-quainoo@aimsammi.org

Abstract

The traditional way of predicting potential trade partners is by using the gravity methods to estimate a trade flow whose value shows how likely any two countries under consideration would trade. This gravity method is, however, a difficult task due to the exponentially growing number of constants that needs to be considered to obtain a better estimate of trade flow. In this study, we present a framework for directly predicting bilateral trade partners from observed trade records. We also classify countries into their various income levels as another downstream task. Learning in each task is done using graph neural networks. We observe high accuracy up to 98% on link prediction and 68% on income level classification graph neural network models.

1 Introduction

International trade involves the exchange of goods, capital and services between countries[1] and where two countries are involved, it is referred to as *Bilateral trade*. Various models have been employed by economists to understand trade patterns and factors that account for the observed trade activities between countries. The Ricardian model, introducing comparative advantage of countries, was one of the earlier attempts to describe the trading behaviour of some countries with others. According to David Ricardo, a country exports more of the goods they can produce at a lower cost and higher quality[2]. The Heckscher-Ohlin theorem also introduces the concept of "factor of abundance" which argues that the trading behaviour of a country is influenced by what they confidently produce in abundance[3]. In this line of interest, the gravity model provides an empirical description of how the gross domestic product (GDP) of any two countries and the geographical distance from their borders influence the extent of trading between them. The models show that countries with high GDPs will have a high trade flow and will more likely than countries with low GDPs[4]. Again, trade flow is smaller if they are distant apart and will less likely trade. Trade flow, therefore, serves as the basis for predicting potential trade partners for countries. This is an important task in economics because it allows policymakers to relax restrictions and trade barriers to foster the partnership between the countries and consequently expand their economic capacities.
One difficulty with using the gravity model is that a lot of other dummy variables like cultural differences, political terms and others must be handcrafted and factored into the equation. This makes the cost of capturing the right information that actually affects trade patterns very expensive when using this model.

Our work is motivated by the difficulty in estimating the trade flow and we intend to tackle trade partner prediction and income level classification from the graph perspective. We capitalize on the

*Code can be found at `https://github.com/panford/BiTrade-Graphs`

Submitted to the African Institute for Mathematical Sciences for a Master's degree in Machine Intelligence (AIMS 2019), Rwanda.

successes of graph neural networks to reason over graph-structured in applications such as social recommendation in the social networks [5] and recommendations in web recommender systems [6]. One important characteristic of graph-structured data is that the interactions between individuals is used to build an adjacency matrix which is leveraged during the learning process.

Our main contributions are;

- To show that international trade data can naturally be modelled as a graph
- Directly predict trade links between countries without first having to estimate the trade flow values between them.
- To show that the trade relationship between countries could be a major ingredient when predicting their income levels.

2 Background

2.1 Country Classification

The World Bank defines four income groups in the world namely: high-, upper middle, lower middle and low-income. The division into these income groups is based on the total annual income called the gross national income (GNI) per capita. The GNI of a country gives and idea what its economic strength and weaknesses are and in general, the standard of living of the average citizen. Countries are classified into various income groups if their GNI falls within a certain threshold defined in the Table 1[7]. This classification by income levels can be used to measure progress over time or analyse data for countries falling into the same income groups.

income group	GNI threshold
lower	1,006 and below
lower middle	1,006 - 3,955
upper middle	3,956 - 12,235
high	12,235 and above

Table 1: Country GNI and income group according to World Bank, 2017

2.2 Bilateral Trade Flows

Inspired by Newton's law of universal gravitation, the gravity model provides a theoretical approach to representing the numerical trade strength between any two countries. The gravity model is used to compute a trade flow value that shows that the strength of trade flow between any two countries increases with increasing respective net income or GDP and decreasing with increasing distance [4, 8]. This formulation is expressed as

$$F_{ij} = M\frac{GDP_i \cdot GDP_j}{D_{ij}},$$

where F_{ij} is the trade flow between countries i an j, GDP_i is the GDP of country i, M is a proportionality constant and D_{ij} is the geographical distance between countries i and j. A more convenient way to deal with this equation is to express it in $\log$ and introduce coefficients and placeholder variables to account for other unanticipated factors which are not exactly deterministic. The gravity equation may then be expressed in the form

$$\ln F_{ij} = c_0 + c_1 \ln GDP_i + c_2 \ln GDP_j + c_3 \ln D_{ij} + c_4 d + c_5 P_{ij} + \epsilon_{ij},$$

The newly introduced terms c_k, P_{ij} and ϵ_{ij} are constants, political influence term and an error correction term respectively.

