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 generations 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 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]–[10]. 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], [10], excluding the benefits and diversity produced by larger subsequence rearrangements. In cases with multi-atom (i.e. molecular fragment) representations, custom mutation operators are still required to generate rearrangements [2], [10]–[13]. Although researcher expertise may generate reasonable mutation operators, 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 [14]. The process of tokenization is necessary to construct a vocabulary of fixed size to capture all possible words encountered by the model. One popular method, WordPiece tokenization [15], [16], 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. A token in the vocabulary may represent multiple characters.

After determining the vocabulary, input text sequences in the training data are tokenized (i.e. the vocabulary is used to map subsequences of text to unique token ids). 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

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).

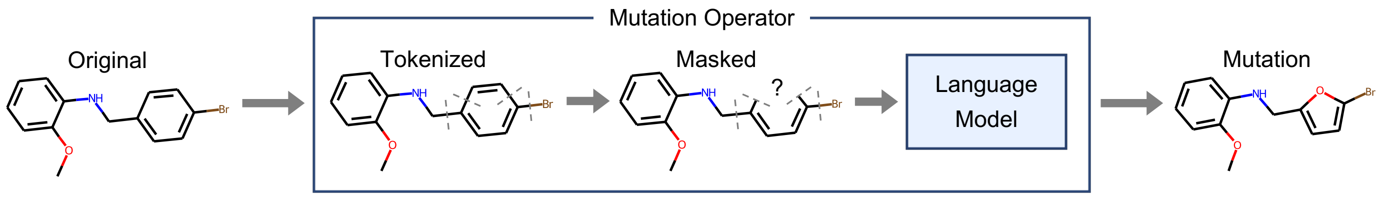

Fig. 1. Strategy to use a masked language model to mutate molecule sequences. First, a given molecule is tokenized (i.e. split into commonly occurring subsequences). Second, certain tokens are masked. Finally, the masked language model ranks possible replacements for the mask based on neighboring tokens,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

original text sequence from the masked sequence [14]. After training on a large unsupervised dataset, the model is then typically trained on smaller fine-tuning tasks for evaluation on specific applications [14], [17], [18]. For practical NLP models, additional tasks beyond mask prediction may be incorporated into the training as well [14], [17], [18].

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 optimization as a use case. A text-based representation known as SMILES, Simplified Molecular Input Line Entry System [19], is used for molecule data; a large compound library [20] is used to assemble 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 an additional test, we utilize iterative training of a mask prediction model without the use of a large compound library. We then apply the best performing models towards a more realistic drug discovery application, optimizing binding affinity for a protein target. 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 metric [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 across a range of molecule generation tasks [9].

In contrast to genetic algorithm studies that utilize single atom changes to generate new molecules [7], [8], [10], multiple investigations have considered the use of multi-atom molecular fragments [2], [11]–[13]. Different methods have been proposed and used to mine common fragments from chemical databases [11]–[13]. For example, Lameijer et al. performed co-occurrence analysis on fragments including rings and linkers to identify unexplored regions of chemical space [12]. For molecule rearrangements, previous work has weighted fragments based on their correlation with a desired property [11]. Within this context, our work to utilize a masked language model to automate mutations shows an alternate strategy for fragment mining (i.e. through tokenization) and rearrangements. The model enables the prediction of fragments based on context to generate new molecules.

Several variations of ML models have been proposed for molecule generation tasks, including generative adversarial networks and recurrent neural networks [21]–[23]. Similar to studies with genetic algorithms, multiple representations of molecules have been used for training data, including graphs and SMILES strings [21], [22]. For models that rely on a string representation, typically a given molecule is tokenized per character (or per atom) [21], [24]–[26]. No investigation has been done into the performance of text based models allowing different subsequence lengths for the vocabulary in molecule generation.

