MHG-GNN: Combination of Molecular Hypergraph Grammar with Graph Neural Network

Akihiro Kishimoto
IBM Research - Tokyo

Hiroshi Kajino
IBM Research - Tokyo

Masataka Hirose
JSR Corporation

Junta Fuchiwaki
JSR Corporation

Indra Priyadarsini
IBM Research - Tokyo

Lisa Hamada
IBM Research - Tokyo

Hajime Shinohara
IBM Research - Tokyo

Daiju Nakano
IBM Research - Tokyo

Seiji Takeda
IBM Research - Tokyo

Abstract

Property prediction plays an important role in material discovery. As an initial step to eventually develop a foundation model for material science, we introduce a new autoencoder called the MHG-GNN, which combines graph neural network (GNN) with Molecular Hypergraph Grammar (MHG). Results on a variety of property prediction tasks with diverse materials show that MHG-GNN is promising.

1 Introduction

Machine learning for effective material design has been a topic of interest. Machine learning models for accurately predicting chemical or physical properties help material scientists have better understandings to the materials of interest more quickly than simulations and trial-and-error experiments.

A major obstacle is the limited availability of training data in material science. A prediction task for a specific target property often has only tens of pairs of molecular structures and their target values [41]. Data-hungry, modern machine learning algorithms cannot benefit from only such a small dataset.

A *foundation model* (FM) [5] trained with a huge unlabeled dataset aims to address the data shortage. With a small labeled dataset and the FM, each prediction task is modeled as a *downstream* task. Because of public, large-scale unlabeled *base* datasets on molecular structures [16, 21], building FMs for materials science [1, 42] has attracted much attention.

One approach discussed in [42] requires the FMs for materials science to meet at least the following three requirements. First, *any* molecule should be embedded into a latent vector representation. Second, the latent representation should effectively represent features of molecules to be able to address a wide variety of downstream tasks including property prediction. Third, a decoder included for molecular optimization should map a latent vector into a *structurally valid* molecule *without any failure*. Satisfying these requirements allows the FM to solve many property prediction tasks with little effort as well as to directly achieve an ultimate goal of optimizing molecules.

As a first step toward this end, we present MHG-GNN, an autoencoder architecture that has an encoder based on GNN and a decoder based on a sequential model with MHG [19]. Since the encoder is a GNN variant, MHG-GNN can accept any molecule as input, and demonstrate high predictive performance on molecular graph data. In addition, the decoder inherits the theoretical guarantee of MHG on always generating a structurally valid molecule as output.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

As initial experiments, we evaluate MHG-GNN with a total of six downstream tasks on three different kinds of materials, and show that MHG-GNN performs better than two other well-known approaches, while opening up further opportunities as a next step to develop the FM for material science.

2 Background

We review the work related to our attempt to develop a unimodal model of MHG-GNN supporting three requirements in Section 1, aiming to be extended to an approach discussed in [42] in the future.

Since SMILES is a text format representing a two-dimensional molecular graph, molecules are regarded either as sequences of characters or as graphs (see [9] for a survey). Irrespective of whether to pretrain models or not, existing approaches perform learning without any labels.

Although the requirements we have discussed include a valid decoder, decoders are not often supported in both representations [1, 4, 37, 38, 46, 47, 48]. Even if the decoders are available, many of them do not handle structural constraints such as the total number of bonds for a carbon, occasionally resulting in invalid molecules violating these constraints [10, 11, 13, 24, 28, 40].

Large language models[3, 30, 43] based on Transformer [44] and its variants [8, 33] attempt to learn efficient latent representations on natural language. There are attempts to deal with SMILES as language e.g., [6, 13]. However, in addition to the decoder issues, misclassification in the latent space may occur if two similar molecules have very different SMILES strings [18].

There are approaches to support valid decoders such as JT-VAE [18], MHG-VAE [19] and SELFIES [27]. However, for a new molecule, JT-VAE can utilize only a part of the latent vector due to its junction tree representation. MHG-VAE may not be able to encode it to a latent vector. From a viewpoint of downstream tasks related to one of the requirements, the latent vector of any molecule must be computed to be able to assess the molecules of interest.

A SELFIES-based decoder can be combined with our approach. However, as a first step, we focus on an MHG-based decoder. A comparison with the SELFIES-based decoder remains future work.

