RESEARCH

Adaptive Language Model Training for Molecular Design

Andrew E Blanchard[1†], Debsindhu Bhowmik[1*], John Gounley[1], Jens Glaser[2], Belinda S Akpa[3,4] and Stephan Irle[1]

[*]Correspondence: bhowmikd@ornl.gov
[1]Computational Sciences and Engineering Division, Oak Ridge National Laboratory, Oak Ridge, TN, 37831, USA
Full list of author information is available at the end of the article

Abstract

The vast size of chemical space necessitates computational approaches to automate and accelerate the design of molecular sequences to guide experimental efforts for drug discovery. Genetic algorithms provide a useful framework to incrementally generate molecules by applying mutations to known chemical structures. Recently, masked language models have been applied to automate the mutation process by leveraging large compound libraries to learn commonly occurring chemical sequences (i.e., using tokenization) and predict rearrangements (i.e., using mask prediction). Here, we consider how language models can be adapted to improve molecule generation for different optimization tasks. We use two different generation strategies for comparison, fixed and adaptive. The fixed strategy uses a pre-trained model to generate mutations; the adaptive strategy trains the language model on each new generation of molecules selected for target properties during optimization. Our results show that the adaptive strategy allows the language model to more closely fit the distribution of molecules in the population. Therefore, for enhanced fitness optimization, we suggest the use of the fixed strategy during an initial phase followed by the use of the adaptive strategy. We demonstrate the impact of adaptive training by searching for molecules that optimize both heuristic metrics, drug-likeness and synthesizability, as well as predicted protein binding affinity from a surrogate model. Our results show that the adaptive strategy provides a significant improvement in fitness optimization compared to the fixed pre-trained model, empowering the application of language models to molecular design tasks.

Keywords: Masked Language Model; Drug Discovery; Genetic Algorithm

Introduction

The goal of rational drug design is to identify molecules with specified properties associated with therapeutic value. Emerging infectious diseases (e.g. SARS-CoV-2 and the associated pandemic) highlight the need for rational design to accelerate the discovery of drugs in response to novel protein targets [1, 2]. Computer aided drug discovery (CADD) provides a set of tools to shorten the time and cost of searching chemical space for new applications [2–7]. In addition to the development of biophysical models and simulations traditionally associated with CADD [5–7], much recent work has focused on using methods from machine learning (ML) and artificial intelligence (AI) for molecular design [4, 5, 7–9].

The use of ML models in drug design has been enabled by the availability of large compound libraries [10] and experimental datasets [11, 12] along with computational libraries for cheminformatics [13]. Within a design application, models generally serve one of two possibly overlapping roles, molecule generation and molecule scoring. Generative models, such as variational autoencoders [8, 14] and generative adversarial networks [15, 16], are capable of sampling new molecules from chemical space based off a training set. Scoring models, on the other hand, take a molecule as input and generate a prediction for a given property (e.g. protein binding affinity). Through iterations of generation and scoring, searches over chemical space can be performed to optimize a given property. The iterative process for optimization is commonly referred to as a genetic algorithm [17].

Genetic algorithms provide a useful strategy for the design of molecular sequences for drug discovery applications. To use a genetic algorithm, a representation for a chemical sequence must be chosen along with a mutation operator to generate new sequences. The mutation operator is then used to explore chemical space and selection is performed according to a pre-defined fitness objective. Previous studies have used genetic algorithms successfully for a range of drug discovery applications [18–22]. Furthermore, benchmark studies have shown that genetic algorithms can achieve state-of-the-art results for molecule generation, comparing favorably to recent machine learning techniques [19, 21].

Despite the success of genetic algorithms, the need to define an appropriate representation for a chemical sequence and a mutation operator poses a challenge. Previous studies have often utilized a simple representation by enumerating individual atoms and bonds within a molecule [18, 19, 22]. For mutation, hand-crafted rules, such as add an atom, delete an atom, or create a ring, have been proposed and used for large scale exploration of chemical space [18]. Additional studies have used data mining techniques to discover commonly occurring multi-atom fragments and used custom mutation operators to rearrange the specified fragments [20, 22–25]. However, specifying fixed rules for rearrangements limits the ability to adapt the optimization procedure to a given task. Ideally, the mutation operator can be automatically inferred from the data, reducing the need for intuition and generalizing the genetic algorithm approach to new molecular design tasks.

Inspired by advances in natural language processing (NLP) [26], recent studies have shown how to automate both the choice of representation for chemical structure and the mutation operator [2, 27]. Starting with a text-based representation for molecules, Simplified Molecular Input Line Entry System (SMILES) [28], the process of tokenization is used to determine commonly occurring subsequences [29, 30]. The subsequences are stored as a vocabulary and are used to map a given molecule sequence into a list of token IDs. Each token ID may correspond to multiple characters (i.e., atoms and bonds) in a given molecule. Once a tokenization scheme is defined, the molecule data can be used to train a masked language model. In the training for such a model, tokens are randomly masked and the loss is determined by how well the model reproduces the original sequence when predicting the masked tokens [26].

