Leveraging Language Models to Efficiently Learn Symbolic Optimization Solutions

Felipe Leno da Silva, Andre Goncalves, Sam Nguyen, Denis Vashchenko, Ruben Glatt, Thomas Desautels, Mikel Landajuela, Brenden Petersen, Daniel Faissol
Lawrence Livermore National Lab
Livermore, CA, USA
{leno,andre,nguyen116,vashchenko1,glatt1,desautels2,landajuelala1,bp,faissol1}@llnl.gov

ABSTRACT

Symbolic Optimization has been used to solve varied challenging and relevant problems such as Symbolic Regression and Neural Architecture Search. However, the current state-of-the-art typically learns each problem *from scratch*, and is unable to leverage pre-existing knowledge and datasets that are available for many applications. Inspired by the similarity between sequence representations learned in Natural Language Processing and the formulation of symbolic optimization as a discrete sequence optimization problem, we propose *Language Model-Accelerated Deep Symbolic Optimization* (LA-DSO), a method that leverages language models to learn symbolic optimization solutions more efficiently. We demonstrate LA-DSO in two tasks: symbolic regression, which allows us to compare against the state-of-the-art approaches, and computational antibody optimization, which shows that our proposal accelerates learning for challenging real-world problems.

KEYWORDS

Transfer Learning, Reinforcement Learning, Discrete Optimization, BERT.

Solutions for many relevant real-world problems can be found by modeling them as Symbolic Optimization problems, where a sequence of tokens that optimizes a black-box reward/fitness function has to be found. Symbolic Regression [17], Neural Architecture Search [39], and Program Synthesis [14] are popular problems that are naturally modeled in this way.

Algorithms for Symbolic Optimization must deal with exploring the huge, combinatorial search space of token sequences. Ideally, such approaches may prune the search space and/or bias the search toward more promising token sequences; however, this might be challenging due to the black-box nature of the fitness function. The Deep Symbolic Optimization (DSO) [26] framework efficiently solves this class of problems by modeling the token sampling process as a Reinforcement Learning (RL) problem. DSO achieves state-of-the-art results in symbolic regression, in part due to its ability to effectively prune and bias the search through *in situ* priors and constraints. However, this process requires a human to manually encode this knowledge, which might be difficult to define and tune. Therefore, we focus on a more automated approach instead of leveraging priors.

On the other hand, transfer learning is paramount to many recent advances in Natural Language Processing (NLP). Inspired by similarities between sequence representations learned in NLP and the formulation of symbolic optimization as a discrete sequence optimization problem, we combine ideas from both fields to integrate transfer learning into symbolic optimization algorithms.

We propose *Language Model-Accelerated Deep Symbolic Optimization* (LA-DSO), a symbolic optimization algorithm that leverages language models trained on real-world token sequences to learn faster. Our contributions in this paper are three-fold: (i) We propose LA-DSO as a way to leverage language models in the learning process to solve symbolic optimization problems more efficiently; (ii) We implement LA-DSO in two challenging real-world domains, symbolic regression and antibody optimization; and (iii) We empirically show significant benefits when using LA-DSO in those domains, compared to when using the basic DSO algorithm without language models.

This paper is organized as follows. In Section 1, we present relevant background for our approach. In Section 2, we describe LA-DSO and discuss its implementation details. In Section 3, we present our empirical setting and results. In Section 4, we describe the main related works and position our proposal within state-of-the-art methodologies. Finally, in Section 5, we summarize our main findings.

1 BACKGROUND

We provide the relevant background knowledge on symbolic optimization and language models to fully understand our proposal.

1.1 Symbolic Optimization

The symbolic optimization problem consists of finding the optimal combination of discrete symbols relative to a quality score function, that is, it is a (constrained) combinatorial optimization problem. Let $\mathcal{L} = \{\tau^1, \ldots, \tau^t\}$ be the *library*, i.e., a set of individual *tokens* τ^i which define the space of possible *token sequences* $\tau^\star = \langle \tau_1, \ldots, \tau_n \rangle$ that can be built for solving the problem at hand. In principle, the sequence can be of any length. The reward function $R : \tau \rightarrow \mathbb{R}$ defines the fitness of each sequence. Any *valid* sequence τ can have its reward value queried by $R(\tau)$. The solution of a symbolic optimization problem is given by:

$$\arg max_{n \in \mathbb{N}, \tau} \left[R(\tau) \right] \text{ with } \tau = \langle \tau_1, \ldots, \tau_n \rangle, \tau_i \in \mathcal{L}. \quad (1)$$

That is, the problem is solved by finding the sequence that maximizes the reward function. The main challenge of this problem is searching the set of possible solutions, which defines an extremely large combinatorial search space. *Learning* to solve this problem by training a machine learning model to sample high-reward token sequences has been a very successful approach that has set new

Proc. of the Adaptive and Learning Agents Workshop (ALA 2022), Cruz, Hayes, Silva, Santos (eds.), May 9-10, 2022, Online, https://ala2022.github.io/. 2022.

state-of-the-art results for relevant applications [26]. We describe our base learning algorithm in the next subsection.

1.2 Deep Symbolic Optimization

Deep Symbolic Optimization (DSO) [26] is a framework for solving Symbolic Optimization problems. By modeling the token sampling process as a Reinforcement Learning problem, each token is sampled sequentially until the whole sequence is formed.

DSO is a policy gradient-based RL method that employs a Recurrent Neural Network (RNN) to sample token sequences. The RNN provides a parameterized distribution over sequences, $p(\tau|\theta)$, with network parameters θ. Sampling a sequence occurs in an autoregressive fashion, in which tokens are sampled sequentially and are independent, given the tokens sampled so far: $p(\tau_i|\tau_{1:(i-1)};\theta) = \text{softmax}(\psi^{(i)})_{\mathcal{L}(\tau_i)}$, where $\psi^{(i)}$ are the outputs of the RNN for the ith sequence position. A batch of promising token sequences $\mathcal{T}$ is generated and their rewards are used to train the RNN. The Risk-Seeking Policy Gradient [26] is a common way of training this RNN:

$$\mathcal{L}(\theta) = \frac{1}{\varepsilon|\mathcal{T}|} \sum_{\tau \in \mathcal{T}} (R(\tau) - \tilde{R}_\varepsilon) \nabla_\theta \log p(\tau|\theta) \mathbf{1}_{R(\tau) > \tilde{R}_\varepsilon} \qquad (2)$$

where ε determines the degree of risk-seeking and $\tilde{R}_\epsilon$ is the empirical $(1-\varepsilon)$ reward quantile of $\mathcal{T}$. This training objective aims to optimize the search for *best* tokens sequences, rather than trying to optimize the estimation of average qualities.

