Understanding and Robustifying Differentiable Architecture Search

Arber Zela
University of Freiburg
zelaa@cs.uni-freiburg.de

Thomas Elsken
Bosch Center for Artificial Intelligence
and University of Freiburg
Thomas.Elsken@de.bosch.com

Yassine Marrakchi
University of Freiburg
marrakch@cs.uni-freiburg.de

Tonmoy Saikia
University of Freiburg
saikiat@cs.uni-freiburg.de

Thomas Brox
University of Freiburg
brox@cs.uni-freiburg.de

Frank Hutter
University of Freiburg and
Bosch Center for Artificial Intelligence
fh@cs.uni-freiburg.de

Abstract

Differentiable Architecture Search (DARTS) has attracted a lot of attention due to its simplicity and small search costs. However, DARTS does not work robustly for new problems: we identify a wide range of search spaces for which DARTS yields degenerate architectures with very poor test performance. We show that, while DARTS successfully minimizes validation loss, the found solutions generalize poorly when they coincide with high validation loss curvature. By adding one of various types of regularization we can robustify DARTS to find solutions with smaller Hessian spectrum and with better generalization properties. Based on these observations we propose simple variations of DARTS with early stopping and adaptive regularization that perform substantially mode robustly. Our observations are robust across five search spaces on three image classification tasks.

1 Introduction

Neural Architecture Search (NAS), the process of automatically designing neural network architectures, has recently attracted attention by achieving state-of-the-art performance on a variety of tasks [Zoph and Le, 2017, Real et al., 2019]. *Differentiable architecture search* (DARTS) [Liu et al., 2019] significantly improved the efficiency of NAS over prior work, reducing its costs to the same order of magnitude as training a single neural network. This expanded the scope of NAS substantially, allowing it to also apply on more expensive problems, such as semantic segmentation [Chenxi et al., 2019] or disparity estimation [Saikia et al., 2019].

However, several researchers have also reported DARTS to *not* work well, in some cases even no better than random search [Li and Talwalkar, 2019, Sciuto et al., 2019]. Why is this? How can these seemingly contradicting results be explained? The overall goal of this paper is to understand and overcome such failure modes of DARTS. Specifically, we make the following contributions:

1. We identify a wide range of search spaces in which standard DARTS yields degenerate architectures with poor test performance (Section 2.1)

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

Figure 1: The poor cells standard DARTS nds on spaces S1-S4. For all spaces, DARTS chooses mostly parameter-less operations (skip connection) or even the harmful Noise operation. Here, we show the normal cells; see Figure 17 for the corresponding reduction cells.

2. By computing the eigenvalues of the Hessian matrix of the validation loss with respect to the architectural parameters, we show that there is a strong correlation between its largest eigenvalue and the architecture's generalization error (Section 2.2).
3. We show that, related to previous work on sharp/ at local minima, regularizing the inner objective of DARTS allows it to nd solutions with smaller Hessian spectrum and better generalization properties (Section 2.3).
4. We propose simple variations of DARTS that stop early and adadaptively set the regularizer based on the eigenvalue spectrum (Section 3).

Our ndings are robust across ve different search spaces evaluated on three different image recognition benchmarks each. They consolidate the ndings of the various results in the literature and lead to a substantially more robust version of DARTS. We provide our implementation and scripts to facilitate reproducibility[1].

2 Failure modes and overcoming them by regularization

We start with four search spaces similar to the CIFAR-10 search space used in the original DARTS paper [Liu et al.] but simpler, and evaluated across three different datasets (CIFAR-10, CIFAR-100 and SVHN). These search spaces use the same macro architecture as DARTS, consisting of normal and reduction cells, but only allow a reduced set of operators for the cell search space (see Appendix A for more details on how DARTS works and Appendix B for more details on these search spaces).

2.1 When DARTS fails

We ran DARTS on each of these spaces, using exactly the same setup as Liu et al. [2019]. Figure 1 shows the poor cells DARTS selected on these search spaces for CIFAR-10 (see Appendix F for analogous results on the other datasets). Already visually, one might suspect that the found cells are suboptimal: the parameter-less skip connections dominate in almost all the edges for spaces S1-S3, and for S4 even the harmful Noise operation was selected for ve out of eight operations. Table 1 con rms the very poor performance standard DARTS yields for all of these settings also compared to Random Search with weight sharing (RS-ws) [Li and Talwalkar, 2019].

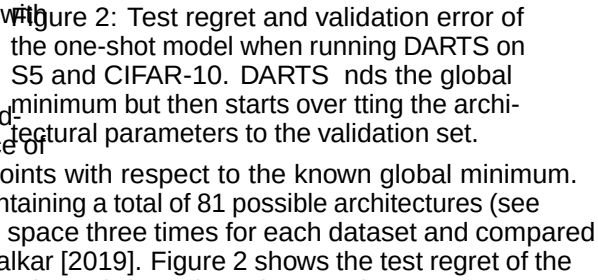


Figure 2: Test regret and validation error of the one-shot model when running DARTS on S5 and CIFAR-10. DARTS nds the global minimum but then starts over tting the architectural parameters to the validation set.

Very small search space with known global optimum. Knowing the global minimum has the advantage that one can benchmark the performance of algorithms by measuring the regret of chosen points with respect to the known global minimum. Therefore, we created a small search space S5 only containing a total of 81 possible architectures (see Appendix B). We then ran DARTS on this search space three times for each dataset and compared its result to the baseline of RS-ws by Li and Talwalkar [2019]. Figure 2 shows the test regret of the architectures selected by DARTS (blue) and RS-ws (green) throughout the search. DARTS manages

[1] https://github.com/automl/RobustDARTS

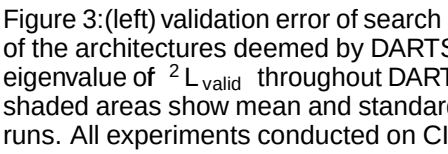


Figure 3: (left) validation error of search model; (middle) test error of the architectures deemed by DARTS optimal; (right) dominant eigenvalue of $\nabla^2 L_{valid}$ throughout DARTS search. Solid line and shaded areas show mean and standard deviation of 3 independent runs. All experiments conducted on CIFAR-10.