Without the need for labels, unsupervised training of masked language models can be performed on large compound libraries (e.g. Enamine *REAL* database) [10]. For a given mask, a trained model will rank possible ways to complete the molecular sequence based on the vocabulary. Therefore, sampling from the top mask predictions provides an automated mutation operator for a genetic algorithm [27]. Therefore, in contrast to manually defining rules for mutations, masked language models provide an automated solution for discovering both useful molecular representations (i.e., through tokenization) and mutations (i.e., through mask prediction) as shown in Figure 1.

Although the use of a fixed pre-trained masked language model provides a useful improvement over manually defined rules, the challenge to adapt molecule generation for different optimization tasks remains. For example, the dataset used for

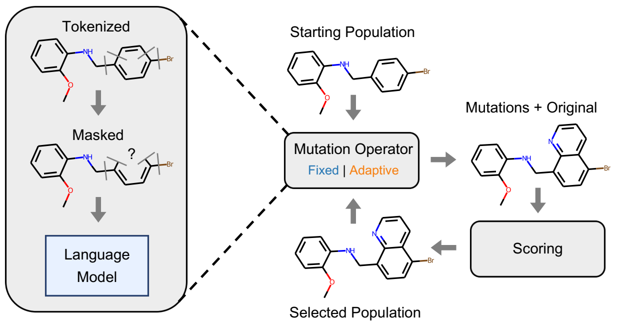

Figure 1 Strategy for molecule optimization using a language model. An initial population of molecules is used as input. The language model then generates mutations using predictions for randomly placed masks. Molecules are ranked according to a specified score and top performers are selected for another round of mutations. Two approaches for the language model are investigated, fixed and adaptive. For the fixed approach, the language model is pre-trained on a large molecule dataset and it does not change during the optimization process. For the adaptive approach, the language model is trained on the selected population, which itself changes during the optimization process.

model pre-training may have certain biases that limit structural rearrangements useful for a new task. In order to overcome this difficulty, we here propose a novel way to use language models within genetic algorithm optimization. Specifically, we continue to train the masked language model on populations selected for a specified fitness objective. By continued training on the selected population, we hypothesized that the language model would adapt to new regions of chemical space useful for optimization.

In order to test our hypothesis, we implemented two approaches for comparison - fixed and adaptive. In the fixed approach, a pre-trained language model was used to generate new molecules. In the adaptive approach, the pre-trained language model is used as a starting point and further trained using mask prediction on a specified population. Continued training is performed after each iteration of the genetic algorithm to produce a new population of molecules. Our results show that the adaptive approach produces data that more closely mimics the genetic algorithm population. For optimization, the adaptive approach leads to increases in fitness for tasks using both heuristic metrics and a ML surrogate model. Therefore, by introducing the adaptive approach for automating mutations we broaden the capabilities of genetic algorithm optimization for molecular design.

Methods

Genetic Algorithm

In this work, we focused on the molecule generation capabilities of a masked language model for fitness optimization. The source code for this work can be found at https://code.ornl.gov/candle/mlmol in the adaptive-lm directory. As described in previous work [27], a masked language model can be used as an automated mutation operator within a genetic algorithm. Figure 1 shows the major components for optimization. An initial population of molecules, in the form of SMILES strings are used as input to the masked language model. Portions of a given SMILES string are then randomly masked and the language model is used to predict mutations to the original molecule. The generated molecules are then scored and selection is performed based on the specified fitness to generate an optimized population. The process of mutation and selection can be repeated for a specified number of iterations.

For the language model acting as the mutation operator, we considered two different training strategies, fixed and adaptive. In both cases, we started by pre-training a masked language model on a dataset with billions of molecules (for further details on the dataset, see Methods Section - Molecule Data). For the fixed strategy, weights of the pre-trained model were frozen, and the model was used only for inference (i.e., mask prediction) as part of the genetic algorithm. For the adaptive strategy, however, model training based on mask prediction was performed for one epoch during each generation, with the current population of molecules used as the training data. The language model, therefore, adapted to the patterns found in the current population of the genetic algorithm before generating mutations.

To distinguish between the optimization performance of the fixed and adaptive strategies, we utilized a relatively simple genetic algorithm with a $(\mu + 5\mu)$ survivor selection scheme. Random uniform sampling with replacement was used to select μ parents from the population, and only mutation was used to generate new molecules, similar to our previous work [27]. A population size (μ) of 10^5 was used for all reported genetic algorithm simulations. Mutations were generated by taking the top 5 predictions from the masked language model for a given set of masks. Validity and uniqueness of the generated molecules were determined using rdkit [13] to convert SMILES strings into canonical form. Only unique molecules were retained in the population. All reported results, except for example histograms, show the mean over six repeated runs, with the standard deviation used to calculate error bars. Example histograms show the distribution of metric values for a single run.

For mask generation, we considered the following different values for the mutation rate (i.e., probability that a given token will be masked): $[0.15, 0.30, 0.45, 0.60, 0.75]$. In addition, three different types of mutation (replacement, insertion, and deletion) were used. For each type, the number of mutations was determined using the binomial distribution for the appropriate number of tokens and mutation rate. A minimum number of 1 mask per molecule was enforced. The locations for each mutation within the molecule string were then randomly sampled. For replacement, the sampled token locations were replaced with a mask. For insertion, one sampled location was used to insert a mask before the given token. Similarly, for deletion, one sampled location was used to delete the token following the mask. The remaining sampled locations for both insertion and deletion were used for replacement.