DSO was shown to be very effective for symbolic optimization problems due to easily allowing the integration of expert domain knowledge into the learning through *in situ* priors and constraints [11, 15, 22, 26, 27].

However, a key limitation of DSO is that it still has to learn the interactions among tokens from scratch for each new problem. This results in a relatively high starting cost to learn how each token interacts with others in the sequence and how this affects the reward landscape.

1.3 Language Models

NLP requires understanding the intricate relationship between the words across sentences and their possible multiple, context-dependent, meanings. The huge amount of textual data currently available enabled training deep learning models, such as BERT [9], that can learn complex representations of human language.

Language modeling is often framed as unsupervised distribution approximation from a set of sentences $(S_1, S_2, S_3, ..., S_N)$, where each S_i is comprised of a list of words (tokens) $s_1, s_2, s_3, ..., s_L$. The task of language modeling is then to learn the conditional probability distribution $p(s_i|C_i)$, where C_i is the context for predicting s_i. In the case of an autoregressive language model, $C_i = \langle s_1, ..., s_{i-1} \rangle$ while in the case of a bidirectional language model, such as BERT [9], $C_i = \langle s_1, ..., s_{i-1}, s_{i+1}, ..., s_L \rangle$. After training, such language models can provide meaningful token representations for downstream tasks [9, 29], often by mapping token sequences to a higher-dimension embedding vector [28], where semantically similar words are clustered together.

Since training such models requires a large corpus of data and intensive computing power, transfer learning [35], in which embeddings or portions of the models are repurposed for other tasks, is common for NLP applications. By fine-tuning and/or replacing layers on a language model network architecture, it is often possible to solve different but related tasks, while leveraging the acquired knowledge of how the human language is structured.

2 INTEGRATING LANGUAGE MODELS INTO DSO

DSO handles discrete tokens, sampled sequentially and concatenated into sequences. Likewise, NLP tasks require handling sentences of words converted into discrete tokens. Therefore, the sentences on NLP tasks can be seen as equivalent to sequences of tokens for DSO (although not all NLP tasks are solved by sampling the "best" sequence of tokens, unlike symbolic optimization problems).

Due to the difficulty of learning the complexities of natural language, NLP tasks are typically not trained from scratch unless abundant data and computational resources are available. In practice, transfer learning is often applied in the form of reusing embeddings or language models (either the entire model or a subset of its layers). Yet, despite the aforementioned similarities, DSO always starts the learning process from scratch.

Figure 1 describes our proposal in high level. Given that language models are already able to handle discrete tokens and to learn the relation between them in well-formed sequences, we assume the domain of interest allows training a language model. The intuition behind this approach is to use the language model to learn the "language" of the symbolic optimization task to be learned. For example, in a neural architecture search task, we could train a language model in a dataset of neural network architectures that have been used to solve real problems. The language model would then learn what a useful architecture "looks like" and how different architectural components are commonly together (e.g., pooling layers often follow CNN layers). Other domains that could leverage language model training are Symbolic Regression and Antibody Optimization, which are further detailed in our experimental evaluation (Section 3).

The knowledge encoded in the language model will be useful for learning how to solve new tasks. The language model can provide contextual information which would otherwise be unavailable to DSO, which we argue can help speeding up the learning process. Although this idea is applied to DSO in this particular paper, this approach is more general and could potentially be applied in any RL domain in which it would make sense to train a language model.

Algorithm 1 presents an algorithmic view of our proposal, hereinafter called *Language Model-Accelerated Deep Symbolic Optimization* (LA-DSO). We consider the language model training as a pre-training process, that has to be executed before DSO is used. Therefore, the algorithm starts by training the language model using a database of token sequences $\mathcal{T}$. The purpose of this database is to "teach" the language model how good token sequences look like. This could contain, for example, sequences extract from public repositories (such as our experiment in Section 3.1).

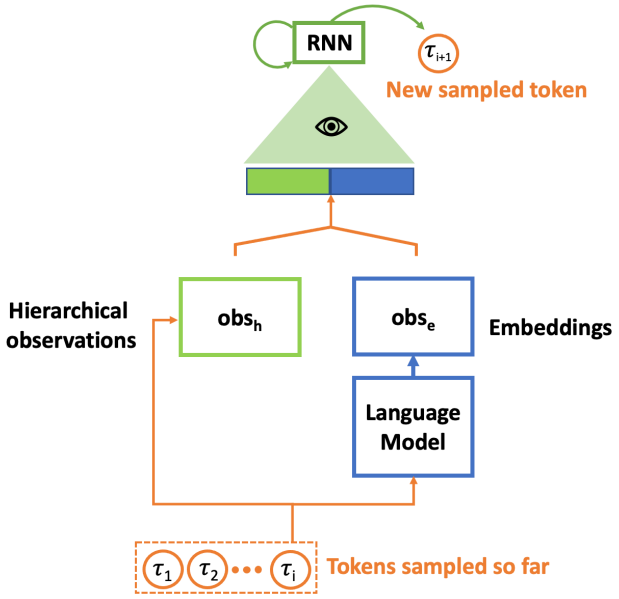

Figure 1: High-level illustration of LA-DSO. We propose to leverage the representation learned by language models to learn faster. At every time the RNN is used to sample a new token, the current partial sequence is forwarded by both DSO's standard featurization process and the language model. The defined features and language model embeddings are then concatenated, before they are used as observation in DSO's RNN.