2.3 Graph Neural Networks

Given a graph $\mathcal{G} = (\mathcal{V}, X, \mathbf{A}, \mathcal{E})$, individual entities are referred to as nodes $\mathcal{V}$ with some characteristic node features $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times D}$, where $|\mathcal{V}|$ and D are the number of nodes and features respectively.

An edge $\mathbf{a}_{ij}$ is said to exist between nodes i and j if they are connected and vice versa. This can be composed into a dense square adjacency matrix $A \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ which may be symmetric or not depending on whether or not the graph is directed. Edges in a directed graph have arrows going from one node i to another node j to show that node i is connected to node j. The reverse is not true and $a_{ij} \neq a_{ji}$. On the other hand, edges in an undirected graph have no direction ie. $a_{ij} = a_{ji}$. Edge weights $\mathcal{E}$ is numerical indication of the strength of relationship between the connected nodes.

Graph Neural Networks (GNN) is a family of approaches that aim to generalize neural networks, developed for Euclidean data, to graphs[9]. To describe how the GNNs learn from graph-structured data, it is worth mentioning the message-passing strategy. Here, information about a node and its neighbourhood is captured and aggregated through an aggregation function **F** and learning the local neighbourhood structure of each node as a result. This information is passed periodically as a message to update a hidden state variable **H**. They can tackle tasks such as node[10][11], graph[12][13][14] and edge classification[15], link prediction[16] and node clustering[17] or community detection[18].

Node, graph and edge classification problems involve discriminating between classes of nodes, graphs and edges and providing labels to unlabeled ones at test time. Link prediction is predicting missing links between two nodes in the graph. Clustering is an unsupervised learning technique that leverages the similarities in features to put data points into inherent groups other than predefined target labels. Node clustering and community detection[2] therefore seek to detect groups of nodes referred to as *clusters* or *communities* in the absence of target labels.

Depending on the task at hand, several Graph Neural Network techniques have been proposed and many improvements have also been developed. These are based on specific applications and other properties of the graph such as being directed (ie. follower on Twitter) or heterogeneous (ie. paper - author nodes in citation network) and edges having weights or features (ie. net import and export trade value between two countries)[19].

2.3.1 ChebNet

Some of the early techniques proposed in graph representation learning sought to learn the local neighbourhood structure of nodes present in graphs by using filters that would share structural resemblances as well as the successes of those originally used on particularly images. Bruna et al. [20] introduced the spectral network, a convolutional network based on spectral filtering. Spectral Graph theory defines the convolution operation of graphs as;

$$g_\theta(L) * x = g_\theta(U\Lambda U^\top)x \tag{1}$$

Here, g is the spectral filter defined over x, U and Λ are the eigenvalues and eigenvectors of the Laplacian L respectively.
[21] uses a local approximation to avoid the expensive computation of the Laplacian eigenvectors (Λ) by defining filters as polynomials of the Laplacian (2).

$$g_\theta(\Lambda) * x = \sum_{k=0}^{K-1} \theta_k \Lambda^k x. \tag{2}$$

They compute an approximation of the Chebyshev polynomial $T_k(\tilde{L})$ in (4) from the truncated Laplacian $\tilde{L}$ shown in equation (3).

$$\tilde{L} = 2L/\lambda_{max} - I_n. \tag{3}$$

$$g_\theta(L) * x = \sum_{k=0}^{K-1} \theta_k T_k(\tilde{L})x. \tag{4}$$

Here, Λ^k is the truncated Laplacian eigenvectors computed from the Laplacian $L = U\Lambda^k U^\top$, θ_k is the Chebyshev coefficient evaluated at the k-th order.

This network is referred to as ChebNet.

[2]Node clustering and community detection are terms used by the machine learning on graphs and data mining communities respectively

2.3.2 Graph Convolutional Network (GCN)

GCN [10] is a variant of Graph Neural Network that extends the concept of Convolutional Neural Networks [22] originally applied successfully to images with regular Euclidean structure on the converse of non-Euclidean graph-structured data to learn an embedding by message passing across the neighbourhood of nodes. GCN approximates the ChebNet to the first-order by setting $K = 1$ and $\lambda_{max} = 2$.

