LoRA-DARTS: Low Rank Adaptation for Differentiable Architecture Search

Arjun Krishnakumar[1,*] **Abhash Kumar Jha**[1,*] **Shakiba Moradian**[1] **Martin Rapp**[2] **Frank Hutter**[3,1]

[1]Department of Computer Science, University of Freiburg
[2]Bosch Center for Artificial Intelligence
[3]ELLIS Institute Tübingen
[*]Equal contribution.

Abstract Gradient-based one-shot neural architecture search (NAS) methods, such as Differentiable Architecture Search (DARTS), have emerged as computationally feasible techniques to explore large search spaces. However, DARTS still suffers from failure modes, such as choosing architectures that prefer skip connections over learnable operations. In this work, we propose that the use of a low-rank adaptation (LoRA) of the weights of the candidate operations can address this failure mode without introducing new regularization terms or significant changes to the DARTS search technique. The code for our work is available at https://github.com/automl/LoRA-DARTS.

1 Introduction

Differentiable Architecture Search (DARTS) (Liu et al., 2019) is a gradient-based neural architecture search (NAS) method that emerged as a viable solution to one of the biggest challenges in NAS – the amount of compute required to effectively search a space of architectures. By training a *supernet*, the sub-networks of which span the entire search space of architectures, and simultaneously learning the strengths of the operations in it, DARTS sped up the search by orders of magnitude compared to the reinforcement learning-based (Zoph and Le 2017; Baker et al. 2017; Pham et al. 2018; Zoph et al. 2018) or blackbox search-based methods (Swersky et al. 2013; Fujino et al. 2017; Kandasamy et al. 2018; Liu et al. 2018; Real et al. 2019) which were prevalent at the time. The idea of training a supernet that subsumes all the architectures in the search space has since gained popularity, with several methods directly improving upon DARTS (Chen et al. 2021b; Dong and Yang 2019) and others training the supernet and then performing blackbox search on it to discover a pareto front of architectures for multiple objectives (Chen et al. 2021a; Cai et al. 2020). However, DARTS is particularly prone to a significant failure mode: it often discovers architectures that are dominated by skip connections. We show that using low-rank adaptation (LoRA) of the parametric candidate operations in the supernet can mitigate this issue. Further, we apply the same method to a weight-entangled DARTS space, following Sukthanker et al. (2023), and show that performance does not deteriorate.

2 Preliminaries

2.1 DARTS Optimization Problem

DARTS defines a search space of possible architectures using a directed acyclic graph (DAG), where each node represents a feature map, and each edge represents a possible operation (e.g., convolution, pooling). It relaxes the choice of operations by representing each edge as a weighted sum of all possible operations. If the candidate operations $\mathcal{O}$ are some function of the form $o(\cdot)$ to be applied

to a feature map x, the continuous relaxation is achieved by considering a mixed operation $\bar{o}_{ij}$ on edge (i, j) as follows:

$$\bar{o}_{ij}(x) = \sum_{o \in \mathcal{O}} \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o' \in \mathcal{O}} \exp(\alpha_{o'}^{(i,j)})} o(x),$$

where $\alpha^{(i,j)}$ is a vector of dimension $|\mathcal{O}|$, indicating the strength of each candidate operation, i.e., the architectural parameters.

The training process utilizes a bilevel optimization framework. The outer optimization level targets the architectural parameters to minimize the validation loss, whereas the inner level optimizes the network weights to minimize the training loss:

$$\min_{\alpha} \ \mathcal{L}_{val}(w^*(\alpha), \alpha) \qquad \text{s.t.} \quad w^*(\alpha) = \arg\min_{w} \ \mathcal{L}_{train}(w, \alpha) \tag{1}$$

Several works have pointed out flaws or *failure modes* in the DARTS method. RobustDARTS (Zela et al., 2020) demonstrated that DARTS suffers from performance degradation due to running into sharp local minima in architecture space, such that the final architecture discretization step yields poor results. FairDARTS (Chu et al., 2020) argues that this failure mode occurs due to the unfair advantage conferred upon non-parametric operations. Furthermore, OLES (Jiang et al., 2024) corroborates the issue of overfitting by showing that even after removing skip-connections from the search space, other operations still overfit during the training phase of DARTS.

2.2 LoRA - Low Rank Adaptation of Large Language Models

LoRA (Hu et al., 2021) is a solution that was introduced to make finetuning of large language models (LLMs) more computationally feasible. The update for a pre-trained weight matrix $W_0 \in \mathbb{R}^{d \times k}$ is represented by a low-rank decomposition $W_0 + \Delta W = W_0 + BA$, where $B \in \mathbb{R}^{d \times r}$, $A \in \mathbb{R}^{r \times k}$, with rank $r \ll \min(d, k)$. W_0 is frozen and receives no gradient updates, while A and B contain trainable parameters. Accordingly, the forward pass of a linear layer $h = W_0 x$ becomes:

$$h = W_0 x + \Delta W x = W_0 x + BAx \tag{2}$$

While LoRA is primarily used with linear layers in transformers, it can be used with convolutional layers, too. Indeed, this is how we leverage LoRA in the convolutional DARTS search space.

