Hyperparameter Optimization of Graph Neural Networks for mRNA Degradation Prediction

Viktorija Vodilovska* Sonja Gievska* Ilinka Ivanoska*
* Faculty of Computer Science and Engineering, Ss Cyril and Methodius University, Skopje, N. Macedonia
viktorija.vodilovska@students.finki.ukim.mk sonja.gievska@finki.ukim.mk ilinka.ivanoska@finki.ukim.mk

***Abstract*—Graph Neural Networks (GNN) emerged as increasingly attractive deep learning models for complex data, making them extremely useful in biochemical and pharmaceutical domains. However, building a good-performing GNN requires lots of parameter choices and Hyperparameter Optimization (HPO) can aid in exploring solutions. This study presents a comparative analysis of several strategies for Hyperparameter Optimization of GNNs. The explored optimization techniques include complex algorithms such as the bio-inspired Genetic Algorithm, Particle Swarm Optimization, and Artificial Bee Colony. In addition, Hill Climb and Simulated Annealing as well as the commonly used methods Random Search and Bayesian Search have also been covered.**

The proposed optimization algorithms have been evaluated on improving the performance of the GNN architectures developed for predicting mRNA degradation. The Stanford OpenVaccine dataset for mRNA degradation prediction has been used for training and testing the predictive models. Finding mRNA molecules with low degradation rates is important in development of mRNA vaccines for diseases such as COVID-19 and we hope to benefit research on ML in this domain. According to the analysis's findings, Simulated Annealing algorithm outperforms other algorithms on both architectures. Furthermore, population based algorithms like Particle Swarm Optimization show promising results, with certain limitations related to the complexity of the algorithms which encourages further exploration of the subject.

Keywords*—*Hyperparameter Optimization, Random Search, Bayesian search, Hill Climbing, Simulated Annealing, Genetic Algorithm, Artificial Bee Colony, Particle Swarm Optimization, GCN, GAT, mRNA degradation, mRNA vaccines

I. INTRODUCTION

Graph Neural Networks (GNN) are state-of-the-art methods for deep learning on data with a complex graph structure. Graphs can represent a wide variety of problems, which allows these models to be very expressive and applicable to arbitrary graph data of different shapes and sizes

Due to the growing quantities of biological and medical data, Graph Neural Networks have become an important tool in bioinformatics over the past years [1]. Graphs have been used to represent various types of biological information like genetic disease association networks, protein interaction networks, structure of molecules and macromolecules (proteins, DNA, RNA), brain networks, etc. With the growing complexity of these graph models, the need for an optimization strategy to build effective models also increased.

A model performance is directly linked to how well it was built to fit the specific data and problem. This requires choosing the right values for a wide range of hyperparameters that lie in both continuous and discrete feature spaces. Hyperparameter optimization (HPO) or tuning is the problem of selecting a set of values for the parameters of a learning algorithm, with the expectation of achieving better model performance. A hyperparameter is a parameter whose value is used to control the algorithm's learning process. As the hyperparameter space gets broader so does the difficulty of finding a good-performing architecture for the given problem. An extensive grid search over all those values is expensive and not always an option.

This study presents an experimental comparative analysis over a larger selection of optimization methods for the Hyperparameter Optimization of GNNs on the task of mRNA degradation prediction. In doing so, the aim is to facilitate future research of GNN application in bioinformatics. The experiments include the following HPO algorithms: Random Search (RS) [2] for benchmark, Bayesian Search (BS) [3], Hill Climb (HC) [4], Simulated Annealing (SA) [5], Genetic Algorithm (GA) [6], Particle Swarm Optimization (PSO) [7] and Artificial Bee Colony (ABC) Optimization [8]. The GNN architectures probed in the HPO experiments are GCN [9] and GAT [10]. The analysis used the Stanford OpenVaccine Dataset [11] for mRNA degradation prediction which is an important issue when developing mRNA vaccines for diseases such as COVID-19, and a relatively novel field. The GNN prediction task involved regressive multi-target node prediction.