Algorithm 1 Language Model-Accelerated Deep Symbolic Optimization

Input: $\mathcal{T}$: database of real-world token sequences; lm: language model; Γ: DSO RNN; R: reward function; n_e: number of training iterations; n_l maximum sequence length.
if lm untrained **then**
  lm.train($\mathcal{T}$)
end if
$best \leftarrow \emptyset$
for $e \in \{1, \ldots, n_e\}$ **do**
  $\boldsymbol{\tau} \leftarrow \emptyset$
  for $i \in \{1, \ldots, n_l\}$ **do**
    $obs_s \leftarrow hierarchical(\boldsymbol{\tau})$
    $obs_l \leftarrow lm.get\_embedding(\boldsymbol{\tau}_i | \boldsymbol{\tau}_{1:i-1})$
    $obs \leftarrow concat(obs_s, obs_l)$
    $\boldsymbol{\tau} \leftarrow concat(\boldsymbol{\tau}, \Gamma.sample(obs))$
  end for
  if $R(\boldsymbol{\tau}) > R(best)$ **then**
    $best \leftarrow \boldsymbol{\tau}$
  end if
  $\Gamma.train(\boldsymbol{\tau}, R)$
end for
Return $best$

After the model has been trained, we enter the main DSO learning loop. DSO samples each token sequentially, taking into account information about the partial sequence sampled so far. Normally, DSO would build a token tree, and the RNN would use only hierarchical information (parent and sibling of the token to be sampled) about the partial sequence to take actions on which next token to sample (we call the process of defining those hierarchical features as *hierarchical* in the algorithm). LA-DSO, however, will also add the language model information to the RNN observation space. This is performed by concatenating the usual observations (obs_s) to the embeddings (obs_l) from the language model (normally the last hidden layer). Any neural network-based language model can be used, to allow flexibility. Based on the concatenated observation obs, the RNN can then sample the next token ($\Gamma.sample(obs)$).

This process is executed iteratively (in batches of sequences) until a maximum sequence length is achieved. This batch of sampled sequences can then be used to train the RNN, which will autonomously bias the search towards the most promising sequences. The knowledge from the language model helps the RNN to bias the search more quickly, by leveraging the representations built for the sequences of tokens.

We perform an empirical evaluation in two different domains of interest in the next section, where we show the advantages of our proposal.

3 EXPERIMENTAL EVALUATION

In this section we present our empirical evaluation in two challenging and relevant domains: *Symbolic Regression* and *Antibody Optimization*.

3.1 Symbolic Regression Leveraging Wikipedia

Symbolic regression aims to identify mathematical expressions that best fit a set of observations. This can be used, for example, to discover equations that explain physical phenomena, by searching over the space of tractable (i.e. concise, closed-form) expressions. Specifically, given a dataset $(\boldsymbol{X}, \boldsymbol{y})$, where each observation $\boldsymbol{X}_i \in \mathbb{R}^n$ is related to a target value $y_i \in \mathbb{R}$, symbolic regression aims to identify a function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ that best fits the dataset, where the functional form of f is a finite mathematical expression. The resulting expression can be readily interpreted and/or provide useful scientific insights simply by inspection. The reward function simply involves evaluation of the expression against the $(\boldsymbol{X}, \boldsymbol{y})$ dataset.

Symbolic regression exhibits several unique features that make it an ideal testbed problem for benchmarking symbolic optimization:

(1) Well-established and challenging benchmarks are available [36];
(2) The success criteria is clearly defined (exact symbolic correctness);
(3) Well-established baseline methods are available; and
(4) Computing the quality of candidate expressions is computationally inexpensive, allowing repeating experiments until statistical significance is achieved.

The space of mathematical expressions is discrete (in model structure) and continuous (in model parameters - when continuous constants are allowed), growing exponentially with the length of

the expression, rendering symbolic regression a very challenging machine learning problem [17].

Humans have handcrafted equations describing varied physical phenomena. For symbolic regression, our language model is trained using this body of knowledge, to learn what human-generated equations typically "look like." As a repository for human-defined equations, we use *Wikipedia* [1]. Wikipedia provides raw text data from its pages as XML (Extensible Markup Language) "dump" files.[2] We parse these files and extract any embedded mathematical expressions, which are annotated via `<math>` tags. Those expressions are represented by LaTeX text, which we then convert into a tree-based representation—called an algebraic expression tree—via the computer algebra system SymPy [19]. Finally, we encode expressions as sequences of tokens corresponding to the pre-order traversal of the algebraic expression tree. This sequence data becomes then directly compatible with the way DSO represents sequences, and is ready for training the language model. Our language model for this domain is an RNN trained to predict the next token in a partial sequence by minimizing the cross-entropy loss between the output of the RNN and the next token. The model has 1 hidden layer with 32 output embeddings. This method of training a simple language model is common in NLP tasks [21].

For empirical evaluation, we use the Nguyen symbolic regression benchmark suite [33]. Nguyen is a set of commonly used benchmark expressions developed and vetted by the symbolic regression community [36]. We evaluate two approaches on twelve Nguyen benchmarks:

- **Baseline**: Regular DSO without use of language models; and
- **Wikipedia**: LA-DSO using our model trained with Wikipedia data.

Both approaches are evaluated with the same hyperparameters. This experiment aims at revealing whether the language model is helpful to the learning process.

3.1.1 Experimental Results. Table 1 depicts the experimental results for all evaluated Nguyen benchmarks. In symbolic regression, finding the best expressions is the priority. When ties in expression quality happen, the next priority is to minimize training time. Using the Wikipedia language model significantly improved the learning results in multiple benchmarks. For Nguyen 5 and 11, LA-DSO found higher-quality expressions than vanilla DSO. For Nguyen 1, 3, 6, 8, 9, and 12, LA-DSO achieved reductions from 5% to 41% in the number of iterations to learn the task, a very significant result considering DSO already was the state-of-the-art in the task.

LA-DSO did show difficulty in solving some of the benchmarks. The reason why LA-DSO resulted in lower quality sequences in Nguyen 7 is understandable by inspecting the ground-truth expression. This benchmark is very unlike the real-world expressions used to train the language model. For evaluation purposes, underperforming in this particular benchmark is not a problem, since in real-world situations the algorithm would be primarily looking for equations compatible with the dataset. Similar reasons hold for Nguyen 2 and 10, where the expressions were different from the ones in the dataset and the increase of network size needed for adding the language model observations outweighed the potential speedup that could be achieved for this particular task.

