Chapter 1

Introduction

Many of the common neurological and neurodegenerative disorders, such as Alzheimer's disease, dementia and multiple sclerosis, have been associated with the abnormal patterns of apparent ageing of the brain [5]. This link has inspired numerous studies in estimation of the apparent brain age using brain imaging data [3]. The resulting *brain age gap* biomarker, defined as the discrepancy between the brain age estimate and the actual chronological age, can be used to understand the biological pathways behind the ageing process, assess an individual's risk for various brain disorders and identify new personalised treatment strategies.

Current state-of-the-art machine learning approaches for brain age gap estimation use structural magnetic resonance imaging (MRI) and DNA methylation data, considering each individual brain in isolation, often separating healthy controls from individuals with brain disorders, or developing separate models for each sex [5, 7]. At the same time, due to high computational complexity and lack of available datasets, few studies consider patterns that are shared across the sub-groups of patients, other important brain imaging modalities such as functional MRI time-series data, and clinical expertise of neurologists and psychiatrists.

This dissertation aims to explore the effectiveness of the *population graph* representation of patients in capturing the richness of a large-scale, population-wide neuroimaging dataset to estimate patients' brain age, and proposes a customisable end-to-end framework of graph neural network architectures – including graph convolutional [6] and graph attention [11] networks – to train them. This is similar to the approach presented in Parisot et al. [8, 9], where population graphs are used to classify patients as healthy or having a disorder such as Alzheimer's or autism spectrum disorder. In those papers, the nodes of the population graph represent subject-specific neuroimaging data, and edges capture pairwise non-imaging similarities between the different subjects.

Unlike in other state-of-the-art models, graph neural networks trained on population graphs have potential to learn from the entire cohort of healthy and affected patients of both sexes at once, capturing a wide range of confounding effects and detecting the variation in brain age trends of different sub-populations of subjects.

Chapter 2

Preparation

This section presents in more depth the brain age estimation method (Section 2.1) and the multiple data modalities that will be used for the brain age estimation task (Section 2.2). It also discusses population graphs (Section 2.3) and the neural network architectures that will be used to train them (Sections 2.4 and 2.5).

2.1 Brain age estimation

TODO this sounds extremely confusing (the concept itself is confusing)

To estimate the brain age gap the difference between the chronological and the brain age must be known. The common technique is to develop a machine learning method to estimate the chronological age from the brain imaging features. The estimated age is then considered to be the brain age although it has been trained to estimate the chronological age and the actual value of the brain age is never known. The reason is why it works... [7]

We represent the brain age y_b as the sum of the known chronological age y_c and the unknown brain age gap ε_g:

$$y_b = y_c + \varepsilon_g. \tag{2.1}$$

On the other hand, the machine learning model estimates the brain age y_b from some function of the brain imaging features X and a prediction error ε_e:

$$y_b = f(X) + \varepsilon_e. \tag{2.2}$$

Expressing y_c as $y_b - \varepsilon_g$ and substituting the previous result, we get the estimate of chronological age as

$$y_c = f(X) + \varepsilon_g - \varepsilon_e, \tag{2.3}$$

Since the machine learning models estimate chronological age as a function of brain imaging features (with some estimation error ε'_e)

$$y_c = f(X) + \varepsilon'_e, \tag{2.4}$$

the error of the proposed machine learning model ε'_e will contain in itself both the brain age gap ε_g (which we are interested in) and the brain age prediction error ε_e.

It is out of scope of this dissertation to prove that the brain age gap component ε_g is larger than the error term; however, the keen reader is referred to Niu et al. [7] where this is verified through correlation of the brain age gaps to the cognitive behaviour scores.

2.2 Neuroimaging dataset

2.2.1 United Kingdom Biobank

The United Kingdom Biobank (UK Biobank) [10] is a continuous population-wide study of over 500,000 participants containing a wide range of phenotypic and genetic data. Of particular relevance to this dissertation are the UK Biobank participants with neuroimaging data records, a total of 17,550 participants. The MRI scan data of those patients has been initially processed (denoised and motion-corrected) with the standard UK Biobank pipelines,[1], and further parcellated at the Department of Psychiatry by Dr Richard Bethlehem, Dr Rafael Romero-Garcia and Dr Lisa Ronan.[2] The details related to this neuroimaging dataset are described below.

