Channel Configuration for Neural Architecture: Insights from the Search Space

Sarah L Thomson, Gabriela Ochoa, Nadarajen Veerapen, Krzysztof Michalak

Channel Configuration for Neural Architecture: Insights from the Search Space

Sarah L. Thomson
University of Stirling
Stirling, United Kingdom
s.l.thomson@stir.ac.uk

Gabriela Ochoa
University of Stirling
Stirling, United Kingdom
gabriela.ochoa@stir.ac.uk

Nadarajen Veerapen
Univ. Lille, CNRS, Centrale Lille, UMR 9189 CRIStAL
Lille, France
nadarajen.veerapen@univ-lille.fr

Krzysztof Michalak
Wroclaw University of Economics
Wroclaw, Poland
krzysztof.michalak@ue.wroc.pl

ABSTRACT

We consider search spaces associated with neural network channel configuration. Architectures and their accuracy are visualised using low-dimensional Euclidean embedding (LDEE). Optimisation dynamics are captured using local optima networks (LONs). LONs are a compression of a fitness landscape: the nodes are local optima and the edges are search transitions between them. Several neural architecture search algorithms are tested on the search space and we discover that iterated local search (ILS) is a competitive algorithm for neural channel configuration. We additionally implement a landscape-aware ILS which performs well. Observations from the search and landscape space analyses bring visual clarity and insight to the science of neural network channel design: the results indicate that a high number of channels, kept constant throughout the network, is beneficial.

CCS CONCEPTS

• **Mathematics of computing** → **Graph algorithms**; *Combinatorial algorithms*; • **Theory of computation** → **Evolutionary algorithms**.

KEYWORDS

Fitness Landscapes, Neural Architecture Search, Local Optima Networks (LONs)

1 INTRODUCTION

Neural architecture search (NAS) [9] is the pursuit of intelligently and automatically designing neural network topology. Architectures engineered in this way have been shown to match or exceed the performance of manually-designed architectures in object recognition [41] natural language processing [17] and dense image prediction [5]. As an optimisation problem, the search space is the set of candidate neural network architectures and fitness is typically the validation accuracy using an architecture, which is to be maximised. Several algorithmic approaches have been proposed for neural architecture search; these include reinforcement learning [41], gradient-based search [27], evolutionary algorithms [33], and random search [16]. Recently, studies have shown that local search is a competitive family of algorithms in this domain [7, 38].

Local optima networks (LONs) [22] are a compression of a fitness landscape. The nodes are local optima, and the edges are heuristic transitions between them. LONs have been employed to explain and predict heuristic search performance on a variety of benchmark combinatorial problems, including NK Landscapes [34]; quadratic assignment [1]; and the travelling salesperson [4]. Lately, LONs have been used to gain insights into aspects of machine learning - autoML search spaces [31, 32], feature selection [19], and also NAS [23, 24]. In this work, we closely analyse the search spaces of neural architecture channel configuration search. As a first step, genotype mapping and Low-Dimensional Euclidean Embedding (LDEE) [18] are employed to visualise and understand the characteristics of the search spaces. Local optima networks are constructed and their structure investigated. In both of these stages, the spaces associated with both validation accuracy and test accuracy are considered, and these are compared to each other. Finally, we execute several NAS algorithms on the channel optimisation space and find that iterated local search is a competitive approach for this problem. The contributions of this work are as follows:

(1) First fitness landscape analysis for neural network channel optimisation, and the largest NAS space to be modelled using LONs
(2) Analysis of the relationship between the validation and test search spaces
(3) Proposal of iterated local search approaches for channel configuration search, including a landscape-aware variant

2 BACKGROUND

2.1 Fitness Landscapes

A fitness landscape [29] is composed of three parts: $(S, N, f) : S$ is the full set of possible solutions; $N : S \longrightarrow 2^S$ is the neighbourhood function, which assigns a set of adjacent solutions $N(s)$ to every $s \in S$; and f is a fitness function $f : S \longrightarrow \mathbb{R}$ that provides a mapping from solution to associated fitness. That fitness can be conceptualised as the solution *height* within the landscape metaphor.

2.2 NAS Landscapes

NAS spaces have been studied previously using fitness landscapes (sometimes called *loss landscapes* in the literature). A recent study [38] found that minimising noise in the neural network training pipeline (the effects of random instantiation of the network weights) also reduces the number of local optima. They also noticed that

increasing the noise caused the basin of attraction surrounding the global optimum to diminish. Another work analysed graph neural network architecture spaces and found that the landscape was straightforward and exhibited very little neutrality [21]. Local optima networks have been constructed for neural architecture search spaces on three occasions recently. One study considered multilayer perceptrons with up to three hidden layers and found that the fitness landscapes typically exhibited a "big valley" (single funnel) global structure [24]. The architectures in the search space struggled to model the cifar10 and cifar100 datasets, although they worked well with other datasets. Convolutional network architectures were considered in another work [26]; a grammar-based approach to solution representation was used to encode hyperparameters such as the number of filters. The search space, at 800 architectures, was small. As a result, the extracted LONs were also small: between two and five nodes.

Another study constructed LONs for the topology search space of the NATS-Bench benchmark and the cifar10, cifar100, and ImageNet16-120 datasets [23]. They found that the landscapes contain quite a low number of local optima and demonstrated that iterated local search was, consequently, a competitive algorithm for traversing the space. Our own study considers the largest NAS space which has been subject to LON analysis, at 32 768 possible architectures; the other works had search space sizes of 800 [26], 1 110 [24], and 15 625 [23] respectively. To the best of our knowledge, our study is the first to conduct landscape analysis of the channel optimisation problem.