Fitness in the genetic algorithm simulations was determined using the harmonic mean of multiple molecular metrics. For example, for two metrics (x_1 and x_2), we used a fitness F given by:

$$F(x_1, x_2) = \frac{2x_1x_1}{x_1 + x_2} \tag{1}$$

By default, we used quantitative estimations of drug-likeness and normalized synthesizability, similar to several previous studies on molecular optimization [15, 16, 31, 32]. To apply the genetic algorithm strategies on a more realistic drug discovery scenario, we also utilized a recently released model for protein binding affinity prediction to generate a molecular metric [33]. Specifically, we used a predicted affinity score for the main protease of SARS-CoV-2. The resulting fitness was, therefore, the harmonic mean of drug-likeness, synthesizability, and the predicted affinity score.

Molecule Data

Similar to previous work [2], we generated a molecule dataset starting from the Enamine *REAL* database [10]. Using a data augmentation strategy with a previously trained language model, we increased the number of molecules to approximately $3.6 \cdot 10^{10}$. In preparation for model training, the dataset was partitioned into $7.2 \cdot 10^4$

files, each with $5{\cdot}10^5$ molecules, stored using the WebDataset [34] library for sharded data loading during model training.

In addition to the constructed molecule dataset, we used two additional datasets as the starting population for genetic algorithm simulations. First, we used a subset of 10^5 molecules from QM9 [35, 36], referred to in the text and figures as GDB9. Second, we selected the top 10^5 in terms of drug-likeness and synthesizability from a hold-out set of the training data, referred to in the text and figures as Top. These two datasets were used to show the difference in performance for the fixed and adaptive strategies when starting from a relatively low and high initial fitness respectively.

Language Model Training

Language model pre-training consists of two different stages, tokenization and mask prediction. During tokenization, a vocabulary is generated for the model based on commonly occurring subsequences within the SMILES string for molecules. Here, we split the SMILES string during pre-processing based on punctuation, which is the default splitting used for the BERT WordPiece tokenizer in the Hugging Face transformers library [37]. The vocabulary for the WordPiece tokenizer was then generated using the full 36 billion molecule dataset, with the vocabulary size set to 32,768.

For mask prediction, we used PyTorch and Hugging Face transformers along with DeepSpeed for distributed training [38]. As described in [2], we used data parallelism with DeepSpeed's fused LAMB optimizer to train at scale on a dataset of 3 billion molecules (i.e., the first 6000 partitions of the full molecule dataset). Pre-training was performed on the Summit supercomputer using 1000 nodes (6 Nvidia 16 GB V100 GPUs per node), with each partition of the dataset assigned to a single GPU. We used a batch size of 80 molecules with 3 gradient accumulation steps per GPU, leading to a global batch size of 1.44 million. Pre-training was done for 7 epochs, taking approximately 2.5 hours, and model validation was done using mask prediction on a hold-out set of molecules. The best validation accuracy occurred for the final epoch, and the resulting model weights were frozen for language model mutations in the fixed strategy. The model weights were used as the initial conditions for continued training in the adaptive strategy.

Surrogate Model for Binding Affinity

In addition to the heuristic metrics for drug molecules, synthesizability and drug-likeness, we also used an ML model to predict protein binding affinity for a given target, in this case the main protease of SARS-CoV-2. As described in previous work [2], the binding affinity model was generated by fine-tuning language models for both molecule and protein sequences. The output of the model is the predicted negative log (base 10) of the binding affinity. To convert to an affinity score for fitness, we divided the prediction by 10 and clipped the resulting values between 0 and 1. Although the validation and discussion of this model are beyond the scope of the current work, we chose it as an example to illustrate that our proposed optimization strategies can be applied to find high-scoring candidates for both heuristic and ML surrogate scoring models.

Results

Fixed and Adaptive Strategies for Molecule Generation

Before analyzing the impact of continued language model training on molecule optimization, we considered a simpler task: generating mutations for a fixed set of initial molecules. We implemented this task by using the genetic algorithm without selection (i.e. the parent population remains unchanged). During each generation, mutations are generated and the resulting unique molecules are saved for further analysis. For the fixed strategy, mutations are generated from the fixed pre-trained model, while for the adaptive strategy, the language model is trained for 1 epoch on the initial data in each generation before producing mutations.

As shown in Figure 2, we used two different initial datasets, GDB9 [36] and Top (see Methods Section - Molecule Data). The mutation rate determines the fraction of tokens that are randomly masked during the generation of new molecules. Each genetic algorithm simulation was run for 5 generations. For each run, the mean drug-likeness and synthesizability scores were calculated for all unique molecules produced in each generation outside of the original data. The histograms show an example of the distributions for novel molecules with a metric value greater than zero produced from the final generation of a single run with a mutation rate of 0.3.

