Neural network design: learning from Neural Architecture Search

Stein, N. van; Wang, H.; Bäck, T.H.W.

Neural Network Design: Learning from Neural Architecture Search

Bas van Stein
LIACS
Leiden University
Leiden, The Netherlands
b.van.stein@liacs.leidenuniv.nl

Hao Wang
LIP6
Sorbonne Université
Paris, France
hao.wang@lip6.fr

Thomas Bäck
LIACS
Leiden University
Leiden, The Netherlands
t.h.w.baeck@liacs.leidenuniv.nl

***Abstract*—Neural Architecture Search (NAS) aims to optimize deep neural networks' architecture for better accuracy or smaller computational cost and has recently gained more research interests. Despite various successful approaches proposed to solve the NAS task, the landscape of it, along with its properties, are rarely investigated. In this paper, we argue for the necessity of studying the landscape property thereof and propose to use the so-called Exploratory Landscape Analysis (ELA) techniques for this goal. Taking a broad set of designs of the deep convolutional network, we conduct extensive experimentation to obtain their performance. Based on our analysis of the experimental results, we observed high similarities between well-performing architecture designs, which is then used to significantly narrow the search space to improve the efficiency of any NAS algorithm. Moreover, we extract the ELA features over the NAS landscapes on three common image classification data sets, MNIST, Fashion, and CIFAR-10, which shows that the NAS landscape can be distinguished for those three data sets. Also, when comparing to the ELA features of the well-known Black-Box Optimization Benchmarking (BBOB) problem set, we found out that the NAS landscapes surprisingly form a new problem class on its own, which can be separated from all 24 BBOB problems. Given this interesting observation, we, therefore, state the importance of further investigation on selecting an efficient optimizer for the NAS landscape as well as the necessity of augmenting the current benchmark problem set.**

***Index Terms*—Neural Architecture Search; AutoML; Deep learning; Exploratory Landscape Analysis**

I. Introduction

Deep learning and deep neural networks (DNNs) are becoming more and more popular due to the way they can automatically perform feature extraction from raw data [23]. These deep neural networks have shown great performance in complex tasks such as image classification and audio analysis. Especially with the use of Convolutional Neural Networks (CNN) [14], on which we will focus during this study.

However, for these networks to perform well, good network architectures and network training methods (optimization methods) are required. These complex and deep architectures are till recently, mostly developed by hand. Recent advances in automatic machine learning (AutoML) also enabled the automatic development of these architectures, called Neural Architecture Search (NAS) [5], [25], [30].

978-1-7281-2547-3/20/$31.00

Using NAS techniques it is possible to build a well performing network in a matter of days without any knowledge of the problem and neural network architecture design. However, there are also quite a few draw-backs to the way NAS techniques are being used. First of all, it requires a lot of computational time and resources to perform NAS experiments [26]. Secondly, most of the results from the architecture search are not used and the algorithms do not learn anything that can be applied on multiple problem instances.

A commonly applied solution when there is not much data or a lot of computational power to develop new architectures, is to use an existing pre-trained architecture and fine tune this network on the data of a specific problem. This solution is also called transfer learning, where the knowledge gained by training on one data set is transferred to a new data set in the hope that the network can apply the same features to the new data set. Recently, in [18], Neural Architecture Transfer (NAT), was proposed where the authors apply transfer learning on architectures by using a so called supernet. The supernet can be incrementally modified and is used to select task specific subnets. While NAT already solves some of the computational issues regarding the development of neural architectures within a specific class of problems, it still cannot be used for solving completely different problem classes efficiently.

One of the main reasons why Deep Neural Networks and transfer learning is actually working, is that the optimization problem (fitting the network to the data) is very robust. There are many network architectures that can solve the same problem. We even dare to claim that there are an infinite number of neural network architectures that can solve a specific image classification problem with a decent accuracy. This makes it possible to find well working architectures by hand, or automatically using a small number of evaluations. It also shows us, that we can learn much more from these network architectures and probably design much smaller and more efficient networks.

In this paper we analyse the optimization landscapes of Neural Architecture Search for three well known image classification data sets. The aim of this research is to extract rules of thumb for designing neural architectures and to use the information to develop more efficient NAS or NAT algorithms.

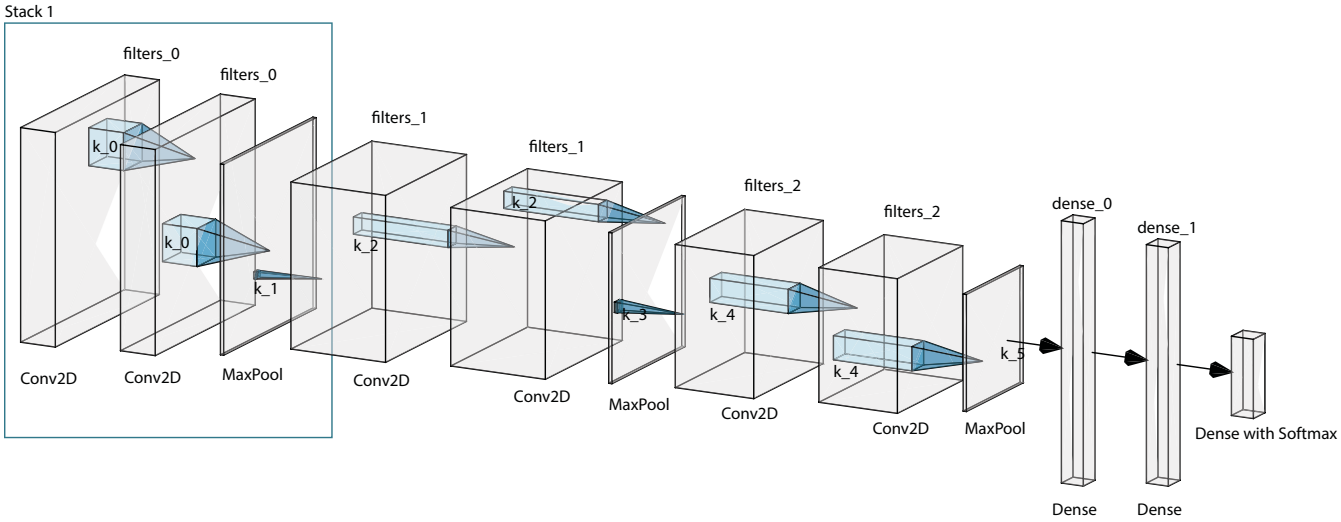


Fig. 1: The base architecture investigated in this paper: its feedforward structure starts with three stacks of layers, each of which comprises two convolutional layers and one max-pooling layer, and it ends with two layers of dense units and a softmax unit as the output. Note that, in each stack, the hyper-parameters such as the number of filters, strides, and the kernel size are independent from other stacks, and those two `Conv2D` layers always have the same number of filters.