3 MHG-GNN

To the best of our knowledge, in designing autoencoder [23], all previous approaches represent their input and output in the *identical* format such as from SMILES string to SMILES string [11, 13]. Using the identical format ends up with only partially resolving the issues discussed so far.

MHG-GNN is an autoencoder with *different* representations as its input and output. This leads to meeting all the requirements to be a part of the FM. MHG-GNN receives a two-dimensional graph as input to its GNN-based encoder, and the output of its RNN-based decoder is a production rule sequence of MHG. Unlike MHG-VAE, MHG-GNN can encode any molecular graph without accounting for the existence of the production rules in the base dataset as well as to directly embed graph structures into their latent space. Additionally, like MHG-VAE, the decoder of MHG-GNN can always return structurally valid molecules stemming from MHG.

3.1 Encoder

Our encoder employs a Graph Isomorphism Network (GIN) [45] variant that additionally accounts for embedding edges [14, 25].

Each atom of a molecule is represented as chemical features, e.g., atomic number, formal charge, and aromaticity. Each atom feature is embedded to a vector with the same dimension size as its corresponding node in GIN. All embedded atom features are summed to represent an initial vector h_i^0 for GIN node i. Each edge of a molecular structure such as bond type is embedded analogously to an embedding vector $e_{j,i}$ on its GIN undirected edge between nodes j and i.

In iteration k, the encoder performs so-called message passing for each node i, defined as follows:

$$h_i^{k+1} = \text{MLP}((1+\epsilon)h_i^k + \sum_{j \in N(i)} \text{ReLU}(h_j^k + e_{j,i}))$$

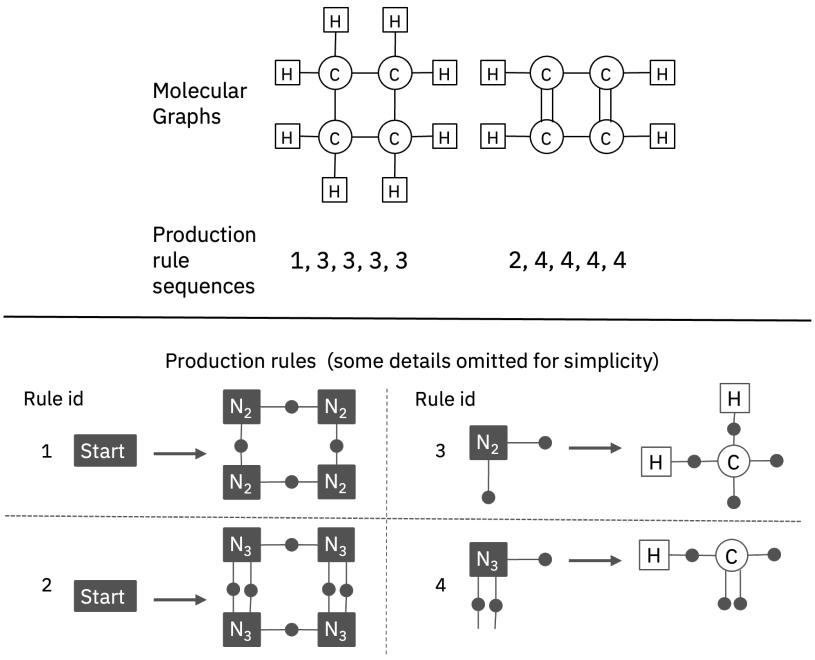

Figure 1: MHG and production rule sequence. Note implicit hydrogen is used in practice.

where $N(i)$ is a set of direct neighbors of i, ϵ is a trainable parameter, MLP is a neural network module, and ReLU is a Rectified Linear Unit. In our encoder, $\text{MLP}(x) := \text{Lin}(\text{ReLU}(\text{BN}(\text{Lin}(x))))$, where Lin is a linear layer and BN is a one-dimensional Batch Normalization layer [15].

Finally, the entire representation h_G of graph G is used as a latent vector to address downstream tasks. As suggested in [45] from a theoretical viewpoint on model expressiveness, h_G is defined as:

$$h_G = \text{CONCAT}(\sum_{i \in V_G} h_i^k | k = 0, 1, \cdots, r)$$

where CONCAT concatenates vectors, V_G is a set of nodes in G, and r is the maximum iteration size.