The convolution filter on the graph reduces to

$$g_\theta * x = \theta \left(I_N + D^{-\frac{1}{2}} A D^{\frac{1}{2}} \right) x. \tag{5}$$

A is the adjacency matrix and D is the degree or the number of neighbours of a node. From the equation 5, A and D can immediately be re-normalized by adding the identity matrix I_N so that the features and states of the node itself are captured and this transforms into the aggregation function $F = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{\frac{1}{2}} X$ and node update $H = F\Theta$.

By stacking a sufficient number of GCN layers, hierarchical information can be extracted from the graph-structured data by propagating local messages across layers. This message propagation rule is given by

$$h^{l+1} = \sigma \left(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{\frac{1}{2}} h^{(l)} W^{(l)} \right), \tag{6}$$

where σ is the activation function, $\tilde{D}$ is the degree of node with self-loops, $\tilde{A}$ is the adjacency matrix with self-loops and W is the weight matrix.

2.3.3 Graph Attention Network (GAT)

Graph Attention Networks (GAT) [23] use self-attention to compute attention coefficients that assign weights by importance to nodes in each ones neighbourhood. It is an extension of attention, as proposed by Badanau et al. [24], that allows each neighbouring node to be attended to. The following equations summarize the attention mechanism used by GAT.

$$e_{ij} = a(\mathbf{W}\mathbf{h}_i, \mathbf{W}\mathbf{h}_j), \tag{7}$$

e_{ij} is the attention score between node i and node j computed from the shared weight W and the hidden states of i and j. This provides a new set of feature information about node i and its neighboring nodes to be learned together with a trainable parameter a which in effect aligns the attention scores e_{ij} with the output features. Using softmax, attention scores are normalised to the resulting equation in (9).

$$\alpha_{ij} = \mathrm{softmax}(e_{ij}), \tag{8}$$

$$\alpha_{ij} = \frac{\exp\left(\mathrm{LeakyReLU}\left(\mathbf{a}^\top [\mathbf{W}\mathbf{h}_i \,\|\, \mathbf{W}\mathbf{h}_j]\right)\right)}{\sum_{k \in \mathcal{N}} \exp\left(\mathrm{LeakyReLU}\left(\mathbf{a}^\top [\mathbf{W}\mathbf{h}_i \,\|\, \mathbf{W}\mathbf{h}_k]\right)\right)}. \tag{9}$$

[23] also defines the multi-head attention on graphs by concatenating a K number of independent attention mechanisms (11). Each of these attention mechanisms implements (10).

$$h^{t+1} = \sigma \left(\sum_{j \in \mathcal{N}} \alpha_{ij} \mathbf{W}\mathbf{h_j} \right) \tag{10}$$

$$h^{t+1} = \Big\|_{k=1}^{K} \sigma \left(\sum_{j \in \mathcal{N}} \alpha_{ij}^k \mathbf{W^k}\mathbf{h_j} \right) \tag{11}$$

In the case where the output of the last hidden layer of the network is used for prediction with a sigmoid or softmax activation, features from the K attention layers are averaged out as demonstrated by equation (12).

$$h^{t+1} = \sigma \left(\frac{1}{K} \sum_{k=1}^{K} \sum_{j \in \mathcal{N}} \alpha_{ij}^k \mathbf{W}^k \mathbf{h_j} \right) \tag{12}$$

2.3.4 Attention-Based Graph Neural Network (AGNN)

AGNNs follows closely the concept of GCN while showing that a simple linear network as the propagation layer of a GCN without the non-linear activation layer performs comparatively well. When coupled with an attentional layer, each neighbouring node is weighted according to its influence on the node under consideration.

