Regularized Meta-Learning for Neural Architecture Search

Rob van Gastel[1] **Joaquin Vanschoren**[2]

[2]Eindhoven University of Technology

Abstract Neural architecture search (NAS) methods have successfully enabled the automated search of neural architectures in various domains. However, most techniques start from scratch with every new task. Techniques have been proposed that generalize across tasks, but don't always adapt well to new tasks. In this work, we consider meta-learning approaches that effectively leverage prior experience to adapt to unseen tasks. We analyze different regularization and training methods to improve the generalizability of meta-learning for NAS. Empirical results on standard few-shot classification benchmarks show that the added regularization and adjustments in the network optimization improve upon previous approaches, such as MetaNAS.[1]

1 Introduction

Neural networks have achieved significant results in a variety of domains, such as image classification (He et al., 2016), machine translation (Liu et al., 2020), and playing games (Silver et al., 2017; Badia et al., 2020). This success is contributed to the careful design of the network architecture and chosen hyperparameters. Network architectures are usually designed by human experts, which is a cumbersome process and prone to delivering underperforming architectures. The goal of neural architecture search (NAS) is to accelerate the designing process of these neural architectures by employing a systematic search strategy to find well-performing neural architectures. Recent developments in NAS are closing the gap between human expert-designed architectures and automated solutions.

An important obstacle for the success of NAS is the large number of model evaluations required to search a vast design space, demanding substantial computational resources. This makes it hard to apply NAS in scenarios where only limited amounts of compute are available. In such scenarios, one has to leverage experience from related tasks to adapt to new tasks efficiently. This type of learning is known as meta-learning. This research aims to improve the generalizability of meta-learning for NAS through regularization, specifically in the setting of few-shot learning.

We build upon MetaNAS (Elsken et al., 2019) (see section 2.3), which marries gradient-based meta-learning with gradient-based NAS. This method aims to learn a meta-architecture (a weighted subspace of architectures) that quickly adapts to a task-specific architecture. This type of search has reduced the architecture search time by several orders of magnitude (Liu et al., 2018). However, these methods also tend to suffer from instability issues, which is referred to as the optimization gap by Xie et al. (2020). The contributions of our work are twofold:

- We evaluate different regularization techniques to regularize meta-learning for NAS, to reduce the problems associated with the optimization gap.
- We propose the use of a different DARTS optimization technique, TSE-DARTS (Fil et al., 2021). This method improves the optimization of DARTS in settings in which a validation set per task is not available, such as few-shot learning.

[1]The code for reproducibility is available at: https://github.com/RobvanGastel/meta-fsl-nas

2 Related Work

We first describe background and related work on one-shot models, and meta-learning for NAS.

2.1 One-Shot Models

Weight-sharing models are proposed to amortize the cost of training many candidate architectures. These models view sampled architectures as sub-graphs of a larger supergraph, the *super-network* (Pham et al., 2018). The training procedure optimizes all the sub-networks at once, such that sampled architectures do not require training from scratch. One-shot models extend the idea of weight-sharing models by combining the learning of the architecture with learning the network weights (Bender et al., 2018). Differentiable architecture search (DARTS) (Liu et al., 2018) is a popular one-shot model approach that makes the search over architectures differentiable by introducing a continuous relaxation of each edge of the super-network and optimizing the network using bilevel optimization.

2.2 The Optimization Gap

These weight-sharing super-networks suffer from search instability issues, specifically sensitivity to random initialization and hyperparameters. This optimization gap (Xie et al., 2020) results in the same algorithm producing considerably different architectures and test accuracy results when executing multiple runs (Li and Talwalkar, 2019; Zela et al., 2019). Consequently, there is no guarantee on the search procedure producing sampled architectures of high quality. In section 3 we introduce regularization methods to mitigate this gap.

