DARTS for Inverse Problems: a Study on Hyperparameter Sensitivity

Jonas Geiping∗
jonas.geiping@uni-siegen.de

Jovita Lukasik∗
jovita@informatik.uni-mannheim.de

Michael Moeller†
michael.moeller@uni-siegen.de

Margret Keuper†
margret.keuper@uni-siegen.de

April 21, 2022

Abstract

Differentiable architecture search (DARTS) is a widely researched tool for the discovery of novel architectures, due to its promising results for image classification. The main benefit of DARTS is the effectiveness achieved through the weight-sharing one-shot paradigm, which allows efficient architecture search. In this work, we investigate DARTS in a systematic case study of inverse problems, which allows us to analyze these potential benefits in a controlled manner. We demonstrate that the success of DARTS can be extended from image classification to signal reconstruction, in principle. However, our experiments also expose three fundamental difficulties in the evaluation of DARTS-based methods in inverse problems: First, the results show a large variance in all test cases. Second, the final performance is highly dependent on the hyperparameters of the optimizer. And third, the performance of the weight-sharing architecture used during training does not reflect the final performance of the found architecture well. Thus, we conclude the necessity to 1) report the results of any DARTS-based methods from several runs along with its underlying performance statistics, 2) show the correlation of the training and final architecture performance, and 3) carefully consider if the computational efficiency of DARTS outweighs the costs of hyperparameter optimization and multiple runs.

1 Introduction

Recent progress in computer vision and related fields has illustrated the importance of suitable neural architecture designs and training schemes He et al. [2015]. Ever deeper and more complex networks show promise, and manual network design is less and less able to explore the desired search spaces. Neural architecture search (NAS) is the task of optimizing the architecture of a neural network automatically without resorting to human selection, scaling to larger search spaces and proposing novel well-performing architectures. NAS, which is an intrinsically discrete problem, has been successfully addressed using black-box optimization approaches such as reinforcement learning Zoph and Le [2017], Zoph et al. [2018] or Bayesian optimization Kandasamy et al. [2018], White et al. [2019], Ru et al. [2020], Lukasik et al. [2021]. However, these approaches are computationally expensive as they require the training of many candidate networks to cover the search space. In contrast, differentiable architecture search (DARTS) Liu et al. [2019] proposes a continuous relaxation of the search problem, i.e. all candidate architectures within a given search space of operations and their connectivity are jointly optimized using shared network parameters while the network also learns to weigh these operations. The final architecture can then simply be deduced by selecting the highest weighted operations. This is appealing as practically good architectures are proposed within a single optimization run. However, previous works such as Zela et al. [2020] also indicate that the proposed

∗Authors contributed equally.
†Authors contributed equally.
results are often sub-optimal, especially when the search space is not well chosen. Specifically, since network weights are randomly initialized, promising operations can have poor initial weights such that the architecture optimization tends to entirely discard them. As a result, the practical relevance of DARTS-proposed architectures depends heavily on network initialization as well as on training hyperparameters. Yet, especially in the context of large-scale computer vision problems such as image classification, a systematic DARTS hyperparameter optimization is hardly affordable.

In this paper, we apply DARTS to inverse problems with the main focus on the analysis of DARTS w.r.t. the impact of domain shifts, training hyperparameter choice and network initialization. Since signal recovery has not received nearly as much attention in the NAS literature as image classification, it allows to study a naive choice of parameters and settings without bias to known results and best practices. In the signal recovery setting, sequential architectures Zhang et al. [2017] yield competitive results when learning to solve inverse problems, such that we can analyze the impact of the complexity of the search space more easily. Specifically, we compare the stability and sensitivity to hyperparameters of DARTS-like architecture optimization in a simple, sequential search space as well as in a non-sequential search space, which we both propose, whereas the latter is based on the search space proposed in Liu et al. [2019]. We investigate two types of one-dimensional inverse problems which allow for extensive experiments for each setting in order to analyze the robustness of DARTS.

We show that DARTS can automatically find well performing architectures, if the search space is well preconditioned. Yet, our study also shows that the performance of DARTS heavily depends on hyperparameter choices. Moreover, DARTS shows a large variance for any set of hyperparameters, such that the suitability of parameters as well as the overall performance can only be judged when considering a large number of runs. Most importantly, we find that the estimated network performance using jointly optimized, shared weights is often not well correlated with the reconstruction ability of the final model after operation selection and re-training, i.e. the continuous relaxation in DARTS seems to be quite loose. In particular, this makes the search for good hyperparameters by optimizing for the DARTS training objective near-impossible. Hyperparameter optimization w.r.t. the final architecture performance on the other hand is even more expensive and seems to increase the variance in the results even further. Yet, overall, our study also shows that DARTS can successfully be applied to inverse problems. Specifically, it improves over the competitive random search baseline by a significant margin, when the search space contains a variety of harmful and beneficial operations. This finding is crucial, since the search space can not always be assumed to be well preconditioned in novel applications.

2 DARTS

We introduce DARTS in a more detailed way in the following, as we build our analysis on technical parts of the differentiable search in DARTS Liu et al. [2019]. While the originally proposed method optimizes so-called cells, which are stacked in order to define the overall neural network architecture, and defines each cell in the form of a directed acyclic graph (DAG), we conduct most parts of our systematic study of the behavior of DARTS on special sequential, and easy-to-interpret meta-architectures to be described below. To exclude that our findings are merely due to this special search space, we also consider experiments resembling the original DARTS setup. Our sequential architecture merely consists of $N$ nodes $x^{(i)}$, where $x^{(0)}$ represents the input data and the result $x^{(i+1)}$ of any layer is computed by applying some operation $o^{(j)}$ to the predecessor node $x^{(j)}$, i.e.,

$$x^{(j+1)} = o^{(j)}(x^{(j)}, \theta^{(j)})$$

(1)

where $\theta^{(j)}$ are the (learnable) parameters of operation $o^{(j)}$. To determine which operation $o^{(j)}$ is most suitable to be applied to the feature $x^{(j)}$, one defines a set of candidate operations $o_t \in \mathcal{O}, t \in \{1, \ldots, |\mathcal{O}| = T\}$ and searches over the continuous relaxation of Equation (1)

$$o^{(j)} = \sum_{t=1}^{T} \beta^{(j)}_{o_t} a_t, \quad \beta^{(j)}_{o_t} = \frac{\exp(a^{(j)}_{o_t})}{\sum_{t'=1}^{T} \exp(a^{(j)}_{o_{t'}})}$$

(2)
where $\alpha = (\alpha_{\alpha_i}^{(j)})$ are architecture parameters that determine the selection of exactly one candidate operation in the limit of $\beta$ becoming binary. Instead of looking for binary parameters directly, the optimization is relaxed to the soft-max of continuous parameters $\alpha$.