2.3 Weight Entanglement

Several works, such as OFA (Cai et al., 2020) and AutoFormer (Chen et al., 2021a), have introduced *weight entanglement* (WE) as an alternative to *weight sharing* (WS), which is more commonly applied in gradient-based one-shot methods such as DARTS. WS shares the weight matrices of an operation with an exponential number of subnetworks in the supernet that use that operation. WE goes a step further by using a subset of the weights of an operation from a larger weight matrix for another operation of the same kind. Consider, for example, two convolutions with kernel sizes 3 and 5. In the WS framework, they both have their own matrices of sizes 3×3 and 5×5. In the WE framework, the smaller convolution uses a 3×3 slice from the center of the larger 5×5 convolution matrix as its own weight. We follow Sukthanker et al. (2023) in implementing WE in the supernet before applying DARTS to it.

3 Methodology

Several works (Liang et al. 2019; Zela et al. 2020) have introduced techniques to early-stop the training of the supernet to avoid performance collapse. This trend is consistent with the assertion of Jiang et al. (2024) that the domination of skip connections is a direct result of the parametric

operations overfitting the training data. We adopt this perspective and tackle the problem of overfitting on the training data using low-rank adaptation of the weights.

Our method requires two changes to be made to existing DARTS-based NAS strategies. First, the supernet is modified to accommodate low-rank adaptation of weights, or *LoRA modules*, as discussed in Section 3.1. Then, we warm-start the model for a few epochs before activating these low-rank modules. More about this in Section 3.2.

3.1 Modifications to the Supernet

We rely on low-rank adaptation of the weights of the candidate operations in the supernet to avoid overfitting the training dataset. In the supernet, candidate operations are designed with an option to activate learnable low-rank modules at any point in the optimization loop. Until activated, these modules function as standard operations with learnable parameters. In all our experiments, we set the rank of these LoRA modules to 1 - the lowest possible value - and do not tune it in any fashion. The memory overhead incurred by the LoRA parameters is negligible.

3.1.1 LoRA Layers for Weight-entangled Operations. Applying LoRA layers to weight-entangled operations constitutes a special case. In the regular case, the candidate operations have their own learnable weights and associated LoRA parameters. In the weight-entanglement case, however, the candidate operations subsample their weights from the largest kernel/weight of the same kind. Accordingly, in spaces with weight-entanglement, we use *one* LoRA layer with dimensions that match the largest kernel of an operation.

3.2 Warm-starting the Supernet

To give the parametric operations a chance to learn meaningful representations first, we allow them to train full rank weight updates for a fixed number of epochs (10, in our case). In this phase, the architectural parameters are frozen and all the operations are assigned equal weight. After warm-starting, the weight matrices of the candidate operations are frozen, and their respective LoRA modules are activated, allowing them to learn the low rank weight updates instead.

4 Experiments

We begin in Section 4.1 by assessing the robustness of LoRA-DARTS to non-parametric operations using the RobustDARTS search spaces. Then, in Section 4.2, we apply the method to the DARTS search space. All experiments are repeated with 3 seeds.

4.1 Robust-DARTS Search Space

Robust-DARTS (Zela et al., 2020) introduced four simple search spaces which share the same macro architecture as DARTS while choosing different sets of candidate operations at the cell level. These search spaces, S1 - S4, were designed to showcase the failure modes of DARTS. Specifically, spaces S2, S3 and S4 demonstrate the tendency of DARTS to choose non-parametric operations, such as *Skip-connections* (in spaces S2 and S3) or even *Noise* operations (in space S4), over parametric operations. For a fair comparison, we run DARTS as well as LoRA-DARTS on spaces S2 through S4 on the same code and hardware setup. The results, summarized in Table 1, show that LoRA-DARTS is robust to the *Noise* operation, while being much less susceptible to picking skip connections compared to DARTS.

4.2 DARTS Search Space

The DARTS search space includes 8 candidate operations in each of the two cell types: *normal* and *reduction*. Out of these 8, 4 are non-parametric, including the *None* operation, which is excluded when discretizing the supernet to determine the best architecture. To estimate the performance

Space (operation)	S2 (*Skip connections*)		S3 (*Skip connections*)		S4 (*Noise*)	
	Normal Cell	Reduction Cell	Normal Cell	Reduction Cell	Normal Cell	Reduction Cell
DARTS	10.33 ± 1.24	5.66 ± 0.47	12.66 ± 0.94	7.33 ± 1.24	8.33 ± 1.70	0
LoRA-DARTS	6.66 ± 1.24	6.33 ± 0.471	9.66 ± 0.471	4.66 ± 1.69	0.66 ± 0.471	0

Table 1: Number of edges where non-parametric operations are the strongest in the supernet in Robust-DARTS spaces S2, S3 and S4. Mean and standard deviation of three runs of each experiment.

Optimizer	Weight Type	Test Accuracy (%)
DARTS	Weight Sharing	96.16 ± 0.26
DARTS	Weight Entanglement	96.09 ± 0.09
LoRA-DARTS	Weight Sharing	96.37 ± 0.06
LoRA-DARTS	Weight Entanglement	96.21 ± 0.13

Table 2: Results of retraining the models from scratch. Mean and standard deviation across three runs.

of DARTS and LoRA-DARTS in its search phase, we query the NAS-Bench-301 surrogate (Zela et al., 2022) predictor at the end of each training epoch. Doing so yields the predicted performance of the dominant architecture at every epoch of supernet training. The dominant architecture is obtained by selecting the candidate operations with the highest architectural weight on every edge of the cells in the supernet (excluding the *None* operation), and discarding all but the two strongest incoming edges of every node in them.

