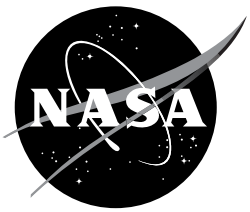
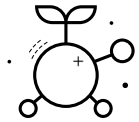


Signatures of Life: Learning Features of Prebiotic and Biotic Molecules

Aaron C. Bell, Insight Edge Inc., Tokyo, Japan
Timothy D. Gebhard, MPI-IS, Tübingen, Germany & ETH Zurich, Switzerland
Jian Gong, Massachusetts Institute of Technology, Cambridge, MA, USA
Jaden J. A. Hastings, XO.LABS, Los Angeles, CA, USA

Atılım Güneş Baydin, University of Oxford, United Kingdom
G. Matthew Fricke, University of New Mexico, NM, USA
Michael Phillips, Johns Hopkins University, Baltimore, MD, USA
Kimberley Warren-Rhodes, SETI Institute, Mountain View, CA, USA

Nathalie A. Cabrol, SETI Institute, Mountain View, CA, USA
Massimo Mascaro, Google Applied AI, Mountain View, CA, USA
Scott Sandford, NASA Ames Research Center, Mountain View, CA, USA



ASTROBIOLOGY

October 29, 2022

ACKNOWLEDGMENT

This work has been enabled by the Frontier Development Lab (FDL.ai). FDL USA is a collaboration between several government agencies, Department of Energy (DOE), National Aeronautics and Space Administration (NASA), and U.S. Geological Survey (USGS), SETI Institute, and Trillium Technologies Inc., in partnership with private industry and academia. This public/private partnership ensures that the latest tools and techniques in Artificial Intelligence (AI) and Machine Learning (ML) are applied to basic research priorities in support of science and exploration of material concerns to human kind.

NASA Cooperative Agreement boilerplate, from Terms and Conditions GCAM D3:

(1) All information disseminated as a result of the award shall contain a statement which acknowledges NASA's support and identifies the award by number (e.g., "the material is based upon work supported by NASA under award No(s) NNX14AT27A.").

(2) Except for articles or papers published in scientific, technical, or professional journals, the exposition of results from NASA supported research should also include the following disclaimer: "Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Aeronautics and Space Administration."

CONTENTS

ABSTRACT

The search for life beyond Earth is complicated by the lack of a consensus as to what "life" actually is—especially if we also want to consider potential forms of life that do not not resemble anything that we have encountered thus far on Earth. Various so-called *agnostic* biosignatures have been proposed already, but one that has received particular attention lately is the concept of molecular complexity. The key hypothesis is that life creates not only more complex molecules, but also greater abundances of complex molecules, than purely abiotic processes. Known challenges with this approach are, for example, that the calculation of molecular complexity metrics can be computationally expensive (depending on the chosen definition of complexity), and that ultimately, we need to be able to actually measure molecular complexity *in situ*, for example on board of a spacecraft probing the surface of another planet.

In this FDL 2022 challenge on Astrobiology, we seek to tackle these challenges through the use of machine learning. As a first step, we generate a large dataset of molecules with corresponding complexity scores, which we plan to make publicly available to the community. Using this dataset, we then illustrate on two tasks the potential benefits of machine learning: First, we show that we can learn models that predict the complexity of a molecule from a suitable representation (e.g., a SMILES string) with low relative errors (less than 5% on average) and at a significantly greater speed than existing baselines. Second, we demonstrate that machine learning models can infer the complexity of a molecule directly from its mass spectrum, with a significantly lower error than the existing proof-of-concept from the literature. This is a first step towards measuring molecular complexity in the field, and may help open new doors for critical robotic missions where autonomous decision-making is required. After all, even if we do not find life beyond Earth, being able to determine the molecular complexity of samples *in situ* can help inform decisions such as which areas to prioritize for exploration, or which data to send back to Earth for detailed analysis.

INTRODUCTION

In this section, we present the idea and motivation of our FDL challenge, introduce the concept of molecular complexity, and discuss the goals and desired outcomes of our project.

Motivation: The Search for Life Beyond Earth

One of the three "big questions" that NASA pursues in its research is: "Are we alone, or does life, be it similar to our own or not, exist elsewhere?" (NASA, 2020). It is a profound questions that remains one of the most challenging frontiers in modern science, and one that is complicated by the challenge of defining the qualities we may be searching for as signs of "life"? There are well over a hundred different definitions of the term in scientific literature (Trifonov, 2011), and new ones keep being proposed; see Bartlett and Wong (2020) for a recent example. Our definitions, however, are presently constrained to our experience of life in its myriad forms on Earth, and its specific range of environmental conditions. There is every expectation that "extraterrestrial life might differ substantially from life on Earth" (Des Marais and Walter, 1999), so how can we be sure that whatever we decide to use as a potential biosignature[1] is also capable of recognizing "life as we do not (yet) know it"? The study of *agnostic* biosignatures concerns itself with the study of "frameworks and techniques for universal life detection that do not presuppose any particular molecular framework (Cronin and Walker, 2016) or evolutionary endpoint (Cabrol, 2016)" (National Academies of Sciences, Engineering, and Medicine, 2019).

One possible agnostic biosignature that has received considerable attention lately is the concept of molecular complexity; an idea that we will introduce more thoroughly in the next sections. However, as motivating examples, we note already that not only has it been hypothesized that increasingly more complex molecules are likely a prerequisite to the emergence of life (see Walker et al., 2017, and references therein), but the 2022 *Decadal Strategy for Planetary Science and Astrobiology* explicitly addressed the idea of molecular complexity as a potential biosignature, noting that the molecular assembly index of Marshall et al. (2017) "can [...] be useful in distinguishing biotic from abiotic molecules" (National Academies of Sciences, Engineering, and Medicine, 2022).[2]

Our brief in the 2022 Astrobiology Concept Note, which was titled *Signatures of Life: Learning Features of Prebiotic and Biotic Molecules*, proposed the following challenge:

> "Finding life in the Solar System is an important frontier in modern science and a core aspect of many NASA missions. The problem is difficult because we lack information about what "life as we don't know it" can look like, as we have not yet observed any living systems beyond the Earth. One promising approach to defining universal life signatures is to use universal principles in assigning complexity measures to molecules. **We will work on building datasets and machine learning approaches based on**

[1] A *biosignature* is commonly defined as an "object, substance, and/or pattern whose origin specifically requires a biological agent" (Des Marais et al., 2008).

[2] The precise question considered in the Decal Strategy is: "What is the extent of molecular complexity (e.g., size, heteroatom diversity, structure, pathway assembly index) and degree of organization (e.g., isomeric preference, polymerization) that can be generated abiotically under habitable conditions? How does this compare to prebiotic experiments to date?" (Q11.1b)

molecular complexity in prebiotic chemistry, to inform future space missions such as NASA's Dragonfly mission to Titan."

Key Hypothesis: Only Life Creates Complexity in Abundance

To understand why molecular complexity has been considered as a potential agnostic biosignature, one needs to understand the following postulate, which we call the *key hypothesis*. In simple terms, the key hypothesis states that, for a suitable definition of what "molecular complexity" means (see below), we expect that it is statistically unlikely that the presence of larger quantities of a particular complex molecule in a given environmental sample is due to random, abiotic processes. Consequently, if we find larger quantities, or proportions, of molecules within a given sample that are highly complex, it follows that there is likely some form of biologically driven activity at work within the local environment.

Background: Notions of Molecular Complexity

The concept of molecular complexity (MC) is itself complex (Randić et al., 2005), and many different metrics have been proposed in the literature; see, for example, the introduction of Böttcher (2016) for an overview. Most fundamentally, complexity is a mapping that assigns a scalar number to a molecule (often using some graph- or information-theoretic invariants), which then allows to order molecules from less to more complex in a somewhat "intuitive" way. As Randić et al. (2005) describe it: "Most authors appear to agree that the complexity of mathematical objects such as chemical structures increases with information content, molecular size, connectivity of the graph, molecular branching, cyclicity of molecules, multiplicity of bonds, and the presence of heteroatoms (coloring of graphs), while it should decrease with increasing symmetry properties of objects under consideration." An even more extensive list of desiderata is found in Bonchev and Polansky (1987).

The original proposal for this challenge was focused on using the molecular assembly index of Marshall et al. (2017) as a metric of molecular complexity. During the course of the project, we have decided to extend this to the following list of three complexity measures, which we believe to represent conceptually different and complementary approaches:

1. **Bertz Complexity** C_T: The "first general index of molecular complexity" (Bertz, 1981). It combines concepts from graph and information theory and is defined as $C_T = C(\eta) + C(E)$, where $C(\eta)$ describes the bond structure and $C(E)$ the complexity due to heteroatoms. Calculating C_T is fast and scales linearly with the molecule size. When we generate our datasets, we compute C_T using the `BertzCT` method from RDKit (The RDKit Team (G. Landrum et al.), no date).
2. **Böttcher Complexity** C_m**:** An information-theoretic metric that is based on the information content in the micro-environments of all atoms (Böttcher, 2016; 2017). It is additive and simple to calculate even for large molecules. Our computation of C_m uses a freely available open source implementation (Boskovic Group, 2020).
3. **Molecular Assembly index (MA):** Also known as pathway complexity, the MA represents the minimum number of steps required to assemble a molecule from fundamental building blocks (Marshall et al., 2017). MA is claimed to be particularly well-suited to biosignature

detection for it can be determined experimentally (Marshall et al., 2021). It is at least as hard as NP-complete to compute (Liu et al., 2021), requiring hundreds of CPU hours even for moderately sized molecules. We have use a (currently non-public) implementation kindly provided by the authors of Marshall et al. (2021) for our calculations.

In the explication of their Molecular Assembly Index (MA), Marshall et al. (2021) reasoned that "high MA molecules cannot form in detectable abundance through random and unconstrained processes, implying that the existence of high MA molecules depends on additional constraints imposed on the process." But, if true, one might conjecture that similar arguments could also be made for other measures of complexity.

We note that very recently (that is, after the end of the eight-week FDL sprint), the Molecular Assembly Index (and Assembly Theory as a whole) has attracted strong criticism from Uthamacumaran et al. (2022), who suggest that MA is just a special case of Huffmann's encoding, and question the claims that MA is more suitable as a biosignature than other metrics.