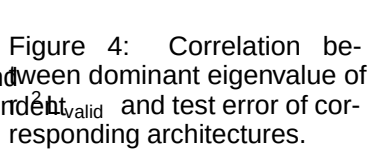


Figure 4: Correlation between dominant eigenvalue of $\nabla^2 L_{valid}$ and test error of corresponding architectures.

to find an architecture close to the global minimum, but around epoch 40 the test performance deteriorates. Note that the one-shot (search model) validation error (dashed red line) does not deteriorate but rather converges, indicating that the architectural parameters overfit to the validation set. In contrast, RS-ws stays relatively constant throughout the search; when evaluating only the final architecture found, RS-ws indeed outperforms DARTS.

2.2 Large curvature and generalization performance

One may hypothesize that DARTS performs poorly because its approximate solution of the bi-level optimization problem by iterative optimization fails, but we actually observe validation errors to progress nicely: Figure 3 (left) shows that the search model validation error converges in all cases, even though the cell structures selected here are the ones in Figure 1.

Rather, the architectures DARTS finds do not generalize well. This can be seen in Figure 3 (middle). There, every 5 epochs, we evaluated the architecture deemed by DARTS to be optimal according to the α values. Note that whenever evaluating on the test set, we retrain from scratch the architecture obtained after applying the $\arg\max$ to the architectural weights. As one can notice, the architectures start to degenerate after a certain number of search epochs, similarly to the results shown in Figure 2. We hypothesized that this might be related to the phenomenon of sharp local minima [Hochreiter and Schmidhuber, 1997, Keskar et al., 2016, Chaudhari et al., 2017, Yao et al., 2018], which have also been observed in the hyperparameter space [Nguyen et al., 2018]. To test this hypothesis, we computed the full Hessian $\nabla^2 L_{valid}$ of the validation loss w.r.t. the architectural parameters on a randomly sampled mini-batch. Figure 3 (right) shows that the dominant eigenvalue λ_{max} (which serves as a proxy for the sharpness) indeed increases in standard DARTS, along with the test error (middle) of the final architectures, while the validation error still decreases (left). We also studied the correlation between λ_{max} and test error more directly, by measuring these two quantities for 24 different architectures (obtained via standard DARTS and the regularized versions we discuss in Section??). For the example of space S1 on CIFAR-10, Figure 4 shows that λ_{max} indeed strongly correlates with test error (with a Pearson correlation coefficient of 0.867).

2.3 Regularization improves generalization

In our bi-level optimization setting, the outer variables' trajectory depends directly on the inner optimization procedure. Therefore, modifying the landscape of the inner objective L_{train} might potentially redirect the outer variables to better areas of the architectural space and keep the dominant eigenvalue low. We investigate two ways of regularizing the DARTS search procedure, namely via data augmentation and L_2 regularization in the inner objective. We highlight that we do not alter the regularization of the final training and evaluation phase, but solely that of the DARTS search phase. Indeed, we find that a stronger regularization factor during search leads to a better generalization of the selected architectures as shown in Figure 5 (see also Figure 10 in the appendix). We also observe the same pattern in the tasks of disparity estimation (see Appendix D) and language modelling (see Appendix E). As suspected, also the architectural parameters stay and end up in flatter areas of the architecture landscape, as shown in Figure 8 (more details in Appendix C.2).

Figure 5: Effect of more L_2 regularization during the DARTS search phase, on the test performance of the architectures discovered by DARTS and DARTS-ES. The results are presented for each of the search spaces S1-S4 and for CIFAR-10, CIFAR-100 and SVHN. The solid lines correspond to DARTS, the dashed lines to DARTS-ES.

2.4 Further analysis

Why does DARTS get attracted to these bad regions in the architecture space? This might arise potentially from some premature convergence in the weights' space, supposing that the landscape of the training loss L_{train} does not change significantly after each update. In a non-convex landscape, this minimizer $w^{(1)}$ of L_{train} might not necessarily be the one that also minimizes L_{valid} (see Franceschi et al. [2018]), i.e. there might exist another $w^{(2)} \in \text{argmin}_w L_{train}(\alpha, w)$ such that $L_{valid}(\alpha, w^{(2)}) < L_{valid}(\alpha, w^{(1)})$. More concretely, the parameterless operations such as skip-connections might get higher weight especially in the beginning of search due to the easiness of gradients to flow through these paths during training. Regularizing the inner objective by adding a convex term or perturbing the inputs will eventually redirect also the gradients flowing backwards, and consecutively also the attention DARTS focuses on different architectural operations.

Now we provide further evidence on the relation between the sharpness of minimas and the generalization error. As it is well known in the settings of large vs. small batch training [Yao et al., 2018, Keskar et al., 2016, Hochreiter and Schmidhuber, 1997], one potential hypothesis explaining the relationship between the sharpness of minimas and generalization properties of a neural network, is based on the fact that the training function is much more sensitive at a sharp minimizer, e.g. to the variations in the input data. This may lead to relatively large accuracy drop even from small discrepancies between training and test data. Analogously, we conjectured that this also holds for rather than w and therefore we investigated the relation between large EV (w.r.t.) - as a proxy for sharp minima - and generalization performance. Similarly, sharp minima would also be much more sensitive to variations in the architecture, which is relevant since DARTS discretizes (by taking the argmax over operations in each edge) the optimal after search, resulting in ^{d} somewhere in the vicinity of . In the case of a sharp minimum , ^{d} might already have a larger objective function value, while in the case of a at minimum, ^{d} is expected to have an objective value similar to .

Figure 6: Drop in accuracy after discretizing the search model. Example of some of the settings.