DARTS formulates this search as a bi-level optimization problem in which both, the network parameters $\theta = \{\theta^{(j)}\}_{j=1}^N$ and the architecture parameters $\alpha$, are jointly optimized on the training and validation set, respectively, via

$$\min_{\alpha} L_{\text{val}}(\theta(\alpha), \alpha)$$

s.t. $\theta(\alpha) \in \arg\min_{\theta} L_{\text{train}}(\theta, \alpha)$,

where $L_{\text{val}}$ and $L_{\text{train}}$ denote suitable loss functions for the validation and training data. The optimization is done by approximating (4) by one (or zero) iterations of gradient descent, and depends on several hyperparameters such as initial learning rates, learning rate schedules and weight decays for both architecture and model parameters.

At the end of the search, the discrete architecture is obtained by choosing the most likely operation $\hat{o}^{(j)} = \arg\max_{o} \alpha_{o}^{(j)}$ for each node. Subsequently, the final network given by the architecture Equation (1), but using $\hat{o}^{(j)}$ instead of $o^{(j)}$, is retrained from scratch. Thus, the fundamental assumption that justifies the idea of DARTS is that the performance reached by the final network architecture on the validation set (the architecture validation) is highly correlated with the performance of the relaxed DARTS approach obtained in (3) (the one-shot validation). Only then, the architecture found during DARTS optimization can also be expected to perform well after retraining. While previous works (to be summarized in the next section) have studied a search-to-evaluation gap, i.e., the effect that the final network architecture’s performance improves significantly by retraining from scratch, we further investigate whether this assumed correlation between one-shot validation and architecture validation is always given and in how far it depends on the choice of hyperparameters.

3 Related Work

In the last years neural architecture search (NAS) gained ever more interest due to its advantage over time-consuming manual trial-and-error network design. Especially DARTS Liu et al. [2019] took a new step in the NAS area, as it was the first method to introduce gradient-based neural architecture search. By relaxing the operation choices in the network and thus allowing for gradient-based optimization, it shows significant advantages in the search of good neural architectures within only a few GPU days, resulting in finding architectures with impressive performances. Building on this pioneering work NAS research has gained significant momentum for further improvements over the DARTS approach Pham et al. [2018], Liu et al. [2018], Dong and Yang [2019], Cai et al. [2019], Xie et al. [2019], Chen et al. [2019], Akimoto et al. [2019], Xu et al. [2020], He et al. [2020], Chen and Hsieh [2020], Wu et al. [2021], Zhang et al. [2021]. For example Chen et al. [2019] presents an algorithm to progressively increase the depth of the searched architecture during training bridging the gap between search and evaluation performances. To narrow down the vast amount of literature building on the DARTS approach, we focus here only on weight-sharing literature that is in line with our case study.

**Stability of DARTS.** There are only few works investigating the stability of DARTS Zela et al. [2020], Xu et al. [2020], Chu et al. [2020], Chen and Hsieh [2020]. RobustDARTS Zela et al. [2020] track the dominant eigenvalue $\lambda_{\max}$ of the Hessian during the architecture search and implement a regularization and early stopping criterion based on this quantity for a more robust DARTS search. Chen and Hsieh [2020] picks up the relationship between the Hessian during the architecture search and the performance gap during search and evaluation time. They propose a perturbation-based regularization to smooth the validation loss landscape. Xu et al. [2020] find that only connecting partial channels into the operation selection leads to a regularized search to improve the stability. Chu et al. [2020] use a sigmoid activation for the architecture weights instead of softmax to eliminate unfair optimization regarding the skip-connection operation. Yang et al. [2020] analyzes the
contribution of each component in a NAS approach within the DARTS search space. They highlight that a performance-boosting training pipeline, often a result of expert knowledge, is more important for the evaluation of architectures than the search itself. These findings motivate our analysis of the potential benefits of DARTS in a different setting than image classification.

Reconstruction. Previous work on reconstruction of inverse problems via learned approaches has often focused on unrolled optimization schemes, such as unrolled PDHG in Riegler et al. [2016] and Adler and Öktem [2018]. These architectures, also referred to as variational networks Klatzer et al. [2016], Hammernik et al. [2017], are constructed by unrolling existing optimization routines that solve inverse problems and adding learning components in blocks which are either recurrent, as e.g. in Aggarwal et al. [2019] or fully independent as in Hammernik et al. [2018]. In this investigation we will focus on parameterized gradient descent layers which can be seen as the most fundamental building block of these optimization routines.

4 Proposed Search Spaces

The significant advantages in computational efficiency over discrete architecture optimization methods along with the impressive performances of the final architectures have made DARTS and its variants highly attractive for automating the search for well-working neural networks. This framework itself is generic and thus applicable to any field of application, such as inverse problems.

Our following analysis of DARTS for inverse problems will deliberately not be targeting settings that yield good results by design. In contrast, we propose two search spaces with different complexities that allow to analyze the stability and performance of DARTS under varying degrees of difficulty, in ascending order:

- finding a good (linear) sequence of operations from meaningful choices of operations,
- finding a good (linear) sequence of operations where the set of operations to choose from contains good operations as well as harmful operations (the model needs to learn to avoid these),
- finding a good non-linear, acyclic computational graph of operations from meaningful choices of operations (this is the conventional DARTS setting),
- finding a good non-linear, acyclic computational graph of operations, where the set of operations to choose from contains good operations as well as harmful operations (the model need to learn to avoid these).