The Bertz and Böttcher scores can be considered as "top down" metrics, because they consider the entire molecule by summing up the information content from all subgraphs (Böttcher, 2016). In contrast, MA can be considered a "bottom up" complexity metric, because it considers the pathway through which a molecule is constructed from a shared pool of building blocks, starting from the fundamental molecular bonds that connect atoms (Marshall et al., 2017; Liu et al., 2021).

All three metrics are intrinsic to a given molecule (i.e., they do not depend on the external environment), and they characterize different aspects of the molecular chemical space and information content; hence we consider them, to an extent, to be complementary. Of course, in practice, one may be interested not only in the complexity of individual molecules, but also in ensembles of molecules or entire reaction networks. Liu et al. (2021) have studied how the definition of the molecular assembly index can be extended to groups of molecules, taking into account potential synergies in the synthesis pathways.

Task: Science Questions and Approach

In the following, we cite the essential parts of "science questions and approach" section of the challenge summary as it was given to us at the start of the project. The rationale here is that, to properly assess the outcomes of our work, we believe one first needs to understand the inputs that defined the starting point and the scope of the project.

> We will work on machine learning (ML) and simulation-based inference (Cranmer et al., 2020) approaches making use of MA index datasets and MA computation software made publicly available[3] by Marshall et al. (2021), in order to develop ML techniques relevant for the detection of universal signatures of life. Crucially, as a first step we will be working on creating a unified dataset of prebiotic molecules and their MA indices that we expect to inform future missions including NASA's Dragonfly mission to Titan, the largest moon of Saturn.
>
> The objectives of this project are as follows:
>
> 1. Identify data sources and pre-process data that can be used for ML, including the supplementary material made available by Marshall et al. (2021), other existing scientific databases, and

[3] Supplementary information: https://www.nature.com/articles/s41467-021-23258-x#Sec15

simulations for synthetic data generation. Compile these into a well-documented ML-ready unified dataset that can be shared with the research community. The dataset should include the following as a minimum:

   a) Molecules that have been found in meteorites (Oba et al., 2020).

   b) The majority of (and if possible, all) molecules discovered in prebiotic chemistry.

2. Formulate ML approaches to tackle the problem of characterizing and detecting biotic versus abiotic molecules, building on the MA theory and the state-of-the-art in relevant fields of science such as computational chemistry. The ML approaches should follow the best practices (Artrith et al., 2021) and might include the following:

   a) Classification of molecules (or groups of molecules) as biotic or abiotic based on the MA index and other features that can be learned via state-of-the-art ML techniques such as graph convolutional neural networks (Kipf and Welling, 2016).

   b) Surrogate modeling approaches (e.g. Shirobokov et al., 2020; Poduval et al., 2021; Himes et al., 2022) that can make the MA index computation significantly faster and parallelizable, in order to enable ML-onboard deployment (e.g., Mateo-Garcia et al., 2021) on a robotic deep space mission with autonomous life detection capability.

   c) Simulation-based inference and probabilistic programming (Baydin et al., 2019) using the MA index computation model by Marshall et al. (2021) in a model-based probabilistic ML setting.

3. Produce proof-of-concept code and run reproducible experiments demonstrating whether ML approaches relevant for the detection of universal signatures of life can be feasible.

We expect the project to produce several important outcomes. Firstly, approaching the problem of life signature detection with state-of-the-art ML techniques, which have been successfully deployed for complex pattern recognition tasks in many domains (Goodfellow et al., 2016), and releasing the datasets we will construct in the process will be important contributions that we expect to pioneer similar follow up work in ML.

Beyond this, we expect our datasets and results to inform autonomous life detection capability on NASA missions to destinations with significant time delay in communications, such as NASA's Dragonfly mission to Titan.

A vision we have is that an ML-based end-to-end pipeline can be constructed that takes in raw data from a mass spectrometry sensor onboard a mission, and produces molecular complexity distributions (MA index histograms) on-the-fly to select interesting targets and samples autonomously. Existing work involving members of our research team (e.g. Shirobokov et al., 2020; Poduval et al., 2021; Mateo-Garcia et al., 2021; Himes et al., 2022) demonstrates that fast ML-based pipelines of complex computational models can be constructed and deployed onboard space hardware with constrained computation and energy budgets.

Analysis: The Case for Machine Learning

Starting from the descriptions on the challenge summary, we began our challenge by researching and exploring the idea of molecular complexity as an agnostic biosignature, and analyzed

possible directions for our work. Ultimately, we identified three potential applications for which we believe machine learning may be useful. These are:

1. Speeding up the computation of the molecular complexity index by learning surrogate models that can predict or estimate the MA from a given molecular representation, such as a graph or a string. This may directly help us to make (approximate) MA values accessible for a larger space of molecules than what is currently possible.
2. Inferring the molecular complexity of a molecule directly from instrument data, more specifically, mass spectrometry data. This is highly relevant for the practicality of molecular complexity as a potential biosignature, because it could enable future spacecrafts to perform fast MC measurements *in situ*.
3. "Hallucinating" molecules for a given molecular complexity, that is, learning a conditional generative model that can, for a given target complexity, produce and suggest new molecules with that complexity. This might become a useful explorative tool to better understand the space of complex molecules.

We will discuss each of these three applications in more detail in the following subsections.

Speeding up the computation of MA

The molecular assembly index, also known as pathway complexity, is a conceptually simple yet very appealing notion of molecular complexity: It is basically defined as the number of steps required to construct a molecule from fundamental building blocks. The biggest challenge with MA is, however, that its computational complexity is at least NP complete (Liu et al., 2021)—in other words, it becomes very expensive very quickly to compute the MA of larger molecules. Different solutions have been proposed to deal with this challenge, such as a Monte Carlo approach (Liu et al., 2021), or the idea of recursive pathway complexity (Marshall et al., 2017) which provides an upper bound on the exact assembly index.

We suggest that learning-based methods may provide yet another fruitful approach to tackle the computational challenges of calculating MA. Of course, such a system will, in general, not produce an exact solution (after all, machine learning is no magic bullet that can circumvent computational complexity), but it could provide an approximate solution very fast. Depending on the specific context, such an approximate solution could then either be "good enough" already, or it could help to inform an exact algorithm and help narrow down the search space. The usefulness of learning-based, data-driven approximations to NP-hard problems has already been demonstrated in the literature (see, e.g., Milan et al., 2017).

We have identified, in principle, two different ways in which we could employ machine learning to help scale up the computation of MA values:

1. *Guided Tree Search.* The standard algorithm to computing MA is essentially a branch-and-bound tree search: Inside the (combinatorially large) assembly tree, we have to find the shortest path from the given building blocks to our target molecule. How *exactly* we explore this search tree, however, is up to us. One potential approach we see is to employ machine learning to find good heuristics that guide the exploration of the search space, similar to the ideas presented in, for example, Li et al. (2018) or Parascandolo et al. (2020). Using

deep neural networks to guide Monte Carlo tree search is also one key elements of the recent successes of *AlphaZero* (Silver et al., 2018) and *AlphaTensor* (Fawzi et al., 2022). Monte Carlo Tree in combination with reinforcement learning has also been proposed in the cheminformatics literature to find new synthesis pathways (Wang et al., 2020); a task that seems conceptually related to MA computation.

The advantage of this approach is in fact two-fold: On the one hand, optimizing the way in which we traverse the search tree increases the chances that we find a "good" solution given only a limited computational budget. On the other hand, given sufficient computational budget, the obtained solutions are still exact—after all, we are still running the same tree search as before, just in a heuristically controlled order.

Unfortunately, in practice, such a guided tree search using learned heuristics would require extensive modifications at the lowest level to the implementation of the MA calculation, which we deemed not feasible given the very short time frame of the FDL research sprint. While still promising from our perspective, we have therefore decided to forego the idea at this time.

2. *Regression.* Conceptually and implementation-wise much easier, the idea here is to use a supervised learning approach and learn a regression model to predict the MA of a molecule directly from a representation of that molecule (e.g., a string). For this approach, we do not have to make any modifications to code that computes MA—all we need is a sufficiently large dataset of molecules with corresponding complexity values which we use to train our model.

   The key challenge with all this is, of course, essentially a matter of out-of-distribution generalization. How can we make sure that the model that we train on molecules with an MA of, say, 0 to 20 also works well for molecules with a (true) MA of 50? This is in fact a rather fundamental issue, because for sufficiently complex molecules, the classic computation is intractable, meaning that we do not even have ground truth data to evaluate the extrapolation or generalization performance of our model.

   Despite this obvious and crucial challenge, we have decided that the regression approach is more practical and more likely to produce some first results before the end of the program.

We will discuss our practical implementation and preliminary results for this task in the next section.

Inferring molecular complexity from mass spectrometry data

Molecular complexity per se is a rather theoretical concept, in the sense that any given notion of it usually comes with a formula or an algorithm that describe how to compute it for a given representation of a molecule (e.g., a graph). However, for MC to be useful as a potential biosignature also in practice, for example during a future space mission, we need to be able to determine it *in situ* for a given sample.[4] One prime candidate to enable this is mass spectrometry (MS): Virtually all upcoming planetary exploration missions, such as the *Dragonfly* mission to Titan (Lorenz et al., 2018; Grubisic et al., 2021), or the proposed *Europa Lander* (Hand et al., 2022) and *Enceladus Orbilander* (MacKenzie et al., 2021), will carry mass spectrometers to analyze their targets. Equipping

[4] Note how in practice, this is further complicated by the fact that we cannot simply sample a single molecule from an environment. Instead, we are always dealing with mixtures of compounds (which may or may not form an entire reaction network), and in the long run, we might even be more interested in characterizing the complexity of environments than of single molecules.

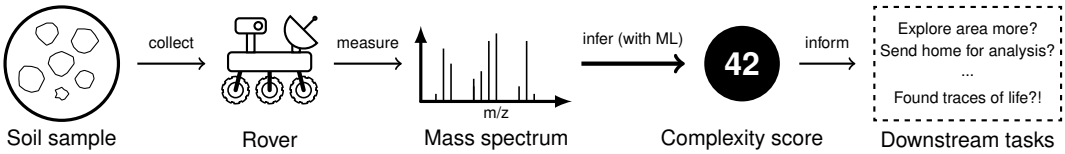


Figure 1: A schematic workflow that illustrates where our work on inferring MC from MS data fits into the "bigger picture" of space exploration and the search for traces of extraterrestrial life.

these missions with the ability to infer MC from MS data may be useful for a variety of tasks even beyond the search for traces of extraterrestrial life. For instance, information about the molecular complexity of a sample may inform decisions such as which areas to prioritize for exploration, or which data to send back to Earth for more detailed analysis. We illustrate in figure 1 a schematic workflow and highlight where the inference of MC from MS data fits into the "bigger picture."