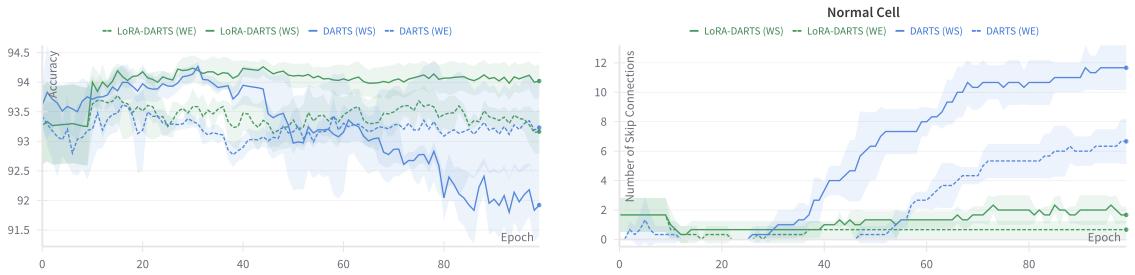
We run two sets of experiments. First, we run the original method as a baseline - the DARTS optimizer is run on the DARTS search space, with classifying the CIFAR-10 dataset as the primary task. Second, we run the LoRA-variant of the method by warm-starting it for 10 epochs and then activating the LoRA layers for the remaining 90 epochs. In both cases, we run separate experiments with the supernet using weight sharing (WS) and weight entanglement (WE). We chose to train the supernet for 100 epochs (instead of 50, as prescribed by the authors of DARTS) to demonstrate the stability of LoRA-DARTS when training longer. The results can be seen in Figure 1a. LoRA-DARTS beats the DARTS baselines in weight-sharing implementation of the DARTS search space, while being comparable in the weight entangled space. Note here that the weight entangled space does not suffer the failure mode of collapsing to architectures dominated by skip connections.

Indeed, these are surrogate predictions. To verify our results, we rely on two methods. First, we analyze the trajectory of the operations in the supernet during its training. Specifically, we count the number of times skip connections are the strongest operation on the edges of the cells, since DARTS is known for collapsing to architectures which are dominated by them. Then, we train from scratch models of the architectures obtained after 50 epochs of training the supernet (since that is how many epochs the supernet is trained in DARTS) and compare their performance. The results, summarized in Figure 1b and Table 2, show that the architectures discovered by the LoRA-DARTS (1) have fewer skip connections and (2) exhibit superior performance when trained from scratch.

We posit that the improved performance stems from the supernet no longer overfitting the training set. This can be seen in the training loss curves of the supernet in Figure 2a. In all cases, the LoRA variants have higher training loss than their baselines. Interestingly, the validation loss of the LoRA methods does not suffer as large a gap compared to the baselines - see Figure 2b.

5 Conclusion

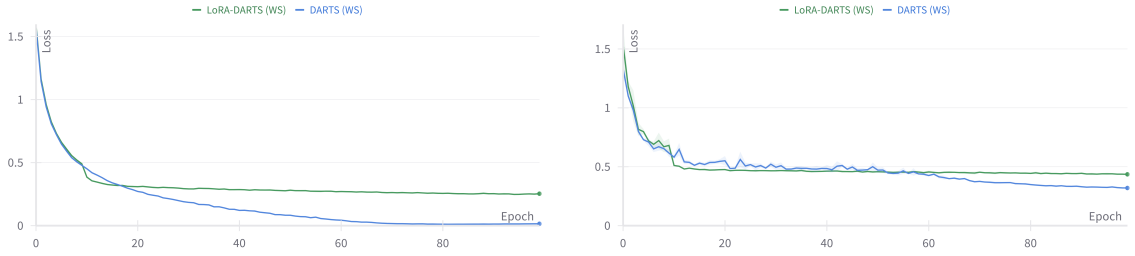
We introduce LoRA-DARTS, a method that avoids the common failure mode in DARTS where the discovered architectures are dominated by skip connections. We find that despite running the DARTS search for twice as long as recommended by Liu et al. (2019), LoRA-DARTS does not



(a) Predicted test accuracy

(b) Number of skip connections

Figure 1: (a) Quality (predicted accuracy) of the intermittent optimal architectures during training with LoRA-DARTS compared to DARTS. (b) The reason is the much lower susceptibility to the failure mode of preferring non-parameteric skip connections. Both plots show mean and standard deviation of three runs.



(a) Train Loss

(b) Validation Loss

Figure 2: Loss curves of supernet training (weight sharing)

collapse to this failure mode. In terms of limitations, while the memory overhead associated with adding LoRA weights to the supernet is negligible, we find that the training time of the supernet increases by a factor of 1.35 for WS and 1.17 for WE implementations, respectively. Future works could explore pruning techniques to speed up supernet training.

