This article has been accepted for publication in IEEE Transactions on Evolutionary Computation. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2024.3471341

Active Learning in Genetic Programming: Guiding Efficient Data Collection for Symbolic Regression

Nathan Haut, *Member, IEEE*, Wolfgang Banzhaf, *Member, IEEE*, Bill Punch

***Abstract*—This paper examines various methods of computing uncertainty and diversity for active learning in genetic programming. We found that the model population in genetic programming can be exploited to select informative training data points by using a model ensemble combined with an uncertainty metric. We explored several uncertainty metrics and found that differential entropy performed the best. We also compared two data diversity metrics and found that correlation as a diversity metric performs better than minimum Euclidean distance, although there are some drawbacks that prevent correlation from being used on all problems. Finally, we combined uncertainty and diversity using a Pareto optimization approach to allow both to be considered in a balanced way to guide the selection of informative and unique data points for training.**

***Index Terms*—Active learning, genetic programming, symbolic regression**

I. Introduction

ACTIVE learning (AL) is a method in machine learning to strategically select training data that will maximally inform the model development process [1]. This is often done in an iterative process, alternating between data collection and model development phases. Active learning can be valuable in scenarios where either data collection or data labelling is time-consuming or expensive, thus we want to minimize the total required data for model training.

Various forms of active learning exist, with three types dominating: pool-based AL, stream-based AL, and membership query synthesis [2]. Figure 1 shows a simple visual representation to compare the three methods of active learning. Pool-based and stream-based methods both have a set of training samples to choose from, with the goal of selecting and training on only a small subset of maximally informative cases. The key difference between pool-based and stream-based methods is that pool-based methods search over a set of data points for the ones that are most informative by rank. Steam-based methods differ by checking each potential training case in order one-by-one, assigning a binary decision to each case, and only admit to the training set the ones marked as "informative". Membership query synthesis approaches do not have a set of already existing training samples to choose from, instead, they search a training space to find and synthesize new training data points that are expected to maximally inform the machine learning model. Once synthesized, a new data point is then labelled by the researcher via experimentation or expert knowledge. In this work, we focus on membership-query synthesis to guide the collection of data where it doesn't yet exist in the training set.

Active learning is a versatile method with uses ranging from effective sub-sampling of data from a huge set for training, sampling of data with specific goals such as to maximize diversity, to guiding experimentation by suggesting experiments that will be most informative to the researcher in the model building process. It can be used to focus on interesting samples from large sets or to expand small data sets while minimizing data collection efforts. For example, AL has recently been used to explore a space of 16 million potential catalysts to maximize the conversion rate of methane to methanol, which without active learning would not have been possible to search effectively within a reasonable time [3]. Active learning has also been shown to effectively sub-sample training data for identifying malware-infected PDF documents [4]. The authors found that when using active learning they could reduce the training set size to 1/30-th of the original size, while maintaining the same performance as models trained on the whole set.

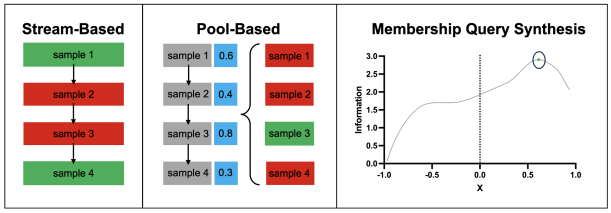

Fig. 1. The three main types of active learning: Stream-based, pool-based, and membership query synthesis are visually demonstrated. Stream-based approaches, shown on the left, search through the samples one at a time and either mark them for labelling or skip them. Green indicates a sample is found to be informative and is marked for labelling, red indicates a sample is skipped. Pool-based approaches, shown in the middle, assigns an information score to each potential training sample and the most informative sample is chosen to be labelled and added to the training set. Membership query synthesis, shown on the right, searches a space of potential points not yet collected while maximizing an information measure and selects a point to be synthesized and labelled that maximizes the information score. The selected point is indicated by the green circle, while the y-axis of the curve represents the informativeness measure and the x-axis is representative of the sample space.

Active learning approaches have been developed for a wide range of machine learning methods, e.g. for support vector machines or neural networks. In support vector machines, for instance, AL has been realized by computing the distance of all points to the separating hyperplane and selecting the point nearest the hyperplane to be labelled [5]. For neural networks, one AL variant has been to select points with the minimum difference between the two most probable predicted labels [6]. This distribution was defined as $M = P(l1|x) - P(l2|x)$, where M is the margin between the two most probable labels, $l1$ is the most probably label for input x, and $l2$ is the second most probable label for input x.

In this contribution, we apply active learning strategies

to genetic programming for symbolic regression tasks. The goal is to exploit some of the features of GP to guide data collection, in particular its reliance on a population of models. We extend our previous work where we presented preliminary results exploring only several uncertainty metrics for use in active learning [7]. Here, we utilize both uncertainty measures in a model population context and diversity measures in a data context to accelerate the discovery of models (physics equations in our study). The idea is to look for disagreement among high-quality individuals in the population as a guide to locate informative data points to add to the training set while also considering data diversity. We further explore how model uncertainty and data diversity can be used together via a Pareto optimization.

II. Related Works

Active learning methods for machine learning have shown to be very successful in applied settings to improve the method of labelling and collecting data with various machine learning types. AL has recently been demonstrated to significantly reduce the labelling efforts required for labelling data associated with identifying heart disease [8]. The authors demonstrated that they could find more accurate models using fewer data points when compared to a random point selection strategy.

AL has been applied to genetic programming classification tasks as well. Using an ensemble of GP models, the models "vote" on the class of data pairs, and points are only labelled when the committee of developing models encounters pairs that can't be classified [9]. This was found to reduce the total effort needed to label training points since only a subset had to be labelled before finding accurate models. Where GP training sets are large, AL has been successfully applied by selecting sub-samples to be used for training [10], [11]. In [11] AL is performed by segmenting the data into smaller blocks and training the models using one randomly selected block at a time using uniform probability. As training continues, bias is introduced into the probability by increasing the tendency to select blocks that haven't been seen in a while, as well as blocks where the models performed poorly during training. AL for sub-sampling with genetic programming was found to decrease training times to find better binary classification models by an order of magnitude [11]. In [10] subsets were selected by dynamically developing a fitness case topology that could be used to create minimally related subsets of data. In this context, the strength of a relationship between two training cases was indicated by the number of individuals that were able to solve both training cases. Active learning has also been applied to the task of discovering regular expressions using genetic programming [12]. In that work, they used a restricted query-by-committee (rQbC) strategy that utilized the top 25% of models in a population to generate "extraction queries", in which the user then indicates whether or not the character string selected by the "extraction query" should be extracted or not by a regular expression.

In the discovery of biological networks AL methods have also been employed successfully [13]. Several different approaches were explored by the authors for determining which new data points would be maximally informative for a wide range of machine learning models, including Boolean networks, causal Bayesian networks, differential equation models, etc. One approach the authors explored was the maximum difference method in which two best-fit models are chosen and a new data point is selected where those two best-fit models have the largest difference in predictions. They also examined entropy score maximization. In that method, a new data point is selected that maximizes an entropy score, where entropy can be thought of as the amount of information to be gained by gathering that data point. The entropy score H_e is computed as follows:

$$H_e = -\sum_{x=1}^{x_e} \frac{e_x}{|M|} \log_2 \frac{e_x}{|M|}$$

where M is the set of Boolean networks, x_e is the number of network states for a given data point, and e is the set of all potential data points.

In chemical engineering AL has been applied to expedite a reaction screening process by only selecting a subset of maximally informative experiments to complete rather than by exhaustively performing all possible experiments [14]. This was done by training neural networks and using them to select a subset of experiments that maximized the information gain. Maximal information gain was determined by looking at the standard deviation of an ensemble of neural networks.

Kotanchek et al. [15] used genetic programming for active design of experiments, where models developed by a GP system are used to find optimal conditions in a system of study. Active design of experiments is an application of active learning, where it has the goal of designing experiments that have specific properties or yield maximal information. The authors proposed to employ ensembles of models from symbolic regression to find regions of uncertainty in order to gather new data with high information content. While this method has been proposed for how an active learning method using model ensembles could be applied to GP for symbolic regression, there has yet to be any research showing how active learning methods affect the performance of GP symbolic regression tasks or how the method to quantify uncertainty affects the quality of points selected for inclusion in the training data. As well, it is yet to be shown that this idea of selecting an ensemble from a model population and searching for points of high uncertainty or disagreement among models is generalizable to any machine learning method where a population of models is available.

