Software Impacts 9 (2021) 100095

Contents lists available at ScienceDirect

Software Impacts

journal homepage: www.journals.elsevier.com/software-impacts

Original software publication

Graph neural networks in TensorFlow-Keras with RaggedTensor representation (kgcnn)

Patrick Reiser [a,b,*], André Eberhard [b], Pascal Friederich [a,b,**]

[a] Institute of Nanotechnology, Karlsruhe Institute of Technology (KIT), Hermann-von-Helmholtz-Platz 1, 76344 Eggenstein-Leopoldshafen, Germany
[b] Institute of Theoretical Informatics, Karlsruhe Institute of Technology (KIT), Am Fasanengarten 5, 76131 Karlsruhe, Germany

ARTICLE INFO

Keywords:
Graph
Neural network
Machine learning

ABSTRACT

Graph neural networks are a versatile machine learning architecture that received a lot of attention recently due to its wide range of applications. In this technical report, we present an implementation of graph convolution and graph pooling layers for TensorFlow-Keras models, which allows a seamless and flexible integration into standard Keras layers to set up graph models in a functional way. We developed the **K**eras **G**raph **C**onvolutional **N**eural **N**etwork Python package ***kgcnn*** based on TensorFlow-Keras which focus on a transparent tensor structure passed between layers and an ease-of-use mindset.

Code metadata

Current code version	v1.0.0
Permanent link to code/repository used for this code version	https://github.com/SoftwareImpacts/SIMPAC-2021-57
Permanent link to reproducible capsule	https://codeocean.com/capsule/8499626/tree/v1
Legal code licence	MIT Licence
Code versioning system used	Git
Software code languages, tools, and services used	Python 3
Compilation requirements, operating environments & dependencies	TensorFlow ≥2.4
If available link to developer documentation/manual	https://kgcnn.readthedocs.io/en/latest/index.html
Support email for questions	patrick.reiser@kit.edu

Software metadata

Current software version	v1.0.0
Permanent link to executables of this version	https://pypi.org/project/kgcnn/
Permanent link to reproducible capsule	https://codeocean.com/capsule/8499626/tree/v1
Legal software licence	MIT Licence
Computing platforms/Operating systems	Linux, Microsoft Windows, Mac OS
Installation requirements & dependencies	TensorFlow ≥2.4
If available, link to user manual - if formally published include a reference to the publication in the reference list	–
Support email for questions	patrick.reiser@kit.edu

The code (and data) in this article has been certified as Reproducible by Code Ocean: (https://codeocean.com/). More information on the Reproducibility Badge Initiative is available at https://www.elsevier.com/physical-sciences-and-engineering/computer-science/journals.

* Corresponding author at: Institute of Nanotechnology, Karlsruhe Institute of Technology (KIT), Hermann-von-Helmholtz-Platz 1, 76344 Eggenstein-Leopoldshafen, Germany.
** Corresponding author at: Institute of Theoretical Informatics, Karlsruhe Institute of Technology (KIT), Am Fasanengarten 5, 76131 Karlsruhe, Germany.
E-mail addresses: patrick.reiser@kit.edu (P. Reiser), pascal.friederich@kit.edu (P. Friederich).

https://doi.org/10.1016/j.simpa.2021.100095
Received 25 May 2021; Received in revised form 1 June 2021; Accepted 2 June 2021

1. Introduction

Graph neural networks (GNNs) are a natural extension of common neural network architectures such as convolutional neural networks (CNN) [1–3] for image classification to graph structured data [4]. For example, recurrent [5,6], convolutional [4,7–9] and spatial–temporal [10] graph neural networks as well as graph autoencoders [11,12] and graph transformer models [13,14] have been reported in literature. A graph $G = (V, E)$ is defined as a set of vertices or nodes $v_i \in V$ and edges $e_{ij} = (v_i, v_j) \in E$ connecting two nodes. There are already comprehensive and extensive review articles for graph neural networks, which summarize and categorize relevant literature on graph learning [15,16]. The most frequent applications of GNNs include node classification or graph embedding tasks. While node classification is a common task for very large graphs such as citation networks [12] or social graphs [17], graph embeddings learn a representation of smaller graphs such as molecules [7,18] or text classifications [19,20]. Early work on applying neural networks to graphs [21] shaped the notion of graph neural networks and was further elaborated on by propagating information iteratively through the graph [5,18,22,23]. Graph networks may operate on the graph's spectrum [24,25] or directly on its structure [26]. One of the most prominent graph convolutional neural network (GCN) introduced by Kipf et al. [4] stacks multiple convolutional and pooling layers for deep learning to generate a high-level node representation from which both a local node and global graph classification can be obtained. Most GCNs can be considered as message passing networks [27,28], a class of graph networks in which information is propagated along edges between neighbouring nodes. In each update step t, the central node's hidden representation h_v is convolved with its neighbourhood given by:

$$h_v^{t+1} = U_t(h_v^t, m_v^{t+1}), \quad (1)$$

where m_v^t denotes the aggregated message and U_t the update function. The message to update is usually acquired from summing message functions M_t from the neighbourhood $N(v) = \{u \in V \,|\, (u, v) \in E\}$ of node v:

$$m_v^{t+1} = \sum_{w \in N(v)} M_t(h_v^t, h_w^t, e_{vw}). \quad (2)$$

More complex aggregation schemes are of course possible [29]. There is a large variety in convolution operators, which can be spectral-based [30] or spatial-based involving direct neighbours or a path of connected nodes to walk and collect information from [14,28]. Moreover, the message and update functions can be built from recurrent networks [31], multi-layer perceptrons (MLP) [32] or attention heads [33] which are complemented by a set of possible aggregation or pooling operations. Aggregation is usually done by a simple average of node representations or by a more refined set2set encoder part [34] as proposed by Gilmer et al. [27]. A reduction of nodes in the graph is achieved by pooling similar to CNNs but which is much more challenging on arbitrarily structured data. Examples of possibly differentiable and learnable pooling filters introduced in literature are DiffPool [35], EdgePool [36], gPool [37] HGP-SL,[35], SAGPool [38], iPool [39], EigenPool [40] and graph based clustering methods such as the Graclus algorithm [25,41–43].

In order to utilize the full scope of different graph operations for setting up a custom GNN model, a modular framework of convolution and pooling layers is necessary. We briefly summarize and discuss existing graph libraries and their code coverage. Then, a short overview of representing graphs in tensor form is given. Finally, we introduce our graph package kgcnn for TensorFlow 2's Keras API [44–46], which seamlessly integrates graph layers into the Keras [47] environment.

Table 1
Mean absolute validation error for single training on QM9 dataset for targets such as molecular orbital energies (HOMO, LUMO) and the energy gap E_G in eV using popular GNN architectures implemented in kgcnn. No hyperparameter optimization or feature engineering was performed.

Model	HOMO [eV]	HOMO [eV]	E_G [eV]
MPNN [27]	0.061	0.047	0.083
Schnet [7]	0.044	0.038	0.067
MegNet [32]	0.045	0.037	0.066

1.1. Graph libraries

Since graph neural networks require modified convolution and pooling operators, many Python packages for deep learning have emerged for either TensorFlow [44,45] or PyTorch [48] to work with graphs. We try to summarize the most notable ones without any claim that this list is complete.

- *PyTorch Geometric* [49]. A PyTorch based graph library which is probably the largest and most used graph learning Python package up to date. It implements a huge variety of different graph models and uses a disjoint graph representation to deal with batched graphs (graph representations are discussed in the Supplementary Information).
- *Deep Graph Library (DGL)* [50]. A graph model library with a flexible backend and a performance optimized implementation. It has its own graph data class with many loading options. Moreover, variants such as generative graph models [51], Capsule [52] and transformers [53] are included.
- *Spektral* [54]. A Keras [47] implementation of graph convolutional networks. Originally restricted to spectral graph filters [30], it now includes spatial convolution and pooling operations. The graph representation is made flexible by different graph modes detected by each layer.
- *StellarGraph* [55]. A Keras [47] implementation that implements a set of convolution layers and a few pooling layers plus a custom graph data format.