2.3 Neural Architecture Benchmark

We consider the "size" search space from the tabular benchmark NATS-Bench [8], made available through queries to the NATS-Bench API[1]. Here, the notion of size does not refer to the number of parameters in the network; but rather, the depth of its convolutions. In the text that follows, we elaborate on what this means and describe the network anatomies. The search space is based around convolutional neural networks (CNNs) and a foundational macro skeleton as the overall network architecture; the skeleton contains cells. Cells represent "repeated motifs" in a network and are alternatively called blocks. An illustration of the skeleton can be seen in Figure 1. An initial 3x3 convolution stem is followed by an alternating sequence of three cell blocks and two residual blocks. Although cells can be stacked (as indicated with $x\ N$ in the Figure), this search space uses a single cell per block. The set of operations inside a cell are the best possible configuration on cifar100 obtained through optimisation of a cell topology search space. Those operations consist of 3x3 convolution, 1x1 convolution, and a skip connection. The final component to the skeleton is a global average pooling layer, which transforms the outputs into a vector of features. Those vectors then pass through a fully connected layer and through the softmax function in order to obtain the network output (prediction).

Each of the cell and residual blocks employ a number of *channels*. In convolutional neural networks, a channel is the depth dimension of the matrices which are used for convolution. An RGB input image has three channels (three matrices) initially: red, green, and blue. These encode the respective amounts of the three colours in the image. The initial number of channels must match the number in the raw image, but the number of channels in the rest of the network can be configured. More channels facilitate more complex and specific features. Channel configuration is important because it can affect performance of the network: more channels can allow the detection of diverse and important features. In the benchmark employed here, eight sizes are allowed for the number of channels: 8, 16, 24, 32, 40, 48, 56, and 64. Duplicate sizes are allowed, giving a total search space size of $8^5 = 32768$ possible solutions.

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

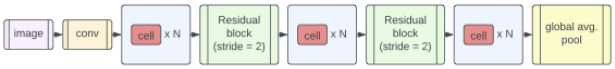


Figure 1: Macro skeleton of candidate architectures in the search space, shared by all architectures

2.4 Channel Optimisation

A prevalent approach to neural network channel design doubles the number of channels at downsampling locations (when the feature map is reduced in size by half). This method has been employed in well-known neural architectures such as VGG [28] and ResNet [12]. Another common approach to configuring channels is progressively increasing them in number as the network gets deeper (without contingency on feature map dimension); this is called a *pyramid structure* [10]. Literature has suggested a linear increase of channels over blocks is beneficial [11] and another promising approach is scaling the number of channels alongside the network depth and image resolution [30]. The paper which proposed the benchmark used in our work [8] found that pyramid-structure neural networks were sub-optimal when it comes to the trade-off between number of parameters and test accuracy. Other recent neural architecture studies focusing on channel configuration have shown that doing so can produce architectures which perform better than standard baselines [13, 35, 36].

We endeavour to understand the search for channel size. To the best of our knowledge there has not hitherto been a fitness landscape analysis for this type of neural architecture space and this is one of the novel contributions. Additionally, we use the search space and landscape analyses to inform our proposal of iterated local search (ILS) approaches for channel optimisation. ILS is a metaheuristic which repeatedly combines large and random mutations (*perturbations*, for the purpose of diversification) with local search (intensification).

3 DEFINITIONS

3.1 Optimisation Problem Formulation

The solution representation, or genotype, used in this study is a vector of integers of length five. The integers each map to a channel configuration from the set of candidate sizes: 8, 16, 24, 32, 40, 48, 56, and 64. Each position in the genotype represents a block in the neural network; these are blocks 3-7 in Figure 1. We use two fitness functions: the validation accuracy and the test-set accuracy, respectively, of neural networks which employ the

specified channel sizes. For both fitness functions, the accuracy is averaged over three random network weight instantiations. The reason for this is that there is data available for exactly three seeds in the queryable benchmark API. The notion of neighbourhood is defined as a 1-change operation; that is, the replacement of one gene with another channel size from the set of eight candidate sizes.

3.2 Local Optima Networks

Local optima networks (LONs) [22] are a means to study the global structure of a fitness landscape. In this work, we consider LONs which are sampled using ILS. This section formally describes this type of LON.

Nodes. The nodes, LO, are the local optima. That is, a local optimum (node) lo_i has superior fitness with respect to the entire neighbourhood. Formally: $\forall n \in N(lo_i) : f(lo_i) > f(n)$ (assuming maximisation, as is the case for this study) where $N(lo_i)$ is the neighbourhood and n is a particular neighbour.

Edges. There is an edge from local optimum lo_i to local optimum lo_j, if lo_j can be obtained after applying a random perturbation (in the present study, this is k-exchange) to lo_i followed by local search, and $f(lo_j) \geq f(lo_i)$. In LON terminology, these are called *escape* edges. The edges are called *monotonic* because they record only non-deteriorating, directed connections between local optima. Edges are weighted with the frequency of transition. The set of edges is denoted by E.

Local optima network (LON). A local optima network, LON = (LO, E), consists of nodes $lo_i \in LO$ which are the local optima, and edges $e_{ij} \in E$ between pairs of nodes lo_i and lo_j with weight w_{ij} iff $w_{ij} > 0$. A LON which only includes neutral or improving transitions between local optima is a *monotonic* LON, or MLON.

4 EXPERIMENTAL SETUP

In the tabular benchmark, each neural architecture is trained, validated, and tested on three image classification datasets: cifar10 and cifar100 [14], alongside ImageNet16-120 [6]. For training, two epoch settings are used: 12 and 90. For each of those scenarios, and for each architecture candidate, training is executed three separate times, differentiated by the random seed used to initialise the neural network weights. All hyperparameters are fixed as the same for each neural network architecture with the exception of number of epochs (already described) and the channel size configurations (these constitute the search space). The hyperparameters are: a Nesterov momentum stochastic gradient descent as optimiser; batch size of 256; weight decay 0.0005; the learning rate decays according to an annealing schedule from 0.1 to 0. A comprehensive description of the experimental setup for the benchmark can be found in the paper which introduces it [8]. Owing to the three random seeds, there is correspondingly three sets of metrics for each epoch scenario. For all aspects of our study (search space analysis; LON construction; optimisation algorithm runs) we used the averaged metrics for a given neural architecture. This helps to de-noise the searches, and we choose to do this in response to findings from previous literature indicating that doing so reduces the difficulty of the landscape [23, 38].