Since the introduction of BERT [14], several investigations have been made in training a transformer model on molecule data [27]–[32]. Similar to text applications, the trained model was applied on fine-tuning tasks concerning chemical property prediction [27]–[30]. Recent work has utilized a trained language model for molecule generation and optimization [29]. Transformer-based models have also been utilized for chemical reaction prediction [31], [32]. However, an investigation into the impact of tokenization on optimization performance has not been conducted. 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 [33], [34]. Although not discussed in relation to genetic algorithms, optimizing text sequences through mask replacement is similar to our proposed strategy.

III. Methods

A. Genetic Algorithm

In this work, we consider the impact of different tokenization schemes and a mask prediction model in automating the mutation of molecule SMILES strings. To focus on the impact of the mutation operator, we utilized a very simple genetic algorithm with a $(2\mu{+}5\mu)$ survivor selection scheme. Random uniform sampling was performed to select μ parents from the population, and only mutation was used to generate new molecules (i.e. no recombination). The mutation operator was applied to each parent to produce 5 offspring. Notice that not all generated offspring are guaranteed to be valid SMILES strings. Also, all offspring were converted to canonical form using rdkit [35] and only unique SMILES were retained.

As shown in Figure 1, the mutation operator actually consists of 3 steps. First, an input molecule is tokenized. Then, randomly selected tokens in the molecule are masked. Here, we utilized a hyperparameter for the maximum number of masks per molecule (N_m). A random integer is generated between 1 and N_m (inclusive) to determine the number of masks for each mutation. Unless otherwise specified, a value of 5 was used for N_m. We considered three different types of mask placement (insertion, replacement, and 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. In the final step, the masks were replaced with tokens from the vocabulary by the language model.

There are two additional hyperparameters for mask replacement by the language model. First, we utilized a batch size of 10 molecules for model predictions. All molecules in each batch had the same randomly sampled mask placement type. Second, the model ranks all tokens in the vocabulary for possible replacements; therefore, we must select the number of top predictions to keep. Here, we used the top five model predictions for mask replacements in all runs. In addition to the language model, we also considered random mask replacement, in which a token is randomly selected from the vocabulary.

With the mutation operator determined, the genetic algorithm simulations were performed for 50 generations. Unless otherwise specified, the population size was set to 1000 (i.e. $\mu = 500$). For fitness F, we used the harmonic mean of the individual metrics under consideration. For example, when optimizing two metrics (x_1 and x_2), we used:

$$F(x_1, x_2) = \frac{2x_1x_2}{x_1 + x_2} \tag{1}$$

By default, we used quantitative estimation of drug-likeness score [36] and normalized synthesizability [22], [37] as the metrics for optimization. Similar to previous work [22], synthesizability was normalized between 0 and 1. Reported values for fitness from the simulations were determined by averaging over a population from a given generation in a genetic algorithm simulation.

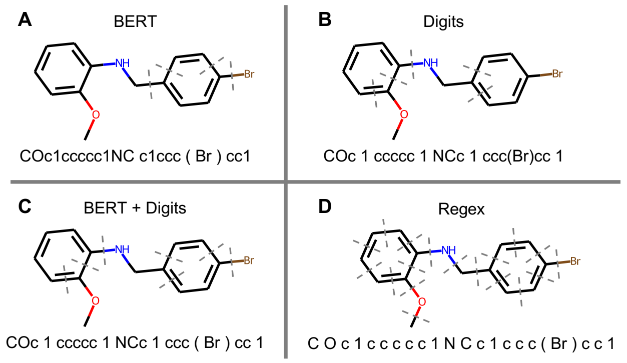
A
BERT
COc1ccccc1NC c1ccc (Br) cc1
B
Digits
COc 1 ccccc 1 NCc 1 ccc(Br)cc 1
C
BERT + Digits
COc 1 ccccc 1 NCc 1 ccc (Br) cc 1
D
Regex
C O c 1 c c c c c 1 N C c 1 c c c (Br) c c 1
NH
O
Br

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-tokenizer based on regex for individual atoms.