The rest of the paper is structured as follows: an overview of the existing literature on the topic is given in section II. The dataset, explored GNN architectures as well as the hyperparameter optimization algorithms are explained in detail in section III. The experimental setup with results and discussion of the experiments are laid out in section IV. Finally, section V concludes the findings, and a reference to supplementary material is given in section VI.

II. RELATED WORK

Hyperparameter optimization (HPO) is a crucial step in the development of any high-performing neural networks.

Supported by Faculty of Computer Science and Engineering, Skopje, N. Macedonia.

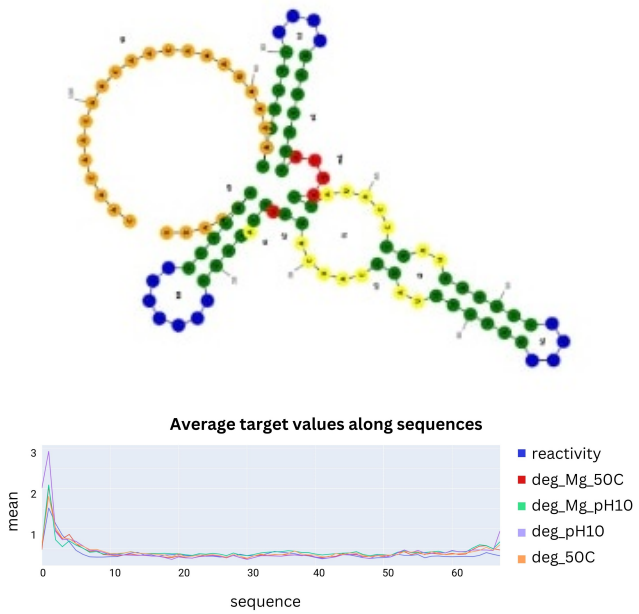

Fig. 1: (a) The 2D structure of an example mRNA molecule. Nodes represent the nucleotides of the mRNA. Image generated with the ViennaRNA package [12]. (b) Average of target values, with each line representing a different target, along all train molecules.

However, only a few studies have been undertaken to tackle the issue of finding a well-performing HPO strategy for GNNs.

A systematic comparison of three HPO algorithms, with RS for baseline, Tree-structured Parzen Estimator (TPE), and Covariance Matrix Adaptation Evolution Strategy (CMA-ES), was conducted in [13] on GNN models for molecular property prediction using three representative datasets from DeepChem's [14] MoleculeNet [15]. The search space includes values for batch size, learning rate, size of fully-connected layer, and size of a graph convolution layer. TPE outperforms RS and CMA-ES, achieving the best overall performance. Different studies such as DeepHyper [16], or [17], [18] suggest a preference for BS HPO techniques and also show promising results on GA for HPO. Moreover, other bio-inspired population algorithms like PSO have been effectively used in hyperparameter optimization tasks for Convolutional Neural Networks and Long Short-Term Memory (LSTM) networks (see [19], [20]).

Neural Architecture Search (NAS) framework can be considered as another automatic approach for choosing hyperparameters. For example the work on AGNN [21] proposes an AutoML framework NAS on GNNs, with the use of reinforcement learning for decisions on new actions for each class, while trying to optimize the objective function - the model performance. [22] proposes another AutoML framework for GNNs called DFG-NAS (Deep and Flexible Graph Neural Architecture Search) that employs an evolutionary algorithm and search space decoupling into propagation and transformation. Both of these works are evaluated on popular benchmarks like Cora, Citeseer, and PubMed, however their optimization frameworks are model specific to the chosen architecture compared to the model-agnostic optimization algorithms explored in this study.