6 Broader Impact Statement

Neural network training, particularly one-shot model training, is computationally expensive and energy-hungry. LoRA-DARTS adds to these requirements by incorporating additional LoRA modules during training, which take (slightly) longer to train. However, these are one-time costs that may amortize over the lifetime of a model because NAS enables finding models that make a better trade-off between accuracy and efficiency, depending on the deployment scenario.

The ultimate goal of NAS research is to fully automate the design of neural network architectures, which is a challenging task due to the vast (even unbounded) search space. By automating this complex design process, NAS reduces the required technical know-how and engineering efforts that would be required for architecture optimization, and making deep learning more accessible to a broad range of users (democratization of AI). This increases the already vast impact of deep learning on society - both good, such as advancements in medical imaging, and bad, such as adversarial attacks. Nevertheless, full automation in NAS has not yet been achieved. For instance, as discussed earlier, differentiable one-shot NAS techniques tend to be brittle and prone to failure modes. Our LoRA-DARTS improves on this second aspect, as it provides a method to mitigate these failure modes, and thereby provides one step further towards democratization of ML.

Acknowledgements. We acknowledge funding by the European Union (via ERC Consolidator Grant DeepLearning 2.0, grant no. 101045765). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them. Robert Bosch GmbH is acknowledged for financial support. This research was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under grant number 417962828. This research was partially supported by TAILOR, a project funded by EU Horizon 2020 research and innovation programme under GA No 952215. Frank Hutter acknowledges the financial support of the Hector Foundation.

References

(2017). *Proceedings of the International Conference on Learning Representations (ICLR'17)*. Published online: `iclr.cc`.

Baker, B., Gupta, O., Naik, N., and Raskar, R. (2017). Designing neural network architectures using reinforcement learning. In icl (2017). Published online: `iclr.cc`.

Cai, H., Gan, C., Wang, T., Zhang, Z., and Han, S. (2020). Once-for-All: Train one network and specialize it for efficient deployment. In *Proceedings of the International Conference on Learning Representations (ICLR'20)*.

Chen, M., Peng, H., Fu, J., and Ling, H. (2021a). Autoformer: Searching transformers for visual recognition. In *Proceedings of the 24nd IEEE/CVF International Conference on Computer Vision (ICCV'21)*, pages 12270–12280. cvfandieee.

Chen, X., Wang, R., Cheng, M., Tang, X., and Hsieh, C.-J. (2021b). Drnas: Dirichlet Neural Architecture Search. In *Proceedings of the International Conference on Learning Representations (ICLR'21)*.

Chu, X., Zhou, T., Zhang, B., and Li, J. (2020). Fair darts: Eliminating unfair advantages in differentiable architecture search. In Vedaldi, A., Bischof, H., Brox, T., and Frahm, J., editors, *16th European Conference on Computer Vision (ECCV'20)*, pages 465–480. Springer.

DeVries, T. and Taylor, G. (2017). Improved regularization of convolutional neural networks with cutout. *arXiv:1708.04552 [cs.CV]*.

Dong, X. and Yang, Y. (2019). Searching for a robust neural architecture in four gpu hours. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition (CVPR'19)*.

Fujino, S., Mori, N., and Matsumoto, K. (2017). Deep convolutional networks for human sketches by means of the evolutionary deep learning. In *2017 Joint 17th World Congress of International Fuzzy Systems Association and 9th International Conference on Soft Computing and Intelligent Systems (IFSA-SCIS)*, pages 1–5. IEEE.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. (2021). Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Jiang, S., Ji, Z., Zhu, G., Yuan, C., and Huang, Y. (2024). Operation-level early stopping for robustifying differentiable nas. *Advances in Neural Information Processing Systems*, 36.

Kandasamy, K., Neiswanger, W., Schneider, J., Poczos, B., and Xing, E. (2018). Neural Architecture Search with bayesian optimisation and optimal transport. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Proceedings of the 31st International Conference on Advances in Neural Information Processing Systems (NeurIPS'18)*.

Liang, H., Zhang, S., Sun, J., He, X., Huang, W., Zhuang, K., and Li, Z. (2019). Darts+: Improved differentiable architecture search with early stopping. *arXiv:1909.06035 [cs.CV]*.

Liu, C., Zoph, B., Neumann, M., Shlens, J., Hua, W., Li, L., Fei-Fei, L., Yuille, A., Huang, J., and Murphy, K. (2018). Progressive neural architecture search. In Ferrari, V., Herbert, M., Sminchisescu, C., and Weiss, Y., editors, *14th European Conference on Computer Vision (ECCV'18)*, pages 19–35.

Liu, H., Simonyan, K., and Yang, Y. (2019). DARTS: Differentiable architecture search. In *Proceedings of the International Conference on Learning Representations (ICLR'19)*.

Loshchilov, I. and Hutter, F. (2017). SGDR: Stochastic gradient descent with warm restarts. In icl (2017). Published online: `iclr.cc`.

Pham, H., Guan, M., Zoph, B., Le, Q., and Dean, J. (2018). Efficient Neural Architecture Search via parameter sharing. In *Proceedings of the 35th International Conference on Machine Learning (ICML'18)*.

Real, E., Aggarwal, A., Huang, Y., and Le, Q. (2019). Regularized Evolution for Image Classifier Architecture Search. In Hentenryck, P. V. and Zhou, Z., editors, *Proceedings of the Thirty-Third Conference on Artificial Intelligence (AAAI'19)*, volume 33, pages 4780–4789. AAAI Press.

