

Automating Genetic Algorithm Mutations For Molecules Using a Masked Language Model

Andrew E. Blanchard, Mayanka Chandra Shekar, Shang Gao, John Gounley, Isaac Lyngaas, Jens Glaser, Debsindhu Bhowmik

Abstract—Inspired by the evolution of biological systems, genetic algorithms have been applied to generate solutions for optimization problems in a variety of scientific and engineering disciplines. For a given problem, a suitable genome representation must be defined along with a mutation operator to generate subsequent generations. Unlike natural systems which display a variety of complex rearrangements (e.g. mobile genetic elements), mutation for genetic algorithms commonly utilizes only random point-wise changes. Furthermore, generalizing beyond point-wise mutations poses a key difficulty as useful genome rearrangements depend on the representation and problem domain. To move beyond the limitations of manually defined point-wise changes, here we propose the use of techniques from masked language models to automatically generate mutations. As a first step, common subsequences within a given population are used to generate a vocabulary. The vocabulary is then used to tokenize each genome. A masked language model is trained on the tokenized data in order to generate possible rearrangements (i.e. mutations). In order to illustrate the proposed strategy, we use string representations of molecules and use a genetic algorithm to optimize for drug-likeness and synthesizability. Our results show that moving beyond random point-wise mutations accelerates genetic algorithm optimization.

I. INTRODUCTION

Built upon the principles of mutation and selection observed in natural systems, genetic algorithms provide a useful optimization method for a variety of problems across multiple scientific and engineering disciplines [1]–[4]. One key advantage of genetic algorithms is the flexibility to choose a problem specific mutation operator and optimization objective, without the need for differentiability. For example, in novelty search, subsequent generated are selected based on the distance of current candidates from previously generated solutions [5]. The flexibility of genetic algorithms, however, can also make applications in new scientific domains difficult, as an appropriate genome representation, mutation operator, and fitness

This manuscript has been authored by UT-Battelle LLC under Contract No. DE-AC05-00OR22725 with the US Department of Energy (DOE). The US government retains and the publisher, by accepting the article for publication, acknowledges that the US government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of the manuscript, or allow others to do so, for US government purposes. DOE will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (<http://energy.gov/downloads/doe-public-access-plan>).

A. E. Blanchard, M. Chandra Shekar, S. Gao, J. Gounley, D. Bhowmik are with the Computational Sciences and Engineering Division, Oak Ridge National Laboratory, Oak Ridge, TN 37830 USA (e-mail: blanchardae, chandrashekm, gaos, gounleyjp, bhowmikd@ornl.gov).

I. Lyngaas and J. Glaser are with the National Center for Computational Sciences, Oak Ridge National Laboratory, Oak Ridge, TN 37830 USA (e-mail: lyngaasir and glaserj@ornl.gov).

objective must be determined based on research intuition or domain expertise [3], [6].

One area where genetic algorithms have seen notable success is computer-aided drug design [2], [7]–[9]. Previous work has utilized hand-crafted mutation rules (e.g. adding an atom, switching an atom type) coupled with selection based on diversity to explore and characterize chemical space [7]. More recently, multiple investigations have shown that genetic algorithms based on different molecule representations can achieve state-of-the-art results for molecule generation tasks, even outperforming current machine learning techniques [8], [9]. Furthermore, intermediate generations during optimization can be tracked and analyzed to better understand beneficial changes for complex optimization metrics.

Despite the many advantages of utilizing genetic algorithms for molecule optimization, the determination of hand-crafted rules for mutation still provides a major difficulty. For example, the appropriate ratio for different mutation types (e.g. change atom type, create a ring, etc...) must be determined for a given optimization function [7]. Furthermore, for manual rules, mutations are often restricted to single atom changes [7], [8], excluding the benefits and diversity produced by larger subsequence rearrangements. Although researcher expertise may generate reasonable mutation operators, in principle hand-crafted rules do not generalize well to new problems or domains.

In order to address the key challenges of automating the mutation operator and extending mutations to utilize larger subsequences, we take inspiration from advances in natural language processing (NLP). Recent NLP models (e.g. BERT) have relied upon tokenization and mask prediction to leverage large amounts of unlabeled text data for training [10]. The process of tokenization is necessary to construct a vocab of fixed size to capture all possible words encountered by the model. One popular method, WordPiece tokenization [11], [12], builds a vocabulary from each character encountered. Then, commonly occurring subsequences (i.e. greater than length 1) are added to the vocabulary until a specified size is reached. Notice that a token in the vocabulary may represent multiple characters.