With PyTorch Geometric and DGL there are already large graph libraries with a lot of contributors from both academics and industry. The focus of the graph package presented here is on a neat integration of graphs into the TensorFlow-Keras framework in the most straightforward way. Thereby, we hope to provide Keras graph layers which can be quickly rearranged, changed and extended to build custom graph models with little effort. This implementation is focused on the new TensorFlow's RaggedTensor class which is most suited for flexible data structures such as graphs and natural language. The main field of applications targeted with this package is graph embedding tasks of e.g. molecules, materials and contextual or knowledge graph learning.

2. Description

A flexible and simple integration of graph operations into the TensorFlow-Keras framework can be achieved via ragged tensors. As mentioned above, ragged tensors are capable of efficiently representing graphs and have inherent access to various methods within TensorFlow (see Supplementary Information). For more sophisticated pooling algorithms which cannot be operated on batches, a parallelization of individual graphs within the batch could be achieved with the TensorFlow map functionality, although this is less efficient than vectorized operations and depends on implementation details.

We introduce a Python package *kgcnn*[1] that uses RaggedTensors, which are passed between graph layers for graph convolution and message passing models. We believe that the use of RaggedTensors

[1] https://github.com/aimat-lab/gcnn_keras.

allows a transparent and readable coding style, enables a seamless integration with many TensorFlow methods which are available for custom layers and makes it easy to debug code. We implemented a set of basic Keras layers for TensorFlow 2 from which many models reported in literature can be constructed. A simple code example is shown in Listing 1.

```python
import tensorflow.keras as ks
from kgcnn.layers.gather import GatherNodes
from kgcnn.layers.keras import Dense, Concatenate  # ragged support
from kgcnn.layers.pooling import PoolingLocalMessages, PoolingNodes

n = ks.layers.Input(shape=(None, 3), name='node_input',
 dtype="float32", ragged=True)
ei = ks.layers.Input(shape=(None, 2), name='edge_index_input',
  dtype="int64", ragged=True)

n_in_out = GatherNodes()([n, ei])
node_messages = Dense(10, activation='relu')(n_in_out)
node_updates = PoolingLocalMessages()([n, node_messages, ei])
n_node_updates = Concatenate(axis=-1)([n, node_updates])
n_embedd = Dense(1)(n_node_updates)
g_embedd = PoolingNodes()(n_embedd)

message_passing = ks.models.Model(inputs=[n, ei], outputs=g_embedd)
```

Listing 1: Python code example for setting up a simple one-layer message passing network in the functional API of TensorFlow-Keras for a graph embedding task. The size of the network and the architecture for this example is chosen arbitrarily.

The Python package implements the following architectures as examples: GCN [4], Interaction network [9], message passing [27], Schnet [7], MegNet [32], Unet [37], GNN Explainer [56], GraphSAGE [29], GAT [33] and DimeNet++[57]. The focus is set on graph embedding tasks, but also node and link classification tasks can be implemented using kgcnn. The models were tested with common bench-mark datasets such as Cora [58], MUTAG [59] and QM9[60]. Typical benchmark accuracies such as chemical accuracy on the QM9 dataset are achieved with the corresponding models implemented in kgcnn (see Table 1).

3. Software impact

Learning dedicated graph embeddings is of high interest in for example classification tasks and molecular property [61–64] and reaction predictions [65–68]. A differentiable and continuous graph convolution model could in principle replace empirical constructed force fields for molecular dynamics simulation (MD) when trained on quantum mechanical calculations [7,69–71]. This includes work of the authors [28,72,73]. In general, representation learning allows us to encode graph structured knowledge about interacting entities (i.e. nodes) and store it into a low-dimensional vector which can be further processed by machine learning (ML) models [74]. The graph models re-implemented in *kgcnn* are widely applicable and are well established in literature [7,27]. We believe that *kgcnn* can be used by researchers in other scientific fields to accommodate different graph learning tasks without detailed knowledge about graph representations or implementation details and by working with the well-known TensorFlow-Keras deep learning environment.

4. Limitations and future development

The current version of *kgcnn* is targeted for graph representation learning of small graphs such as molecules. Training on a single graph instance, e.g. a citation network which does not fit into memory is currently not explored. Although distribution strategies for training are already integrated into TensorFlow's high-level Keras API, they have not yet been tested with the models provided by *kgcnn*. We plan to continue to extend the *kgcnn* library to incorporate new models, in particular pooling methods [35] and to improve functionality. A future goal is to provide extremely large graph neural networks for distributed training on an exhaustive dataset for applications in chemistry and materials science.