To make this more crisp, we conducted the following experiment: after the DARTS search has finished, we discretize the architecture and evaluate it with the search model's weights, rather than retraining, and compare the performance to the one-shot model's performance $|L_{test}( ) - L_{valid}(^d)|$. Figure 6 shows the drop in performance due to this discretization step for some of the settings: this drop is much larger when there is little regularization (drop prob = 0), resulting in large EVs (see Figure 14), corresponding to a sharp

Figure 7: Correlation between $|L_{test}( ) - L_{valid}(^d)|$ and the loss's curvature.

minmum. Figure 7 shows the relationship between the dominant eigenvalues at the end of search and the drop in accuracy when doing the discretization step as described above.

3 Architecture search early stopping and adaptive regularization

Early stopping based on $\nabla^2 L_{valid}$. We just observed that (1) the test error seems to be positively correlated (also see Figure 4 in the appendix) with the largest eigenvalue λ_{max} of the Hessian matrix of the validation loss $\nabla^2 L_{valid}$, and that (2) λ_{max} increases over time. A simple strategy to avoid test errors from increasing would therefore be to stop the optimization when λ_{max} increases too much. To implement this idea, we propose a simple heuristic that worked off-the-shelf without the need for any tuning. Let $\bar{\lambda}_{max}(i)$ denote the value of λ_{max} smoothed over $k = 5$ epochs around i; then, we stop if $\bar{\lambda}_{max}(i-k) / \bar{\lambda}_{max}(i) < 0.75$ and return the architecture from epoch $i-k$. By this early stopping heuristic, we do not only avoid exploding eigenvalues, which are correlated with poor generalization, but also shorten the time of the search. Figure 8 visualizes the eigenvalue trajectory and the early stopping. Table 1 (DARTS-ES) shows the results for running DARTS with this early stopping criterion across S1-S4 and all three image classification datasets. We also apply this procedure when altering the regularization factors. The corresponding test errors are shown in Figures 5 and 10 (dashed lines). Early stopping significantly improved DARTS for most settings without ever harming it.

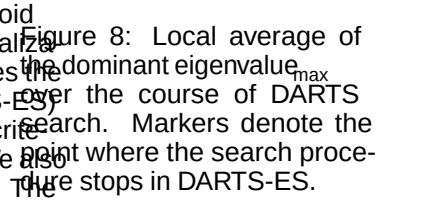


Figure 8: Local average of the dominant eigenvalue λ_{max} over the course of DARTS search. Markers denote the point where the search procedure stops in DARTS-ES.

DARTS with adaptive regularization. One may think of adapting the regularization in DARTS based on the validation loss curvature, instead of tuning that value and keeping it constant during search. The simplest off-the-shelf procedure towards this would be to increase the regularization strength whenever the dominant eigenvalue starts increasing rapidly. Algorithm 1 (DARTS-ADA, in the appendix) shows such a procedure. We use the same criterion (stop_criter) used for DARTS-ES. Whenever this criterion is met, DARTS rolls back to the architectural and search model parameters at stop_epoch, and continues the search with a larger (by a factor of η) regularization value R (e.g. drop path probability, L_2 or both) for the remaining epochs. This procedure is always repeated if the criterion is met, unless the regularization value exceeds some maximum predefined value R_{max}.

Table 1: Performance of architectures found by DARTS (-ES/-ADA) vs. RandomNAS with weight sharing. For each of the settings we repeat the search 3 times and report the mean ± std of the 3 found architectures retrained from scratch.

Setting		RandomNAS		DARTS		DARTS-ES		DARTS-ADA	
C10	S1	3.17	0.15	4.66	0.71	3.05	0.07	3.03	0.08
	S2	3.46	0.15	4.42	0.40	3.41	0.14	3.59	0.31
	S3	2.92	0.04	4.12	0.85	3.71	1.14	2.99	0.34
	S4	89.39	0.84	6.95	0.18	4.17	0.21	3.89	0.67
C100	S1	25.81	0.39	29.93	0.41	28.90	0.81	24.94	0.81
	S2	22.88	0.16	28.75	0.92	24.68	1.43	26.88	1.11
	S3	24.58	0.61	29.01	0.24	26.99	1.79	24.55	0.63
	S4	30.01	1.52	24.77	1.51	23.90	2.01	23.66	0.90
SVHN	S1	2.64	0.09	9.88	5.50	2.80	0.09	2.59	0.07
	S2	2.57	0.04	3.69	0.12	2.68	0.18	2.79	0.22
	S3	2.89	0.09	4.00	1.01	2.78	0.29	2.58	0.07
	S4	3.42	0.04	2.90	0.02	2.55	0.15	2.52	0.06

We evaluated DARTS-ADA with $R = 3 \cdot 10^{-4}$ (DARTS default), $R_{max} = 3 \cdot 10^{-2}$ and $\eta = 10$ on all the search spaces and datasets we use for image classification. Results in Table 1 (DARTS-ADA) show that the improvements are consistent across all settings compared to the default DARTS and leads to better performance than RS-ws [Li and Talwalkar, 2019] in most cases.

4 Conclusions

We showed that DARTS often results in degenerate architectures with relatively sharp curvature of the architecture search objective and poor generalization. Our empirical results show that properly regularizing the search model improves generalization by redirecting the architectural parameters to flatter areas of the landscape. We also proposed a simple early stopping criterion and adaptive regularization for DARTS which substantially improve our understanding of DARTS' failure modes and lead to much more robust versions across many settings and tasks.

Acknowledgments

The authors acknowledge funding by the Robert Bosch GmbH, support by the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation programme through grant no. 716721, and by BMBF grant DeToL.

References

Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In International Conference on Learning Representations, 2017.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. Aging Evolution for Image Classifier Architecture Search. In AAAI, 2019.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In International Conference on Learning Representations, 2019.

Liu Chenxi, Chen Liang Chieh, Schroff Florian, Adam Hartwig, Hua Wei, Yuille Alan L., and Fei Fei Li. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In Conference on Computer Vision and Pattern Recognition, 2019.

