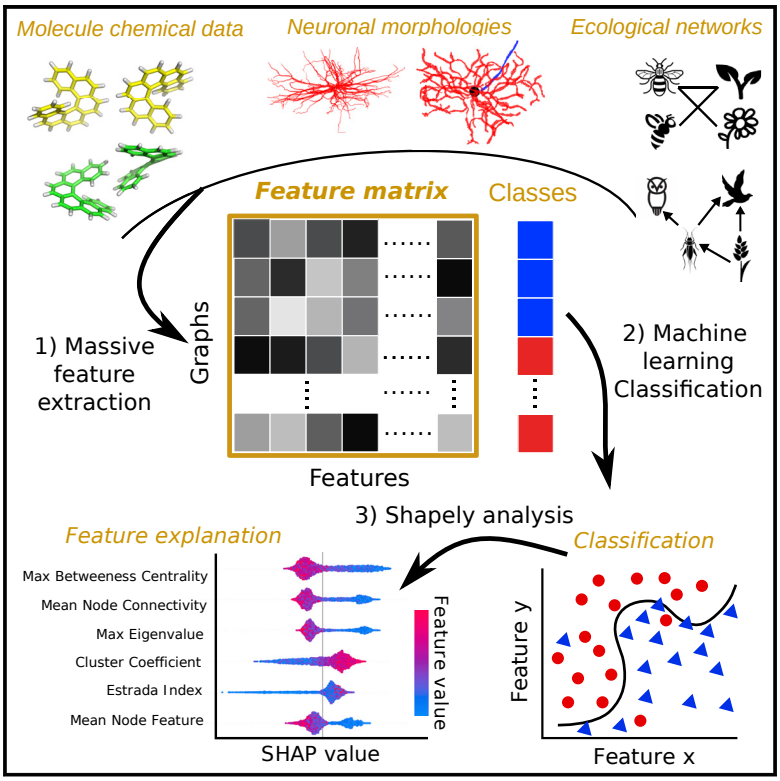
Descriptor

Patterns

HCGA: Highly comparative graph analysis for network phenotyping

Graphical abstract



Highlights

- Draws on decades of graph theory to compute a large set of network properties
- Uses graph features to give quantifiable insights into network data
- State-of-the-art classification accuracy on benchmark datasets
- Tested and documented open-source Python software package

Authors

Robert L. Peach, Alexis Arnaudon, Julia A. Schmidt, ..., Kim E. Jelfs, Sophia N. Yaliraki, Mauricio Barahona

Correspondence

r.peach13@imperial.ac.uk (R.L.P.), m.barahona@imperial.ac.uk (M.B.)

In brief

The wide applicability of graphs to many scientific domains has fueled a large corpus of network theoretical literature. Here, we introduce HCGA, a package that computes a comprehensive collection of network-based properties and uses them as features for the highly comparative analysis of graph datasets. Our methodology achieves state-of-the-art accuracy on benchmarks and provides interpretable insights into novel datasets.

Peach et al., 2021, Patterns 2, 100227
April 9, 2021
https://doi.org/10.1016/j.patter.2021.100227

CellPress

Patterns

CellPress
OPEN ACCESS

Descriptor

HCGA: Highly comparative graph analysis for network phenotyping

Robert L. Peach,[1,5,6,*] Alexis Arnaudon,[2,6] Julia A. Schmidt,[3] Henry A. Palasciano,[1] Nathan R. Bernier,[4] Kim E. Jelfs,[3] Sophia N. Yaliraki,[3] and Mauricio Barahona[1,7,*]

[1]Department of Mathematics, Imperial College London, SW7 2AZ London, UK
[2]Blue Brain Project, École polytechnique fédérale de Lausanne (EPFL), Campus Biotech, 1202 Geneva, Switzerland
[3]Department of Chemistry, Imperial College London, SW7 2AZ London, UK
[4]Miraex, EPFL Innovation Park, 1024 Ecublens, Switzerland
[5]Present address: Department of Neurology, University Hospital Würzburg, Würzburg, Germany
[6]These authors contributed equally
[7]Lead contact
*Correspondence: r.peach13@imperial.ac.uk (R.L.P.), m.barahona@imperial.ac.uk (M.B.)
https://doi.org/10.1016/j.patter.2021.100227

THE BIGGER PICTURE Graphs are used to model data across many scientific domains, from relationships in social networks to the interactions between atoms in a molecule. The study of graphs and networks has a long history, and a vast literature exists characterizing different properties and aspects of their structure. Here, we present HCGA, a software package that computes a large, comprehensive collection of graph properties and uses them as features for the systematic, highly comparative analysis of graph datasets. Our toolbox makes it possible for researchers from diverse scientific fields to easily access an extensive set of graph-theoretical tools to generate interpretable and quantifiable insights into their data.

1 2 3 4 5 **Production:** Data science output is validated, understood, and regularly used for multiple domains/platforms

SUMMARY

Networks are widely used as mathematical models of complex systems across many scientific disciplines. Decades of work have produced a vast corpus of research characterizing the topological, combinatorial, statistical, and spectral properties of graphs. Each graph property can be thought of as a feature that captures important (and sometimes overlapping) characteristics of a network. In this paper, we introduce HCGA, a framework for highly comparative analysis of graph datasets that computes several thousands of graph features from any given network. HCGA also offers a suite of statistical learning and data analysis tools for automated identification and selection of important and interpretable features underpinning the characterization of graph datasets. We show that HCGA outperforms other methodologies on supervised classification tasks on benchmark datasets while retaining the interpretability of network features. We exemplify HCGA by predicting the charge transfer in organic semiconductors and clustering a dataset of neuronal morphology images.

INTRODUCTION

Graphs provide an elegant and powerful formalism to represent complex systems.[1] Across many scientific disciplines there are increasing volumes of data that are naturally described as graphs (or networks) and can leverage the extensive results in graph theory to solve research problems. Among many others, examples include linking the structural motifs of proteins and their function,[2–4] aiding the diagnosis of diseases using fMRI data,[5] understanding structural properties of organic crystal structures for electron transport,[6] or modeling network flows, e.g., city traffic,[7] information (or misinformation) spread in a social network,[8,9] or topic affinity in a citation network.[10] The growing importance of such network data has driven the development of a multitude of methods for investigating and revealing relevant topological, combinatorial, statistical, and spectral properties of graphs, e.g., node centralities,[11,12] assortativity,[13,14] path-based properties,[15] graph distance measures,[16,17]

