Learned Features vs. Classical ELA on Affine BBOB Functions

Moritz Seiler,Urban Škvorc,Gjorgjina Cenikj,Carola Doerr,Heike Trautmann

December 31, 2024

Abstract

Automated algorithm selection has proven to be effective to improve optimization performance by using machine learning to select the best-performing algorithm for the particular problem being solved. However, doing so requires the ability to describe the landscape of optimization problems using numerical features, which is a difficult task. In this work, we analyze the synergies and complementarity of recently proposed feature sets TransOpt and Deep ELA, which are based on deep-learning, and compare them to the commonly used classical ELA features. We analyze the correlation between the feature sets as well as how well one set can predict the other. We show that while the feature sets contain some shared information, each also contains important unique information. Further, we compare and benchmark the different feature sets for the task of automated algorithm selection on the recently proposed affine black-box optimization problems. We find that while classical ELA is the best-performing feature set by itself, using selected features from a combination of all three feature sets provides superior performance, and all three sets individually substantially outperform the single best solver.

Black-box Optimization Exploratory Landscape Analysis Automated Algorithm Selection Deep Learning

1 Introduction

It is well known that an optimization algorithm's performance depends heavily on the specific problem to be solved. Therefore, choosing the most suitable algorithm for a given problem is crucial to achieve good optimization results. Automating this task, known as *Automated Algorithm Selection* (AAS [28]), has long been of interest to the research community. A prerequisite of the AAS task is the representation of optimization problems in terms of numerical features which can be used as input to various machine learning (ML) models to predict the algorithms' performances. In recent years, the development of *Exploratory Landscape Analysis* (ELA [22]), a method of transforming samples of an optimization problem into informative *landscape feature*, has shown promising results for these tasks [1, 18]. However, ELA has shown certain flaws, such

as a large correlation between the individual features, costly computation times, a lack of robustness and expressiveness of some of the features [27], as well as a lack of generalizability to problem sets outside of the widely used *Black Box Optimization Benchmark* (BBOB [14]) [36, 19, 26].

Recently, novel problem sets were proposed on which these features can be further evaluated, e.g. for the AAS task [24, 37, 33, 11], as well as innovative methods for computing landscape features supposed to solve the inherent drawbacks of classical ELA features [29, 6, 31]. In this paper, we focus on two recently proposed approaches that utilize deep learning to learn landscape features automatically. These two approaches are *TransOpt* [6] and *Deep Exploratory Landscape Analysis* (Deep ELA [29]) which we compare with each other and also with classical ELA on a recently proposed set of Affine BBOB problems [38], which consists of linear combinations of the 24 widely used single objective optimization problems from the BBOB suite. Thereby, we compare classical ELA features and learned features on novel optimization problems that classical ELA was not specifically designed for. We split this analysis into two parts:

First, we compare the features themselves, by analyzing their correlation and by using ML to determine if one set of features is predicative of the other. The goal is to understand the complementarity of these features, and the amount of new information that is captured by them.

Second, we examine how these features can be used for AAS, and specifically whether combining all of these feature sets and conducting feature selection outperforms only using a single set.

2 Background

In the following, we briefly describe the differences between the three considered feature sets and provide an improvement to Deep-ELA which is the overall fourth considered feature set. In general, we consider two types of feature sets, classical (human-designed) (see Section 2.1) and learned feature sets (see Section 2.2).

2.1 Classical Exploratory Landscape Analysis

For *Automated Algorithm Selection* (AAS), it is essential to have a method for characterizing the optimization problem's landscape quantitatively, which machine learners can leverage. In current research, the extraction of ELA features is typically used. Pioneered by Mersmann et al. [22], ELA enables the automated extraction of low-level ELA features that directly relate to the high-level characteristics of a problem, like its (multi-)modality or its landscape's ruggedness. ELA features can be derived from a relatively small set of problem samples. Typically, a sample size of $50d$, or for enhanced stability, $250d$, is used. We use the *flacco* [17] `R`-library to compute classical ELA features as follows:

Basic features provide simple information about the sample set, such as the number of observation and their minimum and maximum boundaries [18].

Dispersion features compare the distribution of the full sample set to the distribution of a subset containing samples with the highest fitness scores [20].

y-Distribution features, such as skewness or kurtosis, indicate descriptive statistics of the fitness values' distribution [22].

Levelset features train discriminant analysis models to predict whether the fitness value of each sample falls above or below a specific quantile of all fitness values. The mean misclassification errors are used as features [18].

Meta-Model features are based on trained linear or quadratic models' performance metrics (e.g. R^2 score) [22].

Information Content features measure certain landscape characteristics, such as smoothness and ruggedness, based on statistic summarization gained from a series of random walks [23].

Nearest Better Clustering feature describes the relation between a set of nearest neighbors (based on their location in the decision space) and a set of nearest 'better' neighbors (based on their objective value) [16].

Principal Component Analysis features are derived from dimensionaltiy reduction by PCA [18].

In total, we obtain 93 features for each problem instance. Further, we excluded all cost-related features as these are meaningless for the characterization of optimization landscapes. In some rare cases, the computation of ELA features results in `NaN` or `Inf` values. Instead of removing all features that contain a single observation with either `NaN` or `Inf` values, we simply mean-imputed these values.

2.2 Learned Features

Of particular interest in the current research are feature-free approaches that utilize deep learning to create alternative problem representations (see e.g. [30, 29, 6, 31]). In the following, we particularly address the fundamental idea of *TransOpt features* [6] and *Deep Exploratory Landscape Analysis* (Deep ELA [29]).