III. Methods

We compare two classes of active learning: uncertainty and diversity-based. The implementations are described in detail below and summarized in Figures 2 and 3. We use two random sampling methods as a baseline to compare the performance of the active learning methods. The key features of the GP system we used, StackGP, are also discussed.

A. Active Learning

Two general types of active learning were implemented to work with StackGP for the purpose of accelerating the

This article has been accepted for publication in IEEE Transactions on Evolutionary Computation. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2024.3471341

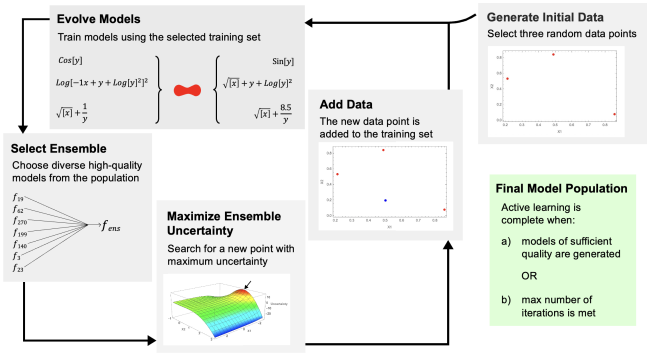

Fig. 2. An overview of the iterative uncertainty-based active learning approach. It begins with an initially randomly selected dataset. It then iteratively evolves models and selects new training points that maximize uncertainty of an ensemble of models. By maximizing ensemble uncertainty to select new training samples, points with relatively high information content are added to the training set each iteration.

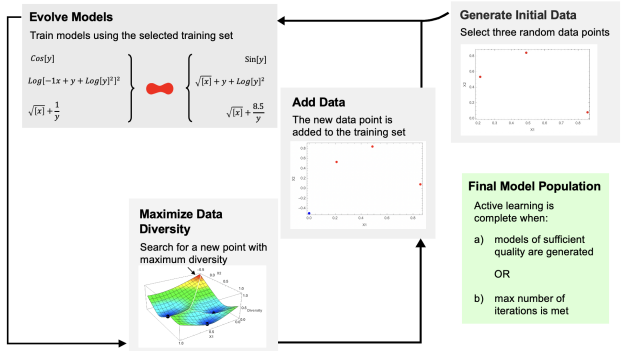

Fig. 3. An overview of the iterative diversity-based active learning approach. It begins with an initially randomly selected dataset. It then iteratively evolves models and selects new training points that maximize data diversity. By maximizing data diversity to select new training samples, points with new information are added to the training set each iteration.

development of models to fit physics data from the Feynman Symbolic Regression Dataset [16]. The first type of active learning explored was uncertainty-based, a model-driven approach to active learning, where an ensemble of diverse, high-quality models from a population was used to search for regions in the search space where there was high uncertainty or disagreement between the models. The goal of uncertainty-based AL is to identify new training points where the models disagree most in the predicted responses/labels given the input features of those points. The second type of active learning explored was diversity-based active learning, where new points are selected that differ maximally from the points already in the training sample. This second type of active learning is a data-driven approach rather than a model-driven approach, unlike traditional active learning approaches. The first type of active learning, uncertainty-based, is summarized in Figure 2 and the second type, diversity-based, is summarized in Figure 3.

Both types of active learning methods were implemented to determine how they each impact the success of evolution in genetic programming symbolic regression tasks. Several different uncertainty and diversity metrics are implemented to determine their respective impact on the success of the task. Success of active learning by maximizing uncertainty would indicate that the diversity of the population can be utilized to guide the collection of informative data. Success of diversity sampling would indicate that GP symbolic regression model development benefits from improved data sampling.

1) Maximizing Uncertainty: Several different uncertainty metrics were explored to determine how different measures impact the success of active learning, and which approach would generally work best. As an overview, each approach begins by selecting an ensemble of models using the same method, then a function that uses the specific uncertainty metric along with the ensemble and current training set is created. This function is then fed to an optimizer to search for regions of relatively high uncertainty. The most uncertain point found is then returned and selected to be added to the training set. In total, there were 6 different uncertainty maximization approaches tested which varied in how they quantified disagreement, whether outlier predictions were considered, and which optimizer was used. The steps and methods will be described in greater detail below and the entire process is depicted in Algorithm 1.

Generating the ensemble is the first step in uncertainty-based active learning. The goals for generating the ensemble were to capture diverse, high-quality individuals from the population while keeping the size of the ensemble relatively small so that the computational cost of optimizing uncertainty is reasonable. The diversity goal is essential to the success of active learning since disagreement between models is a necessary requirement. The method chosen to capture both diversity and quality from the model population works by clustering the training data using the input space and selecting a model that best fits each cluster, ensuring no model is selected more than once. If a model is already selected by another cluster, the next best unselected model is chosen. The minimum number of clusters is set to 3 and the maximum is set to 10. Thus, 3-10 models are chosen for inclusion in an ensemble. Data clustering was chosen with the intent to capture diversity by focusing on models that have biases for different regions of the training space. Quality in the population would be captured since only models with the best fitness were selected for each cluster. The algorithm to generate the ensemble is described in detail in Algorithm 2.

Algorithm 1 AL Process Using Model Uncertainty

$TrainingData \leftarrow 3StartingPoints$ ▷ Generate initial random training data
$Models \leftarrow RandomModels$ ▷ Generate initial random models
$Models \leftarrow Evolve(TrainingData, Models)$ ▷ Train models on starting data
while $BestModelError \neq 0$ **do** ▷ While perfect model not found
  $Ensemble \leftarrow EnsembleSelect(Models)$. ▷ Select ensemble of models
  $NewPoint \leftarrow MaxUncertainty(Ensemble)$ ▷ Find point of max uncertainty
  if $NewPoint \subset TrainingData$ **then** ▷ If point already selected
    $NewPoint \leftarrow MaxUncertainty(SubSpace(Ensemble))$ ▷ Search a subspace
  end if
  $TrainingData \leftarrow Append(TrainingData, NewPoint)$ ▷ Add new point
  $Models \leftarrow Evolve(TrainingData, Models)$ ▷ Evolve new models with new data using best models to seed evolution
end while

Algorithm 2 Ensemble generation process to select diverse high-quality models.

procedure EnsembleSelect($models$,$trainingData$,$responseData$)
  $selectedModels \leftarrow []$ ▷ Initialize ensemble
  $nClusters \leftarrow min(len(trainingData), 10)$ ▷ Determine number of clusters
  $clusters \leftarrow KMeans(nClusters).fit\_predict(trainingData)$
  for $i = 0; i++; i < nClusters$ **do** ▷ Loop over data clusters
    $modelErrors \leftarrow computeError(models, clusters[i])$
    $sortedModels \leftarrow sortBy(models, modelErrors)$
    $j = 0$
    while $sortedModels[j]$ in $selectedModels$ **do** ▷ Find best unselected model
      $j++$
    end while
    $selectedModels = join(selectedModels, sortedModels[j]$ ▷ Add to ensemble
  end for
  return $selectedModels$ ▷ Return ensemble
end procedure

The second step of this method is to utilize the specified uncertainty function with both the current training data and the selected ensemble. The function is then given to the optimizer with the search space boundaries to find a point of relatively high uncertainty. In the case that an already selected point is re-selected, a new search is initiated within a random sub-region until a unique point is added. This ensures that new information is added in each iteration to the training set.

The two methods used for optimization were Scipy Optimize's minimize and differential evolution [17], [18].

