MAKING DIFFERENTIABLE ARCHITECTURE SEARCH LESS LOCAL

Erik Bodin *
Spotify & University of Bristol

Federico Tomasi
Spotify

Zhenwen Dai
Spotify

ABSTRACT

Neural architecture search (NAS) is a recent methodology for automating the design of neural network architectures. Differentiable neural architecture search (DARTS) is a promising NAS approach that dramatically increases search efficiency. However, it has been shown to suffer from performance collapse, where the search often leads to detrimental architectures. Many recent works try to address this issue of DARTS by identifying indicators for early stopping, regularising the search objective to reduce the dominance of some operations, or changing the parameterisation of the search problem. In this work, we hypothesise that performance collapses can arise from poor local optima around typical initial architectures and weights. We address this issue by developing a more global optimisation scheme that is able to better explore the space without changing the DARTS problem formulation. Our experiments show that our changes in the search algorithm allow the discovery of architectures with both better test performance and fewer parameters.

1 INTRODUCTION

Designing neural network architectures improving upon the state-of-the-art requires a substantial effort of human experts. Automating the discovery of neural network architectures by formulating it as a search problem allows us to minimise the human time spent on the search process. Due to the large combinatorial search space of possible neural network architectures, early methods (19; 20; 15) were computationally very demanding, often requiring thousands of GPU days of computation for search, giving rise to high costs. Many neural architecture search (NAS) works have been focused on reducing the computational cost, (10; 1; 6; 14; 2). Among them, Liu et al. (11) proposed a particularly efficient approach by making the search space of architectures differentiable (known as DARTS), which reduced the search cost by several orders of magnitude.

Although being efficient, recent works have shown that DARTS suffers from performance collapse due to the search favouring parameter-less operations like skip connections (5; 18). Many follow-up works have been proposed to fix the performance collapse problem by identifying indicators for early stopping, regularising the search objective to reduce skip connections, or changing the search problem's parameterisation. Chen & Hsieh (3) and Zela et al. (18) proposed to stabilise the search process by regularising the Hessian of the search objective. Chu et al. (5) avoid the advantage of the skip connections in the search phrase by replacing the softmax with the sigmoid function for the switch among edges. Chu et al. (4) avoided the dominance of skip connections by changing the parameterisation of the search space.

In this paper, we hypothesise that performance collapses and the dominance of some operations observed in several works are the consequence of the existence of poor local optima around typical initial architectures and weights. Instead of identifying indicators for early stopping or tweaking the search space's parameterisation, we propose that a more global optimisation scheme should be developed that allows us to avoid bad local optima and better explore the objective over the search space to discover better solutions. We show in experiments that even a simple scheme to make the optimisation more global reduces detrimental behaviours significantly. Importantly, it removes the need to stop the search early in order to avoid reaching detrimental or invalid solutions.

*This work was done during an internship at Spotify.

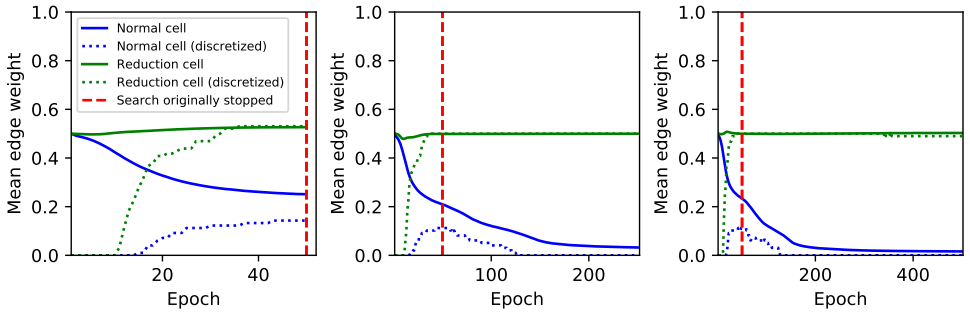

Figure 1: The FairDARTS search needs to be stopped early to avoid the normal cell having no remaining active edges following discretisation. Shown is the mean cell weights (following the sigmoid activation function) for the normal and reduction cell, respectively, for three runs on different budgets. The same issue persisted on every run. The discretisation threshold used is 0.85, but the issue applies to any thresholding rule as all normal cell edge weights tend to zero.

We show that, after searching until convergence, our method can find architectures with better test performance and fewer parameters.

2 Empirical diagnosis

FairDARTS is a state-of-the-art DARTS variant presented in (5). The method includes structural changes to the original DARTS search space, allowing multiple edges per pair of nodes in the searched cell structure. This was implemented by switching the softmax activation function on the weights to a sigmoid function. Another change was adding a regularisation term (a 'zero-one loss'), encouraging the continuous edge activations to better approximate the binary discretisation of the cell happening after the search phase.