Sukthanker, R. S., Krishnakumar, A., Safari, M., and Hutter, F. (2023). Weight-entanglement meets gradient-based neural architecture search. *arXiv preprint arXiv:2312.10440*.

Swersky, K., Duvenaud, D., Snoek, J., Hutter, F., and Osborne, M. (2013). Raiders of the lost architecture: Kernels for Bayesian optimization in conditional parameter spaces. In Hoffman, M., Snoek, J., de Freitas, N., and Osborne, M., editors, *NeurIPS Workshop on Bayesian Optimization in Theory and Practice (BayesOpt'13)*.

Zela, A., Elsken, T., Saikia, T., Marrakchi, Y., Brox, T., and Hutter, F. (2020). Understanding and robustifying differentiable architecture search. In *Proceedings of the International Conference on Learning Representations (ICLR'20)*.

Zela, A., Siems, J., Zimmer, L., Lukasik, J., Keuper, M., and Hutter, F. (2022). Surrogate NAS benchmarks: Going beyond the limited search spaces of tabular NAS benchmarks. In *Proceedings of the International Conference on Learning Representations (ICLR'22)*. Published online: `iclr.cc`.

Zoph, B. and Le, Q. V. (2017). Neural Architecture Search with reinforcement learning. In icl (2017). Published online: `iclr.cc`.

Zoph, B., Vasudevan, V., Shlens, J., and Le, Q. (2018). Learning transferable architectures for scalable image recognition. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition (CVPR'18)*.

Submission Checklist

1. For all authors...

   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? Yes

   (b) Did you describe the limitations of your work? Yes

   (c) Did you discuss any potential negative societal impacts of your work? Yes

   (d) Did you read the ethics review guidelines and ensure that your paper conforms to them? `https://2022.automl.cc/ethics-accessibility/` Yes

2. If you ran experiments...

   (a) Did you use the same evaluation protocol for all methods being compared (e.g., same benchmarks, data (sub)sets, available resources)? Yes

   (b) Did you specify all the necessary details of your evaluation (e.g., data splits, pre-processing, search spaces, hyperparameter tuning)? Yes

   (c) Did you repeat your experiments (e.g., across multiple random seeds or splits) to account for the impact of randomness in your methods or data? Yes

   (d) Did you report the uncertainty of your results (e.g., the variance across random seeds or splits)? Yes

   (e) Did you report the statistical significance of your results? No

   (f) Did you use tabular or surrogate benchmarks for in-depth evaluations? Yes

   (g) Did you compare performance over time and describe how you selected the maximum duration? Yes

   (h) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? Yes

   (i) Did you run ablation studies to assess the impact of different components of your approach? Yes. The LoRA parameters are the only new component in the supernet. We explore the impact they have on weight sharing as well as weight entanglement implementations of DARTS.

3. With respect to the code used to obtain your results...

   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results, including all requirements (e.g., `requirements.txt` with explicit versions), random seeds, an instructive `README` with installation, and execution commands (either in the supplemental material or as a URL)? Yes

   (b) Did you include a minimal example to replicate results on a small subset of the experiments or on toy data? Yes

   (c) Did you ensure sufficient code quality and documentation so that someone else can execute and understand your code? Yes

   (d) Did you include the raw results of running your experiments with the given code, data, and instructions? Yes

(e) Did you include the code, additional data, and instructions needed to generate the figures and tables in your paper based on the raw results? Yes. The plots are from Weights and Biases. If the code is run with WandB logging enabled, the user gets these plots for free.

4. If you used existing assets (e.g., code, data, models)…

   (a) Did you cite the creators of used assets? Yes.

   (b) Did you discuss whether and how consent was obtained from people whose data you're using/curating if the license requires it? Not Applicable

   (c) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? Not Applicable

5. If you created/released new assets (e.g., code, data, models)…

   (a) Did you mention the license of the new assets (e.g., as part of your code submission)? Yes.

   (b) Did you include the new assets either in the supplemental material or as a URL (to, e.g., GitHub or Hugging Face)? Yes

6. If you used crowdsourcing or conducted research with human subjects…

   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? Not Applicable

   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? Not Applicable

   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? Not Applicable

7. If you included theoretical results…

   (a) Did you state the full set of assumptions of all theoretical results? Not Applicable

   (b) Did you include complete proofs of all theoretical results? Not Applicable

A Appendix

A.1 Experiment Details

A.1.1 Architecture Search. We use the same set-up as Liu et al. (2019) for training the supernet, with one exception - we train it for 100 epochs instead of 50. We use a batch size of 64 for both training and validation sets. We use a 50%-50% split of the CIFAR-10 train data to get the training and validation splits. The supernet consists of 8 cells stacked on top of each other, two of which are *reduction cells*, which increase number of channels of the feature maps. The initial number of channels is 16. The weights of the supernet are optimized using SGD with momentum of 0.9 and weight decay of 3×10^{-4}. The learning rate begins at 0.025 and is annealed to 0 using cosine annealing (Loshchilov and Hutter, 2017) without restarts.