Overall, while several studies have explored HPO methods for GNN-based models, our survey aims to provide a comprehensive comparison of seven different HPO algorithms (RS, BS, HC, SA, GA, PSO, ABC) on GNN models like GAT and GCN. This study focuses on the HPO of GNNs built specifically for prediction on mRNA molecules which we found to be a less explored topic. Our search space includes a wide range of micro and macro architecture hyperparameters such as learning rate, dropout rate, number of prediction layers, number of hidden units, number of graph layers. Since the optimization algorithms explored in this solution are model and problem agnostic we hope this study benefits research in other areas as well.

III. Materials and Methods

This section presents the dataset, GNN architecture, and hyperparameter optimization algorithms used.

A. Dataset

The Stanford Eterna Dataset [11] consists of mRNA molecules represented by it's sequence of nucleotides with structural features and targets. A visual representation of an mRNA molecule can be seen on Fig.1. During preprocessing each mRNA molecule was transformed into a graph $G = (V, E)$, with nodes V representing the nucleotides and edges E - the chemical bonds between them. Nucleotide types are used as node features while the primary structure bonds as well as the base pair probabilities are used as edge features, as in [23], [24].

The Eterna Dataset comes separated into a train set of 2400 molecules and a test set of 3634 molecules. The train set contains values for five different prediction targets (including reactivity, deg_Mg_pH10, deg_Mg_50C', 'deg_pH10', 'deg_50C') whereas the test set contains three of them (reactivity, deg_Mg_pH10, deg_Mg_50C'). For data visualization a line plot of the mean targets along all sequences in the training data can be seen on Fig.1. In this study we utilized the training set to perform model model HPO and training. The chosen models from the HPO process are trained on 80% of the train set and are validated on the other 20%. To evaluate each of the full models we utilized the test set from the Eterna Dataset. The data splits are available on the Kaggle link in Section VI along with our prepossessing code.

B. Graph neural networks models

The explored architectures are composed of two machine-learning stacks. A Graph Stack for creating node embeddings from the graph structured input and a Prediction Stack for getting output predictions. Furthermore, two variations of the Graph Stack are implemented using the models GCN and GAT, respectively. An overview of the

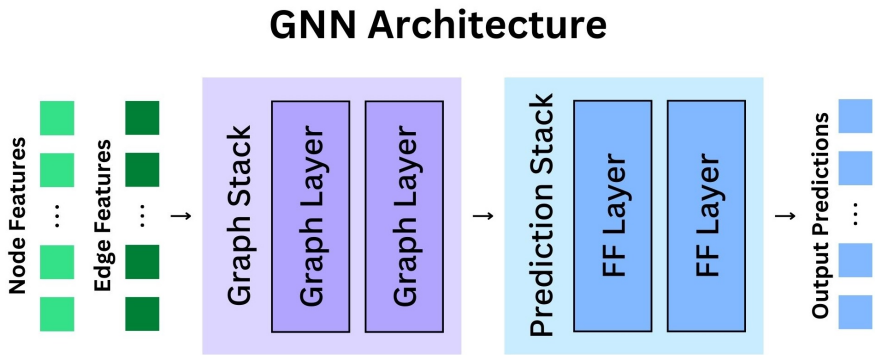

Fig. 2: High level overview of the GNN architecture tuned in the HPO experiments.

architecture can be found in Fig.2 and the hyperparameters of each part are listed in Table I. GNN models have been chosen to fit the problem inspired by other works that explored them [23], [24]. The suitability of the GNN architecture for the chosen data is beyond the scope of this study.

1) Graph Convolutional Network: Graph Convolutional Network (GCN) [9] emerged as an effective approach for semi-supervised learning on graph data. These models learn the features by performing convolution over neighboring nodes directly on the graph. Multiple layers of graph convolution can be added. Feed Forward layers have been added to get output node features. The various hyperparameters (graph layers, hidden size, dropout rate) chosen during HPO are detailed on Table I.

2) Graph Attention Network: The Graph Attention Network (GAT) [10] include attention [25] on top of the convolution. Similar to GCN multiple layers may be stacked on top of each other. Likewise, Feed Forward layers have been added to get output values. The hyperparameters explored in this study can be seen in Table I).