2.2.2 Parcellation

A *parcellation* or *atlas* refers to the way the brain is split into meaningful regions for further analysis as a way to compress the intractable per-voxel measurements into per-parcel summaries. Whether two voxels of a brain belong to the same parcel may depend on their proximity, empirical evidence of the voxels being responsible for the same function and so forth.

Both functional and structural datasets in this dissertation will use one of the most common parcellations, developed by Glasser et al. [4], which divides the brain into 360 cortical regions and 16 subcortical regions.

2.2.3 Structural data

Structural brain imaging data refers to cortical thickness, surface area and gray matter volume, reported at parcel granularity.

These features have been derived from two types of an MRI image called T1-weighted and T2-weighted FLAIR, where each type emphasises different aspects of the scan and

[1]https://biobank.ctsu.ox.ac.uk/crystal/crystal/docs/brain_mri.pdf
[2]https://github.com/ucam-department-of-psychiatry/UKB

highlights some features more than the others. The image processing has been done using the HCP Freesurfer pipeline.[3]

2.2.4 Euler indices

Euler index[4] is a quality control metric which corresponds to the number of times the Freesurfer brain reconstruction software failed to seamlessly connect two 2D slices of an MRI image into the 3D model of the brain. The higher the Euler index, the worse is the quality of the scan. Euler indices might be used to remove the subjects with low-quality scans to avoid them affecting the analysis [5]. Alternatively, they can be used as a covariate in a machine learning model (as a brain similarity metric or a node feature) to correct for any bias in prediction that might be related to scan quality.

2.2.5 Functional data

The resting state functional MRI (rs-fMRI) is the representation of the brain activity over time. In the MRI scanner this is measured through *blood oxygenation level dependent* (BOLD) time-series which represent the changes in blood oxygenation of brain vessels as neural activity regulates the oxygen demand. The time-series are reported at parcel level as an average of all the voxel time series belonging to the parcel.

TODO *figure of BOLD timeseries*

We are interested in estimating which parts of the brain are connected to each other, which we do by making use of the assumption that parts of the brain that have related function would also have similar activity patterns. As a consequence, we would expect higher correlation of the corresponding BOLD time-series. For time-series T_1 and T_2, *Pearson's correlation* (denoted as r) is computed as

$$r(T_1, T_2) = \frac{\operatorname{cov}(T_1, T_2)}{\sigma_{T_1}\sigma_{T_2}} \tag{2.5}$$

where $\operatorname{cov}(\cdot,\cdot)$ denotes covariance and σ stands for standard deviation.

This (or other correlation types) to derive the *functional connectivity matrix* that stores the pairwise correlations between the different voxels as the overall representation of functional brain connectivity. For time-series $T_1, \ldots, T_N$,

$$\operatorname{fcm}(T_1, \ldots, T_N) = \begin{bmatrix} r(T_1, T_1) & \cdots & r(T_1, T_N) \\ \vdots & \ddots & \vdots \\ r(T_N, T_1) & \cdots & r(T_N, T_N) \end{bmatrix}, \tag{2.6}$$

of which (due to symmetry and the non-informative diagonal) only the flattened lower triangle is usually used for the machine learning implementations.

[3] https://www.ncbi.nlm.nih.gov/pubmed/23668970
[4] https://www.ncbi.nlm.nih.gov/pubmed/29278774

2.2.6 Phenotype data

For the population graph construction, the neuroimaging features are associated with the individual subjects (nodes). The similarity metric is defined by phenotype data, which is all important but not directly neuroimaging-related data. Some examples of phenotype data that is related to the brain include the patient's sex, mental health, other potential health issues, full-time education, bipolar disorder status etc. Indeed the metric that is being predicted is correlated with the brain tissue age indicative of various neurological and neurodegenerative diseases.

TODO *a table of the actual phenotypes used when the results are ready*

2.3 Population graphs

We connect the multi-modal MRI imaging (structural and/or functional), quality control and phenotype data of the set of patients S into a sparse *population graph* $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V}$ is the set of graph vertices (with one vertex uniquely representing one patient), and $\mathcal{E}$ is the set of edges (representing the *similarity* of patients).