2.3.5 Graph Auto-Encoder (GAE)

Previous models we have discussed are (semi-) supervised learning methods on graph data where each example has a label showing the class it belongs to. We go further to discuss the Graph Auto-Encoder (GAE) [25], an extension of auto-encoders [26, 27], for unsupervised learning on graphs. GAE primarily learns a latent representation for the data without looking at the labels and the edge directions. GAE is composed of an encoder (inference model) that learns a code which targets minimizing a reconstruction loss and a decoder (generative model) whose output is a reconstruction of the original representation from the code. Kipf and Welling [25] proposed a GCN encoder function

$$f(\mathbf{z}|\mathbf{X}, \mathbf{A}) = GCN(\mathbf{A}) \tag{13}$$

and a decoder function

$$p(\mathbf{A}|\mathbf{z}) = \prod_{i=1}^{N} \prod_{j=1}^{N} p(A_{ij}|\mathbf{z}_i, \mathbf{z}_j), \tag{14}$$

where $p(A_{ij}|\mathbf{z}_i, \mathbf{z}_j)$ can be

$$p(A_{ij}|\mathbf{z}_i, \mathbf{z}_j) = \sigma(\mathbf{z}_i^\top \mathbf{z}_j). \tag{15}$$

specialised in predicting new edges between nodes in the newly constructed adjacency matrix.

2.3.6 Variational Graph Auto-Encoder (VGAE)

The inference model of the Variational Graph Auto-Encoder (VGAE) learns a latent code which follows a controlled distribution with an encoder function $f(\mathbf{z}|\mathbf{X}, \mathbf{A})$. The encoder is generally made up of a simple double-layered GCN whose shared parameters are normally distributed.

$$q(\mathbf{z}|\mathbf{X}, \mathbf{A}) = \prod_{j=1}^{N} q(\mathbf{z}_i|\mathbf{X}, \mathbf{A}), \text{ where } q(\mathbf{z}_i|\mathbf{X}, \mathbf{A}) = \mathcal{N}(\mathbf{z_i}|\mu_i, \text{diag}(\sigma_i^2)). \tag{16}$$

The generative model then reconstructs a new adjacency matrix with a decoder function $p(\mathbf{A}|\mathbf{z})$ (same as shown for GAEs). Training is done by minimizing a variational lower bound:

$$\mathcal{L} = \mathbb{E}[\log p(\mathbf{A}|\mathbf{Z})] - KL[q(\mathbf{Z}|\mathbf{X}, \mathbf{A})||p(\mathbf{Z})], \tag{17}$$

3 Data and Tasks

Here, we show our proposed approach to representing countries in the graph-structured bilateral trade data between countries for income-level classification and trade partner prediction. We first describe the basic components of a typical graph and relate each to a component in our data.

Our primary goals are to (1) classify countries into their respective income levels; high, upper middle, lower middle and low and (2) predict potential trade partners. These are essentially multi-class node classification and link prediction tasks. We trained Graph Neural Network models with other baseline models to perform the aforementioned tasks. We based our evaluation of performance on classification accuracy on test examples and area under the curve (AUC) and average precision (AP) for node classification and link prediction task respectively. All experiments were performed on data specifically collected for this study and we do not include results on other standard benchmark datasets traditionally used to test the efficiency of novel approaches. In the next subsection, we talk about how we collected the data, the representation approach we took and the GNN models adopted for the downstream tasks.

Table 2: Summary of data features and representation

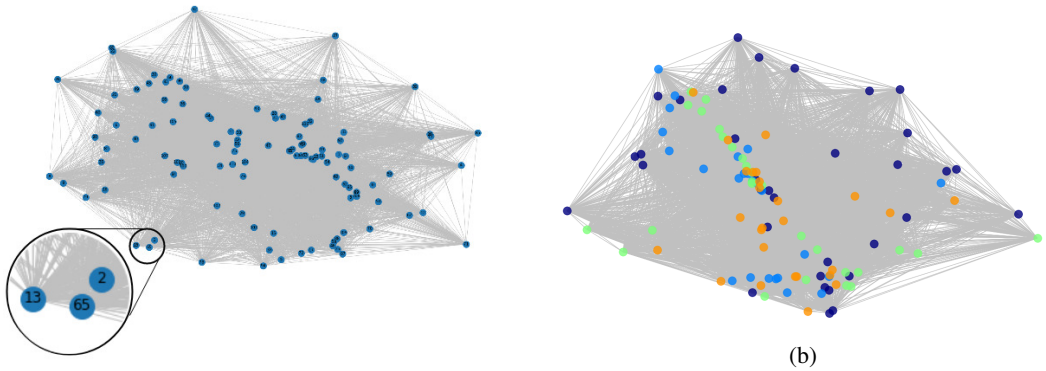
Feature	notation	number	Representation
nodes	$\mathcal{V}$	111	countries
node features	X	38	population etc. [3]
edges	$\mathbf{A}$	476	1 if countries i and j have traded and 0 otherwise
edge weights	$\mathcal{E}$	476	net trade value (USD)
node labels	$\mathcal{Y}$	4	income group