Overall, LA-DSO was better in 8 of 12 benchmarks, with one tie. This corresponds to a significant improvement upon the algorithm that already was the state-of-the-art in this domain.

Finally, Figures 2 and 3 provide a more detailed view of the algorithm learning process. An interesting effect of using the language model was that in most benchmarks, with very few exceptions, the average reward of LA-DSO is higher near the beginning of the training, as the pre-trained observation representations provided by the language model help "kick-start" the learning process. This also lead to finding the correct expression faster in most of cases.

3.2 *In Silico* Antibody Optimization Leveraging BERT

The human immune system's response to pathogens includes antibodies, which are proteins that sensitively and specifically recognize target molecules by the shape and chemistry of their so-called "complementarity determining regions," or CDRs [37]. CDRs and the rest of the antibody protein are composed of chains of amino acids. There are twenty common amino acids, each with distinct sizes, shapes, chemistries, and other characteristics, and sequences of amino acids are conventionally written as sequences of alphabet letters. As in language, the tremendous breadth of antibodies' recognized targets is a result of combinatorial use of this discrete vocabulary.

Instead of relying on each individual's immune system to generate effective antibodies on its own, it is possible to either identify suitable antibodies from other sources or computationally engineer the CDRs of a known, non-binding antibody toward high affinity and high specificity, enabling the generation of countermeasures to infectious diseases such as COVID-19 [6, 8, 24]. This latter approach, *in silico* antibody optimization, employs simulation and optimization methods to identify promising antibody sequences that bind strongly to a target antigen. However, these simulations are computationally expensive and the design space is large and discrete. We therefore treat the important sub-problem of finding antibody sequences that bind strongly in simulation as symbolic optimization task and evaluate LA-DSO on this task.

We demonstrate LA-DSO by starting with an anti-SARS-CoV-1 antibody, that we refer to as the *base antibody*, that is known to bind and neutralize SARS-CoV-1 but *not* the related SARS-CoV-2. Our symbolic optimization task is to modify the amino acids comprising the CDR to bind SARS-CoV-2. Rosetta Flex [4, 16] is a widely used computational tool in antibody optimization which we use as the basis of a reward function. Rosetta Flex can be used to compute the change in binding free energy between two proteins resulting from mutations in one of them, here corresponding to the change in binding strength between a base antibody-derived mutant and SARS-CoV-2.

Each free energy calculation with Rosetta requires several CPU hours to compute. To serve as a quickly-evaluated, surrogate reward capturing binding strength of generated antibody sequences, we use a Gaussian process (GP) model[3]. We start with a pre-computed

[1] https://www.wikipedia.org/
[2] https://dumps.wikimedia.org

[3] A manuscript fully describing how the GP and the ProtBERT models are trained is in preparation.

Benchmark	Expression	Wikipedia Best	Wikipedia Time	Baseline Best
Nguyen-1	$x^3 + x^2 + x$	**1.0**	−32%	1.0
Nguyen-2	$x^4 + x^3 + x^2 + x$	1.0	+22%	**1.0**
Nguyen-3	$x^5 + x^4 + x^3 + x^2 + x$	**1.0**	−41%	1.0
Nguyen-4	$x^6 + x^5 + x^4 + x^3 + x^2 + x$	1.0	0%	1.0
Nguyen-5	$\sin(x^2)\cos(x) - 1$	**0.999**	−37%	0.998
Nguyen-6	$\sin(x) + \sin(x + x^2)$	**1.0**	−5%	1.0
Nguyen-7	$\log(x + 1) + \log(x^2 + 1)$	0.996	−38%	**0.998**
Nguyen-8	$\sqrt{x}$	**1.0**	−23%	1.0
Nguyen-9	$\sin(x) + \sin(y^2)$	**1.0**	−38%	1.0
Nguyen-10	$2\sin(x)\cos(y)$	0.998	+33%	**1.0**
Nguyen-11	x^y	**0.999**	+26%	0.993
Nguyen-12	$x^4 - x^3 + \frac{1}{2}y^2 - y$	**0.812**	−31%	0.804

Table 1: Experimental results from the Symbolic Regression domain over 150 repetitions. The numeric values mean the average quality of the best equation found by the end of the training (*Best*, with 1.0 meaning perfect match in all 150 repetitions) and the percentage of change in learning time compared to the baseline (*Time*, less is better). The maximum number of iterations is limited to 2000. Best results for each benchmark highlighted in bold in the *Best* column.

collection of 122,000 simulations, where each simulation evaluates an estimated change in binding free energy of a base antibody mutant and SARS-CoV-2. Due to the size of the dataset, we trained an approximate sparse Gaussian Process [31] with scaled RBF kernel and hyperparameters selected via marginal likelihood maximization. This GP model is trained on the simulation results and can be evaluated in seconds as opposed to hours.

In our experiments, we considered three distinct base antibodies that are known to bind and neutralize SARS-CoV-1, namely: 80R [32], s230 [34], and m396 [40]. The 80R sequence comprises 239 amino acids, from which 38 positions can be identified as either within the CDRs or being otherwise directly relevant to binding. We refer to these identified positions as *mutable positions*. Similarly, s230 and m396 sequences are 239 and 458-amino acid long with both having 23 mutable positions. Thus, LA-DSO must sample alternate amino acids for a subset of mutable amino acid positions (tokens), which as a sequence define a mutated version of the base antibody. We limit the maximum number of amino acid mutations in a single sequence to 5. The final training loop then becomes sampling sequences of amino acids and using the GP surrogate model to score them.

Given the sequence-based description of proteins as a string of amino acids, using a language model to represent a given partial protein might improve learning. For our LA-DSO language model, we leverage the ProtBERT language model [10], which is a BERT model trained on a large corpus of protein amino acid sequences. Here we use a modified version of this model that was fine-tuned specifically on human antibody sequences[3]. For each of the mutable positions on the CDR, ProtBERT computes the probability of each of the 20 possible amino acids given the context of the rest of the antibody's amino acid sequence. In LA-DSO, we provide the embeddings from the last hidden layer as additional state information to the DSO RNN (i.e., a length-1024 embedding vector).