C. Optimization algorithms

The following optimization methods have been used in this study.

1) Random Search: Random Search is favored for hyperparameter optimization [2], and generally the most simple way to tackle this problem. It replaces the exhaustive enumeration of all combinations by selecting them randomly. Moreover, it is simple to apply to both discrete and continuous spaces. It performs better than Grid search [26] when only a few hyperparameters have an impact on the model performance. RS is a crucial benchmark for evaluating the effectiveness of new hyperparameter optimization techniques.

2) Bayesian Search: Another intuitive approach and a common choice for HPO is Bayesian Search [3]. It is a global optimization method for any noisy black-box functions. BS creates a probabilistic model that maps the values of the hyperparameters to the objective as determined by a validation set. By iteratively evaluating hyperparameter configurations, BS gathers observations about the optimum. This algorithm attempts to find the best solution by maintaining a balance between open exploration - hyperparameters for which the outcome is uncertain, and information exploitation - hyperparameters expected to be close to the optimum.

3) Hill Climbing: Hill Climbing [4], as a member of the family of local search algorithms, has an objective to minimize or maximize a target function by gradually updating the hyperparameter vector. The iterative process starts with a randomly chosen solution that is evaluated to find the direction of progress. Subsequently, in each iteration, HC adjusts a single hyperparameter and determines whether the change improves the performance. It differs from gradient descent methods, which adjust all of the values at each iteration.

4) Simulated Annealing: Simulated annealing [5] is a probabilistic technique for approximating the global optimum of a given function. Applied to hyperparameter tuning, SA is an iterative process which updates the solution while slowly adapting the exploration probability. Values for all hyperparameters are randomly chosen and an initial "temperature" is set. Alteration of the current state is done by updating one hyperparameter with a random value in the immediate neighborhood, with a certain probability based on the temperature. On every n^{th} step the temperature is decreased by a chosen rate. SA is preferred for discrete feature space problems when finding an approximate global optimum quickly is important.

5) Genetic Algorithm: The Genetic Algorithm [6] is an iterative, heuristic method inspired by the process of natural selection where the fittest individuals give rise to the following generation. In the context of HPO each individual is a proposed solution for the hyperparameter values. The population of solutions in each iteration is referred to as a generation. In each generation, the members of the population are evaluated in terms of their fitness, in this context the performance of the solution. The new generation is created by stochastically selecting the fittest solutions from the current population, recombining them, and introducing random mutations. The algorithm gradually generates better solutions by combining the best of each generation.

TABLE I: Summary table containing hyperparameters and explored ranges for optimization.

Configuration	Hyperparameter Name	Description	Hyperparameter Type	Explored Range
GCN Stack	**Graph Layers**	number of GCN layers	discrete	[1, 3]
	Hidden Channels	dimension of the hidden layers	discrete	[32, 128]
	Graph Dropout	dropout rate after GCN layers	continuous	(0.2, 0.5)
GAT Stack	**Graph Layers**	number of GCN layers	discrete	[1, 3]
	Hidden Channels	dimension of the hidden layers	discrete	[32, 128]
	Graph Dropout	dropout rate after GAT layers	continuous	(0.2, 0.5)
	Attention Dropout	attention dropout rate	continuous	(0.2, 0.5)
Prediction Stack	**Prediction Layers**	number of Feed Forward layers	discrete	[1, 3]
	Hidden Channels	dimension of the hidden layers	discrete	[32, 128]
	Prediction Dropout	dropout rate after Feed Forward layers	continuous	(0.2, 0.5)
Training	**Batch Size**	size of batches during training	discrete	[8, 32]
	Learning Rate	model learning rate	continuous	(0.00001, 0.01)
	Loss	loss function	categorical	[CrossEntropy, MSE, MAE]

6) Particle Swarm Optimization: Particle Swarm Optimization [7] is a well-known swarm intelligence technique, inspired by the navigation strategy of bird flocks in quest of food. The technique has been tested on a variety of high-dimensional real-world applications and has shown to be efficient and reliable [27], [28]. It starts with a random set of individual search 'particles', each representing a potential solution. The particles update states iteratively based on others in the swarm. PSO has an advantage in exploring a wide, multi-dimensional search space. While it cannot guarantee to find the global optimum solution it is likely to find a close-to-optimum solution in relatively few generations (iterations).