3.1 Data Collection and Representation

The data used for this study were from two different sources: the United Nations Comtrade Database and Kaggle. The United Nations Comtrade Database is an international trade database containing the reporter-partner trade statistics collated for about 170 countries over a certain period of time. These trade statistics include (1) Imports, exports, re-exports and re-imports, (2)commodities exchanges between the trade partner and reporter, (3) trade value in US dollars.

The data retrieved from Kaggle, on the other hand, had the profile of specific countries which included the geographical, financial, geological and other information. To ensure consistency, we targeted data for a particular year and in all found the trade and profile information for a total of 111 countries together with their income groups used as target labels.

Each country considered as a node had 38 node features being the profile collected for each country. We then used the net trade balance between the countries to construct an adjacency matrix such that there is an edge between the countries if the trade balance was not zero and vice versa. An entry of 1 was assigned where there is an edge and 0 otherwise. The trade values is US dollars made up the edge weight matrix. A summary of features and the representation is provided in Table 2.



(a) (b)

Figure 1: (a) shows the graph of countries (numbered for country reference) and links from trade data. (b) is a graph of countries with coloring indicating their income level.

4 Experiments

We trained the GNN models using PyTorch Geometric codebase[28]. We split the data into 80% and 20% training and test sets respectively for the classification task.

We separately tuned and used the best hyperparameter settings for each model and trained for a sufficient number of epochs.

4.1 Node Classification

We used the GCN, ChebNet, GAT and AGNN models described in section 2 to perform a multi-class node classification with the input graph. We split the data into train and test sets. We do this by

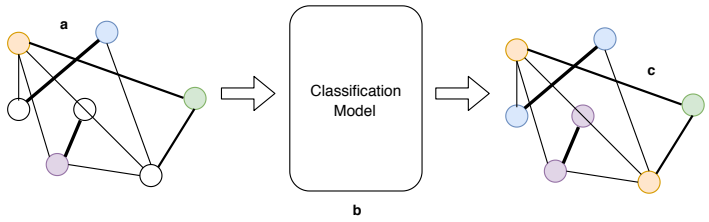


Figure 2: **a** is the input graph with partially labelled nodes and edges of different widths corresponding to the values of trade flow. **b** is the classification model and **c** is the output graph with inferred labels

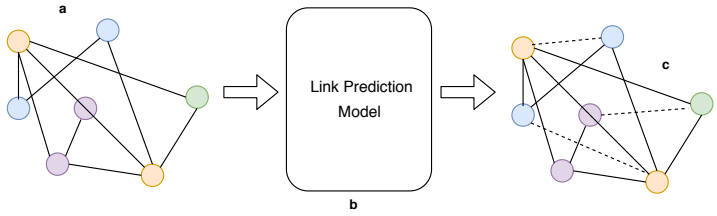


Figure 3: **a** is the input graph. **b** is a link prediction model. **c** is an output graph and dashed lines represents newly predicted links between nodes.

randomly masking out 20% of the total nodes and edge indices from the adjacency matrix and using the 80% for training. The learned model is then used to predict labels for the masked out set of nodes. We then report the classification accuracy on the test accuracy as a measure of model performance.

4.2 Link Prediction

In the link prediction task, we used the GAE and VGAE to learn a latent representation of the input graph. As illustrated in Figure 3, the input graph having few observed edges and hence a sparse adjacency matrix. This input graph is fed into the link prediction model which then reconstructs a new adjacency matrix representing a new neighbourhood structure for each node in the graph. We evaluate the reconstruction accuracy using the area under the curve (AUC) and average precision (AP) metrics.

4.3 GNN Architectures

Now, we show each model's architecture and its implementation details. These include the number of layers, hidden units, activation layers and other parameters. We used 111 input channels (number of nodes) and four output channels (number of node classes) for the GNN models as well as the linear model used for classification. We also optimized the negative likelihood loss using Adam[29].