We evaluate the performance of two approaches in the antigen-binding task:

- **Baseline**: DSO without using language models; and
- **LA-DSO**: LA-DSO using our version of the ProtBERT model.

3.2.1 Experimental Results. Figure 4 shows the observed experimental results. In this domain, using the language model results in a very clear empirical advantage. The average reward of sampled sequences is significantly better throughput the training process. Moreover, the best sequence found by the end of training results in higher reward (i.e., stronger predicted binding). Both the achieved speed up and the best final resulting antibody is consistent across all three evaluated tasks, showing that LA-DSO indeed achieves more efficient learning performance in this domain. Those empirical outcomes show that the language model provided a useful abstraction to learn how to manipulate the amino acid sequences, and represent a very exciting prospect to any task in which the representation provided by the protein language model could be used for.

4 RELATED WORKS

Transfer Learning is pervasive in modern NLP applications [1]. The very existence of embedding learning algorithms such as GloVe [25] and Word2Vec [20] imply the use of knowledge reuse across multiple tasks, as embeddings provide word representations to be used in varied applications. Similarly, successful recent NLP models such as BERT [9] and GPT-3 [5] are already developed to be easily reusable. However, those publicly available general-purpose models and embedding-learning algorithms are focused on natural language specifically, which is not necessarily the case for all symbolic optimization (or more generally decision-making) applications.

Language models have already been used for supporting learning in RL tasks in the literature. Reid *et al.*'s [30] work is perhaps the most similar to ours, where they bootstrap offline RL tasks reusing language models. However, they start from general-purpose models trained in different domains, while we leverage language models based on datasets of well-formed token sequences. Chai *et al.*'s [7] work, in which they use a BERT model to process natural language observations for an RL agent, is also related to ours. Notice,

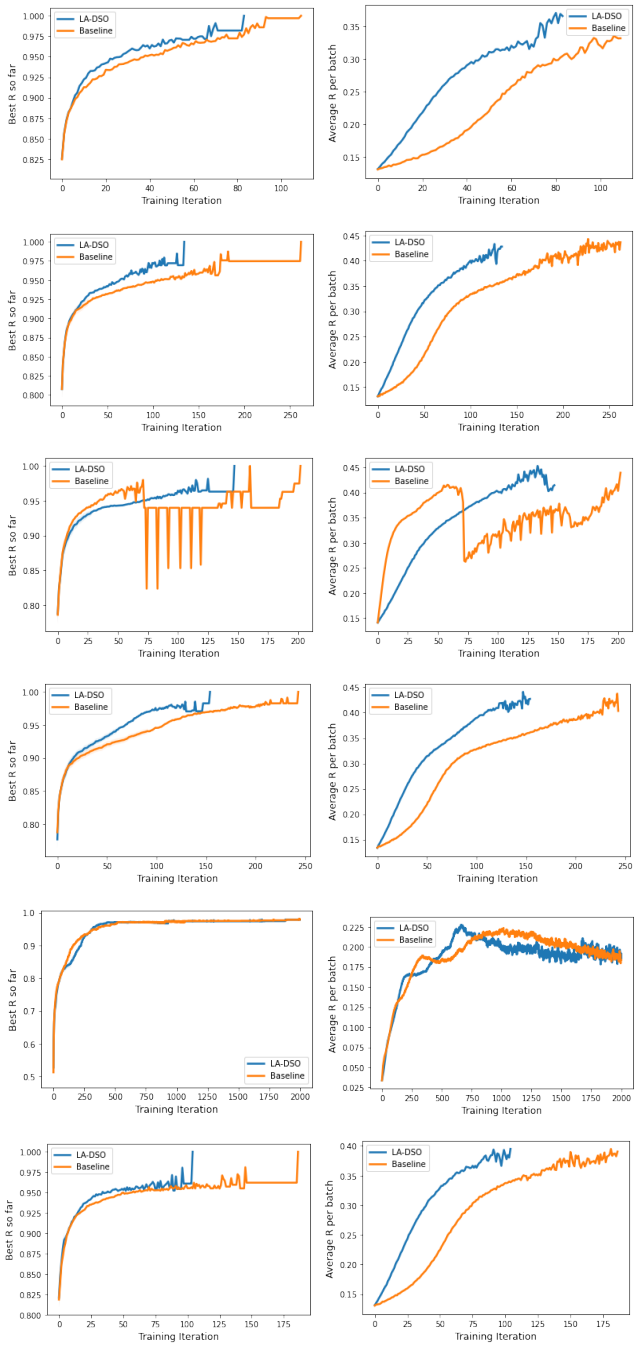

Figure 2: Learning curves for the symbolic regression domain (Nguyen 1-6). Graphs in the left show the reward from the best token sequence found so far and the ones in the right show the average of rewards on sampled sequences in a particular iteration. Shaded area represents the 99% confidence interval.