Genetic algorithm simulations were performed using a single Nvidia 16 GB V100 GPU. A single simulation using drug-likeness and synthesizability for fitness metrics took approximately 5 minutes to complete. Changing the population size to 10^5 (i.e. $\mu = 5 \cdot 10^4$) increased the time to approximately 240 minutes.

B. Training Data

In order to train the masked language model for mutations, a corresponding training dataset is needed. We utilized two different sources for training data: (1) the Enamine *REAL* database [20] and (2) the output population of iterative rounds of genetic algorithm simulations. For Enamine, 1.2 billion SMILES strings were used for tokenizer training, and a subset of 120 million SMILES were used for model training. For iterative training, the final population of a genetic algorithm simulation with a population size of 10^5 (i.e. $\mu = 5 \cdot 10^4$) was used.

C. Tokenizer and Model Training

The tokenizer and transformer libraries provided by Hugging Face [38] 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. As shown in Figure 2, we performed splitting based off of a previously used regex (Regex) [24], punctuation and digits (BERT + Digits), punctuation only (BERT), and digits only (Digits). We denoted 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 vocabulary size was set to $3 \cdot 10^4$.

All model training was done using the tokenizer and transformer libraries provided by Hugging Face [38] and DeepSpeed [39], with sharded I/O performed using the WebDataset library [40]. By default, the learning rate was set to $5 \cdot 10^{-5}$ and the batch size was set to 128 per GPU using the Adam optimizer [41] to train for 4 epochs. The standard 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 120 million SMILES from the Enamine dataset used the Summit supercomputer at the Oak Ridge Leadership Computing Facility at ORNL. A single training run for a given tokenizer used 50 compute nodes, each with 6 Nvidia 16 GB V100 GPUs, for approximately 1.5 hours. Models trained on 10^5 SMILES from the output of a genetic algorithm simulation took less than half an hour on a single V100 GPU.

IV. Results

A. *Masked Language Model Mutations*

As shown in Figure 3, the different tokenization schemes lead to large differences in the number of novel and accepted molecules produced. Here, the starting population of molecules was taken from the first 1000 molecules of the gdb9 dataset [42]. A molecule is novel if the corresponding canonical SMILES has not been encountered before in a given genetic algorithm simulation. The count of accepted molecules shows the number of novel SMILES that have been incorporated into the population during selection. The patterned bars and the dashed lines show the results for random mask replacement, while the solid bars and solid lines show the results for a masked language model trained on the Enamine dataset.

For most of the tokenizers, the mask prediction model resulted in a boost for novel molecules produced. Interestingly, however, random replacement for BERT+Digits produced more molecules than the trained model. A comparison of accepted molecules, as well as fitness values, showed that the mask prediction model outperformed random replacement for all tokenizers considered. The best tokenizer in terms of fitness was BERT, followed closely by Digits, suggesting that for optimizing synthesizability and drug-likeness, a vocabulary with larger subsequences was beneficial. For fitness results with additional values for the maximum number of masks per molecule (N_m), see Table AI.

The Regex tokenization scheme with a mask prediction model produced the highest number of novel and accepted molecules. However, additional molecules did not translate to an increase in optimization performance. One of the challenges for the Regex and BERT+Digits schemes can be seen in the plot of atoms per molecule in the population. With individual tokens representing larger chemical sequences, the BERT and Digits tokenizers were able to generate larger molecules within the first few generations. Notice that, although increases in molecule length were correlated with increases in fitness, the typical sizes of molecules in the population plateaued after a few generations, suggesting that further increases in number of atoms are not beneficial to higher fitness scores.