A naïve approach to infer MC from a sample could then look like this: In a first step, we solve the inverse problem of mass spectrometry, that is, given a mass spectrum, we infer from it the molecule to which it corresponds. In a second step, we can then run any algorithm of choice to compute the MC. In practice, of course, this naive approach is not very practical: First, inverting a mass spectrum is a challenging problem in its own right—see, for example, Wei et al. (2019) or Zhang et al. (2022) for recent machine learning-based approaches to this problem. Second, depending on the choice of MC metric, computing the complexity of the identified compound may easily exceed the computational capabilities of a spacecraft. Sending all data back to Earth for analysis may also not be an option, for deep space communications are generally expensive, low-bandwidth, and have large round-trip delays. It is not least for these reasons that the community has identified a clear need for spacecraft autonomy and is recognizing machine learning as one potential tool to achieve this goal (Theiling et al., 2022).

Looking at the shortcomings of the naive approach and keeping in mind the aforementioned need for spacecraft autonomy, the question arises if there is a perhaps a more direct way to infer molecular complexity from mass spectrometry data. The answer, at least in the case of MA, is yes: Marshall et al. (2021) have already demonstrated that it is possible to infer MC *directly* from MS data by showing that there is a clear correlation between the number of peaks in a high-resolution tandem MS of a molecule and its MA.

We believe that one worthwhile application for machine learning would be to take these studies further and investigate if we can improve on the first proof of principle by using more advanced regression models. Going beyond the molecular assembly index, we can also have a first look at other complexity metrics and to what extent they can be inferred directly from mass spectra, which is something that, to the best of our knowledge, has not been done before.

Hallucinating molecules of a given target complexity

The idea to use conditional generative models to "dream up" new molecules with given target properties is not a new one—the feasibility of this (in the context of *de novo* design of drug-like molecules) has already been demonstrated, for example, by Lim et al. (2018). We believe that, in principle, it should be rather straightforward to extend their method also to molecular complexity. However, due to the time constraints of the FDL research sprint, we did not explore this idea further.

Goals: Desired Outcomes for the Challenge

Following our analysis above, as well as discussions with our stakeholders at NASA, we set ourselves the following three goals for our project. These goals are aimed at measuring our own performance while ensuring outputs that are useful for the astrobiology community.

1. The **Minimum Goal**, defined as what we are confident we can achieve even in the worst case and thus consider the minimal outcome for the project to be considered a success, is to collect and generate a dataset of molecules and corresponding meta-information, including molecular complexity values. This dataset should be well-documented and publicly accessible, in order to be useful for the broader scientific community. Once published, we would encourage the community to find creative ways to examine the dataset and test interesting ideas.
2. The **Target Goal**, given by what we would like to achieve if things (mostly) go to plan, is to use the dataset we created and use it for the various machine learning experiments described in the previous section.
3. The **Bold & Crazy Goal**, or moonshot goal, is something that is well beyond what we believe we can achieve during the research sprint, but serves as "Northern star" of sorts to guide our efforts and help us with the decisions that we will have to make along the way. Our bold and crazy goal was to construct a (prototype of) a machine learning-based systems that could be deployed on a future spacecraft to enable *in situ* measurements of molecular complexity. This would involve taking into account the realistic hardware limitations, as well as considering how to streamline our ML computation algorithms to fit within limited CPU and memory space.

One additional goal, which would be located somewhere between the target and the moonshot goal in terms of ambition, and whose category would perhaps best be described as "if time permits", would be use our dataset to explore the relationship between molecular complexity and ensembles of molecules that form networks of chemical reactions.

Finally, we highlight one more thing that we explicitly chose **not** to make a goal for our challenge, namely, the (machine learning-based) classification of molecules into biotic and abiotic (and possibily prebiotic). The reason, quite simply, is that there is no universally agreed-upon definition of these terms, and no existing labeled data (at scale).

PROCEDURE

This section outlines the approach that we took to generate our dataset, describes the machine learning experiments that we conducted using this data, and discusses our findings and results.

Dataset development

Dataset generation Our minimal goal (see previous section) was to create a dataset of molecules with associated complexity scores that aims to be useful to the wider astrobiology community. The approach that we chose for this was a two-step process where we first curate, from different sources, a minimal list of relevant molecules, and then subsequently augment this list with additional information such as complexity scores, number of atoms and bonds, or mass spectral data.

The first step, also known as the initial curation process, was partially automatic (by downloading lists of molecules from existing databases), and partially manual (i.e., we collected molecules found in relevant publications, such as in the study of organic molecules found meteorites and lab experiments). For each molecule, we started by collecting a minimal set of identifiers and metadata, consisting of:

1. `inchi`, a string with the *International Chemical Identifier* (InChI) of the molecule (Heller et al., 2013; Heller et al., 2015),
2. `smiles`, a string with the (non-unique) *Simplified molecular-input line-entry system* (SMILES) representation of the given molecule (Weininger, 1988),
3. `inchikey`, a fixed-length identifier computed from the SHA-256 hashed of the InChI string,
4. `origin`, the source environment of the molecule (if known),
5. `reference`, the DOI of the publication from which we took the molecule (if applicable), and
6. `comment`, an extra field which can store any additional metadata.

Only one of the first three columns is required. If multiple entries are present, we select the first non-empty column (using the given order) and use it to identify the molecule and populate the other fields.

In a second step, we then augment this minimal list and add additional columns. A full overview is given in table 2. This step combines different data sources: Some information, such as the number of atoms and bonds, can be directly obtained, for example, from the InChI string, which we do using the RDKit library. Other information need to be queried from online databases, such as the CAS number (CAS, 1978), or any look-ups of the InChI key. For this, we make use of the CIRpy library , which provides a simple Python interface for the Chemical Identifier Resolver (CIR) by the CADD Group at the NCI/NIH. For the computation of the different complexity scores, we use a wrapper package that we developed ourselves, complexipy, which we explain in more detail below. We set a time limit of 240 hours (ten days) for the computation of the MA, and many molecules failed because they exceeded this limit. Finally, to obtain mass spectral data, we use the CAS number of a molecule to query the NIST Chemistry WebBook (NIST, 2022) through its public API. (This step actually succeeded only for a small fraction of all molecules.)

Due to the large number of molecules that we wanted to process, the data augmentation was run on Google Cloud, using hundreds of compute-optimized nodes (i.e., C2-type instances) in parallel. Each node ran a custom-made Docker container and was assigned a random subset of the original

minimal list from step 1. The results from each node where uploaded into a Google Cloud Bucket, where they were combined into a single file. All inputs and outputs used the CSV file format.

The final dataset curated during the FDL sprint consists of 407 240 molecules. For a subset of 17 021 of these molecules, we were also able to obtain mass spectrometry data from NIST. We are showing an overview of the distribution the complexity scores in our dataset in figure 2. The major limitation of the dataset generation was the computation of MA, which took well-over 100 000 CPU hours just for the molecules where it ultimately suceeded. Even more CPU hours were spent on molecules which ultimately exceeded the 10-day threshold for the MA computation. By comparison, computing all Böttcher and Bertz scores together took only around 200 minutes of CPU time.

All code that we have used to generate our dataset(s) is available on GitLab.[5]

Complexity computations and complexipy: Not least due to the strict time constraints of the program, we relied on existing implementations for the computation of our three complexity metrics:

1. To compute the Bertz score, we use the `BertzCT` function from RDKit, a powerful open source cheminformatics package that we also use for other computations and conversions.
2. For the Böttcher score, we use the implementation by the Boskovic Group at the University of Kansas, which is available from GitHub under a BSD 3-Clause License. To ensure that the code also works for molecules that contain "non-bonds," which are indicated by a "." in the SMILES string.[6] We have also considered other implementations, including a publicly available implementation by the Forli Lab at Scripps Research. When comparing the different versions, we found that there were sometimes significant differences between the scores returned by the different implementations. Our ultimate decision to use the implementation of the Boskovic Group was mainly based on the fact that it only requires RDKit, but not OpenBabel—another cheminformatics library, which we found non-trivial to install.
3. Public implementations of the MA computation (in C++) are available from the supplementary materials of Marshall et al. (2021) and Liu et al. (2021). However, for our work here, we have made use of a newer, Go-based implementation of the MA computation, which the Cronin Group has kindly agreed to share with us in private, and which is not yet publicly available.

To simplify the usage of the three different code bases during our data generation process, we have written `complexipy`, a simple Python package which constitutes a very thin wrapper around the existing implementations described above, and whose main purpose is to provide a single unified interface for computing molecule complexity scores. Since `complexipy` heavily depends on other people's intellectual property, we have decided not to make it publicly available for the moment.

Extra features: Besides the default set of features summarized in table 2, there is a smaller set of features that we have used for some of our machine learning experiments that are not generated by the main data generation pipeline described above, but are only available through extra scripts. The reason for this is very pragmatic: When we realized that these additional features might be useful for our ML work, the main data generation was already in full swing, and we did not want to abort and restart that process. Additionally, computing these extra features is comparably fast and does not need to be parallelized, but can be run on a suitable machine in a few minutes.

[5] https://gitlab.com/frontierdevelopmentlab/2022-us-astroinfosignatures/dataset-generation

[6] A simple example of such a non-bond is aqueous sodium chloride, which may be written as `[Na+].[Cl-]`.

Table 1: Overview of the chemical databases that we utilized for this FDL challenge.

Database	Description and reference
NIST Chemistry WebBook	Chemical and physical property data for chemical species.
PubChem	Publicly available database of chemical information. See [1].
Reaxys	Expert-curated database owned and operated by Elsevier.
GBD-17	Enumerated 166 billion small organic molecules. See [2].
Rad-6	Enumerated organic molecules and reactions. See [3].
GNPS	Natural product mass spectrometry database. See [4].
Manual	Manually collected molecules from various papers.
MassBank US (MoNA)	2M+ million mass spectra of experimental and in silico origin.
Lipid MAPS Structure DB	47k+ unique structures for biologically relevant lipids.
METLIN Gen2	860k+ molecular standards for tandem mass spectrometry.

References: [1] Kim et al. (2021) [2] Ruddigkeit et al. (2012) [3] Stocker et al. (2020) [4] Wang et al. (2016)

Table 2: Overview of the default features (columns) in our generated dataset.