FairDARTS improved upon the DARTS method, reducing the issue with skip connections dominating, and ultimately lead to better architectures in terms of final test performance compared to DARTS and other variants. However, as we will demonstrate, another similar issue presents itself (still) in the FairDARTS method. What happens is that one of the searched cell types, the 'reduction cell', dominates the other (the 'normal cell'), to the detriment of test performance and reliability. In particular, if searching for longer than a small fraction of as many epochs later used for the training in the final evaluation phase, the test performance decays, and the architectures produced quickly become invalid. We illustrate this in Figure 1 (the experiments were conducted on CIFAR-10 using the implementation and the setup as in (5)). We note that the edge weights associated with the normal cell decrease monotonically after a certain number of epochs. If the search is not stopped early, at the right time, the weights of all operations in the normal cell become zero, resulting in no activations being able to propagate through the cell following discretisation. In (5) the search was stopped after only $1/12$ of the number of epochs later used to train the final architecture.

Being forced to stop the search early to avoid detrimental architectures has two negative consequences. Firstly, the right time to stop the search becomes an additional hyperparameter to tune to obtain good performance. Secondly, it can inhibit better architectures to be found by searching for longer. Both of these aspects are important for building a reliable NAS method for a wide range of datasets and tasks.

3 Global Optimisation for Differentiable NAS

DARTS (11), as similar to prior works (20; 15; 10), searches for a *cell*, which is used as a building block for the final architecture. The cell constitutes a directed acyclic graph of N nodes. Each node x represents a latent representation and each directed edge (i, j) represents an operation $o_{i,j}$. A node depends on all of its predecessors as $x_j = \sum_{i<j} o_{i,j}(x_i)$. Let $\mathcal{O}$ be the set of candidate operations (e.g., convolution, max pooling, skip connection) available for each edge (i, j). FairDARTS (5) defines the choice of operations for an edge as $\bar{o}_{i,j}(x) = \sum_{o\in\mathcal{O}} \sigma(\alpha_{o_i,j})o(x)$, where $\sigma(\cdot)$ is the

Algorithm 1: Doubly Stochastic Coordinate Descent (global step)

Input: Function f defined over $\mathcal{X}$, proposal distribution q, initial $\boldsymbol{x}_{\text{best}}, y_{\text{best}}$
Output: $\boldsymbol{x}_{\text{best}}, y_{\text{best}}$

```
while budget_remaining do
    d = sample_a_random_dimension();
    x ~ q(x|x_best[d], d);
    y = f(x);
    if y < y_best then
        x_best = x, y_best = y;
    end
end
```

sigmoid function. This allows multiple operations per edge to be chosen simultaneously. If no operations are active for a given edge this constitute the zero operation (18).

Let $\boldsymbol{\alpha}$ be the concatenated vector of all operation edge weights representing the architecture, in which the ones associated with the normal cell and the reduction cell are denoted by $\boldsymbol{\alpha}_{\text{normal}}$ and $\boldsymbol{\alpha}_{\text{reduction}}$ respectively, i.e., $\boldsymbol{\alpha} = (\boldsymbol{\alpha}_{\text{normal}}, \boldsymbol{\alpha}_{\text{reduction}})$. Let $\boldsymbol{w}$ be the concatenated neural network parameters associated with all operations, where similarly $\boldsymbol{w} = (\boldsymbol{w}_{\text{normal}}, \boldsymbol{w}_{\text{reduction}})$.

The architecture search problem in DARTS can be stated as a bilevel optimisation problem:

$$\underset{\boldsymbol{\alpha}}{\text{minimize}} \quad \mathcal{L}_{\text{val}}(\boldsymbol{\alpha}, \boldsymbol{w}^*) \tag{1a}$$

$$\text{subject to} \quad \boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \mathcal{L}_{\text{train}}(\boldsymbol{\alpha}, \boldsymbol{w}), \tag{1b}$$

where $\mathcal{L}_{\text{val}}$ and $\mathcal{L}_{\text{train}}$ are the validation loss and training loss, respectively. DARTS approximates the gradient as $\nabla_{\boldsymbol{\alpha}} \mathcal{L}_{\text{val}}(\boldsymbol{\alpha}, \boldsymbol{w}^*) \approx \nabla_{\boldsymbol{\alpha}} \mathcal{L}_{\text{val}}(\boldsymbol{\alpha}, \boldsymbol{w} - \xi \nabla_{\boldsymbol{w}} \mathcal{L}_{\text{train}}(\boldsymbol{\alpha}, \boldsymbol{w}))$, where ξ is the learning rate for the inner optimisation, and gradient-based local optimisation is performed in alternating steps.