Tonmoy Saikia, Yassine Marrakchi, Arber Zela, Frank Hutter, and Thomas Brox. Autodispnet: Improving disparity estimation with automl, 2019.

Liam Li and Ameet Talwalkar. Random search and reproducibility for neural architecture search. CoRR, abs/1902.07638, 2019.

Christian Sciuto, Kaicheng Yu, Martin Jaggi, Claudiu Musat, and Mathieu Salzmann. Evaluating the search phase of neural architecture search. arXiv preprint, 2019.

H. Liu, K. Simonyan, O. Vinyals, C.Fernando, and K. Kavukcuoglu. Hierarchical representations for efficient architecture search. In International Conference on Learning Representations (ICLR) 2018 Conference Track April .

Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. Neural Comput., 9(1):1–42, January 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.1.1.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. arXiv preprint arXiv:1609.04836, 2016.

P. Chaudhari, Anna Choromanska, S. Soatto, Yann LeCun, C. Baldassi, C. Borgs, J. Chayes, Levent Sagun, and R. Zecchina. Entropy-sgd: Biasing gradient descent into wide valleys. In International Conference on Learning Representations (ICLR), 2017.

Zhewei Yao, Amir Gholami, Qi Lei, Kurt Keutzer, and Michael W Mahoney. Hessian-based analysis of large batch training and robustness to adversaries. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, Advances in Neural Information Processing Systems 31, pages 4949–4959. Curran Associates, Inc., 2018.

Thanh Dai Nguyen, Sunil Gupta, Santu Rana, and Svetha Venkatesh. Stable bayesian optimization. International Journal of Data Science and Analytics, 6(4):327–339, Dec 2018. ISSN 2364-4168. doi: 10.1007/s41060-018-0119-9.

Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimiliano Pontil. Bilevel programming for hyperparameter optimization and meta-learning. In Jennifer Dy and Andreas Krause, editors, Proceedings of the 35th International Conference on Machine Learning, volume 80 of Proceedings of Machine Learning Research, pages 1568–1577, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. Journal of Machine Learning Research, 20(55):1–21, 2019a.

Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. In International Conference on Learning Representations, 2017a.

Zhao Zhong, Jingchen Yan, Wei Wu, Jing Shao, and Cheng-Lin Liu. Practical block-wise neural network architecture generation. 2018.

Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. In *Conference on Computer Vision and Pattern Recognition*, 2018.

Kenneth O Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10:99–127, 2002.

Hanxiao Liu, Karen Simonyan, Oriol Vinyals, Chrisantha Fernando, and Koray Kavukcuoglu. Hierarchical Representations for Ef cient Architecture Search. In *International Conference on Learning Representations*, 2018.

Risto Miikkulainen, Jason Liang, Elliot Meyerson, Aditya Rawal, Dan Fink, Olivier Francon, Bala Raju, Hormoz Shahrzad, Arshak Navruzyan, Nigel Duffy, and Babak Hodjat. Evolving Deep Neural Networks. In *arXiv:1703.00548*, March 2017.

Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V. Le, and Alexey Kurakin. Large-scale evolution of image classi ers. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2902–2911, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

Han Cai, Tianyao Chen, Weinan Zhang, Yong Yu, and Jun Wang. Ef cient architecture search by network transformation. In *AAAI*, 2018a.

Han Cai, Jiacheng Yang, Weinan Zhang, Song Han, and Yong Yu. Path-Level Network Transformation for Ef cient Architecture Search. In *International Conference on Machine Learning*, June 2018b.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Simple And Ef cient Architecture Search for Convolutional Neural Networks. In *NIPS Workshop on Meta-Learning*, 2017.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Ef cient multi-objective neural architecture search via lamarckian evolution. In *International Conference on Learning Representations*, 2019b.

Shreyas Saxena and Jakob Verbeek. Convolutional neural fabrics. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4053–4061. Curran Associates, Inc., 2016.

Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. Understanding and simplifying one-shot architecture search. In *International Conference on Machine Learning*, 2018.

Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. Ef cient neural architecture search via parameter sharing. In *International Conference on Machine Learning*, 2018.

Bowen Baker, Otkrist Gupta, Ramesh Raskar, and Nikhil Naik. Accelerating Neural Architecture Search using Performance Prediction. In *NIPS Workshop on Meta-Learning*, 2017b.

Stefan Falkner, Aaron Klein, and Frank Hutter. BOHB: Robust and ef cient hyperparameter optimization at scale. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Research*, pages 1437–1446, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/falkner18a.html .

L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar. Hyperband: Bandit-based con guration evaluation for hyperparameter optimization. In *Proceedings of the International Conference on Learning Representations (ICLR'17)*, 2017. Published online: iclr.cc .

Arber Zela, Aaron Klein, Stefan Falkner, and Frank Hutter. Towards automated deep learning: Ef cient joint neural architecture and hyperparameter search. In *ICML 2018 Workshop on AutoML (AutoML 2018)*, 2018.

Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. SNAS: stochastic neural architecture search. In *International Conference on Learning Representations*, 2019.

Francesco Casale, Jonathan Gordon, and Nicolo Fusi. Probabilistic neural architecture search. *arXiv preprint*, 2019.

Han Cai, Ligeng Zhu, and Song Han. ProxylessNAS: Direct neural architecture search on target task and hardware. In *International Conference on Learning Representations*, 2019.

Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), 2016. arXiv:1512.02134.

D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In A. Fitzgibbon et al. (Eds.), editor, European Conf. on Computer Vision (ECCV), Part IV, LNCS 7577, pages 611–625. Springer-Verlag, October 2012.