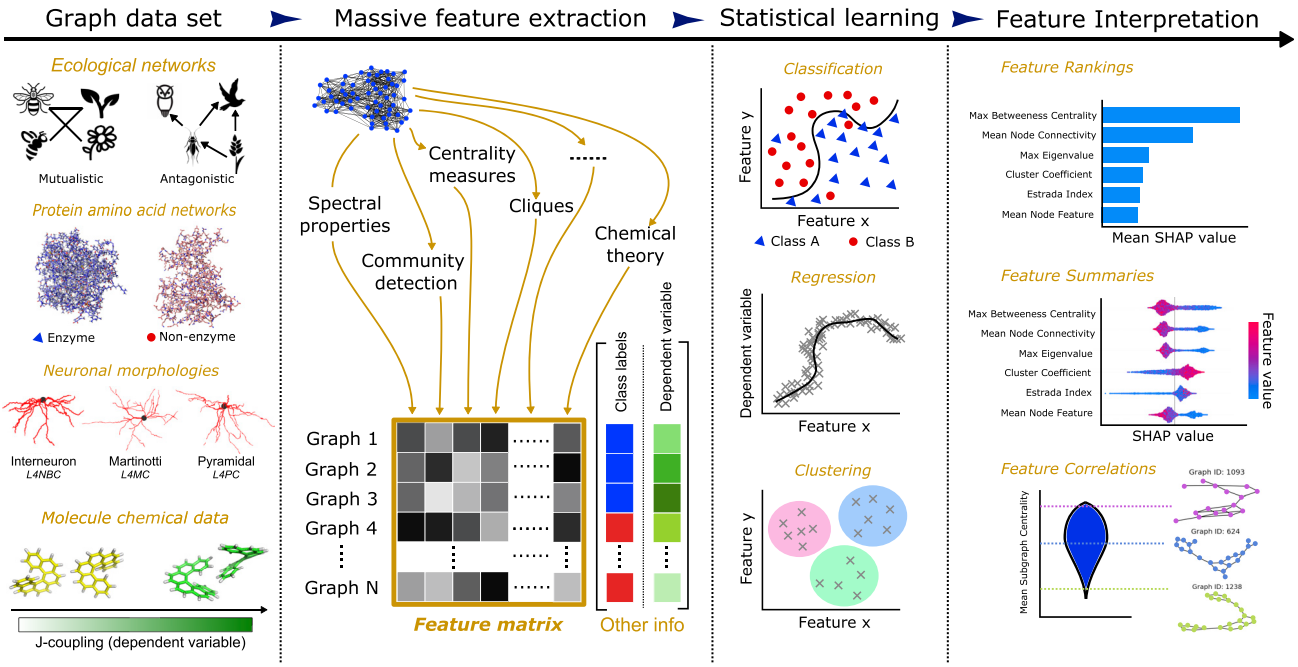


Figure 1. Overview of the HCGA pipeline

HCGA can be applied to a wide range of graph datasets, such as ensembles of synthetic graphs, molecular structures, cell morphologies, and social and ecological networks, among many others. HCGA first constructs a large feature matrix for the dataset by computing for each network a wide range of graph-theoretical properties (several thousands) that have been compiled from classic and more recent literature. This feature matrix is then used to perform statistical analyses on the graph dataset, including supervised and unsupervised learning tasks (e.g., classification, clustering, and regression) in conjunction with any additional information known, such as class labels, a continuous dependent variable, or node features. HCGA preserves the interpretability of the features and provides tools (e.g., Shapley values) to characterize the importance of features for the prediction task as an aid to get insights into network-based discovery for the dataset at hand. HCGA is an open platform which allows for the expansion of the set of graph-theoretical features, as well as allowing for the inclusion of additional statistical analyses.

connectivity,[18] or community detection,[19,20] to name but a few in the highly interdisciplinary area of network science.

When analyzing network datasets, or when posing a research question in terms of network properties, it is usually not immediately clear which graph-theoretical methodologies should be used; particularly in real-world applications, which do not conform to clear-cut constructive assumptions and, hence, where differences between complex systems may be subtle. For example, we may want to identify characteristics of a social network that aid the spread of fake news,[21] or identify structural properties that would help us predict the toxicity of a given molecule[22] without restricting *a priori* the types of graph properties to be considered. The viability and enormous potential of highly comparative data-driven analyses of graphs was previously shown[23]; however, existing software packages that derive summary statistics of graphs often only compute a small set of features that are usually chosen to target a subject area (e.g., neuroscience[24] or biology[25]). Despite the myriad of graph properties, there currently exists no systematic way to leverage the wide spectrum of available measures to identify graph features that best characterize a given problem.

Here, we introduce highly comparative graph analysis (HCGA), a modular, expandable Python software package (the HCGA Python package is available at https://github.com/barahona-research-group/hcga) that allows researchers to perform massive graph feature extraction together with statistical learning and analysis of feature importance. Our computational framework is illustrated in Figure 1 and takes both inspiration and ideas from HCTSA and CATCH22, powerful frameworks for time series feature extraction.[26–29] Given a set of complex systems modeled as networks representing, e.g., molecules, proteins, neuronal morphologies, transportation routes, or ecological or social networks, HCGA first extracts from each network in the set a few thousand graph features, each encoding a different interpretable network property. The selection of features to be extracted is flexible and can be adapted by the researcher to the particular problem at hand. Following the feature extraction step, each graph in the dataset is described as a high-dimensional feature vector, and the whole dataset is encoded as a feature matrix. To facilitate data-driven analyses of the dataset, HCGA includes a suite of tools for statistical learning aimed at classification, regression, and unsupervised learning. Since HCGA preserves the interpretability of the features, our framework also includes in-depth feature importance analysis using Shapley additive values (SHAP) to aid feature selection aimed at deriving scientific insights.[30] HCGA thus removes the time-consuming and subjective task of implementing individual graph-theoretical methods for the analysis of network datasets. The structure of HCGA is modular and open-source, allowing researchers from any research area to contribute further graph-theoretical features; to extend the statistical learning tools; or to improve visualization modules for the use of the research community.

Table 1. Classification results on biochemical benchmark datasets obtained by HCGA using the XGBoost classifier and compared against other methodologies

	Datasets (biochemical networks–benchmarks)				
Method	Enzymes	Proteins	D&D	NCI1	MUTAG
Multi-Hop[34]	56.1 ± 9.1	76.7 ± 2.9	–	77.3 ± 1.7	89.8 ± 5.6
DGCNN[35]	38.9 ± 5.7	72.9 ± 3.5	76.6 ± 4.3	76.4 ± 1.7	–
GIN[36]	59.6 ± 4.5	73.3 ± 4.0	75.3 ± 2.9	80.0 ± 1.4	–
ECC[37]	29.5 ± 8.2	72.3 ± 3.4	72.6 ± 4.1	76.2 ± 1.4	–
DiffPool[38]	59.5 ± 5.6	73.7 ± 3.5	75.0 ± 3.5	76.9 ± 1.9	–
HCGA	69.0 ± 5.5	75.7 ± 4.0	79.9 ± 3.5	79.4 ± 2.0	90.3 ± 6.6
HCGA*	74.0 ± 2.4	79.3 ± 3.5	83.8 ± 2.6	79.5 ± 2.1	91.5 ± 4.9

The HCGA accuracies are obtained from the top 100 features, whereas HCGA* accuracies are obtained from features selected by optimizing against a validation set. The hyperparameters of the XGBoost classifier were not optimized for any of the HCGA runs. The classification accuracies for MUTAG are not available for most methods due to long computation times; however, we retain this example to show that HCGA is also capable of analyzing large datasets. Results are presented as mean ± standard deviation.