After determining the vocabulary, input text sequences in the training data are tokenized. To train the model, random tokens from a given text sequence are masked (i.e. replaced by a mask token), and the training loss is determined by how well the model correctly reproduces the original text sequence from the masked sequence [10]. After training on a large unsupervised dataset, the model is then typically trained on smaller fine-tuning tasks for evaluation on specific applications [10], [13],

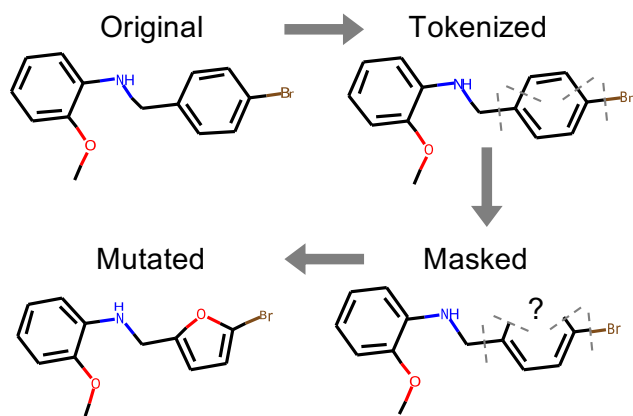


Fig. 1. Strategy to use a masked language model to mutate molecule sequences. First, a given molecule is tokenized (i.e. split into commonly occurring subsequences). Second, certain tokens are masked. Finally, the masked language model ranks possible replacements for the mask, providing possible mutations.

[14]. For practical NLP models, additional tasks beyond mask prediction may be incorporated into the training as well [10], [13], [14].

Here, we present a strategy to incorporate tokenization and mask prediction into genetic algorithms to move beyond random point-wise mutations (Figure 1). Tokenization determines common subsequences that are then represented as a single token by the model. During mask prediction training, the model learns viable combinations of tokens based on context, enabling the ranking of possible mutations. Tokenization and model mask prediction are thus combined to form an automated mutation operator.

To test our approach to automate the mutation operator and incorporate longer subsequence rearrangements, we consider drug molecule string (i.e. smiles) optimization as a use case. We utilize a large compound library [15] as training data for the tokenization and mask prediction model training. To isolate the impacts of tokenization and mask prediction on optimization performance, we generate vocabularies with different constraints. As a control, we consider a standard vocabulary for smiles strings that represents individual atoms. We then include vocabularies that allow longer subsequences to be represented as a single token. As a final test, we utilize iterative training of a mask prediction model without the use of a large compound library. Our results show that enabling subsequence rearrangement beyond single characters in an automated mutation operator improves performance of a genetic algorithm for molecule optimization.

II. RELATED WORK

Genetic algorithms have been used in multiple ways to generate drug-like molecules. A systematic search of chemical space was performed using hand-crafted rules for mutation and recombination [7]. Molecules were selected based on a diversity criteria during generations of the search. Other studies have utilized graph-based or smiles-based representations of molecules to optimize a given drug-related metrics [6], [8].

In each investigation, a set of hand-crafted rules were determined and applied for molecule mutations. Comparisons with alternative optimization techniques have shown that genetic algorithm-based optimization performs well favorably across a range of molecule generation tasks [9].

Several variations of ML models have been proposed for molecule generation tasks, including generative adversarial networks and recurrent neural networks [16]–[18]. Similar to studies with genetic algorithms, multiple representations of molecules have been used for training data, including graphs and smiles strings [16], [17]. For models that rely on a string representation, typically a given molecule is tokenized per character (or per atom) [16], [19]–[21]. No investigation has been done into the performance of text based models allowing different subsequence lengths for the vocabulary.

Since the introduction of BERT [10], several investigations have been made in training a transformer model on molecule data [22]–[27]. Similar to text applications, the trained model was applied on fine-tuning tasks concerning chemical property prediction [22]–[25]. Recent work has utilized a trained language model for molecule generation and optimization [24]. Transformer-based model have also been utilized for chemical reaction prediction [26], [27]. However, an investigation into the impact of tokenization on optimization performance has not been performed. Furthermore, the use of a language model to automate mutation within a genetic algorithm has not been fully investigated. For text-based applications, previous studies utilized mask prediction to generate adversarial examples [28], [29]. Although not discussed in relation to genetic algorithms, optimizing text sequences through mask replacement is similar to our proposed strategy.