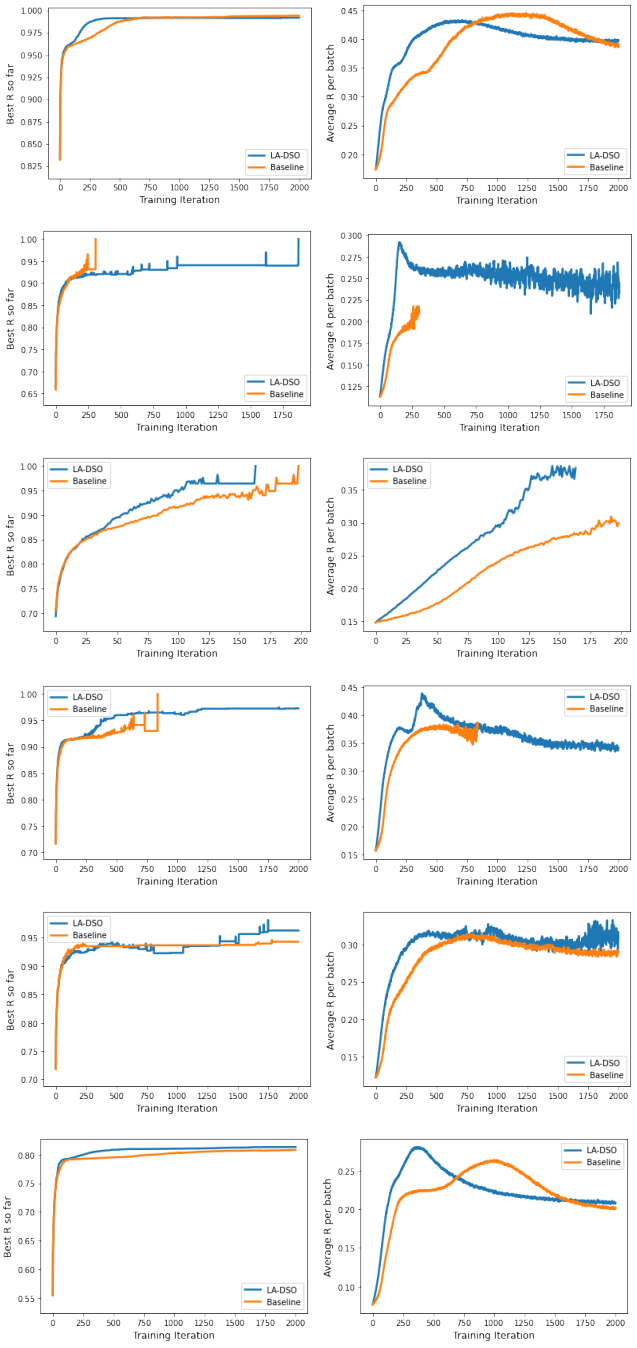

Figure 3: Learning curves for the symbolic regression domain (Nguyen 7-12). Graphs in the left show the reward from the best token sequence found so far and the ones in the right show the average of rewards on sampled sequences in a particular iteration. Shaded area represents the 99% confidence interval.

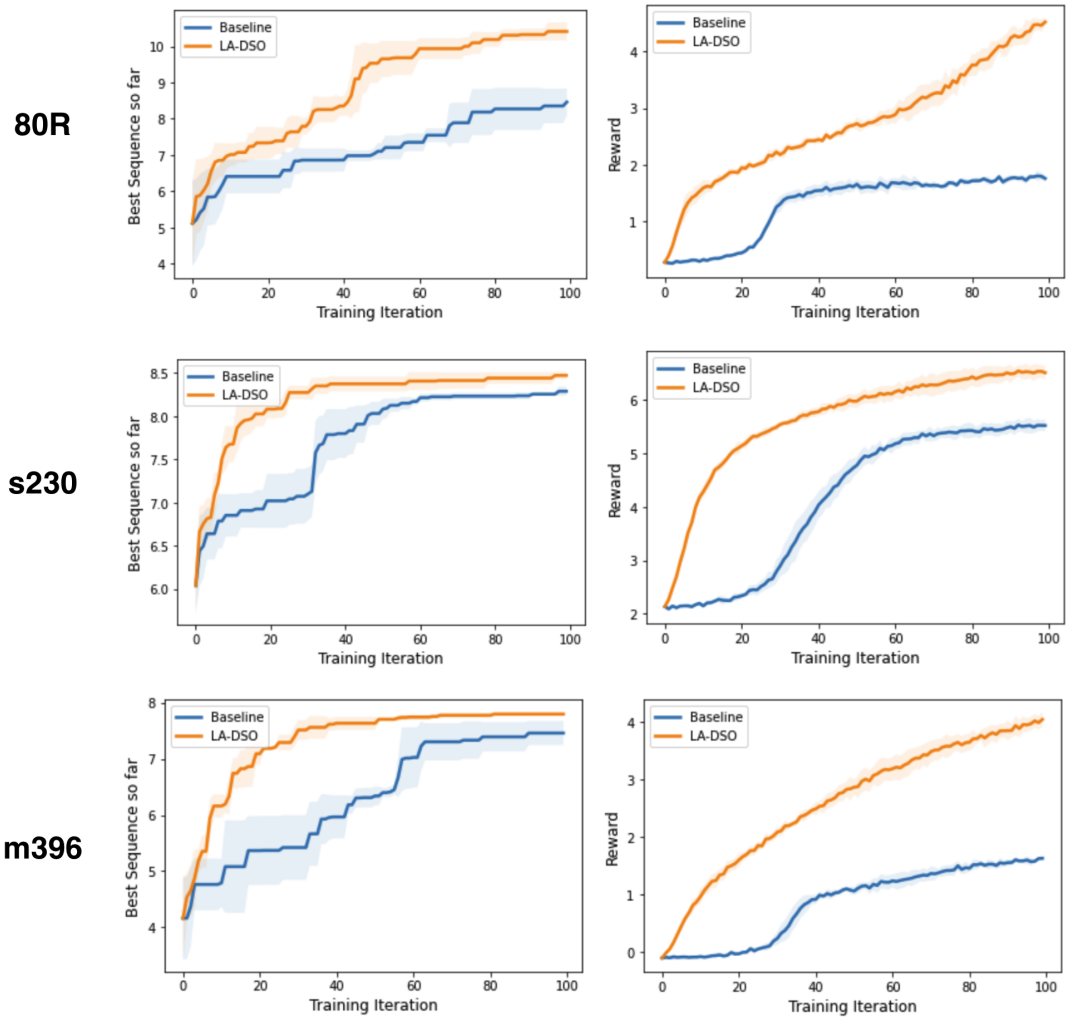

Figure 4: Experimental results in the antibody optimization domain for the three evaluated base antibodies. (left) is the best sequence found so far in a particular iteration and (right) is the average quality of explored sequences at a particular iteration. The results are averaged across 4 repetitions. The shaded area represents the 90% confidence interval.

however, that their approach is limited to applications in which instructions of how to solve the task are given in natural language, and would not be applicable for symbolic optimization (or any of the here-explored applications). Similarly, other works have combined language models with RL, but focusing exclusively on NLP tasks [3, 18].