As another metric for comparison between the tokenization schemes, we considered the similarity of molecules in the final population from a given genetic algorithm simulation. To calculate similarity between two molecules, we generated Morgan fingerprints [43] and used Tanimoto similarity in rdkit [35]. To get a similarity score for the population, we averaged the scores from all unique pairs. As shown in Table I,

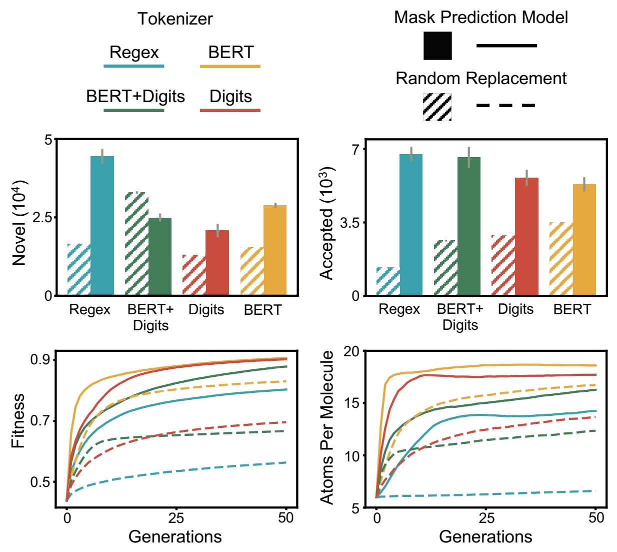

Fig. 3. Metrics for genetic algorithm simulations using both random replacement and the mask prediction model to generate mutations with different tokenizers. We track the number of novel molecules produced over a 50 generation simulation along with the number of molecules accepted into the population during survivor selection. Also, the mean fitness and atoms per molecule are shown. All results show the average of 5 runs. The error bars for Novel and Accepted molecules show one standard deviation.

TABLE I
Similarity scores for the final population with different mutation operators. The similarity score shown is the average of five runs, with one standard deviation reported as the uncertainty.

Random Replacement		Mask Prediction Model	
Tokenizer	Similarity	Tokenizer	Similarity
Regex	0.0682 ± 0.0012	Regex	0.1487 ± 0.0167
BERT+Digits	0.1272 ± 0.0114	BERT+Digits	0.1619 ± 0.0092
Digits	0.1312 ± 0.0014	Digits	0.1635 ± 0.0117
BERT	0.1292 ± 0.0006	BERT	0.1527 ± 0.0135

random replacement generated lower similarity scores than the mask prediction models. Furthermore, the Regex tokenizer generated the lowest scores for both random replacement and mask prediction model. This suggests that the use of vocabularies with multi-atom sequences may result in a decrease in population diversity, however, the decrease may also partially be caused by the increase in population fitness.

In addition to mask prediction with a single tokenization scheme, we also tested optimization based off of two different schemes. For combinations, we used two different mutation operators. For the μ sampled parents, half were sent to each respective mutation operator with the associated tokenizer and model. As shown in Table II, 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 different tokenization schemes (e.g. a BERT tokenizer and a Digits tokenizer) in two mutation operators is

TABLE II
FINAL FITNESS SCORES FOR DIFFERENT TOKENIZATION SCHEMES WITH A MASK PREDICTION MODEL. SCORES ARE THE AVERAGE OF FIVE RUNS, WITH ONE STANDARD DEVIATION REPORTED AS THE UNCERTAINTY.

Tokenizer 1	Tokenizer 2	Fitness
BERT	Digits	0.9482 ± 0.0023
BERT	BERT+Digits	0.9148 ± 0.0041
BERT+Digits	Digits	0.9107 ± 0.0103
BERT	Regex	0.9100 ± 0.0065
BERT	BERT	0.9063 ± 0.0072
Digits	Digits	0.9017 ± 0.0140
Regex	Digits	0.8975 ± 0.0007
BERT+Digits	BERT+Digits	0.8786 ± 0.0198
Regex	BERT+Digits	0.8679 ± 0.0197
Regex	Regex	0.8031 ± 0.0183

different than a single mutation operator that combines two schemes (e.g. BERT+Digits). Using two different tokenization schemes results in two distinct models that have different molecule representations. 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 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.