Such search spaces allow to investigate the properties of DARTS-like methods under various and realistic conditions. Specifically, not for all tasks, we can assume that the set of well-performing, beneficial operations is given or even complete. In such setups, one would ideally want to be able to add new operation candidates to the search space and have the search determine which configuration will work best. Therefore, it is desirable that methods perform reliable even if poor operation choices are available.
4.1 Sequential Search Space

For the simpler, sequential search space, we propose the meta-architecture shown in Figure 1, which should be specifically well-suited for examples of signal recovery from known data formation processes such as blurring and subsampling with noise. From a pre-defined set of operations, we choose operations sequentially before adding the output to a residual branch. Image recovery networks such as DnCNN Zhang et al. [2017] are contained in this meta-architecture. In practice, we search for 10 successive layers. A detailed discussion of the proposed operations is given in Section 4.3. As discussed above, the search space deliberately contains benign as well as harmful operations. This allows the evaluation of the effectiveness of DARTS in any setting via the distinction of two cases: Training on all operations versus training only on beneficial operations. A good architecture search algorithm should reliably find the optimal operations, even when presented with sub-optimal choices.

4.2 Non-Sequential Search Space

For the more complex, non-sequential search space, we construct a cell structure with 5 states, and allow for arbitrary forward connections among the same set of operations as in the sequential setting, but also allowing for a \{zero\} operation, resulting in up to 15 operational connections. The output of the last two states is then concatenated and flattened via a 1D convolution. We utilize two of these cells in succession, so that the depth of this search space with in total 10 nodes is comparable to the sequential search space from above. Figure 2 visualizes an exemplary cell meta architecture during architecture search and the found cell architecture, which is then retrained. Similar to the original DARTS search space, we choose in each cell one operation out of several as a connection between each node in the cell. This setting is more directly comparable to the original DARTS formulation Liu et al. [2019], which contains a cell structure with multiple possible connections between sequential states, allowing for a larger degree in freedom in combining computational results. The selected architecture for retraining then takes the operation between each node with the highest probability.

4.3 Network Operations for Inverse Problems

In both the sequential and non-sequential setting, we search for the optimal architecture that can be defined using operations selected from a defined set \(O_l\). Specifically, we propose to use four operations, two of which are benign and potentially beneficial by design. The first benign operation is motivated from rolled-out-architectures (see e.g. Gregor and LeCun [2010], Schmidt and Roth [2014], Kobler et al. [2017]) and tries to embed model-based knowledge about the recovery problems into the networks architecture. In this paper we consider problems which can be phrased as linear inverse problem, in which the quantity \(x\) ought to be recovered from data \(y = Ax + \text{noise}\) for a linear operator \(A\). While the precise type of algorithm is typically dictated by (smoothness) properties of the regularization, a partially parameterized network-based approach has a lot of freedom to choose...
from template layers based on the mentioned inverse problem \( y = Ax + \text{noise} \)

\[
\arg\min_x D(Au, f),
\]  

(5)

where \( D \) is a data formation term arising from the distribution of noise present, i.e. \( D(v, f) = \frac{1}{2}||v - f||^2 \) for Gaussian noise. This optimization objective yields templates such as a gradient descent layer: \( u^{k+1} = u^k - \tau A^T \nabla_x D(Au^k, f) \), for input \( u^k \) and output \( u^{k+1} \) of a new layer. For suitable chosen \( \tau \), the application of this layer is guaranteed to reduce the objective (5). The gradient layer can be turned into a learnable operation by introducing a learnable mapping \( n(u^k, \theta) \) after the gradient step, 

\[
u^{k+1} = u^k - \tau A^T \nabla_x D(Au^k, f) - \tau n(u^k, \theta),
\]

(6)

as a learnable gradient descent layer in our operation set \( \mathcal{O}_l \). The second benign layer is a fully-learned neural network layer in our operation set \( \mathcal{O}_l \), that learns an appropriate mapping \( n(u^k, \theta) \) without knowledge of the operator \( A \):

\[
u^{k+1} = n(u^k, \theta),
\]

(7)

For both layers, the learnable mapping \( n(u^k, \theta) \) is parametrized by a small convolutional network, consisting of a convolution layer, followed by batch normalization, ReLU and a second convolution layer. These two layers, learnable gradient descent layer and neural network layer, are by design beneficial operations. In contrast to these beneficial layers we also include two negative operations to each operation set; a gradient layer with white Gaussian noise, noise layer, and a roll layer, which rolls the inputs in all dimensions. We provide additional details for these operations in the supp. material. In total, we set \( \mathcal{O}_l = \{ \text{learnable gradient descent}, 2\text{-layer-CNN}, \text{roll}, \text{noise} \} \).

5 Evaluating DARTS for Inverse Problems

In the following, we describe the experimental setting in which we evaluate DARTS for inverse problems. Thereby, we focus on small problem instances to be able to evaluate the framework not once but in several runs such as to evaluate the statistics of the results. This setup also allows to gain insights on the dependence of DARTS’ performance on the chosen hyperparameters.

5.1 Experimental Setup

Data Generation For a fast synthetic test we generate one-dimensional data sampling cosine waves of varying magnitude, amplitude and offset, and search for models to recover these samples from distorted measurements. We consider two distortion processes with varying difficulty: First, Gaussian noise and blurring and second, in addition to these, a subsampling by a factor of 4.

We generate these synthetic one-dimensional cosine data from \( N = 50 \) equally spaced points \( \omega_i \) on the interval \( [-\frac{\pi}{2}, \frac{\pi}{2}] \) with the model

\[ x_i = \cos(f \omega_i + O_x) + O_y \]

for a random frequency \( f \) drawn uniformly from the interval \([0, 2\pi]\) and offsets \( O_x \) and \( O_y \) drawn from a normal Gaussian distribution. Such random drawn waves comprise our ground truth training data. We then generate measured data via the linear operation \( A \) and addition of noise, 

\[ y = Ax + n, \quad n \in \mathcal{N}(0, \sigma_n). \]