Global Optimisation Scheme. We hypothesise that the usage of local search for the $\boldsymbol{\alpha}$ weights in the DARTS' approximation to the bilevel optimisation problem leads to convergence to local optima associated with performance collapse. We propose an optimisation scheme that makes the search for the $\boldsymbol{\alpha}$ weights "more global" in the sense that local valleys can be escaped using a complementary global optimisation routine.

Our optimisation scheme consists of two types of steps: *local* and *global* steps. The algorithm alternates between taking local and global steps, similar to basin-hopping (17) for global optimisation. A local step is a step in the gradient direction, the same as in (11). A global step is taken according to the proposed doubly stochastic coordinate descent (DSCD) algorithm. DSCD follows the stochastic coordinate descent approach (13) and draws a random dimension of which to consider next. In DSCD, only a single (global) step is taken each time a dimension is sampled, and the step is stochastic, where the new position (for the sampled dimension) is a sample from a proposal distribution. The sample is accepted as the new position only if the objective improves upon the best lost within the last K steps [1]. The global step is global in the sense that there does not need to be a monotonically improving trajectory between any two positions (in terms of the loss surface), thus allowing 'jumps' between valleys [2] . The outline of DSCD is shown in Algorithm 1. We propose an annealing scheme for the proposal distribution. The proposal distribution is parameterised as a Beta distribution over a bounded space. At the beginning of the optimisation, the proposal distribution is uniform, and it slowly moves towards a Dirac delta centred at the current position, thus becoming increasingly local as the search progresses. The details of the annealing scheme for the proposal can be found in the appendix. We alternate between taking local and global steps when the following is both true; T consecutive steps of the same type has been taken, and the loss did not improve from the last step to the next. In all experiments, we set $T = 50$, and noticed little to no importance of tuning this parameter. In the appendix we assess the benefit of DSCD on multimodal functions.

[1] In practice we used $K = 1000$. Only considering the best loss within a (relatively large) window, rather than the historical best, we noted was helpful to be robust to outlier losses as a result of the mini-batching.

[2] Strictly this does not need to be true for *stochastic gradient descent* either, but in SGD it is still statistically unlikely to take steps in non-monotonically improving directions.

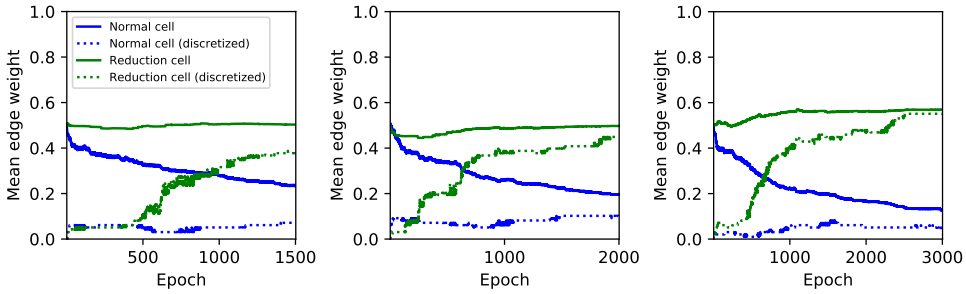

Figure 2: Using the new optimisation scheme the architecture does no longer become invalid by searching for longer. Shown is the mean cell weights (following the sigmoid activation function) for the normal and reduction cell, respectively, on three runs on different budgets.

Table 1: Comparison with FairDARTS for search and evaluation phases (accuracy in %). Split for $\mathcal{L}_{\text{train}}$ and $\mathcal{L}_{\text{val}}$ indicates accuracy measured on the training data for $\mathcal{L}_{\text{train}}$ and $\mathcal{L}_{\text{val}}$ respectively. Search Test indicates the accuracy on the hold-out set using the search network (undiscretised). Eval. Test indicates the test accuracy with the final architecture. "Invalid arch." denotes no valid final architecture after discretisation.