Name	Description
`inchi`	International Chemical Identifier.
`smiles`	Simplified Molecular-input Line-Entry System.
`inchikey`	SHA-256-hashed version of the full InChI string.
`formula`	Traditional chemical formula.
`cas_number`	Chemistry Abstracts Service (CAS) registry number.
`iupac_name`	Official IUPAC name.
`names`	Other common and scientific names.
`num_atoms`	Number of atoms in the molecule (excluding hydrogen).
`num_atoms_all`	Number of atoms in the molecule (including hydrogen).
`mol_weight`	Molecular weight (in mol).
`complexity_bhi`*	Bertz complexity score (Bertz, 1981).
`complexity_boettcher`	Böttcher complexity score (Böttcher, 2016).
`complexity_ma`	Molecular Assembly index (Marshall et al., 2017).
`complexity_bhi_runtime`	Time (in seconds) to calculate `complexity_bhi`.
`complexity_boettcher_runtime`	Time (in seconds) to calculate `complexity_boettcher`.
`complexity_ma_runtime`	Time (in seconds) to calculate `complexity_ma`.
`mass_spectrum_nist`	Mass spectrum from NIST in JCAMP-DX format (if available).
`origin`	Primary environment of the species.
`reference`	DOI of publication (if applicable).
`comment`	Any additional meta-information.

* The abbreviation "BHI" (short for Bertz-Hendrickson-Ihlenfeldt) is a historic artifact: We originally intended to compute this index (which is the default complexity score available from PubChem); however, we did not find a publicly available implementation and therefore ultimately reverted to the regular Bertz complexity.

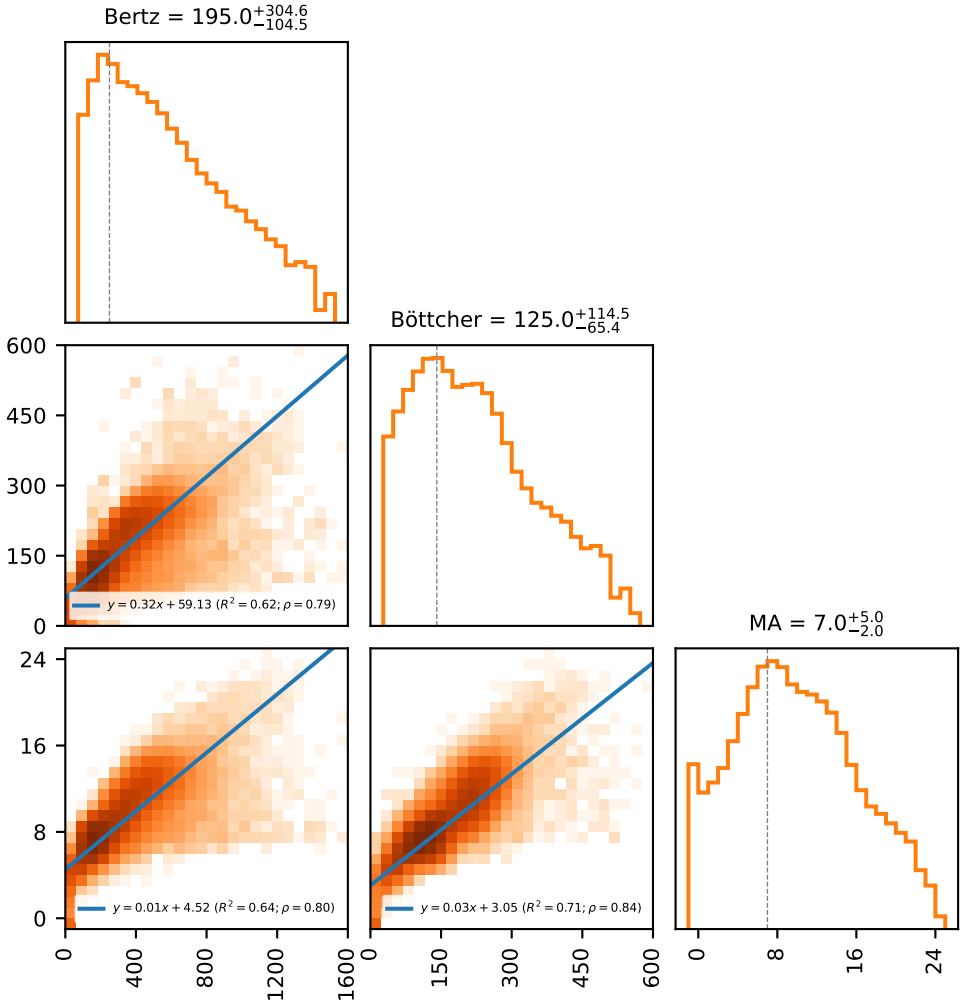


Figure 2: (Joint) distributions of the three complexity scores in our dataset. All histograms (1D and 2D) use a logarithmic scale. The line shows in the 2D histogram shows the best linear fit, for which we also report the coefficient of determination R^2, as well as the Pearson correlation coefficient ρ.

Table 3: Overview of the additional features used for some of our experiments. These features are not produced by default by our data generation pipeline, but are available through extra scripts.

Name	Description and link to used implementation	
`mol2vec`	Embedding computed using mol2vec (Jaeger et al., 2018).	
`mte`	Molecular transformer embedding (Morris et al., 2020).	
`rdkit_fingerprint`	2048-bit molecular fingerprint computed by RDKit.	
`selfies`	SELFIES representation of the molecule (Krenn et al., 2020).	

These extra features, which we also summarize in table 3, are the following:

mol2vec A 300-dimensional vector embedding that is obtained by passing a molecule to a pre-trained instance of the mol2vec model of Jaeger et al. (2018), which is based on the seminal word2vec model introduced in Mikolov et al. (2013). We chose this model because it was one of the first that used machine learning to generate embeddings of molecules.

mte A 512-dimensional vector embedding that is obtained by passing a molecule to a pre-trained instance of the transformer model of Morris et al. (2020). By default, the model returns a $512 \times n$ matrix for a model whose SMILES string representation has lengths n. To obtain a one-dimensional, fixed-sized representation, we take the mean along the second dimension. Our motivation to use this particular model was rather pragmatic: It was the first pre-trained transformer model for molecules that managed to run on our data. Other models that we considered, including for example NVIDIA's MegaMolBART, turned out to be more complicated to use, and we decided not to invest too much time here.

rdkit_fingerprint A molecular fingerprint in the form of a 2048-bit vector, computed using the default algorithm of the RDKit library. There is a large number of fingerprinting algorithms available (see, e.g., Cereto-Massagué et al. (2015) for an overview), and the choice to use the default RDKit one was basically an arbitrary one.

selfies The Self-Referencing Embedded Strings (SELFIES) representation of a molecule, which claims to be "a 100% robust molecular string representation" (Krenn et al., 2020), and has been used for various machine learning-based cheminformatics tasks before; see, for example, Nigam et al. (2019), Shen et al. (2021) or Frey et al. (2022).

The links to the respective Python implementations which we used for our work are given in table 3.

We note that the dataset augmented with these features is slightly smaller than then raw dataset, because the computation of the SELFIES representation fails for around a few dozen molecules.

Train / test split: In supervised machine learning, there are typically three kinds of datasets:

1. *Training data* are used to train the model; for example, in the case of a neural network, the updates to the weights are computed from passing the training data through the network and comparing the result with the respective targets.
2. *Validation data* are used to monitor to training progress, and to assess if a model is overfitting: When the error on the training set decreases and the validation error increases, it is usually time to end the training. (Although there exists also the phenomenon of "double descent.")
3. *Test data*, or evaluation data, are used to evaluate the performance of the final model. They are held out until the training procedure is complete.

As part of our data generation routine, we split out full dataset into two files, one for training (and validation), and one for evaluation. This happens only once, to ensure that the test set is the same for all experiments. The split that we use was 90% training and 10% test in the case of the full dataset. For the subset of the data which have mass spectra available (17 021 in total), we used 12 000 for training and the rest for the evaluation. Splitting the training data into data that are actually used for training and data that are used for validation happens at experiment time and is controlled by the random seed of the experiment. Our typical split size is again 90% training and 10% validation.

Machine Learning Experiments

In this subsection, we first take a look at the experimental pipeline that we implemented before we describe the experiments that we have conducted with it. The background and motivation for these experiments have been discussed in the section ▷ Analysis: The Case for Machine Learning.

Experimental Pipeline Design

To make experimentation simple, fast, and reproducible, we have developed a single, modular framework in the form of a Python package, currently called `moml` ("molecular machine learning").[7] Here, an "experiment" is a combination of different (hyper)-parameters—for example, the dataset, the preprocessing steps, and the exact model specification—for which we train a machine learning model and evaluate that model on a previously held-out part of the data to assess its performance. For illustrative purposes, we are giving an (incomplete) overview of the possible "experiment space" (i.e., possible combinations of parameters) in table 4.

Our framework assumes that the models that we train are typically neural networks (linear models can be considered a special case of a neural network), which we implement using PyTorch (Paszke et al., 2019). There exists, however, also some preliminary code that allows to work with models that use the API of the scikit-learn library (Pedregosa et al., 2011; Buitinck et al., 2013).

Training a model is easy: One only has to run a Python script with the desired set of parameters, which can be passed either in the form of command line arguments or as a configuration file using the YAML format. For example, to train an LSTM-type model that takes in the SMILES representation of a molecule and predicts the MA, one would call the training script as follows:

```
$ python train_pytorch.py --model lstm --feature smiles --target ma ...
```

The script automatically documents the parameters that were used to start an experiment, and stores a `git diff`, in case that the repository containing the training code is not in a clean state. This helps to ensure reproducibility of our results. During training, the script regularly outputs metrics such as the training loss and error to a TensorBoard which allows to monitor the progress visually. Common machine learning practices, such as training on a GPU, checkpointing, early stopping (based on the validation loss), or a learning rate scheduler, are already built in by default. Once training has finished, the final model is automatically exported and saved.

Evaluating a trained model (on a given test) is similarly easy and works as follows:

```
$ python evaluate_pytorch.py --experiment-dir /path/to/experiment
```

By default, we run every experiment several times using different random seeds (which control both the initialization of the model's weights and the split of the data into training and validation). The evaluation script finds and runs all these models on the test set and averages their prediction (ensembling) for each data point. This ensemble average is then what we use to evaluate and compare the performance of a given model configuration.