RESULTS

We illustrate how HCGA can be employed in three tasks aimed at network-based scientific discovery: supervised classification, regression, and unsupervised clustering.

Supervised classification of benchmark graph datasets from biochemistry and social science

We evaluated the performance of HCGA for supervised classification using known collections of labeled graphs that have been used as benchmarks. The sets comprise five biochemical datasets (proteins,[31] enzymes,[32] D&D, NCI1, MUTAG) and six standard social media datasets (collab, reddit-binary, reddit-multi-5k, reddit-multi-12k, IMDB-binary, IMDB-multi); see Table S2 for details. In Tables 1 and 2, we show the classification accuracy achieved on the biochemical and social datasets, respectively, using an XGBoost classifier (with default settings) applied to the top 100 features extracted with HCGA. Our results show that HCGA achieves top accuracy without optimizing the hyperparameters of the XGBoost classifier (as would be the case in a realistic user case scenario) when compared with popular deep-learning methodologies and Kernel algorithms under the Fair Comparison protocol.[33] If we choose the best subset of features through cross-validation, the results are improved further (still without optimizing the classifier).

In addition to its high classification performance, HCGA preserves the interpretability of features, which can then be ordered according to their importance for each modeling problem. In the supplemental information we exemplify a more detailed analysis (Figures S1 and S2) of the features underpinning the classification of the benchmark proteins dataset (classifying proteins as enzymatic or non-enzymatic). Among the features that display the largest impact on the prediction task we find an increased number of cliques in enzymatic proteins, reflecting the structurally stable modules necessary for catalysis (Figure S1). Our deep-dive analysis also reveals that, despite being derived from seemingly different areas of graph theory, many top features are highly correlated (Figure S2) and provide similar predictive power. This observation reflects the mathematical relationships, sometimes not explicitly recognized, between many graph features, which can still afford complementary descriptions of the data. These results highlight the need to consider a broad range of features in the analysis of networks, rather than relying on a narrow set of properties.

Unsupervised clustering: Mapping morphological neuron types onto cell types based on network features

To go beyond supervised classification of benchmark datasets, we used HCGA to analyze a dataset of 444 neuronal cells from 6 layers (L1 to L6) in the rat somatosensory cortex belonging to 24 different morphological types (m-types).[39,40] The neurons are also labeled independently as belonging to three different cell types: pyramidal neurons, interneurons, and Martinotti cells.

We represent the morphology of each neuron as a rooted tree, with the soma as the root and basal dendrites and axons as branches extending from the soma (Figure 2A(i)). A graph is then constructed for each neuron by assigning a node to each section between two branching points of the morphology. Each node is also assigned features: path length, mean diameter of each branch, and an annotation to differentiate the soma from other nodes. See "Neuronal morphology dataset" in the experimental procedures for more details.

Using HCGA, we extracted 2,112 features from the graph of each neuron. Our aim is to compare the similarities between neuron morphological types and their correspondence with cell types. To do so, we construct a similarity matrix between m-types by computing the mean 10-fold classification accuracy between each pair of m-types: low classification accuracy between m-types indicates high pairwise similarity (Figure 2A(ii) and "Neuronal morphology dataset" in the experimental procedures). Applying a simple hierarchical clustering with Ward's linkage[41] to this similarity matrix, we find clusters of m-types that map well to cell types. In particular, a robust clustering into four clusters naturally recovers biologically meaningful groupings of m-types into cell types: pyramidal, Martinotti, other interneurons, as well as a group of the three most common pyramidal m-types (L5_TTPC1, L5_TTPC2, and L23_PC). At a finer resolution, we note that clusters consist of pairs or triplets of the same m-types belonging to different cortical layers. A deeper analysis of feature importance in this dataset may reveal further morphological characteristics that define cell types across layers, but such detailed study is beyond the scope of this initial illustrative analysis.

Regressing graph features of helicene structures against their electronic transport

As a second example of a task in scientific discovery, we apply HCGA to predict the electronic properties of helicenes from their structure. Helicenes are graphene-type spiral molecules with promising optical and electronic applications due to their axial chirality.[42] We use a recently curated dataset composed of 1,344 helicene dimers (i.e., pairs of helicenes in the same translational-motif orientation),[6] see Figure 2B(i). For each helicene dimer, we create a simple molecular graph representing atoms and their van der Waals interactions: an edge is present if the

Table 2. Classification results on social benchmark datasets obtained by HCGA using the XGBoost classifier and compared against other methodologies

	Datasets (social networks–benchmarks)					
Method	Collab	IMDB-B	IMDB-M	REDDIT-B	REDDIT-5K	REDDIT-12K
Multi-Hop (RF)[34]	78.2 ± 1.5	71.6 ± 4.4	45.2 ± 3.5	88.9 ± 2.2	51.3 ± 1.9	43.5 ± 1.0
DGCNN[35]	71.2 ± 1.9	69.2 ± 3.0	45.6 ± 3.4	87.8 ± 2.5	49.2 ± 1.2	–
GIN[36]	75.6 ± 2.3	71.2 ± 3.9	48.5 ± 3.3	89.9 ± 1.9	56.1 ± 1.7	–
ECC[37]	–	67.7 ± 2.8	43.5 ± 3.1	–	–	–
DiffPool[38]	68.9 ± 2.0	68.4 ± 3.3	45.6 ± 3.4	89.1 ± 1.6	53.8 ± 1.4	–
HCGA	82.9 ± 1.5	74.2 ± 4.2	47.3 ± 3.5	91.5 ± 2.1	54.9 ± 2.1	49.6 ± 0.8
HCGA*	83.0 ± 1.6	75.5 ± 4.1	51.0 ± 4.5	93.5 ± 1.9	57.9 ± 1.5	–

The classification accuracies for REDDIT-12K are not available for most methods due to long computation times; however, we retain this example to show that HCGA is also capable of analyzing large datasets. Results are presented as mean ± standard deviation.

separation between two atoms is lower than the sum of their van der Waals radii (plus a buffer of 2 Å), and the edge weight is given by the inverse distance (more details are given in "Organic semiconductors dataset" in the experimental procedures). The atom types are included as one-hot encoded node features. For each dimer, we aimed to predict the electronic transfer integral *J*, or *J*-coupling, which describes the ease with which a charge carrier (electron or hole) can hop from one molecule to another. The *J*-couplings for each helicene dimer were previously calculated using a hybrid DFT molecular pair calculation and the projective method[43] (see "Organic semiconductors dataset" in the experimental procedures). The *J*-coupling depends on various factors, such as the frontier orbitals of the molecule, the molecular packing (i.e., the relative orientations and distances between molecules),[44] as well as molecular vibrations.