Due to the continued training of the language model, the mutations generated by the adaptive strategy are much closer, in terms of synthesizability and drug-likeness,

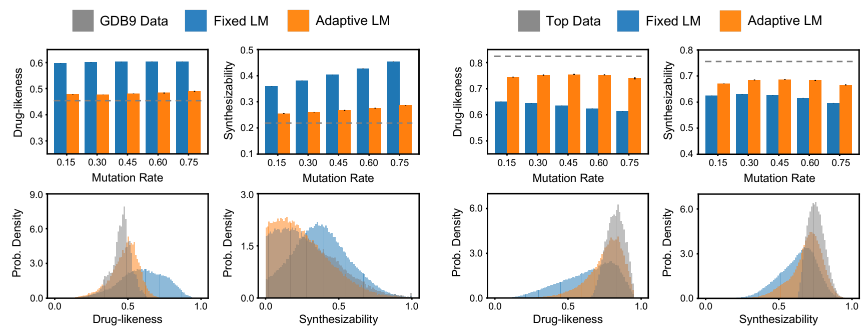

Figure 2 Distributions of molecules produced by a fixed and adaptive approach. Two datasets (GDB9 and a custom dataset with the top scoring molecules for drug-likeness and synthesizability) are used as training data. The fixed approach (blue) generates a broad distribution of molecule scores, while the adaptive approach (orange) more closely mimics the training dataset. Notice that for initial training data with low scores (i.e., GDB9), the adaptive approach produces lower scores on average than the fixed approach, while the situation is reversed for initial training data with high scores (i.e., Top).

to the initial population of molecules. This leads to a decrease in typical values for the GDB9 dataset. However, for the Top molecules, the adaptive strategy produces higher scores. This result can be intuitively explained, as the fixed model is biased by the data used in pre-training (i.e., the pre-trained model will tend to produce mutations that were prevalent in its training dataset). Continued training allows the model to adapt to the new data, either GDB9 or Top.

Fixed and Adaptive Strategies for Molecule Optimization

For molecular optimization, the ability to adapt to a given initial dataset may or may not be beneficial. In the case of initial data with relatively low scores, we expect the adaptive strategy to slow down optimization, as the generated molecules with have scores similar to the poor initial data. To test this hypothesis, we applied a genetic algorithm to optimize molecules for drug-likeness and synthesizability starting from the GDB9 dataset. As shown in Figure 3, the adaptive strategy indeed results in decreased fitness relative to the fixed strategy. The fitness plot shown over five generations was generated with a mutation rate of 0.3. Furthermore, the adaptive strategy produced less valid molecules and less accepted molecules (i.e., molecules accepted into the population during selection) for all mutation rates.

The same genetic algorithm applied to the Top dataset produces the opposite results in terms of fitness. Here, the adaptive strategy outperforms the fixed strategy

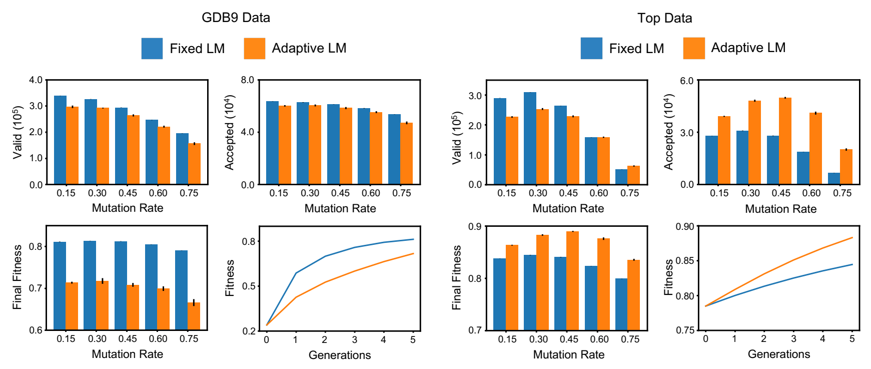

Figure 3 Optimization of molecules for drug-likeness and synthesizability produced by a fixed and adaptive approach. Two datasets (GDB9 and a custom dataset with the top scoring molecules for drug-likeness and synthesizability) are used as initial data. The fixed approach (blue) results in a faster increase in fitness, along with greater valid and accepted molecules for the GDB9 dataset. For the top dataset, however, the adaptive approach leads to a faster increase in fitness along with greater accepted molecules.

for all mutation rates considered. Interestingly, although the adaptive strategy produces fewer valid molecules for most mutation rates (similar to the GDB9 dataset), it produces more accepted molecules in all cases. The decrease in valid molecules can be understood as adaptive training leading to possible issues with over-fitting the current dataset, rather than learning from the large compound library used for pre-training. However, the increase in accepted molecules suggests that molecular rearrangements learned from a high scoring dataset can improve fitness optimization despite the decrease in valid molecules. For the following analysis, we fixed the mutation rate to 0.3 and focused on ways to use the fixed and adaptive strategies together for molecular design.

Combining Fixed and Adaptive Strategies

The trade-off in performance for the fixed and adaptive strategies, depending on the distribution of values in the initial dataset, suggests that mixing fixed and adaptive strategies may be useful for molecular optimization. For a new optimization task, a previously optimized dataset will likely not exist to serve as an initial population. In many cases, generating a reasonably optimized dataset may be the entire goal of applying the optimization procedure. Therefore, we assume that the case with poorly optimized initial data, similar to GDB9, is more representative of a typical molecular design problem. In this case, our results have shown that the fixed strategy outperforms the adaptive strategy for optimization. However, as the fitness of the population increases, we expect that the adaptive strategy may provide a better alternative to optimize fitness.