II. Backgrounds and Problem Statement

In the literature, various approaches have been proposed to handle the NAS problem, e.g., random search [17], Bayesian optimization [8], evolutionary methods [28], and reinforcement learning [33]. Please see [6] for a comprehensive review of this topic. Despite the fruitful results in NAS, the landscape of NAS, along with its properties, are rarely investigated to the best of our knowledge. In this paper, we argue for the necessity of studying the landscape property of NAS, which potentially allows us to understand how hard such a search task practically is and where the difficulty lies in (e.g., the existence of discontinuity, saddle points, and non-convexity, etc.), and would pave the way towards the automated selection of optimizers for the NAS task.

A. Explorative Landscape Analysis

For a real-valued target function, $f\colon \mathcal{X} \to \mathbb{R}$, Explorative Landscape Analysis (ELA) [3], [20], [21] aims to characterize certain features of this function, using some sample points in $\mathcal{X}$ and the corresponding function values on f. For the so-called continuous optimization problem (i.e., $\mathcal{X} \subseteq \mathbb{R}^d$), a large set of landscape features has already been devised to capture the information on dispersion, convexity, level sets, curvature, and ruggedness, which is extensively used in the automated algorithm selection [11], to understand the search space [27], and for multi-objective optimization problems [10]. In this paper, we delve into the feature of performance indicators (e.g., testing accuracy) as the function of network architectures, by evaluating a huge number of randomly-generated network architectures and extracting the ELA features based on the performance of such architectures.

B. Research Questions

The problem that we are addressing in this paper is the time and cost inefficient approach of automatically designing and evolving well performing neural architectures. To improve upon the state-of-the-art approaches, such as NAT, and Meta-Neural Architecture Search [31], we have to gain additional knowledge of the underlying optimization landscape. *"What are the characteristics of the objective space?"*. Characteristics such as the number of local optima, how are those optima located and how easily is it to get trapped in such optima, as well as smoothness, ruggedness etc. *"Are the objective space landscape characteristics shared between instances?"*. If some of the characteristics are shared between several problem instances, or even all NAS problems, this can be used to narrow down the search space or develop more generic approaches.

Knowing the answers to these questions allows us to explore different optimization strategies and apply better fitted algorithms for architecture search and transfer. Additional information about this landscape for a specific set of problems, such as image classification, allows us to establish "rules of thumb" for a good starting neural architecture.

In this paper we focus mainly on three well-known image classification tasks, MNIST digits, Fashion and CIFAR-10. In the next section the empirical setup to analyse these NAS problem instances is explained in detail.

III. Experimental Setup

a) Design Space of Architectures: In this study, we narrowed down the design space to a feedforward convolutional network with stacks of convolutional layers, which is depicted in Fig. 1. Essentially, it entails three stacks of convolutional layers, followed by two layers of densely connected RELU

units and a SOFTMAX unit for the output. Each stack consists of two layers of convolutional filters (those two layers always have the same number of filters), one max-pooling layer, and one dropout layer to prevent over-fitting. We define the design space of architectures by varying the following hyper-parameters,

1) for each convolutional layer, the number of filters, the size and stride of each filter kernel, the L^2 kernel regularization coefficient, and the dropout rate,
2) for each densely connected layer, the number of units and the dropout rate, and
3) the learning rate and the L^2 regularization coefficient.

Each network is trained using the categorical crossentropy as loss function and using Stochastic Gradient Descent (SGD) with momentum as optimizer. The learning rate of SGD is one of the hyper-parameters as well.

We summarize those hyper-parameters and their sampling ranges in Table I, where large sampling ranges are taken, allowing for generating architectures that are either very "deep" or "shallow". Taking the naming convention given in this table, we express the design space as:

$$\mathcal{X} = \texttt{filters\_[0-2]} \times \texttt{k\_[0-5]} \times \cdots \times \texttt{l2},$$

which is a 23-*dimensional space consisting of integers and real values*. Moreover, we include two sets of different ranges for each parameter: The "Initial Range" is relatively wider, which is meant for exploring the NAS landscape with a good coverage and "Reduced Range" is narrowed down intentionally to focus on interesting regions of the design space, which is obtained by analysing the range of parameters of the top-50 architectures in the initial exploration (5% from 1000 in terms of the classification accuracy, please see below).