Method	Search Phase			Final Arch.
	Split for $\mathcal{L}_{\text{train}}$	Split for $\mathcal{L}_{\text{val}}$	Search Test	Eval. Test
FairDARTS (50)	82.02	75.61	76.15	97.36
FairDARTS (75)	87.35	78.10	78.65	97.29
FairDARTS (250)	96.95	81.55	81.52	Invalid arch.
FairDARTS (500)	99.92	83.49	83.26	Invalid arch.
FairDARTS + DSCD (1500)	100.0	83.12	83.40	97.50
FairDARTS + DSCD (2000)	100.0	84.02	84.71	97.25
FairDARTS + DSCD (3000)	100.0	85.51	85.10	96.92

4 Experiments

We previously showed that all the edge weights of normal cells $\boldsymbol{\alpha}_{\text{normal}}$ tend towards zero in FairDARTS, resulting in invalid architectures. We will now demonstrate that our optimisation scheme explores the architecture space better. As a result, it avoids invalid architectures, discovers architectures with better test performance, and converges to good solutions without early stopping. In the experiments, the same setup as in (5) is used, except for "FairDARTS + DSCD" for which we replace the local optimiser (Adam (9)) with the proposed optimisation scheme. In Figure 2 we see that the edge weights of the normal cell no longer become zero, even if searching for much longer, and the resulting architecture can be successfully discretised. The mean weights, after discretisation, slowly move towards the mean weights before discretisation. Importantly, the edges that will be kept (above the 0.85 threshold) remained the same from 1500 epochs, which is indicative of convergence.

In Table 1 we see the accuracy of the final architectures and the searches, corresponding to Figures 1 and 2. Using our optimisation scheme (DSCD), the models produced become increasingly more accurate with more search, while remaining valid. Our method using 1500 epochs for search produces a higher test accuracy during the search phase than FairDARTS, which also results in a high test accuracy with the final architecture. Despite the test accuracy of our method increasing with more search epochs, the test accuracy of the resulting final architectures decreases. We argue that this is due to the fact that the network used during the search phase is different from the network for evaluation (a network trained from scratch using the final architecture) (5). Differences between the search architecture and final architecture include discretisation, that the final architecture is larger and has auxiliary heads (5), as well as that the training paths are different (weights and architecture together versus weights only). A comparison to other DARTS variants is included in the appendix.

5 Conclusion

Neural architecture search requires three things: a space of models with good inductive biases, a loss function to assess models, and an optimisation or inference algorithm to explore the space. In this work we focused on the optimisation algorithm, and we showed that by combining gradient-based, local search with global optimisation techniques, we are able to better explore the space.

REFERENCES

[1] G. Bender, P.-J. Kindermans, B. Zoph, V. Vasudevan, and Q. Le. Understanding and simplifying one-shot architecture search. In *International Conference on Machine Learning*, pages 550–559, 2018.

[2] H. Cai, T. Chen, W. Zhang, Y. Yu, and J. Wang. Efficient architecture search by network transformation. *arXiv preprint arXiv:1707.04873*, 2017.

[3] X. Chen and C.-J. Hsieh. Stabilizing differentiable architecture search via perturbation-based regularization. In *Proceedings of the 37th International Conference on Machine Learning*, pages 1554–1565, 2020.

[4] X. Chu, X. Wang, B. Zhang, S. Lu, X. Wei, and J. Yan. {DARTS}-: Robustly stepping out of performance collapse without indicators. In *International Conference on Learning Representations*, 2021.

[5] X. Chu, T. Zhou, B. Zhang, and J. Li. Fair darts: Eliminating unfair advantages in differentiable architecture search. In *European Conference on Computer Vision*, pages 465–480. Springer, 2020.

[6] T. Elsken, J.-H. Metzen, and F. Hutter. Simple and efficient architecture search for convolutional neural networks. *arXiv preprint arXiv:1711.04528*, 2017.

[7] N. Hansen and S. Kern. Evaluating the cma evolution strategy on multimodal test functions. In *International Conference on Parallel Problem Solving from Nature*, pages 282–291. Springer, 2004.

[8] M. Jamil and X.-S. Yang. A literature survey of benchmark functions for global optimisation problems. *International Journal of Mathematical Modelling and Numerical Optimisation*, 4(2):150–194, 2013.

[9] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations, ICLR*, 2015.

[10] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy. Progressive neural architecture search. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 19–34, 2018.

[11] H. Liu, K. Simonyan, and Y. Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

[12] I. Loshchilov and F. Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[13] Y. Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

[14] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean. Efficient neural architecture search via parameter sharing. *arXiv preprint arXiv:1802.03268*, 2018.

[15] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pages 4780–4789, 2019.

[16] M. Styblinski and T.-S. Tang. Experiments in nonconvex optimization: stochastic approximation with function smoothing and simulated annealing. *Neural Networks*, 3(4):467–483, 1990.

[17] D. J. Wales and J. P. Doye. Global optimization by basin-hopping and the lowest energy structures of lennard-jones clusters containing up to 110 atoms. *The Journal of Physical Chemistry A*, 101(28):5111–5116, 1997.

[18] A. Zela, T. Elsken, T. Saikia, Y. Marrakchi, T. Brox, and F. Hutter. Understanding and robustifying differentiable architecture search. *arXiv preprint arXiv:1909.09656*, 2019.