3.2 MHG

Approaches available to always generate valid molecular structures include MHG [19], Junction Tree (JT) [18] and Reversible JT (RJT) [17]. We choose MHG for MHG-GNN, because MHG enables MHG-GNN to build a simpler architecture as well as to leverage the entire embedding.

Given a dataset on molecular structures, MHG generates production rules on molecular hypergraphs. In this paper, we only describe the notion of MHG and production rules. See [19] for details of the algorithms to generate the rules.

Figure 1 shows an example of four rules necessary to generate two molecules. Each rule has a precondition and an effect defined at the left and right of its arrow, respectively. An edge of a normal molecular graph is split into two parts connected with one small, gray circle called a *hypernode* in its corresponding hypergraph. A *hyperedge* denoted as a square is connected to any number of hypernodes. A hyperedge has two types of symbol. A square or a circle with an atom is called a terminal symbol. A non-terminal symbol includes a non-atom symbol.

For example, in Figure 1, hyperedges N_2 and N_3 connect two and three hypernodes, respectively. Hyperedge C always has a total of four hypernodes, while hyperedge H consists of only one hypernode.

For a molecule M, MHG attempts to find a sequence of production rules to generate M, starting with the *start* symbol. In applying rule r to hypergraph G, MHG checks whether G has a subgraph that matches the precondition of r or not. If G has such a subgraph, that subgraph is replaced with the

Table 1: Summary on datasets

Dataset name	Size	Property value creation	Structural variety	Miscellaneous
Polymer	5000	Simulations	High	Monomer weight ≤ 300
Photoresist	7884	Simulations	High	
Chromophore	346	Experiments	Low	

effect of r. In this rule application, any hypernode in G continues preserving exactly two hyperedges, which is essential to ensure the structural validity. If G does not have any subgraph to match, r is safely ignored. This leads to significantly reducing the number of production rules to consider.

If G has only terminal symbols, G is easily converted to a structurally valid molecule by just removing all hypernodes from G. Otherwise, MHG needs to find an applicable rule that can further reduce G.

For example, in generating a molecule represented as `C1CCC1` in SMILES, MHG starts with the start symbol and applies rule 1. Then, rule 3 is the only applicable rule and the others do not need to be considered. MHG, therefore, applies rule 3 four times, generating a sequence `1, 3, 3, 3, 3` which is a hypergraph identical to `C1CCC1`. Analogously, a production rule sequence `2, 4, 4, 4, 4` leads to `C1=CC=C1`.

3.3 Decoder

While h_G is used as a latent vector, additional layers are prepared before it is processed by the decoder. h_G is passed to a mean module and a log variance module on multi-dimensional Gaussian distributions to perform the reparameterization trick [12]. Each of these modules performs $\tanh(\eta \mathrm{Lin}(h_G))$, where η is a trainable scalar. After its output is adapted with a linear layer, it is passed to the initial hidden unit of the decoder.

The decoder employs an architecture similar to that of MHG-VAE [19]. It consists of layers of Gated Recurrent Unit (GRU) [7] and handles as output a sequence of production rule embeddings. Each rule embedding in the sequence is converted back to a probability constrained by MHG as necessary.

4 Performance Evaluation

Our base dataset consists of 1,381,747 molecules extracted from the PubChem database [21]. 16,362 rules are generated to represent these molecules as production rule sequences.

We implement MHG-GNN in Python using PyTorch [31], PyTorch Geometric (PyG) [32] and RDKit [35], and train its models with the β-VAE loss function [12], the Adam optimizer [22] and the ReduceLROnPlateau scheduler on a cluster of machines consisting of Intel E5-2667 CPUs at 3.30GHz and NVIDIA A100 Tensor Core GPUs.

As defined in PyG, each atom and bond consist of 9 and 3 features, respectively, and are embedded to 256 dimensions for a node in GIN. The dimension size of Gaussian distributions is also set to 256. The values of ϵ and η in MHG-VAE are initialized to 0 and 1. Each GRU layer consists of 384 hidden units and its layer size is set to 3. The production rule embedding size is set to 128.