Orthogonally to our main purpose of reusing knowledge and learning more efficiently, language models have also been used as a way to map similarities across multiple tasks [23] and as a way to easily incorporate natural text data into decision making, either as a way to identify goals [2] or to process text for RL agents [12, 38]. Kim et al. [13] leveraged a language model that predict the likeness of each token to build a *in situ* prior for DSO to bias the search, though the language model itself was not direct used by the DSO RNN.

To the best of our knowledge, LA-DSO is the first approach using language model embeddings to accelerate learning in symbolic optimization tasks.

5 CONCLUSION AND FURTHER WORK

Symbolic Optimization can be used to solve various real-world problems ranging from symbolic regression to antibody optimization. Inspired by the similarity between the token representation used in symbolic optimization algorithms and the tokenized sequences

used for natural language processing, we propose *Language Model-Accelerated Deep Symbolic Optimization* (LA-DSO).

LA-DSO leverages a pre-trained language model by directly observing a language model's embedding when learning to solve the symbolic optimization task. We empirically evaluate our proposal in the two aforementioned domains, and show that LA-DSO presents advantages in both domains, providing significant learning speedups in most cases where the language model was pre-trained with data that matches the real domain.

Our next step is to further improve our approach by considering ways of further refining the language model after LA-DSO starts solving the symbolic optimization problem, which will correspond to integrating the language model and DSO into a shared network architecture. We also intend to explore alternative methods of performing transfer learning to reuse knowledge across different symbolic optimization tasks.

ACKNOWLEDGMENTS

We are grateful to Joanne Kim for the help with the wikipedia-extraction script. We also thank Claudio Santiago and Jiachen Yang for their comments in the preparation of this manuscript. This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC. LLNL-CONF-830942. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

REFERENCES

[1] Paul Azunre. 2021. Transfer learning for natural language processing. Simon and Schuster.

[2] Dzmitry Bahdanau, Felix Hill, Jan Leike, Edward Hughes, Pushmeet Kohli, and Edward Grefenstette. 2018. Learning to Follow Language Instructions with Adversarial Reward Induction. arXiv 1806.01946 (2018). arXiv:1806.01946

[3] Dzmitry Bahdanau, Felix Hill, Jan Leike, Edward Hughes, Pushmeet Kohli, and Edward Grefenstette. 2019. Learning to Understand Goal Specifications by Modelling Reward. In International Conference on Learning Representations (ICRL).

[4] Kyle A. Barlow, Shane Ó Conchúir, Samuel Thompson, Pooja Suresh, James E. Lucas, Markus Heinonen, and Tanja Kortemme. 2018. Flex ddG: Rosetta Ensemble-Based Estimation of Changes in Protein–Protein Binding Affinity upon Mutation. The Journal of Physical Chemistry B 122, 21 (2018), 5389–5399. https://doi.org/10.1021/acs.jpcb.7b11367 arXiv:https://doi.org/10.1021/acs.jpcb.7b11367 PMID: 29401388.

[5] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. arXiv 2005.14165 (2020). arXiv:2005.14165 [cs.CL]

[6] Paul J Carter. 2006. Potent antibody therapeutics by design. Nature reviews immunology 6, 5 (2006), 343–357.

[7] Duo Chai, Wei Wu, Qinghong Han, Fei Wu, and Jiwei Li. 2020. Description Based Text Classification with Reinforcement Learning. In International Conference on Machine Learning (ICML). 1371–1382.

[8] Thomas Desautels, Adam Zemla, Edmond Lau, Magdalena Franco, and Daniel Faissol. 2020. Rapid in silico design of antibodies targeting SARS-CoV-2 using machine learning and supercomputing. BioRxiv (2020). https://doi.org/10.1101/2020.04.03.024885

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, (NAACL-HLT). 4171–4186.

[10] Ahmed Elnaggar, Michael Heinzinger, Christian Dallago, Ghalia Rehawi, Yu Wang, Llion Jones, Tom Gibbs, Tamas Feher, Christoph Angerer, Martin Steinegger, Debsindhu Bhowmik, and Burkhard Rost. 2021. ProtTrans: Towards Cracking the Language of Lifes Code Through Self-Supervised Deep Learning and High Performance Computing. IEEE Transactions on Pattern Analysis and Machine Intelligence (2021).

[11] Ruben Glatt, Felipe Leno da Silva, Van Hai Bui, Can Huang, Lingxiao Xue, Mengqi Wang, Fangyuan Chang, Yi Murphey, and Wencong Su. 2022. Deep Symbolic Optimization for Electric Component Sizing in Fixed Topology Power Converters. In Workshop on AI for Design and Manufacturing (ADAM).

[12] Karl Moritz Hermann, Felix Hill, Simon Green, Fumin Wang, Ryan Faulkner, Hubert Soyer, David Szepesvari, Wojciech Marian Czarnecki, Max Jaderberg, Denis Teplyashin, Marcus Wainwright, Chris Apps, Demis Hassabis, and Phil Blunsom. 2017. Grounded Language Learning in a Simulated 3D World. arXiv 1706.06551 (2017).

[13] Joanne T Kim, Mikel Landajuela Larma, and Brenden K Petersen. 2021. Distilling Wikipedia mathematical knowledge into neural network models. In Mathematical Reasoning in General Artificial Intelligence Workshop.

[14] Emanuel Kitzelmann. 2009. Inductive programming: A survey of program synthesis techniques. In Workshop on Approaches and Applications of Inductive Programming. 50–73.

[15] Mikel Landajuela, Brenden K Petersen, Sookyung Kim, Claudio P Santiago, Ruben Glatt, Nathan Mundhenk, Jacob F Pettit, and Daniel Faissol. 2021. Discovering symbolic policies with deep reinforcement learning. In International Conference on Machine Learning (ICML). PMLR, 5979–5989.

[16] Andrew Leaver-Fay, Michael Tyka, Steven M Lewis, Oliver F Lange, James Thompson, Ron Jacak, Kristian W Kaufman, P Douglas Renfrew, Colin A Smith, Will Sheffler, et al. 2011. ROSETTA3: An object-oriented software suite for the simulation and design of macromolecules. Methods in enzymology 487 (2011), 545–574.