Since the *J*-coupling is a continuous variable, we treat this problem as a regression task. We used HCGA to extract 2,531 features from each molecular graph and regressed them against the logarithm of the transfer integral $log(J)$. Using the full set of features, we achieve a mean absolute error (MAE) of 0.355 ± 0.022 on our evaluation set. To put this into perspective, it is generally expected that the shorter the distance between molecules the higher the charge transfer between them. However, if we use the distance between molecules in the dimer as the only feature, we obtain an MAE of 1.58 ± 0.08, an almost 5-fold decrease in accuracy with respect to the regression against graph features, thus indicating that orientation, atom types, and other structural features of the molecular graphs play a critical role in charge transfer. Furthermore, if we remove the edge-filtering step and consider a fully connected weighted graph based on edge distances (under the hypothesis that the transfer integral should be a smooth function of the dimer configuration) our results slightly worsen, but not significantly (MAE = 0.362 ± 0.030). Graph-theoretical algorithms do not always incorporate edge weights and thus these features would be redundant in a fully connected weighted graph, alternatively longer-range interactions may not be necessarily physically meaningful. In addition, if we remove edge weights of the filtered network then our accuracy decreases further (MAE = 0.375 ± 0.028), highlighting the importance of edge weights in improving accuracy, but also showing that edge weights are not critical in attaining a predictive model.

To facilitate our understanding of the factors affecting electron transfer, we reduced the feature set to the top 10 uncorrelated features ($\leq$0.7 correlation, see "Reduction of the feature set" in the experimental procedures), and we achieved an MAE of 0.361 ± 0.02, indicating that the majority of important information is captured by a small set of features (Figure 2B(ii)).

We used SHAP,[30] a game-theoretic framework that computes the contribution of each feature to the prediction of each sample (Figure 2B(ii)). Features with large absolute SHAP values have an overall large impact on prediction, while the sign of the SHAP values indicates a positive or negative effect of that feature on the prediction of the sample. The sum of absolute SHAP values across all samples for each feature allows us to assess their relative impact on the prediction task.[30] As seen in Figure 2B(ii), the top feature is the variation in the edge betweenness centrality (VEBC) of the molecular graph, and Figure 2B(iii) shows that an increase in VEBC correlates strongly with a decrease in *J*-coupling. Indeed, regressing against VEBC alone already gives an MAE of 1.032 ± 0.08, which is substantially lower than regressing against the distance between the molecules in the dimer. Edge betweenness is a graph property that measures the importance of an edge for mediating the shortest paths between nodes; hence, a low variation in edge betweenness suggests a balanced spread of communication in the dimer, with no single atomic interaction acting as a critical funnel for the shortest paths. Another interesting graph feature in the top set is the maximum rich club coefficient (MRCC). As shown in Figure 2B(iii), a large MRCC is linked to large *J*-coupling. A large MRCC means that the hubs are well connected, and that the global connectivity is resilient to hub removal. In our molecular graphs, such robustness to the removal of particular atoms indicates the existence of alternative paths in the structure that may facilitate charge transfer between the molecules in the dimer. Further examination of the molecular features related to high charge transfer in helicenes lie beyond the aims of this work. However, further research could pose an associated classification task to identity graph features that lead to very high or very low *J*-couplings with the aim to guide the design of more efficient helicene compounds.

DISCUSSION

We have introduced HCGA, a high-throughput computational graph feature analysis package that leverages the giant corpus of graph theory literature. The software package distills a large

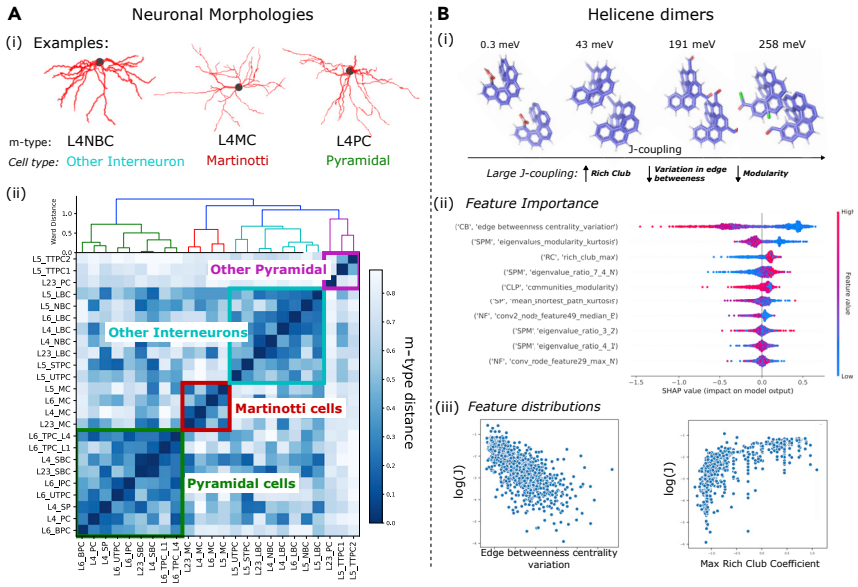


Figure 2. HCGA recovers biological clusters of neuronal morphologies and uncovers interpretable structural properties for effective electronic coupling in semiconductor crystals

(A) Neuronal morphologies. (i) Three examples of neuron morphologies for three different cell types which can be further distinguished by the cortical layer from which they were extracted (in this case from L4). (ii) The inverse classification accuracies between m-type pairs using the HCGA features was used to produce a similarity matrix. Clustering of this similarity matrix recovers the three main morphological cell types (pyramidal, Martinotti, and interneurons) and reveals a new cluster consisting of the most common pyramidal cells. Finer clusters group pairs or triplets of the same m-types from different layers.

(B) Helicene dimers. (i) Four examples of helicene dimers at different magnitudes of J-coupling, colored by atom type. (ii) The SHAP value for each individual sample for the top 10 features listed in descending order. Each sample is colored by their relative feature value normalized between 0 and 1. The sum of absolute SHAP values for individual samples defines the total feature importance SHAP value. (iii) The relationship between log(J) and the top two features are illustrated in a scatter diagram; we notice a strong negative trend with the variation of edge betweenness centrality and a positive non-linear trend with the maximum rich club coefficient.

set of graph-theoretical algorithms making them easily accessible and simple to interpret for a given research problem. Beyond the massive feature extraction, HCGA includes a suite of statistical prediction tools for classification and regression to help researchers analyze their datasets. The highly comparative nature of HCGA provides a framework to identify individual features that play an important role in prediction and reveal scientific insights into their systems.

The use of network data in statistical learning is a current area of intense work.[45] For instance, researchers have turned to graph neural networks to learn structural features through message-passing and non-linear interactions.[46,47] However, such methods lack the interpretability necessary to facilitate discovery science[48] and can be fraught with inaccuracies and inflated performance due to over-engineering.[33] The area closest in essence to HCGA is that of graph embeddings, in which the graph is reduced to a vector that aims to effectively incorporate the structural features.[49] However, the inherent choice of network properties that provide a "good" vector representation of the graph is not known and may differ between scientific domains and the type of statistical learning task. HCGA thus circumvents this critical step in the embedding process through indiscriminate massive feature extraction.

To apply HCGA to a given research problem, a graph dataset must be first constructed which is not always a trivial task. For a social network the structure can be simply built using the exact relationships between individuals; however, as we saw in the previous section with the helicene molecules, there can be various different ways to construct the graphs with differing performances. While the graph construction is outside the scope of this paper, there are a number of studies that examine the optimal construction of graphs for graph learning.[50–54] We advise researchers to think carefully about the meaningful construction of graphs in respect to their data and research.