7) Artificial Bee Colony: Another swarm intelligence algorithm is Artificial Bee Colony [8], inspired by the foraging behavior of honey bee swarms. It contains three main components: employed bees, onlookers, and scouts. The position of the bees represents a set of hyperparameter values. The artificial bees cooperate to find better solutions using neighbouring search by choosing the best directions to move towards. By combining the gathered info, the population iteratively improves its performance.

IV. Experiments, results and discussion

A. Experimental details

The optimization algorithms are implemented to probe the hyperparameter space of two GNN architectures. A detailed list of hyperparameters tackled in the experiments can be found in Table I. The GNN architectures are implemented to perform multi-regression to predict five different numerical prediction targets. The predictions are performed on each node (nucleotide). Model performance is evaluated through the Mean Column-wise Root Mean Squared Error (MCRMSE) metric - following the Kaggle Competition [29] which uses this dataset. The optimization objective function is the training of the GNN configuration as given by the hyperparameters. The objective goal is minimizing the evaluation performance metric - MCRMSE.

In this experiment the GCN and GAT architectures have been optimized for 1h by each algorithm. In each optimization, model configurations were trained and validated for 5 epochs on a subset of 240 molecules with 107 nodes (nucleotides) and 5 targets per node, or a total of 25,680 labeled nodes. The optimal model configuration found by each algorithm was trained and validated on the full training set of 2400 molecules for 10 epochs. The developed models were additionally evaluated on the test set of 3634 molecules with 3 targets per node.

Due to the large differences in the internal process of each optimization method and following the recommendations of [13] the experiments are ran on a fixed time of 1 hour to assess the optimization performance fairly. The parameters of the HPO algorithms, also called meta-hyperparameters are left to default values. Exploration of the meta-hyperparameters is outside the scope of this study. The HPO algorithms are implemented with the NiaPy package [30], whereas the GNN models with PyTorch [31]. All experiments have been performed on an Intel(R) Core(TM) i7-10510U CPU with 1.80GHz and 16GB RAM. Additional details about the implementation can be found in Section VI.

B. Results and discussion

The results show that RS stands its ground as the most common HPO algorithm and manages to find the best performing model configuration for both GCN and GAT architectures on the HPO validation subset with MCRMSE of 0.55 and 0.60 respectively. Although, the fully trained models for RS do not hold the top performance on the train validation subset and are outperformed by ABC's model for GCN and HC's model for GAT. SA however, outperforms RS and all other algorithms for both GCN and GAT on the test set with Test MCMRSE of 2.86 and 3.05 respectively. Albeit the solutions found by the population based methods fall short compared to the simpler methods on the optimization set, PSO and GA offer the next best performances after SA on the evaluation set for GCN, while PSO comes close to SA on the evaluation set for GAT after RS.

By looking at the chosen hyperparameters from each algorithm, high heterogeneity in the values of the top architectures can be concluded (Fig. 3). Some hyperpa-

TABLE II: Summary table containing performance comparison between the HPO algorithms for both the GCN and GAT architectures. HPO MCRMSE is the top validation performance during the 1h HPO experiment. Train and Test MCRMSE are the validation and test performance respectively of the fully trained top architectures (best results with lowest MCRMSE).