4.1 LON Construction

For constructing the LONs, fitness is the averaged *validation* accuracy in the 90-epoch scenario. A hundred independent runs of ILS are logged to collect nodes and edges for a LON. ILS combines local search with random perturbations. As previously mentioned, local search is competitive in NAS as a domain [7, 38] and ILS itself has recently been proposed and demonstrated as promising [23]. Each of the runs terminates after 100 iterations with no improvement in local optimum quality. The local search uses a first improvement pivot rule and changes one component of a solution to another component randomly as mutation (1-change). Perturbation is the 2-change operator. Only improving or equal local optima are accepted during the search, although deteriorating connections between local optima are also logged.

4.2 NAS Algorithm Performance

According to recent literature [7, 23, 37, 38] ILS and other local search-based methods are highly competitive in neural architecture search. We consider two versions of the ILS: `ILS-shuffle` where the values for the 1-change operator are explored in random order and `ILS-ordered` where the 1-change operator uses insights from the search space and landscape analysis we conducted. We found that, generally, high-channel configurations were associated with high accuracy. In view of this, for `ILS-ordered` we systematically explore neighbours using the following ordering of the channel dimensions: 64, 56, 48, 40, 32, 24, 16, 8. Both ILS algorithms use the same design and parameters as the LON construction process detailed in Section 4.1. We contrast the ILS against the following NAS methods, as implemented in [8].

- **Random search** (RANDOM) [39]. This serves as the baseline. It draws sizes at random and returns the best found.
- **Regularised evolution** (REA) [25]. 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 sizes. This serves as a mechanism to handle the noisy performance estimation.
- **Reinforcement learning** (REINFORCE) [41]. This approach frames NAS as a reinforcement learning problem. The generation of solution corresponds to the agent's actions, with the action space identical to the search space. The agent's reward is based on an estimate of the network performance on unseen data.

When running the NAS optimisation algorithms (including `ILS-shuffle` and `ILS-ordered`) we follow the general procedure and parameters provided for experiments in the NATS-Bench proposal article [8], although we opt to update the process slightly by averaging model metrics over the three available seeds (for network weight instantiation). The rest mirrors the setup from the article: searches are directed based on mean validation accuracy using 12 epochs; the mean validation and test accuracy for the obtained architectures in the 90-epoch situation are then extracted. The training time budgets for cifar10, cifar100 and ImageNet datasets are 20 000, 40 000, and 60 000 seconds respectively. Algorithms terminate after the expenditure of this budget, and each algorithm is executed 30 times per dataset.

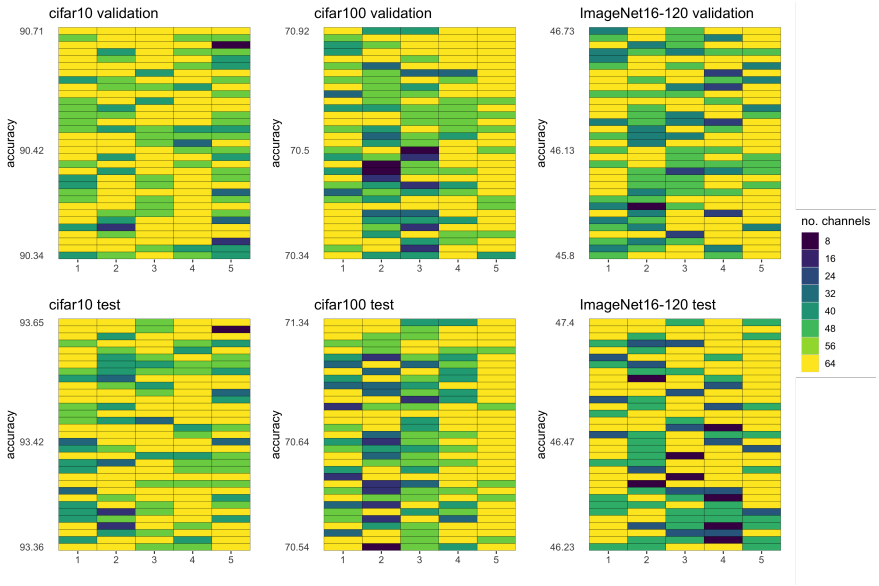


Figure 2: Genotype maps of the best 0.1% performing architectures for all datasets, sorted according to validation accuracy (top row of plots) and test accuracy (bottom row), when trained using 90 epochs. Each horizontal row in the plots visualises an architecture. Individual boxes indicate, by colour, the channel configuration at that cell position in the neural network. In this way, the x-axis represents depth (in cells) of the network

5 RESULTS

5.1 Genotype Maps

Figure 2 shows genotype maps for architecture candidates which have fitness in the top 0.1% of the search space. In each plot, the x-axis is the sequential position of a block in the neural architecture and the y-axis is the validation (top row) or test (bottom row) accuracy. Colours indicate the number of channels present at each position. As shown in the legend, darker colours represent less channels and lighter colours indicate more. Each row in a plot indicates an architecture candidate by its channel configuration.

Surveying the Figure, it can immediately be observed that light colours (high number of channels) are markedly more prevalent than the darker colours (less channels). This is particularly the case for the cifar10 dataset in the two leftmost graphs. Architectures for the other two datasets more commonly involve lower-channel blocks. An interesting trend across all three datasets is the frequency with which the highest number of channels — 64 — is involved in these elite-ranking architectures. This seems to be in opposition to the commonly-used "pyramid" approach to channel configuration [10]; this was also noted in the paper which originally proposed the benchmark we use here [8]. We acknowledge that designing a neural network with high channels throughout brings the inevitable disadvantage of increased computational expense. Notice from the cifar10 figures within the rows containing dark blue at position 5 that having a low-channel final block *can* be associated with high accuracy if the other blocks are high-channel.