[7] Code repository: https://gitlab.com/frontierdevelopmentlab/2022-us-astroinfosignatures/moml/

Table 4: Schematic overview of the potential space of machine learning experiments (informally dubbed the "ML menu"): Each experiment is given by (1) some choice of data source, (2) a representation for that data, (3) any preprocessing or featurization, (4) a type of machine learning model (plus a model-specific choice of hyper-parameters; not shown here), (5) a target for the prediction (potentially with some preprocessing), (6) and a way of handling prediction uncertainties. Of course, not all combinations make sense; for instance, tokenization and one-hot encoding is only appropriate if the input data comes in the form of a string.

Data source	**Representation**	**Featurization**	**Model type**	**Target**	**Uncertainty**
• Molecules – Pubchem – Rad-6 – GDB-11/13/17 – ... • Mass spectra – NIST – MoNA – Massbank – GNPS – ...	• Molecules – String-based * SMILES * InChI * SELFIES – Molecular graph (e.g., molfile) • Mass spectra: – Series of peaks – Binned histogram	• (None) • (Morgan) fingerprints • Tokenization • One-hot encoding • Scalarization: – Molecular weight – String length • Learned embedding: – mol2vec – MegaMolBART	• Linear regression • Classic ML – Boosted trees – Nearest neighbors – SVM – GP regression – ... • Neural networks – MLP – CNN – RNN / LSTM – Transformer – Graph NN	• Prediction target: – Bertz / BHI – Böttcher – MA – Combination of the above (i.e., multiple regression) • Normalization – *z*-transform – min-max-scaling	• Predict distribution • Monte Carlo dropout • Ensembling – Bagging – Different models

Scaling up MA computation with surrogate models

Method: We are comparing four different kinds of models for the task where we predict the molecular assembly index from different molecular representations, namely the following:

1. *Baselines.* The models in this category are trivial: they only ever return the mean, median, or mode of the training distribution, that is, the "predict" the same complexity value for every molecule. These models have, of course, no real practical use, but they can help us put the results of the more sophisticated models into context, and help us understand the effects of the bias that is due to the very non-uniform distribution of the training data.
2. *Linear models.* Models in this category receive a vector-based representation of a molecule as input: either a mol2vec embedding, an RDKit fingerprint, or an embedding from a pre-trained transformer model (see previous section). As an additional baseline, we also add the Bertz and Böttcher score as predictors to check predictive they are of the MA. After all, we know (e.g., from figure 2) that there are strong correlations between the three complexity metrics.
3. *MLPs.* These multi-layer perceptrons (or fully-connected neural networks) take in the same input as the linear models. We are using models with 3 hidden layers and 1024 units each, with LeakyReLU activations and dropout regularization ($p = 0.2$), which are trained using AdamW with a learning rate of 0.0003, and a min-max normalization to $[0, 1]$ for the inputs.
4. *LSTMs.* The last type of models that we consider here are Long Short-Term Memory (LSTM) neural networks (Hochreiter and Schmidhuber, 1997), which are capable of dealing with variable-length inputs—in our case: one-hot encoded string representations of molecules. We use the default `LSTM` implementation from PyTorch, with a hidden state size of 512. To improve the network capacity, we stack 3 LSTMs, and combine them with an small LSTM that allows us to de-couple the size of the hidden state from the size of the final output.

All models were trained using the experimental pipeline described in the previous section. Since the primary goals of this task was to speed up the computation of MA, during evaluation, we also measured the time it took to process out test set (for two of our best-performing models).

Results: The first part of our results—the distribution of the relative prediction errors on the test set—are shown in figure 3. The general ranking of the models is: 1. Baseline, 2. Linear, 3. LSTM, 4. MLP. This matches our expectation: The baselines are trivial and independent of the input, so even a linear model should outperform them. Within the linear models, the embedding-based models perform better than the models that only receive a single number (i.e., the Bertz or Böttcher score) as their input. Finally, non-linear models are generally expected to perform better than linear ones, and given the limited training data we have, it does not seem unreasonable that the MLP-based models that operate in pre-computed embeddings outperform the LSTMs, which, before they can estimate the MA, first need to learn to extract a useful representation from the string of a molecule.

Perhaps more surpring is the fact that within each group the performance does not seem to depend strongly on the respective input feature. For example, all three MLP-type models basically perform on par, regardless whether the input is an embedding from a pre-trained transformer model, from a (conceptually much simpler) word2vec model, or even from a classical fingerprint algorithm. In all cases, our best-performing models have a relative prediction error of around 4%. For our best-performing model, the predicted MA was correct for 73.6% of the molecules in our test set, and for

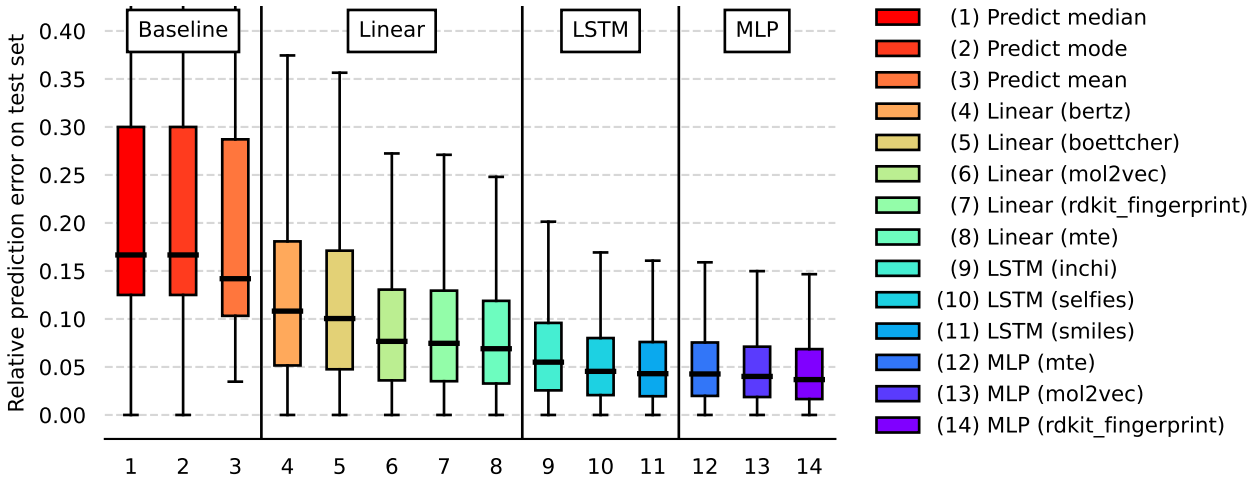


Figure 3: Experiment results for the "MA from molecule representation" task. We show the distributions of the relative predictions errors on the test set (computed as $|t - p|/t$, where t is the true and p the predicted MA given by an ensemble average of five models) for all model types and input features (i.e., molecular representations).

98.1%, the absolute prediction error was $\leq$ 1. Considering how hard it is to compute MA using the "classical" way (i.e., via a branch-and-bound tree search), we find these numbers to be encouraging.

Next, we can look at a comparison of the time it took to compute the MA both using the classical way, and through two of our best-performing models, namely (1) an LSTM operating on a SMILES string, and (2) an MLP that takes in an RDKit fingerprint. We chose these two models for the models for different reasons: (2) gave the overall best performance (i.e., the lowest median relative prediction error on the test set), while (1) can operate "directly" on a common molecular representation and does not need any pre-processing or featurization of the inputs. The final numbers are shown in table 5. The runtime for the classical algorithm, which we ran in parallel distributed over many computing-optimized nodes on Google Cloud (each with 64 cores), was 7.6 million seconds, or around 88 days. By comparison, our fastest model (the fingerprint-based MLP running on a GPU, which was optimized for inference using NVIDIA's TensorRT framework) only took 0.0073 seconds for the same task—a speed-up factor of more than one billion. We believe that we may be able to increase this even further by reducing the numerical precision (e.g., by switching to half precision). However, we but we suspect that this will be of little practical relevance, because at this point, the computational bottleneck is probably already be elsewhere (e.g., the computation of the fingerprints).

Lastly, we can have a look at how our model performs as a function of the (true) MA. The respective plot showing these results in found in figure 4. What we observe is a clear anti-correlation between the amount of training data for a given MA value, and the performance of the model. For example, in the region around MA 7, where we have tens of thousands of molecules in our training set, our median prediction error on the test set is well below 5%. On the other hand, for larger MA value (say, greater than 15) the error increases notably. From a machine learning perspective, this is hardly surprising: Out-of-distribution generalization (i.e., learning a model that also performs well on inputs that differ significantly from what the model has encountered during training) is known to be a very challenging problem, and our specific application is no exception here.

Table 5: Experiment results for the "complexity from molecule representation" task. We are showing the time it took to compute MA values for all molecules in our test set (size: ca. 40k), both with the "classic" algorithm to compute MA, and for two of our best-performing surrogate models, both on CPU and on GPU. The values for our models are computed as averages over multiple runs.

	Classical*	`smiles` + LSTM**	`rdkit_fingerprint` + MLP**
Runtime on CPU (s):	7 597 940.25	456.76	19.21
Runtime on GPU (s):	—	2.60	0.0073***

* The classic algorithm ran on C2-type Google Cloud instances with Intel® Xeon® Gold 6253CL CPUs.
** The evaluation of the ML models ran on a machine with an AMD EPYC 7662 CPU and an NVIDIA A100 GPU.
*** This configuration used NVIDIA's TensorRT framework to optimize the model for high-performance inference.