2.3 Neural Architecture Search for Few-Shot Learning

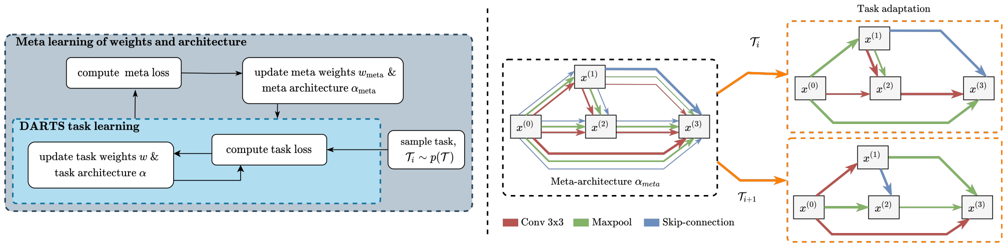
Meta learning of weights and architecture
compute meta loss
update meta weights w_{meta} & meta architecture α_{meta}
DARTS task learning
update task weights w & task architecture α
compute task loss
sample task, $\mathcal{T}_i \sim p(\mathcal{T})$
Meta-architecture α_{meta}
$x^{(0)}$
$x^{(1)}$
$x^{(2)}$
$x^{(3)}$
$\mathcal{T}_i$
$\mathcal{T}_{i+1}$
Task adaptation
🟥 Conv 3x3
🟩 Maxpool
🟦 Skip-connection

Figure 1: The meta-weights and meta-architecture are jointly optimized over a distribution of tasks using a meta-learner. The task-learner adapts the meta-weights and meta-architecture to a task-specific architecture. Illustration inspired by Elsken et al. (2019).

Many prior works explore meta-learning in a few-shot learning setting with a fixed architecture, such as Model-Agnostic Meta-Learning (MAML) (Finn et al., 2017). However, optimizing both the neural architecture and the weights might allow these methods to perform better on few-shot learning tasks from a wider distribution. AutoMeta (Kim et al., 2018) and Continual Architecture Search (Pasunuru and Bansal, 2019) learn an architecture that performs well on average, but do not adapt the architecture to specific tasks, only the model weights.

In contrast, MetaNAS also adapts the architecture to a specific task. The task-learner (DARTS) jointly adapts the weights w_{meta} and architecture parameters α_{meta} to a new tasks $\mathcal{T}_i$ during the inner-loop. The "meta-architecture" is optimized during the outer-loop. The MetaNAS process is visualized in Figure 1. We define a task $\mathcal{T}_i$ part of a distribution $p(\mathcal{T})$ to contain a training set $\mathcal{T}^{train} \sim p(\mathcal{T})$ and test set $\mathcal{T}^{test} \sim p(\mathcal{T})$, in which a task is composed of $\mathcal{T}_i = (\mathcal{D}^{\mathcal{T}_i}_{train}, \mathcal{D}^{\mathcal{T}_i}_{test})$. The outer-loop objective, is taken from Reptile (Nichol et al., 2018),

$$\mathcal{L}_{\text{meta}}\left(w, \alpha, p, \Phi^k\right) = \sum_{\mathcal{T}_i \sim \mathcal{T}^{\text{train}}} \mathcal{L}_{\mathcal{T}_i}\left(\Phi^k\left(w, \alpha, D^{\mathcal{T}_i}_{\text{train}}\right), D^{\mathcal{T}_i}_{\text{test}}\right)$$

where Φ is the task-learner and k are the gradient steps of DARTS. The task-learner updates both w and α with potentially different learning rates, the weight learning rate λ_{task} and architecture learning rate ξ_{task}. The optimization steps of the task-learner are as follows (Elsken et al., 2019),

$$\begin{aligned}\begin{pmatrix} w^{j+1} \\ \alpha^{j+1} \end{pmatrix} &= \Phi\left(w^j, \alpha^j, D_{\text{train}}^{\mathcal{T}_i}\right) \\ &= \begin{pmatrix} w^j - \lambda_{\text{task}}\, \nabla_w \mathcal{L}_{\mathcal{T}}\left(w^j, \alpha^j, D_{\text{train}}^{\mathcal{T}_i}\right) \\ \alpha^j - \xi_{\text{task}}\, \nabla_\alpha \mathcal{L}_{\mathcal{T}}\left(w^j, \alpha^j, D_{\text{train}}^{\mathcal{T}_i}\right) \end{pmatrix}\end{aligned}$$

In contrast to DARTS, the task-learner optimizes both α and w on the training set, due to the unavailability of a validation set in few-shot learning settings.

3 Methodology

In this work, we extend MetaNAS with a range of regularization techniques, detailed below, as well as an optimization technique to improve the task-learner.