Hyper-parameter	Naming	Initial Range	Reduced Range	Nr.
#Filters	`filters_[0-2]`	$(10, 600]$	$(250, 400]$	3
Kernel size	`k_[0-5]`	$(1, 8]$	$(3, 7]$	6
Strides	`s_[0-2]`	$(1, 5]$	$(2, 5]$	3
#Dense nodes	`dense_size_[0-1]`	$(0, 2000]$	$(500, 1500]$	2
Dropout rate	`dropout_[0-6]`	(1e-5, 9e-1]	(1e-1, 4e-1]	7
Learning rate	`lr`	(1e-5, 1e-2]	(4e-3, 9e-3]	1
L^2 coeff.	`l2`	(1e-5, 1e-2]	(5e-4, 3e-3]	1

TABLE I: The design spaces of network architectures for the initial experimentation ("Initial Range") and the refined one ("Reduced Range"). Note that, for each layer, its hyper-parameter is independent from other layers and we denote by *Nr.* the number of hyper-parameters of each kind. Also, each architecture uses `softmax` for the last activation function and `relu` for the convolutional layers.

b) Sample Size and Sampling Strategy: Determining a proper sample size and the sampling strategy for the *Design of Experiments* (DoEs) is still a daunting task in ELA [9], [24]. We decided to use a relatively large sample size of 1000 (compared to the 23-dimensional design space), which would yield reliable feature values (i.e., with acceptable dispersion), despite the enormous computational cost it incurred. Moreover, in [24], it is shown that the landscape features are also susceptible to the sampling strategy and Latin Hypercube Sampling (LHS) [19] is more statistically robust among the conventional sampling/design methods. Hence, we have chosen LHS as the sampling strategy in this study. Throughout this paper, we shall use $f_{\text{acc}}, X \subseteq \mathcal{X}, y = f_{\text{acc}}(X)$ to denote the performance indicator, the design, and the performance value of network architectures, respectively.

c) Training Data Sets and Computation Details: Each configuration from the design of experiments is evaluated on three very popular image classification benchmarks, MNIST [16], FASHION [32] and CIFAR-10 [13][1]. We use a 20% validation set to calculate the accuracy as objective metric. The networks are trained using the SGD optimizer on batches of 100 samples using 50 epochs.

d) ELA feature computation: In the context of continuous optimization problems, more than 300 numerical features have been proposed, which are implemented in the `R`-package **flacco**[2] [10]. In this paper, we manually select 20 features out of 300 for characterizing the landscape of NAS, which settle in five categories. Given the design X of architectures and its performance values y, those selected features are defined as follows:

- The **dispersion** of all the design points in X and that of some top-ranked design points. In this study, we take the top-2% and 5% subset of X (in terms of f_{acc}).
  - `disp.diff_mean_[02|05]`: the difference between the dispersion of X and that of top-2% or 5% subset.
  - `disp.ratio_mean_[02|05]`: the ratio between the dispersions described above.

  It can be observed that the distances between the top 2 and 5 percent with respect to the distances between the complete DOE are close to each other (close to 1.0), but instances in the top 2 percent are slightly closer to each other.
- The y-**distribution** features, `distr.kurtosis` and `distr.skewness` compute the kurtosis and skewness of y, respectively.
- The **Information Content of Fitness Sequences** (ICoFiS) features [22] quantify, through a random walk in $\mathcal{X}$, the smoothness, ruggedness, and neutrality of the landscape. For instance, when taking a random walk $(x_1, x_2, \ldots)$ in X, we can evaluate the maximal entropy (`ic.h.max`) of consecutive fluctuations of the corresponding y-value, i.e., $y_{i+1} - y_i, i = 1, 2, \ldots$ (Note that such a difference is discretized. Please see [21] for details). All information content features are named as `ic.`$\star$ in the following discussions and we omit the description of those for brevity.
- The **meta-model** features take some important properties (e.g., adjusted R^2) of some simple meta-models trained on (X, y).

[1]The DOE, source code, and the experimental results are available at https://github.com/Basvanstein/LearningFromNAS [29].

[2]https://github.com/kerschke/flacco.

  - `lin_simple.adj_r2`: the adjusted R^2 of a simple linear model (w/o interactions between variables).
  - `lin_simple.intercept`: intercept of the linear model described above.
  - `lin_w_interact.adj_r2`: the adjusted R^2 of a simple linear model (w/ interactions between variables).
  - `quad_simple.adj_r2`: the adjusted R^2 of a quadratic model.
- The **nearest better clustering** features compare, for each $x \in X$, the distance from x to its nearest neighbor and that to their *nearest better neighbor*, that is the nearest neighbor having a better y-value than x. All nearest better clustering features are named according to `nbc.*` in the following discussions. Please see [12] for the detailed description of them.

IV. Visualizing the NAS Landscape

In the first experiment, we generated a huge DOE representing 1000 different network architectures, where each architecture is trained with 50 epochs and the same (relatively simple) architecture of three stacks with in each stack two convolutional layers (Figure 1).

In Figures 2a, 2b and 2c, *Multi Dimensional Scaling* (MDS) [15] is used to visualise the accuracy optimization landscape of the three problem instances. It can be observed, that CIFAR-10 is apparently a much harder problem than MNIST, since the mean accuracy of the landscape is much less. It can also be observed that the landscape visualisations are very similar in characteristics. This lead us to believe that the different NAS problems on image classification tasks can be solved using a general approach. This is, off course, also in line with transfer learning and neural architecture transfer. Using this design of experiments, we analyse the top 50 performing networks for each problem instance and look at the distribution of each search space parameter. In Figure 3, the parameters of the best 50 performing networks are visualised on the MNIST problem. The median setting is visualized by a line in each subplot.

To analyse which features are important for the performance of the neural architectures, the Pearson's correlation between each parameter and the CPU-time and accuracy are calculated and shown in Figure 6. The correlations of the features between each data set are almost identical. It is clear that the kernel size of the first layer and the dropout in the first layers are important features for the accuracy, and that there is a positive correlation between the number of filters and the CPU time (which is expected).

Using the distribution from the top 50 network architectures (top 5%), a new search space is defined as specified in Table I. The new search space is a fraction of the original search space, and much more limited, especially for the dropout in the first layers and the learning rate.