5. Conclusion

In summary, we discussed a way to integrate graph convolution models into the TensorFlow-Keras deep learning framework. Main focus of our *kgcnn* package is the transparency of the tensor representation and the seamless integration with other Keras models.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

P.F. acknowledges funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 795206.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.simpa.2021.100095.

References

[1] Y. LeCun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, L.D. Jackel, Backpropagation applied to handwritten zip code recognition, Neural Comput. 1 (4) (1989) 541–551, http://dx.doi.org/10.1162/neco.1989.1.4.541, arXiv:https://direct.mit.edu/neco/article-pdf/1/4/541/811941/neco.1989.1.4.541.pdf.

[2] Y. LeCun, K. Kavukcuoglu, C. Farabet, Convolutional networks and applications in vision, in: Proceedings of 2010 IEEE International Symposium on Circuits and Systems, 2010, pp. 253–256, http://dx.doi.org/10.1109/ISCAS.2010.5537907.

[3] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, Commun. ACM 60 (6) (2017) 84–90, http://dx.doi.org/10.1145/3065386.

[4] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, 2017, ArXiv:1609.02907 [Cs, Stat], http://arxiv.org/abs/1609.02907, URL arXiv:1609.02907.

[5] F. Scarselli, M. Gori, A.C. Tsoi, M. Hagenbuchner, G. Monfardini, The graph neural network model, IEEE Trans. Neural Netw. 20 (1) (2009) 61–80, http://dx.doi.org/10.1109/TNN.2008.2005605, Conference Name: IEEE Transactions on Neural Networks.

[6] H. Dai, Z. Kozareva, B. Dai, A. Smola, L. Song, Learning steady-states of iterative algorithms over graphs, in: International Conference on Machine Learning, PMLR, (ISSN: 2640-3498) 2018, pp. 1106–1114, URL http://proceedings.mlr.press/v80/dai18a.html.

[7] K.T. Schütt, H.E. Sauceda, P.-J. Kindermans, A. Tkatchenko, K.-R. Müller, Schnet – a deep learning architecture for molecules and materials, J. Chem. Phys. 148 (24) (2018) 241722, http://dx.doi.org/10.1063/1.5019779, URL https://aip.scitation.org/doi/full/10.1063/1.5019779, Publisher: American Institute of Physics.

[8] M. Niepert, M. Ahmed, K. Kutzkov, Learning convolutional neural networks for graphs, 2016, ArXiv:1605.05273 [Cs, Stat], http://arxiv.org/abs/1605.05273, URL arXiv:1605.05273.

[9] P.W. Battaglia, R. Pascanu, M. Lai, D. Rezende, K. Kavukcuoglu, Interaction networks for learning about objects, relations and physics, 2016, ArXiv:1612.00222 [Cs], http://arxiv.org/abs/1612.00222, URL arXiv:1612.00222.

[10] B. Yu, H. Yin, Z. Zhu, Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting, in: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, 2018, pp. 3634–3640, http://dx.doi.org/10.24963/ijcai.2018/505, http://arxiv.org/abs/1709.04875, URL arXiv:1709.04875.

[11] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, C. Zhang, Adversarially regularized graph autoencoder for graph embedding, 2019, ArXiv:1802.04407 [Cs, Stat], http://arxiv.org/abs/1802.04407, URL arXiv:1802.04407.

[12] T.N. Kipf, M. Welling, Variational graph auto-encoders, 2016, ArXiv:1611.07308 [Cs, Stat], http://arxiv.org/abs/1611.07308, URL arXiv:1611.07308.

[13] S. Yao, T. Wang, X. Wan, Heterogeneous graph transformer for graph-to-sequence learning, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Online, 2020, pp. 7145–7154, http://dx.doi.org/10.18653/v1/2020.acl-main.640, URL https://www.aclweb.org/anthology/2020.acl-main.640.

[14] B. Chen, R. Barzilay, T. Jaakkola, Path-augmented graph transformer network, 2019, ArXiv:1905.12712 [Cs, Stat], http://arxiv.org/abs/1905.12712, URL arXiv:1905.12712.