To test this hypothesis, we implemented various schedules for combining the fixed and adaptive strategies. As show in Figure 4, the fixed strategy was used initially and then replaced by the adaptive strategy after a specified number of generations. As expected, the optimal strategy involves a combination of the two strategies, with five generations of fixed followed by 20 generations of adaptive. Interestingly, although the purely adaptive strategy (orange) increases much more slowly than the purely fixed strategy (blue), adaptive overtakes fixed in terms of fitness after approximately 15 generations. This suggests that the difficulties associated with fitting more closely to a poor initial dataset can be overcome with the ability to adapt to the population as fitness increases.

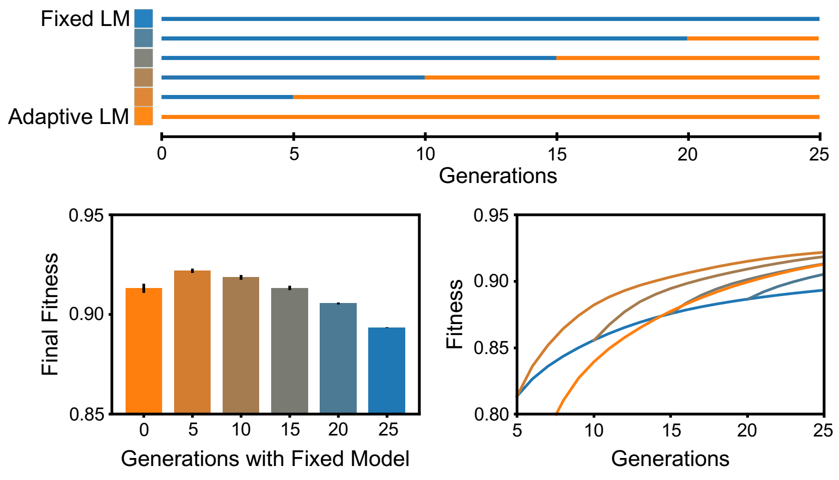

Figure 4 Combining fixed and adaptive approaches during optimization. The fixed approach is used during optimization for 25 epochs. For comparison, the adaptive approach is used starting from the output population of the fixed approach at different generations. The highest fitness is achieved in the case where the adaptive approach is used after 5 epochs of the fixed approach. Notice that the adaptive approach starting from the same initial data as the fixed approach achieves a higher fitness after approximately 15 epochs.

Molecular Optimization Using a Surrogate Model

All of the results we have shown so far have used heuristic functions to score molecules (i.e., synthesizability and drug-likeness scores). However, molecular optimization applications may involve additional ML-based surrogate models for scoring. For example, a ML model may be trained on a limited experimental dataset in order to search for molecules with related properties. Here, we use a previously trained surrogate model, which is available for download [33], developed to predict binding affinity for a given protein and molecule. We fix the protein sequence to the main protease of SARS-CoV-2, as described previously [2], and generate a normalized affinity score to use in fitness calculations. Due to the added computational cost of evaluating the surrogate model, genetic algorithm simulations were run in intervals of five generations. Upon restarting, the model weights were initialized to the fixed pre-trained model.

Building off the results for optimization with heuristic metrics, we compare two optimization schedules. We first apply the fixed strategy for five generations. This is followed by the adaptive strategy for 20 generations, with the continued fixed strategy for comparison. As shown in Figure 5, the adaptive strategy results in a substantial increase in fitness over the fixed strategy for optimization with the surrogate model. By comparing the histograms for synthesizability, drug-likeness, and affinity score, we determined that the increase in fitness values was primarily the result of increases to the affinity score, suggesting that the adaptive strategy is particularly useful for optimizing the ML scoring model. We also show examples of molecules with different values for the three metrics used during fitness optimization. Beyond generating molecules with high values for all three metrics, the examples show how changes in the chemical structure for a family of molecules result in trade-offs amongst synthesizability, drug-likeness, and affinity score.

Discussion

Sequence-Only Models for Drug Design

The models presented in this work for both molecule generation and scoring rely only on the molecular sequence (i.e., the SMILES is the only model input). A sequence-only approach is in contrast to ML models that utilized many local and global features (e.g. molecular fingerprints) [3]. Simulation and modeling approaches outside of ML, such as molecular dynamics and docking, use the full three-dimensional structures of both the protein and molecule to predict binding affinity [5]. The primary strength and weakness, therefore, of sequence-only models is the simplicity of the model input. By using SMILES, the model has no direct information concern-