2.2.1 TransOpt Features

(proposed by Cenikj et al. [6]) are obtained by training a transformer-based model on the supervised learning task of predicting the performances of twelve different configurations of *Particle Swarm Optimization* (PSO [15]) algorithms. The inputs to the transformer [35] are a raw set of candidate solutions and

their respective objective values. These samples are generated using *Latin Hypercube Sampling* (LHS [21]) and a sample size of $50d$, where d is the decision space's dimensionality. The model follows a transformer encoder architecture, producing representations of the samples which are then fed to a regression head, producing a numerical indicator of the performance of each of the algorithms. Further, the models are trained on a problem portfolio generated using the random function generator introduced in [34]. A separate model is trained for each problem dimensionality. 2 638 resp. 2 696 functions are used to train the $3d$ resp. $10d$ model. To generate representations of the affine problems, we remove the regression head (last layers of the TransOpt model) and simply make a forward pass of the samples of the affine problems through the trained transformer architecture to obtain their embeddings.

2.2.2 Deep Exploratory Landscape Analysis

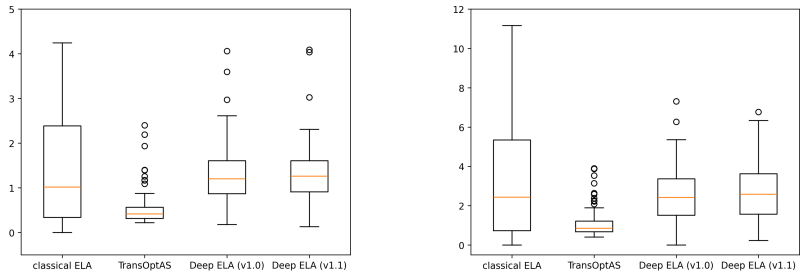
(Deep ELA) was proposed by Seiler et al. [29]. The authors designed also a transformer encoder that takes a set of candidate solutions as input and outputs a feature vector that (supposedly) uniquely describes the landscape. The models were trained on 250 000 000 randomly generated optimization problems — containing single- as well as multi-objective problems based on an approach very similar to the one proposed by van Stein [34]. The training routine was designed as a self-supervised learning task as outlined by Chen et al. [8]. The main advantage of this approach is that there is no need for manual or computationally expensive labeling.

In this work, we slightly updated the Deep-ELA approach. Although the predicted feature vector is guaranteed to be invariant to the order of the candidate solutions, the feature vectors may slightly fluctuate depending on permutations of the decision space, i.e. (x_1, x_2, x_3) will not necessarily yield the same output as (x_3, x_2, x_1). To account for this, we perform not only a single forward pass to compute the feature vectors but ten individual forward passes with ten different random permutations of the decision space to compensate for small fluctuations of the feature vectors. Afterwards, we take the arithmetic mean of the ten feature vectors. We will indicate the Deep-ELA variant as proposed by Seiler et al. [29] as *Deep-ELA (v1.0)* and our, updated variant of it as *Deep-ELA (v1.1)*.

3 Experimental Setup and Methodology

Black-Box Optimization Problems We reuse most of the data from previous related studies [5] to enable directly comparing results. We generate samples of the affine problem instances using LHS [21] in the range [-5,5] with a sample size of $50d$. To enable a fair feature comparison, we use the same set of samples to calculate the classical ELA, TransOpt, and Deep ELA features.

We create affine BBOB recombinations using the initial five instances from each of the 24 BBOB problem classes [38]. This process involved merging instances from varying classes that share the same instance identifiers. For ex-



(a) $3d$ (b) $10d$

Figure 1: Signal to Noise Ratio of the different feature sets. Metrics were created separately for $3d$ (a) and $10d$ (b) data.

ample, the first instance of the first problem class is combined with the first instances of the other 23 problem classes. We need to highlight here that instances with different instance identifiers from different problem classes are not combined to keep the total number of generated instances manageable. To be more specific, we used the following formula to combine two objectives:

$$F_{f_1,f_2}(X) = \exp\left(\alpha \cdot \log(f_1(X - \mathbf{x}_1^*) - \mathbf{y}_1^*) + (1-\alpha) \cdot \log(f_2(X - \mathbf{x}_2^*) - \mathbf{y}_2^*)\right).$$

Here, f_1, f_2 are the two optimization functions and $\mathbf{y}_1^*, \mathbf{y}_2^*$ their true optimal solutions, and $\mathbf{x}_1^*, \mathbf{x}_2^*$ the locations of the true solutions in the decision space. α is the recombination weight. The recombination is performed with α values of 0.25, 0.50, and 0.75 for all pairs of problem instances. This setup results in 8 280 generated problem instances; $24 \cdot 23$ possible recombinations, with three different α values, and five instances. The 24 different functions and their five instances were taken from the BBOB Suite. Further, we considered the 3-dimensional and the 10-dimensional case for all 8 280 functions, resulting in 16 560 instances in total.

Algorithm Performance and Performance Metric The algorithm portfolio contains *Differential Evolution* (DE [32]), *Genetic Algorithm* (GA [7]), *Particle Swarm Optimization* (PSO [15]), and *Evolutionary Strategy* (ES [2]), executed with their default configuration as specified in `pymoo` [3] (Version 0.6.0). All algorithms use LHS to construct the initial population. The population size is set to $50d$, where d is the decision space's dimensionality. The algorithms are executed on the affine problems in a fixed-budget scenario at budgets of 10, 30, and 50 iterations. For a $10d$ problem, a budget of 50 iterations is equivalent to a total of 5 000 function evaluations. Last, we perform ten executions of each algorithm on all problem instances.

To measure the algorithm performance, we consider a custom performance metric that was originally proposed by Cenikj et al. [5]. Most metrics used to measure algorithm selection performance are either time- or trial-based. While the former utilizes i.e. CPU run time or CPU flops, the latter utilizes i.e.

