Chapter 1

Introduction

Many of the common neurological and neurodegenerative disorders, such as Alzheimer's disease, schizophrenia and multiple sclerosis, have been associated with the abnormal patterns of apparent ageing of the brain [1]. This link has inspired numerous studies in estimation of the apparent brain age using brain imaging data [2]. 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 estimation use structural magnetic resonance imaging (MRI) and DNA methylation data, considering each individual brain in isolation, modelling healthy controls separately from individuals with brain disorders, or developing separate models for each sex [1, 3]. 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 proposes a customisable end-to-end framework of graph neural network architectures – including graph convolutional [4] and graph attention [5] networks – as one way to combine the richness of structural, functional MRI and non-imaging modalities when predicting the brain age. The neural networks are trained in a semi-supervised manner on a *population graph* representation of patients, where the nodes of the population graph represent subject-specific neuroimaging data, and edges capture pairwise non-imaging similarities between the different subjects. This follows closely the approach presented in the work of Parisot et al. [6, 7], where population graphs are used to classify patients as healthy or suffering from Alzheimer's disease or autism spectrum disorder.

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

Similar to the model proposed in Niu et al. [3], we express the *brain age* (y_b) as the sum of the known *chronological age* (y_c) and the unknown *brain age gap* (ε_g):

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

The brain age gap estimation methodologies (such as *BrainAGE* [2]) assume that for healthy subjects in the general population the brain age corresponds to chronological age:

$$y_b \approx y_c, \quad \varepsilon_g \text{ small.} \tag{2.2}$$

Our goal is to estimate the brain age y_b by fitting some function f of the brain imaging features X. It will give a prediction error ε_e:

$$y_b = f(X) + \varepsilon_e. \tag{2.3}$$

From Equations 2.1 and 2.3, the estimate of chronological age is, on the other hand,

$$\begin{aligned} y_c &= f(X) + \varepsilon_e - \varepsilon_g && (2.4)\\ &= f(X) + \varepsilon. && (2.5) \end{aligned}$$

Since the true brain age is unknown, any (semi-)supervised machine learning model can only work on chronological age labels following Equation 2.5, where the prediction error ε contains both the brain age gap ε_g and the brain age estimation error ε_e. However, the assumption in Equation 2.2 still permits learning the brain age function f as long as the model is trained only on healthy subjects, so that $\varepsilon \approx \varepsilon_e$. When the same model is applied to subjects with various brain health conditions, the errors outside of the ε_e could be assumed to come from the brain age gap.

An alternative method, which does not restrict training data only to healthy subjects, is proposed in Niu et al. [3]. However, it requires experimentally verifying (e.g. through subjects' performance in cognitive behaviour tests) that ε depends primarily on the brain age gap ε_g and not the brain age prediction error ε_e, which is out of scope of this dissertation.

2.2 Neuroimaging dataset

2.2.1 United Kingdom Biobank

The United Kingdom Biobank (UK Biobank) [8] 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 voxels are split into meaningful regions depending on their proximity, empirical evidence of them being responsible for the same function and so forth. This compresses the high dimensional per-voxel measurements into per-parcel summaries of much lower dimension, the latter being easier to manage in further analysis.

Both structural and functional datasets in this dissertation will use one of the most common parcellations, developed by Glasser et al. [9], 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.

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

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 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 [1]. 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) represents 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 time series of voxels belonging to the same parcel.

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 functions 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

$$r(T_1, T_2) = \frac{\mathrm{cov}(T_1, T_2)}{\sigma_{T_1}\sigma_{T_2}} \tag{2.6}$$

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

The correlations are used to derive the *functional connectivity matrix* storing pairwise correlations between the different voxels (or parcels) as the overall representation of functional brain connectivity. For time-series $T_1, \ldots, T_N$,

$$\mathrm{fcm}(T_1, \ldots, T_N) = \begin{bmatrix} r(T_1, T_1) & \cdots & r(T_1, T_N) \\ \vdots & \ddots & \vdots \\ r(T_N, T_1) & \cdots & r(T_N, T_N) \end{bmatrix}, \tag{2.7}$$

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

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

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

2.2.6 Phenotype data

In this dissertation, the phenotype data is defined as all subject data that does not come from MRI scans. The features related to this project include the subject's sex, the psychiatric disorder diagnoses, mental health status, education and so forth.

The purpose of phenotype data is to define the inter-subject similarity score, which will determine the edges in the population graph. This is further discussed in the next section. The full list of features that were used to define similarity metrics is provided in the Appendix.

TODO Appendix.

2.3 Population graphs

The multi-modal MRI imaging (structural, functional, quality control) and phenotypic data of the set of patients S can be connected into a undirected *population graph* $G = (V, E)$, where V is the set of graph nodes (with one node uniquely representing one subject), and E is the set of edges (representing the similarity of subjects).