The y-axis captures accuracy, so the optimal architectures are at the top of each plot. We note with intrigue (although not surprise) the fact that the optimal architectures according to the validation accuracy are not optimal when ranked using the test accuracy. This can be seen by comparing the top row (that is, horizontal block consisting of five components) in each of the upper three plots with the top row in each of the bottom three plots.

Figure 3 visualises the genotypes of architecture candidates within the lowest 0.1% of fitness. The layout is exactly the same as Figure 2. Notice, however, the difference in y-axis accuracy values: they are noticeably lower than in the first figure.

Considering the plots, the ubiquity of the low-channel genes (8, 16, and 24) is evident by the amount of dark blue. When higher-channel genes do appear here, it is typically early in the network — for example, blocks one or two. The final block at position 5 in the genotype almost always has the lowest number of channels for this set of architectures.

If we contrast the ranges on the y-axes in this set of plots with those of Figure 2, it can be stated with confidence that the channel configuration can have a dramatic effect on accuracy of the neural networks on these datasets — recall that all other hyperparameters are fixed and only the channel numbers vary.

5.2 Low-Dimensional Euclidean Embedding

Combinatorial search spaces can be visualised using *Low-Dimensional Euclidean Embedding* (LDEE) [18]. LDEE employs *t-Distributed Stochastic Neighbor Embedding* (t-SNE) to map genotypes into Euclidean space while maintaining spatial relationships between them, and then uses a *vacuum embedding* method to decide the layout for the solutions in a 2-dimensional grid.

We noticed while visualising the channel configuration spaces using LDEE that some genes in the genotypes are more correlated to accuracy than others. For cifar10, the third gene is the most

correlated; for the other two datasets, the final (fifth) gene is most correlated. The decision was made to weight the Euclidean distance calculations in light of these important genes. Specifically, we added a weight vector to the Euclidean distance calculations: each gene is assigned a weight of 1 except the most-correlated one with respect to validation accuracy — this gene is given a weight of 2. This should have the effect of clustering together solutions which have the same value of the most-correlated gene. Figure 4 shows LDEE plots for the search space of each combination of dataset (cifar10, cifar100, ImageNet16-120) and data split (validation, test) pair. The x and y axes form the grid on which the Euclidean-transformed solutions are visualised, and the heat (colour) of points is indication of the validation or test accuracy.

Notice that the validation search spaces and the test spaces appear, by this method, to be very similar; this can be observed by comparing the first row of graphs with their counterparts on the lower row. This holds true for each of the three datasets: cifar10, cifar100, and ImageNet16-120.

In order to make the differences more pronounced, we plotted (test - validation) accuracy using the same coordinate mapping (4g-4i). Interestingly, for cifar100 (and, to some extent, for cifar10) test accuracy appears to be *better* than validation accuracy where the accuracy values are high and, conversely, to be *worse* than validation accuracy where the accuracy values are low (blue areas in the third row correspond to blue areas in the top two rows). This is not the case for ImageNet16-120 (4i). Another impression from the Figure is that there are groups of solutions which are close both in distance and in fitness. This can be seen in the solid patches of red, and is particularly true with respect to cifar100 (4b and 4e). Recalling that the placement of solutions is biased towards clustering those with the same value for the gene most correlated to validation accuracy, we stipulate that there are obviously groups of high-quality solutions which are "similar" in both fitness and distance. We can see that the ImageNet16-120 search spaces (Figures 4c and 4f) indicate the presence of more lower-quality solutions, and also that these are more spread out in terms of genotype when compared to cifar10 and cifar100. An interesting note is that high-quality solutions for cifar100 (Figures 4b and 4e) are clustered (notice the deep red patches), and that this is also somewhat true for low-quality genotypes (the blue pattern); additionally, these two groups of solutions are quite nearby to one another. This implies that the best genotypes may not be so different in comparison to the worst ones.

5.3 Local Optima Networks

Table 1 shows metrics for the extracted monotonic local optima networks (MLONs). Each column contains values which relate to a dataset and a data split type (validation or test). The provided metrics are: *nodes*: number of nodes (local optima); *global strength*: normalised weighted in-degree of the global optimum (or optima); *path to optimum*: average path length, in LON edges, from source nodes to the global optimum (or optima); and *difference to optimum*: average accuracy difference between nodes and the global optimum (or optima). Notice from the first row of Table 1 that the number of local optima is low. This indicates relatively straightforward landscapes which should be well-suited to local search. For the cifar datasets, paths to a global optimum (*path to optimum*) are longer in the test set than for validation. This can be observed in the third row by comparing columns 2 and 4 with 1 and 3, respectively.