A.1.2 Architecture Selection for Evaluation. To select the architecture to train from scratch, we run the DARTS search, i.e., supernet training, three times. The genotype of the search with the lowest validation error at the 50th epoch is chosen for retraining.

A.1.3 Architecture Evaluation. The selected architecture is trained for 600 epochs with a batch size of 96. 20 cells are now stacked to form the model, two of which are reduction cells. The number of initial channels is increased from 16 to 36, and additional data augmentations such as cutout (DeVries and Taylor, 2017), path dropout with a probability of 0.2, and auxiliary towers with weight 0.4 are employed.

A.1.4 Hardware Details. All experiments were run on Nvidia GeForce RTX-2080 Ti GPUs. Each job consumed 8 cores of a AMD EPYC 7502 32-Core processor. Retraining the models from scratch to evaluate them used the same GPUs but 2 cores of Intel(R) Xeon(R) Silver 4114 Processor per job. The time taken per experiment is given in Table 3.

A.1.5 Code. The code for running our experiments can be found at `https://github.com/automl/LoRA-DARTS`.

Method	Weight Type	Time (GPU hours)
DARTS	Weight sharing	14.4
DARTS	Weight entanglement	12.9
LoRA-DARTS	Weight sharing	19
LoRA-DARTS	Weight entanglement	15.1

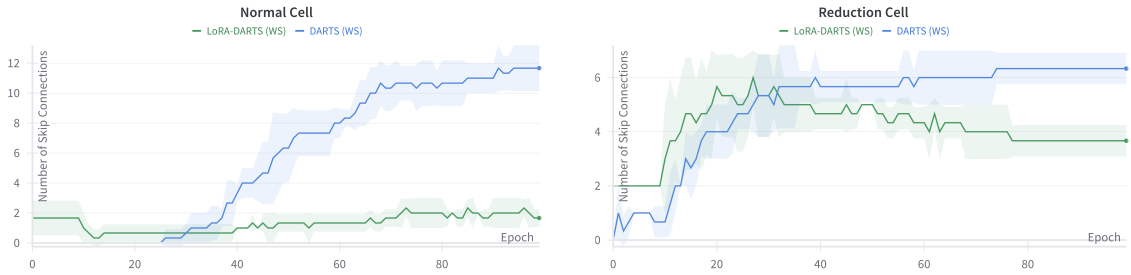
Table 3: Time taken by 1 run of different experiments

A.2 Skip Connections

For both weight sharing (WS) and weight entanglement (WE) implementations of DARTS, we find that LoRA-DARTS chooses fewer skip connections on average as the search progresses. In the case of weight sharing, the general trend that is observed in DARTS is that the number of skip connections *increases* over the epochs. However, this is not the case with LoRA-DARTS, as seen in Figure 3a and Figure 3b. Interestingly, in the latter case, the number of skip connections peaks at around 28 epochs and then declines.

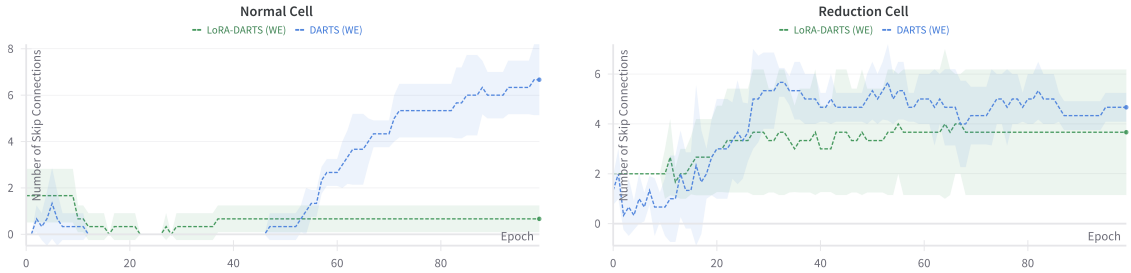
A.3 Loss Curves

As seen in Figure 5, for weight entanglement, there is a significant gap between the train and validation losses of DARTS, but this gap is less pronounced in LoRA-DARTS.



(a) Normal Cell (b) Reduction cell

Figure 3: Number of skip connections in DARTS and LoRA-DARTS (Weight Sharing). Mean and standard deviation of three runs.



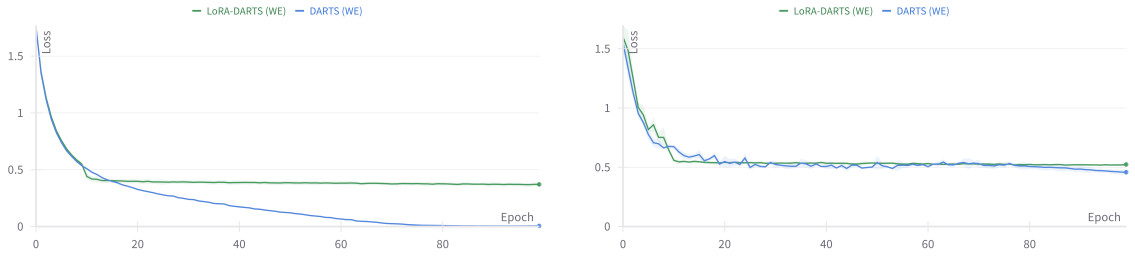
(a) Normal Cell (b) Reduction cell

Figure 4: Number of skip connections in DARTS and LoRA-DARTS (Weight Entanglement). Mean and standard deviation of three runs.