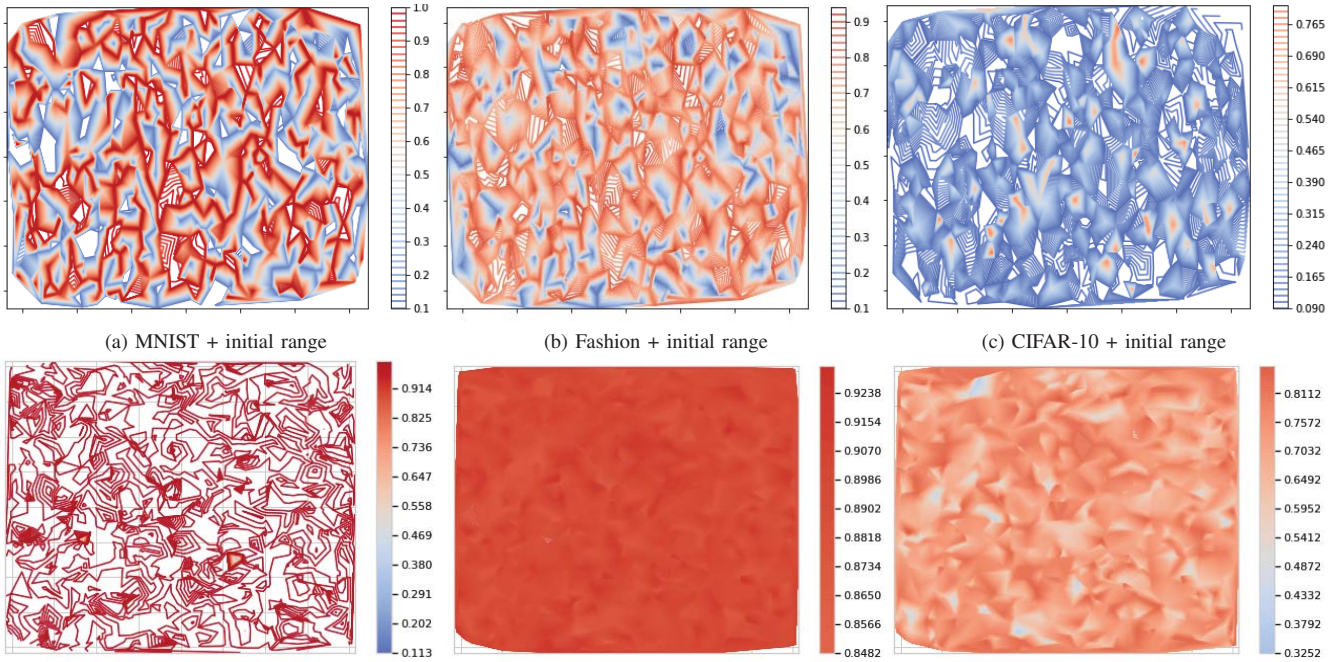
Using this "Reduced Range" search space, a new DOE of another 1000 samples is generated and evaluated. In Figure 2d, 2e and 2f, the MDS visualisations of the new optimization landscapes are shown. It can be observed that the new DOE has superior performance. If we take a look at the accuracy of all 1000 samples from the MNIST neural architecture search, only two of the samples did not perform well, all others are near 100% validation accuracy. We can observe that the architectures from this "Reduced Range" search space are all having a very high accuracy among all three problem instances. Neural Architecture Search might not even be required in this case. The landscape seems to be very rugged but also robust, showing only small performance changes across a wide set of different architectures. As mentioned in [26], gaining a small accuracy increase by performing an expensive NAS procedure is in most cases not worth the time and effort. By starting with a well-performing base architecture, this effort can be significantly reduced or even omitted.

V. Results on ELA features

The aforementioned 20 ELA features are computed on the 1000 design points sampled from the "Initial Range" (the features obtained on the "Reduced Range" are very similar) (see Table I). Also, to estimate the variability of feature values, we calculate those features with a bootstrapping procedure with the bootstrap size of 800 and 30 repetitions, the results of which are depicted as violin charts in Fig. 7. Prior to the feature computation, we re-scale the hyper-parameters to the range $[-5, 5]$ (this is to make the resulting feature comparable to those extracted on the BBOB problem set. See below). It is clear that most features entail a different statistical population on different training data sets, e.g., `distr.skewness` and `ic.eps.s`. For some features (e.g., `disp.ratio_mean_05`), although their absolute variation is marginal across different data sets, they contain enough power to distinguish the NAS landscape over the data sets. We found only `quad_simple.adj_r2` is not very informative according to this figure.

Based on the bootstrapped feature values, we subsequently delved into the clustering pattern that might exist when considering all three data sets together. As illustrated in Fig. 8a, we observed a crystal clear separation among the NAS landscapes on those three data sets, after performing a hierarchical clustering on all 30 bootstrapped feature vectors (Note that, only 10 feature vectors are rendered for each data set in order to make the plot more visually perceivable). In addition, the Euclidean distance is taken as the proximity metric between clusters and it is also obvious that the MNIST and CIFAR-10 data sets are more similar, compared to Fashion.

Naturally, we are also curious about whether the NAS landscape would resemble that of some conventional benchmark problem in terms of feature vectors, since if this is the case, then an optimizer that performs extremely well on such a benchmark problem will also be competitive on the NAS task. For this purpose, we choose the well-known **Black-Box Optimization Benchmarking** (BBOB) [4] problem set, which encompasses 24 continuous problems of different kinds (e.g., (non-)separable, multi-modal, and highly rugged), and



(a) MNIST + initial range (b) Fashion + initial range (c) CIFAR-10 + initial range

(d) MNIST + reduced range (e) Fashion + reduced range (f) CIFAR-10 + reduced range

Fig. 2: Rendered by Multi Dimensional Scaling (MDS), the $2D$ landscape of classification accuracy as a function of network architectures, where the architectures are created using the initial range (top row) and the reduce range (bottom row) and the accuracy value is indicated by the color scale.