GCN : Input $\rightarrow$ GCNConvLayer1 $\rightarrow$ ReLU $\rightarrow$ Dropout (p=0.5) $\rightarrow$ GCNConvLayer2 $\rightarrow$ LogSoftmax

ChebNet: Input $\rightarrow$ ChebyConvLayer1 $\rightarrow$ ReLU $\rightarrow$ Dropout (p=0.5) $\rightarrow$ ChebyConvLayer2 $\rightarrow$ LogSoftmax

GAT : Input $\rightarrow$ GATConvLayer1 $\rightarrow$ eLU $\rightarrow$ Dropout (p=0.6) $\rightarrow$ GATConvLayer2 $\rightarrow$ LogSoftmax

AGNN : Input $\rightarrow$ Dropout (p=0.5) $\rightarrow$ LinearLayer1 $\rightarrow$ ReLU $\rightarrow$ AGNNConvLayer1 $\rightarrow$ AGNNConvLayer2 $\rightarrow$ Dropout (p=0.5) $\rightarrow$ LinearLayer2 $\rightarrow$ LogSoftmax

GAE : Input $\rightarrow$ GCNConvLayer1 $\rightarrow$ ReLU $\rightarrow$ GCNConvLayer2

VGAE : Input $\rightarrow$ GCNConvLayer1 $\rightarrow$ ReLU $\rightarrow$ GCNConvLayer2

4.4 Baseline Models

In addition to the GNN classification models, we tested a neural network with fully connected layers and a logistic regressor as baseline models. The logistic regression model takes an input x

corresponding to the node features of the input graph and no information about the edges. We tune the hyperparameters separately for best performance. Our motivation is to evaluate how the adjacency matrix contributes to the quality of labels predicted for the test set.

4.5 Hyperparameter tuning

In our experiments, we used bayesian optimization[30][31] [4] to tune the hyperparameters. We defined the bounds on the parameter space as shown in the table 3 and obtained the best results for the learning rate of 0.001 and weight decay of 0.005 for all GNN models. The baseline linear model performed better at a slightly different hyperparameter setting of 0.4601 for learning rate and 0.06976 for weight decay.

Table 3: hyperparameters and predefined optimization bounds

hyperparameter	**upper bound**	**lower bound**
learning rate	0.0001	1.0
weight decay	0.005	1.0

4.6 Results and Evaluation

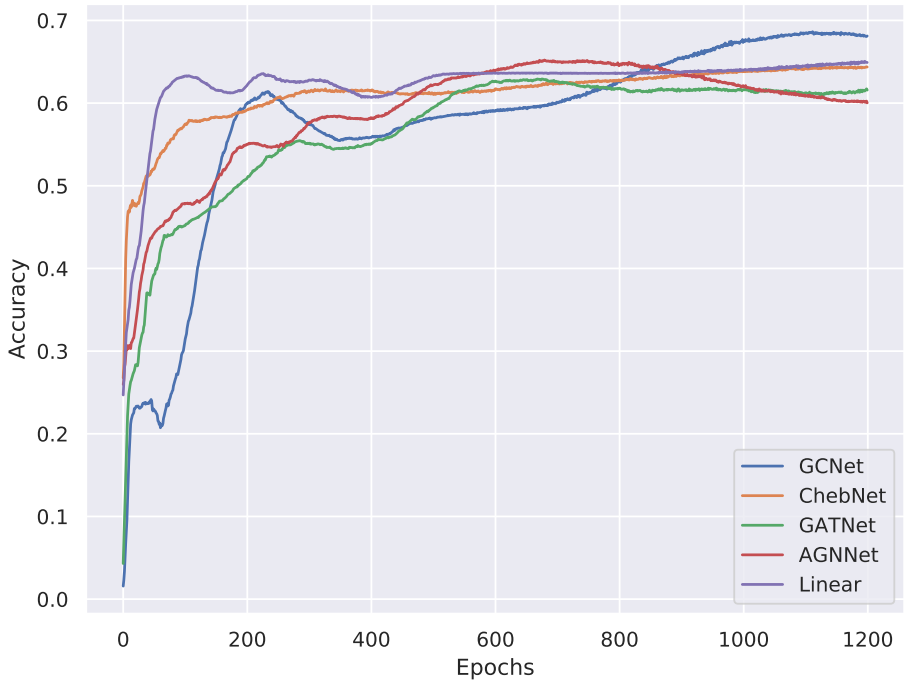
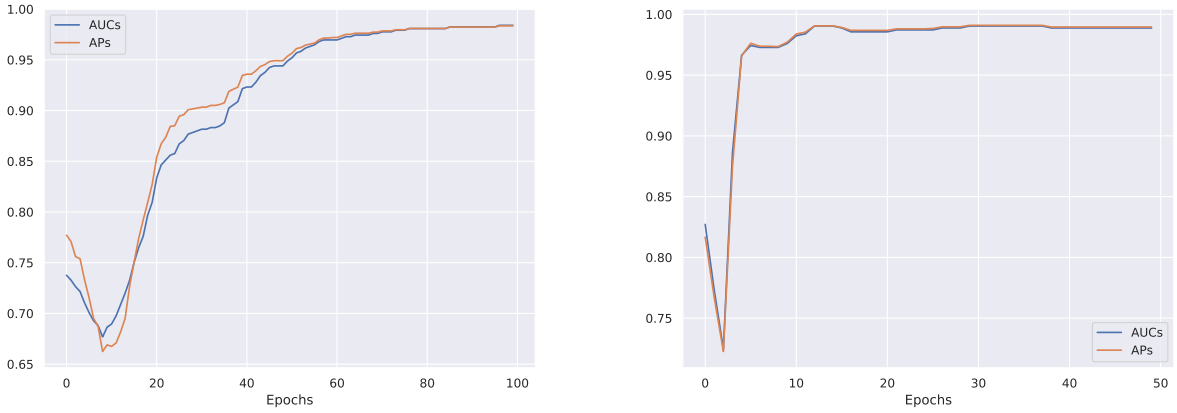


