Neural Architecture Search: A Visual Analysis

Gabriela Ochoa[1][0000−0001−7649−5669] and Nadarajen Veerapen[2][0000−0003−3699−1080]

[1] University of Stirling, Scotland, United Kingdom
gabriela.ochoa@stir.ac.uk
[2] Univ. Lille, CNRS, Centrale Lille, UMR 9189 CRIStAL, F-59000 Lille, France
nadarajen.veerapen@univ-lille.fr

Abstract. Neural architecture search (NAS) refers to the use of search heuristics to optimise the topology of deep neural networks. NAS algorithms have produced topologies that outperform human-designed ones. However, contrasting alternative NAS methods is difficult. To address this, several tabular NAS benchmarks have been proposed that exhaustively evaluate all architectures in a given search space. We conduct a thorough fitness landscape analysis of a popular tabular, cell-based NAS benchmark. Our results indicate that NAS landscapes are multi-modal, but have a relatively low number of local optima, from which it is not hard to escape. We confirm that reducing the noise in estimating performance reduces the number of local optima. We hypothesise that local-search based NAS methods are likely to be competitive, which we confirm by implementing a landscape-aware iterated local search algorithm that can outperform more elaborate evolutionary and reinforcement learning NAS methods.

Keywords: Neural Architecture Search, Fitness Landscapes, Local Optima Networks, Neuroevolution, Neural Networks, Deep Learning

1 Introduction

Neural architecture search (NAS) is a fast growing topic within automated machine learning (AutoML). The idea is to use search methods to automatically design the architecture (or topology) of deep neural networks. NAS has produced neural network models that surpass the performance of human-designed ones in image recognition [1, 2] and natural language processing [1, 3]. NAS is a relatively recent term, coined in 2017 by Zoph and Le [1], but the subject of research overlaps with earlier topics such as hyper-parameter optimisation, meta-learning and neuroevolution.

Neuroevolution, the use of evolutionary algorithms to design neural networks, has a long tradition in evolutionary computation with roots in the late 1980s and early 1990s [4]. Most neuroevolution systems optimise both the neural network topology and its weights. However, when scaling up to contemporary deep models with millions of weights for supervised learning tasks, gradient-based weight

optimisation generally outperforms evolutionary methods. In consequence, many recent neuroevolution systems use gradient-based weight optimisation and only evolve the topology [2, 5]. Other approaches to NAS include random search [6, 7], hill-climbing [7], reinforcement learning [1], Bayesian optimisation [8], and gradient-based optimisation [3].

NAS is generally formulated as a discrete optimisation problem $max_{a \in A} f(a)$, where A denotes a set of architectures (the search space) and $f(a)$ denotes the objective function to be maximised[3], often set to the validation accuracy after training with a fixed set of hyper-parameters. Several search spaces have been studied [9], including chain-structured networks, which encode a sequence of layers; multi-branch networks, which incorporate skip connections; and networks consisting of repeated motifs also called *cells* or *blocks*. These cell-based architectures are designed by combining repeated cells in a predefined arrangement. Despite the underlying complexity of deep neural network architectures, many NAS spaces can be encoded as fixed-length strings of symbols of a given alphabet, where symbols represent predefined operations. This is the case of the search space considered in our study (see Section 2 for details).

The performance of NAS algorithms crucially depends on the search space structure. However, very little work has been devoted to analysing NAS fitness landscapes [7, 10]. Our work is inspired by White et al. [7] findings when studying NAS loss landscapes, which the authors summarise as follows: "... we show that (1) the simplest hill-climbing algorithm is a powerful baseline for NAS, and (2), when the noise in popular NAS benchmark datasets is reduced to a minimum, hill-climbing outperforms many popular state-of-the-art algorithms". Our contributions are to:

- Analyse the fitness landscapes of a established NAS benchmark, using three landscape analysis techniques not previously used in this setting: (i) density-of-states [11], fitness-distance correlation [12], and local optima networks [13]. These techniques explore the landscape global structure and have a strong visual component.
- Explore the impact of reducing the noise in estimating the fitness function (validation accuracy) on the NAS landscape structure.
- Propose a local search-based NAS method informed by the structure of NAS landscapes.

2 The Selected NAS Benchmark

It is challenging to provide fair and statistically sound comparisons among NAS methods due to the different search spaces and training setups, as well as the high computational costs [9, 6]. In response to this challenge, several tabular NAS benchmarks have been proposed, which exhaustively evaluate all architectures in a given search space, and store a wealth of training, evaluation and testing metrics in queryable look-up tables [14–16]. This facilitates the reproducibility

[3] NAS can also be formulated as a minimisation problem (minimising validation loss).