[15] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, P.S. Yu, A comprehensive survey on graph neural networks, IEEE Trans. Neural Netw. Learn. Syst. (2020) 1–21, http://dx.doi.org/10.1109/TNNLS.2020.2978386, http://arxiv.org/abs/1901.00596, URL arXiv:1901.00596.

[16] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, M. Sun, Graph neural networks: A review of methods and applications, 2021, ArXiv:1812.08434 [Cs, Stat], http://arxiv.org/abs/1812.08434, arXiv:1812.08434.

[17] N. Benchettara, R. Kanawati, C. Rouveirol, Supervised machine learning applied to link prediction in bipartite social networks, in: 2010 International Conference on Advances in Social Networks Analysis and Mining, 2010, pp. 326–330, http://dx.doi.org/10.1109/ASONAM.2010.87.

[18] C. Merkwirth, T. Lengauer, Automatic generation of complementary descriptors with molecular graph networks, J. Chem. Inform. Model. 45 (5) (2005) 1159–1168, http://dx.doi.org/10.1021/ci049613b, Publisher: American Chemical Society.

[19] R. Angelova, G. Weikum, Graph-based text classification: learn from your neighbors, in: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, in: SIGIR '06, Association for Computing Machinery, New York, NY, USA, 2006, pp. 485–492, http://dx.doi.org/10.1145/1148170.1148254.

[20] F. Rousseau, E. Kiagias, M. Vazirgiannis, Text categorization as a graph classification problem, in: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), Association for Computational Linguistics, Beijing, China, 2015, pp. 1702–1712, http://dx.doi.org/10.3115/v1/P15-1164, URL https://www.aclweb.org/anthology/P15-1164.

[21] A. Sperduti, A. Starita, Supervised neural networks for the classification of structures, IEEE Trans. Neural Netw. 8 (3) (1997) 714–735, http://dx.doi.org/10.1109/72.572108, Conference Name: IEEE Transactions on Neural Networks.

[22] M. Gori, G. Monfardini, F. Scarselli, A new model for learning in graph domains, in: Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005., Vol. 2, (ISSN: 2161-4407) 2005, pp. 729–734 vol. 2, http://dx.doi.org/10.1109/IJCNN.2005.1555942.

[23] C. Gallicchio, A. Micheli, Graph echo state networks, in: The 2010 International Joint Conference on Neural Networks (IJCNN), (ISSN: 2161-4407) 2010, pp. 1–8, http://dx.doi.org/10.1109/IJCNN.2010.5596796.

[24] J. Bruna, W. Zaremba, A. Szlam, Y. LeCun, Spectral networks and locally connected networks on graphs, 2014, ArXiv:1312.6203 [Cs], http://arxiv.org/abs/1312.6203, URL arXiv:1312.6203.

[25] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, 2017, ArXiv:1606.09375 [Cs, Stat], http://arxiv.org/abs/1606.09375, URL arXiv:1606.09375.

[26] A. Micheli, Neural network for graphs: A contextual constructive approach, IEEE Trans. Neural Netw. 20 (3) (2009) 498–511, http://dx.doi.org/10.1109/TNN.2008.2010350, Conference Name: IEEE Transactions on Neural Networks.

[27] J. Gilmer, S.S. Schoenholz, P.F. Riley, O. Vinyals, G.E. Dahl, Neural message passing for quantum chemistry, 2017, ArXiv:1704.01212 [Cs], http://arxiv.org/abs/1704.01212, URL arXiv:1704.01212.

[28] D. Flam-Shepherd, T. Wu, P. Friederich, A. Aspuru-Guzik, Neural message passing on high order paths, 2020, ArXiv:2002.10413 [Cs, Stat], http://arxiv.org/abs/2002.10413, URL arXiv:2002.10413.

[29] W.L. Hamilton, R. Ying, J. Leskovec, Inductive representation learning on large graphs, 2018, ArXiv:1706.02216 [Cs, Stat], http://arxiv.org/abs/1706.02216, URL arXiv:1706.02216.

[30] R. Levie, F. Monti, X. Bresson, M.M. Bronstein, Cayleynets: Graph convolutional neural networks with complex rational spectral filters, IEEE Trans. Signal Process. 67 (1) (2019) 97–109, http://dx.doi.org/10.1109/TSP.2018.2879624, Conference Name: IEEE Transactions on Signal Processing.