Each node $v \in V$ is the vector containing the individual subject's neuroimaging data, whether structural, functional, or both. The edge $(v, w) \in E$ connects subjects $s_v, s_w \in S$ based on phenotypic similarity, defined by some similarity metric.

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 edges between the nodes with brain imaging data. Defining a good similarity metric is important to correct for the confounding effects on the feature vectors (for example, the subject's sex affects the brain volume) as well as to cluster subjects into the most informative neighbourhoods. For example, in this dissertation the neighbourhoods that have similar brain age gaps could be useful.

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

$$\mathrm{sim}(s_v, s_w) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}[M_i(s_v) = M_i(s_w)]. \tag{2.8}$$

Here $\{M_1, \ldots, M_n\}$ is a set of phenotypic metrics that are used to compute subject similarity 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. (In practice, if the metric is floating point, "matching" could be defined in terms of phenotypic metrics being within some constant ϵ.)

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

$$(v, w) \in E \iff \text{sim}(s_v, s_w) \geq \theta. \tag{2.9}$$

2.3.2 Training task

For node label prediction tasks such as brain age prediction, population graphs are trained in a *semi-supervised* manner: while the entire dataset (every node and edge) is included in the graph, the labels are available only for a subset of nodes [4]. At each training step, the feedback from nodes with available labels is used to update the parameters for the entire graph, which could be seen as information being "propagated" from a labelled node to the neighbours that are similar to it (as defined by the similarity metric). After the model is trained, every node in the population graph has a prediction for its label. The predictive power of a model can be evaluated based on its performance for a set of test nodes, for which the labels had been known but invisible to the model at the training stage.

2.4 Graph convolutional networks

To make the computation independent of (irregular) graph topology, the graph is first transformed from spatial (Euclidean) to spectral (Fourier) domain. The more expensive convolution operation in the Euclidean domain corresponds to a cheaper multiplication operation in the Fourier domain.

2.4.1 Graph spectral decomposition

In spectral analysis, a graph $G = (V, E)$ is represented by its *normalised Laplacian* matrix [10]:

$$\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}, \tag{2.10}$$

where $\mathbf{I}$ is the identity matrix, $\mathbf{D} \in \mathbb{N}^{|V|\times|V|}$ is the diagonal degree matrix of the graph nodes, and $\mathbf{A} \in \{0, 1\}^{|V|\times|V|}$ is the adjacency matrix such that $a_{ij} = \mathbf{1}[(i, j) \in E]$. The Laplacian uniquely represents the graph as it is based on the graph's topology.

The positive semidefinite Laplacian matrix is decomposed as

$$\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\mathrm{T}}, \tag{2.11}$$

where $\mathbf{\Lambda}$ is the diagonal eigenvalue matrix, and $\mathbf{U}$ is the eigenbasis defining the graph Fourier domain: for a signal $\mathbf{x} \in \mathbb{R}^n$ (e.g. features of some node), $\hat{\mathbf{x}} = \mathbf{U}^{\mathrm{T}}\mathbf{x}$ is the *graph Fourier transform* of $\mathbf{x}$, and $\mathbf{x} = \mathbf{U}\hat{\mathbf{x}}$ is its inverse [11].

2.4.2 Spectral graph convolution

For a filter $\mathbf{g} \in \mathbb{R}^m$ with a diagonal matrix $\hat{\mathbf{G}} = \text{diag}(\mathbf{U}^\text{T}\mathbf{g})$ containing the filter's spectral coefficients, the graph convolution of a signal $\mathbf{x}$ is defined as

$$\mathbf{x} *_G \mathbf{g} = \mathbf{U}((\mathbf{U}^\text{T}\mathbf{x}) \odot (\mathbf{U}^\text{T}\mathbf{g})) = \mathbf{U}\hat{\mathbf{G}}\mathbf{U}^\text{T}\mathbf{x} \tag{2.12}$$

where $\odot$ is the element-wise (Hadamard) product [11].

2.4.3 Optimising graph convolutions

This section reviews the steps towards optimised graph convolutional networks (GCNs) as proposed by Kipf and Welling [4].

ChebNets

To avoid the $O(n^2)$ filtering operation due to the matrix-vector multiplications in (2.12), Defferrard et al.'s [10] ChebNets use recursively defined *Chebyshev polynomials*

$$T_i(\mathbf{x}) = 2\mathbf{x}T_{i-1}(\mathbf{x}) - T_{i-2}(\mathbf{x}) \tag{2.13}$$

$$T_1 = \mathbf{x} \tag{2.14}$$

$$T_0 = 1 \tag{2.15}$$

to approximate the filter $\hat{\mathbf{G}}$ as

$$\hat{\mathbf{G}} \approx \sum_{i=0}^{k} \theta_i T_i(\tilde{\mathbf{\Lambda}}) \tag{2.16}$$

where $\tilde{\mathbf{\Lambda}} = 2\mathbf{\Lambda}/\lambda_\text{max} - \mathbf{I}$, λ_max is the largest eigenvalue in $\mathbf{\Lambda}$, and θ_i are coefficients such that $\hat{\mathbf{G}} = \sum_i \theta_i \mathbf{\Lambda}^i$ [11]. The truncation coefficient (the degree of the polynomial) k corresponds to the neighbourhood of at most k hops away from the node of interest.