Figure 4: **a** Accuracy on test sets

4.6.1 Node classification

On the node classification tasks, we summarize results denoting mean results after 100 runs in Table 4. We report that the GCN has the best accuracy score. ChebNet also compares competitively with the linear baseline model.

[4]Code used available on https://github.com/fmfn/BayesianOptimization



(a) (b)

Figure 5: (a) AUCs on reconstruction of adjacency matrix (b) APs on reconstruction of the adjacency matrix

Table 4: Results for multi-class node classification task

	GCN	ChebNet	GAT	AGNN	Linear	Logistic Regression
Test Accuracy	**0.6812**	0.6436	0.6158	0.6003	0.6491	0.5758

Table 5: Results for link prediction task

	GAE	VGAE
AUC	0.9840	0.9888
Average Precision	0.9835	0.9896

4.6.2 Link Prediction

In Table 5, we report high AUC and AP scores for both GAE and VGAE. This is indicative of how well these models are reliably able to reconstruct the adjacency matrix from the learned latent representation.

5 Discussion and Conclusion

We approach downstream tasks — income level classification and potential trade partners prediction among countries — as a graph representation learning by leveraging their historical bilateral trade relationships. We use graph neural networks to classify countries and predict a new neighbourhood structure for each country. Our approach naturally points to a new direction machine learning particularly graph representation learning and application in the field of Economics. From the results we obtained, we confirm that our approach does well for the intended tasks and could be improved for future trade analysis. Since we only considered a standard static graph from data of trade activities between countries in a particular year, interesting future work is to use a time-series data of trade activities over a finite length of time in dynamic graph analysis. This will encourage analysis of (1) how countries evolve from one class to the other with time and (2) how trade activities change with time between countries. In the latter case, a temporal prediction of an edge between any two countries (nodes) will be an indication of how likely they will partner in trade in future.

Acknowledgements

The author would like to thank Michaël Defferrard for his great supervision and Avishek Joey Bose for the frequent discussion and comment after an early draft.

References

[1] Wikipedia. International trade — Wikipedia, the free encyclopedia, 2019. [Online; accessed 31 December 2019].

[2] David Ricardo. *On the Principle of Political Economy and Taxation*, chapter 1. Batoche Books, 52 Eby Street South, Kitchener, Ontario, N2G 3L1Canada, 3 edition, 1817.

[3] Patrick Steiner. Determinants of bilateral trade flows, 2015.

[4] Alan Deardorff. Determinants of bilateral trade: Does gravity work in a neoclassical world? *The Regionalization of the World Economy*, pages 7–32, 1998.

[5] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Yihong Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. *CoRR*, abs/1902.07243, 2019.