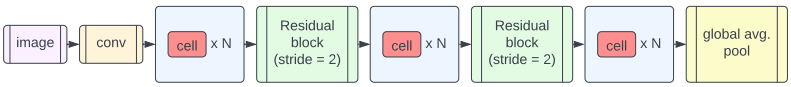


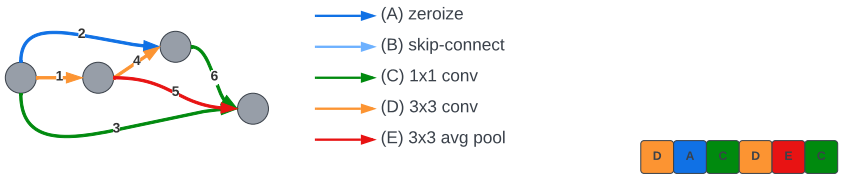
Fig. 1: The macro skeleton of candidate architectures in the search space. The skeleton is shared by all architectures and only the configuration of the cell (visualised in red) is subject to change.

of NAS experiments, drastically reduces the computational costs, and fosters a wider uptake of this topic. Our experiments use one of these tabular benchmarks, specifically, the cell-based topology search space S_t in NATS-Bench [16], also called NAS-Bench-201 [15].

The NATS-Bench Topology Search Space was inspired by the successful cell-based NAS algorithms [1–3], it consists of a predefined macro skeleton where modular (searchable) cells are stacked. Figure 1 illustrates the macro skeleton. The architecture starts with one 3×3 convolution layer with 16 output channels and a batch normalisation layer. The main body contains three stacks of cells, connected by a residual block. Each cell is stacked $N = 5$ times. The architecture ends with a global average pooling layer that flattens the feature map in to a vector. A fully connected layer with softmax is used for the final classification [16]. The macro skeleton remains fixed for all architectures, what changes is the configuration of the red cell in Fig. 1. For a given architecture, all the cells in the macro skeleton will have the same structure. Therefore, searching for a suitable architecture is reduced to searching for a suitable cell.

A cell is represented as a dense directed acyclic graph (DAG), as illustrated in Figure 2a. Each edge in this DAG is associated with an operation that transforms the feature map from the start to the end node. Operations are selected from a predefined set of five: (A) zeroize, (B) skip connection, (C) 1×1 convolution, (D) 3×3 convolution, and (E) 3×3 average pooling layer. The *zeroise* operation simply drops the associated edge. Therefore, the cell topology is not restricted to densely connected DAGs. The nodes represent the sum of the feature maps from the incident edges. The DAG has $V = 4$ nodes. This number was chosen to allow the encoding of basic residual block-like cells, which require 4 nodes. A complete graph with 4 nodes has combinations of 4 in 2, $\binom{4}{2} = 6$ edges. Since each edge can be one of 5 operations, the search space contains $5^6 = 15,625$ unique neural architectures. Each architecture was trained three times using different random seeds on three popular image classification datasets: CIFAR-10, CIFAR-100, and ImageNet-16-120. The training pipeline and hyper-parameters is the same for all architectures. NATS-Bench [16] provides training, validation, and test loss and accuracy metrics for all architectures that can be queried via an API[4] with negligible computational costs.

[4] `https://github.com/D-X-Y/NATS-Bench`



(a) Cell (b) Genotype

Fig. 2: Encoding of an example architecture showing the mapping from a cell to the corresponding linear genotype. (a) A cell is represented as DAG with six edges representing operations taken from a fixed set of five operations (A – E) as indicated in the legend. (b) A candidate solution (genotype) is encoded as a string of six symbols, each representing the operation associated with the numbered edge in the DAG.

3 Fitness Landscape Analysis

A fitness landscape [17] is a triplet (S, N, f) where S is a set of admissible solutions i.e., a search space, $N : S \longrightarrow 2^S$, is a neighbourhood structure, a function that assigns a set of neighbours $N(s)$ to every solution $s \in S$, and $f : S \longrightarrow \mathbb{R}$ is a fitness function that measures the quality of the corresponding solutions. We define below these three components for our NAS formulation.

Search space. The search space consists of strings of length $n = 6$ (the number of edges in the DAG representing the cell, Fig. 2a) in the alphabet $\Sigma = \{A, B, C, D, E\}$, where each symbol represents a predefined operation. An example genotype is given in Fig. 2b, where the symbol at position i corresponds to the operation associated to edge i in the DAG. The size of the search space is $|Sigma|^n$, that is, $5^6 = 15,625$ as indicated in Section 2.