Each vertex $v \in \mathcal{V}$ is the vector containing the individual subject's neuroimaging data, whether structural, functional, or both. The edge $(v, w) \in \mathcal{E}$ connects patients v and w based on phenotypic similarity that is defined by some *similarity metric* as described below.

2.3.1 Similarity metrics

The topology of the graph is determined by a similarity metric that uses the non-imaging (phenotypic) information of the subjects to create connections between the nodes containing brain imaging data. Defining a good similarity metric is important to correct for the confounding effects on the feature vectors (for example, subject's sex has an impact on the brain volume) as well as to cluster patients into the most informative neighbourhoods where the predicted variable (brain age) explains the most variance in node features.

Similarity metrics are defined using some *similarity function* $\mathrm{sim}(\cdot, \cdot)$ which takes two subjects and returns the similarity score between them (the higher the score, the more similar the subjects):

$$\mathrm{sim}(S_v, S_w) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}[M_i(v) = M_i(w)]. \tag{2.7}$$

Here $\{M_1, \ldots, M_n\}$ is a set of phenotypic metrics that are used in computing subject similarity (such as sex, mental health, fluid intelligence score etc.) and $\mathbf{1}[\cdot]$ is an indicator function, in this case returning a non-zero value when values for a given phenotypic feature M_i match for two subjects S_v and S_w.

TODO *Qualitative (categorial) metrics can be defined as Kronecker delta* δ *and quantitative (numerical) metrics have an indicator whether the distance between values* $M_i(v)$ *and* $M_i(w)$ *is within some* ϵ.

To avoid memory issues when $|\mathcal{E}| \sim O(|\mathcal{V}|^2)$ and minimise the size of the neighbourhood to only the highly similar subjects, a *similarity threshold* θ is defined such that

$$(S_v, S_w) \in \mathcal{E} \iff \mathrm{sim}(S_v, S_w) \geq \theta. \tag{2.8}$$

2.3.2 Training task

[6]: The task is to predict the label of the nodes, where the labels are visible only for the training and validation but not the test nodes. Semi-supervised learning allows the label information and parameter weights to spread to node neighbourhoods to the similar nodes as defined by the similarity metric,

A population graph is trained in a *semi-supervised* manner. This means that the entire dataset (all nodes and all labels) is included in the graph. At each training step (*epoch*), the parameters for all nodes are updated but only based on the feedback from training and validation nodes. The final predictive power of the model is evaluated based on its performance on previously hidden test node labels.

Labels are available for a small subset of nodes and are spread across neighbourhoods through a regularisation term such as Laplacian [6], predicting the labels for the remaining nodes

TODO *an illustration of the graph with marked training and validation nodes with visible labels and how the training happens on those followed by how the graph is evaluated on the test nodes where the parameters were updated but the labels never seen*

2.4 Graph convolutional networks

TODO Graph definition through degree and adjacency matrices, the weights of the edges being binary (either no edge or a weight-1 edge). Graph Laplacian. Diagonalisation of Laplacian matrix because it's positive semi-definite. Spatial vs spectral graph domains. Graph Fourier transform. Fourier transform as transformation to the eigenbasis obtained through graph Laplacian diagonalisation. Note that the basis depends completely on the graph structure and slight permutation can change the eigenbasis. Graph convolution with a filter g as multiplication of signal and filter in the Fourier domain. Parametric and non-parametric convolution filters using this g_θ; what does θ mean in relation to the Fourier coefficients; what exactly Fourier coefficients refer to. Eigenvalues as graph frequencies and learning as a low-pass filter of frequencies. Approximation of the Laplacian diagonalisation with Chebyshev polynomials (ChebNet) with the convolution defined for k-th order polynomials corresponding to k hops across neighbourhood. [2] Kipf and Welling simplifying ChebNet hops as stacking single-hop layers and using renormalisation trick for the node to use the most information from itself. [6] Averaging representations of neighbour features and smoothing labels as a consequence. [12] (this citation also very good for follow through for what the training is doing for all layers in general rather than a single feature vector, i.e. the training flow); can also mention [13]; convolution on regular vs irregular grids (i.e. intro to simple convolution before carrying on?); [2]: "convolution is linear operator that diagonalises in the fourier domain represented by the eigenvectors of the Laplacian operator"