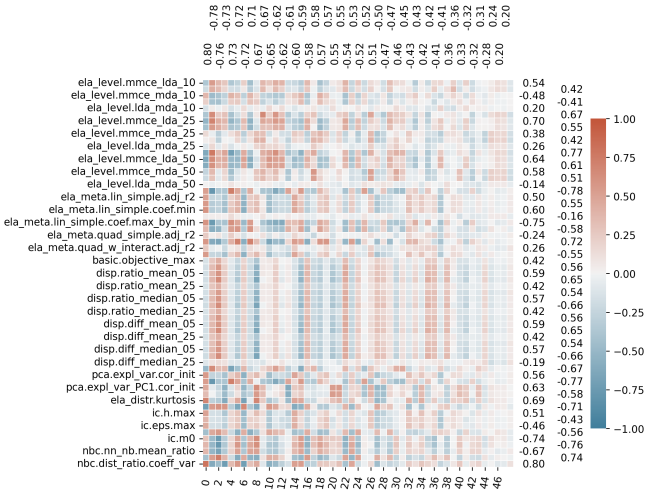


Figure 2: Spearman Correlation between Deep ELA (v1.1) on the x-axis and classical ELA on the y-axis for the $d = 3$ data. Values on the top (right) show the maximal absolute correlation per column (row).

the number of function evaluations. Examples are *Penalized Average Runtime 10* (PAR10) or *Expected Run Time* (ERT). The task of algorithm selection is then to select the algorithm that is expected to solve a given instance the quickest. Yet a major downside of these metrics is their treatment of unsuccessful runs. Not every algorithm is feasible to solve every given instance within a given time budget. Hence, the underlying metric is — de facto — bi-objective. Yet, as machine learners are most often trained on single-objective loss functions, failed and successful runs have to be factored into a single score in some way or another which is often unintuitive and may cause scores to be not fully commensurable to one another.

In our setup, we prioritize a quality-based measure. Instead of selecting the quickest solver, we select the solver that finds the best solution (in comparison to the other algorithms) within a given budget. Thus, we used a different performance metric which uses the best-found solution: the *Normalized Precision* (NP [5]). It scales the range between the best algorithm's best-found solution and the worst algorithm's best-found solution between zero and one. This makes comparisons between the algorithms straightforward as the VBS is always zero which is also the lower bound. Formally, the NP score can be defined as

$$NP_a = \frac{\hat{y}_a - \min \hat{\mathbf{y}}}{\max \hat{\mathbf{y}} - \min \hat{\mathbf{y}}}$$

where $\hat{\mathbf{y}} = \{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_r\}$ is a set of the r algorithms' best-found solutions and $\hat{y}_a$ is the best-found solution of algorithm a. For each of the 8 280 functions, we sampled ten initial candidate solutions with ten different seeds, giving us ten repetitions per instance. The size of this initial set is 150 ($3d$) and 500 ($10d$).

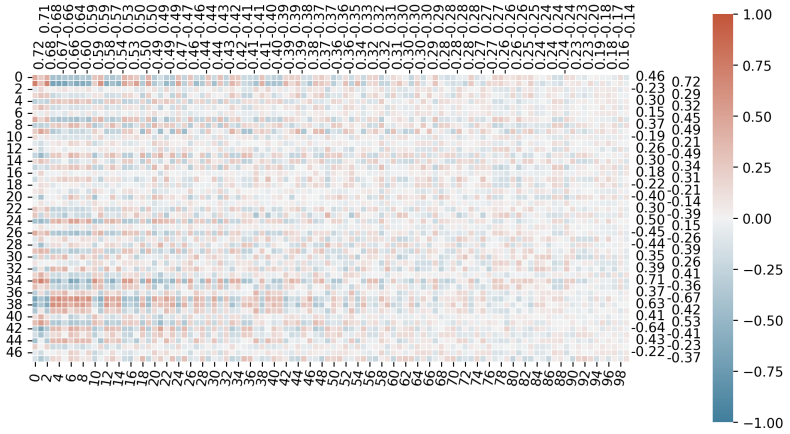


Figure 3: Spearman Correlation between TransOpt on the x-axis and Deep ELA (v1.1) on the y-axis for the $d = 3$ data. Values on the top (right) show the maximal absolute correlation per column (row).

These initial sets are evaluated for all 8 280 instances, for $3d$ as well as $10d$, separately. This guarantees equal samples for all feature sets. Afterwards, the four feature sets were created: (1) classical ELA, (2) TransOpt, (3) Deep ELA (v1.0), and (4) Deep ELA (v1.1). These feature sets are the foundation for the following two studies: (1) we conducted an unsupervised study to compare the four different feature sets to one another, and (2) we performed an automated algorithm selection study using the feature sets separately as well as in conjunction.

Comparison of the Feature Sets In the first part of our experiments, we perform two studies that assess the similarity of the feature sets used in this paper. In the first study, we examine the Spearman correlation between each pair of feature sets. In the second study, we train a *Support Vector Machine* (SVM [9]) for each feature pair that uses the first feature set to predict the features of the second set. The Spearman correlation is calculated using the library `scipy` [39] using default parameters. The SVM models are trained using the default settings of the `Python` library `scikit-learn` [25], but with a linear kernel, 10 000 maximal iterations, and five-fold cross-validation. Additionally, *Recursive Feature Elimination* [13] implemented by `scikit-learn` is used to determine which features from each feature set contribute the most. As SVMs are affected by different ranges, all features are scaled to a range of $(0, 1)$ before training.