In total 5 different uncertainty metrics were used, shown by Equations 1 to 5, where Equation 5 is used twice, once with Scipy's minimize function for optimization, and a second time with Scipy's differential evolution function for optimization.

$$\Delta = \frac{\text{Std(EnsembleResponses)}}{\text{Mean(Abs(EnsembleResponses))}} \tag{1}$$

$$\Delta = \frac{\text{TrimmedStd(EnsembleResponses, 0.3)}}{\text{TrimmedMean(Abs(EnsembleResponses), 0.3)}} \tag{2}$$

$$\Delta = \frac{\text{Std(EnsembleResponses)}}{\text{TrimmedMean(Abs(EnsembleResponses), 0.3)}} \tag{3}$$

$$\Delta = \text{Std(EnsembleResponses)} \tag{4}$$

$$\Delta = \text{DifferentialEntropy(EnsembleResponses)} \tag{5}$$

2) Point Diversity: A data-driven active learning approach was also explored, aiming to maximize data diversity rather than maximize ensemble uncertainty. This approach is described in Algorithm 4. The goal was to determine if GP evolution for symbolic regression tasks would benefit significantly from improved sampling of the data for training. Two different metrics were used to quantify diversity: point distance and point correlation. Point distance was implemented by measuring both the minimum and average Euclidean distance to all points in the training set. Minimum distance indicates the distance to the nearest point in the training set. Mean distance indicates the average distance to all points currently in the training set. Point correlation was defined as the average correlation to all points in the training set. When selecting a new point, the goal was to either maximize the distance metric (minimum or mean) or minimize the correlation to the current training set.

To minimize the correlation when selecting a new point, Pearson's R^2 was computed between each point and the potential new point. The equation for computing Pearson's R is shown in Equation 6. Here y represents the new training point, $\hat{y}$ represents a point already in the set, and each instance d represents the value in the dth dimension of the point with total dimensionality of D. The overall method for computing the joint correlation of a new point to the training set is summarized in Algorithm 3.

$$R = \frac{\sum_{d=1}^{D}(y_d - \bar{y})(\hat{y}_d - \bar{\hat{y}})}{\sqrt{\sum_{d=1}^{D}(y_d - \bar{y})^2 \times \sum_{d=1}^{D}(\hat{y}_d - \bar{\hat{y}})^2}} \tag{6}$$

Algorithm 3 Method to Compute Correlation to Training Data

procedure JointCorrelation($trainingSet$,$newPoint$)
  $r2Values \leftarrow [PearsonR(trainPt, newPoint)^2$ for $trainPt$ in $trainingSet]$ ▷ R^2 vals
  $avgCorr \leftarrow mean(r2Values)$ ▷ Compute average correlation
  Return $avgCorr$
end procedure

Algorithm 4 AL Process Using Data Diversity

$TrainingData \leftarrow 3StartingPoints$ ▷ Generate initial random training data
$Models \leftarrow RandomModels$ ▷ Generate initial random models
$Models \leftarrow Evolve(TrainingData, Models)$ ▷ Train models on starting data
while $BestModelError \neq 0$ **do** ▷ While perfect model not found
  $NewPoint \leftarrow MaxDiversity(TrainingData)$ ▷ Find point of max uncertainty
  if $NewPoint \subset TrainingData$ **then** ▷ If point already selected
    $NewPoint \leftarrow MaxUncertainty(SubSpace(TrainingData))$ ▷ Search a subspace
  end if
  $TrainingData \leftarrow Append(TrainingData, NewPoint)$ ▷ Add new point
  $Models \leftarrow Evolve(TrainingData, Models)$ ▷ Evolve new models with new data using best models to seed evolution
end while

3) Benchmark Testing: Each active learning approach was compared on a benchmark set of 35 of the 100 equations from the Feynman Symbolic Regression Dataset [19]. These particular 35 problems were selected since they were thought to be most appropriate for a study in active learning. In a previous study, 37 other of the 100 equations were consistently found to need just 3 data points to be solved when using StackGP [20]. This would render active learning useless in such cases. The remaining 28 equations generally required all the data points up to 1000 (as we tested) to reach moderate results, so it did not seem that this type of active learning, adding one point at a time, would be appropriate for those problems.

B. StackGP

StackGP is a stack-based genetic programming implementation in Python [20] and is available here ([21]).

1) Model Structure: Similar to PushGP [22], StackGP models use multiple stacks, where the model evaluation is driven by an operator stack while variables, constants, and other data types are stored on separate stacks. For symbolic regression tasks, we have a total of 2 stacks, the operator stack and the variables/constants stack.

2) Correlation Fitness Function: Unlike many symbolic regression implementations that use (R)MSE as the fitness function, we employ correlation as the fitness function, together with a linear scaling post-processing step. This was shown to perform better than (R)MSE in earlier work [23]. The fitness is optimized during search by first maximizing R^2, which is computed using Equation 7, where N is the number of data points i, y_i is the target output, and $\hat{y}_i$ the output calculated by the model.

$$R = \frac{\sum_{i=1}^{N}(y_i - \bar{y})(\hat{y}_i - \bar{\hat{y}})}{\sqrt{\sum_{i=1}^{N}(y_i - \bar{y})^2 \times \sum_{i=1}^{N}(\hat{y}_i - \bar{\hat{y}})^2}} \tag{7}$$

The search is then completed using a post-processing step, which aligns the resulting models via a simple linear regression step (eq. 8), minimizing

$$\underset{a_0,a_1}{\operatorname{argmin}} \sum_{i=1}^{N}(|y_i - (a_1\hat{y}_i + a_0)|) \tag{8}$$

3) Algorithm: An overview of the algorithm is shown in Algorithm 5. The parameters used to run the algorithm are shown in Table I. Note that crossover and mutation calls in the algorithm are simplified and actually represent applying crossover and mutation to the correct fractions of models as shown in the parameters.

Crossover is performed using a 2-point crossover operator where two points are selected in the operator stack of each parent and the operators, along with the associated variables and constants between the points, are swapped between the parents. Mutation has several different forms, each occurring with equal probability: random replacement of a variable, random replacement of an operator, pushing a random operator to the top of the operator stack and pushing variables/constants to the second stack when arity is greater than 1, popping a random number of operators off the operator stack and the correct number of variables/constants off the second stack, inserting a single operator at a random position in the stack, 2-point crossover with a random model, and appending a random operator to the bottom of the operator stack. There is then a repair mechanism that will push variables and constants to the top of the second stack if - after mutation - there are not enough items in the variable/constant stack for the operators.

The tournament selection method used was Pareto tournament selection, where correlation and complexity were the two objectives. Complexity was measured as the combined stack lengths. This implementation of Pareto tournament selection follows [24], where from each tournament the set of all non-dominated individuals across the specified objectives are returned as winners.

TABLE I
STACKGP & ACTIVE LEARNING PARAMETER SETTINGS

Parameter	Setting
Mutation Rate	79
Crossover Rate	11
Spawn Rate	10
Elitism Rate	10
Crossover Method	2 Pt.
Tournament Size	5
Population Size	300
Selection Rate	20
Parallel Runs	4
Generations	1000

Algorithm 5 StackGP Search Algorithm

procedure EVOLVE($trainingData$,$models$)
  for generations 1 to 100 **do**
    $models \leftarrow setModelQuality(models, trainingData)$
    $newPop \leftarrow ElitismSelection(models, 20\%)$
    $models \leftarrow tournamentSelection(models)$
    $newPop \leftarrow newPop + crossover(models) + mutation(models)$
    $newPop \leftarrow newPop + randomNewModels$
    $newPop \leftarrow deleteDuplicates(newPop)$
    $models \leftarrow newPop$
  end for
  $alignedModels \leftarrow alignment(models, trainingData)$
  Return $alignedModels$
end procedure

C. Random Sampling

As a baseline, we used random sampling of data points from uniform and normal distributions to determine if an active learning method improves learning progress over a naive sampling of training data. Uniform random sampling was chosen since it is a commonly used distribution and would likely be a first choice for naively sampling data. A normal distribution was selected since according to the central limit theorem, normal distributions tend to arise in nature, so a data set sampled from natural processes would likely be a normal distribution.

To create a fair comparison against the active learning methods, a simple substitution was made where instead of using active learning to maximize uncertainty or diversity, a random point was added in each iteration, sampled from the specific distribution (uniform or normal). Beyond that substitution, the algorithm remains the same.

The normal distribution for each variable was defined using the midpoint between the sampling bounds as the mean and $1/6$ of the difference between the upper and lower bounds as the standard deviation. This places 99.8% of the distribution between the upper and lower bounds of each variable. If a point is sampled beyond a boundary it is adjusted to be on the boundary instead, although this is unlikely to occur frequently.

IV. RESULTS & DISCUSSION

Several different approaches for computing uncertainty and diversity were compared using the Feynman Symbolic Regression Dataset. We then combine diversity and uncertainty using a Pareto optimization approach and compare that multi-objective method to using both uncertainty and diversity alone. The Pareto approach is then tested on two additional benchmark problems from the SRBench benchmark set.