These pairs \((y, x)\) represent our training data. We sample new examples on-the-fly during both training and validation, so that no confounding effects of dataset size exist. All validation and training loss evaluations are each based on 2432 randomly drawn samples. The performance of all models is evaluated in terms of their average peak signal to noise ratio (PSNR) on validation data. For all experiments we chose \( \sigma_n = 0.01 \). For the blur experiments, the linear operator \( A \) is a Gaussian blur with kernel size 7 and \( \sigma_b = 0.2 \). For the downsampling experiments, this Gaussian blur is followed by a subsampling by a factor 4.
Hyperparameter Optimization  Our one-dimensional case study allows us to optimize DARTS training hyperparameters with more granularity than it would be possible for image classification tasks. While we run our first experiments using manually tuned hyperparameters (see Appendix for details), we also consider the behavior and stability of DARTS under optimized hyperparameters. We stress that we consider this mainly as an analysis tool - given that NAS itself is a hyperparameter optimization on which we stack another, and acknowledging that this optimization is practically intractable when larger problems are considered. To improve hyperparameters, we apply BOHB Falkner et al. [2018], a Bayesian optimization method with hyperband Li et al. [2018] and run BOHB for 128 hyperband iterations, which is an affordable budget in this one-dimensional data setting. It is important to note that BOHB is not an exhaustive search and thus there are no guarantees for success within our budget or even in general in a way, that it finds the globally optimal hyperparameters for the just mentioned objective. The usage of BOHB as such covers the problem of hyperparameter optimization partially, but in general there is no simple fix of DARTS via hyperparameter search - which is itself a notable statement about the algorithm. In particular we optimize the hyperparameters with respect to first the one-shot validation performance, “BOHB-one-shot”, and second the final architecture performance, “BOHB”. Note here is that hyperparameter search that maximizes the final architecture performance instead of the one-shot validation performance is twice as expensive (on top of the already expensive hyperparameter search), due to the need for retraining.

5.2 Results

We first investigate the performance of DARTS on the simplified sequential search space. For our analysis, we are not only interested in the best found architecture but also in the statistics of the search. We therefore evaluate 75 trials of DARTS as well as baselines such as setting all operations to Learnable Grad. or Net (i.e. learnable gradient descent or 2-layer CNN), picking a random architecture as well as performing a random search within an equal time budget as required by a DARTS run. We summarize results in Table 1.

The results indicate that DARTS works well for inverse problems to propose successful architectures given the complete operation set $O_l$ for both considered data formations, blur and downsampling, and outperforms architectures consisting of only one good operation. When comparing DARTS with all operations versus DARTS with only beneficial operations, the best architectures in both settings perform similarly (with some natural advantage for the search in beneficial operations only). However, if we compare DARTS to a random search approach (random selection of the operation at each layer), the latter one outperforms DARTS when only good operations are considered for both data formations and also improves over the median of the DARTS search in case of blur, at a significantly lower runtime (57 sec. versus 2min. 39 sec. on avg.). When performing random search at an equal time budget (Random search in Table 1), we evaluate 5 randomly selected architectures and report the best. For sequential search spaces that purely consist of benign operations, random search outperforms DARTS with a median PSNR of 22.75 versus 21.6 on blur and 17.56 versus 16.66 on downsampling.

This is different when harmful operations are added. For a search on the full operations set $O_l$, DARTS can clearly outperform this simple baseline. For further analysis, we additionally investigate how many random search runs are needed, to improve over the DARTS median for the set of all operations: Random search needs on average (10 runs) 49 random search steps to improve over the DARTS median of 18.57 PSNR. While this observation is overall motivating, we also observe that the performance of DARTS significantly drops on average as well as in the median when all operations are considered (compared to only using benign operations). Especially on the blurred data the PSNR drops in the median from 21.6 (good ops.) to 18.57 (all ops.). This effect is undesired: ideally, DARTS should be able to reliably filter out harmful operations.

Practically, these experiments also lead to a first interesting result for applied inverse problems: The best found architecture is a hybrid version that mixes both beneficial operations, possibly suggesting that the best way to approach inverse problems are neither plain (convolutional) networks nor pure unrolling schemes.
Table 1: Architecture validation PSNR values found for 1D inverse problems. Shown is the maximal, mean and median PSNR over 75 trials.

| Method                      | Blur  | Downsampling |
|-----------------------------|-------|--------------|
| Learnable Grad. only        |      |              |
| Nets only                   | 17.45| 16.36        |
| Good ops                    | 14.35| 13.24        |
| Runtime                     | 0:57  |              |
| Random                      | 24.64| 21.56        |
| Random Search               | 24.04| 22.85        |
| Architecture Val. (PSNR)    | Max. | Mean         |
|                             | Med. |              |
|                             |      |              |
|                                  | 16.49| 16.24        |
|                                  | 14.35| 13.24        |
|                                  | 14.05|              |
|                                  |      |              |
|                                  | 17.66| 16.66        |
|                                  | 16.03| 15.35        |
|                                  | 15.35| 14.35        |
|                                  | 14.05|              |
|                                  |      |              |
|                                  | 17.56| 16.78        |
|                                  | 16.74| 15.01        |
|                                  | 14.05|              |
|                                  |      |              |
|                                  | 2:39 |              |
|                                  |      |              |
|                                  | 14.05|              |
|                                  |      |              |
|                                  | 2:55 |              |

Table 2: Architecture validation PSNR values for 1D inverse problems for the non-sequential search space. Shown is the maximal, mean and median PSNR over 100 trials.

| Method                      | Blur  | Downsampling |
|-----------------------------|-------|--------------|
| Learnable Grad. only        |      |              |
| Nets only                   | 13.19| 12.41        |
| Good ops                    | 11.30| 8.89         |
| Runtime                     | 9.59  |              |
| Random                      | 15.34| 13.08        |
| Random Search               | 16.20| 11.29        |
| Architecture Val. (PSNR)    | Max. | Mean         |
|                             | Med. |              |
|                             |      |              |
|                                  | 12.38| 11.30        |
|                                  | 8.89 |              |
|                                  |      |              |
|                                  | 13.50| 13.63        |
|                                  | 13.07| 13.06        |
|                                  |      |              |
|                                  | 14.02| 13.22        |
|                                  | 10.22|              |
|                                  | 10.43|              |
|                                  |      |              |
|                                  | 14.05| 13.15        |
|                                  | 11.88|              |
|                                  | 5.72 |              |
|                                  |      |              |
|                                  | 8.61 |              |
|                                  | 8.37 |              |
|                                  | 3.21 |              |

Since the original DARTS formulation in Liu et al. [2019] contains a cell structure with multiple possible connections between sequential states, allowing for a larger degree of freedom in combining computational results, it is a-priori conceivable that some of the stability of DARTS could be conferred through this structure. Therefore, we now analyze the DARTS performance on the non-sequential DARTS like search space exemplified in Figure 2. Table 2 however shows that this wider search space does not improve the overall performance. Indeed the non-sequential search space hampers not only the DARTS search significantly but also all other approaches, resulting in lower architecture performances for both data formations. In this setup, the Nets only baseline, that uses the 2-layer CNN for all operations, performs best. As above, we observe a significant drop in the performance of DARTS when harmful operations are included in the search space. In this case, as before, DARTS can significantly outperform the random baseline but not reliable determine the obviously best operation. As the overall performance in this non-sequential search space is lower than in the sequential search space, we consider only the latter one in the following.