Algorithm Selection Study Further, we considered two learners for the algorithm selector: *k Nearest Neighbor* (kNN [10]), and *Random Forest* (RF [4]). After some initial testing, we settled with $k = 15$ for the kNN and 250

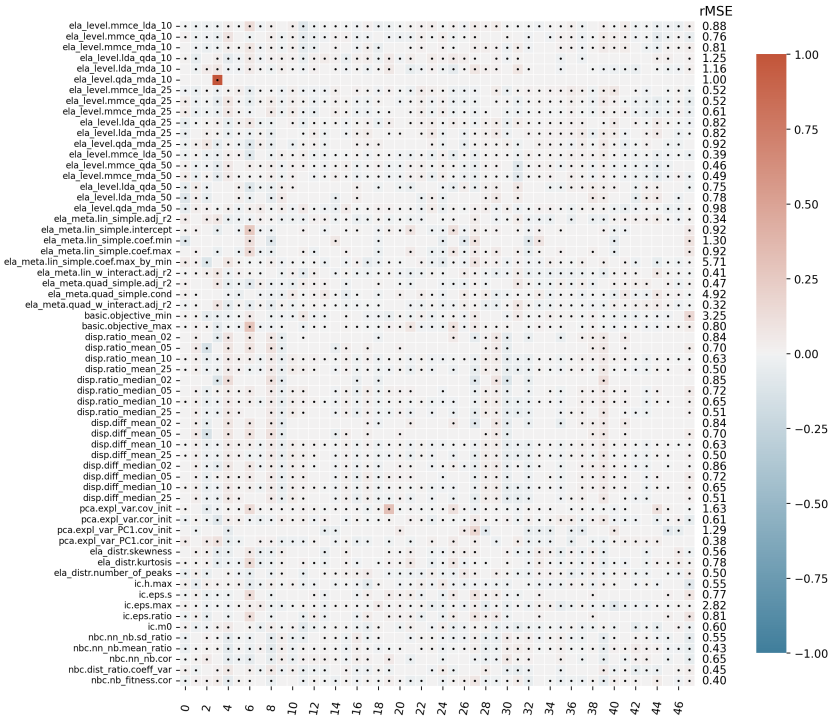


Figure 4: Detailed results of the feature selection experiments showing feature importance for each predicted feature, with Deep ELA (v1.1) features (x-axis) used to predict classical ELA features (y-axis)

trees for the random forest. Other than that, the learned were used with their default configuration as implemented in `scikit-learn`. Input features were min-max normalized for the kNN but left as they are for the random forest.

All selectors were trained on a multi-regression task, predicting the performance of each of the four algorithms. Afterward, the algorithm with the lowest predicted regret is selected. To compute the performance scores, we averaged the best-found solution over all ten repetitions per instance to account for stochasticity. We trained the two selectors on all four feature sets with ten-fold cross-validation and *sequential forward feature selection* (SFFS), whereby the cross-validation is the inner loop and the feature selection is the outer one. This way, all learners within an iteration contain the same set of features.

4 Results

Comparison of the Feature Sets First, we had a look at the Signal-to-Noise (StN) ratio between the different datasets (see Figure 1):

$$R_{\text{StN}} = \left(\frac{s^2_{\text{between}}}{s^2_{\text{within}}} \right)$$

where s_{between} is the observed standard deviation of a single feature across different functions while s_{within} is the observed standard deviation of a single feature across different instances of a single function. So generally speaking, the StN-ratio captures both the desired stability of a feature within instances of a function and its effectiveness in differentiating between functions, aiming for a balance that highlights features with both low within-function variance and high between-function variance.

Our findings are that classical ELA contains both features with the lowest and the highest StN-ratio. Hence, TransOpt as well as Deep ELA features contain fewer noisy features but also fewer highly descriptive features. Next, we found a small improvement between the default Deep ELA features (as proposed by Seiler et al. [29]) and our improved variant; indicating that the multiple-sampling strategy slightly improves the stability of the Deep-ELA features. On the other hand, it demonstrates that Deep ELA features are already very stable to dimensional augmentations. Last, we found that TransOpt features have the lowest average StN-ratio. This indicates that their stability across instances of the same function is low in comparison to instances across different functions. However, TransOpt features were not explicitly trained to remain stable across different instances of the same function, but are instead trained to have similar representations of problems where algorithms perform similarly.

Next, Figures 2 and 3 show the results of the correlation analysis. Precisely, Figure 2 shows the Spearman correlation between the Deep ELA features and the classical ELA features, with the numbers along the rows and columns representing the maximum absolute correlation measured in that specific row and column. To make it easier to examine the maximum correlation, we sort the columns of the table by the maximum column correlation. We only include the results of the experiments that were performed on $3d$ data, as the experiments on $10d$ achieved very similar results.

Figure 2 depicts the correlation between Deep ELA (v1.1) and classical ELA features. We can see that there is some correlation between the two feature sets and that about half of the Deep ELA features have an absolute correlation above 0.5 to at least a single classical ELA feature. However, the other half of the Deep ELA features exhibit a lower correlation. Similar results were observed with the correlation between the TransOpt and the classical ELA features. Further, Figure 3 shows the correlation between the Deep ELA and the TransOpt features. The results are somewhat similar to Figure 2 (classical vs. Deep ELA), with some features being heavily correlated. However, the majority of the features show a lower degree of correlation. This demonstrates that while the feature sets examined in this paper are not entirely distinct, all three still