Next, it can be proved [12] that

$$T_i(\tilde{\mathbf{L}}) = \mathbf{U}T_i(\tilde{\mathbf{\Lambda}})\mathbf{U}^\text{T} \tag{2.17}$$

with $\tilde{\mathbf{L}} = 2\mathbf{L}/\lambda_\text{max} - \mathbf{I}$. The result is the approximated spectral graph convolution:

$$\mathbf{x} *_G \mathbf{g} \approx \sum_{i=0}^{k} \theta_i T_i(\tilde{\mathbf{L}})\mathbf{x}. \tag{2.18}$$

Kipf and Welling's graph convolutional networks

assume $k = 1$, $\lambda_{\max} = 2$ so that the polynomial in 2.18 simplifies to only two learnable parameters but even they can be further simplified to 1 parameter.

renormalisation trick to avoid the instabilities

2.4.4 Layers of the graph convolutional network

Averaging representations of neighbour features and smoothing labels as a consequence. [11]

2.5 Graph attention networks

[5] 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* $e_{ij} = a(\mathbf{W}\mathbf{h}_i, \mathbf{W}\mathbf{h}_j)$

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

$$\alpha_{ij} = \text{softmax}_j(\mathbf{a}^T[\mathbf{W}\mathbf{h}_i \parallel \mathbf{W}\mathbf{h}_j]) \tag{2.19}$$

$$= \frac{\exp(\sigma_1(\mathbf{a}^T[\mathbf{W}\mathbf{h}_i \parallel \mathbf{W}\mathbf{h}_j]))}{\sum_{k \in \mathcal{N}_i} \exp(\sigma_1(\mathbf{a}^T[\mathbf{W}\mathbf{h}_i \parallel \mathbf{W}\mathbf{h}_k]))} \tag{2.20}$$

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. $\parallel$ means concatenation

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

$$\mathbf{h}'_i = \sigma(\sum_{k \in \mathcal{N}_i} \alpha_{ij} \mathbf{W} \mathbf{h}_j) \tag{2.21}$$

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 (2.21) are concatenated:

$$\mathbf{h}'_i = \Big\|_{k=1}^{K} \sigma(\sum_{k \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \mathbf{h}_j) \tag{2.22}$$

Which is averaged in the last layer:

$$\mathbf{h}'_i = \sigma(\frac{1}{K} \sum_{k=1}^{K} \sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \mathbf{h}_j). \tag{2.23}$$

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

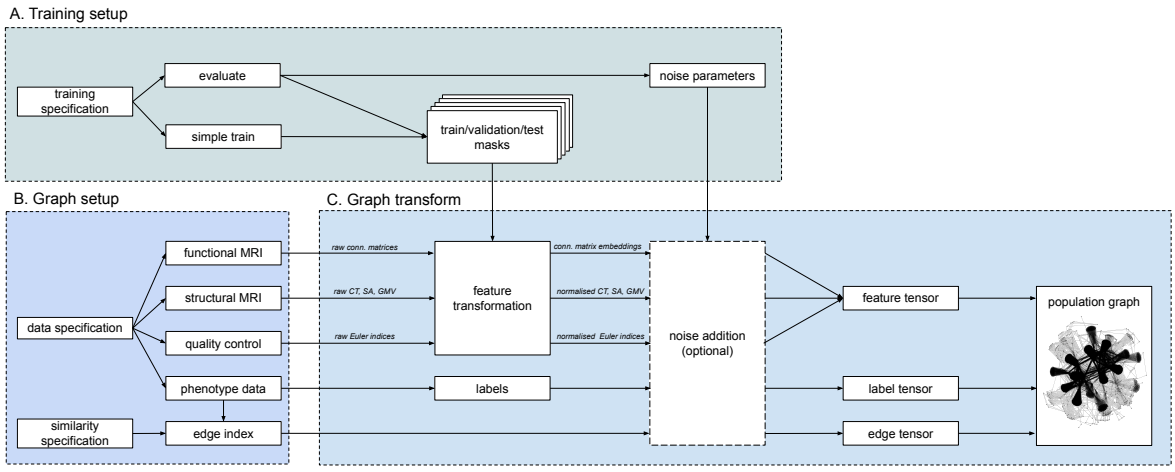

Figure 3.1: Graph pipeline.

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 [1] ($r \sim 0.93$); 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 [13].

Bibliography

[1] 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.

[2] 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.

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

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

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

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

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

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