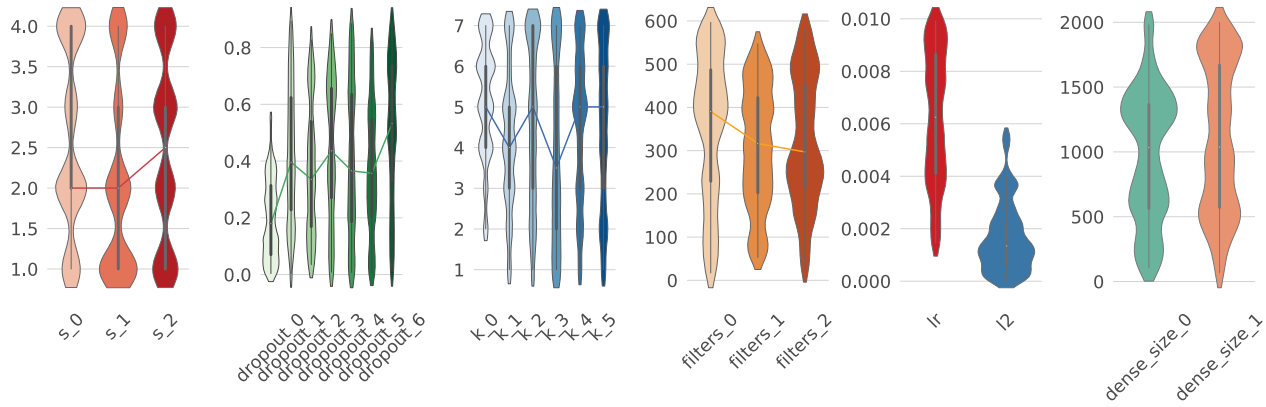


Fig. 3: On MNIST data set, the estimated distribution of hyper-parameters from the top-50 architectures. The distribution (violin) is obtained from kernel density estimation.

compute the same set of ELA features on all 24 problems with 1000 design points obtained from the LHS method. To make the results comparable, we set the dimensionality of BBOB problems to 23 and apply the LHS method in the hyper-box $[-5, 5]^{23}$ (since it is the range on which those problems are defined). The ELA features for our NAS instances are compared to 20 instances of all 24 functions from the BBOB benchmark [4]. In detail, we extracted the ELA features on the first 20 problem instances (by which the original problem is randomly rotated and translated) of each BBOB problem, for improving the robustness of the resulting feature values.

We performed the same hierarchical clustering procedure again for both the NAS and BBOB landscape features, on the mean feature values over 30 bootstraps (NAS) and 20 problem instances (BBOB). The resulting clusters are shown in Fig. 8b, in which we can see that the NAS landscape on three data sets falls into a category on its own and the other BBOB problem clusters are considerably distant to it (supported by the proximity metric on the y-axis).

When looking at all problem instances without any aggre-

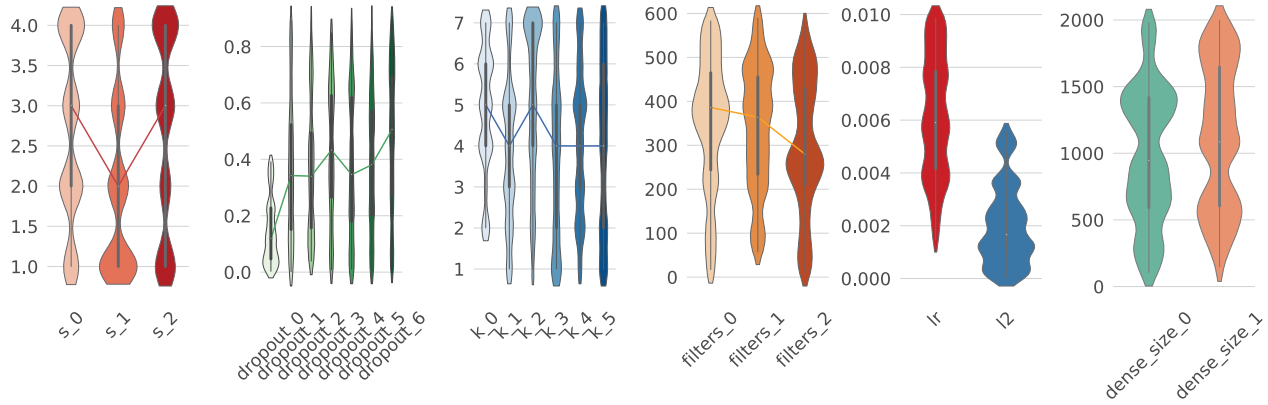


Fig. 4: On Fashion data set, the estimated distribution of hyper-parameters from the top-50 architectures. The distribution (violin) is obtained from kernel density estimation.

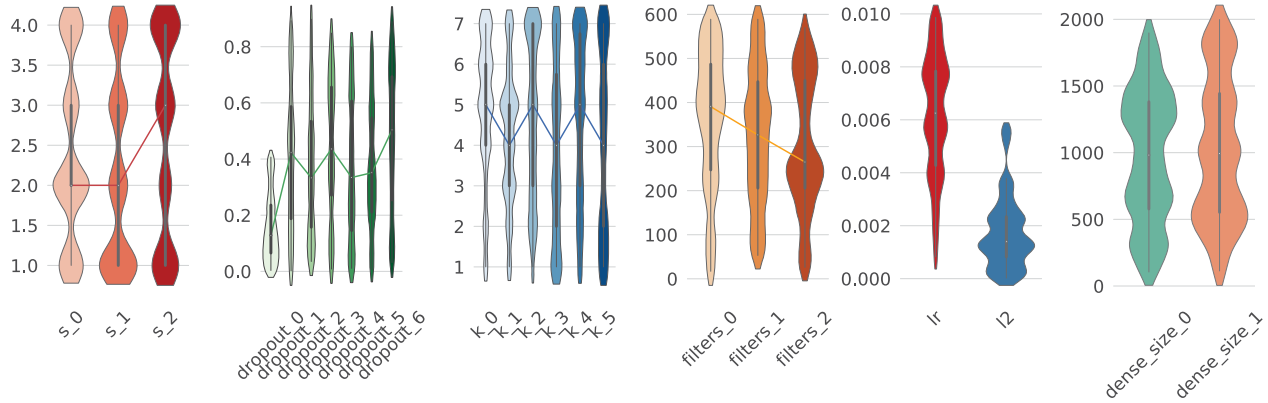


Fig. 5: On CIFAR-10 data set, the estimated distribution of hyper-parameters from the top-50 architectures. The distribution (violin) is obtained from kernel density estimation.