[31] T. Yan, H. Zhang, Z. Li, Y. Xia, Stochastic graph recurrent neural network, 2020, ArXiv:2009.00538 [Cs, Stat], http://arxiv.org/abs/2009.00538, URL arXiv:2009.00538.

[32] C. Chen, W. Ye, Y. Zuo, C. Zheng, S.P. Ong, Graph networks as a universal machine learning framework for molecules and crystals, Chem. Mater. 31 (9) (2019) 3564–3572, http://dx.doi.org/10.1021/acs.chemmater.9b01294, Publisher: American Chemical Society.

[33] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, Graph attention networks, 2018, ArXiv:1710.10903 [Cs, Stat], http://arxiv.org/abs/1710.10903, URL arXiv:1710.10903.

[34] O. Vinyals, S. Bengio, M. Kudlur, Order matters: Sequence to sequence for sets, 2016, ArXiv:1511.06391 [Cs, Stat], http://arxiv.org/abs/1511.06391, URL arXiv:1511.06391.

[35] R. Ying, J. You, C. Morris, X. Ren, W.L. Hamilton, J. Leskovec, Hierarchical graph representation learning with differentiable pooling, 2019, ArXiv:1806.08804 [Cs, Stat], http://arxiv.org/abs/1806.08804, URL arXiv:1806.08804.

[36] F. Diehl, Edge contraction pooling for graph neural networks, 2019, ArXiv:1905.10990 [Cs, Stat], http://arxiv.org/abs/1905.10990, URL arXiv:1905.10990.

[37] H. Gao, S. Ji, Graph U-nets, 2019, ArXiv:1905.05178 [Cs, Stat], http://arxiv.org/abs/1905.05178, URL arXiv:1905.05178.

[38] J. Lee, I. Lee, J. Kang, Self-attention graph pooling, 2019, ArXiv:1904.08082 [Cs, Stat], http://arxiv.org/abs/1904.08082, URL arXiv:1904.08082.

[39] X. Gao, H. Xiong, P. Frossard, Ipool – information-based pooling in hierarchical graph neural networks, 2019, ArXiv:1907.00832 [Cs, Stat], http://arxiv.org/abs/1907.00832, URL arXiv:1907.00832.

[40] Y. Ma, S. Wang, C.C. Aggarwal, J. Tang, Graph convolutional networks with eigenpooling, 2019, ArXiv:1904.13107 [Cs, Stat], http://arxiv.org/abs/1904.13107, URL arXiv:1904.13107.

[41] S. Rhee, S. Seo, S. Kim, Hybrid approach of relation network and localized graph convolutional filtering for breast cancer subtype classification, 2018, ArXiv:1711.05859 [Cs], http://arxiv.org/abs/1711.05859, URL arXiv:1711.05859.

[42] I.S. Dhillon, Y. Guan, B. Kulis, Weighted graph cuts without eigenvectors a multilevel approach, IEEE Trans. Pattern Anal. Mach. Intell. 29 (11) (2007) 1944–1957, http://dx.doi.org/10.1109/TPAMI.2007.1115.

[43] M. Simonovsky, N. Komodakis, Dynamic edge-conditioned filters in convolutional neural networks on graphs, 2017, ArXiv:1704.02901 [Cs], http://arxiv.org/abs/1704.02901, URL arXiv:1704.02901.

[44] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D.G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, X. Zheng, Tensorflow: A system for large-scale machine learning, 2016, pp. 265–283, https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi.

[45] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, Tensorflow: Large-scale machine learning on heterogeneous distributed systems, 2016, ArXiv:1603.04467 [Cs], http://arxiv.org/abs/1603.04467, URL arXiv:1603.04467.

[46] B. van Merriënboer, D. Bahdanau, V. Dumoulin, D. Serdyuk, D. Warde-Farley, J. Chorowski, Y. Bengio, Blocks and fuel: Frameworks for deep learning, 2015, ArXiv:1506.00619 [Cs, Stat], http://arxiv.org/abs/1506.00619, URL arXiv:1506.00619 version: 1.

[47] F. Chollet, Keras, 2015, https://github.com/fchollet/keras, Publication Title: GitHub repository.