A Background and Related Work

A.1 Relation between flat/sharp minima and generalization performance

Already Hochreiter and Schmidhuber [1997] observed that flat minima of the training loss yield better generalization performance than sharp minima. Recent work [Keskar et al., 2016, Yao et al., 2018] focuses more on the settings of large/small batch size training, where observations show that small batch training tends to get attracted to flatter minima and generalizes better. Similarly, Nguyen et al. [2018] observed that this phenomenon manifests also in the hyperparameter space. They showed that whenever the hyperparameters over fit the validation data, the minima lie in a sharper region of the space. This motivated us to conduct a similar analysis in the context of differentiable architecture search in Section 2.2, where we see the same effect in the space of neural network architectures.

A.2 Neural Architecture Search

Neural Architecture Search (NAS) denotes the process of automatically designing neural network architectures in order to overcome the cumbersome trial-and-error process when designing architectures manually. We briefly review NAS here and refer to the recent survey by Elsken et al. [2019a] for a more thorough overview. Prior work mostly employs either reinforcement learning techniques [Baker et al., 2017a, Zoph and Le, 2017, Zhong et al., 2018, Zoph et al., 2018] or evolutionary algorithms [Stanley and Miikkulainen, 2002, Liu et al., 2018, Miikkulainen et al., 2017, Real et al., 2017, 2019] to optimize the discrete architecture space. As these methods are often very expensive, various work focuses on reducing the search costs by, e.g., employing network morphisms [Cai et al., 2018a,b, Elsken et al., 2017, 2019b], weight sharing within one-shot models [Saxena and Verbeek, 2016, Bender et al., 2018, Pham et al., 2018] or multi-fidelity optimization [Baker et al., 2017b, Falkner et al., 2018, Li et al., 2017, Zela et al., 2018], however their applicability still often remains restricted to rather simple tasks and small datasets. A recent line of work focuses on relaxing the neural architecture search space by making discrete decisions continuous and thus learnable via gradient descent [Liu et al., 2019, Xie et al., 2019, Casale et al., 2019, Cai et al., 2019].

A.3 Differentiable Architecture Search (DARTS)

Continuous relaxation of the search space. In agreement with prior work [Zoph et al., 2018, Real et al., 2019], DARTS optimizes only substructures called cells that are stacked to define the full network architecture. Each cell contains N nodes organized in a directed acyclic graph. The graph contains two inputs nodes (given by the outputs of the previous two cells), a set of intermediate nodes, and one output node (given by concatenating all intermediate nodes). Each intermediate node $x^{(i)}$ represents a feature map. See Figure 1 for an illustration of such a cell. Instead applying a single operation to a specific node during architecture search, Liu et al. [2019] relax the decision which operation to choose by computing the intermediate node as a mixture of candidate operations, applied to predecessor nodes $x^{(i)}, i < j$,

$$x^{(j)} = \sum_{i<j} \sum_{o \in \mathcal{O}} \frac{\exp(\alpha^{i,j}_o)}{\sum_{o' \in \mathcal{O}} \exp(\alpha^{i,j}_{o'})} o\left(x^{(i)}\right); \tag{1}$$

where $\mathcal{O}$ denotes the set of all candidate operations (e.g., 3 convolution, skip connection, 3×3 max pooling, ...) and $\alpha = (\alpha^{i,j}_o)_{i,j,o}$ serves as a real valued parametrization of the architecture.

Gradient-based optimization of the search space. DARTS then optimizes both the weights of the search network (often called *one-shot model*, since the weights of all individual sub-graphs/architectures are shared) and architectural parameters α by alternating gradient descent. Learning

the network weights and the architecture parameters are performed on the training and validation set, respectively. This can be interpreted as solving the bi-level optimization problem, where the validation and training loss L_{valid} and L_{train} are the outer and inner problems, respectively, while the architectural parameters and network weights w the outer and inner variables, respectively. Note that DARTS only approximates the inner-level solution by a single gradient step.

At the end of the search phase, a discrete cell is obtained by choosing the k most important incoming operation for each intermediate node while all others are pruned. Importance is measured by the operation weighting factor $\frac{\exp(\alpha^{i,j}_o)}{\sum_{o' \in O} \exp(\alpha^{i,j}_{o'})}$.[2]

B Details on search spaces in Section 2.1 and final architecture evaluations for the image classification task

The search space S1-S4 used throughout the paper are defined as follows:

- S1: This search space uses a different set of two operators per edge, which we identified using an offline process that iteratively dropped the operations from the original DARTS search space with the least importance. This pre-optimized space has the advantage of being quite small while still including many strong architectures. Refer to Figure 9 for an illustration of this pre-optimized space.
- S2: The set of candidate operations per edge is $\{3 \times 3 \text{ SepConv}, \text{SkipConnect}\}$. We choose these operations since they are the most frequent in the discovered cells reported by Liu et al. [2019].
- S3: The set of candidate operations per edge is $\{3 \times 3 \text{ SepConv}, \text{SkipConnect}, \text{Zero}\}$, where the Zero operation simply replaces every value in the input feature map by zeros.
- S4: The set of candidate operations per edge is $\{3 \times 3 \text{ SepConv}, \text{Noise}\}$, where the Noise operation simply replaces every value from the input feature map by noise $\mathcal{N}(0, 1)$. This is the only space out of S1-S4 that is not a strict subspace of the original DARTS space; we intentionally added the Noise operation, which actively harms performance and should therefore not be selected by DARTS.

S5: Very small search space with known global optimum. Differently from S1-S4 this search space consists of only one intermediate node for both normal and reduction cells, and 3 operation choices in each edge, namely 3×3 SepConv, SkipConnection, and 3×3 MaxPooling. The total number of possible architectures in this space is 81, all of which we evaluated a-priori.

B.1 Architecture Evaluation

For CIFAR-100 and SVHN we use 16 number of initial filters and 8 cells when training architectures from scratch for all the experiments we conduct. The rest of the settings is the same as in Liu et al. [2019].