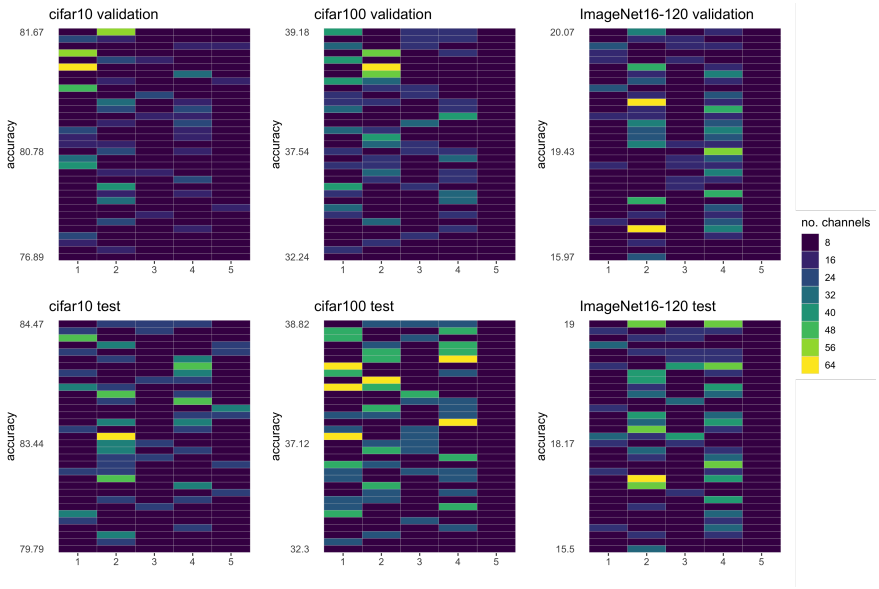
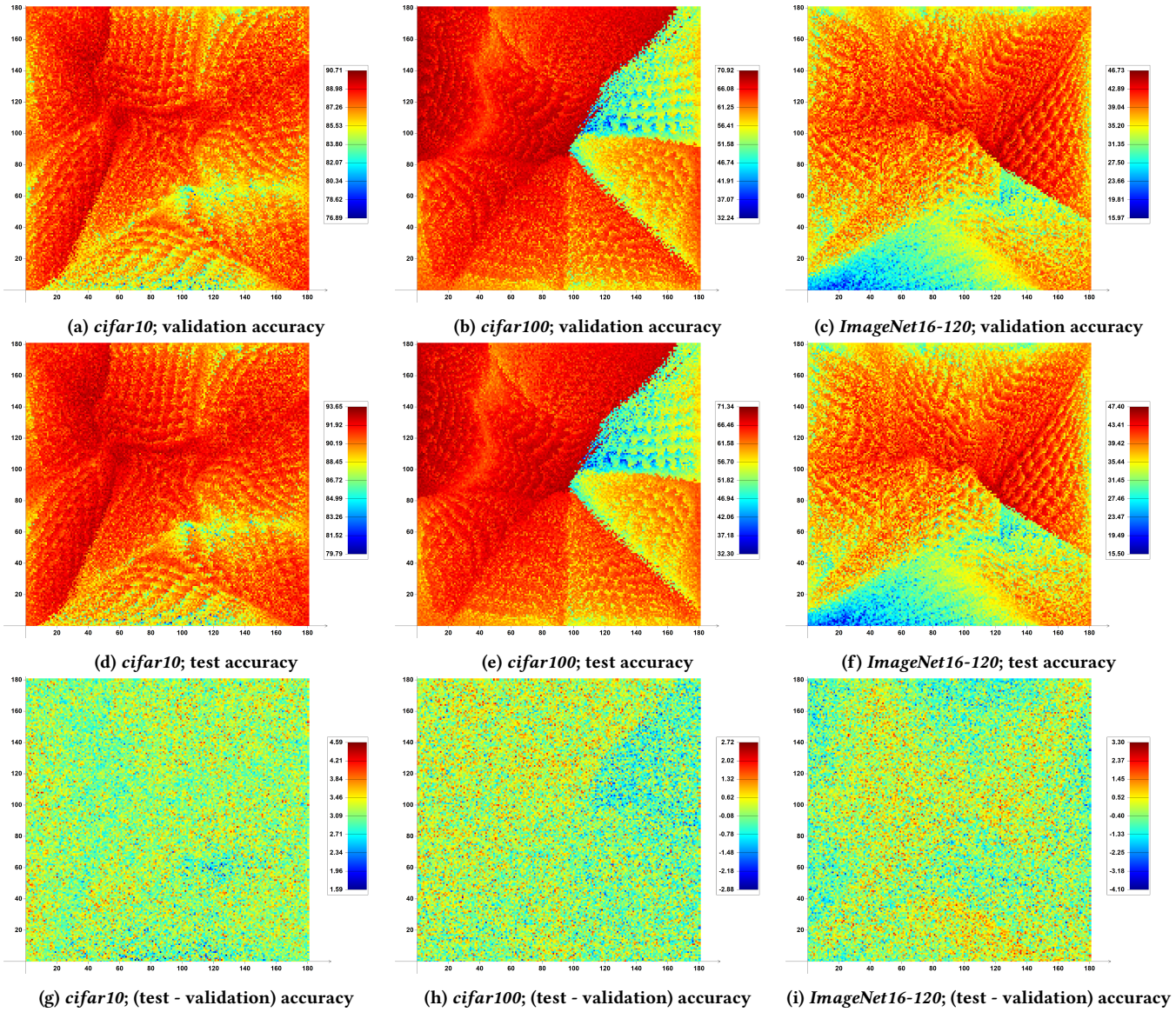


Figure 3: Genotype maps of the worst 0.1% performing architectures for all datasets, sorted according to validation accuracy (top row of plots) and test accuracy (bottom row), when trained using 90 epochs. Each horizontal row in the plots visualises an architecture. Individual boxes indicate, by colour, the channel configuration at that cell position in the neural network. In this way, the x-axis represents depth (in cells) of the network



(a) *cifar10*; validation accuracy (b) *cifar100*; validation accuracy (c) *ImageNet16-120*; validation accuracy

(d) *cifar10*; test accuracy (e) *cifar100*; test accuracy (f) *ImageNet16-120*; test accuracy

(g) *cifar10*; (test - validation) accuracy (h) *cifar100*; (test - validation) accuracy (i) *ImageNet16-120*; (test - validation) accuracy

Figure 4: Low-dimensional Euclidean embedding of the NATS channel configuration search spaces; architectures trained with 90 epochs. The x and y axes form the grid on which the Euclidean-transformed solutions are visualised, and the heat (colour) of points is indication of the accuracy

Table 1: MLON metrics for all datasets

dataset	cifar10		cifar100		ImageNet16	
data split	valid.	test	valid.	test	valid.	test
nodes	25	33	34	31	28	23
global strength	0.42	0.56	0.26	0.26	0.39	0.35
path to optimum	1.57	2.00	1.89	2.00	1.93	1.71
difference to optimum	0.32	0.29	0.60	0.84	0.92	1.13

In the case of cifar100 and ImageNet16, *difference to optimum* is larger for the test set than for validation. We note also that for all datasets and data splits, the *global strength* is between 0.26 and 0.56. This means that the global optima have a relatively large share of the incoming edges in the LONs.