We have illustrated the use of HCGA on a variety of examples drawn from different scientific domains performing different learning tasks (supervised classification, unsupervised clustering, regression). The framework is highly predictive and provides interpretable insights. HCGA has general utility and can be applied to a variety of fields where network datasets constitute the core of experimental outcomes, including applications to functional or structural connectivity networks in brains, revealing properties of computer networks that make them weak to outside attacks, the analysis of the structure of ecological networks and their fragility, the interdependencies in social and economic networks, and many other problems across scientific domains.

EXPERIMENTAL PROCEDURES

Resource availability

Lead contact

Further information and requests for resources should be directed to and will be fulfilled by the lead contact, Mauricio Barahona (m.barahona@imperial.ac.uk).

Data and code availability

The authors declare that the code supporting the findings of this study are available within the paper and its supplemental information files. The code is shared under the GNU General Public License v.3.0.

Reagent or resource	Source	Identifier
Code repository	this manuscript	https://github.com/barahona-research-group/hcga
Code documentation	this manuscript	https://barahona-research-group.github.io/hcga/

(*Continued on next page*)

Continued

Reagent or resource	Source	Identifier
Code examples	this manuscript	https://github.com/barahona-research-group/hcga/tree/master/examples
Helicenes dataset	this manuscript	https://doi.org/10.14469/hpc/7858
Benchmark datasets	TU Dortmund	https://ls11-www.cs.tu-dortmund.de/staff/morris/graphkerneldatasets
Neuronal morphologies dataset	epfl microcircuits	http://microcircuits.epfl.ch/#/article/article_3_mph

Materials availability

The authors declare that no materials were generated or used during this study.

Benchmark datasets

Table S2 provides a high-level description of each dataset. The benchmark datasets were taken from https://ls11-www.cs.tu-dortmund.de/staff/morris/graphkerneldatasets.

Neuronal morphology dataset

The neuron dataset is constructed from the morphological reconstructions used in the Blue Brain Project, freely available at http://microcircuits.epfl.ch/#/article/article_3_mph. A neuron contains a soma in its center, from which several types of branches emerge: axonal branches are carrying electrical signals away from the cells and basal or apical dendrites receive electrical signals from pre-synaptic cells, which will eventually trigger action potentials (or spikes) in the axon initial segment, near the soma. The shape of neurons is thus important in the electrical properties of the cells as well as how and where it connects to pre- and post-synaptic cells in a neuronal circuit. For this reason, an accurate and meaningful classification of shape (m-types) and electrical properties (e-types) of neurons is an active research topic in neuroscience; see, for example, Gouwens et al.[55] Here, we will only consider morphological types, represented as graphs.

Most commonly, morphologies are classified by the reconstructors in morphological types, which depend on the cortical layer (from 1 to 6) from which they were extracted and a subjective interpretation of the shape of the cell. There are two broad classes of cells, pyramidal cells or excitatory cells, which contain apical trees and interneurons, or inhibitory cells, which only have basal dendrites. We selected a subset of neuronal morphologies such that each m-type consists of at least 10 associated morphologies, consisting of 444 morphologies classified in 24 different m-types across 6 layers of the rat somatosensory cortex.

For each cell, we constructed a graph representation by extracting the neuron connectivity between sections using the Python package MorphIO (https://github.com/BlueBrain/MorphIO), where a section is defined as a list of points along branches between two branching points. Each node of the connectivity graph represents a section, and an edge is present between two sections if they are connected by a branching point. This representation allows us to assign node attributes as the path length and the mean diameter of each section. The path length is defined from the three-dimensional location of the points along the section as the sum of the length of each interval. In addition, we added a node label to represent the soma, as $[1, 0]$ for the soma node, and $[0, 1]$ for all other nodes.

To compute the pairwise classification accuracy between each pair of m-types, we used a two-step procedure. We first used the entire feature set extracted with HCGA, secondly we chose a subset of the feature set containing the top 10 features that were less than 0.7 Pearson correlated between themselves. This drastically reduced set of features and increased the classification accuracy on each pair of m-types (likely by reducing over-fitting to a large feature set).

Given the matrix of pairwise classification accuracies, we then applied a sigmoid function $f(x) = 1/(1 + e^{-10(x-0.8)})$ to make low and high accuracies more similar while retaining intermediate accuracies. This step was implemented simply to enable a more robust clustering of the similarity matrix when using the hierarchical clustering algorithm. The groupings of m-types were robust across different parameters for feature extraction (e.g., using all features or choosing only highly interpretable features to produce the similarity matrix), suggesting that the results reflect real clusters of m-types and are not an artifact of our feature choice.

Organic semiconductors dataset

The organic semiconductors (helicenes) dataset was taken from a recent computational screening for high charge-carrier mobility study[6] and it is available at https://doi.org/10.14469/hpc/7858. In the original study, a total of 1,344 potential [6]helicene molecules were screened for their suitability as potential organic semiconductors. Here, we focus on the electronic transfer integral J (J-coupling). The electronic transfer integral J describes the hopping integral, i.e., the ease with which charge can hop from molecule A to molecule B of the same type. This integral is strongly dependent upon the frontier orbitals of molecules and the spatial arrangement and orientation of molecules A and B, and its prediction in new molecules is currently an open challenge. For further details on the dataset, see Schmidt et al.[6] and for access to the data see experimental procedures.

For each dimer (pair of molecules) at its minimum energy separation d_{min}, we computed the transfer integrals by projecting the computed orbitals of the dimer onto the unperturbed localized orbitals of the individual molecules.[43,56] All transfer integrals were computed with B3LYP/6-31G(d) using Gaussian16.[57]

In this dataset only one spatial arrangement was used, the so-called translational-dimer motif. The electronic transfer integral is believed to exponentially decay with intermolecular distance, therefore the minimum energy distance d_{min} at which the J was computed is used as a control with which to compare HCGA.

For graph construction, each node represents an atom and edges corresponded to atom interactions. To define atomistic interactions we used the relative spatial distances between them; if the separation of two atoms was lower than the sum of their van der Waals radii plus 2 Å then we built an edge between the two atoms. The weight of the edge was simply the inverse of the Euclidean distance between the two atoms (calculated given their xyz coordinates). Each node was also assigned features based on the atom type using a one-hot encoded vector.

Method details

Software

Our entire pipeline is self-contained within a Python environment and only requires the user to input their dataset in the appropriate format (format described in the section entitled "inputs").

The pipeline can be implemented via two routes. The first is to use a shell script that allows the continuous execution of the entire pipeline. To test the benchmarks, we have included a script that automatically downloads and pre-processes the necessary dataset. Python 3 scripts then execute the feature extraction which relies on the NumPy, Pandas, NetworkX, and SciPy environments. The post-processing and statistical analysis steps are implemented using the sklearn and SHAP environments.

To enable continued preservation of the Python software package we have designed a modular and robust framework. Features can be added and removed easily and external users are able to make pull-requests to the main repository on github. A series of examples are available and a short introductory tutorial are available in Video S1.