From a theoretical perspective, we argue that DARTS should be able to determine which operations are harmful: If we assume that the validation accuracy during the optimization correlates with the validation accuracy of the final architecture, harmful operations should be excluded early on in the optimization process. Therefore, in the following section, we study this correlation and investigate whether the behavior can be improved by optimizing training hyperparameters.

### 5.3 Correlation of Architecture and DARTS Performance

Figure 3 takes a closer look at the trials considered in Table 1, scattering the values of all trials separately with architecture performance (y-axis), which is computed after retraining the final architecture versus the direct validation performance of the one-shot architecture (x-axis). We also plot a regression line over all trials and report the correlation of all trials in the legend, showing the linear fit has limited expressiveness. As already discussed, the correlation of these quantities is a fundamental assumption of DARTS. However, this first experiment indicates a correlation problem: The assumption that a better one-shot validation implies a better architecture validation does not always seem to be true. Yet these plots show that DARTS’ behavior is highly problem-dependent:
The *downsampling* dataset (right), shows that, although the mean value of DARTS can be non-optimal, search performance and architecture performance are weakly correlated, even if the best architecture only has average search performance. The closely related *blur* dataset (left) shows an entirely different behavior with different “failure” cases, from which we can observe with the given hyperparameters that 1) either DARTS proposes architectures with low (one-shot) search validation PSNR (i.e. it fails), or that 2) DARTS works but does not predict a useful architecture (low architecture validation PSNR), or that 3) DARTS does predict a useful architecture, but is unrelated to its search performance. Only the best proposed architectures perform well in both. To further analyze the correlation, we investigate DARTS behavior with different training hyperparameters.

Figure 3: Scatter plot corresponding to Table 1 showing architecture PSNR (y-axis) plotted against 1-shot validation PSNR (i.e. the validation performance on the DARTS objective). Left: Blur. Right: Downsampling.

Table 3: Architecture validation PSNR values for 1D inverse problems with different hyperparameter. Shown is the maximal, mean and median PSNR over 75 trials.

| Data    | Hyperparameters | Architecture Validation (PSNR) | Good Ops. | All Ops. |
|---------|-----------------|--------------------------------|-----------|----------|
|         |                 | Max. | Mean | Med. | Max. | Mean | Med. |
| Blur    | H1              | 23.46 | 21.56 | 21.60 | 22.86 | 15.64 | 18.57 |
|         | H2              | 23.46 | 21.43 | 21.63 | 23.10 | 16.77 | 19.88 |
|         | BOHB-one-shot-Blur | 22.83 | 20.86 | 20.75 | 22.47 | 15.57 | 18.04 |
|         | BOHB-one-shot-DS | 22.33 | 20.65 | 20.96 | 22.41 | 14.43 | 14.41 |
|         | BOHB-Blur       | **23.57** | **22.05** | **22.38** | 22.94 | 12.76 | 8.21 |
| Downsampling | H1              | 18.03 | 16.36 | 16.66 | 18.01 | 15.39 | 16.12 |
|         | H2              | 18.20 | 16.57 | 16.78 | 17.82 | **15.93** | **16.21** |
|         | BOHB-one-shot-Blur | **18.42** | **16.83** | **16.95** | 17.73 | 14.36 | 14.57 |
|         | BOHB-one-shot-DS | 17.51 | 15.33 | 15.84 | **18.12** | 12.36 | 13.65 |
|         | BOHB-Blur       | 18.26 | 14.63 | 15.93 | 17.91 | 15.04 | 15.44 |

We evaluate DARTS using 5 different training hyperparameter set; two are chosen manually, H1 and H2 (H1 are the hyperparameters used in 5.2), whereas the other three are tuned using BOHB, as described in Section 5.1. We use BOHB to tune hyperparameters for the one-shot validation performance for both *blur* (BOHB-one-shot-Blur) and *downsampling* (BOHB-one-shot-DS), individually, and also to target the final validation performance for *blur* (BOHB-Blur). The DARTS search results for different training hyperparameters are given in Table 3. For additional visualization, we plot the results for all BOHB found hyperparameter trials in Figure 4. As we can see, the correlation for both data formations increases with the corresponding BOHB-one-shot tuned hyperparameters, with also a higher range of the search validation PSNR. This experiment also shows a rather surprising outcome: In the case of *blur*, the average performance is on par with the manually chosen hyperparameters H1 and H2, whereas the performance for *downsampling* decreases, especially when all operations are considered. In addition, the best architecture PSNR over 75 trials decreases in both cases. Overall, the apparent stabilization via optimization of the search loss removes not only negative, but also positive outliers. Hyperparameters optimized for one dataset do not transfer well to the other. Using BOHB to target the final validation performance for *blur* (BOHB-Blur) instead of the one-shot validation performance has also a positive impact on the one-shot validation and
Figure 4: Scatter plot corresponding to Table 3 with BOHB-optimized hyperparameters, showing architecture PSNR (y-axis) plotted against one-shot validation PSNR (x-axis). Top (left): Blur with hyperparameters \textit{BOHB-one-shot-Blur}. Top (right): Downsampling \textit{BOHB-one-shot-DS}. Bottom (left): Blur with hyperparameters \textit{BOHB-Blur}. Bottom (right): Downsampling with hyperparameters \textit{BOHB-Blur}.