B. 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 [24] along with random mask replacement. Rounds of mutations are then sufficient to generate new candidates (i.e. molecules). We began with a single sample in the population, which is a single carbon atom. We then performed a genetic algorithm simulation to generate a population of molecules; the final population was used to train a tokenizer and associated mask prediction model. To prevent overfitting with the dataset produced by the genetic algorithm, we added weight decay ($3 \cdot 10^{-7}$) to training. Iterations of this strategy (i.e. genetic algorithm simulation to generate molecules, train a tokenizer and mask prediction model on the final population, repeat) can be performed for optimization. The use of iterative training allows the tokenizer and mask prediction model to capture changes 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 iterative training, we compared two different approaches for the mutation operator. As a control, we used a single mutation operator with the Regex tokenizer and random mutations.

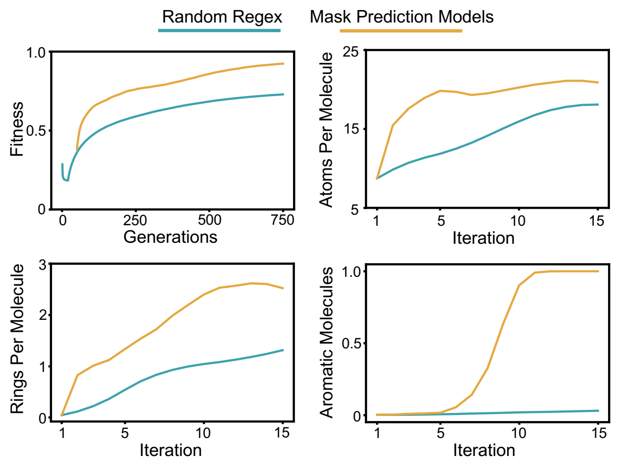

Fig. 4. Tracking the fitness, number of atoms per molecule, number of rings per molecule, and fraction of aromatic molecules in the population for each iteration. Notice that a single iteration corresponds to a single genetic algorithm simulation of 50 generations and associated tokenizer and mask prediction model training. 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).

For comparison, we used a combination of three different mutation operators: a Regex tokenizer with random mutations, a BERT tokenizer with a mask prediction model, and a Digits tokenizer with a mask prediction model. A population size of 10^5 (i.e. $\mu = 5 \cdot 10^4$) was used for the genetic algorithm simulations. For the three tokenizer combination, the following fractions of the μ sampled parents were sent to each respective tokenizer: $\mu/2$, $\mu/4$, $\mu/4$.

As shown in Figure 4, the addition of trained tokenizers and mask prediction models substantially improved 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 4, both strategies were 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. For the fitness results of iterative training with additional hyperparameter values for learning rate and number of epochs, see Table AII.

C. Drug Discovery Applications

Although our results have shown that a masked language model can improve the optimization performance of a genetic algorithm, we have utilized metrics (i.e. synthesizability and drug-likeness) for fitness that are not specific to any given drug-related application. Therefore, to provide an example of optimization towards a more realistic fitness metric for drug discovery, we incorporated the normalized binding affinity predictions of a recently developed ML model [44] in addition

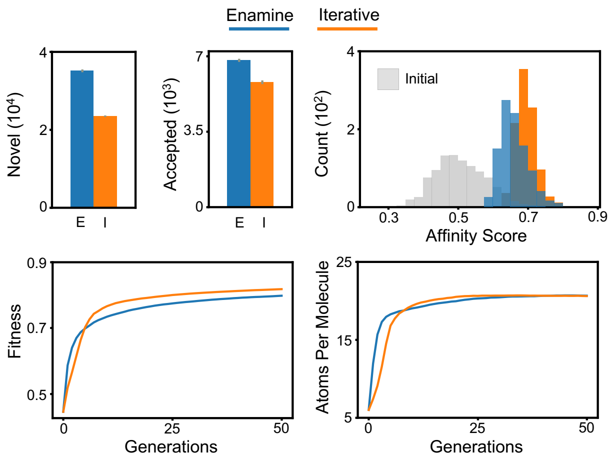

Fig. 5. Metrics for the genetic algorithm simulations with affinity score included in fitness. We track the number of novel molecules produced over a 50 generation simulation along with the number of molecules accepted into the population during survivor selection. The histogram shows the distribution of affinity scores in the initial population and the final population for mask prediction models using two different training datasets from the genetic algorithm simulations with the highest fitness values. Also, the population fitness and atoms per molecule are shown.

to drug-likeness and synthesizability. The ML model was previously used to search for optimized inhibitors of a protein target, Mpro from SARS-CoV-2 [44].