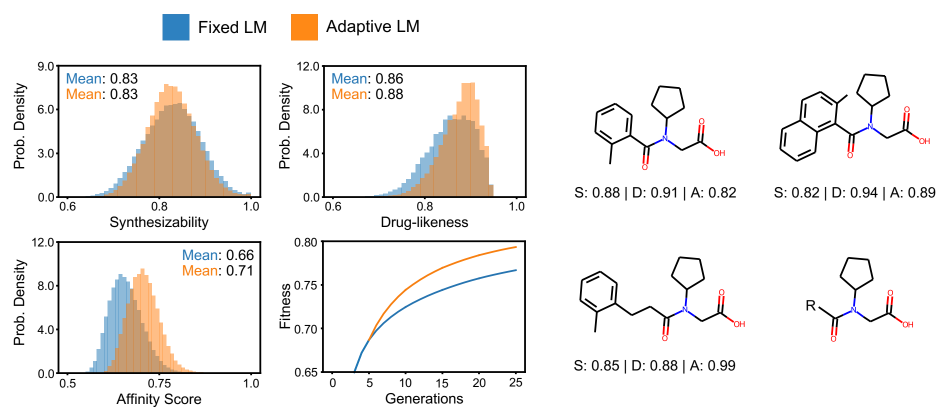

Figure 5 Fixed and adaptive approaches to optimize fitness given by the harmonic mean of synthesizability, drug-likeness, and affinity score. Changing to the adaptive approach after 5 generations results in an increase in fitness as shown by the histograms for drug-likeness and affinity score. The histograms were generated from the final population for the runs with the highest fitness for fixed and adaptive approaches. Sample molecules with similar chemical structures are shown for the adaptive approach. Mutations proposed by the language model show how modifications result in changes in the metrics used to calculate fitness.

ing geometry or chemical properties. However, SMILES enables the model to be used to train and make predictions on data without known three dimensional structures or previously calculated chemical properties, enabling searches and screening over large portions of chemical space. Furthermore, sequence-only models have been shown to compare favorably to more traditional approaches with manually defined features [39–41].

Molecule Generation through Mutations

In this work we have considered molecule generation for design using a language model to generate mutations. This strategy differs from other approaches to develop generative models, such as variational autoencoders [14, 42] and generative adversarial networks [15, 16]. The mutation strategy is dependent on an original molecule in which certain subsequences are changed rather than generating an entire molecule by sampling from a latent space or noise distribution. Although mutation relies upon an original molecule, and thus limits the amount of chemical space accessible for a given round of molecule generation, it has multiple benefits. First, mode collapse should not in principle present a problem for molecule generation through mutation. Because mutations are sampled from each molecule in the population, the full training set is represented in each generation of generated molecules. Second, each round of mutations can be manually inspected along with the scores for each respective molecule, enabling a user to better understand the types of mutations being generated and their impact on fitness. Furthermore, through multiple iterations of mutations and selection, large regions of chemical space can be explored [18], even though a single iteration remains close to the original data.

Conclusions

Masked language models coupled with genetic algorithms provide a useful framework for molecular optimization. During tokenization and pre-training, the model determines commonly occurring chemical sequences and rearrangements that can be leveraged for molecule generation through mutations. Furthermore, the language model can be refined using continued training on populations of molecules selected for desired properties. Here, we have shown that the continued training of a language model during genetic algorithm optimization provides a powerful approach to search for molecules according to both heuristic and ML model scoring functions. Models pre-trained on large compounds libraries serve as a useful starting point for both initial optimization from a poorly optimized dataset and initial weights for continued training.

Availability of data and materials

The source code for this work can be found at https://code.ornl.gov/candle/mlmol in the adaptive-lm directory.

Competing interests

The authors declare that they have no competing interests.

Funding

This research was funded by the AI Initiative, as part of the Laboratory Directed Research and Development Program of Oak Ridge National Laboratory, managed by UT-Battelle, LLC, for the U.S. Department of Energy (DOE); the Exascale Computing Project (ECP) (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.

Author's contributions

All authors contributed in developing the concept for the study. A.E.B., D.B., J. Gounley, and J. Glaser developed the code and trained the models. A.E.B. performed data analysis and generated the figures. All authors assisted in manuscript preparation.

Acknowledgements

An award of computer time was provided by the Innovative and Novel Computational Impact on Theory and Experiment (INCITE) program. This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

Author details

[1]Computational Sciences and Engineering Division, Oak Ridge National Laboratory, Oak Ridge, TN, 37831, USA. [2]National Center for Computational Sciences, Oak Ridge National Laboratory, Oak Ridge, TN, 37831, USA. [3]Biosciences Division, Oak Ridge National Laboratory, Oak Ridge, TN, 37831, USA. [4]Chemical & Biomolecular Engineering, University of Tennessee, Knoxville, TN, 37996, USA.