[11]: graph definition in the Fourier domain based on the eigendecomposition of the graph Laplacian, filters further approximated by Chebyshev expansion of the graph Laplacian avoiding the expensive eigendecomposition operation (ChebNet); further improvements by filter restriction to local neighbourhoods and stacking them if necessary

2.4.1 Graph Fourier transform

2.5 Graph attention networks

[11] More based on the non-spectral (i.e. spatial) approaches where an operator is defined to work on the neighbourhoods of different sizes, maintaining the weight sharing property (usually done by defining a separate operator on the neighbourhoods of those sizes).

Self-attention is also added in to the concept where different parts of the node's neighbourhood are considered with different importance weights, getting a representation of the rest of the neighbourhood.

2.5.1 Graph attentional layer

For the layer of an N-node graph with F input and F' output features,

input node features $\{\mathbf{h_1}, \ldots, \mathbf{h_N}\}, \mathbf{h_i} \in \mathbb{R}^F$

output node features $\{\mathbf{h'_1}, \ldots, \mathbf{h'_N}\}, \mathbf{h'_i} \in \mathbb{R}^{F'}$

linear transformation with weight matrix $\mathbf{W} \in \mathbb{R}^{F' \times F}$

self attention $a : \mathbb{R}^{F'} \times \mathbb{R}^{F'} \to \mathbb{R}$

attention coefficients *can already include neighbourhoods* $e_{ij} = a(\mathbf{Wh}_i, \mathbf{Wh}_j)$

weight vector $\mathbf{a} \in \mathbb{R}^{2F'}$

$$\alpha_{ij} = \mathrm{softmax}_j(\mathbf{a}^T[\mathbf{Wh}_i \parallel \mathbf{Wh}_j]) \tag{2.9}$$

$$= \frac{\exp(\sigma_1(\mathbf{a}^T[\mathbf{Wh}_i \parallel \mathbf{Wh}_j]))}{\sum_{k \in \mathcal{N}_i} \exp(\sigma_1(\mathbf{a}^T[\mathbf{Wh}_i \parallel \mathbf{Wh}_k]))} \tag{2.10}$$

where α_{ij} is the attention coefficient for an edge $i \rightsquigarrow j$ (corresponding to the importance of features in node j to the features in node i), normalised across i's neighbourhood $\mathcal{N}_i$ (defined as e.g. all nodes one hop away); σ_1 is a non-linearity. $\|$ means concatenation

The coefficients α_{ij} and the weight matrix are used to compute the output features:

$$\mathbf{h}_i' = \sigma(\sum_{k \in \mathcal{N}_i} \alpha_{ij} \mathbf{W} \mathbf{h}_j) \tag{2.11}$$

2.5.2 Multiple attention

The above attention mechanism can be repeated several times to stabilise the performance, where one independent application of attention is called and attention head. The outputs of the independent attention heads are concatenated together until the last layer when they are averaged into a single output. For K attention heads, the results of (2.11) are concatenated:

$$\mathbf{h}_i' = \Big\|_{k=1}^{K} \sigma(\sum_{k \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \mathbf{h}_j) \tag{2.12}$$

Which is averaged in the last layer:

$$\mathbf{h}_i' = \sigma(\frac{1}{K} \sum_{k=1}^{K} \sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \mathbf{h}_j). \tag{2.13}$$

TODO Talk about the initial features, *diagram of the attention heads I guess* etc.

2.6 Requirements analysis

Tasks to be implemented (according to proposal: work to be done, success criteria, possible extensions), their relative importance (priority) and difficulty. Provide the order in which the tasks should be carried out to show good planning skills and account for the changes in proposal where the preprocessing pipeline turned out to be more important than the neural network implementation.

2.7 Software engineering practice

Implementing a flexible preprocessing pipeline which could be customised in the future for a variety of machine learning tasks even outside graph neural networks (a package).

Modular structure encapsulating specific task and having well defined documentations of the others.