3.1 Regularization of the Task-Learner

We introduce seven regularization methods to the DARTS task-learner in attempts to shrink the optimization gap. Originally, MetaNAS already uses three methods: *pruning of mixture over operations and over input nodes*, and *thresholding alphas*. The pruning of mixture methods aim to obtain progressively sparser architecture parameters during training to prevent from retraining after finding an architecture (Elsken et al., 2019). Thresholding prunes alphas from the final model below a set threshold to obtain a sparser network.

3.1.1 Search Space Approximation.

Search space approximation, first introduced in P-DARTS (Chen et al., 2019), divides learning into three progressive stages allowing deeper networks with every stage. At the end of every stage, it increases the size of the feature maps, increases the number of normal cells, and decreases the number of operations considered per edge.

3.1.2 Search Space Regularization.

Search space regularization, also introduced by P-DARTS, adjusts the skip-connections. Due to DARTS's preference for skip-connections in early stages of training (Zela et al., 2019). Operation-level Dropout (Srivastava et al., 2014) is applied to every skip-connection. Additionally, skip-connections are limited to two skip-connections per normal cell.

3.1.3 Power Cosine Annealing.

Power Cosine Annealing, as introduced in SharpDARTS (Hundt et al., 2019), adds an additional parameter p to the cosine learning rate annealing scheduler (Loshchilov and Hutter, 2016). The power curve parameter p allows a more optimal learning rate throughout training and observes better results in fewer epochs (Hundt et al., 2019).

3.1.4 SharpDARTS Search Space.

The SharpDARTS search space adjusts convolutions in the DARTS search space to depth-wise separable convolutions (SepConv) based on the MobileNetV2 architecture (Sandler et al., 2018).

3.1.5 Max-W Alpha Regularization.

Max-W alpha Regularization, also introduced by SharpDARTS, adjusts the weighing of architecture parameters during the learning of the weights. The intuition is to keep the largest operation's architecture parameter unchanged and allow for the other architecture parameters to grow, according to the authors of Hundt et al. (2019).

3.1.6 DARTS-.

The DARTS- technique (Chu et al., 2021) attempts to stabilize the architecture search by introducing an auxiliary skip-connection on every edge. The auxiliary operation is introduced with a parameter β, which decays to 0 during training. By adding this operation, the unfair competition from non-parametric operations is reduced.

3.1.7 PC-DARTS-.

PC-DARTS- is introduced by DARTS- as an extension on PC-DARTS (Xu et al., 2019). This method samples a small portion of input channels for every operation to reduce computation and regularize the training using only a subset of input channels during the architecture search.

3.2 Task-learner Optimization Without Validation Set

The original formulation of DARTS assumes the existence of a validation set during training, while in few-shot learning there is no validation set. Therefore, following Fil et al. (2021), we apply a generalization estimator, Training-Speed-Estimator (TSE), to optimize without a validation set. Let $\mathcal{L}$ be the loss function, f_{θ_t} the neural network with parameters θ_t after t iteration, and T the total number of unrolling steps. At every time step t a training data batch is sampled, $\mathcal{D}_t = \{(x_{1(t)}, y_{1(t)}), \ldots, (x_{n(t)}, y_{n(t)})\}$ to update the network. TSE is defined as follows, TSE $= \sum_{t=1}^{T} \mathcal{L}_{train}\left(f_{\theta_t}\left(x_{(t)}\right), y_{(t)}\right)$. The gradient of the estimator is used to update the architecture, $\nabla_\alpha TSE$. For the complete derivation of the TSE gradient, we refer to the original paper (Fil et al., 2021).

4 Experiments

We evaluate the proposed extensions on image classification few-shot learning benchmarks Omniglot (Lake et al., 2015), TripleMNIST (Sun, 2019) and MiniImageNet (Ravi and Larochelle, 2017). We consider n-way, k-shot few-shot learning tasks; each task has n classes with k examples per class. First, we evaluate different regularization methods on MetaNAS. Second, we evaluate optimization without a validation set, TSE-DARTS. For a discussion on the used hyperparameters and experiment setup we refer to Appendix A.1.

4.1 Experiment 1: Evaluation of the Regularization Methods

We compare the seven regularization methods on the TripleMNIST and Omnglot few-shot learning benchmarks. First, to pre-select the best regularization methods, we select the methods outperforming the baseline MetaNAS implementation on 1-shot, 20-way Omniglot (see Appendix A.2).