This article has been accepted for publication in IEEE Transactions on Evolutionary Computation. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2024.3471341

A. Active Learning Uncertainty Sampling

The results of comparing the different uncertainty-based active learning methods are shown in Figures 4 and 5 and the full table is in the Appendix as Table IV. Figure 4 uses uniform random sampling as the baseline for comparison, shown as the blue line in the figure. We also include normally distributed random sampling for comparison as the red distribution. The results show that the relative uncertainty measures, where we divide by the mean or trimmed mean, do not consistently perform better than uniform random sampling. The non-relative uncertainty measures performed well more consistently with the methods that use differential entropy performing best. The fact that standard deviation alone as an uncertainty metric performs consistently well is appealing since it is very cheap and easy to implement relative to some of the others. Differential entropy when using differential evolution as the optimizer performed best. The fact that differential evolution as the optimizer worked best with differential entropy likely indicates that the surface is highly non-convex, so differential evolution was better able to search the uncertainty space.

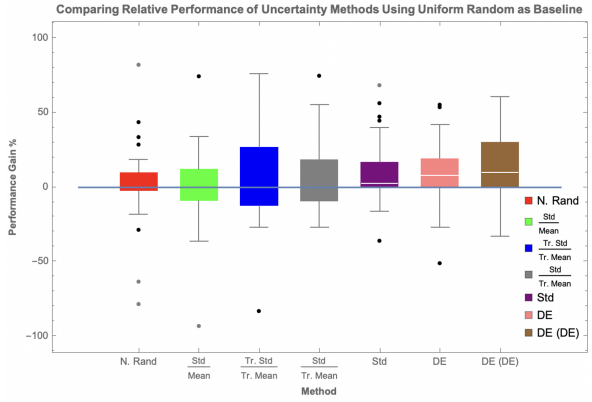

Fig. 4. **Comparing Relative Performance of Uncertainty Methods Using Uniform Random Selection as Baseline.** Shown here are the performance differences of AL uncertainty methods compared to uniform random selection as the baseline (blue line) and normally distributed random selection (red distribution). We see that using the relative uncertainty measures where we divided by the mean we get inconsistent performance, sometimes performing much better than random but sometimes performing much worse. The non-relative approaches all consistently perform better than random selection with the methods that use differential entropy performing best. Using differential entropy with differential evolution (brown) we observe the best performance. The distributions represent the median performances of 100 independent runs across all test problems. For completeness, there is one point not shown for the std/tr. mean approach that is around -200.

Figure 5 compares the performance of each method against uniform random sampling for each problem and displays the number of times each method outperforms or underperforms random sampling. If a method outperforms random sampling that means that the method required fewer points to solve a problem. If a method underperforms random sampling that means that the method required more points to solve a problem. The results show that the methods using differential entropy work best, outperforming in the most number of cases and underperforming in the fewest number of cases. The differential entropy method that used differential evolution as the optimizer worked better than just using differential entropy

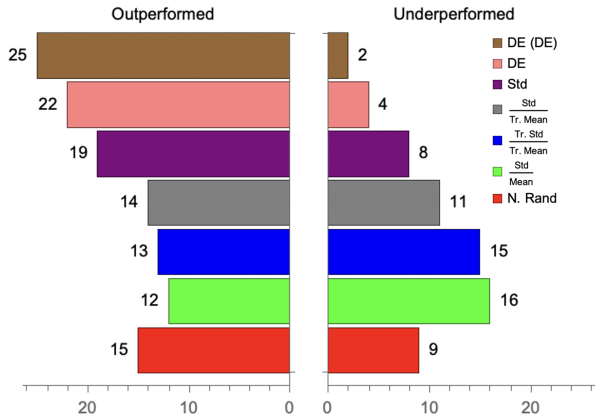

Fig. 5. **Comparing Performance of Uncertainty Methods Against Uniform Random Selection.** Each method is compared to uniform random sampling and the number of times that the method outperforms and underperforms is reported. The number of times each method outperforms is shown on the left and the number of times each method underperforms is shown on the right. Outperforming means that a method used fewer points than uniform random sampling. Underperforming means that it required more points. Ties are not counted but can be easily determined by taking the difference of 35 and the two values reported. The results show that the methods that use differential entropy work well most consistently, outperforming more frequently and underperforming infrequently. We can also see that the relative uncertainty measures were very inconsistent in their performance.

with SciPy Optimize's minimize function. This indicates that differential evolution was able to search the uncertainty surface more effectively. The results also show that the relative uncertainty methods that divided the mean or trimmed mean were not consistent in their performance, frequently having a similar number of cases where the methods outperformed and underperformed.

We see that the relative measures sometimes perform well and sometimes perform poorly, but on average they are centered around the baseline performance. The original assumption was that the relative uncertainty measures would be appealing since it was thought that they would reduce a bias towards selecting points where the predicted response is larger and thus naturally leads to wider distributions of the ensemble. This may have been the case occasionally where those methods did perform much better than uniform random sampling, but they were not consistent. Looking at their formulations there is a risk of selecting points where the mean is near 0 which results in asymptotic behavior of the uncertainty function.

Considering the results, we also see that of the two random sampling methods, normally distributed random sampling seems to perform a bit better than uniform sampling. This indicates that if a researcher does not want to use active learning to guide their data collection, they would typically be better off using a normal distribution than a uniform distribution for their samples.

B. Active Learning Diversity Sampling

The different metrics for determining point diversity were compared to determine if there are clear differences in what they are measuring and also to ensure there aren't any obvious flaws with any of the metrics. When comparing minimum

This article has been accepted for publication in IEEE Transactions on Evolutionary Computation. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2024.3471341

distance and average distance an initial randomly generated training set with 3 data points in 3 dimensions was generated. Figure 6 shows the comparison where new points were selected iteratively to add to the training set using the minimum distance metric for selection. We can see visually that the correlation, R^2, is weak between the two, indicating they are providing different measures. As well, we recorded the Spearman Rho, rank-correlation, since that indicates if the methods are ranking points similarly or not. If methods rank points similarly, then they would likely not provide unique information if used as a diversity metric. It was found that the Spearman Rho was 0.44, which means that the two methods are ranking points differently and could provide unique information.

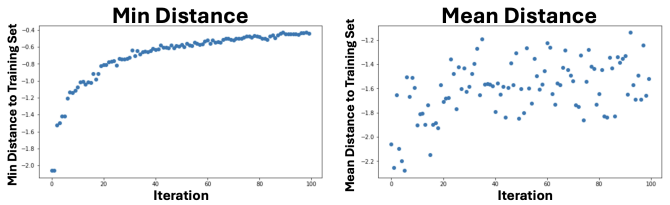

Fig. 6. Comparing minimum Euclidean distance against mean Euclidean distance as a diversity metric. Here minimum distance is used to select the next point in the set and both metrics of those points are displayed. We can see that there is little correlation between the two metrics indicating they provide different information. The R^2 between these two metrics on these points is just 0.37. The Spearman Rho, rank-correlation, is also low at 0.44.

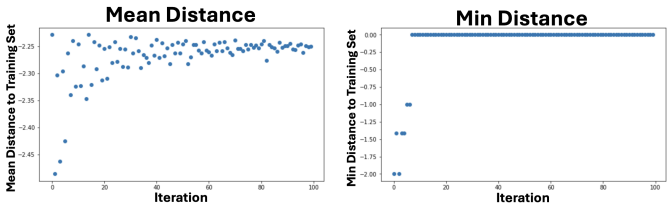

Fig. 7. Comparing minimum Euclidean distance against mean Euclidean distance as a diversity metric. Here mean distance is used to select the next point in the set and both metrics of those points are displayed. We can see that when mean distance is used to select new points, we get many points with a minimum distance of 0. This indicates that we are very frequently reselecting points already in the set. This shows that minimum distance is a better metric than mean distance.

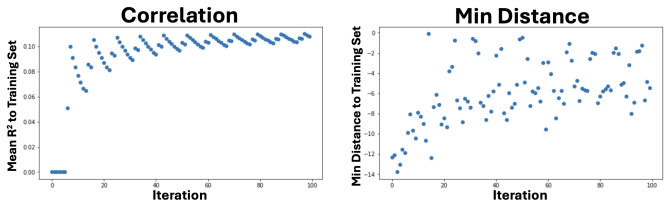

Fig. 8. Comparing minimum Euclidean distance against mean point correlation as a diversity metric. Here minimizing mean correlation is used to select the next point in the set and both metrics of those points are displayed. We can see that there appears to be a weak positive correlation between the two, indicating that they provide some of the same information but are not the same, so may have different advantages. It is also promising that the minimum distance shows that we are not reselecting points already in the training set. Comparing the metrics for these points we get an R^2 of 0.35 and a Spearman Rho of 0.33.