[19] B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

[20] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018.

Appendix

A Comparison with other DARTS methods

We also compared our approach with other state-of-the-art NAS methods in the DARTS family. The results are shown in Table 2.

Table 2: Comparison of state-of-the-art NAS models on CIFAR-10. FairDARTS$*$ differs from FairDARTS in that the former uses additional post-processing of the edge weights after search, with a hard limit on the number of edges kept per node pair.

Method	Params (M)	FLOPS (M)	Accuracy (%)
DARTS (11)	3.3	528	97.00
DARTS- (4)	3.5	583	97.41
FairDARTS$*$ (5)	2.8	373	97.46
FairDARTS	6.4	966	97.36
FairDARTS + DSCD	3.6	532	97.50

B Assessment of DSCD on multimodal functions

To confirm and quantify the beneficial effect of complementing gradient-based, local optimisation (Adam) with the proposed doubly stochastic coordinate descent (DSCD) routine, we performed comparisons with and without the routine on synthetic functions with known properties. For reference, we compare to performing uniform sampling over the domain, as well as Covariance Matrix Adaptation Evolution Strategy (CMA-ES) (7), a popular global optimisation method.

In Figure 3 we show the results on the Styblinski-Tang function (16) and the Schwefel function (8), which are popular functions for benchmarking optimization methods. Both functions have several local minima that are worse than the global minimum. We note that for every setting of Adam with a particular learning rate, or using a learning rate schedule, complementing the local steps with DSCD global steps (Section 3) improves the performance. On the Styblinski-Tang function, the difference is dramatic, as all the Adam variants without DSCD become stuck in a bad local minimum at every run.

C Beta annealing

For setting the proposal distribution, we propose an annealing scheme, which we will refer to as *Beta annealing*. The idea is that, at each step, we will sample a new (scaled) position following a Beta distribution, parameterised to have a varied concentration around the current position.

In practice, for our specific problem of setting operation edge weights going through a sigmoid, we set the proposal domain as $[-3, 3]$ for every dimension, which accounts for the region of the domain with a significant effect on the output. Note, however, that positions outside the domain are still possible to reach as of the local optimiser, although position outside will not be proposed in this step.

The current position we (min-max) normalize using the domain, so that it corresponds to a unit position $v_i \in [0, 1]$. The new proposal unit position, which we address below, is then mapped back to the original domain before the loss evaluation.