III. METHODS

A. Training Dataset

To train the tokenizer, 1.2 billion smiles string were used from the Enamine *REAL* database [15]. A subset of 120 million smiles from the full dataset were used to train the mask prediction model. All smiles were converted to canonical form using rdkit [30] before being used in training. For iterative training, only the population produced by the genetic algorithm (100k molecules) was used for model and tokenizer training.

B. Tokenizer and Model Training

The tokenizer and transformer libraries provided by Hugging Face [31] were used to train all WordPiece tokenizers. For a given tokenizer, a pre-tokenizer performs initial splits of an input sequence. Therefore, we used different pre-tokenizers to enforce constraints. We performed splitting based off of previously used regex (Regex) [19], punctuation and digits (BERT + Digits), punctuation only (BERT), and digits only (Digits). We denote the punctuation-based splitting as BERT because this is the default behavior for the WordPiece tokenizer provided by Hugging Face for the BERT model. The maximum tokenizer vocab size was set to 30k.

All model training was done using the tokenizer and transformer libraries provided by Hugging Face [31] and DeepSpeed [32]. The learning rate was set to 5×10^{-5} and the

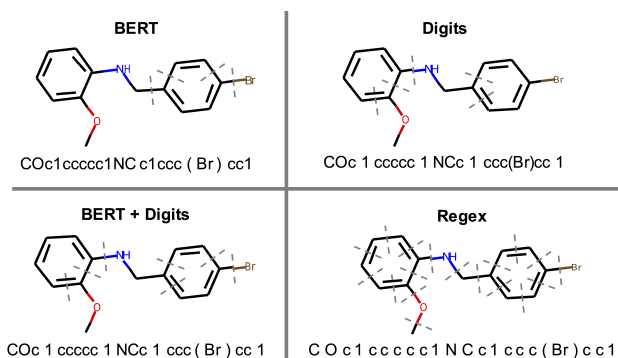


Fig. 2. Different pre-tokenization schemes for smiles strings. Spaces in smiles strings represent splits, which correspond to dashed lines in molecule images. **A** Standard pre-tokenizer for BERT splits on punctuation before training. **B** Pre-tokenizer that splits on digits. **C** Combining BERT pre-tokenizer with splits on digits. **D** Pre-tokenized based on regex for individual atoms.

batch size was set to 128 using the AdamW optimizer [33]. The default BERT model architecture as provided by Hugging Face was used for all models (i.e. hidden size of 768, 12 attention heads and 12 hidden layers). Models trained on a subset of the Enamine dataset were trained for 4 epochs. Models in iterative training used the same parameters with the exception of a slightly smaller learning rate (3×10^{-5}) and larger number of epochs (10).

C. Genetic Algorithm

We utilized a very simple genetic algorithm in which half of the population was sampled for mutation during each generation. After mutation, the original population was combined with the mutated samples and sorted by fitness. We defined fitness as the harmonic mean of the quantitative estimation of drug-likeness score [34] and normalized synthesizability [17], [35]. Similar to previous work [17], synthesizability was normalized between 0 and 1. The top scoring unique members were retained in order to keep a fixed overall population size.

For a given mutation, a random integer was generated between 1 and 5 to represent the number of masks to introduce. We considered three different mutation types (insertion, replacement, deletion). For insertion, masks were inserted between existing tokens or at the beginning or end of the sequence. For replacement, masks replaced sampled tokens. For deletion, the token following a masked replacement was removed. The masks were subsequently replaced either randomly or guided by a mask prediction model to complete the mutation. A total of five candidate replacements were generated for each masked sequence. The type of mutation was selected randomly for population samples in batches of size 10.

IV. RESULTS

A. Random Mutations

To determine the impact of subsequence mutations on genetic algorithm performance, we began by generating different vocabularies using WordPiece tokenization [11], [12] with different pre-tokenization splitting. As shown in Figure 2,

different tokenizers generate different representations of a given molecule. The standard BERT tokenizer splits on punctuation (e.g. parenthesis) before determining commonly occurring subsequences for the vocabulary. Similarly, we can split smiles strings based on digits, punctuation and digits (i.e. BERT+Digits), or a custom regex [19] used in previous work with ML models for smiles. Notice that as we increase the splitting for the molecule, the number of tokens used to represent it by the model increases.