On CIFAR-10, when scaling the ScheduledDropPath drop probability, we use the same settings for training from scratch the found architectures as in the original DARTS paper, i.e. 36 initial filters and 20 stacked cells. However, for search space S2 and S4 we reduce the number of initial filters to 16 in order to avoid memory issues, since the cells found with more regularization usually are composed only with separable convolutions. When scaling the L_2 factor on CIFAR-10 experiments we use 16 initial filters and 8 stacked cells, except the experiments on S1, where the settings are the same as in Liu et al. [2019], i.e. 36 initial filters and 20 stacked cells.

Note that although altering the regularization factors during DARTS search, when training the final architectures from scratch we always use the same values for them as in Liu et al. [2019], i.e. ScheduledDropPath maximum drop probability linearly increases from 0 towards 0.2 throughout training, Cutout is always enabled with cutout probability 1.0, and the L_2 regularization factor is set to 3×10^{-4}.

[2] One usually searches for two types of cells, a reduction cell (which reduces the spatial dimension), and a normal cell (which preserves spatial resolution).

(a) Normal cell space

(b) Reduction cell space

Figure 9: Search space.

C Additional empirical results

Algorithm 1: DARTS_ADA

```
/* E: epochs to search; R: initial regularization value; R_max: maximal regularization value; stop_criter: stopping
   criterion; : regularization increase factor */
Input : E, R, R_max, stop_criter,
/* start search for E epochs */
for epoch in E do
    /* run DARTS for one epoch and return stop=True together with the stop_epoch */
    /* and the architecture at stop_epoch if the criterion is met */
    stop, stop_epoch, arch train_and_eval(stop_criter) ;
    if stop & R R_max then
        /* start DARTS from stop_epoch with a larger R */
        arch DARTS_ADA(E - stop_epoch, R, R_max, stop_criter, ) ;
        break
    end
end
Output : arch
```

Table 2: Validation (train) and test accuracy on CIFAR-10 of the one-shot and final evaluation model, respectively. The values in the last column show the maximum eigenvalue λ_{max} (computed on a random sampled mini-batch) of the Hessian, at the end of search for different maximum drop path probability). The four blocks in the table state results for the search spaces S1-S4, respectively.

	Drop	Valid acc.			Test acc.			Params			λ_{max}		
	Prob.	seed 1	seed 2	seed 3	seed 1	seed 2	seed 3	seed 1	seed 2	seed 3	seed 1	seed 2	seed 3
S1	0.0	87.22	87.01	86.98	96.16	94.43	95.43	2.24	1.93	2.03	1.023	0.835	0.698
	0.2	84.24	84.32	84.22	96.39	96.66	96.20	2.63	2.84	2.48	0.148	0.264	0.228
	0.4	82.28	82.18	82.79	96.44	96.94	96.76	2.63	2.99	3.17	0.192	0.199	0.149
	0.6	79.17	79.18	78.84	96.89	96.93	96.96	3.38	3.02	3.17	0.300	0.255	0.256
S2	0.0	88.49	88.40	88.35	95.15	95.48	96.11	0.93	0.86	0.97	0.684	0.409	0.268
	0.2	85.29	84.81	85.36	95.15	95.40	96.14	1.28	1.44	1.36	0.270	0.217	0.145
	0.4	82.03	82.66	83.20	96.34	96.50	96.44	1.28	1.28	1.36	0.304	0.411	0.282
	0.6	79.86	80.19	79.70	96.52	96.35	96.29	1.21	1.28	1.36	0.292	0.295	0.281
S3	0.0	88.78	89.15	88.67	94.70	96.27	96.66	2.21	2.43	2.85	0.496	0.535	0.446
	0.2	85.61	85.60	85.50	96.78	96.84	96.74	3.62	4.04	2.99	0.179	0.185	0.202
	0.4	83.03	83.24	83.43	97.07	96.85	96.48	4.10	3.74	3.38	0.156	0.370	0.184
	0.6	79.86	80.03	79.68	96.91	94.56	96.44	4.46	2.30	2.66	0.239	0.275	0.280
S4	0.0	86.33	86.72	86.46	92.80	93.22	93.14	1.05	1.13	1.05	0.400	0.442	0.314
	0.2	81.01	82.43	82.03	95.84	96.08	96.15	1.44	1.44	1.44	0.070	0.054	0.079
	0.4	79.49	79.67	78.96	96.11	96.30	96.28	1.44	1.44	1.44	0.064	0.057	0.049
	0.6	74.54	74.74	74.37	96.42	96.36	96.64	1.44	1.44	1.44	0.057	0.060	0.066

C.1 Regularization via data augmentation

We investigate the effect of regularizing via data augmentation, namely masking out parts of the input and intermediate feature maps via Cutout (CO) [DeVries and Taylor, 2017] and ScheduledDropPath (DP) [Zoph et al., 2018], respectively, during architecture search. We apply DP to randomly zero out mixed operations starting with a drop probability of 0 and linearly increasing it over the course of architecture search until it reaches a maximum value. At the same, we randomly zero out patches in the input images by applying CO; the CO probability also linearly increases.

We ran DARTS plus drop-path (with and without our early stopping criterion, DARTS-ES) with four values of the maximum drop-path probability (0.0, 0.2, 0.4 and 0.6) on all three image classification datasets and search spaces S1-S4. Figure 10 summarizes the results: regularization improves the test performance of DARTS and DARTS-ES in all cases, sometimes very substantially. Table 2 provides additional details, also showing that the one-shot model accuracy consistently drops by increasing the drop-path probability, while the test accuracy improves (up to a certain limit). This demonstrates that over fitting of the architectural parameters is reduced due to an implicit regularization effect.