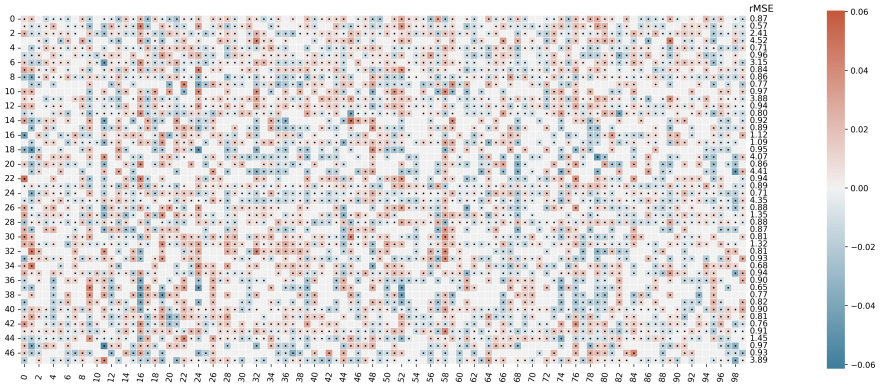
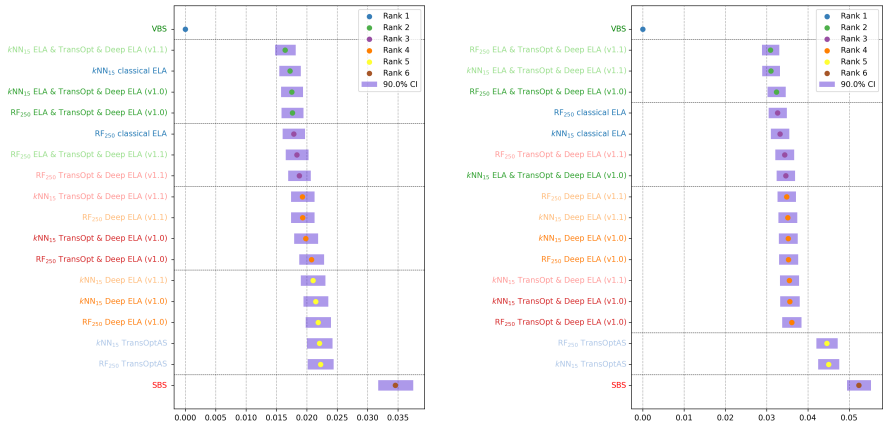


Figure 5: Detailed results of the feature selection experiments showing feature importance for each predicted feature, with TransOpt features (x-axis) used to predict Deep ELA (v1.1) features (y-axis). Note that to make them more readable, the values are capped to the region of $(-0.06, 0.06)$

contain features that are not heavily correlated with the others, which indicates complementarity.

As correlation can only show us pairwise similarity of features, we perform an additional study to further analyze the similarity of the feature sets. We train SVMs, each using one feature set to predict the features of a different set, with a separate model for each predicted feature. The evaluation metric used for each model is its root Mean Squared Error (rMSE) divided by the predicted feature's standard deviation. Due to this, a value of one indicates performance equivalent to a baseline model that always predicts the mean of the target feature while values below one indicate that the model outperforms the baseline. Hence, we can assume that if the value is below 1, the input feature set is likely to contain similar information. Table 1 shows the aggregated mean results of each feature set, with the rows representing the training set and the columns representing the testing set. We can observe that the best results are achieved when using one Deep ELA version to predict the other, which results in an rMSE of 0.47 or 0.59. This is an expected outcome given the similarity of the two feature sets and shows us what accuracy should be expected when comparing two heavily correlated feature sets. Other models perform worse, achieving normalized rMSE scores of around 0.8 to 0.9, but still below 1, except for TransOpt features. When examining the results in more detail, the worse performance of the TransOpt models can be explained by poor performance on a small number of outlier features. This further shows that all the feature sets contain at least some supplementary features.

Figure 4 and 5 contain more detailed results of the feature prediction experiments. Specifically, they show the feature importance assigned by recursive feature elimination for each predicted feature, as well as the achieved rMSE on each feature. Figure 4 shows these results when the Deep ELA (v1.1) features



(a) $3d$ (b) $10d$

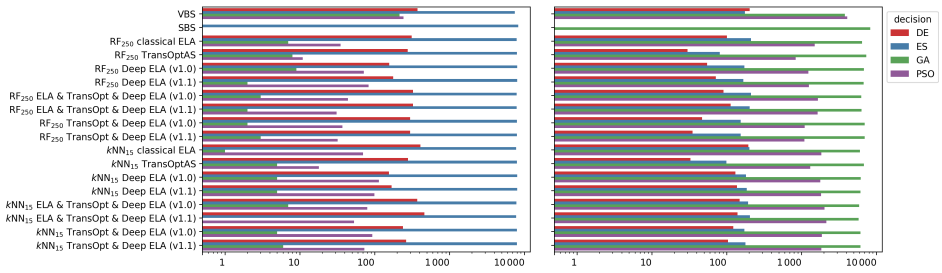
Figure 6: Results of the algorithm selection study. Selectors that share the same rank are stochastically tied to one another. We used the robust ranking technique as presented by Fawcett et al. [12] with $\alpha = 0.1$ and 100 000 repetitions. The median performance is shown on the x-axis.

are used to predict the classical ELA features. We can see that the feature importance is fairly well distributed with a couple of exceptions. Looking at the rMSE for each feature, we can also see that the Deep ELA features are relatively well-suited to predicting most classical ELA features. However, there are a couple of outliers where the model performs extremely poorly.

Figure 5 shows comparable results. The feature importance is evenly distributed, and while the overall rMSE is worse than in the previous figure, there are still noticeable outliers where the model performs worse than the baseline. After examining the data, most (but not all) of the outliers in this figure occur in Deep ELA features that contain very little variance. Since the performance is normalized by the standard deviation, this penalizes such features more harshly, which could explain the relatively poor performance. However, it is also worth noting that, in the algorithm selection study presented in the following chapter, these features are often selected as being highly informative, which substantiates the synergy of these Deep ELA features and the TransOpt features.