To further compare the minimum and mean distance metrics, the analysis was flipped, such that mean distance was used to select new points and both metrics were recorded on the selected points. These results are shown in Figure 7. Here it becomes obvious that mean distance is not a good metric since the minimum distance metric indicates that we are repeatedly selecting points already in the set. This is shown by the consistent minimum distance value of 0 after around 10 iterations. This result led to mean distance being thrown out as a potential choice of metric.

Minimum distance and correlation were also compared to determine if they provide unique measures of diversity. The results are shown in Figure 8. For this analysis, lack of correlation to the training set was used to select new points and both metrics were recorded. This analysis was slightly different than the previous ones since for this problem the points were embedded in a 10 dimensional space instead of just 3. The results show that the two metrics do provide unique information since an R^2 value of 0.35 and a Spearman Rho value of 0.33 were recorded, which are both low. Since these metrics were determined to provide unique information without any clear flaws both were included to be explored, with the one limitation that correlation as a diversity metric could not be used on problems of less than 3 dimensions.

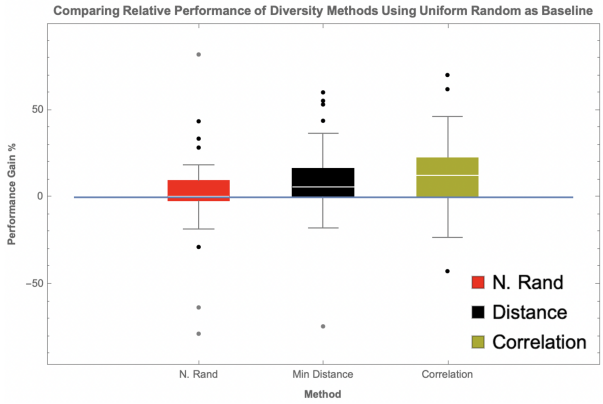

Fig. 9. **Comparing Relative Performance of Diversity Methods Using Uniform Random Selection as Baseline.** Shown here are the performance differences of both the AL diversity methods compared to uniform random selection as the baseline (blue line) and normally distributed random selection (red distribution). We see that using minimum distance (black distribution) performs consistently better than the baseline and correlation (green distribution) works best as a diversity metric. The drawback with using correlation as the diversity metric though is that it requires problems with more than two dimensions, so the problems with two dimensions are ignored when using correlation. The distributions represent the median performances of 100 independent runs across all test problems.

The results of comparing the different data diversity-based active learning methods are summarized in Figures 9 and 10 and the full results are shown in Table V in the Appendix. Figure 9 uses uniform random sampling as the baseline for comparison, shown as the blue line. We again include normally distributed random sampling for comparison as the red distribution. We can see that both diversity metrics have better performance than uniform random sampling, on average requiring fewer training points to find a solution. We also see that correlation as a diversity metric performs best, often

This article has been accepted for publication in IEEE Transactions on Evolutionary Computation. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2024.3471341

requiring the least number of training data points to find a solution. Correlation does have the disadvantage, though, of not working on the problems with just two dimensions. Those two problems are not represented in the correlation bar in the chart since they are not applicable.

Figure 10 shows the number of cases where each method either outperformed or underperformed when compared to uniform random sampling. We see again that correlation has the best performance. This indicates that not only does correlation lead to requiring fewer training points on average, but also indicates that it most consistently requires fewer points. We see that distance as a metric requires fewer points than uniform random and normal random sampling, but is not as consistent as correlation.

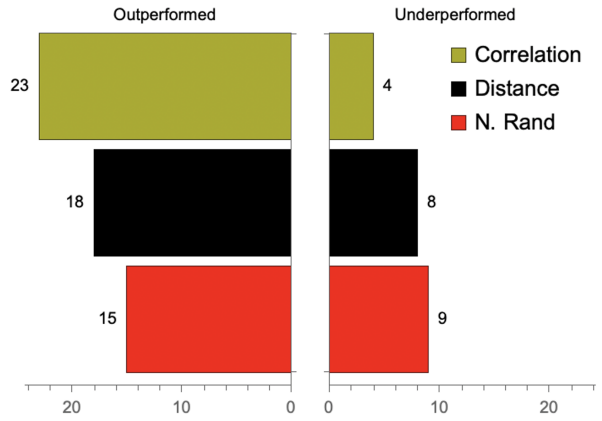

Fig. 10. **Comparing Performance of Diversity Methods Against Uniform Random Selection.** Each method is compared to uniform random sampling and the number of times that the method outperforms and underperforms is reported. The number of times each method outperforms is shown on the left and the number of times each method underperforms is shown on the right. Outperforms means that a method used fewer points than uniform random sampling. Underperforms means it required more points. Ties are not counted but can be easily determined by taking the difference of 35 and the two values reported. The results show that correlation performed best, underperforming the fewest times and outperforming the most.

C. Comparing Diversity, Uncertainty, and Pareto Optimization of Both

Next we explore how the performance compares when using uncertainty and diversity together to see if there are benefits to considering both for selecting training data with AL compared to just uncertainty or diversity alone. For this comparison, we selected one diversity metric and one uncertainty metric. For the uncertainty metric, we chose differential entropy since it was shown to be the best performing metric in Figure 4. For the diversity metric, we chose minimum distance. Although it didn't perform best, it is most versatile since it isn't restricted to problems with more than 2 dimensions. For the combination method, we used a Pareto optimization to find the points with the best trade-off of both the uncertainty and diversity metrics from 10,000 randomly generated points each iteration. From the Pareto front of points that are non-dominated in those two objectives, we ordered them based on their uncertainty score and selected the median point. Note that sorting based on uncertainty is just the reverse order of a sort by diversity, so which objective you choose to sort by shouldn't have a significant impact. The only impact would be on cases where an even number of points are on the front so the point you select isn't the true median but rather one of the points near the median. When this occurs, we round down to select the median point, which would give a slight bias toward uncertainty. By selecting the median point we are attempting to choose a point that has a relatively good balance between the two objectives.

The results of this comparison are shown in Figures 11 and 12 with the results from each problem shown in the Appendix in Table VI. Again in Figure 11, we use uniform random sampling as the baseline (blue line) and include normally distributed random sampling for comparison. The results show that all three methods work better than the baseline and normally distributed random sampling. Using the uncertainty metric, differential entropy, works slightly better than using the distance metric, minimum distance. We also see that there is a benefit to combining both metrics using the Pareto optimization since we see an improvement in the upper quartile of performance. It is also interesting to note, as can be seen in Figure 12, that the diversity metric alone performed worse than uniform random sampling in 8 of the 35 cases, whereas the uncertainty approach and the Pareto approach only performed worse in 4 of the cases, demonstrating that the uncertainty and Pareto approaches offer more consistent improvements. This indicates that it is important to consider the current models to help guide the AL process. This makes sense since the goal is to select training points that will best inform the current model population, using only diversity doesn't consider the current state of models, so it is less likely that the training points selected will most inform those models. Statistical significance tests were also performed and the number of cases determined to be statistically significant are shown in the darker regions in the figure. The Mann-Whitney test was used to test for significance and a threshold of 0.05 was used. The Pareto approach was found to be statistically significant in 18 of the 20 cases where the Pareto approach outperformed. The p-values for the Pareto approach on each problem are shown in the Appendix in Table III.

Looking at the results, there are two instances where the Pareto approach performed considerably worse than the uncertainty and diversity approaches. Those are equations 9 and 71. Table VI in the Appendix shows that the combined method performs worse than focusing alone on either diversity or uncertainty for those two problems. This is likely a result of equations 9 and 71 being higher dimensional problems with 6 and 5 dimensions, respectively, so the 10,000 randomly generated points don't sufficiently fill the search space to find points with high values for both uncertainty and diversity.