Neighbourhood Structure. We use the standard Hamming distance 1 neighbourhood (1-change operator). The Hamming distance between two strings is the number of positions in which they differ. Therefore, the neighbourhood $N(s)$ of solution s includes the set of all solutions at a maximum Hamming distance 1 from s. The size of the neighbourhood is $n \times (|\Sigma| - 1)$, that is, $6 \times 4 = 24$.

Fitness Function. To measure the performance of each cell we consider the validation accuracy metric, to be maximised. In NATS-Bench, every architecture (cell) was independently trained three times using three different random seeds. Therefore, there are three sets of metrics for each image dataset. Since we are interested in exploring the effect of noise in the fitness landscape, we follow the approach in [7], where two ways to draw the validation metric were considered: (i) using a single value, and (ii) using the average of the three values to obtain

a less noisy estimate. We therefore consider two fitness functions that we call f_{sng} and f_{avg}, to refer to using a single validation accuracy or the average of the three available values, respectively.

3.1 Density of States

The density of states (DOS) [11], plots the number of solutions in the search space with a certain fitness value. Normally, this plot requires sampling the search space, but since we have access to the whole space, we do not need a sample and instead use the complete set of solutions. The density of states gives an indication of the performance of random search or a random initialisation of metaheuristics, as it gives the probability of having a given fitness value when a solution is randomly chosen. Moreover, the right tail of the distribution near optimal fitness values gives a measure of the difficulty of a maximisation problem, the faster the decay, the harder the problem.

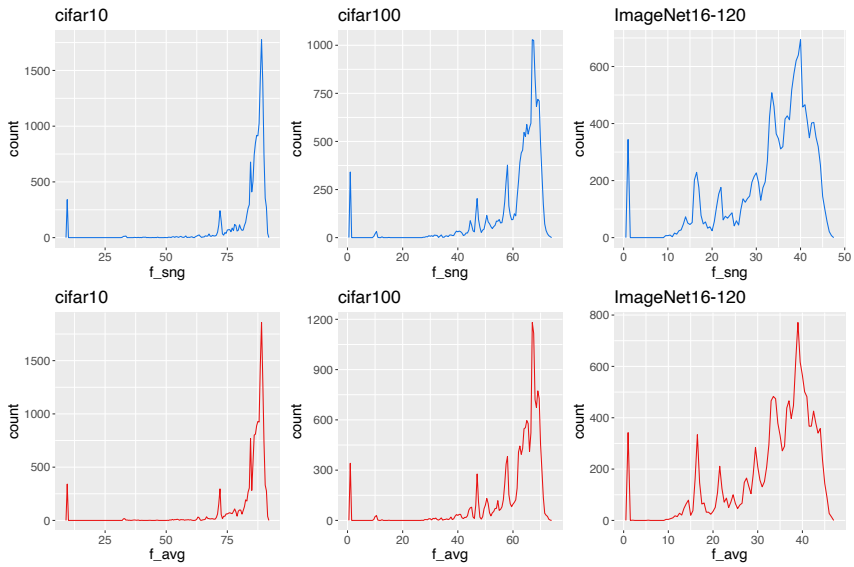


Fig. 3: Density of states for the two fitness functions f_{sng} (top) and f_{avg} (bottom) on all the datasets. The x-axis shows the whole range of validation accuracy values for each dataset, grouped in bins of width 0.5 in order to draw the frequency polygons.

In order to visualise the distribution of fitness function values, Figure 3, shows frequency polygon plots contrasting the distribution across the two fitness functions, f_{sng} (top) and f_{avg} (bottom), for the three image datasets. There is no clear visual difference between the distributions of the two fitness functions.