To train MHG-GNN, we set the batch size, the dropout rate, the frequency of scheduler invocations, λ for Adam, and β for β-VAE to 512, 0.1, 1000, 5.0×10^{-4}, and 0.01, respectively.

4.1 Setup for Downstream Tasks

Table 1 shows our three proprietary datasets containing materials of diverse characteristics.

We evaluate the following methods that perform six downstream tasks on various property predictions: (1) **MHG-GNN**$_r$ with a radius of $r = 3, 5, 6, 7, 8$ (i.e., the number of iterations for neighborhood aggregations) and $256(r + 1)$ dimensions, (2) **ECFP6**, a well-known fingerprint descriptor on two-dimensional molecular substructures with 1024 bit [36], and (3) **Mordred**, a freely-available, high-performing software calculating 1825 two- and three- dimensional descriptors [29]. We exclude three-dimensional descriptors of Mordred for a fairer comparison to the others that consider only two-dimensional structures. This results in Mordred using 1613 dimensions.

Table 2: Performance on various downstream tasks (test datasets, R^2 score)

	Polymer			Photoresist		Chromophore
Method	Density	Bulk modulus	Refractive index	HOMO	LUMO	λ_{max} on NIR
MHG-GNN	**0.578**	**0.516**	**0.865**	**0.896**	**0.845**	**0.845**
ECFP6	0.523	0.482	0.823	0.791	0.782	0.708
Mordred	0.567	0.505	0.859	0.894	0.830	0.842

Since r is regarded as a hyperparameter, which value should be used needs to be decided before the test dataset is evaluated. We select r based on the best R^2 score on the validation dataset.

In our preliminary performance evaluation with a different dataset, we find that MHG-VAE cannot encode roughly 50% of the molecules due to a failure of generating appropriate production rule sequences. Our evaluation, therefore, excludes MHG-VAE here.

We split these downstream datasets into training, validation and test datasets with the ratio of 0.6, 0.2 and 0.2, respectively. We create downstream models by the LightGBM algorithm [20] with the RMSE loss function and optimize their hyperparameters by Optuna [2] using the validation datasets.

4.2 Results

Table 2 shows R^2 scores calculated with the test datasets. The best numbers are marked in bold. Our results clearly demonstrate that MHG-GNN outperforms ECFP6 and Mordred in all downstream tasks, and that MHG-GNN is an important candidate for property prediction tasks on material discovery.

The performance of MHG-GNN tends to be improved when its radius is increased from 3 to 6 or 7 except bulk modulus. Its performance then tends to be worsened when the radius is further increased to 8. One hypothesis to explain this behavior is that MHG-GNN might suffer from an open problem that diminishing returns is clearly observed when GNN is constructed with deeper layers [26]. The other hypothesis is that the radius of 6-7 (i.e., the diameter of 12-14) might be sufficient to capture most of the important structural information in our case. Many of the molecules in the Polymer and Photoresist datasets contain two ring structures, while the Chromophore dataset has molecules with two or three rings.

One final note is that the molecular structures in our datasets are larger than those with at most 9 heavy atoms in the well-known QM9 dataset [34, 39]. While QM9 could be addressed by GNN with a smaller radius, developing effective algorithms for GNN with a larger radius is necessary to be able to practically deal with larger molecules like in our datasets.

5 Conclusions

As a first step to build an FM, we have introduced MHG-GNN and evaluated its performance on various prediction tasks on diverse materials, showing that MHG-GNN outperforms Mordred and ECFP6, two of well-known approaches applicable to property prediction.

There are numerous research directions as future work. For example, more comprehensive understanding to MHG-GNN is necessary, including comparisons with other pretrained models such as [13, 18, 38], integrations with other decoders including SELFIES, and evaluations with a larger base dataset as well as other downstream tasks including molecular optimization. Improving the performance of MHG-GNN on downstream tasks is also of interest, such as development of new model architectures that overcome the performance degradation with many GNN layers. Finally, toward a second step to build the FM, extending MHG-GNN to support multimodalities (e.g., three-dimensional coordinates and properties easily calculated) is an important topic to investigate.

References

[1] W. Ahmad, E. Simon, S. Chithrananda, G. Grand, and B. Ramsundar. ChemBERTa-2: Towards chemical foundation models. `https://arxiv.org/abs/2209.01712`, 2022.