Equation 71 was further explored to see if sampling additional points improved the performance when using the combined diversity uncertainty approach and to verify that sparse sampling was at least part of the issue as suspected. Equation 71 was retested using 100,000 randomly sampled points to search for the best trade-off between diversity and uncertainty. When using 100,000 points the median number of points required to solve the problem decreased to 42

This article has been accepted for publication in IEEE Transactions on Evolutionary Computation. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2024.3471341

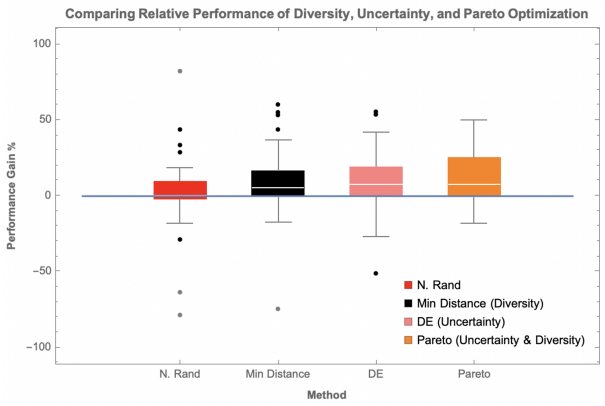

Fig. 11. **Comparing Relative Performance of Diversity, Uncertainty, and Pareto Optimization Using Uniform Random Selection as Baseline.** Shown here are the performance differences of AL diversity, uncertainty and Pareto methods compared to uniform random selection as the baseline (blue line) and normally distributed random selection (red distribution). We see that using the diversity metric, minimum distance (black distribution), performs consistently better than the baseline and the uncertainty metric, DE (pink distribution), performs a bit better than the diversity method. When using a Pareto optimization of both diversity and uncertainty we get even better performance. The distributions represent the median performances of 100 independent runs across all test problems. For completeness, there is a single point around -150 for the Pareto approach.

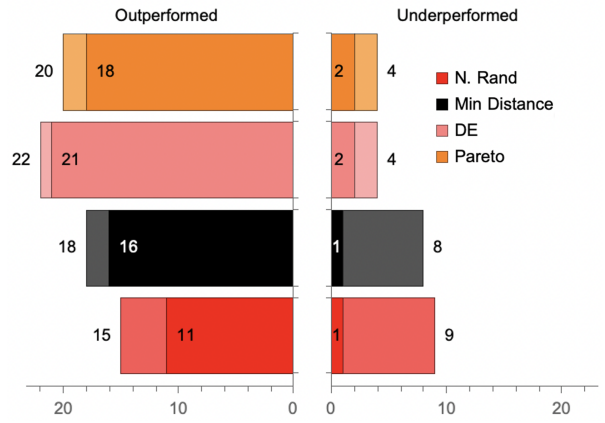

Fig. 12. **Comparing Performance of Diversity, Uncertainty, and Pareto Optimization Against Uniform Random Selection.** Each method is compared to uniform random sampling and the number of times that the method outperforms and underperforms is reported. The number of cases where the differences are statistically significant is shown in the darker regions. The number of times each method outperforms is shown on the left and the number of times each method underperforms is shown on the right. Outperforms means that a method used fewer points than uniform random sampling. Underperforms means it required more points. Ties are not counted but can be easily determined by taking the difference of 35 and the two values reported. The results show that DE, the uncertainty method works best. The Pareto approach ties for the least number of underperforming cases, matching DE, and outperforms between DE and Min. Distance. Statistical significance was determined using the threshold of 0.05 with the Mann-Whitney test.

points from 50.5, confirming that better sampling of the space improves the performance in this higher dimensional problem. The median performance of 42 points is still worse than either of the uncertainty or diversity approaches, so more points could be used, but increasing the number of points beyond 100,000 begins to make that search rather expensive. Rather than randomly sampling the points then selecting the Pareto front from those points, an alternative optimization method, such as NSGA II [25], could be used in future studies which might be cheaper and likely more effective.

D. Additional Benchmark Problems

To further test the Pareto AL approach, we selected two problems from a more recent benchmark set, SRBench [26]. One that is on the easier side for StackGP and one that is a bit more challenging. The easier problem selected was the van der Pol oscillator problem, referred to as "strogatz_vdp1" in SRBench. The equation for the van der Pol oscillator problem that we are trying to rediscover is $x' = 10 * (y - (1)/(3) * (x^3 - x))$. The more challenging problem was the bar magnet problem, referred to as "strogatz_barmag1" in SRBench and the equation for the bar magnet problem that we are trying to rediscover is $x' = 0.5 * sin(x - y) - sin(x)$. As with the previous problems, we performed each experiment 100 times and computed the median number of points to find the solution. The results of those experiments are shown in Table II. We can see that the Pareto approach performs significantly better than randomly sampling from a normal distribution and performs about 27.8% better than randomly sampling from a uniform distribution on the bar magnet problem. The performance gains over the normal and uniform distributed samplings are statistically significant considering a threshold of 0.05 using the Mann-Whitney test. We computed a p-value of $3.490*10^{-11}$ when comparing to the normal distribution and $6.481*10^{-6}$ when comparing to the uniform sampling. We also see better performance on the van der Pol oscillator, although since it was an easy problem there isn't as much opportunity for improvement, so we only see a reduction of a few points. The performance gains over the normal and uniform distributions are again statistically significant with a p-value of $2.51*10^{-7}$ when compared with the results from using normally distributed sampling and a p-value of $4.008*10^{-13}$ when compared with the results from using uniform random sampling.

TABLE II
SHOWN ARE THE MEDIAN NUMBERS OF POINTS NEEDED TO SOLVE EACH EQUATION. A TOTAL OF 100 INDEPENDENT TRIALS WERE PERFORMED FOR EACH EQUATION. WE COMPARE THE ACTIVE LEARNING METHOD THAT USES BOTH DIVERSITY AND UNCERTAINTY AND COMPARE THE PERFORMANCE AGAINST RANDOM SAMPLING ON TWO PROBLEMS FROM THE SRBENCH.

SRBench Problem	N. Ran Data Pts.	U. Ran Data Pts.	Pareto AL Data Pts.
Bar Magnet #1	51	18	13
Van der Pol Osc. #1	10	9	7

V. CONCLUSION

Both uncertainty and diversity metrics for active learning were explored to see how each metric impacts the success of active learning in genetic programming. As well, a Pareto approach was defined that allows both diversity and uncertainty to be considered for active learning. Of the uncertainty

approaches, it was observed that differential entropy performed best. It was also observed that relative uncertainty functions did not perform well. When using differential entropy it was found that performance could be boosted by using differential evolution as the optimizer over Scipy Optimize's minimize function. This indicates that the search space is not convex and requires a good optimizer to find solutions with high uncertainty.

When comparing the data diversity methods, it was found that correlation performed better than minimum Euclidean distance. Although correlation worked better, it does not work on cases with 2 dimensions or less. Thus, minimum Euclidean distance was selected for the Pareto approach. Future implementations may default to using minimum Euclidean distance for all cases with 1 or 2 dimensions and using correlation for higher dimensional problems. Mean distance was considered, but determined to be uninformative due to its frequency of identifying repeat points.

When comparing the Pareto approach which used both differential entropy and minimum Euclidean distance to differential entropy, minimum Euclidean distance, uniform random selection, and normally distributed random selection, it was found that differential entropy worked best, with the Pareto approach performing between differential entropy and minimum Euclidean distance. Looking at individual problems, there were a few cases where the Pareto approach actually worked better than both differential entropy and minimum Euclidean distance on their own, indicating potential benefits of combining the two approaches. For the cases where the Pareto approach did not work as well, it was identified that the multi-objective optimization strategy may have been at fault since it relies on randomly generating N points and selecting the median value in the Pareto front. Better methods such as NSGA-II could be explored in future studies to see if improved optimization methods leads to better active learning performance.

Overall, it was found that active learning can be efficiently utilized with genetic programming to reduce training data requirements. In practice, this would be useful to apply in scenarios where collecting data or labelling data is expensive, and model training is relatively cheap. In these scenarios, active learning could be used to guide data collection and labelling so that good models can be arrived at using as few data points as possible. This application has the potential to accelerate data-driven research, since it could lead to finding solutions with fewer resources in less time.

ACKNOWLEDGMENTS

Computer support by MSU's iCER high-performance computing center is gratefully acknowledged.