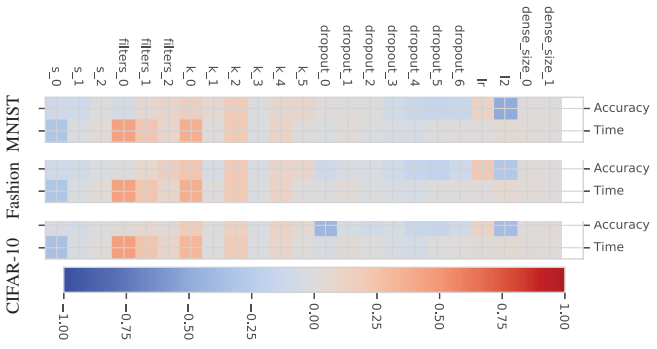


Fig. 6: Pearson's correlation coefficient between the accuracy and CPU time calculated on the architecture sampled in the initial range for data sets MNIST, Fashion, and CIFAR-10, respectively (top to bottom).

gation, we observed that the 20 closest neighbours to MNIST have an Euclidean distance of 3.24 ± 0.03 while the average distance between the neighbours and their own 20 neighbours is 0.71 ± 0.17. The similar pattern is also seen for CIFAR-10 (3.24 ± 0.04 versus 0.72 ± 0.18) and Fashion (2.71 ± 0.04 versus 0.72 ± 0.20). This seconds our previous findings from the hierarchical clustering that the NAS problems are not overlapping with any function group of the BBOB functions, with respect to the ELA features.

Among all BBOB problems, function $f12$, $f11$, and $f13$ are the nearest neighbours to all three of the NAS landscapes. On one hand, $f12$ is the *Bent Cigar* function[3], which has a very narrow ridge that needs to be followed to optimize the function. On the other hand, $f11$ and $f13$ are classified as "ill-conditioned" functions, suggesting that we would opt to apply, for the NAS task, optimizers devised particularly to handle the ill-conditioning scenario. According to the results of BBOB 2019[4], this preference entails algorithms such as *GLOBAL* (Sampling, clustering, and local search

[3]See https://coco.gforge.inria.fr/

[4]GECCO Workshop on Real-Parameter Black-Box Optimization Benchmarking

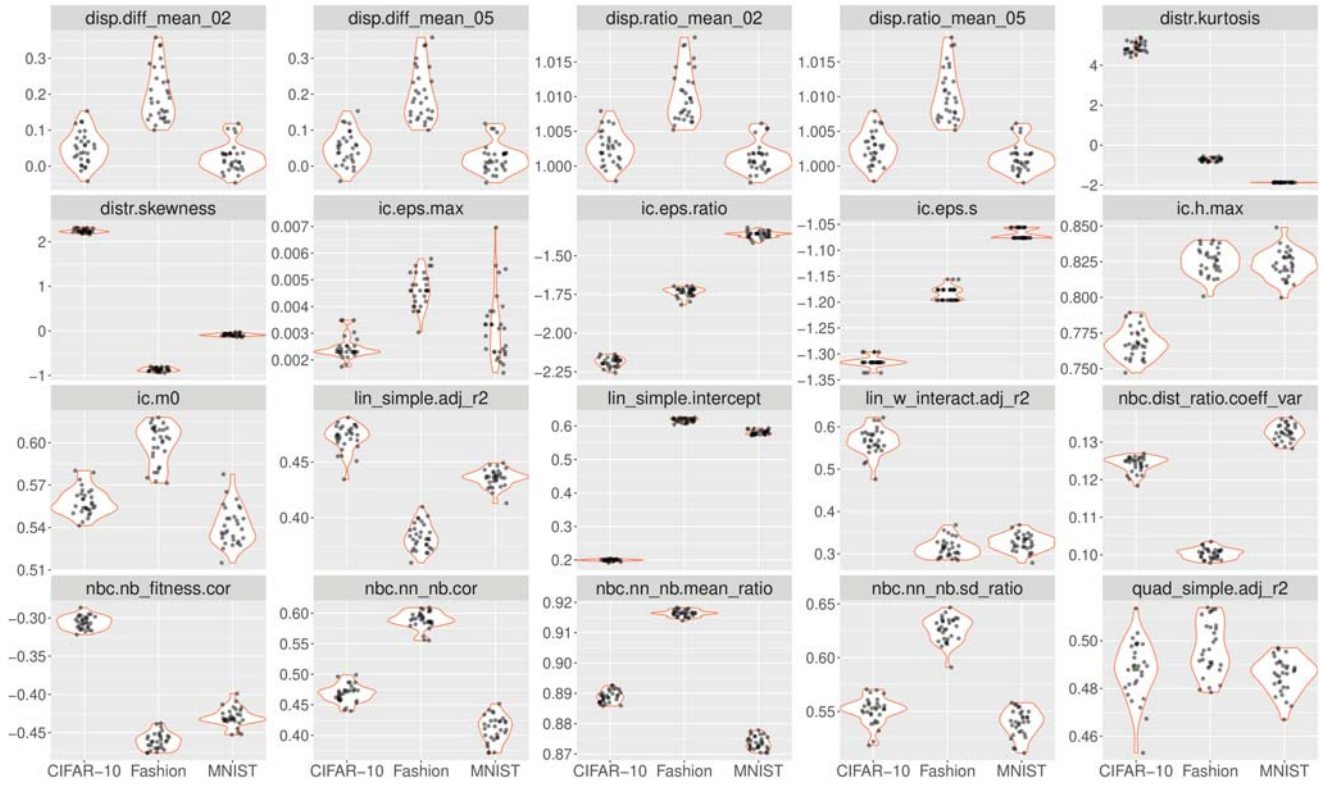
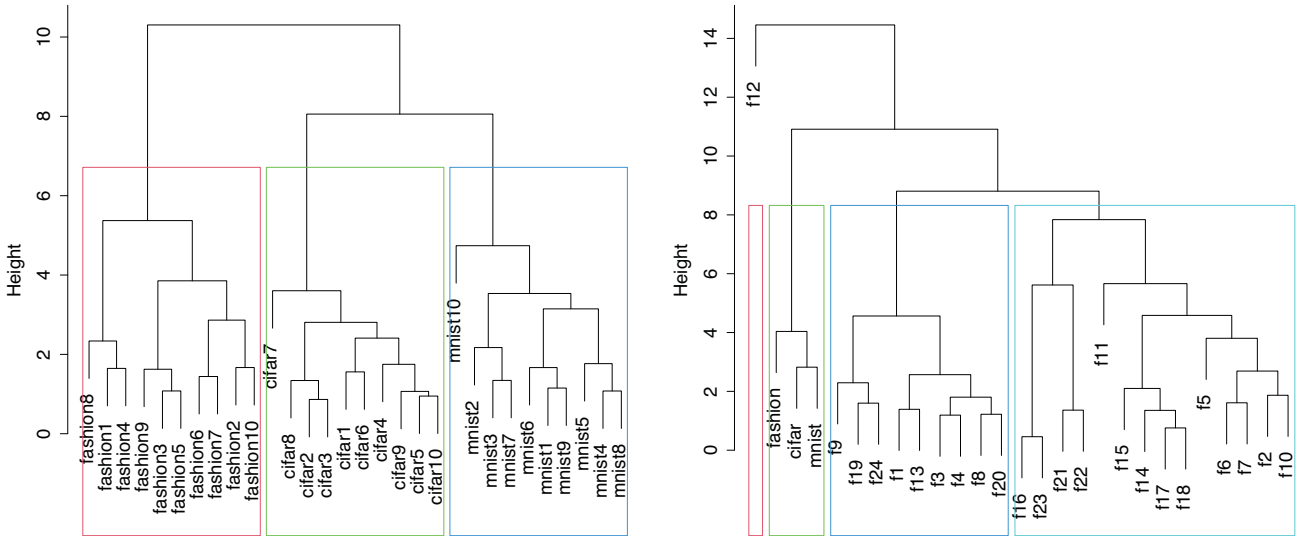