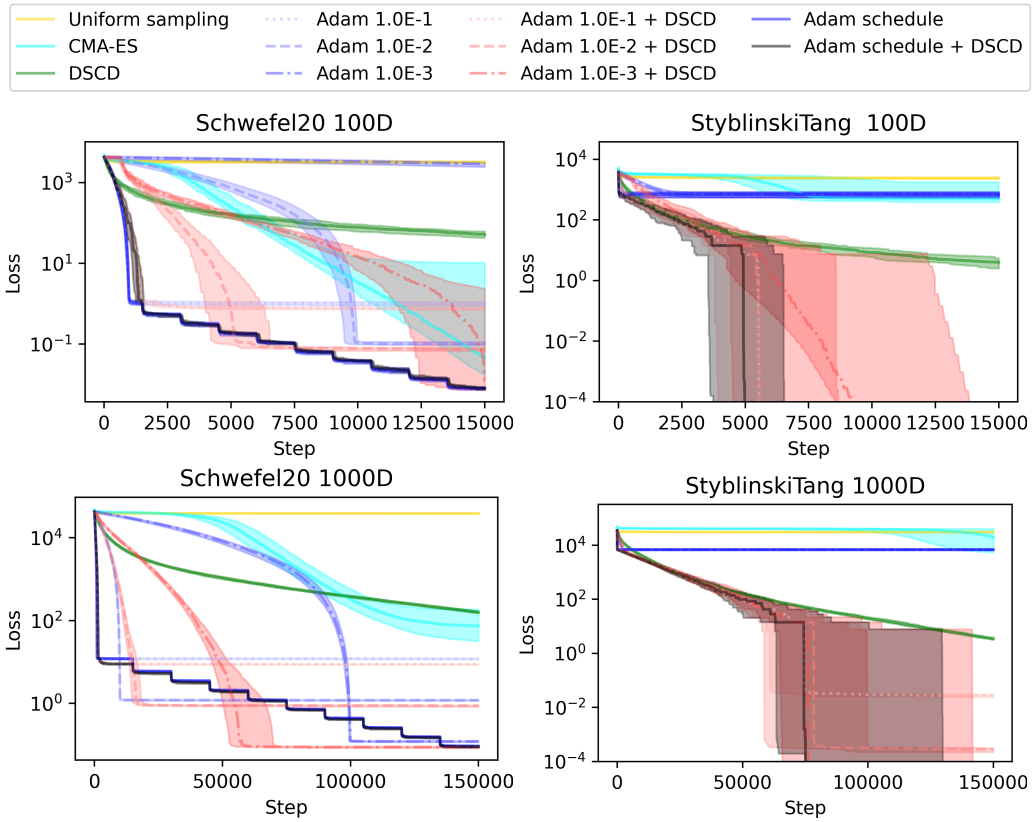

Figure 3: Shown is the median loss of 20 runs from uniformly sampled initial positions. Shaded areas display the 95% CI of the median. The numbers following "Adam" for each entry in the legend denote the used learning rate, where "schedule" denotes a linear learning rate scheduling between 0.001 and 0.1. The postfix "+ DSCD" denotes complementing the method with DSCD (Section 3).

We define a concentration parameter $\phi \in [0, 1)$, where $\phi = 0$ correspond to an uniform distribution of the (unit) domain, and $\phi \to 1$ tends towards a Dirac delta located at the current position. The former represents full global exploration (of the sampled dimension), independent of the current position. The latter represents full local exploitation at the current position. These two extremes are represented as parameterisations of a Beta distribution, and all the intermediate settings are as well. During search we start with $\phi = 0$ and anneal towards $\phi = 1$ at the final epoch. The annealing schedule used for ϕ is cosine annealing, typically used for learning rate scheduling (12).

The proposal (unit) position is sampled as $v_{i+1} \sim Beta(\alpha_i, \beta_i)$, where the α_i, β_i parameters depend on ϕ and the current (unit) position v_i. Specifically, α_i, β_i is derived at each step as following.

The two extremes, the uniform ($\phi = 0$) and Dirac delta ($\phi = 1$), have known α and β parameters, as we can solve for them given their respective (known) mean and standard deviation values,

$$\mu_{\text{unit uniform}} = 0.5, \sigma_{\text{unit uniform}} = {}^{1}/\sqrt{12}. \quad (2) \qquad \mu_{\text{Dirac delta}} = v_i, \sigma_{\text{Dirac delta}} = 0. \quad (3)$$

We linearly interpolate the mean μ and the standard deviation σ parameters to obtain the intermediate Beta distribution parameterisations in between the two extremes,

$$\mu := \phi v_i + (1 - \phi)\mu_{\text{uniform}} \quad (4) \qquad \sigma := (1 - \phi)\sigma_{\text{uniform}}. \quad (5)$$

Note that the standard deviation σ will approach (but never reach) zero as of $\phi < 1$.

We then solve for α and β using the analytical mean and standard deviation of Beta distributions, resulting in

$$\alpha = c_1 \beta \tag{6}$$

$$\beta = \frac{c_1 - c_2}{c_2(c_1 + 1)}, \tag{7}$$

where $c_1 = \frac{\mu}{1-\mu}$ and $c_2 = \sigma^2(c_1 + 1)^2$.

In the supplement we include an animation showing intermediate Beta distributions for various ϕ around a fixed point ($v_i = 0.75$).

D BACKGROUND

In (18) it was shown that detrimental solutions, in particular solutions exhibiting an overly large number of skip connections, coincide with high validation loss curvatures. In their work, they view these as problematic solutions within the solution set of the model. They propose regularisation on the weight space and early stopping, which they show is helpful in avoiding reaching these solutions. (5) instead proposes a change to the model, where different operation edges between the same nodes are not mutually exclusive, and they also propose a regularisation term pushing edge weights towards either zero or one. These alterations they show are beneficial for avoiding an over-reliance on skip connections, as well as reducing the approximation error resulting from the discretisation of the edge weights happening between the search and evaluation phase. In addition, they made the solution set more expressive as of allowing multiple simultaneous operations between the same nodes of a cell. In our work, we show that (5) still suffers from another detrimental effect, similar to the one it was addressing, indicating that the issue has not yet been solved in full. Similar to DARTS (11) and RobustDARTS (18), FairDARTS (5) constructs the architecture from copies of a *normal cell* and a *reduction cell*. What we show is that, using FairDARTS, the search is required to be stopped early to avoid reaching solutions that are detrimental to test performance during the evaluation phase or ultimately reaching invalid solutions post-discretisation. Notably, the architecture - as described by its operation edge weights - changes very little from very early on in the search until it is stopped. After the epoch it would have been stopped, the operation edge weights belonging to nodes in the normal cell all tend to zero. Following discretisation of the edge weights, the normal cell no longer propagates activations through, making the architecture invalid.