Run times

Computational efficiency is a key problem in deriving graph statistics, such as, for example, features based on k-components or node-connectivity may take on order of minutes to compute on a high-end computer. To mitigate these issues we offer the user two options; (1) a time-out option where each feature is given a time-limit (default 10 s) before not a number (NaN) is returned, and (2) a feature speed ranking ("slow," "medium," and "fast") allowing the computation of only a subset of faster features. Feature extraction for the benchmark datasets is normally completed on the order minutes to hours on a local computer. Most features are based on the Python package networkx, but faster implementations of some features may be used. To allow for the use of other network packages to extract features, the internal graph representation is generic, and only at the level of feature extraction is a specific representation used, such as, for example, with networkx graph.

Feature interpretability

A key aspect of HCGA is the feature interpretability that provides researchers key insights. However, even classical graph-theoretic measures can be difficult to interpret in respect to most systems. Therefore, we have manually assigned an interpretability ranking (from most interpretable 5 to least interpretable 1) to each feature allowing users to implement statistical analysis with a chosen level of interpretability.

Feature filtration and normalization

Depending on the dataset, some features may not be computable for particular graphs. These values need to be removed before statistical analyses. Any features with infinity or NaN values and features with zero variance across the dataset are removed from the feature matrix, resulting in a reduced feature matrix. The quantity of removed features is dependent on the dataset, and often small if the dataset is composed of relatively similar graphs. In some cases, particular graphs can result in the removal of a large set of features, therefore it is advised to explore anomalous samples. Finally, each remaining feature is individually standardized to have zero mean and a unit variance to guarantee stable convergence during statistical analysis.

Graph classification

Graph classification is the process of predicting the class for each graph by mapping a function from the feature matrix to a vector of discrete output variables. To use this functionality the user must have input a set of class labels, one for each sample. The HCGA package allows the user to use any classification algorithm; however, the default procedure (and the procedure used within this paper) is to use the XGBoost classification algorithm with default parameters.[58] XGBoost is a decision tree-based ensemble machine-learning algorithm that uses a gradient boosting framework, and is considered the optimal approach for small-to-medium structured/tabular data.

The classification procedure uses a 10-fold stratified cross-validation with no hyper-parameter optimization. Our choice to not use hyper-parameter optimization was to provide a use-case in which a non-machine learning expert was using our software package. With careful tuning, a researcher will be capable of improving their results. Quoted results in this paper are the average across the 10-fold; this was used to prevent over-fitting leading to optimistic performance estimates.

Graph regression

Graph regression is the process of predicting a continuous variable for each graph by mapping a function from the feature matrix to a vector of continuous output variables. To use this functionality the user must have input a continuous output variable for each sample. The HCGA package allows the user to use any regression algorithm; however, the default procedure (and the procedure used within this paper) is to use the XGBoost regressor algorithm with default parameters.[58]

Similarly to classification, no hyper-parameter optimization is implemented and the results are averaged across a 10-fold stratified cross-validation.

Reduction of the feature set

Due to the large feature set, the training set may be overfitted, resulting in a decrease in accuracy on the test set. To remedy to this problem, we implemented a simple procedure for reducing the size of the feature set. From the first classification/regression step with all the features, we computed the SHAP importance value of each feature as well as the Pearson correlations between all of them. We then chose the n most important features that are correlated with a value less than c. These two numbers, n and c, are the only parameter the user can modify to reduce the feature set. We have set as default $n = 100$ and $c = 0.8$ for the results shown in Tables 1 and 2; however, these values can be optimized with a validation set (see starred HCGA results in Tables 1 and 2).

Feature importance

HCGA implements state-of-the-art methods for investigating the contribution or importance of individual features toward the classification or regression problem. Specifically, we compute top features using the SHAP framework, which computes optimal explanations based on game theory through local feature interaction effects.[30] We compute the SHAP values of every feature for each fold in our cross-validation procedure and return an average across the folds. We output a SHAP value for each feature given its ability to separate each class individually. We also average SHAP values for each class to return an overall feature importance measure across the entire set of classes.

Inputs

The benchmarks can be automatically downloaded without any need to provide custom input into HCGA. For custom data we have provided a number of example notebooks to aid the user in their application. We have built a generic Graph object which allows the user to pass their graph to HCGA and be automatically converted to the appropriate graph representation used by the feature extraction module. The user should provide a graph, any node features and a graph label (or continuous variable).

Outputs

The primary outputs from HCGA are the computed features for each input graph. The list of master operations are detailed in Table S1. The secondary outputs of HCGA are a result of the statistical analysis module. This includes a comma-separated value file, that details the importance of each feature (SHAP value) toward the statistical prediction task (classification or regression), and a results report which includes a series of plots to facilitate user insights into their data. The plots include:

1. *Bar plot detailing the mean absolute SHAP value for the top features.* A larger value indicates that the feature had a larger impact on the prediction task.
2. *A sample expanded feature summary.* Detailing the impact of each feature on individual samples.
3. *A heatmap of absolute correlation coefficient between features.* The heatmap is ordered by the feature type. Green dots along the diagonal indicate the top features.
4. *A heatmap of absolute correlation coefficient between features.* The heatmap is clustered according to Euclidean distance within the space of correlations (clustering applied using a Ward linkage algorithm). Green dots along the diagonal indicate the top features.
5. *Plots of individual features.* In classification tasks, these appear as violin plots for each class, and for regression tasks these appear as a scatter-plot of dependent versus independent variable.
6. *Feature summaries for each top feature.* A violin plot is displayed for each feature and representative sample networks are displayed alongside representing different points in the feature distribution.

SUPPLEMENTAL INFORMATION

Supplemental information can be found online at https://doi.org/10.1016/j.patter.2021.100227.

ACKNOWLEDGMENTS

R.L.P., A.A., S.N.Y., and M.B. acknowledge funding through EPSRC award EP/N014529/1 supporting the EPSRC Centre for Mathematics of Precision Healthcare at Imperial. K.E.J. thanks the Royal Society for a University Research Fellowship and for an Enhancement Award 2017. We gratefully acknowledge advice and interpretation of results from Asher Mullokandov and Paul Expert. We also thank Prof Jenny Nelson for insightful discussions.

AUTHOR CONTRIBUTIONS

R.L.P., A.A., and M.B. conceived the framework. R.L.P., A.A., H.A.P., and N.R.B. designed the methodology and wrote the software package. R.L.P., A.A., and J.A.S. performed testing and benchmarking. J.A.S. and K.E.J. developed the semiconductor dataset and interpreted the results. R.L.P., A.A., J.A.S., S.N.Y., and M.B. wrote the manuscript.

DECLARATION OF INTERESTS

The authors declare no competing interests.