VI. APPENDIX

The statistical significance results for the Pareto approach across all of the Feynman problems used in the paper are shown in Table III. The median number of training points required to solve each problem across all uncertainty methods is shown in Table IV. The median number of training points required to solve each problem with the data diversity active learning approaches is shown in Table V. Table VI shows how the Pareto approach compares to using either just data diversity or model uncertainty.

TABLE III
STATISTICAL SIGNIFICANCE OF PARETO AL APPROACH VS. UNIFORM RANDOM SAMPLING. WE ARE USING A THRESHOLD OF 0.05 TO TEST FOR SIGNIFICANCE. THE MANN-WHITNEY TEST WAS USED TO TEST FOR SIGNIFICANCE.

EQ Num	p-value	Significant
2	$9.18*10^{-4}$	Yes
3	$1.40*10^{-11}$	Yes
4	$6.13*10^{-1}$	No
7	$9.61*10^{-6}$	Yes
9	$1.79*10^{-2}$	Yes
10	$3.40*10^{-5}$	Yes
13	$2.38*10^{-5}$	Yes
14	$7.46*10^{-4}$	Yes
23	$3.32*10^{-6}$	Yes
24	$8.65*10^{-5}$	Yes
27	$3.55*10^{-2}$	Yes
32	$1.30*10^{-3}$	Yes
35	$2.90*10^{-3}$	Yes
39	$7.62*10^{-}3$	Yes
41	$5.87*10^{-7}$	Yes
43	$3.38*10^{-3}$	Yes
47	$1.90*10^{-4}$	Yes
48	$1.78*10^{-5}$	Yes
52	$4.33*10^{-1}$	No
55	$1.98*10^{-4}$	Yes
57	$2.52*10^{-5}$	Yes
60	$6.84*10^{-3}$	Yes
61	$1.40*10^{-3}$	Yes
62	$2.69*10^{-20}$	Yes
63	$3.99*10^{-2}$	Yes
66	$1.56*10^{-3}$	Yes
67	$2.45*10^{-1}$	No
71	$3.92*10^{-1}$	No
83	$9.36*10^{-10}$	Yes
85	$1.06*10^{-12}$	Yes
89	$1.41*10^{-1}$	No
93	$1.55*10^{-5}$	Yes
95	$3.46*10^{-2}$	Yes
98	$2.98*10^{-3}$	Yes
99	$5.64*10^{-3}$	Yes
Significance Count	**30/35**	

REFERENCES

[1] D. A. Cohn, Z. Ghahramani, and M. I. Jordan, "Active learning with statistical models," *Journal of Artificial Intelligence Research*, vol. 4, no. 1, p. 129–145, mar 1996.

[2] B. Settles, "Active learning literature survey," University of Wisconsin–Madison, Computer Sciences Technical Report 1648, 2009.

[3] A. Nandy, C. Duan, C. Goffinet, and H. J. Kulik, "New strategies for direct methane-to-methanol conversion from active learning exploration of 16 million catalysts," *Journal of the American Chemical Society Au*, vol. 2, no. 5, pp. 1200–1213, 2022.

[4] Y. Li, X. Wang, Z. Shi, R. Zhang, J. Xue, and Z. Wang, "Boosting training for pdf malware classifier via active learning," *International Journal of Intelligent Systems*, vol. 37, no. 4, pp. 2803–2821, 2022.

[5] J. Kremer, K. Steenstrup Pedersen, and C. Igel, "Active learning with support vector machines," *WIREs Data Mining and Knowledge Discovery*, vol. 4, no. 4, pp. 313–326, 2014.

[6] P. Ren, Y. Xiao, X. Chang, P.-Y. Huang, Z. Li, B. B. Gupta, X. Chen, and X. Wang, "A survey of deep active learning," *ACM Comput. Surv.*, vol. 54, no. 9, oct 2021.

[7] N. Haut, B. Punch, and W. Banzhaf, "Active learning informs symbolic regression model development in genetic programming," in *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, ser. GECCO '23 Companion. New York, NY, USA: Association for Computing Machinery, 2023, p. 587–590. [Online]. Available: https://doi.org/10.1145/3583133.3590577

[8] I. M. El-Hasnony, O. M. Elzeki, A. Alshehri, and H. Salem, "Multi-label active learning-based machine learning model for heart disease prediction," *Sensors*, vol. 22, no. 3, 2022.

[9] J. De Freitas, G. L. Pappa, A. S. da Silva, M. A. Goncales, E. Moura, A. Veloso, A. H. Laender, and M. G. de Carvalho, "Active learning genetic programming for record deduplication," in *IEEE Congress on Evolutionary Computation*, 2010, pp. 1–8.

[10] C. W. Lasarczyk, P. Dittrich, and W. Banzhaf, "Dynamic subset selection based on a fitness case topology," *Evolutionary Computation*, vol. 12, no. 2, pp. 223 – 242, 2004.

[11] R. Curry, P. Lichodzijewski, and M. I. Heywood, "Scaling genetic programming to large datasets using hierarchical dynamic subset selection." *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 37, no. 4, pp. 1065–1073, 2007.

[12] A. Bartoli, A. De Lorenzo, E. Medvet, and F. Tarlao, "Active learning of regular expressions for entity extraction," *IEEE Transactions on Cybernetics*, vol. 48, no. 3, pp. 1067–1080, 2018.

[13] Y. Sverchkov and M. Craven, "A review of active learning approaches to experimental design for uncovering biological networks." *PLOS Computational Biology*, vol. 13, pp. 1–26, 6 2017.

[14] N. Eyke, W. Green, and K. Jensen, "Iterative experimental design based on active machine learning reduces the experimental burden associated with reaction screening." *Reaction Chemistry & Engineering.*, vol. 5, pp. 1963–1972, 2020.

[15] M. Kotanchek, G. Smits, and E. Vladislavleva, "Exploiting trustable models via pareto gp for targeted data collection." in *Genetic Programming Theory and Practice VI*, R. Riolo, T. Soule, and B. Worzel, Eds. Springer, 2009, pp. 145–162.

[16] M. Tegmark. Welcome to the Feynman Symbolic Regression Database! [Online]. Available: https://space.mit.edu/home/tegmark/aifeynman.html#:~:text=As\%20opposed\%20to\%20linear\%20regression,any\%20combination\%20of\%20mathematical\%20symbols.

[17] "scipy.optimize.minimize - scipy v1.11.1 manual." [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html

[18] "scipy.optimize.differential_evolution - scipy v1.11.1 manual." [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.differential_evolution.html

[19] S.-M. Udrescu and T. M., "A physics-inspired method for symbolic regression." *Science Advances*, vol. 6, p. eaay2631, 2020.

[20] N. Haut, W. Banzhaf, and B. Punch, "Active learning improves performance on symbolic regression tasks in stackgp," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, ser. GECCO '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 550–553.

[21] N. Haut. StackGP. [Online]. Available: https://github.com/hoolagans/StackGP

[22] L. Spector. PushGP. [Online]. Available: http://faculty.hampshire.edu/lspector/push.html

[23] N. Haut, W. Banzhaf, and B. Punch, "Correlation versus RMSE Loss Functions in Symbolic Regression Tasks," in *Genetic Programming Theory and Practice XIX*, W. Banzhaf, L. Trujillo, S. Winkler, and B. Worzel, Eds. Springer, 2022.

[24] M. Kotanchek, G. Smits, and E. Vladislavleva, "Pursuing the pareto paradigm: Tournaments, algorithm variations, and ordinal optimization." in *Genetic Programming Theory and Practice IV*, R. Riolo, T. Soule, and B. Worzel, Eds. Springer, 2007, pp. 167–185.

[25] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[26] W. La Cava, P. Orzechowski, B. Burlacu, F. O. de França, M. Virgolin, Y. Jin, M. Kommenda, and J. H. Moore, "Contemporary symbolic regression methods and their relative performance," 2021. [Online]. Available: https://arxiv.org/abs/2107.14351

TABLE IV

SHOWN ARE THE MEDIAN NUMBER OF POINTS NEEDED TO SOLVE EACH EQUATION. A TOTAL OF 100 INDEPENDENT TRIALS WERE PERFORMED FOR EACH EQUATION. THE LAST ROW INDICATES THE NUMBER OF CASES WHERE EACH OF THE ACTIVE LEARNING METHODS MATCHED OR PERFORMED BETTER THAN UNIFORM RANDOM SAMPLING OF TRAINING DATA. WHERE INFORMATIVE, THE MINIMUM NUMBER IS BOLDED.