C.2 A closer look at the dominant eigenvalues

Over the course of all experiments from the previous sections, we tracked the largest eigenvalue across all configuration and datasets to see how they evolve during the search. Figure 12 and 13

Figure 10: Effect of more regularization via ScheduledDropPath during the DARTS search phase, on the test performance of the architectures discovered by DARTS and DARTS-ES. The results are presented for each of the search spaces S1-S4 and for CIFAR-10, CIFAR-100 and SVHN. The solid lines correspond to DARTS, the dashed lines to DARTS-ES.

show the results across all settings. The markers on each line highlight the epochs where DARTS early stops based on the procedure described in Section 3. It can be clearly seen that increasing the inner objective regularization, both in terms of L_2 or data augmentation, helps controlling the largest eigenvalue and keeping it to a small value, which again helps explaining why the architectures found with stronger regularization generalize better. Figure 4 shows that λ_{max} (average over search epochs) indeed correlates with test error (with a Pearson correlation coefficient of 0.867).

The plots in Figure 14 show the full spectrum (sorted based on eigenvalue absolute values) at the end of search. As one can see, not only the dominant eigenvalue is larger compared to the cases when the regularization is stronger and the generalization of architectures is better, but also the other eigenvalues in the spectrum have larger absolute value, indicating a sharper objective landscape towards many dimensions.

D Disparity Estimation

To study whether our findings generalize beyond image recognition, we also analyzed a search space for a very different problem: finding encoder-decoder architectures for the dense regression task of disparity estimation. We base this search space on AutoDispNet-C [Saikia et al., 2019], which used DARTS for a space containing normal, downsampling and upsampling cells. We again constructed a reduced space, using the following candidate operations for each edge in each of these cells: SepConv 3×3, MaxPool 3×3, SkipConnect. Please refer to Saikia et al. [2019] for more details.

D.1 Datasets

We use the FlyingThings3D dataset [Mayer et al., 2016] for training AutoDispNet. It consists of rendered stereo image pairs and their ground truth disparity maps. The dataset provides a training and testing split consisting of 21,818 and 4248 samples respectively with an image resolution of 960×540. We use the Sintel dataset (Butler et al. [2012]) for testing our networks. Sintel is another synthetic dataset from derived from an animated movie which also provides ground truth disparity maps (1064 samples) with a resolution of 1024×436.

D.2 Training

For training the search network, images are downsampled by a factor of two and trained for 300k mini-batch iterations. During search, we use SGD and ADAM to optimize the inner and outer objectives respectively. Differently from the original AutoDispNet we do not warmstart the search model weights before starting the architectural parameter updates. The extracted network is also trained for 300k mini-batch iterations but full resolution images are used. Here, ADAM is used for optimization and the learning rate is annealed to 0 from $1e-4$, using a cosine decay schedule.

D.3 Effect of regularization on the inner objective

To study the effect of regularization on the inner objective for AutoDispNet-C we use experiment with two types of regularization: data augmentation and $L2$ regularization on network weights. We report the average end point error (EPE), which is the Euclidean distance between the predicted and ground truth disparity maps.

Data augmentation. Inspite of fairly large number of training samples in FlyingThings3D, data augmentation is crucial for good generalization performance. Disparity estimation networks employ spatial transformations such as translation, cropping, shearing and scaling. Additionally, appearance transformations such as additive Gaussian noise, changes in brightness, contrast, gamma and color are also applied. Parameters for such transformations are sampled from a uniform or Gaussian distribution (parameterized by a mean and variance). In our experiments, we vary the data augmentation strength by multiplying the variance of these parameter distributions by a fixed factor, which we dub the *augmentation scaling* factor. The extracted networks are evaluated with the same augmentation parameters. The results of increasing the augmentation strength of the inner objective can be seen in Table 3. We observe that as augmentation strength increases DARTS finds networks with more number of parameters and better test performance. The best test performance is obtained for the network with maximum augmentation for the inner objective. At the same time the one-shot validation error increases when scaling up the augmentation factor, which again enforces the argument that the over fitting of architectural parameters is reduced by this implicit regularizer.

Table 3: Effect of more augmentation on the architecture generalization found by AutoDispNet. The search was conducted on FlyingThings3D (FT) and the final architecture was evaluated on both FT and Sintel. Lower is better.

Aug. scale	One-shot valid EPE	FT test EPE	Sintel test EPE	Params (M)
0.0	4.49	3.83	5.69	9.65
0.1	3.53	3.75	5.97	9.65
0.5	3.28	3.37	5.22	9.43
1.0	4.61	3.12	5.47	12.46
1.5	5.23	2.60	4.15	12.57
2.0	7.45	2.33	3.76	12.25

L_2 regularization. We study the effect of increasing regularization strength on the weights of the network. The results are shown in Table 4. Also in this case best test performance is obtained with the maximum regularization strength. At the same time, the one-shot validation error increases when scaling up the regularization factor, demonstrating again that over fitting of architectural parameters is reduced due to an implicit regularization effect.

Table 4: Effect of more L_2 regularization on the architecture generalization found by AutoDispNet. Lower is better.

L_2 reg. factor	One-shot valid EPE	FT test EPE	Sintel test EPE	Params (M)
$3 \cdot 10^{-4}$	3.95	3.25	6.13	11.00
$9 \cdot 10^{-4}$	5.97	2.30	4.12	13.92
$27 \cdot 10^{-4}$	4.25	2.72	4.83	10.29
$81 \cdot 10^{-4}$	4.61	2.34	3.85	12.16

E Results on Penn Treebank

Here we investigate the effect of more L_2 regularization on the inner objective for searching recurrent cells on Penn Treebank (PTB). We again used a reduced search space with only ReLU and identity mapping as possible operations. The rest of the settings is the same as in [Liu et al., 2019].

We run DARTS search four independent times with different random seeds, each with four L_2 regularization factors, namely $5 \cdot 10^{-7}$ (DARTS default), $15 \cdot 10^{-7}$, $45 \cdot 10^{-7}$ and $135 \cdot 10^{-7}$. Figure 11 shows the median test perplexity after 1600 training epochs of the architectures found by DARTS with the aforementioned L_2 regularization values. As we can see, a stronger regularization factor on the inner objective makes the search procedure more robust. The median perplexity of the discovered architectures gets better as we increase the L_2 factor from $5 \cdot 10^{-7}$ to $45 \cdot 10^{-7}$, while the one-shot validation mean perplexity increases. This observation is similar to the ones on image classification, showing again that properly regularizing the inner objective helps reduce over fitting the architectural parameters.