HPO	GCN				GAT			
Algorithm	HPO MCRMSE	Train MCRMSE	Test MCRMSE	# Sol.	HPO MRCSE	Train MCRMSE	Test MCRMSE	# Sol.
RS	**0.552111207**	1.197424498	3.142875612	24	**0.606282744**	1.227286043	3.085664921	18
BS	0.663734097	1.397005282	3.73452739	24	0.701095777	1.237067464	3.607524834	17
HC	1.34278588	1.776945629	3.020120101	17	0.658518997	**0.754333032**	3.621118421	17
SA	0.594058272	1.568616626	**2.858786327**	28	1.19119597	1.19119597	**3.049026682**	18
GA	0.882215188	1.428922103	2.950592929	17	0.841650576	1.430237301	3.205829579	16
PSO	0.799015684	1.348786081	2.91169034	19	0.858330531	1.427626225	3.193924	19
ABC	0.751826537	**1.039043296**	3.686253486	12	0.716234768	2.027690987	3.159572088	18

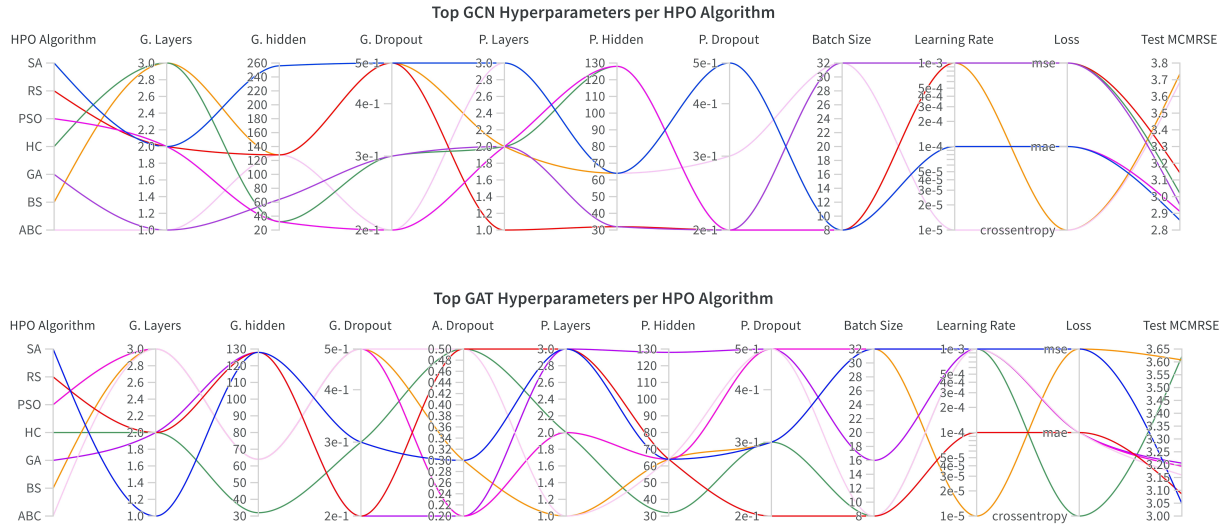

Fig. 3: Parallel coordinate plots showing the hyperparameter values chosen by each HPO algorithm during the 1h optimization experiment for a) the GCN and b) the GAT configurations, along with Test MCMRSE performance values.

rameters values as 2 for graph layers, a batch size of 8, learning rate of 0.001 or the MAE loss type are more dominant and agreed upon by most algorithms for both architectures. Other hyperparameter values for GCN with a majority vote are 0.5 for graph dropout, 2 for prediction layers, 32 for prediction hidden channels and 0.2 for prediction dropout, while there is no consensus on the choice of graph hidden channels. For the GAT architecture the algorithms generally agree more on the chosen values with a consensus on 128 for graph hidden channels, 0.2 for graph dropout, 0.5 for attention dropout, 3 for prediction layers, 64 for prediction hidden channels and 0.5 for prediction dropout. Additionally, the analysis of the optimization runs performance at each step during the 1h tuning shows there is high variation between performances of consecutive steps especially for the GAT. This points to a high level of complexity in the search space and large performance differences for small parameter changes.

Overall, the results suggest that SA might be a good choice for HPO on GNN in this application and also shows promising results for population based algorithms. The present findings confirm the state-of-the-art usage of these methods in line with the ideas of [16], [17], [18], [19], [20]. A summary of the results can be seen in the performance comparison table II. More details on the implementation and performance of each HPO algorithm separately can be seen in Section VI.