EQ Num	U. Ran Data	N. Ran Data	Std/Mean Data	TrStd/TrMean Data	Std/TrMean Data	Std Data	DE Data	DE (DE) Data
2	54.5	97.5	50	39	47	53	82.5	47
3	> 1000	> 1000	876	692	741	> 1000	> 1000	724.5
4	30	21.5	21.5	20	23	20	28	**19.5**
7	88.5	82	23	**21**	22.5	39	52.5	35
9	120.5	155.5	150.5	73.5	359.5	100.5	153	160
10	6	6	6	11	6	7	6	6
13	13	14.5	15	15	14	14	**12**	**12**
14	30.5	33	28	24	31	23.5	24	22.5
23	8	8	7	8	7	8	7.5	7
24	49	58	39	29.5	31	26	**22**	28
27	30	17	20	13	19.5	18	14	15
32	17	16	20	18	21	16	18	**12**
35	19	17	17	**6**	21	18	13.5	12.5
39	10	10	10	12	11	10	9	9
41	7	7	7	8	7	8	7	7
43	453	82	876	218	202.5	144	326	192.5
47	13	12	14	12	13	13	12	12
48	15.5	14	18	17	17.5	14	13	**12.5**
52	9.5	9	10	10	9.5	10	9	9
55	10	10	11	12	10	11	10	**9**
57	30.5	31.5	25.5	27	24	**17**	23	21
60	7	7	7	7	7	7	7	7
61	18.5	20	20	19	18	18	**16**	17
62	34.5	56.5	37.5	34.5	33	34	30	**28.5**
63	14	13	15	16	15.5	14	13	13
66	11	9	15	14	14	15	10	**9**
67	10.5	11	11	10	10	10	11	11
71	51.5	47	34	58	38.5	31	**30**	35
83	5	5	5	5	5	5	5	5
85	4	4	4	4	4	4	4	4
89	5	5	4	5	4	4.5	5	5
93	8	8	8	8	8	7	8	8
95	11	**9**	12	11.5	11	12	10	11
98	8	7	9	9	9	9	7	7.5
99	30	**20**	31	35.5	35	25	24	21
Vs. U. Sampl.	-	-	**19**	**20**	**24**	**27**	**31**	**33**
Vs. N. Sampl.	-	-	**22**	**20**	**20**	**22**	**29**	**30**

TABLE V

SHOWN ARE THE MEDIAN NUMBER OF POINTS NEEDED TO SOLVE EACH EQUATION. A TOTAL OF 100 INDEPENDENT TRIALS WERE PERFORMED FOR EACH EQUATION. THERE ARE 2 EQUATIONS THAT HAVE A DASH INSTEAD OF A NUMBER AND THAT IS BECAUSE THEY HAVE ONLY TWO DIMENSIONS, SO SELECTING POINTS WITH MINIMAL CORRELATION TO THE REST OF THE TRAINING SET IS NOT POSSIBLE. THE APPROACH USING UNIFORMLY RANDOM DATA POINTS WAS INCLUDED IN THE FIRST COLUMN REPRESENTED AS A BASELINE. THE LAST ROW INDICATES THE NUMBER OF CASES WHERE EACH OF THE POINT DIVERSITY METHODS MATCHED OR PERFORMED BETTER THAN THE RANDOM APPROACH.

EQ Num	U. Rand Data Pts.	Pt. Dist Data Pts.	Pt. Corr Data Pts.
2	54.5	**44**	-
3	> 1000	> 1000	> 1000
4	30	**19**	29.5
7	88.5	35.5	60
9	120.5	210.5	**102.5**
10	6	6	**5**
13	13	**12**	14
14	30.5	24	**22**
23	8	**7**	8
24	49	**23**	26.5
27	30	13.5	**11.5**
32	17	15.5	**14**
35	19	**13.5**	15
39	10	11	**9**
41	7	8	**6**
43	453	533.5	**136.5**
47	**13**	14.5	16
48	15.5	**13**	**13**
52	9.5	10	**9**
55	10	10	10
57	30.5	**28**	29.5
60	7	7	7
61	18.5	**17.5**	**17.5**
62	34.5	**29.5**	36.5
63	14	14	**12**
66	11	12	**10**
67	**10.5**	11	15
71	51.5	**29**	30.5
83	5	5	5
85	4	4	-
89	5	5	5
93	8	7.5	**7**
95	11	10	**8**
98	8	8	**7**
99	30	25	**20.5**
Perf. Count	-	27/35	29/33

TABLE VI

SHOWN ARE THE MEDIAN NUMBER OF POINTS NEEDED TO SOLVE EACH EQUATION. A TOTAL OF 100 INDEPENDENT TRIALS WERE PERFORMED FOR EACH EQUATION. HERE THE TRADE-OFF BETWEEN DIVERSITY AND UNCERTAINTY IS EXPLORED. THE SECOND TO LAST ROW INDICATES THE NUMBER OF TIMES EACH APPROACH WAS THE WORST OF THE THREE APPROACHES. THE LAST ROW INDICATES THE NUMBER OF CASES WHERE EACH APPROACH WAS THE BEST OR TIED FOR THE BEST OF THE THREE APPROACHES. MINIMUM POINT DISTANCE WAS USED FOR THE DIVERSITY METRIC AND DIFFERENTIAL ENTROPY WAS USED AS THE UNCERTAINTY METRIC.

EQ Num	Pt. Dist Data Pts.	Pareto Data Pts.	Pt. Unc. Data Pts.
2	44	36.5	82.5
3	> 1000	501	> 1000
4	19	29	28
7	35.5	48.5	52.5
9	210.5	304	153
10	6	6	6
13	12	12	12
14	24	22	24
23	7	8	7.5
24	23	27	22
27	13.5	19	14
32	15.5	14	18
35	13.5	14	13.5
39	11	10	9
41	8	7	7
43	533.5	314.5	326
47	14.5	12	12
48	13	13	13
52	10	10	9
55	10	9	10
57	28	24	23
60	7	7	7
61	17.5	15	16
62	29.5	21.5	30
63	14	14	13
66	12	10	10
67	11	11	11
71	29	50.5	30
83	5	5	5
85	4	4	4
89	5	5	5
93	7.5	8	8
95	10	13	10
98	8	8	7
99	25	25.5	24
Worst Count	13	11	8
Best Count	16	19	21

Nathan Haut Nathan Haut (Member, IEEE) received the B.S. degree from Alma College, Alma, MI, USA in 2020, and the Ph.D. degree from Michigan State University, East Lansing, MI, USA in 2023.

He is currently a Fixed-Term Assistant Professor with the department of Computational Mathematics, Science, and Engineering, Michigan State University, East Lansing, MI, USA. His major research areas include evolutionary computation, genetic programming, and active learning in machine learning.

Wolfgang Banzhaf Wolfgang Banzhaf (Member, IEEE) received the Dr.rer.nat (Ph.D.) degree from the Department of Physics, Technische Hochschule Karlsruhe (now Karlsruhe Institute of Technology), in Germany.

After a stint in industry with Mitsubishi Electric Corporation in Japan and the US he was an Associate Professor of Applied Computer Science at the Technical University of Dortmund in Germany. In 2003 he became full professor of Computer Science at Memorial University of Newfoundland, St. John's, Canada, and served as head of department from 2003 to 2009 and from 2012 to 2016. In 2010 he was awarded a University Research professorship for leadership in research.He currently is the John R. Koza Chair of Genetic Programming with the Department of Computer Science and Engineering

and a member of the BEACON Center for the Study of Evolution in Action, Michigan State University, East Lansing, MI, USA. Beside many contributions in journals and conferences, he is a (co-)author of 3 books and co-editor of multiple volumes of conference and workshop proceedings. His research interests are in the field of bioinspired computing, notably evolutionary computation, and complex adaptive systems, both in aspects of theory and applications.

Bill Punch Bill Punch received the B.S. degree in 1979, the M.S. degree in 1984, and the Ph.D.degree in 1989 from The Ohio State University, Columbus, OH, USA.

Bill Punch is a retired Associate Professor from the Computational Math, Science and Engineering Department of Michigan State University, East Lansing, MI, USA. His research interests are evolutionary computation, high performance computing and computing pedagogy. His book with Rich Enbody titled "The Practice of Computing Using Python" is now in its third edition.