For a given tokenizer and associated vocabulary, we apply a simple genetic algorithm for optimization. First, we apply the tokenizer to the input smiles string. A random number of masks (up to 5) are added which represent possible mutations. Then, the mask is replaced with a randomly chosen token from the vocabulary to produce a mutated sequence. Five candidate replacements are generated for each masked sequence. Mask replacement may occur in three different modes (insert, replace, delete). For insertion, the mask is inserted between two tokens from the original string. For replacement, the mask replaces a token from the original string. For deletion, the mask replaces a token from the original string and the subsequent token is removed.

To assess the performance of different tokenization schemes, we tracked the number of valid and novel molecules along with the fitness of the population. Here, we defined fitness as the harmonic mean of the quantitative estimation of drug-likeness score [34] and normalized synthesizability [17], [35]. The starting population of molecules was taken from the first 1000 molecules of the gdb9 dataset [36]. Each generation, 500 molecules were randomly selected and mutation was applied to generate the child population. The top 1000 unique molecules from parent and child populations were retained for the next generation.

As shown in Figure 3, the different tokenization schemes lead to large differences in the number of valid and novel molecules produced. The schemes with smaller vocabularies (i.e. more splitting during pre-tokenization), produce the most valid and novel molecules when averaging over multiple runs. Intuitively, the decrease in produced molecules for the schemes with larger vocabs (i.e. less splitting) is expected, as random replacement of a larger subsequence of the molecule can easily produce an invalid smiles string.

Contrary to the number of molecules produced, BERT tokenization leads to a notable increase in the fitness for the population. The population is able to more quickly explore favorable areas of parameter space (e.g. longer length molecules as shown in Figure 3) through subsequence rearrangements. Single mutations can result in the addition of entire functional group (e.g. benzene ring) rather than requiring several sequential beneficial mutations.

B. Mask Prediction

Optimization runs with random mutations show that the choice of tokenization scheme can indeed impact optimization both in terms of fitness and novel molecules. However, tokenization is typically only a preliminary step in training a masked language model. Once the vocabulary is determined, the model for mask prediction is developed using an

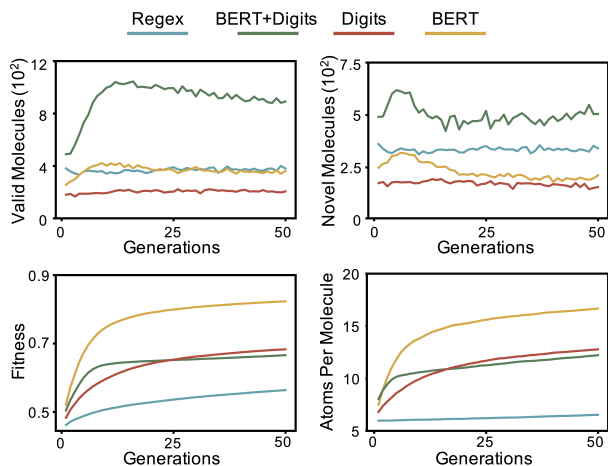


Fig. 3. Tracking the number of valid molecules, number of novel molecules, fitness, and number of atoms per molecules during optimization. The choice of tokenization scheme results in different performance outcomes for random mutations. Plots show the average over five runs.

unsupervised approach where tokens are randomly masked during training [10]. Here, we train a model based off of each tokenization scheme using a dataset of 120 million smiles strings. We then use the model to generate candidates for mask replacement in contrast to random replacement.

As shown in Figure 4, the mask prediction model results in similar performance across tokenization schemes for the number of valid molecules produced. The regex tokenizer produces the largest number of novel molecules, however, it suffers a degradation in fitness compared to the other schemes. The fitness plot shows that tokenizers with more complex vocabularies (i.e. BERT and Digits) yield the best performance, similar to the results from random mutations.