[2] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2623–2631, 2019.

[3] R. Anil, A. M. Dai, O. Firat, and M. J. *et al.* Palm 2 technical report. Arxiv: `https://arxiv.org/abs/2305.10403`, 2023.

[4] D. M. Anstine and O. Isayev. Generative models as an emerging paradigm in the chemical sciences. *Journal of the American Chemical Society*, 145(16):8736–8750, 2023.

[5] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, and S. A. *et al.* On the opportunities and risks of foundation models. Arxiv: `https://arxiv.org/abs/2108.07258`, 2021.

[6] S. Chithrananda, G. Grand, and B. Ramsundar. ChemBERTa: Large-scale self-supervised pretraining for molecular property prediction. Arxiv: `https://arxiv.org/abs/2010.09885`, 2022.

[7] K. Cho, B. van Merriënboer, C. Gucehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*, pages 1724–1734, 2014.

[8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *ACL*, pages 4171–4186, 2019.

[9] F. Faez, Y. Ommi, M. S. Baghshah, and H. R. Rabiee. Deep graph generators: A survey. *IEEE Access*, 9:106675–106702, 2021.

[10] D. Flam-Shepherd, T. Wu, and A. Aspuru-Guzik. Graph deconvolutional generation. Arxiv: `https://arxiv.org/abs/2002.07087`, 2020.

[11] R. Gómez-Bombarelli, J. N. Wei, D. Duvenaud, J. M. Hernández-Lobato, B. Sánchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, T. D. Hirzel, R. P. Adams, and A. Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, 4(2):268–276, 2018.

[12] I. Higgins, L. Matthey, A. Pal, C. B. X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. β-VAE: Learning basic visual concepts with a constrained variational framework. In *ICLR*, 2017.

[13] S. Honda, S. Shi, and H. R. Ueda. SMILES Transformer: Pre-trained molecular fingerprint for low data drug discovery. Arxiv: `https://arxiv.org/abs/1911.04738`.

[14] W. Hu, B. Liu, J. Gomes, M. Zitnik, P. Liang, V. Pande, and J. Leskovec. Strategies for pre-training graph neural networks. In *ICLR*, 2020.

[15] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, pages 448–456, 2015.

[16] J. J. Irwin, K. G. Tang, J. Young, C. Dandarchuluun, B. R. Wong, M. Khurelbaatar, Y. S. Moroz, J. Mayfield, and R. A. Sayle. ZINC20 - a free ultralarge-scale chemical database for ligand discovery. *Journal of Chemical Information and Modeling*, 60(12):6065–6073, 2020.

[17] R. Ishitani, T. Kataoka, and K. Rikimaru. Molecular design method using a reversible tree representation of chemical compounds and deep reinforcement learning. *Journal Chemical Information and Modeling*, 62(17):4032–4048, 2022.

[18] W. Jin, R. Barzilay, and T. Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *ICML*, pages 2328–2337, 2018.

[19] H. Kajino. Molecular hypergraph grammar with its application to molecular optimization. In *ICML*, pages 3183–3191, 2019. Also see the supplementary material available at `http://proceedings.mlr.press/v97/kajino19a/kajino19a-supp.pdf`.

[20] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. LightGBM: A highly efficient gradient boosting decision tree. In *NeurIPS*, pages 3149–3157, 2017.

[21] S. Kim, J. Chen, T. Cheng, A. G. J. He, S. He, Q. Li, B. A. Shoemaker, P. A. T. B. Yu, L. Zaslavsky, J. Zhang, and E. E. Bolton. PubChem in 2021: New data content and improved web interfaces. *Nucleic Acids Research*, 49(D1):D1388–D1395, 2021.

[22] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[23] D. P. Kingma and M. Welling. Auto-encoding variational Bayes. In *ICLR*, 2014.

[24] T. N. Kipf and M. Welling. Variational graph auto-encoders. In *NeurIPS Workshop on Bayesian Deep Learning*, 2016. Available at `https://arxiv.org/abs/1611.07308`.

[25] S. Liu, H. Wang, W. Liu, J. Lasenby, H. Guo, and J. Tang. Pre-training molecular graph representation with 3D geometry. In *ICLR*, 2022.