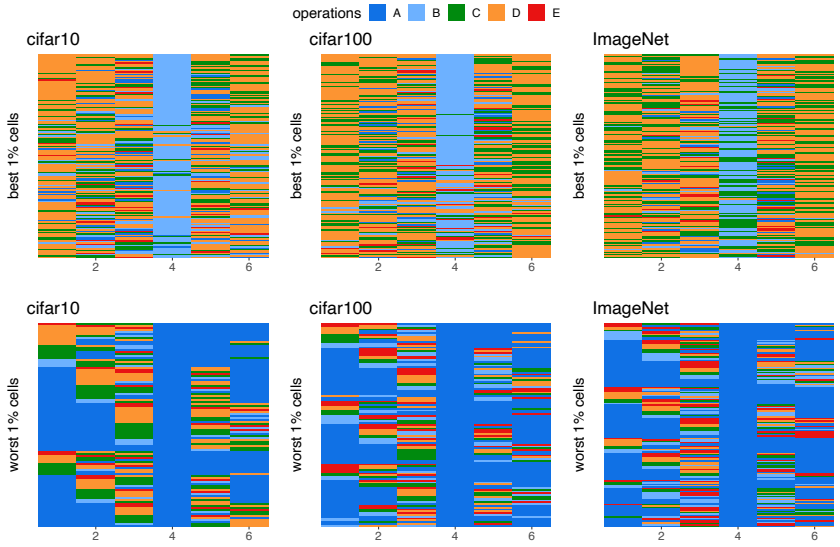


Fig. 4: Genotype maps of the best 1% (top), and worst 1% (bottom) performing cells for all datasets, sorted according to f_{avg}. Each line in the plots visualises a cell where positions are coloured according to the respective operation.

Notice that the range of accuracy values (x-axis) is different for each image dataset, which is consistent with the difficulty of the respective classification task. The DOS curves show a faster decay towards near optimal fitness in all cases, indicating that NAS landscapes are not completely smooth. For ImageNet, there is wider range of accuracy values with high frequency of solutions, indicating a more complex landscape. Another interesting observation is the high frequency of cells with a low accuracy near zero in all plots. A close inspection revealed that these low accuracy cells correspond to genotypes where three or more of the symbols are 'A', that is *zeroise* (dropped) operations, so they are mostly no-operation, empty cells, which explains their low performance.

Figure 4 visualises the configuration of the best 1% (top plots), and worst 1% (bottom plots) performing cells (genotypes) for the three datasets according to f_{avg}. Each line in these plots is a cell configuration (solution in the search space), where positions are visualised with colours identifying operations. The rows are sorted by their f_{avg} value, where the cell with the highest fitness value (highest average accuracy) in the set is the top line of each plot. We can clearly see that the low performing cells (bottom plots) are those containing a majority of 'A' (zeroise, or drop) operations, thus they are mostly empty cells. Specially the 4^{th} positions is always an 'A' in the worst performing configurations. The best performing configurations (top plots) also show a visible pattern, with the most common operations being 'C' (green) and 'D' (orange), corresponding to 1×1 and 3×3 convolutions, respectively. The exception is the 4^{th} position, which for

all datasets is mostly a 'B' (light blue, skip connection) in the best performing cells. The plots suggest that the choice of operation for the 4^{th} position (4^{th} edge in the DAG cell, Fig. 2a) has more impact in performance than the other positions. An analysis of the frequency of operations in the top 1% performing cell across the 3 datasets revealed that they rank as follows: D, C, B, A, E with frequency percentages of: 45.7, 24.8, 19.2, 6.1, 4.1, respectively. We argue that this information can be used to design informed mutation operations that can improve the performance of search heuristics in this domain, and we set to do that in Section 4.

3.2 Fitness Distance Correlation

Since the whole search space is available, and thus the optimal cell is known, we can compute the distances from all cells in the search space to the global optimum. Specifically, for each cell i we have a pair (f_i, d_i), where f_i is the validation accuracy (either f_{sng} or $f_{sng)}$ of cell i and d_i is the Hamming distance to the cell with the highest validation accuracy (global optimal cell). The FDC is calculated as the (Spearman) correlation coefficient of this set of (accuracy, distance) pairs.

Figure 5 shows the FDC plots, as well as the Spearman correlation coefficients (R) with significance level (p-values) for each image classification dataset. The top plots show the measurements with a single training seed (f_{sing}), while the bottom plots show the average of the 3 training seeds available (f_{avg}). The regression lines with confidence regions (95%) are also shown. The horizontal axes show the Hamming distance between all architectures and the global optimal architecture, while the vertical axes show the validation accuracy of each architecture. From these plots we can observe that there is a moderate negative correlation (ranging from -0.33 to -0.46) between distance and fitness (validation accuracy), suggesting a gradient towards the global optimum. However, for all studied scenarios, some configurations that differ in 3 or 4 operations from the global optimum reveal a low accuracy value, these are the cells with a high number of *zeroise* operations. For the three datasets, the correlations coefficients are higher when the less noisy estimation of fitness f_{avg} is considered, supporting the insight from [7] indicating that reducing noise in the estimation of fitness can improve search. The range of possible values for R is $[-1, 1]$ where, for a maxisimisation problem, high negative correlations would be regarded as easier for a hill climber. When FDC was proposed [12], $-0.15 \leq R \leq 0.15$ was classified as hard, and $R \leq -0.15$ was considered as misleading for a minimisation problem. Using these criteria, none of the problems used in this study is regarded as hard or misleading. The ImageNet dataset reveals the lowest correlation coefficient, which supports that this is hardest of the three instances.