Table 1: Mean rMSE normalized by the standard deviation, with rows representing the training feature set and columns representing the testing feature set

	Classical ELA	Deep ELA (v1.0)	Deep ELA (v1.1)	TransOpt
Classical ELA	-	1.0	0.92	0.70
Deep ELA (v1.0)	0.93	-	0.47	0.86
Deep ELA (v1.1)	0.93	0.59	-	0.86
TransOpt	1.3	1.64	1.39	-



(a) $3d$ (b) $10d$

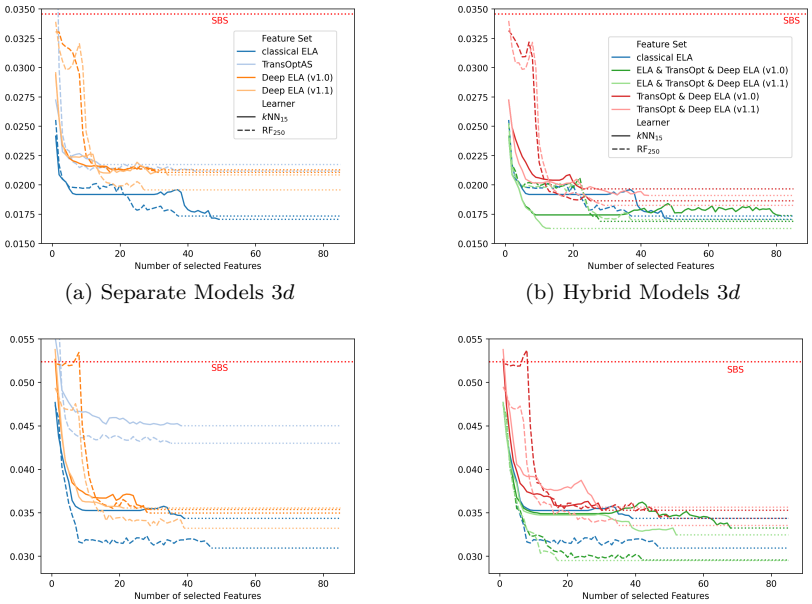
Figure 7: This figure depicts how often a certain algorithm of four total algorithms was selected by one of the algorithm selectors. Again, these metrics were derived separately for the $3d$-case and the $10d$-case.*Please note the log-scaling.*

Algorithm Selection Study As explained in the methodology of this paper, we choose to execute our algorithm selection study on two different data sets ($3d$ and $10d$). The results can be found in Figure 6. In both scenarios, all algorithm selectors provide significantly better performance than the SBS but are still significantly worse than the VBS. Generally speaking, we find that the hybrid selectors, making use of all three feature sets, significantly outperform the other selectors (see Figure 6 (a,b)). Contrary to the $10d$ case, classical ELA is stochastically tied to the hybrid selectors on the $3d$ dataset. Further, the combination of TransOpt and Deep ELA features significantly outperforms both feature sets on their own in the $3d$ scenario (see Figure 6 (a)).

Therefore, our findings are that classical ELA features still provide the most distinct features for algorithm selection, followed by Deep ELA and TransOpt features. Yet, both learned feature sets demonstrate great performance as all selectors outperform the SBS. Further, the best performances were shown by the hybrid models taking all feature sets into account. This indicates that both learned feature sets cover certain characteristics that classical ELA features miss.

Next, we analyzed which algorithm was selected by which selector and how often. The results can be found in Figure 7. The figure reveals that the VBS, as expected, selects from the algorithm portfolio, the most diversely. The SBS, by definition, only selects a single solver. All algorithm selectors select the SBS the most often — and, in particular, more often than the VBS does. Of the trained algorithm selectors, those trained on classical ELA are most diverse in selecting algorithms. This indicates again that classical ELA features still contain the most relevant information for algorithm selection. Still, both TransOpt and Deep ELA contain sufficient information to distinguish between the different algorithms.

Interestingly, all selectors provide more diverse selections in the $10d$ scenario in comparison to the $3d$. This may be because the algorithms GA and PSO are similar as often selected by the VBS. This is contrary to the $3d$ case where the



(a) Separate Models $3d$ (b) Hybrid Models $3d$

(c) Separate Models $10d$ (d) Hybrid Models $10d$

Figure 8: Performance versus number of selected features of all algorithm selectors during the sequential forward feature selection process. Each depicted line symbolizes a feature selection run and stops at the optimum. *Please note that here the mean performance is depicted.*

ES algorithm is selected by far the most often by the VBS. Hence, it may be easier for the selectors to utilize the algorithm portfolio if all algorithms in the portfolio are about equally as important.

Subsequently, we analyzed the number of selected features that were required to achieve the best performance. In Figure 8, we show the performance of each algorithm selector on the y-axis and the number of selected features during the forward pass at the x-axis. To better indicate the optimum, we stop at the optimal number of selected features. First of all, when comparing Figure 8 (a) and (b) as well as (c) and (d) to one another, it becomes apparent that the hybrid models achieve better performance with a lower number of features in comparison to the selectors that are trained solely on separate feature sets. This again demonstrates that every feature set contains highly descriptive features that are very important to the algorithm selection task.

Thereafter, we took a closer look at which features were selected exactly. The selected features for every random forest-based algorithm selector are depicted in Figure 9. Many of the selected learned features for the separate selectors are not included in the hybrid models. In fact, the best-performing selector for the $10d$ data, `ELA & TransOpt & Deep ELA (v1.1)`, only considers four TransOpt and a single Deep ELA (v1.1) feature but also twelve classical ELA features. On

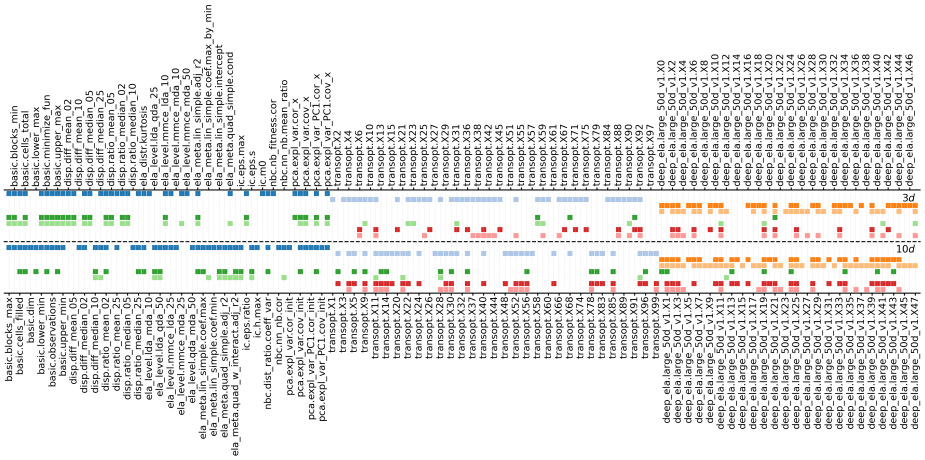


Figure 9: Selected features of each algorithm selector based on a random forest. Colors represent the same selectors as depicted in Figure 8.