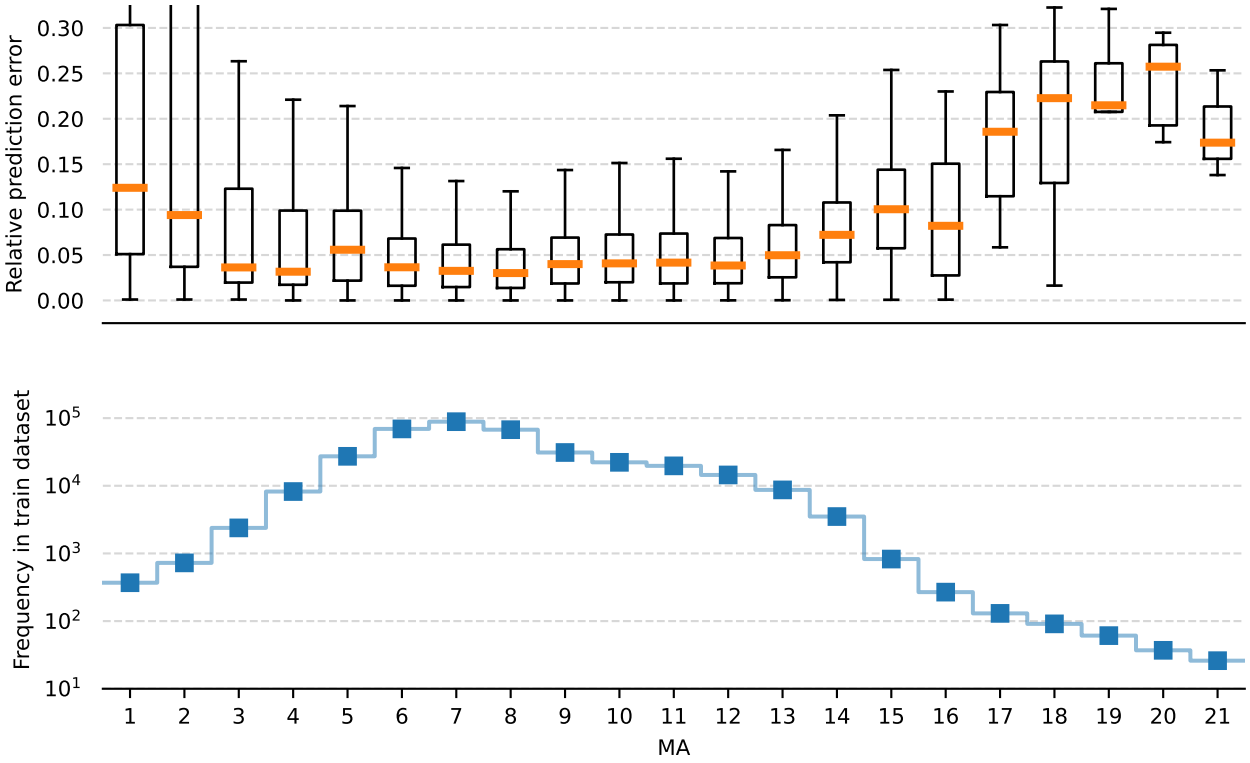


Figure 4: In the top panel, we are plotting the distribution of the relative prediction error on the test set, grouped by different MA values, for our best-performing model (an MLP with an `rdkit_fingerprint` as its input). In the bottom panel, we are showing a histogram of the training data set, that is, the number of molecules in our training set for a given MA value. When taken together, these two panels show that there is a clear anti-correlation between the amount of training data (i.e., number of molecules in training set for a given MA) and the model performance (i.e., relative error on test set for the given MA).

This means that we may be able to use our model to scale up the computation of MA values for molecules in a range for which we already have good training data. However, we need to be extremely careful if we want to use our model also for molecules that are so complex that they are intractable for the classical algorithm. In this case, we are not only asking our model to extrapolate beyond its training distribution—we do not even have access to a ground truth to assess how good or bad our model performs in this regime.

Inferring molecular complexity from mass spectral data

Note: This part of our work was also submitted to and accepted as a short paper at the "Machine Learning and the Physical Sciences" workshop at the NeurIPS 2022 conference.[8]

Method: To infer molecular complexity directly from mass spectrometry data, we compare four different approaches, all of which take a (pre-processed) mass spectrum as their input and output all three complexity measures at once (i.e., multiple regression):

1. *Baseline:* A linear regression (with L_2 regularization) from the number of peaks in the MS to the complexity measure. Marshall et al. (2021) have reported a clear correlation ($\rho = 0.89$) for the case of the MA, hence we consider this our baseline. Pre-processing the MS by removing, for example, all peaks smaller than 5% of the highest peak, did not seem to improve performance.
2. *Linear:* A linear regression (with L_2 regularization) that uses a fixed-length representation of the mass spectra (i.e., a histogram with a 1000 equally spaced bins from 0 Da to 1000 Da).
3. *MLP:* A fully-connected neural network (MLP) that operates on the same binned spectrum as the linear model. All prediction targets were normalized to $[0, 1]$ using a `MinMaxScaler`. The network has 3 `Linear` layers (with 1024 units for the "hidden" layers) and uses LeakyReLU activations, dropout ($p = 0.2$), and batch normalization. Experiments with more layers (or units) did not seem to improve the performance. We found the networks very prone to overfitting, perhaps not least due to the limited training data. Among the various solutions we tried (e.g., increasing dropout), adding random noise to the input spectra during training had the best mitigating effect here.
4. *XGBoost:* Gradient boosted trees as implemented by XGBoost (Chen and Guestrin, 2016), again using the binned mass spectrum. We use default values for all parameters except that we set `n_estimators=1000` and `tree_method="hist"`.

We trained every model five times using different random seeds that control the train / validation split, as well as the model initialization (where applicable). Results are reported as ensemble averages.

Results: Our main results, in the form of relative prediction errors on the evaluation set, are summarized in figure 5. Unsurprisingly, all models outperform the naïve baseline (reducing the error by more than 50% in best case), and non-linear models perform better than the linear one. More interestingly, we find that there is a consistent trend across all models that MA is easier to predict than Böttcher complexity, which in turn is easier to predict than Bertz complexity

[8] Workshop website: https://ml4physicalsciences.github.io

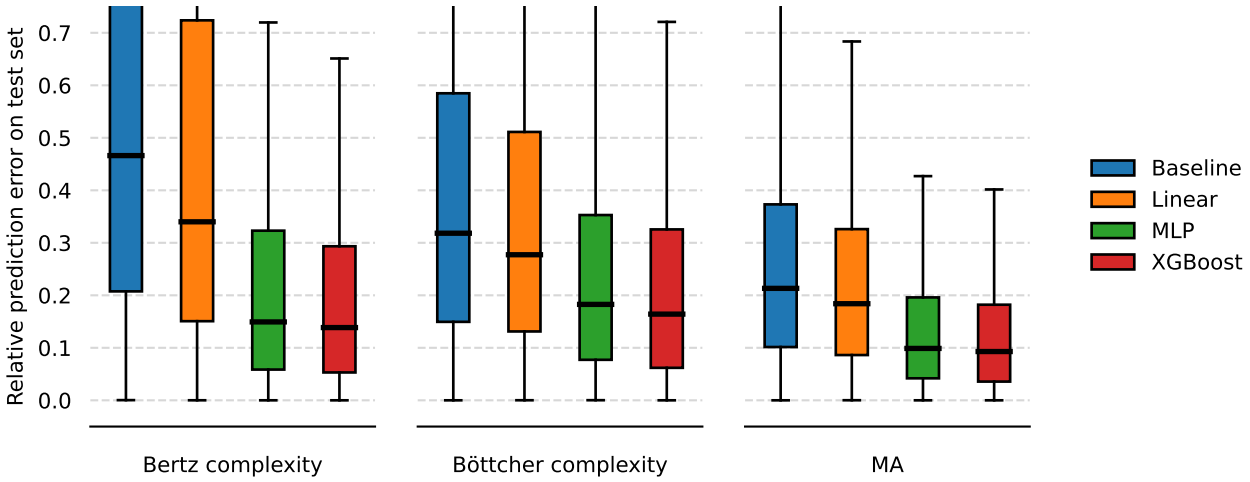


Figure 5: Distribution of relative predictions errors on the test set (computed as $(t - p)/t$, where t is the true and p the predicted complexity value given by an ensemble average of five models corresponding to five different random seeds) for all complexity metrics and model types.

(evidenced by respectively lower predicted errors). We speculate that this may have to do with the definition of the MA, which is, in a way, conceptually similar to the idea of mass spectrometry: The MA counts the number of steps to assemble a molecule from smaller pieces, while MS observers the patterns that emerge when a molecule is fragmented.

Finally, we found that all models are slightly biased towards over-estimating MC. Closer inspection reveals that most of this bias is caused by molecules with (relatively) low MC values, and we hypothesize that the bias may be an effect of the fact that our MC metrics are lower-bounded by 0.

Things that did not work: We also briefly tried the following, methodologically more sophisticated ideas. Both approaches below performed worse than the MLP and XGBoost regressor above:

1. *Encode, Aggregate, Predict:* Every peak of a MS—given by a position-intensity pair (p_i, i_i)—is processed separately by an encoder E that produces a representation $z_i = E(p_i, i_i)$. All z_i of one MS are then aggregated as $z = \text{mean}(z_i)$, and a predictor network P estimates the MC from z.
2. *Transfer learning:* The core idea here was to separate the task of inferring a useful representation of a molecule from the estimation of its complexity, and to allow us to tap into the large dataset for which molecular representations and MC values are available, but no mass spectra. To this end, we look again at one of the models that we trained in the previous section, which consists of an LSTM that takes in the SELFIES representation (Krenn et al., 2020) of a molecule and produces an embedding from which a predictor MLP P then estimates the MC. In our transfer learning approach, we attempt to re-use this pretrained predictor P by training only an encoder network E to take in binned mass spectra and pass the outputs (i.e., abstract representations of the MS) to a frozen version of P to predict the MC.

Of course, these approaches might simply need more training data, or more extensive hyperparameter optimization, to give competitive results. We may therefore revisit them in a future revision.

Future Directions

Dataset improvements: During the FDL 2022 sprint, we gathered more than 1.1 billion molecules with valid structures and string representation using the InChI or SMILES format. The sources of these molecules are PubChem (111M), GDB-13 (977M), GDB-17 (50M random subsample), Rad-6 (10k), Reaxys (26M), NIST (73k) and manually curated prebiotic molecules (343). However, the majority of these molecules are small molecules. For example, the GDB-13 and GDB-17 datasets are computer-generated molecules with a maximum of 13 or 17 carbon atoms, respectively. The potential chemical space of molecules is vast and it is perhaps not meaningful to compute them all. Instead, we chose to randomly sample these datasets into smaller subsets. By design, the target sampling size is 100k molecules from each data source for ML training, another 100k for ML testing and 10k molecules for ML validation.

In reality, we quickly realized that even for these small subsets of molecules (100K) at the mass range from 0–1000 Da, the limitation for completing the dataset generation is MA computation. As outlined in the ▷ Dataset development section of this memo, after 100 000+ CPU hours of computation over two months' time we were only able to generate MA values for about 407k molecules. The distribution of these molecules is not uniform: there is much less really small and really large molecules (cf. figure 2). In addition, we focused on computing for all of the NIST dataset, because a large fraction (about half) of this dataset contain molecules with a mass spectra. We were able to obtain in the end about 17k mass spectra from NIST.