Figure 5 shows visualisations for the constructed local optima networks. Only neutral or improving transitions between local optima are shown; it follows that these are *monotonic* LONs, or MLONs. Each column in the plot matrix is for a dataset (e.g. cifar10); the top row of graphs is based on validation accuracy, while the bottom row are for test accuracy. That distinction refers to the way the fitness landscape was searched and the LON consequently extracted: for example, validation accuracy was used as the fitness

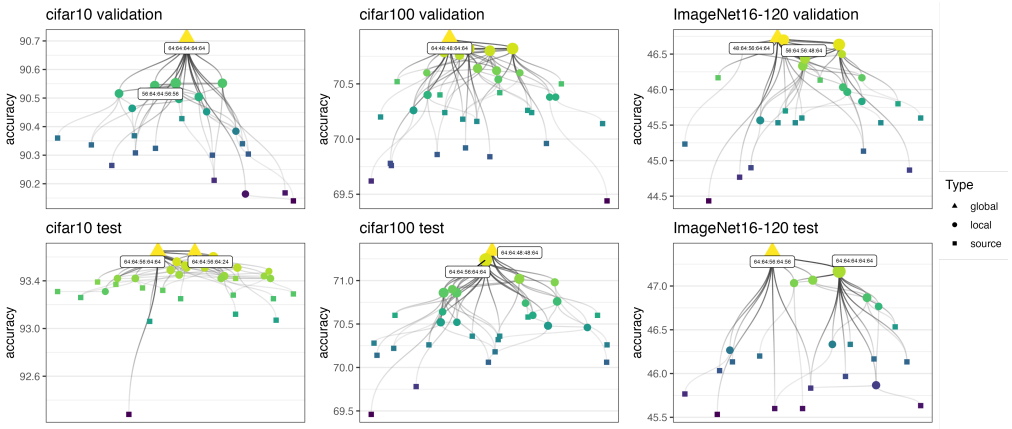


Figure 5: Local optima networks constructed using the validation accuracy (top row) and test accuracy (bottom row) of the architecture (using 90 training epochs) as fitness

function for the networks shown in the validation graphs. In the Figure, node colour indicates fitness quality: darker colours are lower fitness. The graph layout uses fitness as the y-coordinate, while the x-coordinate follows a force-directed graph layout. Nodes with fitness within the top 5% have their genotype shown as an annotation. The size of nodes is proportional to their incoming weighted degree (strength). Global optima are triangular in shape; all other nodes are circles. Only improving or equal transitions between local optima are plotted.

Notice from the Figure that all except one of the LONs have a single global optimum. Note also the pattern of local optima organising into shapes reminiscent of an inverted version of the "big valley" landscape structure which has been observed in other combinatorial problems [2, 3]. Landscape anatomy which exhibits a big valley or "central massif" layout is, intuitively, particularly well-suited to iterated local search because it is designed to ascend through layers of local optima and iteratively drive itself to the global optimum. It appears that there are straightforward heuristic connections between fitness levels in these landscapes (notice the edges between nodes of different colour). We direct your attention to the task of comparing the validation accuracy LONs — top row — with their test accuracy analogues along the bottom row. The highest-quality solutions (the ones with the genotype annotations) are never shared between the validation and test LONs. Of all the elite genotypes shown across the six images, only one appears more than once: the architecture with the highest-channel gene at every position. An unexpected result is that for all the networks, the highest-channel gene is at the first position in eight of the 11 annotated elite local optima. This seems to be in opposition to approaches to channel configuration such as doubling at downsampling locations [28], pyramid structure [10], linear increase [11], and compound scaling [30]. Less surprisingly, eight of the 11 have the highest-channel gene at the final position.

We find the genotypic composition of the two global optima for cifar10 according to test set accuracy (the rightmost lower plot) to be curious. They are almost identical except for the gene at the final block. One has 24 channels at this position and the other has 64, but they have identical fitness. The intrigue of this is compounded by the fact that architectures with the same genotype as the global optima but with channel sizes between 24 and 64 at the final block do not share in the optimal fitness.

5.4 Algorithm Performance

For algorithm performance, we consider test set accuracy only; this is in the interest of space, and also because the test accuracy (and not validation accuracy) is the "real" performance of the NAS-optimised architecture. Figure 6 shows, for the three datasets, the test set accuracy of the final obtained architecture from runs. Each box contains 30 runs of the algorithm labelled on the x-axis. In terms of the medians — indicated with thick horizontal line through the boxes — *ILS-ordered* (the left-most box) has the highest for all three datasets. The conventional ILS (*ILS-shuffle*) has lower medians than the directed version. For cifar10 (the left-most graph) the median is approximately equal with REA; *REINFORCE* is above them, with *RANDOM* the lowest. In the case of cifar100, shown in the middle plot, *REA* has a higher median than *ILS-shuffle*; *RANDOM* and *REINFORCE* have lower lines. For the ImageNet dataset, *ILS-shuffle* has a median approximately equal to that of *REINFORCE*; *REA* is higher than them, with *RANDOM* the lowest. The ILS variants appear to exhibit quite a large variation in behavior according to different runs: for cifar100, *ILS-ordered* has several lower-accuracy outliers. On ImageNet, the two ILS algorithms have have the widest distributions from the five.