We suggest that the cause of this problem is that the detrimental solutions correspond to local minima in the edge weights space, given typical initial positions in the neural network parameters space. In particular, that as a consequence of the reduction cell operations relying on fewer parameters than normal cell operations, such solutions take up a large volume of the neural parameter space.

To see this, let us consider a detrimental solution $\{\boldsymbol{\alpha}, \boldsymbol{w}\}_{\text{detrimental}}$, where all $\boldsymbol{\alpha}_{\text{normal}}$ elements are close to zero. As will be confirmed in experiments, the neural network is sufficiently flexible to produce low loss solutions despite these elements being *close* to zero. Note that as long as activations can propagate through the normal cell, the reduction cell, being sufficiently expressive, can still represent low loss mappings. Furthermore, for constellations where the operations in the normal cell have little to no effect on the loss, this directly translates into invariance to all of the associated $\boldsymbol{w}_{\text{normal}}$ neural network parameters. In other words, such solutions are "large" in the sense that functionally equivalent solutions exist at all positions in the $\boldsymbol{w}_{\text{normal}}$ subspace. We may think of this as an equivalent solution set.

Secondly, consider a random initial set of neural network parameter values, $\boldsymbol{w}_{\text{initial}}$. The "larger" an equivalent solution set is, the more likely it is that $\boldsymbol{w}_{\text{initial}}$ will end up inside or "close" to it. In general, as well known and studied in the optimisation literature, gradient-based local optimisation is subject to finding local which are not necessarily global minima. In many applications, such as optimisation of neural network parameters alone, a local minimum might be "good enough". However, in this application, if it is applied to $\boldsymbol{\alpha}_{\text{normal}}$, it may add a bias towards local solutions, being compatible edge weights with the initial values of the neural network parameters.

E DIFFERENTIABLE NEURAL ARCHITECTURE SEARCH

E.1 ARCHITECTURE

DARTS (11), as similar to prior works (20; 15; 10), searches for a *cell* as the building block for the final architecture. In the case of convolutional networks, the cell is stacked, and for recurrent networks, it is recursively connected.

The cell constitutes a directed acyclic graph of N nodes. Each node x represents a latent representation and each directed edge (i, j) represents an operation $o_{i,j}$. A node depends on all of its predecessors as

$$x_j = \sum_{i<j} o_{i,j}(x_i). \tag{8}$$

The cell is assumed to have two input nodes and a single output node. In the case of convolutional networks, the input nodes are the outputs of the previous two layers, and for recurrent cells, the input nodes represent the current step, and the state carried from the previous step. The cell output is obtained by a reduction operation (e.g. concatenation) to all the intermediate nodes.

Let $\mathcal{O}$ be the set of candidate operations (e.g., convolution, max pooling, skip connection) available for each edge (i, j). (11) proposed a relaxation over the discrete operation choice using softmax

$$\bar{o}_{i,j}(x) = \sum_{o \in \mathcal{O}} \frac{\exp(\alpha_{o_i,j})}{\sum_{o' \in \mathcal{O}} \exp(\alpha_{o'_i,j})} o(x), \tag{9}$$

where the operation weights for a pair of nodes (i, j) are parameterised by a vector $\boldsymbol{\alpha}_{i,j}$ of dimension $|\mathcal{O}|$. Importantly, this makes the search space continuous and allows gradient-based optimisation methods.

FairDARTS (5), building upon (11), proposed replacing Eq. 9 with

$$\bar{o}_{i,j}(x) = \sum_{o \in \mathcal{O}} \sigma(\alpha_{o_i,j}) o(x) \tag{10}$$

where σ is the sigmoid function. This allows multiple operations per edge to be chosen simultaneously. If no operations are active for a given edge, this constitutes the zero operation (18).

For the case of convolutional neural networks, on which we will focus in this paper, both DARTS and FairDARTS searches for a normal cell and a reduction cell to build up the final architecture. The reduction cell, in contrast to the 'normal' cell, reduces the number of activation maps (or channels) out from the cell.

E.2 SEARCH

Let $\boldsymbol{\alpha}$ be the concatenated vector of all operation edge weights representing the architecture, and $\boldsymbol{w}$ be the concatenated neural network parameters associated with all operations. The $\boldsymbol{\alpha}$ vector contains the operation edge weights associated with both the normal cell and the reduction cell that are being searched for, i.e. $\boldsymbol{\alpha} = \{\boldsymbol{\alpha}_{\text{normal}}, \boldsymbol{\alpha}_{\text{reduction}}\}$, and the same applies to the weight parameters, $\boldsymbol{w} = \{\boldsymbol{w}_{\text{normal}}, \boldsymbol{w}_{\text{reduction}}\}$.