The three methods improving baseline MetaNAS are *limit and dropout skip-connections*, Max-W regularization, and power cosine annealing. The limit an dropout on the skip-connections applies Search Space Regularization (section 3.1.2) without the progressive staging of P-DARTS, as we observe empirically better results without staging.

Table 1 displays the performance of these methods on Omniglot and TripleMNIST. We already improve upon the performance of MetaNAS by only applying limit and dropout on the skip-connections in small meta-epoch settings. Additionally, adding max-W regularization and power cosine annealing maintains similar performance while greatly reducing the number of parameters. This reduction might indicate that DARTS finds more task-specific architectures and converges faster.

	Omniglot				TripleMNIST	
Method	parameters	accuracy in 1-shot, 20-way	parameters	accuracy in 3-shot, 20-way	parameters	accuracy in 3-shot, 20-way
MetaNAS	340k	78.20 ±1.18	260k	88.70 ±0.86	389k	90.94 ±0.34
MetaNAS+Limit and Dropout Skip-Connections	335k	80.86 ±1.42	360k	91.55 ±0.81	405k	93.72 ±0.56
MetaNAS+Limit and Dropout Skip-Connections + max-W + Power Cos Annealing	149k	81.62 ±0.22	147k	89.94 ±0.34	195k	92.38 ±0.66

Table 1: The meta-testing accuracy and parameters of the regularization methods compared to the baseline on the Omniglot and TripleMNIST datasets. The accuracy is averaged over two runs with a ±1 standard deviation after 500 meta-epochs with 250 warm-up epochs.

4.2 Experiment 2: Training Without Validation Set

We evaluate TSE-DARTS to improve the optimization of the super-network without the availability of a validation set in few-shot learning. Initially, the architecture and weights are optimized using the training set. However, this can lead to overfitting and poor generalization (Liu et al., 2018). The TSE estimator adjusts the optimization to better fit cases without a validation set and introduces a new parameter. We evaluate TSE-DARTS with a higher learning rate and two unrolling steps. The hyperparameter selection is discussed in Appendix A.3.

Table 2 displays the results, which outperform the baseline by 4.18%. Notably, the TSE-DARTS optimization combined with the previously preselected regularization techniques does not improve accuracy. Improving the performance of TSE-DARTS with regularization might require more careful tuning of the unrolling steps and regularization parameters, such as Dropout.

	Omniglot		MiniImageNet	
Method	parameters	accuracy in 1-shot, 20-way	parameters	accuracy in 5-shot, 5-way
MetaNAS	340k	78.20 ±1.18	155k	42.55 ±2.65
MetaNAS+TSE-DARTS	251k	82.38 ±0.90	415k	49.85 ±3.65
MetaNAS+TSE-DARTS+Limit and Dropout Skip-Connections + max-W + Power Cos Annealing	233k	81.64 ±0.60	100k	37.05 ±3.45

Table 2: The meta-testing accuracy and parameters of the optimization method compared to the baseline on the Omniglot and MiniImageNet datasets. The accuracy is averaged over two runs with a ±1 standard deviation after 500 meta-epochs with 250 warm-up epochs.

5 Conclusions and Future Work

In this work, we presented methods to regularize the meta-learning for NAS. First, we identify and evaluate seven methods that we combine with the MetaNAS framework to improve generalization. The limiting and dropout on skip-connections and max-W regularization with power cosine annealing improve the performance of MetaNAS on the 1-shot, 20-way, 3-shot, 20-way Omniglot, and 3-shot, 20-way TripleMNIST settings. Second, we introduce TSE-DARTS as an adjustment to DARTS optimization. TSE-DARTS fits the problem formulation of few-shot learning in which a validation set is not available. Through applying this optimization we observe the largest improvement compared to MetaNAS improving by 4.18% on the 1-shot, 20-way Omniglot setting.

In future work, we would like to further improve the generalization abilities of MetaNAS. One of the limitations of MAML is that it optimizes the overall performance and not necessarily the task-specific performance (Deleu and Bengio, 2018). By applying meta-reinforcement learning instead of MAML we might find more task-specific architectures. Moreover, we hypothesize that meta-reinforcement learning might deal better with out-of-distribution tasks and in settings with more shots.

6 Limitations and Broader Impact Statement

In a broader context, the goal of NAS research is to have a positive societal impact by making NAS more accessible to the general public. This work introduces regularization methods for MetaNAS to reduce the computation cost required to find good models, mitigating the carbon footprint and energy consumption.