[48] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, Pytorch: An imperative style, high-performance deep learning library, 2019, ArXiv:1912.01703 [Cs, Stat], http://arxiv.org/abs/1912.01703, URL arXiv:1912.01703.

[49] M. Fey, J.E. Lenssen, Fast graph representation learning with pytorch geometric, 2019, ArXiv:1903.02428 [Cs, Stat], http://arxiv.org/abs/1903.02428, URL arXiv:1903.02428.

[50] M. Wang, D. Zheng, Z. Ye, Q. Gan, M. Li, X. Song, J. Zhou, C. Ma, L. Yu, Y. Gai, T. Xiao, T. He, G. Karypis, J. Li, Z. Zhang, Deep graph library: A graph-centric, highly-performant package for graph neural networks, 2020, ArXiv:1909.01315 [Cs, Stat], http://arxiv.org/abs/1909.01315, URL arXiv:1909.01315.

[51] Y. Li, O. Vinyals, C. Dyer, R. Pascanu, P. Battaglia, Learning deep generative models of graphs, 2018, ArXiv:1803.03324 [Cs, Stat], http://arxiv.org/abs/1803.03324, URL arXiv:1803.03324.

[52] S. Sabour, N. Frosst, G.E. Hinton, Dynamic routing between capsules, 2017, ArXiv:1710.09829 [Cs], http://arxiv.org/abs/1710.09829, URL arXiv:1710.09829.

[53] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, 2017, ArXiv:1706.03762 [Cs], http://arxiv.org/abs/1706.03762, URL arXiv:1706.03762.

[54] D. Grattarola, C. Alippi, Graph neural networks in tensorflow and keras with spektral, 2020, ArXiv:2006.12138 [Cs, Stat], http://arxiv.org/abs/2006.12138, URL arXiv:2006.12138.

[55] C. Data61, Stellargraph machine learning library, 2018.

[56] R. Ying, D. Bourgeois, J. You, M. Zitnik, J. Leskovec, Gnnexplainer: Generating explanations for graph neural networks, 2019, ArXiv:1903.03894 [Cs, Stat], http://arxiv.org/abs/1903.03894, URL arXiv:1903.03894.

[57] J. Klicpera, S. Giri, J.T. Margraf, S. Günnemann, Fast and uncertainty-aware directional message passing for non-equilibrium molecules, 2020, ArXiv:2011.14115 [Physics], http://arxiv.org/abs/2011.14115, URL arXiv:2011.14115.

[58] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, T. Eliassi-Rad, Collective classification in network data, AI Mag. 29 (3) (2008) 93, http://dx.doi.org/10.1609/aimag.v29i3.2157, https://ojs.aaai.org/index.php/aimagazine/article/view/2157, Number: 3.

[59] A.K. Debnath, R.L. Lopez de Compadre, G. Debnath, A.J. Shusterman, C. Hansch, Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity, J. Med. Chem. 34 (2) (1991) 786–797, http://dx.doi.org/10.1021/jm00106a046, Publisher: American Chemical Society.

[60] R. Ramakrishnan, P.O. Dral, M. Rupp, O.A. von Lilienfeld, Quantum chemistry structures and properties of 134 kilo molecules, Sci, Data 1 (1) (2014) 140022, http://dx.doi.org/10.1038/sdata.2014.22, https://www.nature.com/articles/sdata201422, Number: 1 Publisher: Nature Publishing Group.

[61] K.T. Schütt, M. Gastegger, A. Tkatchenko, K.-R. Müller, R.J. Maurer, Unifying machine learning and quantum chemistry with a deep neural network for molecular wavefunctions, Nature Commun. 10 (1) (2019) 5024, http://dx.doi.org/10.1038/s41467-019-12875-2, URL https://www.nature.com/articles/s41467-019-12875-2, Number: 1 Publisher: Nature Publishing Group.

[62] O.A. von Lilienfeld, Quantum machine learning in chemical compound space, Ange. Chem. Int. Ed. 57 (16) (2018) 4164–4169, http://dx.doi.org/10.1002/anie.201709686, URL https://onlinelibrary.wiley.com/doi/abs/10.1002/anie.201709686.