The architecture search problem was in (11) stated as the bi-level optimisation problem

$$\underset{\boldsymbol{\alpha}}{\text{minimize}} \quad \mathcal{L}_{\text{val}}(\boldsymbol{\alpha}, \boldsymbol{w}^*) \tag{11a}$$

$$\text{subject to} \quad \boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \mathcal{L}_{\text{train}}(\boldsymbol{\alpha}, \boldsymbol{w}), \tag{11b}$$

where $\mathcal{L}_{\text{val}}$ and $\mathcal{L}_{\text{train}}$ are the validation loss and training loss, respectively.

The proposed optimisation procedure in (11) is to approximate the gradient as

$$\nabla_{\boldsymbol{\alpha}} \mathcal{L}_{\text{val}}(\boldsymbol{\alpha}, \boldsymbol{w}^*) \approx \nabla_{\boldsymbol{\alpha}} \mathcal{L}_{\text{val}}(\boldsymbol{\alpha}, \boldsymbol{w} - \xi \nabla_{\boldsymbol{w}} \mathcal{L}_{\text{train}}(\boldsymbol{\alpha}, \boldsymbol{w})) \tag{12}$$

and perform gradient-based local optimisation, alternating between taking a step in the optimisation problem of $\arg\min_{\boldsymbol{\alpha}} \mathcal{L}_{\text{val}}$ and of $\arg\min_{\boldsymbol{w}} \mathcal{L}_{\text{train}}$. $\boldsymbol{w}$ are the current weights and ξ is the learning

Algorithm 2: Local optimisation with global optimisation backtracking

Input: Function f defined over $\mathcal{X}$, initial $\boldsymbol{x}_{\text{best}}, y_{\text{best}}$, local_step, global_step
Output: $\boldsymbol{x}_{\text{best}}, y_{\text{best}}$

```
reset schedule;
while budget remaining do
    take_global_step = schedule.current();
    if take_global_step then
        x_best, y_best = global_step(x_best, y_best) ;
        x_current, y_current = x_best, y_best;
    else
        x_current, y_current = local_step(x_current, y_current);
        if y_current < y_best then
            x_best, y_best = x_current, y_current;
        end
    end
    schedule.step(y_best);
end
return x_best, y_best;
```

rate for a step in the inner optimisation problem (Eq. 11b). This can be described as, at iteration t, take steps using the gradients defined at

$$\nabla_{\boldsymbol{\alpha}} \mathcal{L}_{\text{val}}(\boldsymbol{\alpha}_t, \boldsymbol{w}_t), \tag{13}$$

followed by

$$\nabla_{\boldsymbol{w}} \mathcal{L}_{\text{train}}(\boldsymbol{\alpha}_{t+1}, \boldsymbol{w}_t), \tag{14}$$

where $\boldsymbol{\alpha}_t$ and $\boldsymbol{w}_t$ is the position of respective parameter at the beginning of the iteration, and $\boldsymbol{\alpha}_{t+1}$ and $\boldsymbol{w}_{t+1}$ the updated positions, respectively.

F Global Optimisation Scheme

In Section D we hypothesised that the usage of local search for the $\boldsymbol{\alpha}$ weights adds bias towards solutions compatible with $\boldsymbol{w}$ solutions that are closer to the initial position. We will now describe a simple hybrid scheme, which makes the search for the $\boldsymbol{\alpha}$ weights "more global" in that it is less subject to the local curvature of the loss surface. We will later evaluate this scheme empirically, contrasting it to the previous, fully local search.

The $\boldsymbol{\alpha}$ parameter, being the collection of operation edge weights representing the architecture, is typically vastly different than the neural network parameters $\boldsymbol{w}$ in dimensionality. $\boldsymbol{\alpha}$ is, for the search spaced addressed, 196-dimensional, while $\boldsymbol{w}$ has millions of parameters. In FairDARTS and other DARTS variants, both parameters are optimised using gradient-based local optimisation, with alternating steps as described in Section E.2. However, the moderate dimensionality of the $\boldsymbol{\alpha}$ parameter makes it practically feasible to apply global optimisation techniques to optimise it. Specifically, we will make use of the idea of coordinate descent, where one coordinate is optimised at a time, as well as annealed sampling. In this section, we will describe a simple hybrid between local and global optimisation, which we later show performs well empirically.

We will first outline the general algorithm of the hybrid approach in Algorithm 1, in turn, parameterised by functions responsible for taking a "local" step and "global" step, respectively. At this abstraction level, we only distinguish between a global and local step, by if after taking the step, the "current position" is the same as the "best observed" position so far, in terms of smallest loss. For brevity, we leave out that the $global\_step$ function considers all observations so far, without loss of generality. The remaining components are specified in Section 3 and Section C.