A.4 Choice of Warm Epochs

In our work, we opted to warm-start the supernet with 10 epochs of training. To evaluate the impact of varying the number of epochs, we also conducted experiments using 5 and 20 epochs. The results are presented in Figure 6. Warm-starting with fewer epochs—5 and 10—yields comparable performance. However, it is evident that using 20 epochs for the warm-start leads to a significantly poorer performance trajectory. We posit that this is because the supernet begins to overfit the training data at later epochs, which is prevented when LoRA layers are activated in earlier epochs. See also Figure 7 for the train and validation losses of the supernet, which support this proposition.

A.5 Architectures Discovered



(a) Train Loss (b) Validation Loss

Figure 5: Loss Curves of supernet training (WE). Mean and standard deviation of three runs.

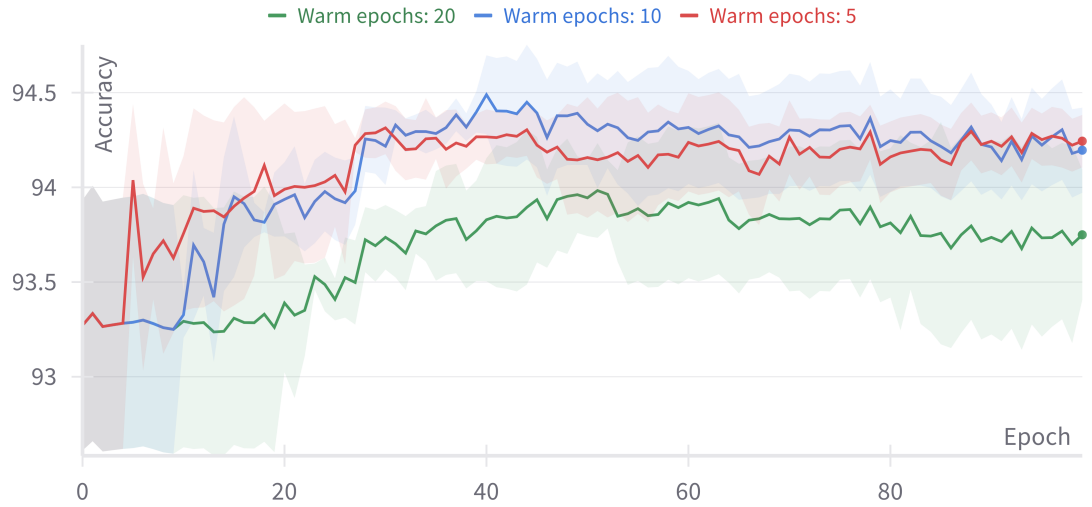
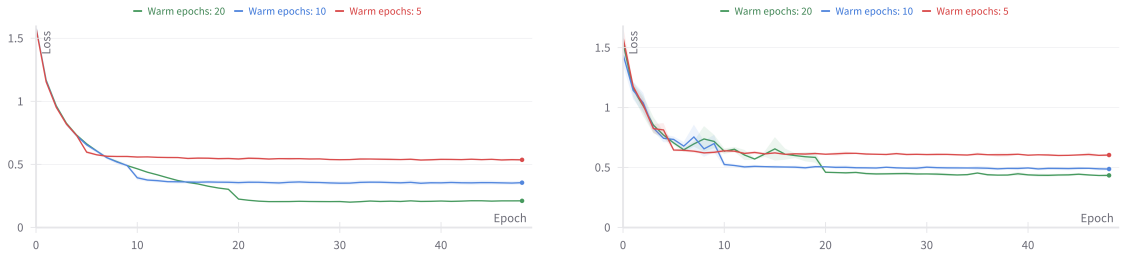
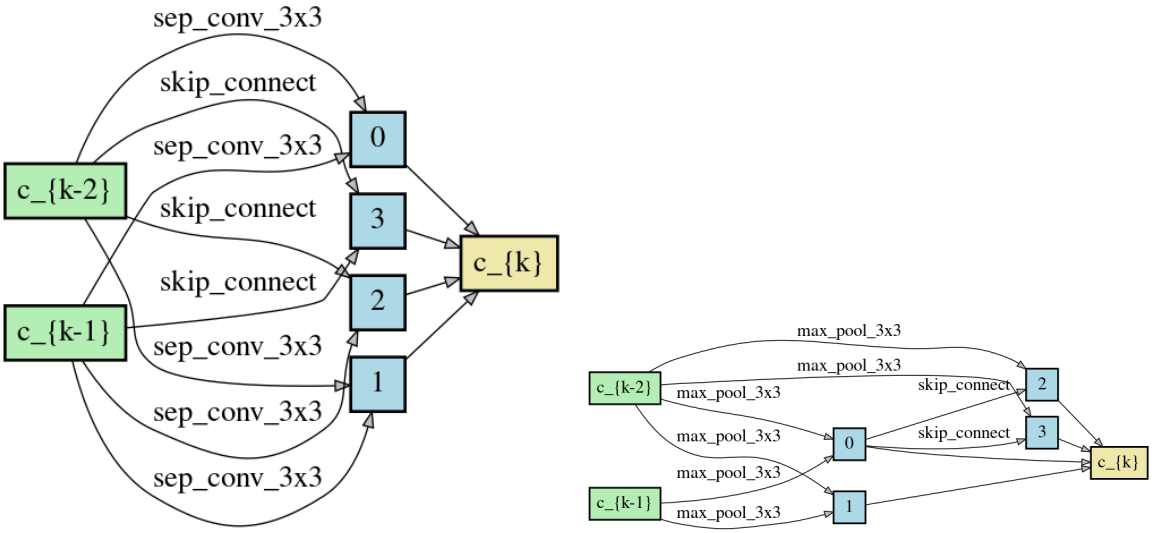