References

1. Dong, E., Du, H., Gardner, L.: An interactive web-based dashboard to track COVID-19 in real time. The Lancet Infectious Diseases **20**(5), 533–534 (2020). doi:10.1016/S1473-3099(20)30120-1
2. Blanchard, A.E., Gounley, J., Bhowmik, D., Chandra Shekar, M., Lyngaas, I., Gao, S., Yin, J., Tsaris, A., Wang, F., Glaser, J.: Language Models for the Prediction of SARS-CoV-2 Inhibitors
3. Minnich, A.J., McLoughlin, K., Tse, M., Deng, J., Weber, A., Murad, N., Madej, B.D., Ramsundar, B., Rush, T., Calad-Thomson, S., Brase, J., Allen, J.E.: AMPL: A Data-Driven Modeling Pipeline for Drug Discovery. Journal of chemical information and modeling **60**(4), 1955–1968 (2020). doi:10.1021/acs.jcim.9b01053
4. Chen, H., Engkvist, O., Wang, Y., Olivecrona, M., Blaschke, T.: The rise of deep learning in drug discovery. Drug Discovery Today **23**(6), 1241–1250 (2018). doi:10.1016/j.drudis.2018.01.039
5. Acharya, A., Agarwal, R., Baker, M.B., Baudry, J., Bhowmik, D., Boehm, S., Byler, K.G., Chen, S.Y., Coates, L., Cooper, C.J., Demerdash, O., Daidone, I., Eblen, J.D., Ellingson, S., Forli, S., Glaser, J., Gumbart, J.C., Gunnels, J., Hernandez, O., Irle, S., Kneller, D.W., Kovalevsky, A., Larkin, J., Lawrence, T.J., LeGrand, S., Liu, S.-H., Mitchell, J.C., Park, G., Parks, J.M., Pavlova, A., Petridis, L., Poole, D., Pouchard, L., Ramanathan, A., Rogers, D.M., Santos-Martins, D., Scheinberg, A., Sedova, A., Shen, Y., Smith, J.C., Smith, M.D., Soto, C., Tsaris, A., Thavappiragasam, M., Tillack, A.F., Vermaas, J.V., Vuong, V.Q., Yin, J., Yoo, S., Zahran, M., Zanetti-Polzi, L.: Supercomputer-Based Ensemble Docking Drug Discovery Pipeline with Application to Covid-19. Journal of Chemical Information and Modeling **60**(12), 5832–5852 (2020). doi:10.1021/acs.jcim.0c01010
6. Cho, E., Rosa, M., Anjum, R., Mehmood, S., Soban, M., Mujtaba, M., Bux, K., Moin, S.T., Tanweer, M., Dantu, S., Pandini, A., Yin, J., Ma, H., Ramanathan, A., Islam, B., Mey, A.S.J.S., Bhowmik, D., Haider, S.: Dynamic profiling of β-coronavirus 3cl mpro protease ligand-binding sites. Journal of Chemical Information and Modeling **61**(6), 3058–3073 (2021). doi:10.1021/acs.jcim.1c00449. PMID: 34124899. https://doi.org/10.1021/acs.jcim.1c00449
7. Chen, S.H., Todd Young, M., Gounley, J., Stanley, C., Bhowmik, D.: How distinct structural flexibility within sars-cov-2 spike protein reveals potential therapeutic targets, 4333–4341 (2021). doi:10.1109/BigData52589.2021.9671323
8. Bhowmik, D., Gao, S., Young, M.T., Ramanathan, A.: Deep clustering of protein folding simulations. BMC Bioinformatics **19**(S18), 484 (2018)
9. Yang, X., Wang, Y., Byrne, R., Schneider, G., Yang, S.: Concepts of Artificial Intelligence for Computer-Assisted Drug Discovery. Chemical Reviews **119**(18), 10520–10594 (2019). doi:10.1021/acs.chemrev.8b00728
10. Enamine *REAL* Database. https://enamine.net/compound-collections/real-compounds/real-database. Accessed: 2020-04-01 through https://virtual-flow.org/
11. Martins, I.F., Teixeira, A.L., Pinheiro, L., Falcao, A.O.: A Bayesian approach to in Silico blood-brain barrier penetration modeling. Journal of Chemical Information and Modeling **52**(6), 1686–1697 (2012). doi:10.1021/ci300124c
12. Subramanian, G., Ramsundar, B., Pande, V., Denny, R.A.: Computational Modeling of β-Secretase 1 (BACE-1) Inhibitors Using Ligand Based Approaches. Journal of Chemical Information and Modeling **56**(10), 1936–1949 (2016). doi:10.1021/acs.jcim.6b00290
13. RDKit: Open-source cheminformatics. http://www.rdkit.org
14. Jacobs, S.A., Moon, T., McLoughlin, K., Jones, D., Hysom, D., Ahn, D.H., Gyllenhaal, J., Watson, P., Lightstone, F.C., Allen, J.E., Karlin, I., Van Essen, B.: Enabling rapid COVID-19 small molecule drug design through scalable deep learning of generative models. International Journal of High Performance Computing Applications (2021). doi:10.1177/10943420211010930
15. Blanchard, A.E., Stanley, C., Bhowmik, D.: Using GANs with adaptive training data to search for new molecules. Journal of Cheminformatics **13**(1), 4–11 (2021). doi:10.1186/s13321-021-00494-3