We empirically demonstrate the effectiveness of regularization, however it would be interesting to further study better meta-learning methods leading to better task-specific architectures through applying more adaptive meta-learners.

7 Reproducibility Checklist

Acknowledgements

This research was partially supported by TAILOR, a project funded by EU Horizon 2020 research and innovation programme under GA No 952215.

References

Badia, A. P., Piot, B., Kapturowski, S., Sprechmann, P., Vitvitskyi, A., Guo, Z. D., and Blundell, C. (2020). Agent57: Outperforming the atari human benchmark. 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 507–517. PMLR.

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

Chen, X., Xie, L., Wu, J., and Tian, Q. (2019). Progressive DARTS: bridging the optimization gap for NAS in the wild. *CoRR*, abs/1912.10952.

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

Deleu, T. and Bengio, Y. (2018). The effects of negative adaptation in model-agnostic meta-learning. *CoRR*, abs/1812.02159.

Elsken, T., Staffler, B., Metzen, J. H., and Hutter, F. (2019). Meta-learning of neural architectures for few-shot learning. *CoRR*, abs/1911.11090.

Fil, M., Ru, B., Lyle, C., and Gal, Y. (2021). DARTS without a validation set: Optimizing the marginal likelihood. *CoRR*, abs/2112.13023.

Finn, C., Abbeel, P., and Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. *CoRR*, abs/1703.03400.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Hundt, A., Jain, V., and Hager, G. D. (2019). sharpdarts: Faster and more accurate differentiable architecture search. *CoRR*, abs/1903.09900.

Kim, J., Choi, Y., Cha, M., Lee, J. K., Lee, S., Kim, S., Choi, Y., and Kim, J. (2018). Auto-meta: Automated gradient based meta learner search. *CoRR*, abs/1806.06927.

Lake, B. M., Salakhutdinov, R., and Tenenbaum, J. B. (2015). Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338.

Li, L. and Talwalkar, A. (2019). Random search and reproducibility for neural architecture search. *CoRR*, abs/1902.07638.

Liu, H., Simonyan, K., and Yang, Y. (2018). DARTS: differentiable architecture search. *CoRR*, abs/1806.09055.

Liu, Y., Gu, J., Goyal, N., Li, X., Edunov, S., Ghazvininejad, M., Lewis, M., and Zettlemoyer, L. (2020). Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742.

Loshchilov, I. and Hutter, F. (2016). SGDR: stochastic gradient descent with restarts. *CoRR*, abs/1608.03983.

Nichol, A., Achiam, J., and Schulman, J. (2018). On first-order meta-learning algorithms. *CoRR*, abs/1803.02999.

Pasunuru, R. and Bansal, M. (2019). Continual and multi-task architecture search. *CoRR*, abs/1906.05226.

Pham, H., Guan, M., Zoph, B., Le, Q., and Dean, J. (2018). Efficient neural architecture search via parameters sharing. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4095–4104. PMLR.

Ravi, S. and Larochelle, H. (2017). Optimization as a model for few-shot learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Sandler, M., Howard, A. G., Zhu, M., Zhmoginov, A., and Chen, L. (2018). Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *CoRR*, abs/1801.04381.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., and Hassabis, D. (2017). Mastering the game of go without human knowledge. *Nature*, 550:354–.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

Sun, S.-H. (2019). Multi-digit mnist for few-shot learning.

Xie, L., Chen, X., Bi, K., Wei, L., Xu, Y., Chen, Z., Wang, L., Xiao, A., Chang, J., Zhang, X., and Tian, Q. (2020). Weight-sharing neural architecture search: A battle to shrink the optimization gap. *CoRR*, abs/2008.01475.

Xu, Y., Xie, L., Zhang, X., Chen, X., Qi, G.-J., Tian, Q., and Xiong, H. (2019). PC-DARTS: Partial Channel Connections for Memory-Efficient Architecture Search. *arXiv e-prints*, page arXiv:1907.05737.

Zela, A., Elsken, T., Saikia, T., Marrakchi, Y., Brox, T., and Hutter, F. (2019). Understanding and robustifying differentiable architecture search. *CoRR*, abs/1909.09656.