In addition to mask prediction with a single tokenization scheme, we also tested optimization based off of two different schemes. For combinations, the mutation samples were split evenly between each scheme, with the total number of samples fixed. As shown in Table I, the combination of BERT and Digits generated the best population fitness. The complimentary nature of the two tokenization schemes is interesting due to the different ways molecules are split. BERT splits based on punctuation, corresponding to branches in smiles molecules, while Digits splits on integers which represent rings. Notice that the use of two tokenization schemes (e.g. a BERT tokenizer and a Digits tokenizer) is different than a single tokenizer (e.g. BERT+Digits) that combines the two schemes. Using two different tokenization schemes results in two distinct models that have different representations for a given molecule. Constructing a single tokenizer by combining the splitting rules for two schemes results in a single representation with shorter subsequences in tokenization than the parent schemes.

Although the improvements to fitness optimization are promising, training of both the tokenizer and mask prediction model relied on a large dataset of viable molecules. For other applications, such a training set may not exist, necessitating

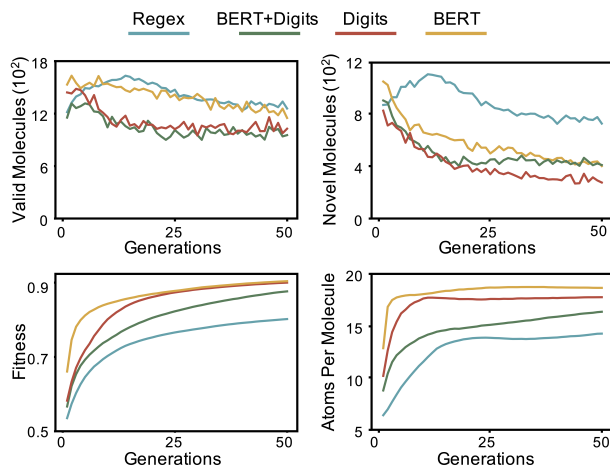


Fig. 4. Tracking the number of valid molecules, number of novel molecules, fitness, and number of atoms per molecules during optimization. The choice of tokenization scheme results in different performance outcomes for mutations selected by the masked language model. Plots show the average over five runs.

TABLE I
FINAL FITNESS SCORES FOR DIFFERENT TOKENIZATION SCHEMES WITH A MASK PREDICTION MODEL. SCORES ARE THE AVERAGE OF FIVE RUNS.

Tokenizer 1	Tokenizer 2	Fitness
BERT	Digits	0.9482
BERT	BERT+Digits	0.9148
BERT+Digits	Digits	0.9107
BERT	Regex	0.9100
BERT	BERT	0.9063
Digits	Digits	0.9017
Regex	Digits	0.8975
BERT+Digits	BERT+Digits	0.8786
Regex	BERT+Digits	0.8679
Regex	Regex	0.8031

the use of random point-wise mutations. Utilizing the top performing combination of tokenization schemes and models, we now consider a possible solution for applications lacking initial training data.

C. Iterative Training

In the absence of initial training data, a masked language model cannot be trained to tokenize molecules and predict possible mutations. We can, however, utilize the vocabulary for molecules as provided by the regex [19] along with random mask replacement. Rounds of mutations (including replacements, insertions, and deletions) are then sufficient to generate new candidates (i.e. molecules). We begin with a single sample in the population, which is a single carbon atom. After several generations, a population of molecules is generated that can be used to train a tokenizer and associated mask prediction model. Iterations of this strategy (i.e. generate molecules, train a tokenizer and mask prediction model, repeat) can be performed for optimization. The use of iterative training allows the tokenizer and mask prediction model to capture changes

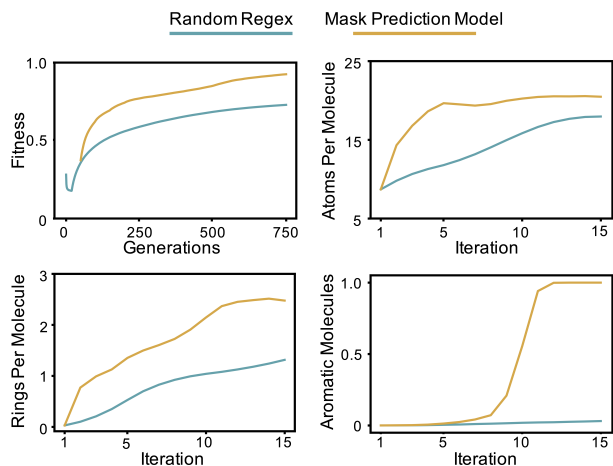


Fig. 5. Tracking the fitness, number of atoms per molecule, number of rings per molecule, and fraction of aromatic molecules in the population for each iteration. In addition to an increase in fitness, adding tokenization and mask prediction leads to larger molecules with more rings and an increase in aromatic molecules compared to random mutations with a regex tokenizer. The initial decrease in fitness for Random Regex occurs as the population is being filled (i.e. any molecule with positive fitness is accepted until the population reaches the maximum size).

in the population as the fitness increases. Notice that for cases with large or representative initial datasets an iterative approach may not be necessary, as the masked language model can determine useful patterns from the training data.