3.3 Local Optima Networks

To further understand the landscapes' global structure, we extract and analyse local optima networks (LONs)[13]. LONs are graph-based model of landscapes

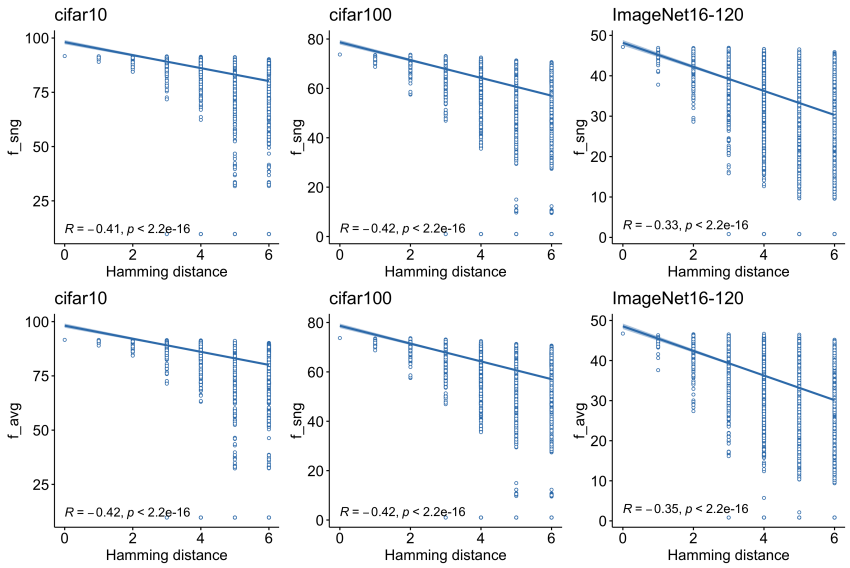


Fig. 5: FDC plots for all datasets. The horizontal axes show the Hamming distance to the global optimum, using the f_{sng} fitness values (top plots), and the less noisy f_{avg} values (bottom plots). The Spearman correlation coefficients with p-value are also shown.

where nodes are local optima and edges are transitions among optima with a given search operator.

Definitions. The relevant definitions, and the procedure to construct the LON modes, are given below.

Local optima. A local optimum, which in our NAS formulation is a maximum, is a solution l such that $\forall s \in N(l)$, $f(l) > f(s)$. Local optima are identified with a best-improvement hill-climbing heuristic using the 1-change (Hamming distance 1) neighbourhood. The set of local optima, denoted by L, corresponds to the nodes in LON model.

Edges. Edges are directed and based on the perturbation operator 2-change. There is an edge from local optimum l_1 to local optimum l_2, if l_2 can be obtained after applying a random perturbation (changing at random 2 locations in the genotype) to l_1 followed by local search. Edges are weighted with estimated frequencies of transition in a sampling process. The weight is the number of times a transition between two local optima occurred when constructing the LON models as detailed below. The set of edges is denoted by E.

LON. The LON is the directed graph LON $= (L, E)$, with node set L, and edge set E as defined above.

LON Sampling and Construction. To construct the LON models for each dataset and fitness function, a sampling process is conducted. It consists in running an iterated local search algorithm (ILS) [18], where the stopping condition is set to $t = 100$ iterations without any improvement. This serves the purpose of empirically estimating the global optimum or the end of a *funnel*, i.e., a solution at the end of an ILS trajectory, where escaping is difficult, if not impossible. While running ILS, we store in a set L all the unique optima obtained after the local search stage, and in a set E all the unique edges obtained after a perturbation followed by local search. To construct the LONs for each image dataset and fitness function, these sets of nodes and edges are aggregated over 1 000 runs, started from different random configurations.