V. Conclusion

The objective of this study was to make an experimental comparison of HPO strategies for Graph Neural Networks implemented in an mRNA node-wise prediction problem. The performance analysis has shown promising results for HPO algorithms such as the SA algorithm as well as population based algorithms like PSO, which performed better than RS at choosing GNN hyperparameters. The SA algorithm found the best-performing GAT and GCN architectures, while being highly efficient and exploring the largest amount of configurations.

The population algorithms including GA, PSO and ABC show promising results and might perform even better given a bigger population size. It must be noted that this type of algorithms give significant memory complexity overhead and therefore had to be limited in population

size. The overhead also explains the smaller amount of configurations explored by them.

This survey offers a practical comparison of multiple algorithms (RS, BS, HC, SA, GA, PSO, ABC) with focus on the HPO of GNNs specifically for prediction on mRNA molecules. We attempted to cover a wide range of hyperparameters and problem agnostic algorithms in hopes of facilitating other areas as well. A more extensive exploration involving diverse meta-hyperparameter choices is needed for a more informed comparison. The overall effectiveness of the GNN architecture itself for the chosen task is a topic for another study. Additionally, this survey needs to be further expanded to other problems and datasets to get a more holistic review of the effectiveness of the HPO algorithms. Future work involves the aim to deepen the exploration of HPO algorithms on GNN architectures for mRNA modelling in order to support researchers in this and other related interdisciplinary fields.

VI. Supplementary Material

The code for all the experiments, GNN models and HPO algorithms, as well as all additional results, are available on the repository https://github.com/ViktorijaVodilovska/mRNA_pred. The data used in this study is available on https://www.kaggle.com/c/stanford-covid-vaccine/overview.

Acknowledgment

This work was partially financed by the Faculty of Computer Science and Engineering at the Ss. Cyril and Methodius University in Skopje.

References

[1] X.-M. Zhang, L. Liang, L. Liu, and M.-J. Tang, "Graph neural networks and their current applications in bioinformatics," *Frontiers in genetics*, vol. 12, p. 690049, 2021.

[2] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization." *Journal of machine learning research*, vol. 13, no. 2, 2012.

[3] M. Pelikan, D. E. Goldberg, E. Cantú-Paz *et al.*, "Boa: The bayesian optimization algorithm," in *Proceedings of the genetic and evolutionary computation conference GECCO-99*, vol. 1. Citeseer, 1999, pp. 525–532.

[4] B. Selman and C. P. Gomes, "Hill-climbing search," *Encyclopedia of cognitive science*, vol. 81, p. 82, 2006.

[5] M. Fischetti and M. Stringher, "Embedded hyper-parameter tuning by simulated annealing," *arXiv preprint arXiv:1906.01504*, 2019.

[6] K. Sastry, D. Goldberg, and G. Kendall, "Genetic algorithms," in *Search methodologies*. Springer, 2005, pp. 97–125.

[7] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95-international conference on neural networks*, vol. 4. IEEE, 1995, pp. 1942–1948.

[8] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," 2005.

[9] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[10] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[11] J. Lee, W. Kladwang, M. Lee, D. Cantu, M. Azizyan, H. Kim, A. Limpaecher, S. Gaikwad, S. Yoon, A. Treuille, R. Das, and E. Participants, "Rna design rules from a massive open laboratory," *Proceedings of the National Academy of Sciences*, vol. 111, no. 6, pp. 2122–2127, 2014. [Online]. Available: https://www.pnas.org/content/111/6/2122

[12] R. Lorenz, S. H. Bernhart, C. Höner zu Siederdissen, H. Tafer, C. Flamm, P. F. Stadler, and I. L. Hofacker, "Viennarna package 2.0," *Algorithms for molecular biology*, vol. 6, pp. 1–14, 2011.