For the iterations, we utilize 50 generations with a population size of 10^5 molecules. In each generation, 5×10^4 molecules are sampled for mutation. As a control for comparison, we use the regex tokenizer with random mutations for 15 total iterations. We also consider the addition of model mutations with a BERT tokenizer and a Digits tokenizer, which displayed the best fitness optimization performance (Table I). The total mutations are split evenly between regex random mutations and the mask prediction model mutations. The population after each iteration is used to train tokenizers (and models) for use in the next iteration.

As shown in Figure 5, the addition of trained tokenizers and mask prediction models substantially improves the fitness optimization for the population. To track the benefits of subsequence rearrangements, we track the length, number of rings, and number of aromatic compounds in the population. It is important to note that an aromatic ring would require multiple random mutations to be generated, as aromatic atoms are denoted by lowercase in smiles. As shown in Figure 5, both strategies are capable of generating aromatic compounds, however, the BERT mask prediction model can incorporate the aromatic ring as a single token allowing aromatic rings to proliferate in the population.

V. DISCUSSION

Evolution in natural systems (e.g. microbial populations) proceeds through a complex series of genetic alterations. Point-wise mutations, including replacements, insertions, and deletions, take place alongside larger subsequence changes induced by mobile genetic elements [37]. Mutation coupled with

selection enables (in part) microbial adaptation and survival in harsh environments (e.g. antibiotic resistance) [37]. Taking inspiration from natural systems, the current work shows that the addition of subsequence rearrangements to typical point-wise mutations in genetic algorithms provides a substantial boost to optimization performance.

The use of tokenization and a mask prediction model does result in certain trade-offs for population optimization. The tokenization scheme and mask prediction model are trained on a specific population, which biases future mutations towards currently occurring subsequences. Therefore, a balance between random point-wise mutations for exploration and subsequence rearrangement for fitness optimization is necessary. Fortunately, a range of tokenization schemes and models can be used to balance the need for novelty with fitness optimization. Furthermore, objective functions that maximize novelty [5], [38] can be utilized for future investigations.

In this work, we have utilized a very simple genetic algorithm, with random sampling for mutation and selection with a fixed population size. The simple form for updating the population was chosen to isolate the impacts of different mutation operators on performance. Mask prediction, however, can be incorporated alongside many standard improvements and additions commonly used in genetic algorithms, such as recombination and elitism [1]. Furthermore, we have focused on optimizing molecules, represented as smiles strings, for a specific metric. Mask prediction models, heavily utilized for text processing tasks [10], can be utilized in any domain where a text-based representation of candidates exists.

VI. CONCLUSION

Genetic algorithms provide a useful optimization technique inspired by the capability of natural systems to adapt and survive in a range of environments. The application of genetic algorithms to a given problem, however, require the definition of a suitable mutation operator. For mutations, simple random point-wise changes are typically used to avoid difficulties with determining useful subsequence rearrangements. Here, we have presented a strategy to generalize and automate genetic algorithm mutations utilizing tokenization and mask prediction. Our approach generated a substantial increase in fitness for a sample optimization problem towards generating drug-like molecules. Furthermore, our approach can be easily generalized to any problem with a text-based representation.

ACKNOWLEDGMENT

This research was supported by the Exascale Computing Project (17-SC-20-SC), a joint project of the U.S. Department of Energy’s Office of Science and National Nuclear Security Administration. This work has been supported in part by the Joint Design of Advanced Computing Solutions for Cancer (JDACS4C) program established by DOE and the NCI of the National Institutes of Health. This work was performed under the auspices of DOE by Argonne National Laboratory under Contract DE-AC02-06-CH11357, Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344, Los Alamos National Laboratory under Contract

DE-AC5206NA25396, and Oak Ridge National Laboratory under Contract DE-AC05-00OR22725. This research used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725. This work was supported in part by DOE CARES emergency funding to the National Center for Computational Sciences at ORNL through the Advanced Scientific Computing Research (ASCR) program.