architecture validation correlation, compared to the manually chosen hyperparameters H1 and H2 in Figure 3, but not to the same amount as for the BOHB-one-shot hyperparameters. However, these hyperparameters successfully increase the max. architecture performance. Overall, hyperparameters optimized with BOHB on the one-shot validation have to be considered with caution. This can be seen by cross-checking their performance, i.e. evaluating the BOHB-one-shot-Blur hyperparameters for Downsampling and the BOHB-DS hyperparameters for blur. For the case of blur and all operations in Table 3, the dedicated BOHB-one-shot-Blur hyperparameters are significantly more stable (measuring median PSNR) than the BOHB-one-shot-DS hyperparameters, although their maximal PSNR is very close. When changing the domain to downsampling, the exact opposite holds: BOHB-one-shot-Blur hyperparameters improve over BOHB-one-shot-DS hyperparameters in terms of stability. Note that this could be due to both, the missing correlation between one-shot and architecture validation as well as the missing guarantee of any Bayesian search to find the optimal hyperparameters. In addition, Table 3 even demonstrates that the manual hyperparameters H1 and H2 lead to a better average performance compared with BOHB tuned hyperparameters.

In conclusion we find two schools of thought when evaluating the performance of DARTS. For maximal performance, we should understand DARTS as a component in a larger search that proposes trial architectures. For average performance, and immediate performance with a single DARTS run, we should be optimizing the search performance and maximize its correlation with architecture performance - although as our experiments show, this is non-trivial even when searching for these hyperparameters in an automated fashion. We stress that the two directions are not at odds with each other, yet problems can arise in the literature when comparing proposed improvements of DARTS across both. Some algorithmic improvements of DARTS are more likely to improve best-case performance, whereas others are more likely to impact single trial stability. If these two are not carefully compared, then best-case results, which do provide better benchmark numbers, can appear to supersede stability results. Here, we discuss this effect for a simplified case study, but for large-scale DARTS in image classification, where trials are expensive and fixed random seeds are tempting, such a dichotomy makes it fairly difficult to evaluate and classify the manifold improvements of DARTS.
5.4 Improving the Initialization

Several works, such as Zela et al. [2020], investigate the instability of the the bi-level approximation of DARTS w.r.t. the weight initialization; the random initialization of the network weights can cause promising operations having poor initialization and thus tend to be entirely discarded during the architecture search. We evaluate the impact of this initialization by modifying the DARTS search, such that it only has to search for the optimal architecture parameters to build the resulting architecture. For this DARTS-single approach we pre-train the operations \{learnable gradient descent\} and \{2-layer-CNN\} as baseline architectures consisting only of each operation respectively and keep the weights fixed. This is generally only possible for the feed-forward architectures that we consider and requires only a weak specialization between layers. Therefore, we avoid the random initialization of the operations weights in the DARTS search and thus can evaluate its effect.

Figure 5 shows the results of DARTS-single search with BOHB-optimized hyperparameters for all operations. Notably, BOHB-optimized hyperparameters for the DARTS-single one-shot validation (Figure 5 left) lead to a positive impact on the correlation of the one-shot and architecture validation PSNR using DARTS-single and to a negative impact for DARTS. In addition, when comparing DARTS and DARTS-single with their hyperparameters being individually optimized with BOHB with respect to their one-shot validation (Figure 5 right), DARTS finds a higher architecture validation PSNR than DARTS-single, whereas DARTS-single becomes more robust against possible outliers, making this search less sensitive.

6 Conclusions

In this paper we analyzed DARTS in a systematic study on one-dimensional inverse reconstruction problems. In this setting, we show that DARTS improves over a random search baseline by a significant margin, especially if the available set of beneficial operation is not determined in advance. In our analysis, we make the following findings: While it is possible to find well-performing architectures using DARTS, multiple runs of the same setting yield a high variance. Moreover, the ability to find well-performing architectures is highly dependent on the specific choice of hyperparameters. Unfortunately, judging the success of any DARTS-based model right after the one-shot training is difficult, since a strong correlation to the actual architecture performance is missing. As such, even automatic hyperparameter searches such as BOHB cannot faithfully be applied to the one-shot loss. Therefore, we emphasize for the future the necessity to (1) look at a full statistical evaluation of DARTS performances over multiple trials, in all applications where this is feasible, (2) show a reasonable correlation between the search and final architecture performances for any method that reports improved results based on a more faithful minimization of the one-shot DARTS objective, and (3) carefully considering the costs of multiple runs and meta-optimization of hyperparameters when aiming for computational advantages of DARTS over zero-order NAS methods.
Acknowledgments

JL and MK acknowledge support by the German Federal Ministry of Education and Research Foundation via the project DeToL.

References

Jonas Adler and Ozan Öktem. Learned Primal-Dual Reconstruction. *IEEE Transactions on Medical Imaging*, 37(6):1322–1332, June 2018. ISSN 1558-254X. doi: 10.1109/TMI.2018.2799231.

Hemant K. Aggarwal, Merry P. Mani, and Mathews Jacob. MoDL: Model-Based Deep Learning Architecture for Inverse Problems. *IEEE Transactions on Medical Imaging*, 38(2):394–405, February 2019. ISSN 1558-254X. doi: 10.1109/TMI.2018.2865356.

Youhei Akimoto, Shinichi Shirakawa, Nozomu Yoshinari, Kento Uchida, Shota Saito, and Kouhei Nishida. Adaptive stochastic natural gradient method for one-shot neural architecture search. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 171–180. PMLR, 2019. URL http://proceedings.mlr.press/v97/akimoto19a.html.

Han Cai, Ligeng Zhu, and Song Han. ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware. *arXiv:1812.00332 [cs, stat]*, February 2019. URL http://arxiv.org/abs/1812.00332.

Xiangning Chen and Cho-Jui Hsieh. Stabilizing differentiable architecture search via perturbation-based regularization. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 1554–1565. PMLR, 2020. URL http://proceedings.mlr.press/v119/chen20f.html.

Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 1294–1303. IEEE, 2019. doi: 10.1109/ICCV.2019.00138. URL https://doi.org/10.1109/ICCV.2019.00138.

Xiangxiang Chu, Tianbao Zhou, Bo Zhang, and Jixiang Li. Fair DARTS: eliminating unfair advantages in differentiable architecture search. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XV*, volume 12360 of *Lecture Notes in Computer Science*, pages 465–480. Springer, 2020. doi: 10.1007/978-3-030-58555-6_28. URL https://doi.org/10.1007/978-3-030-58555-6_28.