Description of software engineering techniques: planning out and executing the project based on requirements analysis, setting tasks, and smoothly meeting the success criteria.

Code reuse (of open source well tested libraries), follow documentation and follow the PEP-8 style guide (or whatever PyCharm encourages).

Incremental development.

Modular structure: e.g. data processing, graph construction, graph neural network modules, robustness evaluation framework. Figure out where validation and cross validation sections should be (while training, separately etc.)

Diagram of the pipelines and module interaction (like in google design docs)

2.8 Choice of tools

PyTorch, PyTorch geometric extension, graph spectral filters/convolutions, message passing, time-series preprocessing into correlation matrices, IDEs, backup strategies

2.9 Starting point

- dataset, preprocessed by Dr Richard A.I. Bethlehem
- PyTorch, PyTorch geometric implementing GCN and GAT APIs and the graph API
- no previous experience with graph neural networks or the mathematics behind it
- no previous experience with PyTorch; limited experience with machine learning frameworks (basics of TensorFlow), no experience with neuroimaging data

Chapter 3

Implementation

Could be split into *preprocessing, parameter tuning, evaluation framework, software engineering techniques, repository overview*

3.0.1 Precomputation and preprocessing of connectivity matrices

Tangent works better than correlation or partial correlation.

3.0.2 Structural and functional data extraction

3.0.3 Graph construction pipeline

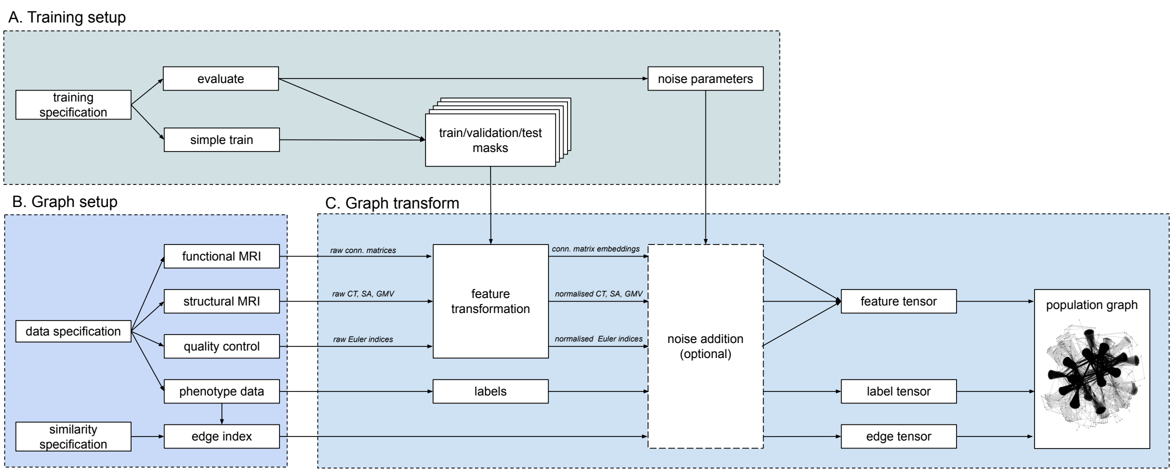

Figure 3.1: Graph pipeline.

3.0.4 Train, test, validation split

3.1 Non-graph baselines

I have played around with *xgboost*, ElasticNet and simple multilayer perceptrons to get a good idea of the minimum baselines my new architectures need to reach to be considered more effective.

3.2 Graph convolutional network

Describe the architecture in `BrainGCN`

3.3 Graph attention network

3.4 Robustness evaluation framework

3.5 Repository overview

TODO rougly split into *precompute* (with tests), *preprocessing* (with tests?), *similarity* (with tests), *evaluation*, *brainGAT, brainGCN*, ...

Figure out how to split framework modules.

Could have a base BrainGNN class which can then be *extended* with BrainGAT, BrainGCN.

Chapter 4

Evaluation

4.1 Overview of the predictive power for the two graph neural network approaches

Discuss Pearson's r, coefficient of determination r^2 and other common performance metrics.

4.2 Hyperparameters

Effects of hyperparameter tuning

4.3 Unit testing

4.4 Other quantitative and qualitative results

Statistical significance of the results compared to the baseline (e.g. Pearson's r method in Python seems to return some p-value of it..?)