REFERENCES

- [1] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*, 2nd ed. Springer-Verlag GmbH Germany: Springer Publishing Company, Incorporated, 2015.
- [2] N. Brown, B. McKay, F. Gilardoni, and J. Gasteiger, "A graph-based genetic algorithm and its application to the multiobjective evolution of median molecules," *Journal of Chemical Information and Computer Sciences*, vol. 44, no. 3, pp. 1079–1087, 2004.
- [3] G. S. Hornby, J. D. Lohn, and D. S. Linden, "Computer-Automated Evolution of an X-Band Antenna for NASA's Space Technology 5 Mission," *Evolutionary Computation*, vol. 19, no. 1, pp. 1–23, 2011.
- [4] G. Morse and K. O. Stanley, "Simple evolutionary optimization can rival stochastic gradient descent in neural networks," in *GECCO 2016 - Proceedings of the 2016 Genetic and Evolutionary Computation Conference*, no. Gecco, 2016, pp. 477–484.
- [5] J. Lehman and K. O. Stanley, "Abandoning objectives: Evolution through the search for novelty alone," *Evolutionary Computation*, vol. 19, no. 2, pp. 189–222, 2011.
- [6] N. Yoshikawa, K. Terayama, M. Sumita, T. Homma, K. Oono, and K. Tsuda, "Population-based De Novo Molecule Generation, Using Grammatical Evolution," *Chemistry Letters*, vol. 47, no. 11, pp. 1431–1434, 2018.
- [7] A. M. Virshup, J. Contreras-García, P. Wipf, W. Yang, and D. N. Beratan, "Stochastic voyages into uncharted chemical space produce a representative library of all possible drug-like compounds," *Journal of the American Chemical Society*, vol. 135, no. 19, pp. 7296–7303, 2013.
- [8] J. H. Jensen, "A graph-based genetic algorithm and generative model/Monte Carlo tree search for the exploration of chemical space," *Chemical Science*, vol. 10, no. 12, pp. 3567–3572, 2019.
- [9] N. Brown, M. Fiscato, M. H. Segler, and A. C. Vaucher, "GuacaMol: Benchmarking Models for de Novo Molecular Design," *Journal of Chemical Information and Modeling*, vol. 59, no. 3, pp. 1096–1108, 2019.
- [10] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, vol. 1, no. M1m, pp. 4171–4186, 2019.
- [11] M. Schuster and K. Nakajima, "Japanese and korean voice search," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012, pp. 5149–5152.
- [12] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, Ł. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean, "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation," pp. 1–23, 2016. [Online]. Available: <http://arxiv.org/abs/1609.08144>
- [13] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "RoBERTa: A Robustly Optimized BERT Pretraining Approach," no. 1, 2019. [Online]. Available: <http://arxiv.org/abs/1907.11692>
- [14] Y. Gu, R. Tinn, H. Cheng, M. Lucas, N. Usuyama, X. Liu, T. Naumann, J. Gao, and H. Poon, "Domain-specific language model pretraining for biomedical natural language processing," *arXiv*, pp. 1–24, 2020.
- [15] "Enamine REAL Database," <https://enamine.net/compound-collections/real-compounds/real-database>, accessed: 2021-05-28.
- [16] M. H. Segler, T. Kogej, C. Tyrchan, and M. P. Waller, "Generating focused molecule libraries for drug discovery with recurrent neural networks," *ACS Central Science*, vol. 4, no. 1, pp. 120–131, 2018.
- [17] N. De Cao and T. Kipf, "MolGAN: An implicit generative model for small molecular graphs," *ICML 2018 workshop on Theoretical Foundations and Applications of Deep Generative Models*, 2018.
- [18] A. E. Blanchard, C. Stanley, and D. Bhowmik, "Using GANs with adaptive training data to search for new molecules," *Journal of Cheminformatics*, vol. 13, no. 1, pp. 4–11, 2021. [Online]. Available: <https://doi.org/10.1186/s13321-021-00494-3>
- [19] P. Schwaller, T. Gaudin, D. Lányi, C. Bekas, and T. Laino, "'Found in Translation': predicting outcomes of complex organic chemistry reactions using neural sequence-to-sequence models," *Chemical Science*, vol. 9, no. 28, pp. 6091–6098, 2018.