[13] Y. Yuan, W. Wang, and W. Pang, "A systematic comparison study on hyperparameter optimisation of graph neural networks for molecular property prediction," in *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, jun 2021. [Online]. Available: https://doi.org/10.1145%2F3449639.3459370

[14] B. Ramsundar, P. Eastman, P. Walters, V. Pande, K. Leswing, and Z. Wu, *Deep Learning for the Life Sciences*. O'Reilly Media, 2019, https://www.amazon.com/Deep-Learning-Life-Sciences-Microscopy/dp/1492039837.

[15] Z. Wu, B. Ramsundar, E. N. Feinberg, J. Gomes, C. Geniesse, A. S. Pappu, K. Leswing, and V. Pande, "Moleculenet: a benchmark for molecular machine learning," *Chemical science*, vol. 9, no. 2, pp. 513–530, 2018.

[16] P. Balaprakash, M. Salim, T. D. Uram, V. Vishwanath, and S. M. Wild, "Deephyper: Asynchronous hyperparameter search for deep neural networks," in *2018 IEEE 25th International Conference on High Performance Computing (HiPC)*, 2018, pp. 42–51.

[17] Y. Liu, J. Liu, and Y. Li, "Automatic search of architecture and hyperparameters of graph convolutional networks for node classification," *Applied Intelligence*, pp. 1–16, 2022.

[18] C. White, W. Neiswanger, and Y. Savani, "Bananas: Bayesian optimization with neural architectures for neural architecture search," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 12, 2021, pp. 10 293–10 301.

[19] Y. Guo, J.-Y. Li, and Z.-H. Zhan, "Efficient hyperparameter optimization for convolution neural networks in deep learning: A distributed particle swarm optimization approach," *Cybernetics and Systems*, vol. 52, no. 1, pp. 36–57, 2020.

[20] P. Singh, S. Chaudhury, and B. K. Panigrahi, "Hybrid mpso-cnn: Multi-level particle swarm optimized hyperparameters of convolutional neural network," *Swarm and Evolutionary Computation*, vol. 63, p. 100863, 2021.

[21] K. Zhou, Q. Song, X. Huang, and X. Hu, "Auto-gnn: Neural architecture search of graph neural networks," 2019.

[22] W. Zhang, Z. Lin, Y. Shen, Y. Li, Z. Yang, and B. Cui, "Dfg-nas: Deep and flexible graph neural architecture search," 2022.

[23] A. Singhal, "Predicting hydroxyl mediated nucleophilic degradation and molecular stability of rna sequences through the application of deep learning methods," 2021.

[24] A. Muneer, S. Fati, N. Akbar, D. Agustriawan, and S. Tri Wahyudi, "ivaccine-deep: Prediction of covid-19 mrna vaccine degradation using deep learning," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, pp. pp. 7419–7432, 10 2021.

[25] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[26] P. Liashchynskyi and P. Liashchynskyi, "Grid search, random search, genetic algorithm: a big comparison for nas," *arXiv preprint arXiv:1912.06059*, 2019.

[27] M. Neethling and A. P. Engelbrecht, "Determining rna secondary structure using set-based particle swarm optimization," in *2006 IEEE International Conference on Evolutionary Computation*. IEEE, 2006, pp. 1670–1677.

[28] A. A. Esmin, G. Lambert-Torres, and A. Z. De Souza, "A hybrid particle swarm optimization applied to loss power minimization," *IEEE Transactions on power systems*, vol. 20, no. 2, pp. 859–866, 2005.

[29] "Openvaccine: Covid-19 mrna vaccine degradation prediction." [Online]. Available: https://www.kaggle.com/competitions/stanford-covid-vaccine/overview

[30] G. Vrbančič, L. Brezočnik, U. Mlakar, D. Fister, and I. Fister Jr., "NiaPy: Python microframework for building nature-inspired algorithms," 10 2018. [Online]. Available: https://github.com/NiaOrg/NiaPy

[31] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035.