16. De Cao, N., Kipf, T.: MolGAN: An implicit generative model for small molecular graphs. ICML 2018 workshop on Theoretical Foundations and Applications of Deep Generative Models (2018)
17. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing, 2nd edn. Springer, Springer-Verlag GmbH Germany (2015)
18. Virshup, A.M., Contreras-García, J., Wipf, P., Yang, W., Beratan, D.N.: Stochastic voyages into uncharted chemical space produce a representative library of all possible drug-like compounds. Journal of the American Chemical Society **135**(19), 7296–7303 (2013). doi:10.1021/ja401184g
19. Jensen, J.H.: A graph-based genetic algorithm and generative model/Monte Carlo tree search for the exploration of chemical space. Chemical Science **10**(12), 3567–3572 (2019). doi:10.1039/c8sc05372c
20. Brown, N., McKay, B., Gilardoni, F., Gasteiger, J.: A graph-based genetic algorithm and its application to the multiobjective evolution of median molecules. Journal of Chemical Information and Computer Sciences **44**(3), 1079–1087 (2004). doi:10.1021/ci034290p
21. Brown, N., Fiscato, M., Segler, M.H.S., Vaucher, A.C.: GuacaMol: Benchmarking Models for de Novo Molecular Design. Journal of Chemical Information and Modeling **59**(3), 1096–1108 (2019). doi:10.1021/acs.jcim.8b00839. 1811.09621
22. Lameijer, E.W., Kok, J.N., Bäck, T., Ijzerman, A.P.: The molecule evoluator. An interactive evolutionary algorithm for the design of drug-like molecules. Journal of Chemical Information and Modeling **46**(2), 545–552 (2006). doi:10.1021/ci050369d
23. Nicolaou, C.A., Apostolakis, J., Pattichis, C.S.: De novo drug design using multiobjective evolutionary graphs. Journal of Chemical Information and Modeling **49**(2), 295–307 (2009). doi:10.1021/ci800308h
24. Lameijer, E.W., Kok, J.N., Back, T., Ijzerman, A.P.: Mining a chemical database for fragment co-occurrence: Discovery of "chemical clichés". Journal of Chemical Information and Modeling **46**(2), 553–562 (2006). doi:10.1021/ci050370c
25. Schneider, G., Lee, M.L., Stahl, M., Schneider, P.: De novo design of molecular architectures by evolutionary assembly of drug-derived building blocks. Journal of Computer-Aided Molecular Design **14**(5), 487–494 (2000). doi:10.1023/A:1008184403558
26. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: 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 **1**(Mlm), 4171–4186 (2019). 1810.04805
27. Blanchard, A.E., Chandra Shekar, M., Gao, S., Gounley, J., Lyngaas, I., Glaser, J., Bhowmik, D.: Automating Genetic Algorithm Mutations for Molecules Using a Masked Language Model. IEEE Transactions on Evolutionary Computation (2022). doi:10.1109/TEVC.2022.3144045
28. Weininger, D.: SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. J. Chem. Inf. Comput. Sci. **28**, 31–36 (1998). doi:10.1021/ci00057a005
29. Schuster, M., Nakajima, K.: Japanese and korean voice search. In: 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 5149–5152 (2012). doi:10.1109/ICASSP.2012.6289079
30. Wu, Y., Schuster, M., Chen, Z., Le, Q.V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, Ł., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., Dean, J.: Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation, 1–23 (2016). 1609.08144
31. Bickerton, G.R., Paolini, G.V., Besnard, J., Muresan, S., Hopkins, A.L.: Quantifying the chemical beauty of drugs. Nature Chemistry **4**(2), 90–98 (2012). doi:10.1038/nchem.1243
32. Ertl, P., Schuffenhauer, A.: Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. Journal of Cheminformatics **1**(1), 1–11 (2009). doi:10.1186/1758-2946-1-8
33. jglaser/protein-ligand-mlp-1. https://huggingface.co/jglaser/protein-ligand-mlp-1
34. Aizman, A., Maltby, G., Breuel, T.: High performance I/O for large scale deep learning. In: 2019 IEEE International Conference on Big Data (Big Data), pp. 5965–5967 (2019). IEEE
35. Ramakrishnan, R., Dral, P.O., Rupp, M., Von Lilienfeld, O.A.: Quantum chemistry structures and properties of

134 kilo molecules. Scientific Data **1**, 1–7 (2014). doi:10.1038/sdata.2014.22

36. gdb9 Dataset. http://deepchem.io.s3-website-us-west-1.amazonaws.com/datasets/gdb9.tar.gz. Accessed: 2021-05-28
37. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T.L., Gugger, S., Drame, M., Lhoest, Q., Rush, A.M.: Transformers: State-of-the-art natural language processing. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pp. 38–45. Association for Computational Linguistics, Online (2020). https://www.aclweb.org/anthology/2020.emnlp-demos.6
38. Rajbhandari, S., Rasley, J., Ruwase, O., He, Y.: Zero: Memory optimizations toward training trillion parameter models. International Conference for High Performance Computing, Networking, Storage and Analysis, SC **2020-Novem**, 1–24 (2020). doi:10.1109/SC41405.2020.00024. 1910.02054
39. Wang, S., Guo, Y., Wang, Y., Sun, H., Huang, J.: 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, 429–436 (2019). doi:10.1145/3307339.3342186
40. Xue, D., Zhang, H., Xiao, D., Gong, Y., Chuai, G., Sun, Y., Tian, H., Wu, H., Li, Y., Liu, Q.: X-MOL: large-scale pre-training for molecular understanding and diverse molecular analysis. bioRxiv (2020). doi:10.1101/2020.12.23.424259
41. Kim, H., Lee, J., Ahn, S., Lee, J.R.: A merged molecular representation learning for molecular properties prediction with a web-based service. Scientific Reports **11**(1), 1–9 (2021). doi:10.1038/s41598-021-90259-7
42. Gómez-Bombarelli, R., Wei, J.N., Duvenaud, D., Hernández-Lobato, J.M., Sánchez-Lengeling, B., Sheberla, D., Aguilera-Iparraguirre, J., Hirzel, T.D., Adams, R.P., Aspuru-Guzik, A.: Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules. ACS Central Science **4**(2), 268–276 (2018). doi:10.1021/acscentsci.7b00572. 1610.02415