[17] Qiang Lu, Jun Ren, and Zhiguang Wang. 2016. Using genetic programming with prior formula knowledge to solve symbolic regression problem. Computational Intelligence and Neuroscience (2016).

[18] Jelena Luketina, Nantas Nardelli, Gregory Farquhar, Jakob Foerster, Jacob Andreas, Edward Grefenstette, Shimon Whiteson, and Tim Rocktäschel. 2019. A Survey of Reinforcement Learning Informed by Natural Language. In International Joint Conference on Artificial Intelligence (IJCAI). 6309–6317.

[19] Aaron Meurer, Christopher P Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K Moore, Sartaj Singh, et al. 2017. SymPy: symbolic computing in Python. PeerJ Computer Science 3 (2017), e103.

[20] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. arXiv 1301.3781 (2013). arXiv:1301.3781 [cs.CL]

[21] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In Eleventh annual conference of the international speech communication association.

[22] Terrell Mundhenk, Mikel Landajuela, Ruben Glatt, Claudio Santiago, Brenden Petersen, et al. 2021. Symbolic Regression via Deep Reinforcement Learning Enhanced Genetic Programming Seeding. Advances in Neural Information Processing Systems 34 (2021).

[23] Karthik Narasimhan, Regina Barzilay, and Tommi Jaakkola. 2018. Grounding Language for Transfer in Deep Reinforcement Learning. Journal of Artificial Intelligence Research (JAIR) 63, 1 (2018), 849–874.

[24] Richard A Norman, Francesco Ambrosetti, Alexandre M J J Bonvin, Lucy J Colwell, Sebastian Kelm, Sandeep Kumar, and Konrad Krawczyk. 2019. Computational approaches to therapeutic antibody design: established methods and emerging trends. Briefings in Bioinformatics 21, 5 (10 2019), 1549–1567.

[25] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In Empirical Methods in Natural Language Processing (EMNLP). 1532–1543.

[26] Brenden K Petersen, Mikel Landajuela, T Nathan Mundhenk, Claudio P Santiago, Soo K Kim, and Joanne T Kim. 2021. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. Proceeding of the International Conference on Learning Representations (ICLR) (2021).

[27] Jacob F Pettit, Brenden K Petersen, Chase Cockrell, Dale B Larie, Felipe Leno Silva, Gary An, and Daniel M Faissol. 2021. Learning sparse symbolic policies for sepsis treatment. Interpretable Machine Learning in Healthcare Workshop at ICML (2021).

[28] Mohammad Taher Pilehvar and Jose Camacho-Collados. 2020. Embeddings in natural language processing: Theory and advances in vector representations of meaning. Synthesis Lectures on Human Language Technologies 13, 4 (2020), 1–175.

[29] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. arXiv preprint arXiv:1910.10683 (2019).

[30] Machel Reid, Yutaro Yamada, and Shixiang Shane Gu. 2022. Can Wikipedia Help Offline Reinforcement Learning? arXiv 2201.12122 (2022).

[31] Edward Snelson and Zoubin Ghahramani. 2006. Sparse Gaussian processes using pseudo-inputs. Advances in neural information processing systems 18 (2006), 1257.

[32] Jianhua Sui, Wenhui Li, Akikazu Murakami, Azaibi Tamin, Leslie J Matthews, Swee Kee Wong, Michael J Moore, Aimee St Clair Tallarico, Mobolaji Olurinde, Hyeryun Choe, et al. 2004. Potent neutralization of severe acute respiratory syndrome (SARS) coronavirus by a human mAb to S1 protein that blocks receptor association. Proceedings of the National Academy of Sciences 101, 8 (2004), 2536–2541.

[33] Nguyen Quang Uy, Nguyen Xuan Hoai, Michael O'Neill, Robert I McKay, and Edgar Galván-López. 2011. Semantically-based crossover in genetic programming: application to real-valued symbolic regression. Genetic Programming and Evolvable Machines 12, 2 (2011), 91–119.

[34] Alexandra C Walls, Xiaoli Xiong, Young-Jun Park, M Alejandra Tortorici, Joost Snijder, Joel Quispe, Elisabetta Cameroni, Robin Gopal, Mian Dai, Antonio Lanzavecchia, et al. 2019. Unexpected receptor functional mimicry elucidates activation of coronavirus fusion. Cell 176, 5 (2019), 1026–1039.

[35] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. 2016. A survey of transfer learning. Journal of Big data 3, 1 (2016), 1–40.

[36] David R White, James Mcdermott, Mauro Castelli, Luca Manzoni, Brian W Goldman, Gabriel Kronberger, Wojciech Jaśkowski, Una-May O'Reilly, and Sean Luke. 2013. Better GP benchmarks: community survey results and proposals. Genetic Programming and Evolvable Machines 14, 1 (2013), 3–29.

[37] Tai Te Wu and Elvin A. Kabat. 1970. An analysis of the sequences of the variable regions of bence jones proteins and myeloma light chains and their implications for antibody complementarity. Journal of Experimental Medicine 132, 2 (1970), 211–250.

[38] Haonan Yu, Haichao Zhang, and Wei Xu. 2018. Interactive Grounded Language Acquisition and Generalization in a 2D World. In International Conference on Learning Representations (ICLR).

[39] Kaicheng Yu, Christian Sciuto, Martin Jaggi, Claudiu Musat, and Mathieu Salzmann. 2020. Evaluating the Search Phase of Neural Architecture Search. In International Conference on Learning Representations (ICLR).

[40] Zhongyu Zhu, Samitabh Chakraborti, Yuxian He, Anjeanette Roberts, Tim Sheahan, Xiaodong Xiao, Lisa E Hensley, Ponraj Prabakaran, Barry Rockx, Igor A Sidorov, et al. 2007. Potent cross-reactive neutralization of SARS coronavirus isolates by human monoclonal antibodies. Proceedings of the National Academy of Sciences 104, 29 (2007), 12123–12128.