The tasks for dataset generation and refinement is a continuous process that requires continuous support well beyond the short period during the FDL 2022 sprint. The dataset used for ML in this memo represents the best that we could do given the time and resources as a team. Here we note down ideas that can be employed to further improve the dataset. In the meantime, we are working releasing version 1 of the dataset for NASA and for the broader scientific community, together with a publication that will describe it in more detail and showing some applications.

1. *Using a centralized queue system for dataset generation*: Currently, the dataset generation uses a CSV file with a (minimal) list of molecules as its input. This list needs to be prepared ahead of time and is then randomly split into equally-sized chunks; one for each worker node (i.e., Google Cloud instance). The number of chunks are set at the start of the data generation and cannot be changed during the generation process. During processing, each chunk is send to a separate Docker container which processes the assigned molecules and runs until the entire chunk of molecules is completed. We designed the Docker container to save output files in a bucket on Google Cloud Storage. While this system does work, we believe that a better way to implement the data generation would be to use a central queue which stores the list of molecules to process, and from which workers *pull* the next batch of molecules, instead of us *pushing* a fixed set of molecules to each node at the start. Such a system is more robust to failures of individual nodes, ensures better utilization of the available resources, and would allow us to keep adding (or removing) molecules to the queue while the data generation is running. With this setup, it would also make sense for the worker nodes to add their results to a single, shared database (e.g., Google BigQuery or Google BigTable), instead of saving the results as CSV files, which then need to be merged.
2. *Improving MA computation by using better algorithms that target MA upper bound*: MA computation for very large molecules currently can take a very long time. We developed the various ML models in this memo to speed it up to try to solve this problem. Nevertheless, computing

exact MA values for sufficiently large quantities of large molecules is required for ML training and validation. The idea here is to improve MA computation by first producing an upper bound on the MA, and then spending a given amount of computation time on reducing that bound. That way, we also get some result for the MA of a molecule, and no computation time is wasted, unlike for our current approach, where we skip on a molecule that we could not fully compute before reaching the timeout, meaning all the computation is essentially lost. This approach would require working collaboratively with authors of Marshall et al. (2021) to modify the source code (written in Go) accordingly. Under this directive, we can also develop a hybrid approach that combine ML and classic computing to target large molecule MA computation.

3. *More molecules and better labelled molecules in association with source environments and chemical reaction networks*: We would like to extend the dataset by computing complexity scores for more molecules, and especially molecules with better labels that contain source environmental information. Having environmental information would be the key to better characterize the link between molecular complexity and chemical reaction networks. This is because, in addition to characterizing the complexity of molecules, the other class of complexity: the complexity of chemical reactions that produce these molecules in the first place is an exciting area of new research. However, this topic remains a challenge primarily because of the lack of data. Typically, most databases of organic molecules are sourced from and heavily biased towards Earth biochemistry. The organic molecules in these databases can be very large and complex, due to the complex pathways that life evolved in order to make them. Consequently, because our Earth is filled with biological processes, we do not have a handle on purely abiotic chemical reactions and their capacity to generate complex molecules. This is an urgent problem that needs to be solved in astrobiology. We think that the framework outlined in our data generation pipeline can be extended to incorporate more data on abiotic chemical reactions. Doing so would require additional support from NASA on database development and curation.

4. *Characterizing molecular mixtures and their environments*: Real world samples are always going to contain mixtures of molecules. With the development of tandem mass spectrometry, these molecular mixtures can be effectively separated and characterized. Therefore, the datasets generated by tandem mass spectrometry systems are likely groups of molecules rather than single molecules. This kind of dataset presents a new opportunity: to characterize groups of molecules all together and map them in the chemical reaction network space. During FDL 2022, we brainstormed how to approach the analysis of such datasets, for example: (1) Plotting the distribution of complexity metrics in a histogram. This distribution may be useful for fingerprinting the source environment. (2) Accessing the complexity of the reaction network space (assuming all molecules are connected through steps of chemical reactions). This problem could be approached for example by examining the shared assembly of the group (Liu et al., 2021). We would need more datasets formulated in this fashion (molecular groups) in order to approach how best to characterize them. These questions would need to be resolved before we can employ meaningful techniques (including new ML algorithms) to access the probability that the molecular mixtures are produced by life or not. Currently, datasets formatted in this fashion are not yet available, but we envision that we could extend the data generation pipeline and design new data attributes to tag molecular groups.

5. *Integrating different types of tandem mass spectrometry data*: Currently we use mass spectrometry data from NIST, which is based on a standard mass spectrometer using electron ionization. However, there are many more configurations of mass spectrometry data that operate in tandem

during the measurement, for example GC-MS, LC-MS or MS-MS. In addition, there are varied ways to produce the ion used in mass spectrometry separation (positive vs. negative ion mode, energy levels, etc.). We did not consider these instrumentation variations during the FDL 2022 sprint. These mass spectra data features should be more carefully evaluated, and likely we would need additional data columns or metadata to properly sort out these variations.

6. *Integrate other types of spectrometry data (optical spectrometry), or other classes of datasets*: It is also possible to incorporate other types of spectrometry data that contain molecular information, such as optical spectrometry using Raman and FTIR. Appending these additional dataset into our data is exciting, which will also offer more ML experiments down the road.

CONCLUSION

In conclusion, during FDL 2022, as a team we worked towards reaching goals set at three different tiers for the sprint: a **Minimal Goal**, a **Target Goal**, and a **Bold & Crazy Goal**.

For the **Minimal Goal**, we carefully curated and generated a dataset of more than 400K molecules for which we computed all three complexity metrics: MA, Bertz and Böttcher scores. This dataset includes 17k unique molecules with mass spectra.

For the **Target Goal**, we conducted various ML experiments aimed at two main tasks: (1) to speed up the computation of MA and (2) to infer molecular complexity directly from mass spectrometry data.

To achieve these goal, we developed two Python packages: `complexipy` and `moml`. `complexipy` is focused on data processing and computation for molecular complexity metrics. `moml` is focused on producing an easy-to-deploy, streamlined ML experimentation platform and for training, characterizing and benchmarking ML models. Our results are promising and we show that not only ML methods are effective at speeding up MA computations using surrogate ML models (speed improvements up to a billion fold faster depending on the hardware used), but also they can be used to predict molecular complexity directly from experimental datasets such as the mass spectra. Compared with the baseline that count the number of peaks in mass spectra (Marshall et al., 2021), linear ML models can reduce prediction error by more than 50%. Non-linear ML models performed better than the linear models. These results demonstrate that ML are effective and practical tools for characterizing molecular complexity from either string representations of molecules as well as raw output data directly from instruments on-board remote robotic spacecrafts. These ML methods can indeed be further integrated in future space missions with enormous potential benefits.

For the **Bold & Crazy Goal**, we characterized the potential benefits of using ML models quantitatively and comparatively. These characterizations will help the future design of such autonomous systems. Some experiments were done, for example, looking at how tolerant are our ML models at ingesting mass spectra at various noise levels. We show that our ML models are effective in this regard, tolerating noise during training without significantly affecting the prediction errors. In addition, we examined ML pruning, which simulates the reduction of hardware capacity (the available artificial neurons) while not sacrificing prediction error. These benchmarks are useful for examining realistic hardware limitations and are the first steps necessary towards ML-deployments on real mission hardware. We are eager to share these results and work with teams at NASA and its collaborators, wherever possible, to further refine our models and find opportunities to deploy them on forthcoming missions to other planetary bodies.

REFERENCES

Artrith, N. et al. (2021). "Best practices in machine learning for chemistry." *Nature Chemistry*, 13(6): 505–508. DOI: 10.1038/s41557-021-00716-z.

Bartlett, S. and M. L. Wong (2020). "Defining Lyfe in the Universe: From Three Privileged Functions to Four Pillars." *Life*, 10(4): 42. DOI: 10.3390/life10040042.

Baydin, A. G., L. Shao, W. Bhimji, et al. (2019). "Efficient Probabilistic Inference in the Quest for Physics Beyond the Standard Model." In: *Advances in Neural Information Processing Systems*. Volume 32. Curran Associates, Inc.

Bertz, S. H. (1981). "The first general index of molecular complexity." *Journal of the American Chemical Society*, 103(12): 3599–3601. DOI: 10.1021/ja00402a071.

Bonchev, D. and O. Polansky (1987). "On the Topological Complexity of Chemical Systems." In: *Graph theory and topology in chemistry*. Amsterdam, Oxford, New York, Tokyo: Elsevier, pages 126–158. ISBN: 0-444-42882-8.

Boskovic Group (2020). *bottchercomplexity*. URL: https://github.com/boskovicgroup/bottchercomplexity.

Böttcher, T. (2016). "An Additive Definition of Molecular Complexity." *Journal of Chemical Information and Modeling*, 56(3): 462–470. DOI: 10.1021/acs.jcim.5b00723.

Böttcher, T. (2017). "From Molecules to Life: Quantifying the Complexity of Chemical and Biological Systems in the Universe." *Journal of Molecular Evolution*, 86(1): 1–10. DOI: 10.1007/s00239-017-9824-6.

Buitinck, L. et al. (2013). "API design for machine learning software: experiences from the scikit-learn project." In: *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122.

Cabrol, N. A. (2016). "Alien Mindscapes — A Perspective on the Search for Extraterrestrial Intelligence." *Astrobiology*, 16(9): 661–676. DOI: 10.1089/ast.2016.1536.

CAS (1978). "CAS Registry System." *Journal of Chemical Information and Computer Sciences*, 18(1): 58–58. DOI: 10.1021/ci60013a609.

Cereto-Massagué, A. et al. (2015). "Molecular fingerprint similarity search in virtual screening." *Methods*, 71: 58–63. DOI: 10.1016/j.ymeth.2014.08.005.

Chen, T. and C. Guestrin (2016). "XGBoost." In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Acm. DOI: 10.1145/2939672.2939785.

Cranmer, K., J. Brehmer, and G. Louppe (2020). "The frontier of simulation-based inference." *Proceedings of the National Academy of Sciences*, 117(48): 30055–30062. DOI: 10.1073/pnas.1912789117.

Cronin, L. and S. I. Walker (2016). "Beyond prebiotic chemistry." *Science*, 352(6290): 1174–1175. DOI: 10.1126/science.aaf6310.