Received: October 17, 2020
Revised: February 2, 2021
Accepted: March 3, 2021
Published: April 2, 2021

REFERENCES

1. Newman, M. (2010). Networks: An Introduction (Oxford University Press).
2. Delvenne, J.-C., Yaliraki, S.N., and Barahona, M. (2010). Stability of graph communities across time scales. Proc. Natl. Acad. Sci. U S A *107*, 12755–12760, https://doi.org/10.1073/pnas.0903215107.
3. Delmotte, A., Tate, E., Yaliraki, S.N., and Barahona, M. (2011). Protein multi-scale organization through graph partitioning and robustness analysis: application to the myosin-myosin light chain interaction. Phys. Biol. *8*, 055010, https://doi.org/10.1088/1478-3975/8/5/055010.
4. Peach, R.L., Saman, D., Yaliraki, S.N., Klug, D.R., Ying, L., Willison, K.R., and Barahona, M. (2019). Unsupervised graph-based learning predicts mutations that alter protein dynamics. bioRxiv. https://doi.org/10.1101/847426.
5. Bullmore, E., and Sporns, O. (2009). Complex brain networks: graph theoretical analysis of structural and functional systems. Nat. Rev. Neurosci. *10*, 186–198, https://doi.org/10.1038/nrn2575.
6. Schmidt, J.A., Weatherby, J.A., Sugden, I., Santana-Bonilla, A., Salerno, F., Fuchter, M., Johnson, E., Nelson, J., and Jelfs, K. (2020). Computational screening of organic semiconductors: exploring side-group functionalisation and assembly to optimise charge transport in chiral molecules. chemrXiv. https://doi.org/10.26434/chemrxiv.12451943.v1.
7. Jia, J., Schaub, M.T., Segarra, S., and Benson, A.R. (2019). Graph-based Semi-Supervised & Active Learning for Edge Flows. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (Association for Computing Machinery), pp. 761–771.
8. Borgatti, S.P., Mehra, A., Brass, D.J., and Labianca, G. (2009). Network analysis in the social sciences. Science *323*, 892–895, https://doi.org/10.1126/science.1165821.
9. Beguerisse-Díaz, M., Garduño-Hernández, G., Vangelov, B., Yaliraki, S.N., and Barahona, M. (2014). Interest communities and flow roles in directed networks: the Twitter network of the UK riots. J. R. Soc. Interfaces *11*, 20140940, https://doi.org/10.1098/rsif.2014.0940.
10. Peach, R.L., Arnaudon, A., and Barahona, M. (2020). Semi-supervised classification on graphs using explicit diffusion dynamics. Found. Data Sci. *2*, 19, https://doi.org/10.3934/fods.2020002.
11. Borgatti, S.P., and Everett, M.G. (2006). A graph-theoretic perspective on centrality. Soc. Netw. *28*, 466–484, https://doi.org/10.1016/j.socnet.2005.11.005.
12. Arnaudon, A., Peach, R.L., and Barahona, M. (2020). Scale-dependent measure of network centrality from diffusion dynamics. Phys. Rev. Res. *2*, 033104, https://doi.org/10.1103/PhysRevResearch.2.033104.
13. Newman, M.E. (2003). Mixing patterns in networks. Phys. Rev. E *67*, 026126, https://doi.org/10.1103/PhysRevE.67.026126.
14. Foster, J.G., Foster, D.V., Grassberger, P., and Paczuski, M. (2010). Edge direction and the structure of networks. Proc. Natl. Acad. Sci. U S A *107*, 10815–10820, https://doi.org/10.1073/pnas.0912671107.
15. Johnson, D.B. (1977). Efficient algorithms for shortest paths in sparse networks. J. ACM *24*, 1–13, https://doi.org/10.1145/321992.321993.
16. Klein, D.J., and Randić, M. (1993). Resistance distance. J. Math. Chem. *12*, 81–95, https://doi.org/10.1007/BF01164627.
17. Brockmann, D., and Helbing, D. (2013). The hidden geometry of complex, network-driven contagion phenomena. Science *342*, 1337–1342, https://doi.org/10.1126/science.1245200.
18. Flandrin, E., Li, H., Marczyk, A., and Woźniak, M. (2007). A generalization of Dirac's theorem on cycles through k vertices in k-connected graphs. Discrete Math. *307*, 878–884, https://doi.org/10.1016/j.disc.2005.11.052.
19. Fortunato, S. (2010). Community detection in graphs. Phys. Rep. *486*, 75–174, https://doi.org/10.1016/j.physrep.2009.11.002.
20. Lambiotte, R., Delvenne, J.-C., Barahona, M., and Jul. (2014). Random walks, Markov processes and the multiscale modular organization of complex networks. IEEE Trans. Netw. Sci. Eng. *1*, 76–90, https://doi.org/10.1109/tnse.2015.2391998.
21. Monti, F., Frasca, F., Eynard, D., Mannion, D., and Bronstein, M.M. (2019). Fake news detection on social media using geometric deep learning. arXiv https://arxiv.org/abs/1902.06673.
22. National Research Council (2007). Toxicity Testing in the 21st Century: A Vision and a Strategy (National Academies Press).
23. Agarwal, S. (2012). Networks in Nature: Dynamics, Evolution, and Modularity, PhD thesis (University of Oxford).
24. Zhou, Z., Chen, X., Zhang, Y., Hu, D., Qiao, L., Yu, R., Yap, P.-T., Pan, G., Zhang, H., and Shen, D. (2020). A toolbox for brain network construction and classification (BrainNetClass). Hum. Brain Mapp. *41*, 2808–2826, https://doi.org/10.1002/hbm.24979.
25. Barnett, I., Malik, N., Kuijjer, M.L., Mucha, P.J., and Onnela, J.-P. (2019). EndNote: feature-based classification of networks. Netw. Sci. *7*, 438–444, https://doi.org/10.1017/nws.2019.21.
26. Fulcher, B.D., Little, M.A., and Jones, N.S. (2013). Highly comparative time-series analysis: the empirical structure of time series and their methods. J. R. Soc. Interf. *10*, 83, https://doi.org/10.1098/rsif.2013.0048.
27. Fulcher, B.D., and Jones, N.S. (2014). Highly comparative feature-based time-series classification. IEEE Trans. Knowledge Data Eng. *26*, 3026–3037, https://doi.org/10.1109/TKDE.2014.2316504.
28. Fulcher, B.D., and Jones, N.S. (2017). hctsa: a computational framework for automated time-series phenotyping using massive feature extraction. Cell Syst. *5*, 527–531, https://doi.org/10.1016/j.cels.2017.10.001.
29. Lubba, C.H., Sethi, S.S., Knaute, P., Schultz, S.R., Fulcher, B.D., and Jones, N.S. (2019). catch22: CAnonical Time-series CHaracteristics. Data Min. Knowledg. Discov. *33*, 1821–1852, https://doi.org/10.1007/s10618-019-00647-x.
30. Lundberg, S.M., Erion, G., Chen, H., DeGrave, A., Prutkin, J.M., Nair, B., Katz, R., Himmelfarb, J., Bansal, N., and Lee, S.-I. (2020). From local explanations to global understanding with explainable AI for trees. Nat. Machine Intellig. *2*, 2522–5839, https://doi.org/10.1038/s42256-019-0138-9.
31. Dobson, P.D., and Doig, A.J. (2003). Distinguishing enzyme structures from non-enzymes without alignments. J. Mol. Biol. *330*, 771–783, https://doi.org/10.1016/S0022-2836(03)00628-4.
32. Schomburg, I., Chang, A., Ebeling, C., Gremse, M., Heldt, C., Huhn, G., and Schomburg, D. (2004). BRENDA, the enzyme database: updates and major new developments. Nucleic Acids Res. *32*, D431–D433, https://doi.org/10.1093/nar/gkh081.
33. Errica, F., Podda, M., Bacciu, D., and Micheli, A. (2020) A fair comparison of graph neural networks for graph classification. In 8th International Conference on Learning Representations (ICLR) .
34. Gutiérrez-Gómez, L., and Delvenne, J.-C. (2018). Multi-hop assortativities for network classification. J. Complex Networks *7*, 603–622, https://doi.org/10.1093/comnet/cny034.
35. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., and Solomon, J.M. (2019). Dynamic graph CNN for learning on point clouds. Acm Trans. Graphics (Tog) *38*, 1–12, https://doi.org/10.1145/3326362.
36. Xu, K., Hu, W., Leskovec, J., and Jegelka, S. (2018). How powerful are graph neural networks? arXiv https://arxiv.org/abs/1810.00826.
37. Simonovsky, M. and Komodakis, N. (2017) Dynamic edge-conditioned filters in convolutional neural networks on graphs. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (IEEE), pp. 3693–3702.
38. Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., and Leskovec, J. (2018). Hierarchical graph representation learning with differentiable pooling. In Advances in Neural Information Processing Systems, S. Bengio, ed. (Neural Information Processing Systems Foundation, Inc. (NIPS)), pp. 4800–4810.
39. Markram, H., Muller, E., Ramaswamy, S., Reimann, M.W., Abdellah, M., Sanchez, C.A., Ailamaki, A., Alonso-Nanclares, L., Antille, N., Arsever, S., et al. (2015). Reconstruction and simulation of neocortical microcircuitry. Cell *163*, 456–492, https://doi.org/10.1016/j.cell.2015.09.029.