the other hand, the stochastically tied performing selector, `ELA & TransOpt & Deep ELA (v1.0)`, considers more than twice the number of features. However, this might be just due to stochastic variations in the feature selection process as the latter selector also provides matching performance with a lower number of features (see Figure 8 (d)). Another reason for the low number of selected Deep ELA features may lie in the findings of our correlation study. Many ELA features are highly correlated to the Deep ELA features and may be removed due to describing similar characteristics. The most important classical ELA features of the hybrid selectors are `basic, dispersion, meta`, and for the $3d$ case also `PCA`-based features. This indicates that neither TransOpt nor Deep ELA covers similar characteristics.

5 Conclusion

We showed that all three feature sets, i.e., classical ELA and the two deep learning-based variants, contain relevant and unique information. This is why the hybrid algorithm selectors, utilizing all three sets, require the fewest number of features while providing the best algorithm selection performance. While the different feature sets contain to some degree shared information, both in terms of correlation and feature predictions, nevertheless, they also contain information that is unique to each feature set. Further, we could demonstrate that those Deep ELA features that were difficult to predict using TransOpt features are important add-ons to the TransOpt features within the AAS Study.

We also observed that classical ELA features are, generally speaking, slightly superior in comparison to learned features. Yet, learned features do provide promising performance. Their main advantage lies in the fact that no manual

crafting of new feature sets is required. Instead, a large foundation model learns all the relevant information by itself. This may provide easy scalability as with the adaption and tuning of the training routine, better foundation models can be trained and, thereby, may surpass the superiority of classical ELA features. Learned features may also prove particularly useful for optimization scenarios where human expertise for feature design is scarce. Rather than comparing hand-designed features with learned ones, as we have done in work, we expect to see future studies using learned features to design "human-readable" ones, for domains where hand-designed features are lacking.

5.0.1

The third author acknowledges support by the Slovenian Research Agency: research core funding No. P2-0098, young researcher grant No. PR-12393 to GC and project No. J2-4460.

5.0.2

The authors have no competing interests to declare that are relevant to the content of this article.

References

[1] Belkhir, N., Dréo, J., Savéant, P., Schoenauer, M.: Per instance algorithm configuration of CMA-ES with limited budget. In: Proceedings of the 19th Annual Conference on Genetic and Evolutionary Computation (GECCO). pp. 681–688. ACM (2017)

[2] Beyer, H., Schwefel, H.: Evolution strategies - a comprehensive introduction. Natural Computing **1**(1), 3–52 (March 2002). https://doi.org/10.1023/A:1015059928466

[3] Blank, J., Deb, K.: Pymoo: Multi-objective optimization in python. IEEE Access **8**, 89497–89509 (2020)

[4] Breiman, L.: Random Forests. Machine learning **45**, 5–32 (2001)

[5] Cenikj, G., Petelin, G., Eftimov, T.: A cross-benchmark examination of feature-based algorithm selector generalization in single-objective numerical optimization. Swarm and Evolutionary Computation **87**, 101534 (2024). https://doi.org/https://doi.org/10.1016/j.swevo.2024.101534, `https://www.sciencedirect.com/science/article/pii/S2210650224000725`

[6] Cenikj, G., Petelin, G., Eftimov, T.: Transoptas: Transformer-based algorithm selection for single-objective optimization (2024). https://doi.org/10.1145/3638530.3654191, in Press

[7] Chahar, V., Katoch, S., Chauhan, S.: A review on genetic algorithm: Past, present, and future. Multimedia Tools and Applications **80** (02 2021). https://doi.org/10.1007/s11042-020-10139-6

[8] Chen, X., Xie, S., He, K.: An empirical study of training self-supervised vision transformers (2021)

[9] Cortes, C., Vapnik, V.: Support-Vector Networks. Machine learning **20**, 273–297 (1995)

[10] Cover, T., Hart, P.: Nearest Neighbor Pattern Classification. IEEE transactions on information theory **13**(1), 21–27 (1967)

[11] Dietrich, K., Mersmann, O.: Increasing the diversity of benchmark function sets through affine recombination. In: Rudolph, G., Kononova, A.V., Aguirre, H., Kerschke, P., Ochoa, G., Tušar, T. (eds.) Parallel Problem Solving from Nature – PPSN XVII. pp. 590–602. Springer International Publishing, Cham (2022)

[12] Fawcett, C., Vallati, M., Hoos, H.H., Gerevini, A.E.: Competitions in AI – Robustly Ranking Solvers Using Statistical Resampling (Aug 2023)

[13] Guyon, I., Weston, J., Barnhill, S., Vapnik, V.: Gene selection for cancer classification using support vector machines. Machine learning **46**, 389–422 (2002)

[14] Hansen, N., Finck, S., Ros, R., Auger, A.: Real-Parameter Black-Box Optimization Benchmarking 2009: Noiseless Functions Definitions. Research Report RR-6829, INRIA (2009), `https://hal.inria.fr/inria-00362633`

[15] Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of ICNN'95 - International Conference on Neural Networks. vol. 4, pp. 1942–1948 vol.4 (1995). https://doi.org/10.1109/ICNN.1995.488968

[16] Kerschke, P., Preuss, M., Wessing, S., Trautmann, H.: Detecting funnel structures by means of exploratory landscape analysis. In: Proceedings of the 17th Annual Conference on Genetic and Evolutionary Computation (GECCO). pp. 265–272 (2015)