Network Visualisation. One advantage of network models is that they can be visualised, bringing useful insight into their structure. Figure 6 illustrates the LONs for all datasets and fitness functions. The networks capture the whole set of sampled nodes and edges in each case. In the plots, each node is a local optimum and edges are perturbation transitions. Plots were produced using *force-directed* layout methods as implemented in the igraph R library [19]. The global optimum, which was unique in all cases, is highlighted in red. The other local optima are painted in grey. The edges' colour indicate whether they end in a node with better fitness (dark gray), worse fitness (orange) or equal fitness (blue). The size of nodes is proportional to their incoming weighed degree, so larger nodes indicate attractors in the search process.

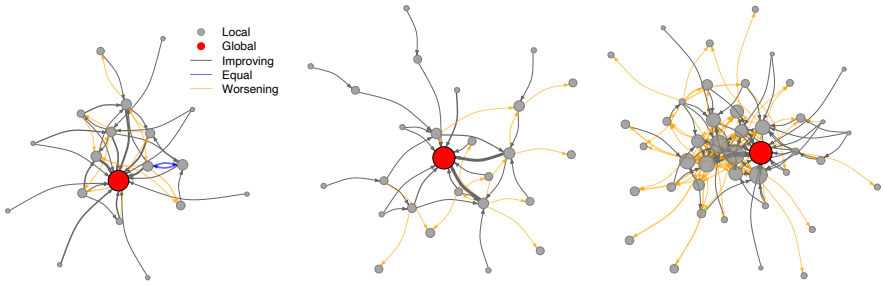
The networks in Fig. 6 indicate that for all datasets and fitness functions, there is a connected component of nodes that can reach the global optimum (red node) in a few search steps. This is indicative of a multi-modal landscape, but where it is not too hard to reach the global optimum. For all datasets, there are fewer local optima when the less noisy fitness function f_{avg} is used (Figs. 6d, 6e and 6f), which indicates that improving the fitness estimate facilitates the search process, as the search paths to the global optimum become shorter.

The LON analysis indicates that the edges considered (changing 2 locations in the genotype), allow escaping local optima in most cases. This suggests that a local search based method, coupled with a escape mechanism can be a suitable NAS search strategy. This is empirically explored in the next section.

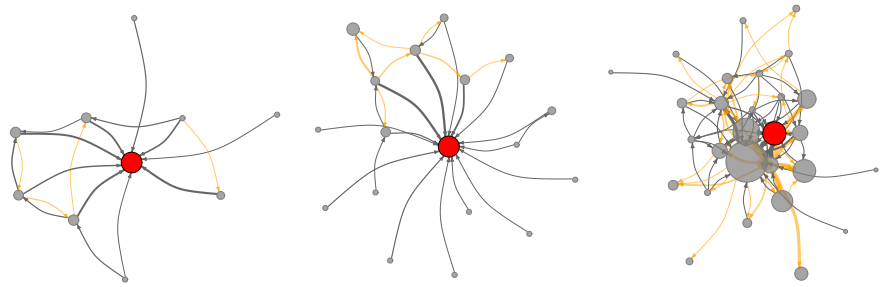
4 Search Performance Analysis

4.1 Competing Algorithms

We propose and implement a local search based algorithm, specifically an iterated local search (ILS) method [18]. ILS is a simple yet powerful metaheuristic that alternates a local search stage with a perturbation stage. We use a first improvement local search with a 1-change neighbourhood, and a perturbation operator that changes 2 positions in the incumbent solution. Only improving moves are accepted. We consider two versions of the ILS: `ILS-shuffle` where



(a) cifar10, f_{sng}, (19,44) (b) cifar100, f_{sng}, (26,41) (c) ImageNet, f_{sng}, (43,145)



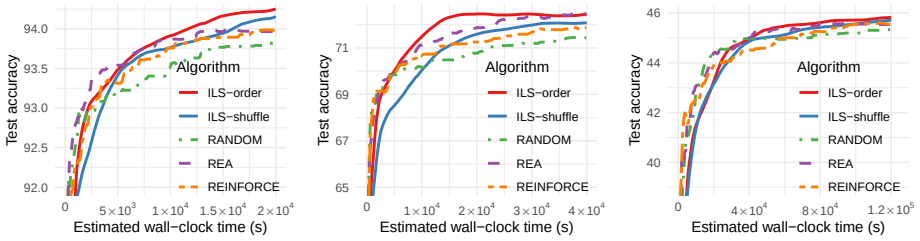
(d) cifar10, f_{avg}, (10,18) (e) cifar100, f_{avg}, (18,27) (f) ImageNet, f_{avg}, (28,98)

Fig. 6: LONs for all datasets and the two fitness functions. For each model, the number of nodes n and edges e are indicated as (n, e).

the values for the 1-change operator are explored in random order and `ILS-order` where the 1-change operator uses insights from the landscape analysis. Specifically, following the frequency profile observed in the 1% best-performing cells (Section 3.1 and Fig. 3) we systematically explore neighbours using the following ordering of the operations: D, C, B, A, E.