4.5 Robustness of the graph neural networks to noisy and missing data

Add noise to the graphs and measure the rate of drop in predictive power.

4.6 Comparison against existing benchmarks

Compare to the Kaufmann et al.'s *xgboost* approach [5] ($r \sim 0.93$); and the other package that was cited in the same paper.

Possibly compare to other non-graph (relatively baseline) (neural network) architectures, e.g. ElasticNet, MLP,...

4.7 Interpretation of the model behaviour

One of the advantages of the graphs that they *should* be interpretable I guess.

Chapter 5

Conclusion

5.1 Successes and failures

5.2 The project in hindsight (lessons learnt)

Lessons learnt: graph *representation* is more important than the framework used. Good representations of the dataset can help guide the learning algorithm in the right way as it gets the intuitions faster just because of the way the data was represented in the first place. It's not good if it captures bias, but in this case representation made the difference between the model not learning anything and the model getting great results.

Was a mistake analysing functional data first without analysing the other methods in detail. Functional imaging data by itself gave very high dimensionality which either could not be learnt by the network because of the low number of examples or other factors. Most of the literature uses just the structural data for age prediction, and indeed this turned out to be more effective. Also makes sense intuitively as structural features would be related to the signs of brain atrophy while it is not clear the pattern of how resting state brain activity would change with ageing brain.

Mention Niu et al. 2019 raising the issue that there is systematic bias in brain age gap prediction but not many studies use this knowledge to correct for it.

5.3 Possible continuations of the project

- Include DNA methylation data as it is widely used in other studies and is claimed to improve the predictive power of the model [1].

Bibliography

[1] James Cole, Stuart Ritchie, Mark Bastin, Maria Valdés Hernández, et al. Brain age predicts mortality. *Molecular Psychiatry*, 23(5):1385–1392, 2018.

[2] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pages 3844–3852, 2016.

[3] Katja Franke and Christian Gaser. Ten years of BrainAGE as a neuroimaging biomarker of brain aging: What insights have we gained? *Frontiers in Neurology*, 10:789, 2019.

[4] Matthew F Glasser, Timothy S Coalson, Emma C Robinson, Carl D Hacker, John Harwell, Essa Yacoub, Kamil Ugurbil, Jesper Andersson, Christian F Beckmann, Mark Jenkinson, et al. A multi-modal parcellation of human cerebral cortex. *Nature*, 536(7615):171, 2016.

[5] Tobias Kaufmann, Dennis van der Meer, Nhat Trung Doan, Emanuel Schwarz, et al. Common brain disorders are associated with heritable patterns of apparent aging of the brain. *Nature Neuroscience*, 22(10):1617–1623, 2019.

[6] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[7] Xin Niu, Fengqing Zhang, John Kounios, and Hualou Liang. Improved prediction of brain age using multimodal neuroimaging data. *Human Brain Mapping*, 2019.

[8] Sarah Parisot, Sofia Ira Ktena, Enzo Ferrante, Matthew Lee, et al. Spectral graph convolutions on population graphs for disease prediction. *MICCAI*, 2017.

[9] Sarah Parisot, Sofia Ira Ktena, Enzo Ferrante, Matthew Lee, et al. Disease prediction using graph convolutional networks: Application to Autism Spectrum Disorder and Alzheimer's disease. *Medical Image Analysis*, 48:117–130, August 2018.

[10] Cathie Sudlow, John Gallacher, Naomi Allen, Valerie Beral, Paul Burton, John Danesh, Paul Downey, Paul Elliott, Jane Green, Martin Landray, et al. UK Biobank: an open access resource for identifying the causes of a wide range of complex diseases of middle and old age. *PLOS Medicine*, 12(3):e1001779, 2015.

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

[12] Felix Wu, Tianyi Zhang, Amauri Holanda de Souza Jr, Christopher Fifty, Tao Yu, and Kilian Q Weinberger. Simplifying graph convolutional networks. *arXiv preprint arXiv:1902.07153*, 2019.

[13] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A comprehensive survey on graph neural networks. *arXiv preprint arXiv:1901.00596*, 2019.