[63] R. Gómez-Bombarelli, J. Aguilera-Iparraguirre, T.D. Hirzel, D. Duvenaud, D. Maclaurin, M.A. Blood-Forsythe, H.S. Chae, M. Einzinger, D.-G. Ha, T. Wu, G. Markopoulos, S. Jeon, H. Kang, H. Miyazaki, M. Numata, S. Kim, W. Huang, S.I. Hong, M. Baldo, R.P. Adams, A. Aspuru-Guzik, Design of efficient molecular organic light-emitting diodes by a high-throughput virtual screening and experimental approach, Nature Mater. 15 (10) (2016) 1120–1127, http://dx.doi.org/10.1038/nmat4717, URL https://www.nature.com/articles/nmat4717, Number: 10 Publisher: Nature Publishing Group.

[64] M. Rupp, Machine learning for quantum mechanics in a nutshell, Int. J. Quantum Chem. 115 (16) (2015) 1058–1073, http://dx.doi.org/10.1002/qua.24954, URL https://onlinelibrary.wiley.com/doi/abs/10.1002/qua.24954.

[65] K. Do, T. Tran, S. Venkatesh, Graph transformation policy network for chemical reaction prediction, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, in: KDD '19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 750–760, http://dx.doi.org/10.1145/3292500.3330958.

[66] M. Gastegger, P. Marquetand, High-dimensional neural network potentials for organic reactions and an improved training algorithm, J. Chem. Theory Comput. 11 (5) (2015) 2187–2198, http://dx.doi.org/10.1021/acs.jctc.5b00211, Publisher: American Chemical Society.

[67] C. W. Coley, W. Jin, L. Rogers, T. F. Jamison, T. S. Jaakkola, W. H. Green, R. Barzilay, K. F. Jensen, A graph-convolutional neural network model for the prediction of chemical reactivity, Chem. Sci. 10 (2) (2019) 370–377, http://dx.doi.org/10.1039/C8SC04228D, https://pubs.rsc.org/en/content/articlelanding/2019/sc/c8sc04228d, Publisher: Royal Society of Chemistry.

[68] V.R. Somnath, C. Bunne, C.W. Coley, A. Krause, R. Barzilay, Learning graph models for template-free retrosynthesis, 2020, ArXiv:2006.07038 [Cs, Stat], http://arxiv.org/abs/2006.07038, URL arXiv:2006.07038.

[69] J. Behler, Representing potential energy surfaces by high-dimensional neural network potentials, J. Phys.: Condens. Matter 26 (18) (2014) 183001, http://dx.doi.org/10.1088/0953-8984/26/18/183001, Publisher: IOP Publishing.

[70] J. Westermayr, M. Gastegger, P. Marquetand, Combining schnet and SHARC: The schnarc machine learning approach for excited-state dynamics, J. Phys. Chem. Lett. 11 (10) (2020) 3828–3834, http://dx.doi.org/10.1021/acs.jpclett.0c00527, Publisher: American Chemical Society.

[71] J. Behler, Constructing high-dimensional neural network potentials: A tutorial review, Int. J. Quantum Chem. 115 (16) (2015) 1032–1050, http://dx.doi.org/10.1002/qua.24890, https://onlinelibrary.wiley.com/doi/abs/10.1002/qua.24890.

[72] P. Reiser, M. Konrad, A. Fediai, S. Léon, W. Wenzel, P. Friederich, Analyzing dynamical disorder for charge transport in organic semiconductors via machine learning, J. Chem. Theory Comput. (2021) http://dx.doi.org/10.1021/acs.jctc.1c00191, Publisher: American Chemical Society.

[73] J. Li, P. Reiser, B.R. Boswell, A. Eberhard, N.Z. Burns, P. Friederich, S.A. Lopez, Automatic discovery of photoisomerization mechanisms with nanosecond machine learning photodynamics simulations, Chem. Sci. 12 (14) (2021) 5302–5314, http://dx.doi.org/10.1039/D0SC05610C, https://pubs.rsc.org/en/content/articlelanding/2021/sc/d0sc05610c, Publisher: The Royal Society of Chemistry.

[74] W.L. Hamilton, R. Ying, J. Leskovec, Representation learning on graphs: Methods and applications, 2018, ArXiv:1709.05584 [Cs], http://arxiv.org/abs/1709.05584, URL arXiv:1709.05584.