We contrast our proposed ILS against the following NAS methods, as implemented in [16].

- **Random search** (RANDOM) [6]. This serves as the baseline. It draws cells at random and returns the best found.
- **Regularised evolution** (REA) [2]. This is a mutation only evolutionary algorithm that uses tournament selection and introduces the notion of age to the individuals. The replacement strategy removes the oldest individual in the population, thus favouring newer cells. This serves as a mechanism to handle the noisy performance estimation.
- **Reinforcement learning** (REINFORCE) [1]. This approach frames NAS as a reinforcement learning problem. The generation of a neural cell correspond to the agent's actions, with the action space identical to the search



(a) cifar10 (b) cifar100 (c) ImageNet

Fig. 7: Evolution of average test accuracy across the three datasets.

space. The agent's reward is based on an estimate of the cell performance on unseen data.

4.2 Empirical Setup

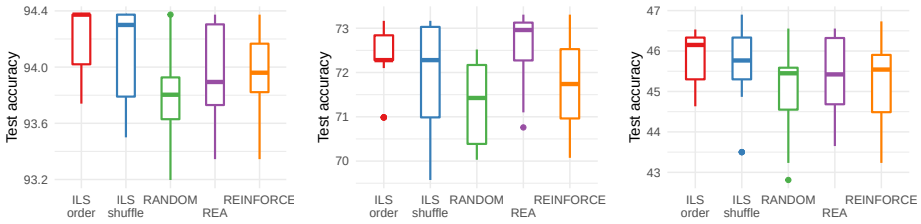
Our experiments follow the protocol suggested by NATS-Bench [16]. The benchmark provides performance data on each neural architecture for two scenarios: one with 12 epochs, the other 200. The epoch indicates the number of times the entire training dataset is used while building the model.

We replicate the NATS-Bench experiments by training the models over 12 epochs and using the accuracy calculated on the validation set as feedback to direct the search. This is meant to simulate a faster but less accurate training step. The configurations obtained are then evaluated against the test set of the 200 epoch scenario. The best solution found using 12 epochs is therefore not necessarily the best for 200 epochs. The training time budgets considered for cifar10, cifar100 and ImageNet datasets are 20 000, 40 000 and 120 000 seconds respectively. Each algorithm is executed 30 times.

4.3 Results

The average test accuracy is presented on Figure 7. The different methods have fairly similar behaviour. ILS, for its part, initially converges slightly slower than the rest since it is costly to evaluate multiple neighbours before accepting a new solution. However, on average, it manages to get ahead of the other approaches within the time budget on the cifar10 and ImageNet datasets. Using `ILS-order` improves convergence speed and the final result. As was previously noted [16], despite its simplicity, random search performs comparatively well, even if it comes in last.

In order to better grasp the overall performance of the different algorithms, Figure 8 presents boxplots of the test accuracy calculated on the cell configurations found at the end of each run. The `ILS-order` boxplots are fairly tight, indicating that there is little spread in the quality of configurations obtained. In contrast, the results for `ILS-shuffle` on cifar10 and cifar100 are much more



(a) cifar10 (b) cifar100 (c) ImageNet

Fig. 8: Test accuracy distribution for configurations found at the end of 30 runs.

spread out. This is because its slower convergence means it hasn't yet reached the very bottom of the landscape. This is not the case on the ImageNet dataset where both ILS versions converge to similar solutions within the allotted budget.

On this problem, the challenge for optimisation methods compared to classic optimisation problems is that the function used to evaluate the end result (test accuracy) is not the same as the objective function (validation accuracy). We know from sampling the landscape and LON analysis that ILS is able to reach the global validation accuracy optima on the benchmarks when there is no time limit, however this is not a guarantee that the same solution will be the best for test accuracy.

Overall, ILS proves to be a viable and competitive approach for optimising neural network topology, especially if appropriate design choices are implemented. Despite its conceptual simplicity, ILS is able to match and even outperform more sophisticated approaches within the time budget in this NAS topology benchmark.