Xuanyi Dong and Yi Yang. Searching for A Robust Neural Architecture in Four GPU Hours. *arXiv:1910.04465 [cs]*, October 2019. URL http://arxiv.org/abs/1910.04465.

Stefan Falkner, Aaron Klein, and Frank Hutter. BOHB: Robust and Efficient Hyperparameter Optimization at Scale. In *International Conference on Machine Learning*, pages 1437–1446. PMLR, July 2018. URL http://proceedings.mlr.press/v80/falkner18a.html.

Karol Gregor and Yann LeCun. Learning fast approximations of sparse coding. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, page 399–406, 2010.

Kerstin Hammernik, Tobias Würfl, Thomas Pock, and Andreas Maier. A Deep Learning Architecture for Limited-Angle Computed Tomography Reconstruction. In *Bildverarbeitung für die Medizin 2017*, Informatik aktuell, pages 92–97. Springer Berlin Heidelberg, 2017. ISBN 978-3-662-54345-0.
Kerstin Hammernik, Teresa Klatzer, Erich Kobler, Michael P. Recht, Daniel K. Sodickson, Thomas Pock, and Florian Knoll. Learning a variational network for reconstruction of accelerated MRI data. *Magnetic Resonance in Medicine*, 79(6):3055–3071, June 2018. ISSN 1522-2594. doi: 10.1002/mrm.26977.

Chaoyang He, Haishan Ye, Li Shen, and Tong Zhang. Milenas: Efficient neural architecture search via mixed-level reformulation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020*, Seattle, WA, USA, June 13-19, 2020, pages 11990–11999. IEEE, 2020. doi: 10.1109/CVPR42600.2020.01201. URL https://doi.org/10.1109/CVPR42600.2020.01201.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *arXiv:1512.03385 [cs]*, December 2015. URL http://arxiv.org/abs/1512.03385.

Kirthevasan Kandasamy, Willie Neiswanger, Jeff Schneider, Barnabás Póczos, and Eric P. Xing. Neural architecture search with bayesian optimisation and optimal transport. In *NeurIPS*, 2018.

Teresa Klatzer, Kerstin Hammernik, Patrick Knobelreiter, and Thomas Pock. Learning joint demosaicing and denoising based on sequential energy minimization. In *2016 IEEE International Conference on Computational Photography (ICCP)*, pages 1–11, May 2016. doi: 10.1109/ICCPHOT.2016.7492871.

Erich Kobler, Teresa Klatzer, Kerstin Hammernik, and Thomas Pock. Variational networks: Connecting variational methods and deep learning. In *Pattern Recognition*, Lecture Notes in Computer Science, pages 281–293. Springer, 2017. ISBN 978-3-319-66708-9. doi: 10.1007/978-3-319-66709-6_23.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization. *Journal of Machine Learning Research*, 18(185):1–52, 2018. ISSN 1533-7928. URL http://jmlr.org/papers/v18/16-558.html.

Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan L. Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part I*, volume 11205 of Lecture Notes in Computer Science, pages 19–35. Springer, 2018. doi: 10.1007/978-3-030-01246-5_2. URL https://doi.org/10.1007/978-3-030-01246-5_2.

Hanjiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable Architecture Search. *arXiv:1806.09055 [cs, stat]*, April 2019. URL http://arxiv.org/abs/1806.09055.

Jovita Lukasik, David Friede, Arber Zela, Frank Hutter, and Margret Keuper. Smooth variational graph embeddings for efficient neural architecture search, 2021.

Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 4092–4101. PMLR, 2018.

Gernot Riegler, Matthias Rüther, and Horst Bischof. Atgv-net: Accurate depth super-resolution. In *European Conference on Computer Vision*, pages 268–284. Springer, 2016. URL http://link.springer.com/chapter/10.1007/978-3-319-46487-9_17.

Binxin Ru, Xingchen Wan, Xiaowen Dong, and Michael Osborne. Neural architecture search using bayesian optimisation with weisfeiler-lehman kernel. *ArXiv*, abs/2006.07556, 2020.
A Hyperparameters

In this section we show the hyperparameters used for our experiments in the main paper. In Table 4 the manually chosen hyperparameters H1 and H2 from Table 2 in the main paper. In addition Table 5 lists all BOHB optimized hyperparameters for the data formations blur and downsampling as well as the hyperparameters optimized for the final architecture accuracy and also for the DARTS-single method; the search range for the BOHB search is given in the second column.

B Non-sequential Search Space Stability

In this section we additionally investigate the hyperparameter stability of the non-sequential search space from Section 4.2 for the blur data formation as well as visualize architectures found by DARTS for two different operation sets.
Table 4: Manually chosen hyperparameters H1 and H2

| Hyperparameter     | H1         | H2         |
|--------------------|------------|------------|
| Param. learning rate | 0.001      | 0.001      |
| Param. weight decay  | 1e-8       | 1e-8       |
| Param. warm up      | False      | False      |
| Alpha learning rate | 0.001      | 0.0001     |
| Alpha weight decay  | 0.001      | 0.0001     |
| Alpha warm up       | True       | True       |
| Alpha scheduler     | Linear     | Linear     |
| Alpha optimizer     | Gradient Descent | Gradient Descent |

Table 5: BOHB optimized hyperparameters for different data formations, objectives and methods.

| Hyperparameter                  | Search Range | BOHB-Blur | BOHB-DE | BOHB-Arch-Blur | BOHB-DARTS-single |
|---------------------------------|--------------|-----------|---------|---------------|-------------------|
| Param. learning rate            | [1e-05,1]    | 0.0014232405 | 0.6020448382 | 0.0020882283 | 0.0014232405      |
| Param. weight decay             | [1e-08,0.1]  | 8.61e-07   | 5.84e-08 | 4.4e-08       | 8.61e-07          |
| Param. warm up                  | [True,False] | False      | True    | False         | False             |
| Alpha learning rate             | [1e-05,0.1]  | 0.0000003746 | 8.4315e-05 | 0.02501237350395577 |
| Alpha weight decay              | [1e-05,0.1]  | 0.0000022776 | 0.00127425783 | 1.390640076880444e-05 |
| Alpha warm up                   | [True,False] | False      | True    | True          | False             |
| Alpha scheduler                 | [None,Linear]| Linear    | Linear  | Linear        | None              |
| Alpha optimizer                 | [Adam, Gradient Descent] | Adam | Gradient Descent | Adam | Adam       |