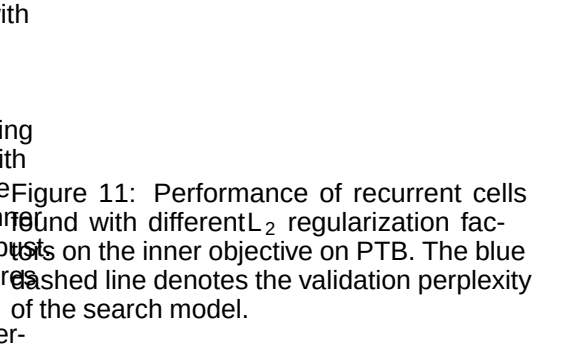


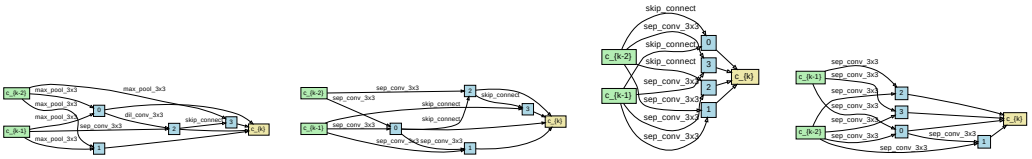
Figure 11: Performance of recurrent cells found with different L_2 regularization factors on the inner objective on PTB. The blue dashed line denotes the validation perplexity of the search model.

Figure 12: Local average of the dominant EV_{max} throughout DARTS search (for different drop path prob. values). Markers denote the early stopping point based on the criterion in Section 3.2.

Figure 13: Effect of L_2 regularization no the EV trajectory. The figure is analogous to Figure 12.

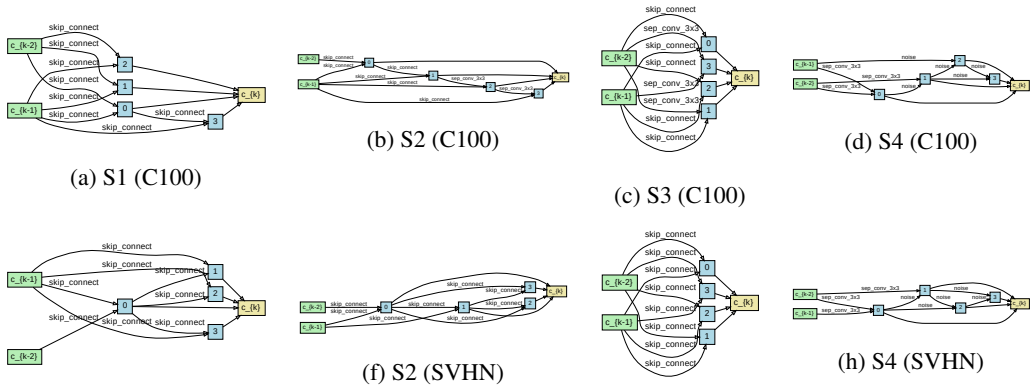
Figure 14: Effect of L_2 regularization (top 3 rows) and augmentation (bottom 3 rows) on the eigenspectrum.

F Discovered cells on search spaces S1-S4 from Section 2.1 on other datasets



(a) S1 (b) S2 (c) S3 (d) S4

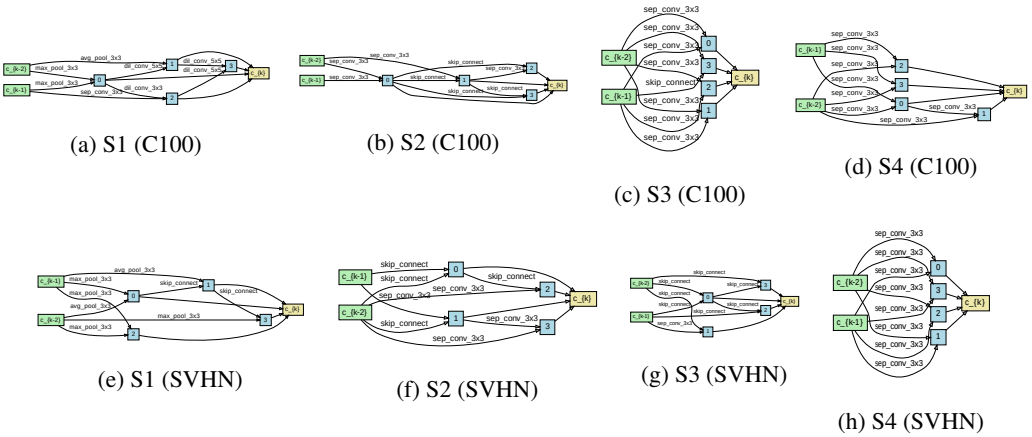
Figure 15: Reduction cells found by DARTS when ran on CIFAR-10 with its default hyperparameters on spaces S1-S4. These cells correspond with the normal ones in Figure 1.



(a) S1 (C100) (b) S2 (C100) (c) S3 (C100) (d) S4 (C100)

(e) S1 (SVHN) (f) S2 (SVHN) (g) S3 (SVHN) (h) S4 (SVHN)

Figure 16: Normal cells found by DARTS on CIFAR-100 and SVHN when ran with its default hyperparameters on spaces S1-S4. Notice the dominance of parameter-less operations such as *skip connection* and *pooling* ops.



(a) S1 (C100) (b) S2 (C100) (c) S3 (C100) (d) S4 (C100)

(e) S1 (SVHN) (f) S2 (SVHN) (g) S3 (SVHN) (h) S4 (SVHN)

Figure 17: Reduction cells found by DARTS on CIFAR-100 and SVHN when ran with its default hyperparameters on spaces S1-S4.