5 Conclusions

We analysed the fitness landscape of a popular tabular, cell-based NAS benchmark for image classification. Our analysis revealed that the landscapes are not trivial to search, they are rugged (multi-modal), however they have a relatively low number of local optima, from which it is not difficult to escape with a simple perturbation operation. Our analysis of the best-performing cells indicated that some operations of the available set appear more frequently than others. We used this information to design a conceptually simple, yet high-performing local search based NAS method. On the studied benchmark, our iterated local search (ILS) implementation outperforms both a reinforcement learning method on the 3 available image datasets, and the state-of-the-art evolutionary method on 2 of the 3 image datasets. Future work will analyse the landscapes of other available NAS-benchmarks, as well as test the performance of the proposed ILS on them. We will also incorporate noise-handling mechanisms into our ILS approach since our landscape analysis reported a smoother landscape when noise is reduced.

References

1. Zoph, B., Le, Q.V.: Neural architecture search with reinforcement learning. In: Conference on Learning Representations, ICLR (2017)
2. Real, E., Aggarwal, A., Huang, Y., Le, Q.V.: Regularized evolution for image classifier architecture search. In: AAAI Conference on Artificial Intelligence, AAAI. pp. 4780–4789. AAAI Press (2019)
3. Liu, H., Simonyan, K., Yang, Y.: DARTS: differentiable architecture search. In: Conference on Learning Representations, ICLR (2019)
4. Schaffer, J., Whitley, D., Eshelman, L.: Combinations of genetic algorithms and neural networks: a survey of the state of the art. In: International Workshop on Combinations of Genetic Algorithms and Neural Networks. pp. 1–37 (1992)
5. Lu, Z., Whalen, I., Boddeti, V., Dhebar, Y., Deb, K., Goodman, E., Banzhaf, W.: Nsga-net: Neural architecture search using multi-objective genetic algorithm. In: Genetic and Evolutionary Computation Conference (GECCO). p. 419–427. ACM, New York, NY, USA (2019)
6. Yu, K., Sciuto, C., Jaggi, M., Musat, C., Salzmann, M.: Evaluating the search phase of neural architecture search. In: Conference on Learning Representations, ICLR (2020)
7. White, C., Nolen, S., Savani, Y.: Exploring the loss landscape in neural architecture search. In: Conference on Uncertainty in Artificial Intelligence, UAI. Proceedings of Machine Learning Research, vol. 161, pp. 654–664. AUAI Press (2021)
8. Kandasamy, K., Neiswanger, W., Schneider, J., Póczos, B., Xing, E.P.: Neural architecture search with bayesian optimisation and optimal transport. In: Advances in Neural Information Processing Systems, NeurIPS 2018. pp. 2020–2029 (2018)
9. Elsken, T., Metzen, J.H., Hutter, F.: Neural architecture search: A survey. J. Mach. Learn. Res. 20, 55:1–55:21 (2019)
10. Rodrigues, N.M., Silva, S., Vanneschi, L.: A study of generalization and fitness landscapes for neuroevolution. IEEE Access 8, 108216–108234 (2020)
11. Rosé, H., Ebeling, W., Asselmeyer, T.: The density of states – a measure of the difficulty of optimisation problems. In: Parallel Problem Solving from Nature, PPSN IV. pp. 208–217. Springer (1996)
12. Jones, T., Forrest, S.: Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In: International Conference on Genetic Algorithms. pp. 184–192. Morgan Kaufmann (1995)
13. Ochoa, G., Tomassini, M., Verel, S., Darabos, C.: A study of nk landscapes' basins and local optima networks. In: Genetic and Evolutionary Computation Conference, GECCO. pp. 555–562. ACM (2008)
14. Ying, C., Klein, A., Christiansen, E., Real, E., Murphy, K., Hutter, F.: NAS-Bench-101: Towards reproducible neural architecture search. In: International Conference on Machine Learning, ICML. vol. 97, pp. 7105–7114. PMLR (2019)
15. Dong, X., Yang, Y.: NAS-Bench-201: Extending the scope of reproducible neural architecture search. In: Conference on Learning Representations, ICLR (2020)
16. Dong, X., Liu, L., Musial, K., Gabrys, B.: NATS-Bench: Benchmarking nas algorithms for architecture topology and size. IEEE Transactions on Pattern Analysis and Machine Intelligence (2021)
17. Stadler, P.F.: Fitness landscapes. Appl. Math. and Comput 117, 187–207 (2002)
18. Lourenço, H.R., Martin, O.C., Stützle, T.: Iterated Local Search: Framework and Applications, pp. 363–397. Springer US, Boston, MA (2010)
19. Csardi, G., Nepusz, T.: The igraph software package for complex network research. InterJournal Complex Systems, 1695 (2006)