A Appendix A

A.1 Hyperparameters and Experiment Setup

A.1.1 Hyperparameters. Our implementation is based on the MetaNAS[2] (Elsken et al., 2019) code. The code for the regularization methods are taken from their original repositories: SharpDARTS[3] (Hundt et al., 2019), P-DARTS[4] (Chen et al., 2019), DARTS-[5] (Chu et al., 2021) and PC-DARTS[6]

[2]https://github.com/boschresearch/metanas

[3]https://github.com/ahundt/sharpDARTS

[4]https://github.com/chenxin061/pdarts

[5]https://github.com/Meituan-AutoML/DARTS-

[6]https://github.com/yuhuixu1993/PC-DARTS

(Xu et al., 2019). Finally, the code for the TSE-DARTS (Fil et al., 2021) optimization is provided in the appendix of the paper. The meta-testing accuracy of all experiments in section 4 are taken at the end of meta-training, without a dedicated meta-testing phase at the end of meta-training. Potentially, a better meta-testing performance could be obtained by running a final evaluation in which MetaNAS adapts during the meta-testing phase for more epochs.

The Hyperparameters are listed in Table 3. The same parameters are used for all experiments in section 4 with minor adjustments for the experiments of section 4.1 and 4.2. The parameters are taken from the original MetaNAS implementation (Elsken et al., 2019) except for the task training steps during meta-training. This parameters is increased from 5 to 6 steps to evaluate P-DARTS in 3 stages with 2 adaptation steps each. To evaluate P-DARTS in 3 stages as done in the P-DARTS paper. The dropout on the skip-connections are set to 0.0, 0.4, 0.7 for stages 1, 2 and 3 respectively. During the training the dropout is exponentially decayed with a factor 0.2. These hyperparameters are taken from the P-DARTS implementation.

All the experiments are executed on a cluster with the following GPUs: Tesla V100-DGXS-16GB, GeForce GTX 1080 Ti, GeForce GTX 970. Moreover, all experiments use a single GPU during training. The candidate set of operations per edge are the same as MetaNAS, namely, the *MaxPool3x3, AvgPool3x3, SkipConnect, Conv1x5-5x1, Conv3x3, SepConv3x3, DilatedConv3x3* operations. The meta-testing training steps are split into two phases, the first 50 adaptation steps update both the weights and architecture and the next 50 steps only adapt the weights. Finally, all experiments are evaluated for two random seeds, seed 1 and 2.

Hyperparameter	Value
batch size	20
meta batch size	10
shots during meta training	15
Number of channels	28
Number of DARTS cells	3
task training steps (during meta training)	6
task training steps (during meta testing)	50 + 50
task learning rate (weights)	$1 \cdot 10^{-3}$
task learning rate (architecture)	$1 \cdot 10^{-3}$
task optimizer (weights)	Adam
task optimizer (architecture)	Adam
meta learning rate (weights)	1.0
meta learning rate (architecture)	0.6
meta optimizer (weights)	SGD
meta optimizer (architecture)	SGD
weight decay (weights)	0.0
weight decay (architecture)	$1 \cdot 10^{-3}$
Stages (P-DARTS)	3
limit number of skip-connections (P-DARTS)	2

Table 3: The hyperparameters of the experiments in section 4 on the Omniglot, MiniImageNet and TripleMNIST datasets for n-shot, k-way few-shot learning.

A.1.2 Experiment Setup.

The experiments in section 4 are executed for 500 meta-epochs of which 250 warm-up epochs. During the warm-up epochs the architecture parameters of the model are not adjusted to warm-up the model. This practice avoids the unstable behavior of gradient-based NAS (Elsken et al., 2019). Similarly, to MetaNAS half of the meta-epochs are warm-up epochs. All experiments for 500 epochs require approximately 15 hours, except for MetaNAS with TSE-DARTS which requires approximately 19 hours.

A.2 Selection of the Regularization Methods

In this section we provide an analysis of all introduced regularization methods to select the best performing methods. The first segment of Table 4 contains the baseline MetaNAS result with an accuracy of 78.20% ± 1.18.