For the genetic algorithm simulations, we used the previous best performing combination of mutation operators, i.e. two mask prediction models with a BERT and Digits tokenizer respectively (see Table II). The starting population was the same as used in Section IV-A. In addition to the models trained on the Enamine dataset, we also used the final models from iterative training. As shown in Figure 5, the models trained on Enamine produced more novel and accepted molecules in comparison to the final iterative models. However, the iterative models produced a population with higher affinity scores and higher fitness values on average. The size of molecules produced by both types of models show similar results. The increased fitness performance for the iterative models is especially interesting, as the training dataset is multiple orders of magnitude smaller (i.e. 10^5 compared to $1.2 \cdot 10^8$). This suggests that the benefits of mask prediction models for mutations are not limited to cases with large existing databases for training data. Furthermore, although the development and analysis of new metrics for drug discovery is beyond the scope of this work, our results suggest that mask prediction models are suitable for optimizing metrics provided by both heuristic scoring functions and ML models.

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 [45]. Mutation coupled with selection enables (in part) microbial adaptation and survival in harsh environments (e.g. antibiotic resistance) [45]. 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. For example training only on the Enamine dataset may limit the diversity of chemical space explored by the mask prediction model. 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], [46] can be utilized for future investigations.

In this work, we have utilized a very simple genetic algorithm, with a $(2\mu + 5\mu)$ survivor selection scheme and only mutation for new molecule generation. 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 diversity maintenance [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 [14], 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, requires the definition of a suitable mutation operator. To move beyond the limitations of random point-wise mutations, 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.

Appendix
Hyperparameters

In addition to the default parameters mentioned in Methods (Section III), we conducted genetic algorithm simulations varying the maximum number of masks per molecule (N_m). Table AI shows the final fitness results for different values of N_m. Although varying the number of masks alters the fitness values, it does not change the relative order for the different tokenization schemes. Notice that the results for $N_m = 5$ correspond to the values shown in Table II.

For iterative training, we varied the learning rate and number of epochs. For the smaller learning rate ($3 \cdot 10^{-5}$), we also

TABLE AI
FINAL FITNESS SCORES FOR DIFFERENT TOKENIZATION SCHEMES WITH A MASK PREDICTION MODEL. THE MAXIMUM MASKS PER MOLECULE (N_m) IS VARIED. SCORES ARE THE AVERAGE OF FIVE RUNS, WITH ONE STANDARD DEVIATION REPORTED AS THE UNCERTAINTY.

Tokenizer	N_m	Fitness
Regex	1	0.8122 ± 0.0016
Regex	3	0.8106 ± 0.0064
Regex	5	0.8031 ± 0.0183
BERT+Digits	1	0.8329 ± 0.0025
BERT+Digits	3	0.8426 ± 0.0043
BERT+Digits	5	0.8786 ± 0.0198
Digits	1	0.8945 ± 0.0008
Digits	3	0.8962 ± 0.0009
Digits	5	0.9017 ± 0.0140
BERT	1	0.8991 ± 0.0048
BERT	3	0.9041 ± 0.0021
BERT	5	0.9063 ± 0.0072

used a linear warmup of 500 steps. As shown in Table AII, all iterative training runs resulted in a large improvement over the random Regex baseline, which had a final population average fitness of 0.7295. For the values explored, our results suggest that increasing the learning rate and number of epochs has a positive impact on optimization performance, however, we leave a more systematic study of iterative training hyperparameters and variations amongst repeated simulations to future investigations.