Fig. 7: On MNIST, CIFAR-10, and Fashion data sets, the distribution of the bootstrapped ELA features of the NAS landscape, which is computed on 1000 architecture DoEs with the bootstrap size of 800 and 30 repetitions.



(a) Bootstrapped ELA features of NAS landcapes

(b) ELA features of NAS + BBOB

Fig. 8: The clustering patterns given by the hierarchical clustering method with a proximity metric that takes the maximal Euclidean distance between two clusters. The "Height" on the y-axis indicates the proximity value for each pair of merged clusters. We additionally marked the major clusters using boxes.

using BFGS or Nelder-Mead) [2], *iAMALGAM* (Adapted Maximum-Likelihood Gaussian Model Iterated Density Estimation Algorithm) [1], and *BIPOP-CMA-ES* (Bi-population Covariance Matrix Adaptation Evolution Strategy with random restarts) [7].

VI. Conclusions and outlook

In this paper, we use the Exploratory Landscape Analysis (ELA) technique to probe into the landscape properties of the Neural Architecture Search (NAS) task. We designed a feedforward convolutional architecture with stacked structure for our purpose. In this way, the design/search space is narrowed down drastically for high accuracy solutions. Using the knowledge, of how the underlying optimization landscape behaves, better fitted algorithms can be used for NAS and a better starting point architecture can be developed from scratch. We have shown that for the three common image classification problems, MNIST, Fashion, and CIFAR-10, the NAS landscapes share many properties and the same basis neural architecture can be used to solve all three problem instances efficiently on a highly reduced search space without even requiring Neural Architecture Search. We have reduced the "Initial Range" of the search space drastically to a fraction of the search space by comparing the distribution of the parameters of the best performing networks among all three NAS tasks. The network architectures resulting from the "Reduced Range" search space showed supreme performance for all three image classification problems.

The ELA features can be used to compare NAS problems with continuous black-box optimization problems in order, for instance, to select a proper optimizer. We showed that the ELA features of the NAS landscape are distinguishable across those three data sets, and are also perfectly separated from the ELA features of all 24 BBOB problems, indicating that the NAS task is an entirely new problem category on its own when considering the dominant benchmark problem sets in the field of continuous black-box optimization. Hence, it would be beneficial, for algorithm selection and development, to study which optimizer of the state-of-the-art would perform well on the NAS task, and for the benchmark community, to augment the current problem sets for incorporating the NAS-like landscape.

In this paper the search space was limited to only real-valued and integer parameters, while categorical variables like the activation function were fixed because ELA features can only be calculated on continuous spaces. However, there are methods to also calculate features for categorical spaces that could be interesting to apply in a similar fashion.

Acknowledgments

Our work was supported by the Paris Ile-de-France Region.

References

[1] Bosman, P.A., Grahl, J., Thierens, D.: Benchmarking parameter-free amalgam on functions with and without noise. Evolutionary computation **21**(3), 445–469 (2013)

[2] Csendes, T.: Nonlinear parameter estimation by global optimization-efficiency and reliability. Acta Cybernetica **8**(4), 361–370 (1988)

[3] Derbel, B., Liefooghe, A., Vérel, S., Aguirre, H.E., Tanaka, K.: New features for continuous exploratory landscape analysis based on the SOO tree. In: Friedrich, T., Doerr, C., Arnold, D.V. (eds.) Proceedings of the 15th ACM/SIGEVO Conference on Foundations of Genetic Algorithms, FOGA 2019, Potsdam, Germany, August 27-29, 2019. pp. 72–86. ACM (2019). https://doi.org/10.1145/3299904.3340308, https://doi.org/10.1145/3299904.3340308

[4] Elhara, O., Varelas, K., Nguyen, D., Tusar, T., Brockhoff, D., Hansen, N., Auger, A.: Coco: The large scale black-box optimization benchmarking (bbob-largescale) test suite (2019)