		Omniglot
Method	Parameters	Acuracy in 1-shot, 20-way
MetaNAS	340k	78.20 ±1.18
MetaNAS+P-DARTS	250k	57.32 ±0.08
MetaNAS+Search Space Regularization (Limit and Dropout Skip-Connections)	212k	58.04 ±1.92
MetaNAS+Search Space Approximation	251k	54.66 ±1.66
MetaNAS+Limit and Dropout Skip-Connections	**335k**	**80.86 ± 1.42**
MetaNAS+Dropout Skip-Connections	**303k**	**81.24 ± 0.20**
MetaNAS+Limit Skip-Connections	352k	77.16 ±0.24
MetaNAS+SharpDARTS	128k	74.40 ±0.32
MetaNAS+max-W+Power Cos Annealing	**98k**	**80.86 ± 0.54**
MetaNAS+max-W+Sharp Search Space	114k	78.38 ±0.18
MetaNAS+Power Cos Annealing+Sharp Space	377k	77.48 ±3.60
MetaNAS+max-W	101k	79.98 ±0.54
MetaNAS+Sharp Search Space	162k	67.74 ±1.94
MetaNAS+Power Cos Annealing	305k	76.32 ±1.04
MetaNAS+DARTS-	265k	74.56 ±1.58
MetaNAS+PC-DARTS-	44k	68.00 ±0.44
MetaNAS+DARTS- fixed $\beta = 0.7$	**240k**	**81.70 ± 0.42**
MetaNAS+DARTS- fixed $\beta = 0.7$ and decay meta-test	231k	78.52 ±2.92

Table 4: The results of the regularization methods on the Omniglot dataset in a 1-shot, 20-way setting with the final meta-testing accuracy. The accuracy is averaged over two runs with a ±1 standard deviation after 500 meta-epochs with 250 warm-up epochs.

The second split includes regularized P-DARTS methods, of which the best results are obtained by limiting and dropout. The difference between Search Space Regularization and limiting and dropout on the skip-connections is the removal of the progressive P-DARTS staging in the latter. This simple modification of DARTS already shows promising results in as few as 500 meta-epochs improving over the baseline by 3.04%.

The third segment evaluates the SharpDARTS components. The best improvement over MetaNAS is obtained by combining max-W alpha regularization and power cosine annealing with an accuracy of 80.86% ± 0.54 and 98k parameters. This adjustment has ±3× fewer parameters and achieves higher accuracy than the baseline. However, the architecture might converge prematurely as the architecture parameters do change much after 100 meta-epochs.

The fourth section contains the DARTS- adjustments. The DARTS- method obtains the best performance by fixing parameter β to 0.7 during meta-testing and training (Chu et al., 2021), which obtains an accuracy of 81.70% ± 0.40. However, adding a skip-connection to every edge without decay introduces an operation kept in the final architecture and thus undesirable as we are introducing heuristics and not leaving the search up to the search strategy. Our attempts at decaying β during meta-testing lead to worse results. Therefore, we decided to further evaluate the best methods of segments two and three.

A.3 Hyperparameter Selection for TSE-DARTS

As TSE-DARTS changes the optimization of DARTS and adds a new hyperparameter, number of unrolling steps, we perform a manual hyperparameter selection. We lower the number of unrolling steps T to $T = 1$ and $T = 2$ compared to $T = 100$ due to few-shot learning setting. Additionally, we evaluate TSE-DARTS with higher learning rate due to the high number of model parameters with the original learning rate. We want to lower the number of parameters to provide a more fair comparison with MetaNAS.

	Omniglot	
Method	parameters	accuracy in 1-shot, 20-way
MetaNAS	340k	78.20 ±1.18
MetaNAS+TSE-DARTS+$T = 1$	381k	89.66 ±0.22
MetaNAS+TSE-DARTS+$T = 2$	391k	90.80 ±0.14
MetaNAS+TSE-DARTS+$T = 1$+DARTS lr weights and architecture of $4 \cdot 10^{-2}$	251k	82.38 ±0.90

Table 5: The TSE-DARTS hyperparameter selection experiments on the 1-shot, 20-way Omniglot dataset with final meta-testing accuracy. The accuracy is averaged over two runs with a ±1 standard deviation after 500 meta-epochs with 250 warm-up epochs.

Table 5 displays the results of our hyperparameter selection. For $T = 1$ we observe a significant improvement in accuracy compared to the baseline improving the baseline by 11.46 and observe a slightly better improvement for $T = 2$. However, using $T = 2$ also increases the running time from approximately 18 hours to 24,5 hours. Therefore, we use $T = 1$ for our TSE-DARTS experiments to maintain comparable running times to the baseline MetaNAS. We also evaluate $T = 1$ with a higher DARTS learning rate for architecture and weights due to the larger amount of parameters, by increasing the learning rate we lower the model parameters.