[17] Kerschke, P., Trautmann, H.: The R-Package FLACCO for Exploratory Landscape Analysis with Applications to Multi-Objective Optimization Problems. In: Proceedings of the IEEE Congress on Evolutionary Computation (CEC). pp. 5262 – 5269. IEEE (July 2016)

[18] Kerschke, P., Trautmann, H.: Comprehensive feature-based landscape analysis of continuous and constrained optimization problems using the r-package flacco. Applications in Statistical Computing: From Music Data Analysis to Industrial Quality Improvement pp. 93–123 (2019)

[19] Kostovska, A., Jankovic, A., Vermetten, D., de Nobel, J., Wang, H., Eftimov, T., Doerr, C.: Per-run algorithm selection with warm-starting using trajectory-based features. In: Parallel Problem Solving from Nature – PPSN XVII: 17th International Conference, PPSN, 2022. pp. 46–60. Springer (2022)

[20] Lunacek, M., Whitley, D.: The dispersion metric and the cma evolution strategy. In: Proceedings of the 8th annual conference on Genetic and evolutionary computation (GECCO). pp. 477–484 (2006)

[21] Menčík, J.: Latin hypercube sampling. In: Mencik, J. (ed.) Concise Reliability for Engineers, chap. 16. IntechOpen, Rijeka (2016). https://doi.org/10.5772/62370, `https://doi.org/10.5772/62370`

[22] Mersmann, O., Bischl, B., Trautmann, H., Preuss, M., Weihs, C., Rudolph, G.: Exploratory landscape analysis. In: Proceedings of the 13th annual conference on Genetic and evolutionary computation (GECCO). pp. 829–836 (2011)

[23] Muñoz, M.A., Kirley, M., Halgamuge, S.K.: Exploratory landscape analysis of continuous space optimization problems using information content. IEEE transactions on evolutionary computation **19**(1), 74–87 (2014)

[24] Muñoz, M.A., Smith-Miles, K.: Generating new space-filling test instances for continuous black-box optimization. Evolutionary Computation (ECJ) **28**(3), 379–404 (2020)

[25] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)

[26] Petelin, G., Cenikj, G.: How far out of distribution can we go with ela features and still be able to rank algorithms? In: 2023 IEEE Symposium Series on Computational Intelligence (SSCI). pp. 341–346 (2023)

[27] Renau, Q., Dréo, J., Doerr, C., Doerr, B.: Expressiveness and Robustness of Landscape Features. In: Proceedings of the 21st Annual Conference on Genetic and Evolutionary Computation (GECCO) Companion. pp. 2048 – 2051. ACM (2019)

[28] Rice, J.R.: The Algorithm Selection Problem. Advances in Computers **15** (1976)

[29] Seiler, M.V., Kerschke, P., Trautmann, H.: Deep-ELA: Deep exploratory landscape analysis with self-supervised pretrained transformers for single- and multi-objective continuous optimization problems. Evolutionary Computation Journal, under revision **arXiv preprint arXiv:2401.01192** (2024), `https://arxiv.org/abs/2401.01192`

[30] Seiler, M.V., Prager, R.P., Kerschke, P., Trautmann, H.: A collection of deep learning-based feature-free approaches for characterizing single-objective continuous fitness landscapes. In: Proceedings of the 24th Annual Conference on Genetic and Evolutionary Computation (GECCO) (2022)

[31] van Stein, B., Long, F.X., Frenzel, M., Krause, P., Gitterle, M., Bäck, T.: Doe2vec: Deep-learning based features for exploratory landscape analysis. In: Silva, S., Paquete, L. (eds.) Proceedings of the 25th Annual Conference on Genetic and Evolutionary Computation (GECCO) Companion. pp. 515–518. ACM (2023)

[32] Storn, R., Price, K.: Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization **11**, 341–359 (01 1997). https://doi.org/10.1023/A:1008202821328

[33] Tian, Y., Peng, S., Zhang, X., Rodemann, T., Tan, K.C., Jin, Y.: A recommender system for metaheuristic algorithms for continuous optimization based on deep recurrent neural networks. IEEE Transactions on Artificial Intelligence **1**(1), 5–18 (2020). https://doi.org/10.1109/TAI.2020.3022339

[34] Tian, Y., Peng, S., Zhang, X., Rodemann, T., Tan, K.C., Jin, Y.: A recommender system for metaheuristic algorithms for continuous optimization based on deep recurrent neural networks. IEEE transactions on artificial intelligence **1**(1), 5–18 (2020)

[35] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)

[36] Vermetten, D., Ye, F., Bäck, T., Doerr, C.: MA-BBOB: A problem generator for black-box optimization using affine combinations and shifts. CoRR **abs/2312.11083** (2023)

[37] Vermetten, D., Ye, F., Bäck, T., Doerr, C.: Ma-bbob: Many-affine combinations of BBOB functions for evaluating automl approaches in noiseless numerical black-box optimization contexts. arXiv preprint arXiv:2306.10627 (2023)

[38] Vermetten, D., Ye, F., Doerr, C.: Using affine combinations of BBOB problems for performance assessment. In: Proceedings of the Genetic and Evolutionary Computation Conference. p. 873–881. GECCO '23, Association for Computing Machinery, New York, NY, USA (2023). https://doi.org/10.1145/3583131.3590412, https://doi.org/10.1145/3583131.3590412

[39] Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S.J., Brett, M., Wilson, J., Millman, K.J., Mayorov, N., Nelson, A.R.J., Jones, E., Kern, R., Larson, E., Carey, C.J., Polat, İ., Feng,

Y., Moore, E.W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E.A., Harris, C.R., Archibald, A.M., Ribeiro, A.H., Pedregosa, F., van Mulbregt, P., SciPy 1.0 Contributors: SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods **17**, 261–272 (2020). https://doi.org/10.1038/s41592-019-0686-2