Figure 6: Predicted accuracy of the intermittent optimal architectures during training with LoRA-DARTS with 5, 10 and 20 epochs of warm-starting.



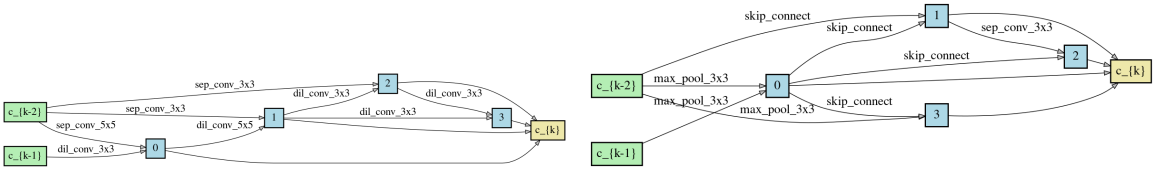
(a) Train Loss (b) Validation Loss

Figure 7: Train and validation loss of the supernet during training with LoRA-DARTS with 5, 10 and 20 epochs of warm-starting.



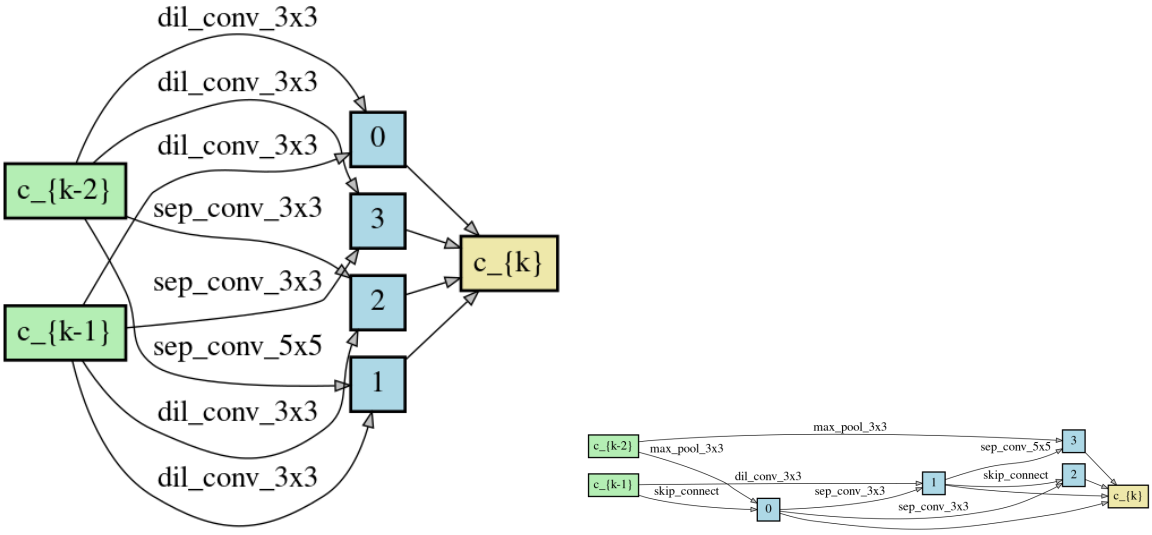
(a) Normal cell (b) Reduction Cell

Figure 8: Architecture found by DARTS on weight shared space



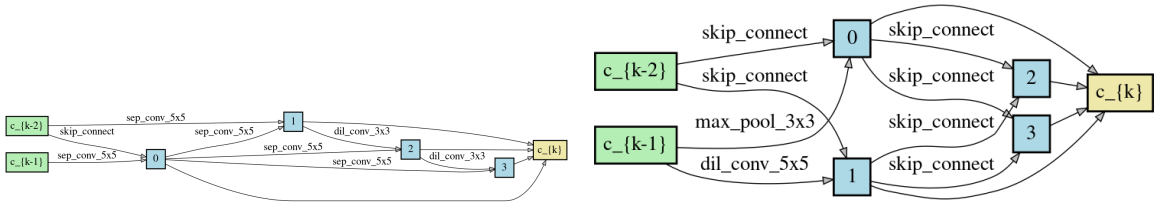
(a) Normal cell (b) Reduction Cell

Figure 9: Architecture found by DARTS on weight entangled space



(a) Normal cell (b) Reduction Cell

Figure 10: Architecture found by LoRA-DARTS on weight shared space



(a) Normal cell (b) Reduction Cell

Figure 11: Architecture found by LoRA-DARTS on weight entangled space