Figure 7 shows convergence plots for the five NAS algorithms. Wall clock time is on the x-axes, with the neural network accuracy on the test set of the current solution on the y-axis. Each line is the mean over 30 runs of a NAS algorithm, as indicated in the legend. Surveying the cifar10 plot in Figure 7, it can be observed that the non-ILS NAS algorithms initially have a steeper climb in accuracy over the ILS variants. This can be seen by comparing the three dotted lines with the two solid ones. By around $5x10^3$ seconds, however, *ILS-ordered* has outpaced all other algorithms. The traditional ILS (*ILS-shuffle*) has the slowest climb of all the algorithms, although it does end up with higher accuracy than *REINFORCE* and

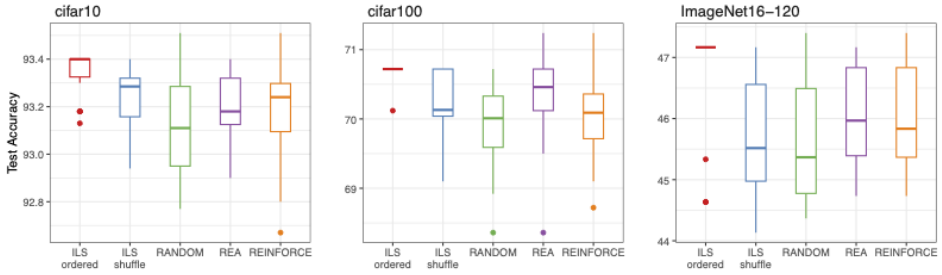


Figure 6: Obtained accuracy distributions from NAS algorithms across the datasets

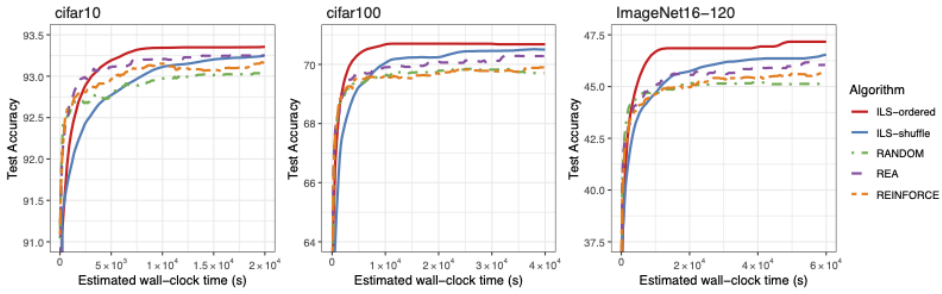


Figure 7: Test accuracy convergence curves for NAS algorithms across the datasets

RANDOM. For cifar100 and ImageNet — the middle and righthand plots — *ILS-ordered* initally climbs with roughly equal steepness to the non-ILS NAS approaches; it then begins to out-perform them well before $1x10^4$ seconds have elapsed. The *ILS-shuffle* variant again has the slowest ascent in accuracy, but overtakes the non-ILS algorithms at around $1x10^4$ and $2x10^4$, respectively.

The results in this Section indicate that iterated local search variants are competitive in a channel configuration NAS space. The *ILS-ordered* seems particularly effective. This explored the neighbourhood in descending order for the channel dimension — a decision we made after noticing that the best architectures typically contain high channel numbers. Of course, *ILS-ordered* has an advantage not afforded to the others: we embedded knowledge of the search space within it by testing large channel numbers first during search. Nevertheless, it has value. We argue that this value lies both in demonstrating the benefit of incorporating search space information into search algorithms, and also in emphasising the result that — at least for this search space and these three datasets — large numbers of channels *throughout* neural networks are advantageous. That goes against some observations in the literature that compression with autoencoders can help with performance in CNNs [20] which has also been noted for the cifar datasets specifically [40]; it could be that high channel numbers perform better within the specific environment of the other components comprising the architecture used — which are kept constant while channels are varied. We also note that the studied image datasets are diverse [15] and have many possible classes (between 10 and 120), which could possibly benefit from high channel dimension to capture the feature nuances needed to differentiate between classes.

6 CONCLUSIONS

We have conducted a search space, fitness landscape, and algorithm performance analysis on a tabular neural architecture search benchmark: the *size* search space from NATS-Bench [8], which considers the optimisation of channel configuration within a neural architecture. The results show that a high number of channels throughout a convolutional neural network (CNN) may be beneficial. This is in contrast to some commonly-adopted configuration styles for channel dimension. The search space analysis brought visual clarity and showed that there are many solutions (architectures) which are both similar in fitness (network accuracy) and in close proximity to each other. We found that the channel fitness landscapes were straightforward and well-suited to local search approaches, particularly iterated local search (ILS). This was then evidenced by experiments which showed that ILS is competitive when it comes to searching the channel configuration space. Going forward, we would like to analyse other neural architecture search spaces; in particular, modelling training accuracy fitness landscapes and comparing them to validation and test landscapes. In this way, insight into overfitting might be obtained. The local optima networks from this work are publicly available[2].

REFERENCES

[1] Marco Baioletti, Alfredo Milani, Valentino Santucci, and Marco Tomassini. 2019. Search moves in the local optima networks of permutation spaces: the QAP case. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. 1535–1542.

[2] Marco Baioletti and Valentino Santucci. 2017. Fitness landscape analysis of the permutation flowshop scheduling problem with total flow time criterion. In

[2] github.com/sarahlouisethomson/channel-configuration-local-optima-networks/

International Conference on Computational Science and Its Applications. Springer, 705–716.

[3] Kenneth Dean Boese. 1995. *Cost versus distance in the traveling salesman problem*. Citeseer.

[4] Wojciech Bożejko, Andrzej Gnatowski, Teodor Niżyński, Michael Affenzeller, and Andreas Beham. 2018. Local optima networks in solving algorithm selection problem for tsp. In *International Conference on Dependability and Complex Systems*. Springer, 83–93.

[5] Liang-Chieh Chen, Maxwell Collins, Yukun Zhu, George Papandreou, Barret Zoph, Florian Schroff, Hartwig Adam, and Jon Shlens. 2018. Searching for efficient multi-scale architectures for dense image prediction. *Advances in neural information processing systems* 31 (2018).

[6] Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. 2017. A downsampled variant of imagenet as an alternative to the cifar datasets. *arXiv preprint arXiv:1707.08819* (2017).

[7] Tom Den Ottelander, Arkadiy Dushatskiy, Marco Virgolin, and Peter AN Bosman. 2021. Local search is a remarkably strong baseline for neural architecture search. In *International Conference on Evolutionary Multi-Criterion Optimization*. Springer, 465–479.

[8] Xuanyi Dong, Lu Liu, Katarzyna Musial, and Bogdan Gabrys. 2021. NATS-Bench: Benchmarking NAS Algorithms for Architecture Topology and Size. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021).