Des Marais, D. J. and M. R. Walter (1999). “Astrobiology: Exploring the Origins, Evolution, and Distribution of Life in the Universe.” *Annual Review of Ecology and Systematics*, 30(1): 397–420. DOI: 10.1146/annurev.ecolsys.30.1.397.

Des Marais, D. J. et al. (2008). “The NASA Astrobiology Roadmap.” *Astrobiology*, 8(4): 715–730. DOI: 10.1089/ast.2008.0819.

Fawzi, A. et al. (2022). “Discovering faster matrix multiplication algorithms with reinforcement learning.” *Nature*, 610(7930): 47–53. DOI: 10.1038/s41586-022-05172-4.

Frey, N. C., V. Gadepally, and B. Ramsundar (2022). *FastFlows: Flow-Based Models for Molecular Graph Generation*. arXiv:2201.12419.

Goodfellow, I., Y. Bengio, and A. Courville (2016). *Deep Learning*. MIT Press.

Grubisic, A. et al. (2021). “Laser Desorption Mass Spectrometry at Saturn’s moon Titan.” *International Journal of Mass Spectrometry*, 470: 116707. DOI: 10.1016/j.ijms.2021.116707.

Hand, K. P. et al. (2022). “Science Goals and Mission Architecture of the Europa Lander Mission Concept.” *The Planetary Science Journal*, 3(1): 22. DOI: 10.3847/psj/ac4493.

Heller, S. et al. (2013). “InChI - the worldwide chemical structure identifier standard.” *Journal of Cheminformatics*, 5(1). DOI: 10.1186/1758-2946-5-7.

Heller, S. R. et al. (2015). “InChI, the IUPAC International Chemical Identifier.” *Journal of Cheminformatics*, 7(1). DOI: 10.1186/s13321-015-0068-4.

Himes, M. D., J. Harrington, A. D. Cobb, et al. (2022). “Accurate Machine-learning Atmospheric Retrieval via a Neural-network Surrogate Model for Radiative Transfer.” *The Planetary Science Journal*, 3(4): 91. DOI: 10.3847/psj/abe3fd.

Hochreiter, S. and J. Schmidhuber (1997). “Long Short-Term Memory.” *Neural Computation*, 9(8): 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

Jaeger, S., S. Fulle, and S. Turk (2018). “Mol2vec: Unsupervised Machine Learning Approach with Chemical Intuition.” *Journal of Chemical Information and Modeling*, 58(1): 27–35. DOI: 10.1021/acs.jcim.7b00616.

Kim, S. et al. (2021). “PubChem in 2021: New Data Content and Improved Web Interfaces.” *Nucleic Acids Research*, 49(D1): D1388–d1395. ISSN: 0305-1048. DOI: 10.1093/nar/gkaa971. (Visited on Oct. 12, 2022).

Kipf, T. N. and M. Welling (2016). *Semi-Supervised Classification with Graph Convolutional Networks*. arXiv:1609.02907.

Krenn, M., F. Häse, A. K. Nigam, et al. (2020). “Self-referencing embedded strings (SELFIES): A 100% robust molecular string representation.” *Machine Learning: Science and Technology*, 1(4): 045024. DOI: 10.1088/2632-2153/aba947.

Li, Z., Q. Chen, and V. Koltun (2018). *Combinatorial Optimization with Graph Convolutional Networks and Guided Tree Search*. arXiv:1810.10659.

Lim, J. et al. (2018). “Molecular generative model based on conditional variational autoencoder for de novo molecular design.” *Journal of Cheminformatics*, 10(1). DOI: 10.1186/s13321-018-0286-7.

Liu, Y., C. Mathis, M. D. Bajczyk, et al. (2021). “Exploring and mapping chemical space with molecular assembly trees.” *Science Advances*, 7(39). DOI: 10.1126/sciadv.abj2465.

Lorenz, R. et al. (2018). “Dragonfly: A rotorcraft lander concept for scientific exploration at Titan.” *Johns Hopkins APL Technical Digest (Applied Physics Laboratory)*, 34(3): 374–387. ISSN: 0270-5214.

MacKenzie, S. M. et al. (2021). “The Enceladus Orbilander Mission Concept: Balancing Return and Resources in the Search for Life.” *The Planetary Science Journal*, 2(2): 77. DOI: 10.3847/psj/abe4da.

Marshall, S. M., C. Mathis, E. Carrick, et al. (2021). “Identifying molecules as biosignatures with assembly theory and mass spectrometry.” *Nature Communications*, 12(1). DOI: 10.1038/s41467-021-23258-x.

Marshall, S. M., A. R. G. Murray, and L. Cronin (2017). “A probabilistic framework for identifying biosignatures using Pathway Complexity.” *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 375(2109): 20160342. DOI: 10.1098/rsta.2016.0342.

Mateo-Garcia, G., J. Veitch-Michaelis, L. Smith, et al. (2021). 11(1). DOI: 10.1038/s41598-021-86650-z.

Mikolov, T. et al. (2013). *Efficient Estimation of Word Representations in Vector Space*. arXiv:1301.3781.

Milan, A. et al. (2017). “Data-Driven Approximations to NP-Hard Problems.” *Proceedings of the AAAI Conference on Artificial Intelligence*, 31(1). DOI: 10.1609/aaai.v31i1.10750.

Morris, P. et al. (2020). “Predicting Binding from Screening Assays with Transformer Network Embeddings.” *Journal of Chemical Information and Modeling*. ISSN: 1549-9596. DOI: 10.1021/acs.jcim.9b01212.

NASA (2020). *Big questions*. URL: https://science.nasa.gov/astrophysics/big-questions.

National Academies of Sciences, Engineering, and Medicine (2019). *An Astrobiology Strategy for the Search for Life in the Universe*. Washington, DC: National Academies Press. DOI: 10.17226/25252.

National Academies of Sciences, Engineering, and Medicine (2022). *Origins, Worlds, and Life: A Decadal Strategy for Planetary Science and Astrobiology 2023–2032*. Washington, DC: National Academies Press. ISBN: 978-0-309-47578-5. DOI: 10.17226/26522.

Nigam, A. K. et al. (2019). “Augmenting Genetic Algorithms with Deep Neural Networks for Exploring the Chemical Space.” arXiv:1909.11655.

NIST (2022). *NIST Chemistry WebBook, NIST Standard Reference Database 69*. DOI: 10.18434/t4d303. URL: https://webbook.nist.gov/chemistry/.

Oba, Y., Y. Takano, H. Naraoka, et al. (2020). “Extraterrestrial hexamethylenetetramine in meteorites — precursor of prebiotic chemistry in the inner solar system.” *Nature Communications*, 11(1). DOI: 10.1038/s41467-020-20038-x.

Parascandolo, G. et al. (2020). *Divide-and-Conquer Monte Carlo Tree Search For Goal-Directed Planning*. arXiv:2004.11410.

Paszke, A. et al. (2019). “PyTorch: An Imperative Style, High-Performance Deep Learning Library.” In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., pages 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Pedregosa, F. et al. (2011). “Scikit-learn: Machine Learning in Python.” *Journal of Machine Learning Research*, 12: 2825–2830.

Poduval, B., A. G. Baydin, and N. Schwadron (2021). “Studying Solar Energetic Particles and Their Seed Population Using Surrogate Models.” In: *Machine Learning for Space Sciences workshop, 43rd Committee on Space Research (COSPAR) Scientific Assembly, Sydney, Australia*.

Randić, M. et al. (2005). “On the Complexity of Fullerenes and Nanotubes.” In: *Complexity in Chemistry, Biology, and Ecology*. Boston, MA: Springer US, pages 1–48.

Ruddigkeit, L. et al. (2012). “Enumeration of 166 Billion Organic Small Molecules in the Chemical Universe Database GDB-17.” *Journal of Chemical Information and Modeling*, 52(11): 2864–2875. DOI: 10.1021/ci300415d.

Shen, C. et al. (2021). “Deep molecular dreaming: inverse machine learning for de-novo molecular design and interpretability with surjective representations.” *Machine Learning: Science and Technology*, 2(3): 03lt02. DOI: 10.1088/2632-2153/ac09d6.

Shirobokov, S., V. Belavin, M. Kagan, et al. (2020). “Black-box optimization with local generative surrogates.” *Advances in Neural Information Processing Systems*, 33: 14650–14662.

Silver, D. et al. (2018). “A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play.” *Science*, 362(6419): 1140–1144. DOI: 10.1126/science.aar6404.

Stocker, S. et al. (2020). “Machine learning in chemical reaction space.” *Nature Communications*, 11(1). DOI: 10.1038/s41467-020-19267-x.

The RDKit Team (G. Landrum et al.) (no date). *RDKit: Open-source cheminformatics*. URL: https://www.rdkit.org.

Theiling, B. P. et al. (2022). “Science Autonomy for Ocean Worlds Astrobiology: A Perspective.” *Astrobiology*, 22(8): 901–913. DOI: 10.1089/ast.2021.0062.

Trifonov, E. N. (2011). “Vocabulary of Definitions of Life Suggests a Definition.” *Journal of Biomolecular Structure and Dynamics*, 29(2): 259–266. DOI: 10.1080/073911011010524992.

Uthamacumaran, A. et al. (2022). *On the Salient Limitations of the Methods of Assembly Theory and their Classification of Molecular Biosignatures*. arXiv:2210.00901.

Walker, S. I., N. Packard, and G. D. Cody (2017). “Re-conceptualizing the origins of life.” *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 375(2109): 20160337. DOI: 10.1098/rsta.2016.0337.

Wang, M. et al. (2016). “Sharing and community curation of mass spectrometry data with Global Natural Products Social Molecular Networking.” *Nature Biotechnology*, 34(8): 828–837. DOI: 10.1038/nbt.3597.

Wang, X. et al. (2020). “Towards efficient discovery of green synthetic pathways with Monte Carlo tree search and reinforcement learning.” *Chemical Science*, 11(40): 10959–10972. DOI: 10.1039/d0sc04184j.

Wei, J. N. et al. (2019). “Rapid Prediction of Electron-Ionization Mass Spectrometry Using Neural Networks.” *ACS Central Science*, 5(4): 700–708. DOI: 10.1021/acscentsci.9b00085.

Weininger, D. (1988). “SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules.” *Journal of Chemical Information and Modeling*, 28(1): 31–36. DOI: 10.1021/ci00057a005.

Zhang, B. et al. (2022). “Prediction of electron ionization mass spectra based on graph convolutional networks.” *International Journal of Mass Spectrometry*, 475: 116817. DOI: 10.1016/j.ijms.2022.116817.