[6] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. *CoRR*, abs/1806.01973, 2018.

[7] World Bank Data Team. New country classifications by income level, 2017. Last accessed 31 December 2019.

[8] Thomas Chaney. The gravity equation in international trade: An explanation. Working Paper 19285, National Bureau of Economic Research, August 2013.

[9] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, Jan 2009.

[10] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[11] Smriti Bhagat, Graham Cormode, and S. Muthukrishnan. Node classification in social networks. *CoRR*, abs/1101.3291, 2011.

[12] Edouard Pineau and Nathan de Lara. Graph classification with recurrent variational neural networks. *CoRR*, abs/1902.02721, 2019.

[13] Jia Li, Yu Rong, Hong Cheng, Helen Meng, Wen-bing Huang, and Junzhou Huang. Semi-supervised graph classification: A hierarchical graph perspective. *CoRR*, abs/1904.05003, 2019.

[14] Antoine Jean-Pierre Tixier, Giannis Nikolentzos, Polykarpos Meladianos, and Michalis Vazirgiannis. Classifying graphs as images with convolutional neural networks. *CoRR*, abs/1708.02218, 2017.

[15] C. Aggarwal, G. He, and P. Zhao. Edge classification in networks. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, pages 1038–1049, May 2016.

[16] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. *CoRR*, abs/1802.09691, 2018.

[17] Ramnath Balasubramanyan, Frank Lin, and William W. Cohen. Node clustering in graphs: An empirical study. 2010.

[18] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75 – 174, 2010.

[19] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Graph neural networks: A review of methods and applications. *CoRR*, abs/1812.08434, 2018.

[20] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *CoRR*, abs/1312.6203, 2013.

[21] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *CoRR*, abs/1606.09375, 2016.

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1609.02907*, page 770–778, 2016.

[23] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. *International Conference on Learning Representations*, 2018. accepted as poster.

[24] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[25] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *NIPS Workshop on Bayesian Deep Learning*, 2016.

[26] Hervé Bourlard and Yves Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biological Cybernetics*, 59:291–294, 1988.

[27] Geoffrey E Hinton and Richard S. Zemel. Autoencoders, minimum description length and helmholtz free energy. In J. D. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems 6*, pages 3–10. Morgan-Kaufmann, 1994.

[28] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[29] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[30] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical bayesian optimization of machine learning algorithms. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'12, page 2951–2959, Red Hook, NY, USA, 2012. Curran Associates Inc.

[31] Eric Brochu, Vlad M. Cora, and Nando de Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *CoRR*, abs/1012.2599, 2010.

Appendix

No.	Node features
1	Surface area (km^2)
2	Population in thousands (2017)
3	Population density (per km^2, 2017)
4	Sex ratio (m per 100 f, 2017)
5	GDP: Gross domestic product (million current US$),
6	GDP growth rate (annual %, const. 2005 prices)
7	GDP per capita (current US$)
8	Economy: Agriculture (% of GVA)
9	Economy: Industry (% of GVA)
10	Economy: Services and other activity (% of GVA)
11	Employment: Agriculture (% of employed)
12	Employment: Industry (% of employed)
13	Employment: Services (% of employed)
14	Unemployment (% of labour force)
15	Agricultural production index (2004-2006=100)
16	Food production index (2004-2006=100)
17	International trade: Exports (million US$)
18	International trade: Imports (million US$)
19	International trade: Balance (million US$)
20	Balance of payments, current account (million US$)
21	Population growth rate (average annual %)
22	Urban population (% of total population)
23	Urban population growth rate (average annual %)
24	Fertility rate, total (live births per woman)
25	Refugees and others of concern to UNHCR (in thousands)
26	Infant mortality rate (per 1000 live births
27	Health: Total expenditure (% of GDP)
28	Health: Physicians (per 1000 pop.)
29	Education: Government expenditure (% of GDP)
30	Seats held by women in national parliaments %
31	Mobile-cellular subscriptions (per 100 inhabitants)
32	Mobile-cellular subscriptions (per 100 inhabitants)
33	Individuals using the Internet (per 100 inhabitants)
34	Threatened species (number)
35	CO2 emission estimates (million tons/tons per capita)
36	Energy production, primary (Petajoules)
37	Pop. using improved sanitation facilities (urban/rural, %)
38	Net Official Development Assist. received (% of GNI)