[9] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. 2019. Neural architecture search: A survey. *The Journal of Machine Learning Research* 20, 1 (2019), 1997–2017.

[10] Dongyoon Han, Jiwhan Kim, and Junmo Kim. 2017. Deep pyramidal residual networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 5927–5935.

[11] Dongyoon Han, Sangdoo Yun, Byeongho Heo, and YoungJoon Yoo. 2021. Rethinking channel dimensions for efficient model design. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*. 732–741.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[13] Mahdi S Hosseini, Jia Shu Zhang, Zhe Liu, Andre Fu, Jingxuan Su, Mathieu Tuli, and Konstantinos N Plataniotis. 2021. CONet: Channel Optimization for Convolutional Neural Networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 326–335.

[14] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).

[15] Kevin Alexander Laube, Maximus Mutschler, and Andreas Zell. 2022. What to expect of hardware metric predictors in NAS. In *International Conference on Automated Machine Learning*. PMLR, 13–1.

[16] Liam Li and Ameet Talwalkar. 2020. Random search and reproducibility for neural architecture search. In *Uncertainty in artificial intelligence*. PMLR, 367–377.

[17] Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2018. DARTS: Differentiable Architecture Search. In *International Conference on Learning Representations*.

[18] Krzysztof Michalak. 2019. Low-dimensional euclidean embedding for visualization of search spaces in combinatorial optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. 27–28.

[19] Werner Mostert, Katherine M Malan, Gabriela Ochoa, and Andries P Engelbrecht. 2019. Insights into the feature selection problem using local optima networks. In *European Conference on Evolutionary Computation in Combinatorial Optimization (Part of EvoStar)*. Springer, 147–162.

[20] Nauman Munir, Jinhyun Park, Hak-Joon Kim, Sung-Jin Song, and Sung-Sik Kang. 2020. Performance enhancement of convolutional neural network for ultrasonic flaw classification by adopting autoencoder. *NDT & E International* 111 (2020), 102218.

[21] Matheus Nunes, Paulo M Fraga, and Gisele L Pappa. 2021. Fitness landscape analysis of graph neural network architecture search spaces. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 876–884.

[22] Gabriela Ochoa, Marco Tomassini, Sebástien Vérel, and Christian Darabos. 2008. A study of NK landscapes' basins and local optima networks. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*. 555–562.

[23] Gabriela Ochoa and Nadarajen Veerapen. 2022. Neural Architecture Search: A Visual Analysis. In *Parallel Problem Solving from Nature, PPSN 2022*. Springer, 603–615.

[24] Isak Potgieter, Christopher W Cleghorn, and Anna S Bosman. 2022. A Local Optima Network Analysis of the Feedforward Neural Architecture Space. In *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–8.

[25] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. 2019. Regularized Evolution for Image Classifier Architecture Search. In *AAAI Conference on Artificial Intelligence, AAAI*. AAAI Press, 4780–4789.

[26] Nuno M Rodrigues, Katherine M Malan, Gabriela Ochoa, Leonardo Vanneschi, and Sara Silva. 2022. Fitness landscape analysis of convolutional neural network architectures for image classification. *Information Sciences* 609 (2022), 711–726.

[27] Xian Shi, Pan Zhou, Wei Chen, and Lei Xie. 2021. Darts-conformer: Towards efficient gradient-based neural architecture search for end-to-end asr. *arXiv preprint arXiv:2104.02868* (2021).

[28] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[29] Peter F. Stadler. 2002. Fitness landscapes. *Biological Evolution and Statistical Physics. Lecture Notes in Physics* 585 (2002), 183–204.

[30] Mingxing Tan and Quoc Le. 2019. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*. PMLR, 6105–6114.

[31] Matheus Cândido Teixeira and Gisele Lobo Pappa. 2022. Analysis of Neutrality of AutoML Search Spaces with Local Optima Networks. In *Intelligent Systems*, João Carlos Xavier-Junior and Ricardo Araújo Rios (Eds.). Springer International Publishing, Cham, 473–487.

[32] Matheus C Teixeira and Gisele L Pappa. 2022. Understanding AutoML search spaces with local optima networks. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 449–457.

[33] Chakkrit Termritthikun, Yeshi Jamtsho, Jirarat Ieamsaard, Paisarn Muneesawang, and Ivan Lee. 2021. EEEA-Net: An early exit evolutionary neural architecture search. *Engineering Applications of Artificial Intelligence* 104 (2021), 104397.

[34] Sébastien Verel, Fabio Daolio, Gabriela Ochoa, and Marco Tomassini. 2011. Local optima networks with escape edges. In *International Conference on Artificial Evolution (Evolution Artificielle)*. Springer, 49–60.

[35] Alvin Wan, Xiaoliang Dai, Peizhao Zhang, Zijian He, Yuandong Tian, Saining Xie, Bichen Wu, Matthew Yu, Tao Xu, Kan Chen, et al. 2020. Fbnetv2: Differentiable neural architecture search for spatial and channel dimensions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 12965–12974.

[36] Yi Ru Wang, Samir Khaki, Weihang Zheng, Mahdi S Hosseini, and Konstantinos N Plataniotis. 2021. CONetV2: Efficient Auto-Channel Size Optimization for CNNs. In *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 998–1003.

[37] Colin White, Sam Nolen, and Yash Savani. 2020. Local search is state of the art for NAS benchmarks. *arXiv preprint arXiv:2005.02960* (2020).

[38] Colin White, Sam Nolen, and Yash Savani. 2021. Exploring the loss landscape in neural architecture search. In *Uncertainty in Artificial Intelligence*. PMLR, 654–664.

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

[40] Junbo Zhao, Michael Mathieu, Ross Goroshin, and Yann Lecun. 2015. Stacked what-where auto-encoders. *arXiv preprint arXiv:1506.02351* (2015).

[41] Barret Zoph and Quoc V. Le. 2017. Neural Architecture Search with Reinforcement Learning. In *Conference on Learning Representations, ICLR*.