- [20] J. Arús-Pous, S. V. Johansson, O. Prykhodko, E. J. Bjerrum, C. Tyrchan, J. L. Reymond, H. Chen, and O. Engkvist, "Randomized SMILES strings improve the quality of molecular generative models," *Journal of Cheminformatics*, vol. 11, no. 1, pp. 1–13, 2019. [Online]. Available: <https://doi.org/10.1186/s13321-019-0393-0>
- [21] F. Grisoni, M. Moret, R. Lingwood, and G. Schneider, "Bidirectional Molecule Generation with Recurrent Neural Networks," *Journal of Chemical Information and Modeling*, vol. 60, no. 3, pp. 1175–1183, 2020.
- [22] S. Wang, Y. Guo, Y. Wang, H. Sun, and J. Huang, "Smiles-Bert: Large scale unsupervised pre-training for molecular property prediction," *ACM-BCB 2019 - Proceedings of the 10th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics*, pp. 429–436, 2019.
- [23] S. Chithrananda, G. Grand, and B. Ramsundar, "ChemBERTa: Large-Scale Self-Supervised Pretraining for Molecular Property Prediction," no. NeurIPS, 2020. [Online]. Available: <http://arxiv.org/abs/2010.09885>
- [24] D. Xue, H. Zhang, D. Xiao, Y. Gong, G. Chuai, Y. Sun, H. Tian, H. Wu, Y. Li, and Q. Liu, "X-MOL: large-scale pre-training for molecular understanding and diverse molecular analysis," *bioRxiv*, 2020.
- [25] S. Honda, S. Shi, and H. R. Ueda, "Smiles transformer: Pre-trained molecular fingerprint for low data drug discovery," *arXiv*, 2019.
- [26] P. Schwaller, T. Laino, T. Gaudin, P. Bolgar, C. A. Hunter, C. Bekas, and A. A. Lee, "Molecular Transformer: A Model for Uncertainty-Calibrated Chemical Reaction Prediction," *ACS Central Science*, vol. 5, no. 9, pp. 1572–1583, 2019.
- [27] P. Schwaller, D. Probst, A. C. Vaucher, V. H. Nair, D. Kreutter, T. Laino, and J. L. Reymond, "Mapping the space of chemical reactions using attention-based neural networks," *Nature Machine Intelligence*, vol. 3, pp. 144–152, 2021. [Online]. Available: <https://doi.org/10.1038/s42256-020-00284-w>
- [28] D. Li, Y. Zhang, H. Peng, L. Chen, C. Brockett, M. T. Sun, and B. Dolan, "Contextualized perturbation for textual adversarial attack," *arXiv*, 2020.
- [29] L. Li, R. Ma, Q. Guo, X. Xue, and X. Qiu, "Bert-attack: Adversarial attack against BERT using BERT," *arXiv*, 2020.
- [30] "RDKit: Open-source cheminformatics," <http://www.rdkit.org>.
- [31] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45. [Online]. Available: <https://www.aclweb.org/anthology/2020.emnlp-demos.6>
- [32] S. Rajbhandari, J. Rasley, O. Ruwase, and Y. He, "Zero: Memory optimizations toward training trillion parameter models," *International Conference for High Performance Computing, Networking, Storage and Analysis, SC*, vol. 2020-Novem, pp. 1–24, 2020.
- [33] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *7th International Conference on Learning Representations, ICLR 2019*, 2019.
- [34] G. R. Bickerton, G. V. Paolini, J. Besnard, S. Muresan, and A. L. Hopkins, "Quantifying the chemical beauty of drugs," *Nature Chemistry*, vol. 4, no. 2, pp. 90–98, 2012.
- [35] P. redErtl and A. Schuffenhauer, "Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions," *Journal of Cheminformatics*, vol. 1, no. 1, pp. 1–11, 2009.
- [36] "gdb9 dataset," <http://deepchem.io.s3-website-us-west-1.amazonaws.com/datasets/gdb9.tar.gz>, accessed: 2021-05-28.
- [37] C. Smillie, M. P. Garcillán-Barcia, M. V. Francia, E. P. C. Rocha, and F. de la Cruz, "Mobility of Plasmids," *Microbiology and Molecular Biology Reviews*, vol. 74, no. 3, pp. 434–452, 2010.
- [38] A. Cully and Y. Demiris, "Quality and Diversity Optimization: A Unifying Modular Framework," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 2, pp. 245–259, 2018.