Table 6: BOHB optimized hyperparameters for the DARTS-like wide non-sequential search space for data formation blur and different objectives.

| Hyperparameter                  | Search Range | BOHB-Wide-Blur | BOHB-Wide-Arch-Blur |
|---------------------------------|--------------|---------------|---------------------|
| Param. learning rate            | [1e-05,1]    | 0.0050969066 | 0.0037014752        |
| Param. weight decay             | [1e-08,0.1]  | 2.423e-07    | 1.4573e-06          |
| Param. warm up                  | [True,False] | False        | False               |
| Alpha learning rate             | [1e-05,0.1]  | 1.32499e-05  | 0.0012395056        |
| Alpha weight decay              | [1e-05,0.1]  | 0.0010171142 | 0.0002855732        |
| Alpha warm up                   | [True,False] | False        | False               |
| Alpha scheduler                 | [None,Linear]| None         | None                |
| Alpha optimizer                 | [Adam, Gradient Descent] | Adam | Adam               |
B.1 Hyperparameter Stability

Figure 6: Scatter plot for the non-sequential DARTS search space corresponding to Table 8, with hyperparameters $H1$ (left) and $H2$ (right), showing architecture PSNR (y-axis) plotted against 1-shot validation PSNR (i.e. the validation performance on the DARTS objective).

To investigate the hyperparameter stability further for this non-sequential search space, we conduct experiments using the same BOHB-optimized hyperparameters as in Section 5.3 and additionally included BOHB-optimized hyperparameters for this non-sequential search space for first targeting the one-shot validation performance (BOHB-Non-Seq-one-shot-Blur) and second targeting the final architecture performance (BOHB-Non-Seq-Blur). Table 8 however shows similar results as in Section 5.3: changing the hyperparameters in this non-sequential DARTS-like search space does not improve the stability of the search process. Figure 6 shows all trials for the non-sequential search space for the manually chosen hyperparameters $H1$ and $H2$. This plot clarifies further, that the search space change does not improve the DARTS search process. The correlation between the one-shot validation and the architecture validation even becomes negative. Yet, these plots also show 2 different “failure” cases for both operations sets, only beneficial operations and all operations, and both data formations: The validation PSNR is stable, whereas the architecture validation performance is clustered in two different regions, one being very low and the other being around 15 PSNR. Note, the mean architecture validation PSNR for all operations in the sequential search space from Section 5.3 in Table 3 is also around 15 PSNR.

For additional visualization, we also display the results using BOHB found hyperparameters in the sequential search space in Figure 7 as well as BOHB found hyperparameters tuned for this non-sequential search space in Figure 8. However, hyperparameter search for the non-sequential search space via BOHB on both the one-shot validation performance and the architecture performance as a target, does not actually improve the stability of the search for this new search space, as demonstrated in Figure 8. Accordingly we find on the one hand that the findings in the main paper regarding stability with DARTS for inverse problems translate to a cell-based search space and on the other hand (investigating the overall performance metrics for both search spaces), that the sequential search space appears to be a helpful prior for architecture search for inverse problems, given that its PSNR scores are overall higher.

C Visualizations

In this section we visualize in Figure 9 two found architectures using the H1 hyperparameters for the operation sets “all operations” and “only good operations” for the data formation blur in the

| Table 7: General Hyperparameters |
|----------------------------------|
| Hyperparameter | Default Value |
| Epochs         | 50            |
| Batch size     | 128           |
| Noise Level    | 0.10          |
Table 8: DARTS-like wide architecture validation PSNR values found in the 1D inverse problems setting with cosine data for the DARTS-like search space. Shown is the maximal, mean and median PSNR over 75 trials.

| Data   | Hyperparameters | Architecture Validation (PSNR) |  |
|--------|-----------------|-------------------------------|---|
|        | Good Ops.       | All                           |  |
|        | Max. | Mean | Med. | Max. | Mean | Med. | Max. | Mean | Med. |
| Blur   | H1     | 15.34 | 13.08 | 12.51 | 16.15 | 13.56 | 13.73 |  |
|        | H2     | 15.38 | 13.17 | 12.52 | 16.28 | 14.11 | 15.58 |  |
|        | BOHB-one-shot-Blur | 16.38 | 13.25 | 12.76 | 16.71 | 11.73 | 11.8 |  |
|        | BOHB-one-shot-DS | 14.93 | 12.73 | 12.44 | 15.86 | 12.37 | 11.72 |  |
|        | BOHB-Blur | 16.50 | 8.84 | 8.09 | 16.82 | 13.96 | 15.5 |  |
|        | BOHB-Non-Seq-one-shot-Blur | 16.74 | 9.73 | 8.11 | 17.03 | 13.45 | 15.42 |  |

Figure 7: Scatter plot for the non-sequential DARTS search space on blur with BOHB-optimized hyperparameters from the DARTS-like search space, showing architecture PSNR (y-axis) plotted against one-shot validation PSNR (x-axis). Top (Left): Blur with hyperparameters BOHB-one-shot-Blur. Top (Right): Blur with hyperparameters BOHB-one-shot-DS. Bottom: Blur with hyperparameters BOHB-Blur

non-sequential search space from the experiments in Section 5.2.

D Computational Setup

All experiments in the main body were run on a single Nvidia GTX 2080ti graphics card of which two were utilized. The hyperparameter tuning with BOHB was conducted on a single Nvidia GTX 1080 Ti graphics card.
Figure 8: Scatter plot for the non-sequential DARTS search space on blur with hyperparameters searched for the DARTS-like search space, showing architecture PSNR (y-axis) plotted against one-shot validation PSNR (x-axis). Left: Blur with hyperparameters BOHB-Non-Seq-one-shot-Blur. Right: Blur with hyperparameters searched for the final architecture performance BOHB-Non-Seq-Blur.

Figure 9: Found architectures in the DARTS-like search space for two different operation sets for the data formation blur. Hyperparameter H1 is used for these searches. Top: all operations. Bottom: only beneficial operations.