40. Ramaswamy, S., Courcol, J.D., Abdellah, M., Adaszewski, S.R., Antille, N., Arsever, S., Atenekeng, G., Bilgili, A., Brukau, Y., Chalimourda, A., et al. (2015). The neocortical microcircuit collaboration portal: a resource for rat somatosensory cortex. Front. Neural Circ. *9*, 44, https://doi.org/10.3389/fncir.2015.00044.

41. Ward, J.H., Jr. (1963). Hierarchical grouping to optimize an objective function. J. Am. Stat. Assoc. *58*, 236–244, https://doi.org/10.1080/01621459.1963.10500845.

42. Yang, Y., Da Costa, R.C., Fuchter, M.J., and Campbell, A.J. (2013). Circularly polarized light detection by a chiral organic semiconductor transistor. Nat. Photon. *7*, 634–638, https://doi.org/10.1038/nphoton.2013.176.

43. Rice, B., LeBlanc, L.M., Otero-de-la Roza, A., Fuchter, M.J., Johnson, E.R., Nelson, J., and Jelfs, K.E. (2018). A computational exploration of the crystal energy and charge-carrier mobility landscapes of the chiral [6] helicene molecule. Nanoscale *10*, 1865–1876, https://doi.org/10.1039/C7NR08890F.

44. Coropceanu, V., Cornil, J., da Silva Filho, D.A., Olivier, Y., Silbey, R., and Brédas, J.-L. (2007). Charge transport in organic semiconductors. Chem. Rev. *107*, 926–952, https://doi.org/10.1021/cr050140x.

45. Bronstein, M.M., Bruna, J., LeCun, Y., Szlam, A., and Vandergheynst, P. (2017). Geometric deep learning: going beyond Euclidean data. IEEE Signal. Process. Mag. *34*, 18–42, https://doi.org/10.1109/MSP.2017.2693418.

46. Henaff, M., Bruna, J., and LeCun, Y. (2015). Deep convolutional networks on graph-structured data. arXiv https://arxiv.org/abs/1506.05163.

47. Zhou, J., Cui, G., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., and Sun, M. (2018). Graph neural networks: a review of methods and applications. arXiv https://arxiv.org/abs/1812.08434.

48. Ying, Z., Bourgeois, D., You, J., Zitnik, M., and Leskovec, J. (2019). Gnnexplainer: generating explanations for graph neural networks. In Advances in Neural Information Processing Systems, S. Bengio, ed. (Neural Information Processing Systems Foundation, Inc. (NIPS)), pp. 9244–9255.

49. Goyal, P., and Ferrara, E. (2018). Graph embedding techniques, applications, and performance: a survey. Knowledg. Based Syst. *151*, 78–94, https://doi.org/10.1016/j.knosys.2018.03.022.

50. Chen, Y., Wu, L., and Zaki, M.J. (2020). Iterative deep graph learning for graph neural networks: better and robust node embeddings. arXiv https://arxiv.org/abs/2006.13009.

51. Kang, Z., Pan, H., Hoi, S.C., and Xu, Z. (2019). Robust graph learning from noisy data. IEEE Trans. Cybernetics *50*, 1833–1843.

52. Halcrow, J., Mosoi, A., Ruth, S., and Perozzi, B. (2020) Grale: designing networks for graph learning. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (Association for Computing Machinery), pp. 2523–2532.

53. Kang, Z., Peng, C., Cheng, Q., Liu, X., Peng, X., Xu, Z., and Tian, L. (2021). Structured graph learning for clustering and semi-supervised classification. Pattern Recogn. *110*, 107627, https://doi.org/10.1016/j.patcog.2020.107627.

54. Qian, Y., Expert, P., Panzarasa, P., and Barahona, M. (2020). Geometric graphs from data to aid classification tasks with graph convolutional networks. arXiv https://arxiv.org/abs/2005.04081.

55. Gouwens, N.W., Sorensen, S.A., Berg, J., Lee, C., Jarsky, T., Ting, J., Sunkin, S.M., Feng, D., Anastassiou, C.A., Barkan, E., et al. (2019). Classification of electrophysiological and morphological neuron types in the mouse visual cortex. Nat. Neurosci. *22*, 1182–1195, https://doi.org/10.1038/s41593-019-0417-0.

56. Xie, X., Santana-Bonilla, A., and Troisi, A. (2018). Nonlocal electron-phonon coupling in prototypical molecular semiconductors from first principles. J. Chem. Theor. Comput. *14*, 3752–3762, https://doi.org/10.1021/acs.jctc.8b00235.

57. Frisch, M.J., Trucks, G.W., Schlegel, H.B., Scuseria, G.E., Robb, M.A., Cheeseman, J.R., Scalmani, G., Barone, V., Petersson, G.A., Nakatsuji, H., et al. (2016). Gaussian16 Revision C.01 (Gaussian Inc).

58. Chen, T. and Guestrin, C. (2016). XGBoost: a scalable tree boosting system. In Krishnapuram, B., Shah, M., Smola, A. J., Aggarwal, C. C., Shen, D., and Rastogi, R., (eds.), Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (Association of Computing Machinery), pp. 785–794.