[5] Elsken, T., Metzen, J.H., Hutter, F.: Neural Architecture Search. Journal of Machine Learning Research **20**, 1—-21 (2019), http://arxiv.org/abs/1905.01392

[6] Elsken, T., Metzen, J.H., Hutter, F.: Neural architecture search: A survey. J. Mach. Learn. Res. **20**, 55:1–55:21 (2019), http://jmlr.org/papers/v20/18-598.html

[7] Hansen, N.: Benchmarking a bi-population cma-es on the bbob-2009 function testbed. In: Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers. pp. 2389–2396 (2009)

[8] Kandasamy, K., Neiswanger, W., Schneider, J., Poczos, B., Xing, E.P.: Neural architecture search with bayesian optimisation and optimal transport. In: Advances in neural information processing systems. pp. 2016–2025 (2018)

[9] Kerschke, P., Preuss, M., Wessing, S., Trautmann, H.: Low-budget exploratory landscape analysis on multiple peaks models. In: Friedrich, T., Neumann, F., Sutton, A.M. (eds.) Proceedings of the 2016 on Genetic and Evolutionary Computation Conference, Denver, CO, USA, July 20 - 24, 2016. pp. 229–236. ACM (2016). https://doi.org/10.1145/2908812.2908845, https://doi.org/10.1145/2908812.2908845

[10] Kerschke, P., Trautmann, H.: The r-package FLACCO for exploratory landscape analysis with applications to multi-objective optimization problems. In: IEEE Congress on Evolutionary Computation, CEC 2016, Vancouver, BC, Canada, July 24-29, 2016. pp. 5262–5269. IEEE (2016). https://doi.org/10.1109/CEC.2016.7748359, https://doi.org/10.1109/CEC.2016.7748359

[11] Kerschke, P., Trautmann, H.: Automated algorithm selection on continuous black-box problems by combining exploratory landscape analysis and machine learning. Evol. Comput. **27**(1), 99–127 (2019). https://doi.org/10.1162/evco_a_00236, https://doi.org/10.1162/evco_a_00236

[12] Kerschke, P., Trautmann, H.: Comprehensive feature-based landscape analysis of continuous and constrained optimization problems using the r-package flacco. In: Bauer, N., Ickstadt, K., Lübke, K., Szepannek, G., Trautmann, H., Vichi, M. (eds.) Applications in Statistical Computing – From Music Data Analysis to Industrial Quality Improvement, pp. 93 – 123. Studies in Classification, Data Analysis, and Knowledge Organization, Springer (2019). https://doi.org/10.1007/978-3-030-25147-5_7, https://link.springer.com/chapter/10.1007/978-3-030-25147-5_7

[13] Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)

[14] Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. pp. 1097–1105 (2012)

[15] Kruskal, J.B.: Multidimensional scaling. No. 11, Sage (1978)

[16] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)

[17] Li, L., Talwalkar, A.: Random search and reproducibility for neural architecture search. In: Uncertainty in Artificial Intelligence. pp. 367–377. PMLR (2020)

[18] Lu, Z., Sreekumar, G., Goodman, E., Banzhaf, W., Deb, K., Boddeti, V.N.: Neural Architecture Transfer (i), 1–17 (2020), http://arxiv.org/abs/2005.05859

[19] McKay, M.D., Beckman, R.J., Conover, W.J.: A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics **42**(1), 55–61 (2000)

[20] Mersmann, O., Bischl, B., Trautmann, H., Preuss, M., Weihs, C., Rudolph, G.: Exploratory landscape analysis. In: Krasnogor, N., Lanzi, P.L. (eds.) 13th Annual Genetic and Evolutionary Computation Conference, GECCO 2011, Proceedings, Dublin, Ireland, July 12-16, 2011.

pp. 829–836. ACM (2011). https://doi.org/10.1145/2001576.2001690, https://doi.org/10.1145/2001576.2001690
[21] Muñoz, M.A., Kirley, M., Halgamuge, S.K.: Exploratory landscape analysis of continuous space optimization problems using information content. IEEE Trans. Evol. Comput. **19**(1), 74–87 (2015). https://doi.org/10.1109/TEVC.2014.2302006, https://doi.org/10.1109/TEVC.2014.2302006
[22] 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 (2015)
[23] Rawat, W., Wang, Z.: Deep convolutional neural networks for image classification: A comprehensive review. Neural computation **29**(9), 2352–2449 (2017)
[24] Renau, Q., Doerr, C., Dréo, J., Doerr, B.: Exploratory landscape analysis is strongly sensitive to the sampling strategy. CoRR **abs/2006.11135** (2020), https://arxiv.org/abs/2006.11135
[25] Rusu, A.A., Rabinowitz, N.C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., Hadsell, R.: Progressive neural networks (2016)
[26] Schwartz, R., Dodge, J., Smith, N.A., Etzioni, O.: Green ai (2019)
[27] Skvorc, U., Eftimov, T., Korosec, P.: Understanding the problem space in single-objective numerical optimization using exploratory landscape analysis. Appl. Soft Comput. **90**, 106138 (2020). https://doi.org/10.1016/j.asoc.2020.106138, https://doi.org/10.1016/j.asoc.2020.106138
[28] Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evolutionary computation **10**(2), 99–127 (2002)
[29] van Stein, B.: Basvanstein/learningfromnas: Initial release (Sep 2020). https://doi.org/10.5281/zenodo.4043005, https://doi.org/10.5281/zenodo.4043005
[30] van Stein, B., Wang, H., Bäck, T.: Automatic configuration of deep neural networks with parallel efficient global optimization. In: 2019 International Joint Conference on Neural Networks (IJCNN). pp. 1–7. IEEE (2019)
[31] Wang, J., Wu, J., Bai, H., Cheng, J.: M-nas: Meta neural architecture search. In: AAAI. pp. 6186–6193 (2020)
[32] Xiao, H., Rasul, K., Vollgraf, R.: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747 (2017)
[33] Zoph, B., Le, Q.V.: Neural architecture search with reinforcement learning. arXiv preprint arXiv:1611.01578 (2016)