[26] Z. Liu, C. Chen, X. Yang, J. Zhou, X. Li, and L. Song. Heterogeneous graph neural networks for malicious account detection. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management (CIKM)*, pages 2077–2085, 2018.

[27] M. K. M, F. Häse, A. Nigam, P. Friederich, and A. Aspuru-Guzik. Self-referencing embedded strings (SELFIES): A 100% robust molecular string representation. *Machine Learning: Science and Technology*, 1(4):045024, 2020.

[28] T. Ma, J. Chen, and C. Xiao. Constrained generation of semantically valid graphs via regularizing variational autoencoders. In *NeurIPS*, pages 7113–7124, 2018.

[29] H. Moriwaki, Y.-S. Tian, N. Kawashita, and T. Takagi. Mordred: A molecular descriptor calculator. *Journal of Cheminformatics*, 10(4), 2018.

[30] OpenAI. GPT-4 technical report. Arxiv: `https://arxiv.org/abs/2303.08774`, 2023.

[31] Pytorch. `https://pytorch.org/`.

[32] Pytorch Geometric. `https://pytorch-geometric.readthedocs.io/en/latest/`.

[33] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving language understanding by generative pre-training. Available at `https://www.mikecaptain.com/resources/pdf/GPT-1.pdf`.

[34] R. Ramakrishnan, P. O. Dral, M. Rupp, and O. A. von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1(140022), 2014.

[35] RDKit. `https://www.rdkit.org/`.

[36] D. Rogers and M. Hahn. Extended-connectivity fingerprints. *Journal Chemical Information and Modeling*, 50(5):742–754, 2010.

[37] Y. Rong, Y. Bian, T. Xu, W. Xie, Y. Wei, W. Huang, and J. Huang. Self-supervised graph transformer on large-scale molecular data. In *NeurIPS*, pages 12559–12571, 2020.

[38] J. Ross, B. Belgodere, V. Chenthamarakshan, I. Padhi, Y. Mroueh, and P. Das. Large-scale chemical language representations capture molecular structure and properties. *Nature Machine Intelligence*, 4:1256–1264, 2022.

[39] L. Ruddigkeit, R. van Deursen, L. C. Blum, and J.-L. Reymond. Enumeration of 166 billion organic small molecules in the chemical universe database GDB-17. *Journal of Chemical Information and Modeling*, 52:2864–2875, 2012.

[40] M. Simonovsky and N. Komodakis. GraphVAE: Towards generation of small graphs using variational autoencoders. In *Proceedings of the International Conference on Artificial Neural Networks*, pages 412–422, 2018.

[41] S. Takeda, T. Hama, H.-H. Hsu, V. A. Piunova, D. Zubarev, D. P. Sanders, J. W. Pitera, M. Kogoh, T. Hongo, Y. Cheng, W. Bocanett, H. Nakashika, A. Fujita, Y. Tsuchiya, K. Hino, K. Yano, S. Hirose, H. Toda, Y. Orii, and D. Nakano. Molecular inverse-design platform for material industries. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2961–2969, 2020.

[42] S. Takeda, A. Kishimoto, L. Hamada, D. Nakano, and J. R. Smith. Foundation model for material science. In *AAAI*, pages 15376–15383, 2023.

[43] H. Touvron, L. Martin, K. Stone, and P. A. *et al.* Llama 2: Open foundation and fine-tuned chat models. Arxiv: `https://arxiv.org/abs/2307.09288`, 2023.

[44] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *NeurIPS*, pages 5998–6008, 2017.

[45] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? In *ICLR*, 2019.

[46] C. Ying, T. Cai, S. Luo, S. Zheng, G. Ke, D. He, Y. Shen, and T.-Y. Liu. Do transformers really perform bad for graph representation? In *NeurIPS*, pages 28877–28888, 2021.

[47] A. Yüksel, E. Ulusoy, and A. Ü. T. Doğan. SELFormer: molecular representation learning via SELFIES language models. *Machine Learning: Science and Technology*, 4(2):025035, 2023.

[48] S. Yun, M. Jeong, R. Kim, J. Kang, and H. J. Kim. Graph transformer networks. In *NeurIPS*, pages 11960–11970, 2019.