TABLE AII
FINAL FITNESS SCORES FOR DIFFERENT HYPERPARAMETERS USED IN ITERATIVE TRAINING. VALUES SHOW THE FINAL POPULATION AVERAGE FOR A SINGLE RUN.

Learning Rate	Epochs	Fitness
$3 \cdot 10^{-5}$	4	0.8367
$3 \cdot 10^{-5}$	10	0.9190
$5 \cdot 10^{-5}$	4	0.9245
$5 \cdot 10^{-5}$	10	0.9260

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] E. W. Lameijer, J. N. Kok, T. Bäck, and A. P. Ijzerman, "The molecule evoluator. An interactive evolutionary algorithm for the design of drug-like molecules," *Journal of Chemical Information and Modeling*, vol. 46, no. 2, pp. 545–552, 2006.

[11] C. A. Nicolaou, J. Apostolakis, and C. S. Pattichis, "De novo drug design using multiobjective evolutionary graphs," *Journal of Chemical Information and Modeling*, vol. 49, no. 2, pp. 295–307, 2009.

[12] E. W. Lameijer, J. N. Kok, T. Back, and A. P. Ijzerman, "Mining a chemical database for fragment co-occurrence: Discovery of "chemical clichés"," *Journal of Chemical Information and Modeling*, vol. 46, no. 2, pp. 553–562, 2006.

[13] G. Schneider, M. L. Lee, M. Stahl, and P. Schneider, "De novo design of molecular architectures by evolutionary assembly of drug-derived building blocks," *Journal of Computer-Aided Molecular Design*, vol. 14, no. 5, pp. 487–494, 2000.

[14] 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. Mlm, pp. 4171–4186, 2019.

[15] 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.

[16] 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

[17] 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

[18] 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.

[19] D. Weininger, “SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules,” *J. Chem. Inf. Comput. Sci.*, vol. 28, pp. 31–36, 1998.

[20] “Enamine *REAL* Database,” https://enamine.net/compound-collections/real-compounds/real-database, accessed: 2020-04-01 through https://virtual-flow.org/.

[21] 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.

[22] 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.

[23] 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

[24] 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.

[25] 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

[26] 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.

[27] 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.

[28] 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

[29] 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.

[30] S. Honda, S. Shi, and H. R. Ueda, “Smiles transformer: Pre-trained molecular fingerprint for low data drug discovery,” *arXiv*, 2019.

[31] 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.

[32] 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

[33] D. Li, Y. Zhang, H. Peng, L. Chen, C. Brockett, M. T. Sun, and B. Dolan, “Contextualized perturbation for textual adversarial attack,” *arXiv*, 2020.

[34] L. Li, R. Ma, Q. Guo, X. Xue, and X. Qiu, “Bert-attack: Adversarial attack against BERT using BERT,” *arXiv*, 2020.

[35] “RDKit: Open-source cheminformatics,” http://www.rdkit.org.

[36] 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.

[37] P. Ertl 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.

[38] 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

[39] 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.

[40] A. Aizman, G. Maltby, and T. Breuel, “High performance I/O for large scale deep learning,” in *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, 2019, pp. 5965–5967.

[41] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2017.

[42] “gdb9 dataset,” http://deepchem.io.s3-website-us-west-1.amazonaws.com/datasets/gdb9.tar.gz, accessed: 2021-05-28.

[43] D. Rogers and M. Hahn, “Extended-connectivity fingerprints,” *Journal of Chemical Information and Modeling*, vol. 50, no. 5, pp. 742–754, 2010.

[44] A. E. Blanchard, J. Gounley, D. Bhowmik, M. Chandra Shekar, I. Lyngaas, S. Gao, J. Yin, A. Tsaris, F. Wang, and J. Glaser, “Language Models for the Prediction of SARS-CoV-2 Inhibitors.” [Online]. Available: https://www.biorxiv.org/content/10.1101/2021.12.10.471928v1

[45] 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.

[46] 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.